ESCOLA SUPERIOR ABERTA DO BRASIL – ESAB
CURSO DE PÓS-GRADUAÇÃO LATU SENSU EM SISTEMAS DE TELECOMUNICAÇÕES

GILMAR SILVESTRE DA CRUZ SILVA

ACIONAMENTO REMOTO DE MOTORES ELÉTRICOS TRIFÁSICOS DE INDUÇÃO: utilização da plataforma microcontrolada arduino e comando via celular com interfaceamento wi-fi

VILA VELHA (ES)
2015

GILMAR SILVESTRE DA CRUZ SILVA

ACIONAMENTO REMOTO DE MOTORES ELÉTRICOS TRIFÁSICOS DE INDUÇÃO: utilização da plataforma microcontrolada arduino e comando via celular com interfaceamento wi-fi

Monografia apresentada ao curso de Pós-Graduação Latu Sensu em Sistemas de Telecomunicações, da Escola Superior Aberta do Brasil, como requisito parcial à obtenção do título de Especialista em Sistemas de Telecomunicações.

Orientadora:
Prof.ª Ma. Janaina Costa Binda

VILA VELHA (ES)
2015

GILMAR SILVESTRE DA CRUZ SILVA

ACIONAMENTO REMOTO DE MOTORES ELÉTRICOS TRIFÁSICOS DE INDUÇÃO: utilização da plataforma microcontrolada arduino e comando via celular com interfaceamento wi-fi

Esta monografia foi julgada adequada à obtenção do título de Especialista em Sistemas de Telecomunicações, sendo aprovada na sua forma final pelo curso de Pós-Graduação Latu-Sensu em Sistemas de Telecomunicações da Universidade Aberta do Brasil (ESAB).

BANCA EXAMINADORA

Prof.ª Ma. Janaína Costa Binda
Orientadora
Escola Superior Aberta do Brasil – ESAB

Prof. 02 <Título>. <Nome Completo>
<Instituição>

Prof. 03 <Título>. <Nome Completo>
<Instituição>

Vila Velha (ES), 11 de fevereiro de 2015.

A Deus, pelo dom da vida.
Aos meus pais: Sr. Geraldo Silvestre Silva e Da. Eugênia da Cruz Silva.
Às minhas irmãs: Vanda Maria da Silva e Luciene Maria da Silva.
Ao amor da minha vida: minha esposa, Felina Santos.

AGRADECIMENTOS

Ao Deus Pai, ao Deus Filho e ao Deus Espírito Santo;

Aos meus queridos e amados pais, Sr. Geraldo Silvestre Silva e Da. Eugênia da Cruz Silva, pelo amor, carinho e dedicação tão necessários à minha formação moral como cidadão e ser humano;

À minha esposa, Felina Santos, pelo seu companheirismo, carinho, compreensão, paciência, amor incondicional e por dividir comigo todos os nossos bons e maus momentos.

À minha orientadora, Prof.ª Ma. Janaína Costa Binda, pelo empenho e dedicação nas muitas horas a fio com as revisões, críticas e sugestões.

Ao Instituto Federal de Sergipe (IFS), pela disponibilização dos equipamentos de acionamentos e comandos elétricos industriais, do MIT utilizado nesse projeto e do seu Laboratório de Instalações Elétricas Industriais, viabilizando, com isso, a realização do presente trabalho.

"Semeia um pensamento e colherás um desejo; semeia um desejo e colherás a ação; semeia a ação e colherás um hábito; semeia o hábito e colherás o caráter".

Tihamer Toth

“O que anda com os sábios se tornará sábio, mas o companheiro dos tolos será destruído.”

Provérbios 13:20

RESUMO

Esse trabalho é o resultado de uma pesquisa que busca aliar as tecnologias Wireless de transmissão de dados aos métodos de acionamentos e comandos de motores elétricos trifásicos de indução. Vários sistemas de hardware e software foram empregados na implantação de um projeto de acionamento de motores elétricos sem requerer a presença física do operador da máquina no local onde se encontra o centro de comando da mesma. O projeto aqui apresentado foi executado utilizando a plataforma microcontrolada Arduino para gerenciar as variáveis elétricas e de dados dos circuitos componentes, um motor elétrico trifásico de indução, um aparelho celular operando com sistema operacional Android e dotado de tecnologia Wi-Fi para comandar o sistema remotamente, um circuito de acoplamento para fazer a ponte entre os estágios de Wi-Fi/controle e de controle/potência, além de outros elementos tipicamente usados nos processos de acionamentos elétricos descritos ao longo do presente trabalho. O software controlador foi implementado com a linguagem C nativa do Arduino e o programa de acionamento residente na memória do aparelho celular foi construído utilizando a plataforma de programação em blocos App Inventor. O projeto proposto foi construído com sucesso e os testes realizados mostraram que o sistema funcionou eficientemente bem, ligando e desligando motores elétricos apenas ativando os botões LIGAR/DESLIGAR do software instalado no dispositivo móvel, utilizando a tecnologia Wi-Fi como interface de comunicação entre o mesmo e o sistema controlador.

PALAVRAS-CHAVE: Automação Industrial. Arduino. Redes Wi-Fi. App Inventor. Motores Trifásicos de Indução.

ABSTRACT

This work is the result of a research that seeks to combine the Wire-less technologies of data transmission methods of drives and controls three-phase electric induction motors. Various hardware systems and software were used in the implementation of a driving design of electric motors without requiring the physical presence of the machine operator on site where the same command center. The project presented here was performed using the Arduino platform microcontrolled to manage the electrical variables and component data circuits, a three-phase electric induction motor, a mobile device operating with Android operating system and equipped with Wi-Fi technology to control the system remotely a coupling circuit to bridge the gap between Wi-Fi stages/control and control/power, and other elements typically used in electric drives processes described throughout this work. The driver software was implemented with the native C language of the Arduino and the resident drive program in the mobile device memory is built using the programming platform in App Inventor blocks. The proposed project has been successfully built and tests showed that the system worked efficiently and on and off electric motors only activating the software ON/OFF buttons installed on the mobile device using Wi-Fi technology as a communication interface between the and even the controller system.

KEYWORDS: Industrial Automation. Arduino. Wi-Fi Networks. App Inventor. Three-Phase Induction Motors.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIAÇÕES E SIGLAS

AAC - *Advanced Audio Coding* (Codificação Avançada de Áudio)
AC - *Alternating Current* (Corrente Alternada)
ADSL - *Asymmetric Digital Subscriber Line* (Linha de Transmissão Digital Assíncrona)
ADT - *Android Development Tools* (Ferramenta de Desenvolvimento Android)
AES - *Advanced Encyptation Standart* (Padrão de Criptografia Avançada)
AGVs - *Automatic Guided Vehicles* (Veículos Automaticamente Guiados)
Anatel - Agência Nacional de Telecomunicações
AP - *Access Point* (Ponto de Acesso)
ARP - *Address Resolution Protocol* (Protocolo de Resoluções de Endereços)
ART - *Android RunTime*
ASCII - *American Standard Code for Information Interchange* (Código Padrão Americano para o Intercâmbio de Informação)
BSS - *Basic Service Set* (Estrutura de Serviços Básicos)
BSSID - *Basic Service Set Identification* (Estrutura de Serviços Básicos de Identificação)
CCMP - *Counter-Mode/Cipher Block Chaining Message Authentication Code Protocol* (Protocolo de Autenticação e Encadeamento de Código Cifrado de Blocos de Mensagens)
CPF - Cadastro de Pessoas Físicas
DC - *Direct Current* (Corrente Contínua)
DHCP - *Dynamic Host Configuration Protocol* (Protocolo de Configuração Dinâmica de Host)
DNS - *Domain Name System* (Sistema de Nomes de Domínios)
EEPROM - *Electrically Erasable Programmable Read-Only Memory* (Memória Somente de Leitura Eletricamente Programável)
ESAB - Universidade Aberta do Brasil
ESS - *Extended Service Set* (Estrutura de Serviços Estendidos)
ESSID - *Extended Service Set Identification* (Estrutura de Serviços de Identificação Estendidos)
FTP - *File Transfer Protocol* (Protocolo de Transferência de Arquivos)
GND - *Ground* (Fio Neutro)
GPS - *Global Positioning System* (Sistema de Posicionamento Global)
HD - *Hard Disk* (Disco Rígido)
HTML - *HyperText Markup Language* (Linguagem de Marcação de Hipertexto)
HTTP - *HyperText Transfer Protocol* (Protocolo de Transferência de Hipertexto)
IBSS - *Independent Basic Service Sets* (Estrutura de Serviços Básicos Independentes)
ICMP - *Internet Control Message Protocol* (Protocolo de Controle de Mensagens de Internet)
ICSP - *In-Circuit Serial Programming* (Protocolo de Programação Serial Com Dispositivo Montado no Circuito)
IDE - *Integrated Development Environment* (Ambiente Integrado de Desenvolvimento)
IEEE - *Institute of Electrical and Electronics Engineers* (Instituto dos Engenheiros Eletricistas e Eletrônicos)

IGMP - *Internet Group Management Protocol* (Protocolo de Gerenciamento de Grupos de Internet)
ISO - *International Organization for Standardization* (Organização Internacional para Padronização)
JDK - *Java Development Kit* (Kit de Desenvolvimento Java)
JVM - *Java Virtual Machine* (Máquina Virtual Java)
LAN - *Local Area Networks* (Redes Locais)
LED - *Light Emitting Diode* (Diodo Emissor de Luz)
LTS - *Long Term Support* (Suporte de Longo Prazo)
MIC - *Message Integrity Chec* (Cheque de Integridade de Mensagem)
MIME - *Multipurpose Internet Mail Extensions* (Extensões Multifunções para Mensagens de Internet)
MIT - Motor Trifásico de Indução
MMS - *Multimedia Messaging Service* (Serviço de Mensagem Multimídia)
MPEG - *Moving Picture Experts Group* (Grupo de Especialistas em Imagens com Movimento)
MTBF - *Mean Time Between Failures* (Tempo Médio Entre Falhas)
MTBR - *Mean Time Between Repair* (Tempo Médio Entre Reparo)
NAT - *Network Address Translation* (Tradução de Endereço de Rede)
NetBIOS - *Network Basic Input/Output System* (Sistema Básico de Rede de Entrada/Saída)
NIST - *National Institute of Standards and Technology* (Instituto Nacional de Padrões e Tecnologia)
OHA - *Open Handset Alliance*
OSI - *Open Systems Interconnection* (Interconexão de Sistemas Abertos)
PDF - *Portable Document Format* (Formato de Documento Portátil)
PWM - *Pulse Width Modulation* (Modulação por Largura de Pulso)
RAM - *Random Access Memory* (Memória de Acesso Aleatório)
RFC - *Request for Comments* (Pedido Para Comentários)
ROM - *Read Only Memory* (Memória Apenas de Leitura)
SDK - *Software Development Kit* (Kit de Desenvolvimento de Software)
SMS - *Short Message Service* (Serviço de Mensagens Curtas)
SPI - *Serial Peripheral Interface* (Interface de Periféricos Seriais)
SRAM - *Static Random Access Memory* (Memória Estática de Acesso Aleatório)
TCP/IP - *Transmission Control Protocol/Internet Protocol* (Protocolo de Controle de Transmissão/Protocolo de Internet)
TKIP - *Temporal Key Integrity Protocol* (Protocolo de Chave de Integridade Temporal
TTL - *Transistor-Transistor Logic* (Lógica Transistor-Transistor)
UART - *Universal Asynchronous Receiver/Transmitter* (Transmissor e Receptor Universal Assíncrono)
UDP - *User Datagram Protocol* (Protocolo de Transporte de Datagramas)
URL - *Uniform Resource Locator* (Localizador Padrão de Recursos)
USB - *Universal Serial Bus* (Barramento Serial Universal)
VGA - *Video Graphics Array* (Padrão de Disposição Gráfica Para Vídeo)
WDS - *Wireless Distribution System* (Sistema de Distribuição Sem-Fio)
WEP - *Wired Equivalent Privacy* (Segurança Equivalente às Redes Com Fio)
Wi-Fi - *Wireless Fidelity* (fidelidade sem fio)
WLAN - *Wireless Local Area Networks* (Redes Locais Sem-Fios)
WPA - *Wi-Fi Protected Access* (Proteção do Acesso Wi-Fi)

SUMÁRIO

1. INTRODUÇÃO

A maioria dos equipamentos, máquinas e sistemas da atualidade, segundo Carvalho (2011), tem como força motora uma máquina trifásica de indução (MIT). Essas máquinas têm suas partidas e paradas comandadas por diversos dispositivos, cujas funções vão desde o simples ligar e desligar, ao controle da sua corrente de partida e à regulação da velocidade de giro dos seus rotores. Por ser um motor de fácil partida, menos ruído e ser mais barato que as máquinas de indução monofásicas projetadas para trabalharem em potências superiores a 2 kW, são preferidos, de acordo com Franchi (2008), nos ambientes industriais e domésticos.

Por outro lado, um dos pontos mais importantes da existência das redes de comunicação está relacionado ao aumento na confiabilidade do sistema como um todo. A redução de custos é outra questão não menos essencial para a utilização das redes de comunicação, uma vez que computadores de pequeno porte apresentam menor relação preço/desempenho que os grandes. Dessa forma, sistemas que utilizariam apenas uma máquina de grande porte e de custo muito elevado podem ser concebidos à base da utilização de grande número de microcomputadores.

Usar algum tipo de cabo, de acordo com Morimoto (2008), seja de par trançado ou fibra ótica, é a forma mais fácil de transmitir dados. Entretanto, ele segue afirmando que o grande problema em utilizar cabos na implementação de redes de comunicação é o crescimento do custo do cabeamento à medida que novos clientes são a ela adicionados, além da distância a cobrir. Dessa forma, uma rede cabeada oferece pouca flexibilidade visto que, caso haja necessidade de adicionar novas estações à rede, ou qualquer outra alteração que não esteja prevista no projeto original da mesma, irá necessitar de que sejam feitas novas alterações no sistema de cabeamento de tal topologia, ainda que seja mínimo.

Com esse pensamento de Morimoto (2008) concorda Stemmer (2001) ao afirmar que, nas redes sem fio, os pacotes são transmitidos através de canais de freqüência de rádio ou infravermelho. Para Stemmer (2001), as redes sem fio são uma boa alternativa para aplicações onde é difícil instalar cabos metálicos ou de fibra ótica. Seu

emprego é especialmente importante para comunicações entre computadores portáteis em um ambiente de rede local móvel ou onde a confiabilidade do meio de transmissão é requisito indispensável (por exemplo, onde o rompimento de um cabo pode paralisar todo o sistema). Este tipo de rede vem ganhando certa aceitação em aplicações de chão de fábrica, devido à flexibilidade de instalação e operação, pois não há necessidade de projeto de layout, nem canaletes para cabos de rede. Equipamentos móveis inteligentes do chão de fábrica podem ser configurados como estações de rede sem fio. Exemplos deste tipo de equipamento são os AGVs (Automatic Guided Vehicles), robôs autônomos e sensores inteligentes.

Outro componente igualmente importante para este trabalho é o sistema operacional móvel Android que, na visão de Lecheta (2010), é uma plataforma de código aberto criada pela empresa multinacional Google Inc. em 2007, com intuito de desenvolver aplicações para dispositivos móveis e tendo seu kernel[1] baseado no sistema operacional Linux. Essa plataforma é mantida pela *Open Handset Alliance* (OHA), um grupo formado por mais de quarenta empresas para inovar e acelerar o desenvolvimento de aplicações e serviços, trazendo aos consumidores experiência mais rica em termos de recursos e menos dispendiosa em termos financeiros para o mercado móvel.

A opção pela utilização da plataforma Arduino se deve ao fato de ser, conforme McRoberts (2011), fácil de utilizar e permitir a criação de programas em intervalo de tempo relativamente curto. Assim, a procura por novos métodos no âmbito industrial que busquem reduzir o tempo e a distância na tomada de decisões, bem como, ligar ou desligar motores sem a dependência da presença física de funcionários da empresa nas proximidades dos locais de acionamento das máquinas, está fundamentado nos propósitos do uso da automação industrial em qualquer processo a que ela é destinada.

Pelo exposto, infere-se que tais sistemas têm por premissa a redução de custos e a consequente elevação da qualidade de funcionamento dos sistemas de controle de

[1] Parte do sistema operacional responsável por ocultar o hardware da máquina sob uma interface de programação abstrata e de alto nível, fornecendo os recursos dos quais os programas necessitam para procederem à interação entre o usuário e os dispositivos físicos (NEMETH et al., 2004, p. 166.).

processos industriais. Dessa forma o presente trabalho apresenta e discute uma solução para o seguinte problema: como acionar máquinas elétricas remotamente utilizando a tecnologia Wi-Fi e sistemas microcontrolados? Tal tecnologia é viável do ponto de vista do ambiente industrial?

O problema acima será discutido levando-se em consideração que a procura por métodos, no âmbito industrial, que busquem reduzir o tempo e a distância na tomada de decisões, bem como, acionar ou desligar motores sem a dependência da presença física de funcionários da empresa nas proximidades dos locais de acionamento das máquinas, está fundamentado nos propósitos do uso da automação industrial em qualquer processo a que ela é destinada, reduzindo custos e elevando a qualidade de funcionamento dos sistemas de controle de processos industriais.

Um dos principais motivos, além do baixo custo de aquisição e manutenção do MIT (comparativamente aos outros tipos de motores), que o faz ser a máquina mais utilizada em tais ambientes é porque os atuais sistemas de distribuição de energia elétrica são trifásicos e de corrente alternada. Esse trabalho os tratará com a ênfase maior, já que na prática, além de constituírem o grande leque dos motores de indução polifásicos, seus princípios construtivos e de funcionamento são de maior conhecimento pelo autor do presente trabalho.

Diante destas circunstâncias, será desenvolvido e testado um projeto com o intuito de realizar o controle do acionamento à distância de motores trifásicos de indução onde, portando um dispositivo móvel, o operador envia comandos, recebe a confirmação, além de poder realizar operações específicas do processo na planta industrial onde a máquina se encontra operando. Ao implementar tais controles, impede-se a ocorrência de possíveis erros ocasionados por falha humana e/ou o deslocamento do operador do equipamento entre a sala de comando dos mesmos e o ambiente fabril.

Tendo em vista o problema acima explicitado, o objetivo geral do presente estudo é projetar e testar um sistema de acionamento remoto de MITs utilizando a tecnologia wireless Wi-Fi e a plataforma microcontrolada Arduino, de tal forma que a máquina possa ser ativada por um dispositivo móvel e o seu estado de funciona-

mento também possa ser visualizado pelo mesmo. Para cumprir com o objetivo proposto, os seguintes objetivos específicos foram traçados:

a) Especificar e implementar o sistema de comunicação entre o dispositivo móvel e o sistema de controle central (Arduino);
b) Projetar, desenvolver e montar o circuito de acoplamento responsável pela interação entre o microcontrolador e o circuito de força do MIT;
c) Especificar, testar e configurar um dispositivo que provenha capacidade de comunicação Wi-Fi ao Arduino;
d) Descrever, aferir e parametrizar um roteador Wi-Fi no modo de infraestrutura;
e) Escrever e averiguar o programa responsável pelo desempenho e funcionamento do sistema de controle central;
f) Desenvolver e avaliar o software responsável pelos comandos entre o dispositivo móvel Android e a plataforma microprocessada Arduino;
g) Idealizar e montar o circuito de força da etapa de potência responsável pelo chaveamento da alimentação do MIT.

2. REVISÃO DE LITERATURA

Para trabalhar o estudo de caso proposto, faz-se necessário conhecer, de forma mais aprofundada, os principais componentes utilizados nesse projeto. Apesar dos demais elementos terem a sua importância, os apresentados nesta seção são os que definem o custo final e o desempenho geral do sistema aqui proposto. Sendo assim, serão discutidos, nesta fundamentação teórica, os conceitos mais significativos e abrangentes do presente projeto.

2.1. MODELOS DE REFERÊNCIA DE REDES DE TELECOMUNICAÇÃO

Conforme Lugli e Sobrinho (2008), o protocolo *Wi-Fi* (Nome dado ao sistema *Ethernet Wireless*) é a tecnologia de interconexão para redes locais e dispositivos sem fios. De acordo com os autores, em meados da década de 1990, um consórcio internacional de especialistas em engenharia (constituído por muitas empresas de tecnologia), começou a trabalhar por meio do Instituto dos Engenheiros Eletricistas e Eletrônicos (IEEE) para desenvolver os padrões sem fio da indústria e a forma como estes novos produtos deveriam interagir entre si. Fora dessa cooperação, o Wi-Fi Alliance® nasceu, tomou essas normas e desenvolveu laboratórios de testes em todo o mundo para aferir e certificar produtos que atendessem aos padrões de interoperabilidade e segurança especificados.

O termo *Wi-Fi* foi escolhido como uma brincadeira com o termo "Hi-Fi" e pensa-se, geralmente, que é uma abreviatura para *Wireless Fidelity*, no entanto, a *Wi-Fi* Alliance não reconhece isso. Comumente, o termo *Wi-Fi* é entendido como uma tecnologia de interconexão entre dispositivos sem fio, usando o protocolo IEEE 802.11. Dessa forma, com a tecnologia *Wi-Fi*, tornou-se possível criar redes locais sem fios com elevadas taxas de transmissão, desde que o dispositivo não esteja distante em relação ao ponto de acesso, segundo Kurose e Ross (2010). Segue, na figura 1, um exemplo básico de uma rede *Wi-Fi*, ou seja, *Ethernet* sem fio.

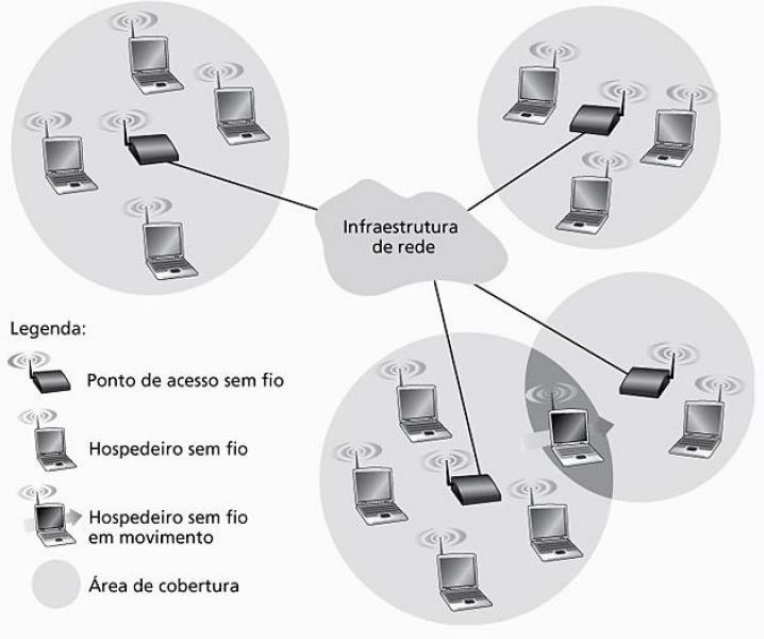


Figura 1 - Topologia de Rede Wireless.
Fonte: Kurose e Ross (2010).

Este protocolo opera em faixas de frequências de 2,4 GHz ou 5 GHz e não necessita de licença para instalação e operação. Pode transmitir dados em velocidades iguais ou superiores a 11 Mbps dentro de um intervalo de 30 metros, estando limitado pela largura de banda disponível. Tal fato torna essas redes também atrativas para aplicações em ambientes industriais, no entanto, para uso comercial no Brasil é necessário obter licença da Agência Nacional de Telecomunicações, Anatel (2015).

As redes sem fio, conforme Lugli e Sobrinho (2014) tendem a ser mais lentas do que as cabeadas. Os padrões dessas redes devem validar cuidadosamente os dados recebidos e se proteger contra possíveis perdas de dados devido a não confiabilidade no meio utilizado. A comunicação confiável somente ocorre quando é levado em consideração o ambiente hostil onde os equipamentos trabalham. Neste aspecto, deve ser considerado que os mesmos estão sujeitos a pó, poeira, calor, umidade (devido à condensação/evaporação de água).

Com o pensamento de Lugli e Sobrinho (2014) concordam Kurose e Ross (2010), ao afirmarem que os índices de Tempo Médio Entre Falhas (MTBF) devem ser altos e o Tempo Médio Entre Reparo (MTBR) deve ser o menor possível. Com relação às severidades encontradas no chão de fábrica, Filho e Yuri (2008), deixam claro que no ambiente industrial as redes sem fio não substituem as redes fixas, em um contexto

geral, trabalham em conjunto e apresentam uma condição tal que o usuário da rede pode estar se movendo e movimentar qualquer dispositivo, dando flexibilidade à aplicação. As redes Wi-Fi apoiam-se nos modelos de referência OSI, TCP/IP e IEEE 802.11.

2.1.1. MODELO DE REFERÊNCIA OSI

Muito embora o modelo de Interconexão de Sistemas Abertos (OSI) seja raramente usado hoje em dia na forma como foi originalmente concebido, de acordo com Santos (2011), é bem amplo e ainda válido. As características de cada camada ainda são muito importantes visto que esse modelo baseia-se em uma proposta desenvolvida pela International Standards Organization (ISO) visando à padronização internacional dos protocolos empregados nas diversas camadas. Esse modelo trata da interconexão de sistemas que estão disponíveis para se comunicarem entre si, estando divido em sete camadas com funções bem definidas, conforme demonstrado na figura 2.

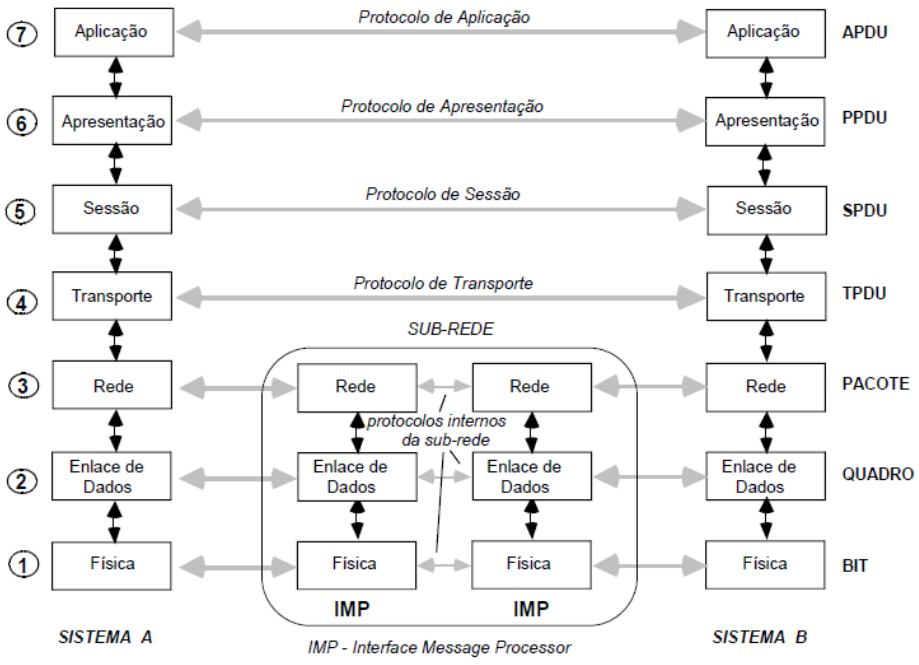


Figura 2 - Modelo ISO/OSI.
Fonte: Stemmer (2001).

De acordo com Torres (2001), a camada de aplicação é responsável pela interface entre o protocolo de comunicação e o aplicativo que pediu ou que receberá a informação através da rede. Uma entre as inúmeras situações onde isto ocorre é quando um aplicativo de e-mail solicita um e-mail para que o usuário possa lê-lo. O aplicativo aciona a camada de aplicação do protocolo de rede e efetua tal solicitação. Tudo nesta camada é direcionado aos aplicativos. Além dos aplicativos de e-mail, o Telnet, o FTP, entre outros são aplicativos de rede que existem inteiramente na camada de aplicação.

A próxima camada é de apresentação. Para Torres (2001), ela tem por função converter os dados recebidos da camada imediatamente superior e os transformar em um formato comum, de tal forma que esses dados possam ser transmitidos (em um formato entendido pelo protocolo que está sendo utilizado). Quando se quer aumentar a segurança, é usual utilizar algum tipo de criptografia nesta camada, onde os dados só serão decodificados na camada seis do dispositivo receptor.

Na descrição de Santos (2011), a camada de sessão tem por papel permitir que as aplicações utilizadas em diferentes dispositivos estabeleçam uma sessão de comunicação. Para isso, as aplicações definem como ocorrerá a transmissão de dados (são colocadas “marcações” nos dados que serão enviados). Assim, caso a rede falhe, os equipamentos podem reiniciar a transmissão das informações partindo da última marcação recebida.

Para Kurose e Ross (2010), a camada de transporte é a peça central das arquiteturas de redes em camadas. Ela desempenha o papel de fornecer serviços de comunicação diretamente aos processos de aplicação que rodam em diferentes hospedeiros. Sua função é receber os dados enviados pela camada de sessão e os dividir em segmentos que serão transmitidos para a camada de rede. Já no receptor, a camada de transporte recebe os pacotes enviados pela camada de rede, e remonta os dados originais antes de enviá-los à camada de sessão.

Para Kurose e Ross (2010), a camada de rede é uma das camadas mais complexas da pilha de protocolos, sendo responsável por receber os pacotes oriundos da camada de transporte, encapsulá-los na forma de datagramas e os enviar até o seu

destino. No hospedeiro receptor, essa mesma camada receberá os datagramas provindos da camada de enlace, extrairá os segmentos da camada de transporte, entregando-os à mesma. Na exposição de Santos (2011) a camada de rede endereça os pacotes, transformando os endereços lógicos em endereços físicos, assim, os pacotes conseguem chegar corretamente ao seu curso. Além disso, a camada de rede determina a rota que os pacotes irão seguir para atingir o destino, baseada em fatores como condições de tráfego da rede e prioridades.

Para Tanenbaum (2003), a principal tarefa da camada de enlace de dados é transformar um canal de transmissão bruto em uma linha que pareça livre de erros de transmissão não detectados para a camada de rede. Para executar essa tarefa, a camada de enlace de dados faz com que o transmissor divida os dados de entrada em quadros (que, em geral, têm algumas centenas ou alguns milhares de bytes), e os transmita sequencialmente. Se o serviço for confiável, o receptor confirmará à recepção correta de cada um deles, enviando de volta um quadro de confirmação.

Na visão de Tanembaum (2003), surge outra questão na camada de enlace de dados (e na maioria das camadas mais altas). Trata-se de como impedir que um transmissor rápido envie uma quantidade excessiva de dados a um receptor lento. Com frequência, é necessário algum mecanismo que regule o tráfego para informar ao transmissor quanto espaço o *buffer* do receptor tem no momento. Muitas vezes, esse controle de fluxo e o tratamento de erros estão integrados.

Tanenbaum (2003) também argumenta que a camada física trata da transmissão de bits brutos por um canal de comunicação. Segundo o autor, o projeto da rede deve garantir que, quando um lado enviar um bit 1, o outro lado receberá exatamente esse mesmo bit 1, não como um bit 0. Nesse caso, as questões mais comuns são a voltagem a ser usada para representar um bit 1 e um bit 0, a quantidade de nanossegundos (ns) que um bit deve durar, o fato da transmissão poder ser ou não realizada nos dois sentidos simultaneamente, a forma como a conexão inicial será estabelecida, de que maneira ela será encerrada quando ambos os lados tiverem terminado, quantos pinos o conector de rede terá e qual será a finalidade de cada pino. Nessa situação, as questões de projeto lidam em grande parte com interfaces

mecânicas, elétricas, sincronização e com o meio físico de transmissão que se situa abaixo da camada física.

2.1.2. MODELO DE REFERÊNCIA TCP/IP

De acordo com Microsoft (2014), o Protocolo de Controle de Transmissão/Protocolo de Internet (TCP/IP) não é um único protocolo, mas sim um conjunto. Por causa da diversidade do mesmo, não utiliza diretamente o modelo OSI e subdivide-se nas camadas elencadas na figura 3.

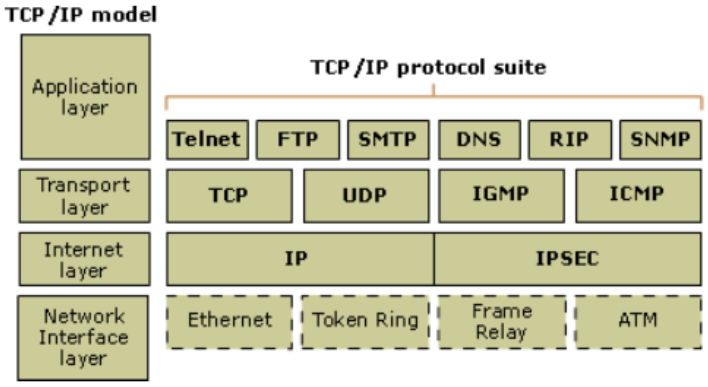


Figura 3 - Modelo TCP/IP.
Fonte: Microsoft (2014).

