ETEC CÔNEGO JOSÉ BENTO
Técnico em Redes de Computadores

Bryan Lima
Carlos Augusto dos Santos Neto
José Danrley da Silva

SISTEMA DE AUTOMAÇÃO RESIDENCIAL INTEGRADO A REDE

Jacareí, SP
2017

Bryan Lima
Carlos Augusto dos Santos Neto
José Danrley da Silva

SISTEMA DE AUTOMAÇÃO RESIDENCIAL INTEGRADA A REDE

Trabalho de Conclusão de Curso apresentado à Escola Técnica Estadual Cônego José Bento de Jacareí, como parte dos requisitos necessários para a obtenção do título de Técnico em Redes de Computadores.

Orientador: Prof. Me. Marcelo Guido

Jacareí, SP
2017

Bryan Lima
Carlos Augusto dos Santos Neto
José Danrley da Silva

SISTEMA DE AUTOMAÇÃO RESIDENCIAL INTEGRADA A REDE

Trabalho de Conclusão de Curso apresentado à Escola Técnica Estadual Cônego José Bento de Jacareí, como parte dos requisitos necessários para a obtenção do título de Técnico em Redes de Computadores.

Composição da Banca

_____

Nome do Componente da Banca, Titulação e Instituição

_____

Nome do Componente da Banca, Titulação e Instituição

_____

Nome do Orientador, Titulação e Instituição

_____/_____/_____

DATA DA APROVAÇÃO

AGRADECIMENTOS

Gostaríamos de agradecer ao **Grupo Feliza – Home e Automação**, ao **Sidney Zanni** da empresa **DHT – Design Home Theater**, e ao **Rennê Venceslau** da empresa **Inmótica**, que nos ajudaram através de contato por *e-mail*, fornecendo informações sobre o custo de mão de obra de um projeto desse ramo.

Também queremos agradecer ao professor **Francisco Silva** que ministra a disciplina “Instalação e Configuração de Redes Sem Fio”, por sempre se dispor a nos ajudar após suas aulas, para sanar as nossas inúmeras dúvidas, dar dicas valiosas para estruturar a documentação e a parte lógica deste projeto, e nos fazer entender que “o importante é fazer o simples funcionar.”

RESUMO

A tecnologia tem avançado consideravelmente durante os últimos anos, o que contribuiu para o surgimento do conceito de Iot (Internet das Coisas) que consiste em conectar à internet diversos dispositivos como geladeira, televisão, sensores, câmeras, e até carros, para um maior conforto e comodidade. À princípio, este conceito parece algo distante de estar presente nas residências atuais, mas é possível aplicar este conceito de modo prático, através da automação residencial. O presente projeto demonstra um protótipo funcional de uma automação residencial usando uma placa microcontroladora Arduino Uno, servo motor e LED's em uma maquete de residência feito em madeirite, acedendo e apagando luzes, abrindo e fechando o portão, e que pode ser gerenciado através de uma página HTML, ou pelo aplicativo OnLight, desenvolvido para este fim, de forma local ou através de acesso remoto.

Palavras-chave: Automação Residencial; IoT; Arduino Uno; Shield Ethernet; DDNS; App Inventor.

ABSTRACT

The technology has advanced considerably during the last years, which contributes to the emergence of the concept of Internet (Internet of Things) that consists of connecting to the internet various devices like refrigerators, television, sensors, cameras, and even cars, for greater comfort and Convenience. At first, this concept seems somehow different from being present in the current residences, but it is possible to apply the concept in a practical way through residential automation. The present project demonstrates a functional prototype of a residential automation using an Arduino Uno microcontroller, Servo Motor and LED's in a wooden model, accessing and erasing lights, opening and closing gates, and can be managed through an HTML Page, or by the OnLight app, developed for this purpose, either locally or through remote access.

Keywords: Residential Automation; IoT; Arduino Uno; Shield Ethernet; DDNS; App Inventor.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Atualmente, um dos temas de destaque em expansão na era da informação, (DEVAL, 2015) é o conceito de sistemas e processos autônomos. Estes utilizam a elaboração e desenvolvimento de uma lógica, composta por recursos tecnológicos, que possibilitam englobar diversos dispositivos em um único sistema, gerando uma funcionalidade dinâmica e autônoma.

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” (MURATORI e DAL BÓ, 2015, p. 70).

A ideia de sistema de automação residencial integrada a rede, tem sua origem primitiva na década de 70 (ROVERI, 2012). Logo em seguida, entre as décadas de 80 e 90, surgiu o conceito de "IoT (*Internet of Things*)" ou simplesmente "Internet das Coisas" (EVANS, 2011), quando o protocolo de transferência de dados aliado ao protocolo de internet, o TCP/IP, começou a se popularizar no cenário mundial.

Bill Joy, cofundador da Sun Microsystems, segundo (COELHO e CRUZ, 2017) “...pensou *Device to Device* (D2D), que introduziu o objetivo de colocar um endereço de protocolo de rede/internet ‘Internet Protocol’ em todos dispositivos de um determinado ambiente”.

O principal foco da automação residencial é buscar uma melhor gestão (economia) da energia elétrica, logo que uma das causas do desperdício de energia é justamente seu uso inapropriado, e principalmente, o mau gerenciamento dos dispositivos domésticos. Além disso, a falta de um padrão para os protocolos de automação faz com que, normalmente, se utilize linguagens privadas, fator que agrega alto valor ao serviço, tornando-o de custo elevado (SILVA, 2014).

A tecnologia presente em uma residência automatizada, que possui seus mecanismos baseados em sistemas eletrônicos pré-programados, é a Domótica, este termo surge a partir da união entre “Domus”, (ou seja, domicílio), e “robótica” (CECATO, 2010).

1.1 Objetivo Geral

Desenvolver sistema de automação residencial.

1.2 Objetivos Específicos

- Pesquisar tecnologias para automação residencial;
- Efetuar análise e triagem das tecnologias pesquisadas;
- Realizar montagem do protótipo para o desenvolvimento do projeto;
- Desenvolver programação do Arduino para o controle do projeto;
- Desenvolver aplicativo *mobile* para interface de controle;
- Implementar WLAN para conexão local, e configurá-la para conectividade WAN, afim de estabelecer conexão remota;
- Efetuar análises e testes do sistema implementado.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo se apresentará todos os conceitos necessários para o pleno entendimento do conteúdo apresentado nesta dissertação. Serão explicados de forma objetiva e concisa, os conhecimentos acerca do *hardware, softwares*, protocolos e serviços utilizados na construção do trabalho.

2.1 IoT

O *IoT* é uma sigla para o inglês *Internet of Things*, que traduzindo para o português significa Internet das Coisas, e pode ser definido como “uma extensão da Internet atual, que proporciona aos objetos do dia-a-dia (quaisquer que sejam), mas com capacidade computacional e de comunicação, se conectarem à Internet. ” (P. SANTOS, p. 2).

Como P. SANTOS também informa, atualmente não é somente os computadores que estão conectados à rede, como também uma gama de equipamentos como TV’s, videogames, *webcams*, *smartphones*, *notebooks*, automóveis, entre outros.

> Neste novo cenário, a pluralidade é crescente e previsões indicam que mais de 40 bilhões de dispositivos estarão conectados até 2020 [Forbes 2014]. Usando os recursos desses objetos será possível detectar seu contexto, controlá-lo, viabilizar troca de informações uns com os outros, acessar serviços da Internet e interagir com pessoas. Concomitantemente, uma gama de novas possibilidades de aplicações surgem (ex: cidades inteligentes (*Smart Cities*), saúde (*Healthcare*), casas inteligentes (*Smart Home*)) e desafios emergem (regulamentações, segurança, padronizações). (P. SANTOS, p. 3).

Para AVILA (2016), o *IoT* possui forte presença no cotidiano das pessoas, gerando praticidade nas mais simples rotinas diárias. CRUZ (2015) complementa com algumas exemplificações das mudanças mais recentes como “geladeira que avisa a hora de fazer novas compras, termostatos acionados via celular, lâmpadas programadas para acender ou apagar, e abertura e fechamento de fechaduras remotamente. ”

2.2 IDE Arduino

É responsável pela facilitação no desenvolvimento da criação dos códigos fontes adicionados à placa Arduino. A IDE (*Integrated Development Environment*, ou “Ambiente de Desenvolvimento Integrado”) é o programa que fica responsável pelo desenvolvimento da parte lógica da placa Arduino, ou seja, os códigos aplicados a ela.

A linguagem utilizada na IDE é baseada em C/C++, e uma vez que transferida para o Arduino, não necessita de sua presença em discos externos para seu funcionamento. O código só precisa ser compilado e armazenado uma vez no Arduino, pois este funciona como computador independente (JUSTEN).

2.3 Arduino

Caracteriza-se por uma plataforma para produção de protótipos de eletrônica. Um de seus objetivos de criação sempre foi a facilitação para criação de projetos em eletrônica que necessitem integrar vários dispositivos, além de dispensar conhecimentos aprofundados sobre eletrônica para o projetista. Estas características são muito importantes para explicar a fama que o Arduino possui hoje, tanto por profissionais experientes como por estudantes.

Com esta plataforma, os desenvolvedores de projetos possuem grande liberdade para integrar e controlar inúmeros dispositivos, como motores, LED's, sensores, dentre outros.

O Arduino Uno, se trata de um modelo de placa Arduino, comumente utilizado em projetos básicos. Possuindo pequenas dimensões e custando por volta de R$ 60,00 – R$80,00 (HACHOUCHE).

Figura 1. Arduino Uno

Fonte: (ELETROGATE)

2.4 Bibliotecas do Arduino

Biblioteca é uma coleção de subprogramas utilizados no desenvolvimento de sistema. Bibliotecas possuem código e dados auxiliares, que provém serviços a programas independentes, o que permite o compartilhamento e a alteração de código e dados de forma modular (TOPOL).

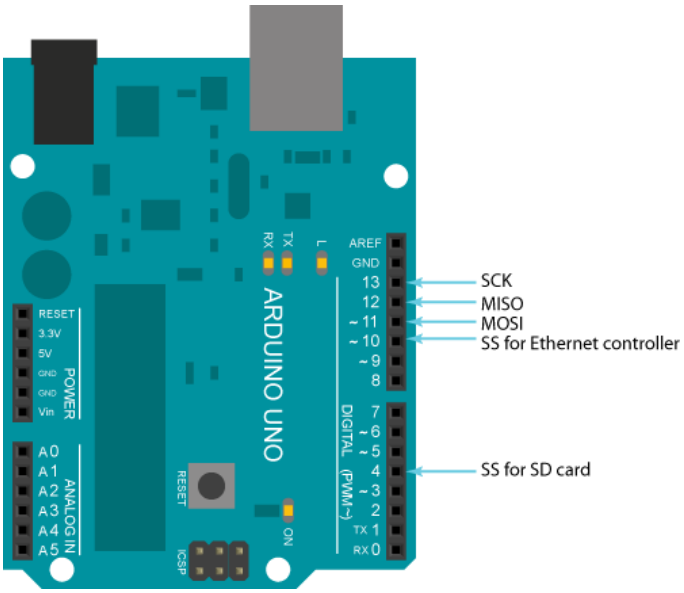
Utilizou-se a biblioteca SPI para a comunicação entre Shield Ethernet e Arduino, para o funcionamento do Shield Ethernet foi inserido a biblioteca Ethernet, e por fim, a biblioteca Servo para o controle do Servo Motor.

2.4.1 Biblioteca Ethernet

A biblioteca “ethernet.h” disponibiliza comandos para a facilitação da programação do dispositivo. A placa pode servir como um servidor que aceita conexões entrantes ou um cliente que faz consultas. A bibliotecas suporta até quatro ligações simultâneas (de entrada ou de saída, ou uma combinação). A biblioteca Ethernet gerencia o *chip* W5100.

Arduino se comunica com o escudo usando o barramento SPI. Isto está nos pinos digitais 11, 12, e 13 (layout de pinos referente ao modelo Uno).

Figura 2. Saídas da placa Arduino



Fonte: (DR. R)

2.4.2 Servo

.

A biblioteca “servo.h” permite que uma placa Arduino controle servo motores com facilidade, devido ao fato de seus comandos serem simples e objetivos. Servos têm engrenagens integradas e um eixo que pode ser controlado com precisão. Por

padrão permitem que o eixo seja posicionado em vários ângulos, geralmente entre 0 e 180 graus. Os servos de rotação contínua permitem que a rotação do eixo seja ajustada para várias velocidades (SERVO LIBRARY).

2.4.3 SPI

Para a comunicação entre o Arduino e o Ethernet Shield, foi utilizado a biblioteca (SPI), que auxilia o gerenciamento de controle sobre Interface Periférica Serial, que é um protocolo de dados em série síncrono usado por microcontroladores para comunicação rápida com um ou mais dispositivos periféricos em distâncias curtas. Também pode ser usado para comunicação entre dois microcontroladores. Com uma conexão SPI há sempre um dispositivo mestre (geralmente um microcontrolador) que controla os dispositivos periféricos.

2.5 LED

A tecnologia LED (*Light Emitting Diode*), que em português equivale a “o diodo emissor de luz”, se baseia em um semicondutor no estado sólido. É uma tecnologia SSL (*Solid State Light*), ou “Luz em Estado Sólido” (LOPES, 2014).

O LED é um *chip* que realiza a transformação de energia elétrica em luz. Este chip possui pequenas dimensões, cerca de 0,25 mm^2 (LOPES, 2014), e tem grande atrativo para uso por conta de sua alta eficiência, baixo custo e larga vida útil (SCOPACASA). Essas características foram fundamentais para a inclusão da tecnologia neste trabalho, assim como a dimensão necessária para seu uso na maquete construída pelo grupo.

2.6 Servo Motor

O servo motor (Figura 3) é um dispositivo que realiza rotações limitadas em relação ao ângulo de giro, podendo variar de 0 – 180° (ZANETTI, et al). É de grande importância para aplicações em demonstrações de funcionalidades de automação, como por exemplo, abertura e fechamento de portões (exemplo a ser explicado neste projeto). As especificações do dispositivo utilizado pelo grupo estão disponíveis em Tabela 1.

Figura 3. Servo Motor

Fonte: (FILIPEFLOP)

Tabela 1. Especificações do Servo Motor.

Voltagem de Operação:	3,0 – 7,2V
Ângulo de rotação:	180 graus
Velocidade:	0,12 seg/60Graus (4,8V) sem carga
Torque:	1,2 kg.cm (4,8V) e 1,6 kg.cm (6,0V)
Temperatura de Operação:	-30C ~ +60C
Tipo de Engrenagem:	Nylon
Tamanho cabo:	245mm
Dimensões:	32 x 30 x 12mm
Peso	9g

2.7 Shield Ethernet

O *Shield Ethernet* é um dispositivo microcontrolador, que permite a conexão da placa Arduino à Internet. Ele é baseado no *chip* Wiznet W5100. Este *chip*, por sua vez, fornece a biblioteca de rede IP, suportando tanto protocolo UTP quanto UDP. (ALENCAR, et al). Possui entrada para micro SD, para armazenamento de qualquer arquivo.

Figura 4. Shield Internet

Fonte: (ALENCAR)

2.8 Cabo UTP (cat 5e)

O Cabo UTP – (*Unshielded Twisted Pair*), ou em tradução livre: “cabo trançado sem proteção” é composto de pares trançados de fios conectores e transmissores de dados. A categoria 5e, por sua vez, é constituído de 4 pares de fios. Esta categoria certifica o cabo UTP para transmissões de 1000 Mbps (TANEMBAUM). A categoria em questão será a utilizada para transmissão de dados do Arduino para o *Access point* no projeto.

2.9 Roteador

A comunicação entre dois computadores depende da ligação entre estes em alguns aspectos. Para realizar comunicação deve haver um meio físico capaz de conduzir dados de um dispositivo ao outro. É necessário que haja um meio físico capaz de conduzir a informação, e um meio lógico, que possa captar e traduzir os dados transmitidos. Para que este meio lógico interligue estes computadores é necessário que os mesmos se encontrem na mesma rede.

