REDE DOCTUM DE ENSINO
INSTITUTO TECNOLÓGICO DE CARATINGA
CURSO SUPERIOR DE ENGENHARIA ELÉTRICA

Automação de Interfone com Arduíno

EDERSON LOPES GOULART

Trabalho de Conclusão de Curso

Caratinga/MG
2017

EDERSON LOPES GOULART

AUTOMAÇÃO DE INTERFONE COM ARDUÍNO

Trabalho de Conclusão de Curso apresentado à Banca Examinadora do Curso Superior de Engenharia Elétrica do Instituto Tecnológico de Caratinga da DOCTUM Caratinga como requisito parcial para obtenção do Grau de Bacharel em Engenharia Elétrica.

Professor Orientador: Vinícius Murilo Lima Rodrigues.

Caratinga/MG

2017

rede de ensino DOCTUM	FACULDADES INTEGRADAS DE CARATINGA	FORMULÁRIO 9
	TRABALHO DE CONCLUSÃO DE CURSO	
	TERMO DE APROVAÇÃO	

TERMO DE APROVAÇÃO

O Trabalho de Conclusão de Curso intitulado: **AUTOMAÇÃO DE INTERFONE COM ARDUÍNO**, elaborado pelo aluno **EDERSON LOPES GOULART** foi aprovado por todos os membros da Banca Examinadora e aceito pelo curso de Engenharia Elétrica das FACULDADES DOCTUM DE CARATINGA, como requisito parcial da obtenção do título de:

BACHAREL EM ENGENAHRIA ELÉTRICA.

Caratinga 06 de julho 2017

VINÍCIUS MURILO LIMA RODRIGUES

RICARDO BOTELHO CAMPOS

DANIEL MAGESTE BUTTERS

DEDICATÓRIA

À Deus que provê tudo em abundância...

AGRADECIMENTOS

Agradeço primeiramente a Deus que sempre me guia.

Aos meus pais, José da Silveira Goulart e Maria da Glória Goulart, que deram todo o suporte necessário para minha formação.

A minha filha Júlia Lima Goulart por seu carinho e compreensão nessa jornada e que é a maior motivação de todos as minhas conquistas.

A todos os professores que se dispuseram a compartilhar com tanta dedicação um pouco de seu conhecimento e suas experiências.

A todos os amigos e colegas que dividiram comigo essa jornada.

“No meio da dificuldade encontra-se a oportunidade."

(ALBERT EINSTEIN)

RESUMO

Conforto é o que todos os usuários esperam de um sistema de automação. Além disso, os usuários desejam praticidade, caso contrário, depois de todo o trabalho de projeto e implantação, o mesmo pode cair em desuso por falta de interesse assim que passa o entusiasmo por uma nova tecnologia, por isso, ao desenvolver qualquer equipamento ou sistema, deve-se levar em consideração os desejos e as necessidades dos usuários.

O presente projeto prevê justamente preencher uma necessidade dos usuários que é poder atender o interfone de qualquer lugar que estiver bastando apenas ter cobertura de telefonia celular ou telefone fixo.

A proposta é um interfone com o recurso de ligação que além de evitar invasões nas residências, permite atender remotamente quem realmente precisa, como por exemplo um carteiro, leiturista ou até mesmo outro morador que tenha perdido as chaves.

Será utilizado a plataforma Arduíno para controlar o interfone, juntamente com alguns de seus módulos: o módulo GSM, que será responsável pela chamada telefônica, o detector de som que irá detectar o toque da campainha e um relé que fará o papel de retirada do fone do gancho simulando o atendimento.

Palavras Chave: Automação, Arduíno, Interfone, GSM.

ABSTRACT

Comfort is what all users expect from an automation system. In addition, users want practicality, otherwise, after all the design and deployment work, it may fall into disuse because of lack of interest as the enthusiasm for new technology passes, so when developing any equipment or system , One must take into account the desires and needs of users.

The present project provides precisely to fulfill a need of the users that is able to answer the intercom from any place that is just having only coverage of cell phone or landline.

The proposal is an intercom with the connection feature that in addition to avoiding invasions in the residences, allows to attend remotely who really needs, such as a postman, reader or even another resident who has lost the keys.

The Arduino platform will be used to control the intercom along with some of its modules: the GSM module, which will be responsible for the telephone call, the sound detector that will detect the ringing of the bell and a relay that will play the role of handset Of the hook simulating the service.

Keywords: Automation, Arduino, Interphone, GSM.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

A evolução das tecnologias sempre caminhou junto com o surgimento das novas necessidades da humanidade, a qual está sempre em busca do desenvolvimento de novos mecanismos que possam lhes trazer benefícios como conforto, qualidade de vida e segurança com baixo custo e que tenha fácil implantação e manutenção barata.

A automação residencial surgiu justamente para melhorar a qualidade de vida, segurança, praticidade e conforto, podendo ser executada remotamente ou localmente, autonomamente ou manualmente, dependendo da necessidade e perfil de cada usuário, além da disponibilidade orçamentária.

A mudança na forma de como as pessoas se conectam e se comunicam hoje em dia, somado a popularização dos equipamentos eletrônicos, proporciona uma vasta gama de praticidades na realização de tarefas do cotidiano jamais antes pensadas.

Muitas residências já possuem, por assim dizer, um certo grau de automação residencial, como alarmes, câmeras, porteiros eletrônicos, etc, porém, na maioria das vezes a questão da segurança não é pensada nem implantada de modo eficiente, ou seja, se limitando apenas a vigilância passiva através de câmeras e alarmes, que somente em caso de alguma ocorrência podem ser utilizados para recuperar uma imagem ou afugentar o invasor.

Com o aumento da utilização da eletrônica e em consequência, a sua popularização, é possível criar soluções baratas e fáceis de implementar visando a prevenção de ocorrências indesejáveis.

Sabemos que a maioria da população fica fora de casa praticamente o dia todo por estarem trabalhando, estudando ou até mesmo viajando e em consequência, as residências ficam vazias e vulneráveis a invasões e furtos, podendo gerar grande prejuízo material.

Este trabalho tem por finalidade propor a integração de um Arduíno e um Módulo GSM a um Interfone que, ao ser acionado, liga para o Celular ou telefone fixo do usuário, possibilitando fazer o atendimento de qualquer lugar coberto pela rede telefônica convencional ou Celular. Este tipo de solução pode ser muito útil para prevenir invasões, já que o bandido vai achar que o proprietário está dentro de casa, além de facilitar a comunicação com outros tipos de visitas.

1.1 OBJETIVOS

Este trabalho propõe a implantação de um interfone que ao toque da campainha, faça uma ligação para um número de telefone pré configurado, permitindo o usuário conversar diretamente com quem estiver chamando ao interfone.

1.2 JUSTIFICATIVA

Este trabalho facilita o atendimento de interfone remotamente, podendo, mesmo à distância, simular a presença dentro da residência para evitar invasão de bandidos ou simplesmente saber quem está chamando ao interfone.

O desenvolvimento deste sistema permitirá um estudo e aplicação dos conhecimentos adquiridos no curso, como eletrônica, lógica de programação e comunicação.

Será aprofundado o conhecimento em programação e eletrônica, através de ensaios em laboratório e leituras sobre os recursos do Arduíno e seus acessórios.

2. FUNDAMENTOS TEÓRICOS

2.1. O ARDUÍNO

O Arduíno permite projetar e testar diversos protótipos de forma simples e prática:

> O Arduíno é uma plataforma de hardware open source, projetada sobre o microcontrolador Atmel AVR, que pode ser programado através de uma linguagem de programação similar a C/C++, permitindo a elaboração de projetos com um conhecimento mínimo ou mesmo nenhum de eletrônica. Foi criado com o objetivo de fornecer uma plataforma de fácil prototipação de projetos interativos, unindo software e hardware, características da Computação Física. (Oliveira et al, 2016, p17)

2.1.1 CARACTERÍSTICAS

Na figura 1 temos o modelo Arduíno UNO, que é o modelo utilizado neste projeto.

> A placa Arduíno é muito similar à de um computador de pequeno porte, sendo composto por um microcontrolador, memória RAM, armazenamento secundário (memória flash) e clock, entre outras funcionalidades. (Oliveira et al, 2016, p17)

Figura 1 - Placa Arduíno UNO - Fonte: O autor, 2017.

2.1.2 ESPECIFICAÇÕES

O modelo de plataforma escolhido para o trabalho é o UNO, ele apresenta 14 pinos que podem ser utilizados como entradas ou saídas digitais (pinos 1 a 14), e os pinos 3, 5, 6, 9, 10 e 11 podem ser utilizados para gerar um conjunto de valores inteiros entre 0 e 1023 pela técnica PWM (Pulse Width Modulation). Os pinos A0 a A5 correspondem às entradas analógicas, enquanto os 3, 3V, 5V e GND (Terra) permitem alimentar os componentes dos circuitos conectados ao Arduíno. Possui um microprocessador ATmega320, com uma memória RAM de 2KB, memória Flash de 32 KB e velocidade de clock de 16MHz.