Para Kurose e Ross (2010), a camada de aplicação é responsável pelos aplicativos do TCP/IP. Há dois tipos de aplicativos nessa camada: baseados em soquetes e no Sistema Básico de Entrada e Saída da Rede (NetBIOS). Aplicativos fundamentados em soquetes existem em todos os clientes que o utilizam. Três elementos são exigidos para os aplicativos baseados em soquetes: um endereço IP, uma porta e um tipo de serviço. Cada cliente que utiliza o protocolo terá um endereço único de 32 bits. Cada endereço tem 65.536 pontos de entradas – cognominados portas. Os aplicativos operam em portas particulares.

Conforme os autores acima mencionados, o propósito da camada de transporte é conectar ou não conectar, dois protocolos são utilizados na camada de transporte: Transmission Control Protocol (TCP) e User Datagram Protocol (UDP). O TCP é uma comunicação confiável orientada para conexão que é mais lenta na transmissão. O UDP é uma comunicação não garantida e não orientada a conexão, que é

mais rápida na transferência de dados. Quando um aplicativo utiliza o TCP para comunicação, um *handshake* de três vias é estabelecido, assegurando que os pacotes são entregues livres de erros, na sequência e sem perda ou duplicação de dados, garantindo a entrega do pacote, porém mais lento na transferência.

Ainda, de acordo com Kurose e Ross (2010), a camada de Internet funciona quase da mesma maneira como a camada de rede do modelo OSI. A camada internet é a principal responsável pelo endereçamento e o roteamento de rede. Além disso, esta camada é responsável pela fragmentação do pacote. Os datagramas de dados são montados e remontados para transmissão nessa camada, sendo muitos os protocolos que nela operam (os mais comuns são: IP, ICMP, ARP e IGMP). Para o autor, a camada de rede corresponde às camadas de enlace e físicas do modelo OSI, sendo responsável pelo acesso à rede. Segundo ele, a primeira se comunica diretamente com a rede, sendo a ligação entre a topologia de rede e a camada de internet.

2.2. O PROTOCOLO HTTP

Para Kurose e Ross (2010) o que mais atrai a maioria dos usuários da Web é o seu funcionamento sob demanda, isto é, as pessoas recebem o que querem e quando querem (diferentemente da transmissão radiotelevisiva – que as obrigam a receberem o conteúdo disponibilizado pelo provedor). Além de funcionar desta forma, segundo o autor, a internet tem muitas outras características fantásticas que as pessoas gostam: facilidade com a qual qualquer uma pode tornar informações disponíveis. Buscadores e hiperenlaces tornam muito prático e rápido o acesso à informação, dispositivos gráficos estimulam os sentidos, além da vasta gama de material multimídia disponibilizados.

De acordo com Kurose e Ross (2010), o Protocolo de Transferência de Hipertexto (HTTP) trabalha na camada de aplicação da Web, sendo definido pelas RFCs 1945 e 2616, além de ser implementado em dois programas: servidor e cliente. Ambas as aplicações são executadas em sistemas finais distintos e interagem entre si por meio da troca de mensagens no formato HTTP. Na visão dos autores, o protocolo define a estrutura dessas mensagens, bem como, a forma pela qual o cliente e servidor se

comunicam. O HTTP especifica como os clientes requisitam páginas Web aos servidores e, como estes respondem aqueles. A figura 4 ilustra como se dá esse processo.

Figura 4 - Modelo Requisição/Resposta do Protocolo HTTP.
Fonte: Kurose e Ross (2010).

Segundo Kurose e Ross (2010), um navegador é essencialmente um programa que pode exibir uma página Web, mas primeiro é necessário encontrá-la. E como ele faz isso? Através do Localizador Padrão de Recursos (URL). Para os autores as URLs podem ser comparadas aos ponteiros que indicam um determinado local, neste caso, uma página Web. Isto é: os URLs são identificadores universais da página, ou seja, não existem URLs repetidas, seria como o Cadastro de Pessoas Físicas (CPF) de uma pessoa no Brasil.

As URLs são formadas por três partes: a primeira contém o tipo de protocolo a ser usado. A segunda, o nome DNS da máquina ou servidor em que está a página e, por último, o nome específico da mesma (normalmente o nome do arquivo o qual ela representa). Um exemplo: a URL para a página principal da Universidade Aberta do Brasil (ESAB) – Campus de Vila Velha/ES - é a seguinte:

```
URL: http://www.esab.edu.br
```

A primeira parte da URL acima destacada diz respeito ao protocolo que está sendo usado, neste casso, o HTTP. A segunda parte mostra o DNS da máquina que ostenta a página <www.esab.edu.br>, também conhecida pelo nome do *Host*. Pode-se dizer que, sob o ponto de vista do usuário, uma solicitação de uma página HTML se resume, às vezes, em apenas um clique do mouse.

Na verdade, ela frequentemente dá origem a várias solicitações HTTP que são enviadas do *browser* para o servidor web e deste para o cliente. Quando um usuário clica em um *Hyperlink*, o navegador executa uma serie de etapas realizadas sucessivamente para buscar a página indicada pelo mesmo. A sequência de passos que o navegador realiza é descrita conforme o exposto abaixo (KUROSE; ROSS, 2010):

1. O navegador determina o URL (verificando o que foi selecionado);
2. O navegador pergunta ao DNS qual é o endereço IP de <www.esab.edu.br>;
3. O DNS responde com 54.94.171.206.
4. O navegador estabelece uma conexão TCP com a porta 80 em 54.94.171.206;
5. Em seguida, o navegador envia um comando solicitando o arquivo /index.html;
6. O servidor <www.esab.edu.br> envia o arquivo /index.html;
7. A conexão TCP é encerrada;
8. O navegador exibe todo o texto de /index.html;
9. O navegador busca e exibe todas as imagens que o arquivo contém.

Para Kurose e Ross (2010) este processo é descrito como uma relação entre o usuário e o servidor. Nem todas as páginas Web contêm documentos no formato HTML. Na prática, podem conter documentos com vários tipos de extensões e podem ser vinculados a um documento HTML, visto não ser simples construir navegadores que interpretem todos os tipos de formatos, o que não seria viável. Se assim fosse, os *browsers* ficariam enormes para poderem interpretar todos os formatos existentes e que crescem a cada dia.

Segundo os autores, uma solução encontrada para esse problema é o uso das Extensões Multifunções para Mensagens de Internet (MIMO). Quando um servidor retorna uma página ele também envia algumas informações adicionais como o MIME. Nesse processo, as páginas do tipo <text/html> são exibidas diretamente, como também, as criadas em outros tipos internos. Se ele não existir nativamente, o navegador consulta a sua tabela de tipos MIME para saber como exibir a página. Essa tabela associa um tipo MIME a um visualizador.

Existem ainda duas outras formas de abrir um arquivo que não é interno do navegador: a instalação de um *plug-in* ou uma aplicação auxiliar. Na primeira, o *plug-in* é uma extensão para o próprio navegador e tem acesso à página atual. Quando o usuário termina o seu trabalho, ele não é mais utilizado e retirado da memória. O segundo caso ocorre quando o navegador não consegue interpretar um documento, por exemplo, um arquivo no Formato de Documento Portável (PDF). Assim, ele pro-

cura no registro do sistema operacional um programa para visualizar o documento. No caso específico da abertura de um arquivo PDF no sistema Ubuntu Linux (por exemplo), ele chama o programa Acrobat Reader ou qualquer outro que esteja instalado no sistema e que seja capaz de abrir o arquivo (IETF, 2014a).

Conforme descrevem Kurose e Ross (2010) um servidor é uma aplicação que espera passivamente, de forma assíncrona, por uma conexão – e não o computador em que a aplicação é executada, como se poderia imaginar. Sendo assim, é incorreto afirmar que um computador com um processador rápido e de alta potencialidade de hardware seria um servidor. De certa forma, um servidor Web se divide em três partes: o servidor propriamente dito, a rede e o *browser* (cliente).

Um servidor web está sempre em um laço infinito, permanentemente aguardando por requisições dos clientes. Nesta espera, existem alguns atrasos que são inevitáveis como a espera pela transmissão dos dados na rede, o acesso ao disco da máquina, o escalonamento dos processos pelo sistema operacional, entre outros. O equipamento que hospeda a página, portanto, deve ser projetado de modo a atender o maior numero de requisições que lhe seja possível. Para que o cliente possa solicitar algum recurso, deve utilizar métodos para fazer negociações ao mesmo. O servidor também se utiliza de métodos para enviar as respostas das solicitações dos clientes (IETF, 2014a).

O protocolo HTTP define um conjunto de métodos com os quais os clientes podem trabalhar e que funcionam como comandos enviados ao servidor web. Os métodos do protocolo HTTP 1.0 descritos na RFC 1945 são: GET, HEAD e POST. As mensagens de requisição e resposta do protocolo HTTP seguem um padrão, ou seja, uma requisição é constituída de uma linha informando o método a ser executado, seguida de um cabeçalho que pode ter mais de uma linha e pelo o corpo da mensagem (caso seja necessário). Segundo Kurose e Ross (2010) o método GET é uma solicitação de uma página ou objeto ao servidor, exemplo:

```
GET /index.html HTTP/1.1
Host: www.esab.edu.br
Connection: close
User-agent: Mozilla/28.0
```

```
Accept-language: br
```

Neste modelo de requisição é possível notar que, na primeira linha, está o tipo do método (GET), o arquivo solicitado e a versão do HTTP usada. Na segunda linha está o endereço do host onde está hospedado o objeto. <Connection: close> mostra que não é uma conexão persistente, <User-agent:> informa qual o *browser* utilizado e <Accept-language:> se refere à linguagem que é solicitada. Caso ela não exista no domínio será usada a linguagem padrão. A resposta do servidor deve ser:

```
HTTP/1.1 200 OK
Connection: close
Date: Mon, 12 Dec 2005 04:15:03 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Sun. 5 May 2005 09:25:23 GMT
Content-Length: 6821
Content-Type: text/html
```

Analisando a mensagem de resposta identifica-se, na primeira linha, o estado da solicitação que – no modelo a cima – foi estabelecida sem erros e o servidor está enviando o objeto solicitado. <Connection: close> está informando que é uma conexão não persistente. <Date:>, é a data de acesso ao objeto. <Server:>, o tipo de servidor web. <Last-Modified:>, data da última alteração ou criação do arquivo, <Content-Length:>, tamanho do arquivo e, por último, <Content-Type:>, responsável por informar o tipo do arquivo. Abaixo segue a relação dos métodos usados pelo protocolo (IETF, 2014b):

a) GET → Método que solicita algum recurso ou objeto ao servidor;
b) HEAD → Solicita informações de um determinado objeto sem que esse seja enviado ao cliente (apenas para testar a validade do último acesso);
c) POST → Método usado para envio de arquivo de dados ou formulário HTML ao servidor;
d) OPTIONS → Por meio desse método o cliente obtém as propriedades do servidor;
e) DELETE → Informa, por meio do URL, o objeto a ser excluído;
f) TRACE → Enviar mensagem do tipo *loopback* para teste;
g) PUT → Aceita criar ou modificar algum objeto do servidor;
h) CONNECT → Comunicar-se com servidores *Proxy*.

O termo *cookie*, segundo Duckett (2012), é derivado do inglês e significa “bolacha” e recebeu esse nome de uma antiga gíria usada pelos programadores. Conforme o autor, tal termo era adotado para definir um programa que deveria chamar um procedimento e receber de volta algo que seria necessário apresentar novamente mais tarde para realizar algum trabalho. Foi criado pela empresa Netscape para solucionar o problema do envio e solicitação de arquivos, que era esquecido pelo servidor e poderia ser usado por outros computadores com o mesmo IP, o que causava problemas, pois não se sabia na realidade se era ou não aquele usuário mesmo.

Os *cookies* são arquivos ou *strings* (não são programas executáveis). Eles são tratados como dados pelo navegador, não existindo nenhuma maneira de serem utilizados como vírus, apensar da possibilidade da exploração de *bugs* no servidor e provocar a ativação de um cookie como vírus por um *hacker*. Basicamente, ele é um grupo de dados trocados entre o servidor de páginas e o navegador, colocado em um ficheiro criado no computador do usuário. Serve para manter a persistência das sessões HTTP (DUCKETT, 2012).

Ele funciona da seguinte forma: Um usuário solicita uma página da Web, nisso o servidor pode fornecer informações adicionais acompanhando a página solicitada. Essas informações podem incluir um *cookie*, um pequeno arquivo ou *string* (com 4 KB no máximo). Este elemento pode ter até cinco campos: Domain, Path, Content, Expires e Secure.

a) Domain → Informa de onde veio o *cookie*. O navegador confirma que os servidores estão enviando dados fieis a respeito de seu domínio. Cada domínio pode armazenar no máximo 20 *cookies* por cliente;
b) Path → É um caminho na estrutura de diretórios do servidor que identifica as partes da árvore de arquivos que podem usar o cookie. Frequentemente, ele obtém o símbolo / (barra), que representa a árvore inteira;
c) Content → Utiliza a forma <nome=valor>, podendo o servidor definir da maneira que quiser tanto o valor quanto o nome e é nele que fica armazenado o conteúdo do *cookie*;

d) Expires → É o campo que faz o *cookie* persistir. Nele estão contidos a data e o horário em que o mesmo irá expirar. Caso esse campo esteja ausente, o navegador o descartará automaticamente após o termino da seção. O último campo define se é seguro ou não.

Os *cookies* são utilizados para identificar o usuário que configurou uma página web de forma que, da próxima vez que a solicitar, esteja da mesma maneira que foi deixada. Pode ser usado, também, quando se faz a solicitação de armazenamento de senha. Em momento posterior, quando navegar pelo site, a sua senha será lembrada (DUCKETT, 2012).

2.3. A LINGUAGEM HTML

A Linguagem de Marcação de Hipertexto (HTML) é utilizada para a criação de documentos do tipo Hipertexto. Uma página HTML é um documento composto de textos e códigos especiais denominados *Tags* que possibilitam a exibição de um documento na *Web*. Mais do que informações textuais, estas informações podem conter imagens, sons, animações e até vídeos. Uma página Web pode se ligar a outra por meio de links endereçáveis para locais dentro do computador ou em qualquer parte do mundo (SILVA, 2008).

Para Silva (2008) o que é chamado de HTML é basicamente uma linguagem simples que tem como princípio fundamental formatar textos criando uma aparência agradável e promover ligações ou *links* com outras páginas. Embora simples, não foge às regras clássicas que as linguagens de programação de computador exigem. O código fonte uma vez pronto é interpretado pelo navegador que se encarregará de executar os comandos colocados dentro do programa para acessar os recursos disponíveis.

Conforme o autor acima mencionado, para escrever um arquivo HTML, o mais simples dos editores disponível pode ser usado. No sistema Ubuntu Linux, pode ser utilizado o editor de texto GEdit, por estar disponível em todas as versões do mesmo. Uma vez que o código HTML for digitado, ele deverá ser visualizado ou

conferido em um *browser*. O navegador padrão do Ubuntu é o Mozilla Firefox que pode ser encontrado no Dash (menu iniciar) de todos os computadores que o tenham instalado.

Um arquivo HTML é construído a partir de um código fonte, composto de texto puro no formato do Código Padrão Americano para o Intercâmbio de Informação (ASCII) que é um conjunto de normas para a representação de caracteres por meio de números binários. Este sistema é amplamente utilizado em informática, telecomunicações, programação, etc.. Analogicamente pode-se pensar em uma receita culinária que quando interpretada pelo cozinheiro torna-se uma iguaria a ser degustada. Como escritores do código fonte, ou criadores da receita, cabe aos programadores determinarem exatamente como desejam que um site se comporte (LUBBERS; ALBERTS; SALIM, 2013).

Um programa HTML é dividido em três partes básicas: a estrutura principal, o cabeçalho e o corpo do programa. Todo site escrito nessa linguagem deve começar com o comando <HTML> e ser encerrado com o comando </HTML>. A área de cabeçalho é opcional, sendo delimitada pelo par de comandos <HEAD> </HEAD>. A maioria dos comandos é adicionada à área do corpo do documento e é formado pelos comandos <BODY> </BODY>. A estrutura mínima de um documento HTML requer todas as linhas obrigatórias e essenciais para o bom funcionamento de uma página *web*. Todas as *Tags* apresentadas aqui aparecem uma única vez em cada arquivo HTML (SILVA, 2008). A figura 5 ilustra essa divisão.

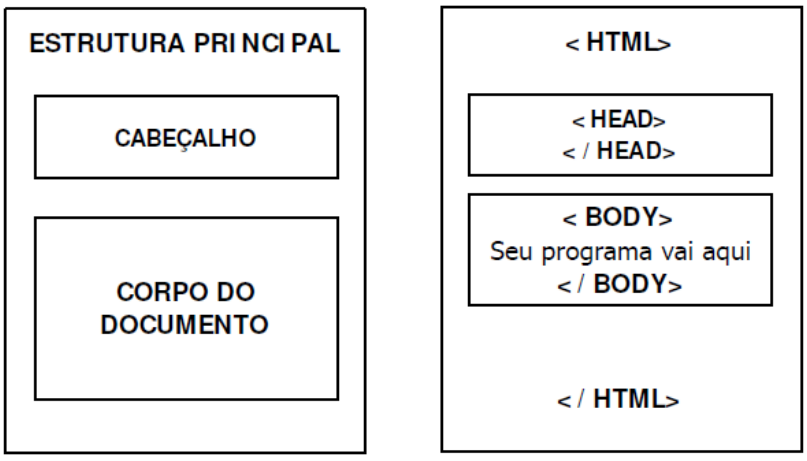


Figura 5 - Estrutura de Um Documento HTML.
(Fonte: Autor).

As Tags iniciam-se com o sinal < (menor que), seguida pelo nome da instrução ou comando e fechada por > (maior que). De um modo geral, aparecem em pares, por exemplo, <H1> Cabeçalho </H1>. O símbolo que termina uma determinada marcação é igual àquele que a inicia, precedido por uma barra. Há exceções a essa regra, por exemplo, a *Tag* que indica um final de parágrafo <P> não necessita de uma correspondente </P>. A marcação que indica quebra de linha
 também não precisa de uma correspondente </BR> (SILVA, 2008).

Existe uma grande quantidade de tags nativas da linguagem. Tais elementos podem ser utilizados para formatar textos, criar *hiperlinks*, inserir tabelas, elaborar *frames*, adicionar elementos multimídia (áudio, vídeo e imagem) e gerar gráficos. Não é objetivo do presente trabalho esgotar o tema, mas apenas introduzi-lo, uma vez que esta linguagem será utilizada em conjunto com a do Arduino para permitir à comunicação de longa distância HTTP, rodando sobre o protocolo TCP/IP em sistemas de comunicação cujo meio de transmissão é o padrão de redes sem-fio IEEE 802.11 (SILVA, 2008).

2.4. PADRÃO IEEE 802.11

Na medida em que a tecnologia de redes sem fio *Wi-Fi* evoluía, o padrão IEEE 802.11 foi expandido de forma a melhorar aspectos da rede, como a taxa de transmissão e a segurança. Estas melhorias foram incorporadas sob a forma de "emendas", designadas por letras acrescentadas ao nome do padrão, como o IEEE 802.11g (SANTOS, 2011). A figura 6 mostra as camadas deste padrão de rede.

Figura 6 - Camadas do Padrão IEEE 802.11.
Fonte: Santos (2011).

Conforme, Rappaport (2009), o padrão 802.11b foi o primeiro padrão IEEE 802.11 a se popularizar. Ele opera na faixa entre 2,4 e 2,4835 GHz e tem a possibilidade de estabelecer conexões nas seguintes velocidades de transmissão: 1 Mbps, 2 Mbps, 5,5 Mbps e 11 Mbps. Já o padrão seguinte, o 802.11a, foi lançado no mesmo ano que o 802.11b (1999) e, apesar de oferecer taxas mais altas, não alcançou a mesma popularidade. Na explanação do autor, as taxas adicionais oferecidas pela emenda "a" são: 6 Mbps, 9 Mbps, 12 Mbps, 18 Mbps, 24 Mbps, 36 Mbps, 48 Mbps e 54 Mbps. As frequências utilizadas por este padrão estão entre 5,725 e 5,875 GHz. Nesta faixa de frequência mais alta, o sinal é mais suscetível a perdas de propagação, diminuindo seu alcance em comparação com a faixa utilizada pelo IEEE 802.11b. Em contrapartida, o uso desta frequência pode ser conveniente por estar menos sujeita a interferência de outros dispositivos.

O padrão 802.11g pode ser considerado o sucessor do padrão 802.11b, pois opera na mesma faixa de frequência. Dispositivos que implementam o 802.11g costumam também ser "retro-compatíveis", isto é, implementam também o 802.11b, sendo muitas vezes especificados como dispositivos 802.11b/g. Sua principal vantagem é a possibilidade de operar com taxas de transmissão de até 54 Mbps, como o IEEE 802.11a, e ao mesmo tempo ter o alcance do IEEE 802.11b (RAPPAPORT, 2009).

De acordo com o autor acima mencionado, o desenvolvimento do padrão 802.11n foi iniciado em 2004 e tinha previsão de publicação para 2010. Este padrão pode operar nas faixas de 2,4 GHz e 5 GHz, o que o torna compatível com os padrões anteriores. Sua principal característica é o aumento considerável das taxas de transferência de dados através da combinação de várias vias de transmissão (múltiplas antenas). Uma das configurações mais comuns é o uso de ponto de acesso com três antenas e estações com a mesma quantidade de receptores. O padrão 802.11n pode operar com taxas de transmissão de dados de até 300 Mbps. Existem no mercado equipamentos que oferecem capacidades baseadas nas versões preliminares do padrão, chamados "pré-n" (RAPPAPORT, 2009).

Em relação à segurança das redes *wireless*, Moreno, Pereira e Chiaramonte (2005), consideram um dispositivo ou uma rede seguro quando atendem a três requisitos relacionados aos recursos que o compõem: confidencialidade, disponibilidade e integridade, ou seja, a confidencialidade garante que a informação só estará disponível para aqueles devidamente autorizados, a integridade garante que a informação não será perdida ou corrompida e a disponibilidade garante que os recursos estarão disponíveis quando necessários. A segurança da informação mostra-se cada vez mais imprescindível, pois se tornou o maior patrimônio de uma empresa.

A integridade, confidencialidade, e a disponibilidade da informação são fatores primordiais para as empresas, e para se obter esta segurança das informações utiliza-se a criptografia. Tal método de proteção da informação pode ser tanto com chave única quanto com duas chaves, uma pública e uma privada, na criptografia com chave única a mesma chave é utilizada tanto para codificar quanto para descodificar as mensagens, é um método eficiente em relação ao tempo de processamento, porém tem a desvantagem de precisar de um meio seguro para compartilhamento da chave. Já no método com duas chaves, uma codifica a mensagem e a outra, descodifica. Neste método, cada pessoa precisa ter duas chaves: uma pública – que pode ser divulgada – e outra privada, que deve ser mantida em sigilo. Uma mensagem codificada com a chave pública só pode ser descodificada com a chave privada correspondente (MORENO; PEREIRA; CHIARAMONTE, 2005).

O protocolo de Segurança Equivalente às Redes Com Fio (WEP) foi o primeiro adotado para segurança de redes sem-fio, que conferia no nível de enlace (nível 2 do modelo OSI), certa segurança semelhante à das redes cabeadas. Este padrão apresentou muitas falhas relativamente simples de serem exploradas, mas manteve a camada de proteção básica que deveria sempre estar ativa. Como o padrão 802.11 trata de um protocolo de segurança do nível de enlace de dados, o WEP teoricamente foi projetado para assegurar privacidade, sendo muito utilizado em sua época de criação. Para tanto, utiliza o algoritmo RC4 para criptografar os pacotes que serão trocados numa rede sem fios a fim de tentar garantir confidencialidade aos dados de cada usuário (DUARTE, 2010).

Quando a segurança no IEEE 802.11 é ativada, cada cliente tem uma chave secreta compartilhada com a estação base, porém, o padrão não especifica como as chaves são distribuídas. Uma vez difundidas, permanecem estáticas por meses ou anos. Ao ser habilitado, o WEP protege somente os dados do usuário do pacote – não os cabeçalhos – para que outras estações possam escutar os dados de controle necessários pra manter a rede. No entanto, as demais estações não serão capazes de decodificar os dados de usuário (D-LINK, 2002).

A privacidade oferecida pelo WEP se baseia em chaves criptográficas simétricas de 40 bits e um vetor de inicialização público de 24 bits (*IV – Initialization Vector*). Para se conectar à rede, cada estação deve conhecer a chave atual, dessa forma, um usuário indesejado somente poderá acessar seus dados se conseguir quebrar a criptografia. O WEP pode ser utilizado entre o Ponto de Acesso (AP) e os clientes da rede (modo com infraestrutura), ou na comunicação direta entre clientes (modo ad-hoc). Esse tipo de criptografia só é aplicada ao tráfego do canal de comunicação sem fio e, portanto, o tráfego roteado para fora da rede *wireless* não possui criptografia WEP (D-LINK, 2002).

Devido a todos os problemas (falhas e vulnerabilidades) detectados no protocolo WEP, um grupo de membros da Wi-Fi Alliance e do IEEE se empenharam em desenvolver um novo protocolo que resolvesse algumas das vulnerabilidades apresentadas pelo WEP. Surgia então, no ano de 2002, a primeira versão do Acesso Protegido ao Wi-Fi (WPA) que também foi chamado de WEP2 ou Protocolo de Checagem de Integridade com Chave Temporal (TKIP). (DUARTE, 2010, p. 36).

O WPA trabalha com a Checagem da Integridade da Mensagem (MIC) e é parte do padrão 802.11. Ele evita que um pacote seja alterado durante transmissão em uma rede sem fio. Como uma parte do TKIP, o MIC provê um campo adicional de 8 bytes dentro da especificação 802.11 que protege tanto os dados de carga útil quanto o cabeçalho do pacote de manipulação indesejada. O algoritmo que implementa o MIC é conhecido como *Michael Shamir* (RAPPAPORT, 2009).

De acordo com Duarte (2010), O WPA corrigiu vários erros do WEP, porém, ainda restaram algumas vulnerabilidades. Além disso, seu desempenho teve uma queda

significativa em termos de estabilidade, por isso, houve a necessidade de se criar outro protocolo que fosse mais seguro e tivesse melhor desempenho. O WPA2 foi especificado com a promessa de ser a solução definitiva de segurança e estabilidade para as redes sem-fio do padrão *Wi-Fi*.

Na visão de Rappaport (2009) a principal mudança entre o WPA2 e o WPA é o método criptográfico utilizado. Enquanto o WPA utiliza o TKIP com o RC4, o WPA2 utiliza o Padrão de Criptografia Avançada (AES) em conjunto com o TKIP com chave de 256 bits, que é um método muito mais poderoso. Também, como o WPA, o WPA2 usa tecnologia de autenticação IEEE 802.1X/EAP ou tecnologia de PSK, mas trabalha com um mecanismo novo de encriptação avançado e mais robusto que o TKIP que usa o Counter-Mode/CBC-MAC Protocol (CCMP), o AES.

Tal sistema de encriptação foi adotado como o padrão oficial pelo Departamento Norte-americano de Comércio e pelo Instituto Nacional de Padrões e Tecnologia (NIST). Na sua especificação, o padrão 802.11i garante que os dados enviados por essas redes sejam criptografados e não sejam violados por nenhum tipo de interceptação (SYMANTEC, 2010).

2.5. ELEMENTOS DE REDES WI-FI

De acordo com Morimoto (2008) uma rede sem fio conecta dispositivos sem usar cabos, utilizando comunicação por rádio para enviar dados entre eles. Em uma rede *Wi-Fi*, é possível se comunicar diretamente com outros equipamentos sem fios ou estabelecer ligação com uma rede existente através de um AP/Roteador *wireless*. Ao configurar o adaptador sem fio, é necessário selecionar o modo de operação que se deseja para cada rede em particular. Os seguintes dispositivos são essenciais para a formação de uma *WLAN* (ver figura 7):

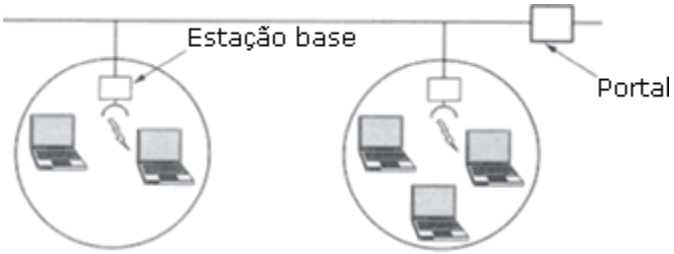


Figura 7 - Dispositivos de Uma Rede WLAN.
Fonte: Kurose e Ross (2010).

Na abordagem de Morimoto (2008) os roteadores são também conhecido como estação-base ou AP, o ponto de acesso funciona como transmissor de rádio e como *bridge* (ponte), transferindo dados dos clientes para a rede de cabos fixos. Fazendo uma analogia com as redes cabeadas, os pontos de acesso substituem os *switches* assim como as ondas de rádio substituem os cabos.

Além de servirem como *bridge* entre a rede sem fio e a rede de cabo, o ponto de acesso provê as funções associadas a um roteador. Eles podem funcionar como um servidor de Configuração Dinâmica de Hosts (DHCP), e fazer Tradução de Endereços de Rede (NAT), para atender a vários usuários utilizando um único endereço IP (KUROSE; ROSS, 2010). Assim, pode–se fazer um balanceamento de carga entre múltiplos pontos de acesso, permitindo que um usuário se desloque de um equipamento a outro sem perder a conexão, funcionalidade conhecida como *roaming*. O número de clientes que podem acessar um único ponto de acesso depende das condições do ambiente físico, tráfego da rede e das aplicações que serão suportadas pela rede sem fio (RAPPAPORT, 2009).

Conforme Kurose e Ross (2010), em uma rede Wi-Fi, os adaptadores de rede possuem um transmissor de rádio que envia dados do computador ou do ponto de acesso para a rede, e um receptor que detecta os sinais de rádio que chegam, os quais contêm dados da rede, e os envia para o destinatário. Existem, porém, vários modelos de adaptadores, cada um com características específicas. Cada equipamento final (notebooks, computadores de mão, etc.) deve ter um adaptador de rede que permita estabelecer a comunicação com os pontos de acesso.

Os adaptadores de rede em cada computador convertem os dados digitais para sinais de rádio, os quais são transmitidos para outros dispositivos na rede e

convertem os sinais de rádio que chegam dos outros elementos da rede de volta para os dados digitais. Esses adaptadores são tipicamente um PC Card (*Personal Computer Memory Card International Association*) com uma antena integrada e desenhada para ocupar um *slot* de expansão do equipamento. Porém, os novos equipamentos já vêm com a capacidade de se conectar a rede sem fio embutida no *hardware* (KUROSE; ROSS, 2010).

O padrão IEEE 802.11 define dois modos de funcionamento para redes Wi-Fi: o modo Infraestrutura (os clientes sem fios são conectados a um ponto de acesso) e o modo Ad-Hoc, onde os clientes são conectados uns aos outros sem nenhum ponto de acesso (MARQUES, 2004). No modo infraestrutura, conforme Marques (2004), cada estação se conecta a um ponto de acesso via ligação sem fios. O conjunto formado pelo ponto de acesso e as estações situadas na sua zona de cobertura é denominado Conjunto de Serviços Básicos (BSS) e constituem uma célula. Cada BSS é identificado com um BSSID, um identificador de 6 bytes (48 bits). No modo infraestrutura, o BSSID corresponde ao endereço MAC do ponto de acesso (MARQUES, 2004).

É possível conectar vários pontos de acesso entre si (vários BSSs) por uma ligação chamada Sistema de Distribuição DS. Para constituir um conjunto de Serviços Vastos (ES). O sistema de distribuição (DS) pode ser igualmente uma rede telegráfica, um cabo entre dois pontos de acesso ou mesmo uma rede sem fios. Um ESS é identificado por um ESSID (Service Set Identifier), ou seja, um identificador de 32 caracteres no formato ASCII que serve de nome para a rede. O ESSID, frequentemente abreviado em SSID, representa o nome da rede e representa um primeiro nível de segurança, na medida em que o conhecimento do SSID é necessário para que uma estação se ligue à rede (KUROSE; ROSS, 2010).

Quando um usuário autônomo migra de um BSS a outro, devido ao seu deslocamento no ESS, o adaptador de rede sem fios do seu equipamento é capaz de alterar o ponto de acesso de acordo com a qualidade de recepção dos sinais que provêm dos diferentes dispositivos. Os *access-points* se comunicam entre si graças ao sistema de distribuição de informações sobre as estações e permitirem, se for caso, a transmissão dos dados das estações móveis. Essa característica, que permite às

estações “passarem de maneira transparente” de um ponto de acesso a outro, chama-se *roaming* (RAPPAPORT, 2009).

Em modo ad-hoc, as máquinas clientes conectam-se umas às outras para constituir uma rede ponto a ponto (*Peer-to-Peer*), ou seja, uma rede na qual cada máquina desempenha ao mesmo tempo o papel de cliente e o papel de ponto de acesso. O conjunto formado pelas diferentes estações chama-se Conjunto de Serviços de Base Independentes (IBSS). Um IBSS é uma rede constituída, no mínimo, por duas estações e que não utiliza ponto de acesso. Ele forma, por conseguinte, uma rede que permite às pessoas situadas numa mesma sala trocar dados. É identificado por um SSID, como o é um ESS em modo infraestrutura (MARQUES, 2004).

