UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO

EDUARDO HENRIQUE HORTENCIO PEREIRA

Soluções inteligentes e de baixo custo para a automação residencial utilizando smartphones

Autor: Eduardo Henrique Hortencio Pereira

Orientador: Prof. Dr. Evandro Luís Linhari Rodrigues

São Carlos – SP
2014

EDUARDO HENRIQUE HORTENCIO PEREIRA

Soluções inteligentes e de baixo custo para a automação residencial utilizando smartphones

Trabalho de Conclusão de Curso
apresentado à Escola de Engenharia de São
Carlos, da Universidade de São Paulo.

Curso de Engenharia Elétrica com ênfase em
Sistemas de Energia e Automação

ORITENTADOR: Prof. Dr. Evandro Luís Linhari Rodrigues

São Carlos – SP
2014

Pereira, Eduardo Henrique Hortencio
P436s Soluções inteligentes e de baixo custo para automação residencial / Eduardo Henrique Hortencio Pereira; orientador Evandro Luís Linhari Rodrigues. São Carlos, 2014.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2014.

1. Automação residencial. 2. Raspberry Pi. 3. IOS. 4. Arduino Uno. 5. Smartphone. 6. Relé. I. Título.

FOLHA DE APROVAÇÃO

Nome: Eduardo Henrique Hortêncio Pereira

Título: "Soluções inteligentes e de baixo custo para a automação residencial utilizando smartphones"

Trabalho de Conclusão de Curso defendido e aprovado em 26 / 11 / 2014,

com NOTA 9,5 (NOVE , CINCO), pela Comissão Julgadora:

Prof. Associado Evandro Luís Linhari Rodrigues - (Orientador - SEL/EESC/USP)

Prof. Dr. José Carlos de Melo Vieira Júnior - (SEL/EESC/USP)

Prof. Dr. Dennis Brandão - (SEL/EESC/USP)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

DEDICATÓRIA

Dedico esse projeto a minha família, que sempre me apoiou mesmo momentos mais difíceis. Dedico também a todos os amigos que fiz durante minha graduação, que sem dúvida tornaram esses anos inesquecíveis para mim.

AGRADECIMENTOS

Agradeço primeiramente à minha família e amigos, por todo o apoio e ajuda prestada.

Gostaria também de agradecer todos os professores que participaram da minha graduação e que contribuíram para o meu desenvolvimento pessoal e profissional. Gostaria de agradecer meu orientador Evandro, por todo o incentivo a participar do intercâmbio, que foi sem dúvidas um dos melhores anos da minha vida. Agradeço também pela confiança e liberdade que me foi dada para elaborar e implementar minhas próprias ideias para esse projeto.

RESUMO

A elaboração desse projeto consistiu no desenvolvimento de um dispositivo para a automação residencial remota via smartphone (plataforma IOS). As plataformas para processamento central utilizadas foram a *Raspberry Pi* e o *Arduino Uno*, que viabilizaram a comunicação com o dispositivo controlador e os relés conectados com os equipamentos domésticos. O usuário entrará com seu comando no aplicativo, que gerará uma mensagem que fluirá pelo *Twitter*, *Raspberry Pi*, *Arduino Uno* até chegar ao relé acionando o dispositivo desejado. Após a implementação do projeto uma limitação foi observada. O transmissor e receptor de radiofrequência utilizados na comunicação entre o a *Raspberry Pi* e o *Arduino Uno* comprometeram a comunicação em algumas situações devido a sua baixa qualidade. Assim, foi possível gerar uma solução de baixo custo para o controle dos dispositivos domésticos que funciona porém que apresenta uma limitação.

Palavras-chave: Automação residencial; *Raspberry Pi*; *IOS*; *Arduino Uno*; *Smartphone*; Relé.

ABSTRACT

The objective of this project was developing a home automation system in which the control is made mainly via smartphone (*IOS* platform). The platforms used to process the information were the *Raspberry Pi* combined with the *Arduino Uno* that allowed the communication between the controlling device and the digital relay connected with the home appliances. The user will enter a command through the application, that will generate a message, which will flow through *Twitter*, *Raspberry Pi*, *Arduino Uno* and reach the relay activating the desired device. After the implementation of the project one limitation was identified. The radiofrequency transmitter and receiver that are responsible for the *Raspberry Pi* and the *Arduino Uno* communication sometimes do not work due its low quality. Therefore, it was possible to generate a low cost solution to control home appliances that works but has a limitation.

Keywords: Home automation; *Raspberry Pi*; IOS; *Arduino Uno*; Smartphone; Relay.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Amplitude Modulation
API	Application Programming Interface
APP	Application
ARM	Advanced RISC Machine
FM	Frequency Modulation
GND	Ground
GPIO	General Purpose Input Output
GPU	Graphics Processing Unit
HMDI	High Definition Multimedia Interface
RAM	Random Access Memory
RF	Radiofrequência
USB	Universal Serial Bus
VCC	Voltage collector-to-collector

SUMÁRIO

1 INTRODUÇÃO

A automação residencial ou a domótica consiste no controle integrado de aparelhos e processos domésticos a fim de substituir grande parte do trabalho humano por sistemas eletrônicos programados. A partir da introdução massiva da tecnologia nas últimas décadas, a programação e controle de aparelhos domésticos tem se tornado cada vez mais viável economicamente. Como exemplo, tornou-se possível controlar à distância as luzes de uma casa, a temperatura ambiente desejada e até mesmo o alarme e as fechaduras das portas.

O crescimento dessa indústria nos tempos atuais é inquestionável globalmente. No Brasil já existem diversas empresas fornecendo serviços e produtos para residências, porém a maioria é de médio ou pequeno porte. O custo para automatizar uma casa ainda não é acessível para todas as classes sociais quando se leva em conta o custo benefício desses serviços.

Existe uma grande variedade de produtos no mercado, apresentando soluções que geralmente envolvem equipamentos eletrônicos, como transmissores e receptores de radiofrequência ou de sinais infravermelho por exemplo.

Os controles remotos universais talvez sejam os produtos de automação residencial mais comuns no mercado. Eles demonstram como componentes eletrônicos citados anteriormente podem se transformar em soluções de automação.

A internet passou a ser utilizada em inúmeras soluções propostas pelo mercado, pois observou-se nela inúmeras possibilidades. Dentre elas pode-se citar o monitoramento das residências à distância, através de câmeras que enviam suas imagens para um servidor web permitindo que o usuário visualize sua casa de qualquer lugar no mundo desde que haja um computador com acesso à internet. Existem também sistemas de alarme no mercado que são capazes de enviar alertas ao morador e acionar a polícia automaticamente quando uma residência é invadida. Portanto a automação residencial combinada à internet deixou de ser apenas um meio de aumentar a comodidade de seus moradores e passou a atuar como um instrumento de auxílio na segurança das moradias.

Dentre os aparelhos eletrônicos de maior popularidade no mundo moderno estão os smartphones. O desenvolvimento de *hardwares* e *softwares* altamente eficientes fez com que estes dispositivos se tornassem parte do cotidiano das pessoas, permitindo a conexão com a internet com a mesma funcionalidade de um computador (RUWAIDA; MINKKINEN, 2013).

Um dos aspectos de maior relevância dos smartphones é a capacidade de conexão e comunicação com outros dispositivos, o que permite o seu uso como *mouse* para o computador, controle remoto para a TV, dentre outros. Assim, o uso do smartphone como ponto central para o controle automatizado de residências tem se tornado tema para a elaboração de várias soluções de mercado, na tentativa de unificar os aparelhos domésticos em um único aplicativo, proporcionando ao consumidor maior controle sobre a sua casa.

A possibilidade de possuir o controle de sua residência na palma das mãos vem movimentando o mercado de automação residencial. O número de produtos que emergiu nos últimos anos com a popularização dos smartphones é extremamente alto.

A grande maioria de produtos de mercado que utilizam smartphones para o controle central são focados nos dois maiores sistemas operacionais para celulares que são o *IOS* da empresa Apple e o *Android* da empresa *Google*. Para este trabalho o smartphone utilizado que é o *Iphone* da empresa *Apple*, logo portador do *IOS*. Para se ter ideia da dimensão de produtos que envolvem essas plataformas basta acessar a *Apple Store* onde se encontram todos os aplicativos e digitar “Automação Residencial“ para visualizar centenas de aplicativos desenvolvidos para esse funcionalidade.

Para que haja comunicação e controle entre os smartphones e os demais aparelhos existentes em uma residência é preciso que exista um ponto que seja capaz de processar toda a informação transmitida e recebida. Os minicomputadores como a *Raspberry Pi* estão se tornando cada vez mais populares e mais acessíveis, portanto estão se tornando uma alternativa cada vez mais usada para realizar todo o processamento de dados.

Minicomputadores como a *Raspberry Pi* são plenamente capazes de realizar o processamento de dados e comunicação, via internet, com um dispositivo controlador, como o smartphone. Lançado em 2012, o *Raspberry Pi* pesa apenas 45 g, o que o torna perfeito para a automação residencial, já que pode ser instalado no interior da caixa elétrica, ou substituir um termostato instalado na parede (DENNIS, 2013). O *Arduíno Uno* é uma opção mais acessível financeiramente, porém possui uma capacidade de processamento reduzida em relação a *Raspberry Pi*. Esses minicomputadores contribuíram para o crescimento do mercado de automação para residências pois tiram a necessidade de se utilizar um computador para o processamento de dados. Portanto a redução de custo com essas mudanças foi um fator determinante para a popularização dos produtos de automação residencial.

Parte da solução implementada no desenvolvimento desse projeto envolve o uso de uma das famosas redes sociais, o *Twitter*. Esta plataforma possui um website onde seus usuários podem postar mensagens com o limite máximo de 140 caracteres.

O *Twitter* possui uma API liberada para desenvolvedores que possibilita a integração dessa rede social com outros aplicativos. Portanto devido ao seu grande potencial para desenvolvedores, por ser uma ferramenta atual que está sendo muito utilizada em soluções inovadoras de tecnologia e por apresentar uma área de suporte muito eficiente esta rede social foi escolhida para fazer parte da solução proposta para este projeto.

1.1 OBJETIVO

O objetivo desse projeto foi elaborar uma arquitetura para um sistema de automação residencial, que permita ao usuário acionar seus dispositivos a distancia. Esse sistema utiliza uma *Raspberry Pi* e um *Arduino Uno* para o processamento central de dados. Foi elaborada então toda a configuração para os dispositivos que serão controlados, o que envolve a configuração da *Raspberry Pi* e toda a estrutura para que sinais de radiofrequência possam ser transmitidos e recebidos pelo *Arduíno Uno*. Foi necessário também desenvolver o aplicativo para IOS que será a interface entre o usuário e os dispositivos controlados. Foi desenvolvido um protocolo de comunicação entre o smartphone e a *Raspberry Pi*, pois o objetivo é que toda informação flua via internet, para que o usuário tenha a liberdade de acionar seus dispositivos de qualquer lugar.

O diagrama da figura 1 descreve todo o fluxo de informações na arquitetura implementada. O usuário pode enviar seu comando através do aplicativo desenvolvido para o *Iphone* ou postando diretamente no *Twitter* de qualquer computador. Esse comando então é publicado no *Twitter* em forma de uma mensagem. A *Raspberry Pi* irá monitorar o *Twitter* e a cada nova mensagem postada, ela irá compará-la com as mensagens que já estão pré-programadas em sua memória. Ao identificar uma dessas mensagens, a *Raspberry Pi* irá transmitir um comando para o *Arduino Uno* através do transmissor RF. O *Arduino Uno* estará ligado a um receptor RF e ao receber essa mensagem irá então acionar o relé desejado, consequentemente acionando o dispositivo conectado a esse relé.

