UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA
CURSO SUPERIOR DE TECNOLOGIA

YURI ESTEVAN SILVERA

PROJETO, DESENVOLVIMENTO E VALIDAÇÃO DE UMA UNIDADE REMOTA *MODBUS* MICROCONTROLADA

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2013

YURI ESTEVAN SILVEIRA

PROJETO, DESENVOLVIMENTO E VALIDAÇÃO DE UMA UNIDADE REMOTA *MODBUS* MICROCONTROLADA

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Automação Industrial do Departamento Acadêmico de Eletrotécnica – DAELT – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito para obtenção do título de Tecnólogo.

Orientador: Prof. M.Sc. Daniel Balieiro Silva

CURITIBA
2013

YURI ESTEVAN SILVEIRA

PROJETO, DESENVOLVIMENTO E VALIDAÇÃO DE UMA UNIDADE REMOTA *MODBUS* MICROCONTROLADA

Este Trabalho de Diplomação foi julgado e aprovado como requisito parcial para a obtenção do Título de **Tecnólogo em Automação Industrial**, do **Curso Superior de Tecnologia em Eletrotécnica** da **Universidade Tecnológica Federal do Paraná**.

Curitiba, 24 de abril de 2013

_____
Prof. José da Silva Maia, M.Sc.
Coordenador de Curso
Departamento Acadêmico de Eletrotécnica

_____
Prof.ª Rosalba da Costa, M.Sc.
Responsável pelo Trabalho de Conclusão de Curso da Tecnologia
Departamento Acadêmico de Eletrotécnica

BANCA EXAMINADORA

_____
Prof Jorge Assade Leludak, M.Sc.
Universidade Tecnológica Federal do Paraná

_____
Prof. Daniel Balieiro Silva, M.Sc.
Universidade Tecnológica Federal do Paraná
Orientador

_____
Prof. José da Silva Maia , M.Sc.
Universidade Tecnológica Federal do Paraná

_____
Prof.ª Rosangela Winter, M.Sc.
Universidade Tecnológica Federal do Paraná

RESUMO

SILVEIRA, Yuri Estevan. Projeto, Desenvolvimento e Validação de uma unidade remota *MODBUS* microcontrolada. 2013. 90 p. Trabalho de Conclusão de Curso - Curso Superior de Tecnologia, Universidade Tecnológica Federal do Paraná, Curitiba.

Unidades remotas microcontroladas são utilizadas para controlar e adquirir dados de equipamentos localizados remotamente e transferir esses dados para um controlador central. Este trabalho apresenta o projeto, desenvolvimento e validação de uma unidade remota *MODBUS* microcontrolada. Este equipamento foi projetado para funcionar com placas de interface para sinais de entradas digitais, saídas digitais e comunicação através do protocolo *MODBUS*. Foram desenvolvidas quatro placas de circuito impresso que funcionam em conjunto com um microcontrolador PIC18F2550. Através de um programa desenvolvido em linguagem C, o microcontrolador executa uma rotina de comunicação para o protocolo de rede industrial *MODBUS* em conjunto com o padrão USB. Para validação do funcionamento do equipamento foi desenvolvida uma aplicação através de um *software* de supervisão, esta aplicação funciona através de um computador configurado como mestre *MODBUS.* A partir do computador pode-se acionar remotamente o dispositivo e também receber sinais remotos. Pretende-se com esse projeto demonstrar que é possível desenvolver equipamentos de automação industrial utilizando recursos e ferramentas acessíveis.

Palavras-chave: *MODBUS*, Microcontrolador**,** *Software* de supervisão.

ABSTRACT

SILVEIRA, Yuri Estevan. Design, Development and Validation of a remote MODBUS microcontrolled**.** 2013. 90 p. Trabalho de Conclusão de Curso - Curso Superior de Tecnologia, Universidade Tecnológica Federal do Paraná, Curitiba.

Microcontrolled remotes are used to control and acquire data from remotely located equipment and transfer this data to a central controller. This work presents the design, development and validation of a remote MODBUS microcontrolled. This equipment is designed to work with interface boards for signs of digital inputs, digital outputs, and communication via MODBUS protocol. We developed four printed circuit boards that operate in conjunction with a microcontroller PIC18F2550. Through a program developed in C language, the microcontroller executes a communication routine for the industrial network MODBUS protocol in conjunction with the USB standard. To validate the operation of the equipment has been developed through an application of a supervisory software, this application works through a computer configured as MODBUS master. From your computer you can remotely trigger the device and also receive remote signals. The intention of this project demonstrate that it is possible to develop industrial automation equipment using tools and resources available.

Keywords: MODBUS, Microcontroller**,** Software of supervision.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ASCII	*American Standard Code for Information Interchange* (Código Padrão Americano para o Intercâmbio de Informação)
CCS	*Custom Computer Services* (Serviços Personalizados de Computação)
CDC	*Communication Device Class* (Dispositivo de comunicação)
CLP	Controlador Lógico Programável
CRC	Check Redundancy Cyclic (Verificação ciclíca redundante)
EIA	*Electronics Industries Association* (Associação das Indústrias Eletrônicas)
LED	*Light Emissor Diode* (Diodo emissor de luz)
OPC	*OLE for Process Control* (OLE para Controle de Processo)
PCI	Placa de Circuito Impresso
PC	Personal Computer (Computador Pessoal)
PIC	Programmable Intelligent Computer (Computador Inteligente Programável)
RD	*Read* (leitura)
RTU	*Remote Terminal Unit* (Unidade Terminal Remota)
SCADA	*Supervisory Control and Data Aquisition* (Sistema de Supervisão e Aquisição de Dados)
TIA	*Telecomunications Industry Association* (Associação das Industrias de Telecomunicações)
USB	Universal Serial Bus (Rede serial Universal)

SUMÁRIO

1 INTRODUÇÃO

Consolidadas na década de 70, com o desenvolvimento do protocolo *MODBUS*, as redes industriais fazem parte da maioria dos processos automatizados atuais (BORGES, 2007).

De acordo com Borges (2007, p.4) as redes industriais surgiram da necessidade de se distribuir os controles de um processo, a fim de eliminar extensos circuitos elétricos para comando e leitura de sinais em diversos locais de uma fábrica. O sistema de redes permite que diversos equipamentos de automação se comuniquem de maneira inteligente, por um único par de cabos de comunicação, eliminando assim os indesejáveis emaranhados de cabos para transmissão de sinais. Para que tal sistema possa ser implantado, os equipamentos que o compõe devem ser compatíveis com a rede desejada, ou seja, sensores, atuadores, unidades remotas, todos tem de possuir um canal de comunicação com a rede. Além disso, a rede precisa de um controlador que solicita informações e ordena comandos a esses dispositivos.

1.1 TEMA

Segundo Clarke e Reynders (2004, p.19) unidade terminal remota é “...uma unidade independente de aquisição de dados e controle, geralmente baseado em um microprocessador, que monitora e controla um equipamento em local remoto...”. De acordo com Clarke e Reynders (2004, p.20) esses dispositivos, que ficam instalados distantes do controlador do sistema, têm a função de receber os comandos do controlador central e acionar saídas digitais de seu próprio *hardware*, sendo também sua função enviar para o mestre os estados dos sinais que estão nas suas entradas digitais. Para isso, esse tipo de equipamento é composto de entradas e saídas de sinais, sejam analógicos ou digitais e também de um processador capaz de comunicar com a rede industrial e enviar informações relevantes sobre seu funcionamento. Assim, a unidade remota recebe os sinais digitais e analógicos de sensores e instrumentos de um sistema, ligados a suas entradas digitais e analógicas, e os envia para o controlador via rede industrial. Esse mesmo controlador, por sua vez, envia os comandos referentes à lógica do sistema para a unidade remota, via rede industrial, que distribui os comandos para

os atuadores ligados às suas saídas digitais e analógicas. Toda a transmissão dos dados entre o controlador e a unidade remota é feita pela rede industrial no formato do protocolo utilizado pela rede.

1.2 DELIMITAÇÃO DO TEMA

O projeto em questão pretende desenvolver uma unidade remota *MODBUS* microcontrolada. Trata-se de um protótipo composto com seis entradas digitais, seis saídas digitais e um canal de comunicação *MODBUS*. Complementando o protótipo, haverá um *software* dedicado que fará o papel de controlador, este será instalado em um computador que, através de uma porta *USB*, fará a comunicação com a remota *MODBUS*, solicitando dados e enviando comandos.

1.3 PROBLEMAS E PREMISSAS

Existem no mercado diversos equipamentos compatíveis com a rede *MODBUS*, porém em sua grande maioria são produtos importados e pouco acessíveis. Entre os fabricantes nacionais existem alguns, poucos, que investem no desenvolvimento de Unidades Remotas *MODBUS*. Por ser uma tecnologia pouco difundida, existe muita dificuldade em se ter profissionais capazes de desenvolver tais equipamentos, e muitas tentativas são barradas pela falta de conhecimento sobre os protocolos. Logo:

É possível desenvolver uma unidade remota *MODBUS* microcontrolada através de recursos e ferramentas acessíveis?

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Desenvolver uma unidade remota *MODBUS* microcontrolada utilizando recursos e ferramentas acessíveis.

1.4.2 Objetivos Específicos

- ✓ Apresentar um estudo sobre o protocolo *MODBUS*, apresentando sua lógica de funcionamento, a função de cada *bit* de comunicação, as aplicações e a importância do mesmo em um processo;

- ✓ Desenvolver um programa no microcontrolador PIC capaz de manipular os dados do protocolo *MODBUS*, e de estabelecer a comunicação entre o protótipo e o controlador;
- ✓ Desenvolver uma aplicação em *software* de supervisão capaz de controlar a unidade remota *MODBUS*;
- ✓ Montar o protótipo em placa de circuito impresso, e simular um pequeno processo;

1.5 JUSTIFICATIVA

O sistema integrado por redes industriais pode garantir a criação de interfaces de visualização detalhada e controle preciso do processo, utilizando-se sistemas supervisores, IHMs e outros dispositivos.

A construção de uma unidade remota *MODBUS* justifica-se pela falta de um equipamento simples para integrar pequenos processos há sistemas supervisores ou IHMs. Através de um microcontrolador é possível desenvolver um *hardware* através de ferramentas e recursos acessíveis, tornando-o aplicável a pequenos processos.

1.6 PROCEDIMENTOS METODOLÓGICOS

O trabalho divide-se em três partes principais, tendo cada uma sua abordagem específica para o melhor entendimento do assunto. A metodologia empregada em cada uma das partes é:

- ✓ **Pesquisa e referencial teórico:** para o desenvolvimento do projeto fazem-se necessários estudos sobre as redes de comunicação. Para isso é preciso ter uma metodologia de pesquisa, que servirá de base para organizar os dados e para organizar um roteiro técnico, com etapas e sub-etapas, de como proceder durante a execução do projeto.
- ✓ **Programação dos dispositivos de comunicação:** os principais equipamentos que compõe o projeto são programáveis, possuem características que os distinguem como flexíveis, o que os tornam adequados para aplicações variadas. Para torná-los funcionais a um sistema de comunicação, faz-se necessário programá-los adequadamente para esse fim. As referências teóricas são importantes para que a programação seja

bem sucedida e ocorra em menor tempo, para isso prevê-se que os trabalhos de programação e pesquisa ocorram juntos.

- ✓ **Confecção do protótipo físico:** a construção do protótipo envolve habilidades manuais e conhecimentos técnicos. Pretende-se montar o *hardware* eletrônico em placas de circuito impresso, utilizando-se *software*s adequados para desenhar os esquemáticos das ligações da placa, além disso, faz-se necessário o uso de habilidades manuais para confeccionar a placa conforme o desenho desenvolvido em *software*. Para o acabamento externo está prevista a utilização de caixas pré-moldadas em plástico, aa quais possuem recortes nas laterais para interligações com os sinais externos, além de uma abertura para porta de comunicação *MODBUS*. A concepção física do projeto é baseada em modelos já existentes de equipamentos de automação, como Controladores Lógicos Programáveis e tem como principal objetivo estar adequar-se a proposta.

2 EMBASAMENTO TEÓRICO

Nesse capítulo serão apresentados os conceitos de rede de comunicação *MODBUS*, microcontroladores, comunicação USB e Sistemas SCADA.

2.1 REDE *MODBUS*

O *MODBUS* é um padrão aberto de rede de comunicação industrial, sendo suas diretrizes mantidas por uma organização de usuários sem fins lucrativos, denominada *MODBUS Organization*. Segundo essa organização, estima-se que existam mais de sete milhões de redes *MODBUS* em operação somente na América do Norte e Europa. Considerado o primeiro protocolo de rede industrial, o *MODBUS* evoluiu com as novas tecnologias de transmissão de dados, mantendo a mesma estrutura básica de troca de mensagens por código de função (*MODBUS-IDA,* 2005).

2.1.1 História do Protocolo *MODBUS*

A história do protocolo *MODBUS* está diretamente relacionada com a história dos Controladores Lógicos Programáveis. De acordo com Laughnton e Warne (2003, p. 16/4) em 1968 foi desenvolvido o primeiro Controlador Lógico Programável pela associação americana *BedFord Associates,* atendendo a um pedido da empresa automobilística *General Motors.* O dispositivo desenvolvido visava atender a flexibilidade da indústria automobilística, cujas linhas de montagem necessitavam de dinamismo para atender a demanda de mercado por veículos de diferentes modelos. Além de flexível o equipamento eletrônico foi projetado de maneira a ser robusto o suficiente para o ambiente industrial.

O dispositivo desenvolvido pela *BedFord* foi o "*Modullar Digital Controll",* que passou a ser fabricado como "MODICON 084", uma referência ao octogésimo quarto projeto da empresa. A empresa MODICON foi criada em 1968 para promover o novo produto. Em 1977 a empresa foi adquirida pela americana *Gould Electronics*, mais tarde foi vendida para a francesa *Schneider Electric*, atual proprietária (DUNN, 2008).

No início da década de 70 iniciaram-se experiências com um dos primeiros protocolos de comunicação entre Controladores Lógicos Programáveis, este foi denominado *MODBUS*. Em 1979 a MODICON tornou pública as especificações do protocolo, tornando o *MODBUS* um protocolo aberto para desenvolvimento de usuários (*MODBUS-IDA,* 2005).

Em pouco tempo diversos fornecedores desenvolveram dispositivos baseados no novo protocolo aberto, dando início a uma grande mudança nas topologias dos sistemas de automação da década 80. Grandes indústrias passaram a migrar de sistemas centralizados para uma composição com vários CLPs comunicando em rede, conhecidos como sistemas distribuídos (BORGES, 2007).

Segundo Silveira e Santos (2007, p.181) “Nunca um par de fios foi tão bem recebido pelos dispositivos de chão de fábrica, juntamente com toda estrutura já consagrada dentro do empreendimento em nível corporativo”.

Atualmente o protocolo *MODBUS* possui três especificações oficiais: *serial line, Plus* e *TCP-IP* (BORGES, 2007).

O *MODBUS serial line,* utilizado nesse projeto, é um protocolo aberto baseado nos meios RS232 ou RS485, podendo operar em dois modos de transmissão serial: ASCII ou RTU (BORGES, 2007).

O *MODBUS Plus* tem os direitos de utilização reservados à Schneider Eletric, portanto é um protocolo fechado. Trata-se de uma versão aprimorada do *serial line*, utiliza-se do meio físico RS485 e está disponível para integrar redes de comunicação com CLPs desse fabricante (BORGES, 2007).

A variante *MODBUS TCP-IP* é um protocolo aberto, desenvolvido em 1999 em uma parceria entre a Schneider e a *MODBUS*-IDA. Trata-se da versão “encapsulada” do *serial line* para o meio *ethernet*, entre as principais vantagens estão: velocidades de transmissão maiores que os similares seriais, facilidade de conexão com computadores com sistema SCADA, possibilidade de transmissão de dados pela rede *Internet*, utilização de roteadores *wireless* e utilização de equipamentos de baixo custo como *switches* (*MODBUS-IDA*, 2005).

De acordo com Borges (2007, p. 9) o diferencial do protocolo *MODBUS* desde sua criação em 1971 é o sistema troca de mensagens. Caracterizado pela existência de dois tipos de dispositivos na rede: mestre e escravo, onde o mestre

requisita através de códigos de mensagem informações a um dispositivo escravo, o qual responde o código da mensagem ao mestre, juntamente com os dados solicitados.

2.1.2 Arquitetura do protocolo

Todos os protocolos de comunicação utilizados internacionalmente devem seguir um modelo de funcionamento determinado pela *Organization for Standardization* (ISO). O *MODBUS* segue esse modelo desde sua publicação em 1979. Para Lopez (2001, p.47) “Um modelo é um padrão que organiza conceitos gerais ou fornece diretrizes como uma descrição facilmente compreendida”.

O modelo citado é conhecido como OSI (*Open System Interconection* – Interconexão de Sistemas Abertos), sendo conceitualmente dividido em sete níveis hierárquicos. De acordo com Silveira e Santos (2007, p.187) cada nível trabalha atendendo ao nível mais alto utilizando-se dos níveis mais baixos. Na Figura 1 está uma representação do modelo OSI.

Figura 1 -Os sete níveis do modelo OSI
Fonte: SILVEIRA e SANTOS (2007, p.187).

De acordo com *MODBUS-IDA* (2006, p.5) o protocolo *MODBUS* em sua variante *serial line* utiliza-se das camadas um, dois e sete do modelo OSI, relacionadas a seguir:

- Nível de aplicação também conhecido como sétima camada, nela o protocolo deve ter uma sub-rotina para que possa trocar dados com o programa de usuário, além de gerenciar essa troca de mensagens (SILVEIRA e SANTOS, 2007);

- Nível de enlace também conhecido como segunda camada, assegura a integridade do conteúdo da mensagem da origem ao destino (SILVEIRA e SANTOS, 2007);
- Nível físico também conhecido como primeira camada, onde são definidas as características do meio de transmissão dos protocolos. Tais como características dos condutores, níveis de tensão, níveis de corrente, padrão dos conectores, limitações das topologias e distribuição dos dispositivos (SILVEIRA e SANTOS, 2007);

Na Figura 2 está uma representação das camadas do modelo OSI aplicada ao protocolo *MODBUS serial line*:

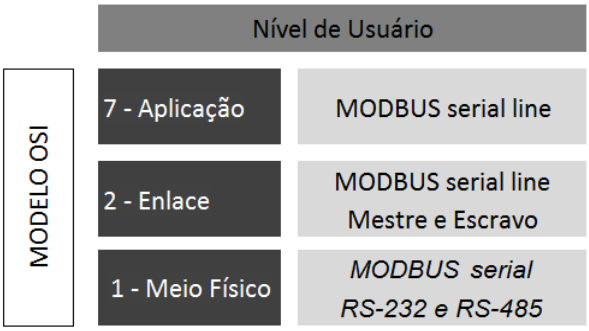

Figura 2 - Modelo OSI *MODBUS serial line*
Fonte: Adaptado de *MODBUS IDA* (2006, p.5).

2.1.3 Meio de transmissão RS232 serial

De acordo com *MODBUS-IDA* (2002, p.5) o RS232 (*Recommended Standard 232*) é um dos meios físicos utilizados pelo *MODBUS serial line*, sendo aplicado apenas para conexão entre dois dispositivos ponto a ponto.

De acordo com Clarke e Reynders (2004, p.35) esse padrão foi desenvolvido em 1969 nos Estados Unidos pelo departamento de engenharia da EIA (*Electronis Industries Association*), tendo sido atualizado em 1991 pela mesma associação. Esse padrão define as características elétricas dos sinais, a interface mecânica dos conectores e o funcionamento dos circuitos eletrônicos.

O modo serial de transmissão através do padrão RS232 consiste na troca de dados em forma binária. Cada bit é transferido sequencialmente e é identificado por sinais elétricos nos emissores e receptores (STRANGIO, 2012).

De acordo com Strangio (2012) os dados são trocados de forma binária através de pacotes de dez ou onze *bit*s, sendo:

- Oito ou sete *bit*s de dados;
- Um *bit* para identificar o início da transmissão, denominado *start bit*;
- Um *bit* para identificar o final da transmissão, denominado *stop bit*;
- Um *bit* para paridade, que é utilizado para verificar a integridade dos dados;

Na Figura 3 está uma representação do pacote de transmissão de 11 *bit*s com paridade.

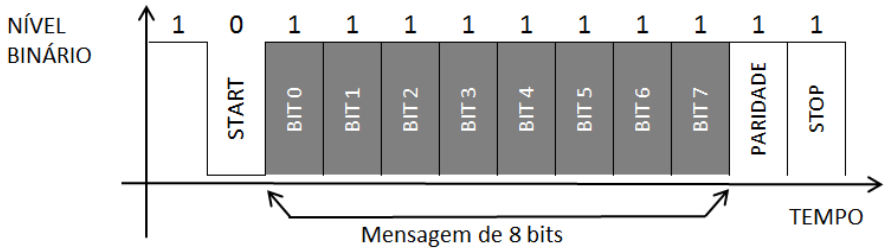

Figura 3 - Pacote de transmissão 11 *bit*s RS232 serial
Fonte: Adaptado de STRANGIO (2012).

O início da transmissão acontece quando é identificada uma mudança no estado de repouso da transmissão com o *start bit* indo a nível lógico “zero”, seguindo-se oito *bit*s de dados, um *bit* de paridade e um *stop bit* em nível lógico “um” finalizando a transmissão de um pacote (STRANGIO, 2012).

De acordo com Strangio (2012) a paridade é um método de detecção de erros nos pacotes transferidos, normalmente ela é configurável através de um *software* que gerencia a porta serial dos computadores, podendo ser de três tipos:

- Nula: Não é feito nenhum tipo de verificação dos dados, estando o *bit* de paridade inoperante (STRANGIO, 2012);
- Par: O *bit* de paridade é ajustado para nível lógico “um” ou “zero” para que o número total de *bit*s em estado “um” seja um número par (STRANGIO, 2012);
- Ímpar: O *bit* de paridade é ajustado para nível lógico “um” ou “zero” para que o número total de *bit*s em estado “um” seja um número ímpar (STRANGIO, 2012);

De acordo com Laughnton e Warne (2012, p.1268), quando o pacote de dados é recebido, é feita a conferência da paridade com o número de *bit*s, se o número total de *bit*s em estado lógico “um” não for correspondente a paridade configurada, par ou ímpar, o pacote é identificado como corrompido.

