unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

SINEZIO HENRIQUE PINTO

Desenvolvimento de software para controle de dispositivos elétricos via Bluetooth e comando de voz

Guaratinguetá
2016

SINEZIO HENRIQUE PINTO

Desenvolvimento de software para controle de dispositivos elétricos via Bluetooth e comando de voz

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador: Prof. Dr. José Feliciano Adami

Guaratinguetá
2016

Pinto, Sinézio Henrique
P659d Desenvolvimento de software para controle de dispositivos elétricos via bluetooth e comando de voz/ Sinézio Henrique Pinto – Guaratinguetá, 2016.
83 f : il.
Bibliografia: f. 73-74

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2016.
Orientador: Prof. Dr. José Feliciano Adami

1. Software-Desenvolvimento. 2. Automação. 3. Arduino (Controlador programável). 4. Dispositivos dielétricos. I. Título

CDU 621.316.57

DESENVOLVIMENTO DE SOFTWARE PARA CONTROLE DE DISPOSITIVOS ELÉTRICOS VIA BLUETOOTH E COMANDO DE VOZ

SINEZIO HENRIQUE PINTO

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE **GRADUADO EM ENGENHARIA ELÉTRICA**

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Prof. Dr. Leonardo Mesquita
Coordenador

BANCA EXAMINADORA:

Prof. Dr. José Feliciano Adami
Orientador/UNESP-FEG

Profa. Dra. Daniela Wolter Ferreira Touma
UNESP-FEG

Profa. Maria Fernanda Trujillo
UNESP-FEG

Dezembro de 2016

DADOS CURRICULARES

SINEZIO HENRIQUE PINTO

NASCIMENTO	08.06.1988 - Aparecida / SP
FILIAÇÃO	Sinezio Pinto Maria Stela Pinto
2010/2016	Curso de Graduação em Engenharia Elétrica na Universidade Estadual Paulista "Júlio de Mesquita Filho" Campus de Guaratinguetá.
2016/2016	Estágio na área de telecomunicações na Universidade Estadual Paulista "Júlio de Mesquita Filho" Campus de Guaratinguetá.

de modo especial, dedico este trabalho aos meus pais, irmãs e minha namorada que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida, aos professores, amigos e colegas, pelo incentivo e pelo apoio constante e a todos aqueles que de alguma forma estiveram e estão próximos de mim, fazendo esta vida valer cada vez mais a pena.

AGRADECIMENTOS

A minha mãe Maria Stela, por todo o amor, apoio e suporte dado nessa trajetória e ao meu pai Sinézio que hoje descansa na morada eterna, mas que com toda certeza tem estado todo tempo junto a mim me dando forças para vencer todos os obstáculos propostos pela vida.

As minhas irmãs Tatiana e Eliana, por sempre serem tão presentes e entusiastas em toda minha trajetória acadêmica.

A minha namorada Kerolene Barboza, pela paciência, compreensão e pelo total apoio durante anos de estudo.

Ao meu orientador Prof. Dr. José Feliciano Adami, pelo apoio e incentivo desde o inicio, orientando e colaborando com o possível e sempre incentivando para o crescimento e desenvolvimento deste projeto.

A todos os amigos e colegas que fazem ou fizeram parte da minha vida.

“Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.”

Charles Chaplin

RESUMO

Nos dias atuais, a utilização de novas tecnologias está cada vez mais evidente no o auxílio de tarefas domésticas e na ajuda para solucionar problemas decorrentes do dia a dia, como por exemplo, o simples uso de um despertador eletrônico contido nos celulares. Isto se deve ao grande impacto em que os *smartphones* causam na vida das pessoas, pois é difícil encontrar alguém que não faça o uso destes novos aparelhos. Entretanto, os celulares *smartphones* não se restringem apenas para a função de comunicação, também abrangem diversas funções, o que aumenta consideravelmente o espaço da automação nas residências e comércios. O objetivo principal deste trabalho foi desenvolver um aplicativo de celulares e *tablets* android para o controle de dispositivos elétricos empregando a tecnologia *Bluetooth* associado ao comando de voz, e um protótipo foi construído para simular o uso da automação em um quarto de hospital. Este ambiente foi escolhido após relatos de pessoas debilitadas que sentiram a dificuldade de executar algumas tarefas simples, como por exemplo, o acionamento de uma lâmpada sem a ajuda de outra pessoa. Para a realização deste projeto foi feito um estudo na área de telecomunicações que abrange o uso da tecnologia *Bluetooth* e o sistema de reconhecimento de voz. Também foi usado o hardware Arduino, responsável pelo acionamento dos dispositivos além da criação de um aplicativo através do site do MIT App Inventor para a comunicação com a placa.

PALAVRAS-CHAVE: Bluetooth. Comando de voz. Arduino. Aplicativo. App Inventor. Automação. Quarto de hospital.

ABSTRACT

Nowadays, the use of new technologies is increasingly evident in the aid of housework and in helping to solve problems resulting from everyday life, for example, the simple use of an electronic alarm clock contained in cell phones. This is due to the great impact that smartphones cause in people's lives, as it is difficult to find someone who doesn't use these new mobiles. However, smartphones aren't only restricted to the communication function, they also cover many functions, which considerably increases the space of automation in homes and businesses. The main objective of this work was to develop a mobile app to control electrical devices employing the Bluetooth technology associated with the voice command, and a prototype was constructed to simulate the use of automation in a hospital room. This place was chosen after reports of debilitated people who felt difficulty to execute some simple tasks without the help of another person. For the realization of this project a study was made in the area of telecommunications that covers the use of Bluetooth technology and the system of voice recognition. The hardware Arduino also was used, which is responsible for triggering the devices beyond the creation of a mobile app through the MIT App Inventor website for communication with the board.

KEYWORDS: Bluetooth. Voice command. Arduino. Mobile app. App Inventor. Automation. Hospital room.

LISTA DE ILUSTRAÇÕES

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AC	*Alternating Current*
ACL	*Asynchronous Connection-Less*
AFH	*Adaptive Frequency Hopping*
CI	Circuito Integrado
CLP	*Power Line Carrier*
CLPs	Controladores Lógicos Programáveis
DC	*Direct Current*
EDR	*Enhanced Data Rate*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
FSK	*Frequency Shift Keying*
GFSK	*Gaussian Frequency Shift Keying*
GHz	Giga hertz
GND	*Ground*
IDE	*Integrated Development Environment*
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
iOS	*iPhone Operating System*
IR	*Infra red*
ISM	*Industrial, Scientific and Medical*
kB	kilobytes
kb/s	kilobites por segundo
kΩ	kilohms
LED	*Light Emitting Diode*
mA	miliamperes
Mbps	Megabits por segundo
Mb/s	Megabytes por segundo
MHz	Mega hertz
MIPS/MHz	Milhão de instruções por segundo por Mega hertz
MIT	*Massachusetts Institute of Technology*
ms	milisegundos
Msps	Mega símbolos por segundo
PC	*Personal Computer*
PSK	*Phase Shift Keying*

PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
RSSI	*Received Signal Strength Indication*
SCO	*Synchronous Connection-Oriented*

LISTA DE SÍMBOLOS

Ω	Ohms
A	Ampére
V	Volt

SUMÁRIO

1 INTRODUÇÃO

A automação é um sistema que aplica procedimentos que comandam e controlam os mecanismos para seu próprio funcionamento. A palavra automação vem do latim *automatus* que significa mover-se por si. (SIGNIFICADOS, 2016)

Nos dias atuais, tem-se o alcance uma gama de possibilidades práticas e econômicas que utilizam a automação, desde a básica até a mais abrangente, em sistemas de integração para diversos ambientes como, por exemplo: Indústrias, imóveis, automóveis, entre outros. (GDS AUTOMAÇÃO, 2016)

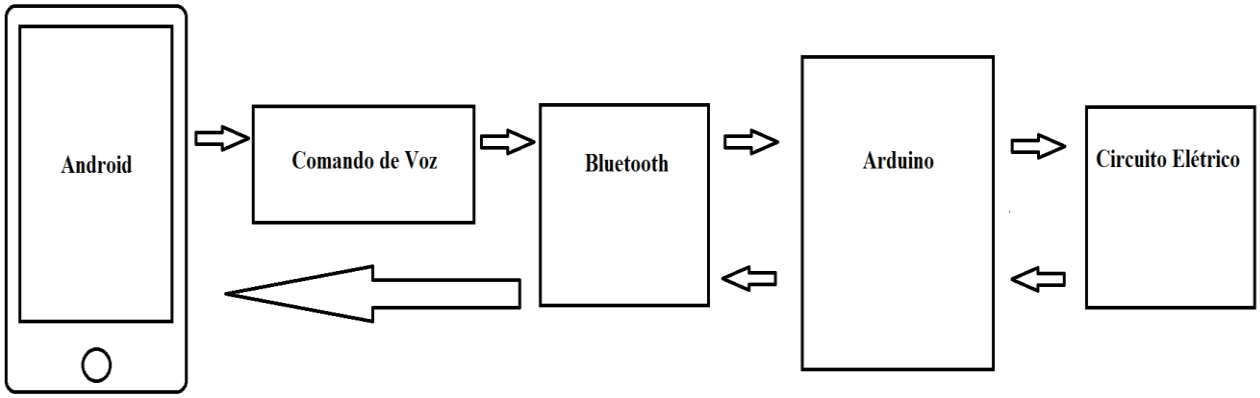
Atualmente a domótica, termo utilizado para automação residencial, tem apresentado um crescimento significativo devido ao avanço tecnológico e ao uso de novas tecnologias, facilitando o controle e gerenciamento dos afazeres domésticos, provendo maior segurança e comodidade ao lar. Com a popularização dos *smartphones* a utilização dessas novas tecnologias ficou a alcance de todos, ampliando ainda mais o mundo da automação e facilitando o surgimento de novas ideias para serem automatizadas de acordo com as necessidades de cada um. (MURATORI, 2011)

Assim, este trabalho parte da ideia de fazer o uso da tecnologia obtida na automação com a possibilidade do acesso de qualquer pessoa a um *smartphone*. Um quarto de hospital é um ambiente ideal para a aplicação desta tecnologia, pois uma pessoa debilitada poderá ser menos dependente para executar certas tarefas usando apenas o celular para acionar dispositivos como, por exemplo, uma lâmpada sem precisar se levantar, somente através de botões ou de comandos de voz, e por isso foi escolhido para ser aplicado este projeto.

A automação do quarto baseia-se na criação de um aplicativo para celulares e *tablets* Android e na adaptação de um novo *hardware* no circuito elétrico que se comunica com o aplicativo através da tecnologia *Bluetooth*. O *hardware* utilizado neste projeto é a placa Arduino UNO, que através de um módulo *Bluetooth*, recebe sinais do aplicativo para acionar cargas como: campainha, lâmpada, televisão, ventilador e motores para abrir e fechar a cortina, subir e descer a cama.

Também está presente no aplicativo a possibilidade do controle de dispositivos via comando de voz, que pode substituir o uso dos botões. A Figura 1 representa o diagrama de blocos do encadeamento de ações que são realizadas quando um dispositivo é controlado pelo aplicativo.

Figura 1 - Diagrama de blocos



Fonte: Produção do próprio autor

A seguir, o capítulo 2 apresenta uma revisão bibliográfica que mostra o estudo de todas as etapas deste projeto. O capítulo 3 mostra como foi desenvolvido o aplicativo e o programa carregado na placa do Arduino, além dos circuitos eletrônicos montados em um protótipo para a simulação deste projeto. No capítulo 4 está contido a viabilidade de implementação do sistema com uma tabela de preços de cada componente e por fim, no capítulo 5 são apresentadas as conclusões sobre este trabalho.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo tem por objetivo fazer um breve estudo sobre a evolução da automação residencial, uma revisão sobre o padrão de comunicação sem fio IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos) 802.15.1 denominado *Bluetooth*, uma introdução ao sistema de comando de voz, o desenvolvimento do protótipo de sistemas microcontrolados com a utilização do Arduino UNO e por último, uma introdução à aplicativos desenvolvidos com o programa App Inventor 2.

2.1 AUTOMAÇÃO RESIDENCIAL: HISTÓRICO, DEFINIÇÕES E CONCEITOS

A automação residencial é uma derivação da automação industrial que através dos dispositivos CLPs (Controladores Lógicos Programáveis), datados na década de 60, teve uma grande revolução, devido aos avanços da microeletrônica. Muitas empresas enxergaram uma nova oportunidade de mercado e migraram das indústrias para as residências com o propósito de facilitar o controle das tarefas domésticas e aumentar a segurança e comodidade do lar. Enquanto que na automação industrial é essencial que os equipamentos operem sem que haja algum tipo de falha e com uma elevada precisão, na automação residencial esses quesitos podem ser menos rigorosos e possuir acabamentos mais elaborados e sofisticados comparados à automação nas indústrias. (SILVA, 2011)

Na década de 70 foram lançados nos Estados Unidos da América os primeiros módulos inteligentes batizados de X-10, o que pode ser considerado o pontapé inicial da automação residencial. O protocolo X-10 fazia o uso da rede elétrica como canal de comunicação entre os múltiplos dispositivos de automação. Trata-se de uma tecnologia CLP (*Power Line Carrier*) que permite o controle de dispositivos remotos sem precisar alterar a infraestrutura elétrica da residência. (FERREIRA, 2010)

Na década seguinte, com a popularização dos computadores pessoais surgiu a ideia de utilizar um PC como central de automação. Porém, essa ideia não era muito vantajosa, pois além de existir um elevado consumo pela necessidade de manter o PC sempre ligado, também havia a desvantagem de ter a centralização do controle que poderia vir a ser falho e comprometer o funcionamento de todo o sistema. A partir desses problemas, parte-se para o desenvolvimento de dispositivos embarcados através da utilização de microprocessadores e microcontroladores e da isenção dos PCs. (BEGHINI, 2013)

Após alguns estudos e avanços, paralelamente surge a internet banda larga que concedeu ao usuário a possibilidade de controlar e monitorar a residência de qualquer lugar, longe ou perto do local, que possa dispor deste serviço. (OLIVEIRA, 2014)

Por fim, a partir do século XXI surgem dispositivos *smarts* (celulares e *tablets*) que incorporam diferentes serviços como telefonia e internet (TECMUNDO, 2014) e são apontados como responsáveis pelo aumento do número de residências automatizadas, pois facilitam o controle e o monitoramento dos lares que podem ser feitos apenas com um toque na tela do celular.

2.2 TECNOLOGIA BLUETOOTH

O *Bluetooth* é uma tecnologia de comunicação sem fio que foi desenvolvida pela empresa de telecomunicações Ericsson, em 1994, e tem como objetivo a transmissão de dados e arquivos de maneira rápida e segura sem a necessidade de conexões de cabos, sendo necessário somente que os aparelhos estejam próximos uns dos outros, dentro de um determinado raio de ação e que possuam esta tecnologia como, por exemplo, celulares, *notebooks*, teclados e *mouses* de computador, fones de ouvido, entre outros. Esta tecnologia foi denominada "*Bluetooth*" em homenagem a um antigo rei da Dinamarca e da Noruega, Harold Blatand (em inglês, *Harold Bluetooth*). Esse nome foi dado a esta tecnologia pelo fato de Harold Blatand ter unificado as tribos norueguesas, suecas e dinamarquesas, já que a tecnologia *Bluetooth* é exatamente uma forma de união de diferentes dispositivos. O logotipo e símbolo do *Bluetooth* também foram baseados no nome de Harold e correspondem às iniciais do nome do rei, "H" e "B", respectivamente. (INFOWESTER, 2013)

A Figura 2 apresenta o logotipo da tecnologia *Bluetooth*.

Figura 2 - Logotipo da tecnologia *Bluetooth*

Fonte: www.bluetooth.com

Devido ao fato do *Bluetooth* ser uma tecnologia aberta, ele possui desenvolvimento e melhora constante, por isso existem versões que aperfeiçoam o dispositivo de acordo com alguma especificação necessária, como por exemplo, o aumento da taxa de transmissão ou a diminuição do uso da bateria. Atualmente o *Bluetooth* está na versão 4.0, com diversas melhorias em relação à primeira versão. A seguir são descritas as principais características das versões existentes.

2.2.1 Bluetooth 1.0 e 1.0B

Por ser primeira versão feita, com as primeiras especificações do *Bluetooth*, foi encontrado pelos fabricantes alguns problemas que dificultavam a implementação e a forma com que os dispositivos se comunicavam entre si via *Bluetooth.* A velocidade padrão do Bluetooth 1.0 e 1.0B é de 721 kb/s.

2.2.2 Bluetooth 1.1

Em fevereiro de 2001 foi lançada a versão 1.1 que marca o estabelecimento do *Bluetooth* como padrão IEEE 802.15. Além disso, foram solucionados alguns dos problemas encontrados na primeira versão e também foi implementado o sistema RSSI (*Received Signal Strength Indication*) que mede a potência de recepção do sinal. A velocidade padrão do *Bluetooth* 1.0 que é de 721 kb/s foi mantida.

2.2.3 Bluetooth 1.2

Dois anos e meio após o lançamento da versão 1.1, a versão 1.2 do *Bluetooth* foi lançada com algumas novidades em relação à versão anterior, pois além de possuir conexões mais rápidas entre dispositivos, também conta com uma melhor proteção contra interferências no sinal, suporte aperfeiçoado a *scatternets* e processamento de voz mais avançado.

2.2.4 Bluetooth 2.0 + EDR (*Enhanced Data Rate*)

Esta versão do *Bluetooth* foi lançada oficialmente em novembro de 2004 com correções de falhas da versão 1.2 e com importantes aperfeiçoamentos à tecnologia, como por exemplo, a diminuição do consumo de energia, (consumia menos energia dos dispositivos que se conectavam ao *Bluetooth*) e teve um aumento significativo na velocidade de transmissão de dados para até 3 megabytes por segundo (Mb/s), 2,1 Mb/s efetivos.

Esse aumento na velocidade de transmissão se deve ao fato de que o Bluetooth 2.0 passou a contar com o padrão EDR que consegue praticamente triplicar a taxa de transferência de dados. Na verdade esse elemento é opcional, pois um dispositivo Bluetooth 2.0 não necessita do EDR para funcionar, porém se o dispositivo desprezar o uso do EDR, as características de evolução da versão 1.2 para a versão 2.0 são mantidas, com exceção da velocidade de transmissão que é de 721 kb/s, a mesma da versão anterior.

2.2.5 Bluetooth 2.1 + EDR

A versão 2.1 do *Bluettoth* foi lançada quase três anos mais tarde, em agosto de 2007, com a ampliação de mais informações nos sinais Inquiry (que permite uma seleção mais apurada dos dispositivos antes de estabelecer uma conexão entre eles), também houve melhorias nos procedimentos de segurança para a conexão, como por exemplo, o uso de senhas, e um melhor gerenciamento no consumo de energia dos dispositivos. A velocidade de transmissão de dados se manteve.