Porém, é necessário que exista comunicação entre redes distintas, sem isso não haveria a Internet. Neste caso, é realizada a adição de um dispositivo que esteja ligado a duas – ou mais – redes ao mesmo tempo, e desta forma, possa criar rotas entre elas, ou seja, encaminhar dados de uma rede à outra, gerando comunicação. O elemento na rede que faz este trabalho recebe o nome de roteador (ROCHA).

No projeto foi utilizado o roteador TL-WR841N, que tem o seguinte modelo exibido abaixo:

Figura 5. Roteador TL-WR841N

Fonte: (TP-LINK)

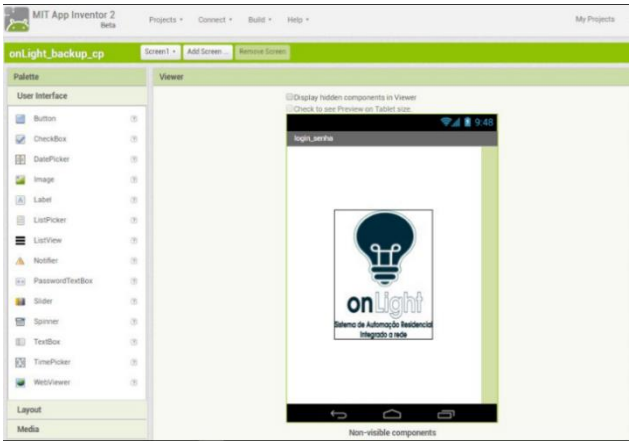
2.10 App Inventor

O App Inventor é uma plataforma de desenvolvimento de aplicativos. Criado pelo Google Labs, hoje a ferramenta que ensina a inúmeros alunos a criar e expor ideias em forma de aplicações pertence ao MIT Labs, do *Massachussets Institute of Technology* (Instituto de tecnologia de Massachussets).

A aplicação permite a criação de App's Android de uma maneira muito mais ágil e simples em relação à programação baseada a linguagens de programações convencionais. A estrutura de construção dos aplicativos segue o sistema de arrastar e soltar, através de blocos, o que permite que iniciantes utilizem o MIT App Inventor para projetos simples (S. SANTOS).

Neste projeto, se utilizará o App Inventor para a criação da interface entre usuário e sistema de automação (Figura 6). O aplicativo projetado controlará os dispositivos distribuídos pela residência.

Figura 6. Interface App Inventor

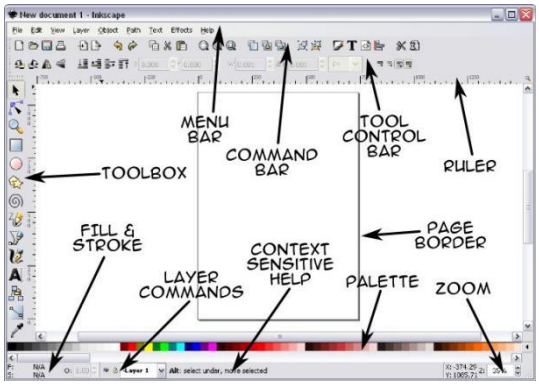


Fonte: (AUTORIA PRÓPRIA)

2.11 InkScape

Ferramenta de desenvolvimento de ilustrações digitais, utilizada por entusiastas de *design* em todo o mundo (INKSCAPE), disponível para as principais plataformas do mercado: Windows, GNU/Linux e Mac OS X. Nela, é possível criar e editar figuras (Figura 7), e assim, aplicá-las em seu projeto da forma desejada. Com recursos simples como criação de formas geométricas básicas, até utilizações mais complexas, como alteração de textura, contraste e cor (GOEDERT).

Figura 7. Interface InkScape



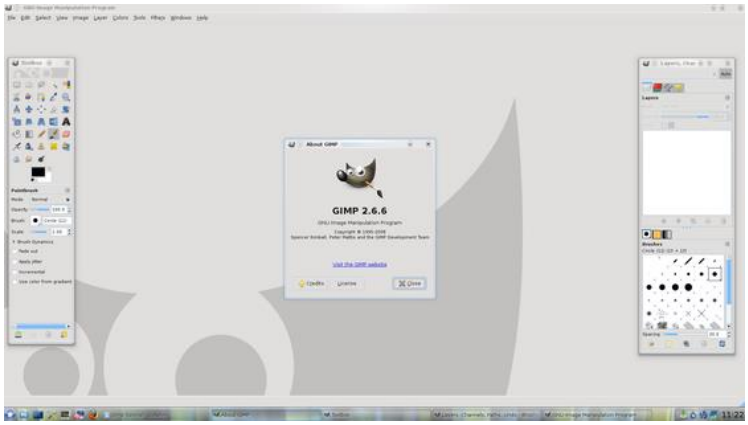
Fonte: (SHAW)

2.12 Gimp

O GIMP, ou "*GNU Imagine Manipulation Program*", é um editor de imagens disponível para as maiores plataformas do mercado (Windows, Linux e Mac OS X).

É um *software* livre, portanto, pode ser modificado e remodelado, criando vários tipos de editores em potencial (Figura 8) e podendo se adequar a inúmeras necessidades em edição de imagem (GIMP). O GIMP foi utilizado neste projeto na área de *design* junto ao InkScape, programas utilizados para criar e editar os ícones aplicados no projeto.

Figura 8. Interface GIMP.



Fonte: (GIMP)

2.13 Wireless

A tecnologia *wireless*, significa redes sem fio pois “não utilizam fios para transmissão de dados e sim somente o ar, realizando isto através de raios infravermelho, rádio, microondas ou laser.” (NÉRIO e RODRIGUES, p.6).

2.14 WLAN

Segundo REIS (2012, p.7) a WLAN (*Wireless Local Area Network*) é um tipo de rede *wireless* e serve como “uma alternativa às redes locais cabeadas ou pode ser usada como uma extensão da rede cabeada utilizando radiofrequência (RF)” objetivando o compartilhamento de recursos computacionais, e para conectar dispositivos que estabelecem comunicação por propagação de ondas de rádio, como *laptops, notebooks, smartphones,* entre outros. (NÉRIO e RODRIGUES, p. 6). Um exemplo muito conhecido desse tipo de rede é o Wi-Fi.

2.15 HTML

Como TANEMBAUM (2003, p. 476) descreve, atualmente as páginas web são escritas em linguagem chamada HTML (*HyperText Markup Language*), o que permite aos usuários produzir páginas que incluem texto, gráficos, imagens e *links* para outras páginas web. A HTML é uma linguagem de marcação, ou seja, ela descreve como os documentos devem ser formatados e assim os navegadores (*browsers*) interpretam esses os comandos e exibe a página web formatada aos usuários.

> Uma página da Web consiste em um cabeçalho e um corpo entre as tags (comandos de formatação) <html> e </html>, embora a maioria dos navegadores não reclame se essas tags não estiverem presentes. (...) o cabeçalho começa e termina com as tags <head> e </head> respectivamente, enquanto com o corpo é delimitado pelas tags <body> e </body>. Os strings entre as tags são chamados diretivas. A maioria das tags de HTML tem esse formato, ou seja, <algo> para marcar o início de alguma coisa e </algo> para marcar seu fim. A maioria dos navegadores tem um item de menu VIEW SOURCE ou algo parecido. Ao selecionar esse item, o código-fonte de HTML da página é exibido em lugar de sua versão formatada. (TANEMBAUM, 2003, p. 476)

Conforme demonstram respectivamente as figuras abaixo, um exemplo de uma página web escrita em linguagem HTML e como esta página é exibida por um navegador quando formatada.

