

Atividades

Laboratório de Educação Digital



Guião 4 – Descobrimos... Pack Módulo I/O

Explorar, Aprender e Inspirar: O Poder dos LED e da TI-Nspire™ CX II-T

- Duração: 100 min;
- Dificuldade: Intermédio;
- Tema(s): STEM; Pensamento Computacional; Módulos de entrada e saída; Automatismos.

1. Objetivos

- Desenvolver competências inerentes ao trabalho STEM;
- Desenvolver programas em Python mais complexos, com estruturas de seleção, entradas e saídas;
- Desenvolver um mecanismo de portão automático, que abre durante o dia e fecha durante a noite.

2. Enquadramento

Neste projeto, vamos desenvolver um mecanismo para um portão automático, que abre durante o dia e fecha durante a noite. Este mecanismo poderia ser instalado, por exemplo, numa quinta, para impedir que os animais frequentem um determinado espaço durante a noite, ou que se ausentem inadvertidamente do seu espaço de recolhimento.

Utilizando programação em Python, a interface TI-Innovator™ Hub e o Pack Módulo I/O, vamos criar um sistema onde se começa por medir a luminosidade que chega a um sensor de luz. Se a luminosidade for grande (dia), o portão é aberto pela rotação de um servomotor e é acionada uma luz de aviso que pisca durante o acionamento do motor. Caso contrário, estaremos na noite, e o portão deve fechar, invertendo o funcionamento do mecanismo de abertura.

3. Material necessário

- Calculadora TI-Nspire™ CX II-T
- TI-Innovator™ Hub¹
- Pack Módulo I/O (Sensor de Luz; Servomotor; LED Socket Kit)
- *Powerbank*

¹ Para instruções detalhadas de configuração e atualização iniciais do equipamento, consultar <https://education.ti.com/pt/produtos/micro-controller/ti-innovator-rover/getting-started>



Este recurso pedagógico é disponibilizado com a respetiva licença Creative Commons

4. Para começar...

Começa por te certificar que a calculadora está devidamente conectada ao Hub, utilizando o cabo mini USB-A / mini USB-B, tal como mostrado na figura 1:

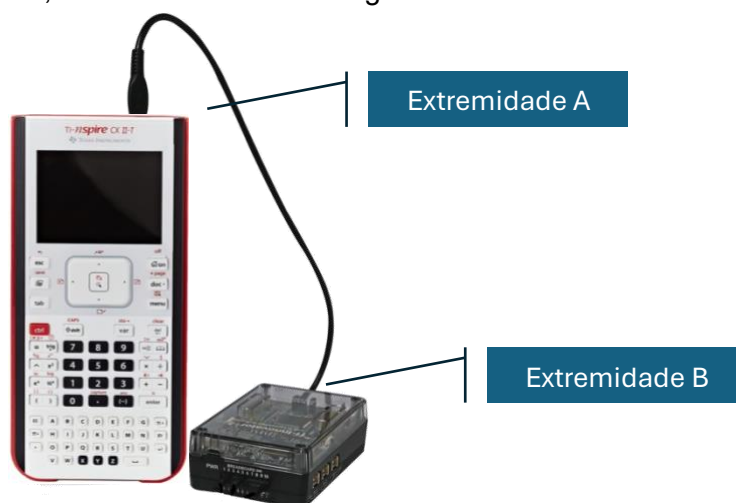


Figura 1 - Conexão do TI Innovator™ Hub à calculadora

Este projeto vai utilizar a TI-Nspire™ CX-II T e o TI-Innovator™ Hub, conectado a um sensor de luz, um LED e um servomotor. Iremos ligar o sensor de luz a uma entrada (IN 1) e o LED e o servomotor a duas saídas, respetivamente OUT 1 e OUT 3.

Cada dispositivo que será ligado a uma entrada ou saída do Hub será ligado por cabos idênticos, que acompanham o conjunto. A *powerbank* liga-se à porta identificada por PWR, num dos topos do HUB. Certifica-te que fazes corresponder cada elemento à entrada ou saída correta, conforme se mostra na figura 2.

