INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE

CAMPUS NATAL ZONA NORTE

Guillherme da Silva Amaral

Victor Emanuel Ribeiro Silva

AUTOMAÇÃO RESIDENCIAL UTILIZANDO A PLATAFORMA ARDUINO E DISPOSITIVOS MÓVEIS

NATAL-RN

2017

Guillherme da Silva Amaral
Victor Emanuel Ribeiro Silva

AUTOMAÇÃO RESIDENCIAL UTILIZANDO A PLATAFORMA ARDUINO E DISPOSITIVOS MÓVEIS

Trabalho de conclusão de curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte como parte das exigências para a obtenção do título de Técnico em Eletrônica.

Orientador: Prof. Marcus Vinicius Araújo Fernandes

NATAL-RN

2017

Guillherme da Silva Amaral
Victor Emanuel Ribeiro Silva
AUTOMAÇÃO RESIDENCIAL UTILIZANDO A PLATAFORMA ARDUINO E DISPOSITIVOS MÓVEIS/ Guillherme da Silva Amaral
Victor Emanuel Ribeiro Silva. – NATAL-RN, 2017-
99 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Marcus Vinicius Araújo Fernandes

Trabalho de Conclusão de Curso – INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS NATAL ZONA NORTE, 2017.

Guillherme da Silva Amaral
Victor Emanuel Ribeiro Silva

AUTOMAÇÃO RESIDENCIAL UTILIZANDO A PLATAFORMA ARDUINO E DISPOSITIVOS MÓVEIS

Trabalho de conclusão de curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte como parte das exigências para a obtenção do título de Técnico em Eletrônica.

Trabalho de conclusão de curso aprovado em ____ /____/____, pela seguinte banca examinadora:

BANCA EXAMINADORA

Prof. Marcus Vinicius Araújo Fernandes
Orientador

Prof. Pedro Ivo de Araujo do Nascimento
Coordenador do curso de Eletrônica

Prof. Arthur Salgado de Medeiros
Professor do curso de Eletrônica

NATAL-RN
2017

Agradecimentos

Agradecemos ao IFRN - Campus Zona norte por disponibilizar o Laboratório de Eletrônica para utilizarmos os equipamentos que foram necessários no desenvolvimento do trabalho.

A todos os professores e servidores do campus que fizeram parte da nossa formação e em especial aos que lecionaram as disciplinas técnicas.

As nossas famílias e amigos por sempre serem críticos sobre nossas decisões e nos incentivar a sempre buscar o melhor.

"Quando algo é suficientemente importante, você faz mesmo que as chances não estejam a seu favor" (Tradução nossa).

(Elon Musk)

Resumo

O objetivo deste trabalho é apresentar o conceito de domótica, também conhecido como automação residencial, e a partir disso desenvolver um sistema que aplique esse conceito utilizando materiais de fácil obtenção e baixo custo. O foco aqui é como a comunicação entre o usuário e os eletrodomésticos dentro de uma residência ocorre, para isso, foi implementado um sistema que permite o controle de aparelhos de DVD e televisores, uma espécie de controle universal. A priori, as possibilidades de uso do sistema desenvolvido podem parecer poucas e limitadas, entretanto ele serve como base para criação de outros projetos, tais como: Controle de temperatura; iluminação; alarmes e outros, sendo necessário apenas modificar a resposta do sistema, ao invés de ele enviar um sinal na faixa do infravermelho para mudar o canal da televisão ele envia um sinal para controlar a iluminação, por exemplo.

Palavras-chaves: Arduino, Domótica, Ethernet Shield, Bluetooth, App Inventor.

Abstract

The aim of this term paper is to introduce the concept of domotic, also known as house automation, and develop a system that apply these concept using materials that can be easily get and low cost. The focus here is how the comunication between the user and the home appliances occur, for this, a system was implemented and he allow to control devices like televisors and DVD's, as a universal control. A priori, the possibilities of use of the system may seem few or limited, however it can be used as base to create other projects to control temperature; ilumination, alarms and many others, just being necessary modify the system answer, instead to send a sinal in a infrared bandwidth to change a television channel, he send a sinal to control the ilumination, for example.

Key-words: Arduino, Domotic, Ethernet Shield, Bluetooth, App Inventor.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AJAX	Asynchronous Javascript and XML
EM	Eletromagnetic
HTML	HyperText Markup Language
ICSP	In-Circuit Serial Programming
IDE	Integrated Development Environment
IR	Infrared
ISP	In-System Programmer
LED	Light-Emitting Diode
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
SCK	Serial Clock
SD	Secure Card
SPI	Serial Peripherical Interface
SS	Slave Select
TCP/IP	Transmission Control Protocol/Internet Protocol
URL	Unifor Resource Locator

Sumário

1 Introdução

A domótica, ou automação residencial, é o uso da tecnologia a serviço do conforto dos moradores de uma casa. Apesar de ser um conceito simples, a domótica toma forma ao disponibilizar de forma simples ao usuário doméstico tecnologias que foram criadas e vem sendo aperfeiçoadas a pelo menos trinta anos (BIONDO, 2011).

A proposta deste trabalho é fornecer um meio de reunir controles de eletrodomésticos e acionamento de interruptores ou tomadas em um sistema que pode ser acessado por dispositivos móveis, como um tablet ou smartphone. Para isso, fez-se o uso de Arduino, Ethernet Shield, módulos Bluetooth e LEDs de infravermelho.

Pelo fato de existir inúmeros eletrodomésticos de tipos e fabricantes diferentes, o trabalho foca em desenvolver controles universais digitais apenas para televisores e aparelhos de DVD, e também para acionar relés que irão fazer o acionamento de cargas, como lâmpadas ou aparelhos ligados a tomadas, sendo que apenas no último é necessário realizar ajustes nas instalações elétricas.

1.1 Objetivos

O objetivo deste trabalho é desenvolver uma plataforma *open source* e *open hardware* que permita a automação no âmbito de uma residência, ou seja, controlar televisores; aparelhos; iluminação ou tomadas da casa. O acesso será feito via internet através de um navegador *web* ou via bluetooth por meio de um aplicativo para o sistema operacional Android. Para isso, serão utilizados componentes eletrônicos de fácil obtenção e que possuem uma grande comunidade em fóruns online que dão suporte, com o intuito de que, caso sejam encontradas dificuldades na implementação deste projeto por terceiros, seja fácil encontrar uma solução.

1.2 Organização

O presente trabalho está organizado em cinco capítulos. O capítulo 2 apresenta todo o embasamento teórico assim como todas as ferramentas utilizadas para o desenvolvimento do trabalho. O capítulo 3 descreve os métodos utilizados para conceber o projeto assim como os testes feitos para comprovar o funcionamento. O capítulo 4 contém algumas ressalvas sobre o este trabalho assim como os resultados obtidos. O capítulo 5 apresenta as conclusões que foram obtidas na realização do presente trabalho.

2 Referencial Teórico

2.1 Microeletrônica e Domótica

O domínio da eletrônica possibilitou avanços tecnológicos jamais vistos na história da humanidade e tudo isso começou com Fleming, em 1904, com a invenção do primeiro dispositivo amplificador, o "tubo de vácuo" ou mais comumente conhecido como "válvula eletrônica". A partir desse período várias outras invenções começaram a surgir até que, em 1947, John Bardeen e Walter Brattain junto ao seu supervisor, William Shockley, desenvolveram o transistor de silício, na Bell Telephone Laboratories (WESTE; HARRIS, 2011). O transistor tem sido aprimorado com o tempo e até hoje é amplamente utilizado nas mais recentes tecnologias.

Em 1958 o primeiro circuito integrado (CI) foi desenvolvido simultaneamente por Jack Kilby na Texas Instruments e por Noyce e Moore na Fairchild Semiconductor, de forma independente. Essa invenção foi o começo de uma revolução na microeletrônica (RASHID, 2011).

O primeiro flip-flop construído em 1958 possuía apenas dois transistores e apenas 50 anos depois, em 2008, o microprocessador da Intel chamado Itanium continha mais de dois bilhões de transistores e 16 gigabytes de memória flash que continha mais de quatro bilhões de transistores (WESTE; HARRIS, 2011). Jamais, nenhuma outra tecnologia se manteve relevante por tanto tempo.

Tal façanha se deve ao fato da contínua diminuição no tamanho do transistor e no aperfeiçoamento do processo de fabricação, isso permite que os circuitos consumam menos energia, custem menos e tenham melhor desempenho (KANG; LEBLEBICI, 2003). A título de comparação, o poder de processamento que uma vez já foi utilizado em um supercomputador hoje cabe dentro de um smartphone. Esse conjunto de fatores não só revolucionou a eletrônica, mas também toda a sociedade que usufrui das tecnologias que se tornaram possíveis com o uso do transistor.

Todo esse avanço tecnológico levou a criação de processadores, memórias e dispositivos programáveis, sendo assim, tornou-se possível a criação de máquinas pré-programadas para executar uma tarefa pré-determinada, diz que existe um "sistema embarcado" na máquina. Não demorou muito para as indústrias começarem a utilizar cada vez mais robôs para agilizar o processo de fabricação, deixando a linha de produção automatizada.

Continuando com os avanços tecnológicos e com circuitos integrados custando cada vez menos e ficando cada vez menores proporcionou a popularização da eletrônica, pois com o baixo custo e fácil acesso a componentes eletrônicos muitas pessoas podem adquirir

esses produtos e desenvolver seus próprios projetos.

Mais recentemente a automação ultrapassou os limites da indústria e chegou na residências, surgindo assim a domótica, que é a aglutinação de duas palavras, a primeira vem do latim “Domus” que quer dizer “Casa”, a segunda é a palavra “Robótica” (CARDOZO; GASPAS; FONTANA, 2013).

2.2 Bluetooth

2.2.1 O Que É

Bluetooth é um protocolo padronizado para comunicação a curta distância via sinais de rádio que operam na largura de banda de 2.402 a 2.485Ghz (SPARKFUN, 2013). A tecnologia foi criada inicialmente para substituir os cabos do padrão RS-232, comumente usados nas portas seriais de computadores. Apesar disso, seu grande sucesso e aceitação fez com que ela tomasse outro rumo. Hoje, a SIG (Bluetooth Special Intrest Group) é o grupo de 30.000 companhias membros que incluem módulos Bluetooth em seus produtos, além de ser responsável pelo desenvolvimento e licenciamento das especificações do protocolo. A proposta atual é de tornar a conexão sem fio entre dispositivos e sensores rápida ao mesmo tempo que opera em baixa potência.

O Bluetooth opera na mesma banda de frequência ISM (UNION, 1989) na qual também está o protocolo de rádio frequência denominado Wi-Fi, entorno de 2.45Ghz. ISM (Industrials, Scientific and Medical Radio Bands) são bandas de rádio frequências reservadas internacionalmente para propósitos industriais, científicos e medicinais tais como: serviços de comunicação aeronáutica, localização geográfica ou exploração terrestre via satélite. Nestes contextos, as ondas são emitidas com intensidade capaz de causar interferência eletromagnética significativa noutras comunicações de rádio que usam a mesma frequência. Por conseguinte, foram separadas oficialmente as bandas nas quais esses serviços devem trabalhar.

2.2.2 Adaptative Frequency-Hopping

Apesar de operar numa das bandas reservadas pela ISM o protocolo Bluetooth usa uma técnica de variação aleatória de frequência chamada Adaptive Frequency-Hopping (AFH), o que permite o funcionamento de várias comunicações sem fio usando a mesma banda de sinal de rádio.

Dentro de seu espectro de operação (de 2.402 a 2.485GHz), os módulos Bluetooth “saltam” aleatoriamente entre os 79 canais de bandas de frequências iguais. Quando conectados, os envolvidos na rede trocam rapidamente a frequência do sinal em que estão conversando. Redes bluetooth próxima não são afetadas significativamente por outras,

uma vez que é extremamente improvável que qualquer dispositivo esteja operando na mesma frequência que outro ao mesmo tempo. Ainda assim, por mais que vários módulos saltem para o mesmo canal ao mesmo tempo, a interferência será minúscula pois todos mudaram, em uma fração de segundo, para outro.

Os saltos também são adaptáveis, significando que os canais nos quais ocorrem interferência são identificados e deixam de ser escolhidos futuramente. Por isso, é possível receber/enviar dados via Bluetooth e via Wireless LAN ao mesmo tempo (HODGDON, 2003).

2.2.3 Como Funciona

Para entender uma Piconet (rede formada via Bluetooth) é preciso entender o modelo mestre/escravo, que define um conjunto de regras sobre como será feita a transmissão dos dados.

O aparelho cujo módulo Bluetooth está configurado para atuar como Mestre pode se conectar a outros com papel de escravos e trocar informações com os que estão conectados a ele. O mestre trabalha recebendo dados ou enviando comandos para até sete escravos simultaneamente. Por outro lado, um escravo só pode se conectar a um mestre e somente com este ele troca mensagens. Logo, mesmo que haja mais de um escravo na mesma piconet, eles não se comunicam diretamente. Caso isso seja necessário, um escravo deve dar a informação ao mestre e este a entregará ao destinatário na mesma rede. Esse arranjo é similar à topologia física de uma rede estrela, usada na conexão de computadores. Nesta ilustração, o hub central da rede sempre é o módulo mestre, e os demais nós são os escravos.

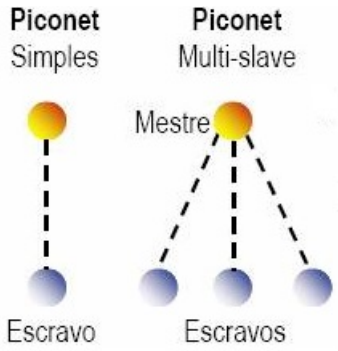


Figura 1 – Esquema de duas piconets
Fonte: https://www.gta.ufrj.br/grad/06_1/bluetooth/topologia.htm

Cada dispositivo Bluetooth tem um endereço único que consiste numa palavra de 48 bits, geralmente escrito em Hexadecimal no chip emissor de sinal. A primeira metade (os vinte e quatro bits mais significativos) caracterizam o fabricante. Logo, cada empresa tem sua sequência definida. Sendo assim, o que torna cada aparelho único é a outra

metade menos significativa da palavra. Concluindo, todo chip Bluetooth consta de doze dígitos hexadecimais formando seu endereço, dentre estes, os seis primeiros identificam o fabricante e os outros seis, é o endereço único de cada dispositivo.

Por exemplo, numa das placas ZS-040 usadas no projeto está gravado o número 143BGQK518CY. A sequência 143BGQ é encontrado impresso em todos os chips Bluetooth da CSR, fabricante da placa. Porém, dentre todos produtos da empresa, o número K518CY pertence exclusivamente a este chip.

Para facilitar seu uso, os módulos, como o contido no ZS-040, têm 248 Bytes reservados para guardar um apelido configurável pelo seu dono. Deste modo, o usuário livra-se de memorizar uma sequência de dígitos sem significado.

2.2.4 Conectando

Uma rede Bluetooth é criada quando pelo menos um escravo se conecta a um mestre. Para que isso seja feito, todos os dispositivos seguem uma sequência de passos para identificar se há outros nas proximidades e quais seus papeis, como cita (DZIWIOR, 2004). Esse processo existe com o intuito de reduzir a interferência de dispositivo fora da rede durante a transmissão de dados com os que estão dentro.

Inquiry é o modo de funcionamento em que dispositivos Bluetooth ficam por padrão sempre que estão fora de uma rede. Nele, é feito o processo de busca e descoberta de outros aparelhos nas proximidades. Dispositivos no papel de mestre e que estão no Inquiry Mode, enviam um sinal a fim de obter resposta dos outros dispositivos próximos. O sinal contém a informação sobre quais tipos de aparelhos devem responder ao chamado. O documento com as especificações do Bluetooth define o código que deve ser enviado de modo a explicitar o que está sendo procurado para fazer conexão: impressoras, celulares, mouses e etc. Já um dispositivo no Inquiry Mode e configurado para ser escravo, liga seu receptor entrando periodicamente no estado de busca. Quaisquer escravos próximos que tenham recebido o código, responde ao mestre com o pacote contendo obrigatoriamente seu endereço, outras informações, como apelido, são opcionais.

A partir daqui, pode haver ou não uma interface gráfica para que o usuário escolha, em uma lista de aparelhos descobertos, qual deve se conectar. Caso isso não exista, após coletar endereços e ser definido com quem será feita a piconet, os dispositivos entram no modo Paging. Nesse modo, o dispositivo mestre gera uma sequência de canais (bandas de frequência) baseada nos dígitos do endereço do escravo, que realiza a mesma ação. Isso porque na rede Bluetooth as mensagens são enviadas na faixa de 2.402 a 2.485Ghz, que por sua vez está dividida em 79 canais. Como os dispositivos alternam de canal 1600 vezes por segundo, ambos os agentes compartilham a chave da sequência de canais que deve ser seguida. Para que a comunicação seja possível, tanto o mestre como o escravo devem

se comunicar sincronizadamente pelo mesmo canal. No Paging Mode, ambos aparelhos se baseiam na mesma chave (endereço do escravo) para gerar a sequência de canais que deve ser seguida durante o Paging. O mestre agora envia várias vezes mensagens indicando que está pronto para parear. Todos os escravos que receberem-na responderão indicando o mesmo. No próximo passo, o mestre cria sua própria sequência usando seu endereço como chave e o envia para o escravo, que deve gerar a mesma ordem de canais.

Ao final do Paging, os módulos entram no estado Connection. Enquanto conectados, dispositivos podem trabalhar em quatro modos. **Active Mode**: habilita o envio e recebimento de dados a qualquer momento. **Sniff Mode**: visando uma economia de bateria, o dispositivo busca por mensagens regularmente em um certo intervalo de tempo. **Hold Mode**: mestres podem ordenar que escravos fiquem neste estado, fazendo-os ficar inativos por um determinado espaço de tempo. **Park Mode**: Ao receber o comando "park", um escravo permanece inativo até que o mestre reenvie mensagem revertendo a inatividade

2.2.5 Perfis

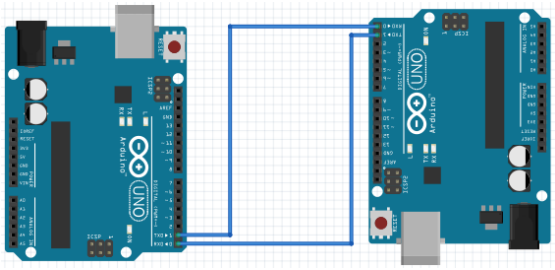
Perfis Bluetooth são protocolos adicionais objetivando definir com mais clareza que tipo de informação um dispositivo está transmitindo. Um perfil provê padrões que os fabricantes devem seguir para permitir o uso do Bluetooth de uma maneira compatível com todos os dispositivos. Serial Port Profile (SPP) é um dos mais comuns. Através dele, os dispositivos podem trocar dados como se estivessem conectados pelos terminais TX/RX fazendo comunicação Serial. Basta que, ao invés de fios, módulos sejam fixados nas portas seriais de cada um.

