UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
DEPARTAMENTO ACADÊMICO DE MECÂNICA
CURSO SUPERIOR DE TECNOLOGIA EM MECATRÔNICA INDUSTRIAL

DANILO MATTIAZZO GORJON
LEANDRO BAGATIM PEDRO

SISTEMA DE AUTOMAÇÃO RESIDENCIAL COM SERVIDOR *WEB*

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2015

DANILO MATTIAZZO GORJON
LEANDRO BAGATIM PEDRO

SISTEMA DE AUTOMAÇÃO RESIDENCIAL COM SERVIDOR *WEB*

Trabalho de Conclusão de Curso de graduação em Tecnologia Em Mecatrônica Industrial, do Departamento Acadêmico de Eletrônica e do Departamento Acadêmico de Mecânica, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de tecnólogo.
Orientador: Prof. Valmir de Oliveira.

CURITIBA
2015

MINISTÉRIO DA EDUCAÇÃO
Universidade Tecnológica Federal do Paraná
Campus Curitiba
Departamento Acadêmico de Eletrônica
Departamento Acadêmico de Mecânica
Curso de Tecnologia em Mecatrônica Industrial

TERMO DE APROVAÇÃO

DANILO MATTIAZZO GORJOM
LEANDRO BAGATIM PEDRO

SISTEMA DE AUTOMAÇÃO RESIDENCIAL COM SERVIDOR WEB

Este trabalho de conclusão de curso foi apresentado no dia 11 de dezembro de 2015, como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial, outorgado pela Universidade Tecnológica Federal do Paraná. O aluno foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Milton Luiz Polli
Coordenador de Curso
Departamento Acadêmico de Mecânica

_____
Prof. Esp. Sérgio Moribe
Responsável pela Atividade de Trabalho de Conclusão de Curso
Departamento Acadêmico de Eletrônica

BANCA EXAMINADORA

_____
Prof. Marcio Augusto Lombardi
UTFPR

_____
Prof. Joel Gonçalves Pereira
UTFPR

_____
Prof. Valmir de Oliveira
Orientador - UTFPR

OBS.: O documento assinado encontra-se em posse da coordenação do curso.

DEDICATÓRIA

A meus pais, Lucia e Marcio, pela infinita paciência, aos amigos que me cobraram e a Emanuelle pela paciência e apoio.
(Danilo Mattiazzo Gorjon)

Um agradecimento especial ao meu pai, Milton, pelas cobranças e apoio, a toda minha família, amigos e a Fran, pela paciência e compreensão.
(Leandro Bagatim Pedro)

AGRADECIMENTOS

Agradecemos em especial ao nosso orientador, Valmir de Oliveira, pois sem a sua orientação esse trabalho não teria se concretizado.

Aos professores do DAELN e DAMEC por todo conhecimento transmitido, a Universidade Tecnológica Federal do Paraná – campus Curitiba, pelo espaço cedido para realização desse trabalho e a todos os demais professores que de alguma forma contribuíram para a nossa formação, obrigada pelo conhecimento transmitido.

Aos professores da banca examinadora, por terem dedicado seus tempos para avaliar e contribuir com o estudo.

E a todos amigos e entes querido que nos ajudaram a superar a nós mesmos e aceitaram nossa ausência no tempo utilizado no desenvolvimento desse trabalho.

RESUMO

GORJON, Danilo Mattiazzo; PEDRO, Leandro Bagatim. **Sistema de automação residencial com servidor *web***. 2015. 60 f. Trabalho de Conclusão de Curso superior de tecnologia em mecatrônica industrial - Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

Este trabalho apresenta um estudo teórico-prático sobre um sistema de automação residencial. Apresenta tópicos sobre microcontroladores, Arduino, redes e programação, descrevendo equipamentos e tecnologias. Discorre sobre a escolha dos equipamentos utilizados, os testes para assegurar um bom funcionamento dos dispositivos e o desenvolvimento de códigos de programação para esse sistema. Como resultado desse trabalho, foi realizado a montagem de um protótipo, capaz de ler sensores e acionar atuadores, e o desenvolvimento de um sistema de supervisão e controle.

Palavras-Chave: Automação residencial. Arduino. Aplicativo *smartphone*. Aplicativo Android.

ABSTRACT

GORJON, Danilo Mattiazzo; PEDRO, Leandro Bagatim. **Home automation system with web server**. 2015. 60 f. Trabalho de Conclusão de Curso superior de tecnologia em mecatrônica industrial - Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

This work presents a theorical and practical study about of a home automation system. It presents topics about microcontrollers, Arduino, networks and programming, describing equipments and technologies. Discusses the choice of the equipaments used, the tests to ensure proper functioning of the devices and the development of programming codes for this system. As the result of this work, it is performed the assembly of a prototype, capable of reading sensors and operate actuators, and the development of a supervision and control system.

Keywords: Home automation system. Arduino. Aplication *smartphone*. Aplication Android.

LISTA DE FIGURAS

LISTA DE SIGLAS E ACRÔNIMOS

AI	Automação Industrial
AJAX	Javascript Assíncrono e XML - *Asynchronous Javascript and XML*
AP	Automação Predial
AR	Automação Residencial
CI	Circuito Integrado
CSS	Folhas de estilo em cascata - C*ascading Style Sheets*
HTML	Linguagem de Marcação de Hipertexto *(HyperText Markup Language)*
IDE	Ambiente de desenvolvimento integrado (*Integrated Development Environment)*
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos *(Institute for Electrical and Electronics Engineers)*
ISO	Organização Internacional para a Normalização *(Internacional Organization for Standardization)*
LAN	Rede de área local (*Local Area Network)*
LED	Diodo emissor de luz (*Light Emitter Diode)*
OSI	Interconexão de Sistemas Abertos (*Open Systems Interconnection)*
PWM	Modulação por largura de pulso (*Pulse-Width Modulation)*
W3C	Consórcio *World Wide Web (World Wide Web Consortium)*
WLAN	Rede de área local sem-fio *(Wireless Local Area Network)*

SÚMARIO

1 INTRODUÇÃO

Domótica, resultado da junção das palavras casa, do latim “*domus*”, com a palavra de origem Tcheca “*robotnik*”, que significa servo, é um termo utilizado para descrever a automação residencial, no sentido de um serviço para a casa. O conceito de automação residencial sugere um produto ou serviço capaz de gerar alguma ação sem a intervenção de uma pessoa. Em níveis mais básicos, pode-se afirmar que um alarme de incêndio ou um despertador são dispositivos de automação residencial, ou seja, trata-se de qualquer tecnologia com o objetivo de coordenar os recursos da residência de forma automatizada.

As primeiras aplicações da tecnologia de automação residencial datam do fim da década de 1970, nos EUA, com módulos que consistiam em comandos enviados pela rede elétrica com o objetivo de ligar luzes ou equipamentos. Com a evolução tecnológica, as residências ficaram cada vez mais equipadas, demandando maior automatização e integração entre seus componentes. Com o advento dos microprocessadores e a popularização dos computadores pessoais, nas décadas de 1980 e 1990, passou a ser possível a aplicação de sistemas de controle residenciais, de forma portátil e inteligente (MURATORI; DAL BÓ, 2013).

Os sistemas de automação residencial são desenvolvidos com o intuito de otimizar o funcionamento da casa, atuando na racionalização de recursos, como energia e água, na segurança patrimonial e no cotidiano do usuário de forma geral. Esses sistemas de controle têm se mostrado muito eficazes para usuários com limitações físicas, oferecendo conforto e independência para essas pessoas.

De forma mais complexa tem-se a ideia de “casa inteligente” que consiste em um sistema totalmente integrado que atenda as programações do usuário e também tenha capacidade de responder automaticamente a alterações no ambiente. Esse conceito ainda é muito restrito, principalmente pelo seu alto custo de implantação, pois exige projeto específico e equipamentos compatíveis entre si. A popularização dos sistemas de automação residencial esbarra na falta de padronização dos dispositivos de automação, dificultando a integração entre os equipamentos e a centralização do comando desses em um único dispositivo de controle.

Com a possibilidade de acesso à internet de casa, do trabalho e até mesmo de um *smartphone*, a utilização de *softwares* para controle e verificação remota de componentes residências, como por exemplo o acesso de câmeras de segurança, é algo cada vez mais comum, trazendo a preocupação com o desenvolvimento de interfaces mais simples e intuitivas.

Dentro desse contexto, o presente trabalho visa o desenvolvimento de um protótipo de baixo custo capaz de realizar controle de automação através de um acesso via internet.

2 OBJETIVO

2.1 Objetivo Geral

O presente trabalho tem por objetivo desenvolver um sistema de automação residencial com equipamentos de baixo custo, utilizando para manipulação e monitoramento dos elementos sob controle de uma residência, um *smartphone* ou uma página *web*.

2.1.1 Objetivos Específicos

Os objetivos específicos desse trabalho são:

- Definir os tipos de sensores a serem utilizados e os equipamentos a serem controlados;
- Definir o microcontrolador a ser utilizado;
- Elaborar projeto de intercomunicação entre dispositivos;
- Construir bancada de teste;
- Configurar equipamentos de rede;
- Desenvolver *softwares* para controle do microcontrolador e *smartphone*;
- Desenvolver página *web* e hospedá-la no microcontrolador;
- Fazer a integração do sistema.

2.2 Justificativa

Os sistemas de controle de automação estão cada vez mais presentes nas residências permitindo através desses, melhorias no dia a dia, facilitando atividades domésticas, otimizando recursos da residência ou mesmo auxiliando na segurança patrimonial.

Hoje a automação residencial não é mais considerada como algo supérfluo, sendo vista como uma ferramenta importante para acompanhar o atual modelo de vida das pessoas.

É possível encontrar no mercado soluções projetadas e desenvolvidas por empresas do ramo, porém o alto custo e a necessidade de mão de obra especializada diminuem a abrangência desse tipo de tecnologia.

Esse trabalho busca viabilizar uma alternativa de baixo custo na automação residencial com equipamentos disponíveis no mercado, além de disponibilizar os códigos-fonte dos *softwares* desenvolvidos aos interessados para futuras aplicações.

3 REVISÃO BIBLIOGRÁFICA

3.1 Automação Residencial

A automação pode ser dividida em três vias principais: a Automação Residencial (AR), a Automação Predial (AP) e a Automação Industrial (AI).

A AR é uma derivação da automação predial responsável pelo controle das atividades domésticas, com o intuito de diminuir a intervenção humana por meio de sistemas de controle de gerenciamento de equipamentos (BOLZANI, 2010). Diferente da AI, onde o perfil de usuário, é padronizado, com funções e tarefas bem definidas, na AR as possibilidades de interação são mais complexas, variando de uma residência para outra, de acordo com os equipamentos a serem automatizados e a forma de integração destes, e de um usuário para outro, na sua forma individual de interagir com sua residência.

