FACULDADE DE TECNOLOGIA SENAI CIMATEC
ESPECIALIZAÇÃO EM AUTOMAÇÃO CONTROLE E ROBÓTICA

VICTOR OTTAN SOARES DE SOUZA LIMA

APLICAÇÃO EM REDE MODBUS UTILIZANDO ARDUINO

Salvador
2015

VICTOR OTTAN SOARES DE SOUZA LIMA

APLICAÇÃO EM REDE MODBUS UTILIZANDO ARDUINO

Trabalho de conclusão de curso apresentado à Faculdade de Tecnologia SENAI CIMATEC como requisito parcial para a obtenção do Título de Especialista em Automação Controle e Robótica

Professor Orientador: Me. Oberdan Rocha Pinheiro

Salvador
2015

Ficha catalográfica elaborada pela Biblioteca da Faculdade de Tecnologia SENAI CIMATEC

L732a Lima, Victor Ottan Soares de Souza

Aplicação em rede modbus utilizando arduino / Victor Ottan Soares de Souza. – Salvador, 2015.

40 f.: il. color.

Orientador: ME. Oberdan Pinheiro Rocha.

Monografia (Especialização em automação controle e robótica) – Programa de Pós-Graduação, Faculdade de Tecnologia SENAI CIMATEC, Salvador, 2015.
Inclui referências.

1. Modbus. 2. Redes Industriais. 3. Arduino. I. Faculdade de Tecnologia SENAI CIMATEC. II. Rocha, Oberdan Pinheiro. III. Título.

CDD: 629.8

VICTOR OTTAN SOARES DE SOUZA LIMA

APLICAÇÃO EM REDE MODBUS UTILIZANDO ARDUINO

Trabalho de conclusão de curso apresentado à Faculdade de Tecnologia SENAI CIMATEC como requisito parcial para a obtenção do Título de Especialista em Automação Controle e Robótica

Aprovada em 24 de Julho de 2015

Banca Examinadora

Oberdan Pinheiro Rocha – Orientador _____
Mestre em Modelagem Computacional e Tecnologia Industrial pelo SENAI – Departamento Regional da Bahia, Salvador, Brasil
Faculdade de Tecnologia SENAI CIMATEC

Milton Bastos de Souza – Membro da Banca _____
Mestre em Engenharia Elétrica pela Universidade Federal da Bahia, Salvador, Brasil
Faculdade de Tecnologia SENAI CIMATEC

24 de Julho de 2015

Aos meus pais, noiva e família.

AGRADECIMENTOS

Aos meus pais, Sonia e José Carlos, por terem me apoiado desde o começo com meu trabalho, estadia em salvador e apoio em geral.
A minha noiva Liliane por ter me ajudado e compreendido todos os momentos que precisei estar ausente para redigir a monografia.
Ao professor, orientador e coordenador Oberdan Rocha Pinheiro pela ajuda em todos os momentos que precisei.
Ao colega Edeilson pela ajuda com a escolha do tema e pelo apoio na confecção do protótipo, além do colega Eduardo pelo companheirismo durante o curso.

“A persistência é o caminho do êxito”

Charles Chaplin

RESUMO

No cenário de redução de gastos e otimização de processos que está o mundo industrial atualmente é importante pensar em soluções que possam baratear tecnologias que são amplamente utilizadas. Assim esse trabalho tem como objetivo aplicar uma solução de transmissão de dados em rede modbus, protocolo bastante difundido na indústria, de forma mais barata. Para tal, foram utilizados dois arduinos sendo um configurado como mestre e o outro como escravo, representando respectivamente o controlador e os sensores envolvidos no processo. No arduino escravo foi conectado dois sensores de temperatura e um de luminosidade a fim de verificar a eficácia de transmissão de diversas informações diferentes na rede de comunicação. Com a configuração da rede e a montagem do protótipo foi verificado sua eficácia e o presente estudo é uma contribuição para pesquisas futuras de unir a tecnologia do arduino com a de CLPs já consolidados no mercado.

Palavras-chave: Modbus. Redes Industriais. Arduino.

ABSTRACT

In the cost reduction scenario and process optimization which the industrial world is now it is important to think about solutions that can abate technologies that are widely used. Thus this work aims to apply a data transmission solution in Modbus network, which is widespread in the industry protocol, more cheaply. Thereunto, we used two Arduino being configured as master and the other as slave, respectively representing the controller and the sensors involved in the process. The Arduino slave has been connected with two temperature sensors and a brightness sensor in order to verify the effectiveness of the transmission of different information in the communication network. With the network configuration and assembly of the prototype was verified its effectiveness and this study is a contribution to future research to unite technology Arduino with the PLCs already consolidated in the market.

Key words: Modbus. Industrial Network. Arduino.

SUMÁRIO

1 INTRODUÇÃO

Durante o passar dos tempos a indústria vem aperfeiçoando e otimizando processos para se destacar no mercado e maximizar seus lucros. Nesse contexto, o controle e supervisão de processos ocupa um espaço significativo, sendo este necessário para o devido gerenciamento de quantidade de insumo gasta, energia consumida e produtos fabricados.

Sendo assim, a integração do chão de fábrica com os operadores e coordenadores se torna cada vez mais crítica e por consequência, necessitando de uma rede industrial mais eficiente e também mais robusta.

Como atualmente a tendência é de corte e redução de gastos foi pensada, dentro do contexto apresentado, uma aplicação de rede de comunicação industrial utilizando a premissa de baixo custo e alto valor agregado. Dessa maneira, o objetivo do trabalho foi em utilizar uma plataforma de programação barata e de fácil manipulação para transmitir dados do chão de fábrica utilizando um protocolo usado em âmbito industrial. Com esta premissa, foi escolhida a plataforma arduino para comunicar em protocolo modbus, bastante utilizado na indústria.

Para demonstrar a aplicação em si, foi realizada uma rede com dois arduinos, sendo um configurado como escravo, representando o sensoriamento de chão de fábrica e o outro como mestre representando o controlador que recebe as informações provenientes da linha de produção. Foram utilizados sensores de temperatura e luminosidade no escravo para simular a informação vinda do chão de fábrica e um controle remoto a fim simular o operador que requisita a informação da linha.

2 REDES INDUSTRIAIS E TIPOS DE PROTOCOLOS

A fim de efetuar a transmissão das informações no ambiente industrial a formação de uma rede é fundamental para essa finalidade. Esses dados trafegam desde o “chão de fábrica” até o meio corporativo com o objetivo de integrar todos esses ambientes da empresa.

Segundo Nogueira (2009, p. 17) com o crescimento dos dispositivos utilizados em uma rede industrial, foi necessário criar padrões de comunicação para conectar os mesmos de uma forma padronizada. De acordo com Mendes (2007, p.18) no início da concepção de redes, cada fabricante possuía seu padrão de comunicação, o que inviabilizava a utilização de equipamentos de fabricantes diferentes em uma mesma rede, e tornava a manutenção da rede muito cara, já que a substituição por equipamentos de outros fabricantes deveria ser integral.

Nesse contexto, a ISO (*International Organization for Standardization*) criou um padrão para a troca de informações entre e em uma rede, chamado de modelo de referência OSI (*Open Systems Interconnection*).

2.1 MODELO OSI

O modelo OSI segue a estruturação de comunicação de dados em multicamadas (7 ao todo). O exemplo dessa estrutura pode ser visto na figura 1.

