UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

Uma Proposta de Sistema de Controle Remoto Universal com Suporte a Reconhecimento e Síntese de Voz

Cassio Trindade Batista

BELÉM - PARÁ

2016

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

Cassio Trindade Batista

Uma Proposta de Sistema de Controle Remoto Universal com Suporte a Reconhecimento e Síntese de Voz

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Engenheiro em Engenharia da Computação, do Instituto de Tecnologia, da Faculdade de Engenharia da Computação e Telecomunicações.

BELÉM - PARÁ

2016

"Inteligência é a capacidade de se adaptar à mudança"

Stephen Hawking

Uma Proposta de Sistema de Controle Remoto Universal com Suporte a Reconhecimento e Síntese de Voz

Este trabalho foi julgado adequado em _____ para a obtenção do Grau de Engenheiro de Computação, aprovado em sua forma pela banca examinadora que atribuiu o conceito _____.

_____

Prof. Dr. Nelson Cruz Sampaio Neto

ORIENTADOR

_____

Prof. Dra. Valquiria Gusmão Macedo

MEMBRO DA BANCA EXAMINADORA

_____

Prof. Dr. Adalbery Rodrigues Castro

MEMBRO DA BANCA EXAMINADORA

_____

Prof. Dr. Francisco Carlos Bentes Frey Müller

DIRETOR DA FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

Dedico esse TCC à minha família e amigos.

Agradecimentos

Agradeço primeiramente a Deus, por toda sabedoria e principalmente pela força e perseverança a mim dada para chegar até aqui.

À minha família, que nunca deixou de me oferecer ajuda e apoio. Em especial, aos meus pais Raimundo Batista e Rosa Batista, pela dedicação, prioridade, amor incondicional e todo o amparo que me deram durante toda a minha vida; e à minha irmã, Carolina Batista, por sempre ter acreditado em mim, até quando eu mesmo não pude. Agradeço também aos meus avós e a todos os meus tios e primos, que sempre estiveram, direta ou indiretamente, ao meu lado.

Aos professores da Engcomp/FCT, sem os quais não teria a disciplina e o conhecimento necessários para o desenvolvimento deste trabalho. Em especial ao meu orientador, Professor Nelson Neto, por toda a paciência e ajuda durante a produção deste trabalho e dos outros que vieram a ser desenvolvidos ao longo de três anos; e ao meu antigo orientador Aldebaro Klautau, juntamente com o pessoal do Laboratório de Processamento de Sinais (LaPS) por todas as cobranças, incentivos e oportunidades.

Aos meus colegas de classe Gabriel Carvalho, Pedro Figueiredo e Thiago Coelho, pela parceria durante as aulas, pela ajuda nos muitos trabalhos ao longo dos quase cinco anos de curso e por todos os *coffee-breaks* nos intervalos do estágio. Agradeço em especial ao Thiago pela ajuda, apoio e paciência durante o intercâmbio.

A todos os amigos que fiz durante essa jornada, os quais não conseguiria listar e, por isso, não ouso citar nomes, mas que sem os quais o caminho até aqui não teria sido o mesmo.

Cassio Trindade Batista

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ARM *Advanced RISC Machines*

API *Application Programming Interface*

App Aplicativo

ASR *Automatic Speech Recognition*

BBB Beaglebone Black

DTO *Device Tree Overlay*

DTS *Device Tree Source*

G2P *Grapheme to Phoneme*

GCC GNU C *Compiler*

GPIO *General Purpose Input/Output*

HMM *Hidden Markov Model*

HTK *HMM Toolkit*

IBGE Instituto Brasileiro de Geografia Estatística

IR *Infrared*

LaPS Laboratório de Processamento de Sinais

LED *Light Emitting Diode*

MFCC *Mel-Frequency Cepstral Coefficients*

MMII Membros Inferiores

MMSS Membros Superiores

PCD Pessoas com Deficiência

PRU *Programmable Real-time Unit*

PRUSS *Programmable Real-time Unit Subsystem*

PT_BR Português Brasileiro

PWM *Pulse Width Modulation*

RISC *Reduced Instruction Set Computer*

SoC *System on Chip*

SUS *System Usability Scale*

TA Tecnologia Assistiva

TTS *Text to Speech*

UFPa Universidade Federal do Pará

VAD *Voice Activity Detection*

Resumo

A evolução tecnológica, num sentido amplo, caminha para que as pessoas possam interagir com os equipamentos eletrônicos de forma mais fácil. Por outro lado, para portadores de deficiência, essa interação torna-se algo mais do que simples: ela se torna possível. Através das Tecnologias Assistivas, os portadores de necessidades especiais alcançam maior independência, qualidade de vida e inclusão social por meio de interfaces alternativas de controle de dispositivos. O presente trabalho busca preparar um servidor local portátil de reconhecimento de voz para o Português Brasileiro baseado na plataforma Beaglebone Black, um microcomputador embarcado que, quando acessado pelo dispositivo cliente que agirá como controle remoto — no caso, um *smartphone* com sistema operacional Android, que também é capaz de fornecer *feedbacks* via voz artificial graças à adição de um módulo de síntese de fala —, será capaz de executar as funções mais básicas de um aparelho televisivo. Ao se alterar os métodos tradicionais de controle, substituindo o uso das mãos e de botões convencionais em um controle remoto por comandos de voz, o sistema ganha características de acessibilidade, já que portadores de necessidades especiais, principalmente visuais e motoras, podem ser significativamente beneficiados. Dentre as aplicações encontradas a respeito de controles remotos alternativos para aparelhos eletrônicos convencionais (que recebem informação remota via luz infravermelha), nenhuma possui suporte a reconhecimento e síntese de voz em Português Brasileiro sem a necessidade de conexão com a Internet; tampouco possui funcionalidades com características de acessibilidade que se encaixem no contexto da Tecnologia Assistiva. Um teste de qualidade e usabilidade foi aplicado a voluntários com o objetivo de avaliar o sistema, o qual mostrou grande interesse por parte dos usuários, apesar da dificuldade relatada por alguns na utilização de *smartphones*. É importante ressaltar que todos os recursos e ferramentas utilizados durante o desenvolvimento são livres e gratuitos.

Abstract

Technological developments converge to make people interact with electronic devices in an easy way. On the other hand, for people with disabilities, that interaction becomes something more than simple: it becomes possible. Through the Assistive Technology, people with special needs can achieve their independence, quality of life and social inclusion by means of alternative control interfaces. The current work seeks to construct a portable and local speech recognition server for Brazilian Portuguese language on a Beaglebone Black microcomputer. When the embedded platform is accessed by the client device that acts as a remote control — i.e., a smartphone with an Android operating system, which is also capable of giving feedbacks via artificial voices due the addition of a speech synthesis module —, it will be able to execute the most basic functions of a television set. Changing the traditional control methods by replacing the use of hands and also the use of conventional buttons in a remote control device for voice commands, the system shows accessibility characteristics, since people with special needs, specially with visual and motor impairments, can significantly be benefited. From the set of applications found with respect to alternative remote controls for conventional electronic devices (i.e., devices that receive the remote information via infrared light), none of them supports speech recognition and speech synthesis in Brazilian Portuguese without the need for Internet connection. None of them shows accessibility characteristics that fit into the Assistive Technology context too. A survey of quality and usability was applied to volunteers in order to evaluate the system. The results showed great interest of the users, despite the difficulty related for some of them during the use of smartphone devices. It is important to emphasize that all the resources and tools used in the project are open-source and free to use.

Capítulo 1

Introdução

1.1 Contextualização

A interface humano-computador encontra-se cada vez mais amigável. O que antes era portado somente por empresas e pessoas com poder financeiro diferenciado e conhecimento acima da média, em termos de tecnologia, é hoje muito mais acessível e simples para usuários domésticos sem profundo entendimento no assunto. Devido à abrangência dos computadores (pessoais e embarcados) e da Internet, novas oportunidades e expectativas em termos de trabalho, estudo e até lazer são criadas a fim de melhorar ainda mais essa comunicação, de modo que a máquina se aproxime mais de ações típicas do ser humano, como pensar e falar.

Acredita-se que as tecnologias de síntese e reconhecimento automático de voz (TTS e ASR, do inglês *text-to-speech* e *automatic speech recognition*, respectivamente) [1, 2] tornam a interface humano-computador muito mais prática e natural, de forma que a comunicação de fato se assemelha àquela estabelecida entre duas pessoas. O ASR refere-se ao sistema que, tomando o sinal de fala digitalizado como entrada, é capaz de gerar o texto transcrito na saída. Já um sistema TTS realiza a função contrária, na qual um sinal analógico de voz é sintetizado de acordo com o texto posto na entrada. Dentre os inúmeros sistemas que envolvem processamento de fala, destacam-se os de automação residencial, os quais promovem comodidade e praticidade às pessoas no controle de equipamentos eletrônicos; e os que visam ajudar pessoas que tenham dificuldades no controle dos desses equipamentos. Normalmente enquadrados no conceito de tecnologia assistiva, tais sistemas tornam-se soluções acessíveis às pessoas com deficiência.

A Tecnologia Assistiva (TA) é um campo da engenharia biomédica dedicado a aumentar a independência e mobilidade de pessoas com deficiência (PCD), englobando metodologias,

práticas e serviços que objetivam promover sua autonomia, qualidade de vida e inclusão social [3, 4]. O termo refere-se à tecnologia que fornece assistência às PCD a fim de reduzir os efeitos das deficiências e permitindo-lhes participar ativamente das suas respectivas rotinas. Segundo a Convenção sobre os Direitos das Pessoas com Deficiência [5], publicada pela Organização das Nações Unidas (ONU) em 2006, os olhares do século XXI deixam de enxergar as PCD como "objetos" de caridade, tratamento médico e proteção social e passam a vê-las sob uma nova perspectiva, na qual as PCD são "sujeitos" ativos na sociedade com direitos, capazes de reivindicar tais direitos e tomar decisões baseadas em seu próprio consentimento.

A TA busca reduzir a dificuldade vivenciada por pessoas que precisam de soluções que não as deixem à margem da utilização de dispositivos eletrônicos. Visando diminuir a exclusão digital imposta às PCD pela incapacidade de manipular certos equipamentos, a acessibilidade é vista como elemento fundamental para elevar a autoestima e o grau de independência dessas pessoas. Além disso, as soluções apresentadas também podem ser úteis para os não portadores de necessidades especiais, já que o controle de equipamentos se torna mais prático e confortável.

Nesse sentido, este trabalho busca preparar um servidor local portátil de reconhecimento de voz em Português Brasileiro (PT_BR) baseado na plataforma Beaglebone Black (BBB), um microcomputador embarcado que, quando acessado pelo dispositivo cliente que agirá como controle remoto — no caso, um *smartphone* com sistema operacional Android, que também é capaz de fornecer *feedbacks* via voz artificial graças à adição de um módulo de síntese de fala —, será capaz de executar as funções mais básicas de um aparelho televisivo. Vale ressaltar que todas as interfaces de programação (APIs, do inglês *application programming interfaces*), ambientes de desenvolvimento (IDEs, do inglês *integrated development environments*), *softwares*, bibliotecas e pacotes utilizados para criação dos sistemas e dos recursos estão disponíveis livremente na Internet [6–10].

O Ato de Americanos com Deficiência (ADA, do inglês *Americans with Disabilities Act*) [11] é um documento publicado em 1990 que regula os direitos dos cidadãos com deficiência nos Estados Unidos, além de prover a base legal dos fundos públicos para compra dos recursos que tais pessoas necessitam. Já o Comitê de Ajudas Técnicas (CAT) [3], criado em 2006, executa papel semelhante, tendo a perspectiva de aperfeiçoar e dar transparência e legitimidade ao desenvolvimento da Tecnologia Assistiva no Brasil.

O presente trabalho, por apresentar um sistema de controle remoto com suporte a reconhecimento e síntese de voz, enquadra-se em duas categorias da TA [12] criadas a partir da ADA

e atualmente existentes nas diretrizes do CAT, as quais serão brevemente descritas a seguir: i) recursos de acessibilidade ao computador; ii) e sistemas de controle de ambiente.

i) Recursos de acessibilidade ao computador: Equipamentos de entrada e saída (síntese de voz, Braille), auxílios alternativos de acesso (ponteiras de cabeça, de luz), teclados modificados, *softwares* especiais (reconhecimento de voz, etc.), que permitem as pessoas com deficiência a usarem o computador.

ii) Sistemas de controle de ambiente: Sistemas eletrônicos que permitem as pessoas com limitações moto-locomotoras controlar remotamente aparelhos eletro-eletrônicos, sistemas de segurança, etc, localizados em seu quarto, sala, escritório, casa e arredores.

1.2 Justificativa

Deficiência, por definição, é a capacidade parcial ou total incapacidade de determinada pessoa realizar tarefas comuns do cotidiano, quando comparada a outras de um grupo padrão de pessoas. A deficiência pode ser a consequência de um comprometimento físico, mental, sensorial, emocional, no desenvolvimento ou uma combinação de alguns desses fatores. O termo "deficiência", por ser bem extenso, abrange impedimentos, limitações na realização de atividades e restrições nas participações em sociedade [13].

No censo realizado em 2010 pelo Instituto Brasileiro de Geografia e Estatística (IBGE), aproximadamente 23,9% dos brasileiros declararam ter alguma deficiência [14]. Esse número é realmente expressivo, pois revela que, em uma população de 190,7 milhões habitantes, cerca de um quarto (46 milhões) é portador de necessidades especiais. Os dados também mostram que, desse total, cerca de 35,7 milhões apresentam dificuldades visuais, e 13,2 milhões apresentam dificuldades motoras. A Tabela 1.1 mostra o perfil da população brasileira com deficiência.

Tabela 1.1: Perfil da população brasileira com deficiência (IBGE, 2010).

Deficiência	Descrição	Número de Pessoas	Porcentagem
Visual	Cegueira ou dificuldades gerais	35.774.392	18,754 %
Motora	Paralisia ou dificuldades gerais	13.265.599	6,95 %
Auditiva	Surdez ou dificuldades gerais	9.717.318	5,094 %
Cognitiva	Problemas mentais ou intelectuais	2.611.536	1,369 %

Em [15], uma entrevista foi realizada com algumas PCD a fim de se construir um "controle de TV ideal" baseado nas características que esses cidadãos gostariam de encontrar em controles remotos convencionais, de modo que suas dificuldades em utilizá-los fossem minimizadas. A sugestão de um *design* mais limpo foi unânime entre os PCD, enquanto os deficientes visuais, em particular, apontaram a necessidade de algum tipo de *feedback* quando os botões fossem pressionados (como um estalo, por exemplo) e a implementação de alguma referência reconhecível pelo tato nos botões. Já os deficientes motores sugeriram um dispositivo menor, que não escorregasse facilmente das mãos, contendo um ângulo de controle mais abrangente (ou seja, que não precisasse ser diretamente apontado para a TV) e com botões maiores para aqueles que não conseguem ter controle absoluto sobre as mãos e/ou dedos.

O reconhecimento e a síntese de fala, assim como a escrita e a expressão gestual, são exemplos de métodos alternativos de interação bastante utilizados em sistemas que utilizam interfaces multimodais. Esses sistemas permitem que o acesso à máquina ocorra de várias maneiras através de dispositivos de entrada e saída de dados que tornem a interação mais natural, não sendo, portanto, limitado apenas aos convencionais botões e teclas. Embora haja diminuição na usabilidade e na rapidez da execução da tarefa quando associadas ao controle de equipamentos, tais técnicas de interação provêem uma experiência muito mais prazerosa e prática quando comparadas aos métodos que forçam o uso das mãos por parte do usuário [16]. Além disso, os métodos alternativos geralmente são a única maneira que as pessoas com deficiência visual e motora dos membros superiores encontram para interagir com determinados aparelhos de maneira independente.

A ideia de associar sistemas ASR e TTS a equipamentos eletrônicos já faz parte de pesquisas significativas na literatura. Em [17], por exemplo, um sistema inteligente de segurança foi implementado em uma Beaglebone Black (BBB), utilizando a biblioteca OpenCV para contar quantos indivíduos encontravam-se próximo à entrada do ambiente a ser vigiado. Um módulo de reconhecimento de fala, baseado no decodificador PocketSphinx, foi utilizado para receber comandos, enquanto o *software* eSpeak foi encarregado de prover as respostas dadas ao usuário via voz sintética. Em [18], o PocketSphinx, novamente, foi embarcado em um *smartphone* Android, permitindo a pessoas com lesões na medula espinhal controlar aparelhos domésticos através da interface de voz. O resultado do reconhecedor de fala era enviado para a placa IOIO-OTG, a qual, estando fisicamente conectada ao controle remoto de uma TV, acionava um determinado comando. Já em [19], um *kit* eletrônico de leitura e escrita foi desenvolvido para

atender a estudantes deficientes visuais da Índia. O módulo TTS, baseado no *software* HTS, foi preparado em conjunto com um sistema de reconhecimento visual de caracteres a fim de fornecer leitura automática em voz alta, enquanto as ferramentas do pacote HTK eram responsáveis por realizar o ASR.

O recente *boom* no desenvolvimento de aplicativos para plataformas móveis (apps) facilitou a concretização da ideia de tornar o *smartphone* um controle remoto universal. Atualmente, o mercado da tecnologia também já disponibiliza diversas soluções cujo objetivo propõe o controle mais prático de equipamentos eletrônicos, evitando a incômoda tarefa de manter uma pilha de controles remotos, um para cada aparelho. Como a maioria dos aparelhos utiliza controles remotos que emitem comandos via luz infravermelha (IR, do inglês *Infrared*), alguns apps necessitam de um dispositivo adicional para converter o sinal elétrico para IR, já que os novos *smartphones* não possuem um emissor de luz infravermelha embutido [20]. O dispositivo externo pode ser um *dongle*, geralmente acoplado ao *smartphone* pelo conector *jack* de áudio; ou um *gateway* (também chamado de *hub*) que recebe os sinais de controle via Wi-Fi ou Bluetooth. A seguir, é feita uma breve descrição sobre alguns aplicativos com função de controle remoto universal para *smartphones*.