Para Muratori e Dal Bó (2011), a característica básica de uma residência automatizada é a integração entre os diversos sistemas da residência aliada a execução de comandos e funções de forma programada. Segundo Dias e Pizzolato (2004), os sistemas centralizados de AR possuem um equipamento central em que todos os outros estão conectados, este recebe as informações de entrada adquiridas através de sensores e realiza o processamento das informações gerando uma resposta que é transmitida para equipamentos atuadores, enquanto nos sistemas distribuídos, há diversos dispositivos com processamento individual que são interligados por uma rede. A dificuldade para integração na AR se deve a diversidade de tecnologias e protocolos de comunicação.

De acordo com o nível de automatização, pode-se classificar os sistemas de AR como: autônomo, onde há apenas a ação de ligar e desligar um sistema isolado da residência, integrado, onde uma central de controle gerencia vários sistemas de forma integrada, e complexo, onde o sistema, que é totalmente integrado e programado de acordo com as necessidades do usuário, é seu próprio gerenciador que recebe e interpreta as entradas de comando, processa e retorna a ação programada (MURATORI; DAL BÓ, 2013).

3.2 Microcontroladores

Os microcontroladores são circuitos integrados que possuem num único dispositivo todos os circuitos necessários para realizar um completo sistema programável, logo, para serem utilizados, necessitam ser programados (FRANCISCO, 2007).

Esses dispositivos se diferenciam de microprocessadores, pois além dos componentes lógicos e aritméticos usuais de um microprocessador, o microcontrolador integra em sua estrutura elementos como memória de leitura e escrita para armazenamento de dados, memória para leitura e armazenamento de programas, conversores analógico/digitais, conversores digitais/analógicos e interfaces de entrada e saída. A Figura 1 mostra a arquitetura básica de um microcontrolador.

Devido as suas interfaces de entrada e saída, os microcontroladores são capazes de se integrar a outros circuitos, de modo a incorporar novas funcionalidades. Dentre estas funcionalidades, está a capacidade da utilização de circuitos que permitem a um microcontrolador se conectar à internet ou rede *ethernet* local e assim controlar e ser controlado/monitorado por outros dispositivos capazes de interpretar este protocolo de rede. Pode também gerenciar o acesso a um dispositivo de armazenamento, sendo assim, é capaz de hospedar uma página *web*, programada, por exemplo, em HTML, de modo a dar uma interface gráfica simples e intuitiva para monitoramento e controle pelo usuário.

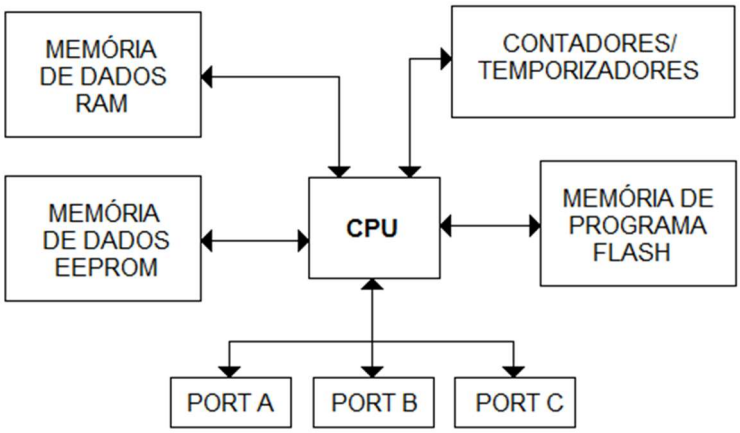

Figura 1 - Arquitetura básica de um microcontrolador
Fonte: Autoria própria

3.3 Arduino e Shield *Ethernet*

O Arduino é uma plataforma de desenvolvimento com *hardware* e *software* livres, isso significa que tanto o seu esquema elétrico quanto a programação estão disponibilizados na internet para serem alterados e utilizados. As placas Arduino são capazes de ler entradas, como um sensor de luminosidade, um botão ou um sensor de temperatura, e transformar isso em uma saída, ativando um motor, ligando um LED ou um *cooler* (ARDUINO, 2015).

A Figura 2 mostra um modelo de placa Arduino, especificamente, o Arduino Uno.

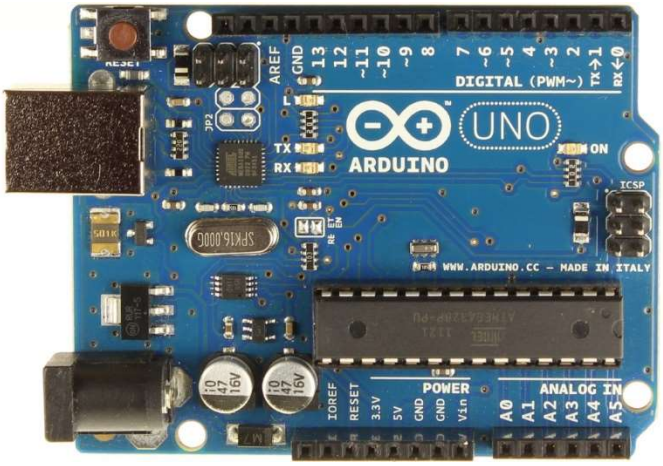

Figura 2 - Arduino Uno.
Fonte: ARDUINO,2015.

A placa consiste em circuitos com entradas e saídas para o microcontrolador AVR, no caso do Arduino Uno utiliza-se o microcontrolador ATmega328P, esse modelo de placa possui 14 pinos de entradas e saídas digitais dos quais 6 deles podem ser utilizados como saídas moduladas por largura de pulso - PWM, 6 pinos de entrada analógica, um oscilador de cristal com frequência de 16 MHz e 32 kb de memória *flash* utilizada para gravação do programa.

Para possibilitar o uso do Arduino através de rede *Ethernet* é necessário a utilização de um *Shield*, que é um circuito desenvolvido especificamente para o Arduino e permite, após acoplado, a disponibilidade de uma saída conectável por um cabo de rede RJ45 e a utilização de um cartão de memória micro-SD.

A figura 3 mostra um *Shield Ethernet* compatível com o Arduino Uno.

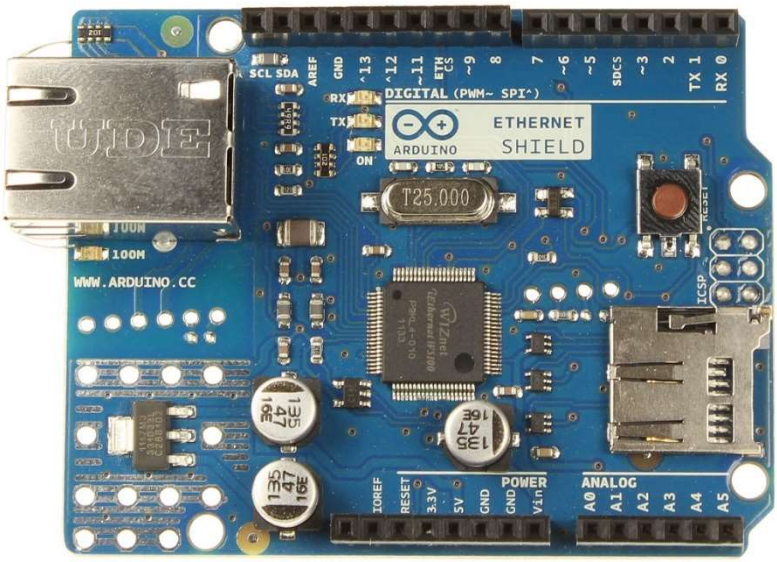

Figura 3 - Shield Ehernet
Fonte: ARDUINO,2015.

3.4 Rede de Computador

De acordo com Torres (2014), as redes de computadores surgiram com a finalidade de possibilitar o acesso de uma informação que está fisicamente longe. Mesmo não sendo uma tecnologia recente, a criação de novas padronizações e tecnologias complementares tornaram possível a comunicação entre equipamentos de forma ágil e com baixo custo, o maior exemplo disso é a internet.

3.4.1 Descrição de Modelo OSI

Para evitar soluções proprietárias, onde os fabricantes eram responsáveis por todos os aspectos das redes (equipamentos, cabos, placas, protocolos, *softwares*), impossibilitando a interconexão de sistemas computacionais, a ISO (*Internacional Organization for Standardization*; em tradução livre, Organização Internacional para Padronização) desenvolveu um modelo referencial teórico chamado de OSI (*Open Systems Interconnection*) (TORRES, 2014).

O modelo OSI teórico possui 7 camadas com um protocolo responsável por camada, a figura 4 mostra a comunicação do ponto de vista do transmissor e do receptor utilizando o modelo.

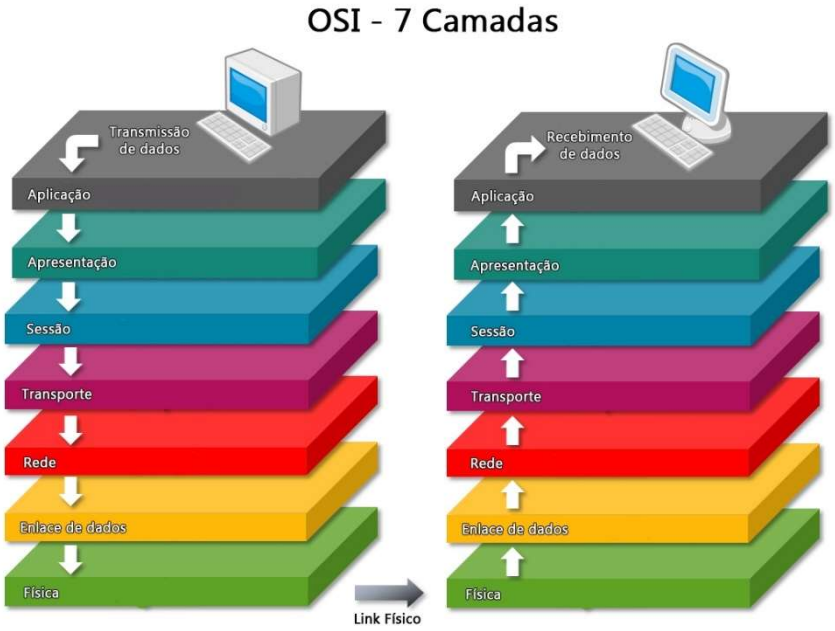

Figura 4 - Camadas modelo OSI
Fonte: https://marrciohenrique.wordpress.com/2014/03/22/modelo-tcpip-e-osi/

As breves descrições das camadas a seguir estão segundo as definições de Tanenbaum (2003) e Microsoft (2013).

