Chapitre 10. Interfaces homme machine et robotique

Table des matières

Remplissez le jupyter notebook suivant en vous aidant de votre livre de Première NSI de Serge BAYS .

- Pour répondre, double-cliquez sur **Réponse** et complétez la zone en-dessous. Puis cliquez sur le bouton *Exécuter*.
- **Important : pour fermer votre jupyter notebook, cliquez sur :**

Fichier / Créer une nouvelle sauvegarde

puis sur :

Fichier / Fermer et Arrêter

- Ecrivez ci-dessous votre prénom et votre nom :

Réponse :

1. Périphériques d'entrées et de sorties

1.1 Introduction

Lisez le paragraphe **Périphériques** en haut de la p. 201

1) Lorsque l'utilisateur agit sur un périphérique d'entrée que fait alors la machine ?

Réponse :

2) Que peut-on dire d'un programme qui reste en attente permanente d'un signal, sur un des périphériques d'entrée, pour gérer cet événement ensuite ?

Réponse :

1.2 Capteurs et actionneurs

Lisez le paragraphe **Capteurs et actionneurs** au milieu de la p. 201

3) De façon plus générale les périphériques d'entrée font partie de l'ensemble des capteurs et les périphériques de sortie font partie de l'ensemble des actionneurs.
On donne ci-dessous quelques capteurs et quelques actionneurs. Classez les dans l'une ou l'autre des deux catégories :

- Buzzer
- Bouton poussoir
- LED
- Ecran
- Capteur de distance
- Manette à 5 boutons
- Servomoteur

Réponse :

- Capteurs :

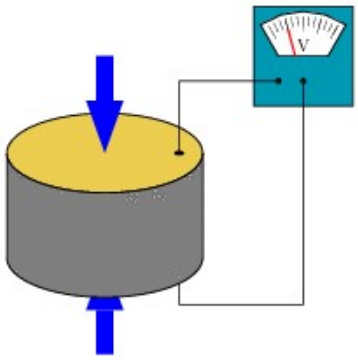
  ...
  ...
  ...

- Actionneurs :

  ...
  ...
  ...
  ...

4) Qu'est-ce qu'un signal analogique ?

Réponse :

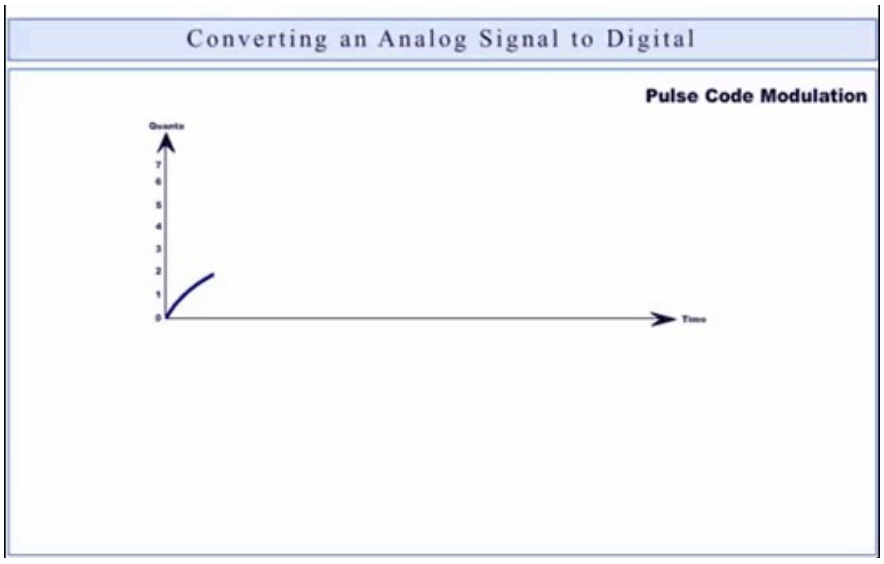


Capteur qui transforme une grandeur physique en signal analogique

Echantillonnage : passer d'un signal analogique à un signal numérique

5) Le signal analogique est échantillonné à des intervalles de temps régulier. Les valeurs des échantillons sont ensuite transformées en valeurs binaires. Quels sont les noms des dispositifs qui transforment les valeurs analogiques en valeurs binaires ?

Réponse :



Conversion du signal analogique en échantillons numériques

6) Supposons qu'un signal analogique puisse varier entre 0 et 5 volts (capteur de pression par exemple). Supposons que le convertisseur analogique numérique ait une sortie codée sur 1 octet. Quelle est la résolution (en mV) des mesures ?

Réponse :

7) Cette résolution est-elle suffisante si la précision garantie par le constructeur du capteur est 10 mV ?

Réponse :

Conclusion :

- Une fois numérisé, le signal peut être traité par un programme informatique.
- La résolution doit être suffisante pour profiter de la précision des mesures du capteur.
- La fréquence d'échantillonnage doit être suffisante pour traduire les variations rapides dans le signal analogique que le capteur peut envoyer.

1.3 Systèmes embarqués

Lisez le paragraphe **Systèmes embarqués** dans le bas de la p. 201 au haut de la p. 202

Système embarqué : des fonctions d'ordinateur dans un objet mobile

8) Un système embarqué (dans une voiture moderne par exemple) comprend :

- Un CPU (Central Processing Unit) c'est à dire un processeur qui effectue des calculs.
- Des mémoires.
- Des ports d'entrée et de sortie pour y connecter les capteurs et les actionneurs.
- Les capteurs (Ex: capteur de présence de goutelettes d'eau sur le pare-brise).
- Les actionneurs (Ex : les moteurs d'essuie-glaces).
- Une IHM.

  Expliquez brièvement ce que sont les IHM :

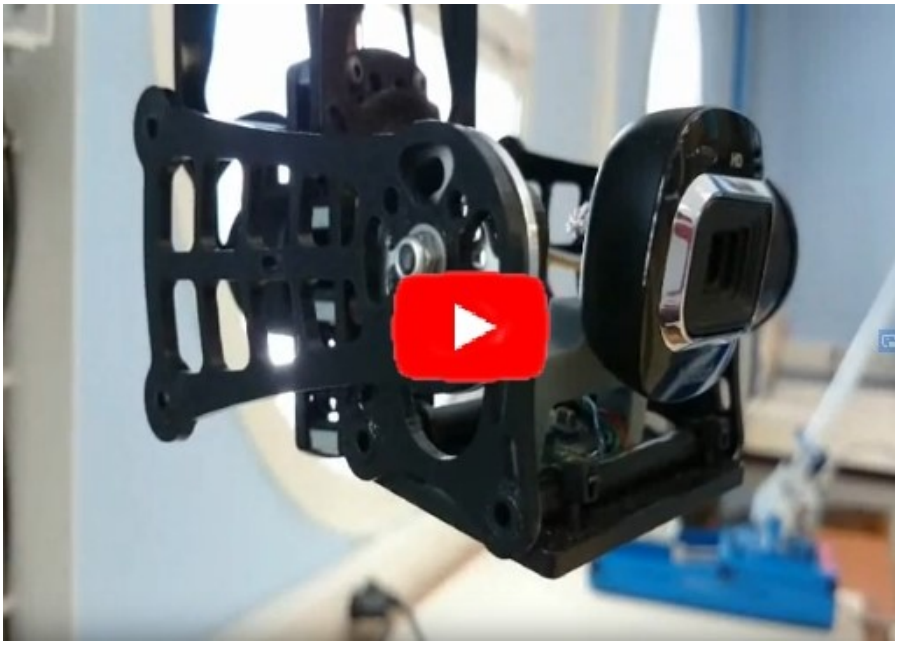
Réponse :

Régulation : des capteurs qui peuvent rétro agir sur les actionneurs pour une meilleure précision

Dans un système embarqué, les capteurs permettent d'allumer ou d'éteindre des actionneurs. Par exemple si la luminosité ambiante devient inférieure à une certaine valeur alors les phares sont allumés. Mais ils peuvent aussi réguler le fonctionnement des actionneurs par une boucle de régulation. Par exemple :

- Le conducteur fixe une vitesse.
- Le capteur mesure la vitesse du véhicule.
- Cette mesure est prise en compte par le logiciel qui commande l'accélérateur et le frein.
- Cet boucle de régulation permet au véhicule d'atteindre la vitesse de consigne de s'y maintenir malgré les perturbations comme du vent, des montés et des descentes.

- Visionnez la vidéo ci-dessous qui **illustre un asservissement d'angle d'une caméra** de drone :
  - En boucle ouverte, c'est à dire qu'il n'y a pas de capteur qui retourne de mesure de l'angle réel.
  - En boucle fermée, c'est à dire que la mesure de l'angle réel est comparée à la consigne de position donnée par l'utilisateur.



(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/bases_de_l_asservissement.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/bases_de_l_asservissement.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/bases_de_l_asservissement.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/bases_de_l_asservissement.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/bases_de_l_asservissement.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/bases_de_l_asservissement.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/bases_de_l_asservissement.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/bases_de_l_asservissement.mp4)

1.4 Objets connectés (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/bases_de_l_asservissement.mp4)

Lisez le paragraphe **Objets connectés** au milieu de la p. 202

9) Qu'est-ce qu'un objet connecté ?

Réponse :

Qu'est-ce qu'une ligne de commande bash ?

- Tout d'abord, il faut définir ce qu'est un shell. Un shell, c'est un interpréteur de commandes, et en même temps un langage de programmation. Lorsque vous tapez des commandes sur une console (ou un terminal), vous envoyez ces commandes au shell, qui va les interpréter pour renvoyer le résultat.
- Bash, c'est l'un des nombreux shells existants dansle monde Unix. C'est en fait un clône du Bourne Shell (bash = Bourne Again SHell). C'est aussi le shell standard sous Linux.

Un exemple le plus simple possible : allumer un relais par une ligne de commande bash, à travers Internet.

D'après une idée de Tommy Desrochers

- Visionnez la vidéo ci-dessous (11mn) qui donne un exemple d'objet connecté. L'objet est un Raspberry et la connexion est une **connexion HTTP** :

Les points suivants sont abordés :

- Mise à jour du Raspberry Pi.
- Installation d'un serveur web et d'un serveur PHP sur le Raspberry Pi.
- Vérification de la présence de la bibliothèque Linux "wiringPi (http://wiringpi.com/)" qui contient les fonctions pour accéder aux entrées / sorties du Raspberry (Si vous ouvrez votre Raspberry, vous pouvez voir un long connecteur 2 fois 20 pins qui sont reliées au "GPIO").

- Pour pouvoir contrôler les ports GPIO en tant qu'utilisateur jdubois par exemple, il est nécessaire que :

- L'utilisateur **jdubois** du Raspberry soit membre du groupe gpio. Pour ajouter jdubois au groupe gpio, saisissez :

```
<pre>grep -i gpio /etc/group
```

  En clair :

  group est un fichier situé dans le dossier /etc

  Ce fichier contient la liste des groupes d'utilisateurs du Rasbpberry.

  grep est un filtre qui n'affiche que la ligne gpio

  -i est un attribut qui précise que la casse (majuscule ou minuscule) n'a pas d'importance.</pre>

- Si le résultat de cette commande est :

```
<pre>gpio:x:997:</pre>
```

  alors jdubois sera incapable d'agir sur les pins du GPIO et les commandes de configuration des pins du gpio ne marcheront pas. Dans ce cas, saisissez :

```
<pre>sudo adduser jdubois gpio</pre>
```

- Vérifiez maintenant que le groupe gpio possède un utilisateur de plus et que c'est jdubois en saisissant :

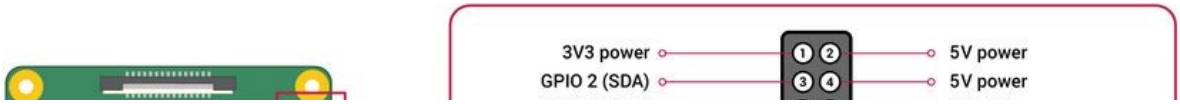
```
<pre>grep -i gpio /etc/group</pre>
```

  Le résultat de cette commande doit être :

```
<pre>gpio:x:997:jdubois</pre>
```

- **Il est indispensable de rebooter le Raspberry pour que cette modification prenne effet.**

  Les ports GPIO (anglais : General Purpose Input/Output, littéralement Entrée-sortie à usage général) sont des ports d'entrées-sorties très utilisés dans le monde des microcontrôleurs, en particulier dans le domaine de l'électronique embarquée, qui ont fait leur apparition aux débuts des années 1980.