Peel Smart Remote é um aplicativo que combina a ideia do controle remoto universal com a tecnologia de *streaming* de mídia e utiliza o emissor IR embutido no *smartphone* para enviar comandos aos dispositivos [21]. O X10 Commander permite que o usuário controle uma grande variedade de dispositivos dentro do ambiente da *smart home*. O app necessita que o protocolo X10, específico para automação residencial, seja executado em um servidor (Windows, Linux ou MacOSX) conectado na mesma rede que os aparelhos a serem controlados [22]. Control4 é um sistema que funciona de forma semelhante ao X10, necessitando que o Sistema Control4 seja adquirido e configurado para permitir o controle de diversos dispositivos domésticos [23]. Já o iRule é um aplicativo baseado em nuvem que permite que a interface seja customizada pelo usuário. No caso de aparelhos que não são capazes de conectar-se à rede (como os que possuem controles baseados em luz IR, por exemplo), um *gateway* é necessário para que os comandos sejam enviados ao aparelho [24]. O Harmony Smart Control é outro sistema cujo aplicativo necessita de um *gateway* para funcionar, o qual recebe comandos do *smartphone* via rede e emula o controle dos aparelhos via Wi-Fi, Bluetooth ou IR [25]. Já o Irdroid, que permite o controle apenas de TVs e DVDs, faz uso de *dongles* que são conectados via *jack*, USB, Wi-Fi ou Bluetooth [26].

Soluções como o X10 Commander e o Control4, por exemplo, são voltadas para o conceito de *smart homes* da computação pervasiva, o que normalmente exige que todos os aparelhos controlados estejam, de alguma forma, conectados à rede do ambiente residencial para serem reconhecidos pelo protocolo do sistema. Outras, ainda, são aplicadas apenas à *smart* TVs, nas quais a informação de controle é transmitida por Wi-Fi ao aparelho. Nenhuma das aplicações encontradas a respeito de controles remotos alternativos para aparelhos eletrônicos convencionais (que recebem informação remota via IR) possui suporte a reconhecimento e síntese de voz em PT_BR sem a necessidade de conexão com a Internet; tampouco possuem funcionalidades com características de acessibilidade que se encaixem no contexto da Tecnologia Assistiva.

Esta pesquisa tem como finalidade apresentar uma solução para diminuir a exclusão digital vivenciada especialmente por pessoas com variados graus de necessidade motora ou visual, as quais estão à margem do mundo eletrônico por conta da ausência de recursos que adaptem os dispositivos às suas necessidades. A tecnologia de reconhecimento de fala tornaria acessível a utilização de qualquer dispositivo eletrônico por usuários incapazes de realizar movimentos específicos com membros superiores, como segurar um controle físico e apertar botões ou digitar, por exemplo. Os portadores de necessidades visuais também poderão ser ajudados, em especial, pelo sistema TTS, já que, como a maioria dos controles não possui referências hápticas, a síntese de fala tornar-se-ia muito importante. Um *feedback* audível certamente seria muito mais natural, prático e cômodo.

1.3 Objetivos

O objetivo principal consiste em criar um protótipo portável, baseado em um microcomputador embarcado, que seja capaz de ajudar PCD com diferentes graus a controlar um aparelho eletrônico através do envio remoto de sinais. A princípio, como prova de conceito, uma única TV, da marca Samsung, será controlada. No entanto, o sistema projetado pode ser considerado de propósito geral e, futuramente, dar suporte a uma maior variedade de dispositivos.

O sistema será configurado como um servidor que disponibiliza um serviço de reconhecimento de fala, de modo que a televisão possa ser remotamente controlada pelo próprio *smartphone* do usuário através da sua voz. O aplicativo, desenvolvido para a plataforma Android, age como cliente na comunicação e é capaz de detectar o silêncio da fala (evitando o uso de um eventual botão de parada), bem como dar *feedbacks* sonoros das ações realizadas

pelo sistema graças ao uso de uma API de síntese de voz.

Testes de usabilidade serão aplicados a voluntários a fim de verificar a qualidade do sistema e identificar possíveis desvantagens e futuras melhorias.

1.3.1 Objetivos Específicos

Os objetivos específicos podem ser divididos em cinco subgrupos principais: i) estudo e implementação das tecnologias de voz; ii) estudo da plataforma embarcada e configuração do servidor; iii) estudo e implementação da comunicação via luz infravermelha; iv) estudo e programação da plataforma cliente; v) testes de usabilidade com voluntários.

Uma breve descrição sobre cada tópico é feita a seguir.

i) Estudo de sistemas que envolvem processamento de voz:
   - Reconhecimento automático de fala;
   - Síntese de fala;
   - Detecção de fala/silêncio.

ii) Estudo e configuração de um servidor local em um computador embarcado:
   - Configuração do sistema de reconhecimento de fala em nuvem;
   - Criação de um banco de dados dos comandos de controle do aparelho eletrônico.

iii) Estudo da comunicação entre controles remotos e aparelhos eletrônicos domésticos via luz infravermelha:
   - Estudo da capacidade de processamento em tempo real dos componentes de *hardware* do computador embarcado;
   - Implementação do protocolo da Samsung no computador embarcado.

iv) Estudo e configuração de um cliente para *smartphones* Android:
   - Configuração do sistema de síntese de voz;
   - Implementação do sistema de detecção de silêncio;
   - Implementação do sistema de transmissão de áudio para o servidor.

v) Testes de usabilidade com voluntários:
   - Aplicação de um questionário capaz de quantificar atributos qualitativos do sistema;
   - Avaliação dos resultados a fim de superar as desvantagens e possivelmente melhorar o sistema nas próximas versões.

1.4 Síntese de Conteúdos

Este capítulo fez uma breve introdução sobre as motivações que levaram ao desenvolvimento do projeto, mostrando com breves detalhes o que foi construído. A seguir, é apresentada uma síntese dos conteúdos que serão abordados nos próximos capítulos.

Capítulo 2. Revisão Teórica. Uma introdução dos principais módulos desenvolvidos no trabalho será feita. Serão abordados os tópicos de reconhecimento e síntese de voz; detecção de silêncio; descrição e comparação da Beaglebone Black com algumas das plataformas embarcadas mais conhecidas; padrões de comunicação via luz infravermelha com ênfase no protocolo da Samsung; técnicas de entrada e saída de dados digitais nos pinos da Beaglebone Black; e a métrica de avaliação utilizada para verificar a qualidade e a usabilidade do sistema.

Capítulo 3. Metodologia. Neste capítulo, as atividades realizadas para a construção do projeto serão detalhadas. Serão descritos com detalhes os procedimentos utilizados para que o cliente criado para um *smartphone* Android, no qual também foi implementado um sistema de síntese de voz, pudesse se comunicar com o servidor configurado na Beaglebone Black, o qual possui o sistema de reconhecimento de fala juntamente com o emulador do protocolo de controle remoto da Samsung.

Capítulo 4. Ambiente de Teste e Resultados. O sistema foi testado com algumas pessoas, as quais forneceram um *feedback* através de um questionário de qualidade e usabilidade. Os detalhes sobre a metodologia dos testes aplicados aos voluntários e os resultados obtidos e avaliados serão discutidos neste capítulo.

Capítulo 5. Considerações Finais. Discussões sobre os resultados dos testes feitos com o grupo de voluntários serão abordadas nesse capítulo, juntamente com a conclusão do trabalho e a perspectiva para as próximas versões do projeto, esta última discutida na seção de trabalhos futuros. Detalhes sobre todas as falhas cometidas e dificuldades gerais encontradas durante o projeto também serão relatados.

Capítulo 2

Revisão Teórica

Este capítulo tem como objetivo apresentar o referencial teórico dos módulos mais importantes construídos neste trabalho, como o i) reconhecimento automático de voz; ii) síntese de voz; iii) detecção de silêncio/fala na aplicação cliente; iv) padrão da comunicação entre o controle remoto e o aparelho eletrônico, v) bem como sua principal técnica de modulação; vi) e escrita/leitura de valores digitais nos pinos da Beaglebone Black.

2.1 Reconhecimento Automático de Voz

Os sistemas de reconhecimento automático de voz (ASR) têm sido estudados desde os anos 50 [2,27,28]. Por exemplo, nos Laboratórios Bell nessa época, foi desenvolvido o primeiro reconhecedor de dígitos isolados com suporte a apenas um locutor. Na mesma década, foi introduzido o conceito de redes neurais, mas devido a muitos problemas práticos, a ideia não foi seguida no âmbito de ASR. Nos anos 70, a técnica predominante era a *dynamic time-warping* (DTW) [29], que consiste em um algoritmo que mede a similaridade entre duas sequências após alinhá-las ao longo do tempo. Com o passar dos anos, a pesquisa foi evoluindo e muitas limitações técnicas foram sendo superadas, além da globalização e popularização dos computadores, o que levou a um aumento no número de pesquisas na área de processamento de voz.

Nos anos 80, os sistemas ASR tentavam aplicar, inicialmente, um conjunto de regras gramaticais e sintáticas à fala [30]. Caso as palavras ditas caíssem dentro de um certo conjunto de regras, o programa poderia determinar quais eram aquelas palavras. Para isso, era preciso falar cada palavra separadamente (sistemas de palavras isoladas), com uma pequena pausa entre elas. Porém, características como sotaques, dialetos e as inúmeras exceções inerentes à língua difi-

cultaram a disseminação dos sistemas baseados em regras. Foi então que vários pesquisadores iniciaram estudos para reconhecimento de palavras conectadas utilizando métodos estatísticos, com maior destaque para os modelos ocultos de Markov (HMMs, do inglês *hidden Markov models*) [31, 32]. Atualmente, o estado da arte em reconhecimento de voz é representado pelo uso de HMMs, incrementado por técnicas como aprendizado discriminativo [33], seleção de misturas de gaussianas [34], entre outras.

Um sistema ASR é, tipicamente, composto por cinco blocos: *front-end*, modelo acústico, modelo de linguagem, dicionário fonético e decodificador, como indicado na Figura 2.1.

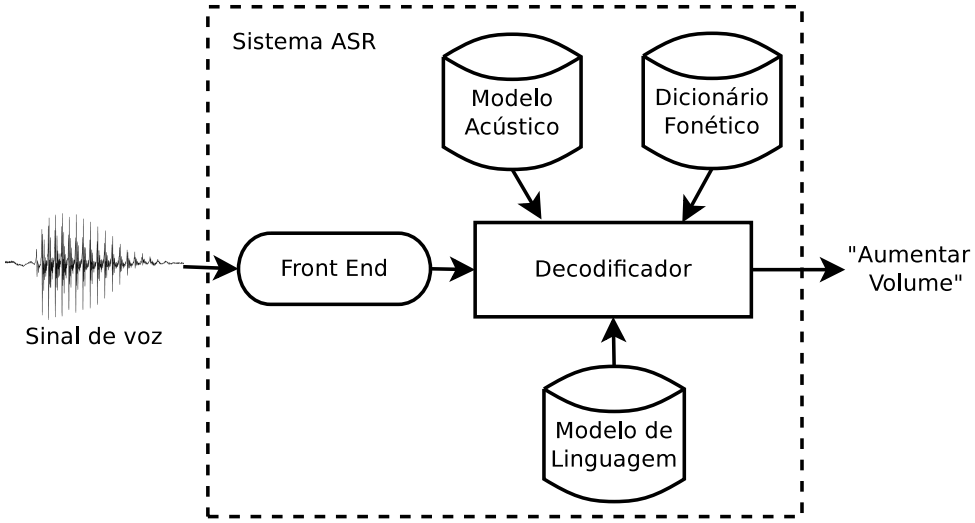

Figura 2.1: Arquitetura de um sistema ASR.

No processo de *front-end* convencional, tem-se a segmentação do sinal de voz em "janelas" curtas (*frames*) de 20 a 25 milissegundos, com o deslocamento da janela de análise sendo tipicamente de 10 milissegundos. Cada *frame* é, então, convertido em um vetor $\mathbf{x}$ de dimensão L (tipicamente, $L = 39$). É assumido aqui que T *frames* estão organizados em uma matriz $\mathbf{X}$ de $L \times T$, representando uma sentença completa.

Existem várias alternativas no que diz respeito à parametrização do sinal de voz [2,35,36]. Entretanto, a análise dos coeficientes cepstrais da escala Mel (MFCCs, do inglês *Mel-frequency cepstral coefficients*) [37] tem provado ser eficiente e é geralmente empregada como entrada para os blocos de *back-end* que compõem um sistema ASR [2].

O modelo de linguagem provê a probabilidade $p(\mathcal{T})$ de observar a sentença $\mathcal{T} = [w_1, \ldots, w_P]$ com P palavras. Conceitualmente, o objetivo é encontrar a sentença $\mathcal{T}^*$ que

maximiza a probabilidade posterior

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} p(\mathcal{T}|\mathbf{X}) = \arg\max_{\mathcal{T}} \frac{p(\mathbf{X}|\mathcal{T})p(\mathcal{T})}{p(\mathbf{X})}, \tag{2.1}$$

onde $p(\mathbf{X}|\mathcal{T})$ é dada pelo modelo acústico. Como $p(\mathbf{X})$ não depende de $\mathcal{T}$:

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} p(\mathbf{X}|\mathcal{T})p(\mathcal{T}). \tag{2.2}$$

Na prática, uma constante empírica é usada para ponderar a probabilidade do modelo de linguagem $p(\mathcal{T})$, antes da mesma ser combinada com a probabilidade do modelo acústico $p(\mathbf{X}|\mathcal{T})$.

Dado o grande volume de sentenças concorrentes, a Equação 2.2 não pode ser calculada independentemente para cada sentença candidata. Para esse fim, os sistemas ASR utilizam-se de estruturas de dados hierárquicas, como árvores léxicas, e do artifício de separar as sentenças em palavras, e as palavras em unidades básicas, chamadas de fones [2]. A busca por $\mathcal{T}^*$ é chamada decodificação e, na maioria dos casos, hipóteses são descartadas ou podadas (*pruning*). Em outras palavras, para tornar viável a busca pela "melhor" sentença, algumas candidatas são descartadas e a Equação 2.2 não é calculada para elas [38, 39].

Um dicionário fonético (também conhecido por modelo léxico) provê o mapeamento das palavras em unidades básicas (fones) e vice-versa.

Em termos de modelo acústico, no intuito de incrementar o desempenho, HMMs contínuas são geralmente empregadas, onde a distribuição de saída de cada estado é modelada por uma mistura de Gaussianas. A topologia "*left-right*" pode ser adotada, onde as únicas transições permitidas são de um estado para ele mesmo ou para o estado seguinte.

O modelo acústico pode conter uma HMM por fone. Isso seria bastante razoável caso um fone pudesse ser seguido por qualquer outro, o que não é verdade, já que os articuladores do trato vocal não se movem de uma posição para outra imediatamente na maioria das transições de fones. Nesse sentido, durante o processo de criação de sistemas que modelam a fala fluente, busca-se um meio de modelar os efeitos contextuais causados pelas diferentes maneiras que alguns fones podem ser pronunciados em sequência. A solução encontrada é o uso de HMMs dependentes de contexto, que modelam o fato de um fone sofrer influência dos fones vizinhos, conhecido como coarticulação [40]. Por exemplo, supondo a notação do trifone a-b+c, temos que b representa o fone central ocorrendo após o fone a e antes do fone c.

A modelagem da língua estima

$$P(\mathcal{T}) = P(w_1, w_2, \ldots, w_P) \tag{2.3}$$

$$= P(w_1)P(w_2|w_1)\ldots P(w_P|w_1, w_2, \ldots, w_{P-1}). \tag{2.4}$$

É impraticável estimar a probabilidade condicional $P(w_i|w_1, \ldots, w_{i-1})$, mesmo para valores moderados de i. Assim, o modelo de linguagem para sistemas ASR consiste em um modelo *n*-gram, que assume que a probabilidade $P(w_i|w_1, \ldots, w_{i-1})$ depende somente das $n-1$ palavras anteriores. Por exemplo, para $n = 3$, a probabilidade $P(w_i|w_{i-2}, w_{i-1})$ expressa um modelo de linguagem trigrama.

Basicamente, existem dois tipos de aplicação para sistemas de reconhecimento de fala: ditado, e comando e controle [2]. O primeiro, por ser capaz de suportar um vocabulário amplo, utiliza o modelo de linguagem *n*-gram e requer um sistema mais robusto, exigindo mais recursos e processamento da máquina. O segundo é relativamente mais simples, pois utiliza uma gramática livre de contexto como modelo, restringindo, assim, a sequência de palavras aceitas.

Resumindo, após o treinamento de todos os modelos estatísticos, um sistema ASR na etapa de teste usa o *front-end* para converter o sinal de entrada em parâmetros e o decodificador para encontrar a melhor sentença $\mathcal{T}$.

2.2 Síntese de Voz

A síntese de voz consiste em produzir a fala humana artificialmente, através da geração automática do sinal de voz a partir de texto [41]. Vários trabalhos em síntese de voz vêm sendo desenvolvidos há décadas e considerável avanço foi alcançado. Porém, a qualidade em termos da naturalidade da voz produzida ainda apresenta lacunas, principalmente no que tange às adaptações que a fala pode sofrer considerando aspectos como entonação e emoção, os quais estão associados à expressividade do conteúdo a ser sintetizado.

O primeiro dispositivo considerado um sintetizador elétrico foi o demonstrador de operação de voz VODER (*Voice Operating Demonstrator*), desenvolvido por Homer Dudley em 1939. Ele era composto por uma barra para selecionar o tipo de voz (vozeada ou não vozeada), um pedal para controlar a frequência fundamental e dez teclas que controlavam o trato vocal artificial. A estrutura básica do VODER é bastante similar aos sistemas baseados no modelo fonte-filtro. Atualmente, dentre as tecnologias envolvendo síntese de voz, destacam-se: por concatenação, articulatória, por formantes (regras) e baseada em HMMs [41].

