UFPA

AUTOMAÇÃO RESIDENCIAL DE BAIXO CUSTO UTILIZANDO A PLATAFORMA DE PROTOTIPAGEM ELETRÔNICA ARDUINO

TIAGO MACHADO WANZELER

1º SEMESTRE DE 2015

UNIVERSIDADE FEDERAL DO PARÁ – UFPA
CAMPUS DE TUCURUÍ – CAMTUC
FACULDADE DE ENGENHARIA ELÉTRICA – FEE
TUCURUÍ–PARÁ–BRASIL
Tucuruí, 08 de Junho 2015.

AUTOMAÇÃO RESIDENCIAL DE BAIXO CUSTO UTILIZANDO A PLATAFORMA DE PROTOTIPAGEM ELETRÔNICA ARDUINO

TIAGO MACHADO WANZELER

Trabalho apresentado ao colegiado do curso de graduação em Engenharia Elétrica da Universidade Federal do Pará no Campus Universitário de Tucuruí para obtenção do grau de Engenheiro Eletricista.

Orientador:
Professor Dr. Ivaldo Ohana

Tucuruí – PA
2015

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA ELÉTRICA

TÍTULO DO TRABALHO DE CONCLUSAO DE CURSO
AUTOMAÇÃO RESIDENCIAL DE BAIXO CUSTO UTILIZANDO A PLATAFORMA DE PROTOTIPAGEM ELETRÔNICA ARDUINO.

DISCENTE	MATRÍCULA
Tiago Machado Wanzeler	201133940022

BANCA EXAMINADORA	CONDIÇÃO
1. Prof. Dr. Ivaldo Ohana - UFPA	Orientador
2. Prof. Dr. Ewerton Ramos Granhen - UFPA	Membro
3. Eng. Jodenilson Alves Silva - UEPA	Membro
4.	

Data da Defesa: 08/06/15	Hora Início: 14:00	Hora do Término: 15:30

Critérios	Nota dos Avaliadores			
	1	2	3	4
Trabalho escrito (Gramática, clareza, etc.)	10	10	9,0	
Conteúdo técnico	10	10	10,0	
Sequência lógica de apresentação	10	10	10,0	
Administração do tempo	10	10	9,0	
Capacidade de expressão oral	10	10	10,0	
Domínio do tema	10	10	10,0	
Média por avaliador	10	10	9,7	
Média Final	9,9			
Conceito Final	EXCELENTE			

Tucuruí-Pará, 08/06/2015

Ivaldo Ohana
Prof. Dr. Ivaldo Ohana

Ewerton R. Granhen
Prof. Dr. Ewerton Ramos Granhem

Jodenilson Alves Silva
Eng. Jodenilson Alves Silva

Á Deus.

Aos meus pais Rozilene e Elizeu.

Aos meus irmãos e amigos.

AGRADECIMENTOS

Á Deus.

Á minha família, em especial a minha mãe Rozilene Melo Machado por todo amor, dedicação, incentivo e apoio, que nunca me deixou faltar base para continuar e tornar esta conquista possível.

Aos meus irmãos Junior, André e Diego, pelo apoio e admiração fornecidos ao longo dos cinco anos de universidade, e aos quais sou muito grato.

A Universidade Federal do Pará, pela oportunidade.

Ao casal e grandes amigos Juliana e Cesar pelo carinho e ajuda fornecidos em muitos momentos do curso e até mesmo antes.

Ao meu amigo Rodolfo Fonseca pelo apoio fornecido em alguns momentos do curso.

Ao meu amigo Diego Pinheiro pelo incentivo e apoio na realização deste trabalho.

Aos meus amigos da turma de Eng. Elétrica de 2011, em especial aos meus grandes e eternos amigos da Camarilha.

Ao Professor Dr. Karlo Queiroz da Costa, pela oportunidade de participar da bolsa de iniciação científica.

Aos colegas de trabalho da UFPA.

A todos os colegas da UFPA de modo geral.

A todos vocês, meu muito obrigado.

“O mundo não está preparado para isso. É algo muito além de nosso tempo, mas as leis vão prevalecer, e um dia farão um sucesso triunfante.”

Nikola Tesla

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIATURAS

Símbolos	Descrição
AR	Automação Residencial.
DIP	*Domotics Integration Project*.
AURESIDE	Associação Brasileira de Automação Residencial e Predial.
Atmel	Fabricante de microcontroladores.
ATmega328	É um microcontrolador de 8 bits da Atmel.
C/C++	É uma linguagem de programação multiparadigma e de uso geral.
CPU	*Central Processing Unit*
PC	*Personal Computer*
USB	*Universal Serial Bus*
PWM	*Pulse width Modulation*
ICSP	*In-Circuit Serial Programming*
AC	*Alternating current*
DC	*Direct current*
IDE	*Integrated Development Environment*
SRAM	*Static Random Access Memory*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
HTML	*HyperText Markup Language*
SPI	*Serial Peripheral Interface*
MISO	*Master In Slave Out*
MOSI	*Master Out Slave In*
SCK	*Serial Clock*
SS	*Slave Select*
MAC	*Media Access Control*

IP	*Internet Protocol*
E/S	Entrada/Saída
CI	Circuito Integrado
V	*Volts*
ESD	*Surge Protection Diodes*
mA	*Mili Ampère*
RISC	*Reduced Instruction Set Computer* (Computador com um Conjunto Reduzido de Instruções)
KB	*Quilobytes*
RAM	*Random Access Memory*
USART	*Universal Synchrounous/Asynchrounous Receiver/Transmitter*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
PoE	*Power over Ethernet*
SCR	Dispositivo semicondutor NPNP de quatro camadas
Triac	Componente de três terminais, um dos quais controla o fluxo de corrente.
NTC	*Negative Temperature Coefficient*
PTC	*Positive Temperature Coefficient*
OSC	*Open Sound Control*
APP	*Aplicativo*
WLAN	*Wireless Local Area Network*
LM35	*Sensor de Temperatura*
LDR	*Light Dependent Resistor*
PIR	*Passive InfraRed Sensor*
RGB	*Red, Green, Blue*

LISTA DE FIGURAS

SUMÁRIO

RESUMO

Este presente trabalho apresenta a implementação de um protótipo de um sistema de automação residencial de baixo custo, através de dispositivos móveis, utilizando a plataforma de prototipagem eletrônica *Arduino*. Visando facilidade e comodidade para o usuário final, o controle do sistema de automação aqui apresentado pode ser realizado por meio de dispositivos móveis, onde serão disponibilizadas todas as funcionalidades do sistema. Esse sistema de controle e automação através de conjunto de ferramentas e dispositivos tem as seguintes funcionalidades: automatizar todo o sistema de iluminação de uma residência, que vai desde um simples ligar ou desligar uma lâmpada remotamente à dimerização das luzes de determinados ambientes; monitoramento da temperatura; e implementação de sistema de alarme, visando a segurança residencial. Todos os módulos do sistema apresentados neste projeto visam à facilidade e dinamismo ao acesso a determinadas funcionalidades para usuários deficientes ou não. De forma inovadora e atendendo as perspectivas de mercado, bem como atendendo ao requisito de praticidade o sistema será controlado por meio de um *smartphone (iPhone 5S),* que terá acesso a todas as funcionalidades do sistema de automação residencial através de simples toques em tela, bastando apenas estes estarem conectados à rede. As solicitações serão enviadas pelo usuário através de um aplicativo (*App Wanzeller's House*) disponível no *smathphone.* Este aplicativo possui uma interface gráfica personalizada e intuitiva, dividido através de módulos representados por cores. Então, os dados serão recebidos pelo microcontrolador e este tomando as ações necessárias inerentes ao controle e automação da residência, o qual realizará a função desejada. Aliado a este *App*, o binômio custo-benefício foi levado em consideração, pois existe um grande valor agregado nos benefícios oferecidos ao usuário, tais como: praticidade, dinamismo e segurança na execução das tarefas com baixo custo de investimento, este último sendo o principal estímulo em melhorias na automação residencial através desta plataforma *open-source* de prototipação eletrônica baseada na flexibilidade, com *hardware* e *software* fácil de usar e que a cada dia vem crescendo no mundo, que é a plataforma *Arduino.*

Palavras – Chave: Automação Residencial, Arduino, *Ethernet Shield*, *AppWanzellersHouse*.

ABSTRACT

This present work presents the implementation of a prototype of a home automation system low cost through mobile devices using the Arduino electronics prototyping platform. Aiming to ease and convenience to the end user control of home automation system low cost presented here can be done through mobile devices, where all system features will be available. This system of control and automation through set of tools and devices have the following features: automate the entire lighting system of a residence, ranging from a simple turn on or off remotely lamp the dimming of lights in certain environments; temperature monitoring; and alarm system implementation to the residential security. All system modules presented in this project aim at ease and dynamism access to certain features disabled or not users. Innovative and given the market outlook manner, and also to the practical requirement of the system is controlled through a smartphone (iPhone 5S), which will have access to all the features of home automation system through simple touches on the screen, these just by being connected to the network. The requests will be sent by the user via an application (App Wanzeller's House) available smathphone. This application has a personalized and intuitive graphical interface, divided by modules represented by colors. Then, the data is received by the microcontroller and this taking the necessary actions inherent in the control and automation of the residence which will perform the desired function. Allied to this App, cost-effective binomial was taken into account, as there is great value in the benefits offered to the user, such as practicality, dynamism and safety in performing the tasks with low investment costs, the latter being the main encouraging improvements in home automation through this open-source platform for electronic prototyping based on flexibility, with hardware and software easy to use and that every day is growing in the world, which is the Arduino platform.

Key - Words: *Home Automation, Arduino Ethernet Shield, AppWanzellersHouse*

CAPÍTULO 1

INTRODUÇÃO

A lista de funcionalidades da Casa Inteligente é enorme. Por controle remoto ou até mesmo pelo celular, é possível controlar todos os aparelhos eletrodomésticos, ar-condicionados, cafeteiras, microondas, TVs, rádios, máquinas de lavar, além de acender e apagar luzes de todos os cômodos, acionar o sistema de segurança e abrir e fechar portas. Dependendo de quanto o cliente quer investir, o céu é o limite [1].

Essa nova abordagem surge com a Automação Residencial e vem apresentando inúmeros benefícios quando comparamos os sistemas eletrônicos e eletromecânicos integrados com os sistemas isolados, de eficiência limitada. Isso é claramente percebido pelo fato de que os sistemas atuando separadamente pressupõe a existência de infraestruturas dedicadas [2]. É fato que nos últimos anos a Automação Residencial (AR) tem novamente despertado o interesse das pessoas, não só aqueles com formação técnica na área. A computação pessoal e a Internet são as principais responsáveis pela naturalidade com que conversamos sobre tecnologias em nossas casas usando um jargão antes restrito apenas aos analistas de sistemas [3]. Um dos principais fatores que contribuem para a procura dessa tecnologia e para o crescimento desse mercado é o ambiente residencial que permanece praticamente inexplorado para a implementação de sistemas de redes e de controle e que a cada dia vem crescendo de acordo com o avanço da tecnologia implementada nos dispositivos

A Automação Residencial é o conjunto de serviços proporcionados por sistemas tecnológicos integrados como o melhor meio de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação. Nesse contexto, costumamos achar mais adequado o termo “domótica”, largamente empregado na Europa, pois é mais abrangente. Corresponde a utilização das inovações tecnológicas para satisfazer as necessidades e, principalmente, o conforto dos integrantes de determinada habitação. A palavra domótica tem origem na palavra latina “Domus” que significa “Casa”, unida a palavra “Robótica”, que é a automação e controle de qualquer processo [4]. No entanto, no Brasil, optamos pela tradução literal de *home automation*, denominação americana mais restrita, uma vez que, conceitualmente, o termo “automação” não englobaria, por exemplo, sistemas de comunicação ou sonorização.

Segundo a *Domotics Integration Project* (DIP), Domótica ou tecnologia da casa inteligente é a integração dos serviços e tecnologias, aplicados a residências, *flats*,

apartamentos, casas e pequenas construções, com o propósito de automatizá-los e obter aumento em relação à segurança e proteção, conforto, comunicação e gerenciamento técnico [5]. Um dos principais motivos para a automatização residencial é a praticidade e facilidade que ela oferece aos seus usuários. Isto porque o principal fator que define uma instalação residencial automatizada é a integração entre os sistemas aliada à capacidade de executar funções e comandos mediante instruções programáveis ajustadas para determinada finalidade.

Atualmente, observa-se que todos estes aparelhos trabalham independentes e isolados em suas funções. A revolução das redes domésticas e, por consequência, a automação residencial, estão baseadas no fato de permitir a comunicação entre estes dispositivos e controlá-los através de um gerenciador central. A automação permite controlar a residência remotamente, poupar tempo com tarefas repetitivas, economizar energia, dinheiro e aumentar o conforto e segurança [6].

A implementação do protótipo aqui apresentado, que consiste em um sistema de automação residencial de baixo custo. Que leva em consideração a relação custo-benefício que quando alcançada agrega uma série de benefícios ao usuário, tais como: praticidade, dinamismo, segurança e simplicidade na execução das tarefas que hoje geram certo desconforto, e tudo isso aliado ao baixo custo de investimento, este último sendo de fundamental importância e estímulo em melhorias na automação residencial através desta plataforma *open-source* de prototipação eletrônica baseada na flexibilidade, com *hardware* e *software* fácil de usar e que a cada dia vem crescendo no mundo, que é a plataforma de prototipagem eletrônica Arduino.

Atualmente a maneira mais habitual de se acionar uma lâmpada nas residências é através do uso de interruptores, sistema de alarme ainda se faz necessário o usuário estar presente para desativar ou ativar, sistemas de controle da temperatura do ambiente ainda pouco difundidos, no entanto estes procedimentos exigem locomoção e a necessidade de estar no mesmo cômodo ou próximo aos acionadores. Com a crescente popularização e incentivo de produção dos dispositivos móveis, como *tablets* e *smartphones* no mercado nacional, estes podem ser utilizados de maneira inovadora dentro de nossas residências para que possamos desfrutar da praticidade e conforto que a automação residencial pode nos proporcionar [7].

O objetivo deste presente trabalho é a implementação de um protótipo de um sistema de automação residencial de baixo custo utilizando uma plataforma de prototipagem eletrônica bastante intuitiva, fácil acesso e de código aberto. O foco principal do trabalho é exatamente a área de automação residencial.

Visando facilidade e comodidade para o usuário final, o controle do sistema de automação residencial de baixo custo aqui apresentado será realizado por meio de um aplicativo

para dispositivos móveis que foi desenvolvido para gerenciar os dispositivos da residência que serão microcontrolados via *Wi-fi* através do protocolo OSC. Este aplicativo foi desenvolvido através do *software TouchOSC Editor* e recebeu o nome de *App Wanzeller's House*, pois faz referência a residência que foi tomada como base para realização do protótipo e futura implantação real do sistema desenvolvido. Este possui uma aparência simples, agradável de usar e intuitiva, dividido através de módulos representados por cores, onde serão disponibilizadas todas as funcionalidades do sistema. A casa que foi utilizada como referência pelo protótipo possui uma sala, três quartos, um banheiro e uma cozinha.

Esse sistema de controle e automação através de conjunto de ferramentas e dispositivos tem como objetivo principal automatizar a iluminação de uma residência, monitorar e controlar a temperatura de determinados ambientes e implementar um sistema de alarme, provendo facilidade e dinamismo para usuários deficientes ou não. De forma inovadora e atendendo as perspectivas de mercado, bem como atendendo ao requisito de praticidade o sistema poderá ser controlado por dispositivos móveis como *tablets* e *smartphones*. Um dos objetivos específicos do sistema de controle e automação aqui apresentado é automatização do sistema de iluminação, onde através do *app* criado será possível ligar ou desligar lâmpadas de determinado cômodo da casa de forma via *Wi-fi*. Apesar deste módulo apresentar apenas o uso do sistema no acionamento das lâmpadas, o sistema poderia ser expandido para outros tipos de equipamentos elétricos, pois o princípio seria praticamente o mesmo. Aumentando ainda mais a eficiência e eficácia do protótipo proposto.

CAPÍTULO 2

AUTOMAÇÃO NO CONTEXTO HISTÓRICO E SOCIAL

O conceito de casa do futuro vem a algum tempo ganhando espaço nos dias atuais, a partir da década de 20, quando surgiram os primeiros eletrodomésticos nos Estados Unidos, os fabricantes já usavam o termo casa do futuro para promover os benefícios que seus equipamentos iriam trazer para a dona de casa na época. Existia a promessa era que eles iriam poupar o tempo das pessoas executando as tarefas rotineiras e cansativas do lar. A ideia de se pensar em controlar determinados dispositivos, como eletrodomésticos ou a iluminação de uma residência era bastante inspiradora [3]. Nos primeiros módulos "inteligentes", os comandos eram enviados pela própria rede elétrica da residência, no conceito de PLC (*Power Line Carrier*) [8]. Apesar da grande perspectiva de crescimento, eram soluções simples e pontuais, como ligar remotamente determinado equipamento ou até mesmos as luzes.

Com o advento dos computadores pessoais e da internet, a explosão da telefonia móvel e outras tecnologias que ingressaram no mundo pessoal dos consumidores, a aceitação das tecnologias residenciais que poderiam ser utilizadas para automatizá-las passou a ter um forte apelo e grande tendência de mercado. Nas economias mais desenvolvidas, o cenário para as chamadas "casas inteligentes" tem evoluído de maneira muito positiva nos últimos anos. Tem contribuído para isso a crescente popularização de diversas tecnologias, seja pelo aspecto educativo do consumidor, seja pelos preços decrescentes. Além disso, temos um ambiente muito propício para o desenvolvimento dos chamados "sistemas domóticos".

A automatização residencial pode abranger vários sistemas tecnológicos de cada residência, como sistema de segurança e iluminação por exemplo. Com o mercado economicamente ativo, a automação residencial ainda é relativamente muito nova e com grandes tendências de crescimento, inúmeras pesquisas existentes mostram esse grande avanço neste setor.

A AR tem mostrado que a integração de dispositivos eletroeletrônicos e eletromecânicos aumenta consideravelmente os benefícios se comparados com os sistemas isolados, de eficiência limitada, por isso esse crescimento no estudo de novas tecnologias para implementação da AR. Esta é também uma aliada na redução do consumo de recursos como água e energia elétrica, além de trazer maior conforto e segurança aos usuários. No entanto, revendo os últimos 20 anos de história, a AR tem sido vítima da problemática em se desenvolver equipamentos e sistemas para o uso residencial. A automação predial e a industrial adotam

como premissa básica a existência do usuário "padrão" que facilita e direciona o planejamento das instalações e equipamentos a fim de acomodar a maioria dos usuários. No caso da Automação Residencial, o usuário interage e interfere no sistema o tempo todo e isso cria diversos entraves na concepção de um equipamento eletrônico dedicado ao ambiente residencial [3].

Em geral, os projetos das residências convencionais não satisfazem por completo aos anseios dos moradores, o que se constitui num contrassenso, pois a habitação, por atender às necessidades básicas do ser humano como as de proteção, segurança e bem estar, é considerada como um dos bens de consumo de maior importância para a maioria das famílias [9], o que a cada dia a automação residencial vem buscando assegurar aos seus usuários. A moradia, o abrigo, o lar, deve ser prazeroso, eficiente, dignificante e, por ser um bem de grande vida útil, flexível às transformações sociais e tecnológicas. Segundo Dias (2004, p. 137), a Domótica, por meio de seu conjunto multidisciplinar de aplicações, bem integrada às residências, é capaz de aumentar a qualidade de vida de quem nelas habita. Para o idoso, a condição de viver só em sua residência pode ser sua opção, e a Domótica, por meio de suas variadas aplicações, oferece elementos para dar suporte a essa opção, na visão de Camarano (2002, p. 7).

Para a atualidade, a AR vem em bom momento, pois novos requisitos têm surgidos nas habitações, como o aumento da expectativa de vida das pessoas, com isso é comum ter pessoas mais idosas em casa, e outro requisito que deve ser levado em consideração além dos idosos são as pessoas portadoras de alguma deficiência física, uma vez que as pessoas idosas e as portadoras de algum tipo de deficiência física possuem certa limitação de suas atividades, exatamente pelo fato de terem que se movimentar para realizar determinada tarefa. A estes e inúmeros outros fatos a automação residencial apresenta valiosos recursos tecnológicos que podem ser incorporados às instalações domésticas e com isso promoverem, além de conforto e segurança, a redução de barreiras que dificultam as atividades das pessoas que possuem algum tipo de limitação. Um simples acionar de um botão através de uma interface no *tablet* ou *smartphone* pode realizar determinada tarefa que demandaria esforço para ser realizado, levando em consideração a proposta do protótipo proposto que é a implementação de um sistema de automação residencial de baixo custo, agrega-se muito valor ao produto. Vale salientar que a inclusão destes elementos promovem maior independência e contribuem para que as pessoas possam continuar residindo em seu domicílio, aumentando a praticidade e segurança no ambiente residencial.

Com esta grande tendência, segundo um relatório da *Parks Associates* 37% das famílias norte-americanas planejam adquirir um ou mais dispositivos de automação residencial no

próximo ano. A popularidade crescente das vendas de dispositivos inteligentes para casas foi impulsionada pela introdução através da *Apple* e Google de novas soluções no decorrer de 2014. A empresa de pesquisa descobriu que também os varejistas estão se preparando para comercializar dispositivos domésticos inteligentes nas suas prateleiras, itens tais como termômetros *Nest*, lâmpadas *Hue* da *Philips*, fechaduras inteligentes e outros produtos nesta tendência [16]. Veja na figura abaixo um gráfico sobre o que os consumidores pretendiam comprar em 2015.

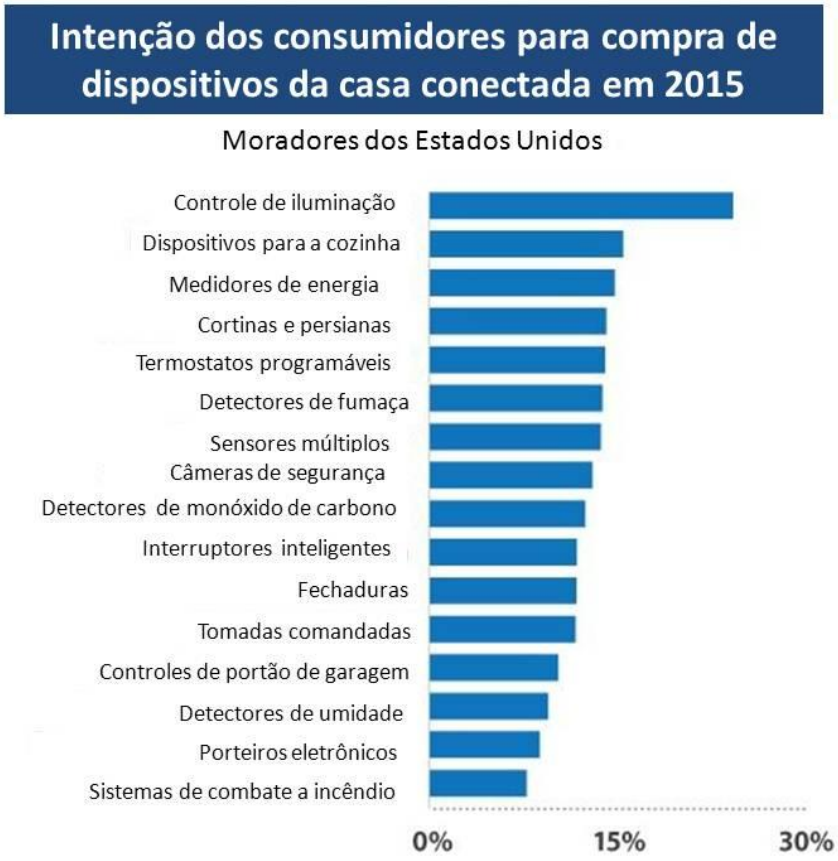


Figura 2.1 - Intenção de Compra dos consumidores para 2015

Fonte: Aureside, 2015

CAPÍTULO 3

BASES METODOLÓGICAS E REFERNCIAL TEÓRICO

O projeto de automação apresentado nesta monografia consiste na implementação de um protótipo de um sistema de automação residencial de baixo custo utilizando a plataforma de prototipagem eletrônica arduino, este projeto possui três funcionalidades básicas, que são: Controle da iluminação da residência, monitoramento da temperatura e implementação de um sistema de alarme residencial, onde os usuários que tenham acesso ao *App Wanzeller's House* poderão acessar o sistema, como mostra a figura 3.1. Contudo, para a implementação desse sistema serão necessários uma série de conhecimentos acerca do assunto (domótica) e dos componentes e métodos que foram utilizados. Sendo assim, neste capítulo será apresentada toda fundamentação teórica para o desenvolvimento desta monografia, bem como um aprofundamento dos componentes que foram utilizados e que merecem destaque e maior compreensão.

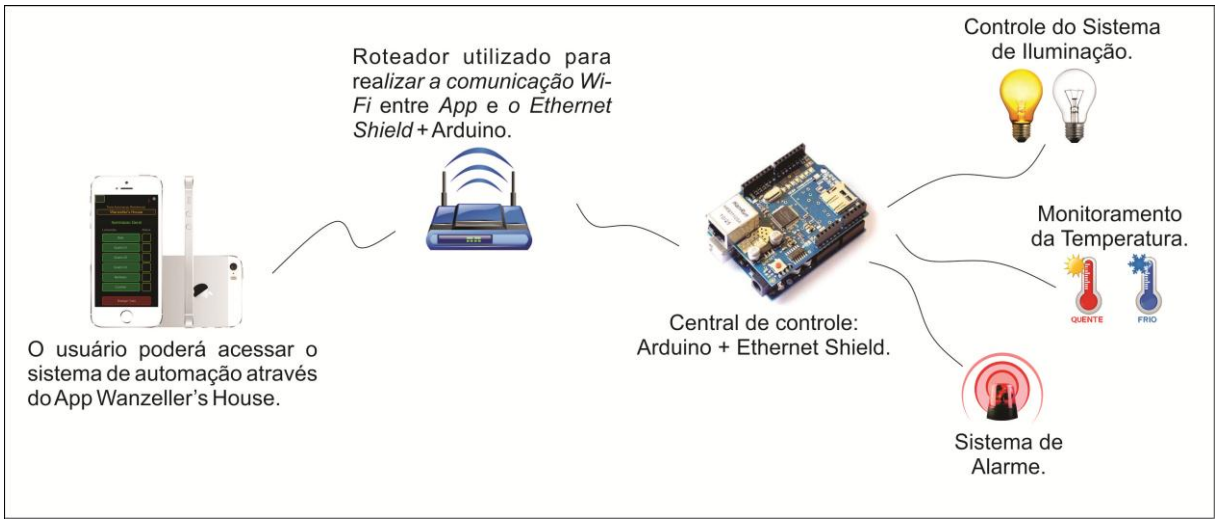


Figura 3.1 - Visão geral do funcionamento do sistema

Fonte: Adaptada pelo autor

3.1. Domótica

O termo domótica deriva da junção da palavra em latim *domus*, que significa casa e a palavra robótica, ou seja, domótica torna possível controlar de forma automática uma residência. (FERREIRA, 2008).

O conceito de domótica segundo Bolzani:

> É a ciência moderna de engenharia das instalações em sistemas prediais. A Domótica é uma ciência multidisciplinar que estuda a relação entre homem e casa. A imersão de pessoas em ambientes computacionalmente ativos revelou a necessidade do uso de técnicas mais sutis que gerenciassem a complexidade e o dinamismo das interações

> dos moderadores com o ambiente residencial saturado de diminutos dispositivos eletrônicos interligados a rede. (BOLZANI, 2010, p. 31).

A automação residencial, assim como a predial, é derivada da automação industrial, porém, com tecnologias adequadas para a realidade de uma residência, onde na maioria das vezes não há espaço suficiente para grandes centrais de controle e pesados sistemas de cabeamento (SENA, 2005).

3.1.1. História da Domótica

Segundo Moya e Tejedo (2010), a origem da domótica remete-se aos anos 70, quando surgiram os primeiros dispositivos de automação de edifícios, baseados na tecnologia X-10. O X-10 é um protocolo que permite controlar dispositivos em uma rede elétrica já existente, evitando a necessidade de novos cabeamentos. É um sistema de fácil instalação, mas bastante instável, uma vez os componentes podem vir a falhar, ou até mesmo estragar devido à falta de energia ou até mesmo por descargas eletromagnéticas.

Nos anos 80, com o advento da computação pessoal, surgiram interfaces gráficas e operações muito mais simples que as já existentes, o que levou ao surgimento de novas possibilidades de automação residencial. Mas foi no final da década de 90 que uma vasta gama de novidades surgiu, incorporadas com os telefones celulares e a *web*.

Atualmente a domótica utiliza esses novos recursos de forma integrada, como por exemplo, em sistemas de controle e monitoramento móvel, através de celulares, *tablets*, ou via *web* (OLIVEIRA, 2012). A figura 3.2 apresenta a evolução da domótica segundo a AURESIDE (Associação Brasileira de Automação Residencial e Predial).

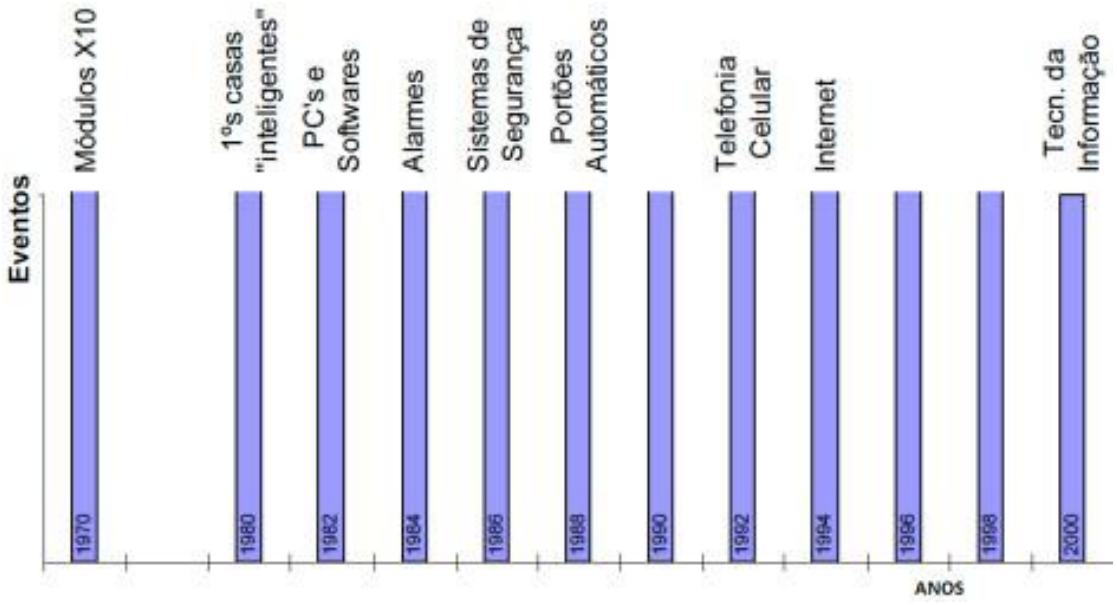


Figura 3.2 - Evolução da Domótica

Fonte: AURESIDE, 2003

3.1.2. Desafios da Domótica

> A próxima grande disputa das empresas de tecnologia será pela automação residencial, um mercado que deve movimentar 250 bilhões de dólares nos Estados Unidos nos próximos sete anos, e 1 trilhão de dólares em todo o mundo no mesmo período. Gigantes como Intel, Motorola, Microsoft e Cisco estão, neste momento, empenhados em criar a casa do futuro. Mas o maior problema é exatamente este: até agora, ninguém sabe o que ela precisa ter para agradar aos consumidores (EXAME 2005).

De acordo com pesquisas da Associação Brasileira de Automação Residencial, a busca por soluções de automação está em diferentes estágios nas regiões do Brasil. Em São Paulo, por exemplo, os interessados já não são mais os entusiastas ou visionários, mas também os pragmáticos e cidadãos comuns de classe média (TURUEL, 2008).

Para um produto ser mantido no mercado ele precisa de um número significativo de compradores, que seria atingido quando uma maioria inicial começasse a adquiri-lo, pois se isso não ocorre, o produto acaba caindo no esquecimento. O gráfico abaixo ilustra como é a curva de adoção de novas tecnologias.

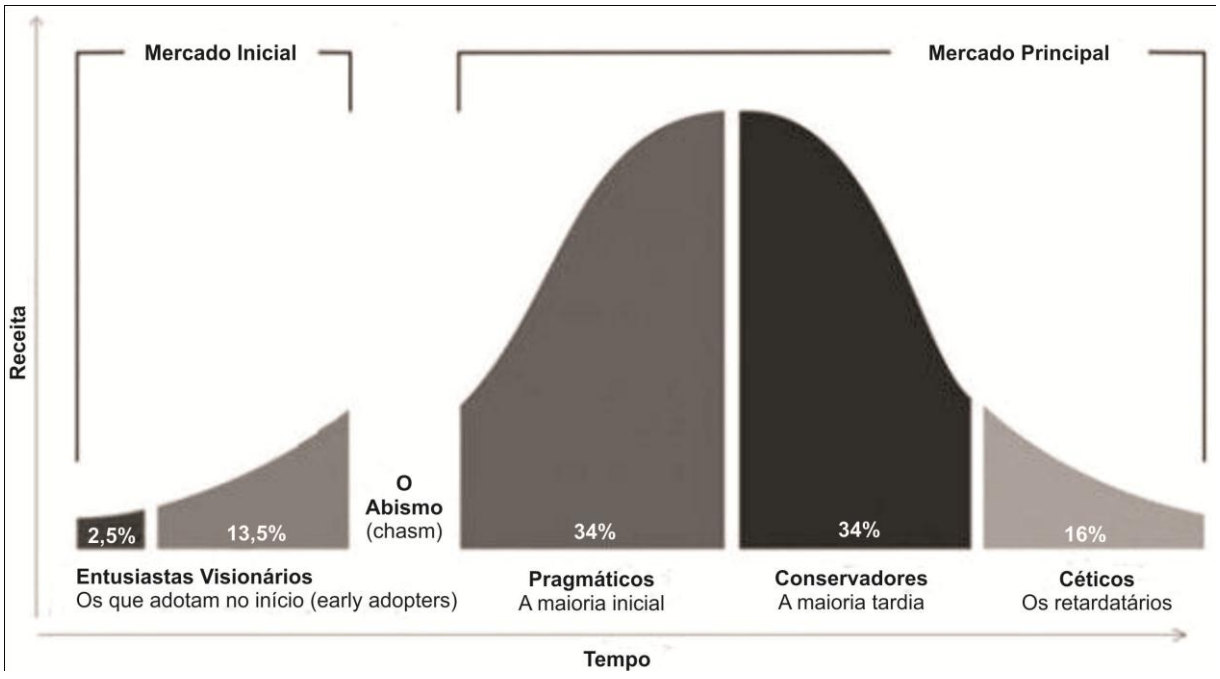


Figura 3.3 - Ciclo de adoção de novos produtos pelos consumidores

Fonte: GRANDO, 2013

A domótica já está deixando para trás esse abismo que existe entre consumidores visionários e pragmáticos. No entanto, um dos grandes desafios que podem ser encontrados para o desenvolvimento da automação residencial, é a resistência dos próprios arquitetos e engenheiros eletricistas, seja por falta de conhecimento, por medo de novas tecnologias ou por necessidade da mudança da forma de trabalhar.

3.1.3. Características de Sistemas Domóticos

No mercado já existem diversas opções de sistemas de domótica, alguns mais simples que executam funções básicas como, por exemplo, controle de iluminação e controle de portas e portões, outros já permitem controlar até mesmo persianas, sons ambientes, temperatura da água da piscina e da banheira, entre outras diversas funcionalidades.

Entretanto, segundo a AURESIDE, algumas características são essenciais a qualquer sistema, como por exemplo:

a) Capacidade de integrar todos os sistemas: os sistemas devem ser interligados através de uma rede, e permitir controle através de uma única interface;

b) Atuação em condições variadas: o sistema deve ser capaz de atuar em condições adversas, como interrupções de energia, climas diversificados entre outros;

c) Fácil relação com o usuário: os usuários muitas vezes não compreendem programações complexas, portanto, deve haver um sistema com interface intuitiva;

d) Monitoramento: o monitoramento desse tipo de sistema é algo crítico, portanto devem ser realizadas auditorias com determinada frequência, e sempre observar relatórios de controle.

3.1.4. Benefícios proporcionados por sistemas domóticos

Os benefícios que podem ser observados imediatamente pelos moradores de uma casa automatizada são:

a) Economia de energia: a energia é utilizada somente quando necessário, pois o controle da intensidade de iluminação, sensores de presença, controle da temperatura ambiente, elimina gastos desnecessários;

b) Conforto: ajuste de temperatura de piscinas, filtros de ar, ar condicionado, entre outros equipamentos através de uma única interface;

c) Conveniência: a temperatura do ambiente pode ser controlada mesmo antes da chegada dos moradores, lâmpadas e portões podem ser controlados de qualquer local;

d) Acessibilidade: sistemas de automação com dispositivos *touchpad* e com reconhecimento de voz proporcionam a portadores de necessidades especiais, a possibilidade de controlar o ambiente que estão, seja ascendendo uma lâmpada, controlando luminárias, televisores, portas, entre outros, devolvendo ao indivíduo sua independência;

e) Segurança: a possibilidade de controle de luminosidade, ar condicionado, televisores e outros dispositivos, podem fazer que a residência pareça sempre ocupada.

Câmeras de monitoramento podem ser acessadas de qualquer local, o que permite que a casa esteja sempre monitorada, um exemplo citado por LAGUÁRDIA, foi o caso do

empresário Ronan Soares. O empresário estava na cidade de Colônia, na Alemanha, ao verificar a câmera de monitoramento de sua residência, percebeu que havia ladrões invadindo o local, rapidamente avisou a esposa que estava no Brasil, que praticamente no mesmo instante acionou a polícia.