Os dados podem ser transmitidos em velocidade de até 20000 *bit*s por segundo, porém não existe regra para um valor mínimo, sendo usualmente usada

as velocidades de taxa de transmissão de 300, 1200, 2400, 9600 ou 19200 *bit*s por segundo. Assim como a paridade, os dois dispositivos devem estar configurados com a mesma velocidade para estabelecer a comunicação (STRANGIO, 2012).

2.1.4 Sistema de mensagens *MODBUS*

Em uma rede *MODBUS serial* os dispositivos que compões a rede podem ser de dois tipos: mestre ou escravo. Segundo *MODBUS*-IDA (2002 p.7) "... somente um mestre é conectado a rede, e um ou muitos (no máximo 247) escravos também são conectados na mesma rede *serial...*". As transmissões sempre devem ser iniciadas pelo dispositivo mestre, sendo que este pode solicitar aos escravos uma informação, caracterizando uma "leitura", ou pode ordenar ao escravo uma execução, caracterizando uma "escrita". Na Figura 4 está uma representação de uma rede *MODBUS* formado por três escravos e um dispositivo mestre.

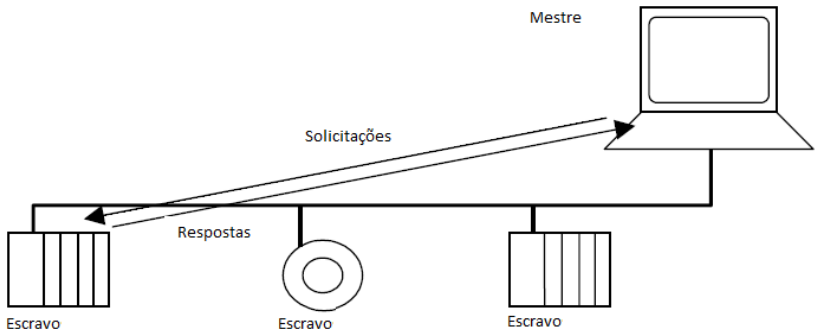

Figura 4 - Rede MODBUS
Fonte: *MODBUS*-IDA (2006, p.7).

As solicitações do mestre podem ser do tipo *unicast mode* ou *broadcast mode*. No *unicast* o mestre requisita informação a somente um escravo, que após receber e processar a mensagem envia uma resposta ao mestre. No *broadcast* o Mestre pode enviar uma mensagem para todos os escravos, nessa solicitação os escravos não devem responder e a mensagem do mestre é necessariamente de "escrita" (*MODBUS*-IDA, 2006).

De acordo com *MODBUS*-IDA (2002, p.7) cada escravo que compõe uma rede deve ter um endereço distinto dos outros escravos, podendo ser um número de 1 a 247. O mestre da rede não possui endereço. O endereço zero é reservado para identificar mensagens do tipo *broadcast* para todos os escravos da rede.

2.1.5 Modos de Transmissão RTU e ASCII

De acordo com Clarke e Reynders (2004, p.46) existem dois modos de transmissão do protocolo *MODBUS*:

- *RTU - Remote Terminal Unit* – Unidade Terminal Remota. Nesse modo os *bytes* de mensagens transmitidos pela serial devem estar em formato hexadecimal (CLARKE e REYNDERS, 2004);
- *ASCII - American Standard Code for Information Interchange* – Código Padrão Americano para o Intercâmbio de Informação. Nesse modo os *bytes* devem ser transmitidos em formato de caractere seguindo uma tabela de códigos padrão ASCII (CLARKE e REYNDERS, 2004).

2.1.6 *Frame MODBUS*

De acordo com *MODBUS*-IDA (2006, p.8) a estrutura básica de um *frame MODBUS* contém quatro partes distintas: o endereço do dispositivo escravo, o comando a ser executado, uma quantidade variável de *bytes* e uma verificação de erros na mensagem. No Quadro 1 tem-se uma representação do formato básico de um *frame* de comunicação *MODBUS:*

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low, CRC Hi

Quadro 1 - *Frame* básico *MODBUS* serial line
Fonte: *MODBUS*-IDA (2006, p.13)

De acordo com MODICON (1996, p.19) quando o mestre solicitar a um escravo leitura de registros de um dispositivo escravo, este deve enviar uma estrutura de comunicação como a do Quadro 2:

Endereço	Comando	Endereço inicial dos registros		Quantidade de registros		CRC	
06h	03h	00h	6Bh	00h	03h	B8h	44h
1 *BYTE*	1 *BYTE*	1 *BYTE*	1 *BYTE*	1 *BYTE*	1 *BYTE*	1 *BYTE*	1 *BYTE*

Quadro 2 - *Frame* de um mestre *MODBUS*
Fonte: Adaptado de MODICON (1996, p.19).

Analisando o quadro 2, o primeiro *byte* da mensagem contém o endereço do escravo, nesse exemplo o endereço é o 06. O segundo *byte* contém o código da função, nesse exemplo é utilizada a função 03, uma função para leitura de múltiplos registros. O terceiro e o quarto *byte* correspondem ao endereço da memória inicial que será acessada no escravo, nesse caso o endereço é o 6B (107). No quinto *byte* e sexto *byte* é escrita a quantidade de memórias a serem lidas do escravo, no exemplo foi solicitada três memórias de registro que correspondem a três bytes. No sétimo e oitavo *byte* estão as informações do CRC (*cyclical redundancy check*), que é uma verificação de erros na mensagem (MODICON, 1996).

De acordo com MODICON (1996, p.19) quando um escravo responder a um mestre a leitura de registros, este deve enviar uma estrutura de comunicação como a do Quadro 3:

Endereço	Comando	Quantidade *bytes*	Dados		CRC	
1h	3h	2h	0h	0h	B8h	44h
1 *BYTE*	1 *BYTE*	1 *BYTE*	1° REGISTRO		1 *BYTE*	1 *BYTE*
			1 *BYTE*	1 *BYTE*		

Quadro 3 - *Frame* de um escravo *MODBUS*
Fonte: Adaptado de MODICON (1996, p.19).

Analisando o Quadro 3, o primeiro *byte* do *frame* de resposta é o endereço do escravo, nesse caso, é o endereço 01. O segundo *byte* contém o código da função solicitada pelo mestre, a função 03 é uma função para leitura de múltiplos registros. O terceiro *byte* informa a quantidade de *bytes* que estão sendo transmitidos, nesse caso são dois *bytes* que correspondem a um registro. O quarto e o quinto *bytes* contém as informações que estavam no primeiro registro, no exemplo os valores é 00. O sexto e o sétimo *bytes* contém as informações do CRC (MODICON, 1996).

2.1.7 Códigos de função *MODBUS*

De acordo com *MODBUS*-IDA (2006, p.11) os códigos de função, também chamados de comandos, podem ser classificados em dois grupos: os códigos para acessar informações de *bit*s e os códigos para acessar informações de 16 *bit*s. Os

*bit*s são identificados como *coils* quando são *bit*s de saída digital, e como *inputs* quando são *bit*s de entrada digital. Já as informações de 16 *bit*s são identificadas como *holding register* quando são memórias retentivas ou saídas analógicas, e como *input register* quando são entradas analógicas.

Os escravos *MODBUS* processam os códigos de função solicitados pelo mestre, porém eles são limitados a alguns tipos de comando. Existem escravos que processam apenas funções de *bit*s, e existem escravos que processam só um ou dois tipos de função de 16 *bit*s, cada dispositivo contém um manual *MODBUS* contendo o mapa de memória com o tipo de função disponível para comunicação, cabe ao desenvolvedor que configura a rede estabelecer as rotinas de comunicação corretas para cada escravo. Na Tabela 1 está a relação de funções mais utilizadas pelo protocolo *MODBUS serial line* (*MODBUS* IDA, 2006).

Tabela 1 - Operandos *MODBUS*

Função	Descrição	Código
ReadCoil Status	Leitura de saída digitais	1h (1)
Read Input Status	Leitura de entradas digitais	2h (2)
Read Holding Register	Leitura de registros retentivos (memórias)	3h (3)
Read Input Register	Leitura de registros de entradas (entradas analógicas)	4h (4)
Force Single Coil	Escrita de uma saída digital	5h (5)
Preset Single Register	Escrita de um único registro (memória)	6h (6)
Force MultipleCoils	Escrita de várias saídas digitais	Fh (15)
PresetMultipleRegister	Escrita de vários registros (memórias)	10h (16)

Fonte: Adaptado de *MODBUS*-IDA (2006, p.11).

2.1.8 Método de detecção de erros CRC

De acordo com MODBU IDA (2002, p.39) o *Cyclical Redundancy Checking* (Conferência de Redundância Cíclica) utiliza dois *bytes* do *frame MODBUS*, sendo o primeiro denominado *low CRC*, referente ao *byte* menos significativo, e o segundo denominado *high CRC*, referindo-se ao *byte* mais significativo.

No processo de rotina para execução do CRC, inicialmente uma variável de 16 *bit*s é carregada com valor hexadecimal FFFFh. Cada *byte* transmitido serialmente na mensagem *MODBUS* é comparado por uma função XOR (OU exclusivo) com o *byte* menos significativo da variável CRC inicial (*MODBUS*-IDA, 2002). .

No Quadro 4 está uma representação dessa operação:

CRC Low								CRC High								Hexadecimal
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFh
Mensagem *MODBUS* (1 byte)																
0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	A3h
OU Exclusivo XOR																
1	1	1	1	1	1	1	1	0	1	0	1	1	1	0	0	FF5Ch
FFh								5Ch								

Quadro 4 - CRC inicial
Fonte: Adaptado de *MODBUS*-IDA (2002, p.41).

De acordo com *MODBUS*-IDA (2002, p.39) os *bit*s resultantes da operação anterior são deslocados uma posição à esquerda, sendo o *bit* mais significativo preenchido por um valor “zero”. O *bit* que foi deslocado para fora da variável é analisado, se ele for “um” é executada mais uma comparação do tipo XOR com uma constante de valor A001h, porém se esse *bit* for zero nada é feito. Os deslocamentos são repetidos oito vezes em todos os *bytes* do *frame*, e sempre que o *bit* deslocado for “1” executa-se mais uma vez a XOR com a constante A001h. O Valor resultante desse processo é dividido em dois *bytes*, o menos significativo é o resultado do *Low CRC*, e o mais significativo é o resultado do *High CRC.* Cada *byte* de mensagem passa por essa rotina sendo incialmente comparado com o valor inicial FFFFh e depois sofrendo oito deslocamentos.

No Quadro 5 está uma demonstração do deslocamento dos *bit*s.

Hexadecimal	Deslocamento da variável																*Bit* deslocado para análise
FFh	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	Constante A001h																
A001h	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	
	OU exclusivo XOR																
89FEh	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	

Quadro 5 - CRC deslocamento dos *bit*s
Fonte: Adaptado de *MODBUS*-IDA (2006, p.41).

O CRC sempre é calculado pelo dispositivo que enviou o *frame*. O dispositivo mestre recalcula o CRC e compara com o valor recebido, se os valores forem diferentes é retornada uma mensagem de erro de CRC (*MODBUS*-IDA, 2006).

2.2 COMUNICAÇÃO USB

De acordo com USB (2000, p.1) O padrão USB (*Universal Serial Bus*) foi desenvolvido por um consórcio de grandes empresas da área de informática. O objetivo era criar uma conexão entre o computador e seus periféricos sem desligar a máquina. Mais do que isso o USB passou a ser um padrão de comunicação entre dispositivos eletrônicos e computadores, não apenas em relação ao protocolo USB, mas também para os cabos, conectores e alimentação de dispositivos.

Segundo Microchip (2008, p.1) o padrão USB “tornou muito simples para os usuários finais conectar dispositivos periféricos a um computador pessoal, eliminando a miríade de diferentes interconexões que costumava ser necessário” . Na área de automação o USB vem substituindo aos poucos as portas seriais do padrão RS232, assim como as antigas portas paralelas. Entre as vantagens do USB estão a velocidade maior, conectores simplificados e um melhor gerenciamento por *software*.

De acordo com USB (2000, p.16) o protocolo USB baseia-se em dois tipos de *hardwares*: o mestre, denominado *host* e o escravo denominado *device*. Todas as comunicações são iniciadas pelo *host*, que normalmente é um computador. O *software* do sistema operacional do computador gerencia o *host*, bem como as conexões e os endereços dos *devices*.

2.2.1 Classes de dispositivos USB

De acordo com USB (2000, p.22) “os dispositivos USB são divididos em classes como: *hub*, *human interface*, *printer*, *imaging* ou *mass storage device*.”. A seguir está a especificação das três classes mais utilizadas:

- *Mass Storage Device* (Dispositivo de armazenamento em massa), utilizado para cartões de memória (USB, 2000);
- *Human Interface Device* (Dispositivo de Interface humana) são dispositivos reconhecidos automaticamente pelos sistemas operacionais sem a necessidade de instalar *drivers*. São periféricos como mouse e teclado (USB, 2000);
- *Communication Device Class* (Dispositivo de comunicação), esses dispositivos simulam uma comunicação serial do tipo RS232 através da USB, com velocidades maiores que os antigos RS232, pode se alcançar até 115 mil *bit*s por segundo de velocidade de transmissão (USB, 2000);

Segundo Microchip (2008, p.1) o padrão USB CDC é “destinado a dispositivos como modems e interfaces de rede, um subconjunto dos recursos CDC pode ser utilizado para emular uma porta serial UART virtual”.

2.3 MICROCONTROLADORES

De acordo com Souza (2005, p.22) “...poderíamos definir microcontrolador como um pequeno componente eletrônico, dotado de uma inteligência Programável...”.

Um microcontrolador possui todos os periféricos necessários para seu funcionamento dentro do mesmo circuito integrado, inclusive um próprio processador (SOUZA, 2005, p.22).

Desenvolvidos no início da década de 70, inicialmente foram utilizados em equipamentos eletrônicos que necessitavam de controle dedicado, como teclados de computadores e calculadoras. Atualmente eles fazem parte de muitos equipamentos populares como rádios, controles remotos, relógios, calculadoras, eletrodomésticos, portões automatizados, entre outros (SOUZA, 2005, p.22).

2.3.1 Arquitetura

De acordo com Souza (2005, p. 22), os microcontroladores são compostos basicamente por um processador, memórias para armazenamento de dados e programa, terminais para sinais de entrada e saída, assim como os periféricos internos que mudam de acordo com o modelo de microcontrolador, os mais comuns são osciladores, interface serial assíncrona, conversores A/D (analógico para digital) e módulos com temporizadores.

De acordo com Souza (2005, p.22) a maneira como o processador acessa as memórias de programa e dados, depende da arquitetura do dispositivo que normalmente pode ser de dois tipos:

- *Harvard:* nessa arquitetura o dispositivo possui duas memórias separadas para dados e para programa, sendo respectivamente do tipo RAM e *flash*. O processador acessa essas memórias através de barramentos de dados separados (SOUZA, 2005);

- *Von Neumann:* nessa arquitetura existe apenas uma memória para armazenar o programa e os dados, consequentemente um único barramento para acessar as memórias (SOUZA, 2005).

De acordo com Souza (2005, p.22) a concepção *Harvard* de arquitetura é mais recente, foi desenvolvida para aperfeiçoar o tempo para executar um programa.

2.3.2 Memórias

O programa do microcontrolador fica armazenado em uma memória do tipo *Flash* denominado não volátil. Ela pode ser modificada inúmeras vezes através de um terminal de programação, nela é armazenada a rotina de execução das instruções e funções. Quando é retirada a energia do dispositivo, o programa permanece na memória e volta a ser executado quando se reestabelece a alimentação (SOUZA, 2002, p.16).

De acordo com Souza (2002, p.13) a memória do tipo RAM (*Random Access Memory*), ou memória de acesso aleatório, é do tipo volátil. Seus dados são apagados sempre que o dispositivo é desligado. Esse tipo de memória permite leitura e escrita de dados, é utilizado nos microcontroladores para armazenar as variáveis e os registradores utilizados pelo programa.

De acordo com Souza (2002, p.16) existe ainda a memória EEPROM, trata-se também de uma memória de dados, porém essa é do tipo não volátil. Ou seja, os dados armazenados nela não são perdidos quando desligado o microcontrolador, trata-se portanto de uma memória retentiva para armazenar os dados.

2.3.3 Ciclos de Máquina

Souza (2008, p. 85) explica que cada instrução de um programa de um microcontrolador é executada em um ciclo de máquina, sendo que os ciclos de máquina são determinados a partir de sinais de *clock*. Esse sinal é gerado através de um oscilador ligado externamente ao dispositivo ou através de um oscilador interno configurado via *software* de programação. Os *clocks* são gerados em frequência de mega Hertz (MHz), e para cada quatro sinais de *clock* um ciclo de máquina é executado. Tomando como exemplo uma frequência de 40 MHz de

clock, a frequência de execução de uma instrução será de 10 MHz, já o tempo do ciclo de máquina será o valor “1” divido por 10 MHz resultando em 100ns.

De acordo com Souza (2005, p. 25) o *clock* dos microcontroladores Microchip é dividido em quatro fases Q1, Q2, Q3 e Q4. Na fase Q1 o microcontrolador indica a próxima instrução a ser executada através do *program counter* (contador de linha de programa). Nas fases de Q2 a Q4 a instrução é buscada na memória do programa e armazenada no registrador de instruções. No próximo ciclo essa instrução é executada no intervalo de Q1 a Q4. Essa característica de funcionamento é conhecida como *pipeline*. Dessa maneira o microcontrolador executa uma instrução em cada ciclo ao mesmo tempo em que indica a próxima instrução a ser executada. Esse tipo de execução só pode ser feito em dispositivos com arquitetura *Harvard*, apresentando uma maior velocidade de processamento que as arquiteturas tradicionais.

Na Figura 5 está uma representação do conceito *pipeline* e as quatro fases do ciclo de máquina.

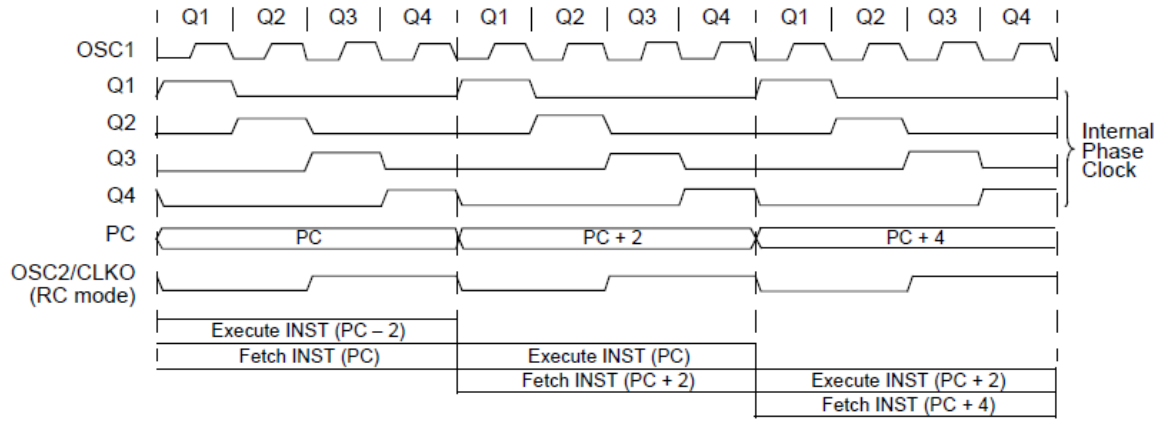

Figura 5 - Pipeline
Fonte: MICROCHIP (2009, p.63).

De acordo com Souza (2008, p.82) o tempo para execução de um ciclo de máquina é importante na utilização de temporizadores, pois utilizam esse valor como referência de incremento de tempo.

2.3.4 Portas de I/O

Segundo Souza (2008, p.93) as portas de I/O “permitem com que o microcontrolador tenha acesso ao mundo externo, permitindo com que ele saiba o estado de um sensor por exemplo, ou que acione alguma saída”.

As portas de I/Os dos microcontroladores são configuráveis através de *software* e seus pinos correspondentes diferem para cada modelo de microcontrolador. Para microcontroladores Microchip são identificados como PORTA e PORTB, estando acionados através de níveis lógicos em seus pinos (SOUZA, 2005, p. 42).

2.3.5 Configurações *FUSES* (*configuration bits*)

Os microcontroladores possuem configurações de *hardware* que são ativadas através de *software*. Existem posições na memória dos dispositivo reservadas para essas configurações (MICROCHIP, 2009, p. 30).

Essas configurações podem variar para cada modelo de microcontrolador. Porém algumas são comuns a todos os modelos: referencia.

- Configuração frequência oscilador;
- Configuração de função *watchdog* (cão de guarda);
- Proteção do código;
- Configuração dos pinos para programação;
- Configuração do pino *MasterClear* como reset da rotina do programa;

2.4 OPC

Segundo Straight (2006), o OPC é “uma especificação ou um conjunto de regras escritas e procedimentos para o modo que múltiplos programas ou aplicações troquem dados entre si”.

As especificações oficiais desse padrão são administradas pela *OPC Foundation*, que desenvolveu esse modelo com a intenção de padronizar a maneira como as aplicações e programas trabalham em conjunto. A necessidade de se padronizar a integração entre aplicações surgiu na indústria, onde diversos CLPs de marcas diferentes precisavam ser integrados com *softwares* de supervisão também diferentes. Dessa maneira se dois aplicativos possuírem especificação OPC, eles podem se conectar através de um *software* OPC (STRAIGHT, 2006).

