Unidade 6: Utilização das bibliotecas TI Hub & TI Rover | **Lição 1: Os sensores integrados no Hub**

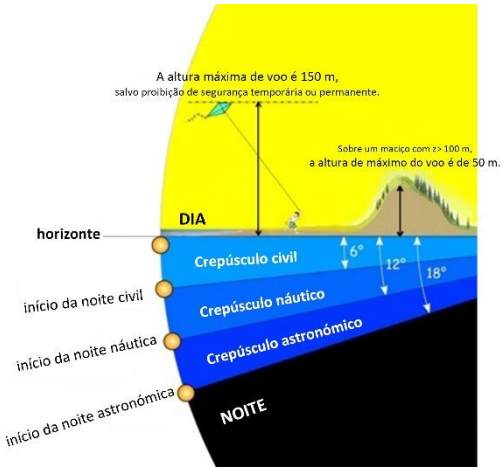
Nesta primeira lição da Unidade 6 vamos aprender como utilizar a biblioteca TI Hub com o objetivo de controlar os dispositivos integrados no TI-Innovator™ Hub.

Objetivos:

- Explorar o módulo **TI Hub**.
- Escrever um programa integrando a biblioteca TI Hub para dispositivos integrados.

Nesta lição, iremos utilizar a biblioteca do **TI Hub** para observar visualmente uma mudança na luminosidade para, subsequentemente, simular uma mudança de crepúsculo ou para registar um conjunto de medições durante o nascer ou o anoitecer.

Começará, também, a considerar como associar esta biblioteca com aquelas que já conhece das Unidades anteriores (**TI PlotLib** e **TI System**) por forma a desenvolver um projeto científico completo.



Considere o algoritmo simples que se apresenta abaixo, o programa que iremos elaborar basear-se-á nele.

ALGORITMO

Medir a intensidade luminosa ambiente:

Lum0 ← medida ± (tolerância?)

Alterar a intensidade luminosa

(# lâmpada; tampa na frente do sensor)

Lum1 ← medida

Se Lum1 > Lum0:

Então ligar LED RGB vermelho (2s)

Senão Se Lum0 < Lum1 :

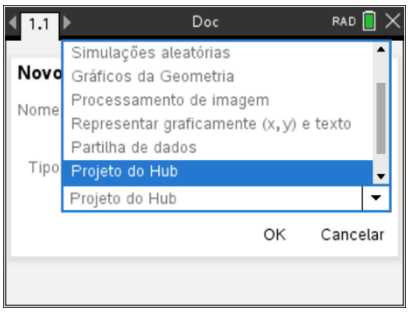
Então ligar LED RGB verde (2s)

Senão não fazer nada

IMPLEMENTAÇÃO DO PROGRAMA:

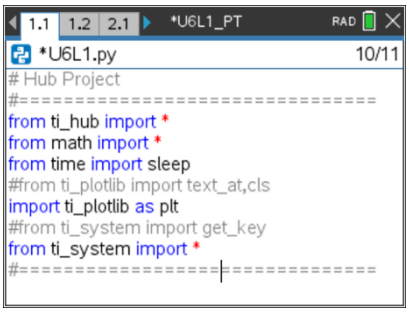
- Inicie um novo programa no editor de TI-Python, designe-o por **U6L1**.
- Este programa deve integrar a biblioteca TI Hub, podendo optar por várias formas de o fazer:

  a) ao adicionar uma página com o editor de TI-Python, especificando o tipo geral de programa. Desta forma irá integrar, pelo menos, a biblioteca correspondente ao nome do tipo.

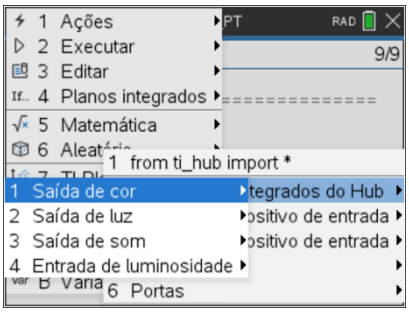


Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

b) Ou, também, pode começar com um programa em branco e, de seguida no início do programa, incorporar manualmente as bibliotecas de que necessita. Estratégia usada nas unidades anteriores.



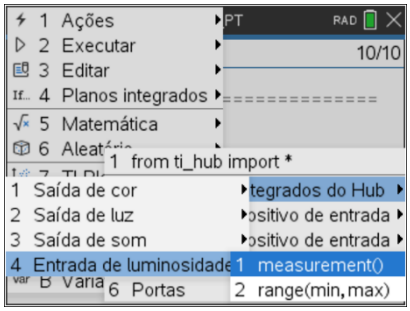
Ao selecionar o tipo de documento como Projeto do Hub, irá obter um ecrã idêntico ao do lado. Ficarão ativas duas bibliotecas TI Hub e Matemática, e três funções de três bibliotecas diferentes. Por vezes é necessário adicionar mais bibliotecas, algo que se pode aperceber ao executar o programa.



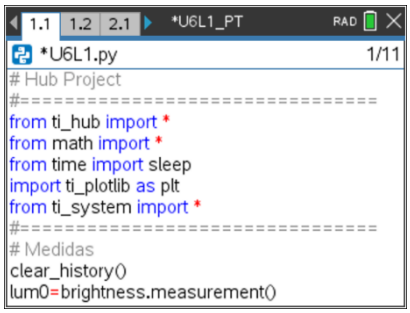
Usaremos o sensor de luz integrado no TI-Innovator, bem como o díodo RGB. Para que o programa seja capaz de os gerenciar, teremos de integrar as bibliotecas correspondentes. Para fazer isso, deve selecionar no menu o módulo **6: TI Hub**, depois opção **1: Dispositivos integrados do Hub**, e por fim **1: Saída de cor**.

