UNIVERSIDADE DE TAUBATÉ

ÁLVARO RODRIGO GUIMARÃES SILVA

COMANDO E MONITORAMENTO REMOTO ATRAVÉS DE PLACA ARDUINO COM INTEGRAÇÃO EM PLATAFORMA ANDROID

Taubaté – SP

ÁLVARO RODRIGO GUIMARÃES SILVA

COMANDO E MONITORAMENTO REMOTO ATRAVÉS DE PLACA ARDUINO COM INTEGRAÇÃO EM PLATAFORMA ANDROID

Trabalho de Graduação apresentado para obtenção do Título de Bacharel em Engenharia de Controle e Automação do Departamento de Engenharia Mecânica da Universidade de Taubaté.

Orientador: Prof. Dawilmar Guimarães de Araújo

Taubaté – SP
2017

ÁLVARO RODRIGO GUIMARÃES SILVA

COMANDO E MONITORAMENTO REMOTO ATRAVÉS DE PLACA ARDUINO COM INTEGRAÇÃO EM PLATAFORMA ANDROID

Trabalho de Graduação apresentado para obtenção do Título de Bacharel em Engenharia de Controle e Automaçdo Departamento de Engenharia Mecânica da Universidade de Taubaté.

Data: _____

Resultado: _____

BANCA EXAMINADORA

Prof. _____ Universidade de Taubaté

Assinatura _____

Prof. _____ _____

Assinatura _____

Prof. _____ _____

Assinatura _____

Silva, Álvaro R. G. **COMANDO E MONITORAMENTO REMOTO ATRAVÉS DE PLACA ARDUINO COM INTEGRAÇÃO EM PLATAFORMA ANDROID**. 2017. 46 f. Trabalho de Graduação (Bacharel Engenharia de Controle e Automação) – Departamento de Engenharia Mecânica – DEM, Universidade de Taubaté, Taubaté.

RESUMO

As residências terão um novo conceito no que se refere a acionamento e desligamento de dispositivos eletrodomésticos. Não é novidade poder acender a luz, ou preparar a banheira para um banho ainda no caminho de casa, porém com um custo elevadíssimo. O fator custo é o principal motivador deste trabalho, pois com os novos recursos de aquisição de dados de baixo custo disponíveis e com alta compatibilidade com diversos tipos de sensores, também de baixo custo, alidos a equipamentos tão rotineiros como um celular ou tablet, podem trazer uma nova experiência de comando a distância ao usuário sem necessariamente haver grande despêndio de recursos financeiros envolvidos.

Assim este trabalho se propõe a desenvolver uma plataforma genérica de acionamento para quaisquer equipamentos que possam ser ligados em uma placa Arduino, uma plataforma que pode ser instalado em qualquer aparelho que possua o sistema operacional Android e microfone.

Alunos de automação e área afins também podem utilizar da plataforma desenvolvida AnAr para criar seus próprios projetos, pois este projeto se caracteriza exatamente por se tratar de uma plataforma genérica onde o usuário com pequenas alterações em sua base consegue acionar ou monitorar qualquer equipamento que desejar.

Palavras-chave: (Automação; Domótica; Acionamento por Voz; Arduino; Android)

Silva, Álvaro R. G. **REMOTE CONTROL AND MONITORING USING ARDUINO BOARD AND ANDROID PLATFORM INTEGRATION 2017**. 46 p. Graduation work (Bachelor's degree in in Engineering of Control and Automation) – Department of Mechanical Engineering – DEM, University of Taubaté, Taubaté, BRAZIL.

ABSTRACT

The residences will have a new concept regarding turn on and turn off electrical appliances. It is not new to be able to turn on the light, or prepare the bath for a bath still on the way home, but at a very high cost. The cost factor is the main motivator of this work, because with the new low cost data acquisition resources available and with high compatibility with several types of sensors, also low cost, attached to equipment as routine as a cell phone or tablet, can to bring a new experience of remote control to the user and it do not need to be expensive for him.

In this terms this work proposes to develop a generic platform of activation for any equipment that can be connected in an Arduino board, a platform that could be installed in any device with Android operating system and microphone for a voice command.

Students of automation and related areas can also use the platform developed AnAr to create their own new projects, because this project is characterized by the fact that it is a generic platform where the user with small changes can activate or monitor any equipment they want.

Keywords: (Automation; Home Automation; Voice Command; Arduino; Android)

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

O objetivo do presente trabalho é o de comandar e obter dados à distância de quaisquer equipamentos compatíveis que estejam ligados à uma placa Arduino, através de reconhecimento de voz na plataforma Android. Esses equipamentos podem ser dos mais variados tipos: Sensores de iluminação, pressão, distância, altitude, ou até mesmo equipamento de atuação como motores, LED's, buzzers, alarmes, etc. Assim é possível instrumentar um ambiente, ou até mesmo uma linha de produção e tornar o processo de obtenção de dados mais hábil e com uma interface amigável ao usuário. Porém no presente trabalho, será abordado de forma mais profunda a aplicação residencial.

Assiste-se um crescente aumento da utilização deste emprego da tecnologia, que deixa de ser apreciada como futurística e se transforma numa importante ferramenta ao público, possibilitando o controle remoto de diversos equipamentos domésticos bem como o monitoramento de sensores. A economia percebe essa transição dos sistemas comuns de residências para este novo modelo (LIMA, NOBRE, de ALENCAR,2016) que tem prospecções positivas para os próximos anos. Pesquisadores como Teruel e Noveli Filo (2008) fizeram um levantamento sobre quais sistemas residenciais a população acreditava ter maior importância para uma aplicação de automação residencial e este levantamento apontou que a maior parte deseja a segurança, a citar: circuitos fechados de TV, detecção de gases, alarme de incêndio e detectores de presença, são exemplos mais comuns de aplicações em segurança que podem ser automatizadas e monitoradas remotamente.

Os sistemas de automação hoje existentes, incluindo os capazes de fazer acionamentos através de comandos de voz, estão cada vez mais presentes no nosso dia a dia, por exemplo nos carros é possível, através da fala, escolher uma música ou pedir ao GPS que mostre o caminho para algum endereço.

Trazer ao usuário uma experiência de abrir o portão ao chegar em casa, ao ligar a piscina e acender as luzes da área externa da sua casa apenas com um comando de voz pelo celular, Tablet ou relógio, compatibilizando a plataforma Android e Arduino através de linguagem HTML, que é a linguagem padrão de comunicação dos navegadores de internet. É possível também, com poucas adaptações, verificar travas de janelas ou portas depois de já ter saído de casa.

Já num ambiente mais exigente é possível instrumentar um laboratório para verificar à distância se a temperatura está dentro de determinado limite e ainda acionar atuadores que possam equilibrar essa temperatura. Isso para citar um exemplo óbvio.

A placa Arduino é uma placa de aquisição de dados de baixo custo e baixo consumo elétrico e possui inúmeras unidades de expansão, bem como versões mais potentes equipados com processadores Intel por exemplo. Por este motivo este hardware foi escolhido para este projeto. A plataforma Android foi escolhida por ser de livre desenvolvimento e contar com recursos Google como o reconhecimento de voz.

A maioria das referências utilizadas para este trabalho são disponibilizadas pelos fabricantes e disponibilizadas em domínios Online.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O presente trabalho irá integrar duas diferentes plataformas – Arduino e Android – para que haja uma interação utilizando recursos próprios de cada uma e assim tornar possível uma nova experiência de acionamento remoto de baixo custo.