Um servidor OPC é um *driver* que fornece dados para outras aplicações de *software*. Existem no mercado diversos modelos de servidores OPC, isso acontece devido aos diferentes fornecedores de *hardware*, sendo que cada um possui interfaces de comunicação com protocolos diferentes (WHAT, 2008).

2.5 SISTEMAS SCADA

De acordo com Clarke e Reynders (2004, p.1) “SCADA (controle de supervisão e aquisição de dados) refere-se à combinação de telemetria e aquisição de dados.”.

Uma aplicação SCADA, também conhecida como sistema de supervisão, pode utilizar um *driver* OPC para acessar dados de uma rede de comunicação, ou, pode utilizar-se de *drivers* de fabricantes de controladores de automação. Em uma topologia básica um computador com sistema SCADA comunica diretamente com um controlador, que por sua vez está conectado a outros dispositivos através de IO’s ou redes de comunicação (CLARKE; REYNDERS, 2004).

Veja representação de um sistema SCADA na Figura 6:

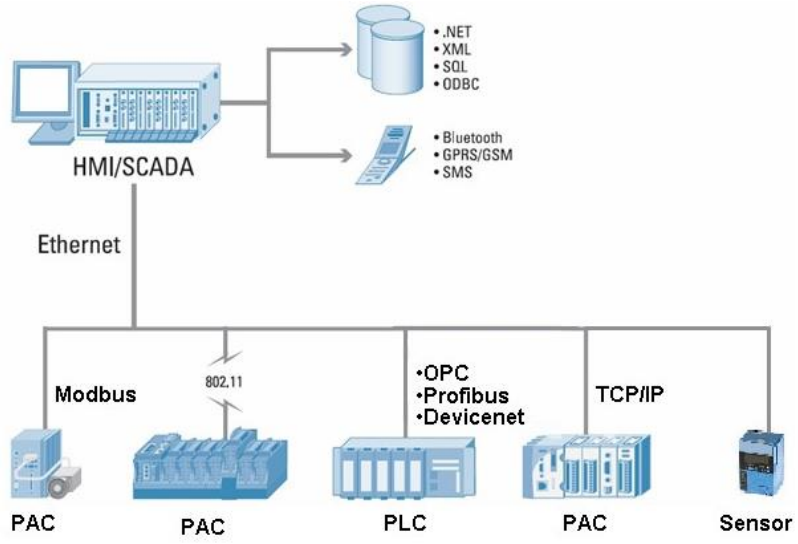

Figura 6 - Topologia SCADA
Fonte: NATIONAL (2012).

Atualmente os sistemas de supervisão tem facilitado o controle de complexos sistemas de automação, proporcionando uma melhor interação entre os

operadores e as máquinas. Utilizando-se de tecnologias de computação e comunicação, esses sistemas permitem um controle abrangente sobre diversos equipamentos localizados remotamente, tornando a interface com o usuário simples e agradável (ROSÁRIO, 2005).

3 DESENVOLVIMENTO

A unidade remota *MODBUS* foi desenvolvida com base no microcontrolador PIC18F2550 do fabricante *Microchip.* Esse dispositivo foi programado para comunicar com um dispositivo mestre *MODBUS* na modalidade *Serial Line RTU*, utilizando-se de uma porta *USB* para o meio físico de transmissão dos dados. A Unidade funciona como um dispositivo escravo que transmite e executa funções a partir das solicitações do mestre *MODBUS*, tendo placas separadas para entradas e saídas digitais.

Para o mestre foi utilizado um microcomputador configurado através de um *OPC Server* para comunicar com o escravo. Uma aplicação supervisório foi desenvolvida para demonstrar o funcionamento da remota. Na Figura 7 está uma representação do sistema desenvolvido:

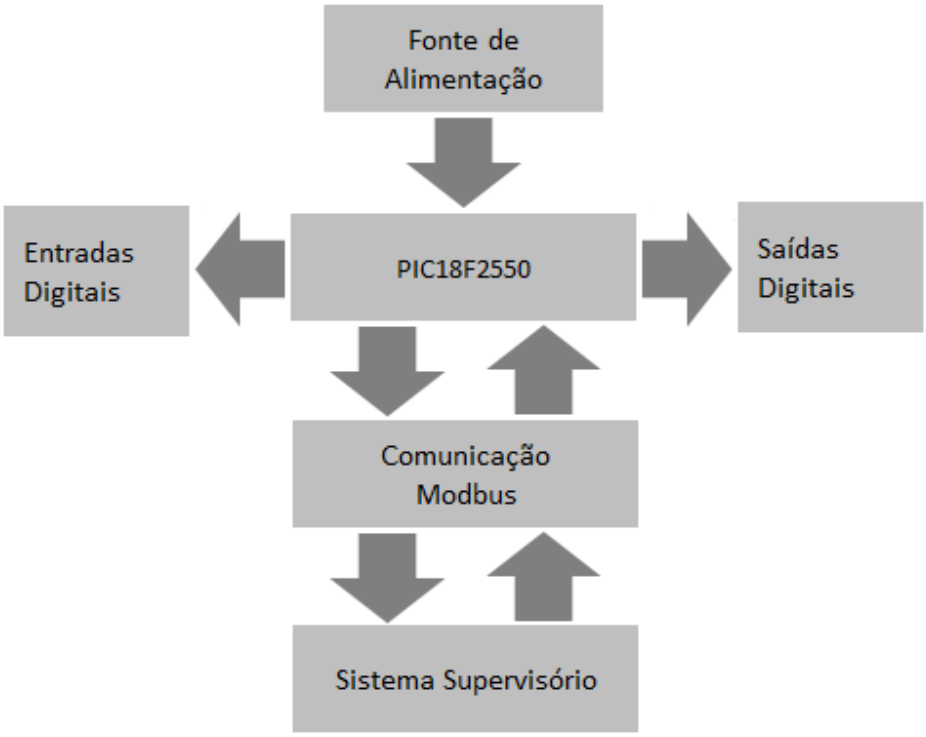

Figura 7 - Fluxograma Básico
Fonte: Autoria própria.

Para demonstrar o funcionamento prático foi desenvolvida uma pequena bancada com dispositivos para acionar as entradas digitais. Nas saídas foram ligadas três lâmpadas, um microventilador e um emissor de som. Na aplicação supervisório, existem animações para as entradas e botões de comandos para acionar as saídas digitais.

3.1 CONSTRUÇÃO DA UNIDADE REMOTA *MODBUS*

Para construção da unidade foram projetados circuitos eletrônicos através do *software* ISIS 7. Foi necessária a elaboração de quatro projetos separados para que a unidade funcionasse de acordo com a proposta, resultando em quatro placas de circuito impresso que foram conectadas umas as outras.

Para as entradas digitais foi projetada uma placa contendo os circuitos que convertem os sinais das entradas digitais de 12Vcc para 5Vcc, isso foi possível através de optoacopladores. Para as saídas digitais foram projetadas duas placas, uma contendo os circuitos de acionamento através de optoacopladores, e uma segunda placa contendo os acionamentos através de relés. Para o microcontrolador foi feita uma quarta placa contendo os componentes necessários para o funcionamento do dispositivo. A disposição final das placas que fazem parte da unidade remota *MODBUS* ficou como na Figura 8.

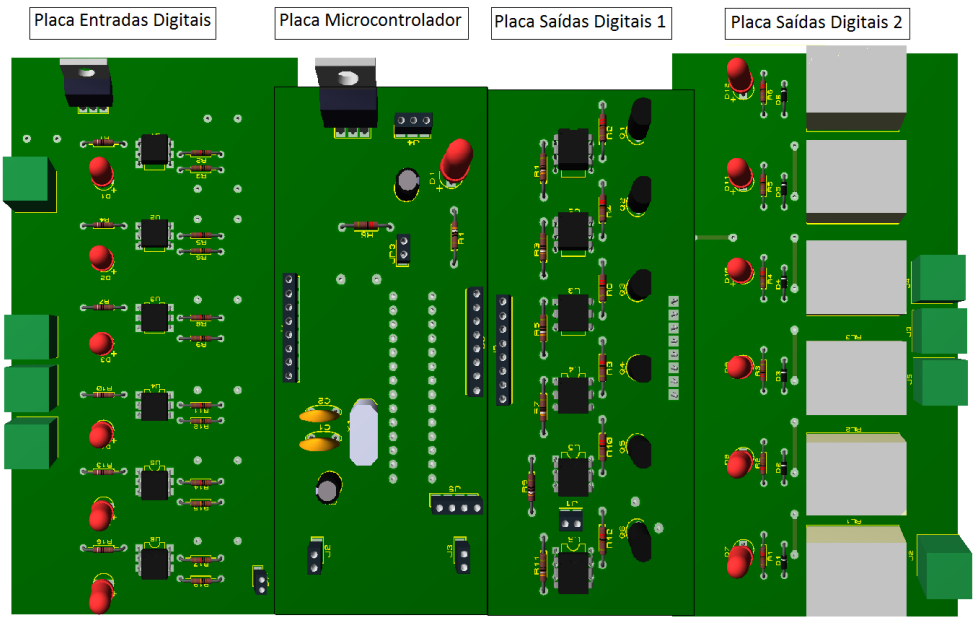

Figura 8 - Disposição Placas Circuito Impresso
Fonte: Autoria própria.

As quatro placas foram projetadas para serem fixadas em uma placa de policarbonato, sendo dispostas lado a lado como na Figura 9. Para interligação entre uma placa e outra foram previstos conectores e cabos próprios para conexões de PCIs (placas de circuito impresso). A fonte de alimentação fica instalada externamente ao circuito e possui tensão fixa de doze volts contínuos (12VCC). A comunicação com o supervisório é feita através de um conector USB do tipo fêmea, fixado próximo a placa.

3.1.1 Placa de Entradas Digitais

Para que a Unidade possa fazer a interface com os sinais de campo foi projetada uma placa com seis entradas digitais, todas para funcionamento em 12 Volts contínuos. A placa também possui os pontos de conexão para fonte de alimentação externa, a partir dela são alimentados os circuitos das outras três placas.

Cada entrada possui um circuito separado para receber o sinal de campo, totalizando seis circuitos para as entradas digitais nessa placa. Cada um é composto por um resistor de entrada, um *LED*, um optoacoplador, e dois resistores para o circuito de *pull-down* para o PIC. Na figura 9 está uma representação do circuito de acionamento da entrada digital.

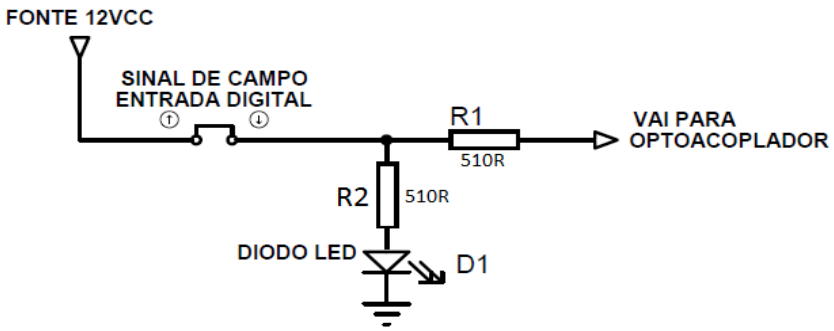

Figura 9 -Circuito *LED* Entrada Digital
Fonte: Autoria própria.

O *LED* é utilizado para indicação de entrada ativa, ele acenderá sempre que a entrada digital possuir doze volts em seu terminal. O resistor R2 e R1 de 510Ω tem a função de limitar a corrente para o *LED* e para o optoacoplador em 20mA. O valor dos resistores foi obtido através da fórmula da primeira lei de ohm.

$$U=R.I \text{ (1° lei de Ohm)}$$

$$U_{fonte} - U_{Led} = R.(I_{Led})$$

$$12VCC - 1{,}96VCC = R.(20mA)$$

$$R = 502\Omega \rightarrow \text{Resistor Comercial} = 510\Omega$$

O optoacoplador foi utilizado para isolar eletricamente os sinais digitais de campo das portas do microcontrolador, evitando também possíveis ruídos elétricos que dificultariam a leitura dos sinais. Esse dispositivo recebe um sinal de entrada e emite um sinal de luz dentro do encapsulamento, esse sinal de luz faz a saturação de um transistor foto sensível, dessa maneira o isolamento é feito através de uma conexão óptica.

Na figura 10 está uma representação do circuito eletrônico com o optoacoplador e os resistores de *pull down*. Pode-se notar pela figura que a placa opera com duas tensões: 12VCC e 5VCC. A segunda tensão é obtida através de um regulador de tensão 7805 que reduz os 12VCC para 5VCC, essa tensão é necessária para ativar os níveis lógicos da porta do microcontrolador.

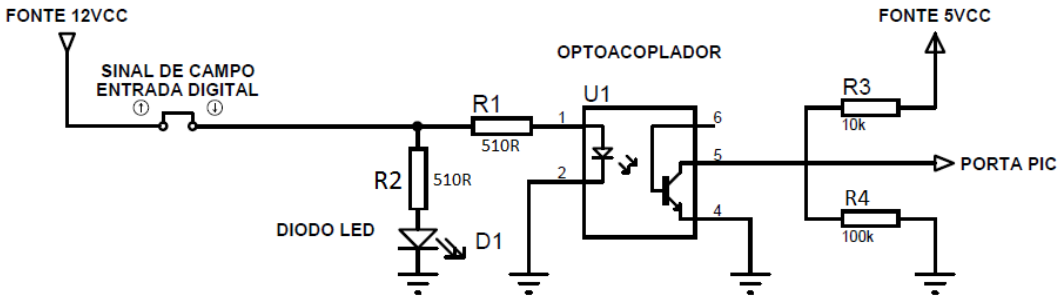

Figura 10 - Circuito Optoacoplador entrada digital
Fonte: Autoria própria.

Com o acionamento do pino “1” o optoacoplador satura o transistor foto sensível, chaveando os pinos “5” e “4” para passagem de corrente. A saída do pino “4” está ligada ao GND, dessa maneira quando é acionada uma entrada digital um nível lógico baixo é reconhecido na porta do microcontrolador.

3.1.2 Placa de Saídas de Digitais 1

Para o acionamento dos dispositivos de saída foram confeccionadas duas placas, uma para o circuito dos optoacopladores e outra para os relés e *LED*s

sinalizadores. A placa de Saídas Digitais 1 faz a interface entre os sinais do microcontrolador e a placa de saídas a relé, utilizando-se de optoacopladores e transistores. Na Figura 11 está uma representação do circuito de saídas digitais.

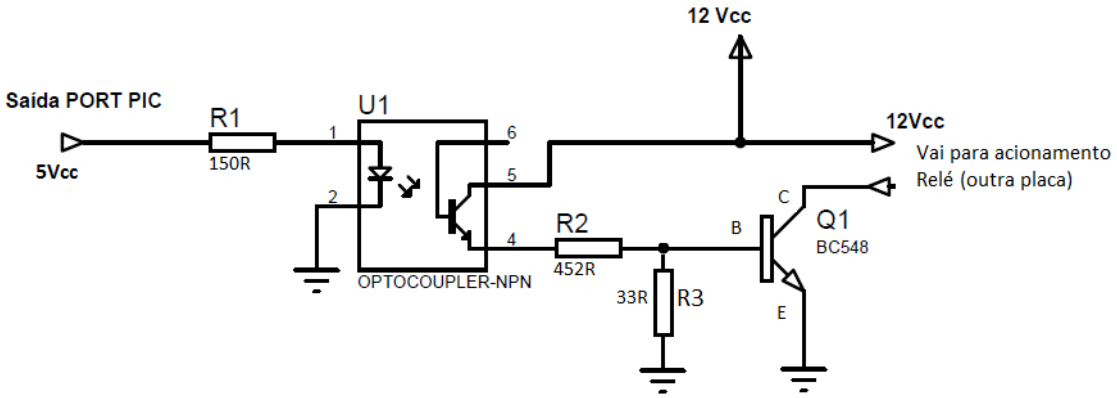

Figura 11 - Circuito Saídas Digitais 1
Fonte: Autoria própria.

O optoacoplador foi utilizado com a finalidade de isolar eletricamente as portas do PIC dos circuitos de acionamento das saídas. Conforme representado na Figura 12, o optoacoplador recebe o nível alto de uma das portas do microcontrolador através do pino “1”, um resistor de 300 ohms foi utilizado entre o PIC e o pino “1” para limitar a corrente em 10mA. O valor do resistor foi obtido através da fórmula da 1° lei de ohm. Veja:

$$U = R1.I \text{ (1° lei de ohm)}$$

$$U \text{ saída microcontrolador} - U \text{ Led} = R1.(I \text{ Led optoacoplador})$$

$$5VCC - 1{,}96VCC = R1.(20mA)$$

$$R1 = 152\Omega \rightarrow \text{Valor comercial} = 150\Omega$$

Com o acionamento do pino “1” o optoacoplador aciona o transistor foto sensível, chaveando os pinos “4” e “5” para passagem de corrente. Na saída do pino “4” está a base de um transistor NPN BC548, que ao ser acionado permite a passagem de corrente para acionamento dos relés da segunda placa.

O BC548 foi utilizado por permitir o uso de correntes de até 100mA, o suficiente para acionar os relés, que consomem cerca de 30mA em suas bobinas e o LED que consome 20mA. Os resistores de 430 Ohms e 33 Ohms foram dimensionados para saturar o transistor, permitindo que a corrente de base “Ib”

seja vinte vezes menor que a corrente do coletor de 50mA, ou seja, em torno de 2,5mA. Esses componentes também funcionam como divisores de tensão permitindo que entre a base e o emissor do transistor tenha 0,7V de tensão. Tanto a corrente de 2,5mA como a tensão de 0,7V são condições para saturar um transistor do tipo BC548 para uma carga de até 50mA. Veja os cálculos:

$$\text{Corrente Coletor} = Ic = 50mA$$
$$\text{Tensão entre Base e Emissor} = Vbe = 0,7V$$

O circuito divisor de corrente deve consumir cerca de dez vezes a corrente da base do transistor “Ib”, logo:

$$I = 10.Ib = 25mA$$

A corrente do resistor R2 deve ser 25mA e o cálculo do resistor é feito pela 1° lei de ohm:

$$U = R2.I \text{ (lei de ohm)} \rightarrow 12Vcc-0,7Vcc = R2.(25mA) \rightarrow R2=452\Omega$$
$$\text{Valor comercial resistor R2} = 430\Omega$$

A corrente do resistor R3 deve ser 22mA e o cálculo do resistor é feito pela 1° lei de ohm:

$$U = R3.I \rightarrow 0,7Vcc = R3.(22mA) \rightarrow R3 = 32\Omega$$
$$\text{Valor comercial resistor R3} = 33\Omega$$

3.1.3 Placa Saídas de Digitais 2

A placa “Saídas Digitais 2” contém os relés para o acionamento dos dispositivos conectados as saídas da unidade remota *MODBUS*.

Para o acionamento dos relés de saídas foi projetada uma segunda placa, contendo seis circuitos iguais para acionamento dos componentes. Cada circuito é composto por um *LED*, um resistor, um diodo e um relé. Na Figura 12 está uma representação do circuito de acionamento da saída digital:

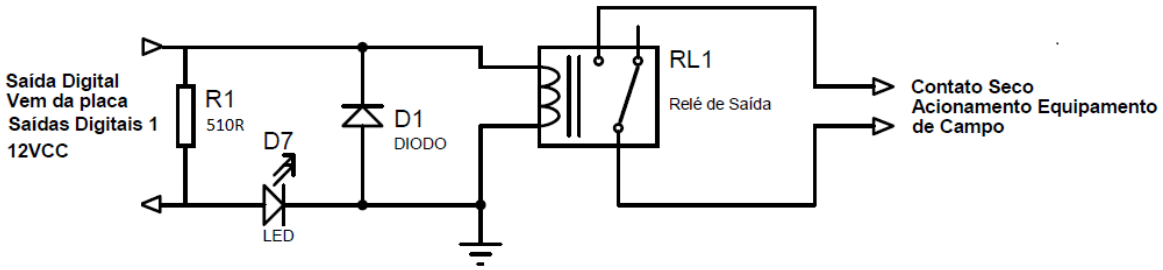

Figura 12 - Saídas Digitais 2
Fonte: Autoria própria.

O *LED* é utilizado para indicação de saída ativa, ele acenderá sempre que o microcontrolador acionar a porta de saída correspondente ao relé. O resistor de 510Ω tem a função de limitar a corrente para o *LED* em 20mA para uma tensão de 12VCC. . O valor do resistor foi obtido através da fórmula da lei de ohm. Veja:

$$U = R.I \text{ (lei de Ohm)}$$

$$U_{acionamento\ saída} - U_{led} = R.(I_{led})$$

$$12Vcc - 1{,}96Vcc = R.(20mA)$$

$$R = 502\ \Omega \rightarrow \text{Valor comercial} = 510\ \Omega$$

O acionamento do relé é feito em tensão de 12VCC, esse dispositivo possui um contato normal aberto e um contato normal fechado com um terminal comum entre eles. Para o circuito da placa foi utilizado apenas o contato normal aberto, este suporta tensões de até 127VAC ou 24VDC com corrente máxima de 15A.

