UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CURITIBA
CURSO DE ENGENHARIA ELÉTRICA

ARTHUR AQUINO LINO DE SOUZA
GERSON SANDRO SANTOS JUNIOR
HENRIQUE KATSUMI SASAKI

DESENVOLVIMENTO DE PROTÓTIPO DE BAIXO CUSTO PARA AUTOMAÇÃO RESIDENCIAL ATRAVÉS DE ARDUINO CONTROLADA POR ANDROID

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2018

ARTHUR AQUINO LINO DE SOUZA
GERSON SANDRO SANTOS JUNIOR
HENRIQUE KATSUMI SASAKI

DESENVOLVIMENTO DE PROTÓTIPO DE BAIXO CUSTO PARA AUTOMAÇÃO RESIDENCIAL ATRAVÉS DE ARDUINO CONTROLADA POR ANDROID

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de Trabalho de Conclusão de Curso, do curso de Engenharia Elétrica do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR), como requisito parcial para obtenção do título de Engenheiro Eletricista

Orientador: Prof. Jorge Assade Leludak

CURITIBA

2018

ARTHUR AQUINO LINO DE SOUZA
GERSON SANDRO SANTOS JUNIOR
HENRIQUE KATSUMI SASAKI

DESENVOLVIMENTO DE PROTÓTIPO DE BAIXO CUSTO PARA AUTOMAÇÃO RESIDENCIAL ATRAVÉS DE ARDUINO CONTROLADA POR ANDROID

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

Curitiba, 20 de novembro de 2018.

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

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

ORIENTAÇÃO

_____
Jorge Assade Leludak, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

BANCA EXAMINADORA

_____
Alexandre José Tuoto Silveira Mello, Dr.
Universidade Tecnológica Federal do Paraná

_____
Jose Da Silva Maia, Mestre
Universidade Tecnológica Federal do Paraná

_____
Marco Antonio Busetti de Paula, Dr.
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia Elétrica

Dedicamos este trabalho aos nossos familiares que durante todo o nosso curso nos apoiaram e torceram por nossa jornada, para que fosse com êxito e obtivéssemos o sucesso.

Também dedicamos ao nosso mestre e orientador Jorge Assade Leludak, por sua competência e atenção.

A educação é a arma mais poderosa que você pode usar para mudar o mundo.
Nelson Mandela

RESUMO

SASAKI, H. K.; JUNIOR, G. S. S.; SOUZA, A. A. L. de. **Desenvolvimento de Protótipo de Baixo Custo para Automação Residencial através de Arduino Controlada por Android**. 2018. 79f. Trabalho de Conclusão de Curso – Curso de Engenharia Elétrica, Universidade Tecnológica Federal do Paraná, Curitiba, 2018.

Este trabalho contempla o desenvolvimento de um protótipo de sistema de automação residencial com baixo custo, tendo como central de automação o Arduino UNO. Para a simulação de controle de alguns processos de uma residência, tais como abertura e fechamento de portão, controle de iluminação e de temperatura, foi desenvolvido um aplicativo para sistema operacional Android através da plataforma App Inventor, onde as cargas citadas são acionadas em uma maquete para ilustração e apresentação de resultados. O Arduino foi conectado a um roteador de borda, e as cargas foram conectadas a esta placa através de fios. Com o IP deste dispositivo foi possível enviar comandos mesmo não estando conectado a mesma rede, ou seja, pode-se controlar a maquete de qualquer lugar que possua IPv6 válido e conexão com a Internet. O protótipo apresentou um desempenho satisfatório, embora o tempo de resposta (do comando do celular até a execução no Arduino) variou muito e o custo total do projeto foi baixo.

Palavras-chave: Android, App Inventor, Arduino UNO, automação, controle, residencial.

ABSTRACT

SASAKI, H. K.; JUNIOR, G. S. S.; SOUZA, A. A. L. de. **Development of a Low Cost Prototype for Home Automation through Arduino Controlled by Android**. 2018. 79f. Trabalho de Conclusão de Curso – Curso de Engenharia Elétrica, Universidade Tecnológica Federal do Paraná, Curitiba, 2018.

The present work is about the development of a low-cost home automation prototype, with the use of Arduino Uno as the automation core. To simulate the control of a few processes in a residence, like the opening and closing of a gate, lighting and temperature control, an application was developed for the operational system Android through the App Inventor platform, where the loads above will be triggered in a model for the illustration and representation of the results. The Arduino was connected to an edge router, and the loads were connected to this has board by cables. With the IP of this device it was possible to send commands even when not connected to the same network, in other words, the model can be controlled from anywhere that has a valid IPv6 and connection with the Internet. The prototype presented a satisfactory performance, even though the response time (from the cellphone's command to the execution in the Arduino) varied a lot and the project's total cost was low.

Keywords: Android, App Inventor, Arduino UNO, automation, control, home.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELA

LISTA DE SIGLAS

ANEEL – Agência Nacional de Energia Elétrica

FFD (Full Function Device)

IANA (Internet Assigned Numbers Authority)

IEEE (Institute of Electrical and Electronics Engineers)

IP (Internet Protocol)

ISDN (Rede Digital de Serviços Integrados)

ISP (Internet Service Provider)

LAN (Local Area Network)

MAC (Media Access Control)

MIT (Massachusetts Institute of Technology)

NA (Normal Aberto)

NAHB (National Association of Home Builders)

NIC (Network Interface Card)

NTC (Negative Temperature Coeficient)

P2P (Peer-to-Peer)

PAN (Personal Area Network)

PLC (Power Line Carrier)

V_{in} (Tensão Nominal)

RFD (Reduced-Function Device)

WAN (Wide Area Network)

SUMÁRIO

1 INTRODUÇÃO

1.1 TEMA

A automação residencial começou nos Estados Unidos, na década de 70. O comando era feito por PLC (*Power Line Carrier*), era um sistema bem simples para resolver problemas pontuais, como acender as luzes ou ligar um equipamento a distância (Muratori, 2011).

Em 1984 o interesse em automação residencial aumentou o suficiente para que a associação comercial NAHB (*National Association of Home Builders*) criassem um novo conceito chamado *Smart House*, com o objetivo de incluir tecnologias no projeto de novas casas (HARPER, 2003).

Fibra óptica, comutação digital, Rede Digital de Serviços Integrados (ISDN) e internet foram algumas das novas tecnologias que surgiram na época para uso doméstico (BARLOW; GANN, 1998).

Casas inteligentes, conhecidas também como casas automatizadas ou domótica, são lares que possuem dispositivos que conseguem monitorar e ligar/desligar sistemas como aquecimento e iluminação, por exemplo. As tecnologias empregadas em domótica permitem também a comunicação entre esses vários sistemas (RICQUEBOURG, V. et al. 2006).

Segundo Muratori, a automação residencial é um conjunto de sistemas integrados capazes de proporcionar segurança, conforto e comunicação ao usuário. Sendo esses sistemas divididos em:

- Instalação elétrica: iluminação, persianas, cortinas, gestão de energia;
- Sistema de segurança: alarmes, circuito fechado de TV, controle de acesso;
- Sistema de multimídia: áudio, vídeo, som ambiente, jogos eletrônicos;
- Sistema de comunicações: TV, telefonia, interfonia, redes domésticas, TV por assinatura.
- Utilidades: aquecimento, bombas, irrigação, etc.

1.2 DELIMITAÇÃO DO TEMA

O presente trabalho irá desenvolver um protótipo de baixo custo para automação residencial através de Arduino controlada por *Android*. Um aplicativo para Android será criado para o controle das cargas: porta, portão, leds e sensor de temperatura. Um Arduino UNO será programado para receber e executar os comandos do aplicativo. O Arduino Ethernet

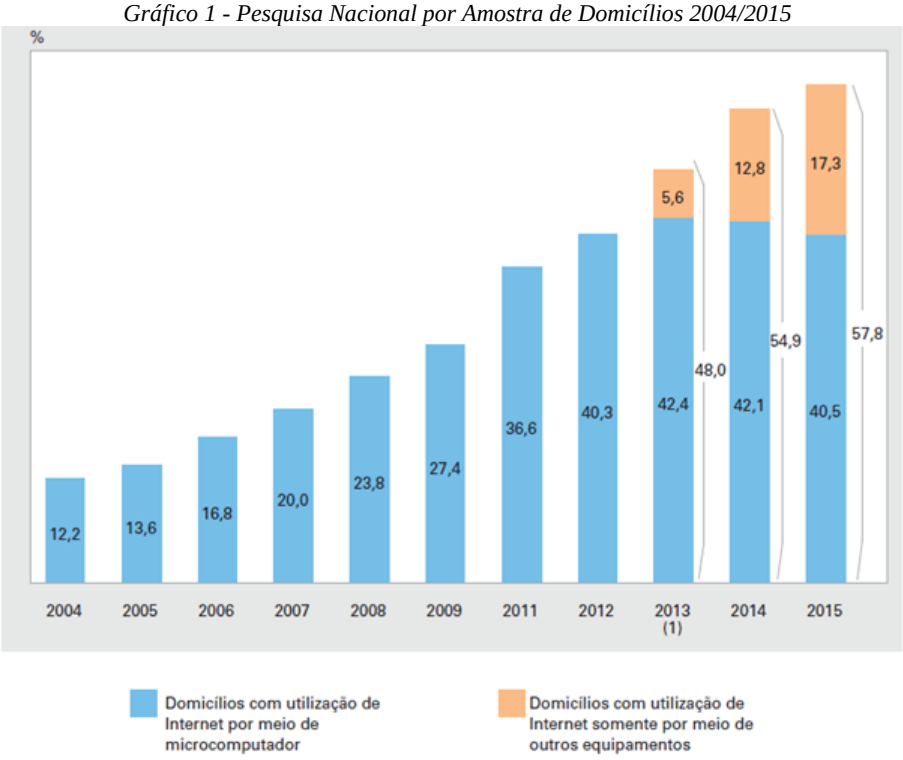
Shield e o celular utilizarão a rede local de um roteador para comunicação.

1.3 PROBLEMA E PREMISSAS

De acordo com Jamille Niero, a pesquisa feita pela Associação Brasileira de Automação Residencial e Predial (Aureside), em 2017, mostrou que a crise econômica brasileira afetou pouco a automação residencial, das 150 empresas consultadas: metade não teve mudança no nível de negócios em relação ao ano de 2016, 25% tiveram aumento e o restante sofreu redução. Porém, isso não significa que o mercado brasileiro para este segmento não tenha espaço para crescimento, muito pelo contrário, conforme o diretor executivo da Aureside, somente 3% das residências brasileiras são automatizadas, já nos Estados Unidos e Europa essa quantia varia entre 20% e 25%.

Os resultados da Pesquisa Nacional por Amostra de Domicílios, PNAD, de 2015 mostra que o número de domicílios particulares permanentes que possuem acesso à internet e pessoas detentoras de celulares têm aumentado de forma significativa a cada ano, portanto o número de casas que podem ser controladas através da internet está crescendo.

Gráfico 1 - Pesquisa Nacional por Amostra de Domicílios 2004/2015



Fonte: IBGE, Diretoria de Pesquisas, Coordenação de Trabalho e Rendimento,

Assim, pode-se ver que a automação residencial no Brasil possui um grande potencial

de crescimento, embora este esteja limitado a classe média alta e alta devido ao alto custo. Com este Trabalho de Conclusão de Curso, iremos desenvolver um protótipo que demonstre a execução de um projeto com baixo custo.

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Projetar uma instalação de automação residencial de baixo custo, com a utilização da plataforma do Arduino e integração com telefonia móvel utilizando o sistema operacional *Android*.

1.4.2 Objetivos Específicos

1. Elaborar um estudo sobre a existência dos sistemas de controle e automação residencial;
2. Realizar pesquisa sobre as vantagens dos diferentes sistemas de controle sem fio existente no mercado;
3. Fazer uma análise do custo benefício do controlador que será aplicado neste trabalho;
4. Apresentar as características técnicas do Arduino UNO;
5. Estudar linguagem de programação para o sistema operacional de *smartphone* Android;
6. Comprar maquete do modelo de um de residência padrão, que represente uma casa de 100 m²;
7. Montar e implementar os sistemas de controle na maquete (portão, portas, janelas, iluminação, ventilação e exaustão).

1.5 JUSTIFICATIVA

Fica mais evidente a cada dia a dependência das pessoas dos sistemas de comunicação via rede de dados, é possível perceber que há um enorme número de adeptos as facilidades e comodidades que um *smartphone* pode trazer. “[...] em 2018, a projeção de densidade de dispositivos conectados à internet será de dois por habitante” (MEIRELLES, 2016).

Pretende-se então aproveitar desta grande onda tecnológica e facilidade de acesso à internet em dispositivos para integrar mais uma tecnologia para o usuário, vinculado ao conforto e baixo custo de desenvolvimento.

É possível perceber o quão benéfico é se ter o controle de acessos, a possibilidade de dar este acesso remotamente a sua residência a um prestador de serviço ou lembrar se trancou a porta dos fundos simplesmente acessando o aplicativo. Estas tarefas são possíveis de ser implementada por causa das redes de comunicação. O presente trabalho abordará tópicos referentes as redes de comunicação, sistemas microcontrolado e instalação elétrica (FOROUZAN, 2012).

1.6 PROCEDIMENTOS METODOLÓGICOS

O presente TCC terá como foco a pesquisa e o protótipo desenvolvido para simulação.

Para a pesquisa seguiremos com os estudos direcionados ao custo x benefício com foco nas aplicações que mais agregam satisfação ao cliente no conjunto imensurável de aparelhos que se podem automatizar, sem deixar de limitar ao baixo custo de produção do produto. Ainda na parte da pesquisa está incluso o estudo e desenvolvimento de programação e comunicação de um aplicativo para Android que possa fazer toda a comunicação entre a casa e o *smartphone,* com uso da internet.

No protótipo será implementado os itens abordados na pesquisa bem como a implementação de todo o sistema de automação necessário obter os resultados satisfatórios próximos aos de uma casa real, utilizando uma maquete para demonstração.

2 REVISÃO BIBLIOGÁFICA

Serão apresentados alguns protocolos de rede usado no mundo para a comunicação sem fio, assim como as características do microcontrolador Arduino UNO, histórico, principais vantagens e outras informações referente ao equipamento que será utilizado. Também será visto a questão da plataforma adequada para a criação do aplicativo que será implementado para realizar os comandos.

Serão também apresentados definições e conceitos sobre as diferentes formas de rede e os protocolos que são usados nos dias de hoje.

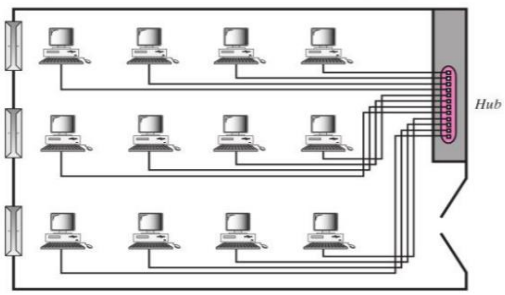
2.1 REDE LOCAL E ETHERNET

O presente item irá abordar definições relacionadas a comunicação de dados, uma rede pode ser comparada como uma autoestrada, pela qual os dados podem trafegar, a internet é uma interligação entre as redes e os protocolos estabelecem os padrões de comunicação. A comunicação é o compartilhamento de informações, que pode ser local ou remota. A comunicação de dados se dá quando há a troca de informações entre dois dispositivos por algum meio de transmissão, por exemplo cabos (FOROUZAN, 2007).

A definição de rede de Forouzan é dada como um conjunto de dispositivos conectados por *links* de comunicação. Estes dispositivos podem ser um computador, impressora ou qualquer dispositivo que esteja conectado à rede desde que esteja enviando ou recebendo dados. Basicamente existem dois tipos de rede, as locais e as de longo alcance.

Uma rede local é também conhecida como LAN (*Local Area Network*), nada mais é do que um grupo de dispositivos conectados entre si, pode ser muito simples com dois computadores em um escritório pessoal como também se estender por uma empresa inteira conectando vários periféricos (FOROUZAN, 2007).

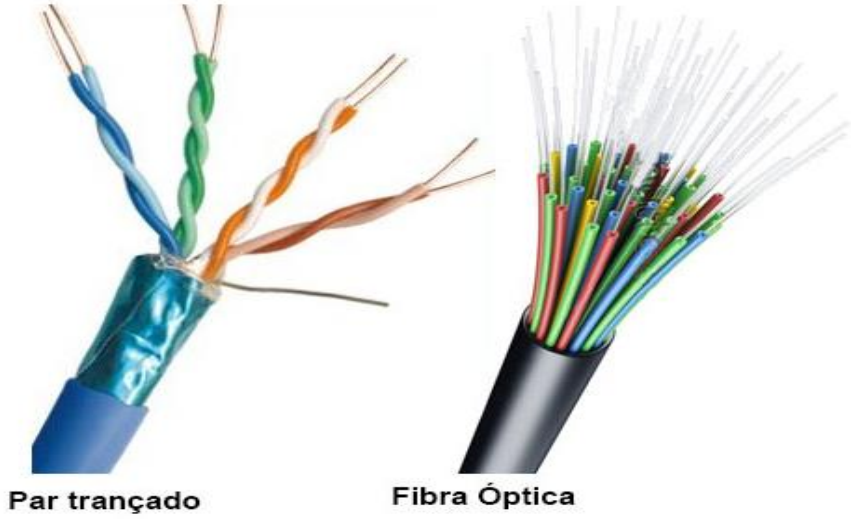
Figura 1 - LAN isolado com 12 computadores conectados a um HUB (concentrador)



Fonte: Forouzan, 2007

Além do seu tamanho a rede LAN se distingue por seu meio de transmissão e topologia. O meio de transmissão geralmente é somente um e as topologias poderão ser: barramento, anel e estrela. A tecnologia de LAN mais utilizada é a Ethernet, que utiliza a topologia em estrela, na qual cada dispositivo (nó) está conectado a outro através de equipamentos ativo de rede, tais como *switches*. A forma de transmissão usada por uma LAN geralmente inclui cabos, podendo ser cabos de par trançado ou fibra ótica. Um cabo de par trançado consiste em 4 pares de fio de cobre trançados, e é usado em seus terminais plug do RJ-45. O comprimento máximo de um cabo de par trançado é 100 m (328 pés), ao passo que o comprimento máximo dos cabos de fibra pode variar de 10 a 20 km, dependendo do tipo de fibra. Dependendo do tipo de cabo (par trançado ou fibra óptica) usado, a atual velocidade de transmissão de dados pode variar de 100 Mbit/s a 10.000 Mbit/s (FOROUZAN, 2007).

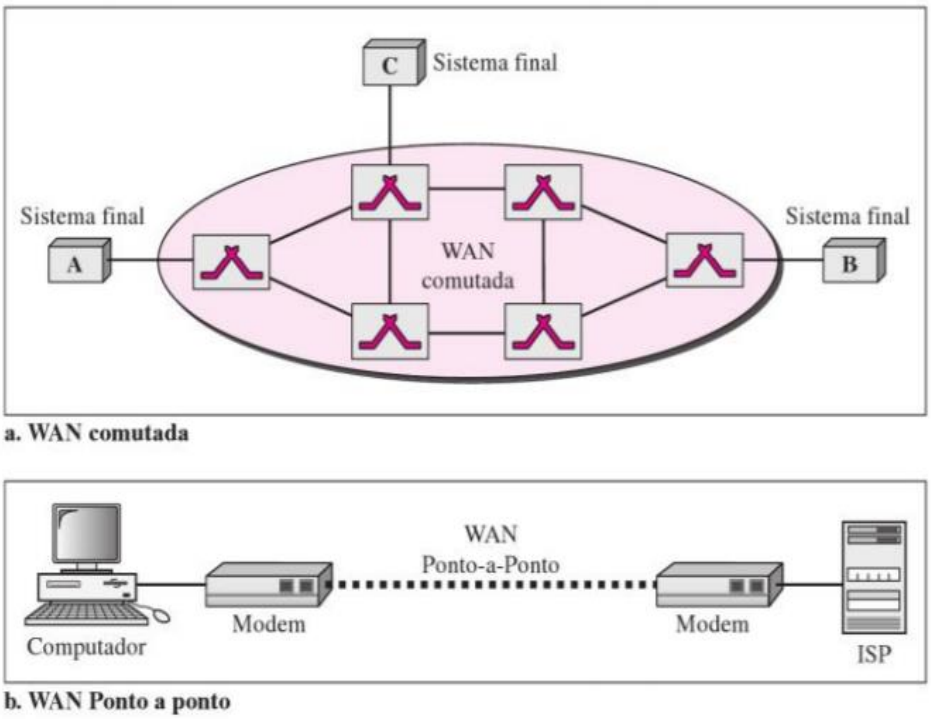
Figura 2 - Meios de transmissão de uma rede LAN



Fonte: os autores

A rede de longo alcance ou ampla abrangência é também denominada rede WAN (*Wide Área Network*), possibilita a transmissão de dados, imagens, áudio e vídeo a longas distâncias, que pode compreender um país, um continente e até mesmo o mundo todo. Uma rede WAN pode ser tão complexa como o caso do *backbone* que interliga a Internet, denominada WAN comutada, ou simples como a linha telefônica que conecta um computador pessoal à Internet está denominada WAN ponto a ponto. A WAN comutada conecta os sistemas finais, geralmente, é formada por um roteador que se conecta a outra LAN ou WAN. A WAN ponto a ponto é constituída por uma linha alugada de uma companhia telefônica ou de uma operadora de TV que conecta o computador de casa ou uma pequena LAN a um provedor de serviços de Internet (FOROUZAN, 2007).

