Comparação Entre Protocolos de Comunicação Serial

Protocolos são utilizados em muitas áreas tecnológicas, pois possibilitam o transporte de informação entre dispositivos, estabelecendo regras e convenções que regem o funcionamento de diferentes comunicações. Por serem utilizados em muitas áreas, diversos protocolos foram desenvolvidos ao longo dos anos para atender da melhor forma possível diferentes aplicações. Alguns exemplos são os protocolos I²C, UART e SPI.

Existem dois formatos que os protocolos podem seguir, **serial** e **paralelo**. A comunicação serial envia e recebe toda a informação sequencialmente, o que permite que o número de fios seja menor. Por outro lado, a comunicação paralela é capaz de enviar e receber dados em mais de uma via de comunicação, ou seja, é capaz de enviar vários bits simultaneamente, resultando em maior rapidez na transmissão da informação.

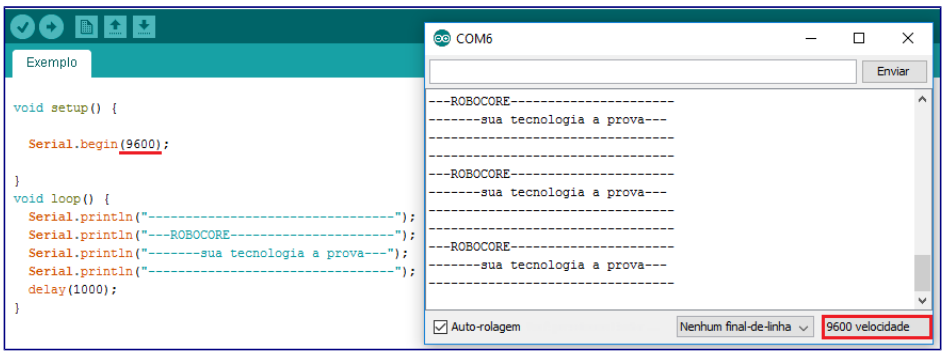
Neste tutorial serão apresentado alguns protocolos de comunicação serial e suas principais diferenças. Entretanto é necessário explicar as principais características que os protocolos têm.

CARACTERÍSTICAS

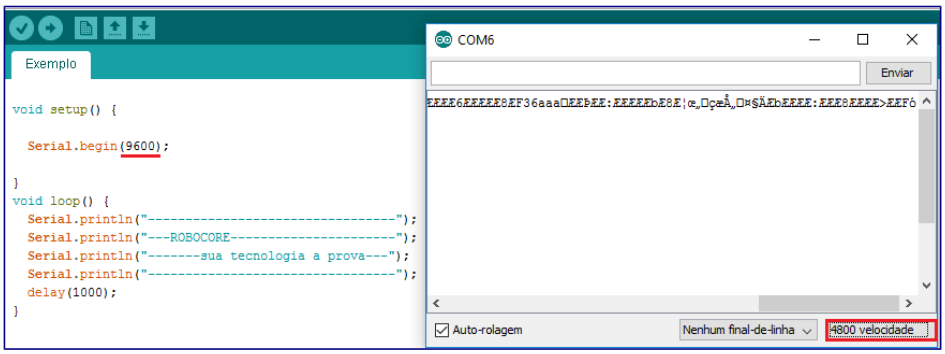
1. **Taxa de comunicação**

   Sua unidade geralmente é bits por segundo (bps) e representa a velocidade de uma comunicação. Por exemplo uma comunicação assíncrona com 9600 bps envia um bit em 0,0001 s. Esta taxa assume diferentes nomes dependendo da comunicação, como em comunicações síncronas que é chamada de "clock" ou em comunicações assíncronas que é conhecida como "Baud Rate".