2.2.6 Bluetooth 3.0 + HS (High Speed)

Em abril de 2009, foi lançada a versão 3.0 que tem como principal atrativo a alta taxa de velocidade de transferência de dados, sendo superior a versão 2.1 mesmo com o uso de EDR. A velocidade chega à 24 Mb/s, 21 Mb/s maior que a versão anterior. Essas altas velocidades da versão 3.0 só podem ser alcançadas em dispositivos compatíveis com as instruções *HS*, característica equivalente à relação entre o Bluetooth 2.0 (ou 2.1) e o EDR. Outra vantagem obtida nesta nova versão é o controle mais inteligente do uso de energia do dispositivo e apesar da grande evolução, a versão 3.0 é compatível com as versões anteriores.

2.2.7 Bluetooth 4.0

A última versão do *Bluetooth* foi anunciada em dezembro de 2009 e a principal diferença com relação à versão anterior é a economia de energia. Esta versão pode exigir menos energia do dispositivo quando o mesmo está “parado”, sem estar sendo utilizado. Essa característica passou a ser interessante para telefones celulares que consomem muita energia quando o *Bluetooth* não está sendo utilizado, mas permanece ativo no aparelho.

Com essa grande economia de energia, tornou-se possível a utilização da tecnologia *Bluetooth* em diversos dispositivos que trabalham com pouco uso de energia como, por

exemplo, fones de ouvido que utilizam uma pequena bateria, suficiente para o funcionamento do *Bluetooth* 4.0.

É importante frisar que apesar de existir várias versões do *Bluetooth*, não significa que dispositivos com versões anteriores não possam estabelecer conexão com as versões mais atuais, embora possa haver exceções. Porém, se um dispositivo que possui a versão 2.0, por exemplo, for conectado a um dispositivo que dispõe de uma versão superior, a velocidade de transmissão de dados entre eles será a mesma do dispositivo com a versão inferior, neste caso, a mesma da versão 2.0. (INFOWESTER, 2013)

2.2.8 Espectro e Interferência

A faixa de frequência com que o *Bluetooth* opera é entre 2,4 e 2,485 GHz, esta faixa de frequência não é licenciada na maioria dos países, sendo conhecida por banda ISM (*Industrial, Scientific and Medical*) usando espectro de difusão, salto de frequência, comunicação *full-duplex* e uma taxa nominal de 1600 saltos por segundo.

Além disso, o *Bluetooth* utiliza uma tecnologia denominada AFH (*Adaptive Frequency Hopping)*. Esta tecnologia foi projetada com intuito de diminuir a interferência entre as tecnologias sem fio que utilizam o mesmo espectro de 2,4 GHz. A AFH opera dentro do espectro para aproveitar toda a faixa de frequência que está disponível, ela detecta outros dispositivos que estão usando o mesmo espectro, evitando o uso das frequências já ocupadas. (SOUSA, 2012)

2.2.9 Padrões de Comunicação

O *Bluetooth* utiliza dois padrões de comunicação: o padrão síncrono conhecido como SCO (*Synchronous Connection-Oriented*) e assíncrono ACL (*Asynchronous Connection-Less)*.

O SCO é um link simétrico que provê de uma conexão ponto-a-ponto entre os dispositivos conectados e é utilizado para transmitir dados contínuos, como por exemplo, a voz. Se houver perda de dados de voz, o receptor acaba produzindo o som como um ruído. A taxa de transmissão de dados no modo SCO é de 432 kbps, sendo de 64 kpbs para voz.

Já o link assíncrono ACL provê de uma conexão assimétrica com multiponto, e se beneficia com relação ao SCO por usar *slots* para a transmissão de dados. Além disso, utiliza chaveamento por pacotes, o que permite o reenvio de pacotes e dados perdidos, sendo útil

para aplicações que necessitam a integridade dos dados transferidos. A taxa de transmissão de dados no modo ACL pode alcançar 721 kbps. (TELECO, 2014)

2.2.10 Modulação

A modulação utilizada pelo *Bluetooth* é a FSK (*Frequency Shift Keying)*, técnica semelhante à Modulação em Frequência (FM), porém a FSK desloca a frequência por meio de dois pontos separados e fixos, enquanto a modulação FM vária a frequência instantaneamente.

Também existe um modo opcional denominado EDR (*Enhanced Data Rate)*, cujo utiliza as modulações PSK (*Phase Shift Keying)* na sequência de sincronização, carga útil e trailer e GFSK (*Gaussian Frequency Shift Keying)* no código de acesso e no cabeçalho. A taxa de transmissão de símbolos é de um mega símbolos por segundo (Msps) tanto em PSK quanto em GFSK. A taxa de transmissão de bits no modo FSK é aproximadamente um Megabits por segundo (Mbps) e no modo EDR chega até 3 Mbps. (SOUZA, 2012)

2.3 SISTEMA DE RECONHECIMENTO DE VOZ

Para que um dispositivo possa reconhecer uma voz, é necessário encadear uma sequência de passos que se inicia com o som captado por algum tipo de aparelho que possua esta tecnologia e termine com uma palavra ou um texto impresso no mesmo.

Na maioria dos celulares, por exemplo, este processo é feito pela empresa *Google* que oferece vinte e oito tipos diferentes de comandos de voz para sistema operacional Android e dezoito para celulares com sistema operacional iOS. (TECHTUDO, 2016)

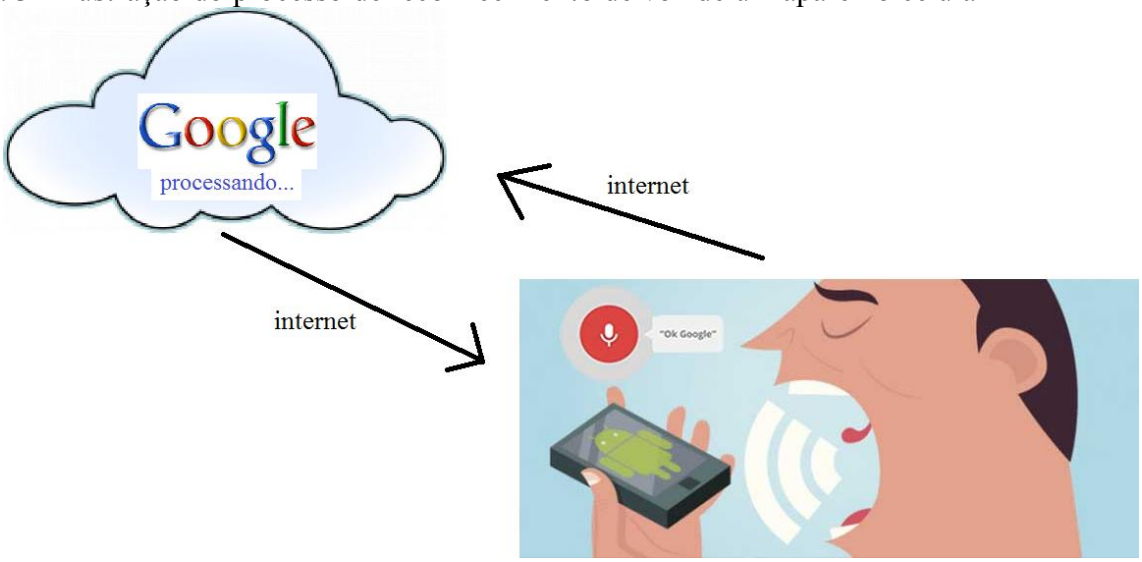
Quando a voz é captada pelo aparelho, o som é recebido via internet pelo sistema de reconhecimento de voz que converte as vibrações sonoras em ondas analógicas e são transformadas em dados digitais. Além disso, este sistema também filtra o som recebido e o separa de interferências e ruídos indesejados. (TECMUNDO, 2009)

O som captado pelo sistema é adaptado para tocar em um nível de volume e velocidades constantes. Estes ajustes são necessários para igualar o som recebido pelo dispositivo com a amostra que está contida na memória do sistema de reconhecimento de voz.

Também é feito uma divisão do som em fonemas (“sílabas sonoras” que formam as palavras). O sistema compara a sequência em que os fonemas são captados com palavras e frases armazenadas na memória. Para identificar palavras que são pronunciadas com sotaques diferentes, este sistema utiliza modelos estatísticos e simula combinações mais prováveis de fonemas. (MUNDO ESTRANHO, 2016)

Por fim, o som retorna para o dispositivo via internet decodificado e transformado em palavras ou frase. Na Figura 3 é ilustrado o processo de reconhecimento de voz de um aparelho celular.

Figura 3 - Ilustração do processo de reconhecimento de voz de um aparelho celular



Fonte: Produção do próprio autor

2.4 ARDUINO

A placa Arduino foi criada na Itália pelo professor Massimo Benzi no ano de 2005. Naquela época o professor desejava ensinar a seus alunos conceitos de programação e eletrônica, mas enfrentava algumas dificuldades pelo fato de que no mercado não havia dispositivos de baixo custo para que todos seus alunos pudessem possuir, cada um sua própria placa. Foi então que Benzi teve a ideia de criar uma placa de baixo custo para fins didáticos com intuito de que todos seus alunos pudessem ter condições de adquirir, facilitando a aprendizagem dos mesmos. Após o feito, em 2006 Benzi recebeu uma menção honrosa na categoria "Comunidades Digitais" pelo sucesso da placa nomeada Arduino.

A placa criada por Benzi é considerada "*Open Sorce*", o que significa que qualquer pessoa ou empresa tem a permissão de copiá-la, alterá-la, ou vende-la sem a necessidade de pagar direitos autorais. Qualquer outra placa criada com a mesma estrutura do Arduino e que utiliza a linguagem padrão da placa (*Arduino Programming Language*) pode realizar as mesmas funções da original. Tal linguagem de programação é baseada nas linguagens C e C++. (ARDUINO APRENDIZES, 2014)

Hoje no mercado existem vários modelos de Arduino, como por exemplo: Arduino UNO, Arduino Leonardo, Arduino Mega, Arduino Nano, entre outras. Todas elas basicamente com a mesma estrutura, diferindo apenas em algumas funções peculiares. Além das placas também há inúmeros componentes extras que podem ser conectados ao Arduino que são conhecidos como *Shields* e vão desde sensores eletrônicos (sensores de presença, temperatura, pressão, entre outros) a módulos Wi-fi e *Bluetooth*, o que ampliam consideravelmente sua área de atuação em desenvolvimento de projetos.

Para o uso da placa é necessário apenas um conhecimento básico em eletrônica e programação, tendo em vista que para a criação de algum projeto no Arduino é necessário conhecer o funcionamento de seu *hardware* (placa) e *software* (programação).

Na internet há o site oficial do Arduino que pode auxiliar no desenvolvimento de projetos, contendo uma loja online para a compra de produtos originais, um amplo material para estudo, tutoriais e fóruns, onde é possível tirar todas as dúvidas na hora da criação. O logotipo do Arduino é mostrado na Figura 4 e está contido no site oficial e impresso em todos modelos existentes da placa.

Figura 4 - Logotipo Arduino

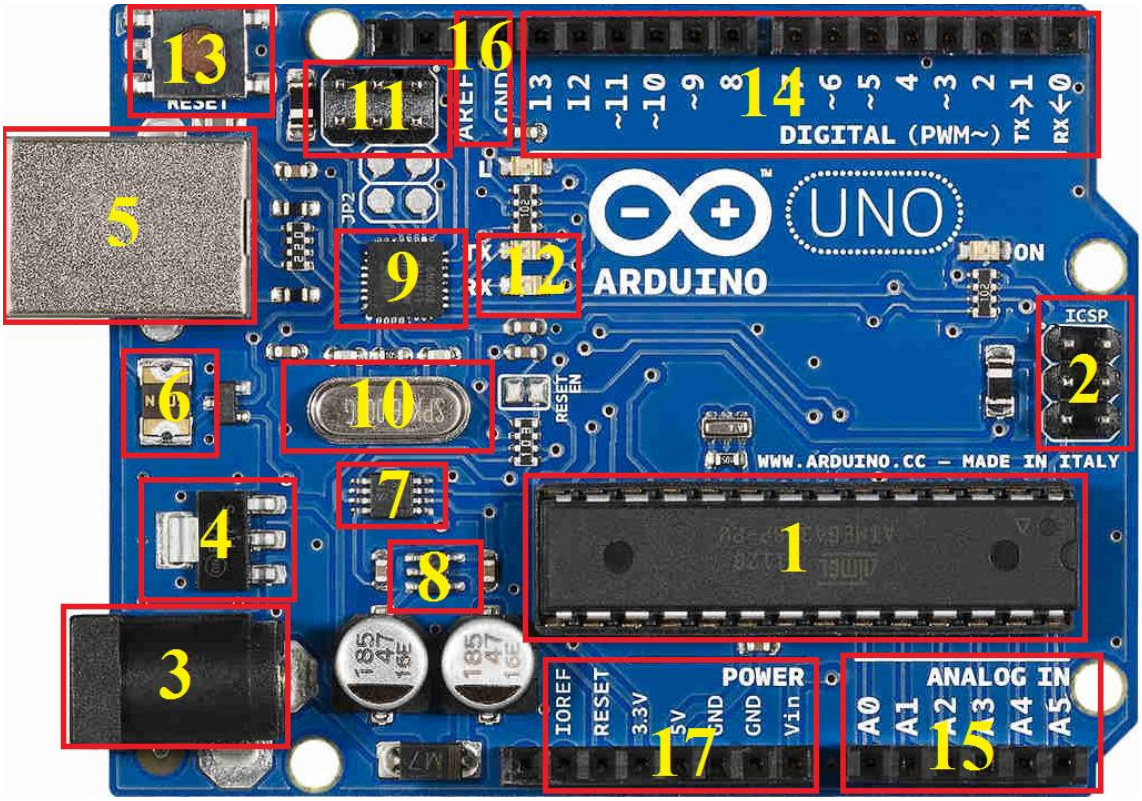
Fonte: https://www.arduino.cc/

2.4.1 Arduino UNO

A placa Arduino UNO possui diversos componentes, ressaltando o microcontrolador ATEMEGA 328 que possui 14 portas digitais de entrada, das quais seis podem ser usadas como saídas PWM. Possui seis entradas analógicas que também podem ser utilizadas como pinos de entradas e saídas digitais, um cristal de 16 MHz, conexão USB (*Universal Serial Bus*), um conector de alimentação, com conector para gravação IN-SYSTEM, e um botão de *reset*. Para utilizar o UNO, basta conectá-lo a um computador com um cabo USB ou ligá-lo com uma fonte DC ou mesmo em uma bateria. Na Figura 5 estão numerados os principais

componentes da placa e o Quadro 1 possibilita analisar suas especificações. (EMBARCADOS, 2013)

Figura 5 - Placa Arduino UNO



Fonte: Adaptada pelo autor

Quadro 1 - Componentes e suas especificações

Número	Componente	Especificação
1	Microcontrolador ATMEGA328	Principal componente da placa, “cérebro” do Arduino Uno
2	Pinos ISCP	Usado para programar diretamente o ATMEGA328
3	Conector Jack	Recebe energia da fonte externa
4	Regulador de tensão da fonte externa	Regula a tensão da fonte externa para uma tensão de saída de 5V
5	Entrada USB	Conecta a placa ao PC
6	Circuito de proteção USB	Protege a porta USB do computador
7	Circuito de seleção de fonte	Comuta a alimentação da USB para a fonte
8	Regulador de tensão de 3,3V	Regula tensão de saída no pino para 3,3V
9	Microcontrolador ATMEGA16U2	Responsável pela comunicação USB do Arduino UNO com o PC
10	Cristal oscilador de 16 MHz	Gera a frequência de operação do Arduino
11	Pinos ISCP	Usado para programar diretamente o ATMEGA16U2
12	LED’s Tx e Rx	Indicam a transmissão e recepção de dados
13	Botão Reset	Reinicia o programa instalado na placa
14	Conectores Digitais	Funcionam como entradas ou saídas de dados
15	Conectores Analógicos	Funcionam como entradas de componentes
16	Conector AREF	Configura a tensão de referência usada para entrada analógica
17	Conectores de alimentação	Usado para a conexão de shields e módulos

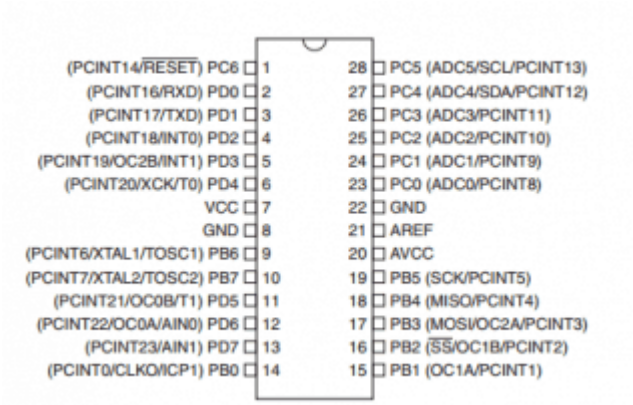
Fonte: Produção do próprio autor

2.4.1.1 Microcontrolador ATMEGA328

A placa Arduíno UNO tem como principal componente um microcontrolador da família ATMEL, o ATMEGA328, modelo de 8 bits e 32 kB de memória *flash*, a qual permite que múltiplos endereços sejam apagados ou escritos em uma só operação, 2 kB de memória RAM (*Random Access Memory)*, tipo de memória que permite a leitura e a escrita de dados, e

1 kB de memória EEPROM (*Electrically-Erasable Programmable Read-Only Memory)*, tipo de memória não volátil usada para armazenar e salvar pequenas quantidades de dados que precisam ser salvos quando a energia é removida. Este microcontrolador possui também um alto desempenho que pode executar instruções com um ciclo de *clock* e faz com que o mesmo alcance um Milhão de Instruções Por Segundo por Mega Hertz (MIPS/MHz), o que permite a criação de um simples programa para piscar um LED até um controle de um robô ou ainda um programa de controle de acesso por rede. Além disso, possibilita ao programador aperfeiçoar o projeto, combinando consumo de potência versus velocidade de processamento (EMBARCADOS, 2013). Sua pinagem é mostrada na Figura 6.

Figura 6 - Pinagem do microcontrolador ATMEGA328



Fonte: http://www.embarcados.com.br/arduino-uno/

Conectado ao ATMEGA328, está um conjunto de seis pinos nomeados de ISP e indicado com a sigla ICSP no Arduino UNO. Tanto a sigla ISP quanto a ICSP se referem a um mesmo conceito, a capacidade de transferir programas para um componente enquanto ele está instalado em um sistema, o que difere é que o ICSP transfere por meio de protocolos seriais.