Uma das aplicações de síntese de voz pode ser encontrada em sistemas TTS, os quais convertem um texto de entrada em uma voz artificial que seja inteligível e mais natural possível. A tarefa dos sistemas TTS é bastante complexa, pois envolve a imitação (*mimicking*) de como

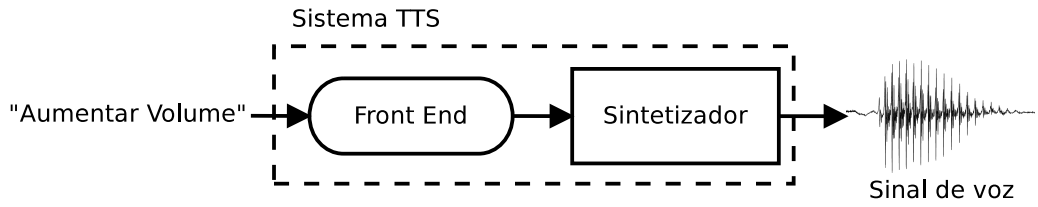

Figura 2.2: Arquitetura de um sistema TTS.

os seres humanos realizam a leitura de um texto. Segundo [1], esses sistemas são compostos por dois componentes principais: *front-end* e sintetizador, conforme mostra a Figura 2.2.

O módulo *front-end* é dependente de língua e realiza a análise do texto para que a informação de saída seja codificada de forma conveniente ao sintetizador. Por exemplo, o *front-end* executa a normalização (ou pré-processamento) do texto, convertendo símbolos, como números e abreviaturas, em palavras equivalentes escritas por extenso. Ele também implementa a conversão grafema-fone (G2P), silabificação e determinação de sílaba tônica. Essas tarefas atribuem transcrição fonética a cada palavra e marcam o texto com informações prosódicas. Transcrições fonéticas e informações prosódicas compõem juntas a representação simbólica linguística que serve de entrada para o sintetizador.

Tipicamente independente de língua, o sintetizador é o bloco que efetivamente gera o som, convertendo a representação linguística em voz. Historicamente, os sistemas TTS evoluíram a partir de um paradigma baseado no conhecimento para uma abordagem pragmática (*data-driven*), como a técnica baseada em HMMs [42].

A síntese de voz baseada em HMMs já é bastante popular devido à sua flexibilidade em sintetizar voz considerando características como estilo e individualidade da fala do locutor, além da possibilidade de expressar aspectos emocionais na voz sem precisar de uma grande quantidade de amostras de dados [43].

Um sistema TTS baseado em HMMs funciona através da execução de duas etapas: treino e síntese. Durante a fase de treinamento os parâmetros referentes ao espectro da voz e à excitação, tais como MFCCs e frequência fundamental (F0), são extraídos de uma base de voz e modelados através de HMMs, considerando aspectos fonéticos, linguísticos e prosódicos da fala. Nesse caso específico, tem-se uma HMM para cada fone (monofone), porém o modelo estatístico pode ser baseado em unidades maiores, como difones, trifones, palavras, etc.

Como resultado dessa primeira etapa, tem-se um *framework* unificado, possuindo modelos dos espectros da excitação e duração da voz. Em seguida, tem-se o processo de síntese

propriamente dito, no qual um texto qualquer de entrada é transformado em uma sequência de fones, em que cada HMM correspondente a um fone é concatenada com a HMM do fone anterior e posterior (dependente do contexto). Após o processo de concatenação, cada HMM emite um conjunto de observações (parâmetros MFCC, F0, entre outros). Por fim, uma forma de onda da voz é sintetizada diretamente a partir dos parâmetros estimados.

2.3 Detecção de Atividade de Voz

A detecção de atividade de voz (VAD, do inglês *voice activity detection*), ou detecção de silêncio/fala, refere-se aos algoritmos criados com o intuito de detectar a presença ou ausência de fala em um sinal de voz. Como existem diversos métodos de detecção e inúmeros tipos de fala e ruído, não se pode dizer que há um algoritmo "perfeito", já que cada um funciona bem para sua devida aplicação (ASR, VoIP, cancelamento de eco, etc) [44]. Um bom algoritmo de VAD deve saber explorar características da fala para que a regra de decisão criada seja eficaz; ser robusto para saber se há ruídos de fundo não estacionários; ter complexidade computacional baixa para aplicações em tempo-real; e, obviamente, possuir poucos erros de detecção [45].

A detecção de silêncio baseada na energia do sinal é uma das mais utilizadas, já que é método simples, com atraso moderado no processamento e que funcionam bem em ambientes "limpos", ou seja, com pouco ruído [46]. No entanto, tais algoritmos costumam falhar quando o ruído varia muito rapidamente ou quando a razão sinal-ruído (SNR, do inglês *signal-to-noise ratio*) é muito baixa (ou seja, quando a energia do ruído é maior do que a dos segmentos de fala). Técnicas mais robustas atingem uma taxa de acerto mais elevada por realizarem a análise do sinal no domínio da frequência, mas a exigência de um maior poder de processamento da máquina tornam-as inviáveis para sistemas *online* de menor porte [47].

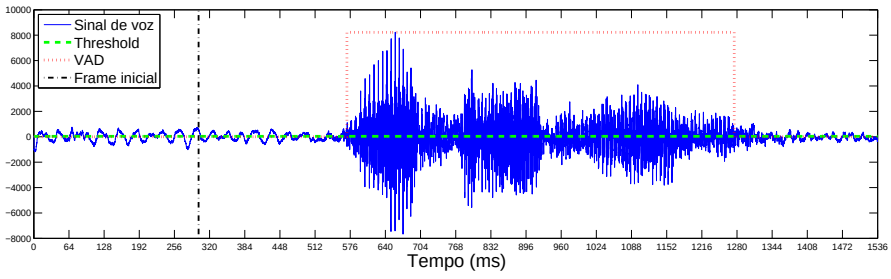

Figura 2.3: Detecção de fala/silêncio em um sinal de voz digitalizado baseada em um valor estático de *threshold* da energia do sinal.

O algoritmo mais simples é baseado em um valor estático de *threshold*. A energia do ruído do início da fala (silêncio) E_R é estimada e multiplicada por um fator de escala $\kappa > 1$ a fim de se calcular um *threshold* T, o qual define um limiar que deve separar segmentos vozeados dos não vozeados. O sinal é, então, dividido em janelas ou *frames*, cuja energia E_F de cada um é calculada e comparada com o *threshold*: se E_F for maior do que T, assume-se que a fala está presente; do contrário, ausente. A Figura 2.3 mostra a separação entre ruído e silêncio em um sinal de voz. O pseudocódigo a seguir ilustra, minimamente, o algoritmo descrito acima.

```
T = κ × E_R
if E_F > T:
    fala = True
else:
    fala = False
```

A energia E de sinais reais e discretos no tempo é calculada através da soma dos quadrados das N amostras do x-ésimo *frame*, conforme mostrado na Equação 2.5.

$$E_x = \sum_{n=x \cdot N}^{(x \cdot N)+N-1} x[n]^2 \tag{2.5}$$

O uso de janelamento ocorre na maioria dos casos, nos quais o comprimento do *frame* de amostras pode variar de 5 ms à 50 ms, dependendo da aplicação. Para transmissão via rede, o sinal é tipicamente janelado a uma frequência de 100 Hz. A estimativa do ruído/silêncio inicial, entretanto, é calculada pela média da energia de um conjunto de 10 a 20 *frames* iniciais, assumindo que não houve atividade do usuário nos 100 ms ou 200 ms iniciais.

2.4 Modulação por Largura de Pulso

A modulação por largura de pulso (PWM, do inglês *pulse-width modulation*), é uma técnica para conseguir resultados analógicos por meios digitais [48]. O controle digital é usado para criar um sinal que oscila entre “ligado” e “desligado”, produzindo uma onda quadrada cuja tensão média é, no caso da Beaglebone Black, um valor entre 0 V e 3,3 V. A maioria dos protocolos de comunicação entre controles remotos e dispositivos eletrônicos utiliza PWM para reduzir a energia ao piscar o LED infravermelho (LED IR) e minimizar interferências e ruídos.

A onda gerada pela PWM possui alguns parâmetros, dos quais destacam-se o período e o *duty cycle*. O primeiro é bem simples, dado pela distância entre um pulso e o próximo

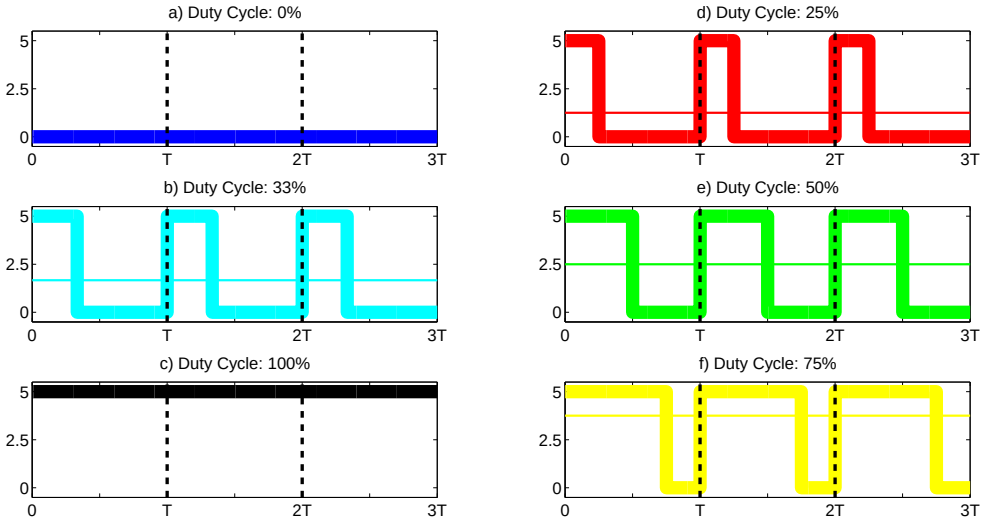

Figura 2.4: Exemplos de PWMs de mesmo período com diferentes *duty cycles*.

(que seriam, em uma onda analógica, duas cristas consecutivas); já o segundo é dado pela percentagem em que esse mesmo período permaneceu em nível alto (ligado), representando o período apenas do pulso. Existem outros parâmetros como largura do pulso e frequência de PWM, mas estes podem ser obtidos a partir das duas informações mencionadas acima.

A Figura 2.4 mostra seis sinais de mesma duração ($3T$) e mesmo período (T), porém com *duty cycle* variado. Há dois casos especiais, mostrados em 2.4a) e 2.4c), nos quais o *duty cycle* é, respectivamente, nulo (0%) e igual ao período da PWM (100%). Nesses casos, a tensão média (representada pela linha horizontal nas demais ondas) será mínima (0 V) e máxima (3,3 V).

2.5 Protocolos de Comunicação via Luz Infravermelha

O funcionamento de controles remotos, com ênfase nos baseados em luz infravermelha para televisores, é explicado de forma clara e detalhada em diversos tutoriais para "curiosos" disponíveis na Internet [49, 50]. Atualmente, a maioria dos aparelhos eletrônicos recebe informação através de sensores infravermelhos localizados em painéis frontais. Entretanto, devido à interferência que surge com a vasta quantidade de informação IR no ambiente (luz solar, lâmpadas fluorescentes, etc), a comunicação entre o controle remoto e a televisão ocorre geralmente em 4 passos: um comando *start* inicia a transferência; seguido dos bits do comando específico (como "aumentar o volume", por exemplo) e do endereço do aparelho; por fim, um

comando de *stop* encerra o envio de bits. Dessa forma, a chance de a informação ser reconhecida por mais de um aparelho é baixa (exceto se forem equipamentos do mesmo tipo e da mesma empresa, como duas TVs Samsung de um modelo hipotético `X`).

Uma das grandes dificuldades relacionadas ao controle de equipamentos eletrônicos é que os 4 passos descritos acima são apenas uma forma genérica de descrever a comunicação. Ou seja, cada empresa praticamente segue o seu próprio padrão para estabelecer a comunicação entre o controle e os dispositivos: o número, a ordem, a duração e o significado dos bits é variado; a modulação e codificação usadas são diferentes; e a frequência dos pulsos pode oscilar entre 32 kHz e 42 kHz, chegando a 50 kHz em determinados aparelhos mais modernos. Além disso, tão rara quanto o seguimento de uma comunicação via infravermelho padronizada é a disponibilização de documentação pelos fabricantes.

O sistema apresentado no trabalho utiliza uma televisão da Samsung. Portanto, somente o protocolo utilizado por essa fabricante será detalhado.

2.5.1 O Protocolo da Samsung

A Samsung, assim como a maioria dos demais fabricantes, utiliza seu próprio padrão para comunicação entre os controles e os dispositivos eletrônicos. Felizmente, o documento de *application notes* do microcontrolador S3F80KB [51] presente em controles da Samsung encontra-se disponível na Internet[1].

O protocolo define uma sequência de 34 bits, onde ambos os valores ‘0’ e ‘1’ são representados por uma mudança no estado e diferenciados pela duração do nível baixo dos pulsos, conforme mostrado na Figura 2.5. Os bits são definidos sempre como uma transição *high-to-low* durante um período, na qual é estabelecida uma única duração de 560 μs para o nível alto do pulso (modo *burst*) e duas para o nível baixo: 1690 μs para o bit ‘1’ e 560 μs para o bit ‘0’. Ou seja, ambos os bits começam seus respectivos períodos com um nível alto, mas o bit ‘1’ diferencia-se por possuir duração mais longa no nível baixo.

O nível alto dos bits normais é caracterizado por uma PWM com frequência de 37,9 kHz e *duty cycle* que pode variar entre 33% e 50% do período. A PWM caracteriza o modo *burst*, em que o LED IR pisca na frequência definida pelo protocolo. Já o nível baixo é caracterizado pela ausência de sinal, ou seja, é o tempo em que o LED IR permanece desligado ou em estado *idle*. O primeiro bit, chamado de *leader*, tem duração mais longa para garantir a sincronia e o

[1] `http://elektrolab.wz.cz/katalog/samsung_protocol.pdf`

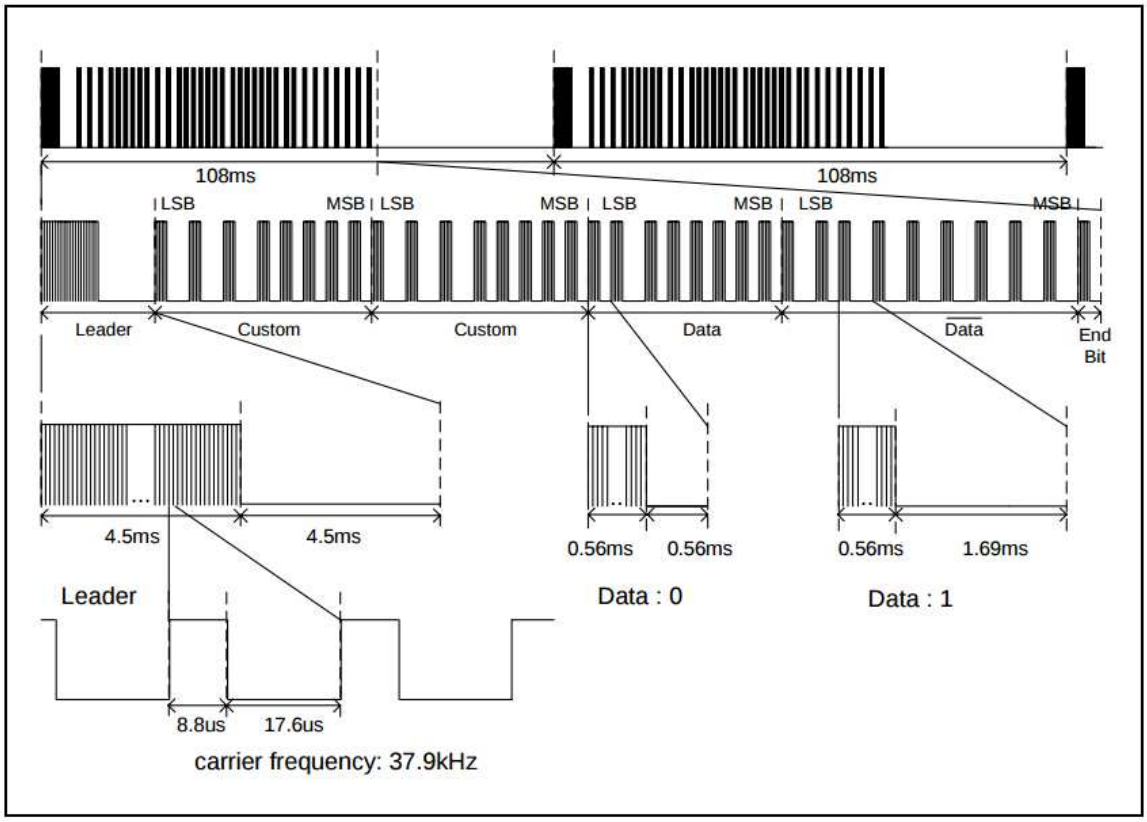

Figura 2.5: Esquemático do protocolo da Samsung (Samsung Electronics, 2008).

ganho no circuito receptor: 4,5 ms no nível alto e 4,5 ms no nível baixo; os 16 bits seguintes são divididos em dois blocos personalizados (definidos como *custom*) exatamente iguais de 8 bits cada, provavelmente definindo o endereço do aparelho; outras duas seções de 8 bits definem o bloco de dados, no qual o segundo bloco é o complemento dos bits do primeiro, que define os comandos propriamente ditos; o 34° bit é sempre baixo e encerra a sequência.

2.6 Beaglebone Black

A Beaglebone Black[2] (BBB) é um microcomputador embarcado, baseado em Linux, com *hardware* e *software* abertos, voltado principalmente para desenvolvedores. Fabricada pela Texas Instruments, a plataforma embarcada "cabe na palma da mão" e, apesar do tamanho reduzido, consegue equiparar-se a computadores moderadamente robustos, mantendo um custo relativamente baixo. A Figura 2.6 mostra alguns detalhes a respeito de componentes de *hardware* da Beaglebone Black.