2.1.3 AMBIENTE DE DESENVOLVIMENTO

Assim como em qualquer dispositivo programável, a plataforma Arduíno necessita que os programas sejam desenvolvidos em uma linguagem de programação, compilados e, posteriormente transferidos para o Arduíno, de modo que seja possível a execução dos comandos utilizados no programa. Com o intuito de facilitar e tornar o processo mais produtivo, utilizamos um programa denominado ambiente integrado de programação, comumente chamado de IDE, do inglês Integrated Development Environment.

De acordo com Oliveira, Cláudio Luís Vieira, Autor do Livro Arduíno Descomplicado – Saraiva 2016,

> Um programa criado para o Arduíno é chamado de sketch, e o ambiente de desenvolvimento apresenta uma interface para o usuário bastante simples e intuitiva, apresentando recursos para abrir e salvar os sketches, transferir os programas criados para a placa, selecionar qual modelo do Arduíno será utilizado, entre várias outras funcionalidades. (Oliveira et al, 2016, p18)

A figura 2 mostra a tela do ambiente de desenvolvimento:

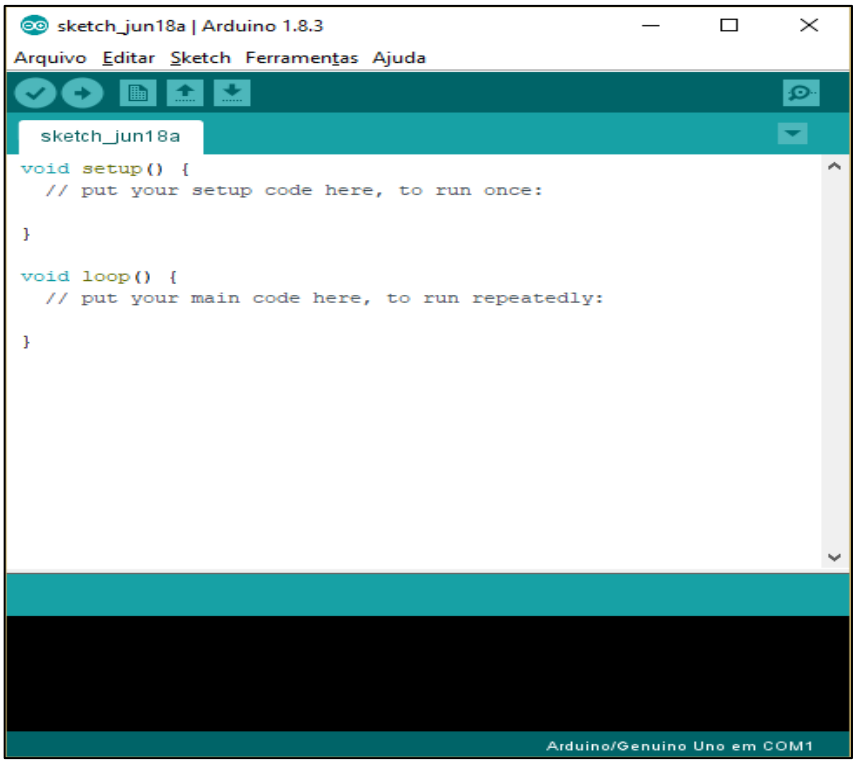

Figura 2 - Ambiente de Desenvolvimento – Fonte: O Autor, 2017.

2.1.4 MÓDULOS E SHIELDS PARA O ARDUÍNO

Módulos e Shields multiplicam as funcionalidades do Arduíno possibilitando várias aplicações, de acordo com Stevan Júnior:

> Um dos grandes legados da plataforma Arduíno é a padronização de características, posição de pinos de entradas e saídas de tensões e a possibilidade de conexões rápidas através de conectores de barras de pinos que permitem a conexão de periféricos sobre a plataforma Arduíno Utilizada. (Stevan Júnior, 2015, p142)

Shields geralmente possuem uma padronização de encaixe conforme diz Stevan Júnior:

> Esses periféricos são chamados de Shields, que são placas de circuito impresso que utiliza padronização geométrica e de pinos para alimentar e se comunicar com os periféricos adicionados, aumentando as funcionalidades disponíveis. (Stevan Júnior, 2015, p143)

Com a mesma aplicação e funcionalidades dos Shields, temos os módulos, porém sem uma padronização de pinagem, portando deve ser observado com atenção como será feita a conexão ao Arduíno, no que diz respeito à alimentação correta, entrada, saída, etc.

Em geral, módulos e Shields podem ser sensores, atuadores ou comunicação, a seguir temos uma lista dos que serão utilizados na realização deste trabalho:

- Sensor de som:

O objetivo deste sensor é medir a intensidade sonora do ambiente ao seu redor, variando o estado de sua saída digital caso detectado um sinal sonoro. Possui um microfone de condensador elétrico e pode ser usado em sistemas de alarme por exemplo.

O limite de detecção pode ser ajustado através do potenciômetro presente no sensor que regulará a saída digital D0. Contudo para ter uma resolução melhor é possível utilizar a saída analógica A0 e conectar a um conversor AD, como a presente no Arduíno por exemplo. Um modelo de sensor pode ser visto na figura 3:

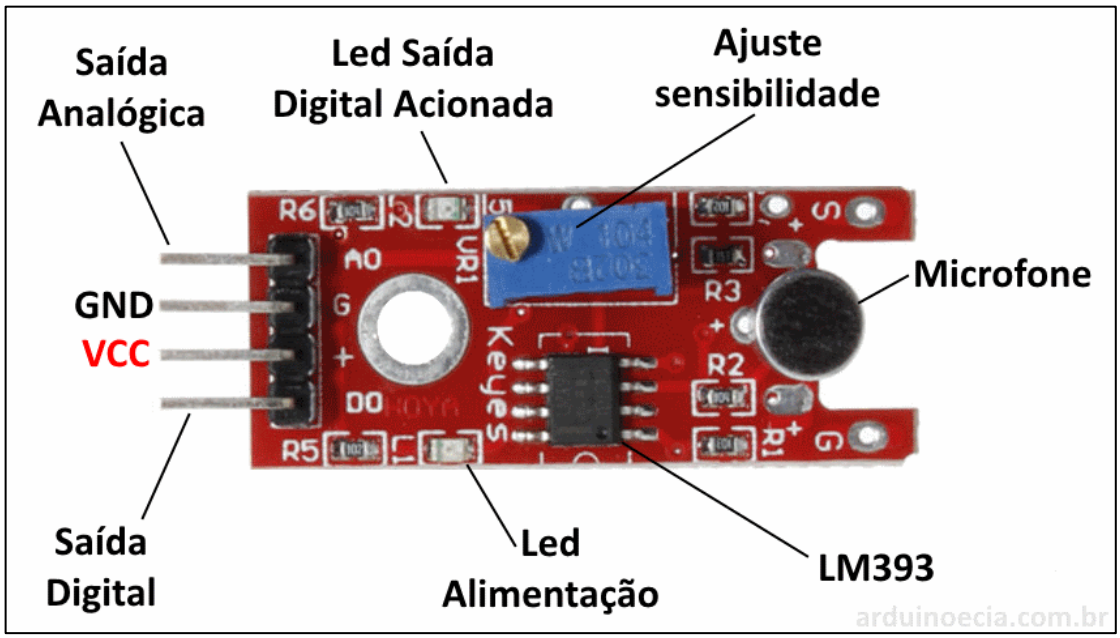

Figura 3 - Sensor de som - Fonte: Arduinoecia, 2017.

- Módulo Relé 5V

O módulo relé de saída dupla funciona com tensão de 5V, e pode acionar cargas de até 250 VAC ou 30 VDC, suportando uma corrente máxima de 10A. Possui led indicador de energia, 2 pinos de energia e 1 de controle, além do borne de saída com parafusos, com 2 saídas NF (Normalmente Fechada), 2 saídas NA (Normalmente Aberta) e 2 saídas Comum, facilitando a conexão dos equipamentos (Figura 4).

Figura 4 - Relé 5v - Fonte: GbkRobotics, 2017.

- Módulo GSM SIM 800L

O módulo GSM funciona com tensão de 3.7 a 4.2V e a interface serial USB-TTL pode ser conectada diretamente no microcontrolador. Possui luz indicadora de conexão na própria placa, e pinos de conexão tanto para **microfone** como para **alto-falante**. A Figura 5 mostra o módulo SIM800L:

Figura 5 - Módulo GSM SIM800L - Fonte: O Autor,2017.

Devido a seus vários recursos e ser um item essencial para este trabalho, este módulo será detalhado na seção específica.

2.2 O MÓDULO GSM

Existem vários módulos GSM que permitem, além de comunicação via celular, conexão de internet via GPRS e envio de mensagens SMS, ampliando mais ainda as possibilidades de controle e monitoramento através do Arduíno.