3.1.5. Mercado da domótica

A automação residencial, ou domótica, como também é conhecida, está em uma fase de muito crescimento no Brasil, nos últimos quatro anos o serviço cresceu 300%, e já existem 25 empresas do setor no país (AURESIDE).

Dados da Associação Brasileira de Automação Residencial revelam que 300 mil residências do Brasil possuem algum tipo de automação, e este número pode vir a crescer muito mais, pois de acordo com pesquisa realizada pela associação, 78% dos brasileiros possuem interesse nesse serviço, um número maior que a média mundial, 66%. Segundo a AURESIDE em 2012 o segmento faturou R$ 4 bilhões e com perspectivas de crescimento de mais de 30% em 2013.

E ainda segundo a AUREDISE o mercado de automação residencial deve atingir 50 bilhões de dólares. Pois, fabricantes de dispositivos móveis, com a *Samsung* e a *Apple*, estão interessadas no mercado de automação residencial com o objetivo de tornar seus aparelhos ainda mais indispensáveis para os usuários. Além disso, a entrada neste mercado pode render bons frutos em longo prazo para a receita dessas empresas. De acordo com a *Strategy Analytics*, o mercado de automação residencial deve atingir um faturamento de 50 bilhões de dólares ainda em 2014 e o volume de negócios deve dobrar até 2018 [10].

3.1.6. Sistemas Domóticos Comercializados

Realizou-se uma pesquisa de sistemas de domótica que estão disponíveis para comercialização no Brasil, observou-se que em todos o principal objeto de automação é sistema de iluminação, como se pode verificar nos sub tópicos a seguir.

3.1.6.1. IHouse

Segundo informações retiradas do próprio site da empresa [11] a *iHouse* foi a pioneira no Brasil a lançar empreendimentos com tecnologia de automação residencial embarcada. Em 2006, uma matéria na revista Exame, com o título “As casas do futuro já estão à venda”, anunciava: “... a primeira safra de apartamentos do futuro começa a se tornar realidade no mês de maio, no bairro do Itaim, zona sul de São Paulo”. Tratava-se do *Brazilian Art*, um projeto com todas as características de um alto padrão, em um dos endereços mais sofisticados da cidade e com o objetivo de também conquistar os aficionados por tecnologia. Empreendimentos

de luxo que foram lançados em várias cidades do país pelas grandes incorporadoras parceiras da *iHouse*.

Mas, se no começo apenas o altíssimo padrão podia receber essa tecnologia, em 2010 a situação começou a mudar, uma série de novos empreendimentos agora voltados para o médio alto padrão foram lançados no mercado e, mais uma vez, graças ao pioneirismo da *iHouse* com o lançamento de pacotes de produtos para essa faixa do mercado. Em 2012, a iHouse lança uma linha de produtos *premium* com tecnologia *wireless* que permite a automação de residências e escritórios sem necessidade de infraestrutura.

O ícone dessa nova safra é o premiado *Wallpad*: É um sistema que promete realizar a automação residencial sem obras ou mudanças na infraestrutura da residência. Proporciona o controle da iluminação, com graduação entre 0 e 100% e apenas um toque para desligar todas a iluminação se for sair da casa, comando de ar condicionado, persianas e ajuste de fluxo e temperatura da água para banho [12]. Uma desvantagem deste sistema é que não proporciona acesso fora da residência, uma vez que o controle e programação se dão somente através do painel do *Wallpad* (um controle que se comunica através de sinal *wireless* com os módulos instalados), este ainda possui *design* sofisticado como ilustrado na figura abaixo.

Figura 3.4 - *Wallpad IHouse*

Fonte: *IHouse*, 2015

3.1.6.2. GDS Automação

Segundo informações retiradas do próprio *site* da empresa [13], a GDS Automação está no mercado a mais de 6 anos e presta serviços e cria soluções de engenharia para diversos ambientes. Propõe soluções para todo e qualquer tipo de ambiente, como casas, apartamentos, escritórios, prédios, teatros, hotéis, salas de conferência, auditórios, galpões, sítios, fazendas e

diversos outros, de acordo com a imaginação do cliente. O resultado final segundo a própria empresa é conforto, beleza e segurança, juntamente com o valor agregado ao local.

A GDS oferece um sistema de automação residencial que proporciona o controle de iluminação, som, *home theater*, ar condicionado e sistemas de irrigação de jardins, através de *tablets* ou celulares. O projeto é desenvolvido de acordo com o desejo e a necessidade do cliente, sendo a interface personalizável para cada projeto. Abaixo a ilustração da tela de controle de uma sala.

Figura 3.5 - Controle de Iluminação do Sistema da GDS Automação

Fonte: GDS Automação, 2015

3.1.6.3. QualiHouse

Segundo informações retiradas do próprio site da empresa [14], a *QualiHouse* Automação Predial surgiu da inquietação de dois engenheiros eletrônicos com desejos comuns: desenvolver tecnologia de ponta na área de automação com foco na melhoria de vida das pessoas. Hoje, a *QualiHouse* é uma empresa com mais de sete anos de mercado e tem utilizado sua expertise no desenvolvimento de equipamentos eletrônicos e *softwares* para conceber e materializar produtos inovadores na área de automação. A empresa ainda tem experiência no desenvolvimento de projetos de infraestrutura nas áreas de cabeamento estruturado, segurança e automação.

A Q*ualiHouse* também tem um forte produto na área de automação residencial, o *Simplifies*. Segundo a própria empresa o sistema de automação residencial *Simplifies* é a solução adequada para residências, condomínios e escritórios interessados em melhorias de conforto, segurança e economia de energia. O sistema é composto pelos módulos de automação, pelo

servidor *SimpleHome*, por sensores, atuadores, câmeras e ativos de rede (*switch*, roteador, etc.). Assim, através de computador, *tablet*, *smartTV* e *smartphones*, na rede local ou internet é possível monitorar e controlar os diversos dispositivos conectados ao sistema como: lâmpadas, condicionadores de ar, cortinas, motores, fechaduras magnéticas, sirenes e sensores.

O *Simplifies* prima pela simplicidade de uso. O controle dos ambientes automatizados é feito através de uma interface *web* simples e intuitiva que permite que o acesso seja feito através de dispositivos com navegador de internet [15]. A figura abaixo apresenta a tela de controle do *Simplifies* que foi obtida pela apresentação do sistema disponibilizada pela empresa em seu sitio.

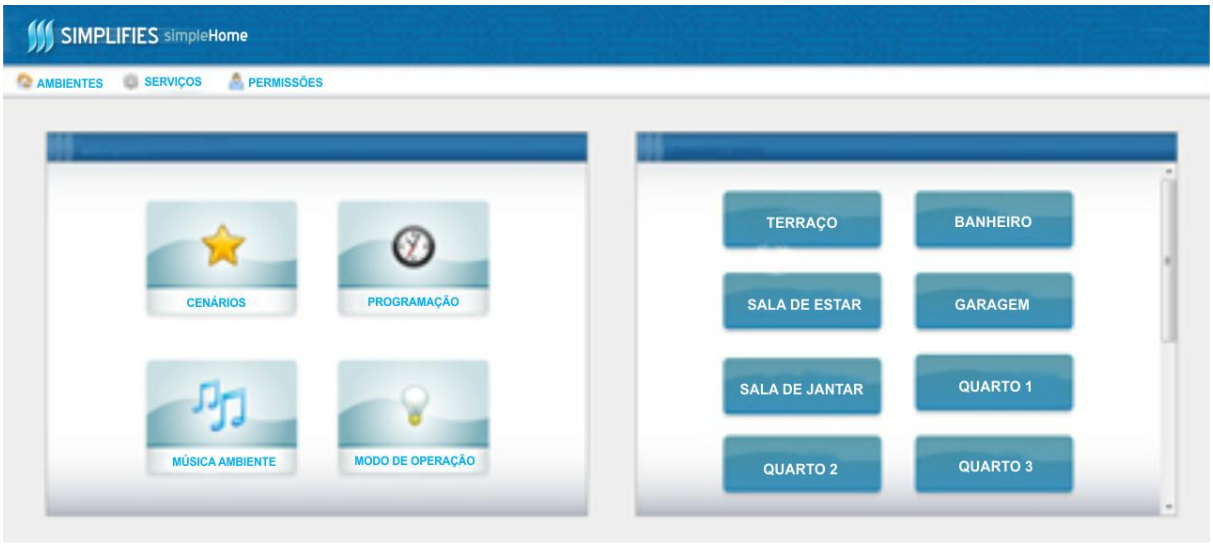


Figura 3.6 - Tela de controle do Sistema Simplifies

Fonte: *QualiHome*, 2015

3.2. Plataforma Arduino

Os microcontroladores, por serem pequenos e apresentarem a melhor relação custo/benefício, estão presentes em quase tudo que envolve a eletrônica. Eles são dispositivos que possuem internamente todos os componentes necessários para seu funcionamento autônomo. Sendo assim, para trabalhar como o "cérebro" do projeto proposto, foi escolhida a plataforma de prototipagem eletrônica Arduino, devido ao fato de ser uma placa robusta, de baixo custo e com linguagem de programação de fácil aprendizagem. A seguir, será apresentado sua e história e como consiste seu funcionamento.

3.2.1. O que é o Arduino?

Segundo informações do site oficial do arduino, esta é uma plataforma *open-source* de prototipagem eletrônica que integra flexibilidade visando facilitar o uso tanto do hardware e do

software (ARDUINO, 2015). Ele é constituído por uma placa única com suporte de entrada/saída, pode captar informações do ambiente através da porta de entrada que permite integrar atuadores com o meio externo, por exemplo, sensores. O microcontrolador na placa do Arduino é um Atmel AVR de 8 *bits* programado usando a linguagem de programação padrão, essencialmente utiliza-se C/C++ para enviar os comandos ao Arduino. Projetos do Arduino podem ser *stand-alone*, ou seja, possuem o código já compilado em seu chip ou podem comunicar com *software* rodando em um computador. Por utilizar os microcontroladores da família AVR [5], essa plataforma é composta por inúmeras versões. Algumas listadas na tabela abaixo:

Tabela 1 - Comparação entre modelos da plataforma arduino.

	Arduino Uno	**Arduino Mega 2560**	**Arduino Mega 1280**
Microcontrolador	ATmega328	ATmega2560	ATmega1280
Nº de portas de E/S digital	14 (sendo que 6 podem ser usadas como PWM)	54 (sendo que 15 podem ser usadas como PWM)	54 (sendo que 14 podem ser usadas como PWM)
Nº de portas de entrada analógica	6	16	16
Flash Memory	32 KB	256 KB	128 KB
Clock Speed	16 MHz	16 MHz	16 MHz
Bootloader	0.5 KB	8 KB	4 KB

Fonte: Arduino (Arduino Uno; Arduino Mega 2560; Arduino Mega 1280, 2015).

O Arduino possui uma interface serial ou USB para interligá-lo a outras placas ou sistemas. Essa interface permite que o Arduino seja programado e/ou que interaja com o ambiente em tempo real. Um importante aspecto é a maneira padrão que os conectores são expostos, permitindo a CPU ser interligado a outros módulos expansivos, conhecidos como *Shields*, que incorporam funções que o Arduino por si só não as possui.

3.2.2. História do Arduino

O Arduino é um projeto iniciado em 2005, na cidade de Ivre, na Itália, pelo professor Massimo Banzi. Este professor queria ensinar eletrônica e programação de computadores para

seus alunos do curso de design, porém encontrava muita dificuldade, uma vez que não eram alunos da área de programação ou eletrônica, e devido à inexistência de placas robustas e de baixo custo no mercado.

Pensando nestes problemas, Massimo Banzi e o engenheiro eletrônico David Cuartielles decidiram projetar sua própria placa, juntamente com a ajuda de um aluno de Massimo, David Mellis, que ficou responsável por desenvolver a linguagem de programação da plataforma.

Gianluca Martino foi o responsável por criar o protótipo comercial para fabricação em grande escala. A princípio foram fabricadas somente duzentas placas, vendidas a escolas com o lucro de cerca de um euro por cada. Meses depois, a empresa americana *Sparkfun*, decidiu comercializar o projeto, começando com 20 placas, em 2010 estima-se que foram vendidas 40.000. Assim iniciou a história do Arduino, que atualmente é utilizado no mundo inteiro, nos mais diversos tipos de projetos.

3.2.3. Ambiente de Desenvolvimento (IDE)

O ambiente de desenvolvimento consiste em um software gratuito, onde será escrito a sequência de instruções que serão interpretadas pelo Arduino. Ele se conecta ao *hardware* para realizar a comunicação e carregar o código desenvolvido. Os códigos escritos neste ambiente de desenvolvimento são chamados de *Sketches*, que são salvos com a extensão .INO. Quando se inicia a IDE (*Integrated Development Environment*) encontra-se a área para escrever o software, a barra de ferramentas, o console de textos, que exibe uma lista completa de erros no código e o resultado das instruções enviadas ao Arduino, e os seguintes botões descritos na tabela a seguir:

Tabela 2 - Descrição dos botões da IDE do arduino.

Verify:	Tem a função de verificar erros no código.
Upload:	Compila o código e carrega para a placa.
New:	Cria um novo esboço.
Open:	Apresenta um menu com vários códigos prontos.
Save:	Salva o *sketch* atual.

Há também alguns comandos adicionais oferecidos para facilitar o desenvolvimento, como por exemplo: *Copy for* fórum, encontrado dentro do menu editar, que torna possível copiar o código para postar em fóruns, *Copy as HTML*, oferece opção de copiar como HTML para inserir em páginas *Web*, *Import Library*, que adiciona uma biblioteca ao projeto, *Examples*, onde encontra-se diversos exemplos de código prontos, entre outras opções que tornam essa

ferramenta muito simples de trabalhar. A IDE do arduino é mostrada na figura abaixo, o *Sketche* contido na imagem da IDE é o código de exmplo *Blink*:

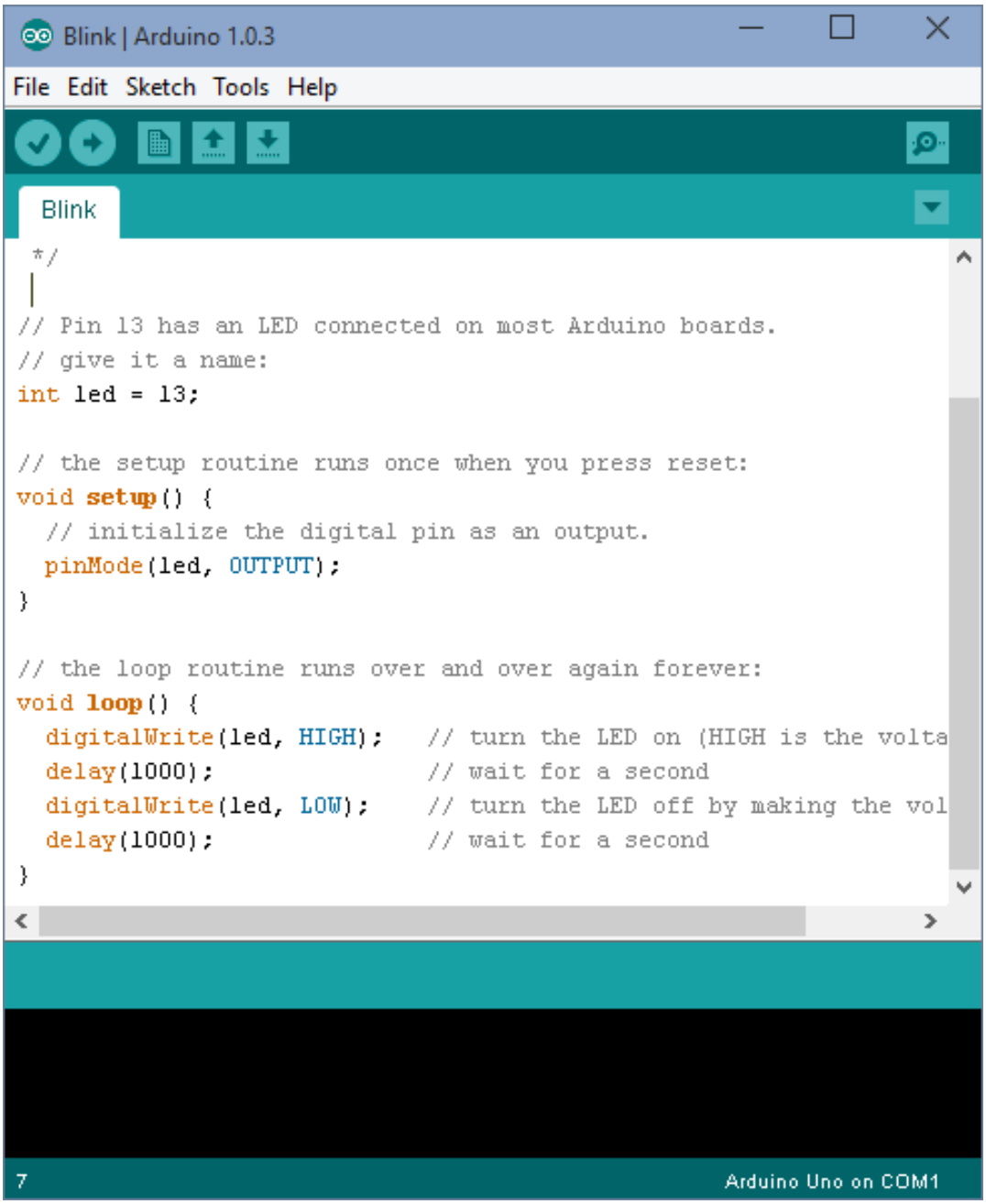


Figura 3.7 - IDE do Arduino

Fonte: Próprio Autor

3.2.4. Bibliotecas Arduino

As bibliotecas são conjuntos de códigos disponibilizados pela desenvolvedora do projeto Arduino, que têm por objetivo facilitar a comunicação com os componentes acoplados à placa. Existem diversas bibliotecas disponíveis, algumas são internas, como por exemplo: *Ethernet*, que foi utilizada no desenvolvimento do projeto de automação residencial aqui

apresentado, outras são disponibilizadas para *download* e podem ser instaladas muito facilmente.

3.2.4.1. Biblioteca SPI

A biblioteca SPI permite a comunicação entre dispositivos de SPI (*Serial Peripheral Interface*) e o Arduino. O SPI é um protocolo de dados seriais síncronos, geralmente utilizado em microcontroladores para comunicação entre um ou mais periféricos. Na comunicação SPI sempre há um periférico mestre, neste caso, o Arduino é o mestre e os demais periféricos, como o módulo *Ethernet*, são escravos. Nessa comunicação há quatro conexões:

Tabela 3 – Conexões Mestre Escravo (SPI)

MISO (Master In Slave Out):	Dados do *Slave* para o *Master;*
MOSI (Master Out Slave In):	Dados do *Master* para o *Slave;*
SCK (Serial Clock):	*Clock* de sincronização para transmissão de dados entre o *Master* e *Slave;*
Hardwate SS (Slave Select):	Seleciona qual *Slave* receberá os dados

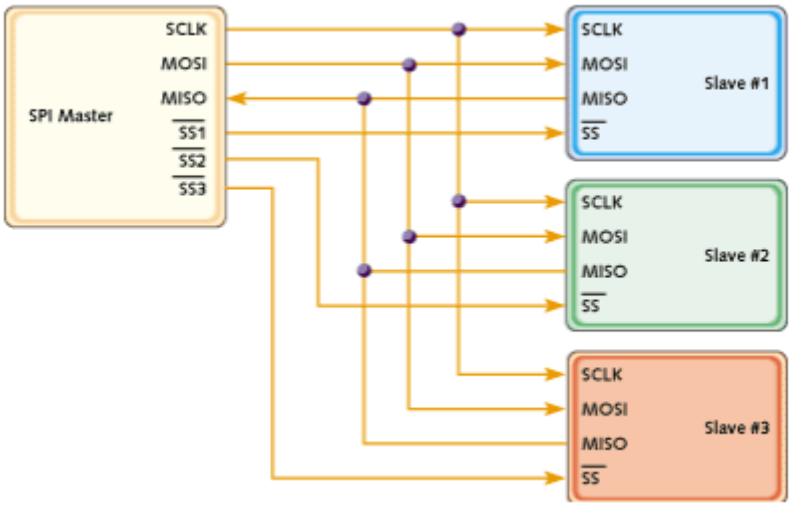


Figura 3.8 - Protocolo de Comunicação SPI

Fonte: Blog Software Livre, 2015

O Arduino Uno se comunica com o *Shield Ethernet* usando o barramento SPI utilizados as saídas digitais 11, 12 e 13. O pino 10 é utilizado como SS para o *Ethernet* e o pino 4 como SS para o leitor de micro-SD. Como mostrado na figura a seguir:

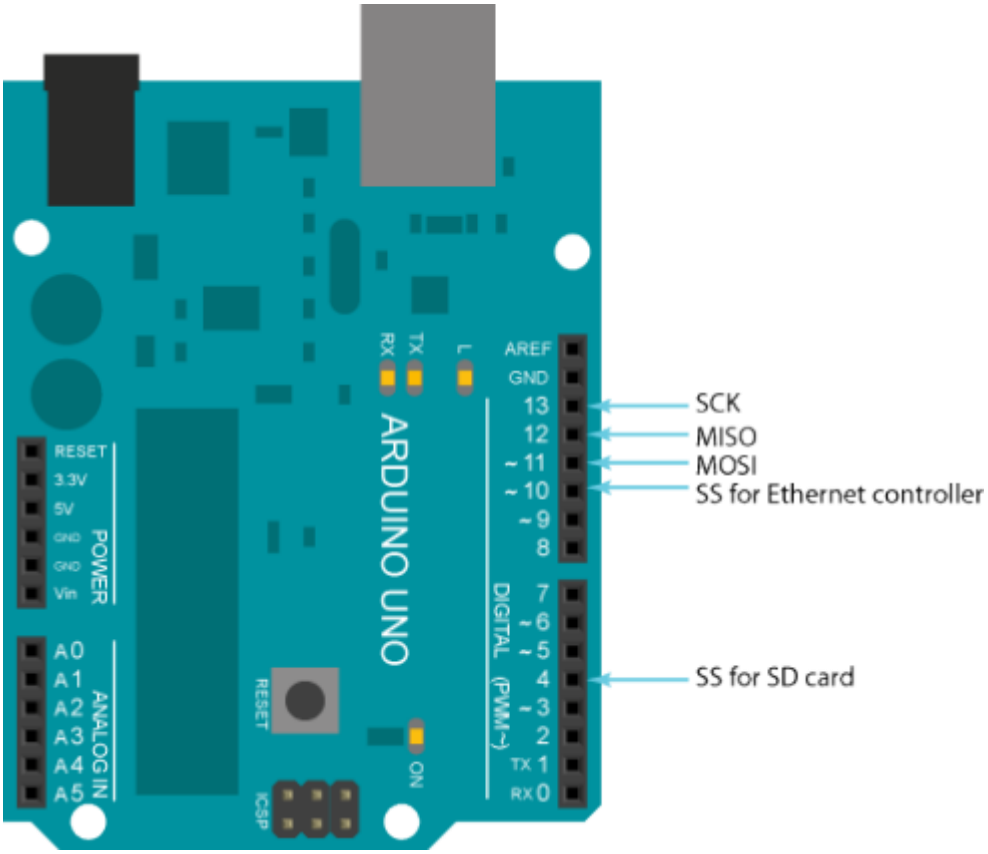


Figura 3.9 - Pinos do Arduino Uno utilizados para comunicação SPI

Fonte: Blog Web das Coisas, 2013

3.2.4.2. Biblioteca Ethernet

Para tornar possível o Arduino se conectar à internet é necessário acoplar a ele um módulo *ethernet (Ethernet Shield)*. E, para uma fácil comunicação entre estes dois componentes utiliza-se uma biblioteca chamada *ethernet*. Para iniciar a biblioteca *Ethernet* é chamado o método *Ethernet.begin*, onde deve-se passar o parâmetros *mac*, IP, *gatway* e *subnet* para que o arduino possa se conectar a rede.

3.2.4.3. Biblioteca ARDOSC

Com a utilização do *OSC Protocol* para comunição, faz-se necessário a utilização da biblioteca ARDOSC, esta trabalha em conjunto com a biblioteca *Ethernet* oficial do Arduino. Sendo assim, qualquer *shield ethernet* baseado no chip W5100 deve usar esta biblioteca para que o sistema funcione sem problemas. A versão desta biblioteca que foi utilizada neste projeto é compatível com a IDE 1.0 do Arduino, sendo por este motivo que foi utilizada esta versão da IDE do arduino para que não viesse ocorrer algum erro durante a execução e gravação do código.

A biblioteca ARDOSC torna bastante simples a configuração da comunicação OSC no Arduino, pois além das definições de interface de comunicação, temos apenas que definir quais são as rotinas que devem ser executadas para cada comando OSC recebido.

3.2.4.4. Biblioteca Twitter

Segundo informações do site oficial da plataforma arduino [25], *Twitter Library* é uma biblioteca para Arduino para envio de *tweets* via arduino com *Ethernet Shield* acoplado a este, ou seja, bast seu Arduino estar conectado à internet para tornar possível o envio de *tweets*. Esta função é extremamente útil em diversas situações, para este projeto o envio de *tweets* automáticos foi utilizado para o envio de notificação em tempo real quando o sistema de alarme de residência for violado. Vale lembrar que os *tweets* são enviados através de um servidor compartilhado. Sendo assim, seguindo orientações do próprio desenvolvedor deve-se evitar o envio de mais de um pedido por minuto não sobrecarregar o servidor.

3.3. Mecanismos de Comunicação do Sistema

O protótipo do sistema de automação residencial apresentado neste projeto, baseia-se na comunicação sem fio. Esta comunicação faz o uso do protocolo OSC, que neste projeto em questão se dá pelo fato do aplicativo *App Wanzeller's House* desenvolvido através da aplicação *TouchOSC Editor* se comunicar com a central de controle utilizando este protocolo de comunicação via *Wi-fi*. Ou seja, este protocolo é utilizado para enviar e receber mensagens entre a aplicação *TouchOSC* (*App Wanzeller's House*) e a central de controle, sendo esta última formada pelo arduino (responsável pelos acionamentos) e acoplado a este um *Ethernet Shield* responsável por conectar o arduino na rede.

Sendo, que a aplicação responsável pelo envio das informações é identificada por um endereço IP e por uma porta (192.168.0.120:8000). As mensagens podem ser enviadas como inteiros, *floats* e *string*. E é sempre iniciada por "/", por exemplo a mensagem contendo o nome botão pressionado e o seu valor deverá ser /ard/nomedobotaovalor, sendo enviada com dois valores, uma *string* e um inteiro, logo os dados enviado via *Wi-fi* e recebidos pelo arduino para tratamento das informações.

3.3.1. OSC Protocol

Open Sound Control (OSC) é um protocolo de comunicação entre computadores, sintetizadores de som e outros dispositivos de multimídia que é otimizado para a tecnologia de

rede moderna. Trazendo os benefícios da tecnologia de rede moderna para o mundo dos instrumentos musicais eletrônicos, vantagens do OSC incluem a interoperabilidade, a precisão, flexibilidade e reforço da organização e documentação. Este protocolo simples, oferece tudo o necessário para o controle em tempo real de som e outros meios de processamento mantendo-se flexível e fácil de implementar.

Características:

- ✓ *Open-ended*, dinâmico, de estilo URL esquema de nomeação simbólica.
- ✓ Simbólico e de alta resolução argumento numérico de dados.
- ✓ Idioma correspondente padrão para especificar vários destinatários de uma única mensagem.
- ✓ Tags de tempo de alta resolução.
- ✓ "pacotes" de mensagens, cujos efeitos devem ocorrer simultaneamente.
- ✓ Sistema de consulta para localizar dinamicamente os recursos de um servidor OSC e obter documentação.

Existem dezenas de implementações de OSC, incluindo som em tempo real e ambientes de processamento de mídia, ferramentas de interatividade *web*, sintetizadores de *software*, uma grande variedade linguagens de programação, e dispositivos de hardware para a medição do sensor. OSC alcançou ampla utilização em áreas como novas interfaces baseados em computador para a expressão musical, de área ampla e de área local em rede de sistemas distribuídos de música, comunicação inter-processo, e até mesmo dentro de uma única aplicação [22].

3.3.2. TouchOSC Editor

O *TouchOSC Editor* é o software que foi utilizado para criação e personalização da interface gráfica que utilizada neste projeto, no qual foi definido como *App Wanzeller's House.* Todos os *layouts* TouchOSC só podem ser criados e modificados com esta aplicação, que é grátis e compatível com Mac OS X, Windows e Linux. O *TouchOSC Editor* pode ser baixado direto do site do desenvolvedor. Vale salientar que a aplicação *editor* é escrito em Java e precisa que do Java para funcionar, este pode ser baixado e instalado a partir do site java.com.

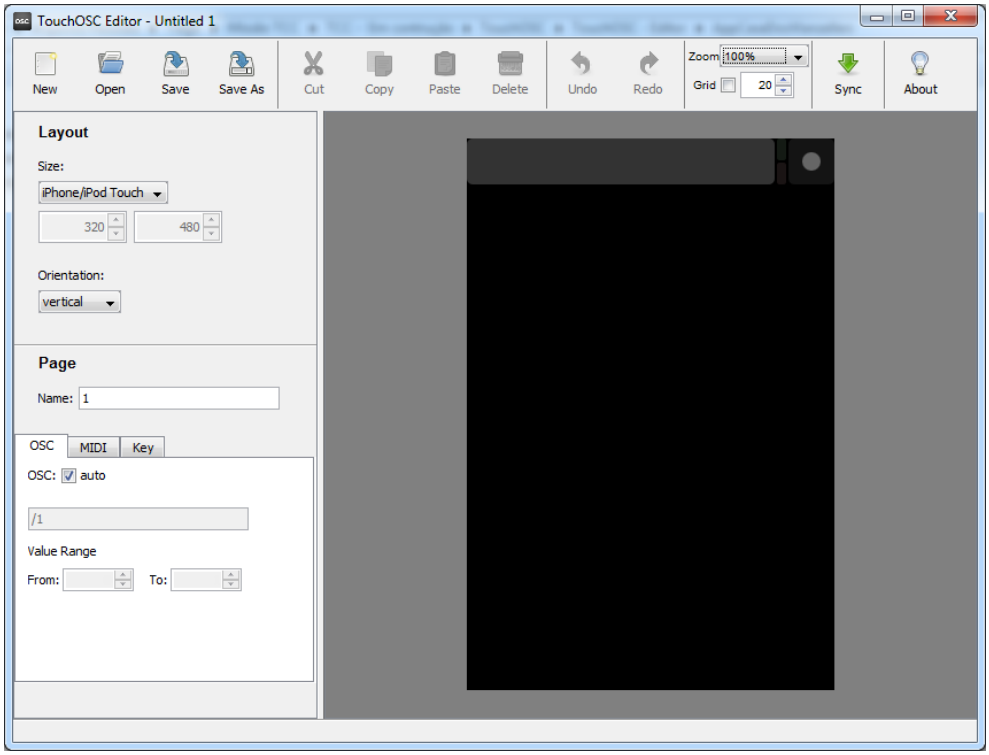


Figura 3.10 - Tela do *TouchOSC Editor*

Fonte: Próprio autor

Ao executar o *software TouchOSC Editor* será exibida a figura acima. Observa-se a partir desta a barra de ferramentas de edição, para a direita é o ponto de vista de edição com uma pré-visualização do layout e para a esquerda, o painel de propriedades que exibe informações contextuais e propriedades de qualquer objeto que está atualmente selecionado no modo de exibição de edição.

3.3.3. Wi-fi

Wi-Fi é um conjunto de especificações para redes locais sem fio (WLAN - *Wireless Local Area Network*) baseada no padrão IEEE 802.11. O nome *Wi-Fi* é tido como uma abreviatura do termo inglês "*Wireless Fidelity*", embora a *Wi-Fi Alliance*, entidade responsável principalmente pelo licenciamento de produtos baseados na tecnologia, nunca tenha afirmado tal conclusão.

A tecnologia *wi-fi* foi utilizada por possibilitar implementar a comunicação e conexão sem fio entre computadores e outros dispositivos compatíveis na rede, como *notebooks, tablets e smartphones*, que estejam próximos geograficamente. Neste projeto ela é o meio de comunicação que possibilita a conexão sem fio e envio de mensagens entre o *App Wanzeller's House* e a central de controle, por não utilizar cabos permitirá o usuário controlar as

funcionalidades do sistema de qualquer ponto dentro dos limites de alcance da transmissão de radiofrequência.

3.4. Componentes Físicos

3.4.1. Arduino Uno

Para o protótipo que foi proposto por esta monografia foi utilizado o Arduino UNO. Segundo informações retiradas do site oficial do arduino [17], o Arduino Uno é uma placa de microcontrolador baseado no ATmega328. Dispõe de 14 pinos digitais de E/S (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um ressonador cerâmico 16 MHz, uma conexão USB, um cabeçalho ICSP, e um botão de reset. Ele contém tudo o necessário para apoiar o microcontrolador, basta conectá-lo a um computador com um cabo USB ou ligá-lo com um adaptador AC-to-DC ou bateria para começar.

A razão por ter sido escolhida a placa Arduino UNO, foi devido ao seu tamanho reduzido e pelo seu preço de mercado está bastante acessível, visto que um dos objetivos principais do projeto é a automação de uma residência com baixo custo de investimento. Outra vantagem do Arduino UNO é a vasta quantidade de informações disponíveis nos fóruns desenvolvedores e sites. A plataforma de programação da placa chamada, *Arduino Environment*, oferece uma IDE para o desenvolvimento de códigos. Para que ocorra a comunicação com o computador através da porta USB, na qual, é utilizada tanto para provimento de energia quanto para transmissão de informação. A figura a seguir mostra a placa Arduino Uno.

Figura 3.11 - Arduino Uno R3

Fonte: Arduino, 2015

Com o objetivo de apresentar mais claramente as características do Arduino UNO que foi utilizado, mostram-se os dados deste na tabela 3. Estas informações estão de acordo com o *site* oficial do arduino (Arduino, 2015):

Tabela 4 - Características Gerais do Arduino UNO

Microcontrolador	ATmega328
Tensão Operacional	5 V
Tensão de Entrada (recomendado)	7 – 12 V
Tensão de Entrada (Limites)	6 – 20 V
Pinos de Entrada/Saídas digitais	14 (dos quais 6 podem ser usados como PWM)
Pinos de Entrada Analógica	6 pinos
Corrente DC por pino de E/S	40 mA
Memória Flash	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Comprimento	68,6 milímetros
Largura	53,4 milímetros
Peso	25 g

A placa Arduino UNO já está em sua terceira revisão (Arduino Uno R3) e você pode baixar seu esquema elétrico em formato PDF ou até mesmo todos os arquivos do projeto para edição. Ela tem duas camadas apenas e várias características interessantes de projeto. A seguir serão apresentadas as principais características do seu hardware [20].

3.4.1.1. Alimentação

A placa pode ser alimentada pela conexão USB ou por uma fonte de alimentação externa, conforme exibido na figura abaixo:

Figura 3.12 - Alimentação do Arduino

Fonte: EMBARCADOS, 2013

A alimentação externa é feita através do conector *Jack* com positivo no centro, onde o valor de tensão da fonte externa deve estar entre os limites 6V a 20V, porém se alimentada com uma tensão abaixo de 7V., a tensão de funcionamento da placa, que no Arduino UNO é 5V, pode ficar instável e quando alimentada com tensão acima de 12V, o regulador de tensão da placa pode sobreaquecer e danificar a placa. Dessa forma, é recomendado para tensões de fonte externa valores de 7V a 12V.

O circuito regulador para entrada externa é exibido a seguir. Nota-se que o CI (Circuito Integrado) responsável pela regulação de tensão é o NCP1117 da *OnSemi*. Destaque para o diodo D1 que protege o circuito caso uma fonte com tensão invertida for ligada.

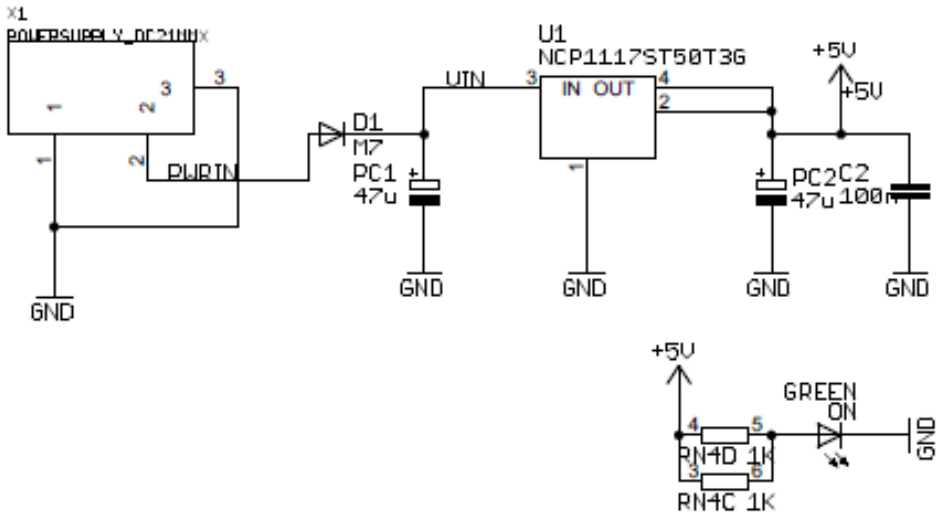


Figura 3.13 - Circuito Regulador de Tensão do Arduino

Fonte: EMBARCADOS, 2013

Quando o cabo USB é plugado a um PC, por exemplo, a tensão não precisa ser estabilizada pelo regulador de tensão. Dessa forma a placa é alimentada diretamente pela USB. O circuito da USB apresenta alguns componentes que protegem a porta USB do computador

em caso de alguma anormalidade. Na figura abaixo é exibido o circuito de proteção da USB da placa Arduino UNO.

