UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

MURILO DALVI PITOL

UMA ESTRATÉGIA PARA O ACIONAMENTO E MONITORAMENTO REMOTO DE EQUIPAMENTOS VIA INTERNET

VITÓRIA – ES
DEZEMBRO/2015

MURILO DALVI PITOL

UMA ESTRATÉGIA PARA O ACIONAMENTO E MONITORAMENTO REMOTO DE EQUIPAMENTOS VIA INTERNET

Parte manuscrita do Projeto de Graduação do aluno **Murilo Dalvi Pitol**, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientadora: Profa. Dra. Rosane Bodart Soares
Coorientador: Prof. Renato B. Cabelino Ribeiro

VITÓRIA – ES
DEZEMBRO/2015

MURILO DALVI PITOL

UMA ESTRATÉGIA PARA O ACIONAMENTO E MONITORAMENTO REMOTO DE EQUIPAMENTOS VIA INTERNET

Parte manuscrita do Projeto de Graduação do aluno **Murilo Dalvi Pitol**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

COMISSÃO EXAMINADORA:

_____

Profa. Dra. Rosane Bodart Soares
Universidade Federal do Espírito Santo - UFES
Orientadora

_____

Prof. Renato B. Cabelino Ribeiro
Instituto Federal do Espírito Santo - IFES
Coorientador

_____

Prof. Dr. Marcelo Eduardo Vieira Segatto
Universidade Federal do Espírito Santo - UFES
Examinador

_____

Eng. Leonardo de Assis Silva
Universidade Federal do Espírito Santo - UFES
Examinador

VITÓRIA – ES
DEZEMBRO/2015

AGRADECIMENTOS

Primeiramente a Deus, por estar sempre comigo e me dar forças.

A minha mãe Patrícia e a minha irmã Camila pelo apoio incondicional e por sempre me motivarem a fazer o meu melhor.

Ao meu pai Antônio, não apenas pelo apoio moral, mas também por me auxiliar na confecção do protótipo.

A todos meus familiares, especialmente ao meu primo Gabriel por compartilhar comigo seus conhecimentos em redes de computadores.

A todos os meus amigos e colegas de curso da UFES, por travarem essa batalha comigo ao longo de todos esses anos e nunca me deixarem desanimar perante as dificuldades. Em especial a todos os membros do grupo “THIS IS ELÉTRICA!!”, vocês são como irmãos para mim.

Por último mas não menos importante, a todos os professores do departamento de Engenharia Elétrica da UFES, em especial a minha orientadora, Rosane, e ao meu coorientador, Renato, por toda a ajuda fornecida para a realização deste projeto de graduação.

A todos vocês, o meu mais sincero MUITO OBRIGADO!

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Domótica

À medida que testemunham-se os avanços da tecnologia nos dias atuais, conceitos que antes pareciam impossíveis vão se tornando coisas naturais no cotidiano do ser humano. Como exemplo deste avanço tecnológico surge o termo domótica, que é o resultado da junção da palavra “domus”, que significa casa, com a palavra “robótica”, que está relacionada a ação de automatizar algo (SISLITE, 2015). Portanto, este termo é usado atualmente para designar a tecnologia responsável pela automatização dos recursos habitacionais, criando os chamados ambientes inteligentes, como mostrado na figura abaixo.

Figura 1 – Domótica

Fonte: SISLITE, 2015.

O ser humano está cada vez mais em busca de conforto e qualidade de vida, de realizar o mínimo esforço com atividades cotidianas, monótonas e rotineiras. Seguindo esta tendência, origina-se o conceito de domótica com o propósito de atender a esta demanda. Por meio dela também é possível obter melhorias em outros aspectos, como a segurança e economia energética, por exemplo.

Sem dúvidas, um grande divisor de águas para a domótica foi o surgimento e aprimoramento de componentes como os microprocessadores, relés e sensores, pois todas as áreas nas quais a automação estava inserida sofreram significativas

melhorias na qualidade de seus equipamentos, inclusive o setor da automação residencial. A partir deste ponto, ideias que até então eram inviáveis foram se tornando plausíveis.

Foi em meados dos anos 1970 que a domótica começou a dar seus primeiros passos. Em 1975, uma tecnologia conhecida como X-10, um protocolo de comunicação entre aparelhos usado para automação residencial, permitiu que aparelhos e luzes se comunicassem entre si, sendo que em 1978 surgiram os primeiros dispositivos X-10 nas prateleiras das lojas. Foi lançado um software de automação para computadores pessoais em 1980 e um sistema de segurança doméstica em 1989. Mas, somente ao longo da década de 1990 que a tecnologia X-10 veio a ganhar popularidade como uma tecnologia de automação residencial (BIONDO, 2011).

Nos dias de hoje, praticamente qualquer sistema eletrônico ou mecânico de uma residência pode ser projetado para funcionar com maior performance através da automação residencial. Há uma gama enorme de funções nas quais esta tecnologia está inserida, desde o acionamento de portões por controle remoto até o controle de equipamentos via Internet, conforme será abordado neste projeto de graduação.

Apesar de haver uma engenharia extremamente complexa envolvida por trás deste conceito, a tecnologia é sempre voltada ao público leigo e portanto deve apresentar uma interface de fácil compreensão e altamente intuitiva. A domótica consegue ir de encontro a essa linha de raciocínio pois ações complexas são possíveis apenas com um simples toque de uma tela ou apertando-se um botão, gerando uma facilidade incrível para o usuário.

Dado o conforto e comodidade que pode proporcionar, aliado a uma redução significativa de preço com o decorrer dos anos, a domótica promete vir a ter cada vez mais adeptos, face a quantidade crescente de projetos com estas soluções de automatização. A partir de então, a facilidade promovida pelo controle remoto de funções gerais em um ambiente, por meio da Internet e dispositivos móveis, deixa de ser uma utopia e se torna uma realidade totalmente palpável.

Nota-se isso com os inúmeros trabalhos acadêmicos existentes na área de domótica, a citar Beghini (2013), no qual foi desenvolvido uma estratégia de automação residencial baseada na plataforma *Arduino* através da criação de um aplicativo para dispositivos móveis com sistema operacional *Android*, servindo como inspiração para a ideia desenvolvida neste trabalho. Porém, diferentemente de Beguini, optou-se por não desenvolver um aplicativo, mas sim um *site* programado em linguagem HTML. Além do mais, foi feito o uso do sensor PIR para a detecção de presença no ambiente, por ser uma solução bem mais barata do que uma câmera IP.

Assim, de forma geral, visa-se à construção de um protótipo buscando o monitoramento por meio de sensores e acionamento de equipamentos (lâmpada e ventilador) de forma remota via *web*. Para tal, será desenvolvido um *web server* hospedado no *Arduino* com auxílio de um *Ethernet Shield*. Este projeto foi desenvolvido de modo que suas aplicações possam ser estendidas para ambientes reais, caso este seja o intuito.

1.2 Motivação

O ser humano está em uma busca constante por conforto e praticidade. É totalmente instintivo utilizar a tecnologia em prol de seu benefício, reduzindo o esforço para realizar atividades cotidianas. A domótica traz essa comodidade ao usuário, se inserindo com perfeição na situação atual da humanidade.

Outro aspecto muito importante e no qual há um enorme interesse é o meio ambiente. Há uma preocupação em garantir que os recursos naturais disponíveis sejam preservados para que as gerações futuras possam usufruir de um planeta melhor, com mais qualidade de vida. Logo, faz-se necessário criar sistemas e equipamentos que vão de encontro a essa vertente, sendo notável que tudo que é produzido atualmente tem a missão de ser ambientalmente viável. Equipamentos são aprimorados para emitirem menos gases poluentes, para serem biodegradáveis ou para serem mais eficientes e consumirem menos energia elétrica.

Porém, de nada adianta o desenvolvimento de equipamentos mais eficientes e econômicos se estes não forem devidamente utilizados. É comum testemunhar desperdício de energia elétrica ao percorrer os corredores de uma escola, por exemplo. No centro tecnológico da UFES é corriqueiro encontrar salas com equipamentos sendo usados de forma desnecessária, como lâmpadas acesas ou o ar condicionado ligado quando não há ninguém no ambiente.

Em suma, este trabalho possui um porquê muito forte. Ele vem para combinar dois conceitos em alta atualmente: A automação residencial e a preservação dos recursos energéticos. Assim, estará sendo criado um ambiente inteligente capaz de proporcionar comodidade ao usuário junto a um menor desperdício de energia elétrica, visto que será possível, de forma remota, desligar equipamentos desnecessariamente ligados quando o ambiente não está sendo utilizado.

1.3 Objetivo geral e objetivos específicos

1.3.1 Objetivo geral

Implementar um sistema e um protótipo com a finalidade de monitorar e acionar equipamentos de forma remota por meio da Internet.

1.3.2 Objetivos específicos

- Fazer uma análise da teoria envolvida no projeto, explicitando minuciosamente toda a engenharia por trás de seu funcionamento;
- Projetar o sistema que permita o monitoramento e controle remoto de equipamentos por meio de interface em dispositivos como computadores *tablets* e *smartphones*;
- Analisar os resultados obtidos;
- Sugerir trabalhos futuros que possam se basear neste projeto.

1.4 Estrutura do trabalho

O Capítulo 1 apresenta a justificativa deste trabalho, seu contexto na sociedade atual, além de expor objetivos gerais e específicos do mesmo.

O Capítulo 2 trata dos aspectos teóricos e principais conceitos referentes aos equipamentos utilizados no desenvolvimento do projeto.

O Capítulo 3 aborda a execução do projeto, explicando o passo-a-passo de cada etapa necessária para sua realização.

O Capítulo 4 exibe os resultados obtidos e discussões pertinentes.

O capítulo 5 conclui o projeto de modo geral, menciona as dificuldades encontradas e faz sugestões para projetos futuros.

Em seguida, são apresentadas as referências utilizadas como base para o desenvolvimento desta monografia.

Finalmente, é apresentado o apêndice contendo o código feito no IDE do *Arduino.*

2 EMBASAMENTO TEÓRICO

Neste capítulo será abordado os aspectos teóricos envolvidos no desenvolvimento deste projeto, assim como os principais componentes e equipamentos que foram utilizados e que merecem um maior detalhamento a respeito de sua estrutura e funcionamento.

2.1 Arduino Uno

O cérebro deste projeto de graduação é a placa *Arduino UNO*, que surgiu na Itália no ano de 2005. Basicamente, ele é uma plataforma de prototipagem eletrônica criada com o objetivo de permitir o controle de sistemas interativos, de baixo custo e acessíveis a todos. Além disso, todo material (*software*, bibliotecas, *hardware*) é *open-source*, o que garante que pode ser reproduzido e usado por todos sem a necessidade de pagamento de direitos autorais (BASCONCELLO, 2012). Sua plataforma é dividida essencialmente em duas partes: O *hardware* e o *software*.

2.1.1 Hardware

As principais especificações técnicas do *Arduino Uno* segundo o próprio *site* do produto (ARDUINO-1, 2015) estão citadas a seguir:

-**Microcontrolador**: ATmega328

-**Tensão de operação**: 5V

-**Tensão de entrada (recomendada)**: 7-12V

-**Tensão de entrada (limites):** 6-20V

-**Pinos de entrada/saída digitais**: 14 (Das quais 6 podem fornecer saída PWM)

-**Pinos de entrada analógica:** 6

-**Corrente DC por pino de entrada/saída**: 20 mA

-**Memória Flash**: 32 KB, (Dos quais 0,5KB são usados para Bootloader)

-**SRAM**: 2 KB

-**EEPROM:** 1 KB

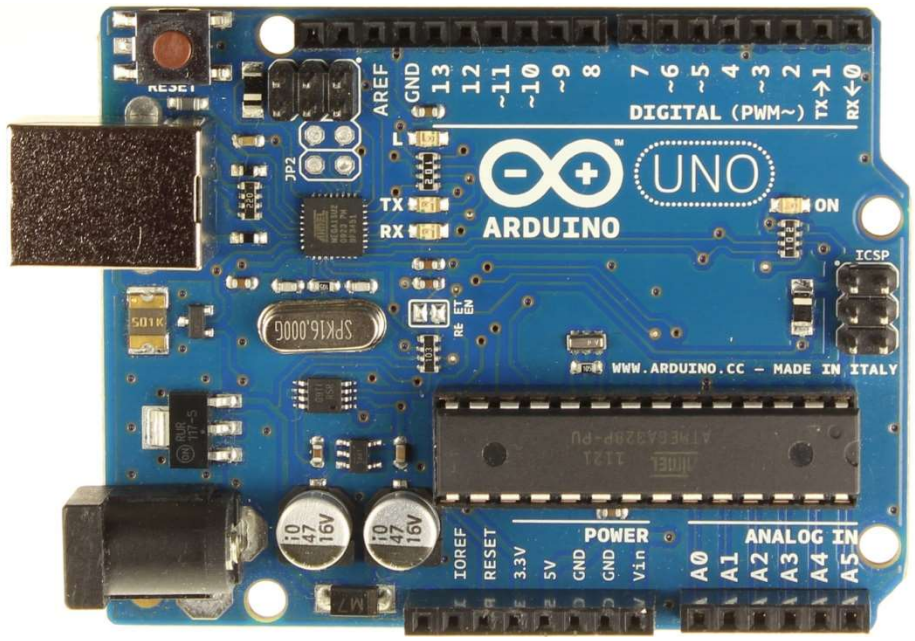
-**Frequência de clock** : 16 MHz

A alimentação da placa pode ser feita via USB ou através de uma fonte externa. Caso seja via conexão USB com um computador, não será necessário regular a tensão. No caso de alimentação externa, a tensão de entrada é filtrada e regulada para 5V, para que só então os demais blocos, circuitos externos ou *shields* possam ser alimentados.

As entradas analógicas são aquelas através das quais podem ser lidos sensores que convertem grandezas físicas em valores contínuos de tensão; As entradas/saídas digitais são informadas ou informam se há ou não tensão em um determinado pino, ou seja, geram sinais de saída e entrada de nível lógico baixo ou alto. Dentre essas entradas/saídas digitais, existem 6 saídas PWM (pinos 3,5,6,9,10 e 11). Estas são tratadas como saídas analógicas, mas na verdade são saídas digitais que geram um sinal alternado no qual a duração do nível lógico alto pode ser controlada.