Figura 3 - Rede WAN: uma rede WAN comutada e uma rede WAN ponto a ponto



Fonte: Forouzan, 2007

2.2 ATRIBUTOS DAS ESTRUTURAS FÍSICAS.

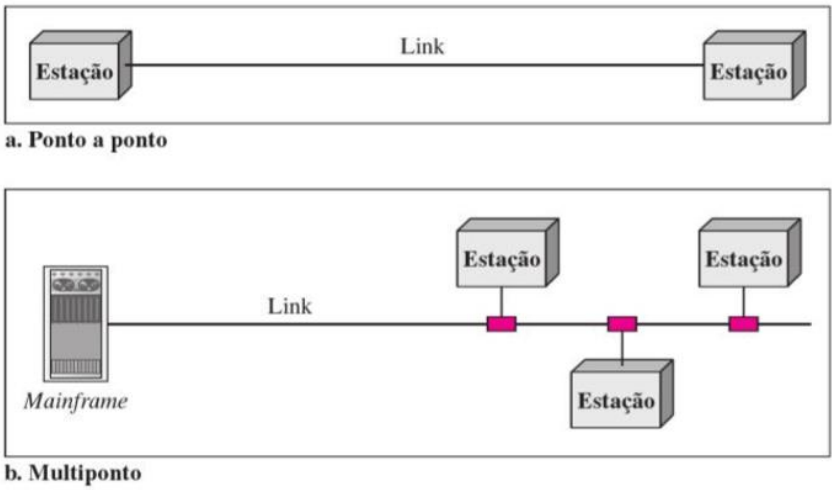
2.2.1 Tipos de conexão

Rede são dois ou mais dispositivos conectados através um caminho de comunicação que transfere dados de um dispositivo a outro. Existem duas maneiras de conexão: ponto a ponto e multiponto (FOROUZAN, 2007).

Uma conexão ponto a ponto fornece um caminho dedicado entre dois dispositivos, toda a capacidade deste caminho está reservada para a transmissão entre os dois dispositivos. A maioria dos sistemas ponto a ponto utilizam cabeamento, mas também há outras formas, como infravermelho, satélite, micro-ondas etc. (FOROUZAN, 2007).

Uma conexão multiponto também é chamada de *multidrop*, é uma conexão na qual mais de dois dispositivos compartilham o mesmo *link*, entenda-se caminho. Neste tipo de conexão a capacidade do canal é compartilhada, seja no espaço ou tempo. Se diversos usuários puderem utilizar este tipo de conexão ao mesmo possui uma conexão compartilhada espacialmente, se tiverem que revezar entre si, chama de compartilhada no tempo (FOROUZAN, 2007).

Figura 4 - Tipos de conexões: ponto a ponto e multiponto.



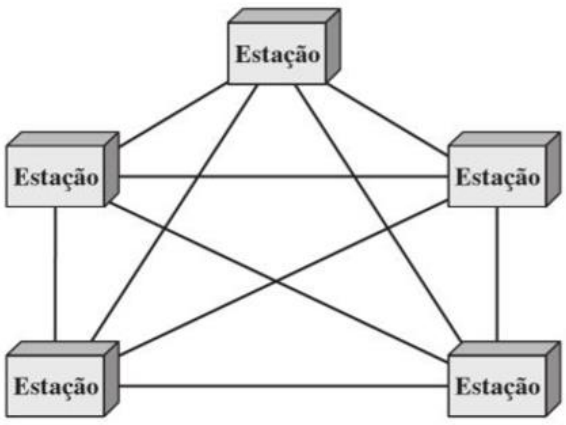
Fonte: Forouzan, 2007

2.2.2 Tipos de topologia

O termo topologia física refere-se a forma que a rede é organizada fisicamente. Dois ou mais dispositivos se conectam a um *link*. Dois ou mais *links* ou mais *links* formam uma topologia, a forma de como são representados geometricamente definem o tipo de topologia, adiante serão apresentados os tipos mais comuns: malha, estrela, barramento e anel.

Malha: na topologia em malha cada dispositivo possui um *link* ponto a ponto dedicado com os demais dispositivos, ou seja, o número de *links* que o dispositivo deve ter para este tipo de topologia é a quantidade de equipamentos a serem conectados menos o dispositivo que será conectado, conforme visto na Figura 5.

Figura 5 - Topologia de malha

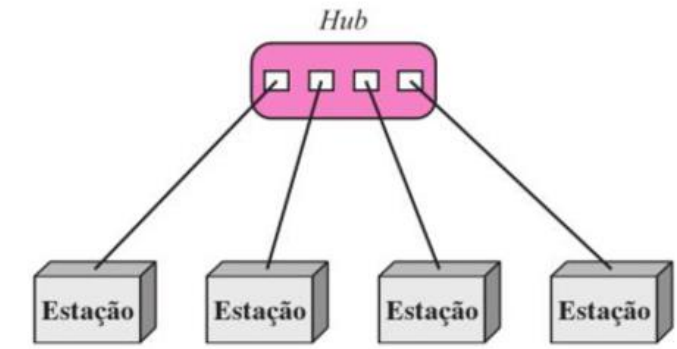


Fonte: Forouzan, 2007

Algumas das vantagens deste tipo de conexão é a exclusividade que os dados terão para trafegar pelo link, a robustez – quando um link deixa de funcionar os demais permanecem inalterados, privacidade, segurança e facilidade na identificação de falhas. Já a desvantagem é a quantidade de cabeamento, bem como também o número de I/Os necessária por dispositivo, o que eleva o custo deste tipo de topologia, sendo empregada em situações extremamente especiais, como a estrutura de um *backbone* (espinha dorsal), conectando os principais computadores de uma rede híbrida, que podem conter diversas topologias diferentes (FOROUZAN, 2007).

Estrela: nesta topologia cada dispositivo dispõe de um *link* conectado a um controlador central, em geral denominado *hub*. O caminho dos dados vai necessariamente até o controlador que irá encaminhar ao destino desejado. Apresenta como vantagem o baixo custo de implantação, a facilidade de ampliação com a troca do componente central do sistema, cada dispositivo precisa somente de um *link*, volume de cabos bem menor, na falha de um *link* não afeta os demais dispositivos e fácil identificação de falhas. Por outro lado, este sistema tem baixa confiabilidade, ou seja, caso o elemento centralizador falhe pode comprometer toda a rede. Esta topologia pode ser observada na Figura 6 (FOROUZAN, 2007).

Figura 6 - Topologia de Estrela



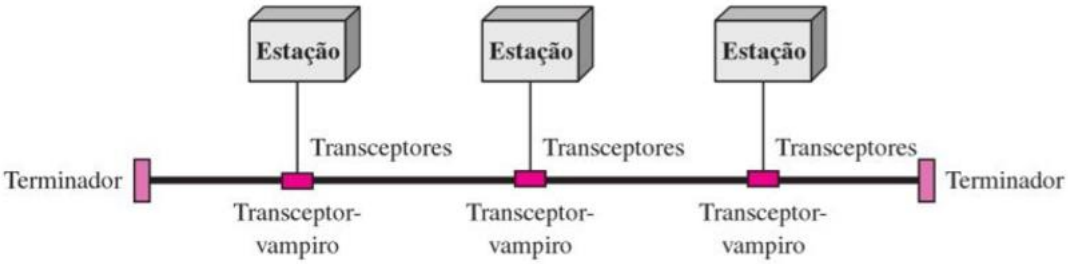
Fonte: Forouzan, 2007

A topologia estrela é usada em rede locais (LAN), que utilizam *hub* central.

Barramento: as topologias anteriores demonstravam um tipo de conexão ponto a ponto, uma topologia barramento é multiponto. Como apresentado na Figura 7, a infraestrutura de rede é interligada por um único meio de transmissão de dados através de um cabo metálico coaxial, assim todos os equipamentos da rede que estão interligados recebem os mesmos dados para o processamento e é definida a prioridade. Como vantagem apresenta baixo custo de implantação e facilidade de interligação do sistema entre as desvantagens estão a dificuldade

de reconfiguração e o isolamento de falhas, caso haja falha no barramento, todo o sistema ficará comprometido. Esse tipo de conexão necessita de um dispositivo eletrônico nas pontas do cabo a fim de eliminar as reflexões de sinal (FOROUZAN, 2007).

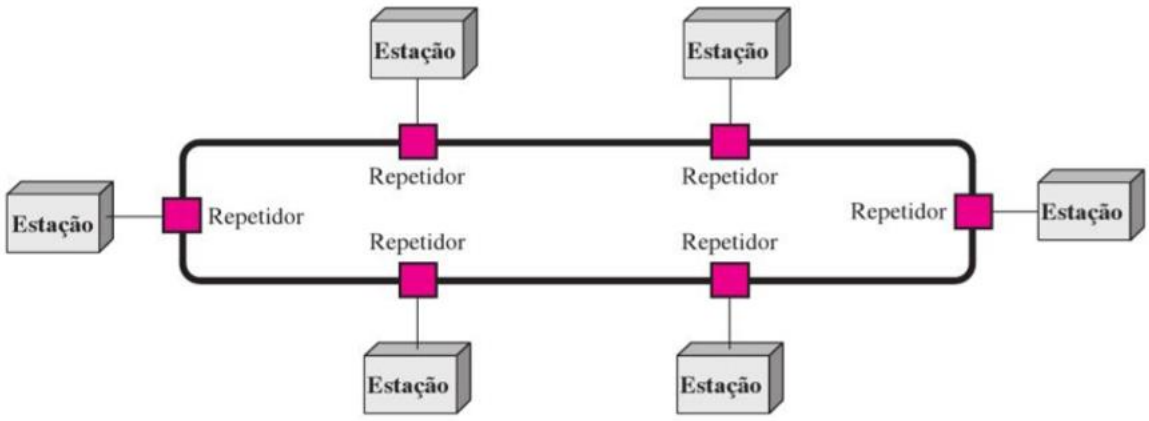
Figura 7 - Topologia de Barramento



Fonte: Forouzan, 2007

Anel: nesta topologia cada dispositivo dispõe de uma conexão ponto a ponto com outros dois dispositivos conectados de cada lado, com forme visto na Figura 8, o sistema dispõe de repetidores ou *switches* onde cada um está ligado a outros dois por um cabo unidirecional a fim de formar uma conexão fechada, com isso cada equipamento comporta-se como um repetidor de sinal conseguindo maiores distâncias na ligação dos componentes de rede. Os fatores limitantes estão relacionados ao meio de transmissão e ao tráfego. O isolamento de falhas é simplificado, em virtude do sinal estar percorrendo a todo o tempo no caso de uma falha um alarme será disparado, no caso de ser um anel unidirecional, uma interrupção pode derrubar toda a rede, para isso empregasse um anel duplo (Foruzan, 2007).

Figura 8 - Topologia em Anel



Fonte: Forouzan, 2007

2.3 INTERCONEXÃO DE REDES: INTERNETWORK

As redes LAN e WAN quando conectadas formam uma *internetwork* ou internet, ou seja, quando duas ou mais redes se conectam entre si juntas formam a internet. A mais notável das internets é a Internet, uma colaboração de centenas de milhares de redes interconectadas, é um sistema organizado e estruturado que permite o acesso a uma ampla gama de informações, que passou a existir em 1969 (FOROUZAN, 2007).

Para enviar dados entre um dispositivo na rede local e outro dispositivo de outra LAN, é necessária uma forma padronizada de comunicação, pois as redes locais podem usar diferentes tipos de tecnologias. Essa necessidade levou ao desenvolvimento do endereçamento IP e dos muitos protocolos IP para comunicação pela Internet, que é um sistema global de redes de computadores interconectadas (FOROUZAN, 2007).

2.4 HARDWARE PARA COMUNICAÇÃO DE DISPOSITIVOS

Quando se tem a necessidade de trocar informação entre diversos dispositivos, faz-se necessário à utilização de *hardware* especial como placas de rede, sendo necessário adquirir uma placa para cada equipamento dependendo de sua necessidade. Nesta seção, será apresentada uma breve descrição sobre placa de rede, *switch* e roteador.

Cartão de interface de rede: o *Network Interface Card* (NIC) ou em português cartão de interface de rede consiste na interface física que se adequa a ligação do processador com os protocolos de comunicação do sistema escolhido podendo ser do tipo fibra óptica, cabo de par trançado, wireless entre outras.

***Switch*:** é um dispositivo simples de pequeno processamento cuja função é de interligar diversos aparelhos, mantendo uma ligação confiável a todos, ele trabalha redirecionando o tráfego de informações enviadas diretamente à porta de destino, mantendo o sistema com o menor congestionamento possível, trabalha com poucas camadas baseado no modelo OSI, sendo elas a camada física e de enlace, podendo chegar a trabalhar com a terceira camada ou camada de rede em versões atuais.

Roteador: equipamento eletrônico com porte para realizar a integração de duas redes distintas e facilitar a comunicação entre as mesmas sempre que necessário, um roteador encaminha informações de uma rede para outra, com base em endereços IP. Ele encaminha apenas os pacotes de dados que precisem ser enviados a outra rede, possui *switch* internamente a fim de aumentar a quantidade conexões físicas e interligar os dispositivos.

2.5 ENDEREÇAMENTO IP

Qualquer dispositivo que deseje se comunicar com outros dispositivos pela Internet deve ter um endereço IP exclusivo e apropriado. Os endereços IP são usados para identificar os dispositivos que enviam e que recebem. Atualmente, há duas versões de IP: IP versão 4 (IPv4) e IP versão 6 (IPv6). A principal diferença entre a duas é que um endereço IPv6 é mais longo (128 bits, contra os 32 bits de um endereço IPv4) - AXIScommunications.

2.6 ENDEREÇOS IPV4

Os endereços IPv4 são agrupados em quatro blocos, cada um separado por um ponto. Cada bloco representa um número entre 0 e 255; por exemplo, 192.168.12.23. Alguns blocos de endereços IPv4 foram reservados exclusivamente para uso privado. Esses endereços IP privados são de 10.0.0.0 a 10.255.255.255, 172.16.0.0 a 172.31.255.255 e 192.168.0.0 a 192.168.255.255. Esses endereços podem ser usados apenas em redes privadas e não podem ser encaminhados através de um roteador para a Internet. Todos os dispositivos que desejarem se comunicar pela Internet devem ter seu próprio endereço IP público. Um endereço IP público é um endereço designado por um provedor de serviços de Internet. Um ISP pode designar um endereço IP dinâmico, que pode mudar durante uma sessão, ou um endereço estático, normalmente cobrado por mês (AXIScommunications).

2.7 ENDEREÇOS IPV6

O IPv6 é a versão 6 do protocolo IP. O IPv6 tem como objetivo substituir o padrão anterior, o IPv4, que só suporta cerca de 4 bilhões (4×10^9) de endereços, enquanto que o IPv6 suporta $3{,}4 \times 10^{38}$ endereços (IPv6.br).

Os endereços IPv6 são normalmente escritos como oito grupos de 4 dígitos hexadecimais. Por exemplo, 2000:0000:85a3:08d3:1319:0000:0000:0000. Esses grupos de zeros podem ser omitidos, o endereço anterior pode ser escrito da seguinte forma 2000:0:85a3:08d3:1319:: (IPv6.br).

Com relação a representação dos endereços IPv6 em URLs (Uniform Resource Locators), estes agora passam a ser representados entre colchetes. Deste modo, não haverá ambiguidades caso seja necessário indicar o número de uma porta juntamente com a URL. Por exemplo a URL: http://[2000:70a:0:4::22]:8070, 8070 representa a porta (IPv6.br).

2.8 PADRÃO IEEE 802.15.4

As principais tecnologias usadas no mercado para controle de casas inteligentes seguem o padrão IEEE 802.15.4 criado pelo IEEE (*Institute of Electrical and Electronics Engineers*) para padronizar a camada física e a subcamada MAC para uso em conexão sem fio e baixa taxa de transmissão de dados com dispositivos de baixo custo e baixo consumo de energia (IEEE 802.15.4, 2015).

Neste padrão são definidas duas categorias de dispositivo: RFD (*Reduced-Function Device*), FFD (*Full Function Device*). O FFD possui duas formas de trabalhar: coordenador PAN ou simplesmente coordenador. A primeira é capaz de criar uma rede e é responsável pela coordenação de toda a rede, portanto deve ser único. A segunda gerencia somente parte da rede, mas pode expandi-la. Já o RFD não é capaz de tais funções, ele foi planejado para aplicações simples, como um interruptor ou um sensor de infravermelho, além disso, comunica somente com um único FFD por vez. Portanto, não precisa de muitos recursos para funcionar (IEEE 802.15.4, 2015).

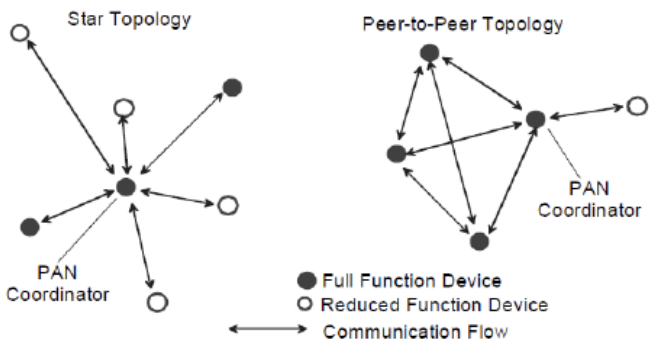
Este protocolo define as frequências 2.4 GHz, 868 MHz e 915 MHz para a subcamada MAC (Media Access Control) da camada de enlace (IEEE 802.15.4, 2015).

A camada física é a primeira camada do modelo OSI. Ela é responsável pela tradução do sinal (que veio do meio físico) em 0s e 1s para ser transmitido para a camada de enlace (TANENBAUM, 2011).

A camada de enlace, segunda camada do modelo OSI, faz o endereçamento, controle de fluxo e de erros que tenham ocorrido no nível físico (FOROUZAN, 2010).

Existem duas topologias possíveis para este protocolo, estrela e *peer-to-peer*. Vide Figura 9.

