UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO

Lucas Domingues Zófoli

Chocadeira Automatizada utilizando Arduino com Monitoramento Remoto

Santa Maria, RS
2022

Lucas Domingues Zófoli

Chocadeira Automatizada utilizando Arduino com Monitoramento Remoto

Trabalho de graduação apresentado no Curso de Engenharia de computação da Universidade Federal de Santa Maria (UFSM, RS) como requisito parcial para obtenção do grau de **Bacharel em** Engenharia de Computação**.**

Orientador: Prof Dr. José Eduardo Baggio

Santa Maria, RS
2022

‘

Lucas Domingues Zófoli

Chocadeira Automatizada utilizando Arduino com Monitoramento Remoto

Trabalho de graduação apresentado no Curso de Engenharia de computação da Universidade Federal de Santa Maria (UFSM, RS) como requisito parcial para obtenção do grau de **Bacharel em** Engenharia de Computação**.**

Aprovado em __ de _____ de 2022:

_____
José Eduardo Baggio (UFSM)
(Presidente/ Orientador)

Santa Maria, RS
2022

AGRADECIMENTOS

À minha família e amigos pelo apoio e incentivo durante minha trajetória no curso.

Ao professor e orientador José Eduardo Baggio pela ajuda e orientação durante o projeto e também no curso de Engenharia de Computação.

Aos professores pelo conhecimento transmitido durante o curso.

Aos colegas Felipe Beling e Marcos Cargnin pelo auxílio e empréstimo de equipamentos para a realização do trabalho de conclusão de curso.

À Nathália Almeida pela paciência, amor, apoio e incentivo durante o projeto.

À todos que contribuíram de forma direta ou indiretamente para a conclusão do projeto e graduação no curso de Engenharia de Computação.

RESUMO

Chocadeira Automatizada utilizando Arduino com Monitoramento Remoto

AUTOR: LUCAS DOMINGUES ZÓFOLI
ORIENTADOR: JOSÉ EDUARDO BAGGIO

Para os criadores de aves, chocar ovos em grande escala é um problema para a produção de aves. A incubação natural através do choco da ave limita fisicamente o processo e não permite ao produtor gerenciar os parâmetros críticos para garantir o sucesso na reprodução. Este projeto propõe o desenvolvimento de uma chocadeira inteligente de baixo custo e de simples operação, que permite reduzir o risco e aumentar a taxa de sucesso no processo de reprodução, através da automatização, além de permitir o aumento de escala. O sistema de controle é baseado na plataforma Arduino, que é programado de forma a controlar a temperatura e umidade da chocadeira, e também permite girar os ovos automaticamente. Adicionalmente, com a implementação do conceito de Internet das Coisas (*IoT*) no projeto, é possível ajudar os criadores a monitorar a chocadeira a distância.

Palavras Chave: Chocadeira Automatizada. Microcontrolador. Arduino. C++. Incubação Ovos. Internet das Coisas. Monitoramento Remoto

ABSTRACT

REMOTE MONITORED SMART EGG INCUBATOR USING ARDUINO PLATFORM

Hatching eggs on a large scale is a major problem for the chicken producers. The incubation process through the natural process physically limits the production, and does not allow the producer to manage the critical parameters to ensure the success of the reproduction. This project proposes the development of a simple and low cost smart egg incubator system, which through automatized processes will reduce the risk of failure and will improve the production and success rate of the entire process of hatching chicken eggs. The control system is based on the Arduino platform, which is programmed to control the temperature and humidity of the egg hatcher, and automatically rotate the eggs. Additionally, with the implementation of the Internet of Things (IoT) concept on the project, it's possible to help the producers remote monitor the egg incubator.

Keywords: Chicken Eggs Smart Incubator. Microcontroller. Arduino. C++. Hatching chicken eggs. IoT. Remote Monitoring

‘

LISTA DE ILUSTRAÇÕES

‘

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A – Ampére

API - Interface de Programação de Aplicação

A/D - Analógico/Digital

CA - Corrente Alternada

CC - Corrente Contínua

DOS - Sistema operacional em disco

EEPROM - *Electrically-Erasable Programmable Read-Only Memory*

FTP – *File Transfer Protocol*

GND - *Ground* (Terra elétrico)

HTTP - Protocolo de Transferência de Hipertexto

I/O - Input/Output

IDE - *Integrated Design Environment*

IoT – Internet of Things

IP - Internet Protocol

JSON - *JavaScript Object Notation*

LCD - *Display* de Cristal Líquido

LED - Diodo Emissor de Luz

mA – Miliampere

MIT - Massachusetts Institute of Technology

mm - Milímetro

ms - Milissegundos

mV – Milivolt

NTC - Termístor Coeficiente Negativo

PHP - *Hypertext Preprocessor*

PWM - Pulse Width Modulation

RISC - Reduced Instruction Set Computing

RTC - Relógio em Tempo-Real

SGBD - Sistema de Gerenciamento de Banco de Dados

SRAM - *Static Random Access Memory*

SQL - Linguagem de Consulta Estruturada

SPI - Serial Peripheral Interface

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

URL - *Uniform Resource Locator*

USART - Universal Synchronous Asynchronous Receiver Transmitter

USB - Universal Serial Bus

V – Volt

VAC - Tensão em corrente alternada

VCC - Tensão em corrente contínua.

LISTA DE ANEXOS

‘

SUMÁRIO

1 INTRODUÇÃO

A avicultura representa grande importância na agricultura familiar tanto na questão de segurança alimentar para a família quanto no aspecto econômico. Além de fornecer esterco aos cultivos, a avicultura aproveita os restos de plantios e dos refugos de frutas e hortaliças (AMANDO, 2018). A criação doméstica pode proporcionar uma fonte de proteína (carne e ovos) saudável e de baixo custo. Porém, o criador não industrial de ave pode enfrentar certas dificuldades em controlar o ciclo completo de reprodução por fatores externos, como por exemplo, infecções, posição incorreta dos ovos, estresse do animal e fatores climáticos que podem vir a alterar as condições ideais de incubação, o que pode resultar numa baixa taxa de sucesso (COBB VANTRESS, 2008). O uso da chocadeira artificial é um importante fator de aumento de produção permitindo um aumento ao redor de 30% de pintos/galinha ao ano (EMBRAPA, 2007).

Para solucionar esse problema, foi elaborado um protótipo de chocadeira, um equipamento de menor custo, em relação aos produtos já encontrados no mercado nacional, que tem como função incubar os ovos permitindo completar o seu ciclo de chocagem. O projeto em questão propõe a comunicação da chocadeira protótipo com um servidor por meio de uma interface de simples acesso, para fins de monitoramento preciso das condições dos ovos na chocadeira, em tempo real. O objetivo é desenvolver um equipamento útil e confiável para médios e pequenos criadores de aves reduzindo ou até mesmo eliminando a necessidade de um plantel de aves chocadeiras para a reprodução.

1.1 MOTIVAÇÃO

As maiores motivações para a realização desse projeto são:

- Estudar e disponibilizar um projeto envolvendo conceitos estudados durante o curso de Engenharia de Computação, como física, programação, eletrônica e sistemas embarcados, direcionado a uma aplicação prática.

- Estudar o conceito de IoT e utilizá-lo para a solução de um problema real e prático.

1.2 OBJETIVOS GERAIS E ESPECÍFICOS

Os principais objetivos do trabalho proposto são:

- Tornar o processo artificial de chocar ovos o mais automático possível para que o usuário se preocupe o mínimo com a manutenção das funções dos dispositivos e, no final do processo, obtenha os animais vivos e saudáveis para seus devidos fins.

- Criar uma comunicação remota entre a chocadeira e um servidor remoto, para fins de monitoramento das condições da chocadeira.

- Obter um resultado que proporcione a eficiência no sistema de incubação, produtividade, comodidade ao produtor oferecendo um produto de baixo custo e operacionalmente intuitivo.

1.3 ESTRUTURA DO TRABALHO

Nesta seção serão apresentados e comentados os capítulos deste trabalho.

O capítulo 2 apresenta informações sobre os assuntos que foram estudados e pesquisados em diversas fontes e os quais ofereceram conhecimento para que a implementação do projeto fosse realizada.

O capítulo 3 trata dos componentes eletrônicos utilizados e procedimentos efetuados para o desenvolvimento da chocadeira, banco de dados e aplicativo para o monitoramento remoto.

No capítulo 4 são mostradas as etapas que foram realizadas para a montagem do protótipo. Neste capítulo também são apresentados e discutidos os resultados obtidos.

O capítulo 5 apresenta as conclusões, ideias de melhorias para projetos futuros e as referências.

Por fim, em anexo, são apresentados os códigos desenvolvidos no projeto.

2 FUNDAMENTAÇÃO TEÓRICA E REVISÃO DE LITERATURA

Neste capítulo são apresentadas as informações adquiridas através de pesquisas em diversas fontes, bem como os conhecimentos obtidos durante o curso de Engenharia de Computação os quais tornaram possível a implementação deste trabalho. Primeiramente é comentado sobre conceitos e funcionamento dos componentes da chocadeira, em seguida são discutidas informações sobre os dispositivos e componentes utilizados no presente trabalho.

2.1 FUNCIONAMENTO DE UMA CHOCADEIRA

Uma chocadeira automatizada é um produto já presente no mercado, porém é muitas vezes inviável para alguns produtores menores. Em uma pesquisa de mercado nacional, encontram-se chocadeiras simples a partir da faixa de 500 reais. Existem vários modelos comerciais com capacidade variável (de 35 a 1.000 ovos ou mais). Podem ser completamente automáticas ou semiautomáticas. No caso de chocadeiras semiautomáticas, a viragem dos ovos e o controle de umidade são realizados manualmente. A principal questão na prática da incubação artificial é o dispêndio do capital necessário à obtenção das chocadeiras. Os preços variam de acordo com o número de ovos, fabricante e nível de automatização.

Na literatura encontram-se vários textos que onde ressaltam a importância da tecnologia na avicultura. O fator determinante para o sucesso da tecnologia na avicultura passa pelo sucesso na adaptabilidade do produtor, que não raras vezes demonstra resistência, por falta de conhecimento (CONOLLY, 2018).

Dentre as principais condições da incubação que afetam a taxa de eclosão e qualidade dos ovos, se destacam a temperatura, umidade, ventilação e a rotação dos ovos (HARB, 2010). Dentre estas condições, a temperatura é a mais importante e que pode ter o principal efeito no sucesso da taxa de eclosão.

Um embrião em desenvolvimento está exposto a uma temperatura que depende da temperatura da chocadeira, do calor da produção embrionária e da condutividade térmica do ovo (N.A.FRENCH, 1997). A temperatura varia de acordo com o tipo de ovo que será incubado, mas para a grande maioria das espécies, a temperatura varia entre 37ºc e 38ºc. Um valor fora deste espectro seria fatal para a

formação e desenvolvimento do embrião incubado (YILMAZ, TEPELI, GARIP, ÇAĞLAYAN, 2011).

O período de incubação é de 17 a 19 dias (N.A.FRENCH, 1997), a umidade ambiente deve ser em torno de 50% a 65%, e os ovos devem ser rotacionados em no mínimo 45º e de 4 em 4 horas, para prevenir o embrião de grudar no interior da casca do ovo (N.A.FRENCH, 1997). Com o ovo na temperatura certa, o processo biológico da incubação do embrião começa a se desenvolver, logo a temperatura do sistema deve ser estável durante todo período de incubação para alcançar uma maior taxa de sucesso na eclosão(THE INCUBATOR SHOP, 2016).

Estudos feitos mostram que um acréscimo ou decréscimo das temperaturas externas também afetam a mortalidade dos embriões, logo é importante projetar uma chocadeira com um bom isolamento térmico, que de fato mantenha a temperatura estável na faixa desejada (N.A.FRENCH, 1997).

Para se obter o resultado desejado, com manutenção da temperatura, rotação dos ovos e ventilação, a seguir são detalhados os principais componentes empregados em chocadeiras automatizadas.

2.2 COMPONENTES DE UMA CHOCADEIRA AUTOMATIZADA

Quando o desenvolvimento do embrião depende de máquinas e equipamentos, define-se como processo de incubação artificial. Nesse processo, o desenvolvimento do embrião nos ovos independe da galinha adulta, pois as chocadeiras têm ambiente controlado. O uso da chocadeira é importante para aumentar a produção, mas, em função do seu custo e da operacionalização, cabe avaliar a viabilidade da sua aquisição e utilização conforme o perfil do produtor, objetivando melhor retorno econômico sem ultrapassar o seu orçamento.

Na incubação artificial, um equipamento é utilizado para manter a temperatura dos ovos até que eles eclodam. A maioria dessas máquinas é elétrica e provida de um termostato que desliga automaticamente o aquecimento, quando a temperatura atinge determinado limite, e volta a ligar quando a temperatura decresce, mantendo assim a temperatura constante dentro de um limite desejável (Mora, 2008). A figura 1,

mostra um exemplo de uma chocadeira industrial encontrada no mercado nacional (ECLOPINTO, 2021).

Figura 1 – Chocadeira automatizada encontrada no mercado nacional

Fonte: https://www.eclopintochocadeiras.com/MLB-1376500680-chocadeira-eclopinto-120-ovos-automatica-com-ovoscopio-_JM?utm_source=google&utm_medium=cpc&utm_campaign=darwin_ss

Uma chocadeira padrão deste modelo ou faixa de preço semelhante, dentre os componentes básicos possui:

- Uma fonte de calor para aquecer os ovos incubados (lâmpada);
- Um sistema de arrefecimento para controle da umidade (ventoinha);
- Um motor atuador, caso a rotação dos ovos seja feita de forma automatizada;
- Sensores de monitoramento das condições de ambiente interno;
- Uma interface para interação com o usuário.

No presente projeto, será implementado um modelo de chocadeira *smart*, com o intuito de replicar e melhorar o conceito já existente de uma chocadeira comum, adicionando algumas possibilidades ao usuário tais como o manejo da temperatura e umidade (possibilidade de chocar ovos de espécies diferentes) e conexão em tempo real com um servidor através de um aplicativo Android de simples manejo, para fins

de monitoramento das condições de chocagem. Abaixo segue a explicação de alguns conceitos e softwares utilizados durante o processo de implementação da chocadeira automatizada.