A Figura 2 mostra o *Arduino Uno*.

Figura 2: Arduino Uno

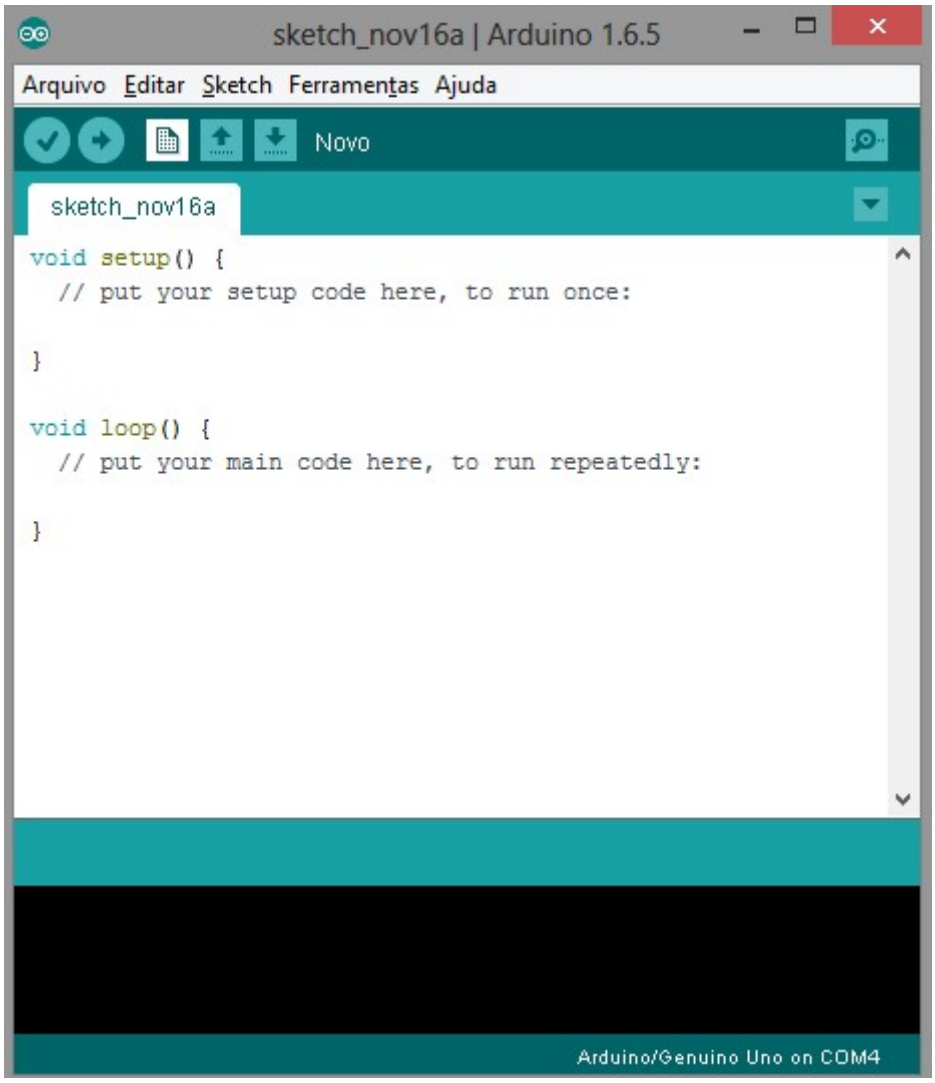


Fonte: Arduino, 2015

2.1.2 Software

O ambiente de desenvolvimento do *Arduino,* exibido na Figura 3, é um compilador gcc (C e C++) que usa uma interface gráfica construída em *Java*. Basicamente se resume a um programa IDE (*integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) muito simples de se utilizar e de estender com bibliotecas que podem ser encontradas sem grandes dificuldades (BASCONCELLO, 2012). As funções da IDE do *Arduino* são basicamente duas: Permitir o desenvolvimento de um *software* para executar determinada função e enviá-lo à placa para que possa de fato ser executado. Por meio deste programa, o *Arduino* realizará o controle de equipamentos de forma remota.

Figura 3: Ambiente de programação do Arduino

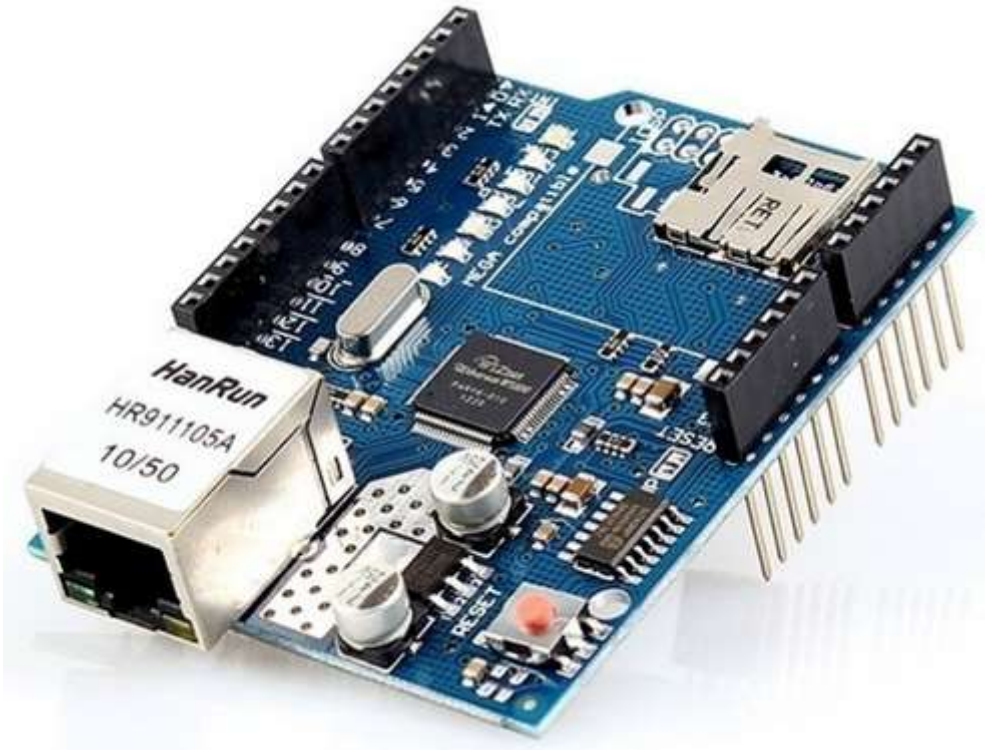


Fonte: Produção do próprio autor

2.2 Ethernet Shield

O *Ethernet Shield,* ou módulo *Ethernet*, consiste de uma placa de circuito impresso que ao ser acoplada ao *Arduino* permite que ele se conecte a Internet por meio de um cabo de rede padrão RJ-45. A figura abaixo mostra o *Ethernet Shield*.

Figura 4: Ethernet Shield



Fonte: FILIPEFLOP, 2015

A conexão do *Ethernet Shield* com o *Arduino UNO* é feita por meio do barramento SPI (*Serial Peripheral Interface*), compreendido pelos pinos digitais 10,11,12 e 13. Por sua vez, a conexão do *Ethernet Shield* com a rede é possível graças ao *chip* W5100 da *Wiznet* que fornece o protocolo TCP/IP à placa *Arduino*. A seleção do W5100 é feita por meio do pino 10 (ARDUINO-2, 2015). .

O TCP/IP é o mais importante conjunto de protocolos de envio e recebimento de dados pela Internet. Ele é composto por 4 camadas, cada uma delas responsável por executar tarefas distintas, de modo a garantir a integridade dos dados que estão trafegando pela rede. (TECMUNDO-1, 2015). Estas são:

Camada de Aplicação: Camada utilizada por programas para enviar e receber informações de outros programas através da rede. Uma vez que os dados tenham sido processados pela camada de aplicação, eles são enviados para a camada seguinte. Nesta camada, por exemplo, encontra-se o famoso protocolo HTTP (*HyperText Transfer Protocol*) usado para navegar na Internet.

Camada de Transporte: recebe os dados da camada anterior, verifica sua integridade e divide-os em pacotes.

Camada de Rede: Os dados empacotados anteriormente são recebidos nesta camada e são endereçados com o IP do dispositivo remetente e do dispositivo receptor.

Camada de Interface: Camada que especifica os detalhes de como os pacotes são transmitidos fisicamente pela rede. Neste trabalho, o padrão de transmissão utilizado será o *Ethernet*, muito usado para interconexões LAN (*Local Area Network*) ou rede local.

Em suma, o protocolo TCP/IP pode ser ilustrado como na figura abaixo.

Figura 5: Representação em camadas do protocolo TCP/IP

Fonte: Produção do próprio autor

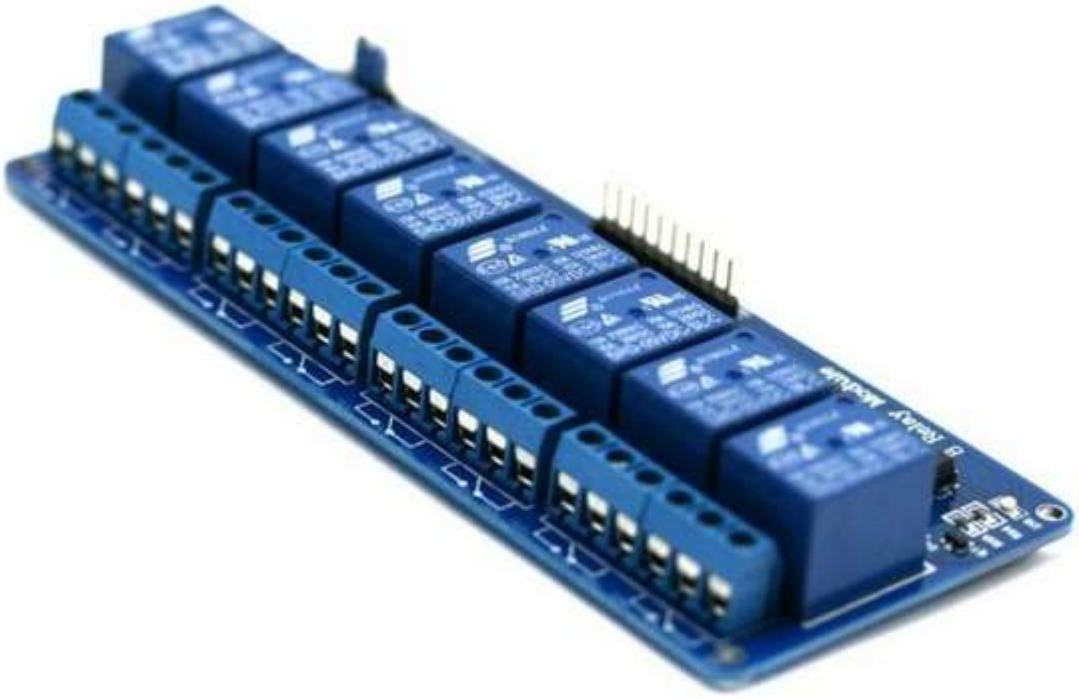
Vale ressaltar que apenas 10 entradas/saídas digitais estarão disponíveis para controle de equipamentos pois os outros 4 pinos serão usados para a conexão com o *Ethernet Shield*.

2.3 Relay Shield

O *Relay Shield*, ou módulo relé, é uma placa de circuito impresso que permite o acionamento de múltiplas cargas de forma fácil, sem a necessidade de confeccionar circuitos com transistores, diodos, entre outros componentes. De forma simplificada, ele será o atuador do sistema.

Neste trabalho, o *Relay Shield* utilizado possui 8 canais, igual ao mostrado na Figura 6. Esta quantidade de canais é suficiente para a automatização de um quarto de uma casa, um escritório ou uma sala de aula, por exemplo. Ele é alimentado em uma fonte DC de 5V, suporta cargas alimentadas em até 220V AC e aciona equipamentos que possuam até 10A de corrente nominal. Sua pinagem é do tipo NF (normalmente fechada), NA (normalmente aberta) e comum.

Figura 6: Relay Shield



Fonte: FILIPEFLOP, 2015

2.4 Sensor PIR

O Sensor PIR (*Passive Infrared Sensor*) é um sensor de movimento capaz de detectar a presença de radiação infravermelha. Esta radiação é emitida por todo objeto que gera calor e é invisível ao olho humano, porém pode ser detectada com o uso de circuitos eletrônicos projetados para tal.

Ele é dito passivo pelo fato de não gerar nenhuma energia para o processo de detecção. O Sensor PIR possui um circuito integrado que tem por função amplificar os sinais analógicos de detecção e modular, a partir dele, um sinal de saída em nível digital. Portanto, ao detectar a presença de radiação, será atribuído nível lógico alto à saída. Caso contrário, a saída será zero.

O sensor utilizado será o PIR DYP-ME003, exibido na Figura 7, que consegue detectar objetos em movimento até uma distância de 7 metros. O tempo que o nível lógico fica presente na saída é ajustável entre 5 a 200 segundos, assim como sua sensibilidade a detecção. Sua tensão de operação varia de 4,5 à 20V (FILIPEFLOP, 2015).

Figura 7: Sensor PIR

Fonte: FILIPEFLOP, 2015

2.5 Sensor de corrente

O sensor de corrente não-invasivo é um transformador de corrente, abreviadamente TC, dispositivo capaz de reproduzir em seu circuito secundário a corrente que circula nos enrolamentos do primário, com mesma posição vetorial e proporção definida. Desta forma pode-se obter correntes suficientemente reduzidas e isoladas do primário, possibilitando que este sinal seja tratado por equipamentos eletrônicos de medição, controle ou proteção. Para efetuar a medição, basta passar um dos fios do equipamento através da abertura do sensor.

Neste trabalho, foi feito uso do modelo SCT-013-000 da YHDC, exibido na Figura 8, que permite a leitura de no máximo 100A, com saída de até 50mA.