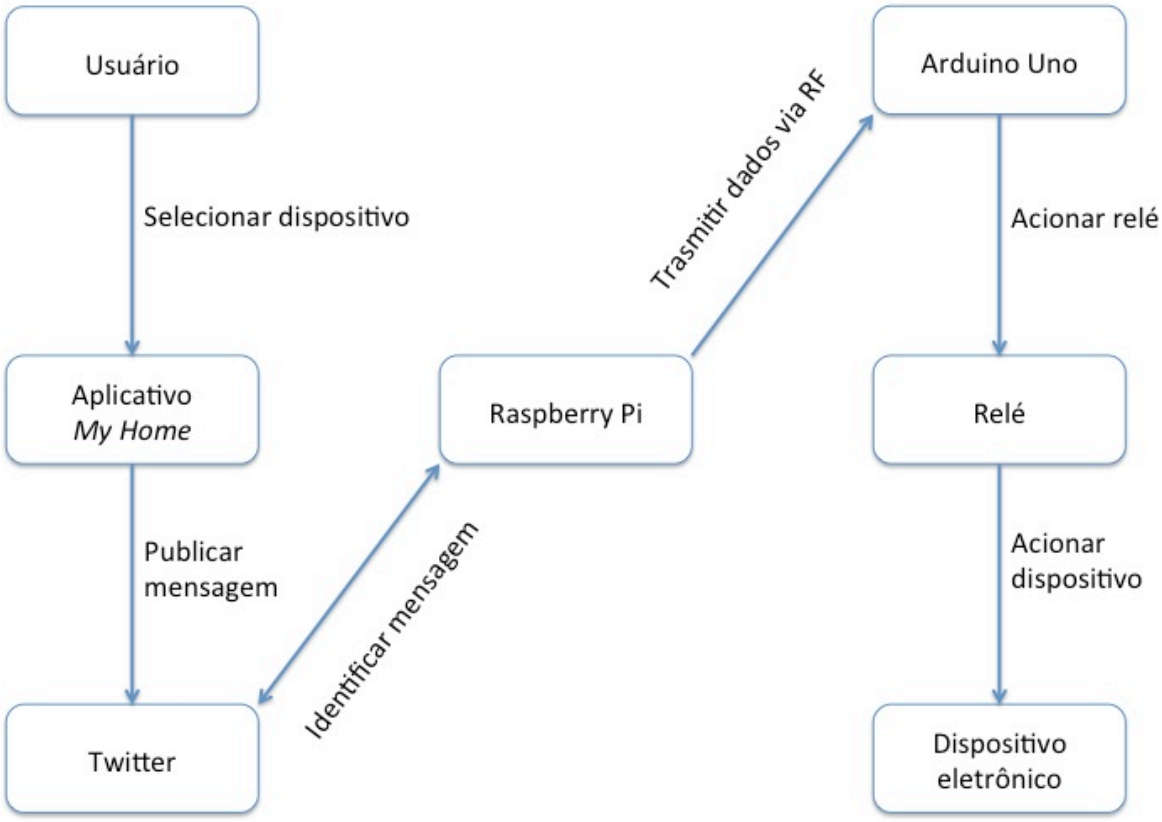

Figura 1 – Diagrama com o fluxo de informações

1.2 JUSTIFICATIVAS/RELEVÂNCIA

Uma vez aperfeiçoado, o sistema desenvolvido poderá contribuir para a modernização de uma residência com baixos investimentos. Essa modernização poderá trazer diversos benefícios para os usuários, como, por exemplo:

- ✓ Redução de gastos desnecessários com energia elétrica, uma vez que os equipamentos poderão ser desligados mesmo que o usuário não esteja na residência.
- ✓ Controle à distância de eletrodomésticos, permitindo o funcionamento desejado sem a presença física do usuário.

Além disso, os conhecimentos adquiridos pela elaboração do projeto, tanto em sistemas embarcados como em programação, podem ser aplicados em diversos

outros projetos de engenharia. Já que os conceitos podem ser aplicados a projetos com diversas finalidades.

2 EMBASAMENTO TEÓRICO

2.1 SISTEMAS EMBARCADOS

Um sistema embarcado tem a função de implementar uma capacidade computacional dentro de um circuito integrado por exemplo. Esses sistemas são completos, independentes e possuem todo o seu hardware e *software* dedicados a realizar tarefas especificas.

Os sistemas computacionais embarcados possuem um baixo custo tecnológico e por essa característica estão cada vez mais presentes no dia a dia da população em dispositivos como celulares, sistemas de controle de automóveis e eletrodomésticos, por exemplo. Os sistemas embarcados então devem apresentar portabilidade, devem ser pequenos, leves e apresentar baixo consumo de energia sem ter seu desempenho reduzido.

Um outro fator que difere esse sistema de sistemas convencionas é o fato de os sistemas embarcados apresentarem uma limitação no armazenamento de *software*s e dados, pois eles são armazenados em memórias FLASH ou ROM e não em discos rígidos como nos computadores convencionais.

Existem diversas arquiteturas para sistemas embarcados, cada uma com suas características que definem para quais projetos elas são melhores aplicáveis. Arquiteturas de 8 bits são mais utilizadas para atividades mais simples, como controle de eletrodomésticos como um liquidificador. Um exemplo de micro controlador com arquitetura de 8 bits é o 8051. Para atividades mais complexas como aplicações em indústrias, por exemplo, a arquitetura de 16 bits é mais indicada pois possui uma capacidade de processamento de dados mais elevada, como exemplo pode-se citar o MSP430. Existe ainda a arquitetura de 32 bits que possui quase o mesmo nível de processamento de dados que os computadores pessoais (OTONI, 2013).

2.2 COMUNICAÇÃO VIA RADIOFREQUÊNCIA

O rádio é um recurso tecnológico de telecomunicações utilizado para permitir a transmissão e recepção de dados pela transcepção de informações previamente codificadas em um sinal eletromagnético. São necessários três elementos para que haja a comunicação: o transmissor, o meio de transmissão e o receptor.

O transmissor converte sinais digitais, analógicos ou sonoros em ondas eletromagnéticas e os envia para o espaço através de uma antena para serem recebidas pelo receptor. O receptor tem a função de decodificar os sinais eletromagnéticos captados pela antena e então transforma-los em ondas digitais ou analógicas ou até mesmo em ondas sonoras.

Existe mais de um modo de transmissão utilizado na comunicação via radiofrequência. O modo simplex é utilizado quando a comunicação é unidirecional no meio da transmissão. Já o modo de transmissão semi duplex é utilizado quando a transmissão é bidirecional, porém executada de maneira alternada em cada sentido. Por fim, o modo duplex é aquele em que a comunicação é bidirecional e simultânea.

O sistema de transmissão também apresenta diferentes classificações. São elas o sistema direcional, que privilegia um destinatário em detrimento de outros, e omnidirecional, que tem como filosofia distribuir o sinal pelo maior número de usuários.

Os projetos que utilizem comunicação via radiofrequência devem ser devidamente planejados, pois a qualidade de recepção dos dados pode ser comprometida. Como exemplo, pode ocorrer uma atenuação do sinal devido ao percurso por longas distâncias, o que resultará em sua “deformação”. Além disso, interferência de outros sinais pode modificar o sinal original, comprometendo a transmissão.

A modulação de sinal pode ser também de dois tipos, designados como AM e FM. Na modulação por amplitude, conhecida como AM, o comprimento da onda é da ordem de metros. Então, são usadas as camadas atmosféricas para a propagação por reflexão. Por outro lado, na modulação por frequência, FM, o comprimento de onda é bem inferior, da ordem de centímetros, o que a torna muito penetrante apesar de pouco refletidas (VIEIRA, 2011).

2.3 PROGRAMAÇÃO ORIENTADA A OBJETOS

Em resumo pode-se afirmar que as linguagens de programação são estruturadas em blocos que possuem um código que realiza uma tarefa, que recebem o nome de função ou procedimento. Uma função corresponde a manipulação de variáveis, que são os elementos utilizados para representar dados. Pode-se concluir que um *software* é então uma sequência de execução de funções atuando sobre funções e consequentemente manipulando dados. Porém essa relação entre o dado e uma função nem sempre é clara.

Em uma linguagem de programação orientada a objetos ocorre uma mudança nesse paradigma. Os *software*s baseados nessas linguagens são modelados com base em objetos e nas relações entre os mesmos. Portanto um *software* desenvolvido nessas bases tende a ser mais intuitivo e real.

Dentro desse modelo de linguagem de programação existem então os objetos que possuem dois conjuntos de elementos que são suas ações e características. As características se referem a atributos ou propriedades. Como exemplo podemos usar um celular como objeto e sua propriedade a cor do mesmo. As ações que um objeto é capaz de fazer são denominadas métodos e usando o exemplo do celular, podemos exemplificar um método como o ato de abrir um aplicativo. Smartphones possuem diversos aplicativos que podem ser considerados objetos, portanto o fato de o celular se comunicar com esses aplicativos exemplifica uma interação entre objetos.

Pode-se definir então um objeto como uma classe instanciada, que nada mais é que um molde preenchido com suas características. É importante ressaltar que para garantir um nível alto de reutilização e otimizar a possibilidade de se realizar mudanças em um projeto, os objetos devem possuir em seu funcionamento um determinado grau de autonomia.

O encapsulamento, exemplificado na figura 2, nesse modelo de linguagem de programação tem por objetivo ocultar características de um objeto que são desnecessárias para seu uso. Pode-se usar aqui o exemplo de um bloco em que se conhece a entrada e a saída porém nem todo o processo que transforma essa entrada em uma saída é conhecido pelo usuário. O processo de encapsulamento facilita o processo de desenvolvimento e reutilização do *software*.

Figura 2 – Encapsulamento

A herança é uma característica presente nas linguagens orientadas a objeto. Pode-se reaproveitar um objeto já criado para gerar um outro objeto, aperfeiçoando e implementando novas funcionalidades ao antigo. Essa característica aumenta a produtividade no desenvolvimento de *softwares*, pois elimina o retrabalho.

Os protocolos são nada mais do que um conjunto de métodos a serem executados por um objeto. Os protocolos em *Objective-C* possuem quase que os mesmos objetivos de outras linguagens de programação mais populares, como por exemplo o Java. Um protocolo então é uma espécie de contrato a ser cumprido por um objeto, onde existem várias ações opcionais e alguma obrigatórias a serem executadas pelo mesmo.

O *Objective-C* foi criado em 1983 na empresa *Stepstone.* Ela nada mais é do que uma “versão“ da linguagem C mas orientada a objetos.

3 MATERIAIS

3.1 RASPBERRY PI®

A *Raspberry Pi* foi lançada com o intuito de incentivar o ensino de sistemas embarcados nas escolas e despertar o interesse dos alunos por este setor (DENNIS, 2013). Para cumprir tal meta, o pequeno computador deveria apresentar um baixo custo e um grande potencial para uso em projetos com diversas finalidades. Como resultado, o *Raspberry Pi* se tornou altamente popular e foi um grande sucesso de vendas. Este minicomputador foi escolhido para a elaboração do projeto devido às inúmeras características, incluindo o custo-benefício.

Segundo Dennis (2013), dois modelos do kit *Raspberry Pi* foram produzidos, sendo denominados modelos A e B. O modelo A não possui conexão Ethernet, mas apresenta consumo consideravelmente menor de energia. O modelo B foi lançado primeiro e é composto por um micro controlador ARM 11 (700MHz), 512 MB de memória RAM e uma GPU dedicada. Porém, a grande vantagem do modelo B consiste na possibilidade de utilização de muitos periféricos, já que o kit apresenta conexão Ethernet, uma saída HDMI, de Áudio, RCA Vídeo, entre outras (Figuras 3 e 4). Assim, foi escolhido como processador de dados neste projeto.

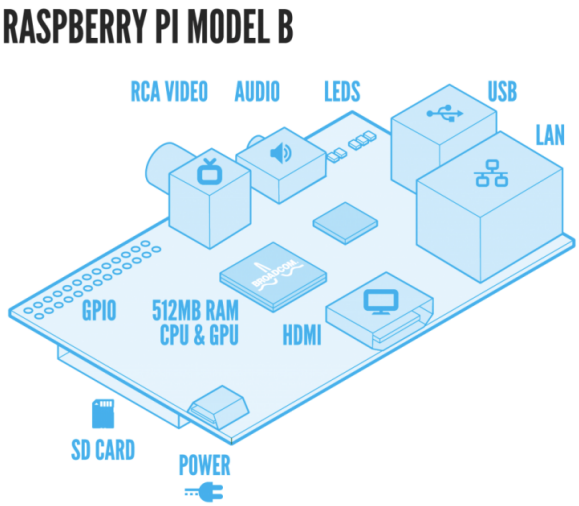

Figura 3 – Entradas e Saídas da *Raspberry Pi* (*RASPBERRY PI* FOUNDATION)

