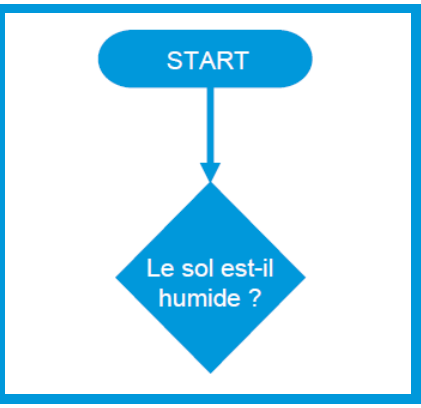
Belgium esero

Fiche pour les élèves

Des plantes sur Mars

Construction d'un système d'arrosage automatique



Les élèves élaborent et programment un système d'arrosage automatique qui mesure l'humidité du sol et actionne un dispositif d'arrosage en conséquence, en utilisant un microcontrôleur Pyboard et le langage Python. Les bases de la programmation en Python sont introduites à l'aide de l'environnement Thonny. Adaptation de la ressource T09 *Un potager sur Mars.*

esa belspo La Scientothèque L'ÉGALITÉ DES CHANCES PAR LES SCIENCES ULB

Des plantes sur Mars

Construction d'un système d'arrosage automatique

Table des matières

BIENVENUE SUR MARS

Introduction

Mars est la quatrième planète à partir du Soleil et la deuxième plus proche voisine de la Terre, après Vénus. La distance minimale entre la Terre et Mars est de 55 millions de kilomètres, alors que la distance entre la Terre et la Lune est relativement courte (380 000 kilomètres). La distance maximale entre la Terre et Mars est d'environ 400 millions de km. Une telle variation de la distance augmente considérablement la complexité de toute mission vers Mars, car il est beaucoup plus coûteux et plus difficile d'envoyer des fournitures.

Une solution possible est que les astronautes emportent des graines de plantes avec eux. Cela permettrait aux astronautes de faire pousser les graines à leur arrivée et de créer une source de nourriture autosuffisante.
Cependant, ce n'est pas une tâche facile. Il existe de nombreux facteurs qui créent un environnement dangereux pour les plantes sur Mars.

Exercices

1. Pour commencer à y réfléchir plus en détail, énumérez quelques-unes des choses dont les plantes et autres organismes vivants ont besoin pour survivre.

2. Discutez avec vos camarades de classe et votre professeur des réponses que vous pensez avoir aux questions suivantes sur la Terre.
    - Quelles sont les causes des saisons sur la Terre ?
    - Quelle est la forme de l'orbite de la Terre autour du Soleil ?
    - Quels sont les principaux éléments présents dans l'atmosphère terrestre ?
    - Qu'est-ce que la zone de la Boucle d'or et la Terre se trouve-t-elle à l'intérieur de cette zone ?

3. Décidez si les affirmations suivantes sont vraies ou fausses.

Déclarations	Vrai ou Faux
Mars connaît des saisons, tout comme la Terre.	
L'orbite de Mars a une forme similaire à celle de la Terre, ce qui signifie que la température à la surface est assez constante.	
L'atmosphère de Mars est épaisse et retient la chaleur du Soleil.	
Mars n'a pas de champ magnétique, ce qui signifie qu'il y a moins de protection contre les rayons UV nocifs et les vents solaires.	
Nous avons trouvé de l'eau liquide à la surface de Mars.	
L'atmosphère de Mars a une composition similaire à celle de la Terre.	
Les plantes de Mars devraient s'adapter aux cycles diurnes et nocturnes très différents sur Mars.	
Mars n'existe pas à l'intérieur de la zone "Boucle d'or" (habitable), il est donc impossible que de l'eau liquide existe à la surface.	

Dans les activités suivantes, nous agirons comme des explorateurs spatiaux en mission vers Mars afin d'établir un avant-poste martien. Pour augmenter les chances de succès de la mission, nous construirons un système d'arrosage automatique des plantes. Nous allons expérimenter et concevoir un prototype ici sur Terre, afin de pouvoir l'adapter plus tard à l'environnement martien !

PRÉPARATION DES COMPOSANTS, CONCEPTION ET PREMIERS TESTS

Pour qu'une mission vers Mars soit réussie, les astronautes devront être aussi autonomes que possible. Cela inclut le recyclage d'un maximum de leurs ressources et la culture de leur propre nourriture.

Les plantes sont une ressource précieuse. Les légumes sont une source d'aliments à forte densité nutritionnelle qui peuvent être cultivés à partir de petites graines et de bulbes, ce qui limite la quantité de matériel transporté à bord du vaisseau spatial. La photosynthèse, un processus effectué dans les plantes pour produire du glucose pour la croissance et la respiration, nécessite du dioxyde de carbone, dont il y a une abondance dans l'atmosphère martienne. Cependant, les plantes nécessitent une surveillance constante si elles veulent produire de bonnes récoltes, surtout si leur environnement n'est pas naturellement riche en ressources dont elles ont besoin.

Le maintien d'un écosystème sur Mars pourrait donc nécessiter de nombreuses heures et prendre une grande partie du temps des astronautes. Notre tâche est de commencer à développer un système qui permettrait à un ordinateur de surveiller à distance le bien-être d'une plante et de prendre des décisions en conséquence. Cela donnerait aux astronautes une plus grande liberté pour effectuer d'autres tâches.

Exercice

Dans l’espace ci-dessous, dessinez et annotez votre premier plan du système d’arrosage.

Nous avons choisi un ensemble spécifique de composants pour développer un éventuel système. Combiné à votre connaissance du fonctionnement des différents composants et à la liste de matériel ci-dessous, votre tâche consiste à concevoir un système qui peut être utilisé pour arroser automatiquement une plante en fonction du niveau d'humidité du sol.