Para deixar claro, enquanto as especificações Bluetooth dizem como a tecnologia funciona, os perfis dizem como ela deve ser usada (GROUP, 2016).

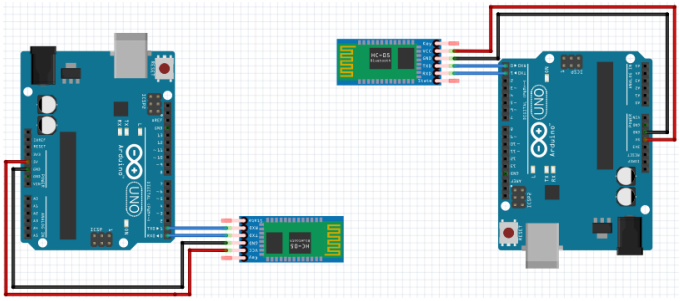
2.3 Módulo Bluetooth

O módulo usado no projeto foi o HC-05, escolhido por ser capaz de trabalhar como mestre ou escravo e por haver grande popularidade entre projetistas também adeptos do Arduino. Isso se deve ao fato do módulo suportar o Serial Port Profile e também por ser possível adquiri-lo embarcado na placa ZS-040, que é uma interface entre o módulo e o Arduino.

Antes de usar qualquer módulo, deve-se configurá-lo. No caso do HC-05, é necessário realizar uma série de passos para que ele possa receber comandos via serial do Arduino. Todas as instruções devem começar com "AT", por isso a maioria das fontes se referem a este modo por "AT Mode". Há várias formas de realizar esta tarefa dependendo

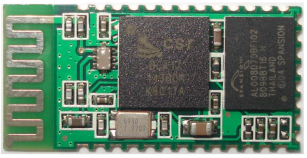


(a) Comunicação com fios

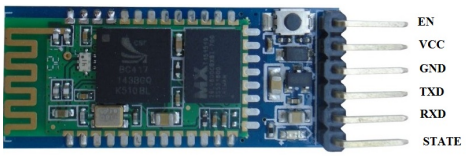


(b) Comunicação sem fios

Figura 2 – Os circuitos em (a) e (b) são equivalentes.
Fonte: Propriedade dos autores



(a) HC-05



(b) ZS-040

Figura 3 – Módulo Bluetooth HC-05 e a placa ZS-040
Fonte: Propriedade dos autores

do modelo da placa. Uma das possíveis possibilidades segundo (CURREY, 2015) será descrita a seguir.

- Com a placa desligada, apertar o botão acima do pino EN e ligá-la mantendo-o pressionado. O LED do HC-05 deve começar a piscar lentamente, indicando que está no modo AT
- Fazer upload no Arduino do código no apêndice A
- Conectar os pinos RX/TX do Arduino respectivamente nos pinos TX/RX do módulo bluetooth e ligá-lo numa saída 5V

- A partir daí o módulo pode receber os comandos AT na Serial do Arduino, como os da lista a baixo

AT + ORGL (Reseta o módulo para a configuração padrão)
AT + RMAAD (remove dispositivos anteriormente pareados)
AT + ROLE = 1 (define o modo de operação do módulo como MESTRE)
AT + RESET (Reset do módulo após a definição do modo de operação)
AT + CMODE = 1 (Permite a conexão automática com qualquer endereço)
AT + INQM = 0,5,10 (Modo de varredura, procura por 5 dispositivos e para a busca após 10s)
AT + PSWD = 1234 (define a senha do módulo mestre, na qual deve ser a mesma do módulo escravo)
AT + INIT (inicializa o perfil para transmissão/recepção)
AT + INQ (inicializa a varredura por dispositivos Bluetooth)

Além de atribuir valores às variáveis, pode ser feita apenas a leitura do estado atual escrevendo o comando sem o operador "=" e tudo que vem após. Exemplificando, para saber com qual papel o HC-05 está trabalhando, envia-se o comando AT+ROLE pela serial do Arduino e ele responderá com 0 caso esteja como escravo ou 1 se estiver como mestre.

A senha configurada de fábrica é o número 1234. Essa identificação serve para selecionar com quais dispositivos o módulo mestre fará conexão. Por exemplo, quando uma placa configurada como mestre for ligada numa fonte de 5V, o HC-05 embutido na mesma se conectará automaticamente somente as outras placas escravas cuja senha é igual a sua.

2.4 Duplex

Toda comunicação feita na piconet idealizada para o projeto se resume em quatro caracteres, note que o módulo escravo nunca é requisitado para falar. Redes na qual só um dos interlocutores falam por vez são chamadas de Half-duplex. Isso porque os módulos comprados não foram construídos para falar e ouvir ao mesmo tempo e, por consequência, não apresentam desempenho esperado quando operam de tal forma. Já dispositivos Bluetooth como os dos celulares atualmente podem trabalhar em redes chamadas Full-Duplex, isto é, transferindo e recebendo arquivos ao mesmo tempo.

2.5 Arduino

2.5.1 O Que É

O Arduino começou como um projeto do professor Massimo Banzi desenvolvido no Interaction Design Institute (Instituto de Design de Interação) em Ivrea, na Itália. No ano de 2005, em conjunto com colegas de trabalho, Banzi teve a ideia de desenvolver sua própria placa de prototipagem eletrônica que possibilitasse atividades escolares de forma a ter um orçamento mais baixo do que o proporcionado pelos sistemas similares da época. Eles formularam uma plataforma na qual seus estudantes pudessem facilmente criar projetos de eletrônica baseados em micro controladores baratos (CULTURA, 2010).

O modelo usado nesse trabalho, o Arduino UNO revisão 3, é uma plataforma de prototipagem eletrônica de hardware livre projetado com o micro controlador Atmega328, fabricado pela Atmel. Tem suporte a entradas e saídas analógicas e digitais além de contar com um ambiente de desenvolvimento integrado (IDE) feito pela equipe criadora do Arduino. Este, é programado em uma linguagem própria, uma variante do C/C++.

Um grande diferencial do Arduino é o fato da sua placa ser open-source. Tanto os esquemas das placas quanto os códigos de aplicações são disponibilizados livremente na internet, podendo ser alterados, redistribuídos, desenvolvidos e aperfeiçoados por qualquer um.

Devido à facilidade de manuseio, considerando-se também o vasto suporte a programadores iniciantes, bem como os constantes aperfeiçoamentos, tornaram a plataforma Arduino uma excelente ferramenta para projetos de nível iniciante.

2.5.2 Memória Interna

Para funcionar corretamente, deve ser feito o upload do código em seu microprocessador, que fará com que ele execute os processos necessários. O Arduino armazena esse código em uma das três memórias internas: a Flash, a SRAM e outra EEPROM, todas discorridas a seguir.

A memória **Flash** é responsável por armazenar o programa que será executado pelo Arduino assim que ligado. O tamanho máximo de acordo com o datasheet do Atmega328 é de 32 kBytes. Esses sketchs podem ser gravados e substituídos até um certo limite de vezes variando para cada modelo, no caso do contido no modelo UNO, esse número é de 10.000. Esta limitação ocorre por causa da deterioração interna do chip durante o processo de apagar e gravar um novo código. Uma característica dessa memória é que, mesmo sem alimentação de uma fonte de energia, o código permanece salvo.

SRAM (Static Random Access Memory), é a memória volátil usada pelo programa para guardar e manipular suas variáveis. Na placa de modelo UNO, a SRAM conta com

nada mais que 2k Bytes de espaço. Tão pouco que o torna inviável para projetos um pouco mais complexos.

EEPROM (Electrically-Erasable Programmable Read-Only Memory), onde são gravadas as informações não voláteis. É o espaço controlado pelo programador, que por sua vez usa-o para gravar valores numéricos duráveis. Tais dados nunca são apagados, apenas sobrescritos. Podem ser acessados ilimitadas vezes, porém sobrescritos em torno de 100.000 vezes. A capacidade da EEPROM do Arduino é de 1kB, para maiores armazenamentos de dados recomenda-se trabalhar com um módulo para cartões de memória.

Apesar de seu uso poder facilitar bastante o curso do projeto, a utilização desse equipamento não foi viável. A biblioteca "SD.h", que implementa as funções de leitura e escrita de texto (ARDUINO, 2005), ocupa um terço da memória destinada à programas. Em contraste com a "EEPROM.h" que, apesar da menor capacidade de armazenamento, realiza papel semelhante utilizando bem menos.

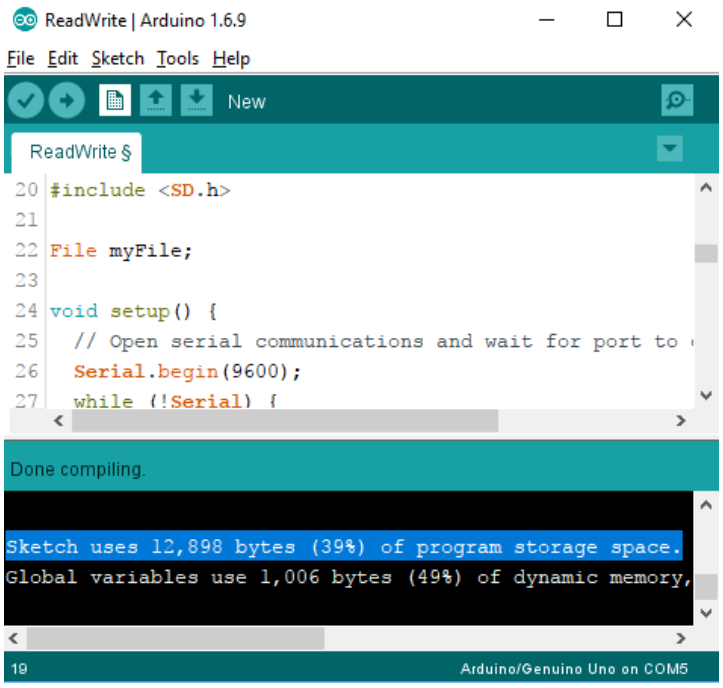


Figura 4 – Exemplo "ReadWrite" da biblioteca "SD.h"
Propriedade dos autores

Por outro lado, para possibilitar aplicações com códigos HTML pesados, geralmente Ethernet Shields são fabricados com uma entrada de cartão microSD embarcada. Deste modo, a página pode ser guardada no cartão e a memória de programa fica ocupada apenas com as poucas instruções seguidas pelo Arduino, ao mesmo tempo que possibilita o uso da biblioteca "SD.h".

2.5.3 IDE

É o ambiente de programação desenvolvido pelos mesmos criadores da placa. Foi feito com a ajuda de um dos criadores do programa Processing, também desenvolvido no MIT. Por consequência, ambos têm praticamente mesma interface e seguem o mesmo paradigma, uma função de configuração e um laço infinito. O laço serve bem ao propósito do micro controlador, que vai desde controlar produtos ou periféricos até automatizar uma ou várias tarefas ao mesmo tempo. Em ambas aplicações, é necessário o controlador que é programado para realizar a mesma tarefa quantas vezes forem necessárias, o que dificilmente sabe-se.

Todos programas para Arduino devem conter as duas funções principais: setup() e loop(). A primeira é executada assim que a placa é ligada, antes de qualquer outra função. Por isso geralmente é usado para configuração dos pinos de entrada/saída da placa e declaração das variáveis. Já a segunda, executada exatamente após o setup(), é um laço infinito, isto é, o bloco de comandos recomeça sempre que ele chega ao fim. A título de curiosidade, é possível chamar uma dessas funções no escopo da outra.

2.6 Controles Remotos

Dentre as várias formas possíveis de enviar e receber comandos remotamente, a mais viável para as fabricantes de eletrodomésticos são as ondas EM no espectro infravermelho. Esses sinais geralmente são enviados por circuitos eletrônicos dispostos de um emissor de radiação eletromagnética, conhecidos como controles remotos. Neles, códigos numéricos representam ações, tais como: ligar, desligar, aumentar ou diminuir volume. Ao apertar uma tecla do controle, este gera o sinal elétrico do código e o envia em forma de luz através do LED infravermelho, como descreve (MONTARROYOS, 2002). Em seguida, o LED receptor no aparelho realiza o processo inverso (decodificando a luz para sinais elétricos) e executa a ação. Como é do conhecimento de todos, um controle não funciona para qualquer eletrodoméstico. Nem um especificamente de TV funciona sequer em todas as TVs. Isso se deve ao fato de existir diferentes protocolos de comunicação em produtos de fabricantes distintos. Um exemplo é o protocolo NEC (LIMITED, 2013), presente no controle Samsung usado nos testes do projeto.

2.7 App Inventor

2.7.1 O Que É

O MIT App Inventor foi a plataforma usada para a programação do aplicativo ArduHouse. Foi criado pelo time liderado por Hal Abelson, professor do Massachusetts Institute of Tecnology (MIT), e Mark Friedman, funcionário do Google, em 2009 e desde

então é mantido pela equipe do MIT's Center for Mobile Learning. O site possibilita que os novatos à ciência da computação possam criar seus próprios aplicativos para o sistema operacional Android através de sua interface gráfica, que substitui a linguagem de programação baseada em textos (APPINVENTOR, 2012).

Isto é, todo o processo de criação da aparência é feito posicionando na tela do celular modelos pré-definidos dos componentes que devem constituir a parte visual do aplicativo. O código, seguindo a tendência adotada pelas outras plataformas de programação criadas no MIT (e.g. Scratch e StarLogoTGN), é feita em blocos. Deste modo, o criador deve se preocupar apenas com a sequência lógica do programa e a posição dos elementos visuais. Uma vez que o código é representado ilustrativamente, não é preciso aprender uma linguagem e, por consequência, não ocorrem erros de sintaxe.

Concluindo, o site é perfeito para quem está afim de testar ideias e torná-las, de modo rápido e prático, em um aplicativo, mesmo sem ter todos os gastos e aptidões que Apps tradicionais exigem. Além disso, a plataforma tem suporte à comunicação Bluetooth e Wi-Fi. Por estes motivos, ela foi escolhida.

2.7.2 Interface

Ao abrir ou criar um novo projeto no App Inventor, o usuário é apresentado à uma das duas telas de desenvolvimento: Designer a Blocks. Na primeira, será construído a interface do aplicativo; na segunda, o código em blocos. É recomendado começar sempre pelo visual do aplicativo. Por fim, cada tela está organizada em seções de acordo com sua função. A parte de Design tem a Palette, Viewer, Componentes, Media e Properties. Já em Blocks, está uma seção de mesmo nome, Media e Viewer.

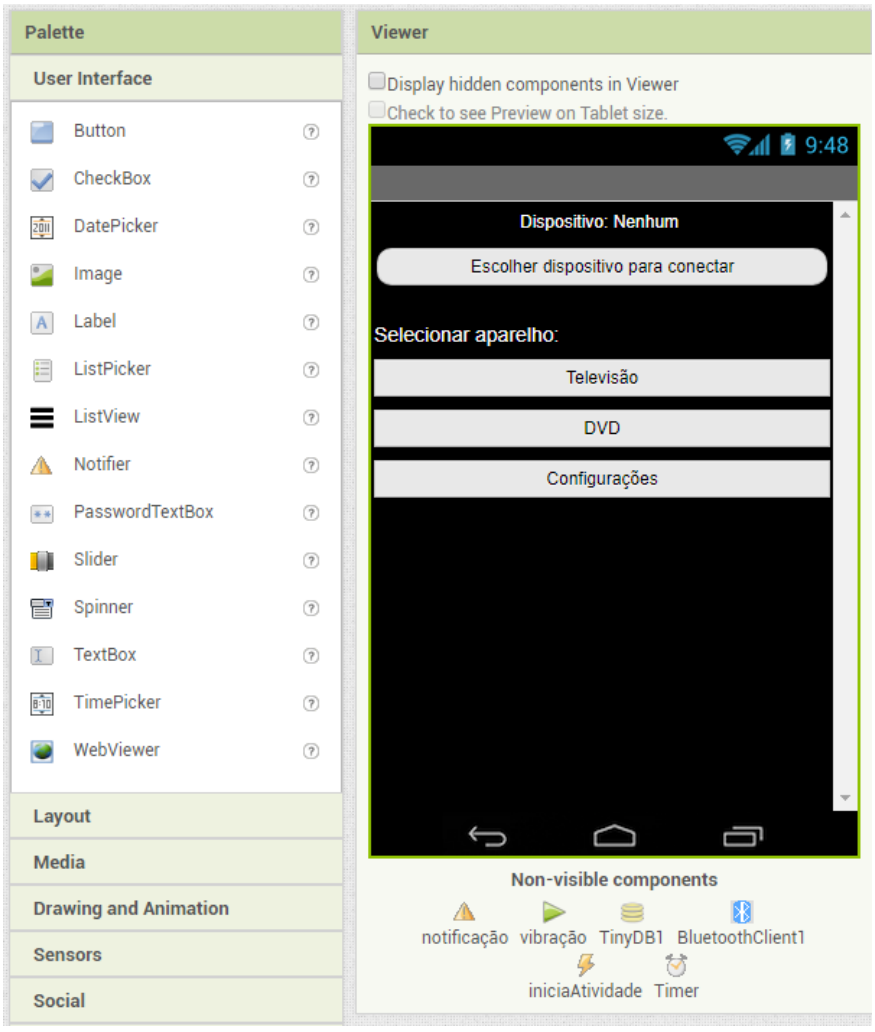
2.7.3 Designer

Na seção Palette (5a), estão listados e subdivididos em categorias todos os recursos (visuais ou internos) candidatos a fazer parte do aplicativo. Pode-se adicionar elementos visíveis (botões, caixas de texto, imagens, listas) e os não visíveis (sensores internos ao smartphone, relógio, banco de dados).