OBSERVAÇÃO:
As outras duas opções dizem respeito aos sensores e atuadores que serão conectados diretamente às portas de entrada IN ... e OUT ... do Hub ou possivelmente a outras portas.

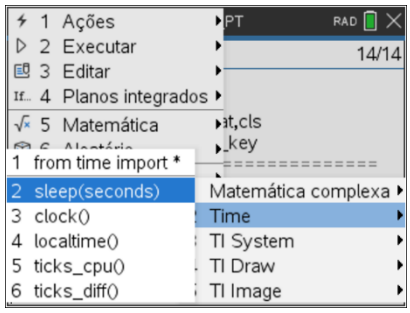
- Para que o programa apresente as informações num ecrã "limpo", limpe-o usando a instrução **clear_history()** acessível através da biblioteca **TI System**. (Tecla menu, seguido de opção **A: Mais módulos**, depois **3: TI System** e por fim **B: clear_history()**)



- Crie uma variável **lum0**, à qual deve atribuir a medida de luminosidade do momento. A instrução **brightns.measurement()** está no módulo **8: TI Hub**, depois opção **2: Dispositivos integrados do Hub**, seguido de **4: Entrada de luminosidade**, e por **1: measurement()**.

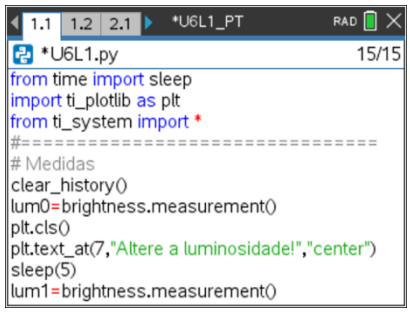


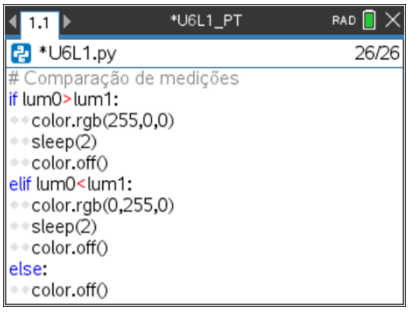
- De seguida insira o código que permita o programa exibir uma mensagem solicitando que o utilizador altere a intensidade da luz nas proximidades do sensor integrado: **plt.text_at()**. Esta função e a função **cls()**, para limpar a janela gráfica, encontram-se no módulo **7: TI PlotLib**.



Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

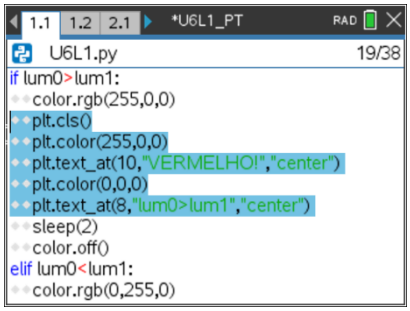
- Adicione um tempo de espera, função **sleep()**, que será útil para efetuar alteração da intensidade da luz. (Tecla menu, seguido de opção **A:Mais módulos**, depois **2:Time** e por fim **2: sleep(seconds)**)
- Crie uma nova variável **lum1** à qual irá atribuir o valor da nova medição da intensidade luminosa.
- Por fim a comparação entre as medidas **lum0** e **lum1**. Utilizando a estrutura de controlo condicional **IF** e em função do resultado lógico da condição **lum0** > **lum1**, o LED RGB ficará verde ou vermelho durante 2 segundos. Caso **lum0** = **lum1**, não ocorrerá nada!
- As funções **color.rgb(Red,Green,Blue)** e **color.off()** encontram-se no módulo **8: TI Hub**, depois opção **2: Dispositivos integrados do Hub**, seguido de **1: Saída de cor**.

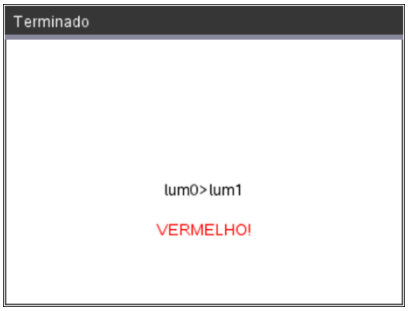


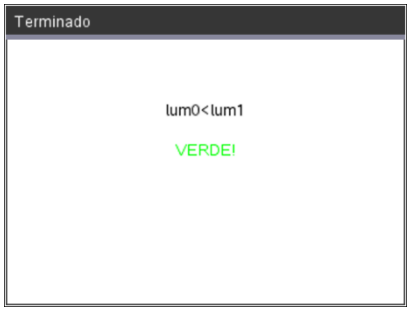


SUGESTÃO:

Pode acrescentar ao seu programa uma instrução que lhe permita observar, no ecrã da unidade portátil, qual a cor que deverá surgir no Hub em função das medidas, para isso bastará acrescentar uma instrução de texto após cada teste. Esta estratégia poderá ser útil para validar programas e verificar se o resultado é o esperado.

