



Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica

IGOR ALMEIDA BARCELOS

**IMPLEMENTAÇÃO DE PROTOCOLO DE COMUNICAÇÃO
INDUSTRIAL EM UM SISTEMA EMBARCADO**

Uberlândia
2018



Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica

IGOR ALMEIDA BARCELOS

**IMPLEMENTAÇÃO DE PROTOCOLO DE COMUNICAÇÃO
INDUSTRIAL EM UM SISTEMA EMBARCADO**

Trabalho apresentado como requisito parcial de avaliação na disciplina Trabalho de Conclusão de Curso 2 do Curso de Engenharia Elétrica com ênfase em Controle e Automação da Universidade Federal de Uberlândia.

Orientador: Dr. Josué Silva de Moraes

Assinatura do Orientador

Uberlândia
2018



Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica

IGOR ALMEIDA BARCELOS

**IMPLEMENTAÇÃO DE PROTOCOLO DE COMUNICAÇÃO
INDUSTRIAL EM UM SISTEMA EMBARCADO**

Trabalho apresentado como requisito parcial de avaliação na disciplina Trabalho de Conclusão de Curso 2 do Curso de Engenharia Elétrica com ênfase em Controle e Automação da Universidade Federal de Uberlândia.

Uberlândia, 14 de dezembro de 2018

Banca Examinadora:

Prof. Dr. Josué Silva de Moraes
(Orientador – UFU)

Prof. Dr. Márcio José da Cunha
(Examinador – UFU)

Prof. Aline de Cássia Magalhães
(Examinador – UFU)

Dedico este trabalho aos meus pais e à minha
namorada, pelo apoio, carinho e
compreensão.

AGRADECIMENTOS

Ao Professor Dr. Josué Silva de Moraes pelo apoio e orientação durante o TCC e toda a graduação.

Aos Professores Dr. Márcio José da Cunha e Dr. Renato Ferreira Fernandes Jr. pelo suporte prestado no decorrer do desenvolvimento deste trabalho.

Aos meus pais, por todos os ensinamentos e esforços para que eu pudesse chegar até aqui.

À minha namorada, pelo apoio incondicional.

BARCELOS, Igor Almeida. **Implementação de Protocolo de Comunicação Industrial em um Sistema Embarcado**. 2018. 70 p. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Universidade Federal de Uberlândia, Uberlândia, 2018.

RESUMO

O projeto apresentado neste trabalho de conclusão de curso constitui-se de uma proposta de implementação do protocolo Modbus em aplicações contemporâneas, relacionadas à Internet das Coisas (IoT), fazendo uso de programação em linguagem C.

Aliando a simplicidade e o pioneirismo do protocolo Modbus com a versatilidade da Beaglebone Black, foram mostrados os passos do desenvolvimento de um protótipo para controle de luminosidade que pode ser utilizado tanto na indústria quanto em residências, exemplificando a possibilidade de elaboração de aplicações que poderão ser desenvolvidas com base nesta, utilizando a mesma lógica de comunicação, leitura de informações do ambiente, otimização de recursos e automatização de tarefas.

Palavras-chave: Modbus, Modbus TCP, Beaglebone Black, IoT, Domótica, Indústria 4.0, Freemodbus.

BARCELOS, Igor Almeida. **Implementation of Industrial Communication Protocol in an Embedded System**. 2018. 70 p. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Universidade Federal de Uberlândia, Uberlândia, 2018.

ABSTRACT

The project presented in this work is a proposal for the implementation of the Modbus protocol in contemporary applications related to the Internet of Things (IoT), making use of programming in C language.

Combining the simplicity and pioneering of the Modbus protocol with the versatility of Beaglebone Black, the steps of the development of a prototype for light control that can be used both in industry and in homes were shown, exemplifying the possibility of applications elaboration, that could be developed on the basis of this, using the same communication logic, reading information from the environment, optimizing resources and automating tasks.

Keywords: Modbus, Modbus TCP, Beaglebone Black, IoT, Home Automation, Industry 4.0, Freemodbus.

Índice Geral:

1 INTRODUÇÃO	13
2 CONCEITOS	16
2.1 IoT	16
2.2 INDÚSTRIA 4.0	19
2.3 DOMÓTICA	22
2.4 REDES (CONCEITO DE COMUNICAÇÃO)	26
2.5 PROTOCOLO MODBUS	31
2.5.1 Modbus RTU	32
2.5.2 Modbus TCP	35
3 DESENVOLVIMENTO DO PROJETO	39
3.1 FERRAMENTAS UTILIZADAS	39
3.1.1 Beaglebone Black	39
3.1.2 FreeModbus	42
3.1.3 Modsim/Modscan	43
3.1.4 AWS Cloud9 IDE	43
3.1.5 Node-Red	44
3.2 ETAPAS	46
3.2.1 Comunicação entre Modscan e Modsim	47
3.2.2 Comunicação entre Beaglebone e Modscan	48
3.2.3 Elaboração de uma Aplicação	51
3.2.4 Elaboração de Sistema Supervisório Utilizando o Node-Red	56
4 CONCLUSÃO	60
4 REFERÊNCIAS	62
APÊNDICE A – CÓDIGO DE APLICAÇÃO UTILIZADA	67

Índice de Ilustrações:

FIGURA 1 - ESQUEMA DE FUNCIONAMENTO IOT. (FONTE: AUTOR)	17
FIGURA 2 - RESUMO DAS TRÊS REVOLUÇÕES INDUSTRIAIS SEGUIDA PELA QUARTA REVOLUÇÃO, OU INDÚSTRIA 4.0. (FONTE: CITISYSTEMS,2018)	20
FIGURA 3 - COLETA E ANÁLISE DE DADOS NA LINHA. (FONTE: NCSYSTEMS, 2018)	21
FIGURA 4 - SENSORES ACOPLADOS A UM SERVOMOTOR. (FONTE: AUTOR)	22
FIGURA 5 - SÉRIADO THE JETSONS. (FONTE: HANNA-BARBERA PRODUCTIONS, 1985).....	23
FIGURA 6 - EXEMPLO DE COMUNICAÇÃO ENTRE ELEMENTOS UTILIZADOS NA DOMÓTICA. (FONTE: CASADOMO, 2010)	24
FIGURA 7 - ARQUITETURA CENTRALIZADA. (FONTE: CASADOMO, 2008)	25
FIGURA 8 - ARQUITETURA DESCENTRALIZADA. (FONTE: CASADOMO, 2008).....	25
FIGURA 9 - COMPONENTES DA INTERNET. (FONTE: KUROSE, 2010)	28
FIGURA 10 - CAMADAS DO MODELO TCP/IP. (FONTE: KUROSE, 2010).....	29
FIGURA 11 - CONFIGURAÇÃO DE PORTA SERIAL E MODELO DE TRANSMISSÃO DE BITS. (FONTE: BLOG EMBARCADOS, 2014)	32
FIGURA 12 - FRAME DO MODBUS RTU. (FONTE: FONTE: BLOG EMBARCADOS, 2014)	35
FIGURA 13 - FRAME DO MODBUS TCP. (FONTE: BLOG EMBARCADOS, 2014)	36
FIGURA 14 - FORMATO E CONSTRUÇÃO DE UM FRAME NA PILHA TCP/IP. (FONTE: SIMPLY MODBUS, 2017)	37
FIGURA 15 - POSSIBILIDADE DE CAMADAS COM PROTOCOLOS. (FONTE: MODBUS ORG, 2002)	37
FIGURA 16 - COMPARAÇÃO ENTRE FRAMES DO MODBUS RTU E TCP. (FONTE: SIMPLY MODBUS, 2017)	38
FIGURA 17 - BEAGLEBONE BLACK. (FONTE: BLOG FILIPEFLOP, 2017)	40
FIGURA 18 - IDENTIFICAÇÃO E LOCALIZAÇÃO DOS PINOS DA BEAGLEBONE BLACK. (FONTE: BLOG VANDERBILT AEROSPACE DESIGN LAB, 2016).....	41
FIGURA 19 - IDE CLOUD9 (FONTE: BEAGLEBONE FOR DUMMIES, 2018)	44
FIGURA 20 - IDE NODE-RED. (FONTE: BLOG FILIPEFLOP, 2018)	45
FIGURA 21 - EXEMPLO DE FLUXO BÁSICO NO NODE-RED (FONTE: BLOG FILIPEFLOP, 2018)	45
FIGURA 22 - IDE CLOUD9. (FONTE: AUTOR).....	49
FIGURA 23 - ORGANIZAÇÃO DAS PASTAS NO CLOUD9. (FONTE: AUTOR).....	50
FIGURA 24 - DEFINIÇÃO DO ENDEREÇO E QUANTIDADE DE UNIDADES DE MEMÓRIA. (FONTE: AUTOR)	51
FIGURA 25 - ESQUEMA DE MONTAGEM DO CIRCUITO. (FONTE: BLOG DO LABORATÓRIO MOBILIS - UFOP, 2014)	52
FIGURA 26 - ESQUEMA DE MONTAGEM DO CIRCUITO NA BEAGLEBONE BLACK. (FONTE: BLOG DO LABORATÓRIO MOBILIS - UFOP, 2014)	53
FIGURA 27 - CIRCUITO COMPLETO MONTADO. (FONTE: AUTOR)	54
FIGURA 28 - RESULTADO DE TESTES DA APLICAÇÃO. (FONTE: AUTOR)	55

FIGURA 29 - NÓS UTILIZADOS PARA A INTERFACE GRÁFICA. (FONTE: AUTOR)	57
FIGURA 30 - CONFIGURAÇÃO DO NÓ DE INTERFACE GRÁFICA. (FONTE: AUTOR).....	57
FIGURA 31 - PROJETO UTILIZANDO BLOCOS MODBUS. (FONTE: AUTOR).....	58
FIGURA 32 - CONFIGURAÇÃO DO NÓ DE FUNÇÃO. (FONTE: AUTOR)	58
FIGURA 33 - STATUS DO SUPERVISÓRIO DESENVOLVIDO. (FONTE: AUTOR)	59

Índice de Tabelas:

TABELA 1 - ENDEREÇAMENTO MODBUS. (FONTE: MODBUS ORG, 2002)	33
TABELA 2 - CÓDIGOS DAS FUNÇÕES SUPORTADAS. (FONTE: MODBUS ORG, 2002).....	33
TABELA 3 - PRINCIPAIS VARIÁVEIS DO ALGORITMO. (FONTE: AUTOR)	54
TABELA 4 - FUNÇÃO DAS PRINCIPAIS VARIÁVEIS DO ALGORITMO. (FONTE: AUTOR)	55

LISTA DE ABREVIATURAS E SIGLAS

ACM	Application Specific Integrated Circuits – Circuitos Integrados de Aplicação Específica
ARPA	Advanced Research Projects Agency – Agência de Projetos de Pesquisa Avançados
ARPANET	Advanced Research Projects Agency Network – Agência de Projetos de Pesquisa relaciona
ASCII	American Standard Code for Information Interchange - Código Padrão Norte-americano para Intercâmbio de Informações)
CRC	Cyclic Redundancy Check – Verificação de Redundância Cíclica
CRC-32	Cyclic Redundancy Check em 32 bits – Verificação de Redundância em 32 bits
DDR3	Double data rate type 3 synchronous dynamic random-access memory - memória de acesso aleatório dinâmica síncrona com fluxo de dados duplo tipo 3
eMMC	Multi Media Card – Cartão Multimídia
FTP	File Transfer Protocol – Protocolo de Transferência de Arquivos
HDMI	High-Definition Multimedia Interface – Interface de Multimídia de Alta Definição
HTTP	HyperText Transfer Protocol – Protocolo de Transferência de Hiper Texto
IDE	Integrated Development Environment – Ambiente de Desenvolvimento Integrado
IoT	Internet of Things – Internet das Coisas
IP	Internet Protocol – Protocolo de Internet
IPv4	Internet Protocol Version 4 – Protocolo de Internet Versão 4
IPv6	Internet Protocol Version 6 – Protocolo de Internet Versão 6
ISP	Internet Service Provider – Provedor de Serviço de Internet
JTAG	Joint Test Access Group - Grupo de Acesso de Teste Conjunto
LAN	Local Access Network – Rede de Acesso Local
LCD	Liquid Crystal Display – Display de Cristal Líquido
LDR	Light Dependent Resistor – Resistor Dependente de Luz

LED	Light Emitting Diode – Diodo Emissor de Luz
MBAP	Modbus Application Header – Cabeçalho de Aplicação do Modbus
OPE	Overall Plant Effectiveness – Eficiência Geral da Planta
PDA	Personal Digital Assistants – Assistentes Digitais Pessoais
PC	Personal Computer – Computador Pessoal
PLC	Programmable Logic Controller – Controlador Lógico Programável
RAM	Random Access Memory - Memória de Acesso Randômico
RFID	Radio-Frequency Identification – Identificação por Rádio Frequência
RTU	Remote Terminal Unit – Unidade de Terminal Remoto (Modbus)
SCADA	Modelo de Interface Homem-Máquina
SMTP	Simple Mail Transfer Protocol – Protocolo de Transferência Simples de E-mail
TCP	Transmission Control Protocol – Protocolo de Transmissão Controlada
UDP	User Datagram Protocol – Protocolo de Datagrama do Usuário
USB	Universal Serial Bus – Barramento Serial Universal

1 Introdução

Desde os primórdios da Revolução Industrial, o homem busca otimizações em processos. Esta intenção começou com a indústria têxtil e hoje abrange várias áreas. Com a crescente busca por lucros e resultados, a evolução do cenário industrial mundial mudou e continua a transformação para se adequar aos agressivos parâmetros de competição do mercado. Com isso, a tecnologia e a ciência, se tornaram grandes aliados para o fortalecimento deste setor, o que alimenta o constante progresso tecnológico dos seres humanos.

A tecnologia pode ser compreendida como a soma de um dispositivo, das suas aplicações, contextos sociais de uso, e arranjos sociais e organizacionais que se constituem ao seu redor, no qual se torna uma simbiose entre o homem e a máquina, em que a segunda funciona como elemento cooperante e ativo durante os procedimentos de raciocínio dos seres humanos.

O crescente desenvolvimento da modernização nos levou a um patamar em que a maioria das ações do cotidiano de uma pessoa ocorre com a ajuda de algum tipo de tecnologia. Se a alguns séculos atrás a inovação era usada para aumentar a produção de tecidos, hoje ela é altamente utilizada em vários momentos do dia na vida dos cidadãos, não só no trabalho, mas também nas horas de lazer e descanso. Um dos grandes responsáveis por isso, foi o surgimento da Internet, que é um sistema global de redes de computadores interligadas que utilizam um conjunto próprio de protocolos com a intenção de conectar usuários e máquinas do mundo todo.

“Nos dias atuais, a internet pública é uma rede de computadores mundial, ou seja, é uma rede que interconecta milhões de equipamentos de computação ao redor do mundo. Não se passou muito tempo, tais equipamentos eram PCs tradicionais de mesa, estações de trabalhos com sistemas *Unix* (Plataforma de sistemas operacionais) e os chamados servidores que armazenam e transmitem informações, como páginas da web e mensagens de e-mail.” (KUROSE; ROSS, 2009)

“Entretanto, sistemas finais que até algum tempo atrás não eram componentes tradicionais na internet como agendas digitais (PDAs), TVs,

computadores portáteis, telefones celulares, automóveis, equipamentos de sensoramento ambiental, telas de fotos, sistemas domésticos elétricos e de segurança, câmeras web e até mesmo torradeiras estão sendo cada vez mais conectadas à internet.” (KUROSE; ROSS, 2009)

Esta predisposição em conectar pessoas e objetos, criou uma tendência no mundo da tecnologia chamada de *IoT* (Internet of Things).

