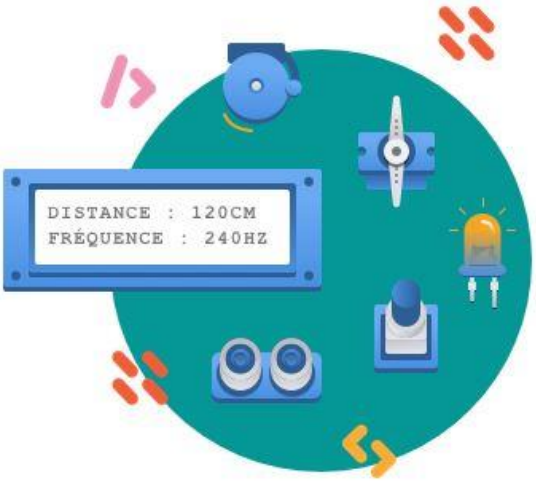
Présentation du parcours 1 : des objets qui réagissent



Ce parcours a pour objectif de vous apprendre à programmer des **objets connectés**. Il s'agit de petits objets électroniques programmables, capables d'allumer des petites lumières, d'émettre du son, de mesurer des distances, de faire tourner des moteurs, de communiquer des informations, etc.

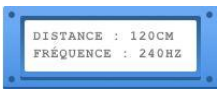
<————— 6 cm —————>

La photo ci-contre montre un tel objet connecté, constitué d'un tout petit ordinateur, appelé *Raspberry Pi zéro*, assemblé avec une carte contenant plusieurs *capteurs* et *actionneurs*.

1 Simulation d'un objet connecté

Pour vous permettre d'apprendre à programmer des objets connectés sans avoir besoin d'acheter du matériel, nous utilisons un **simulateur**.

Ce simulateur représente les différents composants (capteurs et actionneurs) sur votre écran :

	1. Une **LED** est un actionneur. Elle peut émettre une lumière. On aura une LED rouge, une verte, et une bleue.
	2. Un **buzzer** est un actionneur. Il permet de jouer une note : il émet un son à la fréquence désirée.
	3. Un **bouton poussoir** est un capteur. Il permet de déclencher une action lorsqu'on appuie dessus.
	4. Une **manette** est un capteur. Elle permet d'appuyer sur une des 4 directions, ou au milieu.
	5. Un **écran** est un actionneur. Il permet d'afficher un court texte, ou bien des images en noir et blanc.
	6. Un **capteur de distance** permet de mesurer la distance entre l'objet connecté et une cible (avec des ultrasons ou bien un
	7. Un **servomoteur** est un actionneur. C'est un moteur dont on peut orienter l'angle de manière précise.

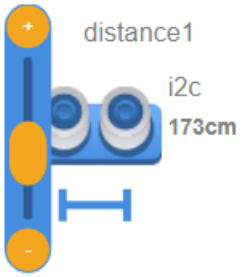
2 Interagir avec le simulateur

- Le simulateur vous permet d'interagir avec les composants. Vous pouvez ainsi modifier les valeurs lues par les capteurs pendant que votre programme s'exécute, et voir comment votre programme réagit.

Par exemple, vous pouvez :

  - Cliquer sur un bouton pour l'enfoncer,
  - Cliquer sur la manette pour appuyer sur une des directions,
  - Voir les LED s'allumer,
  - Lire le texte affiché sur le petit écran,
  - Observer la rotation du servomoteur, etc...

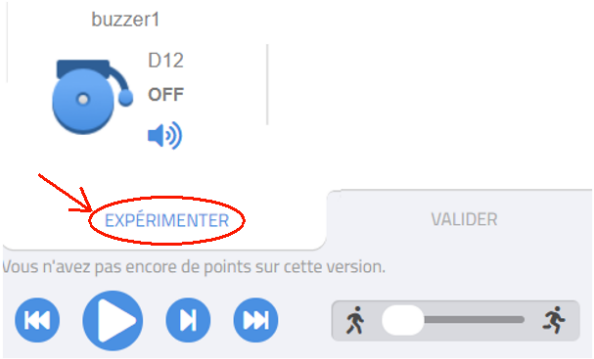
- Pour le capteur de distance, cliquez dessus et faites glisser le curseur pour régler la distance souhaitée entre le capteur et sa cible.



