Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

02/07/2019

AUTOMAÇÃO RESIDENCIAL COM CONTROLE REMOTO VIA SMARTPHONE

Caio de Faria Barros Muniz

PUC RIO

Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

AUTOMAÇÃO RESIDENCIAL COM CONTROLE REMOTO VIA SMARTPHONE

Aluno: Caio de Faria Barros Muniz

Orientador: Mauro Speranza Neto

Trabalho apresentado como requisito parcial à conclusão do curso de Engenharia de Controle e Automação na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

À minha família, amigos e professores.

Resumo

Este projeto consiste em mostrar e explicar a construção de um sistema de automação residencial utilizando um Arduino UNO para controlar os componentes. O sistema se comunicar com um Smartphone através de WI-FI, sendo possível ter acessa-lo em qualquer lugar que possua Internet disponível. Além de toda a programação para sensores e atuadores foi desenvolvido um aplicativo Android a fim de auxiliar a comunicação entre os dois dispositivos.

Palavras-chave: Automação residencial, Arduino, Smartphone, Android

RESIDENTIAL AUTOMATION WITH REMOTE CONTROL VIA SMARTPHONE

Abstract

This project consists of showing and explaining the construction of a residential automation system using an Anduino UNO to control the components. The system will then communicate with a Smartphone through WI-FI, being thus able to have access to the system in any place that has available Internet. In addition to all programming for sensors and actuators, an Android application was developed to aid in the communication between the two devices.

Keywords: Residential automation, Arduino, Smartphone, Android

Sumário

Lista de Figuras

1 Introdução

a Automação residencial

Automação é definida como qualquer atividade que busca e consegue substituir qualquer trabalho, mecânico e mental, de um ser humano ou qualquer outro ser vivo. Portando a automação vem crescendo desde de que se iniciou. Até o momento o principal foco da automação era nas grandes indústrias e fábricas. Com o desenvolvimento de peças menores e a melhor custo, a automação está se expandindo para nossas casas, nossos carros e até mesmo nossos bolsos. Conforme ilustrado na Figura 1.

O conceito de automação residencial não é novo, porém o mesmo tem ganhado uma enorme força em nossos tempos atuais, as pessoas estão aprendendo a aproveitar melhor o seu tempo em casa, no trabalho e em viagens, a fim de garantir este tempo, muitos estão optando pela automação residencial como solução. No mercado já existem diversas empresas que constroem casas automatizadas, porém estas ainda estão muito caras e de difícil acesso. Algumas pessoas compram pequenos acessórios automatizados ou contratam um serviço de instalação de automação[6].

Figura 1: Automação residencial

b Smartphone

Não é nem necessário explicar como os Smartphones estão presentes em nossa vidas. Uma ideia que começou apenas como um simples telefone móvel evoluiu a um nível muito maior que o esperado. Hoje nossos Smartphones possuem milhares de funções, entre as principais encontram-se: Ligações, troca de mensagens, assistir vídeos, consultar e-mails, ... Nossos celulares estão presentes quase tanto como nós mesmos.

c Objetivo

Como o objetivo deste projeto é a tentativa de montar um sistema completo para se automatizar uma residência, que possua internet a baixo custo e com fácil manutenção. Para realizar o controle dos atuadores e ler os sensores será utilizado um Arduino Uno. Para a realização do controle da casa deverá ser utilizado um Smartphone com a plataforma Android e com um aplicativo desenvolvido para o projeto em específico. Veja na figura 2.

Figura 2: Projeto

2 Componentes e plataformas

Nesta seção será indicado todos os sensores, atuadores, controladores e plataformas(Softwares) utilizados para o desemvolvimento do projeto, assim como suas principais caracteristicas.

a Hardware

1 Arduino

O Arduino é um microcontrolador com entradas e saídas analógicas e digitais. Através das placas Arduino é possível controlar diversos atuadores em sincronia com diversos sensores. Seu principal objetivo é ser um controlador de baixo custo e fácil disponibilidade para novatos e veteranos tanto para fins comerciais, domésticos ou móveis. O Arduino é uma placa tão utilizada que é possível comprar vários tipos de sensores, atuadores e SHIELDS compatíveis pela internet sem muito esforço.

Para este projeto a placa utilizada foi "BlackBoard UNO R3" [Figura 3], uma versão fabricada pela Robocore no Brasil. Por ser fabricada pela Robocore a mesma possui algumas funções otimizadas para alguns procedimentos, porém as alterações realizadas durante a fabricação não são necessárias para realização do projeto.

Uma placa de Arduino 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. Veja o desenho esquemático na Figura 4.

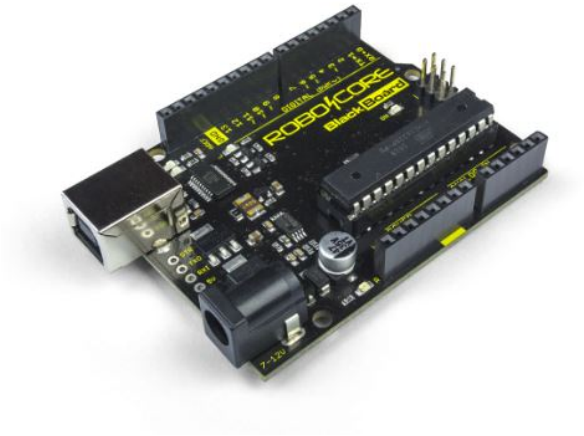

Figura 3: BlackBoard UNO R3

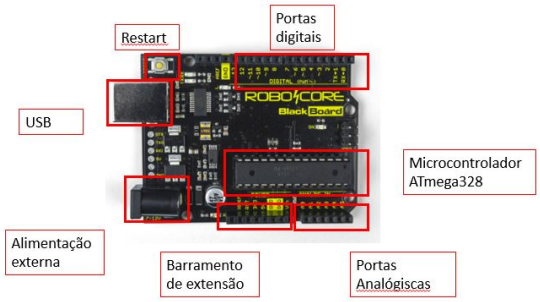

Figura 4: Esquemático da placa BlackBoard UNO R3

2 Ethernet Shield

Existem diversas maneiras de se interagir com o Arduino, seja através do cabo USB, bluetooth, cabo de rede, WI-FI entre outros. O proposito deste projeto é fazer a interação entre o smartphone e o Arduino de forma que qualquer pessoa com acesso ao aplicativo e a qualquer distancia possa controlar a sua casa de forma devida, desta forma foi escolhido o Ethernet Shield W5100 [Figura 5], devido ao preço e disponibilidade no momento da realização do projeto, além da placar apresentar diversas bibliotecas de códigos desenvolvidas pela comunidade. Por se tratar de um Shield que possui entrada para cabo de rede, é solicitado que durante a montagem do projeto o Arduino fique próximo ao moldem, ou um extenso cabo de rede.

Este Arduino Ethernet Shield baseia-se no chip WIZnet ethernet W5100 que fornece acesso à rede (IP) nos protocolos TCP ou UDP e é facilmente utilizado usando a biblioteca Ethernet Library e SD Library. Ele é compatível tanto com o Arduino Uno e Mega e possui um slot para cartão Micro-SD que pode ser usado para armazenar arquivos que vão servir na rede. As portas utilizadas para comunicação do módulo com o Arduino são os 10,11,12 e 13, além de utilizarem dois pinos para alimentação (VCC e GND).

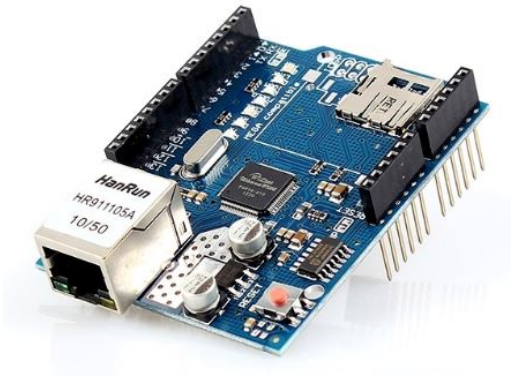

Figura 5: W5100

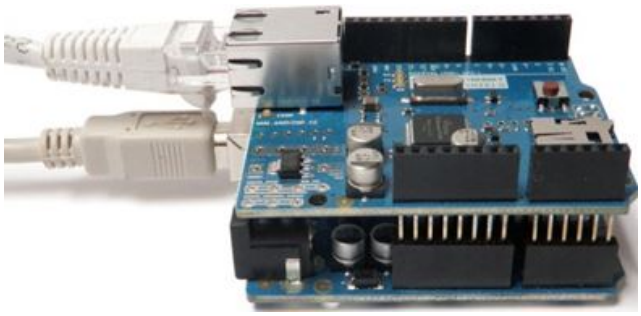

Figura 6: W5100 acoplado ao arduino

3 Shield Relé 1 e 4 Canais

Em uma casa existem vários dispositivos que operam em uma voltagem muito maior que 5V, voltagem fornecida pelo Arduino. Portanto foi criado o Shiled de relés. No mercado existem diversos Shileds de relés disponíveis para compra, contendo apenas um relê até uma quantidade máxima de 8, é possível através de uma fabricação própria produzir Shildes com mais relés, porém como neste projeto é visado apenas operar 5 dispositivos de alta tensão, foi escolhido o Shiled com 1 [Figura 7] e outro com 4 relés [Figura 8].

Cada relé 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 é possível 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. O Shield com 1 canal possui 3 entradas VCC, GND, e 1 entrada para o sinal. O Shield com 4 canais possui 2 entradas de alimentação, VCC - GND, e 4 entradas, cada uma designada para um relê especifico[3].