Figura 8: Sensor de corrente

Fonte: artofcircuit, 2015

2.6 Sensor de tensão

O sensor de tensão da Figura 9 tem a capacidade não somente de detectar a existência ou não de tensão alternada em um determinado circuito, mas também especificar qual o valor dessa diferença de potencial elétrico. Apesar de projetado para trabalhar com uma entrada de 127V até 220V, consegue suportar uma tensão máxima de 311V. Este sensor é dotado de um optoacoplador que garante o isolamento elétrico da entrada para com os demais componentes do circuito. Possui três pinos para conexão: Um para alimentação de 5V, um para *ground* e outro para o sinal de saída. Como este sensor é conectado em paralelo com o circuito desejado, seu circuito drena muito pouca corrente da carga, podendo ser considerada desprezível. (MERCADOLIVRE, 2015).

Figura 9: Sensor de tensão

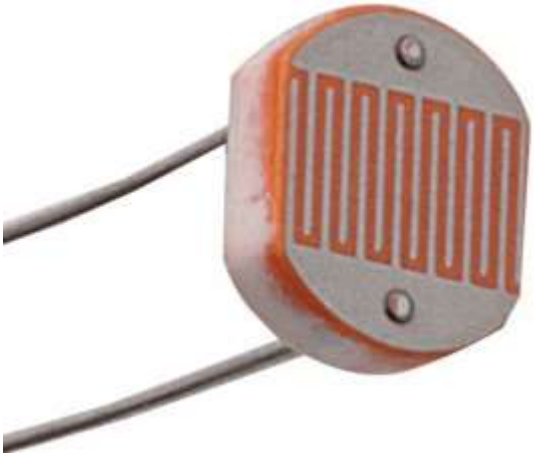
Fonte: Mercadolivre, 2015

2.7 LDR

O LDR, *Light Dependent Resistor* ou Resistor Dependente de Luz, é um resistor que varia sua resistência de acordo com a intensidade luminosa incidente sobre ele.

Este sensor de luz é constituído de Cádmio, um material semicondutor que possui a propriedade de diminuir sua resistência elétrica quando a luminosidade sobre ele aumenta. Já quando submetido a baixa luminosidade, sua resistência é aumentada. É um componente extremamente barato, de fácil acesso e com uma boa confiabilidade. O LDR é ilustrado na Figura 10.

Figura 10: LDR



Fonte: dnatechindia, 2015

2.8 Roteador

Outro equipamento necessário no projeto será um roteador *Wireless*, que basicamente é um aparelho que realiza a conexão sem fio de múltiplos computadores ou aparelhos móveis (*smartphones* e *tablets*) conectados em uma rede local com a Internet, permitindo assim a transmissão e recebimento de pacotes de dados através das melhores e mais eficientes rotas de acesso.

Existem três especificações destes dispositivos que caracterizam a sua eficiência e são extremamente relevantes, sendo a primeira delas a velocidade da transmissão de dados que o roteador consegue atingir. Desta forma, quanto maior for a velocidade de acesso, mais rápido poderão ser os *uploads* e *downloads* realizados por seus dispositivos.

A segunda é observar o ganho da antena, em dBi, que é um dos indicadores do alcance do sinal de transmissão de dados. Desta forma, quanto maior for a potência da antena, maior será a área coberta pela rede local controlada pelo roteador.

Outra informação que é importante conhecer é o padrão de conectividade da rede sem fio, que indica a frequência da transmissão de dados entre seus dispositivos e o roteador e é caracterizado pelos padrões de rede *Wireless* 802.11b (conexão mais antiga), 802.11g e 802.11n, sendo esses dois últimos correspondentes as frequências mais modernas (TECMUNDO-2, 2015). Um exemplo de roteador é mostrado na figura a seguir.

Figura 11: Roteador

Fonte: TECHTUDO, 2015

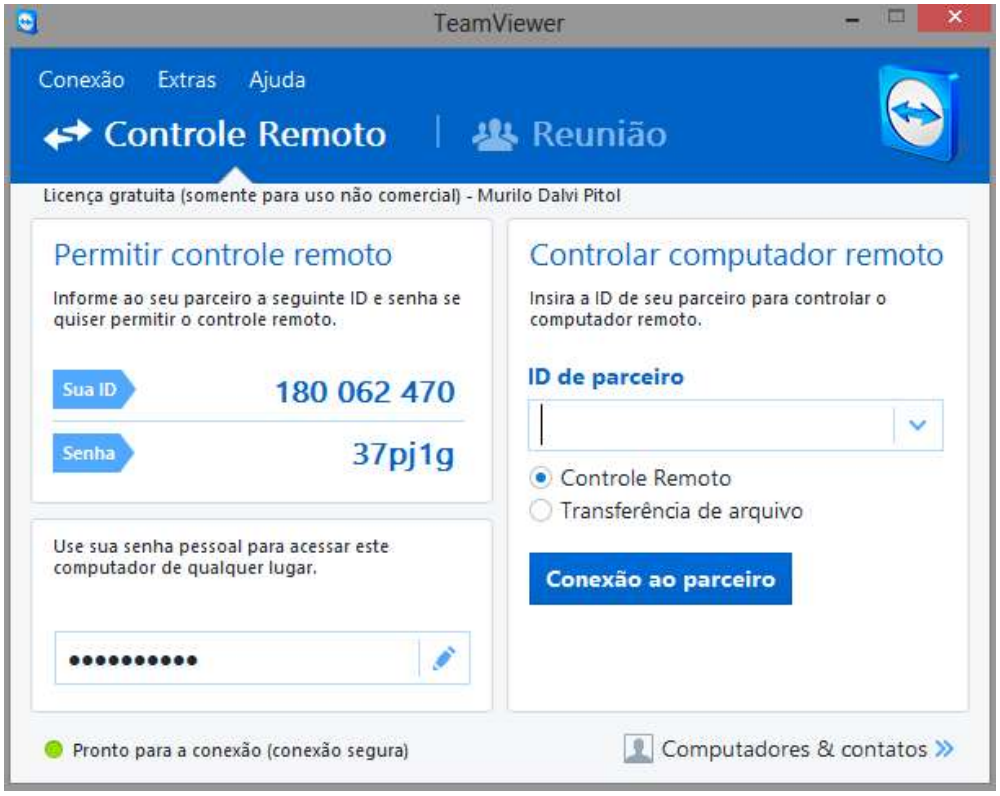
2.9 TeamViewer

TeamViewer é um *software* gratuito criado na Alemanha em 2005 e que permite que um computador possa ser comandado por outro situado em qualquer parte do mundo. Trata-se de um dos *softwares* mais populares e utilizados no mundo todo quando se trata de acesso remoto, compartilhamento de área de trabalho e transferência de arquivos entre computadores. Possui atualmente cerca de 200 milhões de usuários no mundo todo (TEAMVIEWER, 2015).

Apresenta muitos pontos positivos, como uma interface amigável, segurança e fácil utilização. Basta fornecer o ID e senha para que o computador possa ser controlado.

A interface do programa é exibida abaixo.

Figura 12: Interface do software TeamViewer



Fonte: Produção do próprio autor

3 EXECUÇÃO DO PROJETO

Neste capítulo será detalhado todo o passo-a-passo para a execução do trabalho, mostrando a construção e montagem do protótipo, programação do *Arduino* e da pagina *web*, o sensoriamento do sistema, entre outras etapas importantes no desenvolvimento do projeto.

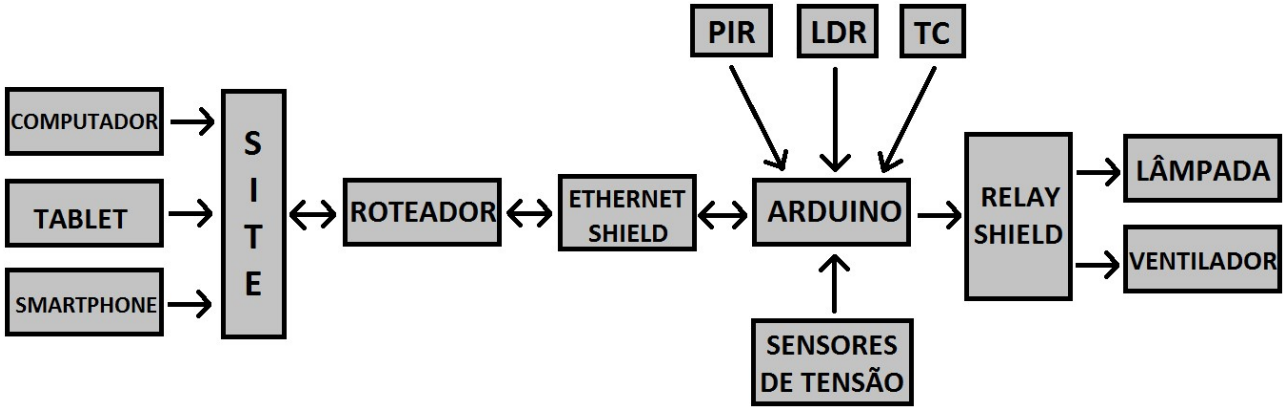
3.1 Diagrama de blocos do sistema

Como mencionado anteriormente, o objetivo deste trabalho é acionar equipamentos de forma remota por meio da Internet. De modo bem resumido, o protótipo funcionará da seguinte forma: O *Arduino* e *Ethernet Shield* desempenharão o papel de um *web server*, hospedando uma pagina *web* em sua memória. Nela será informado se há ou não pessoas no ambiente por meio da coleta de dados de um sensor PIR. Um LDR e um TC informarão o status da lâmpada e do ventilador, respectivamente. Através de sensores de tensão, será explicitado no *site* se os equipamentos estão ou não conectados a tomada. Por ultimo, será também exibido um botão para ligar ou desligar cada equipamento individualmente.

Quando um comando é dado pelo *site*, este será enviado ao roteador, que por sua vez redirecionará o comando ao conjunto *Arduino* e *Ethernet Shield* via cabo de rede RJ-45. O comando será interpretado pelo *Arduino*, que repassará o comando ao *Relay Shield*, ordenando que o relé chaveie para assim ligar ou desligar o aparelho desejado.

O diagrama de blocos do sistema pode ser visto na Figura 13.

Figura 13: Diagrama de blocos do sistema



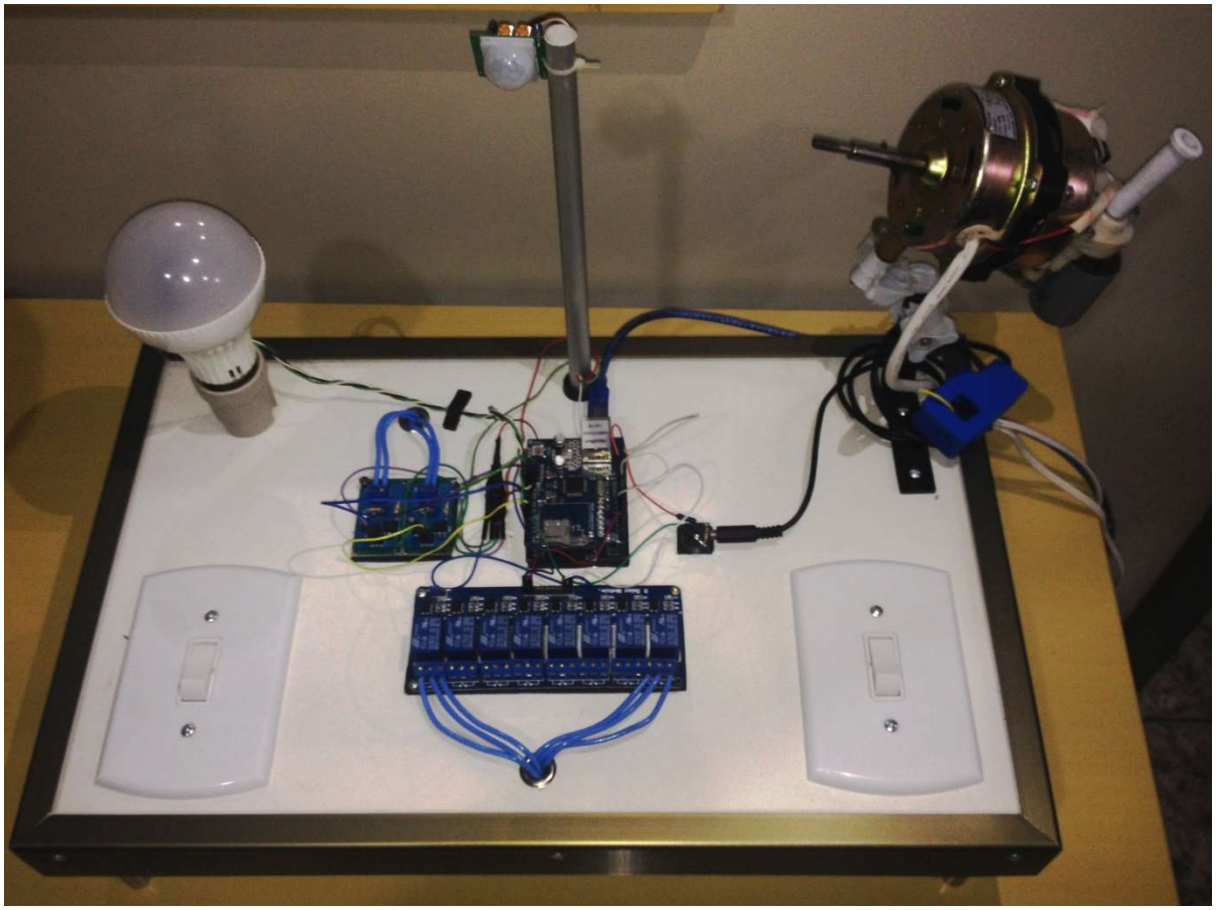
Fonte: Produção do próprio autor

.

3.2 Protótipo

Definido os equipamentos que seriam controlados para os fins deste trabalho, realizou-se a montagem do protótipo constituido de uma bancada com uma lâmpada de LED e o motor de um ventilador, sensores para determinar o status dos equipamentos, além de um sensor PIR para detecção de movimento como mostrado na Figura 14.