Portanto no Arduino UNO, os pinos ICSP se referem à capacidade de programar diretamente os microcontroladores da placa usando o protocolo serial SPI, e além dos pinos de tensão e controle (VCC, GND e RESET), também estão presentes os três pinos necessários para uma conexão ponto-a-ponto usando SPI: MOSI, MISO e SCK, que são, respectivamente, os pinos pelos quais envia dados ao periférico, recebe dados do periférico e controla o pulso/*clock* da conexão (ARDUINO, 2016). Suas configurações são mostradas na Figura 7.

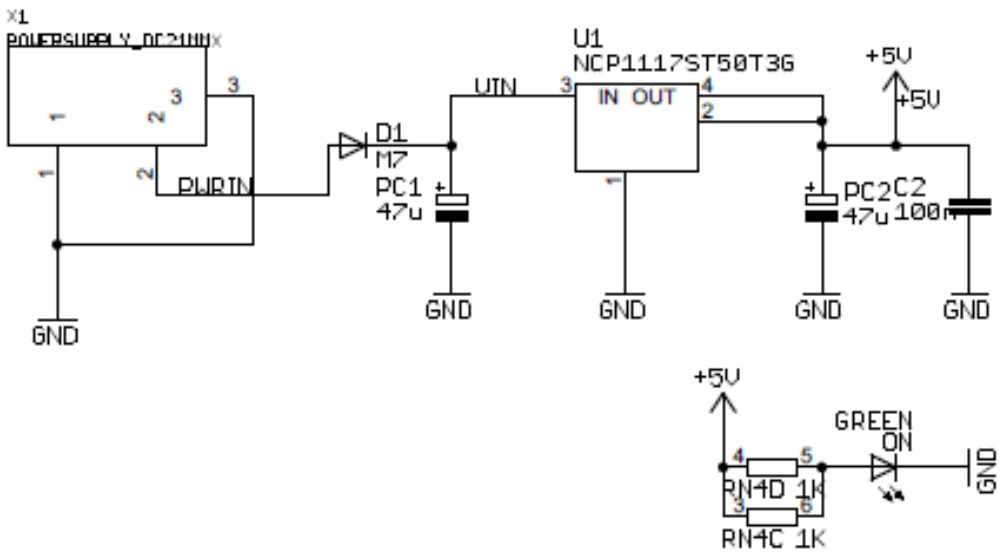
Figura 7 - Configurações dos pinos ICSP.

Fonte: https://www.arduino.cc/en/Reference/SPI

2.4.1.2 Alimentação

A alimentação externa da placa é feita através do conector *Jack* com positivo no centro. O valor ideal de tensão da fonte externa deve ser entre 7 V e 12 V, pois se for alimentada com uma tensão abaixo de 7 V, a tensão de funcionamento da placa, que no Arduino Uno é 5 V, pode ficar instável, ou se ela for alimentada com tensão acima de 12 V, o regulador de tensão da placa pode sobreaquecer e danifica-la (EMBARCADOS, 2013). O circuito deste regulador é mostrado na Figura 8.

Figura 8 - Circuito regulador de tensão.



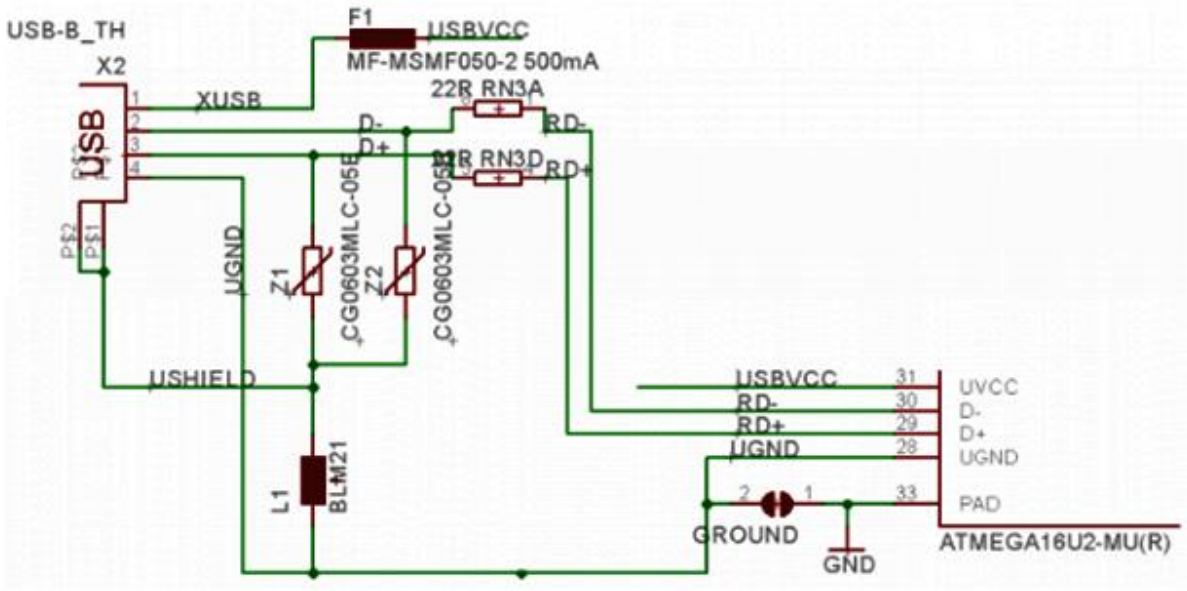
Fonte: http://www.embarcados.com.br/arduino-uno/

O circuito integrado (CI) contido no regulador de tensão é o NCP1117.

O Arduino UNO também pode ser alimentado através do cabo USB conectado a um computador, assim, a tensão não precisa ser estabilizada pelo regulador de tensão. O número 5 contido na Figura 5 indica a entrada do cabo USB na placa.

Também contido no Arduino, está o circuito de proteção da USB, que protege a porta USB do computador em caso de existir alguma irregularidade. Este circuito é mostrado a seguir na Figura 9.

Figura 9 - Circuito de proteção da porta USB do computador.

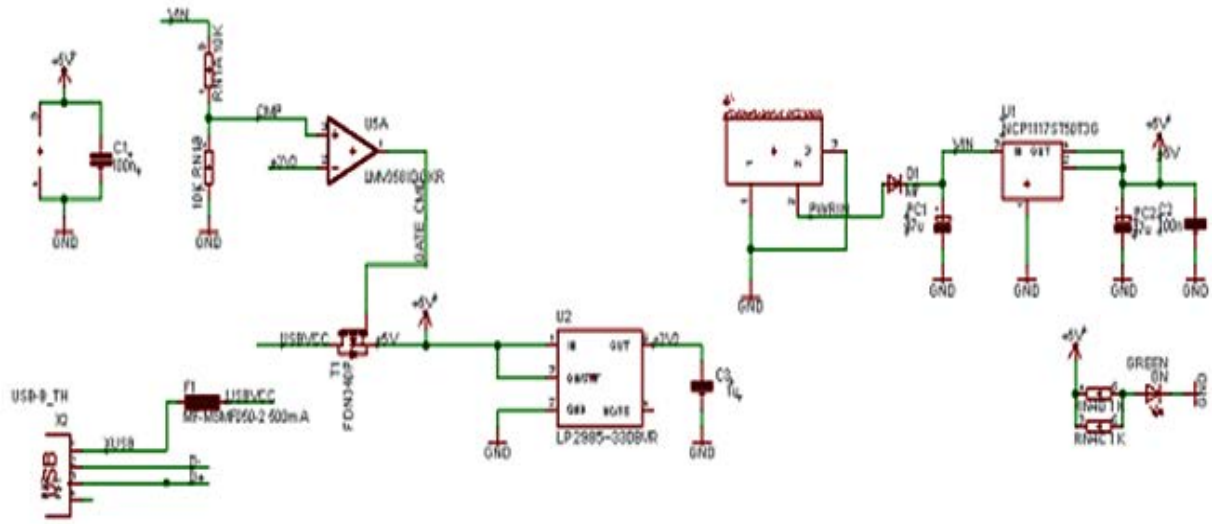


Fonte: http://www.embarcados.com.br/arduino-uno/

A Figura 9 do circuito de proteção da porta USB apresenta um fusível “F1” de 500 mA, ele impede que a porta seja queimada em caso de a corrente ultrapassar os 500 mA.

Também contido na placa, está um circuito de seleção de fonte que é usado para comutar a alimentação automaticamente. Esta mudança é realizada quando a placa está sendo alimentada pela porta USB e em seguida é conectada a ela uma fonte externa, ocorrendo a troca do uso da tensão para a fonte. Assim, a entrada USB passa a ser usada somente para o recebimento de dados (EMBARCADOS, 2013). Este circuito é apresentado na Figura 10.

Figura 10 - Circuito de seleção de fonte.



Fonte: http://www.embarcados.com.br/arduino-uno/

Além do regulador de tensão para fonte externa, a placa Arduino UNO dispõe de um regulador de tensão de 3,3 V e que fornece uma corrente máxima de 50 mA. Este regulador é usado para alimentar circuitos ou *Shields* que necessitam de uma tensão menor que a tensão de funcionamento da placa (5V).

2.4.1.3 Microcontrolador ATMEGA16U2

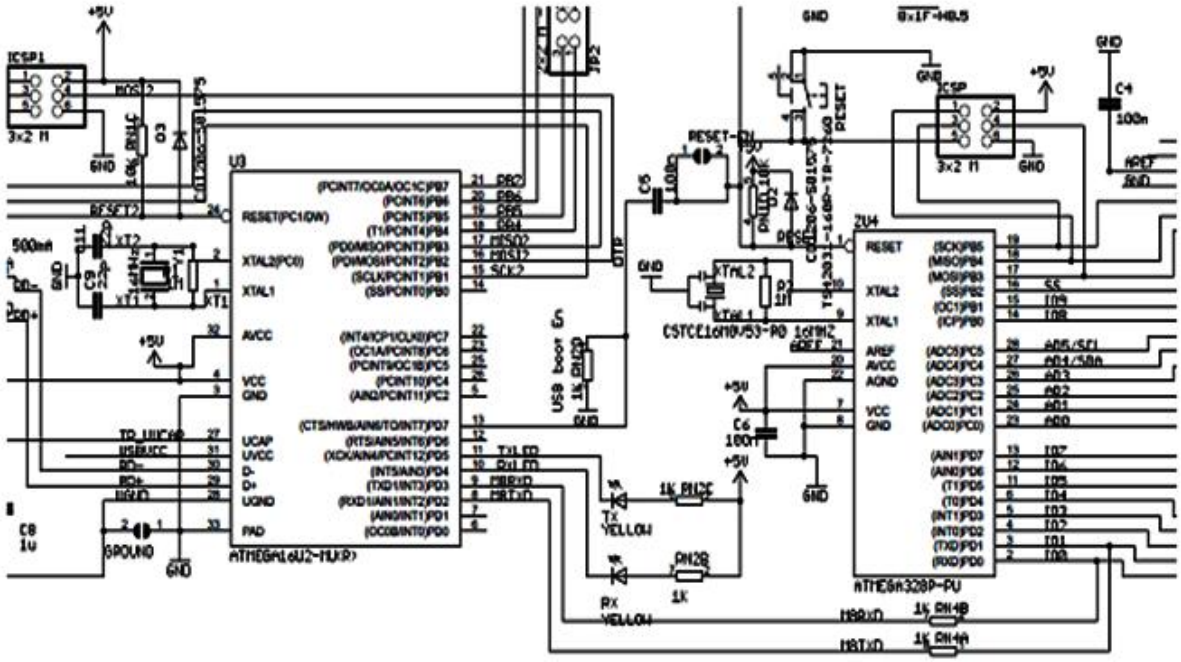
O microcontrolador ATMEGA16U2 e o cristal oscilador de 16 MHz são os responsáveis pela comunicação USB do Arduino UNO com o computador, possibilitando o *upload* do código criado pelo usuário no PC para a placa (EMBARCADOS, 2013). Além disso, também há a conexão de pinos ICSP e suas configurações são as mesmas dos pinos ICSP ligados ao ATMEGA328 e mostrados na Figura 7, porém, a localização deste conjunto de pinos está indicada pelo número 11 na Figura 5.

Representados pelo número 12 na Figura 5, há dois LEDs (Tx e Rx) que são controlados pelo ATMEGA16U2 e indicam a transmissão e recepção de dados da placa para o PC.

No Arduino UNO existe a conexão entre os dois microcontroladores da placa, feita pelo canal serial dos mesmos. Além disso, há também a conexão do pino 13 do ATMEGA16U2 com o circuito de *reset* do ATMEGA328, o que significa que antes do código ser carregado, há automaticamente um *reset* na placa, apagando um eventual código já

existente no Arduino sem precisar pressionar o botão RESET contido na mesma. O circuito da Figura 11 a seguir mostra a conexão serial entre os microcontroladores.

Figura 11 - Conexão seria entre os microcontroladores ATMEGA328 e ATMEGA16U2.



Fonte: http://www.embarcados.com.br/arduino-uno/

2.4.1.4 Conectores

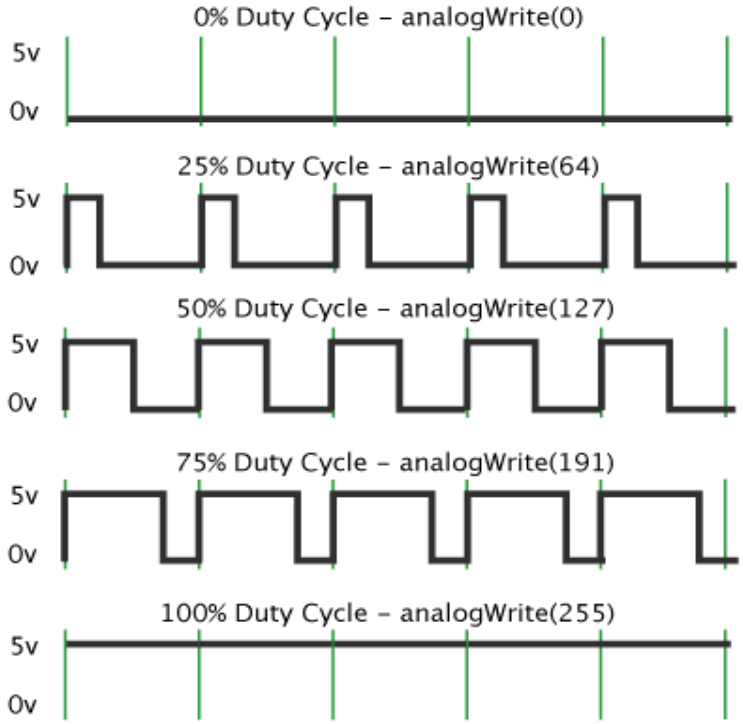
Existem diversos conectores no UNO e alguns diferem em suas funções. Há 14 conectores que são utilizados como entradas ou saídas digitais e são numerados de 0 a 13. Antes de utilizar um dos quatorzes conectores, é necessário configurá-lo como entrada ou saída digital e essa configuração é feita na programação. Por exemplo, se o conector 10 for utilizado para ligar um LED, ele deve ser configurado como saída através do comando "*pinMode (10, OUTPUT)*, ou se o mesmo for utilizado como entrada para ler o estado de um botão, por exemplo, ele deve ser configurado como entrada através do comando "*pinMode (10, INPUT)"*. Cada conector opera com uma tensão de 5V, podendo fornecer ou receber uma corrente máxima de 40 mA e possuem um resistor de *pull-up* interno de 20 kΩ, que pode ser habilitado através da programação para facilitar a ligação de teclas e sensores sem a necessidade de conectar um resistor externo (ARDUINO, 2016).

Alguns dos conectores digitais possuem funções especiais. Os conectores representados pelos números 0 e 1 podem ser usados para comunicação serial e são ligados ao

microcontrolador ATMEGA16U2. Já os conectores 2 e 3 podem ser configurados para gerar uma interrupção externa, através da função *attachInterrupt()*.

Os conectores 3, 5, 6, 9, 10 e 11 podem ser usados como saídas "PWM" (*Pulse Width Modulation*) que é uma técnica utilizada por sistemas digitais para variar o valor médio de uma forma de onda periódica. Esta técnica mantém fixa a frequência de uma onda quadrada e varia o tempo em que o sinal fica em nível lógico alto. Esse tempo é chamado de *duty cycle*, ou seja, o ciclo ativo da forma de onda (ARDUINO, 2016). A Figura 12 mostra algumas modulações PWM.

Figura 12 - Modulações PWM.



Fonte: https://www.arduino.cc/en/Tutorial/PWM

Analisando as formas de onda da Figura 12, nota-se que quando o *duty cycle* está em 0%, o valor médio na saída do conector é de 0 V e quando o *duty cycle* está em 100%, o valor médio na saída do conector é de 5 V. Portanto, para o cálculo do valor médio da tensão em um conector PWM, é usado a fórmula (1).

$$Vout = \left(\frac{duty\ cycle}{100}\right) * Vcc \qquad (1)$$

Onde:

- **Vou**t – tensão em volts na saída do conector PWM.
- **Duty cycle** – valor em porcentagem do ciclo ativo do conector PWM.
- **Vcc** – tensão de alimentação da placa em volts.

Um conector PWM pode ser usado para controlar a velocidade de rotação de motores, variar a luminosidade de um LED, gerar sinais analógicos e de rádio, entre outras funções.

Além dos conectores digitais, o Arduino UNO dispõe de seis conectores analógicos numerados de A0 a A5. Esses conectores são úteis para o uso de componentes como potenciômetros e sensores que leem valores variados em uma determinada faixa. Porém o microcontrolador do Arduino UNO trabalha internamente com dados digitais, portanto é necessário transformar um sinal analógico em um valor digital e está transformação é feita através de um conversor analógico digital ou conversor A/D.

Um conversor A/D quantifica o valor analógico conforme a quantidade de bits de sua resolução. Esta resolução é dada pela fórmula (2).

$$resolução = \frac{Vref}{2^n} \quad (2)$$

Onde:

- **Vref** – tensão de referência do conversor A/D.
- **n** – número de bits de resolução do conversor A/D.

O conversor A/D contido no ATMEGA328 possui 10 bits de resolução e trabalhando com a tensão de referência em 5 V, o menor valor à ser lido é mostrado na equação (1).

$$resolução = \frac{5}{2^{10}} = 4{,}88\ mV \quad (1)$$

Caso opte por trabalhar com a referência interna de 1,1 V, a resolução é dada pela equação (2).

$$resolução = \frac{1{,}1}{2^{10}} = 1{,}07\ mV \quad (2)$$

É notável que a resolução é bem menor para o uso da referência interna.

Caso a referência externa seja selecionada, a resolução dependerá do valor de tensão aplicada ao conector AREF (*Analog Reference*) identificado na Figura 5 pelo número 16.

Também estão contidos na placa, seis conectores de alimentação, que são utilizados para a conexão de *Shields* e módulos. Estes conectores são especificados a seguir.

- **IOREF -** Fornece uma tensão de referência para que *shields* possam selecionar o tipo de interface apropriada. Assim, os *shields* que são alimentadas com 3,3 V podem ser adaptados para serem utilizados em 5 V e vice-versa.