Um aplicativo que deve ser criado para a plataforma Android será responsável por enviar os comandos para a rede, para um IP inserido pelo usuário, que deve corresponder ao web-server Arduino, que por sua vez, será responsável por receber e acionar o requisitado pelo usuário e retornar ao web-client uma resposta.

Além disso a Arduino irá armazenar todos os comandos executados com sucesso em um cartão de memória gerando assim um log para consulta posterior no formato *.txt contendo data e hora.

1.1.2 Objetivos Específicos

Desenvolver uma plataforma combinatória Android e Arduino capaz de reconhecer a fala e enviar dados através de uma rede para acionamento, desligamento ou monitoramento de dispositivo compatível em residência ou outro

ambiente previamente preparado, com a possibilidade de geração de log com data e hora que ficam salvos em cartão de memória para posterior consulta.

1.2 DELIMITAÇÃO DO ESTUDO

O trabalho se limita ao desenvolvimento de uma plataforma genérica para acionamento, desligamento e monitoramento e também da prova de seu funcionamento. Não serão empregados equipamentos de grande porte para demonstração, mas sim aqueles que julgados suficientes para provar a funcionabilidade do projeto.

Serão gerados log de atividade contendo data e hora, porém não será possível identificar qual usuário ou aparelho o responsável pelo comando executado. Julgou-se que em uma primeira instância essa funcionalidade não seja necessária ao projeto, ainda que posteriormente possa ser implementada sem maiores dificuldades, uma vez que a plataforma genérica está pronta.

O projeto não provê usuário prioritário, assim, se dois ou mais usuários acionarem o mesmo equipamento simultaneamente, o web-server respeitará apenas a ordem cronológica de execução, ou seja, o primeiro comando a ser recebido é o primeiro a ser executado.

O trabalho será totalmente aberto, ou seja, na sessão **RESULTADOS** estará disponível link para download de todos os código utilizados. Por se tratar de um projeto open-source não se levou em consideração o desenvolvimento em abiente de programação fechada, por este motivo o iPhone não terá um código equivalente. Além disso o aplicativo conta com um serviço Google de reconhecimento de voz em tempo real não disponível para iPhone.

1.3 RELEVÂNCIA DO ESTUDO

O projeto é de extrema importância no campo da automação pois abre as portas para que novas ideias possam ser implementadas tendo como base a plataforma genérica que fora desenvolvida, assim poupando o despêndio de tempo e de dinheiro, uma vez que os equipamentos envolvidos são de baixo custo: Arduino, Ethernet Shield, Real Time Clock, Cartão de Memória. E os demais são de uso rotineiro como celular e roteador.

1.4 ORGANIZAÇÃO DO TRABALHO

Aqui se dedica um ou mais parágrafos para explicar, de maneira muito resumida, o conteúdo de cada capítulo desenvolvido no trabalho.

2 REVISÃO DA LITERATURA

A Automação Residencial é uma prática que almeja nas residências a praticidade de sistemas de tecnologias presentes do dia a dia proporcionando ao usuário a segurança, praticidade, conforto e atualmente o entretenimento (BOLZANI, 2004).

Automação Residencial, pode ser referenciada também como Domótica (do latim, Domus fundido com Robótica) ou Casa do Futuro. A possibilidade de monitorar sensores, bem como de atuar sob equipamentos através de comandos remotos, contendo barramentos e interfaces, definem o que é a Automação Residencial (Dodonov, Accardi 2012).

O potencial de aplicação de controles residenciais dá-se em diversas áreas, estando discretizadas a seguir as principais: Irrigação de jardins, Controle de Acesso, Controle de Iluminação, Controle de Sistemas de climatização, Sistemas de detecção de alarme de incêndio (Jacques, 2015).

O ponto crucial da Automação é que ela depende de sensores, atuadores e um controlador (Dodonov, Accardi 2012).

Figura 1 - Sistema simplificado de Automação

Fonte: próprio autor

A FIGURA 1 ilustra um sistema onde há um Controlador Central, responsável por adquirir os dados e por enviar comandos aos atuadores.

Uma placa de aquisição de dados pode ser utilizada como Controlador Central. “Arduino é uma plataforma eletrônica de código-aberto”, segundo o próprio fabricante. A placa de entrada da Arduino é a Arduino UNO, com Microcontrolador de fabricação Atmel ATmega328P (domínio Arduino, https://www.arduino.cc/en/Main/ArduinoBoardUno). Mas para um projeto de maior complexidade é comum a utilização da ArduinoMega, com Microcontrolador de fabricação Atmel ATmega2560. Esta placa é capaz de trabalhar com 54

entradas/saídas digitais, ou seja, é possível trabalhar com 27 sensores e 27 atuadores digitais utilizando exclusivamente esta placa. E 16 entradas analógicas para sensores analógicos (domínio Arduino, https://www.arduino.cc/en/Main/ArduinoBoardMega2560). A programação é feita através de IDE do fabricante em ambiente Windows ou Linux. Baseada em Linguagem C, porém trata-se de uma linguagem própria, onde o desenvolvedor necessita constantemente buscar referências do domínio https://www.arduino.cc/en/Reference/HomePage.

Indo mais adiante a respeito do tema é possível inserir no sistema de Automação Residencial o Reconhecimento de Voz, que aprimora ainda mais o sistema. Aempresa Google, é líder de uma aliança que desenvolve uma plataforma de reconhecimento de voz chamado “Speech Recognizer” que segundo a própria empresa, é um ‘Potente modelo de rede neural, capaz de reconhecer mais de 80 linguagens e variações’ (traduzido do site Google Cloud Platform). O mais interessante é que a Google disponibiliza aos desenvolvedores que utilizam sua plataforma Android (Sistema de código aberto desenvolvido originalmente para Smartphones e Tablets e que hoje se ampliou à televisões, relógios, até mesmo carros e aparelhos de rádio), a possibilidade de criar aplicativos para a plataforma utilizando o recurso de fala.

Figura 2. Camadas da plataforma Android

Fonte: Obtido do domínio Android Open Source Project, AOSP, https://source.android.com

Além da existênciada possibilidade de desenvolvimento de aplicativo com reconhecimento de voz, o desenvolvedor dispõe de recursos para transmitir a fala em forma de texto para outro aplicativo, por email, para a internet, ou quaisquer local desejado – Todas essas possibilidades são alavancadas pela própria desenvolvedora da plataforma Android que mantém a disposição para consulta pública seu domínio exclusivo para desenvolvedores https://developer.android.com, onde literalmente ensina como utilizar recursos como o Speech Recognizer dentro de um aplicativo de criação própria.

Basicamente a programação na plataforma se dá em JAVA e em XML. O JAVA uma linguagem operacional que originalmente foi lançada pela Sun Microsystem em 1995. Já o XML se trata de uma linguagem criada para satisfazer necessidades da larga escala de publicações eletrônicas, porém hoje pode ser utilizada em qualquer aplicação que exige características especiais, pertencendo ao grupo W3C (domínio, https://www.w3.org).

3 METODOLOGIA

Para facilitar a compreensão do trabalho, esta sessão será dividida em duas partes: Programação Arduino e Programação Android.

A criação deste projeto exigiu que diferentes componentes fossem acionados, sendo eles:

-ArduinoMega;

-Ethernet Shield para comunicação com a rede ;

-Roteador;

-Navegador WEB conectado a mesma rede – Ou um aplicativo Android, que foi desenvolvido para este fim e possui a possibilidade de acionamento por voz;

-Cartão de memória;

-Motor, LED's, termômetro;

-Relógio digital.

