|  |
| --- |
| **Robotique au jardin** |
|  |

|  |
| --- |
|  |

|  |
| --- |
| Initiation à la programmation |

|  |
| --- |
| 12-18 ans |

|  |
| --- |
| Arrosage automatique |

Séance 4 : Défis Pyboard

Objectif : Programmer la Pyboard afin de contrôler des composants extérieurs à la carte

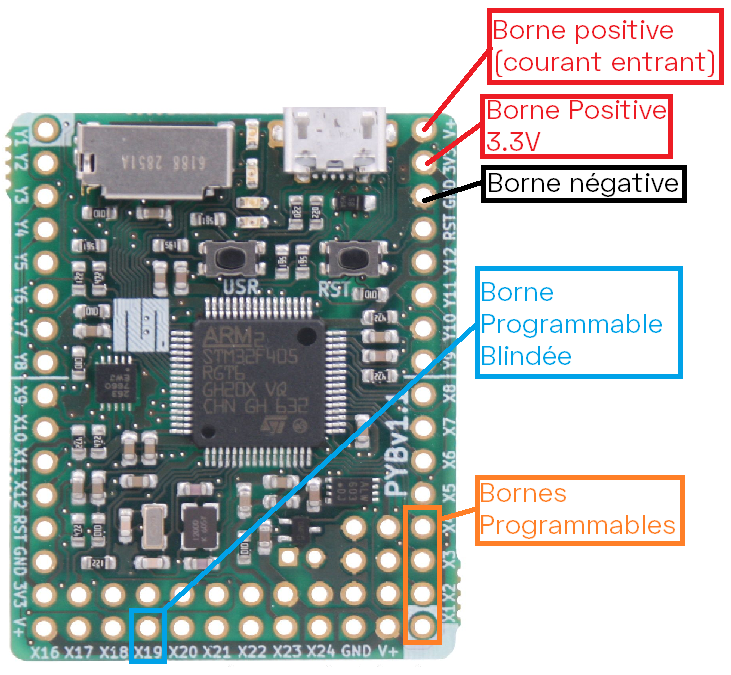
|  |
| --- |
| Notions abordées : Resistance, tension, bornes, programmation, bits |

Durée : 2h

|  |
| --- |
| Matériel  Par participant/groupe :   * Un ordinateur avec Thonny * Un câble micro USB * 3 résistances de 220 Ohms * 3 LEDs de couleur (plus réserves) * Une plaque d’essais * Un servomoteur * Un potentiomètre * Un capteur d’humidité * Jeu de câbles Dupont |

**Defi programmation d’élémenents exterieurs à la carte.**

**Défi 0 :** Brancher la pyboard

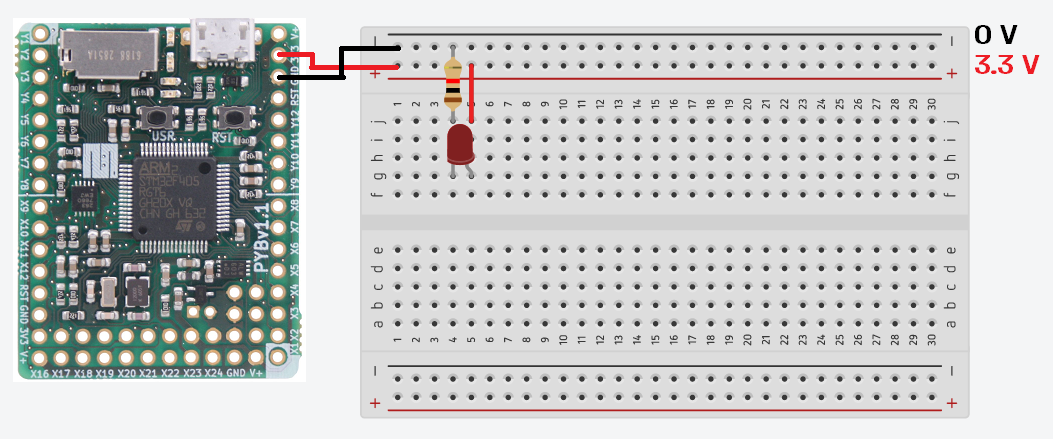
*****En plus des éléments utilisés lors de la précédente séance, nous serons amenés à brancher des bornes aux différents composants électrique.  
3 zones à distinguer que nous utiliserons pour nos défis :*

* *Les bornes + et – (il y a 2 bornes + qui délivrent des tensions différentes)*
* *Les bornes « classiques »*
* *Les bornes blindées*

*Nous détaillerons tout ça au moment opportun.*

*Pour pouvoir les brancher, il est nécessaire que des broches soient soudées aux bornes désirées.*