“A “Internet das Coisas” se refere a uma revolução tecnológica que tem como objetivo conectar os itens usados no dia a dia à rede mundial de computadores. Cada vez mais surgem eletrodomésticos, meios de transporte e até mesmo tênis, roupas e maçanetas conectadas à Internet e a outros dispositivos, como computadores e smartphones.” (ZAMBARDA, 2014)

“A Internet das Coisas é uma rede robusta de dispositivos, todos com eletrônica, software e sensores integrados que lhes permitem compartilhar e analisar dados. Há quase duas décadas, a *IoT* vem transformando o modo como vivemos, abrindo caminho para soluções ágeis, produtos inovadores, fabricação eficiente e, em última análise, novas maneiras incríveis de fazer negócios.” (INTEL, 2017)

Esta evolução que ocorre no cenário de IoT também demonstra força na indústria, no qual várias aplicações com este conceito estão sendo utilizadas para aumentar a eficiência e produtividade da fábrica, o que por consequência gera mais lucros e aumenta a qualidade do produto.

“Com o advento do uso da internet em larga escala, a indústria iniciou uma nova revolução tecnológica, onde estão em foco a automação e a troca de dados utilizando conceitos principalmente de *IoT* e computação em nuvem, que segundo a Microsoft, é o fornecimento de serviços de computação – servidores, armazenamento, bancos de dados, rede, software, análise e muito mais – pela internet (“a nuvem”).” (MICROSOFT, 2017)

“A indústria 4.0 é um conceito que começou a ser implantado a partir de 2013, e que foi originado a partir de um projeto do governo da Alemanha, voltado para novas estratégias que aliam tecnologia e meios de produção. O fundamento básico dessa nova concepção mostra que as fábricas que conectam máquinas e sistemas têm a

capacidade e autonomia para agendar manutenções, prever falhas em processos e se adaptar a mudanças inesperadas que ocorrem nas etapas de produção. Isso significa uma nova era no âmbito das grandes revoluções industriais, pois, com as fábricas inteligentes, diversos setores do mercado sofrerão o impacto provocado pela mudança dos produtos manufaturados.” (ALGAR TELECOM, 2017)

Outro setor que aproveitou a tendência atual foi a domótica. A possibilidade da presença de dispositivos inteligentes nas casas ampliou as perspectivas de automatização e melhorias em alguns fatores. Agora, a residência pode ser facilmente monitorada à distância e várias aplicações vão facilitar a vida de seus moradores. Este conceito de automação tem a intenção de encontrar soluções para atender a algumas necessidades domésticas, para que o morador possa realizar as tarefas cotidianas com o mínimo de esforço, melhorando a qualidade de vida e fazendo com que a pessoa se dedique a outras atividades que exigem um maior esforço intelectual, além de promover otimização do uso de recursos e segurança.

Uma forma de descrever a domótica atual, é como sendo a de utilização de componentes eletrônicos juntamente com a Tecnologia da Informação em um ambiente residencial para que seja possível realizar o monitoramento e controle dos parâmetros de forma local e/ou remota, e automação de tarefas que não agregam valor, de acordo com a necessidade do usuário.

Devido ao crescente uso das tecnologias citadas, se faz necessária a criação de aplicações que atuem de forma confiável, mas que também sejam acessíveis para a grande parte da população. Com isso, o objetivo deste trabalho é colaborar com o desenvolvimento da automatização residencial e industrial, fazendo uso de sistemas embarcados, softwares *open source* e sensores de baixo custo, que possam se comunicar em uma rede local que utiliza o protocolo Modbus TCP/IP, para a otimização e realização de diversas tarefas importantes do dia-a-dia.

2 Conceitos

Com a intenção de situar o leitor de uma maneira conceitual, são apresentados a seguir alguns conceitos básicos relacionados à *IoT*, Indústria 4.0, Domótica, Redes de Comunicação e Protocolo Modbus.

2.1 IoT

Na sua essência, a *Internet of Things* é apenas um ambiente comum entre vários dispositivos que reúnem informações pertinentes para determinadas aplicações. O termo Internet das Coisas foi criado pelo britânico Kevin FOR, que desenvolveu um sistema de sensores que conectou o mundo físico à internet, em um de seus projetos.

O conceito de *IoT* surgiu como fruto de um trabalho desenvolvido no MIT (Instituto de Tecnologia de Massachusetts), com o objetivo de criar um sistema global de registros de bens utilizando um sistema de numeração único, chamado de *Electronic Product Code*. Assim, para que este objetivo fosse alcançado, foi utilizado a tecnologia de Identificação por radiofrequência (*RFID*), e sensores conectados em uma rede wireless.

O termo Coisa, representa um objeto conectado à rede, que pode ser por exemplo, um animal silvestre rastreado, ou um monitor cardíaco de um paciente em recuperação. Assim, os sensores presentes nestas situações descritas, enviam os dados para a internet que serão utilizados em aplicações, que permitem a interface com o usuário.



Figura 1 - Esquema de funcionamento IoT. (Fonte: Autor)

O *IoT* consiste na ideia de relacionar o “mundo real” com o “mundo digital”, fazendo com que o usuário esteja em constante comunicação e interação com dados, pessoas e objetos. O grande resultado dessa interação é um planeta mais inteligente e responsivo.

De acordo com o cientista David Rose, os objetos encantados citados em contos de fadas vão entrar na vida real. Ele cita o espelho que falava com uma princesa, que já sabia qual era a moça mais bonita do reino. Ao se fazer uma analogia com as tecnologias presentes hoje, este espelho poderia ser uma interface homem-máquina que buscaria a informação em um banco de dados de quem é a mulher mais bonita do mundo naquele instante, e forneceria esta informação ao usuário.

O conceito de *IoT* aplicado a cidades, é denominado de *Smart Cities*, que busca o desenvolvimento de tecnologias aliadas a Internet das Coisas para otimizar processos em uma cidade. Com esta tecnologia é possível desenvolver sistemas de transporte, de controle de geração, distribuição e consumo de energia, descarte de resíduos, e outros, tornando-os mais eficientes e corretos. Como consequência, a qualidade de vida dos moradores e a natureza são beneficiados.

Com o crescente uso de *IoT*, aumenta também a produção de informação. Este alto volume de dados, nos faz pensar em como fazer para armazenar, rastrear, analisar e fazer uso dessa grande quantidade de dados. Para que isso seja feito, a análise de Big Data tem papel fundamental.

Um analista famoso deste setor, Doug Laney, articulou a definição de Big Data como os três V's:

Volume: Organizações coletam dados de uma grande variedade de fontes, incluindo transações comerciais, redes sociais e informações de sensores ou dados transmitidos de máquina a máquina.

Velocidade: Os dados fluem em uma velocidade sem precedentes e devem ser tratados em tempo hábil. *Tags* de RFID, sensores, celulares, e contadores inteligentes estão impulsionando a necessidade de lidar com imensas quantidades de dados em tempo real, ou quase real.

Variedade: Os dados são gerados em todos os tipos de formatos – de dados estruturados, dados numéricos em bancos de dados tradicionais, até documentos de texto não estruturados, e-mail, vídeo, áudio, dados de cotações da bolsa e transações financeiras.

“A importância do Big Data não gira em torno da quantidade de dados que você tem, mas em torno do que você faz com eles. Você pode analisar dados de qualquer fonte para encontrar respostas que permitam redução de custos, redução de tempo, desenvolvimento de novos produtos e ofertas otimizadas, e decisões mais inteligentes.” (SAS, 2017)

Atualmente existem mais objetos na internet do que pessoas, o que nos leva a refletir sobre esse processo. Um grande desafio relacionado ao uso do *IoT* é a “lendária” implementação do endereçamento IPv6. Segundo a CISCO, mundo ficou sem endereços IPv4 em fevereiro de 2010. Embora nenhum impacto real tenha sido notado pelo público geral, esta situação pode diminuir o progresso do *IoT*, pois a tendência é o crescimento no número de sensores que precisam ser endereçados, e o IPv6 facilitaria bastante o gerenciamento de redes devido a recursos de autoconfiguração e oferece recursos de segurança aprimorados. Ou seja, a implementação do IPv6 se faz extremamente necessária para o desenvolvimento das tecnologias futuras.

“Outro problema que precisa ser solucionado rapidamente é a alimentação dos sensores utilizados. Para que a *IoT* atinja seu potencial completo, os sensores deverão ser autossustentáveis. Imagine trocar as baterias de bilhões de dispositivos implantados no planeta inteiro e até mesmo no espaço. Obviamente, isso não é possível. O que é necessário são sensores para gerar eletricidade a partir de

elementos ambientais, como vibrações, luz e fluxo de ar. Em uma descoberta relevante, os cientistas anunciaram um nanogerador comercialmente viável, um chip flexível que usa os movimentos do corpo, como a compressão dos dedos para gerar eletricidade.” (CISCO, 2011)

Quando falamos de vários dispositivos conectados à rede, devemos nos preocupar com a segurança das informações trocadas. Alguém estaria apto a “hackear” a sua casa e assim ameaçar toda a sua família. Por este motivo, a Internet das Coisas poderá aumentar os riscos relacionados à segurança do mundo todo.

O mundo está fadado a sempre evoluir, e o *IoT* nos parece ser um caminho para tal. A conexão entre a coleta e compartilhamento de dados, com segurança e proteção, permanece um desafio-chave no desenvolvimento deste setor. Apesar disso, é um segmento animador e que nos trará enormes benefícios se utilizado da maneira correta.

2.2 Indústria 4.0

Hoje vivemos uma transformação significativa em relação à maneira como fabricamos bens de consumo, graças à digitalização das indústrias. O desenvolvimento está tão robusto, que recebeu o nome de Indústria 4.0 para representar a quarta revolução industrial que ocorre na manufatura. Desde a primeira revolução industrial, que nos levou à mecanização de processos com a utilização de vapor até a segunda revolução industrial, que inovou com produção em massa e linhas de montagem usando eletricidade, a quarta revolução industrial eleva a evolução da automatização iniciada na terceira revolução, com o uso de computadores e aprimoramento da automação com sistemas inteligentes e autônomos, alimentados por dados, utilizando *Machine Learning*.

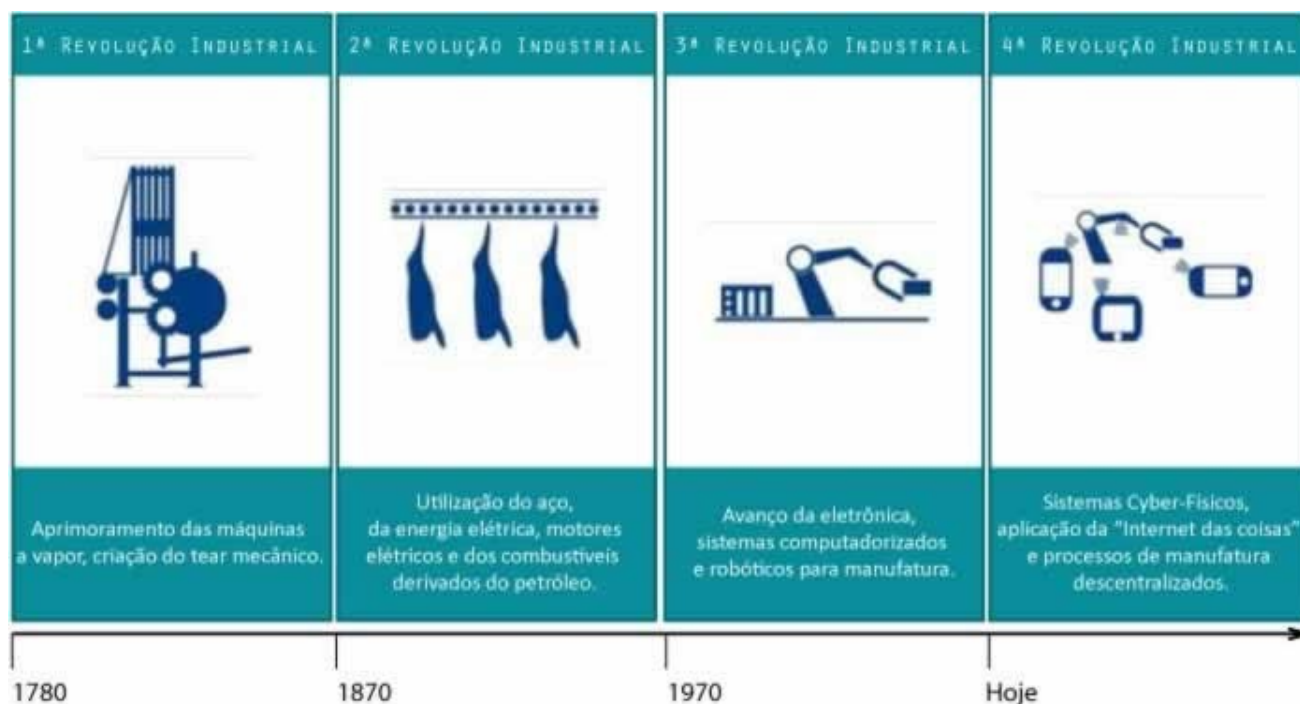


Figura 2 - Resumo das três revoluções industriais seguida pela quarta revolução, ou indústria 4.0.
(Fonte: Citisystems, 2018)

O termo indústria 4.0 é originado de um projeto de estratégia de alta tecnologia do governo alemão, e foi utilizado pela primeira vez em uma feira na cidade de Hanover, na Alemanha, fazendo referência a uma quarta revolução industrial. Este termo se refere a um conjunto de tecnologias que se baseiam nas interações entre os sistemas cyber-físicos (suportes de peças, postos de reunião e produtos), *Big-Data* e *IoT*, no qual ocorre coleta de informações do ambiente industrial para facilitar a tomada de decisões por humanos.

Os fatores que definem a indústria 4.0 são:

- **Interoperabilidade:** Capacidade de comunicação entre sistemas cyber-físicos, humanos e fábricas inteligentes através da internet.
- **Virtualização:** Utilização de sistemas supervisório para permitir o controle e a rastreabilidade de todos os processos da fábrica, por meio de sensores espalhados pela fábrica em pontos estratégicos.
- **Descentralização:** Disponibilização dos dados coletados por sensores em vários locais, para análise.
- **Operação em Tempo Real:** Coleta de dados em pontos estratégicos da fábrica no instante em que ocorrem, para que seja possível tomar decisões no menor tempo possível.

- **Orientação à serviço:** Utilização de arquiteturas de software que possuam aplicações do próprio software disponíveis na internet, como ferramentas de desenvolvimento, plataformas (servidores, armazenamento e comunicação).
- **Modularidade:** Produção de acordo com a demanda recebida, além da flexibilização das linhas para alterar o formato de produto ou *SKU* de acordo com a necessidade.

Portanto, podemos considerar que na Indústria 4.0 temos sensores e sistemas inteligentes que colaboram com a automatização das máquinas, ou seja, os equipamentos recebem informações sobre como devem operar em cada estágio do processo de manufatura, fornecendo dados de feedback, utilizados para o processo de melhoria contínua e eficiência da fábrica (OPE).

Os processos da cadeia de produção devem ter a tendência de autogerenciamento, descentralizados para otimizar a análise e troca de dados através de *Cloud Computing*. Ou seja, na quarta revolução industrial, os sistemas centralizados robustos dão lugar para a inteligência descentralizada com monitoramento e feedback de todos os pontos importantes de uma fábrica.

Com uso de coleta de dados nas linhas de produção da fábrica por meio de sensores, é possível identificar as perdas mais facilmente, o que levam elaboração de ações assertivas contra a causa-raiz dos problemas e por consequência o aumento na eficiência da indústria.



Figura 3 - Coleta e análise de dados na linha. (Fonte: ncsystems, 2018)

Outro exemplo de atuação da indústria 4.0 é no uso da Manutenção Preditiva, que como o próprio nome diz, tem o objetivo de prever quando será necessário a realização de alguma manutenção em equipamentos, evitando assim paradas inesperadas ou no pior caso uma quebra.