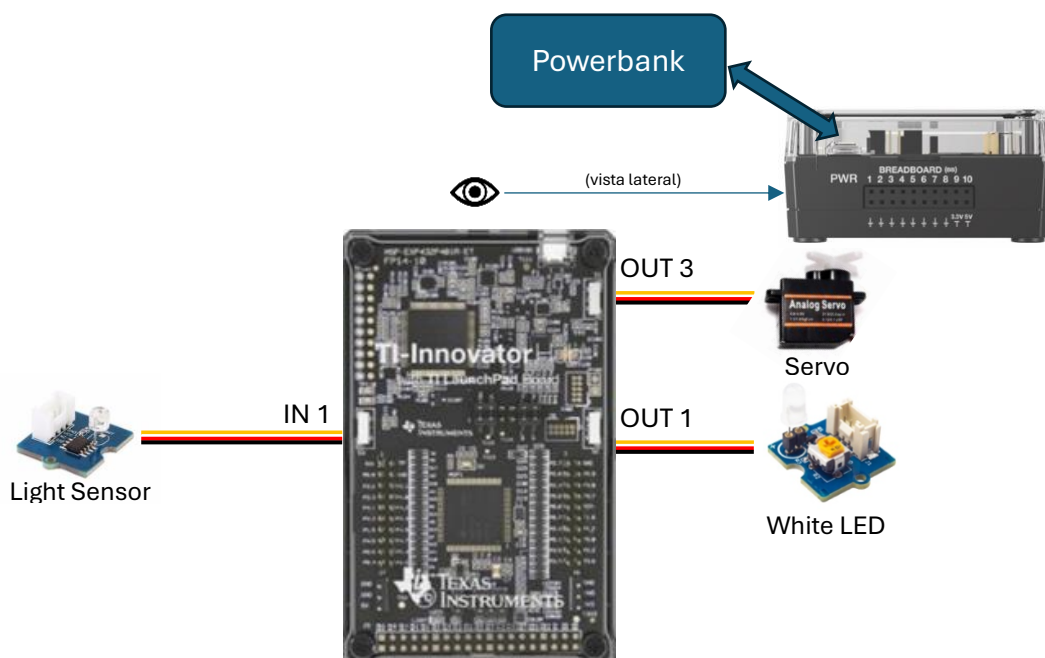


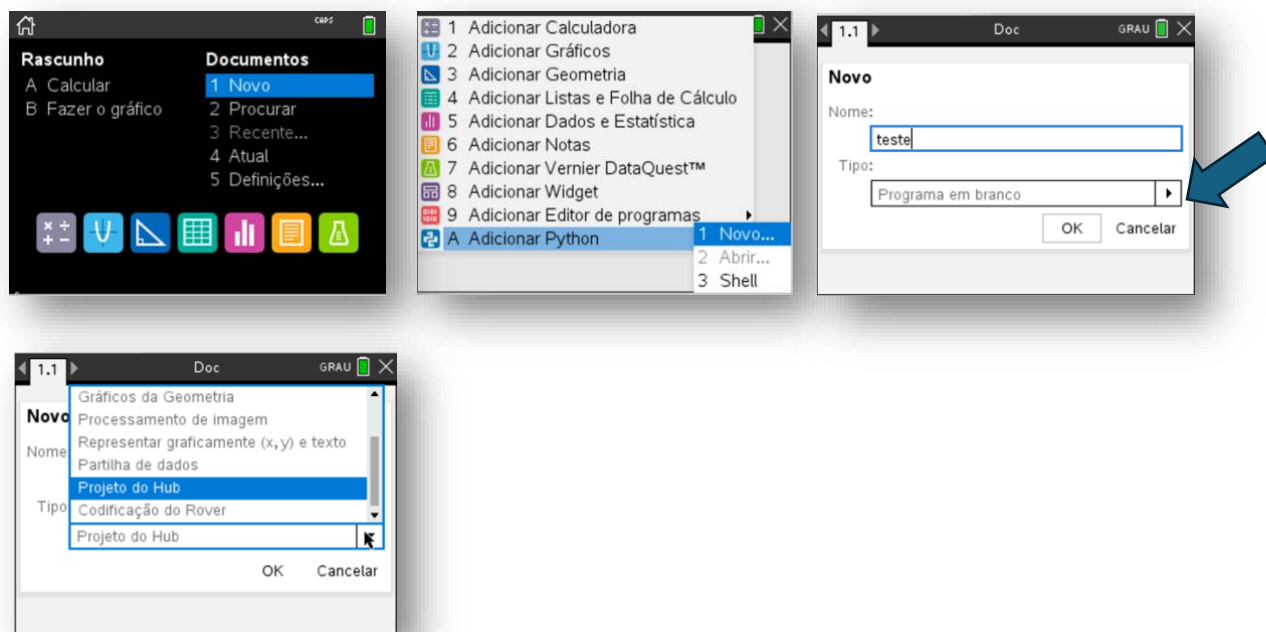
Figura 2 - Conexão de entradas e saídas



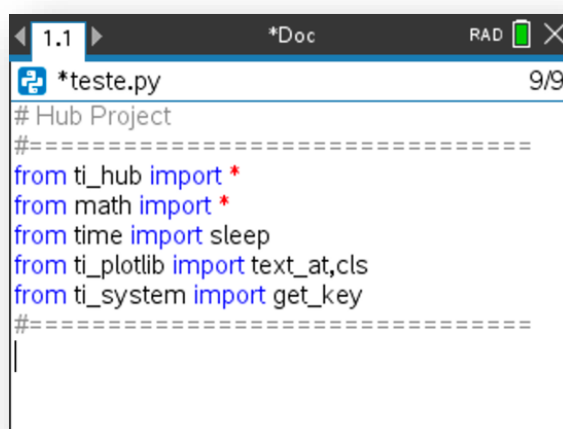
Este recurso pedagógico é disponibilizado com a respetiva licença Creative Commons

O LED deve ser inserido no LED *socket*, tendo o cuidado de te certificares que o pino mais longo do LED (+) seja inserido no polo correspondente, assinalado no encaixe.

Vamos agora criar uma página em Python, pré-configurada para projetos com o Hub. A partir de , segue os passos descritos de seguida:



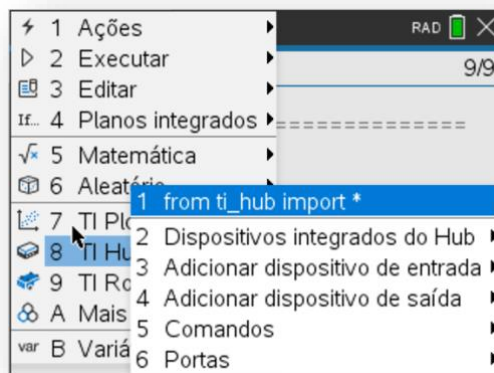
Deverás ter obtido um ecrã semelhante ao da figura:



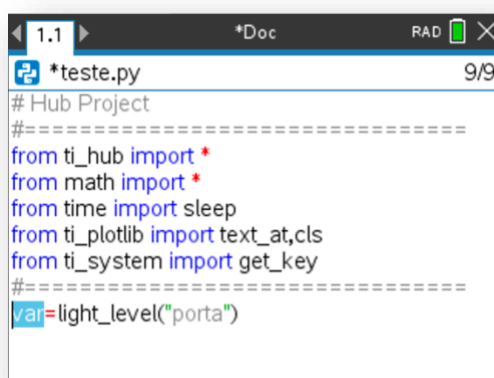
Antes de utilizar cada entrada ou saída, é necessária a sua configuração, associando-a à porta correspondente.



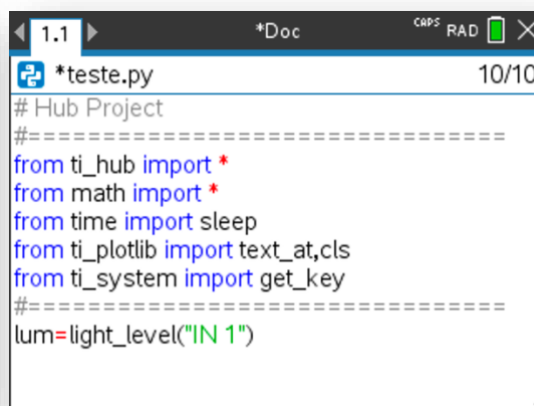
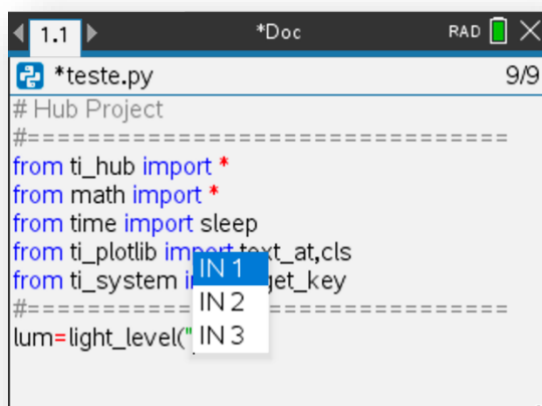
Todos os comandos necessários à construção de programas com o Hub podem ser facilmente acedidos a partir de  | 8 (TI Hub) :



