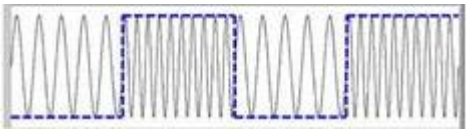


CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto

TRANSMISSÃO SERIAL DE DADOS

A quantidade de bytes transmitidos por minuto no planeta é gigantesca e cresce constantemente. Podemos estimar esse valor com base no tráfego global de internet. De acordo com relatórios da Cisco e outras fontes sobre o tráfego global de dados, em 2023 o tráfego de internet mundial foi aproximadamente de 500 exabytes (EB) por mês, o que nos dá 11,6 petabytes por minutos. E essa quantidade de bytes transmitidos não leva ainda em conta os bytes transmitidos entre elementos de sistemas menores, como dados trocados entre microcontroladores e sensores, atuadores... ou mesmo ainda transmitidos internamente dentro de arquiteturas como os microcontroladores. Grande parte de todos esses dados são transmitidos serialmente. Uma outra forma, mais rápida e menos robusta, seria a transmissão paralela.

Na transmissão serial, os bits são enviados um de cada vez, sequencialmente, por um único canal de comunicação. Esse método é mais lento em comparação à transmissão paralela, mas é mais confiável para comunicações a longa distância. Além disso, a transmissão serial requer menos fios ou pistas condutoras, apresenta menor interferência eletromagnética, e permite comunicação eficiente em distâncias maiores. Exemplos: comunicação em portas USB, RS-232, Ethernet, e comunicação de redes.

Na transmissão paralela, vários bits são transmitidos simultaneamente através de múltiplos canais de comunicação (normalmente 8, 16 ou 32 bits de uma vez). Esse método é mais rápido, mas pode sofrer problemas de distorção de sinal e interferência quando usado em distâncias longas.

A transmissão serial de dados pode ocorrer de diversas formas, dependendo do meio físico, do protocolo utilizado e do contexto da comunicação. Podemos classificar a transmissão de dados seriais segundo vários critérios, como mostrado a seguir:

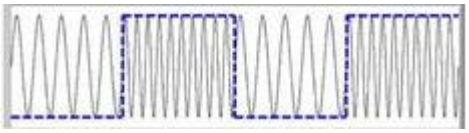
1. Quanto ao Meio de Transmissão

- Transmissão Guiada (com fio): Utiliza cabos físicos para transmitir dados.
    - *Par trançado*: Muito usado em redes Ethernet (ex.: cabos de rede RJ-45).
    - *Cabo coaxial*: Utilizado em redes antigas e televisão a cabo.
    - *Fibra óptica*: Oferece alta velocidade e baixa perda de sinal, muito usada em redes de longa distância.
- Transmissão Não Guiada (sem fio): Envia dados por ondas eletromagnéticas.
    - *Wi-Fi*: Redes locais sem fio, muito comuns em residências e empresas.
    - *Bluetooth*: Comunicação de curto alcance entre dispositivos.
    - *Rádio e Satélite*: Usado para telecomunicações, como telefonia móvel e TV via satélite.
    - *Infravermelho*: Tecnologias mais antigas, usadas em controles remotos.

2. Quanto à Direcionalidade

- Simplex: A transmissão ocorre em apenas um sentido (ex.: rádio, TV).
- Half-duplex: A comunicação ocorre nos dois sentidos, mas não simultaneamente (ex.: walkie-talkie).
- Full-duplex: Transmissão simultânea nos dois sentidos (ex.: ligações telefônicas).

3. Quanto ao Método de Transmissão



- Serial: Os bits são transmitidos um de cada vez, sequencialmente. Muito comum em conexões USB, portas seriais e comunicação de longa distância.
- Paralela: Vários bits são transmitidos simultaneamente, comum em barramentos de computadores e memórias.

4. Quanto ao Modo de Sincronização

- Síncrona: Os dados são transmitidos continuamente em blocos, sincronizados por um relógio comum (ex.: redes Ethernet).
- Assíncrona: Cada caractere é transmitido separadamente, com bits de início e parada (ex.: comunicação via portas seriais RS-232).

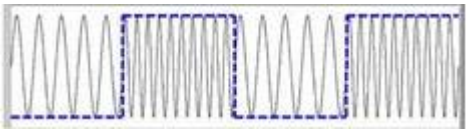
5. Quanto à Forma de Modulação

- Analógica: O sinal é contínuo e pode ser modulado por amplitude (AM), frequência (FM) ou fase (PM).
- Digital: O sinal é transmitido em formato binário, podendo ser modulados por ASK (modulação por amplitude), FSK (modulação por frequência) ou PSK (modulação por fase).

Cada uma dessas formas de transmissão é escolhida com base na aplicação desejada, levando em conta fatores como velocidade, confiabilidade e custo.