- **RESET -** está conectado a pino de RESET do microcontrolador. Pode ser utilizado para um *reset* externo da placa Arduino.

- **3,3 V.** - Fornece tensão de 3,3 V para alimentação de *shield* e módulos externos. A corrente máxima neste conector é de 50 mA.

- **5 V -** Fornece tensão de 5 V para alimentação de shields e circuitos externos.

- **GND –** Conectores de referência (*Ground*).

- **VIN -** Conector para alimentar a placa através de *shield* ou bateria externa. Quando a placa é alimentada por uma fonte externa, através do conector *Jack*, a tensão neste conector será a mesma da fonte.

2.4.2 Software

O software para programação do Arduino é uma IDE (*Integrated Development Environment*) que permite a criação de *sketches* que são transferidas do computador para a placa através de uma comunicação serial. A *sketch* é um código criado pelo usuário para coordenar o que a placa deve executar durante seu funcionamento.

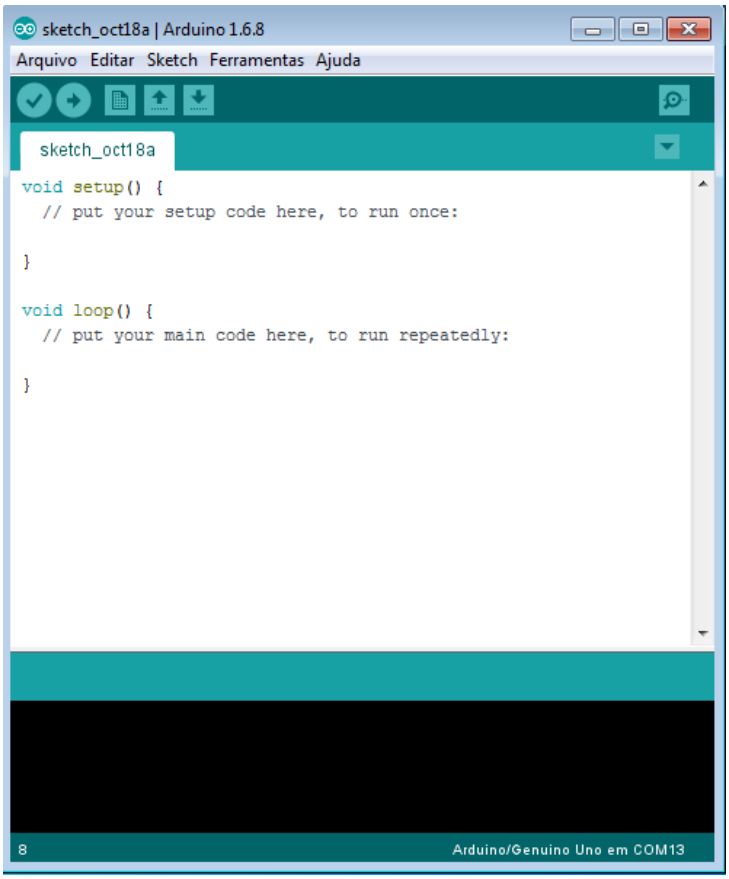
Ao pressionar o botão *upload* na IDE do Arduino, o código escrito é traduzido para a linguagem C e transmitido para o compilador avr-gcc, que traduz os comandos para uma linguagem que pode ser compreendida pelo microcontrolador da placa.

A IDE do Arduino possui sua própria linguagem baseada nas linguagens C e C++. Através dela é possível programar qualquer placa Arduino. O primeiro passo para a criação e transferência do código é selecionar na IDE, por meio do menu *Ferramentas/Placa,* o tipo de

placa usada, em seguida deve-se conectar a placa e o PC através do cabo USB e selecionar no menu *Ferramentas/Porta*, a porta em que o cabo foi conectado. Após estas configurações o ambiente está preparado para o uso e desenvolvimento da *sketch*, finalizando com a compilação e o *upload* do código na placa. (MCROBERTS, 2011)

A Figura 13 permite visualizar a IDE de versão 1.6.8 encontrada no site oficial do Arduino para *download*.

Figura 13 - IDE Arduino



Fonte: Adaptada pelo autor

No *Toolbar* há uma guia com o nome da *sketch* e uma barra de menus com os itens Arquivo, Editar, Sketch, Ferramentas e Ajuda. Os cinco botões encontrados logo abaixo fornecem acesso rápido as seguintes opções:

- Verificar (verifica se existe erro no código)
- Carregar (carrega o código para placa se não houver erro)
- Novo (abre uma nova *sketch*)

- Abrir (abre uma *sketch* feita e salva anteriormente ou os códigos exemplos já contidos na IDE)
- Salvar (salva a *sketch* ativo)

No canto superior direito da Figura 13 existe um botão que habilita o monitor serial, usado para mostrar todas as partes do código impressas na serial como, por exemplo, uma faixa de valores da leitura de um sensor.

2.5 APP INVENTOR 2

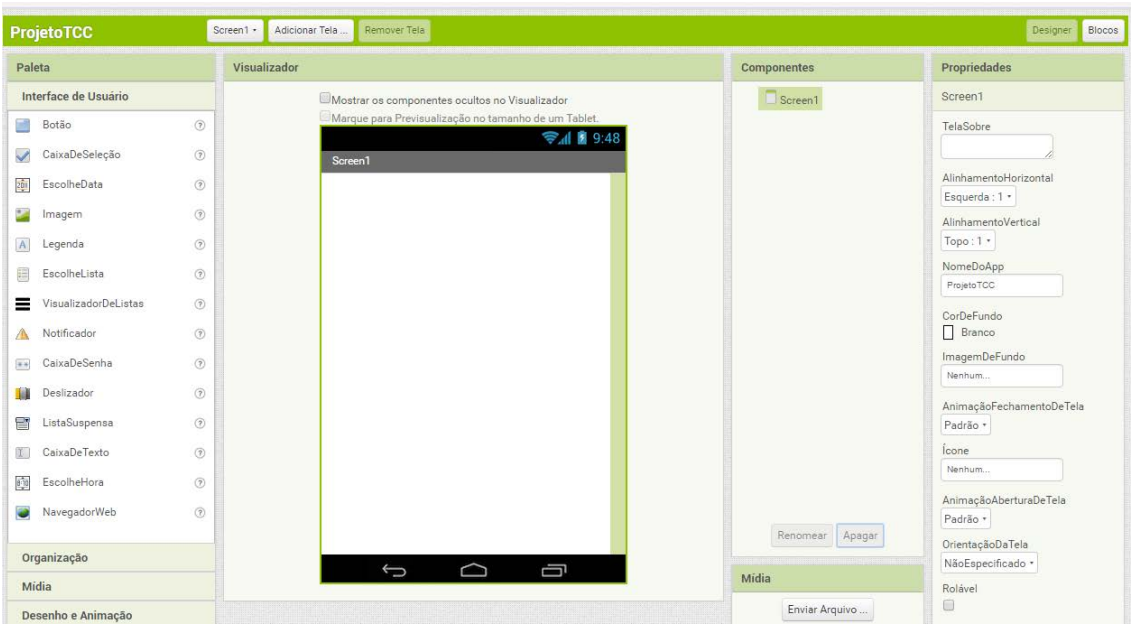
O App Inventor 2 é uma ferramenta desenvolvida pela empresa *Google* que permite a criação de aplicativos para celulares ou *tablets* com sistema operacional Android, sem que seja necessário um amplo conhecimento em programação. Esta ferramenta utiliza uma interface gráfica chamada "programação em blocos", que permite aos usuários arrastar e soltar blocos pré-programados, sem a necessidade de digitar uma linha de caracteres para a criação dos códigos. (TECHTUDO, 2010)

Antes de iniciar o desenvolvimento do aplicativo, é necessário entrar no site do MIT (*Massachusetts Institute of Technology)* App Inventor 2 e abrir uma conta *Google,* com isso o usuário poderá criar vários aplicativos e salvá-los mesmo que eles não sejam concluídos, podendo dar continuidade na criação posteriormente.

Para a criação do aplicativo é necessário seguir um ciclo de ações que começa com a montagem da interface (botões, caixas de textos, caixas de diálogos, entre outros), em seguida inicia-se a programação em blocos que atribui funções aos componentes existentes na interface, depois é feito a simulação através do simulador já contido nesta ferramenta e por fim, a transferência do aplicativo para o *smartphone* ou *tablet*.

A Figura 14 possibilita a visualização da tela inicial do App Inventor 2.

Figura 14 - Tela inicial do App Inventor 2



Fonte: Adaptada pelo autor

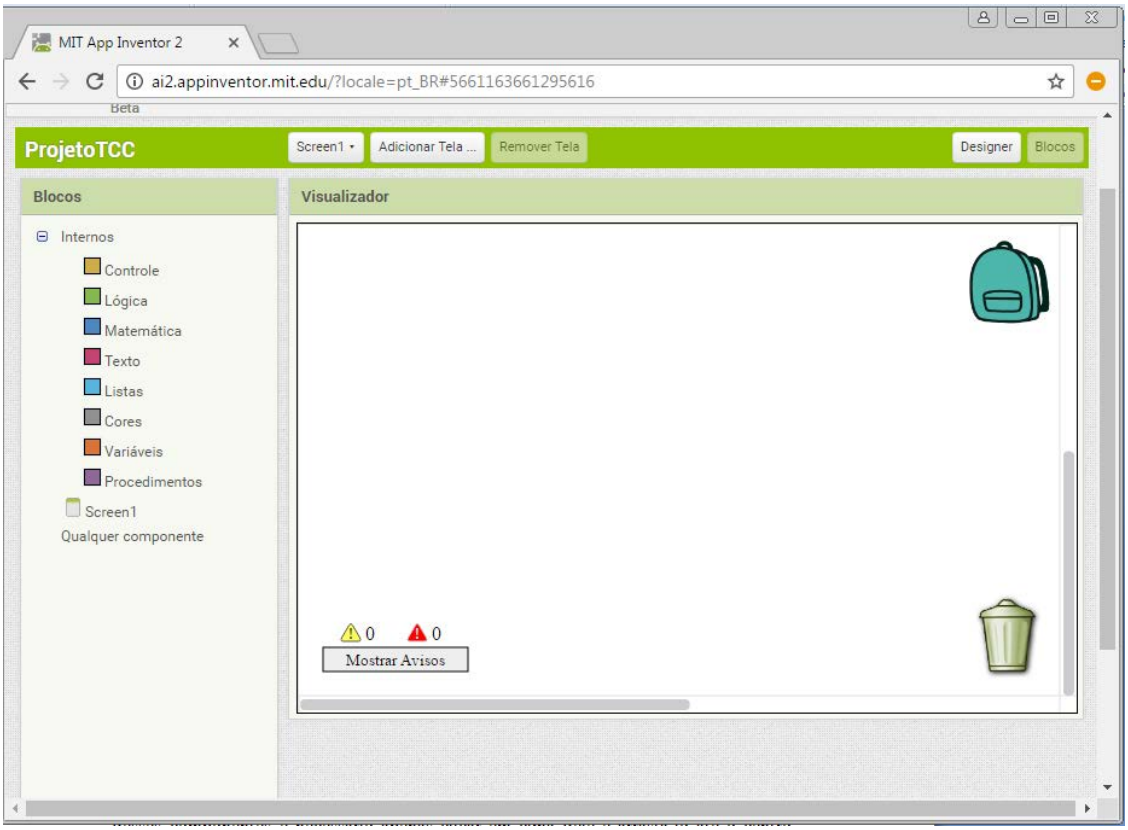
No canto esquerdo da tela inicial estão os componentes que podem ser usados para a construção da interface, a qual interage com o usuário. Para adicionar algum desses componentes basta clicar sobre ele e arrastá-lo até o centro que representa a tela de um aparelho eletrônico.

Além dos componentes de interface de usuário, existem diversas opções para personalizar o aplicativo que vão desde opções mais simples para organizar e enquadrar os componentes na tela até opções mais elaboradas como a adição de componentes que não aparecem na interface do celular, mas que fazem a conexão com a placa Arduino como, por exemplo, a conectividade via *Bluetooth* ou internet.

Do lado direito, aparecem os componentes já adicionados no aplicativo e também as opções para configurá-los, de modo que fiquem como o usuário deseja, alterando o tamanho, comprimento, fonte, cor, entre outras opções.

No canto superior direito está contido o botão “Blocos”, o qual comuta da tela de construção da interface para a tela de adição de blocos de programação, apresentada na Figura 15.

Figura 15 - Tela de adição de blocos de programação



Fonte: Adaptada pelo autor

No canto esquerdo da tela há opções para a programação que são construídos apenas com o arraste dos blocos. Estas opções variam desde blocos simples de lógica até blocos mais complexos como controle de variáveis e conexões com outros dispositivos.

Feito a interface e a atribuição de tarefa para os componentes é possível fazer a simulação do projeto para certificar que o aplicativo irá funcionar corretamente no aparelho, como planejado. Através da Figura 16 observa-se o simulador do App Inventor 2.

Figura 16 - Simulador do App Inventor 2

Fonte: Adaptada pelo autor

Após certificar o funcionamento correto do aplicativo através da simulação, o projeto está pronto para ser transferido e instalado no aparelho. Esta transferência pode ser feita via USB ou através do *QR code* gerado pelo site, mas para que a segunda opção seja realizada é necessário possuir um leitor de *QR code* próprio do App Inventor 2, encontrado na internet para *download*.

3 MATERIAIS E MÉTODOS

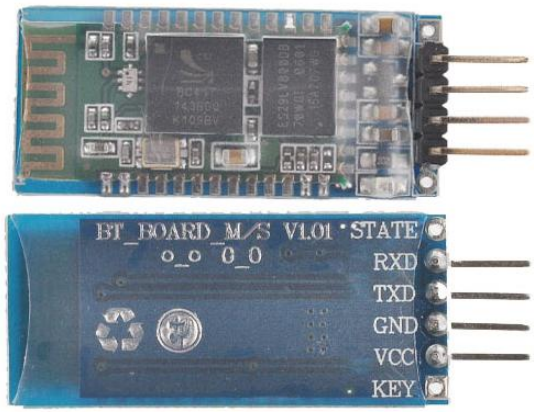
O projeto foi dividido em três etapas. A primeira envolve a criação da programação do Arduino e a simulação do programa, realizada no site 123D Circuits, onde é possível montar circuitos virtuais para simulação e verificação do funcionamento do código. Na segunda etapa foi realizado o desenvolvimento do aplicativo no site do MIT App Inventor 2 e transferido para um celular com sistema operacional Android. Por fim, a última etapa foi a criação de uma protótipo para a simulação do projeto em um quarto de hospital, onde componentes eletrônicos representam dispositivos elétricos que podem ser acionados através do aplicativo via *Bluetooth* e comando de voz. Este protótipo é mostrado na Figura 17 e cada componente será especificado posteriormente.

Figura 17 - Protótipo de um quarto de hospital

Fonte: Produção do próprio autor

Para que haja uma conexão *Bluetooth* entre um celular e o Arduino UNO é necessário o uso de um módulo *Bluetooth*. Este módulo é usado para uma comunicação *wireless* entre a placa e o aparelho e as informações recebidas são passadas para o Arduino via serial. Na Figura 18 é apresentado o módulo Bluetooth HC-06 utilizado neste projeto.

Figura 18 - Módulo Bluetooth HC - 06



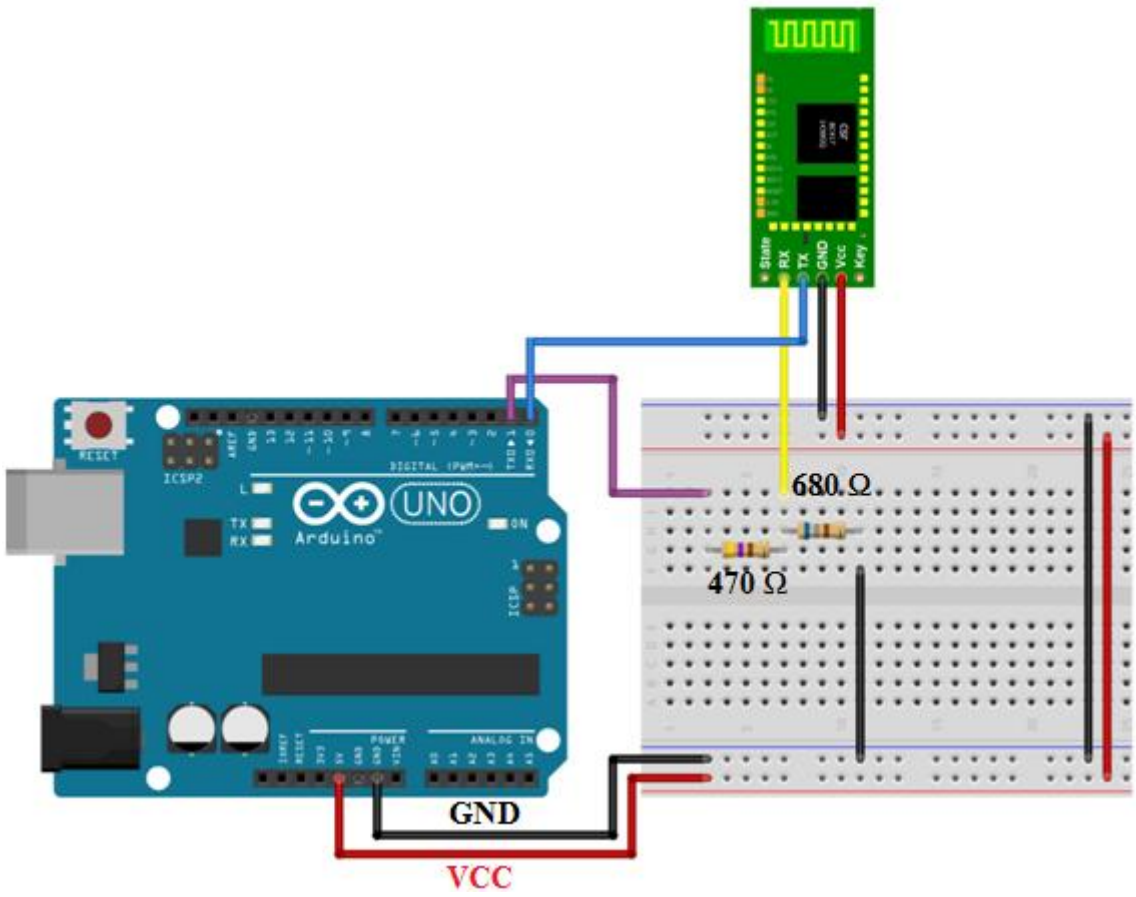
Fonte: https://arduinobymyself.blogspot.com.br/2012/11/arduino-bluetooth.html

O alcance deste módulo segue o padrão da comunicação *Bluetooth*, atingindo um raio de aproximadamente 10 metros. Além disso, funciona apenas em modo *slave* (escravo)**,** ou seja, permite que outros dispositivos se conectem a ele, mas não permite que ele próprio se conecte a outros dispositivos *Bluetooth*.