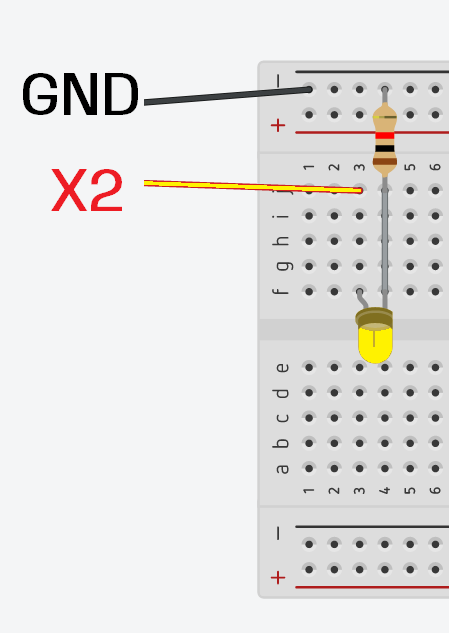
* Brancher la pyboard à la plaque d’essais : Cfr image

*Rien de compliqué, mais il y a quelques conventions à respecter.  
Pour relier des éléments à la borne positive : utilisez des connecteurs rouges, et pour la borne négative du bleu ou du noir.  
Attention aussi au sens des LEDs, la longue « patte » doit être connectée au pôle positif.  
Enfin, insistez pour qu’ils demandent la vérification ( par vos soins) avant de lancer des programmes. A priori il y a des sécurités, mais des courts-circuits peuvent endommager la carte ou le PC.*

Attention à bien brancher la borne 3.3 V et non la V+.

La LED s’allume normalement sans la moindre programmation. Elle est branchée sur une borne négative d’un côté, et positive de l’autre, comme à une pile.  
On met une résistance dans le circuit pour éviter que la LED ne saute. Si on ne met pas de résistance, le courant est trop fort et certains éléments électriques peuvent griller. Pas très grave dans le cas d’une LED, mais certains composants sont plus couteux.  
La résistance peut se mettre avant ou après la LED, tant que les 2 sont en série, il n’y a pas de problème (attention ce n’est pas vrai pour tous les composants, notamment si on veut faire des mesures, selon si on se trouve avant ou après une résistance peut tout changer).

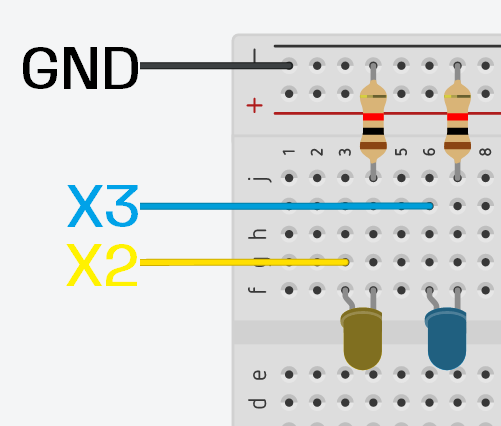
Les broches GND, V+ et 3V3 ne sont pas programmable. Leur seule fonction est de servir de bornes, un peu comme une pile.   
LA broche GND est comme le moins (0V), la 3V3 envoie 3,3 V, et la V+ dépend du courant entrant (dans note cas la carte est branchée par USB et la tension sera donc de 5 V).

**Défi 1 :** Allumer une LED branchée

**import** pyb  
**from** pyb **import** Pin *# importe les fonctions relatives aux “Pins”, aux 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, que la Led y sera allumée. On peut aussi définir la valeur, l’état, de la broche sur 0 (bas), si on veut que le courant ne passe pas.*

***Les leds se branchent sur les broches en bas à droite de la carte. Ces broches sont un peu particulières car elles permettent de contrôler des servomoteurs (nous y viendrons d’ailleurs plus tard).  
Ce sont des broches sur lesquels nous allons pouvoir régler le courant grâce au programme.***

* Brancher une Led supplémentaire sur la broche X3, l’allumer, et éteindre Y.

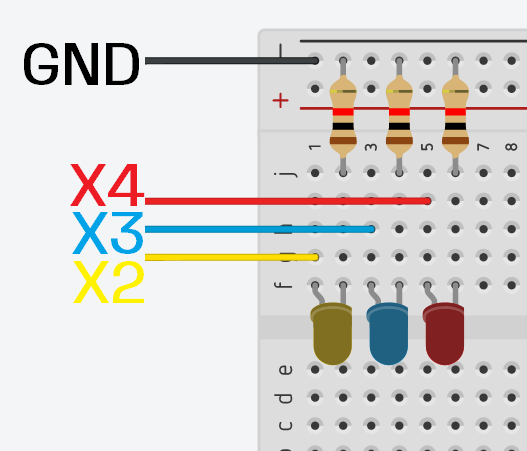
Solution :

**import** pyb  
**from** pyb **import** Pin   
Y = Pin.board.X2   
B = Pin.board.X3  
pyb.Pin( Y , Pin.OUT )   
pyb.Pin( B , Pin.OUT )  
**while**(True):  
 Y.value(0)   
 B.value(1)

**Défi 2 :** Allumer 4 leds en alternance

* Brancher une troisième led sur la broche X4 et allumer les 3 LEDs en alternance avec la LED 1 intégrée à la pyboard.