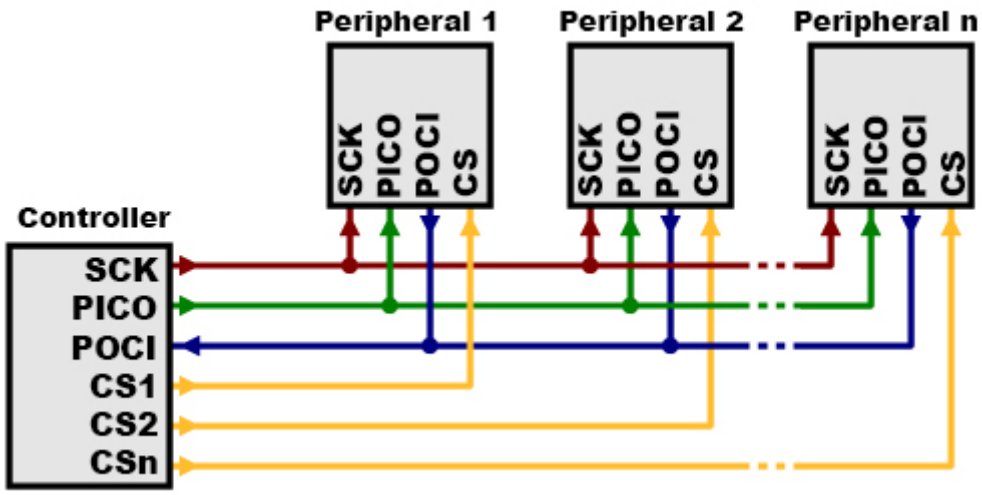
Quando estudamos ou projetamos um sistema de transmissão de dados organizamos o projeto em camadas. Esse conceito de arquitetura em camadas, é utilizado para organizar a comunicação entre dispositivos de forma modular e estruturada. Cada camada tem uma função específica e interage apenas com as camadas adjacentes, facilitando o desenvolvimento, manutenção e padronização dos sistemas de rede. Por exemplo, um projeto ou modelo de transmissão entre sistemas deve estabelecer a estrutura física da transmissão, como quais cabos, quais níveis de tensão elétrica representando zeros e uns, quais conectores etc. Em uma camada acima, deve-se projetar como os hardwares dessa rede de comunicação devem acessar o meio físico e gerar sinais para transmitir os bits, como checar que o bit foi recebido, como obedecer ao controle de tráfego etc. Numa camada superior, deve ser definido uma estratégia de endereçamento e controle de rotas para que cada parte dos dados tenha o destino certo. As camadas podem ser criadas até que cheguemos na camada superior que acaba sendo a aplicação utilizada pelo usuário. Um exemplo de estruturação em camadas para transmissão de dados seria:

Camada	Função Principal
7. Aplicação	Interface com o usuário e aplicações (HTTP, FTP, SMTP).
6. Apresentação	Formatação, criptografia e compressão de dados.
5. Sessão	Controle da comunicação entre dispositivos (inicia e encerra conexões).
4. Transporte	Controle do fluxo de dados, segmentação e confiabilidade (TCP, UDP).
3. Rede	Roteamento e endereçamento IP (IPv4, IPv6).
2. Enlace de Dados	Correção de erros e controle de acesso ao meio físico (Ethernet, Wi-Fi).
1. Física	Transmissão real dos bits pelo meio físico (cabos, ondas de rádio, fibra óptica).



Um primeiro exemplo: Transmissão SPI

SPI (Serial Peripheral Interface) é um protocolo de comunicação serial síncrono usado para a troca de dados entre microcontroladores e dispositivos periféricos, como sensores, displays, memórias e conversores AD/DA. Ele foi desenvolvido pela Motorola nos anos 1980 e é amplamente utilizado devido à sua alta velocidade e simplicidade. Estamos aqui falando de uma camada bem básica!



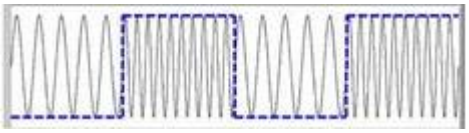
Características Principais do SPI

1. Comunicação Mestre-Escravo
    - Um dispositivo mestre (ex.: microcontrolador) controla a comunicação.
    - Um ou mais escravos (ex.: sensores, memórias Flash) respondem ao mestre.
2. Transmissão Síncrona
    - Usa um clock (SCK) para sincronizar os dados entre os dispositivos.
3. Alta Velocidade
    - Pode operar em frequências de vários MHz, sendo mais rápido que protocolos como I^2C.
4. Modo Full-Duplex
    - Os dados podem ser transmitidos e recebidos simultaneamente, aumentando a eficiência.

Pinos do SPI

O protocolo SPI usa quatro sinais principais:

Pino	Nome	Função
SCLK	Serial Clock	Pulso de clock gerado pelo mestre para sincronizar a comunicação.
MOSI	Master Out, Slave In	Linha onde o mestre envia dados para o escravo.
MISO	Master In, Slave Out	Linha onde o escravo envia dados de volta para o mestre.



Pino	Nome	Função
SS (ou CS)	Slave Select (Chip Select)	Pino que habilita o escravo específico a se comunicar.