Matériel

- Pyboard
- Ordinateur portable + câble micro-USB
- Capteur d'humidité du sol
- Servo-moteur (mini 3-5 V)
- Breadboard
- Un seau
- Tube (un tube d'irrigation fin est parfait)
- Sol/plante
- Câbles Dupont
- Ciseaux/couteaux d'artisanat
- Bouteille vide
- Pâte adhésive/papier collant
- Attaches de câble

Dans la liste ci-dessus figure un servo. Un servo est un petit moteur qui, lorsqu'il est en position fixe, peut être utilisé pour faire tourner une hélice.
Pensons d'abord à l'aspect physique du système - ne vous inquiétez pas pour les connexions électriques spécifiques pour le moment !

Concevez et testez votre réservoir d'eau

Nous avons maintenant une idée de base sur la façon dont notre système d'arrosage des plantes pourrait fonctionner. L'étape suivante consiste à affiner votre conception par des tests ! Dans cet exercice, vous serez guidé dans la construction d'un modèle spécifique. Si votre conception est différente, vous devrez adapter les étapes ou trouver vos propres idées !

Commençons par la conception proprement dite du système. Pour cette étape, vous aurez besoin de :

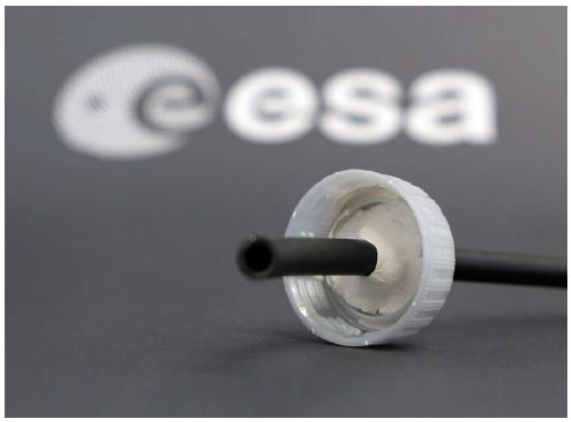
- La bouteille d'eau
- Pâte adhésive
- Ciseaux
- Un tube d'irrigation
- Un seau

Une grande et large bouteille constitue un réservoir parfait pour notre système d'arrosage. Veillez à garder le couvercle et à couper la partie inférieure de la bouteille pour pouvoir continuer à remplir le réservoir.

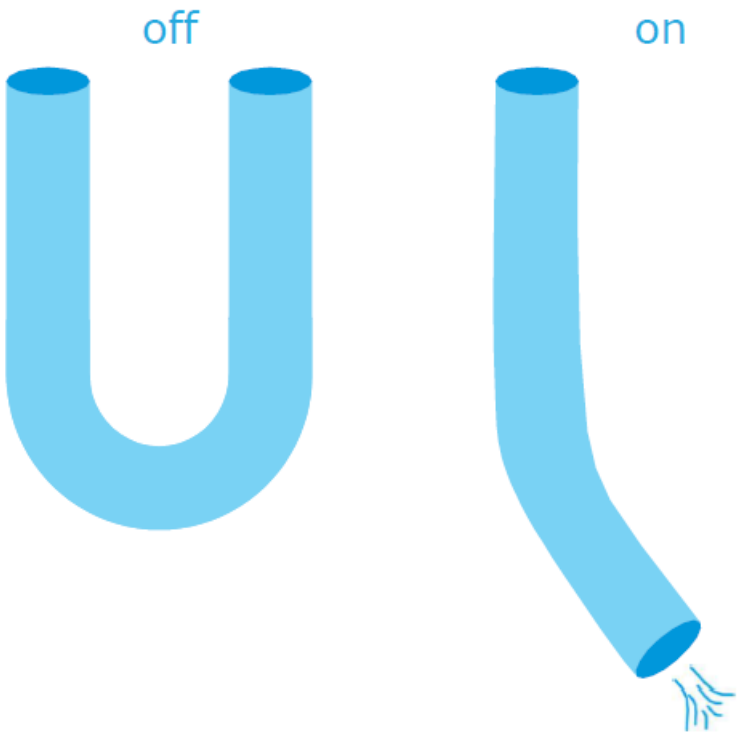
Ensuite, nous devons créer un trou dans le bouchon pour notre tuyau d'arrosage. Cette étape doit être réalisée par un enseignant ! Le mieux est de tailler lentement et soigneusement un trou à la taille requise avec une paire de ciseaux. Testez avec la pipe à eau jusqu'à ce que vous soyez satisfait de l'ajustement. Plus l'ajustement est serré, mieux c'est. La longueur du tube dont vous avez besoin peut varier en fonction de votre espace de travail. Mais n'oubliez pas qu'il n'y a pas de pompe dans ce système, nous comptons donc sur la gravité pour permettre et arrêter l'écoulement de l'eau. Gardez cela à l'esprit !

Nous espérons que vous avez réussi à créer un bon ajustement pour le tube, mais il est probable qu'il n'est pas parfaitement étanche. Ce problème peut être facilement résolu avec

du blu-tack - un pistolet à colle peut fournir une meilleure étanchéité, si vous voulez créer une installation plus permanente, mais ce n'est pas nécessaire pour notre projet.



Commençons maintenant à réfléchir à la façon dont notre système va fonctionner. Nous devons établir une position "marche" et "arrêt" - quand le tube irriguera, et quand il ne le fera pas. Une configuration intuitive consistera à faire pointer l'extrémité du tube vers le haut en position « off » et vers le bas en position « on ». Pour l'instant, il suffit de savoir que notre servomoteur nous aidera en la matière.



Nous avons maintenant une assez bonne idée de ce à quoi notre système ressemblera, mais pour le moment, il nécessite notre contribution et notre intervention pour qu'il fonctionne. Notre objectif est d'automatiser ce système, afin que les astronautes puissent utiliser leur temps de manière plus efficace. L'un des moyens pour y parvenir est d'utiliser un servomoteur. Il faudra le programmer pour le rendre autonome. Pour cela, nous allons utiliser une carte électronique. Les activités suivantes vont nous familiariser avec la programmation en micro-Python.

PRISE EN MAIN DE PYBOARD

Voici Pyboard, une carte électronique capable d'exécuter MicroPython qui va nous permettre de programmer notre système d'arrosage automatique.