Diferentemente de outros modelos, ele já vem integrado na placa, necessitando apenas soldar os pinos (os quais acompanham). Possui integrado um slot para cartão SIM e uma antena de cobre, sem abrir mão de possibilitar conexão com outra antena.

Baseado no módulo wireless SIM800L, que é capaz de encaminhar serviços Quad-Band, ele permite fazer ligações de voz, enviar SMS e trocar dados via Internet GPRS (Serviço de Rádio de Pacote Geral) de forma a receber e realizar o envio de dados do Arduino para locais remotos junto de seu telefone celular GSM, por exemplo.

Após ampla pesquisa, foi escolhido o módulo GSM SIM800L, por ser de baixo custo e fácil implementação.

2.2.1 CARACTERÍSTICAS

As Principais características do módulo GSM SIM800L são:

- Conexão junto à rede GSM;
- Formas de utilização e controle aprimoradas;
- Maior facilidade e rapidez de instalação junto ao Arduíno;
- Baseado no módulo wireless SIM800L;
- Capaz de receber e realizar o envio de dados de locais remotos;
- Operação semelhante a um telefone celular;
- Comunicação rápida e completa a distância;
- Suporte de cartão SIM (Verificar sua localidade e banda em MHz utilizada pela operadora);
- Pinos para conexão com microfone e agentes externos de reprodução de áudio;
- Essencial para quem quer dar um upgrade nos projetos com Arduino;
- Acompanham barras pinos;

- Pinagem especificada diretamente no módulo;
- Baixo custo;

Na figura 6 pode-se ver os detalhes das conexões do módulo:

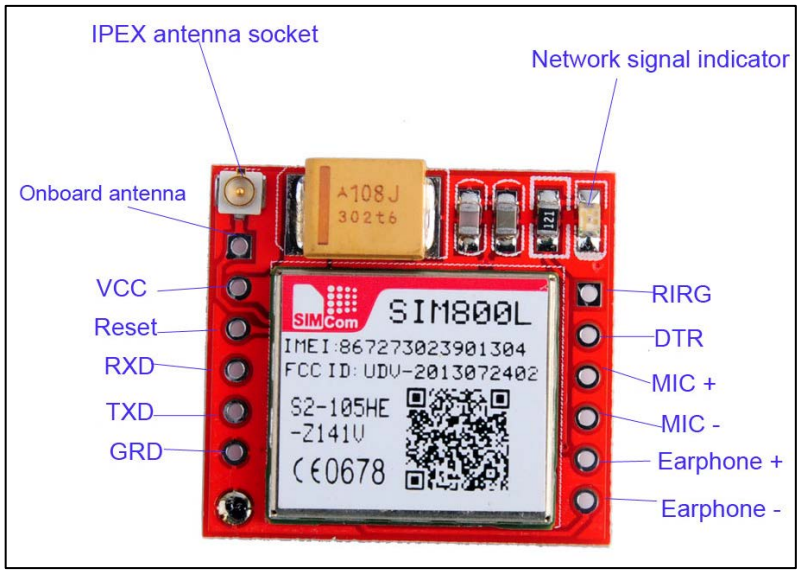

Figura 6 - Pinagem GSM 800L - Fonte: cityelectronics, 2017.

2.2.2 ESPECIFICAÇÕES

O **Módulo GSM SIM800L** foi desenvolvido especialmente para integrar projetos utilizando o microcontrolador Arduino à Rede Mundial de Computadores por meio da comunicação GPRS, realizar ligações (GSM) ou ainda enviar mensagens SMS.

A seguir temos suas especificações:

- Alimentação: 3.7V a 4.2V (2A corrente de pico);
- Quadl-Band: 850/900/1800/1900MHz;
- GPRS classe multi-slot 10/8;
- Estação móvel GPRS classe B;
- Ganho da antena: 3dBi;
- Controle via comandos AT (GPP TS 27, 007, 27, 005 e SIMCOM);
- Temperatura de operação: -40 °C a +85 °C;
- Dimensões (CxL): 25x23mm;

2.3 O INTERFONE

Existem vários fabricantes de interfones no mercado e este trabalho foi feito utilizando o modelo P10 da AGL. A figura 7 mostra o modelo P10 do fabricante AGL:

Figura 7 - Interfone modelo P10 - Fonte: AGL, 2017.

De acordo com o fabricante, estas são as especificações do interfone:

- Com fonte de alimentação : Externa (Linha P - 10).
- Para instalações com distância maior entre interfone e fechadura a melhor opção é a Linha P10.
- Para instalações com distância menores entre interfone e fechadura pode-se utilizar Linha P10 ou P20.
- Instalação : Alimentação 110v ou 220v.
- Acionamento de fechaduras 12 volts.
- Extensões : Possibilita a instalação de até 3 extensões.
- Ajustes : Volume.
- Extras : Protetor de chuva já incorporado.

A parte que interessa ao projeto é o monofone, pois ele é que será conectado ao Arduíno e ao Módulo GSM.

Após abrir a carcaça da base do monofone e o próprio monofone, é possível identificar os fios que ligam o microfone e o autofalante do fone. Na figura 8 é possível ver os detalhes

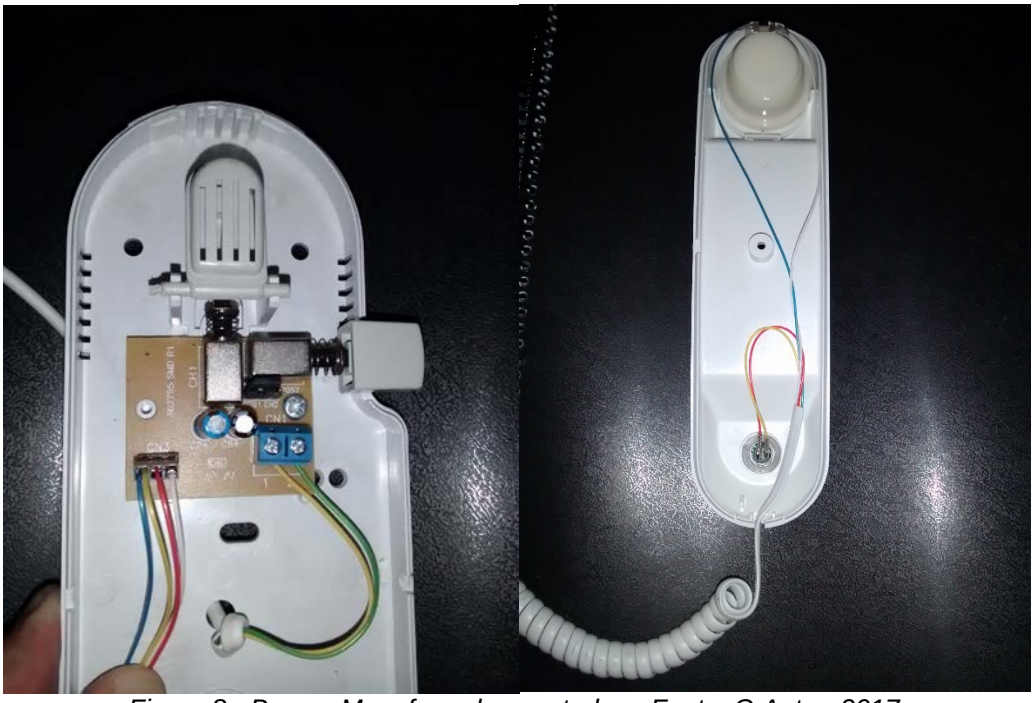

Figura 8 - Base e Monofone desmontados - Fonte: O Autor, 2017.

Assim, é possível ver que os fios amarelo e vermelho são do microfone e o azul e o branco são do falante.

Além disso, para identificar os terminais que o interruptor controla, foi preciso desmontar a placa da base:

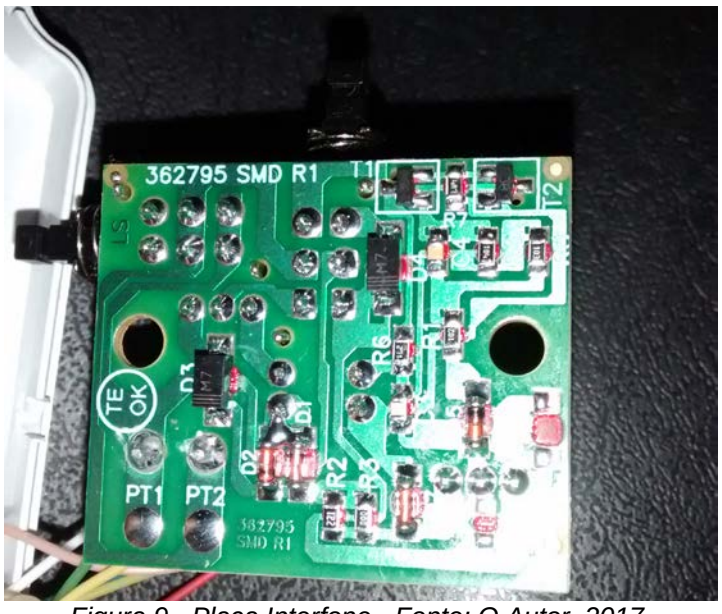

Figura 9 - Placa Interfone - Fonte: O Autor, 2017.

