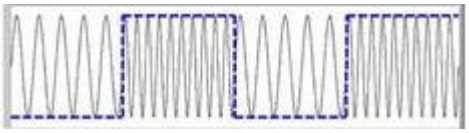


CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto

PROJETO LOOP BACK

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

Entrega:

Esta primeira atividade não é computada para nota. Porém, você e sua dupla (os projetos serão sempre realizados em dupla) deverão mostrar ao seu professor o software em funcionamento durante a aula da próxima quinta (26/08) ou produza um pequeno vídeo (alguns segundos) mostrando o software em funcionamento e envie via BB um link para este vídeo. Não esqueçam de enviar o nome dos dois integrantes da dupla.

Faça testes! Explore as classes! Você irá utilizá-las como base para outros projetos e deverá ter segurança para alterá-las.

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 salva-la como uma cópia. Para isso a recepção do Arduino (*pino rx*) deve estar curto-circuitada com o pino de transmissão (*pino tx*).

Material:

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

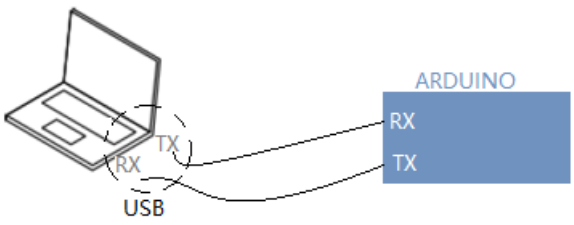
No console python, 'pip install pyserial'

Verificar em gerenciador qual porta com está o Aduino

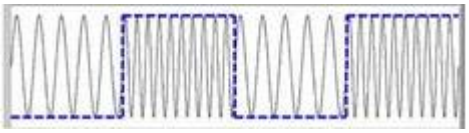
Talvez mudar a porta USB ou tirar e recolocar quando não tem autorização :-\

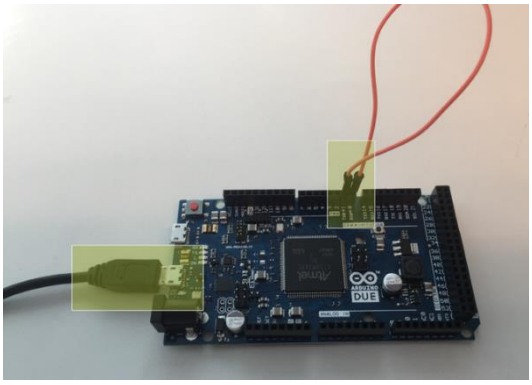
Montagem:

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) 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 PRECISAM FICAR COM O BOTAO DE RESET PRESSIONADO. OU O PINO RESET ATERRADO!

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.

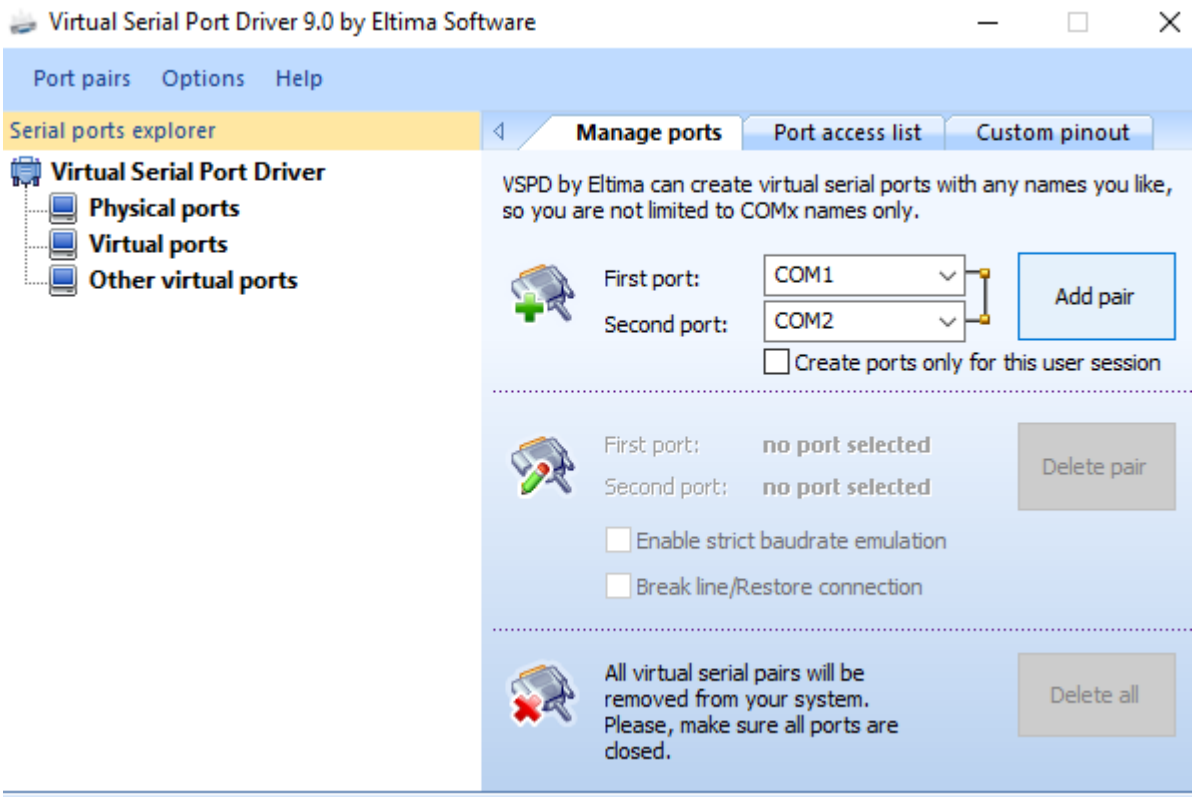
Tente arrumar um modo de medir o tempo decorrido na execução de alguma parte do código. Com isso, tente medir o tempo de transmissão e recepção! Altere o "baudrate" e verifique se o tempo se altera de maneira coerente!

