Leitor Modbus RTU via RS-485

Protocolo implementado do zero no ESP32 - manual prático



Placa	ESP32 ou ESP32-S3
Framework	Arduino Core, via PlatformIO
Bibliotecas	nenhuma - protocolo implementado no próprio arquivo
Transceiver	MAX3485 ou SP3485 (3,3 V). MAX485 exige divisor no RO
Nível	Avançado
Tutorial online	iiot.esp32.tec.br/tutoriais/esp32-lendo-modbus-rtu-via-rs485

Sobre a validação. O algoritmo de CRC16 foi conferido contra os vetores da especificação Modbus: o quadro 11 03 00 6B 00 03 produz CRC 0x8776, transmitido como 76 87. O firmware, porém, não foi validado em bancada contra um equipamento real. Teste com o seu escravo antes de usar em instalação que importe.

1. O essencial do RS-485

Não é uma serial comum

Numa serial comum cada lado tem seu fio de transmissão. No RS-485 existe um par de fios só, compartilhado por todos, e a informação está na diferença de tensão entre os dois. É isso que faz o sinal atravessar centenas de metros num ambiente cheio de motor e inversor, onde a serial comum não anda dez.

A consequência é que **só um dispositivo fala por vez**. Não há colisão detectável nem arbitragem: o mestre pergunta, o escravo responde, e nesse intervalo mais ninguém abre a boca.

O pino que decide tudo

O transceiver tem quatro pinos de controle. O DI recebe o que o ESP32 transmite e o RO entrega o que chega do barramento. O DE habilita o transmissor em nível alto; o RE habilita o receptor em nível baixo.

Como você nunca quer transmitir e receber ao mesmo tempo, o padrão é amarrar DE e RE no mesmo GPIO: alto significa falar, baixo significa ouvir.

```
ESP32 GPIO17 (TX) -> DI
ESP32 GPIO16 (RX) <- RO
ESP32 GPIO4 -> DE e RE juntos
3V3 -> VCC
GND -> GND
```

> Em algumas placas ESP32-S3 os GPIOs 16 e 17 estão ocupados pela PSRAM octal. Se for o seu caso, escolha outros pinos no topo do main.cpp.

2. As três armadilhas

Armadilha 1 - soltar o DE cedo demais

É o bug mais comum é o mais difícil de enxergar. A tentação é escrever no serial e desligar o DE na linha seguinte. Só que o write não espera o dado sair: ele apenas enfileira no buffer da UART. Baixando o DE em seguida, o transmissor desliga no meio do quadro e o escravo recebe mensagem truncada, sem nunca responder.

```
digitalWrite(PINO_DE_RE, HIGH);
delayMicroseconds(50);
Serial2.write(quadro, tamanho);
Serial2.flush();
delayMicroseconds(50);
digitalWrite(PINO_DE_RE, LOW);
```

O flush é a linha que importa: bloqueia até o último bit sair do registrador de deslocamento. As esperas de 50 microssegundos dão folga ao transceiver para comutar.

Armadilha 2 - ordem dos bytes do CRC

O CRC16 entra no quadro com o byte **menos significativo primeiro**, ao contrário de todo o resto do protocolo, que é big-endian. Endereço e quantidade vão com o byte alto na frente; o CRC vai invertido. Errar isso produz um quadro que parece perfeito e nunca é aceito.

```
uint16_t crc = modbusCRC(quadro, 6);
quadro[6] = crc & 0xFF; // byte baixo primeiro
quadro[7] = crc >> 8;
```

Armadilha 3 - o silêncio entre quadros

Modbus RTU não tem marcador de início nem de fim. O que delimita a mensagem é tempo de silêncio: uma pausa de 3,5 caracteres marca o fim, e uma pausa maior que 1,5 caractere no meio invalida o que veio.

Baud	1 caractere	Silencio	Origem
9600	1,042 ms	3,65 ms	calculado
19200	0,521 ms	1,82 ms	calculado
38400	0,260 ms	1,75 ms	fixo por norma
115200	0,087 ms	1,75 ms	fixo por norma

Acima de 19200 a norma abandona o cálculo e fixa 1,750 ms, porque o valor calculado ficaria menor que o tempo de processamento típico de um escravo. Usar o calculado nessas velocidades produz comunicação instável com equipamentos lentos.

3. Passo a passo

Passo 1 - Ligar o transceiver

Com tudo desenergizado, ligue o módulo conforme a tabela da seção 1. Confirme que DE e RE estão amarrados no mesmo GPIO; na maioria dos módulos isso é um jumper ou uma trilha pronta. Alimente em 3,3 V, salvo se colocou divisor no RO.