Figura 14: Protótipo

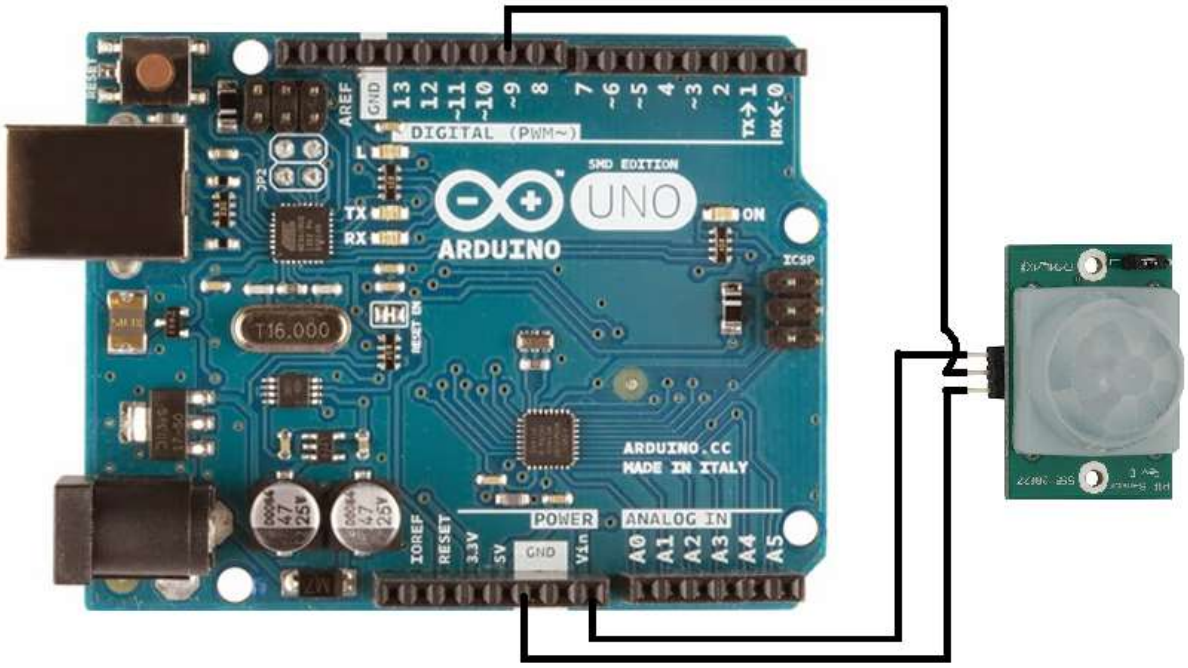


Fonte: Produção do próprio autor

3.3 Detector de presença

Antes de controlar os equipamentos de um ambiente é crucial saber se há ou não pessoas naquele recinto. Para tal finalidade, foi feito o uso de um sensor PIR alimentado em 9V com sinal de saída sendo lido por uma entrada digital denominada “PIR”, conforme o esquemático mostrado na Figura 15.

Figura 15: Esquemático da ligação do PIR no Arduino



Fonte: Produção do próprio autor

O valor dessa porta foi armazenada na variável “statusPIR”. Se “statusPIR” tiver nível lógico alto, houve a detecção de pessoas. Se for baixo, não foi detectado movimento. A sensibilidade foi definida para o valor máximo e a saída do PIR foi ajustada manualmente para se manter em nível lógico alto por aproximadamente 60 segundos. Este valor de tempo foi escolhido arbitrariamente. O trecho do código referente ao funcionamento do PIR está exibido na Figura 16.

Figura 16: Fragmento do código referente ao sensor PIR

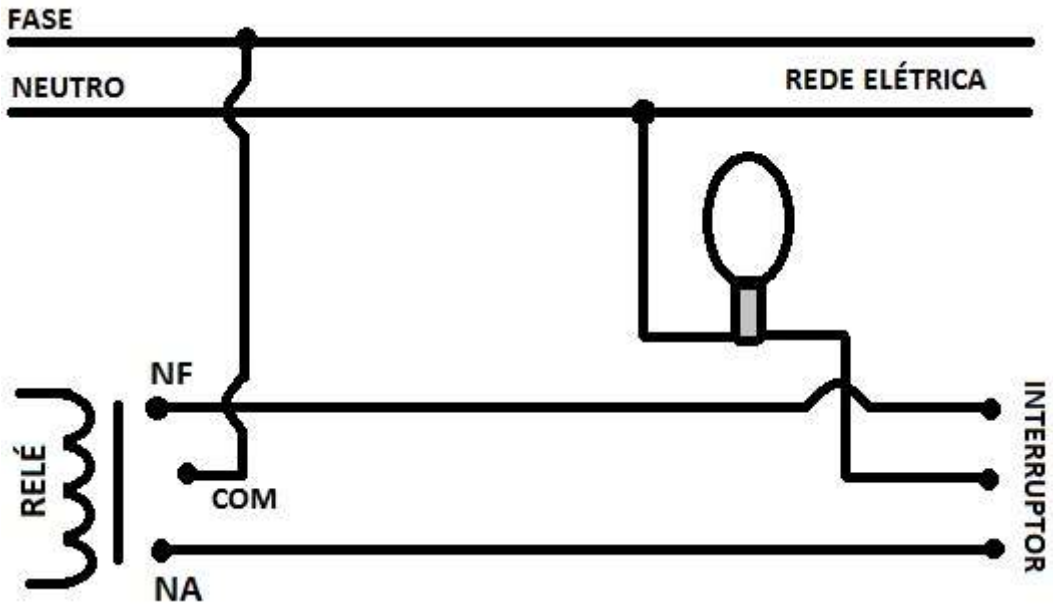
```
#define PIR 9
pinMode(PIR, INPUT); //define PIR como entrada digital 9
statusPIR = digitalRead(PIR);
if (statusPIR == HIGH)
  client.println("<font color=green  >COM PESSOAS");
else if (statusPIR == LOW)
  client.println("<font color=red >SEM PESSOAS");
```

Fonte: Produção do próprio autor

3.4 Controle da lâmpada

Para o acionamento da lâmpada foi feito uso de um interruptor do tipo *Three-Way* conectado em paralelo com o módulo relé, conforme a Figura 17.

Figura 17: Esquema de ligação da lâmpada

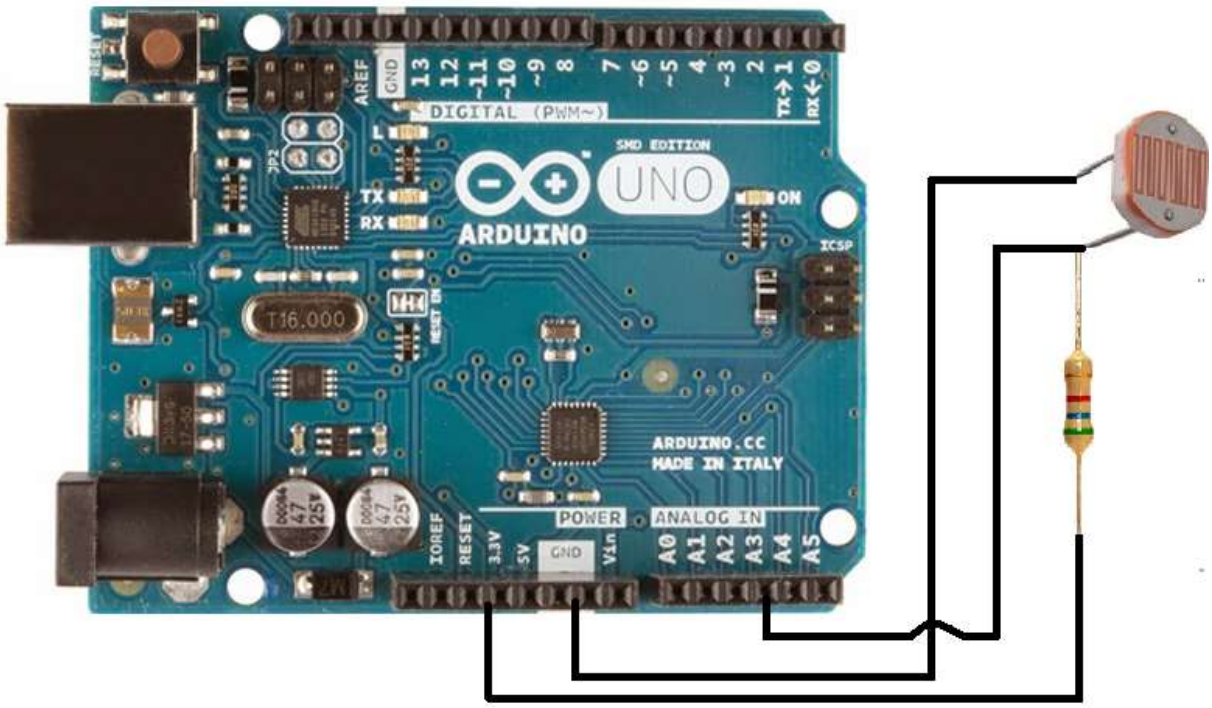


Fonte: Produção do próprio autor

Desta forma, garante-se que será possível controlar a lâmpada tanto pela Internet, por meio de um computador, *tablet* ou *smartphone*, como também da maneira convencional.

Outro ponto importante é determinar o status da lâmpada antes do acionamento, de modo a informar ao usuário se ela está desligada ou ligada. Para esta função, foi feito uso de um circuito muito simples composto por um resistor em série com um LDR, fazendo a leitura da tensão sobre o LDR através de uma das entradas analógicas do *Arduino* denominada “LDR”, conforme a Figura 18. Posicionou-se o LDR envolvendo um dos LEDs da lâmpada, de forma que ao acendê-la ou apagá-la a resistência elétrica do LDR fosse alterada, mudando assim a queda de tensão sobre este componente e o valor lido pela entrada “LDR”.

Figura 18: Esquemático da ligação do LDR no Arduino



Fonte: Produção do próprio autor

Foram feitos diversos testes, imprimindo os valores do sinal lido sobre o LDR quando a lâmpada era ligada ou desligada. Dessa forma, de maneira empírica, foi possível obter um limiar, ou seja, um número para o qual valores acima dele implicariam que a lâmpada estava apagada e valores abaixo significariam que ela estava acesa. Foi usado uma variável denominada “statusLDR” que recebia 0 ou 1 para valores acima e abaixo do limiar, respectivamente. A parte do código referente

a este processo está mostrada na Figura 19. Assim foi possível imprimir na tela o status da lâmpada baseando-se no valor desta variável auxiliar.

Figura 19: Fragmento do código que define valores de statusLDR

```
#define LDR 3
pinMode (LDR, INPUT); //define LDR como entrada analógica 3
tensaoLDR = analogRead(LDR);
if (tensaoLDR > 500) //500 é o valor do limiar
  statusLDR = 0;
else if (tensaoLDR < 500)
  statusLDR = 1;
```

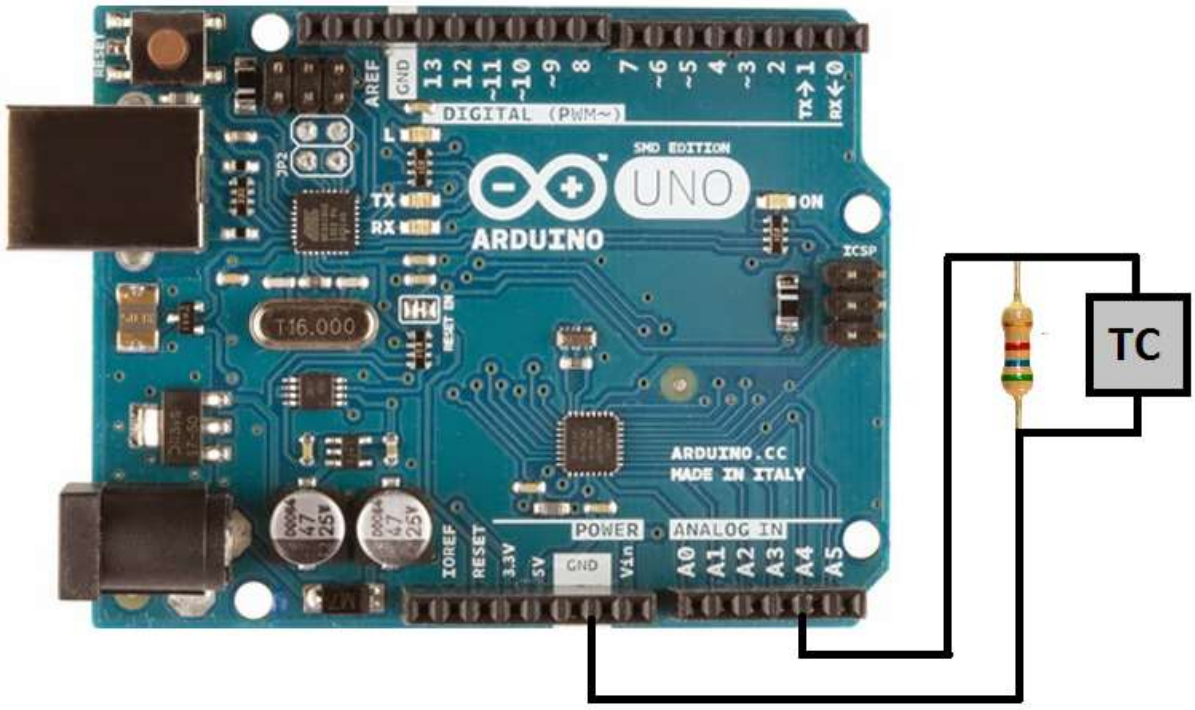
Fonte: Produção do próprio autor

3.5 Controle do Ventilador

Para o acionamento do ventilador foi utilizado o mesmo esquema de ligação de um *Three-Way* em paralelo com o relé, permitindo assim o acionamento via Internet e pelo interruptor.

Para determinar o estado do equipamento, ligado ou desligado, foi feito uso de um TC em paralelo com um resistor. A tensão sobre o resistor foi lida por uma das entradas analógicas do *Arduino*, chamada de “sensor”, conforme a Figura 20. Assim, quando o ventilador for ligado e, portanto, circular uma corrente pelo fio do equipamento, será induzida uma corrente na saída do TC, o que produzirá uma variação na tensão sobre o resistor. Note que para a aplicação deste projeto não se faz necessário saber o valor desta corrente, uma vez que só há interesse em saber se há ou não corrente circulando pelo equipamento. De forma análoga ao caso da lâmpada, foram realizadas várias medidas do sinal sobre o resistor ao ligar e desligar o ventilador para que um limiar pudesse ser definido e assim o estado do equipamento pudesse ser determinado pela variável “statusTC”, que é zero quando o ventilador está desligado e 1 quando ligado.