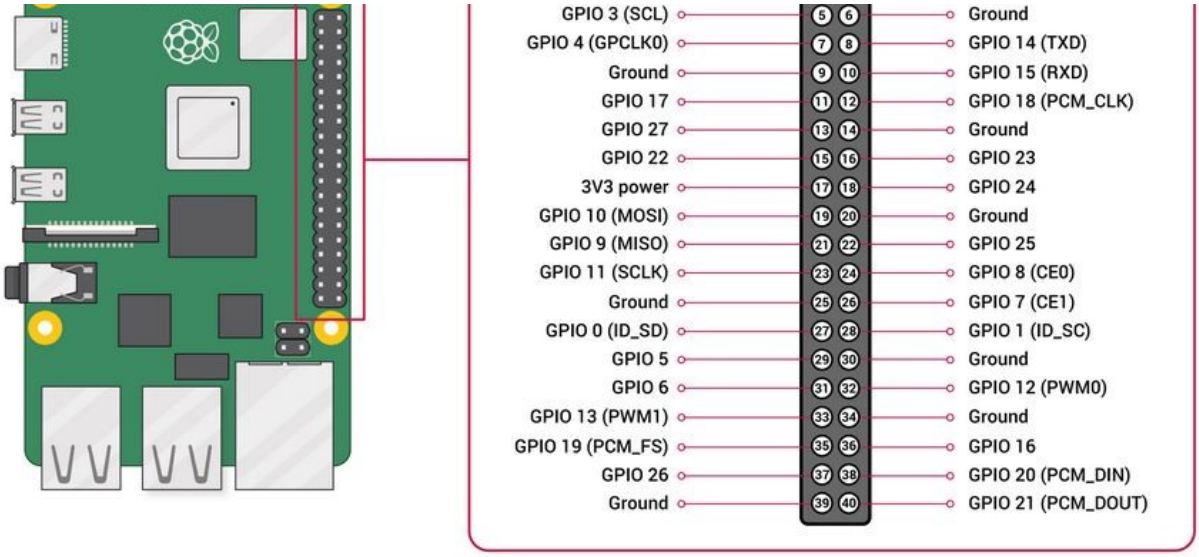
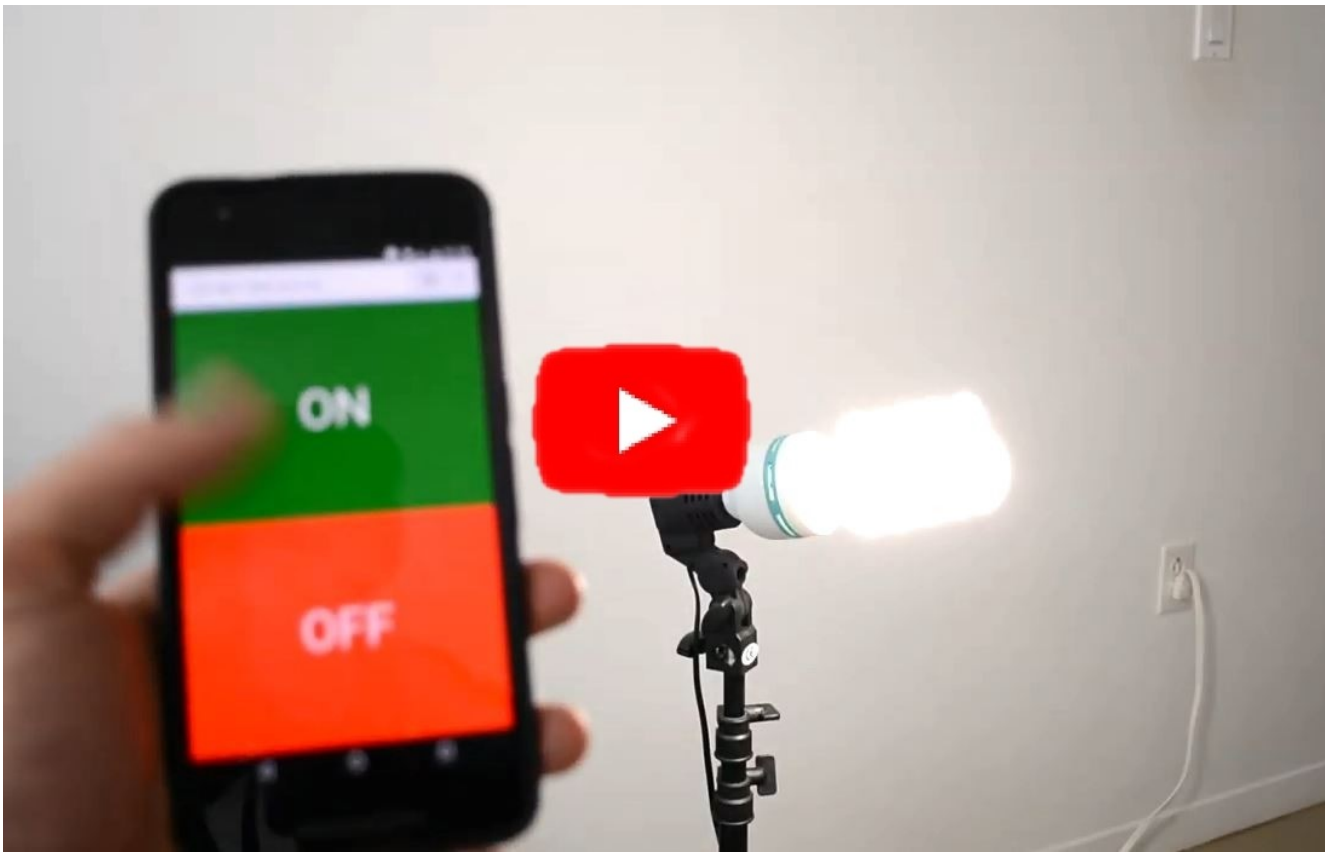


Schéma des pins (broches) du GPIO sur un Raspberry 3B+
"IO" signifie Input Output.
Il faut donc toujours programmer le sens
du port avant l'utilisation.
On le met en Input si le Raspberry doit "lire" à parir d'un capteur.
On le met en Output si le Raspberry doit "écrire" sur un actionneur.
On va utiliser les numéros de ports qui sont à l'extérieur sur ce schéma.
Par exemple la deuxième pin en haut à gauche porte le n°GPIO 2 (et non 3)
La numérotation extérieure est dite "BCM". Elle est plus pratique en programmation.
L'autre numérotation est dite "Board". C'est la numérotation dans l'ordre physique
sur le connecteur. Sur le schéma, ce sont les numéros à l'intérieur des petits cercles blancs.

- Ecriture du fichier index.php qui est la page d'accueil du serveur sur laquelle on programme un formulaire à 2 boutons et qui redirige vers un deuxième fichier script.php
- Ecriture du fichier stylesheet.css qui formate les 2 boutons.
- Ecriture du fichier script.php qui contient des commandes Linux pour :
  - Commander à la pin 17 (en numérotation BCM) du GPIO de se mettre en mode "sortie".
  - Ecrire sur la pin 17 du GPIO (celle qui commande le relais d'allumage de la lampe) selon la valeur qui a été postée par le formulaire à 2 boutons.



(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_objet_connecte.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_objet_connecte.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_objet_connecte.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_objet_connecte.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_objet_connecte.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_objet_connecte.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_objet_connecte.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_objet_connecte.mp4)

Conclusion : (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_objet_connecte.mp4)

Dans cet exemple, nous avons bien les fonctions d'un système embarqué :

- Un CPU (le microprocesseur du Raspberry).
- Des mémoires (celles du Raspberry).
- Des ports d'entrée et de sortie pour y connecter les capteurs et les actionneurs (les ports GPIO du Raspberry).
- Les capteurs (absents ici, mais dans un projet plus complet, on pourrait connecter à une pin du GPIO un capteur de lumière extérieure pour allumer la lampe quand la nuit arrive).
- Les actionneurs (le relais qui sert à allumer la lampe).
- Une IHM (le formulaire HTML de la page index.php, avec les deux boutons qui permettent à une personne d'interagir avec le système embarqué).

Le site de l'auteur de la vidéo :

https://TommyDesrochers.com (https://TommyDesrochers.com)

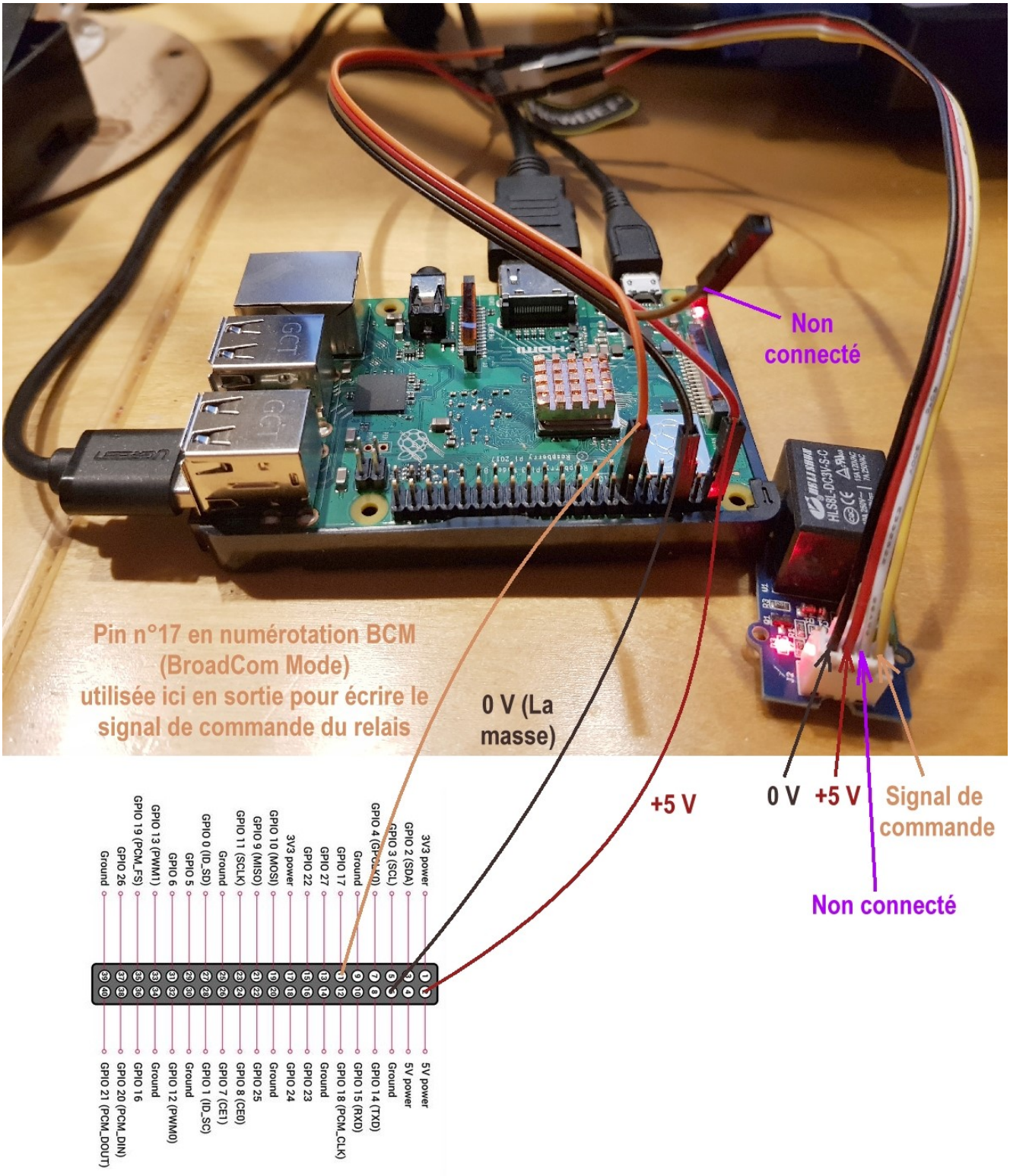
A nous de jouer !

