Pontifícia Universidade Católica do Rio Grande do Sul
Escola Politécnica
Curso de Engenharia Elétrica
Sistemas Embarcados II

Prof. Juliano Benfica

Trabalho 1 – Entrega 05/05

PROTOCOLO INDUSTRIAL MODBUS

Modbus é um protocolo de comunicação de dados utilizado em sistemas de automação industrial. Criado na década de 1970 pela Modicon. É um dos mais antigos protocolos utilizados em redes de Controladores lógicos programáveis (PLC) para aquisição de sinais de instrumentos e comandar actuadores. A Modicon (atualmente parte do grupo Schneider Electric) colocou as especificações e normas que definem o Modbus em domínio público. Por esta razão é utilizado em milhares de equipamentos existentes e é uma das soluções de rede mais baratas a serem utilizadas em Automação Industrial.

O modbus utiliza o RS-232, RS-485 ou Ethernet como meios físicos. O mecanismo de controle de acesso é do tipo mestre-escravo ou Cliente-Servidor. A estação mestre (geralmente um PLC) envia mensagens solicitando dos escravos que enviem os dados lidos pela instrumentação ou envia sinais a serem escritos nas saídas para o controle dos atuadores. O protocolo possui comandos para envio de dados discretos (entradas e saídas digitais) ou numéricos (entradas e saídas analógicas).

Modbus RTU:

Neste modo os dados são transmitidos em formato binário de oito bits, permitindo a compactação dos dados em pequenos pacotes. RTU é a sigla inglesa para Remote Terminal Unit. No modo RTU, os endereços e valores podem ser representados em formato binário. Números inteiros variando entre -32768 e 32767 podem ser representados por 2 bytes. O mesmo número precisaria de quatro caracteres ASCII para ser representado (em hexadecimal) [FONTE MODBUS.ORG].

Frame de comunicação MODBUS:

ENDEREÇO	FUNÇÃO	DADOS	CRC16
8 BITS	8 BITS	N*8BITS	16BITS

Comandos padronizados MODBUS:

FUNÇÃO(HEXA)	NOME PADRÃO MODBUS	DESCRIÇÃO
0X01	READ COIL STATUS	Lê um número variável de saídas digitais
0X02	READ INPUT STATUS	Lê um número variável de entradas digitais
0X03	READ HOLDING REGISTERS	Lê um número variável de registros retentivos (saídas analógicas ou memórias)
0x04	Read input registers	Lê um número variável de registros de entrada (entradas analógicas)
0x05	Force single coil	Força uma única bobina (altera o estado de uma saída digital)
0x06	Preset single register	Preset de um único registro (altera o estado de uma saída analógica)
0x07	Read exception status	Lê exceções (registros de erro)
0x0F	Force multiple coils	Força uma quantidade variável de bobinas (saídas digitais)
0x10	Preset multiple registers	Preset de uma quantidade variável de registros (saídas analógicas)

Códigos de erro MODBUS:

CÓDIGO DE ERRO	DESCRIÇÃO
0X01	Código defunção não válido
0X02	Endereço de registro não válido
0X03	Variáveis com valores errados (Ex. Overflow)
0X04	Erro de CRC
0X06	TIME OUT - Após feita uma transmissão espera-se até 100ms pela reposta, se não volta ao estado inicial para nova recepção. A cada byte recebido pela serial deverá ter um TIME OUT de 20ms, ou seja se não receber outro byte em até 20ms, uma mensagem de erro deve ser enviada.

Frame de Comunicação via MODBUS:

ENDEREÇO	FUNÇÃO	DADOS				CRC	
		ENDEREÇO INICIAL		NÚMERO DE ESTADOS		CRC ALTO	CRC BAIXO
BYTE	BYTE	BYTE ALTO	BYTE BAIXO	BYTE ALTO	BYTE BAIXO	BYTE ALTO	BYTE BAIXO

Exemplo de comunicação para a função 0x01:

Deseja-se ler o status das saídas digitais de 2 à 11 (10 estados) do escravo 17 (0x11).

