UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia de Sorocaba

Luis Henrique Garcia Gonçalves

Automação residencial modular:

automatizando uma residência de forma desacoplada e customizável

Sorocaba

2023

Luis Henrique Garcia Gonçalves

Automação residencial modular:

automatizando uma residência de forma desacoplada e customizável

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Everson Martins

Sorocaba

2023

G635a

Gonçalves, Luis Henrique Garcia
Automação residencial modular: : automatizando uma residência de forma desacoplada e customizável / Luis Henrique Garcia Gonçalves. -- Sorocaba, 2023
60 p. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Everson Martins

1. Internet das coisas. 2. Microcontroladores. 3. Automação residencial. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Automação residencial modular:

automatizando uma residência de forma desacoplada e customizável

Luis Henrique Garcia Gonçalves

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO
COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE
BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÂO

Prof. Dr. Everson Martins

Coordenador

BANCA EXAMINADORA:

Prof. Dr. Everson Martins

Orientador/UNESP - Campus de Sorocaba

Prof. Dr. IVANDO SEVERINO DINIZ

UNESP- Campus de Sorocaba

Prof. Dr. FELIX MAURICIO ESCALANTE ORTEGA

UNESP - Campus de Sorocaba

Novembro de 2023

AGRADECIMENTOS

Agradeço sinceramente à minha família pelo apoio incondicional e pela constante motivação ao longo desta jornada acadêmica. Suas palavras de encorajamento e amor foram fundamentais para que eu chegasse até aqui.

Quero expressar minha profunda gratidão a toda a comunidade da Unesp Sorocaba. Durante esses anos de estudo, aprendi não apenas com os livros e professores, mas também com cada colega e amigo que cruzou o meu caminho. As valiosas experiências e conhecimentos compartilhados tornaram a minha jornada acadêmica rica e significativa.

Um agradecimento especial ao Prof. Dr. Everson Martins, meu orientador no Projeto Final de Curso. Sua disponibilidade, paciência e expertise foram cruciais para o sucesso deste trabalho.

Agradeço a todos que, de alguma forma, contribuíram para a realização deste trabalho e para o meu crescimento pessoal e acadêmico. Suas influências foram fundamentais para o meu desenvolvimento, e por isso, sou eternamente grato.

RESUMO

Na atual era de dispositivos inteligentes e Internet das Coisas (IoT), os projetos de automação residencial estão em ascensão. A disseminação acelerada de tecnologias está familiarizando as pessoas com produtos "Smart" e criando uma demanda cada vez mais exigente. Empresas líderes como Amazon e Google estão investindo de forma proativa nesse mercado para manter sua posição de destaque na indústria tecnológica. Dispositivos como a Alexa, o Chromecast e lâmpadas controladas por aplicativos tornaram-se presença comum em lojas, destacando a crescente procura por soluções de automação residencial no cenário brasileiro. O foco deste trabalho é apresentar um produto modular de automação residencial, organizado em categorias que atendem a diversas necessidades, desde iluminação até agendamento de tarefas e segurança. A essência é estabelecer uma base sólida para a automação residencial através de um protótipo desenvolvido com o kit Arduino. O cerne desse protótipo é a incorporação de um servidor web no Arduino, permitindo uma interação intuitiva com os usuários. Em resumo, este projeto busca fornecer uma solução abrangente para as demandas de automação residencial, impulsionadas pelo uso estratégico de componentes como relés e sensores, aliados à capacidade de conectividade web. A interseção desses elementos permite explorar o potencial da automação residencial de maneira acessível e inovadora.

Palavras-chave: automação residencial; Arduino; IOT (Internet das Coisas); dispositivos inteligentes; conectividade web.

ABSTRACT

In the current era of smart devices and the Internet of Things (IoT), home automation projects are on the rise. The rapid proliferation of technologies is acquainting people with "Smart" products and creating an increasingly demanding market. Leading companies such as Amazon and Google are proactively investing in this field to maintain their prominent positions in the technology industry. Devices like Alexa, Chromecast, and app-controlled light bulbs have become common sights in stores, highlighting the growing demand for home automation solutions in the Brazilian landscape. The focus of this project is to present a modular home automation product, organized into categories that cater to diverse needs, ranging from lighting to scheduled tasks and security. The core objective is to establish a solid foundation for home automation through a prototype developed using the Arduino kit. At the heart of this prototype lies the incorporation of a web server within the Arduino, enabling intuitive interaction with users. In summary, this project aims to provide a comprehensive solution for home automation demands, driven by the strategic use of components such as relays and sensors, combined with web connectivity capabilities. The intersection of these elements allows for the exploration of home automation potential in an accessible and innovative manner.

Keywords: home automation; Arduino; IOT (Internet of Things); smart devices; web connectivity.

SUMÁRIO

1 INTRODUÇÃO 7
2 MATERIAIS E MÉTODOS 11
3 DESENVOLVIMENTO 18
3.1 Software 18
3.2 Hardware 28
3.3 Testes com Usuários 31
4 RESULTADOS E DISCUSSÕES 32
5 CONCLUSÃO 33
REFERÊNCIAS 34
APÊNDICE A – Código do Arduino Mega 2560 35
APÊNDICE B – Código do Arduino UNO 45
APÊNDICE C – Página 401 Error 46
APÊNDICE D – Página 404 Error 47
APÊNDICE E – Página de Login 48
APÊNDICE F – Página Inicial 50
APÊNDICE G – Página Painel de controle 51
APÊNDICE H – Código de definição de estilos 57

1 INTRODUÇÃO

A evolução constante das residências modernas tem estimulado um interesse crescente na automação residencial como meio de otimizar a comodidade, segurança e eficiência. No entanto, essa busca por sistemas aprimorados enfrenta um desafio central: criar soluções acessíveis em termos de custo, ao mesmo tempo em que oferecem um leque versátil de recursos, adaptáveis às necessidades individuais dos usuários. Nesse contexto, este estudo visa aprofundar a investigação sobre o desenvolvimento de um produto de automação residencial modular, que apresente um custo acessível sem comprometer a flexibilidade de funcionalidades.

O questionamento subjacente é: Como é possível abordar a concepção de um produto de automação residencial composto por módulos, que seja financeiramente viável e, ao mesmo tempo, permita a adaptação de suas funcionalidades? Para responder a esta questão é necessário observar o histórico de desenvolvimento das tecnologias de automação residencial para então abordar esta questão de forma sólida.

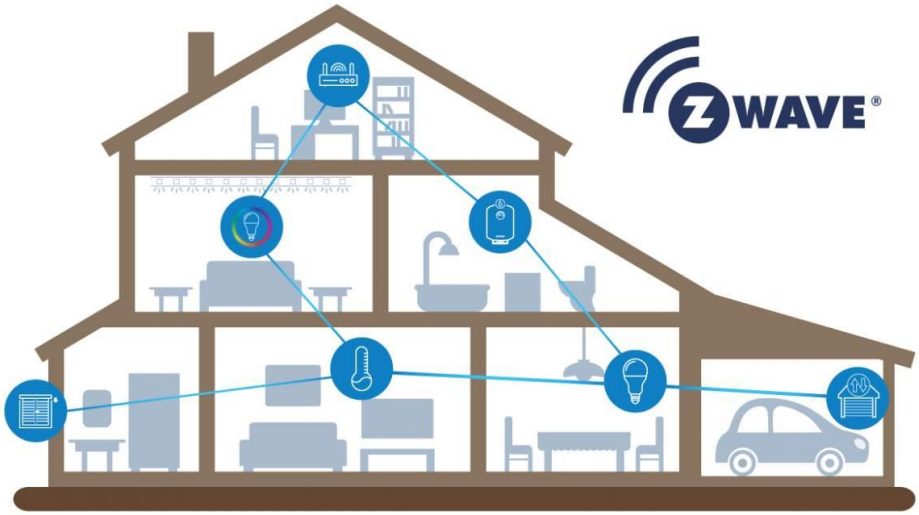
No início do desenvolvimento da automação residencial, em 1975, pesquisadores da Pico Eletronics desenvolveram o protocolo X10[1] que permitia o controle remoto de dispositivos elétricos em casa. Eles estabeleceram os fundamentos para a ideia de que a automação residencial poderia aumentar a comodidade e a eficiência energética em residências. A figura 1 apresenta o módulo x10.

Figura 1 - Módulo x10.

Fonte: Cava (2021)[1].

Alguns anos depois, a evolução da automação residencial foi impulsionada pelo desenvolvimento de protocolos de comunicação confiáveis, como o Z-Wave[2] em 2001 e o Zigbee[3] em 2003. Essas tecnologias permitiram a interconexão de dispositivos em redes domésticas, estabelecendo uma base sólida para a criação de sistemas abrangentes de automação residencial. A figura 2 apresenta de forma simplificada o protocolo z-wave.

Figura 2 - Ilustração do protocolo Z-Wave.



Fonte: Qubino[4].

Nos últimos anos, a automação residencial se beneficiou significativamente do uso de algoritmos de inteligência artificial (IA) e aprendizado de máquina (AM). Demonstrou-se como algoritmos de IA podem ser empregados para otimizar o consumo de energia em residências, adaptando-se automaticamente aos padrões de uso[5].

A crescente complexidade dos sistemas de automação residencial também trouxe à tona questões importantes de segurança e privacidade. Destacou-se a necessidade de abordar vulnerabilidades e estabelecer práticas seguras no design e implementação de sistemas de automação residencial[6].

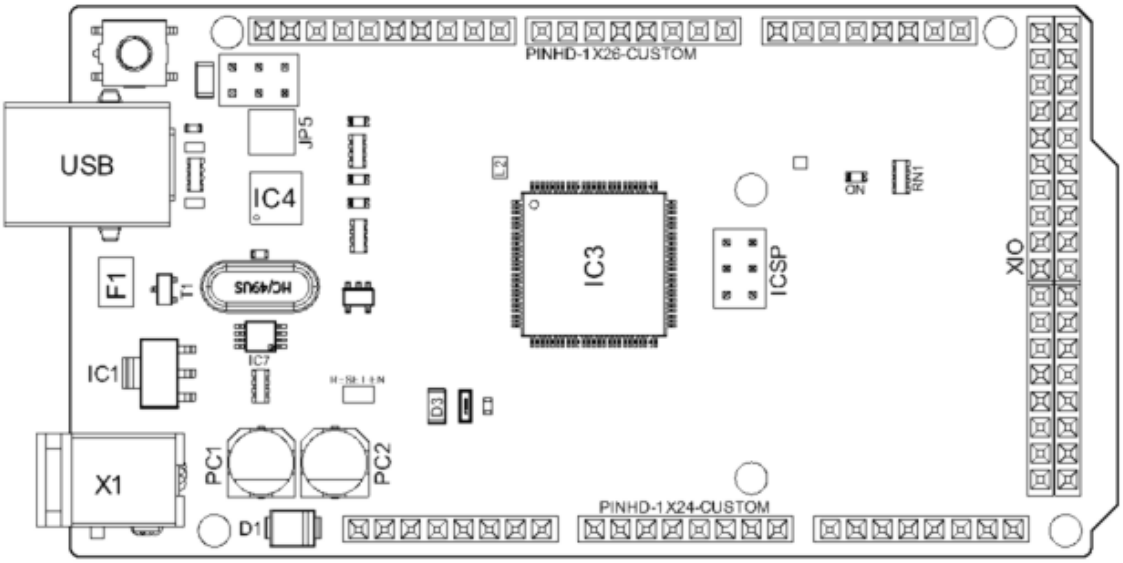
A integração de assistentes virtuais, como Amazon Alexa e Google Assistant, está se tornando cada vez mais comum em residências automatizadas. Analisou-se como esses sistemas podem ser usados para controlar uma ampla gama de dispositivos e melhorar a usabilidade da automação residencial.

No cerne deste projeto, está a criação de um protótipo funcional de uma unidade central de processamento. Essa unidade será responsável por gerenciar uma variedade de sensores e relês, executando comandos acionados pelos usuários por meio de uma interface de página web

integrada. Essa interface, hospedada na própria unidade central, permitirá o controle local através da rede doméstica e oferecerá informações em tempo real sobre as diversas automações implementadas na residência.

Dentre os componentes essenciais utilizados, destaca-se o Arduino[6]. Trata-se de uma plataforma de prototipagem eletrônica de código aberto, que fornece uma base sólida para o desenvolvimento de sistemas interativos. O Arduino, com sua flexibilidade e extensa comunidade de desenvolvedores, possibilita a programação e controle de dispositivos eletrônicos de maneira eficiente e intuitiva. Ele servirá como o "cérebro" da unidade central, coordenando as operações dos sensores e relês para executar as automações desejadas. A figura 3 apresenta o Arduino Mega e indicações de seus principais pinos.

Figura 3 - Arduino Mega 2560 com demarcações de pinagem.