A Manutenção Preditiva pode ser feita principalmente por meio de monitoramento de sinais de vibração, temperatura, velocidade e outros.



Figura 4 - Sensores acoplados a um servomotor. (Fonte: Autor)

Seja com a Manutenção Preditiva ou a coleta de dados em geral, a Indústria 4.0 está cada vez mais presente no cotidiano das fábricas, melhorando a eficiência, produtividade e eliminando custos.

2.3 Domótica

A Domótica é uma tecnologia que permite a gestão e automatização de alguns recursos presentes nas habitações, a fim de que o morador obtenha mais conforto e comodidade na realização das suas tarefas diárias. O termo Domótica é resultante da junção das palavras *domus* e informática, o que pode nos levar ao verdadeiro objetivo desta modernidade.

Apesar de ser um termo muito abordado atualmente, a Domótica existe desde a década de 1980, sendo o principal objetivo controlar a iluminação, climatização e a segurança em alguns edifícios.

“A Automação Residencial pode ser definida como um conjunto de tecnologias que ajudam na gestão e execução de tarefas domésticas cotidianas. A sua utilização tem por objetivo proporcionar um maior nível de conforto, comodidade e segurança, além de um menor e mais racional consumo de energia.” (BOLZANI, 2004).

Desde o início do século XXI, existe uma grande tendência em relação a busca de conforto nas residências, tecnologicamente falando. A tecnologia vem sempre evoluindo a passos largos, em vários aspectos, e com isso, logo se pensa na aplicação desta inovação na vida das pessoas, e por consequência nas suas casas.

A ideia de automatizar as residências está presente no imaginário humano já a bastante tempo, como definição de futuro e evolução, onde as máquinas poderiam fazer todo o serviço pesado nas casas e proporcionar tranquilidade e otimização de tempo para os habitantes.



Figura 5 - Seriado The Jetsons. (Fonte: Hanna-Barbera Productions, 1985)

Para que a automação de uma residência ocorra, é necessário a utilização de objetos como sensores, controladores, atuadores, barramentos e interfaces.

“Os sensores são os dispositivos que detectam estímulos, medem e monitoram grandezas físicas e eventos (temperatura, umidade etc.), convertendo-as em um valor passível de manipulação por sistemas computacionais. São eles que

encaminham as informações aos controladores sobre algum evento, para que os controladores possam enviar os comandos adequados para os atuadores.” (ALMEIDA, 2009).

“Controladores controlam os dispositivos automatizados (sensores e atuadores). Monitora as informações dos sensores, podendo enviar comandos para que um atuador ative ou desative algum equipamento. De maneira geral podem possuir interfaces independentes, na forma de um controle remoto, ou serem sofisticadas centrais de automação.” (ALMEIDA, 2009).

“Os atuadores são dispositivos eletromecânicos, que recebem os comandos do sistema de automação e ativam os equipamentos automatizados. São os módulos de acionamento ligados entre a rede elétrica e os equipamentos.” (ALMEIDA, 2009).

“O barramento é o meio físico responsável pelo transporte das informações (rede elétrica, telefônica etc.)” (CASADOMO, 2010).

“As interfaces são os dispositivos ou mecanismos (navegador de internet, celular, painéis, controles remotos, interruptores etc.) que permitem ao usuário visualizar as informações e interagir com o sistema de automação.” (CASADOMO, 2010).

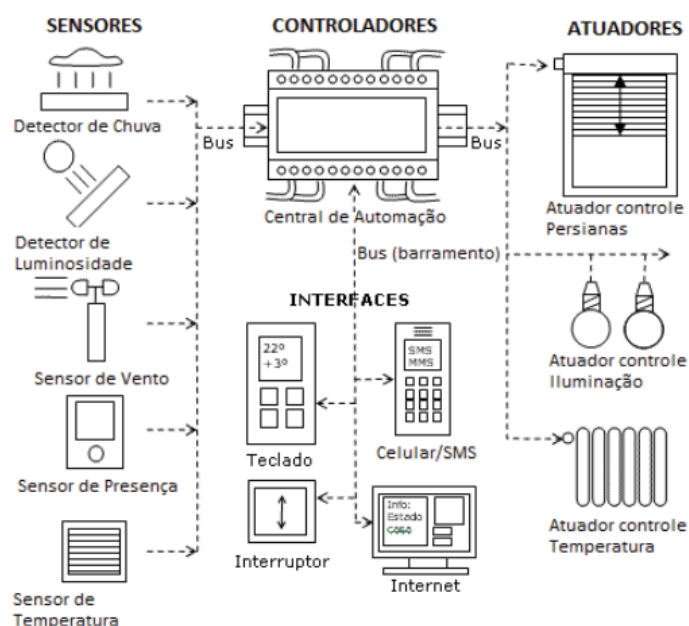


Figura 6 - Exemplo de comunicação entre elementos utilizados na domótica. (Fonte: CASADOMO, 2010)

“Na Automação Residencial, existem dois tipos principais de arquitetura, a centralizada e a descentralizada. Na arquitetura centralizada, todos os dispositivos respondem a um dispositivo central, que deve ser dotado de inteligência e desempenho suficiente para receber e tratar as informações recebidas dos sensores e enviar os comandos aos atuadores.” (ALMEIDA, 2009).

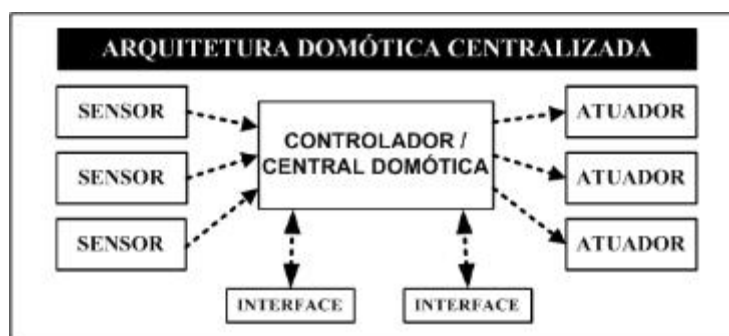


Figura 7 - Arquitetura centralizada. (Fonte: CASADOMO, 2008)

“Em sistemas com arquitetura descentralizada, podem existir vários controladores que, interligados por um barramento, compartilham a administração dos sensores, atuadores e interfaces ligadas aos controladores.” (CASADOMO, 2010).

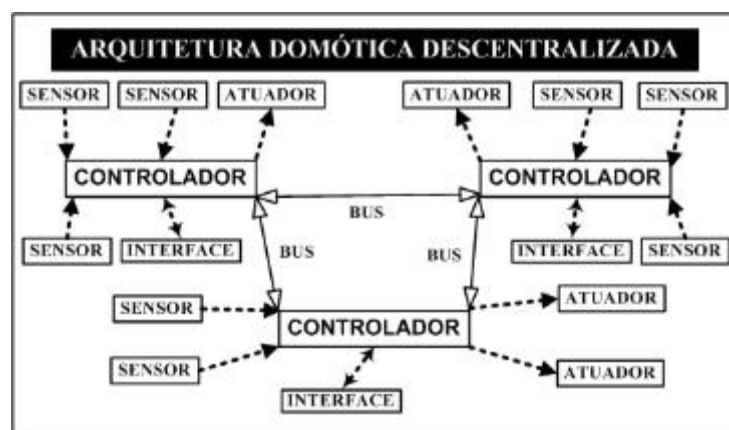


Figura 8 - Arquitetura descentralizada. (Fonte: CASADOMO, 2008)

Nos dias atuais, apesar das novas tecnologias, maior disseminação do conhecimento e melhora nos preços, o mercado relacionado a automação residencial vive incertezas, pois ainda não se encaixa na vida dos usuários de uma maneira essencial e popular. Grande parte das tecnologias criadas se dividem em equipamentos caros e confiáveis, e outros mais baratos, mas que podem deixar a residência em risco. Existe um gap que impede essa popularização, que se resolvido,

poderá melhorar a vida das pessoas como um todo e não só uma pequena porcentagem da população mundial. Unindo a difusão da tecnologia e a elaboração de um sistema de segurança confiável, a Domótica tem tudo para crescer e ser aquilo que promete há tantas décadas.

2.4 Redes (Conceito de Comunicação)

Para iniciar o conteúdo de Redes, é necessário entender um conceito fundamental que é o de comunicação de dados, definido como a troca de informações entre dispositivos constituídos de hardware e software, fazendo uso de um meio físico para a transmissão da informação, no qual deve existir a garantia de que os dados serão enviados e recebidos no destino correto com o cumprimento de um tempo finito, pré-estabelecido.

A eficiência de um sistema de comunicação depende de três características principais:

- **Entrega:** Garantia que a informação será entregue somente ao destino correto.
- **Confiabilidade:** O sistema precisa garantir que os dados serão entregues íntegros.
- **Tempo de Atraso:** Os dados precisam ser entregues dentro de um prazo pré-determinado, pois dados em atraso são pouco úteis para aplicações.

Os principais componentes de um sistema básico de comunicação de dados são:

- **Mensagem:** Informação a ser transmitida.
- **Transmissor:** É o dispositivo (hardware e software) que envia a mensagem.
- **Receptor:** É o dispositivo (hardware e software) que recebe a mensagem.
- **Meio Físico:** É o caminho físico por onde a mensagem é propagada.
- **Protocolo:** É um conjunto de regras que organiza a comunicação de dados entre dispositivos. É um acordo feito entre o transmissor e receptor para a troca de mensagens.

“A comunicação de dados entre locais remotos pode ser realizada através de processo denominado conectividade, que envolve conexão de computadores, meios e ativos de redes (dispositivos).” (Forouzan, 2006).

“Um conjunto de dispositivos (hosts) conectados por links de comunicação ou nós é denominado rede.” (Forouzan, 2006).

Os sistemas finais ou hosts são conectados por enlaces (links) e comutadores de pacotes (roteadores), que são equipamentos que distribuem os pacotes (conjunto de dados) de um enlace de entrada para um enlace de saída.

“São duas as formas possíveis de conexão, ponto a ponto e multiponto. No ponto a ponto existe um link dedicado entre dois dispositivos e toda a capacidade do link é reservada para a comunicação, sendo na maioria das vezes utilizado cabos para a conexão, mas podendo ser utilizado satélite e micro-ondas. No multiponto, mais de dois dispositivos compartilham um único link, a capacidade do canal é compartilhada de forma espacial, ou seja, pela utilização simultânea ou por revezamento chamada de compartilhamento temporal.” (Forouzan, 2006).

Quando o usuário deseja conectar a sua LAN (*Local Area Network*) a uma rede de comunicação global (Internet), é necessário contratar os serviços de *ISPs* (*Internet Service Providers*) ou os Provedores de Serviço de Internet, que são as empresas de telecomunicações que os usuários contratam para obter o fornecimento de acesso à internet. Ou seja, ao contratar este tipo de serviço, o usuário tem a sua disposição todo um serviço de infraestrutura com enlaces, links, roteadores, switches e caminhos para poder acessar outros sistemas finais conectados pelo mundo.

“Os *ISPs* também fornecem acesso a provedores de conteúdo e devem se interconectar através de *ISPs* maiores nacionais ou internacionais com roteadores de alta velocidade e enlaces de fibra ótica, o que é chamado interconexão. Apesar de estarem conectados, cada ISP tem sua gerência de forma independente, utilizando um protocolo IP e seguindo regras de endereçamento.” (Kurose, 2010)

“A informação que trafega utilizando estas redes pode estar em diferentes formas, como caracteres, áudio, imagens etc. Por padrão, um caractere é representado por uma sequência de bits (*Binary Digit*) ou seja, números binários (0 ou

1). Diferentes conjuntos de bits representam os mais diversos caracteres (letras, números, símbolos, imagens), esses conjuntos são denominados códigos e o processo de representação de símbolos codificação.” (Forouzan, 2006)

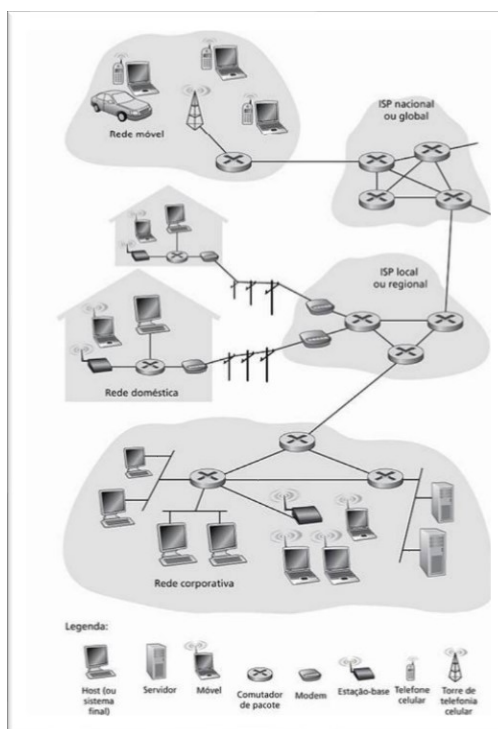


Figura 9 - Componentes da Internet. (Fonte: Kurose, 2010)

Para que o tráfego de dados funcione de forma ordenada e confiável, é necessário uso de Protocolos para estabelecer a comunicação entre os dispositivos. Um dos mais importantes é o TCP (*Transmission Control Protocol* – protocolo de controle de transmissão) e outro é o IP (*Internet Protocol*).

Em um protocolo de rede os componentes de hardware e software trocam mensagens, que possuem os mais diversos fins, seja o estabelecimento de uma informação, até mesmo o monitoramento de um equipamento. “Um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento”. (KUROSE, 2010)

“Redes de computadores, gestores de redes, desenvolvedores e a internet como um todo utilizam intensamente protocolos. Além dos protocolos, são de grande

importância a combinação hardware e software. Hardware é toda a infraestrutura física para transportar os sinais. O software viabiliza o processo de comunicação, afinal são esperados serviços mais complexos do que simplesmente o envio de um sinal.” (Forouzan, 2006).

Para que as interações entre hardware e software ocorram, é necessário a distribuição de algumas atribuições no processo de comunicação de dados que são organizadas de acordo com arquiteturas de camadas. Essas arquiteturas possuem alguns modelos, sendo um dos mais importantes o modelo de camadas da internet, também chamado Modelo da Internet ou pilha de protocolos TCP/IP (Figura 10).



Figura 10 - Camadas do modelo TCP/IP. (Fonte: Kurose, 2010)

De acordo com o modelo de camadas mostrado acima, quando uma mensagem é enviada de um host 1 para o host 2, para que chegue ao seu destino, ela passa por diversas etapas, uma vez que as camadas têm a função de organizar adequadamente o processo de comunicação.

“Os serviços providos em cada camada, quando em uma conexão ativa, conversam apenas com seu par de camada, ou seja, um serviço na camada de aplicação de um host A irá se comunicar com a camada de aplicação do host B, os níveis mais altos do modelo irão chamar os serviços das camadas abaixo. Por exemplo a camada 3 usa os serviços da camada 2 e provê serviços para a camada 4. Os processos em cada host que se comunica na mesma camada são chamados peer-to-peer.” (Forouzan, 2006)

“Esse modelo de camadas traz vantagens conceituais e estruturais, proporcionando um modo estruturado de discutir componentes do sistema, a

modularidade facilita atualizações. Porém uma desvantagem é que a camada pode duplicar a funcionalidade de uma camada inferior.” (Kurose, 2010)

A camada de aplicação possui a função de adequar o conteúdo enviado para outros hosts, de acordo com a necessidade. Nesta camada, estão protocolos importantes que realizam a comunicação fim-a-fim entre as aplicações, como o protocolo SMTP (correio eletrônico), FTP (transferência de arquivos entre dois sistemas finais) e o famigerado HTTP (requisição e transferência de arquivos Web).