Figura 9 - Topologias da IEEE 802.15.4



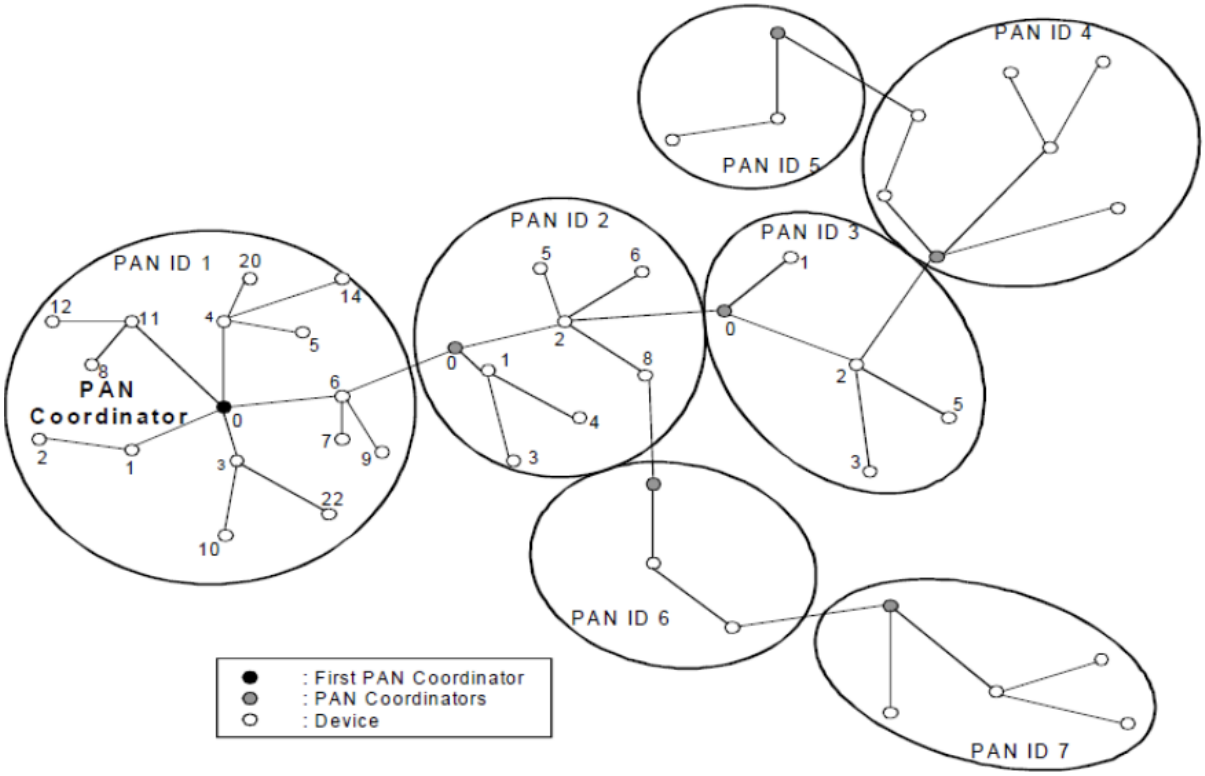
Fonte: IEEE 802.15.4, 2015

Na topologia estrela toda comunicação é entre dispositivos e o coordenador PAN, esta configuração age de forma independente de outras redes tipo estrela. Este tipo de conexão é vantajoso para automação residencial, PCs, jogos e cuidados de saúde pessoal.

Já no P2P (*Peer-to-Peer*) a comunicação pode ser feita por qualquer dispositivo que esteja dentro de seu alcance, isso permite a formação de estruturas mais complexas, como a conexão em malha e árvore.

A maior parte dos dispositivos na rede em árvore, Figura 10, são FFDs e os RFDs são conectados como se fossem uma folha, porque RFDs não permitem que outros dispositivos se associem à ele. Qualquer FFD pode ser um coordenador e prover sincronização para outros dispositivos e coordenadores. Porém, somente um deles poderá ser o coordenador PAN. Este coordenador forma a primeira aglomeração ao escolher um PAN ID que não está sendo usado e transmite *beacons* (sinais) para os dispositivos vizinhos. Um dispositivo que receber esse *beacon* pode pedir ao PAN coordenador para entrar na rede, caso este aceitar, o dispositivo será adicionando como dispositivo filho em sua lista de vizinhos e o dispositivo adiciona o coordenador PAN como pai na sua lista de vizinhos. Em seguida, começa a transmitir *beacons* periodicamente e assim, outros podem se juntar a rede. Caso ele não consiga se juntar a rede, irá procurar outro dispositivo pai. (IEEE 802.15.4, 2015).

Figura 10 - Exemplo de rede cluster tree ou árvore



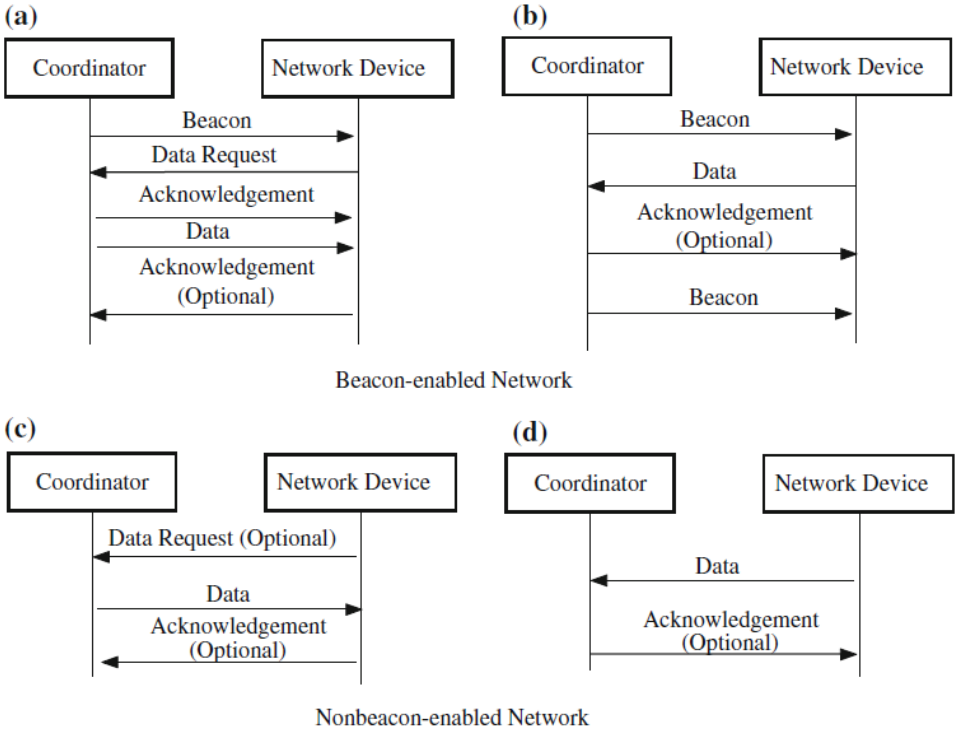
Fonte: IEEE 802.15.4, 2015

Existem dois métodos de transmissão de dados: *beacon* habilitado ou desabilitado, vide Figura 11.

Numa rede com *beacon* habilitado, quando um coordenador quer transmitir dados para o dispositivo, caso a Figura 11, este espera receber um *beacon* (sinal usado para sincronizar a rede, geralmente gerado pelo coordenador PAN da rede) para enviar um pedido requisitando a informação, em seguida o coordenador envia o reconhecimento e o dado. No caso do dispositivo precisar enviar informação para o coordenador, caso b da Figura 11, o primeiro espera receber um *beacon*, para então envia-lo. Este modo de operação permite salvar energia, pois os dispositivos da rede normalmente hibernam e o receptor de sinal, nesse estado, está desligado (YANG, 2014).

Na rede com *beacon* desabilitado, caso o coordenador queria enviar uma informação para o dispositivo, ele pode esperar que destinatário envie um pedido de informação ou simplesmente enviar o dado. Se o dispositivo precisar enviar algo, ele pode transmitir sem precisar esperar (YANG, 2014).

Figura 11 - Sincronização com Beacons e sem Beacons da IEEE 802.15.4.



Fonte: YANG, 2014.

2.9 TECNOLOGIAS APLICADAS EM AUTOMAÇÃO RESIDENCIAL

De acordo com Gomez e Paradells (2010) ZigBee, Z-Wave e tecnologias baseadas em IP são algumas das principais soluções para automação residencial.

Tabela 1 - Sumário das principais características ZigBee e 6LoWPan

		ZigBee	6LoWPAN
Physical Layer	Rf band (MHz)	868/915/2400	
	Range (m)	10-100	
	Bit rate (kb/s)	20/40/250	
	Modula-tion	BPSK/O-QPSK	
	Spreading technique	DSSS	
Physical Layer	Receiver sensitivity (dBm)	-85 or better (2.4 GHz band) -92 or better (868/815 MHz bands)	
Link layer	MAC mechanism	TDMA + CSMA/CA (beacon mode) and CSMA/CA (beaconless mode)	
	Message size (bytes)	127 (maximum)	
	Error control	16 bit CRC, ACKs	

(Continuação)

Comuni-cation modes	Unicast	YES	YES
	Broadcast	YES	YES
	Multicast	Yes(NWK and APL layers). Not supported by MAC	IP multicast. Not supported by MAC
	Other modes	Indirect addressing	IPv6 anycast
Identifiers		16 and 64 bit MAC addresses 16 bit NWK identifiers	16 and 64 bit MAC addresses 28 bit IPv6 address (which can be compressed to 16 bit IDs)
Device Types		Coordinator, router and end device	Edge router, mesh node (mesh under), router and host
Network layer	Multihop solution	Mesh routing, tree routing, and source routing	RPL
Network layer	Hop limit	30/10/5 (mesh routing/tree routing/sou-rce routing)	255
	Multihop solution state	O(N) (mesh routing), O(1) (many-to-one routing)	O(N) (root), O(Ndags) (other devices)
End to end reliability		ACKs and control of duplicate packets	TCP/UDP/other
Application layer	Command space	65,536 (clusters)	-
	Device type space	65,536	-
Security		Integrity, confidential-lity, access control, and key management	Integrity, confidentiality, access control (IEEE 802.15.4). Key management not supported

(Continuação)

Translation gateway needed for Internet connectivity	YES	NO
Implementation size	45-128 kbytes (ROM) 2.7-12 kbytes (RAM)	24 kbytes (ROM) 3.6 kbytes (RAM
Specification publicly available	YES	YES

Fonte: adaptado de Gomez, Paradells, 2010.

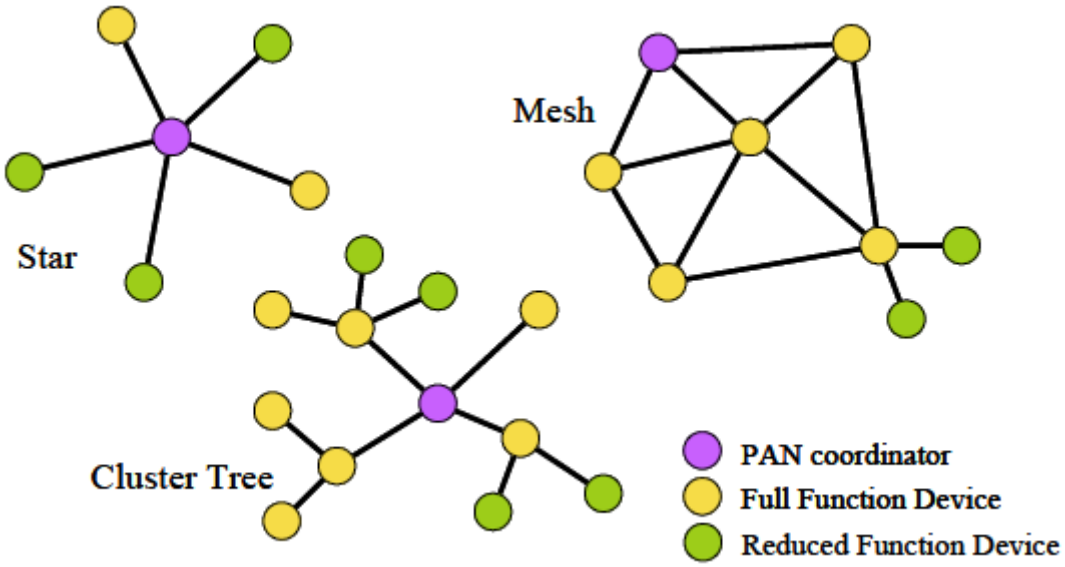
2.9.1 ZigBee

ZigBee é uma tecnologia de rede sem fio desenvolvida pela *ZigBee Alliance* para aplicações de curto alcance (10 a 100 metros), baixo consumo de energia (devido ao pequeno *duty cycle*), porém a velocidade de transmissão de dados é lenta (20 kb/s, 40 kb/s ou 250 kb/s). 868 MHz, 915 MHz e 2.4 GHz são as frequências utilizadas na Europa, Estados Unidos e no mundo, respectivamente. A camada física e a subcamada MAC do ZigBee são definidas pelo padrão IEEE 802.15.4 (GOMEZ; PARADELLS, 2010).

Este protocolo define três tipos de dispositivos: coordenador ZigBee, roteador ZigBee e dispositivo final. Coordenador ZigBee deve ser do tipo FFD e equivale ao coordenador PAN da IEEE 802.15.4, o roteador faz o roteamento dos dispositivos e também deve ser do tipo FFD, já o dispositivo final conversa com um roteador ou coordenador e pode ser RFD ou FFD (AZEVEDO, 2006).

A sincronização segue o modelo IEEE 802.15.4.

Figura 12 - Topologias do ZigBee



Fonte: CRAIG, 2004

2.9.2 6LoWPAN

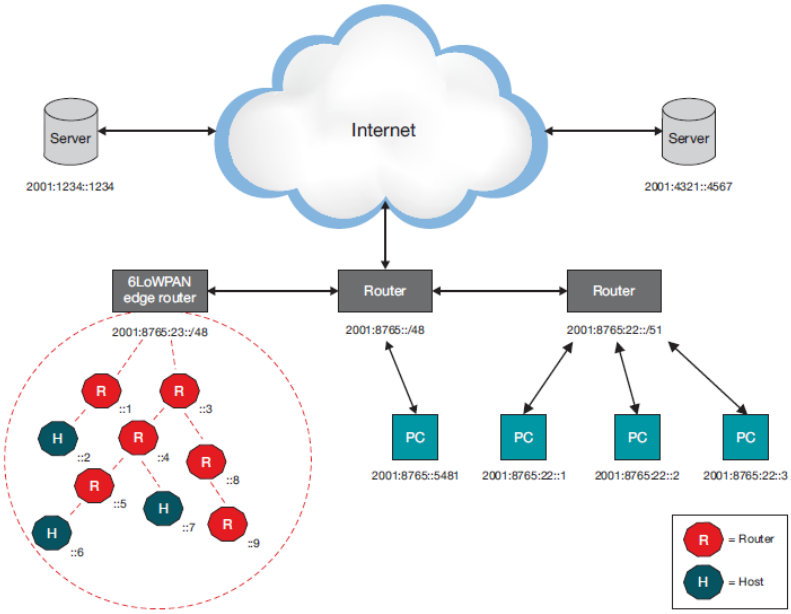
O 6LoWPAN (acrônimo de IPv6 *over Low power Wireless Personal Area Networks*) é uma rede em malha, que necessita de pouca energia, possui um alcance curto (10 a 100 metros) e as frequências disponíveis são 20/40/250 kb/s. O 6LoWPAN é um protocolo baseado no IPv6 (GONNOT; SANNIE, 2014).

O protocolo LoWPAN divide os dispositivos em *host*, *router* e *mesh nodes*. O primeiro é o nó que fica no fim da rede, é a fonte ou o destino dos datagramas do IPv6. *Router* e *mesh node* são FFD e ajudam na transmissão entre o nó de origem e destino. O *router* é responsável pelo o roteamento. Já o *edge router* é um tipo de *router* que faz a ligação entre o LoWPAN e outra rede, por exemplo a Internet. O *mesh node* opera no topo da camada de enlace de dados para o envio de dados através do endereço de enlace. (MISRA; GOSWAMI, 2017).

A sincronização destes componentes ocorre da mesma forma como no *Zigbee*, pode ser feita por *beacons* ou não (GOMEZ; PARADELLS, 2010).

Na Figura 13 temos a conexão de uma rede 6LoWPAN com a internet e roteadores usando IPv6.

Figura 13 - Uma rede IPv6, com acesso à Internet, com uma rede 6LoWPAN



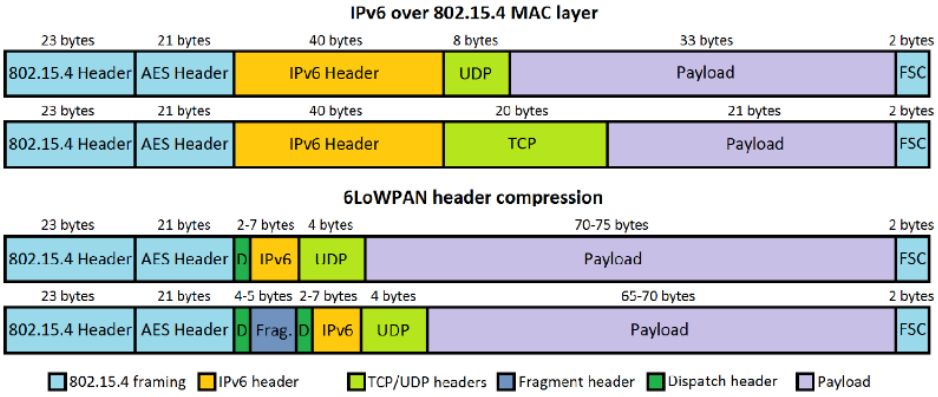
Fonte: Olsson, Texas Instruments, 2014

O 6LoWPAN foi criado para permitir uso do protocolo IPv6 no padrão da IEEE 802.15.4 em conexões sem fio e aumentar o espaço para o *payload* (dados úteis).

O tamanho máximo do frame desse padrão é de 127 bytes, dessa forma, o espaço disponível para o *payload* (dados úteis) é de 70-75 bytes para UDP e 65-70 *bytes* para TCP, praticamente o dobro do espaço disponível no padrão normal do IPv6. Isso se deve à compressão do cabeçalho do IPv6, veja a Figura 14 (GONNOT; SANNIE, 2014).

Como não há suporte para o protocolo TCP e o protocolo UDP não é capaz de verificar se a informação chegou de forma correta, a camada da aplicação ficará responsável pela verificação de possíveis erros (MISRA; GOSWAMI, 2014).

Figura 14 - Ipv6 e 6LoWPAN no pacote padrão do IEEE 802.15.4



Fonte: Gonnot e Saniie (2014)

A segurança deste protocolo fica por conta do cabeçalho AES, definido pela IEEE 802.15.4, que faz a encriptação do frame. Isso permite que uma chave de acesso seja comum para toda a rede (GONNOT; SANNIE, 2014).

Caso um novo dispositivo entre na rede, este precisará ter a chave na memória (GONNOT; SANNIE, 2014).

2.10 O ARDUINO

Segundo McRoberts (2015, p. 22), "O Arduino é o que chamamos de plataforma de computação física ou embarcada, ou seja, um sistema que pode interagir com seu ambiente por meio de *hardware* e *software*."

Massimo Banzi, David Mellis, David Cuartielles, Tom Igoe e Gianluca Martino, da esquerda para a direita, na Figura 15, são os fundadores do Arduino, um projeto de código aberto, onde na figura 16 temos o modelo Uno, utilizado neste projeto. Idealizado na Itália, no ano de 2005, o professor Banzi, do Instituto Ivrea, começou a criação do Arduino com o intuito de ensinar aos seus alunos de design um pouco de eletrônica e programação de dispositivos. Como uma ferramenta de baixo custo não estava disponível no mercado, Banzi teve de criar sua própria placa, e seu aluno David Mellis desenvolveu a linguagem de programação do Arduino. Desta forma surgiu o Arduino, uma das mais populares aplicações de eletrônica. Os demais criadores do Arduino integraram a equipe antes de se tornar um equipamento comercializado, cada qual com sua competência para enriquecer o projeto (EVANS, 2015).

Figura 15 - Desenvolvedores do Arduino

Fonte: Arduino.cc

Existem atualmente inúmeros fabricantes de microcontroladores e plataformas disponíveis para computação física, como por exemplo o Parallax Basic Stamp ou o Handyboard do MIT que oferecem desempenho e funcionalidade de igual nível do fabricante Arduino.

Assim como os demais concorrentes, os microcontroladores Arduino envolvem uma programação bastante simples e funcional ao usuário. O Arduino oferece algumas vantagens para fins didáticos, como para professores, alunos e demais interessados.

2.11 CARACTERISITICAS E VANTAGEM DO ARDUINO

Baixo custo: as placas Arduino tem a característica, muito interessante, do baixo custo, que podem ser encontradas a partir de R$ 50,00, visto que pode ser considerada uma central de automação pronta para uso, que para construção requerem diversos conhecimentos e habilidades.

Ambiente de programação: o *software* de programação para Arduino é o IDE, *Integrated Development Environment*, que permite gravar programas e carrega-los nas placas, atendendo desde usuários iniciantes aos avançados, com um ambiente bastante simples e claro. Pode ser programado *off-line* no computador ou *on-line* direto na nuvem, sem a necessidade de ser instalado.

***Software open source* e extensível:** AVR C é a linguagem base do Arduino, mas pode facilmente ser estendido para a linguagem C++, através de sua biblioteca.

***Hardware open source* e extensível**: os planos das placas Arduino são publicados sob uma licença Creative Commons, desta forma, os usuários podem fazer sua própria versão do módulo, estendendo-o e aprimorando-o de acordo com sua necessidade.

