UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA

CURSO DE ENGENHARIA INDUSTRIAL ELÉTRICA - ELETROTÉCNICA

GUILHERME SARTORI

LEANDRO ARIEL MOLINA

WILLIAN CEZAR GONÇALVES DE LIMA

DESENVOLVIMENTO DE UM SISTEMA MICROCONTROLADO DE BAIXO CUSTO UTILIZANDO SMARTPHONE PARA APLICAÇÕES DE AUTOMAÇÃO RESIDENCIAL

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2015

GUILHERME SARTORI
LEANDRO ARIEL MOLINA
WILLIAN CEZAR GONÇALVES DE LIMA

DESENVOLVIMENTO DE UM SISTEMA MICROCONTROLADO DE BAIXO CUSTO UTILIZANDO SMARTPHONE PARA APLICAÇÕES DE AUTOMAÇÃO RESIDENCIAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Engenheiro Eletricista, do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Amauri Amorin Assef

CURITIBA
2015

Guilherme Sartori
Leandro Ariel Molina
Willian Cezar Gonçalves de Lima

DESENVOLVIMENTO DE UM SISTEMA MICROCONTROLADO DE BAIXO CUSTO UTILIZANDO SMARTPHONE PARA APLICAÇÕES DE AUTOMAÇÃO RESIDENCIAL

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro Eletricista, do curso de Engenharia Industrial Elétrica – Ênfase Eletrotécnica do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, 11 de fevereiro de 2014.

_____
Prof. Emerson Rigoni, Dr.
Coordenador de Curso
Engenharia Elétrica

_____
Profa. Annemarlen Gehrke Castagna, Mestre
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia Elétrica do DAELT

ORIENTAÇÃO

_____
Amauri Amorin Assef, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

BANCA EXAMINADORA

_____
Daniel Balieiro Silva, Me.
Universidade Tecnológica Federal do Paraná

_____
Roberto Cesar Betini, Dr.
Universidade Tecnológica Federal do Paraná

_____
Amauri Amorin Assef, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia Industrial Elétrica – Ênfase Eletrotécnica

Aos pais e familiares, pelo apoio, suporte e incentivo por todos esses anos.

AGRADECIMENTOS

Agradecemos a Deus, pois a Ele pertence todas as coisas.

Aos nossos professores que com toda paciência nos auxiliaram, permitindo a chegada deste momento, principalmente nosso orientador Dr. Amauri Amorin Assef, que muito nos auxiliou para o desenvolvimento deste trabalho.

Aos amigos da UTFPR, cuja sinceridade, carinho e apoio, os mantiveram sempre presentes durante nossa jornada.

RESUMO

SARTORI, Guilherme; MOLINA, Leandro Ariel; DE LIMA, Willian Cezar Gonçalves. **Desenvolvimento de um sistema microcontrolado de baixo custo utilizando smartphone para aplicações de automação residencial**. 2015. 87 f. Trabalho de Conclusão de Curso - Curso de Engenharia Industrial Elétrica. Universidade Tecnológica Federal do Paraná, Curitiba, 2015.

Este trabalho realiza estudos sobre domótica com sistemas eletrônicos destinados à aplicação residencial. Descreve também o método de desenvolvimento de um sistema de automação residencial de baixo custo através de *smartphones* e Arduino, visando o conforto, segurança e bem estar do usuário nas atividades do cotidiano, considerando aspectos econômicos e simplicidade. Desenvolvido para web, o programa garante compatibilidade com diversos sistemas operacionais e interface simples e amigável.

Palavras-chave: Domótica. Automação residencial. Arduino. *Smartphones.* Web.

ABSTRACT

SARTORI, Guilherme; MOLINA, Leandro Ariel; DE LIMA, Willian Cezar Gonçalves. **Development of a low cost system using smartphone for home automation**. 2015. 87 f. Trabalho de Conclusão de Curso - Curso de Engenharia Industrial Elétrica. Universidade Tecnológica Federal do Paraná, Curitiba, 2015.

This work conducts studies about domotic using electronics systems for residential application. It also describes the development method of a low cost home automation system, through smartphones and Arduino, ensuring comfort, safety and well-being of the user in daily activities, considering economic aspects and simplicity. Designed for web, the program assures compatibility on most operating systems and simple and friendly interface.

Keywords: Domotic. Home automation. Arduino. Smartphones. Web.

LISTA DE FIGURAS

Figura 1 – Arduino Mega 15
Figura 2 – Arquitetura Arduino 19
Figura 3 – Conceito de automação residencial 23
Figura 4 – Arduino Uno 35
Figura 5 – Arduino Nano 36
Figura 6 – Arduino Lilypad 36
Figura 7 – Arduino Mega 37
Figura 8 – Arduino Ethernet 38
Figura 9 – Modulo de relés com oito canais 39
Figura 10 – Módulo Dimmer AC 39
Figura 11 – Diagrama em blocos de um microcontrolador genérico 50
Figura 12 – Projeto Automação Residencial controlado por Arduino 60
Figura 13 – Software IDE Arduino (versão para sistema operacional Windows) 61
Figura 14 – Tela do projeto de automação residencial baseado em Arduino. 65
Figura 15 – Tela de controle do XAMPP. 66
Figura 16 – Tela de boas vindas do XAMPP. 67
Figura 17 – Arduino instalado na residência. 69

LISTA DE TABELAS

Tabela 1 – Tipos de variáveis para desenvolvimento com Arduino (BANZI, 2012)...42
Tabela 2 – Palavras-chave e estruturas de programação do Arduino (BANZI, 2012). 43
Tabela 3 – Operadores Aritméticos (BANZI, 2012). 44
Tabela 4 – Operadores Relacionais (BANZI, 2012). 45
Tabela 5 – Funções para a manipulação de entradas e saídas (BANZI, 2012). 46
Tabela 6 – Funções capazes de medir o tempo transcorrido e também pausar o sketch (BANZI, 2012). 47
Tabela 7 – Funções matemáticas e trigonométricas (BANZI, 2012). 47
Tabela 8 – Funções de números aleatórios (BANZI, 2012). 48
Tabela 9 – Tabela de custos dos equipamentos utilizados. 59

LISTA DE QUADROS

Quadro 1 – Programação Ethernet do Arduino 61
Quadro 2 – Programação de variáveis do Arduino.....62
Quadro 3 – Programação de status das Portas. 63
Quadro 4 – Programação em PHP para criação do socket.....64
Quadro 5 – Programação em PHP para o comando socket_write. 64
Quadro 6 – Programação de botões. 65
Quadro 7 – Programação do botão refresh. 65

LISTA DE SIGLAS

AC	*Alternating Current*
ADC	*Analogic Digital Converter*
ANSI	*American National Standards Institute*
AR	Automação Residencial
ARPA	Agência de Projetos de Pesquisas Avançadas
AURESIDE	Associação Brasileira de Automação Residencial
AVAC	Aquecimento, ventilação e ar condicionado
CA	Corrente alternada
CD	*Compact Disc*
CEBus	*Consumer Electronic Bus*
CERN	*Centre European pour la Recherche Nucheaire*
CI	Circuito Integrado
CPU	Central Única de Processamento
D2B	*Digital Domestic Bus*
DARPA	Agência de Projetos de Pesquisas Avançadas do Departamento de Defesa dos Estados Unidos da América
DC	*Direct Current*
DNS	*Domain Name Server*
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
FTDI	*Future Technology Devices International*
HBS	*Home Bus System*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
I/O	*Input/Output*
I2C/TWI	*Two Wire Interface*
ICQ	Mensageiro Instantâneo
ICSP	*In Circuit Serial Programming*
IDE	*Integrated Development Environment*
IP	*Internet Protocol*
LAN	*Local Area Network*
LCD	*Liquid Crystal Display*
LED	*Light Emitter Diode*
MAN	*Metropolitan Area Network*
MCU	*Micro Controler Unit*
MSN	Mensageiro Instantâneo
PAN	*Personal Area Network*
PC	*Personal Computer*
PHP/FI	*Personal Home Page Tools/Forms Interprete*
PWM	*Pulse Width Modulation*
RAM	*Randon Access Memory*
ROM	*Read Only Memory*
RTC	Relógio de Tempo Real
SGML	*Standard Generalized Markup Language*
SPI	*Serial Peripheral Interface*
SRAM	*Static Ramdon Access Memory*
SRI	*Stanford Research Institute*

SSI	*Server Side Includes*
SSL	*Secure Sockets Layer*
TCC	Trabalho de Conclusão de Curso
TCP	*Transmission Control Protocol*
TRIAC	*Triode for Alternating Current*
UART	*Universal Asynchronous Receiver/Transmitter*
UCLA	Universidade da Califórnia – Los Angeles
UCSB	Universidade da Califórnia – Santa Bárbara
ULA	Unidade Lógica Aritmética
USB	*Universal Serial Bus*
VAX	Arquitetura de Computadores de 32 *bits.*
WAN	*Wide Area Network*
WATTS	Unidade de Potência no Sistema Internacional de Unidades
WI-FI	*Wireless Fidelity*

SUMÁRIO

1 INTRODUÇÃO 14
1.1 TEMA 16
1.1.1 Delimitação do Tema 16
1.2 PROBLEMA E PREMISSAS 16
1.3 OBJETIVOS 18
1.3.1 Objetivo Geral 18
1.3.2 Objetivos Específicos 18
1.4 JUSTIFICATIVA 20
1.5 PROCEDIMENTOS METODOLÓGICOS 21
1.6 ESTRUTURA DO TRABALHO 21
2 A AUTOMAÇÃO RESIDENCIAL 23
2.1 DEFINIÇÃO DE AUTOMAÇÃO RESIDENCIAL 23
2.2 BREVE HISTÓRICO DA AUTOMAÇÃO RESIDENCIAL 24
2.3 VANTAGENS DA AUTOMAÇÃO RESIDENCIAL 26
2.4 ESTÁGIO ATUAL DO MERCADO 28
2.4.1 Soluções Comerciais Disponíveis no Mercado 30
3 O ARDUINO E OS MICROCONTROLADORES 32
3.1 O ARDUINO 32
3.1.1 Breve História do Arduino 32
3.1.2 O Arduino e seus Componentes 33
3.1.3 Métodos de Programação para Arduino 39
3.1.3.1 *Software* para Arduino 40
3.1.3.2 A linguagem do Arduino 40
3.1.3.3 Estrutura do Arduino 41
3.1.3.4 Símbolos especiais 41
3.1.3.5 Comentários 41
3.1.3.6 Constantes 42
3.1.3.7 Variáveis 42
3.1.3.8 Estruturas de controle 43
3.1.3.9 Aritmética e fórmulas 44

3.1.3.10 Operadores de comparação 45
3.1.3.11 Funções de entrada e saída 46
3.1.3.12 Funções de tempo 47
3.1.3.13 Funções matemáticas 47
3.1.3.14 Funções de números aleatórios 48
3.2 OS MICROCONTROLADORES 48
4 REDES, LINGUAGENS DE PROGRAMAÇÃO E SERVIDORES WEB 51
4.1 REDES 51
4.1.1 Classificação das Redes 52
4.2 LINGUAGENS DE PROGRAMAÇÃO 53
4.2.1 Compiladores 54
4.2.2 Linguagem C 54
4.2.3 Linguagem para a Web 55
4.2.3.1 PHP 56
4.3 SERVIDORES WEB 57
4.3.1 Servidor Apache 57
5 DESENVOLVIMENTO DO PROJETO 59
5.1 ARDUINO NO CONTROLE DA AUTOMAÇÃO RESIDENCIAL 59
5.1.1 Custos dos Equipamentos Utilizados no Projeto 59
5.1.2 Programação do Arduino 60
5.1.3 Programação do Servidor em PHP 63
5.1.4 Instalação do Servidor Web APACHE 66
5.1.5 Comandos no *Smartphone* 68
5.1.6 Implantação Física do Sistema 69
6 DISCUSSÃO 71
7 CONCLUSÃO 73
REFERÊNCIAS 75
APÊNDICE A – CÓDIGO FONTE DA INTERFACE WEB EM PHP 79
APÊNDICE B – FLUXOGRAMA DO ALGORITMO 84
APÊNDICE C – PLANTA DO SOBRADO AUTOMATIZADO 86

1 INTRODUÇÃO

Nos dias atuais, com o desenvolvimento tecnológico acelerado, no qual a *Internet* está ao alcance de todos através de *Smartphones*, *Tablets*, *I-Pads*, etc., é possível utilizar esta tecnologia no âmbito residencial, tornando as residências mais inteligentes, funcionais e seguras. A larga utilização dos computadores pessoais, telefonia móvel e *Internet*, projeta um mercado emergente nas questões referentes às tecnologias residenciais. Surge, então, o termo “domótica”.

A palavra "domótica" deriva das palavras *domus* (casa) e *robótica* (controle automatizado de algo), definindo-se como a possibilidade de controle de forma automática das residências, tornando-as, no que se costuma designar, por “casas inteligentes” (ALVES & MOTA, 2003). Segundo Alves e Mota (2003), casas inteligentes são as que possuem características capazes de tornar a vida mais simples a quem nelas habita, sendo agrupadas em cinco categorias principais: segurança, economia, conforto, ecologia, integração.

As exigências nas instalações prediais, tais como alta confiabilidade e baixo consumo energético são necessidades tal como a segurança, proteção e facilidade de operação. Tais exigências sempre existem e são cada vez mais rigorosas para atender as necessidades dos usuários. Para isso, são introduzidas novas tecnologias nos processos de construção, tais como monitoramento e controle, introduzindo-se a automatização (MIYAGI; BARRETO; SILVA, 1993).

> Recuemos então um pouco até os anos 70, onde os sistemas AVAC (Aquecimento, Ventilação e Ar Condicionado) foram os primeiros sistemas de edifícios a serem eletronicamente controlados [...]. Foi a partir desta altura que se fomentou o desenvolvimento da ideia de tornar os edifícios dotados de inteligência [...]. Um pouco mais tarde (já nos anos 80) apareceram os sistemas de automação de segurança, iluminação e intrusão, mostrando coordenação entre os componentes do mesmo sistema (ALVES & MOTA, 2003).

De acordo com o *site* AURESIDE (2013) – Associação Brasileira de Automação Residencial –, a automação residencial é percebida pelo cliente como um símbolo de *status* e modernidade em um primeiro estágio. No seguinte, o conforto e a conveniência por ela proporcionados passam a ter uma função decisiva,

pois, em último estágio, ela se tornará uma necessidade vital e um fator de economia.

Desta forma, este trabalho pretende apresentar o desenvolvimento de um dispositivo prático e de baixo custo, baseado na plataforma livre Arduino e *smartphone*, para prover às pessoas maior conforto, segurança e autonomia, por meio do controle e monitoramento remoto dos equipamentos e aparelhos de sua residência.

> O Arduino (Figura 1) é comumente chamado de plataforma de computação física ou embarcada, ou seja, um sistema que pode interagir com seu ambiente por meio de *hardware* e *software*. Há uma grande comunidade de pessoas utilizando Arduinos, compartilhando seus códigos e diagramas de circuito para que outros os copiem e modifiquem. A maioria dessa comunidade também está muito disposta a auxiliar outros desenvolvedores (McROBERTS, 2011).

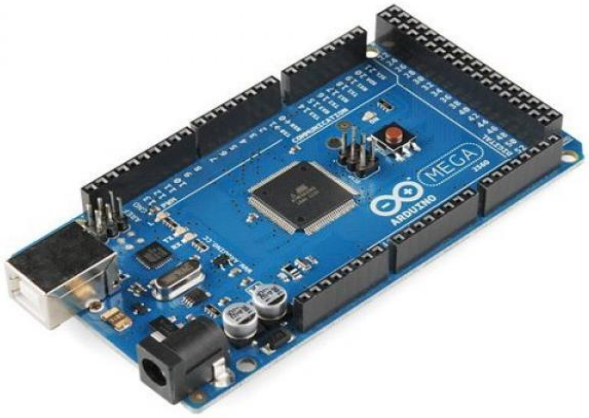


Figura 1 – Arduino Mega
Fonte: Techmount (2013).

"O *hardware* e o *software* do Arduino são ambos de domínio público, o que significa que o código, os esquemas, o projeto, etc., podem ser utilizados livremente por qualquer pessoa e com qualquer propósito" (McROBERTS, 2011).

Segundo o *site* G1 (2015), cada quatro celulares vendidos no Brasil, três são *smartphones.* Desta forma, com o aumento de sua popularidade, é um recurso valioso que pode ser também utilizado para Automação Residencial.

1.1 TEMA

Apresenta-se como tema deste projeto o desenvolvimento de um sistema microcontrolado de baixo custo utilizando a tecnologia *smartphone* para aplicações de automação residencial.

1.1.1 Delimitação do Tema

Desenvolvimento de um sistema microcontrolado de custos inferiores aos praticados no mercado para controle e monitoramento remoto dos equipamentos eletrônicos e aparelhos eletrodomésticos de uma residência, com possibilidade de utilização do *hardware* livre Arduino e implementação de *software* supervisório em *smartphones, tablets ou I-pads.*

1.2 PROBLEMA E PREMISSAS

O objetivo da domótica é controlar e monitorar, local ou remotamente as funções e estados dos equipamentos de uma casa por meio de um sistema supervisório, permitindo com que as tarefas diárias fiquem mais simples de ser realizadas.

Além disso, pode-se afirmar que a domótica promete proporcionar conforto, segurança e economia, além de manter a residência melhor adaptada às inovações tecnológicas futuras, conforme Alves e Mota (2003) apontam: “Do ponto de vista do investimento, há que ter em conta que a evolução tecnológica e o consequente aumento das funcionalidades, elevam as expectativas do comprador modificando assim a tradicional evolução do valor de mercado imobiliário”. Ainda segundo Alves e Mota (2003), o custo de uma casa inteligente orça entre 2% e 10% do valor total

da casa, concluindo tratar-se de um bom investimento, com retorno imediato e futuro.

Uma das principais características do conceito de automação residencial está na forma em que todas as funcionalidades se integram e complementam, com a informação trafegando entre os sistemas de segurança, equipamentos, ar condicionado, portas e janelas. Com isso, é possível, por exemplo, um sinal de abertura de uma porta ser enviado simultaneamente ao sistema de segurança, indicando uma possível intrusão ao imóvel. A residência pode ser monitorada remotamente pelo proprietário. Qualquer ocorrência, medição ou estado pode modificar ao mesmo tempo o funcionamento de qualquer sistema, conforme opção e critério estabelecido pelo morador.

É possível “assim conceber o funcionamento de cada sistema em função do resultado de uma matriz de eventos e resultados, concebida e adaptada às preferências e necessidades de cada indivíduo, preparada para ser ajustada dinamicamente no tempo” (ALVES & MOTA, 2003).