O desenvolvimento iniciou-se utilizando os componentes em separado e implementados para que mais tarde viessem a trabalhar como um conjunto.

3.1 Programação para Arduino

Para a implementação do acionamento à distância, num primeiro momento deve-se utilizar uma rede interna e, ainda sem conexão com a internet apenas com a intranet, estabelecer um método de comunicação.

Como a placa Arduino Mega, que é a opção escolhida para este projeto, não possui por si só uma maneira de se conectar à internet, é preciso utilizar então uma placa de expansão – Shield – que adiciona este tipo de funcionalidade na placa. A Shield utilizada é a Ethernet Shield W5100, atualmente duas versões dessa placa, sendo que a Ethernet Shield 2 é pouco encontrada no Brasil. Esta placa de expansão caracteriza-se por possuir o controlador WizNet 5100, da empresa WizNet, este controlador é o responsável pela comunicação TCP/IP, sendo possível a criação de um MAC Adress e outras funções do tipo *IoT.*

A Shield ainda conta com um slot para leitura e escrita no cartão de memória, onde serão gravados os LOG's de acesso gerados. Para a escrita no cartão a Shield utiliza o pino 4 da arduino, ou seja, não haverá mais a disponibilidade deste pino. Os

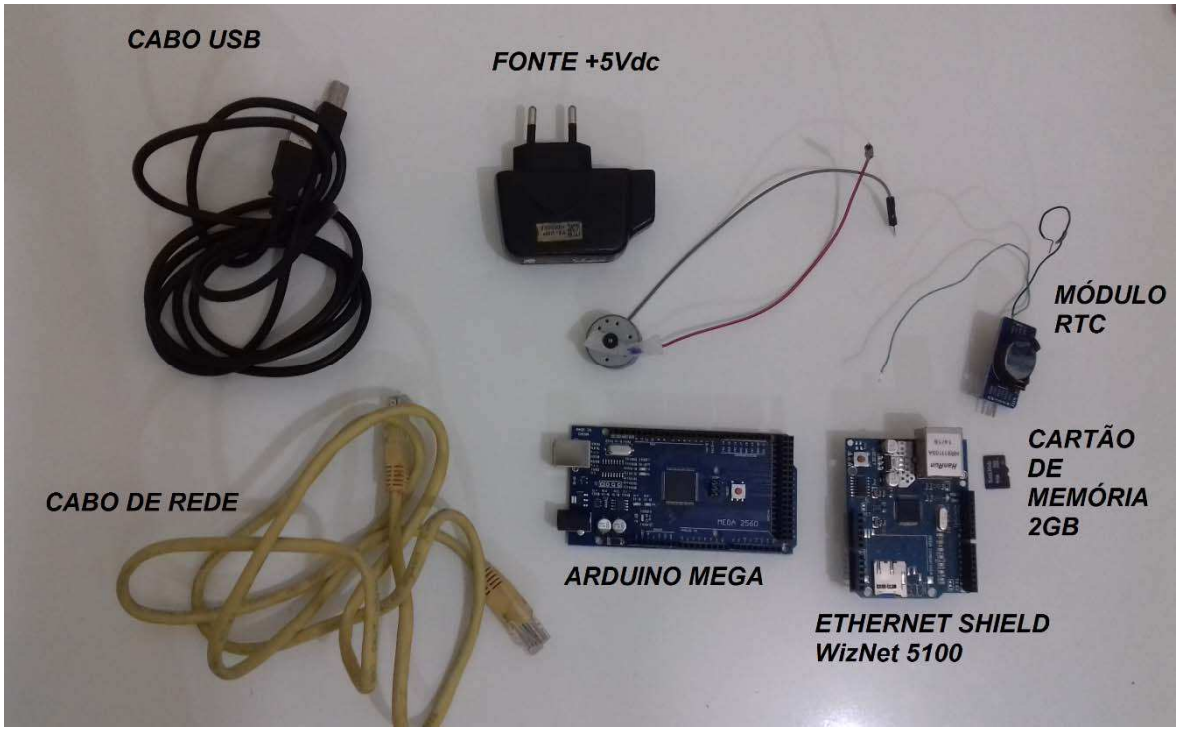
LOG's possuirão data e hora, para isso, será necessária outra Shield para que a função de data e hora seja implementada na placa Arduino.

Existem muitos módulos que trazem a função de RTC – Real Time Clock sendo o mais popular deles é o DS3231, que possui um cristal oscilador e pode ser alimentado por uma bateria de +3.3Vdc. Este é o módulo que será usado no projeto.

Na FIGURA 3 é possível observar todos os componentes utilizados no projeto, adicionalmente, usa-se um celular com Android, ou qualquer dispositivo com navegador de internet, computador, relógios do tipo smart, etc.

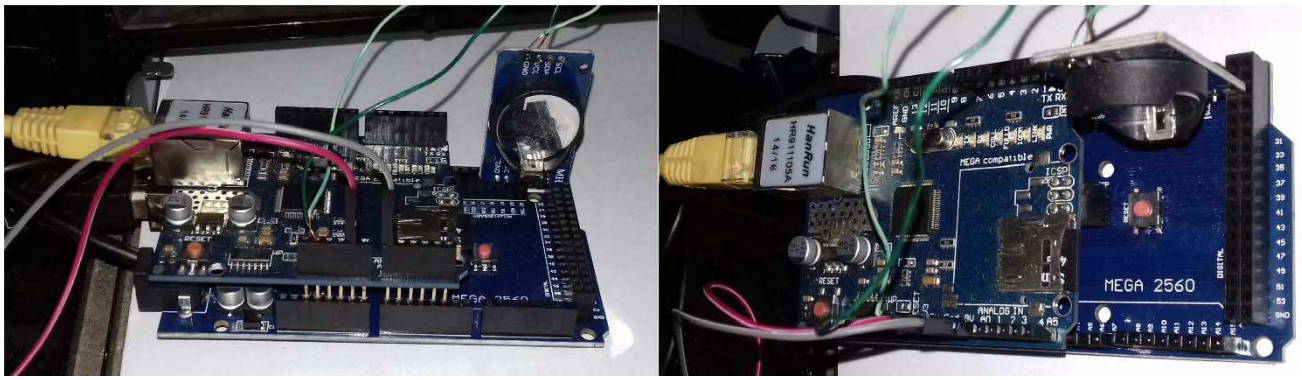
A FIGURA 4 mostra os componentes montados, ArduinoMega na base, e encaixado sobre ela a Ethernet Shield, o módulo RTC à direita na imagem.