Numa rede ad-hoc, o alcance do BSS independente é determinado pelo alcance de cada estação. Isso significa que se duas estações da rede estiverem fora de alcance uma da outra, não poderão se comunicar, ainda que “vejam” outras estações. Com efeito, contrariamente ao modo infraestrutura, o modo ad-hoc não propõe um sistema de distribuição capaz de transmitir os pacotes de dados de uma estação à outra. Assim, um IBSS é por definição uma rede sem fios restrita (RAPPAPORT, 2009; MARQUES, 2004).

2.6. PLATAFORMA ARDUINO

O Arduino surgiu em 2005, na Itália, com um professor chamado Massimo Banzi, que queria ensinar eletrônica e programação de computadores a seus alunos de design para que eles o utilizassem em seus projetos de arte, interatividade e robótica. Porém, ensinar eletrônica e programação para pessoas que não são da área não era uma tarefa simples, além da dificuldade da inexistência de placas poderosas e baratas no mercado. Foi pensando nisso que Massimo e David Cuartielles decidiram criar sua placa própria com a ajuda do aluno de Massimo, David Mellis, que ficou responsável por criar a linguagem de programação do Arduino. Várias pessoas conseguiram utilizar o Arduino e fazer coisas incríveis, surgindo assim essa febre mundial da eletrônica (MCROBERTS, 2005).

O Arduino é uma família de microcontroladores (pequenos computadores) e um ambiente de criação de software que facilita a criação de programas (chamados de *sketches*) que podem interagir com o mundo físico. Os projetos feitos com o Arduino podem perceber e responder ao toque, som, posição, calor e luz. Este tipo de tecnologia, muitas vezes referida como computação física ou embarcada, é utilizada em todos os tipos de sistemas, desde um aparelho de telefone celular até sistemas eletrônicos de automóveis. Essa plataforma de desenvolvimento torna possível qualquer pessoa, com um mínimo de conhecimento em programação ou eletrônica, utilizar esta tecnologia rica e complexa que pode interagir com o ambiente por meio de hardware e software (MARGOLIS, 2012).

A programação do Arduino é feita mediante a utilização de uma variante da linguagem de código aberto *Wiring*, específica para microcontroladores. Ela permite escrever software multiplataforma para controlar dispositivos conectados a uma ampla gama de placas de microcontroladores para criar todos os tipos de codificação criativa, objetos interativos, espaços ou experiências físicas (MCROBERTS, 2005). São expostas aqui apenas as funções e variáveis de manipulação padrão de valores relativos aos pinos do microcontrolador (considerando que nenhum *shield* esteja instalado). Uma vez que tal linguagem é um dialeto do C/C++, valem para tal a maioria dos valores e funções herdados da mesma. A referência completa dos comandos possíveis pode ser encontrada no site do fabricante oficial, no seguinte endereço: http://arduino.cc/en/Reference/HomePage (ARDUINO, 2014b).

A composição básica da linguagem de programação do Arduino é bastante simples. Os *sketches* podem ser divididos em três partes principais: estrutura, valores (variáveis e constantes) e funções. A estrutura é formada por apenas dois blocos de funções que carregam outros blocos de sub-rotinas escritas em linguagem C/C++. O primeiro bloco forma a função **<setup()>** e, o segundo, a função **<loop()>**. (ARDUINO, 2014b). O quadro 1 mostra os detalhes da estrutura de um *Sketche*.

ESTRUTURA DE UM SKETCHE	
FUNÇÃO	**DESCRIÇÃO**
setup()	A primeira função a ser chamada quando o programa inicia, sendo executada apenas uma vez. É uma função de preparação, ditando os comportamentos dos pinos e inicializando a porta serial do Arduino.
loop()	Todas as funções nela contidas são repetidamente executadas. Fica o tempo todo lendo os pinos de entrada do Arduino, comandando os pinos de saída e a porta serial.

Quadro 1 - Estrutura de Um Sketche.

Fonte: Arduino (2014b).

Os pinos digitais estão marcados com o nome DIGITAL, numerados da direita para a esquerda e podem ser configurados pela função <pinMode()> para detectarem ou transmitirem níveis lógicos digitais (verdadeiro/falso, 1/0 ou HIGH/LOW). Alguns deles também podem ser usados para gerarem sinais analógicos com a função <analogWrite()> utilizando a técnica de Modulação por Largura de Pulso - (PWM) – (ARDUNO, 2014b). O quadro 2 mostra as funções de manipulação das portas digitais e as suas principais características.

FUNÇÃO	DESCRIÇÃO	EXEMPLO
digitalRead(pino)	Uma vez configurado um pino com a função piMode(), a informação presente nesse pino pode ser lida com a essa função e armazenada numa variável qualquer.	int bt1 = digitalRead(10);
digitalWrite(pino,valor)	Para enviar um nível lógico a qualquer pino digital do Arduino, utiliza-se essa função. Dois parâmetros são requeridos: o número do pino e o estado lógico (HIGH/LOW) em que esse pino deve permanecer.	digitalWrite(2,HIGH);
pinMode(pino, modo)	Serve para estabelecer a direção do fluxo de informações em qualquer dos pinos digitais. Dois parâmetros devem ser passados à função: o primeiro indica qual pino vai ser usado; o segundo (INPUT/OUTPUT), se esse pino vai ser entrada ou se vai ser saída dessas informações e sempre deve ser escrita dentro da função setup().	pinMode(10,OUTPUT);
pulseIn(pino,valor,espera)	Essa função mede a largura em microssegundos de um pulso em qualquer pino digital. O parâmetro 'valor' diz à função que tipo de pulso deve ser medido, se HIGH ou LOW. O parâmetro 'espera' (time out) é opcional e se passado à função faz com que a medida do pulso só comece após o tempo em microssegundos ali especificado.	pulseIn(4,HIGH);
analogWrite(pino,valor)	O Arduino pode gerar tensões analógicas em vários de seus pinos digitais com a função analogWrite(). Dois parâmetros devem ser passados à função: o primeiro indica em qual pino será gerada a tensão; o segundo determina a amplitude dessa tensão, e deve ter valores entre 0 (para 0 volt) e 255 (para 5 volts).	analogWrite(10,128);
attachInterrupt(pino,função,modo)	Essa função é uma rotina de serviço de interrupção, ou ISR (Interrupt Service Routine) em inglês. Toda vez que ocorrer uma interrupção por hardware no pino digital 2 ou no 3 do Arduino uma outra função, criada pelo programador, vai ser chamada. O terceiro parâmetro, modo, informa como a interrupção vai ser disparada, se na borda de subida do pulso detectado no pino do Arduino, se na borda de descida, se quando o pulso for baixo ou se na mudança de nível desse pulso.	attachInterrupt(0,contador,RISING);

Quadro 2 - Funções de Manipulação das Portas Digitais.
Fonte: Arduino (2014b).

Os pinos analógicos estão dispostos em uma só barra com o nome ANALOG IN, localizada no lado oposto às barras dos pinos digitais. São numerados de 0 a n, da esquerda para a direita. Esses pinos são usados para leitura de sinais analógicos de sensores conectados ao Arduino e podem ser de quaisquer valores entre 0 e 5 V. Os pinos de entradas analógicas não precisam ser previamente configurados com a função pinMode(). O quadro 3 mostra as funções de manipulação das portas analógicas e as suas principais características.

FUNÇÃO	DESCRIÇÃO	EXEMPLO
analogRead(pino)	Essa função lê o nível analógico presente no pino indicado pelo parâmetro entre parênteses e, após a conversão para o seu equivalente em bits, o guarda em uma variável determinada pelo programador.	int sensor = analogRead(A0);
analogWrite(pino,valor)	Grava um valor analógico (PWM) em um pino. Após uma chamada a esta função, o pino irá gerar uma onda quadrada constante do ciclo de trabalho especificado até a próxima chamada para a mesma.	val = analogRead(A1); analogWrite(ledPin, val/4);
analogReference(tipo)	Configura a tensão de referência utilizada para a entrada analógica (ou seja, o valor utilizado como a parte superior do intervalo de entrada).	analogReference(EXTERNAL);

Quadro 3 - Funções de Manipulação das Portas Analógicas.
Fonte: Arduino (2014b).

É por meio das portas seriais que o Arduino se comunica, por meio de um cabo USB do tipo A → B, a um computador ou a outros dispositivos que tenham também uma interface USB e/ou serial. É através do conector USB que o Arduino recebe 5 V diretamente da fonte de alimentação do computador (ARDUINO, 2014b). O quadro 4 mostra as funções de manipulação das portas seriais e as suas principais características.

FUNÇÃO	DESCRIÇÃO	EXEMPLO
Serial.begin(taxa)	Essa função habilita a porta serial e taxa a taxa de transmissão e recepção em bits por segundo entre o computador e o Arduino.	Serial.begin(9600);
Serial.end()	Desabilita a porta serial para permitir o uso dos pinos digitais 0 e 1 para entrada ou saída de dados, devendo também ser escrita dentro da função setup().	Serial.end()
Serial.available()	Retorna o número de bytes disponíveis para leitura no buffer da porta serial.	int total = Serial.available();
Serial.read()	Lê o primeiro byte que está no buffer da porta serial.	int valor = Serial.read();

Serial.print(valor,formato)	Essa função envia para a porta serial um caracter ASCII, que pode ser capturado por um terminal de comunicação. O segundo parâmetro, 'formato', é	Serial.print(1.23456); // transmite 1.23 (default)
		Serial.print(1.23456,3); // transmite 1.234
		Serial.print("Alô Mundo!"); // transmite a frase (string)
		Serial.print('A'); // transmite o caracter A
		Serial.print('A',BIN); // transmite 01000001

	opcional e especifica com quantas casas decimais ou com que base numérica vai ser o número transmitido.	Serial.print('A',OCT); // transmite o octal 101
		Serial.print('A',HEX); // transmite o hexa 41
		Serial.print('A',DEC); // transmite o decimal 65

Serial.println(valor,formato)	Como a anterior, essa função envia para a porta serial um caractere ASCII com os mesmos parâmetros opcionais de 'formato', porém acrescenta ao final da transmissão o caracter Carriage Return (retorno ao início da linha) e o caracter New Line (mudança para a próxima linha).	Serial.println(1.23456); // transmite 1.23 (default)
		Serial.println(1.23456,3); // transmite 1.234
		Serial.println("Alô Mundo!"); // transmite a frase (string)
		Serial.println('A'); // transmite o caracter A
		Serial.println('A',BIN); // transmite 01000001
		Serial.println('A',OCT); // transmite o octal 101
		Serial.println('A',HEX); // transmite o hexa 41
		Serial.println('A',DEC); // transmite o decimal 65

Quadro 4 - Funções de Manipulação das Portas Seriais.
Fonte: Arduino (2014b).

Uma informação importante contida no manual do fabricante é que nem todos os pinos do Arduino Mega 2560 suportam mudanças de interrupção. Dessa forma, apenas as portas 10, 11, 12, 13, 50, 51, 52 e 53 podem ser utilizadas para receber dados (RX), entretanto, o mesmo possui 4 UARTs preparadas especificamente para esse fim.

Para conectar o Arduino a uma rede wireless, será utilizada, no presente trabalho, a biblioteca de software WiFlyHQ, escrita por Darran Hunt (2014) para integrar os módulos WiFly RN-XV da Roving Networks ao Arduino. A função dela é fornecer funções para configurar e gerenciar o módulo, o envio de pacotes UDP e o envio/recebimento de dados através da conexão TCP/IP. A sua forma de instalação e configuração consiste em efetuar o download da mesma do site do desenvolvedor por meio do seguinte link: <https://github.com/harlequin-tech/WiFlyHQ/zipball/master>, descompactá-la e copiá-la para o diretório "*libraries*" do local de instalação padrão da IDE do Arduino.

```
gilmar@je01:~$ cd Downloads/
gilmar@je01:~/Downloads$ wget https://github.com/harlequin-
tech/WiFlyHQ/WiFlyHQmaster.zip
gilmar@je01:~/Downloads$ unzip WiFlyHQmaster.zip
gilmar@je01:~/Downloads$ mv WiFlyHQmaster WiFlyHQ
gilmar@je01:~/Downloads$ sudo cp -R WiFlyHQ /usr/share/arduino/libraries/
```

Essa biblioteca tem algumas dependências que necessitam de serem resolvidas para que a mesma funcione corretamente. Em suas rotinas, ela faz chamadas as *libraries* WiFlySerial, NewSoftSerial, Streaming, Time e PString. A primeira pode ser encontrada no endereço <http://sourceforge.net/projects/arduinowifly/files/> e as demais, no próprio site do Arduino em <http://arduino.cc/en/Reference/Libraries>. A

forma de instalação delas segue o mesmo critério descrito acima para biblioteca Wi-FlyHQ.

Para desenvolver o sketche controlador do Arduino, é necessário ter instalado no computador um Ambiente Integrado de Desenvolvimento (IDE). De acordo com McRoberts (2011), um IDE nada mais é do que um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo. Geralmente as IDEs facilitam a técnica de Desenvolvimento Rápido de Aplicativos (RAD), cuja função principal é aumentar a produtividade dos desenvolvedores de aplicativos.

O IDE do Arduino é dividido em três partes: a *Toolbar*, no topo, o código ou a *Sketch Window* no centro, e a janela de mensagens na base. A *Toolbar* consiste de sete botões e sobre ela há uma guia, ou um conjunto de guias, com o nome do arquivo do *sketch*. Também há um botão posicionado no lado direito. Ao longo do topo há a barra de *menus*, com os itens *File*, *Edit*, *Sketch*, *Tools* e *Help*. Os botões da *Toolbar* (Figura 8) fornecem acesso conveniente às funções mais utilizadas dentro desses *menus* (MCROBERTS, 2011).

Nesse programa, há um monitor serial que é uma ferramenta muito útil, especialmente para depuração de código. O monitor exibe os dados seriais enviados/recebidos pelo computador ao Arduino e pode ser acessado por meio do botão “*Serial Monitor*”. No canto inferior direito é possível visualizar a taxa de transmissão (*Baud Rate*) na qual os dados seriais devem ser enviados de/para o Arduino. A taxa de transmissão é a quantidade por segundo em que alterações dos dados são enviados/recebidos. A configuração padrão é 9.600 baud (MCROBERTS, 2011).

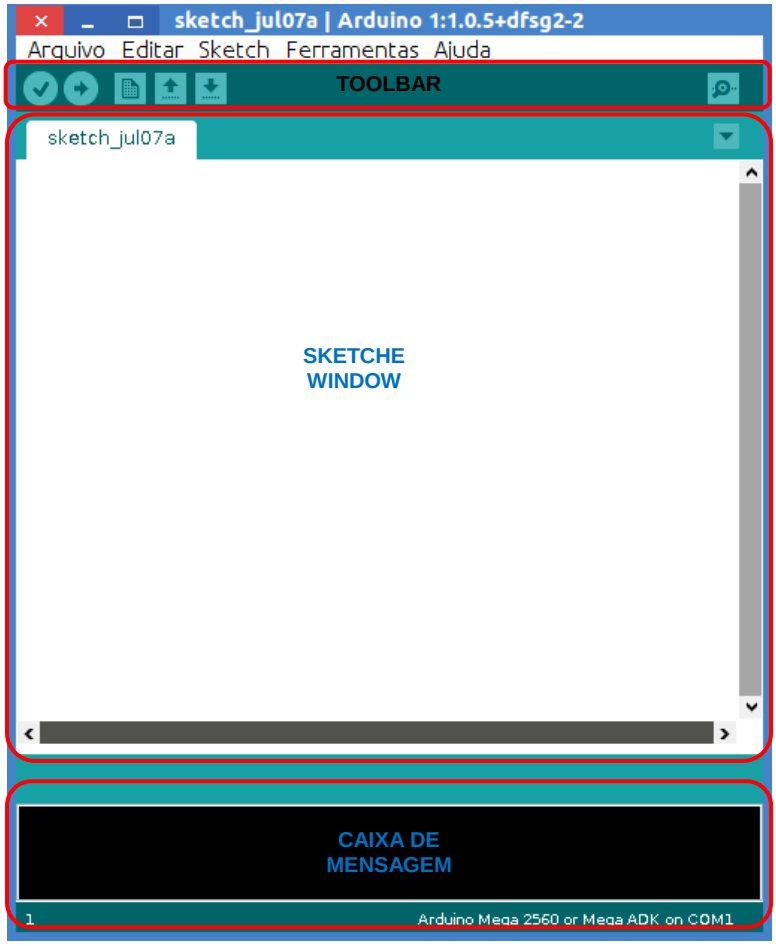


Figura 8 - Subdivisões da IDE de Programação Arduino.
Fonte: (Autor).

2.7. SISTEMA OPERACIONAL GOOGLE ANDROID

Darcey e Conder (2012), afirmam que a comunidade de desenvolvimento móvel está em um ponto de estabilização. Os usuários móveis querem mais opções, mais oportunidades para personalizar seus telefones e mais funcionalidades. Segundo esses autores, as operadoras de telefonia móvel querem fornecer conteúdo de valor agregado aos seus assinantes de forma gerenciável e lucrativa. Desenvolvedores móveis querem a liberdade de criar poderosas aplicações para usuários exigentes, com barreiras mínimas para o sucesso.

Fabricantes de aparelhos querem uma plataforma estável, segura e acessível para alimentar os seus dispositivos. Até 2011, segundo os autores acima citados, uma única plataforma móvel tratou adequadamente as necessidades de todas as partes. O Android é um potencial divisor de águas para a comunidade de desenvolvimento

móvel. Uma plataforma inovadora e aberta que está bem posicionada para atender às crescentes necessidades do mercado móvel.

Conforme Lecheta (2010), o Google Android é um sistema operacional baseado no núcleo do sistema operacional Linux, desenvolvido pela Open Handset Alliance (OHA), liderada pela Google. Segundo o autor, a plataforma é adaptada tanto para dispositivos VGA maiores, gráficos 2D, bibliotecas gráficas 3D baseadas em OpenGL/ES, especificação 2.0 e os *layouts* mais tradicionais de *smartphones*. É utilizado o SQLite para armazenamento de dados e tanto o SMS como MMS são formas disponíveis de envio de mensagens. O navegador web disponível no sistema é baseado no *framework* de código aberto conhecido como WebKit (LECHETA, 2010).

As aplicações são escritas na linguagem de programação Java e compiladas em bytecodes Dalvik, sendo executadas usando a *Dalvik Virtual Machine* (Máquina Virtual Dalvik) que é uma máquina virtual especializada e desenvolvida para uso em dispositivos móveis. Isso permite que programas sejam distribuídos em formato binário (bytecode), podendo ser executados em qualquer dispositivo Android independentemente do processador utilizado (LECHETA, 2010).

Apesar das aplicações Android serem escritas na linguagem Java, ela não é uma máquina virtual Java, já que não executa bytecode JVM. Atualmente, está disponível, em caráter experimental, uma nova máquina virtual chamada ART (*Android RunTime*) que promete otimizar o carregamento de aplicativos e está disponível apenas a partir da versão do Android 4.4 KitKat e superiores (LECHETA, 2010).

O sistema suporta formatos de áudio e vídeo como: MPEG-4, H.264, MP3 e Advanced Audio Coding (AAC). O Android é totalmente capaz de fazer uso de câmeras de vídeo, tela sensível ao toque, GPS, acelerômetros e aceleração de gráficos 3D. Inclui um emulador, ferramentas para *debugging*, memória e análise de desempenho. Além do Android SDK, o Eclipse IDE também pode ser utilizado para desenvolver aplicativos para essa plataforma por meio do plug-in ADT ou *Android Development Tools* (LECHETA, 2010).

2.8. LINGUAGEM DE PROGRAMAÇÃO EM BLOCOS APP INVENTOR

A cultura atual de consumo, conforme destaca Wolber (2011), oferece às pessoas todos os tipos de oportunidades de entretenimento, prazer e aprendizagem. No entanto, em geral, estas são atividades passivas. Segundo o autor, além do apelo ao consumo, há a satisfação de produzir: isto é, de criar. É a alegria e o orgulho resultantes quando o usuário tira uma foto, cria um modelo de avião, ou até mesmo assar pão.

Os objetos da alta tecnologia como telefones celulares, computadores, tablets, TVs, etc. que são usados hoje para consumir entretenimento e informação, são caixas pretas para a maioria das pessoas. Os programas tradicionais são incompreensíveis e, embora existam capacidades que permitam ao usuário tirar fotos, fazer vídeos, etc., não são em si mesmos mídias criativas. Em outras palavras, a maioria das pessoas não pode criar os aplicativos que rodam sobre estes aparelhos (WOLBER, 2011).

E se isso pudesse mudar? E se o usuário pudesse assumir o controle criativo dos seus aplicativos cotidianos dos telefones celulares? E se a construção de um aplicativo para o telefone celular for tão fácil quanto fazer um desenho ou assar um pão? E se fosse possível fechar a lacuna entre os objetos da cultura de consumo e os meios de comunicação da vida criativa de cada usuário? Por um lado, poderia desmistificar esses objetos. Ao invés de serem caixas-pretas, impenetráveis à visão das pessoas, esses dispositivos se tornariam objetos que poderiam ser consertados por elas mesmas (WOLBER et. al., 2011). Eles se tornariam objetos compreensíveis, com uma relação menos passiva e mais criativa para eles, conforme mostra a figura 9.

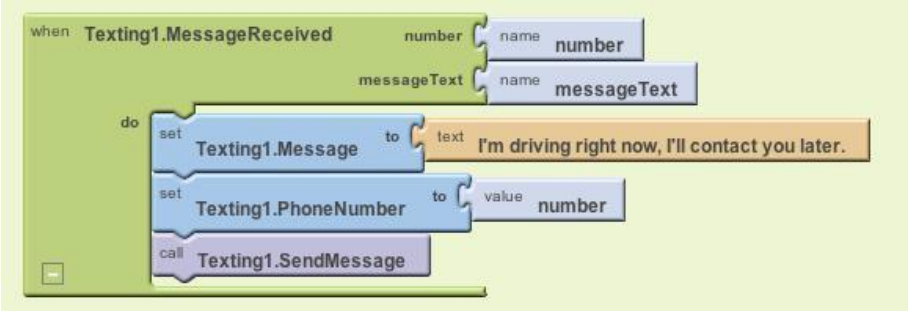


Figura 9 - Exemplo de Bloco de Programação App Inventor.
Fonte: Wolber et. al (2011).

Foi por causa desses questionamentos que surgiu o App Inventor, que é uma interface visual desenvolvido pelo Massachusetts Institute of Technology (MIT) em parceria com o Google. Tem por missão permitir que qualquer um possa programar seus próprios aplicativos, mesmo sem saber construir linhas de código e compilar programas de qualquer forma. O conjunto é composto por duas seções: o App Inventor Designer e o App Inventor Blocks Editor, cada uma com uma função específica (WOLBER, 2011). A figura 10 mostra o processo acima mencionado.

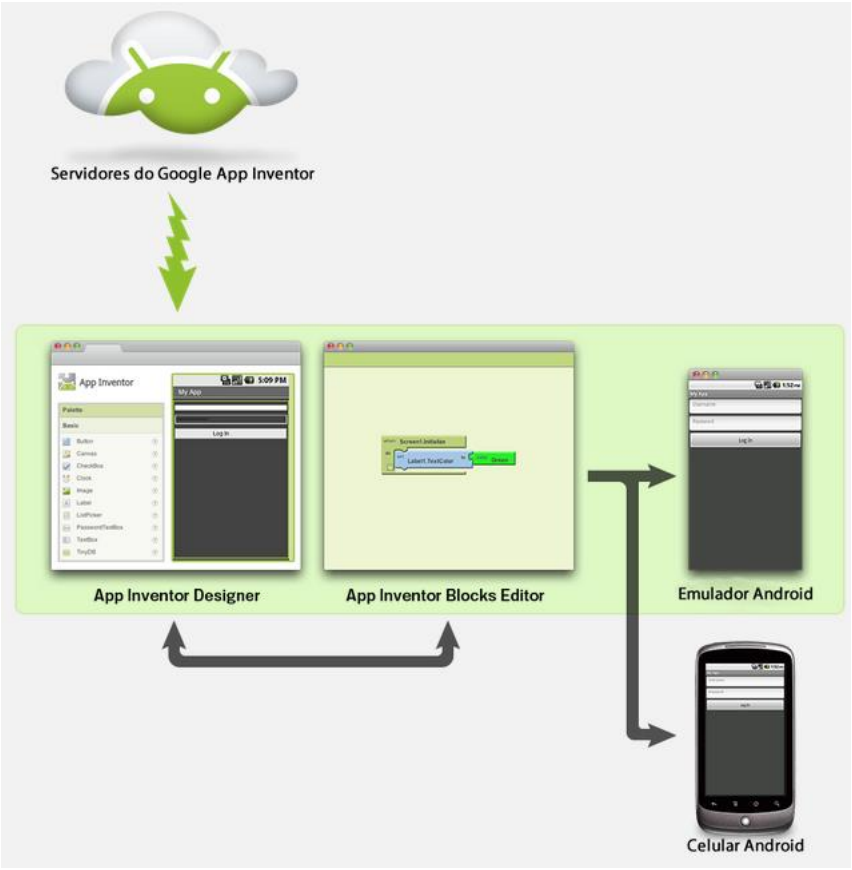


Figura 10 - Etapas de Desenvolvimento de Um Aplicativo App Inventor.

Fonte: MIT App Inventor. Disponível em: <http://appinventor.mit.edu/explore/content/what-app-inventor.html>. Acesso em: 17 jun. 2014.

Os aplicativos são construídos a partir de duas ferramentas de software, como mostra a figura 10, que possuem interface que pode ser usadas de maneira muito simples. A ferramenta que conecta o navegador (Google Chrome, Firefox, IE, etc) aos servidores do Google App Inventor e permite selecionar os componentes para a aplicação é denominada Inventor Design. A figura 11 mostra um exemplo de tela de autenticação desenvolvida pelo autor, utilizada no software do dispositivo móvel empregado neste trabalho, cujo *design* foi produzido no *Inventor Designer*.

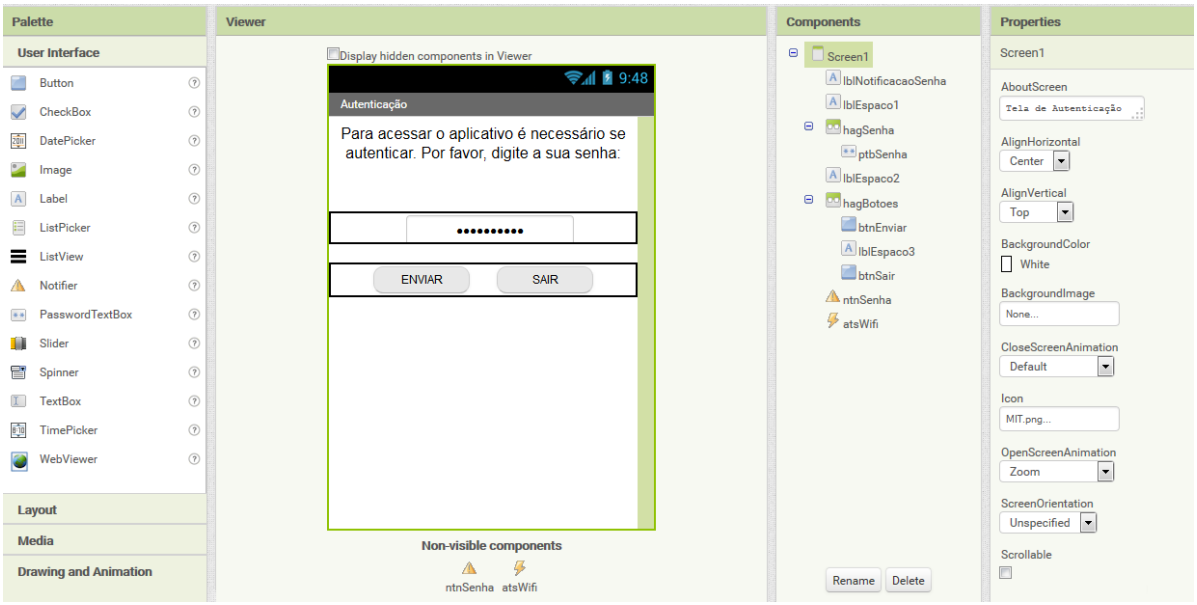


Figura 11 - Tela do App Inventor Design.
Fonte: (Autor).

O editor de Blocos se conecta ao Inventor Designer e permite especificar o comportamento dos componentes que são utilizados no aplicativo. Nele é feita a programação a partir de elementos visuais da mesma forma como montar um quebra cabeça. Os pré-requisitos para começar a criar aplicativos Android são: possuir uma conta do Google para entrar no App Inventor Designer, ter o Oracle JDK e o Android SDK instalados. A figura 12 mostra um exemplo de código em App Inventor.

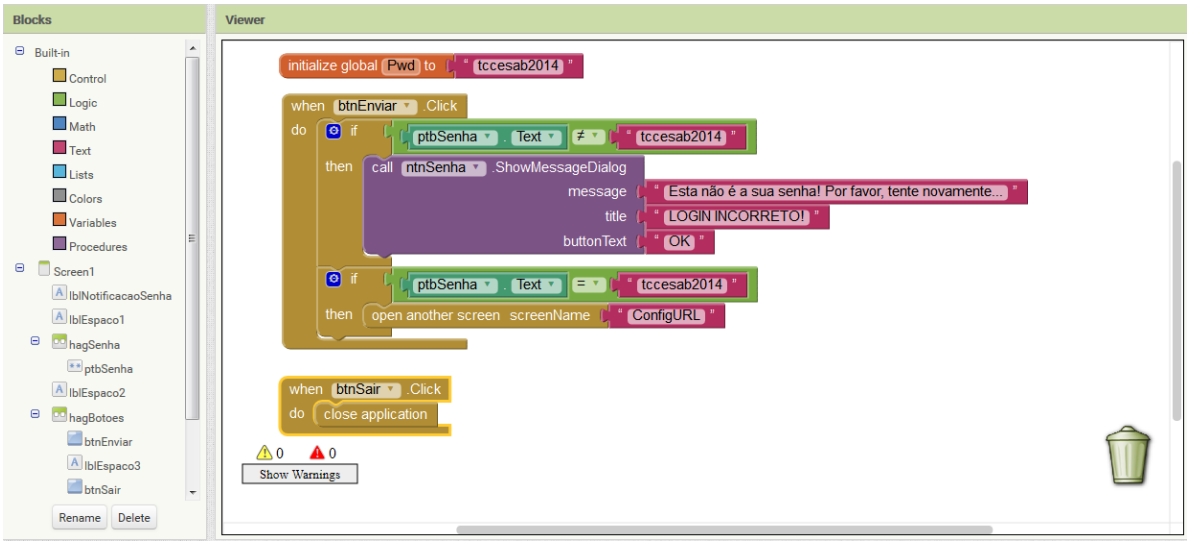


Figura 12 - Tela do Blocks Editor.
Fonte: (Autor).

2.9. MOTORES ELÉTRICOS TRIFÁSICOS DE INDUÇÃO

A maioria dos equipamentos, máquinas e sistemas da atualidade, segundo Carvalho (2011) tem como força motora uma máquina trifásica de indução. Essas máquinas têm suas partidas e paradas comandadas por diversos dispositivos cujas funções vão desde o simples ligar e desligar, do controle da sua corrente de partida até a regulação da velocidade de giro dos seus rotores. Por ser um motor de fácil partida, menos ruído e ser mais barato que as máquinas de indução monofásicas projetadas para trabalharem em potências superiores a 2 kW, são preferidos, de acordo com Franchi (2008), nos ambientes industriais e domésticos, uma vez que os atuais sistemas de distribuição de energia elétrica são trifásicos e em corrente alternada (AC).

2.9.1. TIPOS DE MOTORES TRIFÁSICOS DE INDUÇÃO

Por ser o tipo mais importante de motor aplicado nas indústrias, foram desenvolvidos diversos tipos de MITs. Segue abaixo as características dos principais tipos fabricados e utilizados pelas empresas (a figura 13 mostra os detalhes construtivos de um MIT rotor em gaiola de esquilo).

