CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA – CEFET/RJ

Sistema para Mapeamento de Códigos Infravermelho e Criação de Controles Remotos Virtuais

Calvin Martins de Oliveira

Orientador: Rafael Castaneda

Rio de Janeiro
Fevereiro/2015

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA – CEFET/RJ

Sistema para Mapeamento de Códigos Infravermelho e Criação de Controles Remotos Virtuais

Calvin Martins de Oliveira

Trabalho de conclusão de curso apresentado no CEFET/RJ, como um dos requisitos para a obtenção do título de Tecnólogo em Sistemas para Internet

Orientador: Rafael Castaneda

Rio de Janeiro
Fevereiro/2015

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

O48 Oliveira, Calvin Martins de
Sistema para mapeamento de códigos infravermelho e criação de controles remotos virtuais / Calvin Martins de Oliveira.—2015.
ix, 56f. + apêndice : il.color. ; enc.

Trabalho de Conclusão de Curso (Tecnólogo) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca , 2015.
Bibliografia : f. 55-56
Orientador : Rafael Castaneda

1. Controle remoto. 2. Radiação infravermelha. 3. Arduino (Controlador programável). 4. Android (Recurso eletrônico). I. Castaneda, Rafael (Orient.). II. Título.

CDD 620.46

Resumo

Este projeto visa proporcionar a possibilidade de criação de controles remotos universais programáveis e personalizáveis, que podem ser usados tanto por mera conveniência quanto para substituir controles originais no caso de uma eventual necessidade, como sua perda, quando forem danificados ou durante o período de espera entre cargas de pilha. Este objetivo é alcançado através do desenvolvimento de um sistema de detecção dos comandos enviados pelos controles, que são então aprendidos por um aplicativo de computador e podem ser reproduzidos novamente por ele. Para isto, o programa se comunica com uma interface composta de um sensor infravermelho (IR) e um Diodo Emissor de Luz (LED) infravermelha conectados a uma placa controladora Arduino responsável por processar a entrada e saída dos comandos, na forma de pulsos de luz infravermelha.

Palavras-chave: Controle Remoto, Infravermelho, Arduino, Dispositivos Móveis, Android

Abstract

This project aims for providing the possibility of creating programmable and customizable universal remotes, which can be used both for mere convenience as well as replacing original remotes in the case of eventual need, like if they are lost, damaged or during battery recharge periods. This goal is achieved through the development of a system that detects the commands sent by the remotes, which are then learned by a computer application and can be reproduced back by it. To achieve this, the program communicates with an interface composed of an infrared sensor and an infrared Light Emitting Diode connected to an Arduino controller board which is responsible for processing the input and output of commands, in the form of infrared light pulses.

Keywords: Remote Control, Infrared, Arduino, Mobile Devices, Android

Sumário

Lista de Figuras

Lista de Abreviaturas

Hz – Hertz

ID – Identificador, Identidade

IR – Infravermelho

kHz – Kilohertz

MHz – Megahertz

ms – Milissegundos

μs – Microssegundos

Lista de Siglas

CPU – *Central Processing Unit* (Unidade Central de Processamento)

ER – Entidade-Relacionamento

IDE – *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)

IP – *Internet Protocol* (Protocolo de Internet)

LED – *Light Emitting Diode* (Diodo Emissor de Luz)

MCU – *Microcontroller Unit* (Unidade Microcontroladora)

SQL – *Structured Query Language* (Linguagem Estruturada de Consultas)

USB – *Universal Serial Bus* (Barramento Serial Universal)

1 Introdução

Na era moderna em que vivemos, praticamente qualquer pessoa possui em seu lar pelo menos um dispositivo eletrônico, podendo ser ele um aparelho de televisão, rádio, computador, console de videogame, entre diversos outros tipos, muitos destes possuindo um controle remoto próprio, que como o nome sugere, é um aparelho que pode ser utilizado para passar comandos a estes dispositivos à distância. Estes controles remotos utilizam padrões bem definidos e consolidados para realizar a comunicação com os dispositivos que comandam, algo do qual pode-se tirar vantagem para a criação de sistemas de controle mais convenientes e interessantes, que resolvam não apenas os problemas que podem ocorrer quando se possui muitos aparelhos diferentes para controlar, como também apresentem novas possibilidades, realizando o controle dos dispositivos de formas diferenciadas, que seriam impossíveis utilizando apenas os controles remotos quem vêm com tais dispositivos em suas embalagens. Uma destas formas é a utilização de controles remotos universais, que são aparelhos que funcionam exatamente como controles remotos convencionais, porém não estão associados a um único dispositivo ou marca, sendo capazes de controlar dispositivos de muitos modelos diferentes. O sistema proposto por este projeto é baseado no funcionamento destes aparelhos.

1.1 Motivação

Controles remotos universais são dispositivos que podem ser utilizados para exercer o controle sobre um ou mais aparelhos eletrônicos domésticos sem a necessidade de utilizar seus controles originais vindos de fábrica. A conveniência de se possuir um destes dispositivos se mostra presente no fato de que o mesmo é capaz de proporcionar diversas vantagens ao usuário, especialmente em um ambiente onde haja muitos aparelhos, como um centro de entretenimento, por exemplo. Por outro lado, um controle remoto universal convencional, que se pode obter no mercado pode também possuir algumas desvantagens, como ser demasiadamente caro, complicado de se usar ou incompatível com algumas marcas de aparelhos mais obscuras. Este projeto foi desenvolvido com o intuito de eliminar estas e outras desvantagens, proporcionando um controle virtual de fácil acesso, que possa ser usado por qualquer pessoa independentemente de conhecimento técnico e sem problemas de compatibilidade.

1.2 Objetivos

Para alcançar o objetivo de proporcionar um controle remoto universal verdadeiramente conveniente, este projeto elabora um sistema que:

- Seja capaz de processar o recebimento de representações de códigos infravermelho e armazenar tais representações, organizadas por controle remoto;

- Possa reproduzir os códigos infravermelhos em sua forma original, convertendo-os de volta à mesma a partir da representação armazenada;
- Forneça ao usuário uma ou mais opções para utilizar os códigos armazenados para controlar dispositivos eletrônicos efetivamente

1.3 Metodologia

Este projeto consiste em um sistema capaz de detectar os códigos de comando de controles remotos de aparelhos eletrônicos, armazená-los e mapeá-los para outras interfaces de controle. O projeto utiliza um sensor IR que é responsável por detectar os pulsos de luz do sinal enviado pelos controles remotos, conectado a uma placa controladora Arduino, na qual roda o programa de detecção de códigos. Também está ligado a este esquema um LED infravermelho, que é utilizado para reproduzir os códigos mapeados. A implementação atual permite a criação de controles remotos virtuais aos quais o usuário pode adicionar botões que serão associados aos códigos detectados e pulsar estes códigos no LED quando tais botões forem clicados, reproduzindo o comando através de um sinal infravermelho similar àquele enviado pelo controle remoto original e consequentemente controlando o dispositivo eletrônico. É possível também exportar o controle virtual criado no programa principal para um aplicativo para o sistema operacional Android, que permite que o controle seja carregado na tela de um dispositivo móvel, como smartphone ou tablet e utilizado como um controle remoto substituto. Por fim, há ainda a possibilidade de criar uma associação entre os próprios códigos IR, mapeando os códigos de um controle remoto para outro e usando o controle de um aparelho para controlar outros, independentemente de tipo ou marca.

1.4 Organização dos Capítulos

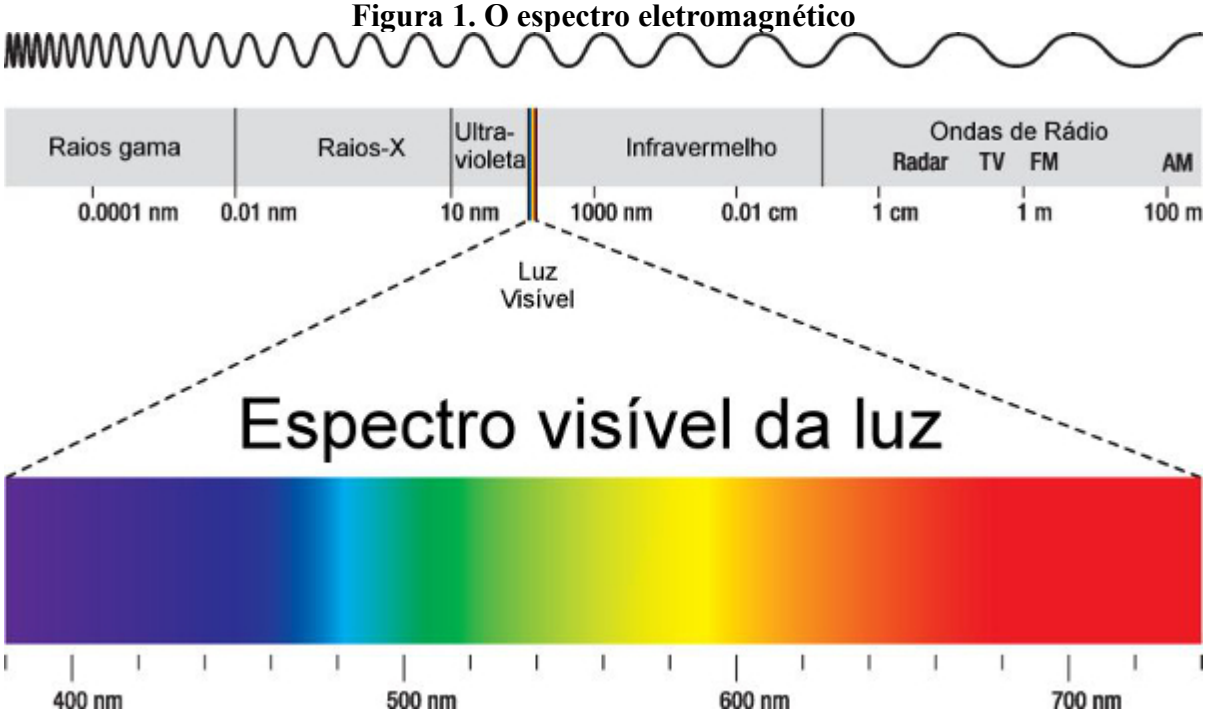
Este documento está estruturado em capítulos que serão chamados de seções, cada uma abordando um aspecto diferente da elaboração do projeto. Na primeira seção, onde este parágrafo se encontra, pode-se ver uma introdução geral do trabalho, que descreve a motivação por trás do mesmo, seus objetivos e metodologia utilizada. Na segunda seção, a base teórica para o estudo realizado na elaboração do projeto pode ser encontrada, sendo abordados nesta os conceitos técnicos existentes no trabalho, como aqueles referentes ao funcionamento de controles remotos e de conexões de rede via a interface de *sockets*. Nesta seção também é feita uma introdução à plataforma Arduino e são mostrados alguns trabalhos relacionados ao projeto. Na terceira seção é descrita a modelagem do sistema e na quarta sua elaboração, sendo esta a parte principal do documento, que é seguida por uma avaliação prática do projeto realizada após o término do mesmo, existente na quinta seção. Por fim, na sexta seção, encontra-se a conclusão do trabalho, incluindo as considerações finais sobre o projeto e trabalhos futuros que podem ser realizados no mesmo.

2 Fundamentação Teórica

2.1 Funcionamento de Controles Remotos e Códigos Infravermelho

Para o correto entendimento deste projeto, é necessário conhecer o funcionamento de um controle remoto convencional, como aqueles que a maioria das pessoas possui em casa e pode ser usado para enviar os mais variados tipos de comandos, como “avançar canal”, “abaixar volume”, “ejetar CD”, “reproduzir”, “pausar”, “parar” e muitos outros, dependendo do dispositivo. Estes comandos são enviados na forma de sinais de luz infravermelha, que, como qualquer outro tipo de luz, compreende uma parte do espectro eletromagnético, de acordo com a teoria do eletromagnetismo de Maxwell[1].

Figura 1. O espectro eletromagnético



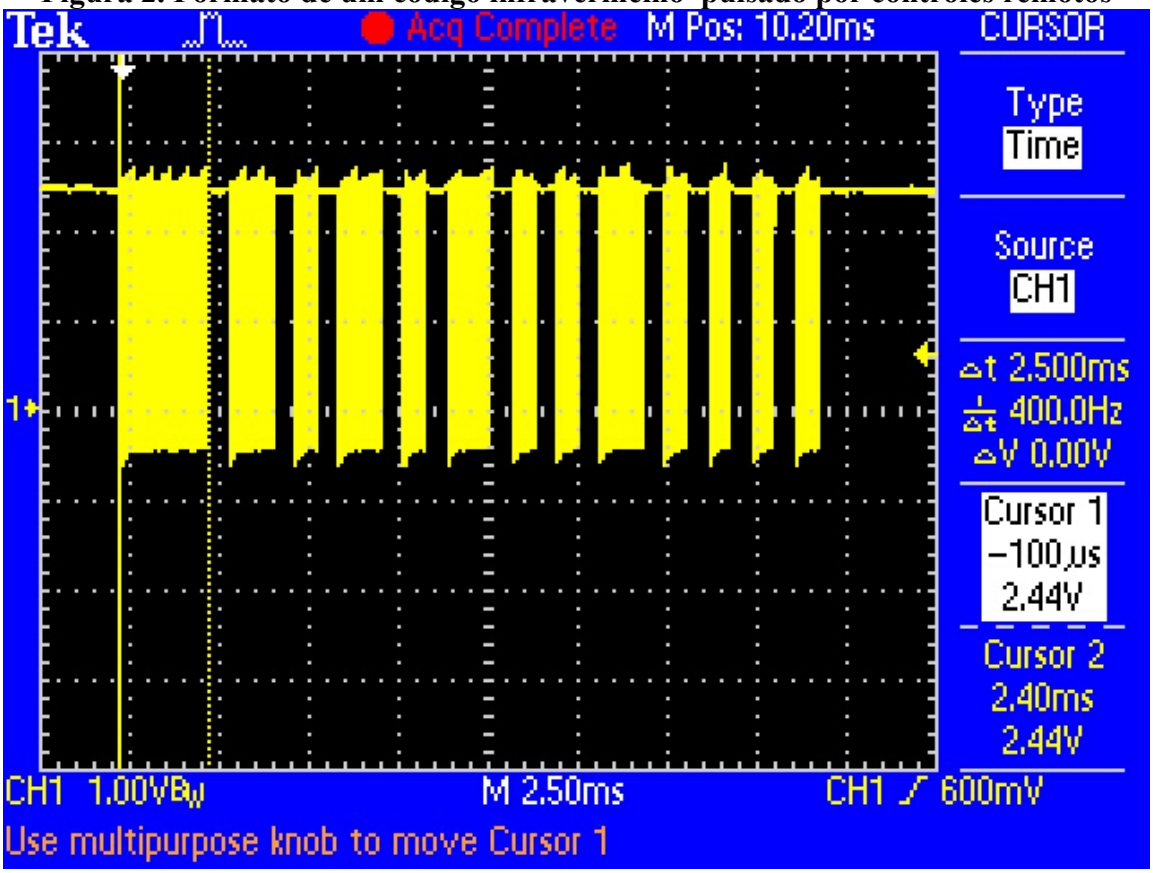
Fonte: Disponível em http://www.infoescola.com/fisica/espectro-eletromagnetico/, acessado em Fevereiro de 2015

A luz infravermelha é invisível ao olho humano, por esta razão, não é possível ver o sinal enviado pelos controles remotos a olho nu, mesmo ao olhar diretamente para o LED existente dentro do controle, que é responsável por pulsar os códigos. Um pulso significa uma alternância entre os estados ativo e inativo do LED, isto é, quando a luz está acesa ou apagada. No caso de pulsos de luz, é possível entendê-los como sendo o momento em que a luz pisca. Esta alternância de estado, para cada comando do controle, é repetida diversas vezes em intervalos de tempo muito pequenos, tal que seriam impossíveis de perceber com os olhos, mesmo que se pudesse enxergar a luz. Uma forma de comprovar isto é utilizando uma câmera

apontada para o LED no momento do pressionamento de algum botão do controle. A câmera, que consegue captar luz infravermelho, capturará o pulso do código e mostrará o LED piscando rapidamente, tornando o sinal “visível” para quem o observar através da tela. Ao executar estas repetições de pulsos com durações variadas, cria-se um código, que funciona basicamente da mesma forma que o código Morse e pode ser interpretado pelos sensores de luz infravermelho que existem dentro dos aparelhos eletrônicos.

Mesmo quando o LED se acende durante a parte ativa de um pulso, ele não se mantém nesse estado por toda a duração desta parte mas na verdade apenas por metade do tempo, já que o sinal que compõe estes códigos, para ser detectado pelos sensores, deve estar pulsando em oscilações ainda mais rápidas, na frequência de 38 Kilohertz (kHz), que é a frequência para a qual este tipo de sensor é ajustado para detectar. Na física, a unidade Hertz (Hz) é utilizada para indicar a repetição de um fenômeno em um período de tempo, sendo 1 Hz o equivalente a uma oscilação (ciclo) por segundo[2]. Isto significa que apenas acender e apagar o LED uma vez para cada pulso faria com que os códigos não fossem detectados efetivamente pelo sensor[3] e que para isto, a luz do LED deve estar pulsando em uma frequência de 38000 ciclos por segundo. A Figura 2 ilustra o formato de um código infravermelho, onde os “blocos” amarelos representam a parte ativa do pulso (*ON*) e os “espaços” entre eles representam a parte inativa (*OFF*).

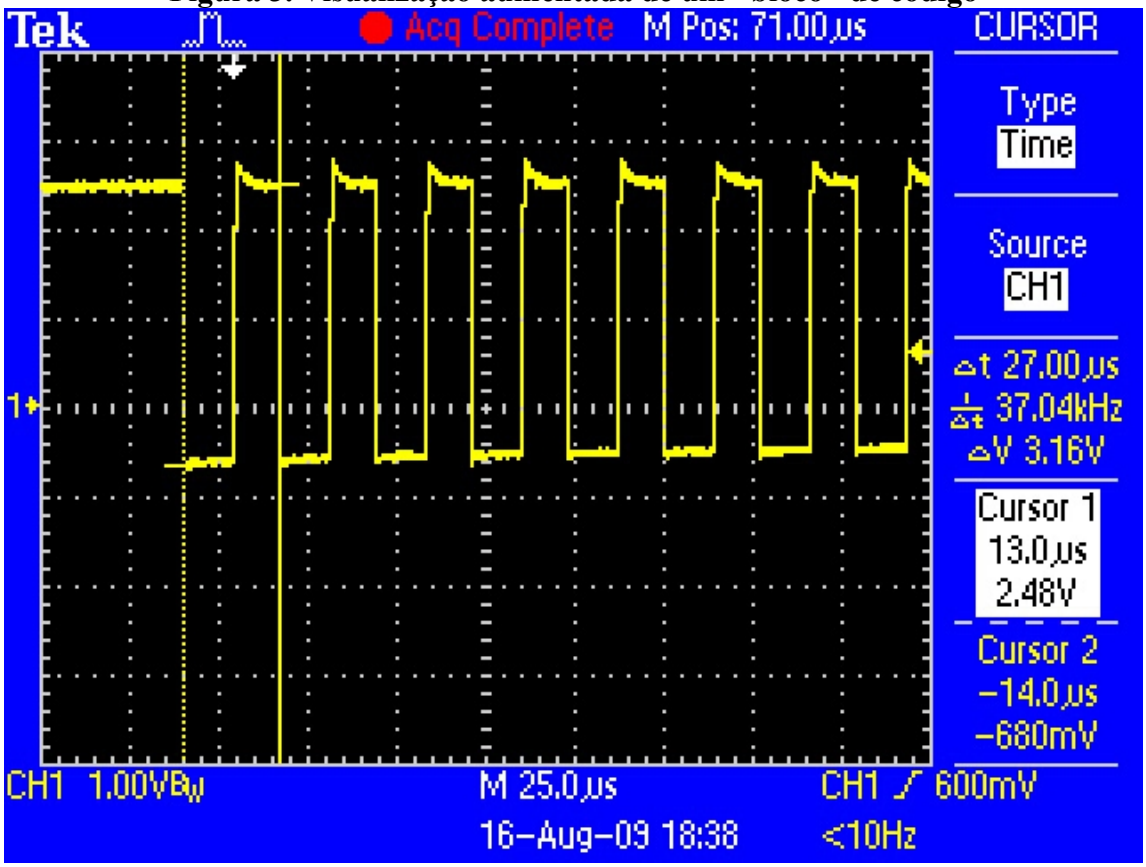
Figura 2. Formato de um código infravermelho pulsado por controles remotos



Fonte: Disponível em http://www.ladyada.net/wiki/tutorials/learn/sensors/ir.html, acessado em Fevereiro de 2015

A Figura 3 mostra uma visualização aumentada de um dos “blocos” observados. É possível perceber que mesmo a parte ativa do pulso possui alguns momentos durante os quais o LED está apagado. Esta oscilação ocorre na casa dos microssegundos (milionésimos de segundo).

Figura 3. Visualização aumentada de um "bloco" de código



Fonte: Disponível em http://www.ladyada.net/wiki/tutorials/learn/sensors/ir.html, acessado em Fevereiro de 2015

Considerando este formato, para que se possa detectar e armazenar um código de forma precisa, é necessário identificar as durações tanto dos “blocos” quanto dos “espaços” e desenvolver uma forma eficaz para transmitir estas informações. A forma como isto foi feito no projeto é abordada na seção 4.1.

2.2 A Plataforma Arduino

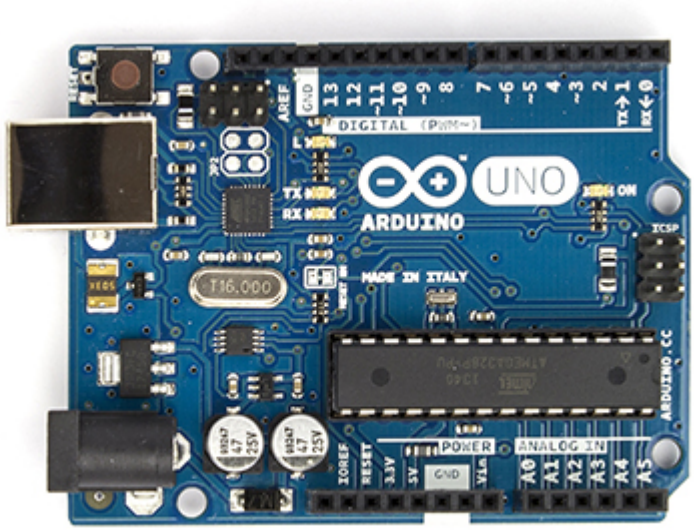
As placas Arduino são microcontroladoras programáveis desenvolvidas a partir de um modelo de hardware aberto, bem semelhante à abordagem do software aberto, mais conhecido como “*open-source*”, cuja licença permite que qualquer pessoa altere os componentes originais da placa e crie suas próprias versões, razão pelo qual há no mercado diversos modelos “compatíveis” de Arduino, que na verdade nada mais são do que versões diferentes da placa criadas por entusiastas ou profissionais da eletrônica a partir do design original.

Esta plataforma foi desenvolvida na Itália em 2005 por Massimo Banzi e mais quatro

engenheiros em prol de criar um modelo de placa controladora barata e de fácil uso para que os estudantes de Banzi pudessem se envolver com eletrônica e desenvolver projetos mais facilmente. A partir de sua criação, a placa rapidamente ganhou sucesso e se mostrou muito utilizada por entusiastas de eletrônica ao redor do mundo para o desenvolvimento de projetos próprios, conceito conhecido como "*Do It Yourself*" (faça você mesmo), ou DIY[4].

Um dos modelos mais amplamente utilizados de placas Arduino é o chamado Uno, que é um dos modelos originais e também o utilizado neste projeto. Este modelo utiliza um chip conversor 16U2 USB-to-serial e um chip ATMega 328p como principal unidade microcontroladora (MCU). Esta placa está disponível em duas versões, DIP e SMD, que diferem na condição da MCU ser removível ou não[5]. Pode-se ver na Figura 4, a aparência da placa Arduino Uno.

Figura 4. A placa Arduino Uno



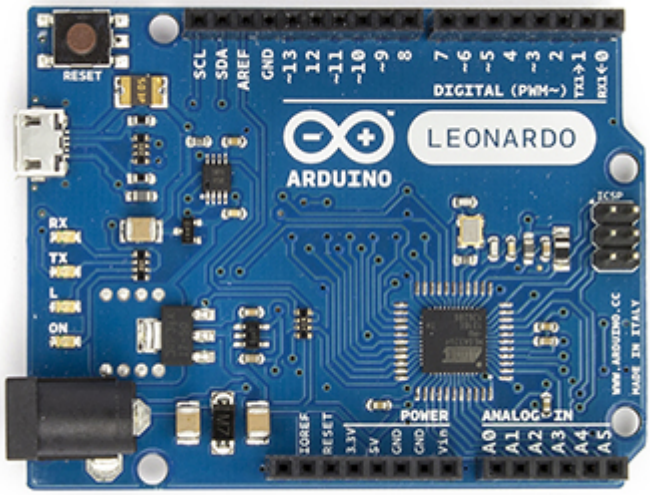
Fonte: Disponível em http://arduino.cc/en/uploads/Main/ArduinoUno_R3_Front_450px.jpg, acessado em Março de 2015

Como este é o modelo utilizado neste projeto, mais informações sobre o mesmo podem ser encontradas na seção 4.1, que trata da documentação da parte do projeto na qual é pertinente a descrição das capacidades da placa. Além desta, diversos outros modelos originais italianos de placas Arduino existem para aquisição do usuário comum, entre eles os seguintes:

- **Leonardo**

Este modelo é baseado no ATmega32u4 e possui 20 pinos de entrada e saída digital, um oscilador de cristal de 16 Megahertz (MHz), uma conexão micro USB, conector de alimentação, conectores ICSP e um botão de reiniciar[6]. Na Figura 5 pode-se ver o design deste modelo.