Assim é possível identificar os 6 terminais que são acionados quando o monofone é retirado/colocado na base.

A instalação do Interfone é bem simples, bastando apenas segui o diagrama do fabricante:

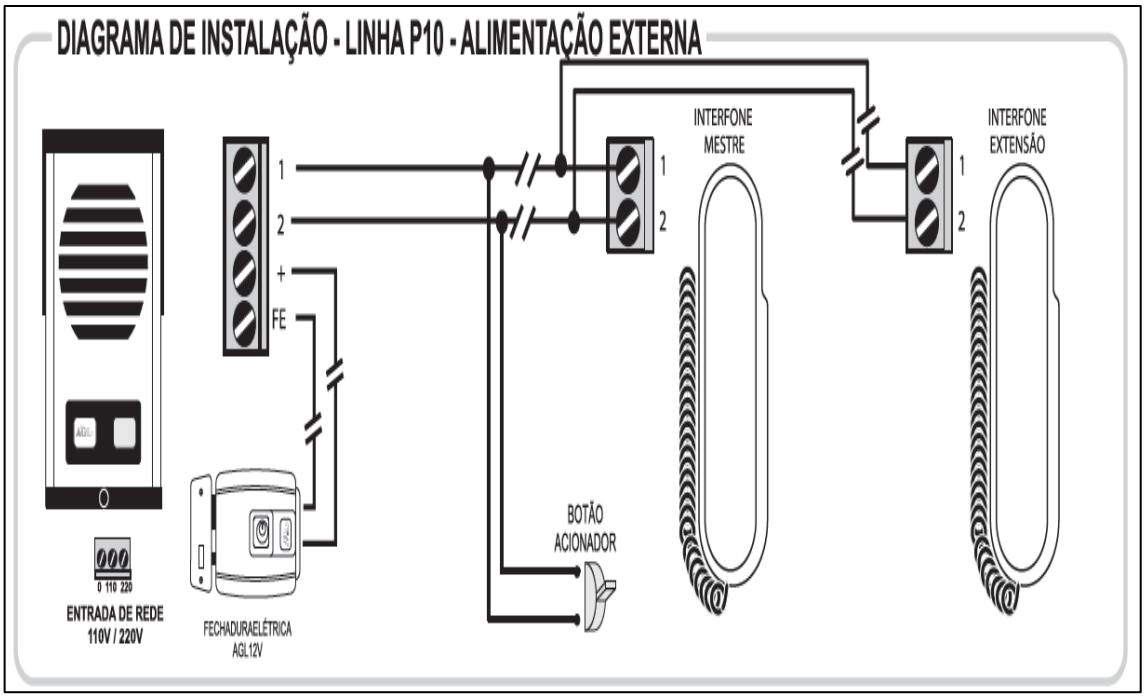

Figura 10 - Sugestão de Instalação do Interfone - Fonte: AGL, 2017.

3. METODOLOGIA APLICADA

3.1 PROJETO, MONTAGEM E CODIFICAÇÃO

Após conhecer as características e especificações dos equipamentos, podemos iniciar a preparação dos mesmos para que possa ser montado e testado.

3.1.1 Preparando o Arduíno

Primeiramente, a alimentação do Arduíno será do próprio cabo USB (figura 11) facilitando assim as conexões e o carregamento do programa que irá controlar o mesmo.

Figura 11-Arduíno Conectado a USB - Fonte: O Autor

Para conectar o Módulo GSM ao Arduíno, utilizaremos as portas de entrada 10 e 11 ligadas aos pinos TX e RX do Módulo GSM respectivamente, ligaremos também o Reset do Módulo ao Reset do Arduíno.

A alimentação do Módulo GSM é feita através do terminal de 5v do Arduíno e para adequar a tensão de trabalho do Módulo, é preciso incluir um diodo IN4007 no terminal positivo para que tenhamos uma queda de tensão de 0,7V, alimentando o Módulo com 4,3V.

O terminal GND (Terra) do Módulo GSM também é ligado ao terminal GND do Arduíno.

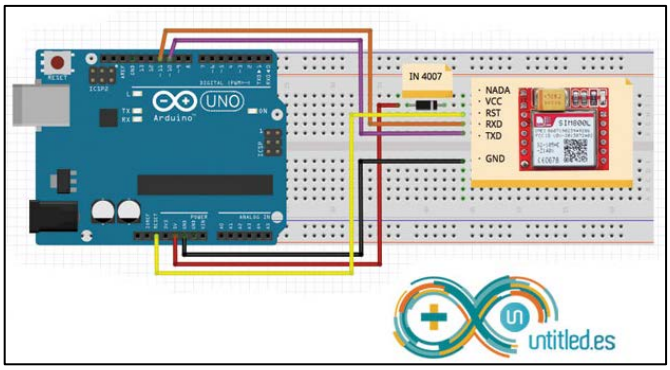

Figura 12 - Esquema de Ligação do Módulo GSM ao Arduíno - Fonte: untitled, 2017.

Após a ligação dos terminais podemos testar uma ligação, para isso inserimos o Chip da operadora no Módulo GSM e carregamos o código que controlará a ligação.

Foi conectado um pequeno auto-falante para apenas monitorar a discagem através do Módulo GSM.

O resultado das conexões dos terminais pode ser visto na figura 13:

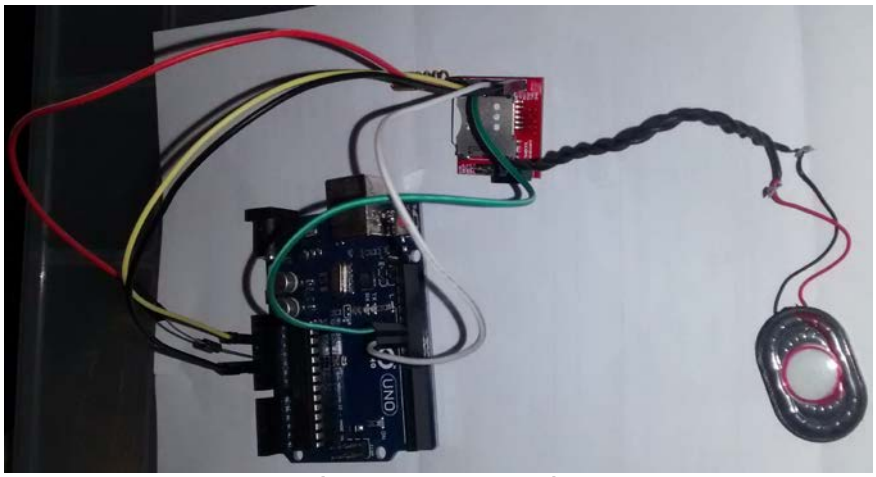

Figura 13 - Ligação do Módulo GSM ao Arduíno - Fonte: O Autor, 2017.

O código a seguir utiliza a demonstra como fazer uma ligação para um número de telefone utilizando a biblioteca do SIM900:

```
1  //Progama: Chamada telefonica usando o GSM Shield
2  //Autor: Arduino e Cia
3
4  #include "SIM900.h"
5  #include <SoftwareSerial.h>
6  #include "call.h"
7
8  CallGSM call;
9
10 void setup()
11 {
12   Serial.begin(9600);
13   Serial.print("Ligando shield GSM SIM800. ");
14   liga_desliga_GSMShield();
15   Serial.println("Testando GSM Shield...");
16   //Comunicacao com o Shield GSM a 2400 bauds
17   if (gsm.begin(2400))
18     Serial.println("nstatus=READY");
19   else Serial.println("nstatus=IDLE");
20 }
21
22 void loop()
23 {
24   Serial.println("Discando....");
25   //Efetua a chamada formato call.Call(<numero a ser chamado>)
26   call.Call("987654321");
27   Serial.println("Ligacao Efetuada!");
28   delay(20000);
29   Serial.println("Encerrando a ligacao...");
30   call.HangUp();
31   Serial.print("Desligando shield GSM SIM800. ");
32   liga_desliga_GSMShield();
33
34   do {} while (1); //Loop parando o programa
35 }
36
37 void liga_desliga_GSMShield()
38 {
39   Serial.print(F("Aguarde..."));
40   pinMode(6, OUTPUT);
41   digitalWrite(6, LOW);
42   delay(1000);
43   digitalWrite(6, HIGH);
44   delay(1000);
45   Serial.println(F("OK!"));
46   digitalWrite(6, LOW);
47   delay(500);
48 }
```

Figura 14 - Código para efetuar chamadas - Fonte: Arduíno e Cia, 2017.

Os comandos AT são um padrão de comunicação internacional que nos permite manipular dispositivos de comunicação através de um terminal.

Existem inúmeros comando AT (para enviar SMS, verificar status da rede GSM, etc), mas o comando que utilizaremos para discar para um número é o ATD <Número a ser chamado> mas para facilitar, utilizamos a biblioteca do SIM900, que possui

funções de chamadas que facilitam a implementação do código, no **Anexo 1** a função call está detalhada, podendo assim verificar os comando AT utilizados.