Figura 9. Exemplo de código HTML de uma página web

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by fax. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a>
   <li> <a href="http://widget.com/products/little"> Little widgets </a>
</ul>
<h2> Telephone numbers </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS
   <li> By fax: 1-415-765-4321
</ul>
</body>
</html>
```

Fonte: TANEMBAUM, 2003, p. 477.

Figura 10. Exemplo de página web formatada.

Fonte: TANEMBAUM, 2003, p. 477.

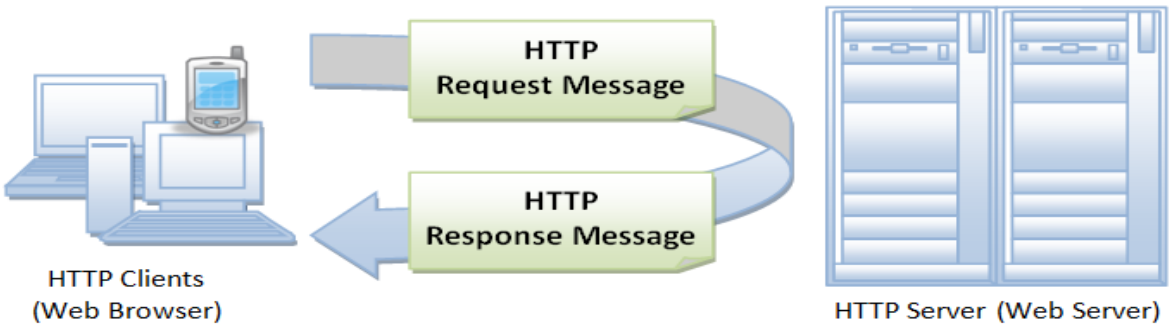
2.16 Protocolo HTTP

O HTTP (*Hypertext Transfer Protocol*), significa Protocolo de Transferência de Hipertexto e “é utilizado para transmitir dados através da *World Wide Web*” (SANTOS, 2017, p.4), conforme o acordo W3C (*World Wide Web Consortium*) que rege as normas e padronização da internet, assinado pelo MIT (*MassachusettsInstitute of Technology*) e o CERN (*Organisation Européenne pour la Recherche Nucléaire* – Centro Europeu para Pesquisa Nuclear), definindo este protocolo como padrão para navegação por páginas de hipertexto. (BEOCK et al., p. 2-3).

Uma comunicação HTTP ocorre segundo o mecanismo do tipo requisição/resposta (*request/reply*) (TEIXEIRA, 2004, p. 16). Este protocolo define como os clientes web requisitam páginas Web aos servidores e como eles as transferem a clientes.

> Quando um usuário requisita uma página Web (por exemplo, clica sobre um *hiperenlace*), o browser envia ao servidor mensagens de requisição HTTP para os objetos da página. O servidor recebe as requisições e responde com mensagens de resposta HTTP que contêm os objetos. (KUROSE e ROSS, 2010, p. 73).

Figura 11. Exemplo de comunicação HTTP (*request/reply*)



Fonte: (HOCK-CHUAN)

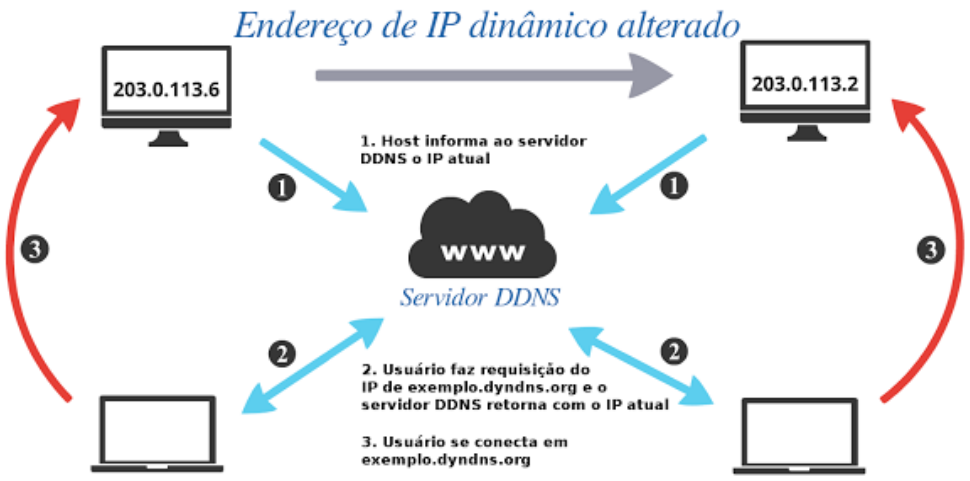
2.17 DNS e DDNS

Para estabelecer comunicação entre computadores na internet, ou em uma rede loca, é necessário que os dispositivos conheçam o endereço um do outro, para que possam saber para onde enviar os dados. Como os endereços são números, e seria muito difícil associar números à um site que se queira acessar, por exemplo, foi criado o serviço DNS (*Domain Name System*), que estabelece uma relação entre um nome e um endereço, fazendo com que se possa acessar domínios apenas com o nome dele.

Uma limitação existente neste serviço é ocasionada pela alteração constante dos endereços públicos nos computadores conectados à Internet, que necessita de um serviço que ligue um computador a um nome, mesmo que seu IP seja alterado após cada vez que este tem um novo endereço IP atribuído.

O DDNS (*Dynamic Domain Name System*) possibilita a alteração do IP relacionado ao nome no DNS, servindo como um complemento ao serviço, fazendo com que seja possível acessar o mesmo computador com “uol.com.br” mesmo que seu IP seja alterado constantemente (CARDOSO), como visto na Figura 12:

Figura 12. Funcionamento do DDNS



Fonte: CARDOSO, 2014.

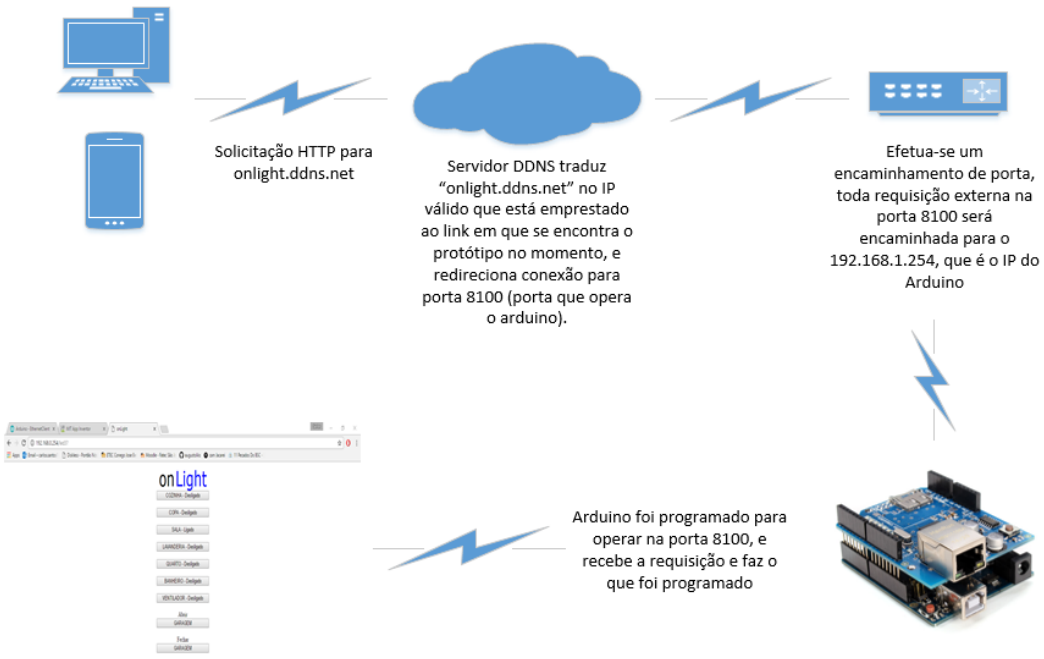
O uso deste serviço foi essencial para estabelecer o acesso externo, pois os IP’s válidos na Internet estão em constante alteração.

3. DESENVOLVIMENTO

Este capitulo visa demonstrar todo o passo a passo do projeto. Os eventos que compuseram sua formação, tais como elementos funcionais e não funcionais, topologia lógica e topologia física.

Ao fim do desenvolvimento, queremos alcançar o seguinte sistema de funcionamento:

Figura 13. Sistema de Automação Final



Fonte: Autoria Própria

Os passos seguem desde a solicitação HTTP até a realização da tarefa ordenada pela placa Arduino Uno. Todos os passos para o funcionamento deste sistema serão relatados neste capítulo.

3.1 Premissas

Para a realização deste projeto de conclusão de curso, foram necessários os seguintes itens, com os respectivos preços, pesquisados no ano de 2017, presentes na Tabela 2.

Tabela 2. Preços – Materiais

MATERIAL	ESPECIFICAÇÃO	QUANTIDADE	PREÇO	TOTAL
Led´s	5VCC	10	R$ 5,00	
Servo Motor	10 A	1	R$ 18,00	
Jumpers	Macho-Fêmea	40	R$ 17,00	
Jumpers	Macho-Macho	65	R$ 16,00	
Shield Ethernet	WG5100	1	R$ 70,00	**R$: 341,00**
Arduino	UNO	1	R$ 70,00	
Estrutura de madeira	Maderite	2 Metros²	R$ 50,00	
Roteador	Tl-wr841n	1	R$ 70,00	
Cabo UTP	Cat 5 e	10 Metros	R$ 10,00	
Cooler	Tech F7-100	1	R$ 15,00	

Fonte: Autoria Própria

O grupo consultou as seguintes empresas do ramo Grupo Feliza – Home e Automação, DHT – Design Home Theater e Inmótica sobre custo de mão de obra. Após coletar as informações concluiu-se que o preço base médio é de R$ 900,00 para um projeto de pequeno porte. Em serviços de maior escala, o valor pode alternar entre 7% e 15% do orçamento total.

Tabela 3. Preços – *Softwares*

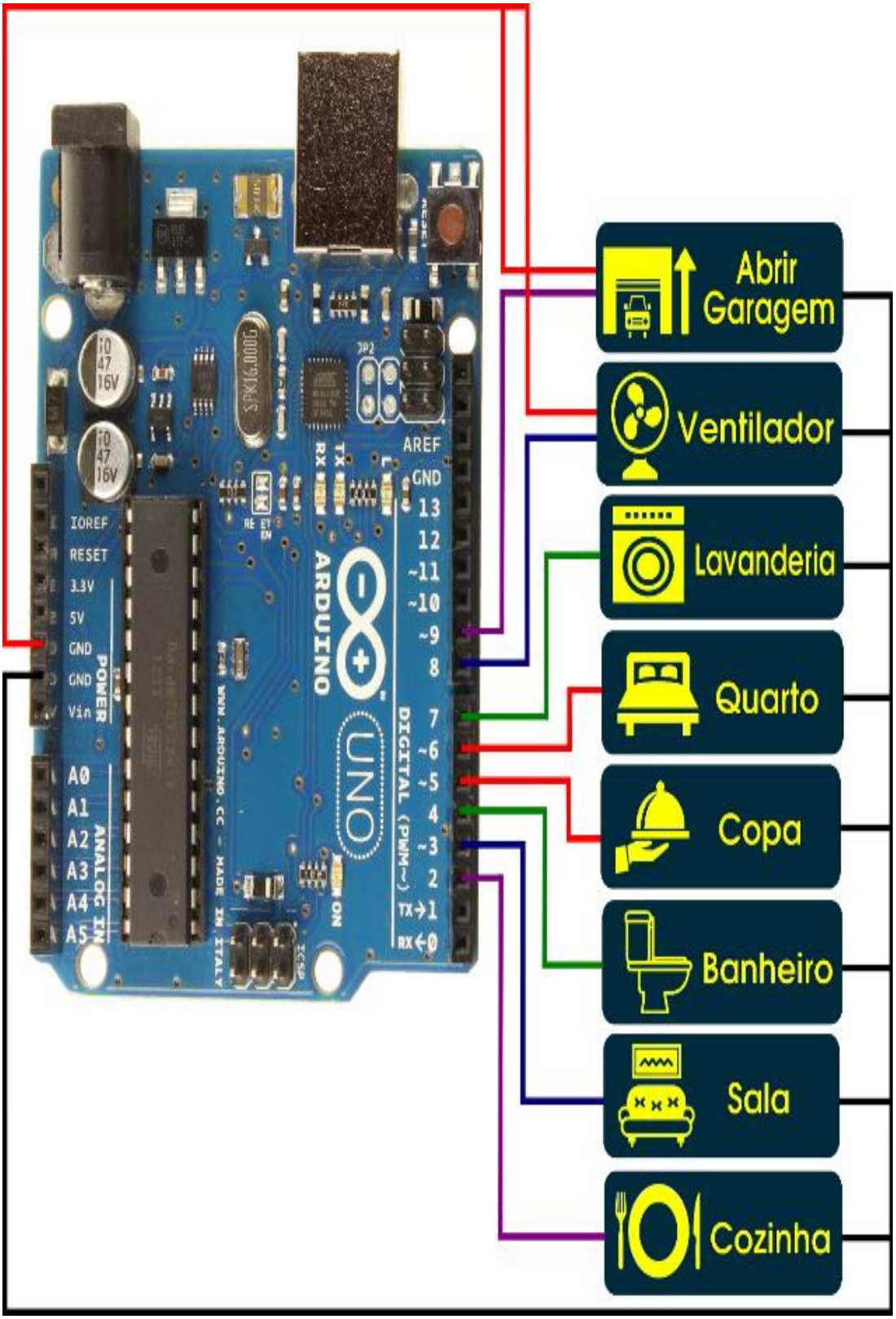
SOFTWARE	ESPECIFICAÇÃO	LICENSA
InkScape	Edição gráfica	Gratuita
Gimp	Edição gráfica	Gratuita
IDE Arduino	Interface de desenvolvimento	Gratuita

Fonte: Autoria Própria

3.2 Topologias

Abaixo, na Figura 14, podemos visualizar a disposição dos pinos da placa Arduino Uno em relação a cada dispositivo distribuído pela casa, na forma dos ícones exibidos aos usuários.

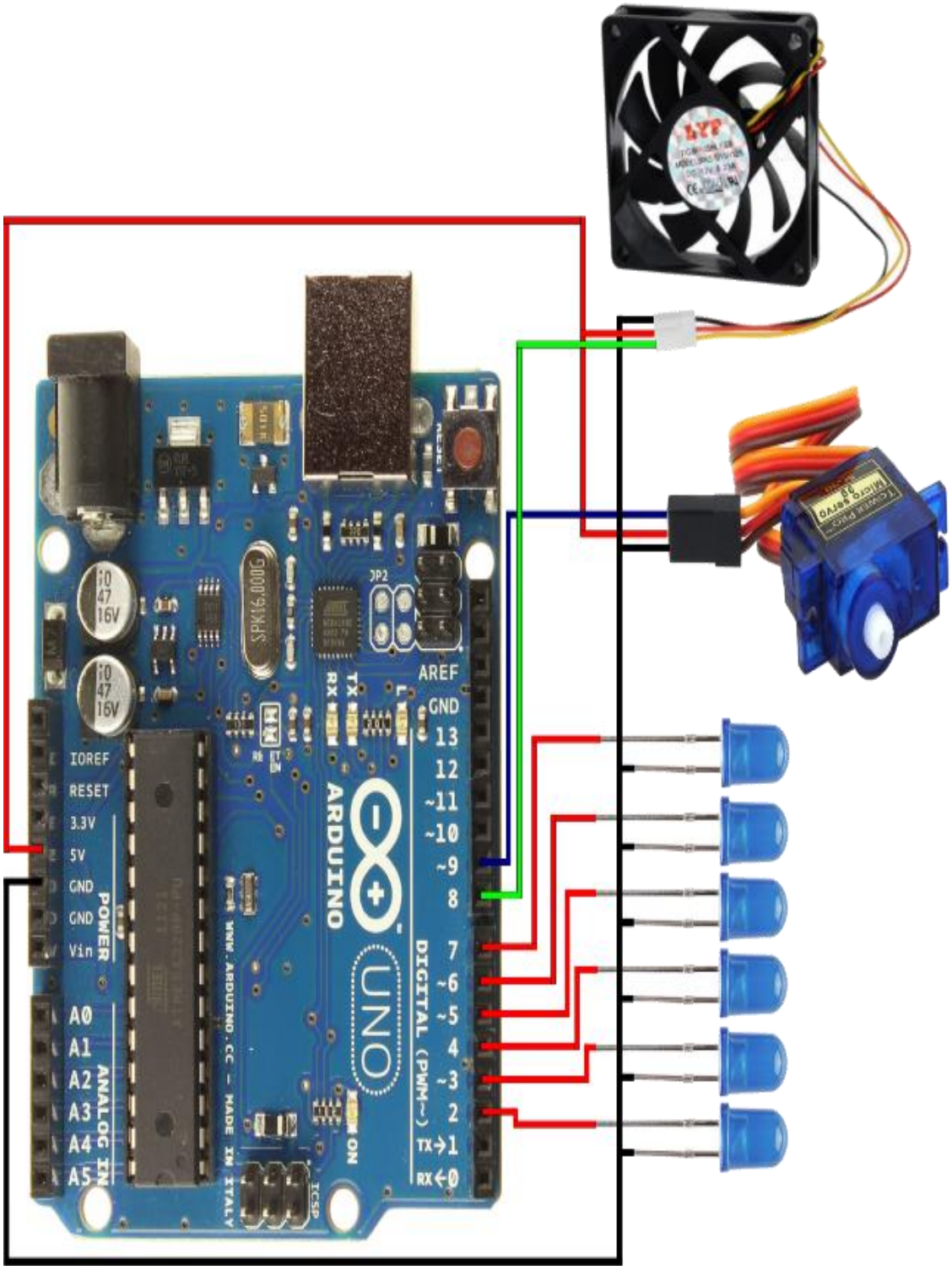
Figura 14. Topologia da placa Arduino



Fonte: Autoria Própria

Já na Figura 15, temos a disposição utilizando as imagens dos dispositivos utilizados no projeto.

Figura 15. Topologia da placa Arduino com dispositivos físicos



Fonte: Autoria Própria (adaptada)

E abaixo temos a disposição dos pinos, listados de acordo com sua numeração e dispositivo responsável, na Tabela 4.

Tabela 4. Topologia da Placa Arduino

PINO	Dispositivo/Componente
2	Led – Cozinha
3	Led – Sala
4	Led – Banheiro
5	Led – Copa
6	Led – Quarto
7	Led – Lavanderia
8	Cooler – Ventilador
9 ~ (PWM)	Servo – Garagem

Fonte: Autoria Própria

3.3 Código no Arduino

Para que o microcontrolador possa obter um funcionamento coeso, foi necessário o desenvolvimento um código fonte que desempenhará as funções de controle dos dispositivos, que em conjunto, realizarão a automação residencial integrada a rede domiciliar, dentro do que foi estipulado no objetivo do projeto.

O código foi desenvolvido na IDE oficial do Arduino, que se caracteriza por ser leve e, além disso, ter um mecanismo simples para a inserção de bibliotecas programáveis.

Abaixo serão listadas as partes mais relevantes do código. O código completo, se encontra no apêndice, ao final deste documento.

3.3.1 Inserção de IP e endereço MAC

Declaração do Mac Address e IP do *Shield Ethernet*, e a porta de operação da página HTML:

Figura 16. Inserindo IP e MAC no Arduino

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Mac do Arduino
byte ip[] = { 192, 168, 0, 254 }; //IP do Arduino

EthernetServer server(8100); //Selecionar porta de saída
```

Fonte: Autoria Própria

3.3.2 Declaração da Pinagem dos Dispositivos e Componentes

Para cada dispositivo, é necessário explicitar uma variável com o respectivo valor do pino que ele será inserido, como mostrado abaixo:

Figura 17. Definindo pinos às variáveis

```
int led1 = 2;   //Cozinha
int led2 = 3;   //Copa
int led3 = 4;   //Sala
int led4 = 5;   //Lavanderia
int led5 = 6;   //Quarto
int led6 = 7;   //Banheiro
int ventilador = 8;
```

Fonte: Autoria Própria

3.3.3 Definição de Dispositivos de Entrada e Saída

É necessário declarar as variáveis dos dispositivos como entrada (INPUT) ou saída (OUTPUT), no caso do projeto as entradas, serão feitas pelo aplicativo ou pela página HTML, então houve apenas saídas de sistema. O servo também teve que ser declarado nessa rotina, por predeterminação de sua biblioteca de controle.

Figura 18. Definição de saída e entrada de sinal, código Arduino

```
void setup() {
  Ethernet.begin(mac, ip);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(ventilador, OUTPUT);
  servo.attach(9);
}
```

Fonte: Autoria Própria

3.3.4 Verificação por URL do Dispositivo Requisitado

A estrutura das condições lógicas, permanecerá quase a mesma para todos os casos, mudando somente o código de controle que será inserido depois do IP, e o componente que será acionado em relação a condição satisfeita.

A linha de código que contém “digitalWrite(led6, !digitalRead(led6))”, é responsável por fazer uma leitura do estado lógico anterior do dispositivo, e quando acionado, mudar o estado lógico, exemplo se o “led6” estiver com sinal lógico baixo, quando for requisitado http://192.168.0.254/led6 ele mudará o estado lógico baixo para estado lógico alto, e vice versa.

Figura 19. Verificação do campo de URL

```
if(readString.indexOf("led6") >= 0) {
  digitalWrite(led6,!digitalRead(led6));
}

if(readString.indexOf("garagemUP") >= 0) {
  servo.write(180);
}

if(readString.indexOf("garagemDOWN") >= 0) {
  servo.write(0);
}

if(readString.indexOf("ventilador") >= 0) {
  digitalWrite(ventilador,!digitalRead(ventilador));
}
```

Fonte: Autoria Própria

3.3.5 Criação da página HTML

Para cada linha de comando em HTML, foi necessário a utilização do comando "*client.PrintLn*("...");" da biblioteca Ethernet, que fará a com que o "*client*", que no caso é o Shield Ethernet, execute o comando dando quebra de linha ao seu final.

Figura 20. Produção de Página

```
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println();
client.println("<!doctype html>");

client.println("<html>");
client.println("<head>");
client.println("<title>onLight</title>");
client.println("<meta name=\"viewport\" content=\"width=320\">");
client.println("<meta name=\"viewport\" content=\"width=device-width\">");
client.println("<meta charset=\"utf-8\">");
client.println("<meta name=\"viewport\" content=\"initial-scale=1.0, user-scalable=no\">");
client.println("</head>");
client.println("<body>");
client.println("<center>");

client.println("<font size=\"10\" face=\"verdana\" color=\"black\">on</font>");
client.println("<font size=\"10\" face=\"verdana\" color=\"blue\">Light</font><br />");
```

Fonte: Autoria Própria

3.3.6 Criação de botões e verificação do estado lógico dos componentes

Os botões foram criados para deixar o sistema interativo, sem a necessidade dos usuários decorarem os códigos de controles.

Figura 21. Adição de botões

```
if(digitalRead(led1)){
  statusLed = "Ligado";
}

else{
  statusLed = "Desligado";
}

client.println("<td> <form action=\"led1\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">COZINHA - "+statusLed+"</button>");
client.println("</form> <br /></td>");
```

Fonte: Autoria Própria

3.4 Encaminhamento de portas

Para que o Arduino possa receber requisições de dispositivos via internet, foi necessário criar um encaminhamento de porta, para que as requisições feitas a partir de dispositivos que se encontra na internet, seja encaminhada para o Arduino, o mesmo executará o que foi programado para fazer.

A figura abaixo, demonstra o encaminhamento de porta criado para as necessidades do projeto (destacado de amarelo). Foi criado uma regra intitulada “OnLight”, que indica que as requisições externas a partir de qualquer porta, será redirecionada para o IP do Arduino pela porta 8100.