Passo 2 - Ligar no escravo

A e B do módulo vão para A e B do equipamento. Fabricantes divergem na nomenclatura: uns chamam de D+ e D-, outros de A e B, e há quem inverta a convenção.

Se não responder, troque A e B antes de mexer no código. É gratuito e resolve boa parte dos casos. Ligue também o terra de referência entre os dois lados.

Passo 3 - Levantar os parâmetros

Você precisa de quatro informações, no manual ou no display: endereço do escravo (1 a 247), baud rate, paridade e o registrador a ler. O padrão mais comum é 9600 8N1 no endereço 1.

O endereço do manual não é o do protocolo. Se o manual diz registrador 40001, no protocolo ele é o registrador 0. Subtraia 40001: o 40011 vira 10, o 40100 vira 99.

Passo 4 - Ajustar e gravar

Edite os parâmetros no topo do src/main.cpp e comece lendo **um registrador só**. Quantidade grande na primeira tentativa dificulta separar erro de parâmetro de erro de comunicação.

```
pio run -t upload
pio device monitor -b 115200
```

Passo 5 - Interpretar o resultado

Uma leitura bem-sucedida imprime o valor bruto. Compare com o display do equipamento: se bater, ou for um múltiplo exato de 10 ou 100, você acertou o registrador e a diferença é a escala, documentada no manual.

Modbus transporta inteiros de 16 bits. Um multimedidor que mostra 127,3 V costuma devolver 1273. Valores acima de 65535, como energia acumulada, ocupam dois registradores: o de menor endereço costuma ser a parte alta, mas há fabricante que inverte.

4. Terminação e barramento

Num barramento longo, o sinal que chega na ponta reflete de volta e interfere com o próximo bit. O resultado é comunicação que funciona na bancada e falha no painel, ou que funciona com um equipamento e para quando você liga o segundo.

Item	Regra
Terminação	120 ohms entre A e B nas DUAS extremidades, nunca no meio
Polarização	680 ohms puxando A ao positivo e B ao terra, contra ruído em repouso
Cabo	par trancado; dois fios paralelos anulam a vantagem do diferencial
Blindagem	aterrar a malha em UM extremo só, para não criar laço de terra
Roteamento	afastar dos cabos de potência, sobretudo de saída de inversor
Endereços	únicos por escravo; dois iguais derrubam o barramento

> Muitos módulos trazem o resistor de 120 ohms soldado ou num jumper. Se o seu ESP32 está no meio da linha e não na ponta, remova.

5. Solução de problemas

Sintoma	Causa provável	O que fazer
O escravo nunca responde	A/B invertidos ou DE solto cedo	Inverta A e B. Depois confirme o flush antes de baixar o DE.
Responde as vezes	Falta terminação ou pausa	120 ohms nas duas pontas e respeite o silêncio entre requisições.
CRC sempre invalido	Ordem dos bytes trocada	O byte baixo do CRC vai primeiro. Confira as duas últimas linhas.
Resposta com bit 0x80	Exceção do escravo	Não é erro seu: o byte seguinte diz o motivo. 0x02 é registrador inexistente.
Valor não bate com o display	Registrador errado ou escala	Verifique o deslocamento de 40001 e o fator no manual.
Funciona com um, para com dois	Terminação ou endereço repetido	Termine só nas pontas e confirme endereços distintos.
Ruído quando o inversor liga	Acoplamento eletromagnético	Par trancado blindado, malha aterrada num extremo, baixe o baud.
Sem resposta só em cabo longo	Baud alto demais para a distância	9600 é bem mais tolerante que 38400. Teste reduzindo.

6. Checklist final

	Item
[ ]	Transceiver de 3,3 V, ou divisor resistivo no RO
[ ]	DE e RE amarrados no mesmo GPIO
[ ]	Terra de referência ligado entre ESP32 e escravo
[ ]	120 ohms nas duas extremidades, e só nelas
[ ]	Par trancado, afastado dos cabos de potência
[ ]	Endereço, baud e paridade conferidos no manual do escravo
[ ]	Deslocamento de 40001 aplicado ao registrador
[ ]	Primeira leitura feita com um registrador só
[ ]	Valor lido bate com o display, considerada a escala
[ ]	Sem falha de CRC em varias leituras seguidas
[ ]	Intervalo entre requisições respeita o silêncio de 3,5 caracteres

Códigos de exceção mais frequentes

Código	Significado	Causa típica
0x01	Função não suportada	escravo não implementa 0x03; tente 0x04
0x02	Registrador inexistente	deslocamento de 40001 esquecido
0x03	Quantidade invalida	pediu mais registradores que o bloco permite
0x04	Falha interna do escravo	problema no equipamento, não no mestre
0x06	Escravo ocupado	aumente o intervalo entre requisições