Figura 20: Esquemático da ligação do TC no Arduino

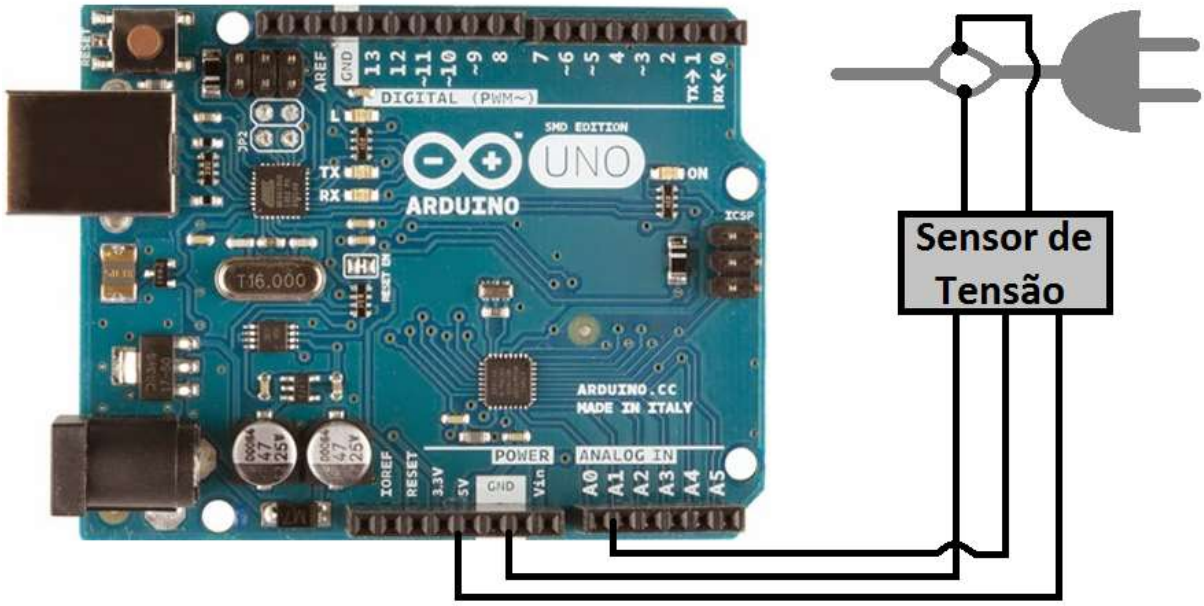


Fonte: Produção do próprio autor

3.6 Sensoriamento de tensão na lâmpada

Antes de acionar a lâmpada, é importante informar ao usuário se o equipamento está ou não conectado a energia elétrica. Para esta função foi utilizado um sensor de tensão cuja entrada é conectada em paralelo ao cabo do equipamento, como explicitado na Figura 21.

Figura 21: Esquemático da ligação do sensor de tensão no arduino



Fonte: Produção do próprio autor

Este sensor é capaz de dizer o valor da tensão inserida em sua entrada quando lido por uma porta analógica do *Arduino.* De forma semelhante ao que foi anteriormente mostrado, testes empíricos foram efetuados com o intuito de definir um limiar de decisão.

Foi feito a leitura da saída do sensor através de uma entrada analógica denominada “tensaoLampada” e este valor foi armazenado na variável “tomadaLampada”. Se o valor de “tomadaLampada” estiver acima do limiar significa que o equipamento está energizado, logo uma mensagem em verde escrita “OK” será exibida no *site*. Caso contrário, um “X” em vermelho será mostrado, explicitando que o aparelho foi removido da tomada.

3.7 Sensoriamento de tensão no ventilador

Para determinar se o motor do ventilador está conectado à tomada, foi utilizado o mesmo sensor e método explicado na sessão 3.6. A parte do código responsável por executar o sensoriamento está explicitada na Figura 22.

Figura 22: Fragmento do código que indica se o motor está ou não energizado

```
#define tensaoMotor 1
pinMode(tensaoMotor, INPUT); //define tensãoMotor como entrada analógica 1
tomadaMotor = analogRead(tensaoMotor);
if (tomadaMotor > 250) //250 é o valor do limiar
  client.println("<font color=green>  OK");
if (tomadaMotor < 250)
  client.println("<font color=red> X");
```

Fonte: Produção do próprio autor

3.8 Lógica de acionamento do programa

A lógica de acionamento da lâmpada e do ventilador são iguais, sendo que a única diferença entre o código das duas são os nomes das variáveis utilizadas. Tendo isso em vista, será feita uma explicação para um equipamento genérico qualquer, apenas para detalhar a lógica da programação feita.

No programa, foi definida para cada equipamento uma saída digital do *Arduino* denominada “rele_equipamento”, a qual foi conectada a entrada do relé. Ou seja, quando escreve-se nível lógico alto nesta saída, o relé é energizado e passa do estado NF para NA. Quando “rele_equipamento” é escrito com nível lógico baixo, a tensão sobre o relé será zero e ele mudará de NA para NF.

Vale lembrar que o acionamento dos equipamentos pode ser feito via interruptor ou pelo relé controlado através do *site*. Sendo que um interruptor possui duas posições possíveis, ligado ou desligado, e um relé pode estar em NA ou NF, conclui-se que o sistema possui quatro possíveis estados. Tendo isso em mente, além do “statusSENSOR”, que é zero quando o equipamento está desligado e 1 quando ligado, foi necessária a criação de uma outra variável chamada “rele”.

É importante mencionar que a variável “rele” é alterada apenas via *click* do botão no *site*, de forma que o acionamento via interruptor não causa nenhuma interferência em seu valor. Ao ser pressionado para ligar ou desligar um aparelho, o botão redirecionará o *site* para um *link* armazenado na string “str”, conforme a Figura 23.

Figura 23: Código dos botões do *site*

```
//ESTADO 00
if (rele == 0 && statusSENSOR == 0)
  client.println("<a href=http://10.0.0.98/p/><button>LAMPADA OFF</button>");

//ESTADO 01
if (rele == 0 && statusSENSOR == 1)
  client.println("<a href=http://10.0.0.98/p/><button>LAMPADA ON</button>");

//ESTADO 10
if (rele == 1 && statusSENSOR == 0)
  client.println("<a href=http://10.0.0.98/g/><button>LAMPADA OFF</button>");

//ESTADO 11
if (rele == 1 && statusSENSOR == 1)
  client.println("<a href=http://10.0.0.98/g/><button>LAMPADA ON</button>");
```

Fonte: Produção do próprio autor

Quando o *link* termina em “p” é atribuido o valor zero a “rele” e nível lógico baixo a “rele_equipamento”. Caso o *link* termine em “g”, a variável “rele” recebe o valor 1 e na saída “rele_equipamento” haverá uma tensão de 5V. Isso é feito pelo comando “str.endsWith()”, como demonstrado da figura a seguir.

Figura 24: fragmento do código que modifica rele e rele_equipamento

```
if (str.endsWith("p"))
{
  rele = 1;
  digitalWrite(rele_equipamento, HIGH);
  delay();
}


if (str.endsWith("g")) {
  rele = 0;
  digitalWrite(rele_equipamento, LOW);
  delay();
}
```

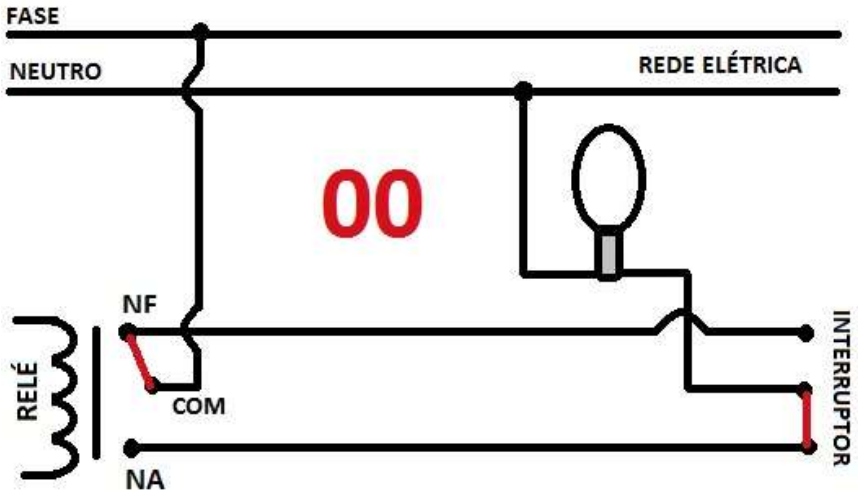
Fonte: Produção do próprio autor

Em suma, quando a variável “rele” for igual a zero, o relé estará na posição NF. Se “rele” for 1, o relé estará chaveado em NA. De posse destas informações, será explicitado os quatro possíveis estados do sistema.

3.8.1 Estado 00

O estado 00 ocorre quando as variáveis “rele” e “statusSENSOR” são zero. Isso significa que o relé está desenergizado, ou seja, na posição de NF e o sensor está detectando o não funcionamento do equipamento. Consiste na situação na qual o equipamento encontra-se desligado. A representação do sistema em estado 00 é mostrada na Figura 25.

Figura 25: Sistema em estado 00

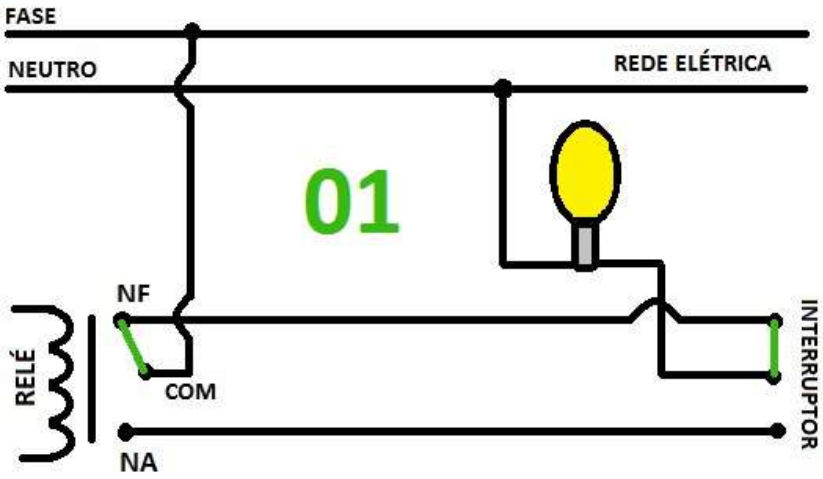


Fonte: Produção do próprio autor

3.8.2 Estado 01

No estado 01, a variável “rele” vale zero, ou seja, o relé está na posição NF e a variável “statusSENSOR” é igual a 1, o que significa que o sensor está acusando que o aparelho está em operação, conforme a figura abaixo.

Figura 26: Sistema em estado 01

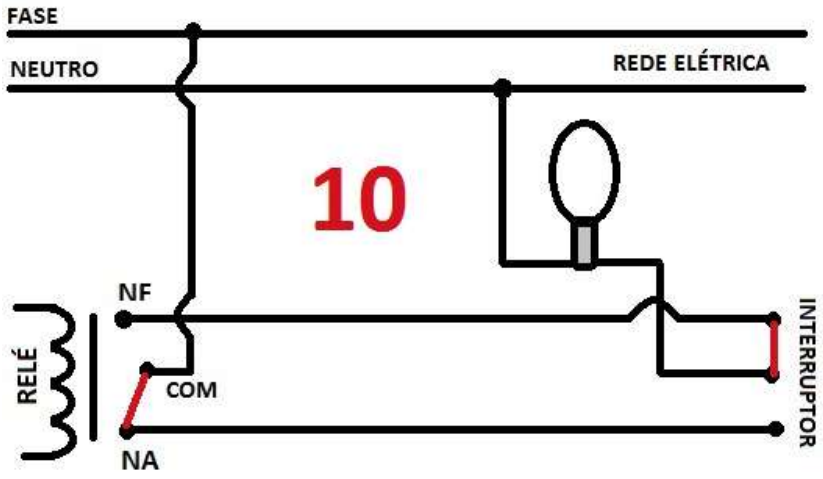


Fonte: Produção do próprio autor

3.8.3 Estado 10

Quando o sistema está em estado 10, o relé está energizado e portanto em posição de NA. Logo, “rele” é igual a 1. Porém o equipamento está desligado pois o circuito está aberto, o que significa que o valor de “statusSENSOR” detectado pelo programa será zero. Na Figura 27 está a representação do sistema em estado 10

Figura 27: Sistema em estado 10

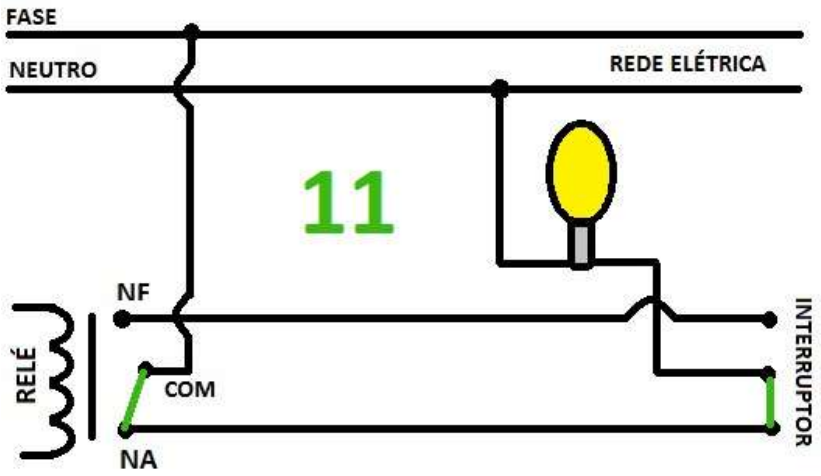


Fonte: Produção do próprio autor

3.8.4 Estado 11

Trata-se do estado quando ambas as variáveis “rele” e “statusSENSOR” são 1. O relé está em posição de NA e o sensor está informando que o aparelho está em operação. O estado 11 está exibido na figura abaixo.