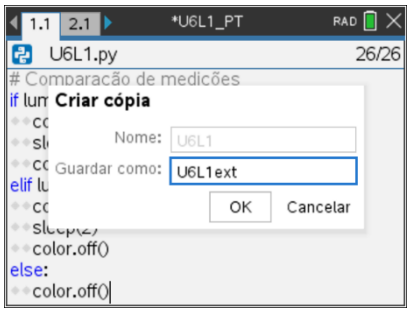






EXTENSÃO DA ATIVIDADE:

Faça uma cópia do programa anterior, designe-o por **U6L1ext** e mova-o para um Problema 2 do seu documento tns. Para mover uma página entre Problemas pode usar a vista de Gestor de Página, clicando ctrl + e usar o menu de contexto, ctrl + menu, sobre a página a mover.



Também poderá, se entender, abri um novo documento e começar de início o programa.

Altere o programa de forma que registe as medidas de luminosidade ao longo de 40 minutos.

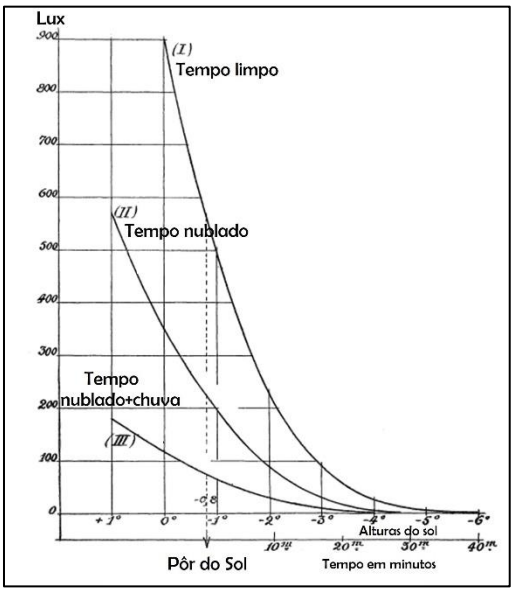
Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

OBSERVAÇÃO:

Note que o sensor de intensidade luminosa do TI-Innovator não é calibrado em Lux, mas isso não é problema, pois estamos interessados apenas nas variações de intensidade e não em sua medida em Lux.

No programa, as medidas da intensidade luminosa serão guardadas numa lista, que designaremos por **r**, que a iniciaremos como vazia **r=[ ]**. O registo dos tempos de recolha dos dados serão também guardados numa lista, **t[ ]**.

Abaixo, no ecrã da esquerda, propõe-se o código para a parte do programa que executará a recolha de dados e os guardará em listas, (#Recolha de dados), e à direita encontra-se um ecrã com o código relativo à representação gráfica dos dados, (#Representação gráfica).



RECOLHA DE DADOS

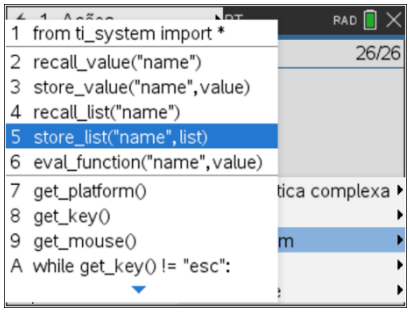
```
1.1 2.1 *U6L1_PT RAD
*U6L1ext.py 18/18
#================================
# Recolha de dados
clear_history()
def bri(n):
  r=[]
  t=[]
  for i in range(n):
    r.append(brightness.measurement())
    t.append(i)
    sleep(60)
  return t,r
```

REPRESENTAÇÃO GRÁFICA

```
1.1 2.1 *U6L1_PT RAD
*U6L1ext.py 29/29
# Representação Gráfica
  plt.cls()
  plt.auto_window(t,r)
  plt.labels("t (min)","r",12,2)
  plt.title("Crepúsculo")
  plt.color(255,0,255)
  plt.scatter(t,r,"+")
  store_list("Tempos",t)
  store_list("Luminosidade",r)
  plt.show_plot()
```

OBSERVAÇÃO:

No ecrã da direita, também se apresenta a proposta de exportação dos dados para listas das restantes aplicações da TI-Nspire CX II, podendo serem exploradas, por exemplo, na aplicação Listas e Folha de Cálculo.



- A função **store.list()** encontra-se no menu **A: Mais módulos**, na opção **3: System** e depois opção **5: store.list("name",list)**.
- A função **bri(n)** realiza aquisição de dados a cada minuto, dada a instrução **sleep(60)**, por n minutos e retorna as listas **t** e **r**.
- De seguida são representados graficamente os dados constantes nas listas **t** e **r** e exportados para as listas Tempos e Luminosidade da TI-Nspire CX II.

Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

Unidade 6: Utilização das bibliotecas TI Hub & TI Rover | **Lição 2: Dispositivos de entrada e saída**

Nesta segunda lição da Unidade 6 vamos aprender como conectar e utilizar um dispositivo, de entrada e saída, integrado no TI-Innovator™ usando a biblioteca TI Hub.

Objetivos:

- Explorar o módulo **TI Hub**.
- Escrever e utilizar um programa para usar um componente de entrada/saída

Nesta lição, iremos utilizar um componente essencial em qualquer cadeia de medição usando sensores: o potenciómetro.

Um potenciómetro é um tipo de resistor variável com três terminais, um dos quais está conectado a um cursor que se move numa trilha resistiva terminada pelos outros dois aos quais a resistência está sujeita.

