CENTRO UNIVERSITÁRIO DO NORTE PAULISTA – UNORP
CURSO DE ENGENHARIA DA COMPUTAÇÃO

DANIEL CORDEIRO JACINTO
MATHEUS VINICIUS MENDONÇA ROSA

AUTOMAÇÃO RESIDENCIAL POR COMANDO DE VOZ UTILIZANDO ARDUINO E EASYVR.

SÃO JOSÉ DO RIO PRETO – SP
2014

DANIEL CORDEIRO JACINTO

MATHEUS VINICIUS MENDONÇA ROSA

AUTOMAÇÃO RESIDENCIAL POR COMANDO DE VOZ UTILIZANDO ARDUINO E EASYVR.

Trabalho de conclusão de curso apresentado ao curso de Engenharia da Computação do Centro Universitário do Norte Paulista – UNORP, como requisito parcial à obtenção do título de Bacharel.

Orientador: José Alexandre Ducatti

SÃO JOSÉ DO RIO PRETO – SP
2014

DANIEL CORDEIRO JACINTO

MATHEUS VINICIUS MENDONÇA ROSA

AUTOMAÇÃO RESIDENCIAL POR COMANDO DE VOZ UTILIZANDO ARDUINO E EASYVR.

Trabalho de conclusão de curso apresentado ao curso de Engenharia da Computação do Centro Universitário do Norte Paulista – UNORP, como requisito parcial à obtenção do título de Bacharel.

Aprovado em: _____/_____/_____.

Conceito: _____.

BANCA EXAMINADORA.

Prof. Esp. e Orientador José Alexandre Ducatti
Centro Universitário do Norte Paulista

Prof. Me. Flávio Henrique Fernandes Volpon
Centro Universitário do Norte Paulista

Prof. Esp. Ivan José dos Reis Filho
Centro Universitário do Norte Paulista

AGRADECIMENTOS

Eu Daniel Cordeiro, agradeço primeiramente a Deus, por colocar as pessoas certas em meu caminho que proporcionaram a minha chegada até aqui, por me fortalecer para que obtivesse vitória em minhas batalhas, por me capacitar para vencer todas as barreiras, me dar conhecimento e força de vontade. Aos meus abençoados tios João Reinaldo Caetano e Silvia Regina Cordeiro Caetano, que me receberam e me apoiaram para que eu realizasse esse grande sonho. Aos meus pais Suely Cordeiro Jacinto e José Carlos Jacinto por estarem sempre me apoiando, e sempre buscando o melhor para mim. Ao meu professor orientador José Alexandre Ducatti e a todos os outros professores que tiveram paciência e transmitiram seus conhecimentos durante esses anos com competência.

Eu Matheus Vinícius. Agradeço em primeiro plano à Deus, pois sem ele em nossas vidas não teríamos forças para batalhar por nossos ideais. Meus amados pais Djalma Correa e Linda Meire, pois sempre fizeram de tudo para que facilitar meus sonhos. Para minha namorada, que desde o primeiro ano está comigo Taina Franchi. E a todos professores, que além de ensinarem se tornaram amigos e conselheiros, mostrando o grande caminho de sucesso que ainda podemos trilhar.

RESUMO

Esta monografia tem por finalidade transmitir informações adquiridas através de pesquisas, para que se possa garantir noções sobre o assunto, e também explicar como foi a elaboração do projeto final que consiste em controlar equipamentos através de comandos de voz. Abrangendo conhecimentos e se aprofundando um pouco mais em assuntos como, automação, reconhecimento de voz, placas e componentes eletrônicos como o *Arduino*, *EasyVR*, microcontroladores, etc. O projeto final consiste em demonstrar uma forma possível de se automatizar uma casa, assim como outros lugares, utilizando a voz como controle de acionamento. Na proposta é demonstrado através de maquete com *leds*, *coolers* e motores, simulando luzes, ventiladores e motores de garagem respectivamente e, também, alguns eletrônicos de alta voltagem que podem ser acionados através de módulos reles.

Palavras – chave: *Arduino, EasyVR*, Automação, Controle por voz.

ABSTRACT

This monograph is intended to convey information acquired through research, so that we can ensure notions on the subject, and also explain the development of the final project, consisting of equipment control through voice commands. Covering knowledge and deepening a little more on issues such as automation, voice recognition, boards and electronic components such as Arduino, EasyVR, microcontrollers, etc. The final project is to demonstrate a possible way to automate a home, as well as other places, using the voice as drive control. In the proposed model is demonstrated through with LEDs, motors and coolers, simulating lights, fans and garage motors and also some high voltage electronics that can be triggered by relay modules.

Keywords: Arduino, EasyVR, Automation, Voice Control.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

A/D	Analógico/Digital
AD	Analog-to-Digital Converter (Conversor analogico para digital)
CALL	Chamada (Instrução em programação que chama outra instrução.
DIP	Dual In-Line Package (Dual pacotes em linha) (Tipo de encapsulamento).
DTMF	Dual Tone Multi-Frequency (Tons de duas frequências)
E/S	Entrada e Saída.
EEPROM	Electrically-Erasable Programmable Read-Only Memory (memória programável eletricamente apagável somente de leitura)
I/O	Input/Output (Entrada/Saída)
I²C	Inter-Integrated Circuit (Circuito Inter-integrado)
IDE	Integrated Development Environment (Ambiente de desenvolvimento integrado)
JMP	Instrução indicadora em programação (jump), pula para outra instrução.
LED	Light Emitting Diode (Diodo emissor de luz)
LCD	Liquid Crystal Display (Tela de Cristal líquido).
mA	Miliampere
MSSP	Master Synchronous Serial Port (Porta mestre serial sincrona)
NVRAM	Non-Volatile Random Access Memory (Memória não volátil de acesso aleatório).
OS	Operational System (Sistema Operacional)
PIC	Programmable Interrupt Controller (Controlador de Interface Programável).
PWM	Pulse-Width Modulation (Modulação por largura de pulso).
RAM	Random Access Memory (Memória de Acesso Aleatório)
RFID	Radio-Frequency IDentification (Identificação de Rádio Frequência)
RISC	Reduced Instruction Set Computer (Computador com um Conjunto Reduzido de Instruções).
RJ45	Registered Jack Type 45 (Conector modular usado em telecomunicações)
ROM	Read Only Memory (Memória de somente leitura)
RTC	Real-time clock (Relógio de tempo real)
SD	Standard Definition (Definição Padrão)
SPI	Serial Peripheral Interface (Interface Periférica Serial)

SRAM	Static Random Access Memory (Memória estática de acesso aleatório)
TWI	Two Wire Interface (Interface de dois fios)
UART	Universal Asynchronous Receiver/Transmitter (Receptor/Transmissor Universal Assincrono)
USART	Universal Synchronous/Asynchronous Receiver/Transmitter (Transmissor/Receptor Universal Síncrono e Assíncrono)
V	Volt
Wav	WAVEform audio format (Formato de áudio em forma de onda)

1 INTRODUÇÃO

Não é novidade que todos buscam maior comodidade e sossego nos dias de hoje, buscando sempre fazer menos movimentos possíveis, ter menos trabalho, pensar menos, enfim, ter mais conforto.

A tecnologia atualmente tem evoluído muito em questão de ajudar as pessoas no seu dia-a-dia, facilitando as tarefas ou até mesmo efetuando-as. As pessoas têm gostado dessas ideias e aderindo cada vez mais tecnologias em suas casas.

A automação residencial é um avanço muito bem vindo, pois traz praticidade e poupa tempo, que nos dias atuais é uma coisa muito valorizada, já que as pessoas cada vez mais se comprometem com algo.

Há várias formas de automação, mas neste trabalho enfatiza-se a automação por comandos de voz, onde um dispositivo capta os comandos ditos e os executa, sem que a pessoa precise fazer qualquer movimento que não seja o do maxilar.

Quem nunca se deitou em sua cama para dormir e se esqueceu de um equipamento ou a luz ligada, sem ter um interruptor por perto, e ter que levantar para desligá-lo. Isso é uma situação simples, mas que dá um certo estresse, principalmente depois de um dia cansativo.

Pense como seria útil ao invés de se levantar, apenas dizer: “apagar luz”, e ela rapidamente se apagar. Ou por exemplo, se você está com calor, poderia ligar seu ventilador ou ar-condicionado simplesmente falando ar. Em uma cozinha, preparando seu jantar com as mãos já envolvidas no alimento, você decide ouvir uma música, seria muito mais fácil apenas dizer para seu radio ligar do que ter que lavar suas mãos para poder tocá-lo.

Essas e outras situações são motivos de se buscar maneiras de se facilitar a vida e diminuir o estresse, abusando da tecnologia a nosso favor. Neste trabalho será abordado outras situações de automação e aprofundamentos de como isso é possível nos dias de hoje e como funciona essa ferramenta.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 AUTOMAÇÃO RESIDENCIAL OU DOMÓTICA

Automação residencial, atualmente também conhecida como domótica (que é a junção da palavra latina *domus* (casa) e do grego *automatika* (automática), não há dúvidas que esta nova tecnologia já está presente em nossa vida.

Não há como negar, que automação residencial veio para facilitar a vidas de todos, e com o desenvolvimento de novas tecnologias está ficando cada dia mais fácil integrá-las em nossas residências. Toma-se como exemplo tecnologias de automação através de comandos de voz e *wireless*. O comando de voz em si de certa forma é usado internamente nas residências, onde será possível acender e apagar luzes, ligar a TV e controlar a seleção de canais, volume e outros, pode-se controlar também ar-condicionado sem haver a necessidade de sairmos de onde estivermos, controlando o acionamento ou desligamento e até mesmo a temperatura, tudo isso através de comandos de voz pré-programados geralmente em microcontroladores, que por sua vez pode ser feito através de um arduino com um *shield easyvr* (que recebe comandos de voz), que será implementado no decorrer deste trabalho.

Já na automação residencial através do *wireless* sempre há uma interface gráfica para que haja a comunicação entre controlador e o microcontrolador, como por exemplo, *tablets*, *smartphones*, até mesmo controles universais desenvolvidos para o proposito da automação. Em *tablets* e *smartphones* sempre há o desenvolvimento de aplicativos específicos, alguns básicos desenvolvidos por usuários comuns, mas que tenham um pequeno conhecimento na área de *mobile*, pois as ferramentas encontradas hoje na internet facilitam muito a criação, podendo assim tornar-se de forma individual o aplicativo desenvolvido. E também têm-se os aplicativos desenvolvidos por empresas que trabalham diretamente com automação residencial, aqui os aplicativos já são desenvolvidos de forma que a interação “controlador-casa” seja facilitada, são aplicativos desenvolvidos para o controle total da residência, podendo algumas vezes serem alterados conforme o interesse do comprador.

Estes eletro portáteis com os aplicativos instalados, fazem com que a conexão seja feita através da internet, pois recebe comandos vindo dos *tablets* ou *smartphones* através de uma interface gráfica em aplicativos ou página *web*, gerando assim comandos que serão tratados no roteador e enviados para o microcontrolador em questão, que neste caso pode ser um arduino com *shield ethernet.*

Hoje pode-se afirmar que a domótica já está ao alcance de uma grande parcte da população, pois os preços estão mais acessíveis, de como por exemplo, na década de 90. O usuário que tem como interesse um ambiente agradável, seguro, bonito e com todas estas tecnologias, atualmente pode se obter facilidade.

Têm-se hoje aplicação de sistemas de automação residencial em praticamente tudo que é encontrado em um ambiente, integrando acionamentos e sempre buscando conforto e facilidade nos comandos. Tudo isso buscando o equilíbrio entre beleza e praticidade, podendo assim valorizar ainda mais o imóvel em questão.

Está tecnologia nos auxilia em fazer trabalhos que até então eram feitos de formas manuais, sejam eles executadas através de comandos de voz, controle remoto, celular, computadores de mesa ou *notebooks* ou até mesmo sem que haja a necessidade de uma ação humana, exemplo de dispositivos inteligentes equipados com sensores (luzes, ar condicionado e outros).

No brasil foi criada AURESIDE – Associação Brasileira de Automação Residencial, fundada em 2000 por um grupo de profissionais bem-sucedidos na área, para a normalização e instituição de normas básicas para que se possa desenvolver um bom trabalho na área de automação e que possa assim criar bons profissionais e ainda divulgar o que há de mais atual no mercado da tecnologia.

Segundo a AURESIDE, em 2013 aproximadamente 300 mil casas já são automatizadas no Brasil.

2.1.1 O Que Automatizar?

Iluminação: podendo acender e apagar luzes em determinados horários programados pelo usuário, a instalação de sensores de presença para que as luzes

acendam quando há alguém no ambiente, e assim que o mesmo se retirar, a luz venha a se apagar ou controlá-la de uma forma diferente da comum.

Sonorização: distribuir som para diferentes áreas da casa de acordo com a necessidade do usuário.

Climatização: ligar e desligar o ar-condicionado já não é um problema, podendo programá-lo pra que ligue e desligue em horários pré-definidos, ou até mesmo desligá-los no decorrer da noite quando atingir uma certa temperatura. Ainda existem os mais inteligentes, equipados com sensores que medem a temperatura do corpo, selecionando assim a temperatura ideal para o ambiente.

Controle do jardim: irrigação de quintais ou jardins, controlando o uso de água podendo assim haver mais economia.

Abertura de portas e portões: Abertura e fechamento de portas e portões, tornando mais prático o acesso, sensores de leitura *RFID* são uma boa opção.

Como foi descrito acima, a seguir na tabela 1 demonstra-se o crescimento da adoção destas novas tecnologias e o crescimento esperado para o próximo ano.