Figura 3 – COMPONENTES UTILIZADOS NO PROJETO



Fonte: próprio autor

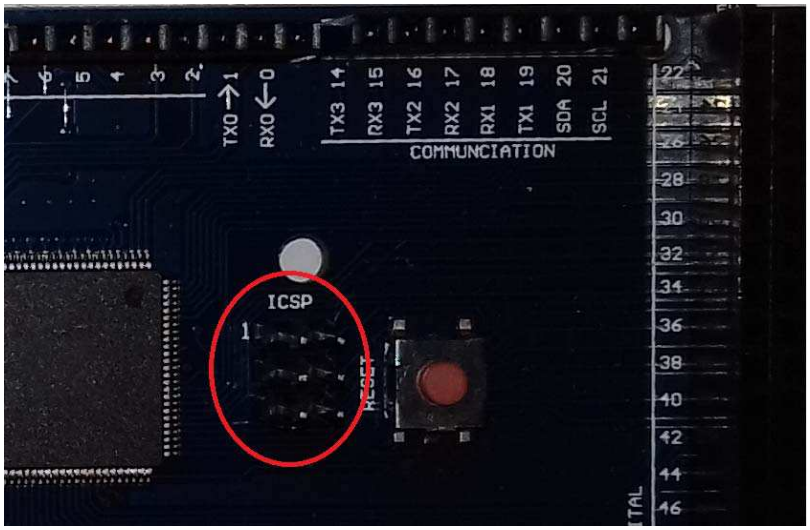
Figura 4 – MONTAGEM DOS COMPONENTES



Fonte: próprio autor

A correta montagem dos componentes exige alguns cuidados pois alguns pinos devem ser conectados especificamente ao seu correspondente na ArduinoMega. Representados na FIGURA 5, uma foto da ArduinoMega, onde são identificados os pinos que devem se encaixar perfeitamente com os mesmos pinos da placa EternetShield ICSP.

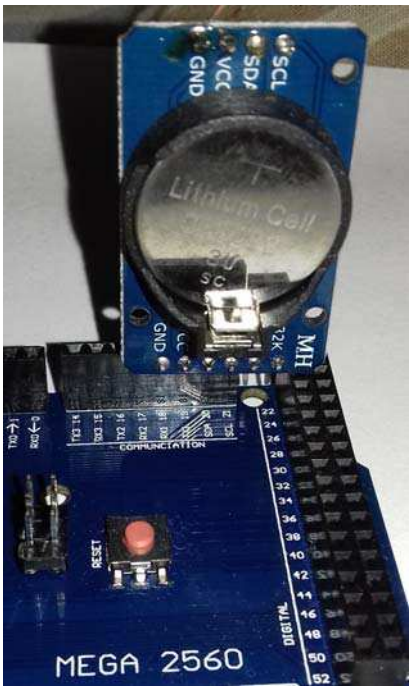
Figura 5 – Pinos ICSP para correta conexão Ethernet Shield



Fonte: próprio autor

O RTC por sua vez trabalha com comunicação I2C. A ArduinoMega possui pinos próprios para módulos que utilizam essa linguagem, os quais podem ser identificados na FIGURA 6.

Figura 6 – Detalhe da conexão módulo RTC aos pinos SDA e SCL



Fonte: próprio autor

Após terem sido adequadamente conectadas as partes, é hora de se iniciar a programação, que por sua vez exige que um algoritmo do funcionamento seja criado. O Algoritmo abaixo, FIGURA 7 mostra a estrutura básica de funcionamento, um detalhe importante aqui é que este algoritmo não se refere a programação da aplicação para Android, o qual será mostrado posteriormente.

Figura 7 -Algoritmo de execução – Lado da programação Arduino

1. Definição de MAC e IP;
2. Definição da ethernet shield como Web Server;
3. Início da estrutura de repetição;
4. Verificação do existência de um Web Client;
5. Caso exista cliente verifica-se se existe alguma requisição no servidor;
6. Determinação da estrutura que irá atender a requisição do cliente, sendo possíveis as seguintes:

a. Acionamento do motor;
b. Desligamento do motor;
c. Visualização de LOG gerados no cartão de memória;
d. Limpeza do cartão de memória;
e. Verificação da temperatura ambiente.

7. Geração de LOG no cartão de memória, incluindo data, hora e requisição do cliente;
8. Acionamento da função requisitada e resposta ao cliente;
9. Fim da execução;
10. Volta passo 3.

Fonte: próprio autor

Foram necessárias mais de dez versões até se chegar um programa totalmente funcional e no início foram programados em separado seguindo a ordem a seguir:

Acionamento do motor, sem nenhuma Shield conectada;

Conexão da ethernet shield como web server e testes com a placa conectada na rede.

Neste ponto foi necessário criar um método de reconhecimento do comando recebido. O Web Client (navegador de internet por exemplo) envia o GET em linguagem HTML como request num endereço IP específico. No projeto o endereço IP da placa é o seguinte: 192.168.0.129.

Então cada função foi definida para os endereços conforme FIGURA 8:

Figura 8 – Endereços programados para requisição de Web-Client

a. Acionamento do motor: 192.168.0.129/14
b. Desligamento do motor: 192.168.0.129/13
c. Visualização de LOG gerados no cartão de memória: 192.168.0.129/15
d. Limpeza do cartão de memória: 192.168.0.129/11
e. Verificação da temperatura ambiente: 192.168.0.129/12

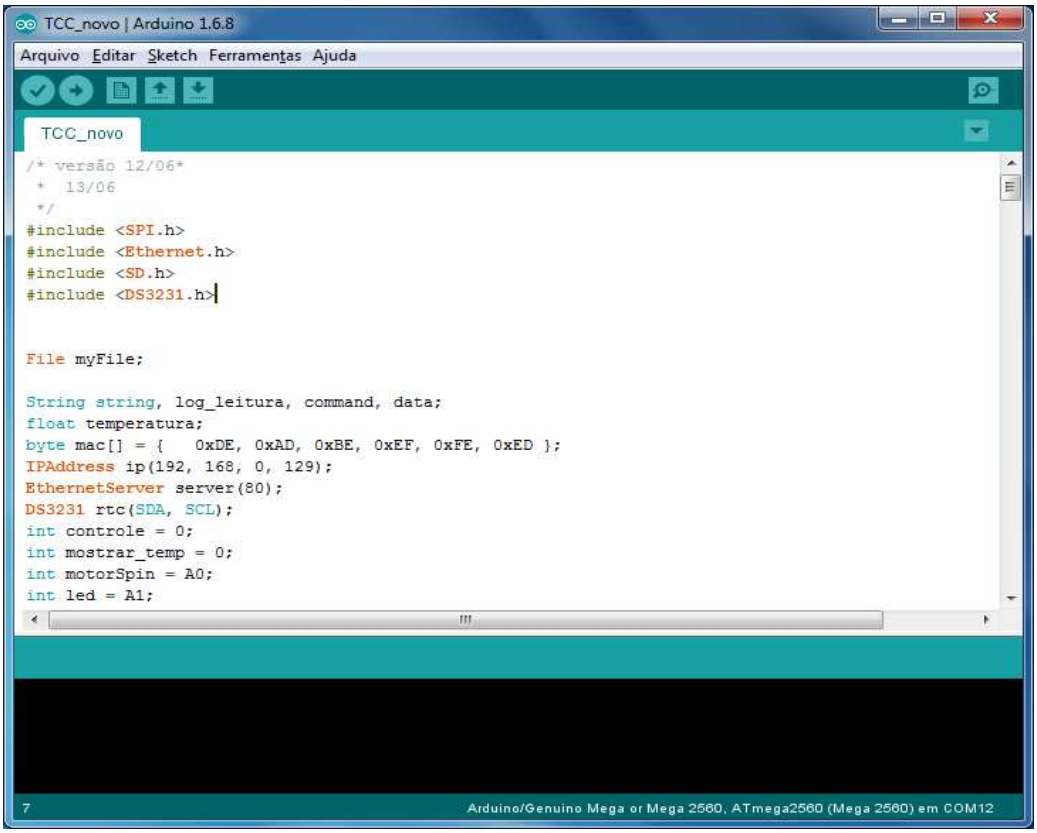
Fonte: próprio autor

Ou seja, qualquer computador na Intranet que acessar esses endereços, através de um visualizador HTML receberá retorno do servidor, com mensagem de falha ou sucesso e acionará a função requisitada.