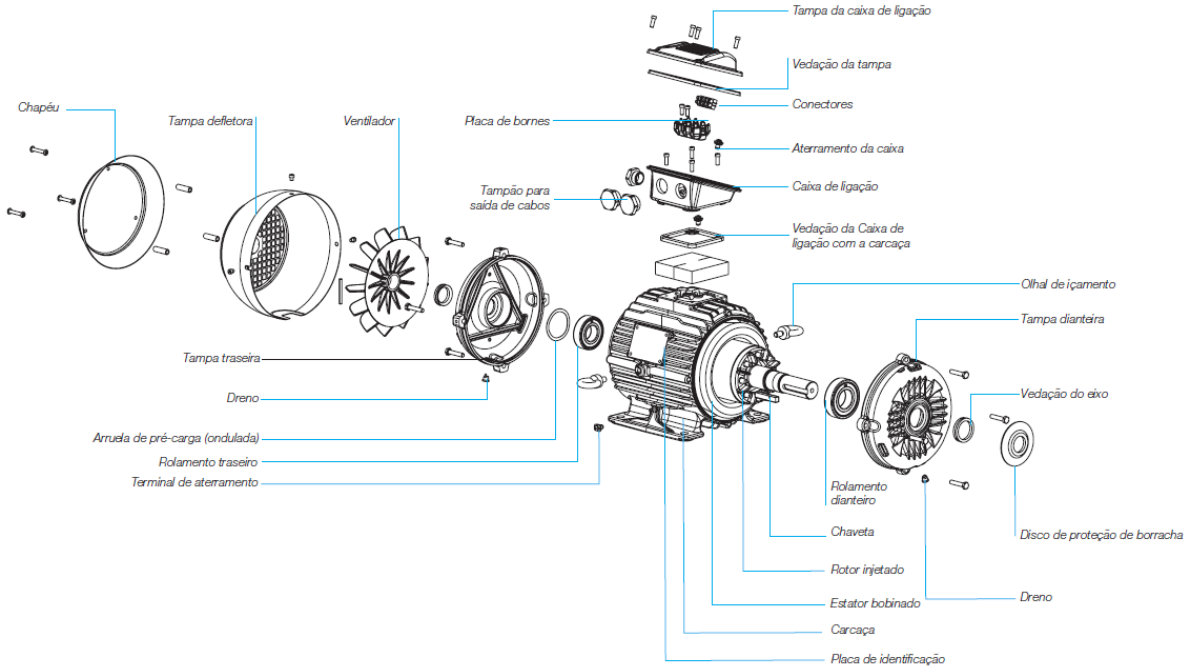


Figura 13 - Vista Explodida de Um MIT.
Fonte: (WEG. W22: Motor Elétrico Trifásico - Catálogo Técnico do Mercado Brasileiro). Disponível em: <http://www.weg.net/files/products/WEG-w22-motor-trifasico-tecnico-mercado-brasil-50023622-catalogo-portugues-br.pdf>. Acesso em: <21 jun. 2014, p. 48>.

O motor trifásico com rotor de gaiola possui, basicamente, um estator com enrolamento trifásico e um rotor tipo gaiola de esquilo. O enrolamento trifásico é responsável pelo campo girante, o rotor de gaiola sofre indução do campo e o tenta acompanhar. Como a velocidade do rotor é sempre menor que a do campo girante, o motor é chamado de assíncrono e a diferença percentual entre a velocidade do campo e do rotor é chamada de escorregamento. No estator são montados grupos de bobinas para cada fase, responsáveis pelo campo magnético girante trifásico dentro da máquina. As bobinas são posicionadas estrategicamente de modo que os arranjos de entrada das fases tenham defasagem de 120º entre elas (CARVALHO, 2011).

A montagem do enrolamento é efetuada de acordo com o número de ranhuras, numero de polos e a potência desejada, podendo ainda ser imbricada ou meio imbricada (CARVALHO, 2011). O enrolamento imbricado possibilita, com a mesma carcaça, obter um aumento na potência do motor. O rotor do tipo gaiola-de-esquilo é montado sobre um eixo que gira dentro do campo magnético girante, suportado por rolamentos instalados nas extremidades do eixo. Na parte traseira do motor há uma

ventoinha instalada cuja função é direcionar ar entre as aletas da carcaça do motor, ajudando a resfriá-lo (CARVALHO, 2011).

Já o motor trifásico com rotor bobinado foi desenhado para atender a uma necessidade específica de partida suave da indústria. A finalidade do rotor bobinado era permitir que fossem inseridas resistências em série com o enrolamento trifásico, controlando a velocidade imprimida. Tem-se, portanto, um enrolamento trifásico no estator e um enrolamento com três saídas no rotor. O contato entre o rotor e o meio externo é feito par escovas conectadas a três anéis fixos no eixo da máquina aos quais estão ligadas as três terminações do bobinado do rotor (CARVALHO, 2011).

O rotor bobinado possui uma impedância maior que a do rotor em gaiola de esquilo por constituir-se de enrolamento de fio de cobre. Com isso, a corrente de partida é menor para desempenhar o mesmo torque que o motor anteriormente mencionado. Esse enrolamento fica disponível na saída por meio de três anéis e permite a inserção de resistências em serie, o que aumenta a impedância do rotor a valores desejados. Com isso, obtém-se o controle da velocidade do motor na partida, suavizando-a, sem prejudicar de maneira extrema o conjugado da máquina (CARVALHO, 2011).

De acordo com Franchi (2008), as opções de velocidade em um motor podem ser conseguidas com a instalação de inversores de frequência, mas em alguns casos onde não há necessidade de variação de velocidade linear, mas apenas duas opções de velocidade é possível utilizar motores com duas velocidades. Há dois tipos de motores que atendem a essa solicitação: *Dahlander* e os de dois enrolamentos.

O motor *Dahlander* é um motor elétrico trifásico que permite o seu acionamento em duas velocidades distintas. As velocidades, que estão relacionadas ao número de rotações no motor, são conseguidas com a estruturação dos enrolamentos do estator deste motor em dois conjuntos promovendo uma relação de 1:2. Em uma forma de ligação, o número de polos é duas vezes maior que a outra, comportamento este que faz a velocidade também ser sempre o dobro da primeira. O Motor *Dahlander* tem em seu estator seis bobinas que podem ser combinadas de duas formas: estrela/triângulo e dupla estrela (CARVALHO, 2011).

Quando a quantidade de polos é maior a velocidade é mais baixa, quando é menor a velocidade é mais alta. Isso decorre da formula: n = 120 x f x (1-s) / p, quando a freqüência é 60 Hz (onde n = velocidade, p o número de polos, s = escorregamento e f a freqüência). Existem três tipos de arranjos de ligação que fornecem três situações: conjugado constante, potência constante e conjugado variável. A escolha depende do tipo de carga que será acionada. Por exemplo: nas bombas centrífugas e ventiladores o conjugado aumenta quadraticamente com a velocidade, portanto é variável (CARVALHO, 2011).

O motor de dois enrolamentos, como o próprio nome sugere, possui dois enrolamentos separados no mesmo estator, porém, com número de polos diferente. Pode-se encontrar motores com dois enrolamentos de 4/6 e 8/12 polos, normalmente com relação 1:1,5. Para selecionar a velocidade de trabalho basta alimentar o enrolamento correspondente e deixar o outro aberto e isolado. Normalmente os dois enrolamentos são fechados intimamente em estrela, portanto são construídos para uma única tensão nominal. Uma consideração importante é que jamais se deve confundir um motor *Dahlander* com um motor de dois enrolamentos, pois normalmente não é necessário fechamento externo em estrela no motor de dois enrolamentos e se for feito por engano, causará danos ao motor (CARVALHO, 2011).

3. MATERIAIS E MÉTODOS

Neste capítulo, apresenta-se o aparato empregado na presente pesquisa. Os materiais e *softwares* utilizados são listados em 3.1 e 3.2 e os procedimentos metodológicos realizados no desenvolvimento deste trabalho são descritos detalhadamente no capítulo subsequente.

3.1. LISTA DE MATERIAIS

No projeto do acionamento do MIT foram utilizados alguns dispositivos pré-existentes a exemplo do aparelho celular, do laptop e do laboratório de instalações elétricas industriais. Os demais equipamentos e componentes necessitaram ser adquiridos e/ou construídos, conforme itens subsequentes, pelo autor do presente trabalho. Os preços de cada um foram pesquisados no mercado de eletrônicos e seus valores se referem ao que era praticado no mês de março do ano de 2014 (dois mil e quatorze).

3.1.1. HARDWARE

O quadro 5 relaciona todos os componentes de hardware utilizados bem como os seus custos de aquisição para o presente trabalho. Os itens marcados com traço nas colunas referentes aos preços significam que o autor já dispunha dos mesmos na data de início do presente trabalho.

ÍTEM	DESCRIÇÃO	QTD	UNID.	PREÇO UNIT. (R$)	PREÇO TOTAL (R$)
1	Placa Arduino Mega 2560 rev3 (original)	1	UNID.	179,00	179,00
2	Wi-Fi Shield (WiFly RN-XV)	1	UNID.	239,00	239,00
3	Relé Tiambo 7 A - 5 V/110/220 V	1	UNID.	9,00	9,00
4	Telefone Celular Samsung Galaxy Y GT-S6102B	1	UNID.	-	-
5	Laptop Itautec InfoWay Note W7655	1	UNID.	-	-
6	Roteador Wireless Netgear WGT624 v4	1	UNID.	-	-
7	Diodo 1N4007	1	UNID.	0,75	0,75
8	Transistor BC337	1	UNID.	0,20	0,20
9	Resistor 1kΩ	1	UNID.	0,20	0,20
10	Resistor 470 Ω	1	UNID.	0,20	0,20
11	LED verde 5 mm	1	UNID.	0,70	0,70
12	Borne 3 pnos	1	UNID.	1,25	1,25
13	Motor trifásico de indução Datapool, 12 terminais, 1 CV, 220 / 380 V, 4 polos, 1725 rpm	1	UNID.	-	-
14	Contator tripolar, 50/60 Hz, 20 A, Cat AC-3/220 V, IP20	1	UNID.	-	-
15	Disjuntor tripolar, termomagnético, 50/60 Hz, 20 A / 4,5 kA, Curva C	6	UNID.	-	-

16	Fusível Diazed Retardado 25/500 A - 700 V_{CA}	3	UNID.	-	-
17	Relé Térmico Bimetálico 30-40 A / 680 V_{CA} / 50/60 Hz – Classe Térmica 10	1	UNID.	-	-
18	Lâmpada de sinalização vermelha 220 V	1	UNID.	-	-
19	Brocas para circuito impresso	3	UNID.	25,00	75,00
20	Esponja de aço	1	UNID.	-	-
21	Folha de papel Glossy Paper	1	UNID.	5,00	5,00
22	Ferro de solda	1	UNID.	-	-
23	Ferro de passar roupas	1	UNID.	-	-
24	Impressora a Laser ou XEROX	1	UNID.	-	-
25	Micro Retífica	1	UNID.	-	-
26	Placa de fenolite virgem	1	UNID.	3,00	3,00
27	Percloreto de ferro	1	UNID.	10,00	10,00
28	Solda eletrônica	1	UNID.	-	-
29	Suporte para placa de circuito impresso Tesoura	1	UNID.	-	-
30	Vasilhas plásticas ou garrafas PET	1	UNID.	-	-

Quadro 5 - Lista dos Materiais Utilizados no Projeto.
Fonte: Resultado da Pesquisa (2015).

3.1.2. SOFTWARES

Para o desenvolvimento do *sketch* necessário ao funcionamento do Arduino e do App para o aparelho celular, fez-se necessário o uso de diversos aplicativos listados no quadro 6. A escolha da plataforma de *softwares* livres foi devido à necessária diminuição de custos relacionados às licenças de uso de programas proprietários, além da possibilidade de modificação e adequação às necessidades específicas do projeto sem a preocupação em obter licenças específicas para tal. A definição de *software* livre, na visão de Nemeth (2004), é qualquer programa que pode ter seu código fonte alterado por qualquer usuário, sem a imposição de licença legal para sua distribuição. O presente trabalho foi desenvolvido utilizando o sistema operacional Ubuntu Linux 14.04 LTS Desktop.

ÍTEM	DESCRIÇÃO	VERSÃO	FUNÇÃO	PÁGINA WEB DO SOFTWARE
1	Ubuntu Linux	14.04 LTS i386 Desktop (kernel 3.11.0-19-generic)	Sistema operacional sobre o qual são executados os aplicativos para desenvolvimento dos *sketches* e do *App* para o aparelho celular.	www.ubuntu.com
2	Arduino IDE	1.0.6	Interface para escrita e depuração dos *sketches* da plataforma Arduino.	www.arduino.cc
3	Oracle Java JDK	8.0.510	Fornecer bibliotecas e outros componentes necessários para o computador executar a IDE do Arduino, *Applets* Java do MIT AppInventor, do Android SDK Tools e do Eclipse.	www.java.com
4	MIT AppInventor	1.0 e 2.0	Interface para criação de aplicativos com base em programação em blocos para o sistema operacional Android.	www.appinventor.mit.edu
5	Google Android	2.3.6 (kernel 2.6.35.7)	Sistema operacional executado pelo aparelho celular.	www.android.com

6	Android SDK Tools	1.16	Interface de desenvolvimento de aplicativos em Java para o sistema Android.	http://developer.android.com
7	Linux Terminal	4.52	Administração do SO e configuração de parâmetros internos do Wi-Fi Shield.	Nativo do sistema operacional.
8	CADe SIMU	1.0	Desenho e simulação do circuito de acionamento de potência do MIT.	http://personales.ya.com/canalPLC

Quadro 6 - Lista dos Softwares Utilizados no Projeto.
Fonte: Resultado da Pesquisa (2015).

3.2. DESENVOLVIMENTO E TESTES DO SISTEMA DE ACIONAMENTO

O objetivo desta seção é elucidar as etapas realizadas para a implantação do sistema construído. Os dispositivos físicos utilizados na implantação do sistema proposto foram especificados como descritos em 3.1 e configurados conforme segue.

3.2.1. ARDUINO MEGA 2560

O Arduino Mega 2560 é uma placa microcontrolada baseado no Atmega2560 (MICROSHIP, 2014a, 2014b). Ele tem 54 pinos de entrada/saída digital (dos quais 14 podem ser usados como saídas PWM), 16 entradas analógicas, 4 UARTs (portas seriais de hardware), um cristal oscilador de 16 MHz, uma conexão USB, um conector de alimentação, um leitor ICSP, um botão de reset e contém todos os componentes para suportar o microprocessador. Basta conectá-lo a um computador com um cabo USB ou ligá-lo com um adaptador AC/DC ou bateria para alimentá-lo.

Ele difere de todos os outros que o precederam na medida em que não utiliza o chip controlador FTDI para serial. Em vez disso, ele apresenta o ATmega16U2 (revisão r1 e r2 das placas) programado como um conversor USB-Serial. Ele é compatível com a maioria dos *Shields* projetados para o Arduino Duemilanove ou Diecimila (ARDUINO, 2014b) e possui uma considerável quantidade de portas, o que viabiliza a implementação de projetos mais complexos, garantindo a eficiência e o baixo custo dos projetos. A figura 14 mostra o mapeamento das portas do dispositivo aqui utilizado.

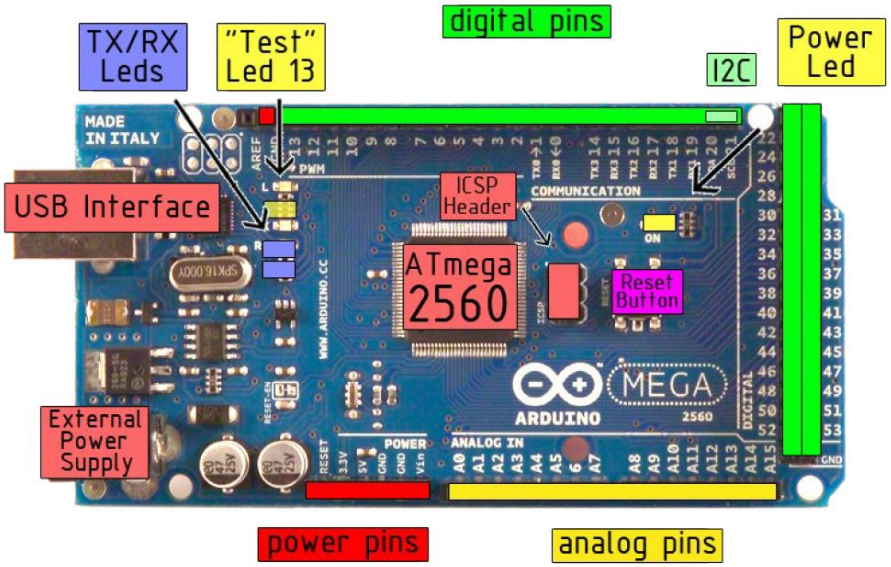


Figura 14 - Mapeamento das Portas do Arduino MEGA 2560.
Fonte: Arduino (2014b).

Conforme dados constantes no quadro 7, a placa utilizada no presente projeto tem o seguinte detalhamento técnico (informações fornecidas pelo fabricante e retiradas do *datasheet* do produto):

COMPONENTES	ESPECIFICAÇÕES
MICROCONTROLADOR	ATmega2560
TENSÃO DE OPERAÇÃO	5 V_{CC}
TENSÃO DE ENTRADA (RECOMENDÁVEL)	7-12 V
TENSÃO DE ENTRADA (LIMITES)	6-20 V
PINOS DIGITAIS (I/O)	54 (14 SAÍDAS PWM)
PINOS DE ENTRADA ANALÓGICA	16
CORRENTE CC POR PINO I/O	40 mA
CORRENTE CC NO PINO 3,3 V	50 mA
MEMÓRIA FLASH	256 Kb (8 kB utilizados pelo bootloader)
MEMÓRIA SRAM	8 kB
MEMÓRIA EEPROM	4 kB
VELOCIDADE DO CRISTAL	16 MHz

Quadro 7 - Especificações Técnicas do Arduino MEGA 2560.
Fonte: Arduino (2014b).

O Arduino Mega 2560 pode ser alimentado através da conexão USB ou com uma fonte de alimentação externa. A fonte de alimentação é selecionada automaticamente e, caso seja externa (não USB), pode ser com um adaptador AC/CC ou uma bateria. O adaptador deve ser do tipo centro positivo e plugue 2,1 mm, conectado ao plugue de energia da placa. Uma bateria pode alimentar os pinos GND e V_{IN} do conector de alimentação. A placa pode operar com fornecimento externo de 6 a 20 V, mas se for fornecido menos de 7 V, o pino de 5 V pode fornecer tensão inferior a

isso e a placa pode ficar instável. Se for utilizado mais do que 12 V, o regulador de tensão pode superaquecer e danificar a placa. O intervalo recomendado é de 7 a 12 volts (ARDUINO, 2014b).

De acordo com as especificações contidas no manual do fabricante, os seguintes pinos podem ser utilizados para alimentar a placa:

a) V_{IN} → A tensão de entrada da placa Arduino, quando se está usando uma fonte de alimentação externa (ao contrário dos 5 V a partir da conexão USB ou outra fonte de alimentação regulada). Pode-se fornecer tensão a este pino através da tomada de energia desde que haja transformação e regulação correta;
b) 5 V → A fonte de alimentação regulada é usada para alimentar o microcontrolador e outros componentes na placa. Essa tensão pode ser originada em V_{IN} ou através de um regulador *on-board*, ou até mesmo ser fornecida por USB ou outra fonte de 5 V regulada;
c) 3,3 V → A oferta de 3,3 volts é gerada pelo regulador *on-board*. A corrente máxima permitida é de 50 mA;
d) GND → Pino de referência (terra).

Cada um dos 54 pinos digitais do Arduino Mega 2560 pode ser utilizado como uma entrada ou uma saída usando as seguintes funções: <pinMode()>, <digitalWrite()>, e <digitalRead()>. Eles operam com 5 V e cada pino pode fornecer ou receber um máximo de 40 mA, além de possuir um resistor *pull-up* interno, desconectado por padrão, de 20 a 50 kΩ. Além disso, alguns pinos têm funções especializadas, conforme segue:

a) Funções Seriais → Usadas para receber (RX) e transmitir (TX) dados TTL seriais. Os pinos 0 e 1 também são ligados aos pinos correspondentes da porta serial USB-TTL do chip ATmega8U2 (quadro 8).

ORDEM	PINO	FUNÇÃO
Serial Principal (0)	0	RX
	1	TX
Serial 1	19	RX
	18	TX

Serial 2	17	RX
	16	TX
Serial 3	15	RX
	14	TX

Quadro 8 - Funções Seriais Nativas do Arduino MEGA 2560.
Fonte: Arduino (2014b).

b) Interrupções Externas → Estes pinos, conforme dados constantes no quadro 9, podem ser configurados para disparar uma interrupção por um valor baixo, uma borda de subida ou de descida ou uma mudança de valor por meio da função <attachInterrupt()>.

PINO	FUNÇÃO
2	interrupt 0
3	interrupt 1
18	interrupt 5
19	interrupt 4
20	interrupt 3
21	interrupt 2

Quadro 9 - Funções de Interrupções Externas do Arduino MEGA 2560.
Fonte: Arduino (2014b).

c) PWM → Pode ser utilizada com os pinos de 0 a 13, visando prover saída PWM de 8 bits com a função <analogWrite()>.
d) SPI → É utilizada nos pinos 50 (MISO), 51 (MOSI), 52 (SCK) e 53 (SS). Estes pinos suportam comunicação SPI que, embora fornecido pelo hardware subjacente, não está inclusa na linguagem padrão do Arduino. Os pinos SPI também são divididos no cabeçalho ICSP, que é fisicamente compatível com as placas Arduino Duemilanove e Diecimila.
e) LED → Há um LED conectado ao pino digital 13. Quando o pino é de alto valor, o LED está ligado, quando o pino é desativado, ele apaga.
f) I^2C → Pinos 20 (SDA) e 21 (SCL). O suporte à comunicação I^2C (TWI) é feito mediante uso da biblioteca Wire, cuja documentação pode ser encontrada no site da mesma.

A placa aqui descrita também tem 16 entradas analógicas, cada uma das quais com 10 bits de resolução (1024 valores diferentes). Por padrão, elas medem até 5 volts, embora seja possível mudar o limite superior de sua faixa usando o pino AREF e a função <analogReference()>. Há ainda um par de outros pinos na placa com funções específicas:

a) AREF → Tensão de referência para as entradas analógicas. Usado com a função <analogReference()>.
b) RESET → Caso esse pino seja colocado em estado baixo, resetará o micro-controlador. Tipicamente usado para adicionar um botão de *reset* aos S*hields* que porventura forem adicionados à placa.

Como descrito anteriormente, o Arduino Mega 2560 tem uma série de facilidades para se comunicar com um computador, outro Arduino ou outros microcontroladores. Ele fornece quatro portas seriais UARTs para TTL (5 V). Um chip ATmega8U2 co-nectado via USB a um dos canais de borda de cada um desses pinos fornece uma porta COM virtual para o software instalado no computador (o sistema operacional Windows XP irá precisar de um arquivo *.inf, mas do Windows Vista em diante, será reconhecido sem a necessidade de quaisquer arquivos de *drivers* adicionais). Entre-tanto, as máquinas OS X e Linux o reconhecerão como uma porta COM automaticamente (MCROBERTS, 2011).

O IDE do Arduino inclui um *Serial Monitor* que permite o envio de dados simples de texto de/para a placa. Os LEDs RX e TX piscam quando os dados estão sendo transmitidos através do chip ATmega8U2 via conexão USB para o microcontrolador e sinalizam em relação à comunicação serial por meio dos pinos 0 e 1. Em teoria, qualquer um dos pinos digitais pode ser utilizado como porta serial bastando, para isso, fazer uso da biblioteca *SoftwareSerial*, todavia, necessitam suportar funções de interrupção para tal (ARDUINO, 2014b).

3.2.2. ROTEADOR WI-FI

O roteador Wi-Fi é um dispositivo que serve para compartilhar um único ponto de rede *Ethernet* entre vários outros dispositivos de comunicação que utilizam tecnolo-gias de transmissão sem fios IEEE 802.11. As redes Wi-Fi funcionam por meio de ondas de rádio, cujos sinais são transmitidos por meio desse tipo de equipamento que recebe os sinais, decodifica-os e os emite a partir de uma antena. Para que um dispositivo tenha acesso a esses sinais, é necessário estar dentro de um determina-do raio de ação conhecido como *hotspot*. (KUROSE; ROSS, 2010).

O dispositivo *wireless* Netgear WGT624 v4 é um equipamento que implementa os padrões IEEE 802.11b/g no intervalo de frequências de 2,4/2,5 GHz, com encriptação WEP e WPA/WPA2-PSK (TKIP/AES). Fornece conexão sem fio para os dispositivos que utilizam tal tecnologia de transmissão através de um meio de acesso de banda larga externa (via cabo ou modem ADSL) que normalmente é destinado ao uso por um único equipamento (NETGEAR, 2014.), conforme mostra a figura 15.

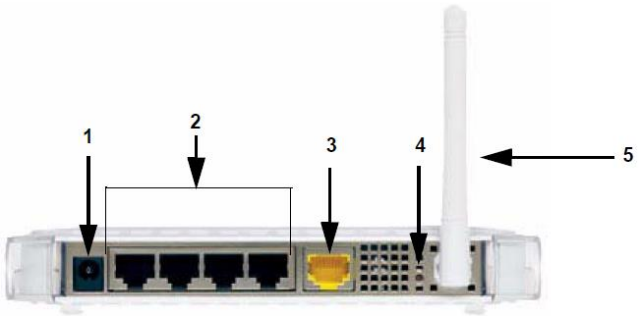


Figura 15 - Roteador Netgear WGT624 v4.
Fonte: Netgear (2014).

O equipamento mostrado acima e utilizado nesse trabalho é composto por uma entrada para suprimento de energia com alimentação (1), quatro portas LANs 10/100 Mbps para conexão em rede local de dispositivos por meio de cabos UTP terminados em conectores 10BASE-T ou 100BASE-TX RJ-45 (2), uma porta WAN para conexão à internet via cabo ou ADSL com terminação 10BASE-T ou 100BASE-TX RJ-45 (3), um botão para restaurar as configurações de fábrica do equipamento (4), uma antena com potência de transmissão de 2 dBi (5) e LEDs frontais cuja função é indicar o estado de funcionamento do mesmo.

Para proceder com o processo de configuração, o equipamento teve a sua configuração de fábrica restaurada, sendo providenciada comunicação via cabo UTP entre o roteador e o computador (não poderia ser wireless, uma vez que o dispositivo tem esse tipo de comunicação desabilitada na configuração de fábrica), conforme detalhamento das informações contidas no manual do fabricante do produto. É necessário conectar uma das pontas do cabo em quaisquer das portas LAN do equipamento e, sequencialmente, na porta do adaptar de rede do laptop.

Com o servidor DHCP nativamente habilitado e a configuração da interface de rede definida como automática, o *laptop* obteve o endereçamento IP disponibilizado pelo roteador. Caso isso não acontecesse, far-se-ia necessário configurar o endereçamento do *notebook* de forma manual, na mesma faixa de rede do equipamento Wi-Fi. Em seguida, foi realizado acesso às configurações padrões internas do roteador, via *brownser*, utilizando os seguintes dados:

```
Endereço: http://192.168.1.1 ou www.routerlogin.net
Login: admin
Senha: password
```

Conforme mostra a figura 16, a primeira pergunta ao acessar o equipamento é se o usuário deseja ativar a interface wireless (é necessário marcar o item “Yes” e clicar em “Next”). O roteador não irá detectar conexão com a internet, mas não há problema. A intenção desse trabalho é interfacear os equipamentos com conexão em rede local, portanto, como a rede sem fio configurada e funcional é suficiente, o *router* não será conectado à internet.

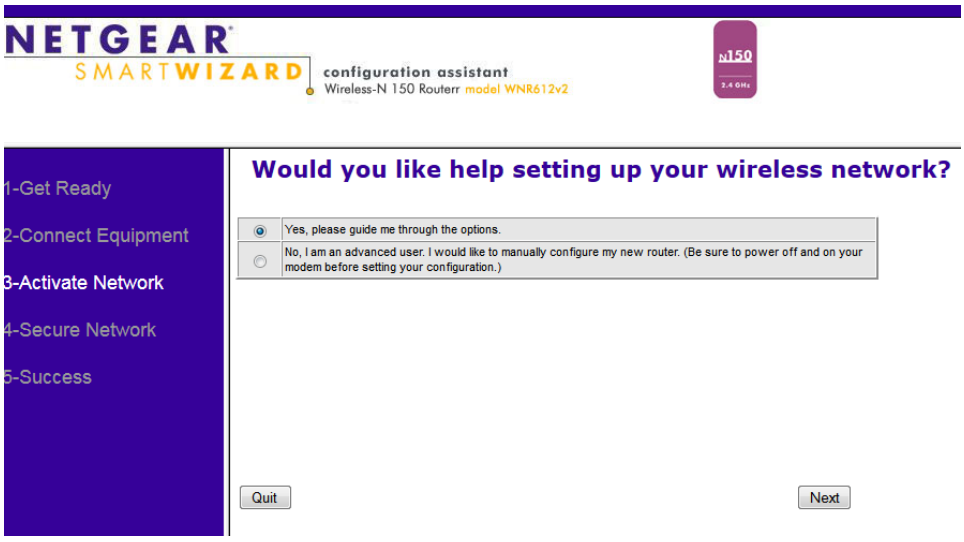


Figura 16 - Tela de Ativação do Wi-Fi no Roteador Netgear WGT624 v4.
Fonte: Resultado da Pesquisa (2015).

A interface do equipamento também possibilita a modificação do idioma da sua página para o português (inglês é o padrão). Após isso, a configuração que mais interessa a este trabalho é a do sistema *wireless*. Para melhor compatibilidade com o Wi-Fi Shield, não devem ser utilizados caracteres especiais nem no SSID nem na senha, além de não habilitar a transmissão em 150 Mbps (802.11N), visto que o pa-

drão citado não é suportado pelo módulo WiFly RN-XV. O quadro 10 mostra os parâmetros com os quais o roteador foi configurado.

PARÂMETROS	ESTADOS
MODO DE OPERAÇÃO	Sem acesso à internet
LINGUAGEM PADRÃO	Português
REGIÃO	América do Sul
TRANSMISSÃO SSID	Habilitado
NOME SSID	TCCEsab
CANAL	13
MODO	54 Mbps
TIPO DE CRIPTOGRAFIA DA SENHA	WPA-PSK (TKIP)
SENHA	tccesab2014

Quadro 10 - Parâmetros de Configuração do Roteador.
Fonte: Resultado da Pesquisa (2015).

Depois das configurações do quadro acima serem adicionadas, basta clicar no botão aplicar e o dispositivo habilitará a interface sem fio já com as definições determinadas aplicadas. A rede *wireless* já pode ser utilizada imediatamente. A figura 17 mostra como ficaram as definições atualizadas na *front-end* de configuração do roteador. Não se fizeram necessárias mudanças na faixa de endereçamento a qual o servidor DHCP do equipamento atribuiu aos seus clientes.

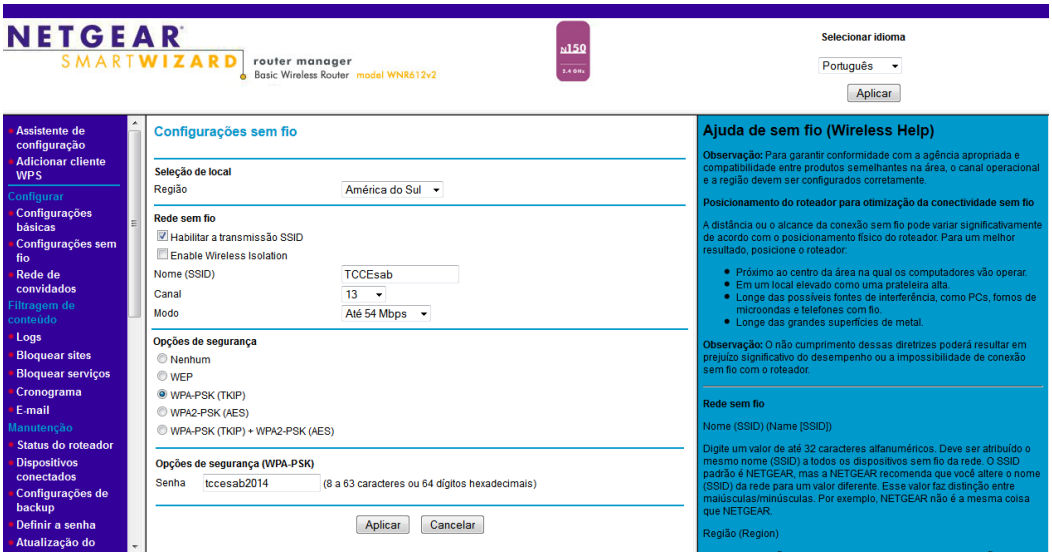


Figura 17 - Demonstração da Configuração Final do Roteador.
Fonte: Resultado da Pesquisa (2015).

3.2.3. WI-FI SHIELD

É composto de um módulo RN-XV que é fabricado pela empresa *Roving Networks*, do ramo de componentes eletrônicos para conectividade e pertencente à *Microchip Technology Inc*. Foi adquirido por meio do Laboratório de Garagem (figura 18) e permite uma rápida conexão com redes 802.11 b/g (Wi-Fi), tendo uma interface de comunicação serial por onde pode ser programado e trocar informações entre o módulo e o Arduino. Possui 8 pinos de entradas e saídas (I/O) de 3,3 V e 3 pinos para entrada de sensores analógicos (0 - 400 mV, com tolerância para 3,3 V). (MICROSHIP, 2014a).

Figura 18 - Módulo WiFly RN-XV.
Fonte: Microship (2014a).

Esse módulo pode ser utilizado, conforme Roving Networks (2011), em vários tipos de redes estruturadas com suporte aos protocolos *Dynamic Host Configuration Protocol* (DHCP), *Domain Name System* (DNS), *Address Resolution Protocol* (ARP), *Internet Control Message Protocol* (ICMP), *File Transfer Protocol* (FTP client), *Hypertext Transfer Protocol* (HTTP client), *Transmission Control Protocol / Internet Protocol* (TCP/IP) e *User Datagram Protocol* (UDP). A figura 19 mostra o diagrama elétrico do mesmo.