Figura 22. Encaminhamento de Porta

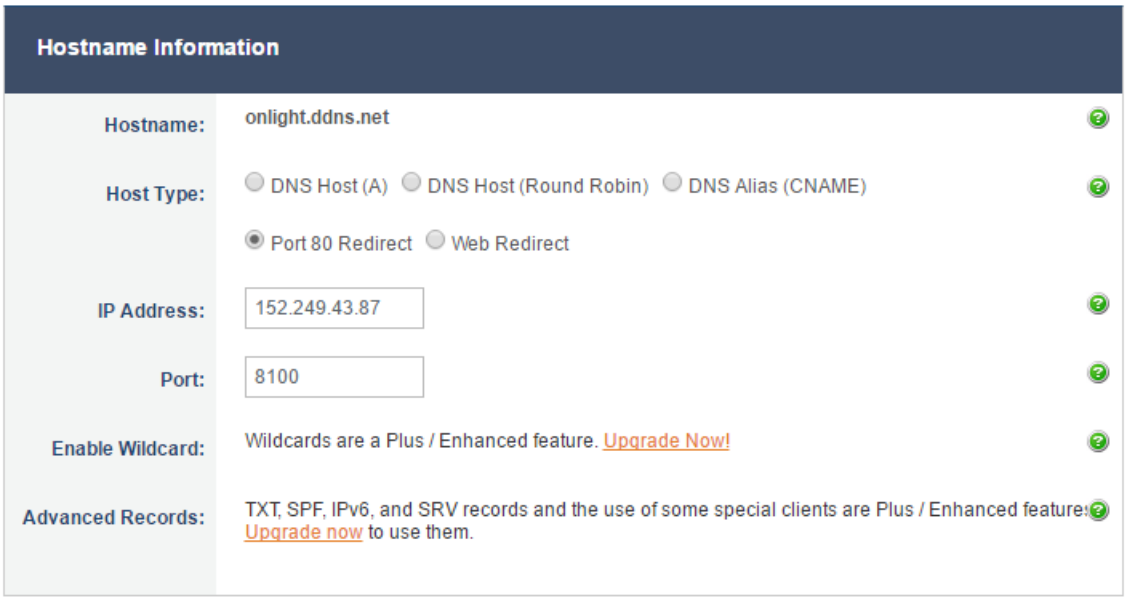
Fonte: Autoria Própria

3.5 Criação de DDNS

Para a criação do DDNS, foi utilizado o provedor de DNS Dinâmico noIp. Os principais fatos que fizeram o noIp estar presente no projeto foi: ser gratuito, ter a opção de redirecionamento de porta do ponto final já que o Arduino estará operando na porta 8100 e não na porta 80, padrão do protocolo HTTP, e o roteador presente no projeto ter um suporte para este provedor de DDNS, automatizando a troca do IP válido do link onde se encontra o protótipo.

A figura abaixo demonstra as características do DDNS criado. A linha que se encontra *Hostname,* é responsável pela a declaração do nome do domínio, no caso foi colocado onlight.ddns.net, *Host Type* é a forma de atuação do DDNS, *port 80 redirect* é a solução para que todas as requisições para onlight.ddns.net feitas na porta 80, traduza para um *socket* (IP e porta de serviço), ou seja, IP válido no que se encontra o Arduino, e a porta 8100 que é a porta encaminhada para o IP fixo local do Arduino.

Figura 23. Configuração DDNS



Fonte: Autoria Própria

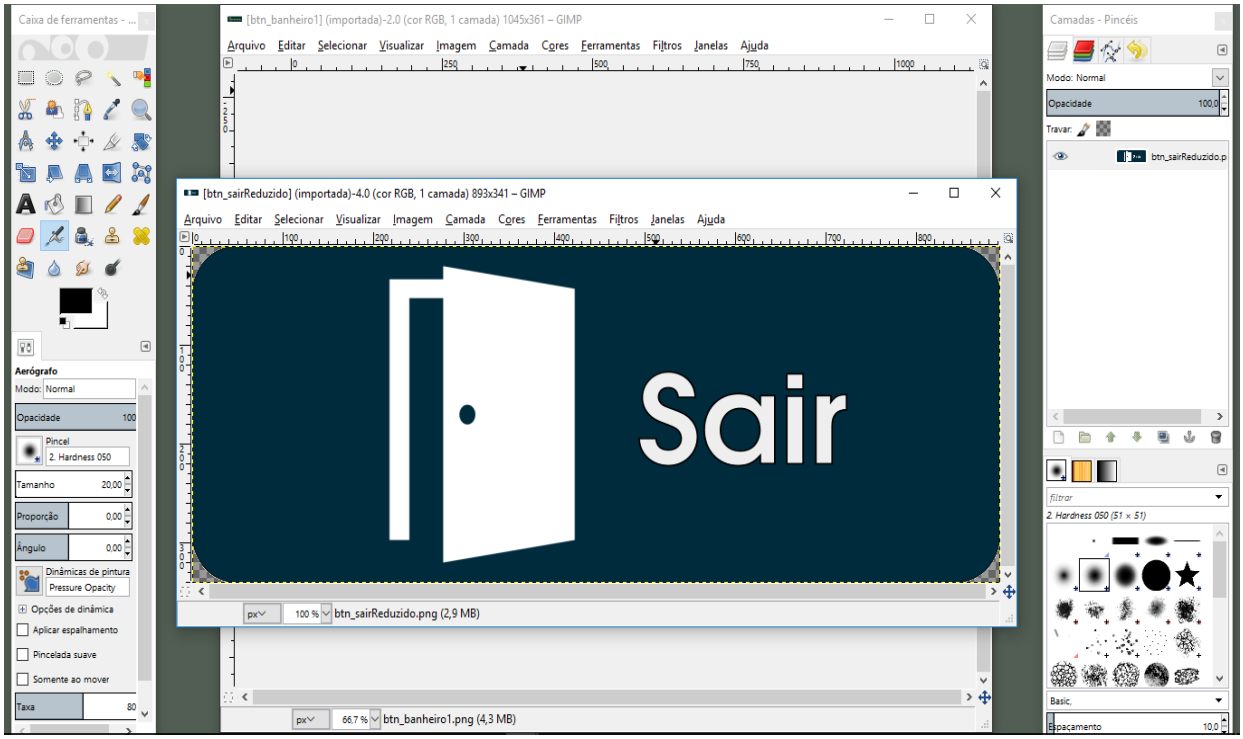
3.6 Layout do Aplicativo

Foi utilizado para edição dos temas do aplicativo, o *software* InkScape, por ser uma ferramenta muito útil para tratar dimensões e cores de imagem, e o *software* Gimp para o tratamento de cores, retirada de fundos indesejados das imagens e conversão de suas extensões.

Figura 24. Criando *Layout* – Parte 1

Fonte: Autoria Própria

Figura 25. Criação de L*ayout* – Parte 2



Fonte: Autoria Própria

Inicialmente foram criados vários ícones para o *layout* do projeto, mas boa parte não está presente na versão final por limitação da plataforma de desenvolvimento do *software* (App Inventor), e mudanças na estrutura do projeto.

3.7 Programação Aplicativo

O aplicativo intitulado “OnLight” contém em sua estrutura 4 telas, e em sua interface serão listados os principais blocos programáveis que compõem o funcionamento do aplicativo. Abaixo serão exibidas as partes mais relevantes do código. O Código completo, se encontra no apêndice, ao final deste documento.

Figura 26. Tela Inicial

Fonte: Autoria Própria

Figura 27. Menu de Opções

Fonte: Autoria Própria

Figura 28. Menu de Opções – Acesso via Internet

Fonte: Autoria Própria

Figura 29. Menu de Opções – Acesso via Rede Local

Fonte: Autoria Própria

Figura 30. Controle Manual – Lista de Blocos

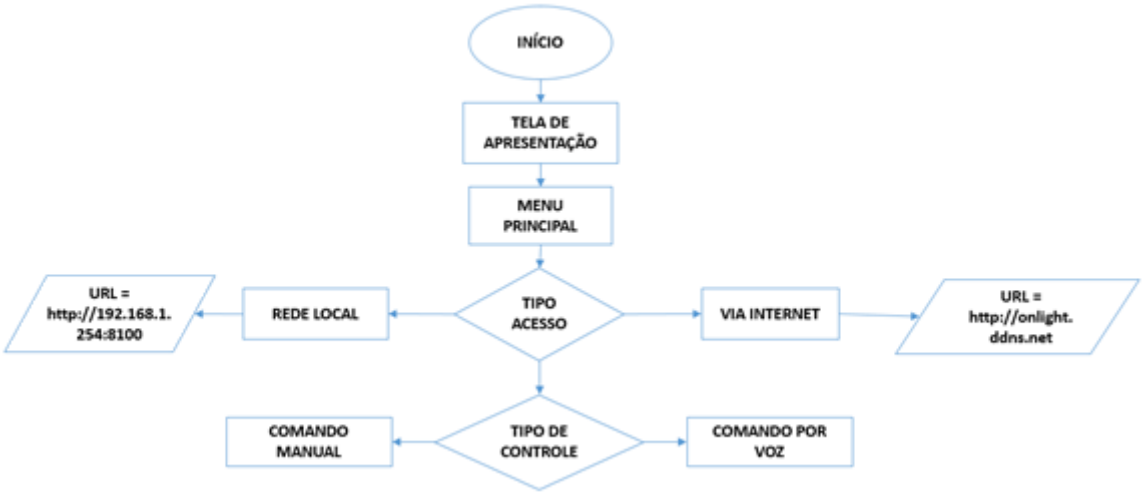
Fonte: Autoria Própria

Figura 31. Tela de Controle de Voz

Fonte: Autoria Própria

Ao fim da consolidação das telas, este será o fluxograma de funcionamento do aplicativo onLight.

Figura 32. Fluxograma de funcionamento – Aplicativo

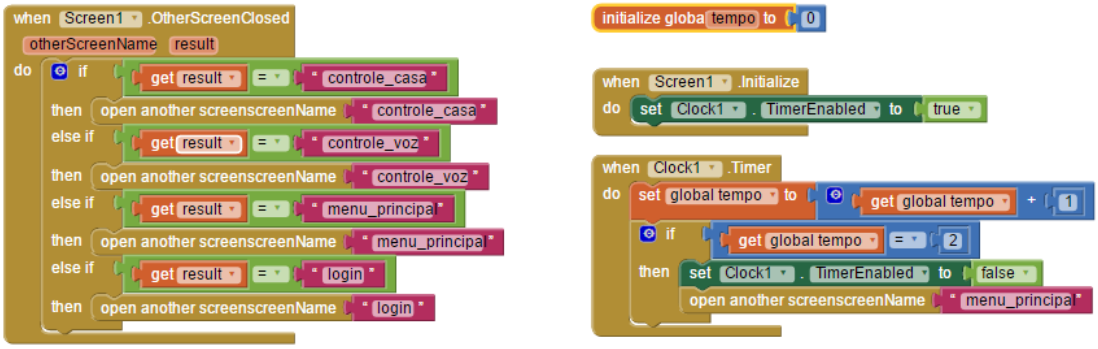


Fonte: Autoria Própria

3.8 Blocos da Screen 1 – Gerenciador de Tela

A *Screen* 1, visualmente não oferece nenhum tipo de ação, apenas uma imagem do logotipo do aplicativo em um fundo branco, porém logicamente ele é responsável pela função de *screen manager*, ou seja, ele é o gerenciador de todas as telas do aplicativo, função que tem o intuito de eliminar telas duplicadas e otimizar a memória reservada para o aplicativo.

Figura 33. *Screen* 1



Fonte: Autoria Própria

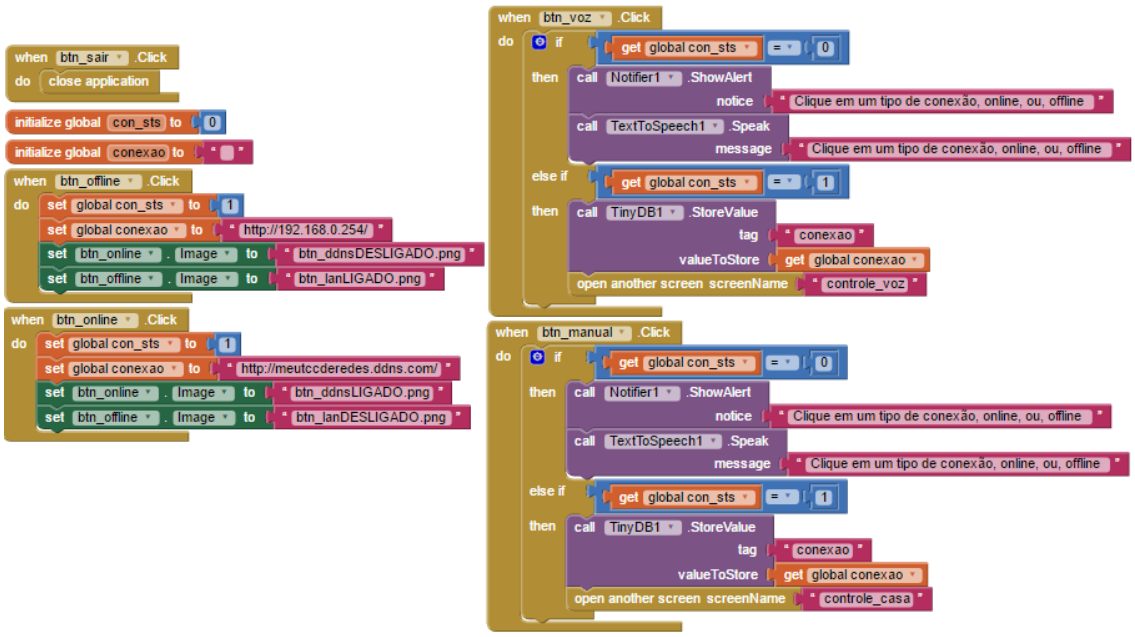
A *Screen* 1 sempre estará aberta em segundo plano, e quando outra tela se fechar, retornará para a Screen1 em formato *String*, o nome da tela que o usuário espera ser aberta, e com este retorno, o *screen* 1 abre a tela com o nome do retorno, mantendo apenas duas telas na memória.

3.9 Blocos da Screen 2 – Menu

A *Screen* 2, é a tela de menu principal, ela que irá possibilitar o usuário escolher de que maneira deseja controlar sua casa, por *touch screen* ou por comando de voz.

Também há a opção de escolher o tipo de conexão, rede local ou via Internet.

Figura 34. *Screen 2*

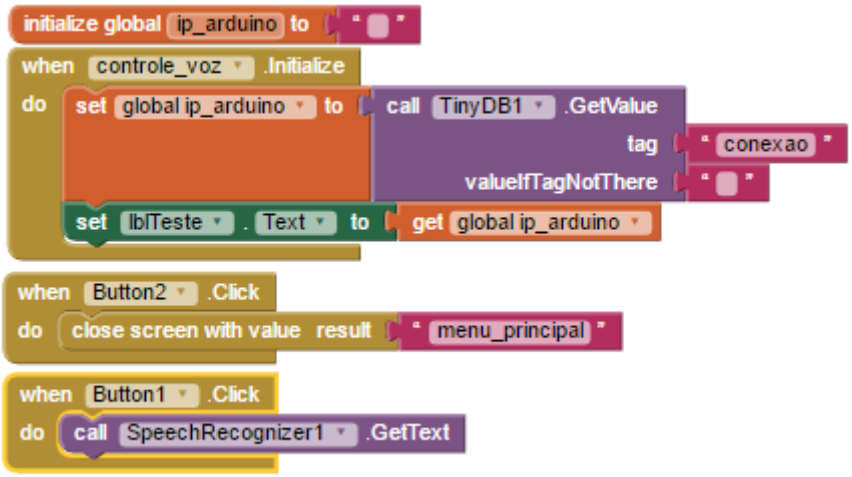


Fonte: Autoria Própria

3.10 Blocos da Screen 3 – Controle por voz

A *Screen* 3 possibilita o usuário fazer acionamento dos dispositivos do sistema por comando por voz. Será captado o que o usuário disse e executado a seguinte condição lógica: Se o sistema identificar as seguintes palavras “sala, lavanderia, copa, banheiro, quarto, cozinha, abrir garagem, fechar garagem” ele irá mudar o estado em que se encontra o dispositivo, exceto “abrir garagem, fechar garagem”, onde o sistema apenas executará o comando que a frase explicita. Vale ressaltar que esta opção necessita de conexão do dispositivo a internet.