[2] `http://beagleboard.org/black`

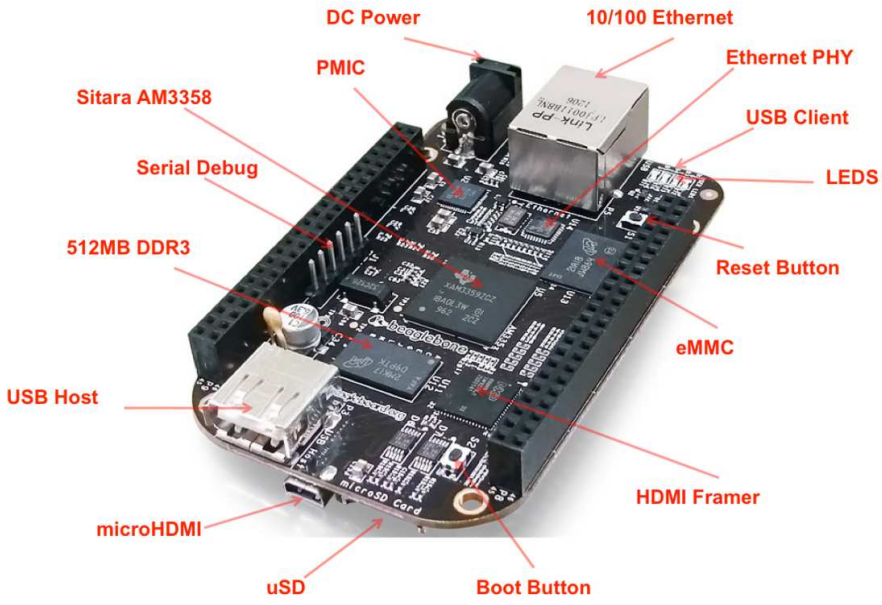

Figura 2.6: Detalhes do *hardware* da Beaglebone Black (Beagleboard.org, 2016).

A Beaglebone Black foi comparada, para fins informativos sobre suas capacidades computacionais, bem como para fins de decisão sobre o componente base do projeto, a duas outras plataformas embarcadas bastante conhecidas no mundo eletrônico: Arduino e Raspberry Pi. As especificações foram consultadas em *blogs* [52–55], documentações oficiais [56–59] e não oficiais [60–62].

O Arduino, apesar de flexível e capaz de estabelecer interfaces com uma vasta quantidade de dispositivos, é uma plataforma simples e não atenderia aos requisitos do projeto. A placa, construída sobre o microcontrolador ATMega, embora sendo de baixo custo e tendo uma interface de programação bem simples, torna-se muito limitada quando um processamento relativamente potente é exigido.

O Raspberry Pi já se enquadra no conceito de microcomputador, possuindo muitas portas USB, uma HDMI e vários pinos de propósito geral. A plataforma requer a instalação de um sistema operacional completo, além de possuir uma excelente placa de vídeo para aplicações gráficas. A conexão com a Internet, no modelo 2 B, dá-se através de um conector Ethernet.

A revisão C da Beaglebone Black é comparável ao Raspberry Pi 2 modelo B. Essas duas versões são as mais modernas das respectivas plataformas [3]. A revisão B da Beaglebone Black foi utilizada no trabalho e, apesar de não ser a versão mais atual, continua sendo uma ótima escolha para projetos relativamente complexos por possuir 512 MB de memória RAM, 2 GB

[3] Comparação feita em dezembro de 2015.

de memória *flash* eMMC para armazenamento, e um poderoso processador ARM Cortex-A8 de 1 GHz. Além de possuir diversas opções de conexão e um total de 92 pinos divididos em dois *headers* (P8 e P9), a BBB une a flexibilidade de interfaceamento do Arduino com a capacidade de processamento rápido do Raspberry Pi. A Tabela 2.1 mostra uma comparação entre as especificações das três plataformas discutidas.

Tabela 2.1: Especificações do Arduino, da Beaglebone Black e do Raspberry Pi.

	Arduino UNO R3	Beaglebone Black B	Raspberry Pi 2 B
SoC	-	TI AM335x	BCM2836
CPU	ATMega328	ARM Cortex-A8	ARM Cortex-A7
CPU Freq.	16 MHz	1 GHz	900 MHz
GPU	-	PowerVR SGX530	Dual Core VideoCore IV
RAM	2 kB SRAM	512 MB SDRAM DDR3L	1 GB SDRAM LPDDR2
RAM Freq.	-	800 MHz	400 MHz
Flash	32 kB	2 GB eMMC + MicroSD	Micro SDIO
GPIOs	20	69	40
Video	-	mini HDMI	HDMI
Sistema Op.	-	Linux, etc	Windows, Linux, etc
Amperagem	42 mA	210-460 mA	700 mA
Voltagem	7-12 V	5 V	5 V
USB	-	1 Host, 1 Mini Client	4 Hosts, 1 Micro Power
Ethernet	-	1 10/100 Mbps	1 10/100 Mbps
Preço[4]	U$ 24.95	U$ 45.00	U$ 39.95

Por ter uma distribuição Linux como sistema operacional de fábrica, a Beaglebone Black possui ampla aceitação dos usuários da comunidade e dá suporte a muitas das bibliotecas disponíveis para as versões *desktop* do sistema operacional, como a de visão computacional OpenCV e a de interação natural OpenNI, por exemplo. A capacidade de interfaceamento com *drivers* de robôs é algo que também chama atenção. Dois bons exemplos são os Hexapods, robôs em forma de caranguejo baseados no *framework* ROS (*robot operating system*) [63, 64]; e o OpenROV [65], o qual utiliza a BBB na construção de um robô subaquático *open source*.

[4]Retirado de `https://www.adafruit.com/` (dezembro de 2015).

2.6.1 Expansão da Pinagem dos *Headers* P8 e P9

Cada pino digital da Beaglebone Black (BBB) possui até 8 modos diferentes de funcionamento, os quais incluem: comunicação UART ou SPI; captura de eventos via eCAP; saída de dados via HDMI; controle da eMMC; PWM; GPIO; etc. O processo de escolha da função do pino, conhecido por multiplexação (*pin muxing* ou simplesmente PinMux), exige que o programador saiba exatamente quais as funções que cada pino pode assumir e se o mesmo está disponível, já que o acesso aos modos de alguns pinos não é recomendado. O PinMux ocorre pelo ajuste da árvore de dispositivos do Linux (*device tree*), permitindo que o sistema operacional identifique, no momento do *boot*, todos os periféricos de *hardware* disponíveis.

Nas versões mais recentes do *kernel* do Linux, recompilação da árvore e reinicialização do sistema operacional em caso de alteração não é mais necessária: o módulo é acrescentado ou "sobreposto" à árvore original e carregado dinamicamente através de um mecanismo chamado *Device Tree Overlay* (DTO) [66]. Os arquivos de descrição de *hardware*, chamados de *Device Tree Sources* (DTS), devem ser criados e convertidos para linguagem de máquina pelo compilador da árvore de dispositivos (DTC, *Device Tree Compiler*), o qual irá gerar o arquivo binário *Device Tree binary Blob* (DTB) que permitirá que os novos dispositivos sejam carregados no sistema [67]. O DTB é uma estrutura de dados que, quando acoplada à árvore principal (a que é carregada no momento do *boot*), faz com que o *kernel* reconheça o novo dispositivo.

A seguir, dois modos de acesso aos pinos serão descritos. O primeiro dá-se através da simples leitura e escrita de arquivos no sistema de arquivos Linux. Já o segundo utiliza recursos específicos do *hardware* da Beaglebone Black através de DTOs.

2.6.1.1 Acesso aos Pinos através do Sistema de Arquivos

A Beaglebone Black possui 69 pinos que podem ser definidos como entrada ou saída de propósito geral (GPIO, do inglês *general purpose input/output*) divididos em 4 bancos de, no máximo, 32 módulos cada. O modo GPIO descreve o comportamento ou direção dos pinos (entrada ou saída) — similar ao que ocorre com o uso da função `pinMode()` do Arduino —, os quais podem ser controlados dinamicamente pelo usuário mediante o ajude de parâmetros de configuração do *kernel* do Linux.

Um dos modos mais simples de controle e acesso aos módulos GPIO foi introduzido com a versão 3.8.x do *kernel* do Linux embarcado. O PinMux ocorre pela simples atribuição de valores a arquivos específicos pelo SysFS, desde que o usuário Linux tenha as permissões de

edição ou seja o *root*. Tal operação oferece uma interface mais amigável para o programador, já que todo o processo de DTO é mascarado.

```
#!/bin/bash
echo 115  > /sys/class/gpio/export
echo out > /sys/class/gpio/gpiochip115/direction
for i in 'seq 10'; do
    echo 1 > /sys/class/gpio/gpiochip115/value
    sleep 0.5
    echo 0 > /sys/class/gpio/gpiochip115/value
    sleep 0.5
done
echo 115 > /sys/class/gpio/unexport
```

Listagem 2.1: Acesso ao GPIO 115 / P9_27 / GPIO3[19] através de arquivos.

Os arquivos de configuração do GPIO estão localizados no caminho `/sys/class/gpio`. Ao se escrever o número do GPIO (`XX`) no arquivo `export`, o qual configura automaticamente o pino para o modo 7 (modo o qual é exclusivo para GPIOs em todos os pinos da BBB), o diretório `gpiochipXX`, que contém os arquivos necessários para o acesso e controle do pino, é criado. Apenas os arquivos `direction` e `value` são necessários para a tarefa de piscar um LED, os quais indicam, respectivamente, se o pino será

Tabela 2.2: Modos dos pinos 13, 27 e 28 do *header* P9 da Beaglebone Black (P9_xx).

	P9_13	P9_27	P9_28
Pino SoC	U17	C13	C12
Nome	UART4_TXD	GPIO3_19	SPI1_CS0
Modo 0	GPMC_wpn	McASP0_fsr	McASP0_ahclkr
Modo 1	MII2_rxerr	eQEP0B_in	EHRPWM0_synci
Modo 2	GPMC_csn5	McASP0_axr3	McASP0_axr2
Modo 3	MII2_rxerr	McASP1_fsx	SPI1_cs0
Modo 4	MMC2_sdcd	EMU2_MUX2	eCAP2_in_PWM2_out
Modo 5	-	pr1_PRU0_PRU_R30_5	**pr1_PRU0_PRU_R30_3**
Modo 6	UART4_txd_MUX2	pr1_PRU0_PRU_R31_5	**pr1_PRU0_PRU_R31_3**
Modo 7	GPIO0[31]	**GPIO3[19]**	GPIO3[17]

de entrada ou saída ("*in*" ou "*out*") e o valor digital ("0" ou "1") que será escrito ou lido.

A Listagem 2.1 mostra um código capaz de piscar um LED conectado ao pino 27 do *header* P9 (P9_27) com frequência de 1 Hz. O número de GPIO é calculado multiplicando-se o número do banco (de 0 a 3) por 32 e somando ao número do módulo (de 0 a 31). No Modo 7, GPIO3[19] significa que o pino P9_27 será multiplexado para o módulo 19 do banco 3 dos GPIOs, conforme visto na Tabela 2.2. Portanto, seu número de GPIO é $\texttt{XX} = 3 \times 32 + 19 = 115$.

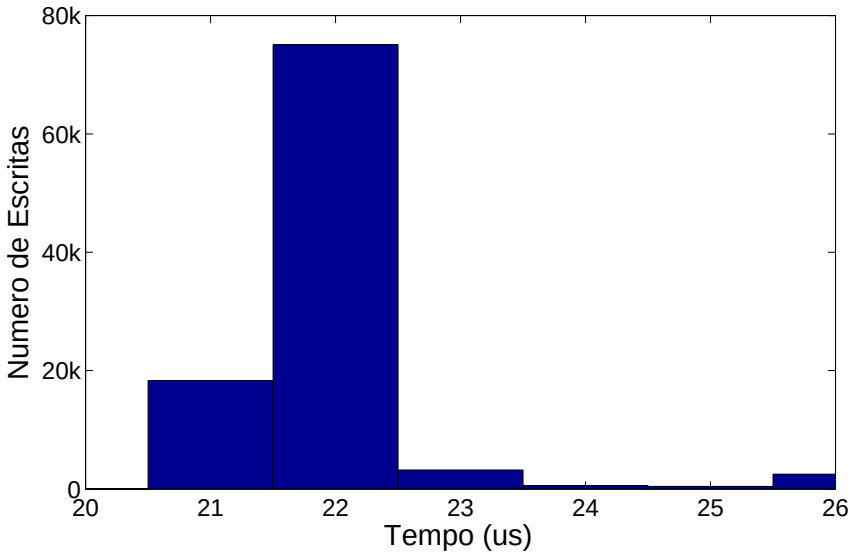

Figura 2.7: Tempo para atribuir valor a um pino pelo método de escrita em arquivos.

Apesar da facilidade em controlar a voltagem dos pinos pela manipulação de arquivos, tal forma de atribuição de valores não atende à frequência exigida pela PWM do protocolo da Samsung. O histograma da Figura 2.7 mostra que a alternância entre os níveis do GPIO dura, na maioria das vezes, 22 μs. Como o período de um ciclo da PWM dura exatos 26,385 μs ($1/37900\ Hz$) e o *duty cycle* dura 33% desse período (8,8 μs), nota-se que é o processo de emular uma PWM por GPIOs é lento demais e, portanto, inviável.

2.6.1.2 Acesso aos Pinos através da Unidade Programável em Tempo Real (PRU)

A Beaglebone Black possui um subsistema de unidades programáveis em tempo real (PRUSS, do inglês *programmable real-time unit subsystem*), que consiste de dois *cores* de 32 bits RISC e *clock* de 200 MHz (PRU 0 e PRU 1, *programmable real-time units*) separados do processador Cortex-A8. Ambos possuem 8 kB de memória dedicada e acesso direto ao barramento de entrada e saída (I/O), permitindo flexibilidade na implementação de interfaces

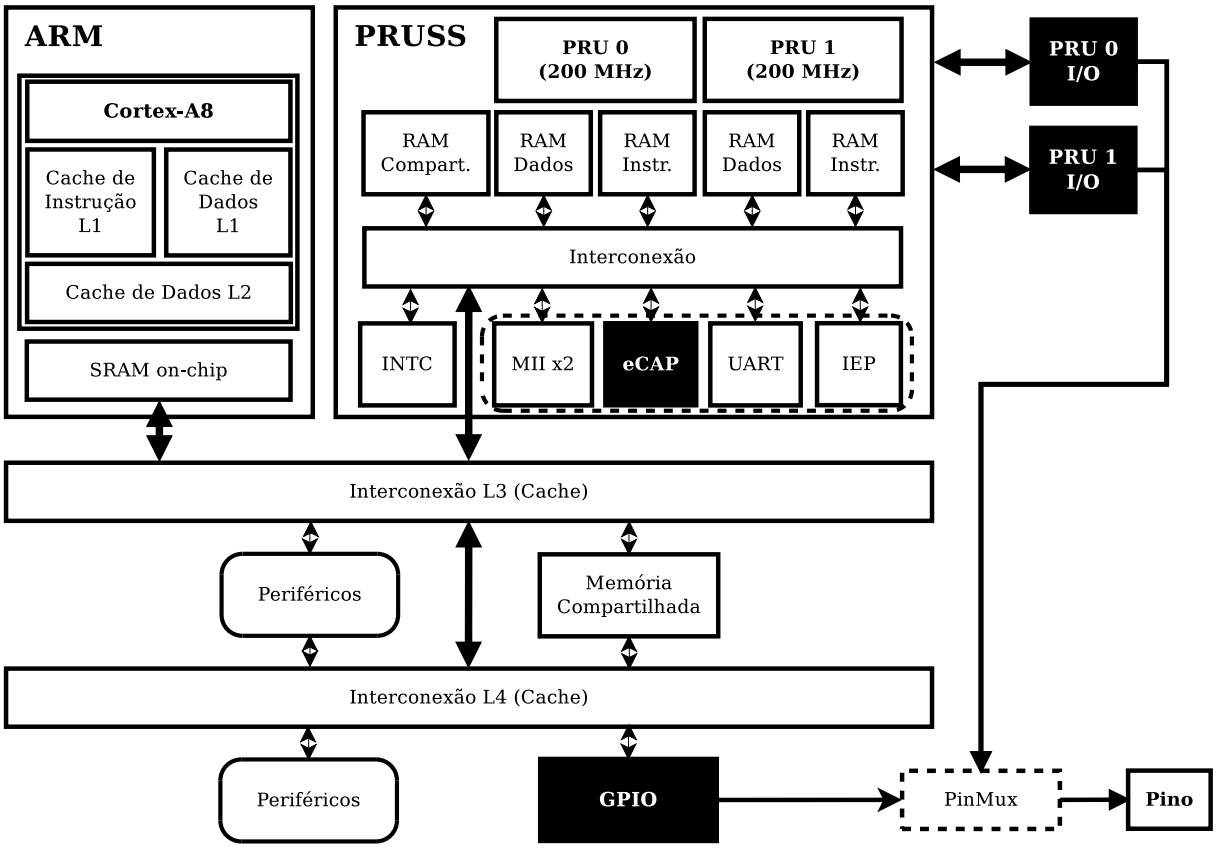

Figura 2.8: Arquitetura do ARM Cortex-A8 e do PRUSS (adaptado de TI Boot Camp, 2014).

periféricas personalizadas, rapidez nas respostas, economia de energia e manipulação especializada de dados [68]. Cada PRU é um microcontrolador de baixa latência integrado ao *system on chip* (SoC) AM335x que executa a maioria dos comandos em um único ciclo de *clock*. Sem a presença de *cache*, *pipeline* e interrupções frequentes, a PRU permite uma previsão determinística do tempo de acesso aos pinos [69, 70].