2.2.1 Microcontrolador

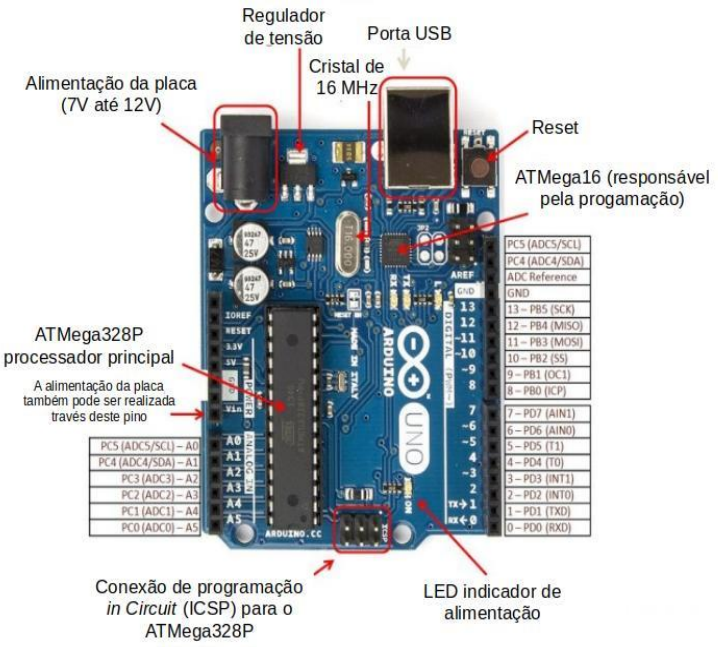
Um microcontrolador é um dispositivo presente em inúmeros aparelhos eletrônicos do dia-a-dia, como celulares, micro ondas, geladeiras, dentre outros. Ele é constituído de um processador, memória de programa, memória de dados, pinos de I/O (Input/Output) e mais alguns periféricos como *timers*, conversores A/D (analógico/digitais), dentre outros, tudo em apenas um encapsulamento, tornando assim menor a complexidade do circuito eletrônico que seria utilizado para uma determinada aplicação de controle. Por esse motivo, os microcontroladores são ótimas ferramentas para a implementação de projetos de sistemas embarcados, onde há necessidade de um controle específico e regulado.

2.2.2 Arduino

Arduino é uma plataforma de código aberto utilizada para construção de projetos eletrônicos. O Arduino consiste de uma placa de circuito programável (também chamado de microcontrolador), e o IDE (Ambiente de Desenvolvimento Integrado) (RAUBER, 2019). Na figura 2 é possível observar com maiores detalhes a placa do Arduino Uno, que fora utilizada no presente projeto.

O Arduino Uno é uma placa versátil utilizada mundialmente em vários projetos simples ou complexos. Conta com uma rica documentação complementar proveniente de Shields, Módulos e Sensores. O IDE utiliza a linguagem C/C++, é programado pelo usuário, e carrega o código computacional para a placa de circuito programável, para assim controlar os pinos do microcontrolador. O IDE está disponível gratuitamente para download, através do site do fornecedor (ARDUINO,2020). Na figura 3 é possível observar com maiores detalhes o IDE da plataforma Arduino

Figura 2 - Placa Arduino Uno



Fonte: https://visiorob.com.br/index.php/2019/09/15/introducao-ao-microcontrolador-atmega328p

Figura 3 - IDE Arduino

Fonte: Próprio autor

As grandes vantagens de se utilizar a plataforma Arduino são:

- Ser uma ferramenta *open-source* (Software/Hardware);
- Baixo custo de prototipagem;
- Não tem a necessidade de operação em conjunto com um computador (*standalone*).
- Grande número de tutoriais, artigos e projetos na internet;
- Extensa comunidade de desenvolvedores e hobbystas;

O Arduino, principalmente dos tipos, Uno e Nano, contam com um microcontrolador chamado ATmega328P, que pode ser classificado como o processador central da placa, ele é pertencente à família AVR lançada pela Atmel.

Ele conta com 23 entradas/saídas programáveis, sendo capaz de executar programas dos mais simples, como por exemplo piscar um LED, até programas complexos, como sistemas autônomos e robôs. Além disso, este microcontrolador possui a capacidade de operar com baixo consumo de potência.

Este modelo de microcontrolador possui um processador RISC (*Reduced Instruction Set Computing*) de 8 bits que utiliza uma arquitetura Harvard modificada, possui 32KB de memória flash, 1KB de EEPROM(*Electrically-Erasable Programmable Read-Only Memory*), 2KB de SRAM (*Static Random Access Memory*), 32 registradores, três *timers* (temporizadores), uma USART (*Universal Synchronous Asynchronous Receiver Transmitter*), portas para comunicação SPI (*Serial Peripheral Interface*), um conversor AD de 10 bits, saídas PWM (*Pulse-Width Modulation*) dentre outros (ATMEGA328 DATASHEET, 2009). Na tabela 1, são mostrados alguns dados do microcontrolador fornecidos pelo fabricante.

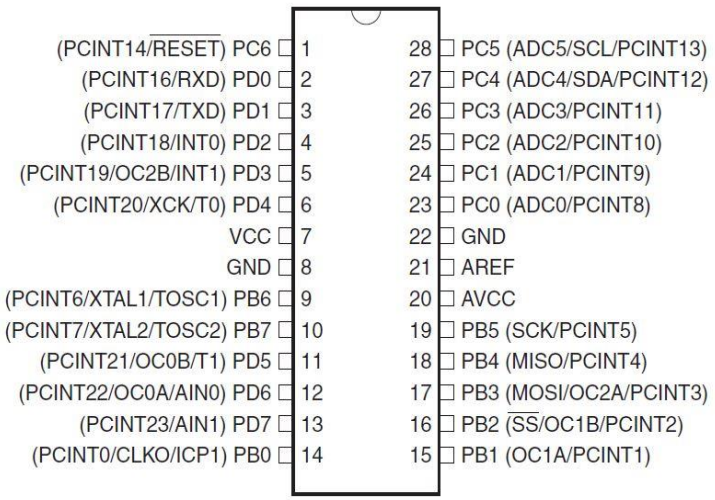
Tabela 1 - Características Placa Arduino Uno

Microcontrolador	ATmega328
Tensão de operação	5V
Tensão de entrada (recomendado)	7-12V
Pinos digitais de I/O	14 (6 com saída PWM)
Pinos de entrada analógica	6
Corrente por pino de I/O	40 mA
Corrente pelo pino de 3,3V	50 mA
Memória Flash	32KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1KB (ATmega328)
Velocidade do Clock	16 MHz

Fonte: https://visiorob.com.br/index.php/2019/09/15/introducao-ao-microcontrolador-atmega328p/

A **pinagem** do microcontrolador descreve as conexões que ele possui a fim de interagir com estímulos externos, tanto recebendo sinais como emitindo sinais. A Figura 4 apresenta uma imagem ilustrativa do ATmega328P e seus pinos:

Figura 4 - ATmega328P



Fonte: https://visiorob.com.br/index.php/2019/09/15/introducao-ao-microcontrolador-atmega328p/

Entendendo cada pino:

VCC: Alimentação do microcontrolador, aceita valores de 2,7V a 5,5V.

GND: Ground.

AREF: Pino de referência de tensão máxima para o conversor de sinais analógico para digital (ADC).

AVCC: Pino de alimentação do conversor ADC deste microcontrolador, deve ser ligado ao VCC.

PORTB: Porta bidirecional de 8 bits. Os pinos correspondentes são o PB0 (14), PB1 (15), PB2 (16), PB3 (17), PB4 (18), PB5(19), PB6 (9) e PB7(10).

PORTC: Porta bidirecional de 7 bits. Os pinos correspondentes são o PC0 (23), PC1 (24), PC2 (25), PC3 (26), PC4 (27), PC5 (28) e PC6 (1).

PORTD: Porta bidirecional de 8 bits. Os pinos correspondentes são o PD0 (2), PD1 (3), PD2 (4), PD3 (5), PD4 (6), PD5 (11), PD6 (12) e PD7 (13).

Os **pinos digitais** podem ser configurados como entrada ou saída e possuem somente dois estados: ligado (*HIGH*) ou desligado (*LOW*). Quando o pino é configurado como ligado ele apresenta uma tensão de saída de 5V e quando configurado como desligado apresenta na saída a tensão de 0V. Esses pinos podem ser utilizados para várias aplicações como acendimento de *leds*, operações com portas lógicas, acionamento de relés, dentre outros.

Os **pinos analógicos** são utilizados na maioria das vezes para leituras de sensores e transdutores que convertem grandezas físicas em tensão elétrica, a qual pode ser medida pela entrada analógica no microcontrolador.

Além dos pinos analógicos e digitais, o ATmega328 possui uma quantidade de pinos com características especiais que podem ser configuradas através da programação. São eles:

- **Interrupção externa**: Existe a possibilidade de se programar um pino para que quando ativado force o processador a parar o que está fazendo e realizar outras operações pré-programadas.

O ATmega328 possui 2 pinos (2 e 3) para interrupções externas. São bastante úteis para economizar processamento em diversas aplicações.

- **PWM**: é gerado um sinal pulsante, com uma determinada frequência, onde se define o tempo em que o sinal fica em nível alto e o tempo que o sinal fica em nível baixo. Desta forma, variando-se a razão cíclica, que é a razão entre tempo em nível

alto e o período do sinal, pode-se controlar o valor médio do sinal PWM. Os pinos que possuem saída PWM são: pinos 3,5,6,10 e 11.

- **Portal Serial USART**: possibilita o microcontrolador se comunicar com um computador, um módulo de *bluetooth* ou outro dispositivo do gênero, através de envio e recebimento de dados no formato serial assíncrono (USART). Os pinos que possuem comunicação serial são: pino 0 (recebe dados) e pino 1 (envia dados).

A **comunicação serial** trata-se de um envio de forma sequencial de bits, por um barramento, em um certo intervalo de tempo, ou seja, de forma sequencial. Esse meio de comunicação é utilizado em diversos dispositivos como a USB (*Universal Serial Bus*), FireWire, RS-232 dentre outros.

O Arduino UNO possui uma porta de comunicação nos pinos digitais 0 (recebimento de sinais digitais) e 1 (Envio de sinais digitais). O IDE do Arduino oferece uma aplicação bastante interessante e útil chamada de *Serial Monitor* que quando utilizada mostra na tela os valores das portas digitais e analógicas.

Um **conversor analógico/digital** tem o objetivo de converter o sinal analógico de sensores e outros dispositivos, para que o Microcontrolador possa processá-lo posteriormente. Este circuito é útil para diversos tipos de aplicações, como aquisição de dados de sensores. O conversor A/D do microcontrolador ATmega328 possui 10 bits de resolução, a sua tensão de entrada pode variar de 0 V até o valor de VCC e possui referência interna selecionável de 1,1 V. Dessa forma quando está trabalhando com a referência em VCC o menor valor que pode ser lido será:

$$\text{Resolução} = (5V / 1024) = 4{,}88\ mV$$

Esse é o valor de degrau para uma conversão em 10 bits com referência em 5 V.

2.3 SOFTWARES E CONCEITOS APLICADOS

Neste projeto serão utilizadas algumas ferramentas que facilitam o desenvolvimento do software para o funcionamento do projeto proposto. Serão utilizadas apenas ferramentas gratuitas, mas que atendem perfeitamente à

necessidade de controle das variáveis de monitoramento, criação de uma API, banco de dados e um aplicativo para Android.

2.3.1 Android

Criado em 2008, o Android é um sistema operacional criado por 47 empresas lideradas pelo Google, dentre elas estão LG, Samsung, HTC, Motorola (hoje pertencente ao Google), Sony Ericsson e a Intel. Ele, primeiramente, foi criado para celulares como um sistema aberto, sem que os fabricantes tivessem que pagar licença de uso. (GUIMARÃES, 2013)

O funcionamento do Android é similar a outros sistemas operacionais (como Windows, Mac OS, Ubuntu, entre outros), cuja função é gerenciar todos os processos dos aplicativos e do hardware de um computador para que funcionem perfeitamente. (UOL, 2015)

2.3.2 Internet das Coisas (IoT)

Segundo Zambarda (2014), a Internet das Coisas se refere a uma revolução tecnológica que tem como objetivo conectar os itens usados do dia a dia à rede mundial de computadores. Dispositivos tidos como comuns poucos anos atrás já estão sendo modificados para atender esta nova realidade, como no caso deste projeto, em que a chocadeira se comunica em tempo real com o dispositivo móvel do usuário para fins de monitoramento.

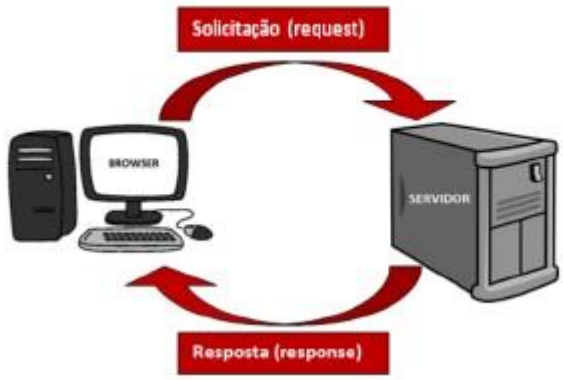
Na última década, houve um grande aumento na demanda de serviços de coleta e troca de dados através da internet de uma maneira eficiente. Neste sentido, a Internet das Coisas (IoT) fornece um armazenamento e troca de dados eficiente através da conexão de dispositivos físicos via sensores eletrônicos e a internet. (BUSINESS WIRE, 2019).

2.3.3 Protocolo HTTP

O protocolo HTTP (*HyperText Transfer Protocol*) fornece uma comunicação cliente-servidor da mesma forma que em uma comunicação interpessoal. Cada transação HTTP possui uma solicitação e uma resposta. O mesmo é a base onde a *Web* foi construída (MITCHELL; KINOSHITA, 2013).

Para cada comunicação o protocolo HTTP estabelece regras de procedimento, tanto para as requisições, como para as respostas. A requisição é composta por um cabeçalho que possui informações complementares, como a URL para a qual a solicitação foi direcionada, o método, outros cabeçalhos, códigos de status e também o corpo das respostas (MITCHELL; KINOSHITA, 2013).