Para configurar o sensor de luz (uma entrada), seleccionamos 3 (Adicionar dispositivo de entrada | 3 (Light level)). Deverás obter um ecrã semelhante ao seguinte:

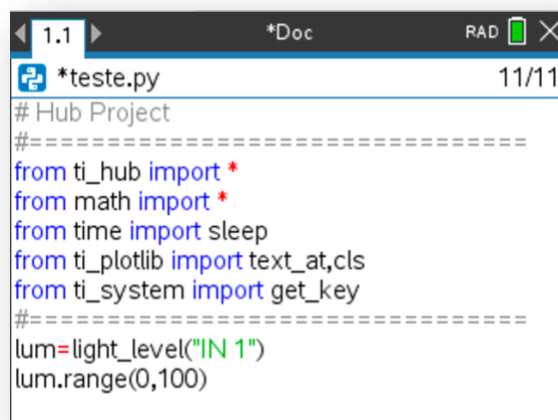


Para que o Python identifique o *objeto* sensor de luz, temos de lhe atribuir um nome e identificar a porta do Hub usada para o conectar. Assim, o texto “var” assinalado a azul deve ser substituído pela designação pretendida, por exemplo, *lum* e, ao deslocar o cursor até onde aparece a designação “porta”, devemos seleccionar IN 1. Podes ver como fazer nas imagens seguintes:



Agora, sempre que for digitado o nome do *objeto lum* seguido de `.` aparece um menu de opções com todas as funcionalidades disponíveis (para o caso deste sensor, *measurement()* ou *range(min,max)*). Para que seja mais fácil prever a diferença de luminosidade entre o dia e a noite (para ativarmos a abertura ou fecho automático do portão) é preferível redefinir uma escala de leitura do sensor, por exemplo, entre 0 e 100. Escolhemos então a opção *range(min,max)* – após digitar *lum* `.` - e preenchemos com os valores que pretendemos.

Deverás ter obtido um ecrã semelhante a este:



```
1.1 *Doc RAD X
*teste.py 11/11
# Hub Project
#=====
from ti_hub import *
from math import *
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
lum=light_level("IN 1")
lum.range(0,100)
```

Agora, o comando *lum.measurement()* passará a devolver um valor mais perto de 0 quando se aproxima a noite (um valor inferior a 30 significa que já estará a escurecer, por exemplo) e mais perto de 100 em plena luz do dia.

5. Desafios

1. Repete os procedimentos anteriores para atribuir as saídas que iremos utilizar: o servomotor e o LED. Não te esqueças de te certificar que há correspondência entre a ligação física dos equipamentos e a informação que é inserida no programa em Python.
2. Vamos agora efetuar um programa de testes, apenas para verificar se todo o equipamento está bem configurado e operacional. Vamos mostrar no ecrã o valor da luminosidade detetada pelo sensor de luz, fazer piscar o LED e rodar o motor 90°. Para mostrar no ecrã a luminosidade, utiliza o comando `print()`, colocando dentro dos parêntesis a opção de medição do sensor de luz (ativada por *lum* `.`, como explicado anteriormente). Tendo o cuidado de inserir um comando em cada linha, adiciona também o comando de piscar o LED e de rodar o motor (tenta descobrir como, digitando o nome que escolheste para designar cada objeto no desafio 1 e premindo `.`). Não te esqueças de ligar a powerbank .
Testa o programa várias vezes (`ctrl` `R`), alterando os ângulos de rotação do motor (valores positivos e negativos) e expondo o sensor de luz a diferentes luminosidades,



para verificar que a leitura está coerente (mais baixa para locais escuros e mais alta para locais iluminados).