Tabela 1 - Evolução na adoção de algumas tecnologias

Tecnologia	2003	2004	2005	2006	2015(*)
Cabeamento estruturado	42%	61%	49%	53%	80%
Monitoramento de segurança	18%	28%	29%	32%	81%
Multiroom audio	9%	12%	15%	16%	86%
Home Theater	9%	8%	11%	12%	86%
Controle de iluminação	1%	2%	6%	8%	75%
Automação integrada	0	2%	6%	6%	70%
Gerenciamento de energia	1%	5%	11%	11%	62%

Fonte http://www.nahb.org

2.1.2 Por Que Automatizar?

Hoje a automação está sendo usada não só procurando conforto, segurança e praticidade, muitos investidores aplicam está nova tecnologia para que possam

valorizar o imóvel a ser vendido. Pode-se afirmar que atualmente a procura pela “casa inteligente” está cada dia maior, pois os novos proprietários que buscam seu primeiro imóvel, já procuram este tipo de tecnologia instalado em suas casas, para que no futuro não precise acontecer a famosa “quebradeira”, para conseguir o conforto e praticidade esperado.

Já as pessoas que já possuem um imóvel, atualmente procuram a automação na parte de iluminação e de segurança. Pois o gerenciamento da iluminação tem por finalidade a economia de energia [...] as luzes serão acessas e apagadas em horários programados ou assim que for necessário, podendo ser instalados sensores de presença, ligando-as quando houver alguma pessoa próximo a ela e sendo desligada no momento que a mesma se retirar local, ocasionando então uma vasta economia. Já no quesito segurança, os mais procurados são portas com senha e leitura biométrica, também podendo ser encontradas através de leitura da retina ou até mesmo por voz. Sensores já são encontrados em residências há anos, desde sensores de movimento ou até mesmo mecânicos (encontrados em cercas elétricas), que assim que acionados ou tocados ativam os alarmes sonoros.

2.1 HISTÓRIA DA AUTOMAÇÃO RESIDENCIAL

Pode-se considerar o ponto de partida para a automação residencial na década de 70, quando nos Estados unidos foram desenvolvidos os primeiros módulos “inteligentes”, os X-10, onde comandos eram enviados através da própria rede elétrica da residência, tratavam-se dos PLC (*Power Line Carrier*), que conseguiam ligar alguns aparelhos ou luzes remotamente, o que mais chamava a atenção nesta tecnologia era a praticidade pois não havia necessidade de mexer na infraestrutura elétrica da residência.

> *A automação residencial é originária da automação industrial que teve nos dispositivos CLPs (Controladores Lógicos Programáveis), datados da década de 60, uma grande revolução, graças aos avanços da microeletrônica. Muitas empresas de tecnologia migraram seu foco da automação industrial para residencial sem que, no entanto, percebessem as peculiaridades que cada um desses mercados demandava. Enquanto que na automação industrial é fundamental que os equipamentos operem*

> *com imunidade total a falhas, com respostas rápidas aos comandos e elevada precisão, na automação residencial essas condições podem ser afrouxadas. Por outro lado, a automação residencial exige equipamentos com um grau de acabamento superior, bem como interfaces muito mais amigáveis e intuitivas. SRA Engenharia.*

Não se pode deixar de citar que a descoberta da eletricidade tenha sido a mais importante descoberta para a automação, pois sem ela não teriam sido desenvolvidos vários dispositivos que hoje são encontrados na automação, como por exemplo os relés ou até mesmo os sensores, mas outras invenções e descobertas também foram essenciais para o desenvolvimento da domótica, como por exemplo os transistores, descobertos em 1947, já em 1977 foi lançado o primeiro microcomputador lançado pela *Apple* chamado *Apple II* e em 1989 a difusão da internet e do celular pelo mundo (SRA Engenharia).

E desde 2005 com a criação do arduino, surgiram vários projetos de automação, muitos deles utilizando o arduino com o auxílio de *Shields*, que foram desenvolvidos para facilitar no desenvolvimento dos projetos. Há vários tipos, mas os mais utilizados nos projetos são: *Bluetooth, ethernet*, comando de voz, entre outros.

Após 3 anos da criação do arduino teve-se no mercado o primeiro *smartphone* com sistema *Android* embutido, o *HTC Magic* ou *G1*, pode-se frisar que o desenvolvimento do Sistema Operacional *Android*, facilitou demais sobre a automação residencial, o que é mais se encontra hoje em dia são interfaces gráficas desenvolvidas em *android* atribuídas em *tablets* ou *smartphones*, aproximando ainda mais “casa-usuário”.

2.2 RECONHECIMENTO DE VOZ

Ao se ouvir o termo “reconhecimento de voz” é comum imaginar aquelas tecnologias super avançadas vistas em filmes em que são usados a voz como maneira de acesso a tal mecanismo ou lugar, também utilizadas em sistemas inteligentes que controlam todo o ambiente em que podem se relacionar com pessoas e receber seus comandos, como o *Jarvis* do filme *IronMan* ou a *Umbrella* do filme *Resident Evil.*

Mas se pararmos para observar bem, esta tecnologia de reconhecimento de padrão já está muito presente em nosso cotidiano. Um exemplo muito comum e que está ficando cada vez mais presente no dia-a-dia são os *smartphones* mais modernos, estes estão repletos de funções capazes de serem acessadas pelo simples ato de falar o que se deseja, como o *SVoice* da *Samsung* ou o *GoogleNow*. Há também aplicativos famosos que podem capitar o som de uma música que está sendo reproduzida no ambiente e conseguir lhe fornecer as informações dela, como nome e autor, por exemplo. Outra ocasião que quase todos já devem ter passado é o autoatendimento eletrônico, comumente implementados por operadoras de celular, em que se ouve uma gravação pedindo para se escolher a "opção desejada". Cruamente falando, estes sistemas capitam o som da fala, armazenam todos os dados colhidos, que são transformados em sinais digitais e fazem uma comparação com os padrões salvos no sistema em busca de semelhanças para se determinar quais palavras mais se aproximam e quais as funções a elas atribuídas.

Estas são poucas das várias aplicações usadas no dia-a-dia que utilizam reconhecimento de voz, por esses exemplos já se pode ter ideia de várias outras do nosso cotidiano.

A partir dessa questão se vem a curiosidade. Como computadores podem reconhecer a voz através da fala? Tudo é feito através do processamento dos sinais de voz. Essa questão será melhor compreendida na próxima seção.

2.2.1 Como é Feito o Reconhecimento De Voz.

Para se entender como é feito o reconhecimento de voz precisa-se estudar um pouco mais a natureza do som, do que ele é constituído e como é gerado.

"Sons consistem de variações na pressão do ar ao longo do tempo em frequências que podemos ouvir". (Carlos Alexandre Mello)

Assim como todo som é um sinal, pode-se plotar um sinal de voz através de forma de onda (como na figura 01 a seguir, gerada no *software Audacity*). Esse sinal na linha do tempo pode ser dividido em amostras subsequentes, pontos que variam na frequência do sinal, para serem melhor calculados.

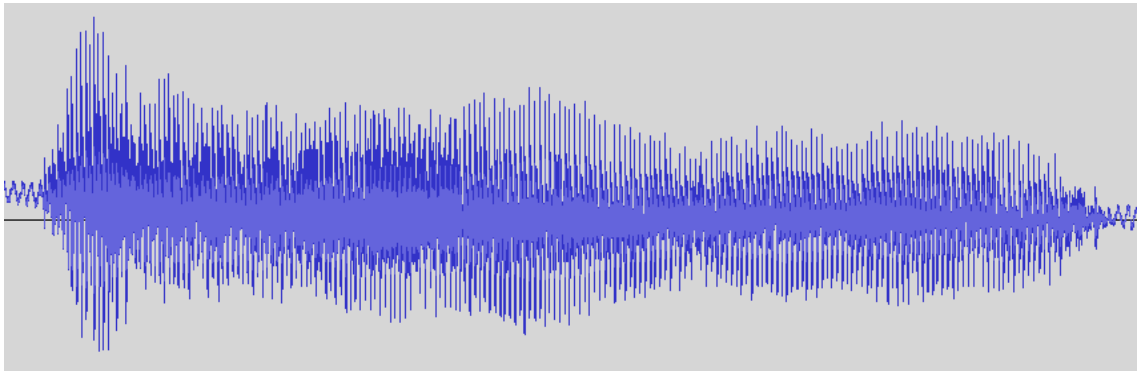

Figura 01- Sinal de voz da pronuncia da vogal "a" por alguns segundos

As palavras são constituídas por fonemas que são as informações fornecidas em um sinal, essa informação pode ser representada por um conjunto de símbolos concatenados finitamente ou seja discreta. Popularmente falando são os elementos mínimos criados pelo idioma, as menores representações de sons criados por nós para formar expressões com sentido. Há 34 fonemas presentes na nossa língua. Já outros idiomas podem ter variação dessa quantidade para mais ou para menos.

Para que se possa ser feito o processamento do sinal da voz no computador, ela precisa ser transformada de analógica para digital utilizando um conversor. Todos os dados, vibrações criadas no ar pela voz são captadas e convertidas em amostras digitais (forma de *bits*) que podem ser entendidas e calculadas pelo computador.

As ondas capitadas são divididas em amostras para facilitar sua filtragem, com no mínimo 8 mil amostras por segundo (taxa de *Nyquist*). Isso servirá para separar os ruídos e interferências dos sinais. O sinal após filtrado é gerado seu domínio espectral (frequências) com suas características próprias. Como geralmente as pessoas nem sempre falam com mesma velocidade ou mesmo tom de voz, um sinal precisa ser sincronizado com o outro que já está armazenado no banco de dados para que se possa ser comparado.

Após isso o sinal é dividido em partes ainda menores, na faixa de milésimos a centésimos de segundos, ou seja, sons fonéticos, menores que uma sílaba. Então o programa compara os fonemas capitados com os presentes no banco de dados para determinação do idioma falado.

Por último, o passo mais complexo, que é analisar os fonemas com os já presentes no banco de dados para tentar determinar a palavra dita pelo usuário. Este passo é o mais difícil por usar muita estatística, e fazer uma grande comparação com todas as palavras, frases e sentenças armazenadas. Após decidir qual mais se encaixou com o que o usuário deve ter dito ele transforma em texto ou comandos para serem efetuados.

Antigamente se era usado um método para reconhecimento que tentava aplicar conjuntos de regras gramaticais e sintáticas à fala, para comparar e ver em que conjunto ela se aproximava mais e assim tentava determinar a palavra dita. Mas este método baseado em regras não teve muito sucesso, pois a linguagem humana nem sempre segue as regras. Sotaques, gírias ou falar rápido juntando as palavras fez com que não fosse muito útil este tipo de reconhecimento. O sistema não era capaz de interpretar uma frase sem pausas, apenas reconhecia palavra por palavra, que tinham que serem ditas pausadamente.

Atualmente os softwares para este trabalho já são muito mais poderosos e complexos e usam funções de matemática e probabilidade para achar o resultado mais próximo do correto, esses são os chamados sistemas de modelo estatísticos.

Esses sistemas geralmente atribuem valores de probabilidade aos sons fonéticos e calculam qual o fonema mais provável a seguir na palavra baseando-se nas informações já contidas em seus dados. Mesmo assim a questão da compreensão de separar as palavras de uma frase ditas rapidamente continua complicada, assim tendo que analisar fonemas já ditos antes para que se possa distinguir.

Isso se faz complicado por haver muitas possibilidades de formação de palavras, por isso o programa precisa ser treinado e se adequar com o usuário, para isso é necessária uma ajuda. John Garofolo, gerente de fala do Laboratório de Tecnologia da Informação do *National Institute of Standards and Technology* (Instituto Nacional de Padrões e Tecnologia) disse o seguinte a respeito do treinamento necessário do programa:

> *Estes sistemas estatísticos precisam de muitos exemplos de treinamento para atingir um desempenho ótimo, chegando à ordem de milhares de horas de fala transcrita e centenas de megabytes de texto. Esses dados de treinamento são usados para criar modelos acústicos de palavras, listas de palavras e [...] redes de*

> *probabilidade de palavras múltiplas. Não deixa de ser arte a maneira como alguém seleciona, compila e prepara esses dados de treinamento para serem "digeridos" pelo sistema e como os modelos de sistema são ajustados para um aplicativo específico. E são esses os detalhes que podem fazer a diferença entre um sistema de bom desempenho e um de mau desempenho, mesmo se usam o mesmo algoritmo básico. Garofolo(2008).*

Além dos criadores desses softwares treinarem os programas, também se faz necessário um pequeno treino com os usuários finais para que o programa possa se adequar com os padrões específicos do usuário (voz dos usuários). Programas específicos para empresas que usam certas palavras usadas somente naquela área tendem a já se apresentarem com esses termos treinados, como por exemplo, na medicina ou no direito.

Mesmo com toda tecnologia aplicada, os sistemas de reconhecimento estão longe de estarem plenamente satisfatórios. Existem vários problemas presentes e que ainda impedem o funcionamento pleno. Ruídos gerados no ambiente, ou mesmo dos equipamentos, falas múltiplas, muitas pessoas falando no mesmo ambiente dificultam a compreensão do sistema com a pessoa que o utiliza, palavras que, quando ditas tem o mesmo som, mas têm significados diferentes e também devido ao alto uso de complexidade desses modelos estatísticos, são necessários hardwares de potência elevada para essas comparações, o que pode fazer os computadores mais rápidos terem dificuldades com frases ou comandos mais complicados.

A evolução no comando de voz está cada vez mais avançada, não é de se duvidar que chegue não só na sua plenitude mas que também deem um passo adiante, compreendendo a voz, assim tornando possível conversar com nossos computadores.

