UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

Automação Residencial utilizando Arduíno e SO Android

Sandro Moura da Silveira

Thadeu Santos Silva Gonçalves

Orientador

Leonardo Luiz Alencastro Rocha

RIO DE JANEIRO, RJ – BRASIL

JULHO DE 2016

Catalogação informatizada pelo autor

S587 Silveira , Gonçalves, Sandro Moura da , Thadeu Santos Silva Automação Residencial utilizando Arduíno e SO Android / Sandro Moura da , Thadeu Santos Silva Silveira , Gonçalves. -- Rio de Janeiro, 2016. 63 Orientador: Leornado Luiz Alencastro Rocha. Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2016. 1. Arduino. 2. Android. 3. Automação. 4. Residencial. I. Rocha, Leornado Luiz Alencastro, orient. II. Título.

Automação Residencial utilizando Arduíno e SO Android

Sandro Moura da Silveira
Thadeu Santos Silva Gonçalves

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Leonardo Luiz Alencastro Rocha (UNIRIO)

_____

Geiza Maria Hamazaki da Silva (UNIRIO)

_____

Luiz Amâncio Machado de Souza Junior (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.
JULHO DE 2016

Agradecimentos

Agradecemos o apoio fundamental de nossas famílias pela força, coragem, compreensão, incentivo nas horas difíceis, que foram muito importantes e fizeram entender que o futuro é feito a partir da constante dedicação no presente.

A Universidade Federal do Estado do Rio de Janeiro, aos docentes do curso de Sistemas de Informação pela troca de conhecimento e experiências em um ambiente criativo e amigável durante todo o curso.

Ao professor/orientador Leonardo Rocha pela oportunidade, suporte, incentivo e correções durante a elaboração desse projeto.

Aos amigos e colegas, essenciais em todas as etapas da graduação.

A todos que direta ou indiretamente fizeram parte de nossa formação, o nosso muito obrigado.

RESUMO

O avanço da tecnologia e o surgimento de diversas plataformas micro controladas aliado a facilidade de acesso vem aumentando o número de possibilidades de uso, entre eles o da automação residencial por proporcionar comodidade, praticidade e segurança. O trabalho objetiva a integração entre a plataforma micro controlada Arduíno e o sistema operacional Android, com a finalidade de controle para dispositivos residenciais. O projeto apresenta um sistema composto pelo microcontrolador Arduíno conectado a uma placa de rede, que possibilitará o controle de iluminação, sistemas de alarmes e dispositivos eletrônicos de uma residência através de um aplicativo desenvolvido para o sistema operacional Android.

Palavras-chave: “Arduíno”, “Android”, “Automação Residencial” .

ABSTRACT

The advancement of technology and the emergence of various microcontrolled plataforns, combined with easy access to them, is increasing the number of possible uses, including residential automation, by providing convenience, practicality and safety. The work aims to integrate the Arduino microcontroller with an Android operating system for the purpose of controlling home devices. The project features a system composed of the Arduino microcontroller connected to a network adapter, which will be able to control lighting, alarm systems and electronic devices of a residence through an application developed for Android operating system.

Keywords: “Arduino”, “Android”, “Residential Automation” .

Sumário

1 Introdução

1.1 Motivação

A sociedade tem vivido em meios a progressos tecnológicos na área de automação e sistemas de telefonia móvel. A cada dia se torna mais presente em nosso cotidiano e cada vez mais imprescindível na vida das pessoas.

Automatizar é substituir o trabalho humano ou animal por máquinas ou sistemas. No início a automação era focada principalmente nas industrias, que após ser aplicada em larga escala nas mais variadas áreas de produção houve um crescimento no mercado de automação de pequeno porte. Introduziu-se o uso de microcontroladores, incentivado pelo desenvolvimento econômico que aumentou o poder aquisitivo das pessoas e pela busca de conforto, praticidade e segurança. Celulares que hoje são comparados a computadores ajudam ainda mais no desenvolvimento do setor de automação.

“Automação residencial é uma coleção de equipamentos, sistemas e subsistemas, que mantêm habilidade para interagir entre si, permitindo o estabelecimento de funções independentes.” (MURATORI, 2004)

“É a atuação de dispositivos nas funções de elétrica, hidráulica e ar condicionado, permitindo o uso customizado de aparelhos elétricos e garantindo economia de energia elétrica e água.” (BOLZANI, 2004)

“Inclui o uso de equipamentos especializados que podem controlar lâmpadas, eletrodomésticos, aquecedores, ar condicionado, e perceber em que local da casa as pessoas estão.” (MEYER, 2004)

Agora automação residencial se apresenta como um novo mercado com potenciais ainda em desenvolvimento que passou a ser realidade no Brasil, e cada vez mais próspero e atraente, com soluções inteligentes e diferenciadas, voltadas para o conforto e acessibilidade do usuário.

1.2 Objetivos

Este projeto de graduação tem por objetivo apresentar a viabilidade da integração entre o microcontrolador Arduíno e smartphones com o sistema operacional Android, direcionados para controlar itens da residência. O usuário terá interação completa com os dispositivos do imóvel conectados ao microcontrolador, que através de uma conexão wireless podem ser controlados pelo aplicativo instalado em seu celular.

1.3 Organização do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, será desenvolvido da seguinte forma:

- Capítulo II: Base Tecnológica - Neste capítulo é apresentado o referencial teórico, apresentando as ferramentas e descrevendo as tecnologias utilizadas na construção do projeto.
- Capítulo III: Detalhamento do Projeto - Este capítulo trata da descrição do desenvolvimento do aplicativo para Android, da comunicação realizada entre o

software e microcontrolador, as funcionalidades do projeto e a montagem do sistema.

- Capítulo IV: Desenvolvimento - O capítulo trata especificamente dos métodos utilizados, contendo sua descrição e resultados esperados durante desenvolvimento do projeto.

- Capítulo V: Conclusões – Reúne as considerações finais, assinala as contribuições da pesquisa e sugere possibilidades de aprofundamento posterior.

2 Base Tecnológica

O projeto fez uso de diversos componentes, tanto hardware como softwares. O texto a seguir traz uma pequena descrição dos componentes físicos utilizados no projeto.

2.1 - Plataforma Arduíno

Arduíno é uma plataforma código aberto de prototipagem eletrônica, onde pode perceber o ambiente através de dispositivos ligados as portas de entrada de dados, e afetar seus arredores por meio de dispositivos motores e atuadores. As placas podem ser produzidas a mão ou compradas pré montadas (Shields), que permitem expandir a as funções do microcontrolador. Seu objetivo é a criação de ferramentas acessíveis, com baixo custo, flexíveis e fáceis de usar voltados para fins comerciais, domésticos ou móveis. Podendo ser utilizado de forma independente ou conectado a um computador que possa oferecer informações, recursos e serviços.

Existem diversos modelos de placas Arduíno, que variam em seu tamanho e quantidade de portas disponíveis para ligar as shields ou dispositivos. O que iremos utilizar no projeto é o modelo UNO, o mais recomendado para quem vai começar com eletrônica e codificação. Sendo a placa mais utilizada e documentada de toda família Arduíno.

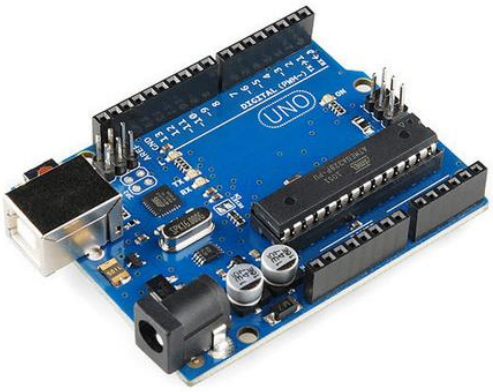

Figura 01 - Arduíno UNO

Uma placa de Arduíno UNO é composta por um microcontrolador ATmega 328 14 portas de digitais, 6 portas analógicas, além de uma interface USB que é utilizada para interligar-se ao hospedeiro com a finalidade de programá-lo ou de interagir em tempo real e uma tomada de energia. Das portas digitais, 6 delas possuem a funcionalidade de serem utilizadas como pseudo-analógicas, onde se torna possível o controle de dispositivos PWM (Pulse Width Modulation ou Modulação de Largura de Pulso).

Além disso possui um barramento de extensão, onde é possível utilizar seus pinos como fonte de alimentação para os dispositivos / sensores conectados ao microcontrolador. Disponibilizando ao usuário 3,3V, 5V e GND.

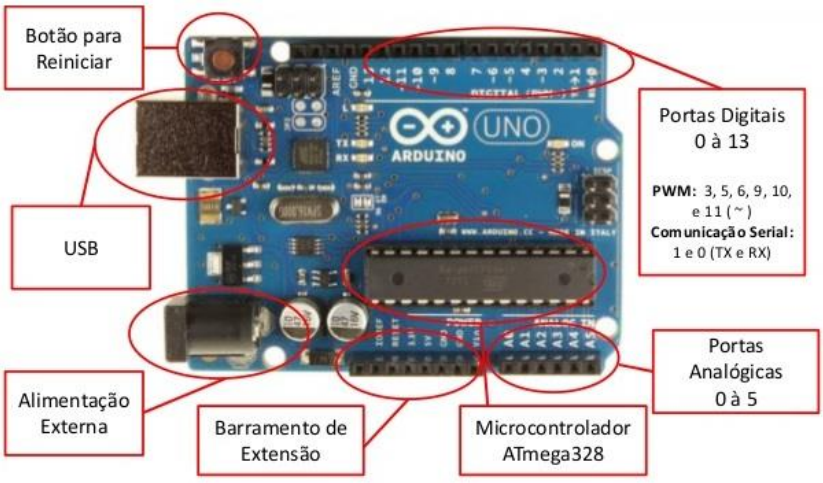

Figura 02 - Localização das portas e componentes Arduíno UNO

2.2 - Ethernet Shield

Controlar sensores ou enviar informações remotamente é um dos principais objetivos de que trabalha com microcontroladores, este modulo permite a comunicação entre o dispositivo Arduíno e outros dispositivos de rede de maneira fácil e rápida através do cabo RJ-45, possibilitando a leitura dos sensores e controle dos atuadores através de navegadores, computadores e celulares.