3. Vamos agora criar um primeiro automatismo, fazendo com que o “portão” abra se estiver de dia e feche se estiver de noite. Vamos utilizar uma estrutura de seleção *If... Else...* para que, **se** (*If...*) estiver escuro, o motor (do “portão”) rode 90° no sentido positivo e a luz pisque lentamente por 2 segundos e, **caso contrário** (*Else...*) o motor rode 90° no sentido negativo e a luz pisque rapidamente por 2 segundos. Para lançar no programa uma estrutura *If...Else...* acedemos a  4 (Planos integrados) | 2 (Contríolo) | 2 (if..else..) Deverás ter obtido um ecrã semelhante a este:



```
1.1 *Doc RAD X
*teste.py 15/28
#=====
lum=light_level("IN 1")
lum.range(0,100)
luz=led("OUT 1")
motor=servo("OUT 3")
print(lum.measurement())

if ExprBooleana:
  ++bloco
else:
  ++bloco
```

Devemos agora definir qual a condição que seleciona se o “portão” abre ou fecha. Essa condição será baseada na luminosidade. Vamos considerar que a **luminosidade ser inferior a 20** (na escala que definimos anteriormente) é o “gatilho” que determina se o portão abre ou fecha. Esta condição deve ser inserida sobre texto “ExpBooleana” assinalado a azul (para inserir <. clicar  = ). No primeiro “bloco” colocamos as instruções referentes ao fecho do portão, e no segundo “bloco” as referentes à abertura. Nota que todas as instruções de cada “bloco” devem respeitar o avanço de 2 espaços (indentação).

Após colocares todas as instruções nos locais corretos, testa o programa várias vezes ( R), alterando, antes de cada execução, a luminosidade que atinge o sensor de luz. Acabaste de desenvolver um sistema automático de abertura e fecho de um portão! Podes concluir o projeto construindo um portão, de tipo “cancela”, para acoplares ao servomotor, tornando o projeto ainda mais realista.

6. Conclusão

Com esta atividade, construístes um primeiro automatismo, utilizando uma estrutura que, em programação, se chama de *estrutura de seleção* e que permite realizar programas muito versáteis e que, de certa forma, conferem uma espécie de “inteligência” às máquinas que nos rodeiam. Da mesma forma, ao aumentares as capacidades do Hub por adicionares objetos de entrada e de saída, percebeste um pouco melhor como se constroem máquinas mais complexas, que cada vez mais nos acompanham para todo o lado.

No entanto, este teu primeiro automatismo só funciona uma vez, não sendo verdadeiramente útil numa situação real, pois obrigaria à sua execução manual todos os dias.

Utilizando os conhecimentos que adquiriste na exploração dos Guiões LED TI anteriores, investiga como poderias modificar o teu programa para que a execução do programa fosse contínua, e o “portão” fosse aberto e fechado todos os dias.

Sugestão: podes criar uma variável apenas para o “portão” *saber* se já está aberto ou fechado (assumindo o valor 1 ou 0), de cada vez que o programa avalia, pela informação do sensor, se está dia ou noite. Assim, o “portão” só irá abrir **SE** estiver de dia **E** estiver na posição fechado. Tem em consideração que, quando usado num “comparador” em Python, a igualdade se representa por “==” e não por “=”.



Este recurso pedagógico é disponibilizado com a respetiva licença Creative Commons

7. Para o professor

Sugestões de resposta aos desafios:

- 1.1

*Doc

RAD

✕

*teste.py

2/14

```
#=====
from ti_hub import *
from math import *
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
lum=light_level("IN 1")
lum.range(0,100)
luz=led("OUT 1")
motor=servo("OUT 3")
```
- 1.1

*Doc

CAPS

RAD

✕

*teste.py

6/22

```
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
lum=light_level("IN 1")
lum.range(0,100)
luz=led("OUT 1")
motor=servo("OUT 3")

print(lum.measurement())
luz.blink(10,2)
motor.set_position(90)
```
- 1.1

*Doc

RAD

✕

*teste.py

20/24

```
lum.range(0,100)
luz=led("OUT 1")
motor=servo("OUT 3")
print(lum.measurement())

if lum.measurement()<20:
    luz.blink(3,2)
    motor.set_position(90)
else:
    luz.blink(10,2)
    motor.set_position(-90)
```

Conclusão

```
*teste.py 23/26
lum.range(0,100)
luz=led("OUT 1")
motor=servo("OUT 3")
print(lum.measurement())
p=0 # variável para indicar estado do portão
while get_key() != "esc":
    if lum.measurement()<20 and p==1:
        luz.blink(3,2)
        motor.set_position(-90)
        p=0 # Informa que portão está fechado
    if lum.measurement()>=20 and p==0:
        luz.blink(10,2)
        motor.set_position(90)
        p=1 # Informa que portão está aberto
```



Este recurso pedagógico é disponibilizado com a respetiva licença Creative Commons