Figura 28: Sistema em estado 11



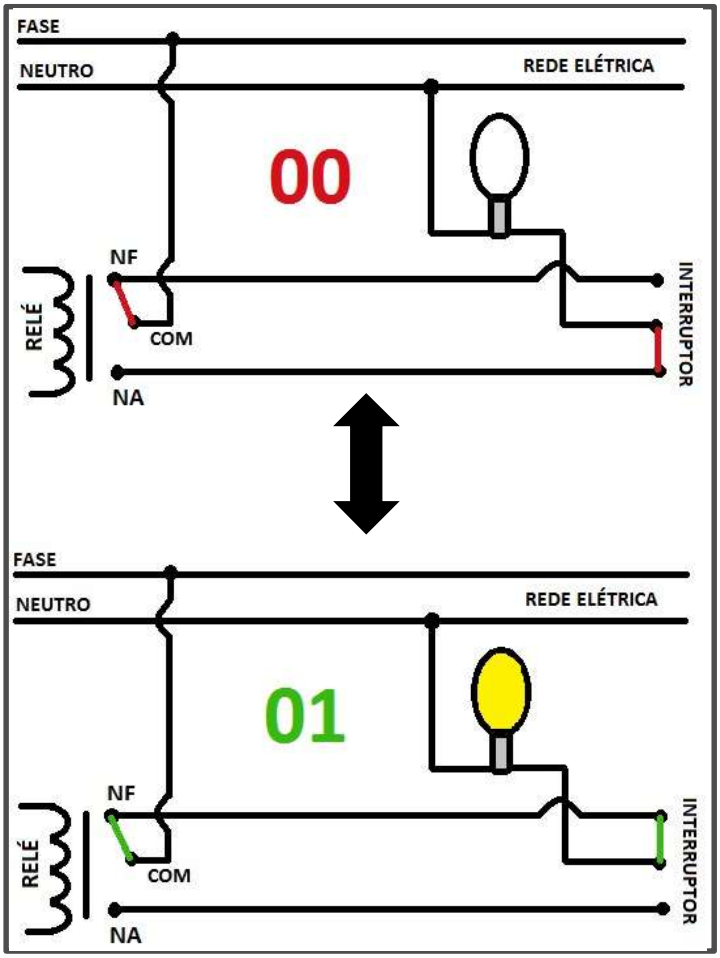
Fonte: Produção do próprio autor

3.8.5 Transição entre estados via interruptor

O acionamento via interruptor implica que apenas a variável “statusSENSOR” será manipulada, uma vez que “rele” só é alterada através do *click* do botão pelo *site*, sendo indiferente ao chaveamento do interruptor. Se inicialmente o sistema estiver em estado 00, o equipamento estará desligado e portanto será exibido no *site* um botão vermelho escrito “EQUIPAMENTO OFF”.

Se o interruptor é pressionado, o equipamento ligará. O sensor detectará o funcionamento do aparelho e a variável “statusSENSOR” passará a ter valor igual a 1. Portanto haverá uma transição do estado 00 para o 01, e a interface será atualizada e passará a exibir um botão verde escrito “EQUIPAMENTO ON”. Se o interruptor for novamente pressionado, o equipamento será desligado e “statusSENSOR” retornará ao valor zero, fazendo o sistema retornar ao estado 00, conforme a figura abaixo.

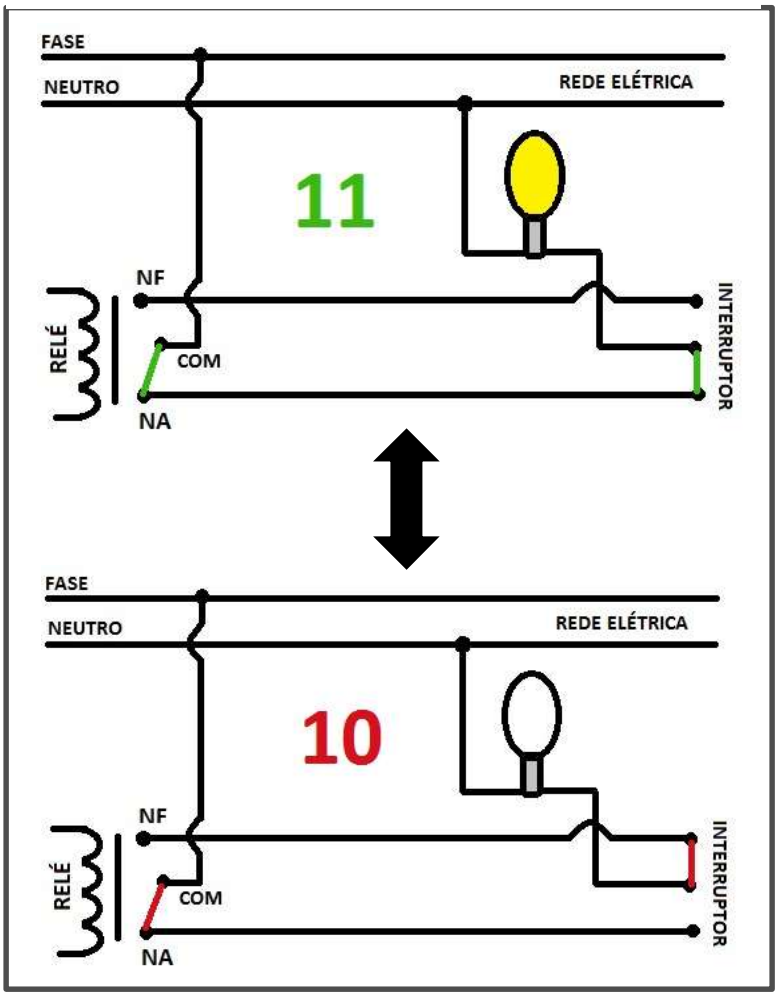
Figura 29: Transição entre os estados 00 e 01



Fonte: Produção do próprio autor

Caso o estado atual do sistema seja 11, o relé está em estado NA e o sensor está identificando que o aparelho está em atividade. Se o interruptor for pressionado, o sistema passará para o estado 10, no qual o relé está energizado mas a variável “statusSENSOR” passou a assumir valor zero pois detectou-se o não funcionamento do aparelho. Se o interruptor for novamente acionado, o equipamento é ligado e “statusSENSOR” volta a ter valor igual a 1. Essas transições são representadas na figura abaixo.

Figura 30: Transição entres os estados 11 e 10



Fonte: Produção do próprio autor

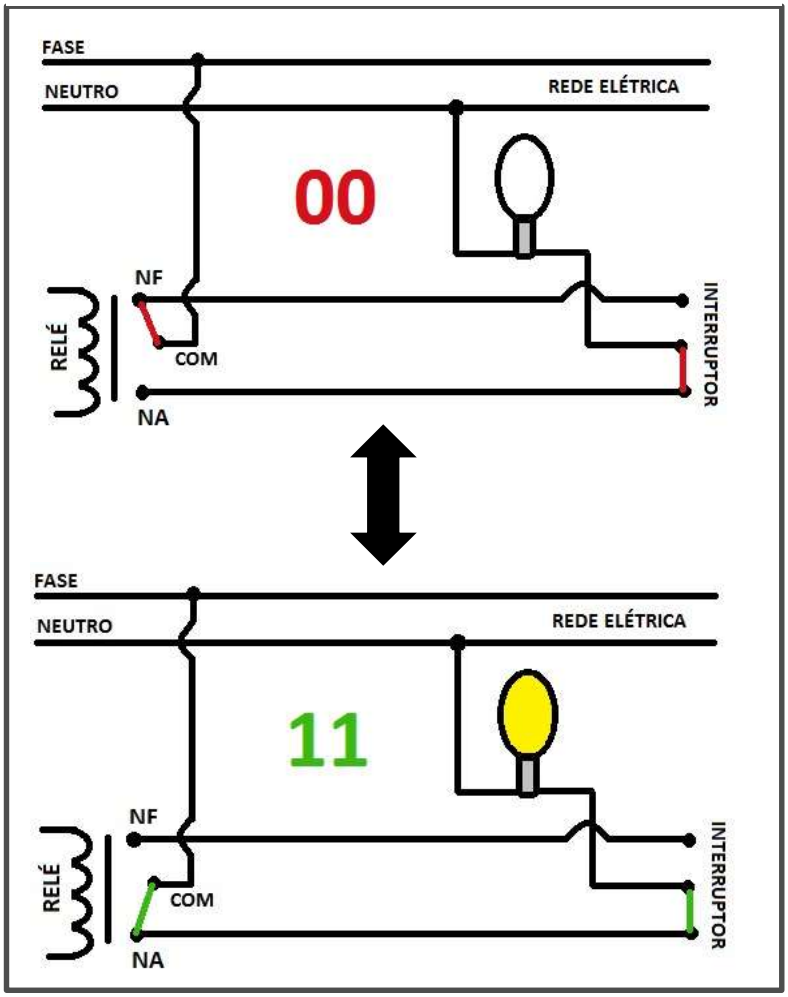
3.8.6 Transição de estados via *site*

É notável que a manipulação da variável “rele” implica na alteração de “statusSENSOR”, pois alterar “rele” significa alterar indiretamente a saída “rele_equipamento”, que por sua vez ligará ou desligará o equipamento, promovendo uma mudança na percepção do sensor e portanto mudando o valor de ”statusSENSOR”. Foi visto anteriormente que a alteração de “statusSENSOR” não implica a mudança de “rele”, no entanto a recíproca não é verdadeira.

Se o sistema encontra-se no estado 00 e o botão no *site* é pressionado, a variável “rele” adquire valor igual a 1, escreve-se nível lógico alto em

“rele_equipamento” e o aparelho será ligado. Portanto “statusSENSOR” também terá valor 1. O sistema então sofrerá transição do estado 00 para o estado 11. Ao encontrar-se no estado 11 e o botão no *site* for pressionado, “rele” será zero, “rele_lampada” irá para nível lógico baixo e o sistema retornará ao estado 00. O esquemático abaixo mostra essas transições.

Figura 31: Transição entres os estados 00 e 11



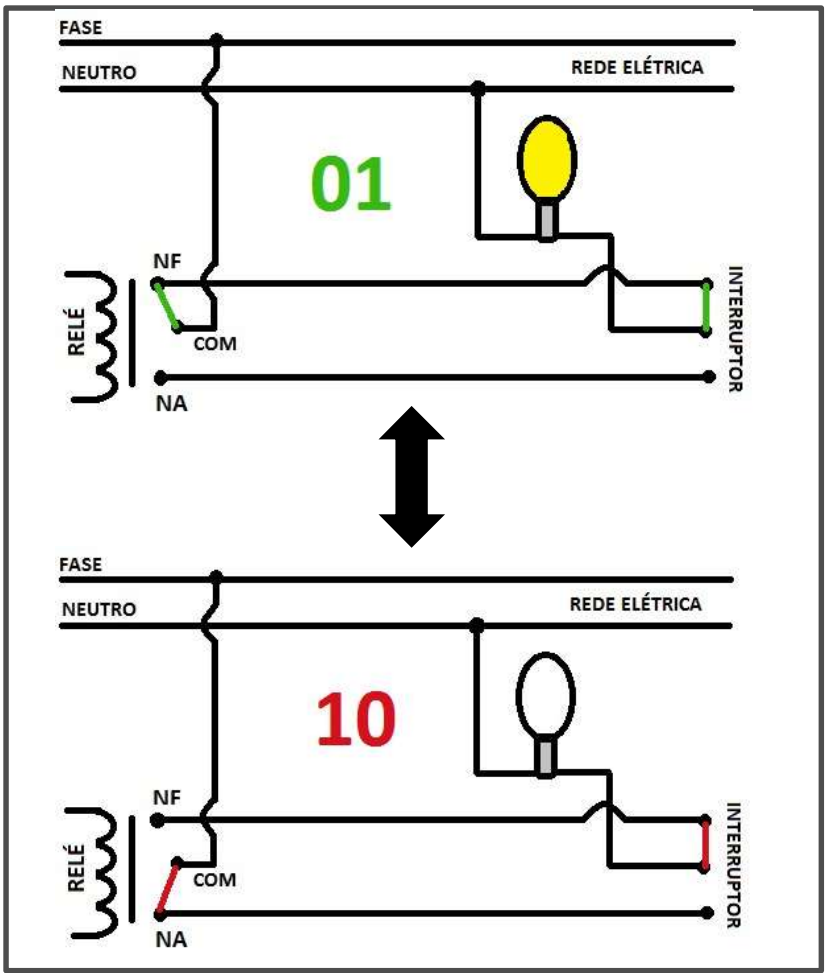
Fonte: Produção do próprio autor

Caso o sistema esteja no estado 01, o relé estará na posição de NF e haverá corrente circulando no circuito. Para desligar o equipamento via internet, é preciso chavear o rele para a posição NA, o que abrirá o circuito. Isso significa mandar o sistema para o estado 10. Para tal, ao ser efetuado o *click* do botão, deve-se atribuir

valor 1 a “rele” e energizar o relé ao por a saída “rele_equipamento” em nível lógico alto.

O caminho inverso de 10 para 01 é obtido ao escrever nível lógico baixo na saída “rele_equipamento” quando o botão for pressionado no *site*. O esquema de transição entre os estados 01 para 10 e vice-versa está mostrado na Figura 32.