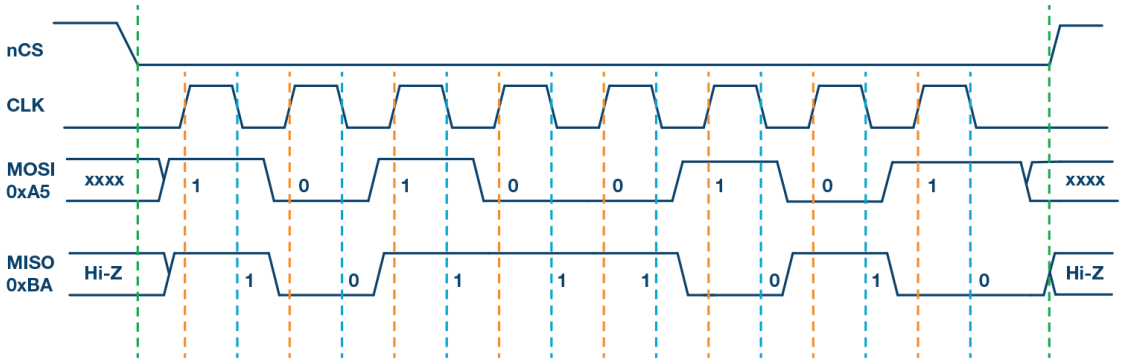
Onde o SPI é Utilizado?

- Displays (OLED, TFT, e-paper).
- Memórias Flash e EEPROMs.
- Sensores de alta velocidade (acelerômetros, giroscópios).
- Conversores AD/DA.
- Módulos de comunicação (RFID, Wi-Fi, LoRa).

O SPI é amplamente usado em sistemas embarcados devido à sua eficiência e velocidade, sendo uma escolha comum para aplicações que exigem transmissão rápida de dados entre dispositivos eletrônicos.

Gráfico de uma transmissão SPI

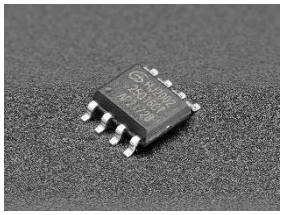
Repare no gráfico abaixo que a transmissão entre um mestre e um escravo foi configurada da seguinte maneira: após cada subida do sinal de clock, o mestre lê o sinal produzido pelo escravo (sinal MISO). Após cada descida do sinal de clock, o escravo lê o valor do sinal produzido pelo mestre (MOSI). Os números no gráfico abaixo mostram os valores lidos.

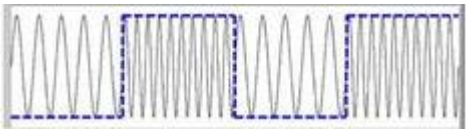


Esses valores lidos constituem bits transmitidos. Como esses bits devem ser transmitidos para formar bytes e informações completas, bem, isso faz parte de uma camada superior!

Chip SPI

O hardware responsável pro gerar o sinal nessa comunicação é o chip SPI. Tal elemento recebe os dados provenientes de uma camada superior e o sequencializa e o transmite bie a bit para outro chip. Quem faz a geração dos dados a serem entregues ao chip também já é uma camada superior. Você pode pensar nesse chip como um veículo que transporta uma encomenda, respeitando os limites de velocidade, sinais de trânsito, utilizando veículos licenciados etc. Quem produz a encomenda, faz parte de uma camada superior.





Outro exemplo: Transmissão UART

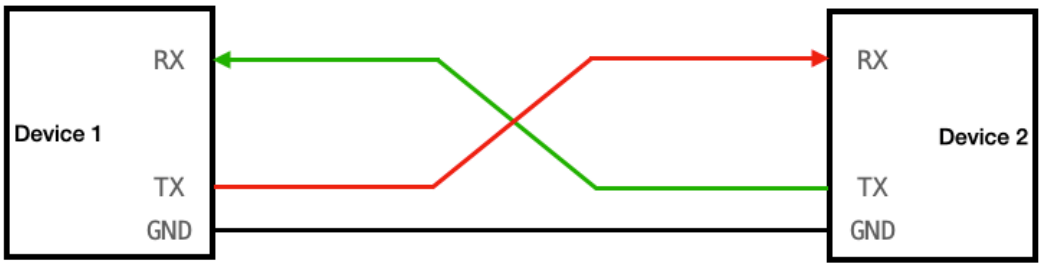
Ainda estamos falando de uma camada bem inferior. Básica. Que pode ser aplicada em uma estratégia complexa, com diversas camadas (como faremos nos próximos projetos).

UART (Universal Asynchronous Receiver-Transmitter) é um protocolo de comunicação serial assíncrono usado para a transmissão de dados entre dispositivos eletrônicos. Ele é amplamente utilizado em sistemas embarcados para comunicação entre microcontroladores, sensores, módulos GPS, Bluetooth, Wi-Fi e outros periféricos.

A comunicação UART envolve dois dispositivos e ocorre por meio de dois fios principais:

- TX (Transmitter – Transmissor): Linha que envia os dados.
- RX (Receiver – Receptor): Linha que recebe os dados.