Au départ, le Raspberry étant **éteint**, nous branchons les 3 fils d'un relais sur le connecteur 2 x 20 broches du GPIO.

- Pour le signal de commande du relais, nous choisissons le port GPIO n°17 (selon la *numérotation BCM*).

Physiquement, c'est la pin (la broche) n°11 (selon la numérotation "board"). C'est la sixième pin de la rangée intérieure dans le Reaspberry, en partant du côté opposé aux prises USB.

- Nous devons aussi alimenter le relais avec une tension de 5V. Pour cela nous avons de fils : le rouge pour le +5 V et le noir pour le ground (la masse) c'est à dire le 0 V.



1. Mises à jour et installations

- Dans un terminal sur le Raspberry on commence par mettre à jour le système d'exploitation Raspbian et les applications. Pour cela on tape les commandes bash suivantes :

```
sudo apt-get update

sudo apt-get upgrade
```

Remarque : pour éviter d'avoir à taper 'O' pour dire Oui, ou 'y' pour dire yes, on peut ajouter l'option -y. Ainsi la commande upgrade répond automatiquement 'oui' à toutes les questions.

```
sudo apt-get upgrade -y
```

- Installer les serveurs Apache et PHP : si vous ne l'avez pas déjà fait, voyez le document :

http://www.astrovirtuel.fr/nsi/premiere/premiere2019-2020/serveurs_http_et_php.pdf (http://www.astrovirtuel.fr/nsi/premiere/premiere2019-2020/serveurs_http_et_php.pdf)

- Vérifier que la bibliothèque WiringPi est déjà installée sur votre Raspberry :

La bibliothèque WiringPi est accompagnée d'un programme nommé gpio. Pour s'assurer que Wiringpi est effectivement installée, demandons à voir la version de gpio. Dans le terminal saisissez la commande :

```
gpio -v
```

La version de gpio installée doit apparaitre.

Puis saisissez :

```
gpio readall
```

Un tableau avec le nom des pins du GPIO doit apparaitre. Conclusion : la bibliothèque WiringPi est effectivement présente !

2. Ecriture de index.php

- Rendez-vous dans var/www/html/ qui est le dossier racine du serveur Apache : supprimez index.php et style.css s'ls existent.
- Rendez-vous dans votre dossier home/jdubois/Documents/webs et supprimez index.php et style.css s'ls existent.
- Avec l'éditeur de programme Geany : Fichier / Nouveau (selon modèle) / file.php

Supprimez les commentaires du début qui sont mis automatiquement par Geany.

Copiez deans le code suivant, puis enregistrez et fermez le fichier index.php.

```
<!doctype html>
<html lang="fr">
<head>
    <title>Contrôle GPIO</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="stylesheet.css">
</head>

<body>
    <!-- Formulaire à deux boutons qui appellent script.php-->
    <form action="script.php" method="post">
        <input type="submit" name="executer" value="ON" class="button" id="mon_bouton_on">
        <!-- "executer" sera un nom de variable connu par le code PHP dans la page script.
php -->
        <br/>
        <input type="submit" name="executer" value="OFF" class="button" id="mon_bouton_of
f">
    </form>
</body>
</html>
```

3. Ecriture du fichier stylesheet.css

- Avec l'éditeur de programme Geany : Fichier / Nouveau

Enregistrez sous le nom stylesheet.css dans le dossier home/jdubois/Documents/webs

Copiez dedans le code suivant, puis enregistrez.

```
html, body
{
    margin: 0;
}

.button
{
    border: none;
    color: white;
    text-align: center;
    font-size: 10em;
    padding: 25px 25px;
    cursor: pointer;
    width: 100%;
    height: 50vh;
}

#mon_bouton_on
{
    background-color: green;
}

#mon_bouton_off
{
    background-color: red;
}
```

Remarque : Dans ce fichier de style, on reconnait les 3 manières de désigner un élément de la page web :

- html, body sont des balises
- .button est une classe
- #mon_bouton_on est un identifiant placé dans une balise

4. Le fichier de commande script.php

- Avec l'éditeur de programme Geany : Fichier / Nouveau / selon modèle / php

Supprimez les commentaires du début qui sont mis automatiquement par Geany.

Supprimez également toute la partie HTML.

Enregistrez sous le nom script.php dans le dossier home/jdubois/Documents/webs

Copiez dedans le code suivant, puis enregistrez.

```php
<?php

// L'instruction system est équivalente à saisir dans le terminal
// une commande. Cette commande est la chaine donnée en argument.
system("gpio -g mode 17 out");


/* Remarquez le symbole dollar qui signifie variable
 * On récupère ici un élément du tableau associatif dollar_POST
 * dont les indices sont les 'name' des éléments du formulaire */

if($_POST['executer'] == 'ON'){
    system("gpio -g write 17 1");
}
else{
    system("gpio -g write 17 0");
}

// La fonction header redirige le navigateur vers index.php
header('Location: index.php');

?>
```

Remarques :

- L'instruction PHP **system()** permet d'exécuter une ligne de commande bash comme si elle était saisie dans un terminal.
- Ce qui veut dire qu'on peut tout à fait saisir dans le terminal l'instruction

```
gpio -g mode 17 out
```

pour définir le port GPIO 17 (en numérotation BCM) comme étant une sortie. C'est normal puisqu'on y connecte un actionneur.

Ensuite on peut saisir dans le terminal :

```
gpio -g write 17 1  (qui active le relais)

gpio -g write 17 0  (qui éteint le relais)
```

- L'attribut -g est là pour dire que la numérotation est en BCM.
- BCM est l'abréviation de "BroadCom Mode"
- La numérotation BCM (on dit aussi la numérotation GPIO) est aussi utilisée par Python en important la bibliothèque RPi.GPIO. On aura l'occasion de voir un exemple avec Python pour l'allumage d'une LED à la fin du § 1.5

- Pour pouvoir contrôler les ports GPIO depuis Internet comme c'est le cas ici, il est nécessaire que :

- L'utilisateur **www-data** du Raspberry (qui n'est autre que le programme du serveur Apache) soit membre du groupe gpio. Pour ajouter www-data au groupe gpio, saisissez :

```
<pre>grep -i gpio /etc/group
```

  En clair :

  group est un fichier situé dans le dossier /etc

  Ce fichier contient la liste des groupes d'utilisateurs du Rasbpberry.

  grep est un filtre qui n'affiche que la ligne gpio

  -i est un attribut qui précise que la casse (majuscule ou minuscule) n'a pas d'importance.</pre>

- Si le résultat de cette commande est :

```
<pre>gpio:x:997:jdubois</pre>
```

  alors le fichier php sera incapable d'agir sur les pins du GPIO et la commande par requête http du relais ne marchera pas. Dans ce cas, saisissez :

```
<pre>sudo adduser www-data gpio</pre>
```

- Vérifiez maintenant que le groupe gpio possède un utilisateur de plus et que c'est www-data en saisissant :

```
<pre>grep -i gpio /etc/group</pre>
```

  Le résultat de cette commande doit être :

```
<pre>gpio:x:997:jdubois,www-data</pre>
```

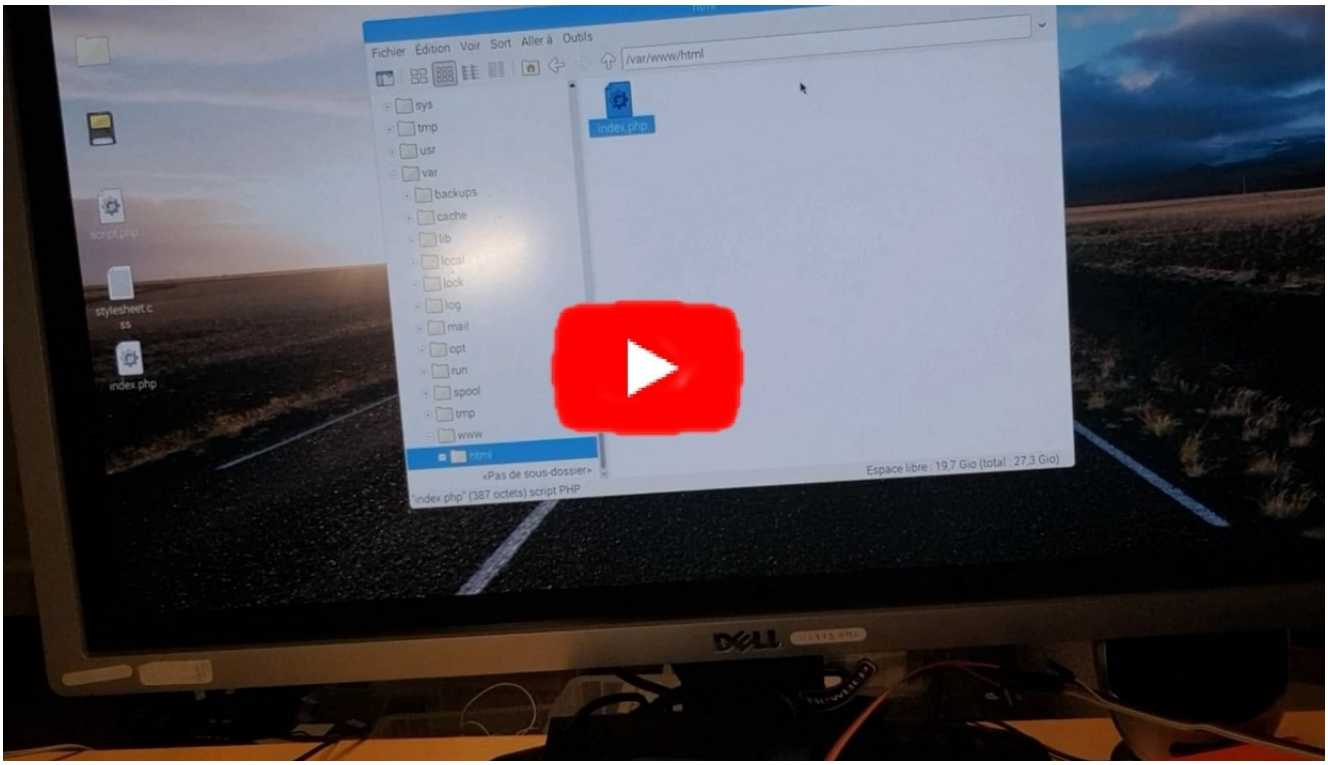
- **Il est indispensable de rebooter le Raspberry pour que cette modification prenne effet.**

- Vous pouvez trouver ici (https://wiki.debian-fr.xyz/Commandes_utilisateurs_et_groupes#adduser) d'autres commandes pour la gestion des utilisateurs et des groupes d'une machine dont vous êtes administrateur. Rappelez-vous que beaucoup de ces commandes qui ont un impact sur le système et doivent donc être exécutées en temps que superutilisateur, c'est à dire qu'il faut commencer par :

```
sudo
```

5. Le montage

- Voyez la vidéo ci-dessous qui montre la réalisation du montage " Commande par HTTP du relais branché au Raspberry ".