2.3 EASYVR

O componente que será utilizado neste trabalho com a função de reconhecer a voz como comando e emitir sons como resposta é o *shield* para *Arduino* composto com um módulo *EasyVR* que pode ser visto na figura 02.

A forma de como isso ocorre já foi tratada e mais detalhada anteriormente no tópico sobre reconhecimento de voz. Aqui é possível visualizar mais sobre este componente.

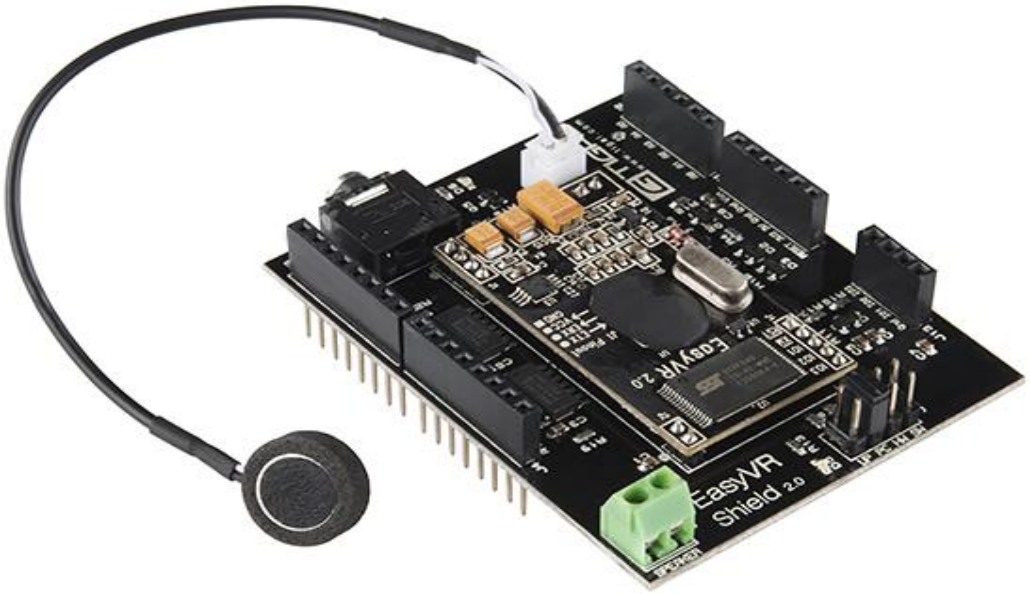

Figura 02 - Shield EasyVR para Arduino.
Fonte: https://www.sparkfun.com/products/12656

A capacidade no reconhecimento da voz, a robustez e a versatilidade fazem do módulo *EasyVR* muito vantajoso de se usar, pois garante um ótimo custo benefício.

O *EasyVR* possui conectores para microfone, saída para alto-falante de 8 *ohms* e saída p2.

Segundo o site Labdegaragem (2014), “é um protocolo simples e robusto (padrão 9600 8-N-1), para acessar suas funções pode ser utilizado uma placa *arduino*. O *EasyVR [...]* tipicamente consome 12mA de corrente em operação.”

Para configurá-lo é necessário o *software EasyVR Commander*. Nele estão disponibilizados vários grupos em que é possível gravar nossa própria voz para se ter um reconhecimento próprio, precisando de um pequeno treino. Outra opção é onde já se encontram várias palavras definidas como comando de voz em que será reconhecido por qualquer pessoa que pronunciá-las, está ferramenta chama-se

Wordset. Há também a possibilidade de se fazer uploads de áudios que podem ser usados posteriormente como respostas aos comandos.

Segundo o site Robocore (2014) as características do *EasyVR* são:

- 28 comandos independentes (SI) prontos para comandos básicos (em inglês, italiano, japonês, alemão, espanhol e francês).
- Suporta até 32 comandos principais (SD) em qualquer linguagem, bem como senhas via voz.
- *SonicNet* - Controle um ou mais *EasyVR* 2.0 de maneira sem fio através de tons gerados pelos módulos ou outra fonte de som.
- Geração de tom DTMF.
- Fácil de usar e possui interface gráfica simples.
- O módulo pode ser usado e acessado via interface UART (tensão de operação: de 3,3V a 5V).
- Protocolo serial simples e robusto para acessar e programar o módulo.
- Faça suas próprias tabelas de som utilizando a ferramenta *Sensory QuickSynthesis5.*
- O novo *EasyVR GUI* inclui um comando para processar e baixar tabelas de sons customizadas (sobrescrevendo as tabelas existentes).
- Conector para entrada de microfone.
- Saída de áudio que suporta alto falantes de 8 ohms.
- Entrada para fone de ouvido.
- Acesso aos pinos de entrada e saída do *EasyVR*.
- *LED* programável mostra o feedback durante o uso.
- Bibliotecas para *Arduino* disponíveis.
- Dimensões: 45 x 24 mm

2.4 ARDUINO

Arduino é uma placa eletrônica de prototipagem *open-source*, ou seja, de código aberto, qualquer um pode copiá-la ou modificá-la. Ela tem sua própria linguagem de programação baseada na linguagem *wiring* que é uma ferramenta

também *open-source* de programação de microcontroladores, o que facilita seu uso. Este por sua vez pode ser baixado gratuitamente (ARDUINO, 2014).

Esta ferramenta é muito útil e tem uma amplitude de usabilidade muito extensa. Utilizando-se de sensores e atuadores pode se fazer diversos projetos, o limite é a imaginação. Sua invenção foi uma grande inovação para quem gosta de criar projetos e poupar tempo não tendo que programar *pics* e fazer as placas do circuito.

A ideia de criação surgiu para tentar facilitar o estudo na área de eletrônica para alunos de *design* para que pudessem trabalhar em seus projetos. Outra dificuldade era que não se encontrava uma placa poderosa e barata para ser usada. Foi feita basicamente em poucos dias, e logo virou uma febre entre os estudantes. Uma breve história de seu surgimento, retirada do livro Arduino em Ação, escrita por Martin Evans, Joshua Noble e Jordan Hochenbaum, pode ser lida a seguir:

> *"O Arduino teve seu início no Interaction Design Institute na cidade de Ivrea, na Itália, em 2005. O professor Massimo Banzi procurava um meio barato de tornar mais fácil para os estudantes de design trabalhar com tecnologia. Ele discutiu seu problema com David Cuartielles, um pesquisador visitante da Universidade de Malmö, na Suécia, que estava procurando uma solução semelhante, e o Arduino nasceu. Os produtos existentes no Mercado eram caros e relativamente difíceis de usar. Banzi e Cuartielles decidiram desenvolver um microcontrolador que poderia ser utilizado pelos seus estudantes de arte e design em seus projetos. As principais exigências eram que fosse barato – o preço almejado não poderia ser mais do que o que um estudante gastaria se saísse para comer uma pizza – e que fosse uma plataforma que qualquer pessoa pudesse utilizar. David Cuartielles desenhou a placa, e um aluno de Massimo, David Mellis, programou o software para executar a placa. Massimo contratou um engenheiro local, Gianluca Martino, que também trabalhou no Design Institute ajudando alunos com seus projetos." (Livro: Arduino em Ação)*

"Fabricado na Itália, com projeto iniciado em 2005, os criadores do *Arduino* foram: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis." (ARDUINO, 2014).

Existem várias versões já criadas do *Arduino*, como o uno, mega, nano, Leonardo, entre outras. São equipadas com microprocessadores da família Atmel. Cada uma com características e especificações diferentes para serem escolhidas

conforme melhor adequação aos protótipos criados, variando em formato, desempenho, etc.

Além das placas, foram criados os *Shields* para elas, que são componentes que se encaixam nas placas para atender certa necessidade de conexão com as mesmas. Por exemplo, *Shield* de *Ethernet*, onde se conecta o cabo RJ45, *Shield Bluetooth* para comunicação via tecnologia sem fio *Bluetooth*, o *Shield EasyVR* que é utilizado no projeto deste trabalho, este reconhece a voz captada por um microfone e também pode transmitir voz ou sons através de um alto-falante conectado a ele.

A figura 03 demonstra a placa *Arduino Uno*.

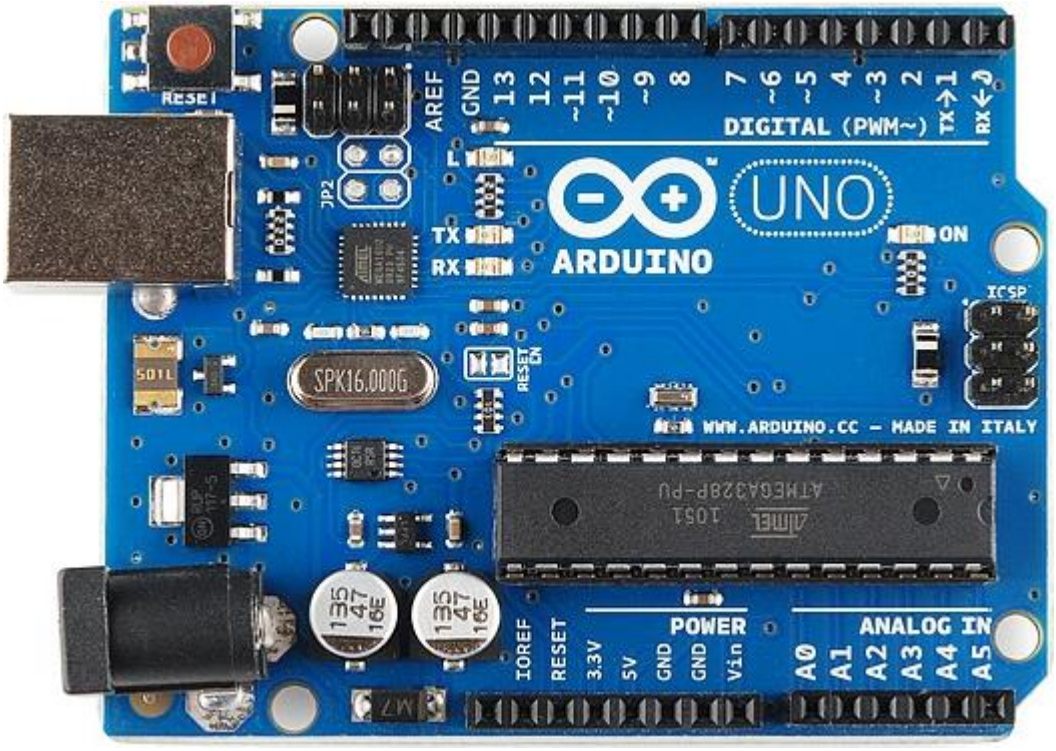

Figura 03 - Placa Arduino UNO.
Fonte: http://multilogica-shop.com/Arduino-Uno-R3

A placa e demais circuitos funcionam com tensões entre 5 e 3,3 volts. Embutido no *Arduino* há ainda um *firmware* – que combina memória ROM para leitura e um programa gravado neste tipo de memória – carregado na memória da placa controladora, que aceita *Windows, Linux* e *Mac OS X*. (TECHTUDO, 2013).

Para que os códigos programados na IDE do *arduino* (*sketchs*) sejam executados por ele é necessário um *software* no seu microcontrolador, mais

precisamente na sua memória *flash*, este software é o *bootloader*, que já vem gravado na *flash* do arduino. Ele é pré-executado quando se vai gravar um outro código, ou seja, ele é o primeiro *software* executado após o *reset* (*boot*), então recebe pela *serial* o novo *software* a ser executado e o carrega na *flash* (*loader*).

Utilizando a IDE do arduino é possível desenvolver softwares que poderão serem executados pelo dispositivo. Com isso abrange-se as formas de automação utilizando a placa, como a em casas, acionando luzes e equipamentos, entre outros projetos mais avançados.

2.5 MICROCONTROLADORES

Controlador embutido, muitas vezes pode-se encontrar microcontroladores sendo chamados assim. Os microcontroladores são computadores de propósito específicos, o programa (tarefa) é gravado em sua memória, geralmente não muda, pois assim que são produzidos, são distribuídos para determinadas áreas da eletrônica, ou seja já são de utilização específica.

Desenvolvidos para executar tarefas pré-definidas, podem ser encontrados atualmente embutidos em vários dispositivos, onde obtém-se a entrada do dispositivo que está controlando e o controla enviando sinais a diferentes componentes do mesmo, como por exemplo, uma TV, pois encontra-se microcontroladores no seu controle remoto e internamente dentro da TV. Com o controle é possível fazer várias alterações, como por exemplo, controlar volume, programá-la para que venha ligar e desligar em determinados horários, etc. Já em carros pode-se encontrar microcontroladores de forma mais comum, como nos sistemas de carburação e injeção, controlando assim os sensores de oxigênio e detonação e de mistura de combustível.

Em um micro-ondas por sua vez têm-se como entrada os botões que já possuem toda uma programação para cada tecla que se possa clicar, como por exemplo, o botão carne que, de um micro-ondas para outro, pode variar o tempo de 12 min à 15 min, pois já é um botão pré-configurado para que, por exemplo, descongele um pacote de carne, nele também tem-se uma resposta de saída feita

através do visor de LCD, em que é possível demonstrar o tempo controlando e sendo assim controlar o relé que desliga o aparelho.

Atualmente, microcontroladores são muito utilizados, devido ao baixo consumo e sua capacidade satisfatória de processamento, para os sistemas onde são embutidos.

Não há dúvidas que a maior vantagem em se usar o microcontrolador é devido ao baixo consumo, pois além de ter seu consumo na casa de *miliwatts*, ainda possui uma particularidade interessante que é o modo de espera (*Sleep* ou *Wait*), aguardando uma entrada de um comando, seja um pressionamento de uma tecla, ou seja por uma interface de dados. Já no modo de espera é possível ter como consumo dos microcontroladores na casa de *nanowatts*, está habilidade é o que mais chama a atenção dos eletrônicos que os utilizam no mercado hoje em dia.