Os itens visíveis devem ser posicionados livremente dentro do Viewer (5a), que contém uma tela inicialmente vazia dando uma prévia do que será o aplicativo. Contudo, geralmente é preciso fazer leituras em alguns dos sensores ou usar certos aparatos nativos dos celulares, como o Bluetooth e o relógio. Os componentes não visíveis são dispostos abaixo da tela simulada indicando quais desses recursos serão usados quando ela for aberta no smartphone. Caso necessário, é possível e recomendado que se crie quantas telas se queira com o botão "Add Screen...".

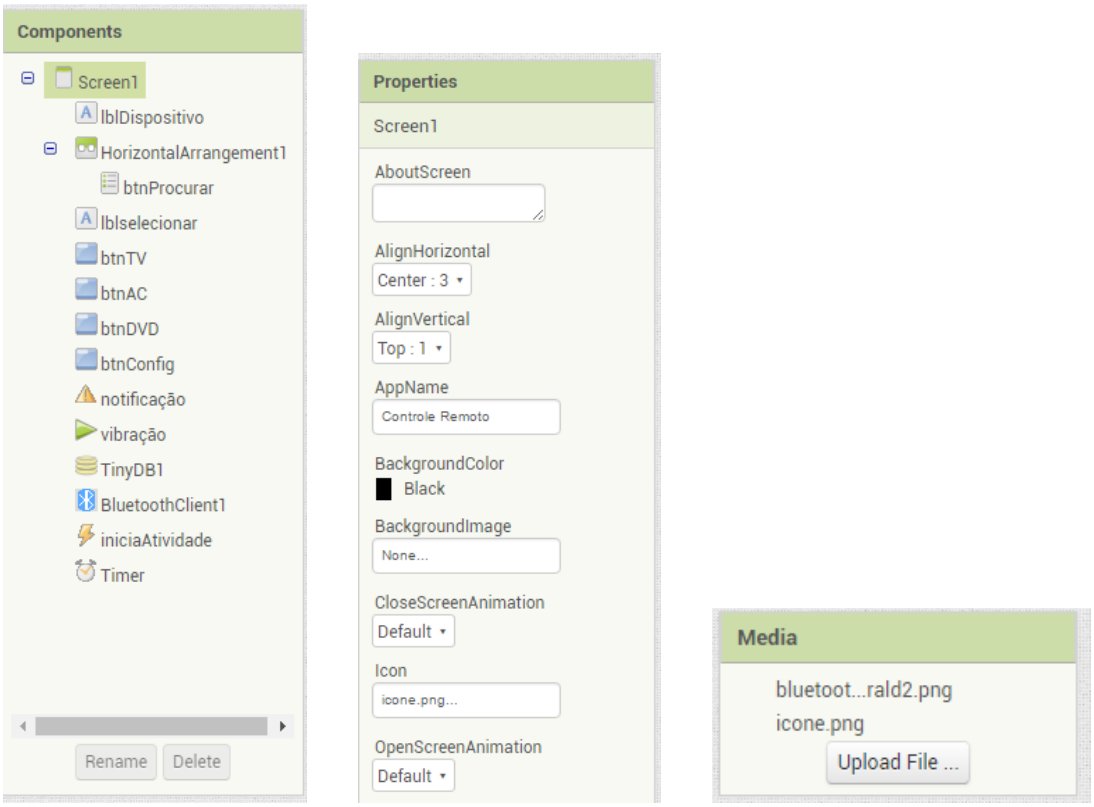
Arquivos de imagem e áudio usados pelo App devem ser carregados ao site através

do espaço Media(6c). E por fim, as propriedades individuais de cada objeto que compõem a aplicação são editadas em Properties(6b).



(a) Caixa de paleta e visor

Figura 5 – Elementos compondo a interface do MIT AppInventor
Fonte: i2.appinventor.mit.edu



(a) Caixa de componentes (b) Caixa de propriedades (c) Caixa de mídia

Figura 6 – Elementos compondo a interface do MIT AppInventor
Fonte: i2.appinventor.mit.edu

2.7.4 Blocks

Finalizada a aparência, é hora de programar cada componente no modo Blocks. Nele, é apresentado todo o repertório de componentes (visuais ou não) que foram escolhidos para compor o projeto e suas possíveis ações. Os blocos com formato de caixa aberta são análogos às funções (ou métodos) na maioria das linguagens de programação, uma vez que em seu interior serão encaixados comandos. Por exemplo, ao selecionar um botão qualquer, o site mostra os métodos que serão ativados quando o botão sofrer algumas das ações: clique curto, clique longo ou solto. Na hora da execução dessa ação, seu respectivo bloco, previamente programado, executa os comandos que o preenchem.

Como visto nas imagens 7 e 8 (página 36), além das ações, podem-se vistas as variáveis relacionadas às características do componente selecionado: cores, textos, tamanho, etc. Todas suscetíveis a alterações.

Todos os componentes abrem uma lista de blocos representando seus respectivos métodos e variáveis nos quais serão dispostos no espaço e branco da seção Viewer.

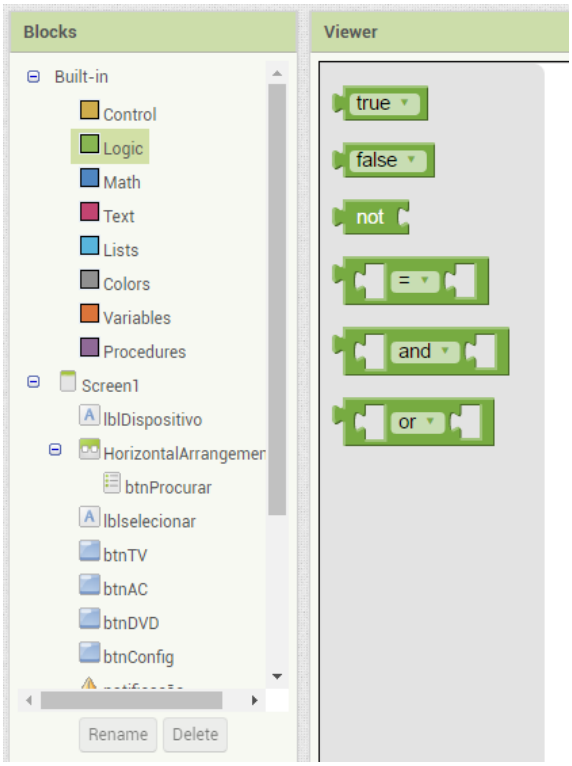


Figura 7 – Ao selecionar um item da lista, são mostradas ações relacionadas
Fonte: i2.appinventor.mit.edu

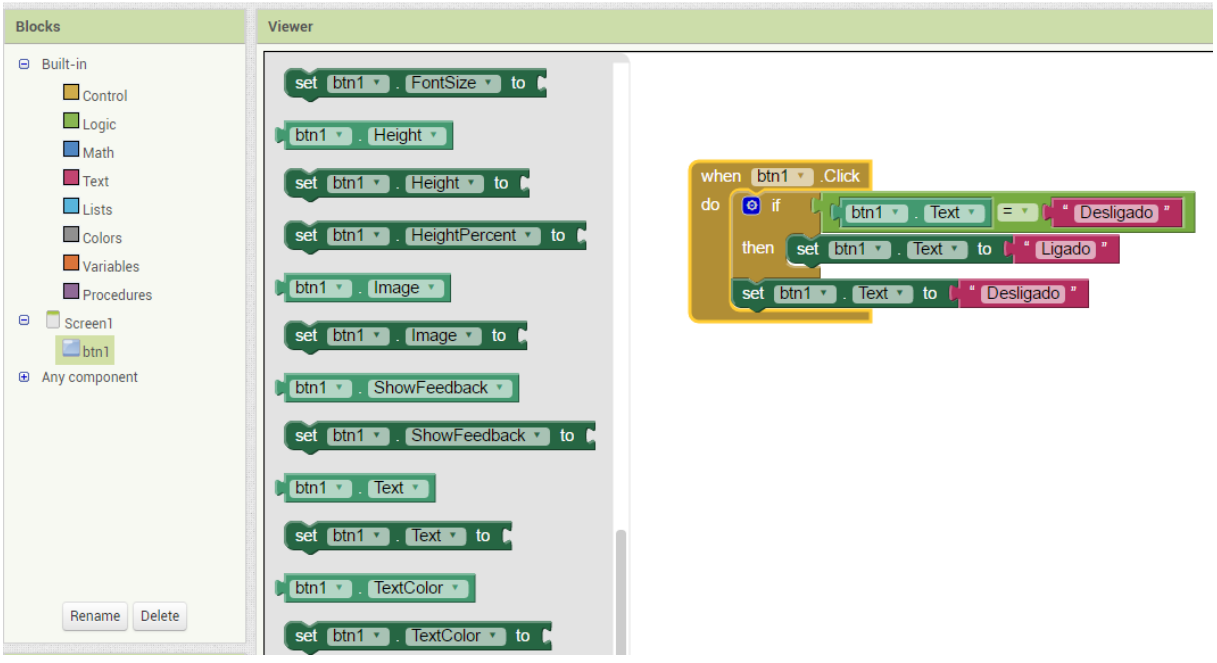


Figura 8 – Blocos relacionados a um botão
Fonte: i2.appinventor.mit.edu

2.8 Ethernet Shield

Segundo a página oficial do Arduino (ARDUINO-ARDUINOSHIELDS, 2005), shields são placas que podem ser plugadas na placa do Arduino de forma a incrementar suas

funcionalidades. No mercado existe uma grande variedade, cada uma destinada para uma função em específico. Este tópico será focado no único shield utilizado no trabalho, o Ethernet Shield, que utiliza o CI W5100 da WIZnet.

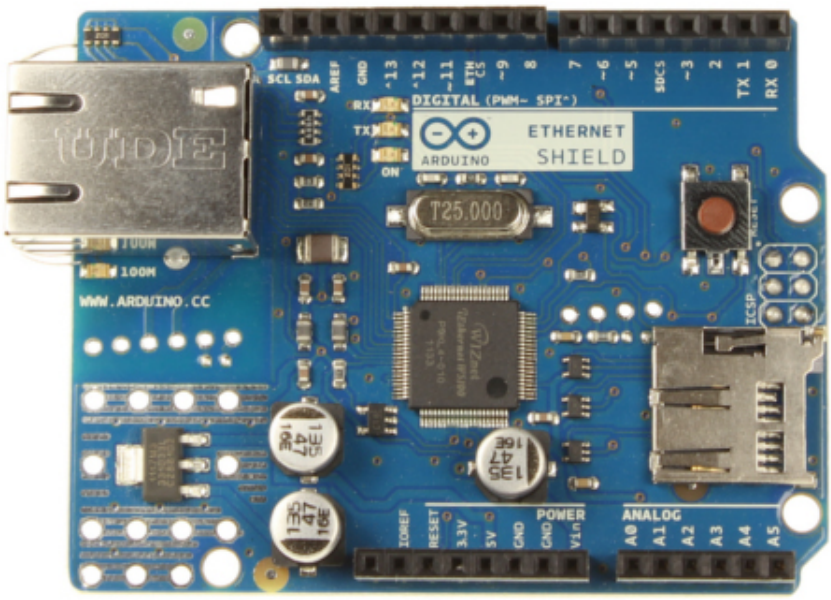


Figura 9 – Ethernet Shield
Fonte: https://www.arduino.cc/en/Main/ArduinoEthernetShieldV1

O Ethernet Shield é responsável por permitir que o Arduino se conecte à web, dessa forma é possível visualizar qualquer tipo de dado que esteja na memória do Arduino através de uma página da web e ainda utilizar desta para passar instruções ao microcontrolador. A comunicação do Ethernet Shield com a web acontece pelo intermédio do conjunto de protocolos TCP/IP que será explicitado adiante.

É preciso mencionar a arquitetura servidor-cliente ou server-client em inglês. Nessa arquitetura o processamento dos dados são divididos em dois processos: Enquanto um é responsável por guardar os dados (servidor), o outro é responsável por solicitar tais dados (cliente). Neste trabalho, o usuário do sistema proposto será o cliente, enquanto que o Arduino conectado ao Ethernet Shield fará o papel de servidor e por isso será referido como Arduino-servidor ao longo do trabalho (a palavra servidor sozinha será utilizada para fazer referência a servidores de forma geral). A figura 10 esclarece a terminologia servidor-cliente.

Segundo a página oficial do Arduino que fala sobre Ethernet Shield (ARDUINO-ETHERNET, 2005), a comunicação entre o Arduino e o shield é feita através dos pinos ISP (In-System Programmer) também chamado de ICSP (In-Circuit Serial Programming). A figura 13 mostra a localização desses pinos na placa de um Arduino Uno.

Os pinos ICSP são gerenciados via SPI (Serial Peripheral Interface), que é um modo amplamente utilizado para comunicação, sempre full-duplex, a curta distância entre

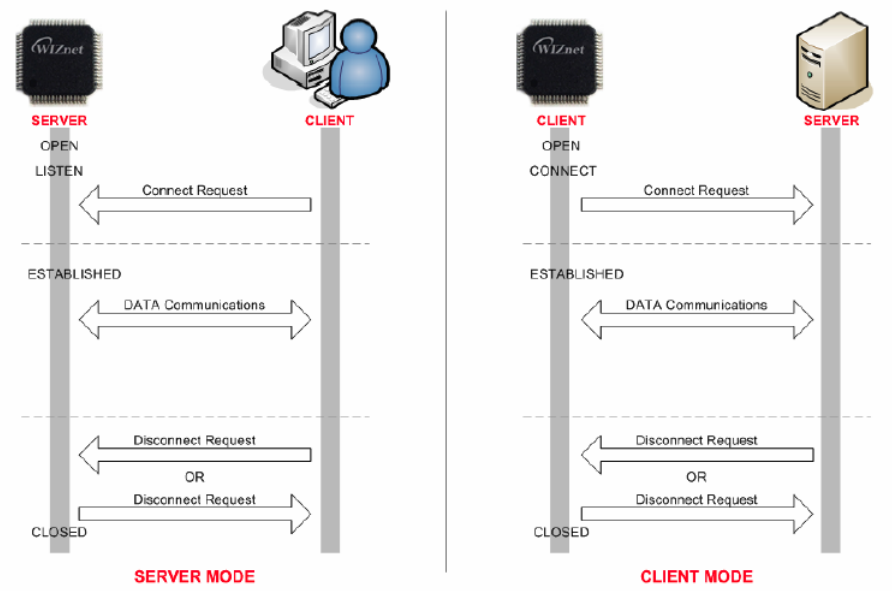


Figura 10 – Ethernet Shield pode operar como servidor ou como cliente
Fonte:
https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf

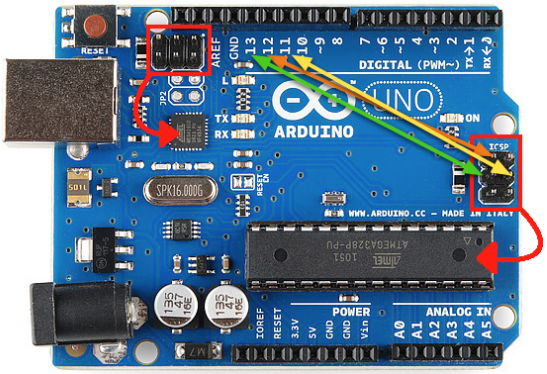


Figura 11 – Indicação dos pinos ICSP na placa Arduino
Fonte: http://br-arduino.org/2015/05/arduino-icsp-attiny-atmega.html

microcontrolador e periféricos (ARDUINO-SPI, 2005). Os pinos ICSP estão interligados com os pinos digitais da placa do Arduino, portanto, é necessário saber com um certo nível de detalhes como a troca de dados via SPI funciona, evitando assim o uso indevido da pinagem e consequentemente evitando possíveis bugs. Mais detalhes sobre SPI estão presentes mais abaixo neste tópico.

2.8.1 TCP/IP

Neste tópico será apresentado de forma resumida o que é TCP/IP, pois esse é um assunto muito extenso e não é o foco desse trabalho entrar em detalhes nesse tema, um conhecimento básico já é o suficiente.

De acordo com (FOROUZAN, 2007), TCP/IP é um conjunto de protocolos que

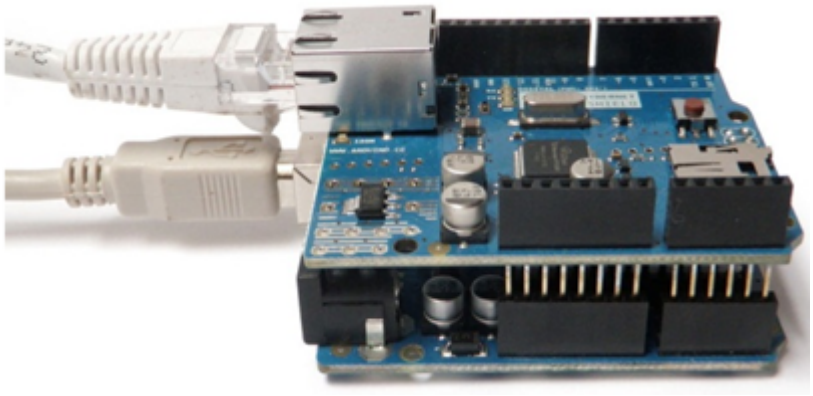


Figura 12 – Arduino e Ethernet Shield conectados
Fonte: http://eletronicaemcasa.blogspot.com.br/2016/04/Arduino-Ethernet-Shield-W5100-Tutorial-de-como-comunicar.html

regem como a informação será enviada de um ponto a outro dentro de uma rede. Os protocolos TCP/IP estão organizados em quatro camadas, sendo que cada camada possui seu próprio conjunto de protocolos e é responsável por uma parte no processo de transferência de informações em uma rede.

As camadas estão ligadas umas as outras seguindo uma hierarquia, a camada mais alta depende dos dados fornecidos pela camada imediatamente abaixo na hierarquia. A camada de mais alto nível trabalha com um nível maior de abstração, ou seja, com informações mais compreensíveis ao usuário, enquanto quanto menor for o nível da camada menor é o nível de abstração e mais próxima da linguagem do computador (FOROUZAN, 2007).