Durante a fase de planejamento do projeto, o grupo optou por utilizar o módulo ENC28J60 uma vez que existem bibliotecas (drivers), no caso a UIPEthernet, que tornam a sua utilização idêntica a da biblioteca padrão do Arduíno e apresentava um custo menor para o projeto, porem durante o desenvolvimento devido a incompatibilidades com as bibliotecas existentes para o controle de dispositivos de infravermelho, optamos pela substituição do modulo ENC28J60 para o shield W5100.

Figura 03 - Módulo Ethernet ENC28J60

O shield W5100 baseia-se no chip WIZnet ethernet W5100 fornecendo acesso à rede nos protocolos TCP ou UDP e compatível com as bibliotecas padrões do Arduíno Ethernet Library e SD Library. Pode ser utilizado tanto no Arduino Uno ou Mega, e possui um suporte para cartão micro-SD que pode ser usado para armazenar arquivos.

As portas utilizadas para comunicação do módulo com o Arduíno são os 10,11,12 e 13, além de utilizarem dois pinos para alimentação (VCC e GND)

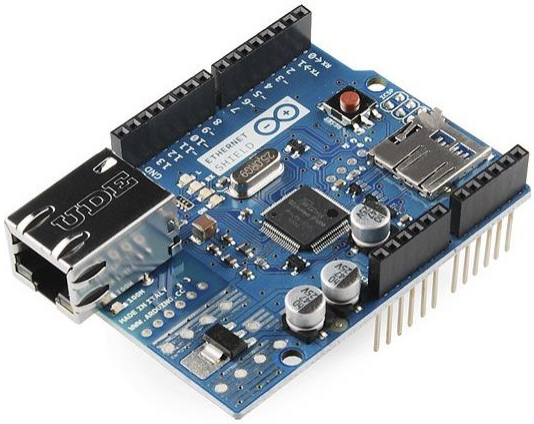

Figura 04 - Shield Ethernet W5100

2.3 - Relê Shield

Rele Shield é uma placa que permite o acionamento de dispositivos em outras tensões de operação. Funciona como um interruptor eletrônico, onde ao aplicar tensão no terminal de entrada é acionada uma bobina que cria um campo magnético capaz de abrir ou fechar os contatos de maneira que possamos controlar as correntes que circulam por circuitos externos. Com isso ele se utiliza de baixa corrente para acionar seu comando e protege o controlador das correntes mais altas que circulam pelo segundo circuito.

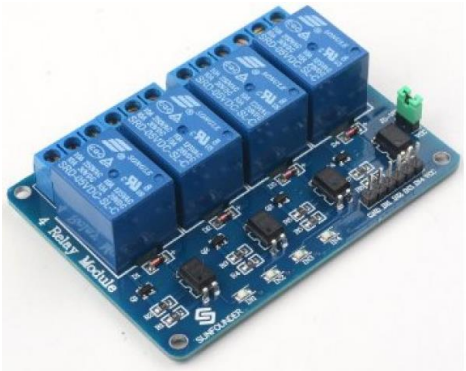

Figura 05 - Rele Shield contendo quatro bobinas, onde é possível o controle de quatro circuitos externos

2.4 - Emissores e Receptores Infra Vermelhos (IR)

O LED (Diodo Emissor de Luz) IR é um diodo semicondutor que ao ser energizado, com tensões de 1,6 a 3,3 V, emite uma luz , a cor da luz emitida vai depender do cristal e da impureza de dopagem com que o componente é fabricado. No caso do infravermelho utiliza o arsenieto de gálio para emitir radiações infravermelhas.

Nos LED's as características de polarização são semelhantes á de um diodo semicondutor, onde a maioria dos fabricantes adota um código de identificação dos terminais: o terminal do catodo é aquele junto a um chanfro na lateral da base do invólucro ou sendo o terminal mais curto.

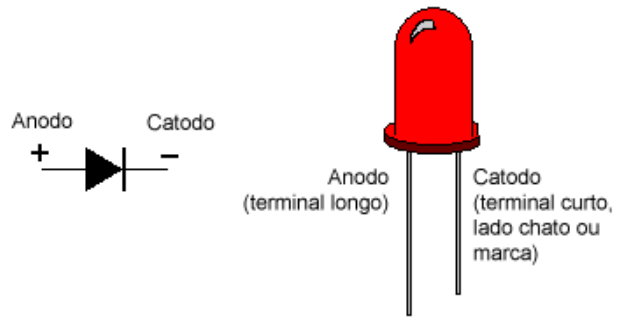

Figura 06 - Esquema que demonstra a identificação dos terminais de um led

Um receptor de infravermelho é um dispositivo capaz de fazer a leitura de mensagens IR, geralmente possui 3 terminais, VCC, GND responsáveis pela alimentação do receptor e OUT que deverá ser ligado a qualquer porta de entrada digital do microcontrolador. É através do monitoramento do terminal OUT que o Arduíno consegue fazer a leitura da mensagem.

Figura 07 - Imagem de um Receptor IR

2.5 - Sensores de temperatura, fumaça e presença

Sensores são dispositivos capazes de ler variáveis físicas e transformá-las em uma informação mensurável que pode ser digital ou analógica. Como os sensores utilizados no projeto fazem uso de portas analógicas e digitais não precisamos de nenhuma biblioteca para a sua utilização.

O sensor de temperatura (LM35) fornece uma leitura dentro da faixa de -55ºC até 150ºC com variações de 1/4ºC até 3/4ºC com exatidão. Ao ser alimentado por uma tensão de 4-20Vdc e GND apresenta em sua saída uma tensão linear referente a temperatura de 10mV para cada grau Celsius. Pode apresentar diversos tipos de encapsulamento sendo o mais comum o TO-92, ficando bastante parecido com um transistor.

Apresenta grande vantagem em relação ao uso de outros sensores, pois seus valores já estão calibrados para Celsius, simplificando a interface de leitura e evitando operações com variáveis para troca da escala de temperatura

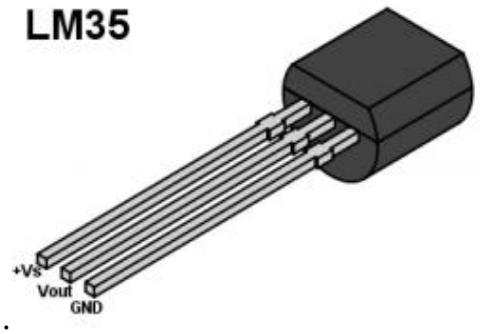

Figura 08 - Imagem de um Sensor de Temperatura do tipo LM35

O sensor de gás inflamável e fumaça (MQ-2) é utilizada principalmente nos projetos de domótica pois consegue detectar a presença de gás de diferentes tipos (GLP, butano, propano, metano, hidrogênio, etc) ou até mesmo fumaça.

Possui os terminais de alimentação (VCC, GND) e pode ser utilizado de duas maneiras, analógica ou digital. Se optarmos por utilizar a saída digital, caso a concentração de gases/fumaça fique acima do ajustado, a saída digital do sensor fica em estado alto e informa o seu acionamento para o microcontrolador. Através da saída analógica, o microcontrolador é capaz de saber o nível de concentração de gases detectados pelo sensor. Quanto maior a concentração, maior o valor passado pela porta analógica.

Figura 09 - Imagem de um Sensor de Gás Inflamável / Fumaça MQ-2

O sensor de presença geralmente utilizam de um sensor de infravermelho passivo como detector de movimento, que quando ocorre uma variação na detecção do sinal de infravermelho entre as faixas a saída é acionado num determinado período de tempo. A lente que cobre o sensor tem como função aumentar o campo de detecção, concentrando a luz em um único ponto.

Figura 10 - Imagem de um Sensor de Presença

2.6 - Roteadores

Aparelho responsável pelo encaminhamento de mensagens entre dispositivos de rede, são capazes de definir a melhor rota que o pacote deverá percorrer na rede até a chegada no nó de destino

Os pacotes são encaminhados de acordo com a tabela de roteamento, que é preenchida e atualizada executando processos e protocolos de atualização de rotas. No método de roteamento dinâmico o conhecimento de rota é atualizado automaticamente sempre que novas informações forem recebidas através da rede.

Figura 11 - Imagem de um Roteador

2.7 - Sistema Operacional Android

Android é um sistema operacional voltado para dispositivos móveis baseado no núcleo do linux e desenvolvido pela Google. Atualmente é o sistema operacional móvel mais utilizado no mundo e esta presente em milhares de aparelhos, de várias marcas.