Os potenciómetros são comumente usados em circuitos eletrónicos. Eles são usados, por exemplo, para controlar o volume de um rádio. Os potenciómetros também podem ser usados como transdutores, pois convertem uma posição em uma tensão. Esse tipo de dispositivo pode ser encontrado em *joysticks*.

Potenciómetro

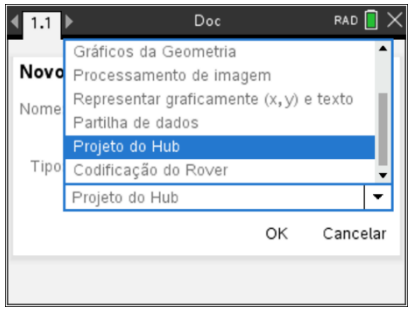
Vamos agora escrever um programa para medir a tensão elétrica entre dois terminais do potenciómetro e, em seguida, apresentar no ecrã.

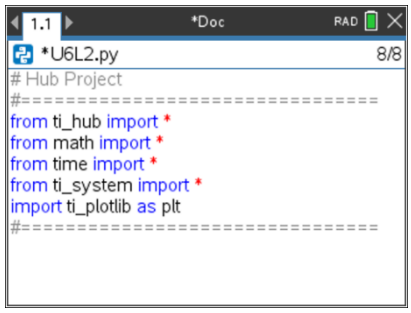
OBSERVAÇÃO:

O objetivo desta lição não é estudar o componente em si, mas sim integrá-lo num programa Python para se obter os dados que ele fornece. Portanto, o programa que irá ser criado será facilmente transponível para qualquer outro tipo de transdutor.

IMPLEMENTAÇÃO:

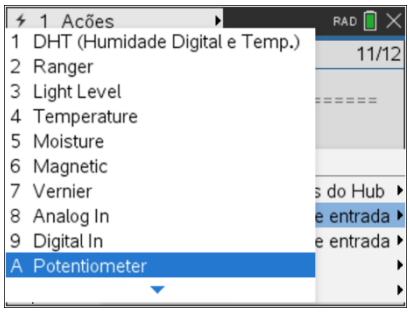
- Inicie um novo programa no editor de TI-Python, designe-o por **U6L2**.
- Selecione tipo de programa **Projeto do Hub**, desta forma automaticamente as bibliotecas **TI Math** e **TI Hub** serão importadas.
- Agora, necessitará de ainda de importar as bibliotecas **TI System** e **Time**, acessíveis na opção **A: Mais módulos** do menu, e eventualmente necessitará também da biblioteca **TI Plot**.
- Crie uma função **pot()** sem qualquer argumento.
- Apague o ecrã usando a instrução **clear_history()** localizada na biblioteca **TI System**.
- Na biblioteca **TI Hub**, selecione o menu **3: Adicionar uma unidade de entrada** e de seguida **A: Potenciómetro**.

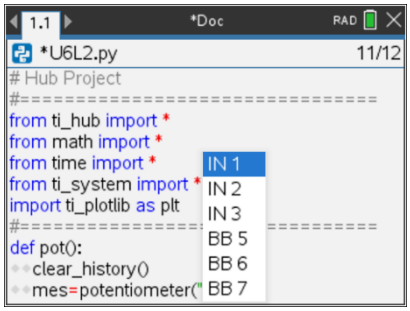




Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

- Atribua o valor a ser registado pela função **potentiometer("porta")** à variável **mes**.
- Conclua esta instrução definindo como argumento da função **potentiometer("porta")** a porta **In1** (entrada 1).
- Utilizando a tecla [tab] pode preencher os campos da função de forma automática e mais rápida.



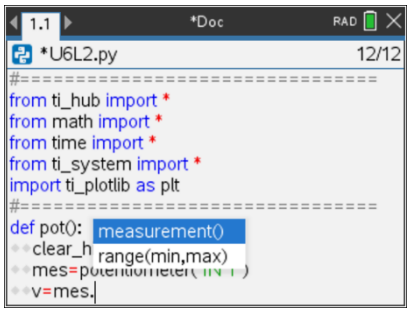


OBSERVAÇÃO:

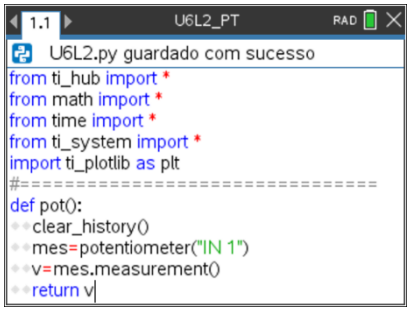
Todos os sensores (dispositivos de entrada) têm um mínimo de duas variáveis:

1: var=sensor("porta") e 2: var.measurement() .

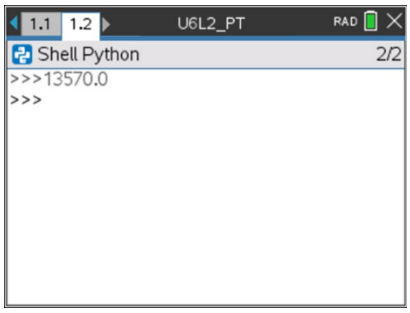
- Crie uma variável **v** que permita recolher a medição do sensor conectado (variável **mes**). No editor de programas, surgirá a proposta de preenchimento automático assim que se escreva **v=mês.** , abrindo uma janela com as opções. A opção **measurement()** não está disponível no menu **TI Hub**.