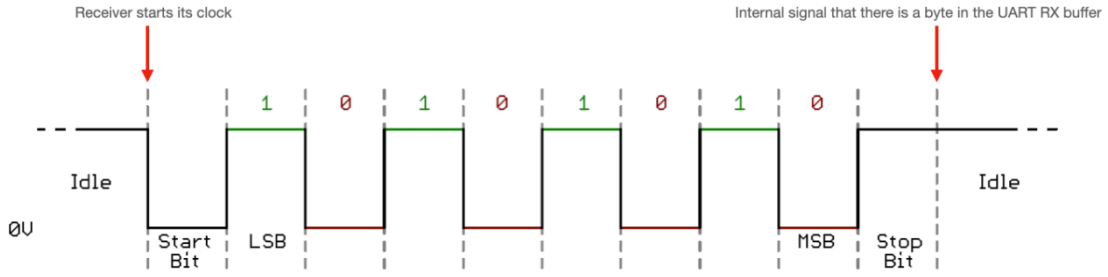
Para que a comunicação funcione, o TX de um dispositivo deve estar conectado ao RX do outro, e vice-versa.



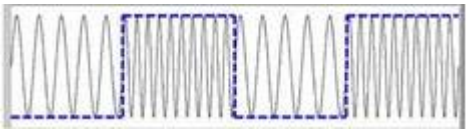
Quando um dado é enviado via UART, ele segue uma estrutura padronizada chamada quadro de transmissão (frame). Esse quadro define como os bits são organizados para que o receptor possa entender a mensagem corretamente.

Cada byte (8 bits de dados) é enviado com bits adicionais para controle, garantindo a integridade da transmissão.

O gráfico abaixo mostra a transmissão de um byte via UART. Repare que para informar que a transmissão irá começar, um bit zero é enviado. Esse bit é chamado de start bit. Depois os bits de dados são enviados, começando pelo bit menos significativo. O tempo para o envio de cada bit deve ser pré acordado entre os elementos. Esse tempo define o número de bits por segundo que podem ser enviados, denominado baud rate.



A velocidade da comunicação UART é então medida em baud rate (bits por segundo). Os dispositivos precisam estar configurados com a mesma taxa de transmissão para que a comunicação funcione corretamente.



Taxas de baud comuns:

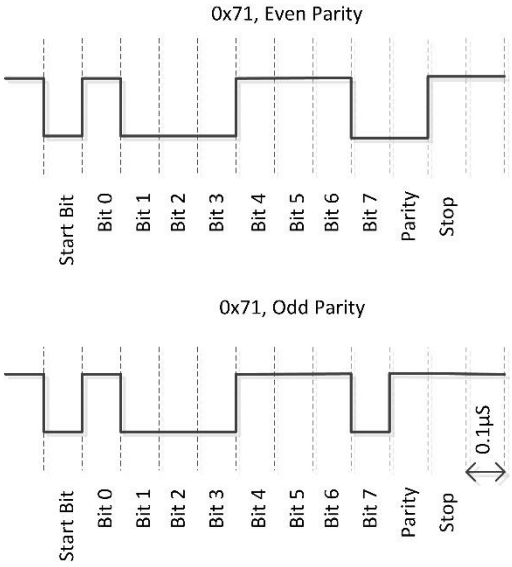
- 9600 bps (padrão em muitas aplicações)
- 19200 bps
- 38400 bps
- 57600 bps
- 115200 bps (usado para comunicação rápida e depuração)
- 1 Mbps ou mais (em algumas implementações avançadas)

Importante: Se os dispositivos estiverem configurados com baud rates diferentes, os dados recebidos serão corrompidos!

O chip UART, que realiza a transmissão dos dados, tem a capacidade de detectar erros de transmissão. Para isso a transmissão pode ser feita com o envio de um bit a mais: o bit de paridade.

A transmissão pode ser configurada de 3 diferentes modos:

- Paridade Par (Even): O número total de bits "1" (incluindo os de dados) deve ser par.
- Paridade Ímpar (Odd): O número total de bits "1" deve ser ímpar.
- Sem Paridade (None): Não há verificação por paridade.



Se o bit de paridade estiver trocado, isso significa que um dos bits foi invertido durante a transmissão. Assim, o chip acusará um erro de paridade: O bit de paridade recebido não corresponde ao esperado.

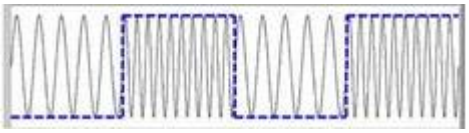
Além disso outros erros podem ser detectados pelo chip:

- Framing Error: O receptor não encontra o bit de stop esperado.
- Overrun Error: O receptor recebe dados mais rápido do que consegue processá-los.

Se um erro for detectado, o sistema pode descartar os dados ou solicitar uma retransmissão. Mas isso deve ser programado em uma camada superior!

Chip UART

Atualmente qualquer arquitetura de microcontroladores possui como periférico um chip UART, que poderá sequenciar e transmitir bytes serialmente. Quem gera, organiza e se comunica com o chip para que os dados sejam enviados, é uma camada superior! Em nosso primeiro projeto, trabalharemos com essa camada superior ao chip e linha de transmissão. E utilizaremos a porta USB de seu computador (fará o papel do chip) para produzirmos uma transmissão serial UART.