Arduino MEGA Top View

Ref.	Description	Ref.	Description
USB	USB B Connector	F1	Chip Capacitor
IC1	5V Linear Regulator	X1	Power Jack Connector
JP5	Plated Holes	IC4	ATmega16U2 chip
PC1	Electrolytic Alumninum Capacitor	PC2	Electrolytic Alumninum Capacitor
D1	General Purpose Rectifier	D3	General Purpose Diode
L2	Fixed Inductor	IC3	ATmega2560 chip
ICSP	Connector Header	ON	Green LED
RN1	Resistor Array	XIO	Connector

Fonte: Arduino[7].

Além disso, serão integrados sensores como os de luz e temperatura. O sensor de luz, por exemplo, pode ser utilizado para ajustar automaticamente a iluminação com base nas condições de luminosidade ambiente, promovendo economia de energia. Já o sensor de temperatura pode contribuir para o controle do sistema de climatização, mantendo o ambiente em uma faixa de temperatura confortável e eficiente.

Os relês, por sua vez, desempenham um papel crucial na automação residencial, permitindo o controle de dispositivos elétricos, como lâmpadas, tomadas e equipamentos. Eles podem ser acionados remotamente para ligar ou desligar aparelhos, bem como para criar cenários personalizados que se adaptem às necessidades dos usuários.

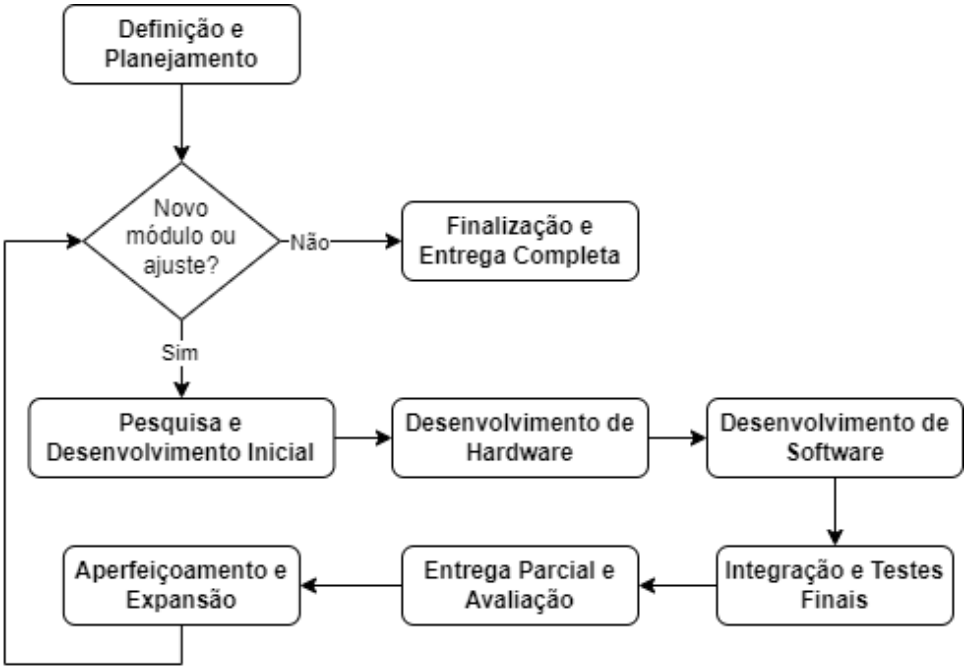
Dentro dos próximos capítulos serão abordados os detalhes que envolvem o desenvolvimento, funcionamento e validação do protótipo nos pilares de Hardware, todos os componentes físicos do projeto, Software, todos os códigos desenvolvidos para gerenciar a interação dos usuários com o Hardware e, para concluir o projeto, serão apresentados discussões e resultados de testes feitos com base no protótipo final.

2 MATERIAIS E MÉTODOS

Neste trabalho, é proposta a investigação e o desenvolvimento de um produto de automação residencial modular com foco na harmonização entre acessibilidade financeira e flexibilidade de funcionalidades. Para alcançar esse objetivo, será abordada a criação de um protótipo funcional de uma unidade central de processamento, que será responsável por gerenciar sensores e relês, permitindo comandos personalizados por meio de uma interface web integrada. A utilização do Arduino e de sensores como os de luz e temperatura desempenhará um papel fundamental na concretização desse sistema adaptável, capaz de atender às diversas necessidades dos usuários modernos, promovendo eficiência, segurança e comodidade nas residências.

A metodologia para desenvolver cada módulo do projeto (iluminação, temperatura, segurança e agendamento de tarefas) é baseada em entregas parciais e em pesquisa e desenvolvimento tanto do software quanto do hardware e pode ser dividida em oito etapas. A figura 4 apresenta o fluxograma desta metodologia.

Figura 4 - Fluxograma da metodologia implementada para o desenvolvimento do projeto.



Fonte: Autoria própria.

Como visto na figura 4, cada módulo segue uma sequência de desenvolvimento até ser validado e entregue. A seguir é apresentado um descritivo de cada etapa que compõem a metodologia, de forma a complementar o fluxograma da figura 4:

Etapa 1: Definição e Planejamento

1.1. **Definição de Objetivos e Requisitos:** Identifica-se os requisitos específicos para cada módulo, considerando as necessidades dos usuários e os recursos disponíveis.

1.2. **Levantamento de Recursos:** Lista-se os recursos necessários, como sensores, atuadores, componentes eletrônicos, plataformas de desenvolvimento etc.

1.3. **Cronograma:** Elabora-se um cronograma de desenvolvimento com marcos temporais e entregas parciais.

Etapa 2: Pesquisa e Desenvolvimento Inicial

2.1. **Pesquisa de Mercado:** Realiza-se pesquisas para identificar as melhores práticas, tendências e produtos similares no mercado de automação residencial.

2.2. **Estudo de Viabilidade:** Avalia-se a viabilidade técnica e econômica de cada módulo e ajustam-se os requisitos conforme necessário.

2.3. **Seleção de Componentes:** Escolhe-se os sensores, atuadores e outros componentes necessários para cada módulo com base na pesquisa realizada.

Etapa 3: Desenvolvimento de Hardware

3.1. **Prototipagem:** Constroem-se protótipos dos módulos de hardware para testar e validar a funcionalidade.

3.2. **Integração de Sensores e Atuadores:** Integram-se os sensores de luz, temperatura e relês aos módulos de hardware, garantindo que funcionem em conjunto.

Etapa 4: Desenvolvimento de Software

4.1. **Programação do Arduino:** Desenvolve-se o código para o Arduino que controlará as operações dos módulos de hardware.

4.2. **Interface de Página Web:** Cria-se a interface de página web que permitirá aos usuários controlarem e monitorar os módulos de automação residencial.

4.3. **Testes de Software:** Realizam-se testes de software para garantir que todas as funcionalidades estejam funcionando corretamente.

Etapa 5: Integração e Testes Finais

5.1. **Integração de Módulos:** Integram-se todos os módulos (iluminação, temperatura, segurança e agendamento de tarefas) em uma única unidade central de processamento.

5.2. **Testes de Integração:** Realizam-se testes para garantir que os módulos funcionem de maneira coordenada e que a interface web esteja sincronizada com o hardware.

Etapa 6: Entrega Parcial e Avaliação

6.1. **Entrega do Protótipo Parcial:** Entrega-se um protótipo parcial que inclua pelo menos um dos módulos funcionais (por exemplo, iluminação) para avaliação e feedback dos usuários.

6.2. **Avaliação de Usabilidade:** Realizam-se testes com usuários para avaliar a usabilidade do protótipo parcial e coletar feedback.

Etapa 7: Aperfeiçoamento e Expansão

7.1. **Iteração:** Com base no feedback dos usuários, fazem-se melhorias no hardware e no software, se necessário.

7.2. **Expansão:** Repete-se o processo de desenvolvimento para os módulos restantes (temperatura, segurança e agendamento de tarefas).

Etapa 8: Finalização e Entrega Completa

8.1. **Testes Finais:** Realizam-se testes abrangentes em todos os módulos integrados para garantir a estabilidade e a segurança do sistema.

8.2. **Documentação:** Prepara-se documentação completa do projeto, relatórios parcial e final, artigo acadêmico e apresentação.

8.3. **Entrega Completa:** Entrega-se o produto de automação residencial modular aos usuários.

Uma metodologia ágil de entregas parciais demonstra sua eficiência no desenvolvimento de projetos, pois proporciona flexibilidade, adaptação contínua e maior envolvimento das partes interessadas ao longo do processo. Ao dividir o projeto em partes menores e entregáveis, os clientes têm a oportunidade de fornecer feedback constante, permitindo ajustes e melhorias ágeis, enquanto o desenvolvedor pode focar em prioridades reais. Essa abordagem promove a transparência, a colaboração e a rapidez, resultando em

entregas mais alinhadas com as necessidades do cliente e em um projeto mais eficaz como um todo.

Dentre os materiais usados no projeto, além do Arduino que serve como orquestrador de todo o sistema, também são usados sensores para coletar dados e atuadores para executar as funcionalidades programadas. O quadro 1 apresenta uma lista de materiais usados no projeto, especificando quantidade e descrição para cada componente.

Quadro 1 - Materiais usados no projeto.

Material	Quantidade	Descrição
Arduino Mega	1	O Arduino Mega é uma placa de desenvolvimento baseada no microcontrolador ATmega2560. Destaca-se por sua ampla gama de pinos de I/O (entradas/saídas), múltiplas portas seriais, interfaces USB e uma grande capacidade de memória. Projetado para aplicações que exigem maior poder de processamento, o Arduino Mega é ideal para projetos eletrônicos complexos.
Arduino UNO	1	O Arduino Uno é uma placa de microcontrolador baseada no ATmega328P. Equipada com 14 pinos de I/O, possui uma arquitetura simplificada para projetos de menor complexidade. A presença de uma interface USB facilita a programação e a comunicação com outros dispositivos.
Bateria 12 V	1	Uma bateria de 11.1V e 1800mAh de 3 células. É uma bateria de íon de lítio com uma voltagem total de 11.1 volts, uma capacidade de 1800 miliampere-horas e composta por três células conectadas em série.
Ethernet Shield	1	O Ethernet Shield do Arduino é um módulo que permite que o Arduino se conecte a uma rede Ethernet, possibilitando a comunicação com outros dispositivos via Internet. Ele possui um conector para cabo de rede e facilita a criação de projetos que exigem conexão com a web. O shield é encaixado na placa Arduino e inclui recursos para facilitar a implementação de protocolos de comunicação via Ethernet.

Display Shield	1	O Display Shield do Arduino é um módulo que adiciona uma tela ao Arduino, permitindo a exibição de informações de maneira visual. Esse shield geralmente inclui um display integrado, como um LCD ou OLED, e botões para interação. Ele simplifica a implementação de interfaces gráficas em projetos, como mostradores de dados, menus interativos ou painéis de controle.
Breadboard	1	Uma breadboard, ou placa de prototipagem, é uma placa plástica com furos conectados eletricamente. Ela é usada para montar e testar circuitos eletrônicos temporários sem a necessidade de soldagem. Os furos são organizados em linhas e colunas, permitindo a conexão fácil de componentes eletrônicos, como resistores, LEDs e fios, por meio de pinos ou jumpers. A breadboard é uma ferramenta essencial para prototipar e experimentar com diferentes configurações de circuitos antes de criar uma versão permanente.
Módulo RFID	1	O módulo RFID do Arduino é um dispositivo que utiliza a tecnologia de identificação por radiofrequência (RFID) para permitir a leitura de informações em tags ou cartões RFID.
Sensor Ultrassónico	1	Um sensor ultrassônico é um dispositivo utilizado para medir distâncias através do princípio da ecolocalização. Ele emite pulsos de ondas ultrassônicas inaudíveis ao ouvido humano e mede o tempo que esses pulsos levam para retornar após atingirem um objeto. Esse tipo de sensor é comumente usado em projetos de robótica, automação e sistemas de detecção de proximidade.
RTC	1	Um módulo RTC (Real-Time Clock) é um dispositivo que permite ao microcontrolador manter o controle preciso do tempo, mesmo quando desligado. Geralmente equipado com um chip de relógio em tempo real e uma bateria de

		backup, o módulo RTC fornece informações como hora, minutos, segundos, dia, mês e ano.
Sensor de Luz (LDR)	1	Um sensor de luz LDR (Light Dependent Resistor), é um componente eletrônico cuja resistência varia em resposta à intensidade da luz incidente sobre ele. Quando exposto à luz, a resistência diminui; em ambientes escuros, a resistência aumenta.
Sensor de temperatura (LM35)	1	O sensor de temperatura LM35 é um dispositivo amplamente utilizado para medir a temperatura em projetos eletrônicos. Ele fornece uma saída de voltagem proporcional à temperatura em graus Celsius.
Relê	6	O relê é um dispositivo eletromecânico utilizado para controlar circuitos elétricos de alta potência por meio de um sinal de baixa potência. Ele opera como um interruptor controlado remotamente, permitindo que um circuito seja aberto ou fechado utilizando um sinal elétrico. Um relê comum consiste em uma bobina e contatos móveis que, quando energizados, abrem ou fecham o circuito principal. Isso é útil em projetos de automação, controle de dispositivos elétricos e sistemas onde é necessário isolar eletricamente componentes de diferentes potências.
Trinco elétrico	1	Um trinco elétrico é um dispositivo eletromecânico usado para controlar o acesso ou travar portas de forma elétrica. Ele é frequentemente utilizado em sistemas de segurança, como fechaduras automáticas controladas remotamente. Quando recebe um sinal elétrico, geralmente de um sistema de controle, o trinco elétrico é acionado para liberar ou travar a porta.
Válvula solenoide	1	Uma válvula solenoide é um dispositivo eletromecânico usado para controlar o fluxo de fluidos, como água, gás ou óleo, em sistemas automáticos. Ela consiste em uma bobina e um êmbolo que, quando a bobina é energizada com eletricidade, cria um campo magnético, movendo o