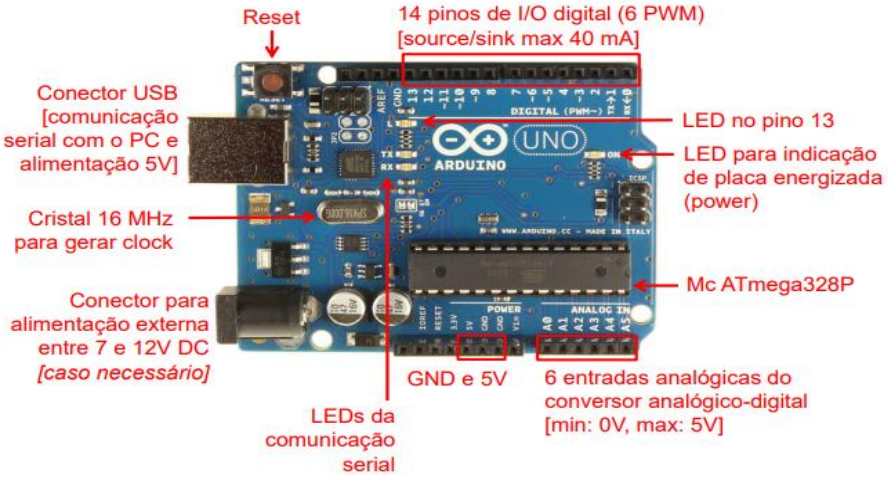
Figura 16 - Arduino Uno

Fonte: Arduino.cc

Para este Trabalho de Conclusão de Curso foi escolhido a placa Arduino Uno, um dos tantos modelos do fabricante, onde além das vantagens já citadas, foi selecionado como modelo ideal para a aplicação proposta pela equipe, visto a grande diversidade de material de estudo e programação disponível na internet, para apoio de todo o projeto.

O Uno detecta o ambiente recebendo sinais de entradas através de sensores, por exemplo, para monitorar o ambiente, e atuar no controle das luzes, motores e outros dispositivos. Na Figura 17 está representado sua anatomia:

Figura 17 - Detalhamento do Arduino Uno



Fonte: Os autores

2.12 ANATOMIA DO ARDUINO UNO

Pinos digitais: são pinos destinados a entradas e saídas digitais, na IDE são acionados pelos comandos de *digitalRead*, *digitalWrite* e *analogWrite* na programação. *AnalogWrite*

funciona apenas nos pinos com o símbolo PWM.

Pino 13 LED: o único atuador embutido em sua placa. Este LED é útil para a depuração.

LED de energia: indica que o Uno está recebendo energia. Útil para depuração.

Microcontrolador ATmega: o coração da placa.

***Analog in*:** Pinos exclusivos para entradas analógicas, na IDE são acionados pelo comando *analogRead* na programação.

GND e pinos de 5V: esses pinos podem ser usados para fornecer energia de + 5V e aterramento aos circuitos.

Conector de energia: o Uno pode ser ligado pela rede, através de uma fonte, quando não está conectado a uma porta USB para se energizar. Pode aceitar tensões entre 7-12V.

Cristal Oscilador 16 MHz: responsável por "informar" ao CI ATmega328 qual a frequência que o mesmo deve trabalhar. No caso do Arduino Uno a frequência é 16 MHz, mas pode ser alterada para valores menores, como 8 ou 4 MHz.

Porta USB: usada para alimentar o Uno, fazer *upload* da programação desenvolvida e para se comunicar (via serial. *Println* etc.).

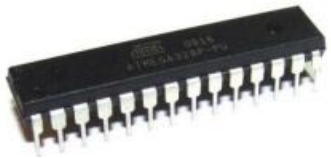
Botão *Reset*: reinicia o microcontrolador ATmega.

2.13 MICROCONTROLADOR ATMEGA 328M

Um microcontrolador pode ser definido como um *single-chip computer* (computador em um único chip), microcomputer, ou ainda como *embedded controller*. O termo MCU (*Micro Controller Unit*) também é utilizado para se designar esse dispositivo. No mesmo chip estão integrados uma CPU, também chamada de core (núcleo), e circuitos auxiliares (periféricos) como memória de programa, memória de dados, circuito de *clock*, interface de comunicação serial, temporizadores/contadores, portas de I/O, etc. Esses diferentes recursos embutidos em um microcontrolador variam em função do modelo e do fabricante. É uma tarefa do desenvolvedor especificar o microcontrolador mais adequado para cada aplicação.

A Figura 17, mostra o local onde o Atmega328 fica localizado na placa. Na Figura 18, podemos ver que o microcontrolador é um chip preto com letras em que se lê "ATMEGA 328P-UP".

Figura 18 - Microcontrolador Atmega328



Fonte: Arduino.cc

O microcontrolador Atmega328 possui 32 KB de memória *flash* (onde é armazenado o s*oftware*), além de 2 KB de SRAM (onde ficam as variáveis) e 1 KB de EEPROM (esta última pode ser lida e escrita através da biblioteca EEPROM e guarda os dados permanentemente, mesmo que desliguemos a placa). A memória SRAM é apagada toda vez que desligamos o circuito.

O microcontrolador pode possuir de 28 ou 100 pinos de conexão, trabalha numa frequência de *clock* 16 MHz. Permite um conjunto de instruções estendidas (múltiplas instruções e instruções para gerenciamento de programas com grandes memórias) e conjunto extensivo de periféricos. A capacidade do microcontrolador ATmega328 e suas características fazem com que a placa Arduino Uno e o modelo Arduino Duemilinove sejam os modelos de placas Arduino mais versáteis existentes atualmente.

2.14 ARDUINO SHIELD

De modo geral, os *shields* são placas que podem ser conectadas na parte superior do Arduino, estendendo seus recursos e existem para diversas aplicações, os mais comuns são os *Ethernet Shield*, *Motor Shield*, *Relay Shield*, LCD *Shield*, *xbee Shield*, etc. Este último, por exemplo o *shield Xbee*, permite a comunicação de várias placas Arduino, sem fio, em distâncias de até 100 pés ou 300 pés, para ambientes internos ou externos, respectivamente, usando o módulo Maxstream Xbee Zigbee, segundo o site oficial.

2.15 ARDUINO UNO E ETHERNETSHIELD

Para este TCC vamos limitar a explicação e uso do Arduino *Ethernet Shield* W5100 (Figura 19), que ao ser acoplado ao Arduino, juntamente com um cabo de rede, faz com que em poucos segundos seja possível monitorar o estado de sensores, chaves e outros dispositivos a partir do *browser* do computador ou celular. Este *shield* é baseado no ethernet chip Wiznet W5100 e fornece um endereço IP compatível com os protocolos, TCP *Transmission Control*

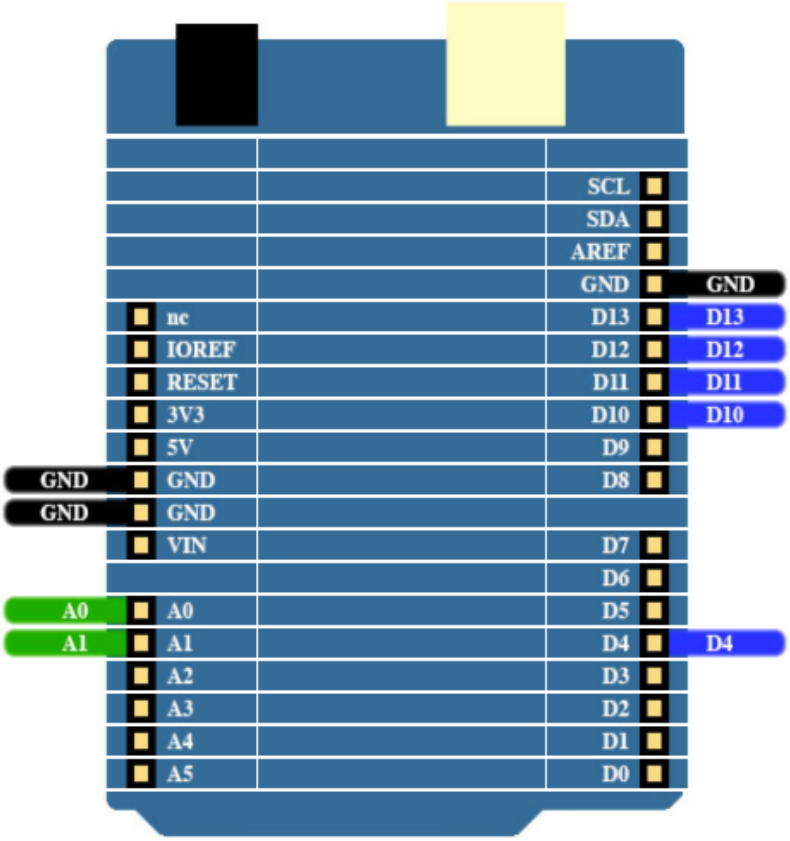
Protocol (Protocolo de Controle de Transmissões) e UDP, *User Datagram Protocol* (Protocolo de Datagramas de Usuário), que são protocolos com base em IPs.

Figura 19 - Ethernet Shield

Fonte: Arduino.cc

Na Figura 20 está representada sua anatomia:

Figura 20 - Anatomia Ethernet Shield



Fonte: Arduino.cc

O Arduino se comunica tanto com o W5100 (via cabo ethernet) como com o cartão SD usando o barramento SPI (através do cabeçalho ICSP). Isso é nos pinos D11, D12 e D13 em modelos de formato "clássico" do Arduino.

O pino D10 é usado para selecionar o W5100 e não pode ser usado para *I/O* gerais.

Já o pino D4 é usado para o cartão SD e só pode ser usado para I/O gerais se o *slot* SD não estiver ocupado.

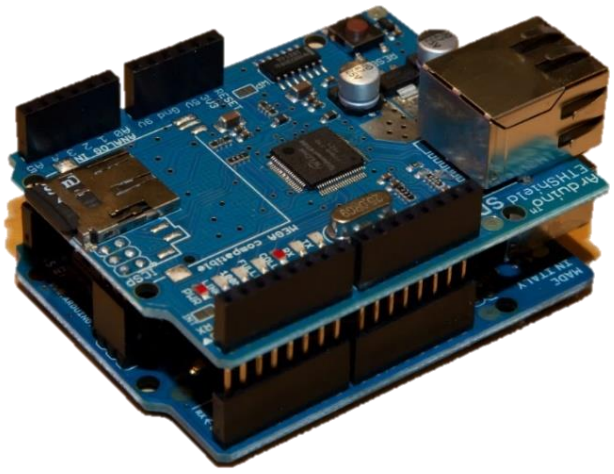
Pode ser observado na Figura 21, o *Ethernet Shield* acoplado sobre a placa do Uno mantendo acessível todos os pinos da placa principal, com exceção dos 4 pinos descritos na anatomia do *Ethernet Shield* para seu funcionamento. Desta forma o Arduino passa a ter mais essa funcionalidade.

O restante do processo é feito via *software* e é necessário a inclusão da biblioteca Ethernet no código e a configuração do IP que o Arduino terá na rede.

O *shield* contém vários LEDs informativos:

- PWR: indica que a placa e o escudo estão ligados
- LINK: indica a presença de um *link* de rede e pisca quando o *shield* transmite ou recebe dados
- FULLD: indica que a conexão de rede é *full duplex*
- 100M: indica a presença de uma conexão de rede de 100 Mb / s (em oposição a 10 Mb / s)
- RX: pisca quando o escudo recebe dados
- TX: pisca quando o escudo envia dados
- COLL: pisca quando as colisões de rede são detectadas

Figura 21 - Arduino Uno com Ethernet Shield



Fonte: Arduino.cc

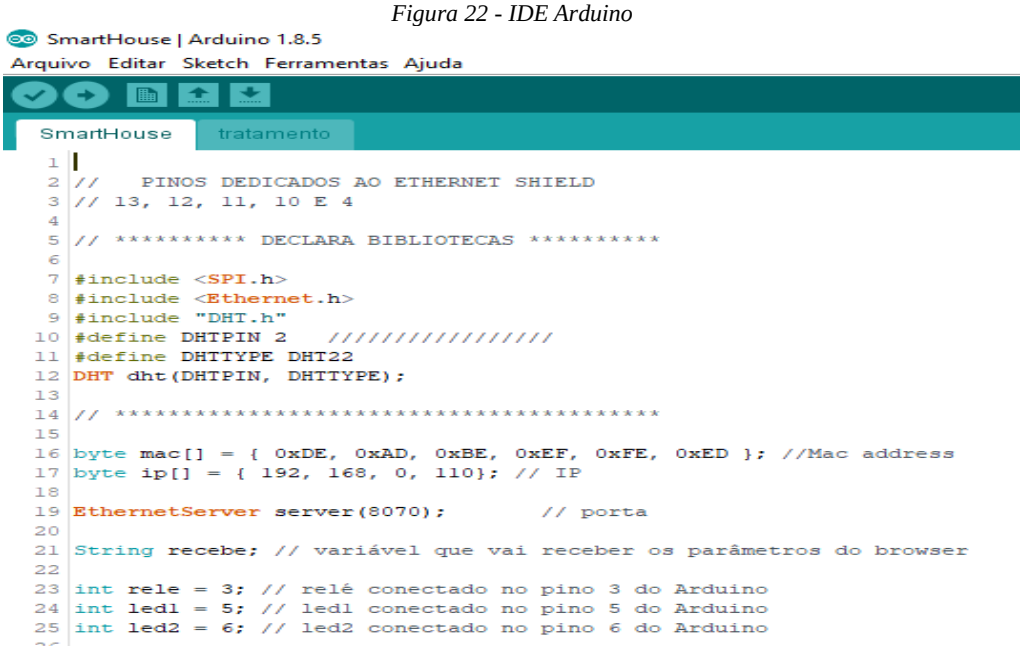
2.16 IDE

O Ambiente de Desenvolvimento Integrado do Arduino (*IDE – Integrated Developmente Environment*) é um *software* que pode ser instalado no computador ou utilizado de forma *on-line*, onde a programação desenvolvida será salva em nuvem mantendo o usuário com a versão mais atualizada do IDE. Neste projeto utilizaremos a versão *desktop* mais atualizada 1.8.5.

O *software* tem a finalidade de facilitar a gravação de código e *upload* para a placa Arduino, com um ambiente de desenvolvimento em Java baseado em *Processing* e outros *softwares* de código aberto.

Na Figura 22, um exemplo de um código desenvolvido na IDE.

Figura 22 - IDE Arduino



Fonte: Arduino.cc

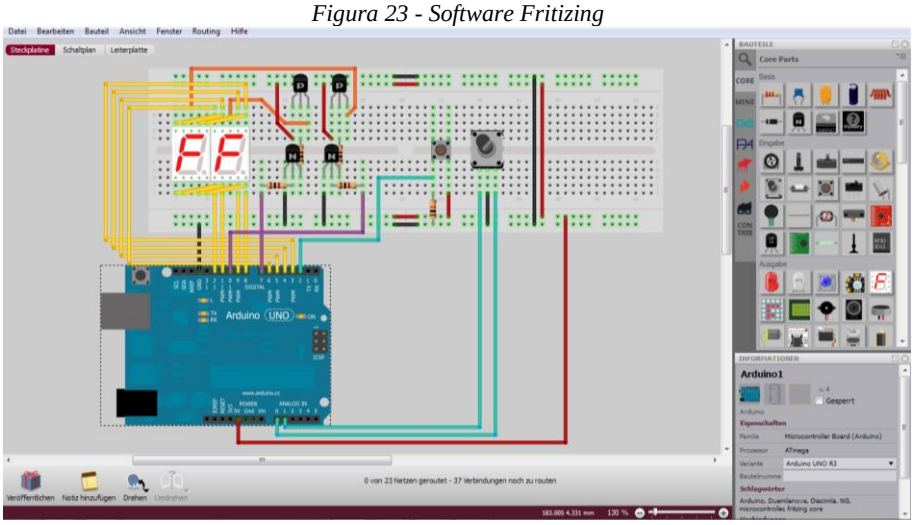
2.17 FRITZING

O Fritzing é um *software* de plataforma aberta destinado a simulação de montagem de circuitos em *protoboards* em um ambiente com vasta biblioteca de componentes. Limitaremos o uso do programa para emulação da placa Arduino com os acionamentos propostos neste projeto.

O *download* gratuito do programa pode ser encontrado em seu site origem: http://fritzing.org/home/.

A Figura 23 mostra um exemplo do Fritzing com diversos componentes integrados ao Arduino e parte da sua biblioteca a direita.

Figura 23 - Software Fritizing



Fonte: fritzing.org

2.18 APLICATIVO APP INVENTOR

O App Inventor for Android é um *software* de programação em blocos, desenvolvido pelo Google e o *Massachusetts Institute of Technology* (MIT) criado para o desenvolvimento simples e intuitivo de aplicativos para dispositivos móveis Android, se comparados a linguagens de programação tradicionais, é basicamente como montar legos, a programação é feita de forma visual, onde o desenvolvedor vai literalmente “juntando” as partes como se fosse um quebra cabeça. Além deste atrativo, o *software* também se destaca pela possibilidade de permitir que o *app* desenvolvido incorpore serviços baseados na *web*, interação com redes sociais, leitura de códigos de barra, interação com sensores de orientação e geolocalização, e de funcionalidades como *text-to-speech* e reconhecimento de fala (MIT, 2012).

A programação de uma aplicação no App Inventor é orientada a eventos, ou seja, o comportamento dos componentes depende, em sua maioria, de eventos provocados pela interação do usuário com o aplicativo.

O desenvolvimento de uma aplicação nesta ferramenta é realizado através de duas janelas: *App Inventor Designer* e *Blocks Editor*. A janela *App Inventor Designer* é executada a partir do navegador e tem a finalidade de editar visualmente a interface do usuário, ao clicar e arrastar os componentes da *Palette*, tais como botões, caixas de texto, figuras, animações, sons, entre outros, para o *Viewer* (Figura 24).

Figura 24 - Janela do App Inventor

Fonte: os autores

A janela *Blocks Editor* por sua vez, permite controlar o comportamento dos componentes definidos na *App Inventor Designer*. Neste ambiente, o usuário encontra blocos conectáveis, que podem ser eventos ou métodos, em uma interface do tipo arrastar e soltar. Estes blocos operam *strings* e listas, realizam ações de controle (e.g. *if*, *else*, *foreach*, etc.) e operações matemáticas, entre outras funcionalidades. É possível executar o teste do aplicativo diretamente em um dispositivo Android (*smartphone* ou *tablet*) que esteja conectado ao computador ou através do emulador que acompanha o *Blocks Editor*.

Figura 25 - Janela de Blocks do App Inventor

Fonte: os autores

3 RESULTADOS

Neste capítulo serão explicados a construção da maquete, a programação do Arduino e o aplicativo de smartphone para sistema operacional Android.

3.1 ASPECTO CONSTRUTIVO DA MAQUETE

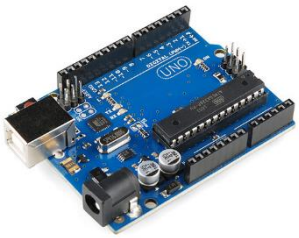
O projeto foi montado em uma maquete de casa térrea feita em MDF 3mm, em escala 1:25. Suas dimensões são 17 cm de altura, 48 cm de comprimento e 40 cm de largura. A maquete foi comprada em um site de vendas, fabricada pela 3DMogi - Figura 26, o preço de venda atual é R$ 79,00.

Figura 26 - Maquete.

Fonte: 3dmogi

Foi adquirido uma placa Uno R3 - Figura 27, com cabo USB para Arduino, o modelo conta com um chip microcontrolador ATmega328, com tensão de operação 5V, possuindo 14 portas digitais podendo 6 serem usadas como PWM. Preço da placa em lojas virtuais brasileiras em torno de R$ 50,00.

Figura 27 - Arduino UNO R3



Fonte: Arduino

Para acesso à internet é necessário o uso do Arduino Ethernet Shield que se baseia no

chip WIZnet ethernet W5100 - Figura 28 que fornece acesso à rede nos protocolos TCP ou UDP e é facilmente utilizado usando a biblioteca Ethernet Library e SD Library. Sendo encontrada no mercado brasileiro por um preço aproximado de R$ 53,90.

Figura 28 - Ethernet Shild

Fonte: Arduino

Para a realização das conexões se usou jumpers macho-macho. Os jumpers possuem a responsabilidade de ligar o fluxo elétrico, cumprindo as configurações específicas do projeto. Os jumpers são ideais para montagem de projetos em protoboard com rapidez, agilidade e limpeza. O valor de um kit com aproximadamente 65 unidades custa em torno de R$ 12,90, em lojas virtuais eletrônicas.