Física: Trata-se da transmissão de *bit* brutos por um canal de comunicação, garantindo que quando o emissor envia um *bit* de valor 1 o receptor recebe o mesmo valor 1, válido também caso emissor envie *bit* de valor 0.

Enlace de dados: Principal função dessa camada é garantir que um quadro, que é um conjunto de centenas ou milhares de *bytes* (conjunto de 8 *bit*s), chegue ao receptor sem erros.

Rede: Responsável por definir como os pacotes serão roteados até o destino. Podem ser uma tabela estática, definido em cada comunicação ou completamente dinâmica. Um projeto ruim dessa camada pode gerar alta instabilidade, maior tempo de trânsito ou retardo.

Transporte: Função básica dessa camada, quando está na ponta emissora, é receber dados da camada de sessão e, se necessário, dividir em pedaços menores e então passar para a camada de rede. Já na ponta receptora é receber dados da camada de rede, organizar os pedaços igual antes da divisão e repassar para camada acima.

Sessão: Permite que diferentes maquinas estabeleçam sessões entre elas, podendo utilizarem-se de serviços, tais quais, controle de diálogo, gerenciamento de *token* e sincronização.

Apresentação: Essa camada se preocupa com conversão dos caracteres para um que independente da aplicação sendo utilizada, compactação da informação e criptografia, ou seja, permite que a mensagem seja transmitida de forma segura, mais rápida e garante que o usuário final possa ler a mensagem com qualquer aplicação compatível.

Aplicação: A camada mais alta faz a interface entre os processos de aplicativos e a pilha de protocolos da rede, importante pois o protocolo a ser utilizado depende do tipo de solicitação realizado.

3.4.2 Ethernet

Segundo Spurgeon (2000), uma rede *Ethernet* é feita de *Hardware* e *Software* trabalhando junto para entregar uma informação digital entre computadores, esse tipo de rede é padronizada pelo *Institute for Electrical and Electronics Engineers* (IEEE) com ajuda de engenheiros que trabalham na indústria de tecnologia, organizações governamentais e voluntários de todos os lugares. Atualmente a IEEE 802.3 é a padronização oficial da *Ethernet*.

Ethernet é um conjunto de tecnologias padronizadas para uso de uma rede local ou LAN (*Local Area Network*) que permite a conexão de uma variedade de dispositivos, com baixo custo e um sistema de rede extremamente flexível. Praticamente todos os modelos de computadores hoje suportam o protocolo *Ethernet* e este amplo alcance somado as vantagens apresentadas acima, são as principais razões para a popularidade da rede *Ethernet*.

3.4.3 Wi-Fi

Como lembra o autor O'HARA (2005), em 1997 foi publicada a primeira padronização para redes sem fio ou WLAN (*Wireless Local Area Network)*, cujo o objetivo era descrever uma WLAN que poderia entregar serviços previamente encontrados apenas em redes com fio, as especificações para esse tipo de rede

podem ser localizadas no padrão IEEE 802.11. O autor ainda diz que a tecnologia WLAN superou seu modesto mercado original, ocasionando uma rápida aceitação e utilização por milhões de usuários sem experiência em suas residências e atribuiu como principal razão para isso a padronização IEEE 802.11 e suas seguintes alterações.

O nome popularizado das redes sem fio, WI-FI, veio da associação Wi-Fi *Alliance* formada por um conjunto de empresas, que tem como objetivo assegurar um funcionamento adequado dos equipamentos utilizando-se do padrão IEEE 802.11 independentemente da marca (WI-FI ALLIANCE, 2015).

A interligação entre equipamentos através do uso de redes WI-FI permite ao usuário utilizar a rede em qualquer ponto dentro dos limites de alcance da transmissão, possibilita a inserção rápida de outros computadores e dispositivos na rede e ainda, evita que paredes ou estruturas prediais sejam furadas ou adaptadas para a passagem de fios.

3.4.4 SMARTPHONE

Um *smartphone* é um termo usado para descrever uma categoria de dispositivos móveis com a funcionalidade de um computador pessoal. Estes dispositivos suportam um sistema operacional completo e têm uma plataforma para desenvolvedores de aplicativos.

Atualmente, as duas principais plataformas de *smartphones* em uso são Android (do Google) e iOS (da *Apple*). Uma aplicação escrita para uma plataforma específica geralmente pode trabalhar em qualquer *smartphone* usando a mesma plataforma. *Smartphones* têm telas maiores e processadores mais rápidos do que os chamados *feature phones* ou *dumb phones.*

3.4.5 SERVIDOR *WEB*

Soares (1986) diz que um servidor é uma estação interligada a uma rede que fornece um serviço para outras estações da rede.

No caso de um servidor *web*, WEB DEVELOPERS NOTES (2015) diz ter dois significados possíveis, pode ser utilizado para descrever o computador em que uma página está armazenada que deve necessariamente estar conectado à internet e possuir um endereço de internet fixo, ou também, como o programa nesse computador que processa solicitações realizadas, normalmente via HTTP, de clientes e as responde.

O uso mais comum de servidores *web* é para hospedar *sites*, mas existem outros usos, tais como jogos, armazenamento de dados, execução de aplicativos empresariais, a manipulação de e-mail, FTP, entre outros.

3.4.6 HTML

A Linguagem de Marcação de Hipertexto normalmente conhecida apenas por HTML vem da abreviação para a expressão em língua inglesa *HyperText Markup Language*, é uma linguagem de hipertexto utilizada para produzir páginas para w*eb* que podem ser interpretadas por navegadores. Para ROBBINS (2010), HTML é uma linguagem de marcação, ou seja, ser um sistema que identifica e descreve os componentes de um documento.

O *World Wide Web Consortium (*W3C) faz revisões do HTML quando necessário para atualizar e ampliar a linguagem possibilitando uma maior aplicação, a versão atual do HTML é a 5 (World Wide Web Consortium – W3C, 2015).

3.5 PROGRAMAÇÃO

Segundo Manzano e Oliveira (2012) o processo de programação pode ser descrito como uma comunicação entre um ser humano tecnicamente preparado e um computador, através de uma linguagem de programação, para que a máquina consiga entender.

Normalmente, o que precisa ser comunicado entre ser humano e um computador é um algoritmo. Cormen *et. al*. (2012) definem como algoritmo, qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída. Ele

ainda diz que um algoritmo é correto se, para toda instância de entrada, ele parar com a saída correta.

Entenda-se por computador, para a programação, um equipamento eletrônico capaz de executar algumas etapas de trabalho, como receber, armazenar, processar lógica e aritmeticamente dados com o objetivo principal de resolver problemas, (MANZANO; OLIVEIRA, 2012).

3.5.1 Linguagem C/C++

A linguagem de programação C foi desenvolvida por Dennis Ritchie, como uma evolução da linguagem B, entre outras características adicionadas a de maior relevância foi a utilização de tipos de dados, antes disso havia a necessidade do programador tratar os dados diretamente. A linguagem C++ foi projetado a partir da linguagem C, onde foram adicionados recursos que possibilitam o seu uso para programação orientada a objetos (DEITEL; DEITEL, 2013a).

C e C++ permitem a utilização de funções, o que agiliza muito o desenvolvimento de projetos e manutenção, pois é possível a utilização de funções já validadas e amplamente utilizadas, não havendo a necessidade de desenvolvimento dessas.

3.5.2 Java

Segundo Deitel e Deitel (2013b), a linguagem de programação Java foi desenvolvida pela empresa Sun Microsystems, no início da década de 90, através de um projeto de pesquisa corporativa, foi inicialmente pensado como uma linguagem para facilitar a convergência de computadores e eletrodomésticos, mas com a popularização da internet foi amplamente utilizada para criação de conteúdo dinâmico para páginas da *web*.

4 PROCEDIMENTOS METODOLÓGICOS

O protótipo foi elaborado com a finalidade de fazer o controle de automação de uma lâmpada, um sensor de temperatura e um servo motor, através de duas interfaces: uma página *web* e um aplicativo para *smartphone*. Para a lâmpada é possível saber seu estado atual e ligar/desligar o dispositivo. O sensor de temperatura faz a medição e são informados os valores em graus Celsius e Fahrenheit. Para o servo motor, é informada sua posição atual e é possível efetuar a atuação do mesmo. A figura 5 mostra uma representação simples do protótipo e nos parágrafos seguintes as descrições dos componentes.

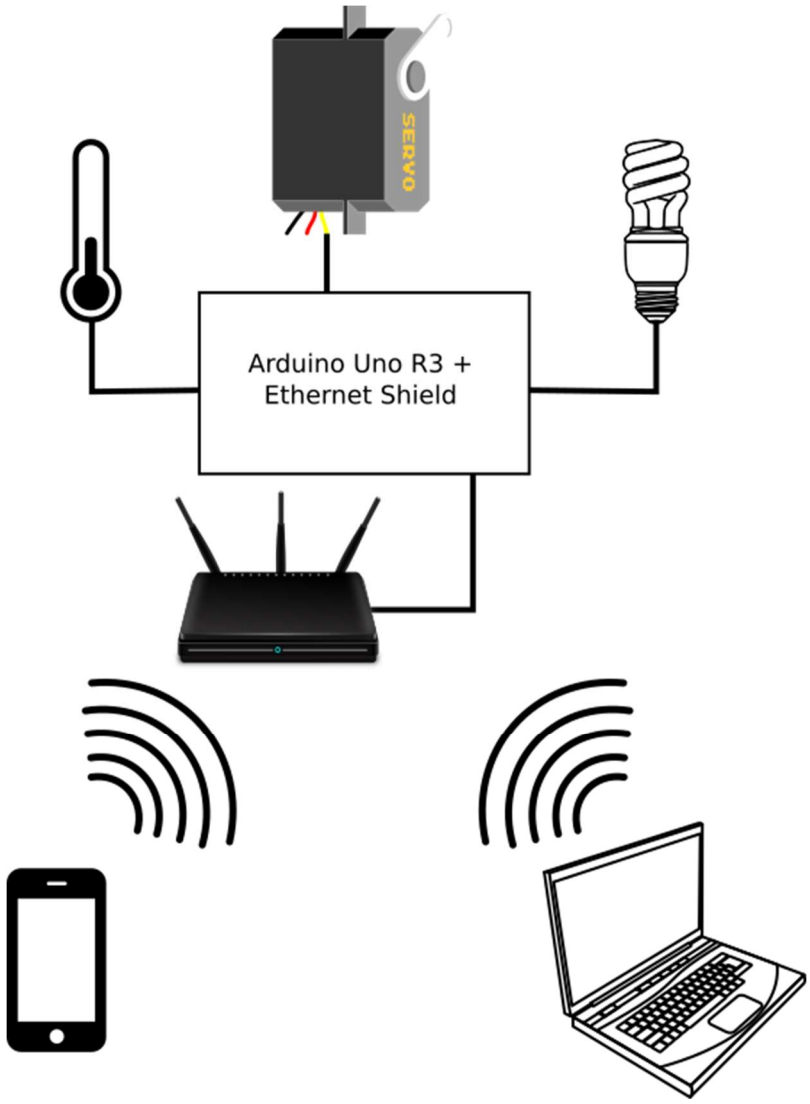

Figura 5 - Diagrama do protótipo
Fonte: Autoria própria

O controle do protótipo é composto por um microcontrolador Arduino Uno baseado no processador ATmega328P com o *Shield ethernet* acoplado, esse subsistema serve como um servidor web. O conjunto foi escolhido principalmente pelo baixo custo relacionado com os vários recursos disponíveis, além de possuir uma base de usuários grande que costumam compartilhar códigos desenvolvidos facilitando alterações nas funcionalidades. Conectados fisicamente à base central existem os componentes periféricos: os sensores, que têm como objetivo alimentar com informações o sistema para que essas sejam visíveis ao usuário e os atuadores, que permitem o usuário intervir no sistema.