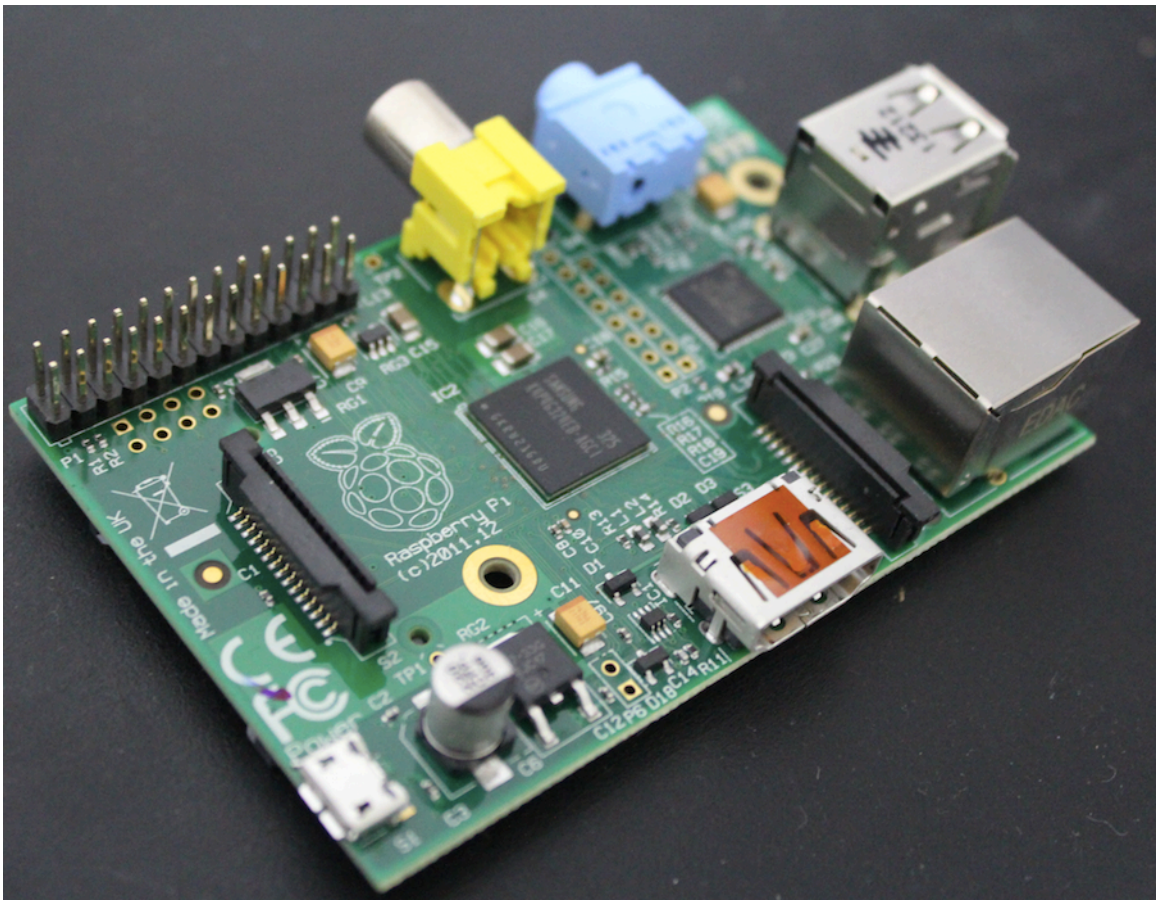

Figura 4 – *Raspberry Pi* modelo B

A alimentação para o modelo B do kit *Raspberry Pi* deve ser de 5 V com uma fonte que forneça até o limite de 700 mA através de uma entrada micro USB. Tais especificações são facilmente encontradas em carregadores de smartphones.

O sistema operacional utilizado na *Raspberry Pi* será o *Raspbian*, que oferece inúmeros pacotes compilados previamente e, consequentemente, muitas opções para os usuários do kit, como centrais de mídia e controle de hardware. O sistema foi escolhido por apresentar um grande suporte de outros usuários, frequentes atualizações para a correção de bugs e pela simplicidade de instalação (GRANA, 2013).

A figura 5 apresenta o esquemático com a numeração dos pinos da *Raspberry Pi*:

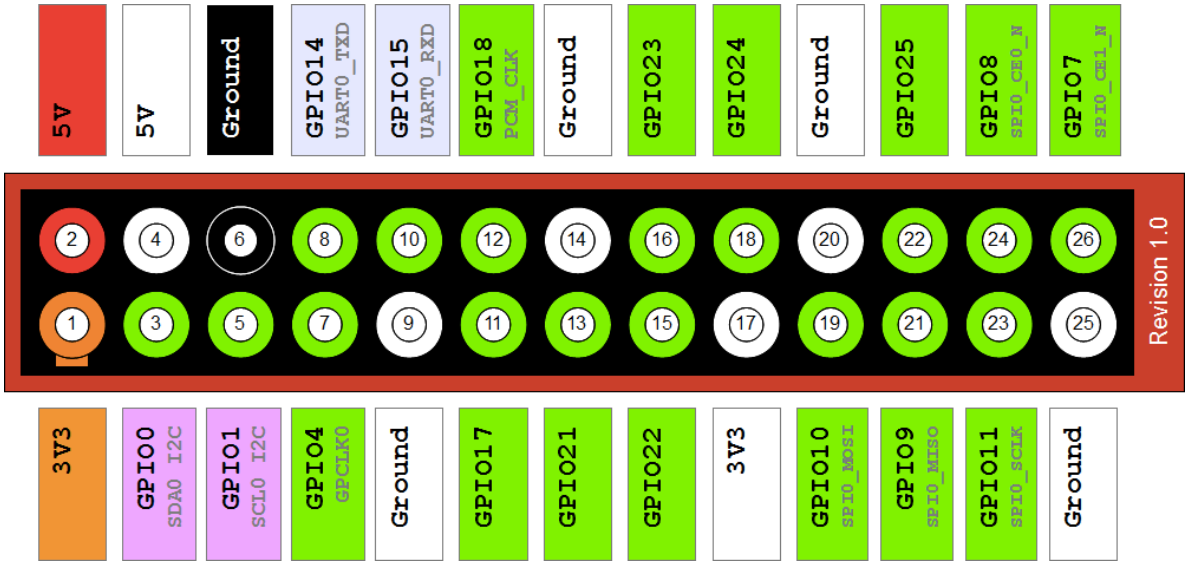

Figura 5 – Numeração dos pinos da *Raspberry Pi*
Fonte: Site http://librehacks.blogspot.com.br/2013/02/eletronica-e-raspberry-pi.html

3.2 MÓDULO RELÉ

O módulo relé utilizado para esse projeto é da marca *Robocore*. Esse módulo foi desenvolvido para ser usado em diversos projetos que visam acionar cargas de até 250VAC @ 7A ou 125VAC @ 10A. Como exemplo pode-se citar cargas como lâmpadas e dispositivos em geral conectados a rede elétrica. O módulo pode ser conectado diretamente a saída digital do *Arduino Uno*, pois o mesmo possui saída padrão de três pinos, VCC, GND e o pino de Sinal Digital. É válido lembrar que, como existe um relé com uma bobina de 5V no módulo, o mesmo deve ser alimentado pela saída de 5V regulada do *Arduino Uno*. É possível observar o estado do relé através do LED que está presente na placa. Pode-se acessar três terminais do relé:

- ✓ Normalmente Aberto (NA);
- ✓ Comum (C);
- ✓ Normalmente Fechado (NF).

O canal comum não se altera, porém os outros dois canais mudam de estado quando o relé recebe um sinal de nível alto no pino de entrada. Quando o mesmo recebe um sinal de nível baixo, os pinos retornam a seu estado original.

Especificações técnicas fornecidas pelo fabricante:

- ✓ Tipo: Digital;
- ✓ Sinal de controle: Nível TTL;
- ✓ Bobina: 5VDC 75mA;
- ✓ Carga nominal do relé: 12A 125VAC, 7A 250VAC;
- ✓ Carga nominal do módulo: 10A;
- ✓ Tempo de acionamento de contato: 10ms.

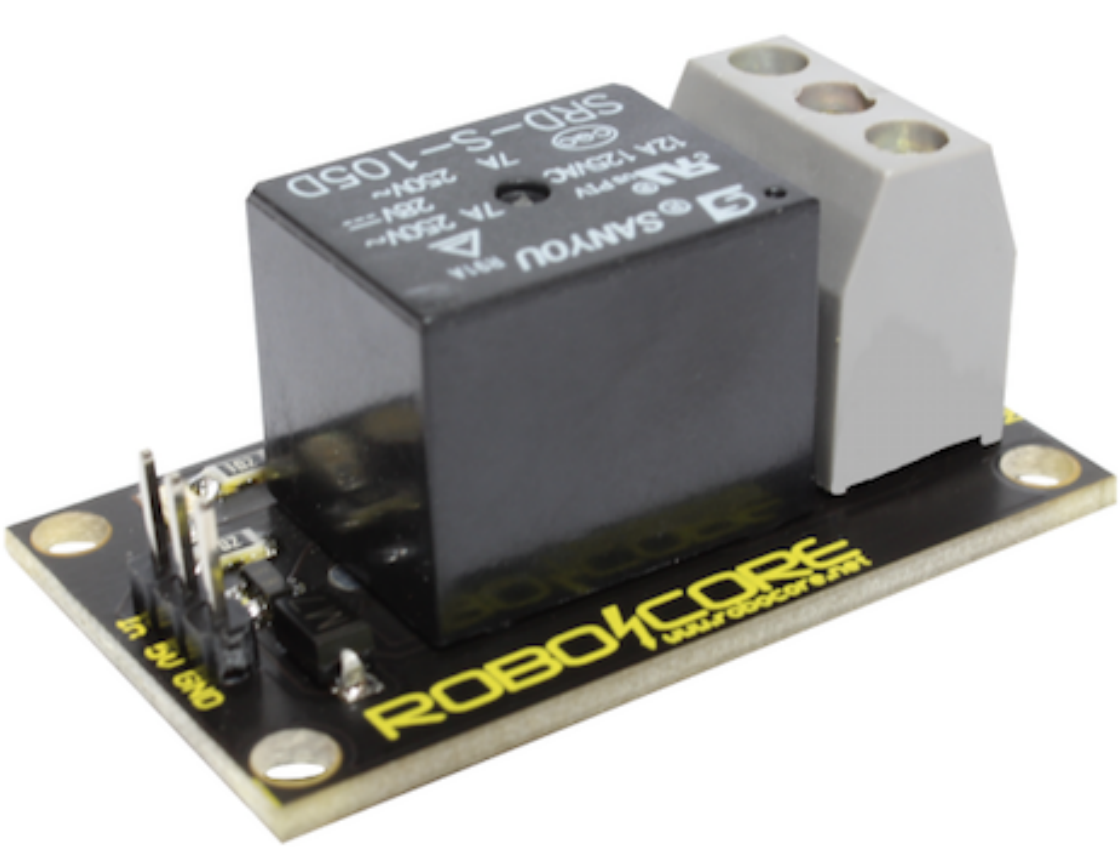

Figura 6 – Relé
Fonte: Site www.robocore.com.br.

3.3 SMARTPHONE (SIMULADOR) E XCODE

Para os testes desse projeto foi usado o simulador presente na ferramenta utilizada, que simula o ambiente do smartphone *Iphone* da empresa *Apple*. Esse ambiente é onde pode-se observar como será o comportamento do aplicativo quando o mesmo estiver no celular. É possível averiguar o espaço que o software ocupara no celular e ainda outros aspectos como o nível de processamento solicitado pelo aplicativo por exemplo.

O uso do simulador é justificado pelo fato de ele ser baseado na plataforma real que é o *Iphone* e por de seu uso ser gratuito, pois para adicionar o aplicativo ao smartphone é necessário adquirir uma conta de desenvolvedor na *Apple* que envolveria um custo e como esse projeto é apenas um protótipo, o simulador se tornou uma opção mais viável.

O *aplicativo* foi desenvolvido para a plataforma *IOS* por meio da ferramenta *Xcode*, fornecida pela própria empresa. A linguagem de programação utilizada pela ferramenta é o *Objective-C*. O aplicativo foi desenvolvido para o IOS 8 para ser compatível com a versão mais atual do sistema operacional.

A figura 7 representa a interface do *Xcode* que é dividida em cinco áreas:

- ✓ ***Toolbar***: onde se encontram todas as ferramentas que podem ser utilizadas no desenvolvimento de um aplicativo;
- ✓ ***Navigator area*****:** onde estão localizados os módulos que compõem o aplicativo;
- ✓ ***Editor area*****:** onde se pode adicionar a declaração de variáveis e métodos, enfim todo o código do *software*;
- ✓ ***Debug area*****:** identificação de erros na compilação;
- ✓ ***Utility area***: onde estão localizados os objetos já existentes e prontos para uso como botões, por exemplo.