A abertura e fechamento do portão da garagem é feita através de um micro servo motor, foi utilizado o modelo 9g SG90 TowerPro - Figura 29, conforme descrição do fabricante é um servo de alta qualidade e excelente para projetos de robótica com Arduino. Possui ângulo de rotação de 180 graus e acompanha um cabo de 3 pinos referente a alimentação/controle e alguns acessórios.

Figura 29 - Mini servo motor

Fonte: Filipeflop

A tensão de operação é de 3,0 à 7,2V, com velocidade: 0,12 seg/60Graus (4,8V) sem carga. Com torque: 1,2 kg.cm (4,8V) e 1,6 kg.cm (6,0V), com temperatura de operação entre -30C à +60C. Com engrenagem de nylon, com cabo 245mm de comprimento, tamanho de 32

x 30 x 12mm e massa 9g. Pode ser encontrada a um preço R$18,90 no mercado nacional.

Para acionamento de cargas com potencias acima do que a fonte do Arduino suporta é utilizado o módulo relé de 5V com 1 canal, que podem controlar lâmpadas, motores, eletrodomésticos e outros equipamentos utilizando apenas um pino de controle, já que o circuito a ser alimentado fica completamente isolado do circuito do microcontrolador. O módulo relé 1 canal - Figura 30, funciona com tensão de 5V, e pode acionar cargas de até 250 VAC ou 30 VDC, suportando uma corrente máxima de 10A. Possui led indicador de energia, 2 pinos de energia e 1 de controle, além do borne de saída com parafusos, facilitando a conexão dos equipamentos, dimensões 34 x 27 x 17mm. Preço pode chegar a R$ 10,00.

Figura 30 - Relé 1 canal

Fonte: Filipeflop

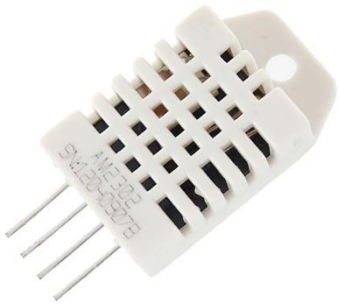
O projeto conta com um sistema de alarme, que é acionada por um sensor de porta - reed switch NA, Figura 31. O *reed switch* é uma chave que funciona por campo magnético, fechando os contatos internos quando aproximamos um ímã do sensor. Ao tirar o ímã, os contatos abrem novamente. Fácil de instalar, pode ser parafusado, ou colado com adesivo dupla face que acompanha. Distância de atuação é 16m ±5mm com tensão continua de 100V, corrente máxima: 300mA, expectativa de vida: 1000000 de acionamentos. Fabricada em ABS, possui fio condutor de 22 AWG, 300mm de comprimento, na cor branco com dimensões de 27x14x8 (mm) e saída NA (normalmente aberto, abre o contato quando distante do atuador). Preço estimado de R$ 3,00.

Figura 31 - Sensor de porta

Fonte: Arduino

O projeto conta com um sensor de umidade e temperatura modelo DHT22 - Figura 32, com tensão de operação de 3,0 a 5,0 V contínuo, corrente: 200uA a 500mA, em *stand by* de 100uA a 150 uA. Faixa de medição de umidade de 0 a 100% e faixa de medição de temperatura de -40º a 80ºC, com tempo de resposta menor de 2s. Dimensões de 25mm x 15mm x 7mm (incluindo terminais).

Figura 32 - DHT22



Fonte: Filipeflop

Este sensor inclui um componente medidor de umidade e um componente NTC para temperatura, ambos conectados a um controlador de 8-bits. As leituras do sensor são enviadas usando apena um único fio de barramento. Preço aproximado R$ 14,00.

Para o sistema de trava da porta foi usado uma mini trava elétrica solenóide 12V (Figura 33) para controle de acesso, feita em aço inoxidável, trava elétrica funciona com tensão de 12V e tem consumo de corrente aproximado de 600mA e com dimensões: 29 x 27 x 18mm. Custo da mini trava R$ 40,00.

Figura 33 - Mini trava elétrica

Fonte: Filipeflop

Para energização fonte DC chaveada 12V 2A plug P4 de uso geral para utilização no Arduino e outra na mini trava. Com plug P4 e fornece na saída uma tensão de 12VDC. É bivolt, ou seja, ajusta automaticamente a energia de entrada, podendo ser utilizada em 127V

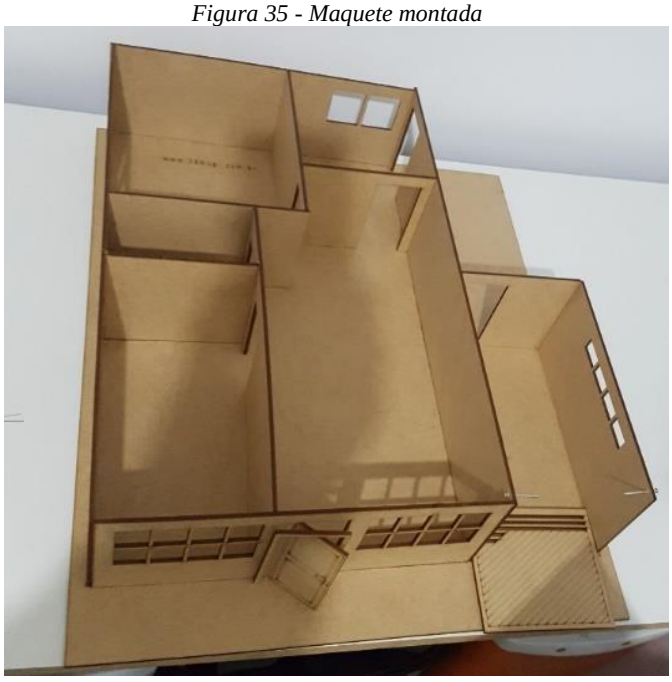
ou 220V. Preço médio R$ 25,00. Modelo da fonte poder visto na Figura 34.

Figura 34 - Fonte chaveada

Fonte: Filipeflop

A maquete foi montada conforme manual de montagem, na Figura 35 pode-se ver a maquete com as repartições encaixadas.

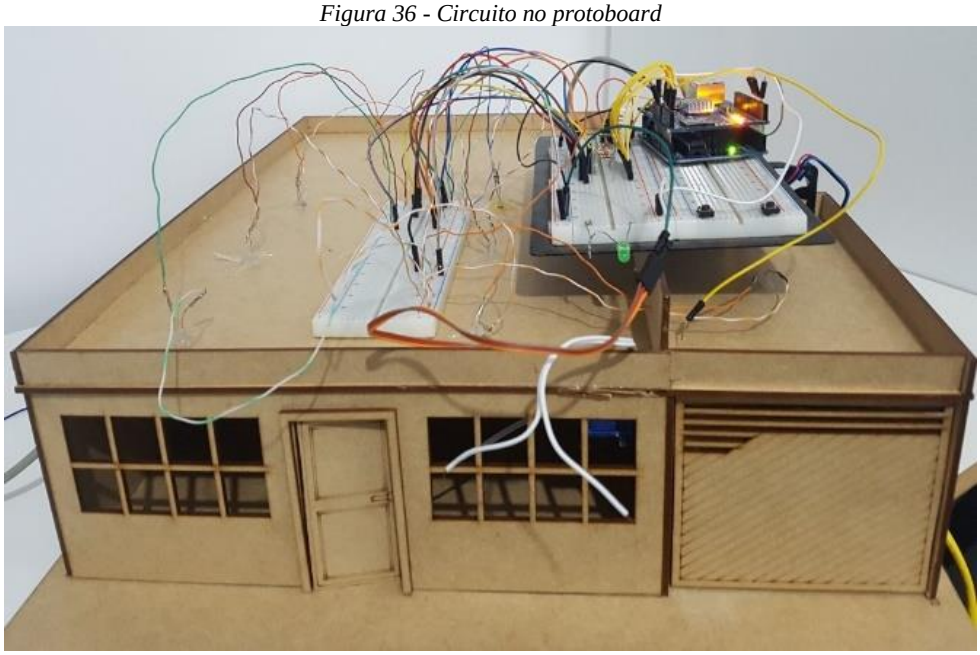
Figura 35 - Maquete montada



Fonte: os autores

Após ser montada foi feito a montagem do circuito sobre o protoboard, Figura 36, nesta montagem presou-se o funcionamento adequado dos componentes e a fixação da lógica que iria ser implantada na automação da maquete.

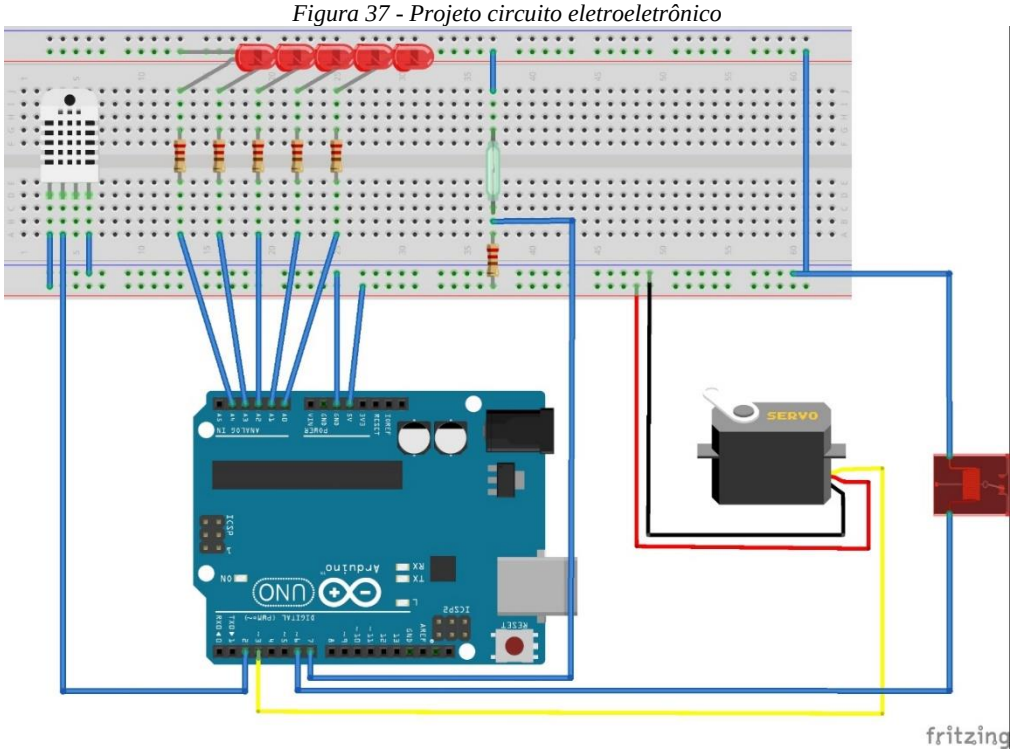
Figura 36 - Circuito no protoboard



Fonte: os autores

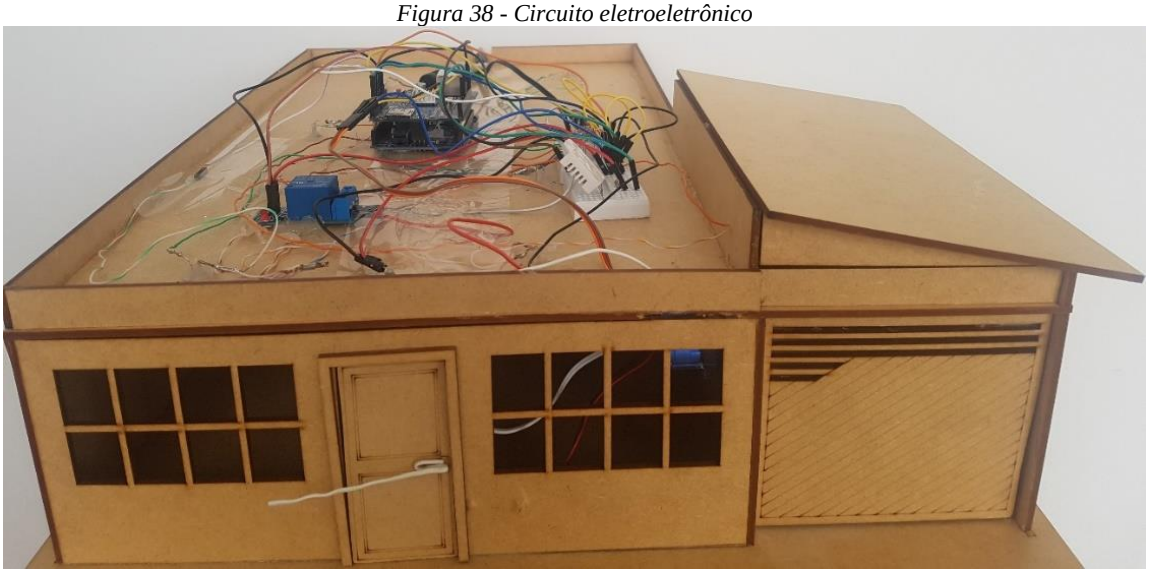
Na sequência foi desenhado o circuito elétrico que foi montado sobre a cobertura, Figura 37, foram usados mini protoboards, todas as conexões das pinagens foram feitas através de jumpers, Figura 38.

Figura 37 - Projeto circuito eletroeletrônico



Fonte: os autores

Figura 38 - Circuito eletroeletrônico



Fonte: os autores

Enfim, com a casa pronta e o circuito montado, Figura 39, foram feitos os testes com o aplicativo por meio de um smartphone.

Figura 39 - Projeto final

Fonte: os autores

3.2 PROGRAMAÇÃO DO ARDUINO

De acordo com a IANA a quantidade de IPv4 válidos está acabando. Isso é um problema, pois o Arduino Ethernet Shield suporta nativamente somente IPv4, sendo assim foi escolhida a biblioteca Ethersia (criada por Nicholas Humfrey) que adiciona suporte para IPv6 e possibilita a criação de um simples servidor HTTP. Para o controle do servo motor tem-se a biblioteca Servo e para o sensor DHT foi escolhida a biblioteca DHT versão 1.2.3 da Adafruit.

É importante notar que todos os comandos (com exceção da temperatura e umidade que devem ter seus valores enviados ao celular) carregam uma página em branco, pois o Arduino executa os comandos conforme a página carregada.

Na Figura 40 inclui-se as bibliotecas mencionadas anteriormente, o pino digital 2 é escolhido para o sensor DHT, define-se o tipo de DHT, DHT22 e o sensor DHT é iniciado. O servidor HTTP (utilizado para receber comandos do aplicativo do celular) e a interface do Ethersia são definidos. Por fim cria-se um objeto tipo Servo.

Figura 40 - Declaração das Bibliotecas

```
// ********** DECLARACAO DAS BIBLIOTECAS **********
#include <Servo.h>                  //Servo.
#include <EtherSia.h>               //Biblioteca do servidor
#include <DHT.h>                    //Sensoriamento de temperatura e
umidade .
#define DHTPIN 2                    //Sensor DHT22 conectado ao pino
digital 2 do Arduino Uno.
#define DHTTYPE DHT22               //Define o tipo de sensor.
DHT dht(DHTPIN, DHTTYPE);           //Inicializa o sensor.

EtherSia_W5100 ether;               //Seleciona a interface, o W5100
corresponde ao Arduino Ethernet Shield.

HTTPServer http(ether);             //Define servidor HTTP.

Servo;                        //Cria objeto tipo Servo
```

Fonte: os autores

Na Figura 41, as varáveis dos LEDs (led0 até led4), porta (mini trava), portão (servo motor), ângulo do servo, temperatura, sensor (para porta), umidade, alarme, intruso, valor anterior, intervalo e ledestado (estas cinco últimas compõem o sistema de alarme) são declaradas.

Figura 41 - Definição das portas e variáveis

```
int led0 = A0;                        //Leds conectados nos pinos A0 a A4.
int led1 = A1;                        //A0 até A5 são pinos analógicos.
int led2 = A2;
int led3 = A3;
int led4 = A4;
int porta = A5;                        //Fechadura da porta no pino
analogico A5.
int portao =  3;                       //Motor do portão no pino 3.
int servoAngle = 90;                   //Posição do servo que começa em 90
graus.

int sensor = 5;                        //Sensor magnético da porta no pino
digital 7.
int est_sensor = 0;                    //Estado do sensor magnético da
porta.
float temp = 0;                        //Temperatura do sensor DHT22.
int umid = 0;                          //Umidade do sensor DHT22.
int alarme;                            //Alarme.
int intruso;                           //Intruso.
unsigned long valoranterior = 0;       //Ultima vez que o led alterou
estado.
const long intervalo = 1000;           //Intervalo do pisca do alarme.
int ledestado = LOW;
```

Fonte: os autores

No void setup, Figura 42, os leds, porta e portão são definidos como saída (OUTPUT) e o sensor para porta como entrada (INPUT). O pino digital 4 pertence ao cartão SD e não deve ser utilizado. São definidos os endereços MAC, IPv6 e o servidor é inicializado com esses dados.

Figura 42 - Void Setup

```
void setup(){
  pinMode     (4,OUTPUT);           //Pino do CARTÃO SD.
  digitalWrite(4,HIGH);             //ANULA LEITURA E ESCRITA NO CARTÃO
SD.
  servo.attach(portao);             //ASSOCIAÇÃO DO PINO DIGITAL AO
OBJETO DO TIPO SERVO.
  pinMode(led0,OUTPUT);             //declara os pinos como SAÍDA ou
ENTRADA.
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(led4,OUTPUT);
  pinMode(portao,OUTPUT);
  pinMode(porta,OUTPUT);
  pinMode(sensor,INPUT);
  MACAddress macAddress("46:29:b1:87:8b:08"); //Seta o endereço MAC.
  ether.setGlobalAddress("2041:0:140F::875B:131B");//Seta o IPv6 Global.
  ether.begin(macAddress); //Inicializa o servidor com o endereço MAC.
}//void setup
```

Fonte: os autores

Na Figura 43 tem-se o loop principal, a umidade, temperatura e estado do sensor da porta que são lidos e armazenados em suas respectivas variáveis. As funções para controle das cargas são adicionadas e o servidor fica pronto para processar pacotes recebidos.

Figura 43 - Void Loop

```
void loop(){
  umid = dht.readHumidity();            //Lê umidade e armazena em umid.
  temp = dht.readTemperature();         //Lê temperatura e armazena em
temp.
  est_sensor = digitalRead(sensor);     //Le o valor do sensor e
armazena no est_sensor
  sensores();                           //Chama a função sensores, faz o
controle do alarme.
  servomotor();                         //Chama função servomotor, faz
controle do motor do portão.
  tratamento();                         //Chama a função tratamento, faz
o controle da porta e iluminação.
  ether.receivePacket();                //Servidor processa o pacote
}//void loop
```

Fonte: os autores

Na Figura 44 é mostrada parte da função tratamento, ela controla os LEDs e a porta. Quando o servidor HTTP receber uma requisição da página /GERALON ele irá enviar o cabeçalho (necessário para a comunicação entre o servidor e o celular) e a porta de todos LEDs mudará para nível alto, no caso de /GERALOFF as portas mudam para nível baixo. O funcionamento é o mesmo para as páginas que controlam individualmente os leds e a trava.

Figura 44 - Void Tratamento

```
void tratamento()
{

if(http.isGet(F("/GERALON")))        // Se a página pedida for "/geralON"
{
  http.printHeaders(http.typeHtml); //Imprime o cabeçalho
  http.sendReply();
  digitalWrite(led0, HIGH);          // liga o pino do led0
  digitalWrite(led1, HIGH);          // liga o pino do led1
  digitalWrite(led2, HIGH);          // liga o pino do led2
  digitalWrite(led3, HIGH);          // liga o pino do led3
  digitalWrite(led4, HIGH);          // liga o pino do led4
}
.
.
.
}//void tratamento()
```

Fonte: os autores

Na Figura 45, quando o servidor receber /PORTAOABRE o ângulo do servo vai diminuindo a cada 30ms até chegar em zero. Quando for /PORTAOFECHA o ângulo aumenta até noventa graus.