Figura 7: Shield relé 1 Canal

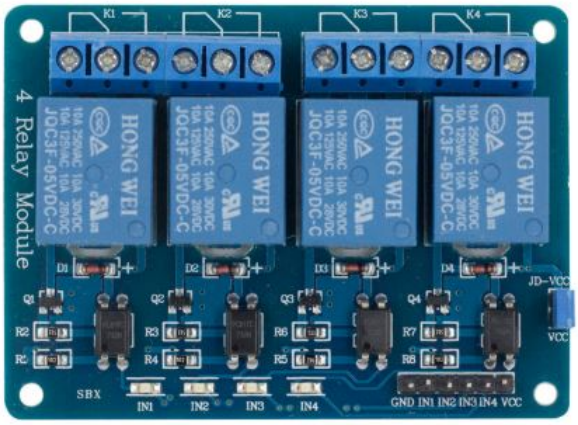

Figura 8: Shiled relê 4 Canais

4 Lampadas de LED

Por limitações de entradas e saídas do Arduino foi definido que a casa para este projeto será controlada somente as luzes da sala, do quarto e da garagem. A fim de economizar energia e por possuir melhor iluminação foi escolhido uma Lâmpada Bulbo LED OsRAM 7014402 12 W Branco com o bocal LED Plafonier Termoplástico, LLUM Bronzearte [Figura 9] [2].

Figura 9: Lâmpada e Bocal

5 Ventilador

Para este projeto foi definido que será utilizado um ventilador comum de 120V, que será controlado de modo ON/OFF através do aplicativo[Figura 10].

Figura 10: Ventilador

6 Sensor de Presença PIR - HC-SR501

Para o projeto foi considerado que a casa possui uma garagem, quando uma pessoa está dirigindo o seu carro, não é recomendada a utilização de um celular, portanto na garagem será instalado um sensor de presença PIR - HC0SR501, veja na Figura 11. Além de ascender a luz quando a pessoa entre em casa com o carro o sensor pode também servir como um dispositivo de segurança par o lugar, sempre indicando que alguém está entrando na garagem.

O sensor PIR HC-SR501 é um dos sensores de movimento mais simples de usar do mercado, e totalmente

compatível com Arduino. São necessários 5 segundos para sua inicialização. Após este período, se qualquer coisa se mover em seu range de leitura, o pino de saída vai para nível lógico alto. Este nível de saída é de 3,3V, portanto pode ser facilmente entendido por uma placa Arduino em uma leitura digital. Apesar de o sensor apresentar falhas em alguns momentos ele foi mantido no projeto por ter uma performance razoável.

Figura 11: Sensor PIR HC-SR501

7 Sensor de Gás e Fumaça MQ-2 e bomba de incêndio

Incêndios são e sempre foram um grande problema na sociedade, eles geralmente ocorrem sem aviso e se não contido de forma imediata podem destruir cômodos, casas, prédios e por vezes até quarteirões. Portanto foi implementado o uso do sensor MQ-2, um sensor pequeno, baixo custo e boa precisão. Para este projeto apenas a utilização de um alinhado com uma bomba d'aqua foi implementado. Devido a complexidade de infraestrutura de se utilizar uma bomba capaz de apagar o incêndio em uma casa, a bomba d'agua será apenas demostrada como um dispositivo genérico no circuito que serão demostrados nos capítulos a seguir.

Figura 12: Sensor MQ-2

O sensor de gás inflamável e fumaça (MQ-2) é capaz de detectar a presença de gás de diferentes tipos: GLP, butano, propano, metano, hidrogênio, ... Uma função extra do sensor MQ-2 é que além de detectar os gases é possível também realizar a detecção de fumaça no ambiente. O sensor possui os terminais de alimentação VCC - GND) e pode ser utilizado de duas maneiras: 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; Analogica - o microcontrolador é capaz de saber o nível de concentração de gases detectados pelo sensor. A resposta analógica é de acordo com o gráfico apresentado na Figura 13; Como alarme será utilizado uma Buzzer passivo de 5V [Figura 14].

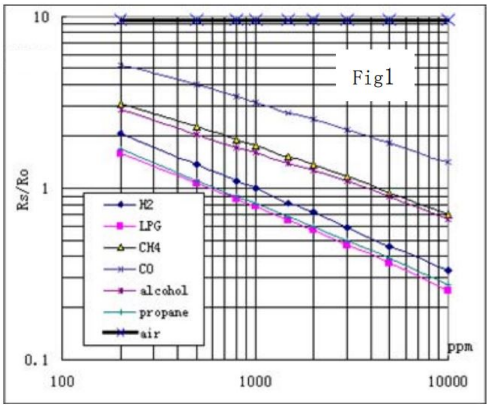

Figura 13: Grafico de Sensibilidade de Sensor

Figura 14: Buzzer Passivo

8 Sensor de temperatura LM35

Algumas pessoas, principalmente moradores do interior do país, gostam de saber a temperatura ambiente da sua casa ou do lado exterior, portanto, foi implementado para este projeto o sensor de temperatura LM35 [Figura 15].

O sensor de temperatura LM35 fornece uma leitura dentro da faixa de -55ºC até 150ºC com variações de 1/4ºC até 3/4ºC com extrema exatidão. Ao ser alimentado por uma tensão de 4V-20V e GND apresenta em sua saída uma tensão linear referente a temperatura, internamente o sensor já possui uma calibração para que a cada 10mV de variação na corrente indica a variação de 1ºC.

Figura 15: Sensor LM35

b Software

1 Arduino IDE

O Arduino IDE [Figura 16] é um ambiente de desenvolvimento integrado, ou seja, um software onde é possível realizar toda a programação de um projeto a ser realizado em um placa Arduino. O ambiente é compatível a todas as placas do Arduino e tem ligação direta para upload do código na placa, portanto ao realizar a conexão entre o Arduino e PC via cabo USB é possível dar Upload no código e testar seu projeto diretamente. O IDE é compatível com Windowns, Linux e Mac OS. Foi desenvolvido utilizando linguagem Java e Processing. Dentro do programa já é possível encontrar várias bibliotecas desenvolvidas pelos fabricantes, monitor serial, correção de erros e destaque de sintaxe[7].

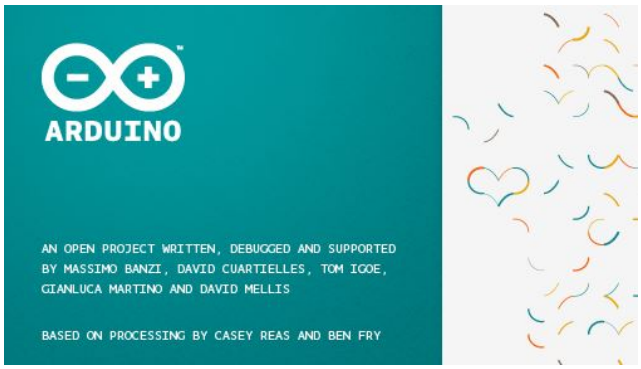

Figura 16: Arduino IDE

Devido a biblioteca chamada "Wiring", já inserida no programa, é possível a realizar toda a programação de C/C++. As funções foram desenvolvidas de maneira intuitivas e de fácil entendimento, a fim de tonar a programação mais acessível a novos programadores e projetistas sem conhecimento em programão. Ao iniciar o aplicativo o programador se depara com as duas funções principais do IDE Setup() e Loop(). Para conectar a placa Arduino ao programa é necessário escolher a placa utilizada e a porta serial que ela se encontra [Figura 17].

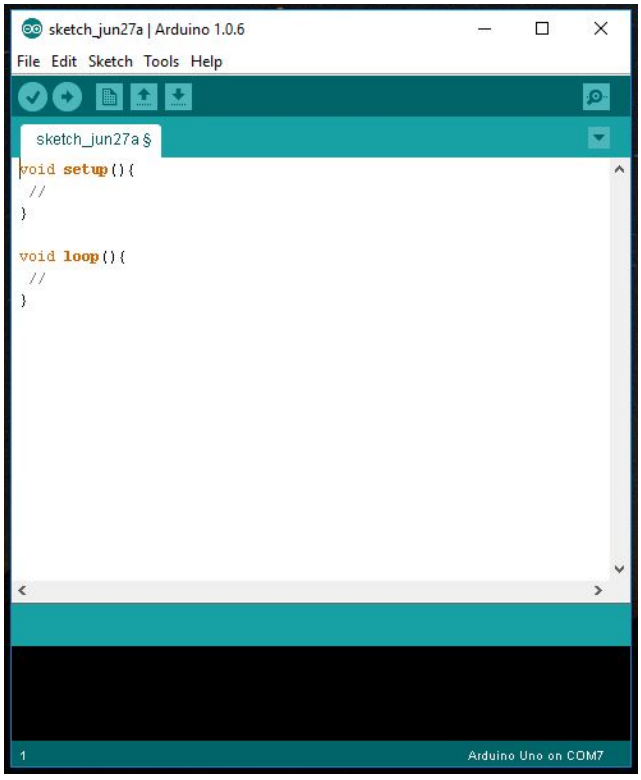

Figura 17: Interface Arduino IDE

Função Setup(): Esta função é a primeira a ser executada no programa, ela será executada apenas uma única vez. Portanto a fim de se economizar velocidade de processamento e processos desnecessários é recomendado colocar todos valores constantes ou/e funções que devem ocorrer apenas quando o Arduino for ligado. Após a execução da função setup() a função loop() é executada.