*Rien de nouveau ici, juste un exercice pour bien faire la différence entre les LEDs intégrées et externes, et réintroduire la fonction « sleep ».*

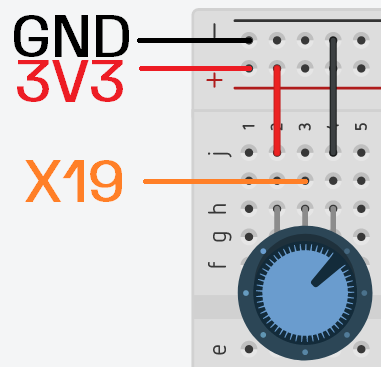
Solution :

**import** pyb  
**import** time  
**from** pyb **import** Pin   
**from** pyb **import** LED  
**from** time **import** sleep  
Y = Pin.board.X2   
B = Pin.board.X3  
R = Pin.board.X4  
pyb.Pin( Y , Pin.OUT )   
pyb.Pin( B , Pin.OUT )  
pyb.Pin( R , Pin.OUT )   
**while**(True):  
 Y.value(1)  
 LED(1).off()  
 sleep(0.5)  
 Y.value(0)  
 B.value(1)   
 sleep(0.5)  
 B.value(0)  
 R.value(1)  
 sleep(0.5)  
 LED(1).on()  
 R.value(0)  
 sleep(0.5)

**Défi 4 :**  Prendre des mesures avec la carte

**import** pyb  
**from** pyb **import** Pin   
**from** pyb **import** ADC *# ADC est un ensemble de fonctions particulier, nous permettant de traduire un signal entrant sur une broche en valeur analogue*  
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 créée une variable « A » qui est égal à la valeur obtenue via adc définit plus haut. Il est important de définir cette valeur dans la boucle sinon, la valeur est fixée avant et même si la valeur enregistrée varie, « 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*

* Brancher un potentiomètre et observer la variation des valeurs mesurées.

Solution : Le code ne change pas. Il faut juste observer les valeurs dans la console.

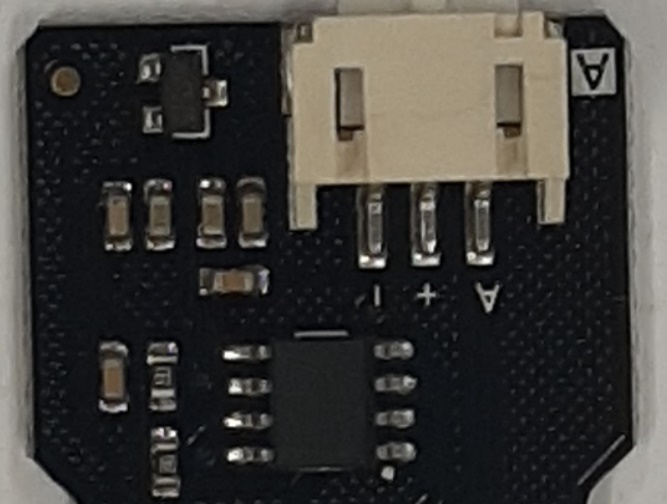
*Le principe de fonctionnement du potentiomètre est supposé connu.*

La broche X19 est dite blindée. Elle est moins sensible aux interférences et offre donc une lecture plus précise.

En tournant le potentiomètre on devrait observer des valeurs allant de 0 à 4096.  
Elles varient de façon proportionnelle au potentiomètre, c’est-à-dire si le potentiomètre est au maximum, la valeur imprimée est proche de 4096, au minimum proche de 0, et au milieu aux alentours de 2000. La valeur est en fait en bits un bit en programmation c’est 2 valeurs possibles : 1 ou 0. Si une information est stockée sur 2 bits, on a 4 combinaisons possibles, sur 3 bits, 8 combinaisons, etc. ce sont les puissances de 2. On a donc une information stockée sur 12 bits, ce qui fait 2^12 combinaisons, soit 4096.  
0 correspond à une mesure de 0 V, 4096 à 3.3V.

