



**CENTRO UNIVERSITÁRIO DE BRASÍLIA
FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS - FATECS
CURSO DE ENGENHARIA DE COMPUTAÇÃO**

LUCAS GOMES SOMBRA

**AUTOMAÇÃO RESIDENCIAL PARA CONTROLE DE ILUMINAÇÃO,
SEGURANÇA E MONITORAMENTO DE TEMPERATURA USANDO O
ARDUINO MEGA.**

BRASÍLIA - DF

2016

LUCAS GOMES SOMBRA

**AUTOMAÇÃO RESIDENCIAL PARA CONTROLE DE ILUMINAÇÃO,
SEGURANÇA E MONITORAMENTO DE TEMPERATURA USANDO O
ARDUINO MEGA.**

Trabalho de Conclusão de Curso apresentado no Centro Universitário de Brasília (Uniceub) à Banca Examinadora da Faculdade de Tecnologia e Ciências Sociais Aplicadas como pré-requisito para angariar o título de Engenheiro de Computação.

Orientador: Prof. MSc. Luciano Henrique Duque.

BRASÍLIA - DF

2016

LUCAS GOMES SOMBRA

**AUTOMAÇÃO RESIDENCIAL PARA CONTROLE DE ILUMINAÇÃO,
SEGURANÇA E MONITORAMENTO DE TEMPERATURA USANDO O
ARDUINO MEGA.**

Trabalho de Conclusão de Curso
apresentado no Centro Universitário de Brasília
(Uniceub) à Banca Examinadora da Faculdade
de Tecnologia e Ciências Sociais Aplicadas
como pré-requisito para angariar o título de
Engenheiro de Computação.

Orientador: Prof. MSc. Luciano Henrique Duque.

BANCA EXAMINADORA

Msc. Prof.º Luciano Henrique Duque
Orientador

Msc. Prof.º Luís Cláudio Lopes de Araujo
Examinador

Prof.ª Claudia Ochoa
Examinadora

BRASÍLIA - DF

2016

AGRADECIMENTOS

Agradeço à minha mãe por ter sempre me orientado a ser uma pessoa que luta próprios sonhos com honra, honestidade e hombridade. À minha avó Gilda dos Reis por ter custeado meu curso de Engenharia de Computação. Ao meu pai por me ensinar que um homem não é feito de erros e sim do aprendizado que obtém dos mesmos. Ao meu avô Evandro por me lembrar que sem ser metódico, toda luta travada é uma luta perdida. À minha avó Celi Sombra por ressaltar que devemos ser humildes às forças da vida e espirituais. Por fim, ao meu tio Edison por me ensinar que todos passamos por dificuldades e que elas existem para nos tornar pessoas melhor.

Finalmente, gostaria de demonstrar meu respeito ao orientador, Luciano Duque, e ao coordenador do curso de computação, Abiézer Amarília. Sem estes, essa jornada não seria possível.

DEDICATÓRIA

Dedico este à minha família. E como o homem de nada seria sem a família que Deus lhe deixou escolher, dedico este também aos meus amigos que me acompanharam durante essa jornada.

EPIÍGRAFE

"Isso é para os loucos, os desajustados, os rebeldes, os encrenqueiros, os círculos encaixados no espaço do quadrado... Os que veem as coisas de um modo diferente – eles não gostam de regras e eles não tem respeito pelo status quo... Você pode citá-los, discordar deles, glorificá-los ou difamá-los, mas a única coisa que você não pode fazer é ignorá-los por que eles mudam as coisas... Eles empurram a raça humana para frente. E enquanto alguns os enxergam como loucos, nós enxergamos gênios, porque as pessoas que são loucas o suficiente para pensar que podem mudar o mundo, são os que mudam... "

– Steve Jobs, Think Different, 1997

RESUMO

Este trabalho visa a construção de uma automação residencial que possibilite controlar remotamente sistemas presentes na maioria das casas dos dias atuais. Este modelo é resultado da integração da automação da iluminação, segurança, monitoramento de temperatura e controle de acesso via uma fechadura eletromagnética. O intuito do modelo funcional gerado por este projeto é promover maior comodidade, segurança, acessibilidade ao gerenciar os diferentes sistemas mencionados através do uso de redes sem fio. Para que esta administração de recursos seja realizada remotamente, o projeto utiliza um micro controlador chamado Arduino Mega em conjunto com um módulo relé responsável por integrar e acionar os sistemas da casa. O Arduino é conectado à uma placa de rede específica para o mesmo, que por sua vez, interligará este a rede sem fio residencial.

Palavras-chave: Automação, Integração, Segurança, Acessibilidade, Iluminação, Arduino Mega.

HOME AUTOMATION FOR RESIDENTIAL LIGHTING CONTROL, SECURITY AND TEMPERAUTE MONITORING USING ARDUINO MEGA.

ABSTRACT

This work aims to build a home automation that allows remote control over different systems present in the most of the houses nowadays. This model is the result of integrating the automation of lighting, security, temperature monitoring and access control via an electromagnetic lock. The purpose of the functional model generated by this project is to promote greater convenience, security, accessibility by managing the different mentioned systems mentioned with wireless networks. In order to have this resource management performed remotely, the project uses a micro controller called Arduino Mega in conjunction with a relay module responsible for integrating and triggering the systems of the house. The Arduino is connected to a specific network adapter, which in turn, will link it to the house wireless network.

Key-words: Automation, Integration, Security, Accessibility, Lighting, Arduino Mega.

Sumário

CAPÍTULO 1: INTRODUÇÃO	14
1.2 MOTIVAÇÃO	16
1.3 OBJETIVOS	17
1.3.1 OBJETIVOS GERAIS	17
1.3.2 OBJETIVOS ESPECÍFICOS	17
1.4 METODOLOGIA	18
1.5 RESULTADOS ESPERADOS	20
1.6 ESTRUTURA DO TRABALHO	20
CAPÍTULO 2: REFERENCIAL TEÓRICO	21
2.1 AUTOMAÇÃO RESIDENCIAL	21
2.2 REDES SEM FIO	22
2.3 MICRO CONTROLADOR ARDUINO MEGA	23
2.4 SHIELD ETHERNET	27
2.5 ACIONAMENTO DE CARGAS ELÉTRICAS COM RELÉS	31
2.6 MEDIÇÃO E MONITORAMENTO DE TEMPERATURA	33
2.7 CONTROLE DE ACESSO FÍSICO	34
2.8 ALARME	36
2.8.1 SENSOR DE PRESENÇA	36
2.8.2 SIRENE PIEZOELÉCTRICA	39
2.9 SOFTWARE: IDE DO ARDUINO	40
2.10 TESTE DE CONFIABILIDADE	42
CAPÍTULO 3: DESENVOLVIMENTO	44
3.1 DESCRIÇÃO DO SISTEMA PROPOSTO	44
3.2 BLOCO DE COMUNICAÇÃO	46

3.2.1 CONFIGURAÇÃO DO ARDUINO MEGA E ETHERNET SHIELD	46
3.2.2 CONFIGURAÇÃO DO ROTEADOR	48
3.2.3 SERVIDOR WEB	49
3.3 BLOCO DA INTERFACE DE ACIONAMENTO	51
3.3.1 CONTROLE DO MODULO RELÉ COM O ARDUINO	51
3.3.2 ACIONAMENTO DA ILUMINAÇÃO	53
3.3.3 ACIONAMENTO DA TRANCA ELÉTRICA	54
3.3.4 ACIONAMENTO DO ALARME	56
3.4 BLOCO DA MONITORAÇÃO DE TEMPERATURA	58
3.5 RESULTADOS FINAIS DO PROJETO	60
CAPÍTULO 4: TESTES E RESULTADOS	64
4.1 PRIMEIRO CENÁRIO	64
4.2 SEGUNDO CENÁRIO	70
4.3 TERCEIRO CENÁRIO	73
CAPÍTULO 5: CONCLUSÃO	77
5.1 CONCLUSÕES	77
5.2 TRABALHOS FUTUROS	78
6. REFERÊNCIAS BIBLIOGRÁFICAS	80
ANEXO A	83

LISTA DE FIGURAS

Figura 1. Metodologia. Fonte: Autor.....	18
Figura 2. Entradas do Arduino. Fonte: Arduino.CC.....	24
Figura 3. Esquemático Elétrico - Arduino Mega. Fonte: ARDUINO.CC.....	26
Figura 4. Entradas e Saídas do Arduino. Fonte: Adaptado de ARDUINO.CC.....	27
Figura 5. Arduino Ethernet Shield. Fonte: barretuino.com.br.....	28
Figura 6. Esquemático do Shield Ethernet. Fonte: elec.com.....	29
Figura 7. Cabeçalho TCP/UDP. Fonte: www.seucurso.com.br.....	30
Figura 8. Esquemático do Relé. Fonte: www.mecatronicaatual.com.br.....	31
Figura 9. Módulo Relé. Fonte: www.codeproject.com.....	32
Figura 10. DHT11. Fonte: AOSONG.....	33
Figura 11. Conectando DHT11 no Arduino. Fonte: www.oarduino.com.br.....	34
Figura 12. Campo Eletromagnético do Solenoide. Fonte: www.ponto.ciencia.com.br.....	35
Figura 13. Interior da Tranca Elétrica. Fonte: adaptado de www.stam.com.br.....	35
Figura 14. Sensor PIR. Fonte: robotgestation.com.....	37
Figura 15. Lente Fresnel. Fonte: www.bhlens.com.....	37
Figura 16. Esquemático Sensor-Sirene. Fonte: www.fazedores.com.....	38
Figura 17. Sirene DNI 4042. Fonte: www.dni.com.br.....	39
Figura 18. Deformação piezoelétrica. Fonte: www.sabereletronica.com.br.....	40
Figura 19. IDE do Arduino. Fonte: joaoshmitt.wordpress.com.....	41
Figura 20. Fluxograma do Protótipo de Automação. Fonte: Autor.....	44
Figura 21. Configuração Placa de Ethernet. Fonte: Autor.....	46
Figura 22. Atribuição Estática do Roteador. Fonte: Autor.....	48
Figura 23. Encaminhamento de Porta. Fonte: Autor.....	48
Figura 24. Linguagem de programação do servidor. Fonte: Autor.....	50
Figura 25. Código que produz o acionamento. Fonte: Autor.....	52
Figura 26. Função void acionaPin. Fonte: Autor.....	53
Figura 27. Automação da Iluminação. Fonte: adaptado de arduinobr.com.br.....	54
Figura 28. Esquemático Elétrico da Tranca. Fonte: Autor.....	55
Figura 29. Esquemático do Alarme. Fonte: Autor, adaptação de www.fazedores.com.....	57
Figura 30. Configuração da senha WPA. Fonte: Autor.....	57
Figura 31. Código do sensor de temperatura. Fonte: Autor.....	59
Figura 32. Código HTML que imprime a temperatura no navegador. Fonte: Autor.....	59
Figura 33. Interface Web. Fonte: Autor.....	60
Figura 34. Frente da maquete. Fonte: Autor.....	61
Figura 35. Traseira da maquete. Fonte: Autor.....	62
Figura 36. Diagrama dos testes de comunicação com o servidor. Fonte: Autor.....	65
Figura 37. Código que gera a mensagem validadora da conexão. Fonte: Autor.....	65
Figura 38. Mensagem validadora de conexão no monitor serial. Fonte: Autor.....	66
Figura 39. Código que gera a mensagem validadora do fechamento da conexão. Fonte: Autor.....	66
Figura 40. Diagrama dos testes de acionamento dos relés. Fonte: Autor.....	67
Figura 41. Mensagens validadas dos testes de acionamento. Fonte: Autor.....	68
Figura 42. Código que grava os status na EEPROM. Fonte: Autor.....	69

Figura 43. Diagrama dos testes do sensor de temperatura. Fonte: Autor.....	70
Figura 44. Mensagens validadoras dos testes de temperatura no monitor serial. Fonte: Autor.....	71
Figura 45. Código que gera as mensagens validadores de temperatura. Fonte: Autor.....	71
Figura 46. Termômetro externo usado. Fonte: Autor.....	72
Figura 47. Diagrama dos testes funcionais do sensor de movimento. Fonte: Autor.....	75

LISTA DE TABELAS

Tabela 1. Resultados dos Testes de acionamento. Fonte: Autor.....	69
Tabela 2. Resultados dos testes do sensor de temperatura. Fonte: Autor.....	72
Tabela 3. Resultados dos testes do sensor de movimento. Fonte: Autor.....	76

CAPÍTULO 1: INTRODUÇÃO

Desde o surgimento da World Wide Web com o advento da Guerra Fria e a sua popularização nos anos 1990 sob o nome de Internet, a velocidade do crescimento tecnológico nos anos a seguir se deu de modo exponencial, fato que pode ser explicado pela facilidade de acesso à informação globalmente distribuída por este meio de comunicação. O ser humano, desde então, vem buscando maior acessibilidade, rapidez, segurança e conforto diariamente como consequência da integração de produtos, pessoas, serviços, mercados e informações providas por este meio.

Automação é representada por um “sistema em que os processos operacionais em fábricas, estabelecimentos, hospitais e casas são controlados e executados por meio de dispositivos mecânicos ou eletrônicos” (DICIO. 2016).

Seguindo esta lógica, automação residencial, é aplicação da tecnologia com o intuito de gerar maior segurança, comodidade, acessibilidade aos habitantes de uma residência ao permitir que tarefas usualmente realizadas por estes mecanicamente; como acendimento de luzes, acionamento de sistema de segurança, controle de consumo de energia, abertura de portões; sejam executadas de forma automática ou ainda remotamente (ASSOCIAÇÃO ESPANHOLA DE DOMÓTICA. 2015).

O início da automação residencial se deu na década de 1970 com o lançamento do X-10 criado pela Pico Eletronics LTDA, um protocolo de comunicação entre dispositivos eletrônicos através da rede elétrica. Este módulo é uma tecnologia PLC (*Power Line Carrier*) ou comunicação via rede elétrica em português, que tornou possível o controle dos dispositivos sem ser preciso alterações na infraestrutura elétrica da residência (CHAGAS, CHRISTIAN. 2010).

Nos anos que se seguiram, diversas tecnologias foram sendo integradas e criadas inteiramente para as chamadas casas inteligentes e, finalmente, a evolução dessas hoje torna possível o controle e monitoramento da residência via a Internet. Com o crescimento tecnológico acelerado do século XXI, houve a criação dos famosos smartphones que unificam em um mesmo aparelho telefonia, acesso à web, a redes sociais e a aplicativos de terceiros. Como resultado, hoje é possível o controle dos diversos aspectos de uma habitação através de um mero dispositivo portátil.

No Brasil, a automação residencial vem tomando seus primeiros passos nos últimos anos. Sendo que, atualmente, representa um mercado de consumo que fatura aproximadamente R\$500 milhões de reais por ano e tem apresentado um crescimento contínuo de 30% ao ano. O principal fator que alavanca essa massa de consumo é o anseio por maior conforto e segurança (TECHINBRAZIL. 2015).

Sistemas de segurança para residências, por exemplo, já possuíam preços elevados e foram considerados sinal de status. Hoje, é possível comprar modelos simples pela internet por preços acessíveis. Sendo apenas necessário a contratação de um terceiro para a instalação, quando o usuário não desejar realiza-la. Pois a maioria dos produtos oferecidos na internet já são projetados para serem instalados pelo próprio comprador.

Entretanto, soluções de automação residencial completas e personalizadas, mesmo após sofrerem reduções de preço devido a popularização da tecnologia nos últimos anos ainda apresentam preços elevados, cujos valores partem de R\$4.000 e podem ultrapassar R\$70.000 (TECHINBRAZIL. 2015).

No país, hoje, as tecnologias mais utilizadas são termostatos, gerenciadores de energia, sistemas de controle de áudio e vídeo, controle do sistema elétrico e sistemas de segurança. A automação está sendo cada vez mais integrada a edifícios. A razão disto é que nestes além da quantidade comprada ser maior, a padronização aplicada permite eliminar custos com soluções individuais.

O grande problema, então, são as casas. Uma vez que estas necessitam de soluções personalizadas desde a concepção até a instalação. Personalização que causa aumento no preço final do produto e causa dificuldade de aquisição deste.

O fruto deste projeto é criação de uma automação que unifique o sistema elétrico, alarme e monitoramento de temperatura. Podendo este ser introduzido a qualquer casa gerando maior conforto e conveniência para os habitantes.

Esta automação é possível graças a utilização de um micro controlador chamado Arduino Mega e uma placa relé. Micro controlador é um “dispositivo semicondutor que integra todas as partes básicas de um microcomputador (memórias voláteis, portas de entrada e saída e microprocessador) geralmente limitado em termos de memória. É utilizado em aplicações

específicas, como em automação residencial, automação predial e industrial” (GIMENEZ, SALVADOR. 2010).

Finalmente, relé é um dispositivo que “tem sua construção baseada num contato metálico que se abre ou fecha sob a influência de um campo eletromagnético induzido por uma bobina no seu interior (PATSKO, LUIS. 2006). ” Assim, o resultado desse projeto é a integração do *Arduino* com uma placa relé e, uma vez integrados, o objetivo é que possam ser controlados remotamente através de uma página HTML usando a rede sem fio.

1.2 MOTIVAÇÃO

Fazendo uma breve análise do mercado de automação de habitações no Brasil nos últimos dez anos se percebe que este ainda está dando seus primeiros passos, mesmo considerando que o número de fabricantes e fornecedores triplicou em menos de cinco anos (MURATORI, JOSE. 2015).

O aumento significativo na disponibilidade causa, como consequência, uma queda nos preços devido a maior concorrência. Assim novos consumidores que já possuem conhecimento das vantagens da automação residencial procuram entrar nesse mercado, portanto, sendo apenas uma questão de tempo para que sua inserção neste conglomerado aconteça.

Um breve estudo realizado pela a AURESIDE em 2013, Associação Brasileira de Automação Residencial e Predial, mostra que existem 1,5 milhão de residências habilitadas para receber projetos de automação. Entretanto, existe ao mesmo tempo apenas 15% da quantidade de profissionais habilitados para atender essa necessidade.

Visto as informações elaboradas acima, este trabalho acadêmico visa criar uma automação que traga como benefício a capacidade de remotamente controlar o funcionamento de sistemas residências, assim gerando um maior conforto, conveniência e segurança.

1.3 OBJETIVOS

1.3.1 OBJETIVOS GERAIS

O objetivo geral deste trabalho foca no desenvolvimento de uma automação residencial que permita o controle remoto de diferentes sistemas residenciais via redes sem fio. Sendo estes, a iluminação, o monitoramento de temperatura e segurança. Este último é composto de uma tranca eletromagnética que representa o controle de acesso da residência e um alarme para identificação de intrusos.

1.3.2 OBJETIVOS ESPECÍFICOS

Com o intuito de que o objetivo geral descrito acima possa ser atingido, existem os seguintes objetivos específicos que devem ser atingidos individualmente:

- Desenvolvimento do esquemático que unifique os diferentes sistemas de uma casa que são representados em uma maquete integrando o *Arduino Mega* e a Placa Relé responsável por acionar os sistemas.
- Desenvolvimento do código fonte em linguagem C que controla o micro controlador, *Arduino Mega*, que por sua vez, aciona e monitora os sistemas da casa;
- Desenvolvimento do código fonte da página e servidor HTML que é acessível via redes sem fio. Ambos, a página e o servidor são programados dentro do *Arduino*;
- Realização de testes de funcionamento visando melhorar o sincronismo dos sistemas ao *Arduino Mega* e a página HTML de acesso remoto.
- Testes com foco na identificação se o sistema de automação atende as funcionalidades propostas e melhoria das mesmas.

1.4 METODOLOGIA

O resultado do projeto proposto se resume na criação de um esquemático que interliga o *Arduino Mega* ao módulo relé que, por sua vez, ativa os diferentes sistemas propostos de uma residência e, finalmente, a disponibilização do acesso pela rede sem fio através de uma placa *Ethernet* conectada ao *Arduino* e um roteador. O resultado proposto é elaborado na figura 1.

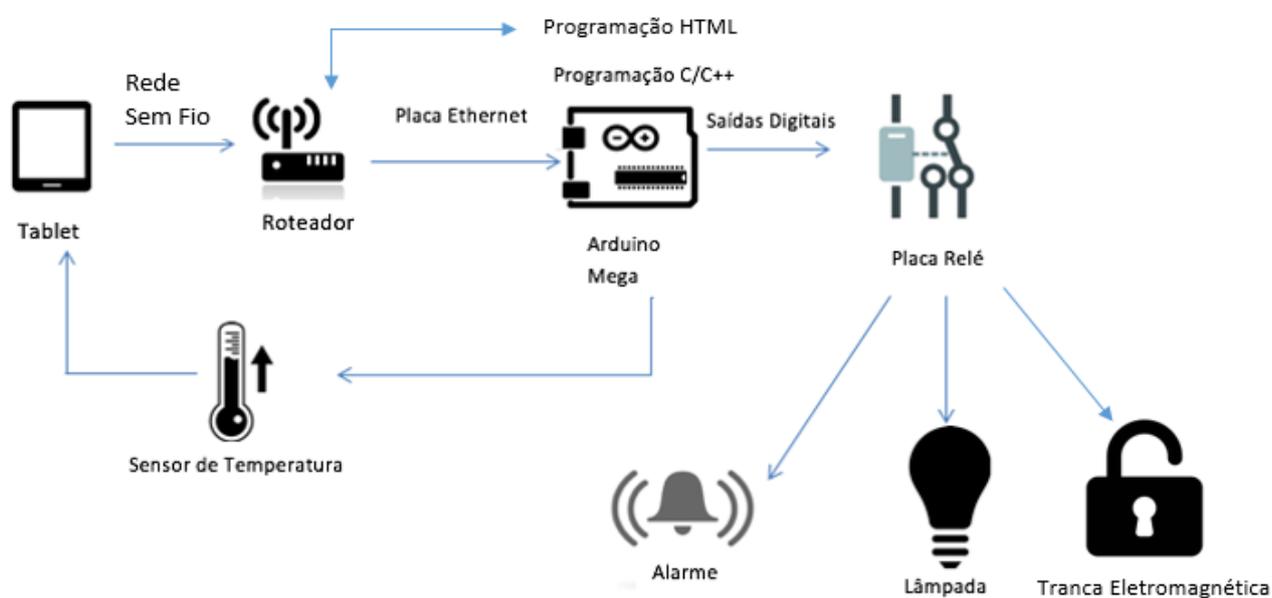


Figura 1 - Metodologia. Fonte: Autor

Na primeira fase, é feita a realização de pesquisas bibliográficas acerca do *Arduino Mega*, o funcionamento da placa de Ethernet, o funcionamento do módulo relé, linguagem de programação C/C++, linguagem de programação HTML, o sensor de temperatura, sensor de movimento e tranca eletromagnética.