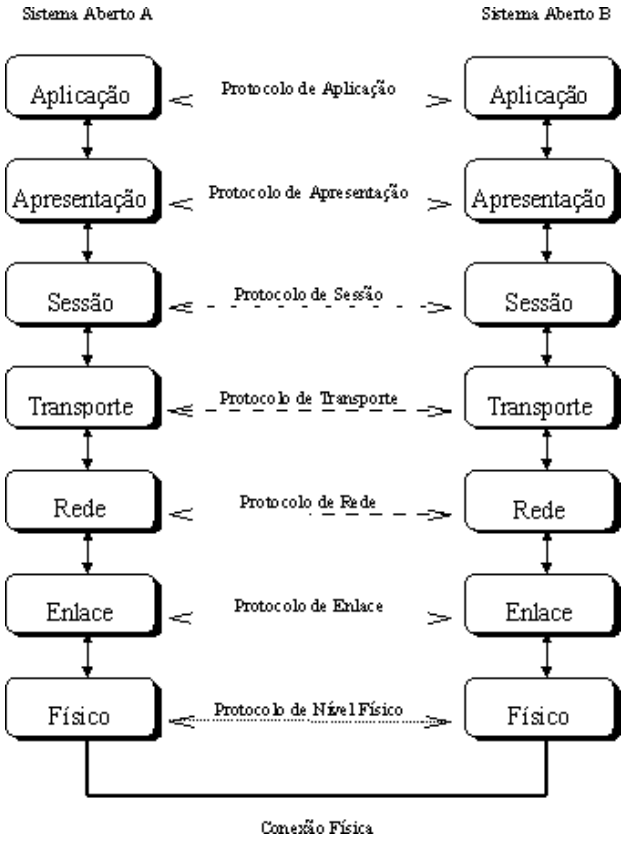

Figura 1: Representação do modelo OSI fonte: http://www.oocities.org/siliconvalley/network/7460/osi.htm

Segundo a CISCO CNNA, "Neste modelo, as informações são passadas de uma camada para a outra, começando na camada de Aplicação no host transmissor, continuando hierarquia abaixo, até a camada Física, passando para o canal de comunicações até o host de destino, onde a informação retorna hierarquia acima, terminando na camada de Aplicação".

De forma resumida, as camadas são:

- **Camada Física:** é responsável pela coordenação do fluxo de *bits* através do meio físico (FOROUZAN, 2008, p33);
- **Camada de Enlace de Dados:** assegura que o conteúdo que chega da camada física será seguro para transmissão para o destino (NOGUEIRA, 2009, p.19);
- **Camada de Rede:** responsável pela entrega do pacote da origem até seu destino final, garantindo a sua entrega (FOROUZAN, 2008, p.36);
- **Camada de Transporte:** supervisiona todo o fluxo de comunicação da rede, garantindo que a mensagem chegue intacta e na sequência correta (FOROUZAN, 2008, p. 44);
- **Camada de Sessão:** responsável por estabelecer uma sessão entre dois ou mais sistemas, controlando o diálogo (TANENBAUM, 2003, p. 47);

- **Camada de Apresentação:** segundo Tanenbaum (2003, p. 47), essa camada "gerencia as estruturas de dados abstratas e permite a definição e o intercâmbio de estruturas de dados de nível mais baixo";
- **Camada de Aplicativo:** gera uma interface ao usuário, permitindo que o mesmo acesse a rede (FOROUZAN, 2008,p. 41);

2.2 PRINCIPAIS PROTOCOLOS

"Um protocolo é um conjunto de regras que controla a comunicação de dados" (FOROUZAN, 2008,p. 19). Em redes industriais existem diversos protocolos que atuam nas diversas camadas do padrão OSI, sendo alguns dos mais utilizados o Modbus, Ethernet Industrial, TCP/IP, Device Net, entre outros.

2.2.1 TCP/IP

Segundo Miller (2009, p. 3) O conjunto de protocolos TCP/IP é responsável por prover comunicação entre computadores. O protocolo da internet (IP) é o mais importante de todos os protocolos e o TCP (*Transmission Control Protocol*), também não deixa de ter uma importante participação na transmissão de dados (MILLER, 2009, p. 3).

O TCP/IP foi incialmente desenvolvido a partir do modelo ARPA (*Advanced Research Project Agency*) pelo o departamento do governo dos Estados Unidos de mesmo nome e junto com o modelo OSI, segundo Reynders (2005, p.112), são utilizados como base para infraestrutura de comunicação além de prover abstrações da mesma realidade.

O conjunto de protocolos TCP / IP, não se limita aos protocolos TCP e IP, mas é composto por uma multiplicidade de protocolos inter-relacionados, que ocupam as três camadas superiores do modelo ARPA (REYNDERS, 2005,p. 112).

Conforme ilustrado na figura 2, um *frame* de transmissão IP de origem em um host específico (computador) iria conter um cabeçalho da rede local (por exemplo, Ethernet) aplicável a esse host (Reynders, 2005, p. 112).

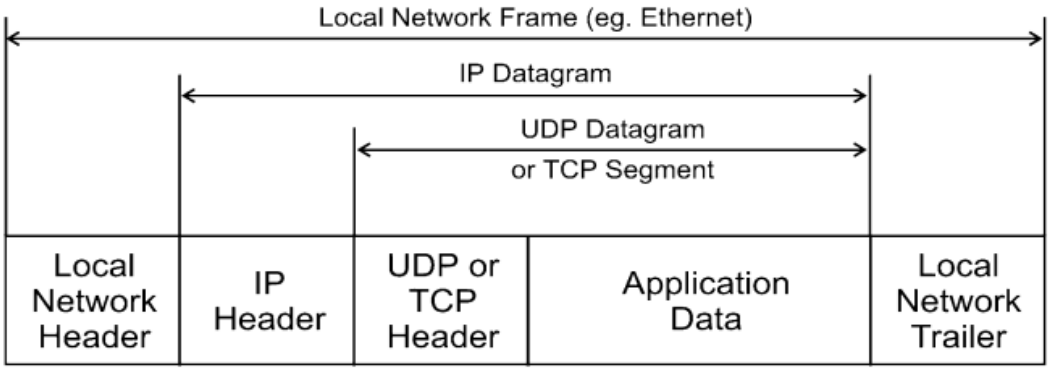

Figura 2: *Frame* do protocolo IP Fonte: Reynders, 2005

2.2.2 Modbus

Modbus é um protocolo de troca de mensagens da camada de aplicação do modelo OSI que provê comunicação de cliente e servidor entre dispositivos conectados em diferentes barramentos. "O protocolo Modbus implementa uma arquitetura de cliente e servidor e essencialmente opera em modo de "pedido e resposta" (REYNDERS, 2005, p. 132).
Um exemplo de como essa comunicação se dá entre dispositivos pode ser vista na figura 3.

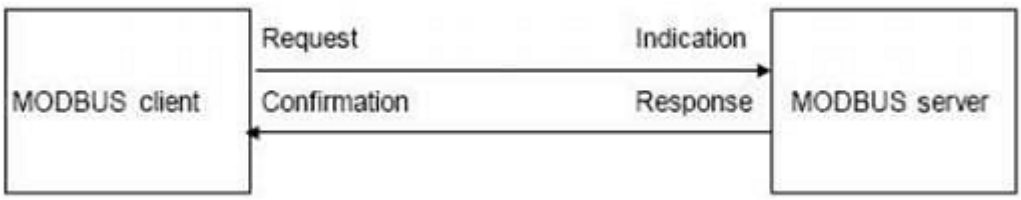

Figura 3: Modelo de comunicação Modbus Fonte: Reynders, 2005

Segundo Reynders (2005, p. 132) no caso de uma aplicação o cliente seria o dispositivo mestre, o qual realiza uma requisição. O servidor (escravo) assim realiza a ação e posteriormente inicia a resposta ao mestre como visto na figura 3.

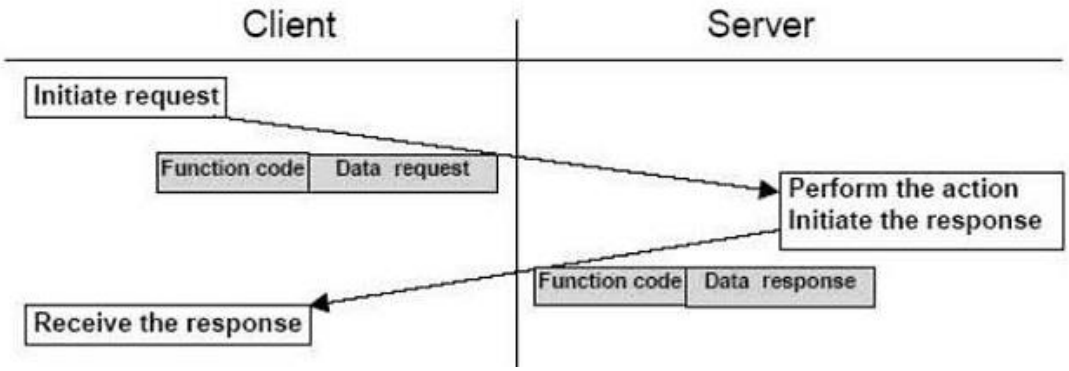

Figura 4: Fluxo de ações de cominicação entre mestre e escravo Fonte: Reynders, 2005

O protocolo Modbus, pode utilizar diversos meios para que a mensagem percorra na rede, sendo TCP/IP e ethernet, RS-232 ou RS-485, por exemplo.