Figura 35. *Screen* 3

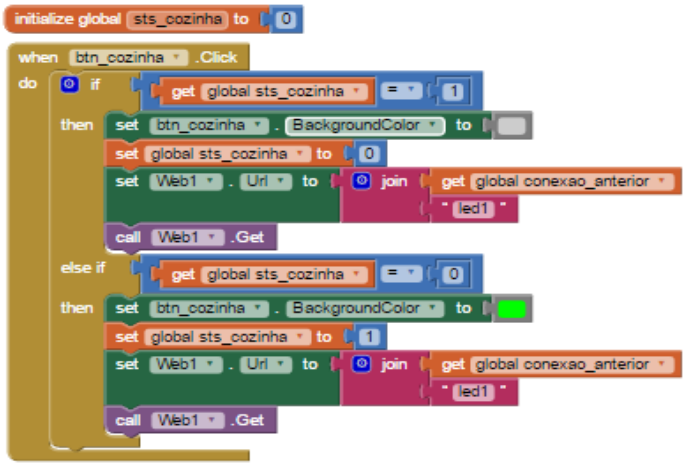


Fonte: Autoria Própria

3.11 Blocos da Screen 4 – Acionamento por botões

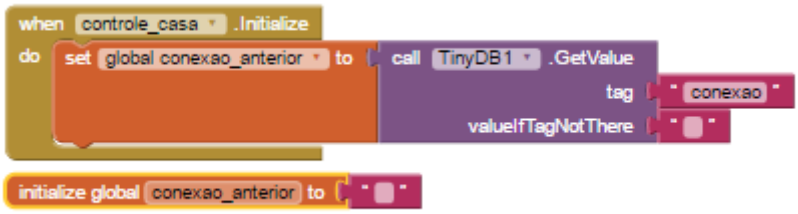
A *screen* 4 possibilita o usuário fazer acionamento dos dispositivos do sistema por botões. Será listado botões com o nome dos cômodos da casa, indicando a localização dos LED's no protótipo, acionamento do *cooler*, além de ter dois botões exclusivos para abrir e fechar o portão da garagem.

Figura 36. *Screen* 4, parte 1



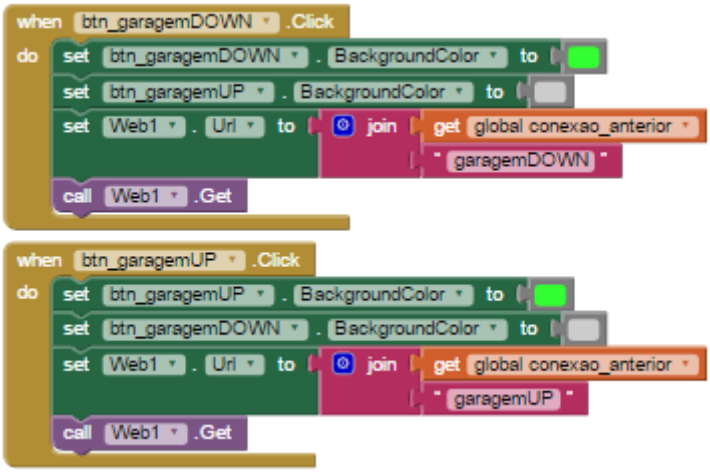
Fonte: Autoria Própria

Figura 37. *Screen* 4, parte 2



Fonte: Autoria Própria

Figura 38. *Screen* 4, parte 3



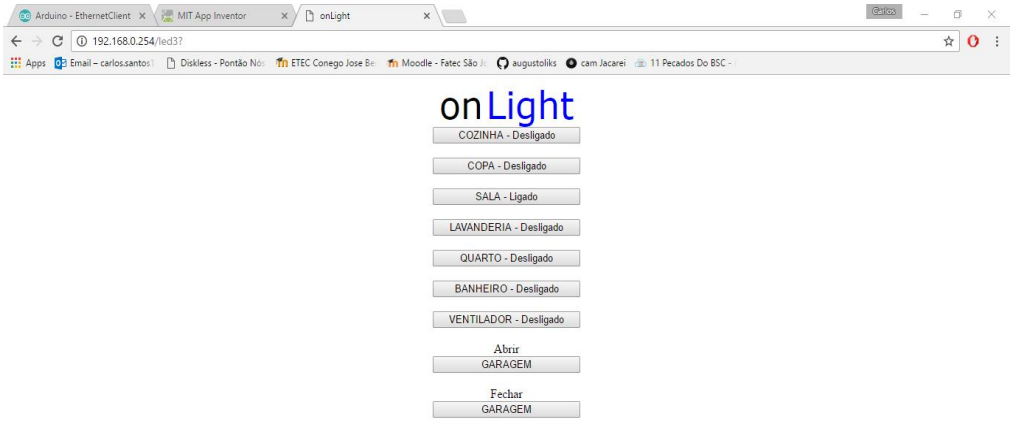
Fonte: Autoria Própria

3.12 Funcionamento

O acionamento dos dispositivos interligados ao Arduino será feito a partir da leitura da URL, ou seja, os botões foram programados para que quando apertados, gerassem um código e executassem a função que foi pré-estabelecida na programação da placa microcontroladora.

No exemplo foi pressionado o botão “Sala”, o Arduino fez uma verificação lógica, e identificou que “192.168.0.254/led3” está programado para acender o “led3” do projeto, e no protótipo, o “led3” está localizada na sala de estar.

Figura 39. Página de controle

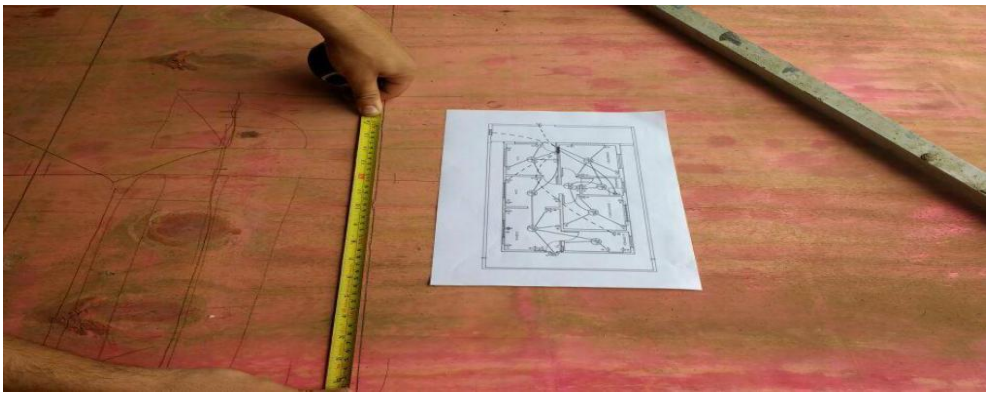


Fonte: Autoria Própria

3.13 Montagem da Maquete

Dentro da parte prática do projeto, está a simulação de uma residência automatizada utilizando uma maquete que simula as funções apresentadas.

Figura 40. Montando maquete de residência automatizada – Parte 1



Fonte: Autoria Própria

Figura 41. Montando maquete de residência automatizada – Parte 2

Fonte: Autoria Própria

Figura 42. Montando maquete de residência automatizada – Parte 3

Fonte: Autoria Própria

Figura 43. Montando maquete de residência automatizada – Parte 4

Fonte: Autoria Própria

Figura 44. Montando maquete de residência automatizada – Parte 5

Fonte: Autoria Própria

Figura 45. Montando maquete de residência automatizada – Parte 6

Fonte: Autoria Própria

3.14 Análises e testes

Os testes foram muito satisfatórios, conseguindo alcançar com êxito os objetivos inicias do projeto.

O aplicativo apresentou os requisitos necessários, porém o tempo de resposta ao toque não é insignificante, com latências consideráveis, mas não prejudiciais ao funcionamento.

Apresentação de falhas: leve atraso ao descer o *scroll* da tela de acionamento dos dispositivos por botões. Apresentaram pequenas falhas no reconhecimento de voz para alguns timbres, apresentando as vezes, confusão entre a palavra “fala” e “sala”.

4. DESAFIOS E DIFICULDADES

Neste capítulo haverá uma breve descrição dos maiores obstáculos e desafios encarados no desenvolvimento do projeto.

4.1 Aplicativo

No desenvolvimento do aplicativo, houve falhas que não condiziam com erros de programação, e foi um enorme obstáculo descobrir que a respectiva falha se dava presentes por limitação da plataforma de desenvolvimento App Inventor, que não conseguia manter muitas imagens (por volta de 6) de alta resolução por tela, encerrando o aplicativo inesperadamente.

4.2 Protótipo

Obteve-se dificuldades na montagem do protótipo, pelo fato de necessitar conhecimento e manuseio de ferramentas e materiais de marcenaria, vale ressaltar a dificuldade em todo o preparo da fiação e fixação dos componentes nos seus devidos lugares.

4.3 Código Arduino

Dificuldades em entender o funcionamento dos comandos das bibliotecas presentes no código fonte.

4.4 Acesso Interno

Modo de recebimento de IP do Shield Ethernet, e acesso a página HTML.

4.5 Acesso Externo

Foi, sem sombra de dúvidas, a maior dificuldade do projeto, e por contrapartida, o maior desafio superado em todo desenvolvimento do mesmo.

O fato de que levou o item a devida dificuldade, foi ter que elaborar análises nos *links* dos provedores de acesso à Internet, pois alguns provedores têm em seu caminho de rede, o bloqueio de várias portas. Já em outros provedores não chegam a disponibilizar o acesso a qualquer encaminhamento de porta pelo cliente, não tornando possível o respectivo encaminhamento para o dispositivo final.

A solução encontrada, foi utilizar o comando *nmap* (comando de linux) para elaborar uma varredura de portas do *link*, e criar um encaminhamento para uma porta da qual não estava sendo utilizada e listada no *nmap*.

5. CONCLUSÃO

Obteve-se êxito nos objetivos do presente projeto, que pôde demonstrar na prática um exemplo do conceito de *IoT* com a realização de um sistema de automação residencial em um protótipo de madeirite.

Houve vários desafios durante o desenvolvimento, dos quais enfatiza-se a dificuldade na conexão e gerenciamento remoto do sistema, pois inicialmente para realizar a conexão remota foram utilizadas as portas 80, 8080 (HTTP), e 443, 8443 (HTTPS), pois estas portas estão sempre abertas, independente do acesso. Porém, a conexão não estava funcionando mesmo com as devidas configurações, e então percebeu-se que devido as portas já serem utilizadas pelos protocolos HTTP e HTTPS, por mais que se encontrassem abertas para utilização, elas já possuem um serviço primário para trafegar dados pela Internet, e por isso, acabavam negando a solicitação de conexão para outros fins. Contudo esta foi uma dificuldade superada ao se utilizar uma porta não muito explorada, a porta 8100, o que trouxe grande aprendizado técnico, pela observação e resolução do problema.

A dificuldade no encaminhamento de portas no DDNS, fez o grupo refletir e observar de outra forma a dinâmica de comunicação de redes de computadores, o que foi um ponto muito produtivo do projeto.

5.1 Trabalhos Futuros

Algumas sugestões de melhoria para este projeto são:

- Implementação de uma VPN para gestão remota via WAN;
- Desenvolvimento de aplicativos de controle para iOS;
- Autenticação por banco de dados em um servidor privado;
- Adição de dispositivos de segurança de residências.

6. REFERÊNCIAS BIBLIOGRÁFICAS

- ALENCAR, et. al. **Automação residencial de baixo custo com Arduino Mega e Ethernet Shield.** PUC Goiás. Disponível em: <http://professor.pucgoias.edu.br/SiteDocente/admin/arquivosUpload/17829/material/ARTIGO01.pdf > Acesso em 06 abr. 2017.

- BONATO, Vanderlei. **Introdução à Ciência da Computação I**: Conceitos básicos sobre computadores. Universidade de São Paulo, 2011. Disponível em < http://wiki.icmc.usp.br/images/4/41/SSC0800_Aula1.pdf >. Acesso em: 04 dez. 2016.

- CECATO, João Carlos. **Casa Inteligente de Baixo Custo.** Universidade São Francisco, 2010. Disponível em: < http://lyceumonline.usf.edu.br/salavirtual/documentos/1903.pdf> Acesso em 15 abr. 2017.

- COELHO, Darlene; CRUZ, Victor. **Edifícios inteligentes: uma visão das tecnologias aplicadas.** Editora Edgard Blücher Ltda, cap 8, 2017. Disponível em: <http://pdf.blucher.com.br.s3-sa-east1.amazonaws.com/openaccess/9788580392210/08.pdf> Acesso em 17 mar. 2017.

- DEVAL, Felipe. **Automação Residencial de Baixo Custo Utilizando Tecnologias Open Source**. Centro Universitário de Araraquara, 2015. Disponível em: <https://www.uniara.com.br/arquivos/file/cca/artigos/2015/felipe-antonio-deval.pdf> Acesso em 03 abr. 2017.

- DR. R. **How to Use Arduino Ethernet Shield**. Disponível em: <http://wiznetmuseum.com/portfolio-items/how-to-use-arduino-ethernet-shield/ jun.> Acesso em: 2017

- EVAN, Daves. **A Internet das Coisas: Como a próxima evolução da Internet está mudando tudo.** Cisco, 2011. Disponível em:

<http://www.cisco.com/c/dam/global/pt_br/assets/executives/pdf/internet_of_things_iot_ibsg_0411final.pdf> Acesso em 12 mar. 2017.

- FILIPEFLOP. **Micro Servo 9g SG90 TowerPro**. Disponível em: <http://www.filipeflop.com/pd-71590-micro-servo-9g-sg90-towerpro.html> Acesso em 04 jun. 2017

- GIMP, autor desconhecido. **About GIMP.** Disponível em: <https://www.gimp.org/about/> Acesso em 15 abr. 2017.

- GOEDERT, Elciana, org. II. **Inkscape, versão 0.1.09: criação e ilustrações vetoriais v.2 Secretaria de Estado da Educação.** Superintendência da Educação. Diretoria de Tecnologias Educacionais, 2010. Disponível em: <http://www.gestaoescolar.diaadia.pr.gov.br/arquivos/File/tutoriais/inkscape2.pdf> Acesso em 15 abr. 2017.

- HACHOUCHE, Anwar. **Apostila Arduino Básico.** Eletrogate. Disponível em: <http://apostilas.eletrogate.com/Apostila_Arduino_Basico-V1.0-Eletrogate.pdf> Acesso em 07 mai. 2017

- HOCK-CHUAN, CHUA. **HTTP (HyperText Transfer Protocol)**, 2009. Disponível em: <https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html> Acesso em: 04 jun. 2017

- INKSCAPE. Autor desconhecido. **Visão geral sobre o InkScape.** Disponível em: <https://inkscape.org/pt-br/sobre/> Acesso em 17 abr. 2017

- JORDÃO, Fábio. **O que é Cooler?** TecMundo. Disponível em: <https://www.tecmundo.com.br/cooler/825-o-que-e-o-cooler-.htm> Acesso em 08 mai. 2017.

- JUSTEN, Álvaro. **Curso de Arduino**. 2011. Disponível em: <http://cursodearduino.com.br/apostila/apostila-rev4.pdf>. Acesso em 04 dez. 2016.