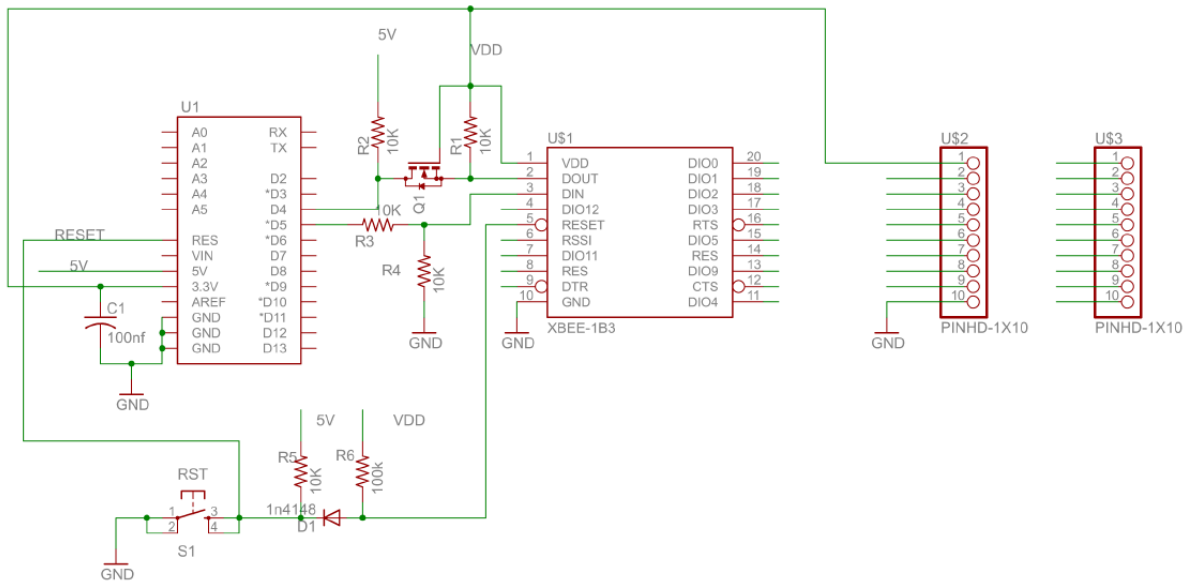


Figura 19 - Diagrama Elétrico do Módulo WiFly RN-XV.
Fonte: Ortega (2014).

O dispositivo também suporta *Wi-Fi Protected Setup* (WPS) com criptografias do tipo *Wired Equivalent Privacy* (WEP), *Wireless Application Protocol* (WAP e WAP2). Há três Diodos Emissores de Luz (LEDs) que indicam os estados de funcionamento do módulo conforme informações contidas no quadro 11, abaixo, sendo bastante útil na detecção imediata de eventuais problemas de comunicação.

LED	CONDIÇÃO			
	APAGADO	PISCANDO DEVAGAR	PISCANDO RÁPIDO	ACESSO
VERDE	-	ENDEREÇAMENTO IP CONFIGURADO CORRETAMENTE	SEM ENDEREÇO IP DEFINIDO	CONECTADO VIA TCP/IP
AMARELO	-	-	TRANFERINDO DADOS	-
VERMELHO	CONECTADO / ACESSANDO À INTERNET	CONECTADO / SEM ACESSO À INTERNET	SEM CONEXÃO	-

Quadro 11 - Indicativo do Estado de Funcionamento do Módulo WiFly.
Fonte: Microship (2014a).

A figura 20 mostra o esquema de interação entre o módulo e os demais componentes de uma rede *Ethernet*. Qualquer dispositivo de comunicação, desde que esteja na mesma rede do Wi-Fi Shield e seja dotado de tecnologia de transmissão TCP/IP pode interatuar com o mesmo. Tal processo se dá pela intermediação do roteador que trabalha fornecendo o canal de comunicação para que todos os equipamentos a ele associados possam estabelecer conexão entre si.

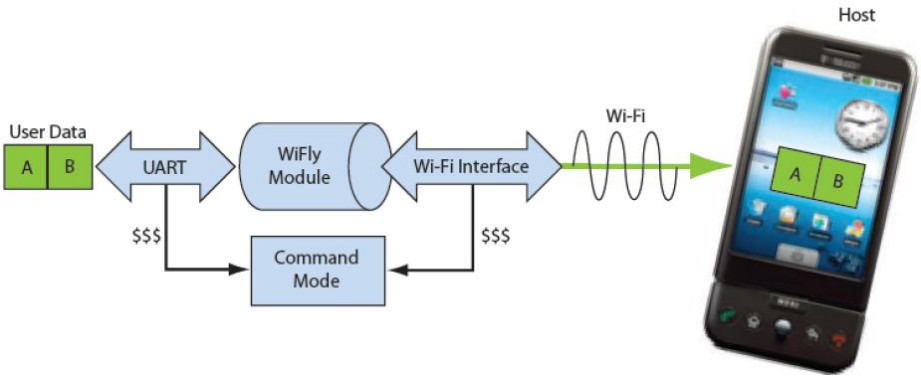


Figura 20 - Esquema de Comunicação do Módulo WiFly.
Fonte: Microship (2014b).

Existem dois modos de operação do módulo: modo de dados ou modo de comando. Quando está transferindo dados, está pronto para aceitar conexões de entrada ou saída. Para configurar os parâmetros e/ou exibir a configuração atual, o dispositivo deve ser colocado em modo de comando (também chamado de modo de configuração). A qualquer momento é possível entrar no modo de comando ou em modo de dados (ROVING NETWORKS, 2011).

Ao ligar, o módulo estará no modo de dados, entretanto, para entrar no modo de comando, deve-se enviar uma sequência de caracteres (conjunto de três caracteres, <$$$> por padrão). O dispositivo irá responder com CMD, indicando que está no modo de comando. Não deve ser utilizado o modo CARRIAGE RETURN (<cr>) antes do envio dos caracteres <$$$> para entrar no modo de comando. No entanto, será necessário usar o CR após cada comando enviado. Na configuração de fábrica, a comunicação serial é conseguida com velocidade de 9600 baud, 8 bits, sem paridade, 1 bit de parada.

Para testar a conexão entre o Arduino e o módulo foram efetuados os seguintes passos aqui descritos: Os pinos PWR (3,3 V), TX (pino 4), RX (pino 5) e GND do Wi-Fi Shield foram conectados, respectivamente, aos pinos de 3,3 V, RX (19), TX(18) e GND do Arduino. Como o Arduino Mega 2560 tem 4 UARTs, a UART 0 é composta pelos pinos 0 (RX) e 1 (TX) sendo utilizada pelo cabo USB para a transferência dos *sketches* do computador para a placa.

O módulo se comunica com o Arduino pelos pinos 4 (RX) e 5 (TX), porém, como esses pinos não têm suporte a mudança de interrupção, não podem ser utilizados para a comunicação serial. A solução para esse impasse foi utilizar uma das outras três UARTs disponíveis, jampeando os pinos correspondentes a UART 1 - pino 18 (TX) e 19 (RX). Dessa forma, o pino 18 (TX do Arduino) foi conectado ao pino 5 (RX do módulo) e o pino 19 (RX do Arduino), ao 4 (TX do módulo). A figura 21 mostra como ficou a montagem.

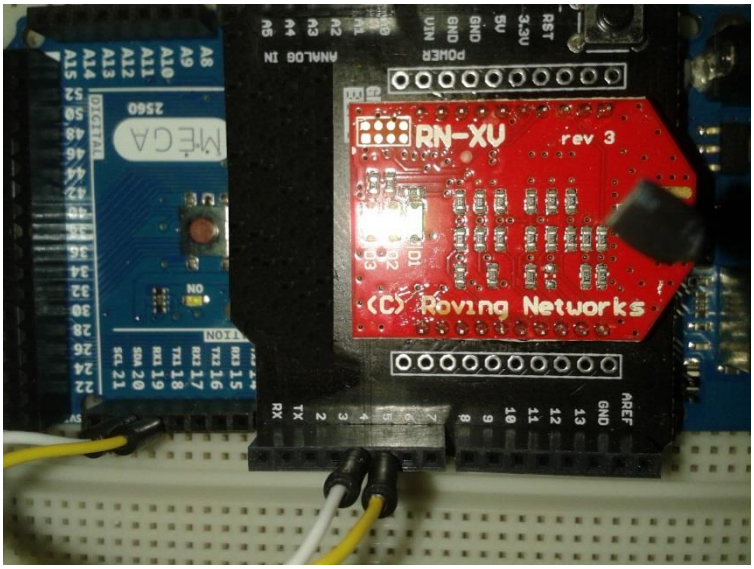


Figura 21 - Interligação da Comunicação Entre o Módulo WiFly e o Arduino.
Fonte: Resultado da Pesquisa (2015).

A placa do módulo deve ser encaixada exatamente sobre os pinos correspondentes do Arduino, sendo os pinos RST e AREF a principal referência para tal. Além disto, foi necessário desenvolver o *Sketch* abaixo e efetuar o *upload* do mesmo ao Arduino. Tal código tem unicamente o propósito de intermediar a comunicação entre o módulo Wi-Fi e o Arduino para fins de acesso ao modo de comando, portanto, das configurações do Wi-Fi Shield.

```
void setup() {
  Serial1.begin(9600); // UART, RX (D19), TX (D18) está conectado ao WiFly
  Serial.begin(9600);  // USB
}

void loop() {
  if(Serial1.available() > 0) // Se o pino RX (digital 19) receber a mensagem
  {
    Serial.write(Serial1.read()); // Escreva-a na porta USB
  }
  if(Serial.available() > 0) // Se a porta USB receber comandos
```

```
  {
    Serial1.write(Serial.read()); // Envie-os para o módulo WiFly
  }
}
```

O dispositivo foi posto para se comunicar com o Arduino à velocidade de 9600 baunds, entretanto, foi posteriormente configurado para trabalhar na velocidade de 115200 baunds. Neste momento, na placa do módulo, havia dois LEDs piscando: um Verde e um Vermelho. Após isso foi digitado os caracteres <$$$> e pressionado o botão <Send>, a resposta foi <CMD>, indicando a entrada no modo de comando. Neste momento o LED Verde estava piscando mais rápido. Inicialmente, o módulo vem com a configuração de fábrica, isto é, ao ser ligado, irá apresentar dois LEDs piscando intermitentemente. O LED vermelho, piscando rapidamente, indica que o módulo não está conectado a nenhuma rede e, o verde, que o mesmo está sem endereçamento IP definido. A figura 22 mostra o resultado.

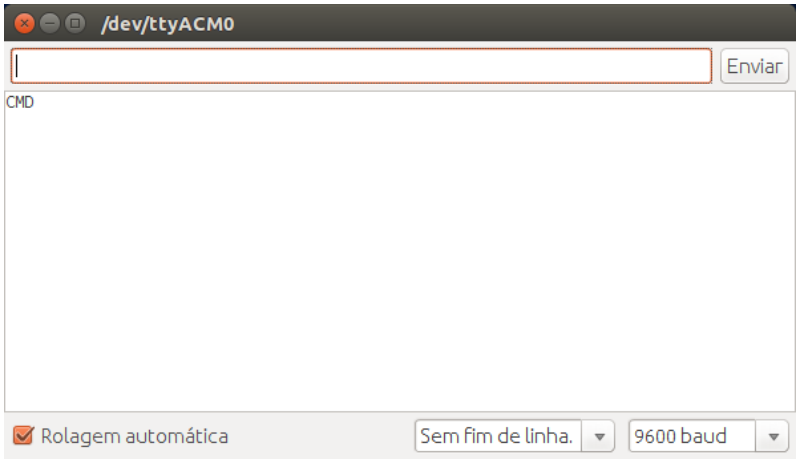


Figura 22 - Módulo WiFly no Modo de Comando.
Fonte: Resultado da Pesquisa (2015).

Então, o *Serial Monitor* deve ser aberto, colocado no modo "*No line ending*" com 9600 baund. Digitar a sequência de caracteres "$$$" e teclar <Enter>. O módulo deve responder "CMD" e os seus LEDs piscarão mais rápido; logo após, trocar o parâmetro "*No line ending*" por "*carriage return*" e digitar o comando <get wlan>. O módulo deve mostrar as informações predefinidas conforme configurações de fábrica (SPARKFUN, 2014).

A configuração propriamente dita começa com a mudança da velocidade da porta serial do módulo. Digitar "set uart baund 115200", "save" e "reboot". Essa nova velo-

cidade de transmissão do WiFly deve ser atualizada no *sketche*, editando-o, compilando-o e gravando-o na placa para que seja possível se comunicar com a mesma novamente. Depois disto, bastou digitar <scan> e enviar ao módulo. Este comando tem por função procurar pelas redes Wi-Fi disponíveis próximas e mostrar as informações referentes às mesmas. A figura 23 mostra o resultado desse comando.

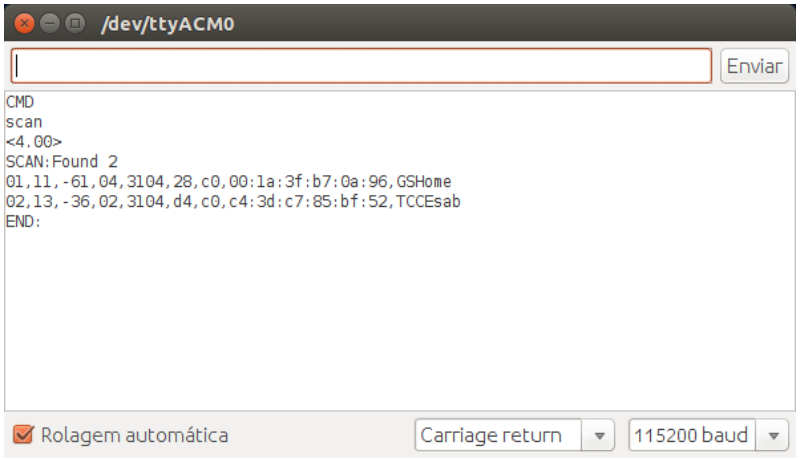


Figura 23 - Retorno do Comando Scan.
Fonte: Resultado da Pesquisa (2015).

Uma vez detectadas, para conectá-lo a uma rede Wi-Fi, bastou digitar os seguintes comandos abaixo, pressionando <Send> ao término de cada linha.

```
set wlan join <1>               // Associar-se automaticamente ao AP configurado
set wlan auth <3>               // Tipo de chave de segurança (WPA e WPA2-PSK)
set wlan channel <13>           // Canal de comunicação (conforme definido no AP)
set wlan ssid <TCCEsab>         // SSID do AP
set wlan phrase <tccesab2014>   // Senha da rede
set ip dhcp <0>                 // Protocolo DHCP desabilitado (IP estático)
set ip address <192.168.1.100>  // Endereço IP do módulo
set ip netmask <255.255.255.0>  // Máscara de rede
set ip gateway <192.168.1.1>    // Endereço IP do roteador
set dns address <192.168.1.1    // Endereço IP do servidor DNS
set uart baund <115200>         // Alteração da velocidade da UART
set ip localport <80>           // Alteração da porta padrão

save
reboot
```

A figura 24 mostra a resposta de cada comando recebido e processado corretamente. Após isso, o LED vermelho apagou, indicando que estava conectado com acesso à internet e o LED verde passou a piscar lentamente, informando que o endereçamento IP atribuído estava correto. Uma observação importante que cabe frisar nesse momento é que apesar do módulo ter sido configurado nesta seção para trabalhar

em conjunto com o Arduino à velocidade de 115200 baunds, foi posteriormente detectado que houve ruído na transmissão, o que ocasionou corrupção de dados. Para que isso não ocorresse, todos os sistemas foram sincronizados para se comunicarem a 9600 baunds.

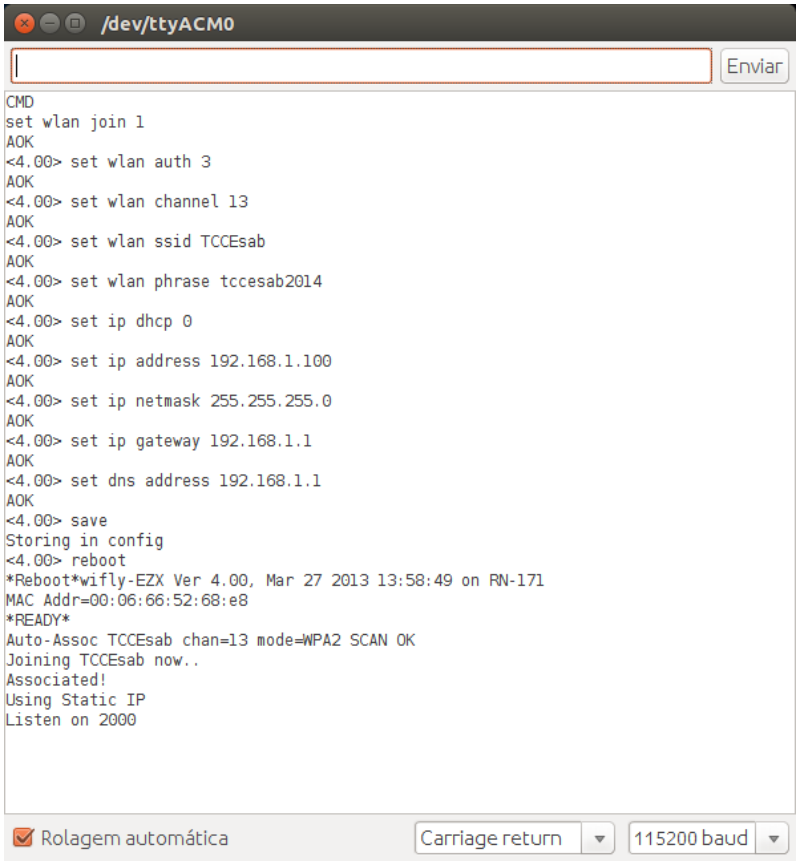


Figura 24 - Resultado dos Comandos de Configuração do Módulo WiFly.
Fonte: Resultado da Pesquisa (2015).

Uma vez configurado, o módulo fica disponível para novos ajustes também por meio do protocolo TCP/IP, via Telnet e escutando na porta 80 (a porta padrão é a 2000). É possível acessá-lo também por meio dos emuladores de terminal (o software TeraTerm é recomendado pelo fabricante, mas o terminal do Ubuntu Linux se mostrou compatível). As informações da rede sem-fio à qual o módulo se encontra conectado podem ser obtidas por meio de diversos comandos (a figura 25 mostra o resultado do comando <get wlan>). Segue abaixo os mais utilizados no presente trabalho.

```
get ip              // Mostra o endereço IP do dispositivo
get mac             // Mostra o endereço MAC do módulo
get wlan            // Mostra os parâmetros da rede Wi-Fi configurada
get uart            // Mostra as configurações da comunicação serial
get sys             // Exibe informações gerais sobre o módulo
```

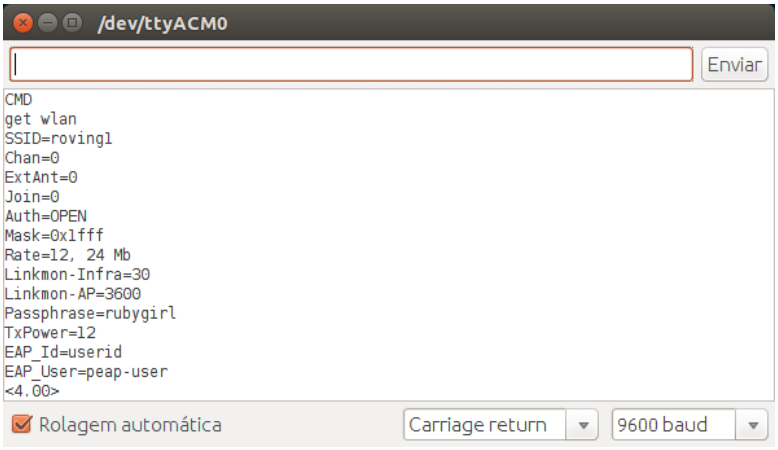


Figura 25 - Resultado do Comado Get WLAN.
Fonte: Resultado da Pesquisa (2015).

Finalizada esta etapa, o módulo fica ativo e pronto para receber conexões e repassá-las de/para o Arduino. Uma observação importante é que o dispositivo para de responder aos comandos enviados por meio do Serial Monitor imediatamente após a gravação de um novo sketche na memória flash do Arduino. Alterações nas configurações do WiFly podem ser feitas por meio de dispositivos que estejam na mesma rede e que disponham de emuladores de terminal com suporte aos protocolos TCP/IP e Telnet. No Ubuntu Linux ele pode ser acessado com o comando <telnet>.

```
gilmar@je01:~$ telnet 192.168.1.100 80
```

A certificação de que o mesmo está pronto para a transmissão de dados pode ser feita rodando o comando <ping>.

```
gilmar@je01:~$ ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100) 56(84) bytes of data.
64 bytes from 192.168.1.100: icmp_seq=1 ttl=255 time=1.57 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=255 time=1.58 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=255 time=1.77 ms
^C
--- 192.168.1.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.572/1.643/1.777/0.105 ms
```

3.2.4. CIRCUITO DE ACOPLAMENTO

O circuito de acoplamento foi construído com relés, contendo um contato reversível com capacidade de acionar cargas de até 7 A / 110 V_{AC} e podendo ser integrado ao Arduino. Para utilizá-lo, foi necessário construir tal circuito com componentes eletrônicos. Isto permite criar aplicações de controle de cargas externas como: luminárias, ventiladores, eletrodomésticos, portões de garagem, robótica, controle industrial, casas inteligentes, entre outros. A figura 26 mostra o diagrama eletrônico do mesmo.

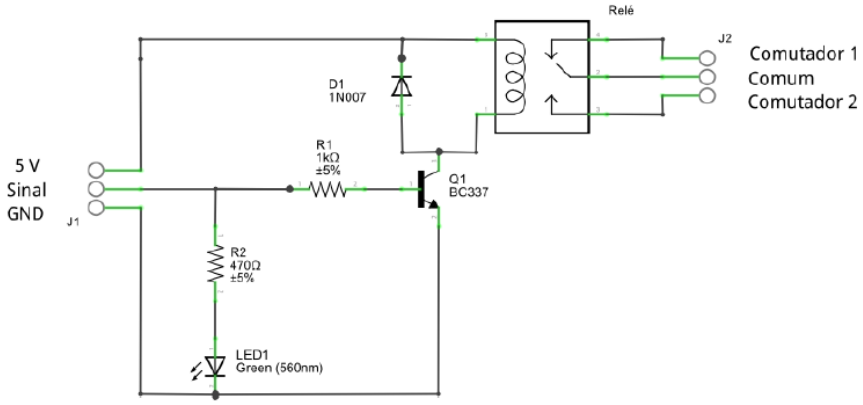


Figura 26 - Diagrama Elétrico do Circuito de Acoplamento.
Fonte: Resultado da Pesquisa (2015).

O circuito acima é controlado diretamente pelo Arduino através dos pinos de I/O digitais, devendo-se colocar o pino referente ao relé em modo *High* (nível alto). No presente circuito há um transistor BC337 funcionando como chave que, quando recebe tensão do Arduino, passa a conduzir e aciona o relé. Este, quando fechado, liga o circuito de força conectado ao mesmo – uma vez que o contato NA do mesmo está interligado à bobina de comando do contator. O diodo *zenner* ligado em paralelo ao relé funciona como componente de proteção, uma vez que, quando o relé for desativado, passará a induzir uma tensão instantânea alta que pode queimar o transistor.

A placa de circuito impresso do esquema acima foi confeccionada utilizando a sistemática exposta por Cunha e Vega (2014). Por ser um método prático e de baixo custo para confecção de projetos de circuitos eletrônicos, esta técnica foi utilizada no esboço e construção do circuito de acoplamento do presente trabalho. Para o dese-

nho do diagrama e do layout do circuito há a necessidade de softwares específicos a exemplo do NI Multisim, Proteus, Eagle, OrCAD, Express PCB e Fritzing.

Neste projeto serão apresentados o diagrama e o Layout do módulo relé utilizado para acionamento de cargas que utilizem tensão de alimentação de 110 V ou 220 V. Este circuito é comumente utilizado em projetos de acionamento de cargas que utilizam o Arduino como plataforma de controle. Para o desenho do circuito foi utilizada a versão gratuita do programa **CadSoft EAGLE PCB Design Software** que pode ser encontrado no site oficial (http://www.cadsoftusa.com/download-eagle/). A figura 27 mostra o resultado final frente e verso após impressão e montagem da PCB.

Figura 27 - Placa PCB Montada do Circuito de Acoplamento.
Fonte: Resultado da Pesquisa (2015).

Uma observação importante é que enquanto os pinos digitais estiverem sendo utilizados pelo Arduino não poderão ser usados por nenhum outro periférico. A alimentação do circuito é feita através da tensão de 5 V_{DC} oriunda da própria placa do Arduino. Cruz (2006) esclarece que a maior vantagem dos relés reside no fato do acionamento de um circuito elétrico de potência poder ser executado por meio de outro circuito elétrico, muitas vezes de menor potência, estando ambos isolados eletricamente entre si já, que o acoplamento entre eles é apenas magnético. A figura 28 mostra a montagem completa do circuito de comando.

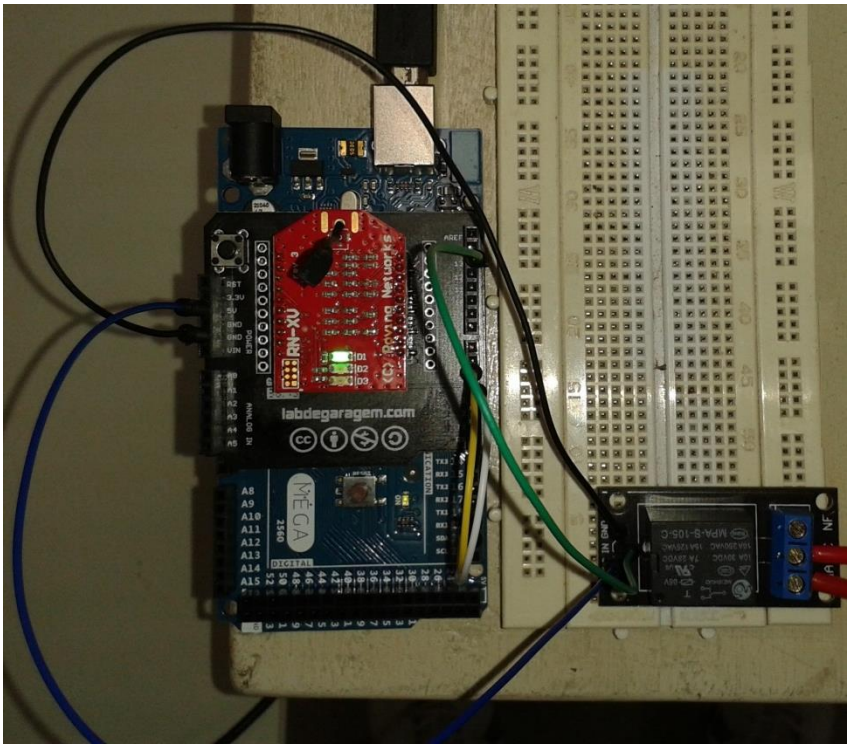


Figura 28 - Montagem do Circuito de Comando.
Fonte: Resultado da Pesquisa (2015).

3.2.5. CIRCUITO DE POTÊNCIA

Acoplado à saída do relé está o circuito de força, cuja função é prover energia ao motor trifásico de indução de 1 CV utilizado nos testes. Tal circuito foi implementado conforme o diagrama esquemático da figura 29, utilizando a rede de alimentação trifásica (220 V) com fechamento em duplo triângulo (ΔΔ). Este circuito foi desenhado e sua simulação foi executada utilizando o software eletrotécnico CADeSIMU.

O aplicativo ora citado tem por função auxiliar na criação de diagramas de comandos elétricos, permitindo realizar a inserção de diversos símbolos, visando desenhar esquemas de acionamentos elétricos. Este software permite o desenvolvimento de diagramas de potência e comando possibilitando, posteriormente, realizar a simulação destes. A sua utilização exige conhecimentos prévios dos conceitos básicos da eletricidade e também de comandos elétricos sendo, no entanto, simples e intuitivo o bastante para o usuário interagir com o mesmo e criar diagramas de baixa complexidade para a realização de simulações.

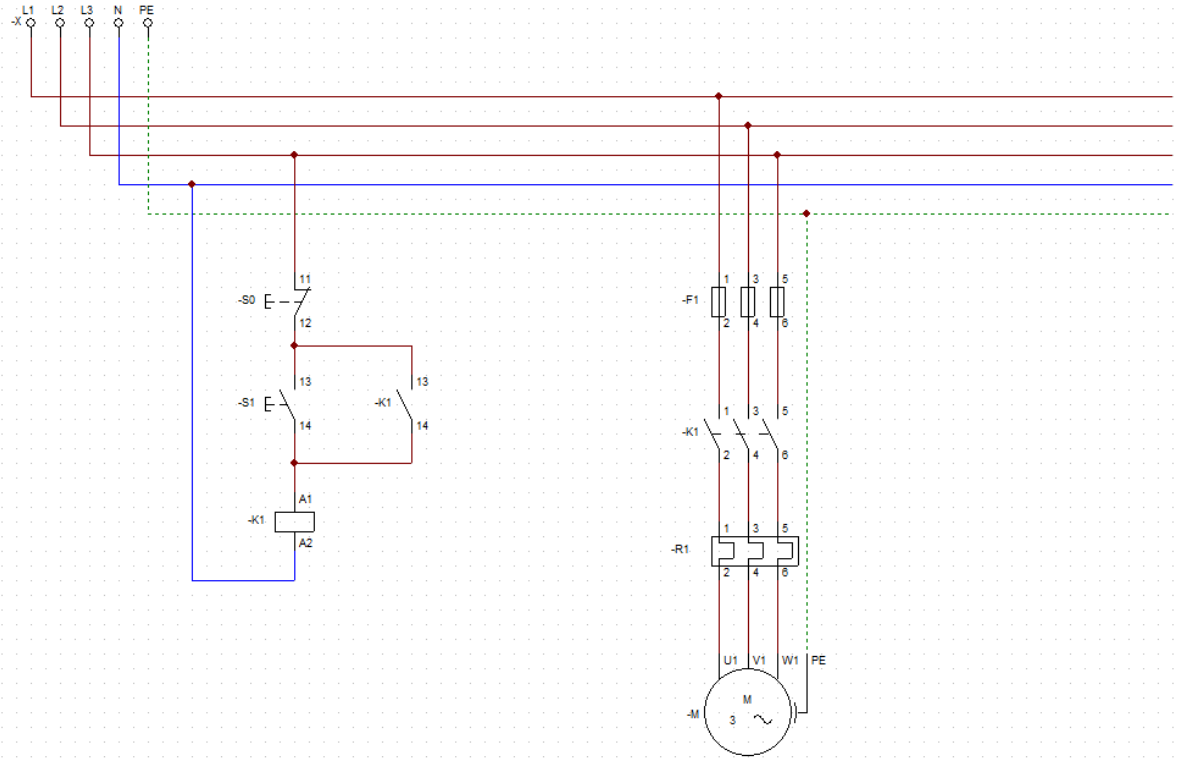


Figura 29 - Diagrama de Acionamento e Comando de Força do Motor.
Fonte: Resultado da Pesquisa (2015).

A figura 29 é o diagrama de força e de comando da partida direta do motor considerado. No lugar das chaves S0 e S1, que guardam as funções respectivas de ligar e desligar a máquina manualmente, foi fechado com contato NA do relé de comando que está conectado ao Arduino e é diretamente acionado pelo mesmo. Outros circuitos eletrônicos utilizando componentes de estado sólido tais como SRCs, DIACs e TRIACs poderiam ser desenvolvidos para executarem a mesma função, porém, não foram utilizados devido ao fato da principal vantagem do relé residir no total isolamento entre o circuito de controle e o de potência.

Toda a etapa de força foi montada no painel de instalações elétricas industriais da Datapool Eletronics, gentilmente cedido para este projeto pelo Instituto Federal de Sergipe (IFS). Neste painel estão montados diversos componentes utilizados em partida, comando e controle de motores elétricos tais como disjuntores, fusíveis, relés temporizadores, contatores, relés térmicos, lâmpadas de sinalização, botoeiras e sensores de temperatura. A figura 30 mostra os componentes utilizados no presente trabalho antes da ligação.