As camadas do modelo TCP/IP são divididas em: Host-rede, internet, transporte e aplicação. A primeira é a mais baixa; a quarta é a mais alta. A função de cada camada, segundo (FOROUZAN, 2007), está detalhada abaixo:

Host-rede: Define como os bits de dados irão trafegar por um meio físico; determina procedimentos a serem executados para que a transmissão seja realizada, tais como taxa de transmissão e sincronização de bits.

Internet: A camada de internet é responsável por atribuir endereços e assim define para onde os dados devem ser enviados e quem deve recebê-los. Aqui o TCP/IP suporta o Internetworking Protocol (IP) e este por sua vez suporta quatro outros protocolos responsáveis por atribuir endereços para cada componente da rede.

Transporte: Esta camada é responsável por fazer a entrega de toda a mensagem das camadas anteriores; garantir que a mensagem seja enviada na sequência correta, uma vez que a camada internet apenas entrega os dados, mas não identifica a relação entre os pacotes de dados.

Aplicação: Permite a conexão entre sistemas que necessitam se comunicar entre

Tabela 1 – Tabela de pinos para conexão SPI dos vários modelos de Arduino

Fonte: (ARDUINO-SPI, 2005)

Arduino/ Genuíno Board	MOSI	MISO	SCK	SS (Slave)	SS (Master)	Level
Uno or Duemilanove	11 or ICSP-4	12 or ICSP-1	13 or ICSP-3	10	-	5v
Mega1280 or Mega2560	51 or ICSP-4	50 or ICSP-1	52 or ICSP-3	53	-	5v
Leonardo	ICSP-4	ICSP-1	ICSP-3	-	-	5v
Due	ICSP-4	ICSP-1	ICSP-3	-	4, 10, 52	3.3v
Zero	ICSP-4	ICSP-1	ICSP-3	-	-	3.3v
101	11 or ICSP-4	12 or ICSP-1	13 or ICSP-3	10	10	3.3v
MKR1000	8	10	9	-	-	3.3v

si; traduz informações como strings, números e outros em um fluxo de bits que será processado nas camadas mais abaixo na hierarquia; e também habilita o usuário a usar a rede.

Portanto, quando o usuário do sistema (cliente) faz um request a informação estará primeiramente na camada de aplicação e será processada até chegar na camada host-rede e depois volta para a camada de aplicação respondendo ao cliente.

2.8.2 SPI

Este tópico tem como objetivo fazer uma breve detalhamento do que é SPI e como funciona, pois isso interfere diretamente na programação e consequentemente no funcionamento adequado do sistema.

Segundo o datasheet do CI que fica na placa Arduino (ATMEL, 2015), na arquitetura do Atmega328p, que é utilizado na placa do arduino Uno, alguns pinos tem mais de uma função e só podem desempenhar uma delas por vez, no caso do uso da SPI os pinos utilizados variam de acordo com o modelo da placa como pode ser visto na tabela 1, que foi retirada da página oficial da plataforma Arduino.

SPI além de ser um barramento de quatro fios utilizado por microcontroladores para comunicação com periféricos de forma rápida e a uma curta distância é também um conjunto de procolos (MICROCHIP, 2015) que regem a forma como a informação irá trafegar do microcontrolador até o periférico. Neste trabalho o periférico é o Ethernet Shield.

O barramento ICSP do Arduino consiste de seis pinos como mostrado na figura 13, onde dois deles são para alimentação, um para reiniciar a operação de transmissão de dados e os outros três pinos são para a SPI e são chamados de MISO, MOSI e SCK ou SCLK.

Figura 13 – Pinagem da ICSP na placa arduino
Fonte: (ARDUINO-SPI, 2005)

Um ponto importante e que merece atenção especial é que foi mencionado que a SPI possui quatro fios, mas a ICSP do Arduino tem apenas três pinos destinados para a SPI, faltando o último pino que é chamado de SS. Consoante (ARDUINO-SPI, 2005), todos os microcontroladores baseados na arquitetura AVR, que são fabricados pela Atmel, possuem pinos digitais específicos para atuarem como SS. Sendo assim é possível utilizar tais pinos para ativar e desativar periféricos conectados a placa Arduino.

O Ethernet Shield possui nativamente um espaço para inserir um cartão SD, o que significa que a SPI do Arduino controla dois dispositivos, o SD e o controlador Ethernet. Isso implica no uso de dois pinos digitais para ativar e desativar a comunicação com cada dispositivo, sendo esses os pinos 4 e 10, respectivamente, no caso do Arduino Uno (ARDUINO-SPI, 2005). Por esse motivo não se deve usar esses pinos para outra finalidade.

SCK ou SCLK é o sinal do clock que é gerado pelo mestre e replicado no escravo para garantir a sincronia no envio de dados (MICROCHIP, 2015), isso pode ser visto na figura 14. A SPI possuem alguns registradores que determinam alguns funções importante para seu funcionamento (TEXASINSTRUMENS, 2012), mas neste projeto não foi necessário ir tão fundo, a configuração padrão já é adequada.

Os fios MOSI e MISO são responsáveis pelo tráfego dos dados propriamente ditos, do modo que o mestre envia pelo primeiro citado e recebe pelo segundo. A figura 14 é uma mostra a direção e a sincronia do envio de dados, mas não é o diagrama de blocos completo de uma SPI.

2.9 Conexão com a internet

O Arduino-servidor criado pode funcionar de duas maneiras dentro de uma rede, elas são chamadas de intranet e internet.

A intranet tem o acesso mais restrito por se tratar de uma rede local – rede doméstica por exemplo, ou seja, aparelhos fora dessa rede não tem acesso as informações que estão passando pelo servidor. Mesmo a intranet tendo essa característica restrita é importante ressaltar sua importância em redes corporativas, uma vez que esse tipo de

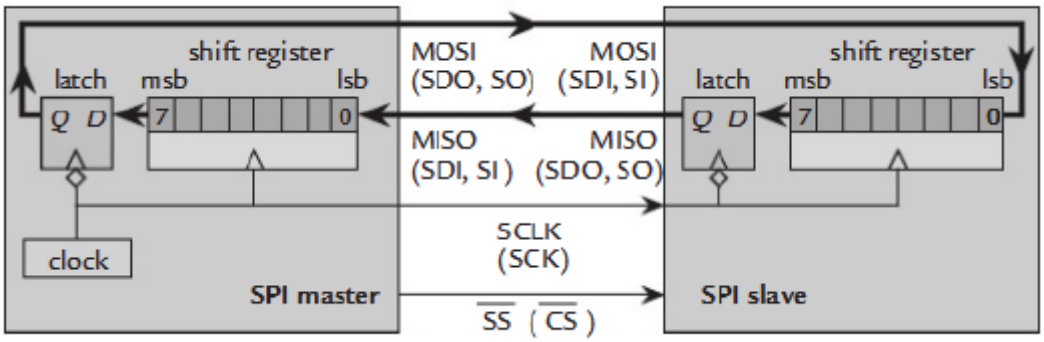


Figura 14 – Diagrama de blocos simplificado de uma SPI
Fonte:
https://courses.cs.washington.edu/courses/cse466/11au/calendar/07-comms-posted2.pdf

restrição é desejada.

A internet é o sistema mundial de rede de computadores que interliga inúmeras redes locais a um servidor com a informação. Utilizando internet é possível acessar o Arduino-servidor de qualquer lugar de mundo.

Neste trabalho ambos os tipos de conexão foram utilizados, sendo que a intranet é substituível pelo aplicativo para Android que se conecta diretamente com os escravos da rede. As figura 15 e 16 mostram as formas de conexão intranet e internet, respectivamente.

Figura 15 – Diagrama de blocos da rede local (intranet)
Fonte: Propriedade dos autores

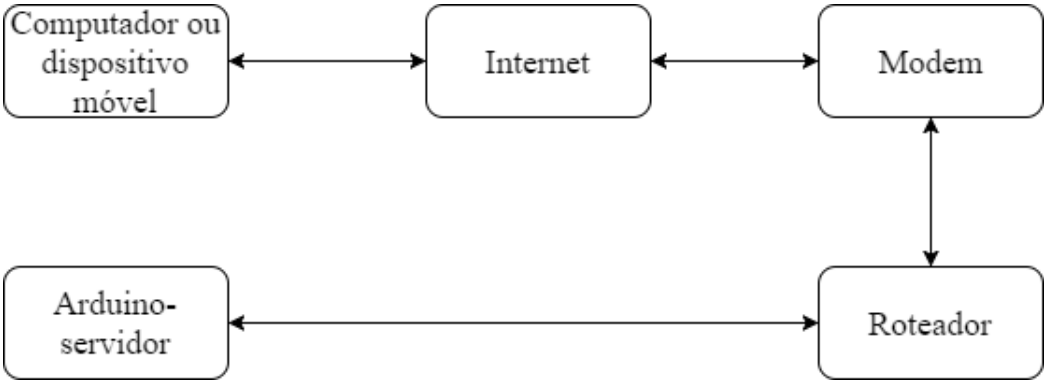


Figura 16 – Diagrama de blocos do acesso pela internet
Fonte: Propriedade dos autores

3 Metodologia

3.1 O Aplicativo

No aplicativo criado, são simulados dois tipos de controles remotos, um de TV e um de DVD. Os controles virtuais podem ser preenchidos pelo usuário com os códigos numéricos associados a cada tecla dos controles reais. Dentro do aplicativo, o usuário pode configurar até cinco controles para o mesmo tipo de aparelho. Os códigos são gravados permanentemente na memória EEPROM do Arduino e não há perdas mesmo que haja desinstalação do aplicativo. Porém, antes de usar um controle virtual, é preciso salvar os códigos numéricos nos quais o aparelho entende por comandos.

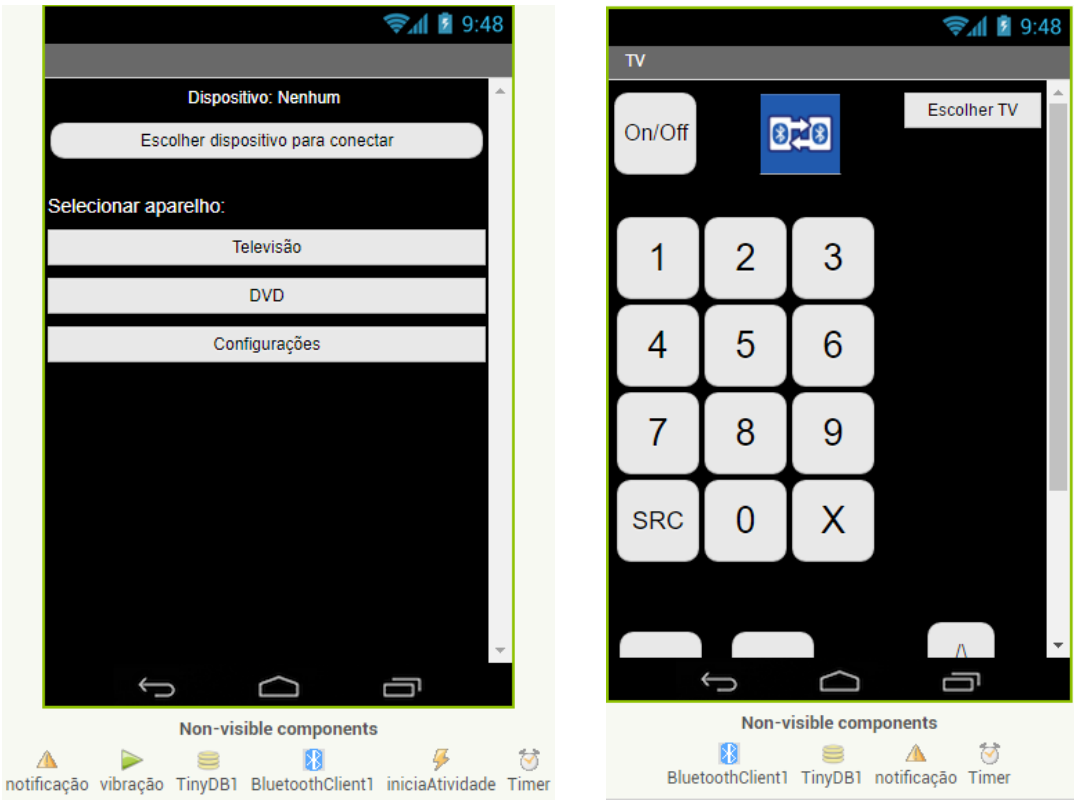
Screen1 é criada automaticamente pelo site a fim de ser a tela principal. Por padrão, essa tela é a mostrada assim que o aplicativo é executado. Não se pode definir outra tela como principal e, ao contrário das outras, é impossível excluí-la. Nela, o usuário é apresentado ao botão que apresenta uma lista dos endereços de todos os dispositivos Bluetooth nas proximidades detectados pelo celular. Na primeira execução logo depois de ser instalado, quando não há dispositivos escolhidos, o cabeçalho mostra o texto "Dispositivo: Nenhum". Mas, após a escolha do módulo a ser usado, o aplicativo recorda seu endereço para uso posterior. Assim, caso o utilizador queira conectar-se ao mesmo aparelho da última sessão, não é necessário selecionar novamente. A próxima etapa é acessar o controle do aparelho que se deseja controlar ou acessar a tela de configuração. As opções são: Televisão, DVD e Configurações. Cada um desses botões leva para uma outra tela que, quando exibidas, faz o smartphone proceder com a conexão Bluetooth. De modo que o celular só permanece conectado quando se está em uma das telas de controle.

TV é a tela desenvolvida para simular as funções básicas de um controle remoto de televisão. Nela estão contidos: botão aumentar ou diminuir volume, mudar de canal, o teclado numérico e teclado de navegação com setas direcionais. Além desses, há outros três botões posicionados no topo que são comuns a todos os controles. O primeiro é o de ligar o aparelho. O azul no centro, quando ativado, faz o celular reestabelecer a conexão com o dispositivo, caso tenha sido perdida. Já o terceiro, serve para selecionar um dentre os cinco controles disponíveis para o mesmo tipo de aparelho.

DVD é onde está o controle para aparelhos reprodutores de mídia digital. Possui duas matrizes de botões com funções comuns à maioria dos controles desse porte e o teclado de navegação.

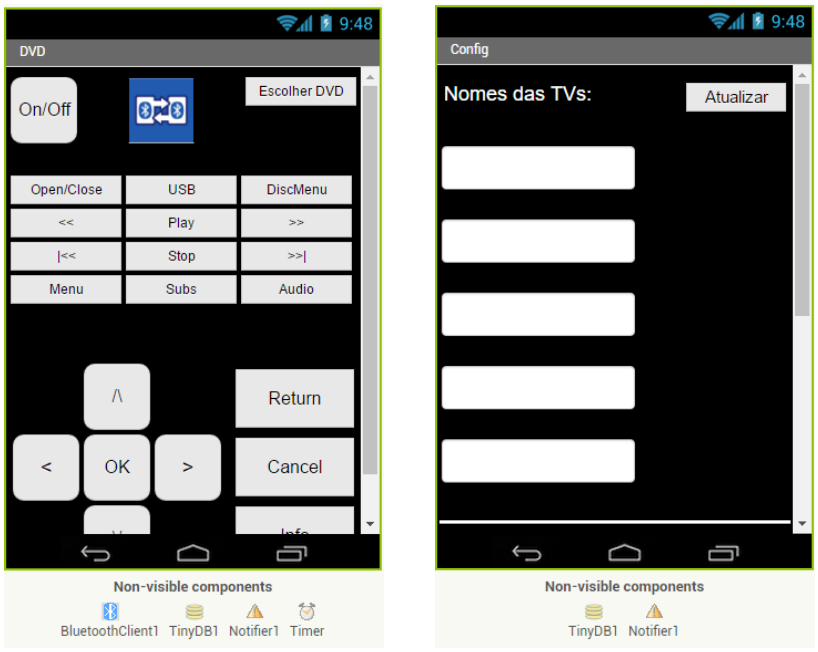
Config é a responsável de pôr os nomes de identificação nos aparelhos. Dados o limitado espaço de memória para guardar códigos, foi decidido um máximo de cinco

controles virtuais com as mesmas funções, que nativamente são nomeados com números de 1 a 5. Ou seja, usuário pode configurar até cinco controles para cada um dos três eletrodomésticos suportados. Desse modo, visando facilitar a experiência, pode-se nomear os controles de acordo com sua função.



(a) Tela inicial, “Screen1” (b) Controle virtual de televisão

Figura 17 – Telas do Aplicativo
Fonte: Propriedade dos autores



(a) Controle virtual de DVD (b) Tela de configuração

Figura 18 – Telas do Aplicativo
Fonte: Propriedade dos autores

3.2 Protótipo offline

Após a aquisição dos componentes, o primeiro protótipo feito numa protoboard foi o circuito que receberá instruções do celular. O mesmo será embutido no aparelho, receberá os comandos via Bluetooth e mandará sinais de luz IR para o objeto controlado. O equipamento disposto na protoboard, conforme a figura 19, consiste de:

1. Arduino UNO
2. módulo Bluetooth
3. módulo para cartão de memória
4. adaptador para cartão microSD
5. LED emissor e receptor de sinais IR.