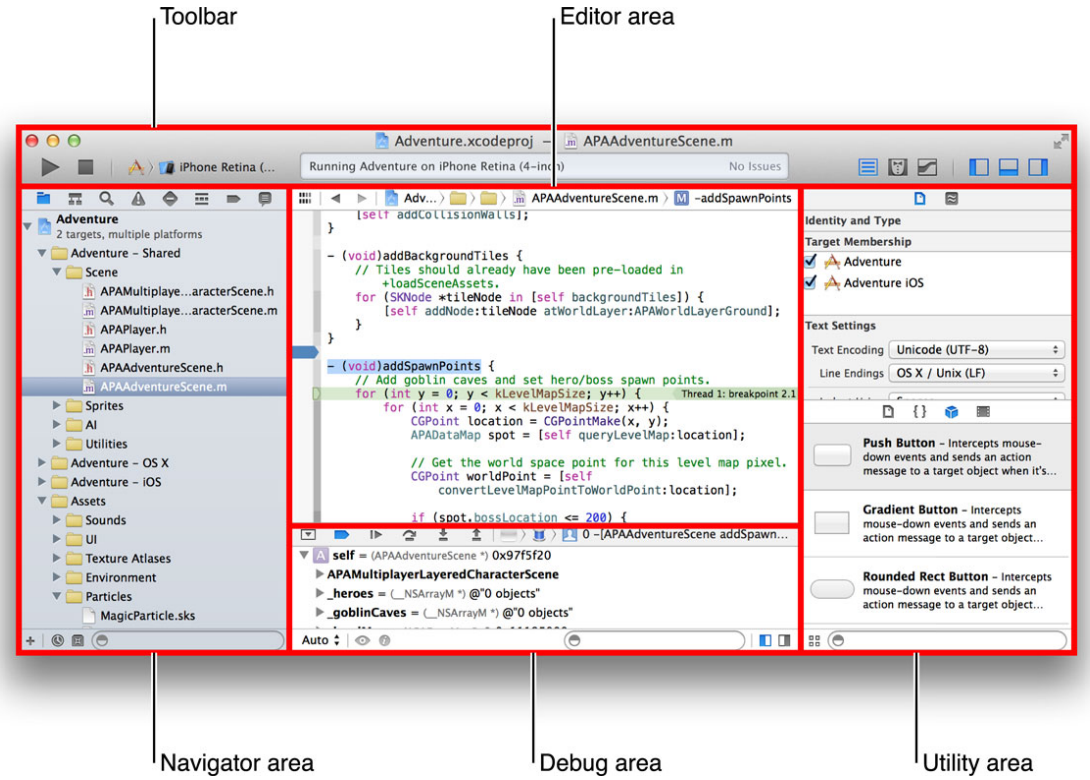

Figura 7 – Interface do *Xcode*
Fonte: Site http://codewithchris.com/xcode-tutorial/

3.4 ARDUINO UNO

O *Arduino Uno* é definido por alguns autores como uma plataforma embarcada de interação com o ambiente por hardware e *software*. Tal atributo permite usar *Arduino Uno* para o controle de inúmeros dispositivos como sensores, LEDS e equipamentos que usam comunicação de radiofrequência por exemplo.

De origem italiana, o *Arduino Uno* surgiu com o intuito de ensinar aos alunos de uma faculdade conceitos de eletrônica e programação. Seu objetivo sempre foi ser simples e de baixo custo, o que facilitou sua inserção no mercado e o transformou em uma das aplicações de eletrônicas mais populares nos dias atuais.

Como o objetivo do *Arduino Uno* está ligado ao ensino, essa plataforma fornece a seu usuário muita liberdade tanto no hardware quanto no *software* para o desenvolvimento.

As especificações técnicas fornecidas pelo fabricante são:

- ✓ Micro controlador: ATmega328;
- ✓ Tensão de operação: 5V;

- ✓ Tensão de alimentação recomenda: 7-12V;
- ✓ Tensão de entrada limites: 6-20V;
- ✓ Entradas/saídas digitais: 14, em que 6 possuem modulação por largura de pulso (PWM);
- ✓ Pinos de entrada analógica: 6;
- ✓ Corrente DC por pino de Entrada e Saída: 40mA;
- ✓ Memória Flash: 32 kB;
- ✓ SRAM: 2kB;
- ✓ EEPROM: 1kB;
- ✓ Frequência de *Clock*: 16MHz.

Como pode-se observar as especificações técnicas, retiradas do site oficial[1], o *Arduino Uno* deve ser usado para aplicações mais simples que não requerem um grau muito alto de processamento, o que condiz com seu objetivo que voltado para o ensino e não para o ambiente profissional.

Pode-se observar na figura 8 o *Arduino Uno*, que é um dos modelos mais utilizados atualmente e o mais abundante no mercado.

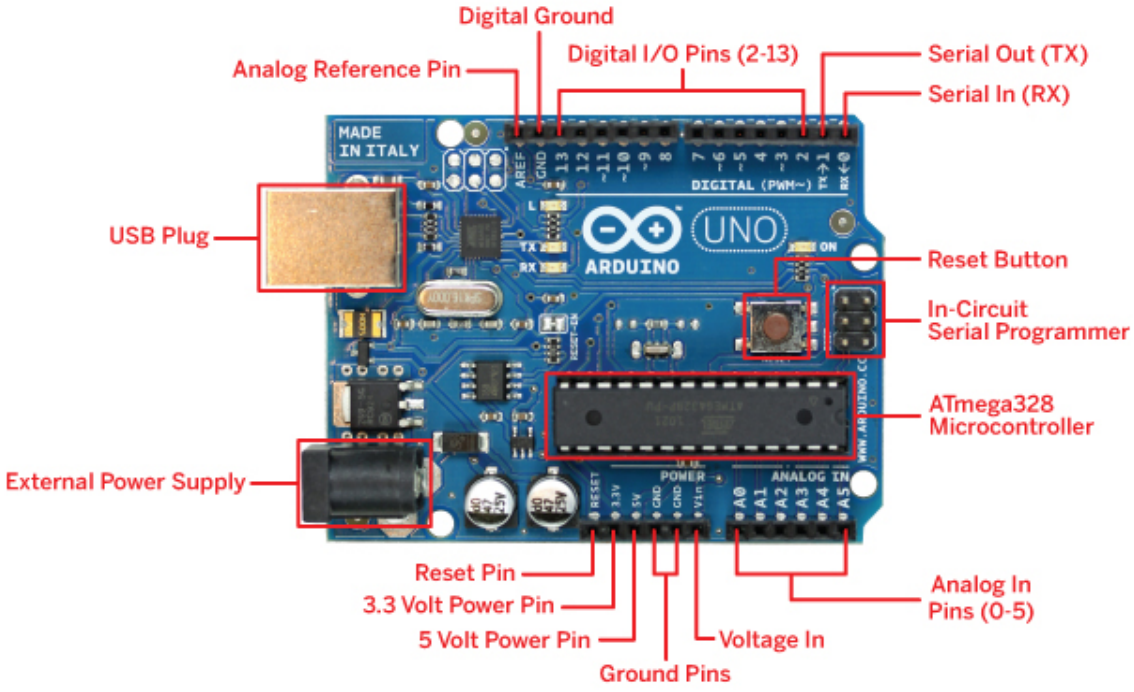

Figura 8 – Arduino
Fonte: Site – http://www.embarcados.com.br/arduino-uno/

[1] Portal *Arduino*. http://www.arduino.cc. Acesso em 29/06/2014.

3.5 TRANSMISSOR E RECEPTOR RF

Foi usado nesse projeto um transmissor e um receptor de 433MHz. A escolha dos modelos foi baseada principalmente no custo, visto que esse projeto é um protótipo e não uma solução comercial.

As especificações técnicas fornecidas pelo vendedor estão listadas abaixo.

Receptor:

- ✓ Frequência: 433,92Mhz;
- ✓ Modelo: MX-05V;
- ✓ Tensão: 5VCC;
- ✓ Corrente: 4mA;
- ✓ Sensibilidade: -105db;
- ✓ Antena: 35cm;
- ✓ Dimensões: 30mm x 17mm x 7mm.

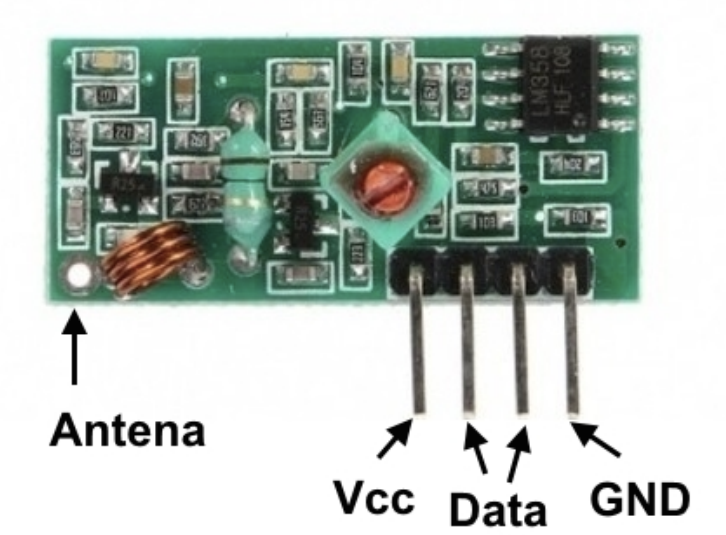

Figura 9 – Receptor RF
Fonte: Site – http://blog.filipeflop.com/wireless/modulo-rf-transmissor-receptor-433mhz-arduino.html

Transmissor:

- ✓ Frequência: 433,92Mhz;
- ✓ Modelo: MX-FS-03V;
- ✓ Distância de transmissão: 20m a 200m (depende da tensão);
- ✓ Tensão: 3,5V ~ 12V;
- ✓ Dimensões: 19mm x 19mm;
- ✓ Tipo de modulação: AM;
- ✓ Potência de transmissão: 10mW;
- ✓ Antena: 25cm;
- ✓ Velocidade de transmissão: 4Kb/s;
- ✓ Pinos: Dados, VCC e GND.

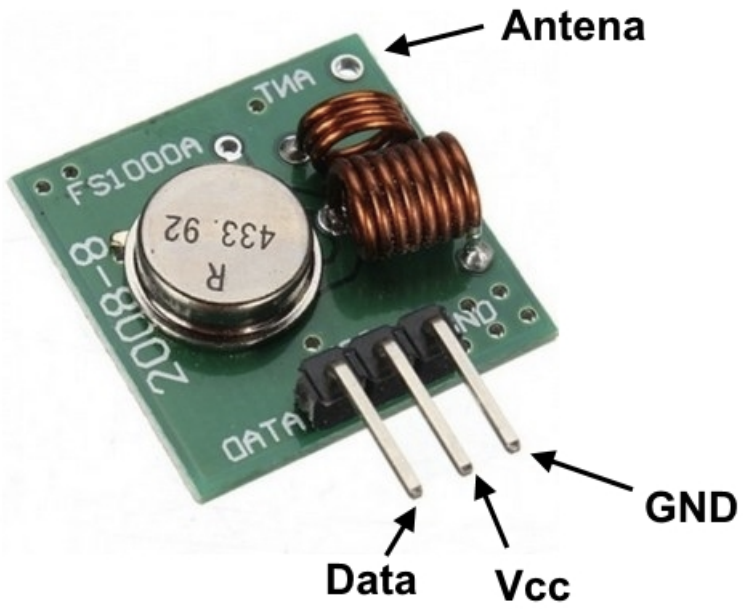

Figura 7 – Transmissor RF
Fonte: Site http://blog.filipeflop.com/wireless/modulo-rf-transmissor-receptor-433mhz-arduino.html)

4 IMPLEMENTAÇÃO E DESENVOLVIMENTO

Neste capítulo serão apresentadas as etapas de desenvolvimento desse projeto.

4.1 CONFIGURAÇÃO DO ARDUINO

O desenvolvimento desse projeto teve início com a obtenção dos códigos que serão transmitidos pela combinação da *Raspberry Pi* com o transmissor RF. Para esta etapa foi usado o *Arduino Uno*, por apresentar uma interface extremamente simples e para que fosse possível se familiarizar com essa plataforma que é uma das mais usadas atualmente entre estudantes.

Inicialmente realizaram-se as conexões necessárias entre o receptor de radiofrequência e o *Arduino Uno*.

Como pode-se observar na figura 11, o pino VCC do receptor foi ligado no pino VCC do *Arduino Uno*, que o forneceu uma alimentação de 5V, o pino de DATA do receptor foi conectado ao pino 2 do *Arduino Uno*, que foi o pino escolhido para a transmissão de dados. Por fim o pino GND do receptor foi ligado no GND do *Arduino Uno*, fechando assim o circuito.

O relé também recebeu a alimentação e o GND do *Arduino Uno*, e para seu controle o mesmo foi ligado no pino 5 do *Arduino Uno*.

O programa desenvolvido para o *Arduino Uno* se chama *tccArduino*. Sua operação consiste em identificar o código recebido pelo receptor de radiofrequência e comparar com os as informações já cadastradas no programa. Portanto se a informação recebida coincidir com a existente em alguma das estruturas de comparação que o programa possui, o software então ativa ou desativa o respectivo pino do *Arduino Uno*, mudando o estado do relé que está conectado ao mesmo.