3.1.4 Placa do Microcontrolador

Para comunicar com o computador via *MODBUS* foi desenvolvido um *hardware* de comunicação a partir de um microcontrolador, o modelo utilizado foi o PIC 18F2550 fabricado pela empresa *MICROCHIP*. O dispositivo executa as rotinas de comunicação, acionamentos das saídas digitais e leitura das entradas digitais. Na Figura 13 está uma representação do dispositivo:

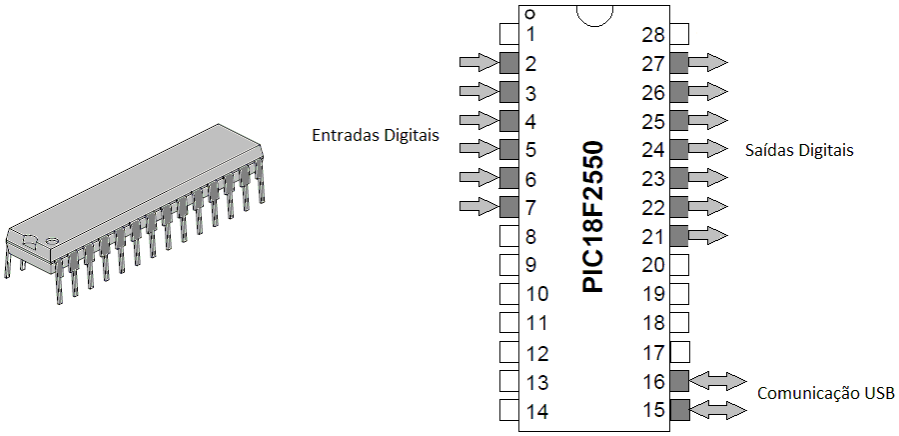

Figura 13 - Esquemático PIC18F2550
Fonte: Adaptado de MICROCHIP (2009).

O PIC possui como periférico um canal para comunicação USB através dos pinos 15 e 16, que correspondem aos PORTS RC4/D- e RC5/D+. Para utilização desse canal foi utilizado um conector USB tipo B que está fixado à placa da unidade remota, esse tipo de conector é o mesmo utilizado em CLPs e impressoras. Para conectar a unidade remota ao computador mestre é necessário um cabo *USB*, como exemplificado na figura 14:

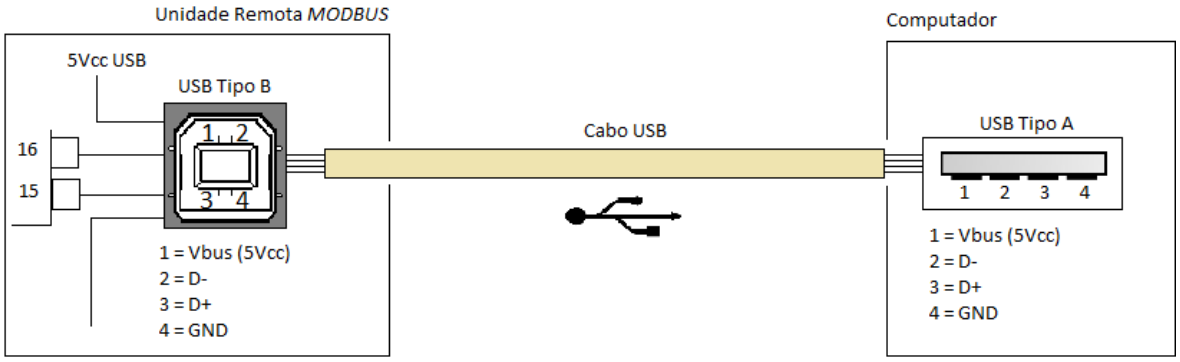

Figura 14 - Conexão USB
Fonte: Autoria própria.

Os computadores normalmente possuem porta *USB* do tipo A, por isso é necessário que o cabo possua uma ponta do tipo A macho, e outra ponta do tipo B macho. A velocidade dos dados no barramento *USB* para esse projeto é de 9600 *bits* por segundo.

Como representado na figura 13 os pinos de “2” a “7” correspondem as entradas digitais da unidade remota, já os pinos de “21” a “27” correspondem as

saídas digitais. Pela arquitetura do PIC18F2550 as entradas digitais estão sendo utilizadas através das portas do registro PORTA do dispositivo, enquanto que as saídas digitais estão sendo utilizadas através das portas do registro PORTB. Na Figura 15 os *bit*s correspondentes ao PORTA estão indicados pelas siglas RA0 a RA6, já os *bit*s do PORTB estão indicados pelas siglas RB0 a RB7.

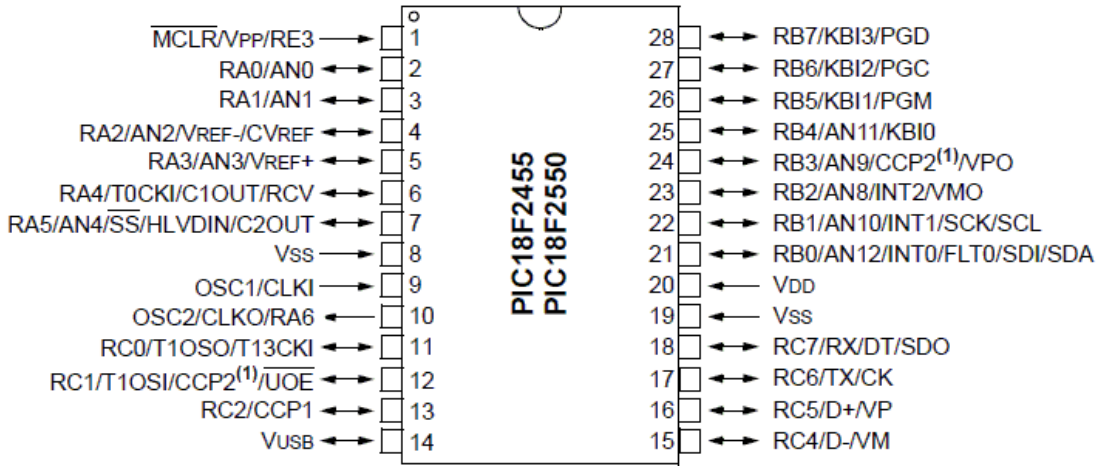

Figura 15 - PORTA e PORTB
Fonte: Adaptado de MICROCHIP (2009).

A tensão necessária para funcionamento do circuito dessa placa é 5VCC, o microcontrolador recebe a tensão positiva nos pinos “1” e “20”, enquanto que os pinos “8” e “19” são conectados ao 0Vcc do circuito. Existem duas opções de fonte alimentação: utilizar a tensão 5VCC proveniente do canal USB do computador mestre, ou utilizar o mesmo valor de tensão proveniente de uma fonte externa.

Para o *clock* de processamento do PIC18F2550 foi utilizado um cristal de 20MHz em conjunto com dois capacitores cerâmicos de 22pF. A frequência gerada por esse componente é responsável pela velocidade de processamento do microcontrolador. O cristal foi conectado aos pinos “9” e “10” do microcontrolador, enquanto que um dos capacitores foi conectado entre o 0VCC e o pino “9” e o outro entre o 0VCC e pino “10”. A função dos capacitores cerâmicos é filtrar o sinal de alta frequência gerado pelo cristal. O circuito está representado na figura 16.

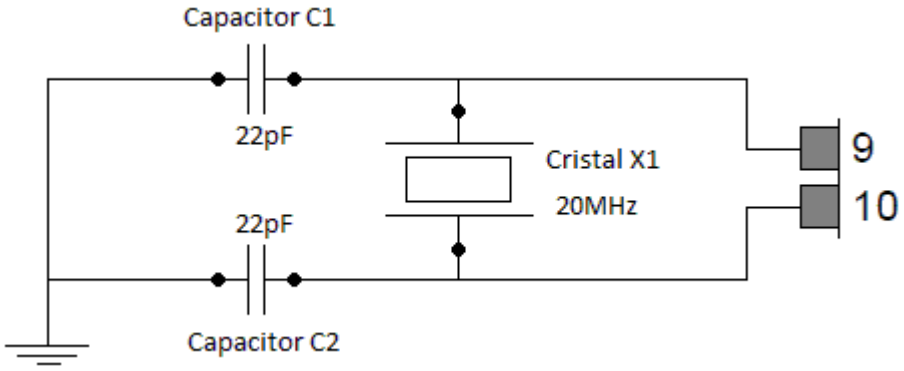

Figura 16 - Cristal
Fonte: Autoria própria.

No Apêndice B (placa do microcontrolador) encontra-se o desenho da placa do microcontrolador para consulta.

3.2 BANCADA DE TESTES

A bancada de testes foi construída para utilização prática da unidade remota *MODBUS* microcontrolada, através dela podem ser testados os sinais digitais e demonstrada a interface de comunicação USB. São seis dispositivos eletromecânicos instalados nas entradas digitas e seis equipamentos elétricos conectados nas saídas digitais.

A bancada foi confeccionada a partir de uma placa de policarbonato branca de 10mm de espessura, nela foram parafusados os equipamentos e instaladas canaletas e caixas de passagem de cabos. Na parte superior da bancada foram instalados os dispositivos que são acionados pelas saídas digitais, enquanto que na parte inferior ficaram os equipamentos que fazem o acionamentos das entradas digitais.

A unidade remota *MODBUS* Microcontrolada está fixada no centro da bancada e conectada eletricamente aos demais componentes. Os cabos utilizados são de espessura de 1,5mm² e devidamente identificados com anilhas apropriadas.

A conexão com o computador mestre é feita a partir da porta USB da unidade remota. Para demonstração de funcionamento do aplicativo supervisório foi utilizado um computador portátil com sistema operacional *Windows Seven Home Basic.* Na Figura 17 está uma representação da bancada de testes.

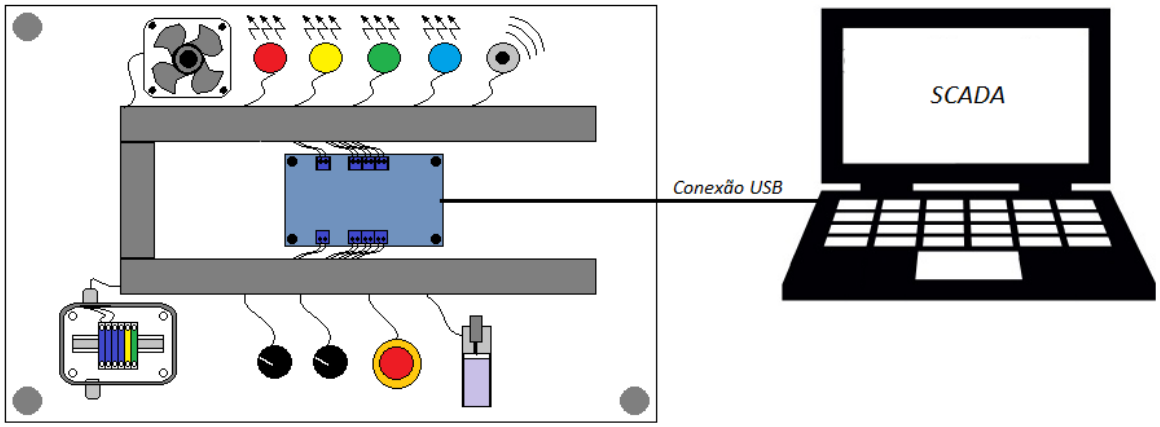

Figura 17 - *Layout* bancada de testes
Fonte: Autoria própria.

Para demonstrar o funcionamento das entradas e saídas digitais foram utilizados os seguintes equipamentos:

- Uma chave três posições com dois contatos normalmente abertos que estão conectadas as entradas digitais “ED0” e “ED1”;
- Uma chave com dois botões reversíveis com contatos normalmente abertos conectados as entradas “ED2” e “ED3”;
- Uma botoeira de emergência com bloqueio no acionamento conectada a entrada digital “ED4”;
- Uma chave fim de curso conectada a entrada digital “ED5”
- Um ventilador 12Vcc conectado a saída digital “SD0”;
- Um indicador luminoso vermelho conectado a saída digital “SD1”;
- Um indicador luminoso amarelo conectado a saída digital “SD2”;
- Um indicador luminoso verde conectado a saída digital “SD3”;
- Um indicador luminoso azul conectado a saída digital “SD4”;
- Um alarme sonoro conectado a saída digital “SD5”;

Todos os equipamentos possuem indicação no aplicativo supervisório através de animações gráficas sempre que acionados. Assim torna-se possível a demonstração das funcionalidades do sistema de comunicação entre mestre e escravo do protocolo *MODBUS*.

3.3 *SOFTWARE* DE SUPERVISÃO

Para demonstrar o funcionamento da unidade remota *MODBUS* foi desenvolvida uma aplicação Supervisório através do *software* Elipse E3. Através de um driver de comunicação OPC foi estabelecida a comunicação entre a unidade remota e o E3. Na Figura 18 está representada a Tela do aplicativo:

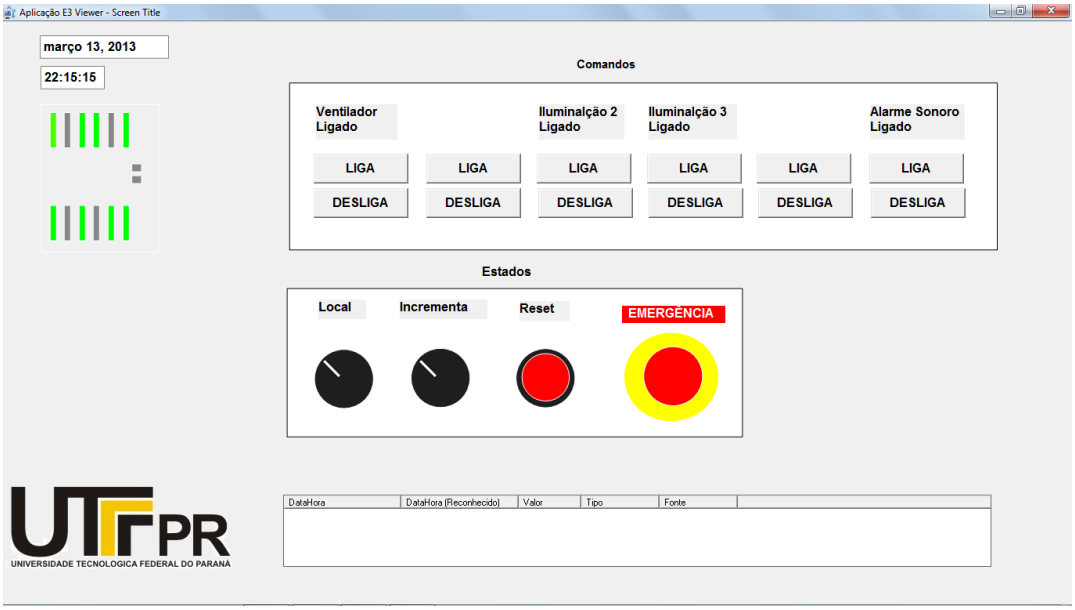

Figura 18 -Tela Aplicação Supervisório
Fonte: Autoria própria.

A interface foi desenvolvida com base nos equipamentos da bancada de testes. Na Figura 18, o retângulo branco na parte superior da tela concentra os comandos que acionam as saídas digitais, o retângulo na parte central reúne os estados das chaves conectadas as entradas digitais da unidade remota, no canto superior esquerdo encontram-se a data e o horário atual, logo abaixo desses itens está uma representação da unidade remota *MODBUS* com suas saídas e entradas digitais.

Os comandos são acionados através de botões com as inscrições “LIGA” e “DESLIGA”, conforme representado na Figura 19. Sempre que uma saída é acionada um aviso correspondente ao componente torna-se visível na tela indicando que está ligado.

Figura 19 -Comandos do Supervisório
Fonte: Autoria própria.

As posições das chaves da bancada podem ser visualizadas através de animações, conforme representado na Figura 20. Sempre que uma chave mudar de posição uma mensagem torna-se visível indicando a função dessa chave.

Figura 20 - Estados da Bancada de Testes no Supervisório
Fonte: Autoria própria.

Os estados de entrada e saídas acionadas da unidade remota *MODBUS* podem ser visualizados através da representação destacada na Figura 21. Uma indicação na cor verde aparecerá na entrada ou saída correspondente.

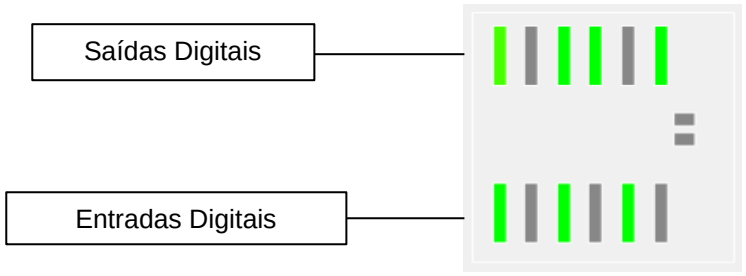

Figura 21 - Estados da unidade remota *MODBUS*
Fonte: Autoria própria.

3.4 PROGRAMA DO MICROCONTROLADOR

O dispositivo de controle da Unidade funciona através de um código de programação. Desenvolvido através de linguagem C, o código contém as

instruções lógicas necessárias para manipular os dados da comunicação, bem como para ler e acionar as portas de interface. Para esse projeto a rotina de programação foi desenvolvida através do *software* compilador PCW da fabricante *Custom Computer Services* (*CCS)*.

O programa código é executado ciclicamente no microcontrolador, este segue etapas de acordo com a ordem em que as instruções foram alocadas nas linhas de programa. A rotina desenvolvida para unidade remota *MODBUS* funciona de acordo com o fluxograma da Figura 22.

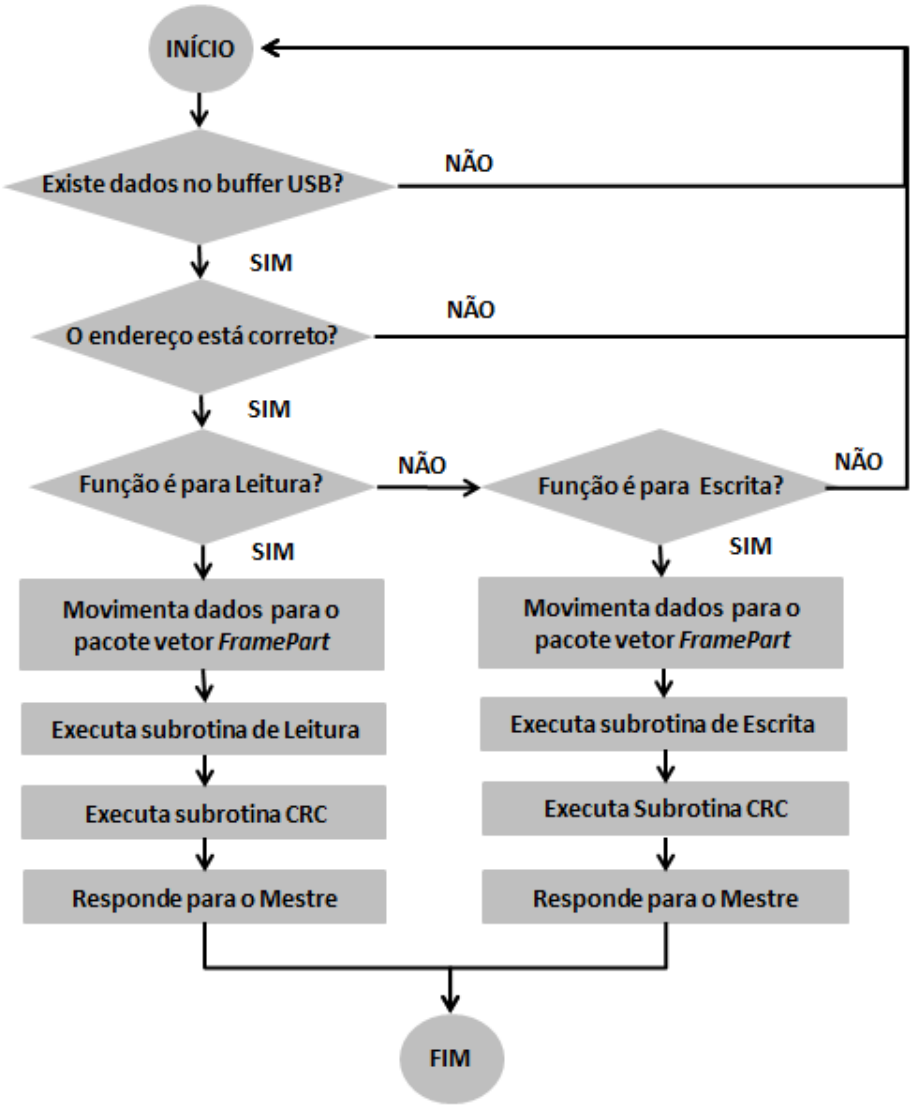

Figura 22 - Fluxograma
Fonte: Autoria própria.

O programa principal utiliza três sub-rotinas principais, essas foram desenvolvidas para simplificar o código, tornando-o mais organizado e diminuindo o número de linhas. A primeira sub-rotina do código é para cálculo do CRC, a segunda é para responder as solicitações de leitura do mestre e a terceira é para função de escrita *MODBUS*.

O programa principal também utiliza recursos de bibliotecas desenvolvidas pelos fabricantes CCS e Microchip, elas são utilizadas principalmente para comunicação USB. Uma biblioteca de usuário foi desenvolvida para separar as configurações do PIC do código principal, essa biblioteca chama-se <ConfigUSB.h>.

3.4.1 Configurações FUSES PIC18F2550

As definições de funcionamento do microcontrolador foram separadas em uma biblioteca nomeada <ConfigUSB.h>, essa contém o código com as configurações *FUSES* do dispositivo.

Como explicado na seção 2.3.5 o microcontrolador possui diversas opções de funcionamento, podendo ser configurado pelo próprio código C ou através da interface do compilador CCS. As configurações dependem da aplicação do projeto eletrônico, e podem variar entre um desenvolvedor e outro.

Para esse microcontrolador foram utilizadas nove configurações, relacionadas na Tabela 2.

Tabela 2 - Configurações *fuses*

FUSES	Descrição
HSPLL	High Speed Control/Resonator with PLL Enabled
PLL5	Divide by 5(20Mhz oscilator input)
USBDIV	USB clock source comes from PLL divide by 2
CPUDIV1	No System Clock Postscaler
VREGEN	USB volatge regulator enabled
NOWDT	No Watch Dog Timer
NOPROTECT	Code not protected from reading
NOLVP	No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
NODEBUG	No Debug mode for ICD

Fonte: Autoria própria.