Figura 5. A placa Arduino Leonardo



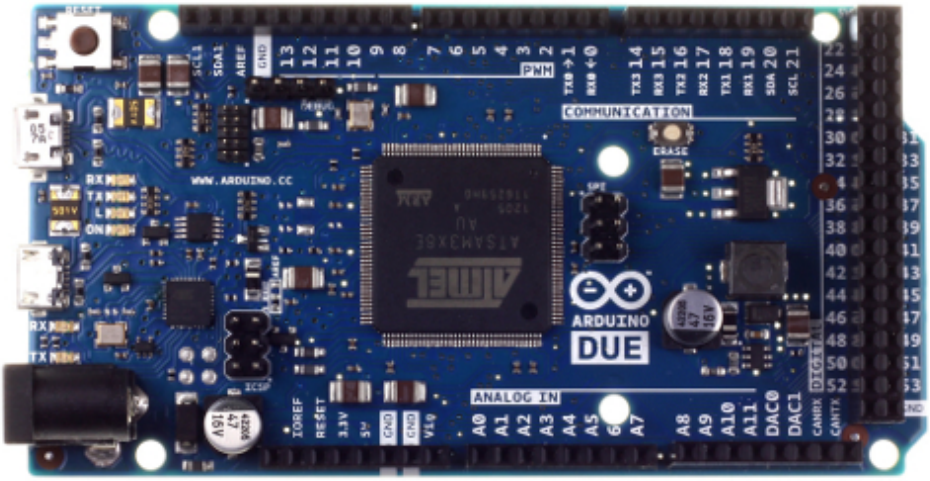
Fonte: Disponível em http://arduino.cc/en/uploads/Main/ArduinoLeonardoFront_2_450px.jpg, acessado em Março de 2015

É possível perceber que este modelo é bem semelhante ao modelo Uno, diferindo do mesmo principalmente no fato de possuir uma conexão micro USB ao invés de USB convencional e pelo fato do chip microcontrolador ser montado na superfície da placa (*surface mount*), o que faz com que ele não seja removível.

- **Due**

Este modelo é baseado na CPU Atmel SAM3X8E ARM Cortex-M3 e é o primeiro Arduino baseado numa microcontroladora ARM de 32 bits. Possui 54 pinos de entrada e saída digital, 12 pinos de entrada analógica, 4 UARTs, um *clock* de 84 MHz, uma conexão USB OTG, 2 conversores *digital-to-analog*, 2 TWI, conector de alimentação, cabeçalho SPI, cabeçalho JTAG, um botão de reiniciar e um botão de apagar[7]. A Figura 6 ilustra este modelo de placa.

Figura 6. A placa Arduino Due



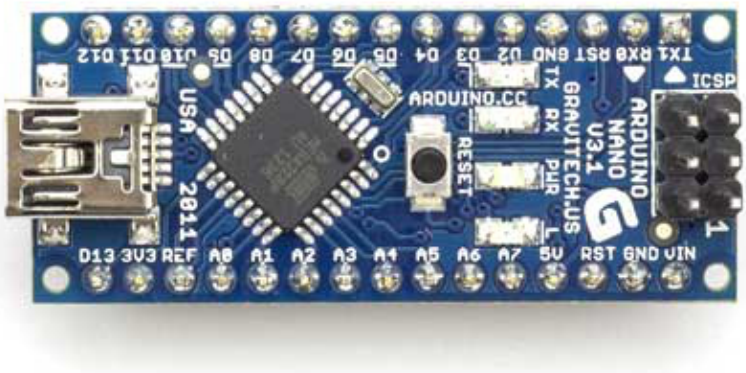
Fonte: Disponível em http://arduino.cc/en/uploads/Main/ArduinoDue_Front.jpg, acessado em Março de 2015

Dada a grande quantidade de recursos desta placa, que incluem um número bem maior de pinos de entrada e saída, conversores de sinal digital para analógico entre muitos outros, pode-se notar que este modelo é voltado para projetos mais robustos, que tenham uma demanda computacional maior.

- **Nano**

Este modelo é baseado no ATmega328 (Arduino Nano 3.x) ou ATmega168 (Arduino Nano 2.x) e é pequeno, completo e tem um design compatível com placas de ensaio (também conhecidas como *protoboards* ou *breadboards*). Difere de outros modelos por não possui um conector de alimentação e utilizar um conector USB Mini-B ao invés de USB convencional[8]. Na Figura 7, pode-se observar a placa.

Figura 7. A placa Arduino Nano



Fonte: Disponível em http://arduino.cc/en/uploads/Main/ArduinoNanoFront_3_sm.jpg, acessado em Março de 2015

Tendo em vista o seu tamanho reduzido, este modelo, que não fica muito atrás de outros no quesito recursos, pode ser utilizado em projetos onde tanto o espaço físico que os

componentes ocupam quanto o peso que possuem é relevante, como projetos de máquinas voadoras por exemplo.

Considerando as diferenças existentes entre os modelos, tanto em design quanto em capacidades, evidentemente cabe ao usuário escolher o apropriado para o desenvolvimento de seu projeto. Estas são apenas algumas entre as muitas placas disponíveis, que incluem outras como o modelo Ethernet, que possui um conector RJ45 para projetos que envolvam conexão de rede através de cabos[9] e o modelo Esplora, que tem um design similar ao de um controle de videogame, possuindo inclusive um *joystick*, além de um acelerador de três eixos, microfone e outros componentes interessantes para projetos voltados à interatividade[10].

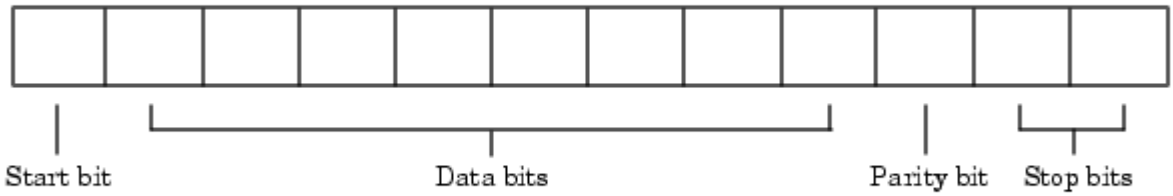
Há ainda os *shields*, que são placas que podem ser ligadas às placas Arduino para estender ainda mais suas funcionalidades, como o Wi-fi Shield, usado para conectar a Arduino hospedeira à internet através de conexões sem fio[11] e o Motor Shield, usado para controlar motores[12]. Como não é utilizado nenhum *shield* neste projeto, eles não serão abordados com maior profundidade.

2.3 Transmissão de Dados Seriais entre o Programa e a Placa Controladora Arduino Uno

Para que os dados detectados pelo sensor infravermelho possam ser enviados ao programa principal, que é responsável por tratá-los e armazená-los durante a detecção de um código e também possam ser enviados do programa para a placa durante uma requisição de pulso, é necessário transmitir estes dados de forma rápida entre as duas plataformas. Isto é feito através de uma conexão serial feita através de um cabo *Universal Serial Bus* (USB).

O formato de dados seriais inclui um bit de início (*start*), cinco a oito bits de dados e um bit de parada (*stop*), com um bit de paridade e um bit de parada adicional podendo ser adicionados. A velocidade de bits transferidos por segundo, incluindo os bits de início, paridade e os de parada é definida pela taxa baud (*baud rate*)[13]. O padrão utilizado pela Arduino é usar oito bits de dados, sem paridade e um único bit de parada, com a taxa baud podendo ser definida para uma série de valores que vão de 300 a 115200[14]. Uma das taxas mais utilizadas para a comunicação com computadores é a de 9600, a mesma que é utilizada neste projeto.

Figura 8. Formato típico de um byte serial



Fonte: Disponível em https://nf.nci.org.au/facilities/software/Matlab/techdoc/matlab_external/ch_seri8.html, acessado em Fevereiro de 2015

As mensagens são trocadas entre a placa controladora e o programa na forma de *strings* ou cadeias de caracteres. Para que os dados sobre os códigos IR possam ser passados de uma plataforma à outra neste formato, é necessário haver uma convenção para sua conversão, que definirá um protocolo para envio das mensagens. Na seção 4.1 pode ser encontrada uma explicação sobre a solução aplicada. A placa Arduino Uno possui um *buffer* de entrada de 64 bytes[15], o que significa que ela pode receber e processar até 64 caracteres de texto em uma única mensagem. Para que mais caracteres possam ser recebidos, é necessário esvaziar o *buffer*, lendo os bytes armazenados e liberando seu espaço.

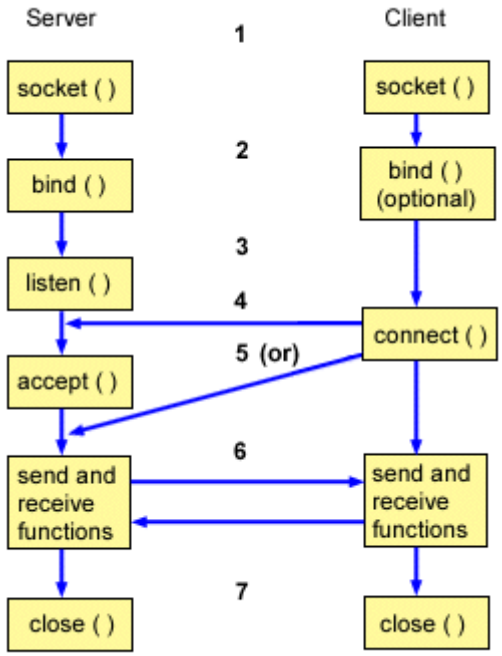
2.4 Transmissão de Dados via *Sockets*

Embora o foco central do projeto seja a detecção e a manipulação de códigos IR, é importante que haja uma forma eficiente para que os dados sobre estes códigos sejam transmitidos via rede, permitindo assim a integração de outros aplicativos ao programa principal que têm como papel estender a funcionalidade do mesmo. Neste projeto, um aplicativo para o sistema operacional Android é utilizado com esta finalidade e para que a comunicação entre as plataformas seja feita, o conceito de *sockets* (ou soquetes) é utilizado.

Sockets são uma abstração através da qual uma aplicação pode enviar e receber dados, da mesma forma que um arquivo aberto permite que uma aplicação leia e escreva dados a um armazenamento estável. Um *socket* permite que uma aplicação se “plugue” a uma rede e se comunique com outras aplicações que estão plugadas à mesma rede. A informação escrita em um *socket* por uma aplicação em uma máquina pode ser lida por uma aplicação em uma máquina diferente, e vice-versa[16]. A interação entre essas duas entidades ocorre na arquitetura cliente-servidor e para ser estabelecida necessita que o cliente conheça o nome ou endereço *Internet Protocol* (IP) do servidor para requisitar uma conexão e este servidor, que deve estar ouvindo por conexões, deve aceitá-la. Se a conexão tem sucesso, o servidor designa um novo *socket* ao mesmo número de porta único especificado na conexão e define um ponto

remoto (*remote endpoint*) para o endereço e porta do cliente. A criação deste novo *socket* é necessária para que o servidor possa continuar ouvido por conexões enquanto atende às necessidades do cliente já conectado[17]. A Figura 9 ilustra a forma básica como as conexões de rede via *socket* são feitas.

Figura 9. Representação gráfica de uma conexão via *sockets*



Fonte: Disponível em http://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_71/rzab6/howdosockets.htm, acessado em Fevereiro de 2015

2.4.1 *Sockets* Síncronos vs. Assíncronos

Em computação, uma operação síncrona é aquela que bloqueia o processo até que a operação se complete, enquanto uma operação assíncrona não o bloqueia e é responsável apenas por seu início[18]. Quando se trabalha com *sockets*, isto não é diferente. *Sockets* síncronos são aqueles que bloqueiam a execução de um processo até que se receba uma resposta da outra parte, e os assíncronos apenas enviam a mensagem que deve ser enviada mas não esperam resposta. Quando se recebe uma mensagem assíncrona de um *socket*, é necessário chamar uma função *callback* previamente definida para tratar os dados recebidos. A solução utilizada no projeto foi a utilização de *sockets* síncronos que funcionam em *threads* separados aos da interface gráfica dos programas. É possível entender *threads* como ambientes de execução separados de um mesmo processo, que funcionam paralelamente. O próprio projeto pode servir como um exemplo para ilustrar a função dos *threads*: há o *thread* principal, responsável por manter a responsividade da interface gráfica, garantindo que os botões sempre responderão quando forem clicados, enquanto em um outro *thread*, o sistema escuta por mensagens vindas dos *sockets*. Caso o projeto utilizasse *sockets* síncronos sem

fazer esta separação de tarefas em *threads*, ao enviar uma mensagem, o programa congelaria até que uma resposta fosse recebida ou o envio falhasse, bloqueando a interface gráfica e fazendo o programa não responder. Esta abordagem é utilizada tanto no programa principal quando no aplicativo para Android. Neste último, porém, em vez de *threads*, é usada a classe AsyncTask, que é uma interface mais simples fornecida pela plataforma para a execução de tarefas curtas em plano de fundo e a publicação de seus resultados no *thread* da interface gráfica[19]. O uso de AsyncTasks é descrito na seção 4.3.

2.5 Trabalhos Relacionados

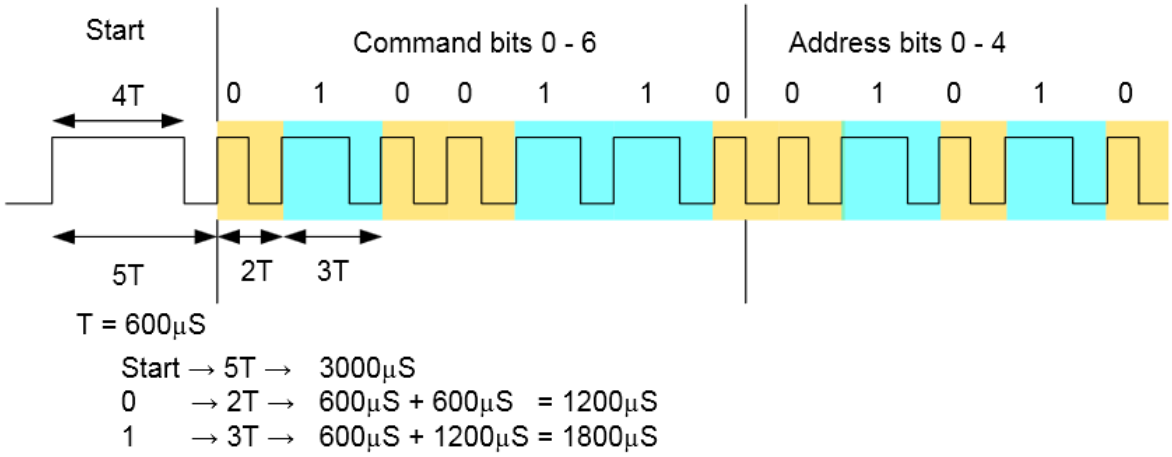
Nesta seção, serão apresentados dois projetos de decodificadores IR, utilizados para interpretar códigos em protocolos específicos, que serão abordados juntamente com as soluções propostas. Embora o produto deste projeto não seja um decodificador, mas apenas um reprodutor de códigos, a análise destes trabalhos é de grande utilidade para o correto entendimento dos conceitos de comunicação via sinais IR. Por fim, serão apresentados alguns aplicativos e acessórios que podem ser utilizados para alcançar uma funcionalidade similar à deste projeto, acessíveis ao público para compra.

2.5.1 Decodificador de Códigos no Protocolo SIRC

O protocolo SIRC se baseia na duração dos pulsos para a transmissão dos dados sobre o código, sendo a duração do período utilizado pelos pulsos um múltiplo de 600 microssegundos (µs), representado por T. Neste protocolo, cada segmento de dados começa com um pulso alto de duração 4T seguido de um pulso baixo (ausência de sinal) de duração 1T, o que faz com que a duração do primeiro pulso - sendo um pulso composto de um par alto/baixo ou *ON/OFF* - seja de 5T, ou 300µs, que é equivalente a 3 milissegundos (ms). Os pulsos seguintes são as representações do “0” e do “1” lógicos, onde um pulso alto de 1T seguido de um pulso baixo de 1T representa um bit 0 e um pulso alto de 2T seguido de um pulso baixo de 1T representa um bit 1, com as durações de 1200µs e 1800µs respectivamente. Cada pulso alto é seguido de um pulso baixo de 1T[20].

Há três versões do protocolo: com 12, 15 e 20 bits. Todas as versões possuem bits reservados para definir o comando e o dispositivo a ser controlado e a versão de 20 bits possui 8 bits extras[21]. Para cada uma destas definições, os bits menos significantes (*Least Significant Bits*) são enviados primeiro. Ao receber dados, é necessário inverter o sinal recebido no pino de entrada do controlador, pois a saída do receptor IR é *active low*, o que significa que quando um sinal infravermelho estiver sendo detectado, a saída será baixa (*low*) e quando nenhum sinal estiver sendo recebido, será alta (*high*). A Figura 10 mostra uma representação gráfica do protocolo SIRC.

Figura 10. O Protocolo SIRC



Fonte: Capturada do arquivo PDF disponível em http://picprojects.org.uk/projects/sirc/sonysirc.pdf, acessado em Fevereiro de 2015

A solução apresentada para detectar os códigos SIRC utiliza como componentes o Chip PIC 16F88 e um sensor infravermelho TSOP2238. A detecção do sinal é feita da seguinte maneira:

1. Após o estado inativo inicial, é necessário detectar a primeira transição de pulso alto para baixo, que é chamada de *falling edge*, com o sentido de uma queda de sinal. Na representação gráfica de um pulso, esta transição é representada pelas partes onde a linha horizontal muda de direção e passa a ir para baixo. Após a detecção, um temporizador (*timer*) de 2500µs é iniciado. Se outra dessas transições for detectada antes do fim do temporizador, a recepção de dados deve ser interrompida. Caso contrário, deve prosseguir com a mesma e começar a receber os bits.

2. Como os bits são recebidos começando pelos menos significantes, os dados são escritos a partir deles, ou "de trás para frente", com os bits mais significantes (*Most Significant Bits*) sendo recebidos depois. Já que o sinal é um comando de 7 bits mais um identificador (ID) de dispositivo de 5 bits, o código deve dividir os bits recebidos em dois grupos, de 7 e 5 bits. Ao serem convertidos para números decimais, estes bits formarão a identificação do dispositivo e o comando.

3. O código de detecção aguarda por uma transição alto-baixo (*falling edge*). Se esta for detectada, inicia um temporizador de 900µs, que ao chegar ao fim escreverá o bit nos dados de entrada. Se a transição não for detectada em até 1200µs, o recebimento é abortado. Se outra transição for detectada antes do fim do temporizador (*timeout*), o recebimento também é abortado.

4. O código de recebido executa em *loop* até que os 12 bits sejam recebidos e então passa a ouvir por mais transições em um período de 10ms. Se alguma for detectada durante este período, fará com que o código de recepção seja interrompido, pois isto é

indicação de que o sinal SIRC recebido é de 15 ou 20 bits, ou então que foi uma recepção enganosa.

Para averiguar os resultados da captura, o autor do projeto faz com que o programa gere uma saída em texto, cujas linhas têm o seguinte formato:

- Um comando cNNNNNNNN
- Um ID de dispositivo dNNNNNNNN
- Um conjunto de bits extras xNNNNNNNN, que aparecerá apenas para os sinais de 20 bits
- A quantidade de bits NN, que pode assumir o valor 12, 15 ou 20

Nas *strings* do comando, ID de dispositivo e bits extras, "N" representa um bit, que assim sendo, pode ter os valores 0 ou 1. Há ainda a possibilidade de, no lugar dos dados descritos acima, as linhas geradas pelo programa serem compostas do caractere "S" ou "D", o primeiro indicando erro no pulso inicial (*start*) e o segundo erro nos pulsos de dados (*data*).

A descrição completa do projeto pode ser encontrada no link <http://picprojects.org.uk/projects/sirc/> e no documento PDF disponível para acesso na mesma página. Os links foram acessados para a utilização neste documento em Fevereiro de 2015.

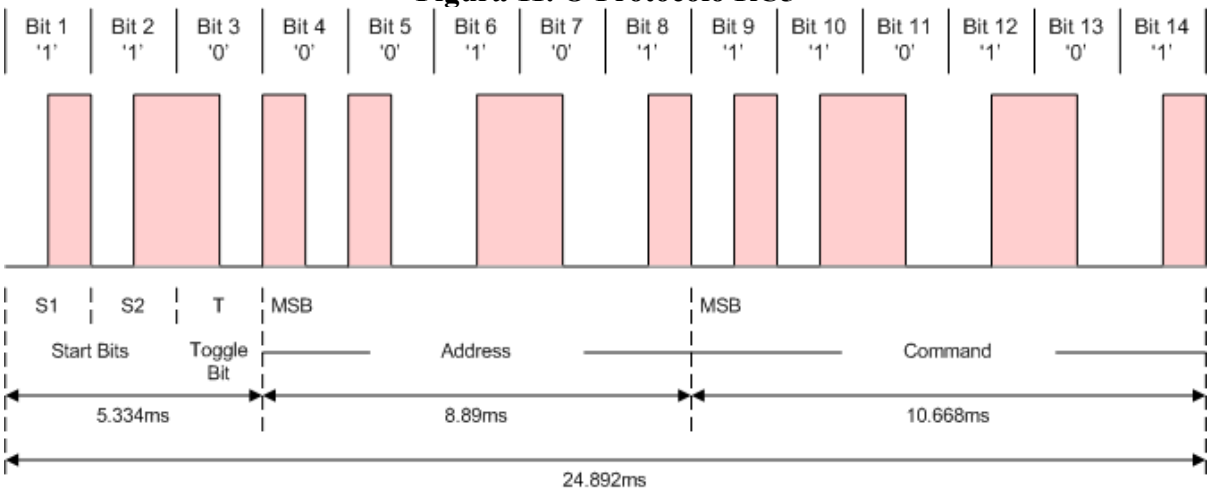
2.5.2 Decodificador de Códigos no Protocolo RC5

O protocolo RC5 utiliza codificação Manchester, que se baseia nas transições de estado do sinal para discernir entre um bit 0 e um bit 1. Isto significa que, diferentemente do protocolo SIRC, no qual o fator determinante é a duração do pulso, todos os pulsos em uma transmissão RC5 possuem a mesma duração, que é de 889µs a uma frequência de 36 kHz. Um "0" lógico, ou bit 0, é representado por um pulso *ON* de 889µs seguido de um pulso *OFF* (ausência de sinal) de mesma duração. Um "1" lógico, ou bit 1, é representado por um pulso *OFF* de 889µs seguido de um pulso *ON* de mesma duração. Assim sendo, o tempo total de transmissão de cada pulso é de 1778µs, ou 1.78 ms.

A mensagem IR enviada quando um botão de um controle remoto que utiliza este protocolo é pressionado possui 14 bits e é composta de dois bits de início (*start*), ambos 1, um bit *toggle* (T), que é invertido cada vez que um botão é solto e pressionado novamente, 5 bits que representam o endereço do dispositivo sendo controlado e 6 bits representando o comando. O bit *toggle* é utilizado para distinguir entre um botão sendo pressionado repetidamente ou sendo mantido pressionado, sendo que neste último caso, a mensagem é repetida a cada 114ms, com o valor do bit *toggle* sendo mantido inalterado. A interpretação deste recurso de repetição automática fica por conta do software instalado no dispositivo receptor. Neste protocolo, os dados são enviados a partir do bit mais significante[22]. A Figura

11 mostra uma representação gráfica do protocolo RC5.

Figura 11. O Protocolo RC5



Disponível em http://techdocs.altium.com/sites/default/files/wiki_attachments/212920/RC5MessageFrame.png, acessado em Fevereiro de 2015

A solução apresentada para detectar os códigos RC5 utiliza como componentes o microcontrolador de baixo consumo MSP430F1121, o regulador de voltagem TPS77033, um inversor SN74AHC1G04 e um sensor infravermelho TSOP1840SS3V, integrados em um circuito que utiliza comunicação serial através de uma interface RS232. A detecção do sinal é feita da seguinte maneira:

1. Dois registradores são utilizados: IRData (R6) recebe os pacotes IR e IRBit (R7) é usado como um contador temporário para rastrear os dados IR enquanto são recebidos. As variáveis BIT_50, que é metade da duração de um bit RC5 (que possui 1.78 ms de duração) e BIT_75, que é três quartos da duração de um bit RC5, são definidas. Para a contagem de tempo, é usada a implementação Timer_A3, da controladora.

2. Uma sub-rotina permite que a registradora de captura/comparação CCR1 capture a registradora de contagem de Timer_A3 (TAR) e requisite uma interrupção na queda do sinal (novamente pode-se ver o conceito de *falling edge*), detectada pelo sensor IR. Nesta transição de sinal, TAR é capturada e automaticamente armazenada em CCR1 e uma interrupção é requisitada.