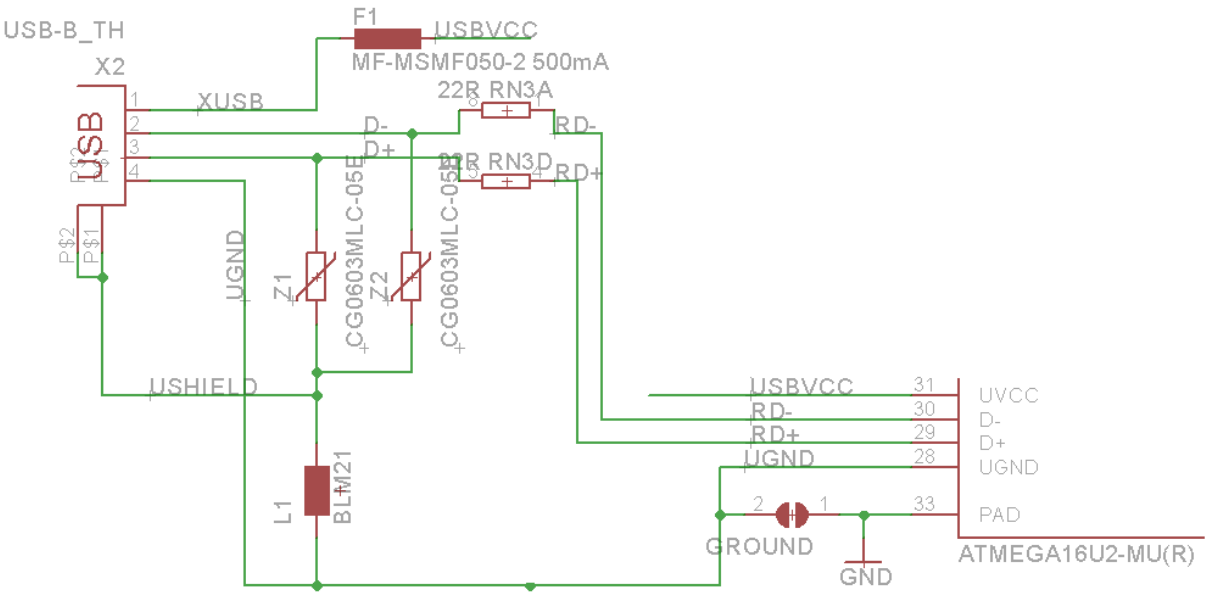


Figura 3.14 - Circuito de Proteção da USB da placa Arduino Uno

Fonte: EMBARCADOS, 2013

Os dois varistores (Z1 e Z2) podem suportar picos elevados de *SURGE* e energias elevadas de transientes. Seria preferível se, ao invés de varistores, fossem conectados diodos supressores de ESD (*Surge Protection Diodes)* que tem capacitância bem baixa, já que estão ligados a pinos rápidos de comunicação, mas o circuito funciona bem mesmo assim. Os resistores de 22 Ohms (RN3A e RN3D) limitam uma corrente resultante de alguma descarga elétrica eventual de um usuário em contato com o conector USB, resultante de transientes rápidos, protegendo, dessa forma, os pinos do microcontrolador.

Podem ser utilizados também para que o fusível resetável (F1) de 500mA impeça que a porta USB do computador queime, caso ocorra algum problema de projeto ou uma falha no circuito e ultrapasse a corrente de 500 mA quando a placa estiver conectada ao PC. O ferrite L1 foi incluído no circuito para que ruídos da USB externa não entrem no circuito da placa Arduino, através de seu terra.

Além dos recursos apresentados anteriormente, a placa conta com um circuito para comutar a alimentação automaticamente entre a tensão da USB e a tensão da fonte externa. Esse circuito está apresentado na figura abaixo. Caso haja uma tensão no conector DC e a USB é conectada, a tensão de 5V será proveniente da fonte externa e a USB servirá apenas para comunicação com o PC.

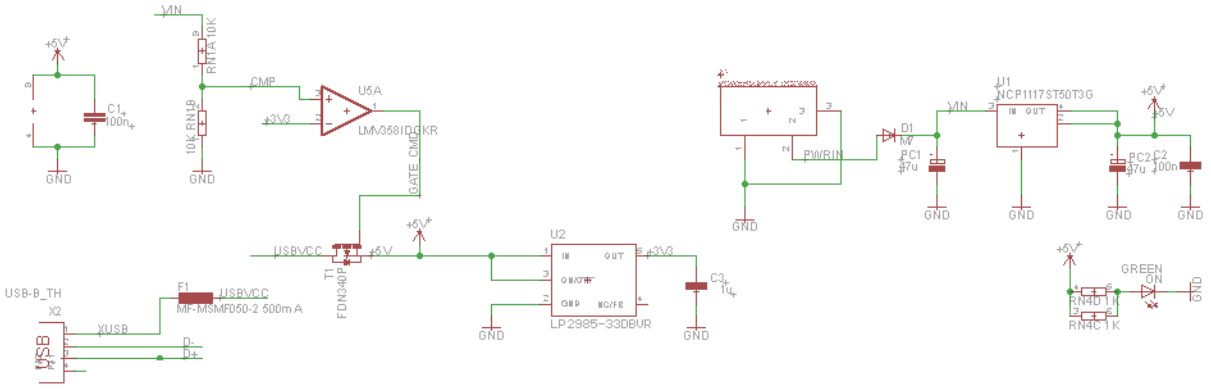


Figura 3.15 - Circuito para comutar a alimentação automaticamente

Fonte: EMBARCADOS, 2013

Como se pode observar na figura anterior que existe na placa um regulador de 3,3V, este componente é responsável por fornecer uma tensão continua de 3,3V para alimentação de circuitos ou *shields* que necessitem desse valor de tensão. Deve-se ficar atento ao limite máximo do valor da corrente que este regulador pode fornecer, que no caso é de 50 mA. A seguir são exibidos os conectores de alimentação para conexão de *shields* e módulos na placa Arduino UNO.

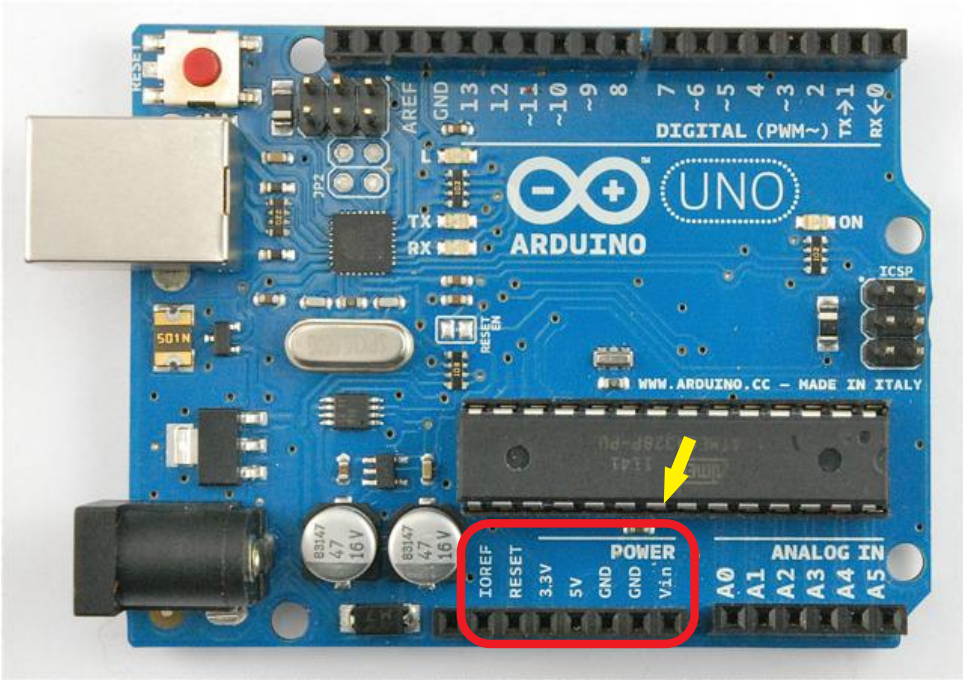


Figura 3.16 - Conectores de Alimentação do Arduino Uno

Fonte: Adaptado pelo autor

A tabela a seguir apresenta mais detalhes sobre os pinos referenciados na imagem anterior:

Tabela 5 - Características dos conectores de alimentação do Arduino Uno.

IOREF	Fornece uma tensão de referência para que *shields* possam selecionar o tipo de interface apropriada, dessa forma *shields* que funcionam com a placas Arduino que são alimentadas com 3,3V podem se adaptar para ser utilizados em 5V e vice-versa.
Reset	Pino conectado a pino de *RESET* do microcontrolador. Pode ser utilizado para um *reset* externo da placa Arduino.
3,3 V	Fornece tensão de 3,3V para alimentação de *shield*s e circuitos externos. Corrente máxima de 50 mA.
5 V	Fornece tensão de 5 V para alimentação de *shields* e circuitos externos.
VIN	Pino para alimentar a placa através de *Shields* ou bateria externa. Quando a placa é alimentada através do conector *Jack*, a tensão da fonte estará nesse pino.

3.4.1.2. Comunicação USB

Como uma interface USB para comunicação com o computador, há na placa um microcontrolador Atmel ATMEGA16U2. Este responsável pela forma transparente como funciona a placa Arduino UNO, possibilitando o *upload* do código binário gerado após a compilação do programa feito pelo usuário. Possui um conector ICSP para gravação de *firmware* através de um programador ATMEL, para atualizações futuras [20]. A figura a seguir apresenta a localização do microcontrolador acima citado.

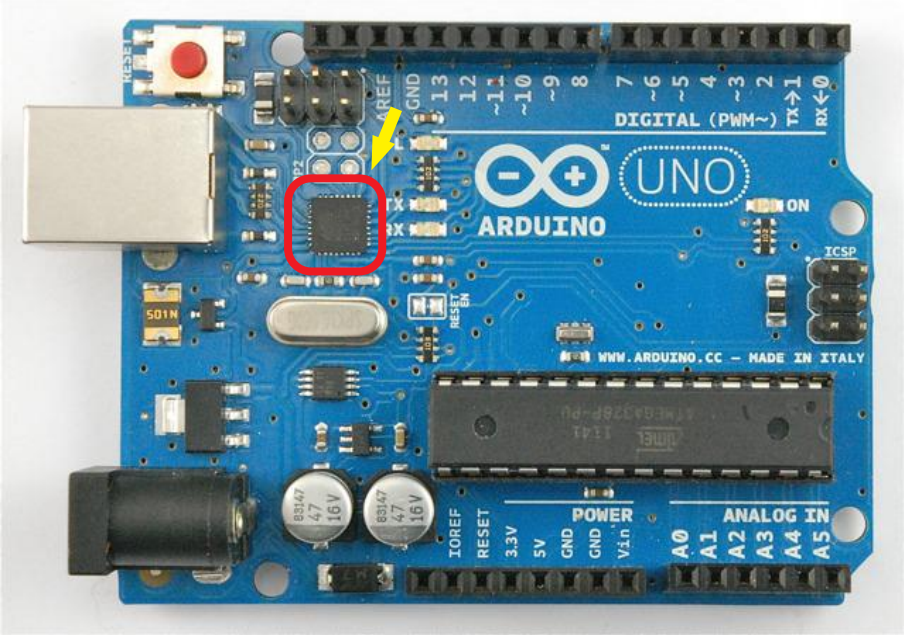


Figura 3.17 - Microcontrolador Atmel ATMEGA16U2 na placa Arduino Uno

Fonte: Adaptada pelo autor

Nesse microcontrolador também estão conectados dois leds (TX, RX), controlados pelo software do microcontrolador, que indicam o envio e recepção de dados da placa para o computador. Esse ATMEGA16U2 possui um cristal externo de 16 MHz. É interessante notar a conexão entre este microcontrolador com o Atmel ATMEGA328, onde é feita pelo canal serial desses microcontroladores. Outro ponto interessante que facilita o uso da placa Arduino é a conexão do pino 13 do ATMEGA16U2 ao circuito de RESET do ATMEGA328, possibilitando a entrada no modo *bootloader* automaticamente quando é pressionado o botão *Upload* na IDE. Essa característica não acontecia nas primeiras placas Arduino, onde era necessário pressionar o botão de RESET antes de fazer o *Upload* na IDE [20].

3.4.1.3. Cérebro do Arduino Uno

O componente principal da placa Arduino UNO é o microcontrolador ATMEL ATMEGA328, um dispositivo de 8 bits da família AVR com arquitetura RISC avançada e com encapsulamento DIP28. Ele conta com 32 KB de *Flash* (mas 512 Bytes são utilizados pro *bootloader*), 2 KB de RAM e 1 KB de EEPROM. Pode operar a até 20 MHz, porém na placa Arduino UNO opera em 16 MHz, valor do cristal externo que está conectado aos pinos 9 e 10 do microcontrolador. Observe que, para o projeto dessa placa, os projetistas escolheram um cristal com dimensões bem reduzidas. Possui 28 pinos, sendo que 23 desses podem ser utilizados como I/O [20]. A imagem abaixo exibe a sua pinagem:

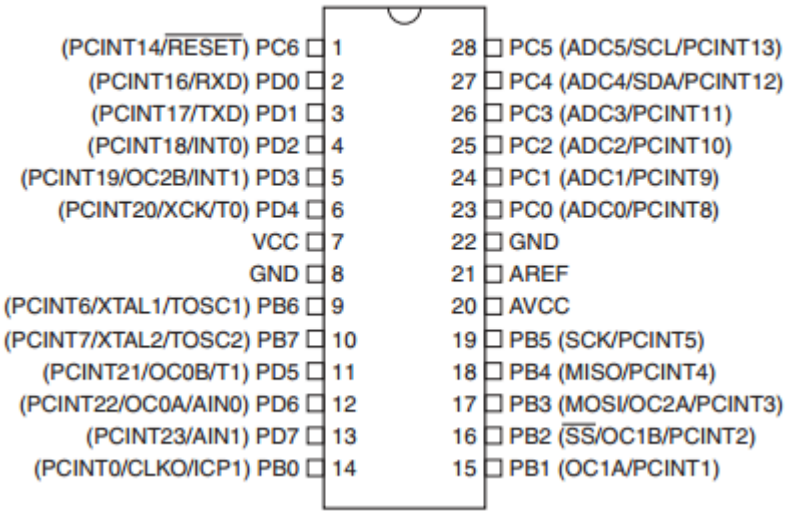


Figura 3.18 - Microcontrolador ATMEL ATMEGA328

Fonte: EMBARCADOS, 2013

Esse microcontrolador pode operar com tensões bem baixas, de até 1,8V, mas nessa tensão apenas opera até 4MHz. Possui dois modos de consumo, super e baixos, o *Power-Down Mode* e o *Power-Save Mode*, para que o sistema possa poupar energia em situações de espera. Possui, como periféricos uma USART (*Universal Synchrounous/Asynchrounous Receiver/Transmitter*) que funciona a até 250kbps, uma SPI, que vai a até 5MHz, e uma I2C que pode operar até 400kHz. Conta com um comparador analógico interno ao CI e diversos *timers*, além de 6 PWMs. A corrente máxima por pino é de 40mA, mas a soma da corrente de todo o CI não pode ultrapassar 200mA. Ele possui um oscilador interno de 32kHz que pode ser utilizado, por exemplo, em situações de baixo consumo.

3.4.1.4. Entradas e Saídas

A placa Arduino UNO possui pinos de entradas e saídas digitais, assim como pinos de entradas e saídas analógicas, abaixo é exibido a pinagem conhecida como o padrão do Arduino Uno:

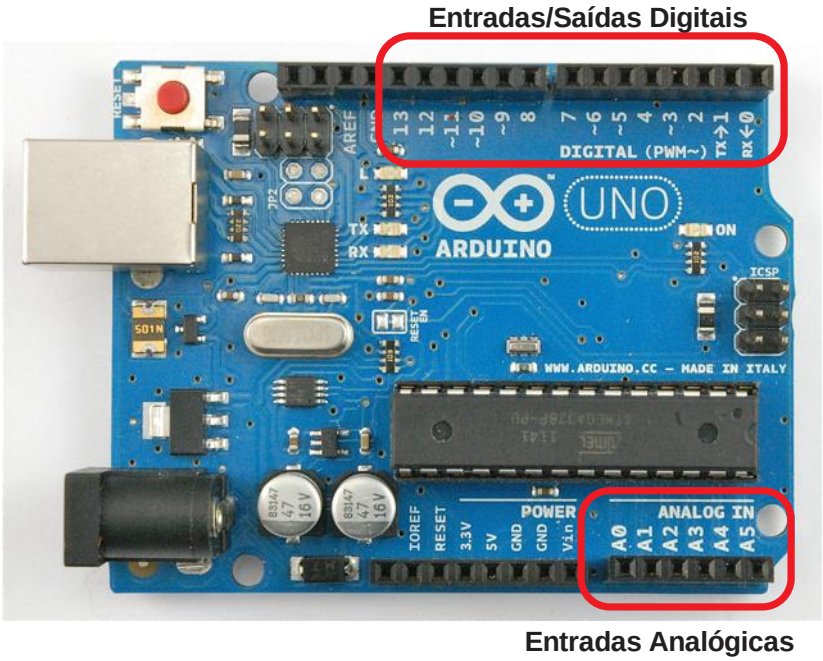


Figura 3.19 - Pinos de E/S do Arduino Uno

Fonte: Adaptada pelo autor

Conforme exibido na figura 3.19, a placa Arduino UNO possui 14 pinos que podem ser usados como entrada ou saída digitais. Estes Pinos operam em 5 V, onde cada pino pode fornecer ou receber uma corrente máxima de 40 mA. Cada pino possui resistor de *pull-up* interno que pode ser habilitado por *software*. Alguns desses pinos possuem funções especiais:

PWM: 3,5,6,9,10 e 11 podem ser usados como saídas PWM de 8 bits através da função *analogWrite()*;

Comunicação serial: 0 e 1 podem ser utilizados para comunicação serial. Deve-se observar que estes pinos são ligados ao microcontrolador responsável pela comunicação USB com o PC;

Interrupção externa: 2 e 3. Estes pinos podem ser configurados para gerar uma interrupção externa, através da função *attachInterrupt()*.

Para interface com o mundo analógico, a placa Arduino UNO possui 6 entradas, onde cada uma tem a resolução de 10 bits. Por padrão a referência do conversor AD está ligada internamente a 5V, ou seja, quando a entrada estiver com 5V o valor da conversão analógica digital será 1023. O valor da referência pode ser mudado através do pino AREF. A figura a seguir exibe a relação entre os pinos do microcontrolador ATMEL ATMEGA328 e a pinagem do Arduino UNO:

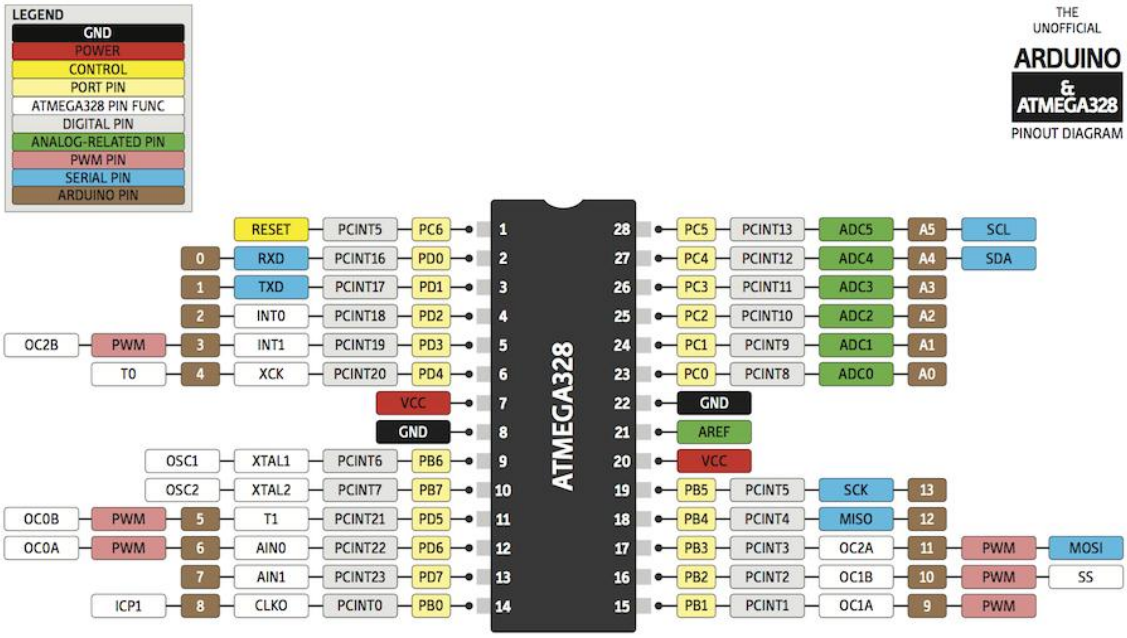


Figura 3.20 - Relação entre o microcontrolador ATMEL ATMEGA328 e o Arduino UNO

Fonte: EMBARCADOS, 2013

Quem manipula a placa e projeta o circuito que será conectado aos seus pinos I/Os deve ter muito cuidado, pois, entre os pinos do microcontrolador e a barra de pinos, não há nenhum resistor, que limite a corrente, além disso, dependendo do local onde está trabalhando pode-se provocar curto circuito nos pinos já que a placa não possui isolação na sua parte inferior, como mostrada na figura a seguir:

Figura 3.21 - Parte Inferior da Placa Arduino Uno

Fonte: ARDUINO, 2015

A placa não conta com botão liga/desliga. Logo, se quiseres desligar a alimentação, tem que “puxar” o cabo que alimenta a placa. O cabo USB tipo B não é tão comum quanto o mini USB, utilizado bastante em celulares. Isso pode ser um problema, caso perca o cabo que veio com a placa [20].

3.4.1.5. Programação da Placa

A placa Arduino UNO é programada através da comunicação serial, pois o microcontrolador vem programado com o *bootloader*. Dessa forma não há a necessidade de um programador para fazer a gravação (ou *upload*) do binário na placa. A comunicação é feita através do protocolo STK500. A programação do microcontrolador também pode ser feita através do conector ICSP (*in - circuit serial programming*) utilizando um programador ATMEL.

3.4.1.6. Características da Placa

A placa Arduino UNO possui pequenas dimensões cabendo na palma da mão, uma das grandes vantagens da utilização desta plataforma. Possui 4 furos para que a mesma possa ser fixada em alguma superfície. A figura a seguir exibe as suas dimensões físicas:

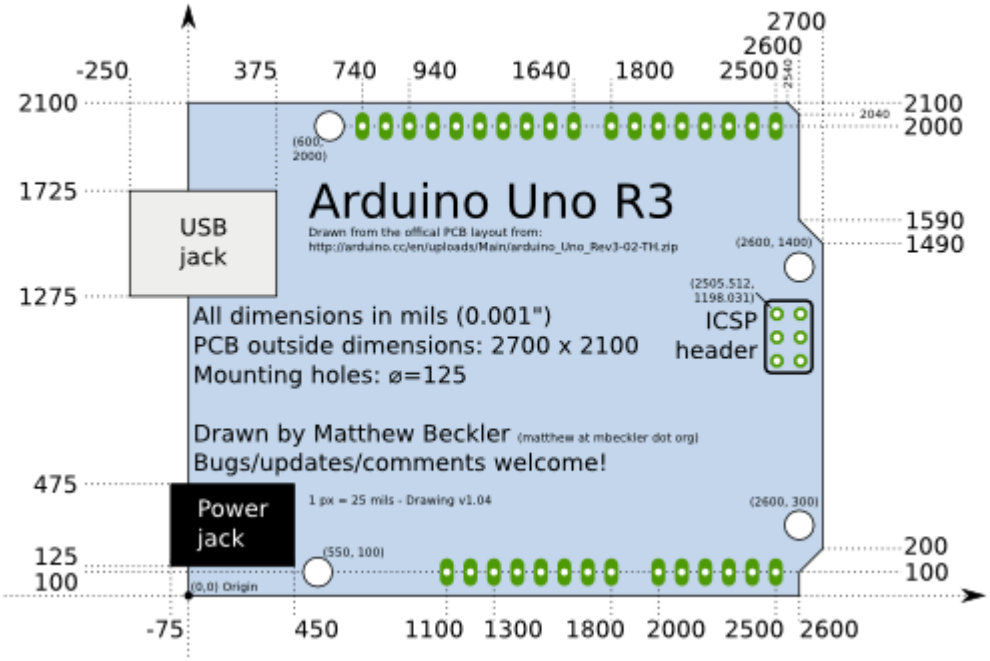


Figura 3.22 - Dimensões Físicas da Placa Arduino Uno

Fonte: EMBARCADOS, 2013

Embora seja limitado para aplicações consideradas de grande porte, devido a pouca capacidade de processamento e baixo número de portas digitais e analógicas disponíveis, o

Arduino Uno ainda é uma ferramenta poderosa para vários tipos de aplicações, além de se tratar de uma opção viável no mercado em termos financeiros, motivo este que foi de fundamental importância na escolha deste modelo.

A figura a seguir apresenta uma placa Arduino Uno com suas características bem detalhadas sendo mostradas na própria imagem para melhor compreensão:

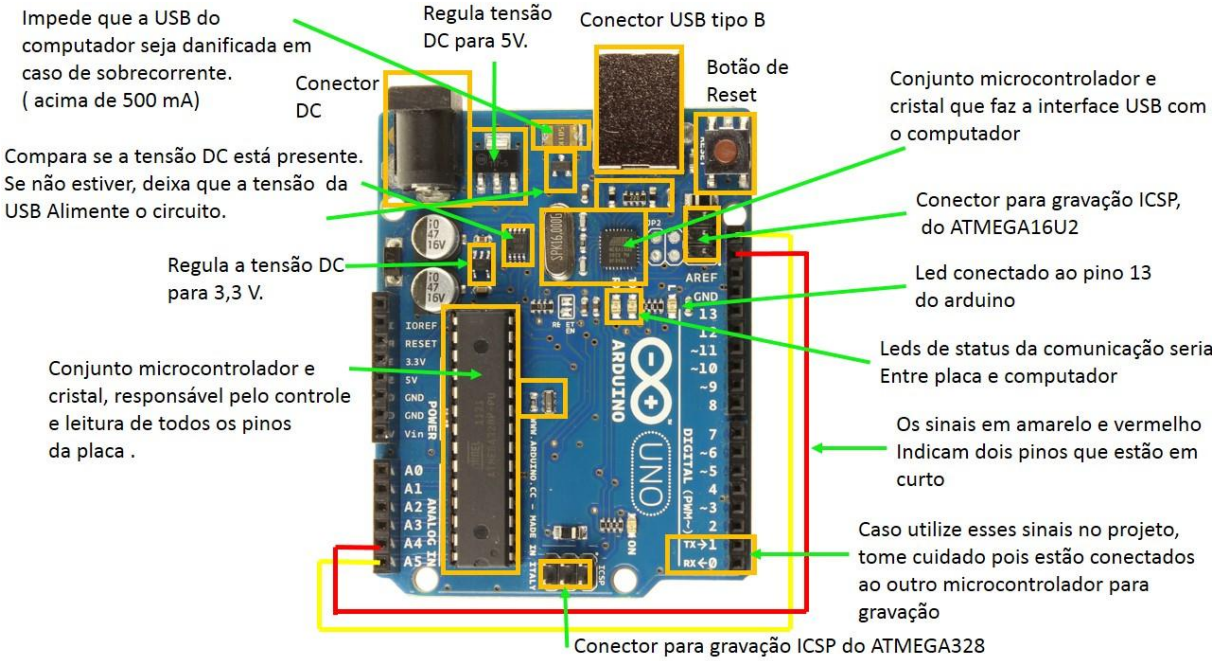


Figura 3.23 - Resumo das Característica da Placa Arduino

Fonte: EMBARCADOS, 2013

3.4.2. Ethernet Shield

Outra vantagem da Plataforma Arduino é a existência de vários *Shields* que permitem ao usuário estender a capacidade do sistema ou especificar uma aplicação desejada. Os *Shields* são placas de circuito impresso que são encaixados à placa principal e cumprem função específica no sistema [5]. Dentro desse contexto e atendendo as necessidades deste projeto existe o Arduino *Ethernet Shield* que foi utilizado para que fosse possível o envio de comandos através de uma página *web*.

Segundo informações retiradas do site oficial (ARDUINO, 2015) [21], o Arduino *Ethernet Shield* permite que uma placa Arduino conecte-se à internet através de um cabo de rede. Ele baseia-se no chip WIZnet *ethernet* W5100. O W5100 WIZnet fornece uma rede IP (*Internet Protocol*) e utiliza os protocolos TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*). Ele suporta até quatro conexões de soquete simultâneas. A Ethernet *Shield* conecta-se a uma placa Arduino usando longos pinos (cabeçalhos que se estendem através do

Shield). Isso mantém a pinagem intacta e permite que outro *shield* possa ser colocado por cima no momento da montagem [21].

A Ethernet *Shield* possui uma conexão RJ45 padrão, com um transformador de linha integrado e *Power over Ethernet* (PoE) habilitado. O PoE é a tecnologia que descreve uma forma segura de passar energia elétrica juntamente com os dados por cabos *Ethernet*. A figura a seguir apresenta uma placa *Ethernet Shield.*

Figura 3.24 - Arduino *Ethernet Shield*
Fonte: ARDUINO, 2015

Observa-se através da imagem também que existe um espaço para cartão microSD, que pode ser usado para armazenar arquivos para servir através da rede. O leitor do cartão microSD fica acessível através da Biblioteca SD do Arduino.

Foi utilizada neste projeto a *Ethernet Shield*, pois ela possui um protocolo de mais fácil utilização (TCP) e confiável, ao contrário da *Shield Xbee* que utiliza o protocolo Zigbee e disponibiliza acesso através de rede sem fio, podendo sofrer maior instabilidade, causada por agentes externos. O TCP é o protocolo utilizado para estabelecer a conexão entre o Arduino e o host externo, sendo este último o responsável pelo envio dos comandos através da interface *web*.

Ainda nesse contexto, segundo o *site* oficial (ARDUINO, 2015), a forma de comunicação com a placa principal é feita utilizando o barramento SPI, através dos pinos 10,

11, 12 e 13. Onde, no pino 10 é feita a seleção do W5100. Este fornece o protocolo TCP/IP para o Arduino na rede, possibilitando toda comunicação com outro dispositivo via internet.

O protocolo TCP/IP é o principal protocolo de envio e recebimento de dados via internet. Por se tratar de na verdade de um conjunto de protocolos integrados, é também conhecido como “Pilha de Protocolos”. E está dividido em quatro camadas distintas, de forma a garantir a integridade dos dados que trafegam pela rede, são elas: aplicação, transporte, rede e interface. Cada uma delas é responsável pela execução de tarefas distintas. Essa divisão em camadas é uma forma de garantir a integridade dos dados que trafegam pela rede (TECHMUNDO, 2012). As camadas são descritas a seguir.

Camada de Aplicação: essa camada é utilizada pelos programas para enviar e receber informações de outros programas através da rede. Nela, você encontra protocolos como SMTP (para email), FTP (transferência de arquivos) e o famoso HTTP (para navegar na internet). Uma vez que os dados tenham sido processados pela camada de aplicação, eles são enviados para a divisão abaixo.

Camada de Transporte: a camada de transporte é responsável por receber os dados enviados pelo grupo acima, verificar a integridade deles e dividi-los em pacotes. Feito isso, as informações são encaminhadas para a camada logo abaixo dela.

Camada de Rede: os dados empacotados são recebidos e anexados ao endereço virtual (IP) do computador remetente e do destinatário. Agora é a vez dos pacotes serem, enfim, enviados pela rede. Para isso, são passados para a camada Interface.

Camada de Interface: a tarefa da Interface é receber e enviar pacotes pela rede. Os protocolos utilizados nessa camada dependem do tipo de rede que está sendo utilizado. Atualmente, o mais comum é o *Ethernet*, disponível em diferentes velocidades.

Em geral, são estes os processos para que ocorrem no W5100 para se receber ou enviar um dado pela internet.

Por fim, para acoplar a placa *Ethernet Shield*, basta esta sobre o Arduino UNO, nos terminais correspondentes, e assim ligar o cabo de rede proveniente do roteador na entrada *RJ45* do *Shield.* A figura a seguir apresenta este esquema montado:

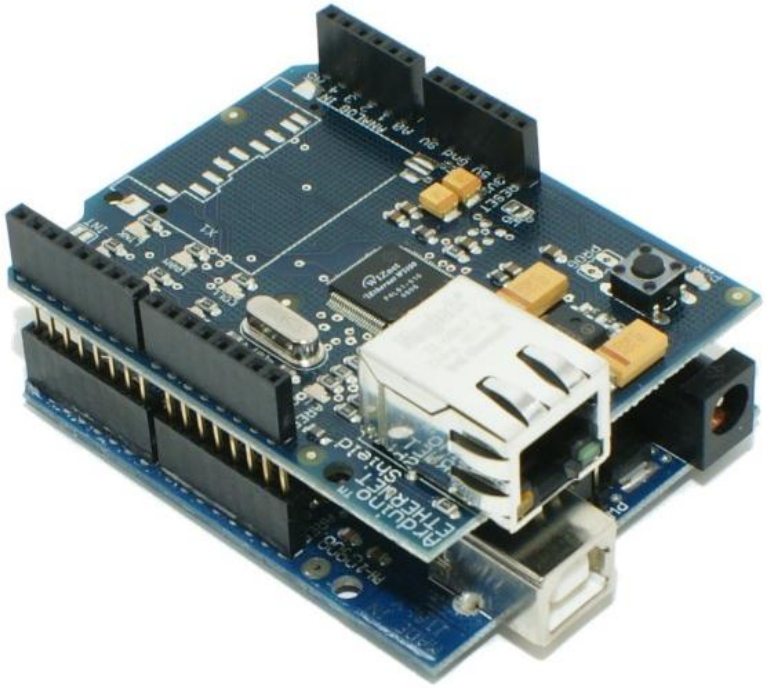


Figura 3.25 - *Ethernet Shield* acoplado ao Arduino UNO
Fonte: LIQUIDWARE, 2015

Pode-se observar na figura acima o *Ethernet Shield* acoplado sobre a placa Arduino UNO. Vale ressaltar que o encaixe deste sistema acaba ocupando 4 pinos digitais do Arduino, diminuindo assim o números de portas digitais disponíveis para utilização no projeto proposto.

3.4.3. Módulo Relé

Os relés são componentes eletromecânicos capazes de controlar circuitos externos de grandes correntes a partir de pequenas correntes ou tensões, ou seja, acionando um relé com uma pequena voltagem 5 volts e 50 miliamperes podemos controlar outro dispositivo que esteja ligado em 127 ou 220 volts em 2 ampères, como por exemplo o realizar o acionamento de lâmpadas através do arduino.

Os relés funcionam da seguinte forma: quando uma corrente circula pela bobina, ela cria um campo magnético que atrai um ou uma série de contatos fechando ou abrindo circuitos. Ao interromper essa corrente o campo magnético também será interrompido, fazendo com que os contatos voltem para a posição original. Os relés podem ter algumas configurações referentes aos seus contatos: eles podem ser NA (normalmente aberto), NF (normalmente fechado) ou ambos. Observe a figura a seguir:

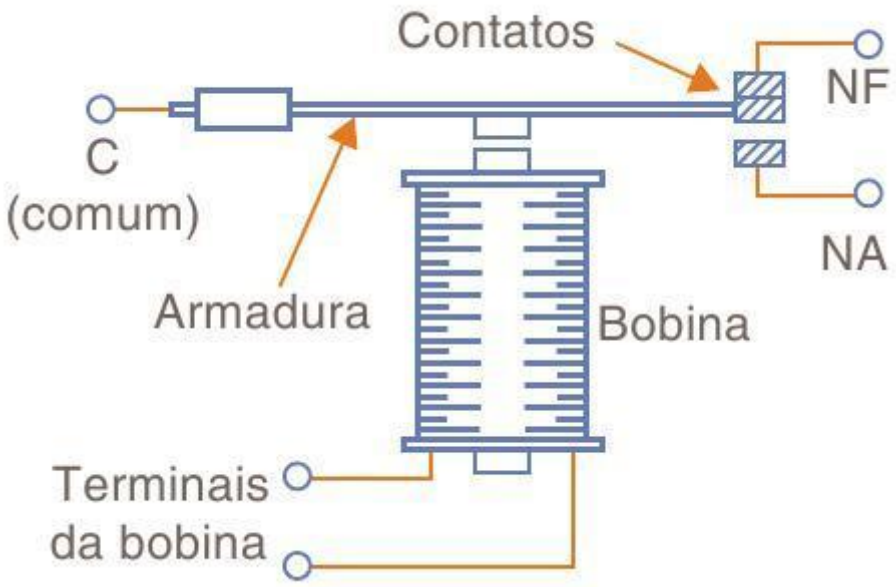


Figura 3.26 – Relé
Fonte: SABER ELETRÔNICA, 2015.

Os contatos NA são os que estão abertos enquanto a bobina não está energizada e que fecham, quando a bobina recebe corrente. Os NF abrem-se quando a bobina recebe corrente, ao contrário dos NA. O contato central ou C é o comum, ou seja, quando o contato NA fecha é com o C que se estabelece a condução e o contrário com o NF.

O objetivo do relé é utilizar pequena quantidade de energia eletromagnética (proveniente, por exemplo, de um pequeno interruptor ou circuito eletrônico simples) para mover uma armadura que possa gerar uma quantidade de energia muito maior.

A principal vantagem dos Relés em relação aos SCR e os Triacs é que o circuito de carga está completamente isolado do circuito de controle, podendo inclusive trabalhar com tensões diferentes entre controle e carga. A desvantagem é o fator do desgaste, pois em todo o componente mecânico há uma vida útil, o que não ocorre nos tiristores.