- *FUSE* ***HSPLL:*** Permite que o microcontrolador opere em alta velocidade, ou seja, em uma frequência de *clock* de até 96MHz. Ainda assim é necessário

utilizar um oscilador externo denominado cristal, para esse projeto foi utilizado um cristal de 20 MHz;

- *FUSE* ***PLL5***, divide a frequência do oscilador externo (20Mhz) por cinco, resultando em 4MHz. Essa frequência é necessária para ativar os 96MHz do HSPLL, conforme fluxograma do fabricante Microchip (MICROCHIP, 2009, p24);
- *FUSE* ***USBDIV:*** Divide os 96MHz por dois, resultando em 48MHz. Essa frequência é necessária para funcionamento da comunicação USB (MICROCHIP, 2009);
- *FUSE* ***CPUDIV1*** divide o *clock* de 96MHz por dois, permitindo que a CPU opere em 48MHz para execução do programa, sem a utilização do oscilador interno;
- *FUSE* ***NOWDT*** desabilita a função *Watchdog*, pois não foi desenvolvida lógica para essa função;
- *FUSE* ***NOPROTECT*** deixa o código desprotegido para cópias por upload;
- *FUSE* ***NOLVP*** desabilita tensão baixa no pino B5 para programação, pois esse pino é utilizado como porta de saída digital**.** Para programação utilizam-se os pinos *Master Clear*, PGD e PGC;
- *FUSE* ***DEBUG*** permite que o dispositivo seja programado por um dispositivo de programação ICD.

O código completo dessa biblioteca encontra-se no Apêndice E (programa do microcontrolador).

3.4.2 Frequência de Funcionamento

O código principal, também chamado *main*, possui a rotina lógica para funcionamento da unidade remota. O programa opera em 48MHz de frequência, o que significa que as instruções de programa serão executadas em 12MHz, um quarto da frequência principal. Nessa frequência uma instrução será executada a cada 80ns. Na demonstração dos cálculos abaixo “Fcy” é o *clock* de processamento das instruções e “Fosc” é o *clock* principal do sistema, veja:

$$Fcy = Fosc/4 = 48MHz/4 = 12MHz$$

$$\text{Ciclo de máquina} = 1/Fcy = 1/12MHz = 80\ ns$$

3.4.3 Bibliotecas

A CCS oferece um conjunto de bibliotecas denominado “*PIC USB Drivers code”*, trata-se de vários códigos que facilitam a integração de dispositivos USB pelo protocolo CDC. A biblioteca <usb_cdc.h> concentra as principais funcionalidades desse conjunto, ela é declarada no código principal do programa e utiliza-se de bibliotecas secundárias para estabelecer a comunicação USB. Na Tabela 3 estão listadas as bibliotecas do fabricante CCS.

Tabela 3 -Bibliotecas utilizadas no programa

Biblioteca	Desenvolvedor	Descrição
usb_cdc.h	CCS	Biblioteca principal para comunicação USB CDC
usb_desc_cdc.h	CCS	Código do driver de comunicação para *Windows*
usb.c	CCS	Interrupções nível de *hardware* (*level hardware layer*)
pic18_usb.h	CCS	Habilita as interrupções de comunicação USB
usb.h	CCS	Contém as configurações do protocolo USB

Fonte: Autoria própria.

- **<usb_cdc.h>:** Código utilizado para estabelecer uma conexão RS232 virtual entre o computador e o PIC. Essa biblioteca tem disponíveis em seu código diversas funções para leitura de dados, *status* de comunicação e configurações de pacotes. Nesse projeto foram utilizadas as funções “usb_cdc_init( )”,” usb_cdc_getc( )” e ” usb_cdc_putc( )”;
- **<usb.h>:** Código utilizado pela biblioteca <usb_cdc.h>, possui as configurações do protocolo USB. Suas funções não precisam ser utilizadas no programa principal;
- **<usb.c>:** Código utilizado pela biblioteca <usb_cdc.h>, possui as interrupções do nível de *hardware*. Suas funções não precisam ser utilizadas no programa principal;
- **<usb_desc_cdc.h>:** Código utilizado pela biblioteca <usb_cdc.h>, trata-se do driver de comunicação para o sistema operacional *Windows*. Suas funções não precisam ser utilizadas no programa principal;
- **<pic18_usb.h>:** Código utilizado pela biblioteca <usb_cdc.h>, ativa os recursos do PIC18F2550 para comunicação USB. Essa biblioteca tem disponíveis em seu código diversas funções para conexão do PIC através da USB. Nesse projeto foram utilizadas as funções “usb_init( )” e “usb_task( )”;

As funções das bibliotecas utilizadas no programa principal estão detalhadas nas seções 3.4.5, 3.4.6 e 3.4.7.

3.4.4 Variáveis

Foram utilizadas sete variáveis declaradas por usuário, elas foram utilizadas nas funções e no programa principal. Na Tabela 4 está a relação dessas variáveis com a descrição básica de cada uma.

Tabela 4 - Variáveis utilizadas no programa

Variável	Tamanho	Descrição
long int checksum = 0xffff	32*bit*s	Verificação CRC
int32 *Frame*Part[100]	32*bit*s	Vetor de 100 posições - Utilizada para o *frame MODBUS*
int TamanhoFrame	8*bit*s	Utilizada para indicar posição do CRC no *frame*
unsigned char *Byte*MaisCRC	8*bit*s	Pacote de dados *byte* menos significativo
unsigned char *Byte*MenosCRC	8*bit*s	Pacote de dados *byte* mais significativo
unsigned int Uint_1	8*bit*s	Auxiliar - Utilizada na subrotina CRC
unsigned int Uint_2	8*bit*s	Auxiliar - Utilizada na subrotina CRC

Fonte: Autoria própria.

A utilização das variáveis está detalhada nas seções 3.4.5, 3.4.6 e 3.4.7.

3.4.5 Programa Principal (main program)

Nas primeiras linhas do código são executadas três sub-rotinas para estabelecer a comunicação USB. Essas sub-rotinas encontram-se nas bibliotecas desenvolvidas pelo fabricante CCS, já citadas anteriormente na seção 3.4.3. Na Tabela 5, estão relacionadas essas sub-rotinas com as respectivas descrições.

Tabela 5 -Subrotinas para comunicação USB

SUB ROTINA	Biblioteca	Desenvolvedor	DESCRIÇÃO
usb_cdc_init();	<usb_cdc.h>	CCS	Inicializa CDC
usb_init();	<pic18_usb.h>	CCS	Inicializa USB
usb_task();	<pic18_usb.h>	CCS	Conexão USB entre PC e PIC

Fonte: Autoria própria.

- usb_cdc_init( ): Inicializa a comunicação USB do tipo CDC. Através dessa função são definidos os parâmetros de comunicação serial, que para essa

biblioteca CCS são: velocidade de 9600 *bit*s por segundo, sem paridade, oito *bit*s de dados e um *stop bit*;

- usb_init( ): Inicializa o periférico USB do PIC e o conecta o dispositivo ao barramento USB;
- usb_task( ): Controla o barramento USB inicializando ou interrompendo a conexão USB pelas funções “usb_attach( )” e “usb_detach( )”;

No Apêndice E (programa do microcontrolador) encontra-se o código completo do programa principal.

Na sequência do código estão as configurações das portas do dispositivo através das funções “set_tris_a( )” e “set_tris_b( )”. Estas funções configuram os *IO’s* dos registros PORTA e PORTB para funcionarem como entradas ou saídas digitais. Como citado na seção 3.1.4 os *bit*s do PORTA foram utilizados para as entradas digitais e os do PORTB como saídas digitais. Na Tabela 6 estão indicados os pinos do microcontrolador, os *bits* dos registradores e as suas configurações de *software*.

Tabela 6 - Registradores da entradas e saídas digitais

Pino PIC	*Bit* do Registrador	Configuração *Software*	Função unidade remota
2	RA0	Entrada	Entrada Digital 0 – ED0
3	RA1	Entrada	Entrada Digital 1 – ED1
4	RA2	Entrada	Entrada Digital 2 – ED2
5	RA3	Entrada	Entrada Digital 3 – ED3
6	RA4	Entrada	Entrada Digital 4 – ED4
7	RA5	Entrada	Entrada Digital 5 – ED5
21	RB0	Saída	Saída Digital 5 – SD5
22	RB1	Saída	Saída Digital 4 – SD4
23	RB2	Saída	Saída Digital 3 – SD3
24	RB3	Saída	Saída Digital 2 – SD2
25	RB4	Saída	Saída Digital 1 – SD1
26	RB5	Saída	Saída Digital 0 – SD0

Fonte: Autoria própria.

Sempre que o mestre *MODBUS* enviar um *frame* ao escravo o programa iniciará uma rotina de identificação, recebimento e verificação dos dados. Havendo conexão USB entre o escravo e o mestre essa rotina será executada em *loop* infinito através do comando “while (1)” presente no código.

Os *frames* enviados por qualquer dispositivo mestre são estruturados de acordo com o padrão internacional determinado pela organização *MODBUS*-IDA,

explicado na seção 2.1.6. O *frame* enviado pelo mestre *MODBUS* pode ser de dois tipos:

- Solicitação de Leitura de Múltiplos Registros: Esta é a função três do protocolo *MODBUS*. Este *frame* contém oito *bytes* para solicitação de leitura de dois registros.
- Solicitação de Escrita de Múltiplos Registros: Esta é a função dezesseis do protocolo *MODBUS*. Este *frame* contém onze *bytes* para solicitação de escrita de dois registros.

Se o mestre solicitar a leitura de dados pela função três, o programa da Remota escrava inicia a transferência dos dados recebidos no *buffer* USB para a variável "*FramePart*", que é uma variável do tipo vetor de cem posições. Após transferir os dados é feita a verificação do CRC do *frame* enviado pelo mestre. Em seguida é executada a função de "leitura( )" que envia ao mestre um *frame* estruturado de resposta com o valor do PORTA (entradas digitais).

Se o mestre solicitar a escrita de dados pela função dezesseis, o programa da Remota escrava inicia transferência dos dados recebidos no *buffer* USB para a variável "*Frame*Part". Após transferir os dados é feita a verificação do CRC do *frame* enviado pelo mestre, em seguida é executada a função "escrita" que aciona as saídas digitais do PORTB. Após o acionamento é enviado ao mestre um *frame* estruturado de resposta.

As estruturas dos *frames* enviados pelo mestre estão representados nos Quadros 6 e 7.

FRAME MESTRE *MODBUS* FUNÇÃO 3 PARA LEITURA DE 01 REGISTRO							
Endereço Unidade Escrava	Função *MODBUS*	Endereço Inicial dos registros do escravo *MODBUS*		Quantidade de Registros a serem Lidos		CRC menos significativo	CRC mais significativo
1	3	0	1	0	1	84	0A
*byte*0	*byte*1	*byte*2	*byte*3	*byte*4	*byte*5	*byte*6	*byte*7

Quadro 6 - *Frame* mestre *MODBUS* Função 3
Fonte: Autoria própria.

FRAME MESTRE *MODBUS* FUNÇÃO 16 PARA ESCRITA DE 01 REGISTRO										
Endereço Unidade Escrava	Função *MODBUS*	Endereço Inicial dos registros do escravo *MODBUS*		Quantidade de Registros a serem Escritos		Tamanho registro em *Bytes*	Dados a serem escritos		CRC menos significativo	CRC mais significativo
1	3	0	0	0	1	2	0	40	A7	A0
byte0	*byte1*	*byte2*	*byte3*	*byte 4*	*byte 5*	*byte6*	*byte 7*	*byte 8*	*byte9*	*byte10*

Quadro 7 - *Frame* mestre *MODBUS* Função 16
Fonte: Autoria própria.

3.4.6 Função de leitura;

A função para leitura de dados do protocolo *MODBUS* é executada na rotina do programa principal do microcontrolador. Sua lógica consiste em responder ao mestre os valores do PORTA (entradas digitais), e atribuir os dados que compõe o *frame* de resposta à solicitação do dispositivo mestre. O *frame* é estruturado de acordo com o padrão internacional determinado pela organização *MODBUS*-IDA, ou seja, segue o padrão explicado na seção 2.1.6.

Ao responder a função três do mestre, a unidade remota *MODBUS* envia um pacote de sete *bytes* com a leitura de dois registros. Cada *byte* contém um valor referente ao *frame* para que haja troca correta de dados entre os dois dispositivos. Veja representação do *frame* no Quadro 8.

byte1	***byte2***	***byte3***	***byte4***	***byte5***	***byte6***	***byte7***
Endereço Unidade Escrava	Função *MODBUS*	Quantidade de registros	Dados do 1° Registro		CRC Menos significativo	CRC Mais significativo

Quadro 8 - *Frame* resposta *MODBUS* função 3
Fonte: Autoria própria.

O microcontrolador responde ao mestre a quantidade de até dois *bytes*: *byte4* e *byte5* do Quadro 8. O primeiro byte contém os dados do PORTA e o segundo byte contém uma constante de valor “0”. Separando o *byte4* em *bit*s têm-se as correspondentes entradas digitais da unidade remota *MODBUS,* conforme representação no Quadro 9:

***Bit*s de um *byte*:**	*bit* 0	*bit* 1	*bit* 2	*bit* 3	*bit* 4	*bit* 5	*bit* 6	*bit* 7
Entradas Digitais:	ED0	ED1	ED2	ED3	ED4	ED5	-	-
PORTA:	RA0	RA1	RA2	RA3	RA4	RA5	RA6	RA7
***Byte*5 (1° Registro):**	*bit* 0	*bit* 1	*bit* 2	*bit* 3	*bit* 4	*bit* 5	*bit* 6	*bit* 7
Exemplo de um pacote:	1	0	1	1	1	0	0	0

Quadro 9 - *Frame* de resposta função 3
Fonte: Autoria própria.

Na última linha do Quadro 9 tem-se um exemplo do primeiro registro de dados contendo as entradas digitais ED0, ED2, ED3 e ED4 acionadas.

Nos dois últimos *bytes* do *frame* de resposta encontram-se os campos destinados ao CRC. Os cálculos dos códigos são feitos através da função “CRC()” que está explicado na seção 3.4.8.

Os *bytes* do *frame* de resposta são transferidos pelo canal USB através da função “usb_cdc_putc( )”.

No Apêndice E (programa do microcontrolador) encontra-se o código dessa função.

3.4.7 Função de escrita;

A função para escrita de dados do protocolo *MODBUS* é executada na rotina do programa principal do microcontrolador. Sua lógica consiste em escrever no PORTB os valores solicitados pelo mestre, atribuir os dados ao *frame* de resposta à solicitação do dispositivo mestre e enviá-los pelo canal USB.

O mestre solicita o acionamento das saídas digitais através de um *frame* de “Escrita de Múltiplos Registros” (função 16), conforme Quadro 7 da seção 3.4.5. O comando para acionar as saídas digitais está no *byte8* do *frame* mestre. Separando o *byte8* em *bits* têm-se as correspondentes saídas digitais da unidade remota *MODBUS*. Veja Quadro 10:

***Bit*s de um *byte*:**	*bit* 0	*bit* 1	*bit* 2	*bit* 3	*bit* 4	*bit* 5	*bit* 6	*bit* 7
Saídas Digitais:	SD0	SD1	SD2	SD3	SD4	SD5	-	-
PORTB:	RB0	RB1	RB2	RB3	RB4	RB5	RB6	RB7
***Byte*8 *frame* mestre:**	*bit* 0	*bit* 1	*bit* 2	*bit* 3	*bit* 4	*bit* 5	*bit* 6	*bit* 7
Exemplo de um pacote:	1	0	1	1	1	0	0	0

Quadro 10 - *Frame* de comando função 16
Fonte: Autoria própria.

A função para escrita transfere esse *byte* para o PORTB e aciona as saídas digitais. Na última linha do Quadro 10 tem-se um exemplo de um registro solicitando o acionamento das saídas digitais SD0, SD2, SD3 e SD4.

Após acionar as saídas é estruturado o *frame* de resposta ao mestre *MODBUS*, o escravo responde um pacote de 8 *bytes*. Veja o Quadro 11:

byte1	***byte2***	***byte3***	***byte4***	***byte5***	***byte6***	***byte7***	***byte8***
Endereço Unidade Escrava	Função *MODBUS*	Endereço Inicial		Quantidade de registros		CRC Menos significativo	CRC Mais significativo

Quadro 11 -*Frame* resposta *MODBUS* função 16
Fonte: Autoria própria.

Este *frame* é estruturado de acordo com o padrão internacional determinado pela organização *MODBUS*-IDA, ou seja, segue o padrão explicado na seção 2.1.6. Os primeiros seis *bytes* contém os mesmos dados recebidos do *frame* do mestre *MODBUS*, os dois últimos *bytes* contém o cálculo do CRC que é feito pela função "CRC( )".

Após completar todos os *bytes* do *frame*, é iniciada a função de envio de dados pela comunicação USB utilizando a função "usb_cdc_putc( )".

No Apêndice E (programa do microcontrolador) encontra-se o código da função.

3.4.8 Função CRC;

Como explicado na seção 2.1.8 o *Check Redundancy Cyclic* é um método de verificação de erros para os *frames* de comunicação. Sempre que um dispositivo envia um *frame* ele calcula o CRC do pacote de dados que está transmitindo, o dispositivo que o recebe efetua o mesmo cálculo e compara com o valor recebido. Se houver divergência o pacote é desconsiderado e identificado com uma mensagem de erro.

A função CRC desse programa é utilizada para verificar os pacotes recebidos e os pacotes enviados. Ela inicialmente confere o tamanho do *frame* a ser verificado, esse projeto trabalha com três tamanhos de mensagens: sete, oito e

onze *bytes*. Os tamanhos variam em função do dispositivo que envia o *frame* e da função *MODBUS* contida no mesmo. Os tamanhos dos bytes correspondentes aos *frames* podem ser consultados no Quadro 12.

	Função *MODBUS*	**Tamanho *frame***	**Tamanho *frame* sem CRC**
Frame* mestre *MODBUS	16	11 *bytes*	9 *bytes*
	3	8 *bytes*	6 *bytes*
Frame* escravo *MODBUS	16	8 *bytes*	6 *bytes*
	3	7 *bytes*	5 *bytes*

Quadro 12 -*Frames* para cálculo CRC
Fonte: Autoria própria.

Os campos com os códigos CRC enviados pelo mestre *MODBUS* não devem passar pela rotina de CRC, apenas o restante do *frame*. A função analisa o primeiro *byte* do *frame* comparando-o através de uma lógica OU-Exclusivo (XOR) com uma *WORD* contendo o valor FFFFH, essa variável é chamada "*checksum*". O resultado dessa comparação é deslocado em direção ao *bit* menos significativo, o *bit* menos significativo é analisado, se ele for "zero" o valor do "*checksun*" permanece como resultado da XOR deslocado, porém se esse *bit* for "um" o "*checksum*" sofre uma nova comparação XOR com o polinômio A001H, resultando em um novo valor de "*checksum*". Esse processo repete-se oito vezes, ou seja, repetem-se até os oito *bits* do *byte* serem analisados. Em seguida o "*checksum*" é comparado com o próximo *byte* do *frame*, o processo é o mesmo do *byte* anterior com sucessivas XORs até completar os oito *bits*. Esse processo repete-se em todos os *bytes* do *frame* até ser gerado um código final de CRC.

No Quadro 13 está um exemplo da execução do CRC em um *byte* com valor 01H em hexadecimal, que em código binário é representado por 00000001. Os *bit*s dos resultados das comparações são deslocados oito vezes até chegar ao resultado do CRC para esse *byte*. Esse resultado é parcial, pois esse código CRC ainda será comparado com todos os *bit*s do restante do *frame*, ou seja, se for uma mensagem com onze *bytes* todos os *bytes* serão analisados para se gerar o código final. O método de detecção de erros desenvolvido para unidade remota foi baseado nas especificações oficiais do protocolo de acordo com *MODBUS*-IDA (2002, p. 40-44).

			Mais significante								Menos Significante									
	checksum Inicial		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
	1° byte *frame* 01H		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		
	Resultado checksum		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0		
1°	Desloca para direita	Inserido	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	Ok sem mudanças
	Resultado checksum		**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**		
2°	Desloca para direita	Inserido	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Nova comparação Polinomio
	Result.Deslocamento		0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
	XOR polinômio A001H		1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1		
	Resultado checksum		**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**		
3°	Desloca para direita	Inserido	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	Ok sem mudanças
	Resultado checksum		**0**	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**		
4°	Desloca para direita	Inserido	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	Nova comparação Polinomio
	Result. Deslocamento		0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1		
	XOR polinômio A001H		1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1		
	Resultado checksum		**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**		
5°	Desloca para direita	Inserido	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	Ok sem mudanças
	Resultado checksum		**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**		
6°	Desloca para direita	Inserido	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	Nova comparação Polinomio
	Result. Deslocamento		0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1		
	XOR polinômio A001H		1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1		
	Resultado checksum		**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**		
7°	Desloca para direita	Inserido	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	Ok sem mudanças
	Resultado checksum		**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**		
8°	Desloca para direita	Inserido	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	Nova comparação Polinomio
	Result. Deslocamento		0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1		
	XOR polinômio A001H		1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1		
	Resultado checksum		**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**		
			Mais significante								Menos Significante									
	CRC para 1° byte em Hex.		80								7E									

Quadro 13 - Código CRC gerado para um *byte* de mensagem
Fonte: Autoria própria.

Para os *frames* enviados pela unidade remota ao mestre os códigos de CRC são posicionados nos últimos dois *bytes* da mensagem, um contendo o *low CRC* que é o *byte* menos significante e outro contendo o *high* CRC que é o *byte* mais significante.

3.5 SIMULAÇÕES DOS CIRCUITOS ELETRÔNICOS

O circuitos que fazem parte das placas da unidade remota *MODBUS* foram simulados através do *software* Multisim 12.0 do fabricante *National Instruments*. A partir desse *software* foi possível verificar os valores de corrente para cada circuito eletrônico, desta maneira pode-se comparar os resultados obtidos em cálculos de dimensionamento com as medições do simulador.