- Verifique o funcionamento do seu sensor após ter colocado previamente o potenciómetro na posição central.
- Conecte o **TI-Innovator™** à calculadora e, em seguida, o potenciómetro à porta **IN1** do Hub.
- Execute o programa, e no interpretador execute a função **pot()**.



Deverá obter dados da mesma ordem de magnitude do ecrã ao lado, mas observe que não é uma tensão, pois seu potenciómetro é alimentado por uma tensão de 3,3V. O valor a ser encontrado pertencerá ao intervalo [0; 3,3].

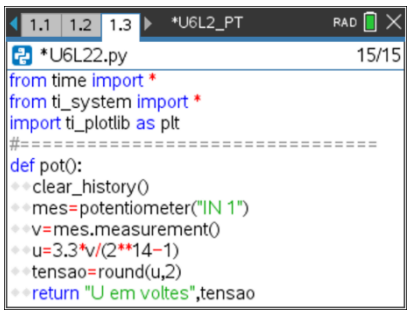


- Efetue uma cópia do programa através do menu, tecla [menu], e da opção **1:Ações**, seguido da opção **3:Criar uma cópia**. Designe a cópia do programa por **U6L22**.

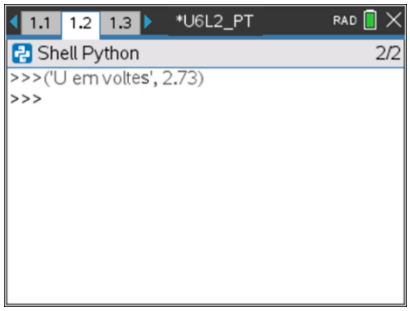
Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

- Altere o programa para levar em consideração a resolução do conversor analógico para digital (14 bits). Assim, a medição da tensão será:

$$u = U_{alim} \times \frac{v}{2^{14}-1}\,.$$



- O valor final da tensão será arredondado às centésimas.



- Execute o programa **U6L22**, atalho ctrl + R, e no interpretador execute a função **pot()**.

MAIS ALÉM:

Algumas ideias para possíveis extensões da lição:

- Use um potenciómetro para construir um sensor angular (uma medição de tensão corresponde ao valor de um ângulo lido em um transferidor), a seguir faça a representação gráfica da função modelada α = f (u).
- Entre 0 e 3,3 V, associe uma faixa de tensão com uma cor usando o LED RGB TI-Innovator™.
- Associe a medição da tensão com as coordenadas de um ponto marcado (princípio de um *joystick*).
- Etc ...

Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

Unidade 6: Utilização das bibliotecas TI Hub & TI Rover | **Lição 3: Dispositivos de entrada e saída**

Nesta terceira lição da Unidade 6 vamos aprender como conectar o TI-Rover usando a biblioteca **TI Rover**.

Objetivos:

- Explorar o módulo **TI Rover**.
- Escrever e utilizar um programa para usar o TI-Innovator™ Rover e os periféricos associados.
- Usar um ciclo aberto e uma função condicional.

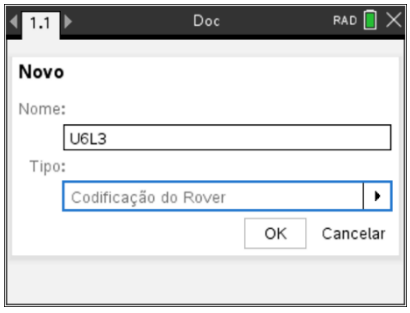
Nesta lição, iremos criar um programa que permita ao TI-Innovator™ Rover ter a possibilidade de realizar um trajeto marcado pela iluminação do díodo RGB, desde que a distância (medida pelo sensor RANGER) respeite um limite definido numa função condicional.

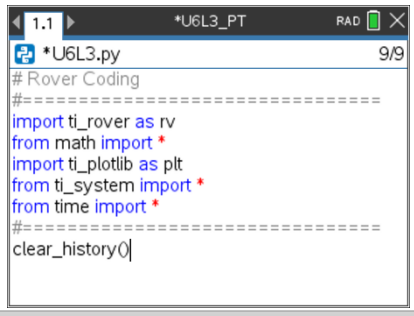
O ALGORITMO

Avance uma distância de 2 metros
  Enquanto o movimento não for interrompido pelo utilizador
    a ← distância medida relativamente a um objeto
    Se a < 2
      Então exiba uma cor vermelha e pare
      Senão exiba uma cor verde e continue
  Pare, exiba uma cor azul
  Espere 1 segundo
  Desligue o díodo
Ligue o díodo em azul para assinalar o fim

IMPLEMENTAÇÃO DO PROGRAMA:

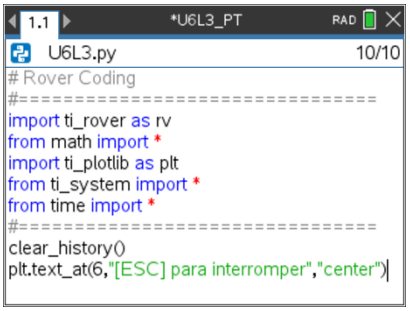
- Inicie um novo programa no editor de TI-Python, designe-o por **U6L3**.
- Selecione tipo de programa **Codificação do Rover**, desta forma automaticamente as bibliotecas bibliotecas **TI Math**, **TI System** e **Time**, serão importadas. Também o módulo **TI PlotLib** ficará implementado.
- Agora, poderá começar a escrever o programa.
- Apague o ecrã usando a instrução **clear_history()** localizada na biblioteca **TI System**.