Como visto na Figura 18, ele possui quatro pinos para ligação que são respectivamente, VCC (alimentação de 3,6 à 6 V), GND**,** RX e TX, os dois últimos utilizados para comunicação com o Arduino via serial.

Para que a ligação do módulo com o Arduino seja feita corretamente, devem-se tomar alguns cuidados. Como especificado anteriormente, a alimentação pode ser feita entre os valores de tensão 3,6 a 6 V, mas no pino de comunicação RX é obrigatório que seja alimentado com no máximo 3,3 V. No pino TX não faz diferença, pois ele envia os 3,3 V para o Arduino que está preparado para receber tensões até 5 V. Já o pino RX recebe a tensão de 5 V do Arduino, podendo danificar a porta receptora do módulo. A solução para este problema é fazer um divisor de tensão no pino de recepção (BUILDBOT, 2015). Na Figura 19 é mostrada a forma correta de conectar o módulo *Bluetooth* ao Arduino.

Figura 19 - Conexão do módulo Bluetooth com o Arduino UNO



Fonte: Adaptada pelo autor

Os resistores utilizados neste projeto para o divisor de tensão possuem valores de 470 Ω e 680 Ω. A equação (3) mostra como é feita essa divisão.

$$Vout = 5 * \left(\frac{680}{470+680}\right) = 2{,}956\ V \qquad (3)$$

O resultado da equação (3) é menor que a tensão máxima permitida (3,3 V), portanto, o módulo pode ser conectado ao Arduino seguramente, sem que o receptor seja danificado.

Além do divisor de tensão, a Figura 19 mostra a ligação do pino TX do módulo à entrada digital 0 (RX) e a ligação do pino RX do módulo à entrada digital 1 (TX) da placa. Essas ligações são necessárias para que a placa e o módulo troquem informações.

Outro ponto importante destacar sobre o módulo *Bluetooth* é que para uma maior segurança, é necessário o uso de uma senha padrão antes de se conectar ao módulo e esta

senha pode ser configurada através da placa Arduino, com isso evita a conexão de um dispositivo intruso.

3.1 CIRCUITOS VIRTUAIS

A programação do Arduino está dividida em partes para que haja um melhor entendimento e todas elas estão nos apêndices A, B, C, D, E e F. É apresentado a seguir cada circuito virtual feito para a simulação do protótipo.

3.1.1 Circuito para iluminação

A iluminação do protótipo é realizada por uma lâmpada de 127 V AC e 9 W. Também está contido no circuito um módulo relé para o acionamento da lâmpada que tem a função de abrir e fechar contato para permitir ou interromper a passagem de corrente na carga. Na Figura 20 é apresentado o módulo relé.

Figura 20 - Módulo relé

Fonte: https://www.robocore.net/loja/produtos/modulo-rele.html

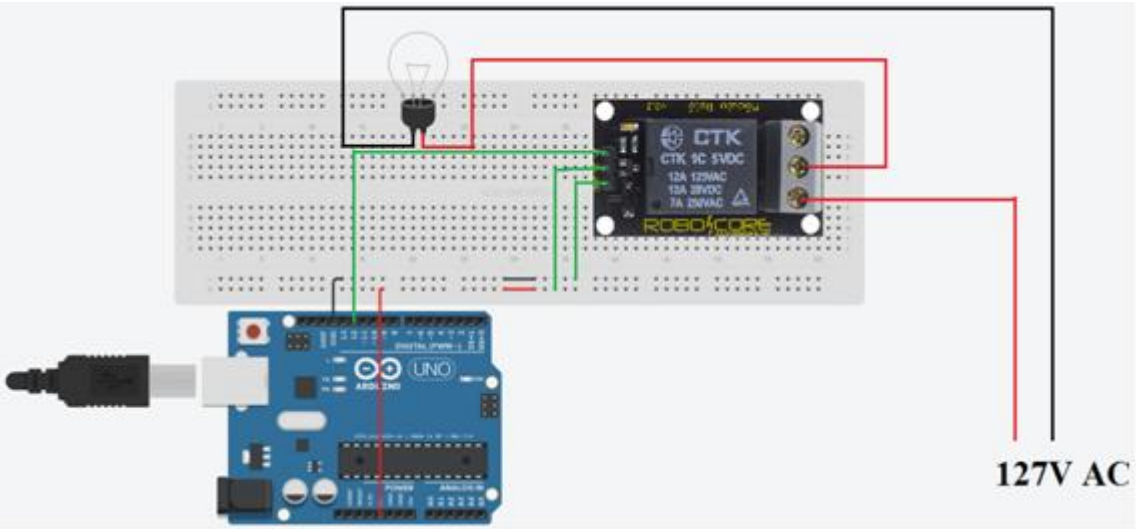
As especificações do módulo são dadas a seguir.

- Carga nominal do relé: 12 A – 125 V AC, 7 A – 250 V AC
- Carga nominal do módulo: 10 A
- Tempo de acionamento de contato: 10 ms

No módulo existem três pinos para a conexão com o Arduino. O pino GND é ligado ao GND da placa e o pino VCC é ligado a entrada de 5 V, pois a tensão de alimentação do

módulo é a mesma tensão de saída do Arduino. Já o pino IN é o de sinal digital e está conectado a entrada digital referente no código, neste caso, a entrada digital 12. Também há três terminais contidos no módulo: Comum, normalmente aberto e normalmente fechado. A conexão da lâmpada é feita pelos terminais “comum” e “normalmente aberto” como mostra o circuito de iluminação na Figura 21.

Figura 21 - Circuito para iluminação



Fonte: Adaptada pelo autor

É importante frisar que tanto no protótipo, quanto em uma situação real a tensão na lâmpada é de 127 V AC.

O código do Arduino para o acionamento do módulo relé é apresentado no APÊNDICE A – Código do Arduino para acionamento de um módulo relé.

3.1.2 Circuito para campainha

O circuito para campainha é composto por seis componentes. Um deles é o *buzzer*, componente eletrônico composto por duas camadas de metal e uma camada interna de cristal piezoelétrico que recebe uma tensão e através dela emite uma frequência sonora. A Figura 22 mostra a imagem de um *buzzer* de 5 V, o qual foi usado no protótipo para representar uma campainha.

Figura 22 - Buzzer

Fonte: http://www.baudaeletronica.com.br/buzzer-5v.html

A função do *buzzer* no protótipo é de alertar alguém (enfermeiro ou médico) que o paciente necessita de algum suporte.

Em uma situação real, o *buzzer* terá de ser substituído por uma sirene que emite uma frequência sonora maior. A Figura 23 apresenta uma sirene de 127 V AC.

Figura 23 - Sirene 127 V AC

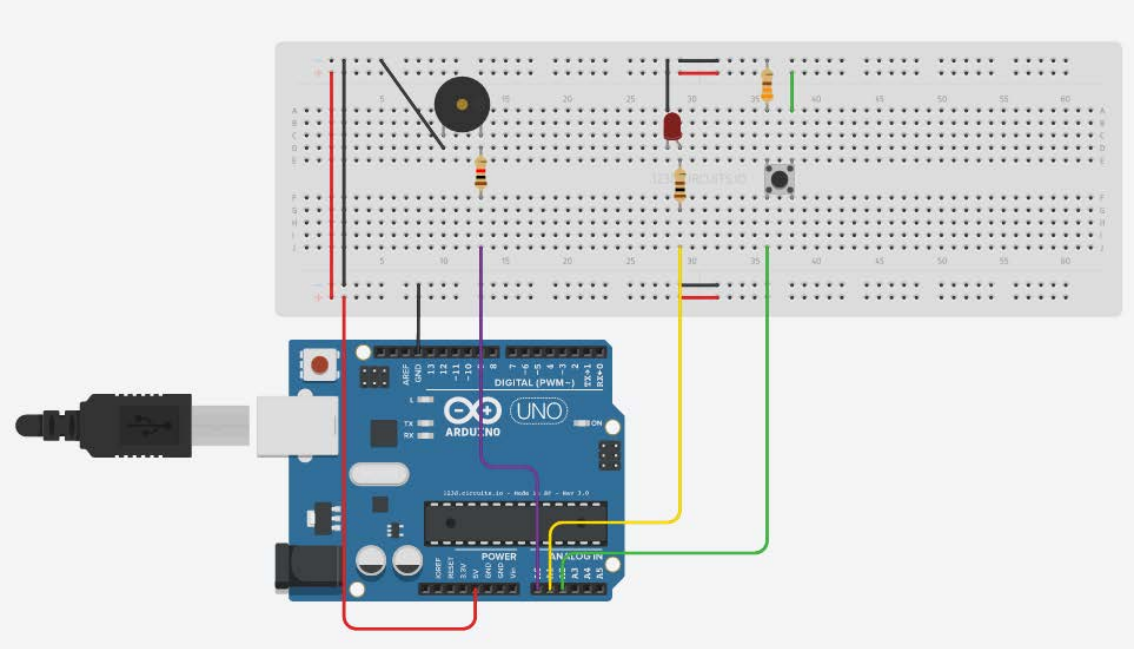
Fonte: http://produto.mercadolivre.com.br/MLB-719315938-sirene-piezoeletrica-127v-residencial-dni-4110-_JM

Outros dois componentes do circuito da campainha são um LED e um *push button*. O LED é usado no protótipo para sinalizar o quarto em que o paciente solicitou apoio tocando a campainha, assim, o médico e o enfermeiro poderão saber em qual quarto foram chamados e o *push button* é usado para apagar o LED. Em uma situação real, o LED e o *push button* ficariam do lado de fora do quarto, assim como estão localizados no protótipo.

Os três componentes restantes no circuito são resistores de 1 kΩ para o *buzzer,* 100 Ω para o LED e 330 Ω para o *push button.* A função dos resistores é limitar a corrente nos componentes para que os mesmos não sejam danificados.

A Figura 24 apresenta o circuito da campainha com os seis componentes.

Figura 24 - Circuito para campainha



Fonte: Adaptada pelo autor

Nota-se que os componentes estão ligados nas portas analógicas A0 (buzzer), A1 (LED) e A2 (*push botton*). Isso se deve ao fato de que há uma grande quantidade de componentes no projeto e as portas digitais foram ocupadas, então a solução foi usar as portas analógicas como portas digitais. Para que isso seja feito, deve-se declarar no código do Arduino as portas analógicas como digitais e o código do circuito para a campainha é encontrado no APÊNDICE B, que mostra como foi feita esta declaração.

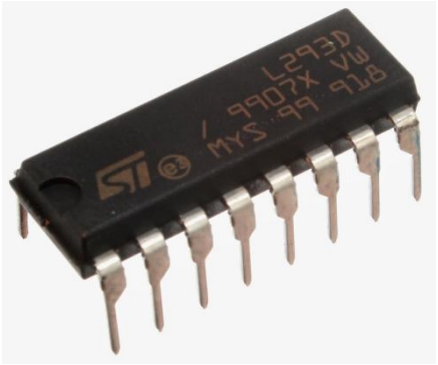
3.1.3 Circuito para atuação do ventilador

Foi usado no protótipo um motor DC de 5 V para representar um ventilador de teto que possui duas funções: ventilar e exaustar. Para que o motor DC funcione com a rotação bidirecional, foi usado o CI L293D titulado de “circuito integrado ponte H”. O motor DC de 5 V e o CI L293D são mostrados nas Figura 25 e Figura 26 respectivamente.

Figura 25 - Motor DC 5 V

Fonte: http://www.shelfkey.com/productSearch.aspx?id=89

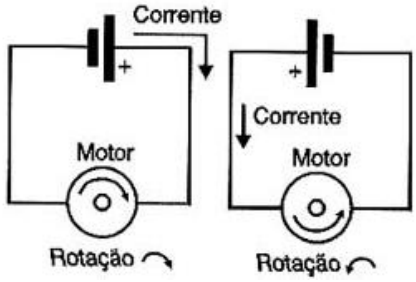
Figura 26 - Circuito integrado L293D



Fonte: http://www.arduinoecia.com.br/2014/04/controle-de-motor-cc-com-o-l293d-ponte-h.html

O sentido de rotação do motor depende do sentido de circulação da corrente pelos seus enrolamentos. Isso significa que ao inverter a polaridade de alimentação do motor irá inverter o sentido de rotação do mesmo (NEWTON C. BRAGA, 2014), conforme é mostrado na Figura 27.

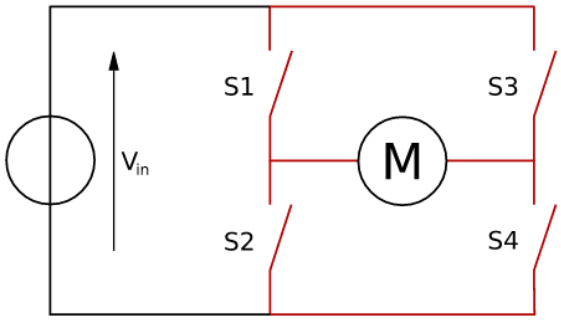
Figura 27 - Sentido de rotação conforme o sentido da corrente



Fonte: http://www.newtoncbraga.com.br/index.php/robotica/5166-mec068a

Uma ponte H pode ser formada por quatro chaves, fazendo com que duas das quatro se fechem de cada vez e de maneira que o sentido de circulação da corrente possa ser invertido. A Figura 28 apresenta um circuito ponte H formado por chaves que fazem o motor girar nos sentidos horário e anti-horário.

Figura 28 - Ponte H

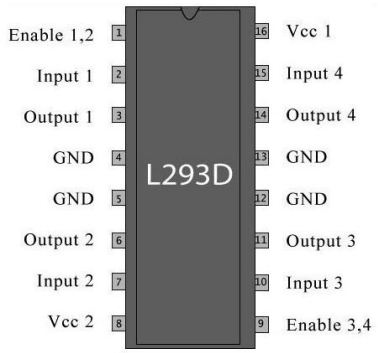


Fonte: https://upload.wikimedia.org/wikipedia/commons/d/d4/H_bridge.svg

Se as chaves S1 e S4 estiverem fechadas e S2 e S3 abertas, a corrente irá circular pelo motor no sentido horário e se as chaves S1 e S4 estiverem abertas e S2 e S3 fechadas, a corrente irá circular pelo motor no sentido anti-horário. Assim, o motor irá girar nos dois sentidos sem que exista uma inversão de polaridade da alimentação (MCROBERTS, 2011).

O CI L293D possui duas pontes H, podendo controlar um ou dois motores DC, porém é necessário o uso de uma fonte extra de 5 V ligada ao CI para alimentá-los. A pinagem desse CI é apresentada na Figura 29 e no Quadro 2 estão registrados os pinos do circuito integrado e onde devem ser conectados.

Figura 29 - Pinagem do CI L293D (ponte H)



Fonte: http://www.comofazerascoisas.com.br/motor-cc-dc-no-arduino-e-ponte-h-dupla-controle-de-velocidade-sentido-da-rotacao.html

Quadro 2 - Pinos do CI L293D e suas conexões

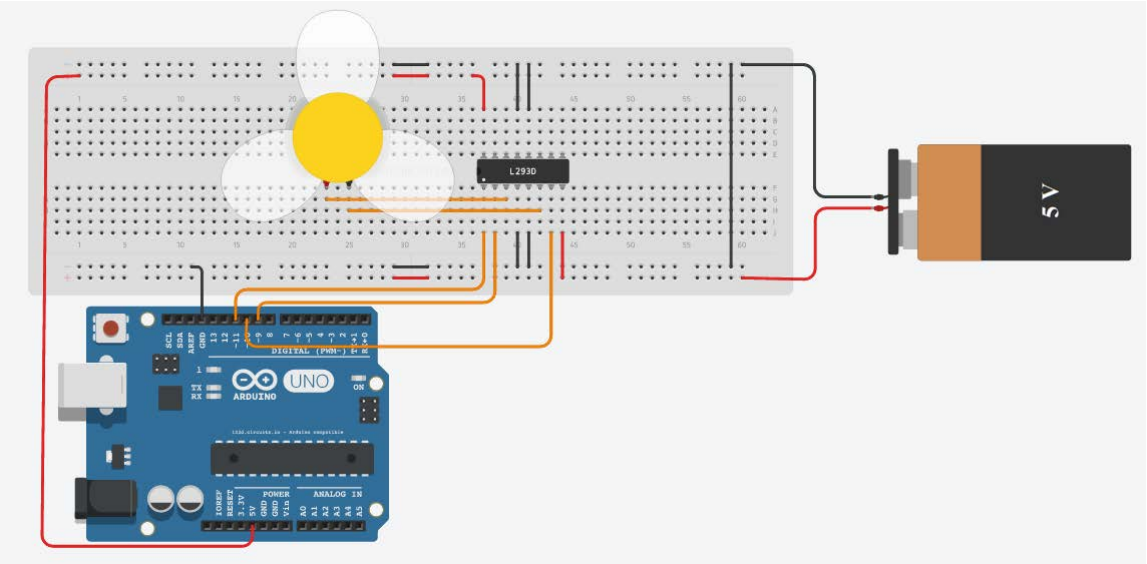
Pinos do CI	Conexões
1 – Enable 1,2	Entrada digital PWM
2 – Input 1	Entrada digital
3 – Output 1	Motor DC 1
4 - GND	GND do Arduino
5 - GND	GND do Arduino
6 – Output 2	Moto DC 1
7 – Input 2	Entrada digital
8 – Vcc 2	Positivo da fonte de alimentação do motor
9 – Enable 3,4	Entrada digital PWM
10 – Input 3	Entrada digital
11 – Output 3	Motor DC 2
12 - GND	Entrada GND do Arduino
13 – GND	Entrada GND do Arduino
14 – Output 4	Motor DC 2
15 – Input 4	Entrada digital
16 – Vcc 1	Entrada 5V do Arduino

Fonte: Adaptada pelo autor

Importante ressaltar que a fonte extra de 5 V é necessária para alimentar os motores DC pois a corrente fornecida pela placa não é suficiente para rotacioná-los. O positivo desta fonte é conectado ao pino 8 do CI L293D como mostrado no Quadro 2 e o negativo é ligado ao GND do Arduino.

Na Figura 30 é mostrado o circuito para atuação do ventilador, que contém uma bateria de 5 V representando a fonte extra.