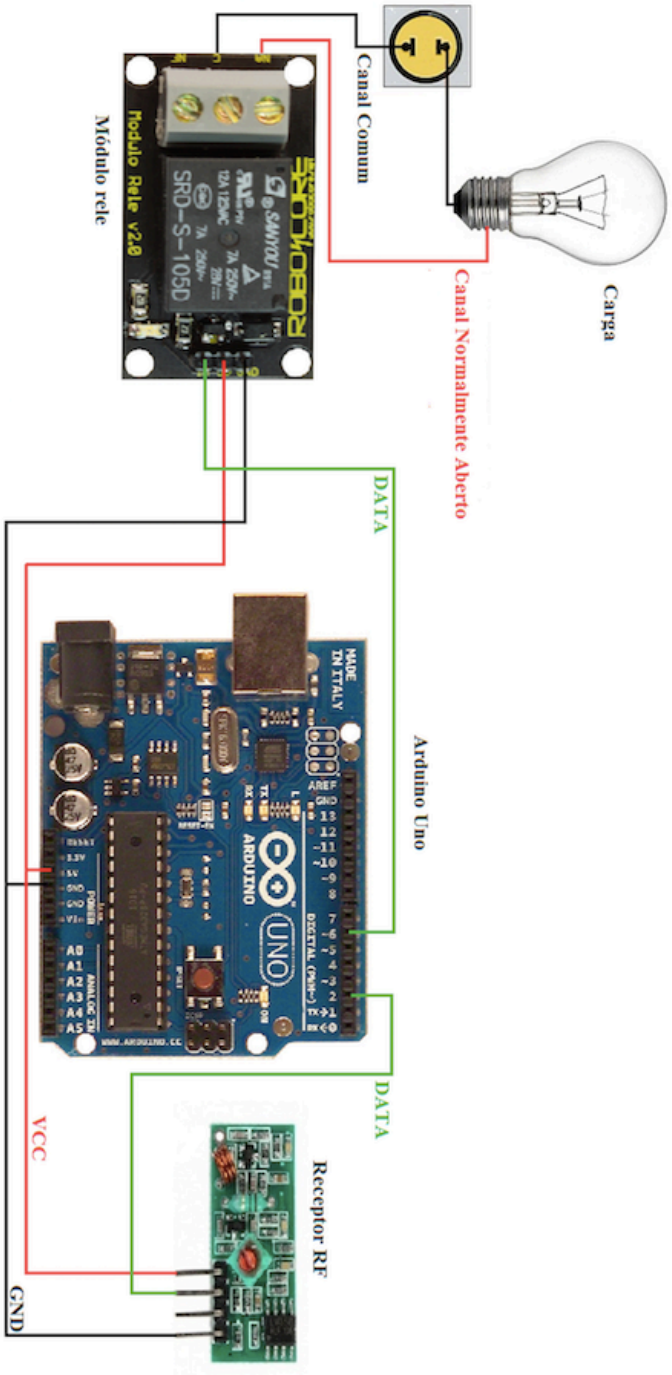

Figura 8 – Esquemático entre *Arduino Uno*, receptor RF e Relé

4.2 CONFIGURAÇÃO DA CONTA DO *TWITTER*

É necessário criar uma conta no site[2]. Ela deve ser configurada para que o usuário possa usufruir das funcionalidades liberadas para desenvolvedores. O próximo passo é então criar um aplicativo que se relacione a essa conta. Esse aplicativo fornecerá um Token que permitirá o acesso a essa conta. Para criar esse aplicativo basta apenas fornecer um nome para o mesmo, uma descrição e um website caso exista algum como segue o exemplo da figura 12:

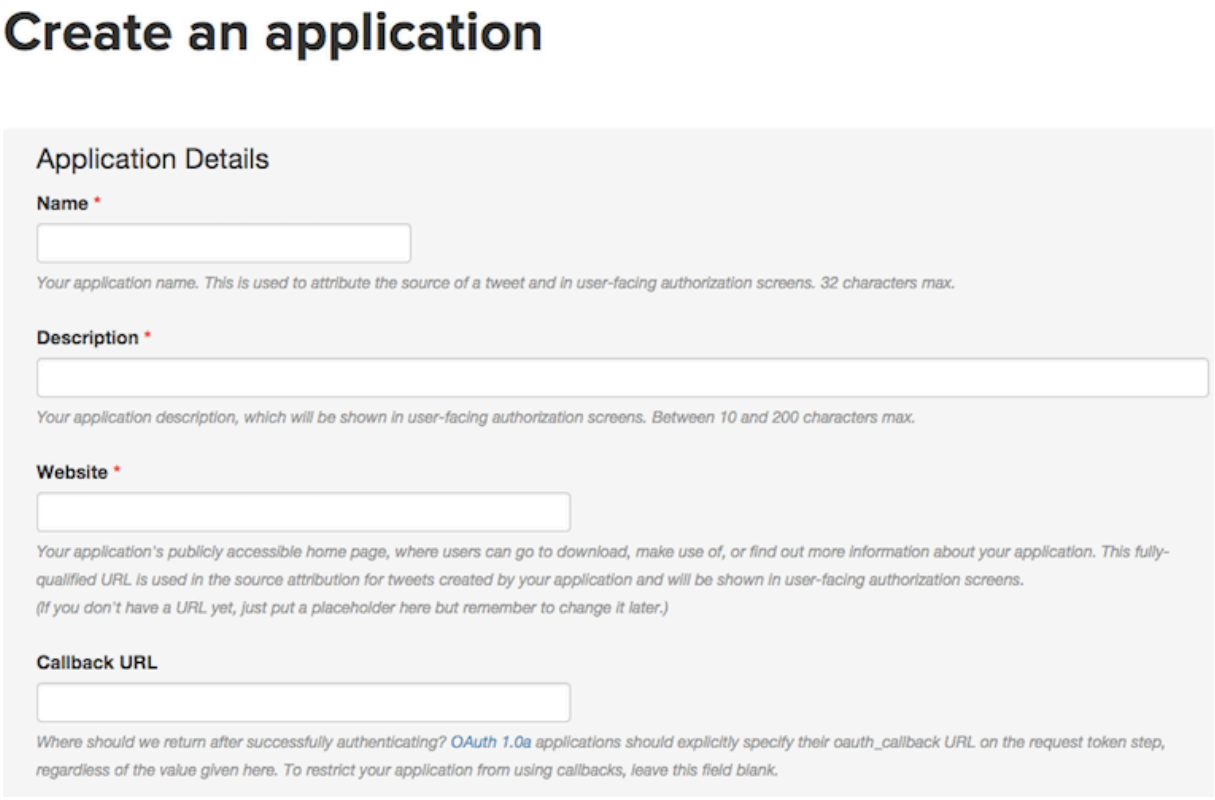

Figura 9 – Tela 1 para configuração da conta do *Twitter*

[2] Portal *Twitter*. Disponível em: www.twitter.com. Acessado em 28/08/2014.

Em seguida deve-se gerar o Token que fornecerá o acesso a essa conta do *Twitter*. Esse Token será inserido na configuração da *Raspberry Pi* para que a mesma possa ter acesso a conta e monitorar o que será postado no *Twitter*. Por fim basta alterar a configuração da conta para que outras aplicações tenham permissão de escrita, leitura e de acesso a mensagens diretamente, como segue figura 13:

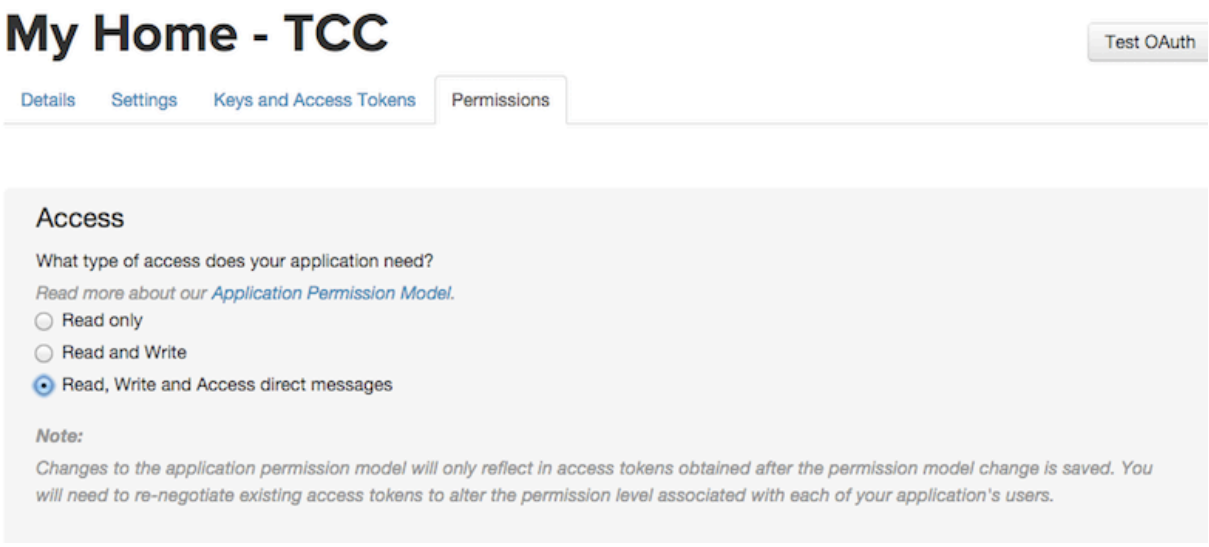

Figura 10 – Tela de permissão de acesso a conta do *Twitter*

4.3 RASPBERRY PI

4.3.1 Configuração da Raspberry Pi para comunicação via RF

A primeira etapa foi conectar a *Raspberry Pi* ao transmissor RF. O transmissor recebeu da *Raspberry Pi* os 5V necessários para sua alimentação e o GND. Os dados foram transmitidos através do pino 11(GPIO17) da *Raspberry Pi* como pode-se observar na figura 14:

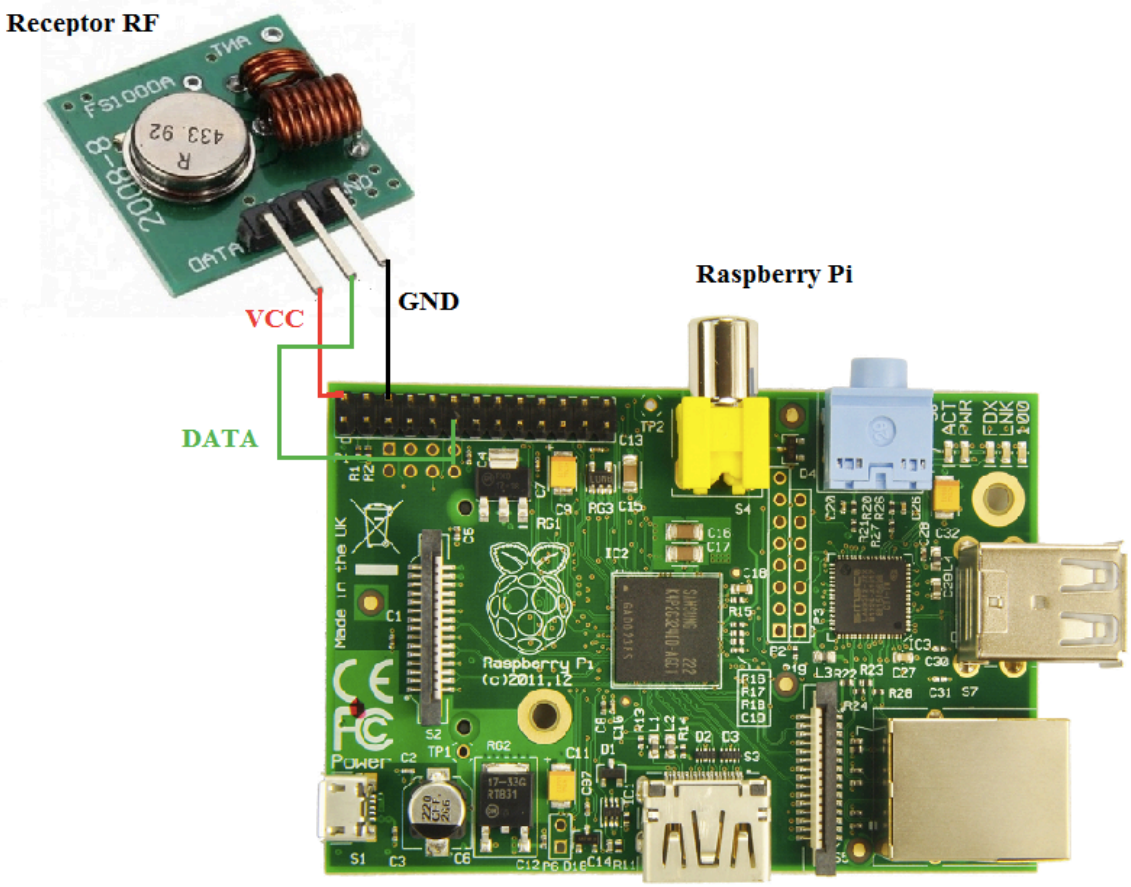

Figura 11 – *Raspberry Pi* e Receptor RF

Para configurar a *Raspberry Pi* inicialmente foram instaladas as bibliotecas wiringPi[3], que permite ao usuário gerenciar com facilidade os pinos de entrada e saída da *Raspberry Pi* e a biblioteca RCSwitch[4] que fornece métodos que auxiliam no envio dos códigos.

[3] Portal *GitHub*. Disponível em: www.github.com/WiringPi. Acessado em 13/08/2014.
[4] Portal *GitHub*. Disponível em: www.github.com/r10r/rcswitch-pi. Acessado em 14/08/2014

Foi adaptado então um programa que utiliza os métodos dessas duas bibliotecas para que a informação desejada chegue até o pino 11 da *Raspberry Pi* onde está conectado o transmissor RF. O nome do programa é *codesend.cpp*. O programa utiliza C++ pois é a linguagem em que estão embasadas as bibliotecas.