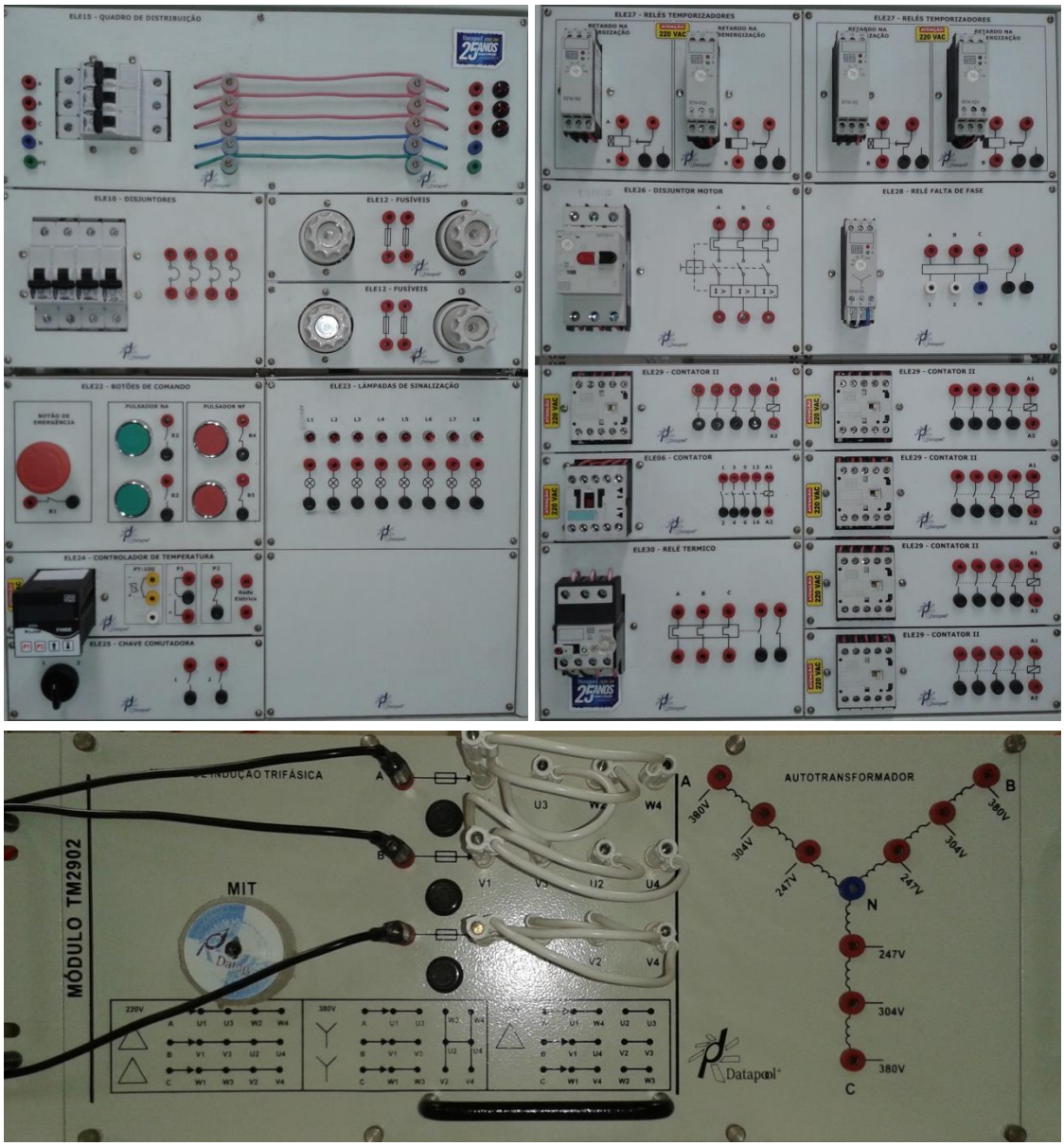


Figura 30 - Dispositivos de Força Utilizados no Acionamento do Motor.
Fonte: Resultado da Pesquisa (2015).

Como o motor utilizado no presente trabalho é de 12 terminais, porém, a tensão da rede trifásica do laboratório onde a montagem do circuito foi executada é de 220 V_{FF} (entre fases), o tipo de ligação do mesmo deve ser duplo triângulo ($\Delta\Delta$). A sequência de montagem da etapa de comando foi executada de tal forma que o Arduino acionasse o relé, a este está conectada a bobina do relé térmico e, aquele, à bobina do contator responsável pelo acionamento direto do MIT.

Em série com a bobina do contator está ligada uma lâmpada sinalizadora cuja função é indicar que o motor está ativado (acesa) ou desligado (apagada). A etapa de potência, responsável pela alimentação elétrica do motor foi montada obedecendo ao sequenciamento exato das fases R, S e T em todos os elementos de força do circuito (disjuntores, fusíveis, relé térmico e contator). A montagem da partida utilizada na inicialização da máquina foi a do tipo “direta” que, apesar das consequências conhecidas em sistemas com cargas de alta inércia, não prejudicou o andamento dos trabalhos nem dos componentes aqui utilizados, uma vez que o motor estava operando em vazio e sem qualquer carga acoplada ao mesmo. A figura 31 mostra o sistema completo após a montagem.

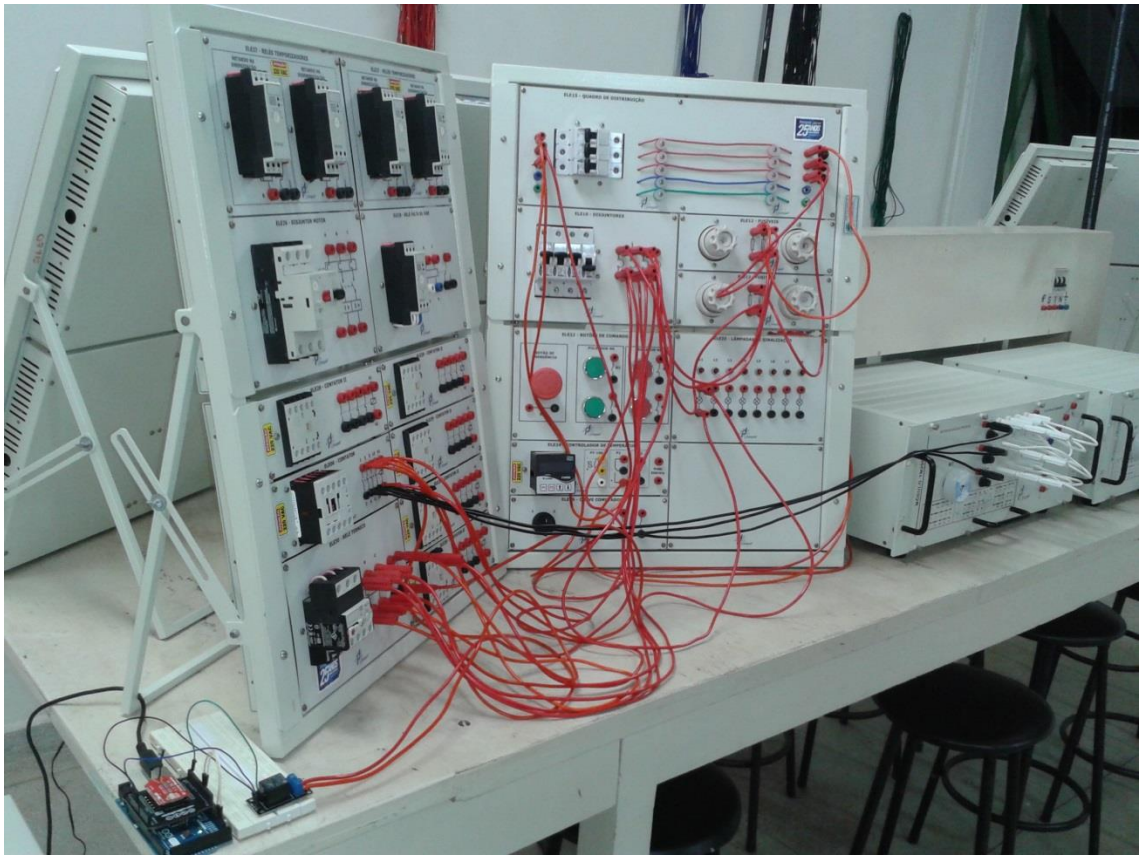


Figura 31 - Circuito de Comando e Força Após Montagem.
Fonte: Resultado da Pesquisa (2015).

3.2.6. CONFIGURAÇÃO DA PLATAFORMA DE DESENVOLVIMENTO

Para o desenvolvimento dos aplicativos de controle, tanto para o dispositivo móvel quanto para o Arduino, fez-se necessária a instalação e configuração dos softwares requeridos, conforme listados em 3.1 e configurados de acordo com o descrito abai-

xo. De acordo com Val (2010), o projeto Ubuntu tem por intenção criar um sistema operacional e uma série completa de aplicações, exclusivamente compostas de Software Livre e de Código Fonte Aberto. O coração da filosofia Ubuntu, sobre a liberdade do software, apoia-se sobre os seguintes ideais (a figura 32 mostra o ambiente de trabalho do sistema operacional considerado):

a) Todas as pessoas deverão ter a liberdade de executar, copiar, distribuir, estudar, partilhar, personalizar, modificar e melhorar o software para qualquer objetivo, sem ter que pagar direitos de licenciamento;
b) Todas as aplicações deverão ser usadas por qualquer pessoa independentemente de sua linguagem materna ou de qualquer deficiência.

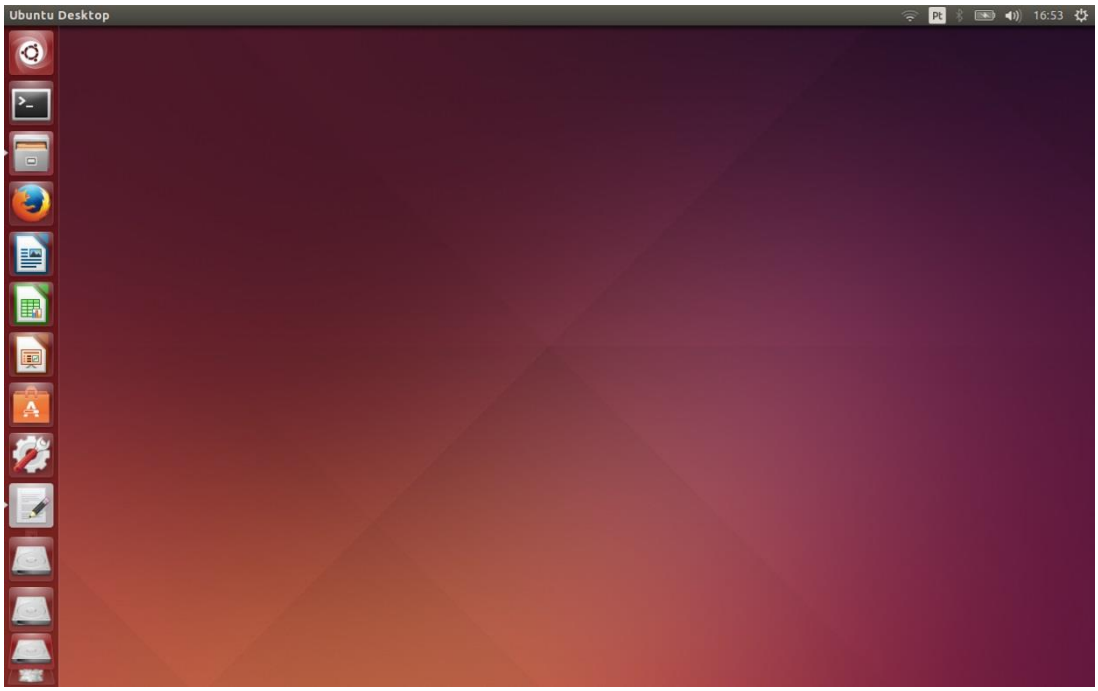


Figura 32 - Área de Trabalho do Ubuntu 14.04.
Fonte: Resultado da Pesquisa (2015).

O Ubuntu Linux foi instalado no notebook Itautec Infoway Note W7655, ao lado de uma instalação pré-existente de outro sistema operacional, necessitando redimensionar o *Hard Disk* (HD) da máquina acima mencionada para liberar espaço em disco suficiente. Todos os comandos podem ser substituídos pelos *front-ends* gráficos da mesma forma que no sistema operacional Windows. O autor optou por trabalhar com a linha de comando devido a sua familiaridade com tais ferramentas de administra-

ção do sistema. O sistema operacional aqui descrito foi instalado com as seguintes configurações:

```
Idioma: Português do Brasil;
Fuso Horário: Brasil (Maceió);
Tipo de Particionamento: Manual (Avançado);
Quantidade de Partições: 3 (Uma para o sistema de arquivo raiz, outra para o diretório Home do usuário e uma terceira para swap);
Tipo de Sistema de Arquivos: ext4;
Tamanho das Partições: 10 GB para o sistema de arquivos raiz, 20 GB para o diretório home do usuário e 1 GB para swap;
Nome de Usuário: gilmar;
Nome do Computador: je01.
```

Para alguns usuários de *desktops*, substituir seus sistemas operacionais atualmente em funcionamento pelo Linux representa um desafio difícil e um processo doloroso, para outros, a troca representa uma evolução prazerosa. Diversos pontos devem ser analisados quando se considerar a hipótese de utilizar Linux no lugar de outros sistemas operacionais, como Microsoft Windows, por exemplo. Na análise destes fatores, a maioria deles aponta diretamente as vantagens de um sistema Linux, dentre os quais, destacam-se neste trabalho: segurança, custo, compatibilidade de programas, melhor aproveitamento do hardware e suporte (JUNIOR, 2007). Após a instalação, configuração básica, atualizações e personalizações, o sistema ficou com a aparência da figura 33.

Figura 33 - Ubuntu 14.04 Personalizado Pelo Usuário.
Fonte: Resultado da Pesquisa (2015).

Após o término do processo de instalação é importante atualizar o sistema, bem como, instalar algumas ferramentas adicionais que ajudam bastante na execução das tarefas de administração do sistema operacional. Isto pode ser feito com a seguinte sequência de comandos:

```
# Atualização do Sistema Operacional
gilmar@je01:~$ sudo apt-get install aptitude –y
gilmar@je01:~$ sudo aptitude update && sudo aptitude upgrade -y

# Instalação do Apache OpenOffice (suíte de escritório)
gilmar@je01:~$ sudo apt-get remove --purge libreoffice*
gilmar@je01:~$ cd Downloads/
gilmar@je01:~/Downloads$ wget
http://ufpr.dl.sourceforge.net/project/openofficeorg.mirror/4.1.1/binaries/pt-
BR/Apache_OpenOffice_4.1.0_Linux_x86_install-deb_pt-BR.tar.gz
gilmar@je01:~/Downloads$ tar -xvzf Apache_OpenOffice_4.1.0_Linux_x86_install-
deb_pt-BR.tar.gz
gilmar@je01:~/Downloads$ cd pt-BR/DEBS/
gilmar@je01:~/Downloads/pt-BR/DEBS$ sudo dpkg -i *.deb
gilmar@je01:~/Downloads$ cd pt-BR/DEBS/desktop-integration/
gilmar@je01:~/Downloads/pt-BR/DEBS/desktop-integration$ sudo dpkg -i *.deb

# Instalação de pacotes diversos
gilmar@je01:~$ sudo aptitude install vim tftp atftpd putty flashplugin-installer
chromium-browser synaptic openbsd-inetd manpages-pt manpages-dev lshw alien build-
essential finger imagemagick fping gettext etherwake libdm0 libfam0 libjs-jquery
gparted lxtask ntfs-config -y

# Compactadores e descompactadores
gilmar@je01:~$ sudo aptitude install unace rar unrar zip unzip p7zip-full p7zip-
rar sharutils uudeview mpack arj cabextract file-roller -y
```

Não há, nos repositórios oficiais do Ubuntu, suporte para o Oracle Java. Em seu lugar é comumente utilizado o OpenJDK. Entretanto, alguns recursos importantes do Android SDK, Eclipse, Arduino IDE e do MIT App Inventor não funcionarão corretamente com ele. Para isso, será usado o repositório do site WebUpd8, que foi criado para tornar o processo de instalação do Oracle Java bem mais fácil para usuários do Ubuntu e seus derivados. Dessa forma, o software pode ser instalado com a seguinte sequência de comandos (sem necessidade de configuração adicional):

```
sudo apt-get remove --purge openjdk*
sudo add-apt-repository ppa:webupd8team/java
sudo aptitude update -y
sudo aptitude install oracle-java8-installer
sudo aptitude install oracle-java8-set-default
```

Para instalar o Kit de Desenvolvimento de Software (SDK) do Android no Ubuntu e poder desenvolver aplicativos para o sistema operacional de *smartphone* da Google é preciso, de antemão, ter instaladas previamente algumas ferramentas necessárias. São elas: Java (no mínimo a versão 6 – nesse caso, foi instalada a versão 8) e o SDK do Android com uma plataforma de desenvolvimento, sendo que nesse trabalho será utilizado o MIT App Inventor em lugar do Eclipse ou da própria IDE do Android SDK.

O download do software foi feito a partir da URL http://developer.android.com/sdk/index.html, selecionado o arquivo referente à plataforma utilizada (Linux x86). Para conseguir baixá-lo, foi necessário aceitar os termos da licença do SDK e, logo a após, o botão *Download* ficou acessível. Seque abaixo os comandos executados para efetuar a descompactação do arquivo SDK e execução do Android, salvo na pasta "Download" do diretório Home do usuário.

```
gilmar@je01:~$ cd Downloads/
gilmar@je01:~/Downloads$ wget http://dl.google.com/android/android-sdk_r23-
linux.tgz
gilmar@je01:~/Downloads$ mkdir -p ~/Programas/Android/Android\ SDK
gilmar@je01:~/Downloads$ mv android-sdk_r23-linux.tgz ~/Programas/Android/Android\
SDK/
gilmar@je01:~/Downloads$ cd ~/Programas/Android/Android\ SDK/
gilmar@je01:~/Programas/Android/Android SDK$ tar -xvzf android-sdk_r23-linux.tgz
gilmar@je01:~/Programas/Android/Android SDK$ rm android-sdk_r23-linux.tgz
```

Os aplicativos do Android SDK requerem suas próprias variáveis de ambiente para poderem funcionar. Quando o aplicativo é invocado, a variável adequada é acessada para indicar onde estão todos os componentes requeridos pelo mesmo. O detalhe é que como o local de instalação do Android SDK fica a critério do usuário, esta variável não é configurada automaticamente, sendo necessária intervenção do mesmo para tal. No Ubuntu, as variáveis de ambiente globais do Android SDK podem ser configuradas para todos os usuários do sistema, sendo necessário editar o arquivo “/etc/bash.bashrc” com permissão de Root e adicionar as seguintes linhas ao final do arquivo:

```
gilmar@je01:~$ sudo vim /etc/bash.bashrc

# Configuração das Variáveis de Ambiente do Android SDK
```

```
export PATH=${PATH}:/home/gilmar/Programas/Android/Android\ SDK/android-sdk-
linux/tools/
export PATH=${PATH}:/home/gilmar/Programas/Android/Android\ SDK/android-sdk-
linux/platform-tools/
export PATH=${PATH}:/home/gilmar/Programas/Eclipse/eclipse/
```

Após isso, é necessário reiniciar o computador para carregar as variáveis corretamente. Como a versão do Android no celular Samsung GT-S6102B é 2.3.6, as ferramentas de desenvolvimento a serem instaladas também devem corresponder a esta versão. A figura 34 mostra as configurações padrões deste componente.

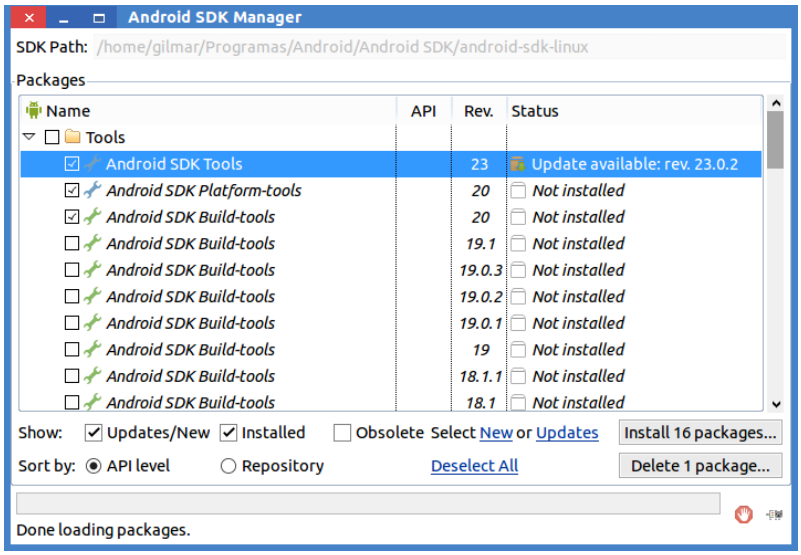


Figura 34 - Tela de Configuração do Android SDK.
Fonte: Resultado da Pesquisa (2015).

Apesar de não ser utilizado o Eclipse para desenvolvimento de software, no presente trabalho ele se faz necessário visto que, além da necessidade de validar o código do Arduino no mesmo, é plenamente possível escrever código para microcontroladores com esta poderosa IDE. Como a versão disponibilizada pelos repositórios oficiais do Ubuntu não contém todos os recursos, foi instalado o software do desenvolvedor oficial e instalado conforme segue abaixo.

```
gilmar@je01:~$ cd ~/Downloads/
gilmar@je01:~/Downloads$ wget
http://espelhos.edugraf.ufsc.br/eclipse//technology/epp/downloads/release/luna/R/e
clipse-jee-luna-R-linux-gtk.tar.gz
gilmar@je01:~/Downloads$ mkdir ~/Programas/Eclipse
gilmar@je01:~/Downloads$ mv eclipse-jee-luna-R-linux-gtk.tar.gz
~/Programas/Eclipse/
gilmar@je01:~/Downloads$ cd ~/Programas/Eclipse/
gilmar@je01:~/Programas/Eclipse$ tar -xvzf eclipse-jee-luna-R-linux-gtk.tar.gz
gilmar@je01:~/Programas/Eclipse$ rm eclipse-jee-luna-R-linux-gtk.tar.gz
```

Quando inicializa pela primeira vez, o Eclipse pede ao usuário para especificar o diretório de trabalho do software. Nesse trabalho, o *workspace* do Eclipse foi atribuído conforme a figura 35.

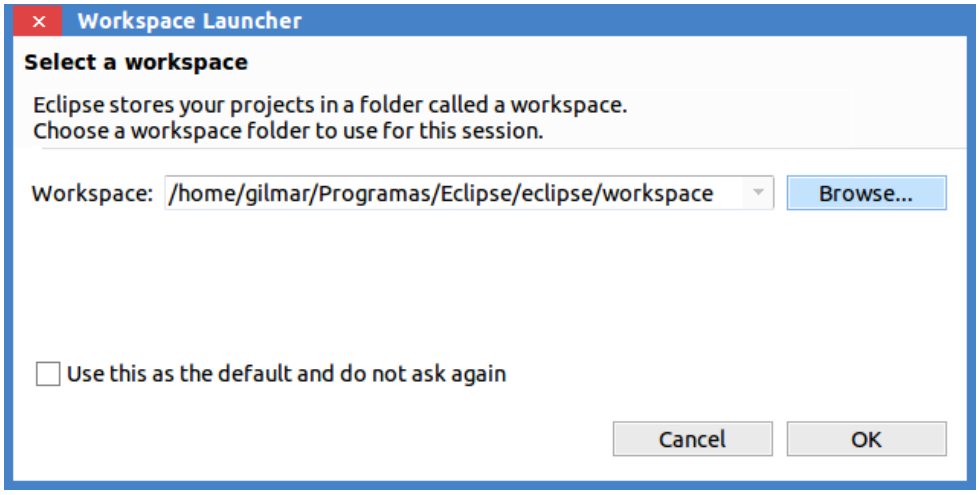


Figura 35 - Configuração do Workspace da IDE do Eclipse.
Fonte: Resultado da Pesquisa (2015).

Após as dependências (Java, Android SDK e Eclipse) estarem devidamente instaladas, configuradas e funcionando corretamente, foi instalado o ADT (*Android Developer Tools*). Para isso, no Eclipse, navegou-se até o *menu* “Help > Install New software. Clicou-se no botão “Add” para informar o item desejado. Na tela aberta, digitou-se o nome que identificou a tarefa “Android ADT”, seguido do seguinte link, que é o local padrão de instalação do ADT mais atual lançado pelo Google, para o Eclipse: “https://dl-ssl.google.com/android/eclipse/”.

Nesse trabalho será utilizada a versão 2.0 do MIT App Inventor. Ela é totalmente *online*, requerendo apenas que a plataforma Java esteja instalada e configurada corretamente. Entretanto, pode haver necessidade de uso da versão 1.0 (como a portabilidade de código). No Ubuntu, os procedimentos de instalação da versão 1.0 foram executados como descritos abaixo.

```
gilmar@je01:~$ cd ~/Downloads/
gilmar@je01:~/Downloads$ wget
http://dl.google.com/dl/appinventor/installers/linux/appinventor-setup_1.1_all.deb
gilmar@je01:~/Downloads$ sudo dpkg -i appinventor-setup_1.1_all.deb
```

Para o sistema operacional Ubuntu Linux há um pacote oficial do Arduino IDE nos repositórios oficiais da distribuição. Dessa forma, não há necessidade de procedimentos específicos para a sua instalação que pode ser feita mediante execução dos comandos abaixo. A figura 36 mostra a interface aberta imediatamente após a instalação e configuração da IDE disponível nos repositórios oficiais da Canonical.

```
gilmar@je01:~$ sudo aptitude install -y arduino arduino-core arduino-mk arduino-
mighty-1284p flashrom
```

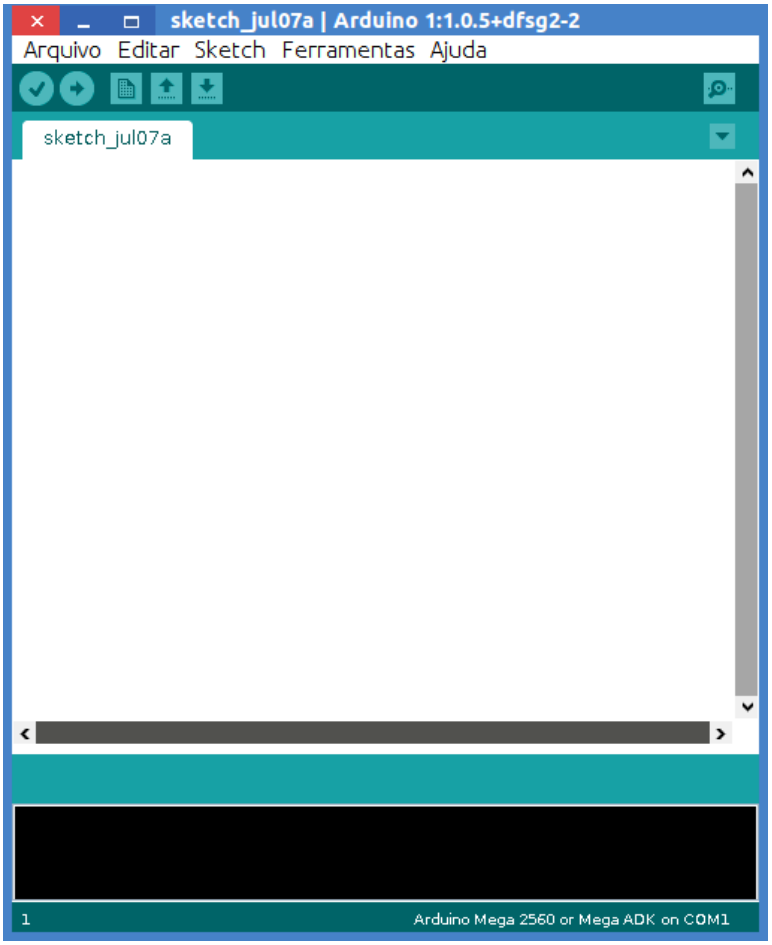


Figura 36 - Interface de Desenvolvimento Arduino Após a Instalação.
Fonte: Resultado da Pesquisa (2015).

Entretanto, a instalação pode ser feita também de forma manual (utilizada neste trabalho porque a versão da IDE disponibilizada pelos repositórios da Canonical na data em que esta monografia estava sendo escrita – 1.0.5 – mostrou-se incompatível com a biblioteca WiFlyHQ). Então, foi feito teste com a versão oficial 1.0.6, resultando no funcionamento correto com as *libraries* aqui utilizadas. Para instalar essa versão, bastou executar a seguinte sequência de comandos:

```
gilmar@je01:~$ sudo aptitude install avr-libc avrdude binutils-avr extra-xdg-menus
gcc-avr  libftdi1 libjna-java librxtx-java
gilmar@je01:~$ cd Downloads/
gilmar@je01:~/Downloads$ wget http://arduino.cc/download.php?f=/arduino-1.0.6-
linux32.tgz
gilmar@je01:~$ mv Downloads/arduino-1.0.6-linux32.tgz Programas/Arduino/
gilmar@je01:~$ cd Programas/Arduino/
gilmar@je01:~/Programas/Arduino$ tar -xvzf arduino-1.0.6-linux32.tgz
gilmar@je01:~/Programas/Arduino$ cd arduino-1.0.6
gilmar@je01:~/Programas/Arduino/arduino-1.0.6$ chmod +x arduino
```

Antes da utilização, é necessário configurar a IDE para utilizar a placa adequada e a porta serial da seguinte maneira:

```
Ferramentas > Placa > Arduino Mega 2560 ou Mega ADK
Ferramentas > Porta Serial > /dev/ttyACM0
```

Além disso, no momento da instalação, o software cria o grupo de sistema “dialout”. Para o correto funcionamento do mesmo, é necessário incluir a conta do usuário que o utilizará nesses dois grupos de sistema com permissão para escrever dados nas portas seriais (tty). Caso a conta de usuário não esteja nesses grupos a IDE não conseguirá fazer upload do código para a placa do Arduino.

```
gilmar@je01:~$ sudo usermod -a -G tty gilmar
gilmar@je01:~$ sudo usermod -a -G dialout gilmar
```

3.2.7. DESENVOLVIMENTO DO SOFTWARE PARA O DISPOSITIVO MÓVEL

O aparelho móvel utilizado para executar o aplicativo dedicado foi o Samsung Galaxy Y Duos S6102B com sistema operacional Android nativo de fábrica, sendo empregado na comunicação com o microcontrolador. Esse interfaceamento foi efetivado por meio de um módulo Wi-Fi Shield acoplado ao próprio Arduino, cuja finalidade é de receber os comandos enviados pelo dispositivo móvel. Esse componente eletrônico, conforme item 6.1.3, é um módulo transmissor e receptor, trabalha com velocidades, no padrão 802.11b (1 – 11 Mbps) e no 802.11g, (6 – 54 Mbps), cobertura de sinal de até 100 m, antena embutida e alimentação de 3,3 Vcc. O fluxograma de comunicação entre os sistemas é mostrado na figura 37.

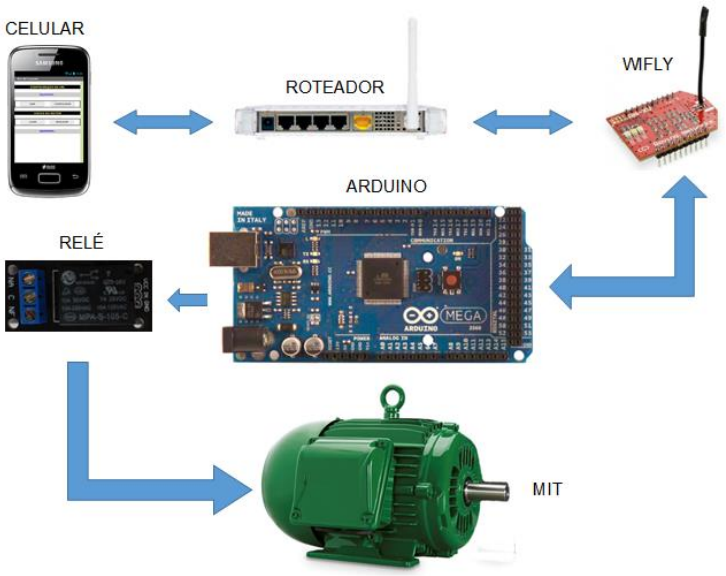


Figura 37 - Fluxograma de Comunicação Entre os Dispositivos.
Fonte: Resultado da Pesquisa (2015).

Para que tal intercâmbio fosse possível, foi necessário criar um programa responsável pelo envio de sinais a partir do dispositivo móvel. Após o botão do programa ser pressionado pelo operador portador do aparelho móvel com o software nele instalado, o sinal enviado pelo celular é transmitido ao Wi-Fi Shield e chega ao microcontrolador. A partir deste, é enviado ao relé Shield, que é responsável por energizar/desenergizar a bobina do contator do circuito de força, ligando/desligando o motor.

O aplicativo para o aparelho celular é composto de duas telas. Na primeira o usuário precisa se autenticar com uma senha padrão gerada no momento da produção do aplicativo, dessa forma, o mesmo somente terá acesso às funções internas caso saiba a senha do mesmo. Na segunda tela estão contidas todas as funcionalidades do software. Todas as funções, tanto da tela de autenticação quanto da principal, foram implementadas de acordo com o fluxograma demonstrado na figura 38, abaixo.