O sistema surgiu em 2003 e foi desenvolvido por empresários que fundaram a Android Inc. na cidade de Palo Alto (Califórnia). A ideia inicial do grupo era desenvolver sistemas para câmeras digitais, quando descobriu que o mercado pra este dispositivo não era grande o suficiente, voltaram sua atenção para o mercado mobile.
Em 2007 o Google articulou a criação da Open Handset Alliance, um consórcio entre grande empresas com o objetivo de criar uma plataforma de código aberto para smartphones.

Seu funcionamento é similar aos outros sistemas operacionais e tem como principal função gerenciar todos os processos dos aplicativos e hardware de um dispositivo para que tenha um correto funcionamento, fornecendo ao usuário uma interface visual capaz de prover interação com o sistema eletrônico.

Em meio a crise que atinge o país, o sistema Android vem aumentando sua participação no mercado brasileiro, ultrapassando facilmente os números do IOS da Apple e também os celulares com Windows Phone, e está consolidado como o sistema operacional mais utilizado em smartphones no mundo. Segundo pesquisa da Kantar o numero de vendas de smartphones com sistema operacional Android no final de 2014 atingia o total de 89% do total, enquanto seus concorrentes diretos (Windows Phone e IOS) juntos somavam 9,5% das vendas. A diferença ao final de 2015 aumentou ainda mais com o Android totalizando 91,8% e os sistemas concorrentes 7,9%.

O grande trunfo do Android encontra-se na maior oferta de aparelhos, com grande variedade de preço, fazendo com que o sistema da Google tenha uma posição vantajosa em relação a seus concorrentes.

3 Detalhamento do Projeto

O projeto é compreendido em 3 divisões básicas: software, comunicação e componentes físicos. O software está dividido em dois ambientes de desenvolvimento, sendo ele o software que será executado no Arduíno, e o aplicativo a ser desenvolvido para sistema operacional Android. A comunicação ocorrerá através de um roteador para gerenciar a troca de pacotes entre os dispositivos . A parte de componentes físicos é composta por uma série de sensores e atuadores que irão operar de acordo com os comandos enviados pelo aplicativo.

Esperamos que ao final do projeto o software consiga enviar e receber dados corretamente entre os nós terminais do sistema, que são Arduíno e o software Android.

3.1 - Ambiente de Desenvolvimento Android

Durante o desenvolvimento da software android foi utilizado a IDE Eclipse com o kit de desenvolvimento para ambiente Android, que foi obtido com a instalação das ferramentas Java Development Kit (JDK) e Android SDK , e com a instalação do plugin Android Development Tools (ADT).

Existem diversas plataformas para desenvolvimento do software Android disponíveis no mercado, a escolha pela IDE Eclipse ocorreu devido a disciplina lecionada na faculdade fazer uso da ferramenta e por isso estarmos familiarizados com a plataforma de desenvolvimento.

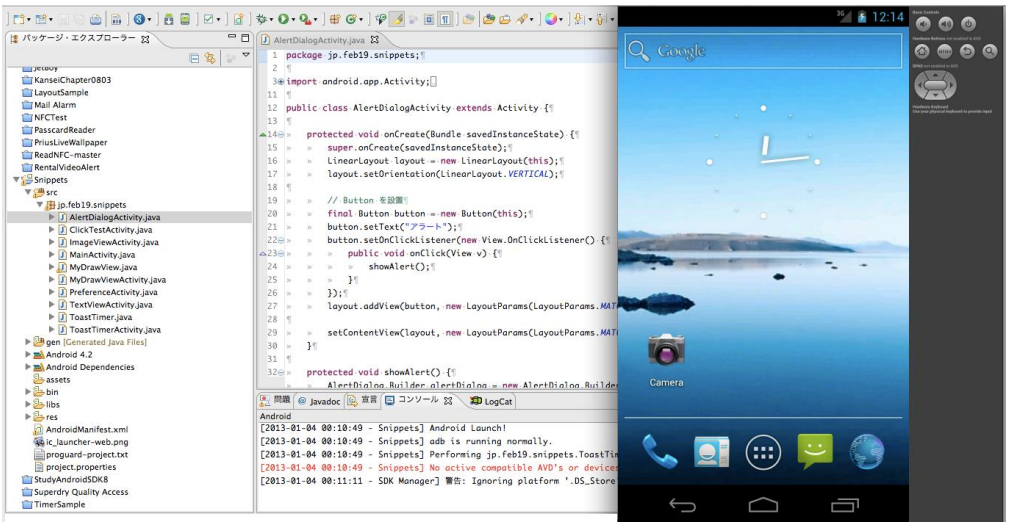

Figura 12 - Captura de tela do Eclipse com ADT já instalado

O primeiro passo no desenvolvimento da aplicação Android, foi a elaboração do layout de acordo com as utilidades do projeto e a navegação entre as telas do aplicativo. A interface com o usuário foi dividida em 7 telas, sendo a tela principal responsável pela apresentação e navegação para as funcionalidades disponíveis. O padrão adotado foi escolhido e elaborado da forma mais simples e intuitiva possível para que qualquer usuário do sistema seja capaz de utilizar todas as suas funcionalidades.

Figura 13 - Layout da Pagina Inicial

Nas telas de controle das funções disponíveis foram utilizados botões do Button que são os responsáveis por armar e desarmar os sistemas que serão controlados pelo Arduíno.

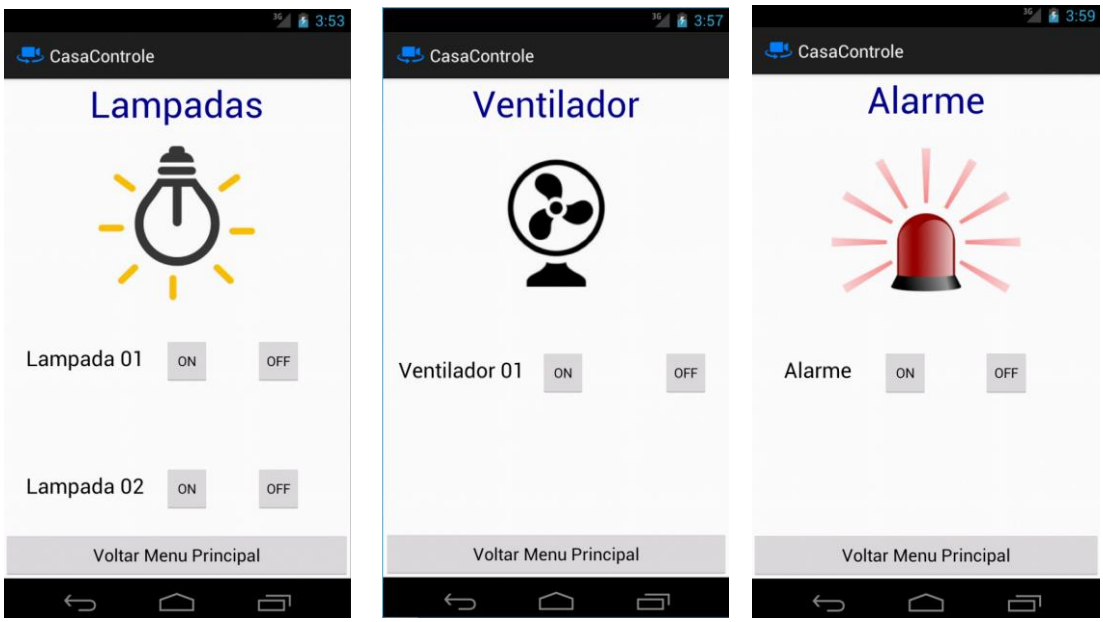

Figura 14 - Layout das paginas de controle

Na tela que controla o sistema de televisão buscou-se um layout similar a um controle de televisão com suas funcionalidades mais usuais, como troca de canal, controle de volume, mudança de source.

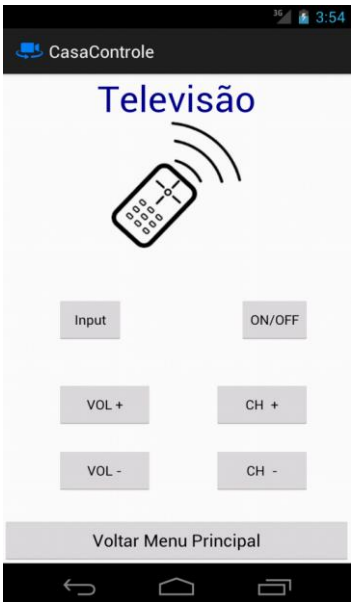

Figura 15 - Layout responsável por controlar o sistema de TV

A tela de configurações apresenta parâmetros de configuração do aplicativo, qual o IP do Servidor, porta utilizada e status da conexão, já na tela de sensores deverá apresentar a temperatura medida pelo sensor e status dos sensores de presença e fumaça.

Figura 16 - Layout Sensores e Configuração

3.2 - Software Arduíno

O Ambiente de Desenvolvimento Integrado Arduíno (IDE Arduino) é um editor de código fonte capaz de compilar e carregar programas, que são desenvolvidos em linguagem de programação baseada em Wiring. Foi desenvolvido para introduzir a programação através de recursos de realce de sintaxe, parênteses correspondentes e identação automática, sendo capaz de compilar e carregar os softwares nele desenvolvidos para o microcontrolador em um único clique.

Primeiro devemos conectar a placa a uma porta USB do computador e então desenvolvemos os comandos da placa e enviamos para o microcontrolador pela própria IDE. Após enviado o código e reiniciado o dispositivo não precisamos mais do computador , pois o Arduíno passará a executar os comandos quando for ligado a uma fonte de energia.