3. O software determina se a interrupção foi disparada por uma captura ou comparação. Como o evento causador da primeira interrupção foi uma captura, BIT_75 (¾ do comprimento de um bit) é adicionada diretamente a CCR1, que registra o tempo exato em que o sinal de saída do sensor IR caiu no meio do primeiro bit.

4. Como a captura ocorreu no meio do primeiro bit, adicionar ¾ do comprimento de um bit irá efetivamente deslocar CCR1 para o meio da primeira metade do próximo bit. CCR1 é então configurada pelo software para o modo comparação.

5. A próxima interrupção está agora programada para o meio da primeira metade do segundo bit *start*. Quando a comparação de CCR1 ocorre, o nível lógico presente na porta de entrada e saída (*I/O port*) P1.2 da controladora será gravada no circuito síncrono de entrada de captura e comparação (SCCI) da registradora, que proporciona a funcionalidade de permitir que o hardware de CCR1 capture e armazene o nível lógico em P1.2 com o tempo exato gerado por Timer_A3, independentemente de outras atividades do processador (CPU). Os dados serão lidos de SCCI bit a bit após o evento de comparação e armazenados na registradora IRData.

6. Após cada bit ser recuperado, CCR1 é reconfigurada para capturar tanto nas quedas de sinal quanto em suas ascensões (*rising edge*). A próxima captura ocorrerá e irá se sincronizar na transição da metade do próximo bit. Para garantir que o pacote de dados está sendo decodificado de forma apropriada, a interrupção em CCR2 também é habilitada e carregada com um valor de metade do comprimento de bit da comparação armazenada em CCR1. A próxima transição de bit recebida deve ocorrer em aproximadamente ¼ do comprimento de bit se o pacote está sendo decodificado corretamente. Em uma captura normal da transição do sinal ocorrente em CCR1, a interrupção em CCR2 é limpa por software. Se CCR1 não faz uma captura normal e CCR2 não é limpa, CCR2 irá causar um *reset* no decodificador, assumindo um erro de transbordamento. Em operação normal, o balanço dos bits é recuperado e recebido em IRData.

Este projeto também possui a funcionalidade de decodificação de sinais SIRC, porém, como o projeto descrito na seção anterior já aborda este protocolo, este recurso do projeto atual será omitido, para evitar redundâncias e manter a brevidade do texto. Uma descrição detalhada do projeto pode ser encontrada no documento PDF disponível em <http://www.ti.com/lit/an/slaa134/slaa134.pdf>. O acesso ao link para a utilização neste documento foi feito em Fevereiro de 2015.

2.5.3 Aplicativos e Acessórios Relacionados

Serão brevemente apresentados nesta seção dois aplicativos comerciais que podem ser utilizados em conjunto com seus acessórios em dispositivos móveis para a finalidade de controlar aparelhos eletrônicos, de forma similar a este projeto, com o qual uma breve comparação será feita, para fins de análise.

- **iRule**

Este é um aplicativo de controle remoto universal compatível com iPhone, iPad, iPod Touch e Android que pode controlar dispositivos controlados por IP, conexão serial (RS-232) e por sinais infravermelho, através uma rede Wi-fi. O aplicativo está disponível na página do fabricante em duas versões, básica e avançada, a segunda possuindo alguns recursos adicionais além de permitir ao usuário a criação de mais controles virtuais e de registrar mais

dispositivos para serem controlados. Por um preço, é possível adicionar ainda mais controles e dispositivos. A interface gráfica dos controles virtuais é criada pelo usuário através de um aplicativo online que pode ser utilizado para adicionar componentes aos controles através de *drag-and-drop* (arrastar e soltar) e armazena as configurações em nuvem. O controle de dispositivos por conexão serial e IR é feito com o auxílio de acessórios vendidos separadamente, que se conectam à rede por cabo ou Wi-fi e são os responsáveis por traduzir os comandos do aplicativo para o formato adequado[23].

Se comparado ao oferecido por este projeto, pode-se perceber algumas vantagens e também desvantagens com relação ao aplicativo. Como vantagens, tem-se a compatibilidade com múltiplas plataformas móveis, a facilidade de criação de controles virtuais através de uma interface de mecânica simples e a possibilidade de controlar dispositivos através de mais de um protocolo de comunicação, recursos que a versão atual deste projeto não possui. Como desvantagens, há o fato de que apenas parece ser possível criar e armazenar controles virtuais através do uso do aplicativo online, sem uma solução local oferecida, o que talvez tornasse o programa inutilizável no caso de não haver uma conexão de rede. Mais uma desvantagem se dá no fato de ser necessário pagar por um ou mais acessórios separados e também para aumentar o número de controles virtuais e dispositivos que se pode armazenar, que, na solução oferecida por este projeto, é ilimitado.

- **Zmart**

Este sistema inclui um aplicativo disponível para as plataformas iOS e Android e um pequeno acessório que pode ser conectado à saída de áudio do dispositivo móvel que converterá os comandos selecionados no aplicativo para sinais IR. Possui um banco de dados de mais de 200,000 dispositivos eletrônicos compatíveis, compreendendo 95% das marcas no mercado e pode aprender os códigos de dispositivos que não estejam neste banco. O sistema é promovido no site da fabricante como “o menor controle remoto universal do mundo” e como sendo muito poderoso e fácil de usar, além de ser leve e estar disponível por um preço muito acessível[24].

Mais uma vez, ao comparar a solução com o produto deste projeto, é possível notar vantagens e desvantagens. Uma vantagem é que, assim como no aplicativo anterior, existe a compatibilidade com mais de uma plataforma. Outra é que já existe um vasto banco de dados de dispositivos que pode ser consultado para que se possa evitar o trabalho de passar pela detecção de cada código de um controle individualmente, processo que pode levar alguns minutos. As características do aplicativo mencionadas acima não estão presentes na solução apresentada por este trabalho em sua versão atual. Já as desvantagens, uma seria o fato de que é possível utilizar o aplicativo apenas em plataformas móveis, não existindo uma versão que se possa usar para controlar os aparelhos eletrônicos através de um computador ou notebook. Uma outra desvantagem é que também não parece haver uma opção para controlar aparelhos via rede, que neste projeto, mesmo estando limitada apenas à rede sem fio local do ambiente na versão atual e também requerer um acessório (a placa Arduino), é oferecida.

3 Especificação do Sistema

Para alcançar os objetivos descritos no primeiro capítulo, foi criado um sistema que contém um aplicativo para computador, um aplicativo para dispositivos Android e um circuito que utiliza uma placa microcontroladora conectada a um LED infravermelho e um sensor infravermelho. No chip da microcontroladora roda o software responsável por detectar os sinais IR e criar representações dos mesmos em formato *string* (sequência de caracteres), que será enviada ao aplicativo de computador – o aplicativo central do projeto – através de uma conexão serial.

Nas seções a seguir serão abordados os conceitos principais de cada um destes aplicativos e como os mesmos operam. O foco será no aplicativo central, que é não apenas o mais complexo dos programas mas também a base para toda a operação do sistema. Os outros aplicativos serão abordados de forma mais superficial.

3.1 Descrição Geral do Problema (Mini-mundo)

Com a evolução da tecnologia e a redução dos preços de aparelhos eletrônicos, é cada vez mais comum que as pessoas tenham acesso a mais e mais destes aparelhos no ambiente de seus lares, e muitos destes vêm da loja contendo o controle remoto que deve ser utilizado para passar comandos aos mesmos. O problema que isto cria é que quanto mais dispositivos eletrônicos se possui, maior será o número de controles remotos espalhados pela casa, o que pode gerar uma série de inconveniências, entre elas: o sumiço de algum controle, quando algum destes estiver quebrado ou danificado de algum modo e quando não há pilhas ou baterias novas para substituir as usadas ou quando estas estejam carregando. Durante qualquer um destes casos, o usuário ficaria impossibilitado de controlar eficientemente seus aparelhos, a não ser que utilizasse diretamente os botões existentes nos mesmos, o que pode ser bem incômodo quando se está assistindo a um filme ou ouvindo música, por exemplo, situações em que às vezes é necessário passar vários comandos ao aparelho, e com certa frequência (aumentar volume ou passar para a próxima faixa do CD são apenas alguns exemplos). Um outro problema é que muitas vezes é necessário controlar diversos aparelhos relacionados e/ou próximos fisicamente, como em um ambiente de lazer, onde pode haver um aparelho de televisão conectado a um reprodutor de BluRay e a um sistema de som, por exemplo. Se for necessário que o usuário utilize o controle próprio de cada um destes aparelhos para executar funções em cada um deles, uma constante troca de controles remotos precisará acontecer, o que pode ser uma situação indesejável para algumas pessoas.

Para resolver os problemas descritos, atualmente no mercado existem as soluções conhecidas como controles remotos universais, que são controles capazes de operar diversos dispositivos, independentemente de suas marcas ou modelos, alguns permitindo inclusive que o próprio usuário o programe e o faça aprender os comandos de dispositivos inicialmente incompatíveis. Enquanto muito conveniente, esta solução ainda apresenta algumas

desvantagens que podem vir a aborrecer um usuário comum, como a dificuldade de uso, que pode se mostrar devido ao fato da grande maioria dos modelos não possuírem uma tela ou nenhum tipo de interface gráfica para que o usuário tenha uma confirmação visual de fácil compreensão sobre o que está fazendo, além de serem muito caros ou simplesmente limitados.

Visando superar algumas destas limitações, este projeto propõe a solução de criar um sistema de controle de aparelhos eletrônicos que seja não apenas fácil de se usar mas também inclua alguns recursos não existentes em controles remotos universais convencionais, possivelmente proporcionando ao usuário uma experiência ainda melhor.

3.2 Especificação dos Requisitos

Aqui serão descritos os requisitos funcionais e não-funcionais do sistema, que descrevem, respectivamente, as funcionalidades do mesmo e suas restrições, assim como outros aspectos críticos de seu desempenho. Ao final, serão apresentadas algumas regras de negócio, que estabelecem normas a serem seguidas pelo sistema.

3.2.1 Requisitos Funcionais do Sistema

Os requisitos funcionais para a elaboração deste projeto de sistema foram:

- O sistema deve permitir que o usuário detecte, nomeie e armazene comandos de controles remotos normais;
- O sistema deve permitir que o usuário edite os controles remotos armazenados, podendo renomear os mesmos ou seus códigos, assim como excluir estes últimos;
- O sistema deve permitir que o usuário cria controles remotos virtuais na tela do computador que possam enviar comandos para aparelhos eletrônicos sem a necessidade da participação de seus controles originais;
- O sistema deve permitir que o usuário exporte os controles virtuais criados para um dispositivo móvel Android e o utilize através da tela deste dispositivo;
- O sistema deve permitir que o usuário mapeie comandos de um controle remoto armazenado para os comandos de outros controles, podendo assim controlar um ou vários aparelhos eletrônicos com o mesmo.

3.2.2 Requisitos Não-funcionais do Sistema

Os requisitos não-funcionais para a elaboração deste projeto de sistema foram:

- O sistema deve estabelecer um tempo limite para que as informações sobre comandos sejam obtidas;
- O sistema não deve fazer alterações no banco de dados caso alguma parte da transação tenha falhado;

- O sistema sempre deve fornecer opções para que o usuário interrompa a operação atual;
- O sistema deve exibir telas que deixem claro para o usuário qual operação está sendo realizada.

3.2.3 Regras de Negócio

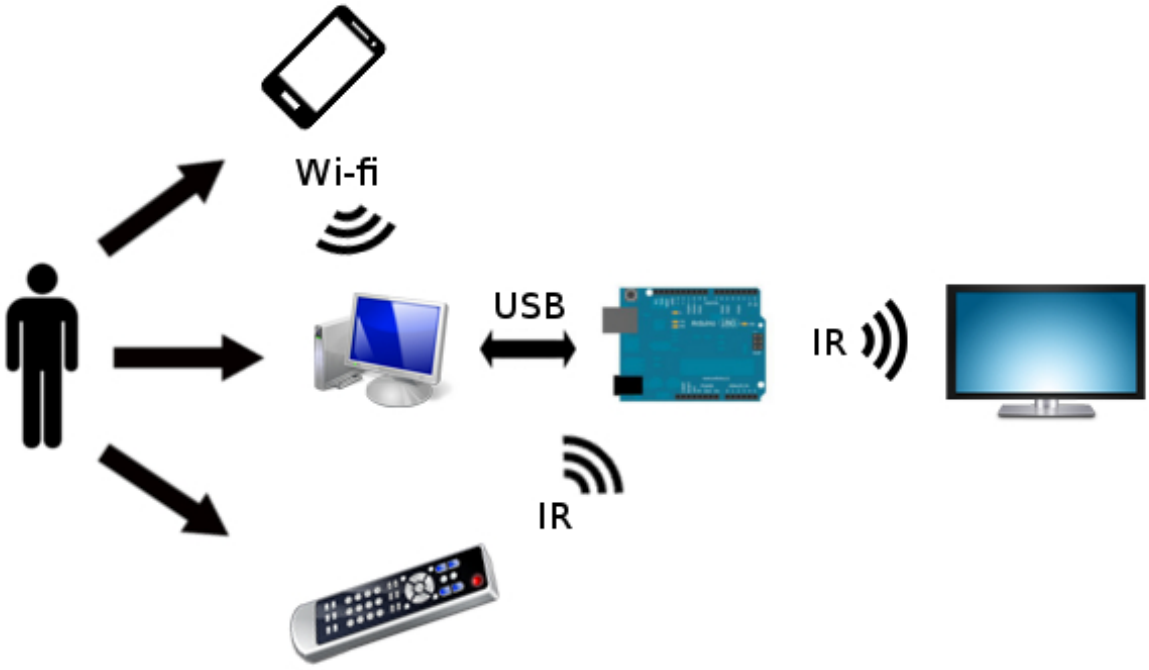
As regras de negócio da elaboração deste projeto de sistema foram:

- Todos os comandos e interfaces de controles armazenados devem estar associados a algum controle remoto;
- Todos os comandos e controles remotos devem possuir um nome;
- Os códigos de um controle remoto não devem poder ser mapeados para outros códigos do mesmo controle.

3.3 Visão Geral da Solução

O sistema contará com a participação de diversas entidades que interagirão entre si durante a execução de cada tarefa. Na Figura 12 é possível observar quais são as interações que ocorrem e de quais formas.

Figura 12. Visão geral da solução



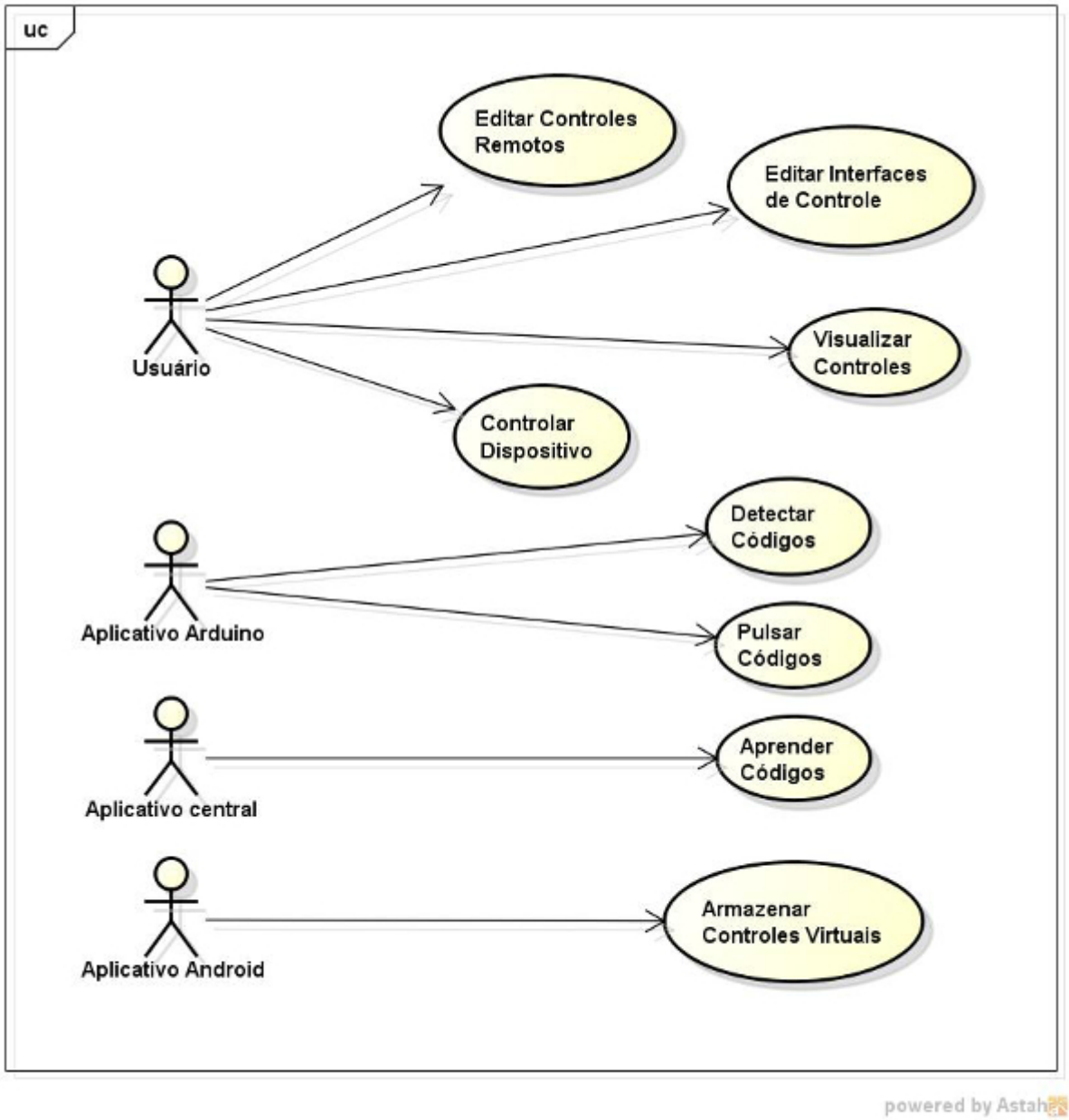
Através da imagem acima é possível perceber que o usuário pode interagir diretamente com três tipos de dispositivos em prol de alcançar a finalidade de controlar um aparelho eletrônico: o computador, que se comunica com a placa controladora através de conexão USB para que os dados sobre os comandos dos controles possam ser enviados e recebidos; O controle remoto físico, que envia sinais IR à controladora, que por sua vez trata de reproduzir os códigos mapeados, enviando-os ao aparelho e por fim, com um dispositivo móvel, que faz a comunicação com o computador através de uma conexão de internet sem fio (Wi-fi) e solicita que determinados códigos sejam pulsados.

Através deste conjunto de interações que pode acontecer em sequências diferentes ou omitindo algumas etapas dependendo de qual operação o usuário está a realizar, o sistema é capaz de enviar comandos a um aparelho eletrônico da mesma forma que um controle remoto convencional o faria, consequentemente exercendo controle sobre o mesmo.

3.4 Modelo de Casos de Uso

Esta seção descreve a forma como as interações ocorrem neste sistema, sejam elas entre o usuário e um aplicativo ou dos próprios aplicativos entre si. O digrama de casos de uso do sistema pode ser visto na Figura 13.

Figura 13. Diagrama de Casos de Uso

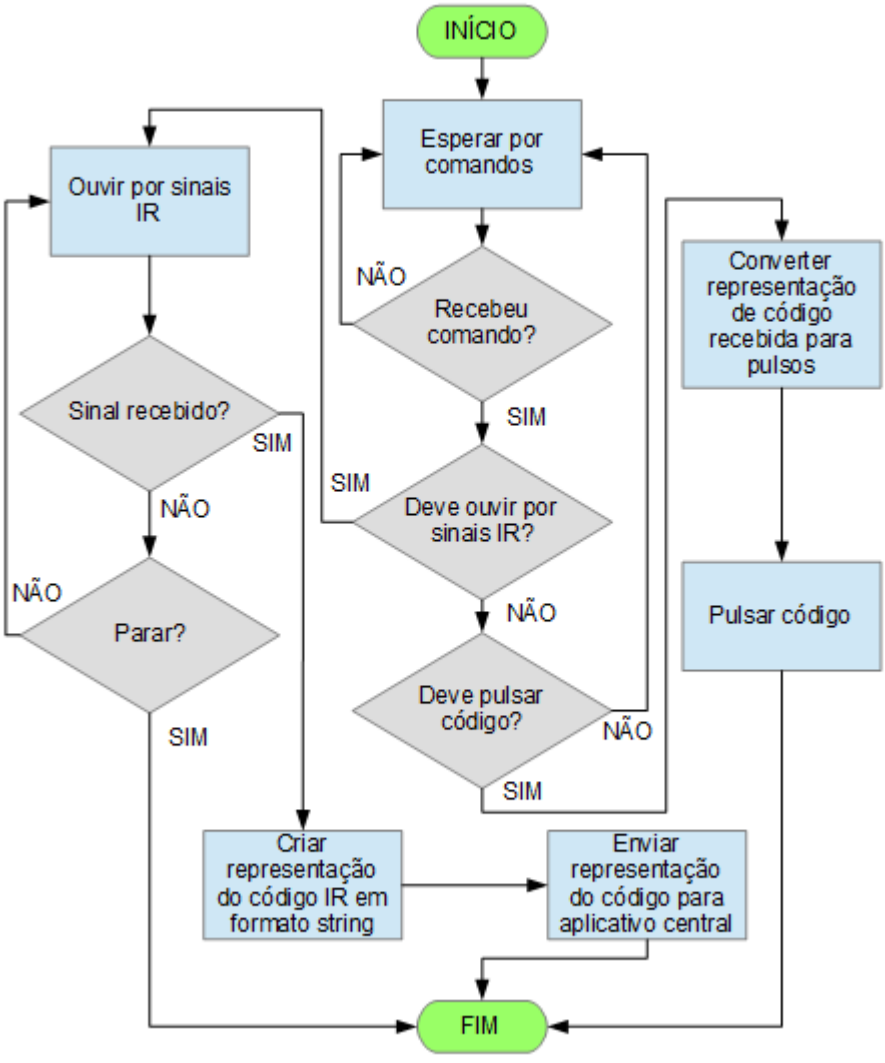


Como este sistema está dividido em três aplicativos, o modelo de casos de uso também será apresentado de forma segmentada, utilizando uma abordagem onde os aplicativos são descritos um de cada vez e os casos de uso referentes a suas funcionalidades são referenciados em meio a uma explicação sobre as mesmas. Para descrever os fluxos de operações que ocorrem nessas funcionalidades, uma abordagem visual será utilizada, através de diagramas de fluxo, que podem incluir mais de um caso de uso por diagrama. Isto é feito por causa da natureza do sistema, onde uma única funcionalidade pode incluir a participação de vários ou até mesmo todos os atores.

3.4.1 O Programa de detecção/execução de códigos IR

A tarefa inicial do programa deve ser entrar no modo de espera por comandos, durante o qual ele irá constantemente checar pela existência de instruções enviadas pelo aplicativo central, que pode comandá-lo para iniciar a detecção de códigos, pulsar um código no LED infravermelho ou voltar ao estado de espera. O diagrama de fluxo na Figura 14 ilustra a forma como este programa deve se comportar.

Figura 14. Fluxo de eventos do programa de detecção de códigos



Como se pode observar no diagrama, o programa sempre deve começar checando se algum comando vindo do aplicativo central foi recebido. Os comandos possíveis são: “ouvir por sinais IR”, “pulsar código” e “voltar ao estado de espera”. O primeiro e o segundo comandos devem ser recebidos quando o programa está em estado de espera e o último quando ele está em modo de detecção de sinal, do qual deve sair quando isto acontecer.

Ao receber o comando “ouvir por sinais IR”, o programa precisa invocar uma função

dentro da qual um novo *loop* será executado. A cada iteração, este *loop* deve checar pela presença de sinal infravermelho sendo recebido pelo sensor e também se um comando para sair do modo de detecção foi recebido. No primeiro caso, as durações de cada pulso do código devem ser registradas e esta informação convertida para um formato textual, que será enviado de volta para o aplicativo central. No segundo, o *loop* deve ser interrompido para que a execução da função termine e mais nada deve ser feito, pois o aplicativo central irá se desconectar da controladora para possivelmente realizar outras tarefas, ou finalizar a sessão de uso do sistema. Este processo descreve o caso de uso Detectar Códigos.

Se o comando "pulsar código" for recebido, ele virá junto com uma representação textual do código a ser pulsado. Após receber esta representação, o aplicativo deve percorrer toda a cadeia de caracteres e usá-la para criar uma matriz (tabela) que contenha as representações numéricas das durações de cada pulso, obtidas através da informação recebida. O aplicativo então deve iterar por cada um dos elementos desta matriz e realizar a operação apropriada durante o tempo especificado, que será manter o LED aceso ou mantê-lo apagado, até a próxima mudança de estado. Este é o processo referente ao caso de uso Pulsar Códigos.