O formato do *frame* do protocolo Modbus é evidenciado na tabela 1:

Tabela 1: *Frame* do protocolo Modbus Fonte: Reynders, 2005

Address Field	Function Field	Data Field	Error Check Field
1 byte	1 byte	Variable	2 bytes

O primeiro campo do *frame* é o *address field* o qual consiste no byte responsável por evidenciar o endereço do escravo que será requisitado uma resposta ou que terá uma resposta enviada. Cada escravo pode ter de 1 a 127 campos de endereço, porém a aplicação prática limita o número de dispositivos de 2 a 3 (REYNDERS, 2005, p. 134).

O segundo campo (*function field)* define a função que o mestre vai realizar no escravo escolhido. Essas funções podem ser vistas na tabela 2.

Tabela 2: Funções Modbus Fonte: Reynders, 2005

Data Type	Absolute Addresses	Relative Addresses	Function Codes	Description
Coils	00001–09999	0–9998	01	Read coil status
Coils	00001–09999	0–9998	05	Force single coil
Coils	00001–09999	0–9998	15	Force multiple coils
Discrete inputs	10001–19999	0–9998	02	Read input status
Input registers	30001–39999	0–9998	04	Read input registers
Holding registers	40001–49999	0–9998	03	Read holding register
Holding registers	40001–49999	0–9998	06	Preset single register
Holding registers	40001–49999	0–9998	16	Preset multiple registers
–	–	–	07	Read exception status
–	–	–	08	Loopback diagnostic test

O terceiro campo é referente aos dados. O seu tamanho é de acordo com a função especificada no *function field.* No *frame* de requisição esse campo é preenchido com informações extras para completar a função requisitada. Já no *frame* de resposta esse campo é preenchido com os dados requisitados do escravo (REYNDERS, 2005, p. 134).

E os dois últimos bytes compõe o campo de checagem de erro (*error checking field*). Esses bytes são valores calculados através de um algoritmo de analise de erros o que impede que mensagens danificadas ou corrompidas sejam lidas.

2.2.3 RS - 485

De acordo com Reynders (2005, p. 92) o protocolo RS-485 é um protocolo de comunicação serial muito versátil, pois além de utilizar somente dois fios permite uma linha de até 1200 m de distância e uma quantidade de 32 dispositivos na mesma linha de comunicação.

Os dois condutores que formam o barramento de comunicação são referenciados como A e B. O condutor A é conhecido como A- e o B como B+. A diferença de tensão entre os terminais que gera um valor de bit, sendo esses:

- -1,5 V a -6 V , no terminal A em relação ao B, para o valor de estado lógico 1.
- 1,5 V a 6 V , no terminal A em relação ao B, para o valor de estado lógico 0.

Além desses dois estados, tem-se o estado de alta impedância em que o dispositivo fica desabilitado.

A topologia física da rede Modbus pode ser vista na figura 5. Dois resistores de terminação são necessários para os casos de linhas longas de transmissão ou de alta taxa de transferência de bit. O valor típico é de 120 ohms. Além disso, dois resistores são adicionados na rede para garantir que não haja comunicação enquanto o dispositivo está no estado de alta impedância. Para este caso, dois resistores de pelo menos 560 ohms já é o suficiente (REYNDERS, 2005, p. 94).

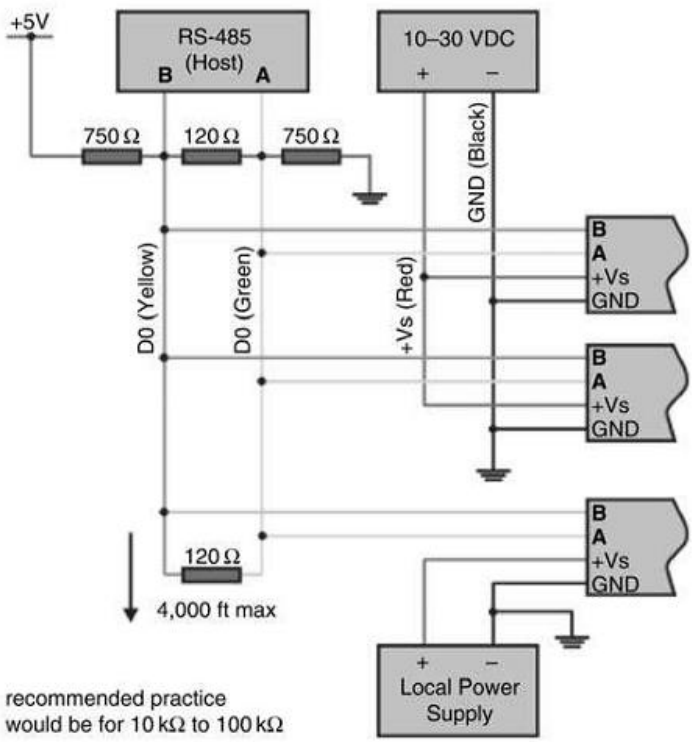

Figura 5: Topologia de uma rede Modbus Fonte: Reynders, 2005.

3 COMPONENTES UTILIZADOS

Para demonstrar a utilização da rede MODBUS será mostrada uma rede com dois arduinos e alguns sensores. A rede é composta então de um mestre e um escravo, sendo que no escravo estão conectados sensores e no mestre está ligado um LCD para mostrar os valores dos sensores. Ao longo desse capítulo serão descritos os componentes utilizados no experimento.

3.1 ARDUINO

> Em termos práticos, um Arduino é um pequeno computador que você pode programar para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele. O Arduino é o que chamamos de plataforma de computação física ou embarcada, ou seja, um sistema que pode interagir com seu ambiente por meio de hardware e software. Por exemplo, um uso simples de um

> Arduino seria para acender uma luz por certo intervalo de tempo, digamos, 30 segundos, depois que um botão fosse pressionado. Nesse exemplo, o Arduino teria uma lâmpada e um botão conectados a ele. O Arduino aguardaria pacientemente até que o botão fosse pressionado; uma vez pressionado o botão, ele acenderia a lâmpada e iniciaria a contagem. Depois de contados 30 segundos, apagaria a lâmpada e aguardaria um novo apertar do botão. Você poderia utilizar essa configuração para controlar uma lâmpada em um closet, por exemplo. (ROBERTS, 2012, p. 22)

O Arduino pode ser ligado em uma rede contendo outros dispositivos diversos para troca de dados. Resumidamente, ele pode receber dados de sensores e envia-los para outros dispositivos para que esses dados sejam expostos através de gráficos. (ROBERTS, 2012, p. 23)

O *software* e *hardware* do arduino são fonte aberta, o que proporciona qualquer pessoa aplicar os programas feitos na plataforma a qualquer propósito.

Dentre os diversos modelos de placas Arduino, foi utilizado dois modelos Uno (figura 6) no projeto. Um foi configurado como escravo e outro como mestre na rede modbus feita.

Figura 6: Arduino Uno Fonte: https://www.arduino.cc

O Arduino Uno possui as seguintes características.