Para realizar a medição de temperatura foi utilizado o CI LM35, devido a seu baixo custo e simplicidade de utilização, não ser necessária nenhuma calibração e possuir precisão de ±¼°C. O CI possui uma saída analógica que varia de acordo com a temperatura do ambiente (10 mV/ºC) e quando acoplado ao Arduino possibilita a conversão dessa variável para escalas comumente utilizadas para indicação de temperatura (ºC ou ºF).

Para o controle da lâmpada e do interruptor através do sistema, foram necessários dois componentes, o primeiro refere-se ao atuador SRD-05VCD-SL-C, figura 6, que é um relé de saída. Após o microcontrolador receber o comando dado via internet ele processa a informação acionando o relé que deixa passar corrente elétrica acendendo a lâmpada, o segundo é o sensor ACS721, figura 7, utilizado para medição da corrente elétrica com a finalidade de informar ao sistema se a lâmpada está ligada.

Figura 6 - Relé de saída SRD-05VCD-SL-C
Fonte: Autoria própria

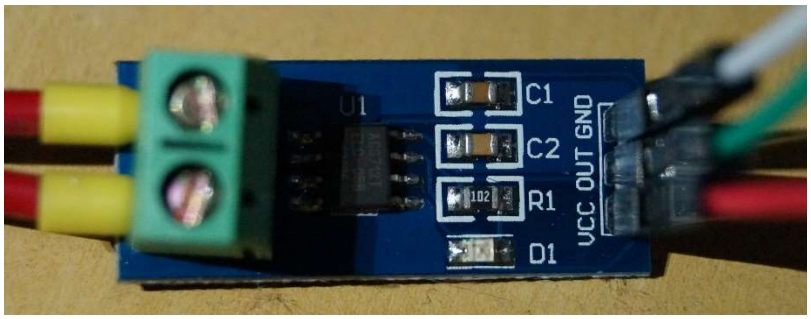

Figura 7 - Sensor de corrente ACS721
Fonte: Autoria própria

O servo motor MG 996R, figura 8, foi adicionado como um equipamento controlável e para permitir o controle preciso da posição do eixo foi desenvolvido uma função no código-fonte, aumentando as possibilidades de utilização do sistema de automação residencial desenvolvido.

Figura 8 - Servo motor MG 996R
Fonte: Autoria própria

4.1 Testes prévio

Para possibilitar o desenvolvimento integrado do sistema, foram necessários testes com os seus componentes, para garantir seu correto funcionamento. Para agilizar os testes foi utilizado uma *protoboard* visando facilitar a ligação elétrica dos componentes.

O primeiro teste realizado foi o desenvolvimento de um programa simples para piscar um LED existente na placa Arduino Uno, esse processo possibilitou a familiarização com as funções pré-definidas em sua biblioteca, o estudo de como realizar as ligações elétricas para alimentação do microcontrolador e a utilização do *software* Arduino IDE 1.0.5 para baixar o programa teste.

Em seguida foram realizados testes acoplando o microcontrolador e o *shield ethernet*, com o objetivo de ha*bit*uar-se com as funções de bibliotecas específicas para comunicação via *ethernet*, atribuindo um endereçamento ao dispositivo e testando a resposta via rede. Ainda com essa configuração verificou-se a eficácia do Arduino em armazenar informações no cartão microSD e através de uma solicitação via rede foi medido tempo de resposta de transmissão dessa informação. Demonstrando a viabilidade do conjunto como um servidor *web*.

4.2 Interligação do sistema

Para alimentação do sistema foi empregada uma fonte normalmente utilizada em computadores, ela transforma a tensão alternada 127 V ou 220 V fornecida pela rede, em tensão contínua 5V necessária para alimentar todos os equipamentos. Os sensores foram ligados nas entradas analógicas fornecidas pelo Arduino Uno, para que pudessem ser interpretadas pelo programa. O servo motor foi conectado a saída com PWM disponível, pois se tivesse sido ligado em uma saída digital sem essa função não seria possível realizar seu controle. O relé de acionamento da lâmpada foi ligado a uma saída digital simples.

O esquema elétrico resultante dessas conexões pode ser verificado na figura 9:

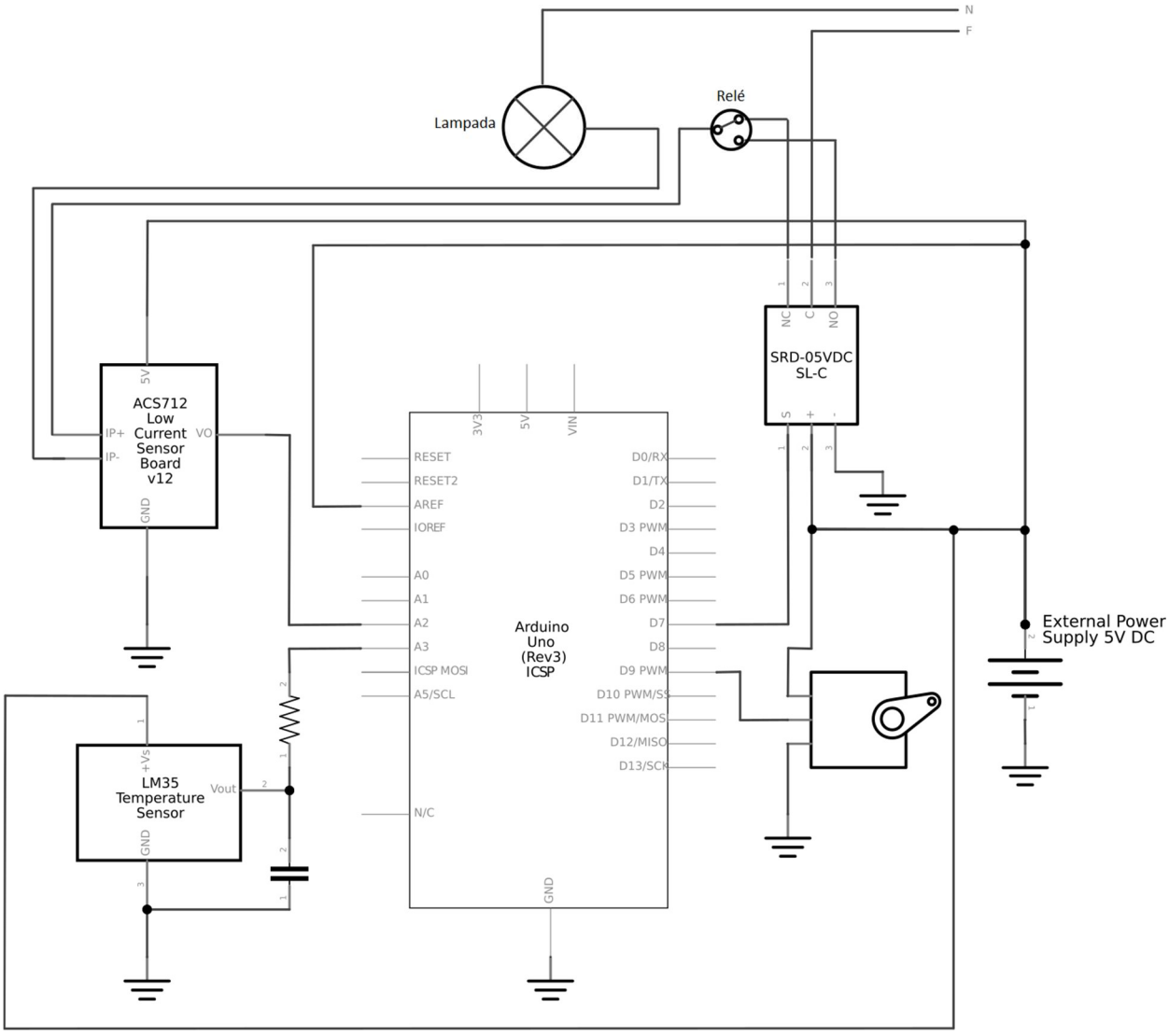

Figura 9 - Esquema elétrico
Fonte: Autoria Própria

A comunicação entre o servidor e o dispositivo utilizado para controle é realizado através de mensagens padronizadas trocadas em javascript e em Java, dependendo de onde o acesso é realizado, mas em ambos os casos o receptor da mensagem a decodifica e interpreta em ações, por exemplo, o servidor envia uma mensagem com valor de temperatura para o dispositivo utilizado na supervisão, esse interpreta que o valor recebido trata-se da temperatura e atualiza o valor mostrado, ou no outro sentido da comunicação, o usuário deseja acender a luz e envia o comando através do seu dispositivo, esse envia a mensagem para o servidor que entende que a luz deve ser acesa e aciona a saída apropriada.

4.3 Interface do usuário

Para permitir ao usuário uma fácil visualização e interação com o sistema, foram desenvolvidos dois métodos possíveis de acesso: o primeiro através de um navegador de internet, acessando uma página *web* desenvolvida em HTML, utilizando os recursos CSS, javascript e AJAX, com um editor de texto padrão, e o segundo, através do aplicativo para *smartphones* android escrito em Java, utilizando o *software* Android *Studio* 1.4.