Na figura 15 é possível ver o Monitor Serial do Arduíno fazendo uma chamada:

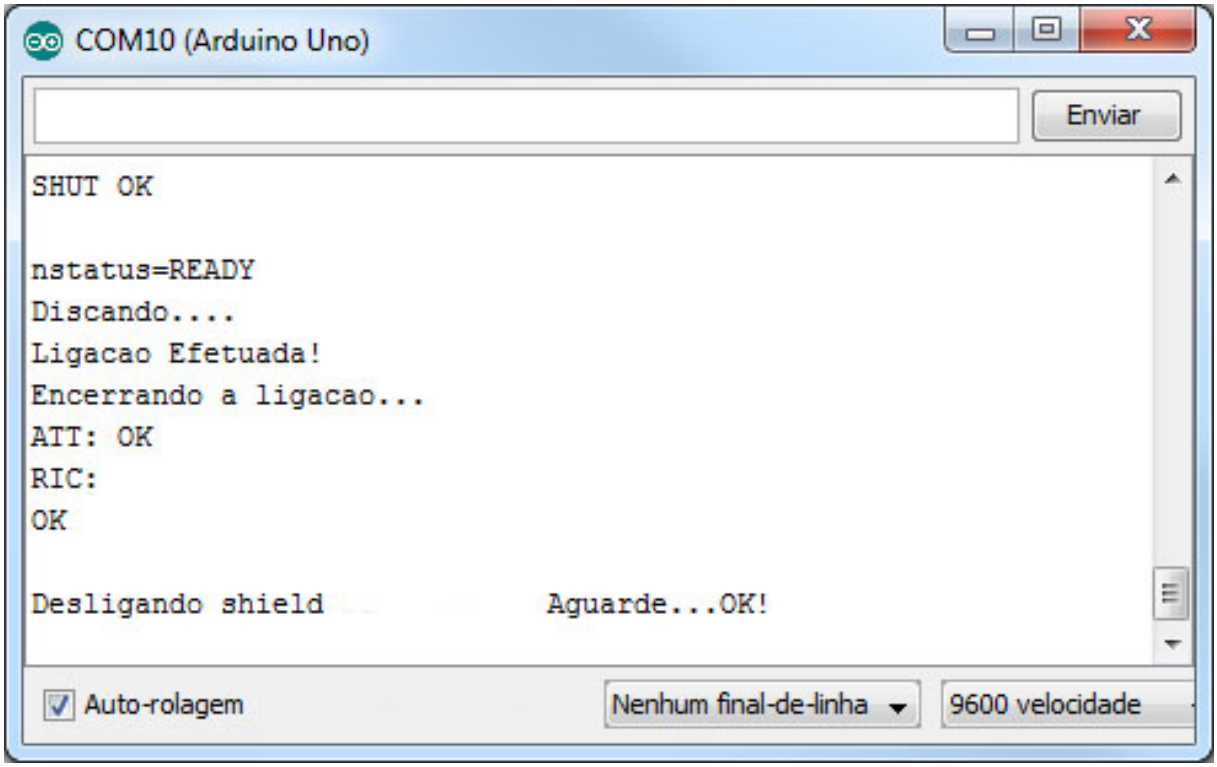

Figura 15 - Resultado do teste de discagem - Fonte: O autor

Depois do teste de discagem, foi feita a ligação de dois jacks fêmea nos terminas do microfone e do falante no módulo GSM (figura 16), para ambos se conectarem em paralelo com o microfone e autofalante do fone do Interfone.

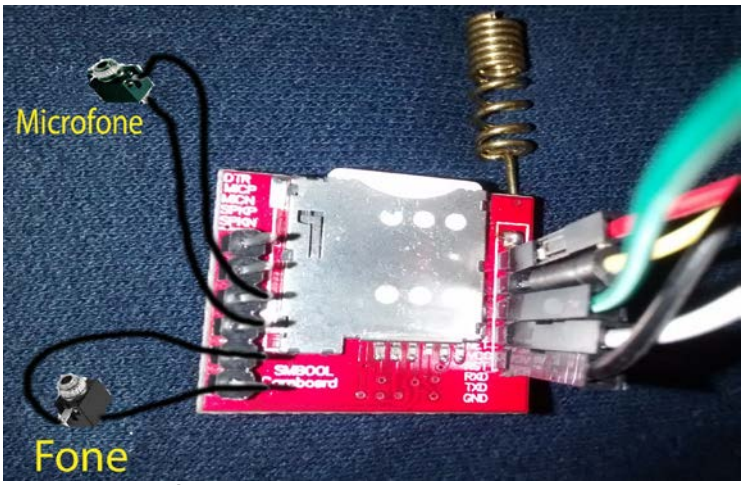

Figura 16 - Módulo GSM com os Jacks -Fonte: O Autor, 2017

O próximo passo foi ligar o terminal de sinal do sensor de som na porta 7 do Arduíno e sua alimentação no GND e 5V, este sensor irá detectar o toque da

campainha e informar ao Arduíno.

Figura 17 - Ligação do sensor de som ao Arduíno - Fonte: O Autor, 2017.

Este sensor foi calibrado para detectar somente o som da campainha, para que nenhum outro som possa interferir no funcionamento do sistema.

O código do **Apêndice A** serve para testar o referido sensor, que ao perceber sinal sonoro acende um LED (que no projeto final acionará o Relé de 2 saídas).

Para simular a retirada do fone do gancho, foi necessário ligar o terminal de sinal do Relé na porta 8 do Arduíno e sua alimentação no GND e 5V (figura 18) e assim finalizamos a preparação do Arduíno e seus componentes, o resultado final pode ser visto na figura 19.

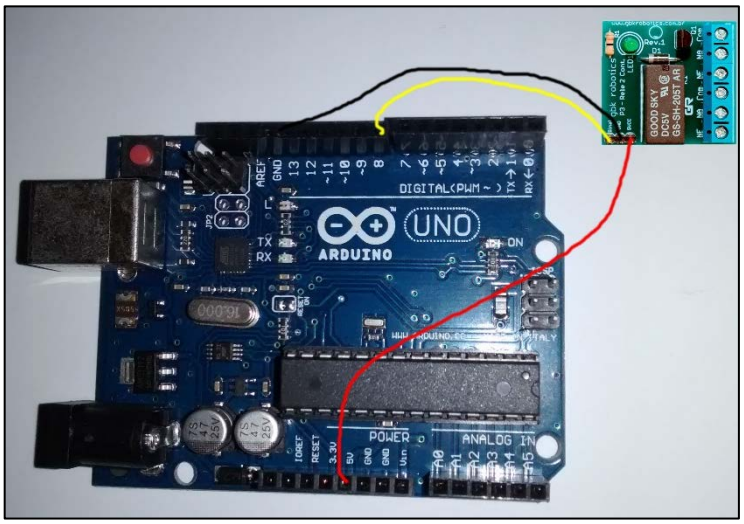

Figura 18 - Ligação do Rele ao Arduíno - Fonte: O Autor, 2017.