   É importante lembrar que todos dispositivos ligados a mesma linha de comunicação devem estar com a mesma taxa de comunicação, principalmente comunicações assíncronas que não usam o clock como referência. Caso a taxa dos dispositivos seja diferente, dificilmente os dispositivos receberão os dados corretamente, como observado nas imagens abaixo



Dispositivos com MESMA taxa.
Fonte: elaborado pelo autor

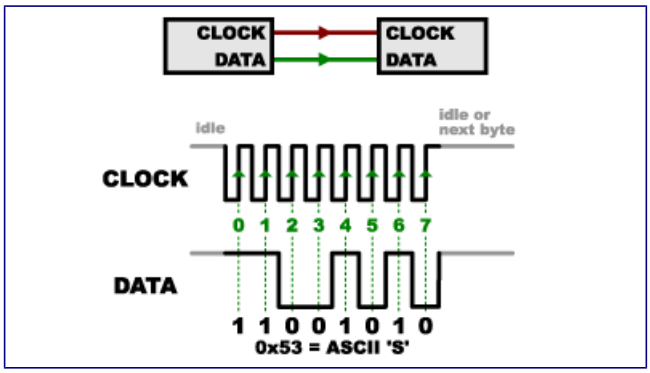


Dispositivos com taxas DIFERENTES.
Fonte: elaborado pelo autor

1. Métodos

2. Síncrono

   É o método de comunicação que depende de um sinal de "clock", ou seja, cada bit ou conjunto de bits enviado depende de um pulso do clock, tendo como principal vantagem sua velocidade de transmissão de dados, em contrapartida é necessario um fio extra para o clock.

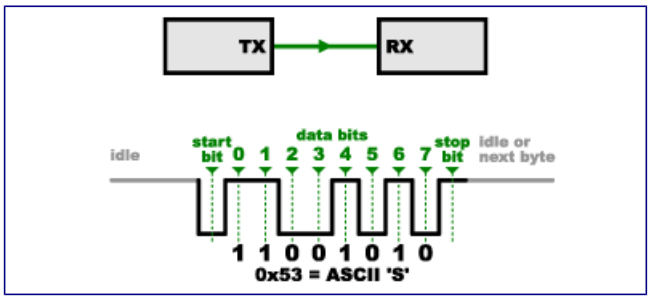


Fonte: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
Acesso em ago. 2016.

3. Assíncrono

   Ao contrário do método acima, este não precisa de um sinal de clock, portanto o número de fios necessários é menor. Contudo, o envio dos dados é mais complicado

e susceptível a erros, por isso alguns parâmetros são necessários para garantir o envio sem erros. Um parâmetro muito evidente em comunicações assíncrona é o Baud Rate que especifica a velocidade de recepção e envio, por isso é muito importante que os dois dispositivos utilizem a mesma taxa.



Fonte: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
Acesso em ago. 2016.

1. **Sentido de Transmissão**

4. Full-duplex

   Indica que o dispositivo pode transmitir e receber dados ao mesmo tempo.

5. Half-duplex

   O dispositivo que comunica dessa forma pode enviar ou receber mas não executa essas funções simultaneamente.

6. Simplex

   Se trata de dispositivos que sua comunicação é unidirecional, ou seja, apenas efetua o envio ou recebimento.

1. **Tensão do protocolo**

É a tensão que os protocolos identificam os níveis lógicos alto e baixo.

TERMINOLOGIA

- **RX/TX**

RX é o termo usado para representar o pino **receptor** de uma comunicação serial e TX representa o **transmissor**. Ao contrário dos pinos de GND ou VCC que são conectados com seus semelhantes, como GND --> GND, o RX/TX tem uma ligação diferente. O TX deve ser ligado no RX, ou seja, transmissor enviando para o receptor, e vice-versa.

- **Mestre e escravo**

É um método de comando centralizado onde apenas o dispositivo mestre pode iniciar uma comunicação, enviando comandos, controlando a taxa de comunicação, etc.

- **Nível lógico**

São os estados que um bit pode assumir, nível alto (1) ou nível baixo (0). Os níveis lógicos são interpretados pelos protocolos baseados nas tensões que recebe. Por exemplo o

protocolo TTL considera de 2V a 5V nível lógico alto (bit 1) e de 0V a 0,8V nível lógico baixo (bit 0).

PROTOCOLOS

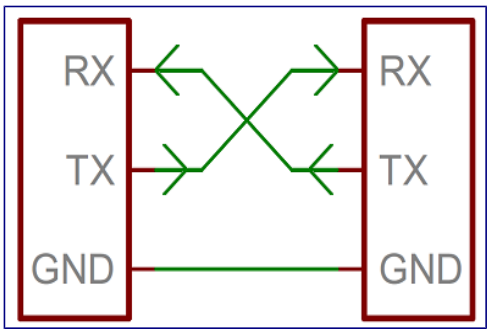
IMPORTANTE: A quantidade de fios ou pinos de cada protocolo é baseado no número de vias de comunicação, portanto é importante lembrar que todos o protocolos tem pinos de GND e alguns também tem pinos de VCC.