Figura 30 - Circuito para atuação do ventilador



Fonte: Adaptada pelo autor

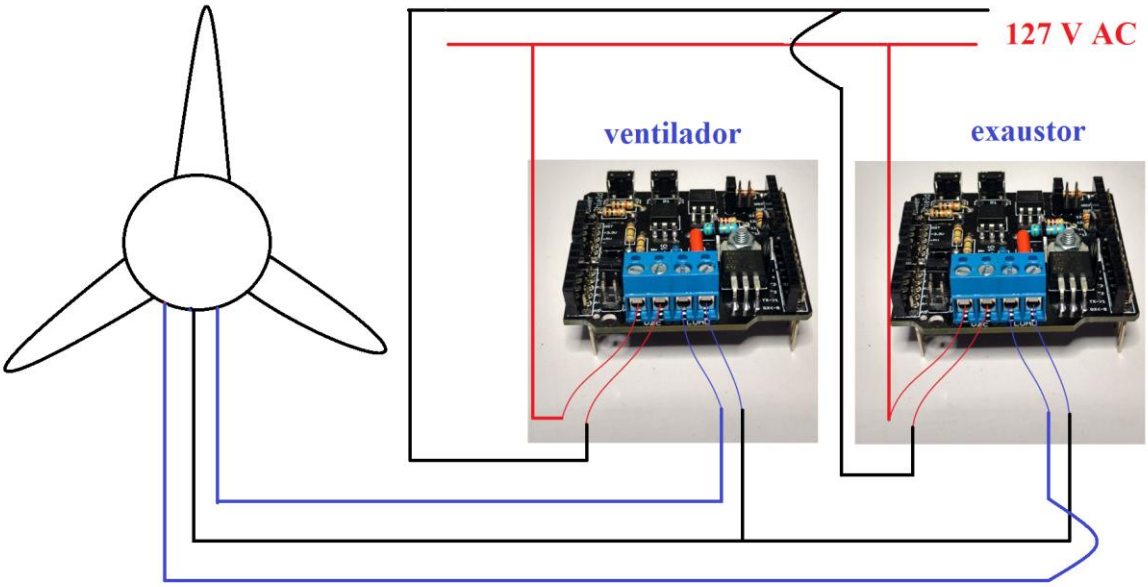
Os pinos 1, 2 e 7 do CI estão ligados respectivamente às entradas 11, 9 e 10 da placa.

O código para o funcionamento deste circuito é apresentado no APÊNDICE C.

Em uma situação real o CI L293D não seria utilizado, pois ele controla apenas motores DC e suporta uma tensão máxima de 36 V. Os motores dos ventiladores são AC e geralmente trabalham com tensões de 127 V ou 220 V, possuem dupla polaridade e podem girar nos sentidos horário e anti-horário. Porém, para fazer o controle de velocidade de um motor AC utilizando o Arduino é necessário o uso do Dimmer Shield.

O Dimmer Shield é um dispositivo que controla a potência de cargas indutivas e resistivas e pode ser acoplado ao Arduino, possibilitando que o controle de velocidade do ventilador seja feita via *Bluetooth*. Na Figura 31 é mostrado como é feita a ligação do Dimmer Shield com a rede elétrica e com o motor do ventilador.

Figura 31 - Dimmer Shield



Fonte: Produção do próprio autor

As especificações do Dimmer Shield são:

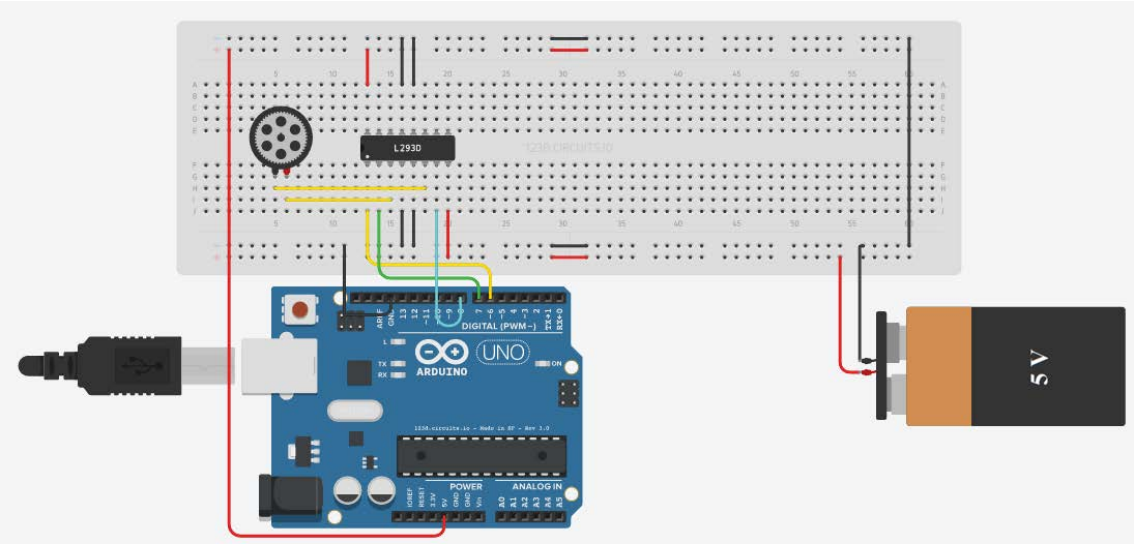
- Corrente máxima – 8 A
- Tensão máxima de pico – 400 V
- Potência máxima com dissipador incluso (127 V) – 400 W
- Potência máxima com dissipador incluso (220 V) – 800 W

As potências máximas podem atingir valores mais elevados se ao shield for conectado um dissipador maior.

3.1.4 Circuito para atuação da cortina

O circuito da cortina utiliza basicamente os mesmos componentes do circuito do ventilador, um motor DC de 5 V para abrir e fechar a cortina, um CI L293D e utiliza a mesma fonte extra para alimentação do motor. O circuito para a atuação da cortina montado no protótipo é apresentado na Figura 32.

Figura 32 - Circuito para atuação da cortina



Fonte: Adaptada pelo autor

Os pinos 1, 2 e 7 do CI estão conectados respectivamente às entradas 6, 7 e 8 da placa.

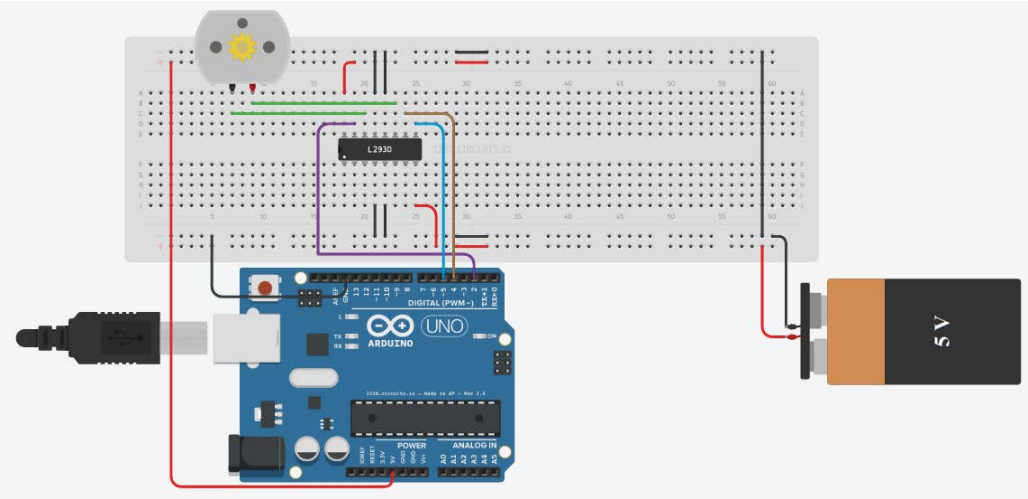
O código utilizado para o funcionamento deste circuito é apresentado no APÊNDICE D.

3.1.5 Circuito para atuação da cama

Assim como o circuito da cortina, o circuito da cama é igual ao do ventilador e possui também um motor DC de 5 V para subir e descer a cama, utiliza o mesmo CI ponte H do circuito da cortina e a mesma fonte extra de 5 V dos circuitos anteriores.

Na figura 33 é mostrado o circuito virtual da cama.

Figura 33 - Circuito para atuação da cama



Fonte: Adaptada pelo autor

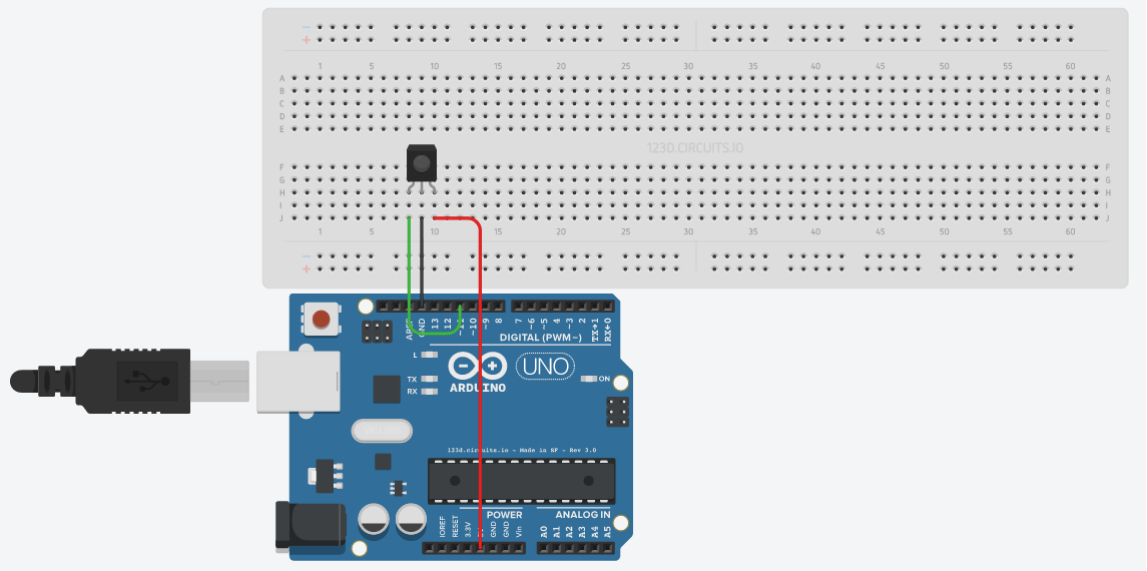
Os pinos 9, 10 e 15 do CI estão conectados respectivamente às entradas 4, 5 e 2 da placa.

O código relacionado a este circuito é apresentado no APÊNDICE E.

3.1.6 Circuito para TV

O circuito eletrônico para a TV é dividido em duas partes. A primeira contém apenas um receptor infravermelho conectado ao UNO que junto ao código pré-programado IRrecvDemo da biblioteca IR Remote do Arduino, capta os sinais infravermelhos emitidos pelo cotrole remoto da TV e apresenta os códigos de cada tecla do controle para serem clonados, sendo adicionados ao programa final. A Figura 34 apresenta o circuito de captação IR (*Infra red*) e em seguida é mostrado o código IRrecvDemo.

Figura 34 - Circuito de captação infravermelho



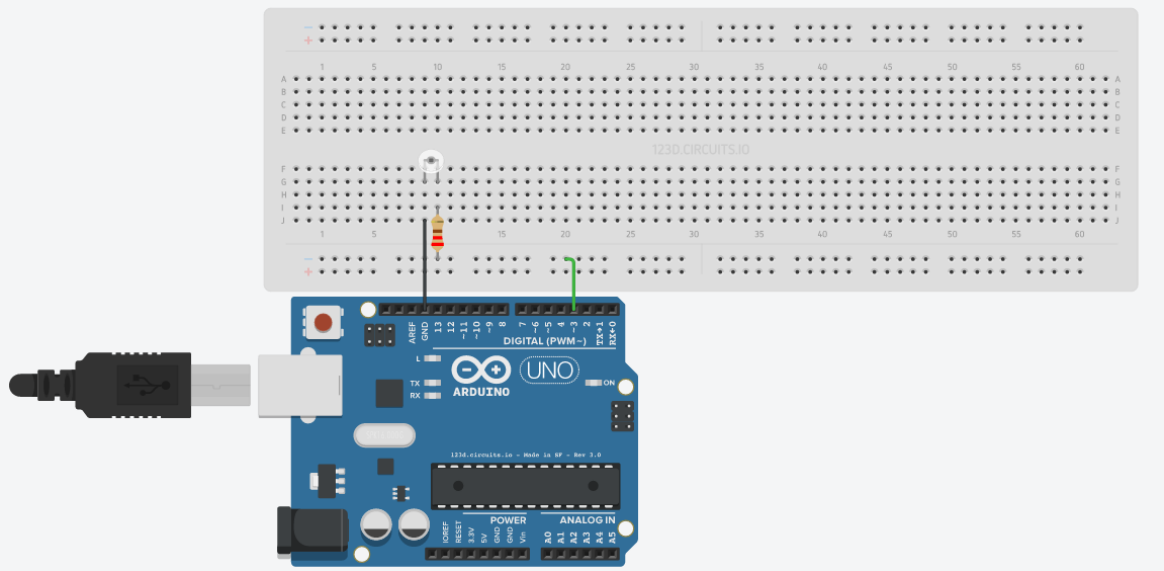
Fonte: Adaptada pelo autor

```
#include <IRremote.h>
int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
```

```
}
void loop() {
if (irrecv.decode(&results)) {
Serial.println(results.value, HEX);
irrecv.resume(); // Receive the next value
}
}
```

Após a captação dos códigos das teclas do controle remoto, o circuito da Figura 34 é desfeito e em seguida é montado o circuito da Figura 35, que contém um LED emissor infravermelho e um resistor de 220 Ω para a proteção do LED conectados à entrada digital 13 da placa.

Figura 35 - Circuito de emissão de sinal infravermelho



Fonte: Adaptada pelo autor

No APÊNDICE F é mostrado o código captado pelo receptor e adicionado ao programa final.

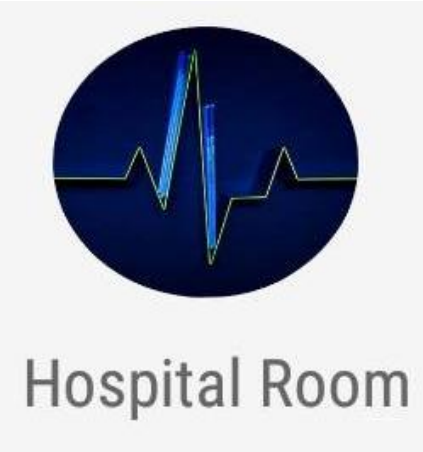
Importante ressaltar que cada marca de TV possui seu código específico.

3.2 DESENVOLVIMENTO DO APLICATIVO

Primeiramente foi desenvolvida a interface gráfica do aplicativo, a qual interage com o usuário. Após esta etapa, foi feita a atribuição de funções para os botões envolvendo a programação em blocos.

Também foi possível criar o ícone do aplicativo com o nome “Hospital Room”, o qual é visualizado no celular e apresentado na Figura 36.

Figura 36 - Ícone do aplicativo



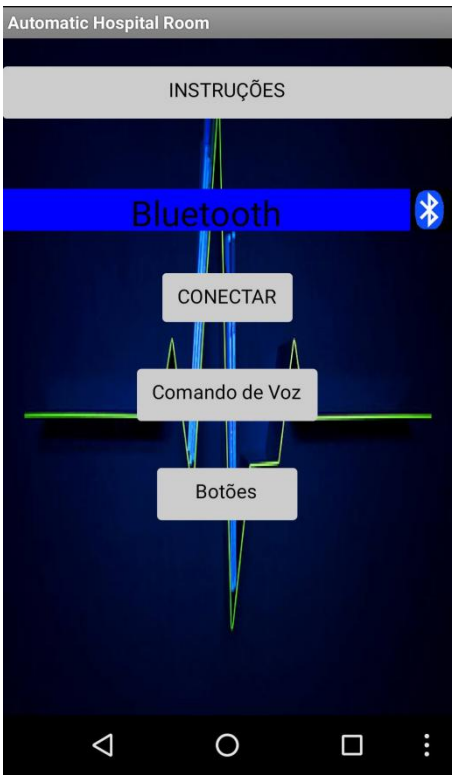
Fonte: Produção do próprio autor

Em seguida são mostrados os procedimentos realizados tanto para criação da interface, quanto para a programação em blocos.

3.2.1 Interface

A interface gráfica dispõe apenas de legendas e botões para que seja feita uma simples comunicação com o usuário. Para a criação dos componentes foi necessário apenas arrastá-los para o centro da tela como dito anteriormente e tal procedimento é feito de maneira bem simples. A Figura 37 apresenta a tela inicial do aplicativo.

Figura 37 - Tela inicial



Fonte: Produção do próprio autor

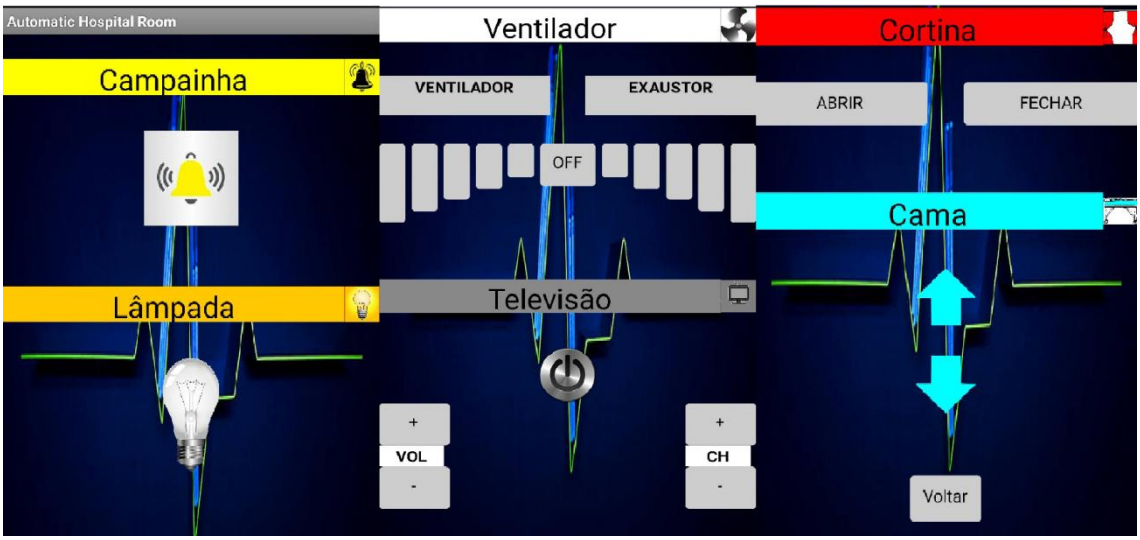
Como visto na Figura 37, há quatro botões na tela inicial que possuem as seguintes funções:

- Instruções – mostra uma mensagem com as frases necessárias a serem ditas no uso do comando de voz.
- Conectar – conecta o celular ou *tablet* com o módulo *Bluetooth*.
- Comando de voz – "chama" a tela instantânea do comando de voz.
- Botões – mostra a tela secundária do aplicativo, cuja qual contém todos os botões para acionamentos dos dispositivos.