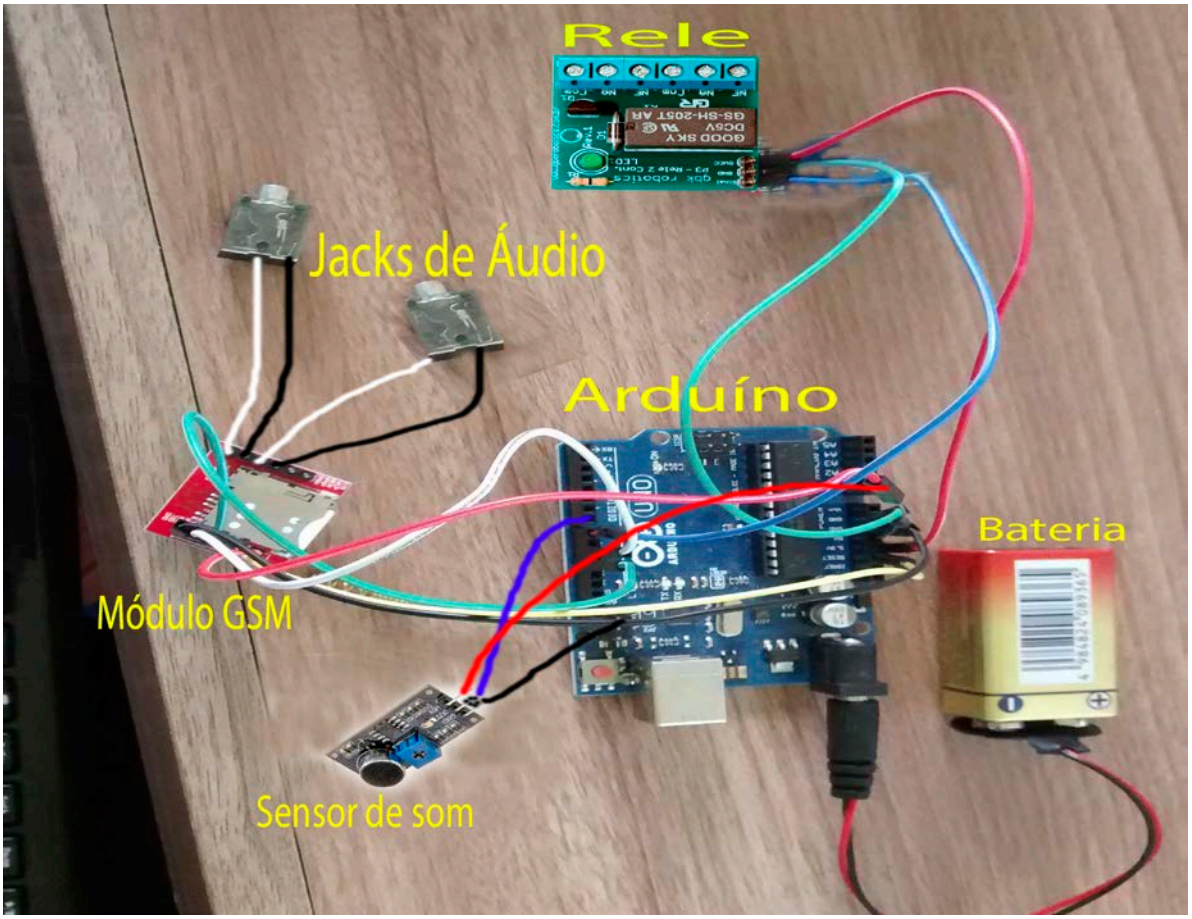

Figura 19 - Arduíno Preparado - Fonte: O Autor, 2017.

3.1.2 Preparando o Interfone

Na placa de circuito impresso da base do monofone foi preciso ligar 6 fios nos terminais do interruptor da base para serem ligados em paralelo com as saídas 1 e 2 do módulo Relé (figura 20).

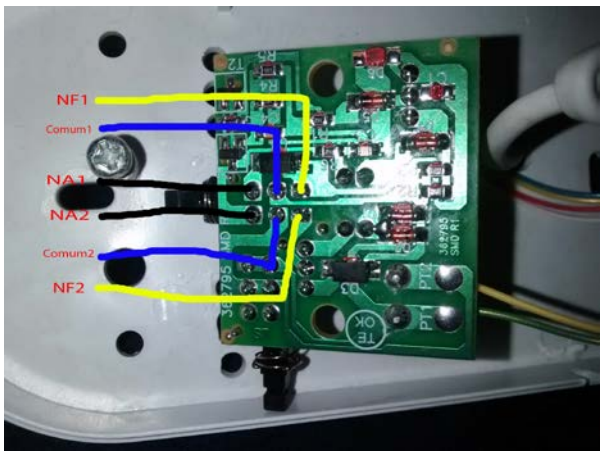

Figura 20 - Identificação dos terminais do Interruptor – Fonte: O Autor, 2017.

Tanto nos terminais do microfone, quanto nos terminais do falante, foi conectado um plug P1 para facilitar a conexão com os terminais de áudio do Módulo GSM conforme figura 21:

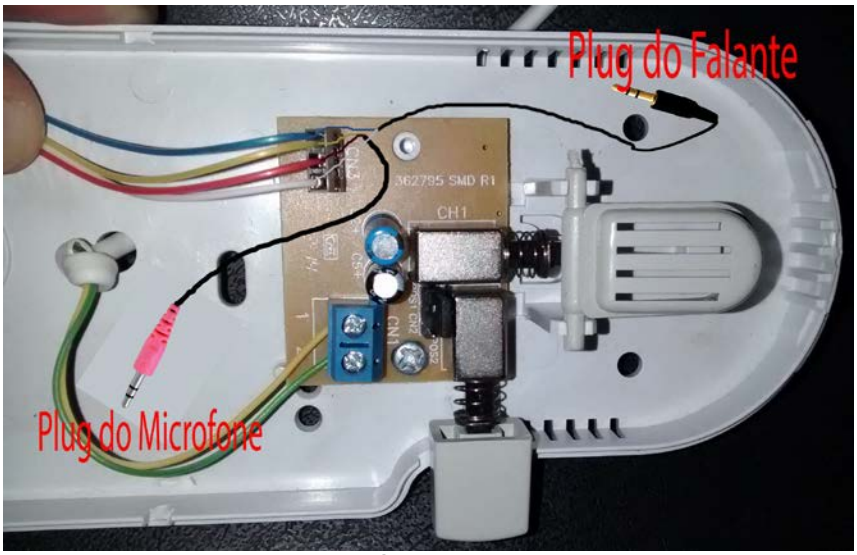

Figura 21 - Plugs Conectados à base do Fone - Fonte: O Autor, 2017.

Após esse passo finalizamos a preparação do Interfone conforme figura 22:

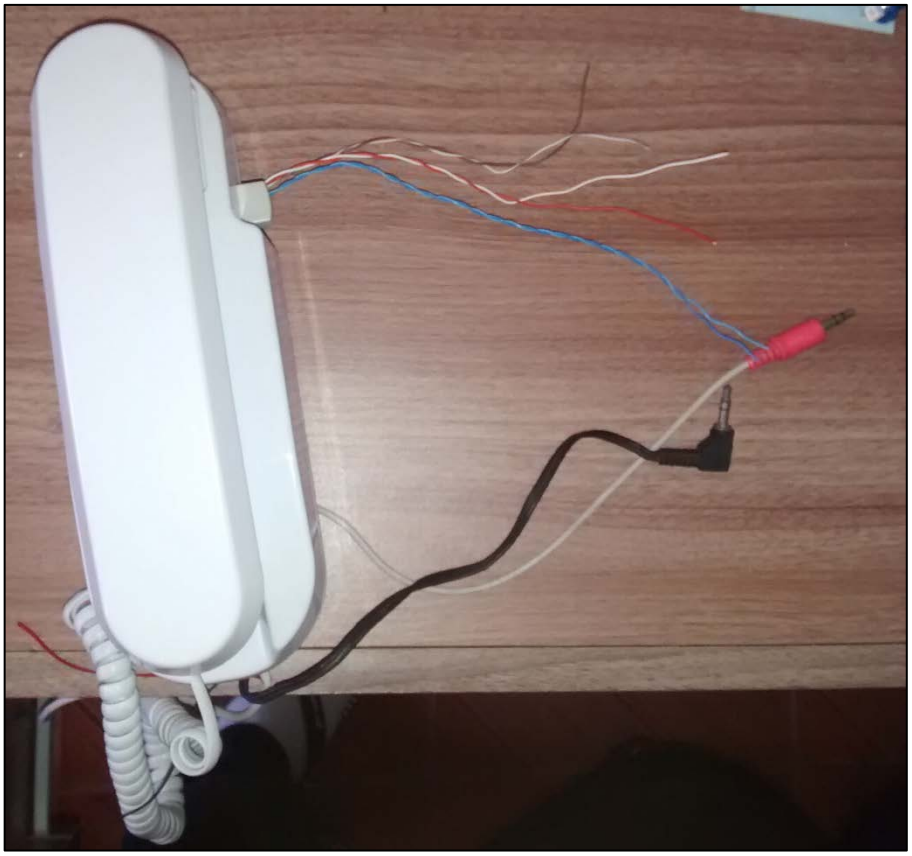

Figura 22 - Interfone finalizado - Fonte: O Autor, 2017.

3.1.3 Ligando o Arduíno ao Interfone

Após a preparação do Arduíno e do interfone, a conexão física entre eles é bem simples: O primeiro passo é conectar os plugs do microfone e do falante nos jacks correspondentes no Módulo GSM e posicionar o sensor de som próximo à campainha.

3.1.4 Codificação

A codificação do Arduíno consiste em ficar monitorando a campainha através do sensor de som e caso ela seja acionada, o Arduíno disca para o telefone configurado, aguarda 3 segundos e ativa o relé para simular o atendimento e conectar a chamada ao Interfone. O código completo pode ser conferido no **APÊNDICE B**.

4. CONCLUSÃO

O desenvolvimento do presente projeto possibilitou a aplicação das mais variadas disciplinas ministradas durante o curso, desde os cálculos, até as técnicas de soldagem e programação.

No seu processo de preparação e montagem foi possível demonstrar como uma boa idéia pode ser implementada de forma facilitada com a escolha de bons componentes como no caso da Plataforma Arduíno e seus módulos. A parte da preparação do Arduíno e seus componentes requer pouco conhecimento em eletrônica pois sua montagem é bem simples e isso facilita seu uso por qualquer pessoa que queira fazê-lo. Já na preparação do Interfone, foi preciso fazer sua desmontagem para identificar as conexões e inserir os terminais e isso já requer um certo conhecimento de eletrônica.