Devem ser observadas as limitações dos relés quanto a corrente e tensão máxima admitida entre os terminais. Se não forem observados estes fatores a vida útil do relé estará comprometida, ou até mesmo a do circuito controlado. A figura a seguir mostra o módulo relé de 2 canais de 5V que foi utilizado neste projeto para fazer o acionamento de dispositivos através de comandos enviados pelo arduino.

Figura 3.27 - Módulo Relé 2 Canais 5V
Fonte: MERCADO LIVRE, 2015

3.4.4. Sensor de Temperatura (LM35)

O sensor LM35 é um sensor de precisão, fabricado pela *National Semiconductor*, que apresenta uma saída de tensão linear relativa à temperatura em que ele se encontrar no momento em que for alimentado por uma tensão de 4-20Vdc e GND, tendo em sua saída um sinal de 10mV para cada Grau Celsius de temperatura, sendo assim, apresenta uma boa vantagem com relação aos demais sensores de temperatura calibrados em "KELVIN", não necessitando nenhuma subtração de variáveis para que se obtenha uma escala de temperatura em Graus Celsius.

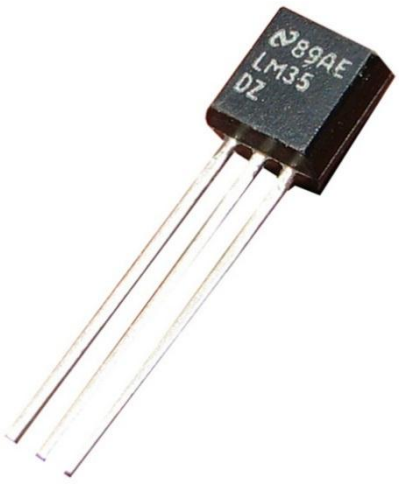


Figura 3.28 - Sensor de Temperatura LM35
Fonte: TRASTEJANT, 2015

O LM35 não necessita de qualquer calibração externa ou *"trimming"* para fornecer com exatidão, valores temperatura com variações de ¼ºC ou até mesmo ¾ ºC dentro da faixa de

temperatura de –55ºC à 150ºC. Este sensor tem saída com baixa impedância, tensão linear e calibração inerente precisa, fazendo com que o interfaceamento de leitura seja especificamente simples, barateando todo o sistema em função disto.

Este sensor poderá ser alimentado com alimentação simples ou simétrica, dependendo do que se desejar como sinal de saída, mas independentemente disso, a saída continuará sendo de 10mV/ºC. Ele drena apenas 60μA para estas alimentações, sendo assim seu auto-aquecimento é de aproximadamente 0.1ºC ao ar livre.

O sensor LM35 é apresentado com vários tipos de encapsulamentos, sendo o mais comum o TO-92, que mais se parece com um transistor, e oferece ótima relação custo benefício, por ser o mais barato dos modelos e propiciar a mesma precisão dos demais. A grande diversidade de encapsulamentos se dá devido à alta gama de aplicações deste integrado.

Existem no mercado hoje em dia, diversos tipos de sensores de temperatura, que vão desde os NTC´s, PTC´s e diodos até os mais variados tipos de termopares, dentre outros. Porém, estima-se que talvez nenhum dos citados anteriormente seja de tão simples manuseio e exija tão poucos aparatos eletrônicos para que funcione, quanto o modelo LM35, pois o circuito usual é bastante simples, necessitando apenas do sensor propriamente dito, um sistema amplificador de sinal e de uma interface que realize a leitura do sinal amplificado, quem sabe até mostrando um valor de temperatura diretamente em um visor ou display ou até mesmo disparando algum elemento eletrônico como, por exemplo, um transistor quando a situação for apropriada. Para este projeto o LM35 será utilizado para medir a temperatura de determinado ambiente, sendo esta apresentada através da interface web desenvolvida.

3.4.5. Sensor de Luminosidade (LDR)

LDR (do inglês *Light Dependent Resistor*), em português **Resistor Dependente de Luz** ou Fotoresistência é um componente eletrônico passivo do tipo resistor variável, mais especificamente, é um resistor cuja resistência varia conforme a intensidade da luz (iluminamento) que incide sobre ele. Tipicamente, à medida que a intensidade da luz aumenta, a sua resistência diminui. A figura a seguir apresenta um LDR:

Figura 3.29 - Sensor de Luminosidade LDR
Fonte: COMOFAZERASCOISAS, 2015

O LDR é construído a partir de material semicondutor com elevada resistência elétrica. Quando a luz que incide sobre o semicondutor tem uma frequência suficiente, os fótons que incidem sobre o semicondutor libertam elétrons para a banda condutora que irão melhorar a sua condutividade e assim diminuir a resistência. Dependendo do tipo, um LDR pode ser sensível às faixas de luz: Infravermelhos(IR), Luz visível ou Ultravioletas (UV) [23].

Neste projeto em questão este sensor foi utilizado para monitor o status da lâmpada, ou seja, verificar se a lâmpada está ligada ou desligada. Assim que a lâmpada é ligada a intensidade luminosa sobre o LDR é muito maior do que quando a lâmpada está desligada, com essas informações podemos saber se a lâmpada está ligada ou desligada. Esta solução não é a ideal para monitorar status das lâmpadas, pois podem existir fontes luminosas externas podendo causar inconsistências das informações, contudo este sensor foi utilizado devido ao seu baixo preço e este ser mais acessível nas eletrônicas do que outros componentes eletrônicos capazes de identificar quando há corrente no circuito.

3.4.6. Sensor de Movimento (PIR)

P Infra**R**ed sensor ou normalmente chamado por PIR, é um sensor que usa Piroeletricidade (do grego pir, fogo e electricidade) que é a capacidade de certos materiais gerarem uma tensão temporária quando são aquecidas ou arrefecidas utilizado infravermelhos. Os PIR também são chamados de “detectores de movimento” e “detectores de infravermelhos passivos”. Eles são chamados de “passivo” porque não projetaram qualquer tipo de feixe.

Os PIR são constituídos por três partes básicas. No lado de fora existe uma lente de Fresnel, que é uma janela de plástico branco opaco, como mostra a figura a seguir. O chip detector fica atrás da lente.

Figura 3.30 - Sensor PIR

Fonte: ARDUINOECIA, 2014

A lente Fresnel é muito parecida com as lentes usadas em faróis. Esta técnica permite que luzes, relativamente pequenas, possam ser vistas a distâncias muito grandes. Neste caso a lente Fresnel serve para amplificar a assinatura de calor infravermelho que está sendo dissipada por uma pessoa ou animal a passar (animais de sangue quente). Para além de amplificar a energia, a lente também divide o sinal de infravermelhos em vários feixes de luz. A figura a seguir representa o funcionamento de uma lente Fresnel.

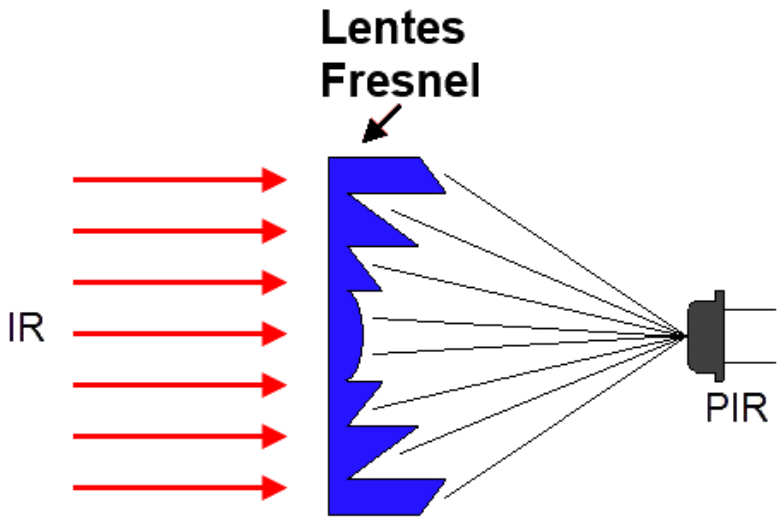


Figura 3.31 - Funcionamento da lente Fresnel

Fonte: ARDUINOECIA, 2014

O chip detector é um pequeno componente electrónico que é sensível à energia infravermelha. O chip vê as ondas infravermelhas e cria um pequeno pico de tensão, o qual é enviado para um circuito e desencadeia um contato, contato esse que pode ser usado para acionar um alarme, acender uma luz, abrir uma porta ou qualquer outra função que se pretenda.

Neste referido projeto o sensor PIR foi utilizado no sistema alarme residencial, onde o usuário pode ativar e desativar o alarme pelo iPhone utilizado pelo protótipo.

Este módulo com o sensor PIR é de fácil utilização quando se trabalha com arduino, além de ser um componente facilmente encontrado nas lojas especializadas e com um preço bastante acessível, contribuindo para a utilização deste no trabalho de automação residencial aqui apresentado. A figura a seguir mostra com mais detalhes este módulo sensor.

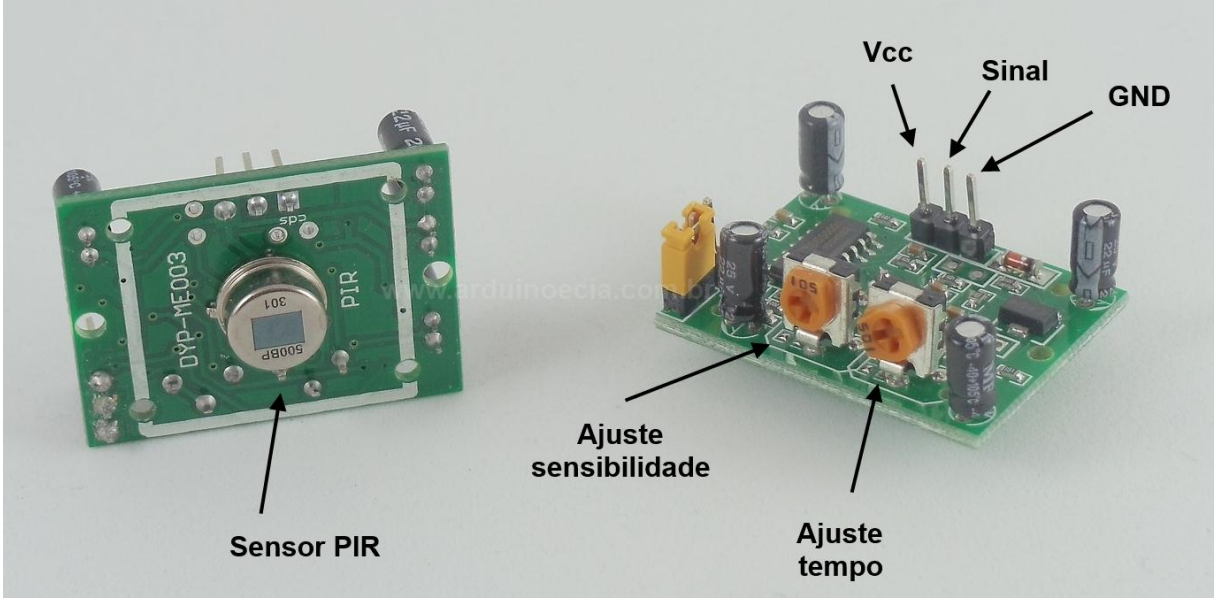


Figura 3.32 - Módulo Sensor PIR Detalhado

Fonte: ARDUINOECIA, 2014

A figura acima apresenta de forma bem ilustrativa todas as partes integrantes deste módulo, este ainda possui ajuste de sensibilidade, onde o usuário pode controlar sensibilidade de detecção do sensor, podendo ajudar a evitar falsos alarmes e também ajuste de tempo, onde pode-se configurar o tempo que o sensor permanece com estado ativado ao detectar algum movimento.

CAPÍTULO 4

METODOLOGIAS DO MODELO PROPOSTO

4.1. Apresentação Geral do Modelo Proposto

O desenvolver deste projeto requer conhecimento de uma série de áreas distintas, como conhecimento na área de redes de computadores, algoritmo e programação, instalações elétricas residenciais, conceitos de eletrônica e até mesmo conceitos arquitetônicos. Existindo assim uma grande interdisciplinaridade, no qual, esta é de grande valia em trabalhos de conclusão de curso, podendo o formando poder aproveitar este conhecimento e tirar melhor proveito disso.

Este projeto desenvolvido consiste em várias funcionalidades que se dividem em três categorias básicas, que serão descritas com mais detalhamentos no decorrer deste capítulo. Em contribuição com a figura 3.1 e para melhor entendimento do que foi desenvolvida neste projeto. A figura a seguir apresenta um esquema geral do funcionamento deste protótipo.

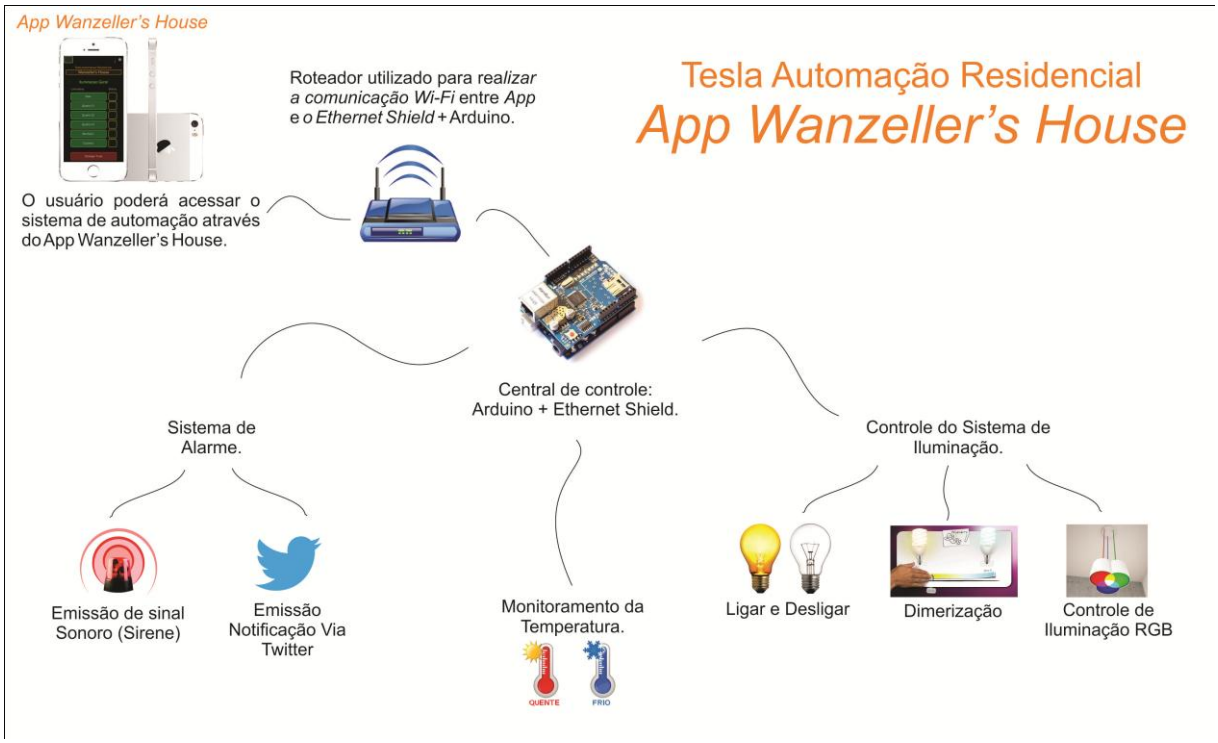


Figura 4.1 - Esquema geral de funcionamento do sistema desenvolvido.

Fonte: Adaptado pelo autor

4.2. Descrição do Funcionamento

Como mencionado anteriormente o sistema de controle possui basicamente três grandes funcionalidades, estas são:

4.1.1. Controle do Sistema de Iluminação da Residência

Através do sistema de controle o usuário poderá ligar ou desligar as lâmpadas via *iPhone*, além de poder contar com um sistema de iluminação dimerizável, onde através deste, o usuário pode controlar a intensidade luminosa de determinada lâmpada, agregando grande utilidade ao sistema, visto que não trabalhar com a iluminação em sua intensidade máxima pode economizar na conta de energia. O sistema de controle de iluminação apresentado aqui nesta monografia em forma de protótipo possui também um sistema de iluminação RGB, onde o usuário pode personalizar determinado ambiente através da variação dessas três cores (**R**ed, **G**reen e **B**lue), podendo acentuar os detalhes arquitetônicos de uma sala ou criar um clima especial, seja ele romântico ou festivo. A figura a seguir mostra um esquema geral de como toda iluminação será controlada através do sistema de controle, desde o envio do comando até o acionamento do relé.

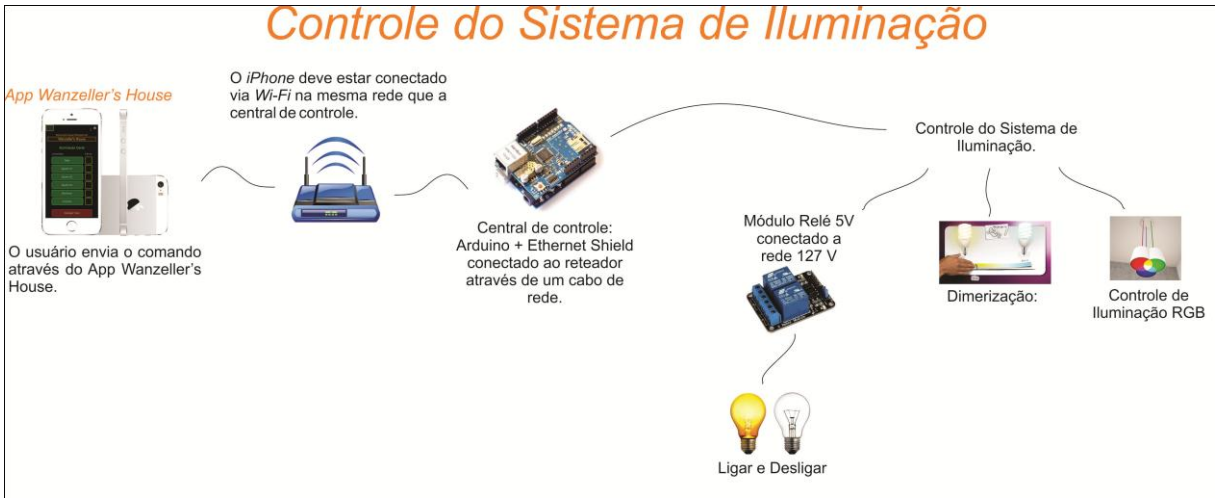


Figura 4.2 - Controle do Sistema de Iluminação

Fonte: Adaptado pelo autor

Vale ressaltar que as luzes automatizadas valorizam a casa e possuem uma série de vantagens agregadas, como:

Economia - Com as luzes automatizadas, as lâmpadas só acendem quando é necessário.

Durabilidade - Por serem acionadas menos vezes, as lâmpadas também ganham vida útil mais longa.

Segurança - Mesmo na ausência dos moradores, a casa é iluminada em momentos determinados, previamente programados.

Comodidade - O acionamento das luzes é mais prático, via dispositivos móveis.

Beleza - Cada cômodo pode ter sua própria luminosidade, regulada de acordo com as rotinas da família.

4.1.2. Monitoramento da Temperatura

Este sistema permitirá também fazer o monitoramento da temperatura de determinado ambiente da casa, no protótipo que foi desenvolvido será monitorada a temperatura da sala, visto que para a implementação nos demais ambientes da casa terá o mesmo princípio. Foi utilizado um sensor LM35 para monitorar esta temperatura, poderia ser adicionado a este módulo algum tipo de controle, como por exemplo ao se chegar a determinada temperatura poderia se acionar determinado dispositivo para resfriar o ambiente, contudo para este projeto fez-se a utilização apenas do monitoramento, mas sabe-se que as opções são inúmeras. Figura a seguir mostra o funcionamento de como será realizado a leitura da temperatura e o envio para o dispositivo.

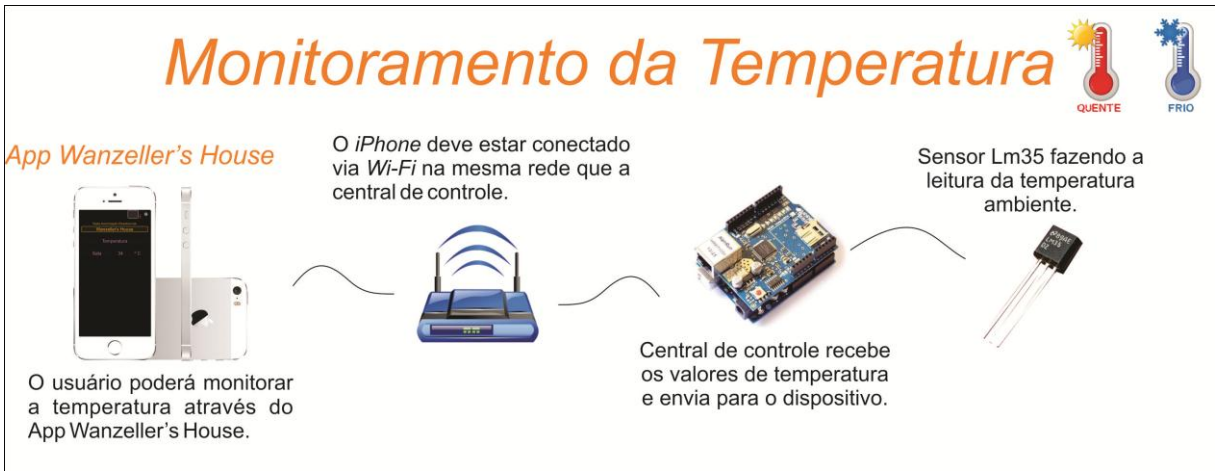


Figura 4.3 - Esquema geral do monitoramento da temperatura ambiente

Fonte: Adaptado pelo autor

4.1.3. Sistema de Alarme

Foi implementado também um sistema de alarme residencial, este baseia-se na utilização de um sensor de movimento, onde após o alarme ser ativado pelo aplicativo, se o sensor (PIR) detectar algum movimento em seu raio de captura o alarme será disparado, o disparo do alarme aciona uma buzina que emitirá um sinal sonoro informando que há um possível intruso na residência e logo após o sistema emite também uma notificação via *twitter* para o usuário informando que o alarme foi violado, bastando apenas o celular está conectado à internet para receber a notificação. Esta notificação via *twitter* foi de grande utilidade, pois além de ser extremamente funcional é de uso gratuito, pois utiliza uma biblioteca disponibilizada no site oficial do arduino [25], bastando para isso o usuário ter uma conta no *twitter* que também é gratuita, alcançando com isso um dos principais objetivos do que foi

proposto por este projeto que é a relação custo benefício. A figura a seguir mostra o esquema de funcionamento geral do sistema de alarme que será desenvolvido.

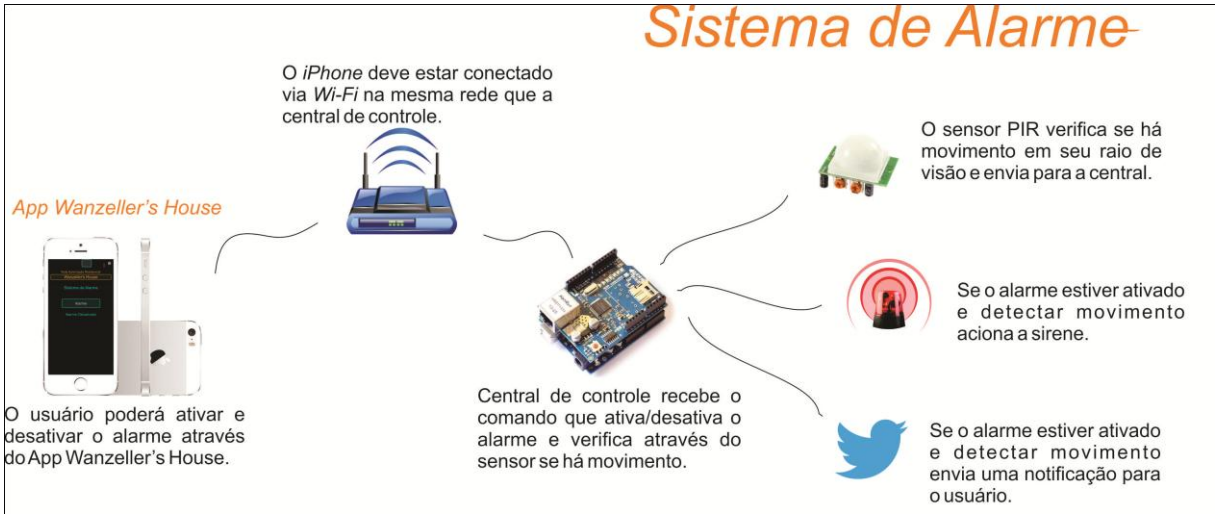


Figura 4.4 - Esquema geral de funcionamento do sistema de alarme

Fonte: Adaptado pelo autor

4.2. Desenvolvimento do Modelo Proposto

Para implementação e em busca de melhores resultados no desenvolvimento deste protótipo do sistema de automação residencial, foi implementado um projeto prático, através de uma maquete residencial para melhor visualização e aplicação dos conceitos já citados anteriormente. Este projeto, cujo sistema de controle de automação será controlado via dispositivos móveis, mais especificamente por um *iPhone 5S,* onde foi passado para este o aplicativo com todas as funções do sistema de automação residencial, nomeado como *App Wanzeller's House.*

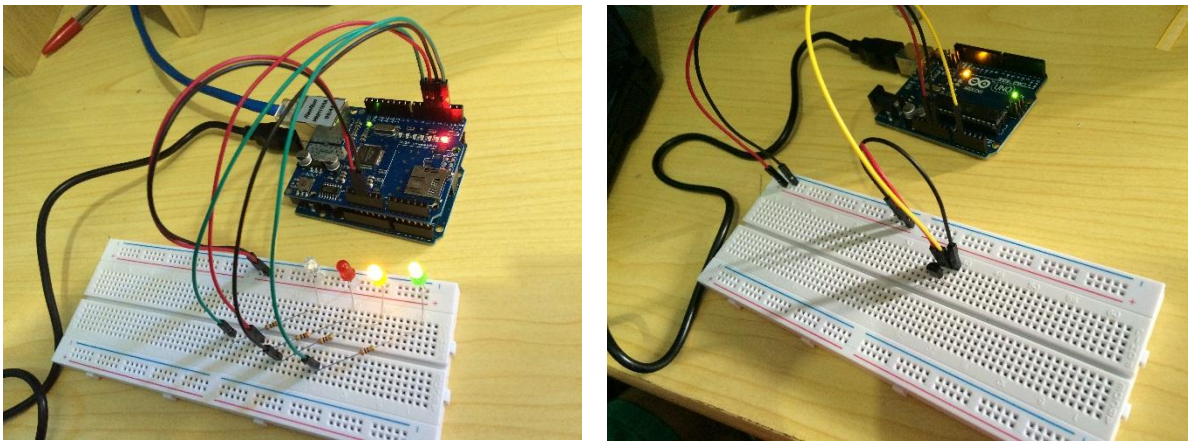
Tem-se por objetivo futuro torná-lo um sistema comercial, mas a princípio será utilizado somente para testes. O objetivo foi desenvolver um sistema capaz de controlar uma maquete de dimensões 80cm x 60cm, através do *App Wanzeller's House* desenvolvido através da aplicação *TouchOSC Editor* que foi passado para o *iPhone* utilizado neste projeto. Antes de desenvolver o projeto, há um aprendizado bem grande a ser absorvido, pois o principal objeto de pesquisa, o Arduino, visto que as vastas aplicações para arduino não são vistas na universidade. Portanto, dividiu-se o projeto em algumas etapas principais, estas vão desde os estudos iniciais até a fase de conclusão do projeto. Portanto, para desenvolver o projeto decidiu-se utilizar as seguintes etapas: 1. Estudo de Metodologias; 2. Construção da Maquete; 3. Implementação Física do Protótipo; e 4. Criação do *App Wanzeller's House;* Sendo assim, estas etapas sequenciadas

foram seguidas de forma a se obter um melhor resultado para análise sobre o projeto aqui desenvolvido.

4.2.1. Estudo de Metodologias

Esta foi a etapa inicial, onde foi realizada uma vasta pesquisa acerca do assunto aqui abordado, bem como quais metodologias que poderiam ser utilizadas para se alcançar o que foi proposto pelo tema desta monografia. Após esta pesquisa inicial, a utilização da plataforma arduino mostrou-se extremamente viável para obtenção dos resultados esperados, pois existe grandes projetos acessíveis na área, além de ser de código aberto e possibilitar implementações funcionais e de baixo custo. Além de ser grande interesse do autor.

Logo após a escolha da plataforma a ser utilizada, foi necessário melhor compreendê-la, assim foram realizados estudos para se familiarizar com a IDE do arduino (Figura 3.7.), bem como suas funcionalidades e linguagens utilizadas. Em seguida, foram realizados uma série de experimentos iniciais para se obter compreensão dos componentes que seriam utilizados no projeto.



(a) (b)

Figura 4.5 - Testes Iniciais com Arduino: (a) Controlando Leds. (b) Testando o LM35

Fonte: Próprio autor

A figura 4.5 acima apresenta alguns dos testes iniciais que foram realizados no arduino na fase de aprendizado desta poderosa plataforma, a figura 4.5 (a) mostra a utilização de leds que podem ser utilizados para realizar a simulação de lâmpadas, visto que acendendo um led é possível acender uma lâmpada, bastando apenas utilizar um relé. A figura 4.5 (b) mostra o circuito que foi utilizado para realizar os primeiros testes com o sensor de temperatura LM35, a figura a seguir mostra o *scketch* que foi utilizado para testar o LM35.

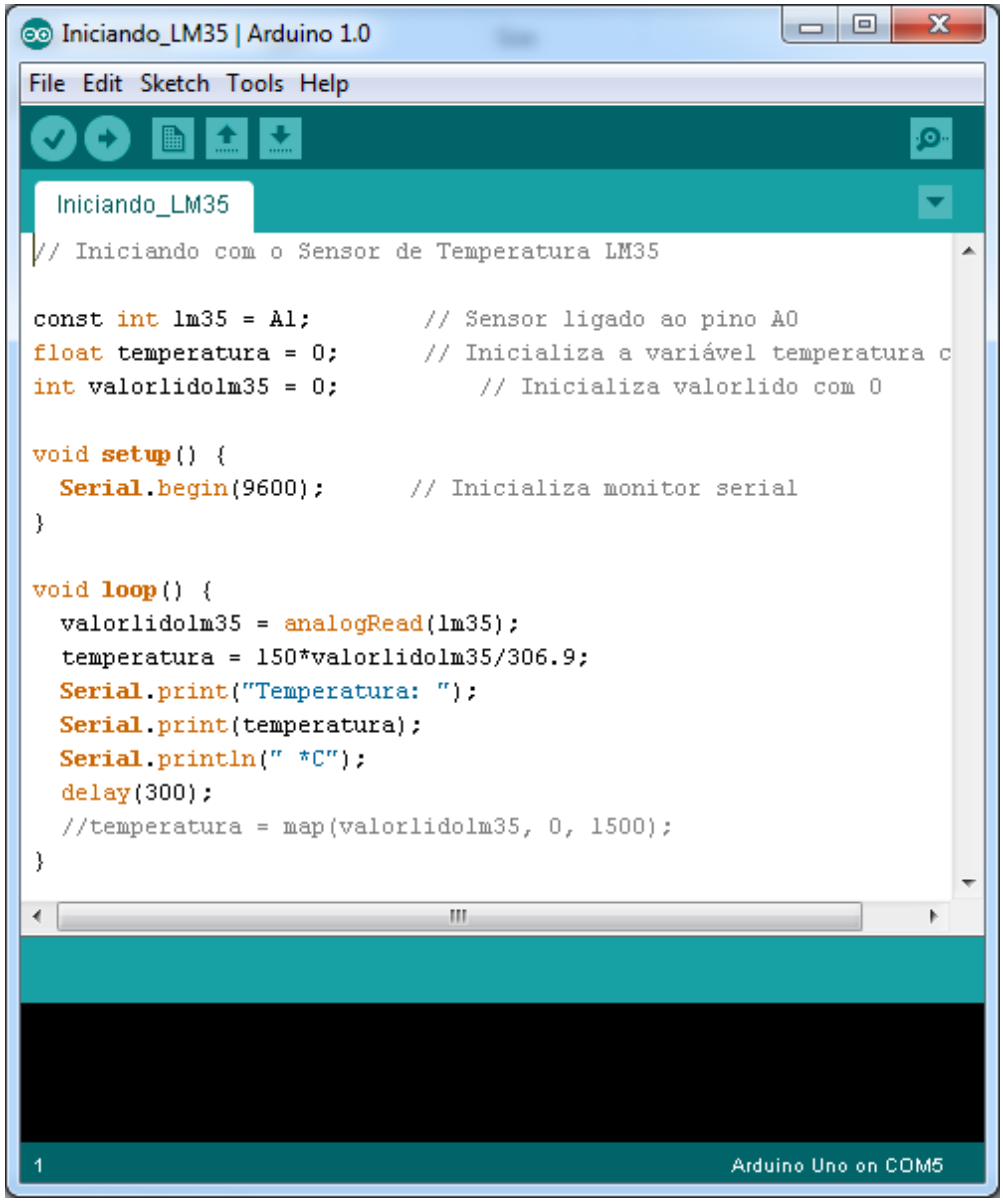


Figura 4.6 - *Scketch* utilizado para testar o LM35

Fonte: Próprio autor

Conforme explicado na seção 3.4.4. o LM35 possui em sua saída um sinal de 10mV para cada Grau Celsius de temperatura, por este motivo observa-se no código acima regra de três simples para converter o valor lido em graus Celsius. Assim como estes testes que foram apresentados na figura 4.5, foram realizados uma série de testes com outros componentes que seriam também utilizados no projeto, como módulo relé, sensor de movimento, sensor de luminosidade, entre outros.

Logo em seguida aos testes com os primeiros componentes, foi realizado um estudo sobre a utilização de um componente que seria uma peça chave neste projeto, que é o *Ethernet Shield*, conforme visto na seção 3.4.2 este pode conectar seu arduino à internet, sendo assim seria possível enviar comandos através de um celular conectado a rede. Então, assim como os

demais componentes a utilização do *Ethernet Shield* foi testada inicialmente com um código simples e posteriormente foi realizada a simulação mais complexa, esta simularia o funcionamento do protótipo que seria construído, onde os *leds* estavam simulando as lâmpadas, um *buzzer* simulando a sirene do sistema de alarme, o LM35 realizando a leitura da temperatura ambiente e um LDR utilizado para verificar a luminosidade ambiente, onde este último foi utilizado no projeto, como alternativa economicamente viável para monitorar o estado (ligada ou desligada) atual da lâmpada. A figurar a seguir apresenta o circuito que foi montado para esta simulação com uma protoboard.

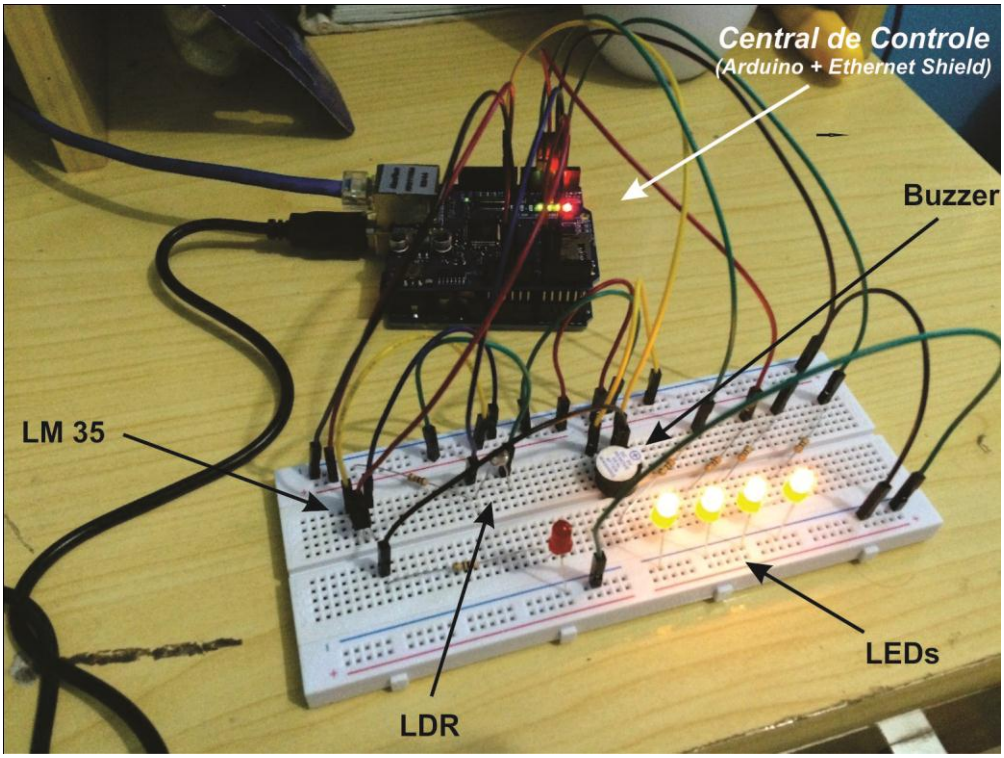


Figura 4.7 - Simulação inicial para o protótipo proposto

Fonte: Próprio autor

Vale ressaltar que até este momento o controle estava sendo realizado via página web, conforme mostra a figura abaixo.