Um detalhe importante é que os dispositivos podem ser acionados por comando de voz ou através de botões, mas ambos utilizam o sinal *Bluetooth* para o acionamento.

A Figura 38 mostra a tela secundária do aplicativo, a qual contém os botões de acionamento de cada dispositivo.

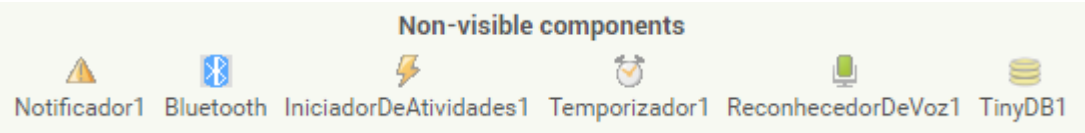
Figura 38 - Tela secundária



Fonte: Produção do próprio autor

Além dos componentes visíveis na tela principal e secundária, há também os componentes invisíveis que complementam o aplicativo e são mostrados na Figura 39.

Figura 39 - Componentes invisíveis



Fonte: Produção do próprio autor

Tais componentes possuem as seguintes funções neste aplicativo:

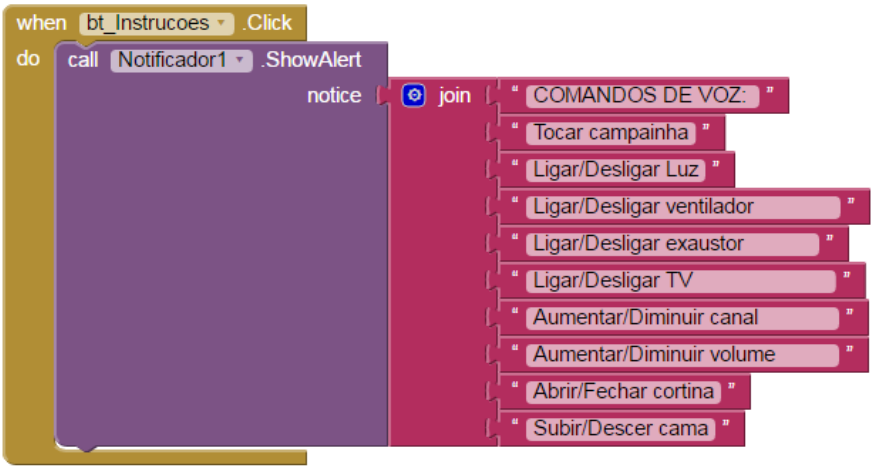
- Notificador: mostra uma notificação na tela do aparelho.
- Bluetooth: responsável pela conexão Bluetooth.
- Iniciador de atividades: ativa o Bluetooth quando o mesmo não está habilitado.
- Temporizador: responsável por receber o status das entradas do Arduino a cada 1 segundo.
- Reconhecedor de voz: ativa o comando de voz do aparelho quando o botão é pressionado.
- Tiny DB: banco de dados do aplicativo, usado para armazenar o status de cada botão.

3.2.2 Programação em blocos

A atribuição de funções dos botões foi feita através da programação em blocos. Cada botão pode possuir um ou mais blocos de programação, dependendo de sua ação. A seguir são mostrados os blocos de todos os botões do aplicativo.

- Botão *Instruções*: Este bloco apresentado na Figura 40 atribui a função de mostrar uma mensagem na tela toda vez em que o botão for pressionado. Essa mensagem apresenta as frases corretas que devem ser ditas no comando de voz.

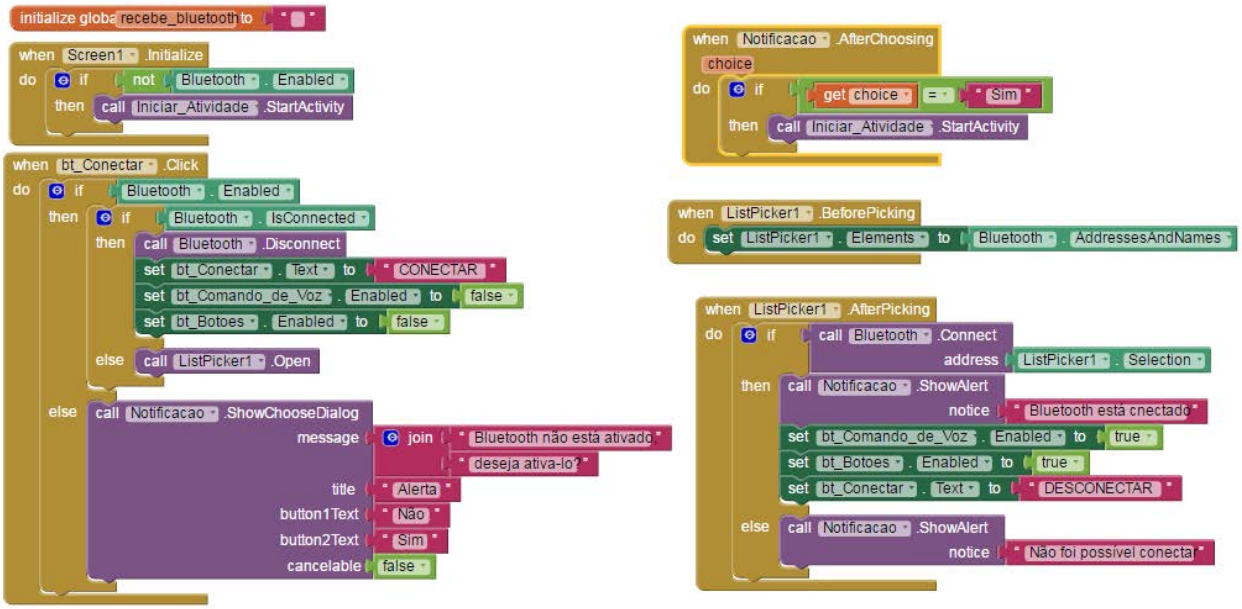
Figura 40 - Bloco do botão Instruções



Fonte: Produção do próprio autor

- Botão *Conectar*: Possui mais de um bloco de programação. Tais blocos são mostrados na Figura 41 e são responsáveis pela criação de uma conexão *Bluetooth* com o módulo e caso a opção *Bluetooth* não esteja habilitada no aparelho, irá aparecer uma tela de notificação alertando o usuário a habilitar esta opção.

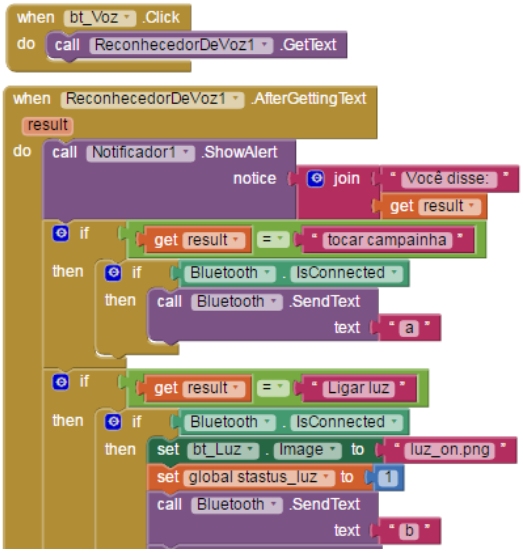
Figura 41 - Blocos do botão Conectar



Fonte: Produção do próprio autor

- Botão *Comando de voz*: Ao pressionar este botão, irá abrir a opção para captura de som que transformará o áudio captado em uma ou mais palavras e o bloco da Figura 42 irá compará-las com as palavras programadas, se forem iguais, enviará uma *string* via *Bluetooth* para a placa.

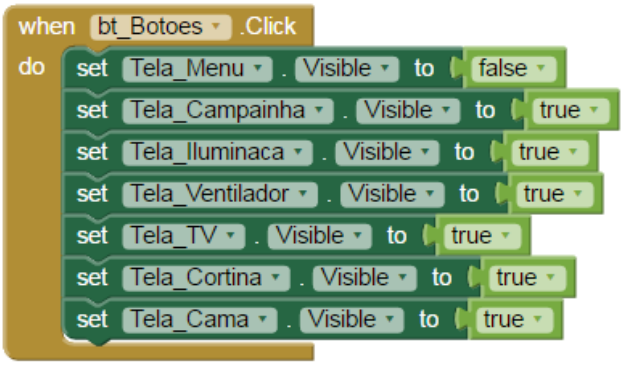
Figura 42 - Blocos do botão Comando de voz



Fonte: Produção do próprio autor

- Botão *Botões:* Ao invés de abrir uma nova janela no aplicativo, o bloco da Figura 43 desabilita a tela principal e habilita a tela secundária que possui os botões específicos para cada dispositivo elétrico. Tal ação é necessária para que não se perca a conexão Bluetooth, pois só é possível fazer a conexão com uma janela de cada vez.

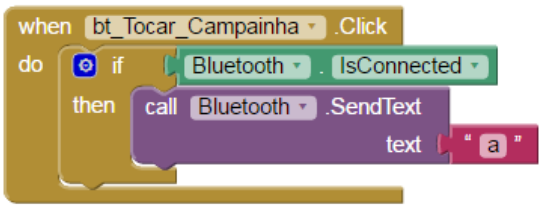
Figura 43 - Blocos do botão Botões



Fonte: Produção do próprio autor

- Botão *Tocar_Campainha*: O bloco da Figura 44 é responsável por enviar o caractere “a” via Bluetooth para mudar o nível lógico das portas A0 e A1 para alto, fazendo com que o buzzer toque e o LED ascenda.

Figura 44 - Bloco do botão Tocar_Campainha

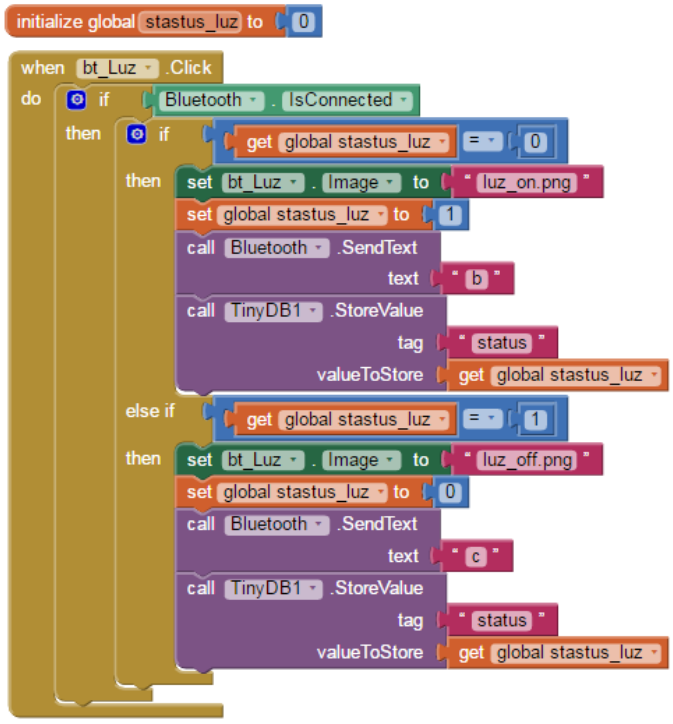


Fonte: Produção do próprio autor

- Botão *Luz*: Os blocos da Figura 45 são responsáveis pelo envio dos caracteres “b” e “c” para acionar a porta digital 12, fazendo com que a lâmpada ascenda ou apague. A variável “status_luz” muda o valor de acordo com o estado da lâmpada (acesa ou

apagada) e é armazenada no banco de dados para que quando o aplicativo for fechado e aberto novamente, salve a última ação ocorrida no botão.

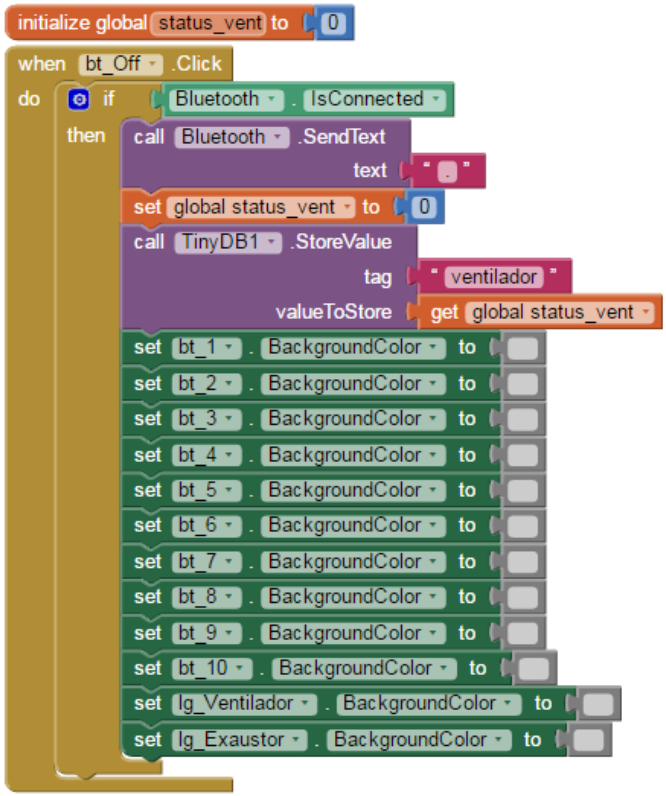
Figura 45 - Blocos do botão Luz



Fonte: Produção do próprio autor

- Conjunto de botões do ventilador: Cada botão do ventilador envia um caractere específico via Bluetooth para ligar/desligar o motor no sentido horário e anti-horário e para o controle de velocidade. Assim como a variável “status_luz”, a “status_vent” é usada para armazenar a última ação do ventilador no banco de dados. Também há a mudança na cor do botão quando o mesmo for pressionado. A Figura 46 apresenta o bloco da variável “status_vent” e do botão Off, responsável pelo desligamento do motor.

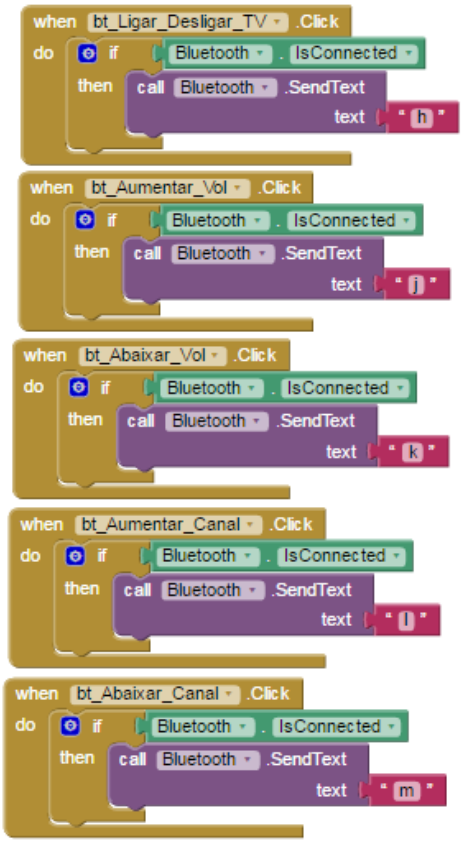
Figura 46 – Bloco da variável status_vent e do botão Off



Fonte: Produção do próprio autor

- Botões da televisão: conjunto de cinco botões que ao serem pressionados separadamente, enviam os caracteres “h”, “j”, “k”, “l” ou “m” para ligar/desligar a TV, aumentar/diminuir o volume e subir/descer o canal. A Figura 47 apresenta os blocos responsáveis pelo envio desses caracteres.

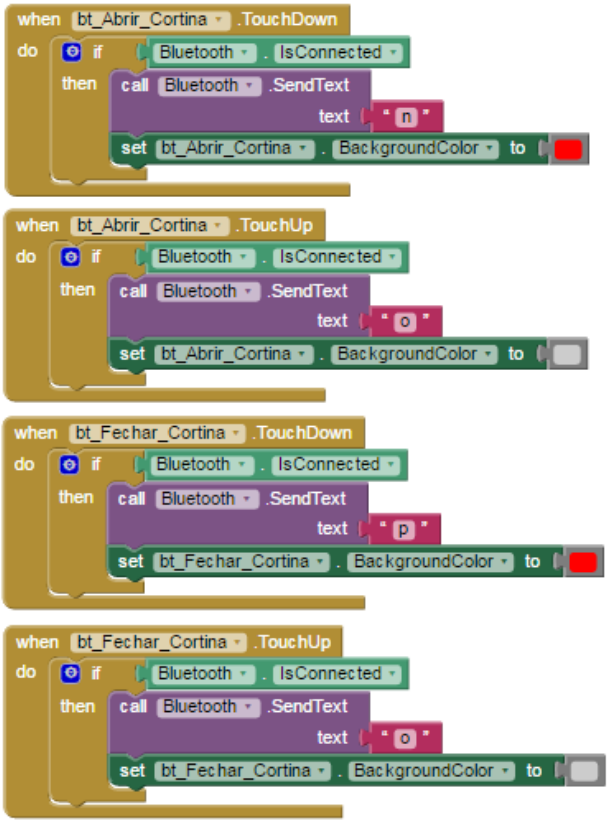
Figura 47 - Blocos dos botões da TV



Fonte: Produção do próprio autor

- Botões da cortina: Na Figura 48 são apresentados dois botões usados para abrir e fechar a cortina, que envia os caracteres “n”, “p” ou “o”. Quando o botão “abrir” é pressionado, envia o caractere “p” para a placa que o interpreta e faz com que o motor gire, mas quando o botão “abrir” é solto, envia o caractere “o” fazendo com que o motor pare de girar. Também há a mudança de cor do botão quando o mesmo é acionado.

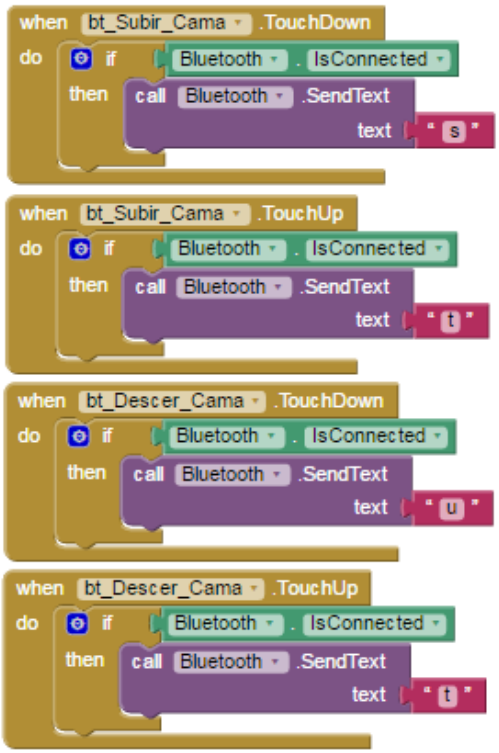
Figura 48 - Blocos dos botões da cortina



Fonte: Produção do próprio autor

- Botões da cama: Duas setas representam os botões para subir e descer a cama. Assim como a cortina, quando o botão é pressionado o motor gira no sentido horário ou anti-horário e quando o botão é solto, o motor para. Os blocos da Figura 49 são responsáveis por enviar os caracteres “s”, “t” ou “u” via *Bluetooth* para a placa.