Figura 5 - Funcionamento protocolo HTTP



Fonte: https:www.stackoverflow.com

2.3.4 API

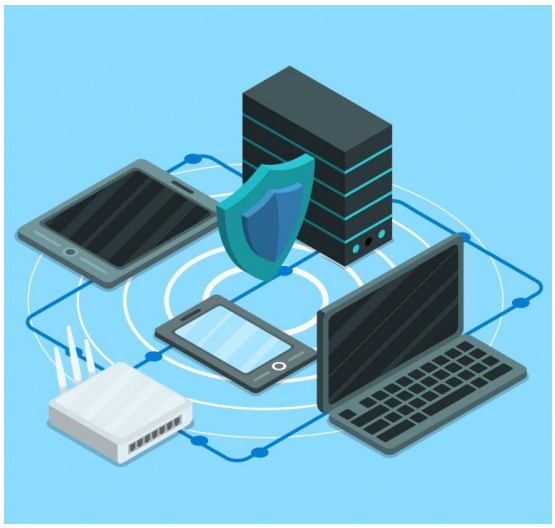
A sigla API corresponde às palavras em inglês “*Application Programming Interface*“, traduzindo para o português “Interface de Programação de Aplicações”. Elas são uma forma de integrar sistemas, possibilitando benefícios como a segurança

dos dados, facilidade no intercâmbio entre informações com diferentes linguagens de programação e a monetização de acessos.

As APIs são um tipo de “ponte” que conectam aplicações, podendo ser utilizadas para os mais variados tipos de negócio, por empresas de diversos nichos de mercado ou tamanho

As possibilidades disponibilizadas pelo uso das APIs proporcionam para os desenvolvedores de softwares e aplicativos a possibilidade de conectar tecnologias heterogêneas, como diferentes bancos de dados, por exemplo. Além disso, é possível fazer com que funcionalidades e ferramentas específicas de determinados aplicativos sejam utilizadas em outros (Fernandes, 2018). A figura 7, representa para que serve uma API.

Figura 6 - Funcionamento API



Fonte: https://vertigo.com.br/o-que-e-api-entenda-de-uma-maneira-simples/

2.3.5 MySQL

O MySQL é um sistema gerenciador de banco de dados relacional de código aberto usado na maioria das aplicações gratuitas para gerir suas bases de dados. O serviço utiliza a linguagem SQL, que é a linguagem mais popular para inserir, acessar e gerenciar o conteúdo armazenado num banco de dados.

Banco de dados é um conjunto organizado de dados, que fica disponível em um sistema informatizado e é controlado por um Sistema de Gerenciamento de Banco de Dados (SGBD), do inglês *Data Base Management System* (DBMS). O último é um software utilizado para gerenciar essa base de dados, permitindo criação, modificação, inserção/remoção de dados e eliminação de banco de dados (COSTA, 2011).

Na criação de aplicações web abertas e gratuitas, o conjunto de aplicações mais usado é o XAMPP, um acrônimo para Linux, Apache, MySQL e Perl/PHP/Python.

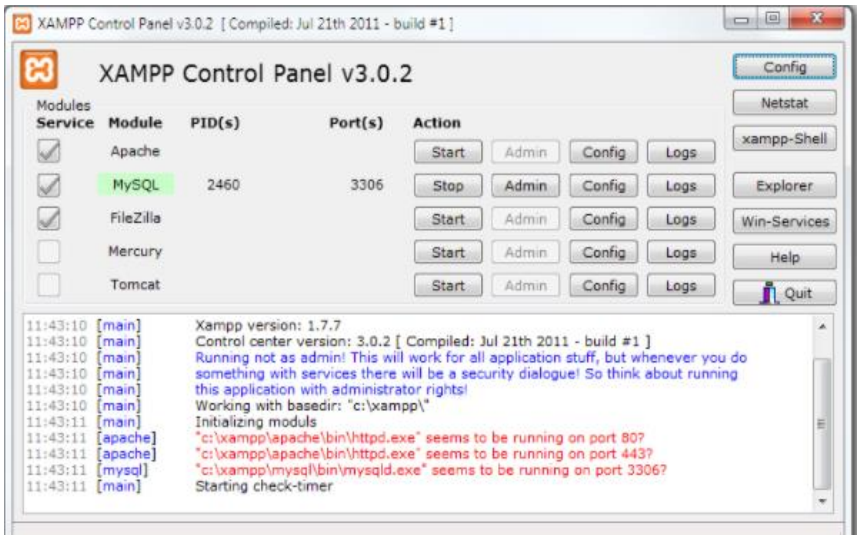
Nesse conjunto de aplicações, inclui-se, respectivamente, um sistema operacional, um servidor web, um sistema gerenciador de banco de dados e uma linguagem de programação. Assim, o MySQL é um dos componentes centrais da maioria das aplicações públicas da Internet (PISA, 2012).

2.3.6 XAMPP

Um servidor web, é um serviço capaz de responder às solicitações de um cliente utilizando-se de várias regras presentes em protocolos de comunicação, como o HTTP. O seu objetivo é oferecer, através de um computador robusto, uma série de serviços que ficam acessíveis aos clientes das mais diversas formas possíveis (DIDONÉ; CHAULET, 2016).

O XAMPP é um pacote com os principais servidores de código aberto do mercado, incluindo FTP, banco de dados MySQL e Apache com suporte às linguagens PHP e Perl. Atualmente, o XAMPP está disponível para quatro sistemas operacionais: Windows, Linux, Mac OS X e Solaris. Na figura 8, é mostrado o painel de controle do software.

Figura 7 - Painel de Controle XAMPP



Fonte: Próprio autor

O XAMPP foi desenvolvido para fornecer um servidor de testes sem complicações. Caso deseje que outra pessoa acesse seu site de testes pela Internet, basta fornecer o próprio número de IP. Para funcionar, é necessário desbloquear a porta 80 (HTTP) no firewall, tanto do sistema operacional quanto do roteador.

A velocidade de acesso dependerá apenas da capacidade de processamento do computador e da velocidade de upload da conexão. Por isso, é inviável o acesso por múltiplas pessoas numa conexão doméstica comum (HIGA, 2012).

2.3.7 MIT APP Inventor

O App Inventor é um ambiente de programação de fácil utilização na área de programação para dispositivos móveis. É intuitivo, possui um ambiente visual de programação que facilita a construção de aplicativos totalmente funcionais para *smartphones* e *tablets*, desde os mais simples, até os mais complexos. Por se tratar de uma ferramenta poderosa e de fácil acesso e entendimento, o App Inventor é amplamente utilizado, sendo possível criar aplicativos para diversas funções. A partir disto, portanto, vem se tornando simples a implementação de um sistema de automação residencial, utilizando apenas um *smartphone* e a comunicação serial ou ethernet.

É uma ferramenta de utilização online mantida pelo *Massachusetts Institute of Technology* (MIT) disponível no site http://ai2.appinventor.mit.edu/ e possui tradução para português (DESTACOM / UFMS, 2015).

Foi criado pelo Professor Hal Abelson e uma equipe do Google Educação em 2010 com o intuito de democratizar o desenvolvimento de software, empoderando todas as pessoas, especialmente as mais jovens, para sair de um simples consumidor, para ser um criador de tecnologia. O MIT APP Inventor, possui mais de 6 milhões de usuários registrados (MIT APP INVENTOR, 2014).

Para o usuário ter acesso ao ambiente do App Inventor é preciso entrar no site http://appinventor.mit.edu/explore/. É necessário, no entanto, ter uma conta no Gmail para salvar os projetos criados.

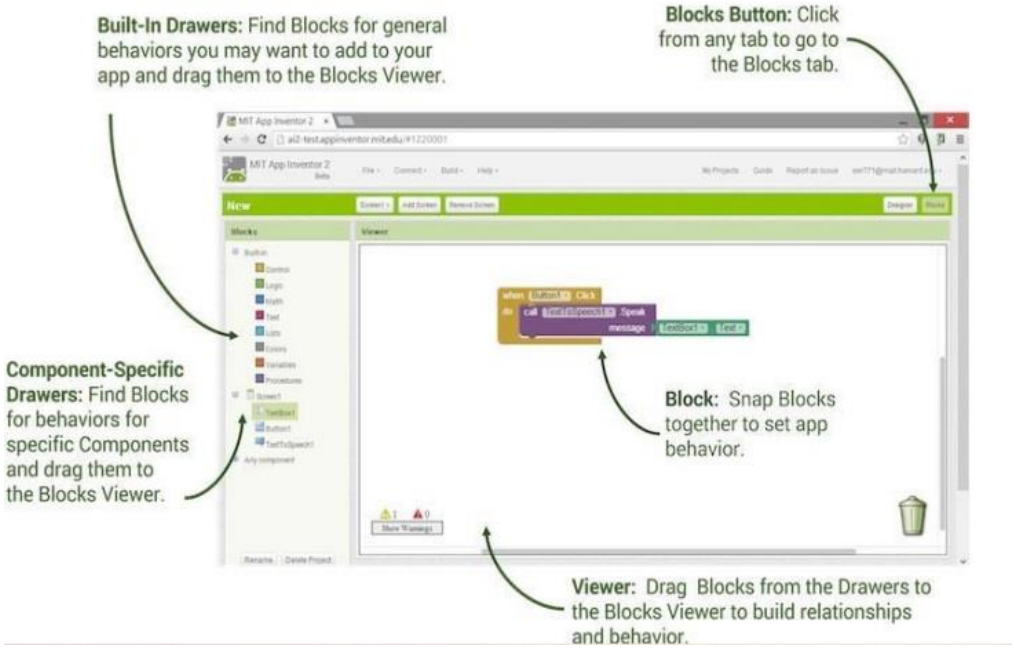
O App Inventor é uma ferramenta de construção de aplicativos Android através da junção de conjuntos de blocos que determinam como sua aplicação deve se comportar. O criador define a lógica de seu aplicativo arrastando blocos específicos. Cada componente em seu projeto tem seu próprio conjunto de blocos específicos para seus próprios eventos, métodos e propriedades.

O App Inventor consiste no *Designer*, que se destina à criação gráfica; e *Blocks*, destinado à programação em blocos. Na figura 8 observa-se as opções que os usuários têm como ferramentas para criação gráfica, bem como as preferências de cores, tamanhos e imagens. Na área *Blocks* realiza-se a programação do aplicativo, de forma que são inseridos os comandos que o aplicativo irá realizar. A programação é baseada na linguagem C. Assim, utiliza-se blocos de lógica, de modo que o criador do aplicativo escolhe a lógica que irá implementar em seu sistema, como componentes de controle, lógica, matemática, texto, variáveis entre outros. Além disso, é possível interagir com a lógica escolhida com os componentes montados na área *Designer*. Abaixo, a figura 9 mostra a área *Blocks.*

Figura 8 - Aba Design App Inventor

Fonte: http://appinventor.mit.edu

Figura 9 - Aba *Blocks* App Inventor



Fonte: http://appinventor.mit.edu

Estando apresentados os principais elementos utilizados neste projeto, a seguir iremos apresentar como tais elementos são organizados para a implementação e operacionalização da chocadeira proposta neste trabalho.

3 MATERIAIS E MÉTODOS

Os seguintes materiais foram utilizados na montagem do projeto:

- Uma lâmpada incandescente 220V/60W e seu respectivo encaixe;
- Sensor de temperatura e umidade DHT11;
- Microcontrolador ARDUINO UNO;
- Fonte de 12V, 3A;
- Servo Motor SG90;
- *Ethernet Shield* W5100;
- 2 Relés 220V, 50/60Hz, 10A;
- *Cooler* 12V;
- Resistores, Capacitores, Potenciômetro, *Protoboards*, botões interruptores;

3.1.1 Módulo Ethernet W5100

O *Ethernet Shield* permite que uma placa Arduino conecte-se à internet. É baseado no chip ethernet da WIZnet ethernet W5100 que fornece acesso à rede (IP) nos protocolos TCP ou UDP e é facilmente utilizado usando a biblioteca Ethernet Library e SD Library. Este shield suporta até quatro conexões de socket simultâneas.

É compatível com as versões do Arduino mais tradicionais, como Arduino Duemilanove, Arduino UNO (utilizado no projeto), Arduino MEGA, etc.

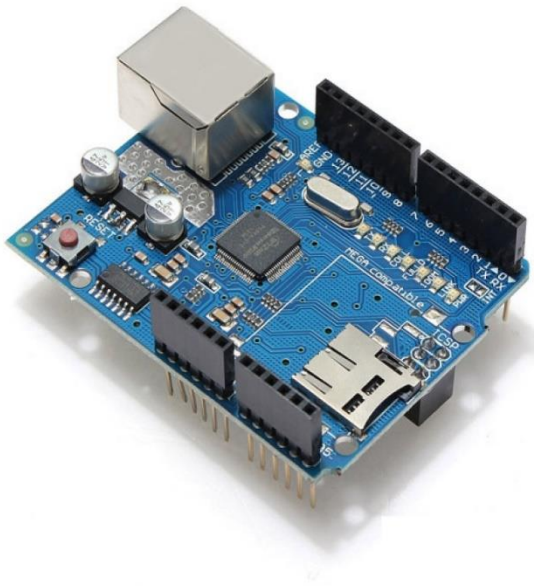
Possui um *slot* para cartão micro-SD, que pode ser usado para guardar e armazenar arquivos de um servidor na rede.

O Arduino se comunica com o W5100 e o cartão micro-SD usando o bus SPI (através do ICSP). Isto significa os pinos digitais 11, 12 e 13. O pino 10 é usado para selecionar o W5100 e o pino 4 para o cartão micro-SD. Portanto, estes pinos não

podem ser usados como entradas/saídas gerais. Este *shield* possui um conector RJ45 padrão.

O *Ethernet Shield* é imprescindível para a aplicação do conceito de IoT no projeto de uma chocadeira automatizada com monitoramento remoto.