Figura 4.8 - Interface de controle para a simulação inicial do sistema

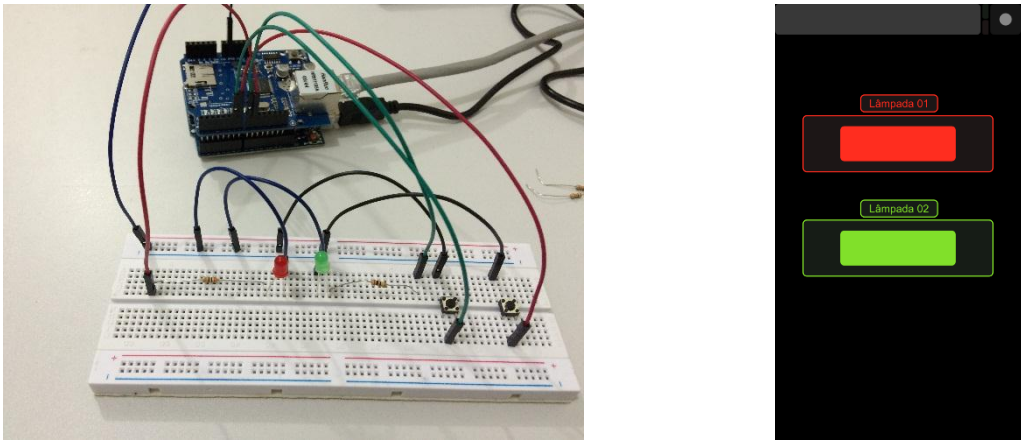
Fonte: Próprio autor

Estes testes iniciais foram de fundamental importância do decorrer do projeto, pois através deles foi possível encontrar alguns erros de projeto e assim buscar novas alternativas e solucionar os eventuais imprevistos, um desses pode ser observado na figura 4.8, onde o sistema está fazendo uma leitura incorreta da temperatura ambiente, neste caso, este foi causado por mau contato no circuito montado.

Assim, após os teste iniciais com o sistema sendo controlado via página web, constatou-se que este não seria o ideal para esta aplicação, pois o sistema estava configurado para atualizar 5 em 5 segundos, para que fosse possível atualizar o valor da temperatura, esse *refresh* leva a tela a carregar a cada 5 segundos, causando inconveniência na utilização. Outros dois fatores que determinaram a não utilização de uma página *web* como para o sistema de controle foram: fator segurança, visto que os comandos estavam sendo enviados ao fim da URL, que torna o sistema passível de invasão, pois bastaria qualquer pessoa ter acesso a rede e acessar a página, ou simplesmente digitar os comandos ao fim na URL utilizada (casatotiafo.no-ip.org/*comando*); e o segundo fator foi visual, pois a interface criada para o controle não se mostrava muito intuitiva e esteticamente agradável.

Assim, no decorrer das pesquisas surgiu a ideia de se criar um aplicativo de celular que se comunicasse com o sistema de controle e que fosse de fácil utilização e com um visual atraente. Fez-se então a utilização da aplicação *TouchOSC*, conforme detalhado na seção 3.3.

que trata acerca do mecanismo de comunicação que foi utilizado para este projeto, esta aplicação utiliza protocolo próprio e a comunicação é feita via *Wi-Fi.* A figura 3.10 mostra a tela inicial do *TouchOSC Editor*. Assim como os demais componentes a aplicação precisava ser testada, no **Blog Elétron Livre** [26] encontra-se um exemplo onde ensina-se o passo a passo para acender dois *leds* a partir de uma aplicação criado pelo *TouchOSC Editor* ou dois botões. Este experimento serviu como base para realizar os primeiros testes iniciais com esta ferramenta. A figura a seguir mostra o resultado desta aplicação inicial criada, onde a figura 4.9. (a) apresenta o circuito que foi montado e a figura 4.9. (b) mostra a interface gráfica criada para controlar este pequeno, porém importante experimento. Visto que a partir desse o autor foi instigado a pesquisar e conhecer melhor esta aplicação. Sendo assim, esta foi utilizada para desenvolver o que foi definido posteriormente de *App Wanzeller's House*, pois é de fácil implementação, configuração e sendo ainda possível a criação de uma interface muito intuitiva e agradável.



(a) (b)

Figura 4.9 - Iniciando com *TouchOSC Editor*: (a) Circuito. (b) Interface do App
Fonte: Próprio autor

Após todo o estudo de metodologias que poderiam ser utilizadas e realização dos testes com os componentes e diferentes aplicações de controle, passou-se para a segunda etapa do desenvolvimento do modelo proposto, construção da maquete que será utilizada.

4.2.2. Construção da Maquete

A segunda etapa no desenvolvimento deste projeto é a construção da maquete residencial para aplicação e simulação prática do sistema de controle e automação residencial que está sendo apresentado. Como futuramente pretende-se implementar este sistema na residência real, a casa que foi utilizada como base é a residência do autor do trabalho, que tem uma sala, três quartos, um banheiro e uma cozinha. A maquete da residência foi construída com

dimensões de 80 cm de comprimento, 60 cm de largura e 25 cm de altura. Para projetar esta maquete foi utilizado um software de engenharia para este fim, a figura a seguir mostra a planta baixa da residência já com as dimensões.

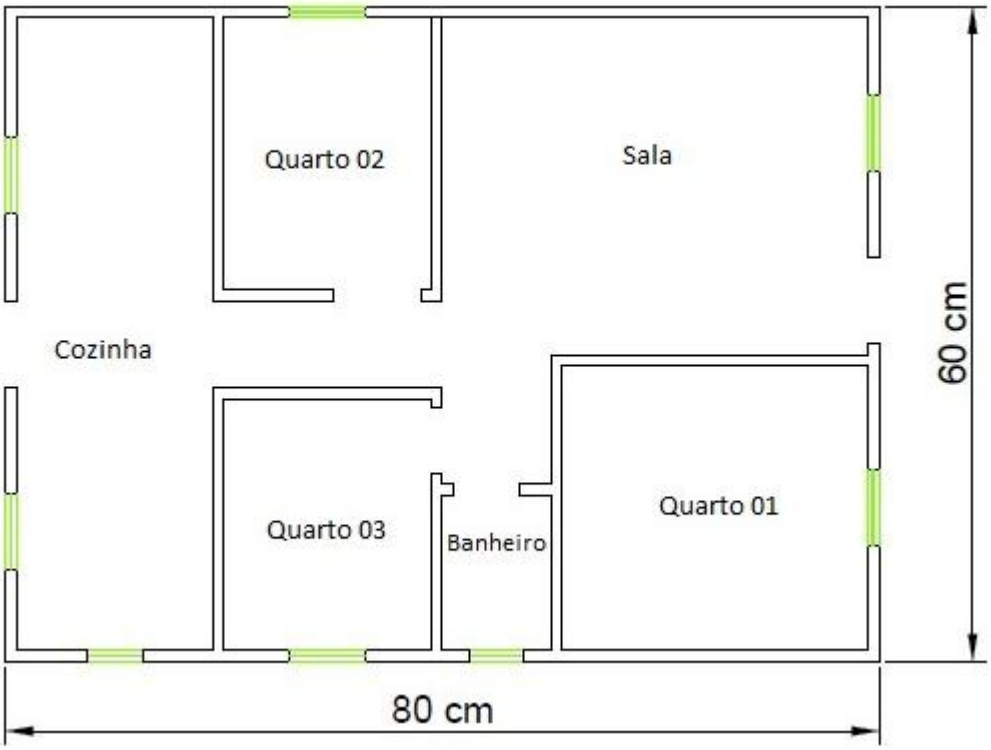


Figura 4.10 - Planta baixa da maquete

Fonte: Próprio autor

Em seguida, ao término de projetar a planta baixa da casa, foi necessário projetar a maquete em 3 dimensões (3D), pois facilitaria na construção da mesma. Vale ressaltar que neste ponto foi pensado em criar um fundo falso com 6 cm de altura em baixo da planta da residência, onde neste espaço seriam colocados todos os equipamentos que seriam utilizados no projeto, como toda a parte elétrica e controle do sistema, tais como a central de controle (Arduino + *Ethernet Shield*), protoboard com todo o circuito do sistema, além da sirene e bateria 12V utilizadas no sistema de alarme. Sendo assim, de acordo com a dimensões desejadas, fez-se o uso novamente de um software de engenharia para a modelagem da maquete em 3D. A figura 4.11 apresenta a maquete após ser projetada em três dimensões.

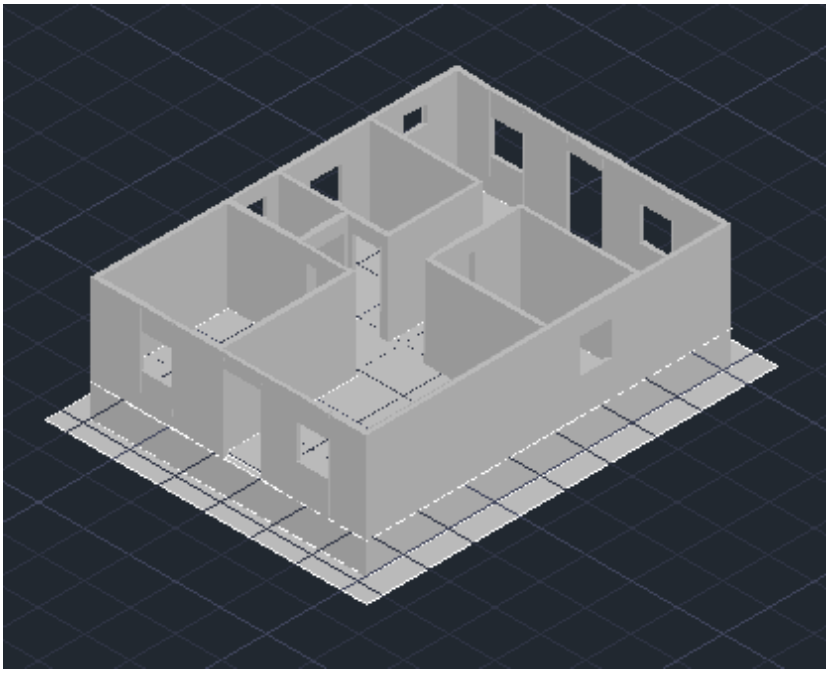
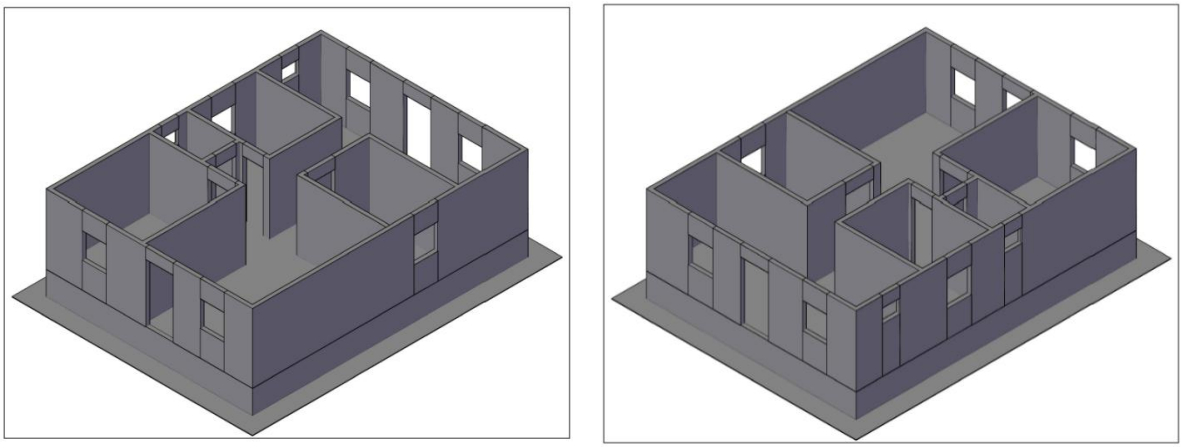


Figura 4.11 - Maquete projetada em 3D

Fonte: Próprio autor

Para dar mais detalhes e precisão na maquete a figura abaixo apresenta a vista em 3D para diferentes ângulos.



(a) (b)

Figura 4.12 - Maquete 3D. (a) Vista de Frente; (b) Vista de Fundo

Fonte: próprio autor

Após o término do projeto da maquete, partiu-se então para construção física na maquete que foi projetada, utilizou-se uma folha de compensado adquirida em uma loja de materiais de construção com dimensões de 2,20 m x 1,6 m x 10 mm e esta foi o suficiente para construção da referida maquete que seria utilizada no projeto de automação residencial apresentada nesta monografia. A figura a seguir apresenta a construção da maquete em sua fase inicial.

Figura 4.13 - Construção da Maquete

Fonte: Próprio autor

Terminada a montagem inicial da maquete, com paredes, portas, janelas e até mesmo telhado (removível). O passo seguinte foi criar uma forma alocar toda a parte elétrica e equipamentos na parte "embaixo" da casa, sendo assim foram colocadas na parte de trás da maquete dobradiças para que se pudesse levantar a casa e ter acesso a parte interna e um pequeno furo para passar os cabos de alimentação da rede para as lâmpadas e o cabo de rede par trançado para conectar o a central de controle (Arduino + *Ethernet Shield*) na internet. Logo em seguida foram feitos "buracos" no chão dos cômodos da casa para encaixar as lâmpadas e os interruptores que fazem parte do projeto de automação residencial. Pois, as lâmpadas poderão ser acionadas tanto pelo *App Wanzeller's House* como pelo sistema convencional (interruptores), uma característica importante deste projeto, visto que a residência não vai ficar dependente do sistema de automação projetado, pois se o sistema por algum motivo falhar o sistema convencional pode ser utilizado. A figura seguir apresenta os furos no interior de cada

cômodo e a forma para se ter acesso a parte interna onde será passado a parte elétrica e será colocado a central de controle.

Figura 4.14 - Parte interna da maquete

Fonte: Próprio autor

E para finalizar a etapa de construção da maquete, a mesma foi pintada para dar um grau de realidade e beleza ao projeto, a figura 4.15 mostra a pintura da maquete.

Figura 4.15 - Pintura da Maquete

Fonte: Próprio autor

E por fim a figura a seguir mostra como ficou a maquete construída depois de pintada. Com isso, concluiu-se com êxito a segunda etapa do desenvolvimento deste projeto, que foi a

construção da maquete. Agora a etapa seguinte será a implementação física da parte elétrica, central de controle e demais componentes do projeto.

Figura 4.16 - Maquete finalizada

Fonte: Próprio autor

4.2.3. Implementação Física do Protótipo

A partir do término da construção da maquete deu-se início a fase de implementação física dos componentes que fariam parte do projeto, nesta etapa será mostrado como foi realizada a montagem dos componentes que seriam utilizados em cada uma das funcionalidades do sistema, que como apresentado anteriormente consiste em: Controle do sistema de iluminação da residência; Monitoramento da temperatura; e Implementação de um sistema de alarme residencial.

4.2.3.1. Controle do Sistema de Iluminação

O protótipo apresentado aqui possui como principal característica poder controlar o sistema de iluminação da residência, o usuário poderá ligar ou desligar as lâmpadas via *iPhone*, além de poder contar com um sistema de iluminação dimerizável, onde através deste, o usuário pode controlar a intensidade luminosa de determinada lâmpada. O sistema de controle de iluminação possui também um sistema de iluminação RGB, onde o usuário pode personalizar determinado ambiente através da variação dessas três cores (**R**ed, **G**reen e **B**lue).

Sendo assim, o controle do sistema de iluminação terá três funcionalidades inerentes: Permitir ligar ou desligar um lâmpada via dispositivo móvel; Dimerizar determinada lâmpada; e controlar um iluminação RGB em determinado ambiente. Com isso, dividiu-se essas três características na residência, onde a iluminação da sala será dimerizável, a iluminação do quarto 01 será a iluminação RGB e os outros cômodos (quarto 02, quarto 03, banheiro e cozinha) da

casa terão lâmpadas fluorescentes convencionais sendo acionadas tanto pelo celular ou pelo interruptor.

As lâmpadas do quarto 02, quarto 03, banheiro e cozinha poderão ser acionadas tanto via celular quanto através de um interruptor. Sendo assim, para implementar este controle é necessário ter conhecimento e cuidado em se trabalhar com instalações elétricas residências, neste caso fez-se a utilização do esquema de ligação em modo *Three Way*, onde o módulo relé 5V descrito na seção 3.4.3 substituirá um dos interruptores, isso torna possível que a lâmpada seja acionada ou pelo interruptor ou pelo arduino, sendo que este último pode controlar o relé. A figura a seguir apresenta este esquema de ligação.

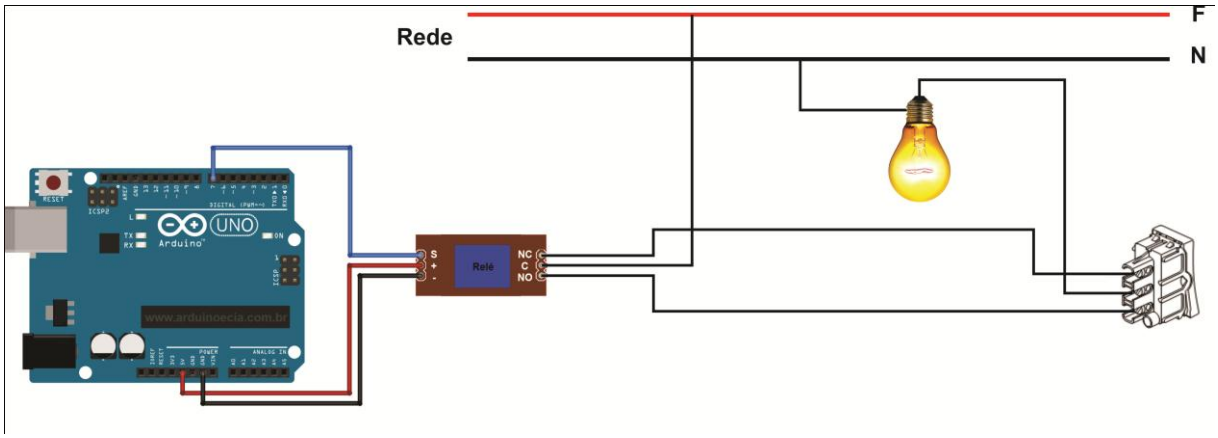


Figura 4.17 - Esquema de ligação das lâmpadas

Fonte: Adaptado pelo autor

Partindo desse conceito, a figura 4.18 apresenta o esquema que foi montado na maquete. Utilizou-se para esta etapa do projeto canaletas para esconder a fiação existente, cabos de 1.5 mm nas cores vermelho (neutro) e azul (fase), interruptores paralelos, lâmpadas fluorescentes de 14 W e módulos relés 5V, todos esses materiais exceto o módulo relé podem ser adquiridos facilmente em loja de materiais de construção, optou-se por adquirir os mais baratos, pois atendem a necessidade.

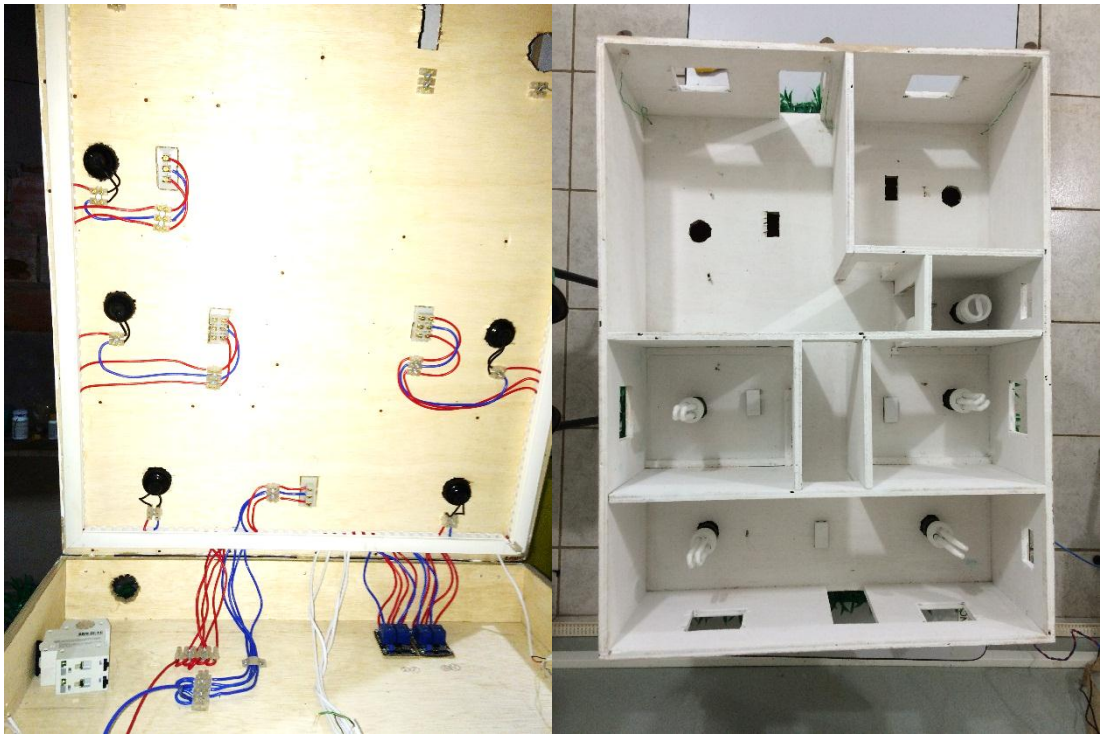


Figura 4.18 - Interruptores e relés montados

Fonte: Próprio autor

Através da figura 4.18 observa-se que existem duas lâmpadas na cozinha, devido esta ser bem maior que os demais cômodos da casa optou-se por utilizar duas lâmpadas em paralelo sendo controladas pelo mesmo interruptor, fazendo a utilização de mais um conceito de instalações elétricas residenciais.

Para dar mais segurança a este projeto, visto que trabalhará com a energia da rede, foi colocado na parte lateral da maquete um disjuntor de proteção bifásico, onde este além de proteger todo o circuito simula o padrão de entrada monofásico de uma residência real. A figura a seguir mostra a inserção deste disjuntor de proteção.

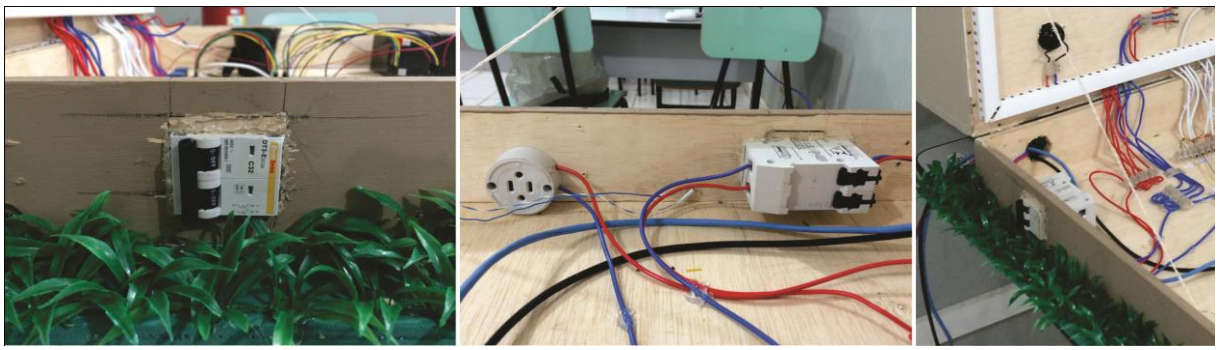


Figura 4.19 - Disjuntor de proteção na maquete

Fonte: Próprio autor

Desta forma o sistema irá funcionar de maneira segura caso ocorra algum curto circuito durante a implementação e testes. E para ligar o sistema da maquete a rede basta conectar à tomada. Como mostra a figura a seguir.

Figura 4.20 - Ligando a maquete à rede

Fonte: Próprio autor

Com este circuito montado será possível controlar as lâmpadas através do interruptor ou do relé. Entretanto, desta forma não será possível monitorar o status (ligada ou desliga) das lâmpadas, pois a central de controle consegue apenas monitorar e controlar o estado do relé (NA ou NF), porém não consegue identificar quando o interruptor foi pressionado, mudando o status da lâmpada. Uma alternativa economicamente viável encontrada para solucionar este impasse foi a utilização de sensores de luminosidades (LDR’s) para monitorar o status das lâmpadas, estes são facilmente encontrados em qualquer eletrônica e ainda são muito baratos.

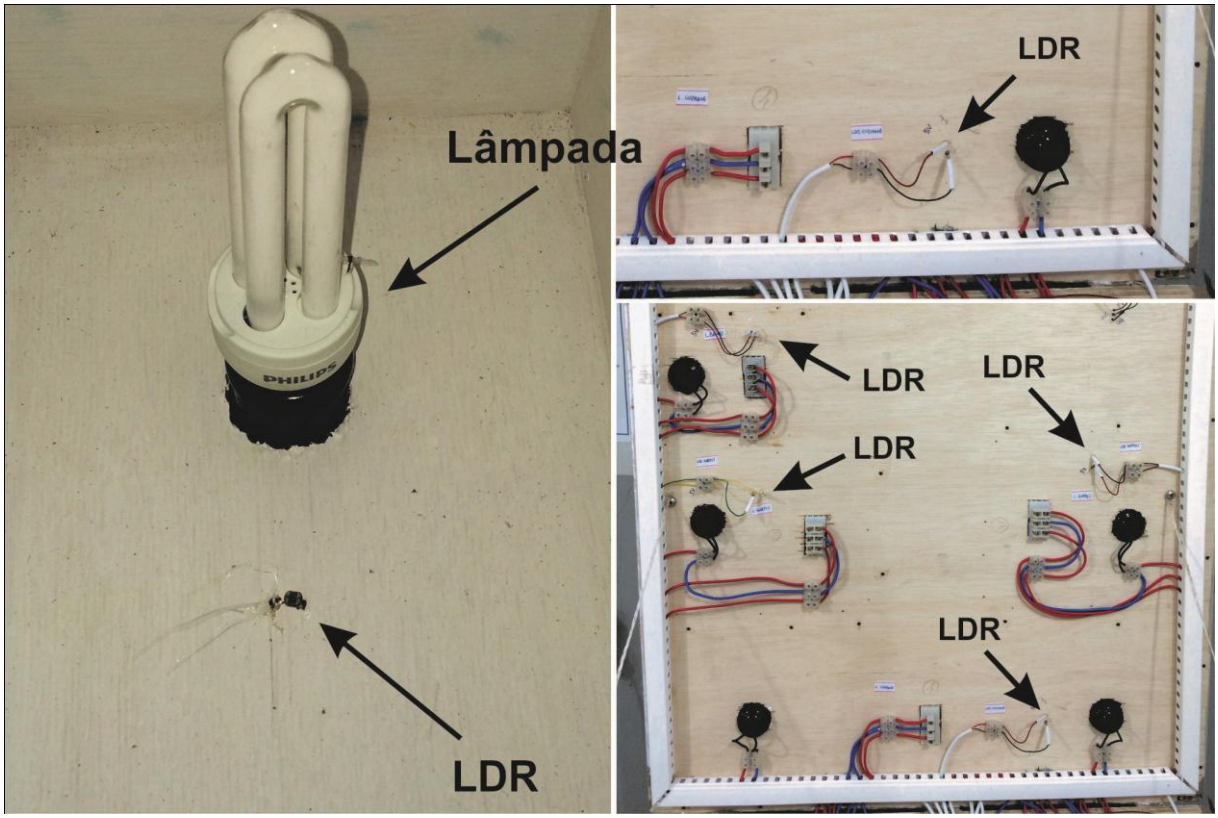


Figura 4.21 - LDR's inseridos próximo as lâmpadas

Fonte: Próprio autor

A figura acima mostra que os sensores de luminosidade foram inseridos próximos a cada lâmpada, onde estes serão utilizados para monitorar o status desses ambiente, ou seja, ao ligar a lâmpada pelo interruptor, mesmo que o arduino não consiga identificar que o interruptor foi pressionado, ao ligar a lâmpada será emitido uma forte intensidade luminosa sobre o sensor. Sendo assim, o arduino recebe esta informação e através da programação na central de controle, quando for detectado forte intensidade luminosa sobre o sensor, esta indicará que a lâmpada está ligada e consequentemente a central envia o comando com o novo status da lâmpada para o aplicativo, assim o usuário saberá quais lâmpadas estão ligadas em sua residência. Esta é mais uma característica importante deste projeto, pois o usuário pode a qualquer momento saber se a lâmpada de determinado cômodo está ou não ligada, mesmo não estando próximo a lâmpada. Isso é muito útil em residência em que existem portadores de alguma deficiência física, pois se alguém esquecer a lâmpada ligada o usuário pode desligá-la via *app* sem a necessidade de se deslocar até o referido cômodo da casa.

Em seguida, ao término da implementação dos componentes que foram utilizados para controlar e monitorar as lâmpadas da cozinha, banheiro, quarto 02 e quarto 03, foi implementado o sistema de iluminação dimerizada na sala de estar, para este tipo de iluminação

foi utilizado um LED alto brilho na cor branca, onde através do aplicativo o usuário poderá controlar a intensidade luminosa simulando um dimmer. O arduino trabalha com tensão de 0 a 5V, sendo através da porta PWM possível variar a intensidade luminosa do LED. Vale ressaltar que foi utilizado um LED para simular a iluminação dimerizada neste projeto, mas em ambiente real poderíamos utilizar um circuito eletrônico extra que utiliza TRIAC e um acoplador óptico na saída do arduino para controlar uma lâmpada convencional. A figura 4.22 mostra como foi inserido o LED na sala que simularia a iluminação dimerizada.

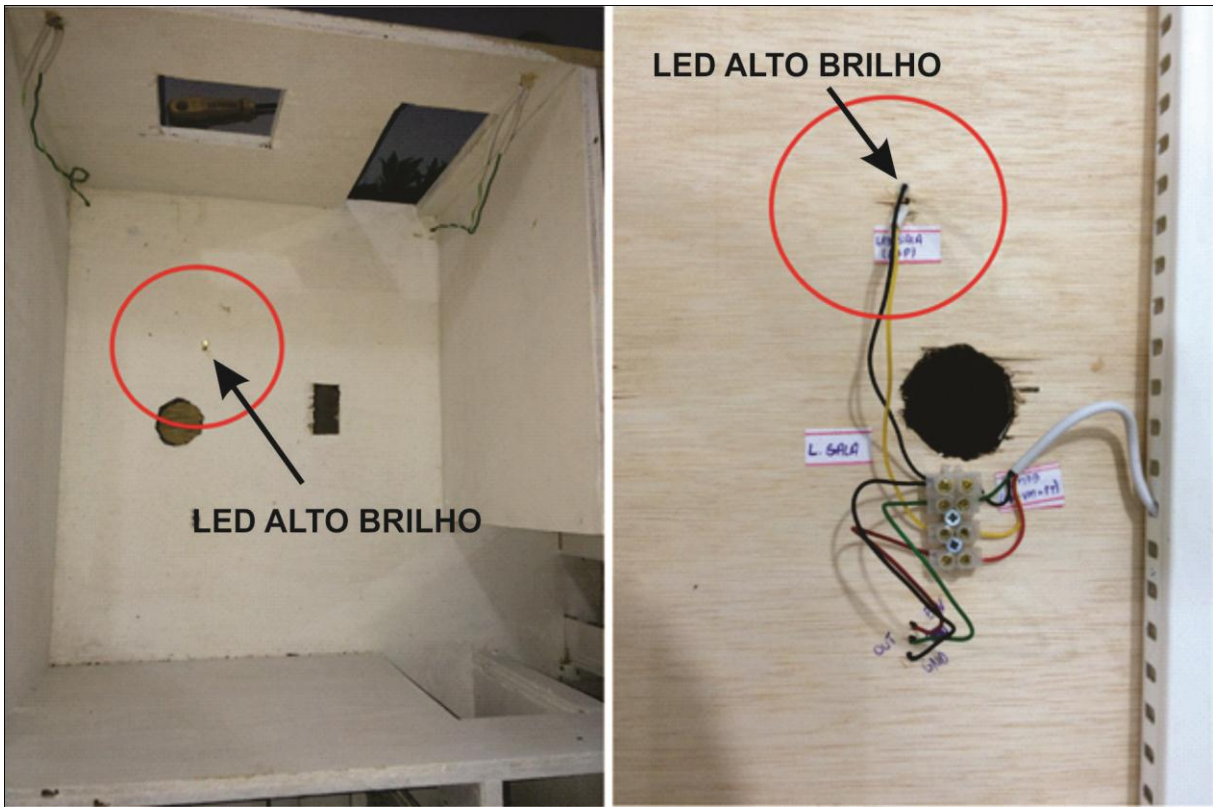


Figura 4.22 - LED alto brilho inserido na sala

Fonte: Próprio autor

Em seguida, partindo do mesmo princípio da simulação da iluminação dimerizada na sala, utilizou-se um LED RGB na quarto 01, este nada mais é do que três LED's de alto brilho embutido em um só, assim ele possui quatro pernas onde a perna maior é comum a todas as cores. E através do sistema de controle disponibilizado no *App Wanzeller's House* o usuário poderá controla e variar a intensidade luminosa nessas três cores, vermelho (Red), verde (Green) e azul (Blue), podendo acentuar os detalhes arquitetônicos do quarto ou criar um clima especial, seja ele romântico ou festivo. A figura a seguir apresenta como foi inserido o LED RGB na maquete.

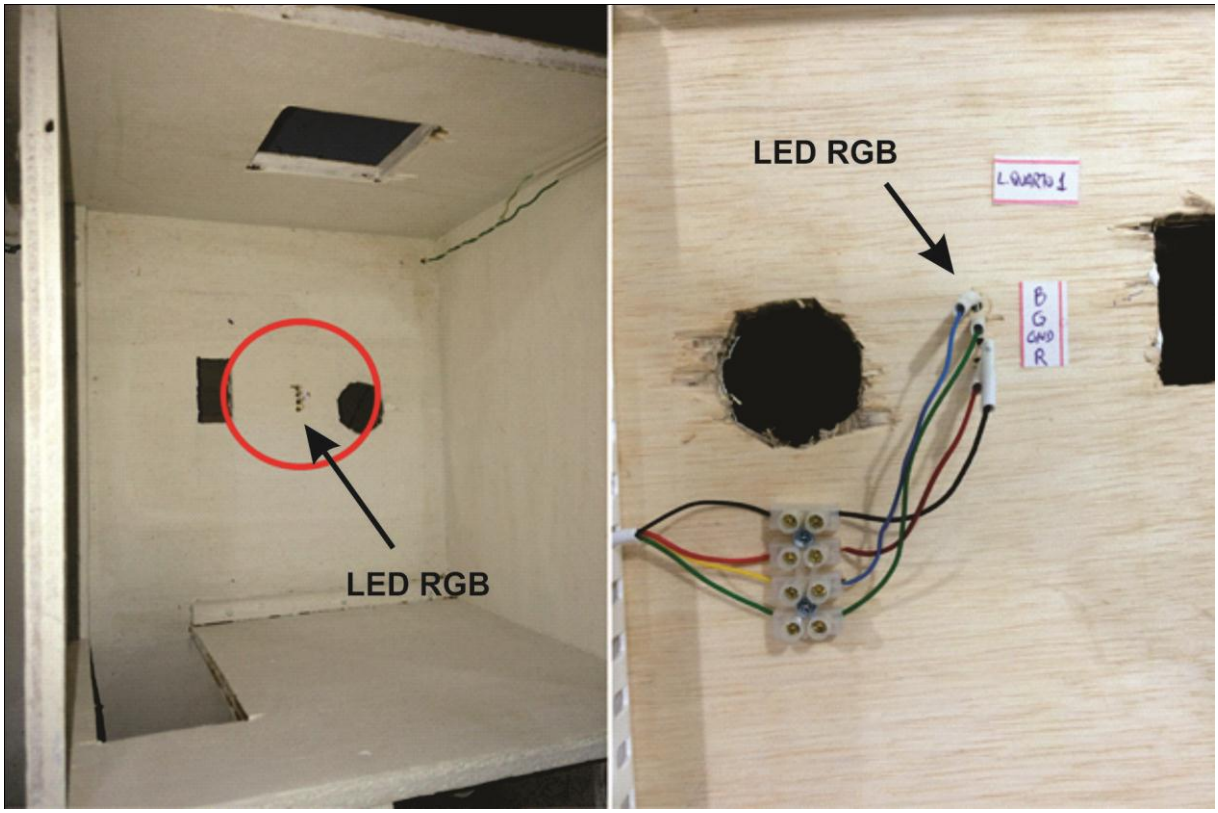


Figura 4.23 - LED RGB inserido na maquete

Fonte: Próprio autor

4.2.3.2. Monitoramento da temperatura

O sistema de monitoramento da temperatura é o mais simples para se implementar (ver figura 4.3), pois utiliza apenas um LM35, este possui três pernas, onde as pernas laterais são conectados o 5V e o GND do arduino e a perna central é conectado a perna analógica do arduino onde será repassado o valor analógico lido, cabendo a central de controle converter o valor lido para graus Celsius e enviar para o dispositivo móvel. A figura 4.24 mostra o sensor de temperatura inserido na sala para monitorar este ambiente. Para este módulo foi utilizado apenas um sensor monitorando a sala de estar, contudo poderia ser utilizado outros sensores para monitorar todos os ambientes da sala, pois o princípio seria o mesmo. Mas de forma a diminuir o números de portas utilizadas no arduino foi utilizado apenas um sensor, este monitorando a temperatura ambiente da sala.