A Figura 2.8 mostra a arquitetura do ARM Cortex-A8 e do PRUSS. Nota-se que o acesso aos pinos de entrada e saída pelas PRUs é direto (PRU I/O 0 e 1, mais à direita), enquanto o processador ARM Cortex-A8 precisa passar por 4 níveis de *caching* (L1 à L4). O acesso aos periféricos locais ou integrados (UART, eCAP, etc., mostrados em linha tracejada) também permite uma latência bastante reduzida quando comparada aos periféricos externos, os quais precisam passar pelos mesmos níveis L1 à L3/L4 de *cache* quando acessados pelo Cortex-A8.

O acesso indireto aos componentes de *hardware* faz com que o ARM tenha um desempenho, no melhor dos casos, 40 vezes menor do que o das PRUs, estando ainda sujeito à *jitter* quando os componentes são acessados por arquivos, já que o *kernel* possui atrasos variados devido à alocação de processos na memória e às interrupções.

Para assinalar ou determinar valores nos pinos no modo GPIO, as PRUs escrevem ou lêem dados em determinados bits de dois registradores específicos: R30 e R31. Na Tabela 2.3, nota-se que há dois modos do pino 41 do *header* P8 (P8_41) que permitem o acesso ao *core* 1 do PRUSS (PRU 1): modos 5 e 6.

No modo 5, é possível declarar o pino P8_41 como saída e definir seu estado digital através da escrita de valores no bit 4 do registrador R30 (R30.04). Em contrapartida, o mesmo pino pode ser definido como entrada se o mesmo for multiplexado para o modo 6, e o seu valor pode ser determinado pela leitura do bit 4 do registrador R31 (R31.04). O registrador R30 sempre será utilizado para escrita/saída de dados, enquanto o R31 sempre será utilizado para leitura/entrada de dados [71]. Normalmente os bits de ambos os registradores serão iguais (bit 4, no caso do pino P8_41), mas nada foi encontrado na documentação que afirmasse que eles têm de ser necessariamente iguais.

Dentre os periféricos, encontra-se o módulo eCAP (*enhanced capture*), o qual pode ser utilizado para gerar um trem de pulsos com a mesma forma que a PWM característica do protocolo de comunicação infravermelha. A frequência e a largura dos pulsos podem ser definidas pelo programador no código do programa. O pino 13 do *header* P8 (P8_13), quando multiplexado para o modo 4, configura uma PWM (EHRPWM, do inglês *enhanced high resolution* PWM) através do eCAP, como mostrado na Tabela 2.3.

Tabela 2.3: Modos dos pinos 41, 13 e 16 do *header* P8 da Beaglebone Black (P8_xx).

	P8_41	P8_13	P8_16
Pino SoC	T1	T10	V13
Nome	GPIO2_10	EHRPWM2B	GPIO1_14
Modo 0	LCD_data4	GPMC_ad9	GPMC_ad14
Modo 1	GPMC_a4	LCD_data22	LCD_data17
Modo 2	-	MMC1_dat1	MMC1_dat6
Modo 3	eQEP2A_in	MMC2_dat5	MMC2_dat2
Modo 4	-	**EHRPWM2B**	eQEP2_index
Modo 5	**pr1_PRU1_PRU_R30_4**	-	-
Modo 6	**pr1_PRU1_PRU_R31_4**	-	**pr1_PRU0_PRU_R31_14**
Modo 7	GPIO2[10]	GPIO0[23]	GPIO1[14]

2.7 Métrica de Avaliação

A escala de usabilidade de sistema (SUS, do inglês *system usability scale*) [72], a qual consegue mensurar a usabilidade de forma confiável, foi utilizada para avaliar o protótipo de forma quantitativa através de atributos qualitativos. O método de avaliação consiste em uma pesquisa de 10 itens, conforme enumerado abaixo, cada uma com 5 opções de resposta, que variam de "discordo fortemente" a "concordo fortemente" (em uma escala 'E' discreta de 1 até 5), permitindo avaliar uma grande quantidade de produtos e serviços, incluindo *hardware*, *software*, dispositivos móveis e aplicações diversas. Os itens são enumeradas abaixo e foram livremente traduzidos do inglês.

1. "Acho que eu gostaria de usar frequentemente esse sistema."
2. "Achei o sistema desnecessariamente complexo."
3. "Achei que o sistema foi fácil de usar."
4. "Acho que eu precisaria de suporte técnico para ser capaz de utilizar o sistema."
5. "Achei que as várias funções desse sistema foram bem integradas."
6. "Acho que há muita inconsistência nesse sistema."
7. "Acho que a maioria das pessoas aprenderiam a usar esse sistema rapidamente."
8. "Achei esse sistema muito difícil/incômodo de se usar."
9. "Senti-me muito confiante utilizando esse sistema."
10. "Precisei aprender muitas coisas antes que eu pudesse utilizar esse sistema."

Todos os itens devem ser preenchidos, não existindo, portanto, a opção "pular item". Caso o entrevistado sinta que não pode responder a algum item, o ponto central da escala, que equivale ao número 3, deve ser marcado. É recomendado que o teste de usabilidade seja aplicado ao usuário imediatamente após o mesmo ter concluído o uso do sistema. Nenhum comentário deve ser feito e nenhuma discussão iniciada antes que o usuário tenha a oportunidade de responder aos itens. Além disso, os usuários não devem pensar sobre os itens por um longo período de tempo. A impressão imediata sobre o item deve ser escrita.

A interpretação da pontuação não é tão simples quanto parece. O escore final do SUS varia de 0 a 100, e cada item tem uma "contribuição" que varia de 0 a 4 de acordo com as operações descritas a seguir. Para os itens de índice ímpar, a contribuição é o valor da escala

menos 1 (E-1). Já para os de índice par, a contribuição é dada ao subtrair-se o valor da escala do número 5 (5-E). Por exemplo, se os 10 valores respondidos pelo usuário, na escala, foram `[5 4 2 1 2 3 2 4 5 2]`, as respectivas contribuições seriam, portanto, `[4 1 1 4 1 2 1 1 4 3]`. Essas contribuições são, então, somadas e multiplicadas por 2,5, o que resulta num escore final de $22 \times 2{,}5 = 55$. A Tabela 2.4 mostra as operações aplicadas aos resultados hipotéticos exemplificados.

Tabela 2.4: Exemplo de cálculo do escore do SUS.

Índice	1	2	3	4	5	6	7	8	9	10	**Soma**
Escala (E)	5	4	2	1	2	3	2	4	5	2	
Operação	E-1	5-E	E-1	5-E	E-1	5-E	E-1	5-E	E-1	5-E	
Contribuição	4	1	1	4	1	2	1	1	4	3	22
Escore											$2.5 \times 22 = 55$

Apesar de a pontuação geral variar de 0 a 100, ela não deve ser considerada uma porcentagem. Baseado em [73], uma pontuação acima de 68 seria considerada acima da média e qualquer outra pontuação, abaixo da média. Assim, uma pontuação 70 equivaleria a algo mais próximo de 50% do que a pontuação 50 propriamente dita, por exemplo.

2.8 Ferramentas

A API `TextToSpeech`, nativa do Android [74], foi utilizada para prover respostas audíveis na forma de voz sintética no *smartphone* do usuário. É necessário que os pacotes para língua portuguesa sejam previamente baixados para o aparelho, para que o sistema não precise conectar-se à Internet e continue funcionando (desde que o aparelho seja configurado corretamente) de forma *offline*.

O Julius é o *software* que realizará o reconhecimento automático de voz, sendo capaz de processar e decodificar áudio em aproximadamente tempo real para tarefas de ditado de até 60 mil palavras [8]. Para que o Julius possa realizar o reconhecimento em Português Brasileiro, serão necessários basicamente três recursos: um modelo acústico, um dicionário fonético e um modelo de linguagem.

Grandes esforços em relação à pesquisa e ao desenvolvimento de ferramentas para reconhecimento automático de fala em Português Brasileiro foram feitos pelo Grupo FalaBrasil,

o qual disponibiliza recursos e uma API gratuita para ASR em PT_BR, chamada Coruja [75]. Modelos acústicos genéricos para PT_BR podem ser encontrados na página FalaBrasil[5], bem como o conversor grafema-fonema que cria o dicionário fonético (G2P, do inglês *grapheme to phoneme*) [76]. Neste trabalho, o modelo acústico não foi treinado, pois o desempenho do modelo disponível no site foi suficiente.

Para ler o sensor IR e configurar a PWM necessária para pulsar o LED IR, foi-se utilizado um *driver* para os microprocessadores da família AM335x, desenvolvido para fácil configuração e manipulação de dados da PRU com muita rapidez. Uma vez que o acesso à unidade é, atualmente, dado apenas[6] por códigos asm, a biblioteca `libpruio` [9] disponibiliza uma coleção de exemplos e *wrappers* com sua API na linguagem C que "escondem" o Assembly, bem como arquivos DTS que mascaram os mapeamentos de memória e demais processos necessários durante o ajuste da árvore de dispositivos.

Para as versões mais antigas do sistema operacional Android, Froyo (2.2) e Gingerbread (2.3), o aplicativo foi desenvolvido utilizando a IDE Eclipse [6] na versão 3.8.1, juntamente com um *kit* de desenvolvimento disponibilizado pela Google para desenvolvedores de aplicativos para Android (SDK e ADT, do inglês *software development kit* e *Android development tools*, respectivamente) [77]. Nas versões mais atuais, Jelly Bean (4.3) e KitKat (4.4.2), a IDE Android Studio [7] na versão 1.0.2 foi utilizada.

[5] `http://laps.ufpa.br/falabrasil/`

[6] Já existe um GCC para a PRU, porém é utilizado somente pela Texas Instruments (setembro, 2015).

Capítulo 3

Metodologia

O sistema foi construído sob a ideia do controle remoto universal e acessível às PCD. A Beaglebone Black age como um *gateway* no ambiente residencial e atua também como um servidor de voz centralizado. O dispositivo tratado como controle remoto é um *smartphone* Android, o qual recebe os comandos de voz do usuário, encaminha-os ao servidor para que o reconhecimento seja efetuado e também é capaz de informar, através de uma voz sintética, o *status* da ação de controle realizada. Obedecendo a exigência imposta pela comunicação infravermelha entre o controle remoto e o aparelho eletrônico, o *gateway* foi posicionado na "linha-de-visão" do aparelho a ser controlado, e o protocolo da comunicação foi analisado, armazenado e codificado via *software*. A Figura 3.1 mostra a arquitetura geral do sistema.

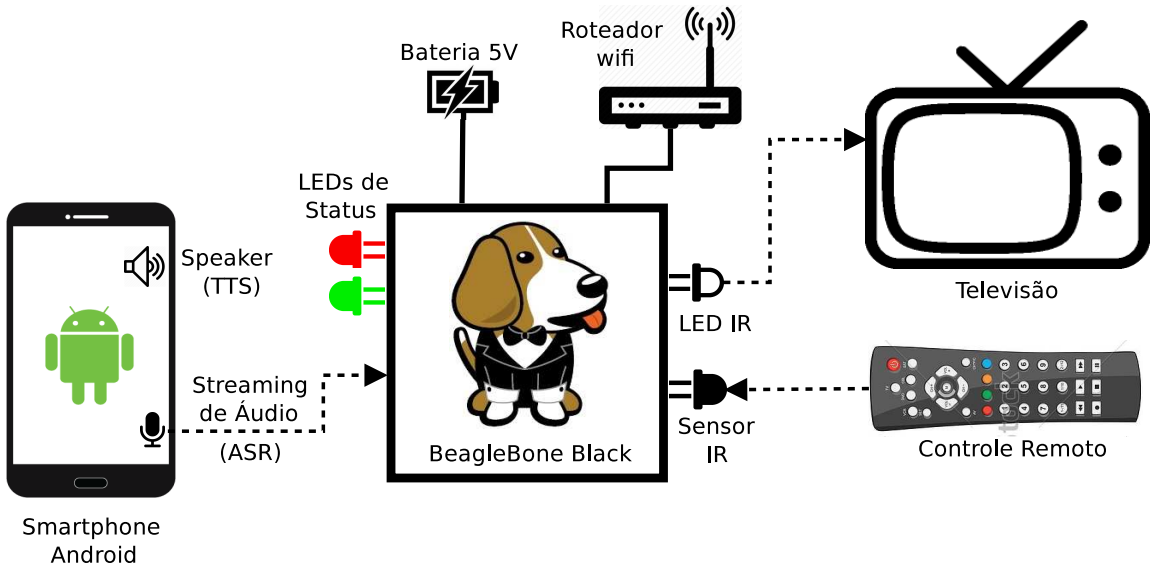

Figura 3.1: Esquemático do projeto.

3.1 Sistema Cliente-Servidor

O *smartphone* Android e o microcomputador Beaglebone Black (BBB) foram configurados para serem, respectivamente, o cliente e o servidor do sistema de controle remoto. O sistema cliente-servidor desenvolvido foi chamado de LaPS CSR (LaPS *Cloud Speech Recognition*, ou reconhecimento de fala em nuvem do Laboratório de Processamento de Sinais) [78]. A comunicação entre os dois é, atualmente, estabelecida em rede local via *socket*. O funcionamento e configuração de cada um será descrito com detalhes a seguir.

3.1.1 Cliente: *Smartphone* Android

Inicialmente, uma aplicação foi construída sobre as plataformas 2.2 e 2.3 do sistema operacional Android (Froyo e Gingerbread, respectivamente) exclusivamente para se comunicar com o servidor. Entretanto, apesar de ter sido desenvolvido para uma versão antiga, o aplicativo já foi atualizado e testado com sucesso em dispositivos nas versões 4.2 (Jelly Bean) e, mais recentemente, 4.4.2 (KitKat). A Figura 3.2 mostra o esquemático de funcionamento do aplicativo e sua interação com a nuvem de reconhecimento de voz.

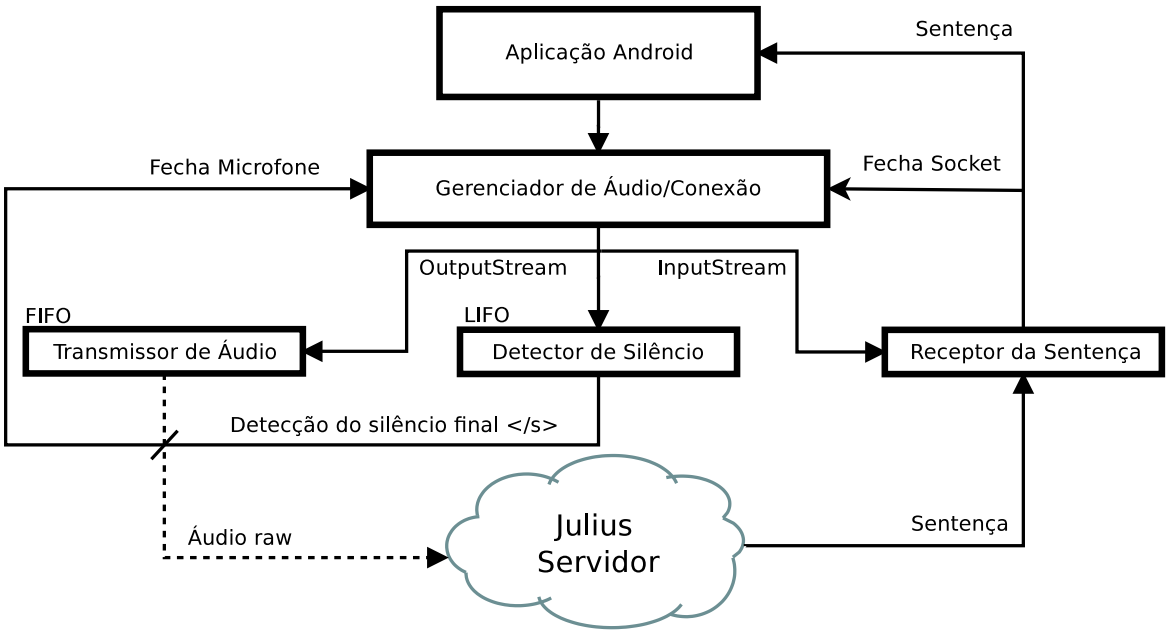

Figura 3.2: Arquitetura do cliente e sua interação com a nuvem (Batista, C. et al., 2014).

Quando um botão de *start* é pressionado no bloco de aplicação, o cliente tenta se comunicar com o servidor via *socket* através de uma *thread* encarregada de gerenciar a entrada de áudio

e conexão na rede. Se a comunicação for estabelecida, o microfone é acionado juntamente com outras três *threads*: transmissor de áudio, detector de silêncio e receptor da sentença. Cada bloco representa uma classe sendo executada no aplicativo.

O transmissor de áudio recebe *buffers* de amostras digitalizadas da fala do usuário oriundas do microfone e armazena-os em uma lista, simulando um FIFO (*first in, first out*), no qual o primeiro elemento a entrar é o primeiro a ser transmitido. As amostras vão sendo, então, enviadas ao servidor através da *output stream* do *socket* até que o usuário termine de falar.

As mesmas amostras são passadas ao detector de silêncio, o qual, simulando uma lista do tipo LIFO (*last in, first out*), analisa os últimos *buffers* a entrar na fila através do algoritmo descrito na Seção 2.3. Quando o silêncio do fim da fala é detectado, a *stream* do microfone é fechada e, consequentemente, as amostras deixam de ser enviadas ao servidor.

O receptor da sentença é responsável por receber na *input stream* do *socket* a *string* gerada pelo Julius como sentença. Quando o resultado encontra-se disponível no lado cliente, o *socket* é fechado no gerenciador de conexão e a sentença é mostrada ao usuário no *layout* da aplicação. A API `TextToSpeech` do Android foi utilizada, então, para realizar a conversão TTS da *string* que compõe a sentença.

