Barramentos Industriais

Projeto 01 – Dispositivo Modbus RTU EIA-485

IFRS – Campus Farroupilha
Engenharia de Controle e Automação
Professor Gustavo Künzel Aluno Pedro Henrique de Assumpção

INSTITUTO FEDERAL
Rio Grande do Sul | Campus Farroupilha

06/05/2025

ÍNDICE

Introdução

Utilizar conceitos de Modbus RTU e comunicação serial UART para desenvolver uma aplicação compatível com o protocolo Modbus RTU.

O objetivo é compreender como deve ser feita a programação do protocolo nos dispositivos.

Nas I/Os do Arduino serão conectados potenciômetros e LEDs, de acordo com a aplicação.

Objetivos

- **Módulo de saídas digitais (8 a 16 saídas):**
  - **Representadas por LEDs;**
  - **Programa do PC deve permitir acionamento de uma ou mais saídas no mesmo comando;**
  - **Função Modbus: 0x0F Write Multiple Coils.**

Justificativa

- **Desenvolver um sistema que permita o controle de uma matriz de 64 LEDs via protocolo Modbus RTU utilizando Arduino. Além disso, busca-se compreender a implementação prática do protocolo e sua aplicabilidade em sistemas embarcados.**

Fundamentação Teórica

Protocolo Modbus RTU

- **Modbus RTU é um protocolo de comunicação serial mestre-escravo que utiliza RS-485 para transmissão de dados. Ele permite a comunicação entre dispositivos de forma eficiente e é padrão em sistemas industriais.**

Fundamentação Teórica

Comunicação Serial e RS-485

- A comunicação serial RS-485 permite a transmissão de dados em longas distâncias com alta imunidade a ruídos. É ideal para aplicações industriais e é compatível com o protocolo Modbus RTU.

Fundamentação Teórica

Write Multiple Coils (0x0F)

Campo	Tamanho (bytes)	Descrição
Endereço do Escravo	1 byte	ID do dispositivo que receberá o comando (ex.: 0x01)
Código da Função	1 byte	0x0F, indicando escrita de múltiplos coils
Endereço Inicial	2 bytes	Endereço do primeiro coil a ser escrito (ex.: 0x0000)
Quantidade de Coils	2 bytes	Número de coils a serem escritos (ex.: 0x0010 = 16 coils)
Byte Count	1 byte	Quantidade de bytes de dados (ex.: 0x02 para 16 coils)
Dados dos Coils	2 bytes	Estados binários dos 16 coils (ex.: resultadoA e resultadoB)
CRC16	2 bytes	Código de verificação de integridade, LSB primeiro

Proposta

Componentes do Sistema

- O sistema é composto por um Arduino Uno, um módulo RS-485 (como o MAX485), e uma matriz de LEDs 8x8. A comunicação entre o Arduino e a matriz é realizada via interface serial, utilizando o protocolo Modbus RTU.

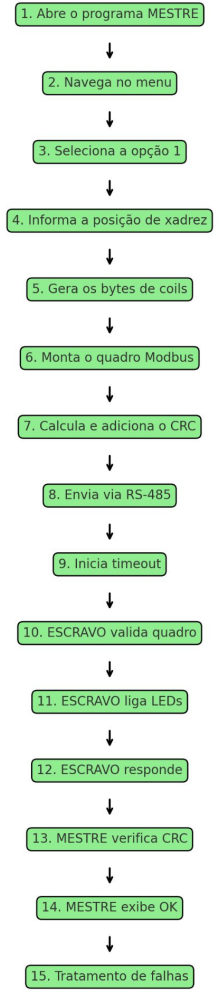
Proposta

Matriz de LEDs

- A matriz de LEDs 8x8 possui 64 LEDs dispostos em linhas e colunas. É controlada por meio de registros que determinam quais LEDs devem ser acesos, permitindo a criação de diversos padrões visuais.