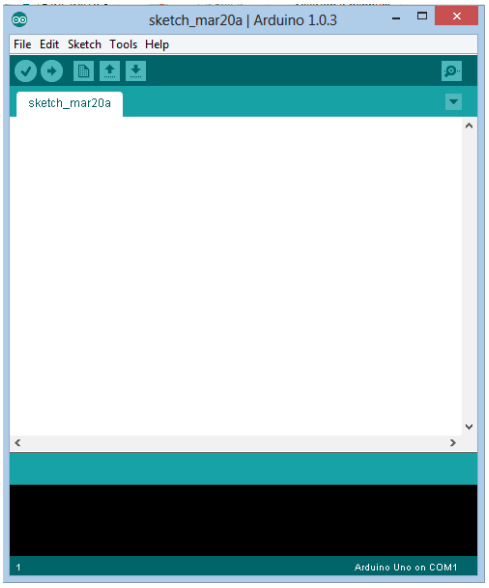

Figura 17 - Captura de tela da IDE do Arduíno

Existe duas funções principais a setup() , a qual é executada na inicialização do programa e utilizada para carregar as configurações inciais do microcontrolador e a função loop() que é responsável por repetir a sequencia de comandos que estão dentro da função de forma infinita até que a energia do microcontrolador seja desligada.

O código do microcontrolador apresenta a função loop que no seu incio verifica os status dos sistemas de alarme e combate a incêndio, realizando tarefas de acordo com as leituras fornecidas pelos sensores e conta também com o servidor HTTP que fica monitorando as mensagens enviadas pelo dispositivo Android.

3.3 - Fluxo de Dados do Projeto

O sistema fará uso de dois meios físicos para a transferência de dados, a comunicação entre Arduíno e roteador sera cabeada, e entre roteador e smartphone utilizara o wi-fi permitindo ao usuário controle de qualquer ponto dentro do limite de alcance do roteador. A comunicação entre o aplicativo e os dispositivos ligado ao Arduíno será feita conforme fluxo abaixo:

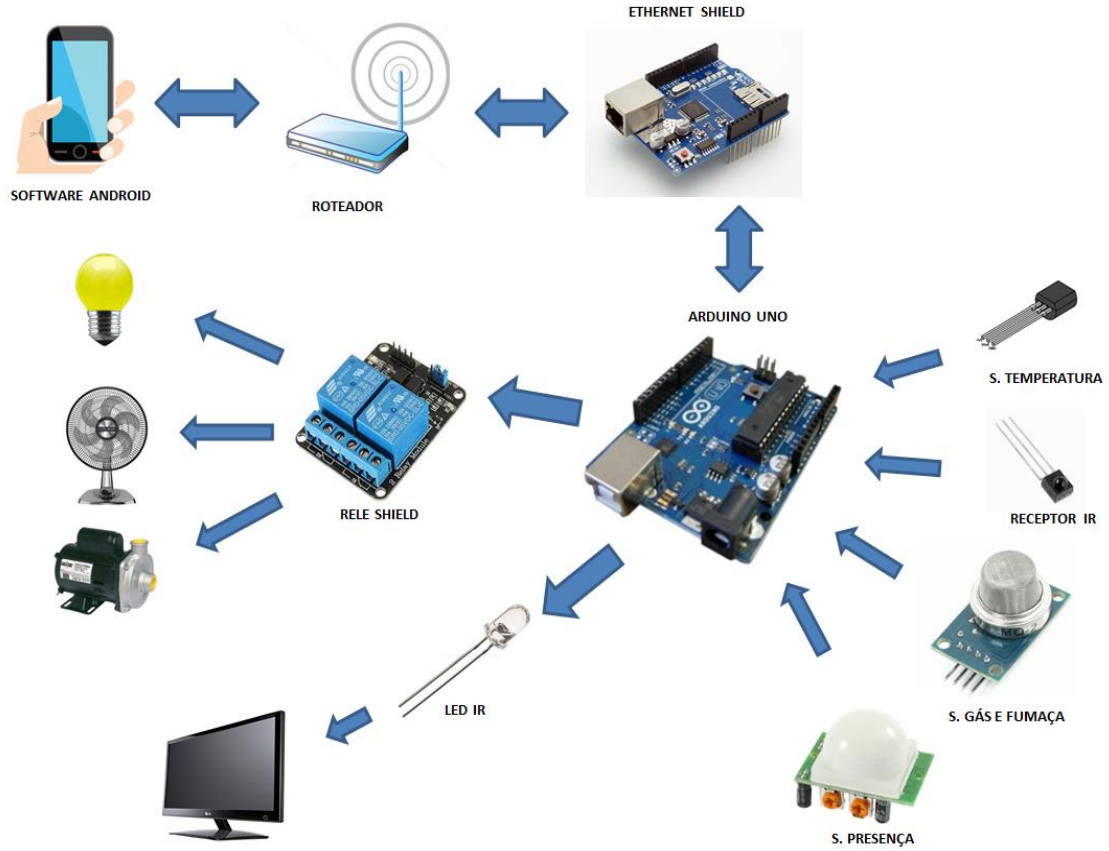

Figura 18 - Fluxo de dados do Sistema

Para configuração de troca de mensagens utilizaremos o microcontrolador Arduino como um servidor que durante a execução da função loop(), aguarda os comandos enviados pelo software Android, que será o nosso cliente.

3.4 - Montagem dos Componentes Físicos

Para o projeto utilizaremos uma maquete representando uma residência com os principais itens a serem controlados pelo aplicativo: lampadas, televisor, ventiladores, alarme de presença e alarme de incêndio. A proposta do projeto é apresentar o controle desses dispositivos de maneira remota e de acordo com a necessidade do usuário.

Figura 19 - Maquete com as lampadas, tomada e o módulo de controle

Como utilizamos um shield ethernet foram necessárias 5 linhas digitais para ligação do módulo, 2 linhas digitais para o sistema de controle de dispositivos IR, 2 linhas digitais para o sistema de iluminação, 1 linha digital para o sistema de ventilação/refrigeração, 2 linhas analógicas para o sistema de incêndio e 1 linhas digitais para o sistema de alarme.

As conexões dos componentes foram realizadas conforme esquema abaixo.

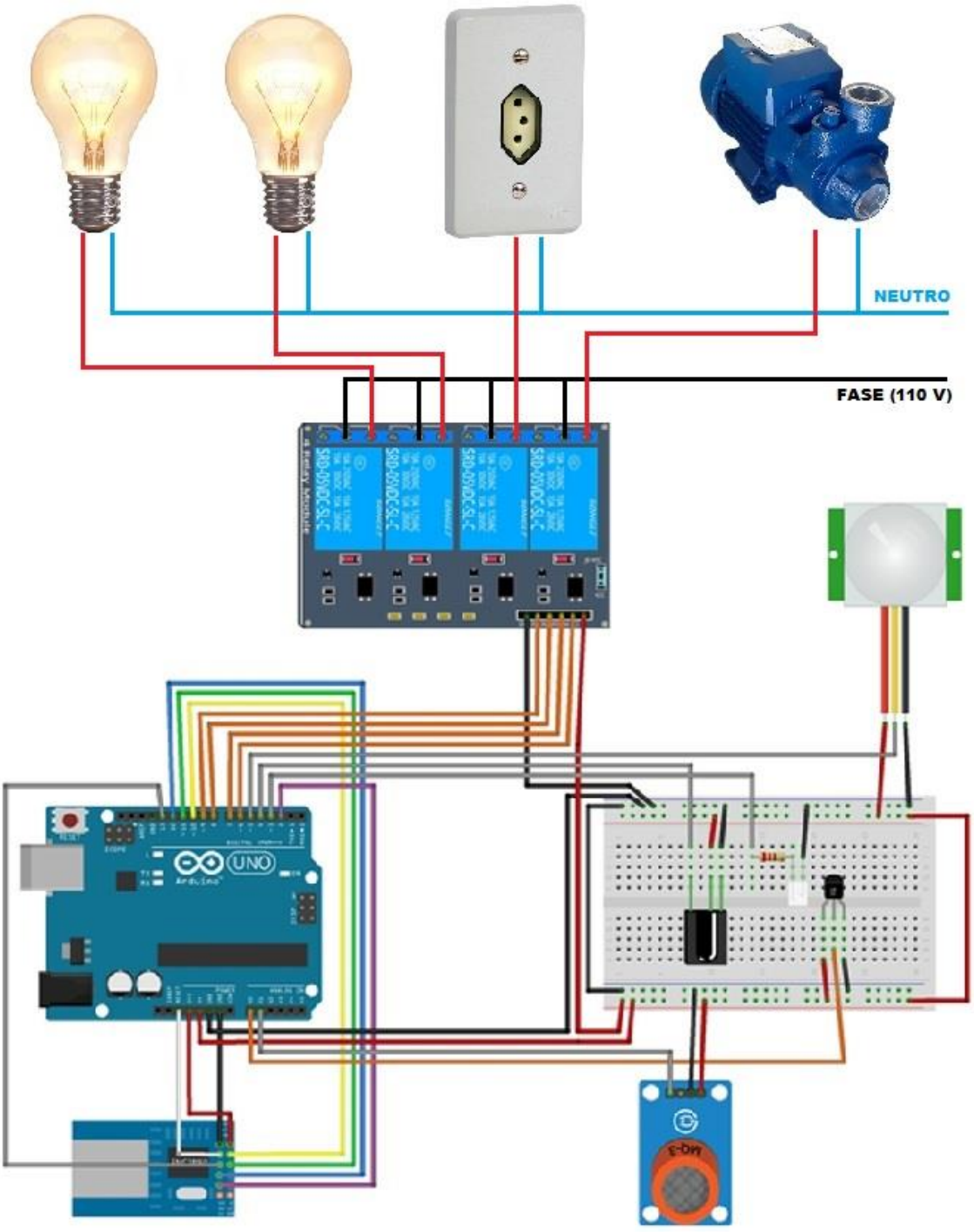

Figura 20 - Esquema de montagem dos componentes

4 Desenvolvimento

No capitulo apresentamos trechos de códigos utilizados tornando possível a funcionalidade definida no escopo do projeto, apresentando as dificuldades encontradas durante a execução até a obtenção de um resultado satisfatório.