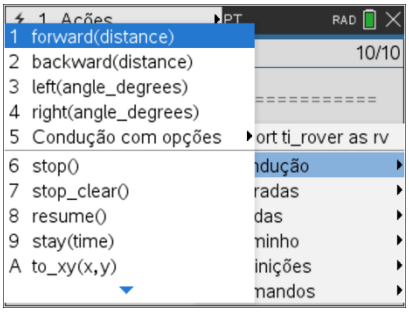


Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

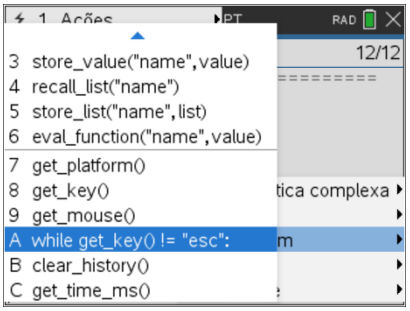
- Para o utilizador interromper o movimento do Rover vamos usar como estratégia colocar o núcleo do programa dentro de um ciclo de controlo **While**, interrompido quando o utilizador clicar na tecla esc. Comecemos por apresentar essa informação ao utilizador usando a função **plt.text_at(linha, "texto", alinhamento)** do módulo **TI PlotLib**.



- Dê instruções ao **TI-Innovator™ Rover** para avançar. A unidade de medida da distância será uma sua opção, sabendo que, por defeito, está definida como 0,1 m. Assim, **rv.forward (20)** fornecerá ao Rover informação para ele se mover para frente uma distância de 2 m. A instrução **rv.forward()** está localizada no menu **9: TI Rover** e, de seguida, em **2: Condução**, e por fim **1: forward(distance)**.



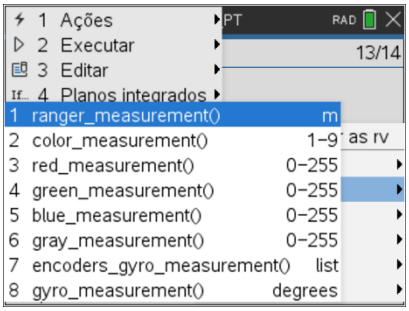
- Em seguida, insira o início de um ciclo **While** já predefindo para a nossa estrategia de interrupção do programa, que pode ser obtido no menu da biblioteca **TI System** (opção **A: Mais módulos** do menu).



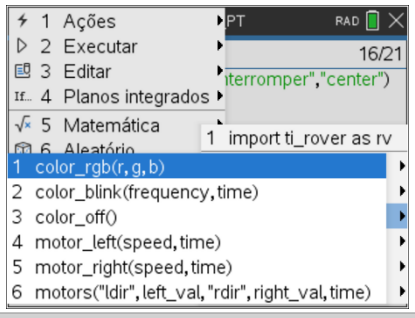
NOTA:

Algumas das instruções disponíveis na biblioteca **TI System**, também estão disponíveis na bibloteca **TI Rover** na opção **7: Comandos**.

- Crie uma variável **a** à qual será atribuida a distância medida pelo RANGER. Para isso, comece por escrever a letra **a=**, depois insira a instrução **rv.ranger_measurement()** localizada no menu **9: TI Rover** e **3: Entradas**, e finalmente **1 rv.ranger_measurement()**. A unidade de medida é o metro.

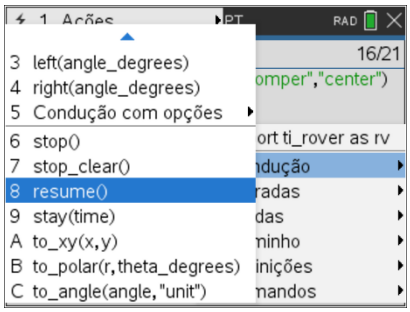


- Agora, implemente a estrutura condicional do nosso algoritmo. Se a distância medida for inferior a 20 cm, o **ROVER** irá parar e o LED RGB acenderá em vermelho. A instrução **rv.color_rgb()** está disponível na biblioteca **9:TI Rover,** na sua opção **4: Saídas**, e por fim **1: color_rgb(r,g,b)**.

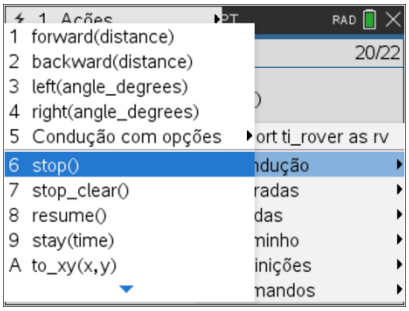


Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

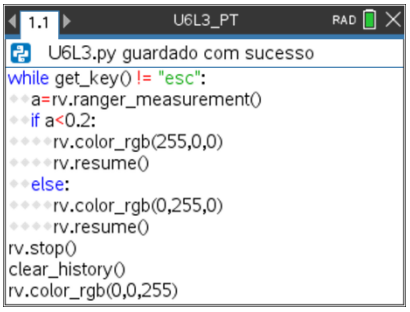
- A instrução **rv.resume()** termina o processamento de ações na fila, e desta forma não teremos sobreposições de instruções sobre o ROVER. Esta função está localizada no menu **9: TI Rover** e, de seguida, em **2: Condução**, e por fim **8: resume()**.