MASTER -> SLAVE

ENDEREÇO	FUNÇÃO	DADOS				CRC	
		ENDEREÇO INICIAL		NÚMERO DE ESTADOS		CRC ALTO	CRC BAIXO
0x11	0x01	0x00	0x01	0x00	0x0A	BYTE ALTO	BYTE BAIXO

SLAVE -> MASTER

ENDEREÇO	FUNÇÃO	DADOS			CRC	
		NÚMERO DE BYTES	ESTADO1	ESTADO2	CRC ALTO	CRC BAIXO
0x11	0x01	0x02	0x11	0x01	BYTE ALTO	BYTE BAIXO

0X11H=00010001B: Saídas 6,2 ON; Saídas 9,8,7,5,4,3 OFF

Exemplo de comunicação para a função 0x02:

Deseja-se ler o status das entradas digitais de 4 à 17 (14 estados) do escravo 17 (0x11).

MASTER -> SLAVE

ENDEREÇO	FUNÇÃO	DADOS				CRC	
		ENDEREÇO INICIAL		NÚMERO DE ESTADOS		CRC ALTO	CRC BAIXO
0x11	0x02	0x00	0x03	0x00	0x0D	BYTE ALTO	BYTE BAIXO

SLAVE -> MASTER

ENDEREÇO	FUNÇÃO	DADOS			CRC	
		NÚMERO DE BYTES	ESTADO1	ESTADO2	CRC ALTO	CRC BAIXO
0x11	0x02	0x02	0X2D	0X3C	BYTE ALTO	BYTE BAIXO

2DH=00101110B: Entradas 9,7,6,5 ON; Entradas 11,10,8,4 OFF

3CH=00111100B: Entradas 17,16,15,14 ON; Entradas 13,12 OFF

Exemplo de comunicação para a função 0x03:

Deseja-se ler um registrador de 32bits que contém um número float no registrador 107 e 108 do escravo 17 (0x11).

MASTER -> SLAVE

ENDEREÇO	FUNÇÃO	DADOS				CRC	
		ENDEREÇO INICIAL		NÚMERO DE ESTADOS		CRC ALTO	CRC BAIXO
0x11	0x03	0x00	0x6B	0x00	0x02	BYTE ALTO	BYTE BAIXO

SLAVE -> MASTER

ENDEREÇO	FUNÇÃO	DADOS					CRC	
		NÚMERO DE BYTES	BYTE 1	BYTE 2	BYTE 3	BYTE 4	CRC ALTO	CRC BAIXO
0x11	0x03	0x04	0XCC	0XCD	0X42	0X8D	BYTE ALTO	BYTE BAIXO

```
//Union para separar um dado de 32 bits em 4 dados de 8 bits para
transmissão pela serial
typedef union
   {
      struct
      {
         unsigned char high;
         unsigned char low;
         unsigned char high1;
         unsigned char low1;
      }parcial;
      float total;
   }INTEIRO;

INTEIRO dado16;
unsigned char buffer[4];
dado16.total = numero_float;
buffer[0] =    dado16.parcial.low;
buffer[1] =    dado16.parcial.high;
buffer[2] =    dado16.parcial.low1;
buffer[3] =    dado16.parcial.high1;
```

Exemplo de comunicação para a função 0x0F:

Deseja-se setar as saídas de 29 a 45 do escravo 17 (0x11) - Como são 17 estados cabe em 3 bytes.

MASTER -> SLAVE

ENDEREÇO	FUNÇÃO	DADOS								CRC	
		ENDEREÇO INICIAL		NÚMERO DE ESTADOS		Nº BYTES	BYTE1	BYTE2	BYTE3	CRC ALTO	CRC BAIXO
0x11	0x0F	0X00	0X1D	0X00	0X11	0X03	0XAC	0X38	0X01	BYTE ALTO	BYTE BAIXO

SLAVE -> MASTER

ENDEREÇO	FUNÇÃO	DADOS				CRC	
		ENDEREÇO INICIAL		NÚMERO DE ESTADOS		CRC ALTO	CRC BAIXO
0x11	0X0F	0x00	0X1D	0x00	0X11	BYTE ALTO	BYTE BAIXO

ACH=10101100B: Saida 36,34,32,31 ON; Saida 35,33,30,29 OFF

38H=00111000B: Saida 42,41,40 ON; Saida 44,43,39,38,37 OFF

01H=00000001B: Saida 45 ON

Exemplo de comunicação para a função 0x10:

Deseja-se modificar 2 registradores, o 302 e o 303 do escravo 17 (0x11).