Quando o assunto é conforto, a climatização passa a ser um fator de grande relevância. O usuário pode, por exemplo, ativar remotamente o sistema de climatização da casa e então ao entrar na residência encontrar o ambiente em condições mais favoráveis se comparado ao clima externo.

Toda esta tecnologia ainda permite inúmeras vantagens, inclusive na adaptação da residência a pessoas da terceira idade e portadores de deficiências físicas por facilitar as tarefas do cotidiano.

Diante disso, como é possível facilitar o manuseio de eletrodomésticos, iluminação, ventilação, controle de acesso dentro de uma residência deixando-a mais confortável, segura e integrada com a palma das mãos e ainda reduzir os custos da implantação de um sistema de automação residencial?

Para responder a este questionamento, foram realizadas pesquisas sobre soluções integradas para o desenvolvimento de um sistema de automação residencial, visando otimizar a relação custo-benefício para o projeto.

Há também a possibilidade futura de desenvolver plataformas que permitam controlar aberturas de portas e janelas, por meio de dispositivos eletromecânicos além de dispositivos para controlar de maneira integrada e inteligente a energia elétrica utilizada e seus custos.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Este trabalho tem como objetivo desenvolver um sistema de automação residencial custos abaixo dos praticados no mercado que permita ao usuário ligar ou desligar lâmpadas e eletrodomésticos de sua residência e verificar remotamente o *status* do que está ligado ou desligado, com o auxílio de microcontroladores integrados com sistemas WI-FI[1] e *Internet*.

O projeto baseia-se na utilização da plataforma livre Arduino, juntamente com a implementação de *software* supervisório que atuará diretamente nos *smartphones, tablets, I-pads,* controlando e monitorando os equipamentos eletrônicos e aparelhos domésticos ligados ao sistema. Desta forma, o usuário terá centralizado em um único ponto o *status* e o controle dos equipamentos.

1.3.2 Objetivos Específicos

Seguem os objetivos específicos conforme as etapas do trabalho.

Etapa 1: Avaliar o *hardware* Arduino com arquitetura apresentada na Figura 2. Esta arquitetura é composta por 4 UARTs (*Universal Asynchronous Receiver/Transmitter*), que são as interfaces seriais de dados utilizadas para a comunicação de transmissão e recepção de dados entre um computador e o *hardware* Arduino. O sistema possui 8 KB de memória volátil SRAM e 256 KB de memória Flash para armazenar as instruções programadas, sendo que destes, 8 KB é utilizado pelo *bootloader*, ou gerenciador de *boot*, que é o código executado durante a inicialização do microcontrolador para transferência da aplicação

[1] "*Wireless Fidelity*" significa "sem fio" em português. Consiste em uma tecnologia de comunicação que não faz uso de cabos e, geralmente, é transmitida através de frequências de radio ou infravermelho.

desenvolvida pelo usuário. Além de possuir um processador de 16 MHz, possui 54 portas digitais de entrada/saída (dos quais 15 podem ser usados como saídas PWM) e 16 entradas analógicas (ARDUINO, 2014).

Etapa 2: A partir do objetivo geral, desenvolver funções para automatizar o funcionamento de lâmpadas, eletrodomésticos, abertura e fechamento de portão eletrônico e portas com comandos através da *Internet*.

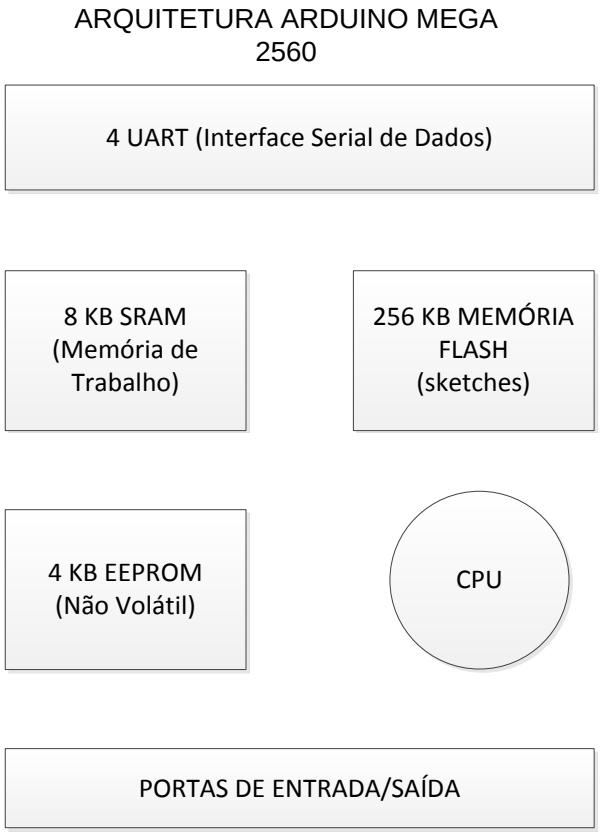


Figura 2 – Arquitetura Arduino
Fonte: Programação com ARDUINO (2013).

Etapa 3: Conectar os equipamentos e aparelhos à automação residencial que será programado para que o usuário possa, através de um *software* instalado em seu dispositivo móvel, monitorar e comandar remotamente seus estados. Além disso, possibilitar o monitoramento remoto do estado dos dispositivos conectados ao sistema supervisório.

Etapa 4: Utilizando todos os sensores e módulos, que podem ser acoplados a um sistema microcontrolado proposto, avaliar a possibilidade futura de

desenvolvimento de outras funcionalidades para este projeto de automação residencial que serão sugeridas neste projeto, porém não executadas.

1.4 JUSTIFICATIVA

Por se tratar de um projeto que poderá beneficiar os usuários com um sistema capaz de monitorar e controlar diversos dispositivos domésticos de forma prática e simplificada é que este trabalho se mostra relevante.

O sistema desenvolvido poderá ser testado em bancada e em qualquer residência com instalação elétrica adequada. Além disso, devido ao baixo custo se comparado com as opções de automatização residencial existentes no mercado, poderá ser replicado facilmente em outras residências de modo a proporcionar às pessoas maior conforto, segurança e comodidade.

Para o trabalho proposto foi adotado o Arduino Mega R3 2560, que é *kit* de desenvolvimento com microcontrolador ATmega2560, devido ao seu baixo custo e capacidade de integração com diversos tipos de sensores e módulos de acionamento. O componente possui 33 entradas/saídas digitais, 8 saídas PWM (*Pulse-Width Modulation*), 16 entradas analógicas (pinos A0 até A15, cristal oscilador de 16 MHz, um conector de alimentação, conector ICSP (*In Circuit Serial Programming*), conexão USB (*Universal Serial Bus*) e um botão de *reset*. A placa também suporta cartão de memória SD de leitura/gravação. Entre as principais vantagens da utilização do *kit*, destaca-se a possibilidade de utilização de placas módulos *shields* com diversos periféricos para avaliação e desenvolvimento, como por exemplo:

- Módulo Ethernet compatível com Arduino Mega. O Arduino Ethernet *Shield* permite que uma placa Arduino se conecte a *Internet*. A placa é baseada no *chip* Wiznet Ethernet W5100 (WIZnet Co., Coreia do Sul);
- Módulo de relés com 8 para Automação com optoacoplador. Possui interface padrão que pode ser controlado diretamente por microcontrolador, onde o relé suporta correntes de 10 A, com indicação de LED de saída do relé;
- Módulo de reconhecimento de comando de voz;
- Módulo LED de iluminação;

- Sensor Analógico de luz ambiente
- Sensor de toque capacitivo;
- Sensor de temperatura LM35 analógica linear;
- Sensor magnético digital;
- Sensor de vibração digital;
- Sensor de inclinação digital;
- Sensor *Big Push Button*;
- Sensor analógico *Grayscale* (tons de cinza).

1.5 PROCEDIMENTOS METODOLÓGICOS

Pesquisa da literatura acerca do *hardware* livre Arduino e verificar suas possibilidades de utilização conforme os objetivos propostos.

Pesquisa acerca da implementação do *hardware* necessário para o tema, utilizando componentes microcontrolados.

Desenvolvimento do *firmware* para programação do microcontrolador.

Desenvolvimento do *software* para utilização nos dispositivos móveis.

Implementação do sistema microcontrolado proposto com conexão de placa de circuito impresso com relés para ligar e desligar os pontos a serem automatizados.

Comparação dos custos entre os dois métodos de implantação do projeto.

1.6 ESTRUTURA DO TRABALHO

Capítulo 1 - Introdução: Contém a apresentação do trabalho, tema, delimitação do tema, problemas, premissas, objetivos, justificativas e procedimentos tecnológicos para a sua realização.

Capítulo 2 - A automação residencial: Este capítulo descreve uma abordagem geral sobre automação residencial e sobre métodos existentes de uma automação residencial.

Capítulo 3 - O Arduino e os Microcontroladores: Abordagem sobre a plataforma de prototipagem eletrônica de *hardware* livre Arduino, sua história, seus componentes, métodos de programação, e demais microcontroladores disponíveis no mercado.

Capítulo 4 - Redes, Linguagens de Programação e Servidores Web: Este capítulo aborda aspectos básicos das Redes, Linguagens de Programação e Servidores Web. Inicia-se com um breve histórico, seguido pelas suas características, métodos e funcionalidades, como base para o desenvolvimento do projeto.

Capítulo 5 - Desenvolvimento do projeto: Métodos utilizados para o desenvolvimento e implementação dos parâmetros para atingir o objetivo esperado. Desenvolvimento da programação implementada para realizar êxito na automação desejada, incluindo *software* de monitoramento e controle para utilização no *smartphone*.

Capítulo 6 - Discussão: As metas alcançadas no capítulo anterior são abordadas neste capítulo. Levantamento das dificuldades encontradas, meios para solução dos problemas e resultados práticos obtidos.

Capítulo 7 – Conclusão: Confronto entre os objetivos propostos, justificativas e resultados obtidos ao longo do trabalho.

2 A AUTOMAÇÃO RESIDENCIAL

2.1 DEFINIÇÃO DE AUTOMAÇÃO RESIDENCIAL

Segundo Prudente (2013), a automação residencial (em inglês, *home automation*) define-se como a tecnologia que estuda a automação de um prédio ou habitação, sendo domótica, que deriva do neologismo francês "*domotique*", o termo que a identifica, significando literalmente "casa automática".

A automação residencial (AR) é um ramo da automação predial que tem como objetivo o controle de processos no universo doméstico, com isso reduzindo a necessidade de intervenção humana no gerenciamento de equipamentos eletroeletrônicos através de sistemas de controle (Figura 3). Informações sobre o ambiente são coletadas por sensores e analisadas para a tomada de decisão segundo um programa predeterminado. Essas decisões podem disparar ações que podem alterar o estado de atuadores modificando as condições do ambiente (BOLZANI, 2004; BOLZANI, 2010).

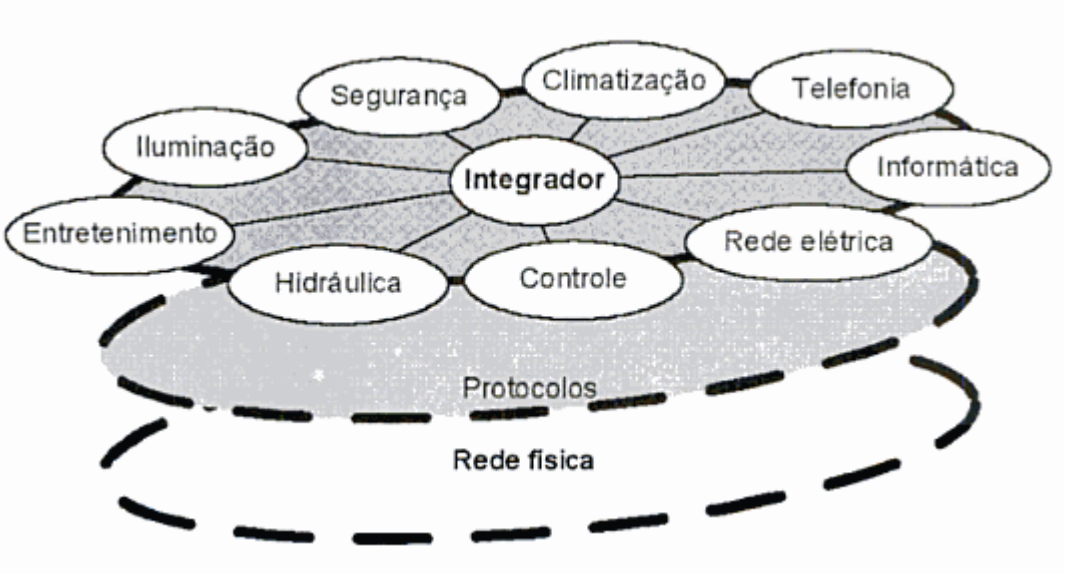


Figura 3 – Conceito de automação residencial
Fonte: Bolzani (2004).

> Cada nova tecnologia traz acoplado um novo vocabulário. Quando o assunto é residência inteligente, não é diferente: casa automática, casa

> inteligente, automação residencial, *retrofitting*[2], domótica[3], etc., mas tudo pode ser resumido a uma só palavra: conforto. Se um sistema eletrônico instalado em um ambiente não oferecer conforto ao usuário, em semanas ele vai ser desligado e deixado de lado [...]. Os equipamentos devem unificar os controles e processos tornando tudo mais simples. Mas é o desejo do usuário que deve prevalecer e não a do PC. A casa automática pode ajudar nas tarefas diárias que tomam muito tempo ou evitar preocupações tais como esquecer as janelas abertas quando a previsão do tempo avisou que ia chover (BOLZANI, 2004).

Ainda, conforme Bolzani (2004), a palavra domótica originou-se do latim *domus* que significa casa, sendo uma técnica moderna de engenharia dos sistemas prediais e de seus equipamentos. Com estas novas tecnologias pretende-se criar sistemas que possuem características de automatizar os processos repetitivos, tornando as residências inteligentes.

Podem ser consideradas ainda um conjunto de serviços com sistemas integrados, que buscam satisfazer necessidades básicas de conforto, segurança e praticidade de uma residência através de equipamentos capazes de se comunicar entre si seguindo uma programação pré-estabelecida pelo proprietário do imóvel, conforme sua vontade. Dessa forma, obtendo uma melhor qualidade de vida, proporcionando bem estar, segurança e redução de serviços domésticos.

Nesse contexto, com a domótica, a informação fornecida por algum sensor estará à disposição através de uma rede interna, podendo ser modificada conforme vontade do usuário a qualquer momento.

2.2 BREVE HISTÓRICO DA AUTOMAÇÃO RESIDENCIAL

A eletricidade quando introduzida nas residências, transformou o modo de vida das pessoas. Um exemplo que pode ser considerado é a iluminação que melhorou a convivência no período noturno, modificando os hábitos das famílias da época. Além disso, a eletricidade permitiu a entrada de diversos aparelhos nas residências, aumentando o nível de conforto e entretenimento dos moradores. O

---

[2] Em inglês, *retrofitting* é o ato de se introduzir uma modificação em algo previamente construído.

[3] A palavra domótica originou-se do latim *domus* que significa casa. É a ciência moderna de engenharia das instalações em sistemas prediais (BOLZANI, 2004).

radio, o telefone e mais recentemente a televisão podem ser citados para exemplificar esse contexto.

> Tipicamente, a forma como se agregam serviços e equipamentos eletrônicos às casas pouco se alterou. Cada um dos serviços mencionados acima é regido por diferentes modelos de negócios. Os equipamentos são instalados com pouca ou nenhuma interação com os demais e não usufruem de uma infraestrutura comum (BOLZANI, 2010).

Citando Bolzani (2010), desde 1920 com o surgimento dos primeiros eletrodomésticos, nos Estados Unidos, já existia o termo “casa do futuro” referindo-se aos benefícios que esses aparelhos trariam as donas de casa, pois estes poupariam o tempo das pessoas na execução das tarefas rotineiras.

“No fim dos anos 1970, a automação residencial estava dando seus primeiros passos, a tecnologia X10 havia sido criada e os primeiros controladores e consoles de comando já eram comercializados pela Sears e RadioShack nos Estados Unidos” (RYE, 1999 apud BOLZANI, 2013).

Nos anos seguintes começaram a serem desenvolvidos os sistemas de controle: *Digital Domestic Bus* (D2B)*,* o *Home Bus System* (HBS)*,* o *Consumer Electronic Bus* (CEBus) e o *SmartHouse*. Na década de 1980, houve o aquecimento do mercado de computadores pessoais com o surgimento de vários sistemas operacionais, bem como da Ethernet que acabaria se tornando o protocolo de transmissão de dados mais utilizado do mundo (BOLZANI, 2013).

Nesse período apareceram os sistemas de automação de segurança, iluminação e intrusão, mostrando coordenação entre componentes do mesmo sistema. O *Lloyds Building* construído em Londres e projetado pela Richard Roger Partnership foi o primeiro edifício no qual se pôde chamar de “Edifício Inteligente” (ALVES & MOTA, 2003).

Com a atual evolução tecnológica, que torna os produtos lançados obsoletos cada vez mais rápido, como por exemplo, a microeletrônica, a melhoria da banda de comunicação e de capacidade de armazenamento, alguns sistemas de controle que no passado eram inviáveis, passaram a ser objetos de pesquisa.

> Esses avanços tecnológicos, associados ao novo contexto socioeconômico, ambiental e de saúde do século XXI, revelam inúmeras oportunidades para o desenvolvimento de sistemas inteligentes para o ambiente residencial. A automação e o gerenciamento remoto de dispositivos têm sido apontados

> como ferramentas importantes para uma gestão eficiente de recursos energéticos e naturais (BOLZANI, 2013).

Ainda hoje, o termo “casa do futuro” existe na venda de eletrodomésticos, como sinônimo de conforto, porém ainda paira a dúvida sobre os benefícios em se conectar aparelhos em rede monitorando seu funcionamento à distância.

> No entanto, nunca houve uma ocasião tão positiva como a atual para a adoção de novas tecnologias que realmente alterassem de maneira significativa o modo como construímos e usamos as residências. O panorama socioeconômico que leva a mulher e o homem a trabalharem o dia todo, fora de casa, bem como a crise energética e a problemática ambiental são fatores que propiciam a criação de soluções de automação e controle de equipamentos residenciais (BOLZANI, 2010).

2.3 VANTAGENS DA AUTOMAÇÃO RESIDENCIAL

Tendo em vista o atual panorama socioeconômico, as dificuldades energéticas encontradas e os fatores ambientais, o modo como são construídas e usadas novas residências demanda alterações. As oportunidades oferecidas atualmente representam uma grande capacidade de impulsionar o mercado das residências inteligentes. Esse contexto favorece o desenvolvimento de serviços socialmente importantes e lucrativos de automação e controle residencial no Brasil e no mundo (BOLZANI, 2013).