Função Loop(): Está função, conforme informado anteriormente, será executada após a logo após a função Stup() terminar. A partir da primeira execução ela será executa infinitas vezes, ou até o botão reset seja pressionado ou o Arduino não possua mais energia. Nesta função deve ser escrita a rotina de seu projeto.

Vale resaltar que uma vez que se faça o upload do código para o Arduino, ele sempre executará o mesmo código, mesmo após se desligado e ligado novamente. Caso se deseje que o Arduino não rode nenhum código deve ser feito o upload de um código totalmente em branco.

2 MIT App Inventor

Para fazer a comunicação entre o smartphone e o sistema físico, foi desenvolvido um aplicativo para Android, sendo assim o acesso do usuário fica muito mais fácil e rápido. A plataforma utilizada para programar o aplicativo foi o MTI App Inventor [Figura 18].

Figura 18: MIT App Inventor

Com o grande sucesso dos smartphones e seus aplicativos o professor Hal Abson em parceria com a Google Education iniciou a criação da plataforma MIT App Invetor, sendo uma plataforma com programação baseada em blocos, totalmente gratuita é possível utiliza-la apenas iniciando o seu navegador [Figura 19]. Sendo assim, a ferramenta se tornou alvo de todo o tipo de programadores e projetistas, hoje conta uma comunidade mundial de quase 3 milhões de usuários, representando 195 países em todo o mundo. Mais de 100 mil usuários ativos semanalmente e que já construíram mais de 7 milhões de aplicativos para Android [4][8].

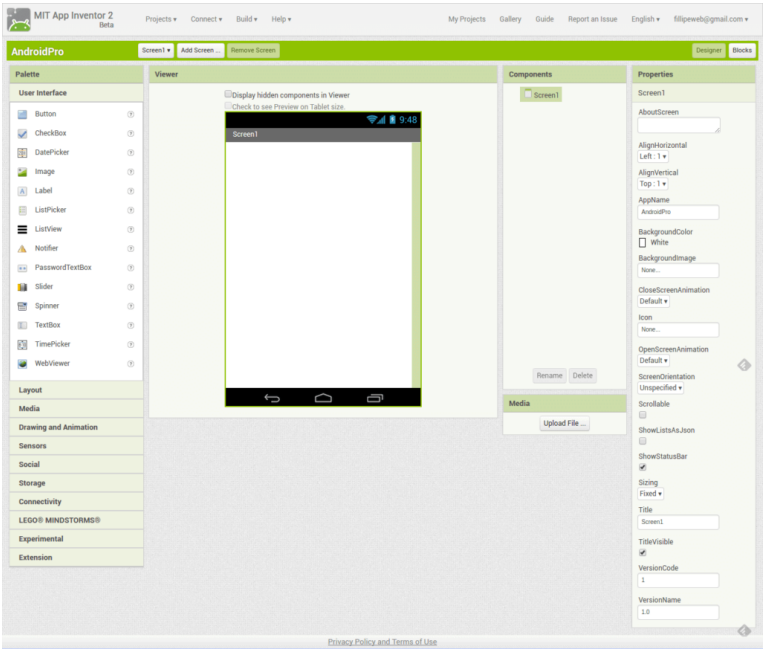

Figura 19: Interface MIT App Inventor

3 Design do Projeto

Este capítulo tem como objeto descrever etapa por etapa da construção do projeto. Será testados nos itens a seguir o desenvolvimento da parte física/elétrica e a parte de programação, tanto do Arduino quanto aplicativo.

a Sistema Fisico e Elétrico

1 Ardunio - Shield relé 1 Canal

A Figura 20 representa o esquema elétrico para realizar a conexão entre o Arduino e o Shield de relé com 1 canal. o Shield possui VCC, GND, I1. Na entrada I1 encontra-se o Ventilador - Pino 8 [Figura 21].

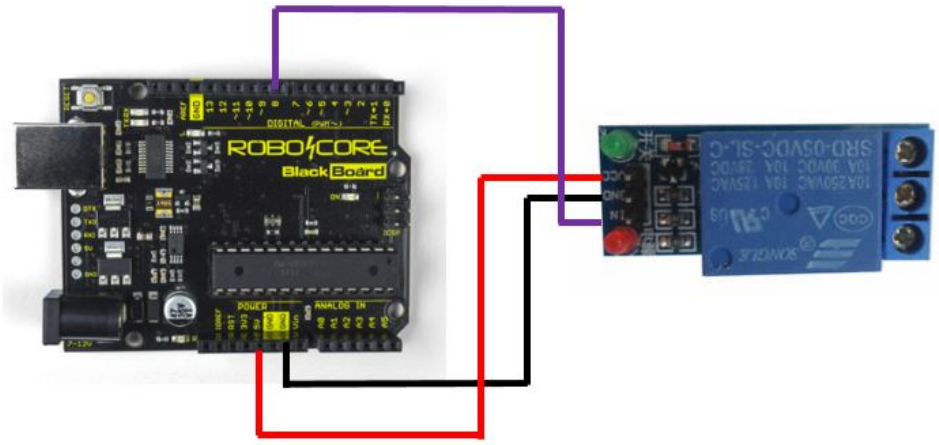

Figura 20: Arduino - Shield relé 1 Canal

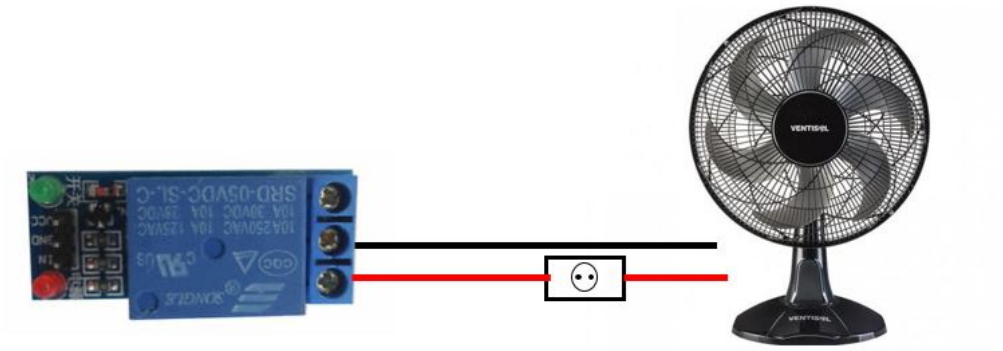

Figura 21: Shield relé 1 Canal - Ventilador

2 Arduino - Shield relé 4 Canais

A Figura 22 representa o esquema elétrico para realizar a colexão entre o Arduino e o Shield de relé. o Shield possui VCC, GND, I1, I2, I3, I4. Na entrada I1 encontra-se a lâmpada da Sala - Pino 3, I2 - lâmpada do quarto - Pino 4, I3 - Lâmpada garagem - Pino 5, I4 - Bomba de incendio - Pino 6 [Figura 23].

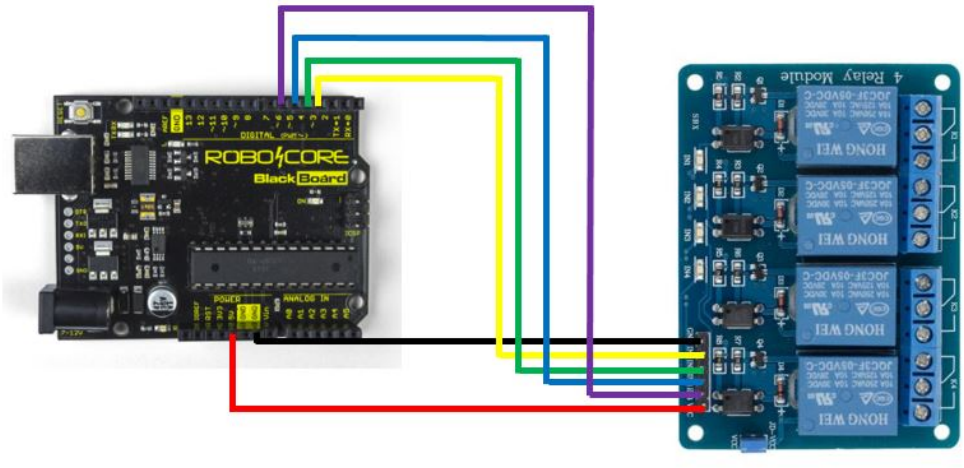

Figura 22: Arduino - Shield relé 4 Canais

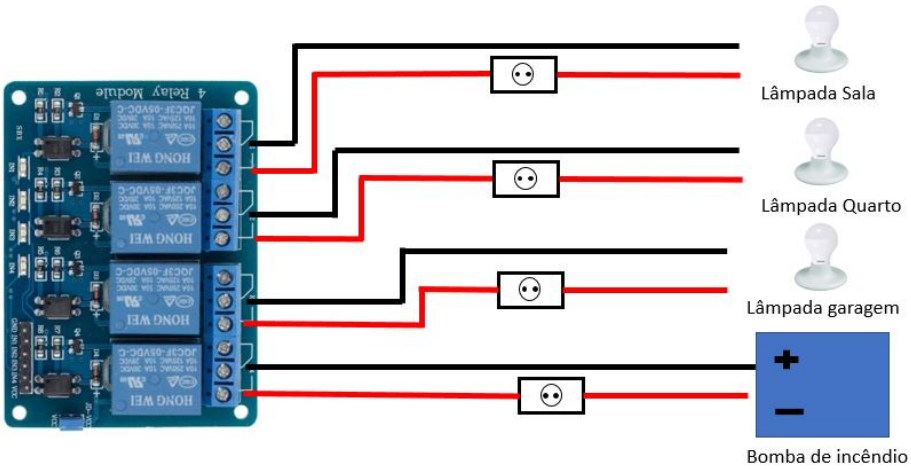

Figura 23: Shield relé 4 Canais - Dispositivos de alta voltagem

3 Arduino - Sensor PIR HC-SR501

A Figura 24 representa o esquema elétrico para realizar a conexão entre o Arduino e o sensor PIR HC-SR501. O sensor possui 3 pinos, VCC, GND e o pino meio é o sinal de presença.