- KOZAK, Dalton Vinicius. **Conceitos Básicos de Informática**. PUC-PR, 2002. Disponível em: <https://chasqueweb.ufrgs.br/~paul.fisher/apostilas/inform/Conceitos.Basicos.da.Informatica.PDF>. Acesso em 04 dez. 2016.

- LIMA, Anderson dos Santos. **Eletrônica (Definição)**. 2016. Disponível em <http://andersonlima.vlog.br/eletronica-2/>. Acesso em 19 out. 2016.

- LOPES, Leonardo. **Uma avaliação da tecnologia LED na iluminação pública.** Politécnica, 2014. Disponível em: <http://monografias.poli.ufrj.br/monografias/monopoli10010665.pdf>, Acesso em 05 abr. 2017.

- MURATORI, José; DAL BÓ, Paulo. **Automação residencial: histórico, definições e conceitos,** 2017. Disponível em: <http://www.osetoreletrico.com.br/web/documentos/fasciculos/Ed62_fasc_automacao_capI.pdf> Acesso em 19 mai. 2017

- NASCIMENTO, Edmar José dos. **Introdução às Redes de Computadores**. Universidade Federal do Vale do São Francisco, 2011. Disponível em: <http://www.univasf.edu.br/~edmar.nascimento/redes/redes_20112_aula02.pdf>. Acesso em 04 dez. 2016.

- PEREZ, Anderson Luiz Fernandes; DARÓS, Renan Rocha. **Oficina de Robótica:** Programação em Arduino – Módulo Básico. 2013. Disponível em <http://oficinaderobotica.ufsc.br/files/2013/04/Programa%C3%A7%C3%A3o-em-Arduino-M%C3%B3dulo-B%C3%A1sico.pdf>. Acesso em 04 dez. 2016.

- ROVERI, Michael. **Automação residencial.** Faculdade Politec, 2012. Disponível em: <http://www.mariolb.com.br/blog/_static/TCC/TCC-AutomacaoResidencial-MichaelRoveri-2012.pdf> Acesso em 04 abr. 2017.

- SCOPACASA, Vicente. **Introdução à tecnologia LED.** Disponível em: <http://www.lumearquitetura.com.br/pdf/LA_Pro1/02%20%20pro_leds_Vis%C3%A3o_Geral.pdf> Acesso em 05 abr. 2017.

- SERVO LIBRARY. **Referências Arduino.** Disponível em: <https://www.arduino.cc/en/reference/servo> Acesso em 15 abr. 2017.

- SHAW, David. **INKSCAPE ADVENTURES INDEX**, 2006. Disponível em: <http://members.gamedev.net/trapperzoid/ia/images/inkscape_main_screen_diagram.jpg> Acesso em: 04 jun. 2017

- SILVA, Bruna. **Sistema de automação residencial de baixo custo para redes sem fio.** UFRS, 2015. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/101188/000931903.pdf> Acesso em 03 abr. 2017.

- SPI LIBRARY. **Referências Arduino.** Disponível em: < https://www.arduino.cc/en/reference/SPI> Acesso em 15 abr. 2017.

- S. SANTOS, Sylvio. **Introdução ao App Inventor.** PUC Minas. Disponível em: <http://www.dai.ifma.edu.br/~mlcsilva/aulassdist/Tutorial%20do%20App%20Inventor.pdf> Acesso em 03 fev. 2017

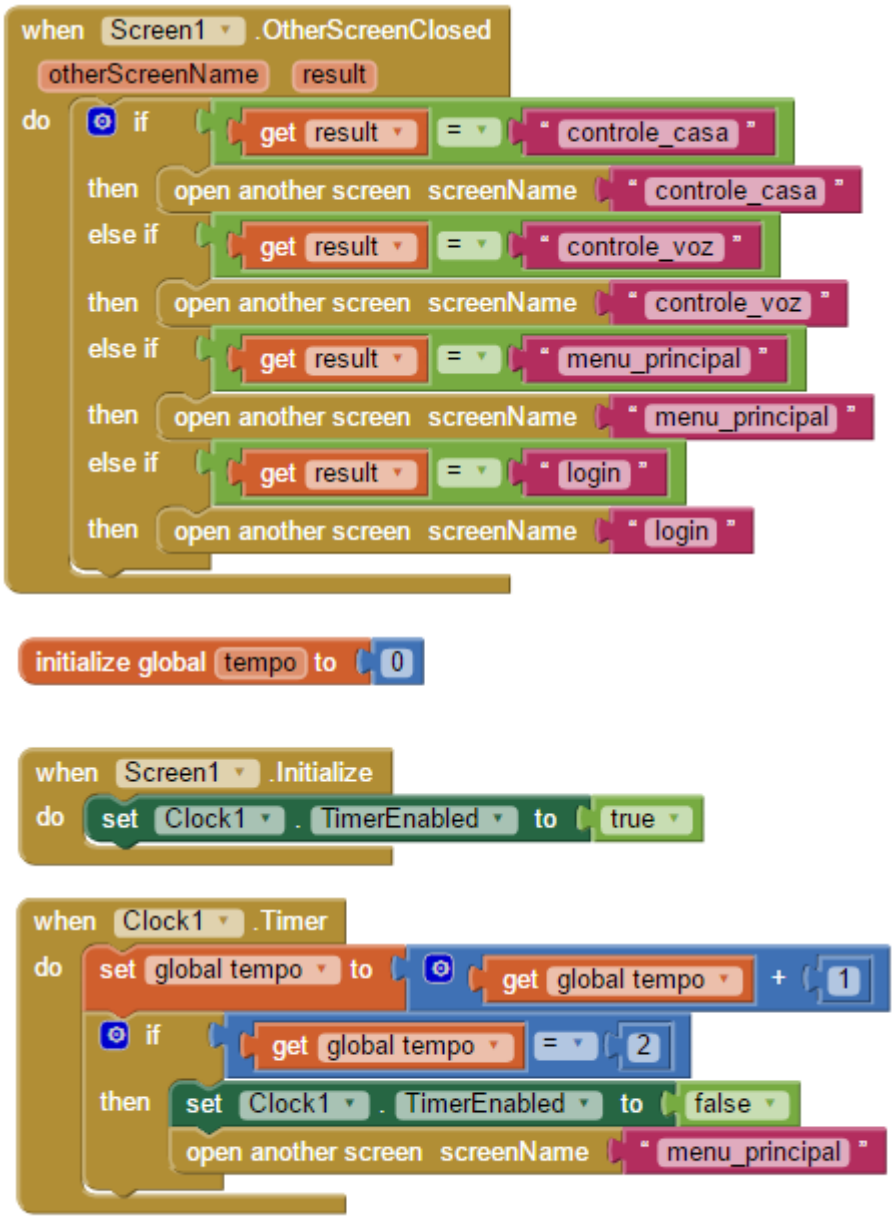
- TP-LINK. **Roteador Wireless N 300Mbps TL-WR841N**. Disponível em: <http://www.tp-link.com.br/products/details/TL-WR841N.html> Acesso em 04 jun. 2017

- ZANETTI, et. al. **Projeto No. 12 – Servo motor.** Clube de Arduino, Fatec Jundiaí. Disponível em: <http://fatecjd.edu.br/site/uploads/files/Projeto-12.pdf> Acesso em 06 abr. 2017.

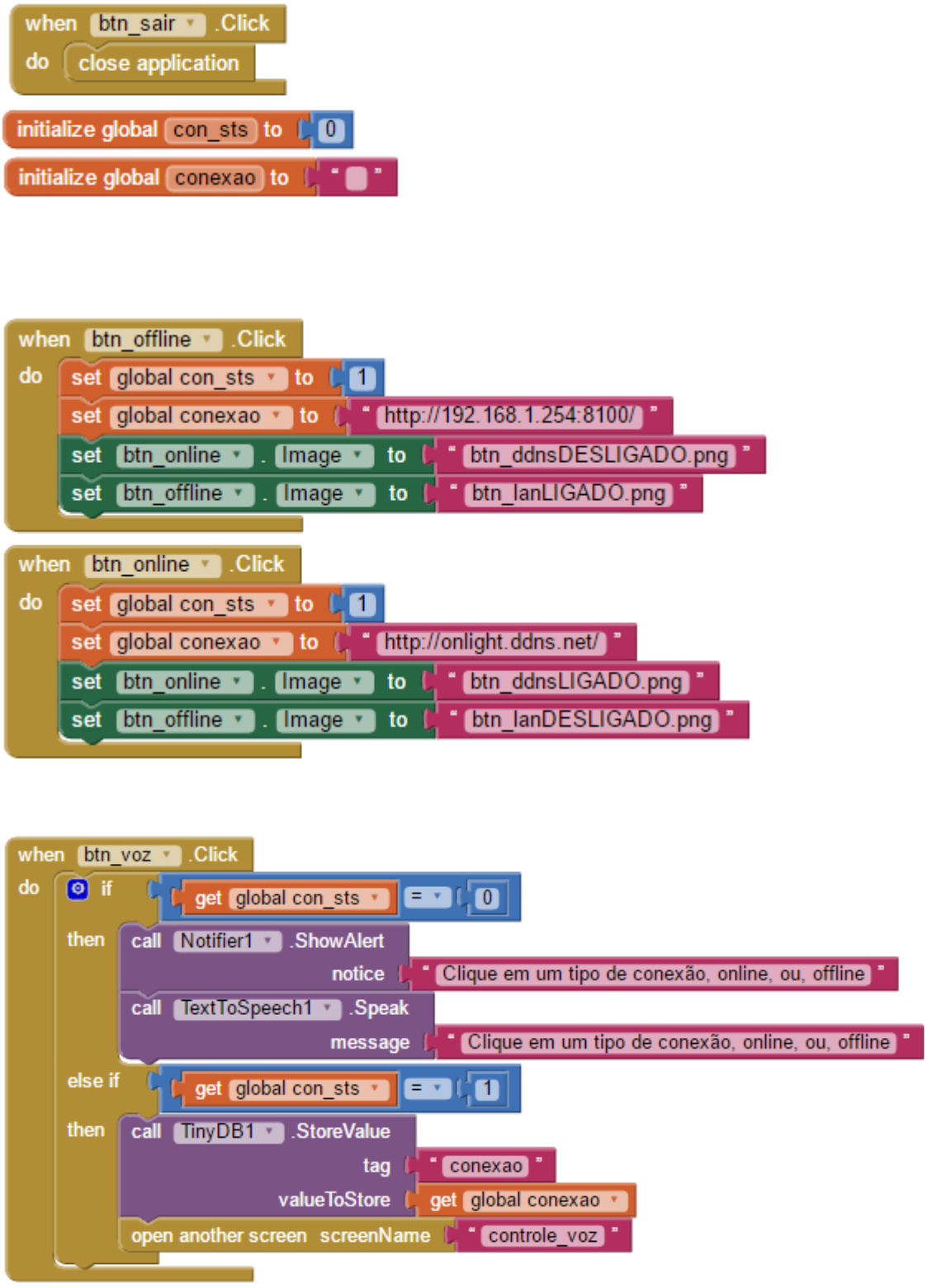
APÊNDICE

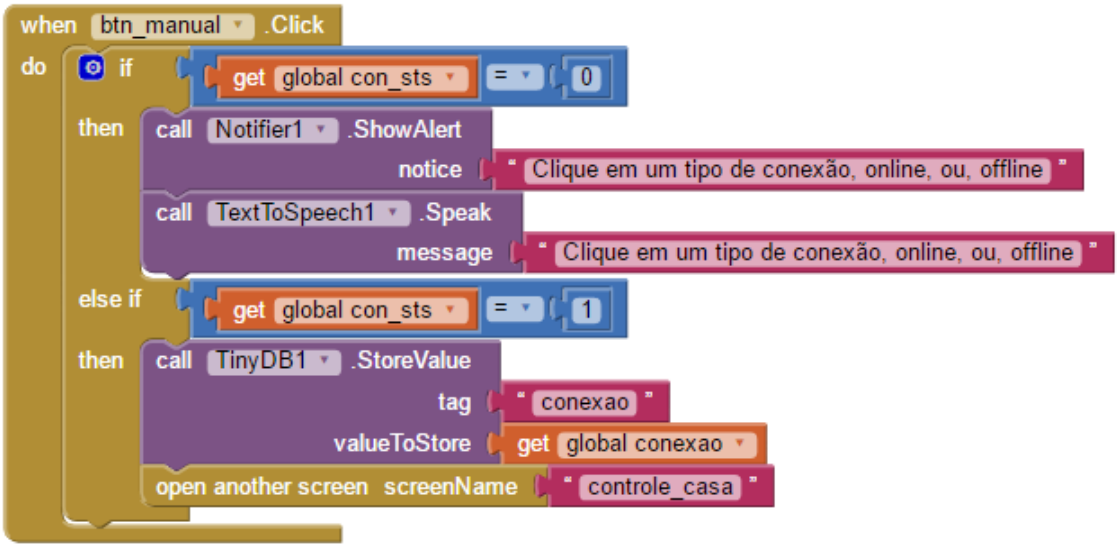
Screen 1

// https://github.com/augustoliks/App-Inventor/tree/master/Tcc%20Redes%20-%20onLight