“O mercado de automação predial é um mercado em forte expansão e consolidado há vários anos, desenvolvendo uma oferta articulada e bem recebida pelos clientes” (PRUDENTE, 2013).

O que muitas pessoas desconhecem nos dias atuais é fato de já possuírem inúmeros computadores em suas residências. Aparelhos como geladeiras, micro-ondas e televisores possuem diversos *microchips* que desempenham diversas funções. Porém para acessar esse e outros equipamentos o usuário precisa ir até eles ou estar na área de cobertura do controle remoto. Por exemplo, em caso de algo que está em forno programado de maneira incorreta e começa a queimar, o usuário deve desligá-lo manualmente no local onde se encontra o aparelho.

Em uma instalação tradicional, as instalações atuam de forma separada, e embora possuam boa confiabilidade, são extremamente limitadas, conforme exemplifica Prudente (2013):

- Nível de integração entre instalações praticamente inexistente;
- Funcionalidades limitadas sendo muita vezes impossíveis de implementar ações mais sofisticadas;
- Instalações pouco flexíveis, na qual qualquer alteração requer frequentemente novas linhas elétricas;
- Pouca atenção aos gastos com energia;
- Baixo nível de conforto.

Conforme Bolzani (2004), o ambiente é responsável pela geração de eventos naturais como mudanças de temperatura, luminosidade, umidade, etc. O usuário pode acionar indiretamente os eventos (através da detecção da sua posição pelos sensores) ou diretamente, agindo sobre a interface (acendendo ou apagando uma lâmpada, por exemplo). O sistema observa as alterações e eventos executando as tarefas conforme programação. Essas tarefas podem ser o ajuste da temperatura de um ambiente, o disparo de mensagens frente a uma emergência, etc.

“A integração de dispositivos eletrônicos a objetos comuns transforma o modo como às pessoas lidam com esses objetos e amplia os mecanismos de interação do corpo humano com eles e com o ambiente” (BOLZANI, 2013).

Segundo Bolzani (2007) a AR tem despertado o interesse das pessoas através da *Internet*, pois os números de publicações crescem e estão mais completos e tecnicamente precisos. Além disso, segundo o mesmo autor, a AR tem demonstrado que dispositivos eletrônicos integrados aumentam consideravelmente os benefícios principalmente se comparados a sistemas isolados.

Nesse mesmo contexto, Prudente (2013) mostra que a domótica oferece várias vantagens sobre a tecnologia convencional:

- Maior conforto – Através da domótica o ambiente torna-se mais acolhedor e agradável, pois as novas tecnologias permitem o controle de parâmetros que favorecem uma melhor qualidade de vida para o usuário;
- Maior segurança – É possível aumentar o nível de segurança em uma residência seja contra ação de terceiros ou eventos perigosos. Por exemplo, é

possível controlar a habitação no caso de uma tentativa de furto ou em um caso de incêndio, vazamento de gás, alagamento, etc. Ainda, interruptores, chaves, sensores, etc., podem funcionar à extra baixa tensão;

- Maior versatilidade – É possível alterar a configuração das instalações e de suas funções através de um *software*;
- Maior economia da gestão da instalação – É possível o controle da energia de uma residência (iluminação, aquecimento, ar condicionado, etc.), permitindo uma economia no custo da energia elétrica na da instalação.

Conforme Alves e Mota (2003), a valorização do imóvel em questão é um ponto fundamental na automação residencial: “A valorização do imóvel pela capacidade de acompanhar os progressos tecnológicos, integrando novas funcionalidades, adquirindo novos patamares de segurança e conforto. Um edifício com domótica vale sempre mais do que o mesmo edifício que utilize tecnologias tradicionais”.

2.4 ESTÁGIO ATUAL DO MERCADO

O mercado de AR no Brasil aos poucos está adquirindo características muito próximas às de mercados mais evoluídos. O surgimento de um novo profissional, de nome Integrador de Sistemas Residenciais, que, em função das diferentes tecnologias, da complexidade de projeto, instalação e programação, e não existência de soluções *plug-and-play*, exige uma especialização do profissional que nele atua (MURATORI & DAL BÓ, 2013).

> Inicialmente, o número de fabricantes presentes no mercado teve um rápido crescimento nos últimos anos. Dentre estas empresas, encontramos desde multinacionais e grupos estrangeiros, que têm na Automação Residencial mais um dos segmentos de atuação dentro de uma ampla gama de produtos; até pequenas empresas nacionais que surgiram e cresceram rapidamente fornecendo soluções especificas. Esta maior oferta trouxe uma queda de preços devido à maior concorrência e maior variedade de soluções à disposição dos consumidores (MURATORI, 2014).

Ainda segundo Muratori (2014) – Diretor executivo da AURESIDE – o número de fornecedores tem aumentado e triplicou em menos de cinco anos, porém ainda o mercado se apresenta na sua forma inicial, necessitando de alguns ajustes. Entretanto o consumidor está descobrindo os benefícios da automação residencial, sendo questão de tempo para que sistemas integrados sejam incluídos cada vez mais nas residências.

Levantamentos realizados pela AURESIDE demonstram que o Brasil teria hoje pelo menos 1,8 milhões de residências com potencial para utilizar sistemas automatizados. No entanto ao final de 2013, apenas 300 mil residências atendidas. Ou seja, um déficit de pelo menos 1,5 milhão de residências que precisariam ser atendidas imediatamente (MURATORI, 2014).

A expectativa de Muratori (2014) é um crescimento anual em média de 30% no segmento, pois a demanda tem sido impulsionada pela acessibilidade de novas tecnologias e pela possibilidade de integração com *smartphones* e *tablets*.

O grande problema enfrentado pelo mercado nacional é a capacitação de profissionais para a área, pois segundo dados da AURESIDE, o Brasil conta com apenas 15% do contingente de profissionais necessários para suprir a demanda. Ainda conforme esta Entidade, o Integrador de Sistemas Residenciais deve projetar, instalar e manter os sistemas de automação residencial com um conhecimento do processo como um todo. Estes profissionais também atuam como consultores com entendimento das instalações residenciais e das tecnologias envolvidas nesse contexto.

Devido ao amplo conhecimento necessário, alguns profissionais optam pela especialidade em determinada área, como por exemplo, *home theaters* e sistemas de segurança. A própria Entidade oferece cursos e certificação para novos profissionais que desejam entrar neste mercado emergente.

Há uma inclinação do mercado por dispositivos *plug-and-play*, pois a possibilidade de integração com qualquer dispositivo, independente de marca ou modelo é um grande apelo comercial. Sistemas *wireless* agregam valor aos sistemas mais antigos sem adicionar cabeamento garantindo a mobilidade do usuário. Estes são fatores que estão modelando a criação das redes domésticas (BOLZANI, 2004).

2.4.1 Soluções Comerciais Disponíveis no Mercado

Realizou-se pesquisa de mercado a fim de apresentar soluções disponíveis em domótica no Brasil. Para isso, várias empresas do setor foram consultadas em todo o país. No entanto, das empresas consultadas, apenas duas responderam com interesse em informar um orçamento. Porém, para tanto, foi exigido um projeto elétrico da instalação residencial pelo qual o orçamento seria cotado. Como o projeto elétrico não faz parte do objeto deste trabalho, um orçamento dos pontos de interesse não foi possível.

Diante disso, optou-se por realizar uma pesquisa baseada em reportagens sobre o assunto. Com os resultados obtidos, pode-se observar que o projeto de automação residencial utilizando o Arduino é viável quando comparados os custos.

Conforme reportagem do jornal paranaense Gazeta do Povo (2013), embora os custos da automação residencial estejam mais acessíveis devido a sua popularização, um pacote básico de automação incluindo iluminação, *home theater* e ar condicionado custa R$ 3.600,00 em determinada empresa, consultada inclusive pela equipe deste trabalho. Caso o usuário procure mais serviços, a automação de uma sala de *home* cinema, em outra empresa, tem o custo de R$ 10.000,00. Vale ressaltar que os valores podem ser alterados conforme os itens nos quais se queiram automatizar, sendo esta também uma justificativa das empresas em exigirem o projeto elétrico.

Ainda segundo a revista *Home Theater* e Casa Digital (2013), um sistema similar, pode ser encontrado a partir de R$ 4.000,00. Tal reportagem ainda afirma que os custos tendem a ser menores em projetos sem fio de pequenas automações.

Já a reportagem do jornal G1 (2013) informa que projetos de automação residencial de ambientes podem variar entre R$ 10.000,00 e R$ 30.000,00. Tal reportagem mostra que determinado projeto incluindo iluminação, ar condicionado e projetor custou R$ 26.000,00 ao cliente.

De acordo com o Tech Tudo (2014), encontram-se valores de sistemas similares a partir de R$ 10.000,00, por determinada empresa. Outra empresa do ramo, que também foi consultada pela equipe deste trabalho informa valores a partir de R$ 5.000,00 em um sistema que permite controlar a intensidade da luz, temperatura do ar condicionado, persianas ou assistir um filme através do relógio

Galaxy Gear da *Samsung.* O *site* ainda faz menção da automação residencial através do Arduino como opção para automação de baixo custo.

Diante do exposto, é possível verificar que o Arduino pode ser utilizado como agente de redução de custos no que se pretende automatizar.

3 O ARDUINO E OS MICROCONTROLADORES

3.1 O ARDUINO

O Arduino é uma plataforma eletrônica de computação física aberta, com uma base em placa simples de entradas/saídas (*input/output* - I/O), assim como em um ambiente de desenvolvimento que implementa esta linguagem *Processing*. O Arduino pode ser utilizado para desenvolver objetos interativos independentes, ou conectado a *softwares* de seu computador (BANZI, 2012).

O Arduino é uma pequena placa de microcontrolador contendo um plugue de conexão USB que permite a ligação com um computador. Além disso, contém diversos outros terminais que permitem a conexão com dispositivos externos, como motores, relés, sensores luminosos, diodos a laser, alto-falantes e outros. Os Arduinos podem ser energizados por um computador através do plugue USB, por uma bateria de 9 V ou por uma fonte de alimentação. Eles podem ser controlados diretamente pelo computador, ou então podem ser programados pelo computador e, em seguida, desconectados, permitindo assim trabalharem independentemente do computador. O projeto da placa é aberto. Isso significa que qualquer um pode construir placas compatíveis com o Arduino. Essa competição resultou em placas de baixo custo (MONK, 2013).

A filosofia do Arduino concentra-se em desenvolver projetos, e não em falar sobre eles. Ela representa uma busca constante por meios mais rápidos e poderosos para o desenvolvimento de protótipos. Citando Banzi (2012): “Exploramos muitas técnicas, de prototipagem e desenvolvemos formas de pensar cada vez mais práticas”.

3.1.1 Breve História do Arduino

O Arduino teve seu início no Interaction Design Institute na cidade de Ivrea, na Itália, em 2005. O professor Massimo Banzi procurava um meio barato de tornar

mais fácil para os estudantes de *design* trabalhar com tecnologia. Ele discutiu seu problema com David Cuartielles, um pesquisador visitante da Universidade de Malmo, na Suécia, que estava procurando uma solução semelhante, resultando na concepção e projeto da plataforma Arduino. Os produtos existentes no mercado eram caros e relativamente difíceis de usar. Banzi e Cuartielles decidiram desenvolver um microcontrolador que poderia ser utilizado pelos seus estudantes de arte e *design* em seus projetos. As principais exigências eram que fosse barato – o preço almejado não poderia ser mais do que o que um estudante gastaria se saísse para comer uma pizza – e que fosse uma plataforma que qualquer pessoa pudesse utilizar. David Cuartielles desenhou a placa, e um aluno de Massimo, David Mellis, programou o *software* para executar a placa. Massimo contratou um engenheiro local, Gianluca Martino, que também trabalhou no Design Institute ajudando alunos com seus projetos. Gianluca concordou em produzir uma tiragem inicial de duzentas placas (EVANS; NOBLE; HOCHENBAUM, 2013).

A nova placa foi chamada de Arduino em referência a um bar local frequentado por membros do corpo docente e alunos do instituto. As placas eram vendidas em forma de *kit* para que os alunos fizessem seus próprios projetos. A tiragem inicial foi rapidamente vendida, e mais unidades foram produzidas para manter a demanda. *Designers* e artistas de outras áreas ouviram falar do Arduino e quiserem usá-lo em seus projetos. Sua popularidade cresceu rapidamente quando o grande público percebeu que o Arduino era um sistema de fácil utilização, de baixo custo e que poderia ser usado em seus próprios projetos, bem como era uma excelente introdução para programação de microcontroladores. O projeto original foi melhorado e novas versões foram introduzidas. As vendas dos Arduino oficiais alcançaram agora a marca de 300 mil unidades, e eles são vendidos em todo o mundo por intermédio de uma série de distribuidores (EVANS; NOBLE; HOCHENBAUM, 2013).

3.1.2 O Arduino e seus Componentes

A computação física significa a construção de sistemas interativos físicos mediante o uso de *software* e *hardware* que integrados podem sentir e responder ao

mundo analógico. Na prática, frequentemente este termo descreve desenhos de projetos ou objetos que utilizam sensores e microcontroladores para traduzir entradas analógicas a sistemas baseados em *software*, ou controlar dispositivos como motores, servos, iluminação ou outro *hardware*. *Open Source Hardware* consiste em dispositivos físicos de tecnologia concebidos e oferecidos pelo movimento de projeto aberto. Tanto o *software* como o *open source hardware* são criados sob o movimento de cultura *open source* e aplica este conceito a uma variedade de componentes. O projeto do *hardware* (ou seja, desenhos mecânicos, esquemas, lista de materiais, dados de *layout*, código fonte e dados do *layout* de circuitos integrados), além do *software* livre que aciona o *hardware*, estão todos liberados com a abordagem livre (MULTILOGICA, 2014).

Trata-se de um ambiente multiplataforma; ele pode ser executado no Windows, Macintosh e Linux. Tem por base o IDE (*Integrated Development Environment*) de programação *Processing*, ambiente de desenvolvimento fácil de ser utilizado e que costuma ser empregado por artistas e *designers*. Pode ser programado utilizando-se uma interface serial USB, sem a necessidade de uma porta serial DB9 ou DB25. Este recurso é útil, uma vez que muitos computadores modernos não têm portas seriais. É um *hardware* e *software* de fonte aberta – cada usuário pode fazer o *download* do diagrama de circuito, comprar todos os componentes e criar seu próprio Arduino, sem ter que pagar nada aos criadores originais. O *hardware* é barato. A placa USB custa cerda de 20 euros (atualmente, algo em torno de 35 dólares) e substituir um *chip* queimado é muito fácil, além de não custar mais do que 4 dólares. Dessa forma, o Arduino Project foi desenvolvido em um ambiente educacional; portanto, é ideal para iniciantes que desejam resultados rápidos (BANZI, 2012).

> A filosofia do Arduino concentra-se em desenvolver projetos, e não em falar sobre eles. Ela representa uma busca constante por meios mais rápidos e poderosos de criarmos protótipos. Exploramos muitas técnicas, de prototipagem e desenvolvemos formas de pensar cada vez mais práticas (BANZI, 2012).

É útil conhecer um pouco das diversas placas de Arduino. Como dispositivo padrão apresenta-se a placa Uno. Segundo Monk (2013), essa placa é a mais usada atualmente, mas todas são programadas com a mesma linguagem e a maioria usa

as mesmas conexões com o mundo exterior, de modo que se pode utilizar facilmente uma placa diferente.

O Arduino Uno (Figura 4) é a última geração de uma serie mais popular de placas Arduino. A série inclui o Diecimila (10.000 em italiano) e o Duemilanove (2009 em italiano). Essas placas mais antigas são muito semelhantes ao Arduino Uno. Todas têm os mesmos conectores e um soquete USB, sendo geralmente compatíveis entre si. A diferença mais significativa entre o Uno e as placas anteriores é que o Uno usa um *chip* USB diferente. Isso não afeta o modo de usar a placa, mas facilita a instalação do *software* e permite velocidades de comunicação mais elevadas com o computador. Com sua fonte de alimentação de 3,3 V, o Uno também pode fornecer uma corrente maior e sempre vem equipado com o ATmega328, com uma memória maior que as versões anteriores (MONK, 2013).

Figura 4 – Arduino Uno
Fonte: Arduino (2014).

O Arduino Nano (Figura 5) é a melhor escolha para projetos restrição de dimensões físicas. Projetado e produzido pela Gravitech, a versão 3.0 do Nano (com processador ATmega328) tem uma mini-USB integrada, um formato compacto para uso em placas testes. O Nano possui funcionalidades similares à do Duemilanove, mas tem dois pinos adicionais de entrada analógica. A alimentação para a placa é

fornecida por USB ou por dois pinos separados: o pino 30 pode receber uma tensão desregulada entre 6 V e 20 V, ou o pino 27 pode receber uma tensão regulada de 5,5 V. A placa seleciona a tensão que for mais alta. O tamanho reduzido da placa faz com que seja ideal para projetos com espaço limitado (EVANS; NOBLE; HOCHENBAUM, 2013).

Figura 5 – Arduino Nano
Fonte: Arduino (2014).

O Arduino Lilypad (Figura 6), desenvolvida pela SparkFun Electronics e pela Leah Buechley, é ótimo para projetos têxteis e em esteiras industriais.

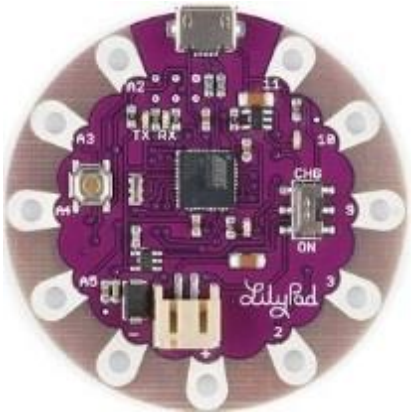


Figura 6 – Arduino Lilypad
Fonte: Arduino (2014).

Arduino Mega (Figura 7) é o irmão mais velho da família Arduino, o Mega, e usa um microprocessador com maior número de pinos. O ATmega1280, ou Mega, foi atualizado ao mesmo tempo que o Uno, e o microprocessador usado é o ATmega2560. A nova versão possui memória Flash de 256 KB, superior aos 128 KB

do original. O Mega fornece um aumento significativo na funcionalidade de entrada-saída em relação ao Arduino padrão; portanto, com o aumento da memória, ele é ideal para aqueles projetos maiores que controlam grandes quantidades de LEDs, possuem um grande número de entradas e saídas ou necessitam de mais de uma porta serial de *hardware* – o Arduino Mega possui quatro. As placas possuem 54 pinos digitais de entrada/saída, 14 dos quais podem fornecer saída analógica PWM, e 16 pinos de entrada analógica. A comunicação é feita com até quatro portas seriais de *hardware*. A comunicação SPI (*Serial Peripheral Interface*) e o suporte para dispositivos I2C/TWI (*Two Wire Interface*) estão também disponíveis. A placa também inclui um conector ICSP e um botão de *reset*. O ATmega8U2 substitui o *chipset* FTDI usado pelo seu antecessor e processa a comunicação serial USB (EVANS; NOBLE; HOCHENBAUM, 2013).

