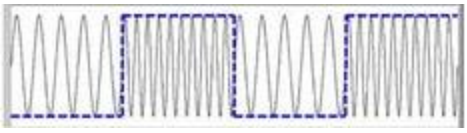


CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto

PROJETO CLIENT-SERVER EM TEMPO REAL

Neste projeto você deverá construir um código em Python para transmissão (client) e recepção (server) serial com uma resposta do server para o client. Você deverá construir seu código modificando o código anterior (loop-back)

Objetivo:

Você deverá ter duas aplicações distintas. Uma aplicação (client) deverá enviar via transmissão serial UART uma sequência de comandos. Cada comando pode ser uma lista com 1, 2, 3 ou 4 bytes. Os comandos serão recebidos pelo server, que deverá "printá-los" na prompt separadamente (1 comando por linha).

O cliente deverá enviar uma quantidade de comandos entre 10 e 30. O server não saberá a quantidade de comandos que irá receber.

Após a recepção, o server deverá retornar ao client uma mensagem informando o número de comandos que foi recebido.

Assim que o client receber a resposta com este número, poderá verificar se todos os comandos foram recebidos corretamente, e o processo termina.

Caso o número de comandos informado pelo server não esteja correto, o cliente deverá expor uma mensagem avisando a inconsistência.

Se o server não retornar nada em até 5 segundos, o cliente deverá expor uma mensagem de "time out"

Importante! Lembre-se que o server não conhece a quantidade de comandos que serão transmitidos! Tão pouco conhece o tamanho de cada comando!

A transmissão deve ser feita com dois Arduinos. Cada aplicação irá se comunicar com um deles.

Comandos existentes:

Comando 1: **00 FA 00 00** (comando de 4 bytes)

Comando 2: **00 00 FA 00** (comando de 4 bytes)

Comando 3: **FA 00 00** (comando de 3 bytes)

Comando 4: **00 FA 00** (comando de 3 bytes)

Comando 5: 00 00 **FA** (comando de 3 bytes)

Comando 6: 00 FA (comando de 2 bytes)

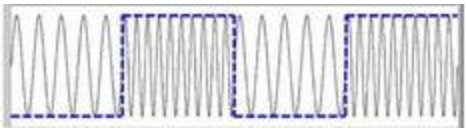
Comando 7: FA 00 (comando de 2 bytes)

Comando 8: **00** (comando de 1 bytes)

Comando 9: **FA** (comando de 1 bytes)

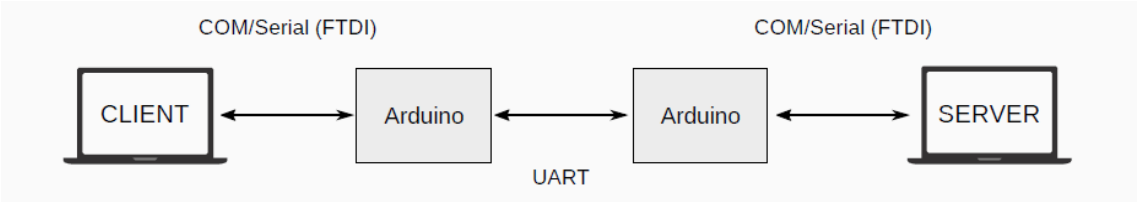
Montagem

Você terá que descobrir como conectar os Arduinos. Precisará de jumpers.



CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto



Gerando a sequência de comandos a serem enviados.

O client deve sortear um número **N** entre 10 e 30, que irá determinar a **quantidades de comandos a serem enviados**. Em seguida determinar quais serão os N comandos que enviará, retirando aleatoriamente N comandos dentre a lista dos **9** possíveis.

Exemplo

Vamos imaginar que o cliente tenha sorteado um N igual a 12. Nesse caso deverá sortear 12 vezes um número inteiro do conjunto {1, 2, 3, 4, 5, 6, 7, 8, 9}, compondo a sequência de comandos. Um possível resultado então seria:

1) Comando 4
2) Comando 8
3) Comando 2
4) Comando 4
5) Comando 7
6) Comando 9
7) Comando 1
8) Comando 9
9) Comando 3
10) Comando 5
11) Comando 4
12) Comando 2

O cliente então deve enviar esses 12 comandos o mais rápido possível. Após enviar a sequência de comandos, o cliente espera receber como resposta do server o número 12, em no máximo 5 segundos. Caso isso não ocorra, deverá exibir a mensagem “time out”.

Entrega

Você e sua dupla deverão apresentar para seu professor o código funcionando em 3 situações.

Você deverá simular um caso de sucesso de transmissão.

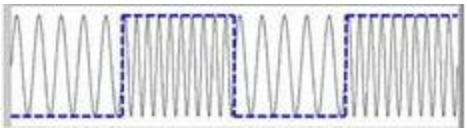
Um caso de erro (server recebeu os comandos com problema de interpretação). Nesse caso você pode forçar um erro com algo “hard coded” no seu código server.

Um caso de ausência de resposta por parte do server e mensagem de time out do cliente.

DATA MÁXIMA PARA Apresentação 02/09/2022

Conselho!

Tente **não alterar** as camadas *enlace, enlaceRx, enlaceTx* e *InterfaceFisica*.



Atenção!

Talvez você se depare com o seguinte erro: *"[ERRO] interfaceFisica, read, decode. buffer :"*

Esse erro ocorre porque, alguns computadores (ou arduínos), no início da primeira transmissão, produzem 2 (ou mais) bytes não pertencentes à mensagem (geralmente \xF0\xF0). Não adianta usar a função flush, pois esses lixos são gerados apenas no início da transmissão. Esses bytes são gerados com mais alguns bits espúrios, e a decodificação de binário para ASCII não pode ser feita, gerando o *except*.

Em alguns casos o erro ocorre no início da transmissão e a aplicação continua sem problemas, mas caso você te nha se deparado com esse problema e ele esteja te impedindo de continuar, aqui vai um *workaround*:

1) No computador que está enviando a mensagem, após declarar a variável enlace, você deve acrescentar o envio de um byte de sacrifício. Esse byte será descartado, servindo apenas para produzir o erro. Depois disso, basta acrescentar um *sleep* de 1 segundo. Copie o código abaixo.

```
com1.enable()
time.sleep(.2)
com1.sendData(b'00')
time.sleep(1)
```

2) Na aplicação de recebimento, você deverá tentar ler esse byte de sacrifício. Esse byte chegará misturado aos bytes de sujeira, corrompido, algumas partes irão gerar dados na variável *buffer*, que você irá limpar. Outras partes irão gerar o erro de decodificação, que poderá ignorar. Dessa maneira, a aplicação que recebe os primeiros dados irá "nascer" esperando 1 byte. Deve ser iniciada antes da outra aplicação. Você pode copiar o código abaixo:

```
com1.enable()
print("esperando  1 byte de sacrifício")
rxBuffer, nRx = com1.getData(1)
com1.rx.clearBuffer()
time.sleep(.1)
```