		êmbolo e abrindo ou fechando o caminho para o fluido. Elas são acionadas por um sinal elétrico.
Emissor IR	1	Um emissor infravermelho (IR) é um componente eletrônico que emite luz infravermelha invisível ao olho humano. Quando energizado, o emissor IR gera pulsos de luz infravermelha que podem ser interpretados por receptores IR. Essa tecnologia é amplamente empregada para transmitir sinais de controle remoto de curta distância, sendo comum em projetos de automação residencial e eletrônicos.
Cartão SD 4 GB	1	Um cartão SD é um dispositivo de armazenamento de dados removível que utiliza tecnologia de memória flash. Com uma capacidade de 4 gigabytes, o cartão SD oferece espaço para armazenar dados digitais, como fotos, vídeos, documentos e outros arquivos. Ele é compatível com uma variedade de eletrônicos que suportam cartões SD. O tamanho compacto e a capacidade moderada tornam esses cartões populares para expansão de armazenamento em dispositivos.
Cabos Jumpers	~	Cabos finos, resistentes e isolados. Adequados para projetos de protótipos compactos.

Fonte: Autoria própria.

3 DESENVOLVIMENTO

Durante o desenvolvimento deste projeto, três pilares foram trabalhados para chegar em um produto, software, hardware e testes com usuários. A investigação minuciosa desses elementos é fundamental para compreender a eficácia e o desempenho do sistema, bem como suas interações com os usuários.

3.1 Software

O código do Arduino Mega (Apêndice A) é um programa escrito em linguagem C++ que controla uma série de dispositivos e sensores, bem como serve uma página da web para interação com esses dispositivos. Ele utiliza diversas bibliotecas, como LiquidCrystal, IRremote, Ethernet, RTClib, Wire e SD, para gerenciar componentes como um display LCD, um transmissor infravermelho, sensores de temperatura e distância, além de controle de iluminação e portões.

O código começa incluindo várias bibliotecas que serão usadas ao longo do programa, como LiquidCrystal, IRremote, Ethernet, RTClib, Wire, SPI, e SD. Em seguida, são definidas algumas constantes e variáveis globais, incluindo endereços IP, pinos de saída e variáveis de controle. A figura 5 mostra a inclusão destes pacotes.

Figura 5 - Inclusão dos pacotes utilizados no código do Arduino Mega.

```
#include <LiquidCrystal.h>
#include <IRremote.hpp>
#include <Ethernet.h>
#include <RTClib.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
```

Fonte: Autoria própria.

Já na figura 6, são definidas constantes que representam endereços IP, tamanhos de buffers e pinos utilizados pelo sistema. Por exemplo, REQ_BUF_SZ define o tamanho do buffer para solicitações HTTP, e ilum_a, ilum_b etc., representam os pinos usados para controlar iluminação e outros dispositivos.

Figura 6 - Constantes utilizadas no código do Arduino Mega.

```
#define IP IPAddress(192, 168, 0, 169)
#define REQ_BUF_SZ 40
#define ilum_a 22
#define ilum_b 24
#define ilum_c 26
#define ilum_d 28
```

```
#define auto_r 30
#define sec_gate 32
#define gateKey 34
#define LDRPin A3
#define tempPin A2
#define trigPin 35
#define echoPin 36
```

Fonte: Autoria própria.

Na figura 7, são definidas variáveis globais que controlam o estado do sistema e armazenam informações importantes. Por exemplo, autoIlum indica se a iluminação está em modo automático, ambTemp armazena a temperatura ambiente, e server é um objeto que representa o servidor web.

Figura 7 - Variáveis globais utilizadas no código do Mega.

```
bool autoIlum = true;
bool autoRega = true;
bool secLamp = true;
bool secGate = true;
float ambTemp = 0;

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);
File webFile;
char HTTP_req[REQ_BUF_SZ] = { 0 };
byte req_index = 0;
bool root = false;
```

Fonte: Autoria própria.

Na figura 8, são criados objetos para controlar componentes físicos, como o display LCD (lcd), o transmissor infravermelho (transmissorIR) e o relógio em tempo real (rtc).

Figura 8 - Configuração de componentes conectados ao Arduino Mega.

```
LiquidCrystal lcd(8, 9, 33, 5, 6, 7);
const int backLight = 31;

IRsend transmissorIR;
RTC_DS3231 rtc;
```

Fonte: Autoria própria.

A figura 9 presenta uma lista de funções presentes no código do Arduino Mega. Essas funções auxiliares executam tarefas específicas no programa, como manipulação de strings, controle de dispositivos, leitura de sensores e processamento de páginas web.

Figura 9 - Funções do código do Arduino Mega.

```
void strClear(char *str, char len) { ... }
byte strContains(char *str, char *sfind) { ... }
void ilumHandler(byte output) { ... }
void tempHandler(int temp) { ... }
void autoHandler(byte output) { ... }
void secHandler(byte output) { ... }
void autoWatch() { ... }
int get_distance() { ... }
void display_print(char text_first_line[16], char text_second_line[16]) {...}
void sd_test() { ... }
void webPage_print(EthernetClient client) { ... }
```

Fonte: Autoria própria.

A seguir, segue um descritivo de cada função auxiliar, levando em consideração seu papel dentro do código e contribuição para o funcionamento do projeto como um todo:

- **Função strClear**: Esta função é responsável por limpar uma cadeia de caracteres (string) definida por um array de caracteres. É usada para redefinir o buffer HTTP_req.
- **Função strContains**: Esta função verifica se uma string contém outra string. É usada para analisar a solicitação HTTP recebida e determinar qual ação deve ser tomada com base nessa solicitação.
- **Funções ilumHandler, tempHandler, autoHandler, secHandler**: Essas funções são responsáveis por controlar dispositivos de iluminação, temperatura, automação e segurança. Elas recebem um valor como entrada e tomam decisões com base nesse valor para controlar os dispositivos correspondentes.
- **Função autoWatch**: Esta função monitora automaticamente o ambiente e faz ajustes nos dispositivos de iluminação, automação e segurança com base na hora atual e em outras condições.
- **Função get_distance**: Esta função mede a distância a partir de um sensor ultrassônico e retorna o valor em centímetros. É usada para detectar objetos próximos e controlar a iluminação de segurança.
- **Função display_print**: Esta função controla um display LCD para exibir informações relevantes. Ela recebe duas strings como entrada e exibe uma em cada linha do display.
- **Função sd_test**: Esta função inicializa um cartão SD e imprime mensagens de status no monitor serial. É usada para verificar se o cartão SD está funcionando corretamente.

- **Função webPage_print**: Esta função lida com solicitações HTTP recebidas de um cliente Ethernet. Com base na solicitação, ela serve páginas da web, controla dispositivos ou fornece informações como temperatura ambiente e status do portão.
- **Configuração Inicial (setup)**: No bloco **setup**, as configurações iniciais são realizadas, como a inicialização de pinos, a definição de estados iniciais para os dispositivos e a configuração do RTC (Relógio de Tempo Real). Além disso, a conexão Ethernet é estabelecida, o servidor web é iniciado e o cartão SD é testado.
- **Loop Principal (loop)**: O loop principal do programa verifica se há clientes Ethernet conectados e chama a função **webPage_print** quando um cliente está disponível. Caso contrário, a função **autoWatch** é chamada para monitorar o ambiente e controlar os dispositivos automaticamente.

O código do Arduino Uno (Apêndice B) é um controle de acesso utilizando um leitor de RFID (Radio-Frequency Identification). O código está escrito na linguagem de programação C++ e utiliza a biblioteca "MFRC522" para a interação com o leitor RFID. Usando um pino digital, o Arduino Uno se comunica com o Mega, indicando se houve a detecção de um cartão ou tag RFID. Com esta comunicação, o Arduino Mega pode abrir ou fechar o trinco elétrico.

O programa começa com a inclusão das bibliotecas necessárias: Wire.h, SPI.h e MFRC522.h. Essas bibliotecas fornecem as funções e definições necessárias para a comunicação com o leitor RFID. Em seguida, algumas constantes e variáveis são definidas, o SS_PIN é definido como 10 e RST_PIN como 9. Essas constantes representam os pinos utilizados para a comunicação SPI com o leitor RFID. O MFRC522 rfid(SS_PIN, RST_PIN) cria uma instância do objeto MFRC522 com os pinos especificados.

A função setup() é responsável por configurar o ambiente do programa. Ela Inicializa a comunicação I2C (Wire.begin()), SPI (SPI.begin()), e o leitor RFID (rfid.PCD_Init()), em seguida, define o pino openGate (pino 8) como saída (OUTPUT) e define o estado inicial do portão como fechado (digitalWrite(openGate, LOW)).

A função loop() é onde o programa principal é executado. Neste caso, ele chama a função leituraRfid() repetidamente. A função leituraRfid() é responsável por ler os cartões RFID e controlar a abertura e fechamento do portão. Esta função verifica se um novo cartão RFID está presente e se conseguiu ler o número de série do cartão, a figura 10 apresenta o trecho do código em que isso é especificado.

Figura 10 - Condição para leitura do cartão ou tag RFID.

```
if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())return;
```

Fonte: Autoria própria.

A função então cria uma string vazia **strID** para armazenar o número de série do cartão e em seguida aciona um Loop que lê os 4 bytes do número de série do cartão e os converte em uma representação hexadecimal, a figura 11 mostra como este loop foi escrito no código da função.

Figura 11 - Loop para conversão de dados do cartão RFID.

```
for (byte i = 0; i < 4; i++) {
    strID +=
      (rfid.uid.uidByte[i] < 0x10 ? "0" : "") +
      String(rfid.uid.uidByte[i], HEX) + (i != 3 ? ":" : "");
  }
  strID.toUpperCase();
```

Fonte: Autoria própria.

Em seguida, a função verifica se o número de série do cartão corresponde a um dos números de série permitidos ("B3:42:0F:0F" ou "5A:6C:EA:5B"). Se corresponder e o estado do portão estiver fechado, abre o portão e atualiza o estado para aberto; caso contrário, fecha o portão e atualiza o estado para fechado. A figura 12 representa a parte condicional que realiza este processamento dentro do código.

Figura 12 - Condições para abrir e fechar o trinco elétrico.

```
if(strID.indexOf("B3:42:0F:0F")>=0||strID.indexOf("5A:6C:EA:5B")>=0){
  if (gateState == false) {
    digitalWrite(openGate, HIGH);
    gateState = true;
  } else {
    digitalWrite(openGate, LOW);
    gateState = false;
  }
```

Fonte: Autoria própria.

Com a leitura efetuada com sucesso e a decisão de abrir ou fechar o trinco concluída, a função aguarda 3 segundos (delay(3000)) antes de verificar outro cartão, finalizando a comunicação com o cartão RFID e limpando o estado de autenticação. A figura 13 mostra as linhas em que ocorrem estes processos.