Figura 7 – Arduino Mega
Fonte: Arduino (2014).

As placas básicas são completadas por placas acessórios, denominadas *Shields*, que podem ser encaixadas por cima da placa do Arduino. Neste projeto serão utilizados 3 *shields*, que permitirão fazer o Arduino funcionar como um pequeno servidor Web para armazenar e trocar informações com outras máquinas (MONK, 2013).

- Modulo Ethernet: o Arduino Ethernet *Shield* (Figura 8) permite que uma placa Arduino se conecte à *Internet*. Segundo Evans et al. (2013), este *shield* oficial Arduino Ethernet, é baseado no WIZnet W500 com sua pilha TCP/IP completa. Possui um conector RJ45 integrado para uma conexão Ethernet e um leitor de cartão micro SD integrado. O Arduino Ethernet não possui um *chip* controlador USB para serial integrada, mas possui um conector de seis pinos que pode ser conectado a um cabo FTDI ou uma porta serial USB para fornecer um *link* de comunicação para que a placa possa ser programada. É ideal para uso em monitoramento remoto de estações de registros de dados com leitor de cartão micro SD integrados e uma conexão com uma rede Ethernet com fio para alimentação.

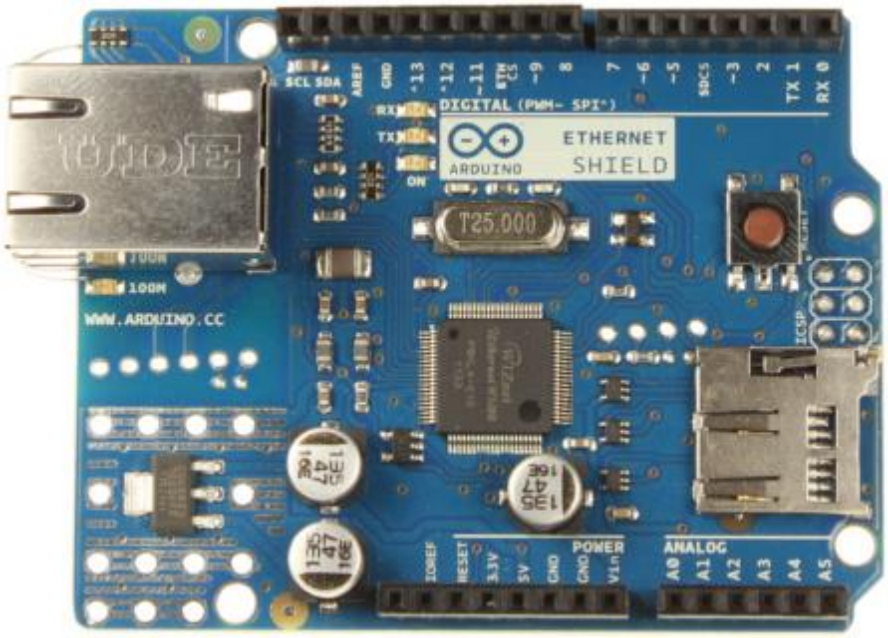


Figura 8 – Arduino Ethernet
Fonte: Arduino (2014).

- Módulo de relés com oito canais de 5 V para automação com optoacoplador (Figura 9): possui interface padrão que pode ser controlado diretamente por microcontrolador, onde o relé suporta correntes de 10 A, com indicação de LED de saída do relé.

Figura 9 – Modulo de relés com oito canais
Fonte: Techmount (2013).

- Módulo *Dimmer* AC (Figura 10): o *Shield* pode acionar de forma proporcional motores e/ou lâmpadas, que operem com sinais de entrada contínuo com nível de 5 V e saída alternada de 110 V ou 220 V.

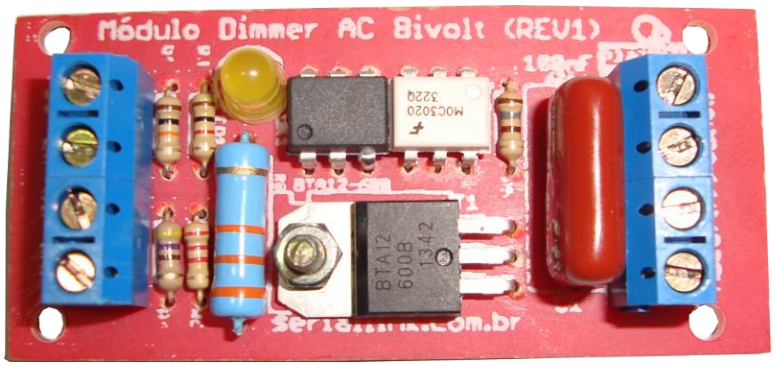


Figura 10 – Módulo *Dimmer* AC
Fonte: Seriallink (2014).

3.1.3 Métodos de Programação para Arduino

A programação do Arduino é desenvolvida com a utilização de um *software* de desenvolvimento, bem como sua linguagem, estrutura e símbolos especiais baseados na linguagem C/C++, todos descritos neste capítulo.

3.1.3.1 *Software* para Arduino

O *Integrated Development Environment* (IDE) é um programa especial executado em seu computador que permite a criação de *sketches* para a placa Arduino em uma linguagem simples, modelada a partir da linguagem *Processing*. No momento do *upload* do *sketch* para a placa, o código escrito é traduzido para a linguagem C (geralmente difícil de ser utilizada por iniciantes) e é transmitido para o compilador avr-gcc, importante *software* de código aberto que realiza a tradução final de seus comandos, agora para uma linguagem que pode ser compreendida pelo microcontrolador. Assim o Arduino simplifica ao máximo as complexidades inerentes à programação de microcontroladores (BANZI, 2012).

O IDE do Arduino fornece todas as funcionalidades necessárias para programação, incluindo vários exemplos de programas ou *sketches* que demonstram a conexão com o microcontrolador e comunicação com alguns dispositivos comuns, tais como LEDs, LCDs e sensores. Assim como o *hardware*, o *software* para o Arduino é de código aberto e pode ser baixado gratuitamente. As versões do IDE estão disponíveis para os sistemas operacionais Windows, Mac OS X e Linux. (EVANS; NOBLE; HOCHENBAUM, 2013).

3.1.3.2 A linguagem do Arduino

A linguagem do Arduino é baseada em C/C++. A linguagem do Arduino foi projetada para tornar a codificação o mais simples possível, permitindo-lhe concentrar-se no que pode ser feito, em vez de como é feito. O Arduino faz uso extensivo de bibliotecas para fornecer funções comuns. As bibliotecas ajudam a linguagem a ocultar muito de sua complexidade e simplificar muitas tarefas comuns, tais como definir os pinos digitais como entrada ou saída, ler valores analógicos, controlar servomotores ou ligar e desligar motores DC. Os *Shields* que adicionam funcionalidades exigem também muitas vezes bibliotecas especializadas, e essas são normalmente fornecidas e mantidas pelos desenvolvedores do módulo (EVANS; NOBLE; HOCHENBAUM, 2013).

3.1.3.3 Estrutura do Arduino

Um *sketch* do Arduino é executado em duas partes:

- *void setup ()* = local em que coloca o código de inicialização, as instruções que preparam a placa antes do início do loop principal do *sketch;*
- *void loop ()* = seção que contém o código principal de seu *sketch* e que deve apresentar um conjunto de instruções a serem repedidas seguidas vezes até que a placa seja desligada (BANZI, 2012).

3.1.3.4 Símbolos especiais

O Arduino inclui alguns símbolos que devem ser empregados em linhas de código, comentários e blocos de código:

- ; (ponto e vírgula) = toda instrução (linha de código) é encerrada por um ponto e vírgula. Essa sintaxe permite que formate o seu código livremente. Podendo até colocar duas instruções na mesma linha, desde que as separe pelo ponto e vírgula, mas isso pode dificultar a leitura de seu código;
- {} (chaves) = essa notação é utilizada para marcar blocos de código. Por exemplo, quando se escreve o código para a função *loop()*, tem que utilizar chaves antes e depois do código (BANZI, 2012).

3.1.3.5 Comentários

Comentários são as porções do texto ignoradas pelo processador do Arduino, mas extremamente úteis para que se explique essa funcionalidade a outras pessoas. Há dois tipos de comentários no Arduino:

- // de linha única: o texto até o fim da linha será ignorado
- /* de várias linhas: pode escrever tudo que seja necessário */

3.1.3.6 Constantes

O Arduino inclui um conjunto de palavras-chave predefinidas com valores especiais. *HIGH* e *LOW* são utilizadas, por exemplo, quando se deseja ligar ou desligar um pino do Arduino. *INPUT* e *OUTPUT* são utilizadas para definir se um pino específico como entrada ou saída. *True* e *False* indicam se uma condição ou expressão é verdadeira ou falsa, respectivamente (BANZI, 2012).

3.1.3.7 Variáveis

Variáveis são áreas nomeadas da memória do Arduino nas quais se pode armazenar dados a serem utilizados e manipulados em seu *sketch*. Como sugere o nome, variáveis podem ser modificadas em qualquer momento. Como o Arduino é um processador muito simples, quando se declara uma variável, deve também especificar seu tipo, ou seja: dizer ao processador o tamanho do valor que se deseja armazenar. Na Tabela 1 são apresentadas resumidamente os principais tipos de variáveis utilizadas nos projetos com Arduino:

Tabela 1 – Tipos de variáveis para desenvolvimento com Arduino (BANZI, 2012).

Variável	**Descrição**
Boolean	Pode ter um de dois valores: verdadeiro ou falso.
char	Armazena um único caractere, como A. Da mesma forma que qualquer computador, o Arduino armazena esse valor como um número, ainda que se veja um texto. Quando caracteres são utilizados para armazenar números, eles podem armazenar valores de -128 a 127.
byte	Armazena um valore entre 0 e 255. Assim como no caso de char, *bytes* utilizam apenas um *byte* de memória.
int	Utiliza dois *bytes* da memória para representar um número entre -32.768 e 32.767; trata-se do tipo de dado mais comum utilizado no Arduino.
unsigned int	Assim como *int*, utiliza 2 *bytes*, mas o prefixo *unsigned* significa que não pode armazenar números negativos. Seu alcance, portanto, é de 0 a 65.535.

Tabela 1 (Continuação) - Tipos de variáveis para desenvolvimento com Arduino (BANZI, 2012).

long	Duas vezes o tamanho de um int e armazena números de -2.147.483.648 a 2.146.483.647
unsigned long	Versão não assinalada de *long*; vai de 0 a 4.294.967.295
float	Tipo de dado de tamanho considerável, capaz de armazenar valores de ponto flutuante, ou seja: números com ponto decimal. Um *float* consumira 4 *bytes* de seus preciosos recursos de RAM. As funções que podem utilizá-lo também consomem muita memória, por isso utiliza-se *floats* apenas quando estritamente necessário.
double	Número ponto flutuante de precisão dupla, com valor máximo de $1,7976931348623157 \times 10^{308}$.
string	Conjunto de caracteres ASCII utilizados para armazenar informações textuais (pode utilizar uma *string* quando quiser enviar uma mensagem por meio de uma porta serial, ou mostrá-la em um monitor LCD). Quanto ao armazenamento, uma *string* utiliza um *byte* para cada caractere, mais um caractere nulo para avisar ao Arduino que o fim da linha foi atingido.
array	Lista de variáveis que podem ser acessadas por meio de um índice. Um *array* é utilizado para criar tabelas de valores que podem ser acessados com facilidade. Por exemplo, caso queira armazenar níveis diferentes de brilho a serem utilizados por um LED, pode criar seis variáveis, nomeadas *light01*, *light02* e assim por diante. A palavra "*array*" não chega a ser utilizada na declaração de variável: os símbolos [] e {} são suficientes.

3.1.3.8 Estruturas de controle

O Arduino inclui palavras-chave para controlar o fluxo lógico de seu *sketch*. As principais palavras-chave e estruturas de programação são apresentadas na Tabela 2:

Tabela 2 – Palavras-chave e estruturas de programação do Arduino (BANZI, 2012).

Estrutura	**Descrição**
if (condição) *else*	Essa estrutura é responsável pela tomada de decisões no programa. *If* deve ser seguido por uma questão, especificada como uma expressão entre parênteses. Se a expressão for verdadeira, o que vier depois dela será executado. Se falsa, o bloco de código que segue *else* será executado.

Tabela 2 (Continuação) - Palavras-chave e estruturas de programação do Arduino (BANZI, 2012).

switch(variável) case	Enquanto a instrução *if* funciona como uma bifurcação que oferece duas opções ao seu programa, *switch case* se parece mais com uma enorme rotatória. Ela permite que seu programa receba várias orientações, dependendo do valor de uma variável, sendo ótima para manter seu código organizado, uma vez que substitui listas de instruções *if*.
while	Semelhante a *if*, *while* executará um bloco de códigos enquanto determinada condição verdadeira.
do while (condição)	Idêntica à *while*. A única diferença é que, aqui, o código será executado antes de a condição ser avaliada. Essa estrutura é utilizada quando se deseja que seu bloco de código seja executado ao menos uma vez antes de conferir a condição.
break	Esse termo permite que saia de um loop e continue a execução do código que aparece depois dele, *break* também é utilizado para separar seções distintas de uma instrução *switch case*.
continue	Quando utilizado dentro de um laço (*loop*), continue permite que se pule o resto do código e faça com a condição seja testada novamente.
return	Interrompe a execução de uma função, retornando-a. Também pode se utilizar esse comando para retornar um valor de dentro da função.

3.1.3.9 Aritmética e fórmulas

As operações aritméticas, bem como as fórmulas são efetuadas com os operadores aritméticos e são descritos na Tabela 3:

Tabela 3 – Operadores Aritméticos (BANZI, 2012).

Símbolo	**Operação**
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Resto inteiro da divisão

É possível ser utilizado quantos níveis de parênteses forem necessários para agrupar as expressões. Ao contrário dos colchetes e chaves que são

reservados a outros propósitos, índices e blocos de *arrays*, respectivamente (BANZI, 2012).

3.1.3.10 Operadores de comparação

Os operadores relacionais, também utilizados na construção dos *sketchs* do Arduino, estão indicados na Tabela 4:

Tabela 4 – Operadores Relacionais (BANZI, 2012).

Símbolo	**Descrição**
==	Igual a
!=	Não igual a
<	Menor que
>	Maior que
<=	Menor ou igual
>=	Maior ou igual
++	Incremento
--	Decremento

- Operadores booleanos: são utilizados quando se quer combinar várias condições. Há três operadores: e, representado por *&&*; ou, representado por ||; por fim, não, representado por ! (símbolo de exclamação);
- Operadores compostos: são operadores especiais utilizados para tornar o código mais conciso para realizar operações comuns, como incrementar o valor de uma variável;
- Incremento (++) e decremento (--): incrementam e decrementam determinado valor em uma unidade. Ao escrever *i*++ incrementa-se o valor de *i* em uma unidade e avalia um resultado equivalente a *i+1*; ++*i*, por sua vez, avalia o valor de *i* e só então faz o incremento (BANZI, 2012).

3.1.3.11 Funções de entrada e saída

O Arduino ainda inclui funções para a manipulação de entradas e saídas, conforme Tabela 5:

Tabela 5 – Funções para a manipulação de entradas e saídas (BANZI, 2012).

Função	**Descrição**
pinMode(pino,modo)	Figura um pino digital para que ele se comporte como a entrada ou saída.
digitalWrite(pino,valor)	Liga ou desliga um pino digital. Pinos devem ser marcados explicitamente como saída, utilizando *pinMode*, antes que *digitalWrite* possa ter qualquer efeito.
int digitalRead(pino)	Lê o estado de um pino de entrada, retornando *HIGH* se o pino recebe voltagem, ou *LOW* se não há nível lógico '1' aplicado.
int analogRead(pino)	Lê a tensão aplicada a um pino de entrada analógica e retorna um número entre 0 a 1023, representado tensões entre 0 e 5 V.
analogWrite(pino, valor)	Altera a taxa PWM em um dos pinos marcados PWM, *pin* pode ser 11,10,9,6,5 e 3, *value* pode ser um número entre 0 e 255, representado a escala entre 0V e 5V da tensão de saída.
shiftOut(dataPin,clockPin,bitOrder,valor)	Envia dados para um registrador de deslocamento, dispositivo utilizado para expandir o numero de saídas digitais. Esse protocolo utiliza um pino para dados e um para o *clock*. *bitOrder* indica a ordem dos *bytes* (do menos significativo para o mais significativo) e *valor* é o *byte* em si que será envidado.
unsigned long pulseIn(pino,valor)	Mede a duração de um pulso vindo de uma das entradas digitais. Isso é útil para ler sensores infravermelhos ou acelerômetros que emitem seus valores como pulsos de duração diferente.

3.1.3.12 Funções de tempo

As funções responsáveis pela medição de tempo transcorrido, além da função responsável por pausar o *sketch* são descritas na Tabela 6:

Tabela 6 – Funções capazes de medir o tempo transcorrido e também pausar o *sketch* (BANZI, 2012).

Função	**Descrição**
unisigned long millis()	Retorna o número de milissegundos (ms) transcorridos desde que o *sketch* teve início.
delay(ms)	Pausa o programa pelo tempo especificado em milissegundos.
delayMicroseconds(us)	Pausa o programa pelo tempo especificado em microssegundos.

3.1.3.13 Funções matemáticas

As funções matemáticas e trigonométricas estão representadas na Tabela 7:

Tabela 7 – Funções matemáticas e trigonométricas (BANZI, 2012).

Função	**Descrição**
min(x,y)	Retorna o menor *x* e *y.*
max(x,y)	Retorna o maior *x* e *y.*
abs(x)	Retorna o valor absoluto de *x*, o que transforma números negativos em positivos. Por exemplo, se *x* for 5, ela retornará 5; se x for -5, ela ainda retornará 5.
constrain(x,a,b)	Retorna o valor de *x*, entre *a* e *b*. Se *x* for menor que *a*, ela retorna *a*. Se *x* for maior que *b*, ela retorna *b*.
map(valor,fromLow,fromHig,toLow,toHigh)	Mapeia um valor no intervalo de *fromLow* a *maxLow*, a um intervalo entre *toLow* e *toHigh*. Esse recurso é muito útil para processar valores de sensores analógicos.
doublepow(base expoente)	Retorna o resultado da potencialização de um número (base) por um valor específico (expoente).
double sqrt(x)	Retorna a raiz quadrada de um número.