3.1.1 Bibliotecas pré-existentes utilizadas

A programação é de autoria própria excetuando-se a comunicação Ethernet que foi retirada do próprio exemplo existente na IDE da Arduino apenas tendo sido removido o 'refresh' do trecho HTML de comunicação com o navegador. A biblioteca de comunicação SPI.h é fundamental para a utilização da Shield, onde os pinos 11, 12 e 13 estão conectados ao ICSP da placa Arduino. Para o caso da utilização do cartão de memória, a biblioteca SD.h deve ser inserida, a biblioteca é ajustada para utilizar o pino 4 como pino de gravação no cartão. O Real Time Clock possui uma biblioteca própria, o arquivo DS3231.h, o ajuste de data e hora, uma vez que fundamentado nesta biblioteca, não está em português, mas em inglês. O autor entende que não faz nenhum sentido recriar programas para substituir as bibliotecas uma vez que as útlimas funcionam perfeitamente para o propósito do projeto.

Figura 9 – IDE Arduino para ambiente Windows



Fonte: próprio autor

3.1.2 Estrutura HTML

O trecho do código mostrado na FIGURA 10 é utilizado todas as vezes que uma função é acionada. A função client.println irá ecrever no cliente as linhas e a sequência mostrada é exigida para estabelecer comunicação entre Web Client e Web Server.

Figura 10 – Trecho de escrita no WEB-server para o WEB-Client em HTML através da placa Ethernet

```
1.    void funcao14 (){
2.    EthernetClient client =server.available();
3.    while(client.connected()){
4.    if(client.available()){
5.    if(client){
6.    client.println("HTTP/1.1 200 OK");
7.    client.println("Content-Type: text/html");
8.    client.println("Connection: close");
9.    client.println();

10.   client.println(command);

11.   client.println();
12.   delay(10);//10 ms para escrita no cliente
13.   client.stop();
```

Fonte: próprio autor

Este foi um dos motivos pelo qual a utilização da Arduno UNO se tornou inviável. Como estas são rub-rotinas, valeu-se de cansáveis tentativas de a partir de uma sub-rotina chamar uma outra sub-rotina contento os parâmetros de escrita HTML, para que não fosse necessário repetir essas linhas sempre que uma nova função fosse executada, porém sem sucesso.

Então, como o texto tem que ser repetido para todas as funções, para uma perfeita execução da comunicação, a memória programável (SRAM, site do fabricante) da Arduino UNO que é de 2KB tornou-se insuficiente para armazenar todos os dados.

Por este e outros motivos que serão adiante explanados, optou-se pela utilização da ArduinoMega, que possui uma área programável de 8KB, ou seja quadriplica a extensão de memória que pode ser usada pelo desenvolvedor.

No programa Arduino, existem algumas strings, sendo cada uma responsável por uma função ao longo da execução do programa. A principal delas é usada na

comparação dos endereços requisitados pelo web-client o qual irá acionar a função desejado, analisar o código abaixo permite a melhor compreensão desta.

Notar que a string “command” mais tarde será usada para responder a solicitação do web-client, mostrando ao usuário no lado do aplicativo qual foi a atitude tomada pela placa Arduino e como ela interpretou seu comando.

A FIGURA 11 mostra a rotina de detecção elaborada para identificar qual função dentro do programa será executada de acordo com a requisição recebida do usuário do celular. Os endereços estão de acordo com os indicados na FIGURA 8 e o endereço completo fica salvo na string chamada string.

Figura 11 – Rotina de detecção de comando recebido

```
if(client){
booleancurrentLineIsBlank= true;
while(client.connected()){
if(client.available()){
char c =client.read();
string.concat(c);
if(string.endsWith("15")){//se o endereço termina com /15 irá chamar a
função de leitura do cartão
command="LOG";
leituraSD();
}
if(string.endsWith("/14")){
command="MOTOR ON";//se o endereço termina com /14  será camada a
função responsável pelo acionamento do motor
funcao14();
}
if(string.endsWith("/13")){
command="Stop MOTOR";
// Serial.println("Recebido comando  no Internet Shield");
funcao13();
}
if(string.endsWith("/12")){
command="Temperature";
// Serial.println("Recebido comando  no Internet Shield");
funcao12();

}
if(string.endsWith("/11")){
command="Deletado";
// Serial.println("Recebido comando  no Internet Shield");
funcao11();
   }
```

Fonte: próprio autor

3.2 Programação Android

Esta é segunda parte do projeto, de um lado a programação Arduino e Linguagem C de outro lado a programação em para Android em linguagem JAVA e XML.

A experiência de desenvolver para Android com fim de utilizar das ferramentas disponíveis pela empresa Google de reconhecimento de voz (Speech Recognizer) trouxe à tona o aplicativo nomeado AnAr, que quer dizer Android + Arduino.

O algoritmo gerado no processo se encontra na FIGURA 12.

Figura 12 – Algoritmo resumido do aplicativo Android

1. Verificar disponilidade de conexão com internet, caso contrário, irá retornar "Sem conexão";
2. Verificar disponibilidade de conexão com o endereço inserido pelo usuário no campo adequado na ausência de resposta irá retornar "Checar Servidor";
3. Aguardar comando de voz ou de botão do usuário;
4. Se o usuário acionar o botão de reconhecimento de voz, analisar e retornar uma string ao programa principal equivalente a fala;
5. Enviar uma REQUEST ao servidor no endereço a que cabe o comando;
6. Reproduzir na tela a resposta enviada pelo servidor.

Fonte: próprio autor

As FIGURAS 13, 14 e 15 são do aplicativo criado, que será explicado nesta seção.

Figura 13 – Aplicativo Android AnAr instalado no celular, próximo ao ícone WhatsApp

Fonte: próprio autor

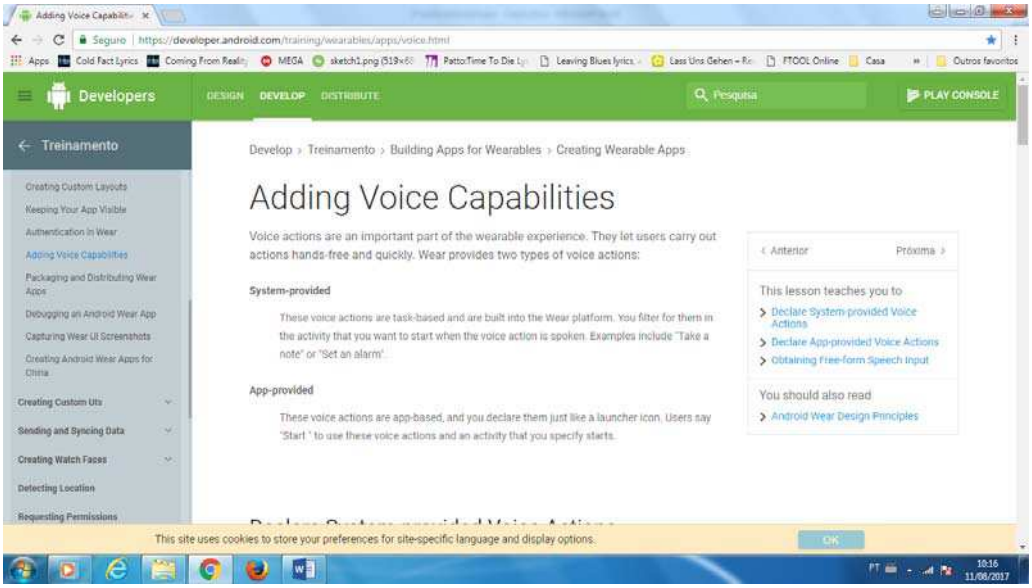
Figura 14 – APP AnAr sendo executado com comando SpeechRecognizer acionado