3.1.2 Servidor: Beaglebone Black

O servidor, por ser o elemento chave na consolidação do projeto, deve ser o módulo a ser prioritariamente configurado, a fim de ser preparado para atender às devidas requisições e executar a aplicação solicitada. A instalação do sistema operacional Debian foi tomada como primeiro passo, visto que houveram muitos problemas na instalação do Ubuntu e do Ångström. As dependências a serem instaladas por `apt-get` são mostradas na Listagem 3.1.

```
alsa-tools alsa-base alsa-utils sox # Básico (para áudio)
build-essential device-tree-compiler # Básico
libasound2 libasound2-dev # Julius
apache2 libapache2-mod-fastcgi # Apache server
mysql-server libapache2-mod-auth-mysql php5-mysql # MySQL
libmysqlclient-dev # MySQL C Connector
am335x-pru-package # PRU
```

Listagem 3.1: Dependências instaladas no servidor embarcado.

É importante ressaltar que os sistemas operacionais embarcados são simplificações de sistemas mais robustos, tendo a maior parte das suas funcionalidades reduzidas para se adequar a uma plataforma de menor porte. A preparação deve ocorrer a partir dos pacotes mais básicos, enquanto os demais podem ser instalados de forma gradual, como os requeridos pelo Julius, libpruio e os necessários para a implementação do módulo servidor em C.

O *software* Julius foi configurado para funcionar em modo servidor através da opção nativa `adinnet` (do inglês *analog-digital input from network*, conversão analógico-digital com entrada pela rede). Isso permite que o Julius receba amostras de áudio via *streaming* através de uma comunicação em uma rede TCP/IP com um cliente genérico via *socket*. O código foi alterado para que o resultado gerado pelo Julius, também conhecido como sentença, seja retornado ao cliente através desse mesmo *socket*.

A construção do dicionário fonético para o PT_BR se dá por meio do conversor grafema-fonema, também chamado de `lapsg2p`, um *software* que recebe uma lista de palavras como entrada e gera suas transcrições fonéticas [76]. O dicionário fonético, mostrado na Listagem B.1 é utilizado para converter os símbolos terminais (palavras) em fonemas, os quais são os *labels* utilizados no aprendizado do modelo acústico.

Já a gramática, que é utilizada para restringir o vocabulário, define as regras de produção das palavras, de modo a gerar somente uma das sentenças listadas previamente definidas. A gramática segue o estilo da notação BNF (Backus-Naur *Form*), e é mostrada na íntegra no Apêndice B. Como apenas seis comandos básicos foram implementados, a gramática livre de contexto é composta por apenas seis sentenças possíveis: “aumentar volume”, “diminuir volume”, “canal mais”, “canal menos”, “ligar televisão” e “desligar televisão”. As regras de produção utilizadas para gerar os símbolos (sentenças produzidas pelo Julius) são definidas no arquivo de extensão `.grammar`, mostrado na Listagem B.2.

A Listagem B.3 mostra detalhes das regras de produção da gramática livre de contexto, definidas em um arquivo de extensão `.voca`. Cada símbolo não terminal (em maiúsculo, previamente definidos no arquivo *.gram*) aponta para uma palavra, juntamente com sua respectiva transcrição fonética gerada pelo dicionário. Detalhes sobre a construção da gramática livre de contexto para o Julius são descritos na página oficial [79].

3.2 Análise do Sinal Infravermelho

O receptor IR VS 18388 foi utilizado para captar os sinais infravermelhos advindos do controle remoto da TV Samsung LT22B300LBMZD. Um código em C foi escrito para fazer um *dump* dos sinais, traduzindo a informação em um vetor de tempos de atividade e inatividade do LED IR. Para fins de análise, o MATLAB foi utilizado para converter o vetor (o qual alterna seus índices pares e ímpares entre a duração do modo *burst* e a do modo *idle*) em uma onda para, posteriormente, gerar o gráfico. A Figura 3.3 mostra a forma de onda quadrada obtida após o pressionamento consecutivo de 4 botões diferentes. Nota-se que os sinais são idênticos até o final do segundo bloco *custom*, o qual é imediatamente seguido pelos blocos de comando.

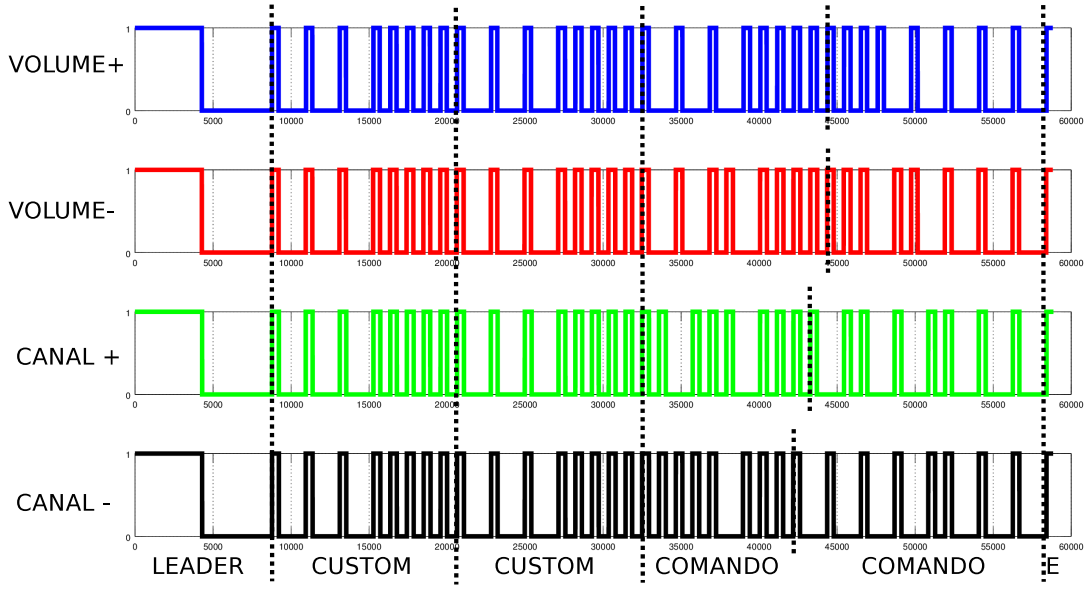

Figura 3.3: Forma de onda para quatro comandos diferentes no protocolo Samsung.

A Listagem 3.2 mostra um trecho de código que utiliza a biblioteca `libpruio` para ler, com o mínimo de latência, o pino P8_16, o qual tem conectado um sensor IR, a fim de captar eventos advindos do controle remoto. O arquivo DTS utilizado para o PinMux é mostrado no Apêndice A. A função `gettimeofday()`, definida em `sys/time.h`, permite contar os microssegundos em que o pino permaneceu com o mesmo valor. Assim, pode-se saber quanto tempo o LED IR do controle permaneceu nos modos *burst* e *idle*. Os valores seriam exibidos na tela na forma de uma matriz de duas colunas e várias linhas. A função `pruio_gpio_Value()` lê um valor binário, sendo equivalente à função `digitalRead()` do Arduino. Esta e as demais funções para configuração e acesso à PRU estão definidas em `pruio.h`.

```
#define PIN P8_16
pruIo *io = pruio_new(PRUIO_DEF_ACTIVE, 0x98, 0, 1);
struct timeval stop_low, start_low;
struct timeval stop_high, start_high;
while(1) {
    gettimeofday(&start_low, NULL); //start time counter
    while(((int) pruio_gpio_Value(io, PIN)) == 1); //burst
    gettimeofday(&stop_low, NULL); //stop time counter

    gettimeofday(&start_high, NULL); //start time counter
    while(((int) pruio_gpio_Value(io, PIN)) == 0); //idle
    gettimeofday(&stop_high, NULL); //stop time counter

    printf("%lu %lu\n",
            (stop_low.tv_usec - start_low.tv_usec),
            (stop_high.tv_usec - start_high.tv_usec));
}
```

Listagem 3.2: *Dump* de um sinal infravermelho com auxílio da `libpruio`.

3.3 Envio do Sinal Infravermelho

A informação oriunda do banco de dados é uma *string* de 34 elementos, na qual cada caractere representa um bit. Assim, dado que a PWM já teve a frequência configurada para 37,9 kHz; e sabendo que o os bits são definidos como uma transição *high-to-low*, numa sequência de estados *burst-idle* com duração de 560 μs e 1690 μs para o bit '1', e duração de 560 μs e 560 μs para o bit '0'; duas funções foram criadas para emular o padrão seguido pelo protocolo da Samsung: a função `high_pulse()`, que define o *duty cycle* da PWM para 33%; e a função `low_pulse()`, que define o *duty cycle* para 0%. Em outras palavras, `high_pulse()` e `low_pulse()` definem, respectivamente, os estados alto e baixo da forma de onda que representa o comando, e recebem como parâmetro o tempo, em milissegundos, em que a PWM deve ser mantida naquele estado. O trecho de código abaixo mostra, de forma genérica, o conteúdo de ambas, as quais chamam a função `pruio_pwm_setValue()`. A diferença, além do tempo de *delay*, é o valor de *duty cycle* definido. O LED IR está conectado ao pino P8_13, cuja configuração através do arquivo DTS é mostrada no Apêndice A.

```
pruio_pwm_setValue(io, P8_13, 37900, duty);
usleep(delay-100);
```

A Listagem 3.3 mostra a emulação do protocolo da Samsung para o comando “aumentar volume” recuperado do banco de dados através da função `get_element()`.

```
MYSQL *con = mysql_init(NULL);
mysql_real_connect(con, "localhost",
            "root", "beagle1234", NULL, 0, NULL, 0);
mysql_select_db(con, "ir_db");
char *bitstream = get_element("vol_mais", "samsung", con);
high_pulse(4500); // Leader: 4.5 ms HIGH
low_pulse(4500);// Leader: 4.5 ms LOW
for(int i=1; i<34; i++) {
    high_pulse(560);
    if(bitstream[i] == '1') {
        low_pulse(1690);
    } else {
        low_pulse(560);
    }
}
low_pulse(2000); // encerra
```

Listagem 3.3: Emulação do protocolo da Samsung.

3.4 Banco de Dados MySQL

Como o sistema foi projetado para dar suporte ao controle de diversos aparelhos eletrônicos, a adoção de um banco de dados foi vista como solução para facilitar o acesso a uma eventual variedade de dispositivos. Inicialmente, apenas a entidade “Televisão” foi criada, na qual uma tabela intitulada `TV` contém atributos como a marca do aparelho e os comandos a serem transmitidos, como mostrado na Tabela 3.1. Com isso, quaisquer campos podem ser satisfatoriamente acessados através de um simples código SQL (como o mostrado abaixo).

```
SELECT vol_mais FROM TV WHERE marca='samsung'
SELECT ch_menos FROM TV WHERE marca='philips'
```

Assim, os bits de referência podem ser recuperados e codificados de acordo com o protocolo para que, finalmente, a informação possa ser transmitida para o aparelho através do LED IR. A biblioteca `MySQL Connector/C` (ou `libmysqlclient`) [80], permite que o banco de dados seja acessado por códigos na linguagem C. As *queries* são realizadas pela passagem dos comando SQL como *strings* para funções pré-definidas.

Tabela 3.1: Exemplo hipotético de uma tabela intitulada 'TV' no banco de dados.

Marca	vol_mais	vol_menos	ch_mais	ch_menos	on_off
toshiba	110111011	111110110	111000010	100010111	101011111
sony	111110110	101011111	100011111	110111011	111000000
samsung	100110111	111000010	111010000	111011110	111011100
philips	100011111	110111011	111000000	111011010	111010111
panasonic	110101011	101010110	111010010	101110111	101011101

As funções capazes de configurar e operar o banco de dados para a aplicação apresentada são mostradas na Tabela 3.2. O trecho de código em C utilizado para configurar o banco de dados, bem como o seu equivalente em SQL, é mostrado no Apêndice C.

Tabela 3.2: Funções para acesso ao banco de dados pela linguagem C.

Função	MySQL Equivalente	Descrição
`get_element`	SELECT FROM	Recupera um valor de um campo da tabela
`set_element`	UPDATE SET	Insere um valor em um campo da tabela
`insert_table`	INSERT INTO	Insere uma nova linha (marca) na tabela
`grant_access`	GRANT ALL	Concede privilégios a um usuário específico
`create_db`	CREATE DATABASE	Cria um novo banco de dados
`drop_db`	DROP DATABASE	Exclui um banco de dados
`create_table`	CREATE TABLE	Cria uma nova tabela (aparelho) no banco
`drop_table`	DROP TABLE	Exclui uma tabela do banco de dados

Capítulo 4

Ambiente de Testes e Resultados

Os testes foram realizados com pessoas de diferentes faixas etárias e graus de instrução, que se declararam portadoras de necessidades motoras e/ou visuais, ou mesmo sem deficiência alguma. Vale ressaltar que deficiência visual não é sinônimo de cegueira, assim como necessidade motora dos membros superiores (MMSS) não significa amputação ou paralisia total, conforme mostrado previamente nas descrições da Tabela 1.1.

Inicialmente, o texto contendo a definição de deficiência, previamente mostrado na Seção 1.2 parágrafo 1, foi apresentado aos participantes, os quais o leram de forma individual. Os participantes foram, então, instruídos e submetidos à realização de quatro tarefas simples: ligar a TV, mudar o canal, ajustar o volume e, finalmente, desligar a TV. O comando a ser dado ao sistema de reconhecimento de voz foi listado ao lado de cada ação do roteiro, e o processo de funcionamento do sistema foi explicado verbalmente. Além disso, o próprio autor do trabalho demonstrou a execução das tarefas do roteiro como forma de exemplo prático do funcionamento do sistema, enquanto os voluntários assistiam atentamente. O roteiro e as respectivas sentenças de entrada para o sistema ASR são mostrados na Tabela 4.1.

Tabela 4.1: Roteiro de tarefas a serem completadas pelo usuário.

Tarefa	Comando de voz (sentença)
Ligar televisão	“ligar televisão”
Mudar para o próximo canal	“canal mais”
Ajustar o volume	“volume mais” ou “volume menos”
Desligar televisão	“desligar televisão”

O ambiente de testes não foi controlado, o que significa que havia ruído de fundo presente, o que poderia ter sido prejudicial tanto para a entrada de áudio, referente ao reconhecimento da fala pelo sistema, quanto para a saída de áudio do TTS, referente ao entendimento da voz sintética pelos participantes. Os participantes foram instruídos a desistir após a terceira tentativa de cada comando, caso o sistema não se comportasse conforme desejado.

O questionário SUS foi aplicado para verificar a utilidade do protótipo, bem como a necessidade de um sistema alternativo de controle de aparelhos eletrônicos que seja acessível a pessoas em geral, especialmente às PCD. No total, 10 pessoas foram entrevistadas, sendo 4 do sexo masculino e 6 do sexo feminino, com idades entre 23 e 62 anos.

Perguntas simples foram adicionadas ao questionário de usabilidade da Seção 2.7, de modo que um perfil de usuários fosse mantido para auxiliar na interpretação dos resultados dos testes. Os voluntários foram questionados sobre i) faixa etária; ii) grau de escolaridade; iii) tipo de deficiência; iv) e se possuíam familiaridade com o uso de *smartphones*. O perfil demográfico dos voluntários é mostrado na Figura 4.1. As possíveis respostas para cada questão são mostradas em detalhes a seguir.

i) Faixa etária:
   a) Até 30 anos;
   b) De 31 à 49 anos;
   c) 50 anos ou mais.

ii) Grau de escolaridade:
   a) Fundamental (menor que, incompleto ou completo);
   b) Médio (incompleto ou completo);
   c) Superior (incompleto cursando, completo, ou com quaisquer pós-graduações).

iii) Tipo de deficiência:
   a) Visual;
   b) Membros superiores (MMSS);
   c) Nenhuma.

iv) Familiaridade com o uso de *smartphones*:
   a) Não ou muito pouca;
   b) Relativamente;
   c) Sim.

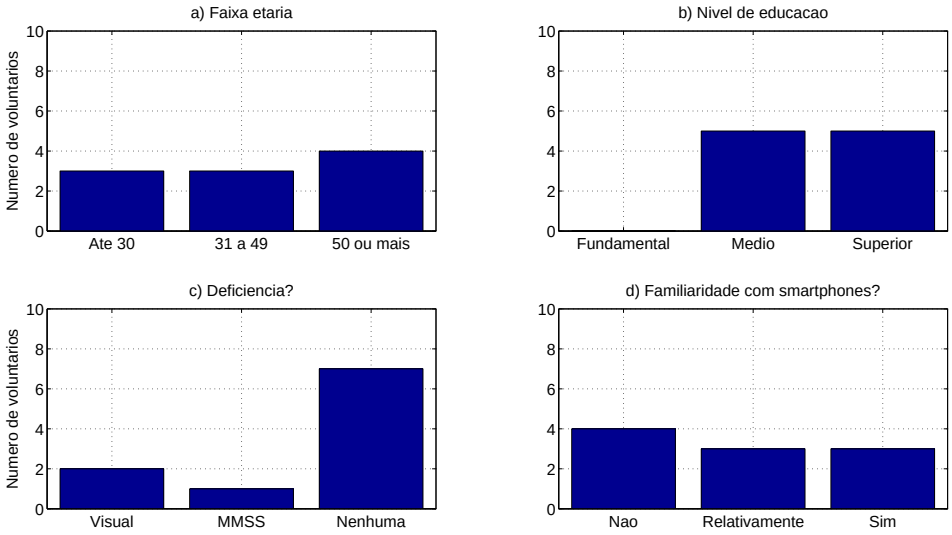

Figura 4.1: Perfil demográfico dos participantes.

A análise dos resultados finais do SUS mostrou que os dados coletados no perfil demográfico influenciaram fortemente na usabilidade do sistema. Os grupos de pessoas que obtiveram um escore acima da média de 68 pontos foram: a) pessoas de até 49 anos; b) pessoas com nível de educação superior; c) pessoas que declararam não ter deficiência alguma; d) e pessoas que têm relativa ou muita familiaridade com *smartphones*. A Figura 4.2 mostra as médias de escore por grupo demográfico. Os resultados serão discutidos com mais detalhes a seguir.