A camada de transporte é responsável pelo transporte de mensagens da camada de aplicação entre o transmissor e receptor da aplicação. Os protocolos envolvidos são o TCP e UDP, responsáveis pelo envio das mensagens, com a diferença que: TCP provê serviços orientados a conexão e fornece entrega garantida e controle de fluxo (este protocolo é ideal para aplicações que necessitam de confiabilidade); já o UDP provê serviço não orientado para conexão, porém é um serviço econômico e consegue ser mais rápido que o TCP, o que o faz ideal para streaming.

A camada de rede ou “camada de roteamento” é responsável pelo envio de pacotes ou datagramas de um host a outro. O host de origem passa o segmento da camada de transporte e um endereço de destino para a camada de rede, ou seja, ela entrega o segmento a camada de transporte no destinatário. Possui dois componentes importantes, o protocolo IP, que define os campos no datagrama e o outro é o protocolo de roteamento que define as rotas que estes datagramas seguem da origem ao destino.

A camada de enlace trabalha em conjunto com a camada de rede. A camada de rede roteia o datagrama em uma sequência de roteadores, e para levar o datagrama de um nó a outro ele utiliza os serviços da camada de enlace, que leva o datagrama ao longo da rota ao nó seguinte e entrega a camada de rede. Os serviços prestados dependem do protocolo empregado neste, alguns exemplos de protocolos são *Ethernet*, *Wi-fi* e *PPP*.

Camada física é a camada responsável por movimentar bits individuais, que estão dentro do quadro (conjunto de dados da camada de enlace) para um nó

seguinte. Para este trabalho, outro ponto importante acerca do conhecimento de redes é sua infraestrutura e meios de transmissão utilizados para a comunicação de dados.

2.5 Protocolo Modbus

Modbus é um protocolo de comunicação de dados desenvolvido pela empresa Modicon, publicado em 1979 para uso em programação em Controladores Lógico Programáveis. À princípio, foi desenvolvido para ser um método utilizado na transmissão de informação sobre portas seriais entre dispositivos eletrônicos. O equipamento que requisita informações é chamado de Mestre ou Cliente Modbus, e o que envia as informações requisitadas é denominado Escravo ou Servidor Modbus.

Posteriormente, a Modicon foi adquirida pela Schneider e os direitos do protocolo foram liberados pela Associação Modbus. Atualmente, este é um protocolo aberto, o que significa que não é necessário o pagamento de royalties para o seu uso. Anteriormente, o Modbus havia se tornando um padrão de comunicação entre equipamentos industriais, e ainda hoje é utilizado para conectar equipamentos eletrônicos nas fábricas. Este protocolo vem sendo usado ao longo do tempo por várias indústrias, representando um grande marco na história da automação mundial.

O protocolo Modbus é tipicamente utilizado para transmitir sinais de instrumentos e equipamentos de controle para um controlador principal, ou banco de dados, como um computador. Um exemplo de aplicação pode ser um sistema que mede a temperatura e umidade e que comunica o resultado para um computador. O Modbus é comumente utilizado para conectar um sistema supervisório com um terminal remoto, fazendo uso de sistemas do tipo SCADA.

Existem versões do protocolo para cabos seriais (Modbus RTU e Modbus ASCII) e Ethernet (Modbus TCP).

2.5.1 Modbus RTU

Modbus RTU (Remote Terminal Unit) é um protocolo aberto, que utiliza portas seriais como a RS-232 e RS-485, e é derivado da arquitetura Mestre/Escravo. Cada mensagem de 8 bits contém dois caracteres hexadecimais de 4 bits. Cada mensagem deve ser transmitida em um fluxo contínuo de caracteres.

Em uma rede Modbus RTU, existe um mestre e no máximo 247 escravos, que possuem um endereço único que varia de números inteiros entre 1 a 247. O mestre possui a responsabilidade de escrever e ler as informações contidas nos escravos pertencentes à sua rede.

De uma maneira geral, o protocolo Modbus RTU funciona realizando uma transmissão de dados sobre um cabo serial. A configuração mais simples pode ser considerada como sendo um único cabo serial conectando as portas seriais de dois dispositivos, um mestre e um escravo.

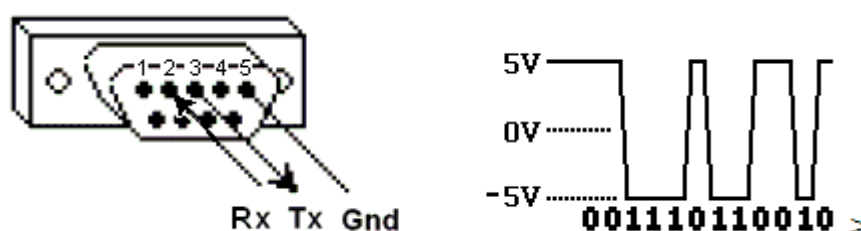


Figura 11 - Configuração de porta serial e modelo de transmissão de bits. (Fonte: Blog Embarcados, 2014)

Os dados são enviados em séries de zeros e uns, que são denominados bits. Cada bit é enviado com sendo uma tensão que varia entre dois valores extremos, sendo um extremo considerado zero, e o outro um. A velocidade de transmissão destes bits é de geralmente 9600 *baud*, ou bits por segundo.

O armazenamento das informações pode ser feito em quatro diferentes tabelas. Duas delas, armazenam valores binários discretos (bobina), e as outras duas restantes armazenam valores numéricos analógicos (registradores). As tabelas de dados discretos e analógicos possuem um esquema onde é permitido somente a leitura, e outra que autoriza a escrita e a leitura.

Cada tabela possui 9999 valores. Cada bobina ou contato equivale a 1 bit, endereçado entre 0000 e 270E.

Cada registrador equivale a 1 *word*, que é a mesma coisa que 16 bits ou 2 bytes, endereçados entre 0000 e 270E.

Bobina/Registrador	Endereçamento	Tipo	Nome da Tabela
1-9999	0000 a 270E	Leitura e Escrita	Saída Discreta
10001-19999	0000 a 270E	Somente Leitura	Entrada Discreta
30001-39999	0000 a 270E	Somente Leitura	Entrada Analógica
40001-49999	0000 a 270E	Leitura e Escrita	Saída Analógica

Tabela 1 - Endereçamento Modbus. (Fonte: Modbus Org, 2002)

Como dito anteriormente, em uma rede, cada escravo tem um endereço único que está entre 1 e 247. Quando o mestre requisita informações, o primeiro byte enviado ao escravo representa o endereço dele. Dessa maneira, cada escravo “sabe” quando a requisição é para ele ou não.

O segundo byte enviado ao escravo pelo mestre é o código da função a ser requisitada. Este número “diz” ao escravo qual tabela deverá ser acessada e qual a localização da informação a ser lida ou escrita.

Código da Função	Descrição
1	Leitura do bloco de bits do tipo <i>coil</i> (saída discreta)
2	Leitura de bloco de bits do tipo entradas discretas
3	Leitura de bloco de registradores do tipo holding
4	Leitura de bloco de registradores do tipo input
5	Escrita em um único bit do tipo <i>coil</i> (saídas discretas)
6	Escrita em um único registrador do tipo holding
7	Ler o conteúdo de 8 estados de exceção
8	Prover testes para verificar a comunicação e erros internos
11	Obter o contador de eventos
12	Obter relatório de eventos
15	Escrita em bloco de bits do tipo <i>coil</i> (saída discreta)
16	Escrita em bloco de registradores do tipo holding
17	Ler algumas informações do dispositivo
20	Ler informações de um arquivo
21	Escrever informações em um arquivo
22	Modificar o conteúdo de registradores de espera através de operações lógicas
23	Combinar leitura e escrita em registradores numa única transação
24	Ler o conteúdo da fila FIFO de registradores
43	Identificação do modelo do dispositivo

Tabela 2 - Códigos das funções suportadas. (Fonte: Modbus Org, 2002)

Para que os bits enviados e recebidos sejam verificados, é utilizado o CRC (*Cyclic Redundancy Check*). Os últimos dois bytes de cada frame, no Modbus RTU são utilizados para a detecção de erros. Cada byte na mensagem é utilizado para calcular o CRC. O dispositivo receptor da mensagem também realiza o cálculo, e compara com o CRC obtido pelo emissor. Com isso, caso haja alguma diferença entre os bits do CRC, será enviada uma mensagem específica, correspondente ao erro resultante.

Outro ponto importante é a definição na ordem em que os dados são armazenados nos registradores, e isso pode ser feito de duas formas, transmitindo o byte menos significativo primeiro (*little endian*), ou então o mais significativo (*big endian*).

Para o melhor entendimento do processo de comunicação genérico entre um mestre e um escravo, abaixo pode ser verificado um exemplo, no qual será utilizado a função de código 05.

Este comando escreve um bit na bobina número 173 do escravo número 17 para que este fique em estado de ligado.

Mensagem de requisição: 11 05 00AC FF00 4E8B

Onde:

- **11**: O endereço do escravo (11 hex = endereço 17);
- **05**: O código da função (Escrever em uma única bobina);
- **00AC**: O endereço da informação contida na bobina. (bobina #173 – 1 = 172 = AC hex) em que, (00AC hex = 172, +1 offset = bobina #173);
- **FF00**: O valor a ser escrito na bobina (FF00 = Ligado, 0000 = Desligado);
- **4E8B**: O CRC para verificação de erros.

A resposta normalmente esperada é a repetição da requisição, após a escrita ser realizada na bobina especificada.

Resposta: 11 05 00AC FF00 4E8B

*Texto adaptado de www.simplymodbus.ca.

A indicação de quando uma nova mensagem enviada começa ou termina é feita pela ausência de transmissão de dados na rede, que é considerado um tempo mínimo de 3,5 vezes o tempo de transmissão médio. Assim, os equipamentos irão assumir que o primeiro caractere enviado após o período de “silêncio”, corresponde ao início de um novo frame. A mesma coisa ocorre para o término de uma sequência de bits, pois um caractere é considerado o último quando após o seu recebimento ocorre o tempo mínimo de espera.

O tempo de transmissão de uma palavra do quadro é de aproximadamente 573 μ s para taxas de comunicação acima de 19200 *bits/s* e 9 ms para 1200 *bits/s*. Caso o tempo entre os bytes for maior que o tempo mínimo estipulado, o frame será considerado inválido, fazendo com que todos os dados daquela mensagem sejam descartados. Neste protocolo, o CRC é utilizado para a verificação de erros.

Address	Function	Data	CRC Check
8 bits	8 bits	N x 8 bits	16 bits

Figura 12 - Frame do Modbus RTU. (Fonte: Fonte: Blog Embarcados, 2014)

2.5.2 Modbus TCP

Apesar de ter sido criado na década de 70, o protocolo Modbus vêm evoluindo com o tempo, e segue a tendência das Redes Industriais de utilizar conexões derivadas do protocolo Ethernet. A versão do Modbus TCP é praticamente o transporte da sua versão RTU para o padrão de comunicação Ethernet.

O Modbus TCP é baseado na pilha de protocolos do modelo TCP/IP, na qual faz uso do formato para a comunicação e adiciona ao frame Modbus um cabeçalho chamado de MBAP (Modbus Application Protocol). O modelo de comunicação utilizado é o de Cliente-Servidor, diferentemente do Modbus RTU, que utiliza o modelo Mestre-Escravo.

O cabeçalho MBAP possui o tamanho de 7 bytes, e é composto pelos seguintes campos:

- **Transaction Identifier:** Utilizado para a identificação da resposta para a transação, tamanho de 2 bytes;
- **Protocol Identifier:** Identifica o protocolo utilizado, sendo que o valor zero indica o uso do Modbus. Tamanho de 2 bytes;
- **Unit Identifier:** 1 byte definido pelo Cliente e ecoado pelo Servidor para identificação de um escravo remoto conectado em uma linha serial ou em outros barramentos.

O Modbus TCP utiliza o CRC-32 para a verificação de erros, pois se trata de um frame ethernet. Para que uma conexão TCP seja iniciada, deve existir comunicação entre um servidor e pelo menos um cliente, fazendo uso da porta 502, que é a padrão neste protocolo.

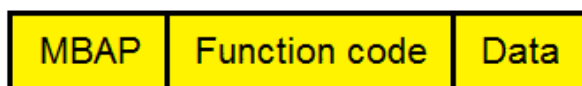


Figura 13 - Frame do Modbus TCP. (Fonte: Blog Embarcados, 2014)

Para exemplificar uma requisição de um cliente para um servidor, o equivalente ao frame Modbus RTU: 11 03 006B 0003 7687, é o frame:

0001 0000 0006 11 03 006B 0003

Onde:

- **0001:** Identificador de transação;
- **0000:** Identificador de protocolo;
- **0006:** Tamanho da mensagem;
- **11:** Identificador da Unidade (17 = 11 hex);
- **03:** Código da função (leitura de saída analógica);
- **006B:** Endereço do primeiro registro requisitado (40108 – 40001 = 107 = 6B hex);
- **0003:** O número total de registros requisitados (leitura de 3 registros: 40108 a 40110).

A figura 14 representa a construção genérica de um PDU (Protocol Data Unit) segundo o modelo OSI, que é utilizado pelo Modbus TCP:

CONSTRUCTION OF A TCP/IP-ETHERNET DATA PACKET

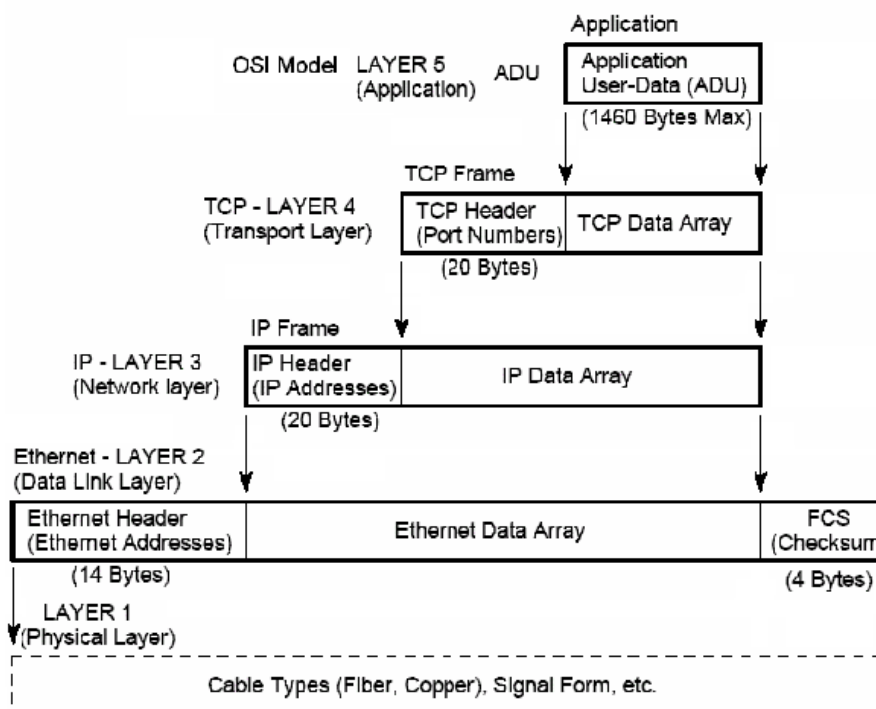


Figura 14 - Formato e construção de um frame na pilha TCP/IP. (Fonte: Simply Modbus, 2017)

Na figura 15 é possível comparar a estruturação dos diversos tipos de protocolo Modbus, com relação à pilha de protocolos do modelo OSI, no qual o protocolo Modbus TCP se modela.