Na segunda fase, o estudo e desenvolvimento da linguagem de programação C/C++ que controla o acionamento da placa relé através dos pinos analógicos e digitais de saída do Arduino, que por sua vez, ligará as lâmpadas, tranca e alarme individualmente.

Na terceira fase, o desenvolvimento da linguagem de programação C/C++ utilizada para realizar as medições de temperatura, cujos valores serão fornecidos por um sensor específico que se comunica com o micro controlador.

Na quarta fase, é realizado a integração do *Arduino Mega* ao modulo relé e este a cada um dos sistemas a serem automatizados. Os sistemas a serem automatizados no neste caso são: iluminação, tranca elétrica e o alarme (representado pelo conjunto sirene e sensor de presença).

Na quinta fase, é testado o acionamento de cada um dos sistemas mencionados no último parágrafo individualmente.

Na sexta fase, o desenvolvimento do software em linguagem de programação HTML que disponibiliza em conjunto com a placa de Ethernet e o roteador o acesso da casa inteligente através de redes sem fio.

Na sétima fase, são realizados os testes de acionamento através da página Web. Assim, é acionado pelo dispositivo remoto cada um dos sistemas residenciais.

Na oitava fase, são realizados os testes de monitoramento de temperatura remotamente. Nesta é acessado a página Web e verificado se o *Arduino* retorna os valores corretos de temperatura.

Na nona fase, são realizados testes visando buscar possíveis falhas ou melhorias com todos os sistemas integrados. Estes testes serão realizados acionando os diferentes sistemas do conjunto durante simulações através da interface final do sistema.

Na décima fase, é montada uma estrutura transportável, na qual todo o sistema é montado para demonstração do funcionamento deste projeto. Esta estrutura é composta de um Quadro VDI (Quadro fabricado em PVC antichama para abrigar dispositivos eletrônicos e fiação) no qual é abrigado o micro controlador, roteador e o relé *shield*. A estrutura transportável também inclui uma maquete feita em madeira, na qual são instalados os diferentes componentes automatizados da casa.

1.5 RESULTADOS ESPERADOS

O resultado final esperado é o desenvolvimento de uma automação prática e funcional, cujo objetivo é a possibilidade de ligar ou desligar e monitorar os sistemas mencionados acima que compõem habitações do nosso país. Modelo que possa ser melhorado de acordo com novas soluções apresentadas ao longo dos anos.

Ao mesmo tempo, também é esperado que essa automação atinja o objetivo destacado no último parágrafo de modo intuitivo, ou seja, transparente para o usuário final de modo que qualquer pessoa possa utilizar a interface final da aplicação Web com facilidade.

1.6 ESTRUTURA DO TRABALHO

Segue a descrição do modo sob o qual o trabalho está estruturado:

Capítulo 1: Apresenta a introdução do projeto, seus objetivos, motivação e metodologia.

Capítulo 2: É descrito todo o referencial teórico dos sensores, do Arduino Mega, da placa da Ethernet, e fundamentos básicos necessários.

Capítulo 3: Desenvolvimento do projeto, sendo este composto do modo como o funcionamento deste modelo de automação funciona.

Capítulo 4: Este capítulo é voltado para análise, teste, anotação, melhorias, detalhamento de erros enfrentados e como estes foram sanados.

Capítulo 5: Sendo o último, visa apresentar conclusões e sugestões de trabalhos futuros.

CAPÍTULO 2: REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico que vai subsidiar o desenvolvimento do sistema de automação proposto. Assim, é abordado os conceitos utilizados, métodos, fundamentos teóricos necessários, componentes e contextualização do uso destes.

2.1 AUTOMAÇÃO RESIDENCIAL

É caracterizada por um “conjunto de serviços proporcionados por sistemas tecnológicos integrados com o objetivo de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação” (MURATORI, JOSE. 2015).

A Associação Espanhola de Domótica define automação residencial como:

Sistemas que mediante o uso de equipamentos dispõem da capacidade para se comunicar interativamente entre eles e/ou com capacidade de seguir as instruções de um programa previamente estabelecido pelo usuário da residência e/ou com possibilidades de alterações conforme seus interesses. Em consequência, a domótica permite maior qualidade de vida, reduz o trabalho doméstico, aumenta o bem-estar e a segurança, racionaliza o consumo de energia e, além disso, sua evolução permite oferecer continuamente novas aplicações.

Deste modo, o conceito de automação residencial é baseado em instalações nas quais há a integração de sistemas residenciais aliado à capacidade de executar funções e comandos perante instruções programáveis. Os sistemas de uma residência que a automação residencial pode abranger são:

- Instalação Elétrica, que corresponde: iluminação, persianas e cortinas e gestão de energia;
- Sistema de segurança: alarmes contra intrusos, alarmes técnicos (fumaça, vazamento de gás, inundação), câmeras de segurança e controle de acesso;

- Sistemas de entretenimento: áudio e vídeo, som ambientes, imagens e sons sob demanda;
- Sistema de telefonia;
- Utilidades: irrigação, ar condicionado, aspiração central, aquecimento de água, bombas;

Simplificando, automação residencial consiste em controlar quaisquer dos itens descritos acima utilizando micro controladores como o *Arduino* presente neste projeto, cuja função é internalizar as preferências do usuário. E ao mesmo tempo, estes componentes residenciais devem ser monitoráveis e controláveis remotamente através do uso de recursos de comunicação, assim como redes sem fio presentes na solução proposta.

2.2 REDES SEM FIO

Uma rede sem fio é como seu próprio nome indica, uma rede na qual dois dispositivos eletrônicos podem trocar informações sem a necessidade de uma ligação física entre eles, permitindo se manter conectado mesmo se deslocando geograficamente. Atualmente existem diversos tipos de redes sem fio, entre eles o IEEE 802.11 (popularmente Wifi), 2G, 3G, 4G, etc.

O padrão 802.11 comumente conhecido como Wifi vem se tornando marcante no último século. Se trata de uma rede que funciona de 11 Mbps a centenas de Mbps composta de clientes representados por computadores portáteis, smartphones e os pontos de acesso que emitem e recebem sinais de radiofrequência recebidos pelos clientes, assim integrando estes a rede (TANEBAUM, ANDREW. 2011).

O problema do padrão 802.11 é que dependendo das “condições da rede sem fio, que variam até mesmo com pequenas mudanças no ambiente”, a rede pode sofrer interferências. Considerando que as frequências utilizadas para 802.11 são compostas de ondas de rádio, elas podem ser refletidas por objetos sólidos causando lentidão e dificuldade de acesso (TANEBAUM, ANDREW. 2011).

Assim, com o intuito de sanar o problema de pequena mobilidade oferecida pelas redes Wifi em 1991 foi implantado o 2G ou GSM (Global System For Mobile Communications) que

rapidamente se tornou o sistema mais usado do mundo, sendo o padrão que permite até hoje a multiplexação da banda, ou de forma mais simples múltiplas conversações ao mesmo tempo em uma única torre de celular. É este fator que garante que cada usuário de um celular possua um número de identificação individual, apesar de, todos os usuários da mesma operadora se comunicarem com a torre de dados sob o mesmo tipo de onda de rádio frequência.

Em 2001, empresas de telecomunicação começaram a fazer grandes investimentos na Terceira Geração de acesso à Internet, ou 3G. Sendo representado por sistemas que garantem acesso à Internet com velocidades acima de 384 Kbps. Com o surgimento deste houve uma grande revolução, pois a internet se tornou acessível através de qualquer lugar, gerando assim uma conectividade nunca vista antes (KUROUSE, JAMES & ROSS, KEITH. 2005).

Finalmente, nos últimos anos houve o surgimento do LTE (Long-Term Evolution) que “tem suas raízes na tecnologia 3G e possui potencial para alcançar velocidades superiores a 10 Mbits/s”. O LTE hoje presente no Brasil e no mundo revolucionou a Internet, com ele pessoas podem enviar e-mails, fazer streaming de vídeos, se conectar a redes sociais com grande facilidade e sem a lentidão apresentas pelas tecnologias anteriores (KUROUSE, JAMES & ROSS, KEITH. 2005).

É a conectividade possibilitada por tecnologias como o Wifi que permitirão os usuários controlar o sistema de automação proposto. Este controle é viável, pois o micro controlador *Arduino Mega* utilizado no projeto é conectado a uma placa de Ethernet que fornece os protocolos de comunicação necessários para se conectar remotamente.

2.3 MICRO CONTROLADOR ARDUINO MEGA

O *Arduino* é uma plataforma de desenvolvimento aberta chamada de *OpenSource* utilizada com muita frequência na atualidade devido ao seu baixo preço de custo e facilidade de uso. O *Arduino Mega* utilizado nesse projeto é uma placa de prototipagem baseada em um micro controlador ATmega2560 que possui 54 pinos de entrada e saídas digitais, sendo que 15 destas podem ser usadas como saídas PWM.

A alimentação do *Arduino* pode ser feita pela porta USB, *Universal Serial Bus*, como por uma alimentação externa com os valores de tensão entre 6 *Volts* a 20 *Volts*. Na sequência

são mostrados os pinos para conexão de *shields* (placas conectadas diretamente no *Arduino* estendendo suas funcionalidades) e módulos (placas conectadas aos pinos do *Arduino* indiretamente via o uso de fios) (ARDUINO.CC. 2016).

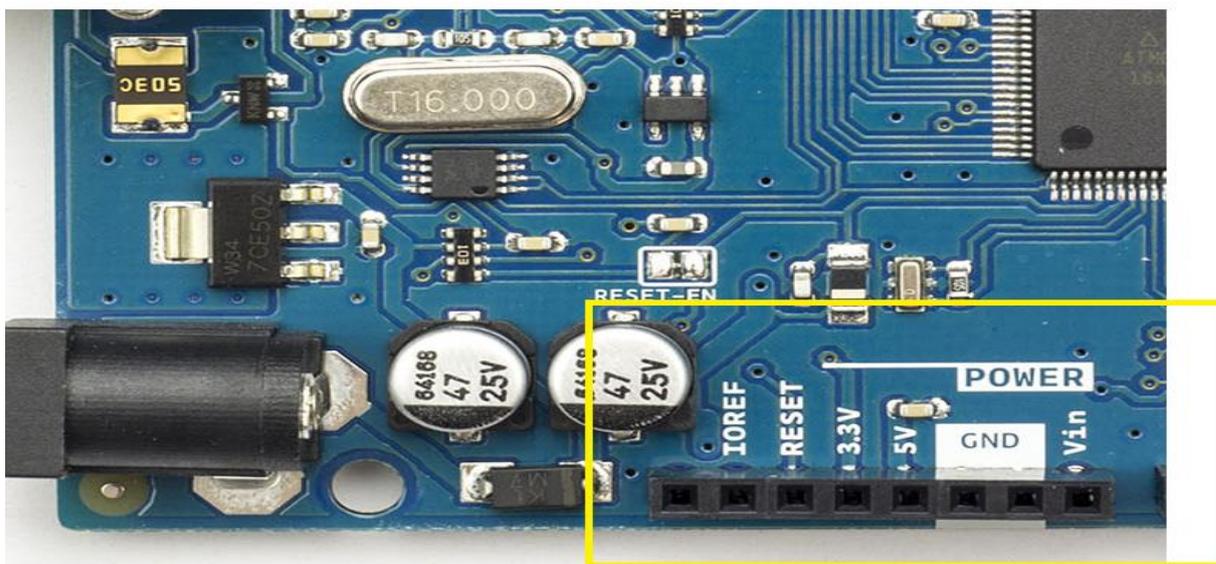


Figura 2. Entradas do Arduino. Fonte: Arduino.CC

A figura 2 mostra entradas e saídas do *Arduino*:

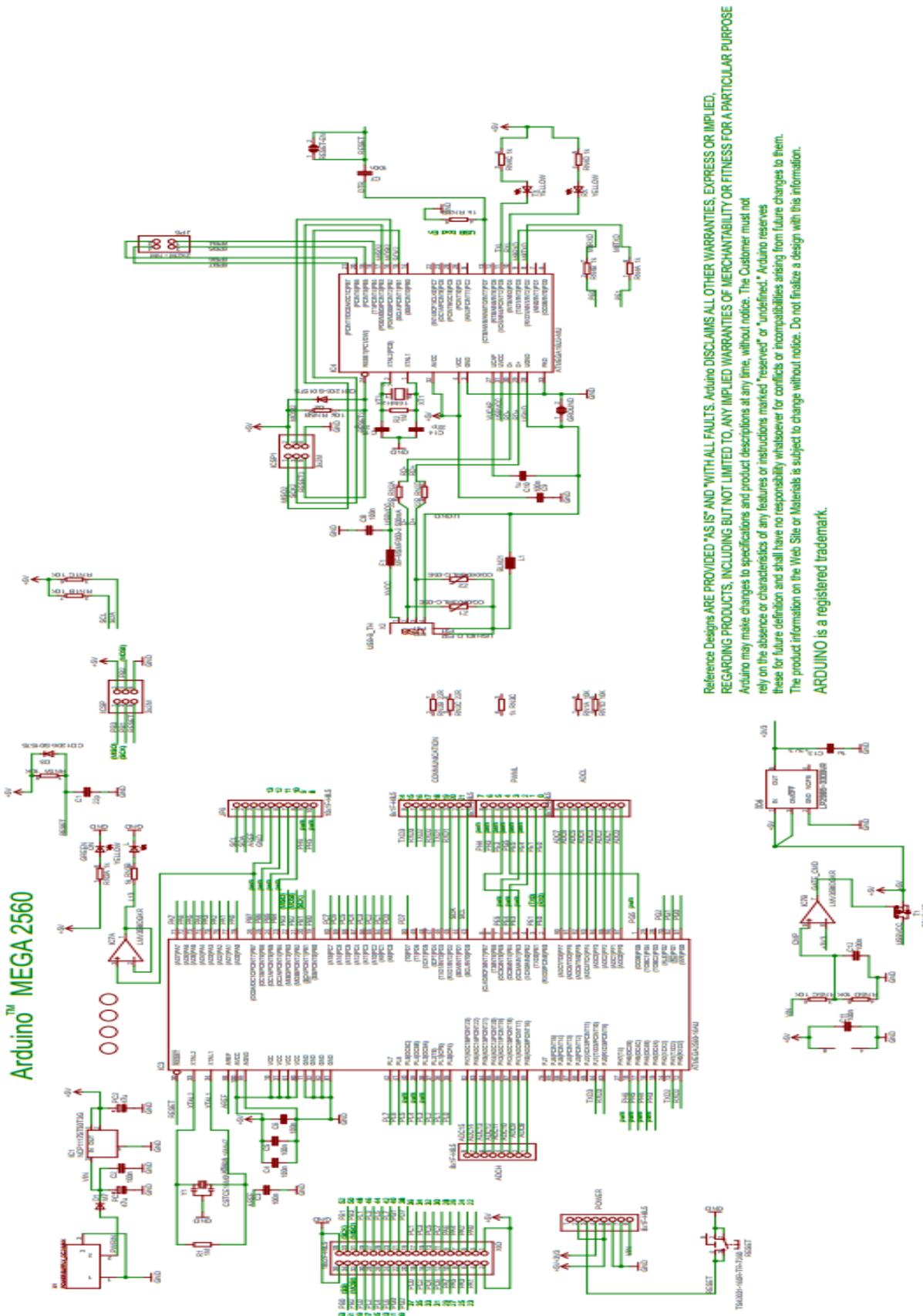
- IOREF: Fornece alimentação para *Shields* em até 3.3V;
- RESET: Entrada utilizada para fazer a placa voltar para as configurações de fábrica caso necessário;
- 3,3V: Pino de saída de 3,3v para alimentação de módulos externos com corrente máxima de 50 mA;

- 5V: Fornece uma tensão de saída de 5V para alimentação externa;
- GND: Terra;
- VIN: Pino para alimentar a placa usando um *shield* ou *jack*;

Na próxima página é mostrada a figura 3 contendo o esquema elétrico do *Arduino Mega*. Neste esquema é o micro controlador é dividido em três conjuntos, sendo eles o micro controlador principal, processador USB e a alimentação.

O micro controlador da placa é o ATMEGA2560 que possui 8 bits de arquitetura RISC, 256 KB de Flash, 8 KB de Ram e 4KB de EEPROM. Sendo que ele opera numa taxa de 16MHZ, uma velocidade considerável para os micros controladores disponíveis no mercado e a quantidade necessária para o projeto em questão. O processador do ATMEGA2560 é responsável por interpretar toda a informação que é fornecida para o *Arduino* pelos sensores, módulos e *Shields*, e assim tomar decisões lógicas de ações segundo o código programável pelo usuário em linguagem C/C++. O processador também é responsável por receber, enviar e processar os sinais recebidos do processador USB do micro controlador (ARDUINO.CC. 2016).

O processador USB presente é o Atmega16U2 e ele exerce a função de controlar a comunicação entre a plataforma e o computador no qual é compilado o código que rege o funcionamento do *Arduino*. Este é vital, pois o micro controlador não possui compatibilidade para conexões USB, sendo assim este converte os dados recebidos pela USB para um sinal serial que é repassado para o ATMega2560 (ARDUINO.CC. 2016).



Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information. ARDUINO is a registered trademark.

Figura 3. Esquemático Elétrico - Arduino Mega. Fonte: ARDUINO.CC

Na figura 4 é apresentada o *Arduino Mega* e nesta são nomeados os diferentes componentes da estrutura do mesmo.

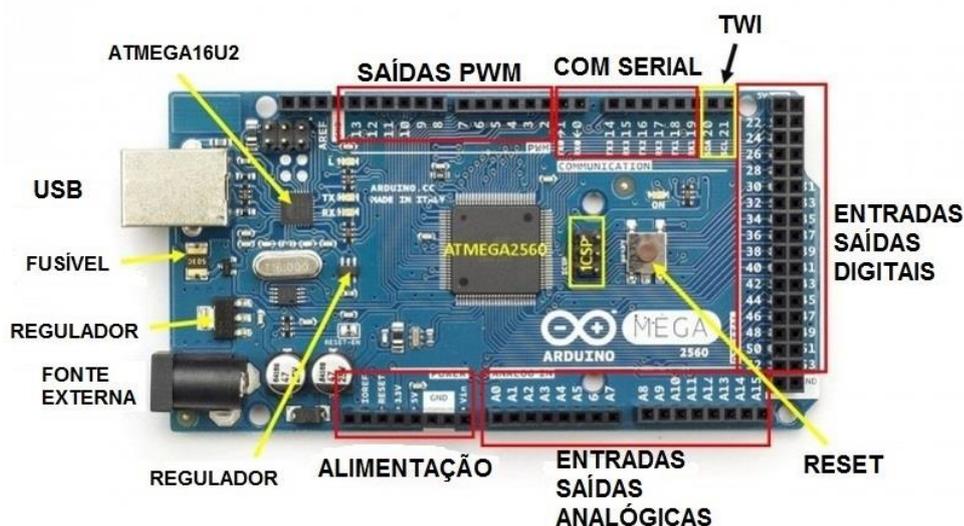


Figura 4. Entradas e Saídas do Arduino. Fonte: Adaptado de ARDUINO.CC

Neste projeto utilizaremos majoritariamente as entradas e saídas digitais do *Arduino* para conectar o *shield Ethernet* que é tratado logo a frente, o sensor de temperatura, as entradas da placa de relé. A saída de 5V e o terra ainda serão utilizados para energizar estes componentes.

Finalmente, como o *Arduino* é acomodado neste projeto dentro de um quadro VDI mencionado anteriormente, e considerando o fato de que o micro controlador necessita de alimentação própria é utilizado uma fonte externa de alimentação.

2.4 SHIELD ETHERNET

O sistema de automação que é desenvolvido nesse projeto é baseado na comunicação com dispositivos elétricos e eletrônicos remotamente. Com intuito de atingir esse propósito é necessário que o micro controlador escolhido, o *Arduino*, possa se conectar a um roteador sem

fio, que por sua vez, disponibiliza o acesso ao mesmo de modo remoto através do uso de portas de comunicação específicas cujo funcionamento é explicado mais a frente. Continuando, o *Arduino* sozinho é incapaz de se conectar ao dito roteador sem fio, assim há a necessidade do uso de um *Shield Ethernet*.

O *Shield Ethernet* em questão (figura 5) utiliza o controlador *Ethernet Wiznet W5100*, este é escolhido devido sua alta compatibilidade com o micro controlador tanto fisicamente como no seu modo de conduzir a sincronização lógica. De modo análogo ao *Arduino* que possui toda sua documentação de funcionamento aberta, a placa de *Ethernet Wiznet* possui todos os códigos e bibliotecas de funcionamento necessárias disponibilizadas na Internet o que apenas realçando sua escolha.