A página *web* desenvolvida para controle do sistema fica salva no cartão microSD localizado no subsistema central, após a solicitação da página ao endereço correto programado no Arduino, este acessa o cartão microSD e transmite a página ao usuário solicitante. O que ele visualizará no navegador será uma página semelhante à figura 10:

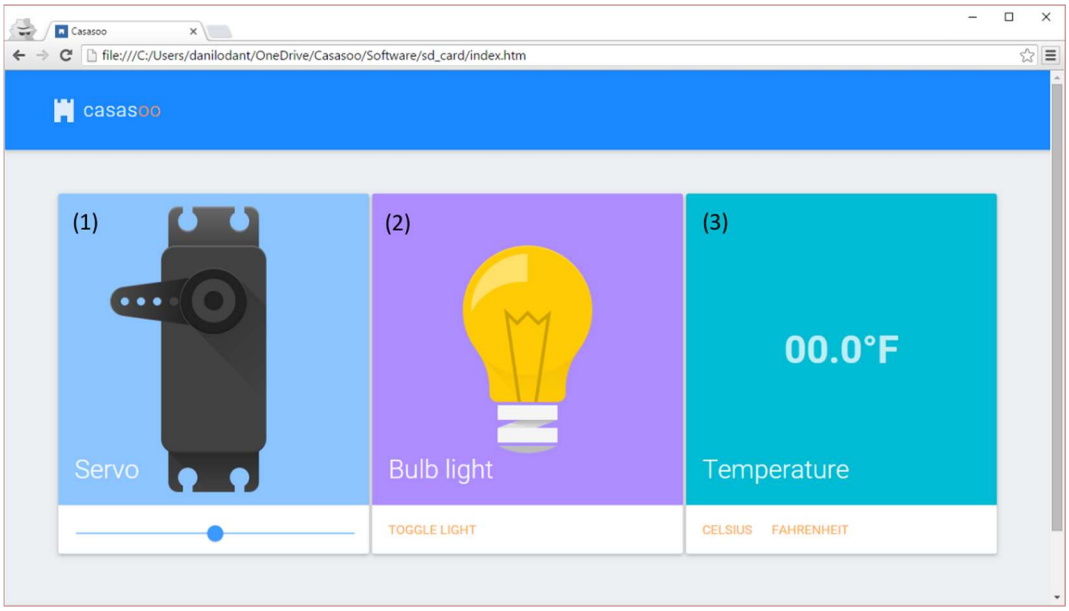

Figura 10 – Interface da página *WEB*
Fonte: Autoria própria

Desmembrando a figura 4, o quadro (1) mostra a imagem do servo na sua posição atual e a barra de *slide* onde é possível realizar o controle posicional do servo motor. No segundo quadro a lâmpada (*bulb light)* controlada está ilustrada em seu estado atual e abaixo dela é possível, através de um clique do *mouse,* alternar entre os estados de acesa ou apagada. Em (3) é mostrado a temperatura medida pelo sistema com a possibilidade de trocar a visualização entre graus Celsius e Fahrenheit.

O aplicativo deve ser instalado em um *smartphone* android e acessado para realizar a supervisão e controle do sistema, lembrando que grande parte dos *smartphones* possuem navegadores e é possível acessar e controlar através da página, mas o aplicativo tem a vantagem de ter sido desenvolvido e otimizado para permitir uma melhor experiência para quem for utilizar o celular como forma de acesso.

Semelhante ao controle previamente descrito para a página web a Figura 11 ilustra as três telas de controle no aplicativo, em (1) o controle do servo motor, em (2) o estado e acionamento da lâmpada e em (3) a medição de temperatura.

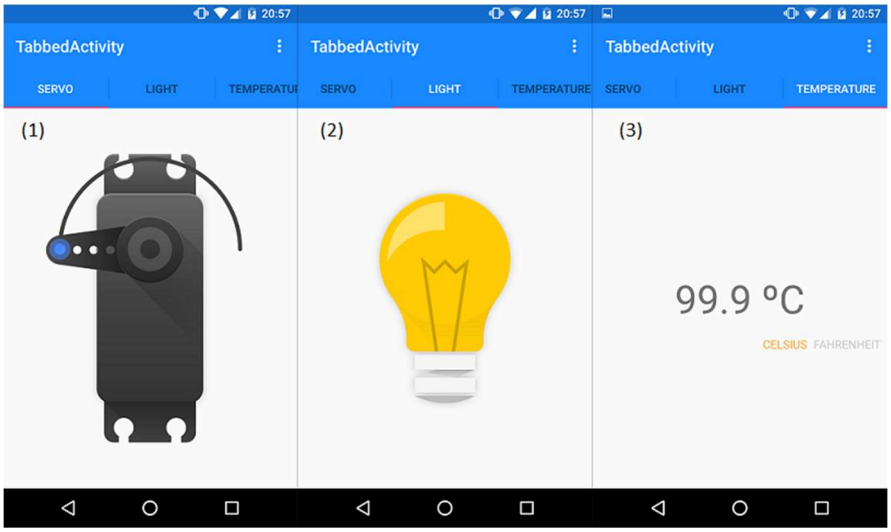

Figura 11 - Interface do aplicativo
Fonte: Autoria própria

Nos apêndices podem ser encontrados os códigos desenvolvidos para ambas interfaces de usuário. Os três arquivos da página *web* não necessitam de configurações para sua reprodução, porém necessitam de alteração caso deseje implementar essa mesma página em um número maior de navegadores, já o código para android pode ser reproduzido sem alterações após a correta configuração do *software* Android *Studio* IDE, porém, essa configuração não foi abordada nesse trabalho.

5 RESULTADOS E DISCUSSÕES

O protótipo funcionou de forma adequada, cumprindo o objetivo proposto de desenvolver um sistema de AR com interfaces de controle via página web e aplicativo android, com equipamentos de baixo custo. Alguns obstáculos foram encontrados durante a construção do protótipo, sendo necessária a adição de um equipamento que estava fora do planejamento.

A ideia inicial do projeto, levando em conta apenas o acionamento da lâmpada, consistia no controle, unicamente, através do comando via Arduino, porem em alguns casos em que o acesso ao servidor ficou indisponível, como quando aconteciam problemas de comunicação ou mesmo a falta de bateria no *smartphone*, mostrou-se necessário outro método de controle. Para solucionar essa questão foi utilizado um interruptor em paralelo com o acionamento via sistema. Isso gerou um outro problema, quando a lâmpada era acesa pelo interruptor o sistema não tinha essa informação e a visualização do usuário não correspondia a realidade, a solução encontrada foi adicionar um sensor de corrente na linha de alimentação da lâmpada, dessa forma, independente de como foi realizado o acionamento, a informação mostrada ao usuário correspondia a realidade.

O acesso através de página web inicialmente mostrou-se ineficaz devido ao tempo de leitura do cartão microSD pelo Shield, pois era necessário carregar a página e figuras todas do Arduino. Para solucionar esse problema foram adotadas duas soluções: para acessos realizados através de dispositivos conectados à internet, as imagens passaram a ser carregadas em servidor online, para quando esse tipo de acesso não fosse possível, as imagens foram simplificadas, diminuindo o tamanho, otimizando o processo pois seria necessário menos tempo de leitura do Arduino.

Ainda no acesso via navegador, outra questão encontrada foi em relação a medição de temperatura, que era informada em graus Fahrenheit, ou Celsius. O padrão da página é a exibição em graus Fahrenheit, porém a graduação por Celsius é a mais usual no Brasil, o que traria incomodo ao usuário que teria que alternar o modo exibição a cada acesso, inclusive na atualização da página. A solução encontrada foi a utilização de cookies do navegador, que salvam uma informação específica, nesse exemplo a preferência do usuário. A cada acesso os navegadores

atuais verificam automaticamente de já existe um cookie criado para aquela página e informam da preferência automaticamente.

O sistema de comunicação funcionou de forma eficaz e o de supervisão e controle desenvolvido mostrou-se de fácil uso e intuitivo, tornando a interação entre o usuário e o sistema uma experiência agradável.

6 CONCLUSÃO

Atualmente, a AR tem sido considerada necessária como forma de facilitar e melhorar a comodidade de seus usuários. Empresas brasileiras têm investido neste segmento, embora, o país passe por um momento econômico desfavorável, esse tema tem-se revelado como uma linha de pesquisa em constante evolução.

O desenvolvimento de um sistema de AR com equipamentos de baixo custo se mostrou viável e eficiente. Com componentes de fácil acesso e manipulação relativamente simples, o protótipo funcionou satisfatoriamente para o controle do servo motor, o acionamento da lâmpada e para medição de temperatura. Mesmo se tratando de um sistema simples, que atuou de forma isolada no controle dos equipamentos, o método utilizado se mostrou como uma boa base para implementação de sistemas mais complexos e abrangentes.

Páginas web e aplicativos para android são tecnologias já inseridas no cotidiano das pessoas, por isso, as interfaces desenvolvidas com esses recursos se mostraram práticas e intuitivas, sendo ferramentas eficientes para o controle remoto das residências.

O trabalho atingiu seu objetivo, podendo servir de embasamento para estudos semelhantes e, com a execução de um fechamento para proteção dos componentes, ser utilizado como um sistema simplificado de AR.

Sugere-se para próximos estudos: a incorporação de comunicação via *bluetooth*, a restrição de acesso via *login* e senha, monitoramento residencial por câmeras acessíveis via internet ou até mesmo o controle de mais dispositivos como fechadura eletrônica, portão e ar condicionado.

É razoável pensar que num futuro próximo, a exemplo da tecnologia da telefonia móvel, as residências serão cada dia mais adaptadas com recursos tecnológicos, onde a automação terá maior relevância.

REFERÊNCIAS

ARDUINO. **What is Arduino?**. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 27 set 2015.

BOLZANI, Caio Augustus Morais. **Análise de Arquiteturas e Desenvolvimento de uma Plataforma para Residências Inteligentes**. 155 f. Dissertação (Doutorado em Sistemas Eletrônicos) - Escola Politécnica da Universidade de São Paulo, Brasil. 2010.

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. **Algoritmos:** teoria e prática. 3ed. Elsevier, 2012.

DEITEL, Paul J.; DEITEL, Harvey. **C: como programar**. 6ed. Pearson. 2013a.

_____. **Java: como programar**. 8ed. Pearson. 2013b.

DIAS, César Luis de Azevedo; PIZZOLATO, Nelio Domingues. **Domótica**: Aplicabilidade e Sistemas de Automação Industrial. Vértices (Campos dos Goitacazes), v. 6, p. 9-32, 2004.

FRANCISCO, Antônio. M. S. **Autômatos Programáveis**. 4ed, 2007.

MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueredo de. **Algoritmos**: lógica para desenvolvimento. 26ed. Érica, 2012.

MICROSOFT. **Definição das sete camadas do modelo OSI e explicação de suas funções**, 2013. Disponível em: <https://support.microsoft.com/pt-br/kb/103884> Acesso em: 20 out 2015.

MURATORI, José Roberto; DAL BÓ, Paulo Henrique. **Automação Residencial**: Conceitos e Aplicações. 1ed. EDUCERE. 2013.

O'HARA, Bob; PETRICK, Al. **IEEE 802.11 Handbook** – A Designer's Companion. 2ed. Standards Information Network IEEE Press, 2005.