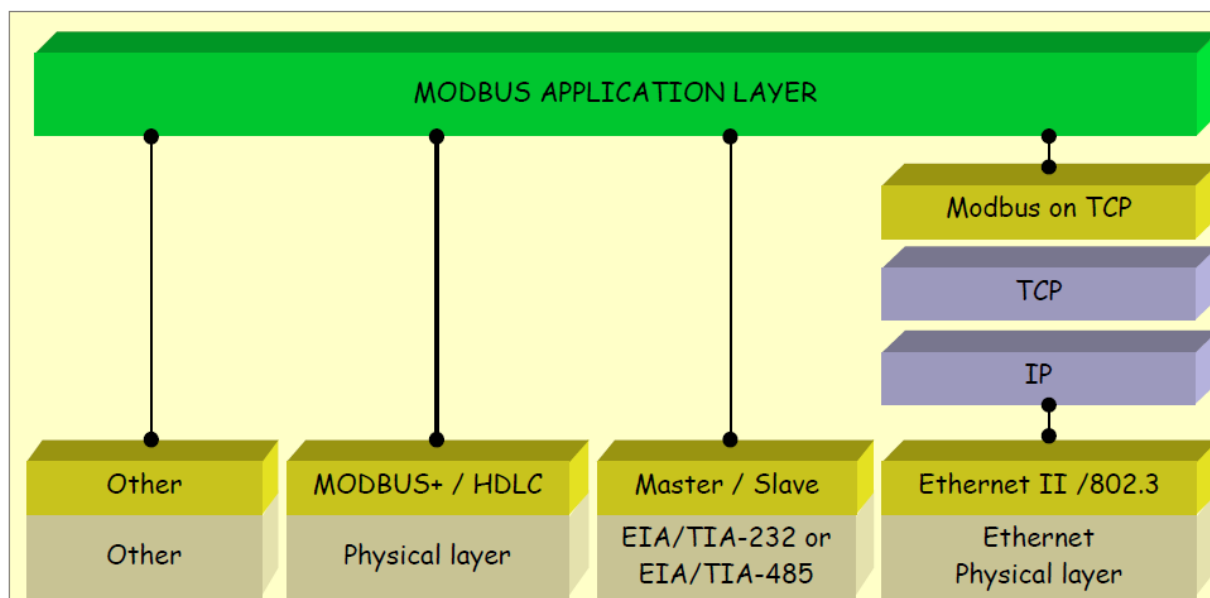


Figura 15 - Possibilidade de camadas com protocolos. (Fonte: Modbus Org, 2002)

A figura 16 mostra as diferenças estruturais do frame do Modbus RTU e do Modbus TCP:

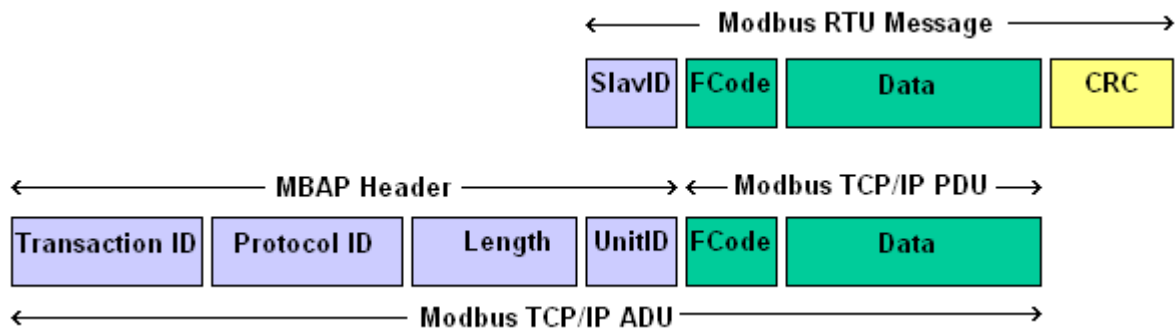


Figura 16 - Comparação entre frames do Modbus RTU e TCP. (Fonte: Simply Modbus, 2017)

3 DESENVOLVIMENTO DO PROJETO

3.1 Ferramentas Utilizadas

As principais ferramentas utilizadas para o desenvolvimento do projeto estão descritas abaixo:

3.1.1 Beaglebone Black

Beaglebone Black é uma plataforma com suporte de uma comunidade própria, para desenvolvimento de projetos baixo-custo. (Adaptado do blog Beagleboard, 2018)

A BeagleBone Black Rev C é um kit de desenvolvimento baseado no processador AM3358 que integra um ARM Cortex™-A8 core, rodando a 1GHz e disponibilidade de vários periféricos. Além disso a placa possui várias interfaces como Ethernet, USB, OTG, cartão TF, serial, JTAG (sem conector), HDMI tipo D< Emmc, ADC, I2C, SPI, PWM e LCD. Ela é capaz de realizar mais de 3 milhões de operações *Dhrystone* por segundo e operações aritméticas com vetores de ponto flutuante. Consegue ainda controlar drivers de motores para robótica, sensores de localização ou de pressão e câmeras em 2D ou 3D, e também executar *OpenCV*, *OpenNI* e outros softwares de coleta de e análise de imagens para reconhecer os objetos em torno de seu robô e os gestos que você pode fazer para controlá-lo. (Blog Filipeflop, 2014)

Dados Técnicos:

- Especificações do Processador:
 - Chip: AM335x 1GHz ARM® Cortex-A8
 - Memória: 512MB DDR3 RAM
 - Armazenamento: 4GB 8-bit eMMC on-board flash storage
 - 3D graphics accelerator
 - NEON floating-point accelerator
 - 2x PRU 32-bit microcontrollers

- Interface:
 - USB 2.0 Client, para alimentação e comunicação
 - USB 2.0 Host
 - 10/100M Ethernet (Conector RJ45)
 - Interface LCD
 - HDMI
 - Slot cartão TF
 - 2x 46 Pinos
- Compatibilidade de Softwares:
 - Debian
 - Android
 - Ubuntu
 - Cloud9 IDE
 - e mais...
- Especificações Gerais:
 - Tensão de operação: 5V/0,35A
 - Temperatura de operação: 0-70°C
 - Dimensões: 86,36 x 54,61mm

Para começar a utilizar a Beaglebone Black, é necessário o uso de um cabo com conector mini USB incluso. É por meio deste cabo que a comunicação com o computador é feita (e por consequência a alimentação). Na imagem abaixo, é possível verificar os principais componentes da placa:

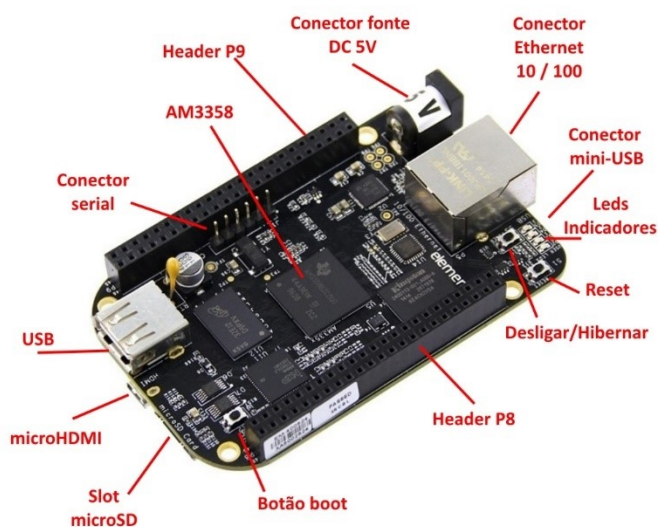


Figura 17 - Beaglebone Black. (Fonte: Blog FilipeFlop, 2017)

Os *Headers* na lateral da placa são as portas de entrada e saída (GPIO, 65 no total), que podem ser utilizadas para monitorar e controlar sensores, botões, módulos e outros dispositivos, de forma semelhante ao Raspberry e Arduino. A alimentação externa da BeagleBone é feita por uma fonte de no máximo 5V, com pelo menos 1A de corrente.

Com o slot microSD é possível carregar uma imagem de um dos sistemas operacionais disponíveis, e transferi-lo para a memória da BeagleBone. Assim, a placa poderá ser utilizada como um computador, ligando a placa à um monitor utilizando o conector microHDMI (resolução máxima de 1280 x 1024 pixels). A Beaglebone Black também permite a conexão de um teclado e mouse, mas com o uso de um *hub USB*.

Cape Expansion Headers

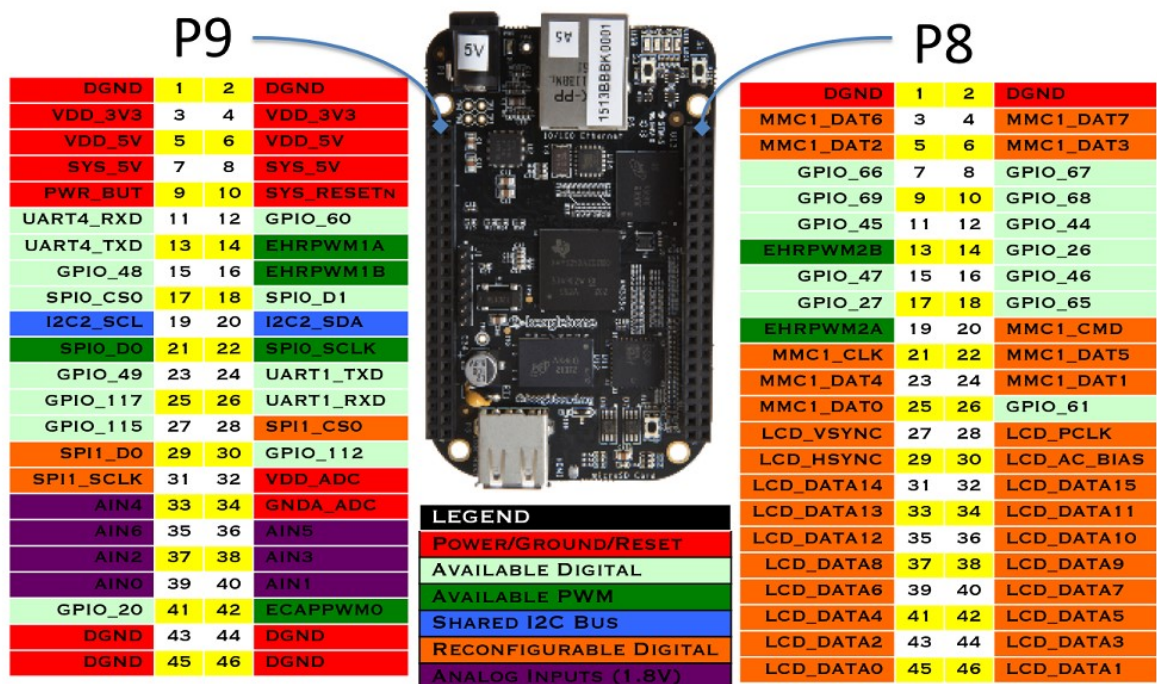


Figura 18 - Identificação e localização dos pinos da Beaglebone Black. (Fonte: Blog Vanderbilt Aerospace Design Lab, 2016)

3.1.2 FreeModbus

O FreeModbus é uma implementação gratuita do popular protocolo MODBUS em linguagem C, especialmente direcionado para sistemas embarcados. O Modbus é um protocolo de rede popular no ambiente de produção industrial. Uma pilha de comunicação Modbus requer duas camadas. O Protocolo de Aplicação Modbus que define o modelo de dados e funções e uma camada de Rede. Em sua versão atual, o FreeModbus fornece uma implementação do “Protocolo de Aplicação Modbus v1.1” e suporta os modos de transmissão RTU / ASCII definidos na especificação de linha serial Modbus 1.0. Desde a versão 0.7 FreeModbus também suporta MODBUS TCP definido no Modbus *Messaging* no TCP / IP *Implementation Guide* v1.0a. Está licenciado sob o BSD que permite seu uso em ambientes comerciais. As seguintes funções Modbus são atualmente suportadas:

- Ler *Input Registers* (0x04)
- Ler *Holding Registers* (0x03)
- Escrever *Single Register* (0x06)
- Gravar *Multiple Registers* (0x10)
- Ler / gravar *Multiple Registers* (0x17)
- Ler *Coil* (0x01)
- Escrever *Single Coil* (0x05)
- Escrever *Multiple Coils* (0x0F)
- Ler *Discrete Inputs* (0x02)
- Informar *ID* do escravo (0x11)

A implementação é baseada nos padrões mais recentes e deve ser totalmente compatível com o padrão. O recebimento e transmissão de quadros Modbus RTU / ASCII é implementado como uma máquina de estados (Threads) que é acionada por retornos de chamada da camada de abstração de hardware. Isso facilita a portabilidade para novas plataformas. Se um quadro estiver completo, ele será passado para a Camada de Aplicação Modbus, onde seu conteúdo é inspecionado. (Adaptado do blog SILA Embedded Solutions, 2018)

3.1.3 Modsim/Modscan

Modscan é uma aplicação paga, para o Sistema Operacional Windows, que funciona como um Mestre/Cliente Modbus. Este software permite acessar e mudar dados de Escravos/Servidores Modbus que estejam conectados.

Já o Modsim é uma aplicação paga, para o Sistema Operacional Windows, que funciona como um Escravo/Servidor Modbus. Este software permite responder requisições de Mestres/Clientes Modbus que estejam conectados.

3.1.4 AWS Cloud9 IDE

“O AWS Cloud9 é um ambiente de desenvolvimento integrado (IDE) que permite escrever, executar e depurar código usando apenas um navegador. O ambiente inclui um editor de código, um depurador e um terminal. O Cloud9 é fornecido com ferramentas essenciais para linguagens de programação comuns, incluindo JavaScript, Python e PHP, entre outras, para que não precise instalar arquivos ou configurar a máquina de desenvolvimento para iniciar novos projetos. Como o Cloud9 IDE é baseado na nuvem, pode-se trabalhar nos projetos no escritório, em casa ou em qualquer lugar usando uma máquina conectada à Internet. Além disso, o Cloud9 oferece uma experiência transparente para desenvolvimento de aplicativo sem servidor. Dessa forma, é possível definir recursos, depurar e alternar entre execução local e remota de aplicativos sem servidor com facilidade. Com o Cloud9, pode-se compartilhar rapidamente o ambiente de desenvolvimento com uma equipe, o que possibilita a programação em pares e o controle mútuo das entradas em tempo real.” (AWS, 2017)

Para o desenvolvimento de projetos na Beaglebone Black, o Cloud9 se torna uma ótima opção, pois suporta várias linguagens de programação, incluindo C, C ++, PHP, Ruby, Perl, Python, JavaScript com Node.js e Go.

