Ecole Robots
Apprendre à coder avec les robots éducatifs

Kit de programmation
CURSUS EDUCATION NATIONALE

Python - Niveau collège

Programmation appliquée

Partie 1 : La barrière automatique

Prénom : _____ Nom : _____ Classe : _____

ARTEC
Speechi

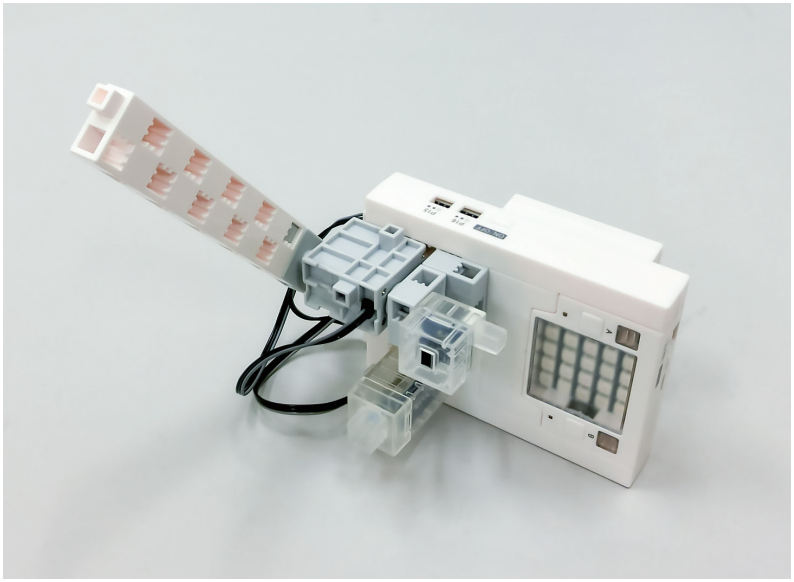
Les programmes nécessaires à la réalisation des robots sont disponibles en téléchargement sur le site www.ecolerobots.com.

Toutes les boîtes et les pièces détachées sont aussi disponibles sur le site www.ecolerobots.com.