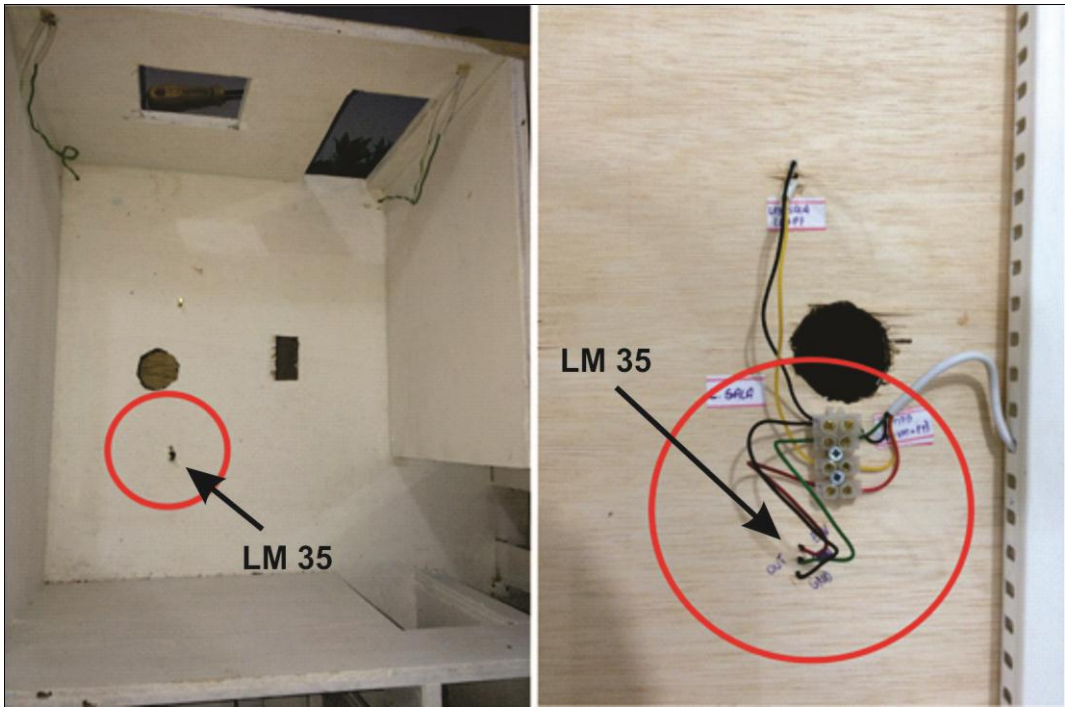


Figura 4.24 - Sensor de temperatura

Fonte: Próprio autor

4.2.3.3. Sistema de Alarme

A implementação física do sistema de alarme consiste em instalar o sensor de movimento em um local estratégico e a inserção da sirene juntamente com bateria que a alimenta. Sendo assim, optou-se por inserir o sensor PIR na parte da frente da casa, pois a residência possui muros que a cercam. Assim, quando o alarme for ativado pelo usuário através do celular, o sensor vai verificar se há algum movimento em seu raio de visão, caso positivo uma saída alta (*HIGH*) é emitida para a central e esta aciona a sirene e envia uma notificação via *tweet* para o usuário. A figura logo abaixo apresenta como ficou a inserção do sensor de movimento na parte frontal da maquete e internamente a sirene e uma bateria 12 V para alimentar a sirene.

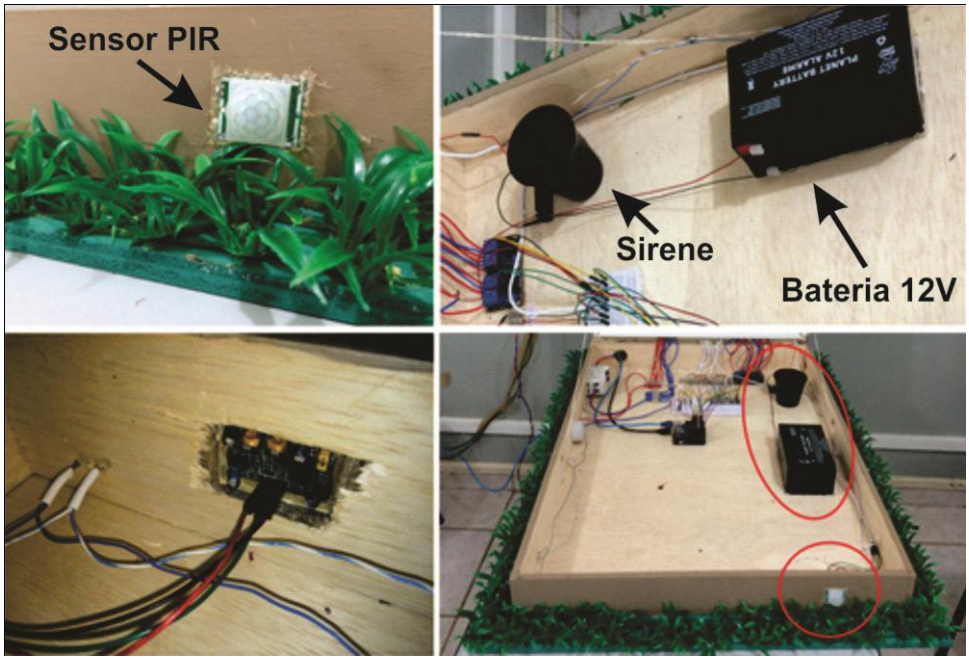


Figura 4.25 - Implementação dos componentes do sistema de alarme

Fonte: Próprio autor

4.2.3.4. Central de Controle

Para finalizar a montagem dos componentes na maquete, estava faltando o principal peça do projeto, que é a central de controle (Arduino + *Ethernet Shield*). Sendo assim, antes disso é necessário dispor todos a fiação de forma que ficasse acessível a central de controle, dessa forma utilizou-se canaletas para organizar toda a fiação elétrica e de sensores até a base do compartimento interno e assim colocamos conectores sindal para melhorar na organização e facilitar o acesso a central de controle. A figura a seguir mostra como ficou essa organização de cabos nas canaletas e a utilização de conectores que darão a central de controle acesso aos componentes comandados.

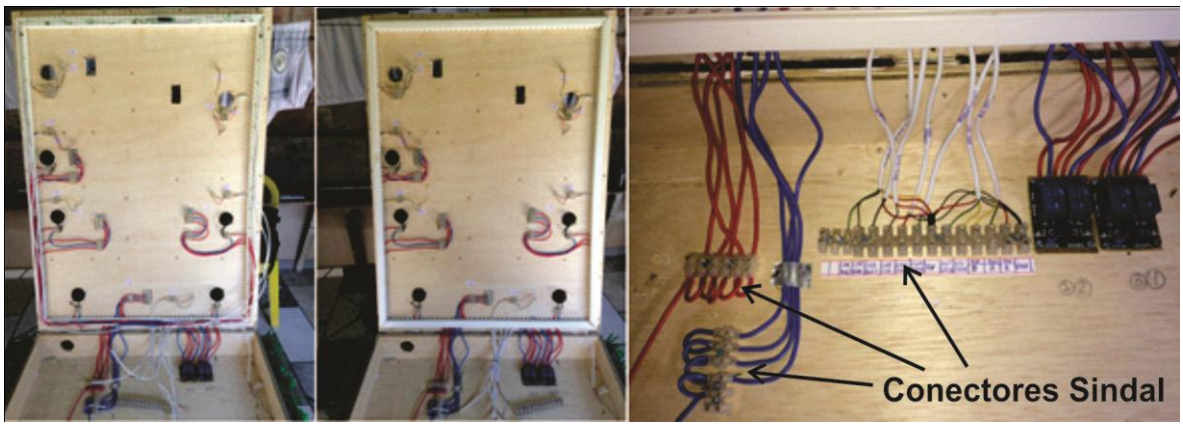


Figura 4.26 - Organização da fiação

Fonte: Próprio autor

Assim, com todos os cabos organizados e fixos, faltava apenas inserir próximo a eles o *protoboard* e o arduino com o *ethernet shield* conectado a ele. Foi necessário fixar estes componentes para que não mudassem de lugar podendo danificar o circuito, então o arduino foi parafusado na madeira e a protoboard foi fixada com cola quente assim como as conexões dos fios jumpers que se conectaram nela. E para o sistema funcionar independentemente de computador, foi inserido na parede lateral deste compartimento um tomada onde será conectado uma fonte 9V que fará a alimentação do arduino, substituindo assim o cabo USB que o alimenta. A figura a seguir mostra como foi realizada a inserção da central de controle na maquete e também a tomada que irá alimentá-la.

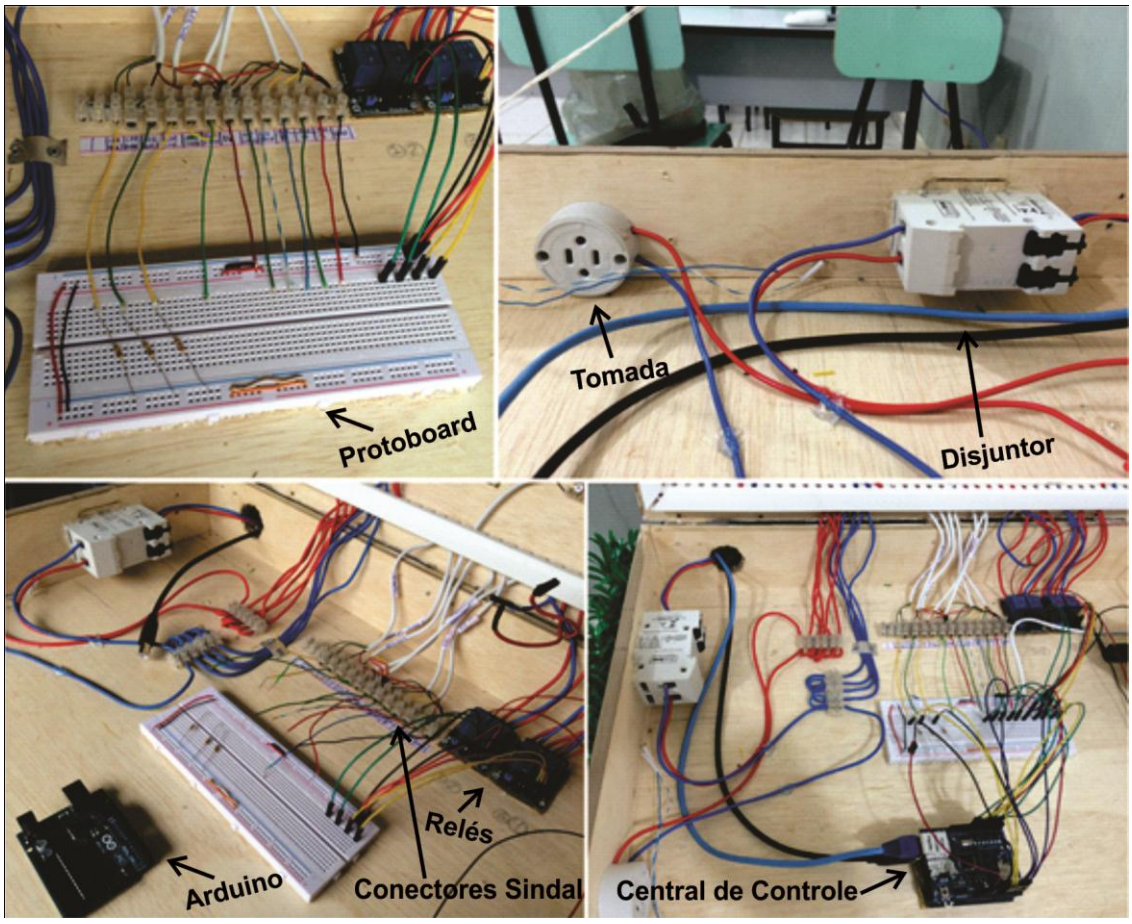


Figura 4.27 - Central de controle inserida na maquete

Fonte: Próprio autor

Assim, finalizou-se a terceira etapa do desenvolvimento do projeto, a figura 4.28 mostra como ficou a maquete com todo o sistema pronto e montado. Então, neste ponto do projeto bastava desenvolver a aplicação no *TouchOSC Editor* bem como seu código no arduino.

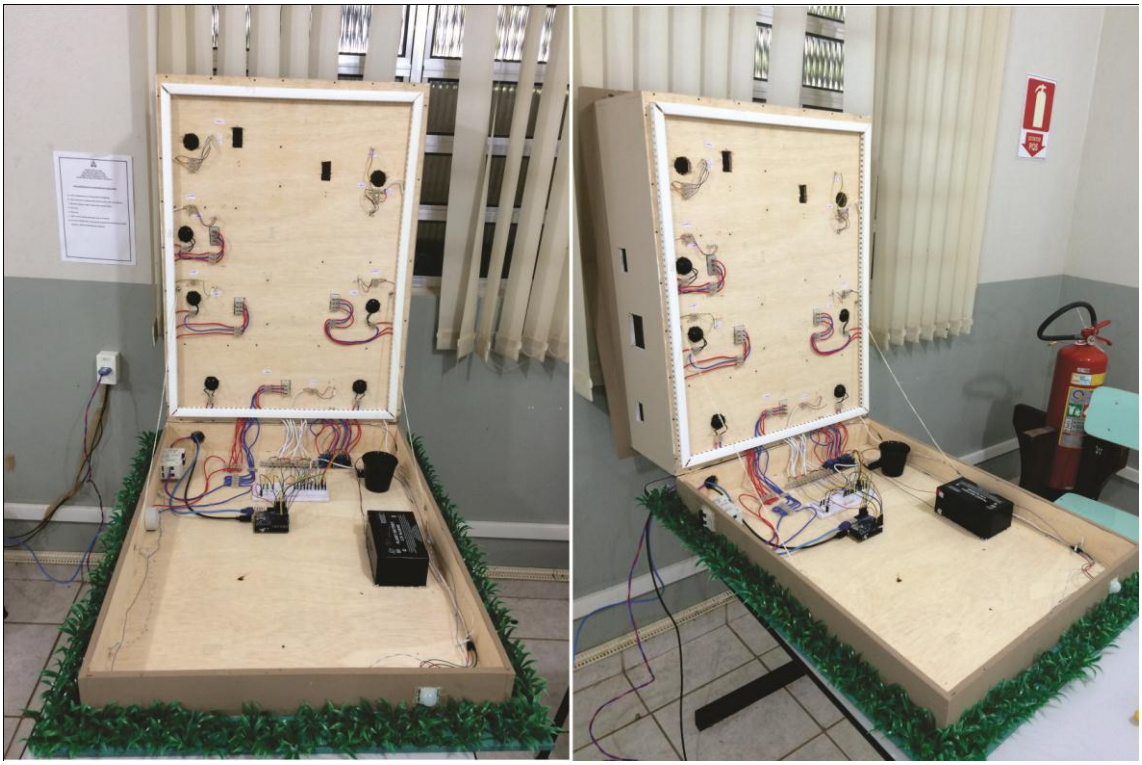


Figura 4.28 - Maquete com todo sistema já implementado

Fonte: Próprio autor

4.2.4. Criação do *App Wanzeller's House*

Nesta etapa do projeto foi utilizada aplicação *TouchOSC Editor* para se criar o *App Wanzeller's House,* será apresentado toda forma de criação do interface e configuração para a comunicação entre o dispositivo móvel de comando (*Iphone/Ipad/Ipod/Android)* e o Arduino que é o núcleo da central de Automação. Para estabelecer essa comunicação foi o protocolo OSC (*Open Sound Control*) que é uma evolução do protocolo MIDI adaptada para transporte via TCP/IP. As principais vantagens de usar esse protocolo são as seguintes:

- Apresenta resposta rápida aos comandos
- Biblioteca ARDOSC disponível para Arduino
- Disponibilidade de ferramenta simples para o desenvolvimento de interfaces gráficas customizadas para *Iphone, Ipad, Ipod,* e *Android* chamada *TouchOSC Editor*.
- Com essa ferramenta, não é necessário conhecer muito sobre programação IOS para desenvolver interfaces gráficas de comando para automação.

A biblioteca ARDOSC trabalha em conjunto com a biblioteca *Ethernet* oficial do Arduino, sendo assim qualquer *shield ethernet* baseado no chip W5100 deve funcionar sem problemas. A versão da biblioteca ARDOSC que foi utilizada neste projeto foi a disponibilizada no **Blog Elétron Livre** [26], e esta é compatível com a IDE 1.0 do Arduino, sendo que esta

versão da IDE do Arduino que foi utilizada neste projeto. Outra característica importante desta biblioteca é que ela torna bastante simples a configuração da comunicação OSC no Arduino, pois além das definições de interface de comunicação, temos apenas que definir quais são as rotinas que devem ser executadas para cada comando OSC recebido.

Sendo assim, para se criar o aplicativo que posteriormente será transferido para o celular que controlará o residência foi necessário primeiramente fazer o *download* gratuito do *software TouchOSC Editor* no site do desenvolvedor, depois de baixar e instalar a versão deste aplicativo compatível com o sistema operacional utilizado deu-se início a criação do app. A figura 3.10 mostra a tela inicial deste *software.*

Para a criação desta aplicação de controle foi necessário conhecer a fundo esta ferramenta de desenvolvimento. O topo mostra a barra de ferramentas de edição, para a direita é a vista de edição com uma visualização do *layout* e para a esquerda do painel de propriedades que mostra informações contextuais e propriedades de qualquer objeto que está atualmente selecionado na vista de edição. A barra de ferramentas no topo da janela do editor detém ícones para todas as funções básicas de edição. A figura a seguir mostra estas funções básicas de edição.

(a) (b)

Figura 4.29 - Funções básicas de edição do *TouchOSC Editor*

Fonte: Próprio autor

Primeiro, através da figura 4.29. (a) podemos visualizar que existe ícones para criar um novo *layout*, a abertura de um *layout* existente e salvar / guardar uma cópia do *layout* atual. Em seguida, a figura 4.29. (b) mostra que há os ícones para o cortar / copiar / colar / apagar. Alguns deles podem ser desativados a qualquer momento, porque não há nada selecionado para recortar / copiar ou eliminar, ou porque a área de transferência está vazia e não há nada para colar. A figura a seguir mostra outros ícones com funções básicas.

(a) (b)

Figura 4.30 - Função de Zoom e Sincronização do *TouchOSC Editor*

Fonte Próprio autor

Estas são as configurações são para controlar o zoom e usando uma grade para ajudar com a edição de layouts. Quando a grade é ativado todas as operações de controlo realizadas com o mouse, como mover e redimensionar, se ajustará automaticamente para as linhas de grade. A resolução da grade pode ser ajustado usando o controle de número à direita da caixa de seleção da grade. Afigura 4.30. (b) mostra botões para começar a sincronizar o *layout* para um dispositivo e mostrando a caixa sobre com informações sobre a versão. Um *layout TouchOSC* tem duas propriedades que definem seu tamanho e orientação. Estas propriedades podem ser editadas no painel de propriedades à esquerda quando não há nada selecionado na área de edição.

TouchOSC permite alternar para uma página no aplicativo editor da mesma forma que faria no aplicativo no dispositivo, clicando com o botão esquerdo em qualquer um dos separadores na barra de guia na parte superior do layout. As páginas podem ser adicionados e removidos, clicando com o botão direito em qualquer uma das guias na barra de guia na parte superior do *layout* e selecionando Adicionar página ou Remover página da paleta. Conforme figura abaixo. Vale saber que ao remover uma página todos seus componentes também serão excluídos. Esta inserção de várias páginas permitiu que fossem criadas vários módulos (páginas) para nossa aplicação.

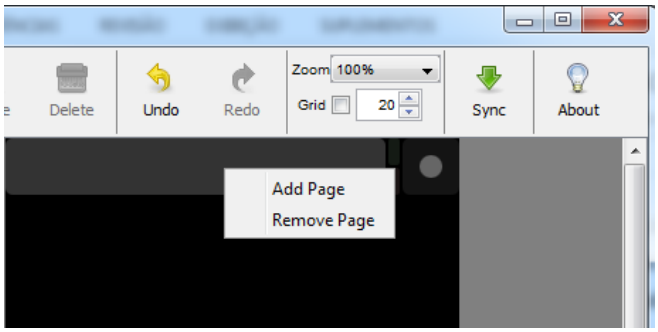


Figura 4.31 - Adicionando e removendo páginas

Fonte: Próprio autor

Sendo assim, após conhecer as funcionalidades básicas desta aplicação que foi utilizada para desenvolver o *App Wanzeller's House.* Primeiramente foi necessário customizar o *layout* do dispositivo que seria utilizado para o controle, em "*Layout>Size*" selecionou-se "*iPhone 5*" e em "*Layout>Orientation*" selecionou-se "*Vertical*". Estas são as configurações do aparelho que foi utilizado neste projeto, porém podendo ser personalizada de acordo com o dispositivo que foi utilizado para gerenciar a automação.

Com o intuito de tornar a aplicação de controle com uma interface simples, funcional e elegante, as funcionalidades do sistema foram divididas em módulos e cores. O *software*

TouchOSC Editor permite personalizar a tela de acordo com o desejado. Pois através dele é possível inserir textos (*Label*), botões (*Push Button* e *Toggle Button*), barra de fade (*Fader*) e nas configurações de cada elemento configurar os parâmetros que serão enviados para o arduino. Para isso basta clicar com o botão direito sobre a área de desenho e escolher qual item deseja ser adicionado ao aplicativo, conforme mostra a figura a seguir.

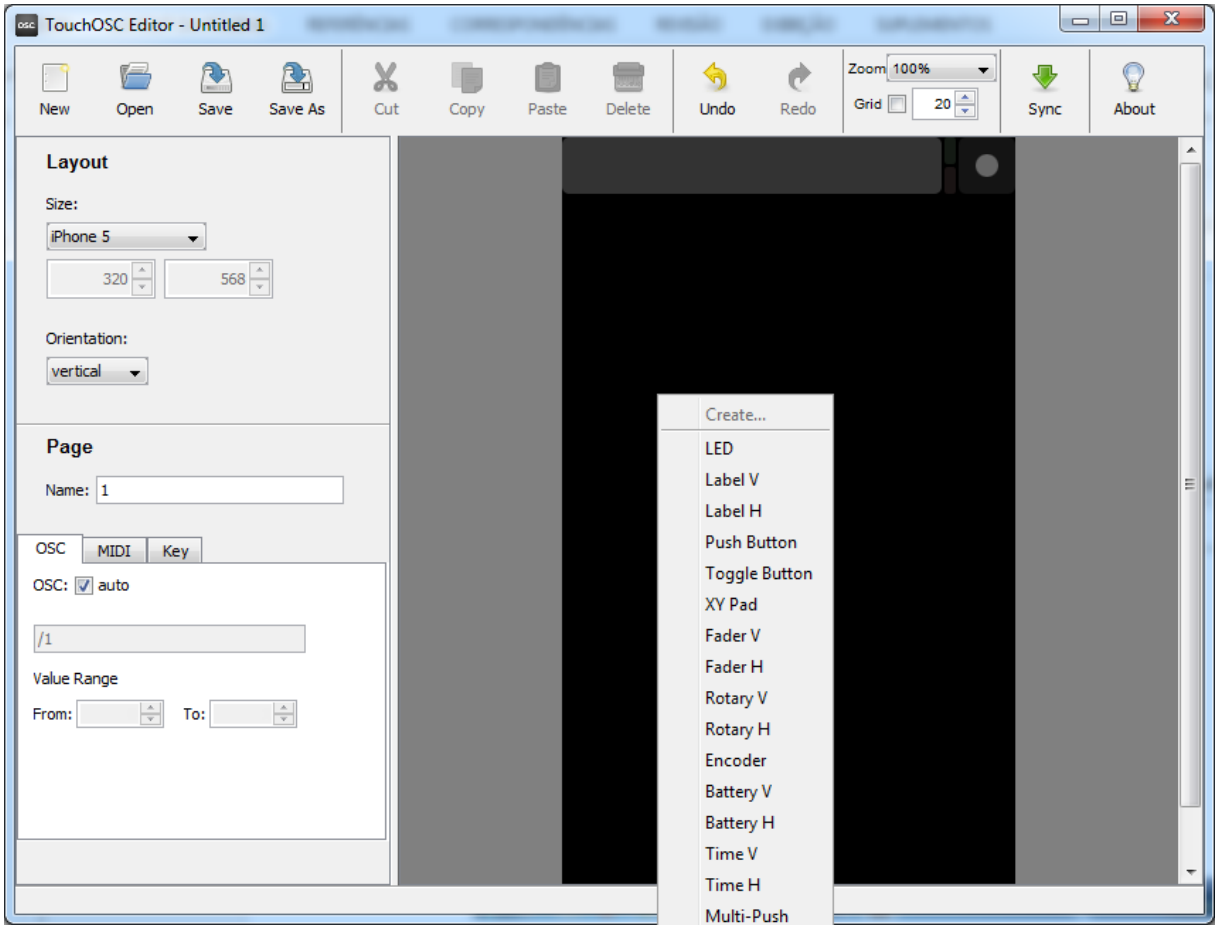


Figura 4.32 - Inserindo elementos no *TouchOSC Editor*

Fonte: Próprio autor

Criou-se então cinco módulos no aplicativo que foi desenvolvido, o primeiro módulo representado pela cor verde é o módulo *Iluminação Geral*, onde neste foram colocados botões (*Push Button*) que terão a função de realizar o acionamento das lâmpadas. O segundo módulo representado pela cor cinza é o módulo *Iluminação Sala*, este fará o controle de dimerização da iluminação da sala. O terceiro módulo representado pela cor amarela é o módulo *Iluminação Quarto 01*, neste módulo serão colocadas três barras de fade para controlar a variação de cores do LED RGB. O quarto módulo é o *Sistema de Alarme*, este módulo possuirá um botão onde o usuário pode ativar e desativar o sistema de alarme quando for conveniente. E por fim, o quinto

módulo do *App Wanzeller's House* é o módulo *Temperatura*, neste será inserido um *label* que receberá o valor da temperatura vindo do central de controle.

Para criar o primeiro módulo do aplicativo inseriu-se na tela de desenho alguns *Labels Horizontais*, os *labels* são rótulo que suas propriedades permitem alterar o texto que eles apresentam, sendo assim primeiramente foi criado o título do aplicativo e depois o título do módulo, na paleta de propriedades pode-se escolher a cor (*Color>Green*) que representa este módulo. A figura a seguir mostra o resultado deste processo.

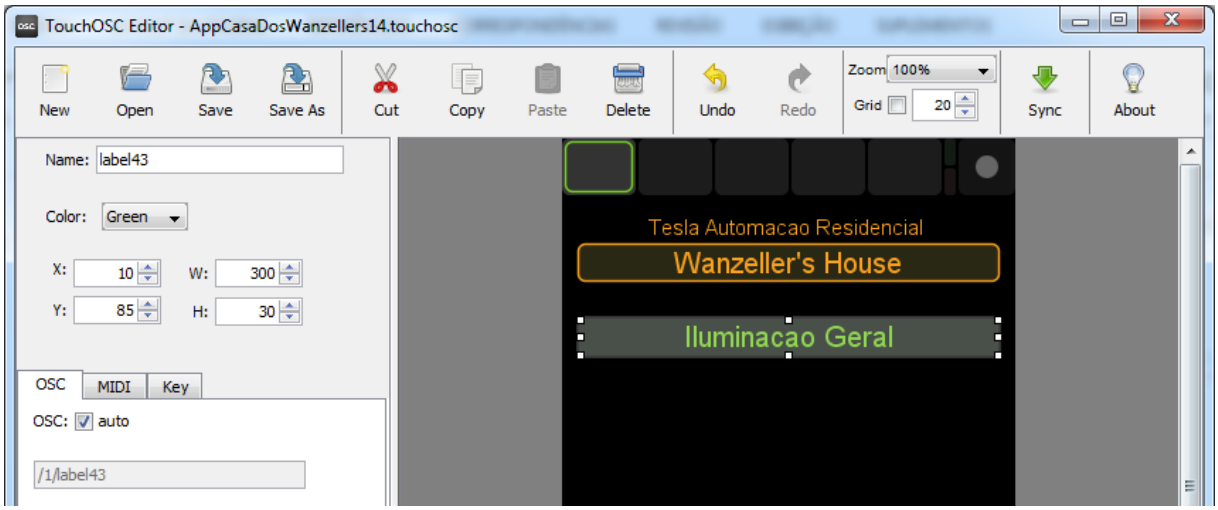


Figura 4.33 - Começando o desenvolvimento do aplicativo

Fonte: Próprio autor

Em seguida, inseriu-se o primeiro botão (*Push Button*) responsável pelo acionamento lâmpada da sala e ao lado deste um outro botão (*Toggle Button*) responsável pelo monitoramento do status desta iluminação. Conforme apresenta a figura a seguir. Neste ponto é importante realizar as configurações que identificarão para o arduino que este botão foi pressionado e quais valores serão repassados através da mensagem OSC. A figura a seguir mostra os botões de acionamento da lâmpada e de monitoramento de status.

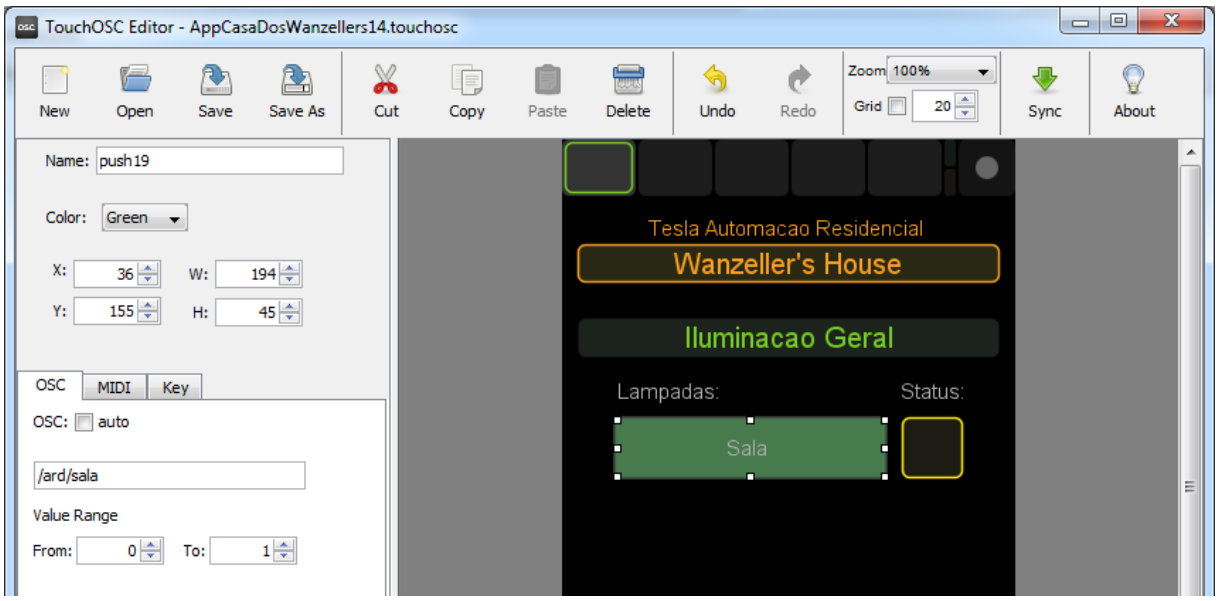


Figura 4.34 - Configurando botões de acionamento das lâmpadas

Fonte: Próprio autor

Ao clicar sobre o botão recém criado as propriedades deste aparecem ao lado esquerdo, o campo "*Name*" nos permite renomear este botão, podendo identificá-lo posteriormente. No campo "*Color*" selecionou-se a cor verde, em seguida desmarcou-se a opção "*Auto*" em OSC e na caixa de texto digita-se o comando que identifica esse botão na central de controle, para este botão definiu-se o comando "*/ard/sala*" e em "*Value Range*" foram definidos os argumentos que serão enviados (*From: 0* e *To: 1*) para a central de controle, ou seja, quando o *Push Button* for pressionado envia para o arduino através do comando "*/ard/sala*" o valor 1 e quando for solto envia o valor 0. Nas propriedades do botão que monitora o status da lâmpda o comando foi alterado para "*/ard/StatusSala*", pois através desse comando a central irá enviar o status da lâmpada. Da mesma forma foram criados os todos os outros botões, com um comando que os define e com seus valores. A figura a seguir mostra o módulo *Iliminação Geral* depois da inserção de todos os botões. Vale salientar que o acionamento das lâmpadas e os status são independentes, pois o botão envia o comando para ligar as lâmpadas, já os status são alterados pela central através dos valores lidos pelos sensores de luminosidade.

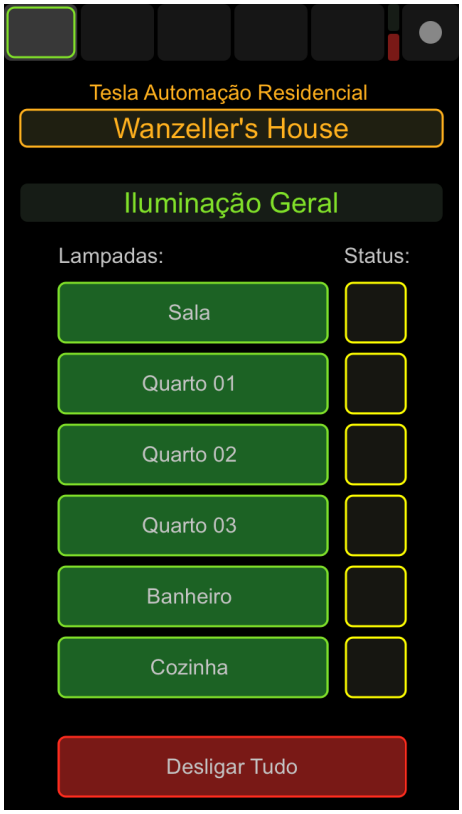


Figura 4.35 - Módulo 01: Iluminação Geral

Fonte: Próprio autor

Observa-se através da figura 4.35 um botão "*Desligar Tudo*", este botão foi inserido do aplicativo, pois permite o usuário em determinada ocasião desligar todas a iluminação da casa com apenas um clique, função extremamente útil quando deseja-se sair de casa. E através deste botão não será mais necessário ir de quarto em quarto desligando as lâmpadas que encontram-se ligadas, economizando assim tempo e dinheiro.

É extremamente importante configurar corretamente a faixa da valores que será passado e o comando definido, este comando será identificado pelo arduino indicando que determinado botão foi acionado, a figura a seguir mostra um pedaço do código onde os comandos que serão enviados através do aplicativos serão recebidos pelo arduino e identificado através deste comando configurado anteriormente, sendo assim cada comando define um função criada e cabe a esta função realizar os respectivos comando e acionamentos.

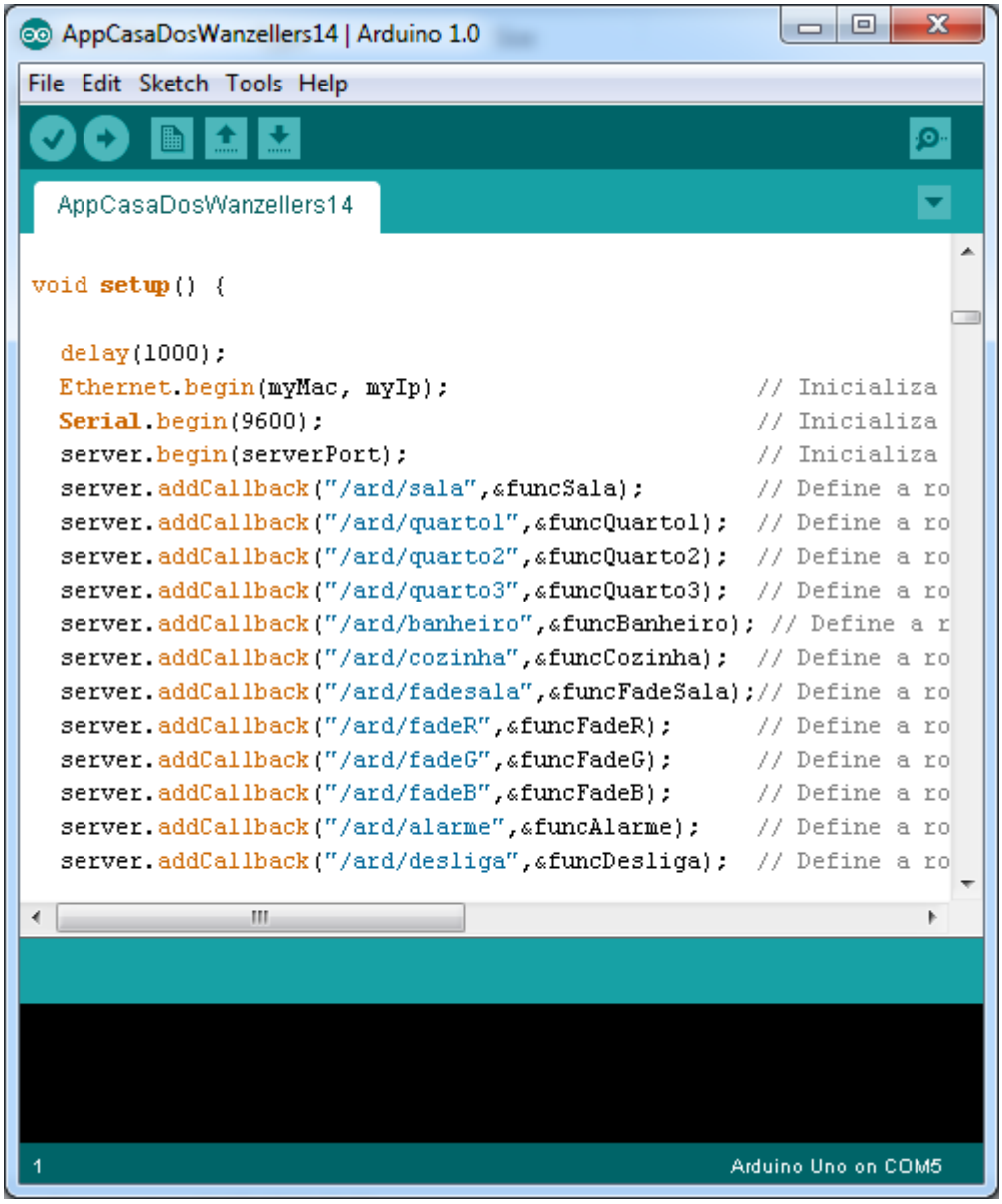


Figura 4.36 - Definição das funções referentes aos comando enviados

Fonte: Próprio autor

Todo o *scketch* do arduino utilizado neste trabalho está disponível no apêndice A e pode ser consultado com mais detalhes.