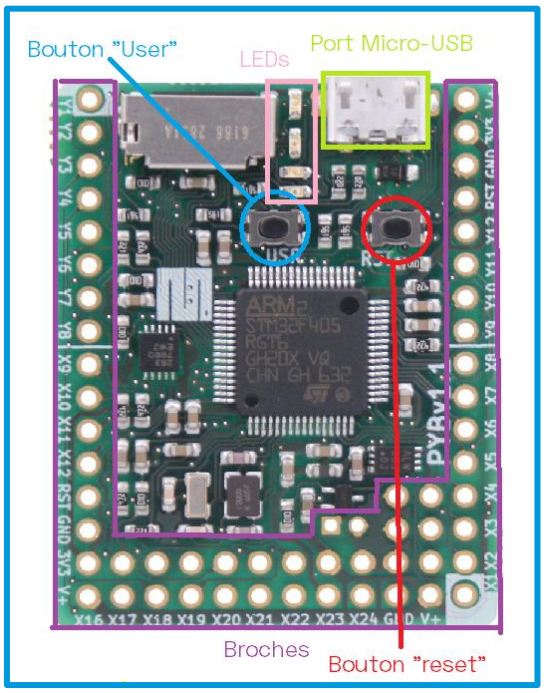
Les éléments qui nous seront utiles :

Le port micro-usb est l'alimentation et doit être connecté à un ordinateur via un câble micro-USB pour télécharger le code de l'ordinateur vers la carte.

Le bouton « user » est un élément programmable, de même que les leds.

Le bouton « reset » n'est pas programmable ; il sert à réinitialiser la carte.

Les broches permettent de brancher des éléments extérieurs à la carte (elles ne seront pas utilisées dans cette première partie).



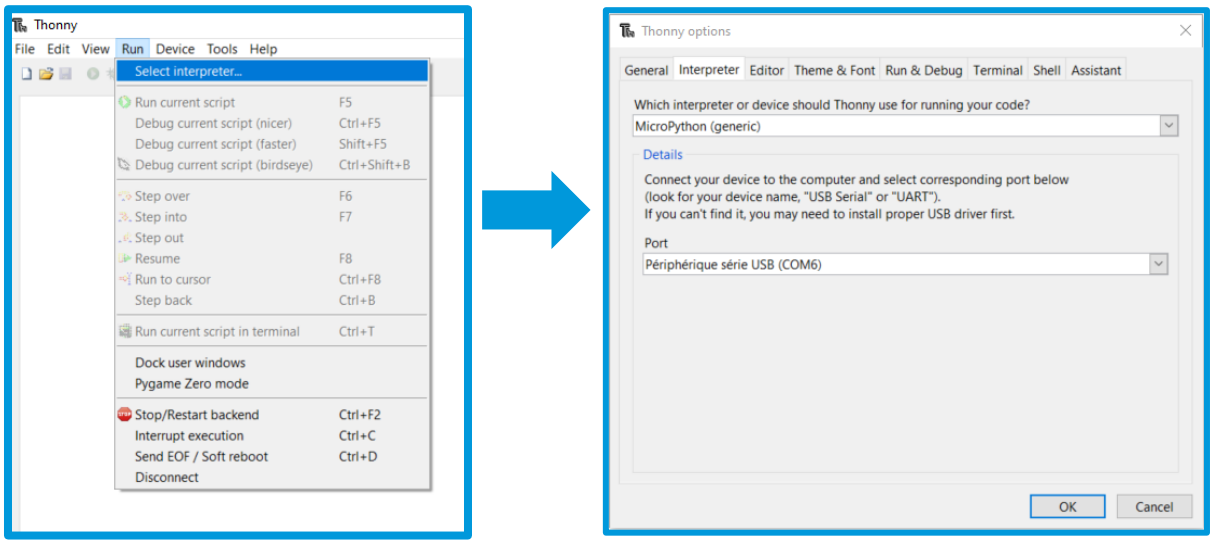
Pour élaborer les programmes et les transférer sur la carte Pyboard, nous allons utiliser le logiciel Thonny.

Défi 0 : Brancher Pyboard à l'ordinateur et le sélectionner comme interpréteur

Par défaut, Thonny utilise le programme comme interpréteur (c'est-à-dire comme entité qui exécutera le code). Or nous voulons utiliser Pyboard.

Afin de définir l'interpréteur, suivez les instructions suivantes :

a. Allez dans l'onglet *Exécuter* (ou *Run* si le programme est en anglais) et, dans le menu déroulant, cliquez sur « Sélectionner l'interpréteur ». Une fenêtre s'ouvrira sur l'onglet *Interpréteur* (si ce n'est pas le cas, sélectionnez l'onglet *Interpréteur*).
b. Dans le menu déroulant « Quel interpréteur ou appareil Thonny doit-il utiliser pour exécuter votre code ? », sélectionnez « MicroPython (generic) ».
c. Si ce n'est pas encore fait, connectez Pyboard à l'ordinateur à l'aide du câble Micro-USB.
d. Dans le menu déroulant « Port », sélectionnez « Périphérique série USB (COM6) » (le numéro après COM dans les parenthèses sera différent).



Défi 1 : Écrire une phrase dans la console

La fonction « print » permet d'écrire dans la console. Ici nous choisissons d'écrire du texte mais on peut imprimer des valeurs de variables, de mesures, etc.

```
print ('Hello world')
# Ceci est un commentaire
# Ce code ne fonctionnerait pas : print (bienvenue à tous)
# Celui-ci non plus : print 'bienvenue à tous'
```

Placer le signe # permet d'insérer un commentaire dans le code, c'est-à-dire des lignes qui ne sont pas du code à proprement parler et seront ignorées lors de l'exécution du programme

mais qui peuvent servir à expliquer une ligne de code à destination des autres utilisateur·rices. Dans l'exemple ci-dessus, les 3 dernières lignes de couleur grise sont des commentaires qui donnent des explication supplémentaires. Elles ne doivent pas être recopiées lors de l'écriture du programme.