Um terceiro exemplo: Transmissão I2C

A transmissão I²C (Inter-Integrated Circuit) é um protocolo de comunicação serial desenvolvido pela Philips (hoje NXP) que permite a comunicação entre dispositivos eletrônicos em um sistema. O I²C é bastante utilizado para comunicação entre microcontroladores, sensores, memórias e outros componentes em sistemas embarcados devido à sua simplicidade e economia de pinos de conexão.

A transmissão I²C é baseada em dois fios principais:

1. SCL (Serial Clock Line) – Linha de clock, usada para sincronizar a comunicação entre os dispositivos.
2. SDA (Serial Data Line) – Linha de dados, utilizada para transmitir os dados entre os dispositivos.

Essas linhas são compartilhadas entre todos os dispositivos conectados no barramento I²C. Para que a comunicação seja realizada, um dispositivo age como mestre (master) e os outros como escravos (slaves).

Características principais:

- Mestre e escravo: O dispositivo mestre controla o barramento e inicia a comunicação, enquanto os escravos respondem conforme solicitado. Cada dispositivo tem um endereço único.
- Endereçamento: Cada dispositivo conectado ao barramento tem um endereço único, normalmente de 7 ou 10 bits. O mestre usa esse endereço para se comunicar com um dispositivo específico.
- Taxa de transferência: A velocidade de transmissão do I²C pode variar, com padrões comuns de 100 kbit/s (modo padrão), 400 kbit/s (modo rápido) e até 1 Mbit/s (modo rápido-plus).
- O I²C é um sistema half-duplex, ou seja, os dados podem ser transmitidos ou recebidos, mas não simultaneamente.
- Tensões: O I²C normalmente usa níveis de tensão de 3,3V ou 5V, dependendo dos dispositivos conectados.

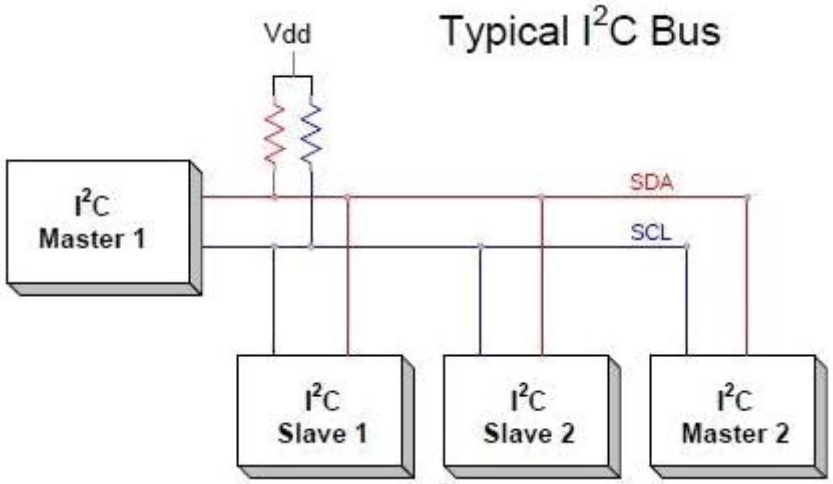
Como ocorre a comunicação?

1. O mestre envia um endereço do escravo para indicar a quem quer se comunicar.
2. O escravo com o endereço correspondente responde com um acknowledge (ACK) para confirmar a recepção.
3. Em seguida, o mestre envia os dados (ou comandos), e o escravo responde conforme necessário.
4. Se a transmissão for bem-sucedida, o escravo envia um ACK. Caso contrário, ele envia um NACK para indicar um erro ou falha na comunicação.

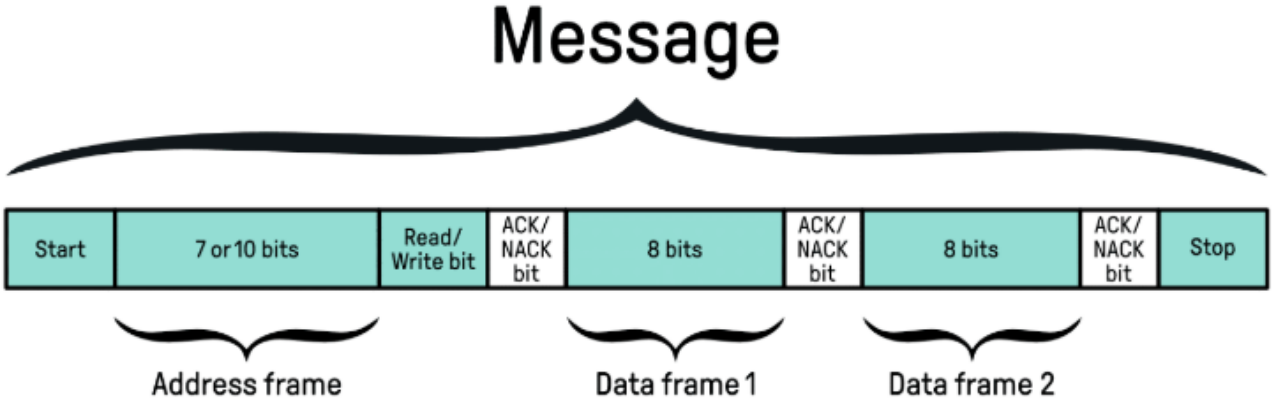
Vantagens do I²C:

Embora a comunicação I²C seja de fácil implementação e expansibilidade (por exemplo, até 128 elementos se utilizados 7 bits como identificador), essa forma de transmissão serial é mais lenta não ideal para distâncias maiores, pois o barramento é compartilhado. Além disso, a velocidade é inferior a outros protocolos de comunicação como SPI ou UART.