3.5.1 Simulação Circuito Eletrônico de Entradas Digitais;

Para o circuito de entradas digitais foi elaborado um circuito em ambiente de simulação conforme representado na Figura 23.

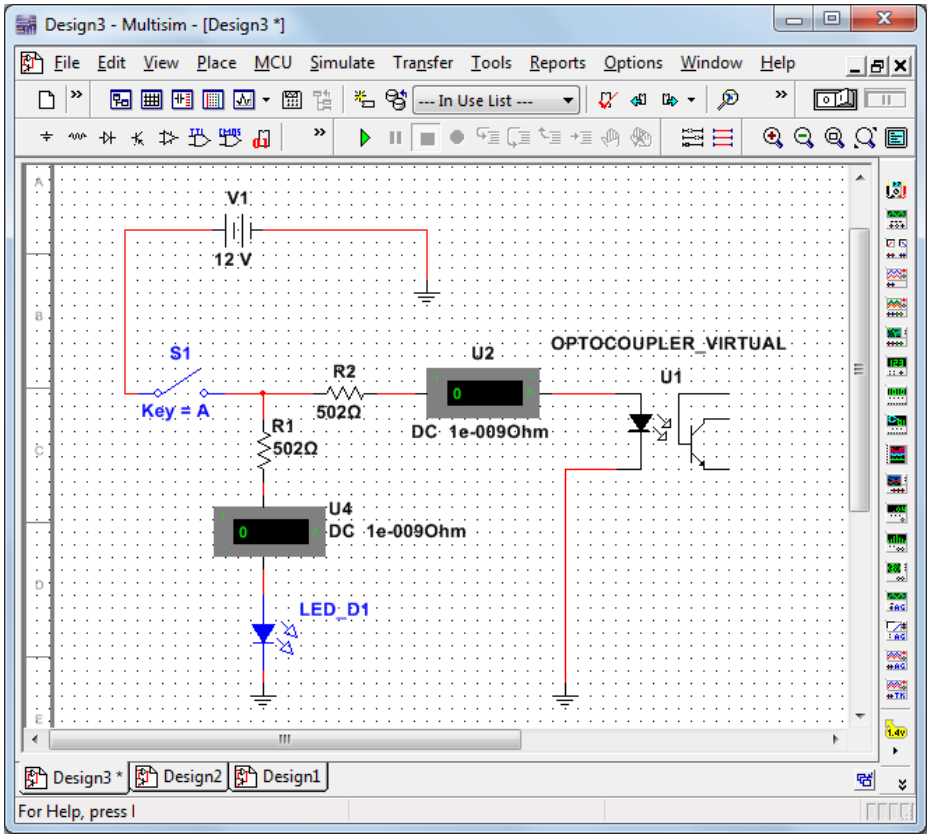

Figura 23 -Simulador Multisim Entradas Digitais
Fonte: Autoria própria.

Para o circuito de entrada digital que aciona o LED e o optoacoplador foram simuladas duas situações envolvendo os resistores R1 e R2, o *led* D1 e o optoacoplador U1. A primeira foi utilizando os resistores com os valores obtidos no cálculo de dimensionamento da seção 3.1.1 de 502Ω. A segunda foi utilizando o valor comercial do resistor de 510Ω.

Na Tabela 7 está representado o comparativo entre o valores teóricos, o valor da simulação 1 e o valor da simulação 2.

Tabela 7 - Simulação Circuito Entradas Digitais

	Valor resistor (ohms)	Corrente R1 (mA)	Corrente R2 (mA)
Cálculo teórico	502	20	20
Simulação 1	502	20	22
Simulação 2	510	20	22

Fonte: Autoria própria.

Analisando a Tabela 7, pode-se notar que nas simulações “1” e “2” o resistor R2 apresentou um consumo 10% maior que o calculado.

3.5.2 Simulação do Circuito Eletrônico de Saídas Digitais;

Para o circuito de saídas digitais foi elaborado o circuito da Figura 24 em ambiente de simulação:

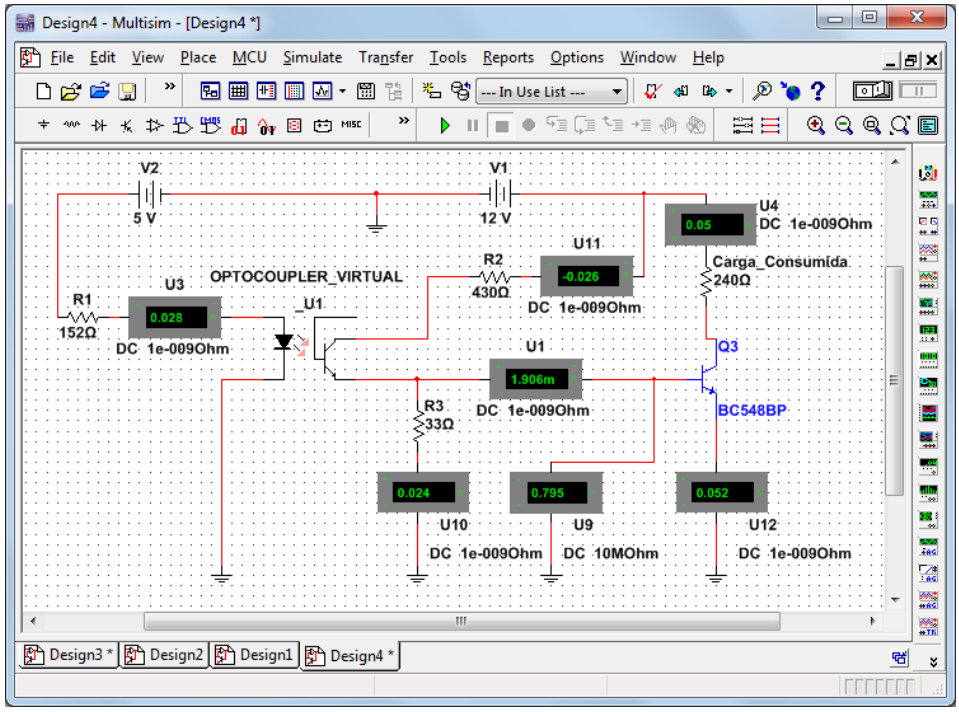

Figura 24 -Simulador Multisim Saídas Digitais 1
Fonte: Autoria própria.

Simulação foi separada em quatro etapas para confrontar os cálculos da seção 3.1.2 para os resistores R1, R2, R3 e transistor BC548.

Para o resistor R1 foram feitas três simulações:

- Simulação 1 – Resistor R1 de entrada do optoacoplador com valor 152Ω teórico;
- Simulação 2 – Resistor R1 de entrada do optoacoplador com valor 150Ω comercial;
- Simulação 3 (ajuste simulador) – O resistor R1 foi ajustado manualmente para se chegar a corrente desejada.

Na Tabela 8 estão apresentados os dados obtidos nas simulações para o resistor R1:

Tabela 8 - Simulação Resistor R1 Saídas digitais 1

	Valor resistor (ohms)	Corrente R1 (mA)
Cálculo teórico	152	20
Simulação 1	152	28
Simulação 2	150	28
Simulação 3 Ajuste Simulador	210	20

Fonte: Autoria própria.

Analisando a Tabela 8, pode-se notar que os resultados obtidos em simulador apresentaram corrente 40% maior que o cálculo teórico. O simulador não possui biblioteca com os dados de um optoacoplador real, o *software* utiliza um optoacoplador virtual. Para se obter os 20mA calculados foi necessário ajustar o resistor R1 do simulador para 210Ω.

Para o resistor R2 do divisor de tensão foram feitas duas simulações:

- Simulação 4 – O resistor R2 do divisor de tensão com valor 452Ω teórico;
- Simulação 5 – O resistor R2 do divisor de tensão com valor 430Ω comercial;

Na Tabela 9 estão apresentados os dados obtidos nas simulações para o resistor R2:

Tabela 9 - Simulação Resistor R2 Saídas Digitais 1

	Valor resistor (ohms)	Corrente R2 (mA)
Cálculo teórico	452	25
Simulação 4	452	25
Simulação 5	430	26

Fonte: Autoria própria.

Analisando a Tabela 9, pode-se notar que os resultados obtidos na simulação 4 foram idênticos aos valores teóricos. Na simulação 5 houve uma pequena variação em relação a corrente calculada, isso se deve ao fato de não existir comercialmente a venda o resistor de 452Ω.

Para o resistor R3 do divisor de tensão foram feitas duas simulações:

- Simulação 6 – Resistor R3 do divisor de tensão com valor 32Ω teórico;
- Simulação 7 – resistor R3 do divisor de tensão com valor 33Ω comercial;

Na Tabela 10 estão apresentados os dados obtidos nas simulações para o resistor R3:

Tabela 10 - Simulação Resistor R3 Saídas Digitais 1

	Valor resistor (ohms)	Corrente R3 (mA)
Cálculo teórico	32	22
Simulação 6	32	24
Simulação 7	33	24

Fonte: Autoria própria.

Analisando a Tabela 10, pode-se notar que os resultados obtidos através de simulação demonstraram uma corrente consumida 9% maior que o valor teórico calculado. Os resistores R3 e R2 são utilizados para saturar o transistor BC548 através de valores de corrente e tensão em sua base, nos testes feitos através do simulador uma corrente de 0,3mA demonstrou-se suficiente para saturar o transistor, enquanto que para os cálculos teóricos foi levado em consideração uma corrente de 2,5mA para saturação.

Reunindo os dados obtidos em simulação para os resistores R2 e R3 com os dados de simulação do transistor BC548, chegou-se aos resultados da Tabela 25. Na tabela estão apresentadas três situações:

- Valores obtidos por cálculos;
- Valores obtidos simulando-se os valores teóricos no *software* Multsim;
- Valores obtidos simulando-se os valores comerciais dos componentes no *software* Multisim;

Os dados podem ser visualizados na Tabela 11:

Tabela 11 - Simulação transistor BC548

	Resist. R2	Resist. R3	Corrente R2 (mA)	Corrente R3 (mA)	Tensão base transistor	Corrente base transistor	Corrente coletor transistor	Status saturação
Cálculo teórico	452	32	25	22	0,7	3	50	Ok
Simu. Teórico	452	32	25	24	0,778	0,3	49	Ok
Simu. Comercial	430	33	26	24	0,794	1,733	50	Ok

Fonte: Autoria própria.

Analisando a Tabela 11 pode-se notar que nas duas simulações obteve-se êxito na saturação do transistor. Porém, na simulação com os valores comerciais dos componentes a corrente na base do transistor BC548 apresentou valor elevado se comparado ao valor calculado.

3.6 VALIDAÇÃO DA UNIDADE REMOTA *MODBUS*

Após o desenvolvimento do sistema supervisório, da construção das placas e da banca de testes, foi feita a validação da unidade remota *MODBUS.* Na Figura 25 está um foto do conjunto em processo de validação.

Figura 25 - Bancada de testes
Fonte: Autoria própria.

3.6.1 Validação da placa de entradas digitais

A validação do funcionamento da placa de entradas digitais dois foi feita através de acionamento dos circuitos. Primeiramente foi alimentada a placa através de uma fonte externa de 12Vcc e posteriormente aplicada essa mesma tensão em cada um dos terminais de entrada e verificando se os LEDs emitiam luz. Veja Tabela 12:

Tabela 12 - Primeira validação da placa de entradas digitais

	Tensão aplicada	Status do Led	Validação
Entrada ED0	12Vcc	aceso	Ok
Entrada ED1	12Vcc	aceso	Ok
Entrada ED2	12Vcc	aceso	Ok
Entrada ED3	12Vcc	apagado	Falha
Entrada ED4	12Vcc	aceso	Ok
Entrada ED5	12Vcc	apagado	Falha

Fonte: Autoria própria

Foi constatado durante a validação que as entradas ED05 e ED03 não estavam funcionando devido a ruptura das trilhas da placa de circuito impresso durante a construção. A mesma teve que ser refeita com trilhas mais grossas para

garantir a integridade do circuito e o funcionamento adequado. Os testes desta segunda placa com trilhas mais grossas demonstrou a funcionalidade de todas entradas digitais, como pode ser conferido na Figura 26 e nos registros da Tabela 13:

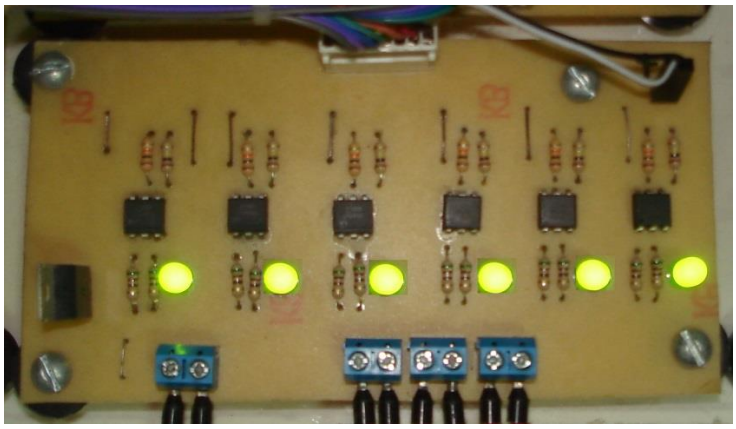

Figura 26 - Entradas digitais
Fonte: Autoria própria

Tabela 13 - Segunda validação da placa de entradas digitais

	Tensão aplicada	Status do Led	Validação
Entrada ED0	12Vcc	aceso	Ok
Entrada ED1	12Vcc	aceso	Ok
Entrada ED2	12Vcc	aceso	Ok
Entrada ED3	12Vcc	aceso	Ok
Entrada ED4	12Vcc	aceso	Ok
Entrada ED5	12Vcc	aceso	ok

Fonte: Autoria própria.

3.6.2 Validação da placa de saída digitais 1

A validação do funcionamento da placa de saídas digitais 1 foi feita através de acionamento dos circuitos das saídas. Primeiramente foi alimentada a placa com tensão de 12Vcc proveniente de uma fonte externa e posteriormente foram acionados os circuitos através do microcontrolador. Veja Tabela 14:

Tabela 14 - Primeira validação da placa de saídas digitais 1

	Tensão aplicada	Tensão conector de saída	Validação
Entrada SD0	5Vcc	12Vcc	Ok
Entrada SD1	5Vcc	12Vcc	Ok
Entrada SD2	5Vcc	0Vcc	Falha
Entrada SD3	5Vcc	12Vcc	Ok
Entrada SD4	5Vcc	0Vcc	Falha
Entrada SD5	5Vcc	0Vcc	Falha

Fonte: Autoria própria

Foi constatado durante a validação que os componentes das saídas digitais SD2, SD4 e SD5 não estavam funcionando, foi verificado que os optoacopladores estavam danificados devido a um erro durante a soldagem dos componentes. Os componentes foram trocados e soldados a placa, em novo teste realizado as saídas digitais funcionaram corretamente como registrado na Tabela 15:

Tabela 15 - Segunda validação da placa de saídas digitais 1

	Tensão aplicada	Tensão conector de saída	Validação
Entrada SD0	5Vcc	12Vcc	Ok
Entrada SD1	5Vcc	12Vcc	Ok
Entrada SD2	5Vcc	12Vcc	Ok
Entrada SD3	5Vcc	12Vcc	Ok
Entrada SD4	5Vcc	12Vcc	Ok
Entrada SD5	5Vcc	12Vcc	Ok

Fonte: Autoria própria.

3.6.3 Validação da placa de saídas digitais 2

A validação do funcionamento da placa de saídas digitais 2 foi feita através de acionamento dos circuitos. Primeiramente foi alimentada a placa com tensão de 12Vcc proveniente de uma fonte externa e posteriormente foram acionados os circuitos através do microcontrolador e da placa de saídas digitais 1. Veja Tabela 16:

Tabela 16 - Primeira validação da placa de saídas digitais 2

	Tensão aplicada	Estado do LED	Validação
Entrada SD0	12Vcc	apagado	falha
Entrada SD1	12Vcc	apagado	falha
Entrada SD2	12Vcc	apagado	falha
Entrada SD3	12Vcc	apagado	falha
Entrada SD4	12Vcc	apagado	falha
Entrada SD5	12Vcc	apagado	falha

Fonte: Autoria própria.

Foi constatado durante a validação que nenhuma das saídas digitais que das saídas estavam funcionando, foi verificado que a placa estava com erros no projeto eletrônico. O projeto foi corrigido e a placa redesenhada e reconstruída. Em novo teste realizado as saídas digitais funcionaram corretamente como registrado na Tabela 17 e na Figura 27:

Figura 27 -Validação das placa de saídas digitais
Fonte: Autoria própria.

Tabela 17 - Segunda Validação da placa saídas digitais 2

	Tensão aplicada	Estado do LED	Validação
Entrada SD0	12Vcc	Aceso	Ok
Entrada SD1	12Vcc	Aceso	Ok
Entrada SD2	12Vcc	Aceso	Ok
Entrada SD3	12Vcc	Aceso	Ok
Entrada SD4	12Vcc	Aceso	Ok
Entrada SD5	12Vcc	Aceso	Ok

Fonte: Autoria própria.

3.6.4 Validação da conexão USB

A validação da conexão USB consiste em verificar se o sistema operacional do *Windows* identifica a unidade remota *MODBUS* como um dispositivo conectado. Para isso deve-se conectar o equipamento em uma das portas USB do computador, depois, verificar no “Gerenciador de Dispositivos” se a opção “Portas (COM e LPT)” possui um dispositivo na “*Communications Port*”. Na Figura 28 está representada a conexão da unidade remota *MODBUS*:

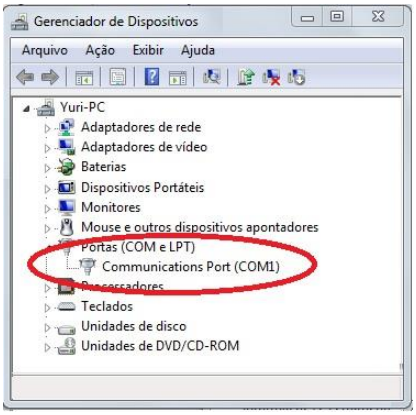

Figura 28 - Conexão Windows 7 unidade remota *MODBUS*
Fonte: Autoria própria.

Como a unidade remota utiliza o modo CDC da comunicação USB o dispositivo conecta-se através de uma porta COM virtual. Na Figura 28 pode-se notar que o dispositivo está conectado na porta de comunicação “COM1”, ele está identificado como um equipamento do fabricante *Microchip Technology, Inc.* A conexão foi estabelecida com velocidade de 9600 *bit*s por segundo, 8 *bit*s de dados, sem paridade, 1 *bit* de parada e sem controle de fluxo.

3.6.5 Validação da Comunicação *MODBUS*

Após a fase de desenvolvimento a unidade remota *MODBUS* passou por diversos testes para validação da comunicação. Esse procedimento foi feito utilizando-se *softwares* de comunicação *MODBUS*. Os *softwares* utilizados foram:

- *MODBUS* Tester v0.3;
- *MODBUS* Poll – 64 *bit*. Version 5.5.0, Build 657, 2012;

Para estabelecer a comunicação com os *softwares MODBUS* é necessário inserir no dispositivo mestre as configurações do dispositivo escravo, conforme registrado no Quadro 14:

Endereço dispositivo:	1
Tipo de dado:	Holding register
Endereço Inicial:	0
Tamanho:	1 word
Velocidade de Varredura:	1ms a 5min
Formato dos dados:	Binário

Quadro 14 - Configurações *MODBUS* unidade remota
Fonte: Autoria própria.

Após inserir as informações é possível estabelecer a comunicação entre os dispositivos. O mestre solicita a leitura das entradas digitais da unidade remota a um tempo configurável entre 1ms e 5 minutos, se a unidade remota responder corretamente o simulador informa um *status* de “read ok” em sua tela principal, e na tela *“Communication Spy”* informa uma mensagem “[*Valid Response]*”. Caso a unidade responda um dado incorreto o simulador informa um *status* “*ilegal response*” em sua tela principal, e na tela *“Communication Spy”* informa uma mensagem “[*Bad Response*]”. O simulador possui dois totalizadores em sua tela principal:

- *Valid Responses*: Número de Respostas Válidas enviadas pelo escravo;
- *Polls*: Número de solicitações enviadas pelo mestre;

A unidade remota foi testada em tempos de varredura de 1ms a 5 minutos, porém, devido a limitação da velocidade de comunicação de 9600*bit*s por segundo, as respostas tiveram variações a partir de 20ms. Nenhum pacote de dados foi perdido ou respondido errado, atribui-se a isso a alta velocidade de processamento das instruções do PIC18F2550. No teste de desempenho com tempo de varredura de 20ms a unidade remota Respondeu durante 2 minutos a uma quantidade aproximada de 902 solicitações do mestre. Na Tabela 18 estão os resultados do teste de desempenho:

Tabela 18 - Teste de Desempenho comunicação *MODBUS*

Tempo de Varredura	Tempo de Amostragem (aproximado)	Solicitações do mestre	Respostas Corretas	Eficiência
1ms	120s	983	983	100%
20ms	120s	902	902	100%
40ms	120s	828	828	100%
80ms	120s	643	643	100%
120ms	120s	519	519	100%
560ms	120s	180	180	100%
1s	120s	120	120	100%
10s	120s	12	12	100%
30s	120s	4	4	100%
1min	1min	1	1	100%
5min	5min	1	1	100%

Fonte: Autoria própria.

Os números da coluna eficiência da Tabela 18 foram determinados a partir da comparação entre as contagens de *Pools* e *Valid Responses* do simulador.