4.1 - Comunicação entre dispositivos

Para utilização do shield Ethernet W5100 foi utilizada a biblioteca padrão do Arduíno (Ethernet) e a escolha deve-se ao fato da biblioteca possui funções que tornam a sua utilização mais simples. A comunicação entre o software e hardware foi feita através do protocolo HTTP, onde a mensagem é disparada pelo cliente presente no smartphone e recebida no servidor que foi configurado no Arduíno, para isso definimos o endereço de IP do microcontrolador como fixo e configuramos o modulo ethernet. Alem das configurações feitas no microcontrolador, foi configurado no roteador o redirecionamento de portas que fica responsável por encaminhar as mensagens que chegam através da porta 8888 para o IP configurado no Arduíno.

```
Serial.begin(9600);
EthernetServer servidor = EthernetServer(8888);
uint8_t EnderecoMAC[6] = {0x00,0x01,0x02,0x03,0x04,0x05};
IPAddress IP(192,168,1,10);
Ethernet.begin(EnderecoMAC,IP);
servidor.begin();
```

Figura 21 - Trecho de código onde são definidas as configurações do módulo ethernet

Na parte do cliente, foi desenvolvida uma classe que aloca as funções de

comunicação do software Android, que ao pressionar um botão do aplicativo que comanda as funções implementadas no Arduíno, a classe inicia uma tarefa “Thread” que cria o cliente, prepara uma solicitação e envia para o microcontrolador através de uma URI (Identificador Uniforme de Recurso) passada na função. Cada função do sistema corresponde a uma URI que será enviada pelo Android. O caminho passado na URI segue um formato específico IP Arduíno + Porta + Comando.

```
public Cliente(String Mensagem){
    super();
    URL = Mensagem;
    Thread msg = new Thread(this);
    msg.start();
}


@Override
public void run() {
    try{
        HttpClient cliente = new DefaultHttpClient();
        HttpGet solicitar = new HttpGet();
        solicitar.setURI(new URI(URL));
```

Figura 22 - Trecho de código responsável por definir a URI

[parei aqui]

O Arduíno, configurado como um servidor, recebe a mensagem e verifica qual funcionalidade foi solicitada, executa e retorna uma resposta para o cliente, esse retorno foi utilizado para enviar status dos sensores para o Android, pois grande parte do tratamento de dados ficou no aplicativo Android, devido as limitações de espaço do Arduíno.

```
EthernetClient cliente = servidor.available();
 if (cliente) {
   boolean linhaVazia = true;
   String linhaComando="";
   while (cliente.connected()) {
     if (cliente.available()) {
       char c = cliente.read();
         if(linhaComando.length()<20){
           linhaComando.concat(c);
         }
```

Figura 23 - Trecho de código responsável pela leitura do comando enviado pelo Android

Abaixo o trecho de código responsável por retornar os valores do sensores de temperatura, presença e fumaça. O código responsável pela alerta dos sistemas de alarme e combate a incêndio são processados dentro do Arduíno, que envia valores para o aplicativo caso algum dos sistemas tenha disparado.

```
if (c == '\n' && linhaVazia) {
cliente.println("HTTP/1.1 200 OK");
cliente.println("Content-Type: text/html");
cliente.println();
cliente.println(temperatura);
cliente.println(sPren);
cliente.println(sFum);
cliente.println(codigoAlarme);
cliente.println(codigoIncendio);
break;
```

Figura 24 - Retornando os dados dos sensores para o Android

O aplicativo Android recebe os dados do Arduíno através do HTTPResponse e armazena os valores em um vetor, a cada chamada da pagina principal o aplicativo executada a função decodificaLeitura() que atualiza as variáveis de acordo com os valores armazenados no vetor.

```
public void decodificaLeitura(String []leitura){
    temperatura=leitura[0];
    codPresenca=leitura[1];
    codFumaca=leitura[2];
    codAlarme=leitura[3];
    codIncendio=leitura[4];
}
```

Figura 25 - Armazenando os valores recebidos nas suas variáveis.

4.2 - Sistema de Controle de Lampadas e Ventiladores

Definimos os pinos digitais 9, 8 e 7 como responsáveis pelo acionamento dos

sistemas, que funcionam da seguinte forma. Quando o Arduíno recebe o comando do software instalado no dispositivo móvel, ele altera o status da porta de acordo com a mensagem. podendo trocar entre os estados “HIGH”, onde a porta encontra-se com sinal, ou “LOW”, que deixa a porta sem sinal. Para a utilização de circuitos que operam em voltagem acima da suportada pelo Arduíno, devemos utilizar reles que são acionados de acordo com o status da porta.

A seguir o trecho do código responsável por alterar os status da porta de acordo com a leitura da mensagem.

```
if(linhaComando.indexOf("LAMP01ON") != -1 ){
  Serial.println("Lampada 01 Acessa");
  digitalWrite(lamp1, HIGH);
}
if(linhaComando.indexOf("LAMP01OFF") != -1 ){
  Serial.println("Lampada 01 Apagada");
  digitalWrite(lamp1, LOW);
}
if(linhaComando.indexOf("LAMP02ON") != -1 ){
  Serial.println("Lampada 02 Acessa");
  digitalWrite(lamp2, HIGH);
}
if(linhaComando.indexOf("LAMP02OFF") != -1 ){
  Serial.println("Lampada 02 Apagada");
  digitalWrite(lamp2, LOW);
}
```

Figura 26 - Interpretando os comandos enviados pelo Android e alterando os satus das portas

O sistema de controle de ventilação funciona de forma idêntica ao sistema de controle de lampadas. Onde o comando enviado pelo Android altera o estado da porta digital acionando o relê ligando ou desligando o ventilador.

4.3 - Sistema de Controle de Televisores

Para o sistema de controle de dispositivos IR a biblioteca utilizada foi a

IRremote.A biblioteca possui dois comandos: IRsend e IRrecv, onde o IRrecv recebe e decodifica uma mensagem IR, e o IRSend faz uso de um led IR para enviar a mensagem.

4.3.1 - Leitura dos Códigos Infra Vermelho

Devido a limitação de armazenamento do código Arduíno e tendo em vista o escopo do projeto, realizamos a leitura dos comandos de IR com código secundário que é diferente do utilizado pelo Arduíno para o controle dos dispositivos.

O código utilizado permita receber um código enviado por um controle remoto comercial e que apresente na tela do PC, através do monitor serial, o protocolo e o código utilizados na mensagem recebida.

Quando o usuário ligar o Arduíno o hardware encontra-se em modo de leitura e apto a receber um novo código IR, quando a leitura for realizada pelo receptor IR, o programa verifica o tipo de protocolo utilizado e código recebido através da função decodifica(), caso seja um dos reconhecidos pela biblioteca o sera impresso na tela as informações da mensagem, informando o usuário que a leitura foi realizada com sucesso.

```
void decodifica(decode_results *Leitura){
    Serial.print("Protocolo Utilizado: ");
     if (Leitura->decode_type == UNKNOWN) {
          Serial.print("Desconhecido");
     }
     else {
          if (Leitura->decode_type == NEC) {
               Serial.print("NEC");
          }
          else if (Leitura->decode_type == SONY) {
               Serial.print("SONY");
          }
          else if (Leitura->decode_type == RC5) {
               Serial.print("RC5");
          }
          else if (Leitura->decode_type == RC6) {
               Serial.print("RC6");
          }
     }
     Serial.print(" - Codigo Enviado: ");
     Serial.println(Leitura->value,HEX);
```

Figura 27 - Código responsável por imprimir na serial o protocolo utilizado e valor da mensagem IR

Com a leitura realizada definimos os valores a serem inseridos no código principal e inserimos ela de forma manual no campos correspondentes.