Na idealização do projeto, o pretendido era usar um módulo de cartão microSD para guardar os sinais do controle. A escolha se deu após o conhecimento de que os dados armazenados seriam números inteiros de 32 bits, algo que (em tese) só pode ser armazenado na forma de texto no Atmega328. Depois de vários testes, tornou-se inviável usar arquivos de textos devido ao grande consumo de memória causado pela biblioteca “SD.h”. Em outras palavras, a plataforma não suportou o programa, que excedia sua

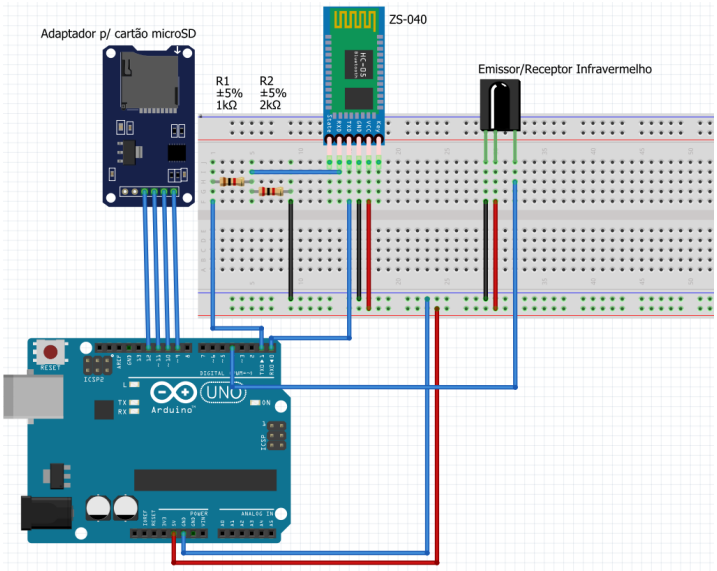


Figura 19 – Primeiro protótipo feito
Fonte: Propriedade dos autores

capacidade de armazenamento. Assim, optamos por usar a EEPROM interna do Arduino, mesmo sendo um caminho mais difícil.

Como visto na imagem, há um divisor de tensão entre o HC-05 e a placa. A explicação para tal é a incompatibilidade entre as tensões do terminal RX do módulo e o pino TX no Arduino. O primeiro trabalha recebendo sinais de 3.3V; o segundo, enviando sinais com 5V. Logo, o circuito divisor protege o módulo contra danos. Não foi necessário fazer o mesmo com o TX do módulo, uma vez que ele produz sinais de apenas 3.3 Volts, algo suportável pelo RX do Arduino.

Em sua segunda versão, o protótipo apresentava a mesma configuração da imagem 19 com exceção do módulo para microSD. Desenvolvido o programa correto, seu funcionamento estava em acordo com o desejado. O circuito recebia comandos via conexão Bluetooth e comandava a TV usada nos testes.

3.3 Protótipo online

Este bloco permite que o usuário controle a televisão ou aparelho de DVD utilizando algum dispositivo que tenha acesso a internet, ou seja, smartphones; tablets ou computadores. O acesso pode ser feito tanto através de uma LAN quanto de uma WAN, isso significa dizer que o usuário pode acessar o sistema com um aparelho que esteja conectado a algum computador na mesma rede em que o servidor está ou a partir de uma rede externa, respectivamente.

Para isso ser possível é necessário configurar o roteador ao qual o Ethernet Shield está conectado. A configuração a ser feita é o redirecionamento de IP. Basicamente, o

usuário faz um request (solicitação) ao Arduino-servidor através do endereço de IP fornecido pelo provedor de internet, o roteador redireciona essa solicitação para um IP local, definido pelo usuário nas configurações do roteador, e esse IP é o utilizado pelo Ethernet Shield e na programação.

O roteador utilizado no projeto é do modelo Intelbras WRN240. A figura 20 mostra a configuração realizada no roteador.

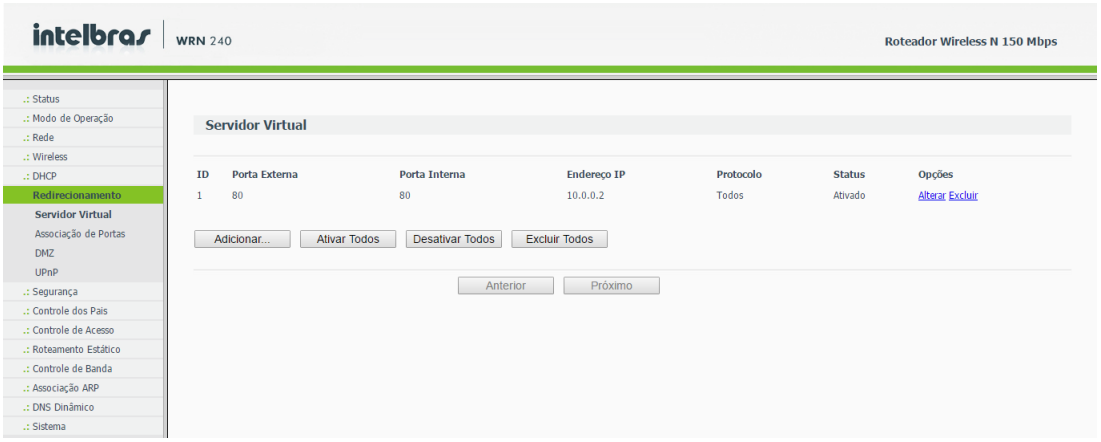


Figura 20 – Configuração de redirecionamento de IP no roteador
Fonte: Propriedade dos autores

A forma de acesso ao sistema é por meio de uma página web, desenvolvida em HTML em conjunto de scripts simples em JavaScript e CSS para deixar a página visualmente intuitiva. O software utilizado para o desenvolvimento foi o Notepad++ versão 6.9.2 e atualizações posteriores até a versão 7.2. Apesar de não oferecer ferramentas poderosas de para facilitar e agilizar o desenvolvimento ele foi escolhido, pois é gratuito e o código fonte da página criada é simples.

Em uma tentativa de promover segurança ao acesso do sistema foi criada uma tela para o usuário efetuar login, ela funciona e faz a checagem se os campos “E-mail” e “Senha” foram preenchidos corretamente - a verificação é feita com uma String pré-definida na programação do arduino, isso será abordado mais a frente - e caso essas informações estejam corretas a tela inicial é exibida. Contudo esse mecanismo possui defeitos que serão discutidos posteriormente. A figura 21 mostra a tela de login criada e a figura 22 ilustra a tela inicial da página.

Na tela inicial vale destacar o significado de alguns elementos. “Lâmpada 1” e “Lâmpada 2” mostram o estado atual de duas portas digitais do arduino-servidor que foram configuradas como entrada, então assim que é alterado o valor lógico da porta o valor é atualizado automaticamente na tela, sem recarregar a página por completo. Para isso foi utilizado uma ferramenta conhecida como AJAX.

“Temperatura” mostra o valor de uma das portas analógicas do arduino-servidor e serve para ilustrar que é possível colocar qualquer tipo de informação de forma dinâmica

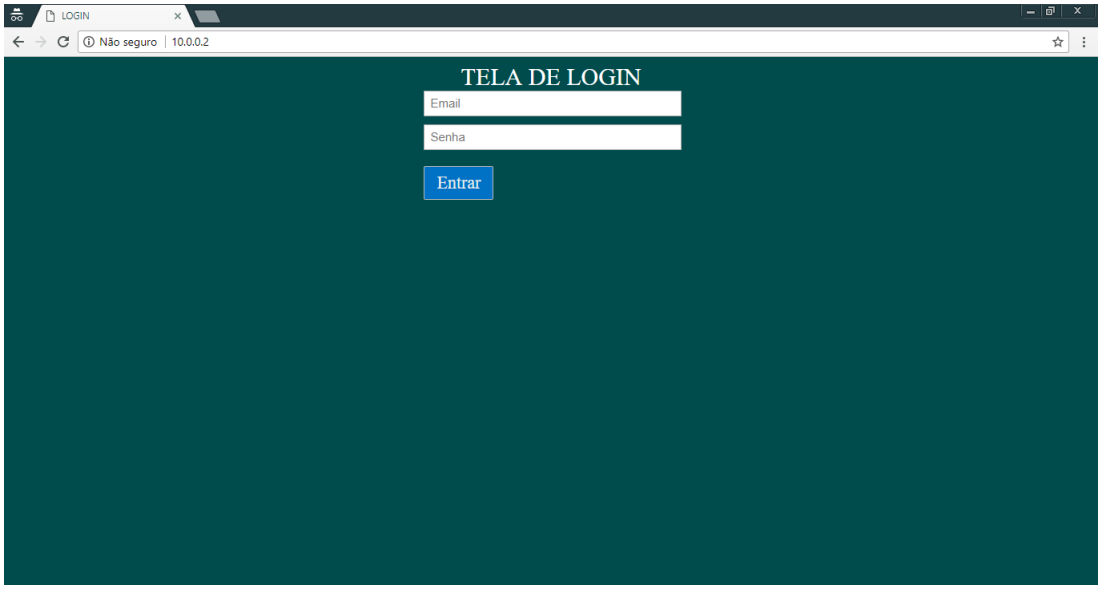


Figura 21 – Tela para login no sistema
Fonte: Propriedade dos autores

Figura 22 – Página inicial
Fonte: Propriedade dos autores

na tela, dependendo apenas da necessidade. “Sinal Analógico” faz o mesmo, mas adiciona o elemento fixo na tela “/1023” para indicar que o valor antes da barra pode ir até 1023.

Os três botões chamados de “TV”, “Relé” e “DVD” são utilizados para acessar os respectivos “controles virtuais”. Assim como no aplicativo Android também é possível ter mais de um aparelho do mesmo tipo, aqui foram feitos apenas três como a figura 23 indica, mas a adição de mais telas é simples, bastando adicionar mais um elemento da “drop-down list”.

O Ethernet Shield possui um espaço para inserir um cartão micro SD e a biblioteca “SD.h” do Arduino é capaz de acessar e ler um arquivo por vez, desde que o mesmo esteja

Figura 23 – Página inicial
Fonte: Propriedade dos autores

armazenado nesse cartão, dessa forma não é necessário inserir o código fonte do HTML direto no Arduino, mas sim embarcar esse código, assim utiliza-se menos memória de programa do microcontrolador e aumenta consideravelmente a quantidade e o tamanho das páginas web que o Arduino pode ler e processar.

Uma ressalva importante a ser feita é que é comum em projetos de criação de páginas web envolvendo html, javascript e css que cada tipo de arquivo seja feito de forma separada e "linkados" através de instruções dentro dos próprios arquivos, contudo o Arduino só consegue ler um arquivo por vez, então cada página web deste projeto foi criada em um único arquivo ao invés de três. As figuras 24, 25 e 26 mostram os controles feitos para cada tipo de aparelho.

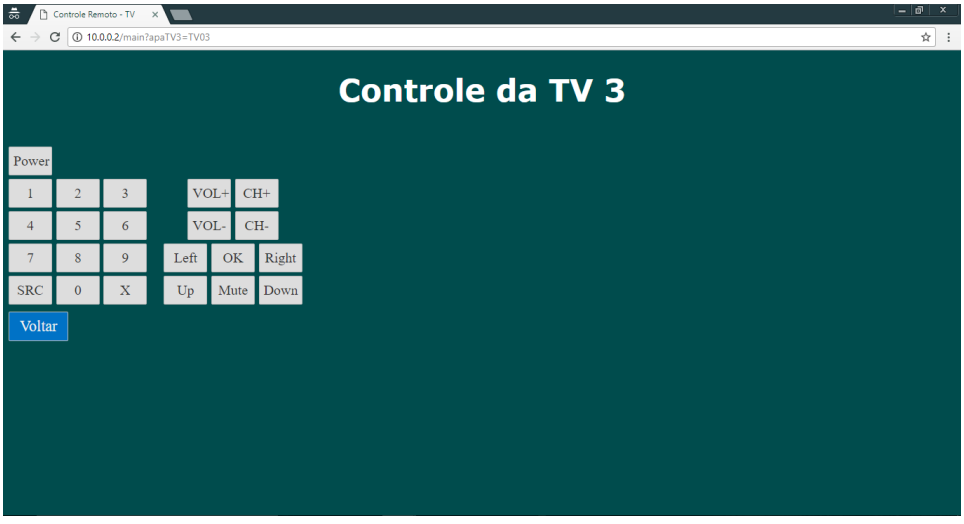


Figura 24 – Controle virtual para uma televisão qualquer
Fonte: Propriedade dos autores

Figura 25 – Controle virtual para relés acionando cargas na casa
Fonte: Propriedade dos autores

Figura 26 – Controle virtual para um aparelho de DVD qualquer
Fonte: Propriedade dos autores

O método utilizado para reconhecer que um botão foi pressionado foi a mudança de URL. Toda vez que o usuário clica em um botão o URL da página muda de acordo com o botão precionado e o novo URL é impresso na serial do Arduino, onde ele pode ser lido e interpretado de forma adequada. A figura 27 mostra o URL ao selecionar o televisor três.

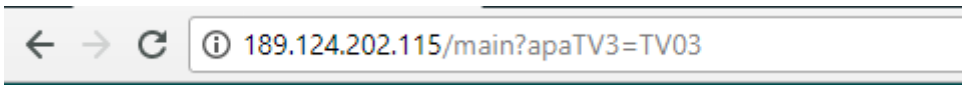


Figura 27 – URL da página assim que o TV3 é escolhida
Fonte: Propriedade dos autores

O número “189.124.202.115” corresponde ao endereço de IP fornecido pelo provedor de internet; “apaTV3” indica qual aparelho foi escolhido e também é a String que o

Arduino busca em sua serial para saber qual arquivo ".html" deve ler do cartão micro SD para abrir a página correta.

Utilizando o mesmo método acima para modificar o URL da página, ao clicar no botão "power" o trecho "button=power" é adicionado no final do URL e posteriormente ao clicar no botão "VOL+" o trecho "button=power" é substituído por "button=VOL%2B", sempre mantendo o trecho "apaTV3", dessa forma o microcontrolador consegue reconhecer qual é o aparelho que está sendo controlado e qual sinal deve ser enviado.

A partir das informações do aparelho e do botão, O código genérico de três caracteres, que será discutido na seção a seguir, é gerado e transmitido via Bluetooth para o Arduino escravo que recebe este código, interpreta-o e retorna uma mensagem em hexadecimal que é enviada para o aparelho correspondente via emissor de infravermelho.

Concluídos os testes, a plataforma UNO não é mais necessária e pode ser descartada, somente seu microcontrolador terá utilidade. Os Atmega328, em conjunto com o resto do circuito, podem ser encaixados cada um numa protoboard ou soldados numa PCB, assim como o Ethernet Shield. Fazendo isto, o tamanho é reduzido a dois chips e um LED, além de fazer com que a placa retirada possa ser reaproveitada em outro projeto.

3.4 Protocolos

3.4.1 Entre Arduino e Aplicativo

As mensagens mandadas via Bluetooth pelo aplicativo consistem em palavras de três caracteres. A palavra é enviada sempre que um botão sofre uma das duas ações programadas: clique rápido ou pressionamento longo. A ação está contida no primeiro caractere, um clique é representado por "c" e o pressionamento por "p". Ao receber uma palavra com inicial "c", o Arduino sabe que deve enviar o código da tecla clicada. Caso a inicial fosse "p", ele começaria o processo de armazenar o código que receberá do controle remoto. Ou seja, a primeira letra indica se o sinal infravermelho deve ser enviado ou recebido.

O segundo caractere representa qual botão sofreu a ação. Cada botão de cada controle tem sua letra do alfabeto associada arbitrariamente. Por este motivo, o máximo de botões no mesmo controle são vinte e seis. O da TV, por exemplo, contém vinte e dois botões, sendo necessário usar as letras de "a" até "v". O código de cada botão pode ser encontrado no apêndice D.

O caractere seguinte representa qual o tipo de aparelho está sendo controlado. No projeto é possível controlar três diferentes eletrodomésticos, numerados de 0 a 2. Logo, o terceiro caractere tem três valores possíveis: "0", "1" que correspondem respectivamente à televisão e ao reprodutor de DVD/CD.

Para fixação, tomemos como exemplo as palavras “ca0” e “pa2”. O primeiro é o comando para enviar o código do botão representado pela letra “a” no controle de TV; o segundo, comanda o Arduino para que fique em modo de espera para receber o código infravermelho. Este, será guardado na memória e será enviado quando o botão de letra “a” no controle do DVD for pressionado.

3.4.2 Samsung Protocol

Fabricantes de eletrodomésticos com controle remoto estabelecem regras de comunicação para que ambos funcionem sem causar nem sofrer interferência, por mais que estejam recebendo sinais oriundos de outras fontes. Os testes foram realizados em uma televisão da Samsung, que usa um protocolo próprio e privado, por isso é quase impossível achar documentos oficiais da empresa detalhando o processo de comunicação. Na internet foi achado um arquivo datado de 2008 no qual a empresa dá detalhes de como funcionava o micro controlador S3F80KBX, usado me controles remotos na época. A mensagem transmitida tem o seguinte corpo:

1. Um sinal de 9ms indicando o início da mensagem
2. Um intervalo de 4.5ms em nível baixo
3. O endereço de 8 bits do aparelho receptor
4. O mesmo endereço acima com os bits logicamente invertidos
5. O código do comando
6. O mesmo código acima com os bits barrados (logicamente invertidos)
7. Um pulso final de 562.5 microssegundos indicando o fim da transmissão

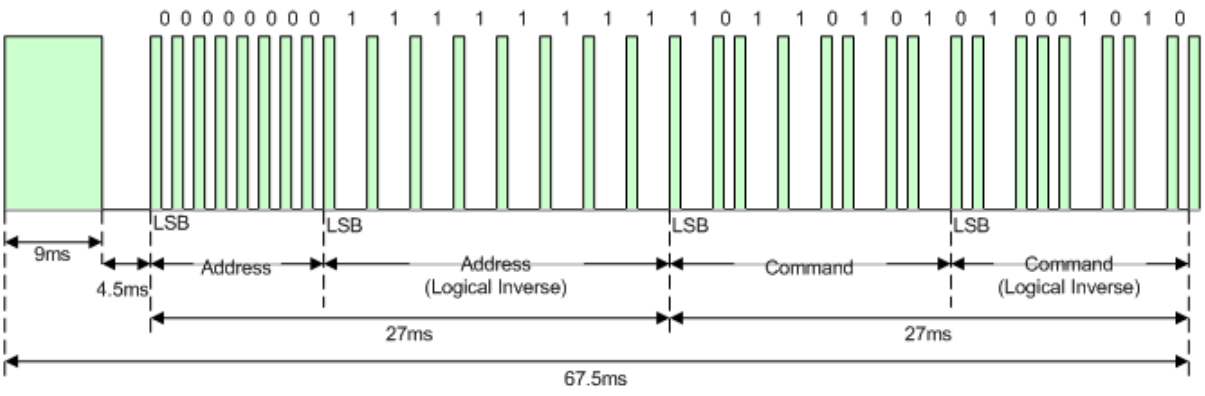


Figura 28 – Representação de um comando enviado usando o protocolo NEC
Fonte: http://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol#

É esse tipo de informação que teríamos de buscar se quiséssemos adaptar nosso sistema para aparelhos de outras empresas. Decodificando um controle remoto qualquer e identificando qual padrão o mesmo usa para se comunicar, poderíamos criar uma função na IDE do Arduino que o fizesse enviar sinais através do LED IR imitando o protocolo do controle decodificado. Felizmente, o trabalho de criar tal código já está feito e disponibilizado gratuitamente pelo usuário do GitHub, Rafi Khan. Em seu repositório "Arduino-IRremote" (SHIRRIFF, 2009b), encontram-se bibliotecas e exemplos de programas relacionados ao uso de controle remoto. A primeira versão da biblioteca foi criada por Ken Shirriff (SHIRRIFF, 2009a), mas já recebeu várias contribuições de usuários do Github.com.