→ **Exercice :** écrire une phrase au choix à l'aide de la fonction « print ».

Défi 2 : Allumer les leds intégrées à la carte

« Import » permet d'importer des fonctions spécifiques. Ici, nous allons importer « pyb » qui est l'ensemble des fonctions de Pyboard.

Certaines fonctionnalités sont incluses dans d'autres « registres » : par exemple, « LED » est l'ensemble des fonctions des leds de Pyboard ; il se trouve dans le registre « pyb ».

```
import pyb # permet d'utiliser des fonctionnalités propres à Pyboard
from pyb import LED # permet d'utiliser les fonctions liées aux leds intégrées sur Pyboard
LED(1).on() # allume la led numéro 1 (il y a 4 leds au total).
```

→ **Exercice :** identifier la couleur de chacune des 4 leds.

Défi 3 : Éteindre toutes les leds de la carte

→ **Exercice :** éteindre toutes les leds en même temps.

Défi 4 : Insérer des délais

La bibliothèque « time » comprend des fonctions liées au temps telles que lancer un chronomètre, poser des délais, etc. Dans ce défi, nous aurons recours à la fonction « sleep » qui permet d'introduire des délais dans l'exécution des instructions du programme.

```
import pyb
import time # importe l'ensemble des fonctions liées au temps
from pyb import LED
print ('0')
time.sleep(1) # met le programme "en pause" pour 1 seconde : le programme ne s'arrête pas vraiment, mais attend 1 seconde avant d'exécuter l'instruction suivante. L'écriture "time.sleep" signifie que l'on va chercher dans la bibliothèque "time" la fonction « sleep »
print ('1')
time.sleep(1)
print ('2')
```

→ Exercice : allumer la led 1 pendant 1 seconde puis l'éteindre et allumer la led 2 pendant 2 secondes, l'éteindre, attendre 1 seconde et allumer la led 3.

Défi 5 : Utiliser une variable

Une variable est une donnée dont la valeur est susceptible de changer durant l'exécution du code. Attention à ne pas faire d'amalgame avec une variable mathématique. En programmation, une variable a une valeur précise. On ne le connait pas spécialement à tout moment (en particulier dans un cas comme le nôtre où nous utilisons des valeurs aléatoires) mais elle est définie en amont, soit dans le code (comme dans le cas présenté ici), soit via des instruments de mesures (thermomètre, capteur de lumière, etc.).
Dans l'exemple ci-dessous, nous introduisons une variable que nous nommons « temps ».

```
import pyb
import time
from pyb import LED
from time import sleep # importe la fonction « sleep » ou attendre. Cela permet de ne pas avoir à taper « time.sleep » dans le code ; on peut simplement écrire « sleep ». L'unité est en secondes.
from random import * # importe la bibliothèque « * » des fonctionnalités aléatoires
temps = random() # on définit la variable « temps » : sa valeur est donnée par la fonction « random() » qui génère un nombre aléatoire dans la plage [0,0 ; 1,0].
# insérer une fonction
sleep(temps)
# insérer une fonction
```

→ **Exercice :** allumer les leds rouge, verte et jaune à intervalles réguliers, mais aléatoires, et imprimer la valeur de l'intervalle dans la console.

Défi 6 : Utiliser des « boucles »

La fonction « boucle » n'existe pas en tant que telle pour Pyboard mais il y a plusieurs façons de programmer une boucle.
Deux programmation de boucle sont présentées ci-dessous ; chacune a ses avantages et ses inconvénients.

Faire une boucle avec « for » :

```
import pyb
import time
from pyb import LED
from time import sleep
from random import *
temps = random()
i=1
for i in range(1,4):
    print(i)
    sleep(temps)
```

Faire une boucle avec « while » :

```
import pyb
import time
from pyb import LED
from time import sleep
from random import *
temps = random()
i=1
while(i < 5):
    print(i)
    sleep(temps)
    i=i+1
```

Si l'on traduit en texte la boucle établie avec la fonction « for », cela donne :

> *Pour i allant de 1 à 4, écrire sur la console la valeur de i, puis attendre un nombre de secondes égal à la valeur de la variable « temps ».*

Pour la boucle établie avec la fonction « while », on lit :

> *Tant que i est plus petit que 5, écrire sur la console la valeur de i, puis attendre un nombre de secondes égal à la valeur de la variable « temps », puis ajouter 1 à la valeur de i.*

La fonction « for » incrémente automatiquement la variable de 1 (la valeur de la variable augmente de 1). L'utilisation de « while » nécessite de redéfinir la variable en fin de boucle, ce qui représente une ligne de code supplémentaire mais permet de travailler avec des décimales.

Notez que les lignes sous les fonctions « for » et « while » sont décalées, on parle d'« indentation » : on met des espaces en début de ligne. Toutes les lignes indentées sont incluses dans la ligne supérieure, elles forment le bloc lié à la fonction. Dans le cas des boucles ci-dessus, les lignes indentées sont les lignes qui seront répétées. Pour sortir du bloc et continuer à écrire le code, il suffit d'écrire les lignes sans indentation, sans mettre d'espaces au début de la ligne donc.

→ **Exercice :** faire clignoter 4 fois une led avec un intervalle de temps aléatoire.

Défi 7 : Faire un blink infini

→ **Exercice :** faire clignoter une led indéfiniment.