O *Ethernet Shield* se conecta ao *Arduino* de forma empilhável, significando que todos os pinos e conectores do micro controlador são conectados diretamente pela mesma. O acesso às portas não utilizadas é transportado, sendo assim possível conectar outro *shield* em cima dele. Esta funcionalidade é vista na figura 5.



Figura 5 – *Arduino Ethernet Shield*. Fonte: barretuino.com.br

Na figura 6 é visto um esquemático estrutural da placa de Ethernet utilizada. Ela é gerida pelo controlador *Wiznet W5100*, um chipset com compatibilidade ao IEEE 802.3 e o 802.3u fornecendo desse modo acessibilidade cabeada a conexões de rede a velocidades de 10 a 100 Mb/s. No esquemático ainda se destacam um botão reset utilizado para reestabelecer as configurações de fábrica, uma entrada para cartões micro SD, a entrada para cabos de rede RJ45 e, finalmente leds indicadores TX e RX para sinalizar informações transmitidas e recebidas (WIZNET. 2008).

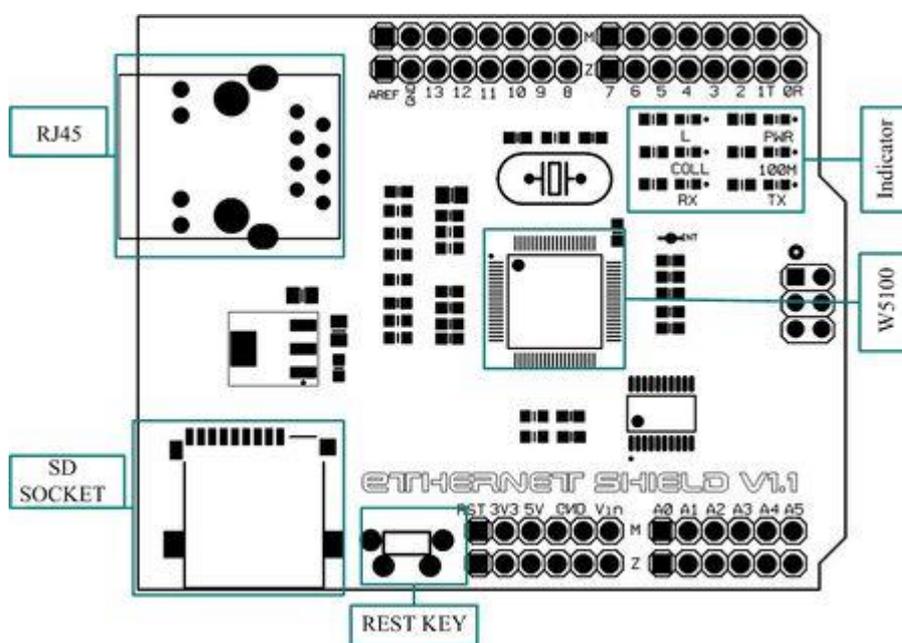


Figura 6. Esquemático do Shield Ethernet. Fonte: elec.com

O *Ethernet Shield* fornece uma rede capaz de se comunicar usando os protocolos TCP e UDP da camada de transporte utilizando seu conector de rede RJ-45 para a conexão física. “Os serviços baseados em TCP e UDP rastreiam várias aplicações” e com intuito de diferenciar estas usam um cabeçalho (figura 7) no qual “em cada segmento ou mensagem, há a informação da porta de origem e destino de cada aplicação”. No caso deste projeto a porta de destino é o micro controlador que atua como servidor nas comunicações (INTERSABERES. 2014).



Figura 7. Cabeçalho TCP/UDP. Fonte: www.seucurso.com.br

O conector RJ-45 é ligado ao roteador wireless usando um cabo de rede padrão, o roteador finalmente se comunica via *sockets* (portas de comunicação mencionadas) com o *Arduino* e este com um smartphone ou qualquer ou dispositivo capaz de acessar a interface web desenvolvida em HTML. Resumindo, o *shield Ethernet* é responsável por transmitir os dados para os sensores e portas do relé que acionam diferentes componentes residenciais.

Seguem as informações técnicas do Wiznet W5100:

- Chipset Wiznet5100;
- Transformador de linha integrado com módulo *Power over Ethernet* (PoE) disponível;
- Soquete de cartão de memória micro-SD;
- Buffer interno de 16k;
- Velocidade de conexão de 10/100Mb;
- Tensão de operação de 3,3 – 5V;

2.5 ACIONAMENTO DE CARGAS ELÉTRICAS COM RELÉS

Neste projeto de automação é realizado o acionamento de cargas elétricas por um módulo relé composto de oito canais operando sob uma voltagem de 5 *volts*. Um relé é essencialmente segundo Newton Braga “um dispositivo comutador eletromecânico” composto de um eletroímã representado por uma bobina e uma armadura móvel como pode ser visto na figura 8. Esta armadura tem a finalidade de fechar ou abrir um circuito quando atraída magneticamente pelo campo magnético criado pela bobina ao ser eletrificada (BRAGA, NEWTON. 2012).

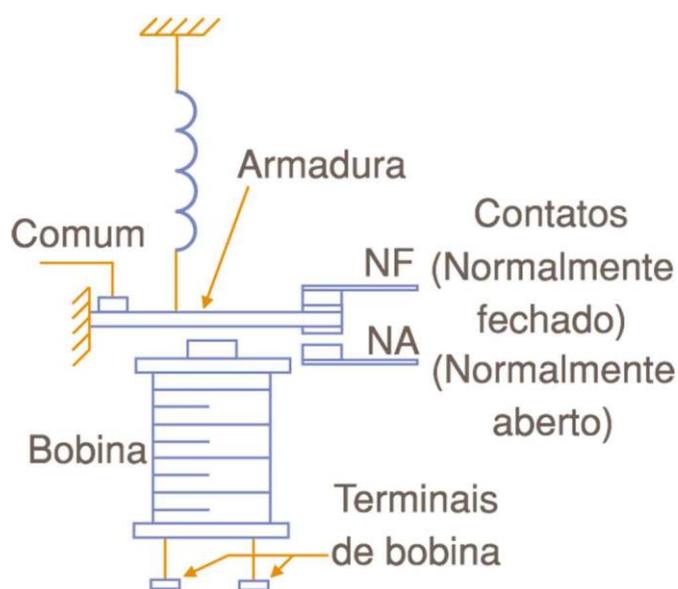


Figura 8. Esquemático do Relé. Fonte: www.mecatronicaatual.com.br

O módulo relé escolhido é o Keyes 8R1A e este conecta à porta de 5 *volts*, ao terra do *Arduino*, e aos pinos digitais do *Arduino*. Sendo que cada pino digital do micro controlador conectado ao módulo ativa quando enviado um pulso, um dos oitos relés de acordo com o dispositivo residencial que se deseja ligar.

O esquema elétrico das conexões entre o *Arduino* e a placa relé é elaborado na figura 9. Como pode ser percebido, para cada relé que se deseja ativar é conectado um *jumper* (fio específico para ligação de saídas e entradas de micro controladores e seus módulos) na saída digital do *Arduino*, cuja função é enviar um sinal que ativa o relé desejado.

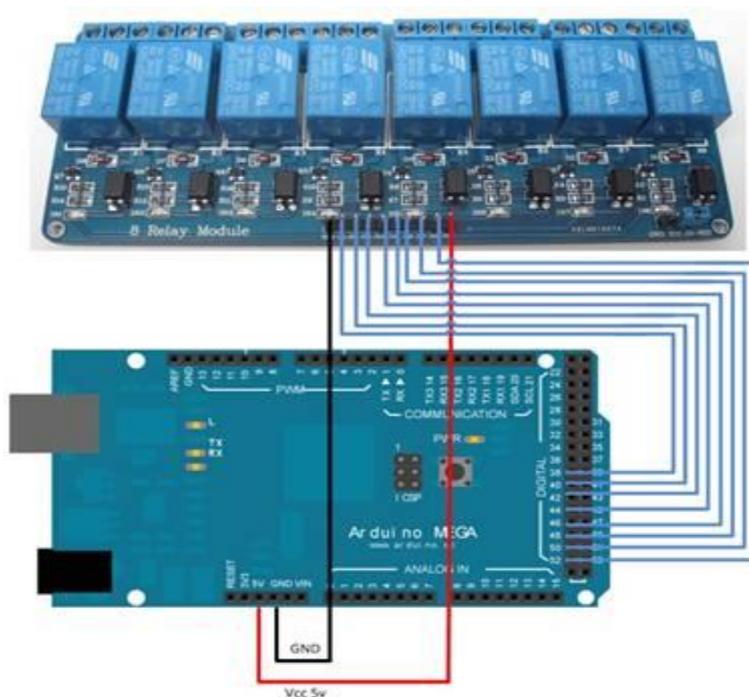


Figura 9. Módulo Relé. Fonte: www.codeproject.com

As características de funcionamento do módulo em questão são:

- Tensão de operação de 5 volts;
- Tensão de sinal: Padrão lógico TTL;
- Corrente de operação de 15 até 20 miliampéres;
- Os contatos do relé suportam até 250 volts operando a 10 A;
- Indicação por LED do status de funcionamento de cada relé;
- Tempo de reposta de 5 até 10 milissegundos;
- Dimensões: 140mm x 54mm x 18 mm

2.6 MEDIÇÃO E MONITORAMENTO DE TEMPERATURA

Com o intuito de disponibilizar para o usuário da automação um monitoramento contínuo da temperatura ambiente do interior da residência é utilizado nesse projeto um sensor de temperatura de umidade DHT11 produzido pela AOSONG na China. O DHT11 se trata de um sensor digital que permite realizar leituras de temperaturas entre 0 a 50 graus Celsius. Um esquemático do sensor é visto na figura 10 (AOSONG. 2015).

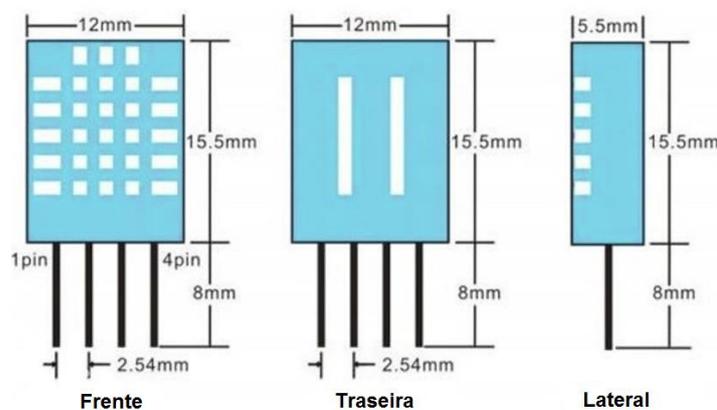


Figura 10. DHT11. Fonte: AOSONG

O sensor de temperatura do DHT11 é formado por um termistor NTC, dispositivo elétrico que possui uma resistência variável com seus valores sofrendo alterações de acordo com a temperatura sob o qual é submetido.

A ligação do sensor ao *Arduino* é feita usando três de seus pinos. Sendo um para voltagem de 5 volts, dados e negativo. O pino de dados do DHT11 é conectado a entrada analógica do micro controlador que converte os dados enviados pelo DHT11 em lotes de 8 bits para valores decimais de temperatura. O processo de conversão dos dados é invisível para o usuário, pois o sensor possui uma biblioteca pública específica para o *Arduino*, assim o micro controlador realiza o processo de tratamento de dados automaticamente. A figura 11 ilustra o processo de conexão do sensor no *Arduino* (AOSONG. 2015).

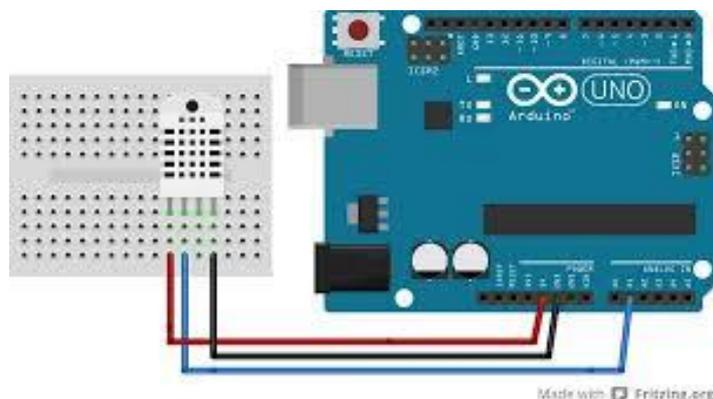


Figura 11. Conectando DHT11 no Arduino. Fonte: www.arduino.com.br

Abaixo seguem as especificações técnicas do DHT11:

- Alimentação: 3 a 5 volts;
- Corrente: 200uA a 500mA, em stand by de 100uA a 150 uA;
- Faixa de medição de umidade: 20 a 90% UR;
- Faixa de medição de temperatura: 0° a 50° C;
- Precisão de medição de umidade: ± 5 UR;
- Precisão de medição de temperatura: ± 2 °C;
- Tempo de resposta: <5s;

2.7 CONTROLE DE ACESSO FÍSICO

A automação proposta possui entre os seus objetivos controlar o acesso de pessoas a residência. Assim, entre os componentes residenciais que se deseja controlar de modo remoto tem-se uma tranca elétrica. Tranca habitualmente utilizada nas casas para controlar o acesso das pessoas que entram e saem das mesmas.

ponto:ciência

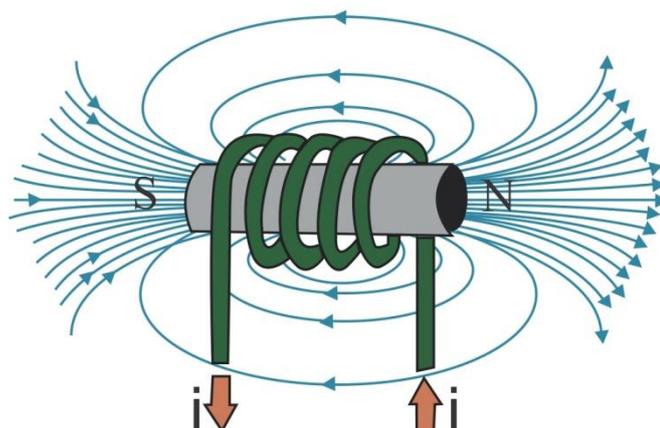
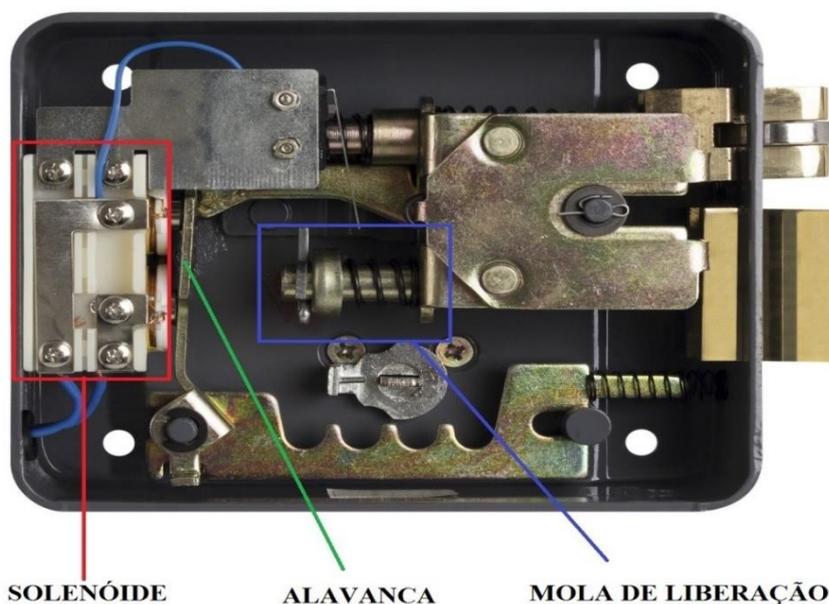


Figura 12. Campo Eletromagnético do Solenoide. Fonte: www.ponto.ciencia.com.br

A fechadura elétrica utilizada nesse caso é da marca STAM. Esta possui um solenoide (condutor elétrico na forma de espiral) no seu interior. O solenoide ao ser percorrido por uma corrente “i” externa gerada por uma fonte de 12 volts cria um pequeno campo eletromagnético (figura 12).

Na sequência, o campo eletromagnético inicia o processo de abertura da tranca ao retrainir uma alavanca. Como consequência da retração da dita alavanca, uma mola de liberação é acionada gerando a abertura da tranca, cujo interior é visto na figura 13 com suas devidas partes nomeadas.



SOLENOÍDE

ALAVANCA

MOLA DE LIBERAÇÃO

Figura 13. Interior da Tranca Elétrica. Fonte: adaptado de www.stam.com.br

As características da fechadura elétrica são:

- Cilindro: Fixo com 40 mm de comprimento;
- Material: Aço plástico ABS e latão;
- Ajustes: Na pressão da mola;
- Frequência: 50/60 Hz;
- Alimentação: 12 volts;
- Consumo: 15 *Watts*;
- Tratamento anticorrosivo;

2.8 ALARME

O alarme de segurança deste projeto de automação é formado pelo conjunto de dois dispositivos, um sensor de presença e uma sirene.

2.8.1 SENSOR DE PRESENÇA

A vertente de segurança do projeto de automação é representada por uma solução na qual um sensor de movimento aciona uma sirene quando o espaço protegido por este é invadido por um corpo não desejado. O sensor de movimento utilizado é o Qualitronix Qa25.

O Qualitronix Qa25 é um sensor de presença infravermelho passivo chamado de sensor PIR (Passive Infrared Sensor), sensor passivo infravermelho em português. Sensores infravermelhos tem seu funcionamento baseado na detecção de energia infravermelha liberada, energia que é emitida por todos os corpos vivos (FOGAÇA, JENNIFER. 2015).

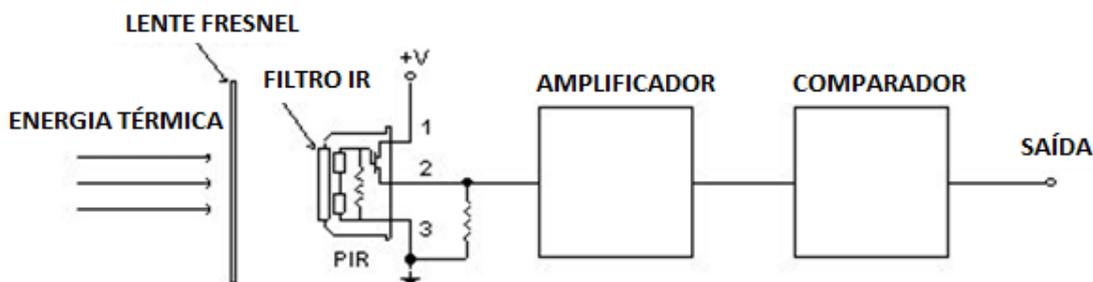


Figura 14. Sensor PIR. Fonte: robotgstation.com

O processo se inicia com a energia térmica atravessando uma lente Fresnel como pode ser visto na imagem da figura 14. A lente Fresnel condensa a luz e torna a gama de raios infravermelhos visível para o sensor infravermelho. Esses comprimentos de luz são por sua vez convertidos em corrente elétrica cujos valores são amplificados e logo depois enviados para um módulo de comparação. Finalmente, o sensor gera uma saída específica quando grandes variações de emissão infravermelha são percebidas. O processo de condensação da luz na lente Fresnel é semelhante ao da figura 15. (ADA, LADY. 2015)

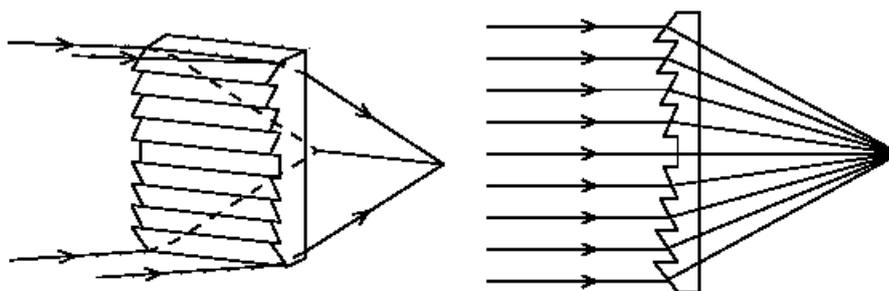


Figura 15. Lente Fresnel. Fonte: www.bhlens.com

No caso do sensor Qualitronix escolhido, uma vez que este identifica variações infravermelhas, ele realiza a abertura de um relé interno. Este relé é ligado a uma fonte e esta a uma sirene piezoelétrica, assim a sirene é ativada alarmando da existência de corpos vivos indesejados no recinto. O esquema de ligação do sensor na rede elétrica e do seu retorno na fonte da sirene é encontrado na figura 16.

Finalizando o processo de automação deste alarme, a alimentação do sensor é conectada a uma das entradas da placa relé abordada no item 2.4 deste capítulo, assim possibilitando ao usuário controlar a ativação e desativação dele via redes sem fio.