Figura 10 - *Ethernet Shield*



Fonte: www-eletrodex-com-br/ethernet-shield-w5100-para-arduino-uno-e-mega-html

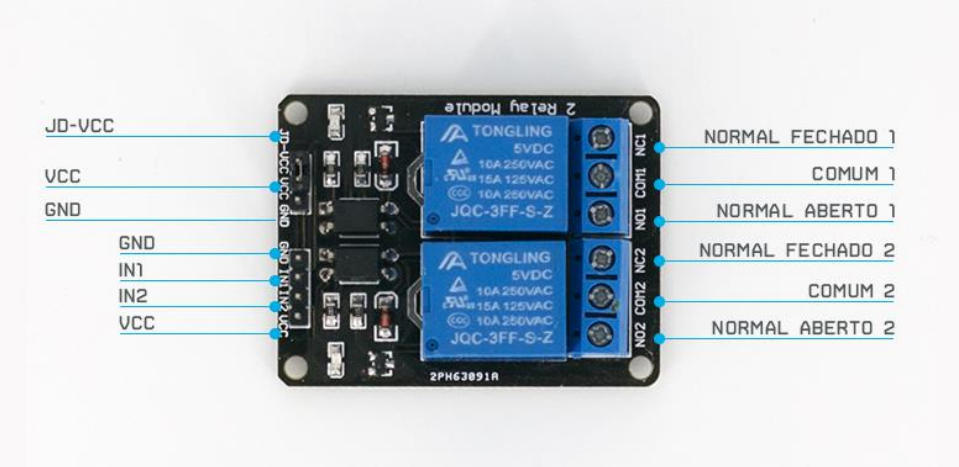
3.1.2 Módulo relé 12V 10A 2 canais

O Módulo relé 12V 10A 2 Canais permite o acionamento de cargas de corrente alternadas de 125VAC à 250VAC em até 10A, ou corrente contínua de 28VCC à 30VCC em até 10A.

O relé funciona como um interruptor de lâmpada, mas com a possibilidade de ser acionado remotamente através de microcontroladores, com ele é possível ligar ou desligar praticamente qualquer eletrodoméstico, ou aparelho ligado à energia, desde que respeite os limites de operação do relé.

Ideal para projetos de automação, o Módulo Relé 12V 10A pode ser utilizado para controle de iluminação, eletrodomésticos, motores CA ou CC, e aparelhos elétricos diversos (THOMSEN, 2013).

Figura 11 - Módulo Relé 12V



Fonte: https://www.filipeflop.com/blog/controle-modulo-rele-arduino/

Especificações do módulo relé utilizado:

- Tensão de operação: 5V CC
- Permite controlar cargas de 220V CA
- Nível de sinal dos pinos IN1 e IN2: 5V CC
- Corrente de operação: 15 ~ 20 mA
- Tempo de resposta: 5 ~ 10 ms
- 4 furos de 3mm para fixação, nas extremidades da placa
- Dimensões reduzidas: 51 x 38 x 20 mm

3.1.3 Ventoinha 12v (3 Fios)

A ventoinha (ou *Cooler*) é usada para evitar superaquecimento de placas controladoras, dissipadores de calor, motores. Também pode ser utilizada em outras aplicações para resfriamento de equipamentos e componentes. No presente projeto a ventoinha é utilizada para o controle e manutenção dos índices de umidade no interior da chocadeira.

Funciona com alimentação de 12V e tem 95mm de diâmetro, utilizando rolamento para redução de ruídos.

Figura 12 - Ventoinha 12v

Fonte: Próprio autor

3.1.4 Servo Motor SG90

Servo motores são motores elétricos de alto torque, são comumente utilizados em projetos de robótica e aplicações de automação diversas. Possui uma haste conectada a uma engrenagem que é controlada eletronicamente para girar um grau por vez. Diferentemente dos motores CA normais, o servo motor tem além dos pinos

VCC e GND, o pino de sinal, que é utilizado para controlar o servo motor, controlando a posição da haste para qualquer ângulo desejado entre 0º e 180º.

Figura 13 - Servo Motor SG90

fonte https://www.electronics-lab.com/project/using-sg90-servo-motor-arduino/

3.1.5 Sensor de Temperatura e Umidade DHT11

Sensores são dispositivos eletroeletrônicos que têm a propriedade de transformar em sinal elétrico a transformação de uma grandeza física que está relacionada a uma ou mais propriedades do material de que é feito o sensor" (STEFFENS, 2016). Faz parte do projeto sensores de temperatura, umidade e luminosidade.

No presente projeto foi utilizado o sensor DHT11, que é útil devido à sua capacidade de realizar medições de umidade e de temperatura com boa fidelidade ao ambiente, resposta rápida sem interferências e baixo custo (DROBOTICS, 2010).

Composto por um componente medidor de umidade e um termístor NTC para temperatura, ambos conectados a um controlador de 8-bits, o DHT11 utiliza um protocolo simples para enviar as leituras dos sensores usando apenas um fio de barramento.

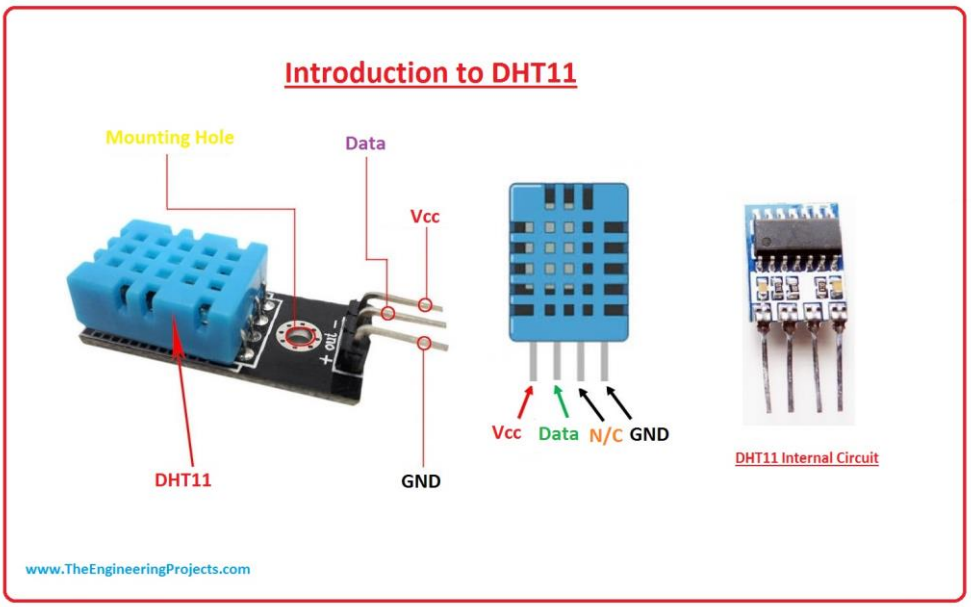
O sensor DHT11 detecta vapor de água medindo a resistência elétrica entre dois eletrodos. Uma umidade relativa mais elevada diminui a resistência entre os

eletrodos, enquanto uma menor umidade relativa aumenta a resistência entre os eletrodos.

O termístor, componente utilizado pelo DHT11 para medir temperatura, é um semicondutor sensível à temperatura. Ele funciona basicamente como um resistor cuja resistência varia em função da temperatura.

Na figura 16 é demonstrado o sensor DHT11 e sua pinagem.

Figura 14 - Pinagem Sensor DHT11



fonte: theengineeringprojects.com

3.1.6 *Display* LCD 16x2

Um *display* muito comum, com controlador HD44780, que se adapta aos mais diversos projetos, podendo ser usado com vários modelos de placas e microcontroladores como Arduino, Raspberry Pi, PIC, etc.

O *display* LCD utilizado no projeto tem 16 colunas e 2 linhas, com *backlight* (luz de fundo) azul e letras na cor branca. Para conexão, são 16 pinos, dos quais usamos 12 para uma conexão básica, já incluindo as conexões de alimentação (pinos 1 e 2), *backlight* (pinos 15 e 16) e contraste (pino 3) (THOMSEN, 2011).

Na figura 17 é apresentado o *display* LCD 16x2 e na tabela 2 é apresentada sua pinagem.

Figura 15 - *Display* LCD 16x2

fonte: https://www.filipeflop.com/produto/*display*-lcd-16x2-*backlight*-verde/

Tabela 2 - Pinagem *Display* LCD 16x2

Conexões LCD 16x2 - HD44780		
Pino LCD	**Função**	**Ligação**
1	Vss	GND
2	Vdd	Vcc 5V
3	V0	Pino central potenciômetro
4	RS	Pino 12 Arduino
5	RW	GND
6	E	Pino 11 Arduino
7	D0	Não conectado
8	D1	Não conectado
9	D2	Não conectado
10	D3	Não conectado
11	D4	Pino 5 Arduino
12	D5	Pino 4 Arduino
13	D6	Pino 3 Arduino
14	D7	Pino 2 Arduino
15	A	Vcc 5V
16	K	GND

Fonte: https://www.filipeflop.com/blog/controlando-um-lcd-16x2-com-arduino/

Estando apresentados os principais elementos utilizados neste projeto, a seguir iremos apresentar como tais elementos são organizados para a implementação e operacionalização da chocadeira proposta neste trabalho.

4 DESENVOLVIMENTO

Este capítulo explica o desenvolvimento do projeto da chocadeira monitorada. Primeiramente na seção 4.1, é comentado sobre as ferramentas de hardware utilizadas, o funcionamento do circuito e suas respectivas conexões. Na seção 4.2 serão comentadas as etapas realizadas pelo software do projeto, incluindo o código do sistema da chocadeira e do sistema de monitoramento pelo aplicativo.

4.1 FERRAMENTAS DE HARDWARE UTILIZADAS

Nesta seção são apresentadas todas as ferramentas e componentes que de alguma forma foram utilizados no desenvolvimento deste projeto, seguido de uma breve explicação e como fora utilizado no projeto.

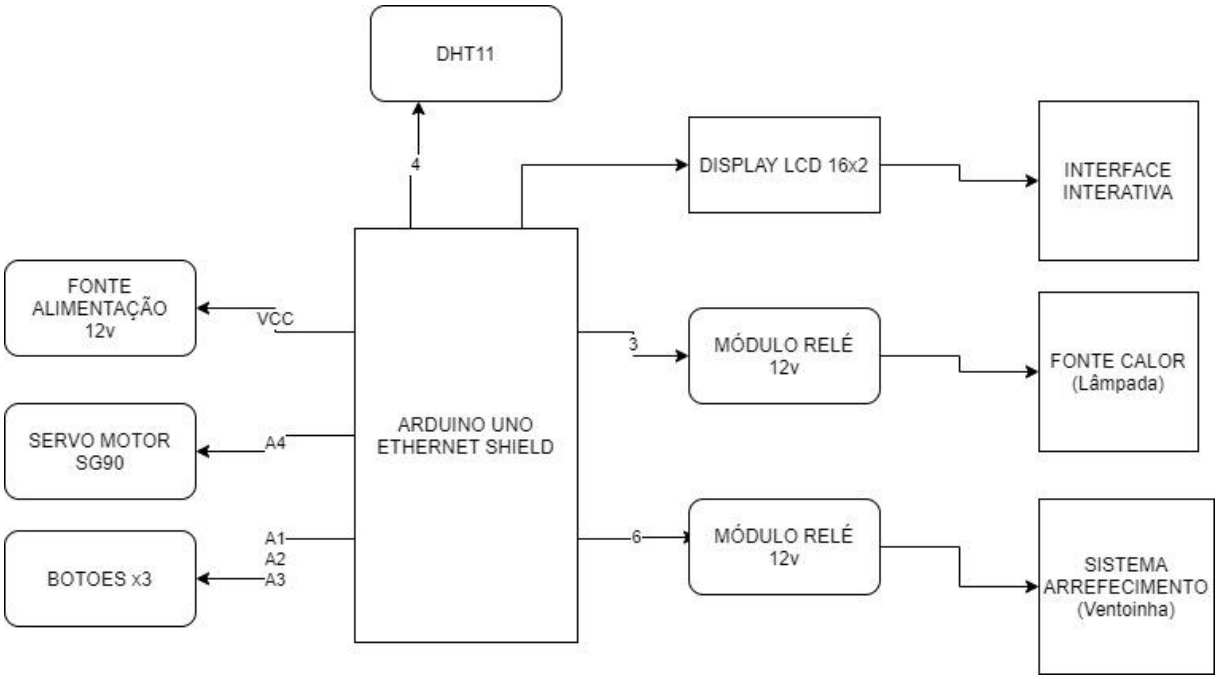
A escolha pela plataforma Arduino foi baseada em sua documentação, consolidação no mercado e facilidade de programação, além da fácil integração de componentes adicionais, permitindo a criação de um completo sistema de automação. Em conjunto com o Arduino foi utilizado o *Ethernet Shield* W5100, que permitiu realizar tanto a comunicação com API e posteriormente com o aplicativo.

A escolha pelo *Ethernet Shield* baseou-se nas possibilidades de conexão diversas e funcionalidades adicionais que poderiam ser agregadas no projeto, como a gravação de dados em cartão de memória.

Juntamente foram utilizados dois módulos relé, que permitiram realizar o acionamento/desligamento da fonte de calor (lâmpada) e resfriamento (ventilador). Também foi utilizado, para a rotação dos ovos um servo motor SG90. Para a alimentação do sistema, foram utilizadas fontes de 220v e 12v.

Inicialmente foi realizada uma apuração de todos os dispositivos que seriam necessários e utilizados no projeto. Um diagrama de blocos da chocadeira proposta, é apresentado na figura 16, mostrando o microcontrolador com seus respectivos componentes e conexões. Para isso, foram atribuídos aos dispositivos um ou mais pinos que os atendiam, após o acoplamento do módulo *Ethernet Shield*.

Figura 16 - Esquemático Projeto



Fonte: Próprio autor

O microcontrolador utilizado, teve sua programação desenvolvida de forma que toda a comunicação realizada passasse pela API remota. Para realizar essa comunicação o microcontrolador foi acoplado ao *Ethernet Shield*, por onde pode receber dados através de uma rede cabeada ou conectado a um roteador móvel ou Wi-Fi.