Fonte: próprio autor

Figura 15 – APP An Ar sendo executado com retorno do web server de Stop MOTOR

Fonte: próprio autor

Figura 16 – Página do desenvolvedor Android – Na seção Creating Wearable Apps



Fonte: próprio autor

3.2.1 TRECHO DO CÓDIGO

O código abaixo foi retirado da página do desenvolvedor Android (FIGURA 16) e é responsável por permitir ao aplicativo obter uma forma livre de reconhecimento, ou seja, irá se conectar com o servidor Google responsável pelo reconhecimento e reproduzirá a fala através de uma string dentro da aplicação em JAVA.

Figura 17 – Código responsável pela detecção da fala no Android

```
privatestaticfinalint SPEECH_REQUEST_CODE =0;

// Create an intent that can start the Speech Recognizer activity
privatevoiddisplaySpeechRecognizer(){
    Intent intent=newIntent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
// Start the activity, the intent will be populated with the speech
text
startActivityForResult(intent, SPEECH_REQUEST_CODE);
}

//Esta função é chamada quando o Speech Recognizer retorna um valor.
// Aqui é onde o texto resultado é processado
@Override
protectedvoidonActivityResult(intrequestCode,intresultCode,
        Intent data){
if(requestCode== SPEECH_REQUEST_CODE &&resultCode== RESULT_OK){
        List<String> results =data.getStringArrayListExtra(
RecognizerIntent.EXTRA_RESULTS);
        String spokenText=results.get(0);
// tomar alguma ação spokenText
}
super.onActivityResult(requestCode,resultCode, data);
}
```

Fonte: próprio autor

A plataforma utilizada para o desenvolvimento foi o Android Studio, da empresa IntelliJ, o código completo contém poucas linhas de programação e vem sendo desenvolvida cerca de dois anos, porém, na primeira versão não se pensava em utilizar comando de voz, mas apenas botões que acessariam endereços específicos no servidor. Este serviu de base para a implementação do comando de voz, uma vez que as sub-rotinas acionadas por reconhecimento de voz, são exatamente as mesmas acionadas pelos botões.

Assim o aplicativo pode comandar a Arduino tanto por reconhecimento de voz como por botões existentes no aplicativo.

No código abaixo é possível observar as adaptações realizadas para que cada comando de voz seja responsável por chamar uma sub-rotina correspondente:

Figura 18 – Tratamento do retorno do spokenText para envio ao Web Server

```
protectedvoidonActivityResult(intrequestCode,intresultCode,
                                    Intent data){
if(requestCode== SPEECH_REQUEST_CODE &&resultCode== RESULT_OK){
            List<String> results =data.getStringArrayListExtra(
RecognizerIntent.EXTRA_RESULTS);
            String spokenText=results.get(0);

if((spokenText.equals("mostrar log")||(spokenText.equals("mostrar dados do
cartão")))){
cont=("/15");
bt1m();
}
if(spokenText.equals("Ligar motor")){
cont=("/14");
bt1m();
}
if((spokenText.equals("Desligar motor")||(spokenText.equals("desligar
motor")))){
cont=("/13");
bt1m();
}
if((spokenText.equals("Qual a temperatura")||(spokenText.equals("qual é a
temperatura")))){
cont=("/12");
bt1m();
}
if((spokenText.equals("Deletar log")||(spokenText.equals("deletar log")))){
cont=("/11");
bt1m();
}
```

Fonte: próprio autor

É possível perceber que apenas as seguintes falas, entre aspas, serão reconhecidas:

-“Mostrar dados no cartão” – Irá exibir no aplicativo, os logs gerados ao longo da execução até aquele dado momento.

-“Ligar motor “– Irá enviar ao servidor o comando para acionar o motor ligado à arduino;

-“Desligar motor” - Irá enviar ao servidor o comando para delisgar o motor ligado à arduino;

-“Qual a temperatura?” – Essa fala irá fazer com que a Arduino leia os dados do sensor de temperatura e retorne o valor ao aplicativo;

-“Deletar log” - Essa fala irá fazer com que a Arduino apague todos os dados do cartão de memória, deixando o LOG sem nenhum registro.

O aplicativo foi desenvolvido em API Level 22, à grosso modo quer dizer que para funcionar requer um aparelho com a versão Lollipop do Android

Figura 19 – Tela inicial da plataforma de desenvolvimento Android Studio

Fonte: próprio autor

3.2.2 Driver de Desenvolvedor do fabricante do celular

Cada fabricante de celular possui um driver de desenvolvedor para que seja possível ao usuário desenvolver aplicativos android e instalá-los em seu celular a partir das IDE's, como a supracitada Android Studio.

Isso também torna possível verificar em tempo real todos os eventos que estão sendo executados no celular e isso auxilia o desenvolvedor na detecção de bugs ou falha no seu projeto.

4 RESULTADOS E DISCUSSÃO

O projeto AnAr, foco do presente trabalho, apresentou resultados satisfatórios e eficácia em todos os tipos de acionamentos possíveis, através do computador, inserindo o endereço IP, através do aplicativo pelos botões e também pela fala.

Para comprovar a eficácia do projeto um vídeo demonstrando seu funcionamento está disponível no domínio YouTube no seguinte endereço https://www.youtube.com/watch?v=YcFz0RYFAHs, onde é possível observar o acionamento e desligamento do motor, geração de log no cartão de memória, a exclusão do arquivo de log e também a leitura da temperatura.

Para a apresentação um roteador conectado à Internet foi usado e o celular conectado a mesma rede que provê conexão com a internet para a obtenção do texto resultante da fala através do mecanismo de reconhecimento de voz Google.

Uma vez que o projeto funciona como planejado é possível adentrar com mais equipamentos ou dispositivos para serem acionados ou monitorados com pequenas modificações em seu código principal, apenas adicionando linhas para tomar ação em relação ao texto obtido pelo comando de voz no aplicativo Android e inserir funções de ação na Arduino.

Assim o projeto pode se tornar grande aliado para projetos futuros de estudantes de Engenharia ou usuários que desejam habilitar funções de acionamento em sua própria residência, uma vez que a base para este estará disponível no domínio GitHub.

Numa residência ou outro ambiente talvez se faça necessária a criação de usuário com prioridade sobre outros, assim, para o caso de mais de um usuário o AnAr deve tomar por prioridade um destes para que não ocorra sobreposição de comandos, o que pode tornar a execução falha.

6 CONCLUSÃO

No início do projeto não se poderia prever como algumas facilidades por parte dos desenvolvedores do sistema Android podem se aliar a ideias novas e trazer a tona métodos de automação residenciais de baixo custo e baixo consumo de energia. Toda elaboração do projeto é simples e de fácil compreensão para qualquer usuário com apenas o mínimo de conhecimento no uso de celular e da internet.