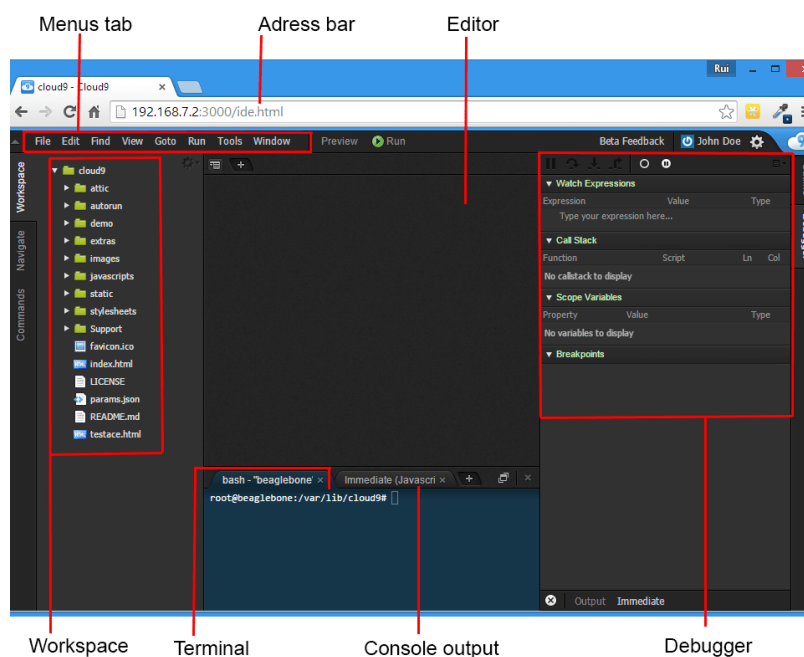


Figura 19 - IDE Cloud9 (Fonte: Beaglebone For Dummies, 2018)

3.1.5 Node-Red

“O *Node-Red* consiste em uma ferramenta bastante versátil, voltada principalmente para o desenvolvimento de aplicações relacionadas ao conceito de *Internet de Coisas (IoT)*. Este recurso utiliza uma abordagem de *programação gráfica*, ou seja, através do mesmo, torna-se possível elaborar uma aplicação por meio do estabelecimento de conexões entre blocos que possuem códigos predefinidos (conhecidos como *nós*) para a realização de determinadas tarefas.” (Blog FilipeFlop, 2017)

“O Node-Red tem como função primordial auxiliar na elaboração de aplicações propondo uma maneira mais simples para o usuário que não domina muito as ferramentas de programação e também não possui conhecimento suficiente sobre estruturas utilizadas no desenvolvimento de sistemas embarcados, como um *Arduino UNO* ou um *Raspberry PI*. Entretanto, também é possível desenvolver uma vasta gama de aplicações avançadas, de maneira rápida e fácil.” (Blog FilipeFlop, 2017)

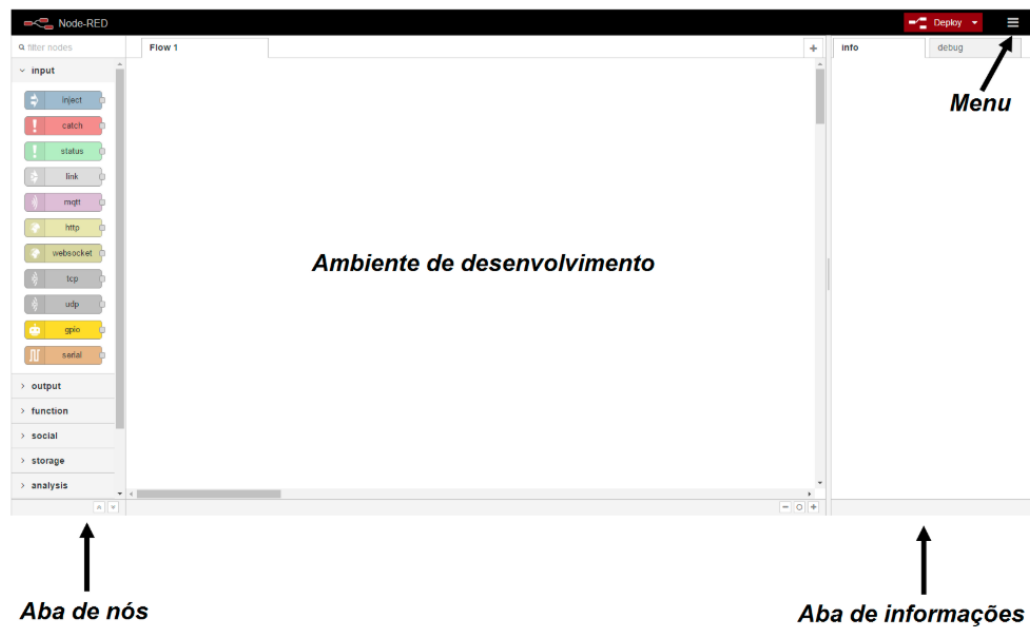


Figura 20 - IDE Node-Red. (Fonte: Blog FilipeFlop, 2018)

No Node-Red existem três tipos principais de nós, eles são:

- **Nós de Entrada:** Permitem a inserção de dados em uma aplicação.
- **Nós de Processamento:** Permitem a manipulação dos dados inseridos para disponibilização no nó de saída.
- **Nós de Saída:** Permitem o envio de dados para outra aplicação.

Quando estes nós estão conectados, eles geram uma estrutura denominada fluxo.

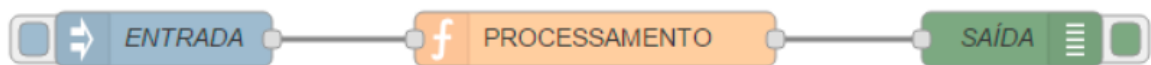


Figura 21 - Exemplo de fluxo básico no Node-Red (Fonte: Blog FilipeFlop, 2018)

3.2 Etapas

No início do projeto, a intenção foi de construir algo relacionado ao IoT, Indústria 4.0 e Domótica, fazendo uso de ferramentas *open source*. Após uma extensa pesquisa sobre ferramentas e protocolos comunicação, a que se mostrou mais interessante foi o Protocolo Modbus TCP, pois apesar de ser um protocolo pioneiro e bastante empregado na indústria, não chega a ser o mais empregado em IoT. Aproveitando que o protocolo havia se tornado livre, a ideia de utilizar um protocolo famoso e antigo para novas tecnologias pareceu intrigante e desafiador, além de ser uma oportunidade de trabalhar para mostrar novas alternativas para a comunicação entre dispositivos.

A partir disso, este projeto tem o objetivo de implementar o protocolo Modbus baseado no FreeModbus, para demonstrar uma alternativa de comunicação simples e eficaz para este cenário IoT que vivemos.

Para a construção do protótipo, a estratégia adotada foi a divisão em etapas, visando o aprendizado do autor e a organização do trabalho. Com isso, para a construção de um sistema de controle de luminosidade como aplicação de uso do protocolo Modbus, as etapas são descritas nos próximos tópicos.

3.2.1 Comunicação entre Modscan e Modsim

No início do projeto, para que o autor obtivesse um conhecimento mais profundo sobre o protocolo, foi necessária uma revisão bibliográfica nas documentações relacionadas ao Modbus TCP. E para que fosse possível aplicar o conhecimento obtido de forma prática, o primeiro passo foi realizar a comunicação entre um Servidor e um Cliente, simulando a troca de frames para os quatro tipos de armazenamento de dados: Coil Status (leitura e escrita), Input Coil (leitura), Holding Register (leitura e escrita) e Input Register (leitura).

Test Plan: Realizar a comunicação entre o Modscan e Modsim, abrangendo os quatro tipos de armazenamento de dados, e utilizando o protocolo Modbus TCP.

Test Case: Para iniciar o processo de comunicação, foi necessário realizar algumas configurações para que ocorresse a comunicação entre o Cliente (Modscan) e o Servidor (Modsim). O primeiro passo foi escolher o tipo de conexão como “Remote TCP/IP server” no Modscan e “Modbus TCP Svr” no Modsim. Com isso, a próxima etapa foi configurar o IP do servidor como sendo o IP de *loopback* (127.0.0.1), além da porta padrão 502.

Após realizar a conexão entre o Modscan e Modsim, foi a vez de testar a comunicação entre os programas, para entender melhor na prática a diretriz do protocolo de comunicação estudado. O teste foi feito utilizando dados do tipo Holding Register, no qual os valores escritos no servidor são lidos pelo cliente.

3.2.2 Comunicação entre Beaglebone e Modscan

A próxima etapa envolve a implementação do Freemodbus na Beaglebone Black, por meio da IDE do Cloud9. A Beaglebone Black funciona como um servidor para responder e executar ações requisitadas por clientes.

Test Plan: Portar o Freemodbus para a Beaglebone Black para que funcione como um servidor Modbus e realizar a conexão com o Modscan.

Test Case: Para iniciar a tarefa, o primeiro passo foi a instalação do Sistema Operacional Debian na versão 9.2, utilizando um cartão de memória micro SD. O sistema operacional pode ser encontrado no site: <https://beagleboard.org/latest-images>. Se o sistema for rodar diretamente no cartão de memória, para iniciá-lo, é necessário pressionar o botão próximo ao slot de cartão de memória antes de realizar a alimentação da Beaglebone Black, e só soltar após os “User Leds” se apagarem.

Com o Debian rodando na Beaglebone Black, foi necessário encontrar uma interface de desenvolvimento de software compatível com a linguagem C e suportada pelo sistema embarcado. A solução encontrada foi utilizar a IDE Cloud9, que proporciona suporte à programação em linguagem C, e funciona diretamente no navegador (é recomendável utilizar o navegador Google Chrome). A opção utilizada do Freemodbus foi a “LINUXTCP”, que é uma das 19 opções de portabilidade oferecidas.

Com a Beaglebone Black conectada ao computador, para utilizar o Cloud9 é preciso digitar o IP configurado na beaglebone (normalmente é 192.168.7.2) e acessar a porta 3000. Ou seja, o endereço que precisa ser inserido na barra de endereço do navegador é este: **192.168.7.2:3000**.

Como o Freemodbus possui a opção de uso dos protocolos Modbus ASCII, RTU e TCP, no arquivo, foi necessário alterar o código do arquivo para habilitar o funcionamento do Modbus TCP, no arquivo chamado “mbconfig.h”.

Após a escolha do protocolo, foi necessário revisar e estudar o código, além de corrigir erros identificados no processo de compilação. Como o Freemodbus utiliza vários arquivos, como os do tipo “.h” e “.c” um makefile foi usado e os arquivos principais foram organizados em pastas, para executar todos os que são necessários ao mesmo tempo.

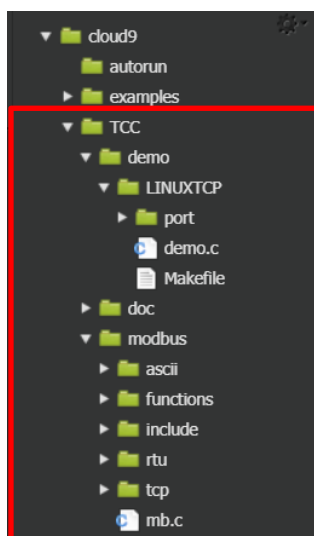


Figura 23 - Organização das pastas no Cloud9. (Fonte: Autor)

Para inserir os arquivos do Freemodbus no IDE do Cloud9 é só “arrastá-los” da sua pasta de origem para o workspace, situado no lado esquerdo da IDE. As pastas devem estar organizadas como na figura 23 para que fiquem de acordo com a programação feita no makefile, que está descrito no Anexo I, ao final deste trabalho.

Com o código organizado e revisado, para executá-lo foi necessário utilizar os comandos “make” e “./tcpmodbus”. É importante notar que para a execução do código, utilizando a porta 502, é preciso realizar o login como usuário root.

No caso de o comando “make” já ter sido executado mais de uma vez, é necessário utilizar o comando “make clean” antes, para limpar a construção do projeto anterior.

Após a execução do comando “./tcpmodbus” (nome do executável escolhido no makefile), o terminal lista opções por meio de uma “função switch” para a utilização da aplicação, onde deverá ser escolhido alguma opção, digitando a letra correspondente e apertando a tecla enter. Para habilitar a pilha de protocolos do Modbus TCP, basta escolher a opção “e”. Assim, o servidor estará apto a se conectar aos clientes.

Com a pilha do protocolo Modbus TCP em funcionamento, o próximo passo é então conectar um cliente para testar o funcionamento do servidor. Para isso, foi utilizado o software Modscan, que foi configurado com o IP do servidor (192.168.7.2), e a porta 502.

Para que o Modscan se conecte ao servidor sem exibir erros, é necessário utilizar o endereço de memória condizente com a aplicação, configurado no cabeçalho do código principal do Freemodbus. Como é mostrado na figura 24, as variáveis “REG_COILS_START”, “REG_DISC_START”, “REG_HOLDING_START” e “REG_INPUT_START” representam o valor inicial do endereço de memória de cada aplicação.

```
#define REG_COILS_START 1
#define REG_DISC_START 10001
#define REG_HOLDING_START 30001
#define REG_INPUT_START 40001

#define REG_COIL_NREGS 8
#define REG_DISC_NREGS 8
#define REG_INPUT_NREGS 10
#define REG_HOLDING_NREGS 130
```

Figura 24 - Definição do endereço e quantidade de unidades de memória. (Fonte: Autor)

Já as variáveis “REG_COIL_REGS”, “REG_DISC_NREGS”, “REG_INPUT_NREGS” e “REG_HOLDING_NREGS” representam o tamanho ou quantidade de unidades de memória para cada tipo de aplicação. Ou seja, é possível escrever ou ler 8 coils.

Foram feitos testes semelhantes com os outros três tipos de endereçamento, e os resultados obtidos foram satisfatórios.

3.2.3 Elaboração de uma Aplicação

Após a finalização do objetivo principal do trabalho, que é implementação do Freemodbus na Beaglebone Black e comunicação com um cliente, a próxima etapa tem a intenção de utilizar a ferramenta em uma aplicação prática, como uma base para implementação de outras aplicações para Indústria 4.0 e Domótica. Neste caso, o objetivo é criar um sistema de controle de iluminação.

Test Plan: Como exemplo de aplicação, desenvolver um sistema de controle de iluminação que meça a luminosidade do ambiente e que desligue ou ligue uma Lâmpada de acordo com a necessidade de iluminação.

Test Case: Neste caso para desenvolver a aplicação, será necessário o uso de um LDR, LED (para simular uma lâmpada) e resistências para limitar a potência nas portas GPIO da Beaglebone, LED e LDR. O projeto desenvolvido para a funcionamento do sensor de luz está representado abaixo e foi adaptado do blog Mobilis, da Universidade Federal de Ouro Preto.

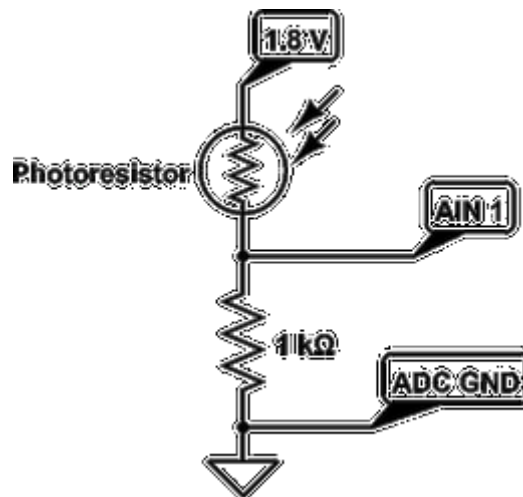


Figura 25 - Esquema de montagem do circuito. (Fonte: Blog do Laboratório Mobilis - UFOP, 2014)

A tensão de 1,8V é obtida diretamente no pino 9_32 (VDD_ADC) da BBB. A saída o sensor LDR é ligada ao resistor de $1K\Omega$ e derivada para a entrada analógica da BBB no pino 9_40 (AIN1). O outro terminal do resistor é conectado ao terra (GND) pelo no pino 9_34 (GNDA_ADC). Este pino se comportará como uma entrada analógica.