Para estabelecer a integração do *Ethernet Shield* é necessário realizar a instalação da biblioteca Ethernet, presente no repositório oficial do Arduino. Após isso, é necessário adicionar ao topo do código principal a importação da biblioteca e as configurações da rede local. Esta estrutura de configuração pode ser conferida na seção 4.2

Para o acionamento da lâmpada e ventoinha, fora conectado a duas das portas do Arduino Uno, dois módulos relé. Os módulos relés são responsáveis pelo acionamento e desligamento do circuito elétrico que controla a lâmpada para o aquecimento, e para a ventoinha, que resfria o ambiente. Quando um dos relés for acionado pelo Arduino, então a lâmpada também será acionada ou não, o mesmo vale para a ventoinha.

O relé, conectado à ventoinha, foi alimentado por uma fonte 12V. Outro, para a lâmpada, na rede elétrica doméstica, de tensão 220V. Quando houver mudança de dados, o relé responsável emitirá um pulso no circuito para executar a ação.

Na seção 4.2.1 é exibido em detalhes, o código responsável pelo acionamento dos relés após a verificação das variáveis de controle, recebidas através dos sensores, resfriando, aquecendo ou rotacionando os ovos.

O servo motor é responsável por rotacionar os ovos em até 90º, de 4 em 4 horas, mantendo assim, o embrião intacto. Na chocadeira, o servo motor é ligado a uma bandeja, conectado por um eixo, para realizar a rotação da mesma.

Para estabelecer a integração do Servo Motor ao sistema é necessário realizar a instalação da biblioteca “Servo.h”, presente no repositório oficial do Arduino. Após isso, é necessário adicionar ao topo do código principal a importação da biblioteca.

O sensor DHT11, foi programado e realiza a leitura de dados, a cada 500 ms (0.5s).

Para estabelecer a integração do sensor DHT11 é necessário realizar a instalação da biblioteca “DHT.h”, presente no repositório oficial do Arduino. Após isso, é necessário adicionar ao topo do código principal a importação da biblioteca.

O *Display* LCD, foi programado para exibir mensagens de orientação para o usuário inserir dados de temperatura e umidade. Sua função é implementada em conjunto com 3 botões que estão disponíveis para que o usuário interaja com a interface desenvolvida. Inseridos os dados a serem seguidos pelo sistema, exibe-se a temperatura e umidade captados pelo sensor, juntamente com o aplicativo.

Para estabelecer a integração do *Display* LCD 16x2 é necessário adicionar ao topo do código principal a importação da biblioteca “LiquidCrystal.h”.

4.2 FERRAMENTAS DE SOFTWARE UTILIZADAS

A programação dos componentes de hardware do Arduino foi desenvolvida em sua própria IDE, chamada de Arduino IDE, na versão 1.8.13. A programação do aplicativo foi desenvolvida no ambiente de desenvolvimento do MIT App inventor.

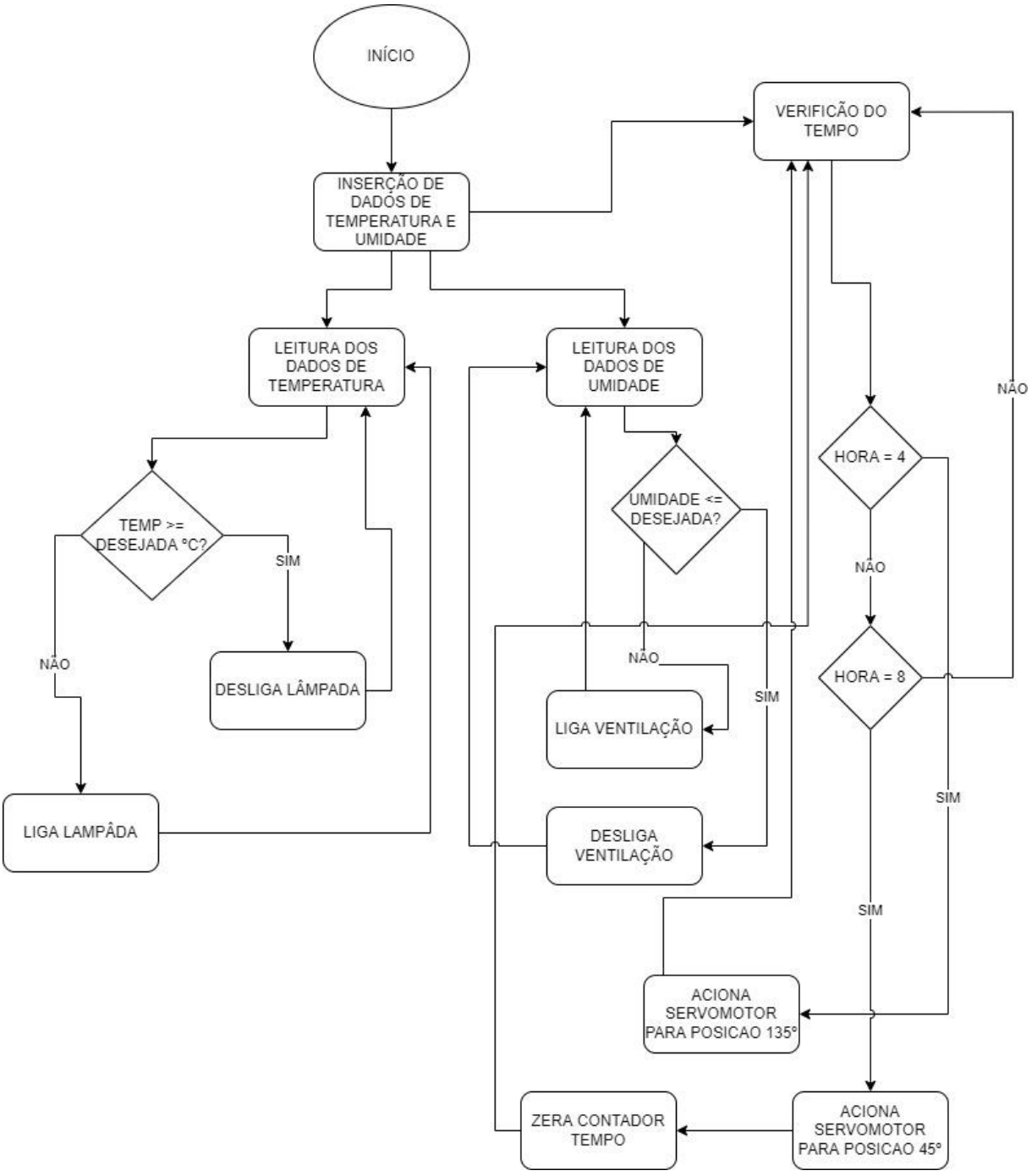
Nas seções seguintes, também serão apresentados os procedimentos realizados para a criação do aplicativo e da API, para que numa possível ampliação

de projeto possa ser modificado. Inicialmente, explica-se como foi criada a interface gráfica na área *Designer* e em seguida as programações feitas na tela *Blocks* e por fim as imagens do próprio aplicativo.

No anexo A se encontra todo o código implementado para o funcionamento do sistema.

O fluxograma na figura 17, demonstra a lógica de funcionamento do software e programação desenvolvidos no IDE do Arduino.

Figura 17 - Lógica Controle de Temperatura



Fonte: Próprio autor

4.2.1 Programação IDE Arduino

Primeiramente, no IDE do Arduino se inserem as devidas bibliotecas dos periféricos e dispositivos utilizados para a realização do projeto da chocadeira.

Figura 18 - Bibliotecas Arduino IDE

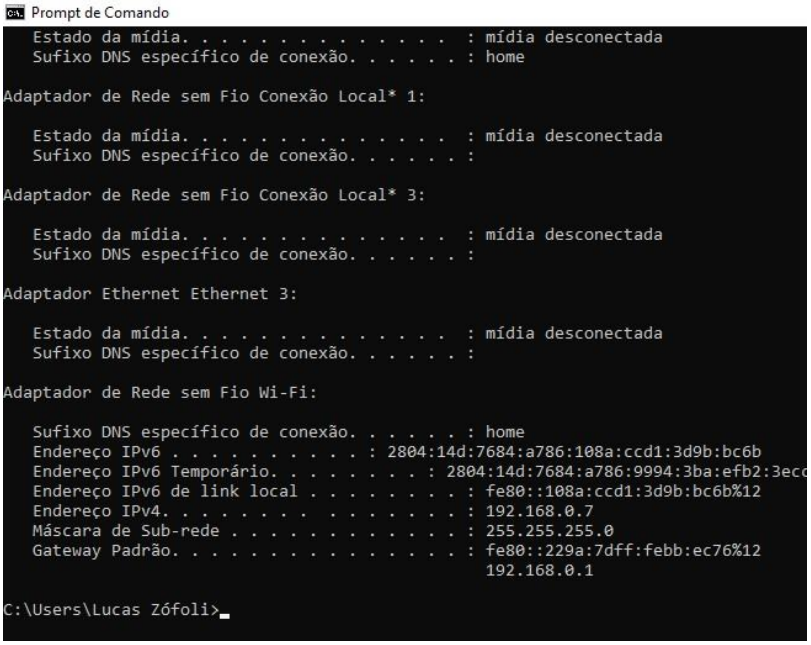
```
#include <LiquidCrystal.h>
#include <Servo.h>
#include <SPI.h>
#include <Ethernet.h>
#include "DHT.h"
#define DHTTYPE DHT11;
#define DHTPIN 4;
DHT dht(DHTPIN, DHTTYPE);
```

Fonte: Próprio autor

Utilizando o prompt de comando do Windows insere-se o comando ipconfig, o qual mostra todas as configurações de rede, bem como Gateway padrão, máscara de sub-rede, endereço IPv4, entre outras configurações. A figura 21 exibe as configurações do roteador.

Como é possível observar na figura abaixo, o gateway padrão é 192.168.0.1. Com este dado já é possível realizar a configuração do Arduino e a rede.

Figura 19 - Comando ipconfig no Windows



Fonte: Próprio autor

No Sketch primeiramente inserem-se as bibliotecas para realizar a comunicação do Shield com o Uno R3. Para isso utiliza-se a biblioteca SPI, que é responsável por essa comunicação serial. A ligação é realizada pelos pinos 10;11;12;13, como anteriormente citado no tópico 3.1.1, sendo assim esses pinos ficam impossibilitados de serem usados para outras funções. Também é inserido a biblioteca Ethernet para a comunicação com a internet, com a instrução #include e inclui-se as bibliotecas no programa. Em seguida, realiza-se as configurações de rede, inserindo-se o *mac adress* e o IP. Para o *mac address* é utilizado o comando *byte mac*, e nele é inserido um código em hexadecimal que irá buscar automaticamente o endereço mac do *shield*. Para a configuração do IP, deve ser diferente do IP do computador e do gateway padrão. Para o projeto foi utilizado o IP, em seguida, utilizando o comando EthernetServer server (80), define-se a porta 80 do roteador para a comunicação do Arduino com a Internet

Figura 20 - Configuração *Ethernet Shield*

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = {192, 168, 0, 23 }; //Enter the IP of ethernet shield
byte serv[] = {192, 168, 0, 7} ; //Enter the IPv4 address

EthernetClient cliente;
```

Fonte: Próprio autor

Na sequência, são definidas e implementadas as variáveis referentes a entradas e saídas das portas lógicas correspondentes aos pinos de comando do Arduino Uno. Também são definidos alguns valores de variáveis, referentes ao monitoramento do tempo, para a implementação do controle do servo motor e funcionamento dos botões (*Push Buttons*). As figuras 21 e 22 mostram esta implementação em detalhes.

Figura 21 - Variáveis e Pinagem

```
const int ok = A1;
const int UP = A2; //BOTOES
const int DOWN = A3;

const int bulb = 3; //LAMPADA
const int vap = 6; //FAN

const int rs = 8;
const int en = 9;
const int d4 = 10;
const int d5 = 11; //DISPLAY
const int d6 = 12;
const int d7 = 13;

int ack = 0;
int pos = 0;
int sec = 0;
int Min = 0;   // TEMPO E SERVO MOTOR
int hrs = 0;
int T_threshold = 30;
int H_threshold = 60;
int SET = 0;
int Direction = 0;
boolean T_condition = true;
boolean H_condition = true;
```

Fonte: Próprio autor

Figura 22 - Entradas e saídas lógicas

```
void setup()
{

Serial.begin(9600); //setting
Ethernet.begin(mac, ip);
dht.begin();
pinMode(ok, INPUT);
pinMode(UP, INPUT);
pinMode(DOWN, INPUT);
pinMode(bulb, OUTPUT);
pinMode(vap, OUTPUT);
digitalWrite(bulb, LOW);
digitalWrite(vap, LOW);
digitalWrite(ok, HIGH);
digitalWrite(UP, HIGH);
digitalWrite(DOWN, HIGH);
motor.attach(A4);
motor.write(pos);
lcd.begin(16, 2);
Serial.begin(9600);
```

Fonte: Próprio autor

Para o funcionamento do sistema, primeiramente o usuário, por meio de 2 botões interruptores (*Push Buttons*), irá selecionar a temperatura e umidade desejada, incrementando ou decrementando os valores manualmente, em seguida por intermédio de um terceiro botão, confirma os dados exibidos no *display* LCD. Em um sistema de aquecimento e resfriamento, é aconselhável usar um valor de *Threshold*, que nada mais é do que um limite mínimo de variação de valores que deve ser observado para que o sistema considere realmente que houve uma variação. Assim, o que para evitar com que o sistema fique oscilando, ligando e desligando o aquecimento muito rapidamente, o que no caso de um relé poderia levar a superaquecimento.

E assim, os dados recebidos pelos sensores começarão a ser exibidos no *display* LCD. Nas figuras 23 e 24, é mostrada a programação para o funcionamento dos botões e *display* LCD.

Figura 23 - Implementação botões e LCD (1)

```
if (SET == 0)
{
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Insira Temperatura:");
lcd.setCursor(0, 1);
lcd.print(T_threshold);
lcd.print(" *C");
while (T_condition)
{
if (digitalRead(UP) == LOW)
{
T_threshold = T_threshold + 1;
lcd.setCursor(0, 1);
lcd.print(T_threshold);
lcd.print(" *C");
delay(200);
}
if (digitalRead(DOWN) == LOW)
{
T_threshold = T_threshold - 1;
lcd.setCursor(0, 1);
lcd.print(T_threshold);
lcd.print(" *C");
delay(200);
}
if (digitalRead(ok) == LOW)
{
```

Fonte: Próprio autor

Figura 24 - Implementação botões e LCD (2)