- **UART**

Vantagem: simplicidade do protocolo, full-duplex.

Funcionamento: o pino de transmissão (Tx) do protocolo envia um pacote de bits que será interpretado bit a bit pelo pino receptor. Cada pacote enviado contém 1 start bit que indica o início da mensagem, 1 ou 2 stop bits para indicar o final da mensagem, 5 a 9 bits de informação e 1 bit de paridade para evitar a recepção de erros.

Ligação: por ser uma comunicação assíncrona a comunicação é feita por dois pinos Rx/Tx que dependem do baud rate como referência.



Fonte: https://learn.sparkfun.com/tutorials/serial-communication
Acesso em ago. 2016.

O que usa este protocolo?

É um protocolo utilizado por muitos microcontroladores, pois é responsável pela conversão da comunicação paralela em serial, que na maioria das vezes é convertida em outro protocolo como por exemplo o controlador da placa Blackboard ou Arduino Uno, que utiliza o protocolo UART mas tem o protocolo convertido para USB.

- Placa RC FTDI V1.1
- BlackBoard V1.0
- BlackBoard Pro Mini - 5V/16MHz

Exemplo de aplicação

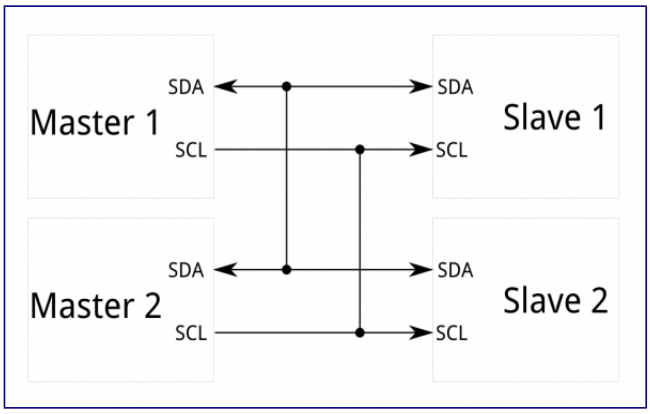
- Comunicação entre Arduinos: UART

- **I²C/TWI**

Vantagem: possibilita comunicar com vários dispositivos utilizando poucos fios, além de possibilitar que mais de um mestre controle os escravos.

Funcionamento: para que a informação seja enviada, o dispositivo mestre deve informar aos dispositivos escravos o início da comunicação, ou "Start condition". Neste caso o pino SCL deve estar em nível lógico alto e o pino SDA em nível lógico baixo. Quando isso ocorrer, todos os escravos estarão prontos para receber a primeira informação que é o endereço do escravo que comunicará com o mestre, junto com a operação que este escravo desempenhará. Em situações em que houver mais de um mestre na comunicação, terá preferência o mestre que sinalizar mais rápido o inicio de uma transmissão. Depois que o endereço é enviado, o escravo que tiver o endereço correspondente realizará a operação de leitura ou escrita da informação até que o dispositivo mestre envie uma "stop condition" para interromper a comunicação.

Ligação: este protocolo utiliza apenas dois pinos, SDA que é o sinal de dados e SCL o clock. Com isso é possível concluir que este protocolo é half-duplex, pois contém apenas um pino para envio de dados, e síncrono, pois usa um pino de clock.



Fonte: https://learn.sparkfun.com/tutorials/i2c
Acesso em ago. 2016.

O que usa este protocolo?

- LCDs 16X2 com protocolo I²C
- Acelerômetro de 3 Eixos MMA8452Q
- Sensor de Temperatura Infravermelho - MLX90614

Exemplo de aplicação:

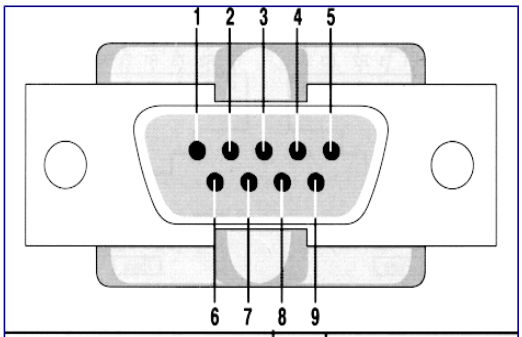
- Comunicação entre Arduinos: I2C - Parte 1

- **RS 232**

Vantagem: a distância que o protocolo alcança é maior que a do protocolo UART, além de ser um protocolo full-duplex.