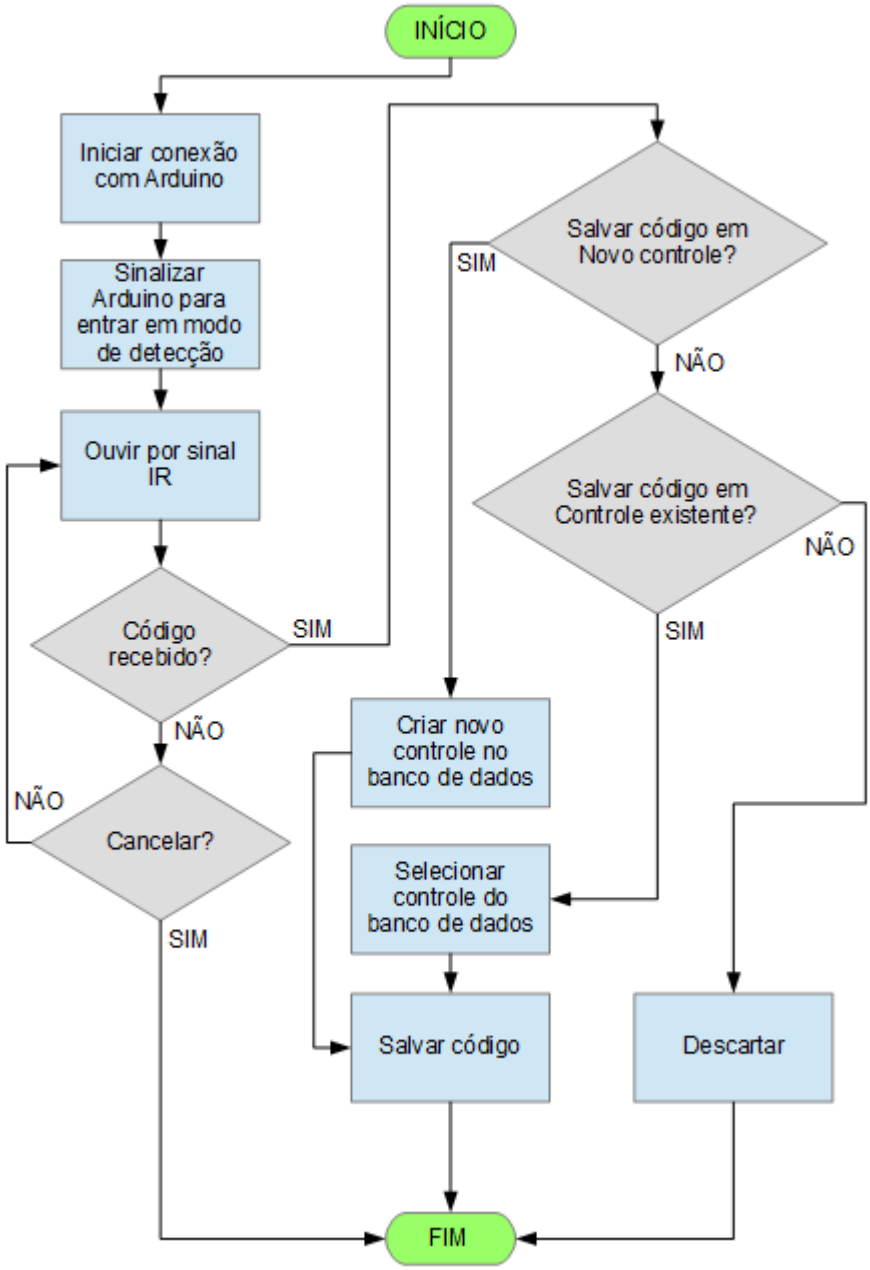
3.4.2 O Aplicativo Central

Este é o programa mais complexo dentre os utilizados no sistema, pois é nele que devem se iniciar todas as operações e também é o que conterá a interface gráfica através da qual o usuário terá sua experiência. As diversas funcionalidades deste programa são descritas logo abaixo.

- **Aprender Códigos**

Ao selecionar esta opção, o aplicativo central deve entrar em contato com a controladora e a colocar em modo de detecção de códigos. Quando um código é detectado, deve ser possível salvá-lo como parte de um controle remoto, que está associado a um dispositivo. Se não houver controles remotos salvos no banco de dados da aplicação, um novo controle pode ser criado. A Figura 15 mostra a sequência de procedimentos para o aprendizado de novos códigos.

Figura 15. Fluxo de eventos da operação de adição de novos códigos do aplicativo central



Quando um código é detectado, sua representação em formato texto deve ser armazenada no banco, associada a um dispositivo. Também deve ser possível descartar o código recebido e continuar ou sair do modo de detecção. Se o usuário escolher fazer com que o programa continue ouvindo por códigos, o processo deve se repetir quando um novo código é detectado. Estes procedimentos fazem referência tanto ao caso de uso Aprender Códigos, do aplicativo central, quanto ao caso Editar Controles Remotos, que apesar de ser uma

funcionalidade fornecida pelo aplicativo, é uma ação exercida pelo usuário, razão pela qual este se mostra como o ator associado ao caso.

- **Ver Controles**

Esta opção permite que o usuário acesse os controles remotos salvos no banco de dados. Se não houver controles salvos, um alerta deve ser exibido e a operação interrompida. Caso contrário, uma interface de visualização deve ser mostrada, onde se possa alterar estes códigos e também excluí-los, se assim for necessário. Adicionar novos códigos não será possível nesta interface, mas apenas através do modo de detecção de códigos. Este processo se refere ao caso de uso Visualizar Controles, que novamente, está atrelado ao ator Usuário, já que este é o responsável por de fato fazer a visualização.

- **Mapear Comandos**

Esta é a opção que permitirá que sejam criados mapeamentos dos códigos aprendidos pelo aplicativo central para outras interfaces de controle, o que constitui o ponto central do projeto. Na implementação atual, é possível utilizar três formas distintas de controle. A primeira é um controle remoto virtual, do qual cada botão, assim como em um controle remoto convencional, está associado a apenas um código. Estes controles virtuais porém, devem poder ser editados a qualquer momento e terem os códigos de seus botões alterados, assim como ter botões adicionados, modificados e removidos de acordo com a vontade do usuário.

A segunda forma é o controle através de dispositivo móvel, que utilizará o aplicativo para Android que é descrito mais adiante. Através deste aplicativo, deve ser possível utilizar os controles virtuais editados pelo usuário na tela de um dispositivo que utilize o sistema operacional em questão. Assim como na transmissão de códigos entre o aplicativo central e a controladora, uma representação em texto dos dados deve ser criada, para que a tarefa de transmitir os controles virtuais via rede seja simples. O aplicativo central deve preparar esta representação quando o usuário selecionar a opção “Exportar para Smartphone”, que existirá na interface de edição de controles virtuais. Ao receber a representação, o aplicativo no dispositivo móvel deve adicionar o controle remoto ao seu banco de dados e carregá-lo com a mesma aparência que ele tem no aplicativo central, de forma que cada botão esteja associado ao ID de um código.

A terceira forma não envolve interfaces virtuais, mas sim um controle remoto físico. Esta opção permitirá que o usuário associe os códigos do controle sendo editado para outros códigos existentes no banco, criando uma espécie de “tradução” de códigos. Isto significa que deve ser possível controlar um dispositivo que não é o alvo original do controle remoto em uso, assim como controlar diversos dispositivos diferentes com um único controle. Este tipo de mapeamento deve ser criado através da associação entre IDs de código.

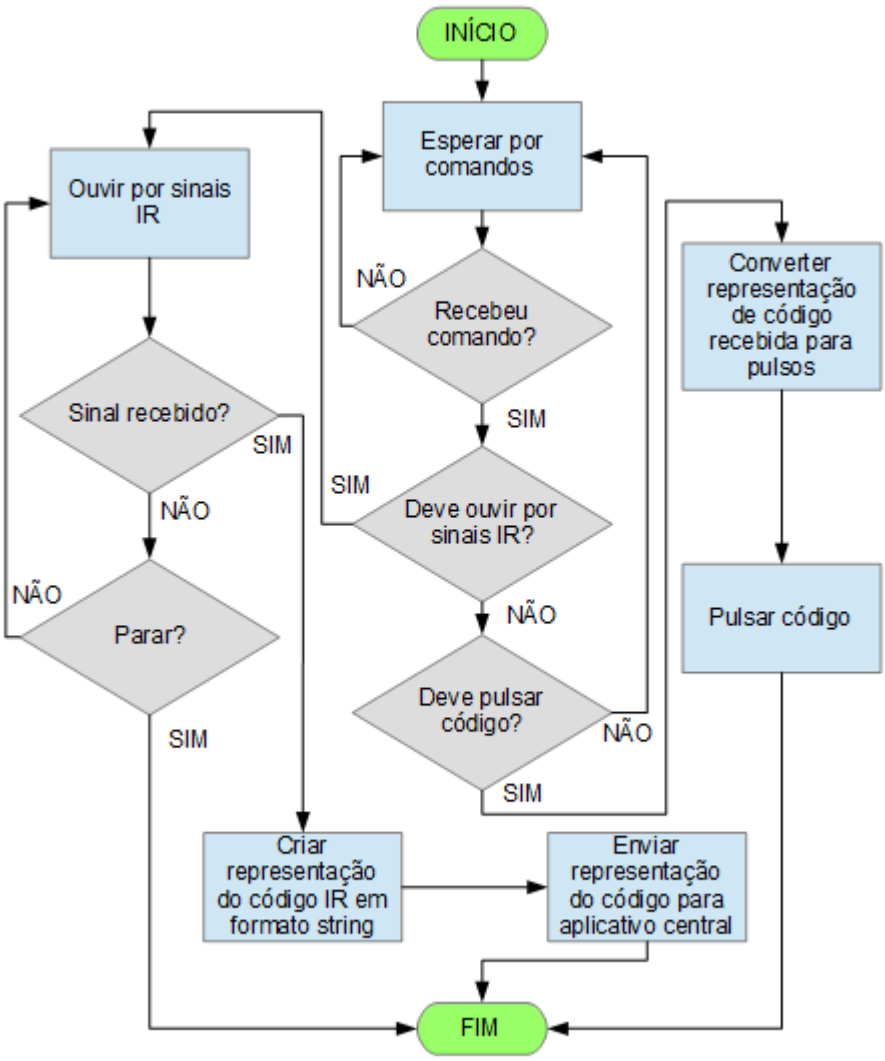
Esta funcionalidade está associada ao caso de uso Editar Interfaces de Controle, já que

é nela que o usuário – que outra vez aparece como o ator – poderá realizar esta tarefa.

- **Controlar Dispositivo**

Ao selecionar esta opção, o usuário poderá selecionar uma das interfaces de controle criadas na opção “Mapear Comandos” e colocá-la em uso. A Figura 16 mostra o fluxo de eventos que devem ocorrer quando esta opção é selecionada.

Figura 16. Fluxo de eventos do recurso de controle de dispositivos do aplicativo central



Uma tela que permite que uma interface de controle seja escolhida deve ser mostrada e ao selecionar uma delas, o usuário vê uma janela que indica que o modo de controle se iniciou. Se a interface sendo utilizada é um controle remoto virtual, ele deve ser carregado nessa janela e toda vez que um botão for clicado deve ser iniciada uma conexão com a controladora para enviar o código a ser pulsado. Se for o mapeamento entre códigos, essa janela deve exibir um alerta dizendo que o programa já está pronto para receber a entrada de

códigos IR. O controle remoto físico pode então ser utilizado e quando um código deste controle for recebido, o programa deve comparar seus pulsos com os pulsos dos códigos armazenados no banco de dados e obter o ID do código quando uma correspondência surgir. É então necessário identificar para qual ID de código este está mapeado e as informações sobre pulsos do código identificado são obtidas e enviadas para a controladora, que mais uma vez deve pulsar o código no LED. A qualquer momento, o usuário pode interromper o modo de controle, sendo que, se o modo atual for o de controle por mapeamento entre códigos, isto inclui sinalizar a controladora para que saia do modo de detecção de códigos.

Como o nome sugere, esta funcionalidade está relacionada ao caso de uso Controlar Dispositivo, que, mais uma vez, é uma ação do usuário apesar de ser um recurso do aplicativo central.

- **Utilizar Smartphone Como controle**

Esta funcionalidade é basicamente a mesma que a dos controles remotos virtuais, porém a interface com a qual o usuário interagirá será a tela do dispositivo móvel ao invés do próprio computador no qual o aplicativo central esteja rodando. Ao receber um ID de código via rede, o programa consultará o banco de dados para obter as informações sobre os pulsos do código, que devem então enviadas para a controladora em forma de texto e convertidas de volta para pulsos IR que serão reproduzidos no LED conectado ao sistema. Esta funcionalidade também está relacionada com o caso de uso Controlar Dispositivo, já que, apesar da forma de controle ser diferente, a finalidade é a mesma.

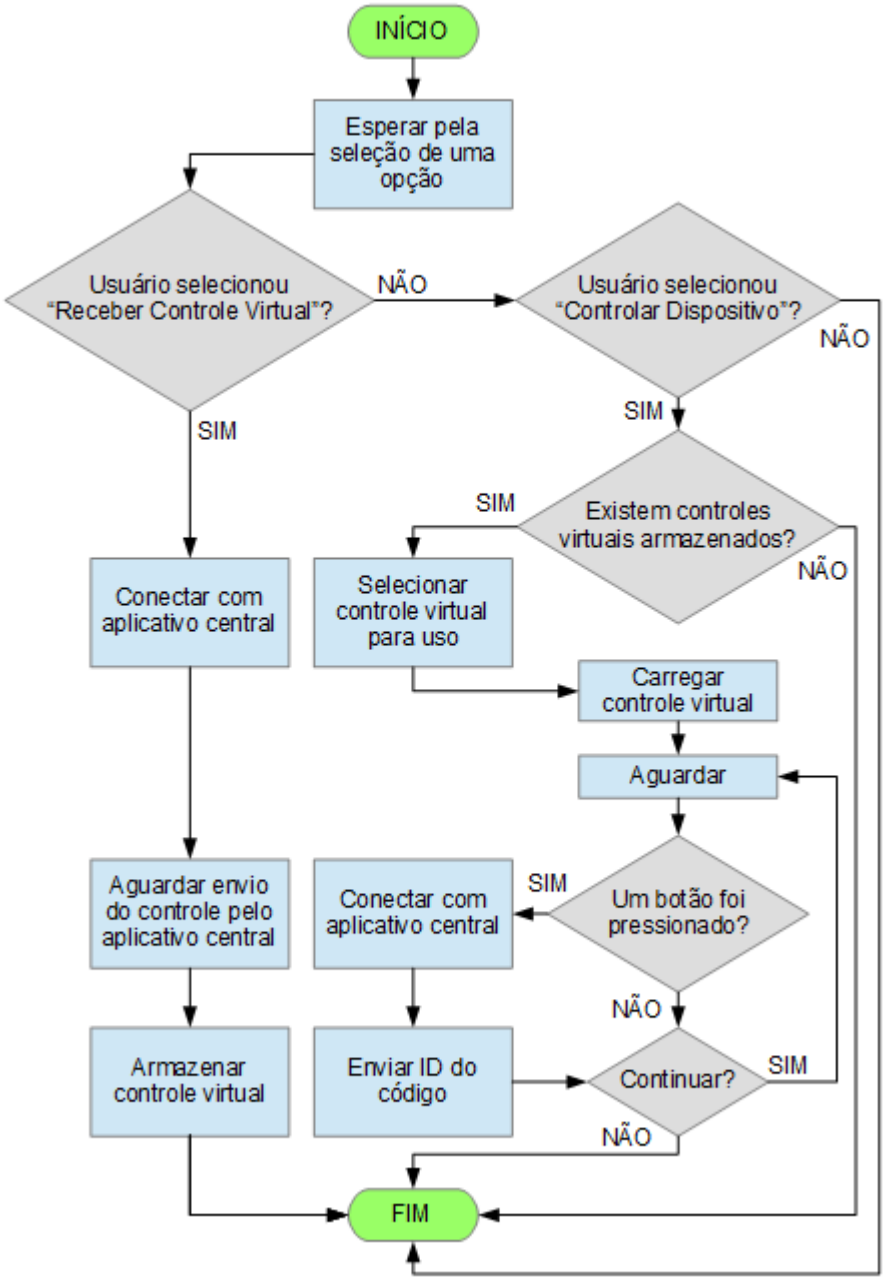
- **Configurar Porta COM**

Esta opção apenas deve exibir uma interface de configuração da porta de comunicação através da qual o aplicativo central irá se comunicar com a controladora. Uma lista das portas disponíveis no computador é mostrada e a porta correta deve ser selecionada pelo usuário. Estas portas normalmente são identificadas pela palavra “COM” seguida de um número e cabe ao usuário saber qual é a porta correta associada à controladora. Esta informação pode ser obtida facilmente acessando o gerenciador de dispositivos do sistema operacional do computador, que deve ser alguma versão do Windows.

3.4.3 O Aplicativo para Android

Neste aplicativo, existirão duas opções que podem ser selecionadas pelo usuário: receber novos controles virtuais e controlar um dispositivo utilizando aqueles já recebidos. A Figura 17 mostra o funcionamento deste aplicativo.

Figura 17. Fluxo de eventos do aplicativo para Android



Ao selecionar a primeira opção, uma conexão com o aplicativo central deve ser estabelecida e o dispositivo Android passará então a aguardar pelo envio da representação textual do controle remoto virtual. Ao terminar a recepção, o novo controle deve ser processado e adicionado à lista, para que possa então ser utilizado através da segunda opção. Este procedimento faz referência ao caso de uso Armazenar Controles Virtuais.

Se a segunda opção for selecionada, deve ser mostrada ao usuário uma tela onde ele possa escolher entre os controles virtuais armazenados, se houver algum. Após fazer sua

escolha, o controle virtual é carregado na tela do dispositivo e quando um de seus botões é pressionado, o ID do código associado a ele deve ser enviado ao aplicativo central através de uma conexão de rede. Este processo é então repetido enquanto o usuário continuar pressionando botões.

3.5 Modelo de Classes

O diagrama de classes mostra os tipos de objetos existentes em um programa e a forma como interagem. O diagrama apresentado a seguir é referente ao aplicativo central e mostra as classes mais importantes para o funcionamento deste aplicativo, que é a base do sistema. Deve-se observar que, em um programa desenvolvido utilizando Windows Forms (desenvolvimento gráfico de aplicações através de IDE) como este projeto, a maior parte das classes existentes – consequentemente onde está encontrada a maior parte do código-fonte – são classes geradas automaticamente quando se cria uma janela ("*form*", como é chamada uma janela internamente no programa). Isto significa que cada tela do programa possui uma classe correspondente, que herda da classe base Form, disponibilizada pelo *framework* .NET. Estas classes foram omitidas do diagrama pois o mesmo ficaria extremamente extenso caso isto não fosse feito.

3.5.1 Dicionário de Classes

Aqui, são apresentadas de forma textual as classes existentes no aplicativo central do sistema. Mais uma vez, classes de formulário serão omitidas pelo bem da simplicidade.

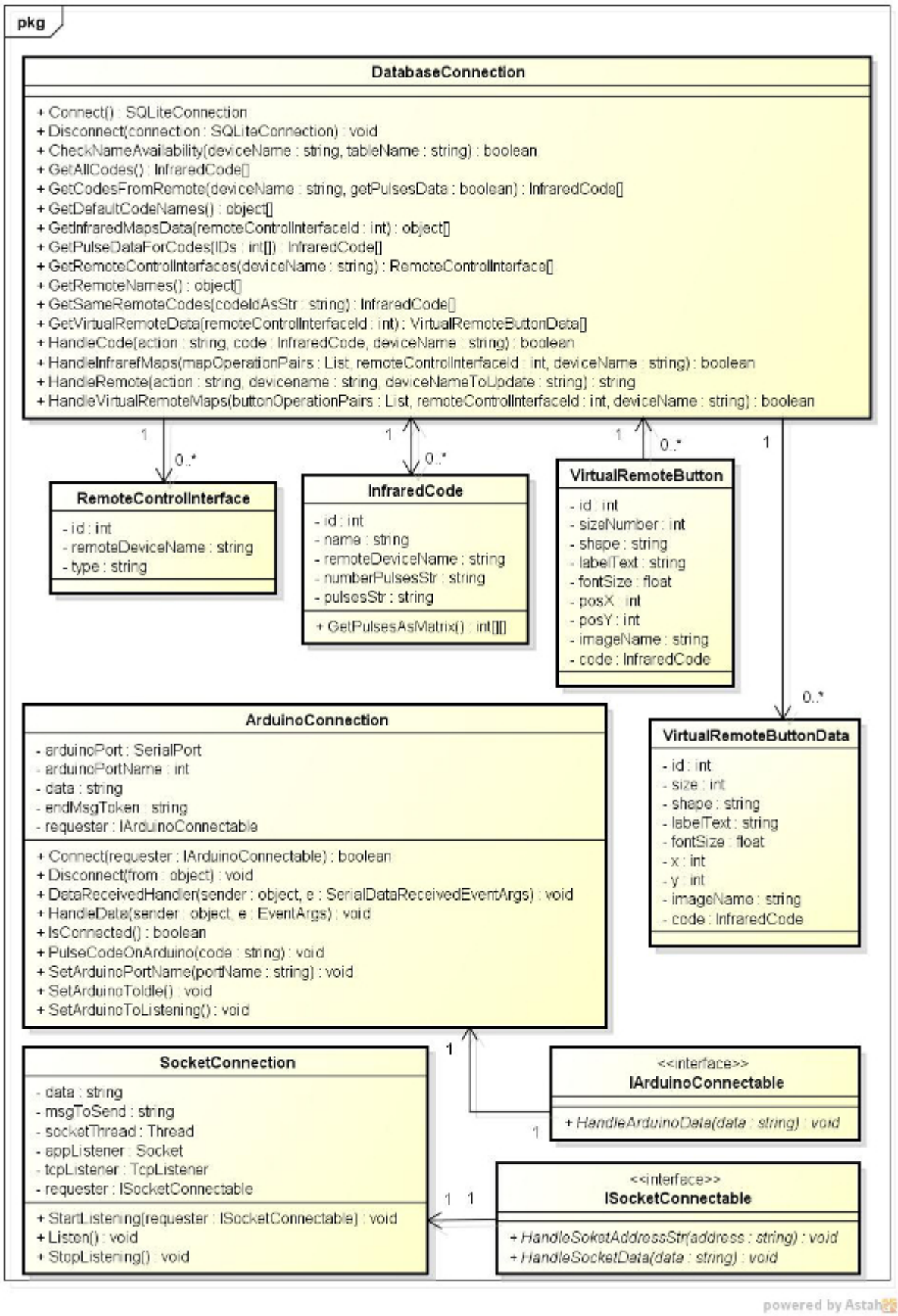
- DatabaseConnection – Realiza a conexão com o banco de dados SQLite de forma isolada e faz as devidas operações no banco, como consulta, adição, alteração e exclusão de registros.
- ArduinoConnection – Realiza a conexão com a microcontroladora Arduino de forma isolada e para que possa haver a comunicação entre esta e o aplicativo central de modo que as informações sobre códigos infravermelhos possam ser transmitidas.
- SocketConnection – Realiza a conexão com o aplicativo para Android de forma isolada através da interface de *sockets*, estabelecendo a comunicação entre os aplicativos que permite que dados sobre códigos infravermelhos e controles virtuais sejam transmitidos.
- IArduinoConnectable – Interface que define um método para o tratamento de dados recebidos através de uma conexão com a placa Arduino. Deve ser implementada por quaisquer classes que requisitem esta conexão.
- ISocketConnectable – Interface que define um método para o tratamento de dados recebidos através de uma conexão com o aplicativo para Android via *sockets*. Também define um método que serve para receber os dados sobre o endereço da conexão

(geralmente utilizados para atualizar a interface gráfica, mostrando ao usuário o endereço ao qual ele deve ser conectar). Deve ser implementada por quaisquer classes que requisitem esta conexão.

- RemoteControlInterface – Representa uma interface de controle criada pelo usuário. Seu atributo "*type*" (tipo) é o responsável por definir qual é o tipo da interface. Na atual implementação, existem os tipos "virtual_remote" e "infrared", que representam respectivamente controles remotos virtuais e mapeamentos entre códigos infravermelhos.
- InfraredCode – Representa uma instância de um código infravermelho. Esta classe contém as informações sobre os códigos infravermelhos que representam comandos de controles remotos e um objeto dela é criado toda vez que é necessário manipular tais códigos no programa, seja para a criação de listas, para transmitir dados sobre eles (como informação de pulsos) ou qualquer outro propósito.
- VirtualRemoteButton – É uma classe que herda da classe Button fornecida pelo *framework* .NET. Esta classe implementa alguns atributos adicionais que otimizam a funcionalidade destes botões para sua adição em controles virtuais e a manipulação de comandos infravermelhos, como o atributo "*code*", que é uma referência à classe InfraredCode e é responsável por associar um código a um botão.
- VirtualRemoteButtonData – Esta classe possui basicamente os mesmos atributos que a classe VirtualRemoteButton, tendo como principal diferença o fato de que a mesma não herda da classe Button, sendo uma classe independente. Esta classe existe para que seja possível manipular dados sobre botões de controles virtuais sem que seja necessário instanciá-los, já que são elementos gráficos e nem sempre é necessário desenhá-los na tela quando suas informações precisam ser acessadas.

A Figura 18 mostra o diagrama de classes do aplicativo central.