Tabela 7 (Continuação) - Funções matemáticas e trigonométricas (BANZI, 2012).

double sin(rad)	Retorna o seno de um ângulo especificado em radianos.
double cos(rad)	Retorna o cosseno de um ângulo especificado em radianos.
double tan(rad)	Retorna a tangente de um ângulo especificado em radianos.

3.1.3.14 Funções de números aleatórios

Na Tabela 8, estão representadas as funções de números aleatórios:

Tabela 8 – Funções de números aleatórios (BANZI, 2012).

Função	**Descrição**
RandomSeed (semente)	Redefine o gerador de números pseudoaleatórios do Arduino. Ainda que a distribuições dos números retornados por *random()* seja essencialmente aleatória, a sequência que ela oferece é previsível. Por isso, deve-se redefinir o gerador utilizando um valor aleatório. Caso se tenha um pino analógico conectado, este deve captar ruídos variados do ambiente (como ondas rádio, interferência eletromagnética de celulares, luzes fluorescentes e assim por diante) que podem ser utilizados nesse caso.
long random(max) e long random (min,max)	Retorna um valor de inteiro *long* pseudoaleatório, entre *min* e *max* -1. Se o valor mínimo não for especificado, seu limite será 0.

3.2 OS MICROCONTROLADORES

Devido a sua enorme versatilidade de *Hardware* e *Software*, o Microcontrolador, também chamado de “microcomputador-de-um-só-*chip*”, facilita de sobremaneira o projeto de sistemas eletrônicos digitais e analógicos, pois reúne num só *chip* vários elementos que nos sistemas baseados em microprocessadores, tinham suas funções realizadas por *chips* independentes, como RAM, ROM,

temporizadores, contadores de eventos, canal de comunicação serial, portas de I/O, etc. (VIDAL, 1999; MORAES, 2007).

Em meados da década de 80, surgiram os primeiros MCUs (*Micro Controler Unit* ou Unidade de Microcontrolador). Os microcontroladores, comparados aos microprocessadores, são dispositivos mais simples, com memórias RAM e ROM internas, oscilador interno de *clock*, I/O interno, entre outros, sendo por isso chamados muitas vezes de computadores em único *chip*. Tais características tornam mais simples o projeto de dispositivos inteligentes, pois os MCUs raramente necessitam de CIs (circuitos integrados) externos para funcionar, contribuindo para diminuição de custos e tamanhos (PEREIRA, 2005).

Um microcontrolador é um circuito integrado digital, construído em um único *chip*, possuindo uma única CPU e vários periféricos embutidos no mesmo CI, tais como: memórias Flash e RAM, módulos UART, temporizador, PWM, conversor analógico-digital (ADC), relógio de tempo real (RTC), entre outros (MARSCZAOKOSKI et al., 2013).

Geralmente utilizado no controle de periféricos como *displays*, relés e sensores, têm suas ações lógicas executadas por um programa interno através de uma unidade lógica aritmética (ULA), na qual as operações são executadas.

A popularização dos microcontroladores se dá devido ao seu baixo custo e baixo consumo de energia, sendo utilizados como soluções para projetos nos quais se buscam um melhor custo benefício. Os microcontroladores são diferenciados pela capacidade de processamento e de armazenamento formando famílias de processadores cujas funções são similares. Com famílias que executam poucas funções, usados para operações mais simples que não necessitem de muitos recursos e outras mais complexas com maior capacidade de armazenamento de dados, e outras funcionalidades, os sistemas microcontrolados podem atuar nas mais diversas áreas, incluindo a automação predial, objetivo deste trabalho (MARSCZAOKOSKI et al., 2013).

Os mais diversos tipos de microcontroladores existentes no mercado são diferenciados pela quantidade de memória interna de armazenamento de dados e instruções de programas, velocidade de processamento, quantidade de portas configuráveis de entrada e saída, bem como quantidades e tipos de periféricos acoplados, a arquitetura do microcontrolador (*Harvard ou Von-Neumann*) e

finalmente o número de instruções existentes internamente ao microprocessador (MARTINS, 2005).

Como exemplo, a Figura 11 ilustra um diagrama de blocos de um microcontrolador genérico:

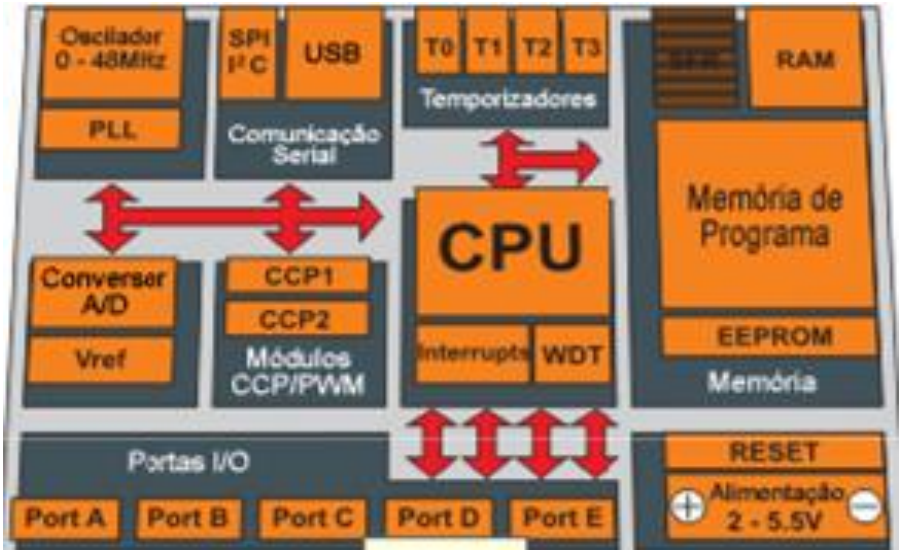


Figura 11 – Diagrama em blocos de um microcontrolador genérico
Fonte: Adaptado de Bastos (2014).

4 REDES, LINGUAGENS DE PROGRAMAÇÃO E SERVIDORES WEB

4.1 REDES

De acordo com Mendes (2007), no final da década de 1960 a ARPA (Agência de Projetos de Pesquisas Avançadas do Departamento de Defesa dos Estados Unidas da América) que posteriormente foi chamada de DARPA começou a consolidar uma rede experimental de computadores de longa distância que foi chamada de ARPANET. Seu original objetivo era permitir aos fornecedores do governo compartilhar os escassos e caros recursos computacionais. No início a ARPANET permitia que as universidades UCLA (Universidade da Califórnia – Los Angeles, EUA), Universidade de Utah, UCSB (Universidade da Califórnia – Santa Bárbara, EUA) e o SRI (Stanford Research Institute – Stanford, EUA) pudessem trocar informações entre si. Os usuários desta rede usavam-na desde o início para trocas de arquivos e programas, mensagens por correio eletrônico, além de desenvolvimento conjunto de pesquisas usando os computadores compartilhados remotamente.

A partir de 1970, com a chegada dos minicomputadores de 32 *bits*, os grandes fabricantes como IBM, HP e Digital, iniciaram o planejamento de soluções com o objetivo de distribuir o poder de processamento dos mainframes e facilitar o acesso às informações. Em 1977 a Digital lançou o VAX com o objetivo de criar uma arquitetura de rede de computadores. Com isso, ela esperava levar vantagem ante sua rival Big Blue (IDEPAC, 2014).

Quando um VAX era iniciado, ele imediatamente começava a procurar por outras maquinas para se comunicar, um procedimento um tanto ousado numa época em que poucas pessoas tinham ideia do que era uma rede. O método teve êxito e o VAX alcançou grande popularidade, principalmente em aplicações cientificas e de engenharia. Anos mais tarde, a Digital acabaria sendo comprada pela Compaq, que por sua vez, foi incorporada a HP. Mas as inovações surgidas com o VAX tiveram grande influência para as tecnologias que viriam no futuro.

A partir de 1995, segundo Morimoto (2006), com a abertura do acesso à *Internet*, tudo ganhou uma dimensão nova e a função principal da maioria das redes

tornou-se simplesmente compartilhar a conexão com a Web. Depois veio uma segunda mudança, que é o uso da Web não apenas com o objetivo da comunicação, mas como uma forma de rodar aplicativos. Inicialmente surgiram os *webmails*, depois os clientes de bate papo, tais como MSN e ICQ, e agora há também processadores de texto, planilhas e demais aplicativos que rodam diretamente na rede.

4.1.1 Classificação das Redes

De acordo com Tanembaum (2003), as redes de computadores podem ser classificadas pelo seu tamanho, topologia, meio físico e protocolo utilizado.

Os principais tipos de redes levando-se em conta a classificação por seu tamanho têm as seguintes denominações:

- **PAN** (*Personal Area Network*) – São redes de área pessoal, normalmente usadas para comunicação entre dispositivos dentro de uma distância muito pequena, seu máximo alcance não passa de 10 metros. Alguns exemplos de redes PAN são as redes Bluetooth, ZigBee e Ultrawideband – UWB (TELECO, 2014);
- **LAN** (*Local Area Network*) – São redes de área local, normalmente utilizadas em um único edifício ou campus universitário com no máximo alguns quilômetros de extensão. São muito usadas para conectar estações de trabalho de empresas e indústrias com o objetivo de compartilhar recursos e trocar informações, tais como compartilhamento de arquivos e impressoras (TANEMBAUM, 2003);
- **MAN** (*Metropolitan Area Network*) – São redes de área metropolitana, que tem abrangência de uma cidade inteira. Elas interligam várias LANs através de nós para que elas possam se comunicar da mesma maneira de como se estivessem na mesma LAN. Exemplos de MANs são a interligação de várias redes LAN de uma empresa em diferentes pontos, redes de TV a cabo e *Internet* via rádio (CARMONA, 2007);
- **WAN** (*Wide Area Network*) – São redes de longa distância, que integram diversas localizações geográficas abrangendo diversos países e continentes.

Uma WAN engloba todas as diversas redes de menor alcance geográfico. Um exemplo de uma WAN é a *Internet* (KUROSE & ROSS, 2010).

4.2 LINGUAGENS DE PROGRAMAÇÃO

Linguagens de programação são métodos padronizados para comunicar instruções para um computador. São conjuntos de regras sintáticas e semânticas usadas para definir um *software* de computador. Elas permitem que um programador seja capaz de especificar precisamente sobre quais dados um computador atuará e como eles serão armazenados ou transmitidos, bem como quais ações devem ser realizadas sob diversas circunstâncias (DERSHEM & JIPPING, 1995).

Segundo Sebesta (2003), entre 1936 e 1945 o cientista alemão Konrad Zuse construiu uma série de computadores complexos e sofisticados utilizando relés eletromecânicos. Porém, em 1945 a guerra destruiu quase todos, restando apenas um de seus últimos modelos, chamado de Z4. Trabalhando sozinho Zuse desenvolveu uma das primeiras linguagens de programação com o objetivo de utilizá-la no seu computador. Esta linguagem foi chamada de *Plankalkül,* e era notavelmente completa com avançados recursos de estrutura de dados, incluía até mesmo matrizes e registros. Com o passar dos anos os computadores começaram a ficar mais disponíveis, porém eram dotados de memórias extremamente pequenas e muito difíceis de programar, principalmente pela falta de *software* de apoio.

O desenvolvimento das linguagens de programação foi evoluindo ao longo do tempo e várias outras foram surgindo. A maioria delas foi surgindo a partir do aprimoramento baseado no conhecimento de linguagens anteriores. Porém algumas tiveram um pioneirismo ao introduzir novos paradigmas como as linguagens de alto nível FORTRAN e LISP. Diversas outras linguagens surgiram ao longo do tempo, das quais pode-se destacar as linguagens ALGOL-60, PASCAL, C, C++, EIFFEL, CEDAR/MESA, JAVA, dentre muitas outras (FILHO, 2007).

4.2.1 Compiladores

A única maneira de se comunicar com um computador é através de um programa e a única linguagem que o computador entende é a linguagem de máquina. Deste modo para que um programa se comunique com o computador este deve estar em linguagem de máquina. Para que um programa esteja em linguagem de máquina, que na prática são “zeros” e “uns”, eles precisam de outros programas que façam esta tradução, os quais são chamados de compiladores. Um compilador lê a primeira instrução de um programa, faz uma análise de sua sintaxe, e se constatar que não há nenhum problema no código, converte-a para a linguagem de máquina e segue para a próxima instrução repetindo este processo até que todas as linhas de código sejam atingidas com êxito ou até que seja encontrado algum erro (MIZRAHI, 2008).

Após a etapa de compilação, ou seja, a conversão de um código em linguagem de máquina, o compilador cria um arquivo com as instruções já traduzidas de acordo com a linguagem de programação utilizada com as bibliotecas necessárias e após gera um arquivo executável no qual pode ser executado diretamente do sistema operacional. Porém esta etapa só será realizada se o programa estiver absolutamente livre de erros (adaptado de MIZRAHI, 2008).

4.2.2 Linguagem C

A linguagem C foi criada e implementada inicialmente por Dennis Ritchie em um DEC PDP-11 que utilizava o sistema operacional UNIX. C é o resultado do desenvolvimento de uma linguagem chamada BCPL, ainda em uso na sua forma original, na Europa. A linguagem BCPL foi desenvolvida por Martin Richards e acabou gerando uma linguagem chamada B, criada por Ken Thompson, que posteriormente na década de 1970 levou ao desenvolvimento da linguagem C. Com a popularidade dos microcomputadores uma grande quantidade de implementações em C foram criadas. Incrivelmente, os códigos-fontes criados por essas implementações eram altamente compatíveis, podendo ser compilados por outros

computadores. Mas como não existia nenhum padrão, havia algumas discrepâncias que posteriormente foram solucionadas com a criação de um comitê pela ANSI (*American National Standards Institute*) em 1983 onde foi definido um padrão para a linguagem C (SCHILDT, 1997).

Segundo Schildt (1997), C é classificada como uma linguagem de médio nível porque combina elementos de linguagens de alto nível com funcionalidades da linguem Assembly, linguagem de mais baixo nível, com comunicação direta com o *hardware*.

A linguagem C foi desenhada para que o programador possa programar programas estruturados e modulares, tendo como resultado mais legibilidade e documentação, além de ter a vantagem de serem bastante compactos e de execução rápida. Ela é uma linguagem amiga do programador por ser suficientemente estruturada para encorajar bons hábitos de programação. Programas em C podem ser desenvolvidas em várias partes separadas por diferentes usuários e depois unidas para formar um único programa, isto significa que podem ser criadas bibliotecas de funções para serem distribuídas e usadas sem que necessariamente seja conhecido o código fonte de cada uma delas (MIZRAHI, 2008).

4.2.3 Linguagem para a Web

Uma página da Web é uma fonte de informações que está adequada à *World Wide Web* e que pode ser acessada por um navegador Web. Ela é um documento feito para atender aos requisitos da rede mundial de computadores sendo capaz de ser visualizada mediante a utilização de um programa específico para esse fim (GALLOIS, 2008).

Criada por Tim Berners-Lee, no início da década de 1990, a *World Wide Web* veio para aperfeiçoar a comunicação no CERN (Centre European pour la Recherche Nucleaire). Berners-Lee criou o HTML (*HyperText Markup Language*), uma linguagem de marcação baseada na linguagem SGML (*Standard Generalized Markup Language*), com o objetivo de formatar os documentos que seriam distribuídos na rede. Também desenvolveu os protocolos de comunicação para

tornar o seu novo sistema de informações em hipertexto viável (BERNERS-LEE, 1999).

De acordo com Gallois (2008), as páginas Web podem ficar localizadas em um computador local ou remoto para serem disponibilizadas através de um servidor Web. O acesso às páginas pode ser feito pelas redes locais, apenas no próprio computador ou serem publicadas na *Internet*. A requisição e o acesso são feitos por meio de um protocolo de transferência de hipertexto chamado HTTP (*Hypertext Transfer Protocol*).

Inicialmente, as páginas Web eram arquivos de texto estáticos guardados dentro de servidores Web. Atualmente, encontram-se servidores que geram dinamicamente os arquivos HTML de acordo com o que é requisitado pelo navegador.

4.2.3.1 PHP

Segundo Melo & Nascimento (2007), em 1994 Rasmus Lerdorf criou uma série de utilitários para monitorar sua página Web pessoal a fim de obter informações sobre quem a visitava. Com o passar do tempo, foram requeridas mais funcionalidades, então Rasmus desenvolveu uma implementação usando a linguagem C, dando origem ao que ficou conhecido como PHP/FI (*Personal Home Page Tools/Forms Interpreter*).

O PHP é uma linguagem de criação de scripts do lado do servidor que foi criada para ser usada especificamente na Web. Pode ser embutido um código PHP dentro de uma página HTML que será executado cada vez que página for acessada. O código PHP é interpretado pelo servidor Web, gerando o HTML que será visualizado através do navegador de quem a acessa (WELLING & THOMSON, 2005).

PHP tem a vantagem de ser um produto de código fonte aberto, dando a possibilidade de qualquer pessoa ter acesso ao seu código e também a liberdade de alterá-lo e redistribuí-lo sem ônus financeiro. Além de que ele é muito poderoso e eficiente, podendo atender até mesmo milhões de acessos ainda que hospedado

num servidor barato. Segundo o *site* oficial do PHP (2014), em janeiro de 2013 já havia 244 milhões de páginas Web utilizando sua linguagem.

4.3 SERVIDORES WEB

Segundo Morimoto (2006), um servidor é uma máquina que fica ligada constantemente, sempre servindo para o mesmo propósito. Há vários tipos de servidores, tais como servidores Web, servidores de arquivos, servidores de impressão, etc., sendo que uma única máquina apenas pode rodar vários serviços simultaneamente, o que depende apenas da capacidade do *hardware* e da demanda.

Um servidor Web é um servidor responsável por gerenciar, armazenar e distribuir páginas de um determinado *website*, na qual sua apresentação se dá através de comandos requisitados pelos clientes através dos navegadores (GRUPPEN, 2014).

4.3.1 Servidor Apache

O servidor Web Apache é um servidor Web de código aberto desenvolvido e disponibilizado livremente pela Apache Software Foundation. De acordo com a Netcraft, o servidor Web Apache, detinha aproximadamente 60% do mercado de servidores Web no início de 2007. Atualmente há versões para as principais plataformas existentes no mercado, dentre elas o Windows e o Linux, sendo altamente recomendado para o Linux devido a questões de desempenho. Possui um rol de características, configurações e ferramentas, dentre os quais um poderoso sistema de registro de *logs*, altíssima segurança, configuração detalhada, implementação de *hosts* virtuais, suporte a SSL (*Secure Sockets Layer*), entre outros (MELO & NASCIMENTO, 2007).