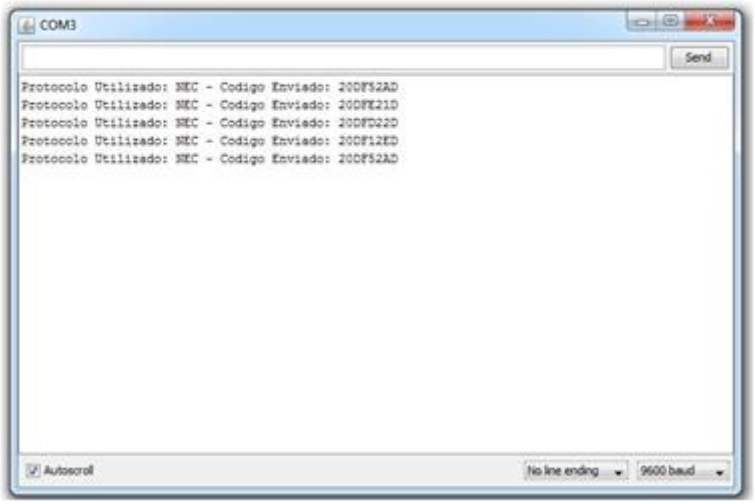

Figura 28 -Monitoramento da Serial demonstrando a impressão em tela dos códigos recebidos

4.3.2 - Envio dos Códigos Infra Vermelho

Com as mensagens e protocolos inseridos previamente no código do Arduíno, que foram obtidos através da leitura de IR feita com o código secundário, o programa aguarda a função ser chamada pelo aplicativo do celular, caso positivo ele enviará através da função enviarMensagem() os parâmetros pré-estabelecidos.

```
void enviarMensagem(struct mensagem_IR mensagem){
  if(mensagem.tipo==NEC){
    emissorIR.sendNEC(mensagem.valor,mensagem.bits);
  }
  else{
    if(mensagem.tipo==SONY){
      emissorIR.sendSony(mensagem.valor,mensagem.bits);
    }
    else if(mensagem.tipo==RC5){
      emissorIR.sendRC5(mensagem.valor,mensagem.bits);
    }
    else if(mensagem.tipo==RC6){
      emissorIR.sendRC6(mensagem.valor,mensagem.bits);
    }
  }
}
```

Figura 29 -Código responsável por enviar as mensagens de acordo com o protocolo

As mensagens foram definidas como blocos que armazenam diversas informações (structs) e seus valores iniciais foram inseridos dentro da função setup() conforme mostrado abaixo.

```
msgbotao01.valor=1086308415;
msgbotao01.bits=32;
msgbotao01.tipo=NEC;

msgbotao02.valor=1086316575;
msgbotao02.bits=32;
msgbotao02.tipo=NEC;

msgbotao03.valor=1086283935;
msgbotao03.bits=32;
msgbotao03.tipo=NEC;

msgbotao04.valor=1086320655;
msgbotao04.bits=32;
msgbotao04.tipo=NEC;

msgbotao05.valor=1086310455;
msgbotao05.bits=32;
msgbotao05.tipo=NEC;
```

Figura 30 -Armazenando as informações nas structs

4.4 - Leitura de Sensores (Temperatura, Presença e Fumaça)

Os valores enviados pelos sensores são atualizados a cada passagem do loop principal do Arduíno, sendo lidos em momentos e funções diferentes.

Sensor Temperatura

Para realizar a leitura do sensor de temperatura utilizamos uma porta analógica e a é uma fórmula para converter os valores recebidos pelo sensor pra graus Celsius (ºC).

O código receberá o valor da leitura realizada na porta analógica, que varia de 0 a 1023, onde 0 corresponde a 0Volts e 1023 corresponde a 5Volts. Como sabemos, 1ºC é igual a 10mV. Sendo assim, temos:

Tensão = (Valor lido na porta analógica)*(5/1023)

Temperatura = Tensão /10mV

Transformando a fórmula em linguagem de programação, teremos o código a seguir.

```
temperatura = float(analogRead(sTemperatura))*5/(1023)/0.01 ;
```

Figura 31 - Trecho de código responsável pela conversão do valor recebido na serial em graus Celsius

Sensor de Presença

Para utilização do sensor de presença foi especificada uma porta digital onde é monitorado o seu status, caso o sensor detecte presença o status da porta é alterado para HIGH, e quando não detecta permanece em LOW.

```
if(digitalRead(sPresenca)==HIGH){
  sPren=1;
}
if(digitalRead(sPresenca)==LOW){
  sPren=0;
}
```

Figura 32 -Trecho de código onde alteraramos a variável que armazena o status do sensor de presença

Sensor de Fumaça

O sensor de fumaça está ligado a uma porta analógica, assim o valor lido na porta (sFumaca) é comparável a uma variável previamente calibrada (nivelFumaca) alterando assim o status para detectado ou normal.

```
if(analogRead(sFumaca)>nivelFumaca){
  sFum=1;
}

if(analogRead(sFumaca)<nivelFumaca){
  sFum=0;
}
```

Figura 33 -Trecho de código que altera a variavel do sensor de fumaça

O valor da variável nivelFumaca foi definido através de testes realizados anteriormente, onde foram feitas leituras da porta analógica em situação normal, ou seja sem a presença de fumaça e calculado a media dos valores obtidos.

Os valores obtidos em uma situação “normal” foram próximos a 65 e alterou-se rapidamente para valores acima de 100 na presença de fumaça.

4.5 - Sistema de Alarme

Para o sistema de alarme trabalhamos com três variáveis, uma responsável por armazenar a leitura do sensor de presença (sPren) e outra para indicar caso o sistema

tenha sido ativado pelo aplicativo Android (alarme).

Se o sistema encontra-se ativo e foi detectada a presença através do sensor usamos a terceira variável (codigoAlarme) responsável pela comunicação com o aplicativo.

```
if(alarme==true && sPren==1){
    codigoAlarme=1;
    Serial.println("Alarme Ativo - Presenca Detectada");
}

if(alarme==false){
  codigoAlarme=0;
}
```

Figura 34 -Trecho de responsável por alterar a variável enviada pelo HttpResponse

A variável codigoAlarme é uma das variáveis enviadas via HttpResponse e responsável, de acordo com seu valor, por iniciar a função que exibe o alerta no dispositivo móvel.

```
public void alertaPresenca() {
    AlertDialog alertaPresenca;
    AlertDialog.Builder alerta = new AlertDialog.Builder(this);
    alerta.setTitle("Alerta Presenca");
    alerta.setMessage("Foi detectado presenca no local, ativar alarme sonoro");
    alerta.setPositiveButton("Ativar",new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Cliente cliente =new Cliente(URLCMD+"SIRENEON");
        }
    });
    alerta.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {

        }
    });
    alertaPresenca = alerta.create();
    alertaPresenca.show();
}
```

Figura 35 - Função que exibe na tela o alerta de presença

Quando a mensagem de alerta de presença é exibida na tela do dispositivo, o

usuário tem a opção de acionar um alarme sonoro no local que encontra-se instalado o sistema.

4.6 - Sistema de Combate a Incêndio

A lógica do sistema de combate a incêndio funciona da seguinte forma, o sensor de presença ao detectar fumaça, armazena a ultima leitura de temperatura na variável tempInicial. Para garantir que o sistema só dispare o alerta em situações reais de incêndio, monitoramos a diferença da temperatura atual da temperatura armazenada na variável tempInicial, durante alguns loops do Arduino. Se dentro do período determinado a diferença supere 10ºC altera-se o valor da variável passada para o aplicativo, e o mesmo dispara a mensagem de alerta de incêndio para o usuário.

```
if(sFum==1){
  if(indexIncendio==0){
    tempInicial=temperatura;
  }
  if(indexIncendio<500){
    indexIncendio=indexIncendio+1;
    float dif = temperatura-tempInicial;
      if(dif>10){
        codigoIncendio=1;
         Serial.println("Incendio Detectado");
        indexIncendio=0;
      }
      if(dif<10){
        codigoIncendio=0;
      }
  }
  if(indexIncendio==500){
    tempInicial=0;
    indexIncendio=0;
    codigoIncendio=0;
  }
}
```

Figura 36 - Código Arduíno responsável por ativar o alerta do sistema de incêndio

O usuário receberá na tela do aplicativo o alerta de incêndio e será questionado

do acionamento do sistema de combate, que consiste no acionamento de uma bomba d´água responsável por irrigar o ambiente.

```
public void alertaIncendio() {
    AlertDialog alertaIncendio;
    AlertDialog.Builder alerta = new AlertDialog.Builder(this);
    alerta.setTitle("Alerta Incendio");
    alerta.setMessage("Deseja ativar o sistema de combate a Incendio");
    alerta.setPositiveButton("Ativar",new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Cliente cliente =new Cliente(URLCMD+"INCENDIO");
        }
    });
    alerta.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
    });
    alertaIncendio = alerta.create();
    alertaIncendio.show();
}
```

Figura 37 - Código responsável pelo alerta e acionamento do sistema de combate a incêndio

5 Conclusão

5.1 - Conclusões

Algumas das funcionalidades definidas no pré-projeto, tiveram que ser expurgadas ou sofreram redução ou alteração no escopo devido a limitação de memória de armazenamento do Arduíno.

Durante o projeto foi utilizado o módulo ethernet ENC28J60, porém o mesmo apresentou incompatibilidade entre bibliotecas, uma vez que para utilização do módulo seria necessário a biblioteca UIPEthernet, onde a mesma entrava em conflito com a biblioteca IRRemote. A solução adotada foi o uso do Shield Ethernet que utiliza as bibliotecas padrões Ethernet e SPI.

Visto que o projeto teve seu escopo inicial o uso pessoal do sistema, foram utilizados códigos diferentes para leitura e envio de mensagens IR, caso o projeto seja transformado em produto deverão ser feitas alterações que permitam ao usuário final fazer a leitura e envio com um mesmo código.

Das funcionalidades implementadas, todas funcionaram de forma satisfatória com a resposta do microcontrolador em tempo aceitável.

5.2 - Sugestões para Trabalhos Futuros

Importar a lógica do funcionamento do sistema de combate a incêndio para o Arduíno, fazendo com que o seu acionamento fique independente da confirmação no aplicativo caso extrapole um tempo previamente definido após o envio do alerta.