O programa inicialmente armazena na variável "*PIN*" o pino em que será transmitida a informação e na variável *"code"* a mensagem, em decimal, que é necessário ser enviada para o *Arduino Uno*. Para que esse programa seja executado deve-se digitar no terminal da *Raspberry Pi* o comando "*sudo/codesend*" seguido do código(decimal) que deseja que seja enviado pelo transmissor RF. O programa então responde a essa ação mostrando a mensagem "*sending code* [12345]" e então executa ação de transmitir essa mensagem, como pode-se observar no código:

```
pi@raspberrypi ~ $ cd 433Utils/RPi_utils
pi@raspberrypi ~/433Utils/RPi_utils $ sudo ./codesend 12345
sending code[12345]
```

A tabela 1 relaciona as ações que ocorrem quando os códigos específicos são transmitidos.

Tabela 1 – Relação entre códigos e os relés

	Mensagem monitorada pela *Raspberry Pi*	Código enviado pelo Transmissor RF
Relé 1	Ligar R1	7689219
	Desligar R1	7689228
Relé 2	Ligar R2	7689264
	Desligar R2	7689408

4.3.2 Configuração da Raspberry Pi para monitorar o Twitter e enviar os códigos desejados.

Para esta etapa inicialmente se instalou a biblioteca *Twython* na *Raspberry Pi,* que fornece métodos que podem ser usados para acessar as funcionalidades do *Twitter*.

O programa desenvolvido para esta etapa se chama ”*tccv1.py*” e utiliza a linguagem de programação *python,* devido ao fato de biblioteca *Twython* ser desenvolvida nessa linguagem.

Esse programa então executa duas funções, a primeira é monitorar no *Twitter* o conteúdo definido no código. Inicialmente então é necessário definir qual conteúdo irá monitorar, para este caso foi definido que a *Raspberry Pi* deverá monitorar os termos “Ligar R1“, “Ligar R2“, “Desligar R1“ e “Desligar R2”. Esses termos foram salvos na *string “TERMOS*“ como pode-se observar no trecho do programa tccv1.py:

```
# Termos que serão monitorados
TERMOS = 'Ligar R1, Ligar R2, Desligar R1, Desligar R2'
```

Em seguida é necessário fornecer todas chaves de segurança para que o *software* possa ter acesso ao conteúdo da conta do *Twitter*. Essas chaves são geradas pelo aplicativo que foi criado na conta do *Twitter* e devem ser salvos nas respectivas variáveis.

```
# Chaves para o acesso ao twitter
APP_KEY = 'j7kYAYgsd4dlqRUQK6Xrv81rU'
APP_SECRET = 'Wie2l9Pc3hOuVBfji07YyH5vW0EQdi2cUPkfvi5OkZmnlmTkIL'
OAUTH_TOKEN = '2837657895-oMqK5BGzAq4pbrk95iBOZHyjQRKUlZDvNd3o1OD'
OAUTH_TOKEN_SECRET = 'v4Y7pnl5ELe1VKgHMF0xafsF5jmxpHcc3OpQZBWxiOjkl'
```

O *tccv1.py* possui a classe *BlinkyStreamer* que inicia o monitoramento do conteúdo do *Twitter*. Quando algum dos termos é postado no *Twitter* o teccv1.py detecta e o imprime no terminal da *Raspberry Pi,* em seguida é enviado também para o terminal o comando que aciona o programa *codesend.cpp* descrito anteriormente.

Como exemplo, inicialmente devemos acessar a pasta que contém o programa *tccv1.py* e em seguida executa-lo para que comece o monitoramento. Deve-se então entrar com os seguintes comandos no terminal da *Raspberry Pi*:

```
pi@raspberrypi ~ $ cd 433Utils/RPi_utils
pi@raspberrypi ~/433Utils/RPi_utils $ sudo python tccv1.py
```

O *Twitter* então já estará sendo monitorado pela *Raspberry Pi*. É necessário então postar no *Twitter* algum dos termos definidos no código do *tccv1.py*.

O tccv1.py identifica que dentro da mensagem, contém o termo “Ligar R1“ que é uma das mensagens monitoradas e a imprime no terminal da *Raspberry Pi*, em executa o comando ”*sudo ./codesend 7689219*” que aciona o programa codesend.cpp enviando o sinal para *Arduino Uno* acionar o relé correspondente.

```
pi@raspberrypi ~ $ cd 433Utils/RPi_utils
pi@raspberrypi ~/433Utils/RPi_utils $ sudo python tccv1.py
Ligar R1

sending code[7689219]
```

4.4 DESENVOLVIMENTO DO APLICATIVO MY HOME

A última etapa do projeto foi o desenvolvimento de um aplicativo para *Iphone*. A plataforma utiliza é o *Xcode* e a linguagem de programação é o *Objective-C*. O *software* possui 2 telas e recebeu o nome de *My Home.* Para cada tela criada no aplicativo, são gerados dois arquivos, um com extensão “*m*” e outro com extensão “*h*”. A figura 15 representa como esses arquivos estão organizados no aplicativo desenvolvido.

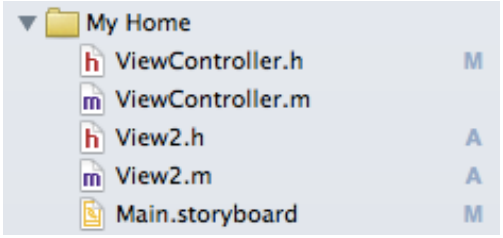

Figura 15 – Módulos do My Home

Os arquivos “*ViewController.h*“ e “*ViewController.m“* são referentes a primeira tela do aplicativo, e os outros arquivos que se encontram na figura 15 são referentes a primeira e segunda telas respectivamente. O arquivo de extensão “*h*” é onde deve ser declarado todos os objetos que serão usados na respectiva tela. O arquivo de extensão “*m*” é onde se atribui os métodos aos objetos e toda a lógica do programa.

Figura 16 – Tela 1 e 2 do aplicativo *My Home*

A primeira tela do aplicativo identificada na figura 16 é de abertura. Essa tela fornece ao usuário o botão "*My Home*", que o leva a segunda tela com as demais funcionalidades do *software*.

A figura 16 contém também a segunda tela do aplicativo, que é referenciada pelos arquivos *View2.h* e *View2.m* Nela encontram-se os botões utilizados para ligar e desligar cada um dos relés.

Cada um dos relés possui um botão para ligar e um diferente para desligar. A lógica por trás desses botões é exatamente a mesma, o que os diferencia é a mensagem que é pré-carregada em cada um deles. Por exemplo, ao se apertar o botão "ON" do Relé 1 e em seguida o "OFF" do Relé 1, as seguintes telas aparecerão conforme a figura 17:

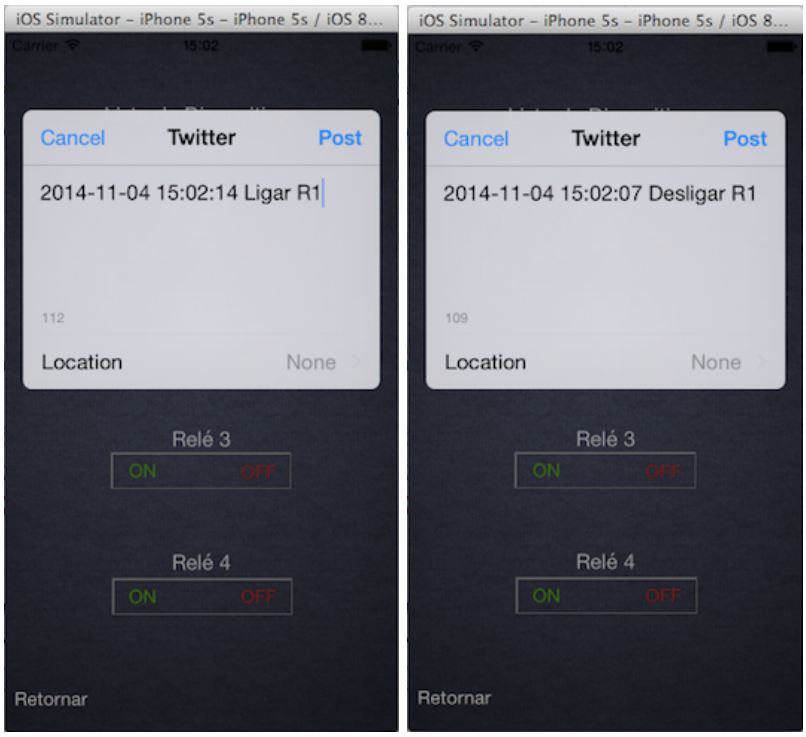

Figura 17 - Mensagem que aparece ao se apertar os botões ON e OFF respectivamente

Nesta etapa, se o usuário clicar em "*Post*" ele estará confirmando a ação e publicando essa mensagem no *Twitter*. Essa mensagem é pré-carregada no arquivo View2.h na *string* "mensagem" e é apresentada na tela do *Twitter* através do método "*setInitialText*" como pode-se observar abaixo:

```
NSString *mensagem = [NSString stringWithFormat:@"%@ Ligar R1", dateString];
[tweetSheet setInitialText:mensagem];
```

Ao se clicar nos botões "*ON*" ou "*OFF*" o aplicativo verifica se existe uma conexão com o *Twitter* disponível. Se o *Iphone* estiver sem conexão com a internet ou sem uma conta do *Twitter* cadastrada essa verificação retornará uma mensagem de erro, que é gerada pelo código:

```
UIAlertView *alertView = [[UIAlertView alloc]
initWithTitle:@"Ops"
message:@"Erro devido à falta de conexão à internet ou não há uma conta do Twitter cadastrada"
```

O *Twitter* possui uma regra que impede o usuário de publicar a mesma mensagem dentro de um período de tempo inferior a 60 minutos. Essa regra

impediria então o usuário de ligar e desligar o mesmo dispositivo mais de uma vez dentro de uma hora. Portanto como solução para esse problema foi inserido na mensagem pré-carregada de cada botão uma *string* que contém a data e a hora atual, dessa maneira as mensagens nunca serão iguais, como pode-se observar na figura 17.

A segunda tela possui ainda o botão “Retornar“ que retorna para a primeira tela. Os botões referentes aos relés 3 e 4, estão desativados pois ainda não existem tais relés conectados ao sistema.

O diagrama de fluxo de dados do aplicativo está explicitado na figura 18 e indica quais são as entradas e saídas do *software*.

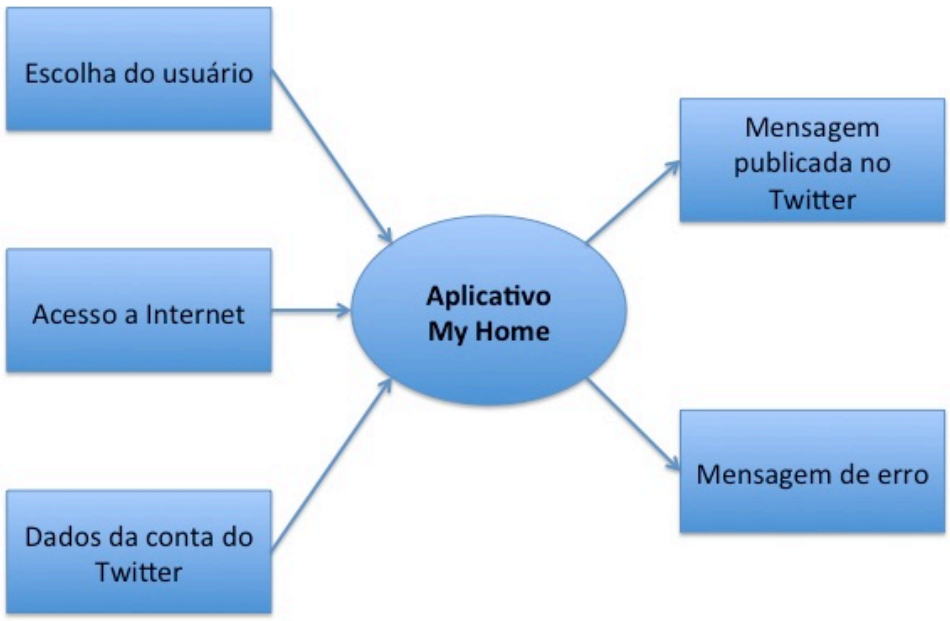

Figura 18 – Diagrama do aplicativo *My Home*

O *software* recebe de fontes externas três informações. O acesso à internet e os dados para acesso a conta do *Twitter* são fornecidos pelo próprio celular. Após receber esses dados, o *software* realiza a validação desses dois acessos. A informação do usuário é utilizada para definir as decisões que o *software* irá tomar, como por exemplo qual tela ele irá abrir ou qual mensagem deve ser postada no

Twitter. O *software* então poderá responder com duas saídas após processar todos os dados. A mensagem poderá ser postada no *Twitter*, ou caso alguma das duas validações não seja confirmada, o aplicativo mostrara uma mensagem de erro.