MASTER -> SLAVE

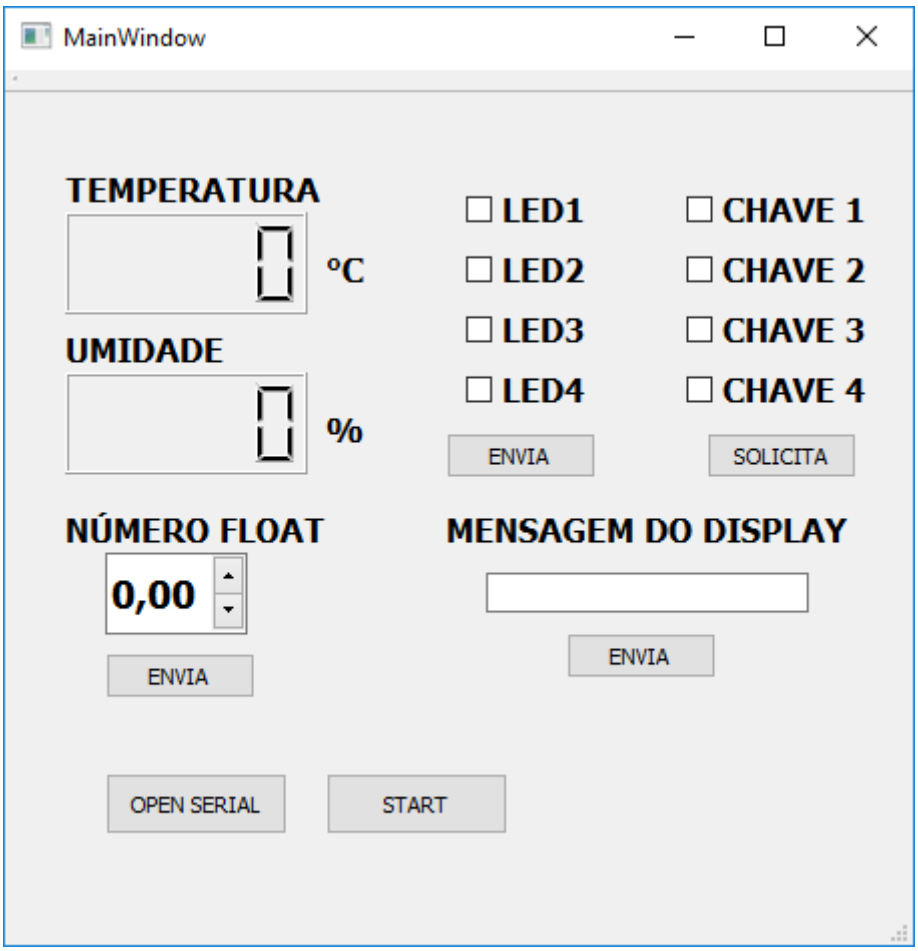
ENDEREÇO	FUNÇÃO	DADOS									CRC	
		ENDEREÇO INICIAL		NÚMERO DE ESTADOS		Nº BYTES	BYTE1	BYTE2	BYTE3	BYTE4	CRC ALTO	CRC BAIXO
0x11	0x10	0x01	0x2D	0X00	0x02	0x04	0x00	0x0A	0x0B	0x0C	BYTE ALTO	BYTE BAIXO

SLAVE -> MASTER

ENDEREÇO	FUNÇÃO	DADOS				CRC	
		ENDEREÇO INICIAL		NÚMERO DE ESTADOS		CRC ALTO	CRC BAIXO
0x11	0x10	0x01	0x2D	0X00	0x02	BYTE ALTO	BYTE BAIXO

OBJETIVO:

- Criar uma biblioteca modbus.c e modbus.h para utilização genérica.
- Controlar o kit NÚCELO CORTEX F4 via serial com protocolo MODBUS.
- **Implementar uma interface em Qt que faça a comunicação em MODBUS e exiba os dados e faça os controles solicitados abaixo**.