Figura 49 - Blocos dos botões da cama



Fonte: Produção do próprio autor

Além dos procedimentos para a criação da interface e dos blocos de programação, também houve a parte “estética” do aplicativo, onde foi adicionado figuras que substituem os botões, além de imagens na tela para uma melhor apresentação.

4 VIABILIDADE DE IMPLEMENTAÇÃO

Na Tabela 1 é mostrado os componentes necessários para a implementação deste projeto em um quarto de hospital e seus respectivos custos.

Tabela 1 - Componentes e seus respectivos custos

Componente	Quantidade	R$	Total R$
Placa Arduino UNO	1	40,00	40,00
Fonte 12V	1	10,00	10,00
Módulo Bluetooth	1	24,00	24,00
Módulo relé	6	7,00	42,00
Sirene 127V AC	1	60,00	60,00
Dimmer shield	2	60,00	120,00
Receptor IR	1	6,00	6,00
LED IR	1	5,00	5,00

Fonte: Produção do próprio autor

O uso de seis módulos relés é para o acionamento da sirene, da lâmpada e dois para cada motor (cama e cortina). Já o uso de dois Dimmer Shields é para o controle de velocidade do ventilador nos sentidos horário e anti-horário. Também pode-se incluir no custo total o valor de uma caixa, similar a caixa de disjuntores, para armazenar a placa e os módulos, porém isto se torna opcional.

O valor total aproximado para a implementação do projeto é de aproximadamente R$ 307,00 para cada quarto, o que pode ser considerado viável já que seria um diferencial do hospital e proporcionaria ao cliente um enorme conforto. Porém, quanto maior for a quantidade de dispositivos para serem controlados, maior será o custo.

É importante ressaltar que não foi considerado no custo total o valor do *smartphone*.

5 CONCLUSÃO

A utilização da tecnologia *Bluetooth* para o controle de dispositivos elétricos em um quarto de hospital mostrou-se eficaz, pois o alcance é satisfatório visto que não é necessário que o sinal atinja um longo comprimento e possui um baixo custo em relação a outros meios de comunicação, como por exemplo, um módulo wifi que pode custar até o triplo do valor do módulo *Bluetooth*. Também deve-se ressaltar que o uso do *Bluetooth* para este projeto é viável em relação à segurança, pois cada módulo pode ser configurado uma senha específica, o que evita a conexão de aparelhos intrusos.

Outro ponto importante neste projeto é em relação à economia de energia, pois ao acessar equipamentos de forma remota, haverá por consequência, uma redução no consumo de energia devido ao acionamento dos dispositivos serem feitos de maneira inteligente, somente quando for necessário.

Porém, a principal motivação para a realização deste trabalho é fazer com que o auxílio de pequenas tarefas para o paciente em um quarto de hospital seja minimizado, pois em um ambiente automatizado ele não necessitaria de ajuda para realizar tarefas simples, tonando-se menos dependente no período em que se encontra hospedado.

REFERÊNCIAS

APP INVENTOR. Disponível em < http://ai2.appinventor.mit.edu/> Acesso em 2 ago. 2016.

ARDUINO. Disponível em < https://www.arduino.cc > Acesso em 20 set. 2016.

ARDUINO APRENDIZES. **Arduino**. Disponível em <https://arduinoaprendizes.wordpress.com/2015/04/22/historiaarduino/ > Acesso em 20 set. 2016.

BEGHINI, L. B. **Automação residencial de baixo custo por meio de dispositivos móveis com sistema operacional Android**. 2013. 76 f. Trabalho de Graduação (Graduação em Engenharia Elétrica com ênfase em Eletrônica) – Universidade de São Paulo, São Paulo, 2013.

BUILDBOT. **Configuração do módulo Bluetooth**. Disponível em <http://buildbot.com.br/blog/configuracao-do-modulo-bluetooth-hc-06-com-arduino/> Acesso em 10 out. 2016.

CIRCUITS. Disponível em < https://circuits.io/> Acesso em 10 ago. 2016.

EMBARCADOS. **Arduino UNO**. Disponível em <https://www.embarcados.com.br/arduino-uno/ > Acesso em 20 set. 2016.

FERREIRA, V. Z. G. **A domótica como instrumento para a melhoria da qualidade de vida dos portadores de deficiência**. 2010. 30 f. Trabalho de Graduação (Tecnologia em Automação Industrial) - Instituto Federal de Educação, Ciência e Tecnológica da Paraíba, João Pessoa, 2010.

GDS AUTOMAÇÃO. **Automação residencial**. 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 20 ago. 2016.

IFOWESTER. **Bluetooth**. Disponível em < http://www.infowester.com/bluetooth.php > Acesso em 12 set. 2016.

MCROBERTS, Michael. **Arduino Básico**. São Paulo: Novatec, 2011. 453 p.

MUNDO ESTRANHO. **Reconhecimento de Voz em aparelhos eletrônicos**. Disponível em < http://mundoestranho.abril.com.br/alimentacao/como-funciona-o-reconhecimento-de-voz-em-aparelhos-eletronicos/ > Acesso em 18 set. 2016.

MURATORI, J. R.; BO, P.H.D. Automação residencial: histórico, definições e conceitos. **O Setor Elétrico**, v.62, n. 2, p. 70-77, 2011.

NEWTONCBRAGA. **Robótica**. Disponível em <http://www.newtoncbraga.com.br/index.php/robotica/5166-mec068a> Acesso em 5 out. 2016.

OLIVEIRA, D. V. G.; PETREK, F. J. **Sistema de automação residencial controlado via web**. 2014. 165 f. Trabalho de Graduação (Graduação em Engenharia Industrial Elétrica com ênfase em Eletrônica e Telecomunicações) – Universidade Tecnológica Federal do Paraná, Curitiba, 2014.

SIGNIFICADOS. **Automação**. Disponível em <https://www.significados.com.br/automacao/ > Acesso em 20 ago. 2016.

SILVA, P. H. O. **Sistema de segurança de tranca de porta e controle de acesso**. 2013. 78 f. Trabalho de Graduação (Graduação em Engenharia de Computação) - Centro Universitário de Brasília, Brasília, 2013.

SOUZA, D. A. R. **Monitoramento e controle sem fio de sistemas de refrigeração.** 2012. 102 f. Trabalho de Monografia (Monografia de Especialização) - Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Florianópolis, 2012.

TECHTUDO. **App Inventor**. Disponível em < http://www.techtudo.com.br/tudo-sobre/google-app-inventor.html > Acesso em 2 out. 2016.

TECHTUDO. **Comando de Voz**. Disponível em <http://www.techtudo.com.br/noticias/noticia/2016/04/ok-google-now-conheca-todos-os-comandos-de-voz-da-assistente-virtual.html> Acesso em 18 set. 2016.

TECMUNDO. **Reconhecimento de Voz**. Disponível em <https://www.tecmundo.com.br/curiosidade/3144-como-funciona-o-reconhecimento-de-voz-.htm > Acesso em 18 set. 2016.

TELECO. **Tutorial Bluetooth**. Disponível em <http://www.teleco.com.br/tutoriais/tutorialblue/pagina_3.asp> Acesso em 12 set. 2016.

APÊNDICE A - Código do Arduino para acionamento de um módulo relé

```
int lampada = 12;    // pino IN do relé conectado à entrada 12


void setup()
{
Serial.begin(9600); //inicializa a comunicação serial na velocidade 9600
pinMode(lampada, OUTPUT); //lâmpada é uma saída
}

void loop()
{
char comando = Serial.read(); //variável para ler os dados da serial
if(comando == 'b') // se o botão no celular for pressionado para ligar a lâmpada
{
digitalWrite(lampada,HIGH); //acende a lâmpada
}
if(comando == 'c') // se o botão no celular for pressionado para desligar a lâmpada
{
digitalWrite(lampada,LOW); // desliga a lâmpada
}
}
```

APÊNDICE B – Código do Arduino para o circuito da campainha

```
int buzzer = A0;  // buzzer conectado à entrada A0

int led = A1; // LED conectado à entrada A1
int botao = A2;  // botão conectado à entrada A2
int estadoBotao = 0;  // variável que contém o estado inicial do botão

void setup()
{
Serial.begin(9600); // inicializa a comunicação serial na velocidade 9600
pinMode(botao, INPUT);  // botão é uma entrada
pinMode(buzzer, OUTPUT); // buzzer é uma saída
pinMode(led, OUTPUT);    // LED é uma saída
}

void loop()
{
char comando = Serial.read(); // variável para ler os dados da serial
if(comando == 'a')// se o botão no celular for pressionado para tocar a campainha
{
digitalWrite(led,HIGH); // liga o LED
tone(buzzer, 440); // aciona o buzzer na ferquência de 440Hz
delay(1000);       // o som do buzzer é emitido na frequência de 440Hz por 1s
tone(buzzer, 392); //aciona o buzzer na ferquência de 392Hz
delay(1000);       //o som do buzzer é emitido na frequência de 392Hz por 1s
}
else
{
noTone(buzzer); // se o botão não for apertado o buzzer não toca
}
//comando para desligar o led quando o botão for pressionado
estadoBotao= digitalRead(botao);// a variável estado do botão lê o botão
if(estadoBotao == HIGH) // se o botão for pressionado
{
digitalWrite(led,LOW); // desliga o led
}
}
```

APÊNDICE C – Código do Arduino para o circuito do ventilador

```
int ventilador = 9; // controla um lado do motor

int comandoVent = 11; // CI na entrada PWM para controlar o sentido da rotação
int exaustor = 10; // controla outro lado do motor

void setup()
{
Serial.begin(9600); //inicializa a comunicação serial na velocidade 9600
pinMode(comandoVent, OUTPUT); // CI L293D é uma saída
pinMode(ventilador, OUTPUT); // ventilador é uma saída
pinMode(exaustor, OUTPUT); // exaustor é uma saída
}

// Controle do Motor do ventilador
void motor(int mode, int percent)
{
// muda a percentagem de 0 até 100 no PWM
int duty = map(percent, 0, 100, 0, 255);
switch(mode)
{
case 0: //desativa o motor
digitalWrite(comandoVent, LOW);
break;
case 1: //define o sentido horário
digitalWrite(ventilador, HIGH); //define ventilador HIGH
digitalWrite(exaustor, LOW); //define exaustor LOW
analogWrite(comandoVent, duty); //usa o PWM para controlar a velocidade do motor
break;
case 2: //define o sentido anti-horário
digitalWrite(ventilador, LOW); //define ventilador  LOW
digitalWrite(exaustor, HIGH); //define exaustor  HIGH
analogWrite(comandoVent, duty); //usa o PWM para controlar a velocidade do motor
break;
case 3: //Travar o motor
digitalWrite(ventilador, LOW); //define ventilador  LOW
digitalWrite(exaustor, LOW); //define exaustor  LOW
analogWrite(comandoVent, duty); //usa o PWM para travar o motor
break;
}
}

void loop()
{
char comando = Serial.read(); //variável para ler os dados da serial
if(comando == '1') // liga o motor do ventilador com velocidade 50%
{
motor(1, 50);
}
```

```
if(comando == '2') // liga o motor do ventilador com velocidade 40%
{
motor(1,40);
}
if(comando == '3') // liga o motor do ventilador com velocidade 30%
{
motor(1,30);
}
if(comando == '4') // liga o motor do ventilador com velocidade 20%
{
motor(1,20);
}
if(comando == '5') // liga o motor do ventilador com velocidade 10%
{
motor(1,10);
}
if(comando == '.') // desliga o motor
{
motor(3,100);
}
if(comando == '6') // liga o exaustor com velocidade 10%
{
motor(2, 10);
}
if(comando == '7') // liga o exaustor com velocidade 20%
{
motor(2,20);
}
if(comando == '8') // liga o exaustor com velocidade 30%
{
motor(2,30);
}
if(comando == '9') // liga o exaustor com velocidade 40%
{
motor(2,40);
}
if(comando == '0') // liga o exaustor com velocidade 50%
{
motor(2,50);
}
}
```

APÊNDICE D – Código do Arduino para o circuito da cortina

```
int comandoCortina = 6; // CI na entrada PWM  para controlar o sentido da rotação

int motor1Cortina = 7; // controla um lado do motor da cortina
int motor2Cortina = 8; // controla outro lado do motor da cortina

void setup()
{
Serial.begin(9600); //inicializa a comunicação serial na velocidade 9600
pinMode(comandoCortina, OUTPUT); // CI L293D é uma saída
pinMode(motor1Cortina, OUTPUT);  // motor1Cortina é uma saída
pinMode(motor2Cortina, OUTPUT);  // motor2Cortina é uma saída
}

// Controle do Motor da Cortina
void motorCortina(int mode, int percent)
{
//muda a percentagem de 0 até 100 no PWM
int duty = map(percent, 0, 100, 0, 255);
switch(mode)
{
case 0: //desativa o motor
digitalWrite(comandoCortina, LOW);
break;
case 1: //define o sentido horário
digitalWrite(motor1Cortina, HIGH); //define motor1Cortina HIGH
digitalWrite(motor2Cortina, LOW); //define motor2Cortina LOW
analogWrite(comandoCortina, duty); //usa o PWM para controlar a velocidade
break;
case 2: //define o sentido anti-horário
digitalWrite(motor1Cortina, LOW); //define motor1Cortina LOW
digitalWrite(motor2Cortina, HIGH); //define motor2Cortina HIGH
analogWrite(comandoCortina, duty); //usa o PWM para controlar a velocidade
break;
case 3: //Travar o motor
digitalWrite(motor1Cortina, LOW); //define motor1Cortina LOW
digitalWrite(motor2CortinaA, LOW); //define motor2Cortina LOW
analogWrite(comandoCortina, duty); //usa o PWM para travar o motor
break;
}
}

void loop()
{
char comando = Serial.read(); //variável para ler os dados da serial
if(comando == 'n') // quando o botão no celular é pressionado para fechar a cortina
 {
 motorCortina(1, 70);  // motor no sentido horário com 70% de velocidade
 }
```

```
if(comando == 'o') // quando o botão no celular é solto o motor para
{
motorCortina(3,100);  // o motor para
}
if(comando == 'p')  // quando o botão no celular é pressionado para abrir a cortina
{
motorCortina(2, 70); // motor no sentido anti-horário com 70% de velocidade
}
if(comando == 'q') // quando o comando de voz é acionado para fechar a cortina
{
motorCortina(1, 70);  // motor no sentido horário com 70% de velocidade
delay(2500);     //  motor gira por 2,5 segundos
motorCortina(3,100);  // o motor para
}
if(comando == 'r')  // quando o comando de voz é acionado para abrir a cortina
{
motorCortinaA(2, 70); // motor no sentido anti-horário com 70% de velocidade
delay(2500);     // motor gira por 2,5 segundos
motorCortinaA(3,100);  // o motor para
}
}
```

APÊNDICE E– Código do Arduino para o circuito da cama

```
int comandoCama = 5; // CI na entrada PWM  para controlar o sentido da rotação

int motor1Cama = 2; // controla um lado do motor da cama
int motor2Cama = 4; // controla outro lado do motor da cama

void setup()
{
Serial.begin(9600); //inicializa a comunicação serial na velocidade 9600
pinMode(comandoCama, OUTPUT); // CI l293d é uma saída
pinMode(motor1Cama, OUTPUT);  // motor1Cama é uma saída
pinMode(motor2Cama, OUTPUT);  // motor2Cama é uma saída
}

// Controle do Motor da Cama
void motorCama(int mode, int percent)
{
//muda a percentagem de 0 até 100 no PWM
int duty = map(percent, 0, 100, 0, 255);

switch(mode)
{
case 0: //desativa o motor
digitalWrite(comandoCama, LOW);
break;
case 1: //define o sentido horário
digitalWrite(motor1Cama, HIGH); //define motor1Cama HIGH
digitalWrite(motor2Cama, LOW); //define motor2Cama LOW
analogWrite(comandoCama, duty); // usa o PWM para controlar a velocidade do motor
break;
case 2: //define o sentido anti-horário
digitalWrite(motor1Cama, LOW); //define motor1Cama LOW
digitalWrite(motor2Cama, HIGH); //define motor2Cama HIGH
analogWrite(comandoCama, duty); // usa o PWM para controlar a velocidade do motor
break;
case 3: //Travar o motor
digitalWrite(motor1Cama, LOW); //define motor1Cama LOW
digitalWrite(motor2Cama, LOW); //define motor2Cama LOW
analogWrite(comandoCama, duty); //usa o PWM para travar o motor
break;
}
}

void loop()
{
char comando = Serial.read(); //variável para ler os dados da serial
if(comando == 's') // quando o botão no celular é pressionado para subir a cama
{
motorCama(1, 100);  // Motor no sentido horário com 100% de velocidade
```

```
}
if(comando == 't') // quando o botão no celular é solto o motor da cama para
{
motorCama(3,100);
}
if(comando == 'u')  // quando o botão no celular é pressionado para descer a cama
{
motorCama(2, 50); // Motor no sentido anti-horário com 100% de velocidade
}
if(comando == 'v') // quando o comando de voz é acionado para subir a cama
{
motorCama(1, 100);  // Motor no sentido horário com 100% de velocidade
delay(1000);    // motor gira por 1 segundo
motorCama(3,100);  // após 1 segundo o motor para
}
if(comando == 'x')  // quando o comando de voz é acionado para descer a cama
{
motorCama(2, 50); // Motor no sentido anti-horário com 100% de velocidade
delay(1000);  // motor gira por 1 segundo
motorCama(3,100);   // após 1 segundo o motor para
}
}
```

APÊNDICE F– Código do Arduino para o circuito da TV

```
#include <IRremote.h>

IRsend irsend; // LED conectado ao pino PWM 3

void setup()
{
Serial.begin(9600);
}

void loop()
{
char comando = Serial.read();
if(comando == 'h') // h = bt_ligar
{
// 0x(hexadecimal); 2FD48B7(codigo do botão); 32(numero de bits)
irsend.sendNEC(0x2FD48B7, 32); // TV SEMP TOSHIBA
delay(40);
}
if(comando == 'j') // j = bt_vol+
{
irsend.sendNEC(0x2FD58A7, 32);
delay(40);
}
if(comando == 'k') // k = bt_vol-
{
irsend.sendNEC(0x2FD7887, 32);
delay(40);
}
if(comando == 'l') // l = bt_ch+
{
irsend.sendNEC(0x2FDD827, 32);
delay(40);
}
if(comando == 'm') // m = bt_ch-
{
irsend.sendNEC(0x2FDF807, 32);
delay(40);
}
}
```