Em seguida, adicionou-se uma nova página para a construção do segundo módulo. Então, para manter o mesmo padrão, o título que foi criado no módulo 1 foi copiado para o módulo 2 e a este foi inserido um objeto di tipo "*Fader H"*, este será responsável para dimerizar o LED na sala de estar, após inserir este elemento foi necessário alterar algumas configurações deste objeto. A propriedade "*Color*" foi alterada para "*Gray*" que é a cor padrão escolhida para este módulo, desmarca-se a opção "*Auto*" em OSC e define o comando como "*/ard/fadesala*", em "*Value Range*" alterou-se o valor para variar entre 0 e 255, pois este valor será utilizado na

saída da porta PWM que controla o LED, variando assim sua intensidade luminosa, criando com isso o efeito de dimerização. A figura abaixo mostra todas as configurações que foram feitas e como ficou o módulo 2.

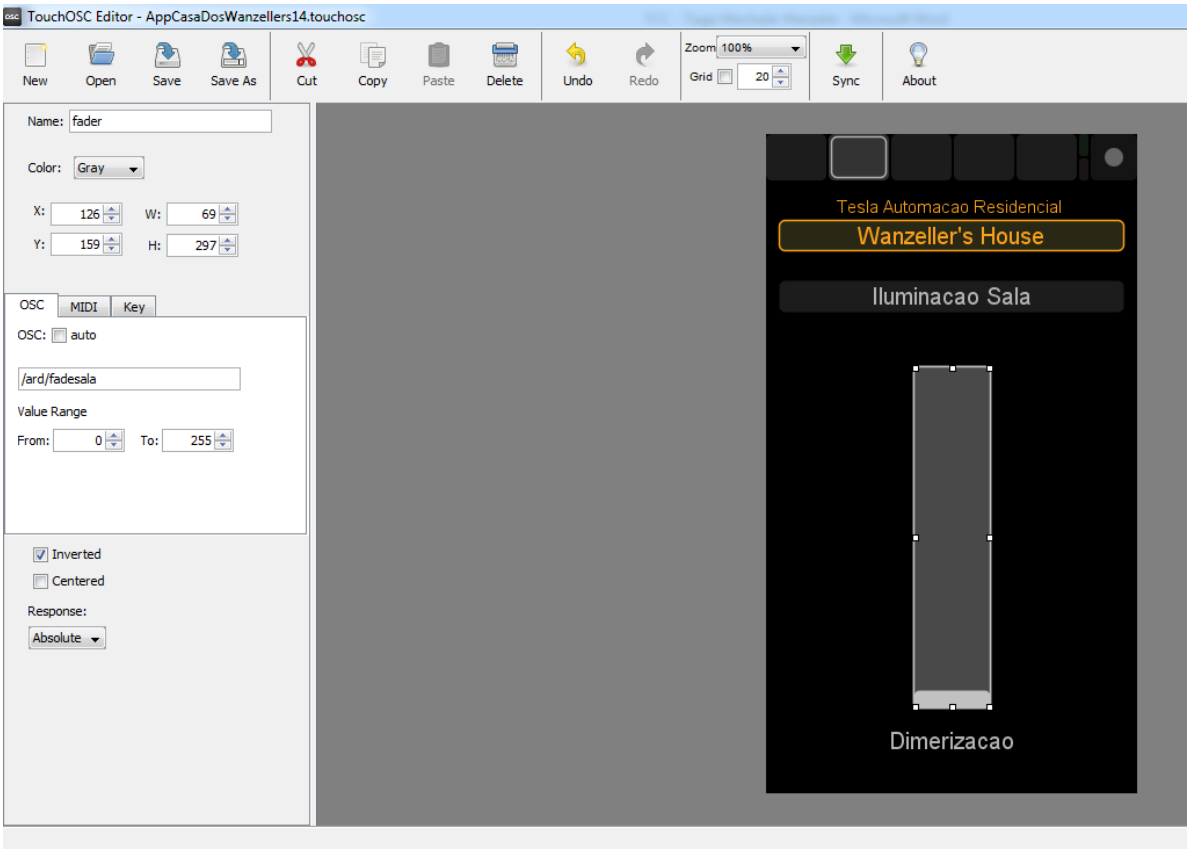


Figura 4.37 - Configuração do Fader Sala

Fonte: Próprio autor

Em seguida, adicionou-se mais uma página para criar agora o módulo 3, que contém o controle do LED RGB do quarto 01. Então novamente copiou-se o título do módulo 1 e alterou-se a cor para amarele e o título para "*Iluminação Quarto 01*". Neste módulo foram adicionado três objetos do tipo "*Fader H*" e em cada um alterou-se o "*Value Range*" para variar entre 0 e 255. Alterando também o comando que seria enviado para cada um desses três objetos, o *Fade R* representa o controle da cor vermelha do LED e seu comando foi alterado para "*/ard/fadeR*", o *Fade G* representa o controle da cor verde do LED e seu comando foi alterado para "*/ard/fadeG*" e o *Fade B* representa o controle da cor azul do LED e seu comando foi alterado para "*/ard/fadeB*". Sendo assim, o módulo 3 (*Iluminação Quarto 01*) foi finalizado, conforme mostra a figura abaixo.

Figura 4.38 - Módulo 3: Iluminação Quarto 01

Fonte: Próprio autor

Ao término da configuração da interface do módulo 3, foi adicionado uma nova página a aplicação para criar a interface do módulo que controlaria o sistema de alarme. Este módulo é representado pela cor azul e possui além do título, apenas dois elementos: Um botão (*Push Button*) para ativar ou desativar o sistema de alarme e um texto (*label*) cuja função é receber da central de controle o status do alarme. Assim, inseriu-se esses dois objetos na área de edição e alterou-se algumas de suas propriedades, em ambos elementos alterou-se a propriede "*Color*" para "*Blue*" e desmarcou-se "*auto*" em OSC e definiu-se os comandos que os referenciariam, para o botão colocou-se *"/ard/alarme*" e para o texto que monitoraria o status colocou-se "*/ard/StatusAlarme*". Basicamente, este módulo funciona da seguinte forma, ao apertar o botão, será enviado um sinal e através do comando "*/ard/larme*" o arduino identifica através da função criada para este que o botão foi pressionado, assim através do código o alarme é ativado ou desativado e em seguida em envia uma mensagem OSC de volta para aplicação com o estado atual do alarme. A figura abaixo apresenta o módulo do sistema de alarme depois de finalizado.

Figura 4.39 - Módulo Sistema de Alarme

Fonte: Próprio autor

E para finalizar a criação do aplicativo que fará o controle do sistema de automação residencial apresentado nesta monografia foi adicionado maus uma página para se criar o último módulo deste sistema, o módulo que realizará o monitoramento da temperatura. Este módulo é representado pela cor roxa e possui apenas objetos do tipo *Label*, pois a função deste módulo no sistema não é realizar nenhum tipo de acionamento é apenas receber da central de controle uma mensagem OSC com o valor da temperatura lida pelo sensor de temperatura LM35 que foi inserido na sala de estar da maquete residencial. Sendo assim, utilizou-se um *label* para receber e mostrar essa informação, cujo sua propriedade OSC foi alterada para "*/ard/temp*". Conforme mostra a figura a seguir.

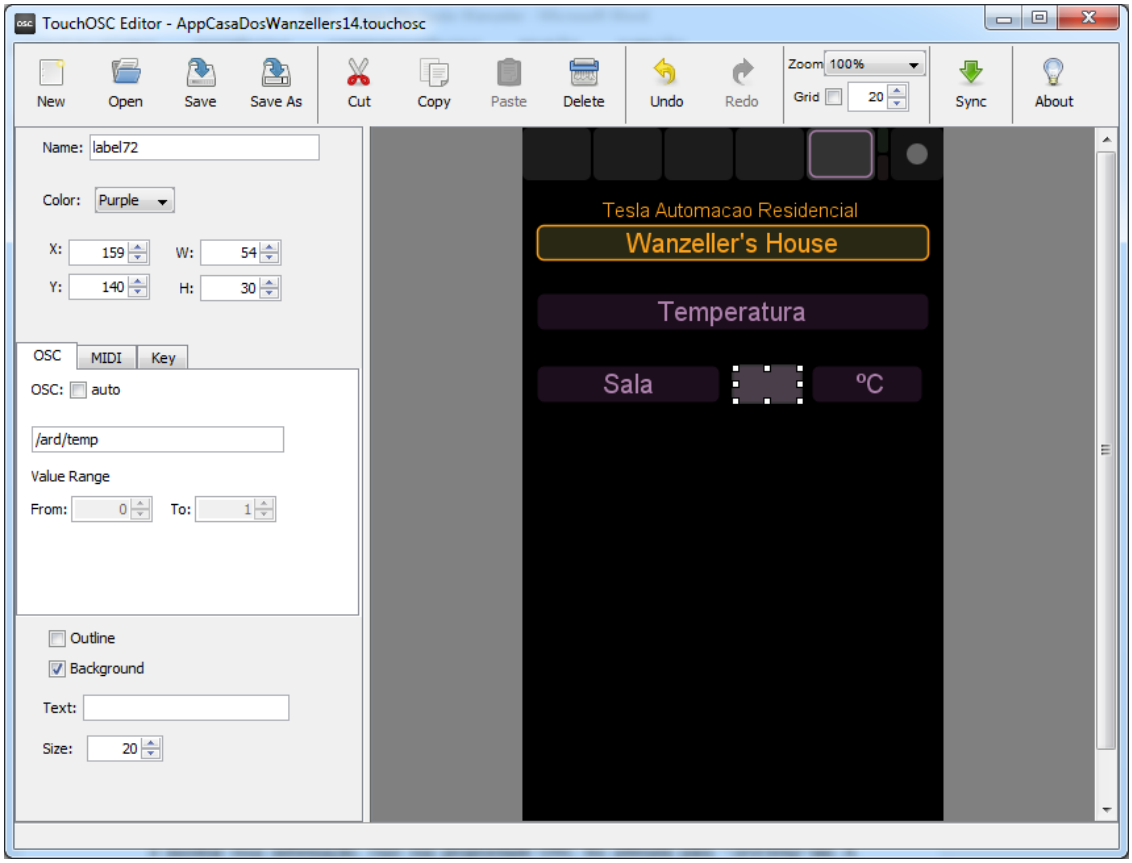


Figura 4.40 - Edição do módulo que realizará o monitoramento da temperatura

Fonte: Próprio autor

Com isso, finalizou-se a criação da aplicação que foi definida como *App Wanzeller's House*, agora basta salvar o programa criado e iniciar o processo de transferência para o celular utilizado, como citado anteriormente, neste projeto foi utilizado um *iPhone 5S*. Para transferir aplicação recém criada para o dispositivo móvel foi necessário realizar o *download* do aplicativo *TouchOSC* na *apple store*. Ao executar pela primeira vez este aplicativo foi exibida a tela abaixo, neste ponto foi necessário realizar as configurações de rede para que este se comunique com a central de controle.

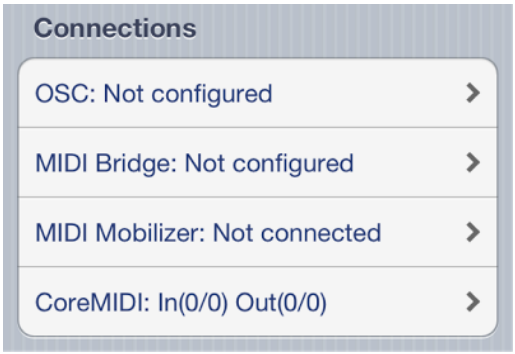


Figura 4.41 - Tela de início do *TouscOSC*

Fonte: Adaptada pelo autor

Selecionou-se "*OSC: Not configured*" e em seguida a propriedade "*Enable*" foi posicionada em "*ON*", no campo "*Host*" foi inserido o IP que foi definido para a central de controle (Arduino + *Ethernet Shield*), neste projeto foi configurado o IP 192.168.0.120, em "*Port (outgoing)*" e "*Port (incoming)*" foram inseridas as portas que foram definidas no código do arduino (8000 e 9000 respectivamente) e por fim o campo "*Local IP address*" é o IP que seu dispositivo móvel possui na rede, este atribuído automaticamente pela aplicação *TouchOSC* e deve ser configurado também no código do arduino, como será visto mais adiante. Ao retornar para a tela de início as opções "*MIDI Bridge: Not Configured*" e "*MIDI Mobilizer: Not Configured*" foram desativadas. A figura a seguir mostra o resultado da configuração do *TouchOSC.*

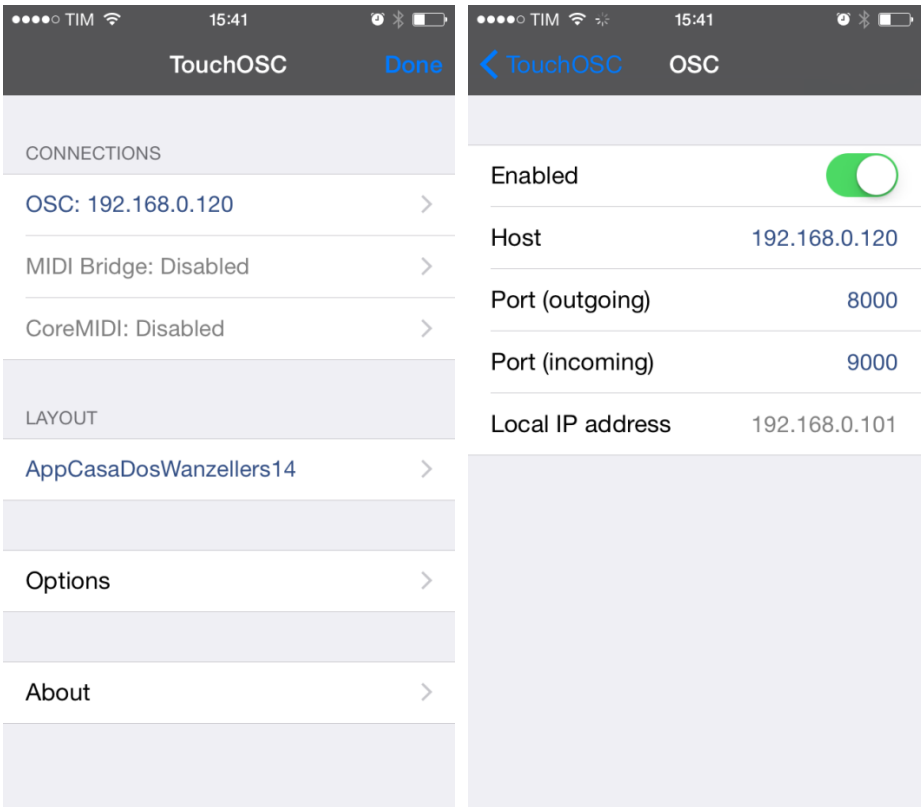


Figura 4.42 - Configuração do TouchOSC

Fonte: Próprio autor

Feito isso bastou-se sincronizar o *layout* criado para com o celular já configurado. Sendo assim, finalizou-se o desenvolvimento do protótipo de automação residencial proposto neste trabalho. O capítulo a seguir apresenta uma breve análise do código do arduino e a aplicação prática do modelo proposto, bem como o seu funcionamento.

CAPÍTULO 5

ANÁLISE DO CÓDIGO E APLICAÇÃO PRÁTICA DO MODELO PROPOSTO

Este capítulo apresenta uma breve análise do código que foi desenvolvido para esta aplicação bem como o funcionamento do modelo proposto. Vale ressaltar que todo o código que foi utilizado poder ser consultado do no apêndice A deste trabalho.

5.1. Análise Geral do Código

Assim, primeiramente para desenvolver o código deste projeto foi necessário realizar as configurações de rede que o arduino receberá e definir o IP do *host* de destino. Então as primeiras linhas do *scketch* criado são referentes as bibliotecas que foram utilizadas e a definição das configurações de rede. Como pode ser observado na figura abaixo.

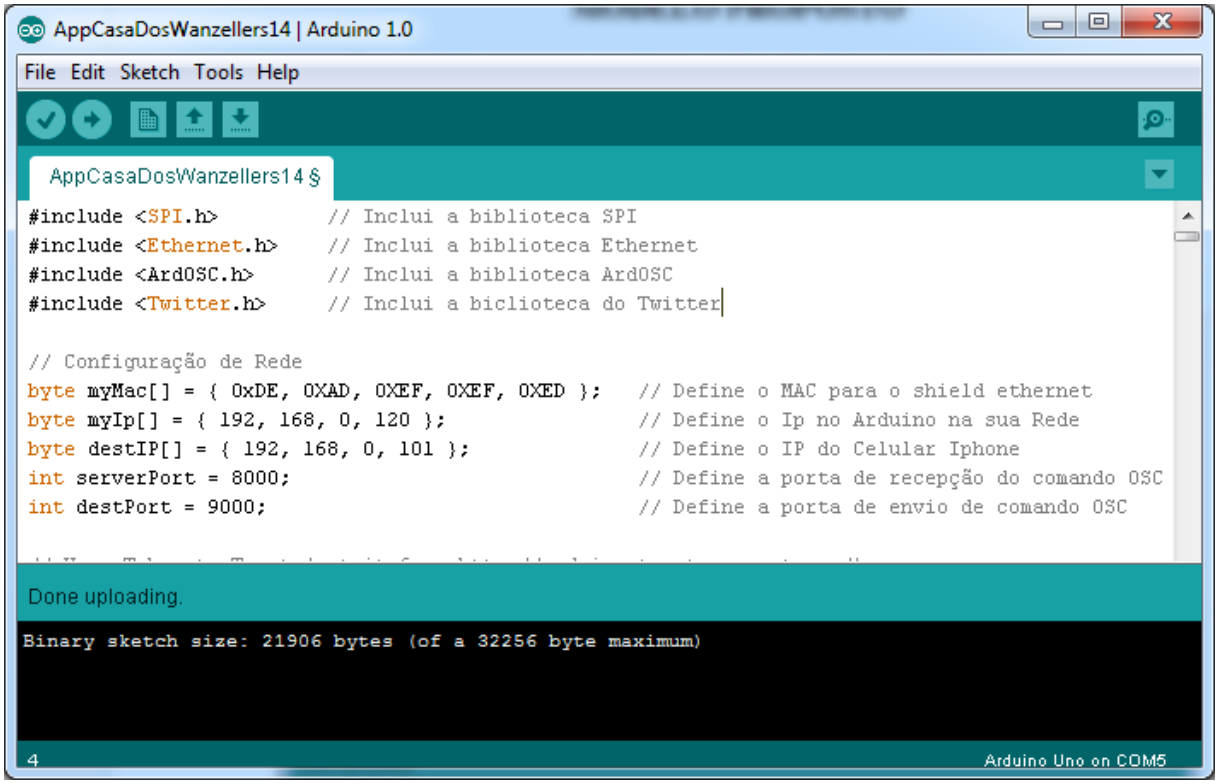


Figura 5.1 - Configurações inicias no *scketch* do arduino

Fonte: Próprio autor

Em seguida, foram definidos todos pinos que seriam utilizados e em "*void setup*" foram inicializadas as configurações de rede e definidas as funções que seriam executadas para cada comando OSC recebido, conforme visto na figura 4.36.

No *loop* principal do programa foram inseridas as linhas de código responsáveis pelo envio dos status que estavam sendo monitorado durante a execução do programa, como status das lâmpadas e sistema de alarme, além do envio dos valores de temperatura para o módulo responsável pela apresentação desta no app. Por fim, ao término do *loop* principal foram inseridas todas as funções que foram criadas para realizarem os acionamentos após serem referenciadas pelo comando definidos no *TouchOSC Editor*.

5.2. Descrição Aplicada do Modelo

Para o sistema alarme foi utilizado um sensor de movimento, ou seja, quando o usuário ativa o alarme e após isso se o sensor detectar algum movimento o alarme será acionado, e consequentemente a sirene será ativada emitindo sinal sonoro, a sirene que foi utilizada neste projeto funciona com 12V, por isso foi utilizado um relé conectado ao uma bateria 12V (ver figura 4.25), onde o relé funciona como um interruptor, ou seja, ao alarme ser acionado a central de controle envia um saída alta na bobina do relé, este fecha o contado e alimenta a sirene, emitindo assim um sinal sonoro informando que o alarme foi disparado. A figura 4.39 apresenta a tela do *App Wanzeller's* responsável por ativar e desativar o alarme. Porém, para aumentar a eficiência do sistema de alarme pensou-se em alguma forma de além de acionar uma sirene, fosse emitido alguma notificação para o usuário que o alarme de sua residência foi violado. Pois, se o mesmo não tiver em casa não saberá que o alarme de sua residência disparou.

Nesse contexto, pensou-se na utilização de um *Shield SMS* para enviar mensagens de texto ao usuário toda vez que o alarme fosse acionado, contudo este *Shield* disponível para arduino possui um preço muito alto, o que inviabilizou sua utilização. Pois, pois o objetivo principal deste trabalho é desenvolver uma automação residencial de baixo custo. Uma alternativa viável e que se mostrou muito eficiente é a utilização do envio de notificações através do *Twitter*.

Com a popularização das redes sociais e da internet, uma nova possibilidade para alertar as pessoas sobre o que está acontecendo em qualquer lugar está em evidência. Ao integrar um alarme com redes sociais, as limitações de um sistema sonoro seriam eliminadas ou minimizadas. Algumas redes sociais, como o *Twitter*, permitem enviar e receber atualizações pessoais de um usuário em tempo real e enviá-las a outros usuários que tenham optado por recebê-las. Ou seja, ao enviar uma mensagem com o *Twitter*, todos os seguidores receberão a mensagem no momento em que ela foi enviada, sem a necessidade de acessar o perfil ou página da pessoa que enviou a mensagem. No *Twitter* ainda há a possibilidade de chamar a atenção de um usuário específico, aumentando as chances da mensagem ser visualizada. Além disso, o

Twitter pode ser integrado a *emails*, *blogs*, celulares, sem a necessidade de estar logado no site para receber alguma mensagem. Um alarme integrado em uma rede social que utiliza o envio de mensagens em rede possibilita que as pessoas que estão recebendo a mensagem saibam exatamente do que se trata imediatamente. Um alarme sonoro pode ser confundido com sirenes ou simplesmente as pessoas podem demorar a perceber o propósito do alerta.

Nesse contexto, utilizou-se uma biblioteca disponível para arduino que possibilita o envio de *tweets*, sendo assim criou-se um perfil na rede social com o nome de usuário *@AlarmeTiago* e através deste foi implementado uma rotina no código do programa para que toda vez que o alarme fosse acionado enviaria um *tweet* com uma mensagem informado que o alarme foi disparado. Em seguida o perfil do usuário responsável pela residência (próprio autor) foi configurado para receber notificação em tempo real toda vez que o usuário *@AlarmeTiago* enviar um *tweet*. A figura a seguir mostra o *tweet* enviado por este perfil quando o alarme foi disparado e também apresenta o recebimento desta notificação no celular do usuário responsável pela residência.

(a) (b)

Figura 5.2 – Notificação via *Twitter*. (a) Envio do *Tweet*; (b) Recebendo a notificação no celular

Fonte: Próprio autor

Outro ponto que se mostrou extremamente favorável a utilização deste sistema de notificação, foi o fato de que mesmo que o usuário não esteja usando o telefone no momento do envio da notificação, ou seja, mesmo que o celular esteja bloqueado a notificação será exibida. Como mostra figura logo abaixo.

Figura 5.3 - Notificação com o celular bloqueado.

Fonte: Próprio autor

O monitoramento da temperatura também funcionou de acordo com o esperado, foi inserido no *loop* principal do código no arduino uma rotina para receber o valor lido pelo sensor LM35 e logo após converter para graus Celsius este valor é enviado para o módulo responsável pelo seu monitoramento. A figura a seguir mostra este módulo em funcionamento.

Figura 5.4 - Monitoramento da Temperatura

Fonte: Próprio autor

Por fim, foi analisado o funcionamento do sistema de iluminação do protótipo de automação residencial aqui apresentado, este também se mostrou muito útil e prático. A iniciar pelo controle de lâmpadas da residência, como informado no decorrer do trabalho foram

inseridas lâmpadas convencionais de 14W na cozinha, no banheiro, no quarto 02 e no quarto 03, estas foram ligadas à rede 127 Vac através de um interruptor em paralelo com um relé de 5V, controlado este último sendo controlado pelo arduino, disponibilizando assim o controle tanto pelo celular quanto pelo sistema convencional. A figura a seguir apresenta o acionamento das lâmpadas da cozinha.

Figura 5.5 - Acionamento das lâmpadas da cozinha

Fonte: Próprio autor

Como o acionamento das lâmpadas também ser feito pelos interruptores, este foi testado, e verificado seu bom funcionamento. Característica importante do sistema como mencionando anteriormente. Com isso percebeu-se também o funcionamento do sensor de luminosidade fazendo o papel de monitorar o status da lâmpada. Logo após a lâmpada ser acionada o status da lâmpada passou a indicar que a mesma estava ligada (ver figura 5.5), em seguida ao desligar a lâmpada pelo interruptor o status da lâmpada passou a indicar que a mesma estava desligada. Em seguida, o acionamento das outras lâmpadas da residência foram acionadas, para verificar o funcionamento destas. A figura a seguir mostra os quatros cômodos com suas respectivas lâmpadas acionadas pelo celular. Observa-se também que o status delas foi alterado indicando que elas permaneciam acesas.

Figura 5.6 - Acionamento de todas as lâmpadas

Fonte: Próprio autor

Em seguida foram realizados os testes com a iluminação da sala e do quarto 01. A iluminação da sala e do quarto 01 podem ser acionadas tanto pelo módulo 1 do aplicativo quanto pelo seus respectivos módulo de controle. A iluminação desses dois ambientes estão sendo simuladas com LED's, contudo conforme foi explicado nas seções anteriores poderiam ser utilizadas lâmpadas convencionais com o auxílio de um circuito extra. A figura a seguir mostra o acionamento da iluminação da sala pelo módulo 01 – *Iluminação geral.* Onde o status da lâmpada também é referenciado, agora não mais pelo sensor de luminosidade e sim pela saída da porta que está sendo utilizada para seu acionamento, bastando apenas utilizar o comando *digitalRead(lsala)* para verificar se a saída utilizada por esta iluminação está ligada (*HIGH*) ou desligada (*LOW*).

Como mencionado a iluminação da sala possui um diferencial, que é o efeito de dimerização, ou seja, o usuário pode controlar a intensidade luminosa da lâmpada através do módulo 2 que possui *fader* para controle desta luminosidade, podendo através deste efeito criar ambientes específicos para determinada situação, como reduzir a luminosidade quando foi assistir a um filme ou jogo de futebol em sua sala. Dando um toque de sofisticação em seu ambiente.

Figura 5.7 - Acionamento da iluminação da Sala

Fonte: Próprio autor

Com um efeito similar ao efeito de dimerização da iluminação da sala, a iluminação do quarto 01 simula uma iluminação RGB sendo controlada pelo módulo 3 do *App Wanzeller's House*, onde através dos 3 *fader's* disponíveis neste módulo o usuário pode controlar estas 3 variações de cores, podendo tornar seu ambiente em um clima especial. Seja para valorizar determinada local, ou para iluminar um ambiente festivo ou até mesmo para criar um ambiente romântico através dessa variação de cores, podendo resultar em diversas cores. A figura a seguir mostra alguns dos testes que foram realizados com este tipo de iluminação bem como suas combinações de cores.

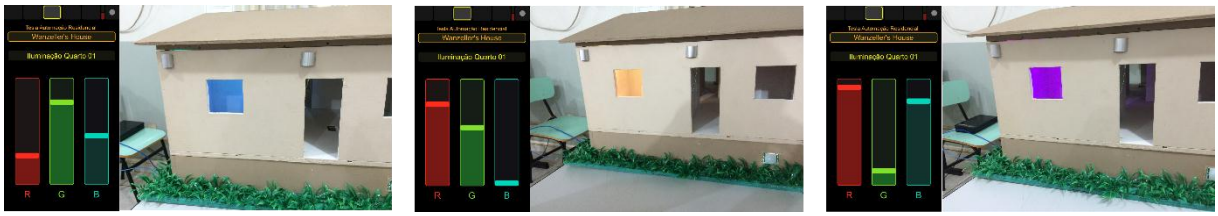


Figura 5.8 - Diversos ambientes através da iluminação RGB do Quarto 01

Fonte: Próprio autor

Com isso finalizando os testes e aplicação prática do modelo de automação residencial proposto pelo projeto.

5.3. Custos do Modelo Proposto

Um dos objetivos do projeto foi desenvolver um protótipo de automação residencial de baixo custo utilizando a plataforma livre arduino. Vale ressaltar que hoje a automação está a cada dia se tornando mais acessível, contudo esta ainda é restrita a usuários com alto poder aquisitivo. Projetos de automação que podem ser contratados por empresas já especializadas podem variar entre 10 e 50 mil reais, segundo a AURESIDE. Estes variam de simples automatização das luzes a complexos sistemas de automação residencial. A tabela abaixo apresenta os custos que foram gastos para desenvolver este protótipo de automação residencial, podendo futuramente lançar um produto bom e mais acessível no mercado, levando em consideração as melhorias que precisam ser feitas.

Tabela 6 - Custos do Modelo Proposto

Material	Quantidade	Preço Unitário	Total
Kit Arduino Básico	1 Un.	R$ 110,00	**R$ 110,00**
Maquete	1 Un.	R$ 100,00	**R$ 100,00**
Arduino	1 Un.	R$ 50,00	**R$ 50,00**
Ethernet Shield	1 Un.	R$ 70,00	**R$ 70,00**
Bateria 12 V	1 Un.	R$ 40,00	**R$ 40,00**
Módulo Relé 5 V – 2 Canais	2 Un.	R$ 12,00	**R$ 24,00**
Fonte Bivolt P/ Arduino	1 Un.	R$ 20,00	**R$ 20,00**
Sirene 12 V	1 Un.	R$ 14,00	**R$ 14,00**
Sensor de Luminosidade	4 Un.	R$ 2,00	**R$ 8,00**
Total			**R$ 436,00**

CAPÍTULO 6

CONSIDERAÇÕES FINAIS

No contexto atual do controle de residências através de sistemas embarcados, utilizando microcontroladores como central da automação, o Arduino Uno mostrou-se uma ferramenta de fácil implementação além de possuir uma boa relação custo-benefício para o controle de determinados processos residenciais, embora limitado em alguns aspectos, como capacidade de processamento de dados e números de portas disponíveis que poderiam ser utilizadas no projeto. Esta última foi uma dificuldade encontrada na execução deste projeto, pois a medida que o projeto cresce, este fica limitado ao número de portas disponíveis do Arduino. Sendo que, apenas pelo fato de se utilizar o *Ethernet Shield* acoplado ao arduino já foram inutilizadas para projeto quatro portas e além de que as portas 0 e 1 (Rx e Tx) também não poderiam ser utilizadas, pois são utilizadas internamente para envio e recepção de dados através da comunicação serial (esta foi utilizada para monitorar o que estava acontecendo durante a execução do programa), ou seja, para iniciar um projeto deste tipo com o arduino uno, das 20 portas disponíveis apenas 14 podem ser utilizadas realmente no projeto para monitoramento e acionamentos de dispositivos. Sendo um ponto desfavorável, pois se pensou no acréscimo de mais funcionalidades ao sistema, como iluminação externa e jardim, além de realizar o acionamento de cargas conectadas a rede. Uma solução para este problema seria a utilização de um Arduino Mega, pois esse possui maior número de portas e maior capacidade de processamento, entretanto aumentaria o custo do projeto.

Porém, a utilização da aplicação *TouchOSC Editor* para desenvolver o aplicativo (*App Wanzeller's House*) que veio a controlar todo nosso sistema, mostrou-se uma alternativa extremamente viável para desenvolver aplicações para dispositivos móveis, pois pode-se desenvolver tanto aplicações para o sistema operacional iOS quanto para Android, este último em maior utilização por parte dos usuários. E ainda, mesmo sem ter experiência alguma em programação. E aliada a grande utilidade desta ferramenta, a mesma é fácil de se manipular e pode criar interfaces intuitivas e personalizadas para dispositivos móveis, área que está cada vez mais explorada com a popularização de *smarthphones* e *tablets*. Pois, o aplicativo criado se mostrou bem simples e funcional, um ponto favorável à utilização deste sistema.

Contudo, apesar da aplicação ser bastante intuitiva e funcional, durante os testes da aplicação prática do modelo proposto, em determinadas situações aplicação parava de responder a alguns comandos, outros testes foram realizados e constatou-se que estes

travamentos poderiam estar sendo ocasionados pelo grande fluxo de mensagens que estavam sendo enviadas simultaneamente, tanto do aplicativo para o celular, quanto do celular para o aplicativo. Ou ainda por falhas de comunicação com com a rede *wi-fi* utilizada durante os testes. Este erro de comunicação com o *Wi-Fi* ou o grande fluxo de informações poderia estar travando o sistema, sendo necessária a reinicialização do sistema para que este voltasse a funcionar. Entretanto, estas situações não eram constantes. Sendo assim, exceto por esses imprevistos a aplicação mostrou-se em bom funcionamento.

Outro ponto que se mostrou favorável foi o fato de que o sistema possui uma relação custo-benefício bem atraente, pois todos os benefício que a automação residencial pode trazer são vários, como mostrados nas seções anteriores. Ademais, a área de automação residencial está em crescente evolução, e a tendência é a utilização de sistemas mais robustos e com maior capacidade de processamento de dados, integrando o maior número possível de aplicações e fazendo com que o binômio custo-benefício tenha cada vez mais relevância.

Por fim, neste trabalho foi apresentada a elaboração de um protótipo de automação residencial utilizando a plataforma de prototipagem eletrônica arduino. Buscando aliar um baixo custo de investimento ao projeto que foi proposto. Então, a partir dos fatos relatados e resultados encontrados no decorrer do desenvolvimento e testes do modelo proposto, pode-se concluir que o objetivo do trabalho foi alcançado através dos métodos que foram desenvolvidos.

6.1 Trabalhos Futuros

Como sugestões para trabalhos futuros, consideram-se:

- ✓ Substituir a utilização do Arduino Uno por um Arduino Mega, visto que este possui maior número de portas e maior capacidade de processamento de dados. Podendo através dessa substituição resolver o problema de disponibilidades de portas e consequentemente aumentar as funcionalidades do sistema.
- ✓ Desenvolver através de outros métodos uma aplicação para dispositivos móveis que possa ser acessado via internet, ou seja, o usuário poderá controlar todo o sistema desenvolvido de qualquer lugar do mundo, bastando para isso que o usuário tenha acesso à internet. Pois, apesar de ser uma excelente ferramenta, a aplicação *TouchOSC Editor* não possibilita o acesso externo ao sistema.
- ✓ Fazer a utilização de módulos que possam se comunicar via *Wireless* com a central de comando, possibilitando assim redução de custo de aplicação em um projeto real. Pois, reduziria consideravelmente a utilização de cabos que ligam a central a outros cômodos da casa.
- ✓ Tornar o protótipo um sistema comercializável.

REFERÊNCIAS BIBIOGRÁFICAS

[1] **Por dentro da casa inteligente**. Publicado na Revista TI. Disponível em: < http://www.aureside.org.br/artigos/default.asp?file=01.asp&id=71>. Acesso em: 10/03/2015.

[2] OSORIO, A. S.; **Automação Residencial**. AURESIDE. Artigo disponível em: <http://www.aureside.org.br/temastec/automacao_residencial_final.pdf>. Acesso em: 25/02/2015.

[3] OLIVEIRA, A. M. **Automação Residencial**. 2005. 53 f.. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Departamento de Ciências da Administração e Tecnologia, Centro Universitário de Araraquara, 2005.

[4] BEGHINI, L. B. **Automação residencial de baixo custo por meio de dispositivos móveis com sistema operacional Android**. 2013. 50 f.. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) - Escola de Engenharia de São Carlos, da Universidade de São Paulo, 2013.

[5] AZEVEDO, C. L.; PIZZOLATO, N. D.; **Aplicabilidades e Sistemas de Automação Residencial**. Disponível em: < http://essentiaeditora.iff.edu.br/index.php/vertices/article/viewFile/1809-2667.20040015/86>. Acesso em: 16/03/2015.

[6] OLIVEIRA, A. M. **Automação Residencial**. 2005. 53 f.. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Departamento de Ciências da Administração e Tecnologia, Centro Universitário de Araraquara, 2005.

[7] OLIVEIRA, A. M. **Automação Residencial**. 2005. 53 f.. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Departamento de Ciências da Administração e Tecnologia, Centro Universitário de Araraquara, 2005.

[8] MURATORI, J. R.; **Automação Residencial: Histórico, Definições e Conceitos**. Disponível em: < http://www.osetoreletrico.com.br/web/documentos/fasciculos/Ed62_fasc_automacao_capI.pdf>. Acesso em: 16/03/2015.

[9] AZEVEDO, C. L.; PIZZOLATO, N. D.; **Aplicabilidades e Sistemas de Automação Residencial**. Disponível em: <

http://essentiaeditora.iff.edu.br/index.php/vertices/article/viewFile/1809-2667.20040015/86>. Acesso em: 16/03/2015.

[10] Disponível em: < http://www.aureside.org.br/noticias_recentes/default.asp?file=01.asp&id=366>. Acesso em: 19/03/2015.

[11] Disponível em: < http://www.ihouse.com.br/ihouse.php >. Acessado em 19/03/2015.

[12] Disponível em: < http://www.ihouse.com.br/caracteristicas-do-wallpad.php >. Acessado em: 19/03/2015.

[13] Disponível em: < http://www.gdsautomacao.com.br/site/index.php?option=com_content&view=article&id=57&Itemid=37 >. Acessado em: 19/03/2015.

[14] Disponível em: < http://www.gdsautomacao.com.br/site/index.php?option=com_content&view=article&id=47&Itemid=27#sol1 >. Acessado em: 19/03/2015.

[15] Disponível em: < http://www.qualihouse.com.br/simplifies.php >. Acessado em: 19/03/2015.