3 Expérimenter et valider votre programme

- Pour chaque défi :

1. **Écrivez le code** de votre programme.
2. **Testez votre programme dans l'onglet "Expérimenter"**. Exécutez votre programme en interagissant comme vous le souhaitez avec les capteurs.
3. **Validez votre programme dans l'onglet "Valider"**. **Exécutez** votre programme. Un test automatisé est alors utilisé pour déterminer si vous avez réussi le défi.



- Boutons de contrôle :

	**Exécuter** votre programme.
	**Arrêter** l'exécution et remettre à zéro.
	Exécuter les instructions une par une.
	Exécuter le programme en mode accéléré.
	Régler la vitesse d'exécution.

4 Aide, menu

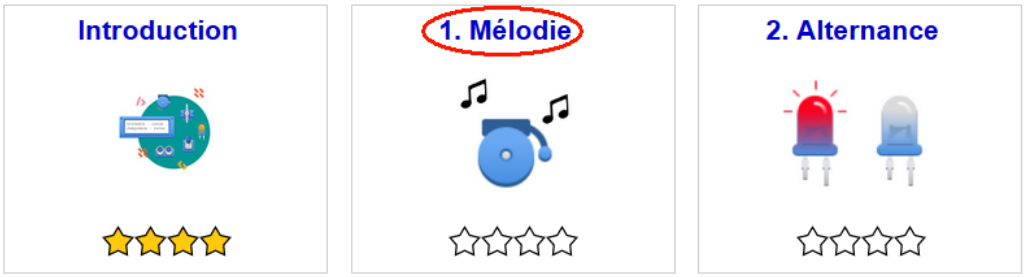
- En haut à droite de l'éditeur, vous trouverez trois boutons utiles :

	Un bouton qui ouvre un menu avec des outils, par ex. copier/coller une partie de programme.
	Un bouton pour passer l'éditeur en mode plein écran, très utile si vous avez un petit écran.
	Un bouton pour ouvrir l'aide sur les capteurs et concepts utilisés dans le défi courant.

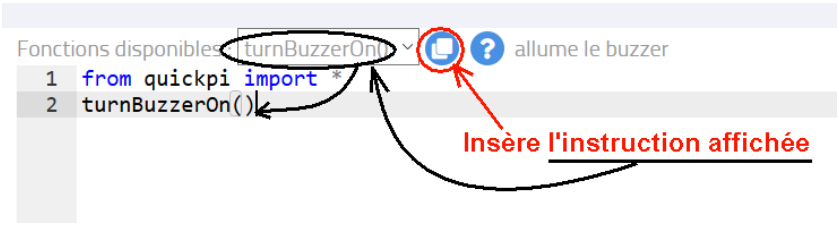
5 À vous de jouer !

- Cliquez sur le bouton "Retour à la liste des questions" tout en haut de cette page

- Cliquez sur le sur le défi numéro 1, niveau 2 étoiles.



- Choisissez l'instruction puis cliquez sur le bouton Insère.



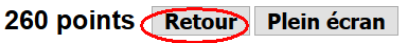
- 1. Dans l'onglet "Expérimenter",
- 2. appuyez sur la flèche qui permet d'exécuter le programme.
- 3. Puis, si le programme fonctionne, allez dans l'onglet "Valider". Et à nouveau cliquez sur la flèche qui permet d'exécuter le programme.

6 Quel parcours faire ?

- Il est conseillé de suivre la progression :

1. **Mélodie** niveaux 2 étoiles, 3 étoiles, 4 étoiles.
2. **Alternance** niveaux 2 étoiles, 3 étoiles, 4 étoiles.
3. **Show lumineux 1** niveaux 2 étoiles, 3 étoiles, 4 étoiles.
4. **Quelle direction ?** niveaux 2 étoiles, 3 étoiles, 4 étoiles.
5. **Instrument** niveaux 2 étoiles, 3 étoiles, 4 étoiles.
6. **Show lumineux 2** niveaux 2 étoiles, 3 étoiles, 4 étoiles.
7. **Avertisseur** niveaux 2 étoiles, 3 étoiles, 4 étoiles.
8. **Servo chronométré** niveaux 2 étoiles, 3 étoiles, 4 étoiles.