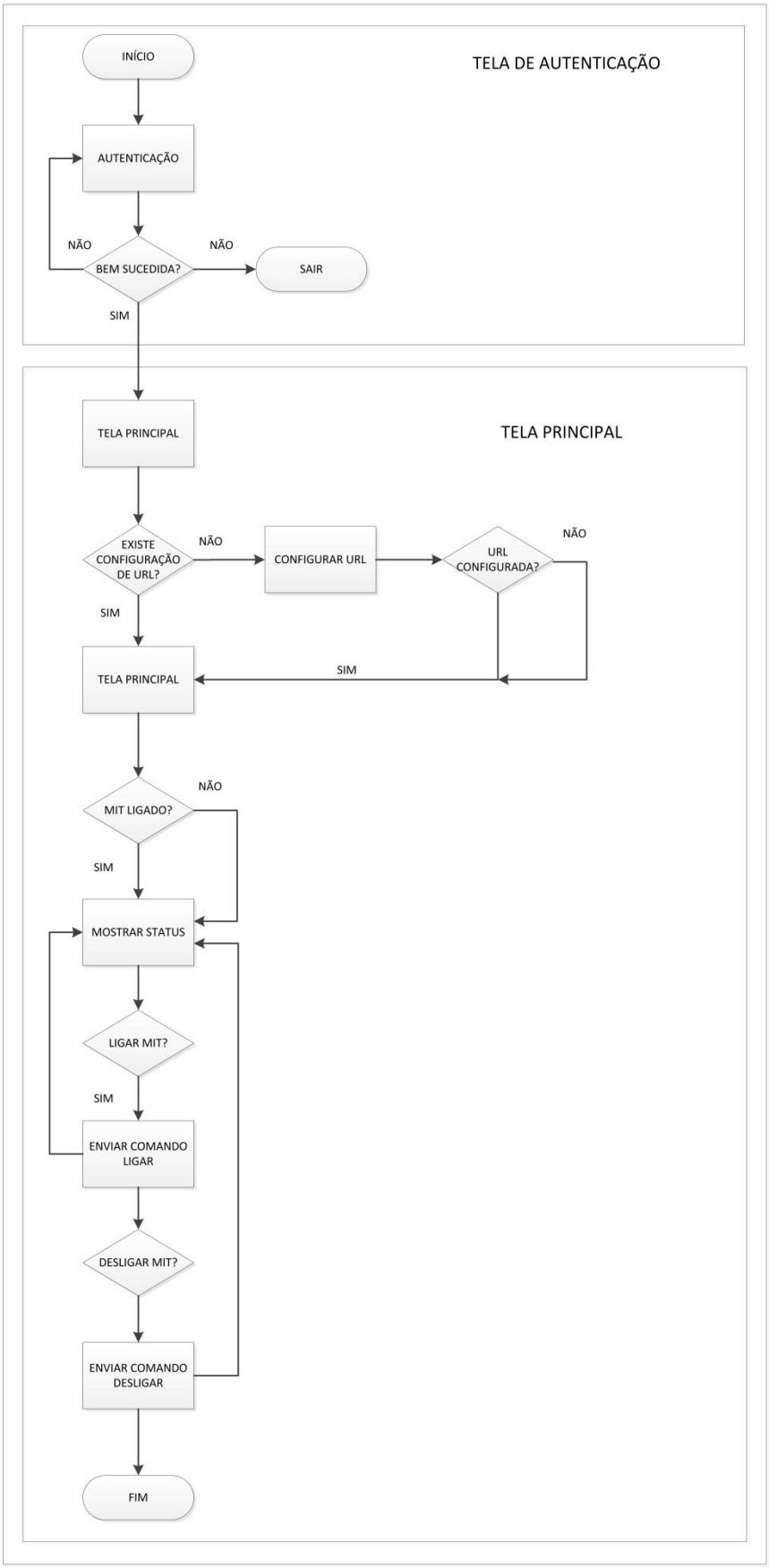


Figura 38 - Fluxograma do Aplicativo App Inventor.
Fonte: Dados da Pesquisa (2015).

A estrutura de programação foi construída através do sistema de fluxograma denominado App Inventor. Conforme Wolber, et al. (2011), a interface desse aplicativo é disposta de maneira similar às IDEs mais tradicionais. Contudo, em vez de se visualizar a árvore de arquivos ao lado esquerdo da tela, os elementos são organizados de maneira a dar ao usuário a liberdade de montá-los conforme a sua necessidade. A figura 39 contém a visualização da tela de autenticação do software (adotada por medida adicional de segurança), bem como os componentes do App Inventor utilizados na sua construção.

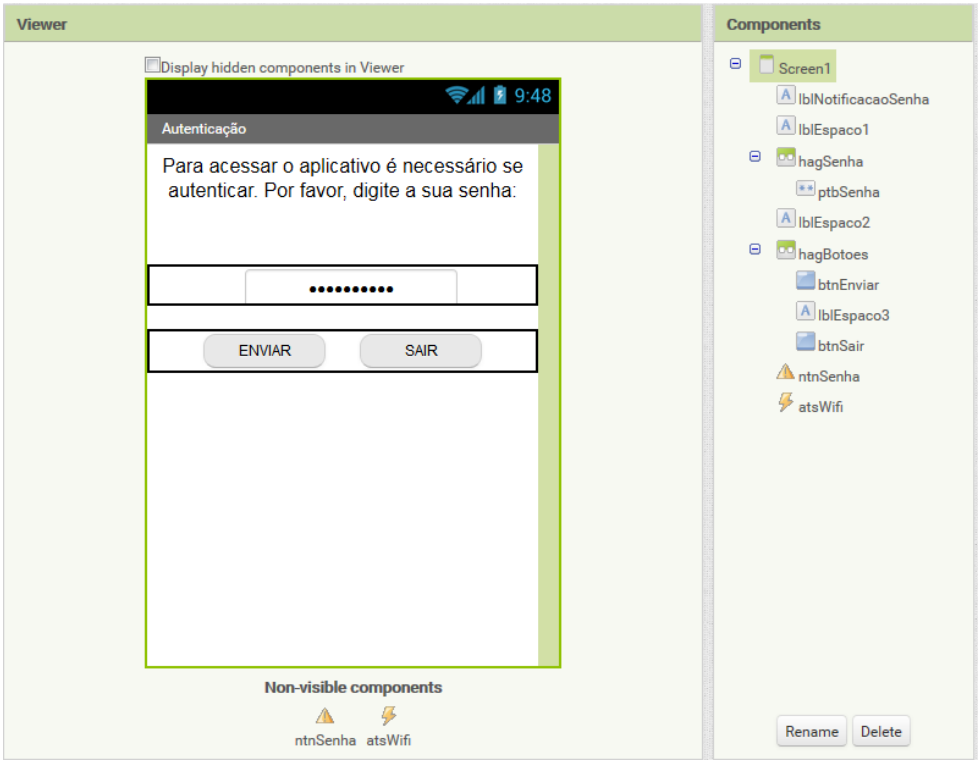


Figura 39 - Design da Tela de Login do Aplicativo App Inventor.
Fonte: Dados da Pesquisa (2015).

A figura 40 mostra o código da tela de autenticação correspondente à imagem acima. Nela é possível visualizar a variável <Pwd> criada para armazenar a senha do programa, pedida no início da sua inicialização. Se o usuário digitar a senha incorreta o programa irá adverti-lo disso, além de não disponibilizar as demais funções ao mesmo; caso contrário, irá abrir a tela principal e disponibilizará todas as funções nele disponíveis.

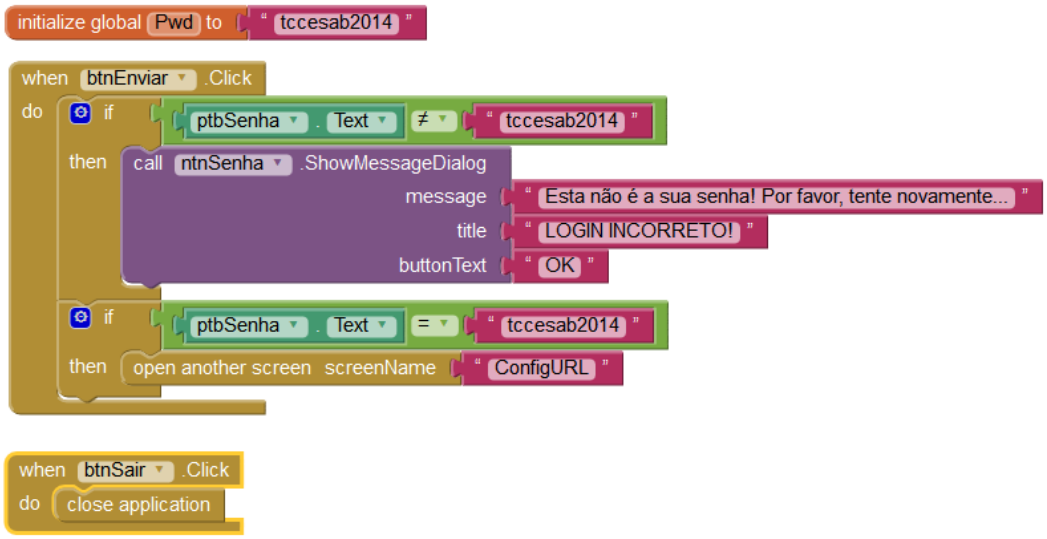


Figura 40 - Code Blocks da Tela de Autenticação.
Fonte: Dados da Pesquisa (2015).

A tela principal é composta do botão de configuração da URL (endereço do módulo WiFly), do botão que disponibiliza a função de sair do aplicativo e dos botões que ligam e desligam o motor. A figura 41 mostra o *design* da mesma e o arranjo dos elementos em seu espaço.

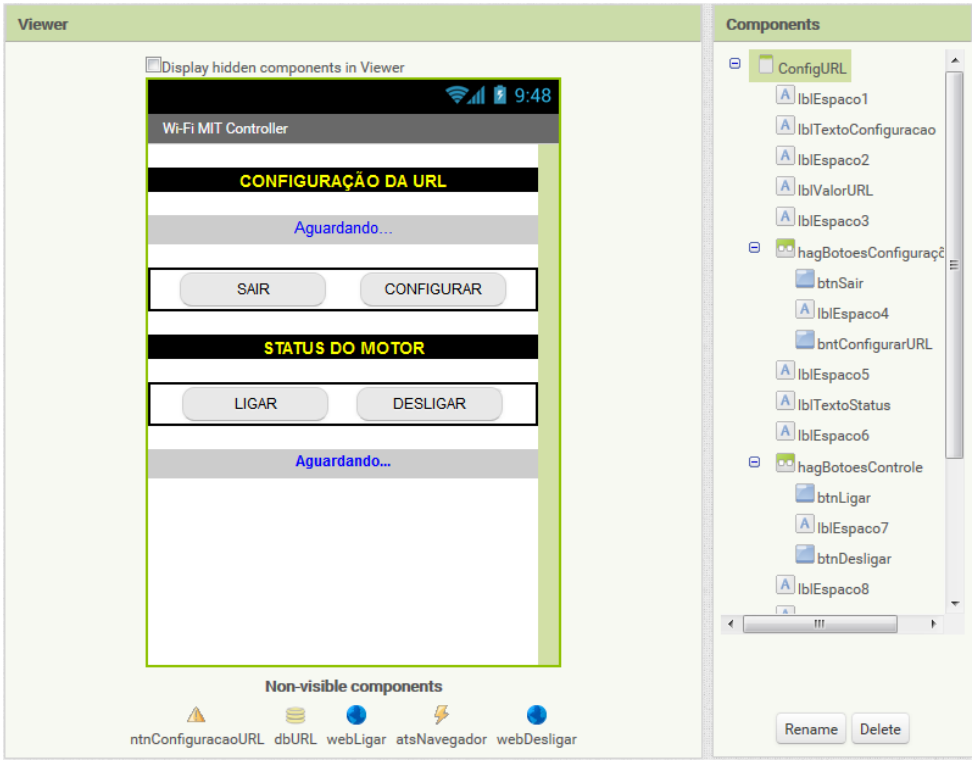


Figura 41 - Design da Tela Principal.
Fonte: Dados da Pesquisa (2015).

A figura 42 mostra o código da parte de configuração. Tal código é desenhado de tal forma que haja dois botões e uma *label* onde as informações referentes ao estado da mesma são visualizadas. A URI do arduino é gravada de forma permanente no aparelho por meio do componente TinyDB e recuperada no momento em que os botões ligar/desligar são acionados.

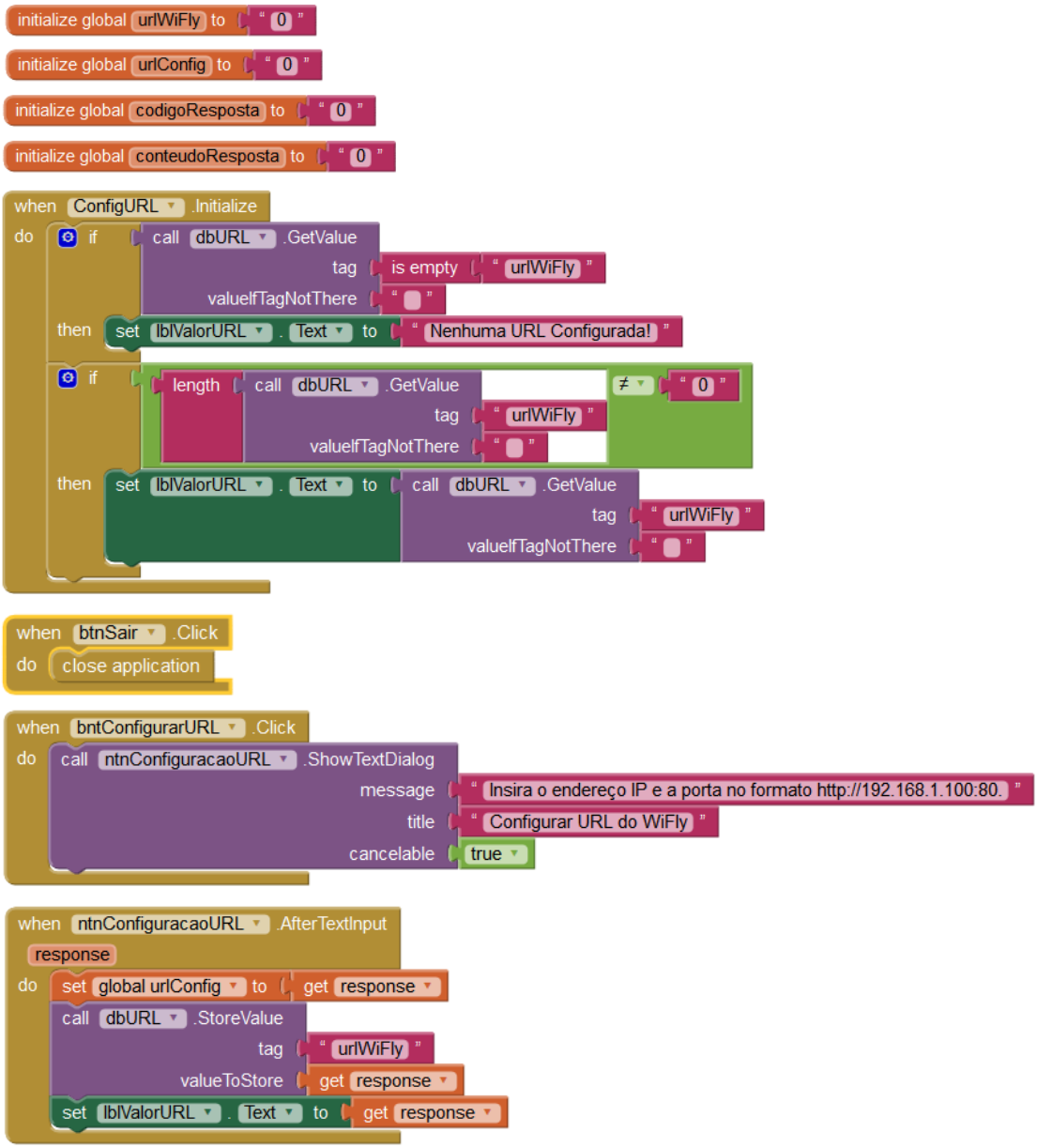


Figura 42 - Code Blocks da Configuração de URL.
Fonte: Dados da Pesquisa (2015).

A figura 43 mostra o código inerente à parte de ativação e desativação do motor, além da comunicação entre o dispositivo móvel e o roteador Wi-Fi.

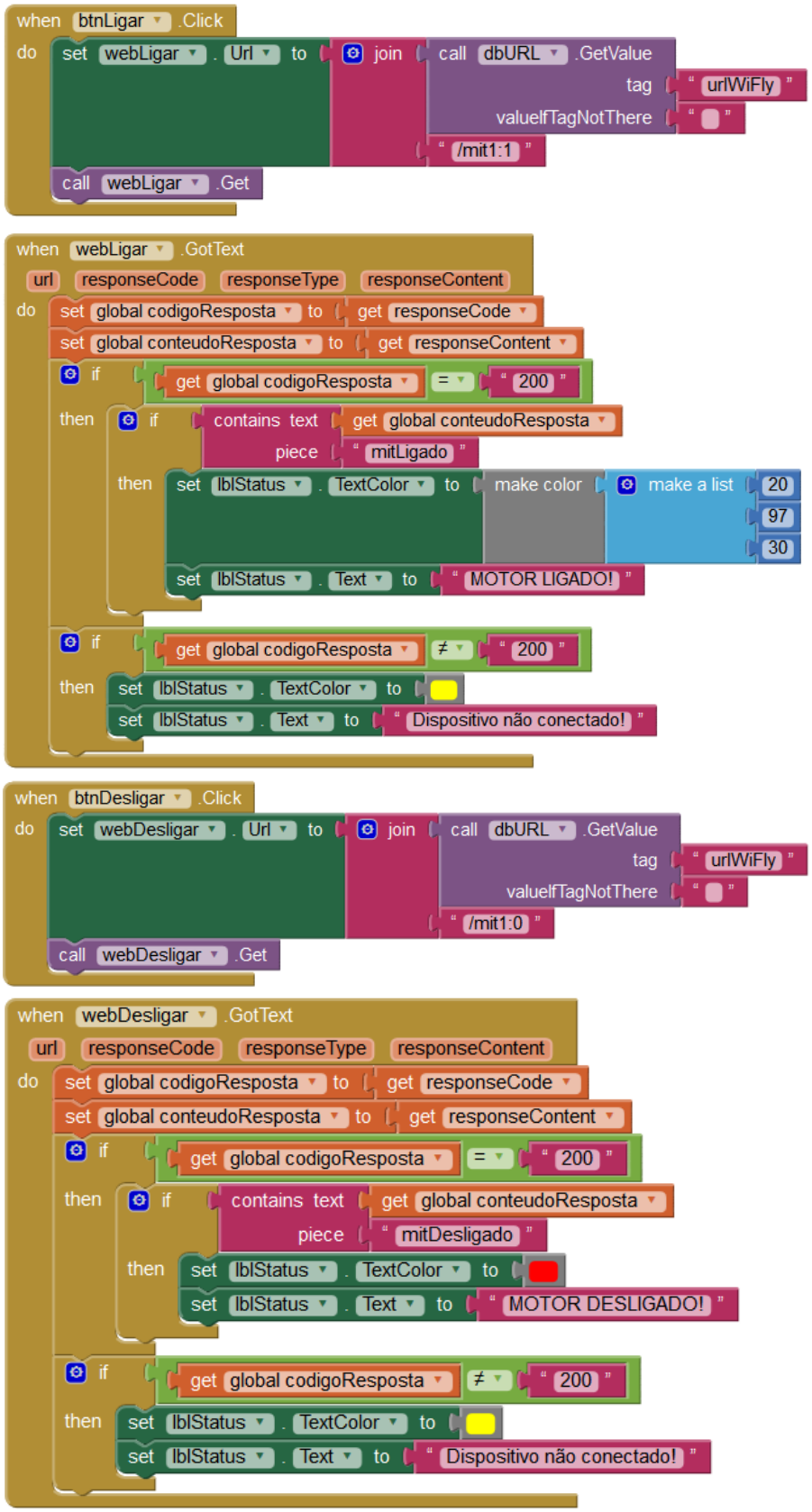


Figura 43 - Code Blocks do Acionamento do Motor.
Fonte: Dados da Pesquisa (2015).

3.2.8. DESENVOLVIMENTO DO SKETCHE CONTROLADOR DO ARDUINO

Segundo McRoberts (2011), os programas para Arduino são chamados *sketches*. Um *sketch* tem obrigatoriamente duas funções em seu corpo: <setup ()> e <loop()>. A primeira função é executada apenas uma vez, ao inicializar o programa, sendo responsável por conter as instruções gerais que preparam o mesmo para o *loop* principal ser executado. A segunda função é fundamental e fica sempre em execução enquanto o Arduino estiver ligado, indefinidamente. As variáveis de entrada foram definidas de acordo com a sintaxe própria da linguagem do Arduino, fixando as configurações do mesmo. Após isso, abriu-se o bloco de configuração no qual foi feita a definição dos pinos de entrada e saída, além da inserção das funções <Serial.begin()> e <wifly.begin()>, responsáveis por ajustarem a taxa da transmissão serial do circuito, devendo ser as primeiras a serem chamadas. Após configurar essa função e definir todos os pinos, o bloco de código deve ser fechado.

No grupo de repetição estão os comandos dos motores, no entanto, antes disso, é necessário habilitar a comunicação do Wi-Fi Shield. O estado de tal condição, a princípio, é nulo. Por conseguinte, foi gerada a concatenação dos caracteres que direcionam o comando para o motor alvo. Junto à condição anterior, há as funções <Serial.available()>, <Serial.read()>, <wifly.available()> e <wifly.read()>. Elas são de fundamental importância em conjunto com as outras, pois será por meio delas que o envio/recebimento de caracteres de/para o Arduino é verificado, ou seja, elas estão na escuta aguardando sinais.

As sequências associam as informações recebidas pelo microcontrolador, decodificando-as de acordo com a programação criada. Caso seja necessário ligar o motor por meio de um dispositivo móvel, a informação que deve ser recebida no microcontrolador é “mit1:1”. A primeira parte designa o equipamento a ser ligado. Como futuramente esse mesmo código será reaproveitado em novas aplicações, ficou convencionado que “mit1” é uma *string* para ligar o motor 1. A segunda parte é o comando de ligar (1) ou desligar (0). A figura 44 mostra o fluxograma do sketche controlador do Arduino.

```
mit1:1 -> Ligar o motor 1
mit1:0 -> Desligar motor 1
```

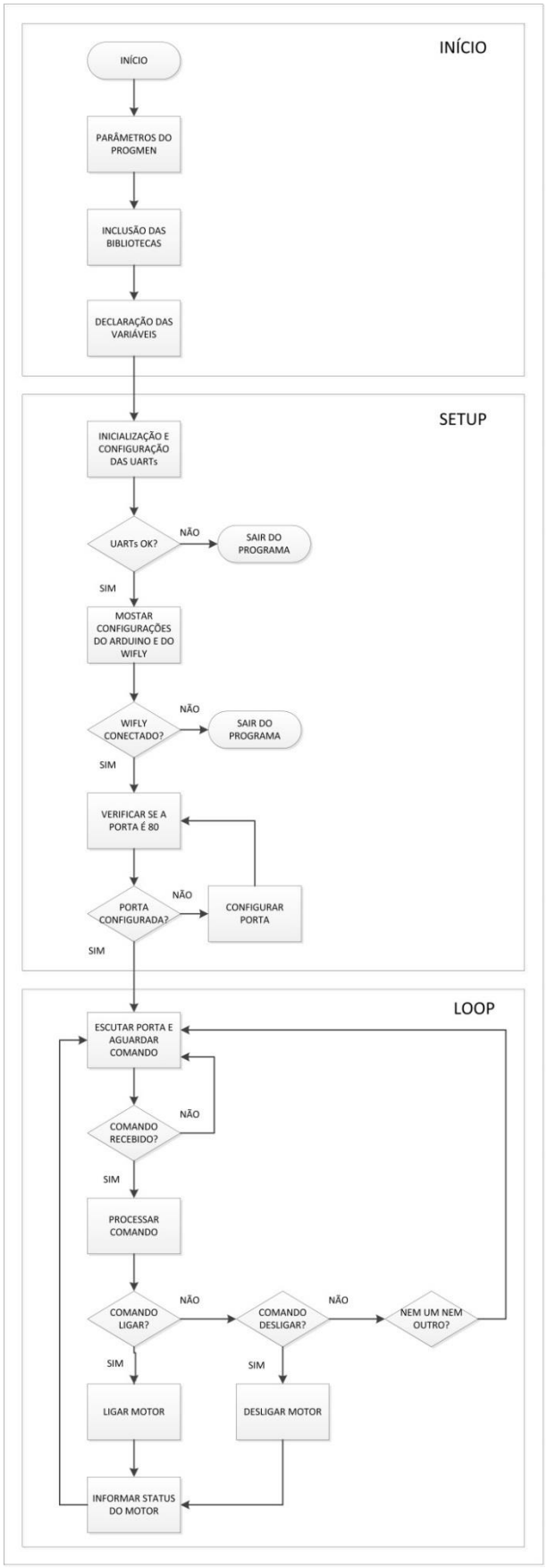


Figura 44 - Fluxograma do Código do Sketche do Arduino.
Fonte: Dados da Pesquisa (2015).

Quando o aplicativo instalado no celular faz uma solicitação ao Arduino para “ligar” ou “desligar” o motor, ele a envia na forma de um cabeçalho HTTP. Para tratar esses dados adquiridos pela porta serial, faz-se necessário recorrer ao recurso fornecido pelas *Strings*, que são matrizes de 1 linha por várias colunas (ou vários caracteres). No Arduino, é possível declarar uma variável como *String* e não precisar ler caractere a caractere como em muitas linguagens de programação, ficando assim, mais fácil de trabalhar e programar. Por exemplo, quando o botão “ligar” é ativado, o celular envia o seguinte cabeçalho HTTP para o Arduino:

```
*OPEN*GET /mit1:1 HTTP/1.1
User-Agent: Dalvik/1.6.0 (Linux; U; Android 2.3.6; GT-S6102B Build/IMM76I)
Host: 192.168.1.100
Connection: Keep-Alive
Accept-Encoding: gzip
```

Ao utilizar a função <wifly.read()> os dados são recebidos, um de cada vez, (isto é, caractere a caractere). Para ficar mais fácil de tratá-los, os mesmos são guardados em uma matriz *array* e, depois, colocados em uma variável *String*. Agora, basta comparar os valores padronizados com os destacados na cor azul. Segue abaixo o código do *sketche* controlador do Arduino desenvolvido para o presente trabalho. Uma observação importante é que o mesmo suporta apenas uma conexão ativa por cliente, desta forma, não é possível instalar o aplicativo desenvolvido no App Inventor em mais de um aparelho celular para controlar o motor de forma simultânea.

```
/*
 * Sketche para receber dados via módulo WiFly RN-XV e ativar pinos digitais
 * Escrito por Gilmar Silva
 * Em 07 de dezembro de 2014
 */

// Parâmetros do PROGMEN
#undef PROGMEM
#define PROGMEM __attribute__(( section(".progmem.data") ))
#undef PSTR
#define PSTR(s) (__extension__({static prog_char __c[] PROGMEM = (s); &__c[0];}))

// Inclusão das bibliotecas e dependências
#include <Arduino.h>
#include <SPI.h>
#include <WiFlyHQ.h>
#include <String.h>
#include <PString.h>


// Variáveis do programa
```

```
WiFly wifly;
char buf[30];
char cmd[1000];
void mit1_1();
void mit1_0();

// Configuração do hardware
void setup()
{

  // Comportamento do pino 13
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);

  // Comportamento da UART 0
  Serial.begin(9600);
  Serial.println(F("Inicializando"));
  Serial.print(F("Memória livre: "));
  Serial.println(wifly.getFreeMemory(),DEC);

  // Comportamento da UART 1
  Serial1.begin(9600);
  wifly.begin(&Serial1, NULL);
  if (!wifly.begin(&Serial1, &Serial)) {
    Serial.println(F("Comunicação serial mau sucedida!"));
    wifly.terminal();
  }

  // Verifica se o módulo está associado à alguma rede
  if (wifly.isAssociated())
  {

  // Mostra informações da conexão
  Serial.print(F("Dispositivo associado a rede: "));
  Serial.println(wifly.getSSID(buf, sizeof(buf)));
  Serial.print(F("MAC: "));
  Serial.println(wifly.getMAC(buf, sizeof(buf)));
  Serial.print(F("IP: "));
  Serial.print(wifly.getIP(buf, sizeof(buf)));
  Serial.print(F(" Port: "));
  Serial.println(wifly.getPort());
  Serial.print(F("Mascara de Rede: "));
  Serial.println(wifly.getNetmask(buf, sizeof(buf)));
  Serial.print(F("Gateway: "));
  Serial.println(wifly.getGateway(buf, sizeof(buf)));
  Serial.print(F("DNS: "));
  Serial.println(wifly.getDNS(buf, sizeof(buf)));
  } else {
      Serial.println(F("O dispositivo não está associado a nenhuma rede sem-
fio!"));
      wifly.terminal();
         }

  // Atribui um nome ao dispositivo
  wifly.setDeviceID("WiFi-MIT-Controller");

  // Desliga UPD broadcast
  wifly.setBroadcastInterval(0);
```

```
  // Verifica se o módulo está conectado
  if (wifly.isConnected())
  {
    Serial.println(F("O modulo WiFly está conectado!"));
  } else {
      Serial.println(F("O dispositivo não conseguiu estabelecer uma conexão!"));
      wifly.terminal();
    }

  // Verifica a configuração da porta padrão
  if (wifly.getPort() != 80)
  {
    Serial.println(F("Fechando outras conexões ativas"));
    wifly.close();
    wifly.setPort(80);
    wifly.save();
    Serial.print(F("A porta padrão foi modificada para: "));
    Serial.println(wifly.getPort());

    // Habilitar o protocolo TCP/IP
    wifly.setProtocol(WIFLY_PROTOCOL_TCP);

    // Reiniciar o módulo para alterar o número da porta
    Serial.println(F("Reiniciando..."));
    wifly.reboot();
    delay(3000);
  }
  Serial.println(F("Pronto"));
}

// Loop principal
void loop() {

  // Verifica se há dados provenientes do WiFly
  if (wifly.available() > 0) {

    // Caso haja dados, armazena-os na string “cmd”
    if (wifly.gets(cmd, sizeof(cmd))) {

      // Verifica se é um GET na página "/mit1_1"
      if (strncmp_P(cmd, PSTR("GET /mit1:1"), 12) > 0) {
            mit1_1();         // Pula para a rotina que liga o motor
            wifly.flushRx();  // Descarta o restante da transmissão
          } else {

            // Verifica se é um GET na página "mit1_0"
            if (strncmp_P(cmd, PSTR("GET /mit1:0"), 12) > 0) {
                 mit1_0();         //Vai para rotina que desliga o motor
                 wifly.flushRx();  // Descarta o restante da transmissão
            }
         }
      }
    }
  }



// Rotina responsável por ativar o motor
```

```
void mit1_1() {

// Ativa o pino 13
digitalWrite(13,HIGH); // Liga o MIT

  // Envia Cabeçalho HTML
  wifly.println(F("HTTP/1.1 200 OK"));
  wifly.println(F("Content-Type: text/html"));
  wifly.println(F("Transfer-Encoding: chunked"));
  wifly.println();
  wifly.sendChunkln(F("<html>"));
  wifly.sendChunkln(F("mitLigado"));
  wifly.sendChunkln(F("</html>"));
  wifly.sendChunkln();

  // Fecha a conexão
  wifly.close();
}

// Rotina responsável por desativar o motor
void mit1_0() {

// Desativa o pino 13
digitalWrite(13,LOW); // Desliga o MIT

  // Envia Cabeçalho HTML
  wifly.println(F("HTTP/1.1 200 OK"));
  wifly.println(F("Content-Type: text/html"));
  wifly.println(F("Transfer-Encoding: chunked"));
  wifly.println();
  wifly.sendChunkln(F("<html>"));
  wifly.sendChunkln(F("mitDesligado"));
  wifly.sendChunkln(F("</html>"));
  wifly.sendChunkln();

  // Fecha a conexão
  wifly.close();
}
```

4. RESULTADOS E DISCUSSÃO

Todo o sistema foi construído de forma modular e progressiva. A parte mais complexa de ser implementada se constituiu na geração do software controlador do Arduino (visto que é o cerne do sistema de acionamento aqui proposto, pois dele depende o funcionamento do processo). Foi preparado para ligar e desligar apenas um motor, independentemente do tipo e da potência do mesmo, já que a forma de acionamento da etapa de potência é totalmente autônoma da plataforma de comando. Isso permite que esse projeto tenha diversas aplicações, como por exemplo: acionar motores individualmente, acionar as máquinas localmente ou à distância utilizando o aparelho celular com possibilidade até mesmo de comando via internet.

Para a aplicação descrita nas seções anteriores foi construída uma página HTML, desenvolvida diretamente na interface de programação do Arduino. Isso implicou no estudo das trocas de mensagens via protocolo HTTP, uma vez que o Arduino foi posto a funcionar como um servidor Web, respondendo às requisições, recebendo e tratando os comandos oriundos do aparelho celular. O *sketche* implementado neste trabalho tem embutido em seu código tags HTML responsáveis por enviar os cabeçalhos de estado e tratamento das conexões.

Tentou-se implementar algumas funções condicionais em meio ao código HTML, mas este recurso não funcionou corretamente visto que era necessário alterar o tempo de espera (delay) no código do App Inventor, o que não foi possível neste momento. Tais elementos serviram para demonstrar que encadeamentos de código contendo muitas variáveis recursivas e loops que quebram a sequência de execução do firmware, consomem mais memória e podem comprometer o funcionamento do programa.

Outro fator que também mereceu atenção foi a velocidade de transmissão das UARTs. Inicialmente elas foram configuradas para trabalhar com baund rate de 115200, entretanto, notou-se que os dados chegavam corrompidos ao Arduino devido à interferência do roteador nos jumpers, quando o módulo se encontrava próximo ao mesmo. Tais problemas foram resolvidos por meio da reconfiguração da velocidade

do módulo para o padrão de fábrica (9600 baunds) e pondo o mesmo para trabalhar de forma sincronizada com o Arduino, ou seja, todos à mesma taxa de transmissão.