[16] Disponível em: < http://www.aureside.org.br/noticias_recentes/default.asp?file=01.asp&id=373 >. Acessado em: 19/03/2015.

[17] Disponível em: < http://arduino.cc/en/main/arduinoBoardUno >. Acessado em: 19/03/2015.

[18] Disponível em: < http://softwarelivre.blog.br/2014/06/21/comunicacao-spi-em-linux/ >. Acessado em: 19/03/2015.

[19] Disponível em: < http://webdascoisas.com/blog/robotica/arduino-ethernet-shield-w5100 >. Acessado em: 19/03/2015.

[20] Disponível em: < http://www.embarcados.com.br/arduino-uno/ >. Acessado em: 20/03/2015

[21] Disponível em: < http://arduino.cc/en/Main/ArduinoEthernetShield >. Acessado em: 22/03/2015.

[22] Disponível em: < http://opensoundcontrol.org/introduction-osc >. Acessado em: 11/04/2015.

[23] Disponível em: < http://pt.wikipedia.org/wiki/LDR >. Acessado em: 16/04/2015.

[24] Disponível em: < http://artefects.com/blog/?p=12 >. Acessado em: 28/04/15.

[25] Disponível em: < http://playground.arduino.cc/Code/TwitterLibrary > Acessado em: 05/05/2015.

[26] Disponível em: < http://blog.eletronlivre.com.br/2012/12/automacao-residencial-com-arduino-na.html > Acessado em: 08/04/2015.

APÊNDICE

A. Código do Programa Desenvolvido na IDE do Arduino

```
#include <SPI.h>         // Inclui a biblioteca SPI
#include <Ethernet.h>    // Inclui a biblioteca Ethernet
#include <ArdOSC.h>      // Inclui a biblioteca ArdOSC
#include <Twitter.h>     // Inclui a biclioteca do Twitter

// Configuração de Rede
byte myMac[] = { 0xDE, 0XAD, 0XEF, 0XEF, 0XED };   // Define o MAC para o shield ethernet
byte myIp[] = { 192, 168, 0, 120 };              // Define o Ip no Arduino na sua Rede
byte destIP[] = { 192, 168, 0, 101 };            // Define o IP do Celular Iphone
int serverPort = 8000;                          // Define a porta de recepção do comando OSC
int destPort = 9000;                           // Define a porta de envio de comando OSC

// Your Token to Tweet (get it from http://arduino-tweet.appspot.com/)
Twitter twitter("3222284829-AezUcUHZLpGR9xL2MhuIBxyS5uoTAU2RNVUGaxG");
int disparo = 1;

// Declaração dos Pinos de Iluminação
const int lbanheiro = 2;        // Define o pino 2 associado a iluminação do banheiro
const int lq1_fadeR = 3;        // Define o pino 3 (PWM) associado ao led R (RGB)
const int lcozinha = 4;         // Define o pino 4 associado a iluminação da cozinha
const int lq1_fadeG = 5;        // Define o pino 5 (PWM) associado ao led G (RGB)
const int lq1_fadeB = 6;        // Define o pino 6 (PWM) associado ao led B (RGB)
const int lquarto2 = 7;         // Define o pino 7 associado a iluminação do quarto 2
const int lquarto3 = 8;         // Define o pino 8 associado a iluminação do quarto 3
const int lsala = 9;            // Define o pino 9 associado a iluminação da sala

// Deckaração dos ldr's para monitorar status das lâmpadas
const int sensorldrQ2 = A0;     // Define o pino A0 associado ao LDR no quarto 2
const int sensorldrQ3 = A1;     // Define o pino A1 associado ao LDR no quarto 3
const int sensorldrCoz = A2;    // Define o pino A2 associado ao LDR na cozinha

// Declaração do Sensor de Temperatura
const int sensorlm35 = A3;      // Define o pino A3 associado ao LM35 na sala

// Declaração das variáveis do sistema de alarme
const int sensorpir = A4;       // Define o pino A4 associado ao Sensor PIR
const int buzina = A5;          // Define o pino A5 associado à Buzina
boolean alarme = false;         // Variável para monitorar o status do alarmer
boolean envia = false;          // Variável para controlar o envio de tweets

OSCServer server;    // Inicializa o servidor OSC (Serviço que trata a recepação da mensagem O SC)
OSCClient client;    // Inicializa o cliente OSC (Serviço que trata o envio da mensagem OSC)

void setup() {

  delay(1000);
  Ethernet.begin(myMac, myIp);                  // Inicializa a interface Ethernet
  Serial.begin(9600);                          // Inicializa a porta serial
  server.begin(serverPort);                     // Inicializa o Servidor OSC definindo a porta de RX
  server.addCallback("/ard/sala",&funcSala);        // Define a rotina "funcSala" sera executado paro o
comando "/ard/sala"
  server.addCallback("/ard/quarto1",&funcQuarto1);  // Define a rotina "funcQuarto1" sera executado
paro o comando "/ard/quarto1"
  server.addCallback("/ard/quarto2",&funcQuarto2);  // Define a rotina "funcQuarto2" sera executado
paro o comando "/ard/quarto2"
```

```
  server.addCallback("/ard/quarto3",&funcQuarto3);  // Define a rotina "funcQuarto3" sera executado
paro o comando "/ard/quarto3"
  server.addCallback("/ard/banheiro",&funcBanheiro);// Define a rotina "funcExterna" sera executado
paro o comando "/ard/banheiro"
  server.addCallback("/ard/cozinha",&funcCozinha);  // Define a rotina "funcCozinha" sera executado
paro o comando "/ard/cozinha"
  server.addCallback("/ard/fadesala",&funcFadeSala);// Define a rotina "funcFadeSala" sera executado
paro o comando "/ard/fadesala"
  server.addCallback("/ard/fadeR",&funcFadeR);      // Define a rotina "funcFadeR" sera executado paro
o comando "/ard/fadeR"
  server.addCallback("/ard/fadeG",&funcFadeG);      // Define a rotina "funcFadeG" sera executado paro
o comando "/ard/fadeG"
  server.addCallback("/ard/fadeB",&funcFadeB);      // Define a rotina "funcFadeB" sera executado paro
o comando "/ard/fadeB"
  server.addCallback("/ard/alarme",&funcAlarme);    // Define a rotina "funcAlarme" sera executado
paro o comando "/ard/alarme"
  server.addCallback("/ard/desliga",&funcDesliga);  // Define a rotina "funcDesliga" sera executado
paro o comando "/ard/desliga"

  pinMode(lcozinha, OUTPUT);    // Define lcozinha como saida
  pinMode(lq1_fadeR, OUTPUT);   // Define lq1_fadeR como saida
  pinMode(lq1_fadeG, OUTPUT);   // Define lq1_fadeG como saida
  pinMode(lq1_fadeB, OUTPUT);   // Define lq1_fadeB como saida
  pinMode(lquarto2, OUTPUT);    // Define lquarto2 como saida
  pinMode(lquarto3, OUTPUT);    // Define lquarto3 como saida
  pinMode(lbanheiro, OUTPUT);   // Define lbanheiro como saida
  pinMode(buzina, OUTPUT);      // Define buzina com saída
  pinMode(lsala, OUTPUT);       // Define lsala como saida
  pinMode(sensorpir, INPUT);    // Define o sensor de presença com entrada

  // Define os pinos nos relés como alto
  digitalWrite(lbanheiro, HIGH);
  digitalWrite(lcozinha, HIGH);
  digitalWrite(lquarto2, HIGH);
  digitalWrite(lquarto3, HIGH);
}

void loop() {
  if(server.aviableCheck()>0){                  // Verifica se há alguma mensagem OSC
  }

 // Definição do Sistema de Alarme
  if(alarme == true) {                          // Verifica se alarme está ativado

    // Status Sistema de Alarme
      OSCMessage txMesSA;                       // Cria uma nova mensagem
      txMesSA.setAddress(destIP,destPort);      // Define endereço e porta da mensagem OSC
      txMesSA.beginMessage("/ard/StatusAlarme"); // Define comando OSC
      txMesSA.addArgString("Alarme Ativado");   // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
      client.send(&txMesSA);                    // Envia mensagem OSC para atualizar o estado da lampada no
app

    int ValorSensorPir = digitalRead(sensorpir);// Faz a leitura do sensor de presença
    if(ValorSensorPir == HIGH){                 // Verifica se o sensor detectou algum movimento
      digitalWrite(buzina, HIGH);               // Ativa sinal sonoro com saída baixa
      if(envia == true) {
        Serial.println("Alarme Disparou.");     // Imprime que o Alarme foi disparado
        Serial.println("Tantativa de Envio:");  // Imprime que o Alarme foi disparado
        post();
```

```
      envia = false;
    }                              // Finaliza (envia == true)
   }                               // Finaliza (ValorSensorPir == HIGH)
  } else {                           // se alarme estiver desativado

    OSCMessage txMesSA;                    // Cria uma nova mensagem
    txMesSA.setAddress(destIP,destPort);     // Define endereço e porta da mensagem OSC
    txMesSA.beginMessage("/ard/StatusAlarme");// Define comando OSC
    txMesSA.addArgString("Alarme Desativado");// Define valor do estado que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSA);                // Envia mensagem OSC para atualizar o estado da lampada no
app

    digitalWrite(buzina, LOW);             // Desativa o sinal sonoro com saída alta
  }

  // Definição do Sensor de Temperatura do Sistema
  int valorlm35 = analogRead(sensorlm35);      // Faz a leitura do sensor lm35
  int temp = 150*valorlm35/306.9;             // Converte o valor analógico para º C

  OSCMessage txMesT;                     // Cria uma nova mensagem
  txMesT.setAddress(destIP,destPort);      // Define endereço e porta da mensagem OSC
  txMesT.beginMessage("/ard/temp");        // Define comando OSC
  txMesT.addArgFloat(temp);               // Define valor da temperatura que será enviado no argumento
OSC para o Iphone
  client.send(&txMesT);                  // Envia mensagem OSC para atualizar o estado da temperatura no
app

  // Definição do Status das lâmpadas

  // Status Iluminação Quarto 2
  int valorldrQ2 = analogRead(sensorldrQ2);       // Faz a leitura do sensor de luminosidade
  if(valorldrQ2 > 900){
    OSCMessage txMesSQ2;                    // Cria uma nova mensagem
    txMesSQ2.setAddress(destIP,destPort);          // Define endereço e porta da mensagem OSC
    txMesSQ2.beginMessage("/ard/StatusQuarto2");    // Define comando OSC
    txMesSQ2.addArgFloat(HIGH);                  // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSQ2);                     // Envia mensagem OSC para atualizar o estado da lampada
no app
  } else {
    OSCMessage txMesSQ2;                    // Cria uma nova mensagem
    txMesSQ2.setAddress(destIP,destPort);          // Define endereço e porta da mensagem OSC
    txMesSQ2.beginMessage("/ard/StatusQuarto2");    // Define comando OSC
    txMesSQ2.addArgFloat(LOW);                   // Define valor do estado que será enviado no argumento
OSC para o Iphone
    client.send(&txMesSQ2);                     // Envia mensagem OSC para atualizar o estado da lampada
no app
  }

  // Status Iluminação Quarto 3
  int valorldrQ3 = analogRead(sensorldrQ3);       // Faz a leitura do sensor de luminosidade
  if(valorldrQ3 > 900){
    OSCMessage txMesSQ3;                    // Cria uma nova mensagem
    txMesSQ3.setAddress(destIP,destPort);          // Define endereço e porta da mensagem OSC
    txMesSQ3.beginMessage("/ard/StatusQuarto3");    // Define comando OSC
    txMesSQ3.addArgFloat(HIGH);                  // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSQ3);                     // Envia mensagem OSC para atualizar o estado da lampada
no app
```

```
  } else {
    OSCMessage txMesSQ3;                        // Cria uma nova mensagem
    txMesSQ3.setAddress(destIP,destPort);          // Define endereço e porta da mensagem OSC
    txMesSQ3.beginMessage("/ard/StatusQuarto3");    // Define comando OSC
    txMesSQ3.addArgFloat(LOW);                    // Define valor do estado que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSQ3);                      // Envia mensagem OSC para atualizar o estado da lampada
no app
  }

  // Status Iluminação Cozinha
  int valorldrCoz = analogRead(sensorldrCoz);       // Faz a leitura do sensor de luminosidade
  if(valorldrCoz > 900){
    OSCMessage txMesSC;                         // Cria uma nova mensagem
    txMesSC.setAddress(destIP,destPort);           // Define endereço e porta da mensagem OSC
    txMesSC.beginMessage("/ard/StatusCozinha");     // Define comando OSC
    txMesSC.addArgFloat(HIGH);                    // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSC);                       // Envia mensagem OSC para atualizar o estado da lampada
no app
  } else {
    OSCMessage txMesSC;                         // Cria uma nova mensagem
    txMesSC.setAddress(destIP,destPort);           // Define endereço e porta da mensagem OSC
    txMesSC.beginMessage("/ard/StatusCozinha");     // Define comando OSC
    txMesSC.addArgFloat(LOW);                     // Define valor do estado que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSC);                       // Envia mensagem OSC para atualizar o estado da lampada
no app
  }

  // Status Iluminação Banheiro
  int StatusBanheiro = digitalRead(lbanheiro);      // Faz a leitura do pino
  if(StatusBanheiro){
    OSCMessage txMesSBan;                       // Cria uma nova mensagem
    txMesSBan.setAddress(destIP,destPort);          // Define endereço e porta da mensagem OSC
    txMesSBan.beginMessage("/ard/StatusBanheiro");    // Define comando OSC
    txMesSBan.addArgFloat(LOW);                   // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSBan);                     // Envia mensagem OSC para atualizar o estado da
lampada no app
  } else {
    OSCMessage txMesSBan;                       // Cria uma nova mensagem
    txMesSBan.setAddress(destIP,destPort);          // Define endereço e porta da mensagem OSC
    txMesSBan.beginMessage("/ard/StatusBanheiro");    // Define comando OSC
    txMesSBan.addArgFloat(HIGH);                  // Define valor do estado que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSBan);                     // Envia mensagem OSC para atualizar o estado da
lampada no app
  }

}

// Função que envia um tweet informando que o alarme foi disparado
void post(){

  char msg[40];
  sprintf(msg, "Alerta de intruso. Yago Invadiu! N00%d", disparo);
  disparo ++;

  Serial.println(msg);
```

```
  Serial.println("connecting ...");

  if (twitter.post(msg)) {
    int status = twitter.wait();
    if (status == 200) {
      Serial.println("OK.");
  } else {
      Serial.print("failed : code ");
      Serial.println(status);
    }
  } else {
    Serial.println("connection failed.");
  }
//  delay(1000);
}

void funcAlarme(OSCMessage *_mes){            // Rotina que trata o comando OSC "/ard/alarme"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);       // Armazena em value o argumento do comando OSC
"/ard/alarme"
  if(value==HIGH){                          // Verifica se value está em estado alto (botão ativado)
  // Serial.println("Botao Alarme! ");     // Imprime que o botão foi pressionado

    if(alarme == false){                    // Verifica se value está em estado alto (botão ativado)
      alarme = true;                        // Variável alarme recebe HIGH (Alarme Ativado)
      envia = true;                         // Variável envia recebe true para o evio do tweet
      Serial.println("Alarme Ativado!");       // Imprime Alarme Ativado
    } else {                                // Se value não for alto
      alarme = false;                       // Variável alarme recebe LOW (Alarme Desativado)
      Serial.println("Alarme Desativado!");    // Imprime Alarme Desativado
      }
  } else {                                  // Verifica quando o botão foi solto
//    Serial.println("Botao Alarme Solto! ");          // Imprime que o botão foi solto
  }
}


void funcSala(OSCMessage *_mes){              // Rotina que trata o comando OSC "/ard/sala recebido
do Iphone
  int value = (int)_mes->getArgFloat(0);        // Armazena em value o argumento do comando OSC
"/ard/sala"
  if(value == HIGH) {                       // Verifica se o botão está pressionado
    Serial.println("Sala Pressionado! ");       // Informa que o botão foi pressionado
    int state = digitalRead(lsala);             // Faz a leitura do estado atual
    if (state > 0){                         // Se led fade sala maior que 0
      digitalWrite(lsala, LOW);                 // Desliga a iluminação da sala
      Serial.print("Sala: ");                 // Imprime que a da sala foi desligada
      Serial.println(LOW);                    // Imprime que a da sala foi desligada

      OSCMessage txMesSS;                     // Cria uma nova mensagem
      txMesSS.setAddress(destIP,destPort);       // Define endereço e porta da mensagem OSC
      txMesSS.beginMessage("/ard/StatusSala");   // Define comando OSC
      txMesSS.addArgFloat(LOW);                 // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
      client.send(&txMesSS);                  // Envia mensagem OSC para atualizar o estado da sala

      OSCMessage txMesFS;                     // Cria uma nova mensagem
      txMesFS.setAddress(destIP,destPort);       // Define endereço e porta da mensagem OSC
      txMesFS.beginMessage("/ard/fadesala");     // Define comando OSC
```

```
    txMesFS.addArgFloat(0);                 // Define novo estado 0 do fade sala que será enviado no
argumento OSC para o Iphone
    client.send(&txMesFS);                  // Envia mensagem OSC para atualizar o estado do fade sala

   } else {
    digitalWrite(lsala, HIGH);              // Liga a iluminação da sala
    Serial.print("Sala: ");                 // Imprime que a da sala foi ligada
    Serial.println(HIGH);                   // Imprime que a da sala foi ligada

    OSCMessage txMesSS;                     // Cria uma nova mensagem
    txMesSS.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesSS.beginMessage("/ard/StatusSala");  // Define comando OSC
    txMesSS.addArgFloat(HIGH);              // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSS);                  // Envia mensagem OSC para atualizar o estado da sala

    OSCMessage txMesFS;                     // Cria uma nova mensagem
    txMesFS.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesFS.beginMessage("/ard/fadesala");  // Define comando OSC
    txMesFS.addArgFloat(255);               // Define novo estado 255 do fade sala que será enviado no
argumento OSC para o Iphone
    client.send(&txMesFS);                  // Envia mensagem OSC para atualizar o estado do fade sala

    }                                       // Finaliza o else (state > 0)

  } else {
   Serial.println("Botao Sala Solto! ");    // Informa que o botão foi solto

  }                                         // Finaliza o else (value == HIGH)
}                                           // Finaliza funcSala

void funcQuarto1(OSCMessage *_mes){         // Rotina que trata o comando OSC "/ard/quarto1"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);    // Armazena em value o argumento do comando OSC
"/ard/quarto1"
  if(value == HIGH) {                       // Verifica se o botão está presionado
   Serial.println("Quarto1 Pressionado!");  // Imprime que o botão quarto 1 foi pressionado

   int stateR = digitalRead(lq1_fadeR);     // Faz a leitura do estado atual de lq1_fadeR
   int stateG = digitalRead(lq1_fadeG);     // Faz a leitura do estado atual de lq1_fadeG
   int stateB = digitalRead(lq1_fadeB);     // Faz a leitura do estado atual de lq1_fadeB

   if ((stateR > 0) || (stateG > 0) || (stateB > 0)) {
    digitalWrite(lq1_fadeR, LOW);           // Desliga o lq1_fadeR
    digitalWrite(lq1_fadeG, LOW);           // Desliga o lq1_fadeG
    digitalWrite(lq1_fadeB, LOW);           // Desliga o lq1_fadeB
    Serial.print("Quarto 1: ");
    Serial.println(LOW);                    // Imprimi que a lampada do quarto 1 foi desligada

    OSCMessage txMesSQ1;                    // Cria uma nova mensagem
    txMesSQ1.setAddress(destIP,destPort);   // Define endereço e porta da mensagem OSC
    txMesSQ1.beginMessage("/ard/StatusQuarto1");// Define comando OSC
    txMesSQ1.addArgFloat(LOW);              // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSQ1);                 // Envia mensagem OSC para atualizar o estado da lampada

    OSCMessage txMesFR;                     // Cria uma nova mensagem para ser enviada para fadeR
    txMesFR.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesFR.beginMessage("/ard/fadeR");     // Define comando OSC
```

```
    txMesFR.addArgFloat(0);              // Define novo estado do fadeR que será enviado no argumento
OSC para o Iphone
    client.send(&txMesFR);               // Envia mensagem OSC para atualizar o estado do fadeR

    OSCMessage txMesFG;                  // Cria uma nova mensagem para ser enviada para fadeG
    txMesFG.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesFG.beginMessage("/ard/fadeG");     // Define comando OSC
    txMesFG.addArgFloat(0);              // Define novo estado do fadeG que será enviado no argumento
OSC para o Iphone
    client.send(&txMesFG);               // Envia mensagem OSC para atualizar o estado do fadeG

    OSCMessage txMesFB;                  // Cria uma nova mensagem para ser enviada para fadeB
    txMesFB.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesFB.beginMessage("/ard/fadeB");     // Define comando OSC
    txMesFB.addArgFloat(0);              // Define novo estado do fadeB que será enviado no argumento
OSC para o Iphone
    client.send(&txMesFB);               // Envia mensagem OSC para atualizar o estado do fadeB
   } else {                              // Se nenhuma dos leds (RGB) > 0
    digitalWrite(lq1_fadeR, HIGH);       // Liga o lq1_fadeR
    digitalWrite(lq1_fadeG, HIGH);       // Liga o lq1_fadeG
    digitalWrite(lq1_fadeB, HIGH);       // Liga o lq1_fadeB
    Serial.print("Quarto 1: ");
    Serial.println(HIGH);                // Imprimi que a lampada do quarto 1 foi ligada

    OSCMessage txMesSQ1;                 // Cria uma nova mensagem
    txMesSQ1.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesSQ1.beginMessage("/ard/StatusQuarto1");// Define comando OSC
    txMesSQ1.addArgFloat(HIGH);          // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSQ1);              // Envia mensagem OSC para atualizar o estado da lampada

    OSCMessage txMesFR;                  // Cria uma nova mensagem para ser enviada para fadeR
    txMesFR.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesFR.beginMessage("/ard/fadeR");     // Define comando OSC
    txMesFR.addArgFloat(255);            // Define novo estado do fadeR que será enviado no
argumento OSC para o Iphone
    client.send(&txMesFR);               // Envia mensagem OSC para atualizar o estado do fadeR

    OSCMessage txMesFG;                  // Cria uma nova mensagem para ser enviada para fadeG
    txMesFG.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesFG.beginMessage("/ard/fadeG");     // Define comando OSC
    txMesFG.addArgFloat(255);            // Define novo estado do fadeG que será enviado no
argumento OSC para o Iphone
    client.send(&txMesFG);               // Envia mensagem OSC para atualizar o estado do fadeG

    OSCMessage txMesFB;                  // Cria uma nova mensagem para ser enviada para fadeB
    txMesFB.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesFB.beginMessage("/ard/fadeB");     // Define comando OSC
    txMesFB.addArgFloat(255);            // Define novo estado do fadeB que será enviado no
argumento OSC para o Iphone
    client.send(&txMesFB);               // Envia mensagem OSC para atualizar o estado do fadeB

   }                                     // Finaliza senao ((stateR > 0) || (stateG > 0) || (stateB > 0))

  } else {
   Serial.println("Quarto 1 Solto! ");   // Informa que o botão foi solto
  }
}
```

```
void funcQuarto2(OSCMessage *_mes){                // Rotina que trata o comando OSC "/ard/quarto2"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);           // Armazena em value o argumento do comando OSC
"/ard/quarto2"
  if(value == HIGH) {                              // Verifica se potão foi pressionado
    Serial.println("Botao Quarto 2 Pressionado! ");  // Imprime que o botão foi pressionado

    int state = digitalRead(lquarto2);             // Faz a leitura do estado atual
    digitalWrite(lquarto2, !state);                // Muda o estado atual do relé (liga ou desliga)
    Serial.print("Quarto 2: ");
    Serial.println(!state);
  } else {                                         // Verifica quando o botão foi solto
    Serial.println("Botao Quarto 2 Solto! ");      // Imprime que o botão foi solto
  }
}

void funcQuarto3(OSCMessage *_mes){                // Rotina que trata o comando OSC "/ard/quarto3"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);           // Armazena em value o argumento do comando OSC
"/ard/quarto3"
  if(value == HIGH) {                              // Verifica se potão foi pressionado
    Serial.println("Botao Quarto 3 Pressionado! ");  // Imprime que o botão foi pressionado

    int state = digitalRead(lquarto3);             // Faz a leitura do estado atual
    digitalWrite(lquarto3, !state);                // Muda o estado atual do relé (liga ou desliga)
    Serial.print("Quarto 3: ");
    Serial.println(!state);

  } else {                                         // Verifica quando o botão foi solto
    Serial.println("Botao Quarto 3 Solto! ");      // Imprime que o botão foi solto
  }
}

void funcBanheiro(OSCMessage *_mes){               // Rotina que trata o comando OSC "/ard/banheiro"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);           // Armazena em value o argumento do comando OSC
"/ard/banheiro"
  if(value == HIGH) {                              // Verifica se potão foi pressionado
    Serial.println("Botao Banheiro Pressionado! ");  // Imprime que o botão foi pressionado

    int state = digitalRead(lbanheiro);            // Faz a leitura do estado atual
    digitalWrite(lbanheiro, !state);               // Muda o estado da lâmpada
    Serial.print("Banheiro: ");
    Serial.println(!state);

  } else {                                         // Verifica quando o botão foi solto
    Serial.println("Botao Banheiro Solto! ");      // Imprime que o botão foi solto
  }
}

void funcCozinha(OSCMessage *_mes){                // Rotina que trata o comando OSC "/ard/cozinha"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);           // Armazena em value o argumento do comando OSC
"/ard/cozinha"
  if(value == HIGH) {                              // Verifica se potão foi pressionado
    Serial.println("Botao Cozinha Pressionado! ");   // Imprime que o botão foi pressionado

    int state = digitalRead(lcozinha);             // Faz a leitura do estado atual
    digitalWrite(lcozinha, !state);                // Muda o estado atual do relé (Liga ou desliga)
    Serial.print("Cozinha: ");
```

```
    Serial.println(!state);

  } else {                                  // Verifica quando o botão foi solto
    Serial.println("Botao Cozinha Solto! ");          // Imprime que o botão foi solto
  }
}


void funcFadeSala(OSCMessage *_mes){        // Rotina que trata o comando OSC "/ard/fadesala"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);    // Armazena em value o argumento do comando OSC
"/ard/fadesala"
  analogWrite(lsala, value);                // Escreve value (0 - 255) na led alto brilho da sala
  Serial.print("fadesala: ");
  Serial.println(value);                    // Imprime valor de fadesala atual

  if(value == LOW) {
    OSCMessage txMesSS;                     // Cria uma nova mensagem
    txMesSS.setAddress(destIP,destPort);    // Define endereço e porta da mensagem OSC
    txMesSS.beginMessage("/ard/StatusSala");  // Define comando OSC
    txMesSS.addArgFloat(LOW);               // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSS);                  // Envia mensagem OSC para atualizar o estado da sala
  } else {
      OSCMessage txMesSS;                   // Cria uma nova mensagem
      txMesSS.setAddress(destIP,destPort);  // Define endereço e porta da mensagem OSC
      txMesSS.beginMessage("/ard/StatusSala");  // Define comando OSC
      txMesSS.addArgFloat(HIGH);            // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
      client.send(&txMesSS);                // Envia mensagem OSC para atualizar o estado da sala
  }
}

void funcFadeR(OSCMessage *_mes){           // Rotina que trata o comando OSC "/ard/fadeR"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);    // Armazena em value o argumento do comando OSC
"/ard/fadeR"
  analogWrite(lq1_fadeR, value);            // Escreve value (0 - 255) na led Red do Quarto 1
  Serial.print("Quarto 1 (Vermelho): ");
  Serial.println(value);                    // Imprime valor de fadeR

  if(value == LOW) {
    OSCMessage txMesSQ1;                    // Cria uma nova mensagem
    txMesSQ1.setAddress(destIP,destPort);   // Define endereço e porta da mensagem OSC
    txMesSQ1.beginMessage("/ard/StatusQuarto1"); // Define comando OSC
    txMesSQ1.addArgFloat(LOW);              // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSQ1);                 // Envia mensagem OSC para atualizar o estado da sala
  } else {
      OSCMessage txMesSS;                   // Cria uma nova mensagem
      txMesSS.setAddress(destIP,destPort);  // Define endereço e porta da mensagem OSC
      txMesSS.beginMessage("/ard/StatusQuarto1");// Define comando OSC
      txMesSS.addArgFloat(HIGH);            // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
      client.send(&txMesSS);                // Envia mensagem OSC para atualizar o estado da sala
  }

}
```

```
void funcFadeG(OSCMessage *_mes){              // Rotina que trata o comando OSC "/ard/fadeG"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);       // Armazena em value o argumento do comando OSC
"/ard/fadeG"
  analogWrite(lq1_fadeG, value);               // Escreve value (0 - 255) na led Green do Quarto 1
  Serial.print("Quarto 1 (Verde): ");
  Serial.println(value);                       // Imprime valor de fadeG

  if(value == LOW) {
    OSCMessage txMesSQ1;                       // Cria uma nova mensagem
    txMesSQ1.setAddress(destIP,destPort);      // Define endereço e porta da mensagem OSC
    txMesSQ1.beginMessage("/ard/StatusQuarto1"); // Define comando OSC
    txMesSQ1.addArgFloat(LOW);                 // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSQ1);                    // Envia mensagem OSC para atualizar o estado da sala
  } else {
      OSCMessage txMesSS;                      // Cria uma nova mensagem
      txMesSS.setAddress(destIP,destPort);     // Define endereço e porta da mensagem OSC
      txMesSS.beginMessage("/ard/StatusQuarto1");// Define comando OSC
      txMesSS.addArgFloat(HIGH);               // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
      client.send(&txMesSS);                   // Envia mensagem OSC para atualizar o estado da sala
  }

}

void funcFadeB(OSCMessage *_mes){              // Rotina que trata o comando OSC "/ard/fadeB"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);       // Armazena em value o argumento do comando OSC
"/ard/fadeB"
  analogWrite(lq1_fadeB, value);               // Escreve value (0 - 255) na led Blue do Quarto 1
  Serial.print("Quarto 1 (Azul): ");
  Serial.println(value);                       // Imprime valor de fadeB

  if(value == LOW) {
    OSCMessage txMesSQ1;                       // Cria uma nova mensagem
    txMesSQ1.setAddress(destIP,destPort);      // Define endereço e porta da mensagem OSC
    txMesSQ1.beginMessage("/ard/StatusQuarto1");// Define comando OSC
    txMesSQ1.addArgFloat(LOW);                 // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSQ1);                    // Envia mensagem OSC para atualizar o estado da sala
  } else {
      OSCMessage txMesSS;                      // Cria uma nova mensagem
      txMesSS.setAddress(destIP,destPort);     // Define endereço e porta da mensagem OSC
      txMesSS.beginMessage("/ard/StatusQuarto1");// Define comando OSC
      txMesSS.addArgFloat(HIGH);               // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
      client.send(&txMesSS);                   // Envia mensagem OSC para atualizar o estado da sala
  }


}

void funcDesliga(OSCMessage *_mes){                // Rotina que trata o comando OSC "/ard/desliga"
recebido do Iphone
  int value = (int)_mes->getArgFloat(0);           // Armazena em value o argumento do comando OSC
"/ard/desliga"
  if(value == HIGH){                               // Se velue for alto
    Serial.println("Desligar Pressionado!");       // Imprime que o botão desligar foi pressionando
```

```
  //Rotina para desligar as lampadas
  int StatusBanheiro = digitalRead(lbanheiro);
  if(StatusBanheiro == LOW){                    // Verifica se a iluminação banheiro está ligada
    digitalWrite(lbanheiro, HIGH);              // Apaga a lâmpada
    Serial.println("Banheiro Desligou!");       // Imprime que a lampada foi apagada
  } else {                                      // Se não
     Serial.println("Banheiro Ja Desligada!");  // Imprime que a lampada já estava apagada
  }

  int valorldrQ2 = analogRead(sensorldrQ2);     // Faz a leitura do sensor ldr do quarto 2
  if (valorldrQ2 > 900) {                       // Verica se a luminosidade está alta indicando lampada acesa
    int stateQ2 = digitalRead(lquarto2);        // Faz a leitura do estado atual do relé
    digitalWrite(lquarto2, !stateQ2);           // Inverte o estado atual do relé, desligando a lâmpada
    Serial.println("Quarto 2 Desligou!");       // Imprime que a lâmpada do quarto 2 foi desligada
  } else {
     Serial.println("Quarto 2 Ja Desligado!");  // Imprime que a lâmpada do quarto 2 já estava
desligada
  }

  int valorldrQ3 = analogRead(sensorldrQ3);     // Faz a leitura do sensor ldr do quarto 3
  if (valorldrQ3 > 900) {                       // Verica se a luminosidade está alta indicando lampada acesa
    int stateQ3 = digitalRead(lquarto3);        // Faz a leitura do estado atual do relé
    digitalWrite(lquarto3, !stateQ3);           // Inverte o estado atual do relé, desligando a lâmpada
    Serial.println("Quarto 3 Desligou!");       // Imprime que a lâmpada quarto 3 foi desligada
  } else {
     Serial.println("Quarto 3 ja Desligado!");  // Imprime que a lâmpada quarto 3 já estava desligada
  }

  int valorldrCoz = analogRead(sensorldrCoz);   // Faz a leitura do sensor ldr da cozinha
  if (valorldrCoz > 900) {                      // Verica se a luminosidade está alta indicando lampada acesa
    int stateCoz = digitalRead(lcozinha);       // Faz a leitura do estado atual do relé
    digitalWrite(lcozinha, !stateCoz);          // Inverte o estado atual do relé, desligando a lâmpada
    Serial.println("Cozinha Desligou!");        // Imprime que a lâmpada da cozinha foi desligada
  } else {
     Serial.println("Cozinha Ja Desligada!");   // Imprime que a lâmpada da cozinha já estava
desligada
  }

  int StatusSala = digitalRead(lsala);
  if(StatusSala == LOW ){                       // Verifica se a lampada está ligada
    Serial.println("Sala Ja Desligada!");       // Imprime que a lampada já estava apagada
  } else {                                      // Se não
    digitalWrite(lsala, LOW);                   // Apaga a lâmpada
    Serial.println("Sala Desligou!");           // Imprime que a lampada foi apagada

    OSCMessage txMesSS;                         // Cria uma nova mensagem
    txMesSS.setAddress(destIP,destPort);        // Define endereço e porta da mensagem OSC
    txMesSS.beginMessage("/ard/StatusSala");    // Define comando OSC
    txMesSS.addArgFloat(LOW);                   // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
    client.send(&txMesSS);                      // Envia mensagem OSC para atualizar o estado da sala

    OSCMessage txMesFS;                         // Cria uma nova mensagem
    txMesFS.setAddress(destIP,destPort);        // Define endereço e porta da mensagem OSC
    txMesFS.beginMessage("/ard/fadesala");      // Define comando OSC
    txMesFS.addArgFloat(0);                     // Define novo estado do fade sala que será enviado no
argumento OSC para o Iphone
    client.send(&txMesFS);                      // Envia mensagem OSC para atualizar o estado do fade sala
  }
```

```
   int stateR = digitalRead(lq1_fadeR);             // Faz a leitura do estado atual do Led R
   int stateG = digitalRead(lq1_fadeG);             // Faz a leitura do estado atual do Led G
   int stateB = digitalRead(lq1_fadeB);             // Faz a leitura do estado atual do Led B
   if(stateR>LOW || stateG>LOW || stateB>LOW){       // Se um dos leds estiver ligado
     digitalWrite(lq1_fadeR, LOW);                  // Desliga o led R
     digitalWrite(lq1_fadeG, LOW);                  // Desliga o led G
     digitalWrite(lq1_fadeB, LOW);                  // Desliga o led B
     Serial.println("Led RGB Desligou! ");          // Imprime que o led RGB foi apagado

       OSCMessage txMesSQ1;                        // Cria uma nova mensagem
       txMesSQ1.setAddress(destIP,destPort);       // Define endereço e porta da mensagem OSC
       txMesSQ1.beginMessage("/ard/StatusQuarto1");// Define comando OSC
       txMesSQ1.addArgFloat(LOW);                  // Define valor do estado atual que será enviado no
argumento OSC para o Iphone
       client.send(&txMesSQ1);                    // Envia mensagem OSC para atualizar o estado da lampada

       OSCMessage txMesFR;                         // Cria uma nova mensagem
       txMesFR.setAddress(destIP,destPort);        // Define endereço e porta da mensagem OSC
       txMesFR.beginMessage("/ard/fadeR");         // Define comando OSC
       txMesFR.addArgFloat(0);                     // Define novo estado 0 do led que será enviado no
argumento OSC para o Iphone
       client.send(&txMesFR);                      // Envia mensagem OSC para atualizar o estado do fadeR

       OSCMessage txMesFG;                         // Cria uma nova mensagem
       txMesFG.setAddress(destIP,destPort);        // Define endereço e porta da mensagem OSC
       txMesFG.beginMessage("/ard/fadeG");         // Define comando OSC
       txMesFG.addArgFloat(0);                     // Define novo estado 0 do led que será enviado no
argumento OSC para o Iphone
       client.send(&txMesFG);                      // Envia mensagem OSC para atualizar o estado do fadeG

       OSCMessage txMesFB;                         // Cria uma nova mensagem
       txMesFB.setAddress(destIP,destPort);        // Define endereço e porta da mensagem OSC
       txMesFB.beginMessage("/ard/fadeB");         // Define comando OSC
       txMesFB.addArgFloat(0);                     // Define novo estado 0 do led que será enviado no
argumento OSC para o Iphone
       client.send(&txMesFB);                      // Envia mensagem OSC para atualizar o estado do fadeB

   } else {
       Serial.println("RGB Ja Desligado!");        // Informa que o led RGB já estava apagado
     }
  } else {                                  // Ao soltar o botão
       Serial.println("Botao Desligar Solto!");    // Imprime que o botão foi solto
   }
}
```