O esquema para montagem do circuito na Beaglebone Black pode ser verificado na figura 26:

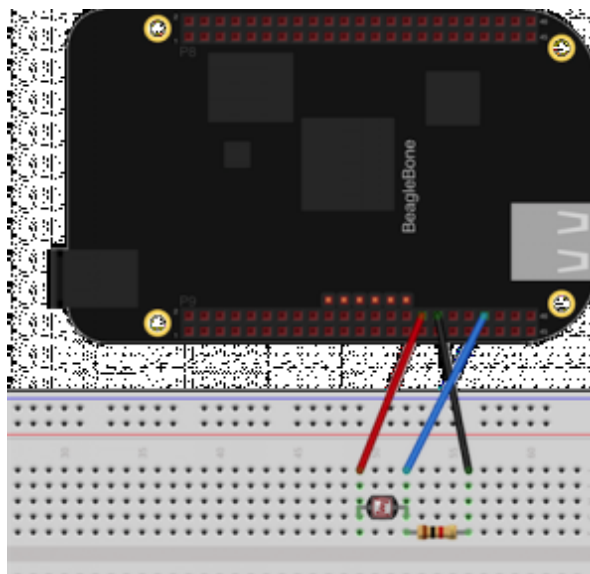


Figura 26 - Esquema de montagem do circuito na Beaglebone Black. (Fonte: Blog do Laboratório Mobilis - UFOP, 2014)

Para realizar montagem do circuito acima, é importante se atentar a algumas características de hardware do conversor ADC da BBB:

- Tamanho do valor convertido: *12 bits* (valores de saída na faixa de 0-4095)
- Tempo da amostragem: *125 ns*
- Intervalo de tensão de entrada: de *0* a *1,8V* (Importante não ultrapassar!)
- A corrente que flui para o pino ADC nessa faixa de operação é de *2uA*.
- Uma vez que se realizam medições em *mV*, com tolerância de erro de *0,1%* nos resistores, deve ser utilizado um divisor de tensão.
- Se estiver usando um divisor de tensão, o resistor conectado ao terra deve ser menor ou igual a *1 KΩ*. Esse circuito de divisor de tensão será utilizado como exemplo desse tutorial.
- A BBB fornece tensão de referência de *1,8V* na porta *VDD_ADC* – pino *9_32* e terra (GND) na porta *GNDA_ADC* – pino *9_34*.

A segunda parte da montagem do circuito, é a inserção de um LED para simular o funcionamento de uma lâmpada (saída digital). Este LED foi conectado à porta GPIO número 67, ou *gpio67*, e configurado como uma saída. Ou seja, de acordo com a lógica do projeto, quando for necessário utilizar a luz de uma lâmpada, a Beaglebone Black disponibilizará uma tensão de *1,8 V* para ligar o LED com um resistor de *330 KΩ* em série. O circuito completo pode ser visualizado na figura 27.

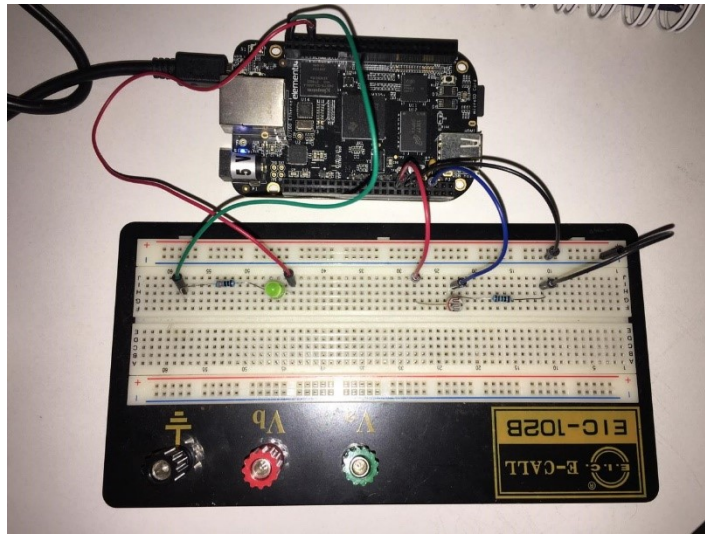


Figura 27 - Circuito completo montado. (Fonte: Autor)

Vale salientar que este circuito foi montado para efeitos de teste, sendo que para uma aplicação real, o sensor de luz não poderia estar no mesmo ambiente em que a luz artificial controlada pelo algoritmo.

Na montagem da lógica de programação desta parte do projeto, foram consideradas as seguintes variáveis, no qual a relação com os “vetores Buffer” podem ser verificadas na tabela 3:

Nome	Tipo	Índice no Buffer
aux0	Holding Register	usRegHoldingBuf[0]
aux1	Coil	ucRegCoilBuf[0]
Value	Input Register	usRegInputBuf[0]

Tabela 3 - Principais variáveis do algoritmo. (Fonte: Autor)

Na lógica do algoritmo criado para a aplicação, as variáveis acima têm a função descrita na tabela abaixo:

Nome	Função
aux0	Receber valor (0, 1 ou 2) para indicação de modo automático, lâmpada ligada no modo manual ou desligada no modo manual

aux1	Receber valor (0 ou 1) para indicação de acionamento da lâmpada
value	Receber o valor proveniente da entrada ADC conectada ao sensor LDR

Tabela 4 - Função das principais variáveis do algoritmo. (Fonte: Autor)

Baseado nisso, primeiramente o usuário deverá escolher se o sistema operará em modo manual ou automático. Se optar pelo modo manual, deverá acionar definir se a lâmpada estará ligada ou desligada de acordo com a sua vontade. Já no modo automático, a aplicação definirá se a lâmpada estará ligada ou não, de acordo com a luminosidade do ambiente.

O range escolhido para que a lâmpada acenda no modo automático é de 0 a 80% de luminosidade, detectada pelo LDR. A fórmula para chegar neste percentual, está descrita abaixo:

$$\text{Percentual} = \frac{\text{Valor Real} - \text{Valor mínimo}}{\text{Valor máximo} - \text{Valor mínimo}} \times 100$$

No qual, o valor mínimo e máximo encontrados no conversor AD da Beaglebone Black foram 3891 e 3977, respectivamente. Sendo que o este valor representa a tensão encontrada no resistor divisor de tensão do circuito, pois pelo princípio de funcionamento do LDR, quanto maior a luminosidade, menor a resistência do componente, o que faz com que a queda de tensão no resistor seja maior.

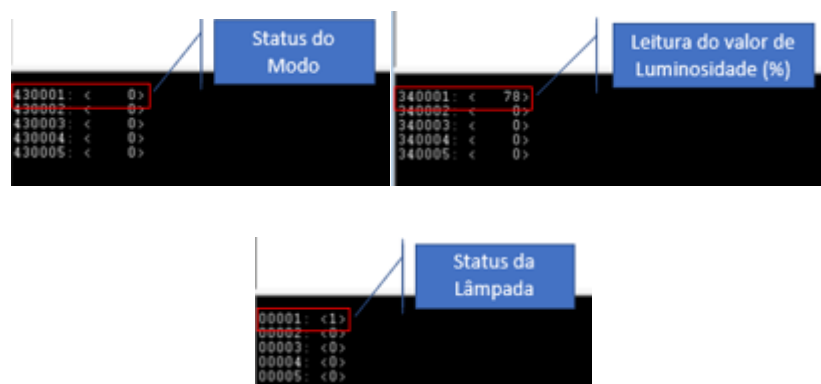


Figura 28 - Resultado de Testes da Aplicação. (Fonte: Autor)

Os testes realizados com o circuito foram bem sucedidos, e o mesmo modelo de projeto poderá ser utilizado como base na implementação de sistemas de detecção de umidade, no caso de automatização de irrigação em residências ou agricultura, monitoramento de corrente para manutenção preditiva ou em projetos de otimização de consumo, entre outros, “abrindo um leque de opções” para a criação de novos projetos relacionados a *IoT*.

3.2.4 Elaboração de Sistema Supervisório Utilizando o Node-Red

Na criação de um sistema de comunicação entre cliente e servidor, é necessário a utilização de um sistema supervisório para criar uma interface mais amigável com o usuário. A interface homem-máquina é basicamente utilizada para requisitar e fornecer dados por meio do usuário para que a aplicação execute o seu papel devido.

Para executar este papel, foi escolhido um software open source de fácil manipulação e comunicação com diversas plataformas. Este software é o Node-Red.

Test Plan: Criar um sistema supervisório para atuar como uma interface mais amigável entre o usuário e o servidor, para que as funções elaboradas no algoritmo relacionadas ao cliente possam ser operadas pelo usuário da ferramenta.

Test Case: O desenvolvimento da aplicação foi feito na plataforma Node-Red, utilizando um computador como referência de dispositivo para interação com o usuário. Para a criação do supervisório, foram utilizados os pacotes de desenvolvimento de interface gráfica, chamado de dashboard e para a realização da comunicação entre o supervisório e o servidor, foram utilizados blocos de comunicação proveniente de um pacote Modbus encontrado no site de suporte do Node-Red.

O início da construção do algoritmo começou pela interface gráfica, no qual foram utilizados um bloco do tipo “dropdown” para a escolha do modo de operação (automático ou manual), um bloco do tipo “button” para a escolha do status de funcionamento da lâmpada em modo manual (ligado ou desligado), e por fim um bloco do tipo “gauge” para realizar a leitura do percentual de luminosidade do ambiente.



Figura 29 - Nós utilizados para a interface gráfica. (Fonte: Autor)

Estes objetos foram criados utilizando o mesmo grupo e “tab”, os quais podem ser visualizados na figura 30:

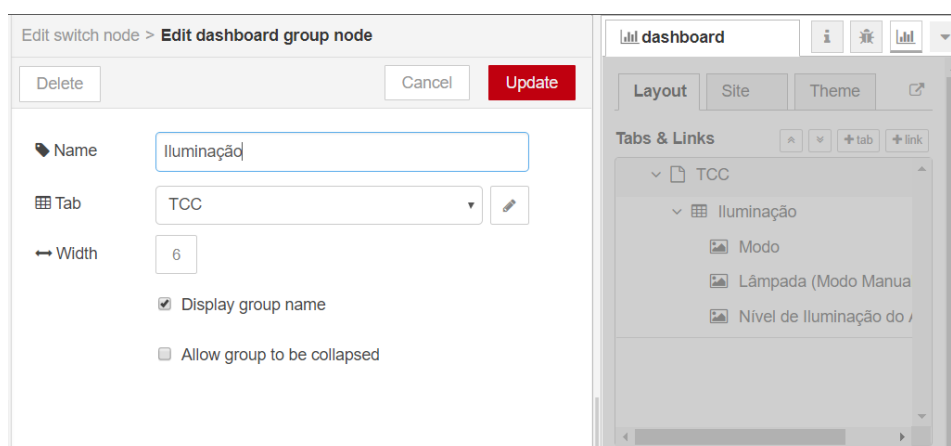


Figura 30 - Configuração do nó de interface gráfica. (Fonte: Autor)

Após a finalização da implementação dos objetos gráficos do supervisor, foi a vez de inserir os blocos relacionados à comunicação Modbus com o servidor em funcionamento na Beaglebone Black. Para isso foram utilizados 2 blocos, nos quais um para a escrita e outro para leitura.

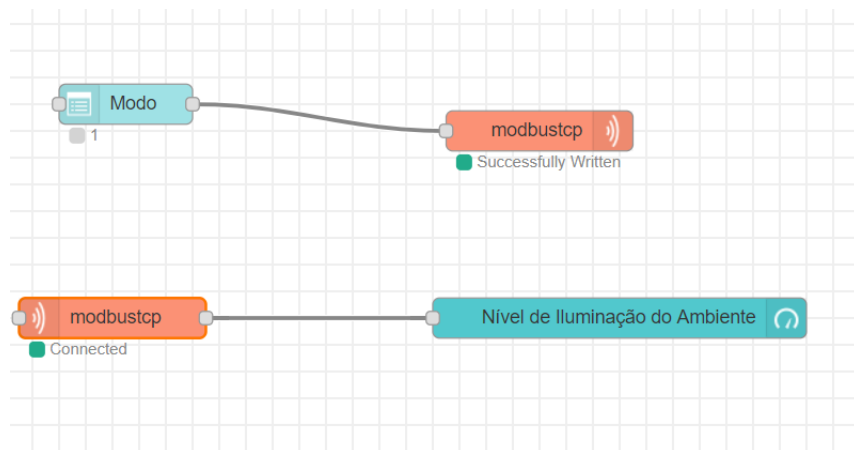


Figura 31 - Projeto utilizando blocos Modbus. (Fonte: Autor)

Para a realização das ações de leitura e escrita, é preciso associar cada bloco Modbus a um servidor, endereço de memória e código de função para a execução da tarefa, como mostrado na figura 32:

Figura 32 - Configuração do nó de função. (Fonte: Autor)

Após clicar no botão “deploy”, ao acessar o endereço “localhost:1880/ui/”, o resultado obtido com a junção dos nós pode ser visualizado na figura 33:

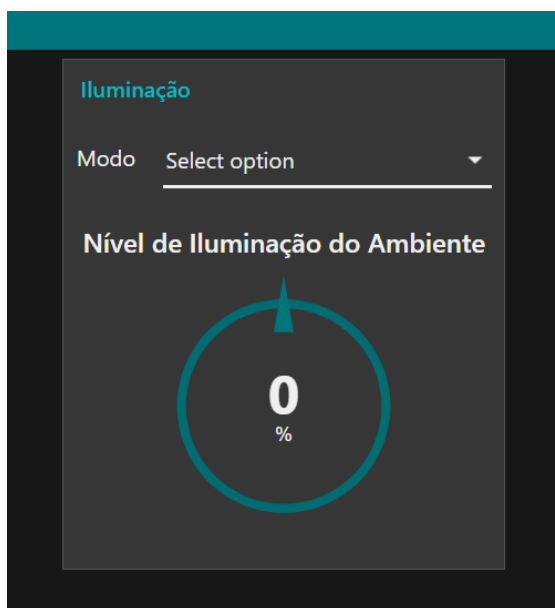


Figura 33 - Status do Supervisório Desenvolvido. (Fonte: Autor)

Durante os testes, foi constatado que o bloco de escrita funcionou normalmente, mas o bloco de leitura não obteve um desempenho satisfatório, onde o erro não pôde ser encontrado há tempo de realizar a apresentação deste trabalho.

4 CONCLUSÃO

Desde os primórdios o homem está em busca de otimização de processos, e como toda certeza podemos identificar esta tendência olhando para a história humana. Com o passar do tempo, as atividades manuais estão sendo automatizadas, e ser humano de acordo com a sua evolução intelectual foca cada vez mais em trabalhos nos quais o raciocínio lógico é primordial. Todo este cenário está sendo refletido em ondas tecnológicas cada vez mais significativas, como é o caso do *IoT* nos tempos atuais. Esta onda se reflete em vários setores como é o caso da Domótica, e a Indústria 4.0.

A Domótica, após um período de desenvolvimento lento, agora tem um novo combustível. Com o advento da democratização e o avanço da tecnologia, este tipo de conceito se torna extremamente viável para a maioria da população. Em um futuro próximo, a tendência é de que as pessoas deixem de trabalhar para manter suas casas, para que as próprias casas trabalhem para os seus proprietários.

O *IoT* também serviu de combustível para a explosão da Indústria 4.0. As indústrias são locais onde a otimização de tempo e recursos são essenciais, o lucro e o bem-estar dos funcionários movem as melhorias na maioria das empresas de sucesso. É aí que entra o interesse pela automação. Com o uso de robôs e sensores, a maioria das pessoas que executam atividades de operação manual podem ser capacitadas para ocuparem posições mais nobres como as de receitas de linhas de produção, ou técnicos de instrumentação, que vão agregar muito mais valor para o negócio e aumentar a eficiência e produtividade da indústria.

O projeto desenvolvido neste trabalho de conclusão de curso vem com a intenção de criar uma base versátil, dinâmica e de baixo custo para que sejam construídos projetos ligados ao *IoT*, com foco na indústria e automação residencial. Para que isso ocorra, foram utilizadas ferramentas (softwares) *open source* e equipamentos de baixo custo, ligados a comunidades que tem o objetivo de compartilhar experiências e desenvolver novas ideias e tecnologias.