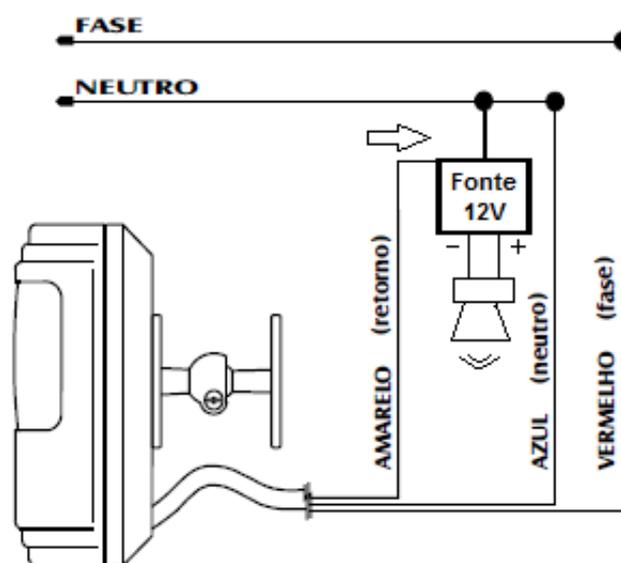


Figura 16. Esquemático Sensor-Sirene. Fonte: www.fazedores.com

As características do Qualitronix Qa25 seguem:

- Suporta tensões de 127 ou 220 volts;
- Monitora uma área de 180° graus na horizontal
- Monitora uma área de 60° graus na vertical;

- O tempo de ativação da sirene pode ser programado entre 15 a 8 minutos;

2.8.2 SIRENE PIEZOELÉCTRICA

O sensor de presença usado ativa uma sirene piezoelétrica, sendo que a sirene em questão é uma DNI 4042 de 120 decibéis cujo funcionamento ocorre sob uma tensão de 12 volts e uma corrente de 0,25 Amperes (figura 17).



Figura 17. Sirene DNI 4042. Fonte: www.dni.com.br

As sirenes piezoelétricas são vastamente utilizadas em alarmes, isso ocorre pois são fabricadas de materiais piezoelétricos. Estes materiais apresentam a característica de se deformarem uniformemente quando são polarizados. (Deformação na figura 18). Assim, estas deformações em altas frequência criam um som associado pelos seres humanos como indicador de eventos indesejados.

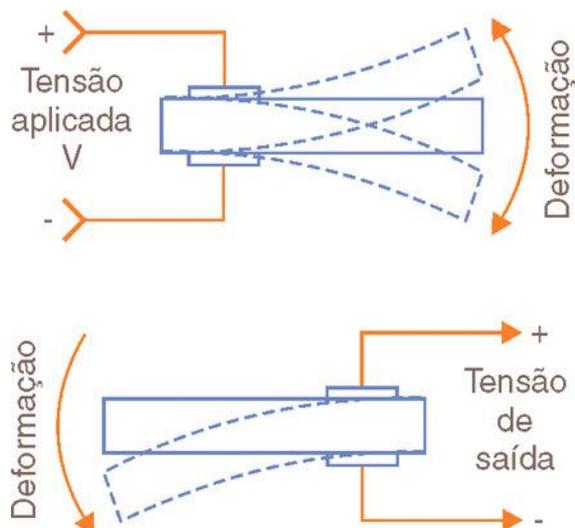


Figura 18. Deformação piezoelétrica. Fonte: www.sabereletronica.com.br

2.9 SOFTWARE: IDE DO ARDUINO

O ambiente de desenvolvimento de software do micro controlador *Arduino* é ideal para este projeto, pois através dele é fácil ser realizado o processo de programação e envio dos dados para o mesmo. O software é compatível com todos os modelos de *Arduino* disponíveis no mercado.

Como pode ser visto na figura 19 se trata de um ambiente de desenvolvimento completo no qual é possível programar o micro controlador na linguagem de programação C/C++, validar o código criado fazendo a compilação do mesmo, ter detalhamento de eventuais erros, abrir projetos básicos pré-carregados e fazer melhoria destes, entre outros recursos.

Entre eles se destaca um monitor serial no qual se podem ver variáveis obtidas a partir de sensores externos, assim eliminando a necessidade de um visor LCD externo para a realização de testes por exemplo (ARDUINO. CC. 2016).

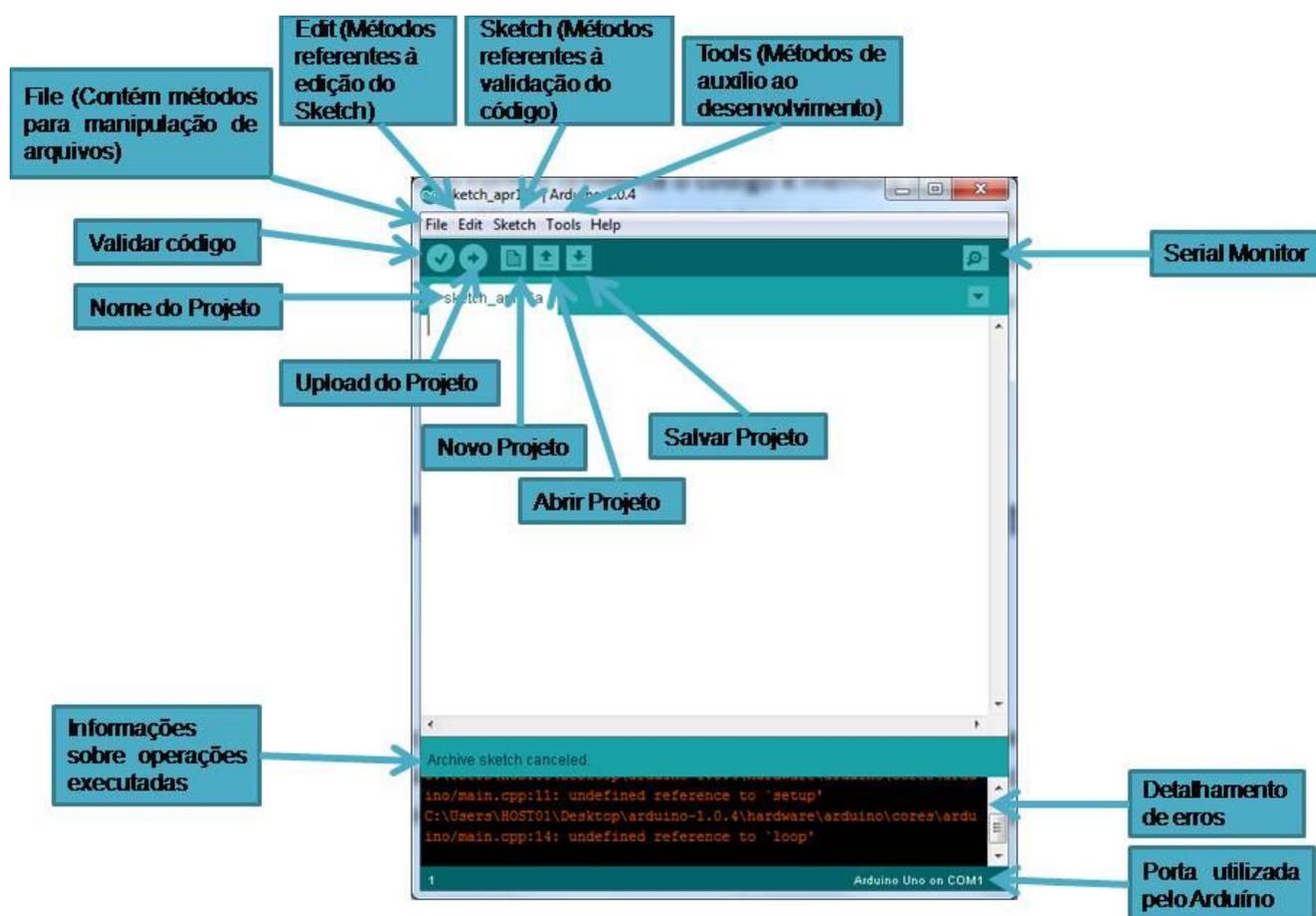


Figura 19. IDE do Arduino. Fonte: joaoshmitt.wordpress.com

O grande fator que torna esse ambiente de desenvolvimento em uma experiência única são as bibliotecas dos módulos e *Shields* vendidos para o micro controlador. Estas bibliotecas fazem com que o uso destes seja natural para o usuário porque a linguagem de programação fornecida por elas é a mesma do *Arduino*. Há ainda a possibilidade de adicionar bibliotecas de funcionamento pessoais, tornando assim possível controlar uma gama de acessórios para todos os tipos de aplicações (ARDUINO.CC. 2016).

Finalmente, a plataforma passa por atualizações periodicamente e é compatível com os sistemas operacionais Windows, Mac OS e Linux que representam os mais usados do mercado na atualidade.

2.10 TESTE DE CONFIABILIDADE

Nesse projeto é utilizado um sensor de presença externo cuja funcionalidade quando ativado pelo usuário é identificar a presença de corpos indesejados na área em que este é instalado. Por se tratar de um dispositivo que visa garantir segurança residencial é interessante testar a confiabilidade apresentada por este.

Com intuito de se testar a confiabilidade do funcionamento desta solução de segurança é necessário a realização de um procedimento de amostragem do funcionamento. Segundo FONSECA, JAIRO & MARTINS, ANDRADE, amostra é o subconjunto de uma população que se deseja analisar tendo em vista que certos sistemas possuem uma quantidade de variáveis muito extensa, se tornando inviável testar todas existentes. Neste caso é impossível testar a solução de segurança ao longo de um ou mais anos, período que representa o uso desta intervaladamente. Assim, o processo de amostragem garante a visualização dos resultados possíveis do produto analisando uma quantidade menor de tempo.

Neste caso, a população é representada por uma quantidade de horas nas quais o sensor de presença está ativado e deve apresentar alertas no caso de intrusos. Para estabelecer a quantidade de horas mínimas sob as quais o sistema deve ser testado na busca de falhas, a seguinte fórmula de amostragem é utilizada:

$$n = \frac{z^2 x \sigma^2 x N}{d^2(N-1) + Z^2 \sigma^2} \quad (1)$$

A equação 1 é uma fórmula para amostragem de dados intervalados onde Z é nível de confiança desejado, σ é desvio-padrão, N o tamanho da população e d o erro amostral. Finalmente, o resultado fornecido é um número “ n ” de horas de teste sob as quais o sistema deve ser testado intervaladamente, e após esses testes é gerada a porcentagem que representa o nível de confiança do mesmo.

Finalmente, estes testes são realizados segundo uma metodologia que é explicada no capítulo 4 deste material e visa simular o uso de sistemas de segurança mais próximo da realidade.

CAPÍTULO 3: DESENVOLVIMENTO

Este capítulo visa mostrar o desenvolvimento necessário para o processo de construção da automação residencial proposto no resumo deste material.

3.1 DESCRIÇÃO DO SISTEMA PROPOSTO

O sistema proposto busca a criação de um protótipo de automação com base no uso do micro controlador *Arduino Mega* que disponibiliza o acesso remoto via o uso de uma aplicação HTML e o *Ethernet Shield* mencionado no referencial teórico. Possibilitando assim, o acionamento de atuadores que controlam a iluminação, monitoramento de temperatura, abertura de uma tranca eletromagnética, e ativação de um sensor de movimento que em conjunto com uma sirene integram um alarme.

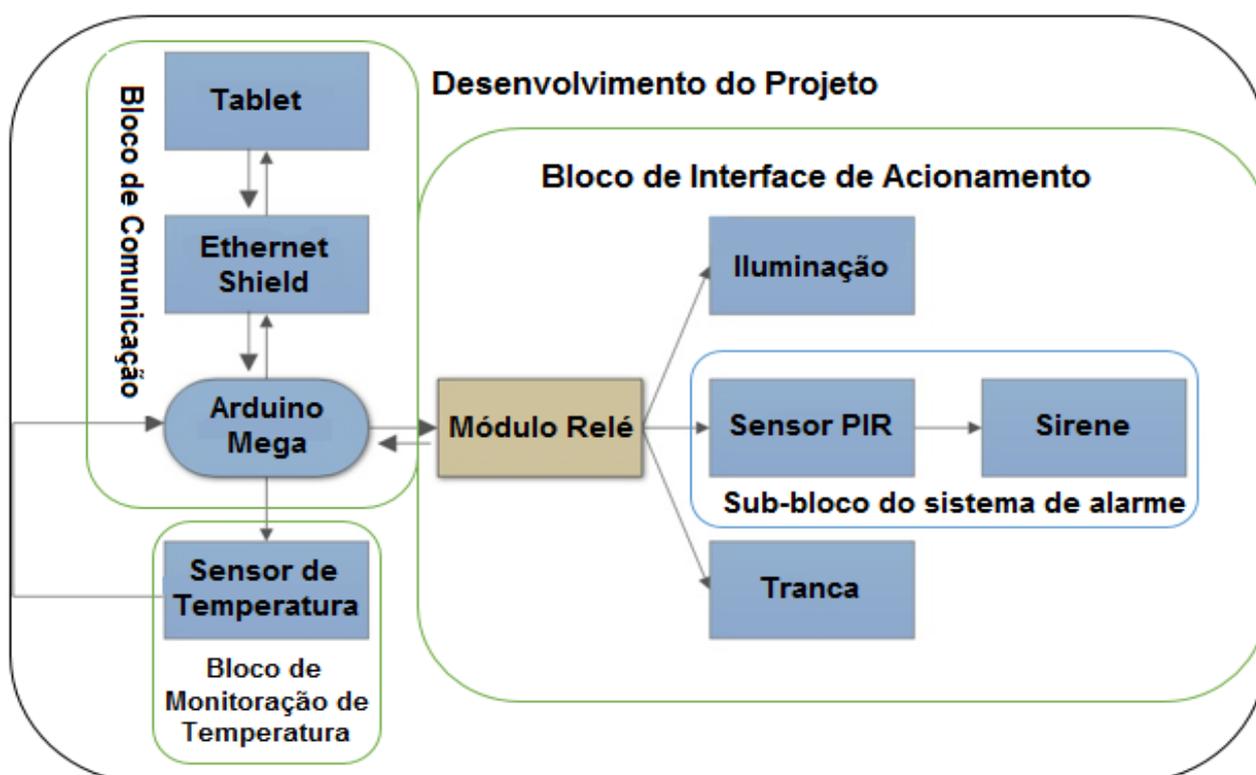


Figura 20. Fluxograma do Protótipo de Automação. Fonte: Autor.

O fluxograma da figura 20 ilustra de forma breve a lógica funcional da integração dos componentes que permitem que este projeto de automação atinja os objetivos específicos elencados no item 1.3.2 deste material. Objetivando uma explicação transparente de como esta integração ocorre, o capítulo do desenvolvimento é dividido nas seguintes partes:

✓ *Bloco de Comunicação*

1. *Configuração do Arduino Mega e Ethernet Shield;*
2. *Configuração do Roteador;*
3. *Servidor Web;*

✓ *Bloco da Interface de Acionamento*

1. *Configuração e controle do Módulo Relé e com este:*
2. *Acionamento das Lâmpadas;*
3. *Acionamento da Tranca Eletromagnética;*
4. *Acionamento do Alarme;*

✓ *Bloco da Monitoração de Temperatura;*

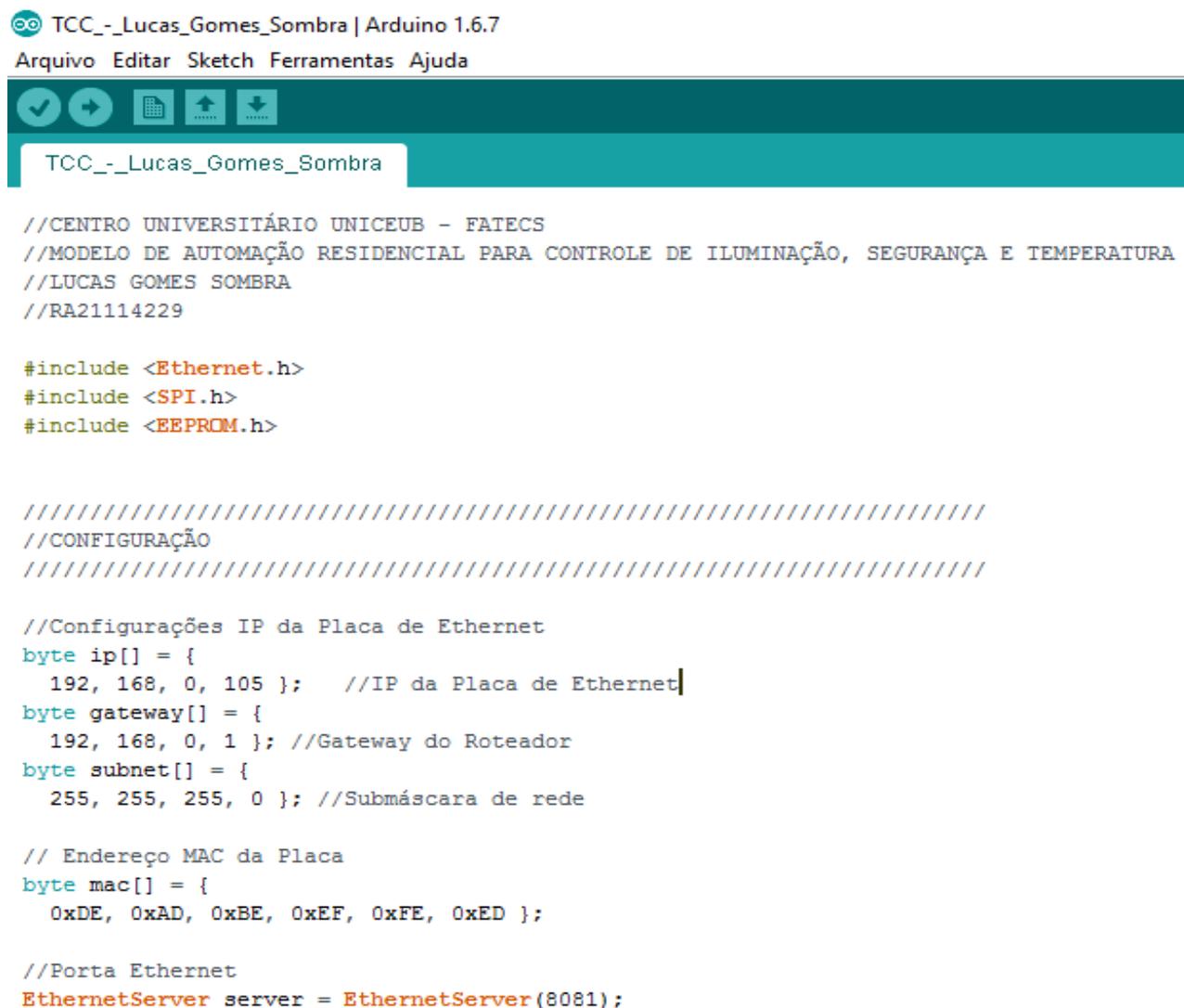
✓ *Ornamentação do Protótipo Final na Maquete Proposta*

3.2 BLOCO DE COMUNICAÇÃO

3.2.1 CONFIGURAÇÃO DO ARDUINO MEGA E ETHERNET SHIELD

O primeiro passo do projeto de automação em questão é a configuração da conexão do *Arduino Mega* com a placa de *Ethernet*, cuja funcionalidade é criar uma conexão através do uso de *sockets* ou portas de transporte de dados UDP e TCP como explicado anteriormente no referencial teórico. Para que seja realizado tal configuração se deve conectar o micro controlador ao computador via a entrada lógica de dados USB e implementar o código de programação em linguagem C na IDE (Interface Programável do *Arduino*) fazendo referência às bibliotecas de funcionamento da placa citada.

A figura 21 contém uma cópia do código desenvolvido para gerir a conexão da placa de *Ethernet* com o *Arduino*.



```

TCC_-_Lucas_Gomes_Sombra | Arduino 1.6.7
Arquivo Editar Sketch Ferramentas Ajuda

TCC_-_Lucas_Gomes_Sombra

//CENTRO UNIVERSITÁRIO UNICEUB - FATECS
//MODELO DE AUTOMAÇÃO RESIDENCIAL PARA CONTROLE DE ILUMINAÇÃO, SEGURANÇA E TEMPERATURA
//LUCAS GOMES SOMBRA
//RA21114229

#include <Ethernet.h>
#include <SPI.h>
#include <EEPROM.h>

////////////////////////////////////
//CONFIGURAÇÃO
////////////////////////////////////

//Configurações IP da Placa de Ethernet
byte ip[] = {
  192, 168, 0, 105 }; //IP da Placa de Ethernet
byte gateway[] = {
  192, 168, 0, 1 }; //Gateway do Roteador
byte subnet[] = {
  255, 255, 255, 0 }; //Submáscara de rede

// Endereço MAC da Placa
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

//Porta Ethernet
EthernetServer server = EthernetServer(8081);

```

Figura 21. Configuração Placa de Ethernet. Fonte: Autor.

No código mostrado na figura 21 é encontrado a adição da biblioteca da placa de rede na IDE com a linha de código `#include <Ethernet.h>`. Esta é basicamente responsável por notificar o micro controlador que esta placa está sendo usada e que os métodos usados para controlar a mesma estão disponíveis na mesma pasta, na qual o IDE foi instalado na máquina do utilizador dentro de uma subpasta específica com o nome Ethernet.

Continuando, é necessário configurar os valores de rede que a placa assumirá quando conectada com o roteador responsável por disponibilizar o acesso remoto. Estas configurações variam de acordo com o roteador, sendo assim necessário verificar qual o espaço de endereçamento aceito para o IP, gateway e sub-máscara de rede em cada caso.

Desta forma, define-se uma variável chamada “*byte mac*” que estabelece o valor MAC a ser assumido. O valor MAC é um endereço hexadecimal que cada placa de rede deve possuir para que possa realizar “o endereçamento único, não-ambíguo das interfaces dentro de uma LAN”. Ele é de grande importância para este projeto, pois é a associação dele com a porta de comunicação definida com *EthernetServer(8081)* que garantem que todos os dados que trafeguem pelo roteador pela porta 8081 sejam endereçados apenas para o *Arduino* com este MAC específico e o cliente representado por um smartphone ou qualquer outro dispositivo com acesso a redes sem fio (OLIFER, NATÁLIA. 2008).

Na sequência é demonstrado como é configurado o roteador em questão para aceitar, identificar e encaminhar os dados enviados pela porta 8081 para o micro controlador e o cliente de acesso remoto.