Figura 45 - Void Servomotor

```
void servomotor()
{
//servo motor no relé 8

if(http.isGet(F("/PORTAOABRE")))  // Se a página for "/PORTAOABRE"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  for(servoAngle = servoAngle; servoAngle > 0; servoAngle--){  //PARA
ANGULO IGUAL A 90, ENQUANTO O ANGULO FOR MAIOR QUE 0, DECREMENTA O
ANGULO
  servo.write(servoAngle); //GIRA O SERVO NO VALOR ATUAL DA VARIÁVEL
servoAngle
  delay(30); //INTERVALO DE 35 MILISSEGUNDOS
  }//for servoAngle--

}//if PORTAOABRE

if(http.isGet(F("/PORTAOFECHA"))) // Se a página for "/PORTAOABRE"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  for(servoAngle = servoAngle; servoAngle < 90; servoAngle++){  //PARA
ANGULO IGUAL A 180, ENQUANTO O ANGULO FOR MAIOR QUE 0, DECREMENTA O
ANGULO
  servo.write(servoAngle); //GIRA O SERVO NO VALOR ATUAL DA VARIÁVEL
servoAngle
  delay(30); //INTERVALO DE 35 MILISSEGUNDOS
  }//for servoAngle++
}//if /PORTAOFECHA
}//servomotor
```

Fonte: os autores

A Figura 46 mostra o código que envia a temperatura, umidade e faz o controle do alarme. Quando a página for /SENSOR, o servidor irá enviar o valor da temperatura e umidade numa página WEB, o aplicativo irá, então, ler os valores. Caso a página seja /ALARMEON o alarme será ativado, assim, se o sensor detectar porta aberta o Arduino irá piscar todos os LEDs. Se o usuário desligar o alarme (/ALARMEOFF) os LEDs ficarão ligados.

Figura 46 - Void Sensores

```
void sensores(){
  if(http.isGet(F("/SENSOR"))){
    http.printHeaders(http.typeHtml);
    http.println("<meta http-equiv=Content-Type content=text/html;
charset=utf-8>"); //Habilita caracteres especiais
    http.println(temp);
    http.println("°C");
    http.println(umid);
    http.println("%");
    http.sendReply();
  }// if SENSOR

  //*********************************************************************
unsigned long valoratual = millis();
  if(http.isGet(F("/ALARMEON")))
  {
  http.printHeaders(http.typeHtml);
  http.sendReply();
  alarme=1;
  }// if ALARMEON

  if(http.isGet(F("/ALARMEOFF")))
  {
    http.printHeaders(http.typeHtml);
    http.sendReply();
    alarme=0;
    intruso=0;
    digitalWrite(led0, HIGH);         // liga o pino do relé
    digitalWrite(led1, HIGH);         // liga o pino do relé
    digitalWrite(led2, HIGH);         // liga o pino do relé
    digitalWrite(led3, HIGH);         // liga o pino do relé
    digitalWrite(led4, HIGH);         // liga o pino do relé
  }// if ALARMEOFF

  if(est_sensor == HIGH && alarme)    //Se o alarme e o sensor forem
igual a 1
  {
  intruso=1;
  }// /est_sensor == HIGH && alarme

  if(intruso == 1){
    if (valoratual - valoranterior >= intervalo){
      valoranterior = valoratual;
      if (ledestado == LOW){
        ledestado = HIGH;
        }// ledestado == LOW
      else{
        ledestado = LOW;
        }// else
        digitalWrite(led0, ledestado);
        digitalWrite(led1, ledestado);
        digitalWrite(led2, ledestado);
        digitalWrite(led3, ledestado);
        digitalWrite(led4, ledestado);
    }// valoratual - valoranterior >= intervalo
  }// intruso==1

}// void sensores
```

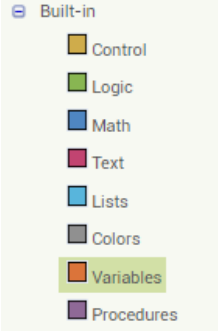
Fonte: os autores

3.3 APLICATIVO SMARTHOUSE PARA SMARTPHONE

O aplicativo desenvolvido para este TCC utiliza a plataforma *App Inventor*, do MIT, e possui três telas (Login, Inicial e Iluminação). A fim de expor com clareza o funcionamento de cada uma delas, em seu *design* e programação, cada tela será apresentada separadamente e detalhada conforme necessidade.

Inicialmente, vale compreender que na aba de desenvolvimento de blocos, cada cor no bloco corresponde a um conjunto de rotinas particulares, podendo ser: controle, lógica, cálculos matemáticos, textos, listas, alteração de cor, variáveis ou procedimentos, conforme exposto na barra a esquerda da tela de programação de blocos, conforme Figura 47.

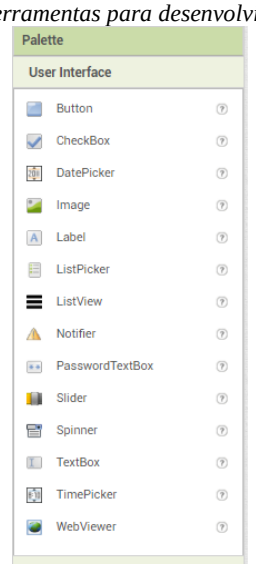
Figura 47 - Cores para construção de blocos conforme rotina



Fonte: AppInventor

Para o desenvolvimento do *design*, o *App Inventor* possui uma barra de ferramentas com diversas aplicações, as mais utilizadas aqui serão para inserir botões, imagens, caixa de texto, password e notificações, conforme Figura 48.

Figura 48 - Barra de ferramentas para desenvolvimento de Designer App

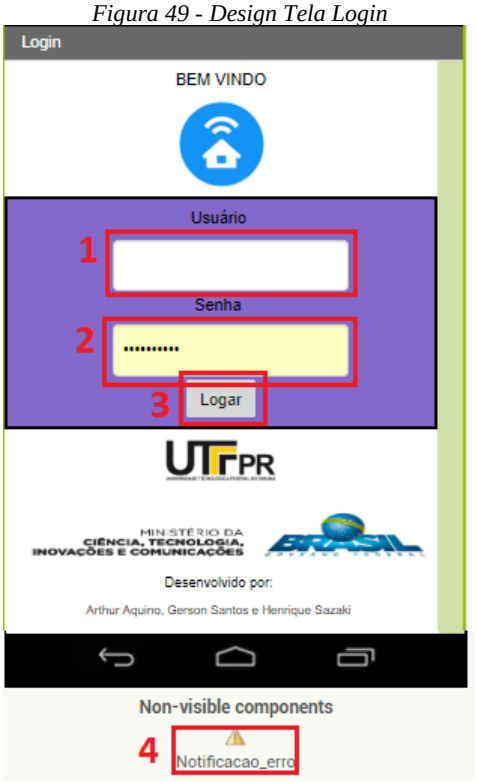


Fonte: AppInventor

3.3.1 Tela Login

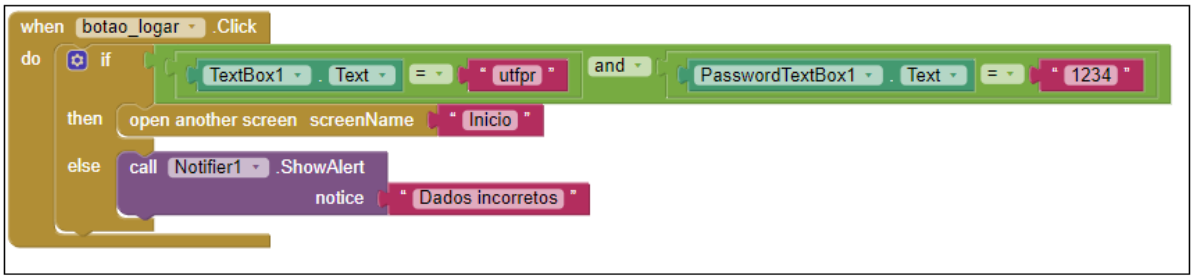
O *design* da tela de login, Figura 49, é programado conforme desenvolvimento em blocos mostrados na Figura 50, ou seja, quando *(when)* o usuário clicar no botão “logar” (3), é feito a comparação dos textos inseridos nos campos de “usuário” (1) e “senha” (2). Se (*if*), as duas informações estiverem conforme programadas, isto é, no campo usuário estiver digitado “utfpr” e (*and*) no campo senha “1234”, então *(then)* abrirá a tela Início *(open another screen)*, se não estiverem corretos *(if else)* os dados, será emitido um *popup* de notificação *(ShowAlert)* (4) de alarme com a informação “Dados Incorretos”.

Figura 49 - Design Tela Login



Fonte: os autores

Figura 50 - Bloco de programação para login



Fonte: os autores

3.3.2 Tela Inicio

A tela de início, Figura 51, é aberta no App após o *login* correto, e a partir dela, o usuário pode escolher as ações (ativar ou desativar cargas), que deseja executar.

Como as aplicações necessitam de comunicação com a internet, para enviar as informações do aplicativo para o Arduino Ethernet Shield, é inserido o ícone da *Web1* (0) que ficará oculto para o usuário, e na tela de programação utiliza-se o comando *Set URL*, na sequência se indica qual o endereço a ser acessado quando apertado algum dos botões.

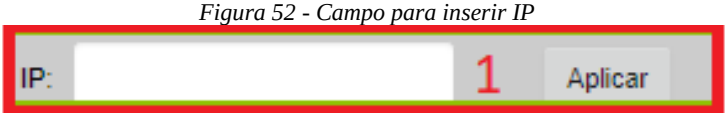
Para comunicação com a internet, o aplicativo precisa estar com o mesmo IP (1), Figura 52, que está endereçado no Arduino Ethernet Shield, e como é um IP público e variável, é necessário a inserção deste manualmente, neste projeto utilizou-se IPV6. Todas as ações tomadas nesta tela, precisam estar antes com o valor de IP definido.

O campo em branco foi nomeado como “Texto_IP” e o botão aplicar “B_IP”. Assim, quando o botão para aplicar IP for apertado, o texto inserido no campo em branco é armazenado na variável global, visto na Figura 53.

Figura 51 - Design Tela Inicio

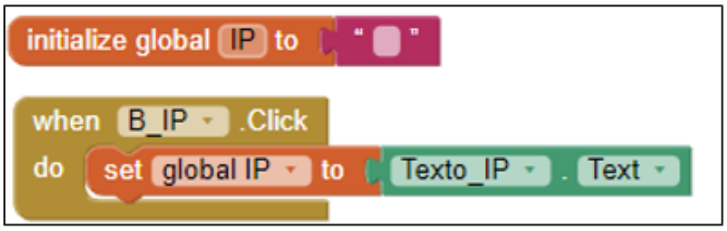
Fonte: os autores

Figura 52 - Campo para inserir IP



Fonte: os autores

Figura 53 - Blocos de programação para IP



Fonte: os autores

Para acesso a tela de iluminação, Figura 54, o App não executa nenhuma aplicação externa ao App, ao clicar neste botão ele abre a tela de acesso aos comandos de toda a iluminação, e ao abrir a nova tela, já carrega o valor do IP inserido, conforme programação de blocos demonstrado na Figura 55.

Figura 54 - Botão iluminação

Fonte: Icons8

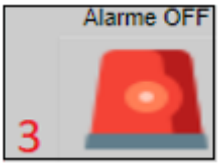
Figura 55 - Blocos de programação para acesso a tela de iluminação

Fonte: os autores

O botão (3) da Figura 56, tem como objetivo alterar o status de alarme da casa, cuja configuração inicial de status é *OFF*, ou seja, indica que o alarme está desativado, e após acionamento será alterado para "*ON*".

No bloco de programação, Figura 57, é verificado o status atual do alarme (*if* ou *if else*), e a depender do status atual, altera o endereço da página a ser carregada e atualizada para ativar a porta correspondente no Arduino, conforme detalhado no capitulo 3.2 (Programação do Arduino).

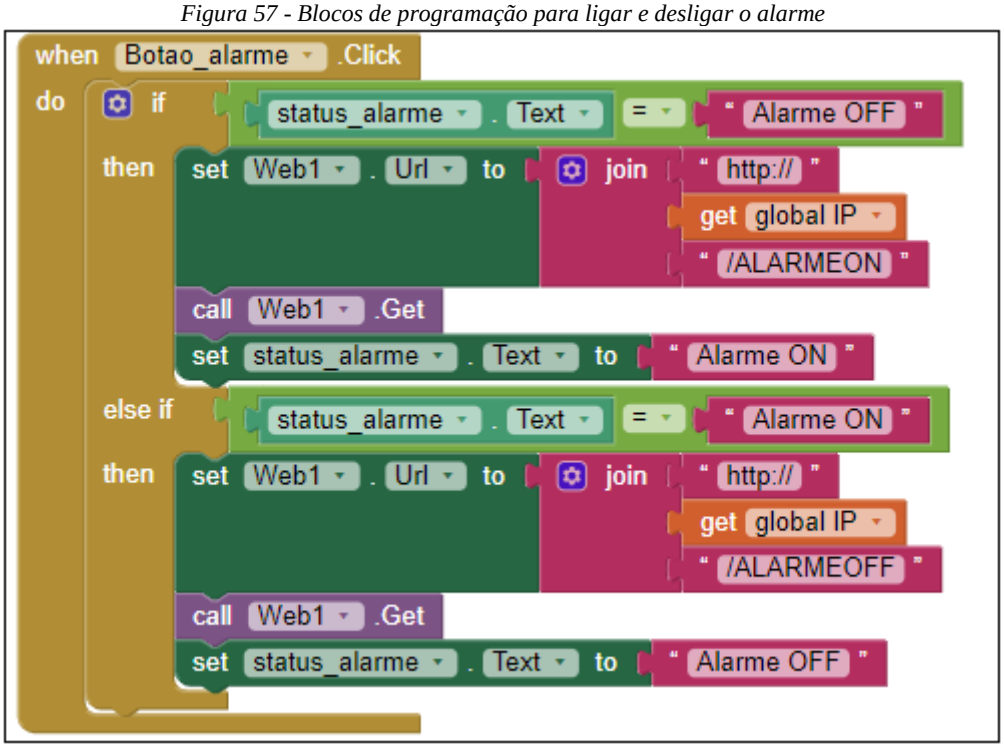
Figura 56 - Botão de alarme



Fonte: Icons8

Por exemplo, se o IP inserido for 2001:f016:ac5:fe1:80a:30e:8c9:cdb0, o endereço para ativar o alarme será: “http://” + Global IP “2001:f016:ac5:fe1:80a:30e:8c9:cdb0” + /ALARMEON”, ou seja, http://[2001:f016:ac5:fe1:80a:30e:8c9:cdb0]/ALARMEON. Para desativar, é acessado http://[2001:f016:ac5:fe1:80a:30e:8c9:cdb0]/ALARMEOFF.

Figura 57 - Blocos de programação para ligar e desligar o alarme



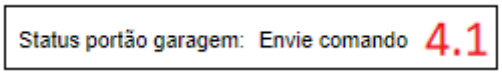
Fonte: os autores

Os dois botões do ícone (4) da Figura 58, tem como objetivo abrir e fechar o portão da garagem e abaixo do ícone, é possível saber o status atual do portão, visto na Figura 59, aberto ou fechado, o procedimento novo desta programação, na Figura 60, é que o texto de status atual do portão altera de cor, conforme a mensagem exibida, onde verde significa que o portão está aberto, e vermelho, fechado.

Figura 58 - Botão para acionamento do portão da garagem

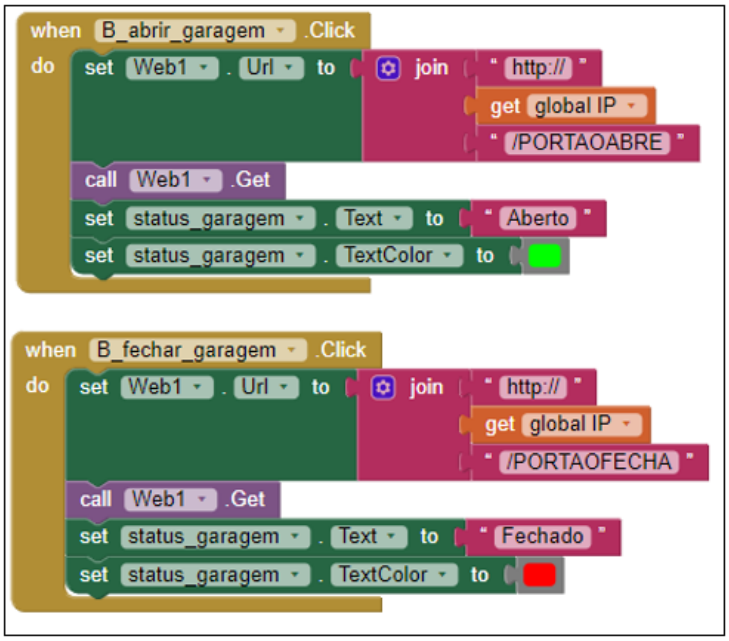
Fonte: Icons8

Figura 59 - Status portão garagem



Fonte: os autores

Figura 60 - Blocos de programação para controle do portão da garagem



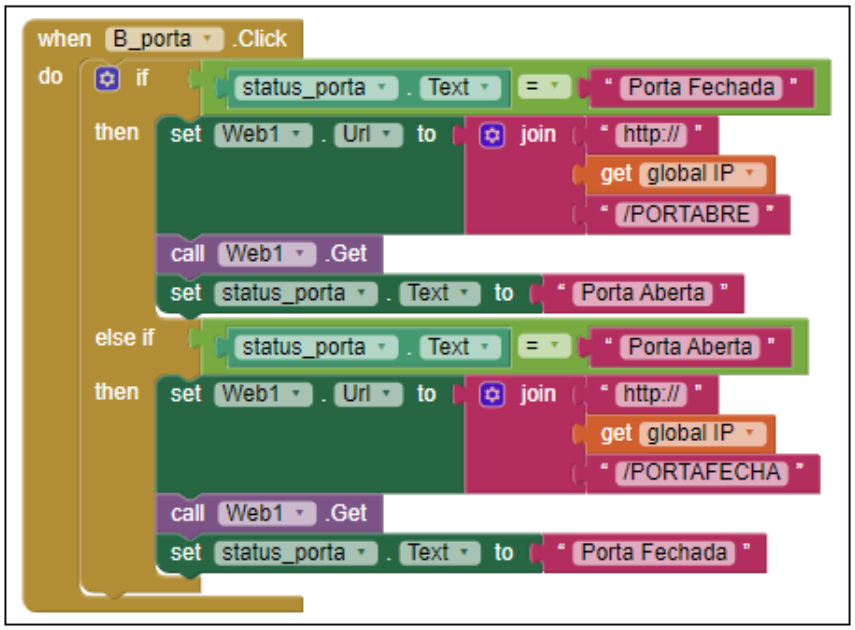
Fonte: os autores

Para abrir e fechar a porta da casa, é utilizado o botão representado na Figura 61, e conforme programação descrita na Figura 62, funciona da mesma forma que o botão alarme.

Figura 61 - Botão para abrir e fechar porta

Fonte: os autores

Figura 62 - Bloco de programação para porta

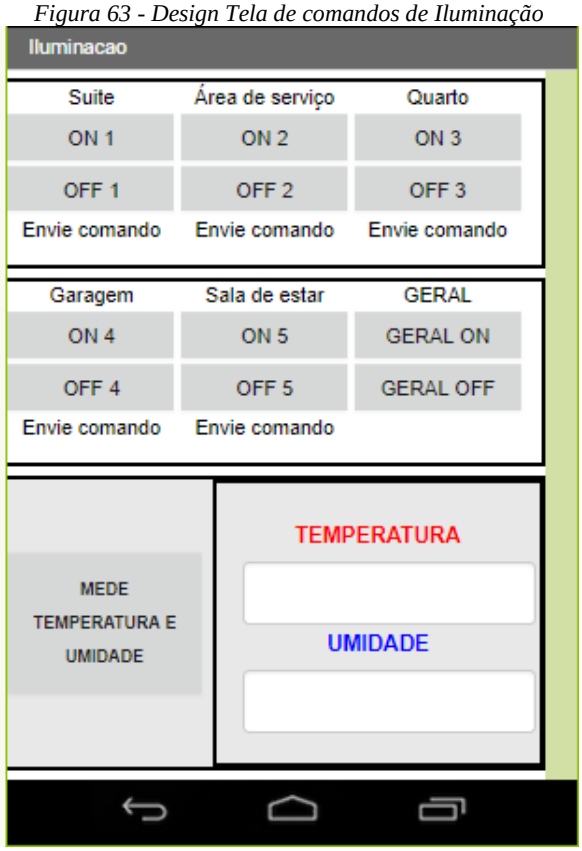


Fonte: os autores

3.3.3 Tela: Iluminação

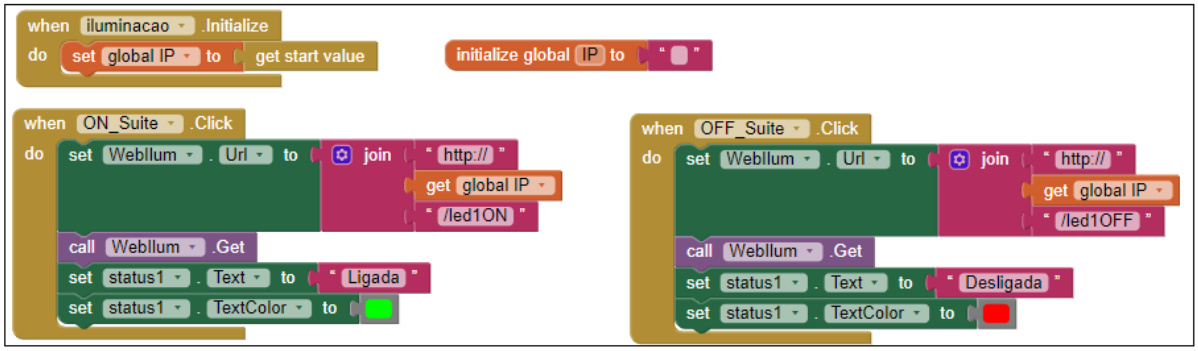
A terceira tela desenvolvida, Figura 63, é dedicada a fazer o controle de toda a iluminação da casa bem como informar os dados de temperatura e umidade do ambiente e para a programação de iluminação dos ambientes, conforme Figura 64 e Figura 65, é possível acender o respectivo led pelo botão ON e apagar com o botão OFF, e obter pelo botão da Figura 66, a leitura de temperatura e umidade dos dados climáticos locais, conforme apresentado na programação da Figura 67.