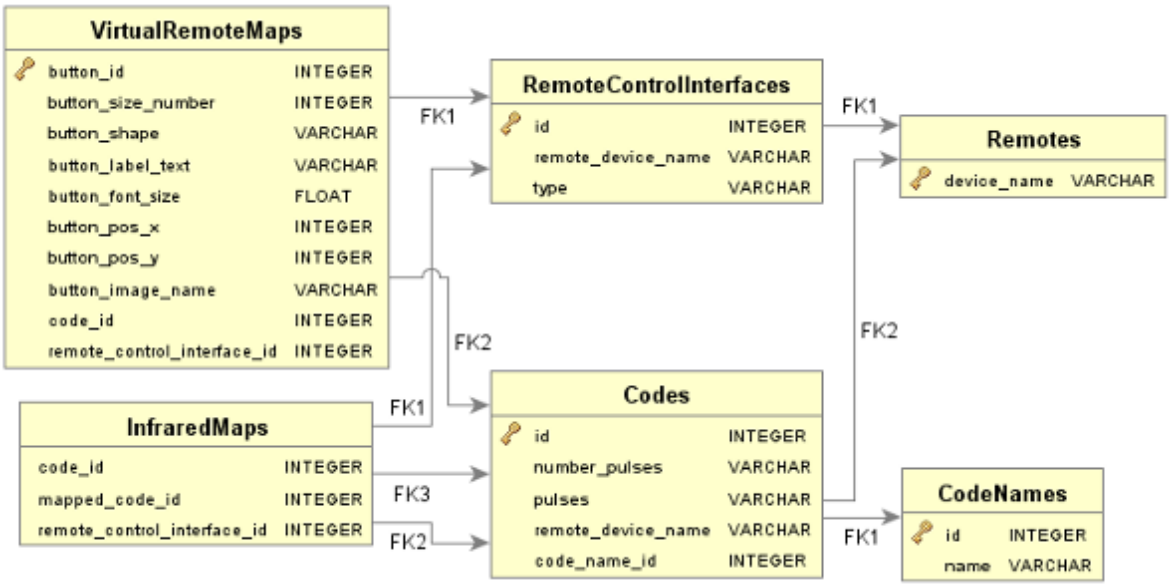
Figura 18. Diagrama de Classes



3.6 Projeto de Banco de Dados

O aplicativo central utiliza um banco de dados local SQLite que armazena no disco rígido do computador onde roda todas as informações necessárias, como controles remotos disponíveis, códigos IR armazenados e interfaces de controle editadas pelo usuário. A Figura 19 mostra o projeto físico do banco de dados utilizado na forma de um diagrama Entidade Relacionamento (ER).

Figura 19. Projeto físico do banco de dados do aplicativo central



A tabela “Codes” é a tabela central, que armazena os dados relevantes sobre os códigos IR, como número de pulsos, as durações dos pulsos e o nome do dispositivo cujo controle remoto os códigos em questão fazem parte. Este último é uma chave estrangeira apontando para a tabela “Remotes”, que tem como função armazenar os nomes de tais dispositivos, que devem ser únicos. Por este motivo e pelo fato de não haver outros dados na tabela, esses nomes também desempenham o papel de chave primária na mesma. Todas as outras tabelas possuem um ID numérico incrementado automaticamente na adição de um novo registro, com exceção da tabela “InfraredMaps”, que armazena os mapeamentos entre códigos IR. Estes mapeamentos não precisam ser únicos, basta que possuam uma referência à interface de controle a qual fazem parte, razão pela qual a tabela não possui uma chave primária. A tabela “RemoteControlInterfaces” armazena as interfaces de controle editadas pelo usuário, enquanto a tabela “VirtualRemoteMaps” trata de armazenar todos os dados sobre os botões adicionados aos controles remotos virtuais criados pelo usuário, como seu tamanho, posição na interface gráfica do controle, texto e todas as demais informações que sejam relevantes para que o controle virtual seja mostrado corretamente na tela quando é carregado. Por último, há a tabela “CodeNames”, que contém os nomes dados pelo usuário aos códigos IR armazenados. É importante observar que esta tabela contém alguns registros que são adicionados logo que o banco é criado, correspondentes a alguns nomes padrões de

comandos, como “Power”, “Volume +”, “Volume -”, “Channel +”, “Channel -”, entre outros nomes comuns que o usuário pode dar aos códigos detectados. Há ainda uma tabela extra chamada “sqlite_sequence”, criada automaticamente pelo *framework* do SQLite para manter a contagem dos valores das chaves primárias numéricas, que são incrementados automaticamente. Esta tabela não possui relação direta com as outras e não é relevante na descrição do funcionamento do projeto, por isto foi omitida do diagrama.

O banco de dados é acessado através da classe “DatabaseConnection”, que deve ter uma instância inicializada em todo objeto (por exemplo, janelas) que envolvam operações no banco. Um objeto desta classe é responsável por iniciar e terminar a conexão com o banco e também por todas as alterações no mesmo, feitas através de comandos na sintaxe da linguagem estruturada de consultas (SQL). Esta classe possui vários métodos de diversos propósitos, como lidar com códigos, controles virtuais, mapeamentos e quaisquer outras operações que precisem acessar ou alterar o banco de dados, que recebem como argumentos os valores necessários e realizam internamente tais operações, sem que os objetos requisitantes precisem fazer qualquer interação direta com o banco.

4 Elaboração da Solução

A partir deste ponto, será apresentada a forma como a solução foi desenvolvida de modo mais aprofundado do que o que foi visto na seção “Especificação do Sistema”, desta vez dando destaque aos elementos práticos e técnicos do projeto, como a estrutura do sistema ligado à microcontroladora, algumas das diversas telas da interface gráfica de usuário dos aplicativos desenvolvidos, o projeto do banco de dados utilizado e quaisquer outros aspectos relevantes.

Este projeto consiste nas seguintes partes:

- Um circuito que utiliza um LED infravermelho TIL32 5 milímetros e um sensor infravermelho IRM3638 conectados a uma placa microcontroladora Arduino Uno R3, que se comunica com o aplicativo central do projeto através de uma conexão serial por cabo USB tipo A/B.
- Um aplicativo para um computador com sistema operacional Windows, escrito na linguagem C# utilizando o *framework* .NET na versão 4.5, embora a versão alvo do framework para a qual o aplicativo foi compilado seja a versão 3.5, para maior retrocompatibilidade.
- Um aplicativo para dispositivos Android, que utiliza a linguagem Java e tem como alvo a versão 21 do kit de desenvolvimento (SDK) da plataforma Android.

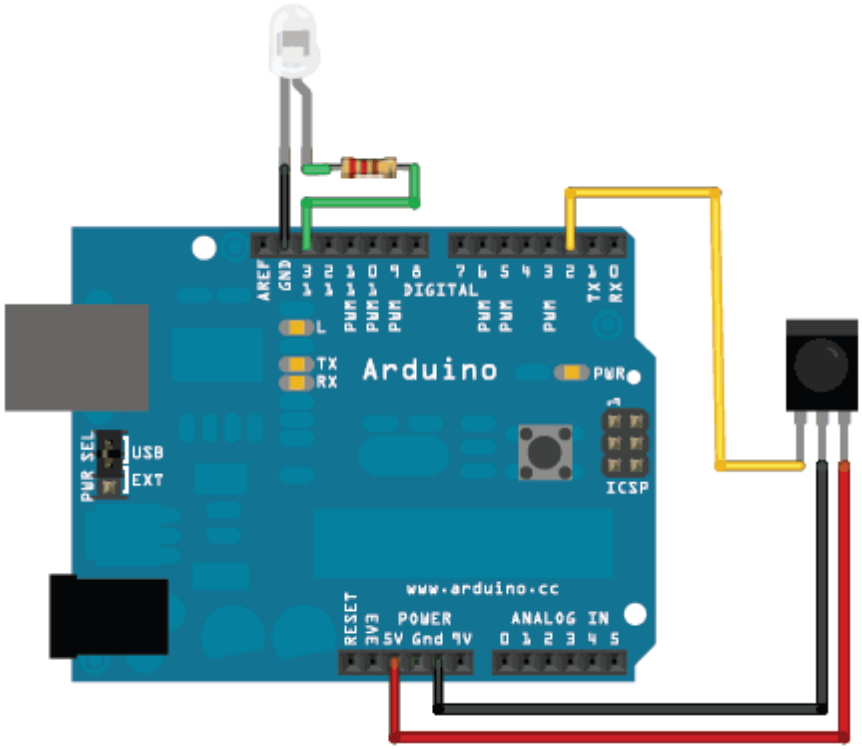
Para escrever o código dos programas em questão foram utilizados os ambientes de desenvolvimento integrado (IDEs) Visual Studio 2012 Express, Eclipse Kepler e uma IDE própria para desenvolvimento com Arduino, para os aplicativos central, de Android e da

microcontroladora, respectivamente. As ferramentas de gerenciamento e emulação de dispositivos virtuais do kit de desenvolvimento para Android e a extensão SQLite Manager para o navegador Mozilla Firefox foram usados para testes e depuração de erros (*debug*). Todo o projeto foi desenvolvido em um ambiente Windows.

4.1 Funcionamento do Programa de Detecção de Códigos

Para realizar a transmissão das representações dos códigos IR da microcontroladora (que faz a detecção) para o aplicativo central do projeto rodando em um computador e também em sentido inverso é utilizada uma conexão USB que serve tanto para transmissão de dados quanto para alimentar a microcontroladora, a qual estão conectados os demais componentes de acordo com o esquema ilustrado na Figura 20:

Figura 20. Microcontroladora e componentes



Fonte: Disponível em <www.ladyada.net/images/sensors/intervalometer.gif>, acessado em Fevereiro de 2015.

A placa microcontroladora Arduino Uno R3, utilizada no projeto, fornece 14 pinos de entrada e saída digital, um dos quais é utilizado para a conexão com o LED. Outro destes pinos é usado para a conexão de dados com o sensor, responsável por transmitir o sinal infravermelho recebido. O sensor também está conectado aos pinos de alimentação da placa, que fornecem energia para ela e para os componentes. O sensor utilizado opera em de forma ideal sob uma voltagem de 2.7V a 5.5V[25], recebendo 5V através da conexão USB. O LED

também é alimentado por esta conexão, que é limitada por um resistor de 180 Ohm para não sobrecarregá-lo, já que apenas 1.6V são necessários para acendê-lo.

Ao conectar a controladora à uma entrada USB de um computador, o programa de detecção de códigos começa a rodar e permanece à espera de um comando, executando em *loop*. Os comandos são recebidos em forma de um byte que representa um caractere de texto. O caractere literal 'l' representa o comando “ouvir por sinais IR” ou “entrar em modo de detecção”, que faz com que um método “Listen” (“ouvir”) seja chamado e passe a rodar também em um *loop*, substituindo o principal até que um sinal seja recebido ou o usuário interrompa a detecção. Este método faz duas verificações: se um sinal IR foi recebido, o que está diretamente ligado à entrada de dados através do sensor, e se um byte foi recebido pela conexão serial com o computador, neste caso, isto significa um comando para parar a detecção. Geralmente, este byte representa o caractere literal 'i', que significa “voltar para o estado de espera”, fazendo o programa voltar a executar o *loop* principal, mas qualquer caractere recebido pela conexão serial durante a execução da função Listen serviria para este mesmo propósito. Quando um sinal IR é detectado (se o usuário apontar um controle remoto em direção ao sensor e pressionar um botão com o programa em modo de detecção), é necessário verificar constantemente o estado do pino de dados do sensor: se um sinal IR ainda está sendo recebido, um contador que representa a duração em μs deste sinal é incrementado, em incrementos de 50. Quando o estado muda (um sinal não está sendo detectado) a duração registrada no contador deve ser adicionada em uma matriz que representa um código IR e o mesmo processo se repete, desta vez, utilizando um outro contador que registra a duração da ausência do sinal. A matriz utilizada para armazenar estes valores tem tamanho 100, o que significa que um código de até 100 pulsos pode ser registrado, tamanho que é suficiente para este propósito. Registrar com sucesso estes dois valores, significa que um pulso (um par *ON/OFF*, respectivamente LED aceso e LED apagado) foi armazenado. Os contadores são então zerados e a detecção do próximo pulso se inicia, até o fim do código. A última etapa do processo de detecção é criar uma representação textual do código detectado e enviá-la ao aplicativo central, o que é feito utilizando uma convenção específica descrita logo adiante.

O caractere 'p', quando recebido, significa “pulsar código” e deve ser sucedido por um número que representa o número de pulsos de um código e uma sequência de caracteres representando o código em si, mais o caractere 'e' no fim da mensagem, indicando seu término. A convenção utilizada nesta representação é a seguinte:

pNN+PPPP-PPPPe

Onde:

- “p”: Sinaliza o programa para pulsar um código
- NN: Representa o número total de pulsos do código
- “+”: Indica que o próximo valor representa um pulso alto (LED aceso)

- “-”: Indica que o próximo valor representa um pulso baixo (LED apagado)
- PPPP: Duração de um pulso em microssegundos
- “e”: Indicador de fim de mensagem

Eis um exemplo de mensagem recebida quando o usuário envia o comando para pulsar um código:

p34+4400-4450+500-1700+500-1650… [demais pulsos]… *+550-1650+500-1700+500-1650e*

Para enviar os códigos detectados da controladora para o aplicativo central no computador, a convenção utilizada é a mesma, com a diferença de não haver o caractere 'p' no início da mensagem, pois diferentemente do programa de detecção, o aplicativo central não precisa ser sinalizado para entrar em modo de recepção de códigos, pois este processo é iniciado manualmente pelo usuário.

Como mencionado, o caractere 'i' é um dos comandos que podem ser recebidos neste programa e tem a função de fazê-lo retornar ao estado inicial, aguardando por novos comandos. Outros caracteres também poderiam ser utilizados, com exceção de 'p' e 'l', já que estes são os únicos caracteres pelos quais o programa realmente “escuta” e quaisquer outros são interpretados por padrão como sinalizadores da opção “voltar ao estado de espera”, quando o programa não está pulsando um código. Durante o modo de detecção, não há restrições sobre quais caracteres podem ser usados para interrompê-lo, já que qualquer byte serial recebido irá engatilhar este evento, porém se algum dos caracteres sinalizadores de comando forem utilizados, eles serão interpretados como tal quando o programa voltar ao modo de espera, o que poderia causar efeitos indesejados. O caractere 'i' apenas é utilizado para fazer o programa voltar a este estado por motivos de clareza e para manter uma convenção bem definida, além de evitar os efeitos mencionados.

O programa de detecção de códigos, por rodar apenas na controladora, não possui nenhuma interface gráfica e também não faz nenhum tipo de armazenamento local. Este programa foi baseado no código disponível no artigo sobre sensores infravermelhos disponível em <http://www.ladyada.net/wiki/tutorials/learn/sensors/ir.html>, acessado em outubro de 2014.

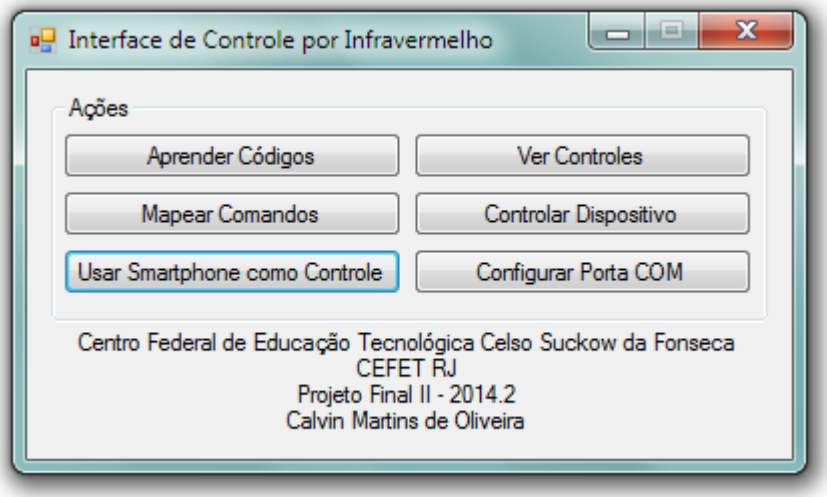
4.2 Funcionamento do Aplicativo Central

Este é o programa principal do sistema, com o qual o usuário faz a maior parte da interação. É também a parte que liga todas as outras, estando presente de um jeito ou de outros em todos os processos executados pelo projeto.

A Figura 21 ilustra a primeira tela que se vê quando o aplicativo é aberto, que mostra

de prontidão as opções disponíveis para o uso do programa. Para uma explicação teórica sobre a funcionalidade de cada uma destas opções, consultar a seção 3.4.2.

Figura 21. Tela inicial do aplicativo central



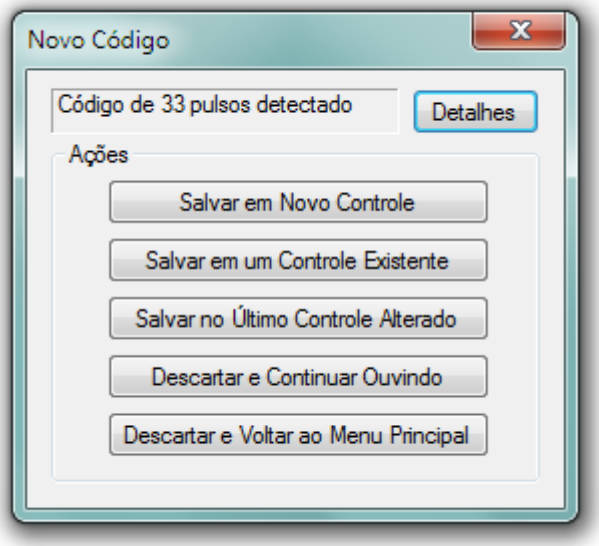
Adiante, serão descritos os procedimentos executados em todas as etapas de cada uma das opções.

- **Aprender Códigos**

Após configurar a porta de conexão com a microcontroladora usando a opção "Configurar Porta COM", esta opção fica disponível e exibe uma tela de espera utilizada para aguardar enquanto o usuário aponta um controle remoto para o sensor IR e pressiona um botão. Para detectar este evento, é necessário abrir uma conexão serial com a microcontroladora, feita através da classe "SerialPort" fornecida pelo *framework* .NET. Deve-se observar que esta conexão deve ser aberta em um *thread* de execução separado daquele no qual a interface gráfica do programa está em execução. Uma explicação breve sobre o motivo pelo qual é necessário executar operações deste tipo em *threads* independentes pode ser encontrada na seção 2.4.1. Para processar os dados enviados pela controladora e mostrá-los na interface gráfica, é necessário invocar o *thread* do qual esta faz parte, o que pode ser feito utilizando a função "Invoke" da classe "Control", que é a classe base para todos os elementos gráficos de um programa do tipo Windows Forms. Ao invocar o *thread* da interface gráfica, é necessário que o objeto responsável por iniciar a conexão com a controladora (neste caso, a janela que está em uso quando o programa está ouvindo por códigos) implemente a interface "IArduinoConnectable", que define um método para tratar os dados recebidos pela placa. Este método é chamado durante a invocação do *thread* e, no caso da adição de códigos, sua implementação é responsável por enviar os dados sobre o código IR recebido para a janela de salvamento de códigos, mostrada na Figura 22 logo adiante. O usuário pode interromper a detecção de códigos a qualquer momento clicando no botão "Cancelar", presente na tela de detecção, fazendo a desconexão do programa com a controladora e voltando à tela inicial.

Para realizar as operações de conexão, desconexão, recebimento e envio de dados da microcontroladora, é necessário instanciar um objeto da classe “ArduinoConnection”, que, de forma semelhante à classe “DatabaseConnection”, realiza estas operações isoladamente e faz as devidas checagens.

Figura 22. Tela de salvamento de códigos



As opções presentes nesta tela são bastante explicativas por elas mesmas: através delas é possível realizar o tratamento do código detectado de algumas formas diferentes, que basicamente se resumem a salvá-lo em um controle remoto ou descartá-lo. É possível também clicar no botão “Detalhes” e ver a representação textual dos dados sobre os pulsos, na convenção mostrada na seção 4.1, porém sem caracteres de controle e com o número de pulsos sendo mostrado separadamente.

Se o usuário escolher uma das opções para salvar o código, ele será levado à tela de edição de controles remotos, ilustrada mais adiante.

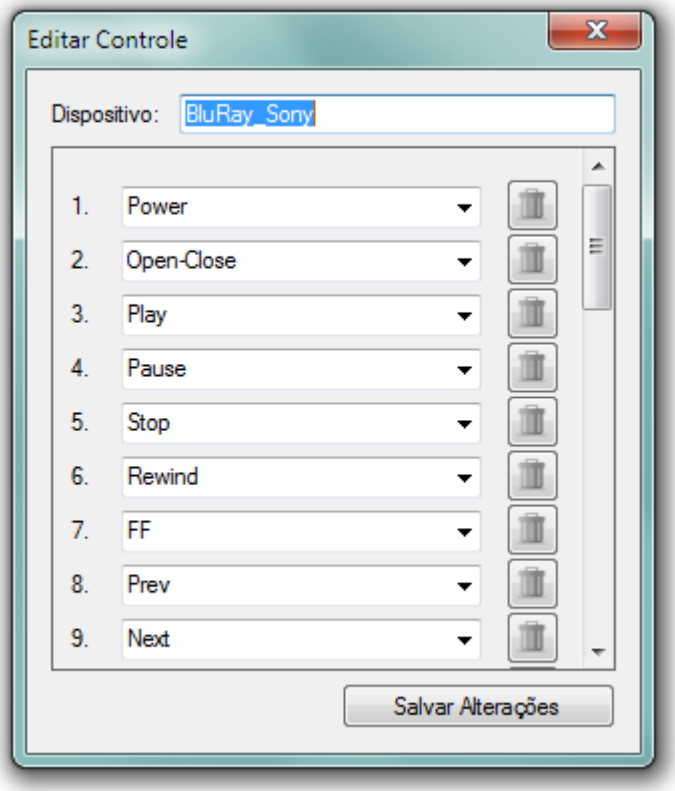
- **Ver Controles**

Esta opção permite selecionar um controle remoto salvo, se houver algum, e mostrá-lo na tela de edição de controles, que é a mesma que aparece quando o usuário seleciona salvar um código. Esta tela pode ser vista na Figura 23.

Fonte: Calvin Martins de Oliveira, Fevereiro de 2015

Nesta tela é possível alterar o nome do dispositivo cujo controle remoto está sendo editado assim como os nomes de cada um dos códigos, que podem ser digitados ou selecionados a partir de uma lista de nomes padrões disponíveis. No modo de edição, também é possível excluir códigos do controle clicando no botão com um ícone de lixeira presente ao lado do nome de um código.

Figura 23. Tela de edição de controles remotos



Quando esta tela é mostrada durante a adição de um código, apenas o nome do código sendo adicionado pode ser modificado e também não é possível excluir códigos: estes estão presentes apenas para exibição, para que usuário saiba quais códigos já existem no controle ao adicionar um novo. O nome do dispositivo também não pode ser alterado.

Durante a edição, os códigos carregados ficam disponíveis em uma lista local e são copiados para uma de duas outras listas quando são alterados ou removidos, uma para cada operação. Quando o usuário clica em "Salvar Alterações", as listas de alteração e remoção são iteradas e o banco de dados é atualizado em uma única transação. Em caso de erros em qualquer etapa da edição, nenhuma alteração no banco é efetuada. As modificações também não são salvas se o usuário sair da tela de edição fechando a janela.

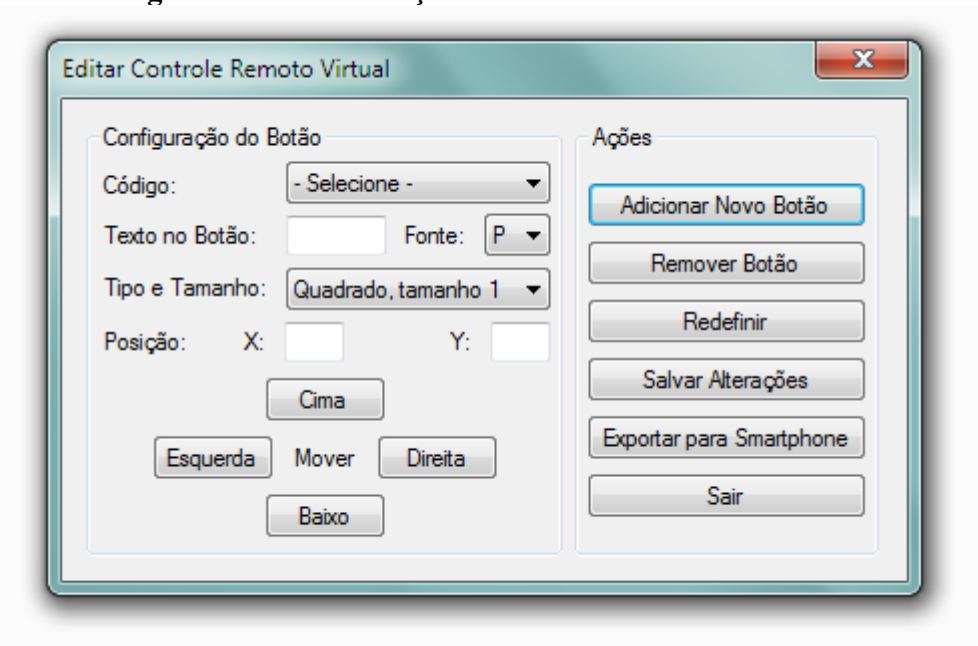
- **Mapear Comandos**

Esta é opção que permite ao usuário criar ou editar as interfaces de controle que ele deseja utilizar para controlar os dispositivos eletrônicos. Na atual implementação, dois tipos de interfaces podem ser criadas: controles remotos virtuais e mapeamentos do tipo comando para comando.