- **Microcontrolador:** ATmega328
- **Tensão de Operação:** 5 volts
- **Pinos de entrada e saída digitais:** 14 sendo 6 possuem saída PWM.
- **Entradas Analógicas:** 6
- **Clock:** 16MHz

3.2 MÓDULO RS485

O módulo RS485 (figura 7) é utilizado para transmitir dados do arduino em longas distâncias (acima de 1 Km). (UNDERSTANDING... 2015)

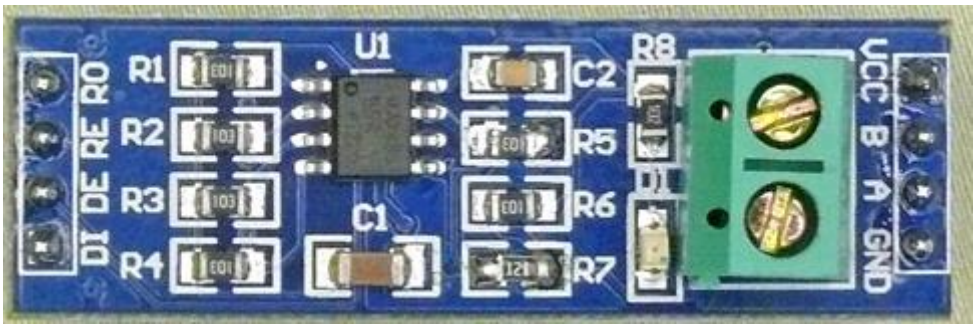

Figura 7: Módulo RS485 Fonte: http://arduino-info.wikispaces.com/

Como pode ser visto no esquema elétrico do módulo (figura 8) cada dispositivo já possui o resistor de terminação de valor 120 ohms e os dois resistores de 20K ohms para manter um sinal conhecido quando o módulo não estiver transmitindo. (UNDERSTANDING... 2015)

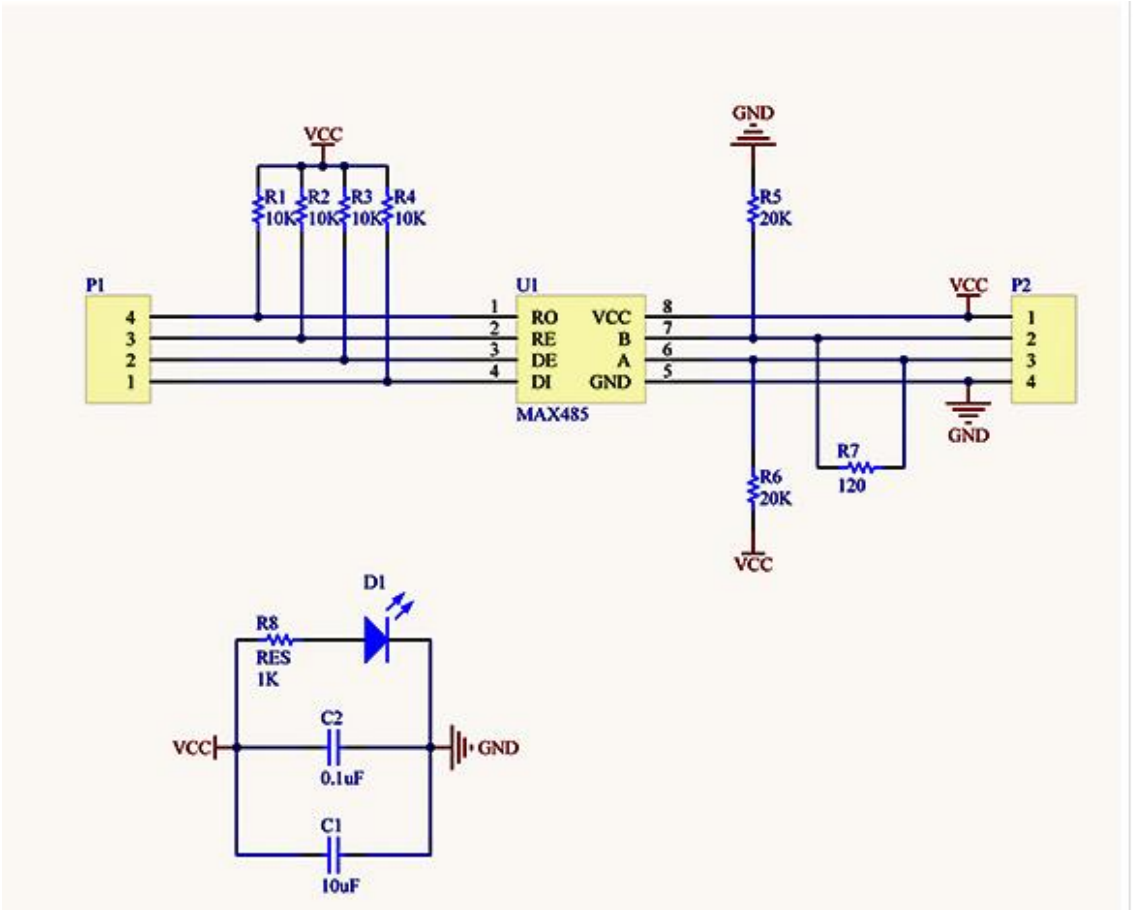

Figura 8: Esquema elétrico do módulo RS485 Fonte: http://arduino-info.wikispaces.com/

Foi utilizada a biblioteca *simple_modbus* (*SIMPLE*-MODBUS, 2015) para a utilização desse módulo. Assim, foi realizado seguinte esquema de ligações com o arduino: os pinos RO e DI são ligados, respectivamente, os pinos de recepção (Rx) e transmissão (TX) de informação serial do arduino. Os pinos DE e RE são ligados com uma saída digital do Uno que controla quando o módulo está transmitindo ou recebendo informação.

3.3 LDR

O LDR (figura 9), do inglês *Light Dependent Resistor*, é um resistor feito com elementos semicondutores. Os LDRs variam sua resistência com a quantidade de luminosidade ao qual são expostos. (BRINDLEY, 2005)

Figura 9: LDR Fonte: https://pt.wikipedia.org/wiki/LDR

Em sua superfície o LDR possui um material semicondutor que possui uma resistência elétrica elevada. Quando em contato com a luz esse material libera elétrons e assim a sua resistência diminui, permitindo uma maior passagem de corrente através dele. (WHAT... 2015)

No projeto o LDR foi utilizado com um circuito em série com um potenciômetro de 10 K ohms (figura 10). Assim, foi feito um ajuste da tensão referente ao ambiente que foram feitos os testes para que a tensão máxima fosse com a luz de uma lanterna de telefone celular ligada apontando diretamente para o LDR e a tensão mínima com as lâmpadas do quarto desligadas.

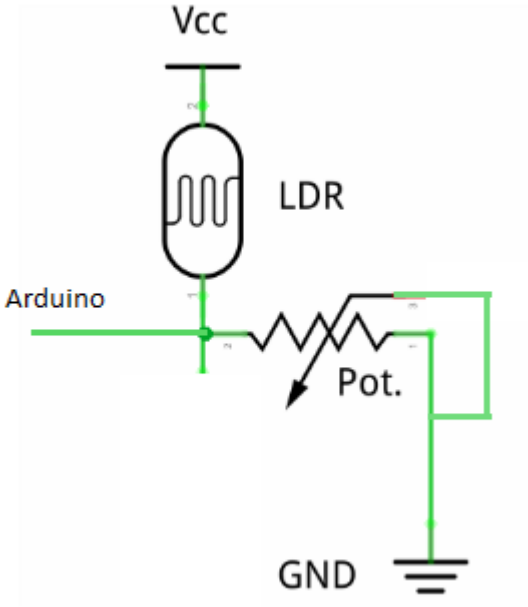

Figura 10: Circuito para LDR

3.4 TERMISTOR

Termistor (figura 11) é um resistor que o valor muda com a temperatura. Ele é feito de material semicondutor e eles variam em tamanho e forma, dependendo da forma como foram construídos. (BRINDLEY, 2005)

Figura 11: Termistor **Fonte: https://expressassembliesltdblog.files.wordpress.com**

Na aplicação o termistor que foi utilizado possui resistência de 10 K ohms a temperatura ambiente. Assim, foi colocado ele em série com um resistor de 10 K ohms (figura 12) e utilizado uma tabela em conjunto com algoritmo para gerar valores próximos do linear. (Apêndice 1)