Défi 8 : Utiliser des conditions

Les conditions permettent de réaliser des actions différentes en fonction du résultat d'une mesure, d'un état, …
S'il n'y a qu'une condition, on emploiera juste un « if », et si il n'y en a que 2, on ajoutera un « else ». Si les conditions se multiplient, il faudra utiliser des « elif » entre les deux fonctions précédentes. Il remplit la même fonction que les « if » mais vient après ceux-ci. Si on ne mettait que des « if », plusieurs peuvent s'actionner en même temps (c'est parfois ce qu'on recherche mais pas dans notre cas). Ici, si le « if » s'active, le code ne prendra pas les « elif » ni le « else » en compte.

```
import pyb
while True:
    sleep(1)
    test=randint(1,6) # on crée une variable appelée test qui a une valeur aléatoire entre 1 et 6.
    if 0 <test<=2: # « if » permet d'établir une condition. Si la condition est remplie, tout ce qui est indenté dans le « if » sera exécuté.
    # Il y a plusieurs possibilités pour écrire cette instruction : « if 1<=test<=2 » est correct aussi, etc.
        print(1)
    elif 2 <test<=4: # « elif » est la contraction de « else » et « if ». Il agit comme un « if », mais doit être précédé d'un « if ». Il peut y avoir plusieurs « elif ».
        print(2)
    else # else veut dire « sinon » et couvre tous les cas qui ne rentrent pas dans les conditions des « if » et « elif » qui le précèdent.
        print(3)
```

→ **Exercice :** Tirer un nombre aléatoire entre 1 et 10 et, en fonction du résultat :
1 ou 2 : allumer la led verte pendant 2 s
3 ou 4 : allumer la led jaune pendant 2 s
5 ou 6 : allumer la led rouge pendant 2 s
7 ou 8 : allumer la led bleue pendant 2 s
9 ou 10 : guirlande (= blink vert, jaune, rouge, bleu ; délai 0,25 s)

Défi 9 : Utiliser des états pour un capteur

Nous allons utiliser ici le bouton poussoir dont la sortie est un état logique : les valeurs possibles sont soit « True » (lorsqu'il est enfoncé) soit « False » (lorsqu'il est relâché).

```
import pyb
from pyb import LED
from pyb import Switch # la fonction « Switch » permet d'utiliser les boutons poussoirs intégrés à Pyboard.
sw= Switch()
while True:
    print(sw.value()) # les "valeurs" possibles sont soit « True » (enfoncé), soit « False » (relâché).
```

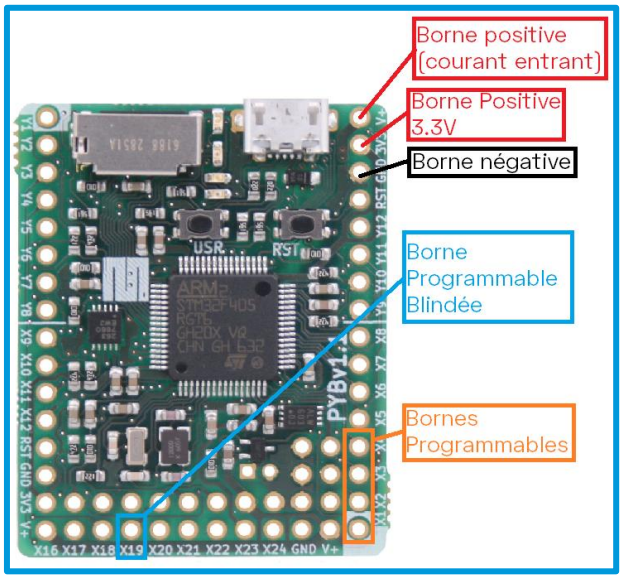
→ **Exercice :** Utiliser le bouton poussoir pour allumer ou éteindre une led.

Programmation d'éléments extérieurs à Pyboard

En plus des éléments utilisés précédemment, nous allons maintenant brancher différents composants électriques aux bornes de Pyboard.

Défi 0 : Connecter une breadboard à Pyboard

Les bornes que nous utiliserons dans la suite de l'activité sont indiquées ci-dessous.



Les bornes GND, 3V3 et V+ ne sont pas programmables. Leur seule fonction est de servir de borne, comme une pile.

- La borne **GND** a une tension de **0 V**.
- La borne **3V3** délivre **3,3 V**.
- **La tension de la borne V+ dépend du courant entrant** (si la carte Pyboard est branchée à un port USB d'un ordinateur, la tension de la borne V+ sera de 5 V).

Parmi les bornes programmables, nous utiliserons les bornes X1 à X4 qui permettent, entre autres, de commander des servo-moteurs.

Pour brancher la plaque d'essai (breadboard) à Pyboard et réaliser le premier circuit, suivez les étapes suivantes (schéma du montage ci-dessous) :

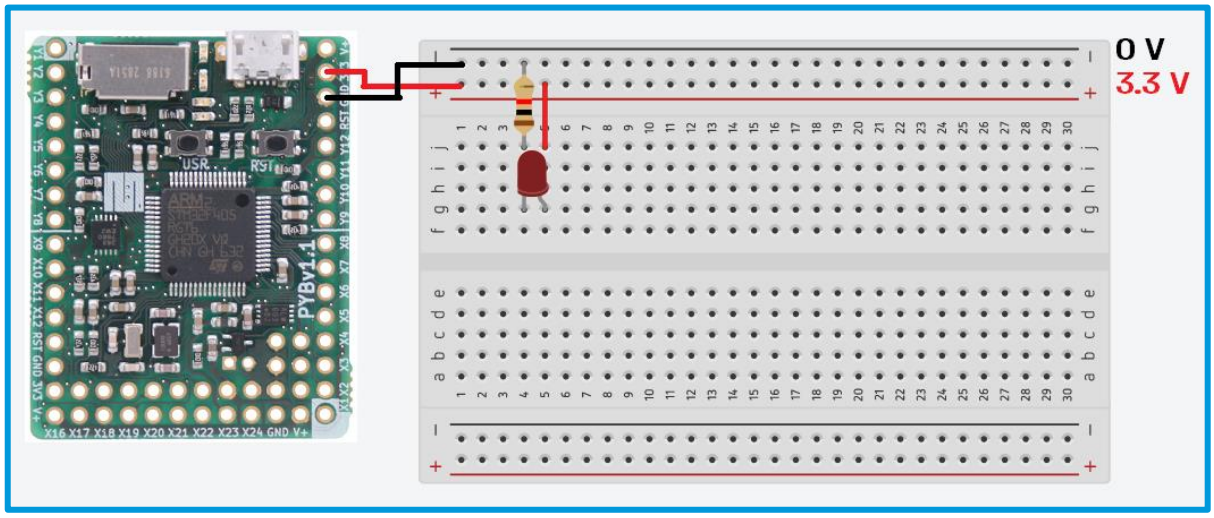
a. Connecter la borne positive 3V3 à la ligne « + » de la breadboard et la borne négative GND à la ligne « - ». Pour respecter les conventions, utilisez des connecteurs rouges pour relier des éléments à la borne positive et bleus ou noirs pour la borne négative.