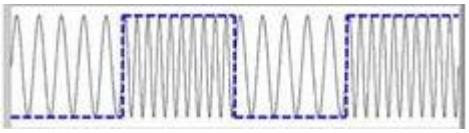
Essa transmissão é especialmente popular em sistemas embarcados como sensores, displays e outros dispositivos periféricos de baixo custo, onde a simplicidade e a flexibilidade são essenciais.



A transmissão com esse protocolo funciona da seguinte maneira: após o start bit, de 7 a 10 bits são produzidos para definir o destinatário. Após isso, o bit seguinte é o R/W (read/write). Se o R/W for 0 significa escrita, ou seja, o mestre quer que o escravo escreva um byte. No contrário, se esse bit for 1 significa leitura, ou seja, o mestre quer que o escravo escreva receba o byte. Após o bit R/W, o escravo produz um bit para acknowlodge, sendo 0 significa que está disponível para se comunicar, se for 1 é um not acknowlodge (algo errado com o identificador ou escravo indisponível). Em seguida vêm os bits de dados seguidos de mais um bit acknowlodge ou not acknowlodge, indicando se a transmissão foi realizada com sucesso ou houve alguma incoerência. O tratamento desse bit deverá ser feito em camadas superiores. Para finalizar, há um bit de parada.

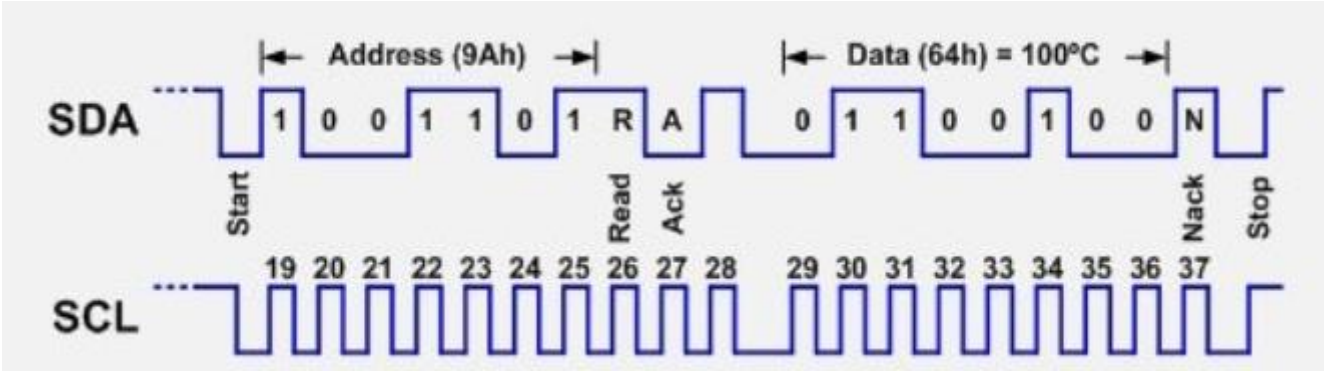


Exemplo do byte 01100100 (em decimal = 100) sendo transmitido para o escravo identificado como 1001101. Ao final da transmissão do byte, o not acknowlodge (nack) pode ser usado também para indicar que a “conversa” entre as duas partes acabou, não haverá mais dados.



CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto



Chip I2C

Quem gera, organiza e se comunica com o chip para que os dados sejam enviados, é uma camada superior, como nos outros protocolos!

Outros protocolos

Existem diversos protocolos de comunicação serial. Você pode usar sua aplicação favorita de IA para pesquisar sobre algumas, como RS 232, RS 485, CAN ...

PROJETO 1 - LOOP BACK

Neste projeto você deverá construir um código em Python para transmissão e recepção serial simultâneas!

O software será uma camada entre o usuário e o chip UART, que irá sequenciar os bits de cada byte seguindo o devido protocolo. Para isso voce usará os 5 arquivos fornecidos. Por ora, você deverá apenas editar o arquivo "aplicação.py". Os demais arquivos farão o trabalho de fornecer ao chip UART do seu computador os bytes que voce deseja enviar através da função de envio e também disponibilizar os dados recebidos.

Os arquivos fornecidos já estão prontos para o envio e recebimento de uma pequena mensagem de 4 bytes, ou seja, se voce rodar o main, 4 bytes serão enviados, rebatidos pelos arduínos (conforme explicado em aula) e recebido pela aplicação, sendo printados.

Leia com atenção os comentários no código.

Objetivo:

Ao rodar seu arquivo aplicação, o seu software deve:

1) Enviar uma imagem (a menor possível) através da porta de comunicação serial.
2) Receber a imagem simultaneamente ao envio e salvá-la como uma cópia. Para isso a recepção do Arduino (*pino rx*) deve estar curto-circuitada com o pino de transmissão (*pino tx*).
3) Adquirir compreensão do código base de transmissão UART.