Como resultado, foi possível montar um equipamento de grande utilidade, que realizará uma ligação telefônica através do interfone e permitirá seu atendimento de qualquer lugar que tenha cobertura telefônica.

Além da possibilidade de uma simples ligação telefônica através do interfone, a implementação de novos recursos pode ser estudada para incrementar ainda mais suas funcionalidades, como por exemplo a possibilidade de abrir o portão remotamente, poder receber chamadas pelo interfone ou até mesmo conectar o interfone à Internet.

5. REFERÊNCIAS

EMPRESA AGL. **Porteiro eletrônico**. 2016. Dispinível em <http://www.aglfechaduras.com.br/porteiro-eletr-nico-agl-abs> . Acessado em: 30 de maio de 2017.

EMPRESA ARDUÍNO E CIA, **Tutorial: como acessar a internet com o Arduino GSM Shield SIM900,** 2015. Disponível em: <http://www.arduinoecia.com.br/2015/11/acessar-internet-arduino-gsm-shield-sim-900.html>. Acessado em: 25 de abril de 2017.

EMPRESA FACA COM ARDUÍNO. **Acendendo LED com sensor de som**, 2015. Disponível em: <http://facacomarduino.info/projeto-45-acendendo-led-com-sensor-de-som.html>. Acessado em: 20 junho de 2017.

EMPRESA INTELBRAS, VIDEO PORTEIRO IV 7000 HV, 2016. Disponível em: <http://www.intelbras.com.br/empresarial/controle-de-acesso/linha-residencial/videoportero/iv-7000-hs>. Acessado em 02 fevereiro de 2016.

MONK, Simon. **Programação com Arduíno**. 1ª ed. Porto Alegre. Bookman, 2013.

MONK, Simon. **Programação com Arduíno II**. 1ª ed. Porto Alegre. Bookman, 2015.

MONK, Simon. **Projetos com Arduíno e Android**. 1ª ed. Porto Alegre. Bookman, 2014.

OLIVEIRA, Cláudio Luis Vieira. **Arduíno descomplicado: como elaborar projetos de eletrônica**. 1ª ed. São Paulo. Érica, 2015.

OYOLA, Ignacio. **SIM 800L com Arduíno**. 2016. Disponível em: <http://untitled.es/sim800l-arduino/> . Acessado em: 05 de janeiro de 2017.

RODRIGUES, Vinicius. **Arduino GPRS - SIM900**, Fazendo ligação, 2013. Disponível em: <http://microembarcado.blogspot.com.br/2013/09/arduino-gprs-sim900-fazendo-ligacao.html>.

SOUZA, Fábio. **Arduino - Saídas PWM. 2014**. Disponível em: <https://www.embarcados.com.br/arduinosaidas-pwm/>. Acesso em: 02 de Outubro de 2016.

STEVAN JUNIOR, Sergio Luiz. **Automação industrial com Arduíno: Teoria e projetos.** 1ª ed. São Paulo. Erica, 2015.

THOMSEN, Adilson. **Fazendo chamadas com Arduíno**. 2014 Disponível em: <http://blog.filipeflop.com/wireless/tutorial-arduino-gsm-shield.html>. Acessado em 02 de março de 2017.

6. APÊNDICES

APÊNDICE A

```
//TESTE SENSOR DE SOM

int microfone = 7; // Pino digital ligado ao sensor de som
(DO)
int led = 8; // Pino ligado ao LED
int contPalmas = 0; // Contador de Palmas
int palmasRequeridaLigaLed = 2; // Contagem para acender o LED
int palmasRequeridaDesligaLed = 2; // Contagem para apagar o
LED
// Tempo máximo entre o pulso seguinte
unsigned long tempMaxSom = 1000;
unsigned long tempMinSom = 300;
unsigned long compriSonoro = 100; // Comprimento do som
esperado
unsigned long time;
unsigned long startTime = 0;
void setup() {
  pinMode(microfone, INPUT); // Inicia o pino do microfone
como entrada
  pinMode(led, OUTPUT); // Inicia os pino do LED como saída
  // Desliga LED (o LED é invertido HIGH = desliga / LOW =
liga)
  digitalWrite(led, HIGH);
}
void loop() {
  // Inicia a contagem de tempo
  time = millis();
  // Verifica o tempo entre o pulso inicial e o seguinte
  unsigned long tempoAposPalma = time - startTime;
  if (tempoAposPalma >= compriSonoro && digitalRead(microfone)
== LOW) {
    // Verifica se o pulso recebido respeita o intervalo entre
1 pulso e outro
    if (tempoAposPalma < tempMinSom || tempoAposPalma >
tempMaxSom) {
      // Caso contrario o intervalo resetara a contagem e o
tempo
      contPalmas = 0;
      startTime = millis();
    }
    else {
      // Iniciada a contagem de pulso e o tempo
      contPalmas ++;
      startTime = millis();
    }
```

```
    // Verifica se a contagem de palma é igual ou superior ao
número...
    //esperado e se o LED esta desligado
    if ((contPalmas >= palmasRequeridaLigaLed - 1) &&
(digitalRead(led) == HIGH)) {
      // Acende o LED e zera a contagem
      digitalWrite(led, LOW);
      contPalmas = 0;
    }
    // Verifica se a contagem de palma é igual ou superior ao
número...
    //esperado e se o LED esta aceso
    if ((contPalmas >= palmasRequeridaDesligaLed - 1) &&
(digitalRead(led) == LOW)) {
      // Desliga LED e zera contagem
      digitalWrite(led, HIGH);
      contPalmas = 0;
    }
  }
}
```

APENDICE B

```
//CONEXAO ARDUINO INTERFONE
#include "SIM900.h"
#include <SoftwareSerial.h>
#include "call.h"

int microfone = 7; // Pino digital ligado ao sensor de som
(DO)
int rele = 8; // Pino ligado ao Relé
int contToque = 0; // Contador de Toques
int toqueRequeridaLigarele = 1; // Contagem para Ativar o Rele

unsigned long compriSonoro = 100; // Comprimento do som
esperado
unsigned long time;
unsigned long startTime = 0;
void setup() {
  pinMode(microfone, INPUT); // Inicia o pino do microfone
como entrada
  pinMode(rele, OUTPUT); // Inicia os pino do Rele como saída
  Serial.begin(9600);
  liga_desliga_GSMShield();
}
void loop() {
  // Inicia a contagem de tempo
  time = millis();
       }
       if ((contToque >= toqueRequeridaLigarele - 1) &&
(digitalRead(rele) == HIGH)) {

       //Chama a funcao de fazer a chamada
      fazchamada();
      // Conta 3 segundos e ativa o rele
      delay(3000)
      digitalWrite(rele, HIGH);
      ;
    }

  }
}
void fazchamada ()
CallGSM call;
void loop()
{
  //Efetua a chamada formato call.Call(<numero a ser chamado>)
  call.Call("987654321");
   delay(20000);

  call.HangUp();
    liga_desliga_GSMShield();
```

```
  do {} while (1); //Loop parando o programa
}
void liga_desliga_GSMShield()
{
 // Desativa o Rele
 digitalWrite(rele, LOW);
```