3.2.2 CONFIGURAÇÃO DO ROTEADOR

O roteador *wireless* utilizado é o RE063 da fabricante brasileira Multilaser. Vale a pena ressaltar que pode ser utilizado qualquer roteador sem fio com compatibilidade aos protocolos UDP e TCP. O fator de importância é que o produto usado seja capaz de realizar o roteamento estático do *Arduino Mega*, ou seja, assegurar que toda vez que o micro controlador se conectar à rede, este assumo o mesmo endereço de IP. No nosso caso 192.168.0.105, como mencionado no item 3.2. A figura 22 mostra a atribuição estática realizada no Multilaser.

MULTILASER
TECNOLOGIA E TRANSFORMAÇÃO

Configurações Avançadas | Configurações Sem Fio | Servidor DHCP | Servidor Virtual | Configurações de Segurança | Configurações de Roteamento | Ferramentas de Sistema

Servidor DHCP | **Lista Cliente DHCP**

Atribuição Estática

Endereço de IP: 192.168.0.105

Endereço MAC: DE:AD:BE:EF:FE:ED

NO.	Endereço de IP	Endereço MAC	Deletar
1	192.168.0.105	DE:AD:BE:EF:FE:ED	<input type="button" value="Deletar"/>

Figura 22. Atribuição Estática do Roteador. Fonte: Autor.

MULTILASER
TECNOLOGIA E TRANSFORMAÇÃO

Configurações Avançadas | Configurações Sem Fio | Servidor DHCP | Servidor Virtual | Configurações de Segurança | Configurações de Roteamento | Ferramentas de Sistema

Encaminhamento Intervalo de Portas | Anfitrião DMZ | **Configurações UPNP**

Encaminhamento intervalo de Portas configura serviços públicos na sua rede, tais como servidores de web, servidores ftp, servidores de e-mail, e outras aplicações especializadas de Internet. Quando você configurou um serviço, então as solicitações de comunicação da Internet para a Porta WAN do seu roteador serão convertidas para o endereço especificado do IP LAN.

NO.	Porta de início- Porta de fim	LAN IP	Protocolo	Habilitar	Deletar
1.	8081-8081	192.168.0.105	Ambos ▼	<input type="checkbox"/>	<input type="checkbox"/>

Figura 23. Encaminhamento de Porta. Fonte: Autor

Entretanto, o processo de comunicação não é concluído apenas com a atribuição estática do IP, pois esta apenas garante que o micro controlador se conecte e transmita informações sempre com o IP definido, mas para que o roteador encaminhe os dados enviados do *Arduino* para um cliente sem fio através da porta 8081 é necessário a alocação desta. Ou seja, todas as comunicações para esta porta sempre devem ser redirecionadas para o IP do *Arduino*, o servidor nesse caso.

Na figura 23 pode ser visto o processo de alocação ou encaminhamento de porta no roteador da Multilaser. Neste é estipulado a porta de início e fim; o protocolo usado que pode ser TCP, UDP ou Ambos; e o IP do servidor de destino dos dados (Este é o IP que definimos para o micro controlador no item 3.2). Finalmente, desta forma, é criada uma conexão de ponta a ponta com os dispositivos envolvidos.

Na sequência é demonstrado como o servidor HTML no *Arduino* é configurado na linguagem de programação C.

3.2.3 SERVIDOR WEB

Como explicado acima, um dispositivo móvel, se comunica com o servidor Web HTML instanciado no micro controlador via o uso da porta de comunicação 8081. A função do servidor nesse projeto é disponibilizar uma página HTML através da qual o cliente se conecte e possa enviar solicitações que são retornadas ao micro controlador. O servidor Web também indica quais relés estão acionados e a temperatura da residência.

Na figura 24 segue uma breve cópia da linguagem de programação que cria o servidor na porta 8081.

```

void checkForClient(){

    EthernetServer server = EthernetServer(8081);
    EthernetClient client = server.available();

    if (client) {

        while (client.connected()) {
            if (client.available()) {
                // define a variável usada para troca de informações
                char c = client.read();

                if(c == '*'){

                    printHtmlCabecalho(client); // imprime o cabeçalho HTML da página
                    printLoginTitulo(client);
                    printHtmlCorpo(client);

                    break;
                }
            }
        }
    }
}

```

Figura 24. Linguagem de programação do servidor. Fonte: Autor.

O servidor Web HTML é implementado usando variáveis de conexão comum da IDE usada. Com o comando *EthernetServer server = EthernetServer(8081)* a IDE cria a conexão na porta 8081 e logo em seguida é gerado um elemento *client* em *EthernetClient client = server.available* cuja função é esperar a conexão de terceiros na porta do servidor.

Depois que o cliente se conecta e enquanto conectado, são realizadas duas tarefas que finalizam os procedimentos essenciais para estabelecimento de um meio que atinja os objetivos do projeto. A primeira é o estabelecimento de um caractere comum para troca de informações entre o cliente e o servidor em questão. Este caractere é definido com o comando *char c = client.read()*, onde o caractere *c* é alocado exclusivamente para entrada de dados do cliente que são lidos pelo método *read*. O servidor implementado com o *read* aguarda o cliente realizar a entrada de dados, enquanto o mesmo estiver online

Finalmente, a segunda e última tarefa é a impressão dos cabeçalhos, rodapés e corpo HTML que constituem a página Web à ser acessada. Tarefa realizada em *printHtmlCabecalho(client)* e *printHtmlCorpo(client)*. Assim, se cria a interface HTML necessária para a troca de dados de forma transparente para o usuário final.

Por último, é preciso definir quais são as entradas de dados possíveis pelo usuário usando o caractere *c* e quais resultados lógicos são gerados destas entradas. Os resultados neste projeto são a ativação dos diferentes relés que controlam os sistemas habitacionais. Uma cópia completa do código usado está disponível no anexo I deste projeto para melhor compreensão.

3.3 BLOCO DA INTERFACE DE ACIONAMENTO

Neste projeto para proporcionar o acionamento dos sistemas residenciais de iluminação, alarme e a tranca eletromagnética é usado um módulo relé com entrada para até oito atuadores.

3.3.1 CONTROLE DO MÓDULO RELÉ COM O ARDUINO

O processo de conexão física do módulo relé ao *Arduino* já foi explicado previamente neste documento, sendo preciso assim abordar o procedimento pelo qual o micro controlador aciona os diferentes relés do mesmo remotamente. O procedimento de acionamento remoto é controlado pelo código fonte ilustrado na figura 25.

```

if(reading){
  //se o caractere c = H
  if(c == 'H') {
    outp = 1;
  }
  //se o caractere c = L
  if(c == 'L') {
    outp = 0
  }

  Serial.print(c);
  switch (c) {
    case '0':
      acionaPin(outputAddress[0], client, outp);
      break;
    case '1':
      acionaPin(outputAddress[1], client, outp);
      break;
    case '2':
      acionaPin(outputAddress[2], client, outp);
      break;
    case '3':
      acionaPin(outputAddress[3], client, outp);
      break;
    case '4':
      acionaPin(outputAddress[4], client, outp);
      break;
    case '5':
      acionaPin(outputAddress[5], client, outp);
      break;
  }
}

```

Figura 25. Código que produz o acionamento. Fonte: Autor.

O código fonte da figura 25 é implementado resumidamente realizando a leitura do caractere enviado pelo cliente remoto através do cabeçalho HTML usando as portas de comunicação pré-definidas anteriormente. Sendo que quando o usuário enviar a letra “H”, o sistema gera como saída o dígito “1” e este por sua vez leva ao acionamento de uma das saídas digitais do micro controlador que assim ativa um dos relés. Seguindo a mesma lógica, ao receber “L”, a saída gerada é “0” e como consequência é desligado um dos atuadores. Esse processo no qual os dígitos são interpretados pelo micro controlador como acionamentos ocorre na função implementada *void acionaPin* presente na figura 26. Essa função é chamada pela função *case*, depois de ela identificar qual relé deve ser acionado.

```
void acionaPin(int pin, EthernetClient client, int outp){  
  
    if(outp == 1) {  
        digitalWrite(pin, HIGH);  
    }  
    if(outp == 0){  
        digitalWrite(pin, LOW);  
    }  
}
```

Figura 26. Função void acionaPin. Fonte: Autor.

Assim, para identificar qual relé se deseja ativar, o sistema usa a função *case*, e quando o usuário envia “H0” por exemplo, o sistema interpreta que deve acionar o primeiro dos oito relés disponíveis. É dada continuidade a esta lógica desde “H0” até “H7” o que torna possível o acionamento individual de cada um dos relés do módulo usado neste projeto.

No caso da automatização proposta é utilizado apenas a função *case* de “H0” até “H3”. Sendo o “H0” usado para o controle da iluminação, o “H1” da tranca eletromagnética, “H2” para o sistema de alarme, e “H3” para a sirene do alarme. As outras opções são declaradas por mera conveniência do usuário que pode usar estas entradas remanescentes do relé de acordo com sua vontade.

3.3.2 ACIONAMENTO DA ILUMINAÇÃO

Este projeto gera o acendimento das lâmpadas residências remotamente e assim, como consequência, a geração de maior conveniência e conforto ao usuário final. Isso é possível graças ao uso da lógica de acionamento do servidor Web em conjunto com um esquemático elétrico ligado ao módulo relé e ao *Arduino* que ao ser solicitado aciona as luzes. Esse esquemático está disposto na figura 27.

Ligação mista com interruptores e relé controlado pelo Arduino

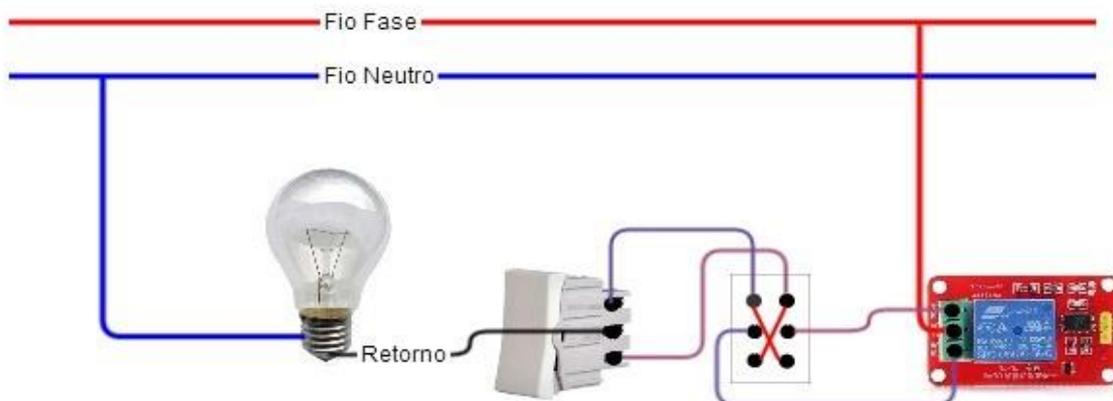


Figura 27. Automação da Iluminação. Fonte: adaptado de arduinobr.com.br

Não se deve esquecer que o usuário final ainda necessita de um interruptor convencional para usos esporádicos. Deste modo como pode ser visto no esquemático, entre o módulo relé e a lâmpada, usa-se um interruptor *three way* (um interruptor que prevê a entrada de duas fases e um retorno, tornando possível o acionamento elétrico em um segundo ponto.).

A entrada do módulo relé, por sua vez, é conectada a saída digital do *Arduino* fechando o conjunto necessário para realizar o acionamento das luzes que são ativadas quando usando a aplicação Web, o usuário enviar o comando “H0” convencional nesse material para acionamento exclusivo da iluminação.

3.3.3 ACIONAMENTO DA TRANCA ELÉTRICA

Seguindo a mesma lógica do processo de automação da iluminação, o acionamento da tranca eletromagnética remotamente é possível usando o servidor Web e o módulo relé. Entretanto, nesse caso o processo se difere pelo fato da tranca necessitar de uma fonte de

eletricidade de 12 volts. A fonte mencionada é conectada na alimentação elétrica da residência e, na sequência, é interligada no módulo relé que permite o acionamento remoto.

O esquemático da figura 28 ilustra como é possível atingir o objetivo de liberação da tranca remotamente e sumariza o parágrafo anterior.

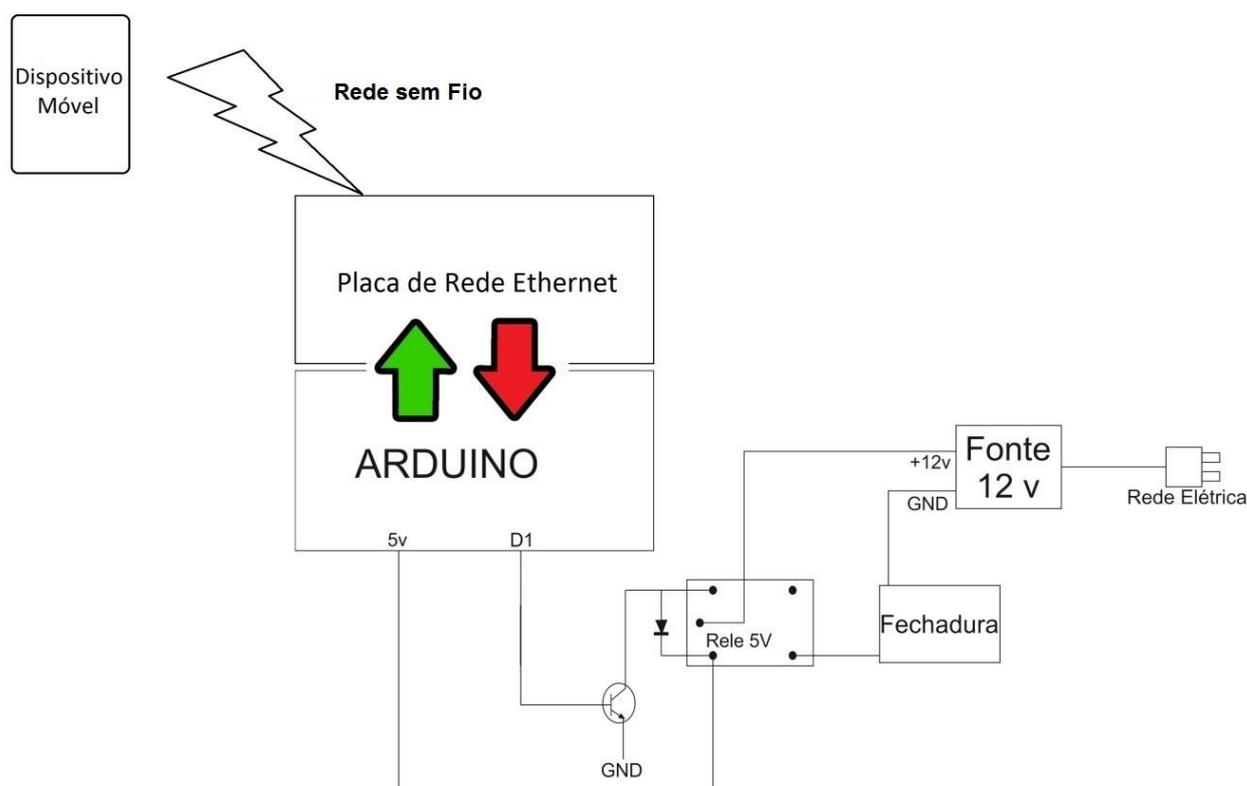


Figura 28. Esquemático Elétrico da Tranca. Fonte: Autor.

Para se abrir a tranca, o *Arduino* emite um sinal digital após receber o comando “H1” no cabeçalho HTML enviado pelo dispositivo móvel, acionando o relé da fechadura, permitindo a passagem de corrente proveniente da fonte. A corrente da fonte cria finalmente o campo eletromagnético no solenoide presente dentro da fechadura explicado no item 2.6, assim a mesma é aberta.

3.3.4 ACIONAMENTO DO ALARME

O acionamento do Alarme ocorre quando o usuário acessa o servidor Web e envia o cabeçalho “H2” pela página HTML, ativando o relé no qual está ligado a fase do sensor de movimento. Este, uma vez ativado, inicia o processo de identificação de variações infravermelhas que são interpretadas como a presença de um intruso no ambiente do mesmo.

Assim que um intruso é identificado no local em que o sensor é posicionado, este sensor de movimento PIR com seu relé interno, permite a passagem de corrente elétrica para o fio do retorno. O fio de retorno ilustrado na figura 29, alimenta a fonte da sirene piezoelétrica, gerando no fim deste processo o resultado esperado, um sinal sonoro característico dos corpos indesejados no ambiente.

Sempre que ativado, o sensor fica em funcionamento intermitentemente, até o momento no qual seja desativado pelo usuário. A proteção de acesso do mesmo nesse projeto é representada pela a senha de acesso criptografada em WPA AES da rede sem fio que possibilita o acesso servidor Web no *Arduino*, e conseqüentemente, ao controle de acionamento do sensor.

O WPA AES é um padrão de criptografia de 128 bits é uma forma de criptografia adotado pelo governo dos Estados Unidos desde 2001. É disponibilizado pela maior gama de fabricantes de roteadores, sendo muito usado na atualidade e apresentando um nível aceitável de segurança para este projeto, considerando a busca por uma solução de baixo custo.

O procedimento de configuração da chave de segurança é diferente de acordo com o roteador sem fio utilizado. Entretanto, esse processo é explicado no manual do usuário de cada equipamento desde que habilitado para fornecer essa solução de criptografia de acesso.

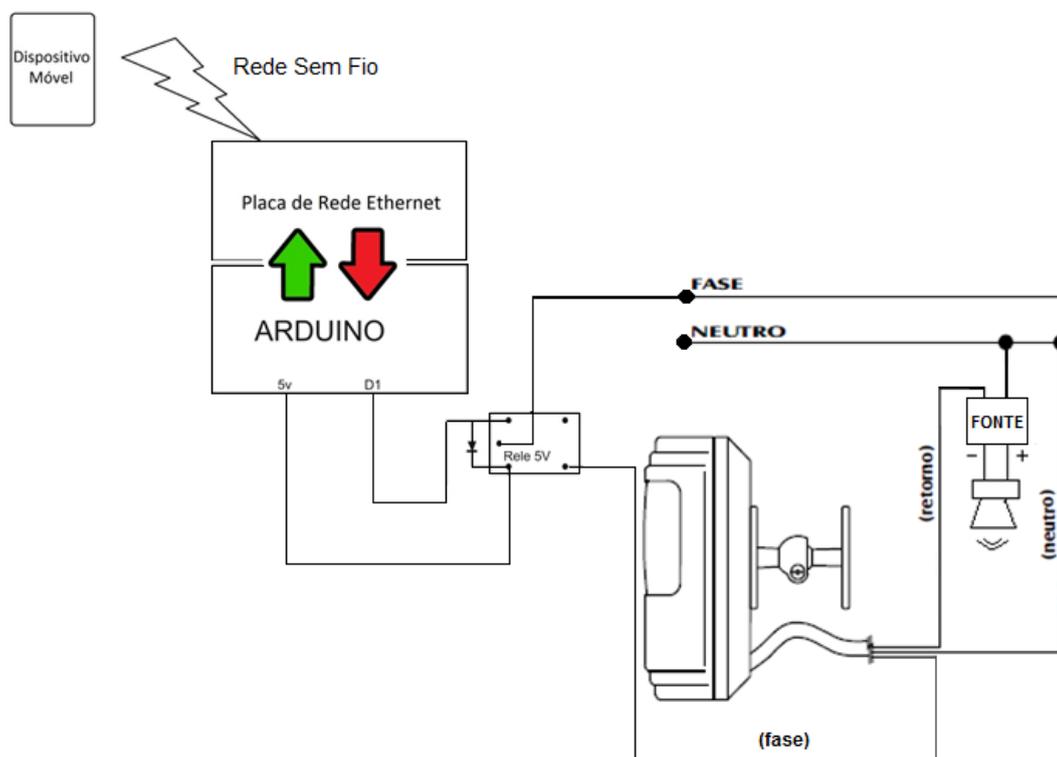


Figura 29. Esquemático do Alarme. Fonte: Autor, adaptação de www.fazedores.com

Para contextualização da configuração do roteador Multilaser usado nesse projeto verifique a figura 30.



Figura 30. Configuração da senha WPA. Fonte: Autor

3.4 BLOCO DA MONITORAÇÃO DE TEMPERATURA

A monitoração de temperatura é exercida pelo sensor DHT11 que lê a temperatura ambiente, depois envia esta informação para o *Arduino* que utilizando as bibliotecas do mesmo trata os dados binários recebidos para um valor em Celsius. O *Arduino* para finalizar o processo envia a cada 15 segundos a última leitura de temperatura para o dispositivo móvel. Isso é possível, pois o servidor Web realiza um *refresh* (atualização automática) da página HTML nesse intervalo (AOSONG. 2015).

A leitura da temperatura é um procedimento breve, pois o tratamento dos dados é transparente para o usuário final da IDE do micro controlador, uma vez referenciada a biblioteca do sensor, basta fazer a chamada das variáveis utilizadas para retornar os valores desejados.

Para referenciar o sensor, usa-se o comando *#include <DHT.h>* que informa para o *Arduino* com o *include* (comando de inclusão) da necessidade de uma biblioteca dentro da pasta “DHT”. A informação de funcionamento de todos acessórios, sensores e módulos do *Arduino* segue uma padronização na qual as bibliotecas sempre são salvas dentro da pasta *libraries* no caminho em que a IDE deste foi instalada em cada computador. Assim, uma vez baixada a biblioteca do DHT, basta copiá-la e atribuir o mesmo nome que é usado no comando *include*.