3.5 Configurando o controle virtual

Na prática, o aplicativo permite que o usuário pressione um de seus botões e salve qualquer sinal IR detectado pelo LED. Porém, a aplicação foi criada pensando em transferir os códigos associados aos botões de um controle comum para seus respectivos representantes virtuais.

No momento em que o Arduino recebe um comando com inicial "p", é iniciado a rotina de leitura de sinais através do LED IR. Nesse momento, o usuário deve mandar o código de luz apertando o botão no controle. Para evitar erros causados por interferência, a gravação é feita somente após recebê-lo dez vezes seguidas. Feito isso, o sinal é salvo por meio da biblioteca "IRremote.h", feito para facilitar a decodificação de controles remotos. Em poucos passos, é obtido o comando escrito em oito algarismos na base hexadecimal e guardado numa variável do tipo String. Um número em hexadecimal de oito dígitos equivale a um de 32 dígitos (bits) em binário. Entretanto, o Atmega328 apresenta uma limitação: só armazena valores inteiros de 8 bits (0 a 255) em sua EEPROM (CIA, 2015). Logo, foi necessário realizar algumas operações antes de armazenar os dados.

Assim que o número é obtido, são executados os seguintes passos:

- A String é partida em outras quatro, cada uma com dois dígitos do número hexadecimal.

- Os novos arrays são passados como parâmetro para uma função, interpretando os caracteres como um número em hexadecimal e retornando numa variável inteira na base 10. Tal função foi implementada na IDE do Arduino.

- Este inteiro já pode ser guardado numa variável e armazenada numa posição da memória EEPROM.

Desse modo, cada comando de 32 bits utiliza quatro posições de memória. Para enviar o comando, é feito o processo inverso:

- O Atmega faz a leitura dos espaços de memória específicos do botão pressionado
- Aplica os valores lidos a uma função que retorna uma String com os dois dígitos na base 16
- Junta tudo em uma única variável com oito caracteres
- Envia-o usando as devidas funções da biblioteca “IRremote.h”.

4 Resultados e Discussões

4.1 Dificuldades

A primeira dificuldade enfrentada foi a configuração do chip Bluetooth. Os autores optaram por deixar detalhadamente registrado neste documento o processo de configuração do HC-05 justamente pela dificuldade de se encontrar um método que funcionasse. Há várias fontes na internet apresentando formas de fazer o módulo ficar em "AT MODE", nenhuma delas tão simples quanto a descrita nesse TCC.

No andamento do projeto, surgiu um problema relacionado ao servidor conectado à internet. O módulo Bluetooth adquirido não traz a opção de se desconectar de uma rede via software. Para tal, é necessário que o mesmo seja desligado da energia ou saia do alcance dos outros dispositivos. No celular do usuário, há a opção de desfazer o pareamento, já entre dois módulos HC-05, não há alternativa prática. Por isso, não foi possível fazer o servidor se conectar somente ao aparelho sendo controlado. Uma solução para esse problema (usando os mesmos equipamentos) seria reduzir a quantidade de aparelhos no sistema para, no máximo, sete e fazer com que o Arduino servidor esteja sempre em rede com todos eles. Apesar de simples, a falha foi notada tardiamente, provavelmente por causa dos autores sempre disporem de um ou dois aparelhos para realizar os testes. Para essa quantidade de objetos, foi possível realizar o previsto. Em resumo, durante o desenvolvimento, foram encontradas limitações não previstas na idealização do projeto, fazendo com que algumas lacunas fossem deixadas. Ainda sim, o objetivo ao qual o projeto se propôs foi atingido.

Outro problema encontrado está relacionado com a diversidade de protocolos presentes no mercado. O protótipo criado por exemplo, foi testado em uma TV da Samsung e, por isso, foi usada a função do "IRremote.h" que simula exatamente o padrão que os controles dessa empresa utiliza para se comunicar com seus aparelhos. Para fazê-lo funcionar em TVs de outra fabricante, seria necessário alterar o código carregado no microcontrolador para a função correta. Dado a quantidade de aparelhos disponíveis no mercado, criar um sistema que abranja todas as possibilidades possíveis demanda um esforço inviável em um projeto amador/caseiro. Uma saída para esse problema seria programar individualmente os microcontroladores escravos, antes de fixados em seus aparelho, de forma que seja chamado o método de envio apropriado para cada objeto.

Umas das maiores dificuldades enfrentadas se deu devido ao formato de memória do microcontrolador. O mesmo conta com 1024 espaços de 1 Byte, porém, o comando transmitido pelos controles remotos usados constam de 32 bits (4 Bytes). O método

criado para contornar esse problema está discorrido na seção 3.5 (página 53).

Com relação ao aplicativo, graças ao AppInventor, não houveram grandes dificuldades. Neste momento, já existe na internet vários tutoriais de como comandar o Arduino via celular usando a plataforma do MIT. Terminada essa parte, construir os controles remotos não é tarefa difícil quando pode-se arrastar livremente os itens da tela para a posição desejada.

4.2 Resultados

A integração de Arduino com Ethernet Shield funcionou do modo esperado, tanto na Internet quanto na Intranet. Isso foi comprovado fazendo com que ao pressionar um botão na página web uma porta digital do Arduino, conectada a um LED, fosse acionada e desacionada. Em seguida, isso foi substituído pela geração do código de quatro caracteres como está descrito na seção 3.4 e no apêndice D.

Iniciando no quesito segurança, o sistema deixa a desejar, como foi mostrado na seção 3.3, a senha digitada pelo usuário é mostrada na URL da página. Além disso, uma vez feito o login, caso algum smartphone ou computador acesse o IP do Arduino servidor funcionando no modo Internet, o que acontecerá será que o usuário que recém mandou uma solicitação ao servidor será redirecionado para a página atual e não para a de login, o que é uma falha grave de segurança. A solução para esse problema não foi abordada neste trabalho, mas serve como motivação para um trabalho futuro, aprimorando o atual.

A ideia inicial do projeto era fazer também um controle para ar condicionado, contudo a biblioteca utilizada suportava no máximo sinais de 32 bits, enquanto que os controles de ar condicionado emitem sinais de 48 bits, então para realizar esta tarefa seria necessário criar uma nova biblioteca e para isso seria preciso pesquisar qual é o protocolo utilizado pelos fabricantes para a partir daí ser possível decodificar o sinal em binário obtido pelo LED receptor de infravermelho. Devido a todas essas complicações que não faziam parte da proposta inicial do projeto a ideia de fazer um controle virtual para o ar condicionado foi abandonada.

No quesito segurança, feitas as ressalvas acima, o resultado obtido foi satisfatório uma vez que cumpre com sua proposta, ser uma interface entre o usuário e os eletrodomésticos da casa além de possibilitar o incremento de funcionalidades de uma forma simples, fazendo com que o presente projeto tenha uma utilidade abrangente para um uso pessoal. Entretanto, para o uso deste sistema em um ambiente mais sério, como uma indústria, melhorias na segurança são indispensáveis.

5 Conclusão

Diante de tudo o que foi exposto e discutido neste trabalho, conclui-se que a combinação de componentes escolhidos para realizarem a comunicação sem fio - tanto de forma interna quanto externa à residência - funciona, mas com certas limitações. Para casas pequenas, onde a distância entre o Arduino-servidor e/ou aplicativo para os escravos são curtas, as limitações são irrelevantes. Para aplicações mais robustas e complexas existem alternativas como o uso de transceptores de radiofrequência, WiFi Shield ou uma rede de ZigBees pode ser mais apropriada.

Além da automação de uma residência, o presente trabalho pode servir de base para outros que precisem de uma forma de enviar dados de um lugar para outro, bastando para isso alterar o conteúdo dos botões na página web (do aplicativo também), e alterar o conteúdo que o Arduino irá usar como referência para identificar qual botão foi pressionado. Sobre os escravos, a única alteração necessária é modificar a resposta do Arduino dado o código de quatro letras, que não precisa ser alterado.

Para pessoas com maior conhecimento em programação mobile várias possibilidades são deixadas em aberto, entre elas: Adicionar mais funcionalidades; melhorar a robustez e design do aplicativo; e expandir para outras plataformas, como iOS.

Por fim, com progresso no avanço tecnológico sendo feito dia-a-dia, cada vez mais alternativas semelhantes e até melhores que o Arduino em alguns aspectos surgem no mercado como: Raspberry pi, que requer integração com Linux; e a placa BeagleBone.

Referências

APPINVENTOR. *About AppInventor*. 2012. Disponível em: <http://appinventor.mit-.edu/explore/about-us.html>. Acesso em: 23.3.2016. Citado na página 33.

ARDUINO. *SD Library*. 2005. Site www.arduino.cc. Disponível em: <https:/-/www.arduino.cc/en/Reference/SD>. Acesso em: 23.3.2016. Citado na página 31.

ARDUINO-ARDUINOSHIELDS. *Shields*. 2005. Site www.arduino.cc. Disponível em: <https://www.arduino.cc/en/Main/ArduinoShields>. Acesso em: 21.4.2016. Citado na página 36.

ARDUINO-ETHERNET. *Ethernet Shield*. 2005. Site www.arduino.cc. Disponível em: <https://www.arduino.cc/en/Reference/Ethernet>. Acesso em: 21.4.2016. Citado na página 37.

ARDUINO-SPI. *SPI library*. 2005. Site www.arduino.cc. Disponível em: <https://www-.arduino.cc/en/Reference/SPI>. Acesso em: 21.4.2016. Citado 3 vezes nas páginas 38, 40 e 41.

ATMEL. *ATMEL 8-BIT Microcontroller with 4/8/16/32KBytes In-System Programmable Flash Datasheet*. 2015. Site www.atmel.com. Disponível em: <http://www.atmel.com-/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf>. Acesso em: 3.2.2017. Citado na página 40.

BIONDO, R. M. *Automação Residencial (domótica) com Controle por Celular*. 2011. Site http://www.tcc.sc.usp.br/. Disponível em: <http://www.tcc.sc.usp.br/tce/disponiveis-/18/180450/tce-19102011-122815/?lang=br>. Acesso em: 24.4.2016. Citado na página 21.

CARDOZO, A. J.; GASPAS, G. A.; FONTANA, F. B. *Automação Residencial (domótica) com Controle por Celular*. 2013. Site eventoscientificos.ifsc.edu.br/. Disponível em: <http://eventoscientificos.ifsc.edu.br/index.php/sepei/sepei2013/paper/viewFile/101-/260>. Acesso em: 22.6.2017. Citado na página 24.

CIA, A. e. *Como gravar dados na memória EEPROM do Arduino*. 2015. Site arduinoecia.com.br. Disponível em: <http://www.arduinoecia.com.br/2015/03/gravar-dados-memoria-eeprom-arduino.html>. Acesso em: 14.7.2017. Citado na página 53.

CULTURA, B. *Arduino, o documentário*. 2010. Via upload no vimeo.com. Disponível em: <https://vimeo.com/31389230>. Acesso em: 23.3.2016. Citado na página 30.

CURREY, M. *Diferentes formas de configurar o módulo HC-05*. 2015. Site pessoal de Martyn Currey: www.martyncurrey.com/. Disponível em: <http://www.martyncurrey-.com/arduino-with-hc-05-bluetooth-module-at-mode/>. Acesso em: 23.3.2016. Citado na página 28.

DZIWIOR, P. *Bluetooth Introduction*. 2004. Site pessoal de Peter Dziwior: www.dziwior.org. Disponível em: <http://www.dziwior.org/Bluetooth/index.html>. Acesso em: 23.3.2016. Citado na página 26.

FOROUZAN, B. A. *Data Communications and Networking*. New York, NY, USA: McGrall Hill, 2007. Citado 2 vezes nas páginas 38 e 39.

GROUP, B. S. I. *Bluetooth Core Specification v5.0*. 2016. Site oficial: bluetooth.com. Disponível em: <https://www.bluetooth.com/specifications/adopted-specifications>. Acesso em: 23.3.2016. Citado na página 27.

HODGDON, C. *Adaptive Frequency Hopping for Reduced Interference between Bluetooth and Wireless LAN*. 2003. Site www.design-reuse.com. Disponível em: <https://www.design-reuse.com/articles/5715/adaptive-frequency-hopping-for-reduced-interference-between-bluetooth-and-wireless-lan.html>. Acesso em: 23.3.2016. Citado na página 25.

KANG, S.-M.; LEBLEBICI, Y. *CMOS: Digital Integrated Circuits*. New York, NY, USA: McGrall Hill, 2003. Citado na página 23.

LIMITED, A. *NEC Infrared Transmission Protocol*. 2013. Http://techdocs.altium.com. Disponível em: <http://techdocs.altium.com/display/FPGA-/NEC+Infrared+Transmission+Protocol>. Acesso em: 23.3.2016. Citado na página 32.

MICROCHIP. *Overview and Use of the PICmicro Serial Peripheral Interface*. 2015. Site www.microchip.com/. Disponível em: <http://ww1.microchip.com/downloads-/en/devicedoc/spi.pdf>. Acesso em: 21.4.2017. Citado 2 vezes nas páginas 40 e 41.

MONTARROYOS, W. C. M. e E. *Decodificando o Controle Remoto com a Placa de Som do PC*. 2002. Revista Brasileira de Ensino de Física. Disponível em: <http://www-.sbfisica.org.br/rbef/pdf/v24_497.pdf>. Acesso em: 2017-07-10 +0000. Citado na página 32.

RASHID, M. H. *Microelectronic Circuits: Analysis and Design*. Connecticut, CT, USA: Cengage Learning, 2011. Citado na página 23.

SHIRRIFF, K. *A Multi-Protocol Infrared Remote Library for the Arduino*. 2009. Site pessoal de Ken Shirriff: http://www.righto.com. Disponível em: <http://www.righto-.com/2009/08/multi-protocol-infrared-remote-library.html>. Acesso em: 23.3.2016. Citado na página 53.

SHIRRIFF, R. K. e K. *Biblioteca IRremote*. 2009. Github.com. Disponível em: <https://github.com/z3t0/Arduino-IRremote>. Acesso em: 08.7.2017. Citado na página 53.

SPARKFUN. *Bluetooth Basics*. 2013. Sparkfun.com. Disponível em: <https://learn-.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works>. Acesso em: 23.3.2016. Citado na página 24.

TEXASINSTRUMENS. *KeyStone Architecture Serial Peripheral Interface (SPI)*. 2012. Site www.ti.com/. Disponível em: <http://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf>. Acesso em: 21.4.2017. Citado na página 41.

UNION, I. T. *International Telecommunication Regulation*. 1989. World Administrative Telegraph and Telephone Conference (WATTC). Disponível em: <https://www.itu-.int/dms_pub/itu-t/oth/3F/01/T3F010000010001PDFE.pdf>. Acesso em: 03.6.2016. Citado na página 24.

WESTE, N. H. E.; HARRIS, D. M. *CMOS VLSI Design*. Massachusetts, MA, USA: Pearson, 2011. Citado na página 23.