5 RESULTADOS E DISCUSSÕES

A partir desse projeto pode-se perceber que o tema Automação Residencial é muito discutido atualmente. Existem diversas soluções propostas para esse âmbito, a ideia desse projeto foi usar elementos que são fáceis de se obter para elaborar uma solução própria. Os materiais desse projeto apresentaram algumas dificuldades para sua implementação que serão discutidas a seguir.

O *Arduino Uno* e a *Raspberry Pi* como já mencionado anteriormente se tornaram plataformas extremante populares, portanto é possível achar vários fóruns de discussões sobre soluções para essas plataformas, existem sites especializados nesse tema, enfim existe uma gama muito grande de suporte para esses dispositivos. O resultado de tamanha popularidade é que surgiram varias ferramentas para essas plataformas que estão disponíveis para serem usadas. A dificuldade associada a esse fato é que nem toda a informação existente sobre essas plataformas está organizada e nem todas as ferramentas existentes realmente funcionam ou estão adaptadas para as necessidades especificas de um projeto. Portanto para elaborar este projeto foi necessário realizar o trabalho de um engenheiro que é, primeiramente definir quais são as necessidades do projeto, pesquisar as ferramentas já existentes, analisar cada uma delas e separar as que podem ser adaptadas para suprir as necessidades listadas e por fim escolher a mais simples e eficaz de ser implementada. Portanto para o desenvolvimento dos *softwares* desse projeto*,* foram selecionadas bibliotecas que possuíam métodos que auxiliaram a resolver as necessidades do projeto. No caso do *Arduino Uno*, a biblioteca *RCSwitch*[5] fornece o método *getReceivedValue ()* que permite interpretar a mensagem recebida pelo receptor RF, quando implementada com a lógica correta. A *Raspberry Pi* utilizou a mesma biblioteca *RCSwitch*, que fornecia ao transmissor RF a mensagem que era desejada transmitir, também quando implementada com a lógica correta. Portanto, essas ferramentas se mostraram extremamente eficientes para permitir a comunicação entre o *Arduino Uno* e a *Raspberry Pi.*

[5]PORTAL GitHub. Disponível em: https://github.com/ninjablocks/433Utils/tree/master/RPi_utils. Acesso 28/08/2014.

A *Raspberry Pi* ainda contou com uma outra biblioteca, a *Twython*[6], que fornece métodos para que a mesma possa se comunicar com o *Twitter*. Essa ferramenta se mostrou um pouco mais difícil de se implementar devido a lógica necessária e ao fato de ser necessário utilizar a linguagem *python*. Porém, com muito estudo, foi possível aproveitar uma das funcionalidades dessa ferramenta que é monitorar as palavras desejadas no *Twitter*.

O *Twitter* foi escolhido por se mostrar uma ferramenta mais simples de se configurar em relação a outras pesquisadas como o No-IP, que requer uma série de configurações para que se possa transmitir mensagens através da internet. O *Twitter* por possuir todo um suporte para desenvolvedores acabou se tornando uma opção prática, simples e eficaz para simples funcionalidades como a desejada nesse projeto.

O aplicativo desenvolvido para a plataforma IOS se mostrou muito desafiador por apresentar uma linguagem de programação não muito popular e pelo fato de a plataforma sofrer constantes atualizações. Primeiramente é preciso se familiarizar com a plataforma de desenvolvimento, o *Xcode*, o que não foi uma tarefa simples, pois ela possui inúmeras funcionalidades diferentes, o que dificulta quando se está procurando uma específica. Após a familiarização com o ambiente de desenvolvimento então, é necessário conhecer a linguagem Objetive-C. Para esta etapa então foi desenvolvido um aplicativo simples seguindo tutoriais encontrados na internet que imprimia a mensagem *Hello World* na tela do simulador quando se apertava um botão. Esse primeiro aplicativo permitiu então a familiarização com funções básicas da linguagem, como declarar variáveis, *labels,* objetos e como atribuir métodos a um objeto. Após a familiarização com a plataforma e com a linguagem de programação pôde-se desenvolver o aplicativo final desse projeto. O principal objetivo desse aplicativo é tornar todo o processo de ativar ou desativar os relés o mais simples possível, e esse objetivo de fato foi alcançado, pois para realizar essa ação o usuário precisa apenas apertar dois botões na tela do celular, o botão "*ON*" ou "*OFF*" referentes ao relé desejado e o botão "*Post*" para confirmar sua ação. O usuário poderia também controlar os relés diretamente do site do *Twitter*, postando lá as mensagens especificas, porém para isso ele precisaria acessar o site do *Twitter*, entrar os dados da conta, digitar a mensagem e publica-la. O usuário

[6]PORTAL GITHUB. Disponível em: https://github.com/ninjablocks/433Utils/tree/master/RPi_utils.

teria ainda que se certificar de que a mensagem que será publicada está exatamente igual a mensagem que é monitorada pela *Raspberry Pi,* caso contrário não ocorrerá o controle dos relés. Portanto com o aplicativo todo o processo é simplificado para o usuário.

O uso do transmissor e receptor RF para a comunicação entre o *Arduino Uno* e a *Raspberry Pi,* foi escolhido por apresentar uma solução simples e de baixo custo para o projeto. Uma alternativa seria realizar essa comunicação através da internet, porém para isso seria necessário adquirir o *Arduino Ethernet Shield,* porém o custo desse dispositivo no mercado é de aproximadamente R$50,00[7], enquanto o kit com o transmissor e receptor RF possui um custo de aproximadamente R$10,00[8]. Foram levadas em conta também as limitações que a comunicação via RF possui, como limitação da distância entre o transmissor e o receptor e a não transmissão do sinal devido a obstáculos no caminho como paredes por exemplo. Essas limitações puderam ser observadas na prática, pois em alguns testes a informação não era recebida pelo receptor, portanto pode-se constatar que o receptor e transmissor RF embora sejam uma solução viável financeiramente em algumas situações pode comprometer o funcionamento do sistema. Outro fator que contribuiu para o uso da comunicação via RF foi a oportunidade de adicionar o conceito de transmissão de dados via RF ao projeto, enriquecendo o aprendizado com o desenvolvimento do mesmo.

Os relés utilizados se mostraram muito eficientes nos testes e de fácil controle, o que comprova que foi uma ótima escolha para o projeto.

Com toda a arquitetura implementada então pode-se comprovar o funcionamento do sistema, com um certo atrasado entre a entrada do usuário e o acionamento do relé. Esse atraso está em torno de um a dois segundos, causado principalmente pelo tempo que se é necessário para publicar a mensagem no *Twitter* e para que a mesma seja identificada pela *Raspberry Pi.* Embora esse atraso exista, ele não afeta o uso do sistema, portanto pode-se afirmar com base nos testes realizados que a implementação de todo o sistema foi realizada sucesso.

O sistema final está ilustrado na figura 19.

[7] PORTAL ML. Disponível em: www.mercadolivre.com.br. Acessado em 15/06/2014.
[8] PORTAL ML. Disponível em: www.mercadolivre.com.br. Acessado em 15/06/2014.

Figura 19 – Sistema Final

6 CONCLUSÕES E CONSIDERAÇÕES FINAIS

Durante todos os estudos feitos para esse projeto, pode-se perceber o quão grande é o leque de ferramentas disponíveis para serem utilizadas em processos de automação residencial. Os critérios para a seleção dessas ferramentas foram basicamente, custo, eficácia, simplicidade e o quanto determinada ferramenta permitiria a exploração e implementação de novos conceitos.

Considerando todos os fatos aqui relatados assim como os resultados obtidos, pode-se concluir que através das soluções propostas alcançou-se o objetivo deste projeto. O uso da *Raspberry Pi* e do *Arduino Uno* como central de processamento de informações para a automação residencial se mostrou muito eficaz considerando o fator custo para o controle de processos residenciais. Existem limitações resultantes dessas plataformas, como o número de portas disponíveis por exemplo, portanto para atividades com nível de complexidade elevada sugere-se realizar uma pesquisa ainda mais profunda sobre as mesmas para se ter certeza que elas atendam as necessidades de tal atividade. Para aplicações simples como a desse projeto essas plataformas são altamente recomendáveis devido a sua eficácia, fácil acesso e simplicidade.

O transmissor e receptor RF utilizados para a comunicação entre o *Arduino Uno* e *Raspberry Pi* apresentaram uma baixa eficácia para a transmissão de mensagens, portanto para outros projetos recomenda-se o uso de transmissores e receptores com um maior nível de qualidade.

A plataforma *Xcode,* embora seja um pouco complexa para ser utilizada de início, após a familiarização com o ambiente pode-se notar a sua riqueza em ferramentas para implementar nos aplicativos buscando sempre facilitar o desenvolvimento. O aplicativo desenvolvido possui uma interface simples e intuitiva, cumprindo seu papel principal que é facilitar a interação entre o usuário e todo o processo envolvido no projeto.

O *Twitter* se mostrou uma ferramenta muito prática para aplicações simples como essa, dispensando a necessidade de se configurar um servidor web quando a única intenção é se transmitir alguns comandos via *web.*

Pode-se concluir então que com esse projeto foi possível conciliar vários conceitos diferentes como comunicação RF, programação orientada a objetos e sistemas embarcados. A integração de todos esses conceitos foi o resultado de muito estudo e pesquisa, resultando em um enorme aprendizado em todos os temas citados. Conclui-se que o tema automação residencial é muito vasto e que a combinação de conhecimentos diferentes pode resultar em soluções inovadoras e criativas para a área. A solução proposta por esse projeto se mostrou capaz de permitir que o usuários acionem seus dispositivos a distância, porém ainda sim apresenta limitações devido aos materiais utilizados.

- Trabalhos Futuros

Para trabalhos futuros, sugere-se que seja implementado um sistema de monitoramento dos equipamentos eletrônicos que estão ligados aos relés através do uso de sensores. Com esse sistema de monitoramento todo o projeto ficará mais inteligente, pois será possível verificar através do aplicativo quais objetos estão em uso. Esse sistema mais inteligente dará ao usuário a oportunidade de ter um maior controle sobre sua residência possibilitando até que haja uma economia de energia elétrica, uma vez que equipamentos eletrônicos deixados ligados quando não estão sendo usados poderão ser desligados a distância.

Uma outra melhoria que pode ser implementada no projeto é o uso de transmissores de sinais infravermelhos. Com essa nova funcionalidade e algumas adaptações no aplicativo desenvolvido será possível controlar uma televisão através do celular, mudando seus canais ou usando qualquer outra função do controle remoto.

Enfim, o intuito desse projeto foi desenvolver um sistema de automação residencial simples e adaptável para novas soluções que por mais simples que sejam, possam contribuir para o aumento do controle do ambiente residencial para o usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

DENNIS, A. K. ***Raspberry Pi* home automation with Arduino:** Automate your home with a set of exciting projects for the Raspberry Pi. Birmingham: Packt Publishing Ltd, 2013.

GRANA, A. L. S. **Sistema de controle de navegação embraçado para um robô móvel autônomo baseado no computador de baixo custo Rasperry Pi.** 2013. 64 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica com ênfase em Eletrônica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

OTONI, P. P. **Ambiente para automação via Web Semântica utilizando Linus embarcado em micro controladores ARM.** 2013. 81 f Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

RUWAIDA, B.; MINKKINEN, T. **Home automation system:** A cheap and open-source alternative to control household appliances. 2013. 45 f. Tese (Bacharelado) – School of Information and Communication Technology (ICT), KTH Royal Institute of Technology, Estocolmo. 2013. Disponível em: <http://www.diva-portal.org/smash/record.jsf?pid=diva2:679674>. Acesso em: 29/06/2014.

VIEIRA, M. A. **Sistema de telemetria para robôs móveis.** 2011. 91 f. Trabalho de Conclusão de Curso (Graduação em Engenharia da Computação) – Universidade Federal do Vale do São Francisco, Juazeiro, 2011. Disponível em: < http://www.univasf.edu.br/~ccomp/monografias/monografia_3.pdf>. Acesso em: 29/06/2014

BIBLIOGRAFIA COMPLEMENTAR

Shivaram, K.; Rajendra, N.; Mahesh, K.; Murthy, K.N.B.; Jawahar, P. "**Power line communication based automation system using a handheld Wi-Fi device**", Consumer Electronics (ISCE), 2012 IEEE 16th International Symposium on, On page(s): 1 – 6. Acesso em 14 jun. 2014

BRUCE, J. **How to build a home automation system with *Raspberry Pi* and Arduino.** Makeuseof, Copyright, ago. 2013. Disponível em: < http://www.makeuseof.com/tag/how-to-build-home-automation-system-raspberry-pi-and-arduino/>. Acesso em: 29/06/2014.