A segunda bateria de testes foi feita com acionamento de cada entrada digital e análise do *frame* recebido pelo Simulador mestre *MODBUS.* Veja Tabela 19:

Tabela 19 - Teste *frame* Entradas Digitais

Teste	*Frame* enviado pelo mestre	Entradas Acionadas	*Frame* recebido pelo mestre	Status
1	01-03-00-00-00-01-84-0A	ED0	01-03-02-**01**-00-B9-D4	Read Ok
2	01-03-00-00-00-01-84-0A	ED0, ED1	01-03-02-**03**-00-B9-D4	Read Ok
3	01-03-00-00-00-01-84-0A	ED0, ED1, ED2	01-03-02-**07**-00-B9-D4	Read Ok
4	01-03-00-00-00-01-84-0A	ED0, ED1, ED2, ED3	01-03-02-**0F**-00-B9-D4	Read Ok
5	01-03-00-00-00-01-84-0A	ED0, ED1, ED2, ED4	01-03-02-**1F**-00-B9-D4	Read Ok
6	01-03-00-00-00-01-84-0A	ED0, ED1, ED2, ED4, ED5	01-03-02-**3F**-00-B9-D4	Read Ok

Fonte: Autoria própria.

Analisando a Tabela 19, o *byte* destacado em negrito na coluna *"frame* recebido pelo mestre" corresponde as entradas digitais da unidade remota.

Na Tabela 20 está uma representação dos *bytes* de dados recebidos pelo mestre. Foram analisados os *bytes* de seis *frames* diferentes, cada um corresponde a um conjunto de entradas digitais acionadas. Os *bytes* estão convertidos em código binário, as correspondentes entradas digitais estão representadas na última linha da tabela 20.

Tabela 20 - Teste Entradas Digitais

Teste	***byte* 4**	***bit*7**	***bit*6**	***bit*5**	***bit*4**	***bit*3**	***bit*2**	***bit*1**	***bit*0**	**Status**
1	1	0	0	0	0	0	0	0	1	Read Ok
2	3	0	0	0	0	0	0	1	1	Read Ok
3	7	0	0	0	0	0	1	1	1	Read Ok
4	0F	0	0	0	0	1	1	1	1	Read Ok
5	1F	0	0	0	1	1	1	1	1	Read Ok
6	3F	0	0	1	1	1	1	1	1	Read Ok
		ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0	

Fonte: Autoria própria.

Na tabela 20 os *bits* indicados com valor "1" correspondem as entradas digitais acionadas.

O teste das saídas digitais consistiu em utilizar o simulador enviando um *byte* de informações com o acionamento de cada saída digital. A análise do *frame* de escrita das saídas digitais pode ser conferida na Tabela 21.

Tabela 21 - Teste *Frame* Saídas Digitais

***Frame* enviado pelo mestre**	**Saídas para acionar**	***Frame* recebido pelo mestre**	**Status**
01-10-00-00-00-01-02-00-**80**-A7-F0	SD0	01-10-00-00-00-01-01-C9	Wr. Ok
01-10-00-00-00-01-02-00-**C0**-A6-00	SD0, SD1	01-10-00-00-00-01-01-C9	Wr. Ok
01-10-00-00-00-01-02-00-**E0**-A7-D8	SD0, SD1, SD2	01-10-00-00-00-01-01-C9	Wr. Ok
01-10-00-00-00-01-02-00-**F0**-A6-14	SD0, SD1, SD2, SD3	01-10-00-00-00-01-01-C9	Wr. Ok
01-10-00-00-00-01-02-00-**F8**-A7-D2	SD0, SD1, SD2, SD4	01-10-00-00-00-01-01-C9	Wr. Ok
01-10-00-00-00-01-02-00-**FC**-A6-11	SD0, SD1, SD2, SD4, SD5	01-10-00-00-00-01-01-C9	Wr. Ok

Fonte: Autoria própria.

Analisando a Tabela 21, o *byte* destacado em negrito na coluna *"frame* enviado pelo mestre" corresponde as saída digitais da unidade remota.

Na Tabela 22 está uma representação dos *bytes* de dados enviados pelo mestre. Foram analisados os *bytes* de seis *frames* diferentes, cada um corresponde a um conjunto de saídas digitais a serem acionadas.

Tabela 22 - Teste Saídas Digitais

Teste	***byte* 9**	***bit*7**	***bit*6**	***bit*5**	***bit*4**	***bit*3**	***bit*2**	***bit*1**	***bit*0**	**Status**
1	80	1	0	0	0	0	0	0	0	Write Ok
2	C0	1	1	0	0	0	0	0	0	Write Ok
3	E0	1	1	1	0	0	0	0	0	Write Ok
4	F0	1	1	1	1	0	0	0	0	Write Ok
5	F8	1	1	1	1	1	0	0	0	Write Ok
6	FC	1	1	1	1	1	1	0	0	Write Ok
		SD0	SD1	SD2	SD3	SD4	SD5	SD6	SD7	

Fonte: Autoria própria.

Os *bytes* da Tabela 22 estão convertidos em código binário, as correspondentes saídas digitais estão representadas na última linha. Os *bits* indicados com valor “1” correspondem as saídas digitais acionadas.

4 CONCLUSÕES

O objetivo de se construir um dispositivo *MODBUS* a partir de recursos e ferramentas acessíveis foi concluído. Diversas disciplinas lecionadas na Universidade contribuíram para o desenvolvimento das diversas partes desse projeto: placas de circuito impresso, programação do microcontrolador, aplicação supervisório e construção da bancada de testes.

O objetivo específico de se apresentar um estudo sobre o protocolo *MODBUS* foi alcançado através de consultas às especificações oficiais do protocolo. O funcionamento do protocolo *MODBUS* pôde ser compreendido através do detalhamento dos *frames* de comunicação que são transmitidos serialmente.

O objetivo específico de desenvolver um programa em um microcontrolador do tipo PIC foi alcançado através da utilização de bibliotecas de comunicação USB e *serial* dos fabricantes *Microchip* e *CCS.* O desenvolvimento de um código baseado nas especificações oficiais do protocolo proporcionou um funcionamento correto da unidade remota como aferido na validação do equipamento.

O objetivo específico de desenvolver uma aplicação em *software* de supervisão foi alcançado através da utilização do *software Elipse E3.* A aplicação de interface desenvolvida comunicou com a unidade remota *MODBUS* e interagiu com a bancada de testes, simulando assim um pequeno processo.

O objetivo específico de se construir um protótipo através de placas de circuito impresso foi alcançado através da confecção de quatro placas eletrônicas. Através de *softwares* e simuladores pôde-se chegar a um protótipo que funcionou em conjunto com a bancada de testes.

Propostas para trabalhos futuros:

Implementar entradas e saídas analógicas na unidade remota *MODBUS* utilizando-se um microcontrolador com mais PORTs.

Implementar uma interface para configurar o endereço da unidade remota *MODBUS*, através de *software* dedicado ou de micro chaves seletoras.

Implementar uma porta de comunicação RS485 através do canal de comunicação serial.

REFERÊNCIAS BIBLIOGRÁFICAS

BORGES, Fatima; **Redes de comunicação industrial: documento técnico n°2.** Portugal: Centro de formação Schneider Electric Portugal, 2007. Disponível em: <http://www.schneiderelectric.pt/documents/productservices/training/doctecnico _redes.pdf>. Acesso em 05 jan. 2013.

CLARKE, Gordon; REYNDERS, Deon; **Modern SCADA Protocols: DNP3, IEC 60870.5 and related systems**. Great Britain: Newnes, 2004.

DUNN, Alison. The Father of Inventation: Dick Morley looks back on the 40th anniversary of the CLP. **Manufactoring Automation**, Ontario - Canadá, 2008. Disponível em: <http://www.automationmag.com/features/the-father-of-invention-dick-morley-looks-back-on-the-40th-anniversary-of-the-plc.html>. Acesso em 13 dez. 2012.

LAUGHNTON, Mike; WARNE, David. **Electrical Engineer's Reference Book.** Great Britain: Newnes, 2003.

LOPEZ, Ricardo A. **Sistema de Redes para Controle e Automação.** Rio de Janeiro: Book Express , 2002.

MICROCHIP, **USB CDC Class on an Embedded Device AN1164.** EUA: Microschip, 2008. Disponível em: <http://ww1.microchip.com/downloads/en/AppNotes/01164a. pdf>. Acesso em: 20 jan. 2013.

MICROCHIP, **PIC18F2455/2550/4455/4550 Data Sheet**. EUA: Microchip, 2009. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf>. Acesso em: 05 jan. 2012.

MODBUS-IDA. ***MODBUS over Serial Line Specification and Implementation Guide* V1.02.** EUA: MODBUS-IDA, 2002. 44 p. Disponível em: <http://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf>. Acesso em: 22 abril 2012.

MODBUS-IDA. ***MODBUS application protocol specification***. EUA: MODBUS-IDA, 2006. 51 p. Disponível em: <http://www.modbus.com/docs/Modbus_Application_Protocol_V1_1b.pdf>. Acesso em 25 abril 2012.

MODBUS-IDA. *FAQ: What is Modbus protocol?. Hopkinton: Modbus Organization*. EUA: **MODBUS-IDA**, 2005. Disponível em: <http://www.modbus.org/faq.php>. Acesso em: 20 jan. 2013.

*MODICON. **Modicon Protocol Reference Guide Ver. J.*** EUA: MODICON, 1996. Disponível em: <http://modbus.org/docs/PI_MBUS_300.pdf>. Acesso em: 14 abril 2012.

ROSÁRIO, João Maurício. **Princípios de mecatrônica.** São Paulo: Prentice Hall, 2005. 356p.

SILVEIRA, Paulo R.; SANTOS, Winderson E. **Automação e Controle**. São Paulo: Érica, 2007.

SOUZA, David J. **Conectando o PIC.** São Paulo: Mosaico Engenharia Eletrônica, 2002.

SOUZA, David J. **Desbravando o PIC.** São Paulo: Érica, 2005.

SOUZA, Vitor A. S. **Programação em C para dspic.** São Paulo: Ensino Profissional, 2008.

STRAIGHT. Talk About What is OPC. North Carolina: **Software Toolbox**, 2006. Disponível em: <http://www.opcactivex.com/What_Is_OPC/what_is_opc.html>. Acesso em: 05 maio 2012.

STRANGIO, Christopher E. Data Comunication Basics: A One-Page Introduction for each important feature. Massachussets: **Cami Research Inc,** 2012. Disponível em: <http://www.camiresearch.com/Data_ComBasics/data_com_tutorial.html#anchor205159 >. Acesso em 20 out. 2012.

STRANGIO, Christopher E. The RS232 Standard: A tutorial with signal names and definitions. Massachussets: **Cami Research Inc,** 1997. Disponível em: <http://mil.ufl.edu/4744/docs/RS232_standard_files/RS232_standard.html#anchor1154232>. Acesso em 20 out. 2012.

USB. **Universal Serial Bus Specification.** EUA: USB, 2000. Disponível em: <http://www.usb.org/developers/docs/>. Acesso em 24 jan. 2013.

APÊNDICE A - PLACA DE ENTRADAS DIGITAIS 1

Para as entradas digitais da unidade remota foi projetada uma placa de circuito impresso através do *software* ISIS 7. O circuito contém seis entradas digitais para tensão de 12VCC, sendo que cada circuito de entrada contém quatro resistores, um led e um optoacoplador. A placa ainda conta com um regulador de tensão 7805 que reduz os 12VCC da fonte para 5VCC, permitindo a alimentação do circuito do microcontrolador que fica em outra placa. Segue na Figura 29 o circuito da placa de entradas digitais e no Quadro 15 a relação de componentes utilizados.

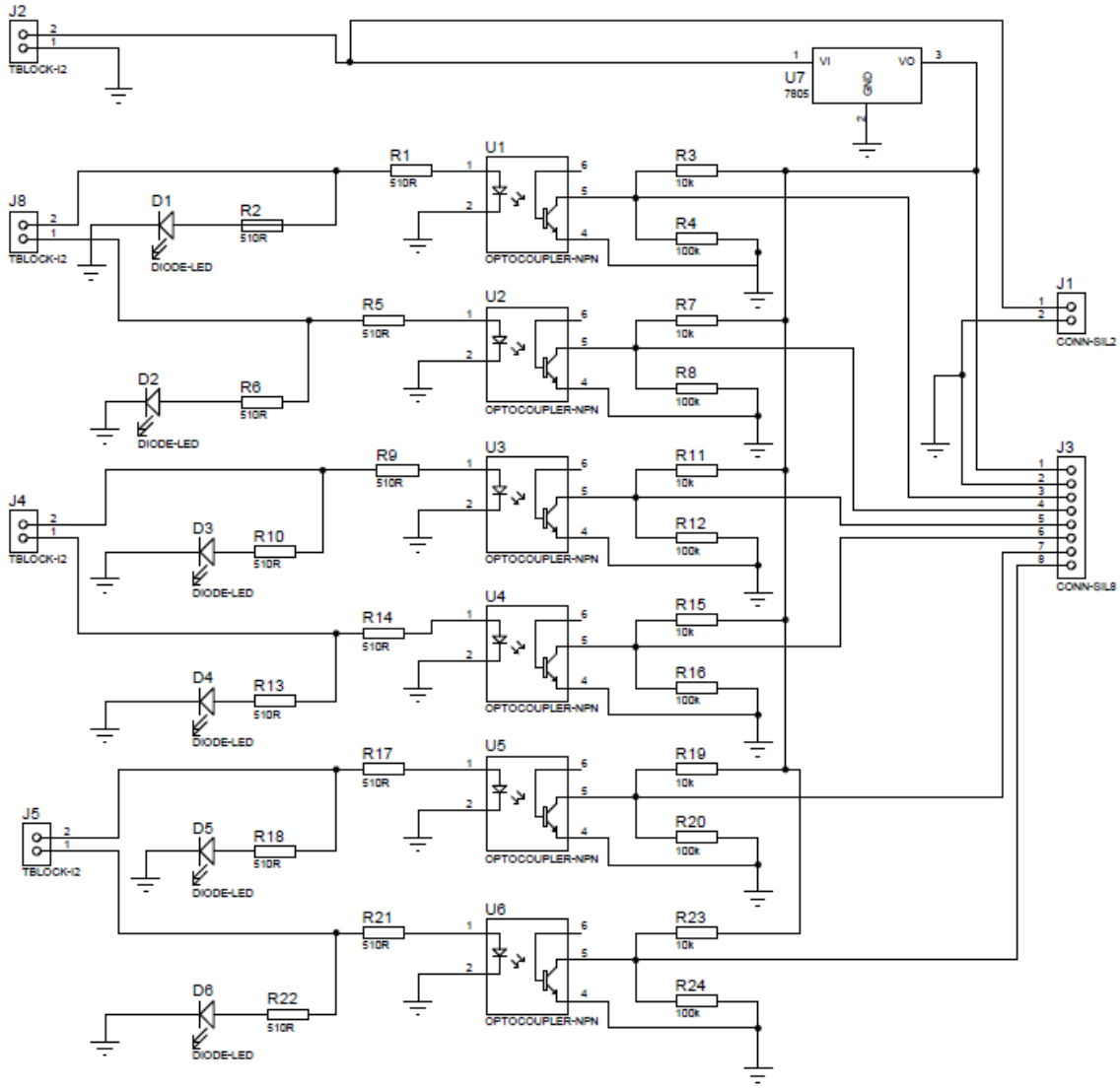

Figura 29 -Circuito Eletrônico Placa de Entradas Digitais
Fonte: Autoria própria.

Componente	Valor
R1, R2, R5, R6, R9, R10, R13, R14, R17, R18, R21, R22,	510 Ω
R3, R7, R11, R15, R19, R23	10K Ω
R4, R8, R12, R16, R20, R24	100K Ω
D1, D2, D3, D4, D5, D6	Led Verde
U1, U2, U3, U4, U5, U6	Optoacoplador 4N25
U7	Regulador de tensão 7805

Quadro 15 - Componentes da Placa de Entradas Digitais
Fonte: Autoria própria.

O *Layout* da placa com a disposição dos componentes encontra-se na Figura 30. Essa projeção foi gerada através do *software* ISIS 7 ARES, que utiliza uma biblioteca de componentes para gerar o desenho em três dimensões da placa.

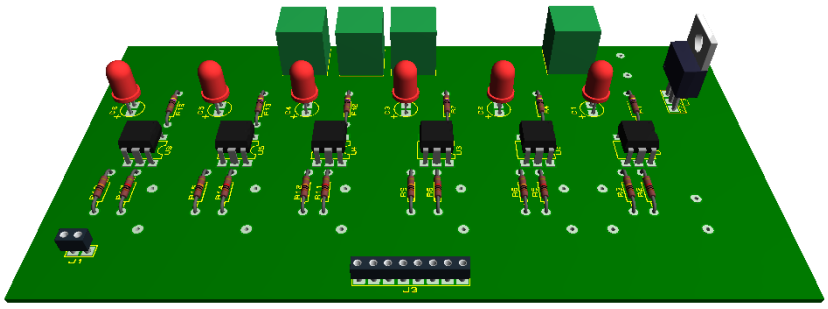

Figura 30 -*Layout* 3D Placa de Entradas Digitais
Fonte: Autoria própria.

Na Figura 31 está representado o desenho das trilhas da placa. Esse desenho é feito através do mesmo *software* ISIS 7 ARES, a disposição das trilhas correspondem aos componentes utilizados e no projeto do circuito eletrônico.

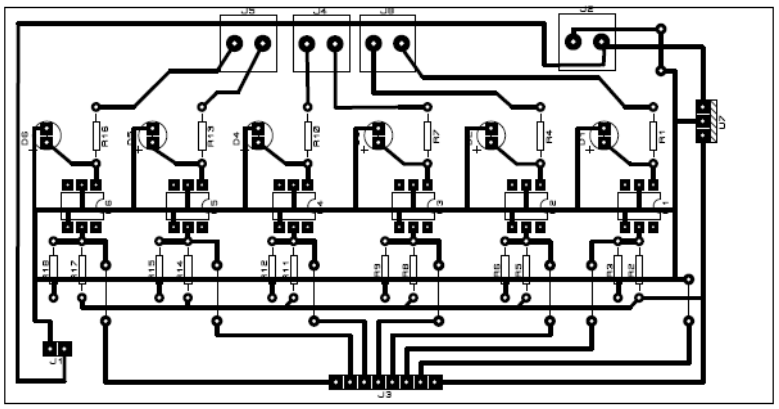

Figura 31 - *Layout* 2D Placa de Entradas Digitais
Fonte: Autoria própria.

APÊNDICE B - PLACA DO MICROCONTROLADOR

A placa do circuito do microcontrolador foi projetada através do *software* ISIS 7. Nessa placa estão instalados os componentes para o funcionamento do PIC18F2550, como o cristal oscilador e os capacitores eletrolíticos. Nesta placa também se encontra a porta de comunicação USB, que faz a conexão com o dispositivo mestre *MODBUS*. Na Figura 32 encontra-se o circuito da placa e no Quadro 16 a relação dos componentes utilizados.

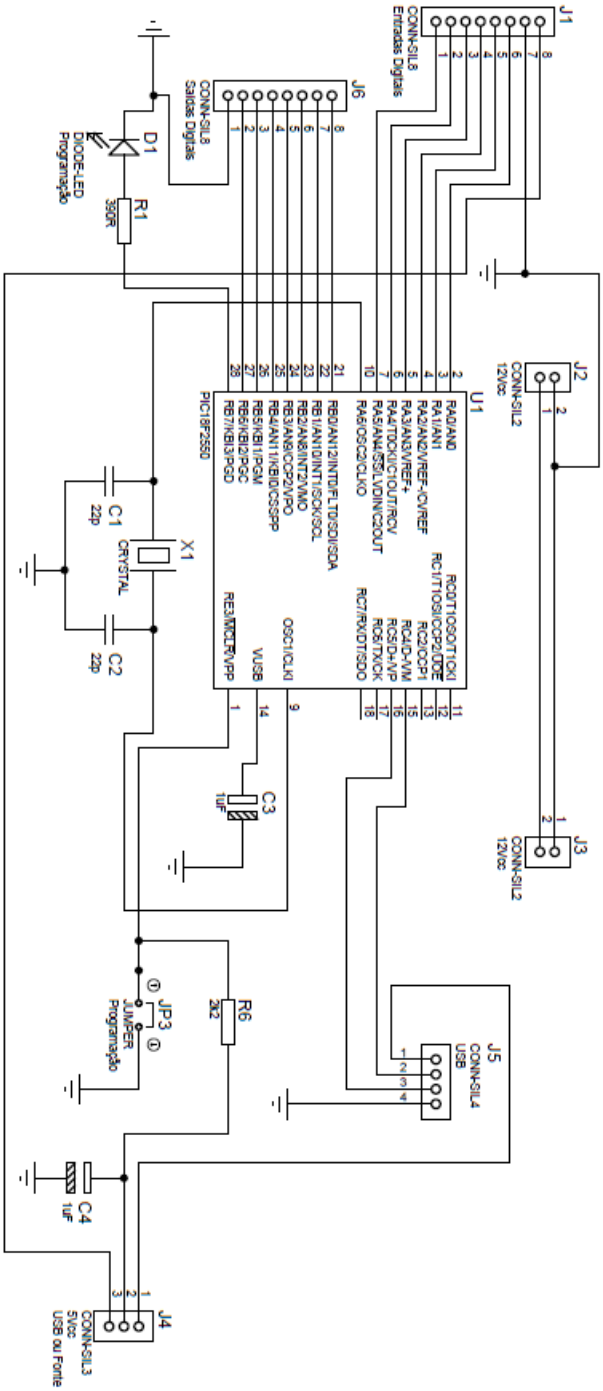

Figura 32 -Circuito Eletrônico Placa do Microcontrolador
Fonte: Autoria própria.

Componente	Valor
R1	390 Ω
R6	2K2 Ω
C1, C2	22p F
C3, C4	1u F
D1	Led vermelho
X1	Cristal 20MHz
U1	PIC 18F2550

Quadro 16 - Componentes da Placa do Microcontrolador
Fonte: Autoria própria.

Na Figura 33 está a representação 3D da placa de circuito impresso, nela pode-se ver a disposição dos componentes eletrônicos.