É apenas preciso identificar qual o modelo do sensor DHT e através de qual porta este realiza o envio dos dados. Na imagem da figura 31 pode se observar a linha de código “*DHT dht (A1, DHT11)*”. Ela tem o objetivo de informar que o sensor é conectado na porta analógica A1 e o modelo deste, DHT11.

```

#include <DHT.h>
DHT dht (A1,DHT11);

/////////////////////////////////////////////////////////////////
//LOOP
/////////////////////////////////////////////////////////////////

void loop() {

  //Lê a temperatura do sensor
  tempInValue = analogRead(tempInPin);
  tempOutDeg = dht.readTemperature();
}

```

Figura 31. Código do sensor de temperatura. Fonte: Autor

Resta, deste modo, fazer a chamada das variáveis que realizam a leitura dos dados. Estas variáveis são a *tempInValue* que com o *analogRead* (método de leitura de dados na entrada analógica) orienta o micro controlador a esperar a entrada de dados analógicos na entrada *tempInPin* (neste caso a analógica A1). A variável *tempOutDeg* por sua vez gera o valor final com o método de leitura *read* em *dht.readTemperature*.

Neste momento, o micro controlador já possui o valor da temperatura atualizada, porém ainda é necessário imprimir esse na página Web. Assim, basta fazer a chamada da variável *tempOutDeg* no corpo do código HTML da página Web, também presente no micro controlador na partição do Servidor Web referenciado anteriormente. O código abaixo (figura 32) é o responsável por realizar esse último processo com o comando *client.print(tempOutDeg)*.

```

//Imprimindo a temperatura na página HTML
client.print("<tr>\n");

client.print("<td><h4>");
client.print("Temperature");
client.print("</h4></td>\n");
client.print("<td></td>");
client.print("<td>");
client.print("<h3>");
client.print(tempOutDeg);

```

Figura 32. Código HTML que imprime a temperatura no navegador. Fonte: Autor.

3.5 RESULTADOS FINAIS DO PROJETO

Os resultados finais do projeto final consistem em duas vertentes, a primeira é a implementação final do código de programação que gera para o usuário uma interface fácil e intuitiva de uso e a segunda é uma maquete na qual estão presentes todos os sistemas residenciais que se deseja controlar.

A figura 33 é o resultado da primeira vertente, ou seja, a concatenação da linguagem de programação C e linguagem de programação HTML que permitem o funcionamento lógico do *Arduino* e tornam o mesmo acessível remotamente de modo simples.

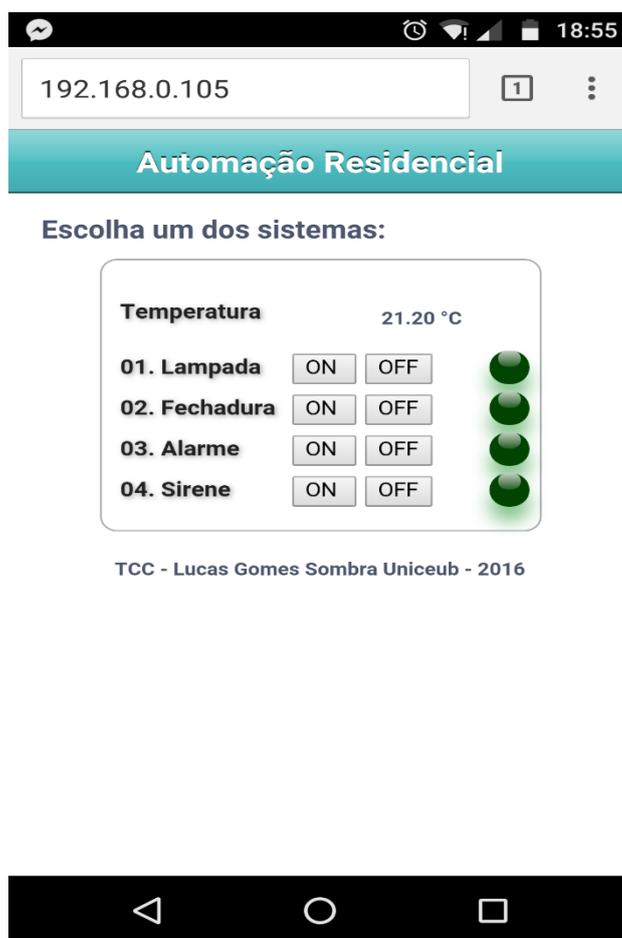


Figura 33. Interface Web. Fonte: Autor

A figura 34, presente na sequência é o primeiro lado da maquete montada para este projeto. Deste lado foram instalados uma luz que é automatizada e pode ser acionada pela interface Web ou pelo interruptor, uma sirene, e o sensor de movimento.



Figura 34. Frente da maquete. Fonte: Autor.

O sensor de movimento possui duas saídas, a primeira saída do sensor é o aviso sonoro provocado pela sirene. A sirene é ativada exclusivamente pela interface web. A razão disto é que como mencionado no tópico referente ao desenvolvimento do alarme para se acessar a interface Web é necessário acessar primeiramente, a rede sem fio disponibilizada pelo roteador

que, por sua vez, é criptografada. Assim, isso garante que uma vez acionada, apenas usuários confiáveis poderão desativar a sirene.

Entretanto, a mesma sirene provoca sons muito agudos que são indesejados para a realização dos testes deste projeto. A contar deste fato, é instalado uma lâmpada de teste como uma segunda saída do sensor de movimento, e esta é acionada pelo segundo interruptor presente na figura 34. Esta lâmpada aumenta as possibilidades do alarme, uma vez que com ela, o alarme pode ser sonoro, visual ou ambos.



Figura 35. Traseira da maquete. Fonte: Autor.

Na parte de trás da maquete (figura 35) é encontrado a caixa VDI, e dentro da mesma, o conjunto que atinge o propósito de automatizar todos os sistemas residenciais. Este conjunto é composto do roteador *wireless*, *Arduino* com *Ethernet Shield*, e módulo relé.

CAPÍTULO 4: TESTES E RESULTADOS

Este capítulo aborda os testes realizados no protótipo com intuito de validar se os objetivos específicos descritos no primeiro capítulo deste material são alcançados.

Para melhor dimensionar os testes, estes são divididos nos seguintes cenários:

- I. Primeiro Cenário: Testes da interface do Bloco de Comunicação e testes do Bloco de Comunicação com Bloco de Acionamento.
- II. Segundo Cenário: Testes do Bloco de Monitoramento de Temperatura.
- III. Terceiro Cenário: Testes do sub-bloco do sistema de alarme.

4.1 PRIMEIRO CENÁRIO

Para que essa automação seja atingida é preciso haver duas interfaces, a de comunicação e acionamento funcionando em conjunto para garantir o acesso remoto. Ou seja, trabalhando separadamente enquanto, passando para o usuário final transparência e integridade.

O acesso remoto é possível nesse projeto graças ao uso do micro controlador *Arduino Mega* que junto ao *Ethernet Shield* disponibiliza o servidor Web. Deste modo, para testar a comunicação entres esses componentes, eles são conectados como explicado no desenvolvimento e o código fonte que rege o funcionamento do *Arduino* é implementado. A lógica usada é a do diagrama mostrado na figura 36.

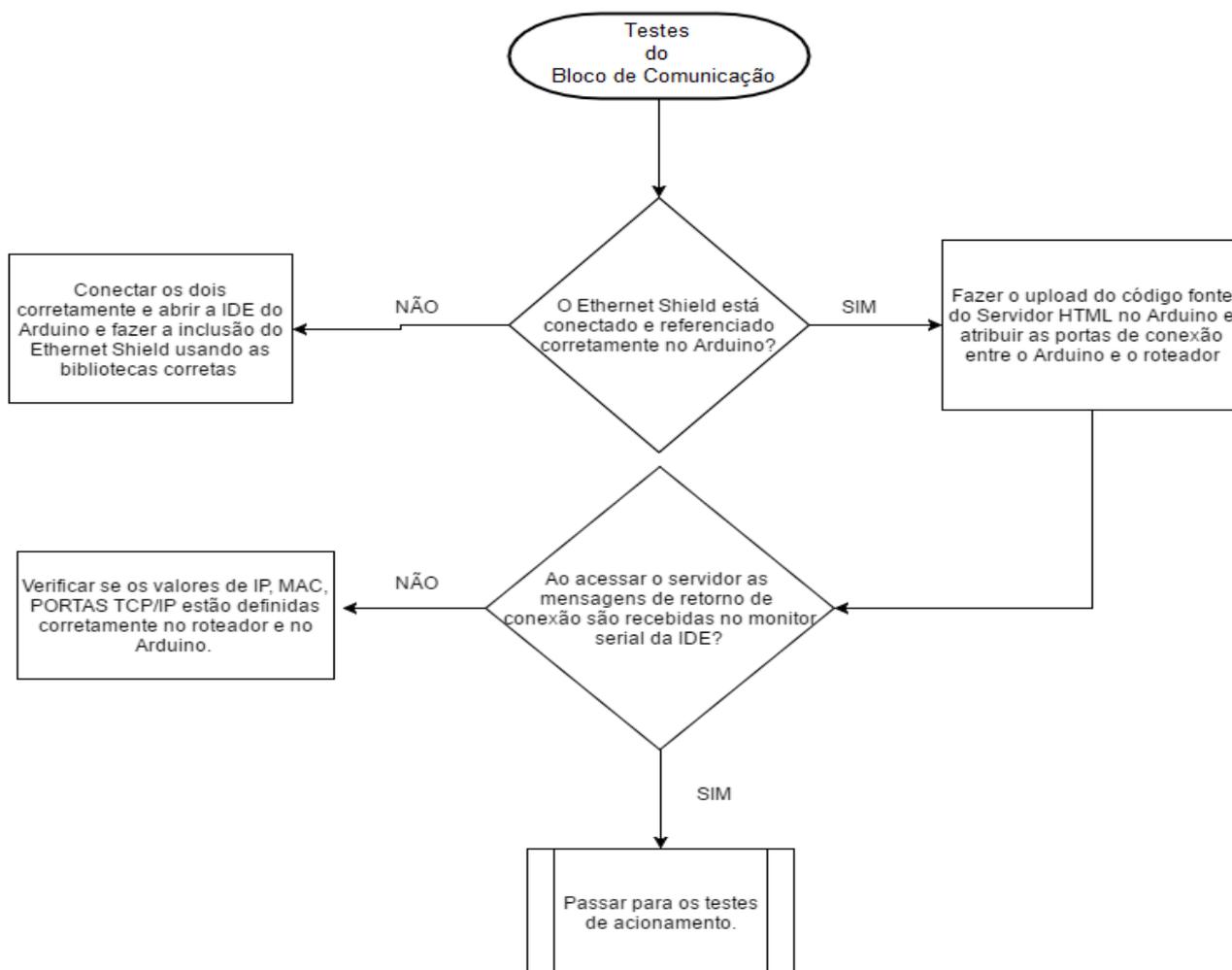


Figura 36. Diagrama dos testes de comunicação com o servidor. Fonte: Autor.

Inicialmente, a comunicação é testada acessando a página HTML em um dispositivo remoto qualquer, enquanto é verificando o retorno no monitor serial da IDE do *Arduino* de mensagens validadoras. O Monitor Serial da IDE do *Arduino* é uma janela do software, e esta é disponibilizada pelo fabricante exatamente com o propósito de facilitar a realização de testes.

O monitor serial opera mostrando todas as mensagens escolhidas pelo programador que são chamadas no código fonte dentro do *Arduino* pelo comando “*Serial.print(variável ou texto)*”. Na figura 37 é mostrado o código que gera a primeira mensagem validadora.

```

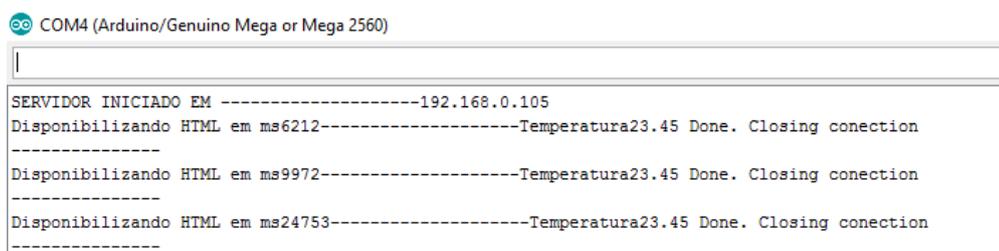
void printHtmlCorpo(EthernetClient client){
  Serial.print("Disponibilizando HTML em ms");
  timeConnectedAt = millis();
  Serial.print(timeConnectedAt);
  writeToEeprom = true;
}
  
```

Figura 37. Código que gera a mensagem validadora da conexão. Fonte: Autor.

Neste caso, a mensagem validadora escolhida é “Disponibilizando HTML em x ms”. Esta mensagem de retorno é adicionada ao *sketch* (código-fonte) dentro do micro controlador na função *printHtmlCorpo*, vista na imagem da figura 37. O objetivo dessa função é imprimir o *layout* da página HTML acessada.

Assim, apenas após o cliente se conectar e fazer o download da página HTML no navegador, a mensagem validadora é gerada garantindo que a conexão ponta a ponta foi bem-sucedida. Esta ainda inclui o tempo de execução no qual o cliente se conectou, o que é possível pois na IDE existe a função *millis* que conta o tempo de execução dos comandos desde o início.

O resultado gerado na janela do monitor serial é apresentado na figura 38:



```

COM4 (Arduino/Genuino Mega or Mega 2560)
SERVIDOR INICIADO EM -----192.168.0.105
Disponibilizando HTML em ms6212-----Temperatura23.45 Done. Closing connection
-----
Disponibilizando HTML em ms9972-----Temperatura23.45 Done. Closing connection
-----
Disponibilizando HTML em ms24753-----Temperatura23.45 Done. Closing connection
-----

```

Figura 38. Mensagem validadora de conexão no monitor serial. Fonte: Autor.

Ainda é visível na figura 38 que é exibida uma segunda mensagem validadora, “Done. Closing Connection”. A importância desta é de garantir que logo após o cliente realizar o download da página, o servidor feche a conexão TCP-UDP com o comando *client.stop()* presente na imagem da figura 39. Ele é vital, pois evita congestionamentos no meio de comunicação e a execução de processos desnecessários, deixando as portas de conexão sempre livres para novas solicitações e evitando o gasto desnecessário de memória.

```

delay(1);

client.stop(); // FECHA A CONEXÃO

Serial.println(" Done. Closing connection");
Serial.println("-----");

```

Figura 39. Código que gera a mensagem validadora do fechamento da conexão. Fonte: Autor.

Perceba que o código `Serial.println` (“Done. Closing Conection”) desta mensagem validadora é colocado apenas após o comando `cliente.stop()` que fecha a conexão (figura 39). Assim, esta mensagem validadora garante a integridade do meio de comunicação.

Visto que a conectividade entre o cliente e o servidor no *Arduino* funcionam corretamente, é testado o Bloco da Interface de Acionamento. Estes testes são realizados enviando os caracteres responsáveis por produzir a saída sinal digital no *Arduino* que produz a ativação dos relés desejados. No diagrama da figura 40 é sumarizado a lógica utilizada.

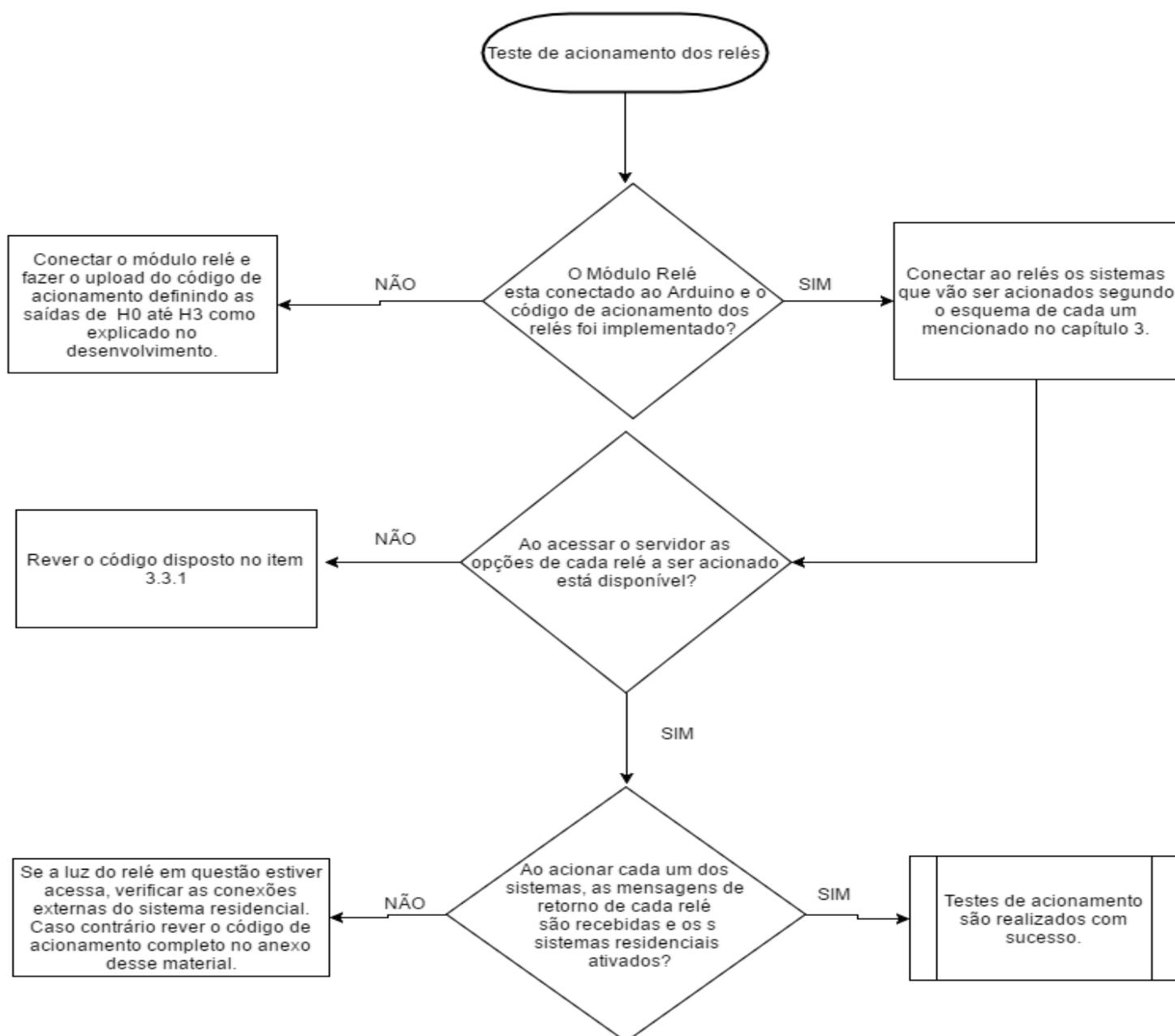


Figura 40. Diagrama dos testes de acionamento dos relés. Fonte: Autor.

A imagem da figura 41 é o resultado dos testes de acionamento realizados no monitor serial. Como pode ser visto, toda vez que o cliente aciona um rele, são enviados os caracteres que causam esse acionamento para validação na janela do monitor serial.

Sendo que, primeiramente sempre, é enviado o identificador “?” que orienta o servidor a ficar no aguardo para um comando, seguindo de “H” no caso de acionamento ou “L” para desativação, e o número do relé desejado. Depois, que todos os caracteres são enviados, a conexão é sempre fechada novamente.

```

Disponibilizando HTML em ms68167-----?-----H-----1----- Done. Closing conection
-----
Disponibilizando HTML em ms70856-----Temperatura23.45 Done. Closing conection
-----
Disponibilizando HTML em ms72855-----?-----H-----2----- Done. Closing conection
-----
Disponibilizando HTML em ms73109-----Temperatura23.45 Done. Closing conection
-----
Disponibilizando HTML em ms74055-----?-----L-----2----- Done. Closing conection
-----
Disponibilizando HTML em ms74360-----Temperatura23.45 Done. Closing conection
-----
Disponibilizando HTML em ms75457-----?-----H-----3----- Done. Closing conection
-----
Disponibilizando HTML em ms75749-----Temperatura23.45 Done. Closing conection
-----
Disponibilizando HTML em ms76856-----?-----L-----3----- Done. Closing conection
-----
Disponibilizando HTML em ms77146-----Temperatura23.45 Done. Closing conection
-----

```

Figura 41. Mensagens validadoras dos testes de acionamento. Fonte: Autor.

Um problema encontrado durante os acionamentos foi que ao se sair da aplicação Web e retornar depois de certo tempo, ela não apresentava novamente o status dos acionamentos que tinham sido realizados. O problema foi solucionado com o uso do contador de milissegundos já demonstrado mencionado e o uso da memória EEPROM do *Arduino*.

Para resolução, bastou definir um limite de tempo que depois de ser atingido, gerasse a gravação do último status dos sistemas da casa na EEPROM. (Memória não volátil). Assim,

conforme o código da figura 42 quando o tempo de conexão estoura 60 segundos a função *void gravaEeprom* é ativada e salva os últimos valores dos *outputStatus* que representam o estado dos sistemas da casa.

```

if (millis() > (timeConnectedAt + 60000){
    if (gravadoNaEeprom == true){
        gravaEeprom();
        Serial.println("Sem clientes por mais de um minuto - Atualizando EEPROM.");
        gravadoNaEeprom = false;
    }
}

void gravaEeprom(){
    for (int adr = 0; adr < outputQuantity; adr++) {
        EEPROM.write(adr, outputStatus[adr]);
    }
}

```

Figura 42. Código que grava os status na EEPROM. Fonte: Autor.