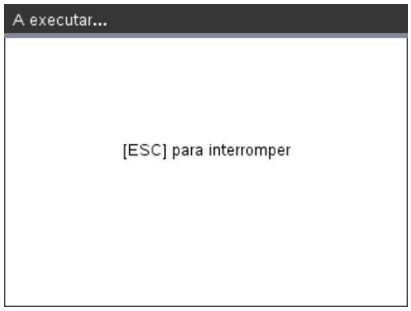
- Iremos utilizar a função **rv.stop()** para parar o **ROVER**, sendo uma instrução de condução está, portanto, localizada no menu correspondente do módulo **TI Rover**. Caso contrário, o díodo RGB fica verde e o **ROVER** continua o seu percurso até encontrar a distância definida.



- Por fim, após o ciclo, inserir as seguintes instruções:
  - Parar o ROVER - **rv.stop()** ;
  - Apagar o ecrã - **clear_history()** ;
  - Exibir a cor azul no díodo - **rv.color_rgb(0,0,255)** .



- O programa está terminado. Conecte a unidade portátil ao Hub e este ao Rover, ligue o Rover. Coloque o kit TI-Nspire CX II + TI-Innovator Hub + Rover numa superfície plana segura.



- Execute o programa, atalho ctrl + R, e observe o Rover.

Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

Unidade 6: Utilização das bibliotecas TI Hub & TI Rover — **Aplicação: Representação de um percurso**

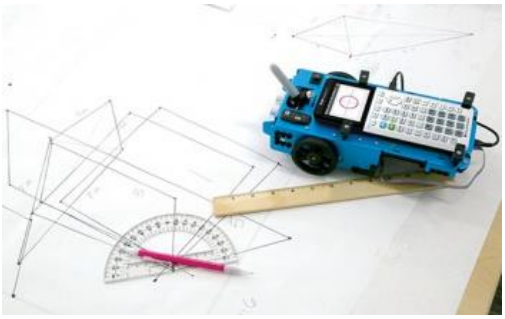
Nesta aplicação da Unidade 6 irá ligar o **TI-Innovator™ Rover** utilizando a biblioteca **TI Hub** e construir um programa para registar as coordenadas dos pontos durante um percurso e, de seguida, representá-los graficamente.

Objetivos:

- Explorar o módulo **TI Rover**.
- Escrever e utilizar um programa para usar o **TI-Innovator™ Rover** e os periféricos associados.
- Utilizar um ciclo fechado.
- Representar graficamente um conjunto de dados.

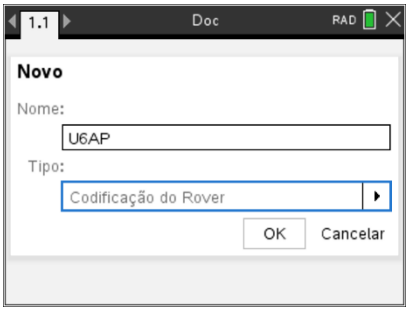
Nesta lição, iremos criar um programa que dará ao TI-Innovator™ Rover a capacidade de realizar um caminho correspondente ao desenho de um polígono regular.

As coordenadas dos vértices do polígono regular serão guardadas em listas e posteriormente representadas graficamente usando as instruções da biblioteca **TI Plot**.



IMPLEMENTAÇÃO DO PROGRAMA:

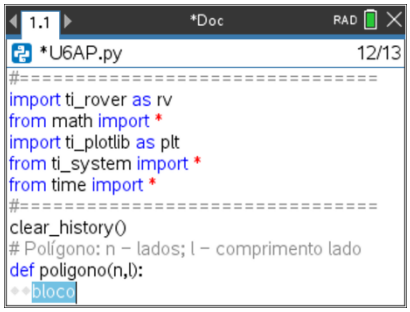
- Inicie um novo programa no editor de TI-Python, designe-o por **U6AP**.



- Selecione tipo de programa **Codificação do Rover**, desta forma automaticamente as bibliotecas bibliotecas **TI Math**, **TI System** e **Time**, serão importadas. Também o módulo **TI PlotLib** será importado para o programa agora iniciado.

- Agora, poderá começar a escrever o programa.

- Embora possa não ser necessário, as instruções para limpar o ecrã e não mostrar o cursor são sempre mais agradáveis para o utilizador. Assim, colocaremos, sempre que nos parecer conveniente a instrução **clear_history()** localizada na biblioteca **TI System**.



- Crie uma função **poligono()**, tomando como argumentos **n** o número de lados do polígono e **l** o comprimento, na unidade por defeito do TI-Innovator™ Rover, ou seja, o dm.

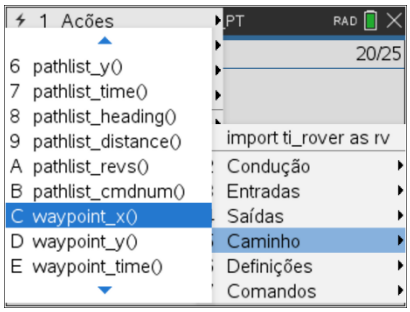
- A medida da amplitude do ângulo externo de cada polígono será, portanto, igual **a** = $\frac{360}{n}$. Porquê o ângulo externo?

- Crie duas listas vazias, **abcissas** e **ordenadas**, destinadas a guardar as coordenadas de cada um dos **n** vértices.