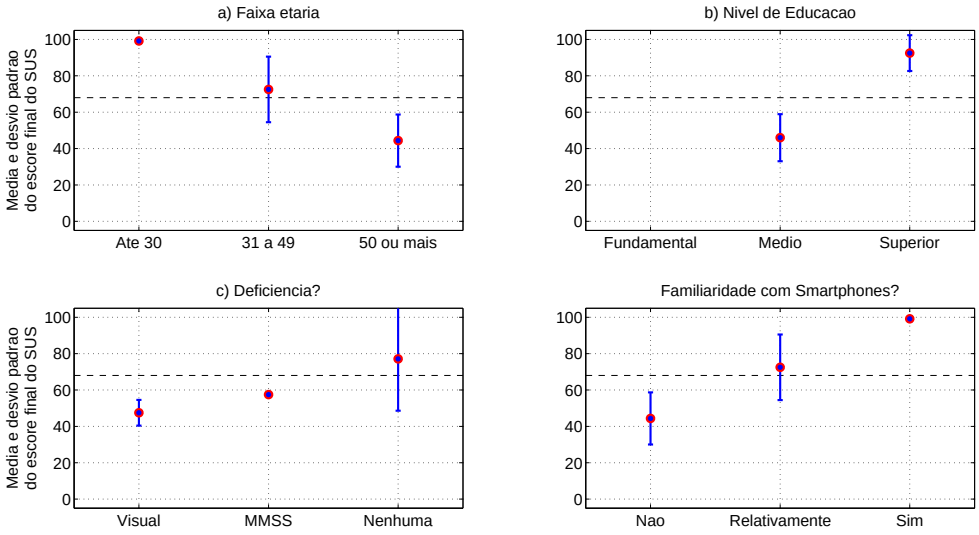

Figura 4.2: Média e desvio padrão dos escores finais do SUS por perfil demográfico.

a) Faixa etária: para as pessoas mais velhas, com mais de 50 anos, a média do escore final foi de, aproximadamente, 45 pontos. Para os que têm idades entre o intervalo de 31 a 49 anos, o resultado foi de aproximadamente 70 pontos. Já para os mais jovens, de até 30 anos, o resultado foi incrivelmente alto, alcançando o valor máximo de 100 pontos.

b) Nível de educação: as pessoas que estudaram até o nível médio obtiveram uma média de 44 pontos, enquanto as que têm nível superior alcançaram a média de 92 pontos, aproximadamente. Nenhuma pessoa com nível de escolaridade fundamental ou inferior foi entrevistada.

c) Deficiência: as pessoas que declararam ter algum tipo de dificuldade visual que se enquadrasse no conceito de deficiências a elas apresentado alcançaram uma média de 44 pontos no escore final do SUS, enquanto o(a) voluntário(a) que declarou ter alguma deficiência motora dos membros superiores obteve um escore de 59 pontos. As pessoas declaradas sem deficiência alguma ficaram acima da média, com um escore final de aproximadamente 79 pontos.

d) Familiaridade com *smartphones*: Pessoas com pouca ou nenhuma familiaridade com *smartphones* ficaram abaixo da média, alcançando um escore de, aproximadamente, 43 pontos. Já as que declararam ter relativa e muita familiaridade, obtiveram, aproximadamente, 70 e 99 pontos na média do SUS, respectivamente.

A primeira impressão que se tem, a partir da análise dos resultados, é de que as pessoas que se julgam deficientes consideram o sistema pouco útil, já que o escore do SUS obtido por essas pessoas aponta uma baixa usabilidade. No entanto, as pessoas que obtiveram o escore final mais alto são as mesmas pessoas que declararam não possuir deficiência alguma, são mais jovens e têm nível de educação mais elevado, possuindo também grande familiaridade com *smartphones*.

Apesar de ser composta de pessoas com diferentes níveis de instrução, diferentes graus familiaridade com *smartphones*, e dificuldades físicas e faixa etárias variadas, a amostra populacional de voluntários, infelizmente, não foi composta somente por PCD, para o qual o sistema é voltado. No entanto, pode-se nitidamente observar grupos específicos de pessoas para os quais o sistema foi considerado muito útil e de alta qualidade, considerando-se o resultado da escala de usabilidade.

Capítulo 5

Considerações Finais

O presente trabalho contribui com um protótipo de controle remoto universal direcionado especialmente para pessoas com necessidades motoras e visuais, com objetivo de tornar o controle de equipamentos eletrônicos do ambiente doméstico de fato acessível.

Os produtos revisados apresentam funcionalidade semelhante a do sistema proposto e geralmente possuem um número considerável de equipamentos cadastrados em seus respectivos bancos de dados. Já os aplicativos que não necessitam de *hardware* adicional (como *dongles* ou *gateways*, por exemplo) requerem unicamente que o usuário possua um *smartphone*. Portanto, conclui-se que o projeto apresentado ainda é inviável por conta da reduzida abrangência no controle de equipamentos e do custo bastante elevado do computador embarcado.

No entanto, as funcionalidades de síntese e reconhecimento de voz tornam o produto muito mais interessante por conta da comodidade e da acessibilidade. Nenhuma das aplicações pesquisadas, que existem atualmente no mercado, apresenta características de Tecnologia Assistiva, tampouco possui o propósito voltado para os portadores de necessidades especiais.

Segundo a Nielsen Ibope [81], o número de brasileiros que já possui um *smartphone* é de aproximadamente 72 milhões. Porém, de acordo a mesma com a pesquisa, a grande maioria das pessoas utiliza *smartphones* apenas para acessar as redes sociais, comunicar-se via texto, compartilhar e reproduzir mídia, jogar e, obviamente, fazer chamadas telefônicas. Algumas das pessoas inclusive alegam dificuldade em utilizar aplicativos novos, dando a justificativa de "baixa usabilidade".

Em outras palavras, por considerarem "difícil de usar", poucas pessoas costumam utilizar todas as funcionalidades de um *smartphone*, o que torna os aplicativos para controle remoto — apesar de muito úteis —, bem como outros aplicativos interessantes, subutilizados.

O escore das pessoas que têm experiência com *smartphones* alcançou uma média de 99 pontos na escala do SUS. Pouco auxílio foi dado durante a execução do roteiro e o tempo para completá-lo foi relativamente baixo, apesar de ter sido, obviamente, maior do que o tempo levado para executarem as tarefas utilizando o próprio controle remoto do aparelho.

Já o *feedback* das pessoas que não têm familiaridade com *smartphones* sugere que a usabilidade do sistema deve ser melhorada, de modo que pessoas que não costumam ter contato com equipamentos eletrônicos modernos também possam usufruir das vantagens oferecidas pela TA. O *trade-off* entre os items 1 e 4 do questionário apresentado na Seção 2.7 mostrou que essas pessoas realmente têm interesse em utilizar produtos desse tipo, mas gostariam que a interface fosse mais simples ou, em outras palavras, que o sistema fosse mais fácil de ser utilizado de forma independente. Esse grupo de voluntários demonstrou uma real necessidade de auxílio durante os testes, demorando muito mais tempo, em relação ao grupo mencionado previamente, para completar o roteiro. A média do escore dessas pessoas foi de 45 pontos na escala do SUS.

É importante ressaltar que "idade" e "familiaridade com tecnologia" são fatores inversamente correlacionados, pois, normalmente, quanto mais avançada é a idade de uma pessoa, menos acostumada com produtos eletrônicos modernos ela é. Além disso, o fator do nível de escolaridade também pode ter um impacto significativamente negativo na usabilidade, já que pessoas com maior grau de instrução possuem, geralmente, mais facilidade de aprender e adaptar-se a novas tecnologias. No entanto, o grau de familiaridade com *smartphones* foi considerado, durante os testes, como o principal fator para a baixa usabilidade do sistema, já que baixa escolaridade e idade avançada não significam impedimento algum na capacidade de aprender a utilizar *smartphones* tão bem quanto pessoas mais jovens e instruídas.

Espera-se que, nas próximas versões, o sistema fique mais barato, atenda a um maior número de pessoas de diversos grupos demográficos e, por fim, envolva uma gama maior de deficiências assistidas ao ganhar uma característica multimodal. Como o barateamento do *hardware* vem sendo uma tendência, o sistema deve tornar-se economicamente viável dentro de um futuro próximo. Computadores "do tamanho de um cartão de crédito" encontram-se cada vez mais frequentes no mercado, como é o caso dos novos Raspberry Pi Zero e C.H.I.P., que custam US$ 5.00 e US$ 9.00 dólares, respectivamente (sem impostos).

No entanto, é possível que as *smart* TVs também se tornem mais populares e facilmente adquiridas. A comunicação por rede local via Wi-Fi, por conta da praticidade e conforto, poria fim na tecnologia infravermelha que atualmente domina o controle remoto dos equipamen-

tos eletrônicos domésticos. Porém, de acordo com as estatísticas consultadas, é impossível prever por quanto tempo a população brasileira continuará a usar majoritariamente aparelhos eletrônicos que utilizam IR como forma de controle.

5.1 Dificuldades e Soluções

Durante a construção no projeto, diversos problemas foram enfrentados no que diz respeito à instalação, configuração e implementação dos diversos módulos individuais que compõem o sistema. Os mais relevantes são enumerados a seguir.

1. O Ångström e o Ubuntu foram os principais candidatos à sistema operacional base no protótipo: o primeiro, por ser o sistema operacional de fábrica da Beaglebone Black e o segundo, pela similaridade com o sistema `apt-get` da distribuição *desktop*. Entretanto, devido ao grande espaço ocupado por ambos na memória *flash* (eMMC) do microcomputador (1.4 GB, cerca de 80% dos 2 GB disponíveis) além de outros problemas como falhas na conexão com a Internet, nomenclatura incompatível e acesso restrito ao repositório de pacotes, a solução foi a instalação do Debian 7.8 Wheezy, o qual ocupou apenas 500 MB da *flash*.

2. A comunicação entre o sistema que emula o controle remoto e a TV era estabelecida através de um Arduino UNO, já que os GPIOs da Beaglebone Black mostraram-se lentos demais e não se tinha noção alguma de como utilizar a PRU através de códigos Assembly. Felizmente, a documentação e exemplos de códigos sobre a `libpruio` foram encontrados livremente na Internet.

3. O módulo de síntese de voz era executado na Beaglebone Black com auxílio de uma API em C criada sobre o *software* eSpeak. Entretanto, além da dificuldade em encontrar um alto-falante USB para reproduzir o áudio sintético, a qualidade das vozes para PT_BR era, perceptualmente, muito baixa. A solução encontrada foi movê-lo do servidor para o cliente, passando a utilizar o sintetizador de voz na aplicação Android.

4. Antes de se utilizar a API `TextToSpeech`, nativa do Android, no aplicativo do *smartphone*, tentou-se utilizar uma API desenvolvida em [82], baseada no sintetizador HTS, um *software* livre que gera voz artificiais para sistemas embarcados e *desktops*. No entanto, a dificuldade com a instalação do pacote NDK na IDE do Android Studio, dentre

outros problemas na configuração da API para o HTS, impossibilitaram a agregação do *software* à aplicação cliente.

5. Para fazer *streaming* do áudio do cliente Android para o Julius no servidor, deve-se primeiramente enviar o tamanho do *buffer*, seguido do *buffer* de dados propriamente dito. Além disso, o inteiro que representa o tamanho do *buffer* deve ser enviado também em um *buffer*, na qual os 4 bytes têm suas ordens invertidas de *little-endian* para *big-endian*. Uma análise minuciosa do cliente do próprio Julius foi feita para que o problema do envio do áudio pelo *smartphone* fosse resolvido.

5.2 Trabalhos Futuros

Como o sistema proposto ainda encontra-se inviavelmente caro, já que a revisão B da Beaglebone Black custa em média US$ 45.00, uma das prioridades é reduzir o custo do projeto. Nesse sentido, o novo microcomputador da Next Thing Co. chamado C.H.I.P.[1], que custa apenas US$ 9.00[2], parece a melhor alternativa para substituir a Beaglebone Black na função de *gateway* centralizado de controle remoto.

Atualmente, o sistema dá suporte apenas aos módulos de reconhecimento e síntese de voz (ASR e TTS, respectivamente), os quais auxiliam na minimização das dificuldades apenas dos portadores de necessidades visuais e motoras dos membros superiores (MMSS). No entanto, pessoas que por algum motivo não podem utilizar os controles remotos tradicionais e nem usufruir dos benefícios oferecidos pelo ASR também têm o direito de controlar os equipamentos eletrônicos de forma independente, como qualquer outra pessoa. Nesse sentido, a ideia seria adicionar a funcionalidade de reconhecimento ativo de gestos (AGR, do inglês *active gesture recognition*) [83], para capturar movimentos do braço e da cabeça, a fim de também utilizá-los como entradas alternativas para o sistema de controle.

Os módulos de rastreamento e reconhecimento de movimentos dos braços e da cabeça seriam úteis para pessoas com deficiência na fala, por exemplo, que não podem utilizar o sistema ASR; ou pessoas com amputação ou limitações severas no movimento das mãos e dos dedos, mas com movimento da cabeça e/ou dos braços preservados, impossibilitadas de utilizar o controle físico convencional, pressionar botões ou segurar o *smartphone*.

[1] `https://getchip.com/pages/chip`

[2] Valor líquido, sem considerar frete, IOF, ICMS e demais impostos.

Uma *webcam* de baixo custo faria a captura das imagens processadas pelo sistema AGR, o qual, inicialmente, seria baseado na biblioteca de computação visual OpenCV [84]. Redes neurais do tipo teoria de ressonância adaptativa (ART, do inglês *adaptive resonance theory*) [85] também devem sem investigadas, já que bons resultados foram obtidos com esse tipo de algoritmo para reconhecimento de padrões biométricos [86].

Atualmente, um *smartphone* é necessariamente requerido para funcionar como acionador e entrada do sistema. Espera-se que, nas próximas versões, esse dispositivo seja utilizado apenas como opção alternativa secundária de controle, movendo o microfone, o alto-falante e o sistema de síntese de voz para o microcomputador embarcado. Um acionador externo seria construído na forma de um botão físico comum ou na forma de um pedal, este último para aqueles que possuem alguma dificuldade em apertar botões e que, obviamente, possuem os movimentos dos membros inferiores (MMII) preservados. O acionador comunicar-se-ia com a plataforma embarcada através de alguma tecnologia de comunicação sem fio, como rádio-frequência sub-GHz, Bluetooth, dentre outras ainda a serem investigadas.

A Tabela 5.1 mostra as atuais e as possíveis futuras funcionalidades do projeto, bem como as deficiências minimizadas, as habilidades físicas mínimas requeridas para a utilização e as desvantagens de cada módulo relacionadas a possíveis deficiências dos usuários.

Como o sistema atual dá suporte a apenas uma única TV específica, espera-se que haja uma expansão para vários equipamentos, o que tornaria o microcomputador um controle universal centralizado no ambiente doméstico. Isso acarretaria um número maior e mais complexo de tabelas no banco de dados, contendo os diversos aparelhos e seus comandos. Além disso, o MySQL parece ser um sistema de gerenciamento robusto demais para a aplicação. Portanto, a substituição pelo conjunto de bibliotecas SQLite ou por banco de dados não-relacionais, como o Cassandra e o MongoDB, por exemplo, deve ser investigada.

Uma padronização do envio e análise de sinais infravermelhos deve ser pesquisada. A biblioteca LIRC (*Linux Infrared remote control*) [87], por exemplo, possui um amplo banco de dados e exemplos de códigos que permitem enviar e decodificar sinais infravermelhos de muitos controles remotos comumente usados. Outra possibilidade seria adaptar para o *gateway* bibliotecas já conhecidas para microcontroladores, como a IRremote Arduino [88], por exemplo.

Uma página web acessível em rede local para fins de configuração dos atuais e eventuais futuros elementos do sistema deve ser criada. Atualmente, a configuração do sistema ASR e do banco de dados MySQL é feita manualmente através da linguagem SQL, ou por meio dos

Tabela 5.1: Futuras funcionalidades do sistema e suas relações com algumas deficiências.

Funcionalidade (módulo)	Deficiência minimizada	Habilidade preservada	Deficiência não minimizada
ASR	Visual e motora MMSS (mãos).	Fala.	Mudez e afins.
TTS	Visual.	Audição.	Surdez e afins.
AGR (braços)	Auditiva, na fala e motora MMSS (mãos/dedos).	Motora MMSS.	Paralisia superior completa e deficiência cognitiva severa.
AGR (cabeça)	Auditiva, na fala e motora MMSS (braços).	Cabeça.	Paralisia superior completa e deficiência cognitiva severa.
Acionador externo (botão)	Visual, auditiva, na fala e motora MMII.	Motora MMSS.	Deficiência cognitiva severa, tetraplegia e múltiplas deficiências.
Acionador externo (pedal)	Visual, auditiva, na fala e motora MMSS.	Motora MMII.	Deficiência cognitiva severa, tetraplegia e múltiplas deficiências.

códigos implementados na linguagem C exclusivamente para a aplicação. É preciso, portanto, centralizar a configuração de dispositivos futuros, como câmera e alto-falante, bem como a dos parâmetros dos sistemas ASR, TTS e AGR.

Referências Bibliográficas

[1] Paul Taylor, *Text-To-Speech Synthesis*, Cambridge University Press, 2009.

[2] X. Huang, A. Acero, and H. Hon, *Spoken Language Processing*, Prentice-Hall, 2001.

[3] Comitê de Ajudas Técnicas, *Tecnologia Assistiva*, Subsecretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência., Brasília, Brasil, 2009.

[4] Gert J. Gelderblom and Luc P. de Witte, "The assessment of assistive technology: Outcomes, effects and costs," 2002, vol. 14, pp. 91–94, IOS Press.

[5] The United Nations, "Convention on the rights of persons with disabilities," *Treaty Series*, vol. 2515, December 2006, `http://www.un.org/disabilities/convention/conventionfull.shtml`.

[6] "Eclipse: The Eclipse Foundation open source community website," Visitado em maio, 2013, `https://eclipse.org/`.

[7] "Android Studio," Visitado em março, 2015, `https://developer.android.com/studio/index.html`.