Apêndices

APÊNDICE A – Programa de configuração do módulo Bluetooth

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(11, 12); // RX | TX
//SoftwareSerial BTSerial(2, 3); // RX | TX

void setup(){
  Serial.begin(9600);
  Serial.println("Enter AT commands:");
  BTSerial.begin(38400);  // HC-05 default speed in AT command
}

void loop(){
  // Keep reading from HC-05 and send to Arduino Serial Monitor
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  // Keep reading from Arduino Serial Monitor and send to HC-05
  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

APÊNDICE B – Programa no Arduino offline

```
#include <SoftwareSerial.h>
#include <IRremote.h>
#include <EEPROM.h>
SoftwareSerial blueSerial(6, 7);
IRsend irsend;
byte recvPin = 10;
IRrecv irrecv(recvPin);
decode_results  results;

unsigned long int soma[9];
unsigned long int num = 0;
unsigned long int resultadoValor = 0;
char hex[9];
int j = 0, cont=1, parcela1=0, parcela2=0,  dividendo=0, resto=0;

char Char = ' ';
char recebe[5];
byte indice = 0, parte[4];
String strBotao, strBotaoAnterior="nada";
char arrayBotao[9];

int hexTOdec(char c);
char decTOhex(int n);
long int arrayTOdec(char arrayBotao[9]);
void recebeCod();

void setup(){
  Serial.begin(9600);
  blueSerial.begin(9600);
  irrecv.enableIRIn();
}


void loop(){
    indice = 0; j=0;
    for(int i=0; i<9; i++){
      arrayBotao[i] = '\0';
    }
   for (int i = 0; i < 4; i++) recebe[i] = ' ';
    while (blueSerial.available() > 0){
```

```
41      if (indice < 4){
          Char = blueSerial.read();
43        recebe[indice] = Char;
          indice++;
45        recebe[indice] = '\0';
        } if (indice == 4) break;
47    }
    if (recebe[0] != ' '){
49      for (int i = 0; i < 4; i++) Serial.print(recebe[i]);
      Serial.println("");
51    }

53    if (irrecv.decode(&results)){
      Serial.println(results.value, HEX);
55      irrecv.resume();
    }
57    delay(100);

59    parcela1=0; parcela2=0; dividendo=0; resto=0; cont=1;
    if(recebe[2] != '-' &&  results.value != 0xFFFFFFFF){
61    if (recebe[0] == 'c'){
      if (recebe[3] == '0'){
63      for(int i=0, k=1; i<4; i++, k+=4){
          int pos = ((recebe[1]-97)*4+i)+((recebe[2]-48)*88 +1);
65        dividendo = EEPROM.read(pos);
          do{
67          resto = dividendo%16; dividendo /=16;
            arrayBotao[k] = resto<10 ? resto+48 : decTOhex(resto);
69          k--;
          } while(dividendo!=0);
71      }

73      irsend.sendSAMSUNG(arrayTOdec(arrayBotao), 32); irrecv.
    enableIRIn();

75    }else if (recebe[3]=='1'){
      for(int i=0, k=1; i<4; i++, k+=4){
77        int pos = ((recebe[1]-97)*4+i)+((recebe[2]-48)*84 +1) + 264;
          dividendo = EEPROM.read(pos);
79        do{
            resto = dividendo%16; dividendo /=16;
81          arrayBotao[k] = resto<10 ? resto+48 : decTOhex(resto);
            k--;
83        } while(dividendo!=0);
      }
85      irsend.sendSAMSUNG(arrayTOdec(arrayBotao), 32); irrecv.
    enableIRIn();
```

```
      }
87
    }else if (recebe[0] == 'p'){
89    do recebeCod(); while(cont<=9);

91    if (recebe[3] == '0'){
        for(int i=0, k=1; i<4; i++, k+=4){
93        int pos = ((recebe[1]-97)*4+i)+((recebe[2]-48)*88 +1);
          EEPROM.write(pos, parte[i]);
95        Serial.print("Valor ");
          Serial.print(parte[i]);
97        Serial.print(" gravado na pos. ");
          Serial.println(pos);
99      }
      }else if(recebe[3] == '1'){
101     for(int i=0, k=1; i<4; i++, k+=4){
          int pos = ((recebe[1]-97)*4+i)+((recebe[2]-48)*84 +1) + 264;
103       EEPROM.write(pos, parte[i]);
          Serial.print("Valor ");
105       Serial.print(parte[i]);
          Serial.print(" gravado na pos. ");
107       Serial.println(pos);
        }
109   }
    }
111 }
  Serial.flush(); Serial.print("-");
113}

115void recebeCod(){
      if (irrecv.decode(&results)){
117   strBotao = String(results.value, HEX);
      if(strBotao.equals(strBotaoAnterior)) cont++; else cont=1;
119   Serial.print(cont); Serial.print(" - recebido: "); Serial.println(
   results.value,HEX);
      strBotaoAnterior = strBotao;
121   irrecv.resume();
    }
123  delay(100);

125   if(cont==10){
      strBotao.toUpperCase();
127   strBotao.toCharArray(arrayBotao, 9);
      if(strBotao.length()==7){
129     for(int i=7; i>=0; i--) arrayBotao[i] = arrayBotao[i-1];
        arrayBotao[0] = '0';
131   }
```

```
      parcela1=0; parcela2=0; dividendo=0; resto=0;
133
      for(int i=0, k=0; i<4; i++, k+=2){
135     parcela1 = isDigit(arrayBotao[k]) ? arrayBotao[k]-48 : hexTOdec(
    arrayBotao[k]);
        parcela2 = isDigit(arrayBotao[k+1]) ? arrayBotao[k+1]-48 :
    hexTOdec(arrayBotao[k+1]);
137     parte[i] = parcela1*16 + parcela2;
      }
139   }
}
141
long int arrayTOdec(char arrayBotao[]){
143  for(int i=0; i<=7; i++){
      soma[i] = 0; Serial.print("arrayBtoao= "); Serial.print(arrayBotao[i
    ]);
145  }
  Serial.println("");
147
  long pot16 = 1;
149  for (int i = 7; i >= 0 ; i--){
    if(i!=7) pot16*=16;
151   Serial.print("Peso: "); Serial.print(pot16); Serial.print(" / ");

153   if (isDigit(arrayBotao[i])){
      soma[i] = (int(arrayBotao[i]) - 48) * pot16;
155   }else if (isAlpha(arrayBotao[i])){
      soma[i] = hexTOdec(arrayBotao[i]) * pot16;
157   }
    Serial.print("soma["); Serial.print(i); Serial.print("]= "); Serial.
   print(soma[i]); Serial.print(" / arrayBotao["); Serial.print(i);
    Serial.print("]= "); Serial.println(arrayBotao[i]);
159  }
  num=0;
161  for(int i=7; i>=0; i--){
    num += soma[i];
163  }
  Serial.print("num = ");
165  Serial.print(num);
  Serial.print(" = ");
167  Serial.println(num, HEX);
  return num;
169}

171
int hexTOdec(char c){
173  if(c=='A'||c=='a') return 10;
```

```
  else if(c=='B'||c=='b') return 11;
  else if(c=='C'||c=='c') return 12;
  else if(c=='D'||c=='d') return 13;
  else if(c=='E'||c=='e') return 14;
  else return 15;
}

char decTOhex(int n){
  if(n==10) return 'A';
  else if(n==11) return 'B';
  else if(n==12) return 'C';
  else if(n==13) return 'D';
  else if(n==14) return 'E';
  else return 'F';
}
```

APÊNDICE C – Programa no Arduino online

```
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
#define buff 80

#define loginFileName "Senha1~1.htm"
#define pageFileName "indexV~3.htm"
#define tvFileName "btnTV~1.htm"
#define dvdFileName "dvd.htm"
#define releFileName "rele~1.htm"
#define checkOk "?User=kek&password=123"
#define entrada "ajax_inputs"

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(10, 0, 0, 2);
EthernetServer server(80);

bool testeSD(const char fileName[]);
bool confereString(char *c, char confere[]);
void XML_response(EthernetClient cl);
void openFileSD(char fileName[], EthernetClient client);
void tv(EthernetClient client, char page[]);
void dvd(EthernetClient client, char page[]);
void rele(EthernetClient client, char page[]);

byte numApa = 0;
bool flag = 0;
File loginFileSD, pageFileSD;
boolean login = true;

void setup() {
  pinMode(10, OUTPUT);
  digitalWrite(10, HIGH);
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  server.begin();
  if (testeSD(loginFileName) == 0)
    return;
  if (testeSD(pageFileName) == 0)
    return;
```

```
  pinMode(8, INPUT_PULLUP);
  pinMode(9, INPUT_PULLUP);
}

void loop() {
  EthernetClient client = server.available();
  if (client) {
    boolean currentLineIsBlank = true;
    char page[buff] = {0};
    byte i = 0;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (i <= buff) page[i++] = c;
        if (c == '\n' && currentLineIsBlank) {
          client.println("HTTP/1.1 200 OK");
          if (confereString(&page[0], entrada)){
            client.println("Content-Type: text/xml");
            client.println("Connection: keep-alive");
            client.println();
            XML_response(client);
          }
          else if (confereString(&page[0], "aparelho")){
            client.println("Content-Type: text/xml");
            client.println("Connection: keep-alive");
            client.println();
            numAparelho(client);
          }
          else {
            client.println("Content-Type: text/html");
            client.println("Connection: keep-alive");
            client.println();
            if (confereString(&page[0], checkOk)){
              flag = 1;
              login = false;
            }
            if (login) openFileSD(loginFileName, client);
            else if (confereString(&page[0], "btnLogoff=Sair")){
              login = true;
              openFileSD(loginFileName, client);
            }
            else if (confereString(&page[0], "btnLogoff=Voltar")){
              flag = 1;
              openFileSD(pageFileName, client);
            }
            else if (confereString(&page[0], "apaTV")){
```

```
88                flag = 0;
                  openFileSD(tvFileName, client);
90                tv(client, page);
                }
92              else if (confereString(&page[0], "apaDVD")){
                  flag = 0;
94                openFileSD(arFileName, client);
                  dvd(client, page);
96              }
                else if (confereString(&page[0], "rele")){
98                flag = 0;
                  openFileSD(releFileName, client);
100               rele(client, page);
                }
102             else openFileSD(pageFileName, client);
              }
104           break;
            }
106         if (c == '\n') currentLineIsBlank = true;
            else if (c != '\r') currentLineIsBlank = false;
108       }
      }
110   delay(1);
      client.stop();
112 }
}
114

bool testeSD(const char fileName[]){
116   static unsigned int lup = 0;
    if (lup++ == 0) {
118     if (!SD.begin(4)){
        return 0;
120     }
    }
122 if (!SD.exists(fileName)) return 0;
    return 1;
124 }

126 void XML_response(EthernetClient cl) {
    int pot = analogRead(0);
128 int ldr = analogRead(1);
    cl.print("<?xml version=\"1.0\"?>"); cl.print("<inputs>"); cl.print("<
     b1>");
130 digitalRead(8) ? cl.print("ON") : cl.print("OFF");
    cl.print("</b1>"); cl.print("<b2>");
132 digitalRead(9) ? cl.print("ON") : cl.print("OFF");
```

```
  cl.print("</b2>"); cl.print("<a0>"); cl.print(pot); cl.print("</a0>");
    cl.print("<a1>"); cl.print(ldr); cl.print("</a1>"); cl.print("</
   inputs>");
}

void numAparelho(EthernetClient cl) {
  cl.print("<?xml version=\"1.0\"?>"); cl.print("<inputs>"); cl.print("<
   apa>"); cl.print(numApa); cl.print("</apa>"); cl.print("</inputs>");
}

bool confereString(char *c, char s[]){
  int tamS = strlen(s);
  int j = 0;
  for (int i = 0; i < 80; i++) {
    if (*(c + i) == s[j]) j++;
    else if (*(c + i) == s[0]) j = 1;
    else j = 0;
    if (j == tamS) {
      if (flag == 1)
        numApa = *(c + i + 1) - 48;
      return true;
    }
  }
  return false;
}

void openFileSD(char fileName[], EthernetClient client){
  File fileSD = SD.open(fileName);
  if (fileSD) {
    while (fileSD.available())
      client.write(fileSD.read());
    fileSD.close();
  }
}

void tv(EthernetClient client, char page[]){
  char deviceNumber = char (numApa + 47);
  char s[4] = {'c', '-', deviceNumber, '0'};
  if (confereString(page, "button=Power")) s[1] = 'a';
  else if (confereString(page, "button=CH\%2B")) s[1] = 'b';
  else if (confereString(page, "button=CH-")) s[1] = 'c';
  else if (confereString(page, "teste=VOL\%2B")) s[1] = 'd';
  else if (confereString(page, "button=VOL-")) s[1] = 'e';
  else if (confereString(page, "button=1")) s[1] = 'f';
  else if (confereString(page, "button=2")) s[1] = 'g';
  else if (confereString(page, "button=3")) s[1] = 'h';
  else if (confereString(page, "button=4")) s[1] = 'i';
```

```
  else if (confereString(page, "button=5")) s[1] = 'j';
  else if (confereString(page, "button=6")) s[1] = 'k';
  else if (confereString(page, "button=7")) s[1] = 'l';
  else if (confereString(page, "button=8")) s[1] = 'm';
  else if (confereString(page, "button=9")) s[1] = 'n';
  else if (confereString(page, "button=0")) s[1] = 'o';
  else if (confereString(page, "button=SRC")) s[1] = 'p';
  else if (confereString(page, "button=X")) s[1] = 'q';
  else if (confereString(page, "button=Up")) s[1] = 'r';
  else if (confereString(page, "button=OK")) s[1] = 's';
  else if (confereString(page, "button=Down")) s[1] = 't';
  else if (confereString(page, "button=Left")) s[1] = 'u';
  else if (confereString(page, "button=Right")) s[1] = 'v';
  else if (confereString(page, "button=Mute")) s[1] = 'x';
  if (Serial.available()) Serial.write(s, 4);
  delay(10);
}

void dvd(EthernetClient client, char page[]){
  char deviceNumber = char (numApa + 47);
  char s[4] = {'c', '-', deviceNumber, '1'};
  if (confereString(page, "button=Power"))  s[1] = 'a';
  else if (confereString(page, "button=Abrir")) s[1] = 'b';
  else if (confereString(page, "button=Parar")) s[1] = 'c';
  else if (confereString(page, "button=Menu")) s[1] = 'd';
  else if (confereString(page, "button=Legenda")) s[1] = 'e';
  else if (confereString(page, "button=Audio")) s[1] = 'f';
  else if (confereString(page, "button=Iniciar")) s[1] = 'g';
  else if (confereString(page, "button=USB")) s[1] = 'h';
  else if (confereString(page, "button=Pause")) s[1] = 'i';
  else if (confereString(page, "button=Retroceder")) s[1] = 'j';
  else if (confereString(page, "button=Avancar")) s[1] = 'k';
  else if (confereString(page, "button=Acelerar"))  s[1] = 'l';
  else if (confereString(page, "teste=MenuDisco")) s[1] = 'm';
  else if (confereString(page, "button=Up")) s[1] = 'n';
  else if (confereString(page, "button=OK")) s[1] = 'o';
  else if (confereString(page, "button=Down")) s[1] = 'p';
  else if (confereString(page, "button=Left")) s[1] = 'q';
  else if (confereString(page, "button=Right")) s[1] = 'r';
  else if (confereString(page, "button=Retomar")) s[1] = 's';
  else if (confereString(page, "button=Cancelar")) s[1] = 't';
  else if (confereString(page, "button=Informacao")) s[1] = 'u';
  if (Serial.available()) Serial.write(s, 4);
  delay(10);
}

void rele(EthernetClient client, char page[]){
```

```
224   char deviceNumber = char (numApa + 47);
      char s[4] = {'c', '-', deviceNumber, '2'};
226   if (confereString(page, "RESET")) s[1] = 'a';
      else if (confereString(page, "SALA1"))  s[1] = 'b';
228   else if (confereString(page, "SALA2"))  s[1] = 'c';
      else if (confereString(page, "SALA3"))  s[1] = 'd';
230   else if (confereString(page, "QUARTO1"))  s[1] = 'e';
      else if (confereString(page, "QUARTO2"))  s[1] = 'f';
232   else if (confereString(page, "QUARTO3"))  s[1] = 'g';
      else if (confereString(page, "LUZ1")) s[1] = 'h';
234   else if (confereString(page, "LUZ2")) s[1] = 'i';
      else if (confereString(page, "LUZ3")) s[1] = 'j';
236   if (Serial.available()) Serial.write(s, 4);
      delay(10);
238 }
```

APÊNDICE D – Código alfabético dos botões

Botões da TV	Código	Botões do DVD	código
On/Off	a	On/Off	a
SubirCanal	b	Abrir	b
DescerCanal	c	Parar	c
SubirVol	d	Menu	d
DescerVol	e	Legenda	e
1	f	Audio	f
2	g	Iniciar	g
3	h	USB	h
4	i	Pause	i
5	j	Retroceder	j
6	k	Avançar	k
7	l	Acelerar	l
8	m	MenuDisco	m
9	n	Cima	n
0	o	OK	o
SRC	p	Baixo	p
X	q	Esquerda	q
Cima	r	Direita	r
OK	s	Retomar	s
Baixo	t	Cancelar	t
Esquerda	u	Informação	u
Direita	v		