Figura 32: Transição entre os estados 01 e 10



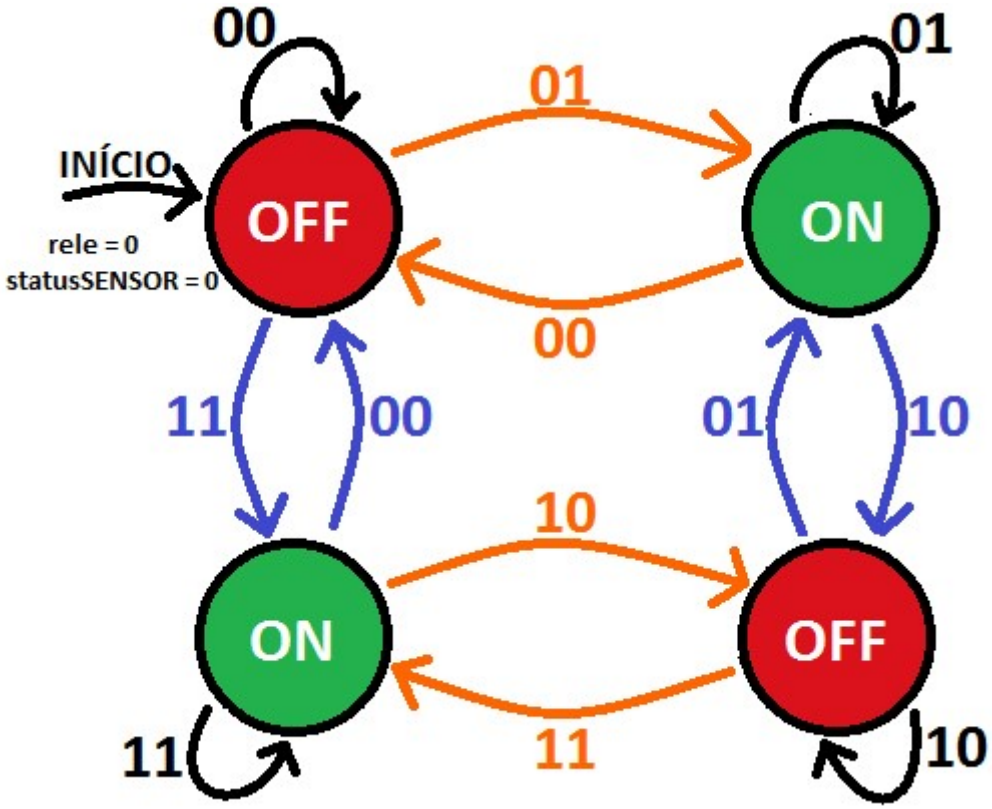
Fonte: Produção do próprio autor

3.9 Máquina de estados do sistema

Por fim, podemos resumir o funcionamento do sistema através de uma máquina de estados como indicado na Figura 33. As setas em cor azul representam

as transições de estados realizadas pelo *click* do botão pelo *site*; As setas em cor laranja mostram as transições de estado possíveis através do interruptor; As setas em cor preta representam os estados nos quais o sistema se encontram quando nenhum comando é dado. É notável que algumas transições estão faltando à máquina apresentada, mas estas foram omitidas propositalmente pelo fato de não serem realizáveis na prática.

Figura 33: Máquina de estados teórica do sistema



Fonte: Produção do próprio autor

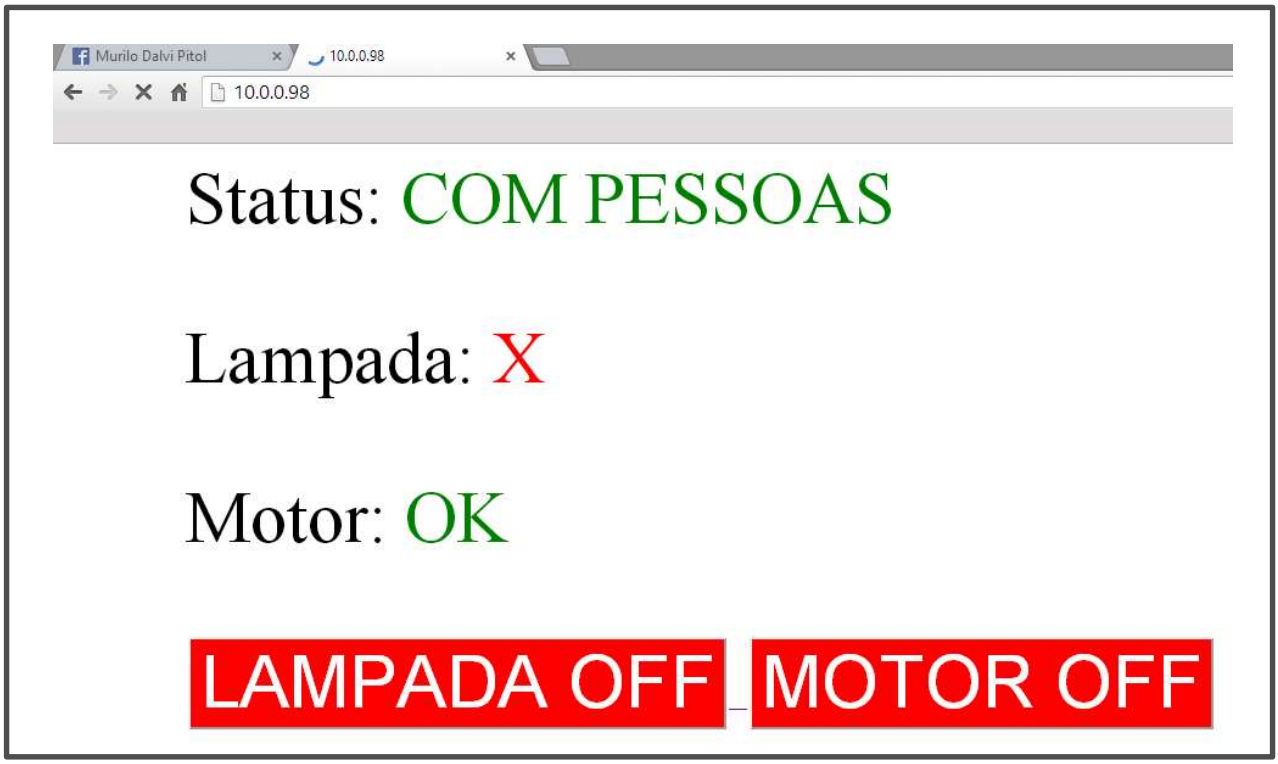
3.10 Código Arduino e Web Server

Como mencionado anteriormente, toda a programação envolvida neste projeto foi desenvolvida dentro do IDE do *Arduino*, tanto a parte referente à lógica das entradas e saídas quanto o código HTML que compõe o *site*.

A página foi programada de modo a ser atualizada a cada 1 segundo para que o programa possa obter os valores das variáveis referentes aos sensores e atualizar

a interface do *site* quando necessário. No *site* é exibido se há ou não pessoas presentes no ambiente, se os aparelhos estão ou não conectados a tomada, se os equipamentos estão ligados ou desligados e um botão que permite acionar cada um dos aparelhos, como mostra a Figura 34.

Figura 34: Interface do *site*



Fonte: Produção do próprio autor

O código completo desenvolvido está contido no Apêndice.

3.11 Acesso remoto

Nessa sessão serão descritos os procedimentos usados para realizar o acionamento remoto dos equipamentos.

3.11.1 Acesso via LAN

Uma LAN é uma rede na qual computadores e dispositivos móveis estão conectados de forma restrita, não permitindo assim nenhum tipo de acesso externo. Em suma, é extritamente necessário que o *Arduino* e o dispositivo que emitirá o comando estejam na mesma rede local.

Quando conecta-se o *Ethernet Shield* ao roteador é possível atribuir ao *Arduino*, via *software*, um endereço IP. Pode-se escolher arbitrariamente qualquer número que pertença a faixa de IPs válidos para *hosts* desta rede local, desde que o endereço escolhido não esteja sendo usado por outro dispositivo. Para verificar o número de *hosts* permitidos na Intranet deve-se acessar as configurações do roteador e verificar a máscara de sub-rede, alterando-a caso seja necessário.

Logo, para acessar o *site* desenvolvido basta digitar o endereço IP do *Arduino* em qualquer navegador de Internet instalado no computador, *tablet* ou *smartphone.*

3.11.2 Acesso via Internet

Uma alternativa para o acesso remoto pela Internet é por meio do *software TeamViewer*. Basicamente a idéia seria acessar remotamente um computador que pertença a mesma LAN na qual o *Arduino* está inserido e dessa forma acessar a página *web* ao digitar o IP do *Arduino* no navegador.

Um problema com o *TeamViewer* é o fato de a senha mudar a cada conexão. Uma forma de prevenir isso é criando uma conta e vinculando-a a um computador. Para tal, basta pressionar a opção “Inscreva-se” e definir uma senha como mostrado na Figura 35.

Figura 35: Etapas da criação de uma conta no TeamViewer

Fonte: Produção do próprio autor

A partir de agora haverá um computador vinculado a esta conta no *TeamViewer*, o que garantirá que a senha de acesso não mudará caso o computador seja reiniciado automaticamente, como comumente acontece nas atualizações do sistema operacional, por exemplo. Assim é possível aumentar a confiabilidade do método que está sendo utilizado.

4 RESULTADOS E DISCUSSÕES

Os resultados obtidos foram demonstrados através do funcionamento do protótipo. Naturalmente, as soluções aqui apresentadas são extensíveis, caso esse seja o intuito, para mais de uma lâmpada e para quaisquer equipamentos dotados de um motor. A lógica de funcionamento do programa permite controlar quantos aparelhos se desejar, basta que seja feito o uso de um *Arduino* que possua um número maior de portas, mais memória para armazenamento do programa e melhor capacidade de processamento. O protótipo foi desenvolvido para possibilitar a aplicação desta solução em uma sala ou cômodo real, mas o foco principal deste trabalho consistia apenas no desenvolvimento de uma estrategia para o acionamento remoto dos equipamentos em si, e não sua real implementação na prática.

Referente ao sensoriamento da lâmpada, este processo também poderia ter sido feito através de um sensor de corrente, mas o uso do LDR se mostrou como uma solução muito mais interessante pelo fato de ser extremamente barata e apresentar complexidade de implementação quase nula. Trata-se de um circuito divisor de tensão muito simples e que apresentou uma boa confiabilidade e precisão nos testes efetuados, não deixando nada a desejar em comparação ao sensoriamento do motor efetuado pelo TC.

Outro ponto que merece destaque é quanto ao funcionamento do programa quando o acionamento é feito pelo *click* do botão pelo *site*. Por exemplo, considerando a lógica de programação desenvolvida, o programa deveria saltar do estado 00 direto para 11 e vice-versa. Porém quando estava sendo executado na prática, antes de chegar ao estado final, o programa passava por um estado intermediário. A tabela 1 mostra o estado intermediário de cada transição.

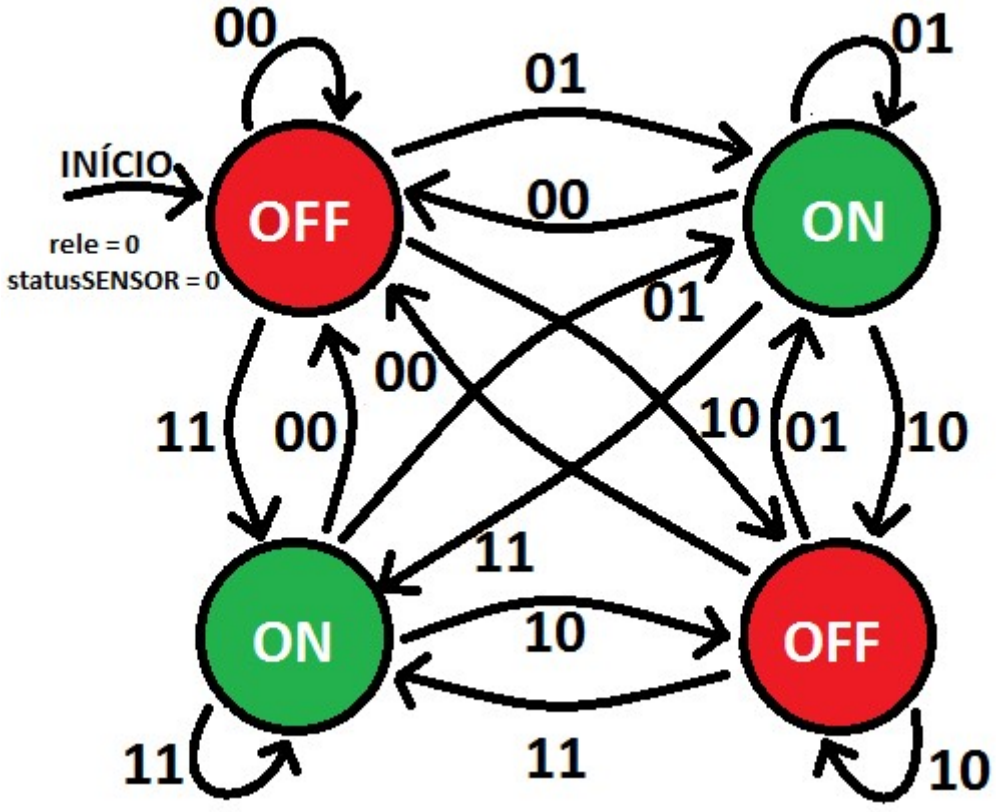
Tabela 1: Estados intermediários entre as transições de estado

ESTADO INICIAL	ESTADO INTERMEDIÁRIO	ESTADO FINAL
00	10	11
01	11	10
10	00	01
11	01	00

Fonte: Produção do próprio autor

Levando em consideração os estados intermediários, a máquina de estados completa do sistema fica conforme mostrado na figura abaixo.

Figura 36: Máquina de estados com estados intermediários



Fonte: Produção do próprio autor

Esse fenômeno pode ser facilmente explicado pelo fato de “rele” ser uma variável controlada puramente via *software*, enquanto “statusSENSOR” precisa que haja o chaveamento do relé e que o equipamento entre em funcionamento para só

então ter seu valor alterado. Naturalmente, a variável “statusSENSOR” demora um tempo muito maior para ser atualizada por ser um processo físico e ter uma dinâmica bem mais lenta. Uma solução para este problema foi inserir um *delay* no programa para dar tempo do equipamento ser ligado ou desligado e o sensor ser atualizado corretamente antes da decisão pela condicional ser efetuada. Após colocado o *delay* no programa, o sistema passou a se comportar conforme a máquina de estados exibida na Figura 33.

Quanto ao *site*, como já mencionado, este fornece as seguintes informações para o usuário: Se o ambiente está sendo usado ou não, se a lâmpada e o ventilador estão ou não ligados à tomada, o status dos equipamentos (ligado ou desligado) e um botão para realizar a mudança do status de cada aparelho. Considerando a quantidade extremamente limitada de memória do *Arduino Uno*, levando em conta as atualizações constantes da página e o atraso inerente ao sistema, foi desenvolvido uma interface leve com apenas as informações julgadas cruciais para o usuário, sem o uso de figuras, animações ou outros recursos que pudessem atrasar a dinâmica e assim reduzir o desempenho e velocidade do *site*.

Quanto ao controle remoto, considerando o perfeito funcionamento dos sensores, existem alguns possíveis motivos que impossibilitam o seu correto funcionamento: A falta de energia elétrica ou conexão com a Internet no recinto, o equipamento ter sido removido da tomada ou a presença de algum defeito no aparelho.