(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/relais2.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/relais2.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/relais2.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/relais2.mp4)

- Dans cet exemple, on a ainsi transformé notre Raspberry en objet connecté qui communique par HTTP sur une interface Web.

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/relais2.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/relais2.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/relais2.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/relais2.mp4)

1.5 Les robots (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/relais2.mp4)

Lisez le paragraphe **Les robots** du milieu de la p. 202 au milieu de la p. 204

10) Lorsqu'on envoie une tension aux moteurs des roues d'un robot, qu'est-ce qu'un capteur de vitesse par le robot permet de faire en fonction de la vitesse recueillie ?

Réponse :

11) Dans quel langage sont souvent programmés les robots ?

Réponse :

12) Quel langage de programmation peut-on utiliser pour programmer un nano ordinateur Raspberry Pi muni d'un connecteur GPIO ?

Réponse :

- De façon générale, tout système automatique capable de manipuler des objets ou d'exécuter des opérations selon un programme fixe, modifiable ou adaptable peut être qualifié de "robot" (ou "bot" en abrégé). Par exemple un logiciel qui donne des ordres d'achat et de vente d'actions en bourse, un logiciel qui lit et écrit des messages sur une messagerie ou sur un chat, un logiciel qui joue à un jeu etc.
- Mais dans le cas d'un système embarqué, on pourra le nommer "robot" quand il est assez évolué pour avoir comme actionneurs des moteurs permettant de le faire se déplacer ou de prendre des objets et des capteurs comme des caméras permettant la reconnaissance de formes etc.
- Le robot est en général télécommandé ou au moins supervisé à distance par un humain. Mais dans certains cas, le logiciel du robot doit être capable de beaucoup d'autonomie face aux situations rencontrées.

Exemple 1 :

Le robot *Opportunity* qui s'est posé sur Mars le 25 janvier 2004 dans la région équatoriale de Terra Meridiani avait pour but d'étudier la géologie de Mars et de déterminer en particulier le rôle joué par l'eau dans l'histoire de la planète. Sa mission qui devait durer 90 jours s'est achevée officiellement en février 2019. Il devait se déplacer en évitant les obstacles. Le pilotage par un humain à disance, en temps réel était impossible. A cause de la très grande distance entre la Terre et Mars, il s'écoulait plusieurs dizaines de minutes entre le moment où un ordre était envoyé du centre de contrôle et le moment où le signal radio était de retour sur Terre.

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/opportunity.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/opportunity.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/opportunity.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/opportunity.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/opportunity.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/opportunity.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/opportunity.mp4)

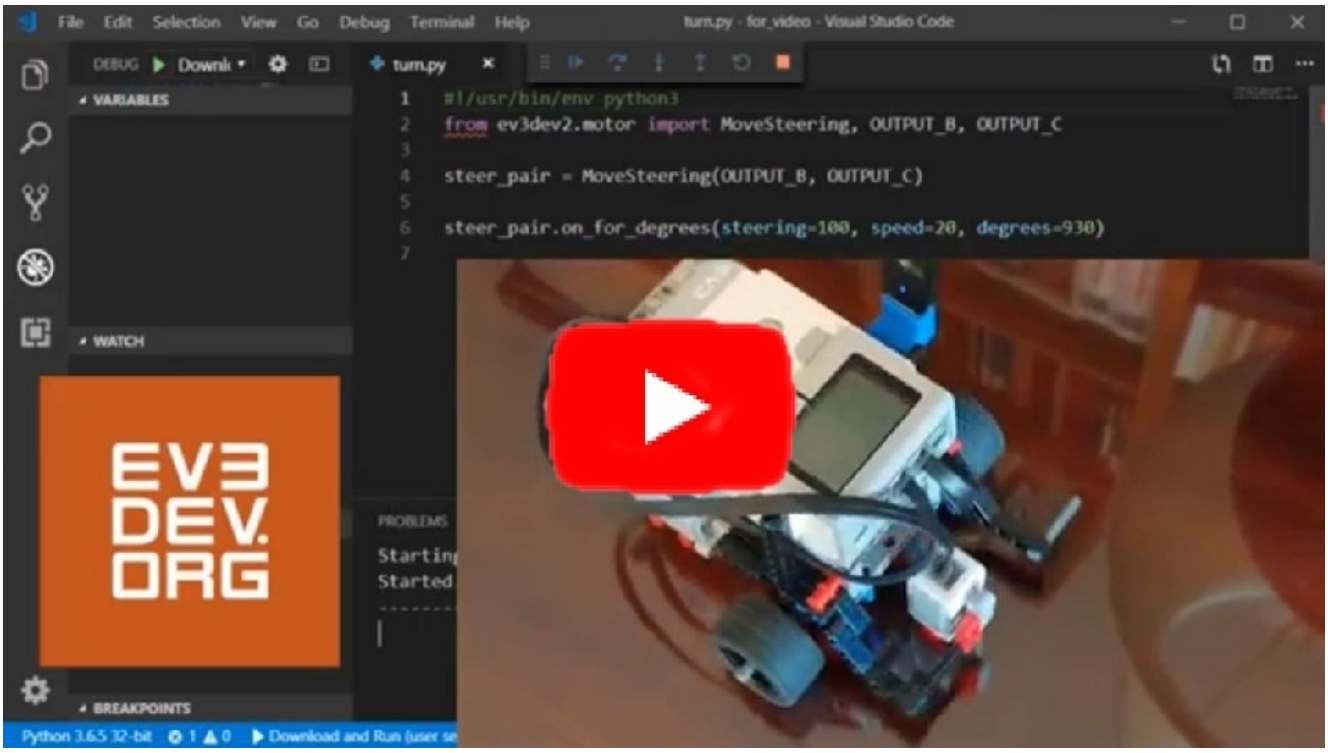
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/opportunity.mp4)

Exemple 2 : (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/opportunity.mp4)

Le robot *Lego EV3* (pour Evolution, 3rd generation) :

- Un CPU (d'architecture ARM tout comme le microprocesseur du Raspberry). Il tourne avec le système d'exploitation ev3dev qui est basé sur Linux (Debian plus précisément).
- Des mémoires.
- Des ports d'entrée et de sortie pour y connecter les capteurs et les actionneurs.
- Les capteurs (capteur de contact, capteur de distance, capteur de lumière etc).
- Les actionneurs (moteurs).
- Une IHM (l'écran monochrome).

C'est aussi un objet connecté puisqu'il peut communiquer par Wi-Fi ou par bluetooth avec un ordinateur ou un smartphone.



(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/robot_ev3.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/robot_ev3.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/robot_ev3.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/robot_ev3.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/robot_ev3.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/robot_ev3.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/robot_ev3.mp4)

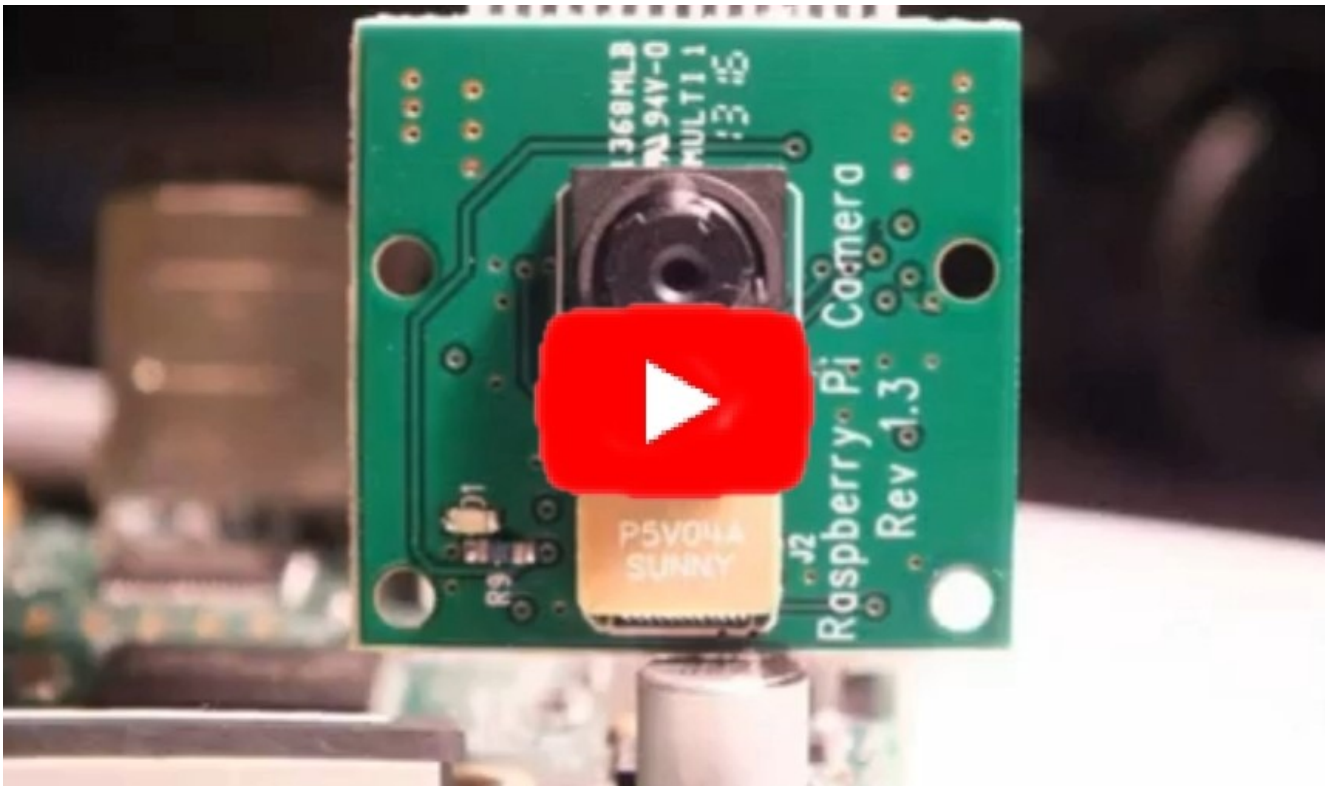
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/robot_ev3.mp4)

Un capteur particulier : la Picamera commandée en Python sur le Raspberry lui-même (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/robot_ev3.mp4)

D'après le livre dernière ligne de la p. 202 et p. 203

- Branchement d'une Picaméra

1.Installation d'une PiCaméra sur une Raspberry Pi :



(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/installation_picamera.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/installation_picamera.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/installation_picamera.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/installation_picamera.mp4)

2.Dans un terminal, saisissez la ligne de commande qui vous enregistre dans les utilisateurs du groupe video :

```
sudo usermode -a -G video jdubois
```

Remplacez jdubois par votre identifiant puis redémarrer le Raspberry.

3.Ecriture et test des programmes :

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/installation_picamera.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/installation_picamera.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/installation_picamera.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/installation_picamera.mp4)

Premier programme : live_picamera.py (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/installation_picamera.mp4)

Ce programme est le premier programme de la p. 203. Il retransmet une vidéo en direct pendant 5 secondes.