ROBBINS, Jennifer Niederst. **Aprendendo Web Design**: guia para iniciantes. 3ed. Bookman, 2010.

SOARES, Luiz Fernando G. **Redes Locais**. 1ed. Campus, 1986.

SPURGEON, Charles E.; ZIMMERMAN, Joann. **Ethernet**: The Definitive Guide. 2ed. O'Reilly, 2014.

TANENBAUM, Andrew S. **Redes de Computadores**. 11ed Campus, 2013.

TORRES, Gabriel. **Redes de Computadores**. 2ed. Novaterra, 2014.

WORLD WIDE WEB CONSORTIUM. **About W3C**. Disponível em: <http://www.w3.org/Consortium/>. Acesso em: 20 out 2015.

WEB DEVELOPERS NOTES. **What is web server** - a computer of a program? Disponível em: <http://www.webdevelopersnotes.com/basics/what_is_web_server.php>. Acesso em: 20 out 2015.

WI-FI ALLIANCE. **Who we are**. Disponível em: <http://www.wi-fi.org/who-we-are> Acesso em 26 out 2015.

APÊNDICE A – CÓDIGO DESENVOLVIDO PARA O ARDUINO

```
/**
 * Project  : Casasoo
 * File     : main.cpp
 * Version  : 2.1.1
 * Date     : 2015.10.04
 * Author   : leandrobpedro@gmail.com
 */

#include <Ethernet.h>
#include <math.h>
#include <Servo.h>
#include <SD.h>
#include <SPI.h>

#define ACS712  A2  // Current sensor.
#define LM35    A3  // Temperature sensor.

// Quiescent output voltage.
int quiescentVoltage;

// Servo
Servo servo;

// Ethernet configuration.
byte mac[] =
    {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; // Mac address.
IPAddress ip(10, 0, 0, 210);              // Ip address.
IPAddress mask(255, 255, 255, 0);         // Subnet mask.
IPAddress gateway(10, 0, 0, 254);         // Gateway address.
IPAddress dnsserver(10, 0, 0, 1);         // Dns server ip.

// Create a server that listens for incoming connections.
EthernetServer server(80);

float get_temp(int pin) {
  float temp = 0;
  int sample = 5;

  for (int i = 0; i < sample; i++) {
    temp += analogRead(pin);
    delay(1);
  }

  return (float) temp / sample;
}

int get_quiescent_voltage(int pin) {
  int sample = 1000;
  long quiescentVoltage = 0;

  // Read samples to stabilise value.
  for (int i = 0; i < sample; i++) {
    quiescentVoltage += abs(analogRead(pin));
    delay(1);
  }

  quiescentVoltage /= sample;
```

```
  return (int) quiescentVoltage;
}

boolean read_current(int pin) {
  float sensor = 0, aux = 0;
  int sample = 5, sensibility = 5;

  // Read samples to stabilise value.
  for (int i = 0; i < sample; i++) {
    aux = abs(analogRead(pin) - quiescentVoltage);
    if (aux > sensibility) {
      sensor += aux;
    }
    delay(1);
  }

  return (sensor / sample) > 1 ? true : false;
}

boolean xhr(EthernetClient client, const String request) {
  String cmd =
      request.substring(request.indexOf("?cmd=") + 5,
request.indexOf("&"));
  String value =
      request.substring(request.indexOf("&value=") + 7,
request.indexOf("\n"));

  if (cmd.equals("toggle")) {
    pinMode(value.toInt(), OUTPUT);
    digitalWrite(value.toInt(), !digitalRead(value.toInt()));
  } else if (cmd.equals("servo")) {
    servo.write(value.toInt());
  } else if (cmd.equals("refresh")) {
    client.println();                    // Blank line.
    client.print("?servo=");             // Servo.
    client.print(servo.read());          // Servo current position.
    client.print("&light=");             // Light.
    client.print(read_current(ACS712)); // Light state.
    client.print("&temp=");              // Temperature.
    client.print(get_temp(LM35));        // Temperature value.
  } else {
    return false;
  }

  return true;
}

void listen_ethernet_clients () {
  EthernetClient client = server.available(); // Client connected.
  const int bufsiz = 99;
  char request[bufsiz], c;
  int index = 0;
  int isXhRequest = 0;

  if (client) {
    if (client.connected()) {
      // Get client request.
      while (client.available()) {
        c = client.read();
```

```
        if (c != '\n' && c != '\r') {
          request[index] = c;
          index++;
        } else {
          break;
        }
      }
    }

    request[index] = NULL;

    // Format client request.
    if (strstr(request, "GET /") != 0) {
      (strstr(request, " HTTP"))[0] = 0;

      // Remove "GET /" (5 chars)
      if (strstr(request, "GET /")) {
        memmove(request, request + 5, 1 + strlen(request) + 5);
      }

      if (!strlen(request)) {
        // TODO: check if request is empty
         strcpy(request, "index.htm");
      }
    }

//    Serial.println(request);

    // Send a standard http response header.
    client.println("HTTP/1.1 200 OK");

    // MIME Types - Multipurpose Internet Mail Extensions.
    if (strstr(request, ".htm")) {                  // .htm
      client.println("Content-Type: text/html");
    } else if (strstr(request, ".css")) {           // .css
      client.println("Content-Type: text/css");
    } else if (strstr(request, ".js")) {            // .js
      client.println("Content-Type: application/javascript");
    } else if (strstr(request, ".png")) {           // .png
      client.println("Content-Type: image/png");
    } else if (strstr(request, ".jp")) {            // .jpg, .jpeg
      client.println("Content-Type: image/jpeg");
    } else if (strstr(request, "?cmd")) {           // Ajax request.
      client.println("Content-Type: text");
      isXhRequest = xhr(client, request);
    } else {
      client.println("Content-Type: text");
    }

    // If it's a file.
    if (!isXhRequest) {

      client.println();

      // Open file.
      File webfile = SD.open(request);

      if (webfile) {
        while(webfile.available()) {
          client.write(webfile.read());
```

```
        }
        webfile.close();
      }
    }

    // Give some time to browser receive data.
    delay(1);
    client.stop();
  }
}

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

  // For external power supply.
  // The voltage applied to the AREF pin (0 to 5V) is used as the
reference.
  analogReference(EXTERNAL);

  // Set pins as output.
  pinMode(ACS712, INPUT);
  pinMode(LM35, INPUT);

  // Determine quiescent output voltage.
  quiescentVoltage = get_quiescent_voltage(ACS712);

  Serial.print("Quiescent voltage: ");
  Serial.println(quiescentVoltage);

  // Initialize the ethernet device.
  // Ethernet shield attached to pins 10, 11, 12, 13.
  Ethernet.begin(mac, ip, dnsserver, gateway, mask);

  // Initialize server.
  server.begin();

  // Servo motor.
  servo.attach(9);  // Pin.
  servo.write(90);  // Initial position.

  // initialize SD card.
  Serial.println("Initializing SD card...");
  if (!SD.begin(4)) {  // cs pin at 4
    Serial.println("ERROR - SD card initialization failed!");
    return;
  } else {
    Serial.println("SUCCESS - SD card initialized.");
  }

  // Check for index.htm file.
  char *c = new char[12];
  strcpy(c, "index.htm");

  if (!SD.exists(c)) {
    // Can't find index file.
    Serial.println("ERROR - Can't find index.htm file!");
    return;
  } else {
    Serial.println("SUCCESS - Found index.htm file.");
  }
```

```
  // Print out the IP address.
  Serial.print("IP = ");
  Serial.println(Ethernet.localIP());
}

void loop () {
  // Listen for incoming Ethernet connections.
  listen_ethernet_clients();
}
```

APÊNDICE B – CÓDIGO ANDROID – MAINACTIVITY.JAVA

```
package vc.zz.casasoo.casasoo;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;

public class MainActivity extends Activity {
  protected static final String TAG = MainActivity.class.getSimpleName();
  public static String address = "http://10.0.0.210";

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button button = (Button) findViewById(R.id.button);
    SeekBar seekbar = (SeekBar) findViewById(R.id.seekbar);

    button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        // new RequestTask(MainActivity.this).execute(address +
"?cmd=toggle&value=7");
        Intent intent = new Intent(MainActivity.this,
TabbedActivity.class);
        startActivity(intent);
      }
    });

    seekbar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
      @Override
      public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
        // Log.d(TAG, String.valueOf(progress));
        new RequestTask(MainActivity.this)
            .execute(address + "?cmd=servo&value=" +
String.valueOf(progress));
      }

      @Override
      public void onStartTrackingTouch(SeekBar seekBar) {

      }

      @Override
      public void onStopTrackingTouch(SeekBar seekBar) {

      }
    });
  }

  @Override
```

```
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is
present.
    getMenuInflater().inflate(R.menu.menu main, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
      return true;
    }

    return super.onOptionsItemSelected(item);
  }
}
```

APÊNDICE C – CÓDIGO ANDROID – REQUESTTASK.JAVA

```
package vc.zz.casasoo.casasoo;

import android.app.Activity;
import android.content.res.Resources;
import android.graphics.ColorMatrix;
import android.graphics.ColorMatrixColorFilter;
import android.net.Uri;
import android.os.AsyncTask;
import android.util.DisplayMetrics;
import android.util.Log;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.text.NumberFormat;

public class RequestTask extends AsyncTask<String, Void, String> {
  protected static final String TAG = RequestTask.class.getSimpleName();

  // User for update resources in another activities.
  Activity mActivity;

  public RequestTask() {
    // Required empty public constructor.
  }

  public RequestTask(Activity activity) {
    mActivity = activity;
  }

  @Override
  protected String doInBackground(String... urls) {
    // Log.d("MainActivity", urls[0]);
    // params comes from the execute() call: params[0] is the url.
    String response = null;

    try {
      URL url = new URL(urls[0]);
      URLConnection urlConnection = url.openConnection();
      String line;

      urlConnection.setRequestProperty("Accept-Charset", "UTF-8");

      // working
      BufferedReader in =
          new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));

      while ((line = in.readLine()) != null) {
        if (!line.equals("")) {
          response = line;
        }
      }

      return response;
    } catch (IOException e) {
```

```
      return "Unable to retrieve web page. URL may be invalid or server is
down.";
    }
  }

  @Override
  protected void onPostExecute(String result) {
    // Update user interface.
    updateUi(result);
  }

  // Update user interface.
  public void updateUi(String response) {

    float dpi = Resources.getSystem().getDisplayMetrics().density;

    final float offset = (29 * dpi);

    // Return if arduino is disconnected.
    if (response == null || response.isEmpty()) return;

    // Used for set png to grey scale.
    ColorMatrix cm = new ColorMatrix();
    cm.setSaturation(0);
    final ColorMatrixColorFilter f = new ColorMatrixColorFilter(cm);

    // Creates a Uri which parses the given encoded URI string.
    Uri uri = Uri.parse(response);

    // Searches the query string for the first value with the given key.
    String servo = uri.getQueryParameter("servo");
    String light = uri.getQueryParameter("light");
    String temp = uri.getQueryParameter("temp");

    if (servo != null && !servo.isEmpty()) {
      ServoFragment.axis.setPivotX(ServoFragment.axis.getWidth() + offset);
      ServoFragment.axis.setRotation(Float.valueOf(servo));
    }

    if (light != null && !light.isEmpty()) {
      if (Integer.valueOf(light) > 0) {
        // Colorful.
        LightFragment.light.setColorFilter(null);
      } else {
        // Grey scale.
        LightFragment.light.setColorFilter(f);
      }
    }

    if (temp != null && !temp.isEmpty()) {
      float result = Float.valueOf(temp);
      int id;
      String unit = null;
      NumberFormat nf = NumberFormat.getNumberInstance();

      nf.setMaximumFractionDigits(1);
      nf.setMinimumFractionDigits(1);

      // Celsius or Fahrenheit.
      id = TemperatureFragment.radiogroup.getCheckedRadioButtonId();
```

```
      switch (id) {
        case R.id.radio_celsius:
          result = (float) (result * (0.4887585532746823));
          unit = (char) 0x00B0 + "C";
          break;
        case R.id.radio_fahrenheit:
          result = (float) (result * (0.4887585532746823) * 1.8 + 32);
          unit = (char) 0x00B0 + "F";
          break;
      }

      // Update text view.
      TemperatureFragment.texttemp.setText(nf.format(result) + " " + unit);
    }
  }
}
```