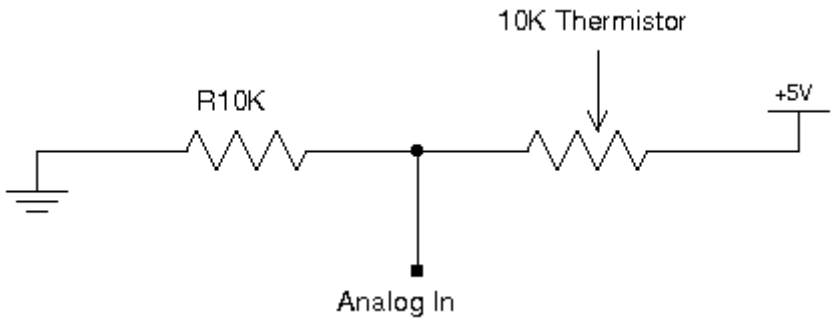

Figura 12: Circuito para Termistor **Fonte: http://playground.arduino.cc/**

3.5 LM35

O LM35, evidenciado na figura 13, é um dispositivo de precisão com uma tensão de saída linearmente proporcional à temperatura em graus centígrados (TEXAS INSTRUMENTS, 2015).

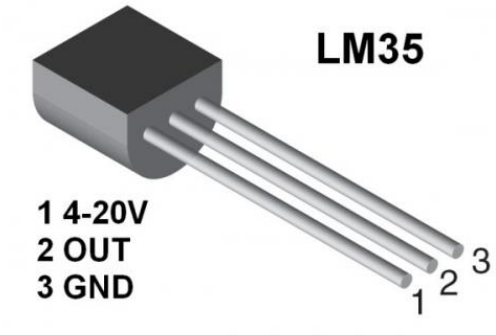

Figura 13: LM35 **Fonte: Texas Instruments, 2015**

Sua escala é de 10 mV / ºC e a alimentação que foi utilizada foi de 5 V. A figura 14 demonstra o circuito que foi realizado do LM35, sendo que o pino de saída (pino dois) foi ligado a entrada analógica do Arduino.

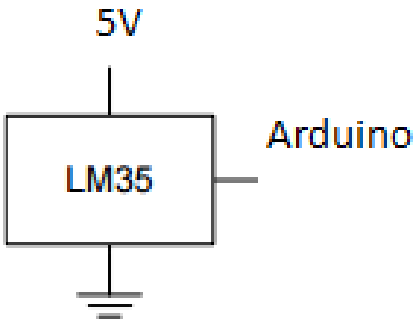

Figura 14: Esquema de conexão do LM35

3.6 TSOP1738

O TSOP1738 é um módulo de recepção infravermelho para sistemas de controle. Pela sua arquitetura interna, o mesmo possui uma robustez a luz do ambiente e a interferência provida de campos elétricos (VISHAY *SEMICONDUCTOR*, 1998). Na figura 15 pode ser visto o TSOP1738.

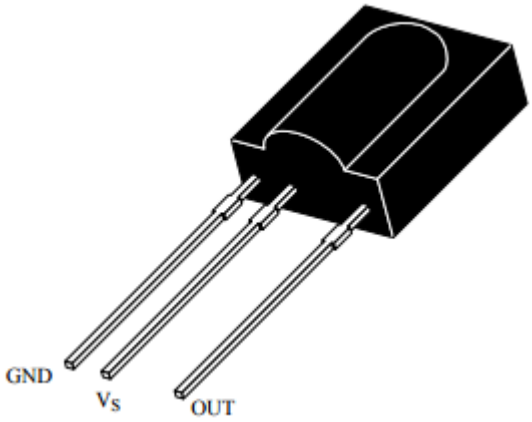

Figura 15: TSOP1738 Fonte: Vishay Semiconductor, 1998

O modo de funcionamento do TSOP1738 se dá da seguinte forma: o controle remoto emite pulsos de luz infravermelha que são lidos pelo sensor do TSOP. Esse sinal é demodulado pelo circuito interno do módulo infravermelho e assim o sinal pode ser lido por uma porta digital de um microcontrolador.

Na aplicação o TSOP foi utilizado para requisitar do arduino escravo os dados de leitura de um dos sensores.

3.7 DISPLAY DE LCD 16x2

Foi utilizado um display de LCD 16x2 como visto na figura 16 com o intuito de mostrar os dados coletados pelos sensores.

Figura 16: LCD 16 x 2 **Fonte: http://www.gravitech.us/**

O LCD utilizado tem sua comunicação em paralelo através de 4 bits.

Nesse capítulo foram abordados os componentes da aplicação realizada. Quais foram os sensores, displays e meios físicos de comunicação utilizados.

4 APLICAÇÃO EM MODBUS

Nesse capítulo será evidenciada a aplicação de modbus realizada. Serão abordados topologia de rede, programa utilizado e os resultados da aplicação.

4.1 TOPOLOGIA DE REDE

Para a aplicação foi utilizada uma rede com dois arduinos, sendo um com a função de mestre e outro escravo. A topologia da rede pode ser evidenciada na figura 17.

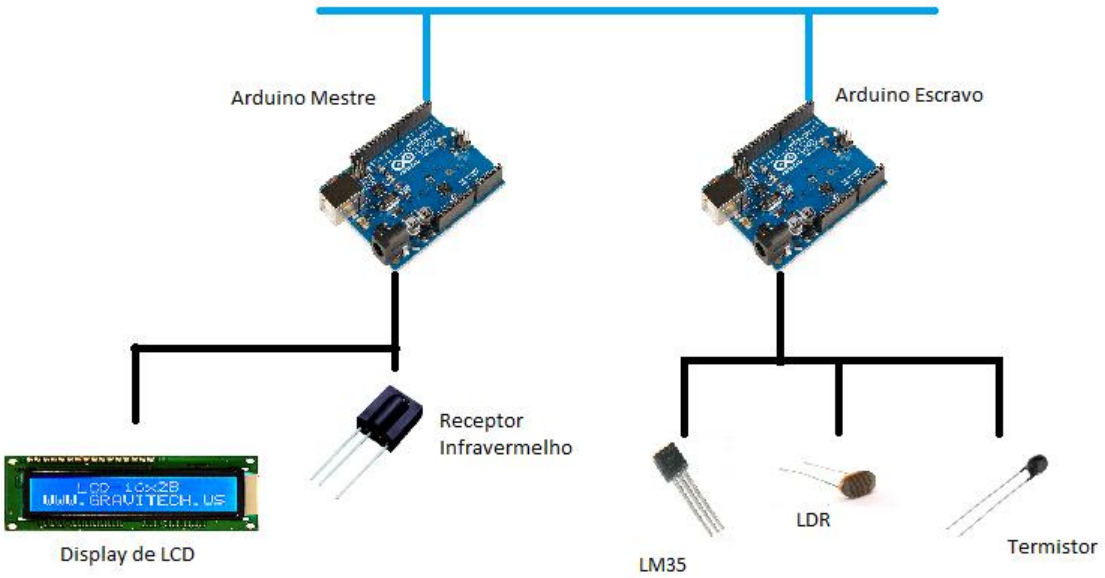

Figura 17: Topologia de rede da aplicação em Modbus

No esquema pode ser visto que o receptor infravermelho está ligado ao mestre e os sensores ao arduino escravo. Quando um comando é enviado através de um controle remoto apontado para o receptor infravermelho o mestre envia a informação de qual botão foi pressionado. Por sua vez o escravo envia a informação de tensão para o mestre de um dos três sensores para o mestre mostrar no LCD. Como o mestre já sabe qual sensor está sendo lido naquele momento, a conversão é realizada de forma correta e o valor correto é mostrado no LCD junto com o nome do sensor apresentado.

Para uma melhor visualização do processo, na figura 18 é evidenciado seu fluxograma.