Ao selecionar a opção, o usuário deve escolher o controle remoto que deseja carregar para criar o mapeamento. Após fazer isto, uma outra tela é mostrada onde aparecem as interfaces de controles existentes para aquele controle remoto, se houver alguma. Caso não

haja, o usuário pode clicar no botão “Nova” e começar a criar uma. Se já existirem interfaces de controle armazenadas, uma delas pode ser selecionada através de uma lista *drop-down* e aberta para edição ao clicar no botão “OK”.

Figura 24. Tela de edição de controles remotos virtuais



Caso o usuário escolha criar uma interface de controle, ele deve definir seu tipo, o que também definirá qual tela será aberta em seguida. Ao escolher “Controle Remoto Virtual”, duas novas janelas são mostradas, sendo uma a pré-visualização do controle sendo criado e a outra uma janela de edição. Na visualização do controle, alguns botões comuns em controles remotos reais são adicionados automaticamente, para a conveniência do usuário, entre eles um botão liga e desliga (*power*), botões de controle de volume, de canal e um teclado numérico. Se desejar, o usuário pode remover estes botões um a um ou todos de uma vez, clicando no botão “Redefinir” na janela de edição. O usuário aqui pode selecionar entre uma série de opções para a edição dos botões do controle virtual, entre elas adicionar, remover, remover todos e alterar várias características dos botões, como seu tamanho, posição no controle, texto, e claro, o comando associado ao mesmo. Pode-se observar na Figura 24 abaixo, a aparência da tela de edição de controles remotos virtuais.

Ao clicar em um botão na representação do controle virtual na outra tela, as opções pertinentes a esse botão são automaticamente carregadas nesta tela. Nela o usuário pode utilizar os botões “Mover” para mudar a posição do botão no controle ou fazer isto manualmente, digitando as coordenas X e Y (horizontal e vertical, respectivamente) nos campos apropriados. É possível também escolher o estilo do botão (formato e tamanho) e o tamanho da fonte do texto.

Uma funcionalidade extra deste modo é a capacidade de mapear automaticamente os códigos com nomes padrões para seus respectivos botões, isto é, no caso do usuário ter

utilizado estes nomes durante a criação do controle na tela de edição de controles remotos (Figura 23). Isto significa que, por exemplo, se o usuário criou um controle remoto, adicionou a ele um código de ligar e desligar e o nomeou como “Power” (um dos nomes padrões disponíveis para seleção), ao criar um controle virtual para este controle remoto, o código Power já vai estar associado ao botão de ligar e desligar adicionado automaticamente. Isto é válido para quaisquer códigos que utilizarem nomes padrões e pode ser desfeito pelo usuário se ele assim desejar, clicando no botão na representação do controle virtual e selecionando um comando diferente – ou a opção referente a nenhum comando – na lista *drop-down* na tela de edição. Comandos com nomes customizados (digitados pelo usuário) não são mapeados automaticamente, sendo necessário clicar em um botão e fazer a seleção manual na lista.

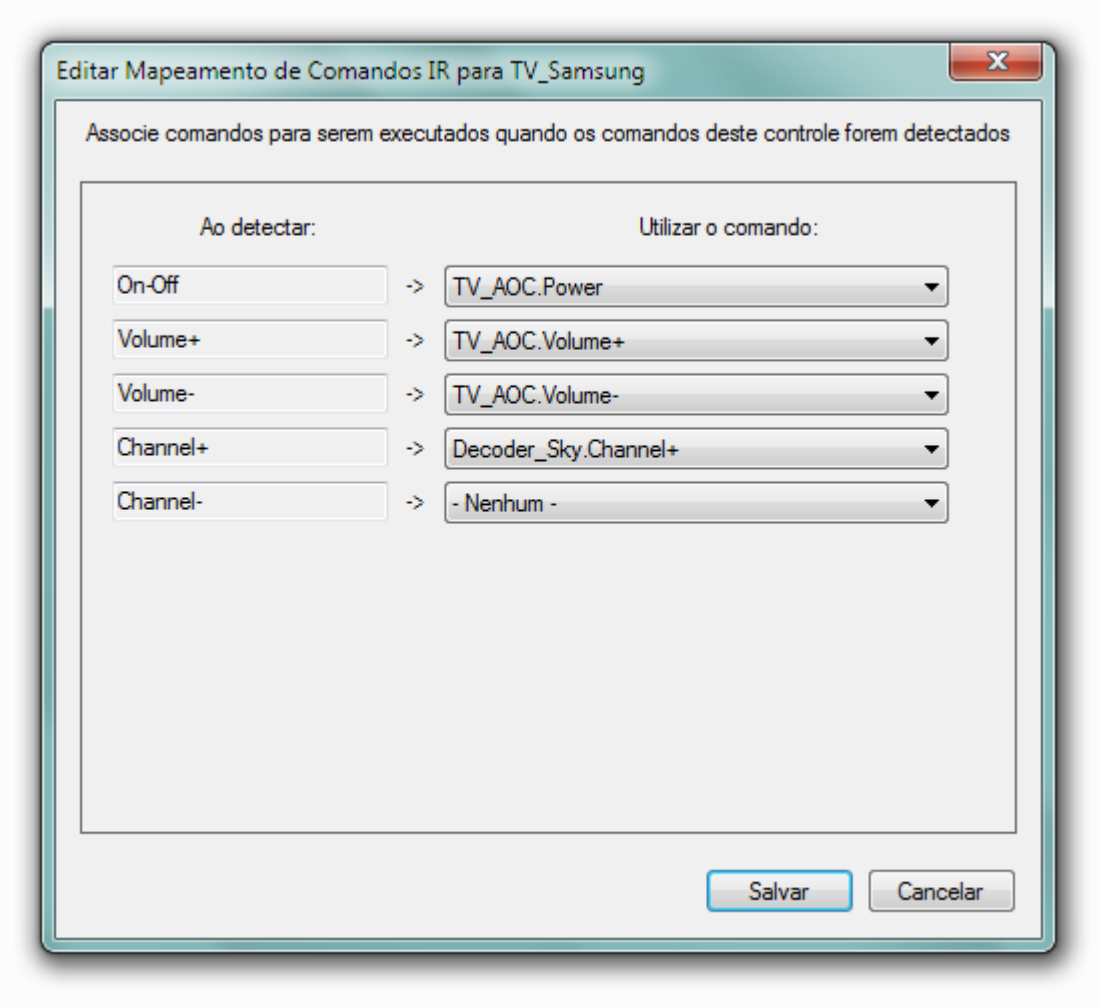
Assim como na tela de edição de controles remotos, as alterações feitas pelo usuário são gravadas no banco de dados apenas quando o botão “Salvar Alterações” é clicado, sendo possível fechar a janela a qualquer momento e descartá-las. Se o controle virtual sendo editado é um novo e isto for feito, ele não será salvo. O funcionamento deste recurso porém é levemente diferente de como ele aparece na edição de controles remotos, pois aqui, uma única lista de alterações no banco é utilizada, no lugar de uma para edição e outra para remoção. Esta é uma lista de botões virtuais do controle, que contém para cada posição um par composto de botão e operação. Quando um botão é alterado, ele é adicionado nesta lista com a operação “edit” (editar) e quando é removido, é adicionado com a operação “remove” (remover). Ao ser adicionado um novo botão, este já entra na lista com a operação “add” (adicionar). Esta lista de botões e operações é então passada para a classe DatabaseConnection – que, como visto, é a classe que lida com as operações no banco de dados – e os botões são inseridos, modificados ou excluídos do banco de acordo com suas marcações na lista. Se o controle virtual editado é novo, ele é adicionado antes dos botões na tabela apropriada e após sua adição, seu ID no banco de dados é recuperado e utilizado como chave estrangeira na tabela que armazena os mapeamentos de botão, cujos registros são adicionados por último. Novamente, se houver erro em alguma destas etapas, nenhuma alteração no banco de dados é realizada.

A tela da representação do controle virtual é a mesma utilizada quando o usuário seleciona o controle para uso e será mostrada no trecho sobre a opção “Controlar Dispositivo”. A funcionalidade “Exportar para Smartphone” será discutida no trecho sobre a opção “Usar Smartphone como Controle”.

Ainda nesta opção, é possível criar, em vez de um controle virtual, um conjunto de mapeamentos de comando para comando, o que pode ser feito se o usuário escolher a opção “Mapeamento IR para IR” na tela de adição de nova interface de controle. Se isto for feito, no lugar das telas de edição de controle virtual descritas acima, o usuário vê uma tela de associações entre comandos, que possui os nomes dos comandos do controle remoto selecionado em uma coluna à direita e uma série de listas *drop-down* com comandos de outros controles à direita, estando o comando em uma coluna associado diretamente ao comando ao

lado, em cada linha. Abaixo, na Figura 25, pode ser vista a tela de edição de mapeamentos IR.

Figura 25. Tela de edição de mapeamentos IR para IR



Como é possível observar, estes mapeamentos não estão limitados a um único controle, sendo possível associar comandos de quaisquer controles remotos existentes àqueles do selecionado. Com isto, é possível controlar diversos dispositivos utilizando um único controle remoto físico, desde que estejam todos dentro do alcance dos pulsos infravermelhos feitos pelo LED na controladora. Esta associação é feita da seguinte forma: Ao carregar a tela, uma lista de comandos do controle remoto selecionado é criada, assim como uma contendo os comandos de todos os outros controles. Quando um comando é selecionado em uma das listas da direita, seu ID de código (que é chave primária na tabela que armazena os códigos, sendo portanto único) é obtido, assim como o ID de código do comando à esquerda, do controle atual. Este par de IDs é então gravado numa terceira lista – desta vez global, ou seja, presente durante todo o processo de edição – que armazena os mapeamentos. Ao terminar de criar ou editar esses mapeamentos para o controle selecionado, o usuário pode selecionar a opção "Salvar", que grava as modificações no banco, da mesma forma que nos processos de edição vistos anteriormente, fazendo todas as alterações necessárias no banco de uma vez e, em caso de erro, nenhuma. A forma como estes mapeamentos são marcados para adição, alteração ou

remoção é a mesma utilizada na edição de controles remotos virtuais: utilizando uma lista que grava os mapeamentos (pares de IDs de código) e a respectiva operação que deve ser feita com eles no banco. Ao cancelar ou fechar a janela, nenhum dado é gravado.

- **Controlar Dispositivo**

A partir do momento em que alguma interface de controle foi criada, o usuário pode então utilizar esta opção no menu principal para selecionar uma dessas interfaces e utilizá-la para controlar o aparelho eletrônico de sua escolha, o que é o propósito maior deste projeto. Após clicar no botão "Controlar Dispositivo" o usuário escolhe o controle remoto que deseja utilizar a partir de uma lista e serão mostradas todas as interfaces de controle disponíveis para o escolhido, se houver alguma. Estas interfaces serão mostradas em uma outra lista cujos itens contém o nome do dispositivo associado àquele controle e o tipo da interface (controle remoto virtual ou mapeamento comando para comando). Ao selecionar um dos itens, a próxima tela carregada deve ser uma de duas: a visualização do controle virtual ou uma tela que indica que o programa está à espera de comandos de um controle físico, dependendo do tipo da interface selecionada. A tela do controle virtual é a mesma utilizada na edição de controles virtuais, porém desta vez, ao clicar em um botão o comando associado a ele é enviado para a controladora e então pulsado no LED IR. Esta tela pode ser vista na Figura 26 abaixo.

Figura 26. Tela do controle remoto virtual

Quando um botão é clicado no modo de controle, os dados sobre o código IR do comando associado a ele são agrupados numa mensagem que é então enviada à controladora através de uma conexão feita com o auxílio da classe ArduinoConnection. Isto é feito através de um método chamado "PulseCodeOnArduino", que quebra a mensagem em pedaços de até 64 bytes (64 caracteres) – que, como mencionado anteriormente, é o tamanho do *buffer* de entrada da Arduino Uno R3 – e envia cada um destes pedaços à placa esperando 1ms entre os envios, uma medida para evitar que dados sejam enviados mais rápido do que podem ser processados pela controladora, pois caso aconteça essa sobrecarga, a mensagem contendo os dados sobre o código não será recebida de forma intacta e o pulso não poderá ocorrer. Este atraso de 1ms é feito utilizando a função "Sleep" da classe "Thread" do .NET Framework, que recebe como argumento um número representando a duração de um intervalo durante o qual se deseja interromper a execução de código no *thread* de onde é chamada.

Se a interface de controle carregada for um mapeamento entre comandos, o programa

passa a esperar pela entrada de comandos de um controle remoto real, sendo o procedimento a ser executado quando um é recebido o seguinte: primeiro, utiliza-se os dados sobre os pulsos IR do comando detectado para criar uma instância de um código IR, que é representada no programa por um objeto da classe "InfraredCode". Esta classe, ao ser instanciada com os dados recebidos da controladora sendo passados para seu construtor, cria automaticamente a representação textual do código, armazenando os dados sobre os pulsos em variáveis. Ela também possui um método nomeado "GetPulsesAsMatrix", que realiza uma operação semelhante à feita pelo programa de detecção: transformar a representação textual do código IR em uma matriz de pulsos *ON/OFF*. Este método é usado para obter as durações dos pulsos do código detectado em formato numérico para que este código possa então ser comparado com aqueles do controle remoto selecionado (cujos comandos foram mapeados para comandos de outros controles). Estes códigos são carregados de antemão do banco de dados e armazenados em uma lista assim que a tela é iniciada, assim como os códigos mapeados, para que todo o processo seja mais rápido. Se após a comparação dos pulsos for concluído que o código detectado é um dos códigos do controle selecionado, o ID deste código é obtido. É importante observar que é necessário que haja certa tolerância com relação à exatidão dos valores das durações dos pulsos na hora de fazer a comparação, pois os códigos nem sempre são recebidos com exatamente a mesma duração. Por exemplo, um pulso com duração de 600µs pode algumas vezes ser registrado com a duração de 550 ou 650µs, já que o valor de incremento dos contadores de tempo dos pulsos é 50, como dito anteriormente. A tolerância utilizada no projeto é de 20%, o que significa que durações de pulso variando em até 20% para mais ou para menos são consideradas iguais. Após a identificação do código recebido, os mapeamentos, que também foram carregados do banco antecipadamente, são checados para obter o ID do código para qual o detectado está mapeado. Este ID então é usado para obter os dados sobre os pulsos do código mapeado da lista que foi criada e formar uma mensagem que será enviada à microcontroladora utilizando mais uma vez a função PulseCodeOnArduino. Se o código recebido não for identificado, um mapeamento não for encontrado ou qualquer outra falha acontecer, nenhum código é pulsado. Independentemente de um código ser pulsado ou não, o programa volta a esperar por comandos de um controle remoto e apenas deixa de fazer isto quando o usuário clica em no botão "Parar", o que faz com que o programa retorne à tela principal.

- **Usar Smartphone como Controle**

Esta opção funciona de forma muito semelhante à opção de controlar um dispositivo através de um controle remoto virtual, com a diferença de depender do aplicativo que roda em um dispositivo Android para exibir o controle ao invés do próprio aplicativo central no computador. A primeira etapa para utilizá-la é exportar o controle virtual criado usando a opção "Exportar para Smartphone" na tela de edição de controles virtuais. Ao clicar nesta opção, uma tela que exibe o endereço do computador rodando o aplicativo central é mostrada, com uma mensagem requisitando ao usuário que abra o aplicativo para Android e selecione a

opção “Receber Controle Virtual” disponível nele. Se o usuário assim o fizer, uma conexão entre os dois aplicativos é iniciada e o controle virtual é recebido e armazenado pelo dispositivo Android, enquanto o aplicativo central volta à tela inicial. Após este processo, quando o usuário clica na opção “Usar Smartphone como Controle”, a mesma tela exibindo o endereço do computador é mostrada, porém desta vez com uma mensagem pedindo que o usuário selecione e opção “Controlar Dispositivo” no aplicativo para Android. O programa então passa a escutar por mensagens vindas deste aplicativo, que devem conter o ID de um código a ser pulsado e se alguma for recebida, este ID é utilizado para obter os dados sobre os pulsos do código e o mesmo processo feito nos modos de controle por controle remoto virtual e mapeamento entre comandos é então executado para pulsar o código.

Para realizar a conexão com o aplicativo para Android, a classe “SocketConnection” é utilizada. Esta classe funciona de forma semelhante às já mencionadas DatabaseConnection e ArduinoConnection, executando todos os procedimentos do processo de conexão de forma isolada dos outros objetos no programa. A interface “ISocketConnectable” deve ser implementada por qualquer objeto que requisite uma conexão via *sockets* e define um método para lidar com os dados recebidos através desta conexão. Assim como a conexão com a controladora, também é necessário realizar a conexão via *sockets* em um *thread* separado, para que a interface gráfica do programa se mantenha funcional, pois a solução implementada neste projeto é a de *sockets* síncronos, o que significa que a responsividade do programa apenas é restabelecida quando uma mensagem enviada é respondida pela outra parte. Quando a função “StartListening” da classe SocketConnection é chamada, este novo *thread* de execução é criado e nele o programa passa a ouvir por mensagens vindas do aplicativo para Android através desta conexão, enquanto mantém a interface gráfica do programa funcional, permitindo ao usuário interromper o processo a qualquer momento clicando em “Cancelar”, fazendo com que seja chamada a função “StopListening”, responsável por forçar a interrupção da conexão via *sockets*, independentemente de uma mensagem ter sido recebida. O método responsável por fazer a verificação do recebimento de dados via *socket* assim como atualizar a interface gráfica com o endereço IP do computador e a porta de conexão (invocando o *thread* necessário através do método Invoke, assim como é feito na conexão serial descrita no trecho sobre a opção “Aprender Códigos”) é o método “Listen”. Através da mesma técnica de invocação utilizada para atualizar a interface gráfica com o endereço de conexão do computador também é feito o processamento das informações enviadas pelo aplicativo para Android.

Ao receber um ID de código, o programa acessa o banco de dados e faz uma consulta para obter as informações sobre os pulsos deste e também de todos os códigos do mesmo controle remoto, dados que então salva em uma lista armazenada localmente. Esta operação tem por objetivo agilizar o processo de pulsar os códigos, consultando a lista criada no lugar do banco de dados para os ID seguintes que forem enviados. Isto é feito apenas na primeira vez que um ID de código válido é recebido.

Mais sobre a conexão entre o aplicativo central e o aplicativo para Android pode ser encontrado na seção seguinte, que descreve o funcionamento deste último.

4.3 Funcionamento do Aplicativo para Android

O aplicativo que permite carregar controles remotos virtuais na tela de um dispositivo Android funciona de forma bastante simples, tendo apenas duas funções principais: o armazenamento de controles virtuais enviados pelo aplicativo central e o carregamento destes na tela do dispositivo para serem utilizados assim como são no computador.

Este aplicativo também utiliza um banco de dados SQLite local, porém sua estrutura é bem mais simples do que a do banco utilizado pelo aplicativo central, contendo apenas duas tabelas: uma para o armazenamento dos nomes dos dispositivos cujos controles virtuais recebidos controlam e outra para armazenar os dados sobre a aparência dos botões destes controles além dos IDs de código associados a eles, esta última sendo muito semelhante à tabela “VirtualRemoteMaps” do banco de dados do aplicativo central. Devido à simplicidade deste banco, não será usando um diagrama ER para ilustrá-lo.

Para estabelecer uma conexão via *sockets* com o aplicativo central, é utilizada a classe customizada “ClientTask”, que estende a classe “AsyncTask” fornecida pelo Android. Esta classe oferece uma forma simples para realizar a execução de tarefas em ambientes de execução separados sem que haja a necessidade de se lidar diretamente com conceitos mais complicados como os de *threads*. Quaisquer classes que herdem desta devem sobrescrever o método “doInBackground”, que é o responsável por executar em plano de fundo as tarefas necessárias, sem comprometer a interface gráfica do programa. No caso deste projeto, o método é responsável por ouvir por mensagens vindas do aplicativo central após a conexão ser estabelecida. Essas mensagens serão lidas byte a byte e armazenadas numa variável que será mais tarde usada para tratá-la. Opcionalmente, pode-se também sobrescrever os métodos “onPreExecute” e “onPostExecute”, cujas implementações são executadas respectivamente antes e depois das tarefas principais realizadas em doInBackgrond, no mesmo *thread* da interface gráfica, o que é um meio muito conveniente para a realização de ações como mostrar os resultados destas tarefas na interface gráfica, por exemplo. Neste programa, este recurso é utilizado para mostrar uma caixa de mensagem com a confirmação de que um controle virtual foi recebido com sucesso ou informar ao usuário sobre algum erro ocorrido no recebimento deste.

Pelo fato da interface gráfica de usuário deste aplicativo ser demasiada simples, contendo pouquíssimos elementos em cada tela, não serão utilizadas imagens para descrevê-la. A tela principal do aplicativo consiste apenas de dois botões, um para a opção “Receber Controle Virtual” e outro para a opção “Controlar Dispositivo”. Ao selecionar a primeira opção, o programa mostra uma tela com um campo de texto e que requisita ao usuário que digite o endereço do computador com o qual o aplicativo deve se conectar, que pode ser encontrado na tela de conexão com o dispositivo Android no aplicativo central. Digitando o

endereço e pressionando o botão de confirmação, é então aberta uma tela que contém apenas uma mensagem de espera que diz que o aplicativo está no momento se conectando com o computador. É neste momento que o controle virtual é recebido, através de uma representação em formato texto, assim como são enviados os códigos entre aplicativo central e o programa de detecção na controladora. O formato das mensagens de comunicação entre os aplicativos será descrito mais adiante. Quando a representação do controle é recebida com sucesso, ela é interpretada e os dados sobre tal controle são gravados no banco de dados da aplicação.

Ao selecionar a segunda opção, o usuário vê uma tela que contém nada mais que uma lista dos controles virtuais disponíveis e dois botões, um de confirmação e o outro para cancelamento, que faz o aplicativo voltar à tela principal. Caso selecione um controle e pressione o botão de confirmação, o usuário é então direcionado à mesma tela de conexão com o computador que vê quando requisita o recebimento de um controle virtual, onde deve digitar o endereço do computador, que será utilizado quando for necessário enviar um ID de código para o aplicativo central. Após entrar com o endereço e confirmar, é então carregado na tela do dispositivo o controle virtual selecionado, da mesma forma que aparece na tela do computador durante o modo de controle do aplicativo central. Um fator obtido multiplicando a altura em *pixels* da tela do dispositivo por 0.0019 é utilizado para dimensionar o controle virtual na tela adequadamente, para que o mesmo seja mostrado com aproximadamente o mesmo tamanho em dispositivos com telas de diferentes tamanhos. Este fator é utilizado não apenas para ajustar o tamanho do controle virtual mas também o da maioria dos componentes gráficos deste aplicativo. Depois de fazer o carregamento do controle virtual na tela, o usuário pode então pressionar seus botões como se estivesse usando um controle remoto real, ação que fará com que a conexão com o aplicativo central via *sockets* seja iniciada e o ID do código associado ao botão pressionado seja enviado para esse aplicativo via rede.

A convenção utilizada para a troca de mensagens entre o aplicativo central e este aplicativo é a seguinte:

- '$' representa um comando. Este é o primeiro caractere que deve aparecer em todas as mensagens trocadas entre os dois aplicativos e é sucedido pelo número do comando, que pode ser 1 ou 2, na implementação atual.
- “<EOF>” sinaliza o fim da mensagem. Aparece no final de todas as mensagens.
- '@' existe apenas nas mensagens que contém representações de controles virtuais, enviadas no sentido aplicativo central para Android e representa um botão do controle. Este caractere será sucedido de uma série de propriedades compostas de pares nome e valor, que descrevem o botão.
- ',' é o separador de propriedades de um botão de um controle virtual, como “shape:square”, por exemplo, que significa que o formato do botão é quadrado.
- ':' é o caractere que separa o nome de uma propriedade de seu valor

Para ilustrar o modo como as mensagens são trocadas utlizando esta convenção, seguem alguns exemplos de mensagens.

$1<EOF> - sentido Android → Aplicativo central

Esta mensagem (comando 1) significa que o aplicativo está pronto para receber um controle virtual.

$2NomeDispositivoLLLxAAA@shape:square,size:3,... [demais propriedades]...
codeid:NN@... [demais botões]... *<EOF>* - sentido Aplicativo central → Android

Esta mensagem é a resposta do aplicativo central para a mensagem anterior, contendo a representação em texto de um controle virtual. "LLL" representa a largura em *pixels* do controle virtual e "AAA" sua altura. "NN" representa o ID do código IR associado com o botão em questão e pode ser tanto um ID válido quanto -1 que significa que o botão não possui nenhum código associado (portanto nada acontecerá quando ele for clicado).

$2NN<EOF> - sentido Android → Aplicativo central

Esta mensagem (comando 2) enviada pelo aplicativo de Android é a requisição do mesmo para que um código seja pulsado, cujo ID é representado na mensagem por "NN".

$1<EOF> - sentido Aplicativo central → Android

Esta mensagem é apenas uma confirmação dada pelo aplicativo central de que o ID foi recebido. É importante notar que isto não é garantia de que um código será pulsado, pois para isto o ID precisa ser válido, verificação que é feita somente após o recebimento.