APÊNDICE D – CÓDIGO ANDROID – SERVOFRAGMENT.JAVA

```
package vc.zz.casasoo.casasoo;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

public class ServoFragment extends Fragment {
  protected static final String TAG = ServoFragment.class.getSimpleName();
  protected static ImageView axis;
  private static final String ARG_SECTION_NUMBER = "section_number";

  public static ServoFragment newInstance(int sectionNumber) {
    ServoFragment fragment = new ServoFragment();
    Bundle args = new Bundle();
    args.putInt(ARG_SECTION_NUMBER, sectionNumber);
    fragment.setArguments(args);
    return fragment;
  }

  public ServoFragment() {
  }

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {

    final View rootView = inflater.inflate(R.layout.fragment_servo,
container, false);
    axis = (ImageView) rootView.findViewById(R.id.axis);


    CircularSeekBar circularSeekBar =
        (CircularSeekBar) rootView.findViewById(R.id.circular_seekbar);
    circularSeekBar.setMax(180);

    circularSeekBar.setOnSeekBarChangeListener(new
CircularSeekBar.OnCircularSeekBarChangeListener() {
      @Override
      public void onProgressChanged(
          CircularSeekBar circularSeekBar, int progress, boolean fromUser)
{
        axis.setPivotX(ServoFragment.axis.getWidth() +
            getResources().getDisplayMetrics().density * 29);
        axis.setRotation(progress);
      }

      @Override
      public void onStopTrackingTouch(CircularSeekBar seekBar) {

TabbedActivity.handler.postDelayed(TabbedActivity.updateTimerThread, 100);
        new RequestTask(getActivity())
            .execute(MainActivity.address + "?cmd=servo&value=" +
                String.valueOf(seekBar.getProgress()));
      }
```

```
        @Override
        public void onStartTrackingTouch(CircularSeekBar seekBar) {

TabbedActivity.handler.removeCallbacks(TabbedActivity.updateTimerThread);
        }
      });
      return rootView;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
    }
}
```

APÊNDICE E – CÓDIGO ANDROID – TABBEDACTIVITY.JAVA

```
package vc.zz.casasoo.casasoo;

import android.app.ActionBar;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.Context;
import android.os.Bundle;
import android.support.v13.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;

import java.util.Locale;


public class TabbedActivity extends Activity implements
ActionBar.TabListener {
  protected static final String TAG = TabbedActivity.class.getSimpleName();
  // protected static float DPI;

  protected static android.os.Handler handler = new android.os.Handler();
  protected static Runnable updateTimerThread = new Runnable() {
    public void run() {
      new RequestTask().execute(MainActivity.address + "?cmd=refresh");
      handler.postDelayed(this, 2000);
    }

  };

  SectionsPagerAdapter mSectionsPagerAdapter;

  /**
   * The {@link ViewPager} that will host the section contents.
   */
  ViewPager mViewPager;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_tabbed);

    // Set up the action bar.
    final ActionBar actionBar = getActionBar();
    actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

    // Create the adapter that will return a fragment for each of the three
    // primary sections of the activity.
    mSectionsPagerAdapter = new SectionsPagerAdapter(getFragmentManager());

    // Set up the ViewPager with the sections adapter.
    mViewPager = (ViewPager) findViewById(R.id.pager);
    // Set the number of pages that should be retained to either side of
```

```
    // the current page in the view hierarchy in an idle state.
    mViewPager.setOffscreenPageLimit(mSectionsPagerAdapter.getCount());
    mViewPager.setAdapter(mSectionsPagerAdapter);

    // When swiping between different sections, select the corresponding
    // tab. We can also use ActionBar.Tab#select() to do this if we have
    // a reference to the Tab.
    mViewPager.setOnPageChangeListener(new
ViewPager.SimpleOnPageChangeListener() {
      @Override
      public void onPageSelected(int position) {
        actionBar.setSelectedNavigationItem(position);
      }
    });

    // For each of the sections in the app, add a tab to the action bar.
    for (int i = 0; i < mSectionsPagerAdapter.getCount(); i++) {
      // Create a tab with text corresponding to the page title defined by
      // the adapter. Also specify this Activity object, which implements
      // the TabListener interface, as the callback (listener) for when
      // this tab is selected.
      actionBar.addTab(
          actionBar.newTab()
              .setText(mSectionsPagerAdapter.getPageTitle(i))
              .setTabListener(this));
    }
  }

  @Override
  protected void onResume() {
    super.onResume();
    handler.postDelayed(updateTimerThread, 1000);
  }

  @Override
  protected void onPause() {
    super.onPause();
    handler.removeCallbacks(updateTimerThread);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is
present.
    getMenuInflater().inflate(R.menu.menu_tabbed, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
      return true;
    }
```

```
    return super.onOptionsItemSelected(item);
  }

  @Override
  public void onTabSelected(ActionBar.Tab tab, FragmentTransaction
fragmentTransaction) {
    // When the given tab is selected, switch to the corresponding page in
    // the ViewPager.
    mViewPager.setCurrentItem(tab.getPosition());
  }

  @Override
  public void onTabUnselected(ActionBar.Tab tab, FragmentTransaction
fragmentTransaction) {
  }

  @Override
  public void onTabReselected(ActionBar.Tab tab, FragmentTransaction
fragmentTransaction) {
  }


  /**
   * A {@link FragmentPagerAdapter} that returns a fragment corresponding
to
   * one of the sections/tabs/pages.
   */
  public class SectionsPagerAdapter extends FragmentPagerAdapter {

    public SectionsPagerAdapter(FragmentManager fm) {
      super(fm);
    }

    @Override
    public Fragment getItem(int position) {
      // getItem is called to instantiate the fragment for the given page.
      // Return a PlaceholderFragment (defined as a static inner class
below).

      // Log.d("TabbedActivity", String.valueOf(position));

      switch (position) {
        case 0:
          return ServoFragment.newInstance(position);
        case 1:
          return LightFragment.newInstance(position);
        case 2:
          return TemperatureFragment.newInstance(position);
        default:
          return null;
      }
    }

    @Override
    public int getCount() {
      // Show 3 total pages.
      return 3;
    }

    @Override
```

```
    public CharSequence getPageTitle(int position) {
      Locale l = Locale.getDefault();
      switch (position) {
        case 0:
          return getString(R.string.servo).toUpperCase(l);
        case 1:
          return getString(R.string.light).toUpperCase(l);
        case 2:
          return getString(R.string.temperature).toUpperCase();
      }
      return null;
    }
  }

}
```

APÊNDICE F – CÓDIGO ANDROID TEMPERATUREFRAGMENT.JAVA

```
package vc.zz.casasoo.casasoo;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RadioGroup;
import android.widget.TextView;

public class TemperatureFragment extends Fragment {
  protected static final String TAG =
TemperatureFragment.class.getSimpleName();
  protected static RadioGroup radiogroup;
  protected static TextView texttemp;
  /**
   * The fragment argument representing the section number for this
   * fragment.
   */
  private static final String ARG_SECTION_NUMBER = "section_number";

  /**
   * Returns a new instance of this fragment for the given section
   * number.
   */
  public static TemperatureFragment newInstance(int sectionNumber) {
    TemperatureFragment fragment = new TemperatureFragment();
    Bundle args = new Bundle();
    args.putInt(ARG_SECTION_NUMBER, sectionNumber);
    fragment.setArguments(args);
    return fragment;
  }

  public TemperatureFragment() {}

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
    final View rootView = inflater.inflate(R.layout.fragment_temperature,
container, false);

    radiogroup = (RadioGroup) rootView.findViewById(R.id.radio_group);
    texttemp = (TextView) rootView.findViewById(R.id.text_temperature);

    return rootView;
  }

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
  }
}
```

APÊNDICE G – CÓDIGO ANDROID – LIGHTFRAGMENT.JAVA

```
package vc.zz.casasoo.casasoo;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

public class LightFragment extends Fragment {

  private static final String ARG_SECTION_NUMBER = "section_number";

  protected static ImageView light;

  /**
   * Use this factory method to create a new instance of
   * this fragment using the provided parameters.
   * @param sectionNumber section number.
   * @return A new instance of fragment LightFragment.
   */

  public static LightFragment newInstance(int sectionNumber) {
    LightFragment fragment = new LightFragment();
    Bundle args = new Bundle();
    args.putInt(ARG_SECTION_NUMBER, sectionNumber);
    fragment.setArguments(args);
    return fragment;
  }

  public LightFragment() {
    // Required empty public constructor.
  }

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
  }

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
    // Inflate the layout for this fragment.
    View rootView = inflater.inflate(R.layout.fragment_light, container,
false);

    light = (ImageView) rootView.findViewById(R.id.light);

    light.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        new RequestTask(getActivity())
            .execute(MainActivity.address + "?cmd=toggle&value=7");
      }
    });

    return rootView;
  }
}
```