Après avoir branché la picaméra sur votre Raspberry, ouvrez le logiciel Spyder (Cliquez en haut à gauche de votre écran sur la framboise, puis allez dans Programmation). Copiez le code ci-dessous dans l'éditeur (partie gauche de Spyder). Puis exécutez le programme (cliquez sur la flèche verte). Vous devriez voir à l'écran ce que voit la caméra pendant 5 secondes.

In [ ]:
```
from picamera import PiCamera  #  Importation du module PiCamera de la bibliothèque pic
amera
from time import sleep  #  Importation de la fonction sleep de la bibliothèque time

camera = PiCamera()  #  Création d'un objet du type PiCamera nommé camera
camera.start_preview()
sleep(5)  #  5 secondes de pose.
camera.stop_preview()
```

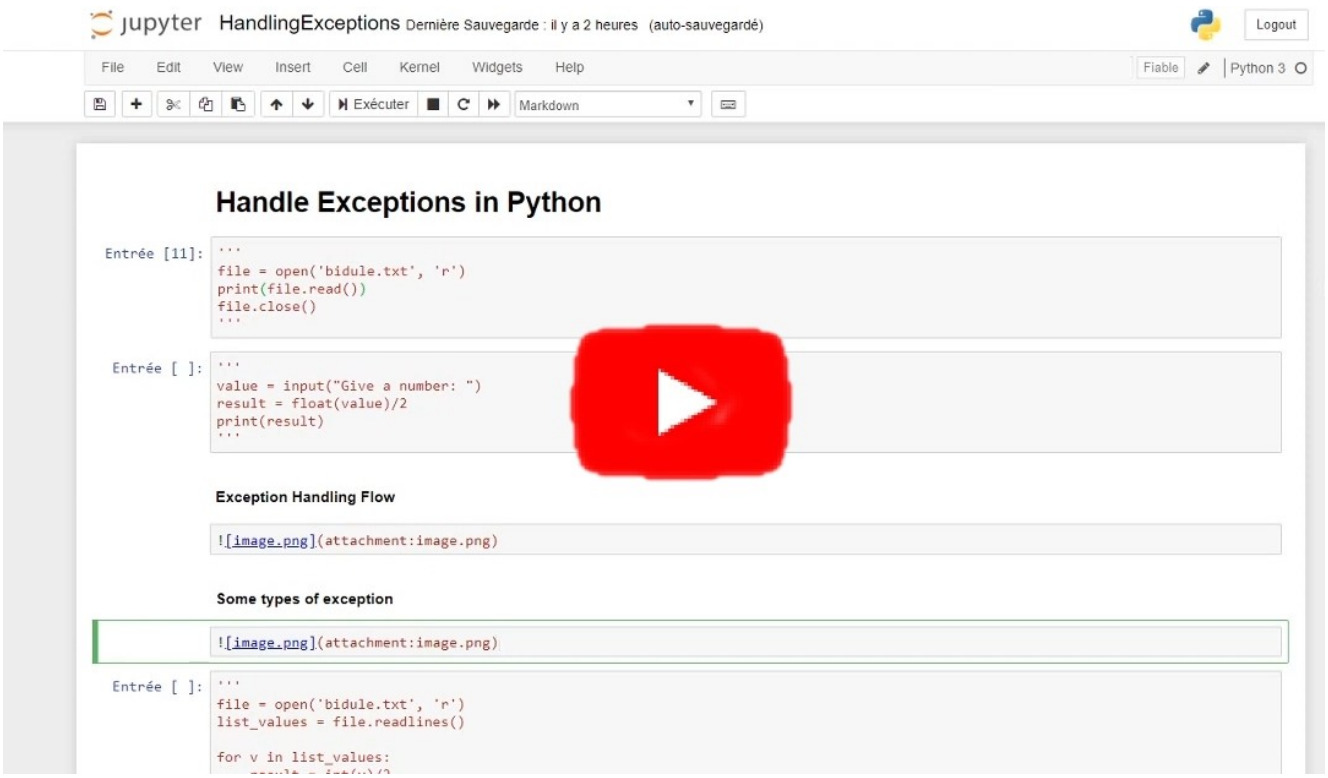
- Voyez ici (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/1.picamera.mp4) une vidéo de démonstration de l'exécution du fichier live_picamera.py
- Comme on le constate dans la vidéo, le fonctionnement du programme n'est pas satisfaisant. En effet, après une première exécution, le programme ne peut pas être relancé. Il faut fermer la console en cours afin d'obliger Spyder à relancer un nouveau noyau du programme Python.
- En fait le bug provient du fait qu'on ne s'est pas assuré de fermer l'objet 'camera' après utilisation. Et donc Spyder voit que la caméra n'est pas arrêtée et ne peut donc pas la démarrer.
- Pour réparer ce bug, modifiez ainsi le code :

In [ ]:
```
from picamera import PiCamera  #  Importation du module PiCamera de la bibliothèque pic
amera
from time import sleep  #  Importation de la fonction sleep de la bibliothèque time

camera = PiCamera()  #  Création d'un objet du type PiCamera nommé camera
try:
    camera.start_preview()
    sleep(5)  #  5 secondes de pose.
    camera.stop_preview()
finally:
    camera.close()  #  Dans la section finally on met les instructions indispensables à
notre programme (ici la fermeture).
```

- Cette fois le programme fonctionne correctement : on peut le relancer sans devoir fermer la console en cours dans Spyder.
- Les instructions qui ont été utilisées ici font partie d'une structure plus vaste qu'on appelle "gestion des exceptions" ou "gestions des erreurs". Cette structure contient en fait quatre instructions :
  - try
  - except
  - else
  - finally

Si vous voulez en savoir plus sur cette structure, voyez la vidéo ci-dessous (21 mn) :



(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/gestion_des_exceptions_en_python.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/gestion_des_exceptions_en_python.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/gestion_des_exceptions_en_python.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/gestion_des_exceptions_en_python.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/gestion_des_exceptions_en_python.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/gestion_des_exceptions_en_python.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/gestion_des_exceptions_en_python.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/gestion_des_exceptions_en_python.mp4)

Deuxième programme : une_photo_picamera.py (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/gestion_des_exceptions_en_python.mp4)

Ce programme est le deuxième programme de la p. 203. Il fait une temporisation de 5 secondes puis il prend une photo et la stocke dans le dossier Images de l'utilisateur.

In [ ]:
```
from picamera import PiCamera  #  Importation du module PiCamera de la bibliothèque pic
amera
from time import sleep  #  Importation de la fonction sleep de la bibliothèque time

camera = PiCamera()  #  Création d'un objet du type PiCamera nommé camera
camera.start_preview()
sleep(5)  #  5 secondes de pose.
camera.capture('/home/jdubois/Images/img1.jpg')  #  Remplacez jdubois par votre nom d'u
tilisateur.
camera.stop_preview()
```

- Voyez ici (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/2.picamera.mp4) une vidéo de démonstration de l'exécution du fichier une_photo_picamera.py
- Le programme fonctionne, mis à part le bug déjà signalé
- Pour réparer ce bug, modifiez ainsi le code :

In [ ]:
```
from picamera import PiCamera  #  Importation du module PiCamera de la bibliothèque pic
amera
from time import sleep  #  Importation de la fonction sleep de la bibliothèque time

camera = PiCamera()  #  Création d'un objet du type PiCamera nommé camera
try:
    camera.start_preview()
    sleep(5)  #  5 secondes de pose.
    camera.capture('/home/jdubois/Images/img1.jpg')  #  Remplacez jdubois par votre nom
d'utilisateur.
    camera.stop_preview()
finally:
    camera.close()  #  Dans la section finally on met les instructions indispensables à
notre programme (ici la fermeture).
```

Troisième programme : dix_photos_picamera.py

Ce programme est inspiré du troisième programme de la p. 203. Il fait une temporisation de 5 secondes puis il prend 10 photos et les stocke dans le dossier Images de l'utilisateur.

In [ ]:
```
from picamera import PiCamera  #  Importation du module PiCamera de la bibliothèque pic
amera
from time import sleep  #  Importation de la fonction sleep de la bibliothèque time

camera = PiCamera()  #  Création d'un objet du type PiCamera nommé camera
camera.start_preview()
sleep(5)  #  5 secondes de pose.
for i in range(10):
    sleep(5)
    camera.capture('/home/jdubois/Images/img%s.jpg' % i)  #  % est un opérateur de form
atage de chaine de caractères
                                                          #  %s est remplacé dans la ch
aine par la valeur (convertie
                                                          #   en chaine de caractères)
de ce qui suit l'opérateur %.
camera.stop_preview()
```

- Voyez ici (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/3.picamera.mp4) une vidéo de démonstration de l'exécution du fichier une_photo_picamera.py
- Le programme fonctionne, mis à part le bug déjà signalé
- Pour réparer ce bug, modifiez ainsi le code :

In [ ]:
```
from picamera import PiCamera  #  Importation du module PiCamera de la bibliothèque pic
amera
from time import sleep  #  Importation de la fonction sleep de la bibliothèque time

camera = PiCamera()  #  Création d'un objet du type PiCamera nommé camera
try:
    camera.start_preview()
    sleep(5)  #  5 secondes de pose.
    for i in range(10):
        sleep(5)
        camera.capture('/home/jdubois/Images/img%s.jpg' % i)  #  Remplacez jdubois par
votre nom d'utilisateur.
    camera.stop_preview()
finally:
    camera.close()  #  Dans la section finally on met les instructions indispensables à
notre programme (ici la fermeture).
```

Quatrième programme : une_video_picamera.py

Ce programme est inspiré du quatrième programme de la p. 203. Il commence à enregistrer une vidéo au format flux vidéo brut (format .h264) et fait une temporisation de 10 secondes. Donc on aura une vidéo de 10 s. Il la stocke dans le dossier Images de l'utilisateur. Evitez le dossier Vidéo à cause du é qui n'est pas un caractère ASCII.

In [ ]:
```
from picamera import PiCamera  #  Importation du module PiCamera de la bibliothèque pic
amera
from time import sleep  #  Importation de la fonction sleep de la bibliothèque time

camera = PiCamera()  #  Création d'un objet du type PiCamera nommé camera
camera.start_preview()
camera.start_recording('/home/jdubois/Images/ma_video.h264')
sleep(10)  #  10 secondes de pose (la durée de l'enregistrement).
camera.stop_recording()
camera.stop_preview()
```

- Voyez ici (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/4.picamera.mp4) une vidéo de démonstration de l'exécution du fichier une_photo_picamera.py
- Le programme fonctionne, mis à part le bug déjà signalé
- Pour réparer ce bug, modifiez ainsi le code :

In [ ]:
```
from picamera import PiCamera  #  Importation du module PiCamera de la bibliothèque pic
amera
from time import sleep  #  Importation de la fonction sleep de la bibliothèque time

camera = PiCamera()  #  Création d'un objet du type PiCamera nommé camera
try:
    camera.start_preview()
    camera.start_recording('/home/jdubois/Images/ma_video.h264')
    sleep(10)  #  10 secondes de pose (la durée de l'enregistrement).
    camera.stop_recording()
    camera.stop_preview()
finally:
    camera.close()  #  Dans la section finally on met les instructions indispensables à
notre programme (ici la fermeture).
```

Conversion de la vidéo au format brut .h264 en un format standard .mp4

- Il faut installer l'outil de conversion "MP4Box" sur le Raspberry.

Pour cela, ouvrez un terminal et saisissez dedans :

```
sudo apt install -y gpac
```

- Ensuite se placer, toujours avec le terminal dans le dossier où se trouve la vidéo ma_video.h264 à convertir.