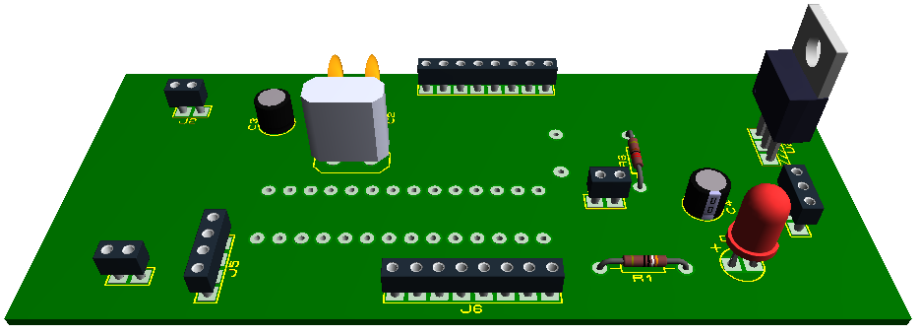

Figura 33 -Layout 3D Placa do Microcontrolador
Fonte: Autoria própria.

Na Figura 34 está o desenho das trilhas da placa de circuito impresso, geradas manualmente através do software ISIS 7 ARES. Foi utilizada a biblioteca de componentes desse *software* para realizar os furos da placa, esses furos correspondem aos pinos dos dispositivos. Através dessa ferramenta foi possível ter uma projeção das dimensões da placa com os componentes.

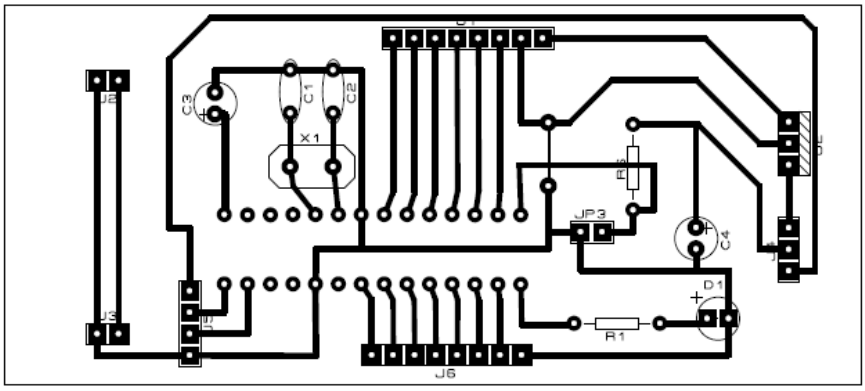

Figura 34 - Layout 2D Placa do Microcontrolador
Fonte: Autoria própria.

APÊNDICE C - PLACA DE SAÍDAS DIGITAIS

A Placa de Saídas Digitais 1 foi projetada através do *software* ISIS 7, conforme o circuito da Figura 35. Esse circuito é formado por optoacopladores que isolam as saídas do Microcontrolador, além de transistores que chaveiam o circuito de acionamento dos relés da segunda placa de saídas digitais, como explicado na seção 3.1.2.

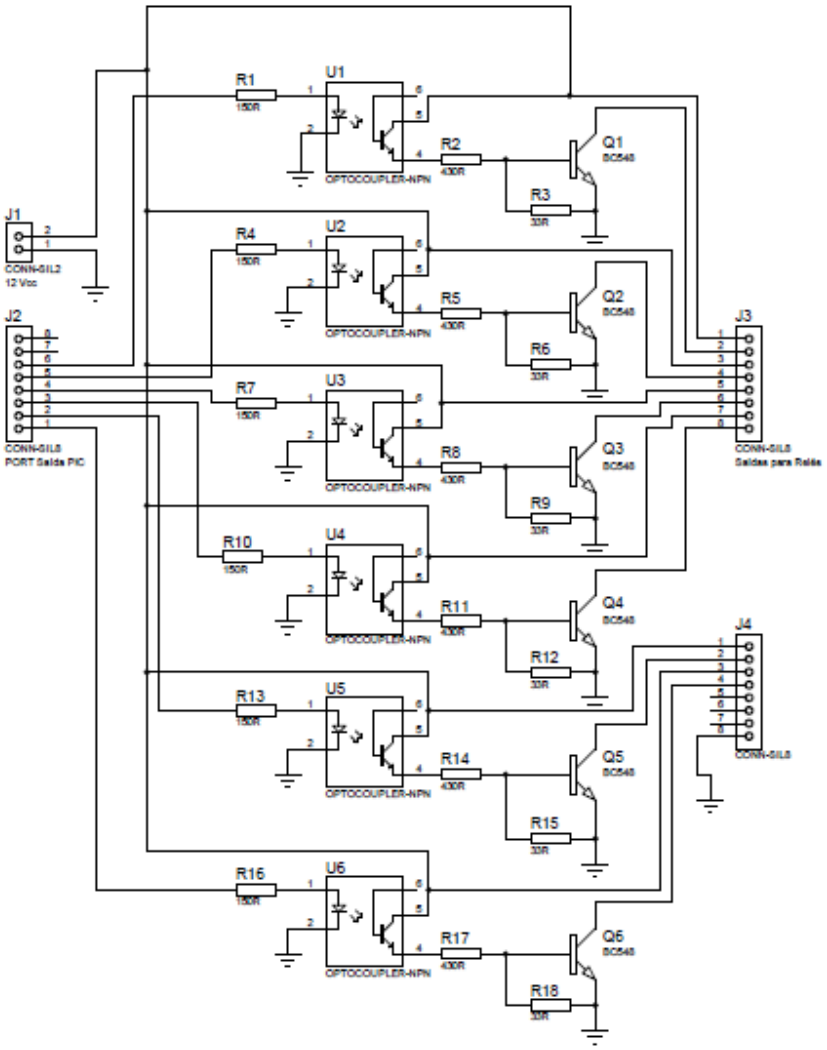

Figura 35 - Circuito Eletrônico Placa de Saídas Digitais 1
Fonte: Autoria própria.

No Quadro 17 encontra-se a relação de componentes eletrônicos utilizados nessa placa.

Componente	Valor
R1, R4, R7, R10, R13, R15	resistor 150 Ω
R2, R5, R8, R11, R14, R17	resistor 430 Ω
R3, R6, R9, R12, R15, R18	resistor 33 Ω
Q1, Q2, Q3, Q4, Q5, Q6	Transistor BC548
U1, U2, U3, U4, U5, U6	Optoacoplador 4N25

Quadro 17 - Componentes da Placa de Saídas Digitais 1
Fonte: Autoria própria.

Na Figura 36, os componentes eletrônicos juntamente com a placa de circuito impresso estão representados em uma projeção. Esse modelamento em três dimensões é gerado automaticamente pelo *software* ISIS 7 ARES, que utiliza as posições dos componentes escolhidas pelo usuário para gerar a projeção.

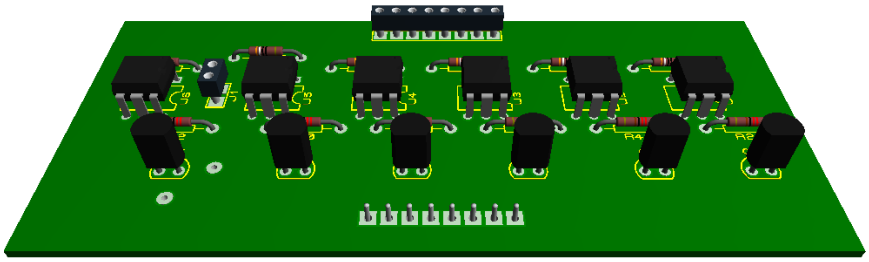

Figura 36 -Layout 3D Placa de Saídas Digitais 1
Fonte: Autoria própria.

O desenho das trilhas que interligam os componentes está na Figura 37. Pode-se notar que as trilhas possuem diferentes espessuras, isso de deve ao fato de as espessuras serem determinadas pelo próprio *software,* que sugere a utilização de trilhas maiores para os pontos de maior consumo de corrente.

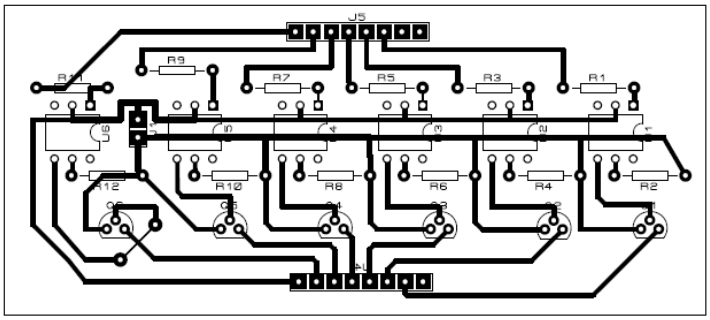

Figura 37 -Layout 2D Placas de Saídas Digitais 1
Fonte: Autoria própria.

APÊNDICE D - PLACA DE SAÍDAS DIGITAIS 2

Para a Placa de Saídas Digitais 2 foi projetado um circuito eletrônico através do *software* ISIS 7, conforme Figura 38. Esse circuito é formado por relés que acionam os equipamentos ligados as saídas digitais. Além dos relés, a placa é composta por LEDs que indicam o estado ativo das saídas digitais.

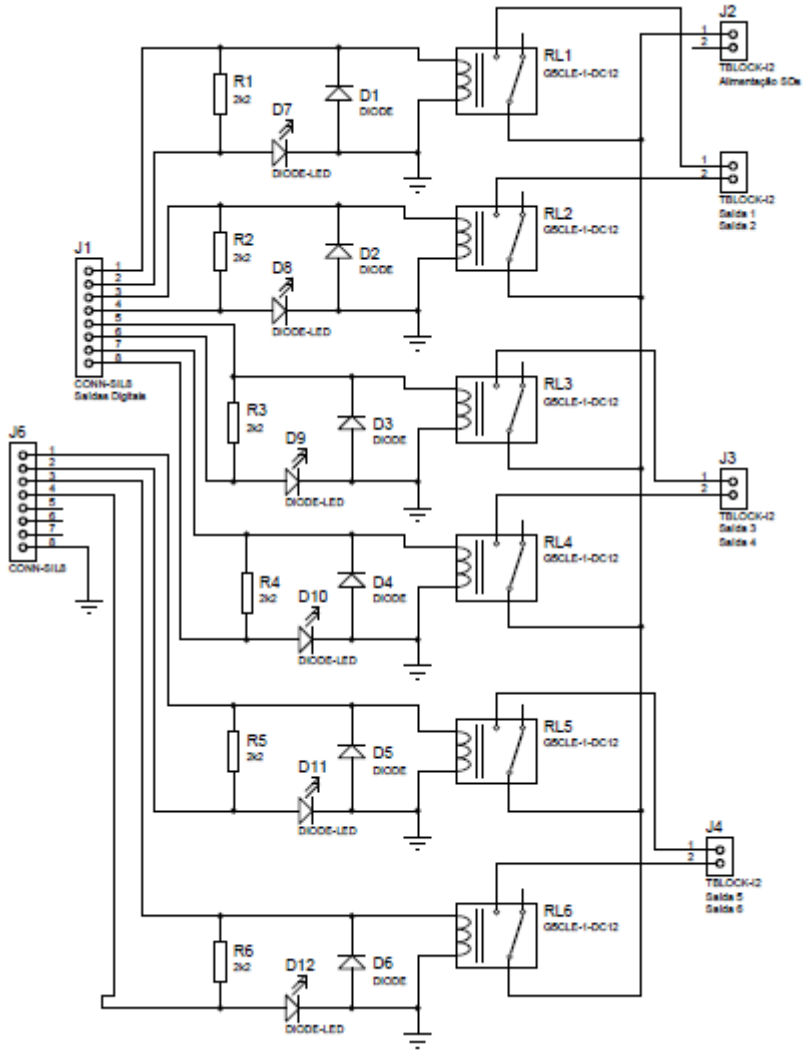

Figura 38 -Circuito Eletrônico Placa de Saídas Digitais 2
Fonte: Autoria própria.

No Quadro 18 encontra-se a relação de componentes eletrônicos utilizados nessa placa.

Componente	Valor
R1, R2, R3, R4, R5, R6	resistor 2k2 Ω
D1, D2, D3, D4, D5, D6	Diodo 1N4007 LD
D7, D8, D9, D10, D11, D12	Led Verde
RL1, RL2, RL3, RL4, RL5, RL6	Relé 12Vcc

Quadro 18 - Componentes da Placa de Saídas Digitais 2
Fonte: Autoria própria.

Na Figura 39 a placa de saídas Digitais 2 está representada através de uma projeção gerada pelo *software* ISIS 7 ARES. A placa foi construída baseando-se na projeção de três dimensões, essa ferramenta proporciona uma prévia dimensional do projeto através desse recurso visual.

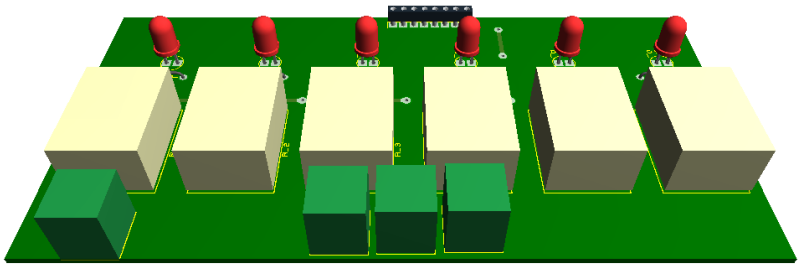

Figura 39 - Layout 3D Placa de Saídas Digitais 2
Fonte: Autoria própria.

Na figura 40 está o desenho das trilhas que interligam os componentes da placa de circuito impresso.

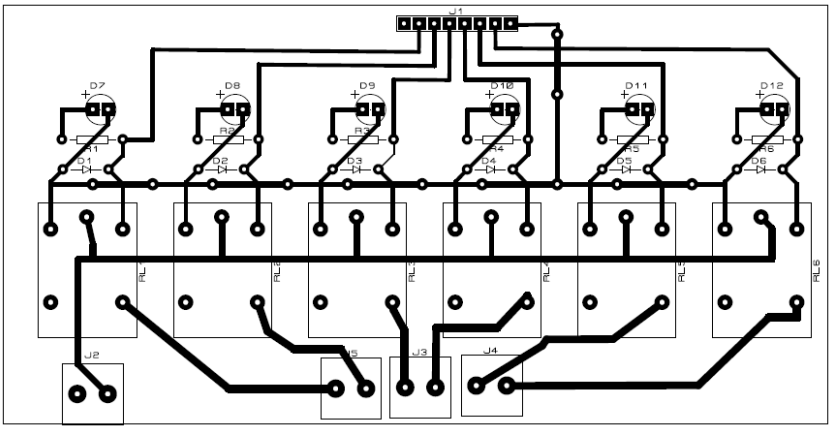

Figura 40 -Layout 2D Placas de saídas Digitais 2
Fonte: Autoria própria.

APÊNDICE E - PROGRAMA DO MICROCONTROLADOR

O programa do microcontrolador foi desenvolvido através do *software* compilador PCW do fabricante CCS. O código foi escrito em linguagem C, e utiliza-se de bibliotecas do fabricante CCS, conforme explicado na seção 3.4.3.

O código está dividido em: configurações do microcontrolador, declaração de variáveis, sub-rotina de CRC, sub-rotina de leitura, sub-rotina de escrita e rotina principal.

```
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////CONFIGURAÇÕES PIC18F2550////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
//Biblioteca 18F2550
#include <18F2550.h>

//Configuração dos Fusíveis (configuration bits):
#fuses HSPLL,PLL5, USBDIV,CPUDIV1,VREGEN,NOWDT,NOPROTECT,NOLVP,NODEBUG

//Frequência do canal USB para 48MHz
#use delay(clock=48000000)

//Definindo Espaços na memória ROM (Vetor Reset das variáveis quando reinicializado programa)
#build(reset=0x1000, interrupt=0x1000+0x08)
#org 0, 0xfff {}

//////////////////////////////////////////////////////////////////////////////////
//////////////////////////DECLARAÇÃO DAS VARIÁVEIS////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
//Declaração das variáveis utilizadas no programa principal
long int checksum = 0xffff;          // Variável para soma de verificação CRC (32 bits)
int32 FramePart[100];                // Variável 32 bits de 100 posições - Utilizada para o Frame Modbus

int TamanhoFrame;                    // Variável 8 bits - Utilizada para indicar posição do CRC no Frame
unsigned char ByteMaisCRC;           // Variável Pacote de dados byte menos significativo (8 bits)
unsigned char ByteMenosCRC;          // Variável Pacote de dados byte mais significativo (8 bits)
unsigned int Uint_1;                 // Variável Auxiliar (8 bits) - Utilizada na função CRC
unsigned int Uint_2;                 // Variável Auxiliar (8 bits) - Utilizada na função CRC

//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////SUBROTINA CRC///////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
//Função de Usuário CRC (Check Redundancy Cyclic)
void CRC (void)
{

//Executa o CRC para cada byte de acordo com o tamanho do frame
  for (Uint_1=0; Uint_1<TamanhoFrame; Uint_1++)
  {

//Aponta o byte da sequência a ser calculado, compara com checksum inicial FFFFh
    checksum = checksum^(unsigned int)FramePart[Uint_1];
```

```
//Executa o CRC em cada bit do byte apontado
     for(Uint_2=8;Uint_2>0;Uint_2--)
{

//Comparação AND: checksum & 00000000 00000001
 if((checksum)&0x0001)

//Desloca um bit direita e faz XOR com 10100000 00000001
 checksum = (checksum>>1)^0xa001;
 else
 checksum>>=1;
    }
  }
  ByteMaisCRC = checksum>>8;
  checksum<<=8;
  ByteMenosCRC = checksum>>8;
  FramePart[TamanhoFrame] = ByteMenosCRC;    // Byte menos significativo CRC
  FramePart[TamanhoFrame+1] = ByteMaisCRC;   // Byte mais significativo CRC
  checksum = 0xffff;
}

*****************************************************************************
*****************************SUBROTINA DE LEITURA*****************************
*****************************************************************************
void leitura (void)           //Função de Usuário Leitura
{
 delay_ms(5);                 //Aguarda 5ms
 FramePart[2]=0x02;           //Escreve quantidade de registros = 2 bytes (1 word)
 FramePart[3]=port_a;         //Escreve Primeiro Byte = Valor do PORTA (entradas digitais)
 FramePart[4]=0x00;           //Escreve Segundo Byte = 0
TamanhoFrame=5;               //Indica a posição do CRC. FramePart[5]->(CRC-) FramePart[6]->(CRC+)
 CRC();                       //Função calculo CRC e move-os para o FramePart[5] e FramePart[6]

printf(usb_cdc_putc,"%c%c%c%c%c%c%c",FramePart[0],FramePart[1],FramePart[2],FramePart[3],F
ramePart[4],FramePart[5],FramePart[6]);
}

*****************************************************************************
*****************************SUBROTINA DE ESCRITA*****************************
*****************************************************************************
void escrita (void)           //Função de Usuário Escrita
{
 delay_ms(5);                 //Aguarda 5ms
 TamanhoFrame = 6;            //Indica posição do CRC. FramePart[6]->(CRC-) FramePart[7]->(CRC+)
 CRC();                       //Função calculo CRC e move-os para o FramePart[6] e FramePart[7]
 port_b = FramePart[8];       //Escreve no PortB o valor enviado pelo Mestre no FramePart[8]

printf(usb_cdc_putc,"%c%c%c%c%c%c%c%c",FramePart[0],FramePart[1],FramePart[2],FramePart[3
],FramePart[4],FramePart[5],FramePart[6],FramePart[7]);
}
```

```
*****************************************************************************
*****************************PROGRAMA PRINCIPAL******************************
*****************************************************************************
//Bibliotecas utilizadas no Projeto
#include <ConfigUSB.h>        //Inclui biblioteca de configurações PIC
#include <usb_cdc.h>          //Inclui biblioteca de comunicação Serial

main() //Programa Principal
{

  usb_cdc_init();              // Inicializa CDC pela biblioteca usb_cdc.h
  usb_init();                  // Inicializa USB pela biblioteca pic18_usb.h
  usb_task();                  // Conexão USB entre PC e microcontrolador (biblioteca pic18_usb.h)
  set_tris_b(0b00000000);  // Configura PORTB para operar como saídas
  set_tris_a(0b111111);     // Configura PORTA para operar como entrada
  while(1)                     //Loop infinito


if (usb_cdc_kbhit())
    {
      FramePart[0]=usb_cdc_getc();
      if (FramePart[0]==1)                    //Verifica se o endereço é igual a 1
      {
        FramePart[1]=usb_cdc_getc();        //Verifica a função contida no byte FramePart[1]
            }
      if (FramePart[1]==16)                   //Verifica se a função é para Escrita
            {
             FramePart[2]=usb_cdc_getc();     //Frame recebido do Mestre
             FramePart[3]=usb_cdc_getc();     //Frame recebido do Mestre
             FramePart[4]=usb_cdc_getc();     //Frame recebido do Mestre
             FramePart[5]=usb_cdc_getc();     //Frame recebido do Mestre
             FramePart[6]=usb_cdc_getc();     //Frame recebido do Mestre
             FramePart[7]=usb_cdc_getc();     //Frame recebido do Mestre
             FramePart[8]=usb_cdc_getc();     //Frame recebido do Mestre
             FramePart[9]=usb_cdc_getc();     //Frame recebido do Mestre
             FramePart[10]=usb_cdc_getc();    //Frame recebido do Mestre
             escrita();                        //Executa função de Escrita
      }
      if (FramePart[1]==3)                   //Verifica se a função é para Leitura
            {
            FramePart[2]=usb_cdc_getc();      //Frame recebido do Mestre
            FramePart[3]=usb_cdc_getc();      //Frame recebido do Mestre
            FramePart[4]=usb_cdc_getc();      //Frame recebido do Mestre
            FramePart[5]=usb_cdc_getc();      //Frame recebido do Mestre
            FramePart[6]=usb_cdc_getc();      //Frame recebido do Mestre
            FramePart[7]=usb_cdc_getc();      //Frame recebido do Mestre
            leitura();                         //Executa função de Leitura
     }
    }
  }
}
```