Uma das principais vantagens do servidor Web Apache, além de ser um *software* livre, é o fato de dar suporte uma enorme gama de páginas Web com diferentes linguagens e protocolos, tais como (MARCELO, 2005):

- Suporte a HTTP 1.1 para a criação de *hosts* virtuais baseados em DNS;
- Suporte a *Secure Socket Layer* (SSL) para transações seguras;
- Suporte a CGIs, Perl e PHP;
- Suporte a autenticação baseada em HTTP;
- Suporte a *Server Side Includes* (SSI);
- Suporte a *Servlets* Java;
- *Logs* customizáveis;
- Configuração rápida e simples.

5 DESENVOLVIMENTO DO PROJETO

5.1 ARDUINO NO CONTROLE DA AUTOMAÇÃO RESIDENCIAL

A implementação do sistema de automação residencial para até 8 pontos de controle foi realizada através do Arduino Mega 2560, pois este possui as melhores características que satisfazem as necessidades das práticas a serem elaboradas. Neste capítulo será descrito o programa inicial e seus periféricos.

A escolha do Arduino foi realizada devido a sua compatibilidade com o *Shield* Ethernet, ideal para uso em monitoramento remoto de estações de registros de dados, além de possuir conexão com rede Ethernet através de conectores do tipo RJ45.

5.1.1 Custos dos Equipamentos Utilizados no Projeto

Na Tabela 9 é apresentada a lista inicial dos módulos básicos utilizados neste trabalho, bem como seus custos de aquisição:

Tabela 9 – Tabela de custos dos equipamentos utilizados.

Descrição	Unidade	Quantidade	Preço	Total
Arduino Mega 2560 R3 Compatível	peça	1	R$ 79,00	R$ 79,00
Ethernet Shield Sd Card - W5100	peça	1	R$ 59,90	R$ 59,90
Módulo de Relés 8 Canais 5v para Automação Optoacoplador	peça	1	R$ 69,90	R$ 69,90
Módulo Dimmer AC (Bivolt)	peça	2	R$ 44,90	R$ 89,80
Fonte de alimentação 9V 1A para Arduino	peça	1	R$ 19,90	R$ 19,90
Caixa Sistema VDI 20x20 Sobrepor Tigre	peça	1	R$ 35,00	R$ 35,00
Cabo Elétrico Flexível 2,5mm²	metros	50	R$ 0,90	R$ 45,00
Tomada de Sobrepor com 3 tomadas	peça	1	R$ 10,00	R$ 10,00
Cabo de Rede Trançado Azul	metros	20	R$ 1,65	R$ 33,00
Conector para Cabo de Rede RJ45 Macho	peça	2	R$ 0,33	R$ 0,66
Mão de Obra para 2 eletricistas	horas	8	R$ 100,00	R$ 800,00
			TOTAL	**R$ 1.242,16**

Todos os módulos foram adquiridos pela equipe para desenvolvimento do TCC, sendo que a Figura 12 apresenta uma foto da disposição das cinco placas que constituem o sistema: placa Arduino Mega (1), modulo de relés com oito canais (2), dois módulos Dimmer AC (3) e placa Arduino Ethernet (4).

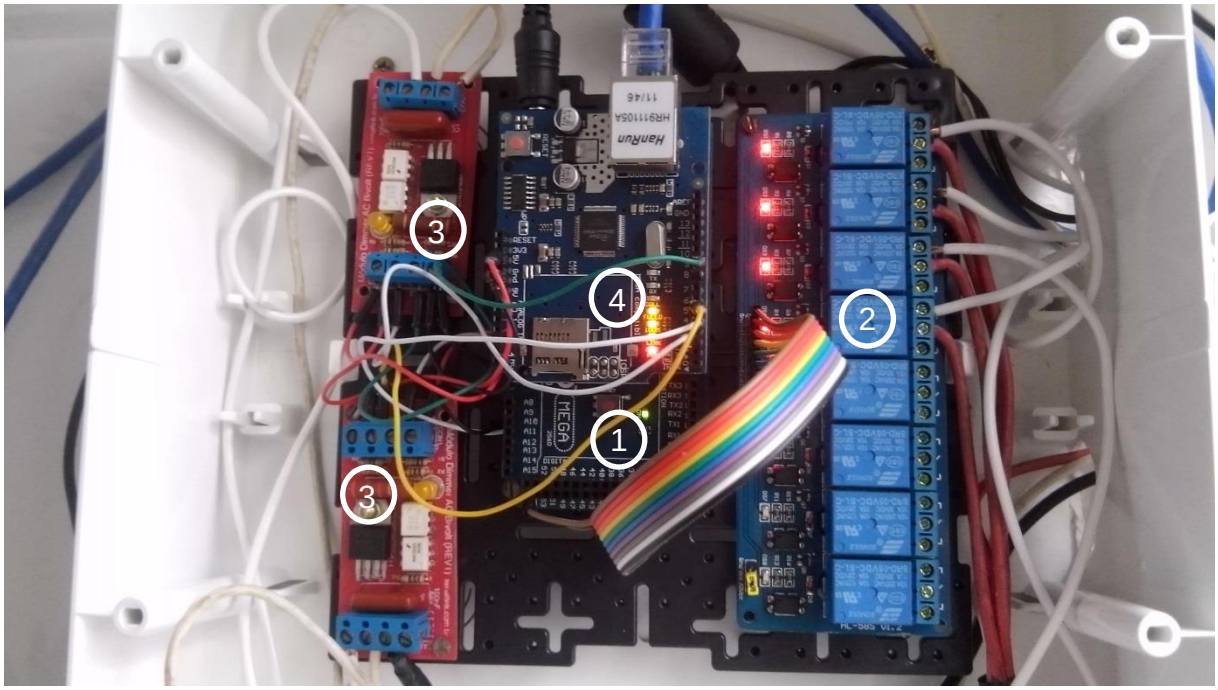


Figura 12 – Projeto Automação Residencial controlado por Arduino
Fonte: Autoria própria.

5.1.2 Programação do Arduino

Através da instalação do *software* livre IDE do Arduino, em sua versão para o sistema operacional Windows, foi possível realizar sua programação através da conexão com o computador pelo cabo USB, item adquirido junto com a placa Arduino Mega 2560.

Com o *software* inicializado em sua pagina de programação (Figura 13), pode-se acessar a biblioteca Ethernet existente no Arduíno e configurar um endereço IP, o *gateway*, e por fim a submáscara para ser utilizada na rede, conforme descrito no Quadro 1.

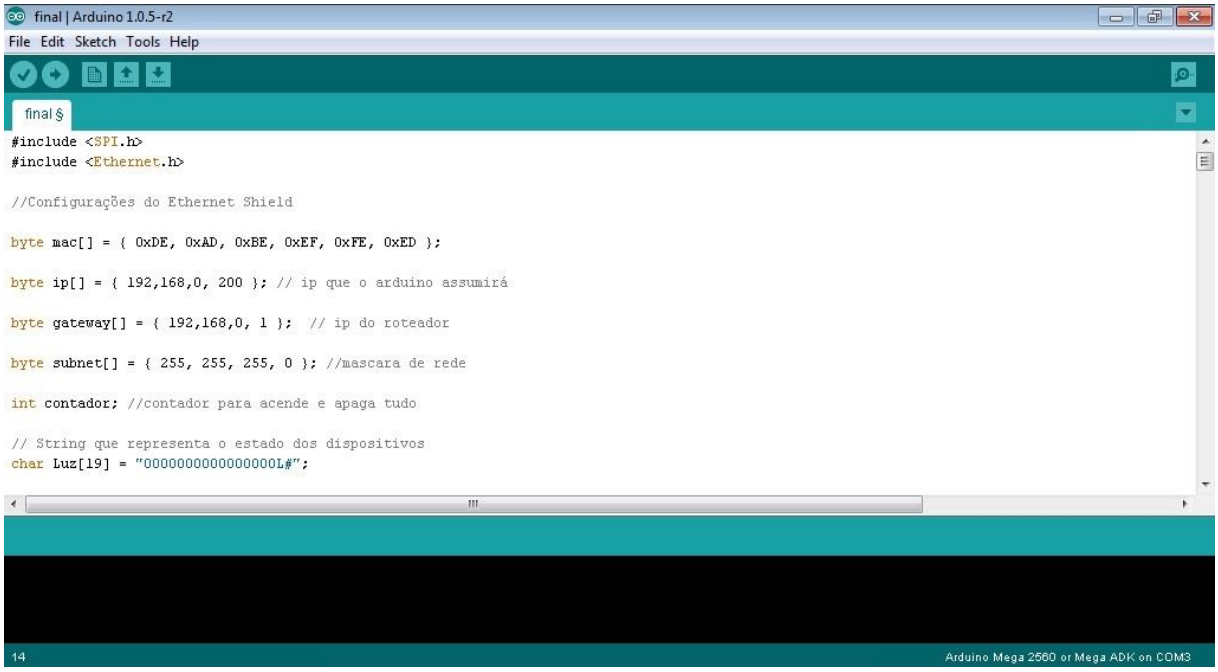


Figura 13 – *Software* IDE Arduino (versão para sistema operacional Windows)
Fonte: Autoria própria.

```
#include <Ethernet.h>
//Configurações do Ethernet Shield
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192,168,100, 82 };        // ip que o Arduino assumirá
byte gateway[] = { 192,168,100, 1 };    // ip do roteador
byte subnet[] = { 255, 255, 255, 0 };
```

Quadro 1 – Programação Ethernet do Arduino
Fonte: Autoria própria.

Conforme mostrado no Quadro 2, as principais variáveis foram declaradas como *char* (8 *bits*). Neste caso, cada variável representa o estado dos dispositivos, nomeados como "*char Luz*", e os *status* das portas e mensagens recebidas através da variável "*char msg*".

```
// String que representa o estado dos dispositivos
char Luz[19] = "0000000000000000L#";
// String onde é guardada as msgs recebidas
char msg[19] = "0000000000000000L#";
```

Quadro 2 – Programação de variáveis do Arduino.
Fonte: Autoria própria.

Com a utilização da *sketch* "*voidsetup"* definem-se os pinos que serão utilizados como entradas e saídas (Quadro 3). O comando "*void loop*", utilizado posteriormente, chama a biblioteca EthernetClient, conforme apresentado no Quadro 3. Caso a resposta seja um *caractere*, será armazenada o *status* de cada porta utilizada na *string msg*. Após receber o *caractere*, automaticamente é devolvido o *status* do dispositivo através da *string Luz.* Dessa forma, fica atualizado o *status* da leitura das informações de cada porta. Por exemplo, quando o microcontrolador recebe um *caractere* de comando, é acionada a porta registrada para receber este comando e alterado o *status* atual. Considerando que a porta esteja desligada, esta será acionada mudando *status* para 1. Caso já esteja acionada, seu o *status* será alterado para 0.

```
void setup() {
Ethernet.begin(mac, ip, gateway, subnet);
pinMode(22,OUTPUT);
pinMode(23,OUTPUT);
pinMode(24,OUTPUT);
pinMode(25,OUTPUT);
pinMode(26,OUTPUT);
pinMode(27,OUTPUT);
pinMode(28,OUTPUT);
pinMode(29,OUTPUT);
void loop() {
  EthernetClient client  = server.available();
   // SE receber um caracter...
  if (client) {
    // guarda o caracter na string 'msg'
msg[1]=msg[2];       msg[2]=msg[3];       msg[3]=msg[4];       msg[4]=msg[5];
msg[5]=msg[6];msg[6]=msg[7];msg[7]=msg[8];msg[8]=msg[9];msg[9]=msg[10];msg[
10]=msg[11];msg[11]=msg[12];msg[12]=msg[13];
    msg[13]=msg[14];msg[14]=msg[15];msg[15]=msg[16];msg[16]=msg[17];msg[17]
```

```
=msg[18];
    msg[18] = client.read();

    if (msg[18]=='#') {
      switch(msg[17]) {
        case '1':
//Controle de Portas com Status
case '1':
// Caso 1#, aciona a porta 1
if (Luz[0]=='0') {
digitalWrite(22,HIGH);
Luz[0]='1';}
else {
digitalWrite(22,LOW);
Luz[0]='0';}
break;
```

Quadro 3 – Programação de *status* das Portas.
Fonte: Autoria própria.

5.1.3 Programação do Servidor em PHP

No cabeçalho do PHP, com a criação de um *socket*, é informada a conexão de IP, a porta do Arduino e em seguida os comandos de ação correspondente ao botão acionado de cada porta do Arduino (Quadro 4). Dessa forma, o Arduino recebe o comando do botão a ser acionado, executando o comando conforme a programação (Quadro 3).

```
$sock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP); // Se conecta ao IP e
Porta:
socket_connect($sock,"192.168.0.200", 26);
// Executa a ação correspondente ao botão apertado.
if(isset($_POST['bits'])) {
  $msg = $_POST['bits'];
  if(isset($_POST['1'])){ $msg = '1#'; }
  if(isset($_POST['2'])){ $msg = '2#'; }
```

```
  if(isset($_POST['3'])){ $msg = '3#'; }
  if(isset($_POST['4'])){ $msg = '4#'; }
  if(isset($_POST['5'])){ $msg = '5#'; }
  if(isset($_POST['6'])){ $msg = '6#'; }
  if(isset($_POST['7'])){ $msg = '7#'; }
  if(isset($_POST['8'])){ $msg = '8#'; }
```

Quadro 4 – Programação em PHP para criação do *socket*.
Fonte: Autoria própria.

Através do comando *socket_write*, é solicitado o estado de cada porta do Arduino. Este requisita o *status* do sistema, e após interpretar a resposta, define a cor dos botões de acordo com o status recebido. Se receber *status* 0 muda a cor para vermelho, se receber o *status* 1 muda para a cor verde (Quadro 5). Pode ser melhor visualizado na Figura 14, na tela do projeto de AR.

```
socket_write($sock,'R#',2); //Requisita o status do sistema.
 // Espera e lê o status e define a cor dos botões de acordo.
$status = socket_read($sock,18);
$temp = socket_read($sock,5);
if (($status[16]=='L')&&($status[17]=='#')) {
  if ($status[0]=='0') $cor1 = 'lightcoral';// vermelho
    else $cor1 = 'lightgreen'; // verde
```

Quadro 5 – Programação em PHP para o comando *socket_write*.
Fonte: Autoria própria.

Para o desenvolvimento da interface *web* PHP, foi utilizado o editor NotePad++(disponível em http://notepad-plus-plus.org), na qual os códigos foram inseridos e salvos com extensão “.php” e seu arquivo salvo com o nome “Pagina.php” no diretório “c:\\xampp\htdocs”, para que possa ficar hospedado no servidor *web* APACHE que está integrado com o XAMPP. Deste modo quando requisitado por um navegador *web* o endereço *IP* do servidor, será interpretado e exibido na tela a página de comando e controle da automação residencial.

Os códigos em PHP foram desenvolvidos com o auxílio de pesquisas e tutoriais encontrados no seu *site* oficial (Disponível em http://php.net/manual/pt_BR/index.php).

Figura 14 – Tela do projeto de automação residencial baseado em Arduino.
Fonte: Autoria própria.

Como ilustrado na Figura 14, para definir o estilo do botão (fonte, tamanho) e inserir a nomenclatura de cada porta do Arduino é utilizado o comando "*echo*" (Quadro 6).

```
  echo "<form method =\"post\" action=\"Pagina.php\">";
  echo "<input type=\"hidden\" name=\"bits\" value=\"$status\">";
  echo "<button style=\"width:400; background-color: $cor1 ;font: bold 50px
Arial\" type = \"Submit\" Name = \"1\">1</button>      ---------      ";
```

Quadro 6 – Programação de botões.
Fonte: Autoria própria.

Por fim, o botão *refresh* (Quadro 7), faz a releitura do *status* das portas atualizando a página, para quando há mais de um acesso com outro *smartphone.*

```
  echo "<button style=\"width:400;font: bold 50px Arial\" type = \"Submit\"
Name = \"Refresh\">Refresh</button></br></br>";
  echo "</form>";
      echo "$tatus";
```

Quadro 7 – Programação do botão *refresh*.
Fonte: Autoria própria.

5.1.4 Instalação do Servidor Web APACHE

Para a instalação do Servidor Web APACHE com suporte a PHP, optou-se por instalar o XAMPP, *software* livre, que possui a característica de ser um servidor independente de plataforma. A ferramenta contém o Servidor Web APACHE e os interpretadores para a linguagem de *scripts* PHP.

Os procedimentos para instalação são os descritos a seguir:

- *Download* do XAMPP para Windows Versão v5.6.3 (PHP 5.6.3) no endereço eletrônico https://www.apachefriends.org/pt_br/index.html;
- Instalação do XAMPP no computador que servirá como servidor;
- Instalação do código fonte da interface *web* de controle dos comandos;
- Inicialização do APACHE.

Após a conclusão do *download* do XAMPP deve-se executar o arquivo de instalação e instalar com todas as opções padrões, bastando clicar em "*Next*" até a janela com o botão "*Finish*" para concluir a instalação. Neste momento, deve-se executar o atalho do XAMPP CONTROL PANEL para iniciar a tela de controle do XAMPP (Figura 15). Por padrão o XAMPP é instalado na pasta "c:\xampp".

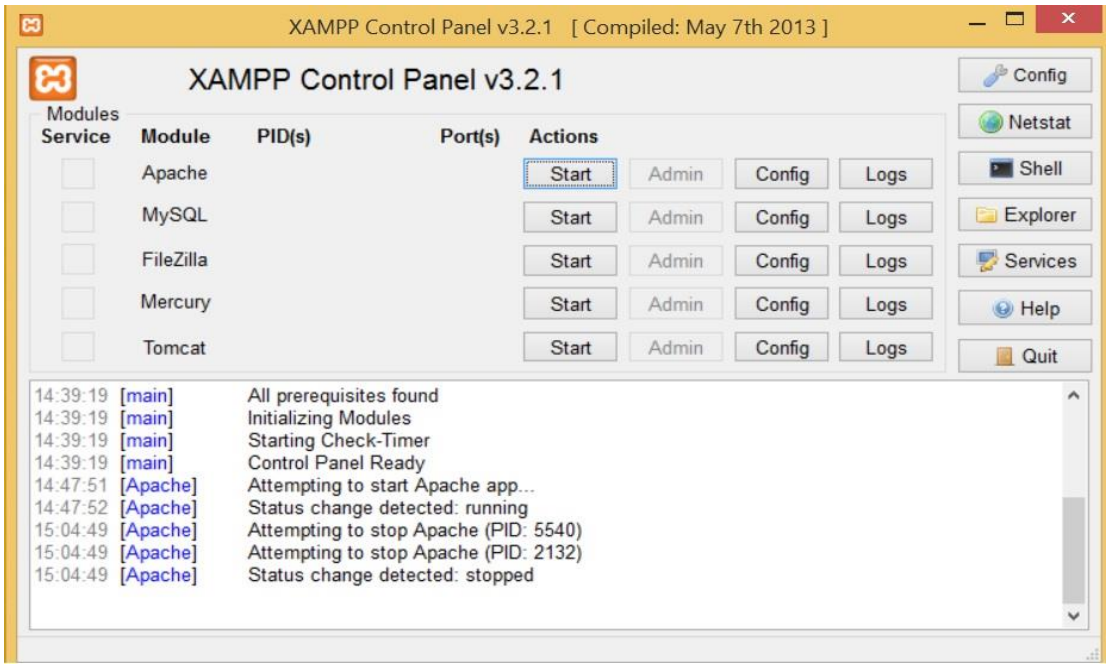


Figura 15 – Tela de controle do XAMPP.
Fonte: Autoria própria.