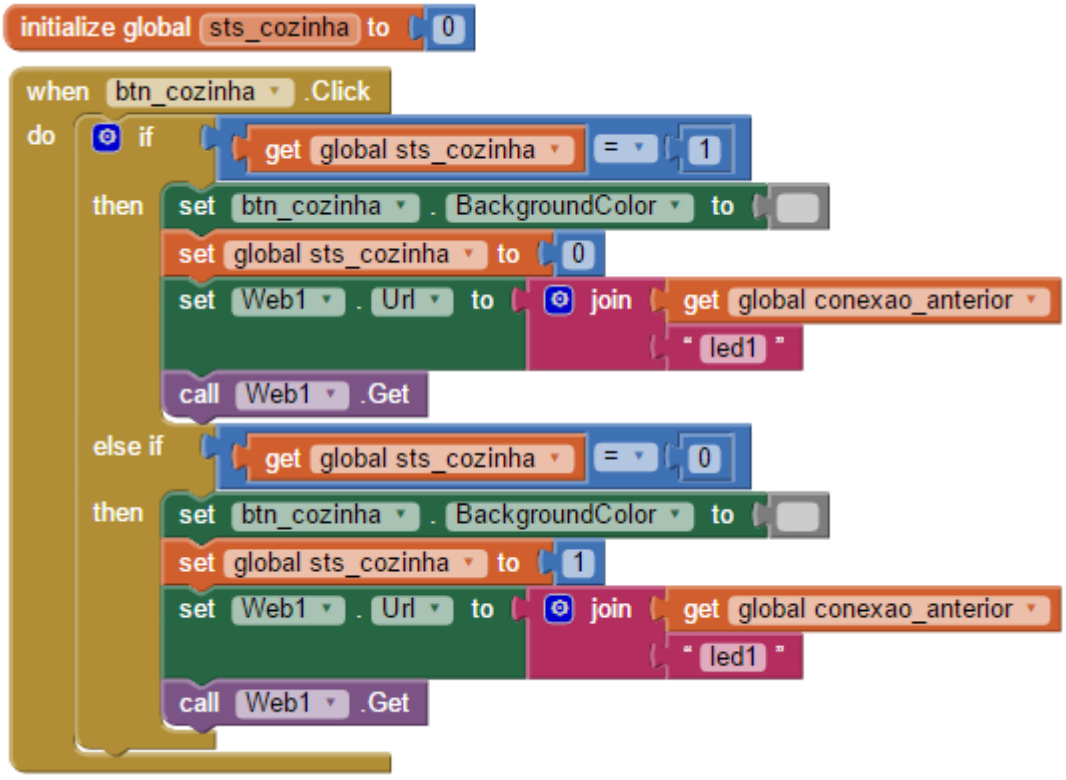


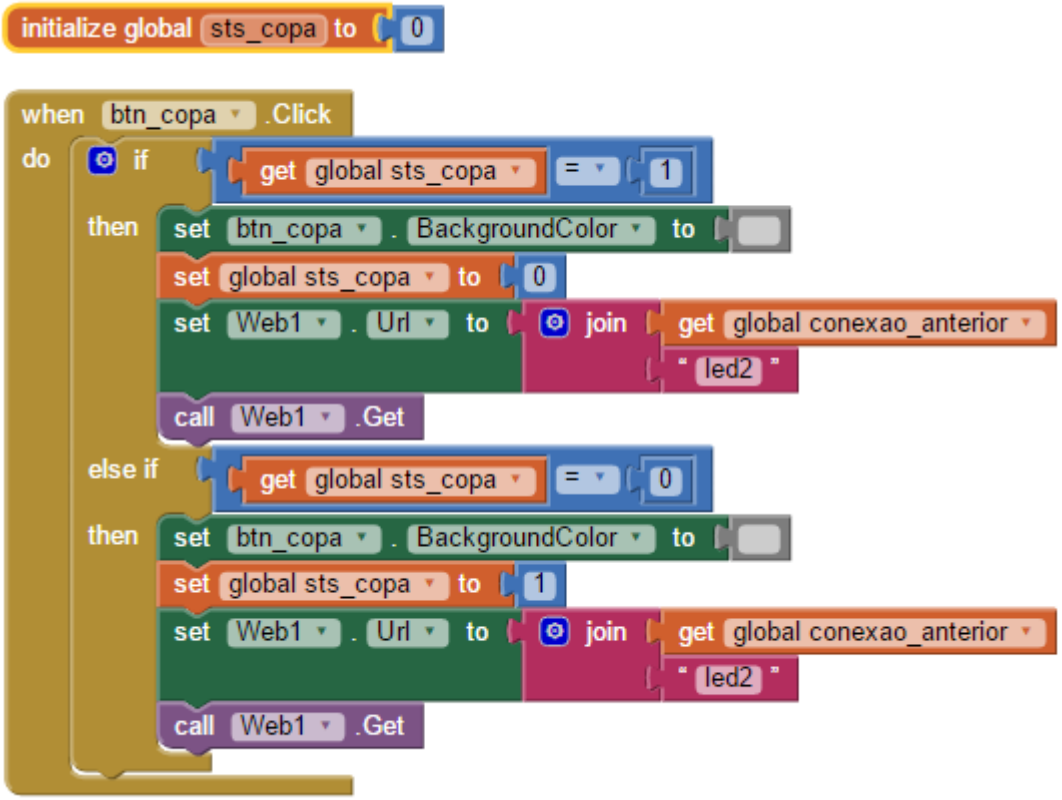
Screen 2

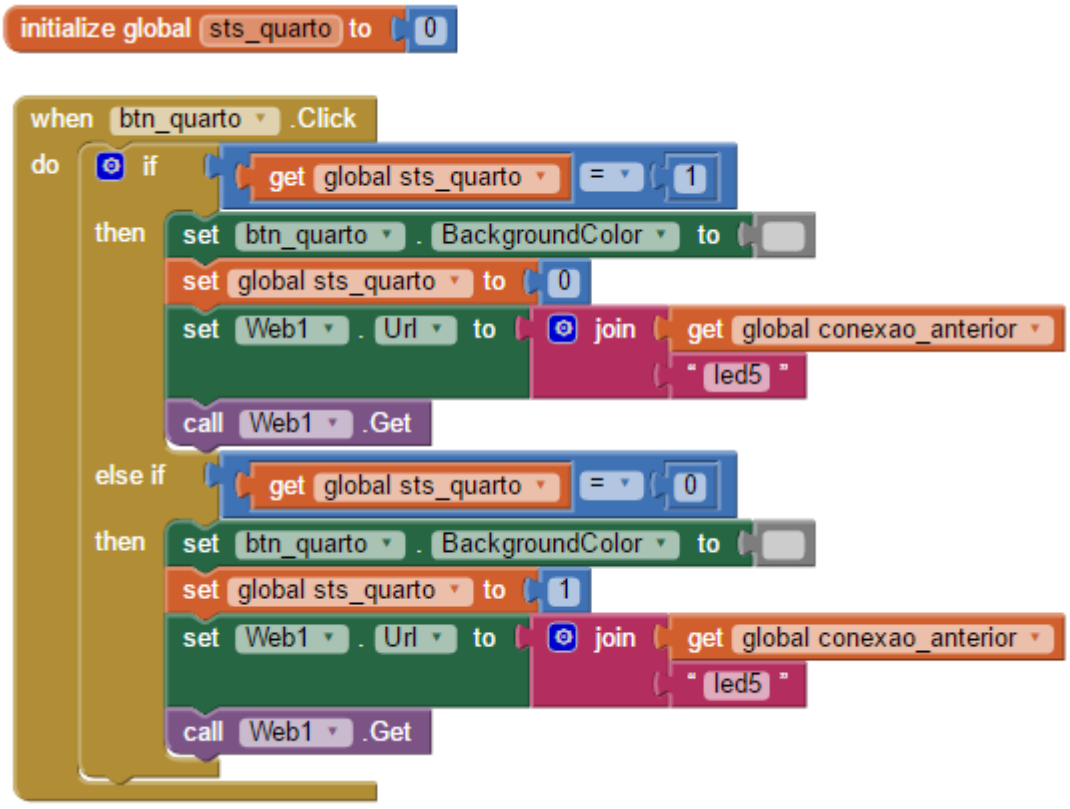

when btn_sair .Click
do close application
initialize global con_sts to 0
initialize global conexao to
when btn_offline .Click
do set global con_sts to 1
set global conexao to http://192.168.1.254:8100/
set btn_online . Image to btn_ddnsDESLIGADO.png
set btn_offline . Image to btn_lanLIGADO.png
when btn_online .Click
do set global con_sts to 1
set global conexao to http://onlight.ddns.net/
set btn_online . Image to btn_ddnsLIGADO.png
set btn_offline . Image to btn_lanDESLIGADO.png
when btn_voz .Click
do if get global con_sts = 0
then call Notifier1 .ShowAlert
notice Clique em um tipo de conexão, online, ou, offline
call TextToSpeech1 .Speak
message Clique em um tipo de conexão, online, ou, offline
else if get global con_sts = 1
then call TinyDB1 .StoreValue
tag conexao
valueToStore get global conexao
open another screen screenName controle_voz

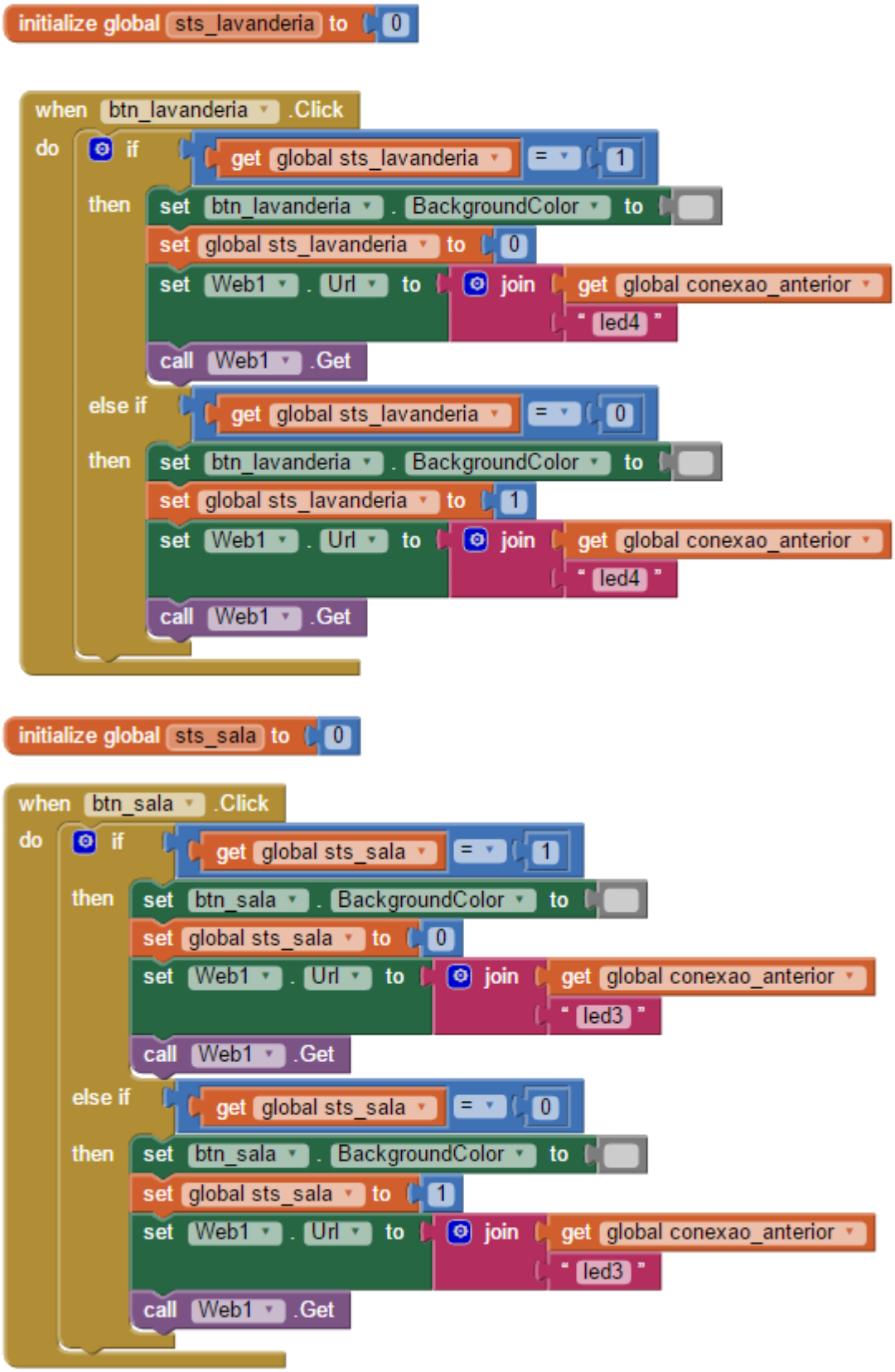

when btn_manual .Click
do if get global con_sts = 0
then call Notifier1 .ShowAlert
notice " Clique em um tipo de conexão, online, ou, offline "
call TextToSpeech1 .Speak
message " Clique em um tipo de conexão, online, ou, offline "
else if get global con_sts = 1
then call TinyDB1 .StoreValue
tag " conexao "
valueToStore get global conexao
open another screen screenName " controle_casa "

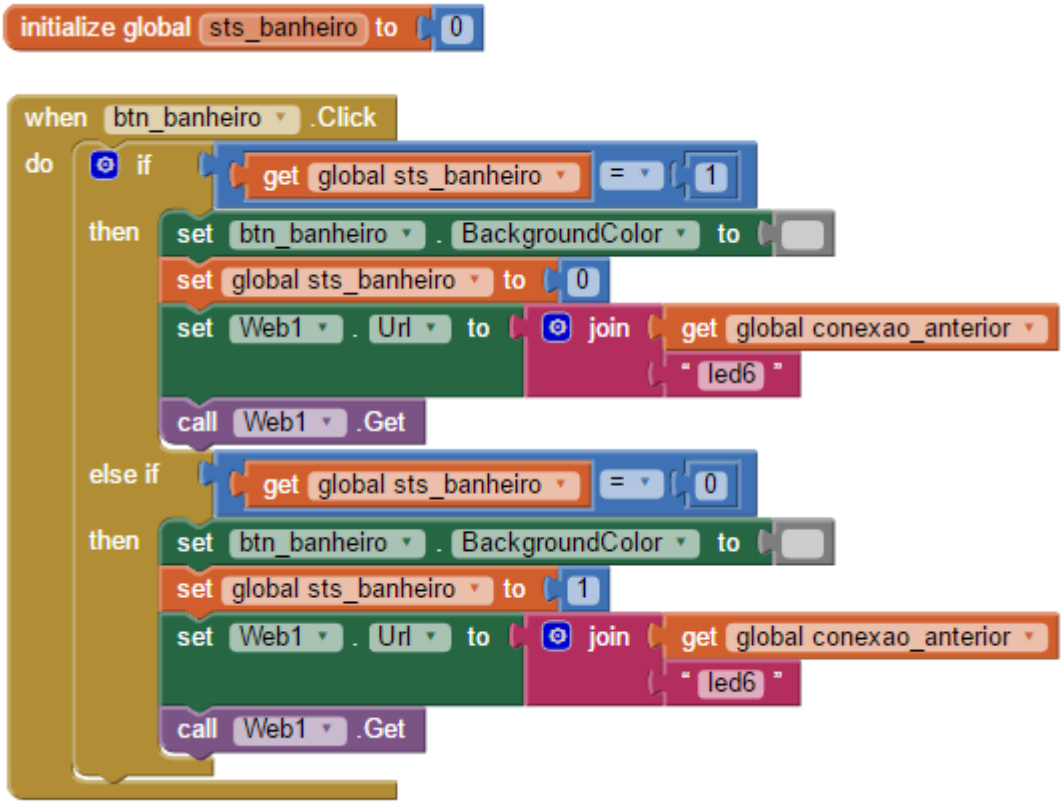
Screen 3

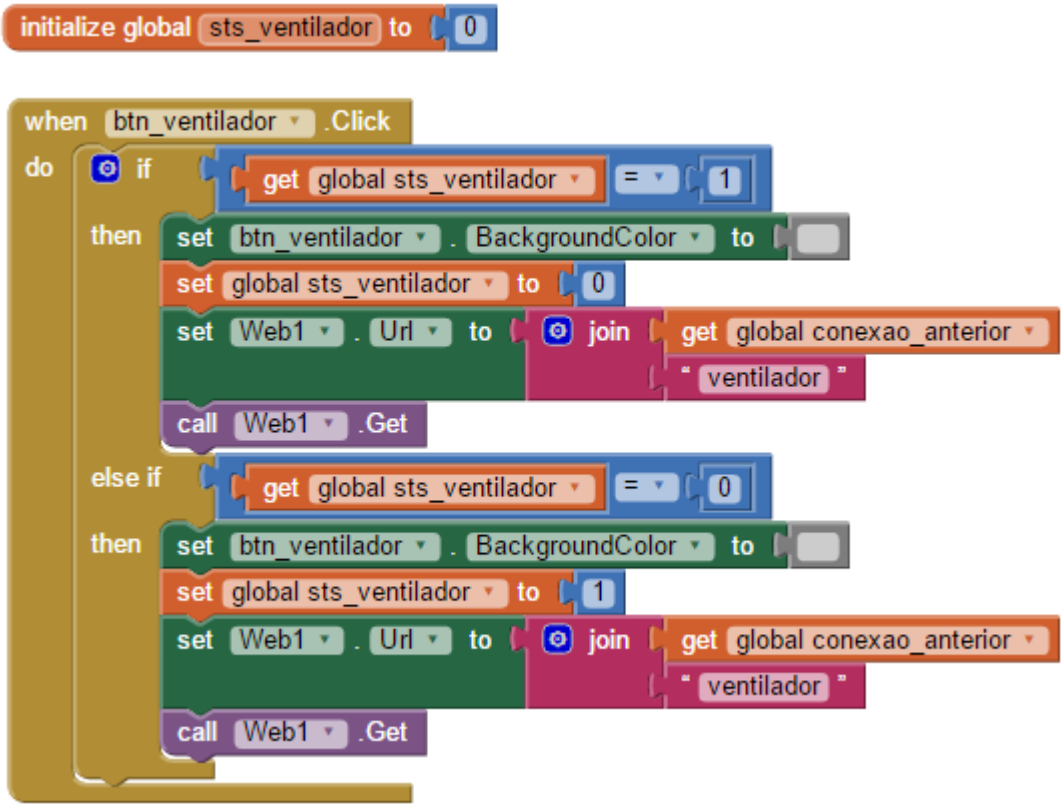

initialize global sts_cozinha to 0
when btn_cozinha .Click
do if get global sts_cozinha = 1
then set btn_cozinha . BackgroundColor to
set global sts_cozinha to 0
set Web1 . Url to join get global conexao_anterior
" led1 "
call Web1 .Get
else if get global sts_cozinha = 0
then set btn_cozinha . BackgroundColor to
set global sts_cozinha to 1
set Web1 . Url to join get global conexao_anterior
" led1 "
call Web1 .Get