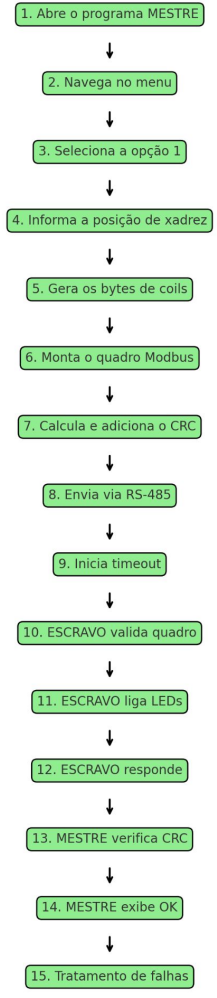
Fluxograma explicado

1. Usuário abre o programa MESTRE
O software em C para Linux é executado no terminal pelo usuário.
2. Navega no menu do terminal
Um menu interativo com opções de controle da matriz de LEDs é exibido.
3. Seleciona a opção 1
O usuário escolhe a função de envio de comando para acender LEDs.
4. Informa a posição de xadrez desejada (ex: D4)
O programa converte essa posição em um padrão de bits correspondente aos LEDs que devem acender.
5. Gera os bytes de dados dos coils (`resultadoA` e `resultadoB`)
O padrão de 16 bits é dividido em dois bytes, representando os estados dos LEDs.



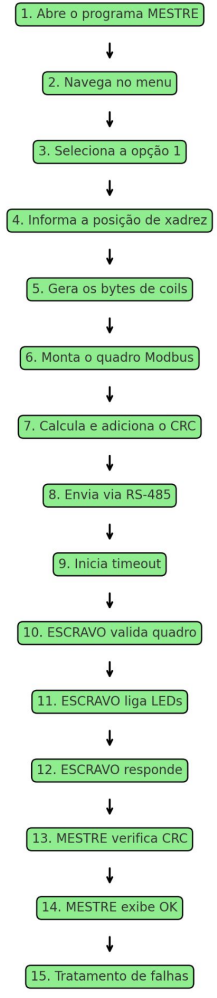
Fluxograma explicado

6. Monta o quadro Modbus RTU (função 0x0F)
O mestre constrói o quadro com endereço do escravo, função, quantidade de coils e dados.
7. Calcula o CRC16 e adiciona ao quadro
Um código de verificação (CRC) é gerado usando a biblioteca `crc16.h` e anexado ao final do quadro.
8. Envia o quadro pela interface serial RS-485
O mestre transmite o quadro pela porta `/dev/ttyUSB0` para o Arduino escravo.
9. Inicia contagem de tempo (timeout)
A função `millis_now()` é usada para garantir que a resposta do escravo chegue em tempo hábil.
10. ESCRAVO (Arduino) recebe e valida o quadro
O Arduino verifica o endereço, função e CRC para confirmar se o quadro é válido e para ele.