Já que a ideia central dos microcontroladores não é um grande poder de processamento e sim aplicações especificas, os microcontroladores têm como frequência de *clock* na casa de MHz (*Megahertz*) ou até menos, mas para controlarmos, como por exemplo, uma lava roupas ou uma esteira, são mais do que ideais, pois não há necessidade de uma alta frequência de *clock.*

A figura 04 mostra a estrutura de um microcontrolador genérico, geralmente a estrutura mais utilizada em microcontroladores.

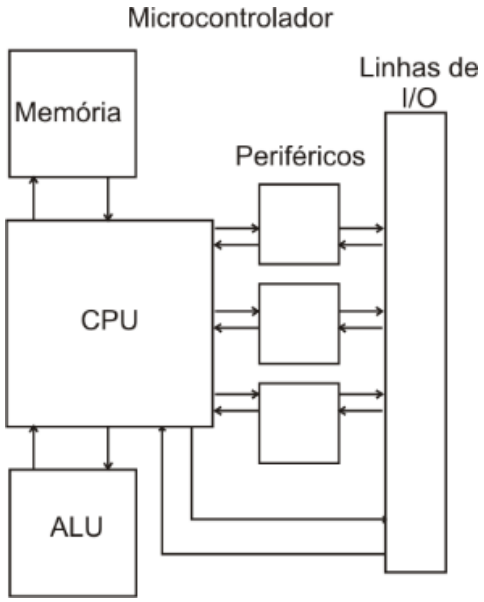

Figura 04 - Estrutura de um microcontrolador genérico.
Fonte: http://www.arnerobotics.com.br/eletronica/Microcontrolador_PIC_teoria_1.htm

O fato que mais chama a atenção, sobre a estrutura dos microcontroladores é a simplicidade, onde:

2.5.1 **CPU** ou “Unidade de processamento central” tem a responsabilidade de processar os dados contidos no microcontrolador e é aqui onde são interpretados os comandos e leitura de dados, para ativar portas I/O’s ou outros periféricos, caso necessário.

2.5.2 **ALU** (Unidade de lógica e aritmética) responsável por realizar os cálculos que envolvem os registros e/ou cálculos lógicos que servem para tomadas de decisões, no microcontrolador ela é controlada e ligada pela CPU.

2.5.3 **Memória** são encontradas em 3 tipos (RAM – *Random Acess Memory*, ROM – *Read-Only Memory*, Hibridas – ex *Flash*, NVRAM, EEPROM). As mais utilizadas são as memórias ROM, mesmo quando não há alimentação elétrica a memória ROM consegue reter dados, diferentemente da memória RAM, já as memórias híbridas, como o próprio nome já pressupõe, é o que há de melhor na memória RAM e na ROM, a particularidade de ser escrita e reescrita ela absolve da RAM, já a habilidade de reter dados, mesmo depois de desligada, isso foi adquirido da memória ROM.

2.5.4 **I/O’s** – No microcontrolador as linhas de I/O são responsáveis pela “entrada” e “saída” do mesmo. É aqui que é feita a ligação entre o microcontrolador e onde ele está embutido, pois aqui pode-se colher dados e, por estas mesmas vias de entrada de dados, é possível também respondê-las.

2.5.5 **Periféricos** são circuitos que ajudam a uma comunicação mais fácil entre microcontrolador e controlador. Eles variam de acordo com a necessidade como, por exemplo, no micro-ondas há como periféricos, o teclado com várias teclas já programadas para executar determinadas tarefas e também o visor LCD onde pode-se visualizar a tarefa que foi executada.

Os periféricos mais comuns nos microcontroladores são: USART’s, A/D, Portas I2C e SPI, *Timer’s, Watchdog Timer* e Osciladores.

2.6 MICROCONTROLADORES ATMEGA

Neste tópico é possível conhecer um pouco mais do “cérebro” de um *arduino*, que é seu microcontrolador. No caso do *Arduino* Uno, o microcontrolador usado é o ATmega328 da Atmel.

Este microcontrolador trabalha em 8 *bits* e tem uma arquitetura *Harvard* modificada. Pertencente à família AVR da Atmel onde todos os modelos dessa família compartilham um conjunto de instrução básicas e arquitetura, são eles, os grupos TinyAVR (microcontroladores ATtiny), megAVR (os ATmega) e XMEGA (os Atxmega).

Na figura 05 está ilustrado um microcontrolador ATmega328.

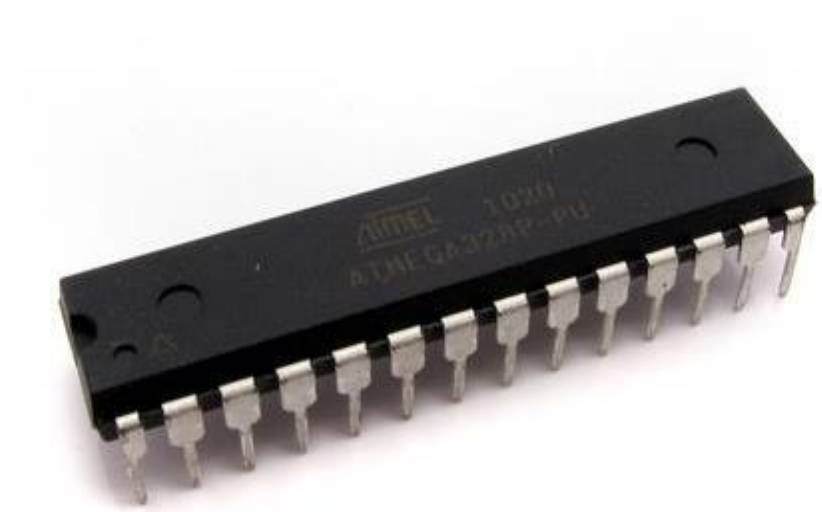

Figura 05 - O microcontrolador Atmega328.
Fonte: Anúncio Mercado Livre

A evolução dos microcontroladores Atmel usados no arduino pode ser notada quando analisada sua expansão na memória *flash*. Os primeiros *arduinos* usavam um microcontrolador Atmega8, que possuía 8k de memória *flash*, logo após vem o Atmega168 com 16k na memória e com mais recursos de entrada e saída, posteriormente o Atmega328 possuindo 32k de memória *flash* sendo 2k usados pelo *bootloader*. Os três modelos com a versão DIP possuem a mesma pinagem com 28 pinos, sendo que o Atmega168 e o Atmega328 podem ter usos

diferenciados nos pinos. O 328 possui 23 pinos de I/O sendo 6 canais para ADC de 10 *bits* e executam até 20Mhz com cristal externo. Já o arduino Mega usa uma versão ainda mais evoluída, equipado com um microcontrolador Atmega2560 possuindo 256k de memória *Flash* e com capacidade de entrada e saída muito mais elevada.

Mais algumas características do Atmega328:

- EEPROM Memória de dados: 1 *kbytes*
- SRAM Memória de dados: 2 *kbytes*
- Pinos I/O: 23
- *Timers*: Dois 8-*bit* / Um 16-*bit*
- Conversor A/D: 10-bit Seis Canais
- PWM: Seis Canais
- RTC: Sim com oscilador separado
- MSSP: SPI e I²C *Master* e *Slave* Suporte
- USART: Sim
- Oscilador externo: até 20MHz
- Tensão: 5 Volts
- Velocidade de *Clock*: 16Mhz
- Temperatura de funcionamento contínuo MIN: -40ºC a 85ºC.

O Atmega328 com encapsulamento DIP pode receber o *Bootloader* do *arduino* assim sendo possível programá-lo na IDE do *arduino* ou também poderá ser programado em linguagem C ou *Assembly*.

A seguir, na figura 06 a referência de pinagem do Atmega328.

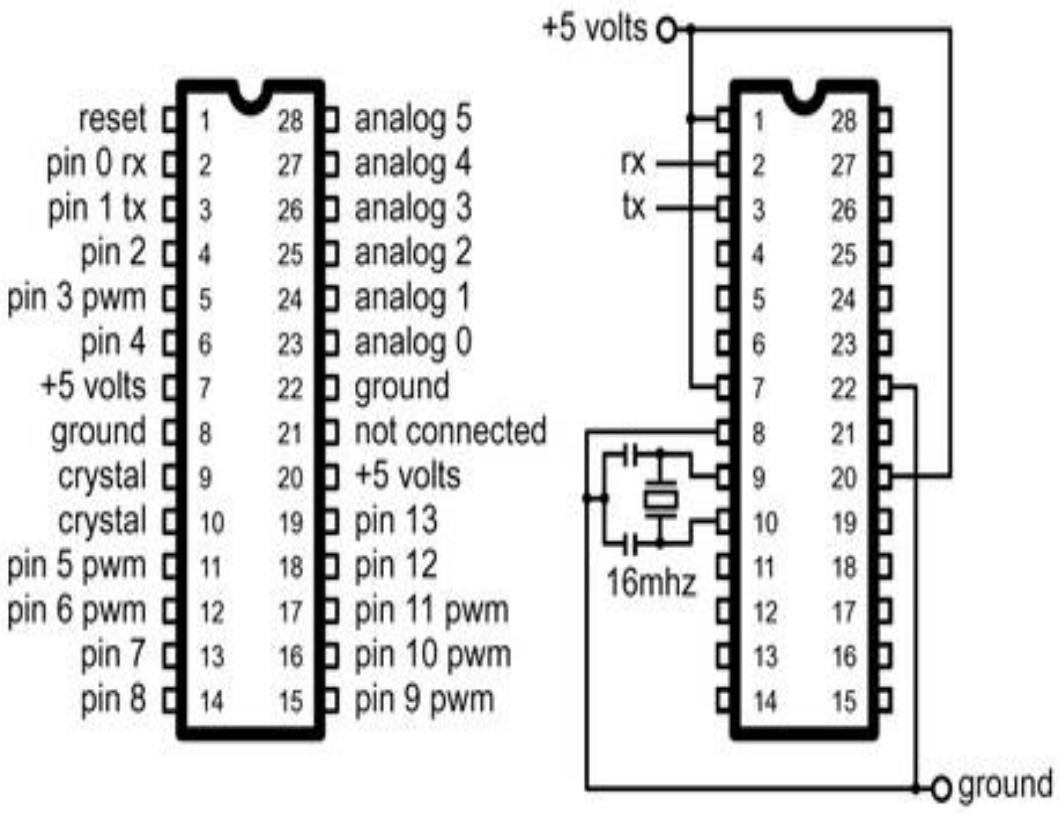

Figura 06 - Referência de Pinagem ATmega328.
Fonte: Anúncio Mercado Livre

Para um melhor entendimento do funcionamento do microcontrolador Atmel328 é possível analisar seu diagrama com seus principais blocos, em que foi extraído de seu *datasheet*. Esta análise pode ser observada na figura 7 logo abaixo:

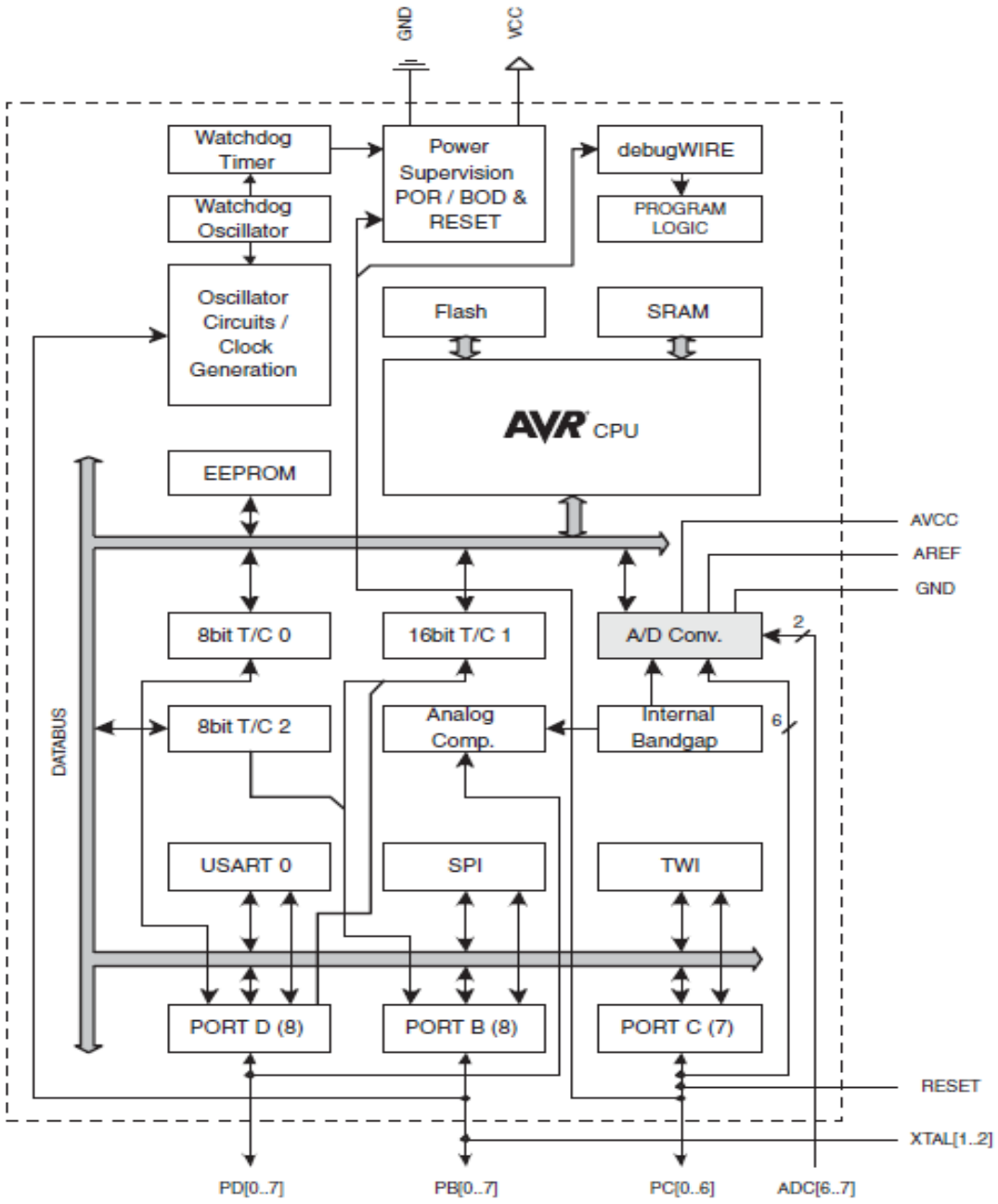

Figura 07 - Diagrama de blocos ATmega328.
Fonte: DQSOFT

É possível notar que a memória *Flash* e SRAM estão ligadas a CPU separadamente, este tipo de ligação (vias de dados separadas para programa e dados) é característico de uma arquitetura *Harvard*. O que torna os modelos da família AVR a serem considerados a utilizarem arquitetura *Harvard* modificada é que, nos modelos as duas vias têm-se 8 bits de largura e a memória *Flash* pode armazenar dados constantes. Porém, apenas as instruções que estiverem contidas

na *Flash* poderão ser executadas, ou seja, o código que estiver na SRam não poderá ser executado.