Como tentativa de tornar o protótipo mais móvel, sugere-se uma alteração na forma de comunicação entre smartphone e roteador, utilizando a internet para transmitir as mensagens e com isso aumentando o raio de utilização do aplicativo, que passará a funcionar de qualquer lugar bastando possuir o acesso a internet disponível no celular e no roteador conectado ao Hardware.

Como proposta de melhoria para o sistema de controle de dispositivos IR, estudar a possibilidade de conseguir fazer a leitura e enviar os códigos onde o protocolo é do tipo UNKNOWN.

Além das automações realizadas, poderia implementar novos itens ao aplicativo, como controle de acesso de pessoas, acesso ao sistemas de câmeras (CFTV), uma integração entre o sistema de alarme de incêndio e o corpo de bombeiros.

Referências Bibliográficas

MURATORI, José Roberto; FORTI, José Cândido; OMAI, Paulo (2004). Associação Brasileira de Automação Residencial : Home Cabling Training Manual.

BOLZANI, Caio Augustus M. (2004) Residências Inteligentes. São Paulo: Livraria da Física.

MEYER, Gordon (2004). Smarth Home Hacks: Tips & Tools for Automating Your House. Sebastopol: O'Reilly Média.

Site Oficial do Arduíno. Disponível em: <http://www.arduino.cc/>. Acesso em: 13/05/2014.

A Multi-Protocol Infrared Remote Library for the Arduino. Disponível em: <http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html>. Acesso em: 14/05/2014.

Diodo emissor de luz. Disponível em: <http://pt.wikipedia.org/wiki/Diodo_emissor_de_luz>. Acesso em: 13/05/2014.

Controle remoto. Disponível em: <http://www.diy.com.br/projeto/controle-remoto-da-vovo-com-arduino>. Acesso em: 13/05/2014.

Relay Shield for Arduino. Disponível em: <http://www.dfrobot.com/wiki/index.php/Relay_Shield_for_Arduino_V2.1_(SKU:DFR0144)>. Acesso em: 13/06/2016.

Tudo Sobre Relés. Disponível em: <http://www.newtoncbraga.com.br/index.php/como-funciona/597-como-funcionam-os-reles?showall=1&limitstart=>. Acessado em: 13/06/2016.

Arduino Ethernet Shield. Disponível em: <https://www.arduino.cc/en/Main/arduinoEthernetShield>. Acessado em: 09/06/2016.
Sensor de presença com módulo PIR DYP-ME003. Disponível em: <http://www.arduinoecia.com.br/2014/06/sensor-presenca-modulo-pir-dyp-me003.html> . Acessado em: 01/06/2016.

Smartphone OS sales market share. Disponível em: <http://www.kantarworldpanel.com/global/smartphone-os-market-share/>. Acessado em:26/07/2016

http://blog.vidadesilicio.com.br/Arduíno/basico/lm35-medindo-temperatura-com-Arduíno/ Acessado em:26/07/2016

Alarme sensor de gás com o módulo MQ-2.Disponível em: <http://www.Arduinoecia.com.br/2015/01/alarme-sensor-de-gas-modulo-mq-2.html> Acessado em: 24/07/2016

Dê upgrade à sua casa com ideias de automação tecnológica. Disponível em : <http://tecnologia.terra.com.br/inovacoes-tecnologicas/de-upgrade-a-sua-casa-com-ideias-de-automacao-tecnologica,045ed8c6b2838410VgnVCM3000009af154d0RCRD.html> Acessado em: 24/07/2016

O que é um roteador?Disponivel em:< http://br.ccm.net/faq/9417-o-que-e-um-roteador> Acessado em: 24/07/2016

Roteador. Disponivel em:<https://pt.wikipedia.org/wiki/Roteador> Acessado em: 24/06/2016

http://www.embarcados.com.br/Arduíno-primeiros-passos/ Acessado em: 24/06/2016

Colocando o Arduino na rede com o ENC28J60 Ethernet Shield. Disponivel em : <http://www.Arduinobr.com/Arduino/Arduino_shield/colocando-o-Arduino-na-rede-com-o-enc28j60-ethernet-shield/> Acessado em: 22/07/2016

Curso de arduino. Disponível em : <http://pt.slideshare.net/wellingtoncf1/curso-de-arduino> Acessado em: 20/06/2016

Anexo 01 - Código Arduíno

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


EthernetServer servidor = EthernetServer(8888);

//Definindo Pinos Arduino
int lamp1 = 9;
int lamp2 = 8;
int vent1 = 7;
int bomba = 6;
int sPresenca = 5;
int receptor = 4;
int emissor = 3;
int sTemperatura = A0;
int sFumaca = A1;

int nivelFumaca=85;
int sFum=0;
int codigoIncendio=0;
int indexIncendio=0;
float tempInicial=0;
float temperatura=0;

boolean alarme=false;
int sPren=0;
int codigoAlarme=0;

IRsend emissorIR;
IRrecv receptorIR(receptor);

struct msgIR{
  unsigned long valor;
  int bits;
};

struct msgIR msgonoff;
struct msgIR msgcanmais;
struct msgIR msgcanmenos;
struct msgIR msgvolmais;
struct msgIR msgvolmenos;
struct msgIR msgsource;
```

```
void setup()
{
  msgonoff.valor=1086308415;
  msgcanmais.valor=1086320655;
  msgcanmenos.valor=1086310455;
  msgvolmais.valor=1086316575;
  msgvolmenos.valor=1086283935;
  msgsource.valor=1086285975;

  pinMode (lamp1, OUTPUT);
  pinMode (lamp2, OUTPUT);
  pinMode (vent1, OUTPUT);
  pinMode (bomba, OUTPUT);
  pinMode (emissor, OUTPUT);
  pinMode (sPresenca, INPUT);
  pinMode (sFumaca, INPUT);


 Serial.begin(9600);
 uint8_t EnderecoMAC[6] = {0x00,0x01,0x02,0x03,0x04,0x05};
 IPAddress IP(192,168,1,10);
 Ethernet.begin(EnderecoMAC,IP);
 servidor.begin();
}

void loop(){

 EthernetClient cliente = servidor.available();
  if (cliente) {
    boolean linhaVazia = true;
    String linhaComando="";
    while (cliente.connected()) {
      if (cliente.available()) {
        char c = cliente.read();
          if(linhaComando.length()<20){
            linhaComando.concat(c);
          }

          if (c == '\n' && linhaVazia) {
          cliente.println("HTTP/1.1 200 OK");
          cliente.println("Content-Type: text/html");
          cliente.println();
          cliente.println(temperatura);
          cliente.println(sPren);
          cliente.println(sFum);
          cliente.println(codigoAlarme);
```

```
  cliente.println(codigoIncendio);
  break;

}
if (c == '\n') {
    if(linhaComando.indexOf("LAMP01ON") != -1 ){
      Serial.println("Lampada 01 Acessa");
      digitalWrite(lamp1, HIGH);
    }
    if(linhaComando.indexOf("LAMP01OFF") != -1 ){
      Serial.println("Lampada 01 Apagada");
      digitalWrite(lamp1, LOW);
    }
    if(linhaComando.indexOf("LAMP02ON") != -1 ){
      Serial.println("Lampada 02 Acessa");
      digitalWrite(lamp2, HIGH);
    }
    if(linhaComando.indexOf("LAMP02OFF") != -1 ){
      Serial.println("Lampada 02 Apagada");
      digitalWrite(lamp2, LOW);
    }
    if(linhaComando.indexOf("ONOFF") != -1 ){
        Serial.println("Botao TV ON/OFF");
        emissorIR.sendNEC(msgonoff.valor,32);
    }
    if(linhaComando.indexOf("VOLMAIS") != -1 ){
        Serial.println("Botao TV Volume Mais");
        emissorIR.sendNEC(msgvolmais.valor,32);
    }
    if(linhaComando.indexOf("VOLMENOS") != -1 ){
        Serial.println("Botao TV Volume Menos");
        emissorIR.sendNEC(msgvolmenos.valor,32);
    }
    if(linhaComando.indexOf("CANMAIS") != -1 ){
        Serial.println("Botao TV Canal Mais");
        emissorIR.sendNEC(msgcanmais.valor,32);
    }
    if(linhaComando.indexOf("CANMENOS") != -1 ){
        Serial.println("Botao TV Canal Menos");
        emissorIR.sendNEC(msgcanmenos.valor,32);
    }
    if(linhaComando.indexOf("SOURCE") != -1 ){
        Serial.println("Botao TV Source");
        emissorIR.sendNEC(msgsource.valor,32);
    }
    if(linhaComando.indexOf("VENT01ON") != -1 ){
        Serial.println("Ventilador ON");
        digitalWrite(vent1, HIGH);
    }
```