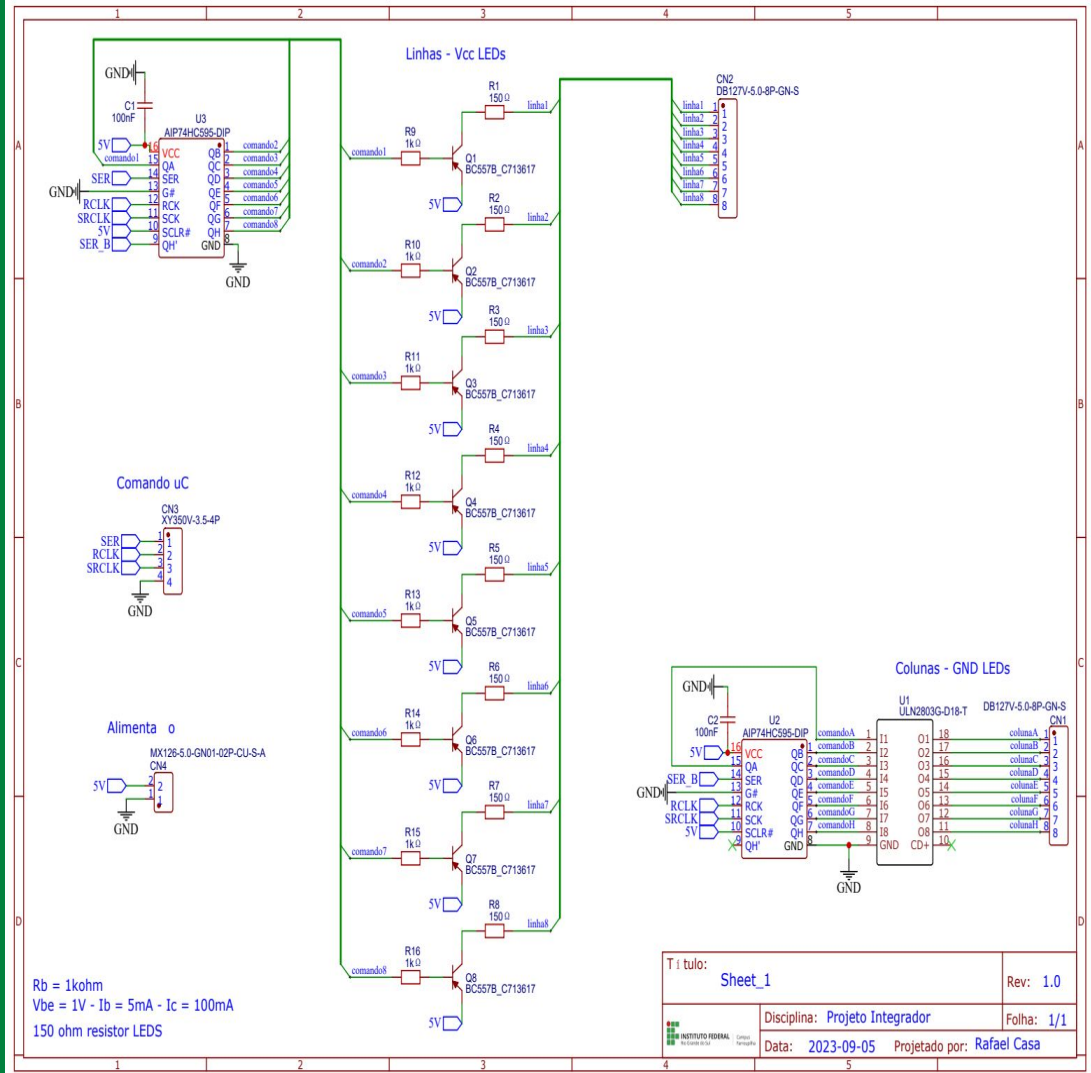


Fluxograma explicado

11. Escravo interpreta os dados e atualiza os LEDs
Os bytes recebidos (`resultadoA` e `resultadoB`) são usados para acender os LEDs correspondentes na matriz 8x8.
12. Escravo monta e envia resposta Modbus
O Arduino responde com um quadro de confirmação contendo os dados esperados.
13. Mestre recebe a resposta e verifica o CRC
O software mestre valida a integridade da resposta com novo cálculo de CRC.
14. Mestre exibe mensagem de sucesso (“OK”)
Se a resposta for válida, o terminal exibe que os LEDs foram atualizados com sucesso.
15. Tratamento de falhas
Se o tempo de resposta for excedido ou o CRC estiver errado, o mestre exibe erro e permite nova tentativa.