```
H_condition = false;
}
}
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Insira a Umidade:");
lcd.setCursor(0, 1);
lcd.print(H_threshold);
lcd.print("%");
delay(100);
while (H_condition)
{
if (digitalRead(UP) == LOW)
{
H_threshold = H_threshold + 1;
lcd.setCursor(0, 1);
lcd.print(H_threshold);
lcd.print("%");
delay(100);
}
if (digitalRead(DOWN) == LOW)
{
H_threshold = H_threshold - 1;
lcd.setCursor(0, 1);
lcd.print(H_threshold);
lcd.print("%");
delay(200);
}
if (digitalRead(ok) == LOW)
```

Fonte: Próprio autor

O sensor DHT11 realiza verificações de temperatura e umidade a cada 0.5 segundos, enviando os dados para o Arduino, e consequentemente para o aplicativo. De acordo com o valor das variáveis de temperatura e umidade recebidas, a ação de acionamento ou desligamento dos relés é executada, assim acionando ou não a fonte de calor (lâmpada) e de resfriamento (ventoinha), como mostrado nas figuras 26 e 27. O envio da medição da temperatura e umidade ambiente é um processo simples. Para isto, foi utilizada a instrução “client.print” seguida da variável temperatura, essa instrução é responsável por enviar os dados ao servidor que o cliente está conectado (ARDUINO, 2015), como mostrado na figura 28. A atualização da leitura é configurada na construção do aplicativo, que no caso, ocorre a cada 1 segundo. Simultaneamente os dados recebidos de temperatura e umidade, são exibidos no *display* LCD.

Figura 25 – Leitura do sensor DHT11

```
ack = 0;
int chk;
chk = DHT.read(DHT11_PIN);    // LEITURA DO SENSOR DHT
switch (chk)
{
case DHTLIB_OK:
//Serial.print("OK,\t");
break;
case DHTLIB_ERROR_CHECKSUM:
//Serial.print("Checksum error,\t");
ack = 0;
break;
case DHTLIB_ERROR_TIMEOUT:
//Serial.print("Time out error,\t");
ack = 0;
break;
default:
//Serial.print("Unknown error,\t");
break;
}
// DISPLAY DATA
Serial.print("DHT11, \t");
Serial.print(DHT.temperature,1);
Serial.print(",\t");
Serial.println(DHT.humidity,1);
delay(100);
```

Fonte: Próprio autor

Figura 26 – Controle da temperatura e umidade (1)

```
if (DHT.temperature >= T_threshold)  //CONTROLE DE TEMPERATURA
{
delay(500);
if (DHT.temperature >= T_threshold)
{
digitalWrite(bulbo, LOW);
}
}
if (DHT.humidity >= H_threshold)
{
delay(500);
if (DHT.humidity >= H_threshold)
{
digitalWrite(fan, HIGH);
}
}
```

Fonte: Próprio autor

Figura *27* - Controle da temperatura e umidade (2)

```
}
if (DHT.temperature < T_threshold)
{
delay(500);
if (DHT.temperature < T_threshold)
{
digitalWrite(bulbo, HIGH);
}
}
if (DHT.humidity < H_threshold)
{
delay(500);
if (DHT.humidity < H_threshold)
{
digitalWrite(fan, LOW);
}
}
```

Fonte: Próprio autor

Fora desenvolvida utilizando HTML e CSS, uma página que apresenta os dados de temperatura, umidade e data de leitura dos sensores;

Para os testes é necessário abrir um navegador de internet e nele inserir o IP configurado no Arduino. Ao acessar é possível verificar a página formatada de acordo com a programação realizada no IDE. Quando conectado ao servidor, os dados mais recentes são enviados e salvos no banco de dados, como é possível conferir na figura 30. Foi criada uma pasta chamada ‘ethernet’ e um arquivo nomeado de ‘connection.php’, para fazer a conexão do Arduino com o Banco de Dados. Também foi criado outro arquivo na pasta ethernet, chamado de ‘data.php’, para realizar o registro dos valores lidos, no Banco de Dados. Por fim, foi criado outro arquivo PHP, denominado “*display*”, que lê os dados enviados ao Banco de Dados e os apresenta de acordo com a programação realizada no arquivo.

Figura 28 – Envio dos dados ao servidor

```
 if (cliente.connect(serv, 80)) { //Connecting at the IP address and port we saved before
Serial.println("connected");
cliente.print("GET /ethernet/data.php?"); //Connecting and Sending values to database
cliente.print("temperature=");
cliente.print(temperature);
cliente.print("&humidity=");
cliente.print(humidity);


Serial.print("Temperature= ");
Serial.println(temperature);
Serial.print("Humidity= ");
Serial.println(humidity);

cliente.stop();
}
else {
// if you didn't get a connection to the server:
Serial.println("connection failed");
}
delay(5000);
}
```

Fonte: Próprio autor

O código verifica o tempo (segundos, minutos e horas) de funcionamento da chocadeira. Quando a variável horas recebe o valor de 4, o servo motor é acionado, girando sua haste da posição 45º para 135º, uma mensagem de aviso no *display* é exibida. Passando mais 4 horas, a variável horas recebe o valor 8, o servo motor é acionado, girando sua haste da posição 135º para 45º e outra mensagem de aviso é exibida. Em seguida a variável hora é decrementada, e recomeça a contagem do tempo. A implementação deste trecho do código está na figura 29, abaixo.

Figura 29 – Acionamento SG90