b. Placez une led (*light-emitting diode* ou diode électroluminescente) reliée d'un côté au pôle négatif et de l'autre au pôle positif. La longue patte de la led, l'anode, doit être connectée au pôle « + ».

c. On met une résistance dans le circuit pour limiter le courant qui traverse la led afin de ne pas dépasser son courant nominal, ce qui aurait pour conséquence de la griller. La

résistance peut se mettre avant ou après la led ; tant que les deux éléments sont en série, il n'y a pas de problème (attention ce n'est pas vrai pour tous les composants : notamment, si l'on veut faire des mesures, se trouver avant ou après une résistance peut tout changer).

d. Brancher Pyboard à l'ordinateur.

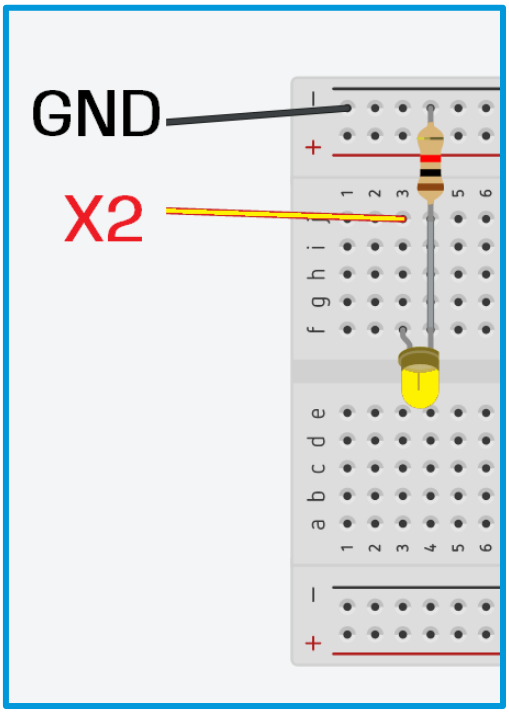


→ La led est-elle allumée ou éteinte ? Pourquoi ?

Défi 1 : Allumer une led extérieure à la carte

Les leds se branchent sur les broches X1 à X4. Dans l'exemple suivant, la led est connectée à la broche X2.

```
import pyb
from pyb import Pin # importe les fonctions relatives aux "Pins", càd les broches
Y = Pin.board.X2 # on donne le nom « Y » à la broche X2
pyb.Pin(Y , Pin.OUT) # on définit « Y » comme une sortie, c'est-à-dire qu'on pourra lui envoyer des informations.
while True:
    Y.value(1) # on définit la valeur de « Y » sur 1, ce qui veut dire un état haut, c'est-à-dire que le courant passe : la Led branchée sur cette broche sera allumée. Si on veut que la led soit éteinte, donc que le courant ne passe pas, on définit la valeur de la broche sur 0 (état bas).
```



→ **Exercice :** Brancher une led supplémentaire sur la broche X3, l'allumer et éteindre Y.

Défi 2 : Allumer 4 leds en alternance

→ **Exercice :** Brancher une troisième led (sur la broche X4) et allumer les 3 leds sur la breadboard et la led 1 intégrée à Pyboard en alternance pendant 0,5 s.

Défi 3 : Prendre des mesures

```
import pyb
from pyb import Pin
from pyb import ADC # ADC (Analog to Digital Converter) est un ensemble de fonctions particulier qui permet de lire des valeurs analogiques et les convertit en valeurs digitales.
adc = ADC( 'X19' ) # on définit la broche  X19 comme « lecteur », qui sera traduit avec les fonctions ADC
import time
from time import sleep
while True:
    A = adc.read() # on crée une variable « A » qui est égale à la valeur obtenue via adc défini plus haut. Il est important de définir cette valeur dans la boucle sinon la valeur est fixée avant et, une fois dans la boucle, même si la valeur enregistrée varie (adc), la valeur « A » ne changera pas.
    print(A)
    sleep(1) # on définit ici un sleep pour que le print ne se fasse pas en continu, ce qui au bout d'un moment fait planter le programme.
```

La broche X19 est une entrée analogique qui permet de lire la valeur d'une tension (située entre 0 et 3,3 V) présente sur la broche. Le convertisseur analogique (ADC) convertit la valeur analogique en une valeur numérique.

→ **Exercice :** Brancher un potentiomètre sur la broche X19 et observer la variation des valeurs mesurées en fonction de la position du curseur.

Défi 4 : Tester le capteur d'humidité

Afin d'automatiser entièrement le système d'arrosage des plantes, nous devons savoir quand la plante a besoin d'être arrosée. Dans cette activité, les élèves sont donc initiés au capteur d'humidité du sol et vont le calibrer pour pouvoir déterminer la valeur qu'il faudra utiliser par la suite pour déclencher/stopper l'arrosage.

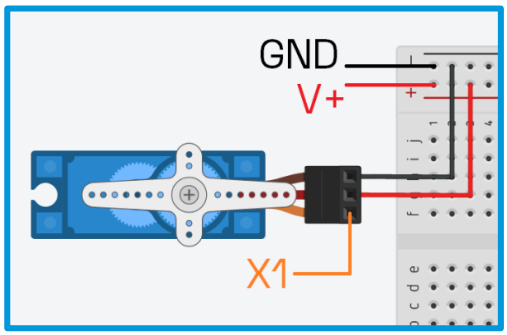
→ **Exercice :** Remplacer le potentiomètre par le capteur d'humidité et mesurer l'humidité de différents milieux.
- Quelle valeur le capteur donne-t-il lorsqu'il est placé dans l'eau ?
- Quelle valeur le capteur donne-t-il dans l'air "sec" ?