Funcionamento: seu funcionamento é similar ao do protocolo UART, ou seja, é enviado 1 start bit + 8 bits de dados + 1 stop bit. A diferença está nas tensões que o protocolo utiliza e a possibilidade de adicionar mais pinos para checar as informações enviadas.

Ligação: existem dois padrões de conectores que o protocolo adota, com 25 pinos (DB-25) e 9 pinos (DB-9), que basicamente usam os pinos TD (Transmitted Data) e RD (Received Data) para realizar a troca de informação e mais 7 pinos (padrão DB-9) para melhorar a confiabilidade da comunicação controlando o fluxo da comunicação.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Fonte:https://www.arduino.cc/en/Tutorial/ArduinoSoftwareRS232
Acesso em ago. 2016.

O que usa este protocolo?

- Sensor ultrasônico - Maxbotix LV-EZ0

Exemplo de aplicação:

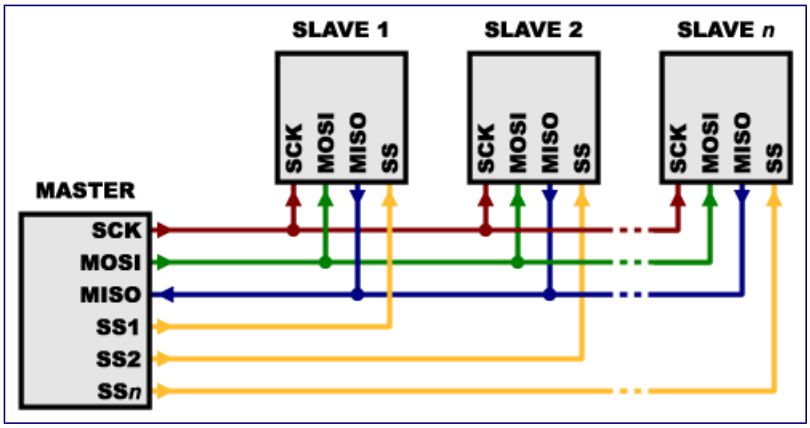
- Comunicação entre PC e Arduino com RS-232

- **SPI**

Vantagem: não há limite para o número de escravos, a comunicação é full-duplex e possui boa velocidade de comunicação.

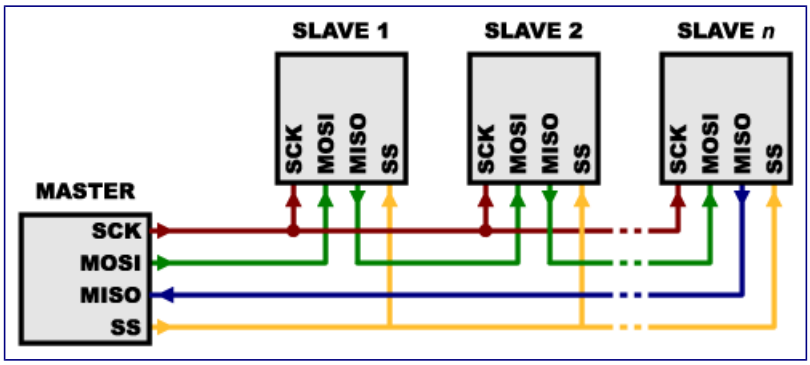
Funcionamento: primeiramente o mestre gera um clock e seleciona através do pino SS com qual dispositivo será efetuada a comunicação. Em seguida os dados são enviados para o dispositivo de destino pelo pino MOSI e então o dispositivo escravo envia uma resposta (se necessário) ao mestre pelo pino MISO.

Ligação: existem dois tipos de ligação para o protocolo SPI, ligação paralela e ligação em cascata.

- Ligação paralela É a ligação mais comum deste protocolo que utiliza 1 pino de MOSI, MISO e clock para todos dispositivos e 1 pino SS para cada escravo ligado a comunicação. 