Puis saisir la commande :

```
MP4Box -add ma_video.h264 test.mp4
```

- Mais il se peut que la vidéo en mp4 ne se déroule pas à la bonne vitesse. Cela vient du fait que le mp4 se déroule à 25 images par seconde tandis que le h264 se déroule à 30 images par seconde. Pour corriger le problème, nous précisons quelques valeurs de paramètres :

In [ ]:
```
from picamera import PiCamera  #  Importation du module PiCamera de la bibliothèque pic
amera
from time import sleep  #  Importation de la fonction sleep de la bibliothèque time

camera = PiCamera()  #  Création d'un objet du type PiCamera nommé camera
camera.resolution = (640, 480)  #  Largeur x hauteur en pixels
camera.framerate = 25  #  25 images par seconde
try:
    camera.start_preview()
    camera.start_recording('/home/jdubois/Images/ma_video.h264')
    sleep(10)  #  10 secondes de pose (la durée de l'enregistrement).
    camera.stop_recording()
    camera.stop_preview()
finally:
    camera.close()  #  Dans la section finally on met les instructions indispensables à
notre programme (ici la fermeture).
```

Un exemple un peu moins simple : Faire clignoter une LED par un programme Python, à travers Internet.

D'après une idée de Tommy Desrochers, en s'inspirant du programme du livre p. 204

- **Rappel si vous n'avez pas fait la manipulation avec le relais décrite avant :**
  Pour pouvoir contrôler les ports GPIO depuis le Raspberry et depuis Internet comme c'est le cas ici, il est nécessaire que :

- L'utilisateur courant **jdubois** (par exemple) et **www-data** (qui n'est autre que le programme du serveur Apache) soient membres du groupe gpio. Pour ajouter jdubois et www-data au groupe gpio, saisissez :

```
<pre>grep -i gpio /etc/group
```

  En clair :

  group est un fichier situé dans le dossier /etc

  Ce fichier contient la liste des groupes d'utilisateurs du Rasbpberry.

  grep est un filtre qui n'affiche que la ligne gpio

  -i est un attribut qui précise que la casse (majuscule ou minuscule) n'a pas d'importance.</pre>

- Si le résultat de cette commande est :

```
<pre>gpio:x:997:</pre>
```

  alors le fichier php sera incapable d'agir sur les pins du GPIO et la commande de la LED ne marchera pas ni en utilisation directe sur le terminal, ni en se connectant au serveur Apache par le Web. Dans ce cas, saisissez :

```
<pre>sudo adduser jdubois gpio</pre>
<pre>sudo adduser www-data gpio</pre>
```

- Vérifiez maintenant que le groupe gpio possède deux utilisateurs de plus et que ce sont jdubois et www-data en saisissant :

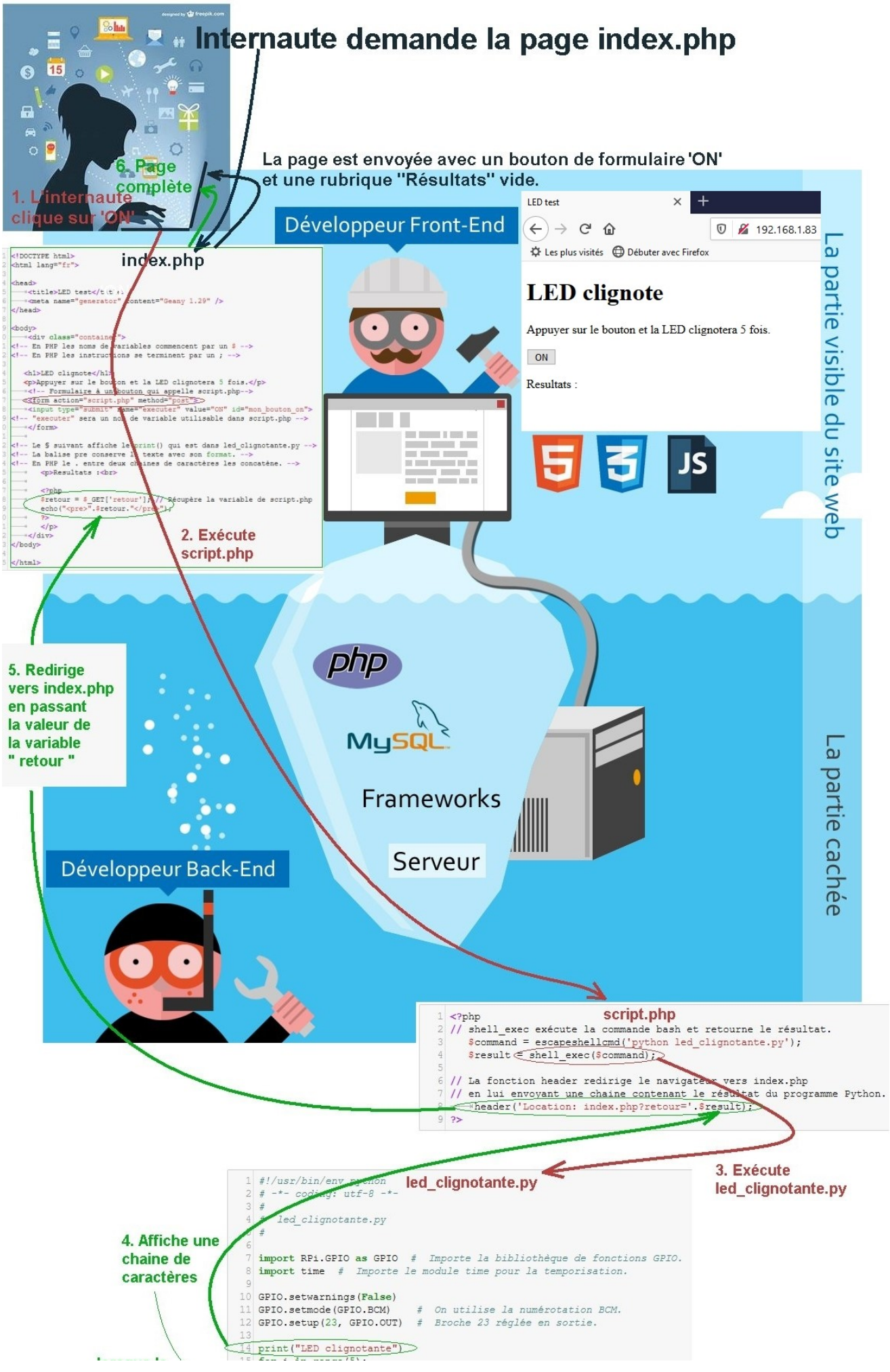
```
<pre>grep -i gpio /etc/group</pre>
```

  Le résultat de cette commande doit être :

```
<pre>gpio:x:997:jdubois,www-data</pre>
```

- **Il est indispensable de rebooter le Raspberry pour que cette modification prenne effet.**

- Comme dans le cas de la commande de relais, l'objet connecté est un Raspberry et la connexion est une **connexion HTTP**, mais cette fois, le fichier script.php n'agit pas directement au moyen d'une ligne de commande bash sur le Raspberry.
- script.php appelle le fichier led_cligontante.py en Python ce qui nous permettrait une programmation plus sophistiquée. Dans cet exemple nous n'irons pas plus loin que faire clignoter la LED 5 fois.
- Nous écrirons trois fichiers avec Geany, tous les trois placés dans le dossier Documents/webs puis copiés dans le dossier /var/www/html pour être visibles sur le web.
- Le fichier visible par l'internaute sera **index.php** et constitue donc le "front-end".
- Les fichiers côté serveur, invisibles par l'internaute, seront **led_clignotante.py** et **script.php**. Ces deux fichiers constituent le "back-end" de notre site.


Internaute demande la page index.php
1. L'internaute clique sur 'ON'
6. Page complète
La page est envoyée avec un bouton de formulaire 'ON' et une rubrique "Résultats" vide.
Développeur Front-End
index.php
LED test
192.168.1.83
Les plus visités
Débuter avec Firefox
LED clignote
Appuyer sur le bouton et la LED clignotera 5 fois.
ON
Resultats :
La partie visible du site web
2. Exécute script.php
5. Redirige vers index.php en passant la valeur de la variable " retour "
php
MySQL
Frameworks
Serveur
Développeur Back-End
La partie cachée
script.php
// shell_exec exécute la commande bash et retourne le résultat.
$command = escapeshellcmd('python led_clignotante.py');
$result = shell_exec($command);
// La fonction header redirige le navigateur vers index.php
// en lui envoyant une chaine contenant le résultat du programme Python.
header('Location: index.php?retour='.$result);
?>
3. Exécute led_clignotante.py
led_clignotante.py
#!/usr/bin/env python
-*- coding: utf-8 -*-
#
led_clignotante.py
#
import RPi.GPIO as GPIO # Importe la bibliothèque de fonctions GPIO.
import time # Importe le module time pour la temporisation.
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM) # On utilise la numérotation BCM.
GPIO.setup(23, GPIO.OUT) # Broche 23 réglée en sortie.
print("LED clignotante")
4. Affiche une chaine de caractères

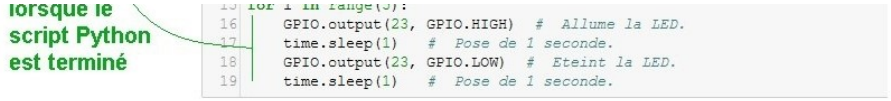


Schéma de la circulation de l'information
entre le côté front-end et le côté back-end

1. Ecriture du fichier led_clignotante.py

C'est le code ci-dessous très inspiré de celui du livre p. 204

Attention, il y a une coquille dans le livre : à la cinquième ligne c'est "GPIO.setup(23, GPIO.OUT)" au lieu de "GPIO.setmode(23, GPIO.OUT)".

In [ ]:
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
#  led_clignotante.py
#

import RPi.GPIO as GPIO  #  Importe la bibliothèque de fonctions GPIO.
import time  #  Importe le module time pour la temporisation.