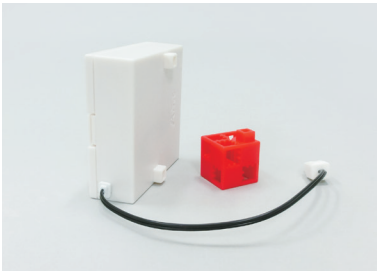
Sommaire

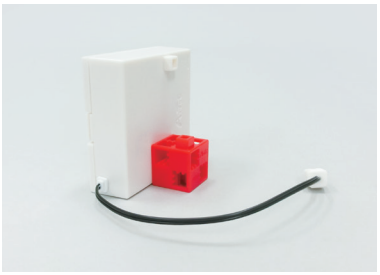
1. Notions de base sur les servomoteurs

1.1 Construction d'une barrière automatique

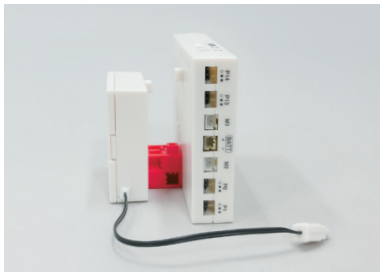


①

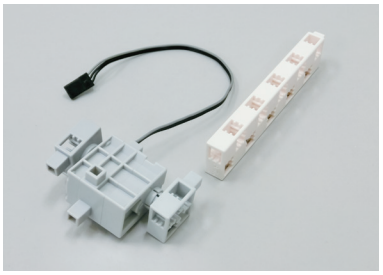


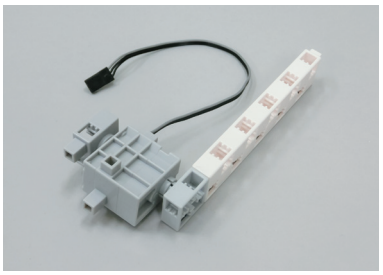


②

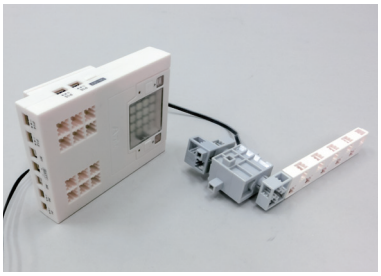


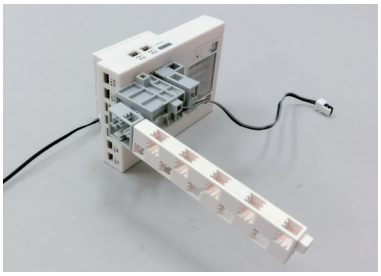
③

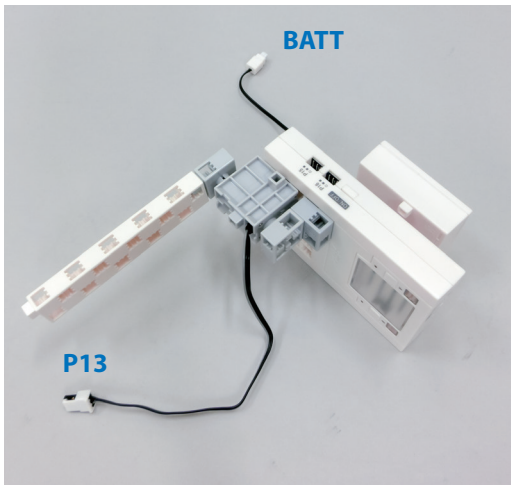


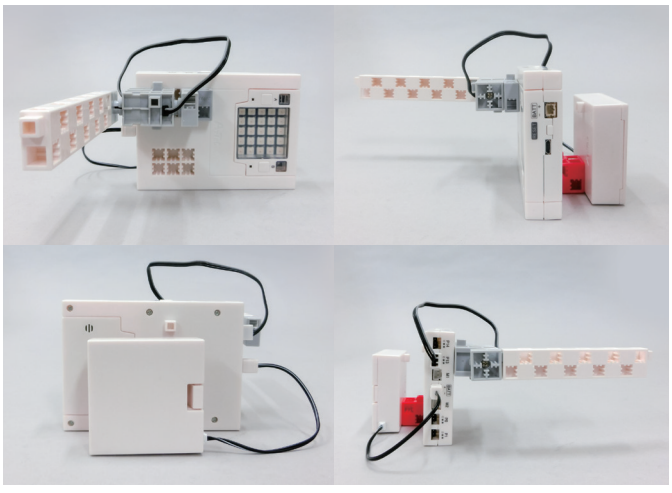


④







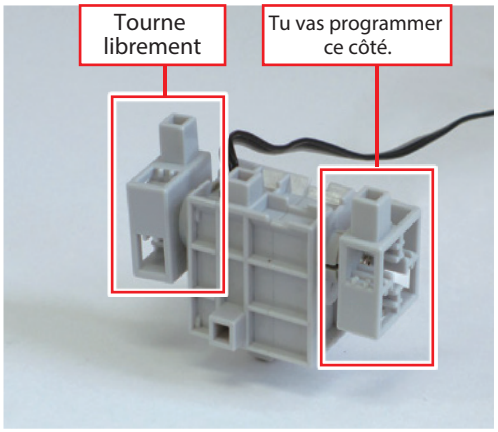


1.2 Comment fonctionne un servomoteur ?

1.2.1 Qu'est-ce qu'un servomoteur ?

- C'est un moteur que tu peux programmer pour tourner à un angle spécifique.
- Les angles peuvent être choisis entre 0 et 180 degrés.
- Le côté qui ne tourne pas facilement est celui que tu vas programmer !

★ Il faut ménager ton servomoteur !
Ne le tourne pas trop fort et n'essaie pas non plus de le forcer à tourner !