No Painel de Controle do XAMPP, mostrada na Figura 15, deve-se iniciar o servidor APACHE clicando no botão *"Start"*, localizado imediatamente à direita do módulo APACHE, deixando as demais configurações como estão.

Para se certificar de que o XAMPP foi instalado corretamente é necessário abrir um navegador *web*, como por exemplo, o "Internet Explorer", e digitar no campo de endereços: "http://localhost". Após este procedimento, retorna-se a tela de boas vindas do XAMPP, informando que foi corretamente instalado (Figura 16).

Figura 16 – Tela de boas vindas do XAMPP.
Fonte: Autoria própria.

Com a tela de boas vindas executada com sucesso, é possível seguir para o passo seguinte onde é instalado o arquivo "Pagina.php", com os códigos fonte da página *web* para que possa ser executado pelo servidor *web* APACHE. O código fonte desenvolvido neste trabalho é apresentado no Apêndice A.

5.1.5 Comandos no *Smartphone*

Para utilização dos comandos no *smartphone*, basta acionar os botões correspondentes e o Arduino interpretará, conforme o caso, se liga ou desliga uma das interfaces disponíveis. Assim que o botão correspondente por pressionado, este mudará de cor para indicar o seu estado. Na cor verde o botão indica ligado e na cor vermelha, desligado.

A diferença encontra-se nos pontos em que se utiliza o *dimmer*. Neste caso, cada acionamento altera um estágio até chegar ao máximo de potência do aparelho. Analogamente, na máxima potência selecionada basta acionar o respectivo botão que irá diminuir um estágio até o desligamento do aparelho.

Os estágios acima citados foram divididos em cinco. Como o *dimmer* utilizado possui um TRIAC na passagem por zero, cada semiciclo da senóide da rede elétrica foi divido nestes cinco estágios, sendo então o ângulo de disparo do TRIAC é igual a incrementos de 36°. Resumindo: a cada acionamento do botão, o ângulo de disparo aumenta 36° até completar o semiciclo – mínima potência entregue a carga – e após diminui os mesmos passos de 36°, até a máxima potência entregue a carga.

Por sua vez, o botão “*refresh”* faz o papel de atualizar o estado dos aparelhos na tela do *smartphone*. Isso se deve ao caso de a residência possuir mais um telefone celular capaz de executar os comandos aqui descritos, pois o estado apenas estará atualizado automaticamente para o *smartphone* que acionar o botão. Isto é, o outro usuário deve acionar o “*refresh”* para identificar qual aparelho está ligado ou desligado, antes de efetuar qualquer comando.

Já os botões “acende tudo” e “apaga tudo”, enviarão o sinal para que o Arduino interprete qual aparelho ainda não está ligado, no caso do acende tudo, ou desligado, no caso de apaga tudo, executando então o comando conforme o caso. Ou seja, acendendo tudo irá ligar os aparelhos que estão desligados e vice-versa. Cabe salientar que este comando não irá acionar os aparelhos que utilizam *dimmers.*

O fluxograma principal de funcionamento do sistema desenvolvido é apresentado no Apêndice B.

5.1.6 Implantação Física do Sistema

O sistema de automação desenvolvido neste trabalho foi implementado na casa de um dos componentes da equipe com o objetivo de verificar seu comportamento quando submetido a condições reais de utilização.

A instalação do Arduino (Figura 17) foi realizada na lavanderia da residência através de uma caixa de sobrepor na parede logo acima do quadro de distribuição de energia. Desse modo, foi facilitada a passagem da fiação para os disjuntores de proteção dos circuitos utilizados para a instalação dos pontos automatizados.

Figura 17 – Arduino instalado na residência.
Fonte: Autoria própria

Na implantação física foram automatizados sete pontos de iluminação, um ventilador e deixado dois pontos para reserva, nos quais não foram executados testes, uma vez que foram deixados para substituir algum ponto que porventura

apresentasse defeito. Para um ponto de iluminação e ventilador foram utilizados *dimmers* para regular a intensidade de luminosidade e velocidade do ventilador, cujo funcionamento mostrou-se confiável durante o tempo em que foi exigido. A iluminação e o ventilador responderam satisfatoriamente aos comandos do usuário, ligando e desligando quando solicitados durante os 25 dias em que esteve implantado.

A planta do sobrado automatizado é apresentada no Apêndice C.

6 DISCUSSÃO

Conforme observado na pesquisa de preços de mercado, é possível verificar que a automação residencial com sistemas similares ao proposto neste trabalho, têm seus custos a partir de R$ 3.600,00. Já o sistema desenvolvido pela equipe obteve custos de aproximadamente R$ 1.500,00. Portanto, a utilização do sistema desenvolvido têm resultados satisfatórios quanto ao custo de implantação. No entanto, é importante salientar que tais custos podem ser diferentes dependendo da necessidade do usuário, ou seja, quantos pontos a automatizar, quais aparelhos e equipamentos se deseja inserir no sistema. Também pode haver variação do custo de mão de obra para instalação do sistema e cablagem, mas deve levar-se em consideração de que o sistema proposto é de fácil implementação e pode ser instalado pelo próprio usuário.

Por todo o tempo em que esteve conectado, 25 dias, o comportamento do sistema se mostrou estável e respondendo aos comandos do usuário. Porém, tal resultado deve ser visto com cautela, uma vez que, por ser curto o tempo pelo qual o sistema ficou ligado, não é possível chegar a conclusões confiáveis e definitivas a cerca da robustez do equipamento.

Durante estes 25 dias em que esteve conectado, o sistema foi utilizado conforme rotina do morador, simulando assim uma situação real para o qual foi desenvolvido. Nesse cenário, foram efetuados os comandos conforme a necessidade do usuário. Como trata-se principalmente do uso de lâmpadas, a média de comandos efetuados foi de 3 acionamentos diários, totalizando 450 comandos nos 6 pontos mais utilizados de iluminação.

Quanto aos pontos com *dimmers,* o que possui o ventilador teve acionamentos diários em velocidade máxima, devido à temperatura ambiente elevada (mês de Janeiro de 2015). Como para acionar este ponto ao máximo são necessários 5 comandos, sendo o mesmo para desligá-lo, o total de comandos realizados neste período para ligar e desligar totalizou 250. Já a lâmpada com *dimmer* teve seu uso de forma mais tímida, sendo acionado, com um total de 60 comandos, incluindo ligar e desligar. Durante este período não foram detectados problemas quanto ao acionamento dos pontos automatizados, ou seja, 100% dos comandos enviados responderam satisfatoriamente.

Findado este período de 25 dias, foram realizados testes de maneira contínua no sistema, com comandos nos pontos de interesse durante um período de 70 minutos, sendo divididos 10 minutos para cada ponto de iluminação e 5 minutos para cada *dimmer,* com execução de 4 comandos por minuto por ponto de iluminação e 10 comandos por ponto com *dimmer*, incluindo ligar e desligar. Totalizando 40 comandos em cada ponto de iluminação e 50 para os *dimmers.* Durante este período não foram detectados problemas quanto ao acionamento dos pontos automatizados, ou seja, 100% dos comandos enviados responderam satisfatoriamente.

Porém, ao final destes testes contínuos, constatou-se ao verificar as placas instaladas, um aumento na temperatura do módulo de relés ao contato físico com a mão. Todavia, o módulo não deixou de funcionar, mas deve-se levar em consideração quanto à determinada quantidade de operações.

Como já descrito anteriormente, tais resultados devem ser observados com cuidado caso se trate da capacidade de operação do equipamento, pois para que o sistema apresente confiabilidade satisfatória, deve ser testado por mais tempo e submetido às mais diversas situações para garantir o funcionamento, tanto quanto ao tempo de utilização como à quantidade de comandos suportados pelos equipamentos eletrônicos que fazem parte do sistema proposto.

Para o controle supervisório, foi decidido pela página *web* em detrimento a um aplicativo para *smartphone*, devido àquela ser facilmente acessível, visto que o *smartphone* e o acesso à *Internet* por este dispositivo estão cada vez mais comuns entre os usuários de telefonia celular nos dias atuais – o código da página *web* utilizado neste trabalho está disponível no CD do Trabalho de Conclusão de Curso e na biblioteca da UTFPR, bem como no Apêndice A. Além disso, celulares que possuem apenas acesso a um navegador podem ser utilizados neste sistema. Por outro lado a criação de um aplicativo demandaria custos adicionais ao desenvolvimento deste trabalho, afastando-se do propósito de reduzir custos. Ainda, os aplicativos dependem do sistema utilizado pelo usuário como, por exemplo, *Windows Fone, IPhone* ou *Android,* ou seja, para atingir todos os usuários, seria necessário o desenvolvimento de diferentes aplicativos conforme o sistema utilizado pelo proprietário do imóvel.

7 CONCLUSÃO

Com a interface gráfica utilizada, verificou-se uma grande facilidade de comandar o sistema. A interface *web* simples mostrou-se bastante eficiente e didática no que tange aos comandos necessários para se ligar ou desligar os aparelhos. Dessa forma, cabe ressaltar a possibilidade de avaliação dos recursos da residência nos conceitos de economia, conforto, segurança e comodidade.

A utilização do PHP facilitou o desenvolvimento do sistema supervisório, pois é gratuito e possibilita rápida criação das interfaces, além de possuir diversos tutoriais disponibilizados pelos usuários ao redor do globo para auxiliar quando da necessidade de ajuda. Além disso, pode ser utilizado com diversos sistemas operacionais via *web*.

Por sua vez o Arduino permitiu grande facilidade na automação dos aparelhos, pois possui um sistema modular (*Shields*) que pode ser adaptado conforme a necessidade de utilização por parte do usuário, pois possui bibliotecas já consolidadas que auxiliam na programação dos equipamentos eletrônicos.

No que tange à perspectiva social do trabalho, verificou-se que a solução adotada possui um custo menor se comparado a produtos similares do mercado, mesmo com a variação de preço apresentadas pelas empresas do setor, inclusive levando-se em consideração a quantidade de pontos a serem automatizados, mostrando-se uma alternativa barata e de simples implementação para usuários que desejam se aventurar na automação de suas residências. No entanto o usuário deve estar ciente de que não foram realizados testes em longo prazo para verificar a confiabilidade e robustez do sistema.

Embora a automação residencial ainda seja recente no país, com o mercado em crescimento, o desenvolvimento deste sistema pode ajudar na popularização deste segmento, desde uma pessoa que não possua o conhecimento técnico sobre domótica, até pesquisadores que possam aperfeiçoar o sistema, visando sempre o conforto, segurança e comodidade para o usuário.

A automação residencial permite vários estudos e implementações na Engenharia. Diminuição de custos, facilidade, conforto, segurança, são fatores que podem ser explorados para melhorar o bem estar do usuário. Com o Arduino, é possível criar métodos para automatizar os mais diversos equipamentos da

residência, visando alcançar maior comodidade nas tarefas do cotidiano. Entretanto, é necessário saber como o Arduino se comporta se utilizado continuamente nesta função, ou seja, se com determinado tempo de uso e quantidade de operações mantém as mesmas características de funcionamento para verificação de viabilidade de sua utilização em longo prazo. A confecção de um módulo similar pode ser utilizada como comparação de rendimento e eficácia no que tange o propósito deste trabalho.

Além disso, as interfaces gráficas podem ser melhoradas, com vistas a facilitar o processo de gestão da residência pelo usuário, bem como a visualização e correção de problemas, melhorando assim sua eficiência. Neste caso, um aplicativo para *smartphone* pode ser desenvolvido para auxiliar o usuário na operação do sistema, conforme o sistema operacional utilizado. Como neste trabalho foi utilizado rede WI-FI local, pode-se desenvolver um sistema cujas operações possam ser realizadas remotamente via *Internet*, para executar comandos fora do alcance da rede WI-FI.

REFERÊNCIAS

ALVES, J. A.; MOTA, J. **Coleção Soluções, Casas Inteligentes, Inova**. Portugal: 2003.

ARDUINO. Disponível em: <http://www.arduino.cc>. Acesso em: jul. 2014.

AURESIDE, Associação Brasileira de Automação Residencial. Disponível em: <www.aureside.org.br>. Acesso em: nov. 2013.

BANZI, M. Primeiros Passos com Arduino. São Paulo: Novatec, 2012.

BASTOS, A. V. **Microcontroladores**: Curso Engenharia de Controle e Automação. Disponível em <http://www.decom.ufop.br/alex/arquivos/sof_bas_ECA/Microcontroladores.pdf>. Acesso em: ago. 2014.

BERNERS-LEE, T. **Weaving the Web:** the original design and ultimate destiny of the World Wide Web by its inventor. 1.ed. San Francisco, CA: Harper San Francisco, 1999.

BOLZANI, C. A. M. **Desenvolvimento de um Simulador de Controle de Dispositivos Residenciais Inteligentes: Uma Introdução aos Sistemas Domóticos**. 2004 115f. Dissertação (Mestrado) – Escola Politécnica da Universidade de São. Paulo. Departamento de *Engenharia* de *Sistemas* Eletrônicos, 2004.

BOLZANI, C. A. M. **Análise de Arquiteturas e Desenvolvimento de uma Plataforma para Residências Inteligentes.** 2010 155f. Tese (Doutorado) – Escola Politécnica da Universidade de São. Paulo. Departamento de *Engenharia* de *Sistemas* Eletrônicos, 2010.

BOLZANI, C. A. M. **Residências Inteligentes**. São Paulo: Editora e Livraria da Física, 2004.

BOLZANI, C. A. M. **Domótica, a nova ciência do século XXI***.* Revista Fonte, p105, dez. 2013. Disponível em: <http://www.bolzani.com.br/artigos/revistafonte.pdf> Acesso em jul. 2014.

BOLZANI, C. A. M. **Desmistificando a Domótica***.* Sinergia, p17, jan./jun. 2007. Disponível em: <http://www.bolzani.com.br/artigos/art01_07.pdf> Acesso em jul. 2014.

CARMONA, T. **Guia Técnico de Redes de Computadores.** São Paulo: Digerati Books, 2007.

DERSHEM, H. L; JIPPING, M. J. **Programming Languages:** Structures and models. 2. ed. Boston: PWS Publishing Company, 1995.

EVANS, M.; NOBLE, J; HOCHENBAUM, J. **Arduino em Ação.** São Paulo: Novatec, 2013.

FILHO, C. F. **História da Computação:** O Caminho do Pensamento e da Tecnologia. Porto Alegre: EDIPUCRS, 2007.

GALLOIS, F.. **Curso Básico de HTML.** Joinville, 2008. (Apostila)

GAZETA DO POVO. Disponível em: <http://www.gazetadopovo.com.br/imobiliario/conteudo.phtml?id=141997>. Acesso em dez. 2013.

GRUPPEN. Disponível em: <http://www.gruppen.com.br/oqueeumservidor/oqueeumservidor.html>. Acesso em jul. 2014.

G1. Disponível em: <http://g1.globo.com/economia/pme/noticia/2013/05/empresa-fatura-r-1-milhao-com-servicos-de-automacao-residencial.html>. Acesso em dez. 2014.

G1. Disponível em: < http://g1.globo.com/tecnologia/noticia/2014/07/smartphones-ja-sao-tres-cada-quatro-celulares-vendidos-no-brasil.html>. Acesso em fev. 2015.

HOME THEATER. Disponível em: <http://revistahometheater.uol.com.br/site/tec_artigos_02.php?id_lista_txt=8367>. Acesso em dez. 2013.

IDEPAC. Disponível em: <http://www.fundacaosergiocontente.org.br/wp-content/uploads/2013/01/redes-avulso.pdf>. Acesso em jul. 2014.

KUROSE, J. F; ROSS, K. W. **Redes de Computadores e a Internet:** Uma Abordagem Top-Down. 5. ed. São Paulo: Pearson, 2010.

MARCELO, A. **Apache:** configurando o servidor WEB para Linux. 3 ed. Rio de Janeiro: Brasport, 2005.

MARSCZAOKOSKI, F. G.; CRUZ, R. P. da; ABREU E SILVA, W. de. **Sistema Microcontrolado de Irrigação Aplicado a Morangueiros.** Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná, 2013.

MARTINS, N. A. **Sistemas Microcontrolados**: uma abordagem com o microcontrolador PIC16F84. São Paulo: Novatec, 2005.

MCROBERTS, M. **Arduino Básico.** São Paulo: Novatec, 2011.

MELO, A. A. de; NASCIMENTO, M. G. F. **PHP Profissional.** NOVATEC, 2007.

MENDES, D. R. **Redes de Computadores:** Teoria e Prática. São Paulo: Novatec, 2007.

MIYAGI, P. E.; BARRETO, M. P. R.; SILVA, J. R.. **Domotica**: Controle e Automação - Tomo II. Argentina: EBAI, 1993.

MIZRAHI, V. V. **Treinamento em Linguagem C.** São Paulo: Pearson Prentice Hall, 2008.

MONK, S. **Programação com Arduino**. São Paulo: Bookman, 2013.

MORAES, C. C.de. **Engenharia de Automação Industrial.** Rio de Janeiro: LTC, 2007.

MORIMOTO, C. E. **Redes e Servidores Linux.** 2. ed. Porto Alegre: GDH Press e Sul Editores, 2006.

MULTILOGICA. Disponível em: <http://www.multilogica-shop.com>. Acesso em: mai. 2014.

MURATORI, J. R.; DAL BÓ, P. H. Automação Residencial, Conceitos e Aplicações. São Paulo. Educere, 2013.