A tabela 1 resume os testes de acionamento, os testes consistiram em acionar cada relé individualmente 10 vezes, enquanto buscavam falhas de acionamento. Felizmente, estes testes não apresentaram erros durante sua execução

Sistema residencial testado	Quantidade de sucessos	Quantidade de falhas	Taxa de Sucesso
Iluminação	10	0	100%
Tranca eletromagnética	10	0	100%
Alarme (Sirene e Sensor PIR)	10	0	100%

Tabela 1. Resultado dos Testes de acionamento. Fonte: Autor.

4.2 SEGUNDO CENÁRIO

Neste cenário o Bloco da monitoração de temperatura é testado. Ele é constituído de um sensor de temperatura DHT11 que é conectado ao micro controlador e a disponibilização dos valores lidos pelo sensor na página HTML de forma clara para o usuário.

Para se testar essa funcionalidade o método usado baseou na lógica do diagrama da figura 43. Resumindo, se conecta o sensor ao *Arduino* e ao acessar a página HTML busca-se verificar as medições de temperatura tanto na própria página, quanto no monitor serial da IDE.

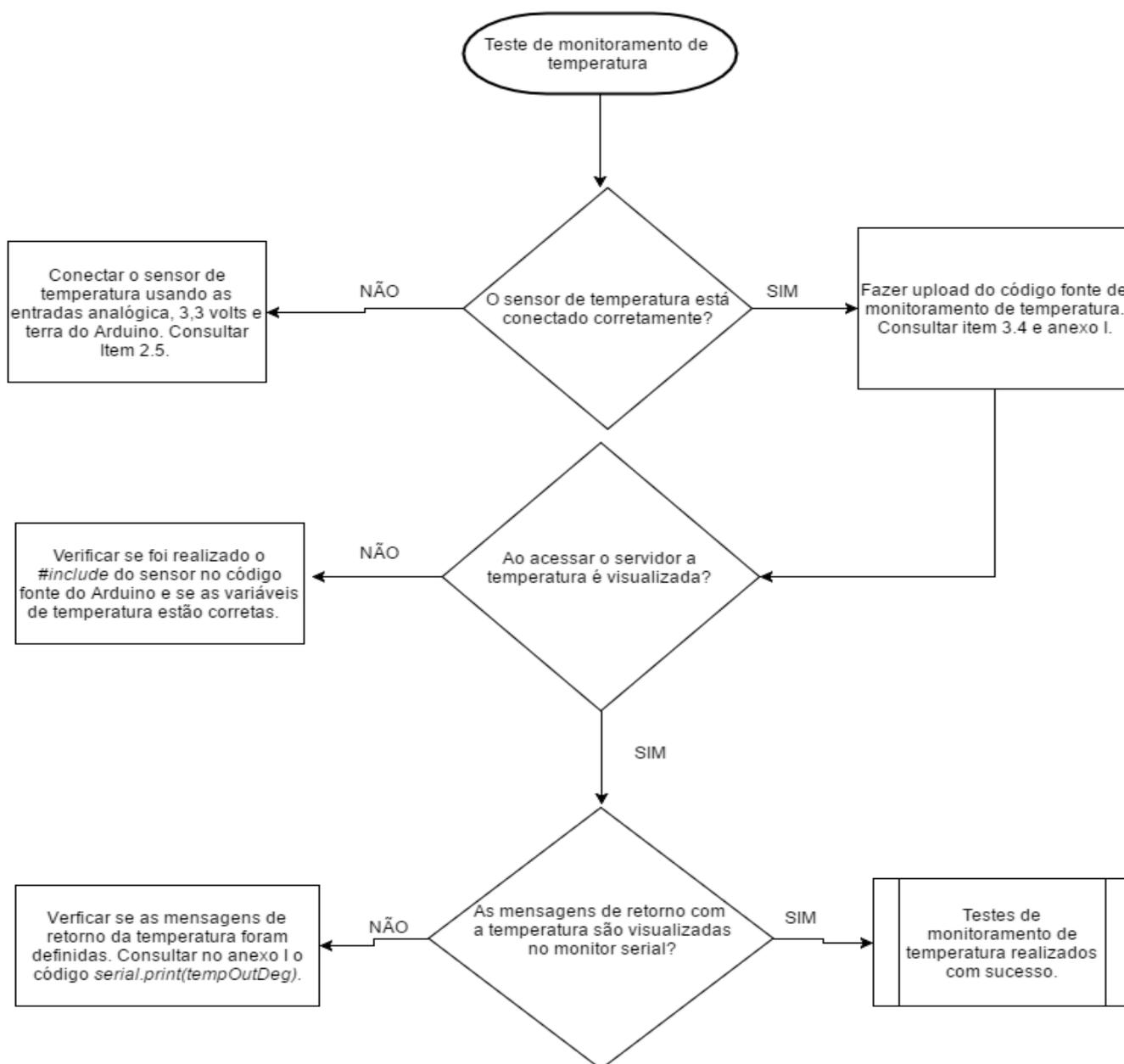


Figura 43. Diagrama dos testes do sensor de temperatura. Fonte: Autor.

COM4 (Arduino/Genuino Mega or Mega 2560)

```

Disponibilizando HTML em ms689602-----Temperatura26.54 Done. Closing conection
-----
Disponibilizando HTML em ms695306-----Temperatura26.54 Done. Closing conection
-----
Disponibilizando HTML em ms696098-----Temperatura26.54 Done. Closing conection
-----
Disponibilizando HTML em ms701770-----Temperatura26.54 Done. Closing conection
-----
Disponibilizando HTML em ms702052-----Temperatura26.54 Done. Closing conection
-----
Disponibilizando HTML em ms703115-----Temperatura26.54 Done. Closing conection
-----
Disponibilizando HTML em ms703398-----Temperatura26.54 Done. Closing conection
-----
Disponibilizando HTML em ms718399-----Temperatura25.54 Done. Closing conection
-----
Disponibilizando HTML em ms718680-----Temperatura25.54 Done. Closing conection
-----
Disponibilizando HTML em ms733669-----Temperatura25.54 Done. Closing conection
-----

```

Figura 44. Mensagens validadoras dos testes de temperatura no monitor serial. Fonte: Autor.

Na imagem da figura 44 é verificado o retorno das medições de temperatura gerados no monitor serial pelo código `serial.print(tempOutDeg)` visto na imagem da figura 45. São gerados diferentes valores de temperatura, pois o servidor atualiza a página HML a cada 15 segundos enquanto o cliente estiver com ela aberta no navegador aberto.

```

//IMPRIME A TEMPERATURA NA PÁGINA
client.print("<tr>\n");

client.print("<td><h4>");
client.print("Temperatura");
Serial.print("Temperatura");
client.print("</h4></td>\n");
client.print("<td></td>");
client.print("<td>");
client.print("<h3>");
client.print(tempOutDeg);
Serial.print(tempOutDeg); //IMPRIME MENSAGEM DE RETORNO COM A TEMPERATURA NO MONITOR SERIAL.

```

Figura 45. Código que gera as mensagens validadores de temperatura. Fonte: Autor

De modo análogo ao acionamento dos relés, sempre depois que a temperatura é enviada pela conexão por `sockets`, a conexão é fechada com a mensagem “Done. Closing Conection”. Ao mesmo tempo, se o cliente sair do navegador ou ficar mais de um minuto sem acessar o mesmo, ele é desconectado até que ocorram novos acessos. Essa segunda política de acesso

também aumenta a qualidade das conexões, evitando gastos desnecessários de processos de envio de dados.

Em todos os testes executados, a temperatura medida pelo sensor do projeto foi contrastada com um termômetro externo TFA Germany (figura 46) e para todos os valores a diferença percentual apresentada foi de menor que 1% como pode ser visto na tabela 2 no fim desta página. Conclui-se que como o sensor não apresentou erros de conectividade ou medição, este atingiu os propósitos funcionais deste projeto.



Figura 46. Termômetro externo usado. Fonte: Autor.

Testes do Sensor de Temperatura

Temperatura da automação	Temperatura do termômetro externo	Diferença	Diferença percentual
26.54 °C	26.4 °C	0.14 °C	0.53%
26.54 °C	26.4 °C	0.14 °C	0.53%
26.54 °C	26.4 °C	0.14 °C	0.53%
26.54 °C	26.4 °C	0.14 °C	0.53%
26.54 °C	26.4 °C	0.14 °C	0.53%
26.54 °C	26.4 °C	0.14 °C	0.53%
26.54 °C	26.4 °C	0.14 °C	0.53%
25.54 °C	25.6 °C	0.06 °C	0.23%
25.54 °C	25.6 °C	0.06 °C	0.23%
25.54 °C	25.6 °C	0.06 °C	0.23%

Tabela 2. Resultados dos testes do sensor de temperatura. Fonte: Autor.

4.3 TERCEIRO CENÁRIO

É abordado no referencial teórico que a automação de segurança desse projeto passa pela execução de um teste de confiabilidade no qual é amostrado o funcionamento do sensor de movimento. A razão para execução deste teste é tentar verificar a existência de erros por parte do sensor de movimento durante o seu uso. Sendo definido como erro toda vez em que um corpo que emite radiações infravermelhas invadindo o perímetro do sensor e este não o detecta.

Como já explicado anteriormente é inviável testar o sensor ao longo de semanas para visualizar o comportamento do mesmo. Sendo assim, faz-se necessário o uso de uma fórmula de amostragem que providencie uma quantidade de horas no qual o mesmo é passível de ser testado. A fórmula de amostragem usada é a seguinte:

$$n = \frac{z^2 x \sigma^2 x N}{d^2(N-1) + Z^2 \sigma^2} \quad (1)$$

Considerando a natureza funcional desse projeto, os testes realizados no mesmo seguem a mesma metodologia de testes. Teste funcionais ou testes de caixa preta são aqueles nos quais são verificadas as funções do sistema sem se preocupar com os detalhes de implementação. Deste modo, o teste funcional se baseia em duas etapas, na primeira são identificadas as funções que o sistema deve executar (nesse caso o aviso sonoro para intrusos) e na segunda criar entradas (movimentações no perímetro do sensor) capazes de testar se essas funções estão sendo realizadas (CARLOS, JOSÉ; FRANCINE, ELLEN; AURI, MARCELO. 2004).

Dado isso, para os testes realizados nesta etapa considera-se o período amostral de 360 horas (Metade de um mês). Estipula-se este período, pois entende-se que os usuários do sensor de presença usam o mesmo quando fora de suas residências ou a noite durante o repouso. Assim, foi determinado para esse teste que o período no qual o alarme fica ativo é de 12 horas por dia.

Após a inclusão desses dados na fórmula apontada na página anterior, temos:

$$n = \frac{1,96^2 \times 3,6^2 \times 360}{2^2(360-1) + 1,96^2 \times 3,6^2} = 23.34 \text{ horas} = 24 \text{ horas}$$

Para estes testes foi usado o valor de Z como 1.96 o que representa um nível de confiança de 95% e um erro amostral σ de 1% das horas testadas, ou seja, 3.6 horas. O resultado obtido é de 23.34 horas de testes a serem realizadas e este é arredondado para 24 horas por conveniência.

Baseando-se na metodologia de testes funcionais, o sensor é testado ativando a automação de segurança ao longo de dois períodos de 12 horas e uma vez por hora o sensor sofre uma invasão na busca de erros de identificação do invasor. O perímetro da casa de teste usado é de 2.8 m por 4.8m, ou seja, uma área total de 13,44 metros quadrados.

Na figura 47 está disponível um diagrama que visa resumir o processo utilizado nesse terceiro cenário.

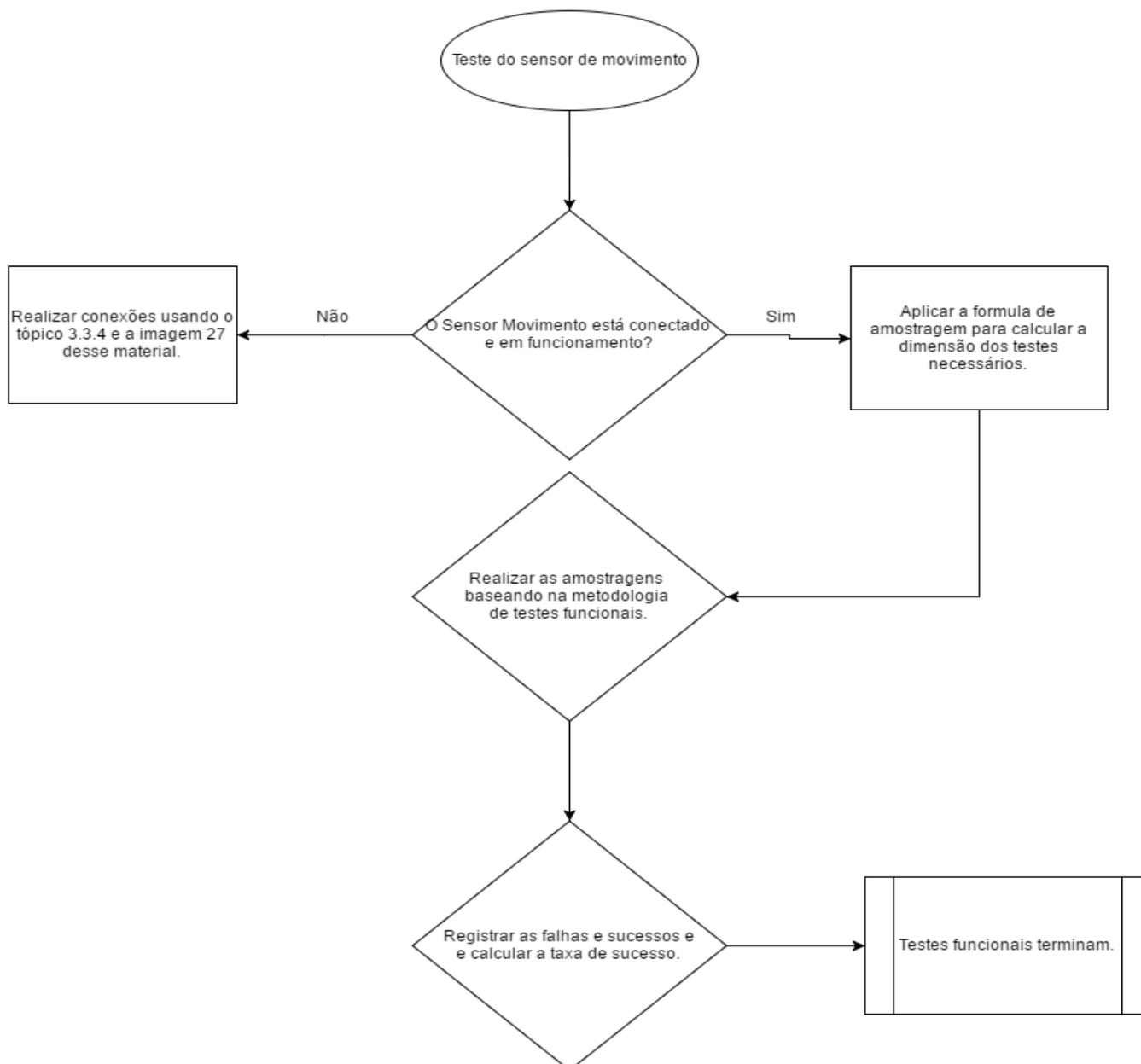


Figura 47. Diagrama dos testes funcionais do sensor de movimento. Fonte: Autor.

O resultado dos 24 testes do sensor de movimento é sumarizado na tabela 3, sendo que, foi constatada apenas uma falha do mesmo. Consideramos falha para estes testes quando ao ser realizada uma das amostragens, o sensor de movimento não apresenta como resultado a ativação da sirene do alarme. Como resultado dos testes se tem uma taxa de sucesso de 95,83% e esta representa a probabilidade de funcionamento correto para o período de um mês amostrado.

Para os fins deste material a taxa de sucesso apresentada foi aceitável, uma vez que, para todos os sistemas criados pelo ser humano que é imperfeito, sempre haverá melhorias que podem ser implementadas.

O motivo da falha que ocorreu é o suprimento de energia de Brasília, pois a casa teste sofreu uma falta de energia que causou o desligamento da automação residencial. As melhorias passíveis de serem realizadas que possam evitar o erro identificado ou ainda trazer melhorias para o sistema, serão tratadas no próximo capítulo juntamente ao tópico de trabalhos futuros.

Teste do Sensor de Movimento	
Número do Teste	Sucesso/Falha
1	Sucesso
2	Sucesso
3	Sucesso
4	Sucesso
5	Sucesso
6	Sucesso
7	Sucesso
8	Sucesso
9	Sucesso
10	Sucesso
11	Sucesso
12	Sucesso
13	Sucesso
14	Sucesso
15	Sucesso
16	Sucesso
17	Falha
18	Sucesso
19	Sucesso
20	Sucesso
21	Sucesso
22	Sucesso
23	Sucesso
24	Sucesso
Taxa de Sucesso	95,83%

Tabela 3. Resultados dos testes do sensor de movimento. Fonte: Autor

CAPÍTULO 5: CONCLUSÃO

Neste capítulo há a apresentação das conclusões feitas ao final do processo de desenvolvimento da automação e realização dos testes. Finalizando o trabalho há a sugestão de projetos futuros.

5.1 CONCLUSÕES

A solução de automação é projetada visando de possibilitar o controle e monitoramento de diferentes sistemas residenciais com o uso de micro controladores e redes sem fio gerando, como consequência, um maior conforto e conveniência.

Este objetivo geral é atingido, uma vez que, a automação implementada controlou com sucesso de forma transparente todos os sistemas residências desejados, ou seja, a iluminação, o acesso remoto via tranca eletromagnética, alarme representado por um sensor de temperatura e uma sirene, e o monitoramento de temperatura.

A esquemático implementado que integra os diferentes sistemas automatizados ao modulo relé e ao micro controlador possibilitou o acionamento remoto com sucesso de todos os sistemas residenciais desejados. Sendo que, na automação projetada são usados apenas quatro dos oito relés do módulo escolhido. Assim, como a programação explicada no desenvolvimento permite o uso das oito saídas, a automação pode ser expandida futuramente.

Segundo os testes realizados, todos, a página HTML, o servidor HTML e a programação em C que orienta o micro controlador e o esquemático elétrico-lógico que integra os sistemas

residenciais funcionam em sincronismo. Deste modo, é possível estabelecer o objetivo específico de criação desta interface de comunicação e acionamento é atingido com sucesso.

Os testes de comunicação e acionamento realizados mostraram a ocorrência de um problema relacionado a disponibilização dos *status* dos sistemas depois de períodos de tempo desconectado da página HTML, e conseqüentemente do servidor. Este problema foi sanado com o uso da memória EEPROM, permitindo assim aos usuários uma monitoração intervalada dos sistemas que são usados.

No sub-bloco do sistema de alarme o único erro encontrado foi a falha na detecção do sensor de movimento em um momento de falta de suprimento de energia. Mesmo após considerar essa falha, o sensor de movimento apresentou uma taxa de sucesso de 95,83%. No próximo tópico referente a trabalhos futuros é sugerido como sanar o mesmo.

5.2 TRABALHOS FUTUROS

Nesta seção encontram-se algumas opções de melhoria do sistema proposto que retratam ideias não implementadas:

1. Primeiramente para projetos futuros é interessante mencionar a melhoria da estética do conjunto da automação agregado dentro da caixa de automação representada nesse primeiro trabalho por um quadro plástico de VDI.
2. Adição de uma fonte de energia interna que seja acionada quando houver falta de suprimento de energia fornecida pela concessionária, assim garantindo o contínuo funcionamento da solução de alarme, conseqüentemente sanando a falha apresentada para este cenário.

3. Adição de novas funcionalidades como o controle de dispositivos com o uso de emissores e receptores IR, assim estendendo consideravelmente as funcionalidades da automação por uma quantia financeira irrisória. E, ao mesmo tempo, permitindo o uso de controles remotos como uma alternativa.
4. Criação de uma interface com banco de dados que permita a internalização de programações com preferencias de uso gerando deste modo mais conveniência para o usuário final.
5. Estudo e implementação de outros sistemas residenciais e sensores que podem ser controlados pelo uso de micro controladores como o *Arduino Mega* utilizado no projeto em questão. Como exemplos, podem ser citados sistemas residenciais que usam motores para controle de bombas d'água e ainda sensores de gases nocivos para identificação de situações de risco.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ADA, LADY. **How PIR Sensors Work**. Adafruit. EUA, 2015. Disponível em <<https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work>>.

Acessado em 28 de março de 2016.

AOSONG, **Temperature and humidity module DHT11 Product Manual**. China, 2015. p. 2-4. Disponível em <<https://akizukidenshi.com/download/ds/aosong/DHT11.pdf>>.

Acessado em 06 de Abril de 2016.

ARDUINO.CC. **Arduino Software (IDE)**. EUA, 2016. Disponível em <<https://www.arduino.cc/en/Guide/Environment>>. Acessado em 28 de março de 2016

ARDUINO.CC **Arduino Mega 2560**. EUA, 2016. Disponível em <<https://www.arduino.cc/en/Main/ArduinoBoardMega2560#>>. Acessado em 25 de março de 2016.

ASSOCIAÇÃO ESPANHOLA DE DOMÓTICA. **Automação residencial: histórico, definições e conceitos**. 2015. Disponível em <<http://docplayer.com.br/2355303-Automacao-residencial-historico-definicoes-e-conceitos.html>>. Acessado em 8 de março de 2016.

AURESIDE, **Automação Residencial: Demanda na construção civil**. São Paulo, 2013. Disponível em <<http://www.aureside.org.br/noticias/automacao-residencial--demanda-na-construcao-civil>>. Acessado em 08 de março de 2016.

BRAGA, Newton C; **Tudo Sobre Relés**, Instituto Newton C. Braga, Abril de 2012. São Paulo.

CARLOS, Jose; FRANCINE, Ellen; AURI, Marcelo. **Introdução a testes de software**. Universidade de São Paulo. 2004. Disponível em http://www.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_ND_65.pdf. Acessado em 27 de abril de 2016.