Como estão situados na mesma instalação elétrica, uma queda de energia elétrica resultaria no desligamento tanto dos aparelhos quanto do *Arduino* e *Ethernet Shield*, o que impossibilitaria o acesso a página. A ausência de Internet no ambiente também deixaria o *site* em modo *offline*. Caso haja sucesso em acessar o *site*, isso confirma a existência de energia elétrica e conexão com a rede no ambiente. Quanto ao equipamento ter sido removido da tomada, esta possibilidade foi tratada como especificado nas sessões 3.6 e 3.7. Por último, não é possível saber com precisão se houve ou não uma queima da lâmpada ou defeito no ventilador, pois estes equipamentos não são dotados de uma inteligência ou sensoriamento que informe isso ao usuário. Porém, se o *site* estiver *online* e o status do aparelho estiver como ligado à tomada, e ao efetuar o *click* no *site* o status não mudar, isso é um forte

indício de que há algum defeito no equipamento cujo acionamento foi tentado. Dessa forma foi possível cercar e tratar possíveis erros que possam eventualmente surgir no processo de acionamento a distância.

Relacionado ao acionamento remoto externo a Intranet, a ideia inicial baseava-se na técnica de redirecionamento de portas no roteador com a construção de um domínio dinâmico atrelado ao IP do *Arduino.* Seria uma solução mais rebuscada, mas a dificuldade em ter acesso às portas TCP/IP e a própria natureza do código desenvolvido foram entraves para o desenvolvimento deste método, uma vez que o código foi programado para alterar o *link* exposto no *browser* toda vez que um *click* é efetuado pelo *site*. Assim, em tese seria possível acessar a página inicial através do domínio registrado, mas quando algum comando fosse dado a string do *site* mudaria para algo que não faria sentido para o serviço provedor de domínio, o que geraria um erro na página. A solução via *TeamViewer*, apesar de não ser tão sofisticada como a idéia inicial, se mostrou satisfatória para cumprir o que foi proposto, permitindo o acesso remoto dos equipamentos de forma eficaz, rápida e livre de maiores complicações ou empecilhos.

5 CONCLUSÃO

Analisando os resultados obtidos, pode-se concluir que o projeto atendeu seus objetivos gerais e específicos com sucesso. Mesmo com as constantes mudanças de estratégias devido a dificuldades encontradas no caminho, ao fim foi possível efetuar o controle remoto dos aparelhos com boa rapidez, confiabilidade e eficiência.

Apesar de suas limitações quanto a capacidade de processamento e principalmente de memória, o *Arduino Uno* se mostrou uma ferramenta versátil, de baixo custo, fácil implementação e possuindo uma gama muito grande de aplicações possíveis. Não é preciso ser detentor de um conhecimento profundo em

programação para sua utilização, o que foi um ponto muito forte e de grande auxílio no desenvolvimento desse projeto como um todo.

No mais, o ramo de automação residencial, ou domótica, vem apresentando uma crescente evolução e a tendência é que mais soluções desse tipo surjam com o passar dos anos. Portanto este trabalho está inserido em um contexto atual no qual o ser humano busca na tecnologia uma forma de obter mais comodidade, conforto e economia.

5.1 Dificuldades encontradas

Uma série de dificuldades foram encontradas ao decorrer do percurso. O primeiro grande problema foi a memória do *Arduino Uno*, que conta somente com 2KB de memória dinâmica. A linguagem HTML demanda muita memória, de modo que algumas linhas de código já eram o suficiente para atingir o limite máximo. Esse também foi um dos motivos para a interface do *site* ser a mais básica possível. *O Arduino Uno* possui *slot* para cartão de memória, porém foram testados 3 cartões diferentes na tentativa de salvar os códigos em seu interior, mas todos eles apresentaram funcionamento inadequado. Apenas o simples fato de inserir o cartão no *slot* já era o suficiente para fazer o *site* apresentar *bugs* inexplicáveis, de forma que essa estratégia teve que ser abandonada.

Outro problema foi a dificuldade em ter acesso as portas TCP/IP, devido a problemas com *Firewalls*, tanto o do sistema operacional quanto o do roteador, além do fato dos provedores de Internet bloquearem o acesso a determinadas portas. Também vale ressaltar que a natureza do funcionamento do programa desenvolvido, como explicado anteriormente, também impossibilitou a implementação do método de acionamento remoto via Internet inicialmente almejado.

5.2 Trabalhos Futuros

Para trabalhos futuros, levando em consideração que o foco foi o desenvolvimento de um protótipo, sugere-se a implementação do projeto em um ambiente real. Para tal, é aconselhável a utilização de um *Arduino* com maior capacidade de processamento, mais memória e com mais portas de entrada e saída, tal como um *Arduino MEGA.* Assim, será possível o controle de mais equipamentos e o desenvolvimento de uma interface mais elaborada para o *site*, sem a necessidade de alocar o código HTML em cartões de memória ou em outros recursos externos ao *Arduino*. Talvez seja recomendável a utilização de um *Wi-fi Shield* no lugar do *Ethernet Shield,* no qual a conexão se dá via *Wireless,* não havendo assim a necessidade de conexão diretamente com o cabo de rede. Se as dimensões do ambiente que almeja-se automatizar forem muito grandes e o *Arduino* necessitar ficar muito longe do roteador, a conexão via cabo de rede pode ser inviável, tornando a conexão sem fio uma solução mais eficiente.

Também há a possibilidade de se utilizar outras plataformas de *hardware* e *software*, como por exemplo a *RaspBerryPi*, que é uma ferramenta mais poderosa que o *Arduino*. Por outro lado, isso demandaria maiores habilidades de programação e conhecimentos em *Linux* por parte do programador.

Sugere-se também o desenvolvimento de um método mais sofisticado para efetuar o acesso remoto por meio externo a Intranet.

REFERÊNCIAS

SISLITE. **O que é a domótica?.** 2015. Disponível em: < http://www.sislite.pt/domus.htm> Acesso em 14 mar. 2015.

BASCONCELLO, Daniel. Robotizando, **Curso de Arduino.** 2012. Disponível em: < http://www.robotizando.com.br/curso_arduino_o_que_e_arduino_pg1.php >. Acesso em 29 mar.2015

ARDUINO-1. **Arduino Uno.** Disponível em:<http://arduino.cc/en/Main/ArduinoBoardUno> Acesso em 29 mar. 2015

ARDUINO-2. **Arduino Ethernet Shield**. Disponível em: <http://arduino.cc/en/Main/ArduinoEthernetShield> Acesso em 29 mar. 2015

TECMUNDO-1, 2015. **O que é TCP/IP?**. Disponível em:<http://www.tecmundo.com.br/o-que-e/780-o-que-e-tcp-ip-.html> Acesso em 11 abr. 2015

TECMUNDO-2, 2015. **O que é roteador?.** Disponível em: http://www.tecmundo.com.br/conexao/1258-o-que-e-roteador-.htm Acesso em 15 abr. 2015

TEAMVIEWER, 2015. **Apresentação.** Disponível em: <https://www.teamviewer.com/pt/index.aspx> Acesso em 16 out. 2015

BEGHINI, L. B. **Automação residencial de baixo custo por meio de Dispositivos móveis com sistema operacional Android.** 2015. Trabalho de Conclusão de Curso, Universidade de São Paulo, São Carlos, 2013

FILIPEFLOP, 2015. **Sensor de Movimento Presença PIR**. Disponível em: < http://www.filipeflop.com/pd-6b901-sensor-de-movimento-presenca-pir.html>. Acesso em 3 mai. 2015

MERCADOLIVRE, 2015. **Sensor de Tensão Ac 127v/220v – Automação – Arduino.** Disponível em: < http://produto.mercadolivre.com.br/MLB-690247235-sensor-de-tenso-ac-127v220v-automaco-arduino-_JM>. Acesso em 3 nov. 2015
BIONDO, R. M. **Domótica: Sistemas e Aplicabilidade. 2015.** Trabalho de Conclusão de Curso, Universidade de São Paulo, São Carlos, 2011

APÊNDICE

```
#include <SPI.h>
#include <Ethernet.h>
#include "EmonLib.h"

#define tensaoLampada 0
#define tensaoMotor 1
#define rele_lampada 2
#define LDR 3
#define sensor 4
#define rele_motor  5
#define PIR 9

int statusPIR;
int statusLDR;
int statusTC;
int rele1 = 0;
int rele2 = 0;
int tomadaLampada;
int tomadaMotor;
float tensaoLDR;
float correnteTC;
byte mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0x9B, 0x36 };

IPAddress ip(10, 0, 0, 98);
EthernetServer server(80);
EnergyMonitor emon1;    // cria uma instancia

void setup()
{
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.begin(9600);
  pinMode(rele_lampada, OUTPUT);
  digitalWrite(rele_lampada, LOW);
  pinMode(rele_motor, OUTPUT);
  digitalWrite(rele_motor, LOW);
  emon1.current(sensor, 10000);
}


void loop()
```

```
{
  // listen for incoming clients
  EthernetClient client = server.available(); // recebe clientes
  if (client){
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    String str ; //"10.0.0.98";

    while (client.connected())
    {
      if (client.available())
      {
        char c = client.read();
        str.concat(c);

        // --------altera valores de rele1 e rele2 ----------

        if (str.endsWith("1p"))
        {
          rele1 = 1;
          digitalWrite(rele_lampada, HIGH);
          delay (500);
        }

        if (str.endsWith ("1g"))
        {
          rele1 = 0;
          digitalWrite(rele_lampada, LOW);
          delay (500);
        }

        if (str.endsWith("2p"))
        {
          rele2 = 1;
          digitalWrite(rele_motor, HIGH);
          delay(5);
        }

        if (str.endsWith("2g")) {
          rele2 = 0;
          digitalWrite(rele_motor, LOW);
          delay(5);
        }
```

```
        //-------------------------------------------------------

        if (c == '\n' && currentLineIsBlank)
        {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html"); //inicializa o arquivo html
          client.println();
          client.println("<META HTTP-EQUIV=REFRESH CONTENT=1.0 URL=>"); // atualiza a pagina a cada 1
segundo
          client.println();
          client.println("<UL>"); // iniciar lista não numerada
          client.println("<UL>"); // iniciar lista não numerada
          client.println("<font SIZE=30>"); //define fonte tamanho 30
          client.println("Status:");

          //------------------------- sensor PIR -------------------------
          pinMode(PIR, INPUT);
          statusPIR = digitalRead(PIR);
          if (statusPIR == HIGH)
            client.println("<font color=green  >COM PESSOAS");
          else
            client.println("<font color=red >SEM PESSOAS");
          //-----------------------------------------------------------
          client.println("<p>");
          client.println("<font color=black >Lampada:");

          //---------------- sensor de tensão da Lampada --------------------
          pinMode(tensaoLampada, INPUT);
          tomadaLampada = analogRead(tensaoLampada);
          if (tomadaLampada > 250)
            client.println("<font color=green > OK");
          else
            client.println("<font color=red > X");
          //-----------------------------------------------------------------
          client.println("<p>");
          client.println("<font color=black >Motor:");

          //-------------sensor de tensão do motor --------------------------

          pinMode(tensaoMotor, INPUT);
          tomadaMotor = analogRead(tensaoMotor);
          if (tomadaMotor > 250)
            client.println("<font color=green>OK");
```

```
        else
          client.println("<font color=red>X");
        //---------------------------------------------------------------
        client.println("<p>");

        //------------ leitura do LDR, definindo valor de SstatusLDR----------
        tensaoLDR = analogRead(LDR);
        if (tensaoLDR > 500)
          statusLDR = 0;
        else
          statusLDR = 1;
        //---------------------------------------------------------------
        //--------------------------------------------------------------- LAMPADA----------------------------------------------------------
        //ESTADO 00
        if (rele1 == 0 && statusLDR == 0)
          client.println("<a href=http://10.0.0.98/1p/><button style = color:white;background-color:red><font size =
50>LAMPADA OFF</button>");

        //ESTADO 01
       if (rele1 == 0 && statusLDR == 1)
          client.println("<a href=http://10.0.0.98/1p/><button style =color:white;background-color:green><font size =
50>LAMPADA ON</button>");

        //ESTADO 10
        if (rele1 == 1 && statusLDR == 0)
          client.println("<a href=http://10.0.0.98/1g/><button style =color:white;background-color:red><font size =
50>LAMPADA OFF</button>");

        //ESTADO 11
        if (rele1 == 1 && statusLDR == 1)
          client.println("<a href=http://10.0.0.98/1g/><button style =color:white;background-color:green><font size =
50>LAMPADA ON</button>");
        //---------------------------------------------------------------------------------------------------------------------------------
        //---------------- definindo o valor de statusTC--------------------------------
        correnteTC = emon1.calcIrms(1480);
        // client.println(correnteTC);
        if (correnteTC < 70)
          statusTC = 0;
        else
          statusTC = 1;
        //-------------------------------------------------------------------------------
```

```
        //------------------------------------------------------------------ MOTOR ----------------------------------------------------------------
        //ESTADO 00
        if (rele2 == 0 && statusTC == 0)
          client.println("<a href=http://10.0.0.98/2p/><button style =color:white;background-color:red><font size = 50>MOTOR OFF</button>");

        //ESTADO 01
        if (rele2 == 0 && statusTC == 1)
          client.println("<a href=http://10.0.0.98/2p/><button style =color:white;background-color:green><font size = 50>MOTOR ON</button>");


        //ESTADO 10
        if (rele2 == 1 && statusTC == 0)
          client.println("<a href=http://10.0.0.98/2g/><button style =color:white;background-color:red><font size = 50>MOTOR OFF</button>");

        //ESTADO 11
        if (rele2 == 1 && statusTC == 1)
          client.println("<a href=http://10.0.0.98/2g/><button style =color:white;background-color:green><font size = 50>MOTOR ON</button>");
       //-------------------------------------------------------------------------------------------------------------------------------------------
       client.println("</UL>");
       break;
     }
     if (c == '\n')
     {
       currentLineIsBlank = true;
     }
     else if (c != '\r')
     {
       currentLineIsBlank = false;
     }
    }
  }
  delay(1); // tempo para o browser receber a informação
  client.stop(); // encerra a conexão
}
}
```