Fonte: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
Acesso em ago. 2016.

- Ligação em cascata Este tipo de ligação demanda menos pinos de comunicação, pois é utilizado 1 pino SS para todos os dispositivos ligados na comunicação, porém causa perda de velocidade na comunicação.



Fonte: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
Acesso em ago. 2016.

O que usa este protocolo?

- Matriz de LEDs RGB

Exemplo de aplicação:

- Comunicação entre Arduinos: SPI

- **1Wire**

Vantagem: utiliza apenas 1 pino para a comunicação, cada dispositivo tem um endereço único.

Funcionamento: este protocolo funciona em 3 fases. A primeira fase é responsável por habilitar a comunicação e identificar os escravos ligados, a segunda fase seleciona qual escravo receberá os comandos e, por fim, na terceira fase ocorre a leitura ou escrita de dados.

Ligação: pode ser feita de duas formas, parasita ou com alimentação externa.

- Ligação Parasita: com esta ligação a alimentação de um dispositivo 1-wire é realizado com o próprio pino de comunicação, como pode ser visto na imagem a seguir. 

Fonte: elaborado pelo autor

- Ligação Alimentação Externa: é mais indicado porque garante uma comunicação mais confiável, principalmente quando os dispositivos então afastados a mais de 6 m de distancia do mestre. 

Fonte: elaborado pelo autor

O que usa este protocolo?

- Sensor de Temperatura Digital DS18B20

Exemplo de aplicação:

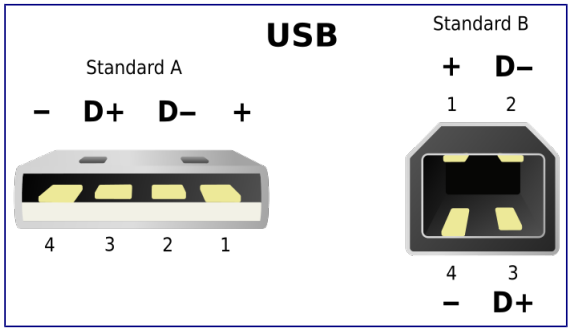
- Leitura de sensor de temperatura

- **USB 2.0**

Vantagem: muitos dispositivos usam este protocolo, velocidade alta, possibilita a comunicação com vários dispositivos.

Funcionamento: os dispositivos que usam este protocolo precisam enviar 3 pacotes para realizar o envio de dados. Primeiro é enviado um "Token Packet" que informa o que será realizado na comunicação, se a informação será escrita ou lida e o endereço do dispositivo ao qual a mensagem será direcionada. Segundo, um "Data Packet" que é o pacote de dados que será escrito ou lido dependendo do comando dado no "Token Packet". E por último um "Handshaking Packet" que informa se os dois primeiros pacotes foram enviados corretamente.

Ligação: apesar do protocolo utilizar dois pinos (D+ e D-), a versão 2.0 do protocolo é half-duplex, pois utiliza o método "differential signaling" que corrige possíveis interferências de ruídos, por outro lado a versão 3.0 do protocolo é full-Duplex e por consequência utiliza um número maior de pinos.



Fonte:https://en.wikipedia.org/wiki/USB
Acesso em ago. 2016.

O que usa este protocolo?

- BlackBoard V1.0
- Placa RC FTDI V1.1
- Xbee Explorer Stick

RESUMO

PROTOCOLO	Taxa de comunicação (bps)	Sentido de Transmissão	Método	Nº Fios	Tensão do protocolo (V)	Max de dispositivos comunicando
UART	1200 a 115200	Full-Duplex	Assíncrono	2	0 a 5	1
SPI	0 a 10M (depende do dispositivo)	Full-Duplex	Síncrono	3 + (1 para cada slave)	0 a 5V	não há
I²C	100k ou 400k	Half-Duplex	Síncrono	2	0 a 5V	127 ou 1024
RS 232	1200 a 115200	Full-Duplex	Síncrono ou Assíncrono	2	-25 a +25	1
1 wire	0 a 16,3k ou 0 a 142k	Half-Duplex	Assíncrono	1	2,8 a 6,0	2^{56}
USB	1,5M a 4,8G	Half-Duplex (2.0) ou Full-Duplex (3.0)	Assíncrono	2	0 a 20	127

Link: https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html