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)     #  On utilise la numérotation BCM.
GPIO.setup(23, GPIO.OUT)   #  Broche 23 réglée en sortie.

print("LED clignotante")
for i in range(5):
    GPIO.output(23, GPIO.HIGH)  #  Allume la LED.
    time.sleep(1)    #  Pose de 1 seconde.
    GPIO.output(23, GPIO.LOW)  #  Eteint la LED.
    time.sleep(1)    #  Pose de 1 seconde.
```

Quand elle est appelée par la ligne de commande sur le terminal

```
python led_clignotante.py
```

cette fonction Python affiche "LED clignotante" sur le terminal et allume et éteint 5 fois de suite la LED branchée sur le port GPIO n°23 (numérotation BCM).

- Cependant, dans notre cas, elle ne sera pas appelée directement par l'utilisateur en ligne de commande bash.
- Elle sera appelée par le fichier script.php

2. Ecriture du script en PHP

Voici le code du fichier script.php. Rappel, en PHP les lignes de commentaires commencent par //

Et si on veut mettre plusieurs lignes en commentaire, on ouvre en mettant :

```
/*
```

et on ferme en mettant :

```
*/
```

In [ ]:
```php
<?php
// shell_exec exécute la commande bash et retourne le résultat.
   $command = escapeshellcmd('python led_clignotante.py');
   $result = shell_exec($command);

// La fonction header redirige le navigateur vers index.php
// en lui envoyant une chaine contenant le résultat du programme Python.
        header('Location: index.php?retour='.$result);
?>
```

Tout se passe comme si l'internaute tapait lui-même dans le terminal la commande

```
python led_clignotante.py
```

Cela lance l'exécution avec python du fichier led_clignotante.py

La LED clignotera 5 fois.

Mais la différence avec une exécution directement dans le terminal, est qu'ici l'affichage du print("LED clignotante") ne pourra pas être exécuté avant les 5 clignotements de la LED (puisqu'il n'y a pas de terminal).

Cette chaine de caractères "LED clignotante" sera passée à la variable PHP $result par la fonction shell_exec.

Donc la dernière ligne est équivalente à :

```
header('Location: index.php?retour=LED clignotante');
```

puisque la variable $result a été concaténée à l'aide du point "." qui est l'opérateur de concaténation en PHP.

3. Ecriture de la page front-end index.php

Cette page est codée principalement en HTML avec de courts passages en PHP :

In [ ]:
```
<!DOCTYPE html>
<html lang="fr">

<head>
        <title>LED test</title>
        <meta name="generator" content="Geany 1.29" />
</head>

<body>
        <div class="container">
<!-- En PHP les noms de variables commencent par un $ -->
<!-- En PHP les instructions se terminent par un ; -->

   <h1>LED clignote</h1>
   <p>Appuyer sur le bouton et la LED clignotera 5 fois.</p>
        <!-- Formulaire à un bouton qui appelle script.php-->
        <form action="script.php" method="post">
        <input type="submit" name="executer" value="ON" id="mon_bouton_on">
<!-- "executer" sera un nom de variable utilisable dans script.php -->
        </form>

<!-- Le $ suivant affiche le print() qui est dans led_clignotante.py -->
<!-- La balise pre conserve le texte avec son format. -->
<!-- En PHP le . entre deux chaines de caractères les concatène. -->
           <p>Resultats :<br>

           <?php
           $retour = $_GET['retour']; // Récupère la variable de script.php
           echo("<pre>".$retour."</pre>");
           ?>
           </p>
        </div>
</body>

</html>
```

- Fonctionnement de index.php :

Cette page front-end est donc celle que verra l'internaute.

En-dessous de son gros titre **LED clignote** et d'un paragraphe "Appuyer sur le bouton et la LED clignotera 5 fois", l'internaute verra un bouton de formulaire qui sera marqué "ON" (sans aucun style ici).

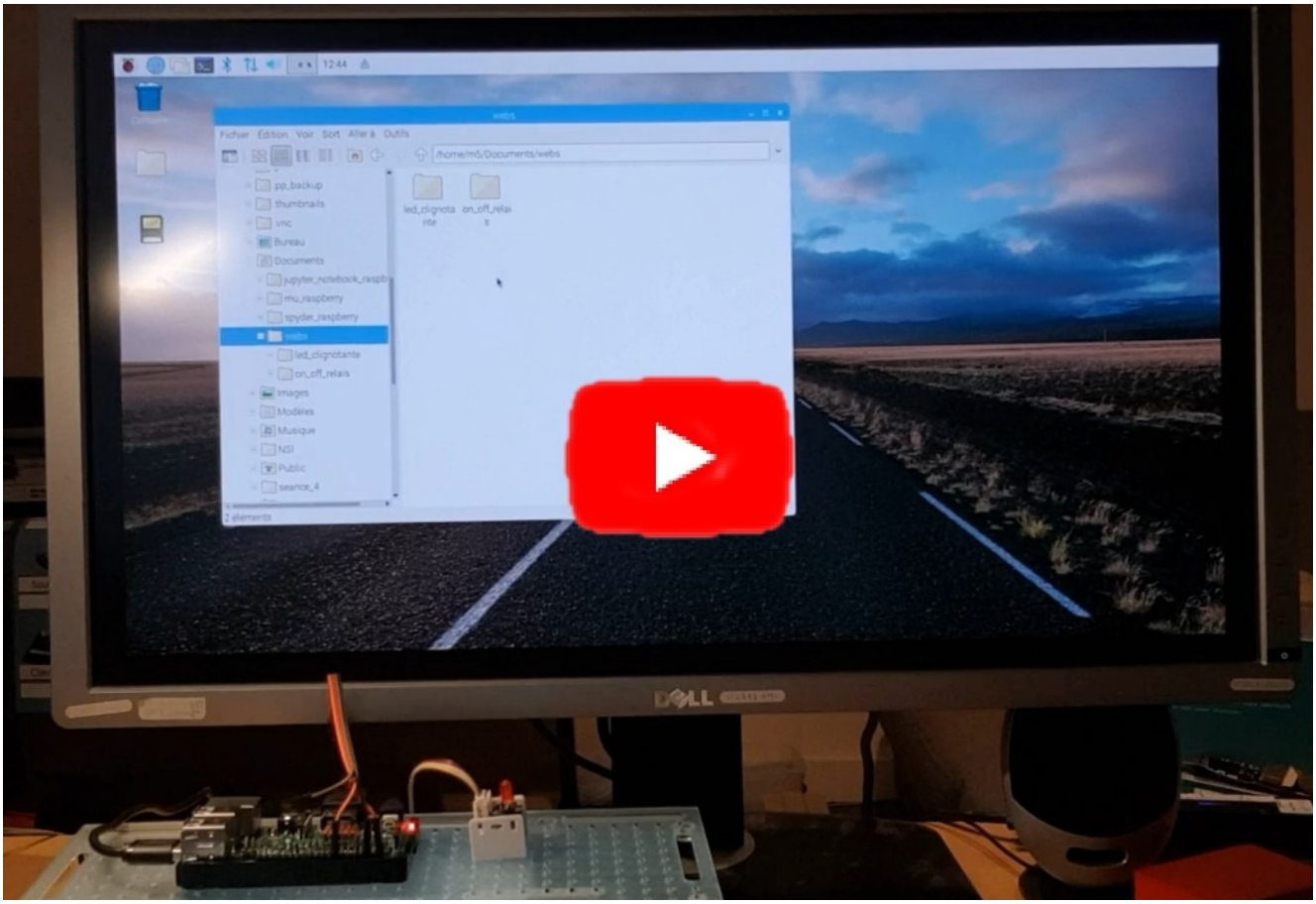
Quand il clique dessus, cela exécute le code PHP de la page script.php. Le code PHP de script.php lance le programme python led_clignotante.py par la commande "python led_clignotante.py".

Après les cinq clignotements de la LED, l'internaute n'a pas le temps de voir la page script.py s'afficher car celle-ci redirige aussitôt vers index.php.

Enfin le code PHP de index.php et cette fois, le petit bout de code PHP en bas fait s'afficher sur l'écran du navigateur le résultat qui est ce qu'a imprimé la fonction Python (c'est à dire pour nous le texte "LED clignotante").

4. Le montage

- Voyez la vidéo ci-dessous qui montre la réalisation du montage " Commande par HTTP du programme Python led_clignotante.py sur le Raspberry ".



(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_led_clignotante.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_led_clignotante.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_led_clignotante.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_led_clignotante.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_led_clignotante.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_led_clignotante.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_led_clignotante.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_led_clignotante.mp4)

Le projet d'équipe selon la méthode scrum (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/raspberry_pi_led_clignotante.mp4)

- Lorsqu'un projet complexe (c'est à dire comprenant plusieurs parties) doit être développé par plusieurs personnes, chacune "spécialiste" d'un domaine, il peut naitre des difficultés d'entente au sein de l'équipe.
- Pour palier ces difficultés et gagner en efficacité, l'équipe peut suivre un ensemble de pratiques réunies dans un schéma d'organisation.
- Au début des années 2000, un ensemble de pratiques de réalisation de projets a été rédigé dans ce qui est nommé le "manifeste Agile".
- L'une des méthodes Agile est *Scrum*. Scrum en anglais signifie "mêlée". Ce mot a été choisi en référence à l'état d'esprit nécessaire pour qu'une équipe de rugby gagne.
- *Scrum* est un schéma d'organisation qui peut se réveler très efficace lorsque, par exemple, il s'agit de faire travailler ensemble des développeurs back-end et des développeurs front-end pour le compte d'un client de l'entreprise et qu'il y a des dates d'échéances à respecter.
- Visionnez la vidéo ci-dessous qui décrit l'état d'esprit que doivent avoir, selon la méthode scrum, les membres d'une équipe de développement.
- Répondez ensuite à la question.

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/scrum.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/scrum.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/scrum.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/scrum.mp4)

13) Lorqu'une équipe de développeurs est composée de développeurs Back-End et développeurs Front-End organisée selon la méthode scrum, quel doit être l'état d'esprit des développeurs ?

Réponse :

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/scrum.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/scrum.mp4)
(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/scrum.mp4)

(http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/scrum.mp4)

2. Interface Homme-Machine (http://www.astrovirtuel.fr/jupyter/19_pnsi_cours/scrum.mp4)

Lisez le paragraphe **Interface Homme-Machine** du milieu de la p. 204 à la p. 205

14) Dans quel domaine s'est spécialisée le chercheuse en informatique Joëlle COUTAZ ?

Réponse :

15) Quelle a été la première interface homme-machine ?

Réponse :

16) Quelle entreprise américaine, spécialisée dans les photocopieurs, a été pionnière en interfaces utilisateurs ?

Réponse :

17) Jusqu'en 1983, quelle était la seule façon de communiquer avec les ordinateurs ?

Réponse :

18) Qu'est-ce qu'une GUI ?

Réponse :

19) Qu'est-ce qu'un NUI ? Citez un exemple.

Réponse :

20) Donnez la composition d'une IHM minimale.

Réponse :

21) Par quoi faut-il commencer lorsqu'on veut réaliser une IHM ?.

Réponse :

22) Citez deux façons de programmer une interface graphique pour réaliser une IHM sur un ordinateur, tablette ou smartphone.

Réponse :

Programmes en Python pour contrôler des actionneurs et recevoir des données des capteurs.

- Dans cette dernière partie, nous allons utiliser 3 actionneurs et 4 capteurs :

1. Une LED est un actionneur. Elle peut émettre une lumière.

2. Un buzzer est un actionneur. Il permet de jouer une note.

3. Un bouton poussoir est un capteur.

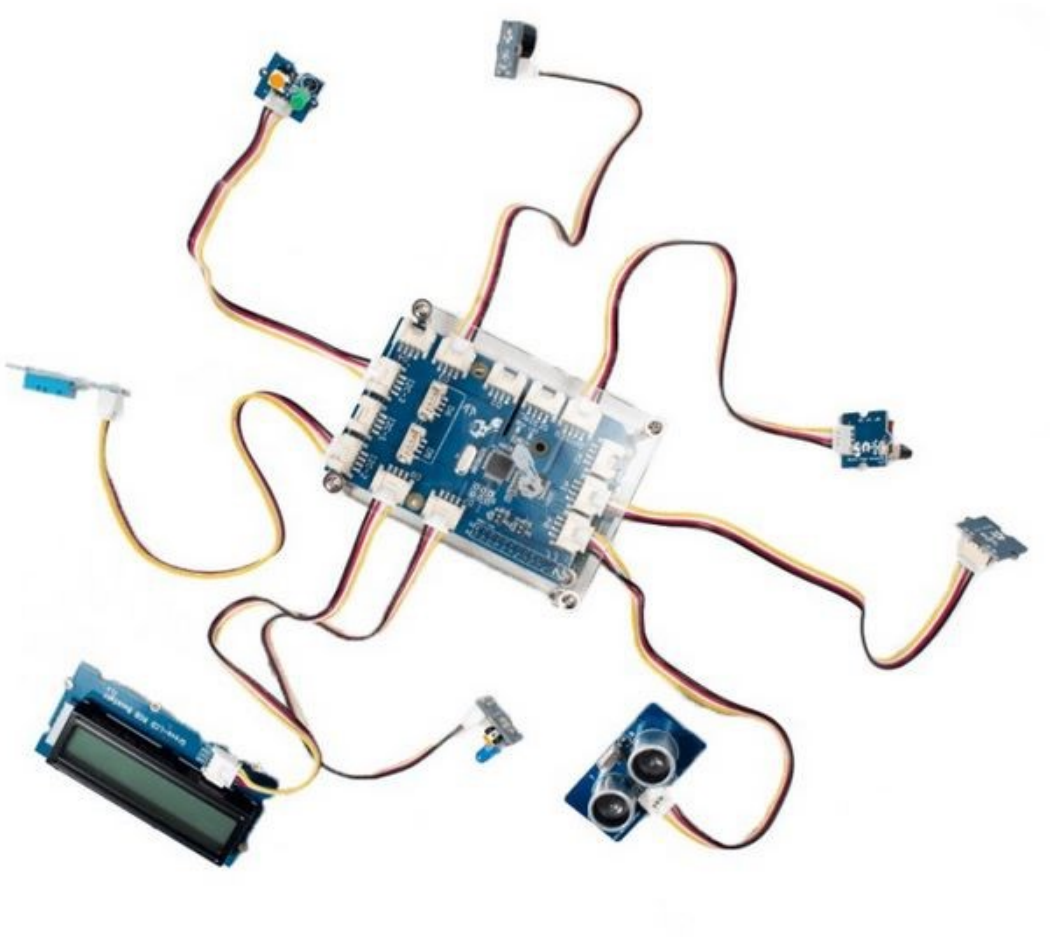
4. Une manette est un capteur. Elle permet d'appuyer sur une des 4 directions ou au milieu

5. Un écran à deux lignes est un actionneur. Il permet d'afficher un court texte.

6. Un capteur émetteur-récepteur à ultrasons permet de mesurer la distance.

7. Un servomoteur est un actionneur. C'est un moteur dont on peut orienter l'angle de manière précise.

- Ces capteurs et actionneurs peuvent être montés sur le Raspberry Pi au moyen d'un *shield* c'est à dire une carte additionnelle qui se monte sur le connecteur GPIO 2 x 20 broches.
- La connexion des capteurs et des actionneurs est rendue plus facile et plus sûre avec les connecteurs *Grove* (prises blanches).
- De plus une bibliothèque Python spéciale peut être installée sur le Raspberry. Elle pourra être importée dans les programmes de manière à bénéficier des fonctions et méthodes liées aux différents actionneurs et capteurs.



Rasberry Pi équipé de son shield GrovePi+ et de sept actionneurs et capteurs

Parcours 1 sur le concours du castor informatique : des objets qui réagissent

1. Nous allons utiliser un simulateur en ligne pour simuler le comportement de sept capteurs et actionneurs.
2. Téléchargez (http://www.astrovirtuel.fr/nsi/premiere/premiere2019-2020/mode_d_emploi_parcours1.pdf) le guide du parcours 1 et lisez-le.
3. Ensuite demandez à votre professeur un code.
4. Pendant le parcours, vous complèterez vos réponses sur le site. Mais pour en garder la trace, chaque fois que vous avez franchi un niveau, vous recopierez votre code qui fonctionne ci-dessous.
5. Commencez le parcours 1 : des objets qui réagissent sur https://concours.castor-informatique.fr/ (https://concours.castor-informatique.fr/)

Mélodie :

Lisez la présentation du buzzer et de la gestion du temps.

Écrire un programme qui active le buzzer pour jouer un son.

In [ ]:
```
from quickpi import *
```

Écrire un programme qui joue la note "la", à la fréquence 440Hz, pendant une seconde.

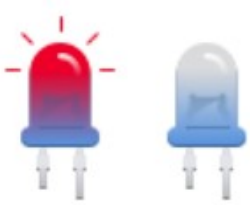
In [ ]:
```
from quickpi import *
```

Écrire un programme qui joue la mélodie “do ré mi ré do”, en jouant chaque note pendant 500ms, l'une après l'autre, puis éteint le buzzer.

Les fréquences des notes sont : Do : 523Hz, Ré : 587Hz, et Mi : 659Hz.

In [ ]:
```
from quickpi import *
```

Alternance



Lisez la présentation de la boucle de répétition avec l'instruction `for` , des LEDs et de la gestion du temps.

Écrire un programme qui fait clignoter la LED cinq fois : c'est à dire l'allume pendant 1s, puis l'éteint pendant 1s, puis recommence quatre autres fois.

In [ ]:
```
from quickpi import *
```

Écrire un programme qui allume en alternance les LED rouge et bleue pendant 500ms chacune : la rouge pendant 500ms, la bleue pendant 500ms, etc.

Chacune des LEDs doit être allumée 5 fois au total.

Tout doit être éteint à la fin.

In [ ]:
```
from quickpi import *
```

Écrire un programme qui allume en alternance les LED rouge et bleue 5 fois au total, chaque fois pendant 500ms mais allumant la suivante 100ms avant d'éteindre la précédente : on allume la rouge au tout début, puis allume la bleue au temps 400ms, puis éteint la rouge au temps 500ms, puis allume la rouge au temps 800ms, etc.

In [ ]:
```
from quickpi import *
```

Show lumineux 1

Lisez la présentation de la boucle de répétition avec l'instruction `for` , des LEDs et de la gestion du temps.

Écrire un programme qui dans l'ordre :

- Allume la LED rouge pendant 1s
- Fait clignoter la LED bleue 5 fois : 500ms allumée puis 500ms éteinte.
- Allume la LED verte et la laisse allumée.

In [ ]:
```
from quickpi import *
```

Écrire un programme qui dans l'ordre :

- Allume la LED rouge pendant 1s
- Fait clignoter la LED verte 5 fois : 500ms allumée puis 500ms éteinte.
- Fait clignoter la LED bleue 3 fois : 500ms allumée puis 500ms éteinte.
- Rallume la LED verte et la laisse allumée.

In [ ]:
```
from quickpi import *
```

Écrire un programme qui joue la séquence suivante, où chaque tiret représente un état allumé de 500ms, et chaque point un état éteint de 500ms.

Notez que les deux lignes se jouent en même temps :