**Défi 5 :** Mesurer l’humidité à l’aide d’un capteur

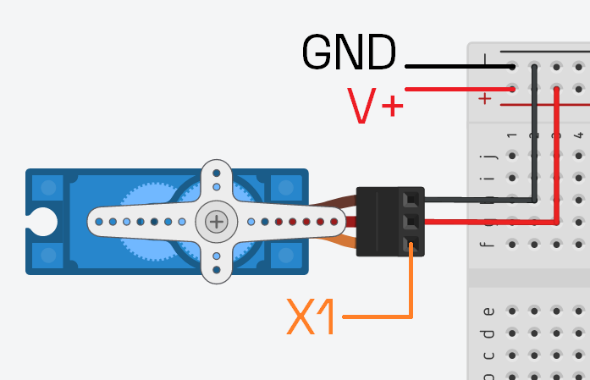
* Remplacez le potentiomètre par un capteur d’humidité et mesurer l’humidité de différents récipients

Le code sera le même que pour le défi 4 mais on débranchera le potentiomètre pour brancher un capteur d’humidité.  
La couleur des connecteurs peut varier mais on peut retrouver les indications près du port.

Plongez donc le capteur dans pots avec de la terre/sable avec des niveaux d’humidité différents et observez : plus la terre est humide, plus la valeur est basse, et inversement. Le capteur mesure en fait une résistance et pas l’humidité, puis convertit sa mesure en courant.

**Défi 6 :** Controller un servomoteur

**import** pyb  
**from** pyb **import** Pin   
**from** pyb **import** Servo *# importe les fonctions de la pyboard pour les servomoteurs.***import** time  
**from** time **import** sleep  
servolposition = Servo( 1 ) *# on définit la première broche X1 pour notre servomoteur***while**(True):  
 servolposition.angle(90) *# on positionne le servomoteur à un angle de 90°*

  
***Attention aux branchements***

*Le servomoteur se branche sur le V+ et non le 3.3V contrairement aux autres composants.  
Soyez attentifs, si le servo est branché sur 3.3 V ce n’est pas très grave, mais il faut surtout que tous* ***les autres composants ne soient pas sur le V+.***

*Pour ce qui est du code, rien de compliqué, on importe encore une série de fonction pour contrôler un nouveau type d’éléments, puis cet élément utilisent ces fonctions.  
  
Rmq : les broches X1 à X4 sont des broches servo, quand on parle de la broche « servo 1, » la carte sait qu’on parle de « X1 » et pas d’une autre broche, d’où le fait que « X1 » n’apparaisse pas dans le code.*

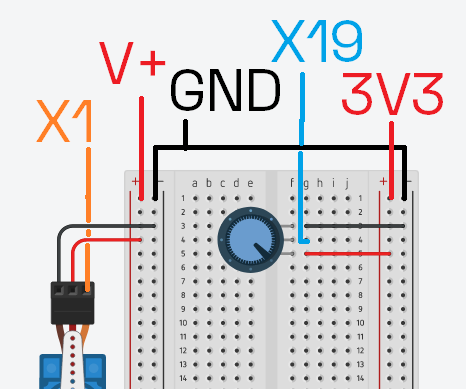
* Trouver la position 0 du servomoteur

*Dépendant du servomoteur et de la position à laquelle il est arrêté lorsqu’on le débranche, l’angle 0 du servo n’est pas spécialement tout à droit ou tout à gauche. Il faut donc un peu chipoter pour l’identifier.  
C’est important pour plus tard car un servomoteur ne peut pas faire de tour complet. Il couvre généralement un angle de plus ou moins 180°, et si le 0 n’est pas au 0 « physique », les angles à entrer dans le code seront différents (on peut par exemple avoir des angles négatifs.  
J’utilise par exemple un servomoteur qui est décalé de 75°. Pour avoir un angle de 0° réels, je devrais régler mon servo dans le programme sur -75°, et pour 180° réels, sur 105°.*Solution : Recopier le code, et tester des angles. Ils peuvent tester plusieurs angles en une seule exécution du programme, en enchainant la commande « servolposition.angle() » et en insérant un sleep entre chaque changement de position pour avoir le temps d’observer.

**Défi 7 :** Controller le servomoteur à l’aide du potentiomètre.

* Régler la position du servomoteur en fonction du potentiomètre.  
  Solution :

**import** pyb  
**from** pyb **import** Pin   
**from** pyb **import** Servo **import** time  
**from** time **import** sleep  
servolposition = Servo( 1 )   
**from** pyb **import** ADC  
adc = ADC( 'X19' )**while**(True):  
 A = adc.read()  
 **if**(A<=1350):  
 servolposition.angle(-15)  
 **elif**(1350<A<=2700):  
 servolposition.angle(45)  
 **else**:  
 servolposition.angle(105)

*Tenez évidemment compte du décalage observé au défi précédent pour vos angles.  
Encore une fois attention à ne brancher que le servomoteur sur V+, le potentiomètre doit être sur le 3V3.*