```
sec = sec + 1;   //CONTROLE DO SERVO
if (sec == 60)
{
sec = 0;
Min = Min + 1;
}
if (Min ==60)
{
Min = 0;
hrs = hrs + 1;
}
if (hrs == 4 && Min == 0 && sec == 0)
{Serial.println("  ROTACIONANDO PARA FRENTE  ");
for (pos = 45; pos <= 135 ; pos += 1)
{
motor.write(pos);
delay(25);
}
}
if (hrs == 8 && Min == 0 && sec == 0)
{Serial.println("  ROTATING PARA TRAS  ");
hrs = 0;
for (pos = 135; pos <= 45; pos -= 1)
{
motor.write(pos);
delay(25);
```

Fonte: Próprio autor

4.2.2 Programação do Aplicativo, API e Banco de Dados

Nesta seção, são apresentados os procedimentos realizados para a criação do aplicativo através do App Inventor, para que numa possível ampliação de projeto possa estar modificando suas funcionalidades.

Com necessidade de armazenar dados no servidor, criou-se um banco de dados MySQL, para registrar os valores de temperatura, umidade e informações como índices, datas e horários de registro e gerar relatórios. Com isso, realizou-se a inserção de uma tabela de nome “Data”, q pode ser conferida na figura 32 abaixo.

Figura 30 - Tabela Data

	#	Nome	Tipo	Agrupamento (Collation)	A
☐	1	id	int(11)		
☐	2	event	timestamp		
☐	3	temperature	varchar(10)	utf8mb4_general_ci	
☐	4	humidity	varchar(10)	utf8mb4_general_ci	

Fonte: Próprio autor

A API responsável pela integração entre o aplicativo Android, banco de dados e o Arduino foi construída utilizando a linguagem PHP e hospedada em um servidor web Apache do Xampp. Nela há possibilidade da inserção, deleção e modificação dos dados presentes no banco de dados MySQL. A figura 31 mostra como a API está posicionada dentro do projeto.

Figura 31 - Estrutura API

Fonte: Próprio autor

Conforme a figura 31, a API recebe e envia dados para todas estruturas. O Aplicativo e o Arduino necessitam requisitar diretamente a ela para obter dados. Essa estrutura permite uma maior facilidade na manutenção e integração com outros sistemas. Toda a troca de dados que é realizada entre a API, Aplicativo e Arduino é feita no formato de dados JSON.

A comunicação da API com o banco de dados foi utilizada a extensão MySQL, Ela faz a conexão diretamente com o banco de dados e permite que utilize consultas para obter o resultado desejado.

O aplicativo foi desenvolvido visando um método simples para o recebimento e visualização de dados coletados pelos sensores do Arduino. Na área *Designer*, para o menu principal. Lá se insere quais serão as conectividades do aplicativo que são componentes não visíveis, implantando Web1 para realizar a comunicação com o localhost ou Internet, que fornece funções do tipo HTTP GET, POST, PUT (MIT APP INVENTOR).

Para a tela de apresentação dos dados de temperatura e umidade foram colocados os *textbox* e *labels* com as denominações referentes, que recebem os valores lidos pelo sensor. Essas informações não podem ser alteradas pelo usuário do aplicativo, e os dados são atualizados a cada intervalo de tempo, conforme definido pelo *Clock*.

Figura 32 - Apresentação Aplicativo no Ambiente de Desenvolvimento

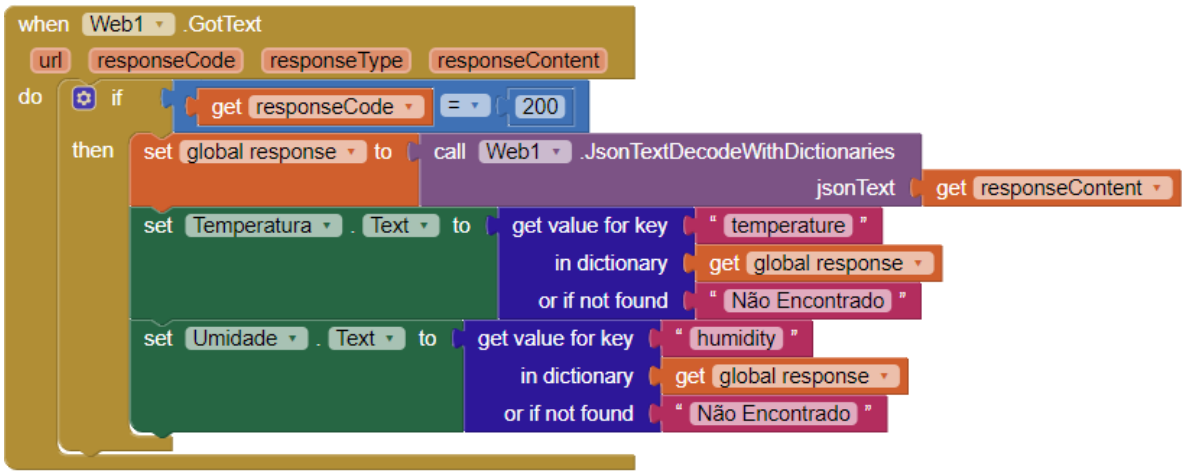
Fonte: Próprio autor

A seção *Blocks* no App Inventor ficou separada de acordo com as telas criadas na parte *Designer*. Nesta seção, cada componente criado na seção anterior tem diversas funções na qual o usuário define qual vai ser utilizada, como por exemplo, as instruções quando um botão é clicado, definições de variáveis, recebimento de dados,

dentre outras funcionalidades. Um ponto importante na criação de um aplicativo é a definição de *tags* para os componentes a fim de facilitar a programação.

Para a tela de visualização dos dados, é necessária a interpretação dos dados que os sensores do Arduino enviam ao banco de dados e consequentemente são recebidos pelo aplicativo. Utilizando e direcionando a função de conectividade *Web1* para a URL local que contém o arquivo com os dados em formato JSON captados, o aplicativo, através da programação realizada em bloco, recebe os dados de temperatura e umidade, e os exibe nos campos correspondentes, de acordo com o *clock* configurado na área *Design*. A figura 33 mostra como é a estrutura responsável pela integração e apresentação dos dados recebidos e enviados pela API.

Figura 33 – Bloco de Programação API



Fonte: Próprio autor

5 MONTAGEM DO PROJETO

Neste capítulo será explicado como se deu o processo de montagem, programação, dificuldades enfrentadas, testagem e coleta de dados durante todo o processo do trabalho de conclusão de curso.

5.1 DESENVOLVIMENTO DO PROTÓTIPO DE CHOCADEIRA

Inicialmente, pesquisou-se sobre a utilização do microcontrolador Arduino, sua IDE e periféricos para aquisição de dados de um sensor de temperatura e controle da mesma. Na sequência foi estudada a possibilidade de criação de um aplicativo para *smartphones* utilizando o ambiente MIT App Inventor, que possui uma grande praticidade de montagem de aplicativos e diferentes métodos de comunicação com o Arduino.

Logo após, durante 30 dias, foram montados em uma *protoboard* vários circuitos, com diferentes propostas de projetos, para maior entendimento das ferramentas a serem utilizadas para o trabalho de conclusão do curso. Desde simples projetos como acender e controlar LEDs, até projetos mais elaborados, com finalidades reais, com controle de relés e diversas variáveis.

Para o controle da ventilação do sistema analisou-se diferentes modelos de ventoinhas, em diferentes projetos, sendo uma ventoinha de 12V escolhida.

Após a montagem do hardware, foi realizada a programação da página do servidor do Banco de Dados (MySQL) e programação no IDE do Arduino, e em seguida foi iniciada a programação para a montagem do aplicativo. Criou-se um projeto bastante simples, todavia, surgiram dificuldades na realização da programação do aplicativo para fazer com que ele se conectasse com a API. Consequentemente, optou-se por utilizar o *Ethernet Shield*. Ele é conectado ao Arduino e com isso um *smartphone* ou um notebook conectado à mesma rede pode ter acesso aos dados do Arduino, decidiu-se assim esta opção de comunicação a ser utilizada no projeto.

A coleta de dados se deu pelo monitoramento da umidade, temperatura, dia e hora, obtidas pela leitura dos sensores presentes na chocadeira.

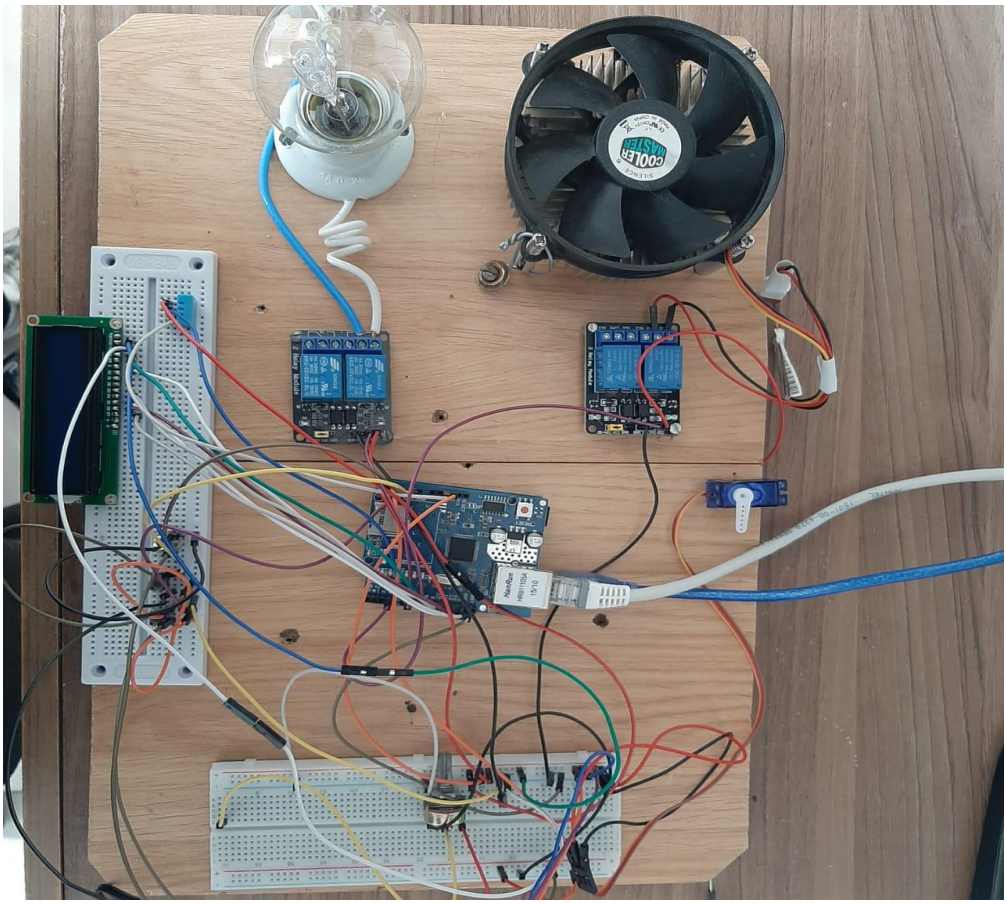
5.2 ORÇAMENTO

Os recursos para a execução do projeto foram provenientes de investimentos pessoais, materiais como multímetro e fontes para testes foram cedidas pelo professor orientador.

5.3 AMBIENTE DE TESTES E RESULTADOS

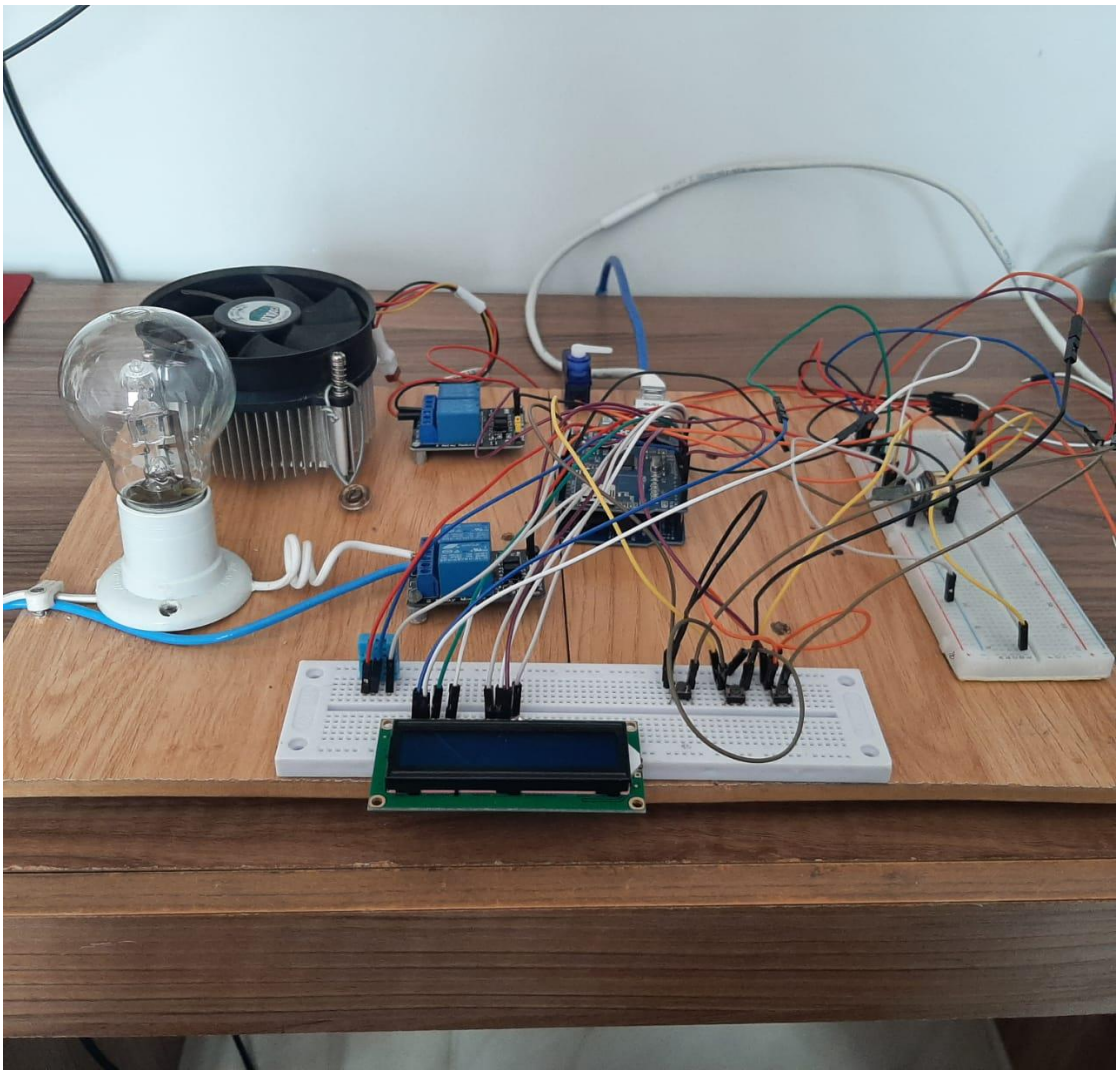
Por conta da pandemia de covid-19, e suas implicações, todos os testes foram realizados na residência do aluno.

Figura 34 – Protótipo montado da Chocadeira Automatizada (1)



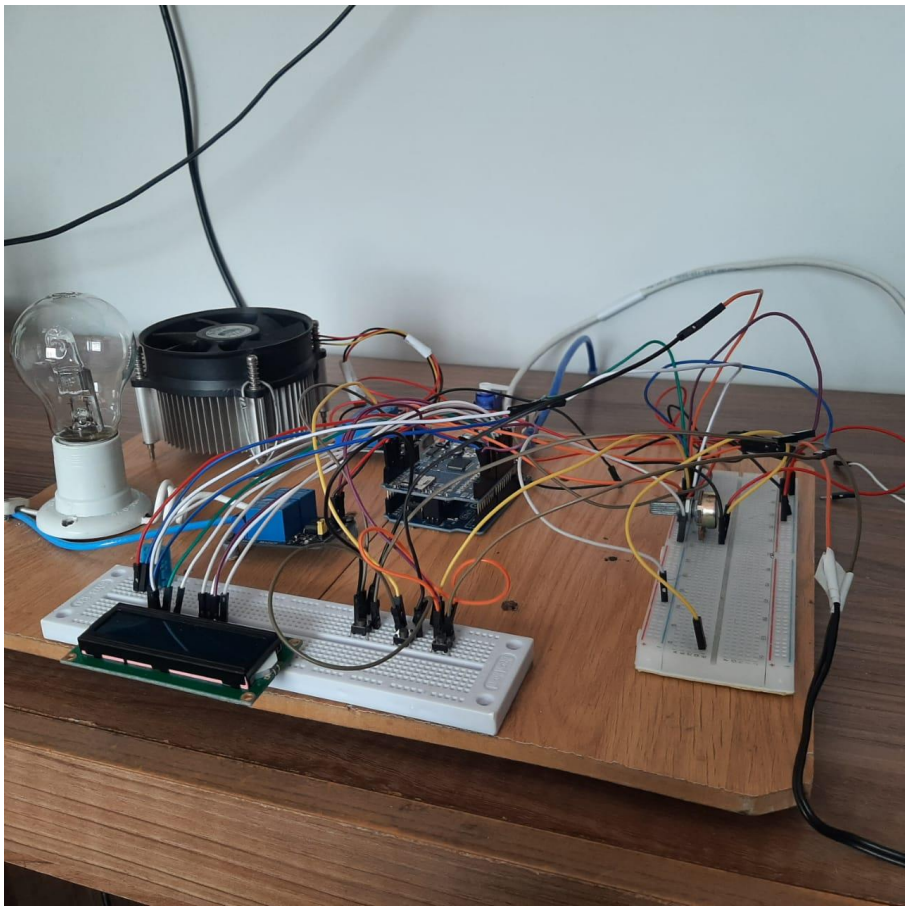
Fonte: Próprio autor

Figura 35 – Protótipo montado da Chocadeira Automatizada (2)



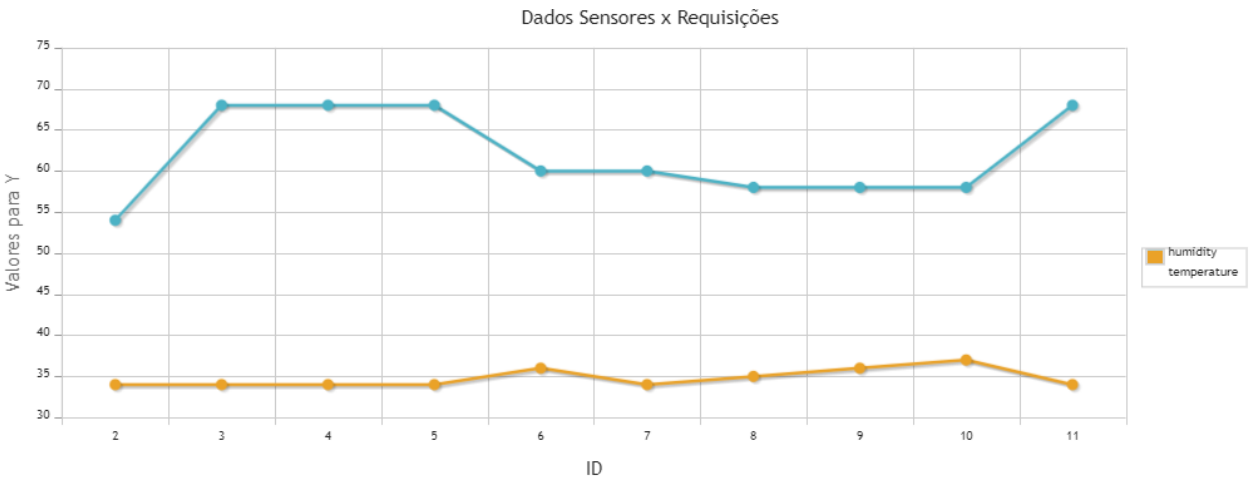
Fonte: Próprio autor

Figura 36 – Protótipo montado da Chocadeira Automatizada (3)



Fonte: Próprio autor

Figura 37 – Dados dos Sensores x Requisições



Fonte: Próprio autor

Figura 38 - Print da Tela do Celular com o App em Funcionamento

Fonte: Próprio autor

O teste com a chocadeira não durou o ciclo completo para a chocagem dos ovos. Utilizou-se as medições de acordo com os padrões que cada sensor deveria emitir, como por exemplo, o sensor de temperatura atingiu de acordo com a programação do Arduíno o valor adequado para a chocagem dos ovos, ou seja, o sensor apresentou valores satisfatórios para todo o ciclo dos ovos.

O teste com o protótipo de chocadeira atingiu os resultados esperados de medição, manutenção do ambiente para chocagem dos ovos, coleta e envio de dados através de um servidor. Todo o sistema é controlado pelo microcontrolador, junto com fontes, relés, circuitos e programação. Todos os sensores, periféricos, servidor e aplicativo funcionam de acordo com a programação.

Espera-se também que o monitoramento remoto, acessado pelo usuário do sistema, haja uma diminuição na perda de chocagens devido a variações na temperatura, variações na umidade, não rotação dos ovos, falta de ventilação (o que

ocasiona contaminação por bactérias) e morte por chocagens que não foram retiradas da chocadeira após a eclosão.

As maiores dificuldades encontradas estão relacionadas à escassez de material em português e inglês sobre a comunicação do *Ethernet Shield* com os aplicativos criados no App Inventor. Em grande parte de projetos que de alguma forma utilizavam controle de temperatura, utilizava-se o módulo *bluetooth* e diferentes visões de controle. Por esta razão, tornou-se necessário entender sobre os laços de controle utilizado pelo App Inventor e após consultas em artigos e documentações foi possível obter uma melhor compreensão sobre o funcionamento do software.

Para a implementação do sistema em um modelo real foi realizado um levantamento dos custos, tomando por base produtos semelhantes encontrados no mercado brasileiro e o quanto foi gasto com os materiais adquiridos. Na tabela 3 abaixo estão os custos aproximados, baseados no que foi gasto para uma implementação prática do projeto proposto por este trabalho.

Tabela 3 - Valores de Implementação do Projeto

EQUIPAMENTO	QUANTIDADE	VALOR	TOTAL
Kit Arduino Uno R3	1	R$ 120,00	R$ 120,00
Ethernet Shield	1	R$ 70,00	R$ 70,00
Módulo Relé 2 Canais	2	R$ 15,00	R$ 30,00
Lâmpada	1	R$ 8,00	R$ 8,00
Ventoinha	1	R$ 30,00	R$ 30,00
Valor Total	**R$ 258,00**		

Fonte: Próprio autor

6 CONCLUSÃO

Este trabalho teve como objetivo principal integrar hardware e software, permitindo a automatização e monitoramento de um sistema para chocar ovos. Tendo em vista os resultados encontrados durante o desenvolvimento, conclui-se que esse objetivo foi cumprido.

Em virtude disso, foi possível demonstrar que a criação de aves e a tecnologia oferecem boas oportunidades para automatização de processos, oportunizando novos negócios e produtividade para o produtor. Apesar das dificuldades impostas pela pandemia e pela falta de documentação em determinados processos, foi possível testar o sistema.

Todo conhecimento adquirido, tanto de hardware, software, web services e microeletrônica, foi de grande valia para a experiência profissional.

6.1 TRABALHOS FUTUROS

Apesar do sucesso da proposta do trabalho, muitos outros requisitos ainda podem ser implementados para que o sistema se torne estável, podendo aplicá-lo como produto final. Podemos iniciar verificando do ponto de vista do usuário final para maior monitoramento dos ovos, seria a implementação de uma câmera interna, associada a notificação do produtor e a interação pelo aplicativo para manutenção das condições de temperatura através de requisições PUT. Também poderia ser implementado o uso de uma bomba d'água para auxílio no sistema de arrefecimento e controle da umidade, acoplamento de um módulo RTC para maior otimização e monitoramento na questão do tempo de incubação e a montagem de uma estrutura externa em MDF para conservação das condições de temperatura e umidade.

Outro ponto a ser discutido e estudado em um projeto futuro, seria um cálculo de eficiência energética do projeto como um todo, o uso de uma resistência no lugar de uma lâmpada, pois a vida útil da mesma é de no máximo 1000 horas.

6.2 REFERÊNCIAS

AMANDO, Maria Roziane da Silva. **A importância da criação de galinhas como fonte de renda no assentamento Mandassaia-Orocó/PE,** 2018. Monografia - Universidade Federal Do Vale De São Francisco- UNIVASF

ARDUINO, print(), 2015, Disponível em:<**https://www.arduino.cc/en/Reference/ClientPrint>.** Acesso em 16/06/2021

BUSINESS WIRE, **CES 2019 prova que a IA e o 5G transformarão o futuro**, 2019 Disponível em: https://www.businesswire.com/news/home/20190112005015/pt/>. Acesso em 30/05/2021

CONOLLY, Aidan. **Era digital: o futuro da tecnologia avícola,** 2018**.** Disponível em:<htttps://www.aviculturaindustrial.com.br/imprensa/era-digital-o-futuro-da-tecnologia-avicola/20180522-093843-w047> Acesso em 10/10/2020

COBB VANTRESS, inc. **Guia de manejo de incubação**, 2008. Disponível em: https://wp.ufpel.edu.br/avicultura/files/2012/04/Guia_incuba%C3%A7%C3%A3o_Cobb.pdf. Acesso em: 10/07/2021

COSTA, C. Sistemas de Informação em Organizações. **ITML press/Lusocredito**, 2011. Disponível em: <https://books.google.com.br/books?id=GWyDDwAAQBAJ>. Acesso em 10/07/2021

DESTACOM/UFMS, **Despertando Talentos em Computação - Universidade Federal do Mato Grosso do sul** 2014, Disponível em :<https://destacom.ufms.br/sobre-o-app-inventor/>. Acesso em : 10/05/2021

DIDONÉ, D.; CHAULET, F. J. Implantação e Administração de Serviços web. **Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco**, 2016. Disponível em: <https://www.ufsm.br/unidades-universitarias/ctism/cte/wp-content/uploads/sites/413/2018/12/arte_implantacao_administracao_servicos_web.pdf>. Acesso em 13/07/2021

FERNANDES, André. O que é API? Entenda de uma maneira simples, **Vertigo Tecnologia**, 2018. Disponível em <: https://vertigo.com.br/o-que-e-api-entenda-de-uma-maneira-simples/>. Acesso em 10/07/2021

FRENCH, N. A. Modeling Incubation Temperature: The Effects of Incubator Design Embryonic Development, and Egg Size. **Poultry Science,** vol. 76, 1997**.** Disponível em :<https://www.sciencedirect.com/science/article/pii/S0032579119405257>. Acesso em 10/10/2020

GUIMARÃES, Gleyser. A história do Sistema operacional Android, **Departamento de Sistemas e Computação da Universidade Federal de Campina Grande**, ago. 2013. Disponível em:< http://www.dsc.ufcg.edu.br/~pet/jornal/agosto2013/materias/historia_da_computacao.html>. Acesso em: 10/06/2021

HARB, S. K.; HABBIB, Y. A.; KASSEM A. M.; EL RAIES, A. Energy Consumption For Poultry Egg Incubator To Suituit Small Farmer, **Egypt Journal for Agricultural Research**, vol. 88, 2010. Disponível em :<https://agris.fao.org/agris-search/search.do?recordID=EG2012000235>. Acesso em 10/10/2020

HIGA, Paulo, **O que é XAMPP e para que serve. TECHTUDO, 2012.** Disponível em: <https://www.techtudo.com.br/dicas-e-tutoriais/noticia/2012/02/o-que-e-xampp-e-para-que-serve.html>. Acesso em 11/05/2021

MIT APP INVENTOR, **About us**, 2014. Disponível em :<https://appinventor.mit.edu/about-us>. Acesso em 10/05/2021

MITCHELL, L.; KINOSHITA, L. **WEB SERVICES EM PHP: APIS PARA A WEB MODERNA.** 2013. Disponível em: <https://books.google. com.br/books?id=YYM1nQEACAAJ>. Acesso em 13/07/2021

MORA, L. A. **Processo de incubação artificial de ovos: desenvolvimento de sistemas de medição de temperatura em massa**. 2008. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas.

PISA, Pedro, **O que é e como usar o MySQL. UOL**, 2012**.** Disponível em :< https://www.techtudo.com.br/artigos/noticia/2012/04/o-que-e-e-como-usar-o-mysql.html>. Acesso em 10/05/2021

RAUBER, Rafaela, **Introdução ao Microcontrolador ATMega328P,** 2019. Disponível em :< https://visiorob.com.br/index.php/2019/09/15/introducao-ao-microcontrolador-atmega328p/>. Acesso em 30/05/2021

THE INCUBATOR SHOP, **Incubation Guide**, 2016. Disponível em:<http://www.theincubatorshop.co.uk/. Accesso em 10/10/2020

THOMSEN, Adilson, **Controlando um LCD 16x2 com arduino,** 2013, Disponível em: https://www.filipeflop.com/blog/controlando-um-lcd-16x2-com-arduino/>. Acesso em 30/05/2021

THOMSEN, Adilson, 2013, Disponível em: :<https://www.filipeflop.com/blog/controle-modulo-rele-arduino/>. Acesso em 10/5/2021

UOL, ANDROID, 2015, Disponível em :< https://assistenciatecnica.uol.com.br/dicas/o-que-e-android.html#rmcl>. Acesso em 10/05/2021.

YILMAZ, A.; TEPELI C.; GARIP, M.; ÇAĞLAYAN, T. The effects of incubation temperature on the sex of Japanese quail chicks, **Poultry Science** , vol. 90, 2011. Disponível em :<https://www.sciencedirect.com/science/article/pii/S0032579119422979>. Acesso em 10/10/2020

ZAMBARDA, 2014, Disponível em: :<https://www.techtudo.com.br/noticias/noticia/2014/08/internet-das-coisas-entenda-o-conceito-e-o-que-muda-com-tecnologia.html>. Acesso em 10/05/2021

6.3 ANEXOS

ANEXO A - CÓDIGO IDE ARDUINO