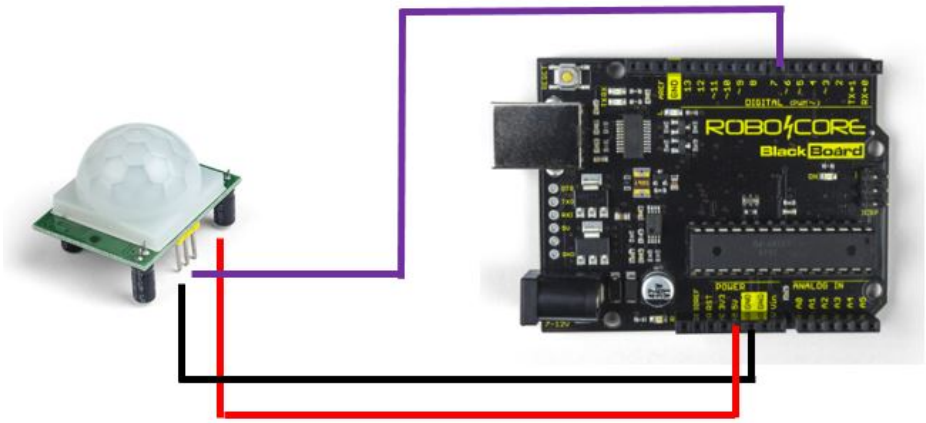

Figura 24: Arduino - Sensor PIR HC-SR501

4 Arduino - Sensor MQ-2

A Figura 25 representa o esquema elétrico para realizar a conexão entre o Arduino e o sensor MQ-2. O Sensor possuiu 4 entradas, VCC, GND, Pino Digital e Pino analógico. Foi definido para este projeto que será utilizado o pino analógico conforme será explicado na programação.

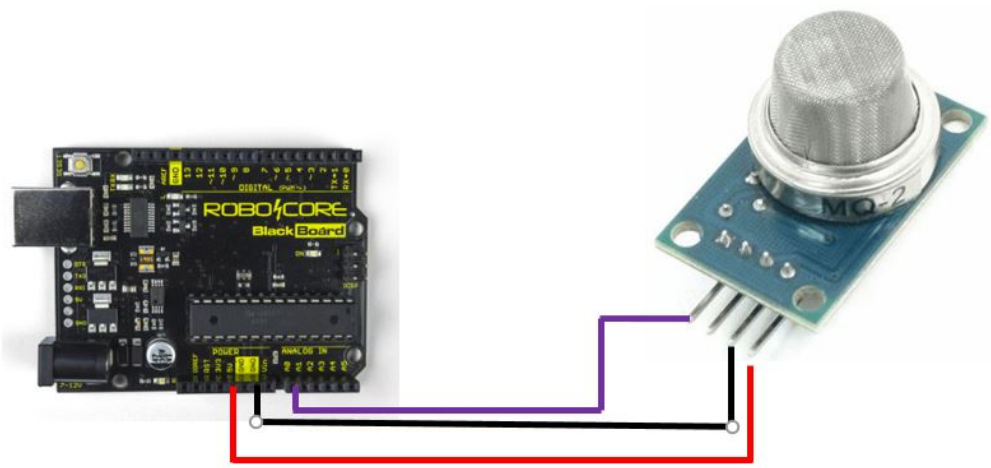

Figura 25: Arduino - Sensor MQ-2

5 Arduino - Sensor LM35

A Figura 26 representa o esquema elétrico para realizar a conexão entre o Arduino e o sensor LM35. O sensor possui 3 pinos , VCC ,GND e i pino central da o retorno com base na temperatura medida.

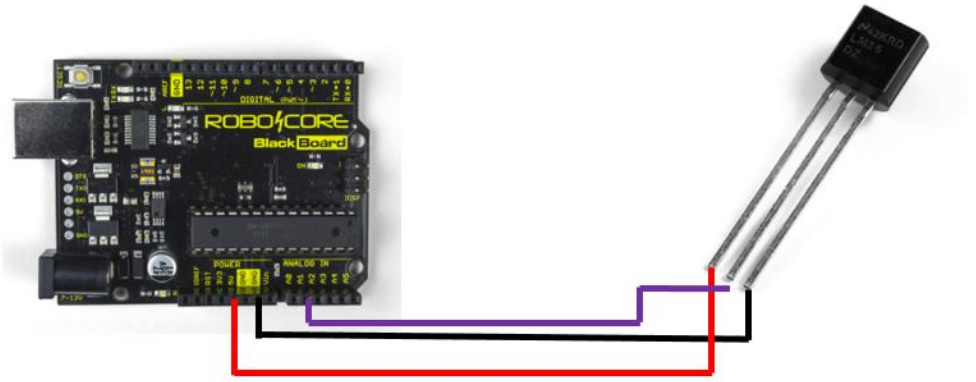

Figura 26: Arduino - Sensor LM35

6 Arduino - Buzzer

A Figura 27 representa o esquema elétrico para realizar a conexão entre o Arduino e o Buzzer Passivo. O Buzzer possui 2 pinos , Entrada ,GND.

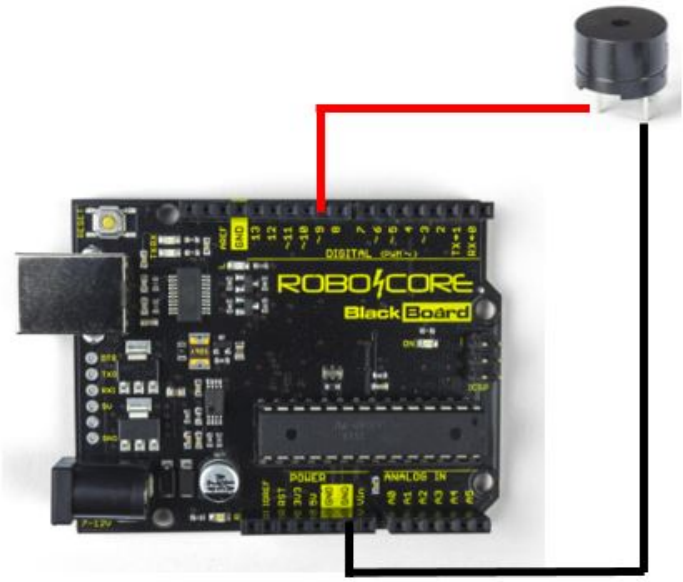

Figura 27: Arduino - Buzzer

b Comunicação entre os dispositivos

A comunicação entre o sistema físico e aplicativo é realiza através do protocolo HTTP. O Ethernet Shield possui várias bibliotecas prontas para tornar a utilização da Internet algo fácil e viável para qualquer programador iniciante, portanto foi decidido utiliza-las.

Primeiramente é necessário através do Arduino IDE configurar o IP e endereço MAC do Ethernet Shiled para serem um número fixo e não mudarem a todo instante. Em paralelo o modem da casa também para fixar o número IP do Arduino, sendo evitando a troca de IP caso o Arduino seja desligado por algum motivo desconhecido. É necessário também configurar o as portas do modem para enviar qualquer mensagem que chegue pela porta 80 para o IP, agora fixo, do Arduino. Desta forma, quando o cliente, aplicativo Android, encaminhar comando ou/e solicitar respostas desta porta, será possível realizar a comunicação [1].

Após todo este processo o sistema estará pronto para funcionar. Qualquer solicitação realizada pelo celular será enviada ao Arduino e o mesmo irá enviar respostas através do protocolo HTTP que pode ser coletada pelo aplicativo. Porém até o momento o sistema só vai funcionar se o celular estiver conectado a mesma rede do Arduino. Ao realizar uma pesquisa intensa sobre o fato, foi constado que deveria ser fixado o DNS dinâmico.

O DNS dinâmico (DDNS ou Dynamic DNS) é um método para atualização automática de um servidor de nomes no Domain Name System (DNS), em tempo real, com a configuração ativa de DDNS dos nomes de host configurados, endereços ou outras informações. Está medida é adotada por todos os provedores de Internet no Brasil, sua principal função é fornecer segurança para as redes caseiras, quando uma casa possuiu DNS dinâmico a cada 24 horas o seu IP de internet muda, sendo assim hackers têm mais dificuldade de acessar a sua rede e os seus dispositivos, provendo, então maior segurança aos consumidores.

Para acessar a rede é necessário utilizar um serviço que de alguma forma conheça seu IP real e crie um link entre o aplicativo e o moldem através de um DNS fixo. Diversas empresas proveem este tipo de serviço. Cada modem é diferente, e é necessário entrar nas configurações para identificar qual serviço é aceito pelo seu provedor e pelo seu modem, no caso deste projeto o site utilizado para realizar a comunicação foi o NO-IP [5]. No-ip é um serviço gratuito e realiza exatamente o esperado para este projeto. Foi então criado um servidor com o nome:http://cmunizsolutions.ddns.net/. Além disso foi necessário, através das configurações do moldem, dar permissão por acesso remoto.