CHAGAS, Christian R. **Transferência de dados via rede elétrica baseado no protocolo X10**. Uniceub. Brasília, 2010. Disponível em: <<http://www.repositorio.uniceub.br/bitstream/123456789/3368/3/20064256.pdf>>. Acessado em 3 de março de 2016;

DICIO. **Definição: Automação**. Brasil, 2016. Disponível em <<http://www.dicio.com.br/automacao>>. Acessado em 28 de fevereiro de 2016.

FOGAÇA, Jennifer Rocha. **Raios infravermelhos**; Brasil Escola. Brasil, 2015. Disponível em <<http://brasilecola.uol.com.br/quimica/raios-infravermelhos.htm>>. Acesso em 26 de março de 2016.

FONSECA, Jairo S. e MARTINS, Andrade G., **Curso de Estatística**, Editora Atlas, 6ª edição. São Paulo, 1996.

GIMENEZ, Salvador Pinillos, **Micrfo controladores 8051 – Teoria e Prática**, Editora Abril, 1ª edição. São Paulo, 2010.

INTERSABERES, **Redes**. Editora Intersaberes, 1ª edição. Curitiba, 2014.

KUROUSE, James; KEITH, Ross; **Redes de Computadores e a Internet: Uma abordagem top-down**, Pearson. 3ª edição. São Paulo, 2005.

MURATORI, Jose Roberto; **Os desafios do mercado de Automação Residencial**, Portal da Arquitetura e Construção. Brasil, 2015. Disponível em: <http://www.aecweb.com.br/cont/a/os-desafios-do-mercado-da-automacao-residencial_8192>. Acessado em 1 de março de 2016.

OLIFER, Natalia. **Redes de Computadores: princípios, tecnologías e protocolos para o projeto de redes**. LTC, 1ª Edição. Rio de Janeiro, 2008.

PATSKO, Luís F. , **Tutorial Controle de Relés**, Maxell Bohr, PdP – Pesquisa e Desenvolvimento de Produtos, 18 de Dezembro de 2006. p.1-2. Brasil, 2006.

TANENBAUM, Andrew S. ; Wetherrall, DAVID . **Redes de Computadores**, Pearson. 5ª edição. São Paulo, 2011.

TECHINBRAZIL. **Vale a pena investir em automação para a casa? .** Brasil, 2014. Disponível em: <<http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/06/vale-pena-investir-em-automacao-para-casa.html>>. Acessado em 1 de março de 2016.

TECHINBRAZIL. **Integradores da Automação Residencial.** Brasil, 2015. Disponível em: <<https://techinbrazil.com.br/integradores-de-automacao-residencial-no-brasil>>. Acessado em 1 de março de 2016.

WIZNET, **W5100 Ethernet Shield Datasheet.** EUA, 2008. Disponível em <https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf> . Acessado em 06 de Abril de 2016.

ANEXO A

Na sequência é disponibilizado o código em linguagens de programação C e HTML que gerenciam o funcionamento do micro controlador utilizado neste projeto de automação para controle de sistemas residenciais automatizados através do uso de dispositivos remotos e redes sem fio.

```
//Centro Universitário de Brasília
// Engenharia de Computação - FATEC
// Automação Residencial para controle e monitoramento da
Iluminação, Segurança e Controle de Acesso utilizando o Arduino Mega
// Lucas Gomes Sombra - RA21114229
```

Este código foi adaptado do original criado por Claudio Vella Malta, 2012.

```
#include <Ethernet.h>
#include <SPI.h>
#include <EEPROM.h>
#include <DHT.h>
#define DHTPIN A1L
#define DHTTYPE DHT11

// REFERÊNCIA DO SENSOR DE TEMPERATURA

DHT dht(DHTPIN, DHTTYPE);
DHT dht (A1,DHT11);

//VARIÁVEIS DE CONEXÃO

byte ip[] = {
  192, 168, 0, 105 }; //IP DO ARDUINO
byte gateway[] = {
  192, 168, 0, 1 }; //GATEWAY DO ROTEADOR
byte subnet[] = {
  255, 255, 255, 0 }; //MÁSCARA SUBNET

// ENDEREÇO MAC DO ARDUINO
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

```

//NUMERO DE RELÉS USADOS
int outputQuantity = 4;
boolean outputInverted = false;

//TEMPO DE ATUALIZAÇÃO DA PÁGINA WEB
int refreshPage = 15;
int ligarTodosBotoes = false;

//NUMERAÇÃO DOS PINOS ALOCADOS
int outputAddress[4] = { 22,28,46,30};

//DESCRIÇÃO DOS PINOS ALOCADOS
String buttonText[4] = {
  "01. Lampada", "02. Fechadura", "03. Alarme", "04. Sirene" };

// DETERMINA A CONDIÇÃO INICIAL DOS RELPES
int retainOutputStatus[4] = {0,0,0,0};

//DECLARAÇÃO DAS VARIÁVES USADAS PARA CONTROLE

int outp = 0;
boolean imprimeUltimoComando = false;
boolean imprimeMenuUmaVez = false;
boolean initialPrint = true;
String allOn = "";
String allOff = "";
boolean reading = false;
boolean outputStatus[10];
unsigned long timeConnectedAt;
boolean gravadoNaEeprom = false;
const int tempInPin = A1;
int tempInValue = 0;
float tempOutDeg = 0.0;

//EXECUTA UMA VEZ

void setup(){
dht.begin();
Serial.begin(9600);
iniEepromValor();
lerEepromValor();

//DETERMINA OS PINOS DO ARDUINO COMO SAÍDA
boolean currentState = false;
for (int var = 0; var < outputQuantity; var++){

```

```

pinMode(outputAddress[var], OUTPUT);

if(outputInverted == true) {
  if(outputStatus[var] == 0){currentState = true;}

  else{currentState = false;}

  digitalWrite(outputAddress[var], currentState);
}
else{

  if(outputStatus[var] == 0){currentState = false;}

  else{
    currentState = true;}

  digitalWrite(outputAddress[var], currentState);
}
}

Ethernet.begin(mac, ip, gateway, subnet);

server.begin();
Serial.print("SERVIDOR INICIADO EM ");
Serial.print("-----");
Serial.println(Ethernet.localIP());
}

void loop(){ //LOOP QUE IMPLEMENTA A LEITURA DA TEMPERATURA

  tempInValue = analogRead(tempInPin);
  tempOutDeg = dht.readTemperature();
  checkForClient();

}

```

```
//FUNÇÃO QUE AGUARDA O CLIENTE WEB
////////////////////////////////////
```

```
void checkForClient(){
```

```
    //PORTA ETHERNET PARA O SERVIDOR
    EthernetServer server = EthernetServer(8081);
    EthernetClient client = server.available();
```

```
    if (client) {
```

```
        while (client.connected()) {
            if (client.available()) {
```

```
                // DEFINE A VARIÁVEL USADA PARA TROCA DE INFORMAÇÕES
                char c = client.read();
```

```
                    if(c == '*'){
```

```
                        printHtmlCabecalho(client);
                        printLoginTitulo(client);
                        printHtmlCorpo(client);
                        break;
                    }
```

```
                if(!sentHeader){
```

```
                    printHtmlCabecalho(client);
                    printHtmlTitulo(client); //IMPRIME O TÍTULO NA PÁGINA
```

```
                    sentHeader = true;
                }
```

```
            //CASO A ENTRADA DO USUÁRIO FOR NULL
```

```
            if(reading && c == ' '){
                reading = false;
            }
```

```
            //CARACTERE QUE COLOCA O MICRO CONTROLADOR EM ESPERA
```

```
            if(c == '?') {
                reading = true;
            }
```

```
// LÊ SE É UM COMANDO DE ACIONAMENTO OU DESARMAMENTO  
if(reading){
```

```
    if(c == 'H') {  
        outp = 1;  
    }
```

```
    if(c == 'L') {  
        outp = 0;  
    }
```

```
Serial.print(c); //IMPRIME O VALOR NA SAIDA SERIAL  
Serial.print("-----");
```

```
switch (c) {
```

```
    case '0':
```

```
        acionaPin(outputAddress[0], client, outp);  
        break;
```

```
    case '1':
```

```
        acionaPin(outputAddress[1], client, outp);  
        break;
```

```
    case '2':
```

```
        acionaPin(outputAddress[2], client, outp);  
        break;
```

```
    case '3':
```

```
        acionaPin(outputAddress[3], client, outp);  
        break;
```

```
    case '4':
```

```
        acionaPin(outputAddress[4], client, outp);  
        break;
```

```
    case '5':
```

```
        acionaPin(outputAddress[5], client, outp);  
        break;
```

```
    case '6':
```

```
        acionaPin(outputAddress[6], client, outp);  
        break;
```

```
    case '7':
```

```

        acionaPin(outputAddress[7], client, outp);
        break;
    case '8':

        acionaPin(outputAddress[8], client, outp);
        break;
    case '9':

        acionaPin(outputAddress[9], client, outp);
        break;
    }
}

if (c == '\n' && currentLineIsBlank){
    imprimeUltimoComando = true;
    imprimeMenuUmaVez = true;
    acionaPin(777, client, outp);
    break;
}
}
}

printHtmlCorpo(client); //IMPRIME O RODAPÉ HTML
}
else

{ //SE NÃO HOVER CLIENTE

    if (millis() > (timeConnectedAt + 60000)){

        if (gravadoNaEeprom == true){
            gravaEeprom();
            Serial.println("Sem clientes por mais de um minuto - Atualizando EEPROM.");
            gravadoNaEeprom = false;
        }
    }
}
}
}

```

FUNÇÃO DE ACIONAMENTO DOS PINOS

////////////////////////////////////

```

void acionaPin(int pin, EthernetClient client, int outp){
    //GERA O COMANDO DO ARDUINO PARA O RELE

```

```

if (pin != 777){

    if(outp == 1) {
        digitalWrite(pin, HIGH);
    }

    if(outp == 0){
        digitalWrite(pin, LOW);
    }

}

//ATUALIZA AS LEITURAS
readOutputStatuses();

//IMPRIME OS BOTÕES HTML
if (imprimeMenuUmaVez == true){
    imprimeBotoesHtml(client);
    imprimeMenuUmaVez = false;
}

}

void printHtmlBotoes(EthernetClient client){

//CRIA UMA TABELA NA PÁGINA HTML
client.println("");
client.println("<FORM>");
client.println("<table border=\"0\" align=\"center\">");

//IMPRIME A TEMPERATURA NA PÁGINA
client.print("<tr>\n");

client.print("<td><h4>");
client.print("Temperatura");
Serial.print("Temperatura");
client.print("</h4></td>\n");
client.print("<td></td>");
client.print("<td>");
client.print("<h3>");
client.print(tempOutDeg);
Serial.print(tempOutDeg);
//IMPRIME MENSAGEM DE RETORNO COM A TEMPERATURA NO MONITOR SERIAL.

client.print("<td></td>");

```

```

    client.print("</tr>");
//FECHA A TABELA

for (int var = 0; var < outputQuantity; var++) {

    allOn += "H";
    allOn += outputAddress[var];
    allOff += "L";
    allOff += outputAddress[var];

    client.print("<tr>\n");
    client.print("<td><h4>");
    client.print(buttonText[var]);
    client.print("</h4></td>\n");

//IMPRIME OS BOTÕES ON

    client.print("<td>");
    client.print("<INPUT TYPE=\"button\" VALUE=\"ON \");
    client.print("" onClick=\"parent.location=?H");
    client.print(var);
    client.print(""></td>\n");

//IMPRIME OS BOTÕES OFF

    client.print(" <td><INPUT TYPE=\"button\" VALUE=\"OFF");
    client.print("" onClick=\"parent.location=?L");
    client.print(var);
    client.print(""></td>\n");

    if (outputStatus[var] == true ){
        if (outputInverted == false){
            client.print("<td><div class='green-circle'><div
class='glare'></div></div></td>\n");
        }
        else{
            client.print(" <td><div class='black-circle'><div
class='glare'></div></div></td>\n");
        }
    }
    else
    {
        if (outputInverted == false){

```

```

        client.print(" <td><div class='black-circle'><div
class='glare'></div></div></td>\n");
    }
    else{
        client.print(" <td><div class='green-circle'><div
class='glare'></div></div></td>\n");
    }
}

    client.print("</tr>\n");
}

if (ligarTodosBotoes == true){

}

//FECHA A TABELA HTML COM OS BOTÕES ON, OFF E AS DESCRIÇÕES.
client.println("</table>");
client.println("</FORM>");
//client.println("</p>");

}

LÊ AS O STATUS DE SAÍDA
////////////////////////////////////

void readOutputStatuses(){
    for (int var = 0; var < outputQuantity; var++) {
        outputStatus[var] = digitalRead(outputAddress[var]);
        //Serial.print(outputStatus[var]);
    }
}

}

LÊ O STATUS NA EEPROM
////////////////////////////////////

void lerEepromValor(){
    for (int adr = 0; adr < outputQuantity; adr++) {
        outputStatus[adr] = EEPROM.read(adr);
    }
}
}

```

GRAVA OS VALORES ATUALIZADOS NA EEPROM

//

```
void gravaEeprom(){
  for (int adr = 0; adr < outputQuantity; adr++) {
    EEPROM.write(adr, outputStatus[adr]);
  }
}
```

```
void iniEepromValor(){
  for (int adr = 0; adr < outputQuantity; adr++){
    if (EEPROM.read(adr) > 1){
      EEPROM.write(adr, 0);
    }
  }
}
```

GERA O LAYOUT DO CORPO DA PÁGINA

//

```
void printHtmlCorpo(EthernetClient client){
  Serial.print("Disponibilizando HTML em ms");
  timeConnectedAt = millis();
  Serial.print(timeConnectedAt);
  gravadoNaEeprom = true;
  Serial.print("-----");

  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connnection: close");
  client.println();
  client.println("<!DOCTYPE HTML>");
  client.println("<head>");

  client.println("<title>Automação Residencial</title>");
  client.println("<meta name=\"description\" content=\"Automação Residencial\"/>");

  client.print("<meta http-equiv=\"refresh\" content=\"\"");
  client.print(refreshPage);
  client.println("; url=^>");

  client.println("<meta name=\"apple-mobile-web-app-capable\" content=\"yes\">");
```

```

    client.println("<meta name=\"apple-mobile-web-app-status-bar-style\"
content=\"default\">");
    client.println("<meta name=\"viewport\" content=\"width=device-width, user-
scalable=no\">");

```

```

client.println("<style type=\"text/css\">");
client.println("");

```

```

client.println("html { height:100%; }");

```

```

client.println(" body {");
client.println("  height: 100%;");
client.println("  margin: 0;");
client.println("  font-family: helvetica, sans-serif;");
client.println("  -webkit-text-size-adjust: none;");
client.println(" }");
client.println("");
client.println("body {");
client.println("  -webkit-background-size: 100% 21px;");
client.println("  background-color: #FFFFFF;");
client.println("  background-image:");
client.println("  -webkit-gradient(linear, left top, right top,");
client.println("  color-stop(.75, transparent),");
client.println("  color-stop(.75, rgba(255,255,255,.1) ));");
client.println("  -webkit-background-size: 7px;");
client.println(" }");
client.println("");
client.println(".view {");
client.println("  min-height: 100%;");
client.println("  overflow: auto;");
client.println(" }");
client.println("");
client.println(".header-wrapper {");
client.println("  height: 44px;");
client.println("  font-weight: bold;");
client.println("  text-shadow: rgba(0,0,0,0.7) 0 -1px 0;");
client.println("  border-top: solid 1px rgba(255,255,255,0.6);");
client.println("  border-bottom: solid 1px rgba(0,0,0,0.6);");
client.println("  color: #fff;");
client.println("  background-color: #4cbec1;");
client.println("  background-image:");
client.println("  -webkit-gradient(linear, left top, left bottom,");
client.println("  from(rgba(255,255,255,.4)),");
client.println("  to(rgba(255,255,255,.05)) );");
client.println("  -webkit-gradient(linear, left top, left bottom,");
client.println("  from(transparent),");
client.println("  to(rgba(0,0,64,.1) ));");
client.println("  background-repeat: no-repeat;");

```

```

client.println("  background-position: top left, bottom left;");
client.println("  -webkit-background-size: 100% 21px, 100% 22px;");
client.println("  -webkit-box-sizing: border-box;");
client.println(" }");
client.println("");
client.println(".header-wrapper h1 {");
client.println("  text-align: center;");
client.println("  font-size: 20px;");
client.println("  line-height: 44px;");
client.println("  margin: 0;");
client.println(" }");
client.println("");
client.println(".group-wrapper {");
client.println("  margin: 9px;");
client.println(" }");
client.println("");
client.println(".group-wrapper h2 {");
client.println("  color: #4c566c;");
client.println("  font-size: 17px;");
client.println("  line-height: 0.8;");
client.println("  font-weight: bold;");
client.println("  text-shadow: #fff 0 1px 0;");
client.println("  margin: 20px 10px 12px;");
client.println(" }");
client.println("");
client.println(".group-wrapper h3 {");
client.println("  color: #4c566c;");
client.println("  font-size: 12px;");
client.println("  line-height: 1;");
client.println("  font-weight: bold;");
client.println("  text-shadow: #fff 0 1px 0;");
client.println("  margin: 20px 10px 12px;");
client.println(" }");
client.println("");
client.println(".group-wrapper h4 {");
client.println("  color: #212121;");
client.println("  font-size: 14px;");
client.println("  line-height: 1;");
client.println("  font-weight: bold;");
client.println("  text-shadow: #aaa 1px 1px 3px;");
client.println("  margin: 5px 5px 5px;");
client.println(" }");
client.println("");
client.println(".group-wrapper table {");
client.println("  background-color: #fff;");
client.println("  -webkit-border-radius: 10px;");
client.println("  -moz-border-radius: 10px;");
client.println("  -khtml-border-radius: 10px;");
client.println("  border-radius: 10px;");
client.println("  font-size: 17px;");

```

```

client.println(" line-height: 20px;");
client.println(" margin: 9px 0 20px;");
client.println(" border: solid 1px #a9abae;");
client.println(" padding: 11px 3px 12px 3px;");
client.println(" margin-left:auto;");
client.println(" margin-right:auto;");
client.println(" -moz-transform :scale(1);");
client.println(" -moz-transform-origin: 0 0;");
client.println(" }");
client.println("");
client.println(".green-circle {");
client.println(" display: block;");
client.println(" height: 23px;");
client.println(" width: 23px;");
client.println(" background-color: #0f0;");
client.println(" -moz-border-radius: 11px;");
client.println(" -webkit-border-radius: 11px;");
client.println(" -khtml-border-radius: 11px;");
client.println(" border-radius: 11px;");
client.println(" margin-left: 1px;");

client.println(" background-image: -webkit-gradient(linear, 0% 0%, 0% 90%,
from(rgba(46, 184, 0, 0.8)), to(rgba(148, 255, 112, .9)));@");
client.println(" border: 2px solid #ccc;");
client.println(" -webkit-box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px;");
client.println(" -moz-box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px; /* FF 3.5+
*/");
client.println(" box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px; /* FF 3.5+ */");

client.println(" }");
client.println("");

client.println(".black-circle {");
client.println(" display: block;");
client.println(" height: 23px;");
client.println(" width: 23px;");
client.println(" background-color: #040;");
client.println(" -moz-border-radius: 11px;");
client.println(" -webkit-border-radius: 11px;");
client.println(" -khtml-border-radius: 11px;");
client.println(" border-radius: 11px;");
client.println(" margin-left: 1px;");
client.println(" -webkit-box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px;");
client.println(" -moz-box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px; /* FF 3.5+
*/");
client.println(" box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px; /* FF 3.5+ */");
client.println(" }");
client.println("");

```

```

client.println(" .glare {");
client.println("   position: relative;");
client.println("   top: 1;");
client.println("   left: 5px;");
client.println("   -webkit-border-radius: 10px;");
client.println("   -moz-border-radius: 10px;");
client.println("   -khtml-border-radius: 10px;");
client.println("   border-radius: 10px;");
client.println("   height: 1px;");
client.println("   width: 13px;");
client.println("   padding: 5px 0;");
client.println("   background-color: rgba(200, 200, 200, 0.25);");
client.println("   background-image: -webkit-gradient(linear, 0% 0%, 0% 95%,
from(rgba(255, 255, 255, 0.7)), to(rgba(255, 255, 255, 0)));");
client.println(" }");
client.println("");
client.println("</style>");
client.println("</head>");
client.println("<body>");
client.println("<div class=\"view\">");
client.println("  <div class=\"header-wrapper\">");
client.println("    <h1>Automa&ccedil;&atilde;o Residencial</h1>");
client.println("  </div>");

```

```

} //FINAL DO CORPO HTML

```

GERA O LAYOUT DO CABEÇALHO

```

////////////////////////////////////
void printHtmlCabecalho(EthernetClient client){

```

```

  imprimeUltimoComando = false;
  imprimeMenuUmaVez = false;
  allOn = "";
  allOff = "";

```

```

  client.println("\n<h3 align=\"center\">TCC - Lucas Gomes Sombra");
  client.println(rev);
  client.println("</h3></div>\n</div>\n</body>\n</html>");

```

```

  delay(1);

```

```

  client.stop(); // FECHA A CONEXÃO

```

```

  Serial.println(" Done. Closing conection");
  Serial.println("-----");

```

```
    delay (2);
```

```
}
```

```
GERA O TÍTULO DA PÁGINA
```

```
////////////////////////////////////
```

```
void printHtmlTitulo(EthernetClient client){  
    client.println("<div class=\"group-wrapper\">");  
    client.println("  <h2>Escolha um dos sistemas:</h2>");  
    client.println();  
}
```

```
}
```

```
====-=====-----======FIM=====-----=====
```