Apesar do HTML ser uma linguagem poderosa e que permite expandir as capacidades do Arduino, tem limitações no que diz respeito à quantidade de memória e de processamento do mesmo, ficando este projeto restrito ao hardware do equipamento e não podendo estender tanto as suas capacidades. Tais restrições podem ser superadas em um projeto futuro, com o uso de um servidor Web externo ao Arduino, cujas funções poderiam ser muito bem executadas pelo Apache Web Server configurado com suporte ao PHP (*Hypertext Preprocessor*) e ao banco de dados MySQL.

Ao iniciar pela primeira vez, o *sketche* envia algumas informações importantes para o *Serial Monitor*, tanto em relação ao estado do módulo quanto da conexão com o roteador. Além disso, o mesmo imprime na porta serial principal dados inerentes ao endereçamento tais como a quantidade de memória livre, rede Wi-Fi à qual o mesmo se encontra associado, endereço MAC, endereço IP e porta, máscara de rede, gateway e servidor DNS. Tais informações estão na seção “setup” e são mostradas ao usuário apenas uma vez, durante a primeira execução do *firmware*. A figura 45 mostra as informações enviadas durante a inicialização do Arduino ao *Serial Monitor*.

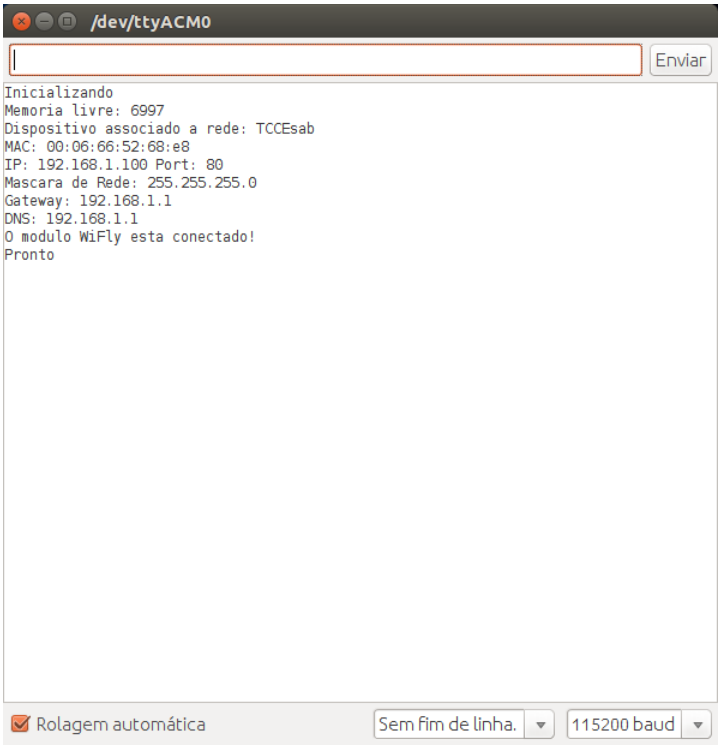
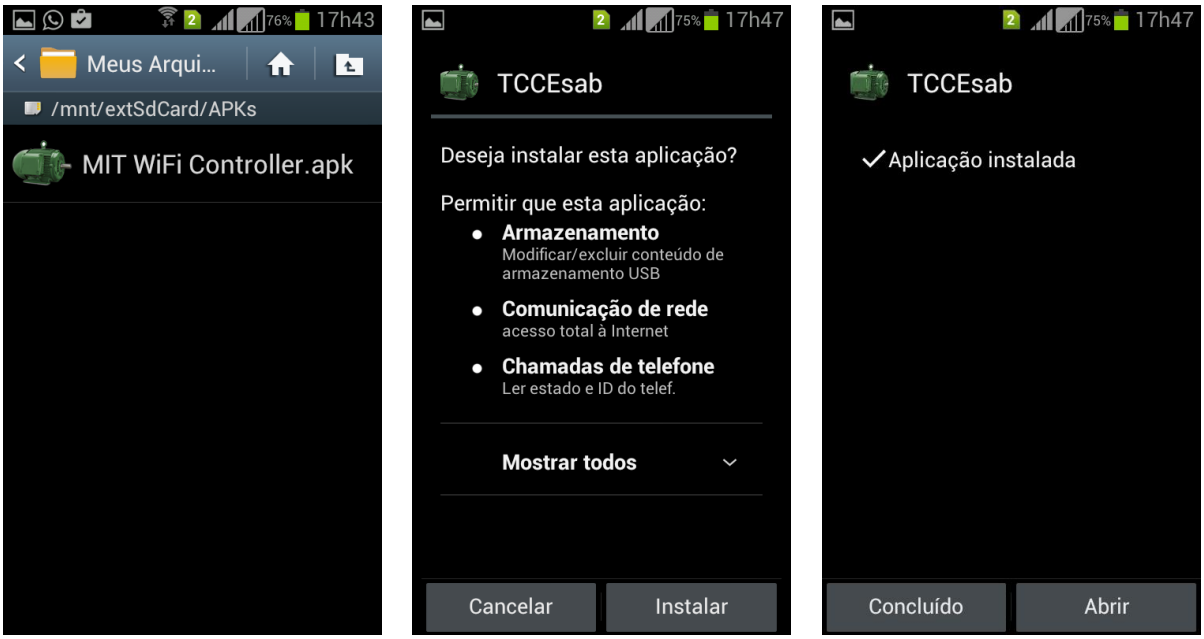


Figura 45 - Inicialização do Sketche.
Fonte: Dados da Pesquisa (2015).

Após a finalização do processo de desenvolvimento e testes do *sketche* controlador do Arduino, deu-se início a geração do software do dispositivo móvel, sendo o mesmo, depois de finalizado, compilado por meio da opção “Build > App (Save .apk to my computer)”. A própria interface Web se encarrega de enviar o arquivo contendo os parâmetros de projeto para ser empacotado já com as opções de segurança predefinidos pelos servidores do MIT. Como o programa não está na loja de aplicativos do Google, faz-se necessário autorizar que o sistema operacional do aparelho instale aplicações não Market. Isso é feito acessando a opção “Configurações > Segurança > Administração do dispositivo > Fontes desconhecidas > Permitir a instalação de aplicativos fora da PlayStore”. A figura 46 ilustra essa situação.

Figura 46 - Autorização de fontes de App Desconhecidas no Android.
Fonte: Dados da Pesquisa (2015).

Dessa forma, o Android permitirá que a instalação prossiga normalmente. Então, conforme mostra a figura 47, basta copiar o arquivo ".apk" para o dispositivo celular (a) e instalá-lo normalmente (b). Após isso, o sistema emitirá uma mensagem avisando que o aplicativo foi instalado com sucesso e que se encontra pronto para uso (c).



(a) (b) (c)

Figura 47 - Instalação do Aplicativo no Dispositivo Móvel.
Fonte: Dados da Pesquisa (2015).

Ao abrir o aplicativo, conforme mostra a figura 48, o usuário se depara com uma tela de autenticação na qual o mesmo precisa digitar a senha de acesso às funções nele disponibilizadas (a). Caso a senha esteja correta, a tela principal é aberta (b), no entanto, para que o App funcione é necessário configurar a URL de acesso ao Arduino conforme (c).

(a) (b) (c)

Figura 48 - Inicialização e Configuração do Aplicativo Móvel.
Fonte: Dados da Pesquisa (2015).

Por fim, vem a utilização do App propriamente dita. Conforme ilustrado na figura 49, caso o aparelho celular não esteja conectado ao módulo, o programa irá mostrar a mensagem contida em (a), avisando que o dispositivo não pode obter nenhuma resposta do Arduino. Todavia, se o aparelho celular estiver conectado ao microcontrolador e o usuário clicar no botão “Ligar”, o software enviará a *String* “mit1:1” ao Arduino que a processará. Se a comunicação for bem sucedida (b), o mesmo ativará o seu pino 13 e devolverá a *String* “mitLigado”.

Ao receber o status do motor, o software mostrará a mensagem “MOTOR LIGADO!” na cor verde. Porém, caso o usuário clique no botão desligar o mesmo processo acontece, entretanto, será enviada a *String* “mit1:0” ao Arduino e o mesmo, além de desativar o pino correspondente ao motor considerado, deve responder com a *String*

“mitDesligado”. Ao receber a atualização do status do motor (c), o App mostrará na cor vermelha o texto “MOTOR DESLIGADO!”.

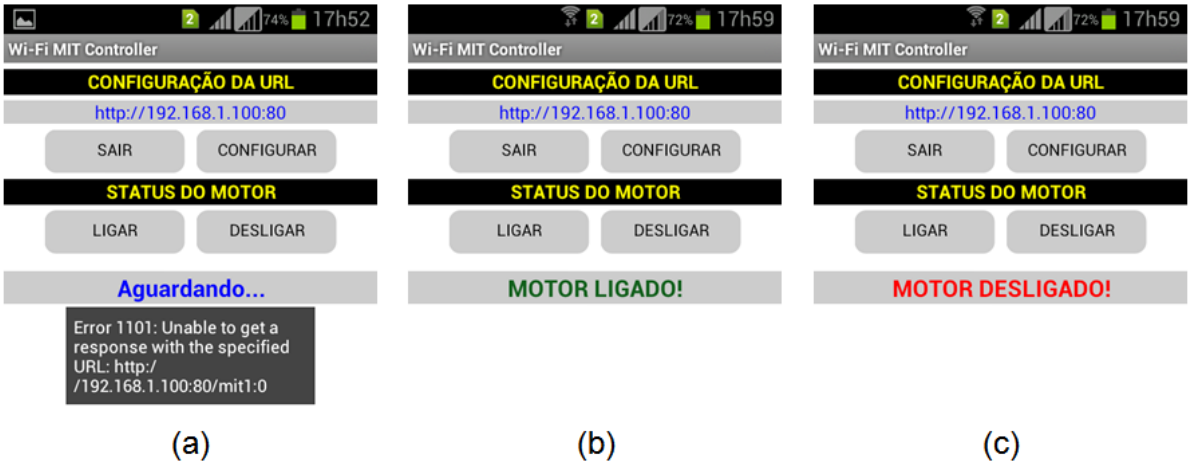


Figura 49 - Ativação do Motor Com o Aplicativo Móvel.
Fonte: Dados da Pesquisa (2015).

Apesar dos objetivos deste trabalho terem sido concluídos com êxito, é necessário ressaltar que o MIT App Inventor não é um substituto às linguagem de programação de aplicativos do Android, muito embora seja essa a premissa à qual o projeto está seguindo. Não é uma ferramenta para criar jogos e aplicativos comerciais. O principal público do App Inventor, no momento, está dentro das universidades, para que alunos e professores possam entender o básico, criando programas para fins educacionais e ajudando a desenvolver um cenário de programação que não irá precisar de conhecimento técnico de programação para funcionar.

Um aplicativo feito no App Inventor pode servir a propósitos específicos para o uso pessoal ou profissional. Pode preencher algumas lacunas nas quais os aplicativos de terceiros pecam, além de levar usuários comuns a se interessarem em começar a programar, elevando assim o nível de qualidade dos aplicativos que são feitos e a quantidade de softwares disponíveis para a plataforma. Não fosse o App Inventor os usuários teriam que se contentar com aplicativos desenvolvidos por terceiros sendo que, muitas vezes, programas específicos possuem apenas versões pagas nas lojas dos desenvolvedores.

Já em relação às dificuldades que tiveram que ser superadas, a mais difícil foi a configuração do Wi-Fi Shield. Tais problemas residiram no fato das informações sobre o

mesmo serem escassas e, quando encontradas, serem disponibilizadas em outros idiomas. Uma busca intensa sobre o processo de configuração do Wi-Fi Shield com o Arduino Mega 2560 foi feita na internet, mas retornou poucos resultados práticos. Entretanto, devido à habilidade do autor com a leitura da língua inglesa, esse desafio foi superado.

Durante todo o processo de configuração do módulo WiFly as únicas fontes de informações realmente úteis e determinantes para o sucesso da empreitada foram o manual de referência da linguagem do Arduino e os *datasheets* disponibilizados pelos fabricantes. Apesar desse contratempo, valeu a pena tal esforço uma vez que o módulo não faz uso de cabo, é baseado no robusto módulo RN-171 que incorpora o protocolo IEEE 802.11b/g, processador de 32 bits, pilha TCP/IP, real time, unidade de gerenciamento de alimentação e entradas para sensores analógicos, além de carregar um firmware desenvolvido pelo fabricante que facilita a integração e minimiza o tempo de desenvolvimento, necessitando apenas de quatro conexões (PWR, TX, RX e GND) para criar uma conexão de dados sem fio, além de se mostrar estável (enviando e recebendo dados sem corrompê-los e sem intermitência).

Dessa forma, apesar desse projeto ter sido feito para acionar apenas uma máquina, na verdade, a única limitação do mesmo são os recursos físicos disponíveis no Arduino (memória, entradas e saídas digitais). Expandindo as capacidades do mesmo, a quantidade de máquinas que poderão ser acionadas aumenta proporcionalmente, abrindo um leque de possibilidades de integração com *softstarters* e inversores de frequência (que será feito em trabalhos futuros).

Outra limitação do presente projeto é que a aquisição de dados de funcionamento em tempo real do motor somente é possível implementando melhorias no código do software do Arduino e, consequentemente, nos circuitos de força por meio da adição de sensores de tensão, corrente elétrica e de temperatura. Tais dispositivos identificarão se a máquina está ou não energizada e retornará esses valores ao sistema controlador, que os processará e decidirá o que fazer com eles, de acordo com os estados informados. Tais melhorias automatizam o processo de partida e parada das máquinas, informando em tempo real a situação de cada uma delas e permitindo identificar situações de sobrecarga nos circuitos elétricos ou no equipamento.

Apesar disso, esse projeto inova no quesito acionamento – por adicionar um sistema de comunicação não convencional nos sistemas locais de comandos de máquinas elétricas – que são as redes wireless. Nos testes realizados, a rede WLAN criada entre o aparelho celular e o módulo Wi-Fi Shield se comportou conforme planejado. Não havia interferências no local dos testes que afetassem o aparelho celular, portanto, as especificações do protocolo foram atendidas sendo verificada a distância máxima permitida entre os dispositivos.

Dentro desses limites e resolvido o problema da interferência do roteador nas UARTs do Arduino, os comandos enviados pelo aparelho móvel foram recebidos sem falhas pelo sistema de controle, que os processou instantaneamente e forneceu as respostas esperadas: ligar o motor respectivo, caso fosse apertado o botão “Ligar” ou desativá-lo, caso fosse pressionado o botão “Desligar”.

O envio e o recebimento dos comandos foram testados em diferentes situações. Com visada direta, o roteador foi capaz de captar o sinal do aparelho quando o mesmo se encontrava até 20 m de distância. Também foi possível o processamento de sinais corretamente mesmo quando havia obstáculos entre os dois dispositivos. Alguns testes foram feitos com obstruções entre o aparelho celular e o roteador (que foi isolado em um local com a porta fechada e sem janela). Nessas condições, o mesmo conseguiu receber corretamente os comandos oriundos do dispositivo móvel mesmo quando separados por duas paredes espaçadas de 5 m, ou uma laje de 15 cm de espessura.

Na situação onde a conexão com o roteador se encontrava na distância limite, o maior tempo entre o envio, recebimento, processamento e a consequente ativação/desativação do motor foi próximo a 800 ms e se deu no momento em que os dois dispositivos se encontravam separados por duas paredes com espessura de 15 cm, cada uma. Dentro do campo de alcance do sinal do roteador com visada direta, esse processo ocorria instantaneamente, mostrando-se versátil, rápido e estável na aplicação à qual se destinou.

Outra situação que mereceu atenção foi a partida direta de motores elétricos. Os motores de indução são tirados da inércia por correntes elétricas muitas vezes superiores à nominal, tipicamente de seis a oito vezes. Nos testes realizados utilizando motor de indução com rotor em gaiola de esquilo funcionando em vazio, isso não chegou a ser um problema. No entanto, em motores com potências superiores a 3 CVs, faz-se necessária a implementação de um método de partida que limite a corrente máxima absorvida pela máquina. Caso essa condição não seja observada, acarretará sobrecarga na fiação dos circuitos elétricos da instalação da máquina, quedas de tensão na rede e atuação da proteção devida à alta corrente solicitada pelo equipamento.

No quesito segurança, foi implementado o sistema de autenticação oferecido pelo protocolo, por meio de uma senha. Somente poderia se conectar ao Arduino e enviar comandos ao mesmo quem tivesse a aplicação instalada em seu aparelho ou acesso a algum *brownser* cuja máquina estivesse associada ao AP e soubesse os comandos aos quais o Arduino responde. Algumas funcionalidades interessantes podem ser implementadas futuramente no software do dispositivo móvel, tais como: alteração de senha e verificar se a rede está ativada antes de solicitar a autenticação. Além disso, não foram instaladas chaves de comando manual no circuito de potência, construído especificamente para esse fim, e que acumula tanto essa função como a de alocar todos os demais dispositivos de acionamento, controle e comando de força do sistema.

Boa parte do período de desenvolvimento e testes do presente trabalho esteve voltada à configuração da plataforma de software, tanto na preparação do sistema operacional e seus pré-requisitos quanto dos programas (Arduino e App Inventor) propriamente ditos. Tais circunstâncias demandaram uma curva de aprendizado elevado, uma vez que o projeto aqui desenvolvido requereu o estudo de três linguagens de programação (App Inventor, Wiring e HTML), entendimento do funcionamento dos protocolos TCP/IP e HTTP, além da construção de diversos componentes de hardware.

Por fim, um dado que merece bastante atenção é que a grande parcela da força motriz que de fato utiliza MITs está nas indústrias. Quando se trata de tecnologia

wireless, tais sistemas requerem simplicidade, robustez, orientação para entender o processo e coexistência entre as tecnologias. Apesar dos sistemas wireless já terem conquistado o seu espaço no mercado industrial, os usuários desse setor ainda estão confusos quanto às potencialidades e às ciladas que um mundo sem-fio pode vir a oferecer.

A compatibilidade com as redes cabeadas também é uma questão a ser levada em conta. Num mundo onde os negócios e as redes de controle são mantidos separados e os dados são cuidadosamente controlados como dados estratégicos ao serem repassados deste nível para o nível do controle do negócio, a tecnologia wireless pode abalar esse paradigma, visto que ambas as redes estão compartilhando as mesmas ondas de rádio.

Outros obstáculos que entravam a adoção das redes sem-fio nas indústrias são os padrões numerosos e conflitantes nos quais as frequências são difíceis de serem entendidas e gerenciadas pelo novo usuário. O custo também deve ser levado em conta, pois só a parte de economia de capital é prontamente quantificada. Novas aplicações exigem novos estudos para determinar as verdadeiras economias, a eficiência e as melhorias.

A orientação se torna um fator importante para o sucesso da tecnologia wireless. Apesar de, hoje em dia, existirem milhões de transmissores inteligentes em plantas industriais, estudos mostram que as informações de diagnósticos ainda estão sendo relativamente subutilizadas. Apesar das tecnologias sem-fios poderem liberar esses dados represados, de fato é necessário deixar que os usuários constatem o potencial de tais conhecimentos. Na maioria dos casos, ocorre uma subcompreensão das informações disponíveis e de como se pode usá-las.

5. CONSIDERAÇÕES FINAIS

O sistema idealizado no presente trabalho foi desenvolvido e funcionou conforme o esperado. Em relação ao objetivo principal deste estudo, que envolveu o projeto e os testes do sistema de acionamento remoto de MITs utilizando a tecnologia *wireless* Wi-Fi e a plataforma microcontrolada Arduino, foi alcançado com sucesso. Cada um dos objetivos específicos os quais a presente pesquisa se propôs a realizar também foi finalizado com êxito, isto, é: a especificação e implementação do sistema de comunicação entre o dispositivo móvel e o sistema de controle central (Arduino), o projeto, desenvolvimento e montagem do circuito de acoplamento responsável pela interação entre o microcontrolador e o circuito de força do MIT foram construídos e testados com sucesso, funcionando satisfatoriamente bem.

A especificação, testes e configurações do dispositivo que proveio capacidade de comunicação Wi-Fi ao Arduino (WiFly RN-XV), bem como a descrição, aferição e parametrização do roteador Wi-Fi no modo de infraestrutura foram também executados com êxito. A escrita e averiguação do programa responsável pelo desempenho e funcionamento do sistema de controle central, igualmente, do desenvolvimento e avaliação do software responsável pelos comandos entre o dispositivo móvel Android e a plataforma microprocessada Arduino também foram finalizados de maneira aceitável. A idealização e montagem do circuito de força da etapa de potência, responsável pelo chaveamento da alimentação do MIT, foi a última etapa do presente trabalho, sendo verificado o seu correto funcionamento.

As hipóteses de McRoberts (2011), referentes as vantagem do Arduino sobre as outras plataformas de desenvolvimento de microcontroladores serem em relação à facilidade de sua utilização, puderam ser comprovadas aqui. As pessoas, mesmo as que não são da área técnica podem, rapidamente, aprenderem o básico e criarem seus próprios projetos em um intervalo de tempo relativamente curto. O Arduino pode ser utilizado para desenvolver objetos interativos independentes ou pode ser conectado a um computador, a uma rede, ou até mesmo à Internet para recuperar e enviar dados e atuar sobre eles (funcionalidades difíceis de serem implementadas nos microcontroladores tradicionais). No caso do presente projeto, ele foi utilizado

para ligar e desligar, de forma satisfatória, um motor trifásico de indução (MIT) à distância, fazendo uso exclusivamente de um dispositivo móvel conectado à mesma rede que o mesmo.

Na presente pesquisa ficou também evidenciado que as faixas de frequências utilizadas pelos dispositivos Wi-Fi – nome dado ao sistema Ethernet wireless – é a mesma conhecida como ISM (Industrial, Scientific and Medical), que são faixas liberadas para uso geral e não necessitam de licenciamento. Se isto facilita a sua utilização devido à eliminação da burocracia, por outro lado, também faz com que esses dispositivos tenham que conviver com outras fontes de radiofrequência (RF) na mesma faixa, a exemplo dos telefones sem fio, Bluetooth e até mesmo fornos de microondas, como descrito em Rappaport (2009).

Com isso, comprovou-se que os dispositivos de redes sem fios devem possuir meios de continuar operando em áreas onde outros equipamentos de RF estejam compartilhando a faixa do espectro (2,4 GHz para os padrões IEEE 802.11b/g). Os próprios fabricantes de equipamentos *wireless* disponibilizam softwares para medição do nível de sinal e taxa de transferência em redes WI-FI para serem utilizados em um *notebook* ou PDA (como o modelo AirMagnet Handheld da empresa Fluke Networks). Caso as fontes de interferência não sejam outras redes sem-fios, uma pesquisa mais profunda com equipamentos mais específicos será necessária. Nos casos onde os equipamentos elétricos não transmitem ou interferem em frequências conhecidas, deve-se fazer o mapeamento eletromagnético com o auxílio de um analisador de espectro e antenas adequadas à faixa de freqüência da rede utilizada, conforme recomendações de Kurose e Ross (2010).

Dessa forma, a conclusão é de que a tecnologia *wireless* abriu um espaço amplo, porém nebuloso, e que ainda traz muitas incertezas aos usuários das tecnologias industriais. Eles estão sendo confrontados com a necessidade de provar a robustez da nova tecnologia em termos que só são compreendidos num ambiente com fio. Por essas ponderações, fica óbvio que esse projeto não está maduro o suficiente para ser implementado e utilizado em acionamentos de plantas industriais, entretanto, sua eficiência é significativa quando consideradas as suas aplicações nas edificações residenciais e comerciais, uma vez que esses ambientes não possuem a

severidade encontrada nas indústrias, conclusões estas também demonstradas por Santos (2010).

Por fim, percebeu-se que há a possibilidade de aprofundamento sobre o tema, uma vez que os MITs são utilizados em diversas atividades e o seu maior inconveniente é a alta corrente de partida. Dessa forma, pode-se pesquisar como controlá-los por meio da rede Wi-Fi utilizando *softstarters* e inversores de frequência. Além disso, a capacidade de processamento e a quantidade de memória física disponível no Arduino são bastante reduzidos, então, é possível adicionar velocidade, versatilidade e recursos de hardware por meio da configuração de um servidor Web rodando Apache Web Server em um host dedicado, mas como isso pode ser feito? Tais estudos futuros visam garantir maior robustez a presente pesquisa, além de aprofundar ainda mais o tema aqui tratado.

REFERÊNCIAS

ANATEL. **Regulamento do Serviço de Comunicação Multimídia**. Disponível em: <http://legislacao.anatel.gov.br/resolucoes/2013/465-resolucao-614>. Acesso em: 07 de fev. 2015.

ARDUINO. **Reference Manual of the Arduino Mega 2560**. Disponível em: <http://arduino.cc/en/Main/arduinoBoardMega2560>. Acesso em: 15 abr. 2014a.

ARDUINO. **Language Reference**. Disponível em: <http://arduino.cc/en/Reference/HomePage>. Acesso em: 15 abr. 2014b.

ATMEL. **Datasheet 8-bit Atmel Microcontroller With 64K/128K/256K Bytes In-System Programmable Flash (ATmega640/V, ATmega1280/V, ATmega1281/V, ATmega2560/V and ATmega2561/V)**. Disponível em: <http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf>. Acesso em: 15 abr. 2014.

CARVALHO, Geraldo. **Máquinas Elétricas: Teoria e ensaios**. 4. ed. São Paulo, SP: Érica, 2011.

CRUZ, Eduardo. **Eletricidade Aplicada em Corrente Contínua: Teoria e exercícios**. 2. ed. São Paulo, SP: Érica, 2007.

CUNHA, Tiago; VEGA, Alexandre. **Tutorial Sobre Confecção Artesanal de Circuitos Impressos**. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/apostilas/arduino/tutorial_confeccao_circuito_impresso.pdf>. Acesso em: 10 set. 2014.

DARCEY, Lauren; CONDER, Shane. **Desenvolvimento de Aplicativos Wireless Para o Android**. 3. ed. São Paulo, SP: Ciência Moderna, 2012. 1 vol.

D-LINK. **D-Support for Wireless LAN. D-PR: D-Linker Professional Resellers for wireless**. 2002.

DUARTE, Carlos. **A Evolução dos Protocolos de Segurança das Redes Sem Fio: Do WEP ao WPA 2 Passando Pelo WPA**. 2010. Monografia de Pós-graduação em Redes de Computadores, Escola Superior Aberta do Brasil (ESAB), Vila Velha (ES), 2010.

DUCKETT, Jon. **Introdução à Programação Web com HTML, XHTML e CSS**. 2. ed. São Paulo, SP: Ciência Moderna, 2012.

FILHO, Pedro; DIAS, Yuri. 2008. 60f. **Acionamento de Potência para Rede de Automação Wireless**. Graduação em Engenharia de Controle e Automação, Faculdade de Tecnologia, Universidade de Brasília (UNB), Brasília (DF), 2008.

FRANCHI, Claiton. **Acionamentos Elétricos**. 3. ed. São Paulo, SP: Érica, 2008.

HUNT, Darran. **WiFly RN-XV Arduino Library**. Disponível em: <https://github.com/harlequin-tech/WiFlyHQ>. Acesso em: 03 jul. 2014.

IETF. **Hypertext Transfer Protocol: HTTP/1.0**. Disponível em: <http://tools.ietf.org/html/rfc1945>. Acesso em: 12 jun. 2014a.

IETF. **Hypertext Transfer Protocol: HTTP/1.1**. Disponível em: <https://www.ietf.org/rfc/rfc2616.txt>. Acesso em: 12 jun. 2014b.

ITAUTEC. **Manual Eletrônico (W7655)**. Disponível em: <https://s3-sa-east-1.amazonaws.com/driversitautec/TI/W7655/W7655_v_1.5.pdf>. Acesso em: 15 abr. 2014.

JUNIOR, Antônio. **A Computação Pessoal e o Sistema Operacional Linux**. 1. ed. Guariba, SP: Espírito Livre, 2007.

KUROSE, James; ROSS, Keitch. **Redes de Computadores e a Internet: Uma abordagem Top-Down**. 5. ed. São Paulo, SP: Pearson Education, 2010.

LdG. **Manual do Wi-Fi Shield**. Disponível em: <http://labdegaragem.com.br/loja/wifishield.pdf>. Acesso em: 13 abr. 2014.

LECHETA, Ricardo. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. 2. ed. São Paulo, SP: Novatec, 2010.

LUBBERS, Peter; ALBERTS, Brian; SALIM, Frank. **Programação Profissional em HTML 5: APIs poderosas para o desenvolvimento de aplicações para a internet com mais recursos**. 1. ed. Rio de Janeiro, RJ: Alta Books, 2013.

LUGLI, Alexandre; SOBRINHO, Darlan. **Tecnologias Wireless Para Automação Industrial: Wireless Hart, Bluetooth, Wisa, Wi-Fi, Zigbee e SP-100**. Disponível em: <http://www.inatel.br/biblioteca/index.php/producao-cientifica/artigos-cientificos-2/doc_download/6771-redes-wireless-para-automacao-industrial>. Acesso em: 21 abr. 2014.

MARGOLIS, Michael. **Arduino Cookbook**. 2. ed. Sebastopol, CA (EUA): O'Reilly, 2012.

MARQUES, António. **O Guia Prático das Redes Sem Fios**. 1. ed. Lisboa (PRT): Centro Atlantico, 2004.

MCROBERTS, Michael. **Arduino Básico**. 1. ed. São Paulo, SP: Novatec, 2011.

MICROSHIP. **Data Sheet of Module RN-171-XV**. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/rn-171-xv-ds-v1.04r.pdf >. Acesso em: 13 abr. 2014a.

MICROSHIP. **RN-XV Dual Relay Evaluation Board User's Guide**. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/50002222A.pdf>. Acesso em: 13 abr. 2014b.

MICROSHIP. **WiFly Command Reference, Advanced Features and Applications User's Guide**. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/50002230A.pdf>. Acesso em: 13 abr. 2014c.

MICROSOFT. **O Modelo TCP/IP**. Disponível em: <http://technet.microsoft.com/pt-br/library/cc786900%28v=ws.10%29.aspx>. Acesso em: 21 abr. 2014.

MORENO, Edward; PEREIRA, Fábio; CHIARAMONTE, Rodolfo. **Criptografia em Software e Hardware**. São Paulo, SP: Novatec, 2005.

MORIMOTO, Carlos. **Redes: Guia Prático**. 2. ed. Porto Alegre, RS: Sul Editores, 2008.

NEMETH, Evi. et al. **Manual Completo do Linux: Guia do administrador**. 1. ed. São Paulo, SP: Pearson Makron Books, 2004.

NETGEAR. **108 Mbps Wireless Router WGT624 v4 (Reference Manual)**. Disponível em: <http://www.downloads.netgear.com/files/WGT624v4_RM_22May07.pdf>. Acesso em: 15 abr. 2014.

ORTEGA, Maurício. **Diagrama Esquemático do Wifly Shield**. Disponível em: <http://www.labdegaragem.com.br/loja/schem/Wifly_Mod.pdf>. Acesso em: 26 jun. 2014.

RAPPAPORT, Theodore. **Comunicações Sem Fios: Princípios e práticas**. 2. ed. São Paulo, SP: Pearson Prentice Hall, 2009.

ROVING NETWORKS. **User Manual and Command Reference: 802.11 b/g Wireless LAN Modules**. Los Gatos, CA (EUA): University Avenue, 2011.

SAMSUNG. **Manual do Usuário (GT-S6102B)**. Disponível em: <http://downloadcenter.samsung.com/content/UM/201205/20120515012555752/GT-S6102B_Emb_BR.pdf>. Acesso em: 15 abr. 2014.

SANTOS, Fábio. **Implantação de Redes Wireless para Melhoria do Controle e Monitoramento de Automação Industrial**. 2011. 63f. Trabalho de Graduação - FATEC de São José dos Campos (SP).

SILVA, Maurício. **Criando Sites com HTML: Sites de Alta Qualidade com HTML e CSS**. 1. ed. São Paulo, SP: Novatec, 2008.

SPARKFUN. **WiFly Shield Hookup Guide**. Disponível em: <https://learn.sparkfun.com/tutorials/wifly-shield-hookup-guide>. Acesso em: 17 jun. 2014.

STEMMER, Marcelo. **Sistemas Distribuídos e Redes de Computadores Para Controle e Automação Industrial**. (Apostila). Centro Tecnológico - Departamento de Automação e Sistemas. Florianópolis, SC: UFSC, 2001.

SYMANTEC. **Uma Atualização sobre a Segurança Wireless**. Disponível em: <http://www.symantec.com/region/br/enterprisesecurity/content/framework/BR_4708.html>. Acesso em: 21 abr. 2010.

TANENBAUM, Andrew. **Redes de Computadores**. 4. ed. Rio de Janeiro, RJ: Campus, 2003.

TORRES, Gabriel. **Redes de Computadores: Curso completo**. Rio de Janeiro, RJ: Axcel Books do Brasil, 2001.

VAL, Carlos. **Ubuntu: Guia do Iniciante**. 1. ed. Vitória, ES: Espírito Livre, 2010.

WOLBER, David. et al. **AppInventor: Create Your Own Android Apps**. 1. ed. Sebastopol, CA (EUA): O'Reilly, 2011.