[8] "The Julius decoder," Visitado em maio, 2015, `julius.sourceforge.jp/en/`.

[9] "libpruio: Input/Output Driver for Digital/Analog Lines on Beagleboard Hardware," Visitado em Outubro, 2015, `http://users.freebasic-portal.de/`.

[10] "FalaBrasil: Tecnologias de Voz para o Português Brasileiro," Visitado em Julho, 2015, `http://www.laps.ufpa.br/falabrasil/`.

[11] United States Government, "Americans with disabilities act of 1990," July 1990, Public Law 101-336, 104 Stat. 327.

[12] "Assistiva: Categorias de Tecnologia Assistiva," Visitado em setembro, 2016, http://www.assistiva.com.br/tassistiva.html#categorias.

[13] "Disabled World: Disability News and Information," Visitado em setembro, 2016, http://www.disabled-world.com/disability.

[14] "Censo Demográfico ," Visitado em Abril 2015, www.ibge.gov.br/home/estatistica/populacao/censo2010/.

[15] L. C. P. Costa, I. K. Ficheman, A. G. D. Correa, R. D. Lopes, and M. K. Zuffo, "Accessibility in digital television: Designing remote controls," in *2012 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2012, pp. 676–677.

[16] I. Wechsung and A. B. Naumann, "Evaluating a multimodal remote control: The interplay between user experience and usability," in *Quality of Multimedia Experience, 2009. QoMEx 2009. International Workshop on*, July 2009, pp. 19–22.

[17] M. S. Sefat, A. A. M. Khan, and M. Shahjahan, "Implementation of vision based intelligent home automation and security system," in *Informatics, Electronics Vision (ICIEV), 2014 International Conference on*, May 2014, pp. 1–6.

[18] S. Barrena, L. Klotz, V. Landes, A. Page, and Y. Sun, "Designing android applications with both online and offline voice control of household devices," in *2012 38th Annual Northeast Bioengineering Conference (NEBEC)*, March 2012, pp. 319–320.

[19] V. Govardanam, T. N. V. Babu, and N. S. H. Kavin, "Automated read-write kit for blind using hidden markov model and optical character recognition," in *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Oct 2015, pp. 828–831.

[20] Ahsan K., Iqbal S., Nadeem A., and Sarim M., "Unicon remote control model – a mobile system for assistive technology," *Research Journal of Recent Sciences*, vol. 3, no. 4, pp. 95–102, April 2014.

[21] "Peel Smart Remote: Free smart TV Remote," Visitado em setembro, 2016, https://www.peel.com/.

[22] "X10 Commander," Visitado em setembro, 2016, http://melloware.com/x10commander/.

[23] "Control4: Home Automation and Smart Home Systems," Visitado em setembro, 2016, http://www.control4.com/.

[24] "iRule: The Ultimate Remote Control for iOS & Android Devices," Visitado em setembro, 2016, http://getirule.com/.

[25] "Logitech Harmony Smart Control," Visitado em abril, 2015, http://www.logitech.com/en-us/product/harmony-smart-control.

[26] "IRdroid: Universal Remote for Android," Visitado em abril, 2015, http://www.irdroid.com/.

[27] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*, PTR Prentice Hall, Englewood Cliffs, N.J., 1993.

[28] A. M. da Cunha and L. Velho, "Métodos probabilísticos para reconhecimento de voz," Tech. Rep., Laboratório VISGRAF - Instituto de Matemática Pura e Aplicada, 2003.

[29] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. on ASSP*, vol. 26, no. 1, pp. 43–49, 1978.

[30] F. Jelinek, *Statistical methods for speech recognition*, MIT Press, 1997.

[31] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–86, Feb. 1989.

[32] H. Juang and R. Rabiner, "Hidden Markov models for speech recognition," *Technometrics*, vol. 33, no. 3, pp. 251–272, 1991.

[33] P. Woodland and D. Povey, "Large scale discriminative training of hidden Markov models for speech recognition," *Computer Speech and Language*, vol. 16, pp. 25–47, 2002.

[34] Akinobu Lee, Tatsuya Kawahara, and Kiyoshiro Shikano, "Gaussian mixture selection using context-independent HMM," *In Proceedings IEEE-ICASSP*, 2001.

[35] J. Picone, "Signal modeling techniques in speech recognition," *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215–47, Sep. 1993.

[36] J.-C. Junqua and J.-P. Haton, *Robustness in Automatic Speech Recognition*, Kluwer, 1996.

[37] S. Davis and P. Merlmestein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. on ASSP*, vol. 28, pp. 357–366, Aug. 1980.

[38] N. Deshmukh, A. Ganapathiraju, and J. Picone, "Hierarchical search for large-vocabulary conversational speech recognition," *IEEE Signal Processing Magazine*, pp. 84–107, 1999.

[39] N. Jevtić, A. Klautau, and A. Orlitsky, "Estimated rank pruning and Java-based speech recognition," in *Automatic Speech Recognition and Understanding Workshop*, 2001.

[40] P. Ladefoged, *A Course in Phonetics*, Harcourt Brace, 4 edition, 2001.

[41] Thierry Dutoit, *An Introduction to Text-To-Speech Synthesis*, Kluwer, 2001.

[42] Keiichi Tokuda, Heiga Zen, and Alan Black, "An HMM-based speech synthesis system applied to English," *Proceedings of IEEE Workshop on Speech Synthesis*, pp. 227–230, 2002.

[43] Keiichi Tokuda, *An HMM-Based Approach to Flexible Speech Synthesis*, vol. 4274/2006 of *Lecture Notes in Computer Science*, Springer Berlin, November 2006.

[44] Kirill Saknov, Ekaterina Verteletskaya, and Boris Simak, "Approach for energy-based voice detector with adaptive scaling factor," pp. 394–399, November 2009.

[45] R. Venkatesha Prasad, A. Sangwan, H. S. Jamadagni, M. C. Chiranth, R. Sah, and V. Gaurav, "Comparison of voice activity detection algorithms for voip," in *Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on*, 2002, pp. 530–535.

[46] Koichi Yamamoto, Firas Jabloun, Klaus Reinhard, and Akinori Kawamura, "Robust endpoint detection for speech recognition based on discriminative feature extraction," in *IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP'06, Toulouse, France*, May 2006, pp. 805–808.

[47] Philippe Renevey and Andrzej Drygajlo, "Entropy based voice activity detection in very noisy conditions," in *EUROSPEECH 2001 Scandinavia, 7th European Conference on*

Speech Communication and Technology, 2nd INTERSPEECH Event, Aalborg, Denmark, Paul Dalsgaard, Børge Lindberg, Henrik Benner, and Zheng-Hua Tan, Eds. September 2001, pp. 1887–1890, ISCA.

[48] "Arduino: PWM," Visitado em setembro, 2016, https://www.arduino.cc/en/Tutorial/PWM.

[49] "Como funciona o controle remoto?," Visitado em abril, 2015, http://mundoestranho.abril.com.br/.

[50] "Como funcionam os controles remotos," Visitado em abril, 2015, http://tecnologia.hsw.uol.com.br/.

[51] Samsung Electronics, *S3F80KB Microcontroller for IR Remote Controller*, 2008.

[52] "Raspberry Pi or Beaglebone Black," Visitado em abril, 2015, http://michaelhleonard.com/.

[53] "Arduino vs. Raspberry Pi vs Beaglebone," Visitado em abril, 2015, http://randomnerdtutorials.com/.

[54] "Arduino Uno vs. Beaglebone vs. Raspberry Pi," Visitado em abril, 2015, http://makezine.com/.

[55] "Everything you need to know about Beaglebone Black," Visitado em novembro, 2015, http://www.tested.com/.

[56] "Raspberry Pi 2 Model B," Visitado em novembro, 2015, https://www.raspberrypi.org/.

[57] "Arduino UNO," Visitado em novembro, 2015, https://www.arduino.cc/.

[58] "Beaglebone Black Wiki," Visitado em novembro, 2015, http://elinux.org/Beagleboard.

[59] Gerald Coley and Robert P J Day, *Beaglebone Black System Reference Manual*, Beagleboard.org, January 2014, Revision B.

[60] Adafruit Industries, *Raspberry Pi 2, Model B*, http://www.adafruit.com/pdfs/raspberrypi2modelb.pdf.

[61] “Raspberry Pi Comparison Chart,” Visitado em novembro, 2015, http://www.makershed.com/.

[62] “Beaglebone Black – Rev B,” Visitado em novembro, 2015, https://www.adafruit.com/product/1278.

[63] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

[64] “ROS: The Robot Operating System,” Visitado em novembro, 2015, http://www.ros.org/.

[65] “OpenROV: Underwater Exploration Robots,” Visitado em novembro, 2015, http://www.openrov.com/.

[66] Thomas Petazzoni, *Device Tree for Dummies*, 2014.

[67] “Device Tree,” Visitado em dezembro, 2015, http://elinux.org/Device_Tree.

[68] Texas Instruments, *AM335x PRU-ICSS Reference Guide*, May 2012, Revised on June 2013.

[69] “TI AM33xx PRUSS v2,” Visitado em novembro, 2015, http://elinux.org/Ti_AM33XX_PRUSSv2.

[70] Texas Instruments and Sitara ARM Processors, “Building Blocks for PRU Development,” Tech. Rep., Boot Camp Training Series, October 2014, PRU Hardware Overview.

[71] “Using Device Trees to Configure PRU IO Pins,” Visitado em abril, 2015, http://www.ofitselfso.com/.

[72] John Brooke, “SUS: A quick and dirty usability scale,” 1996, Redhatch Consulting Ltd.

[73] James R. Lewis and Jeff Sauro, “The factor structure of the system usability scale,” in *Proceedings of the 1st International Conference on Human Centered Design: Held As Part of HCI International 2009*, Berlin, Heidelberg, 2009, HCD 09, pp. 94–103, Springer-Verlag.

[74] "Android Developers – TextToSpeech," Visitado em setembro, 2016, https://developer.android.com/reference/android/speech/tts/TextToSpeech.html.

[75] "Coruja (LaPSAPI)," Visitado em junho, 2013, code.google.com/p/lapsapi/.

[76] Ana Siravenha, Nelson Neto, Valquíria Macedo, and Aldebaro Klautau, "Uso de regras fonológicas com determinação de vogal tônica para conversão grafema-fone em Português Brasileiro," *7th International Information and Telecommunication Technologies Symposium*, 2008.

[77] "Android ADT Plugin," https://developer.android.com/studio/tools/sdk/eclipse-adt.html.

[78] C. Batista, T. Coelho, B. Haick, N. Neto, and A. Klautau, "LaPS CSR: A free distributed cloud speech recognition system," in *Fiatal Müszakiak Tudományos Ülésszaka (FMTU)*, March 2014, pp. 421–424.

[79] "The recognition grammar format of Julius," Visitado em maio, 2015, julius.sourceforge.jp/en_index.php?q=en_grammar.html.

[80] "MySQL C API Implementations," Visitado em dezembro, 2015, http://dev.mysql.com/doc/refman/5.7/en/c-api-implementations.html.

[81] "Nielsen: Brasileiros com Internet no smartphone já são mais de 70 milhões," Visitado em setembro, 2016, http://www.nielsen.com/br/pt/press-room/2015/Brasileiros-com-internet-no-smartphone-ja-sao-mais-de-70-milhoes.html.

[82] Marcelo Takashi Hashimoto, "Desenvolvimento de recursos para síntese de voz em português brasileiro para desktop e sistemas embarcados," Universidade Federal do Pará, Instituto de Tecnologia, 2016, Trabalho de Conclusão de Curso.

[83] Trevor Darrell and Alex Pentland, "Active gesture recognition using learned visual attention," in *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, November 27-30, 1995*, 1995, pp. 858–864.

[84] “OpenCV: Open Source Computer Vision Library,” Visitado em setembro, 2016, http://opencv.org/.

[85] David Kriesel, *A Brief Introduction to Neural Networks*, 2007.

[86] J. A. R. Quintana, M. I. C. Murgia, and Jose F. C. Hinojos, “Artificial neural image processing applications: A survey,” *International Association of Engineers (IAENG), Engineering Letters*, vol. 20, no. 1, pp. 95–102, February 2012.

[87] “LIRC: Linux Infrared Remote Control,” Visitado em dezembro, 2015, http://www.lirc.org/.

[88] “IRremote Arduino Library,” Visitado em setembro, 2016, https://github.com/z3t0/Arduino-IRremote.

Apêndices

Apêndice A

Ajuste da Árvore de Dispositivos

```
/{
    compatible = "ti,beaglebone", "ti,beaglebone-black";
    part_number = "BS_PINMODE_P8_16_0x26";

    exclusive-use = "P8.16", "pr1_pru0_pru_r31_14";

    fragment@0 {
        target = <&am33xx_pinmux>;
        __overlay__ {
            bs_pinmode_P8_16_0x26: pinmux_bs_pinmode_P8_16_0x26 {
                pinctrl-single,pins = <0x038 0x26>;
            }; }; };

    fragment@1 {
        target = <&ocp>;
        __overlay__ {
            bs_pinmode_P8_16_0x26_pinmux {
                compatible = "bone-pinmux-helper";
                status = "okay";
                pinctrl-names = "default";
                pinctrl-0 = <&bs_pinmode_P8_16_0x26>;
            }; }; };
};
```

Listagem A.1: Arquivo DTS para a captura do sinal pelo sensor IR no pino P8_16.

```
/ {
    compatible = "ti,beaglebone", "ti,beaglebone-black";
    part-number = "BS_PWM_P8_13_0x14";
    exclusive-use = "P8.13", "ehrpwm2B";

    fragment@0 {
        target = <&am33xx_pinmux>;
        __overlay__ {
            bs_pwm_P8_13_0x14: pinmux_bs_pwm_P8_13_0x14 {
                pinctrl-single,pins = <0x024 0x14>;
            }; }; };

    fragment@1 {
        target = <&ocp>;
        __overlay__ {
            bs_pwm_test_P8_13 {
                compatible = "pwm_test";
                pwms = <&ehrpwm2 1 500000 1>;
                pwm-names = "PWM_P8_13";

                pinctrl-names = "default";
                pinctrl-0 = <&bs_pwm_P8_13_0x14>;

                enabled = <1>;
                duty = <0>;
                status = "okay";
            }; }; };
};
```

Listagem A.2: Arquivo DTS para envio do sinal pelo LED IR no pino P8_13.

Apêndice B

Gramática Livre de Contexto para o Julius

```
<s>         sil
</s>        sil
ligar       l i g a X
desligar    d e s l i g a X
televisão   t e l e v i z a~ w~
aumentar    a u~ m e~ t a X
diminuir    dZ i~ m i~ n u j X
volume      v o l u~ m i
canal       k a n a w
mais        m a j s
menos       m e~ n u s
```

Listagem B.1: Dicionário fonético utilizado na aplicação (`.dict`).

```
S: NS_B SENTENCE NS_E
SENTENCE: ACAO_CONTROLE DISPOSITIVO
SENTENCE: ACAO_VOLUME VOLUME
SENTENCE: CANAL ACAO_CANAL
```

Listagem B.2: Gramática estilo BNF para o Julius (`.gram`).

```
% NS_B
<s>     sil

% NS_E
</s>    sil

% ACAO_CONTROLE
ligar       <s> l i g a X </s>
desligar    <s> d e s l i g a X </s>

% ACAO_VOLUME
aumentar    <s> a u~ m e~  t a X </s>
diminuir    <s> dZ i~ m i~ n u j X </s>

% ACAO_CANAL
mais        <s> m a j s </s>
menos       <s> m e~ n u s </s>

% DISPOSITIVO
televisao    <s> t e l e v i z a~ w~ </s>

% VOLUME
volume      <s> v o l u~ m i </s>

% CANAL
canal       <s> k a n a w </s>
```

Listagem B.3: Transcrições fonéticas na gramática (`.voca`).

Apêndice C

Configuração do Banco de Dados

```
DROP database if EXISTS ir_db;
CREATE database if NOT EXISTS ir_db;
USE ir_db;

DROP table if EXISTS tv;
CREATE table if NOT EXISTS tv (
marca varchar(40) NOT null,
vol_mais varchar(40),
vol_menos varchar(40),
ch_mais varchar(40),
ch_menos varchar(40),
on_off varchar(40),
primary key(marca)
);

INSERT into tv(marca, vol_mais, vol_menos, ch_mais, ch_menos, on_off)
values ("samsung",
"11110000011100000111000000000111110", -- volume mais
"11110000011100000110100000010111110", -- volume menos
"11110000011100000010010001011011110", -- canal mais
"11110000011100000000010001111011110", -- canal menos
"11110000011100000010000001011111110"  -- on/off
);
```

Listagem C.1: Código SQL para configuração do Banco de Dados

```
int setup() {
    MYSQL *con = mysql_init(NULL);  // inicia conexao

    /* conecta ao MySQL (mysql -u root -p [beagle1234]) */
    mysql_real_connect(con, "localhost",
                "root", "beagle1234", NULL, 0, NULL, 0);

    drop_db(con, "ir_db"); // exclui banco
    create_db(con, "ir_db"); // cria banco
    mysql_select_db(con, "ir_db"); // seleciona/muda de banco

    drop_table(con, "tv"); // exclui tabela, caso exista
    create_table(con, "tv"); // cria tabela

    /* set branch on primary key and values for remaining fields */
    insert_table(con, "marca", "samsung");
    set_element(con, "vol_mais",
                "11110000011100000111000000000111110", "samsung");
    set_element(con, "vol_menos",
                "11110000011100000110100000001011110", "samsung");
    set_element(con, "ch_mais",
                "11110000011100000010010001011011110", "samsung");
    set_element(con, "ch_menos",
                "11110000011100000000010001111011110", "samsung");
    set_element(con, "on_off",
                "11110000011100000010000001011111110", "samsung");

    mysql_close(con); // fecha conexao com o banco

    return EXIT_SUCCESS;
}
```

Listagem C.2: Função `setup()` para configuração do Banco de Dados