Este microcontrolador possui também a memória do tipo EEProm, assim como outros microcontroladores AVR. Mas essa memória não é acessada normalmente pelas instruções de acesso à memória, pois, ela está conectada na via de conexão aos periféricos.

Também nota-se no diagrama as portas de Entrada e Saída Digital (Portas D, C, B), os três *timers* (TC0 e TC2 de 8 *bits* e o TC1 de 16 *bits*), o conversor Analógico Digital (A/D Conv.), o comparador analógico e as interfaces seriais SPI, TWI (compatível com I2C) e USART. (DQSOFT, 2011)

Vamos conhecer um pouco mais do principal componente de um microcontrolador, a CPU. O Atmega328 utiliza uma AVR do tipo "*enhanced core*", pode-se ver a arquitetura na figura 08 logo abaixo, retirada do site DQSOFT.

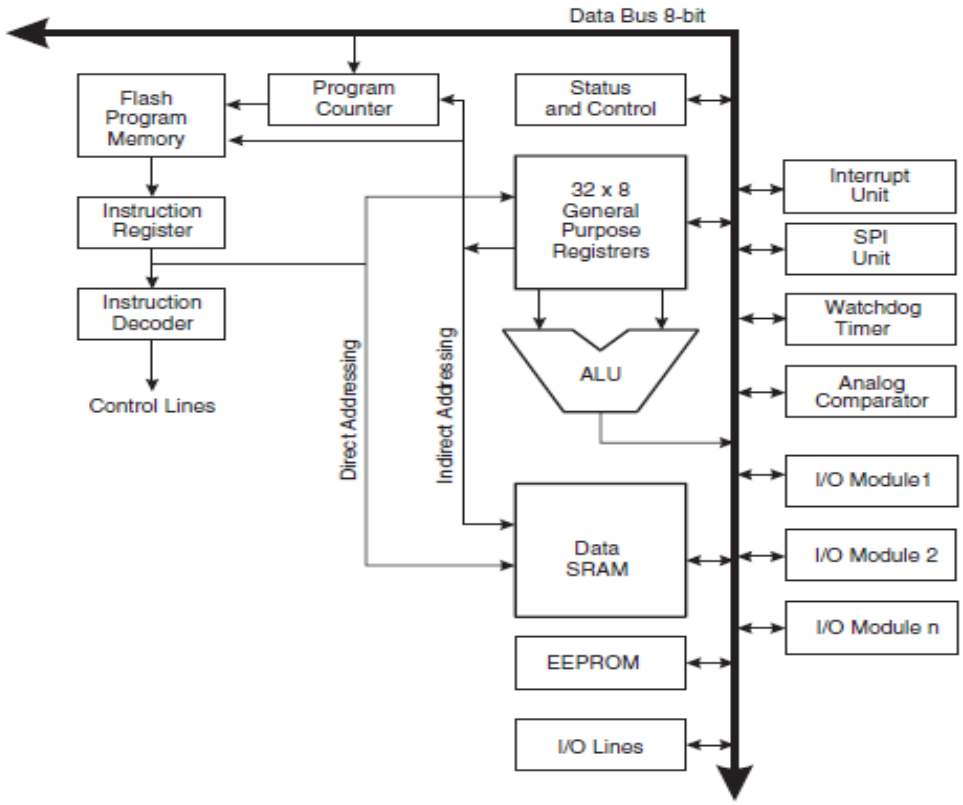

Figura 08 - Arquitetura CPU AVR do Atmega328.
Fonte: DQSOFT

A família AVR possui uma generosa coleção de 32 registradores de uso geral, todos de 8 *bits*. Os seis últimos registradores podem ser usados aos pares

como três registradores de 16 *bits* (X, Y e Z) para endereçamento indireto da memória. (DQSOFT, 2011)

A Unidade Lógica Aritmética (ALU) trabalha com 8 *bits*; obtêm os operandos dos registradores e coloca o resultado no primeiro operando (exceto na multiplicação). Todas as operações lógicas e aritméticas são executadas em um ciclo, exceto pela multiplicação que demora dois ciclos. (DQSOFT, 2011)

A composição da memória interna dessa CPU pode ser definida pelos registradores de uso geral, os registradores de E/S nas 64 posições seguintes, com o início do endereçamento da Sram no 0x60.

Têm-se as seguintes instruções de endereçamento, descrita por Daniel Quadros no site DQSoft 2011,

- Endereçamento direto a registrador: usado nas operações lógicas e aritméticas; os números dos registradores usados estão na instrução.
- Endereçamento direto a E/S: o endereço do registrador de E/S está na instrução.
- Endereçamento direto a dados: um endereço de memória de 16 bits faz parte da instrução.
- Endereçamento indireto: um dos registradores X, Y ou Z contém o endereço da memória. Existem variações deste endereço com pré-decremento ou pós-incremento do registrador.
- Endereçamento indireto com deslocamento: o endereço da memória é a soma do conteúdo do registrador Y ou Z com um deslocamento de 0 a 63 contido na instrução.
- Endereçamento indireto da memória de programa: usado nas operações de leitura da *Flash*, usam o endereço no registrador Z; uma variante deste modo realiza o pós-incremento do registrador.
- Endereçamento direto da memória de programa (para desvio): usado nas instruções JMP e CALL, um endereço de 22 *bits* faz parte da instrução.

- Endereçamento indireto da memória de programa (para desvio): usado nas instruções IJMP e ICALL, usa o registrador Z para obter um endereço de 16*bits*.
- Endereçamento relativo da memória de programa (para desvio): a instrução contém um deslocamento de 11 *bits* (com sinal) que é somando ao contador de programa.

Adotando a filosofia RISC os microcontroladores da AVR têm uma grande flexibilidade de escolha de registradores que são operados por instruções lógicas e aritméticas, sendo um conjunto de 131 instruções no Atmega328.

O arduino disponibiliza quase todos os pinos do 328 para que se encaixe *shields*, e como um bom microcontrolador possui uma diversidade de periféricos internos.

No *arduino* pode se usar as portas analógicas como portas digitais, totalizando 23 dos 28 pinos do ATmega328 disponíveis para entrada e saída digitais. Pode-se alternar o uso dos pinos, possuindo outras funções, por exemplo, alguns pinos do *arduino* são usados para funções, como, XTAL1, XTAL2 e *Reset*, assim deixando esses pinos exclusivamente para entrada e saída digital.

Outra opção dos 23 pinos digitais do *arduino*, é poder configurá-los (independente se foram setados para entrada ou saída) para aceitar interrupções quando acontece mudanças de sinais. Cada interrupção pode ser controlada pino a pino.

A memória *Flash* do ATmega328 é onde se armazena os códigos a serem executados e esse tipo de memória é a encontrada em cartões de memória, *pen drives*, etc. Elas são não voláteis, ou seja, não precisam de alimentação para manter guardada as informações e podem ser alteradas quando necessário. Não tão rápida quanto a memória RAM, memórias *flash* não precisam de muitos cuidados, mas a sua gravação é um pouco mais complexa. Para se escrever é necessário apenas a mudanças de bits de 1 para 0, mas para reverter os *bits* é necessário a função de apagamento, isso é uma operação que afeta várias posições da memória e se torna bem lenta. O problema é que essas operações, escritas e apagamentos, são limitadas, no caso do ATmega328 são garantidas pela Atmel 10000 mudanças.

3 DESENVOLVIMENTO

O projeto tem a intenção em demonstrar como é possível automatizar elementos do nosso cotidiano utilizando apenas a pronuncia de palavras. Para isso será utilizado um módulo de reconhecimento de voz, que neste caso será o *EasyVR*, responsável por receber os comandos de voz e convertê-los para sinal digital e enviá-los à uma placa de prototipagem que é utilizada para programar um microcontrolador responsável por emitir os comandos eletrônicos recebidos para os componentes a serem automatizados, este por sua vez será a placa *Arduino UNO R3.*

Através de uma maquete poderá ser demonstrado e exemplificado como uma casa pode ser automatizada. Os elementos da maquete simulam equipamentos desta casa ou até mesmo os próprios equipamentos eletrônicos, tais como uma lâmpada ou um ventilador. É demonstrado neste trabalho também a função do *EasyVR* de responder aos comandos com áudios pré-gravados na placa.

O projeto divide-se em etapas que são, desde a montagem da maquete, a gravação de vozes na placa *EasyVR* e programação do *Arduino*. Estas etapas serão abortadas no decorrer deste capitulo.

3.1 A Montagem

A montagem da maquete foi feita em madeira, sem muitos detalhes, simplesmente para simular o aspecto de uma casa. Foram feitas adaptações para receber lâmpadas, *cooler*, leds, servo motor, fiações, e outros componentes necessários.

Na figura 09 a seguir é apresentada a maquete ainda crua, sem os componentes instalados.

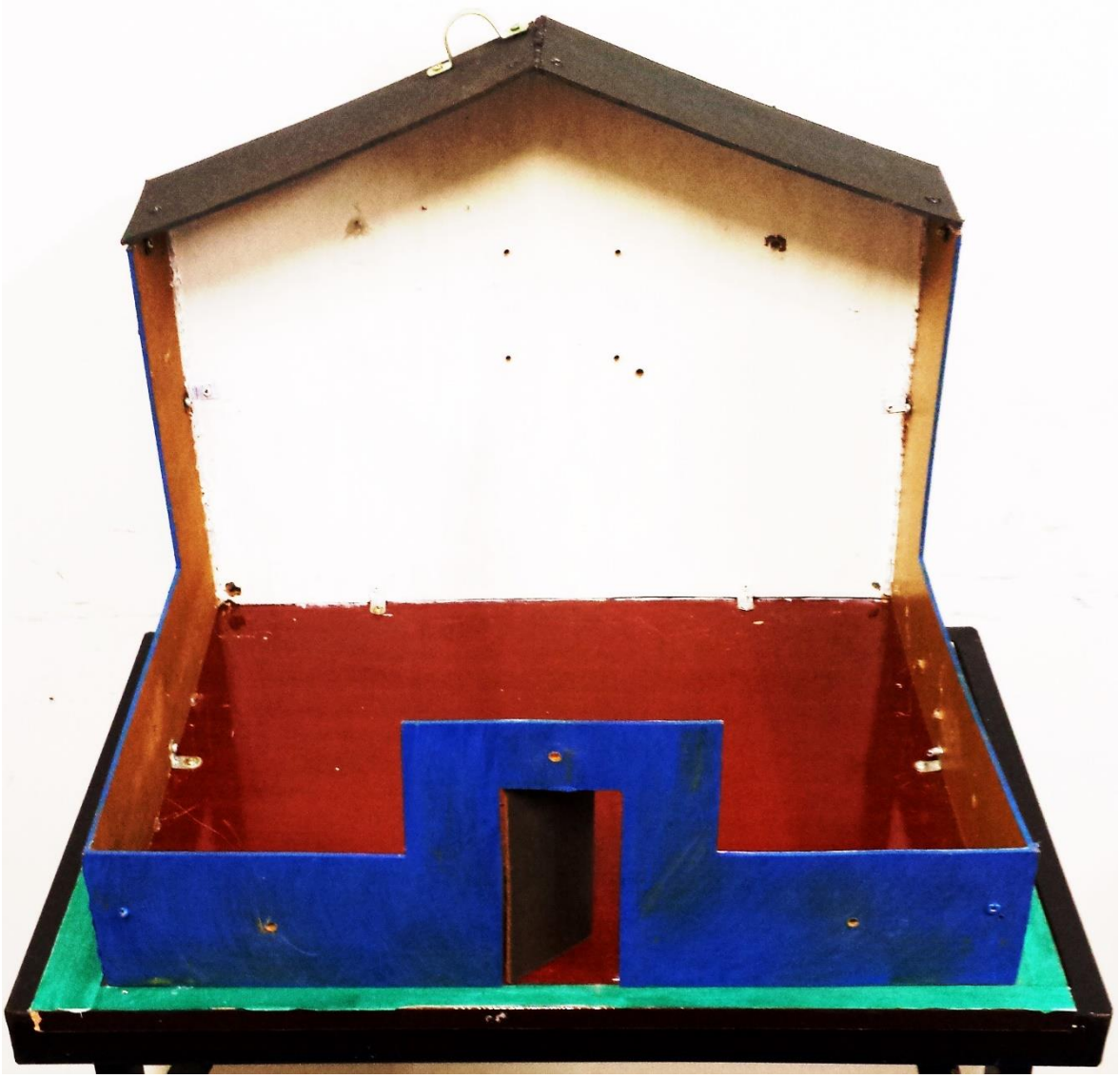

Figura 09 - A maquete, sem instalações.