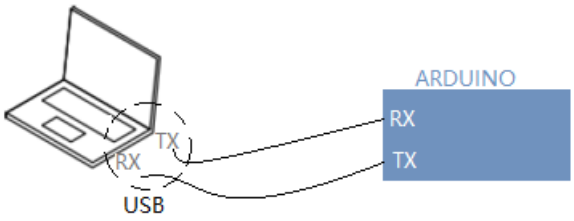
Material:

- Você irá utilizar um Arduino, um computador e 5 arquivos código fornecidos.

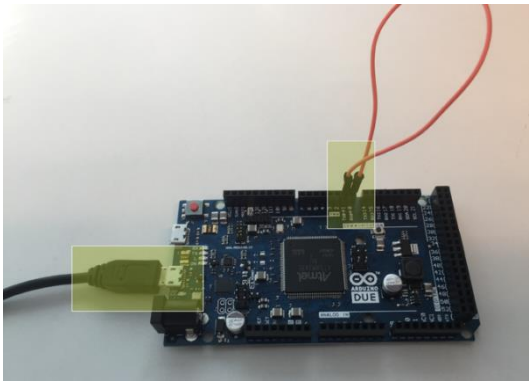
- No console python, 'pip install pyserial'
- Verificar em gerenciador de dispositivos qual porta COM está o Aduino. Ajustar o arquivo aplicação para essa porta!

Montagem:

Seu sistema operacional irá abrir uma porta de comunicação serial com o Arduino. Através dessa comunicação você terá acesso a tudo que o Arduino enviar ao seu computador (saindo do pino TX do Arduino e RX da sua porta USB). Tudo que seu computador enviar ao Arduino (saindo do pino TX da sua USB em seu computador). Desta maneira, ao conectar o Arduino ao seu computador, o pino de envio do seu computador (TX) estará conectado ao pino de recepção do Arduino (RX). Da mesma forma, o pino de envio do Arduino (TX) estará conectado ao pino de recepção do seu computador (RX).

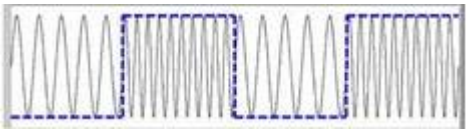


O queremos é que ao enviarmos uma mensagem (lista de bytes ou bytearray) ao Arduino, este responda com os mesmos bytes. Queremos que o Arduino seja um espelho para os bytes enviados. Para isso basta que conectemos o pino TX do Arduino ao seu pino RX! Assim, ao enviarmos uma sequência de bytes para o RX do Arduino, a mesma sequência de zeros e uns são produzidas no pino de envio do Arduino e recebidos de volta em seu computador. Observe a figura:



Você precisará verificar qual os pinos TX e RX de seu Arduino.

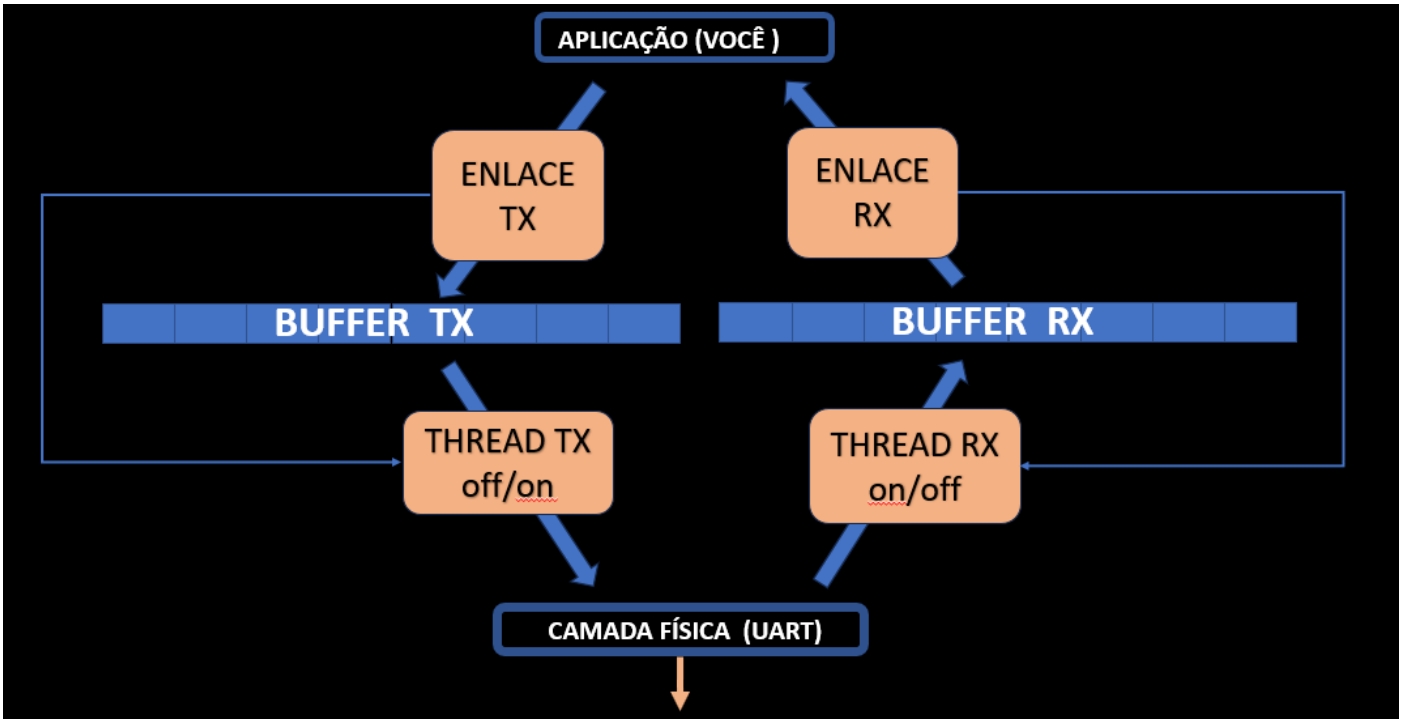
ATENÇÃO! ALGUNS ARDUINOS (UNO) PRECISAM FICAR COM O BOTAO DE RESET PRESSIONADO. OU O PINO RESET ATERRADO!



CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto

A estrutura da camada enlace:



O maior desafio e objetivo do projeto 1 é que você entenda como as funções das classes fazem o trabalho de envio e recebimento full-duplex representado nesse esquema. Você deve seguir as funções envolvidas no envio e recebimento de um bytearray. Se familiarizar com elas. Ser capaz de utiliza-las, modifica-las e responder a algumas perguntas sobre elas. Para isso, não tem outro modo a não ser se debruçar sobre as funções, seguindo todas as chamadas de métodos envolvidos no recebimento “enlace.get()” e recebimento emlace.send()”.

Imagens em python.

Você terá que transformar uma imagem em uma lista de bytes. No mesmo modo, salvar a lista de bytes recebida, como imagem. Para isso pode se basear nos seguintes trechos de código:

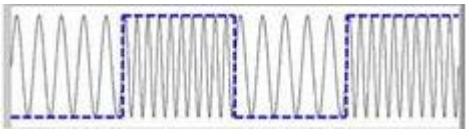
Caminhos das imagens

```
# Endereco da imagem a ser transmitida
imageR = "./imgs/image.png"

# Endereco da imagem a ser salva
imageW = "./imgs/recebidaCopia.png"
```

Lista de bytes com a imagem a ser transmitida

```
# Carrega imagem
print ("Carregando imagem para transmissão :")
print (" - {}".format(imageR))
print("-------------------------")
txBuffer = open(imageR, 'rb').read()
```



Escreve arquivo cópia

```
print ("Salvando dados no arquivo :")
print (" - {}".format(imageW))
f = open(imageW, 'wb')
f.write(rxBuffer)

# Fecha arquivo de imagem
f.close()
```

Pronto! Consegui!

Se você conseguiu fazer o arquivo cópia através da comunicação serial e este arquivo abre normalmente, parabéns! Vamos aproveitar então para explorar um pouco mais nossas classes. Iremos fazer um estudo mais profundo das classes na aula 2, mas já é necessário que entendam as funções e estrutura dos arquivos.

Entrega

Os projetos dessa disciplina serão sempre avaliados presencialmente. Um dos seus professores observar uma breve apresentação feita por você e sua dupla e lhes fazer algumas perguntas. Isso ocorrerá em um momento solicitado por você, dentro de uma data limite. Caso você solicite a apresentação após a data limite, você será penalizado.

Conceito C: Mostrar a transmissão e recepção da imagem ocorrendo corretamente.

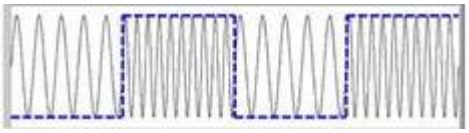
Conceito B: responder as questões feitas pelo seu professor (no momento da apresentação) a respeito das seguintes funções:

- `getBufferLen`
- `getAllBuffer`
- `getBuffer`
- `getNData`
- `sendBuffer`

Conceito B+: Além dos conceitos C e B, tente entender o que significa cada um dos 10 termos presentes na comunicação UART:

1. Transmissão assíncrona
2. UART – Start bit
3. UART – Stop bit
4. UART – TX, RX, GND
5. UART – Baud rate
6. UART – Bit rate
7. UART – Buffer
8. UART – Frame
9. UART – Bit de Paridade
10. UART – CRC

Conceito A+: A função *getStatus* não está funcionando corretamente. Verifique o que essa função deveria retornar e encontre uma solução para corrigi-la.



CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto

Data limite: 13/02 - Após esse período sua nota terá uma redução de 25% a cada semana.

Leituras:

Para se aprofundar, voce poderá ler sobre transmissão serial UART nos seguintes links:

https://docs.freebsd.org/pt-br/articles/serial-uart/

http://www1.rc.unesp.br/igce/demac/alex/disciplinas/MicroII/EMA864315-Serial.pdf

https://www2.pcs.usp.br/~labdig/pdffiles_2012/tx_e_rx_as.pdf

https://ece353.engr.wisc.edu/serial-interfaces/uart-basics/