5 Avaliação Experimental

Nesta seção, serão descritos alguns testes práticos feitos para comprovar a eficácia do projeto, com respeito aos objetivos que foram alcançados. Deve-se observar que, por causa da dificuldade de obtenção do equipamento necessário para a realização dos testes (isto é, a necessidade de se utilizar diversos controles remotos e aparelhos eletrônicos diferentes para que se possa fazer uma quantidade apropriada de testes, assim como testes diversificados), estes acabaram por ser bastante simples e em pequena quantidade.

O objetivo dos testes foi controlar com sucesso um aparelho de televisão da marca Samsung através dos métodos propostos por este projeto, avaliando fatores como a velocidade de resposta, estabilidade e eficiência de cada um deles. Apenas as funções básicas de controle foram testadas.

- **Controlando o aparelho por controle remoto virtual**

Este, por ser o método de mais simples execução, é também o mais rápido. O processo de identificação de um código, seu envio para a placa controladora e reprodução de seus pulsos no LED ocorre em aproximadamente 1 segundo, o que certamente constitui um atraso

maior do que aquele que se pode perceber quando se usa um controle remoto convencional, porém não chega a ser um incômodo. É relevante notar que, por causa do processo de conversão do código entre texto e pulsos, o tempo total de execução destes processos varia de acordo com o comprimento do código. Por exemplo, um código de apenas 20 pulsos será pulsado mais rápido do que um código de 60 pulsos, uma diferença de tempo que também não é perceptível ao usar um controle remoto físico, já que o único processo que acontece nesse caso é a reprodução dos códigos, que como observado no estudo do tema e mencionado anteriormente, ocorre de forma muito rápida (na casa dos microssegundos).

Em alguns testes realizados durante o desenvolvimento do projeto, para avaliação de progresso, pôde-se notar certa instabilidade tanto na detecção quanto na reprodução dos códigos pela controladora. Em certos casos, um código era detectado com um número de pulsos diferente do normal, contendo uma espécie de "poluição" nos dados dos pulsos. Nas fases finais do desenvolvimento, este problema cessou em ocorrer, porém sua causa não pode ser identificada com exatidão. O fato de não haver um filtro infravermelho (para que apenas luz no espectro infravermelho fosse detectada) diante do sensor é uma das causas suspeitas dessa instabilidade na detecção, já que a luz infravermelha presente no ambiente pode causar interferências. Isto pode ser comprovado ao se observar que, durante o modo de detecção de códigos, certas vezes sinais IR são detectados mesmo quando não são utilizados controles remotos perante o sensor e identificados erroneamente como códigos de controles, por causa de ações como acender e apagar uma lâmpada convencional na presença do sensor ou movê-lo. Com relação à instabilidade na execução dos códigos, este problema, que não apareceu no último teste que foi feito, aparentemente deixou de ocorrer após alguns ajustes que foram feitos na fase final no programa da controladora com relação ao recebimento de dados seriais no *buffer* de entrada, que estava sendo limpo desnecessariamente em certos momentos e por causa disto, às vezes acabava por ignorar comandos dados pelo programa central.

- **Controlando o aparelho pelo aplicativo para Android**

Este método foi testado de duas formas diferentes: no primeiro teste, o aplicativo central rodava em um *desktop* (computador de mesa) conectado à internet via cabo *ethernet* e no segundo, ele rodava em um computador móvel, conectado à rede por Wi-fi. Em ambos os testes, o aparelho utilizado para rodar o aplicativo para Android foi um tablet Samsung Galaxy Tab 3, conectado à internet também através de Wi-fi.

Com relação a este método de controle, não houve nenhum problema em nenhum dos dois testes, com o aparelho de televisão sendo controlado com sucesso em aproximadamente o mesmo intervalo de tempo que no modo utilizando controle virtual, pelo fato dos processos por trás deste método também serem bem simples, consistindo apenas no envio de uma curta mensagem contendo o ID do código IR através da rede e a reprodução dos pulsos deste respectivo código na controladora, com o programa central agindo como uma ponte. Uma diferença que se pôde notar entre os dois testes, porém, é que no segundo, onde ambos os dispositivos utilizados se conectavam à rede de modo sem fio, alguns códigos eram pulsados

com certo atraso, sendo reproduzidos praticamente juntos com o código do próximo comando dado. Acredita-se que esta instabilidade tenha sido causada pelo fato de apenas uma conexão sem fio ter sido utilizada no teste (que está mais sujeita a interferências do que uma conexão via cabo) e o fato de haver algumas paredes diminuindo a intensidade do sinal Wi-fi que chegava aos dispositivos.

- **Controlando o aparelho com um controle remoto de outro**

Pelo fato de ser o modo mais complexo de controle, onde o maior número de operações consecutivas e que fazem uma maior demanda de tempo serem realizadas, este acaba por ser o mais demorado. Neste teste, foi possível controlar o aparelho de televisão utilizando o controle remoto de um leitor de BluRay da marca Sony, porém, dada a complexidade do método, que requer a identificação de um código através de seus pulsos, obtenção do ID do código associado e o envio dos dados sobre os pulsos deste código à controladora, o tempo de resposta entre o pressionamento de um botão no controle Sony e a execução do comando no televisor Samsung sofreu atrasos de aproximadamente 2 a 3 segundos. Um atraso desta duração pode causar incômodo e fazer o usuário sentir que esta forma de controle não é muito natural. Outro fator que pode se mostrar como uma inconveniência neste método de controle é que os botões do controle remoto físico sendo utilizado devem ser pressionados brevemente, para que o código não se repita mais do o necessário, pois se isto acontecer, ele não será identificado como o mesmo código que se armazenou no banco de dados da aplicação, não equivalendo com àquele utilizado no mapeamento e consequentemente não causando nenhum efeito no aparelho controlado. Esta limitação também está presente no modo de detecção de códigos, que pode facilmente identificar um código IR como tendo o dobro de sua duração real pelo fato do usuário ter pressionado o botão por tempo demais, o que faria com que esta versão duplicada do código fosse aquela a ser armazenada no banco.

- **Validade do experimento**

É importante ressaltar que os testes realizados foram apenas os essenciais para comprovar o funcionamento correto do projeto: testes muito mais extensos podem ser realizados para identificar mais pontos onde o mesmo pode ser melhorado e possivelmente identificar erros a serem corrigidos. Devido à baixa amostra de códigos IR que foi possível registrar no programa e utilizar nos testes, é possível que haja algumas marcas de controles remotos cujos códigos possuam peculiaridades com as quais os programas não estejam preparados para lidar, devendo ser moldados de forma que isto passe a ser possível.

Como mencionado, um dos fatores que pode ter se mostrado um empecilho na execução dos testes é a falta de um filtro infravermelho, que possa separar as luzes de outras faixas do espectro da luz infravermelha e fazer com que apenas esta última chegue ao sensor, o que resultaria na detecção de um sinal mais “puro” e consequentemente, mais preciso.

Outro fator importante que se deve observar para a realização de experimentos

adequados é o desempenho do computador no qual o programa central é executado. Em um dos testes realizados, não documentado nas seções acima, feito em um computador móvel diferente do utilizado no teste de controle por dispositivo Android, que tinha por objetivo controlar um aparelho de televisão da marca AOC, isto se mostrou impossível pelo fato de a comunicação entre o computador e a microcontroladora, por algum motivo, ocorrer de forma lentíssima, fazendo com que os códigos levassem tempos absurdos de 8 a 20 segundos para serem transmitidos entre estes dois meios. Neste mesmo computador, ao conectar a controladora em uma porta USB diferente, os códigos passaram a ser transmitidos em uma velocidade mais próxima da normal, o que levantou a suspeita de que houvesse algum problema com a porta USB utilizada anteriormente. Estes e outros fatores podem comprometer a validade dos testes se não forem observados e devidamente tratados.

6 Conclusão

Neste projeto, foi realizado o desenvolvimento de um sistema capaz de controlar com sucesso aparelhos eletrônicos remotamente com o uso de diferentes formas de controle, cumprindo a maior parte dos objetivos estabelecidos antes de seu desenvolvimento. Houve porém, alguns aspectos que ocasionaram em limitações ao projeto, que podem ser abordados em estudos ou trabalhos futuros e serão discutidos adiante.

A limitação mais importante a ser discutida é que o modo de controle através do mapeamento de códigos IR não funciona de forma adequada quando o controle mapeado utiliza o padrão de controles remotos Philips (possivelmente todos os controles que utilizam protocolo RC5 de comunicação), pois os comandos destes controles são enviados através de códigos IR que variam em número de pulsos para um mesmo comando. Isto significa que um mesmo comando (por exemplo, ligar/desligar) pode ser enviado com, digamos, 18 pulsos em certas ocasiões e 19 pulsos em outras, alternando a cada pressionamento de botão. É possível que isto esteja ligado ao bit *toggle* utilizado em comandos que seguem este padrão, que é utilizado para determinar se um botão foi pressionado e solto ou mantido pressionado, o que poderia causar esta discrepância de pulsos (ver seção 2.5.2 para mais informações referentes ao padrão de controles discutido neste parágrafo). Isto faz com que seja impossível identificar com precisão os códigos enviados por controles deste tipo – para serem comparados com os códigos armazenados – em 100% das ocasiões, a menos que ambas as versões de cada um dos códigos fossem armazenadas, o que certamente não é viável nem conveniente.

Outra limitação que poderia ser eliminada fosse aplicado um trabalho mais aprofundado no projeto é o espaço adicional sendo utilizado no banco de dado para armazenar códigos que sempre têm seus pulsos repetidos. Para estes códigos, independentemente do quão breve é o pressionamento do botão feito pelo usuário, certo padrão de pulsos será inevitavelmente repetido, com ausências de sinal um pouco maiores entre as repetições, para sinalizar isto. Alguns códigos que puderam ser observados durante os testes chegaram a ser repetidos um mínimo de 6 vezes, fazendo com que as informações sobre estes pulsos

repetidos fosse armazenada como uma desnecessariamente longa cadeia de caracteres no banco de dados. Uma possível solução para este problema seria, no ato da detecção de um código, identificar tais repetições e armazenar apenas uma vez no banco a sequência que se repete, assim como a duração do pulso que as separa (aquele com a ausência de sinal mais longa) e o número de repetições, para que o código, a partir destes dados, pudesse ser novamente montado quando precisasse ser utilizado. É claro que isto faria com que a complexidade de procedimentos como os realizados, por exemplo, no modo de controle por mapeamento IR ficasse ainda maior, já que seriam necessárias mais operações para identificar um código baseado em seus pulsos, tendo como efeito um aumento indesejado no tempo de execução do processo, consequentemente fazendo com que o tempo de resposta neste modo, que já é demasiadamente longo, se tornasse ainda mais.

Desconsiderando estas e outras pequenas limitações existentes no projeto, o resultado final alcançado se mostrou muito próximo do almejado, com a maioria dos objetivos iniciais sendo cumpridos. Em linhas gerais, a principal contribuição do trabalho, além do melhor entendimento sobre o funcionamento da comunicação por infravermelho, foi a criação de um sistema base com grande potencial, que, com o devido investimento de tempo e recursos, pode ser estendido e transformado em algo realmente conveniente de se ter no dia a dia. Fossem adicionados os devidos componentes e feitas as modificações necessárias, o produto desenvolvido neste projeto poderia facilmente ser integrado, por exemplo, a um sistema de automação residencial, o que certamente proporcionaria a seus usuários um aumento de conforto no ambiente de seu lar através da possibilidade de controlar os dispositivos ali presentes via diversos tipos de interação.

Para trabalhos futuros, uma proposta interessante é a criação de um aplicativo para Smartphone que funcione de forma independente, isto é, inclua em sua funcionalidade as partes que são executadas na atual implementação pela placa Arduino e pelo computador que roda o programa principal, sendo capaz de detectar e pulsar códigos por conta própria além de permitir que o usuário monte os controles virtuais diretamente na tela do dispositivo móvel. A detecção e pulsagem de códigos poderia tirar proveito das capacidades que alguns modelos destes dispositivos possuem de lidar com sinal infravermelho, que são utilizadas normalmente para transferência de arquivos. Para criar os controles virtuais na tela, poderia existir uma interface onde o usuário, para a sua conveniência, pudesse selecionar de um painel botões pré desenhados e arrastá-los diretamente para o controle, alterando suas características em um menu à parte conforme necessário.

Outra possível adição seria o controle de aparelhos através de outros protocolos além do infravermelho, como via IP, por exemplo. Isto possibilitaria que o controle acontecesse através da rede enquanto não limitado-o ao alcance de um sinal Wi-fi local, permitindo assim que seja exercido de qualquer lugar, desde que o usuário tenha seu aplicativo conectado à internet. Isto também permitiria que toda uma nova gama de aparelhos eletrônicos pudesse ser controlada, desde ares-condicionados inteligentes a projetores sem fio, além de muitos outros.

7 Referências

[1] CARVALHO, T. **Espectro Eletromagnético**. Disponível em <http://www.infoescola.com/fisica/espectro-eletromagnetico/>, acessado em Fevereiro de 2015.

[2] ENCYCLOPEDIA BRITANNICA. **Hertz**. Disponível em <http://www.britannica.com/EBchecked/topic/263882/hertz>, acessado em Fevereiro de 2015.

[3] FRIED, L. **tutorials:learn:sensors:ir.html**. Disponível em <http://www.ladyada.net/wiki/tutorials/learn/sensors/ir.html>, acessado em Fevereiro de 2015.

[4] BLUM, J. **Exploring Arduino: Tools and Techniques for Engineering Wizardry**. 1ª edição. Indianápolis: Wiley, 2013.

[5] IEEE SPECTRUM. **The Making of Arduino**. Disponível em <http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino>, acessado em Março de 2015.

[6] ARDUINO.CC. **ArduinoBoardLeonardo**. Disponível em <http://arduino.cc/en/Main/ArduinoBoardLeonardo>, acessado em Março de 2015.

[7] ARDUINO.CC. **ArduinoBoardDue**. Disponível em <http://arduino.cc/en/Main/ArduinoBoardDue>, acessado em Março de 2015.

[8] ADUINO.CC. **ArduinoBoardNano**. Disponível em <http://arduino.cc/en/Main/ArduinoBoardNano>, acessado em Março de 2015.

[9] ARDUINO.CC. **ArduinoBoardEthernet**. Disponível em <http://arduino.cc/en/Main/ArduinoBoardEthernet>, acessado em Março de 2015.

[10] ARDUINO.CC. **ArduinoBoardEsplora**. Disponível em <http://arduino.cc/en/Main/ArduinoBoardEsplora>, acessado em Março de 2015.

[11] ARDUINO.CC. **ArduinoWiFiShield**. Disponível em <http://arduino.cc/en/Main/ArduinoWiFiShield>, acessado em Março de 2015.

[12] ARDUINO.CC. **ArduinoMotorShieldR3**, Disponível em <http://arduino.cc/en/Main/ArduinoMotorShieldR3>, acessado em Março de 2015.

[13] NATIONAL COMPUTATION INFRASTRUCTURE. **Serial Port I/O**. Disponível em <https://nf.nci.org.au/facilities/software/Matlab/techdoc/matlab_external/ch_seri8.html>, acessado em Fevereiro de 2015.

[14] ARDUINO.CC. **Referência do método "begin"**. Disponível em <http://arduino.cc/en/Serial/begin>, acessado em Fevereiro de 2015.

[15] ARDUINO.CC. **Referência do método "available"**. Disponível em

<http://arduino.cc/en/Serial/available>, acessado em Fevereiro de 2015.

[16] MAKOFSKE, D. B., DONAHOO, M. J., CALVERT, K. L. **TCP/IP Sockets in C#: Practical Guide for Programmers**. 1ª edição. San Francisco: Morgan Kaufmann, 2004.

[17] ORACLE JAVA DOCUMENTATION. **What is a Socket?**. Dispoível em <http://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>, acessado em Fevereiro de 2015.

[18] DEWAN, P. **Synchronous vs Asynchronous**. Disponível em <http://www.cs.unc.edu/~dewan/242/s07/notes/ipc/node9.html>, acessado em Fevereiro de 2015.

[19] ANDROID DEVELOPERS. **Referência da classe "AsyncTask"**. Disponível em <http://developer.android.com/reference/android/os/AsyncTask.html>, acessado em Fevereiro de 2015.

[20] PICPROJECTS. **Sony SIRC Decoder**. Disponível em <http://picprojects.org.uk/projects/sirc/>, acessado em Fevereiro de 2015.

[21] PICPROJECTS. **Sony SIRC Infrared Protocol**. Disponível em <http://picprojects.org.uk/projects/sirc/sonysirc.pdf>, acessado em Fevereiro de 2015.

[22] ALTIUM. **Philips RC5 Infrared Transmission Protocol**. Disponível em <http://techdocs.altium.com/display/FPGA/Philips+RC5+Infrared+Transmission+Protocol>, acessado em Fevereiro de 2015.

[23] IRULE. **iRule for DIY AV Enthusiasts**. Disponível em <http://www.iruleathome.com/irule-for-diy-enthusiasts>, acessado em Fevereiro de 2015.

[24] VIATEK CONSUMER PRODUCTS GROUP. **Zmart Remote**. Disponível em <http://viatekproducts.com/zmart-remote/>, acessado em Fevereiro de 2015.

[25] EVERLIGHT ELECTRONICS CO., LTD. **Technical Data Sheet - Infrared Remote-control Receiver Module**. Disponível em <http://pdf.datasheetcatalog.com/datasheets2/20/201002_1.pdf>, acessado em Fevereiro de 2015.

Apêndice

Manual do Usuário

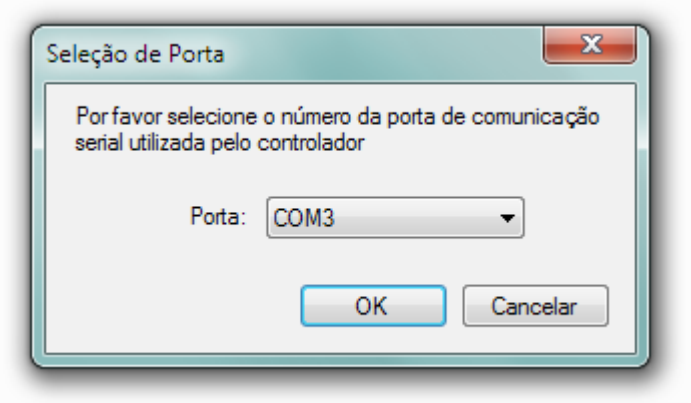
Este manual descreve as funcionalidades do programa IRConnection, fornecendo instruções que visam a facilitação de seu uso por usuários de qualquer nível de conhecimento técnico, tornando possível que tirem proveito máximo da solução.

Tarefa 1: Configuração da porta de comunicação com a placa controladora

Para todas as tarefas que envolvem a transmissão ou recebimento de comandos, a comunicação do programa com a placa microcontroladora é necessária. Para que esta conexão possa ser feita, as seguintes etapas devem ser realizadas:

1. Na tela principal, selecione a opção “Configurar Porta COM”.
2. Uma lista com as portas disponíveis será mostrada. Verifique que a controladora está conectada em uma porta USB funcional e selecione a porta de comunicação utilizada por ela na lista.

Figura 27. Tela de configuração de porta



Nota: se o número da porta não for conhecido, esta informação pode ser obtida no Gerenciador de Dispositivos do Windows, na seção “Portas COM e LPT”.

3. Clique em “OK” para confirmar.

Esta configuração precisa ser realizada antes de quaisquer tentativas de conexão com a controladora, pois se isto não for feito, será impossível conectar: o programa apenas mostrará uma mensagem de erro e pedirá que a configuração seja feita, voltando à tela principal e interrompendo o prosseguimento de qualquer operação.

Tarefa 2: Adicionar novo comando

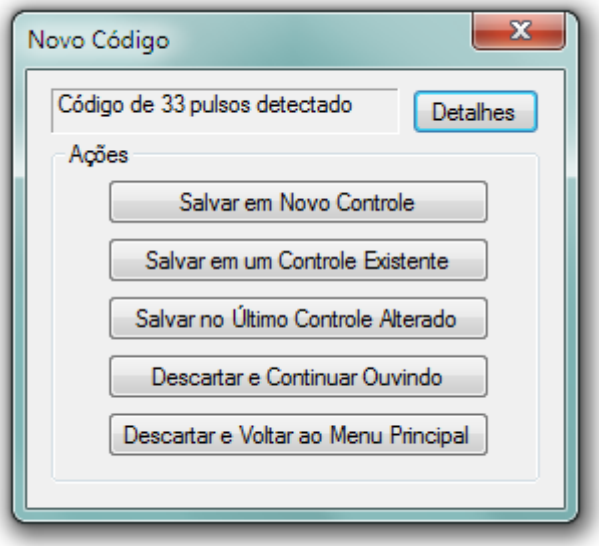
1. Na tela principal, selecione a opção “Aprender Códigos”.
2. Será aberta uma tela de espera, durante a qual o programa entrará no estado de detecção de códigos. Aponte um controle remoto para o sensor infravermelho

conectado à placa controladora e pressione um botão. É aconselhável pressioná-lo pelo tempo mais breve possível, para evitar que o código infravermelho referente ao comando se repita mais vezes do que o necessário. Para interromper a detecção e voltar à tela inicial, clique em “Cancelar”.

Figura 28. Tela de espera durante a detecção de códigos

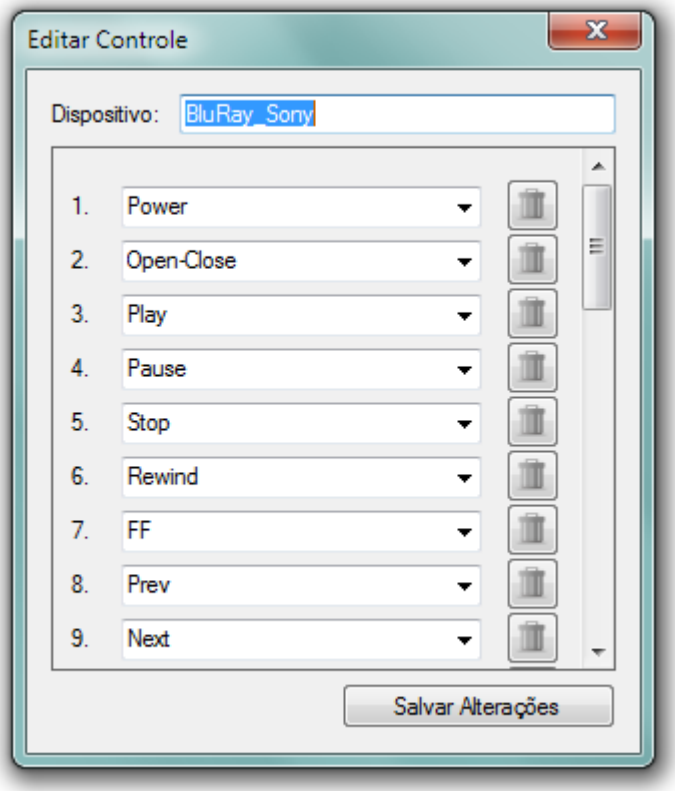
3. Ao prosseguir com a detecção de um comando, a tela “Novo Código” será mostrada. Se o comando for de um controle remoto que não foi armazenado no sistema anteriormente, selecione a opção “Salvar em Novo Controle” para abrir a tela de edição de controles remotos. Alternativamente, caso se esteja adicionando mais códigos a um controle existente, selecione a opção “Salvar em um Controle Existente” e selecione a partir da lista que aparecerá o controle remoto necessário. Pode-se ainda utilizar a opção “Salvar no Último Controle Alterado” caso um controle remoto tenha sido editado durante a mesma sessão de uso do programa. Isto é útil quando se está adicionando vários comandos de um mesmo controle, pois permite editá-lo sem precisar selecioná-lo da lista a cada adição de comando. Todas estas opções levarão à tela de edição de controles remotos. Pode-se também clicar em “Descartar e Continuar Ouvindo”, que retorna o programa à tela de detecção, ou “Descartar e Voltar ao Menu Principal”, que retorna à primeira tela, sem salvar o comando detectado em ambas as opções.

Figura 29. Tela de salvamento de códigos



4. Na tela de edição de controles remotos, insira um nome de dispositivo para o controle remoto (caso seja um novo) e um nome para o comando adicionado ou selecione um nome pré-definido a partir da lista que se abre ao clicar na pequena seta ao lado do campo de texto do nome do comando. Esta lista contém nomes comuns de comandos, como “Power”, “Channel [+/-]”, “Volume [+/-]” entre outros. Se estes nomes forem escolhidos, será possível tirar vantagem do mapeamento automático de comandos quando se estiver criando um controle virtual. Clique em “Salvar Alterações” para confirmar as mudanças no controle ou feche a janela para descartar o comando detectado e voltar à tela inicial.

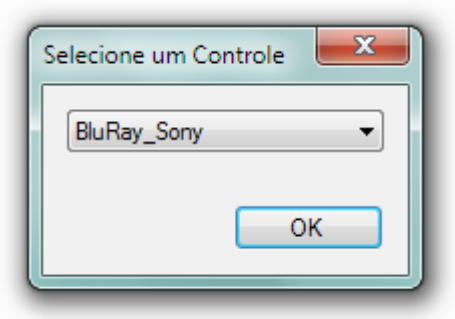
Figura 30. Tela de edição de controles remotos



Tarefa 3: Visualizar/Editar controles remotos

1. Na tela principal, selecione a opção “Ver Controles”.
2. Se não existirem controles remotos armazenados, uma mensagem contendo esta informação será mostrada e o programa retornará à tela inicial. Não será possível prosseguir enquanto não for adicionado pelo menos um novo controle. Se existirem controles, selecione um a partir da lista e clique em “OK”.