ETEKCITY | **5 pack Wireless Remote Control Outlet Light Switch Socket with 1 remote.** Copyright Etekcity Corporation 2014. Disponível em < http://www.etekcity.com/>. Acesso em 29 jun. 2014.

GADGETOID. Disponível em: http://pi.gadgetoid.com/article/clockatoo-*Twitter*-feed. Acesso em: 15/08/2014.

LEARN SPARKFUN. Disponível em: https://learn.sparkfun.com/tutorials/raspberry-pi-*Twitter*-monitor. Acesso em: 15/08/2014.

PORTAL ARDUINO. Disponível em: http://www.arduinoecia.com.br/2013/02/ligando-uma-lampada-com-rele.html. Acesso em: 28/10/2014.

PORTAL CODE. Disponível em: https://code.google.com/p/tweepy/wiki/GettingStarted. Acesso em: 28/10/2014.

PORTAL EMBARCADOS. Disponível em: http://www.embarcados.com.br/arduino-uno/. Acesso em: 28/10/2014.

PORTAL HOAGIESHOUSE. Disponível em: **http://www.hoagieshouse.com/RaspberryPi/RCSockets/RCPlug.html. Acesso em: 28/10/2014.**

PORTAL HOMAUTOMATION. Disponível em: http://www.homautomation.org/2014/03/02/433mhtz-rf-communication-between-arduino-and-raspberry-pi-arduino-as-receiver/. Acesso em: 15/08/2014.

PORTAL RASPBERRYPI. Disponível em: **http://www.raspberrypi.org/forums/.** Acesso em 28/10/2014.

PORTAL RASPBERRYPI. Disponível em: http://www.raspberrypi.org/forums/viewtopic.php?f=32&t=80334. Acesso em 28/10/2014.

PORTAL STACKOVERFLOW. Disponível em: **http://stackoverflow.com/questions.** Acesso em: 28/10/2014.

RASPBERRY PI FOUNDATION. Desenvolvido por Jack Lang, David Braben, Louis Glass, Pete Lomas e Alan Mycroft. United Kingdom: **UK Registered Charity 1129409**. 2012. Disponível em: < http://www.raspberrypi.org/>. Acesso em: 29/06/2014.

PORTAL GITHUB. Disponível em: **https://github.com/ninjablocks/433Utils/tree/master/RPi_utils**. Acesso em: 15/08/2014.

RAYWENDERLICH. Disponível em: **http://www.raywenderlich.com/21558/beginning-*Twitter*-tutorial-updated-for-ios-6.** Acesso em: 15/08/2014.

APENDICE A - Software para *Raspberry Pi* – tccv1.py

```
# -*- coding: utf-8 -*-
import time
import os
import RPi.GPIO as GPIO
from twython import TwythonStreamer
# Termos que serao monitorados
TERMOS = 'Ligar R1, Ligar R2, Desligar R1, Desligar R2'
# Chaves para o acesso ao twitter
APP_KEY = 'j7kYAYgsd4dlqRUQK6Xrv81rU'
APP_SECRET = 'Wie2l9Pc3hOuVBfji07YyH5vW0EQdi2cUPkfvi5OkZmnlmTklL'
OAUTH_TOKEN = '2837657895-oMqK5BGzAq4pbrk95iBOZHyjQRKUlZDvNd3o1OD'
OAUTH_TOKEN_SECRET = 'v4Y7pnl5ELe1VKgHMF0xafsF5jmxpHcc3OpQZBWxiOjkl'

# Monitora o twitter
class BlinkyStreamer(TwythonStreamer):
    def on_success(self, data):
        if 'text' in data:
            print data['text'].encode('utf-8')
            print
        if data['text'].find('Ligar R1') != -1:
            os.system("sudo ./codesend 7689219")
        if data['text'].find('Desligar R1') != -1:
            os.system("sudo ./codesend 7689228")
        if data['text'].find('Ligar R2') != -1:
            os.system("sudo ./codesend 7689264")
        if data['text'].find('Desligar R2') != -1:
            os.system("sudo ./codesend 7689408")
# Inicio do monitoramento
try:
    stream = BlinkyStreamer(APP_KEY, APP_SECRET, OAUTH_TOKEN,
OAUTH_TOKEN_SECRET)
    stream.statuses.filter(track= TERMOS)
except KeyboardInterrupt:
    GPIO.cleanup()
```

APENDICE B - Software para *Raspberry Pi* – codesend

```
#include "RCSwitch.h"
#include <stdlib.h>
#include <stdio.h>


int main(int argc, char *argv[]) {

    // Esse pino corresponde ao 11 da raspberry pi
    int PIN = 0;

    //  Armazena o código recebido
    int mensagem = atoi(argv[1]);
    // Lógica para enviar a mensagem
     if (wiringPiSetup () == -1) return 1;
        printf("sending code[%i]\n", mensagem);
        RCSwitch transmitirCodigos = RCSwitch();
        transmitirCodigos.enableTransmit(PIN);

    transmitirCodigos.send(code, 24);

        return 0;

}
```

APENDICE C - Software para Arduino - tccArduino

```
#include <RCSwitch.h>

RCSwitch codigo = RCSwitch();

void setup() {
Serial.begin(9600);
codigo.enableReceive(0);
//pinos conectados aos reles
pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop() {
  if (codigo.available()) {

    int value = codigo.getReceivedValue();

    if (value == 0) {
      Serial.print("Unknown encoding");
    } else {
      Serial.print("Codigo recebido: ");
      Serial.print( codigo.getReceivedValue() );
      Serial.print(" / ");
      Serial.print( codigo.getReceivedBitlength() );
      Serial.print("bit ");

      if (codigo.getReceivedValue() == 7689219 ){
            digitalWrite(5, HIGH);
            Serial.println("R1 On");
      }
      if (codigo.getReceivedValue() == 7689228 ){
            digitalWrite(5, LOW);
            Serial.println("R1 Off");
      }
      if (codigo.getReceivedValue() == 7689264 ){
```

```
            digitalWrite(6, HIGH);
            Serial.println("R2 On");
      }
      if (codigo.getReceivedValue() == 7689408 ){
            digitalWrite(6, LOW);
            Serial.println("R2 OFF");
      }
    }

    codigo.resetAvailable();
  }
}
```

APENDICE D - Módulos do aplicativo *My Home*

View2.h

```
//
//  View2.h
//  iControl
//
//  Created by Eduardo on 10/4/14.
//  Copyright (c) 2014 Eduardo. All rights reserved.
//

#import "ViewController.h"

#import <UIKit/UIKit.h>
#import <Twitter/Twitter.h>
#import <Social/Social.h>

@interface View2 : ViewController{

IBOutlet UILabel *label1;
IBOutlet UILabel *label2;
IBOutlet UILabel *label3;
IBOutlet UILabel *label4;
IBOutlet UILabel *label5;
}
-(IBAction)R1on:(id)sender;
-(IBAction)R1off:(id)sender;
-(IBAction)R2on:(id)sender;
-(IBAction)R2off:(id)sender;
-(IBAction)R3on:(id)sender;
-(IBAction)R3off:(id)sender;
-(IBAction)R4off:(id)sender;
-(IBAction)R4off:(id)sender;

@end
```

Módulo do aplicativo *My Home* – View2.m

```
//
//  View2.m
//  iControl
//
//  Created by Eduardo on 10/4/14.
//  Copyright (c) 2014 Eduardo. All rights reserved.
//

#import "View2.h"
#import <Social/Social.h>
#import <UIKit/UIKit.h>
#import <Twitter/Twitter.h>
#import <EventKit/EventKit.h>
#import <EventKitUI/EventKitUI.h>
#import <LocalAuthentication/LocalAuthentication.h>

@interface View2 ()

@end

NSString *dateString;

@implementation View2

//Função para se obter a data e a hora
-(void)currentDateTime{

    NSDateFormatter *dateFormat = [[NSDateFormatter alloc] init];

    [dateFormat setDateStyle:NSDateFormatterMediumStyle];
    [dateFormat setTimeStyle:NSDateFormatterNoStyle];
    dateString=[dateFormat stringFromDate:[NSDate date]];
    [dateFormat setDateFormat:@"yyyy-MM-dd HH:mm:ss"];
    dateString=[dateFormat stringFromDate:[NSDate date]];
}


//liga rele 1
-(IBAction)R1on:(id)sender{

    [self currentDateTime];

    if ([SLComposeViewController isAvailableForServiceType:SLServiceTypeTwitter])
        {
            SLComposeViewController *tweetSheet = [SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTwitter];
            NSString *mensagem = [NSString stringWithFormat:@"%@ Ligar R1", dateString];
            [tweetSheet setInitialText:mensagem];
            [self presentViewController:tweetSheet animated:YES completion:nil];

        }
        else
```

```
        {
            UIAlertView *alertView = [[UIAlertView alloc]
                                      initWithTitle:@"Ops"
                                      message:@"Erro devido a falta de conexão com a internet ou nao há
uma conta do twitter cadastrada"


                                      delegate:self
                                      cancelButtonTitle:@"OK"
                                      otherButtonTitles:nil];
            [alertView show];
        }
}

//desligar rele 1
-(IBAction)R1off:(id)sender{

    [self currentDateTime];

      if ([SLComposeViewController isAvailableForServiceType:SLServiceTypeTwitter])
   {
       SLComposeViewController *tweetSheet = [SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTwitter];
       NSString *mensagem = [NSString stringWithFormat:@"%@ Desligar R1", dateString];
       [tweetSheet setInitialText:mensagem];
       [self presentViewController:tweetSheet animated:YES completion:nil];

   }

   else
   {
       UIAlertView *alertView = [[UIAlertView alloc]
                                 initWithTitle:@"Ops"
                                 message:@"Erro devido a falta de coneção a internet ou nao há uma
conta do twitter cadastrada"


                                 delegate:self
                                 cancelButtonTitle:@"OK"
                                 otherButtonTitles:nil];
       [alertView show];
   }
}

//ligar relé 2
-(IBAction)R2on:(id)sender{

   [self currentDateTime];

   if ([SLComposeViewController isAvailableForServiceType:SLServiceTypeTwitter])
   {
       SLComposeViewController *tweetSheet = [SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTwitter];
       NSString *mensagem = [NSString stringWithFormat:@"%@ Ligar R2", dateString];
       [tweetSheet setInitialText:mensagem];
```

```
        [self presentViewController:tweetSheet animated:YES completion:nil];

    }

    else
    {
        UIAlertView *alertView = [[UIAlertView alloc]
                                  initWithTitle:@"Ops"
                                  message:@"Erro devido a falta de coneção a internet ou nao há uma
conta do twitter cadastrada"


                                  delegate:self
                                  cancelButtonTitle:@"OK"
                                  otherButtonTitles:nil];
        [alertView show];
    }
}



//desligar relé 2
-(IBAction)R2off:(id)sender{

    [self currentDateTime];

    if ([SLComposeViewController isAvailableForServiceType:SLServiceTypeTwitter])
    {
        SLComposeViewController *tweetSheet = [SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTwitter];
        NSString *mensagem = [NSString stringWithFormat:@"%@ Desligar R2", dateString];
        [tweetSheet setInitialText:mensagem];
        [self presentViewController:tweetSheet animated:YES completion:nil];

    }

    else
    {
        UIAlertView *alertView = [[UIAlertView alloc]
                                  initWithTitle:@"Ops"
                                  message:@"Erro devido a falta de coneção a internet ou nao há uma
conta do twitter cadastrada"


                                  delegate:self
                                  cancelButtonTitle:@"OK"
                                  otherButtonTitles:nil];
        [alertView show];
    }
}

//Botões ainda não utilizados
-(IBAction)R3on:(id)sender{
    UIAlertView *mensagem = [[UIAlertView alloc] initWithTitle:@"Ops" message:@"Ainda
não existe um relé associado a esse botão" delegate:nil cancelButtonTitle:@"Voltar"
```

```
otherButtonTitles:nil];
    [mensagem show];}
-(IBAction)R3off:(id)sender{
    UIAlertView *mensagem = [[UIAlertView alloc] initWithTitle:@"Ops" message:@"Ainda
não existe um relé associado a esse botão" delegate:nil cancelButtonTitle:@"Voltar"
otherButtonTitles:nil];
    [mensagem show];}
-(IBAction)R4on:(id)sender{
    UIAlertView *mensagem = [[UIAlertView alloc] initWithTitle:@"Ops" message:@"Ainda
não existe um relé associado a esse botão" delegate:nil cancelButtonTitle:@"Voltar"
otherButtonTitles:nil];
    [mensagem show];}
-(IBAction)R4off:(id)sender{
    UIAlertView *mensagem = [[UIAlertView alloc] initWithTitle:@"Ops" message:@"Ainda
não existe um relé associado a esse botão" delegate:nil cancelButtonTitle:@"Voltar"
otherButtonTitles:nil];
    [mensagem show];}



- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
    }
    return self;
}

- (void)viewDidLoad
{

    [super viewDidLoad];
    }

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}

@end
```