Foram instalados leds nas paredes externas da casa para simular iluminação externa, estes foram ligados em série juntamente com um resistor adequado para a fonte de tensão utilizada, que foi de 22 volts. Está fonte também foi utilizada para alimentar o cooler acoplado ao centro da parede central para simular um ventilador. Estes dois circuitos foram chaveados utilizando transistores Tip29C, que são ativados e desativados pela tensão do arduino.

Dois circuitos de lâmpadas de 110 volts foram instalados aos lados do cooler, estas são alimentadas pela corrente alternada de uma tomada qualquer, porém o chaveamento de interrupção ou ativação de energia foi feito através de módulos reles.

O circuito para abrir a porta foi elaborado utilizando um servo motor, que tem um fio acoplado à porta, fazendo-a com que acompanhe seu movimento.

Para a reprodução do som de resposta dos comandos foram colocadas caixas de som no centro da maquete.

A figura 10 mostra o projeto já terminado com todas as instalações.

Figura 10 - Projeto com instalações.

3.2 A Gravação De Vozes e Comandos

Primeiramente deve-se pensar em quais comando serão utilizados e quais frases serão respondidas pelo protótipo, posteriormente gravar as respostas aos comandos programados futuramente. Para isso, utiliza-se o *software* Loquendo TTS 7 *Voice Experience*, que permite que se digite uma palavra ou frase, e o *software* reproduza o som do que está escrito, podendo "brincar" com o som da voz que será reproduzida, editando-a. Após isto é necessário salvar os arquivos em formato *wave* (.wav).

A Interface do *software* Loquendo TTS 7 *Voice Experience* pode ser vista na figura 11 a seguir:

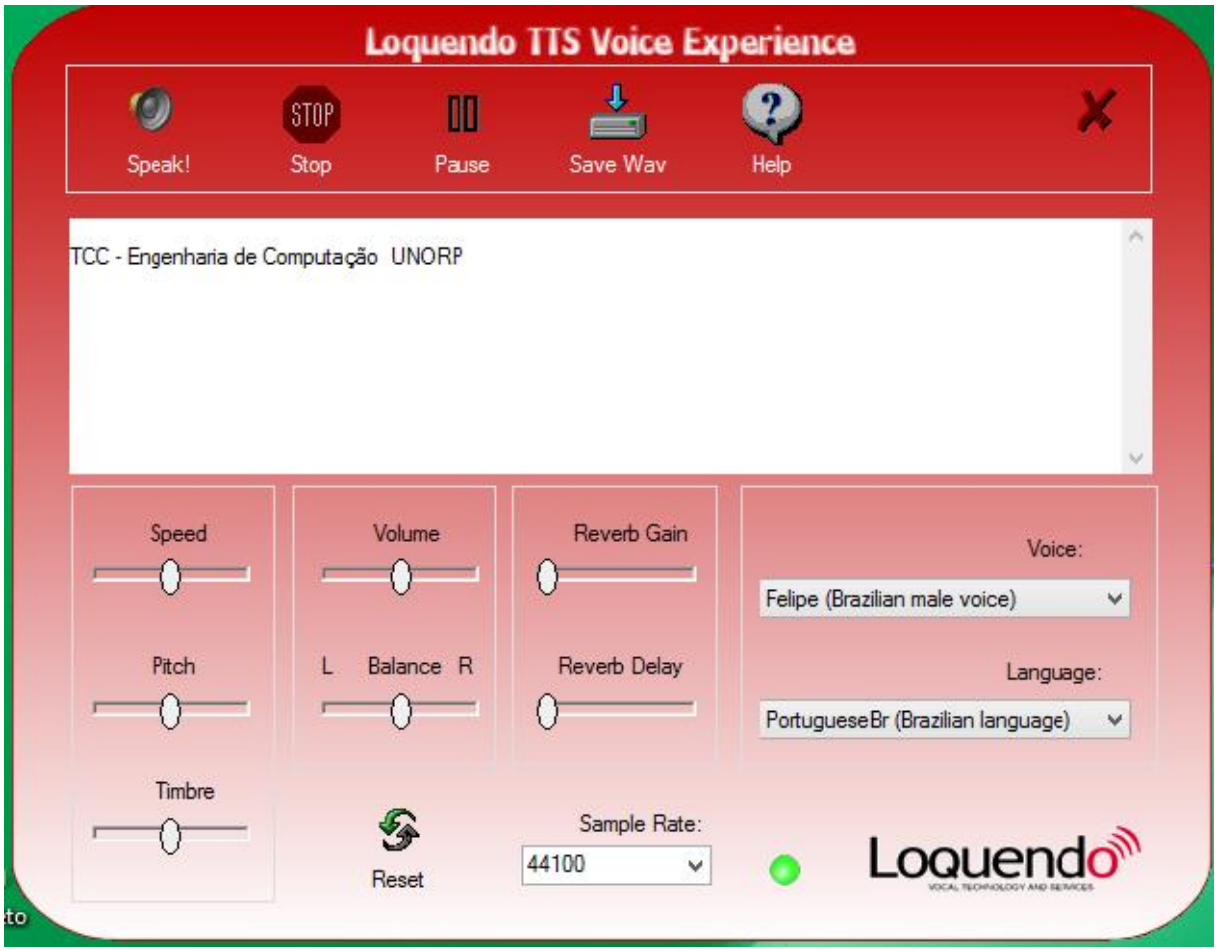

Figura 11 - Interface do Software Loquendo.

Pode-se gravar as respostas de outras maneiras, uma delas é utilizando a própria voz como resposta, isto é possível por meio de outros *software*s, como o

Audacity, por exemplo, ambas as formas necessitam mudar o *software* para gravação em modo mono.

O passo seguinte é a importação destes áudios para a placa EasyVR. Neste caso é necessária uma ligação do *EasyVR* com o *Arduino*. O *shield* do *EasyVR* só recebe os dados através do *Arduino* pois somente ele que tem conexão USB de modo a conectar ao computador.

A figura 12 e a figura 13 mostram como encaixar o *Arduino* e o *Shield*:

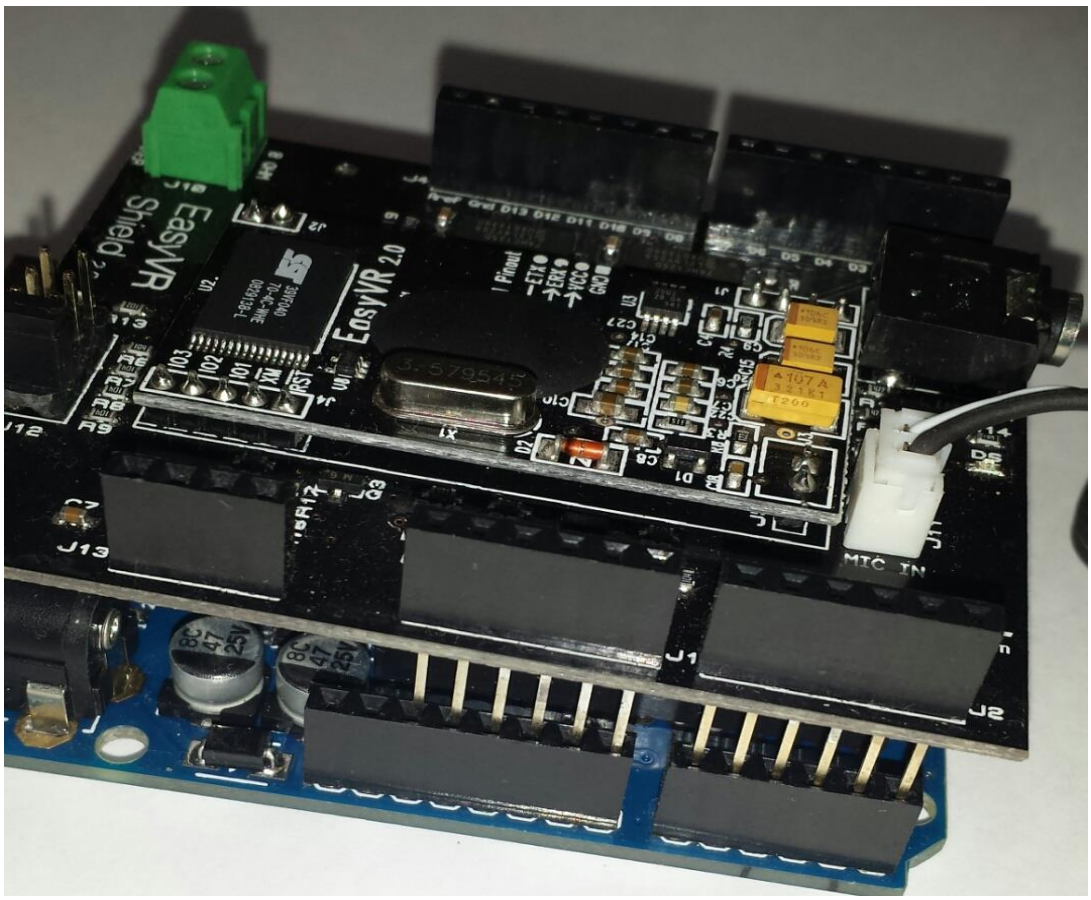

Figura 12 - Conexão Arduino com shield EasyVR

Os pinos do *shield* devem se encaixar perfeitamente no arduino, ficando igualmente o demonstrado na figura 13, a seguir.

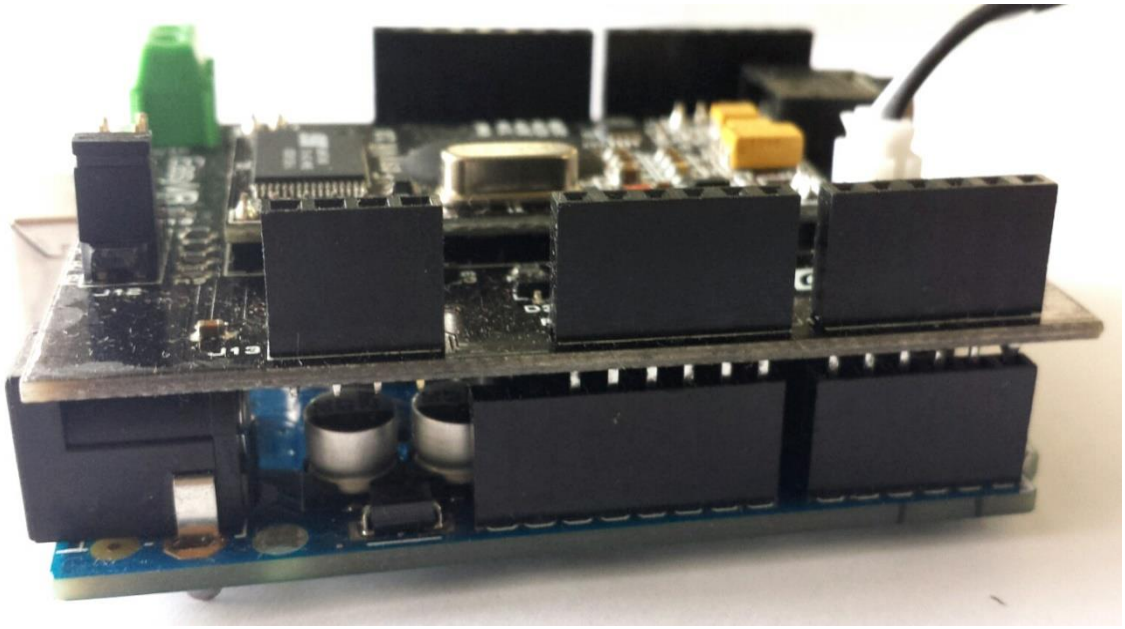

Figura 13 - Shield EasyVR acoplado no Arduino UNO.

Para configuração é necessário a instalação de outro *software* chamado *EasyVR Commander*, e caso for necessário a utilização de gravações como resposta, utiliza-se também do *software Sensory QuickSynthesis 5* para fazer o *upload* dos áudios para o *EasyVR*, nota-se que o *Shield* tem um *jumper* que pode ser acoplado em quatro posições diferentes, que são, UP, PC, HW e SW, e nesta etapa é necessário que ele esteja na posição UP.