Figura 31. Tela de seleção de controles remotos



3. A mesma tela de edição de controles remotos usada na adição de comandos é mostrada, porém desta vez não há restrições sobre quais dados do controle se pode alterar. Para renomear o controle, insira o novo nome no campo de texto apropriado.

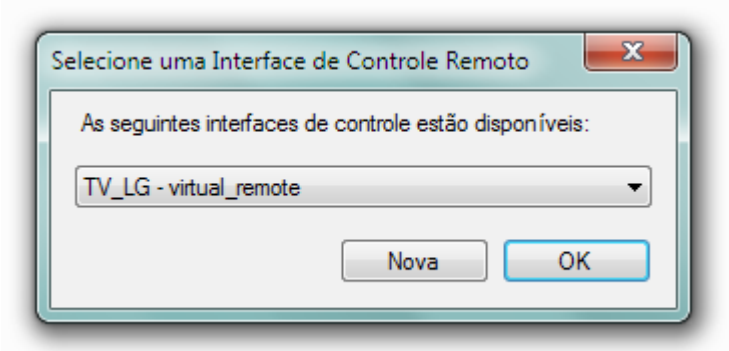
Não é possível salvar um controle sem um nome de dispositivo e nem com o mesmo nome de algum outro controle existente.

4. Para renomear comandos, insira um nome diferente no campo de texto ou selecione um dos nomes padrões a partir da lista que aparecendo ao clicar na seta ao lado. Não é possível salvar um código sem nome.
5. Para excluir comandos, clique no botão com o ícone de lixeira presente ao lado do nome do comando que se deseja excluir.
6. Clique em "Salvar Alterações" para aplicar as mudanças ou feche a janela para descartá-las e retornar à tela inicial.

Tarefa 4: Criar um controle remoto virtual

1. Na tela principal, selecione a opção "Mapear Comandos".
2. Selecione um dentre os controles remotos armazenados para criar um controle virtual para o mesmo. Se não houver nenhum controle armazenado, o programa exibirá uma mensagem com esta informação e voltará à tela inicial. Não será possível prosseguir enquanto não houver ao menos um controle remoto armazenado no sistema.
3. Na tela "Selecione uma Interface de Controle Remoto", clique em "Nova".

Figura 32. Tela de seleção de interfaces de controle

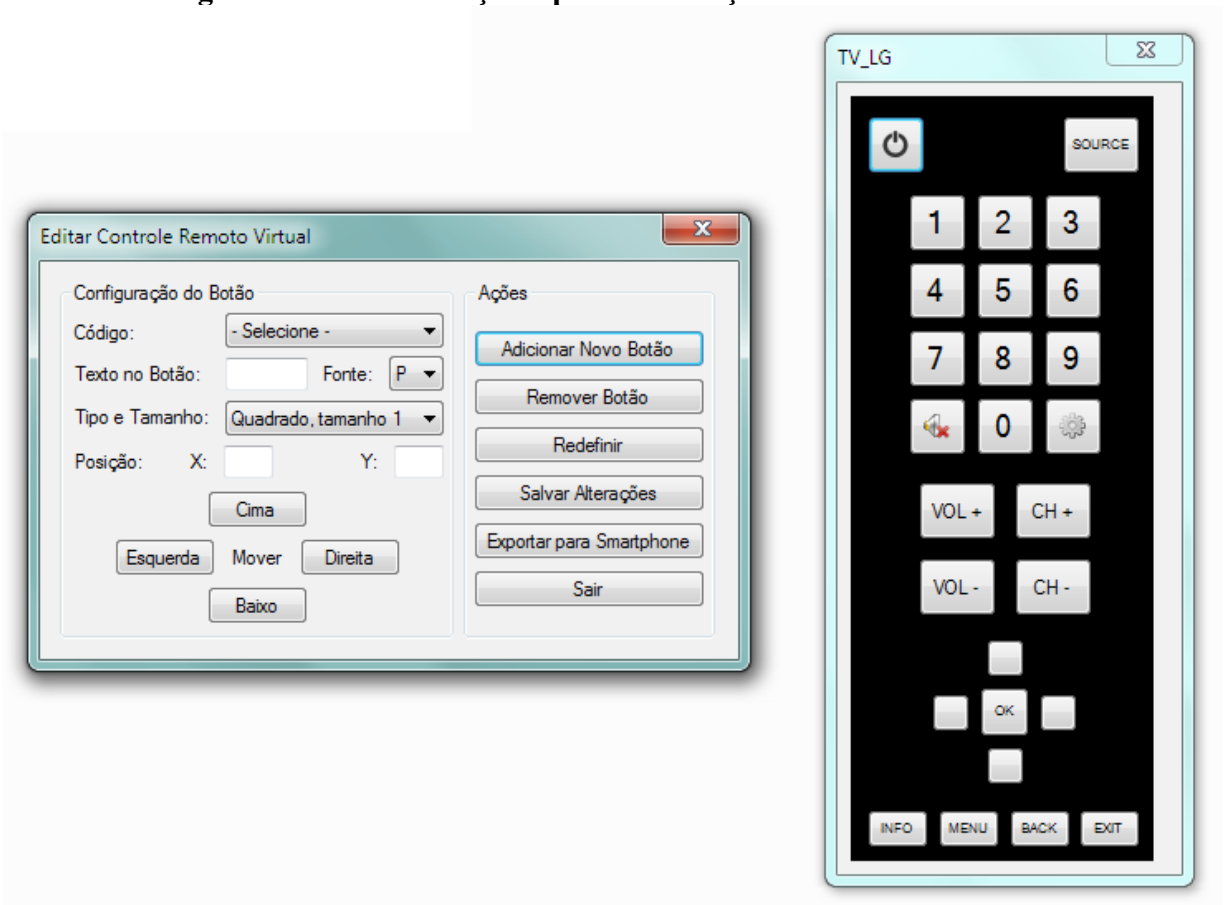


4. Na tela seguinte, selecione o tipo "Controle Remoto Virtual" e clique em "OK".

Figura 33. Tela de adição de interface de controle

5. Uma janela contendo um controle remoto virtual com alguns botões já adicionados aparecerá, junto de uma janela contendo alguns controles para edição de botões. Clique num botão do controle virtual para editá-lo ou adicione seus próprios botões clicando em "Adicionar Novo Botão".

Figura 34. Telas de edição e pré visualização de controles virtuais



6. Para remover um botão desnecessário, pode-se clicar nele para selecioná-lo e depois clicar em “Remover Botão”. Para remover todos os botões de uma vez, utilize a opção “Redefinir”.

7. Ao selecionar um botão para edição, seja um novo ou um já existente, as informações sobre o mesmo serão carregadas na janela de edição já aberta. Nela pode-se alterar várias características do botão, que incluem seu posicionamento no controle, o texto a aparecer sobre o botão, tamanho da fonte, formato (quadrado ou retangular), tamanho e, logicamente, o comando ao qual o botão deverá ser associado. Atualmente, não é possível associar uma imagem a um botão, então se os botões que contém imagens forem removidos, eles apenas poderão ser adicionados novamente criando-se um novo controle virtual.

8. Para associar um comando a um botão, após selecioná-lo, clique na lista “código” e selecione um entre os comandos disponíveis no controle remoto carregado. Para deixar um botão sem comando algum, apenas mantenha selecionada a opção padrão (“- Selecione -”). Se os nomes de códigos utilizados durante sua adição ao controle remoto foram os nomes padrões (aqueles existentes na lista para seleção), os comandos referentes a estes nomes serão automaticamente associados a seus respectivos botões. Para comprovar isto, basta clicar em um destes botões e observar o valor selecionado na lista “Códigos”, que será diferente do valor padrão. Por exemplo, se um código “liga/desliga” foi adicionado ao controle remoto e a ele foi dado o nome “Power”, ao selecionar o botão com o ícone de energia, o código Power já estará selecionado na lista.

9. Clique em “Salvar Alterações” para armazenar o controle virtual ou em “Sair” para descartá-lo e voltar à tela principal. Também é possível sair sem salvar as alterações fechando qualquer uma das duas janelas.

Tarefa 5: Editar um controle remoto virtual

1. Siga os dois primeiros passos do procedimento de criação de controles virtuais.

2. Na tela “Selecione uma Interface de Controle Remoto”, selecione uma das interfaces de controle existentes que sejam do tipo “virtual_remote” – se houver alguma – e clique em “OK”.

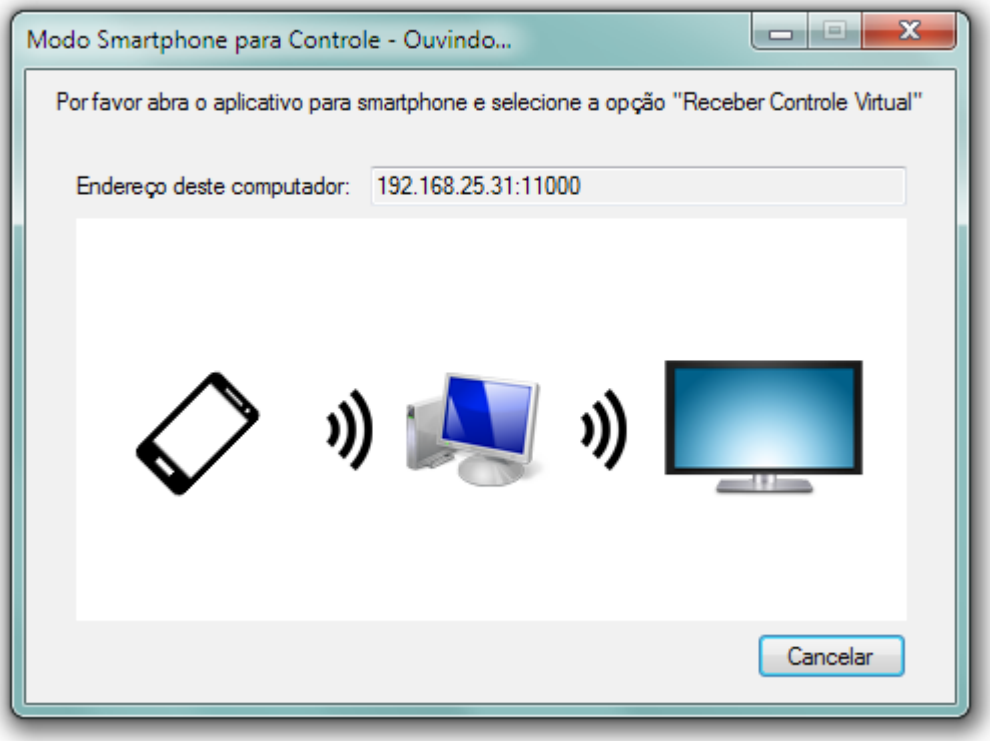
3. A interface escolhida será aberta para edição, com o programa mostrando em seguida as mesmas duas janelas de edição de botão e visualização do controle virtual que são exibidas durante a criação do mesmo.

4. Siga os passos do procedimento de criação de controles virtuais do passo 5 em diante. Se alguma das janelas for fechada ou o botão “Sair” for clicado, quaisquer alterações feitas no controle virtual são descartadas.

Tarefa 6: Enviar um controle remoto virtual para um dispositivo móvel Android

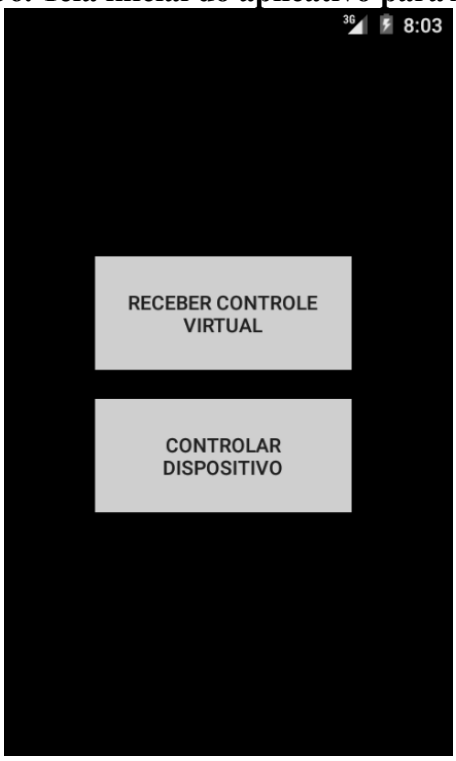
1. Instale o aplicativo IRConnectionApp em seu dispositivo Android.
2. Siga o procedimento para a criação de controle virtual até a etapa 8.
3. Após terminar de criar um controle virtual como desejado, selecione a opção "Exportar para Smartphone".
4. Uma tela contendo um endereço composto de IP e número de porta será mostrada. Esta é uma tela de espera que será exibida até que uma conexão com um dispositivo Android seja feita.

Figura 35. Tela de comunicação com o aplicativo para dispositivos móveis



5. Na tela principal do aplicativo para Android, selecione a opção "Receber Controle Virtual".

Figura 36. Tela inicial do aplicativo para Android



6. Na tela seguinte, digite o endereço mostrado na tela de espera do programa no computador e clique em “OK” para confirmar.

Figura 37. Tela de preenchimento de endereço para comunicação com o aplicativo central

7. O controle virtual será recebido e armazenado no aplicativo para Android e uma

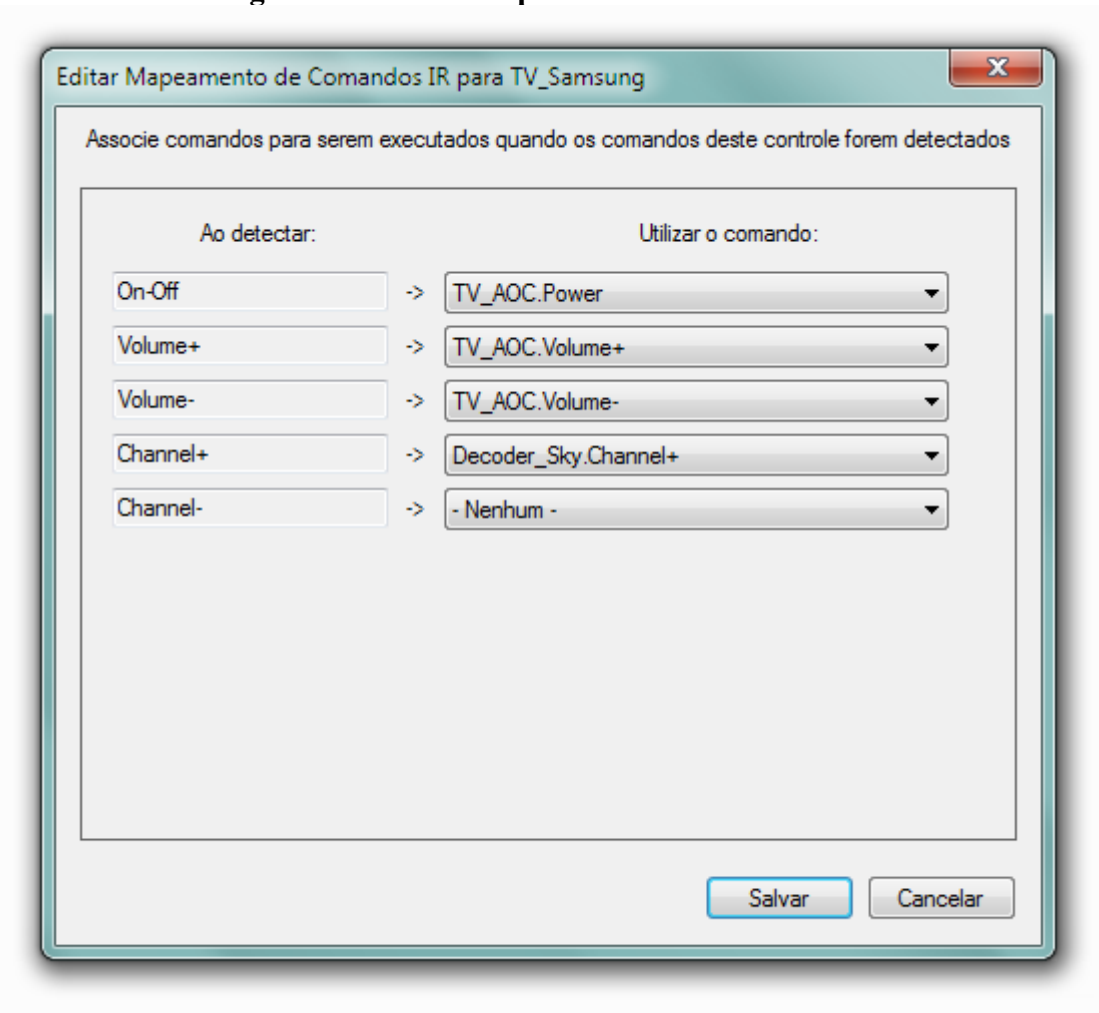
mensagem confirmando isto será mostrada na tela do aparelho caso a operação tenha sucesso. O programa de computador voltará à tela inicial.

Figura 38. Tela de comunicação com o aplicativo central exibindo mensagem de sucesso

Tarefa 7: Criar um mapeamento entre comandos de diferentes controles

1. Siga os três primeiros passos das instruções para a criação de controle remoto virtual descritas acima.
2. Na tela de adição de interface de controle remoto, selecione o tipo “Mapeamento IR para IR” e clique em “OK”.
3. Uma lista com os comandos do controle remoto selecionado aparecerá na coluna da esquerda e uma série de listas contendo os códigos de todos os outros controles aparecerá na coluna da direita. Para criar um mapeamento entre dois comandos, selecione a partir da lista da direita um comando para associar ao comando diretamente ao lado.

Figura 39. Tela de mapeamento de comandos IR



4. Para criar mais mapeamentos, repita o processo escolhendo comando a partir das listas da direita para os outros comandos do controle sendo editado que deseje mapear.

5. Ao terminar, clique em “Salvar”. Pode-se clicar e, “Cancelar” ou fechar a janela a qualquer momento para sair sem salvar os mapeamentos, voltando à tela inicial.

Tarefa 8: Editar um mapeamento entre comandos de diferentes controles

1. Siga os dois primeiros passos do processo de criação de controle virtual.

2. Na tela “Selecione uma Interface de Controle Remoto”, escolha uma dentre as interfaces de controle do tipo “infrared” existentes – se houver alguma – e clique em “OK”.

3. A mesma tela de associações entre códigos vista no processe de adição de mapeamento será mostrada. Altere, adicione ou remova quantas associações quiser e cliquem em “Salvar Alterações”. Se a janela for fechada, nenhuma alteração no conjunto de mapeamentos carregado será efetuada.

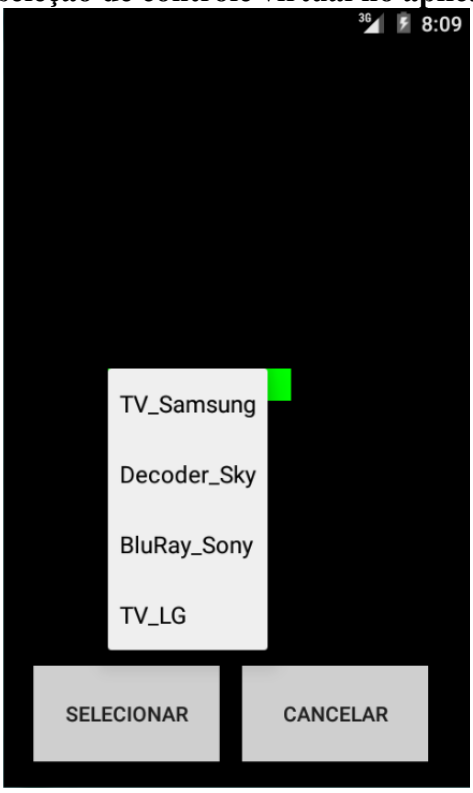
Tarefa 9: Controlar um aparelho eletrônico utilizando controle virtual

1. Na tela principal, selecione a opção “Controlar Dispositivo”.
2. Selecione um controle remoto existente para ver as interfaces de controles disponíveis para o mesmo. Se não houver um controle remoto armazenado, o programa o informará sobre isto e voltará à tela inicial.
3. Selecione a partir da lista de interfaces de controle alguma interface do tipo “virtual_remote”. Este tipo aparecerá ao lado do nome do controle remoto, como um item da lista. Se não houver nenhuma interface de controle armazenada para o controle remoto selecionado, o programa exibirá uma mensagem com esta informação e retornará à tela inicial.
4. A próxima tela será uma janela que carregará o controle virtual selecionado, idêntica à janela que mostra a pré-visualização do controle virtual durante sua edição, porém sem a janela de edição de botão acompanhando desta vez. Clique num botão para estabelecer uma comunicação com a placa controladora, que enviará o comando associado ao mesmo ao aparelho eletrônico, acionando-o da mesma forma que faria um controle remoto real.
5. Ao terminar, feche a janela do controle virtual para voltar à tela inicial.

Tarefa 10: Controlar um aparelho eletrônico utilizando um dispositivo móvel Android

1. Na tela principal, selecione a opção “Usar Smartphone como Controle”.
2. A mesma janela de espera contendo um endereço de conexão que pode ser vista durante o processo de exportação do controle virtual é mostrada. No aplicativo para Android, após realizar o procedimento de envio de um controle virtual para o dispositivo móvel pelo menos uma vez, selecione a opção “Controlar Dispositivo”. Se isto for feito sem que exista um controle virtual salvo, o aplicativo apenas mostrará uma mensagem com esta informação e não prosseguirá.

Figura 40. Tela de seleção de controle virtual no aplicativo para Android



3. Ao escolher um controle virtual existente, insira o endereço fornecido pelo programa no computador no campo de texto e clique em “OK”.
4. Neste momento, o controle virtual será carregado na tela do dispositivo Android. Ao pressionar um botão, uma conexão com o programa no computador é estabelecida é uma requisição de envio de comando é feita. O programa então envia o código infravermelho à controla que aciona o comando no aparelho eletrônico.

Figura 41. Tela para utilização de controle virtual no aplicativo para Android

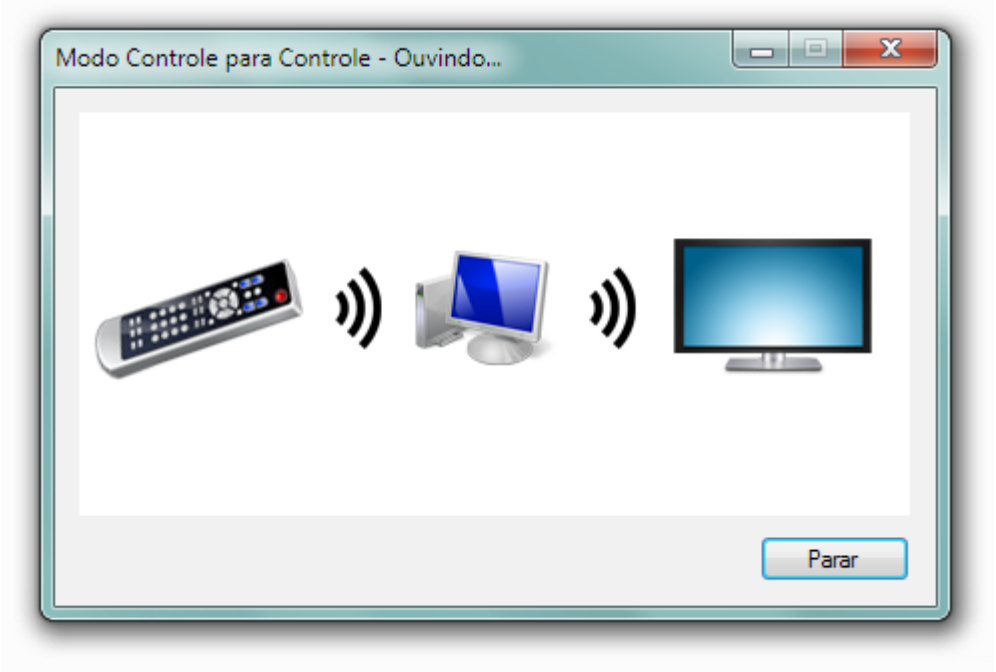
5. Ao terminar, clique em “Voltar” na tela do dispositivo móvel para voltar à tela inicial e em “Cancelar” no programa de computador para fazer o mesmo.

Tarefa 11: Controlar aparelhos eletrônicos através de controle remoto mapeado

1. Siga os três primeiros passos das instruções para controlar um aparelho através de controle virtual, porém no passo 3, selecione uma interface de controle do tipo “infrared”.

2. Uma tela que indica que o programa está no momento esperando por comandos vindos de um controle remoto aparecerá. Tendo em mãos o controle remoto selecionado, aponte-o para o sensor IR na controladora e pressione um dos botões cujo comando foi mapeado para outro. Se um comando que não foi mapeado for passado, nada acontecerá.

Figura 42. Tela de recebimento de códigos no modo de controle por mapeamento IR



3. O programa identificará o comando recebido, buscará pelo comando mapeado e enviará este último para a controladora, que o reproduzirá como se o próprio controle remoto que contém este comando estivesse sendo utilizado, acionando o comando no aparelho. Para terminar e voltar à tela inicial, clique em “Parar”.