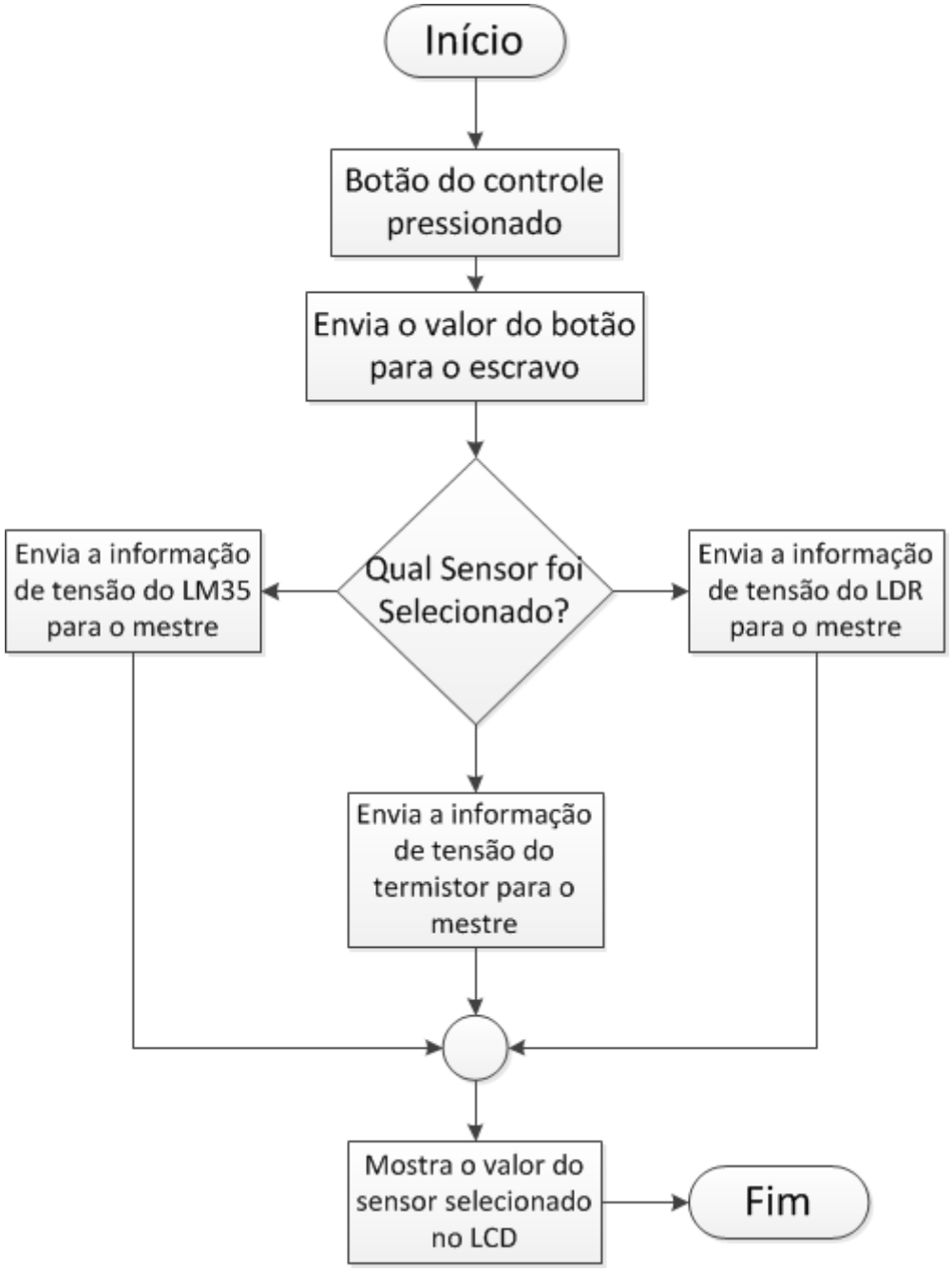

Figura 18: Fluxograma do processo

4.2 PROGRAMA DO MESTRE

O programa do arduino mestre tem como objetivo requisitar as informações dos sensores do escravo através de uma rede modbus. Para tal são utilizadas as funções *read holding registers* e *set holding registers* da biblioteca *simple modbus* para arduino. Dessa forma, são criados dois *frames* diferentes no mestre para cada função.

Cada *frame* criado possui associado a ele um pacote (*packet*) de informações que deve ser inicializado no início do programa.

4.2.1. Função *Read Holding Registers*

A função *read holding registers* é utilizada para quando se quer ler registradores de algum escravo determinado. Para a construção da função mencionada foi utilizado a linha:

- modbus_construct(&packets[PACKET1], 1, READ_HOLDING_REGISTERS, 0, 1, 0);

Esse construtor é chamado na função de *setup* do arduino utilizando alguns argumentos pré-determinados:

- **1ºArgumento &packets[PACKET1]:** Determina que um pacote de informações será mandado ao escravo determinado com a função Modbus que está sendo inicializada.

- **2ºArgumento 1:** Número de identificação do dispositivo escravo que será lido o valor do registrador.

- **3ºArgumento READ_HOLDING_REGISTERS:** Função Modbus utilizada no *frame*. Responsável por requisitar e ler os registradores do dispositivo escravo.

- **4º Argumento 0:** é o endereço do registrador que está sendo lido do escravo

- **5º Argumento 1:** é a quantidade de registradores que estão sendo usados na função

- **6º Argumento 0:** é o endereço de início que está sendo usado do registrador do mestre para armazenar a informação do escravo.

Na aplicação foi utilizado, no programa do escravo, um único registrador que guardava o valor do sensor selecionado. Assim, foi utilizado somente um registrador para a função *read holding registers*.

4.2.2 Função *Set Holding Registers*

Já a função *set holding registers* modifica um valor de um registrador no escravo. É utilizada essa função para enviar o valor do botão pressionado do controle remoto para que o sensor seja selecionado no escravo. Para a construção da função mencionada foi utilizada a linha:

- modbus_construct(&packets[PACKET2], 1, SET_HOLDING_REGISTERS, 1, 1, 0);

Esse *construtor* também é chamado na função de *setup* do arduino utilizando alguns argumentos diferentes. Os argumentos que se diferenciam da função *read holding registers* são:

- **1ºArgumento &packets[PACKET2]:** Como se trata de outra função, outro pacote é necessário

- **3ºArgumento SET_HOLDING_REGISTERS:** Função Modbus utilizada no *frame*. Responsável por modificar os registradores do dispositivo escravo.

- **4º Argumento 1:** é o endereço do registrador que está sendo modificado do escravo

O valor do controle remoto que é enviado para o escravo é em hexadecimal e corresponde a tecla do controle que foi pressionada.
A comunicação dos arduinos é estabelecida através da função *modbus_update*(). Esta realiza a ação de atualizar os registradores do escravo com o valor do registrador do mestre a ser enviado pela função *set holding registers* e de ler e atualizar o registrador do mestre com o valor do registrador do escravo através de *read holding registers*.

4.3 PROGRAMA DO ESCRAVO

O arduino escravo recebe a requisição vinda do mestre para enviar ou receber informação, porém nele também são realizadas algumas configurações. A configuração da comunicação do arduino escravo com o mestre é dada pela linha:

- modbus_configure(&Serial, 9600, SERIAL_8N2, 1, 2, HOLDING_REGS_SIZE, holdingRegs);

Os argumentos de configuração são os seguintes:

- **1º e 2º Argumentos &Serial, 9600:** Configuração da porta serial e do *baud rate*.

- **3ºArgumento SERIAL_8N2:** Formato do Byte que será utilizado na transmissão dos dados.

- **4º Argumento 1:** É o endereço do escravo

- **5º Argumento 2:** É o pino utilizado para alterar o shiled 485 de transmissor para receptor.

- **6º Argumento HOLDING_REGS_SIZE:** É o tamanho do *array* de registradores

- **7º Argumento holdingRegs:** É o *array* de registradores que será enviado para o mestre.