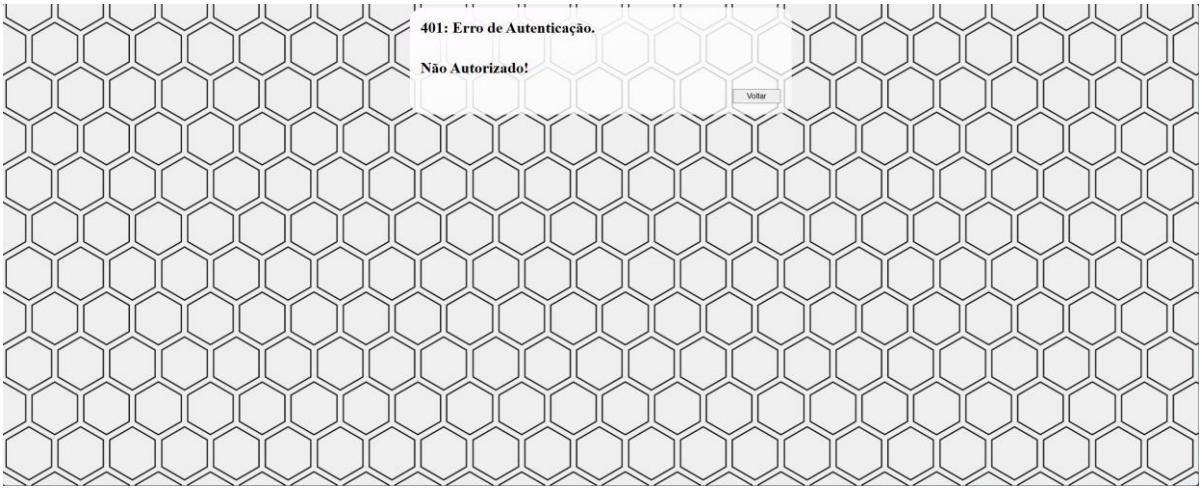
Figura 13 - Parte final da função de leitura do cartão RFID.

```
rfid.PICC_HaltA();
rfid.PCD_StopCrypto1();
```

Fonte: Autoria própria.

O código Página 401 Error (Apêndice C) é um documento HTML que exibe uma página de erro 401 (Erro de Autenticação) O objetivo desta página é servir como mensagem de erro em caso de algum usuário tentar acessar uma página do sistema em que não possui permissão ou sem estar devidamente autenticado. A página contém um único botão de input que está alinhado à direita. O botão possui o texto "Voltar". Quando o botão é clicado, ele redireciona o usuário para a página "login.htm". A figura 14 mostra como está página é apresentada.

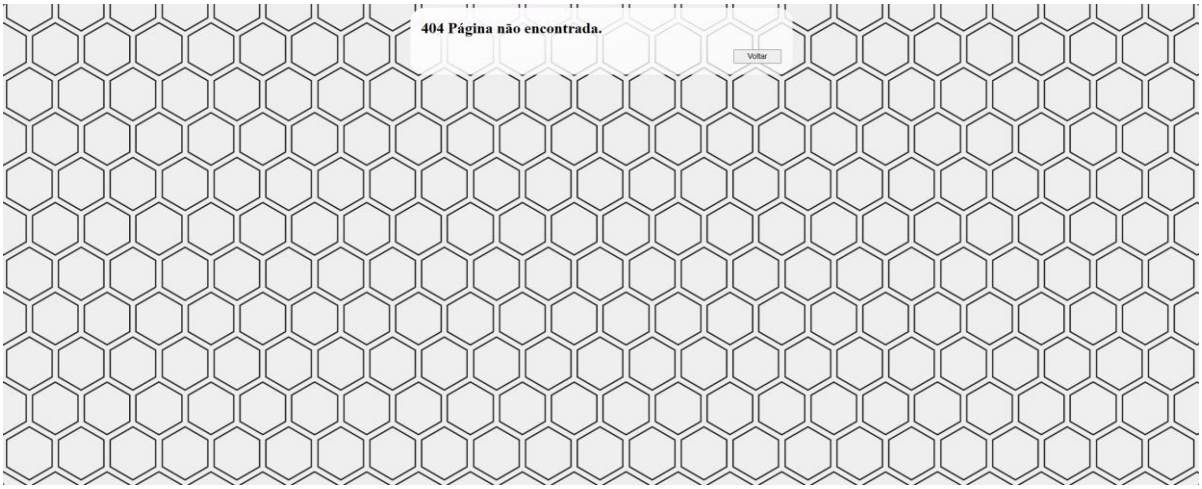
Figura 14 - Página 401.html.



Fonte: Autoria própria.

O código HTML Página 404 Error (Apêndice D) é um documento da web que define uma página com um título "404 Error". O erro 404 indica que a página procurada pelo usuário não existe ou não está disponível no sistema. No canto direito inferior do quadro da mensagem, há um botão de entrada (input) com o rótulo "Voltar". O botão possui um evento onclick que redireciona o usuário para a página "login.htm" quando clicado. A figura 15 apresenta a página.

Figura 15 - Página 404.html.

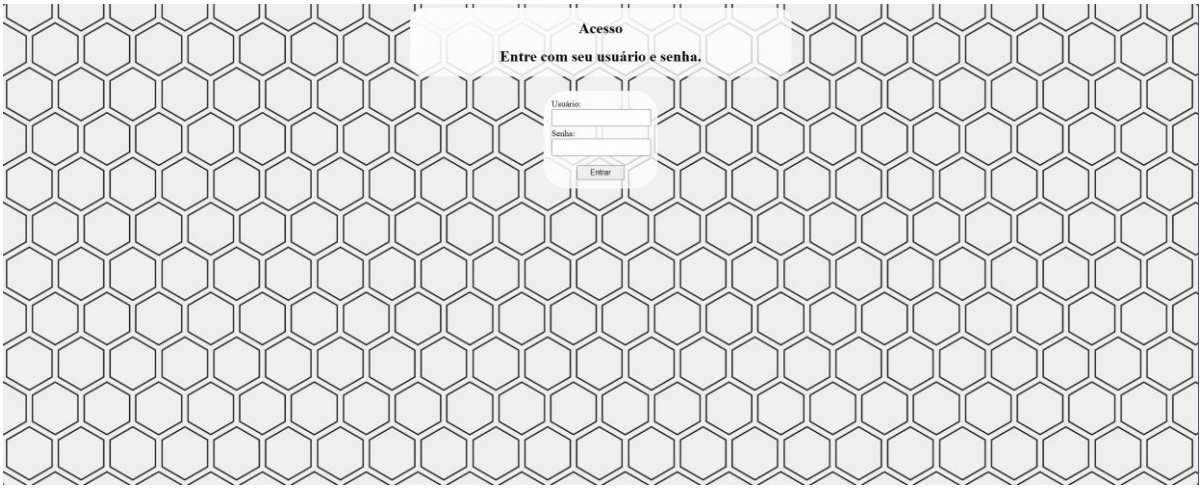


Fonte: Autoria própria.

O código HTML Página de Login (Apêndice E) apresentado é a página responsável por receber as informações de autenticação do usuário, usuário e senha, e em seguida enviá-las para que o servidor valide. Caso as informações estejam corretas, o usuário recebe acesso ao sistema.

Para funcionar desta forma, a página conta com um script que contém funções JavaScript. O JavaScript é usado para tornar a página interativa aos usuários através da função login(). A função login() é definida para processar o login do usuário quando o botão "Entrar" é clicado. Ela obtém os valores dos campos de usuário e senha, desabilita temporariamente o botão para evitar cliques repetidos e faz uma solicitação de fetch a um servidor. Dependendo da resposta do servidor, ele redireciona o usuário para 'main.htm' se o login for bem-sucedido ou limpa o campo de senha se o login falhar. A figura 16 apresenta a página de login.

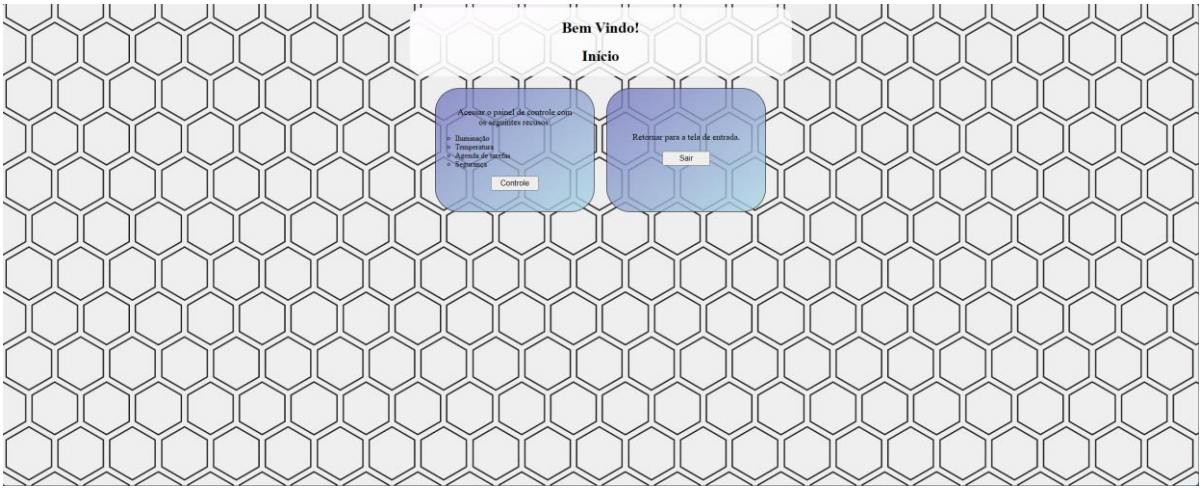
Figura 16 - Página login.html.



Fonte: Autoria própria.

O código HTML Página Inicial (Apêndice F) apresenta uma página de boas-vindas ao usuário já autenticado pelo sistema. A tela apresenta duas opções de navegação para o usuário, a primeira redireciona o usuário para a página "control.htm" e apresenta uma descrição das funções disponíveis na página de destino como uma prévia para o usuário. A segunda célula contém outro parágrafo com texto informativo e um segundo botão que, quando clicado, redireciona o usuário para a página "login.htm". desta forma a navegação entre as páginas se torna mais autoexplicativa e intuitiva. A figura 17 apresenta uma captura de tela da página inicial do sistema, apresentando todas as características descritas anteriormente, nelas é possível identificar as descrições de navegação posicionadas acima dos botões controle e sair.

Figura 17 - Página main.html.



Fonte: Autoria própria.

O código HTML Página Painel de controle (Apêndice G) é uma página da web que permite ao usuário controlar várias funções em um ambiente. A página é carregada com informações iniciais, como valores de saída para iluminação, temperatura e automação, bem como informações de temperatura ambiente.

A página possui quatro seções principais: Iluminação, Temperatura, Agenda de Tarefas e Segurança. Cada seção é representada por um formulário HTML. Na seção de Iluminação, o usuário pode ligar ou desligar várias lâmpadas em diferentes áreas, como Área Externa, Área Interna, Portão e Quarto, usando botões de alternância. A função JavaScript ilumHandle é usada para controlar o estado dessas lâmpadas e atualizar as saídas de iluminação.

Na seção de Temperatura, o usuário pode controlar o sistema de ar-condicionado (A/C) usando um botão de alternância "Ligar A/C". Também é possível ajustar a temperatura desejada usando botões "+" e "-", com a temperatura atual exibida na página. A função JavaScript tempHandle é usada para controlar essas operações e atualizar as saídas de temperatura.

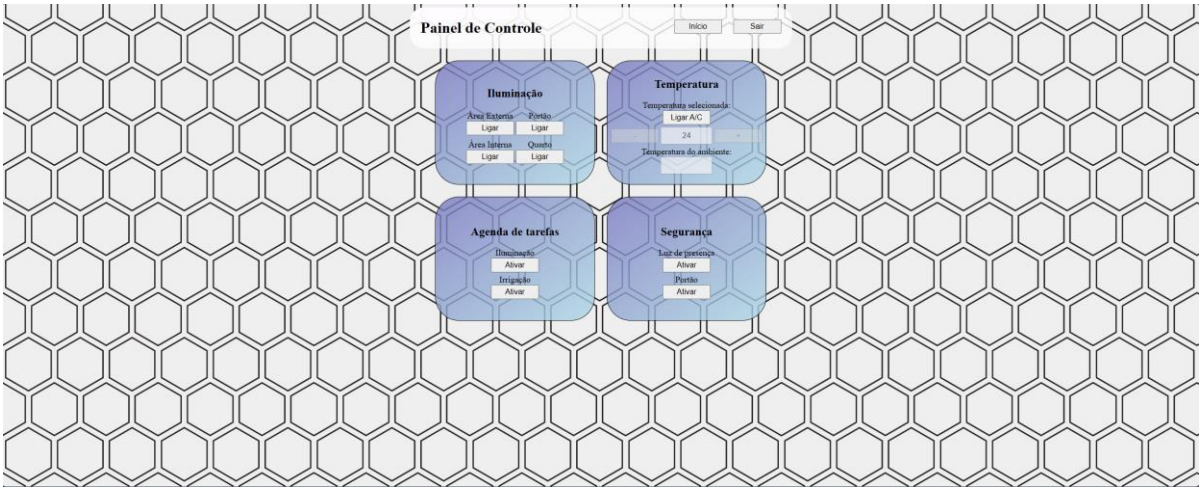
A seção de Agenda de Tarefas permite ao usuário ativar ou desativar a automação para iluminação e irrigação. Os botões "Ativar" e "Desativar" são usados para controlar essas tarefas, e a função JavaScript scheduleHandle gerencia essas operações.