Figura 63 - Design Tela de comandos de Iluminação



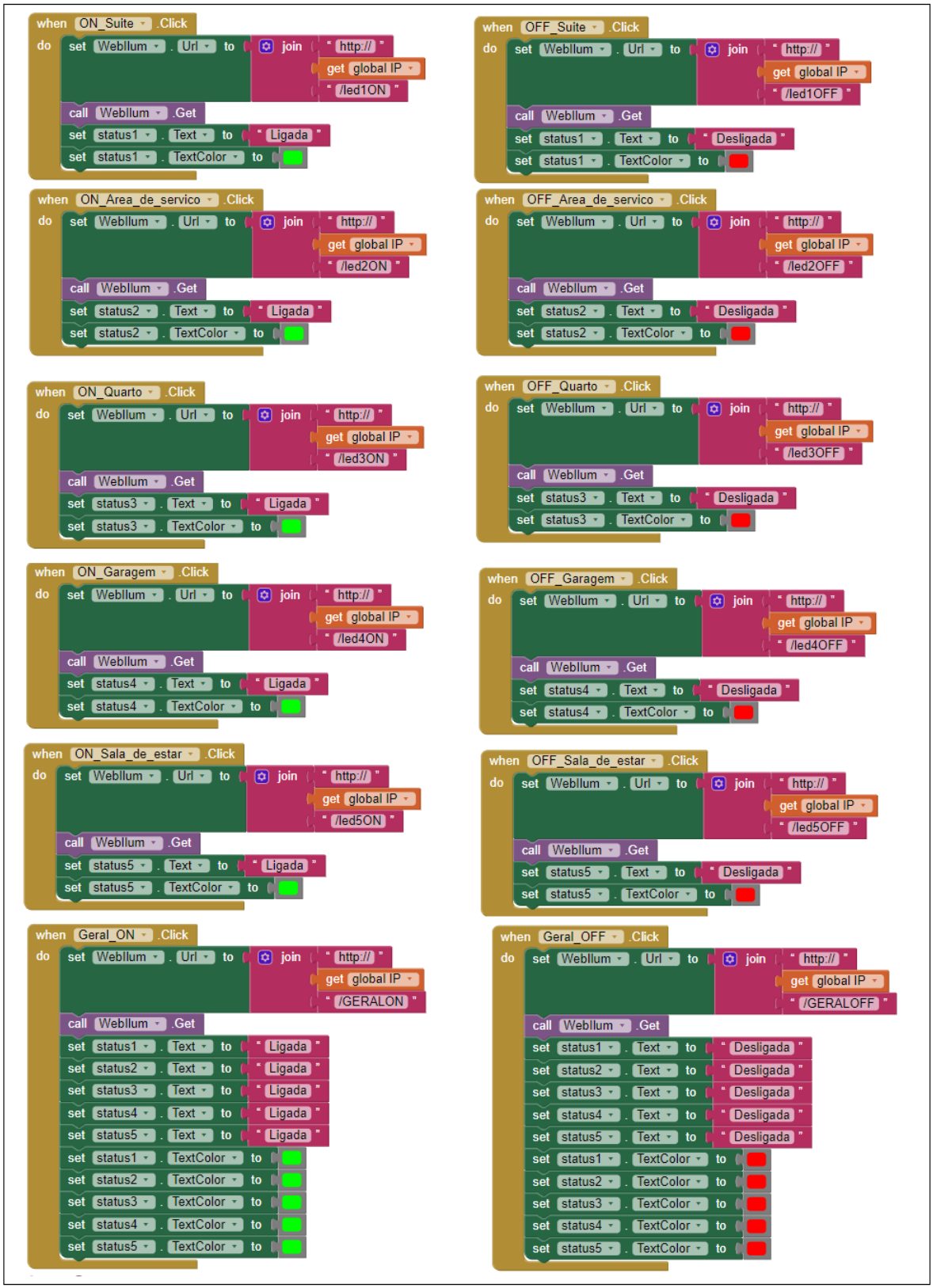
Fonte: os autores

Figura 64 - Bloco de programação para iniciar tela de iluminação e comando de iluminação da suíte.



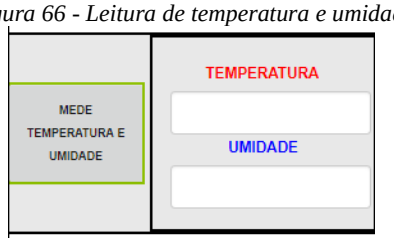
Fonte: os autores

Figura 65 - Bloco de programação para acionamento de todos os Leds



Fonte: os autores

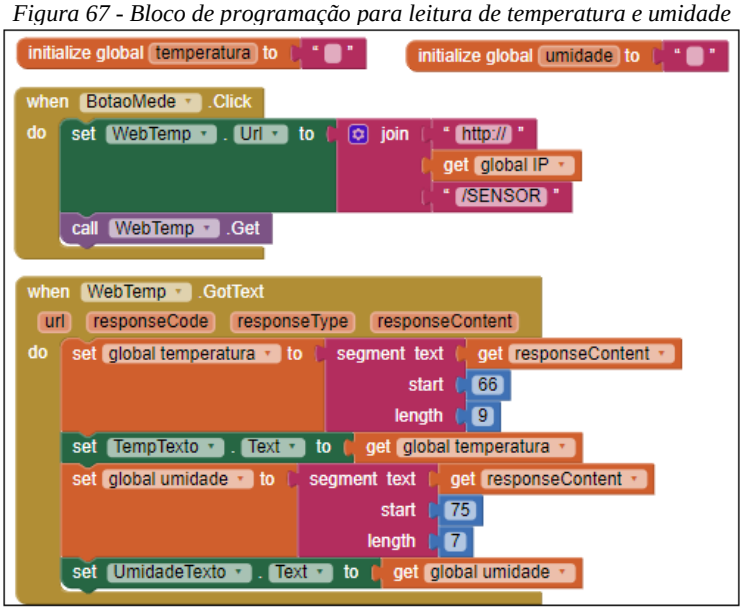
Figura 66 - Leitura de temperatura e umidade



Fonte: os autores

É importante destacar que a página WEB para leitura de temperatura e umidade não é em branco, como os demais, ele já contém uma informação em HTML, conforme Figura 68, por isso foi utilizado o bloco "*when WebTemp.GotText*". No HTML descrito, são destinados 9 caracteres, onde o primeiro deles está na posição 66, incluindo espaços e termina no caractere de posição 64, item 1 da Figura 68, para informações de temperatura, e o mesmo ocorre com a umidade, que inicia no caractere 75 e tem tamanho de 7 caracteres, conforme item 2 da Figura 68 e também apresentado pelos blocos azuis na Figura 67.

Figura 67 - Bloco de programação para leitura de temperatura e umidade



Fonte: os autores

Figura 68 - Código HTML para temperatura e umidade

```
<meta http-equiv=Content-Type content=text/html; charset=utf-8>\n
23.10\n
°C\n
72\n
%
```

1

2

Fonte: os autores

O aplicativo consegue enviar e receber informações do Arduino. Porém não consegue requisitar automaticamente IP válido do roteador, fazendo com que o usuário tenha que inserir um IP. O aplicativo também não possui um sistema para cadastro de novos usuários e não recebe confirmação do recebimento do comando por parte do Arduino.

3.4 CONSIDERAÇÕES FINAIS

O controlador escolhido, foi o Arduino Uno, que possui 14 pinos digitais (0 a 13) e 6 analógicos (A0 a A5), porém 7 pinos possuem uso específico: pinos 0 e 1 são necessários para transferência de dados entre o Arduino e o PC; 10, 11, 12 e 13 são usados para comunicação entre o Arduino Ethernet Shield e o Arduino Uno, e o pino 4 é utilizado para selecionar o cartão SD. Dessa forma, sobraram 13 pinos para controle das cargas. Cada pino suporta até 40 mA e 5 V, portanto não é possível instalar cargas pesadas diretamente ao Arduino. A quantidade de pinos que o Arduino Uno possui foi o suficiente para o controle de 5 leds de iluminação interna, 1 sensor de temperatura, 1 trava eletromagnética para porta e um mini servo motor para acionamento do portão em uma maquete com cinco cômodos (suíte, área de serviço, quarto, garagem e sala de estar), sendo que sobraram 4 pinos.

Houveram alguns momentos em que o microcontrolador parou de responder, num determinado acionamento de carga através do relé de um canal o shield de Ethernet resetava, para solucionar tal falha foi posto um diodo polarizado inversamente, foi identificado que havia uma interferência eletromagnética na desmagnetização da bobina do relé de acionamento.

Em relação a comunicação e transferência de dados via Internet, durante o desenvolvimento do projeto, surgiu a dificuldade referente ao protocolo de comunicação que pudesse integrar o Arduino com o aplicativo de celular, uma vez que muitos provedores de Internet em Curitiba e região não oferecem IPv4 públicos válidos para seus clientes, notou-se então, que seria necessário o uso de IPv6 no Arduino Ethernet Shield, porém esta placa suporta nativamente somente o IPv4. A solução deste problema veio através da biblioteca Ethersia criada por Nicholas Humfrey, que possibilita a criação de um servidor HTTP com IPv6 no Arduino Ethernet Shield, e isto não seria possível caso o Arduino não fosse uma plataforma aberta.

Os equipamentos utilizados neste TCC, Arduino Uno e Ethernet Shield, custaram juntos R$100,00, o que é relativamente baixo para os benefícios que o mesmo trás.

4 CONCLUSÃO

Neste TCC procurou-se apresentar um estudo sobre a existência dos sistemas de controle e automação residencial, foram automatizadas cargas como lâmpadas, relés e motores.

Foi realizado um trabalho de pesquisa sobre as vantagens dos diferentes sistemas de controle sem fio existente no mercado, abordado tecnologias como o 6LoWPAN e o ZigBee.

A partir do conhecimento levantado sobre a diferentes tecnologias, foi implementado uma análise do custo benefício do controlador que foi utilizado neste trabalho.

Apresentado as características técnicas do Arduino UNO, como já comentado no escopo do trabalho.

O Arduino usa um sistema linguagem de programação tipo C e para a criação do aplicativo foi aplicada a plataforma desenvolvida pelo MIT chamada App Inventor, que possibilita a criação de aplicativos totalmente funcionais para smartphones e tablets.

Após a elaboração e o projeto do protótipo foi adquirido uma maquete do modelo de um de residência padrão, que melhor representasse uma casa de 100 m², a maquete escolhida possuía 6 cômodos, sendo dois quartos, área de serviço, sala, banheiro e garagem.

Foi feito a montagem e implementado os sistemas de controle na maquete, foi instalado um led por cômodo representando o sistema de iluminação da casa, feito a instalação de um mini servo motor no portão da garagem e uma trava eletromagnética da porta de entrada desta maquete.

A implementação deste projeto cujo fim é a promoção do conhecimento adquirido durante sua elaboração, pode se dar por meio de plataformas de uso profissional. Os recursos usados e aplicados aqui foram de base educacional, todos sendo numa plataforma aberta.

Como recomendação para projetos futuros, é desejável a criação de uma placa que suporte correntes mais elevada, que possua mais confiabilidade em termos de operação contínua e consiga requisitar automaticamente um IP válido, e em relação ao desenvolvimento do aplicativo, um sistema de cadastro de novos usuários pode ser implementado, bem como um banco de dados que registre os últimos comandos enviados e recebidos, para que todos os dispositivos estejam sempre sincronizados com os status reais das cargas.

REFERÊNCIAS

ARDUINO. **Arduino UNO Board Anatomy**. Disponível em: <https://www.arduino.cc/en/Guide/BoardAnatomy>. Acesso em: 29 mar. 2018.

ARDUINO. **What is Arduino?** Disponível em: <https://www.arduino.cc/en/Guide/Introduction?setlang=en>. Acesso em: 29 mar. 2018.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 12721: **Caracterização dos projetos** – padrão conforme Rio de Janeiro 2005. Disponível em: <https://goo.gl/oNkVVr>. Acesso em: 4 out. 2017.

AXIS, Comunications. **Network Technologies**. Disponível em: <https://www.axis.com/pt-br/learning/web-articles/technical-guide-to-network-video/lan-and-ethernet>. Acesso em: 16 abr. 2018.

AZEVEDO, Tiago. **Roteamento ZigBee**. CPE 825 – Roteamento em Redes de Computadores. Programa de Engenharia de Sistemas e Computação, Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia, Universidade Federal do Rio de Janeiro, 2006. 8p. Disponível em: <https://www.gta.ufrj.br/ensino/CPE825/2006/resumos/TrabalhoZigbee.pdf> Acesso em: 20 maio 2018.

BARLOW, J.; GANN, D. **A Changing Sense of Place: Are Integrated IT Systems Reshaping the Home?** Durham, abr. 1998. 23 p. Disponível em: <www.sussex.ac.uk/spru/documents/sewp18>. Acesso em: 10 abr. 2018.

CRAIG, William C. **Zigbee: "Wireless Control That Simply Works"**. ZMD America, Inc. 2004. 7p. Disponível em: <https://bit.ly/2J0eBtO>. Acesso em: 22 maio 2018.

FGV. **Número de smartphones no Brasil chega a 168 milhões**, diz estudo. *Folha de São Paulo,* São Paulo, abr. 2016. Disponível em: < https://goo.gl/Yh7NdJ>. Acesso em: 4 out. 2017.

FOROUZAN, B. A. **Data Communications and Networking**. 5. ed. New York: McGraw-Hill Companies, 2012.

FOROUZAN, B. A. **Comunicação de Dados e Redes de Computadores**. 4. ed. Porto Alegre: AMGH, 2010.

GOMES, Carles; PARADELLS, Josep. **Wireless Home Automation Networks: A Survey of Architectures and Technologies**. IEEE Comunications Magazine, jun. 2010. Disponível em: <https://ieeexplore.ieee.org/document/5473869>. Acesso em: 9 abr. 2018.

GONNOT, Thomas; SANNIE, Jafar. **User Defined Interactions between Devices on a 6LoWPAN Network for Home Automation**. Chicago, Illinois, Illinois Institute of Technology, 2014. 4 p. Disponível em: <https://ieeexplore.ieee.org/document/6918618>. Acesso em: 15 abr. 2018.

HARPER, Richard. **Inside the Smart House**. Londres: Springer, 2003. 264 p.

IANA, **Internet Assigned Numbers Authority**. IANA IPv4 Address Space Registry. Disponível em: < https://goo.gl/f7PX11 >. Acesso em: 25 out. 2018

IBGE. Coordenação de Trabalho e Rendimento. **Acesso à Internet e à Televisão e Posse de Telefone Móvel Celular para Uso Pessoal: 2015**, Rio de Janeiro, 2016. Disponível em: <https://goo.gl/KhHRiR >. Acesso em: 5 out. 2017.

IEEE P802.15 Working Group, **IEEE Standard 802.15.4 for Low-Rate Wireless Networks**, abr. 2016. Disponível em: < https://goo.gl/bvcx1u>. Acesso em: 22 abr. 2018.

IPv6.br, **Endereçamento**. Núcleo de informação e Coordenação do ponto BR – NIC.br., maio.2012. Disponível em: <http://ipv6.br/post/enderecamento/>. Acesso em 03 out 2018.

HUI, J.; THUBERT, P. **Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks**. Internet Engineering Task Force. RFC 6282. 2011. Disponível em: <https://tools.ietf.org/html/rfc6282>. Acesso em: 22 maio 2018.

MISRA, Sudip; GOSWAMI Sumit. **Network Routing Fundamentals, Applications, and Emerging Technologies**. 1 ed. Pondicherry: Wiley, Ano. 431 p.

MURATORI, José R.; BÓ, Paulo H. D. **Automação residencial: histórico, definições e conceitos**, Revista O Setor Elétrico, abr. 2011. Disponível em: <https://goo.gl/F95edA >. Acesso em: 4 out. 2017.

NIERO, Jamille. **Crise não derruba mercado de automação**. Federação do Comércio de Bens, Serviços e Turismo do Estado de São Paulo**,** São Paulo, maio 2017. Disponível em: < https://goo.gl/Ydnuza >. Acesso em: 23 out. 2017.

OLSSON, Jonas. **6LoWPAN demystified**. Texas Instruments, out. 2014. Disponível em: <www.ti.com/lit/wp/swry013/swry013.pdf >. Acesso em: 20 maio 2018.

RICQUEBOURG, V. et al. **The Smart Home Concept : our immediate future**. IEEE. 2006. Disponível em: <ieeexplore.ieee.org/document/4152762>. Acesso em: 18 maio 2018.

TERUEL, E. C. **Uma proposta de framework para sistemas de automação residencial com interface para WEB**. 2008. 158 f. Dissertação (Mestrado em Tecnologia: Gestão, Desenvolvimento e Formação) – Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2008.

THOMSEN, Adilson. **O que é Arduino, FILIEPEFLOP**. Disponível em: < https://goo.gl/bfmmaA >. Acesso em: 4 out. 2017.

YANG, Shuang-Hua. **Wireless Sensor Networks Principles, Design and Applications.** Londres: Springer, 2014. 293 p.