O arduino escravo possui dois registradores. Um deles guarda o valor de um dos sensores, dependendo da escolha feita pelo controle, e o outro é o valor em hexadecimal correspondente ao botão pressionado.
Quando um botão do controle remoto é pressionado esse valor é enviado do mestre para o escravo. O registrador reg[1] (Anexo II) do arduino escravo é modificado e assim assume o valor do botão do controle remoto. Dependendo desse valor, o sensor é selecionado e assim o nível de tensão do respectivo sensor é enviado para o mestre e com isso, evidenciado em um display LCD.

4.4 MONTAGEM

A aplicação foi montada em um protoboard e testada, sendo que houve total estabilidade do sistema. A figura 19 mostra o os arduinos montados.

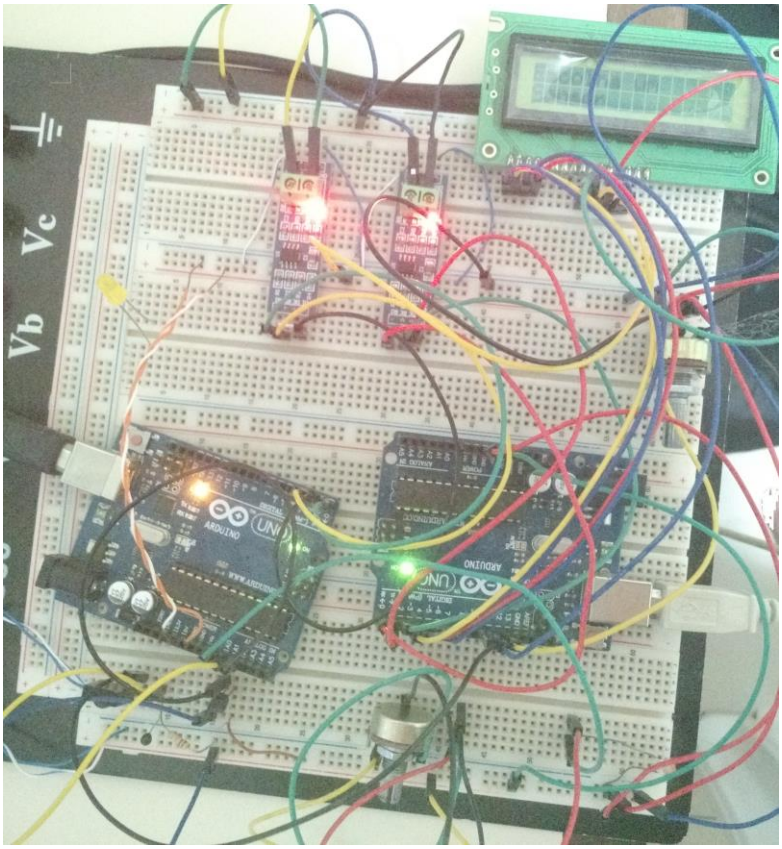

Figura 19: Protótipo Montado

Assim, a aplicação foi realizada com sucesso, utilizando o protocolo modbus RTU através da biblioteca *simple modbus* e do meio físico RS 485.

5 CONSIDERAÇÕES FINAIS E RECOMENDAÇÕES

Com essa aplicação, foi visto que o objetivo de montar uma rede modbus de baixo custo com arduinos foi um sucesso.

A transmissão de dados foi estável em praticamente todo o período do teste tendo alguns momentos de travamento, que por sua vez foram casos pontuais. Além disso, o processamento da informação por parte do arduino e a velocidade de comunicação entre mestre e escravo se mostrou rápida e satisfatória para o trabalho.

Como a rede modbus é utilizada amplamente na indústria, os equipamentos que convertem e utilizam esse tipo de protocolo para transmissão de dados são equipamentos caros. Assim, como o arduino é um equipamento barato para os padrões industriais, é um passo para reduzir o custo de uma tecnologia restrita a um nicho de mercado onde os equipamentos são de um valor agregado elevado.

Para a continuidade do estudo, devem-se desenvolver os seguintes aspectos do tema:

- Obter a comunicação entre um arduino e um Controlador Lógico Programável (CLP), já que o protocolo utilizado é compatível com esses dispositivos.
- Verificar a possibilidade de integração de outras plataformas mais robustas na rede modbus, pois em uma aplicação real os arduinos podem não responder de forma satisfatória devido a sua baixa velocidade de processamento.
- Desenvolver dispositivos sensores já integrados com modbus e dessa forma, criar alternativas mais baratas de transmissores no mercado.

REFERÊNCIAS BIBLIOGRÁFICAS

BRINDLEY, Keith. **Starting Electronics Construction: Techniques, Equipment and Projects**. Burlington: Newnes, 2005.

CISCO. **CISCO CCNA**. Disponível em: <https://learningnetwork.cisco.com/community/learning_center/ccna_rs_study_sessions_live >. Acesso em: 10 abr. 2015.

FOROUZAN, Behrouz A.. **Comunicação de Dados e Redes de Computadores.** Nova Iorque: Mcgraw Hill, 2007.

MENDES, Douglas Rocha. **Redes de Computadores Teoria e Prática.** São Paulo: Novatec, 2007.

MCROBERTS, Michael. **Arduino Básico.** São Paulo: Novatec Editora, 2012.

MILLER, Philip M. **TCP/IP: The Ultimate Protocol Guide, Volume 1.** Boca Raton: Brown Walker Press, 2009.

NOGUEIRA, Thiago Augusto. **Redes de comunicação para sistemas de automação industrial**. 2009. 83 f. Tese (Doutorado) - Curso de Engenheiaria de Controle e Automação, Universidade Federal de Ouro Preto, Ouro Preto, 2009.

REYNDERS, Deon. **Practical Industrial Data Communications Best Practice Techniques.** Burlington: Butterworth-heinemann, 2005.

SIMPLE-MODBUS: **Modbus RTU libraries for Arduino**. Disponível em: <https://code.google.com/p/simple-modbus/>. Acesso em: 05 mar. 2015.

TANENBAUM, Andrew S. **Redes de Computadores**. Rio de Janeiro: Pearson Education - Br, 2003.

Texas Instruments Inc.**LM35 Precision Centigrade Temperature Sensors**, 2015, SNIS159E UNDERSTANDING RS422 and RS485. Disponível em: <http://arduino-info.wikispaces.com/RS485Info>. Acesso em: 25 maio 2015.

Vishay Semiconductor. **Photo Modules for PCM Remote Control Systems**, 1998, Rev. A5

WHAT is a Light Dependent Resistor? Disponível em: <http://www.wisegeek.com/what-is-a-light-dependent-resistor.htm>. Acesso em: 12 jun. 2015.