MURATORI, J. R. **Os desafios do mercado da Automação Residencial.** Disponível em: <http://www.aecweb.com.br/cont/a/os-desafios-do-mercado-da-automacao-residencial_8192>. Acesso em ago. 2014.

PEREIRA, F. **Microcontroladores PIC**. São Paulo: Érica, 2005.

PHP, php.net. Disponível em: < http://php.net/usage.php>. Acesso em: ago. 2014.

PRUDENTE, F. **Automação Predial e Residencial:** Uma introdução. Rio de Janeiro: LTC, 2013.

RYE, D. My life at x10. **HomeToys EMagazine**, v. 4, n. 5, 1999.

SCHILDT, H. **C, Completo e Total.** 3 ed. São Paulo: Pearson Makron Books, 1997.

SEBESTA, R. W. **Conceitos de Linguagens de Programação.** 5. ed. Porto Alegre: Bookman Companhia Ed, 2003.

SERIALLINK. Disponível em: <http://www.seriallink.com.br>. Acesso em: jun. 2014.

TANEMBAUM, A. S. **Redes de Computadores.** 4. Ed. Rio de Janeiro: Elsevier, 2003.

TECHMOUNT. Disponível em: <http://www.techmount.com.br>. Acesso em: dez. 2013.

TECH TUDO. Disponível em: <http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/06/vale-pena-investir-em-automacao-para-casa.html>. Acesso em dez. 2014.

TELECO. Disponível em: < http://www.teleco.com.br/tutoriais/repan1/pagina_1.asp>. Acesso em jul. 2014.

VIDAL, P. da S. J. **Aplicações Práticas do Microcontrolador**. São Paulo: Érica, 1999.

WELLING, L.; THOMSON, L. **PHP e MySQL:** Desenvolvimento Web. 3. ed. Rio de Janeiro: Elsevier, 2005.

APÊNDICE A – CÓDIGO FONTE DA INTERFACE WEB EM PHP

```
<html>
<head></head>
<body>

<?php
$sock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
// Se conecta ao IP e Porta:
socket_connect($sock,"192.168.0.200", 26);

// Executa a ação correspondente ao botão apertado.
if(isset($_POST['bits'])) {
  $msg = $_POST['bits'];

  if(isset($_POST['1'])){ $msg = 'F#'; }
  if(isset($_POST['2'])){ $msg = 'T#'; }
  if(isset($_POST['3'])){ $msg = 'H#'; }
  if(isset($_POST['4'])){ $msg = 'C#'; }
  if(isset($_POST['5'])){ $msg = 'c#'; }
  if(isset($_POST['6'])){ $msg = 'g#'; }
  if(isset($_POST['7'])){ $msg = 'S#'; }
  if(isset($_POST['8'])){ $msg = 'O#'; }
  if(isset($_POST['Lamp'])){ $msg = 'P#'; }
  if(isset($_POST['Vent'])){ $msg = 'p#'; }

 // if(isset($_POST['Interna'])){ if($msg[2]=='1') { $msg[2]='0'; } else { $msg[2]='1'; }}
 // if(isset($_POST['Fundos'   ])){ if($msg[3]=='1') { $msg[3]='0'; } else { $msg[3]='1'; }}
 // if(isset($_POST['5'   ])){ if($msg[4]=='1') { $msg[4]='0'; } else { $msg[4]='1'; }}
```

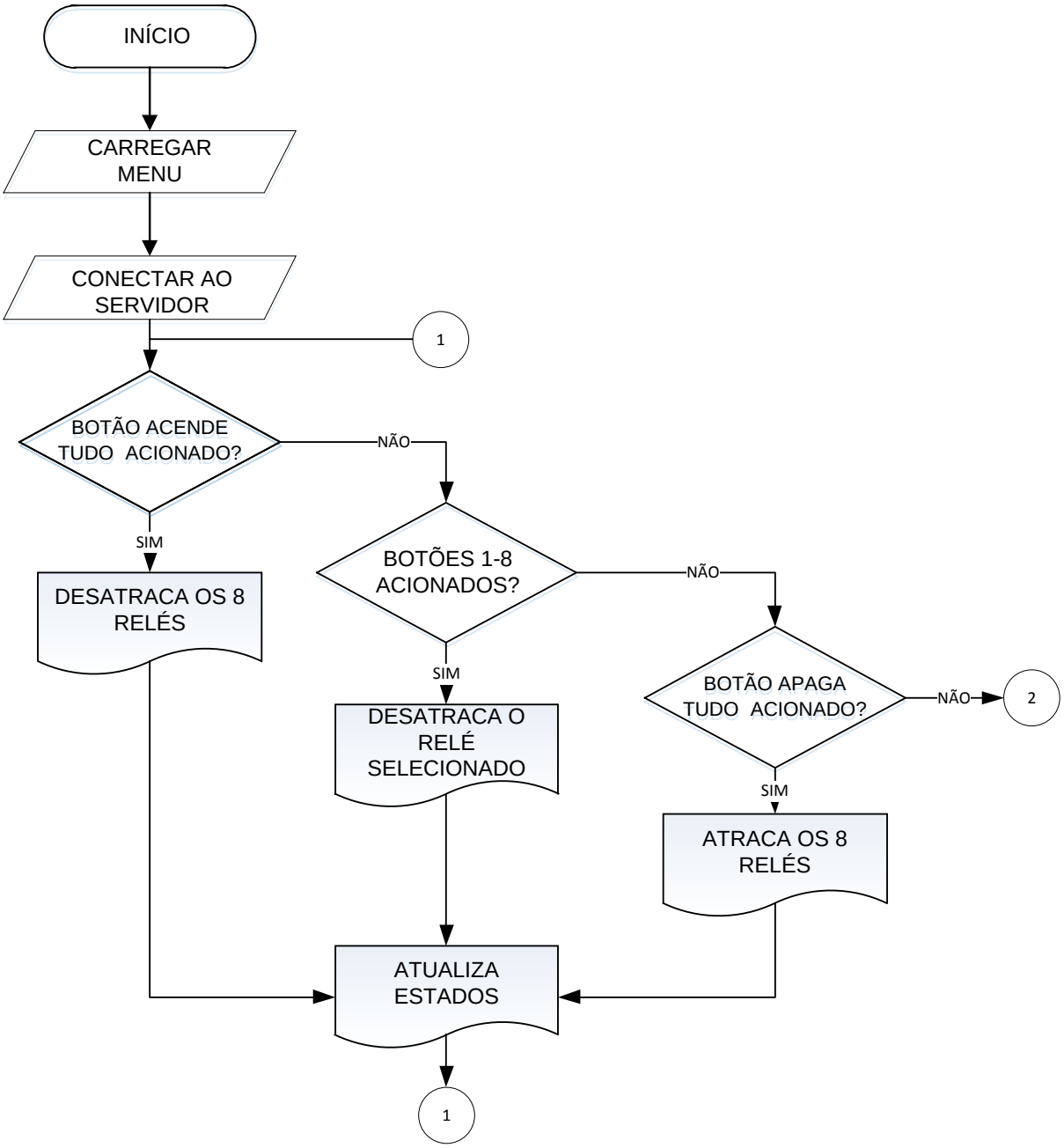
```
// if(isset($_POST['6'   ])){ if($msg[5]=='1') { $msg[5]='0'; } else { $msg[5]='1'; }}
 //if(isset($_POST['7'   ])){ if($msg[6]=='1') { $msg[6]='0'; } else { $msg[6]='1'; }}
// if(isset($_POST['8'   ])){ if($msg[7]=='1') { $msg[7]='0'; } else { $msg[7]='1'; }}
// if(isset($_POST['9'   ])){ if($msg[8]=='1') { $msg[8]='0'; } else { $msg[8]='1'; }}
// if(isset($_POST['10'   ])){ if($msg[9]=='1') { $msg[9]='0'; } else { $msg[9]='1'; }}
// if(isset($_POST['11'   ])){ if($msg[10]=='1') { $msg[10]='0'; } else { $msg[10]='1'; }}
//  if(isset($_POST['12'   ])){ if($msg[11]=='1') { $msg[11]='0'; } else { $msg[11]='1'; }}
// if(isset($_POST['13'   ])){ if($msg[12]=='1') { $msg[12]='0'; } else { $msg[12]='1'; }}
//  if(isset($_POST['14'   ])){ if($msg[13]=='1') { $msg[13]='0'; } else { $msg[13]='1'; }}


 if(isset($_POST['ApagaTudo'   ])){
        $i = '0';
        while ($i<=15){
        //      $msg[$i]='0';
                $status[$i]='0';
                $i++;
        }
        $msg = 'a#';
}
 if(isset($_POST['AcendeTudo'   ])){
$i = '0';
        while ($i<=15){
                //$msg[$i]='1';
                $status[$i]='1';
                $i++;
        }
        $msg = 'A#';
}

 socket_write($sock,$msg,strlen($msg));
```

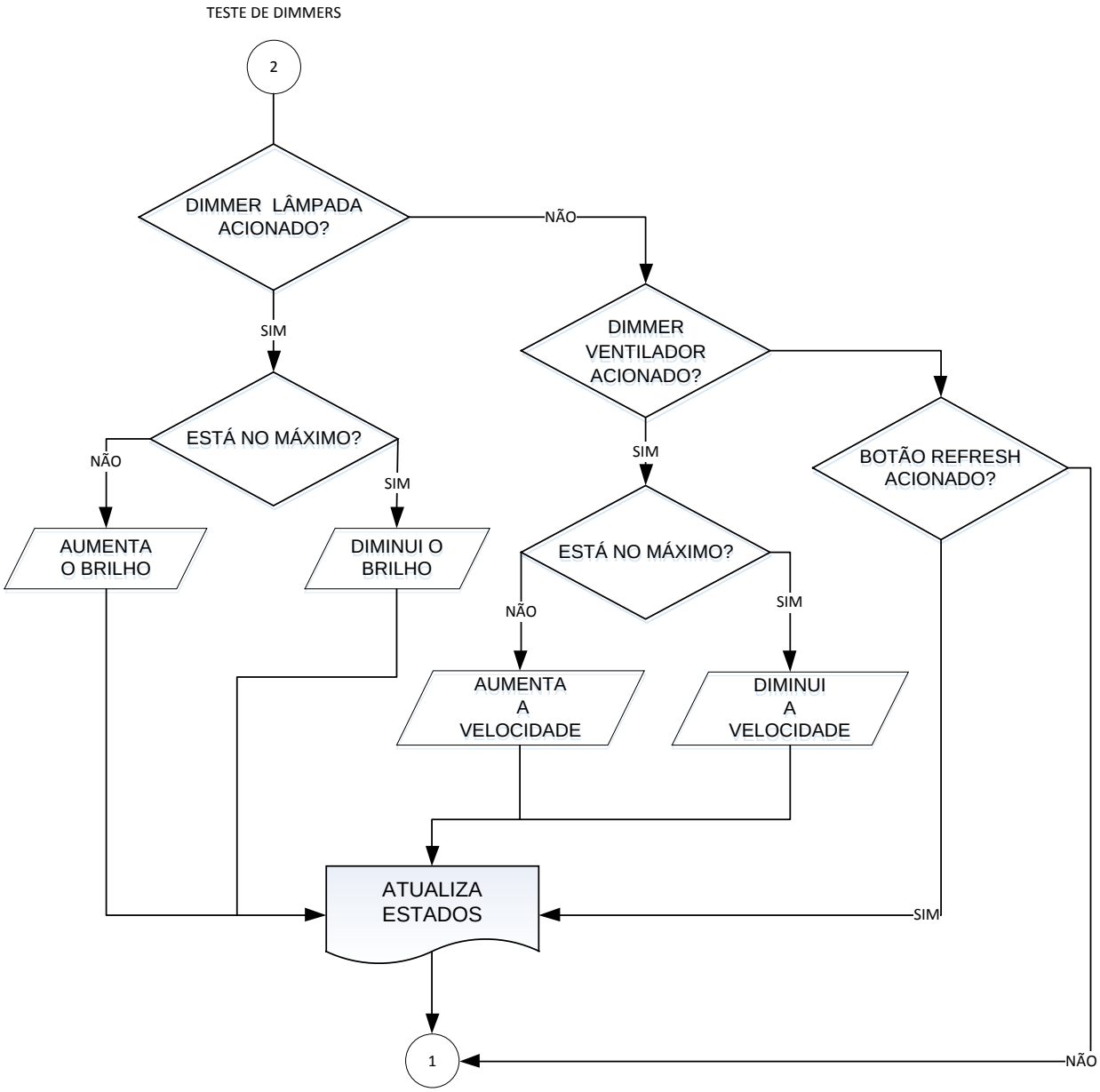
```
}

socket_write($sock,'R#',2); //Requisita o status do sistema.

 // Espera e lê o status e define a cor dos botões de acordo.
$status = socket_read($sock,18);
$temp = socket_read($sock,5);
if (($status[16]=='L')&&($status[17]=='#')) {
  if ($status[0]=='0') $cor1 = 'lightcoral';
    else $cor1 = 'lightgreen';
  if ($status[1]=='0') $cor2 = 'lightcoral';
    else $cor2 = 'lightgreen';
  if ($status[2]=='0') $cor3 = 'lightcoral';
    else $cor3 = 'lightgreen';
  if ($status[3]=='0') $cor4 = 'lightcoral';
    else $cor4 = 'lightgreen';
  if ($status[4]=='0') $cor5 = 'lightcoral';
    else $cor5 = 'lightgreen';
  if ($status[5]=='0') $cor6 = 'lightcoral';
    else $cor6 = 'lightgreen';
  if ($status[6]=='0') $cor7 = 'lightcoral';
    else $cor7 = 'lightgreen';
  if ($status[7]=='0') $cor8 = 'lightcoral';
    else $cor8 = 'lightgreen';
  if ($status[8]=='0') $cor9 = 'lightcoral';
    else $cor9 = 'lightgreen';
  if ($status[9]=='0') $cor10 = 'lightcoral';
    else $cor10 = 'lightgreen';
   if ($status[10]=='0') $cor11 = 'lightcoral';
    else $cor11 = 'lightgreen';
```

```
  if ($status[11]=='0') $cor12 = 'lightcoral';
   else $cor12 = 'lightgreen';
  if ($status[12]=='0') $cor13 = 'lightcoral';
   else $cor13 = 'lightgreen';
  if ($status[13]=='0') $cor14 = 'lightcoral';
   else $cor14 = 'lightgreen';
  if ($status[14]=='0') $cor15 = 'lightcoral';
   else $cor15 = 'lightgreen';
  if ($status[15]=='0') $cor16 = 'lightcoral';
   else $cor16 = 'lightgreen';

  //  echo "Temperatura $temp Cº";

  echo "<form method =\"post\" action=\"Pagina.php\">";
  echo "<input type=\"hidden\" name=\"bits\" value=\"$status\">";
  echo "<button style=\"width:400; background-color: $cor1 ;font: bold 50px Arial\" type = \"Submit\"
Name = \"1\">1</button>      ---------      ";
  echo "<button style=\"width:400; background-color: $cor2 ;font: bold 50px Arial\" type = \"Submit\"
Name = \"2\">2</button></br></br>";
  echo "<button style=\"width:400; background-color: $cor3 ;font: bold 50px Arial\" type = \"Submit\"
Name = \"3\">3</button>      ---------      ";
  echo "<button style=\"width:400; background-color: $cor4 ;font: bold 50px Arial\" type = \"Submit\"
Name = \"4\">4</button></br></br>";

  echo "<button style=\"width:400; background-color: $cor5 ;font: bold 50px Arial\" type = \"Submit\"
Name = \"5\">5</button>      ---------      ";
  echo "<button style=\"width:400; background-color: $cor6 ;font: bold 50px Arial\" type = \"Submit\"
Name = \"6\">6</button></br></br>";
  echo "<button style=\"width:400; background-color: $cor7 ;font: bold 50px Arial\" type = \"Submit\"
Name = \"7\">7</button>      ---------      ";
  echo "<button style=\"width:400; background-color: $cor8 ;font: bold 50px Arial\" type = \"Submit\"
Name = \"8\">8</button></br></br>";
```

```
 // echo "<button style=\"width:400; background-color: $cor9 ;font: bold 50px Arial\" type = \"Submit\" Name = \"9\">9</button>      ---------      ";

 // echo "<button style=\"width:400; background-color: $cor10 ;font: bold 50px Arial\" type = \"Submit\" Name = \"10\">10</button></br></br>";

 // echo "<button style=\"width:400; background-color: $cor11 ;font: bold 50px Arial\" type = \"Submit\" Name = \"11\">11</button>      ---------      ";

 // echo "<button style=\"width:400; background-color: $cor12 ;font: bold 50px Arial\" type = \"Submit\" Name = \"12\">12</button></br></br>";


 // echo "<button style=\"width:400; background-color: $cor13 ;font: bold 50px Arial\" type = \"Submit\" Name = \"13\">13</button>      ---------      ";

//  echo "<button style=\"width:400; background-color: $cor14 ;font: bold 50px Arial\" type = \"Submit\" Name = \"14\">14</button></br></br>";

  echo "<button style=\"width:400; background-color: $cor15 ;font: bold 50px Arial\" type = \"Submit\" Name = \"ApagaTudo\">ApagaTudo</button>      ---------      ";

  echo "<button style=\"width:400; background-color: $cor16 ;font: bold 50px Arial\" type = \"Submit\" Name = \"AcendeTudo\">AcendeTudo</button></br></br></br>";

echo "<button style=\"width:400;font: bold 50px Arial\" type = \"Submit\" Name = \"Lamp\">Lamp</button>      ---------      ";

 echo "<button style=\"width:400;font: bold 50px Arial\" type = \"Submit\" Name = \"Vent\">Vent</button></br></br>";

// echo "<button style=\"width:400;font: bold 50px Arial\" type = \"Submit\" Name = \"Refresh\">Refresh</button></br></br>";

 // echo "</form>";


    echo "$status";

            echo "$msg";

}

// Caso ele não receba o status corretamente, avisa erro.

else { echo "Falha ao receber status da casa."; }

socket_close($sock);

?>


</body>

</html>
```

APÊNDICE B – FLUXOGRAMA DO ALGORITMO




TESTE DE DIMMERS
2
DIMMER LÂMPADA ACIONADO?
NÃO
SIM
ESTÁ NO MÁXIMO?
NÃO
SIM
AUMENTA O BRILHO
DIMINUI O BRILHO
DIMMER VENTILADOR ACIONADO?
SIM
ESTÁ NO MÁXIMO?
NÃO
SIM
AUMENTA A VELOCIDADE
DIMINUI A VELOCIDADE
BOTÃO REFRESH ACIONADO?
SIM
ATUALIZA ESTADOS
1
NÃO

APÊNDICE C – PLANTA DO SOBRADO AUTOMATIZADO