Para solucionar este problema de forma mais simples pode-se alugar um servidor de alguma empresa e realizar a troca de dados através deste servidor. O controle de software foi realizado de forma que as informações sejam comparadas nos dois sistemas, garantindo mais segurança ao sistema. Sendo assim os dados são tratados em ambas plataformas, MTI app Inventor e Arduino IDE.

c Arduino IDE

1 Bibliotecas

Para a realização do projeto foram incluidas duas bibliotecas "Ethernet.h", responsável por fornecer as funções para troca de informação com Internet e "SPI.h", responsável por fornecer as funções entre troca de informações entre o Arduino e o Ethernet Shiled. Veja o código abaixo.

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

2 Constantes e funções

Foi necessário fixar o IP do Arduino e indicar qual é o MAC do Ethernet shield utilizado, abrir a porta/servidor para comunicação e executar a função “readString” para ler o comando enviado pelo aplicativo. Além disto foi definido as constantes e pinosde saída do Arduino, conforme a Tabela 1, para seu respectivo sensor e nomes de variáveis a serem utilizadas conforme o programa a seguir.

Sensor/Atuador	Pino de Saida/Entrada
LM35	1
MQ-2	0
Lâmpada Sala	3
Lâmpada Quarto	4
Lâmpada Garagem	5
Bomba de Incêndio	6
PIR HC-SR501	7
Ventilador	8
Alarme Sonoro	9

Tabela 1: Pinos

```
byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
byte ip[] = { 192, 168, 1, 22 };

EthernetServer server(80);

String readString;

const int LM35 = 1;
float temperatura = 0;
int ADClido = 0;
int Gas = 0;
int GasLido = 0;
```

```
int LimGas = 70;
int lamp_sal = 3;
int lamp_qua = 4;
int lamp_gar = 5;
int bomba =6;
int presenca = 7;
int Ventilador = 8;
int buzz = 9;
int ssal =0;
int squa=0;
int sgar=0;
int sbomb=0;
int sacinoamento=0;
int svent=0;
int salarm=0;
String sal;
String qua;
String gar;
String bomb;
String acinoamento;
String vent;
String alarm;
```

3 Setup()

Na função Setup() será aberto o canal de comunicação entre o Enthernet Shield e o Arduino através da função "Ethernet.begin" e iniciar o servidor. Também é aberto o canal com o monitor serial do próprio programa para testes, caso seja necessário em algum momento. Define-se cada pino como entrada ou saída do Arduino [Tabela 2], caso seja um sensor o pino deve ser definido com entrada, caso controle um atuador o pino é definido com saída. Por último é executado a função "analogReference" para normalizar as entradas analógicas. Veja o código utilizado a seguir.

```
void setup(){
  Ethernet.begin(mac, ip);
  server.begin();
  server.begin();
  pinMode(LM35, INPUT);
```

```
  pinMode(Gas, INPUT);
  pinMode(Lamp_Sal, OUTPUT);
  pinMode(Lamp_qua, OUTPUT);
  pinMode(Lamp_gar, OUTPUT);
  pinMode(bomba, OUTPUT);
  pinMode(presenca, INPUT);
  pinMode(ventilador, OUTPUT);
  pinMode(buzz, OUTPUT);

  analogReference(INTERNAL);
}
```

Sensor/Atuador	Pino de Saida/Entrada
LM35	INPUT
MQ-2	INPUT
Lâmpada Sala	OUTPUT
Lâmpada Quarto	OUTPUT
Lâmpada Garagem	OUTPUT
Bomba de Incêndio	OUTPUT
PIR HC-SR501	INPUT
Ventilador	OUTPUT
Alarme Sonoro	OUTPUT

Tabela 2: Saidas e Entradas

4 Loop()

Durante a função loop, o Arduino começa conferindo se o servido está ligado. Caso o servidor não apresente problemas o programa entra em um "while(client.connected)", ou seja, enquanto o cliente (servidor aberto) estiver conectado o programa vai repetir todas a linhas de comando dentro do "while". O primeiro passo a se realizar é a leitura do "comando enviado" pelo cliente. Neste caso, comando enviado pelo aplicativo Android. Por questões internas do Arduino o tamanho máximo de Sting no enviado pelo cliente deve ser de 100 caracteres, como enviaremos apenas um comando por vez pelo aplicativo, isto não será um fator complicador para este projeto. A sequência de comandos é:

```
void loop(){
  EthernetClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
```

```
if (readString.length() < 100) {
  readString += c;
}
```

Se algum comando for recebido pelo Arduino, o programa entra no "if" onde todo o codigo base está inserido. Caso nenhum comando seja recebido toda o while se repete até que um comando seja enviado. Sesta forma é possível evitar delays de leituras por parte do Arduino. Ao receber o comando o Arduino inicia uma pagina em protocolo HTTP no servidor, quem tem como endereço o IP do Arduino. A página tem como objetivo realizar a troca de dados entre aplicativo e Arduino. A sequência de comandos é:

```
if (c == '\n') {

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

          client.println("<HTML>");
          client.println("<BODY>");
          client.println("<H1>CM solutions </H1>");
          client.println("<hr />");
          client.println("<br />");
```

Neste ponto o Arduino começará a enviar à página web, quais são os estados dos atuais das saídas, ou seja, dos dispositivos integrados aos sistemas. Veja na Tabela 3 os nomes das variaveis dedicadas a cada sensor. O estado dos dispositivos será alterado de acordo com o comando lido do aplicativo em parte seguinte do código. A fim de facilitar a instalação foi inserido no código, um teste, onde é possível alterar o status de uma saída apenas pelo click na página web do servidor. a página é então finalizada e a comunicação entre Arduino e o cliente finalizada, de acordo com o código abaixo:

```
          client.println(temperatura); //printa temp
          char buff[3];
          sprintf (buff,"%03d",GasLido);
          client.println(buff); // printa gas
          client.println(sal); // printa estado sala
          client.println(qua);// printa estado quarto
          client.println(gar);// printa estado garagem
          client.println(bomb);// printa estado bomba
          client.println(vent);// printa estado ventilador
          client.println(alarm);// printa estado alarme
```

```
client.println("<a href=\"/lamp01\">Ligar o led</a>"); // teste
client.println("<a href=\"/lamp02\">Desligar o led</a><br />"); //teste

client.println("</BODY>");
client.println("</HTML>");

delay(1);
client.stop();
```

Sensor/Atuador	Variavel
LM35	temperatura
MQ-2	buff
Lâmpada Sala	sal
Lâmpada Quarto	qua
Lâmpada Garagem	gar
Bomba de Incêndio	bomb
Ventilador	vent
Alarme Sonoro	alarm

Tabela 3: Variavel refente ao Status do dispositivo

Após a finalização de comunicação com o servidor é realizada a leitura dos sensores do sistema, primeiramente é iniciado a leitura da temperatura seguida pela leitura de gás e do sensor de presença.

```
ADClido = analogRead(LM35);//le sensor de temp
temperatura = ADClido * 0.1075268817204301;
GasLido = analogRead(Gas); //Le sensor de gas
GasLido = map(GasLido, 0, 1023, 0, 100);
acionamento = digitalRead presenca);
```

A partir da leitura dos sensores é realizada a alteração do status dos atuadores do sistema: Lâmpada Sala, Lâmpada Quarto, Lâmpada Garagem, Bomba de incêndio, Ventilador e Alarme Sonoro. Veja na Tabela 4 o nome das variáveis utilizadas de acordo com o status desejado. A Bomba de incêndio é ligada se o sensor de gás ler um valor maior que 80 por cento de sua sensibilidade, junto com a bomba é acionado o alarme, tanto no aplicativo quanto o alarme sonoro do sistema físico. Por fim o comando lido do aplicado é reinicializado. A sequência de comando é:

```
if(readString.indexOf("luz%20sala") > 0)
  {
    if(ssal==1){
    digitalWrite(lamp_sal, LOW);//Desliga lampada sala
    sal="lamp01";
```

```
  ssal=0;
  }
  else
  {
  digitalWrite(lamp_sal, HIGH);//liga lampada sala
    sal="lamp02";
    ssal=1;
  }
}
    if(readString.indexOf("luz%20quarto") > 0)
{
  if(squa==1){
  digitalWrite(lamp_qua, LOW);//desliga lampada Qauarto
  qua="lamp03";
  squa=0;
  }
  else
  {
  digitalWrite(lamp_qua, HIGH);//liga lampada quarto
    qual="lamp04";
    squal=1;
  }
}


  if(acionamento==0)//desliga lampada garagem
{
  digitalWrite(lamp_gar, LOW);
  gar="lamp05";
  sgar=0;
}

if(acionamento==1)//liga lampada garagem
  {
    digitalWrite(lamp_gar, HIGH);
    gar="lamp06";
    sgar=1;
```

```
        }
        if(readString.indexOf("ventilador") > 0)
      {
        if(svent=1){
        digitalWrite(ventilador, LOW);//desliga ventilador
        vent="vent01";
        sven=0;
        }
        else
        {
           digitalWrite(ventilador, HIGH);//liga Ventilador
          vent="vent02";
          sven=1;
        }
      }

       if (GasLido > LimGas) || readString.indexOf("ventilador") > 0
       {

           digitalWrite(buzz, LOW);//desliga alarme
           digitalWrite(bomba, LOW);//desliga bomba
          bomb = "bomb01"
           alarm = "alarm01"
        }
        else
        {
           digitalWrite(buzz, HIGH);//liga alarme
           digitalWrite(bomba, HIGH);//liga bomba
          bomb = "bomb02"
           alarm = "alarm02"
        }


      readString="";
    }
  }
}
```

```
  }
}
```