O QUE DEVE SER IMPLEMENTADO:

- Tudo deverá ser em simulação usando o Software Proteus em conjunto com o
- ModbusPoll e QT. O microcontrolador a ser utilizado será o será o ATMEGA328 através do kit Arduino Uno disponível no Proteus para simulação.
- O endereço do escravo vai ser o mesmo do número do grupo no Moodle;
- Cada frame transmitido deve ser gerado o CRC e comparado com o CRC enviado na mensagem. Se erro, enviar frame de erro de CRC;
- Ligar e desligar 4 leds ligados no kit usando o comando 0x0F. O endereço deste periférico é 0x01. Este valor será enviado pelo Qt.

- Ler o status de 4 switches usando o comando 0x02. O endereço deste periférico é 0x02. Este valor será lido pelo Qt.
- Escrever uma mensagem de 1 a 16 caracteres na linha 1 do LCD (shield LCD) usando o comando 0x10. O endereço deste periférico é 0x00. Este valor será enviado pelo Qt
- Ler a temperatura usando o sensor SHT15 em ponto flutuante (float) usando o comando 0x03. O endereço deste periférico é 0x04. Este valor será lido pelo Qt.
- Ler a umidade usando o usando o sensor SHT15 em ponto flutuante (float) usando o comando 0x03. O endereço deste periférico é 0x05. Este valor será lido pelo Qt.
- Ler um valor em ponto flutuante (float) usando o comando 0x10. O endereço deste periférico é 0x06. Este valor será enviado pelo Qt e recebido pelo Arduino no Proteus e apresentado no LCD este valor.
- Sugere-se criar uma máquina de estados (FSM) para implementar o protocolo.

VALIDAÇÃO:

O protocolo implementado poderá ser validado e testado antes da implementação no Qt utilizando o software de comunicação ModBusPoll disponível no Moodle.

AVALIAÇÃO:

ITEM	NOTA
Implementação do Qt	2,0 Pontos
Comparação de CRC	1,0 Ponto
Acionamento dos LEDS pelo protocolo	1,0 Ponto
Leitura das Chaves pelo protocolo	1,0 Ponto
Escrita da MSG no LCD pelo Protocolo	1,5 Ponto
Escrever valor FLOAT no LCD pelo protocolo	0,5 Ponto
Leitura da Temperatura pelo Protocolo	1,5 Ponto
Leitura da Umidade pelo Protocolo	1,5 Ponto

FUNÇÃO DE GERAÇÃO DE CRC16:

```
unsigned short CRC16 (unsigned char *puchMsg, unsigned short usDataLen) /* The function returns the
CRC as a unsigned short type */
{
    unsigned char uchCRCHi = 0xFF ; /* high byte of CRC initialized */
    unsigned char uchCRCLo = 0xFF ; /* low byte of CRC initialized */
    unsigned uIndex ; /* will index into CRC lookup table */
    while (usDataLen--) /* pass through message buffer */
    {
        uIndex = uchCRCLo ^ *puchMsg++; /* calculate the CRC */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex];
        uchCRCHi = auchCRCLo[uIndex] ;
    }
    return (uchCRCLo << 8 | uchCRCHi);
}

/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;


/* Table of CRC values for low-order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
};
```

OBS: As soluções de contorno para funcionamento apropriado do projeto deverão ficar a cargo de cada grupo. LEMBRANDO QUE ESSAS SÃO AS FUNCIONALIDADES MÍNIMAS PARA FUNCIONAMENTO. A CRIATIVIDADE E ACRESCIMOS DE FUNÇÕES AO PROJETO É SEMPRE BEM VINDO.

O código em C **E DEVERÁ SER ENTREGUE NO MOODLE.** No dia da apresentação o código será conferido e também serão efetuadas perguntas sobre o desenvolvimento.