Finalmente, na seção de Segurança, o usuário pode ativar ou desativar a luz de presença e o portão. Os botões "Ativar" e "Desativar" são usados para controlar essas funções, e a função JavaScript secHandle cuida disso.

Além disso, a página é carregada com valores iniciais e atualizações automáticas de temperatura ambiente. Há também um botão "Início" para voltar à página inicial e um botão "Sair" para sair do painel de controle.

Esse código HTML interage com funções JavaScript para fornecer um controle simples e interativo das funcionalidades do ambiente representado, sendo a principal janela para comunicar usuários com o controle do Hardware instalado na residência. Afigura 18 apresenta a página de controle.

Figura 18 - Página control.html.



Fonte: Autoria própria.

O código de definição de estilos (Apêndice H) define a estilização de elementos HTML usando CSS. O código é estruturado em regras que se aplicam a diferentes tipos de elementos HTML.

Para o elemento body, a cor de fundo é definida como uma imagem de fundo que está localizada no arquivo "back.jpg", elementos input, são definidas propriedades como largura (width), altura (height), tamanho da fonte (font-size), espaço inferior (margin-bottom), e alinhamento de texto (text-align), elementos button têm propriedades semelhantes às dos elementos input, como largura, altura e tamanho da fonte elementos label têm o tamanho da fonte definido como grande (font-size: large), elementos p têm tamanho de fonte grande, largura, altura e margem centralizada, elementos ul têm o estilo de lista definido como "circle" e o alinhamento do texto definido como esquerda.

Para a classe .controlMenuTable, é aplicado um estilo específico. A margem é centralizada, a cor de fundo é uma imagem de gradiente linear, e o canto é arredondado. A largura é definida como 800 pixels.

Para a classe .controlTable, a margem é centralizada, e o espaçamento entre as células da tabela é definido como 25 pixels.

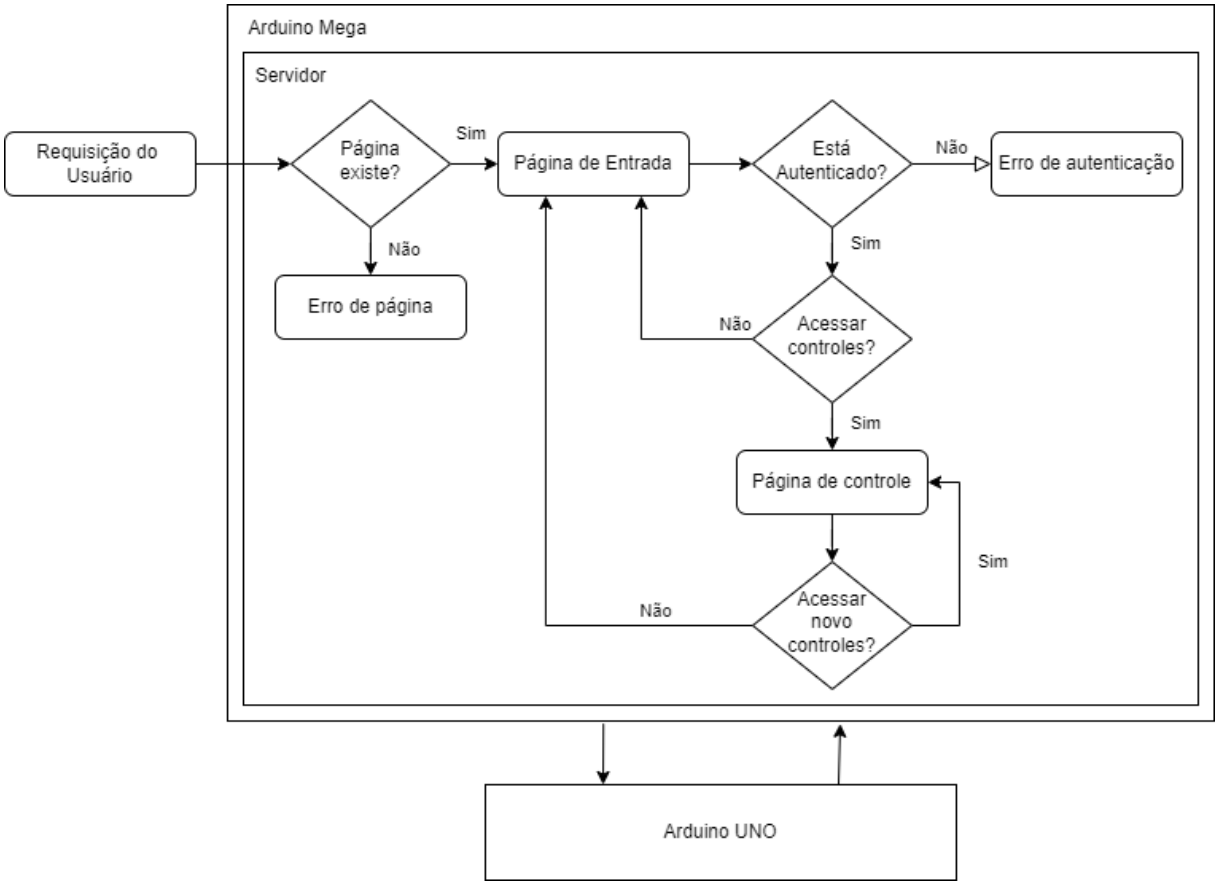
Para a classe .controlTd, as células da tabela têm uma borda sólida, largura, altura e alinhamento do texto definidos. Além disso, elas têm um canto arredondado e uma cor de fundo que é uma imagem de gradiente linear.

A classe .login aplica um estilo a um elemento com a classe "login". Define a margem, borda, preenchimento, cor de fundo e largura e altura, além de um canto arredondado.

Por fim, a classe .loginForm aplica um estilo a elementos com a classe "loginForm" e define uma margem de 5 pixels.

Em resumo, o código CSS apresentado é responsável por estilizar vários elementos HTML em uma página da web, aplicando larguras, alturas, tamanhos de fonte, margens, alinhamentos, cores de fundo e outros estilos específicos a cada tipo de elemento ou classe. Isso é feito para controlar a aparência e o layout da página da web.

Figura 19 - Fluxograma de relacionamento entre os códigos do projeto.

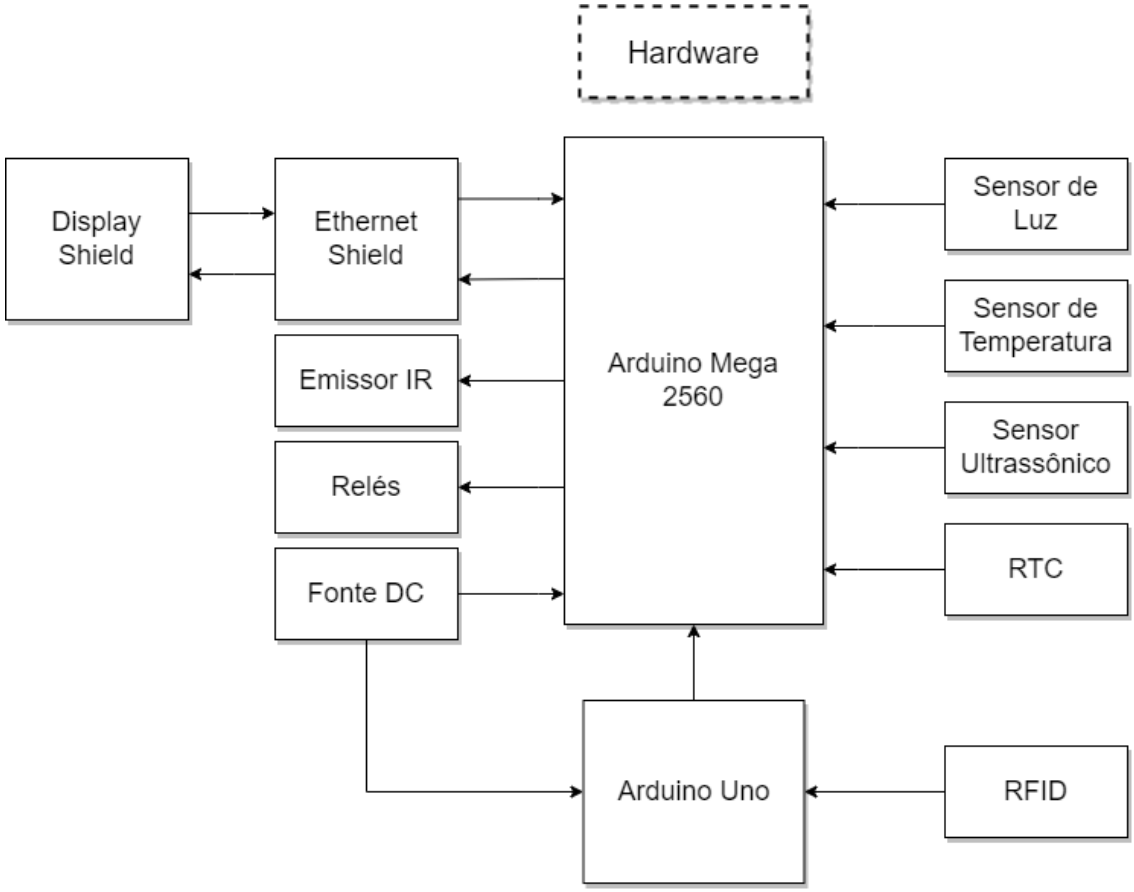


Fonte: Autoria própria.

3.2 Hardware

O presente projeto engloba uma variedade de componentes eletrônicos interconectados de forma sinérgica para criar um sistema multifuncional controlado por microcontroladores Arduino. O núcleo central dessa estrutura é o Arduino Mega, que desempenha um papel fundamental na coordenação e controle de todos os componentes. A figura 20 apresenta um diagrama do hardware desenvolvido, englobando cada componente chave descrito a seguir.

Figura 20 - Diagrama de blocos do hardware desenvolvido.



Fonte: Autoria própria.

O Arduino Mega é uma placa de desenvolvimento microcontrolada baseada no microcontrolador ATmega2560. Com suas múltiplas portas de entrada e saída digital e analógica, o Mega oferece a capacidade de conectar e controlar uma variedade de periféricos. No presente projeto, o Mega está conectado a seis relés, permitindo o controle independente de até seis dispositivos elétricos (sendo quatro lâmpadas, uma válvula hidráulica e um trinco elétrico de porta), bem como a um Arduino Uno, estabelecendo uma comunicação bidirecional entre os dois microcontroladores.

Os relés, por sua vez, são interruptores controlados eletronicamente que permitem a ativação ou desativação de dispositivos elétricos usando diferentes fontes e tensões, como lâmpadas ou eletrodomésticos. No contexto deste projeto, dois dos seis relés são utilizados para simular uma fonte de alimentação de 12 V, alimentada por uma bateria conectada à protoboard. Esta bateria, fornecendo energia para os relés, demonstra a versatilidade do sistema ao simular condições práticas de operação. Os outros quatro relés podem alimentar lâmpadas com tensões de 127 V ou 220 V dependendo das características elétricas da residência.

O Arduino Mega também se conecta a um sensor RFID (Radio-Frequency Identification), um dispositivo capaz de identificar e rastrear etiquetas RFID. Esta funcionalidade adiciona uma camada de segurança ao sistema, permitindo a autenticação baseada em identificação por radiofrequência. O módulo RFID é usado para autenticar a abertura e fechamento do trinco elétrico através do Arduino UNO.

A presença de um Real Time Clock (RTC) no circuito permite a marcação precisa do tempo, essencial para aplicações que requerem sincronização temporal. O RTC garante que eventos específicos ocorram em momentos pré-determinados, otimizando a eficiência do sistema. Este componente é usado no Arduino Mega para controle do agendamento de tarefas. As duas tarefas que são passiveis de agendamento são a irrigação e o acionamento de uma das lâmpadas.

Sensores adicionais, como o sensor de luz, o sensor de temperatura e o sensor ultrassônico, contribuem para a capacidade do sistema em reagir ao ambiente circundante. O sensor de luz monitora as condições de luminosidade e indica qual o momento ideal para acionamento de lâmpadas, enquanto o sensor de temperatura fornece informações sobre as condições térmicas do entorno, que são apresentadas ao usuário via página web. O sensor ultrassônico, por sua vez, é empregado para medir distâncias e é usado no módulo de segurança para verificar se ocorreu algum movimento dentro da área de atuação do sensor.

A comunicação de dados é possibilitada pelo uso de um Ethernet Shield, que permite a conexão do Arduino Mega à rede local. Esta funcionalidade possibilita a integração do sistema em redes mais amplas, bem como o controle remoto e monitoramento do sistema. É através desta conexão que usuários podem enviar comandos ao sistema.