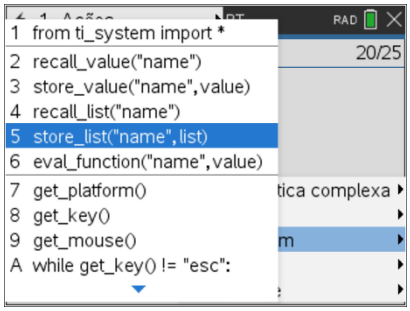
Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

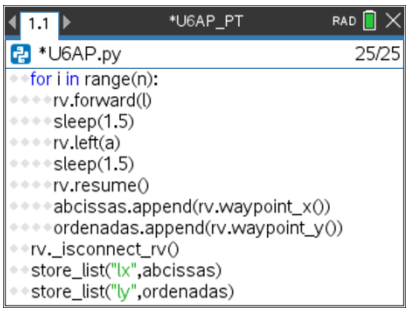
- Crie um ciclo aberto, **While**, de tamanho **n**.

- As instruções que se seguem, e que permitirão que o **Rover** percorra o percurso desenhado pelo polígono considerado, podem ser obtidas na biblioteca **TI Rover**. Estas instruções repetir-se-ão **n** vezes, tamanho do ciclo **While**.



  - Mover para a frente o **Rover** em **l** decímetros – **rv.foward(l)**.
  - Virar à esquerda com um ângulo de amplitude **a** – **rv.left(a)**.
  - Colocar, entre cada etapa, uma espera de 1,5s – **sleep(1.5)**.
  - Obter e guardar as coordenadas dos vértices nas listas **abcissas** e **ordenadas**. – **abcissas.append(rv.waypoint_x())** e **ordenadas.append(rv.waypoint_x())** .
  - Desconectar o **Rover** no final do percurso – **rv._isconnect_rv()**.
  - Exportar os dados das listas **abcissas** e **ordenadas**, respetivamente, para as listas **lx** e **ly**, possibilitando a representação gráfica e/ou exploração dos dados numa outra aplicação da TI-Nspire™ CX II (Gráficos, Listas e Folha de Cálculo, …).





OBSERVAÇÕES:

  - As instruções **rv.waypoint_x()** e **rv.waypoint_y()** encontram-se no módulo **TI Rover**, na opção **5: Caminho**, e de seguida a opção **C: waypoint_x()** .
  - A instrução **rv._isconnect_rv()** deve ser escrita a partir do teclado.
  - A instrução **store_list()** encontra-se no módulo **TI System**, na opção **5: store_list("name",list)**.

- Terminada a recolha de dados e o deslocamento do Rover, passemos para a representação gráfica dos dados. Podemos incluir este código na função já criada, ou, se for vantajoso, criar no mesmo programa uma nova função para a representação gráfica, como se procedeu em alguns exemplos das lições anteriores (Unidade 5).

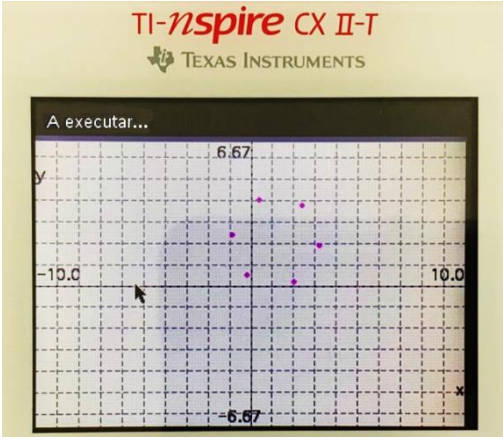
- Todas as instruções usadas para a representação gráfica dos dados estão disponíveis no módulo **TI PlotLib**, que obrigatoriamente terá a sua biblioteca importada, nas opções **2: Configurar** e **3: Desenhar**.

Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/

EXECUÇÃO DO PROGRAMA:

- Execute o seu programa, atalho ctrl + R, e, no interpretador, execute a função **poligono()** para, por exemplo, percorrer e desenhar um quadrado com o lado de comprimento 1 dm: **poligono(4,1)**.

- Continue a testar o programa, agora com um hexágono regular com 2 dm de lado, **poligono(6,2)**, obtendo a seguinte representação.



OBSERVAÇÃO:

Para este tipo de exercício, evite utilizar as instruções **rv.pathlist_x()** e **rv.pathlist_y()**.

Pois, ao desenhar um segmento, serão registadas as coordenadas dos pontos de um segmento do polígono e, em seguida, no segmento seguinte serão registados novamente as coordenadas do último ponto do segmento anterior como o primeiro ponto do atual segmento.

Mais ainda, porque no nosso código colocamos um atraso de 1.5 s entre cada movimento do Rover. Portanto, as instruções **rv.path…** são, para o nosso caso, inapropriadas.

Usando as instruções **rv.pathlist_x()** e **rv.pathlist_y()**, obteríamos as coordenadas do ponto final duas vezes.

NOTE QUE:

Deve ajustar o formato de sua grelha de fundo, dependendo dos polígonos que deseja desenhar. Isso foi intencionalmente definido aqui nas configurações por defeito, a fim de observar a precisão da pista do TI-Innovator™ Rover.

Também deve ter em atenção a natureza da superfície na qual o Rover se irá deslocar. O piso não deve oferecer muita resistência ao movimento ou, pelo contrário, favorecer o deslizamento.

Disponibilizado sob a Licença Pública Creative Commons: Atribuição – Não Comercial – Compartilha Igual 3.0 Portugal.
https://creativecommons.org/licenses/by-nc-sa/3.0/pt/