Sensor/Atuado	Variavel	Status
Lâmpada Sala	lamp01	Desligado
Lâmpada Sala	lamp02	Ligado
Lâmpada Quarto	lamp03	Desligado
Lâmpada Quarto	lamp04	Ligado
Lâmpada Garagem	lamp05	Desligado
Lâmpada Garagem	lamp06	Ligado
Bomba de Incêndio	bomb01	Desligado
Bomba de Incêndio	bomb02	Ligado
Ventilador	vent01	Desligado
Ventilador	vent02	Ligado
Alarme Sonoro	alarm01	Desligado
Alarme Sonoro	alarm02	Ligado

Tabela 4: Status do dispositivo

d MIT App Inventor

1 interface

A interface do aplicativo foi desenvolvida de modo a deixar a interação entre o dono da casa e o sistema físico o mais simples possível. Portanto o aplicativo apresenta apenas uma tela, contendo 9 espaços, ou 9 blocos, cada um tendo sua função específica para atender aos sensores e atuadores específicos.

Figura 28: Interface do Aplicativo

2 Inicialização

Ao inicializar o aplicativo, 3 variáveis globais são inicializadas. A primeira, referente ao resultado do recurso de reconhecimento de voz, a segunda o “IParduino” e por último as possíveis cores para personalização [Figura 29].

Figura 29: Váriaveis Globais

Se o aplicativo já foi inicializado anteriormente, isto quer dizer que o IP do servidor do Arduino já foi definido anteriormente, por consequência o programa já armazena na variável "iparduino", o IP guardado no banco de dado, no MIT app inventor denominado como “TintDB1”. Caso está não seja a primeira vez utilizando o app, é possível opera-lo sem nenhuma configuração adicional [Figura 30].

Figura 30: Definindo "iparduino"

3 Inserindo Novo ID

Durante a instalação é necessário cobrir todos os pontos definidos nos capítulos anteriores, portanto ao final da instalação o usuário vai obter um endereço com DNS dinâmico. A configuração de um novo IP deve sempre ser utilizada quando for a primeira utilização do app, ou quando, por algum motivo desconhecido o endereço IP do Arduino for alterado. Na interface inicial do app tem um campo onde é possível digitar o endereço IP do sistema. No caso deste projeto foi criado o endereço "http://cmunizsoluitons.ddns.net/", este endereço será utilizado para realizar a comunicação entre Arduino e aplicativo, sendo possível enviar comandos e receber o status dos dispositivos físicos [Figura 31].

Figura 31: Digitar Novo IP

Após aplicar o novo IP, clicando no botão "OK", o endereço novo é salvo no banco de dados no lugar do antigo, ou em caso de primeira utilização, é criado um novo espaço na memória para isto. O endereço então é armazenado e uma mensagem "Alterado com sucesso!" aparece na tela [Figura 32].

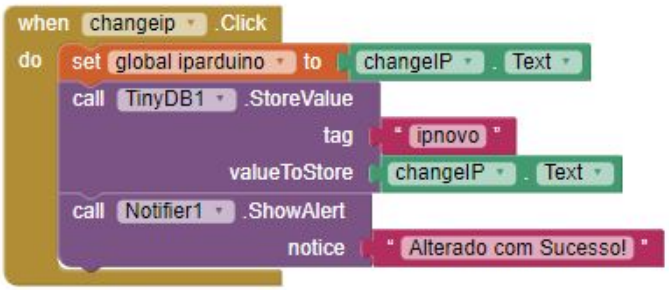

Figura 32: Configurando Novo IP

4 Envio de Comandos

As duas primeiras imagens do aplicativos são referentes ao controle das lâmpadas da sala e do quarto. Ao clicar em uns dos botões o aplicativo acesa a web com o ip concatenado com o chamando desejado. Por exemplo [Figura 33]:

"http://cmunizsoluitons.ddns.net/luz%20sala"

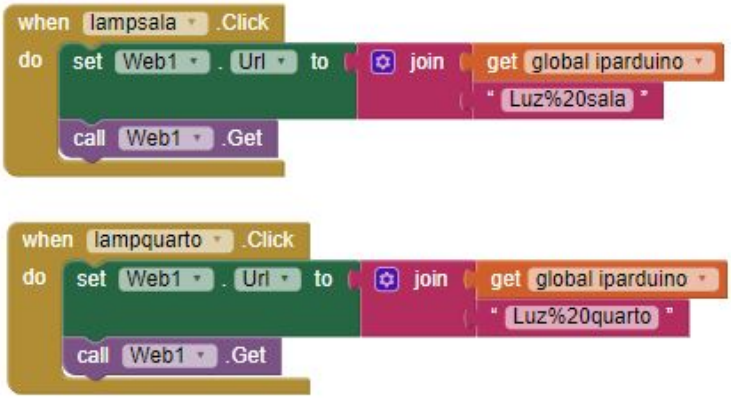

Figura 33: Enviando comando para Lampadas

O terceiro bloco, ou botão, é destinado para comando de voz, neste projeto foi definido que é possível enviar 4 comandos de voz para o aplicativo. O aplicativo vai ler o comando de voz interpretado pelo próprio aplicativo da Google. Os comandos possíveis para interagir com o aplicativo são: "Luz Sala", "Luz Quarto", "Ventilador" e por fim "Desligar Alarme", qualquer outro comando ou fala será totalmente ignorado pelo App. O comando é então enviado ao Arduino da mesma forma que os botões [Figura 34].

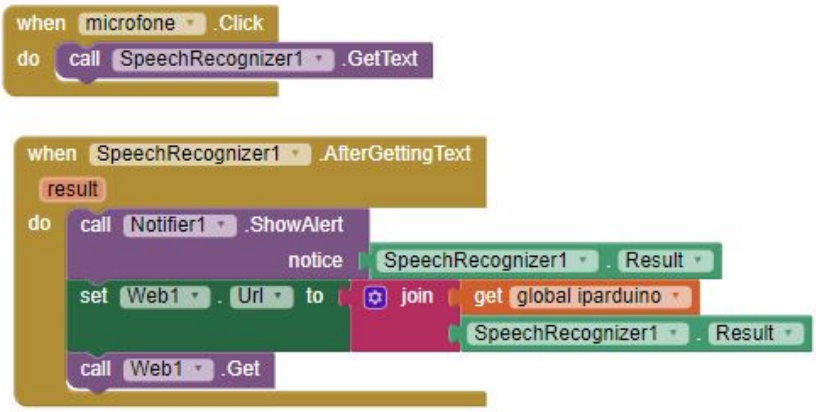

Figura 34: Enviando Comando de Voz

O sexto bloco é responsável por enviar o comando de ligar e desligar o ventilador. Ao apertar no botão o aplicativo envia para o acessa o endereço abaixo fazendo com que o ventilador seja ligado ou desligado [Figura 35].

"http://cmunizsoluitons.ddns.net/ventilador"

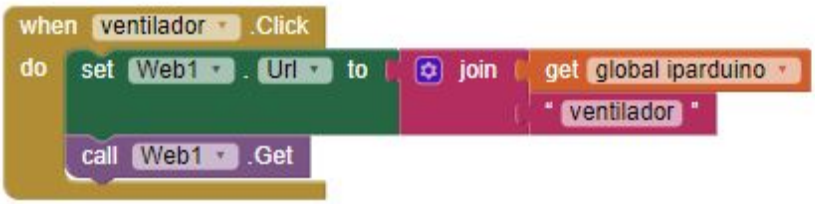

Figura 35: Enviando Comando de Ligar ou desligar o ventilador

Para finalizar os botões comando do aplicativo, o oitavo botão tem a função de realizar o desligamento tanto a bomba de incêndio quanto o alarme sonoro físico do sistema interligado ao Arduino, como também é pausado o alarme sonoro do próprio aplicativo [Figura 36].

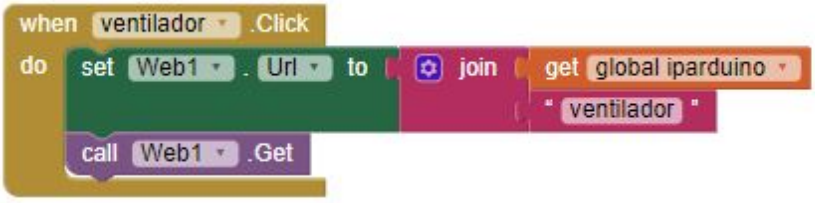

Figura 36: Enviando Comando de desligar Alarme de incêndio

5 Leitura de Status

Sempre que o código envia um comando, realizado a captura de todas as informações da página para atualização do aplicativo referente ao estado real dos dispositivos, a fim de deixar o aplicativo sempre atualizado, foi criado um timer onde a cada 3000 milissegundos [Figura 37] o aplicativo se atualiza sozinho, sem a necessidade de um novo comando. Desta forma é garantido que a temperatura e o sensor de gás sempre estarão atualizados em sua tela.