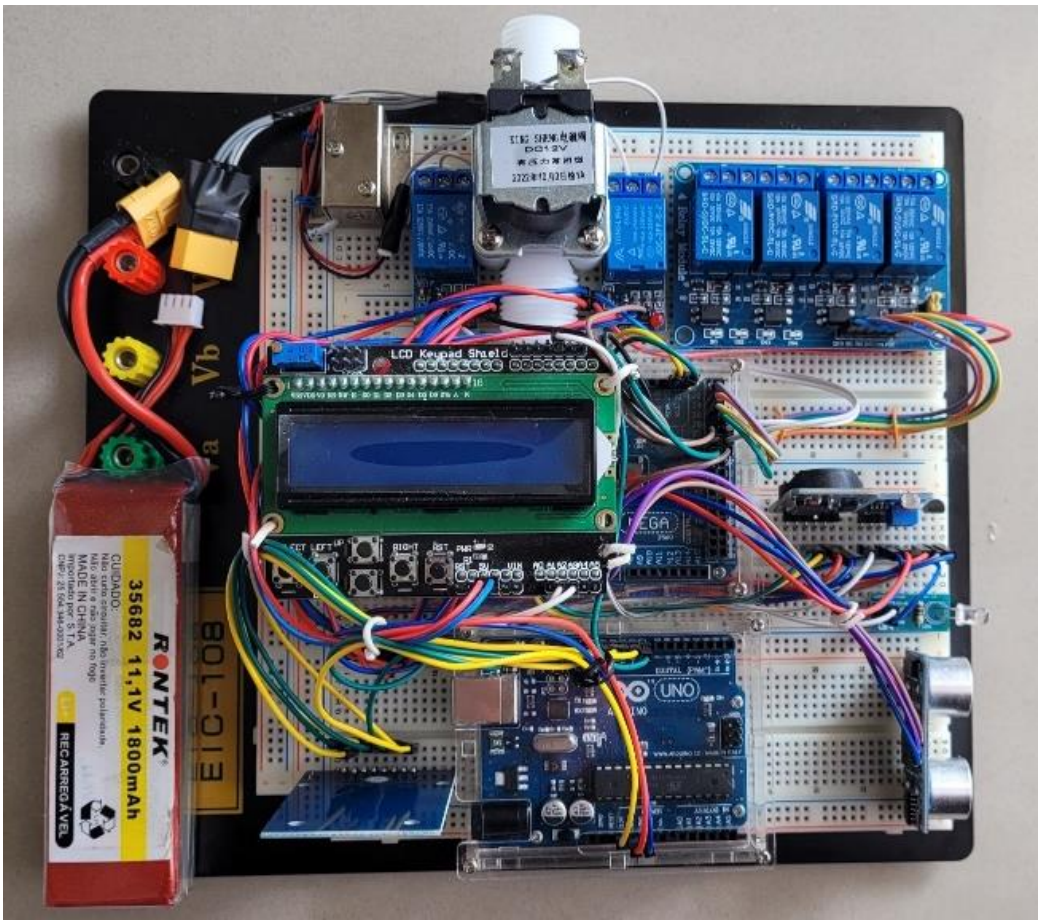
Além disso, um Display Shield é utilizado para a visualização de informações em tempo real. Este componente está fisicamente montado sobre o Ethernet Shield, otimizando o espaço na protoboard e proporcionando uma interface de usuário simplificada para interação com o

sistema. O Display apresenta o status do servidor que está em operação através do Ethernet Shield.

Por fim, o Arduino Uno, conectado ao Arduino Mega, atua como um módulo adicional para acionamento específico do relé que controla o trinco elétrico. Em caso de queda de energia ou internet, é possível manter o módulo funcionado usando uma bateria ou ‘no break’ e acionar o trinco sem o uso de internet ou energia da residência.

Em síntese, este projeto incorpora uma variedade de componentes eletrônicos interconectados de maneira coesa, demonstrando a versatilidade e a aplicabilidade dos microcontroladores Arduino em aplicações práticas. A integração harmoniosa desses componentes resulta em um sistema funcional e adaptável, capaz de atender a diversas demandas em ambientes controlados e automatizados. Desta forma o projeto se mostra modular e desacoplado. A figura 21 apresenta uma foto da parte física do projeto, nela é possível verificar que além do que foi descrito, há também um trinco elétrico e uma válvula hidráulica usados para testes práticos.

Figura 21 - Hardware do projeto.



Fonte: Autoria própria.

3.3 Testes com Usuários

Durante o desenvolvimento e teste deste projeto de automação residencial, foram coletadas críticas de três usuários.

O primeiro usuário destacou uma ambiguidade na tela de interface com o usuário, especificamente na página control.html, relacionada à parte da lâmpada que é acionada automaticamente de acordo com o horário. Como solução, foram implementadas travas que impedem o uso da lâmpada automatizada enquanto ela estiver sendo controlada automaticamente de acordo com o horário. Essas travas bloqueiam a capacidade de acionar ou desligar a lâmpada por meio do botão.

O segundo usuário observou que o servidor apresentava uma aparência muito simplista, que não condizia com os padrões de um projeto de automação residencial moderno. Para abordar esse problema, todas as páginas hospedadas no servidor foram redesenhadas para proporcionar uma experiência mais agradável aos usuários.

O terceiro usuário fez comentários sobre a disposição física da CPU, expressando a sensação de desorganização, mesmo considerando que o projeto era um protótipo. Como medida para resolver essa questão, os componentes foram realocados na protoboard, visando a organização dos cabos da maneira mais eficiente possível e a separação dos componentes de acordo com sua categoria e função.

4 RESULTADOS E DISCUSSÕES

O protótipo opera por meio da integração de medições realizadas pelos sensores com comandos remotos fornecidos pelos usuários. Para acessar as funcionalidades, o usuário realiza o login na plataforma utilizando seu nome de usuário e senha diretamente na página de entrada. Após o login bem-sucedido, é redirecionado para a página inicial do projeto, onde tem a opção de sair da plataforma ou acessar o menu de controle.

Ao optar por acessar o menu de controle, o usuário é direcionado para uma página que apresenta os quatro módulos do projeto: iluminação, temperatura, agenda de tarefas e segurança. No módulo de iluminação, há quatro botões representando diferentes áreas da residência: quarto, área interna, área externa e portão. Cada botão aciona um relé correspondente à iluminação específica descrita.

O módulo de temperatura oferece a possibilidade de ligar e desligar um ar-condicionado. Ao acionar o botão de ligar ar-condicionado, o aparelho é ligado à temperatura de 24°C, podendo ser ajustada através dos botões laterais + e -. A temperatura ambiente é exibida abaixo do indicador de temperatura do ar-condicionado e atualizada automaticamente a cada cinco segundos.

No módulo de agendamento de tarefas, o usuário pode programar a irrigação automática de um jardim. Ao ativar a irrigação automática, a válvula libera água por duas horas a partir do horário configurado. Também é possível programar o acionamento de uma lâmpada, mantendo-a ligada durante o período definido. Durante o controle automático, a lâmpada selecionada não pode ser controlada manualmente. Ao desativar o controle automático, a lâmpada mantém seu último estado.

No módulo de segurança, é possível acionar ou trancar um portão remotamente. Para evitar problemas de acesso em caso de falta de energia ou internet, o acionamento é realizado por meio de um módulo paralelo alimentado por uma fonte alternativa de energia. Em caso de falha no servidor, o usuário pode destrancar o portão utilizando uma tag RFID.

5 CONCLUSÃO

Para concluir este trabalho, foram apresentados detalhes importantes sobre o software e o hardware envolvidos em um sistema de automação residencial baseado em Arduino. Além disso, foram compartilhadas as experiências de testes com usuários e as melhorias implementadas com base no feedback recebido.

No que diz respeito ao software, destacou-se a complexidade do código Arduino Mega, que controla uma ampla gama de dispositivos e sensores, além de fornecer uma interface web para interação com o sistema. Foi detalhada a organização do código, desde as bibliotecas e definições de constantes até as funções auxiliares responsáveis por tarefas específicas, como o controle de dispositivos e a leitura de sensores. Além disso, foram explicados os blocos de configuração inicial e loop principal do programa.

O segundo aspecto abordado foi o código Arduino Uno, que implementa um controle de acesso com leitor de RFID. Este código foi explicado em termos das bibliotecas utilizadas, configurações iniciais e operações de leitura de cartões RFID.

No que diz respeito ao hardware, foi detalhada a configuração de ambos os dispositivos Arduino, bem como a interconexão entre eles para controlar a abertura e fechamento de um portão com base na autenticação de cartões RFID. Foi destacada a utilização de componentes como display LCD, transmissor infravermelho, sensores de temperatura e distância, relés e placas Ethernet para proporcionar funcionalidades de automação e segurança.

Um aspecto relevante deste trabalho foi a coleta de feedback de três usuários durante o desenvolvimento e teste do projeto. Suas observações e críticas foram fundamentais para a identificação de melhorias necessárias na interface de usuário, na organização dos componentes físicos e no design das páginas da web hospedadas no servidor. Como resultado, foram implementadas travas de segurança, uma interface mais atraente e a reorganização eficiente dos componentes físicos.

Em suma, este trabalho demonstrou a complexidade e a interdisciplinaridade envolvidas em um projeto de automação residencial baseado em Arduino. A integração de software, hardware e interação com usuários desempenha um papel fundamental na criação de sistemas eficientes e amigáveis. As melhorias feitas com base no feedback dos usuários demonstram o comprometimento em aprimorar a experiência do usuário e a eficiência do sistema como um todo. Portanto, este projeto serve como um exemplo de como a tecnologia pode ser aplicada para melhorar a automação e segurança residencial.

REFERÊNCIAS

[1] CAVA, T. **X10:** the revolutionary smart home device you've never heard of. [S. l.]: Medium, 2021. Disponível em: https://medium.com/digitalshroud/x10-the-revolutionary-smart-home-device-youve-never-heard-of-48790e272e1f. Acesso em: 28 dez. 2023.

[2] Z-WAVE. **Safer, Smarter Homes Start with Z-Wave**. [S. l.]: Z-Wave, 2023. Disponível em: https://www.z-wave.com/. Acesso em: 28 dez. 2023.

[3] CSA. **Zigbee:** the full-stack folution for all smart devices. [S. l.]: CSA, 2022. Disponível em: https://csa-iot.org/all-solutions/zigbee/. Acesso em: 28 dez. 2023.

[4] QUBINO. **Z-Wave:** 9 features that make it awesome. [S. l.]: Qubino, 2019. Disponível em: https://qubino.com/z-wave-9-features-that-make-it-awesome/. Acesso em: 28 dez. 2023.

[5] FERSMAN, E.; PETTERSSON, J. ; HÖGLUND, A.; SANDERS, E.; ELEFTHERIADIS, L. Intelligent sustainability: the role of AI in energy consumption, management and new revenues. [S. l.]: Ericsson, 2023. Disponível em: https://www.ericsson.com/en/blog/2023/6/intelligent-sustainability-ai-energy-consumption. Acesso em: Acesso em: 28 dez. 2023.

[6] SECURITY. **What Is Home Automation and How Does It Work:** Home automation makes your life more convenient by making certain tasks occur automatically. [S. l.]: Security, 2023. Disponível em: https://www.security.org/resources/smart-home-automation-guide/. Acesso em: 28 dez. 2023.

[7] ARDUINO. **Arduino Mega 2560 Rev3**. [S. l.]: Arduino, 2023. Disponível em: https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf. Acesso em: 28 dez. 2023.