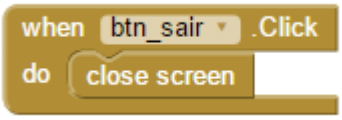

initialize global sts_copa to 0
when btn_copa .Click
do if get global sts_copa = 1
then set btn_copa . BackgroundColor to
set global sts_copa to 0
set Web1 . Url to join get global conexao_anterior
" led2 "
call Web1 .Get
else if get global sts_copa = 0
then set btn_copa . BackgroundColor to
set global sts_copa to 1
set Web1 . Url to join get global conexao_anterior
" led2 "
call Web1 .Get


initialize global sts_quarto to 0
when btn_quarto .Click
do if get global sts_quarto = 1
then set btn_quarto . BackgroundColor to
set global sts_quarto to 0
set Web1 . Url to join get global conexao_anterior
" led5 "
call Web1 .Get
else if get global sts_quarto = 0
then set btn_quarto . BackgroundColor to
set global sts_quarto to 1
set Web1 . Url to join get global conexao_anterior
" led5 "
call Web1 .Get

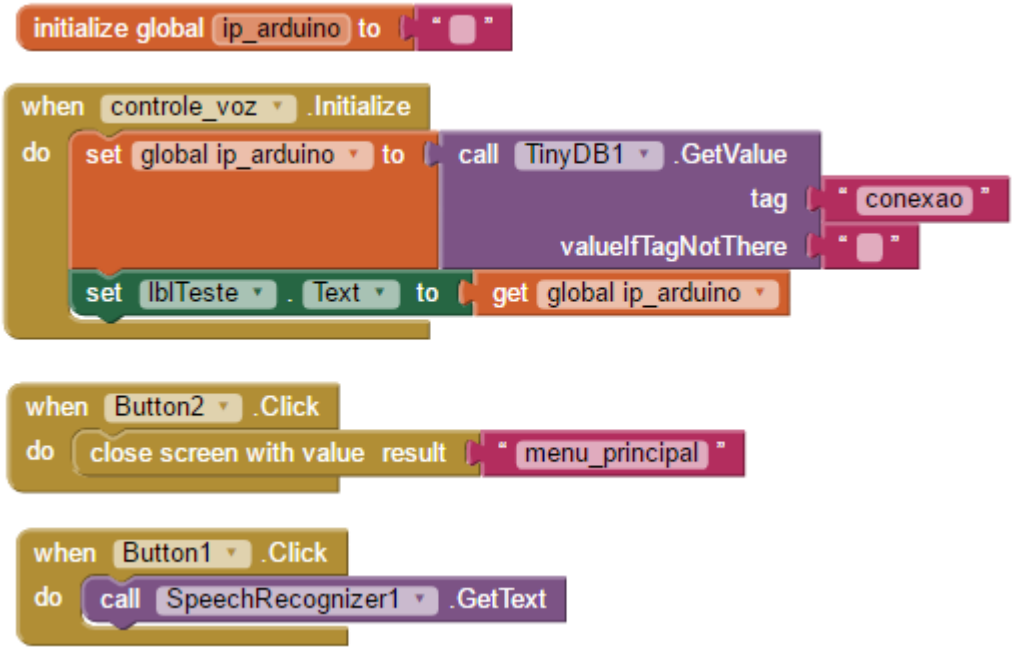

initialize global sts_lavanderia to 0
when btn_lavanderia .Click
do if get global sts_lavanderia = 1
then set btn_lavanderia . BackgroundColor to
set global sts_lavanderia to 0
set Web1 . Url to join get global conexao_anterior
" led4 "
call Web1 .Get
else if get global sts_lavanderia = 0
then set btn_lavanderia . BackgroundColor to
set global sts_lavanderia to 1
set Web1 . Url to join get global conexao_anterior
" led4 "
call Web1 .Get
initialize global sts_sala to 0
when btn_sala .Click
do if get global sts_sala = 1
then set btn_sala . BackgroundColor to
set global sts_sala to 0
set Web1 . Url to join get global conexao_anterior
" led3 "
call Web1 .Get
else if get global sts_sala = 0
then set btn_sala . BackgroundColor to
set global sts_sala to 1
set Web1 . Url to join get global conexao_anterior
" led3 "
call Web1 .Get

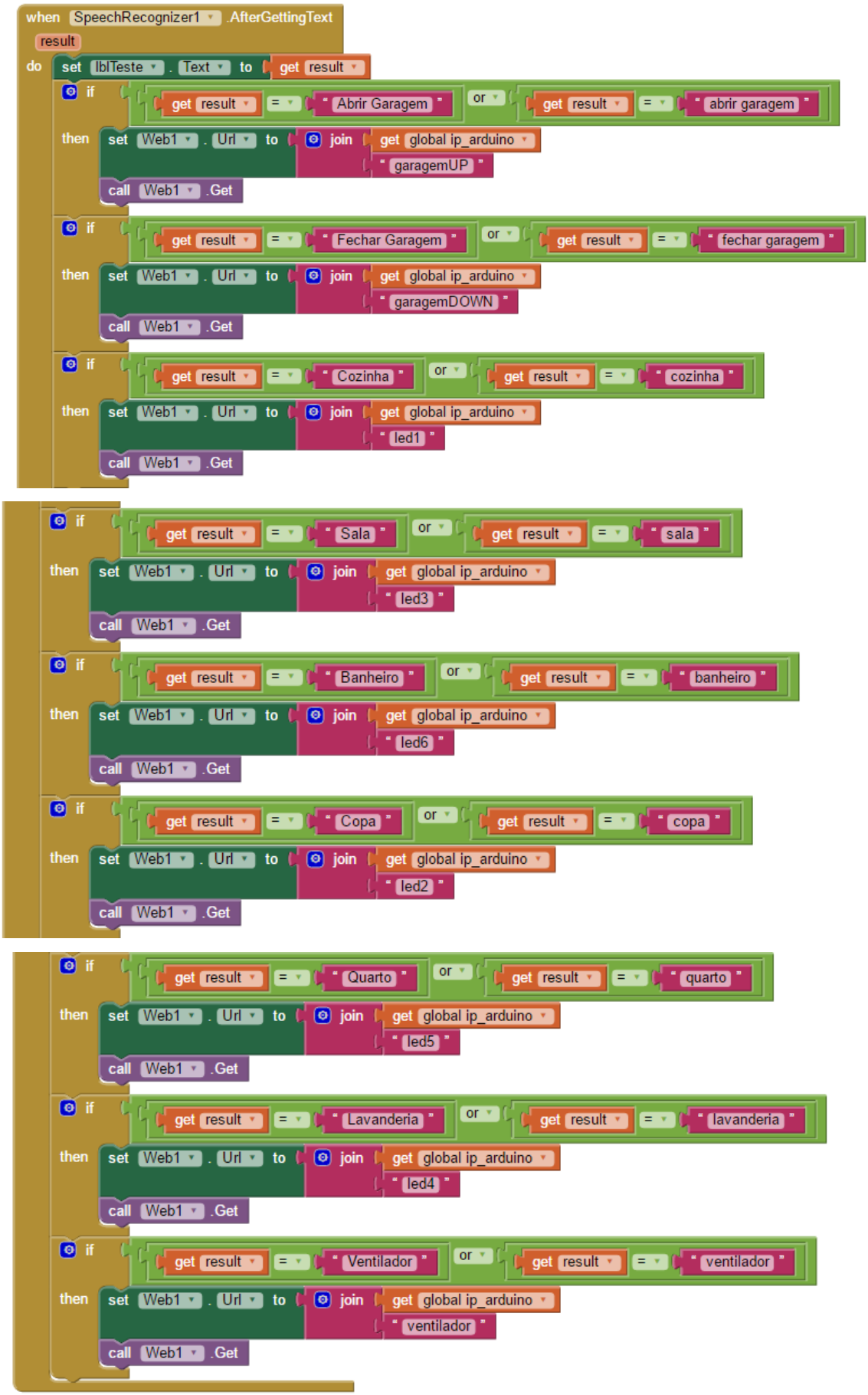

initialize global sts_banheiro to 0
when btn_banheiro .Click
do if get global sts_banheiro = 1
then set btn_banheiro . BackgroundColor to
set global sts_banheiro to 0
set Web1 . Url to join get global conexao_anterior
" led6 "
call Web1 .Get
else if get global sts_banheiro = 0
then set btn_banheiro . BackgroundColor to
set global sts_banheiro to 1
set Web1 . Url to join get global conexao_anterior
" led6 "
call Web1 .Get


initialize global sts_ventilador to 0
when btn_ventilador .Click
do if get global sts_ventilador = 1
then set btn_ventilador . BackgroundColor to
set global sts_ventilador to 0
set Web1 . Url to join get global conexao_anterior
" ventilador "
call Web1 .Get
else if get global sts_ventilador = 0
then set btn_ventilador . BackgroundColor to
set global sts_ventilador to 1
set Web1 . Url to join get global conexao_anterior
" ventilador "
call Web1 .Get


when btn_sair .Click
do close screen

Screen 4


initialize global ip_arduino to " "
when controle_voz .Initialize
do set global ip_arduino to call TinyDB1 .GetValue
tag " conexao "
valueIfTagNotThere " "
set lblTeste . Text to get global ip_arduino
when Button2 .Click
do close screen with value result " menu_principal "
when Button1 .Click
do call SpeechRecognizer1 .GetText


when SpeechRecognizer1 .AfterGettingText
result
do set lblTeste . Text to get result
if get result = Abrir Garagem or get result = abrir garagem
then set Web1 . Url to join get global ip_arduino garagemUP
call Web1 .Get
if get result = Fechar Garagem or get result = fechar garagem
then set Web1 . Url to join get global ip_arduino garagemDOWN
call Web1 .Get
if get result = Cozinha or get result = cozinha
then set Web1 . Url to join get global ip_arduino led1
call Web1 .Get
if get result = Sala or get result = sala
then set Web1 . Url to join get global ip_arduino led3
call Web1 .Get
if get result = Banheiro or get result = banheiro
then set Web1 . Url to join get global ip_arduino led6
call Web1 .Get
if get result = Copa or get result = copa
then set Web1 . Url to join get global ip_arduino led2
call Web1 .Get
if get result = Quarto or get result = quarto
then set Web1 . Url to join get global ip_arduino led5
call Web1 .Get
if get result = Lavanderia or get result = lavanderia
then set Web1 . Url to join get global ip_arduino led4
call Web1 .Get
if get result = Ventilador or get result = ventilador
then set Web1 . Url to join get global ip_arduino ventilador
call Web1 .Get

CÓDIGO DO ARDUINO

```
#include <SPI.h>
#include <Ethernet.h>
#include <Servo.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Mac do Arduino
byte ip[] = { 192, 168, 1, 254 }; //IP do Arduino

EthernetServer server(8100); //Selecionar porta de saída
Servo servo;

int k;
int x=0;
int led1 = 2; //Cozinha
int led2 = 3; //Copa
int led3 = 4; //Sala
int led4 = 5; //Lavanderia
int led5 = 6;   //Quarto
int led6 = 7; //Banheiro
int ventilador = 8;

String readString = String(30);
String statusLed;

void setup() {
      Ethernet.begin(mac, ip);
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
pinMode(ventilador, OUTPUT);
servo.attach(9);
}

void loop() {
servo.write(x);
EthernetClient client = server.available();

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

```
if(client.available()){
char c = client.read();


if(readString.length() < 30) {
readString += (c);
}

if(c == '\n'){

  if(readString.indexOf("led1") >= 0) {
digitalWrite(led1,!digitalRead(led1));

}

if(readString.indexOf("led2") >= 0) {
digitalWrite(led2,!digitalRead(led2));
}

if(readString.indexOf("led3") >= 0) {
digitalWrite(led3,!digitalRead(led3));
}

if(readString.indexOf("led4") >= 0) {
digitalWrite(led4,!digitalRead(led4));
}

if(readString.indexOf("led5") >= 0) {
digitalWrite(led5,!digitalRead(led5));
}

if(readString.indexOf("led6") >= 0) {
digitalWrite(led6,!digitalRead(led6));
}

if(readString.indexOf("garagemUP") >= 0) {

for (k=x ; k<180 ; k+=10){
servo.write(k);
delay(100);
}
x=k;
}

if(readString.indexOf("garagemDOWN") >= 0) {
```

```
for (k=x ; k>0 ; k-=10){
servo.write(k);
delay(100);
}
x=k;
}

if(readString.indexOf("ventilador") >= 0) {
digitalWrite(ventilador,!digitalRead(ventilador));
}

client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println();
client.println("<!doctype html>");

client.println("<html>");
client.println("<head>");
client.println("<title>onLight</title>");
client.println("<meta name=\"viewport\" content=\"width=320\">");
client.println("<meta name=\"viewport\" content=\"width=device-width\">");
client.println("<meta charset=\"utf-8\">");
client.println("<meta name=\"viewport\" content=\"initial-scale=1.0, user-
scalable=no\">");
client.println("</head>");
client.println("<body>");
client.println("<center>");

client.println("<font size=\"10\" face=\"verdana\" color=\"black\">on</font>");
client.println("<font size=\"10\" face=\"verdana\" color=\"blue\">Light</font><br />");

if(x)
client.println("Garagem ABERTA!!!");


else
client.println("Garagem FECHADA!!!");


client.println("<td> <form action=\"garagemUP\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">Abrir GARAGEM"
"</button>");
client.println("</form> <br /></td>");
```

```
client.println("<td> <form action=\"garagemDOWN\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">Fechar GARAGEM"
"</button>");
client.println("</form> <br /></td>");

if(digitalRead(led1)){
statusLed = "Ligado";

}

else{
statusLed = "Desligado";
}

client.println("<td> <form action=\"led1\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">COZINHA -
"+statusLed+"</button>");
client.println("</form> <br /></td>");

if(digitalRead(led2)) {
statusLed = "Ligado";
}

else{
statusLed = "Desligado";
}

client.println("<td> <form action=\"led2\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">COPA -
"+statusLed+"</button>");
client.println("</form> <br /></td>");

if(digitalRead(led3)){
statusLed = "Ligado" ;
}

else{
statusLed = "Desligado";
}

client.println("<td> <form action=\"led3\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">SALA -
"+statusLed+"</button>");
client.println("</form> <br /></td>");
```

```
if(digitalRead(led4)){
statusLed = "Ligado" ;
}

else{
statusLed = "Desligado";
}

client.println("<td> <form action=\"led4\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">LAVANDERIA -
"+statusLed+"</button>");
client.println("</form> <br /></td>");

if(digitalRead(led5)){
statusLed = "Ligado";
}

else{
statusLed = "Desligado";
}

client.println("<td> <form action=\"led5\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">QUARTO -
"+statusLed+"</button>");
client.println("</form> <br /></td>");

if(digitalRead(led6)) {
statusLed = "Ligado" ;
}

else{
statusLed = "Desligado";
}

client.println("<td> <form action=\"led6\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">BANHEIRO -
"+statusLed+"</button>");
client.println("</form> <br /></td>");

if(digitalRead(ventilador)) {
statusLed = "Ligado" ;
}

else{
statusLed = "Desligado";
```

```
}

client.println("<td> <form action=\"ventilador\" method=\"get\">");
client.println("<button type=submit style=\"width:200px;\">VENTILADOR -
"+statusLed+"</button>");
client.println("</form> <br /></td>");


client.println("</center>");
client.println("</body>");
client.println("</html>");

      readString = "";
client.stop();

}
}
}
}
}
```