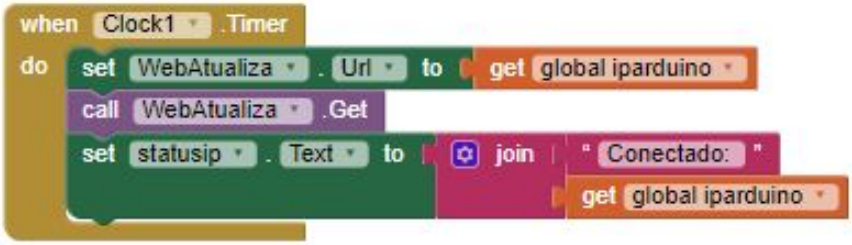

Figura 37: Atualização a cada 3000 milisegundos

Os dois primeiros atributos a serem atualizados são o valor da temperatura e do sensor de gás, a partir da resposta enviada para a página pelo Arduino, é possível utilizar de seguimento de texto do MIT app inventor, a função basicamente coleta x números de caracteres a partir de uma determinada posição. A posição da temperatura para este projeto foi definida como 54, contendo 5 caracteres, "xx.xx" o valor então é acoplado com a unidade Celsius. A medição de Gás inflamável é realizada da mesma maneira, para o projeto a posição do valor do sensor MQ-2 é dada em 61 e possui 3 caracteres já que o Arduino já envia a medida em porcentagem. Caso a medida enviada seja maior que 80 % o alarme sonoro do aplicativo é disparado. O valor é então atualizado na tela do aplicativo, no quinto e nono bloco, ou Label 12 e 13 respectivamente [Figura 38 e 39].

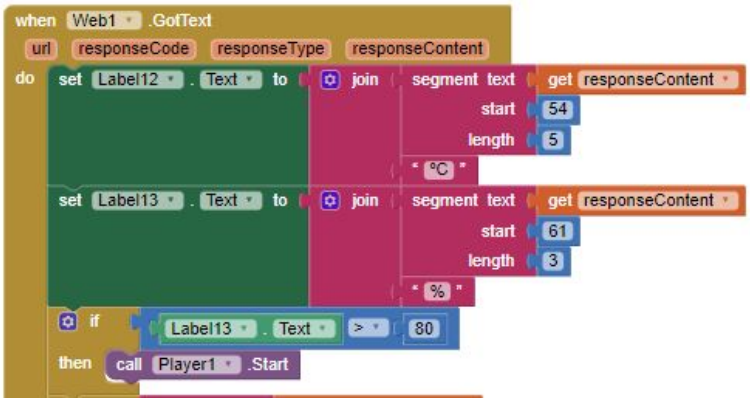

Figura 38: Leitura de temperatura e sensor de gás

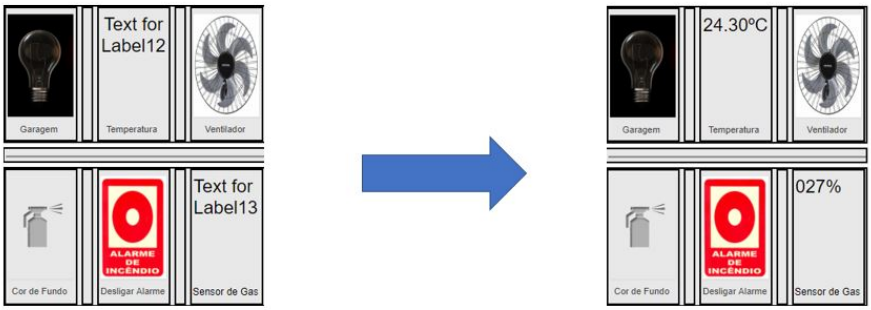

Figura 39: Valor de temperatura e sensor de gás no aplicativo

Os próximos itens a serem alterados são as lâmpadas da sala e do quarto, utilizando a tabela 04. Ao realizar a troca do status de cada lâmpada a imagem do aplicativo é alterada, a fim de informar, de modo visual, ao usuário que as lâmpadas estão acessas ou ligadas [Figura 40 e 41].

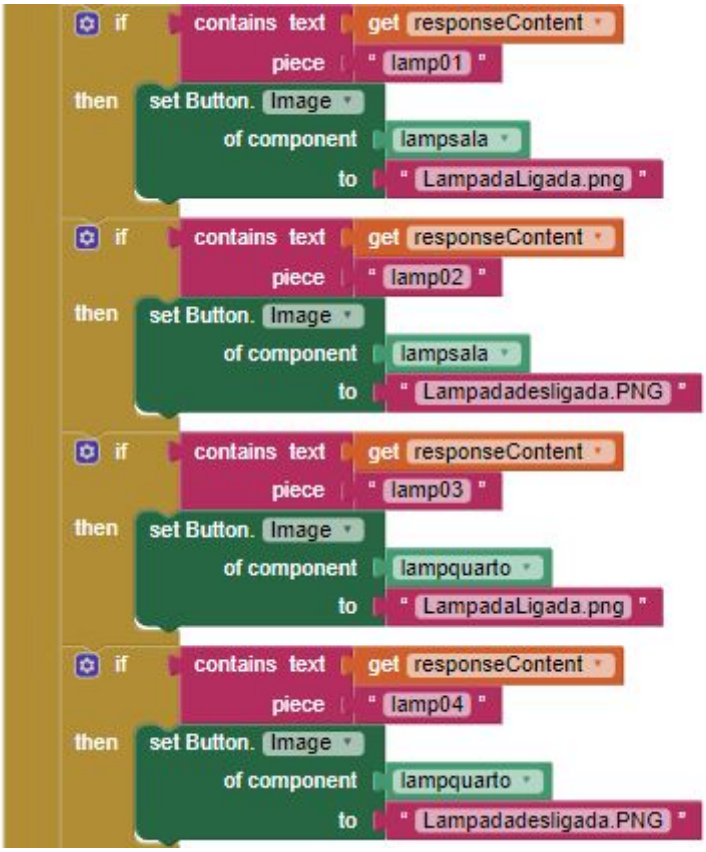

Figura 40: Leitura do status da lâmpada da sala e do quarto

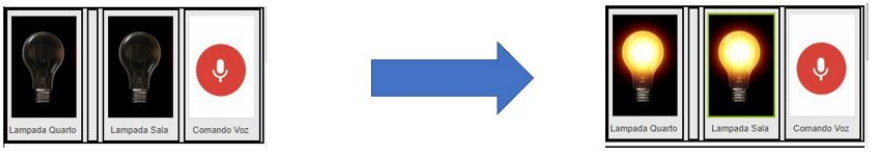

Figura 41: Lâmpada da sala e do quarto no aplicativo

Por fim os últimos dois atributos as serem alterados são referentes a lâmpada da garagem, que vai ser acessa quando o sensor de presença for disparado, portanto a partir do aplicativo é possível saber se há alguém ou pelo menos algo se mexendo dentro da garagem, dando uma maior segurança a casa. Os valores para leitura também são atualizados de acordo com a Tabela 04. Caso não seja possível ler nenhuma resposta do servidor uma mensagem é apresentada na tela do usuário "Erro na conexão com o Servidor", indicando que o sistema necessita de uma manutenção [Figura 42 e 43].

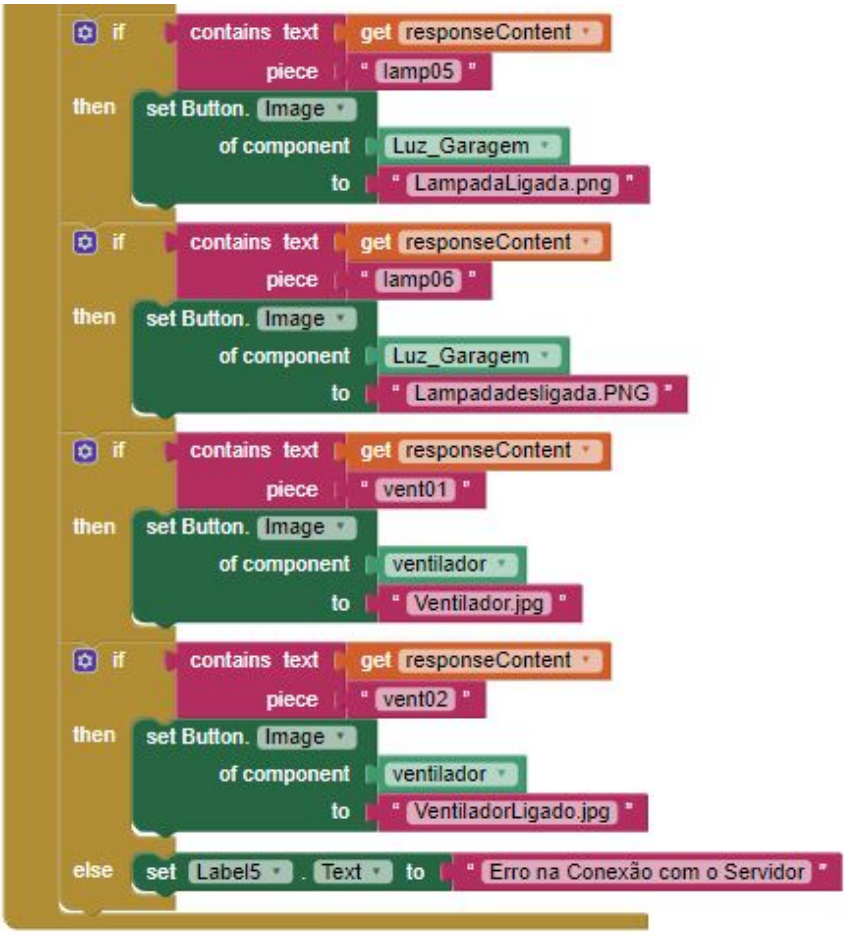

Figura 42: Leitura do Status lâmpada da garagem e Ventilador

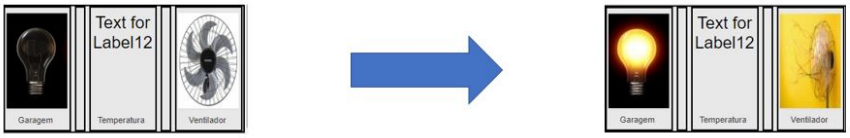

Figura 43: Lâmpada da garagem e Ventilador no aplicativo

6 Personalização

O ultimo recurso do aplicativo desenvolvido para este projeto é referente a personalização da cor da tela. Utilizando o botão cor de fundo do aplicativo é possivel trocar a cor da tela para preto, branco, cinza, azul e rosa. Foram escolhidas apenas algumas cores para demostração, porém durante um implematação é possivel inserir diversas cores para personalização [Figura 44].