APÊNDICE A – Código do Arduino Mega 2560

```
#include <LiquidCrystal.h>
#include <IRremote.hpp>
#include <Ethernet.h>
#include <RTClib.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>

#define IP IPAddress(192, 168, 0, 169)
#define REQ_BUF_SZ 40

#define ilum_a 22
#define ilum_b 24
#define ilum_c 26
#define ilum_d 28

#define auto_r 30
#define sec_gate 32
#define gateKey 34

#define LDRPin A3
#define tempPin A2
#define trigPin 35
#define echoPin 36

bool autoIlum = true;
bool autoRega = true;
bool secLamp = true;
bool secGate = true;
float ambTemp = 0;

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);
File webFile;
char HTTP_req[REQ_BUF_SZ] = { 0 };
byte req_index = 0;
bool root = false;

LiquidCrystal lcd(8, 9, 33, 5, 6, 7);
const int backLight = 31;

IRsend transmissorIR;
RTC_DS3231 rtc;

const int pinIRLED = 3;
unsigned long codigoHex = 0x00000000;
```

```
long duration, distance;
uint8_t timeDetected = 0;
uint8_t time;

void strClear(char *str, char len) {
  for (int i = 0; i < len; i++)
    str[i] = 0;
}

byte strContains(char *str, char *sfind) {
  byte LEN = strlen(str);
  byte found = 0;
  byte index = 0;

  if (strlen(sfind) > LEN) return 0;
  while (index < LEN) {
    if (str[index] == sfind[found]) {
      found++;
      if (strlen(sfind) == found) return index;
    } else found = 0;
    index++;
  }
  return 0;
}

void ilumHandler(byte output) {
  digitalWrite(ilum_a, (output / 1) % 2);
  digitalWrite(ilum_b, (output / 2) % 2);
  digitalWrite(ilum_c, (output / 4) % 2);
  digitalWrite(ilum_d, (output / 8) % 2);
}

void tempHandler(int temp) {
  if (temp == 0) {
    codigoHex = 0x83D6D202;
  } else if (temp == 1) {
    codigoHex = 0x6A724310;
  } else {
    switch (temp) {
      case 18:
        codigoHex = 0x61E2A07E;
        break;
      case 19:
        codigoHex = 0xC0A0C9CB;
        break;
      case 20:
        codigoHex = 0x5C5E9BC8;
        break;
```

```
      case 21:
        codigoHex = 0x766BCB79;
        break;
      case 22:
        codigoHex = 0x925FF636;
        break;
      case 23:
        codigoHex = 0x8621478D;
        break;
      case 24:
        codigoHex = 0x387C324A;
        break;
      case 25:
        codigoHex = 0xCDD8550F;
        break;
      case 26:
        codigoHex = 0xE9CC7FCC;
        break;
      case 27:
        codigoHex = 0xCD1E58D5;
        break;
      case 28:
        codigoHex = 0x9386B09A;
        break;
      case 29:
        codigoHex = 0x874E8D57;
        break;
      case 30:
        codigoHex = 0xA342B814;
        break;
      default:
        codigoHex = 0x83D6D202;
        break;
    }
  }
  transmissorIR.sendNEC(codigoHex, 32);
}

void autoHandler(byte output) {
  autoIlum = ((output / 1) % 2);
  autoRega = ((output / 2) % 2);
}

void secHandler(byte output) {
  secLamp = ((output / 1) % 2);
  secGate = ((output / 2) % 2);
}

void autoWatch() {
```

```
  DateTime now = rtc.now();
  if (now.hour() >= 6 && now.hour() < 11 && autoIlum == false) {
    digitalWrite(ilum_b, HIGH);
  } else if (now.hour() >= 11 && autoIlum == false) {
    digitalWrite(ilum_b, LOW);
  }

  if (now.hour() >= 6 && now.hour() < 11 && autoRega == false) {
    digitalWrite(auto_r, HIGH);
  } else if (now.hour() >= 11 && autoRega == false) {
    digitalWrite(auto_r, LOW);
  } else {
    digitalWrite(auto_r, HIGH);
  }

  if (digitalRead(gateKey) == true) {
    digitalWrite(sec_gate, LOW);
  } else if (digitalRead(gateKey) == false) {
    if (secGate == true) {
      digitalWrite(sec_gate, HIGH);
    } else if (secGate == false) {
      digitalWrite(sec_gate, LOW);
    }
  }
  if (secLamp == false) {
    if (analogRead(LDRPin) > 550) {
      if (get_distance() <= 15) {
        digitalWrite(ilum_a, LOW);
        timeDetected = now.hour() * 60 + now.minute();
      }
    }
  }
  if (timeDetected != 0) {
    time = now.hour() * 60 + now.minute();
    int differenceMinutes = time - timeDetected;
    if (differenceMinutes < 0) {
      differenceMinutes += 1440;
    }
    if (differenceMinutes >= 1) {
      digitalWrite(ilum_a, HIGH);
      timeDetected = 0;
    }
  }
}

int get_distance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
```

```
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) / 29.1;
  int result;
  if (distance >= 200 || distance <= 0) {
    result = 1000;
  } else {
    result = distance;
  }
  return result;
}

void display_print(char text_first_line[16], char text_second_line[16]) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(text_first_line);
  lcd.setCursor(0, 1);
  lcd.print(text_second_line);
}

void sd_test() {
  Serial.println("Initializing SD card...");
  if (!SD.begin(4)) {
    Serial.println("ERROR - SD card initialization failed!");
    display_print("STATUS WEBPAGE", "ERROR");
    return;
  }
  Serial.println("SUCCESS - SD card initialized.");
  display_print("STATUS WEBPAGE", "SUCCES");
}

void webPage_print(EthernetClient client) {
  bool currentLineIsBlank = true;
  while (client.connected()) {
    if (client.available()) {
      char c = client.read();
      if (req_index < (REQ_BUF_SZ - 1)) {
        HTTP_req[req_index] = c;
        req_index++;
      }
      Serial.print(c);
      if (c == '\n' && currentLineIsBlank) {
        if (root) {
          if (strContains(HTTP_req, "GET / ") || strContains(HTTP_req, "GET
/login.htm")) {
            root = false;
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
```

```
            client.println("Connnection: close");
            client.println();
            webFile = SD.open("login.htm");
          } else if (strContains(HTTP_req, "control.htm")) {
            root = true;
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connnection: close");
            client.println();
            webFile = SD.open("control.htm");
          } else if (strContains(HTTP_req, "main.htm")) {
            root = true;
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connnection: close");
            client.println();
            webFile = SD.open("main.htm");
          } else if (strContains(HTTP_req, "ilum?")) {
            client.println("HTTP/1.1 200 OK");
            client.println();
            byte i = strContains(HTTP_req, "ilum?") + 2;
            byte result = 0;
            byte dig = 0;
            for (i; HTTP_req[i] != 'x'; i++) {
              dig++;
              if (dig >= 2) result *= 10;
              result += (HTTP_req[i] - '0');
            }
            ilumHandler(result);
          } else if (strContains(HTTP_req, "temp?")) {
            client.println("HTTP/1.1 200 OK");
            client.println();
            byte i = strContains(HTTP_req, "temp?") + 2;
            byte result = 0;
            byte dig = 0;
            for (i; HTTP_req[i] != 'x'; i++) {
              dig++;
              if (dig >= 2) result *= 10;
              result += (HTTP_req[i] - '0');
            }
            tempHandler(result);
          } else if (strContains(HTTP_req, "auto?")) {
            client.println("HTTP/1.1 200 OK");
            client.println();
            byte i = strContains(HTTP_req, "auto?") + 2;
            byte result = 0;
            byte dig = 0;
            for (i; HTTP_req[i] != 'x'; i++) {
              dig++;
```

```
      if (dig >= 2) result *= 10;
      result += (HTTP_req[i] - '0');
    }
    autoHandler(result);
  } else if (strContains(HTTP_req, "sec?")) {
    client.println("HTTP/1.1 200 OK");
    client.println();
    byte i = strContains(HTTP_req, "sec?") + 2;
    byte result = 0;
    byte dig = 0;
    for (i; HTTP_req[i] != 'x'; i++) {
      dig++;
      if (dig >= 2) result *= 10;
      result += (HTTP_req[i] - '0');
    }
    secHandler(result);
  } else if (strContains(HTTP_req, "ambTemp.txt")) {
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text");
    client.println("Connnection: close");
    client.println();
    ambTemp = (float(analogRead(tempPin)) * 5 / (1023)) / 0.01;
    client.println(ambTemp);
  } else if (strContains(HTTP_req, "statusGate.txt")) {
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: bool");
    client.println("Connnection: close");
    client.println();
    if (secGate == true){
      client.println(bool(digitalRead(gateKey)));
    }
    else {
      client.println(bool(0));
    }
  } else if (strContains(HTTP_req, "style.css")) {
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/css");
    client.println("Connnection: close");
    client.println();
    webFile = SD.open("style.css");
  } else if (strContains(HTTP_req, "icon.ico")) {
    client.println("HTTP/1.1 200 OK");
    client.println();
    webFile = SD.open("icon.ico");
  } else if (strContains(HTTP_req, "back.jpg")) {
    client.println("HTTP/1.1 200 OK");
    client.println();
    webFile = SD.open("back.jpg");
  } else {
```

```
            client.println("HTTP/4.5 404 Not Found");
            client.println("Content-Type: text/html");
            client.println("Connnection: close");
            client.println();
            webFile = SD.open("404.htm");
          }
        } else {
          if (strContains(HTTP_req, "GET / ") || strContains(HTTP_req, "GET
/login.htm")) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connnection: close");
            client.println();
            webFile = SD.open("login.htm");
          } else if (strContains(HTTP_req, "user=root&pass=1234")) {
            root = true;
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connnection: close");
            client.println();
            webFile = SD.open("main.htm");
          } else if (strContains(HTTP_req, "style.css")) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/css");
            client.println("Connnection: close");
            client.println();
            webFile = SD.open("style.css");
          } else if (strContains(HTTP_req, "icon.ico")) {
            client.println("HTTP/1.1 200 OK");
            client.println();
            webFile = SD.open("icon.ico");
          } else if (strContains(HTTP_req, "back.jpg")) {
            client.println("HTTP/1.1 200 OK");
            client.println();
            webFile = SD.open("back.jpg");
          } else {
            client.println("HTTP/4.2 401 Unauthorized");
            client.println("Content-Type: text/html");
            client.println("Connnection: close");
            client.println();
            webFile = SD.open("401.htm");
          }
        }
        if (webFile) {
          while (webFile.available()) {
            client.write(webFile.read());
          }
          webFile.close();
        }
```

```
        req_index = 0;
        strClear(HTTP_req, REQ_BUF_SZ);
        break;
      }
      if (c == '\n') {
        currentLineIsBlank = true;
      } else if (c != '\r') {
        currentLineIsBlank = false;
      }
    }
  }
  delay(1);
  client.stop();
}

void setup() {
  Serial.begin(9600);

  pinMode(ilum_a, OUTPUT);
  pinMode(ilum_b, OUTPUT);
  pinMode(ilum_c, OUTPUT);
  pinMode(ilum_d, OUTPUT);
  pinMode(auto_r, OUTPUT);
  pinMode(sec_gate, OUTPUT);
  pinMode(gateKey, INPUT);
  pinMode(LDRPin, INPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  digitalWrite(ilum_a, HIGH);
  digitalWrite(ilum_b, HIGH);
  digitalWrite(ilum_c, HIGH);
  digitalWrite(ilum_d, HIGH);
  digitalWrite(auto_r, HIGH);
  digitalWrite(sec_gate, HIGH);

  lcd.begin(16, 2);
  pinMode(backLight, OUTPUT);
  digitalWrite(backLight, HIGH);

  rtc.begin();
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  Wire.begin();

  Ethernet.begin(mac, IP);
  server.begin();
  sd_test();

  transmissorIR.begin(pinIRLED);
```

```
}

void loop() {
  EthernetClient client = server.available();
  if (client) {
    webPage_print(client);
  } else {
    autoWatch();
  }
}
```

APÊNDICE B – Código do Arduino UNO

```
#include <Wire.h>
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9

MFRC522 rfid(SS_PIN, RST_PIN);

const int openGate = 8;
bool gateState = false;

void setup() {
  Wire.begin();
  SPI.begin();
  rfid.PCD_Init();
  pinMode(openGate, OUTPUT);
  digitalWrite(openGate, LOW);
}
void loop() {
  leituraRfid();
}
void leituraRfid() {
  if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial()) return;
  String strID = "";
  for (byte i = 0; i < 4; i++) {
    strID +=
      (rfid.uid.uidByte[i] < 0x10 ? "0" : "") + String(rfid.uid.uidByte[i],
HEX) + (i != 3 ? ":" : "");
  }
  strID.toUpperCase();
  if (strID.indexOf("B3:42:0F:0F") >= 0 || strID.indexOf("5A:6C:EA:5B") >= 0)
{
    if (gateState == false) {
      digitalWrite(openGate, HIGH);
      gateState = true;
    } else {
      digitalWrite(openGate, LOW);
      gateState = false;
    }
  }
  delay(3000);
  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();
}
```

APÊNDICE C – Página 401 Error

```
<!DOCTYPE HTML>

<html lang="pt-br">

<head>
    <title>401 Error</title>
    <link rel="stylesheet" href="style.css" />
    <link rel="icon" href="icon.ico" />
    <meta charset="UTF-8">
</head>

<body>
    <table class="controlMenuTable">
        <tr>
            <td>
                <h1 style="margin-left: 20px;">401: Erro de Autenticação.</h1>
            </td>
        </tr>
        <tr>
            <td>
                <h1 style="margin-left: 20px;">Não Autorizado!</h1>
            </td>
        </tr>
        <tr>
            <td style="text-align: right;">
                <input style="margin-right: 20px; margin-bottom: 20px;"
type="button" onclick="location.href='login.htm';" value="Voltar" />
            </td>
        </tr>
    </table>
</body>

</html>
```