A seguir, é mostrada a figura 14 e figura 15 que representam a interface do *software Sensory QuickSynthesis 5* e do *EasyVR Commander*, respectivamente:

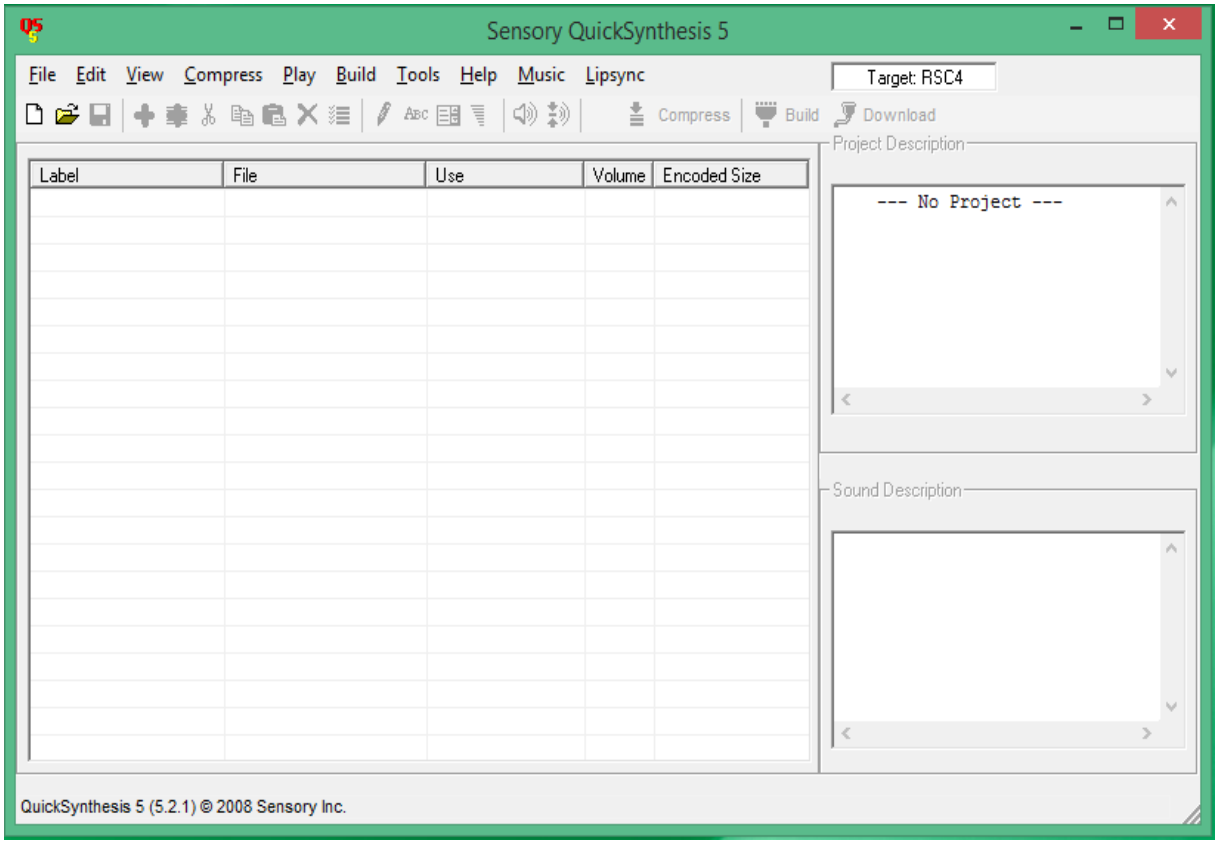

Figura 14 - Interface Sensory QuickSynthesis 5

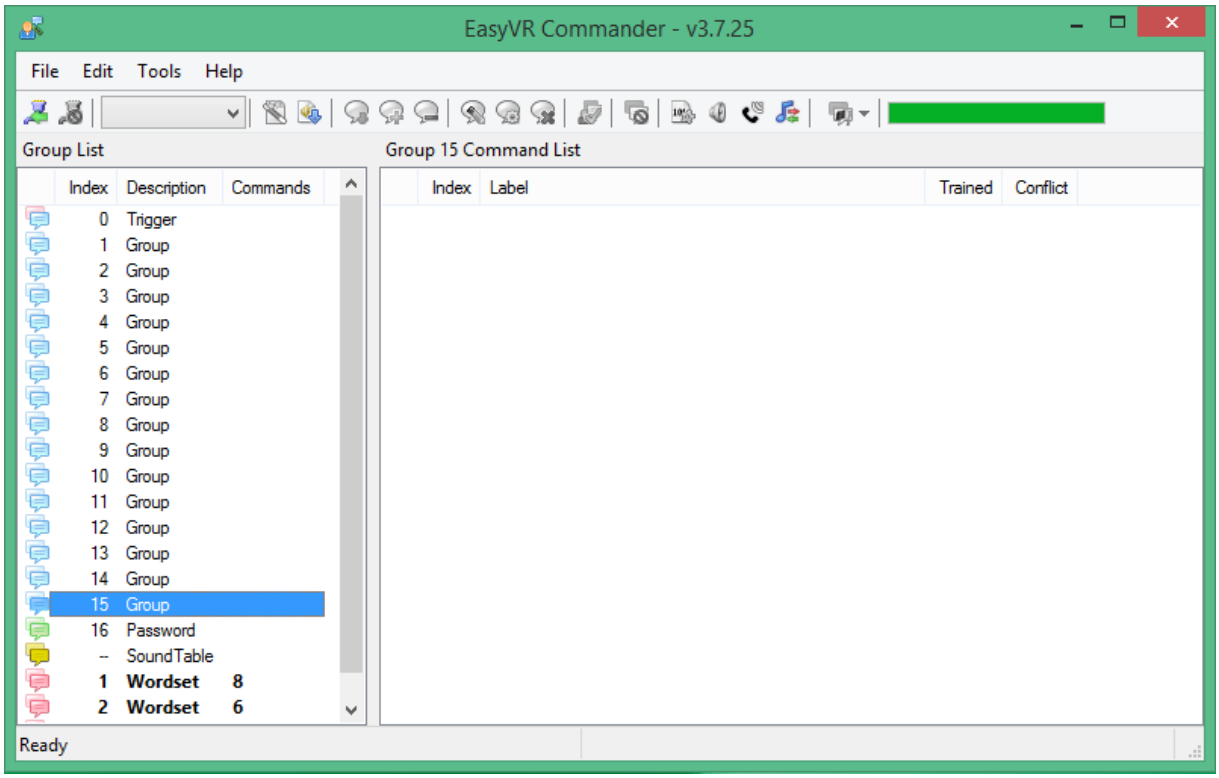

Figura 15 - Interface EasyVR Commander

Com o *Sensory* cria-se um projeto com os áudios gravados, que serão importados posteriormente no *software EasyVR Commander*, ainda com o *jumper* na posição UP.

Após importados os áudios, coloca-se o *jumper* na posição PC, a porta do *arduino* será reconhecida pelo *Commander*, então é só clicar em conectar. Os áudios irão aparecer no grupo *SoundTable*.

Então é possível iniciar a gravação dos comandos que serão reconhecidos pela voz, escolhe-se um grupo onde se gravará alguns comandos e é possível obter uma hierarquia entre os grupos, que posteriormente, serão enfatizadas na programação. É possível também gravar uma senha, onde só serão aceitos comandos após se falar a senha corretamente.

Cada palavra que é colocada nos grupos é treinada duas vezes e deste modo é possível testar estes grupos dizendo cada uma das palavras. Neste momento o programa poderá mostrar se reconheceu alguma delas piscando o elemento correspondente na tabela.

Após todos os comandos serem gravados, é hora de gerar o código. Apenas clicando no botão *Genarate Code* é possível criá-lo com os endereçamentos dos sons, que terão que ser distribuídos corretamente na programação.

3.3 A Programação

A programação do *arduino* é feita na sua própria IDE e em cima do código já criado pelo *software EasyVR Commander*, por se tratar de placas com código livre e *hardware* livre geralmente se têm vários códigos semi prontos para quaisquer projetos. É necessário adicionar algumas configurações e digitar algumas linhas de códigos que serão executadas caso algum comando seja reconhecido, tais como: decidir os áudios que serão ditos como resposta, os pinos que serão acionados, programar as ações que os componentes irão executar, entre outros.

Para compilar o código para o arduino o *jumper* do *EasyVR* deve estar na posição SW.

As mudanças e inclusões que são feitas no código são as seguintes:

Definir quais pinos serão representados por cada elemento, exemplo:

#define luzquarto 3 //Define que a luz do quarto será acionada quando o pino 3 for acionado.

Na função *setup*, definir quais serão as saídas, por exemplo:

pinMode (luzquarto, OUTPUT); //Define o pino luzquarto (pino 3), como saída.

Embora o *EasyVR* reconheça várias línguas, é possível setá-lo para uma linguagem especifica, como ele não possui o português, é preferível (mas não necessário) utilizar a linguagem em espanhol, pois se aproxima mais da nossa língua. Para isso muda-se uma linha de código que está configurada como 0 para a forma seguinte:

De: *Easyvr.setLanguage(0);*

Para: *Easyvr.setLanguage(EasyVR::SPANISH); //mudar a linguagem para espanhol.*

Também na função *setup*, logo embaixo da linha de ajustagem da linguagem, há a linha que define em qual grupo começará a receber os comandos de voz, estruturados na forma *switch*/*case*, de modo a possibilitar que só saia de um grupo de comandos para outro, caso o comando para tal ação seja acionado no grupo em que está atualmente. Assim funciona a senha que pode ser gravada no *Commander*, só irá para o grupo de comandos com acionamentos de equipamentos se a palavra configurada como senha for dita, se o programa reconhecê-la irá pular para o grupo dos equipamentos.

A variável *group* é setada num valor que será referente ao número do grupo que começará. A linha que define o grupo inicial é a seguinte:

group = 16; //Define o programa para iniciar no grupo 16;

A parte das ações, caso algo seja dito, irá na função *loop* do código, onde estão todos os grupos arquitetados em formato *switch*/*case*. Se ele reconhecer um comando de algum grupo, efetuará os códigos descritos, que podem ser: qual som irá ser reproduzido como resposta, que porta será acionada, controlar relés, servos-

motores e etc. e também dar alguma resposta no monitor serial da IDE do *arduino*. Segue abaixo um exemplo caso o comando de uma lâmpada seja dito.

```
case G3_LUZQUARTO:
        easyvr.playSound(5,  EasyVR::VOL_FULL);        // executa o som 5
        //referente à resposta do acionamento da lâmpada.
        digitalWrite (luzquarto,HIGH);        //acende a lâmpada.
        Serial.println("LED AZUL LIGADO");  //escreve no monitor serial.
            group = GROUP_X;            // Utiliza-se este código  caso queira
                                        // ir pra outro grupo.
        break;      //sai do laço de código
```

As configurações básicas são essas, mas pode-se alterar de várias maneiras o código, e utilizar outras funções de programação, de acordo com o nível de experiência de cada um. O código completo pode ser visto no final deste trabalho, no anexo I.

4 CONCLUSÃO

Pode-se concluir que a automação está cada vez mais acessível e cada vez mais simples, bastando algumas pesquisas e equipamentos não muito caros, é possível automatizar, por exemplo, os equipamentos de um quarto com comandos de voz, e garantir mais conforto para si.

Caso se queira automatizar mais elementos, deve se analisar o *Arduino* a ser utilizado, pois se limitam em quantidade de pinos e memória, podendo assim não conseguir alocar todos os equipamentos em suas portas ou seu código ser muito grande para sua memória.

O *EasyVR* por ser um pequeno componente e não ter uma grande capacidade no quesito de hardware não é indicado para grandes automações que exigem mais capacidade, pois além de ele ter um limite de comandos que podem ser gravados, ele se limita com a quantidade de treinos que podem ser armazenados, devido a não disposição de grande quantidade de memória, essa pequena quantidade de treino o faz ter poucas comparações para calcular e reconhecer o comando dito.

Uma outra inconveniência que pode ser encontrada nesse protótipo no decorrer da elaboração, é o microfone que acompanha a placa *EasyVR*, que devido a sensibilidade de ruídos, algumas vezes apresentam problemas de reconhecimento da voz. Para isso, a solução que se faz necessária, seria uma adaptação de um microfone com melhores ajustes para reconhecimento e eliminação de ruídos.

Para um bom funcionamento dos comandos é necessário um certo silêncio no local de modo a eliminar boa parte dos ruídos, pois o módulo pode confundir sons e ter dificuldades para reconhecê-los.

Este conceito em automatizar por comandos de voz não se limita a essas pequenas demonstrações que foram feitas, pode se ir muito além utilizando esta ferramenta. Alguns exemplos de aplicações uteis seriam, abrir uma garagem de carro com o microfone no carro de modo que este envie um sinal de rádio frequência para o receptor no motor do portão, mudar de canais e controlar o volume de uma TV ou controlar um rádio, utilizando controladores infravermelho.

Outro projeto interessante seria o de controlar uma cama de hospital, onde um paciente, sem muita mobilidade, poderia ajustar facilmente a posição de sua cama apenas utilizando sua voz.

Há hoje no mercado outras tecnologias nesta área, com maior poder de hardware e com melhor capacidade, podendo ser adquiridas para projetos mais ambiciosos.

Como apresentado no capítulo sobre reconhecimento de voz, está é uma tecnologia um pouco complexa, mas para um pequeno módulo este é bem vantajoso. Com o avanço da tecnologia, está próximo o momento onde será pleno o reconhecimento da voz e até a conversação com computadores e máquinas, independentemente de sons e ruídos aleatórios.

REFERÊNCIAS

Ana Paula Pereira. **Como funciona o reconhecimento de voz.** 2009. Disponível em: <http://www.tecmundo.com.br/curiosidade/3144-como-funciona-o-reconhecimento-de-voz-.htm> Acesso em:24 jul. 2014.

ARDUINO. **Arduino** 2014 Disponível em: <http://playground.arduino.cc/Portugues/HomePage> Acesso em: 21 Jul. 2014

ARNEROBOTICS. **Microcontrolador PIC**. Disponível em: < http://www.arnerobotics.com.br/eletronica/Microcontrolador_PIC_teoria_1.htm > Acesso em: 17 Ago. 2014

Carlos Alexandre Mello. **Processamento digital de voz**. Disponível em: <http://www.cin.ufpe.br/~cabm/pds/PDS_Aula12_PDV.pdf> Acesso em: 24 jul. 2014.