Tecnicamente dizendo, o ponto alto deste projeto é a implementação do protocolo de comunicação Modbus TCP que é amplamente utilizado na indústria, mas

não é um dos mais “queridos” nas plataformas atuais. Após a compra dos direitos do Modbus pela Schneider, este se tornou um protocolo livre, ou seja, não é necessário obter uma licença para utilizá-lo. Com base nisso, a versão do Modbus adaptado a pilha de protocolos TCP/IP se torna uma opção viável para o uso como protocolo de comunicação de projetos de *IoT*. O seu dinamismo e versatilidade o tornam uma boa opção de uso para projetos desse tipo, além de ser amplamente utilizado ainda hoje na indústria, sendo suportado nos mais diversos equipamentos em uso, como em *PLCs*.

Com a utilização de ferramentas *open source*, e a implementação do Freemodbus em um sistema embarcado popular, como a Beaglebone Black, os objetivos deste trabalho de conclusão de curso são alcançados, formando uma base para a elaboração de novos projetos com foco em “Internet das Coisas” para as mais diversas aplicações.

4 REFERÊNCIAS

MODBUS ORG. **Modbus Application Protocol Specification V1.1**. Estados Unidos: Modbus Organization, 2002, 52p.

SUNLUX. **Modbus Slave Source Code Library**. Índia: SUNLUX TECHNOLOGIES LTD, 2006, 40p.

MODICON. **Modbus Protocol Reference Guide**. Estados Unidos: MODICON, 1996, 121p.

FOROUZAN, B. A. **Comunicação de Dados e Redes de Computadores**. Tradução de Glayson Eduardo de Figueiredo. 3º. ed. Porto Alegre: Bookman, 2006. 840 p.

KUROSE, J. F. E. K. W. R. **Redes de computadores e a Internet: uma abordagem top-down**. Tradução de Opportunity translations. 5º. ed. São Paulo: Addison Wesley, 2010. 615 p.

ACCARDI, A. DODONOV, E. **Automação Residencial: Elementos Básicos, Arquiteturas, Setores, Aplicações e Protocolos**. São Carlos: Tecnologia, Infraestrutura e Software, 2012. Disponível em: < <http://professor.pucgoias.edu.br/SiteDocente/admin/arquivosUpload/17829/material/ARTIGO02.pdf>>. Acesso em: 02 out. 2017.

WALTER, C. **Freemodbus**. Boston, Estados Unidos: Free Software Foundation, INC, 2006. Disponível em: < <https://sourceforge.net/projects/freemodbus.berlios/>>. Acesso em: 02 out. 2017.

PINHEIRO, J.M.S. **A Domótica**. São Paulo/SP, 2005. Disponível em: < https://www.projetoederedes.com.br/artigos/artigo_domotica.php>. Acesso em: 02 out. 2017.

SGARBI, J.A., TONIDANDEL, F. **Domótica Inteligente: Automação Residencial baseada em Comportamento**. São Bernardo do Campo/SP, 2005. Disponível em: < https://fei.edu.br/~flaviot/pub_arquivos/WTDIA06.pdf>. Acesso em: 02 out. 2017.

DEVTECNOLOGIA. **IOT: Internet das Coisas**. São Paulo/SP, 2017. Disponível em: < <http://devtecnologia.com.br/internet-das-coisas-iot/>>. Acesso em: 04 out. 2017.

CISCO, EVANS, D. **A Internet das Coisas**. California, Estados Unidos, 2011. Disponível em: https://www.cisco.com/c/dam/global/pt_br/assets/executives/pdf/internet_of_things_iot_ibsg_0411final.pdf>. Acesso em: 04 out. 2017.

WIKIPEDIA, **Internet das Coisas**. Disponível em: < https://pt.wikipedia.org/wiki/Internet_das_coisas>. Acesso em: 06 out. 2017.

SMART CITY. **3 Exemplos de Smart Cities que Você Precisa Conhecer.** Disponível em: < <http://smartcitylaguna.com.br/3-smart-cities-voce-precisa-conhecer/>>. Acesso em: 06 out. 2017.

PROOF. **Internet das Coisas: Um Desenho do Futuro.** Disponível em: < <http://www.proof.com.br/blog/internet-das-coisas/>>. Acesso em: 06 out. 2017.

SAS. **O que é Big Data?.** Disponível em: https://www.sas.com/pt_br/insights/big-data/what-is-big-data.html#>. Acesso em: 06 out. 2017.

SIGNIFICADOS. **Significado de Internet das Coias.** Disponível em: <<https://www.significados.com.br/internet-das-coisas/>>. Acesso em: 06 out. 2017.

MANTELATO, R. **IOT – Internet das Coisas.** 2013. Disponível em: < <http://dextra.com.br/pt/blog/iot-internet-das-coisas/>>. Acesso em: 06 out. 2017.

SIMPLY MODBUS. **Frequently Asked Questions.** Disponível em: <http://www.simplymodbus.ca/FAQ.htm#Modbus>>. Acesso em: 08 out. 2017.

CITYSYSTEMS. SILVEIRA, CB. **MODBUS.** Disponível em: < <https://www.citisystems.com.br/modbus/>>. Acesso em: 08 out. 2017.

RTA. **MODBUS RTU.** Disponível em: < <https://www.rtaautomation.com/technologies/modbus-rtu/>>. Acesso em: 08 out. 2017.

EMBARCADOS. FREITAS, C.M. **Protocolo Modbus.** 2014. Disponível em: <https://www.embarcados.com.br/protocolo-modbus/>>. Acesso em: 08 out. 2017.

PINHEIRO, J.M.S. **Domótica.** São Paulo/SP, 2005. Disponível em: < https://www.projetoederedes.com.br/artigos/artigo_domotica.php>. Acesso em: 02 out. 2017.

SIMPLY MODBUS. **Modbus TCP/IP.** Disponível em: < <http://www.simplymodbus.ca/TCP.htm> >. Acesso em: 08 out. 2017.

ALGAR TELECOM. **Internet das Coisas.** Disponível em: <https://blog.algartelem.com.br/tendencias/internet-das-coisas-voce-sabe-o-que-significa/>>. Acesso em: 08 out. 2017.

CASADOMO. **Domótica - Introducción.** 2010. Disponível em: <www.casadomo.com/>. Acesso em: 08 out. 2017.

YOUTUBE. PROSOFT TECHNOLOGY. **Understanding Modbus Serian and TCP/IP.** Disponível em: < <https://www.youtube.com/watch?v=k993tAFRLSE> >. Acesso em: 10 out. 2017.

ELINUX. **Beaglebord: C/C++ Programming.** Disponível em: https://elinux.org/Beagleboard:C/C%2B%2B_Programming>. Acesso em: 14 out. 2017.

YOUTUBE. BRINCANDO COM IDEIAS. **Instalação Local do Node-Red no Windows.** Disponível em: < <https://www.youtube.com/watch?v=aSoaCjGGcB8>>. Acesso em: 10 nov. 2018.

YOUTUBE. FREE WAVE TECHNOLOGIES. **Node-Red Modbus.** Disponível em: <<https://www.youtube.com/watch?v=47Lq2WLkfyE>>. Acesso em: 10 nov. 2018.

INFORMÁTICA EDUCATIVA. **Aula Internet das Coisas.** Disponível em: <<http://informaticaeducativa2ano.blogspot.com/>>. Acesso em: 02 nov. 2018.

NODERED. **Node-Red Modbus TCP Nodes for Communicating with a Modbus TCP Server.** Disponível em: <<https://flows.nodered.org/node/node-red-contrib-modbus-tcp>>. Acesso em: 02 nov. 2018.

TRANSFORMAÇÃO DIGITAL. **Industria 4.0: A Quarta Revolução Industrial.** Disponível em: < <https://transformacaodigital.com/industria-4-0/>>. Acesso em: 07 nov. 2018.

PRODWIN TECNOLOGIAS. **O Que é a Industria 4.0 e Como Se Preparar.** Disponível em: < <https://www.prodwin.com.br/pt/blog/o-que-e-a-industria-4-0/>>. Acesso em: 16 nov. 2018.

IT LAW WIKI. **Internet of Services.** Disponível em: <http://itlaw.wikia.com/wiki/Internet_of_Services>. Acesso em: 02 nov. 2018.

CIO. **Seis Princípios Básicos para a Indústria 4.0 para os CIOs.** Disponível em: < <https://cio.com.br/seis-principios-basicos-da-industria-4-0-para-os-cios/>>. Acesso em: 02 nov. 2018.

NCSYSTEMS. **Coleta de Dados.** Disponível em: < <http://www.ncsystems.com.br/coleta-de-dados> >. Acesso em: 04 nov. 2018.

CIMM. **Definição: O que é Manutenção Preditiva?** Disponível em: < <https://www.cimm.com.br/portal/verbetes/exibir/595-manutencao-preditiva> >. Acesso em: 08 nov. 2018.

COSTA, A.H. MARUYAMA, M. H. NETO, R.R.I., **Manutenção Preditiva.** Disponível em: < http://www.feb.unesp.br/jcandido/manutencao/Grupo_8.pdf >. Acesso em: 21 nov. 2018.

VICTORINO, M. **Rede Industrial Modbus Uma das Mais Antigas de Todas as Redes (Fieldbus).** Disponível em: < <http://blog.murrelektronik.com.br/protocolos-de-rede-conheca-os-modbus/> >. Acesso em: 12 nov. 2018.

BEAGLEBOARD. **What is Beaglebone Black?**. Disponível em: <https://beagleboard.org/black> >. Acesso em: 15 nov. 2018.

FILIFELOP. **Beaglebone Black Rev. C**. Disponível em: <https://www.filipeflop.com/produto/beaglebone-black-rev-c/#tab-blogrelacionados> >. Acesso em: 17 nov. 2018.

VANDERBILT AEROSPACE DESIGN LAB. **Setting Up The Beaglebone Black's GPIO Pins**. Disponível em: <https://vadt.github.io/beagleboneblack/2016/07/29/setting-up-bbb-gpio> >. Acesso em: 17 nov. 2018.

SILA EMBEDDED SOLUTIONS. **Freemodbus**. Disponível em: <https://www.embedded-solutions.at/en/freemodbus/> >. Acesso em: 03 nov. 2018.

WIN-TECH. **Modbus Test & Simulation**. Disponível em: <https://www.win-tech.com/html/modbus1.htm> >. Acesso em: 08 nov. 2018.

AMAZON. **AWS Cloud9**. Disponível em: <https://aws.amazon.com/pt/cloud9/?origin=c9io> >. Acesso em: 08 nov. 2018.

RANDOM NERD TUTORIALS. <https://randomnerdtutorials.com/cloud9-ide-on-the-beaglebone-black/>. Disponível em: <https://randomnerdtutorials.com/cloud9-ide-on-the-beaglebone-black/> >. Acesso em: 12 nov. 2018.

SANTOS, R. PERESTELO, L.M.C. **How to Launch The Cloud9 IDE On Your Beaglebone – Beaglebone For Dummies**. Disponível em: <https://www.dummies.com/computers/beaglebone/how-to-launch-the-cloud9-ide-on-your-beaglebone/> >. Acesso em: 25 nov. 2018.

PAULA, F.B. **Tutorial – Beaglebone Black: Conversão Analógico-Digital**. Disponível em: <http://www.decom.ufop.br/imobilis/tutorial-beaglebone-black-conversao-analogico-digital/> >. Acesso em: 25 nov. 2018.

BEAGLEBOARD. **Latest Images**. Disponível em: <https://beagleboard.org/latest-images> >. Acesso em: 15 nov. 2018.

MICROSOFT. **Internet of Everything**. Disponível em: <https://www.microsoft.com/pt-br/internet-of-things/internet-of-everything-iot>>. Acesso em: 12 nov. 2017.

DAMÁSIO, M. J. **Tecnologia e Educação – As Tecnologias da Informação e da Comunicação e o Processo Educativo**. 1ª. Ed., Lisboa, Editora Nova Vega, 2007.

LICKLIDER, J. **Man-Computer Symbiosis**. 1ª Ed., Londres, Editora Norton & Company, 2001, pp. 55-63

ZAMBARDA, P. **‘Internet das coisas’: entenda o conceito e o que muda com a tecnologia**. Disponível em:

<<https://www.techtudo.com.br/noticias/noticia/2014/08/internet-das-coisas-entenda-o-conceito-e-o-que-muda-com-tecnologia.html> >. Acesso em: 17 nov. 2018.

INTEL. **Internet das Coias**. Disponível em: <www.intel.com.br >. Acesso em: 20 ago. 2017.

BOLZANI, C. A. M. **Desmitificando a Domótica**. Revista Home Theater, 2007.

ALMEIDA, R. **A Tecnologia Por Trás da Mágica**. Disponível em: <<http://quicaze.com/126/a-tecnologia-por-tras-da-magica/> >. Acesso em: 20 jun. 2017.

BARBERA, H. **The Jetsons**. Hanna-Barbera Productions, Inc, 1962.

APÊNDICE A – Código de Aplicação Utilizada

```

BOOL
bCreateFBThread( void )
{

    BOOL      bResult;
    pthread_t  xThread;

    if( pthread_create( &xThread, NULL, pvFBThread, NULL ) != 0 )
    {
        /* Can't create the polling thread. */
        bResult = FALSE;
    }
    else
    {
        bResult = TRUE;
    }
    return bResult;
}

void* pvFBThread( void *pvParameter )
{
    #define min 3850
    #define max 4050

    int aux0;
    int aux1;
    float value;
    int valueP;

    FILE *aval, *LED;
    char buf[50] = "/sys/bus/iio/devices/iio:device0/in_voltage0_raw";
    char buf1[50] = "/sys/class/gpio/gpio67/direction";
    char buf2[50] = "/sys/class/gpio/gpio67/value";
    usRegInputBuf[0] = 0;

    for(;;)
    {
        aval = fopen(buf, "r");
        if(aval == NULL) printf("Analog failed to open\n");
        fseek(aval,0,SEEK_SET); //posiciona o início da leitura
        fscanf(aval,"%f",&value); //lê o valor
        fflush(aval); // limpa variável
        fclose(aval); //fecha

        valueP = ((value-min)/(max-min))*100;
        usRegInputBuf[0] = valueP;
    }
}

```

```

printf("%f\n",value);
usleep(10000000);

aux0 = usRegHoldingBuf[0];
aux1 = ucRegCoilsBuf[0];

if (aux0 == 0) //Automático
{
    if (valueP <= 80) //valor int do nível de luminosidade do LDR de 0 a 100.
    {
        //ligar led
        aux1 = 1;
        ucRegCoilsBuf[0] = aux1;
    }

    if (valueP > 80)
    {
        //desligar led
        aux1 = 0;
        ucRegCoilsBuf[0] = aux1;
    }

}

if (aux0 == 1) //Lâmpada ligada no modo manual
{
    //ligar led
    aux1 = 1;
    ucRegCoilsBuf[0] = aux1;
}

if (aux0 == 2) //Lâmpada desligada no modo manual
{
    //desligar led
    aux1 = 0;
    ucRegCoilsBuf[0] = aux1;
}

if (aux1 == 1)

```

```

{
    //direção
    LED = fopen(buf1, "w");
    if(LED == NULL) printf("GPIO failed to open\n");
    fputs("out",LED); //posiciona o início da leitura
    fclose(LED); //fecha

    //valor
    LED = fopen(buf2, "w");
    if(LED == NULL) printf("GPIO failed to write\n");
    fputs("1",LED); //posiciona o início da leitura
    fclose(LED); //fecha
}

else
{
    //direção
    LED = fopen(buf1, "w");
    if(LED == NULL) printf("GPIO failed to open\n");
    fputs("out",LED); //posiciona o início da leitura
    fclose(LED); //fecha

    //valor
    LED = fopen(buf2, "w");
    if(LED == NULL) printf("GPIO failed to write\n");
    fputs("0",LED); //posiciona o início da leitura
    fclose(LED); //fecha
}

}

return 0;
}

```