APÊNDICE E – Site com tela inicial

```
<!DOCTYPE html>
<html>
    <head>
    <style type = "text/css">
      body{
        background: #004c4d;
        color: #fff;
        font-family: Verdana;
      }
      .lol{
      width: 15%;
      margin: auto;
      }
      .exit{
        text-align: right;
        widht: 50px;
      }
      .btn{
      font-family: Lucida;
      font-size: 22px;
      border: none;
      padding: 7px 15px;
      background: #0072c6;
      color: #EEE;
      cursor: pointer;
      border: 1px solid #bbb;
      border-radius: 2px;
      }
      .btn2{
        width: 160px;
        text-align: left;
        font-family: Haveltica;
        font-size: 16px;
        background: #f9f9f9;
        cursor: pointer;
        border: none;
      }
      .dropbtn1, .dropbtn2, .dropbtn3 {
        text-align: center;
        width: 380px;
```

```
    height: 50px;
    font-family: Lucida;
    font-size: 20px;
    background: #EEE9E9;
    color: #333;
    cursor: pointer;
    border-radius: 2px;
    border: line;
    display: block;
  position: relative;
  }

  .cabecalho{
    text-align: center;
    font-size: 22px;
    font-family: Verdana;
    color: #FFF;
  }

  #pos{
    margin: auto;

    display: block;
  }

  .dropbtn:hover, .dropbtn:focus {
    background-color: #3e8e41;
  }

  .dropdown {
    display: inline-block;
  }

  .dropdown-content {
    display: none;
    position: absolute;
    background-color: #EEE9E9;
    min-width: 160px;
    overflow: auto;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    text-align: center;
  }

  .dropdown-content a, input{
    color: black;
    padding: 16px;
    text-decoration: none;
```

```
88          display: block;
          }
90
          .dropdown a:hover {background-color: #CDC9C9}
92
          .btn2:hover {background-color: #CDC9C9}
94
          .dropbtn1:hover {background-color: #CDC9C9}
96        .dropbtn2:hover {background-color: #CDC9C9}
          .dropbtn3:hover {background-color: #CDC9C9}
98
          .show {display:block;}
100
          #esquerda{
102         width: 385px;
            height: 300px;
104         float: left;
            text-align: left;
106       }
          #direita{
108         width: 160px;
            float: left;
110         text-align: left;
          }
112       #btn02{text-align:center}
          .btn:hover{background-color: #0068b3}
114
      </style>
116       <title>Piratino</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
118       <script>
        function GetArduinoInputs()
120     {
          nocache = "&nocache=" + Math.random() * 1000000;
122       var request = new XMLHttpRequest();
          request.onreadystatechange = function()
124       {
            if (this.readyState == 4) {
126           if (this.status == 200) {
                if (this.responseXML != null) {
128               document.getElementById("input1").innerHTML = this.
    responseXML.getElementsByTagName('b1')[0].childNodes[0].nodeValue;
                  document.getElementById("input2").innerHTML = this.
    responseXML.getElementsByTagName('b2')[0].childNodes[0].nodeValue;
130               document.getElementById("input3").innerHTML = this.
    responseXML.getElementsByTagName('a0')[0].childNodes[0].nodeValue;
```

```
                document.getElementById("input4").innerHTML = this.
    responseXML.getElementsByTagName('a1')[0].childNodes[0].nodeValue;
132             }
              }
134           }
        }
136       request.open("GET", "ajax_inputs" + nocache, true);
          request.send(null);
138       setTimeout('GetArduinoInputs()', 1000);
      }
140   </script>
  <script language="JavaScript">
142     javascript:window.history.forward(1);
  </script>
144   </head>
    <body onload="GetArduinoInputs()">
146   <div class="cabecalho">
      <h1>Sele  o de Aparelhos</h1>
148   </div>

150       Lmpada 1: <span id="input1"> </span>
    <div style="height: 5px;"></div>
152       Lmpada 2: <span id="input2"> </span>
    <div style="height: 5px;"></div>
154       Temperatura: <span id="input3"> </span>
    <div style="height: 5px;"></div>
156   Sinal Analgico: <span id="input4"> </span>/1023

158   <div style="height: 25px;"></div>
    <div class="dropdown">
160   <div id="esquerda">
    <div id="pos">
162   <button onclick="myFunction1()" class="dropbtn1">TV</button>
    <div style="height: 5px;"></div>
164   <button onclick="myFunction2()" class="dropbtn2">Rel </button>
    <div style="height: 5px;"></div>
166   <button onclick="myFunction3()" class="dropbtn3">DVD</button>
    </div>
168   </div>

170    <div id="direita">
       <form>
172    <div id="myDropdown1" class="dropdown-content">
       <input type = "submit" name = "apaTV1" value = "TV01" class="btn2"
  >
174    <input type = "submit" name = "apaTV2" value = "TV02" class="btn2"
  >
```

```
      <input type = "submit" name = "apaTV3" value = "TV03" class="btn2"
    >
176   </form>
      </div>
178
      <form>
180   <div id="myDropdown2" class="dropdown-content">
      <input type = "submit" name = "rele1" value = "REL 01" class="
    btn2">
182   <input type = "submit" name = "rele2" value = "REL 02" class="
    btn2">
      <input type = "submit" name = "rele3" value = "REL 03" class="
    btn2">
184   </div>
      </form>
186
      <form>
188   <div id="myDropdown3" class="dropdown-content">
      <input type = "submit" name = "apaDVD1" value = "DVD01" class="
    btn2">
190   <input type = "submit" name = "apaDVD2" value = "DVD02" class="
    btn2">
      <input type = "submit" name = "apaDVD3" value = "DVD03" class="
    btn2">
192   </div>
      </form>
194   </div>
     </div>
196
     <form method = "GET">
198    <div class = "exit" style="text-align: left">
         <input type = "submit" name = "btnLogoff" value = "Sair" class =
     "btn">
200    </div>
     </form>
202
     <script>
204  var a = ["myDropdown1", "myDropdown2", "myDropdown3"];
     var aux = "PQP";
206  function myFunction1() {
     for(var i=0; i<3; i++)
208     document.getElementById(a[i]).classList.remove("show");
       document.getElementById("myDropdown1").classList.toggle("show");
210    aux = ".dropbtn1";
     }
212
     function myFunction2() {
```

```
214     for(var i=0; i<3; i++)
           document.getElementById(a[i]).classList.remove("show");
216       document.getElementById("myDropdown2").classList.toggle("show");
          aux = ".dropbtn2";
218     }

220     function myFunction3() {
        for(var i=0; i<3; i++)
222        document.getElementById(a[i]).classList.remove("show");
          document.getElementById("myDropdown3").classList.toggle("show");
224       aux = ".dropbtn3";
        }
226     window.onclick = function(event) {
          if (!event.target.matches(aux)) {
228       var dropdowns = document.getElementsByClassName("dropdown-content"
       );
          var i;
230       for (i = 0; i < dropdowns.length; i++) {
            var openDropdown = dropdowns[i];
232         if (openDropdown.classList.contains('show')) {
            openDropdown.classList.remove('show');
234         }
          }
236       }
        }
238     </script>
        </body>
240 </html>
```

APÊNDICE F – Site com tela de login

```
<!DOCTYPE html>
<html>
  <head>
    <title>SENHA</title>
    <style type = "text/css">
    .kek{
      text-align: center;
      font-family: Lucida;
      font-size: 30px;
    }
    .texto{
      align: center;
      font-size: 15px;
      border-size: 1px;
      padding-top: 5px;
      padding-left: 6px;
      padding-bottom: 6px;
      line-height: 100%;
    }
    .lol{
      width: 23.5%;
      margin: auto;
    }
    .btn{
      font-family: Lucida;
      font-size: 22px;
      border: none;
      padding: 7px 15px;
      background: #0072c6;
      color: #EEE;
      cursor: pointer;
      border: 1px solid #bbb;
      border-radius: 2px;
    }
    body{
      background: #004c4d;;
      color: #fff;
    }
    .btn:hover{background-color: #0068b3}
    </style>
```

```
  </head>
42
  <body>
44    <form method = "GET">
        <div class = "kek">TESTE DE LOGIN</div>
46      <div class="lol">
          <input type = "text" name = "User" class = "texto" id = "email"
    placeholder = "Email" size = "35px"/>
48        <div style="height: 10px;"></div>
          <input type = "password" name = "password" class="texto" id = "
    passwordId" placeholder = "Senha" size = "35px"/>
50        <div style="height: 20px;"></div>
          <input class = "btn" type = "submit" value = "Entrar">
52      </div>
      </form>
54  </body>
</html>
```

APÊNDICE G – Site com controle de TV

```
<!DOCTYPE html>
<html>
  <head>
    <style type = "text/css">
    body{
        background: #004c4d;
        color: #fff;
        font-family: Verdana;
      }
    .exit{
        text-align: right;
        widht: 50px;
      }
    .btn{
      font-family: Lucida;
      font-size: 22px;
      border: none;
      padding: 7px 15px;
      background: #0072c6;
      color: #EEE;
      cursor: pointer;
      border: 1px solid #bbb;
      border-radius: 2px;
      }
    .btn2{
        width: 62px;
        height: 41px;
        font-family: Lucida;
        font-size: 20px;
        background: #7c828;
        color: #333;
        cursor: pointer;
        border: 1px solid #bbb;
        border: line;
        border-radius: 2px;
        padding-right: 6px;
      }
      #btn02{
        display: inline-block;
      }
```

```
41
      #esquerda{
43      width: 220px;
        height: 235px;
45      float: left;
        text-align: left;
47    }
      #direita{
49      width: 200px;
        height: 235px;
51      float: left;
        text-align: center;
53    }
      .cabecalho{
55      text-align: center;
        font-size: 22px;
57      font-family: Verdana;
        color: #FFF;
59    }
      .btn:hover{background-color: #0068b3}
61    .btn2:hover{background-color: #CDC9C9}
   </style>
63 <title>Controle Remoto - TV</title>
   <meta name="viewport" content="width=device-width, initial-scale
  =1.0, maximum-scale=1.0">
65 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <script>
67   function GetArduinoInputs()
     {
69     nocache = "&nocache=" + Math.random() * 1000000;
       var request = new XMLHttpRequest();
71     request.onreadystatechange = function()
       {
73       if (this.readyState == 4) {
           if (this.status == 200) {
75           if (this.responseXML != null) {
               document.getElementById("input1").innerHTML = this.
  responseXML.getElementsByTagName('apa')[0].childNodes[0].nodeValue;
77           }
           }
79       }
       }
81     request.open("GET", "aparelho" + nocache, true);
       request.send(null);
83     setTimeout('GetArduinoInputs()', 1000);
     }
85 </script>
```

```
  </head>
87  <body onload="GetArduinoInputs()">
  <div class="cabecalho">
89    <h1>Controle da TV <span id="input1"> </span></h1>
  </div>
91    <div style="height: 25px;"></div>
    <div id = "btn02">
93      <form action = "apaTV" method = "GET">
        <div id="esquerda">
95          <input type = "submit" name = "button" value = "Power" class =
    "btn2">
            <div style="height: 5px;"></div>
97          <input type = "submit" name = "button" value = "1" class = "
    btn2">
            <input type = "submit" name = "button" value = "2" class = "
    btn2">
99          <input type = "submit" name = "button" value = "3" class = "
    btn2">
            <div style="height: 5px;"></div>
101         <input type = "submit" name = "button" value = "4" class = "
    btn2">
            <input type = "submit" name = "button" value = "5" class = "
    btn2">
103         <input type = "submit" name = "button" value = "6" class = "
    btn2">
            <div style="height: 5px;"></div>
105         <input type = "submit" name = "button" value = "7" class = "
    btn2">
            <input type = "submit" name = "button" value = "8" class = "
    btn2">
107         <input type = "submit" name = "button" value = "9" class = "
    btn2">
            <div style="height: 5px;"></div>
109         <input type = "submit" name = "button" value = "SRC" class = "
    btn2">
            <input type = "submit" name = "button" value = "0" class = "
    btn2">
111         <input type = "submit" name = "button" value = "X" class = "
    btn2">
        </div>
113     <div id="direita">
          <div style="height: 46px;"></div>
115       <input type = "submit" name = "button" value = "VOL+" class =
    "btn2">
          <input type = "submit" name = "button" value = "CH+" class = "
    btn2">
117       <div style="height: 5px;"></div>
```

```
            <input type = "submit" name = "button" value = "VOL-" class =
    "btn2">
119         <input type = "submit" name = "button" value = "CH-" class = "
    btn2">
            <div style="height: 5px;"></div>
121         <input type = "submit" name = "button" value = "Left" class =
    "btn2">
            <input type = "submit" name = "button" value = "OK" class = "
    btn2">
123         <input type = "submit" name = "button" value = "Right" class =
     "btn2">
            <div style="height: 5px;"></div>
125         <input type = "submit" name = "button" value = "Up" class = "
    btn2">
            <input type = "submit" name = "button" value = "Mute" class =
    "btn2">
127         <input type = "submit" name = "button" value = "Down" class =
    "btn2">
          </div>
129     </form>
        <form action = "main" method = "GET">
131       <div class = "exit" style="text-align: left">
            <input type = "submit" name = "btnLogoff" value = "Voltar"
    class = "btn">
133       </div>
        </form>
135   </div>
  </body>
137 </html>
```

APÊNDICE H – Site com controle de DVD

```
<!DOCTYPE html>
<html>
  <head>
    <style type = "text/css">
    body{
        background: #004c4d;
        color: #fff;
        font-family: Verdana;
        <!--background: f0f0f0;-->
        <!--color: #145678;-->
      }
    .exit{
        text-align: right;
        widht: 50px;
      }
    .btn{
      font-family: Lucida;
      font-size: 22px;
      border: none;
      padding: 7px 15px;
      background: #0072c6;
      color: #EEE;
      cursor: pointer;
      border: 1px solid #bbb;
      border-radius: 2px;
      }
    .btn2{
        width: 110px;
        height: 41px;
        font-family: Lucida;
        font-size: 20px;
        background: #7c828;
        color: #333;
        cursor: pointer;
        border: 1px solid #bbb;
        border: line;
        border-radius: 2px;
        padding-right: 6px;
      }
      #btn02{
```

```
41        display: inline-block;
      }
43
      #esquerda{
45        width: 350px;
          height: 235px;
47        float: left;
          text-align: left;
49    }
      #direita{
51        width: 400px;
          height: 235px;
53        float: left;
          text-align: center;
55    }
      .cabecalho{
57        text-align: center;
          font-size: 22px;
59        font-family: Verdana;
          color: #FFF;
61    }
      .btn:hover{background-color: #0068b3}
63    .btn2:hover{background-color: #CDC9C9}
   </style>
65 <title>Controle Remoto - DVD</title>
   <meta name="viewport" content="width=device-width, initial-scale
  =1.0, maximum-scale=1.0">
67 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <script>
69   function GetArduinoInputs()
     {
71     nocache = "&nocache=" + Math.random() * 1000000;
       var request = new XMLHttpRequest();
73     request.onreadystatechange = function()
       {
75       if (this.readyState == 4) {
           if (this.status == 200) {
77           if (this.responseXML != null) {
               // extract XML data from XML file (containing switch
  states and analog value)
79             document.getElementById("input1").innerHTML = this.
  responseXML.getElementsByTagName('apa')[0].childNodes[0].nodeValue;
           }
81       }
       }
83   }
     request.open("GET", "aparelho" + nocache, true);
```

```
85          request.send(null);
            setTimeout('GetArduinoInputs()', 1000);
87        }
      </script>
89    </head>
      <body onload="GetArduinoInputs()">
91    <div class="cabecalho">
        <h1>Controle do DVD <span id="input1"> </span></h1>
93    </div>
        <div style="height: 25px;"></div>
95      <div id = "btn02">
          <form action = "apaDVD" method = "GET">
97          <div id="esquerda">
              <input type = "submit" name = "button" value = "Open/Close"
     class = "btn2">
99            <input type = "submit" name = "button" value = "Stop &#x220E;"
       class = "btn2">
              <input type = "submit" name = "button" value = "Menu" class =
     "btn2">
101           <div style="height: 5px;"></div>
              <input type = "submit" name = "button" value = "Sub" class = "
     btn2">
103           <input type = "submit" name = "button" value = "Audio" class =
       "btn2">
              <input type = "submit" name = "button" value = "Play &#x25BA
     ;&#x2225;" class = "btn2">
105           <div style="height: 5px;"></div>
              <input type = "submit" name = "button" value = "USB" class = "
     btn2">
107           <input type = "submit" name = "button" value = "Title" class =
       "btn2">
              <input type = "submit" name = "button" value = "REV &#x226a;"
     class = "btn2">
109           <div style="height: 5px;"></div>
              <input type = "submit" name = "button" value = "FWD &#x226b;"
     class = "btn2">
111           <input type = "submit" name = "button" value = "Prev &#x7c;&#
     x226a;" class = "btn2">
              <input type = "submit" name = "button" value = "Next &#x226b
     ;&#x7c;" class = "btn2">
113         </div>
            <div id="direita">
115           <input type = "submit" name = "button" value = "Cancel" class
     = "btn2">
              <input type = "submit" name = "button" value = "Setup" class =
       "btn2">
117           <div style="height: 5px;"></div>
```

```
            <input type = "submit" name = "button" value = "Info" class =
    "btn2">
119         <input type = "submit" name = "button" value = "On/Off" class
    = "btn2">
            <div style="height: 5px;"></div>
121         <input type = "submit" name = "button" value = "Left &larr;"
    class = "btn2">
            <input type = "submit" name = "button" value = "OK" class = "
    btn2">
123         <input type = "submit" name = "button" value = "Right &rarr;"
    class = "btn2">
            <div style="height: 5px;"></div>
125         <input type = "submit" name = "button" value = "Up &uarr;"
    class = "btn2">
            <input type = "submit" name = "button" value = "Mute" class =
    "btn2">
127         <input type = "submit" name = "button" value = "Down &darr;"
    class = "btn2">
          </div>
129     </form>
      <form action = "main" method = "GET">
131     <div class = "exit" style="text-align: left">
          <input type = "submit" name = "btnLogoff" value = "Voltar"
    class = "btn">
133       </div>
      </form>
135   </div>
  </body>
137 </html>
```

APÊNDICE I – Site com controle de Relé

```
<!DOCTYPE html>
<html>
  <head>
    <style type = "text/css">
    body{
        background: #004c4d;
        color: #fff;
        font-family: Verdana;
      }
    .exit{
        text-align: right;
        widht: 50px;
      }
    .btn{
      font-family: Lucida;
      font-size: 22px;
      border: none;
      padding: 7px 15px;
      background: #0072c6;
      color: #EEE;
      cursor: pointer;
      border: 1px solid #bbb;
      border-radius: 2px;
      }
    .btn2{
        width: 105px;
        height: 65px;
        font-family: Lucida;
        font-size: 20px;
        background: #7c828;
        color: #333;
        cursor: pointer;
        border: 1px solid #bbb;
        border:line;
        border-radius: 2px;
        padding-right: 6px;

      }
      .cabecalho{
        text-align: center;
```

```
41        font-size: 22px;
          font-family: Verdana;
43        color: #FFF;
        }
45      #btn02{
          display: inline-block;
47      }
        .btn2:hover{background-color: #CDC9C9}
49      .btn:hover{background-color: #0068b3}
    </style>
    <title>Controle Remoto - Rele</title>
    <meta name="viewport" content="width=device-width, initial-scale
  =1.0, maximum-scale=1.0">
53  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script>
55    function GetArduinoInputs()
      {
57      nocache = "&nocache=" + Math.random() * 1000000;
        var request = new XMLHttpRequest();
59      request.onreadystatechange = function()
        {
61        if (this.readyState == 4) {
            if (this.status == 200) {
63            if (this.responseXML != null) {
                document.getElementById("input1").innerHTML = this.
  responseXML.getElementsByTagName('apa')[0].childNodes[0].nodeValue;
65            }
            }
67        }
        }
69      request.open("GET", "aparelho" + nocache, true);
        request.send(null);
71      setTimeout('GetArduinoInputs()', 1000);
      }
73  </script>
  </head>
75<body>
    <body onload="GetArduinoInputs()">
77  <div class="cabecalho">
      <h1>Controle de Cargas <span id="input1"> </span></h1>
79  </div>
    <center>
81  <div id = "btn02">
      <form action = "rele" method = "GET">
83      <div style="height: 50px;"></div>
        <input type = "submit" name = "button" value = "RESET" class = "
  btn2">
```

```
85          <div style="height: 5px;"></div>
            <input type = "submit" name = "button" value = "SALA1" class = "
    btn2">
87          <input type = "submit" name = "button" value = "SALA2" class = "
    btn2">
            <input type = "submit" name = "button" value = "SALA3" class = "
    btn2">
89          <div style="height: 5px;"></div>
            <input type = "submit" name = "button" value = "QUARTO1" class =
     "btn2">
91          <input type = "submit" name = "button" value = "QUARTO2" class =
     "btn2">
            <input type = "submit" name = "button" value = "QUARTO3" class =
     "btn2">
93          <div style="height: 5px;"></div>
            <input type = "submit" name = "button" value = "LUZ1" class = "
    btn2">
95          <input type = "submit" name = "button" value = "LUZ2" class = "
    btn2">
            <input type = "submit" name = "button" value = "LUZ3" class = "
    btn2">
97          <div style="height: 5px;"></div>
        </form>
99      <form action = "main" method = "GET">
          <div class = "exit" style="text-align: left">
101         <input type = "submit" name = "btnLogoff" value = "Voltar"
    class = "btn">
          </div>
103     </form>
     </div>
105  </center>
   </body>
107 </html>
```