```
#include <LiquidCrystal.h>
#include <Servo.h>
#include <SPI.h>
#include <Ethernet.h>
#include "DHT.h"
#define DHTTYPE DHT11;
#define DHTPIN 4;
DHT dht(DHTPIN, DHTTYPE);

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = {192, 168, 0, 23 }; //Enter the IP of Ethernet Shield
byte serv[] = {192, 168, 0, 7} ; //Enter the IPv4 address

EthernetClient cliente;

const int ok = A1;
const int UP = A2; //BOTOES
const int DOWN = A3;

const int bulb = 3; //LAMPADA
const int vap = 6; //FAN

const int rs = 8;
const int en = 9;
const int d4 = 10;
const int d5 = 11; //DISPLAY
const int d6 = 12;
const int d7 = 13;

int ack = 0;
int pos = 0;
int sec = 0;
int Min = 0;   // TEMPO E SERVO MOTOR
int hrs = 0;
int T_threshold = 30;
int H_threshold = 60;
int SET = 0;
int Direction = 0;
boolean T_condition = true;
boolean H_condition = true;
```

```
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
Servo motor;

void setup()
{

Serial.begin(9600); //setting the baud rate at 9600

Ethernet.begin(mac, ip);

dht.begin();
pinMode(ok, INPUT);
pinMode(UP, INPUT);
pinMode(DOWN, INPUT);
pinMode(bulb, OUTPUT);
pinMode(vap, OUTPUT);
digitalWrite(bulb, LOW);
digitalWrite(vap, LOW);
digitalWrite(ok, HIGH);
digitalWrite(UP, HIGH);
digitalWrite(DOWN, HIGH);
motor.attach(A4);
motor.write(pos);
lcd.begin(16, 2);
Serial.begin(9600);

lcd.clear();
lcd.setCursor(0,0);
lcd.print("Temperature &");
lcd.setCursor(0,1);
lcd.print("Humidity ");
delay (3000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("CONTROLE DA");
lcd.setCursor(0,1);
lcd.print("CHOCADEIRA");
delay (3000);
lcd.clear();
Serial.println("  CONTROLE DA UMIDADE E TEMPERATURA DA
CHOCADEIRA  ");
}
void loop()
{

if (SET == 0)
{
lcd.clear();
lcd.setCursor(0, 0);
```

```
lcd.print("Insira Temperatura:");
lcd.setCursor(0, 1);
lcd.print(T_threshold);
lcd.print(" *C");
while (T_condition)
{
if (digitalRead(UP) == LOW)
{
T_threshold = T_threshold + 1;
lcd.setCursor(0, 1);
lcd.print(T_threshold);
lcd.print(" *C");
delay(200);
}
if (digitalRead(DOWN) == LOW)
{
T_threshold = T_threshold - 1;
lcd.setCursor(0, 1);
lcd.print(T_threshold);
lcd.print(" *C");
delay(200);
}
if (digitalRead(ok) == LOW)
{
delay(200);
T_condition = false;
}
}
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Insira a Umidade:");
lcd.setCursor(0, 1);
lcd.print(H_threshold);
lcd.print("%");
delay(100);
while (H_condition)
{
if (digitalRead(UP) == LOW)
{
H_threshold = H_threshold + 1;
lcd.setCursor(0, 1);
lcd.print(H_threshold);
lcd.print("%");
delay(100);
}
if (digitalRead(DOWN) == LOW)
{
H_threshold = H_threshold - 1;
lcd.setCursor(0, 1);
lcd.print(H_threshold);
```

```
lcd.print("%");
delay(200);
}
if (digitalRead(ok) == LOW)
{
delay(100);
H_condition = false;
}
}
SET = 1;
}
ack = 0;
int chk;
chk = DHT.read(DHT11_PIN);    // LEITURA DO SENSOR DHT
switch (chk)
{
case DHTLIB_OK:
//Serial.print("OK,\t");
break;
case DHTLIB_ERROR_CHECKSUM:
//Serial.print("Checksum error,\t");
ack = 0;
break;
case DHTLIB_ERROR_TIMEOUT:
//Serial.print("Time out error,\t");
ack = 0;
break;
default:
//Serial.print("Unknown error,\t");
break;
}
// DISPLAY DATA
Serial.print("DHT11, \t");
Serial.print(DHT.temperature,1);
Serial.print(",\t");
Serial.println(DHT.humidity,1);
delay(100);

 if (cliente.connect(serv, 80)) { // printando valores para o client
Serial.println("connected");
cliente.print("GET /ethernet/data.php?"); //
cliente.print("temperature=");
cliente.print(temperature);
cliente.print("&humidity=");
cliente.print(humidity);

//Printando valores no monitor serial
Serial.print("Temperature= ");
Serial.println(temperature);
Serial.print("Humidity= ");
```

```
Serial.println(humidity);
cliente.stop(); //Closing the connection
}
else
{
// if you didn't get a connection to the server:
Serial.println("connection failed");
}
delay(5000);

if (ack == 0)
{
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temp:");
lcd.print(DHT.temperature);
lcd.setCursor(0, 1);
lcd.print("Humidity:");
lcd.print(DHT.humidity);
delay(500);

if (DHT.temperature >= T_threshold)  //CONTROLE DE TEMPERATURA
{
delay(500);
if (DHT.temperature >= T_threshold)
{
digitalWrite(bulb, LOW);
}
}
if (DHT.humidity >= H_threshold)
{
delay(500);
if (DHT.humidity >= H_threshold)
{
digitalWrite(vap, HIGH);
}
}
if (DHT.temperature < T_threshold)
{
delay(500);
if (DHT.temperature < T_threshold)
{
digitalWrite(bulb, HIGH);
}
}
if (DHT.humidity < H_threshold)
{
delay(500);
if (DHT.humidity < H_threshold)
{
```

```
digitalWrite(vap, LOW);
}
}
sec = sec + 1;   //CONTROLE DO SERVO
if (sec == 60)
{
sec = 0;
Min = Min + 1;
}
if (Min == 60)
{
Min = 0;
hrs = hrs + 1;
}
if (hrs == 4 && Min == 0 && sec == 0)
{Serial.println("  ROTACIONANDO PARA FRENTE  ");
for (pos = 45; pos <= 135; pos += 1)
{
motor.write(pos);
delay(25);
}
}
if (hrs == 8 && Min == 0 && sec == 0)
{Serial.println("  ROTACIONANDO PARA TRAS  ");
hrs = 0;
for (pos = 135; pos >= 45; pos -= 1)
{
motor.write(pos);
delay(25);
}
}
}
if (ack == 1)
{
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Sem dados do sensor."); //FALHA/DESLIGAR
lcd.setCursor(0, 1);
lcd.print("System Halted.");
digitalWrite(bulb, LOW);
digitalWrite(vap, LOW);
}
delay(500);
}
```