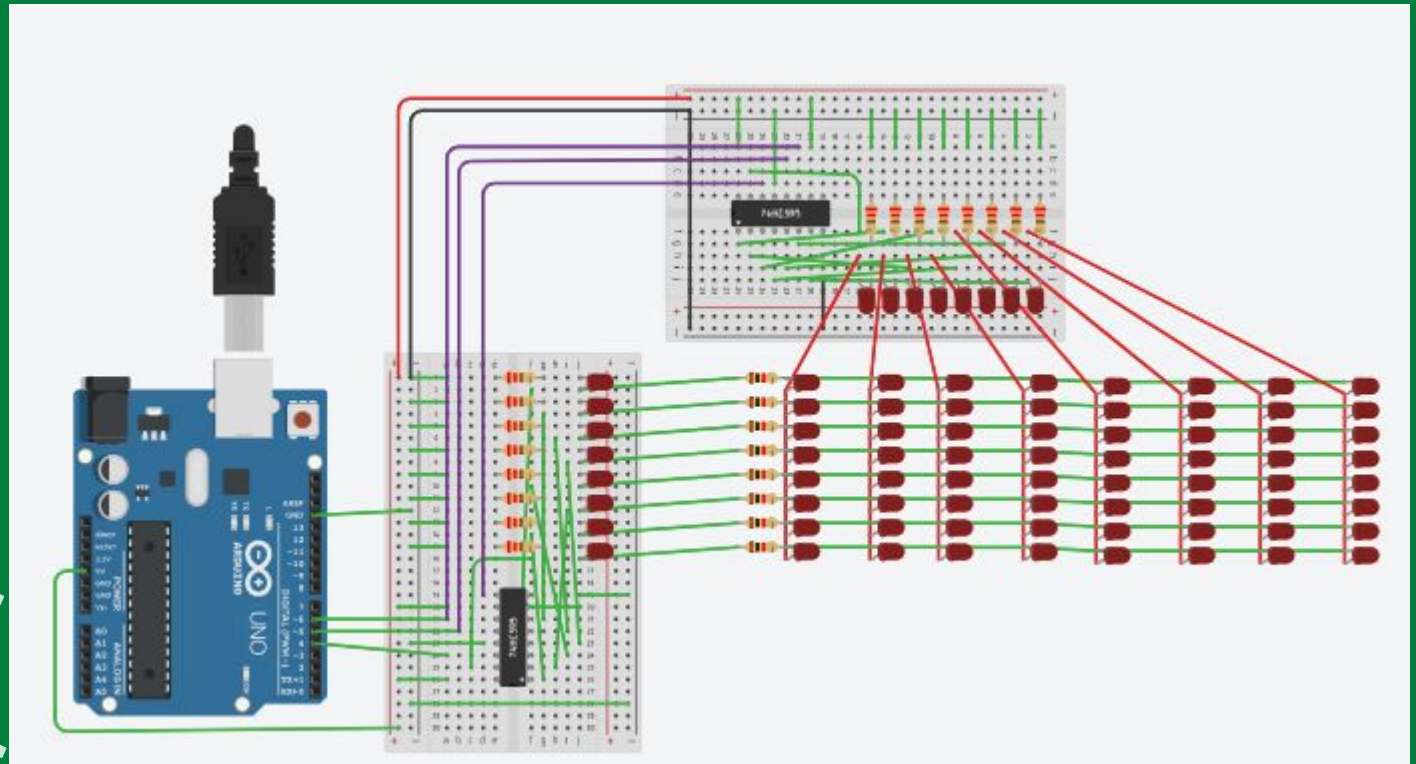


Esquemático da placa de acionamento dos LEDs



Projeto no tinkercad.com . Disponível em

<https://www.tinkercad.com/things/by0YwzXKzMQ/editel?sharecode=d4ml6vyh2X25edsKUoraJBd6mtZ-cWJj8xbMbh_ZS_I >



Testes

Procedimentos de Teste

- **Foram realizados testes para verificar a correta comunicação entre o mestre e o escravo, bem como o funcionamento da matriz de LEDs. Os testes confirmaram a eficácia da implementação.**

Testes

Rotinas de Falhas

- O menu do Mestre possui rotinas que tratam erros como CRC inválido e ausência de resposta do escravo. Se ocorrer falha, o sistema exibe mensagens de erro e permite nova tentativa, sem travar a execução.

Desafios

Troca do ESP32 pelo Arduino

O uso inicial do ESP32 apresentou instabilidade na comunicação Modbus RTU e no acionamento da matriz de LEDs, principalmente por operar com sinais de 3,3V. Essa limitação prejudicou o funcionamento com os módulos RS-485. A substituição pelo Arduino, que opera em 5V, solucionou o problema e garantiu confiabilidade na comunicação e no controle dos LEDs.

Desafios

Melhorias no Programa Mestre

Identificou-se que o programa Mestre pode ser aprimorado com a parametrização de variáveis atualmente estáticas, como o endereço do escravo e a porta serial. Torná-las configuráveis aumentaria a flexibilidade do sistema e permitiria múltiplas execuções com diferentes dispositivos sem alteração no código-fonte.

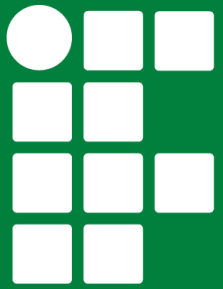
Desafios

Endereçamento e Broadcast

Foi implementado o suporte para múltiplos escravos com endereços de 1 a 4. Também foi considerada a funcionalidade de broadcast, em que todos os escravos recebem o comando, mas nenhum deve responder ao barramento, evitando colisões. Essa abordagem amplia o potencial de expansão do sistema em redes com múltiplos dispositivos.

Referências

- MODBUS ORGANIZATION. Modbus Application Protocol Specification V1.1b3. [S.l.]: Modbus Organization, 2012. Disponível em: https://modbus.org. Acesso em: 5 maio 2025.
- DE ASSUMPÇÃO, Pedro Henrique. ModbusRTU com Arduino. GitHub, 2024. Disponível em: https://github.com/pedrohdea/ModbusRTU. Acesso em: 5 maio 2025.
- ARDUINO. Arduino Uno Rev3 – Datasheet. [S.l.]: Arduino.cc, 2020. Disponível em: https://store.arduino.cc/products/arduino-uno-rev3. Acesso em: 5 maio 2025.



INSTITUTO FEDERAL

Rio Grande do Sul

Campus Farroupilha