- Quelle serait la valeur appropriée pour faire passer votre système de l'état "marche" à l'état "arrêt" ?

Défi 5 : Contrôler un servomoteur

```
import pyb
from pyb import Pin
from pyb import Servo # importe les fonctions de Pyboard pour les servomoteurs.
servo1position = Servo(1) # on définit la première broche, X1, pour notre servomoteur.
while True:
    servo1position.angle(90) # on positionne le servomoteur à un angle de 90°.
```

Le servomoteur se branche sur la broche V+.



→ **Exercice :** Trouver la position zéro du servomoteur.

Défi 6 : Contrôler le servomoteur à l'aide du potentiomètre

→ **Exercice :** Régler la position du servomoteur en fonction du potentiomètre.

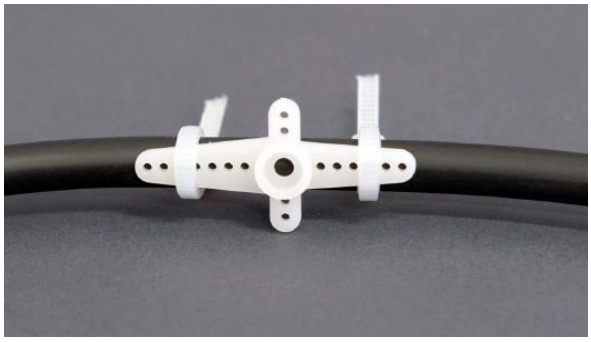
Tenez évidemment compte du décalage observé lors du défi précédent pour vos angles !

MONTAGE FINAL

On y est presque ! Nous avons maintenant une bonne compréhension de tous les éléments de notre système. Il est maintenant temps de les rassembler et de tester le système pour voir si tout fonctionne.

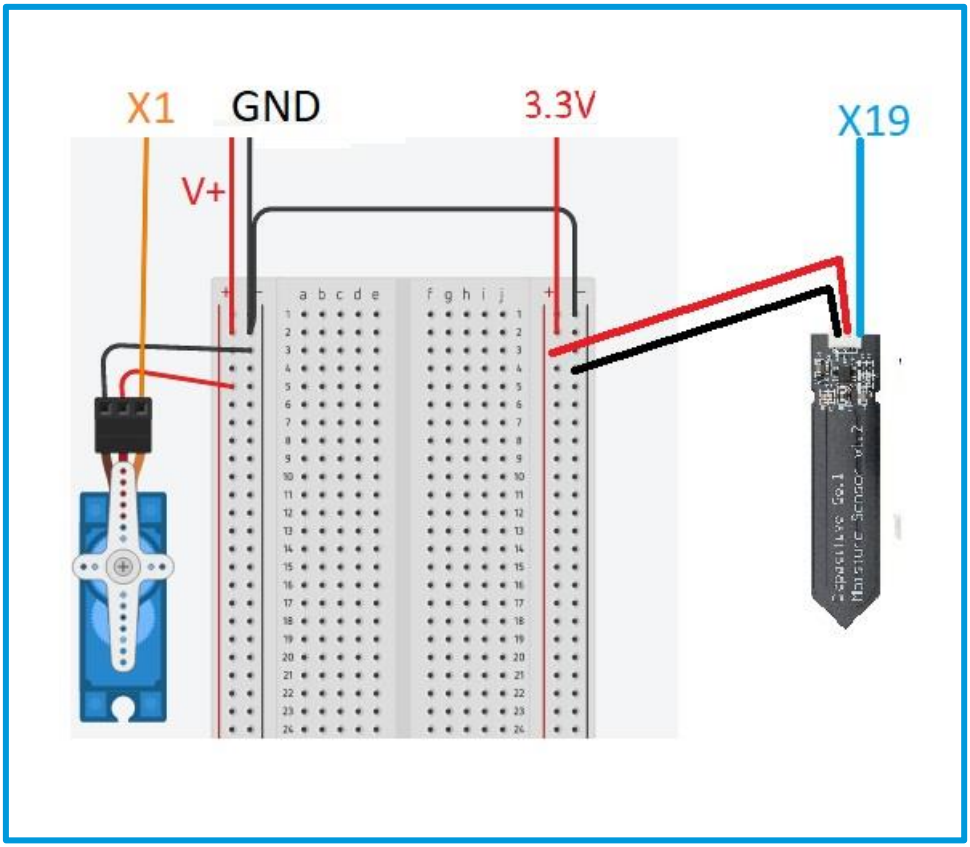
Connectez les composants

Nous devons d'abord monter notre servo et le connecter à la conduite d'eau. La pâte adhésive peut être utilisée pour monter le servo sur un mur approprié. Là encore, un pistolet à colle permettrait d'obtenir une fixation plus permanente, mais il est probable que vous deviez apporter des ajustements à votre installation au cours des premières étapes. Un jeu d'hélices est inclus dans la plupart des kits de servo. Nous utiliserons l'une d'entre elles pour connecter le tuyau au servo, en utilisant deux serre-câbles pour le fixer.



La longueur entre l'extrémité du tuyau et l'hélice est importante - le coude créé par le servo doit être suffisamment long pour arrêter l'écoulement de l'eau. Si le coude est trop petit, l'eau continuera à couler en position "arrêt" - les plantes martiennes ne survivront pas !
Pour connecter le tuyau au servo, il suffit de cliquer pour le mettre en place.

Il ne reste plus qu'à réaliser le circuit avec Pyboard et les différents éléments à programmer. Aidez-vous du schéma des connexions de la page suivante.