DQSOFT, **Microcontrolador Atmega328** 2011. Disponível em: <http://dqsoft.blogspot.com.br/2011/07/microcontrolador-atmel-atmega328-parte.html> Acesso em: 13 Set. 2014.

EVANS, M.; JOSHUA, N.; HOCHENBAUM, J. **Arduino em Ação.** Novatec 2013

GDSAUTOMACAO. **O que é automação residencial**. 2014 Disponível em: <http://www.gdsautomacao.com.br/public/index.php?option=com_content&view=article&id=51:o-que-e-automacao-residencial&catid=1:latest-news> Acesso em: 22 out. 2014.

Grabianowski. Ed. **Da voz para dados.** Disponível em: http://tecnologia.hsw.uol.com.br/reconhecimento-de-voz1.htm>Acesso em: 24 jul.2014

HSW.UOL. **Microcontroladores**. Disponível em: <http://tecnologia.hsw.uol.com.br/microcontroladores1.htm> Acesso em: 16 Ago. 2014.

LABDEGARAGEM. **Tutorial: Reconhecimento de voz com Arduino.** 2014. Disponível em: <http://labdegaragem.com/profiles/blogs/tutorial-como-utilizar-o-easyvr-shield-com-arduino> Acesso em: 23 jul. 2014.

LABDEGARAGEM. **EasyVR Shield de reconhecimento de voz.** 2012 Disponível em: <http://www.labdegaragem.org/loja/38-comunicacao/easyvr-shield-reconhecimento-de-voz.html> Acesso em: 26 jul. 2014.

LABDEGARAGEM. **Artigo os tipos de memória encontrados em microcontroladores**. Disponível em: <http://labdegaragem.com/profiles/blogs/artigo-os-tipos-de-mem-ria-encontrados-em-microcontroladores> Acesso em: 17 Ago. 2014

NERD KING. **O que é arduino?!** 2014. Disponível em: <https://www.youtube.com/watch?v=wQDDibn8VUE> Acesso em: 26 jul. 2014

ROBOCORE. **Arduino Shield – EasyVR – Reconhecimento de Voz**. Disponível em: <https://www.robocore.net/modules.php?name=GR_LojaVirtual&prod=213> Acesso em: 23 Jul. 2014.

ROBOLIVRE. **Microcontroladores**. Disponível em:
< http://robolivre.org/conteudo/microcontroladores> Acesso em: 16 Ago. 2014.

Souza, Emmanuel Cássio Oliveira. **Criando Textos Segundo as normas ABNT,** 2005. Disponível em : <http://pt.slideshare.net/marketingaruja/normas-abntnoword> Acesso em: 12 Set. 2014.

SRAENGENHARIA. **Histórico da automação residencial**. 2013. Disponível em: <http://sraengenharia.blogspot.com.br/2013/01/historico-da-automacao-residencial_10.html> Acesso em: 22 out. 2014.

SUPERTECNO, **Atmega328** 2012. Disponível em: <http://supertecno.com.br/arduino/atmega328ci.html> Acesso em: 13 Set. 2014.

TECHTUDO. **O que é um Arduino e o que pode ser feito com ele?** 2013. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2013/10/o-que-e-um-arduino-e-o-que-pode-ser-feito-com-ele.html> Acesso em: 21 Jul. 2014

ANEXO I – Código de programação.

```
// Declaração externa do SoundTable

extern far cdata SNDTBL_TCC;

// external declarations of addresses of sounds for direct access:

extern far cdata SX_AbrindoPorta;
extern far cdata SX_DeslgandoLuzBanheiro;
extern far cdata SX_DesligandoLuzSala;
extern far cdata SX_DesligandoVentilador;
extern far cdata SX_DigaSenha;
extern far cdata SX_Finalizando;
extern far cdata SX_LigandoLuzBanheiro;
extern far cdata SX_LigandoLuzSala;
extern far cdata SX_LigandoVentilador;
extern far cdata SX_LuzExternaDesligada;
extern far cdata SX_LuzExternaLigada;
extern far cdata SX_SenhaConfirmada;

// constants defining sounds as entries in the sound table:

#define SND_AbrindoPorta       1
#define SND_DeslgandoLuzBanheiro 2
#define SND_DesligandoLuzSala   3
#define SND_DesligandoVentilador 4
#define SND_DigaSenha          5
#define SND_Finalizando        6
#define SND_LigandoLuzBanheiro  7
#define SND_LigandoLuzSala      8
#define SND_LigandoVentilador   9
#define SND_LuzExternaDesligada 10
#define SND_LuzExternaLigada    11
#define SND_SenhaConfirmada     12


//=============================================================================


// Código


#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#include "Platform.h"
#include "SoftwareSerial.h"
#ifndef CDC_ENABLED
// Jumper do shield na posição SW
SoftwareSerial port(12,13);
#else
// Jumper do shield na posição HW (Para Arduino Leonardo)
```

```
#define port Serial1
#endif
#else // Arduino 0022 - use modified NewSoftSerial
#include "WProgram.h"
#include "NewSoftSerial.h"
NewSoftSerial port(12,13);
#endif

#include "EasyVR.h"

#include <Servo.h> //incluindo biblioteca para servo motor

Servo porta;  //nomeando o servo como variavel "porta".

EasyVR easyvr(port);

//************************************************************
#define banheiro 6           // Definindo os pinos a cada elemento
#define sala 7
#define externa 8
#define ventilador 9
//************************************************************

//Grupos e comandos
enum Groups             //*Enumerando os grupos
{
 GROUP_1  = 1,
 GROUP_2  = 2,
 GROUP_16 = 16,
};

enum Group1
{                   //*Aqui são enumerados os comandos gravados
 G1_AUTOMACAO = 0,      // dentro de cada grupo, para facilitar referênciamento.*
};

enum Group2
{
 G2_PORTA = 0,
 G2_BANHEIRO = 1,
 G2_SALA = 2,
 G2_VENTILADOR = 3,
 G2_EXTERNAS = 4,
 G2_CONCLUIR_TRABALHO = 5,
};

enum Group16
{
 G16_ENGENHARIA = 0,
};
```

```
EasyVRBridge bridge;

int8_t group, idx;

//variaveis criadas para acionamento e desacionamento.
int b; //banheiro
int s; //sala
int v; //ventilador
int e; //externas

void setup()    //função setup.
{
  porta.attach(3);              //Definindo pino 3 ao servo "porta"
  pinMode (ventilador, OUTPUT);   // Setando os pinos referentes aos
  pinMode (banheiro, OUTPUT);      // elementos como saída.
  pinMode (sala, OUTPUT);
  pinMode (externa, OUTPUT);

  porta.write(25);        //definindo valores iniciais.
  digitalWrite(sala, HIGH);
  digitalWrite(banheiro, HIGH);

#ifndef CDC_ENABLED
  // bridge mode?
  if (bridge.check())
  {
    cli();
    bridge.loop(0, 1, 12, 13);
  }
  // run normally
  Serial.begin(9600);
  Serial.println("Bridge not started!");
#else
  // bridge mode?
  if (bridge.check())
  {
    port.begin(9600);
    bridge.loop(port);
  }
  Serial.println("Bridge connection aborted!");
#endif
  port.begin(9600);

  while (!easyvr.detect())
  {
    Serial.println("EasyVR não detectado!");
    delay(1000);
  }

  easyvr.setPinOutput(EasyVR::IO1, LOW);
  Serial.println("EasyVR detectado!");
  easyvr.setTimeout(5);
```

```
 easyvr.setLanguage(EasyVR::SPANISH);   //Aqui pode se setar a linguagem.

 group = GROUP_1; // Este comando seta em qual grupo de comandos ira começar
}

void action();

void loop()   //função loop, esta ficara sempre lendo os dados e executando
{           //  ações caso seja acionada algum comando.
 easyvr.setPinOutput(EasyVR::IO1, HIGH); //Led do EasyVR ligado (mostra que está lendo o som)

 Serial.print("Diga um comando no grupo ");
 Serial.println(group);  //Imprime no monitor serial em qual grupo está.
 easyvr.recognizeCommand(group);

 do
 {
  // Pode se fazer algum processo enquanto espera um comando. Foi deixado vazio
 }
 while (!easyvr.hasFinished());

 easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off

 idx = easyvr.getWord();
 if (idx >= 0)
 {
  // built-in trigger (ROBOT)
  // group = GROUP_X; <-- jump to another group X
  return;
 }
 idx = easyvr.getCommand();
 if (idx >= 0)
 {
  // print debug message
  uint8_t train = 0;
  char name[32];
  Serial.print("Command: ");
  Serial.print(idx);
  if (easyvr.dumpCommand(group, idx, name, train))
  {
   Serial.print(" = ");
   Serial.println(name);
  }
  else
   Serial.println();
  easyvr.playSound(0, EasyVR::VOL_FULL); //aciona o beep.
  // perform some action
  action();    // Aqui se chama a função action, onde executará caso a voz reconhecida
 }
 else  // caso erros ou tempo de espera termine
 {
  if (easyvr.isTimeout())
```

```
    Serial.println("Tempo acabado, tente novamente...");
   int16_t err = easyvr.getError();
   if (err >= 0)
   {
    Serial.print("Error ");  //caso nao reconheça ou de algum erro ele imprime o numero do erro.
    Serial.println(err, HEX);
   }
  }
}

void action() //função action. Aqui se encadeia os "switch/case". Onde se decidirá os comandos
{          // efetuados a cada reconhecimento de palavra.
  switch (group)
  {
  case GROUP_1:
   switch (idx)
   {
   case G1_AUTOMACAO:
    easyvr.playSound(5, EasyVR::VOL_FULL);   // A casa será acionada e pedirá uma senha,
    Serial.println("Olá. Diga a senha");
    group = GROUP_16; //Pulará para o grupo 16 onde está a senha.
    break;
   }
   break;

  case GROUP_2:
   switch (idx)
   {
   case G2_PORTA:    //caso reconheça o comando de acionamento da porta
    easyvr.playSound(1, EasyVR::VOL_FULL);   //executa o som gravado como resposta
    Serial.println("Abrindo Porta");      //imprime no monitor serial uma resposta
    porta.write(170);                  // comando para o servo ir para posição 170
    delay(5000);                      // tempo de espera ocioso.
    porta.write(25);                   // servo motor para posição 25
    break;

   case G2_BANHEIRO:
    if (b == 0){     // se o banheiro estiver desligado
     easyvr.playSound(7, EasyVR::VOL_FULL);
     digitalWrite(banheiro, LOW);     // Desliga o relé, acionando a passagem de energia 110v da
lampada.
     Serial.println("Luz do banheiro ligada");
     b = 1; //muda a variavel para 1.
    }
    else{
     easyvr.playSound(2, EasyVR::VOL_FULL);
     digitalWrite(banheiro, HIGH);     // Liga o relé, impedindo a passagem de energia 110v da
lampada.
     Serial.println("Luz do banheiro desligada");
     b = 0;
    }
    break;
```

```
   case G2_SALA:
    if (s == 0){
      easyvr.playSound(8, EasyVR::VOL_FULL);
      digitalWrite(sala, LOW);       // Desliga o relé, acionando a passagem de energia 110v da
lampada.
      Serial.println("Luz da Sala ligada");
      s = 1;
    }
    else{
      easyvr.playSound(3, EasyVR::VOL_FULL);
      digitalWrite(sala, HIGH);   // Liga o relé, impedindo a passagem de energia 110v da lampada.
      Serial.println("Luz da sala desligada");
      s = 0;
    }
    break;

   case G2_VENTILADOR:
    if (v == 0){
      easyvr.playSound(9, EasyVR::VOL_FULL);
      digitalWrite(ventilador, HIGH);      // Aciona a porta referente ao ventilador (cooler), fazendo
com que ligue.
      Serial.println("Ventilador ligado");
      v = 1;
    }
    else{
      easyvr.playSound(4, EasyVR::VOL_FULL);
      digitalWrite(ventilador, LOW);        // Desliga a passagem de energia na porta do cooler.
      Serial.println("Desligando Ventilador");
      v = 0;
    }
    break;

   case G2_EXTERNAS:
    if (e == 0){
      easyvr.playSound(11, EasyVR::VOL_FULL);
      digitalWrite(externa, HIGH);      // liga luzes externas (leds externos da maquete)
      Serial.println("Luzes externas ligadas");
      e = 1;
    }
    else{
      easyvr.playSound(10, EasyVR::VOL_FULL);
      digitalWrite(externa, LOW);         //Desliga a porta dos leds externos
      Serial.println("Luzes externas desligadas");
      e = 0;
    }
    break;

    case G2_CONCLUIR_TRABALHO:        //Aqui é dito para sair dos comandos e voltar para o inicio
      easyvr.playSound(6, EasyVR::VOL_FULL);
      Serial.println("Modulo finalizado");
      for (int x=0; x<20; x++) {      //Comando para piscar os leds ao finalizar.
```

```
        digitalWrite (externa, HIGH);
        delay (100);
        digitalWrite (externa, LOW);
        delay (100);
        digitalWrite(sala, HIGH);
       }
       digitalWrite(sala, HIGH);
       digitalWrite(banheiro, HIGH);
       digitalWrite(ventilador, LOW);
       e = 0;
       s = 0;
       b = 0;
       v = 0;
       group = GROUP_1;    // Volta para o grupo 1.
      break;
    }
    break;

  case GROUP_16:
    switch (idx)
    {
    case G16_ENGENHARIA:
      easyvr.playSound(12, EasyVR::VOL_FULL);     //caso a senha seja dita.
      Serial.println("Senha confirmada");
      group = GROUP_2;      //muda para o grupo 2 de acionamento de equipamentos.
      break;
    }
    break;
  }
}
```