APÊNDICE H – CÓDIGO PAGINA *WEB* – INDEX.HTM

```
<!DOCTYPE html>
<html lang="en">
<!-- head -->
<head>
  <title>Casasoo</title>
  <link rel="shortcut icon" type="image/ico" href="favicon.ico"/>
  <link href="styles.css" rel="stylesheet" type="text/css">
  <link
href="http://fonts.googleapis.com/css?family=Roboto:400,500,300,100,700,900
" rel="stylesheet" type="text/css">
  <script type="text/javascript" language="javascript"
src="script.js"></script>
</head>
<!-- /head -->

<!-- body -->
<body>
  <!-- header -->
  <div id="header">
    <div class="pull-left">
      <a class="header-logo" href="#">
        <span>casas<strong>oo</strong></span>
      </a>
    </div>
  </div>
  <!-- /header -->

  <!--
  <div style="position: relative; top: 123px; z-index: 3;height:
128px;width: 72px;">
    <input id="response"
value="?servo=90&amp;lamp=0&amp;sensor=27.5345&amp;">
  </div>
  <button id="button_foo">Button foo</button>
  -->

  <!-- content -->
  <div id="content">
    <!-- servo -->
    <div class="card">
      <div class="image" style="background-color: #8cc4ff;">
        <img src="http://casasoo.zz.vc/base.png"
onerror="this.src='base.png'">
        <img id="axis" src="http://casasoo.zz.vc/axis.png"
onerror="this.src='axis.png'">
        <span class="title">Servo</span>
      </div>
      <!-- <div class="content"></div> -->
      <div class="action">
        <input id="range" step="10" type="range" min="0" max="180">
      </div>
    </div>
    <!-- /servo -->
    <!-- light -->
    <div class="card">
      <div class="image" style="background-color: #ad8cff;">
```

```
        <img id="light" src="http://casasoo.zz.vc/light.png"
onerror="this.src='light.png'" style="-webkit-filter: none;">
        <span class="title">Bulb light</span>
      </div>
      <!-- <div class="content"></div> -->
      <div class="action">
        <a id="toggle" href="javascript:void(0)">Toggle light</a>
      </div>
    </div>
    <!-- /light -->
    <!-- temperature -->
    <div class="card">
      <div class="image" style="background-color: #00bbd4;">
        <h1 id="temp" class="data">00.0&deg;F</h1>
        <span class="title">Temperature</span>
      </div>
      <!-- <div class="content"></div> -->
      <div class="action">
        <a id="celsius" href="javascript:void(0)">Celsius</a>
        <a id="fahrenheit" href="javascript:void(0)"
onclick="">Fahrenheit</a>
      </div>
    </div>
    <!-- /temperature -->
  </div>
  <!-- /content -->
  <!-- footer -->
  <!-- <div id="footer"></div> -->
  <!-- /footer -->
</body>
<!-- /body -->
</html>
```

APÊNDICE I – CÓDIGO PAGINA *WEB* – STYLES.CSS

```
/**
 * Project  : Casasoo
 * File     : styles.css
 * Version  : 1.0.0
 * Date     : 2015-07-13
 * Author   : leandrobpedro@gmail.com
 */

body {
  background: #ebeff2;
  display: flex;
  font-family: 'Roboto', monospace, sans-serif;
  justify-content: center;
  padding: 24px;
  overflow-y: auto;
}

h1, h2, h3 {
  color: black;
  opacity: .87;
}

p {
  color: black;
  margin-left: 4;
  opacity: .57;
}

#header {
  background-color: #1988ff;
  border: 1px solid #1988ff;
  box-shadow: 0 2px 6px rgba(0,0,0,0.2);
  color: rgba(255,255,255,.87);
  height: 96px;
  left: 0px;
  padding-left: 48px;
  position: fixed;
  top: 0px;
  width: 100%;
  z-index: 2;
}

#content {
  background: #ebeff2;
  height: auto;
  margin: 96px 0;
  padding: 24px;
  width: auto;
}

#footer {
  background-color: #303030;
  border: 1px solid #2b2b2b;
  color: rgba(255,255,255,.87);
  height: 56px;
  left: 0;
  margin-top: -56px;
  position: absolute;
```

```
  width: 100%;
}

/* Servo motor axis */
#axis {
  left: 64px;
  margin: inherit;
  position: absolute;
  top: 104px;
  transform-origin: 131% 50% 0;
  width: 98px;
}

/* Casaoo logo */
.pull-left {
  float: left;
  position: relative;
  top: 50%;
  transform: translateY(-50%);
}

.header-logo span {
  color: #fff;
  font-weight: 300;
  position: relative;
  top: -2px;
}

.header-logo a, a:hover, a:focus, a:active, a:visited, a:link {
  text-decoration: none;
}

.header-logo span strong {
  color: #ff9b06;
  font-weight: 300;
}

.header-logo {
  background: url(http://casasoo.zz.vc/logo.png) no-repeat 0 center,
              url(logo.png) no-repeat 0 center; /* On error. */
  font-size: 24px;
  letter-spacing: 1px;
  opacity: 1;
  padding-left: 48px;
  position: relative;
  top: 1px;
}

/* Inputs. */
input[type="range"]{
  background: #8cc4ff no-repeat;
  height:2px;
  -webkit-appearance: none;
  width: 100%;
}

input[type="range"]::-webkit-slider-thumb{
  background: #3b9aff;
  border-radius: 100%;
  box-shadow: 0 1px 2px rgba(43,59,93,0.29);
```

```
  cursor: pointer;
  height: 20px;
  position:relative;
  -webkit-appearance:none;
  width: 20px;
  z-index: 1;
}

input[type=range], [type=number]:focus {
    outline: none;
}

input[type="number"] {
  border: none;
  color: rgba(0,0,0,0.87);
  font-family: Roboto;
  font-size: 2em;
  text-align: center;
  -webkit-appearance: none;
  width: 100%;
}

input[type=number]::-webkit-inner-spin-button,
input[type=number]::-webkit-outer-spin-button {
  appearance: none;
  margin: 0;
  -moz-appearance: none;
  -webkit-appearance: none;
}

/* Card. */
.card {
  background: #fff;
  border-radius: 3px;
  box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.16), 0 2px 10px 0 rgba(0, 0, 0,
0.12);
  color: #272727;
  display: inline-block;
  margin: 0;
  overflow: hidden;
  position: relative;
  vertical-align: top;
  width: 384px;
}

.card .data {
  color: rgba(255, 255, 255, .87);
  font-size: 3em;
  left: 50%;
  margin: 0;
  position: absolute;
  top: 50%;
  transform: translateX(-50%) translateY(-50%);
}

.card .title {
  font-size: 2em;
  font-weight: 300;
  line-height: 48px;
}
```

```
.card .content {
  border-radius: 0 0 2px 2px;
  font-weight: 300;
  min-height: 80px;
  padding: 20px;
}

.card p {
  margin: 0;
}

.card .action {
  border-top: 1px solid rgba(160, 160, 160, 0.2);
  padding: 20px;
}

.card a {
  color: #ffab40;
  margin-right: 20px;
  text-decoration: none;
  text-transform: uppercase;
  transition: color 0.3s ease;
  -webkit-transition: color 0.3s ease;
}

.card .image {
  height: 384px;
  position: relative;
}

.card .image img {
  bottom: 0;
  left: 0;
  margin: auto;
  position: absolute;
  right: 0;
  top: 0;
}

.card .image .title {
  bottom: 0;
  color: #fff;
  left: 0;
  padding: 20px;
  position: absolute;
}
```

APÊNDICE J – CÓDIGO PAGINA *WEB* – SCRIPT.JS

```
/**
 * Project  : Casasoo
 * File     : script.js
 * Version  : 1.0.0
 * Date     : 2015-07-13
 * Author   : leandrobpedro@gmail.com
 */

window.onload = function () {
  // Update user interface.
  setInterval(function () { sendRequest('?cmd=refresh', updateUi) }, 2000);

  // Servo slider.
  range.addEventListener('mouseup', function () {
    sendRequest('?cmd=servo&value=' + this.value);
  });

  // Toggle.
  toggle.addEventListener('click', function () {
    sendRequest('?cmd=toggle&value=7', null);
  });

  celsius.addEventListener('click', function () {
    setCookie('unit', 'celsius');
  });

  fahrenheit.addEventListener('click', function () {
    setCookie('unit', 'fahrenheit');
  });

  // For test porpouses.
  // this.button_foo.addEventListener('click', function () {});
};


function setCookie(cname, cvalue, exdays) {
  var d = new Date();
  d.setTime(d.getTime() + (exdays * 24 * 3600 *1000));
  var expires = "expires=" + d.toUTCString();
  document.cookie = cname + "=" + cvalue + "; " + expires;
}

function getCookie(cname) {
  var name = cname + "=";
  var ca = document.cookie.split(';');
  for(var i = 0; i < ca.length; i++) {
    var c = ca[i];
    while (c.charAt(0)==' ') c = c.substring(1);
    if (c.indexOf(name) == 0) return c.substring(name.length, c.length);
  }
  return "";
}

function sendRequest(url, callback, postData) {
  var xhr = new XMLHttpRequest();
  if (!xhr) return;
  var method = (postData) ? "POST" : "GET";
  xhr.onreadystatechange = function() {
```

```
    if (xhr.readyState != 4) return;
    if (xhr.status != 200 && xhr.status != 304) return;
    if (callback) callback(xhr);
  };
  if (xhr.readyState == 4) return;
  xhr.open(method, url, true);
  xhr.timeout = 4000;
  xhr.ontimeout = function() {
    xhr.abort();
  };
  xhr.send(postData);
}

function getParameterByName(req, param) {
  param = param.replace(/[\[]/, "\\[").replace(/[\]]/, "\\]");
  var regex = new RegExp("[\\?&]" + param + "=([^&#]*)"),
      results = regex.exec(req);
  return results === null ? "" :
decodeURIComponent(results[1].replace(/\+/g, " "));
}

function updateUi(req) {
  var response = req.responseText;


  var servo = getParameterByName(response, 'servo');
  var light = parseInt(getParameterByName(response, 'light'));
  var temp = parseFloat(getParameterByName(response, 'temp'));

  var x = document.getElementById('axis');
  var y = document.getElementById('light');
  var z = document.getElementById('temp');

  if (getCookie('unit') != 'celsius') {
    document.getElementById('celsius').style.color = "#ccc";
    document.getElementById('fahrenheit').style.color = "#ffab40";
    z.innerHTML = parseFloat(temp * (5 / 1023 * 100) * 1.8 + 32).toFixed(1)
+ " \xB0F";
  } else {
    document.getElementById('celsius').style.color = "#ffab40";
    document.getElementById('fahrenheit').style.color = "#ccc";
    z.innerHTML = parseFloat(temp * (5 / 1023 * 100)).toFixed(1) + "
\xB0C";
  }

  x.style.transform = "rotate(" + servo + "deg)";
  light ? y.style["-webkit-filter"] = "none" : y.style["-webkit-filter"] =
"grayscale(1)";

}
```