O AnAr pode ser usado como base para novos projetos de automação que necessitem de um sistema de reconhecimento de voz associado à uma placa Arduino que possa acionar ou monitorar dispositivos trazendo resultados em tempo real e sem a necessidade de entrada por teclado de computador ou outros meios hoje existente, mas somente através da fala e de forma que exige poucos recursos financeiros, uma vez que basta ter um celular e um roteador seja o suficiente para acionamento da placa.

REFERÊNCIAS

Accardi, Adonis. Dodonov, Eugeni. Título: Automação residencial: elementos básicos, arquiteturas, setores, aplicações e protocolos. T.I.SSão Carlos, v. 1, n. 2, p156-166 nov. 2012

Android Open Source Project. Disponível em:

<https://source.android.com)>. Acesso em, 16 maio de 2017>

ARDUINO Mega 2560. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardMega 2560>. Acesso em: 18 maio 2017

ARDUINO UNO R3. Disponível em:

<http://arduino.cc/en/Main/ArduinoBoardMega 2560>. Acesso em: 18 maio 2017

BOLZANI, Caio A. M. Residências Inteligentes. 1ª ed. São Paulo: Física, 2004.

Jacques, João C. Z. Sistema de automação de segurança e combate à incêndio utilizando a plataforma arduino,Centro Universitário de Barra Mansa2015.

REFERÊNCIAS LINGUAGEM ARDUINO. Disponível em:

<https://www.arduino.cc/en/Reference/HomePage>. Acesso em: 18 maio 2017

XML eXtensibleMarkupLanguage. Disponível em:

<https://www.w3schools.com/xml/xml_whatis.asp>. Acesso em, 19 maio de 2017>

ANEXO A - Código Fonte Arduino

```
/*
 *
  versão final 13/06

 */
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
#include <DS3231.h>

File myFile;

String string, log_leitura, command, data;
float temperatura;
byte mac[] = {   0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 0, 129);
EthernetServer server(80);
DS3231 rtc(SDA, SCL);
int controle = 0;
int mostrar_temp = 0;
int motorSpin = A0;
int led = A1;

void setup() {

//  Serial.begin(9600);
 pinMode (motorSpin, OUTPUT);
  pinMode (led, OUTPUT);
  Ethernet.begin(mac, ip);
  server.begin();
  rtc.begin();

    if (!SD.begin(4)) {

    return;
  }
}
```

```
void loop() {
  EthernetClient client = server.available();
  if (client) {
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        string.concat(c);
        if (string.endsWith("15")) {
              command = "LOG";
              leituraSD();
        }
           if (string.endsWith("/14")) {
          command = "MOTOR ON";
          funcao14();
        }
           if (string.endsWith("/13")) {
          command = "Stop MOTOR";
          funcao13();
        }
         if (string.endsWith("/12")) {
        command = "Temperature";
         funcao12();

}
     if (string.endsWith("/11")) {
command = "Deletado";
funcao11();
}

  if (c == '\n' && currentLineIsBlank) {
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close");
  client.println();
  client.println("<!doctype html>");
  client.println("<html>");
  client.println("Sem comando");
  client.println("</html>");
  client.println();
  delay(1);
  client.stop();
  }

 if (c == '\n') {
            currentLineIsBlank = true;
}
else if (c != '\r') {
```

```
            currentLineIsBlank = false;
          }
          }
          if (controle == 1);
          log_in();
          controle = 0;
        }
      }
  }

  void funcao14 (){
    EthernetClient client = server.available();
      while (client.connected()) {
        if (client.available()) {
      if (client){
     client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connection: close");
            client.println();
            client.println(command);
            client.println();
            delay(10);
            client.stop();
          analogWrite(motorSpin, 230);

     controle = 1;
        }
       }
     }
    }


void funcao13 (){
    EthernetClient client = server.available();
     while (client.connected()) {
       if (client.available()) {
     if (client){
    client.println("HTTP/1.1 200 OK");
           client.println("Content-Type: text/html");
           client.println("Connection: close");
           client.println();
          client.println(command);
          delay(10);
          client.println();
         client.stop();
        }
       }
     }
```

```
  analogWrite(motorSpin, 0);
  controle = 1;
}

void funcao12 (){
  temperatura = (rtc.getTemp());
  EthernetClient client = server.available();
    while (client.connected()) {
      if (client.available()) {
    if (client){
   client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println();
          client.print(temperatura);
          delay(25);
          client.println();
          client.stop();
        }
      }
    }
}
void funcao11 (){
    EthernetClient client = server.available();
    while (client.connected()) {
      if (client.available()) {
    if (client){
   client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println();
         client.println("Remove LOG");
         delay(10);
          client.println();
         client.stop();
        }
      }
    }
     controle = 1;
    if (SD.exists("test.txt")){
      SD.remove("test.txt");
    }
}
```

```
void log_in(){
  if (controle == 1){
   data.concat(rtc.getDOWStr());
    data.concat(" ");
     data.concat(rtc.getDateStr());
      data.concat(" ");
       data.concat(rtc.getTimeStr());
           data.concat(" ");

  myFile = SD.open("test.txt", FILE_WRITE);
  // se tudo estiver OK com o arquivo irá escrever:
  if (myFile) {
    myFile.print("> ");
     myFile.print(data);
      myFile.print(" " );
    myFile.println(command);

    myFile.close();

  } else {
  //Serial.println("Falha ao escrever no arquivo");
  }
   data = (" ");
   controle = 0;
   command = (" ");
}
}


void leituraSD (){
   myFile = SD.open("test.txt");
  if (myFile) {
       // read from the file until there's nothing else in it:
    while (myFile.available()) {
     char try1 = myFile.read();
       log_leitura.concat(try1);
       }}
       else {
     EthernetClient client = server.available();
        if (client){
          boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println("Connection: close");
```

```
    client.println();
    client.println("Erro cartão");
    client.println();
    client.println();
    delay(10);
    client.stop();
    }
      }
    }
  }
  EthernetClient client = server.available();
    if (client){
      boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {

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

    delay(100);
    client.stop();
    log_leitura = (" ");
    }
    }
    }
    // close the file:
   myFile.close();

}
```

ANEXO B – Código Fonte aplicativo AnAr

```
1   /*Ultima versão
2   Foi retirado o Else do OnActivityResult, estava dando um get vazio no servidor
3   o que resultava falha na execução
4
5    */
6   package com.example.alvari.anar;
7
8   import android.content.Context;
9   import android.content.Intent;
10  import android.net.ConnectivityManager;
11  import android.net.NetworkInfo;
12  import android.os.AsyncTask;
13  import android.os.Bundle;
14  import android.speech.RecognizerIntent;
15  import android.support.v7.app.AppCompatActivity;
16  import android.support.v7.widget.Toolbar;
17  import android.util.Log;
18  import android.view.KeyEvent;
19  import android.view.View;
20  import android.view.Menu;
21  import android.view.MenuItem;
22  import android.view.inputmethod.EditorInfo;
23  import android.widget.Button;
24  import android.widget.EditText;
25  import android.widget.TextView;
26
27  import java.io.IOException;
28  import java.io.InputStream;
29  import java.io.InputStreamReader;
30  import java.io.Reader;
31  import java.net.HttpURLConnection;
```

```
import java.net.URL;
import java.util.List;

public class MainActivity extends AppCompatActivity {
    EditText urlText;
    Button bt2, bt3, bt4, bt5, bt6, bt7, audio;
    String cont, cont1;
    TextView testText, testServ, testComm;
    private static final String DEBUG_TAG = "HttpExample";
    private static final int SPEECH_REQUEST_CODE = 0;

    public void onUrlAction() {
        cont1 = ("Urlpronta");
        String url = urlText.getText().toString();
        new DownloadTask().execute("http://" + url + cont);
    }

    public void isOnline() {
        ConnectivityManager connMgr = (ConnectivityManager)
                getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected()) {
            testText.setText("Conexão com a Internet: SIM");

        } else {
            testText.setText("Conexão com a Internet: NÃO");
        }

    }