```
- LED rouge : -.-.-.--.-.-.
- LED verte : ......------
```

Les LEDs doivent être éteintes à la fin du programme.

In [ ]:
```
from quickpi import *
```

Quelle direction ?

Lisez la présentation de : la boucle infinie, l'instruction if, l'écran, le bouton poussoir et la manette.

Écrire un programme qui laisse l'écran vide au début, puis affiche le texte "Bonjour" dès que l'on appuie sur le bouton, et le laisse affiché.

Votre programme devra boucler indéfiniment, et tester en permanence si le bouton est enfoncé. Notez qu'un texte affiché à l'écran y reste jusqu'à ce que l'on affiche autre chose.

In [ ]:
```
from quickpi import *
```

Écrire un programme qui :

- affiche le texte "Appuyez"
- affiche le texte "Merci" dès que l'on appuie sur le bouton, et le laisse affiché.

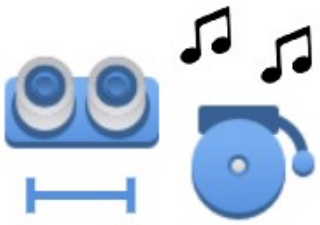
In [ ]:
```
from quickpi import *
```

Écrire un programme qui :

- affiche le texte "Direction ?" dès le début
- puis lorsque l'on appuie sur une direction, affiche le texte "Haut", "Droite", "Bas" ou "Gauche" selon la direction appuyée.

In [ ]:
```
from quickpi import *
```

Instrument



Lisez la présentation de la boucle infinie, de l'instruction if, du capteur de distance et du buzzer.

Écrire un programme qui allume le buzzer quand le capteur détecte un objet à moins de 100cm, et arrête le buzzer dès qu'il n'en détecte plus.

In [ ]:
```
from quickpi import *
```

Écrire un programme qui allume le buzzer quand le capteur détecte un objet à moins de 500cm.

Le buzzer doit jouer un son à une fréquence égale à la distance de l'objet en centimètres. Il doit être éteint s'il n'y a pas d'objet à moins de 500cm.

In [ ]:
```
from quickpi import *
```

Modifier le programme de la version précédente, avec la différence suivante : lorsque le bouton est enfoncé, la fréquence du buzzer doit être doublée.

In [ ]:
```
from quickpi import *
```

Show Lumineux 2 :

Lisez la présentation des boucles imbriquées avec l'instruction for, des LEDs et de la gestion du temps.

Écrire un programme qui fait 3 fois ces deux étapes :

- Faire clignoter la LED rouge 3 fois (200ms allumée puis 200ms éteinte)
- Attendre 1 seconde.

In [ ]:
```
from quickpi import *
```

Écrire un programme qui fait 3 fois ces deux étapes :

- Clignoter la LED rouge 3 fois (200ms allumée puis 200ms éteinte)
- Clignoter la LED verte 3 fois (200ms allumée puis 200ms éteinte)
- Clignoter la LED bleue 3 fois (200ms allumée puis 200ms éteinte)
- Attendre 1 seconde.

In [ ]:
```
from quickpi import *
```

Écrire un programme qui fait en même temps, et pendant 4 secondes au total :

- Clignoter la LED rouge : 1s allumée puis 1s éteinte
- Clignoter la LED verte : 500ms allumée puis 500ms éteinte
- Clignoter la LED bleue : 250ms allumée puis 250ms éteinte

In [ ]:
```
from quickpi import *
```

Avertisseur :

Lisez la présentation de la boucle infinie, de l'instruction if/else, de la boucle de répétition avec l'instruction `for` , du capteur de distance, du buzzer, et de la gestion du temps.

Écrire un programme qui, dès qu'un objet passe à moins de 30cm du capteur, fait dans l'ordre :

- Jouer 3 bips de 100ms espacés de 100ms
- Attendre 500ms

In [ ]:
```
from quickpi import *
```

Modifiez le programme de la version précédente, pour qu'il arrête de jouer des bips dès qu'il n'y a plus d'objet.

S'il y a un bip en cours, il se termine, mais on ne joue pas le suivant.

In [ ]:
```
from quickpi import *
```

Modifiez le programme de la version précédente, pour que lorsqu'un objet passe à moins de 10cm du capteur alors qu'une série de bips n'est pas en cours, alors il joue un bip en continu.

Au dessus de cette distance, il doit faire comme dans la version précédente.

In [ ]:
```
from quickpi import *
```

Servo chronométré :

Lisez la présentation du servomoteur, de la boucle infinie, de l'instruction if/else, de la boucle de répétition avec l'instruction `for` , de la manette, du buzzer, des LEDs, et de la gestion du temps.

Écrire un programme qui met l'angle du servomoteur à 0°, puis lorsque l'on appuie sur le bouton :

- Augmente l'angle 18 fois de 10°, toutes les 50ms
- Joue un bip pendant 500ms
- Remet l'angle à 0°

In [ ]:
```
from quickpi import *
```

Écrire un programme qui met le servomoteur à 10°, puis lorsque que le stick est appuyé :

- À gauche : diminue de 2° puis attend 50ms
- À droite : augmente de 2° puis attend 50ms

Le programme doit ignorer les actions qui ne gardent pas l'angle entre 10° et 170°.

In [ ]:
```
from quickpi import *
```

Modifier le programme de la version précédente, pour qu'il arrête au bout de 5s, puis selon l'angle du servo :

- S'il vaut 90°, allume la LED verte
- S'il est inférieur à 90°, allume la LED rouge
- S'il est supérieur à 90°, allume la LED bleue

Le programme doit ensuite attendre 2s, puis éteindre les LEDs et tout recommencer.

In [ ]:
```
from quickpi import *
```