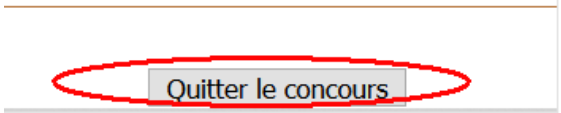
A tout moment vous pouvez revenir en arrière en cliquant sur "Retour"



Pour quitter le concours :

1) Cliquez sur le bouton "Retour"

2) Tout en bas cliquez sur "Quitter le concours"



Pour reprendre :

https://concours.castor-informatique.fr/

et saisir le code qui vous a été donné lorsque vous avez commencé le concours.

7 Documentation :

1 LEDs ou diodes électroluminescentes

Une LED est un composant qui émet de la lumière quand il est parcouru par un courant électrique.
Une LED ne laisse passer le courant électrique que dans un seul sens.

On trouve des LEDs qui émettent de la lumière rouge ou de la lumière verte, ou d'autres couleurs encore.

Fonctions disponibles

Les fonctions `turnLedOn()` et `turnLedOff()` permettent respectivement d'allumer et d'éteindre une LED.
Elles ne peuvent servir que lorsqu'il n'y a qu'une seule LED utilisée.

Une LED possède deux états :

ON : le courant traverse la LED, elle est allumée.

OFF : il n'y a pas de courant, la LED est éteinte.

`setLedState(led,state)`	Cette fonction permet d'allumer ou éteindre une LED. Elle prend en paramètre le nom de la LED et l'état à considérer, True pour l'allumer, False pour l'éteindre. Exemple : `setLedState("led1", True)`
`toggleLedState(led)`	Cette fonction permet d'inverser l'état de la LED entrée en paramètre sous forme de chaine de caractères. Exemple : `toggleLedState("led1")`

2 Buzzer

Un buzzer est un composant qui produit un son lorsqu'il est soumis à un tension électrique.
Le son peut être toujours le même ou être paramétrable.

Un buzzer possède deux états :

ON : le buzzer est soumis à une tension électrique, il sonne.

OFF : sans tension électrique, le buzzer reste silencieux.

Fonctions disponibles

`turnBuzzerOn()` Cette fonction permet d'allumer le buzzer.

`turnBuzzerOff()` Cette fonction permet d'éteindre le buzzer.

`setBuzzerNote(buzzer, frequency)` Pour le buzzer entré en paramètre, cette fonction permet de produire un son à une fréquence donnée.
La fréquence est exprimée en Hertz.
Exemple :

```
setBuzzerNote("buzzer1", 262)
```

permet de jouer la note DO.

Correspondance entre les notes de musique et les fréquences

Oct.	-1	0	1	2	3	4	5	6	7	8	9
DO	16,4	32,7	64	132	262	523	1046	2093	4186	8372	16744
DO#	17,3	34,7	69	138	277	554	1109	2217	4435	8870	17740
RE	18,4	36,7	74	147	294	588	1175	2349	4699	9397	18794
RE#	19,4	38,9	78	155	311	622	1244	2489	4978	9956	19912
MI	20,6	41,2	82	165	330	659	1318	2637	5274	10546	21092
FA	21,8	43,7	88	175	349	698	1397	2793	5588	11175	
FA#	23,1	46,2	92	185	370	740	1480	2960	5920	11840	
SOL	24,5	49	98	196	392	784	1568	3136	6272	12544	
SOL#	26	51,9	104	208	415	831	1661	3222	6645	13290	
LA	27,5	55	110	220	**440**	880	1760	3520	7040	14080	
LA#	29,1	58,3	116	233	466	932	1865	3729	7459	14917	
SI	30,9	61,7	123	247	494	988	1976	3951	7902	15804	

Gestion du temps

`sleep(milliseconds)` Cette fonction permet de stopper l'exécution du programme pendant une durée entrée en paramètre. Cette durée est exprimée en **millisecondes**.

3 Bouton poussoir

Un bouton poussoir est un élément qui possède deux états, relevé et enfoncé.

ON : le bouton est enfoncé.

OFF : le bouton est relevé.

Fonctions disponibles

`isButtonPressed()`

Cette fonction renvoie **True** si le bouton est enfoncé, et **False** s'il est relevé.
Cette fonction est utilisée seulement lorsqu'il n'y a qu'un seul bouton pressoir sur le montage.