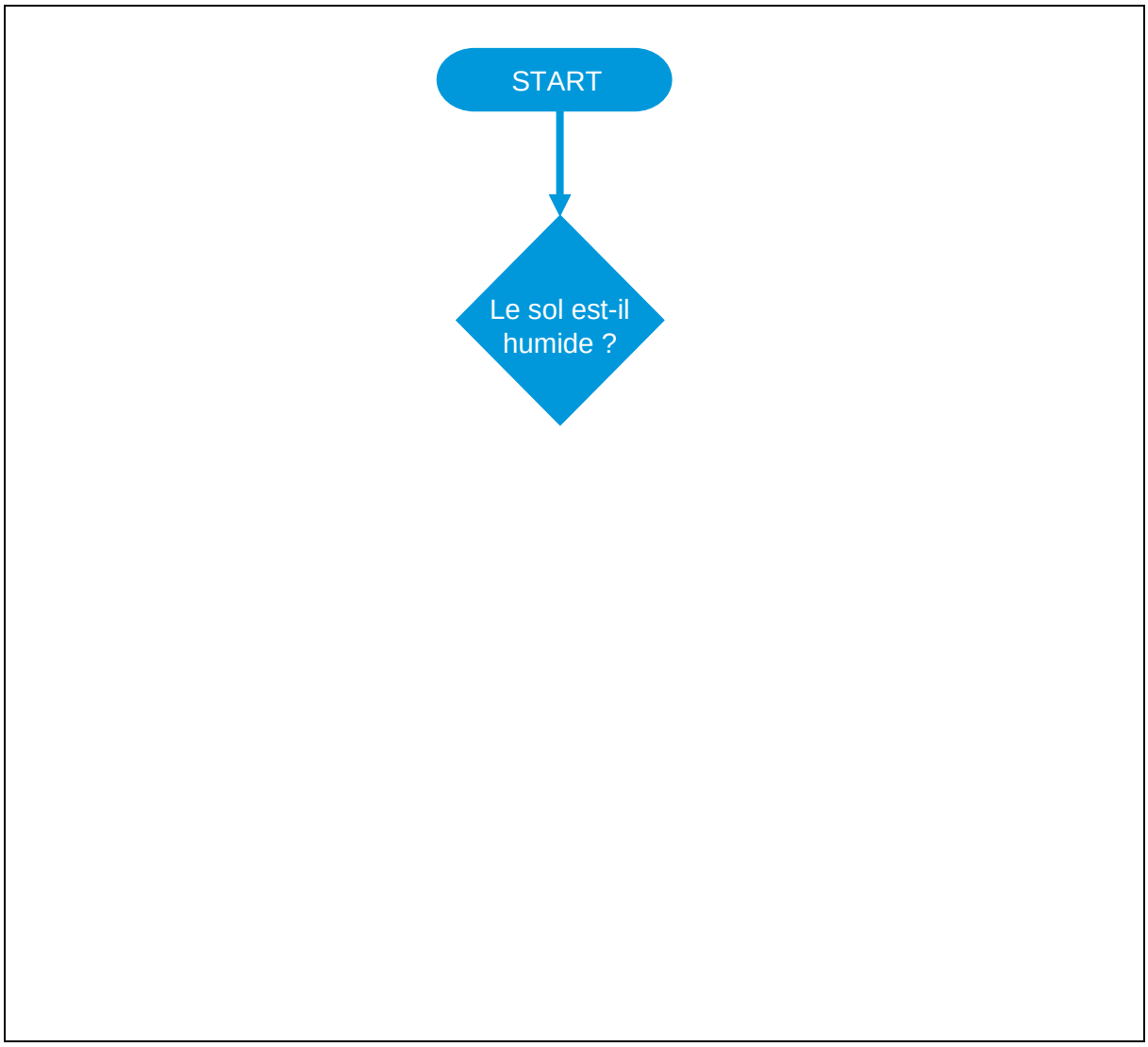


Programmez votre système

Il est maintenant temps de programmer Pyboard pour qu'il fasse fonctionner tous les composants automatiquement.

Pour faciliter l'écriture du code, nous pouvons d'abord réfléchir au problème sur papier, en utilisant la même logique que celle que vous utiliserez au moment d'écrire le code. Cela se fait facilement dans un organigramme. En général, dans un organigramme, un rectangle représente une commande et un losange une question/décision. Des flèches sont utilisées pour indiquer le chemin à suivre dans le diagramme, en fonction des décisions prises.

→ **Exercice 1** : Essayez d'écrire votre "code" sous la forme d'un organigramme. Les deux premiers éléments de l'organigramme sont déjà placés dans le cadre suivant.



L'arrosage sera contrôlé par la position du tuyau, elle-même déterminée par le servomoteur. Nous devons donc connaître les valeurs des angles à mettre dans le programme.

→ **Exercice 2 :** Définir les angles des positions « ON » (arrosage) et « OFF » (arrêt de l'arrosage) du servomoteur.
- Angle pour la position « ON » :
- Angle pour la position « OFF » :

Nous sommes maintenant prêts à rédiger le programme pour faire fonctionner le système ! Essayez d'écrire votre programme en utilisant tout ce que nous avons vu précédemment :
- importer les bibliothèques et fonctions nécessaires pour programmer Pyboard, des pauses, des mesures et le contrôle d'une servomoteur
- faire une boucle infinie
- utiliser des conditions
- définir une variable
- faire une mesure
- contrôler un servomoteur
- la valeur seuil d'humidité pour mettre en route l'arrosage (pour rappel, celle-ci a été définie lors du défi 4 avec le capteur d'humidité)

- les valeurs des angles du servo pour les positions « ON » et « OFF ».

Votre programme :

Vous êtes maintenant prêt à exécuter votre programme ! C'est une bonne idée de faire d'abord un test sans eau dans le système - l'eau et l'électronique ne se mélangent pas bien et le programme pourrait ne pas fonctionner exactement comme vous l'avez prévu… !

Lorsque vous obtenez le résultat souhaité, déconnectez Pyboard de l'ordinateur, remplissez le réservoir d'eau et utilisez des piles (4 x 1,5 V) pour alimenter la carte. Votre système automatique d'arrosage est prêt pour fonctionner tout seul (il faut juste penser à remplir le réservoir d'eau et changer les piles de temps en temps) !