Eu não tenho um Arduino

Utilizando seu Arduino, iríamos estabelecer uma comunicação serial entre seu computador e este dispositivo. Tal comunicação seria feita através das portas de comunicação serial que seu computador cria utilizando um "chip" UART (e a porta física USB para gerar os circuitos de transmissão e recepção). Sem o Arduino, estes canais de comunicação não têm para quem enviar um sinal, e nem de quem receber. Como podemos então nos comunicar serialmente? Que tal este "workaround"? : Existem softwares que emulam dispositivos conectados às portas de comunicação! O que faremos é criar um canal de comunicação virtual. Quando sua aplicação enviar um byte através de uma porta de comunicação estabelecida, o software de emulação receberá esta mensagem, como se houvesse algo realmente conectado ao pino de transmissão. Para sua aplicação, a mensagem foi enviada! O software de emulação faz o papel do Arduino, recebendo um byte. Ele pode também enviar um byte para sua aplicação, como se este byte tivesse chegado no seu pino de recepção.

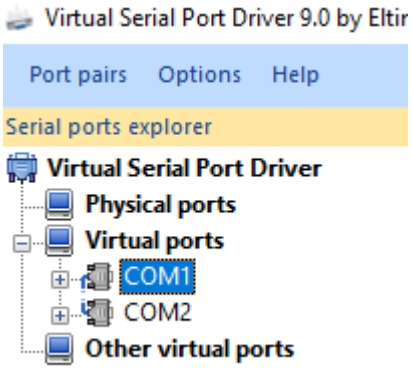
A primeira coisa a fazer é instalar uma versão "trial" desse software. O arquivo está em seu blackboard, na pasta do projeto 1: vspd.exe.

Uma vez instalado, você deverá então criar **um par** de portas de comunicação virtuais. Com o Arduino, apenas uma porta seria suficiente, enviando e recebendo pela mesma porta (TX e RX). Aparentemente o software não tem esta capacidade, então precisaremos de uma porta para enviar e outra para receber (não é um problema).

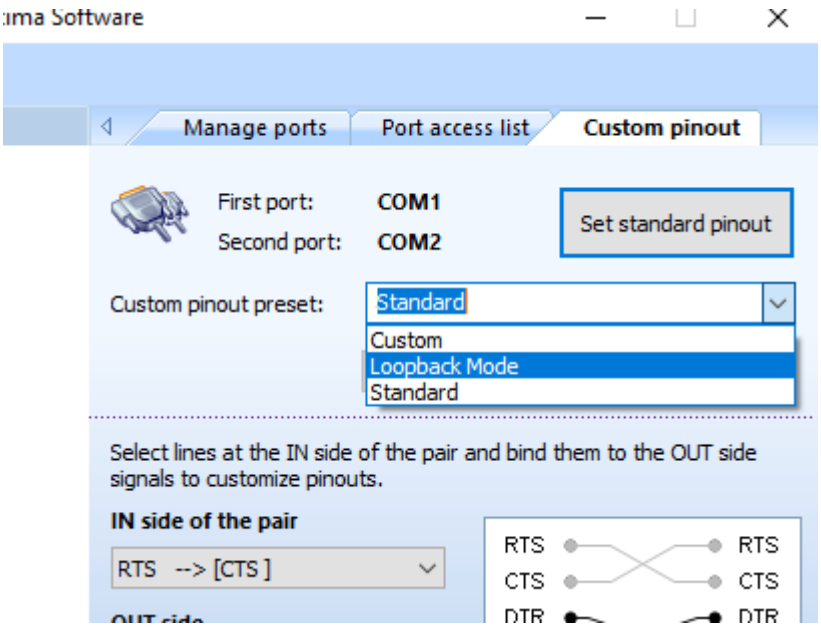


Na figura acima ao clicar em “Add pair” estarei criando dois canais de comunicação serial, como se dois dispositivos tivessem sido conectados ao meu computador. Se já houver portas com esses nomes (COM1 ou COM2) sendo utilizadas em seu computador, o programa deve sugerir outros nomes.

Ao expandir, no menu esquerdo, as portas virtuais, suas portas criadas podem ser vistas:



Agora tudo que tem a fazer é conectar a porta COM2 com a porta COM1 para termos o “LoopBack”. Como faríamos com o Arduino. Assim, ao enviarmos um byte para uma das portas, a outra ira transmitir de volta este byte, como um espelho. Para isso basta que abra o menu “custom pinout” e selecione “loopback mode”.



Pronto! Você pode se comunicar com uma das portas enviando bytes e recebendo de outra. Na sua aplicação python você deverá abrir duas portas:

```
#declaramos um objeto d
#para declarar esse obj
com1 = enlace('COM1')
com2 = enlace('COM2')
```

…enviar por uma das portas

```
com1.sendData(txBuffer)
print('enviou')
```

Receber pela outra

```
    rxBuffer, nRx = com2.getData(txLen)
    print('recebeu dado')


    print (rxBuffer)
```

Não esqueça de fechar as comunicações seriais, caso contrário, ao rodar a aplicação novamente, ela não conseguirá abrir uma nova porta, pois o nome está já sendo utilizado.

```
    # Encerra comunicação
    print("-------------------------")
    print("Comunicação encerrada")
    print("-------------------------")
    com1.disable()
    com2.disable()
except:
    print("ops! :-\\")
    com1.disable()
    com2.disable()
```