`isButtonPressed(button)`

Pour le bouton entré en paramètre sous forme de chaine de caractères, cette fonction renvoie **True** si le bouton est enfoncé, et **False** s'il est relevé.

Exemple :

```
isButtonPressed("button1")
```

4 Manette

Une manette (stick en anglais) est un ensemble de 5 boutons, chacun correspondant à une direction : haut, droite, bas, gauche ou bien centre.

Il s'agit d'un seul composant, mais qui se programme comme 5 boutons différents :

- "stick1.up" pour la direction haut de la manette "stick1"
- "stick1.right" pour la direction droite
- "stick1.down" pour la direction bas
- "stick1.left" pour la direction gauche
- "stick1.center" pour la direction centre

Par exemple pour tester si la direction haut est enfoncée, on utilise :

```
buttonState("stick1.up")
```

5 Écran

L'écran de ce module est un petit écran qui permet d'afficher deux lignes de 16 caractères.

Fonctions disponibles

`displayText(screen, line1, line2)`

Cette fonction permet d'afficher deux lignes de texte sur un écran.
Elle prend en paramètre l'écran à considérer, ainsi qu'une ou deux lignes à afficher, sous forme de chaines de caractères.

Exemple :

avec le code

```
displayText("screen1", "Hello", "World !")
```

on affiche *Hello* sur la première ligne, et *World !* sur la deuxième ligne de l'écran *screen1*.

6 Capteur de distance

Ce capteur permet de mesurer la distance sans contact grâce à un capteur à ultrasons ou à laser. Il a une portée de 3 centimètres à 5 mètres.

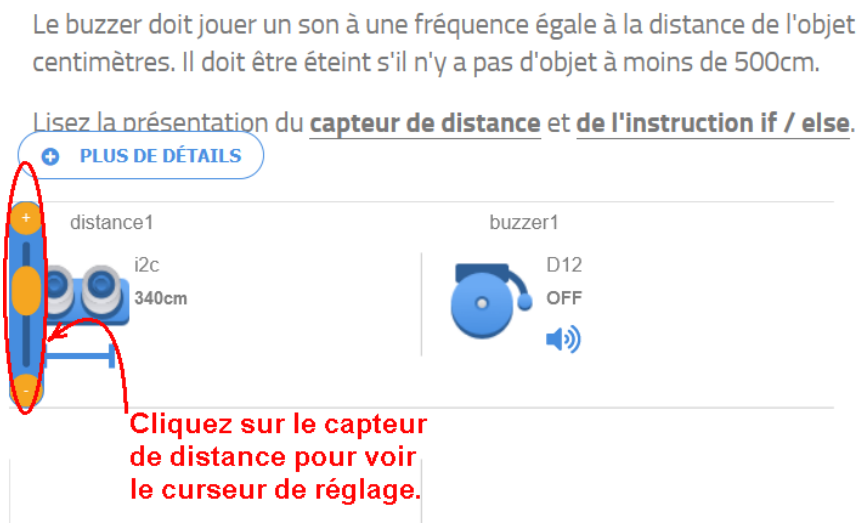
Fonctions disponibles

`readDistance(range)`

Cette fonction renvoie la distance captée par le capteur de distance entré en paramètre.
Cette distance est exprimée en centimètres.

Exemple :

```
readDistance("range1")
```



7 Servomoteur

Le Servomoteur est un petit moteur qui peut tourner précisément jusqu'à un angle donné, entre 0 et 180 degrés. On peut l'utiliser pour contrôler la direction des roues d'un petit véhicule, ou pour ouvrir ou fermer une barrière, etc.

Fonctions disponibles

`setServoAngle(servo, angle)`

Cette fonction permet de modifier l'angle du servomoteur choisi. L'angle est exprimé en degrés, entre 0 et 180 degrés.

Exemple :

```
setServoAngle("servo1", 90)
```

`getServoAngle(servo)`

Ce bloc permet de relire l'angle auquel on a réglé le servomoteur choisi. Ce n'est pas un capteur, mais simplement une mémorisation de la dernière valeur modifiée par une instruction.

On peut par exemple l'utiliser pour augmenter l'angle de 1 degré.

Exemple :

```
setServoAngle("servo1", getServoAngle("servo1") + 1)
```