APÊNDICES

APÊNDICE A – PROGRAMA DO MESTRE

```
////////////// Biblioteca do controle remote ///////////////
#include <IRremote.h>
#include <IRremoteInt.h>

////////////// Biblioteca do LCD ///////////////
#include <LiquidCrystal.h>
////////////// Biblioteca Modbus///////////////
#include <SimpleModbusMaster.h>

//Indica quais pinos do LCD serão utilizados
LiquidCrystal lcd(12, 11, 5, 4, 3, 6);

int receiver = 10; // pin 1 do Receptor Infravermelho para o pino 10 do Arduino
IRrecv irrecv(receiver); // estanciando a função de leitura do infravermelho
decode_results results;

//////////////////// Configuração da Serial  ////////////////////
#define baud 9600
#define timeout 1000
#define polling 200 // Taxa de atualização
#define retry_count 10

// pino que habilita transmissão ou recepção do shield 485
#define TxEnablePin 2


// Total de registradores no mestre
#define TOTAL_NO_OF_REGISTERS 2
```

```
// Essa maneira pode ser criada quantos pacotes desejar
enum
{
  PACKET1,
  PACKET2,
  TOTAL_NO_OF_PACKETS // gera o número do total de pacotes criados
};

// Cria um array de pacotes
Packet packets[TOTAL_NO_OF_PACKETS];

// Array de registradores do mestre
unsigned int regs[TOTAL_NO_OF_REGISTERS];
 int ir;
 float temp, ldr;
void setup()
{

  lcd.begin(16, 2);
   lcd.print("Arduino MODBUS");
  delay(1000);
   irrecv.enableIRIn(); // Inicia o TSOP1738

  // Inicializa cada pacote
  modbus_construct(&packets[PACKET1], 1, READ_HOLDING_REGISTERS, 0, 1, 0);
  modbus_construct(&packets[PACKET2], 1, PRESET_MULTIPLE_REGISTERS, 1, 1, 0);

  // Inicializa a configuração do Modbus
 modbus_configure(&Serial, baud, SERIAL_8N2, timeout, polling, retry_count,
TxEnablePin, packets, TOTAL_NO_OF_PACKETS, regs);

  pinMode(13, OUTPUT);
}
```

```
void loop()
{
  modbus_update(); // Atualiza os registradores do escravo e faz a leitura
  if (irrecv.decode(&results)) // Recebemos um Sinal de Infravermelho?
  {
    regs[1]  = results.value;
    digitalWrite(13, HIGH);   // Liga um led de status da utilização do TSOP1738
    delay(10);
    digitalWrite(13, LOW);    // Desliga um led de status da utilização do TSOP1738
    delay(10);
    irrecv.resume(); // Recebe o valor do código do controle remoto
  }

///////////////Verificação de Qual Sensor foi Selecionado/////////////

else if ( regs[1] == 0x2658 || regs[1]  == 0xA658)  // Sensor LDR
{
    lcd.clear();
    lcd.setCursor(0, 0);
    ldr = (float)(1023-regs[0])/1023;
    ldr = ldr*100;
    lcd.print("Iluminacao");
    lcd.setCursor(0, 1);
    lcd.print(ldr);
    lcd.print("%");
    delay(150);
  }

 else if (regs[1]  == 0x265A || regs[1]  == 0xA65A) // Termistor
  {
    lcd.clear();
    lcd.setCursor(0, 0);
    temp = (float)(regs[0]/10);
```

```
   lcd.print("Temperatura NTC");
   lcd.setCursor(0, 1);
   lcd.print(temp);
   lcd.print("C");
   delay(150);
  }

  else if (regs[1]  == 0x265B || regs[1] == 0xA65B) // LM35
  {
   lcd.clear();
   lcd.setCursor(0, 0);
   temp = (float)(regs[0]*0.48875855327468230694037145 65);
   lcd.print("Temperatura LM35");
   lcd.setCursor(0, 1);
   lcd.print(temp);
   lcd.print("C");
   delay(150);
  }

  else
  {
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Escolha um"); // Mostra a opção para selecionar um Sensor
   lcd.setCursor(0, 1);
   lcd.print("Sensor");
   delay(150);
 }
}
```

APÊNDICE B – PROGRAMA DO ESCRAVO

```
#include <SimpleModbusSlave.h>

////////////////////// Tabela de Linearização do Termistor//////////////////////
 const int temps[] PROGMEM = { 0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 68,
69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 88, 89, 90, 91, 92,
93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111,
112, 113, 114, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 124, 125, 126, 127, 128,
129, 130, 131, 132, 133, 134, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 143, 144, 145,
146, 147, 148, 149, 150, 151, 151, 152, 153, 154, 155, 156, 157, 158, 159, 159, 160, 161, 162,
163, 164, 165, 166, 167, 167, 168, 169, 170, 171, 172, 173, 174, 175, 175, 176, 177, 178, 179,
180, 181, 182, 182, 183, 184, 185, 186, 187, 188, 189, 190, 190, 191, 192, 193, 194, 195, 196,
197, 197, 198, 199, 200, 201, 202, 203, 204, 205, 205, 206, 207, 208, 209, 210, 211, 212, 212,
213, 214, 215, 216, 217, 218, 219, 220, 220, 221, 222, 223, 224, 225, 226, 227, 228, 228, 229,
230, 231, 232, 233, 234, 235, 235, 236, 237, 238, 239, 240, 241, 242, 243, 243, 244, 245, 246,
247, 248, 249, 250, 251, 252, 252, 253, 254, 255, 256, 257, 258, 259, 260, 260, 261, 262, 263,
264, 265, 266, 267, 268, 269, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 279, 280,
281, 282, 283, 284, 285, 286, 287, 288, 289, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298,
299, 300, 301, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 315,
316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334,
335, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352,
353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371,
372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390,
392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 410, 411,
412, 413, 414, 415, 416, 417, 418, 419, 420, 422, 423, 424, 425, 426, 427, 428, 429, 430, 432,
433, 434, 435, 436, 437, 438, 439, 441, 442, 443, 444, 445, 446, 448, 449, 450, 451, 452, 453,
455, 456, 457, 458, 459, 460, 462, 463, 464, 465, 466, 468, 469, 470, 471, 472, 474, 475, 476,
477, 479, 480, 481, 482, 484, 485, 486, 487, 489, 490, 491, 492, 494, 495, 496, 498, 499, 500,
501, 503, 504, 505, 507, 508, 509, 511, 512, 513, 515, 516, 517, 519, 520, 521, 523, 524, 525,
527, 528, 530, 531, 532, 534, 535, 537, 538, 539, 541, 542, 544, 545, 547, 548, 550, 551, 552,
```

```
554, 555, 557, 558, 560, 561, 563, 564, 566, 567, 569, 570, 572, 574, 575, 577, 578, 580, 581,
583, 585, 586, 588, 589, 591, 593, 594, 596, 598, 599, 601, 603, 604, 606, 608, 609, 611, 613,
614, 616, 618, 620, 621, 623, 625, 627, 628, 630, 632, 634, 636, 638, 639, 641, 643, 645, 647,
649, 651, 653, 654, 656, 658, 660, 662, 664, 666, 668, 670, 672, 674, 676, 678, 680, 683, 685,
687, 689, 691, 693, 695, 697, 700, 702, 704, 706, 708, 711, 713, 715, 718, 720, 722, 725, 727,
729, 732, 734, 737, 739, 741, 744, 746, 749, 752, 754, 757, 759, 762, 764, 767, 770, 773, 775,
778, 781, 784, 786, 789, 792, 795, 798, 801, 804, 807, 810, 813, 816, 819, 822, 825, 829, 832,
835, 838, 842, 845, 848, 852, 855, 859, 862, 866, 869, 873, 877, 881, 884, 888, 892, 896, 900,
904, 908, 912, 916, 920, 925, 929, 933, 938, 942, 947, 952, 956, 961, 966, 971, 976, 981, 986,
991, 997, 1002, 1007, 1013, 1019, 1024, 1030, 1036, 1042, 1049, 1055, 1061, 1068, 1075,
1082, 1088, 1096, 1103, 1110, 1118, 1126, 1134, 1142, 1150, 1159, 1168, 1177, 1186, 1196,
1206, 1216, 1226, 1237, 1248, 1260, 1272, 1284, 1297, 1310, 1324, 1338, 1353, 1369, 1385,
1402, 1420, 1439, 1459, 1480, 1502 };

//////////////// Registradores do Escravo ///////////////////
enum
{
  ADC_VAL,
  HOLDING_REGS_SIZE
};

unsigned int holdingRegs[HOLDING_REGS_SIZE]; // Array de registradores utilizados


void setup()
{

  modbus_configure(&Serial, 9600, SERIAL_8N2, 1, 2, HOLDING_REGS_SIZE,
holdingRegs);
modbus_update_comms(9600, SERIAL_8N2, 1);
}
int therm;
```

```
void loop()
{
  else if ( regs[1] == 0x2658 || regs[1]  == 0xA658) // Seleciona o LDR
          holdingRegs[ADC_VAL] = analogRead(A0);

  else if (regs[1]  == 0x265A || regs[1]  == 0xA65A) // Seleciona o Termistor
  {
          therm = analogRead(A1)-238;
          holdingRegs[ADC_VAL] = pgm_read_word(&temps[therm]);
   }

  else if (regs[1]  == 0x265B || regs[1] == 0xA65B) // Seleciona o LM35
  {
     holdingRegs[ADC_VAL] = analogRead(A2);
  }

   modbus_update(); // Atualiza os valores dos registradores
  }
```