APÊNDICE D – Página 404 Error

```
<!DOCTYPE HTML>

<html lang="pt-br">

<head>
    <title>404 Error</title>
    <link rel="stylesheet" href="style.css" />
    <link rel="icon" href="icon.ico" />
    <meta charset="UTF-8">
</head>

<body>
    <table class="controlMenuTable">
        <tr>
            <td>
                <h1 style="margin-left: 20px;">404 Página não encontrada.</h1>
            </td>
        </tr>
        <tr>
            <td style="text-align: right;">
                <input style="margin-right: 20px; margin-bottom: 20px;"
type="button" onclick="location.href='login.htm';" value="Voltar" />
            </td>
        </tr>
    </table>
</body>

</html>
```

APÊNDICE E – Página de Login

```
<!DOCTYPE HTML>

<html lang="pt-br">
    <head>
        <title>Login</title>
        <link rel="stylesheet" href="style.css"/>
        <link rel="icon" href="icon.ico" />
        <meta charset="UTF-8">

        <Script>
            function login()
            {
                const user = document.querySelector('#user')
                const pass = document.querySelector('#pass')
                const self = event.target
                self.disabled = true;
                fetch(`user=${user.value}&pass=${pass.value}`).then(r => {
                    if (r.status == 200) {
                        window.location = 'main.htm'
                    } else {
                        pass.value = ''
                    }
                    self.disabled = false;
                }).catch(e => {
                    self.disabled = false;
                })
            }

            function timeFunction()
            {
                var now = new date();
                return now.toLocaleString();
            }
            setInterval("timeFunction()", 1000);
            var displayTime = timeFunction();
        </Script>

    </head>
    <body>
        <table class="controlMenuTable">
            <tr>
                <td style="text-align: center;">
                    <h1>Acesso</h1>
                    <h1>Entre com seu usuário e senha.</h1>
                </td>
            </tr>
```

```
        </table>

        <div class="login">
        <form class="loginForm">
            <label for="user">Usuário:</label><br>
            <input type="text" id="user" style="width: 200px;"/><br/>
            <label for="pass">Senha:</label><br>
            <input type="password" id="pass" style="width: 200px;"/><br/>
            <div style="text-align: center; margin-top: 15px;"><button
onclick="login()">Entrar</button></div>
        </form>
        </div>

        <Script>

        </Script>
    </body>


</html>
```

APÊNDICE F – Página Inicial

```
<!DOCTYPE HTML>

<html lang="pt-br">

<head lang="pt-br">
    <title>Bem-vindo</title>
    <link rel="stylesheet" href="style.css" />
    <link rel="icon" href="icon.ico" />
    <meta charset="UTF-8">
</head>

<body>
    <table class="controlMenuTable">
        <tr>
            <td style="text-align: center;">
                <h1>Bem-vindo!</h1>
                <h1>Início</h1>
            </td>
        </tr>
    </table>

    <table class="controlTable">
        <tr>
            <td class="controlTd">
                <p>Acessar o painel de controle com os seguintes recusos:</p>
                <ul>
                    <li>Iluminação</li>
                    <li>Temperatura</li>
                    <li>Agenda de tarefas</li>
                    <li>Segurança</li>
                </ul>
                <input type="button" onclick="location.href='control.htm';"
value="Controle" />
            </td>
            <td class="controlTd">
                <p>Retornar para a tela de entrada.</p>
                <input type="button" onclick="location.href='login.htm';"
value="Sair" />
            </td>
        </tr>
    </table>
</body>

</html>
```

APÊNDICE G – Página Painel de controle

```
<!DOCTYPE HTML>
<html lang="pt-br">

<head>
    <title>Painel de controle</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="style.css" />
    <link rel="icon" href="icon.ico" />
    <script>
        var ilumOutput = 15;
        var tempOutput = 24;
        var autoOutput = 3;
        var secOutput = 3;
        var ambTemp = 0;

        function httpGet(theUrl) {
            var xmlHttp = new XMLHttpRequest();
            xmlHttp.open("GET", theUrl, false);
            xmlHttp.send(null);
            return xmlHttp.responseText;
        }
        function ilumHandle(name) {
            var input = document.getElementsByName(name)[0];
            var valor = 0;

            if (name == "lamp 1") {
                valor = 1;
            } else if (name == "lamp 2") {
                valor = 2;
            } else if (name == "lamp 3") {
                valor = 4;
            } else if (name == "lamp 4") {
                valor = 8;
            } else {
                valor = 0;
            }

            if (input.value == "Ligar") {
                ilumOutput -= valor;
                input.value = "Desligar";
            } else {
                ilumOutput += valor;
                input.value = "Ligar";
            }

            //alert(ilumOutput);
```

```
            httpGet("ilum?=" + ilumOutput.toString() + "x");
        }
        function tempHandle(name) {
            var input = document.getElementsByName("setTemp")[0];
            var input_onoff = document.getElementsByName("onOff")[0];

            if (name == "tempU" && tempOutput < 30) {
                tempOutput += 1;
            } else if (name == "tempD" && tempOutput > 18) {
                tempOutput -= 1;
            } else {
                tempOutput = tempOutput;
            }

            //alert(tempOutput);
            if (name == "onOff" && input_onoff.value == "Ligar A/C"){
                input_onoff.value = "Desligar A/C";
                var btu = document.getElementsByName("tempU")[0];
                btu.disabled = false;
                var btd = document.getElementsByName("tempD")[0];
                btd.disabled = false;
                httpGet("temp?=" + "1" + "x");
            } else if (name == "onOff" && input_onoff.value == "Desligar
A/C"){
                input_onoff.value = "Ligar A/C";
                var btu = document.getElementsByName("tempU")[0];
                btu.disabled = true;
                var btd = document.getElementsByName("tempD")[0];
                btd.disabled = true;
                tempOutput = 24;
                input.value = tempOutput.toString();
                httpGet("temp?=" + "0" + "x");
            } else {
                input.value = tempOutput.toString();
                httpGet("temp?=" + tempOutput.toString() + "x");
            }
        }
        function scheduleHandle(name) {
            var input = document.getElementsByName(name)[0];
            var valor = 0;

            if (name == "autoLamp") {
                valor = 1;
            } else if (name == "autoRega") {
                valor = 2;
            } else {
                valor = 0;
            }
```

```
            if (input.value == "Ativar") {
                autoOutput -= valor;
                input.value = "Desativar";
                if (valor == 1) {
                    var input = document.getElementsByName("lamp 2")[0];
                    input.disabled = true;
                    ilumWatch();
                }
            } else {
                autoOutput += valor;
                input.value = "Ativar";
                if (valor == 1) {
                    var input = document.getElementsByName("lamp 2")[0];
                    input.disabled = false;
                    ilumWatch();
                }
            }

            //alert(autoOutput);
            httpGet("auto?=" + autoOutput.toString() + "x");
        }
        function secHandle(name) {
            var input = document.getElementsByName(name)[0];
            var valor = 0;

            if (name == "secLamp") {
                valor = 1;
            } else if (name == "secPorta") {
                valor = 2;
            } else {
                valor = 0;
            }

            if (input.value == "Ativar") {
                secOutput -= valor;
                input.value = "Desativar";
                if (valor == 1) {
                    var inputLamp = document.getElementsByName("lamp 1")[0];
                    inputLamp.disabled = true;
                }
            } else {
                secOutput += valor;
                input.value = "Ativar";
                if (valor == 1) {
                    var inputLamp = document.getElementsByName("lamp 1")[0];
                    inputLamp.disabled = false;
                }
            }
```

```
            //alert(secOutput);
            httpGet("sec?=" + secOutput.toString() + "x");
        }
        function loadValue() {
            var inputSet = document.getElementsByName('setTemp')[0];
            inputSet.value = tempOutput.toString();

            updateValue();
        }
        function updateValue() {
            var inputAmb = document.getElementsByName('ambTemp')[0];
            ambTemp = httpGet("ambTemp.txt");
            inputAmb.value = (ambTemp.toString() + " °C");

            if (!(autoOutput / 1) % 2){
                ilumWatch();
            }

            var inputGate = document.getElementsByName('secPorta')[0];
            statusGate = httpGet("statusGate.txt");
            if (statusGate == 1){
                inputGate.disabled = true;
            }
            else {
                inputGate.disabled = false;
            }

            setTimeout(function(){
                updateValue();
            }, 1000);
        }
        function ilumWatch() {
            var input = document.getElementsByName("lamp 2")[0];
            var hora = new date().getHours();

            if (hora >= 6 && hora < 11 && input.value == "Desligar") {
                input.value = "Ligar";
                ilumOutput += 2;
            } else if (hora >= 11 && input.value == "Ligar") {
                input.value = "Desligar";
                ilumOutput -= 2;
            }
        }
    </script>
</head>

<body onload="loadValue()">
    <table class="controlMenuTable">
        <td>
```

```
            <h1 style="margin-left: 20px;">Painel de Controle</h1>
        </td>
        <td style="text-align: right;">
            <input style="margin-right: 20px;" type="button"
onclick="location.href='main.htm';" value="Início" />
            <input style="margin-right: 20px;" type="button"
onclick="location.href='login.htm';" value="Sair" />
        </td>
    </table>
    <table class="controlTable">
        <tr>
            <td class="controlTd">
                <form>
                    <h2>Iluminação</h2>
                    <table style="margin: auto;">
                        <tr>
                            <td>
                                <label for="lamp1">Área Externa</label><br>
                                <input type="button" name="lamp 1"
value="Ligar" onClick="ilumHandle('lamp 1')"><br>
                            </td>
                            <td>
                                <label for="lamp3">Portão</label><br>
                                <input type="button" name="lamp 3"
value="Ligar" onClick="ilumHandle('lamp 3')"><br>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <label for="lamp2">Área Interna</label><br>
                                <input type="button" name="lamp 2"
value="Ligar" onClick="ilumHandle('lamp 2')"><br>
                            </td>
                            <td>
                                <label for="lamp4">Quarto</label><br>
                                <input type="button" name="lamp 4"
value="Ligar" onClick="ilumHandle('lamp 4')"><br>
                            </td>
                        </tr>
                    </table>
                </form>
            </td>
            <td class="controlTd">
                <h2>Temperatura</h2>
                <form>
                    <label>Temperatura selecionada:</label><br>
                    <input type="button" name="onOff" value="Ligar A/C"
onClick="tempHandle('onOff')"><br>
```

```
                    <input type="button" name="tempD" value="-"
onClick="tempHandle('tempD')" disabled = true>
                    <input type="text" name="setTemp" disabled>
                    <input type="button" name="tempU" value="+"
onClick="tempHandle('tempU')" disabled = true><br>
                    <label>Temperatura do ambiente:</label><br>
                    <input type="text" name="ambTemp" disabled><br>
                </form>
            </td>
        </tr>
        <tr>
            <td class="controlTd">
                <h2>Agenda de tarefas</h2>
                <label for="autoLamp">Iluminação</label><br>
                <input type="button" name="autoLamp" value="Ativar"
onClick="scheduleHandle('autoLamp')"><br>
                <label for="autoRega">Irrigação</label><br>
                <input type="button" name="autoRega" value="Ativar"
onClick="scheduleHandle('autoRega')"><br>
            </td>
            <td class="controlTd">
                <h2>Segurança</h2>
                <label for="secLamp">Luz de presença</label><br>
                <input type="button" name="secLamp" value="Ativar"
onClick="secHandle('secLamp')"><br>
                <label for="secPorta">Portão</label><br>
                <input type="button" name="secPorta" value="Ativar"
onClick="secHandle('secPorta')"><br>
            </td>
        </tr>
    </table>
</body>

</html>
```

APÊNDICE H – Código de definição de estilos

```
body {
  /*background-color: rgb(179, 177, 177);*/
  background-image: url("back.jpg");
}

input {
  width: 100px;
  height: 30px;
  font-size: medium;
  margin-bottom: 5px;
  text-align: center;
}

button {
  width: 100px;
  height: 30px;
  font-size: medium;
}

label {
  font-size: large;
}

p {
  font-size: large;
  width: 250px;
  height: 40px;
  margin: auto;
}

ul {
  list-style-type: circle;
  text-align: left;
}

.controlMenuTable {
  margin: auto;
  background-image: linear-gradient(to bottom right, rgba(255, 253, 255, 1),
rgba(255, 255, 255, 0.7));
  border-radius: 30px;
  width: 800px;
}

.controlTable {
  margin: auto;
  border-spacing: 25px;
```

```
}

.controlTd {
  border: 1px solid black;
  width: 330px;
  height: 255px;
  text-align: center;
  border-radius: 50px;
  background-image: linear-gradient(to bottom right, rgba(121, 121, 192, 0.8),
rgba(173, 216, 230, 0.80));
}

.login {
  margin-left: auto;
  margin-right: auto;
  margin-top: 30px;
  border: 2px solid white;
  padding: 10px;
  background-image: linear-gradient(to bottom right, rgba(255, 253, 255, 1),
rgba(255, 255, 255, 0.7));
  width: 215px;
  height: 180px;
  border-radius: 50px;
}

.loginForm{
  margin: 5px;
}
```