```
        if(linhaComando.indexOf("VENT01OFF") != -1 ){
           Serial.println("Ventilador OFF");
           digitalWrite(vent1, LOW);
        }
        if(linhaComando.indexOf("ALARMEON") != -1 ){
           Serial.println("Alarme ON");
           alarme=true;
        }
        if(linhaComando.indexOf("ALARMEOFF") != -1 ){
           Serial.println("Alarmes OFF");
           digitalWrite(bomba, LOW);
           alarme=false;
        }
        if(linhaComando.indexOf("INCENDIO") != -1 ){
           Serial.println("Bomba Ativada");
           digitalWrite(bomba, HIGH);
        }

        linhaVazia = true;
        linhaComando="";
    }
    else if (c != '\r') {
      linhaVazia = false;
    }
  }
}
delay(10);
cliente.stop();
}


temperatura = (float(analogRead(sTemperatura))*5/(1023))/0.01 ;

if(analogRead(sFumaca)>nivelFumaca){
  sFum=1;
}

if(analogRead(sFumaca)<nivelFumaca){
  sFum=0;
}


if(sFum==1){
  if(indexIncendio==0){
    tempInicial=temperatura;
  }
  if(indexIncendio<20000){
    indexIncendio=indexIncendio+1;
    float dif = temperatura-tempInicial;
```

```
        if(dif>5){
          codigoIncendio=1;
           Serial.println("Incendio Detectado");
          indexIncendio=0;
        }
        if(dif<10){
          codigoIncendio=0;
        }
      }
      if(indexIncendio==20000){
        tempInicial=0;
        indexIncendio=0;
        codigoIncendio=0;
      }
    }



    if(digitalRead(sPresenca)==HIGH){
      sPren=1;
    }
    if(digitalRead(sPresenca)==LOW){
      sPren=0;
    }

    if(alarme==true && sPren==1){
       codigoAlarme=1;
       Serial.println("Alarme Ativo - Presenca Detectada");
    }

    if(alarme==false){
      codigoAlarme=0;
    }


}
```

Anexo 02 - Código Android

```
package com.example.casacontrole;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {
public String ipServ = "192.168.1.10";
public String ptServ = "8888";
public String URLCMD = "http://"+ipServ+":"+ptServ+"/?CMD=";
public String URLLeitura = "http://"+ipServ+":"+ptServ;
public String statusConexao="DESCONECTADO";
public String temperatura="SEM DETECCAO";
public String senPresenca="SEM DETECCAO";
public String senFumaca="SEM DETECCAO";
public String codIncendio="0";
public String codAlarme="0";
public String codPresenca="0";
public String codFumaca="0";
public  String[] valSensores=new String[5];
int i=0;
```

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
carregarPrincipal();
}

public void carregarPrincipal() {
Cliente cliente =new Cliente(URLCMD+"STATUS");
valSensores=cliente.getValSensores();
decodificaLeitura(valSensores);
if(temperatura!=null){
statusConexao="CONECTADO";
if(codPresenca.contains("1")){
senPresenca="DETECTADO";
}
if(codFumaca.contains("1")){
senFumaca="DETECTADA";
}
if(codIncendio.contains("1")){
alertaIncendio();
}
if(codAlarme.contains("1")){
alertaPresenca();
}
}
setContentView(R.layout.activity_main);
Button carregarLampada = (Button) findViewById(R.id.bLampada);
Button carregarTelevisão = (Button) findViewById(R.id.bTelevisao);
Button carregarVentilador = (Button) findViewById(R.id.bVentilador);
Button carregarSensores = (Button) findViewById(R.id.bSensores);
Button carregarAlarme = (Button) findViewById(R.id.bAlarme);
Button carregarConfig = (Button) findViewById(R.id.bConfig);
```

```
carregarLampada.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
setContentView(R.layout.activity_lampadas);
Button voltarPrincipal = (Button) findViewById(R.id.bVoltar);
Button lamp01ON = (Button) findViewById(R.id.blamp01on);
Button lamp01OFF = (Button) findViewById(R.id.blamp01off);
Button lamp02ON = (Button) findViewById(R.id.blamp02on);
Button lamp02OFF = (Button) findViewById(R.id.blamp02off);

lamp01ON.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"LAMP01ON");
}
});
lamp01OFF.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"LAMP01OFF");
}
});
lamp02ON.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"LAMP02ON");
}
});

lamp02OFF.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
```

```
Cliente cliente =new Cliente(URLCMD+"LAMP02OFF");
}
});
voltarPrincipal.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
carregarPrincipal();
}
});
}
});



carregarTelevisão.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
setContentView(R.layout.activity_televisao);
Button voltarPrincipal = (Button) findViewById(R.id.bVoltar);
Button OnOff = (Button) findViewById(R.id.bonoff);
Button AumentarVolume = (Button) findViewById(R.id.bvolmais);
Button DiminuirVolume = (Button) findViewById(R.id.bvolmenos);
Button AumentarCanal = (Button) findViewById(R.id.bcanmais);
Button DiminuirCanal = (Button) findViewById(R.id.bcanmenos);
Button TrocarSource = (Button) findViewById(R.id.bsource);

OnOff.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"ONOFF");
}
});
AumentarVolume.setOnClickListener(new View.OnClickListener() {
@Override
```

```
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"VOLMAIS");
}
});
DiminuirVolume.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"VOLMENOS");
}
});
AumentarCanal.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"CANMAIS");
}
});
DiminuirCanal.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"CANMENOS");
}
});
TrocarSource.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"SOURCE");
}
});
voltarPrincipal.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
carregarPrincipal();
}
```

```
});
}
});



carregarVentilador.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
setContentView(R.layout.activity_ventilador);
Button voltarPrincipal = (Button) findViewById(R.id.bVoltar);
Button vent01ON = (Button) findViewById(R.id.bvent01ON);
Button vent01OFF = (Button) findViewById(R.id.bvent01OFF);

vent01ON.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"VENT01ON");
}
});
vent01OFF.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"VENT01OFF");
}
});
voltarPrincipal.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
carregarPrincipal();
}
});
}
```

```
});



carregarSensores.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
setContentView(R.layout.activity_sensores);
Button voltarPrincipal = (Button) findViewById(R.id.bVoltar);
TextView textoTemperatura= (TextView) findViewById(R.id.textTemperatura);
TextView textoIncendio= (TextView) findViewById(R.id.textStatusIncendio);
TextView textoPresenca= (TextView) findViewById(R.id.textStatusPresenca);

textoTemperatura.setText(temperatura+" °C");
textoPresenca.setText(senPresenca);
textoIncendio.setText(senFumaca);
voltarPrincipal.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
carregarPrincipal();
}
});
}
});



carregarAlarme.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
setContentView(R.layout.activity_alarme);
Button voltarPrincipal = (Button) findViewById(R.id.bVoltar);
Button alarmeON = (Button) findViewById(R.id.balarmeON);
Button alarmeOFF = (Button) findViewById(R.id.balarmeOFF);
```

```
alarmeON.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"ALARMEON");
}
});
alarmeOFF.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Cliente cliente =new Cliente(URLCMD+"ALARMEOFF");
}
});
voltarPrincipal.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
carregarPrincipal();
}
});
}
});
```

```
carregarConfig.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
setContentView(R.layout.activity_config);
Button voltarPrincipal = (Button) findViewById(R.id.bVoltar);
TextView textoIPServidor= (TextView) findViewById(R.id.textIPServer);
TextView textoPortaServidor= (TextView) findViewById(R.id.textPortServ);
TextView textoStatusConexão= (TextView) findViewById(R.id.textStatusConexao);
```

```
textoIPServidor.setText(ipServ);
textoPortaServidor.setText(ptServ);
textoStatusConexão.setText(statusConexao);
voltarPrincipal.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
carregarPrincipal();
}
});
}
});



}

public void alertaPresenca() {
AlertDialog alertaPresenca;
AlertDialog.Builder alerta = new AlertDialog.Builder(this);
alerta.setTitle("Alerta Presenca");
alerta.setMessage("Foi detectado presenca no local, ativar alarme sonoro");
alerta.setPositiveButton("Ativar",new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
Cliente cliente =new Cliente(URLCMD+"SIRENEON");
}
});
alerta.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
}
});
```

```
alertaPresenca = alerta.create();
alertaPresenca.show();
}

public void alertaIncendio() {
AlertDialog alertaIncendio;
AlertDialog.Builder alerta = new AlertDialog.Builder(this);
alerta.setTitle("Alerta Incendio");
alerta.setMessage("Deseja ativar o sistema de combate a Incendio");
alerta.setPositiveButton("Ativar",new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
Cliente cliente =new Cliente(URLCMD+"INCENDIO");
}
});
alerta.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
}
});
alertaIncendio = alerta.create();
alertaIncendio.show();
}
public void decodificaLeitura(String []leitura){
temperatura=leitura[0];
codPresenca=leitura[1];
codFumaca=leitura[2];
codAlarme=leitura[3];
codIncendio=leitura[4];
}
}
```

Anexo 03 - Código Android - Classe Cliente

```
package com.example.casacontrole;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.net.URISyntaxException;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

public class Cliente implements Runnable{

String URL="";
public static String[] valSensores=new String[5];
public static int i=0;

public Cliente(String Mensagem){
	super();
	URL = Mensagem;
	Thread msg = new Thread(this);
	msg.start();
}

public void fim(){
}

@Override
public void run() {
	try{
		HttpClient cliente = new DefaultHttpClient();
		HttpGet solicitar = new HttpGet();
		solicitar.setURI(new URI(URL));
		HttpResponse resposta = cliente.execute(solicitar);
		BufferedReader leitor = new BufferedReader(new
InputStreamReader(resposta.getEntity().getContent()));
		String linha = "";
		while ((linha = leitor.readLine()) != null)
			if(linha!=null){
				valSensores[i]=linha;
				i=i+1;
			}
```

```
                leitor.close();
                i=0;

        } catch (URISyntaxException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (IllegalStateException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }

}
public static String[] getValSensores() {
        return valSensores;
}
}
```