7. ANEXOS

ANEXO 1

```
#include "call.h"

/*****************************************************
Method checks status of call

return:
      CALL_NONE         - no call activity
      CALL_INCOM_VOICE  - incoming voice
      CALL_ACTIVE_VOICE - active voice
      CALL_NO_RESPONSE  - no response to the AT command
      CALL_COMM_LINE_BUSY - comm line is not free
*****************************************************/
byte CallGSM::CallStatus(void)
{
  byte ret_val = CALL_NONE;

  if (CLS_FREE != gsm.GetCommLineStatus()) return
(CALL_COMM_LINE_BUSY);
  gsm.SetCommLineStatus(CLS_ATCMD);
  gsm.SimpleWriteln("AT+CPAS");

  // 5 sec. for initial comm tmout
  // 50 msec. for inter character timeout
  if (RX_TMOUT_ERR == gsm.WaitResp(5000, 50)) {
    // nothing was received (RX_TMOUT_ERR)
    // -----------------------------------
    ret_val = CALL_NO_RESPONSE;
  }
  else {
    // something was received but what was received?
    // ---------------------------------------------
    // ready (device allows commands from TA/TE)
    // <CR><LF>+CPAS: 0<CR><LF> <CR><LF>OK<CR><LF>
    // unavailable (device does not allow commands from TA/TE)
    // <CR><LF>+CPAS: 1<CR><LF> <CR><LF>OK<CR><LF>
    // unknown (device is not guaranteed to respond to
instructions)
    // <CR><LF>+CPAS: 2<CR><LF> <CR><LF>OK<CR><LF> - NO CALL
    // ringing
    // <CR><LF>+CPAS: 3<CR><LF> <CR><LF>OK<CR><LF> - NO CALL
    // call in progress
    // <CR><LF>+CPAS: 4<CR><LF> <CR><LF>OK<CR><LF> - NO CALL
    if(gsm.IsStringReceived("0")) {
      // ready - there is no call
      // ------------------------
      ret_val = CALL_NONE;
    }
    else if(gsm.IsStringReceived("3")) {
```

```
      // incoming call
      // --------------
      ret_val = CALL_INCOM_VOICE;
    }
    else if(gsm.IsStringReceived("4")) {
      // active call
      // ------------
      ret_val = CALL_ACTIVE_VOICE;
    }
  }

  gsm.SetCommLineStatus(CLS_FREE);
  return (ret_val);

}

/*********************************************************
Method checks status of call(incoming or active)
and makes authorization with specified SIM positions range

phone_number: a pointer where the tel. number string of
current call will be placed
              so the space for the phone number string must be
reserved - see example
first_authorized_pos: initial SIM phonebook position where the
authorization process
                      starts
last_authorized_pos:  last SIM phonebook position where the
authorization process
                      finishes

                      Note(important):
                      ================
                      In case first_authorized_pos=0 and also
last_authorized_pos=0
                      the received incoming phone number is
NOT authorized at all, so every
                      incoming is considered as authorized
(CALL_INCOM_VOICE_NOT_AUTH is returned)

return:
      CALL_NONE                   - no call activity
      CALL_INCOM_VOICE_AUTH       - incoming voice -
authorized
      CALL_INCOM_VOICE_NOT_AUTH   - incoming voice - not
authorized
      CALL_ACTIVE_VOICE           - active voice
      CALL_INCOM_DATA_AUTH        - incoming data call -
authorized
      CALL_INCOM_DATA_NOT_AUTH    - incoming data call - not
authorized
```

```
      CALL_ACTIVE_DATA            - active data call
      CALL_NO_RESPONSE            - no response to the AT
command
      CALL_COMM_LINE_BUSY         - comm line is not free
*******************************************************/
byte CallGSM::CallStatusWithAuth(char *phone_number,
                                 byte first_authorized_pos, byte
last_authorized_pos)
{
  byte ret_val = CALL_NONE;
  byte search_phone_num = 0;
  byte i;
  byte status;
  char *p_char;
  char *p_char1;

  phone_number[0] = 0x00;  // no phonr number so far
  if (CLS_FREE != gsm.GetCommLineStatus()) return
(CALL_COMM_LINE_BUSY);
  gsm.SetCommLineStatus(CLS_ATCMD);
  gsm.SimpleWriteln("AT+CLCC");

  // 5 sec. for initial comm tmout
  // and max. 1500 msec. for inter character timeout
  gsm.RxInit(5000, 1500);
  // wait response is finished
  do {
    if (gsm.IsStringReceived("OK\r\n")) {
      // perfect - we have some response, but what:

      // there is either NO call:
      // <CR><LF>OK<CR><LF>

      // or there is at least 1 call
      // +CLCC: 1,1,4,0,0,"+420XXXXXXXXX",145<CR><LF>
      // <CR><LF>OK<CR><LF>
      status = RX_FINISHED;
      break; // so finish receiving immediately and let's go
to
             // to check response
    }
    status = gsm.IsRxFinished();
  } while (status == RX_NOT_FINISHED);

  // generate tmout 30msec. before next AT command
  delay(30);

  if (status == RX_FINISHED) {
    // something was received but what was received?
    // example: //+CLCC: 1,1,4,0,0,"+420XXXXXXXXX",145
    // ---------------------------------------------
```

```
    if(gsm.IsStringReceived("+CLCC: 1,1,4,0,0")) {
      // incoming VOICE call - not authorized so far
      // ----------------------------------------
      search_phone_num = 1;
      ret_val = CALL_INCOM_VOICE_NOT_AUTH;
    }
    else if(gsm.IsStringReceived("+CLCC: 1,1,4,1,0")) {
      // incoming DATA call - not authorized so far
      // ----------------------------------------
      search_phone_num = 1;
      ret_val = CALL_INCOM_DATA_NOT_AUTH;
    }
    else if(gsm.IsStringReceived("+CLCC: 1,0,0,0,0")) {
      // active VOICE call - GSM is caller
      // ---------------------------------
      search_phone_num = 1;
      ret_val = CALL_ACTIVE_VOICE;
    }
    else if(gsm.IsStringReceived("+CLCC: 1,1,0,0,0")) {
      // active VOICE call - GSM is listener
      // ---------------------------------
      search_phone_num = 1;
      ret_val = CALL_ACTIVE_VOICE;
    }
    else if(gsm.IsStringReceived("+CLCC: 1,1,0,1,0")) {
      // active DATA call - GSM is listener
      // ---------------------------------
      search_phone_num = 1;
      ret_val = CALL_ACTIVE_DATA;
    }
    else if(gsm.IsStringReceived("+CLCC:")){
      // other string is not important for us - e.g. GSM
module activate call
      // etc.
      // IMPORTANT - each +CLCC:xx response has also at the
end
      // string <CR><LF>OK<CR><LF>
      ret_val = CALL_OTHERS;
    }
    else if(gsm.IsStringReceived("OK")){
      // only "OK" => there is NO call activity
      // -------------------------------------
      ret_val = CALL_NONE;
    }


    // now we will search phone num string
    if (search_phone_num) {
      // extract phone number string
      // ---------------------------
      p_char = strchr((char *)(gsm.comm_buf),'"');
```

```
      p_char1 = p_char+1; // we are on the first phone number
character
      p_char = strchr((char *)(p_char1),'"');
      if (p_char != NULL) {
        *p_char = 0; // end of string
        strcpy(phone_number, (char *)(p_char1));
      }

      if ( (ret_val == CALL_INCOM_VOICE_NOT_AUTH)
           || (ret_val == CALL_INCOM_DATA_NOT_AUTH)) {

        if ((first_authorized_pos == 0) &&
(last_authorized_pos == 0)) {
          // authorization is not required => it means
authorization is OK
          // ------------------------------------------------
------------
          if (ret_val == CALL_INCOM_VOICE_NOT_AUTH) ret_val =
CALL_INCOM_VOICE_AUTH;
          else ret_val = CALL_INCOM_DATA_AUTH;
        }
        else {
          // make authorization
          // ------------------
          gsm.SetCommLineStatus(CLS_FREE);
          for (i = first_authorized_pos; i <=
last_authorized_pos; i++) {
            if (gsm.ComparePhoneNumber(i, phone_number)) {
              // phone numbers are identical
              // authorization is OK
              // ---------------------------
              if (ret_val == CALL_INCOM_VOICE_NOT_AUTH)
ret_val = CALL_INCOM_VOICE_AUTH;
              else ret_val = CALL_INCOM_DATA_AUTH;
              break;  // and finish authorization
            }
          }
        }
      }
    }

  }
  else {
    // nothing was received (RX_TMOUT_ERR)
    // -----------------------------------
    ret_val = CALL_NO_RESPONSE;
  }

  gsm.SetCommLineStatus(CLS_FREE);
  return (ret_val);
}
```

```
/*******************************************************
Method picks up an incoming call

return:
*******************************************************/
void CallGSM::PickUp(void)
{
  if (CLS_FREE != gsm.GetCommLineStatus()) return;
  gsm.SetCommLineStatus(CLS_ATCMD);
  gsm.SimpleWriteln("ATA");
  gsm.SetCommLineStatus(CLS_FREE);
}

/*******************************************************
Method hangs up incoming or active call

return:
*******************************************************/
void CallGSM::HangUp(void)
{
  //if (CLS_FREE != gsm.GetCommLineStatus()) return;
  gsm.SetCommLineStatus(CLS_ATCMD);
  gsm.SimpleWriteln("ATH");
  gsm.SetCommLineStatus(CLS_FREE);
}

/*******************************************************
Method calls the specific number

number_string: pointer to the phone number string
               e.g. gsm.Call("+420123456789");

*******************************************************/
void CallGSM::Call(char *number_string)
{
  if (CLS_FREE != gsm.GetCommLineStatus()) return;
  gsm.SetCommLineStatus(CLS_ATCMD);
  // ATDxxxxxx;<CR>
  gsm.SimpleWrite("ATD");
  gsm.SimpleWrite(number_string);
  gsm.SimpleWriteln(";");
  // 10 sec. for initial comm tmout
  // 50 msec. for inter character timeout
  gsm.WaitResp(10000, 50);
  gsm.SetCommLineStatus(CLS_FREE);
}

/*******************************************************
Method calls the number stored at the specified SIM position
```

```
sim_position: position in the SIM <1...>
              e.g. gsm.Call(1);
*********************************************************/
void CallGSM::Call(int sim_position)
{
  if (CLS_FREE != gsm.GetCommLineStatus()) return;
  gsm.SetCommLineStatus(CLS_ATCMD);
  // ATD>"SM" 1;<CR>
  gsm.SimpleWrite("ATD>\"SM\" ");
  gsm.SimpleWrite(sim_position);
  gsm.SimpleWriteln(";");

  // 10 sec. for initial comm tmout
  // 50 msec. for inter character timeout
  gsm.WaitResp(10000, 50);

  gsm.SetCommLineStatus(CLS_FREE);
}
```