APÊNDICE A – ARQUIVOS DE PROGRAMAÇÃO ARDUINO SMARTHOUSE

```
// ********** DECLARACAO DAS BIBLIOTECAS **********
#include <Servo.h>                  //Servo.
#include <EtherSia.h>               //Biblioteca do servidor
#include <DHT.h>                    //Sensoriamento de temperatura e
umidade.
#define DHTPIN 2                    //Sensor DHT22 conectado ao pino
digital 2 do Arduino Uno.
#define DHTTYPE DHT22               //Define o tipo de sensor.


// ****************************************
DHT dht(DHTPIN, DHTTYPE);           //Inicializa o sensor.
EtherSia_W5100 ether;               //Seleciona a interface, o W5100
corresponde ao Arduino Ethernet Shield.
HTTPServer http(ether);             //Define servidor HTTP.

Servo;                         //Cria objeto tipo Servo

int led0 = A0;                      //Leds conectados nos pinos A0 a A5.
int led1 = A1;                      //A0 até A5 são pinos analógicos.
int led2 = A2;
int led3 = A3;
int led4 = A4;
int porta = A5;                      //Fechadura da porta no pino
analogico A5.
int portao =  3;                     //Motor do portão no pino 3.
int servoAngle = 90;                 //Posição do servo que começa em 90
graus.
int sensor = 5;                      //Sensor magnético da porta no pino
digital 7.
int est_sensor = 0;                  //Estado do sensor magnético da
porta.
float temp = 0;                      //Temperatura do sensor DHT22.
int umid = 0;                        //Umidade do sensor DHT22.
int alarme;                          //Alarme.
int intruso;                         //Intruso.
unsigned long valoranterior = 0;     //Ultima vez que o led alterou
estado.
const long intervalo = 1000;         //Intervalo do pisca do alarme.
int ledestado = LOW;

void setup(){
  pinMode(4,OUTPUT);                 //Pino do CARTÃO SD.
  digitalWrite(4,HIGH);              //ANULA LEITURA E ESCRITA NO CARTÃO
SD.
  servo.attach(portao);              //ASSOCIAÇÃO DO PINO DIGITAL AO
OBJETO DO TIPO SERVO.

  pinMode(led0,OUTPUT);              //declara os pinos como SAÍDA ou
ENTRADA.
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(led4,OUTPUT);
  pinMode(portao,OUTPUT);
  pinMode(porta,OUTPUT);
  pinMode(sensor,INPUT);
  dht.begin();
```

Fonte: os autores

```
  MACAddress macAddress("46:29:b1:87:8b:08"); //Seta o endereço MAC.
  ether.setGlobalAddress("2041:0:140F::875B:131B");//Seta o IPv6 Global.
  ether.begin(macAddress); //Inicializa o servidor com o endereço
MAC.}//void setup()

void loop(){
  umid = dht.readHumidity();            //Lê umidade e armazena em umid.
  temp = dht.readTemperature();         //Lê temperatura e armazena em
temp.
  est_sensor = digitalRead(sensor);     //Le o valor do sensor e
armazena no est_sensor
  sensores();                           //Chama a função sensores, faz o
controle do alarme.
  servomotor();                         //Chama função servomotor, faz
controle do motor do portão.
  tratamento();                         //Chama a função tratamento, faz
o controle da porta e iluminação.
  ether.receivePacket();                //Servidor processa o pacote
}//void loop
```

Fonte: os autores

```
void tratamento()
{

if(http.isGet(F("/GERALON")))       // Se a página pedida for
"/geralON"...
{
  http.printHeaders(http.typeHtml); //Imprime o cabeçalho
  http.sendReply();
  digitalWrite(led0, HIGH);         // liga o pino do relé
  digitalWrite(led1, HIGH);         // liga o pino do relé
  digitalWrite(led2, HIGH);         // liga o pino do relé
  digitalWrite(led3, HIGH);         // liga o pino do relé
  digitalWrite(led4, HIGH);         // liga o pino do relé
}

if(http.isGet(F("/GERALOFF")))      // Se a página pedida for
"/geralOFF"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led0, LOW);          // desliga o pino do relé
  digitalWrite(led1, LOW);          // desliga o pino do relé
  digitalWrite(led2, LOW);          // desliga o pino do relé
  digitalWrite(led3, LOW);          // desliga o pino do relé
  digitalWrite(led4, LOW);          // desliga o pino do relé
}
//*****************************************************************

//Primeiro Grupo de LED
if(http.isGet(F("/led1ON")))        // Se a página pedida for
"/led1ON"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
```

Fonte: os autores

```
 digitalWrite(led0, HIGH);          // liga o pino do led
}

if(http.isGet(F("/led1OFF")))       // Se a página pedida for
"/led1OFF"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led0, LOW);          // desliga o pino do led
}
//*********************************************************************

//Segundo Grupo de LED
if(http.isGet(F("/led2ON")))        // Se o parâmetro recebido no
celular for "/led2ON"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led1, HIGH);         // liga o pino do led
}

if(http.isGet(F("/led2OFF")))       // Se a página pedida for
"/led2OFF"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led1, LOW);          // desliga o pino do led
}
//*********************************************************************
**

//Terceiro Grupo de LED
if(http.isGet(F("/led3ON")))        // Se o parâmetro recebido no
celular for "/led3ON"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led2, HIGH);         // liga o pino do led
}

if(http.isGet(F("/led3OFF")))       // Se o parâmetro recebido no
celular for "/led3OFF"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led2, LOW);          // desliga o pino do led
}
//*********************************************************************
**

//Quarto Grupo de LED
if(http.isGet(F("/led4ON")))        // Se o parâmetro recebido no
celular for "/led4ON"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led3, HIGH);         // liga o pino do led
}
```

Fonte: os autores

```
if(http.isGet(F("/led4OFF")))        // Se o parâmetro recebido no
celular for "/led4OFF"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led3, LOW);           // desliga o pino do led
}
//*************************************************************************
**

//Quinto Grupo de LED
if(http.isGet(F("/led5ON")))         // Se o parâmetro recebido no
celular for "/led5ON"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led4, HIGH);          // liga o pino do led
}

if(http.isGet(F("/led5OFF")))        // Se o parâmetro recebido no
celular for "/led5OFF"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(led4, LOW);           // desliga o pino do led
}
//*************************************************************************
**

if(http.isGet(F("/PORTABRE")))       // Se o parâmetro recebido no
celular for "/portabre"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(porta, HIGH);         // Porta aberta
}

if(http.isGet(F("/PORTAFECHA")))
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  digitalWrite(porta, LOW);          // Porta fechada
}
//*************************************************************************
***
}// fim do "tratamento"
```

Fonte: os autores

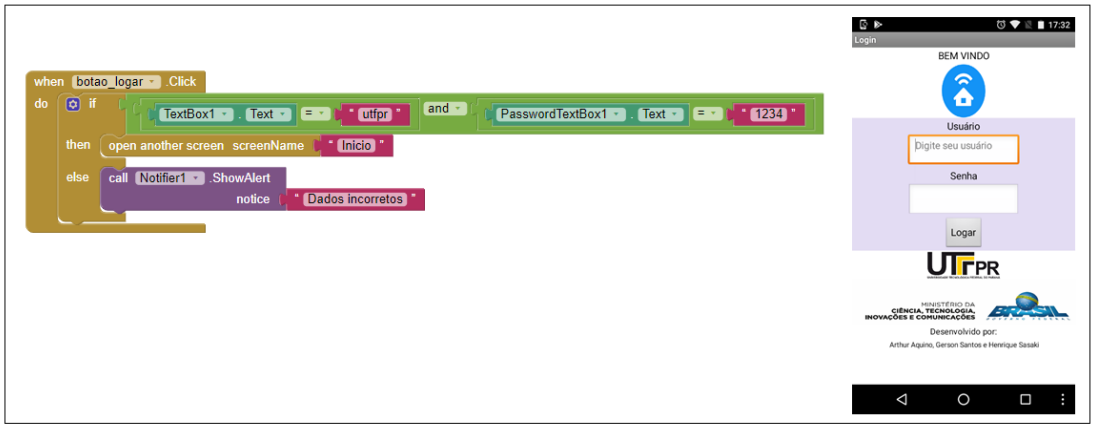
```
void sensores(){
  if(http.isGet(F("/SENSOR"))){
    http.printHeaders(http.typeHtml);
    http.println("<meta http-equiv=Content-Type content=text/html;
charset=utf-8>"); //Habilita caracteres especiais
    http.println(temp);
    http.println("°C");
    http.println(umid);
    http.println("%");
    http.sendReply();
  }// /SENSOR

  //**********************************************************************
***************
unsigned long valoratual = millis();
  if(http.isGet(F("/ALARMEON")))
  {
  http.printHeaders(http.typeHtml);
  http.sendReply();
  alarme=1;
  }// /ALARMEON

  if(http.isGet(F("/ALARMEOFF")))
  {
    http.printHeaders(http.typeHtml);
    http.sendReply();
    alarme=0;
    intruso=0;
    digitalWrite(led0, HIGH);         // liga o pino do relé
    digitalWrite(led1, HIGH);         // liga o pino do relé
    digitalWrite(led2, HIGH);         // liga o pino do relé
    digitalWrite(led3, HIGH);         // liga o pino do relé
    digitalWrite(led4, HIGH);         // liga o pino do relé
  }// /ALARMEOFF

  if(est_sensor == HIGH && alarme)  //Se alarme e sensor forem igual a 1
  {
  intruso=1;
  }// /est_sensor == HIGH && alarme

  if(intruso == 1){
    if (valoratual - valoranterior >= intervalo){
      valoranterior = valoratual;
      if (ledestado == LOW){
        ledestado = HIGH;
        }// ledestado == LOW
      else{
        ledestado = LOW;
        }// else
        digitalWrite(led0, ledestado);
        digitalWrite(led1, ledestado);
        digitalWrite(led2, ledestado);
        digitalWrite(led3, ledestado);
        digitalWrite(led4, ledestado);
    }// valoratual - valoranterior >= intervalo
  }// intruso==1

}// void sensores
```

Fonte: os autores

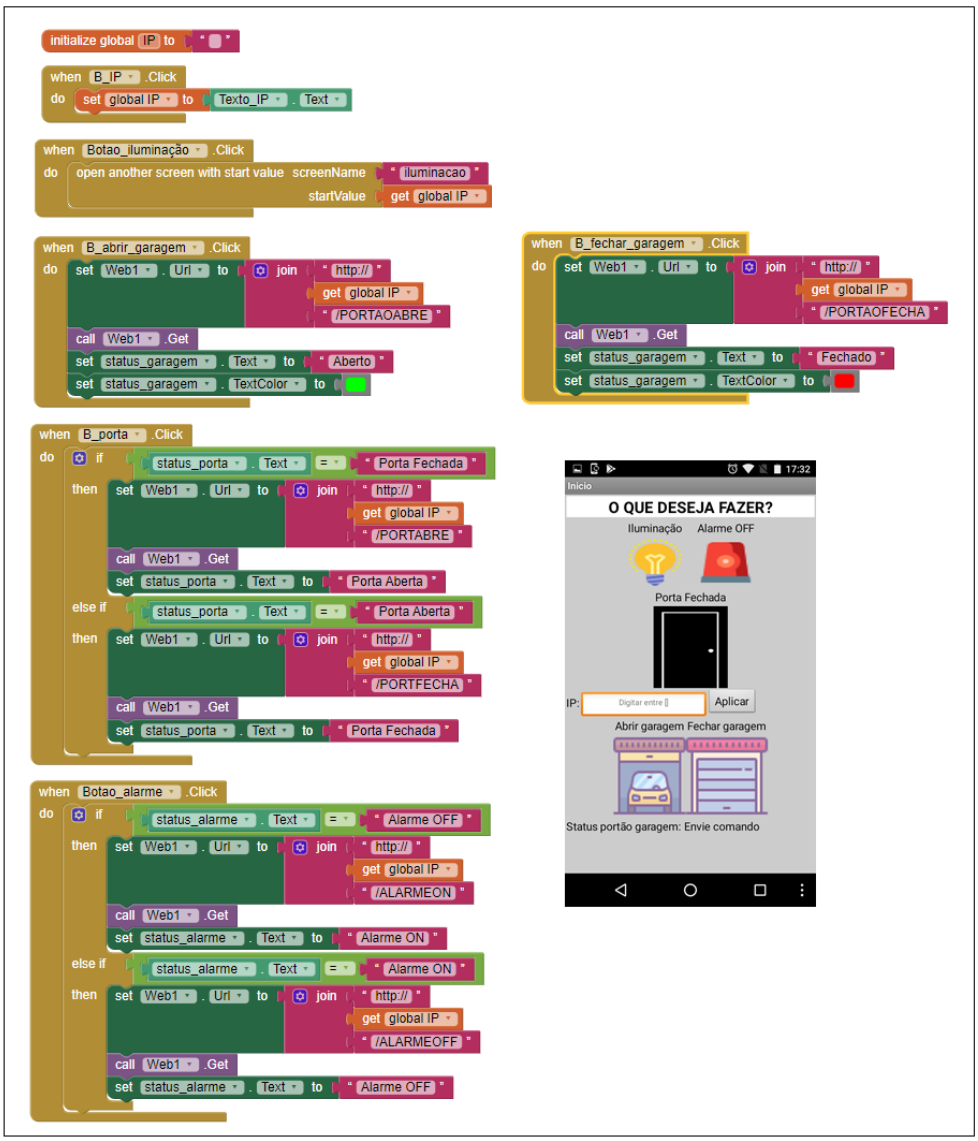
```
void servomotor()
{
//servo motor no relé 8

if(http.isGet(F("/PORTAOABRE")))  // SE o parâmetro recebido no browser
for "/PORTAOABRE"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  for(servoAngle = servoAngle; servoAngle > 0; servoAngle--){  //PARA
ANGULO IGUAL A 90, ENQUANTO O ANGULO FOR MAIOR QUE 0, DECREMENTA O
ANGULO
  servo.write(servoAngle); //GIRA O SERVO NO VALOR ATUAL DA VARIÁVEL
servoAngle
  delay(30); //INTERVALO DE 35 MILISSEGUNDOS
  }//for servoAngle--

}//if PORTAOABRE

if(http.isGet(F("/PORTAOFECHA"))) // SE o parâmetro recebido no browser
for "/PORTAOABRE"...
{
  http.printHeaders(http.typeHtml);
  http.sendReply();
  for(servoAngle = servoAngle; servoAngle < 90; servoAngle++){  //PARA
ANGULO IGUAL A 180, ENQUANTO O ANGULO FOR MAIOR QUE 0, DECREMENTA O
ANGULO
  servo.write(servoAngle); //GIRA O SERVO NO VALOR ATUAL DA VARIÁVEL
servoAngle
  delay(30); //INTERVALO DE 35 MILISSEGUNDOS
  }//for servoAngle++
}//if /PORTAOFECHA
}//servomotor
```

Fonte: os autores

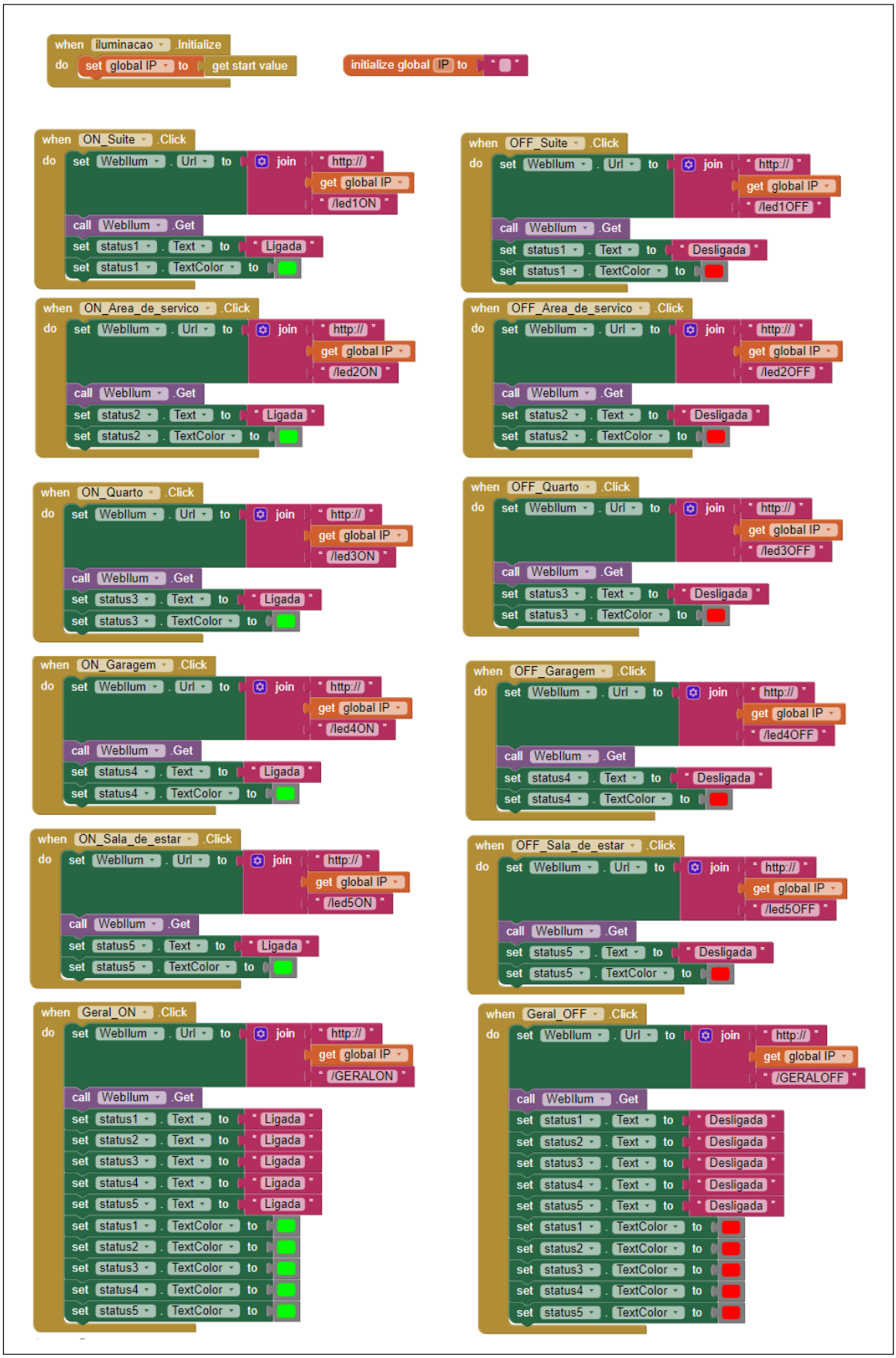
APÊNDICE B – ARQUIVOS DE PROGRAMAÇÃO APLICATIVO SMARTHOUSE



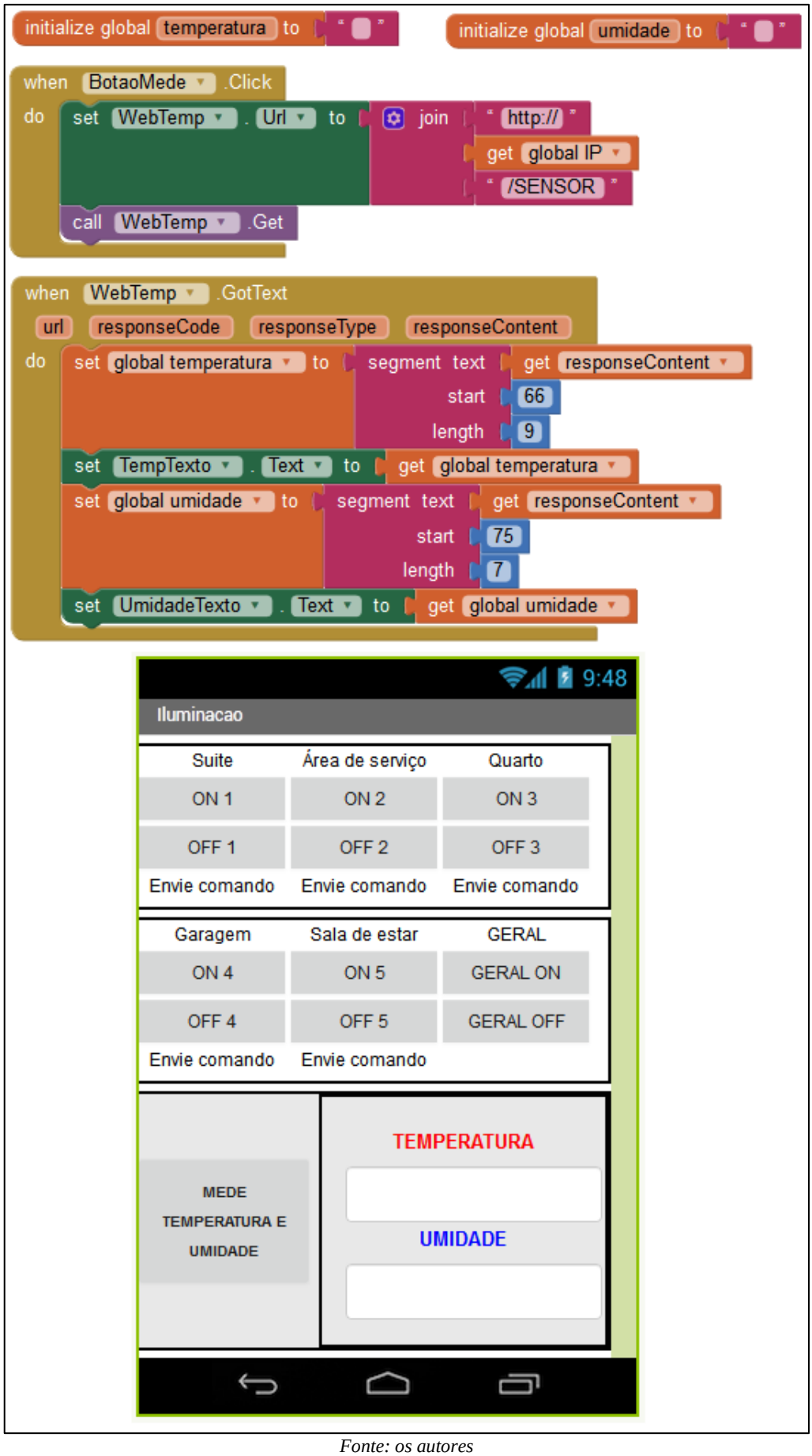
Fonte: os autores



Fonte: os autores



Fonte: os autores



Fonte: os autores