```

```
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
    if (requestCode == SPEECH_REQUEST_CODE && resultCode == RESULT_OK) {
        List<String> results = data.getStringArrayListExtra(
                RecognizerIntent.EXTRA_RESULTS);
        String spokenText = results.get(0);

        if ((spokenText.equals("mostrar log")||(spokenText.equals("mostrar dados do cartão")))) {
            cont = ("/15");
            bt1m();
        }

        if (spokenText.equals("Ligar motor")) {
            cont = ("/14");
            bt1m();
        }
        if ((spokenText.equals("Desligar motor") || (spokenText.equals("desligar motor")))) {
            cont = ("/13");
            bt1m();
        }
        if ((spokenText.equals("Qual a temperatura") || (spokenText.equals("qual é a temperatura")))) {
            cont = ("/12");
            bt1m();
        }
        if ((spokenText.equals("deletar log") || (spokenText.equals("deletar log")))) {
            cont = ("/11");
            bt1m();
        } /*else {
            cont = (" ");
            bt1m();
```

```
92
93              }*/
94
95
96          }
97          super.onActivityResult(requestCode, resultCode, data);
98      }
99
100     @Override
101     protected void onCreate(Bundle savedInstanceState) {
102         super.onCreate(savedInstanceState);
103         setContentView(R.layout.activity_main);
104         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
105         setSupportActionBar(toolbar);
106
107         urlText = (EditText) findViewById(R.id.urlText);
108         urlText.setOnKeyListener((v, keyCode, event)  -> {
111                 if (keyCode == EditorInfo.IME_ACTION_SEARCH ||
112                         keyCode == EditorInfo.IME_ACTION_DONE ||
113                         event.getAction() == KeyEvent.ACTION_DOWN &&
114                                 event.getKeyCode() == KeyEvent.KEYCODE_ENTER) {
115                     onUrlAction();
116                     return true;
117                 }
118                 return false;
119         });
121
122         testText = (TextView) findViewById(R.id.testText);
123         testServ = (TextView) findViewById(R.id.testServ);
124         isOnline();
```

```
125         bt2 = (Button) findViewById(R.id.bt2);
126         bt2.setOnClickListener((v)  -> {
129                 cont = ("/10");
130                 bt1m();
131         });
133         bt3 = (Button) findViewById(R.id.bt3);
134         bt3.setOnClickListener((v)  -> {
137                 cont = ("/11");
138                 bt1m();
139         });
141
142         bt4 = (Button) findViewById(R.id.bt4);
143         bt4.setOnClickListener((v)  -> {
146                 cont = ("/12");
147                 bt1m();
148         });
150
151         bt5 = (Button) findViewById(R.id.bt5);
152         bt5.setOnClickListener((v)  -> {
155                 cont = ("/13");
156                 bt1m();
157         });
159
160         bt6 = (Button) findViewById(R.id.bt6);
161         bt6.setOnClickListener((v)  -> {
164                 cont = ("/14");
165                 bt1m();
```

```
                bt1m();
            });

            bt7 = (Button) findViewById(R.id.bt7);
            bt7.setOnClickListener((v) -> {
                    cont = ("/15");
                    bt1m();
            });

            audio = (Button) findViewById(R.id.audio);
            audio.setOnClickListener((v) -> {

                    // Create an intent that can start the Speech Recognizer activity
                    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
                    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    // Start the activity, the intent will be populated with the speech text
                    startActivityForResult(intent, SPEECH_REQUEST_CODE);

            });
        }

        private void bt1m() {
            textChange();
            String url = urlText.getText().toString();
            new DownloadTask().execute("http://" + url + cont);
        }

        private class DownloadTask extends AsyncTask<String, Void, String> {
            @Override
            protected String doInBackground(String... urls) {

                // params comes from the execute() call: params[0] is the url.
                try {
                    return downloadUrl(urls[0]);
                } catch (IOException e) {
                    return "Checar servidor";
                }
            }
            // onPostExecute displays the results of the AsyncTask.
            @Override
            protected void onPostExecute(String result) {

                testServ.setText(result);
            }
        }

            private String downloadUrl(String myurl) throws IOException {
                InputStream is = null;
                // Only display the first 500 characters of the retrieved
                //alterado
                // web page content.
                int len = 500;

                try {
                    URL url = new URL(myurl);
                    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                    conn.setReadTimeout(10000 /* milliseconds */);
                    conn.setConnectTimeout(15000 /* milliseconds */);
                    conn.setRequestMethod("GET");
```

```
232                 conn.setDoInput(true);
233                 // Starts the query
234                 conn.connect();
235                 int response = conn.getResponseCode();
236                 Log.d(DEBUG_TAG, "The response is: " + response);
237                 is = conn.getInputStream();
238
239                 // Convert the InputStream into a string
240                 String contentAsString = readIt(is, len);
241                 return contentAsString;
242
243                 // Makes sure that the InputStream is closed after the app is
244                 // finished using it.
245             } finally {
246                 if (is != null) {
247                     is.close();
248                 }
249             }
250         }
251
252         public String readIt(InputStream stream, int len) throws IOException {
253             Reader reader = null;
254             reader = new InputStreamReader(stream, "UTF-8");
255             char[] buffer = new char[len];
256             reader.read(buffer);
257             return new String(buffer);
258         }
259
```

```
260         public void textChange() {
261             if (cont.equals("/15")) {
262                 testComm = (TextView) findViewById(R.id.testComm);
263                 testComm.setText("Mostrando LOG");
264             }
265             if (cont.equals("/14")) {
266                 testComm = (TextView) findViewById(R.id.testComm);
267                 testComm.setText("Comando: Acionar Motor");
268             }
269             if (cont.equals("/13")) {
270                 testComm = (TextView) findViewById(R.id.testComm);
271                 testComm.setText("Comando: Desligar Motor");
272             }
273             if (cont.equals("/12")) {
274                 testComm = (TextView) findViewById(R.id.testComm);
275                 testComm.setText("Comando: Mostrar temperatura");
276             }
277             if (cont.equals("/11")) {
278                 testComm = (TextView) findViewById(R.id.testComm);
279                 testComm.setText("Comando: Deletar arquivo de LOG");
280             }
281         }
282
283         @Override
284         public boolean onCreateOptionsMenu(Menu menu) {
285             // Inflate the menu; this adds items to the action bar if it is present.
286             getMenuInflater().inflate(R.menu.menu_main, menu);
287             return true;
288         }
289
```

```
290         @Override
291         public boolean onOptionsItemSelected(MenuItem item) {
292             // Handle action bar item clicks here. The action bar will
293             // automatically handle clicks on the Home/Up button, so long
294             // as you specify a parent activity in AndroidManifest.xml.
295             int id = item.getItemId();
296
297             //noinspection SimplifiableIfStatement
298             if (id == R.id.action_settings) {
299                 return true;
300             }
301
302             return super.onOptionsItemSelected(item);
303         }
304     }
305
```