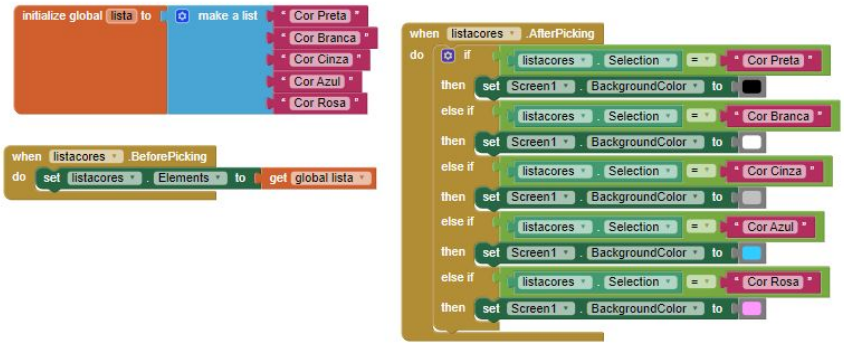

Figura 44: Personalização

4 Conclusão

a Resultados

O sistema se apresentou de forma adequada quando a se controlar uma casa de modo remoto a partir de um aplicativo desenvolvido para Android. Todo o sistema teve baixo custo de desenvolvimento. Junto com o aplicativo o sistema possui fácil manutenção e muita disponibilidade para alteração, sendo possível incluir novos sensores e atuadores de acordo com a capacidade de entradas e saídas do Arduino.

b Melhorias Futuras

Alguns sensores e atuadores possuem baixo desenvolvimento para diminuir o custo do projeto, portanto ao se investir em sensores e atuadores de maior qualidade é possível obter um sistema mais robusto e aplicável a qualquer ambiente.

A placa Arduino UNO não possui muitas entradas e saídas e pode ser trocada por um Arduino MEGA, caso a casa a ser controlada possua muito dispositivos.

c Orçamento

Como o proposito do sistema é ser algo de baixo custo e de facil acesso para qualquer pessoa, veja abaixo um orçamento previo para compra e instalação do sistema em uma casa do tamanho considerado para este projeto.

Peça/Serviço	**Custo (BRL)**
Arduino UNO	85,00
Ethernet Shield	98,90
Shield relé 4 Canais	49,50
Shield relé 1 Canal	12,90
LM35	10,90
MQ-2	15,00
Buzzer passivo	1,50
PIR - HC-SR501	11,50
Fios e Acessórios	35,00
Serviço de Instalção	800,00
Total	1.120,20

Tabela 5: Orçamento

A Codigo completo Ardunino IDE

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

byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
byte ip[] = { 192, 168, 1, 22 };

EthernetServer server(80);

String readString;

const int LM35 = 1;
float temperatura = 0;
int ADClido = 0;
int Gas = 0;
int GasLido = 0;
int LimGas = 70;
int lamp_sal = 3;
int lamp_qua = 4;
int lamp_gar = 5;
int bomba =6;
int presenca = 7;
int Ventilador = 8;
int buzz = 9;
int ssal =0;
int squa=0;
int sgar=0;
int sbomb=0;
int sacinoamento=0;
int svent=0;
int salarm=0;

String sal;
String qua;
String gar;
String bomb;
```

```
String acinoamento;
String vent;
String alarm;


void setup(){

 Ethernet.begin(mac, ip);
  server.begin();
  server.begin();
  pinMode(LM35, INPUT);
  pinMode(Gas, INPUT);
  pinMode(Lamp_Sal, OUTPUT);
  pinMode(Lamp_qua, OUTPUT);
  pinMode(Lamp_gar, OUTPUT);
  pinMode(bomba, OUTPUT);
  pinMode(presenca, INPUT);
  pinMode(ventilador, OUTPUT);
  pinMode(buzz, OUTPUT);

  analogReference(INTERNAL);
}

void loop(){
  EthernetClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();

        if (readString.length() < 100) {
          readString += c;
        }

        if (c == '\n') {

          client.println("HTTP/1.1 200 OK");
```

```
client.println("Content-Type: text/html");
client.println();

client.println("<HTML>");
client.println("<BODY>");
client.println("<H1>CM solutions </H1>");
client.println("<hr />");
client.println("<br />");
client.println(temperatura); //printa temp
char buff[3];
sprintf (buff,"%03d",GasLido);
client.println(buff); // printa gas
client.println(sal); // printa estado sala
client.println(qua);// printa estado quarto
client.println(gar);// printa estado garagem
client.println(bomb);// printa estado bomba
client.println(vent);// printa estado ventilador
client.println(alarm);// printa estado alarme

client.println("<a href=\"/lamp01\">Ligar o led</a>"); // teste
client.println("<a href=\"/lamp02\">Desligar o led</a><br />"); //teste

client.println("</BODY>");
client.println("</HTML>");

delay(1);
client.stop();

 ADClido = analogRead(LM35);//le sensor de temp
 temperatura = ADClido * 0.1075268817204301;
 GasLido = analogRead(Gas); //Le sensor de gas
 GasLido = map(GasLido, 0, 1023, 0, 100);
 acionamento = digitalRead presenca);

  if(readString.indexOf("luz%20sala") > 0)
{
  if(ssal==1){
```

```
  digitalWrite(lamp_sal, LOW);//Desliga lampada sala
  sal="lamp01";
  ssal=0;
  }
  else
  {
  digitalWrite(lamp_sal, HIGH);//liga lampada sala
    sal="lamp02";
    ssal=1;
  }
}
    if(readString.indexOf("luz%20quarto") > 0)
{
  if(squa==1){
  digitalWrite(lamp_qua, LOW);//desliga lampada Qauarto
  qua="lamp03";
  squa=0;
  }
  else
  {
  digitalWrite(lamp_qua, HIGH);//liga lampada quarto
    qual="lamp04";
    squal=1;
  }
}


  if(acionamento==0)//desliga lampada garagem
{
  digitalWrite(lamp_gar, LOW);
  gar="lamp05";
  sgar=0;
}

if(acionamento==1)//liga lampada garagem
  {
    digitalWrite(lamp_gar, HIGH);
```

```
      gar="lamp06";
      sgar=1;
    }
    if(readString.indexOf("ventilador") > 0)
  {
    if(svent=1){
    digitalWrite(ventilador, LOW);//desliga ventilador
    vent="vent01";
    sven=0;
    }
    else
    {
       digitalWrite(ventilador, HIGH);//liga Ventilador
      vent="vent02";
      sven=1;
    }
  }

   if (GasLido > LimGas) || readString.indexOf("ventilador") > 0
   {

       digitalWrite(buzz, LOW);//desliga alarme
       digitalWrite(bomba, LOW);//desliga bomba
      bomb = "bomb01"
       alarm = "alarm01"
    }
    else
    {
       digitalWrite(buzz, HIGH);//liga alarme
       digitalWrite(bomba, HIGH);//liga bomba
      bomb = "bomb02"
       alarm = "alarm02"
    }


  readString="";
}
```

```
        }
      }
    }
  }
```

B Codigo MTI App Iventor

Referências

[1] . *Clube do hardware - diversas duvidas sobre moldem e internet*. URL: https://www.clubedohardware.com.br. (acessado: Varios dias).

[2] Nedo Utilidades Domesticas. *Como instalar soquete de Lâmpada*. URL: https://www.youtube.com/watch?v=eXoS5lV5V_0. (acessado: 20/04/2019).

[3] Athos Eletronics. *Relé - O que é e como funciona*. URL: https://athoselectronics.com/rele/. (acessado: 25/04/2019).

[4] . *MIT APP INVENTOR apostila*. URL: http://www.aedmoodle.ufpa.br/pluginfile.php/170097/mod_book/chapter/2317/MITAPP_Inventor_apostila.pdf. (acessado: 20/05/2019).

[5] . *No-Ip*. URL: https://www.noip.com/. (acessado: Varios dias).

[6] Gerard O'Driscoll. *Essential Guide to Home Networking Technologies, The*. Prentice Hall Ptr; 1st edition, 2000.

[7] . *Site Oficial arduino*. URL: https://www.arduino.cc/en/Main/Software. (acessado: 18/04/2019).

[8] . *Site Oficial da MIT App Inventor*. URL: http://ai2.appinventor.mit.edu. (acessado: 13/05/2019).