1.2.2 Teste ton servomoteur

Testons le servomoteur que tu as branché en P13 pour voir comment la programmation influence son fonctionnement.

★ À chaque fois que tu fais fonctionner des servomoteurs, assure-toi que le boîtier de la batterie est bien branché sur le port BATT de l'unité d'extension du robot !

Voici un programme que tu peux utiliser pour créer un objet qui te permettra de contrôler des servomoteurs :

```
1:  from pyatcrobo2.parts import Servomotor
2:
3:  sv13 = Servomotor('P13')
```

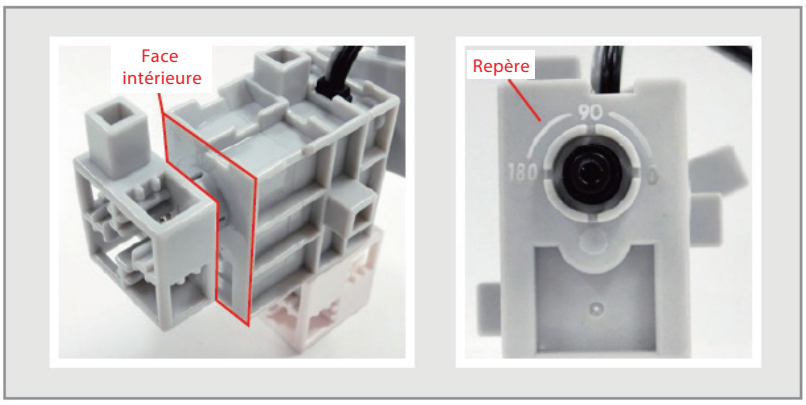
À la ligne 1, la commande **from pyatcrobo2.parts import Servomotor** te permet d'utiliser la classe **Servomotor** du module **pyatcrobo2.parts** de la bibliothèque ArtecRobo 2.0 (tu peux consulter l'**Annexe A** pour en savoir plus sur la classe **Servomotor**).

À la ligne 3, la commande **sv13 = Servomotor('P13')** crée un objet appelé **sv13**.

Tu peux utiliser les propriétés et les méthodes de ce nouvel objet pour contrôler le servomoteur branché sur le port P13 de l'unité d'extension du robot.

1.2.3 Réglage des angles du servomoteur

Tu peux utiliser les repères de graduation situés sur la face intérieure de ton servomoteur pour vérifier l'angle auquel le servomoteur est positionné.

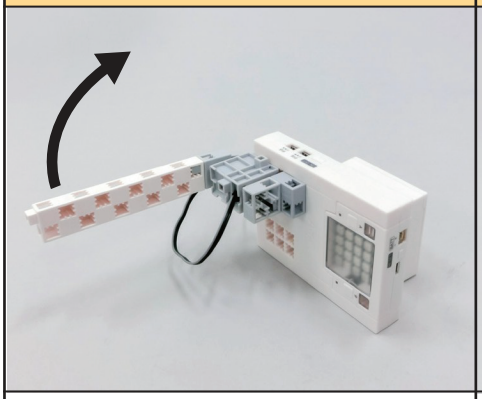
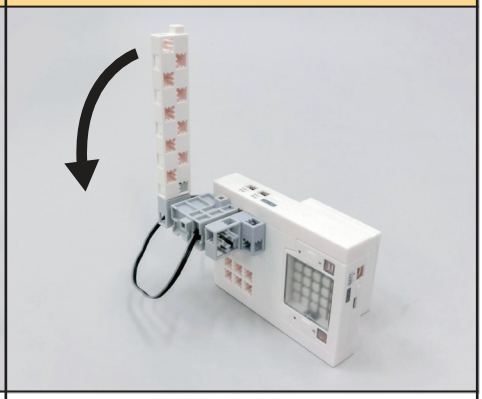


Allume l'alimentation, puis utilise la méthode **set_angle()** de la classe Servomotor pour changer l'angle du servomoteur.

Le paramètre de la méthode **set_angle()** peut être réglé sur n'importe quel angle entre 0 et 180 degrés.

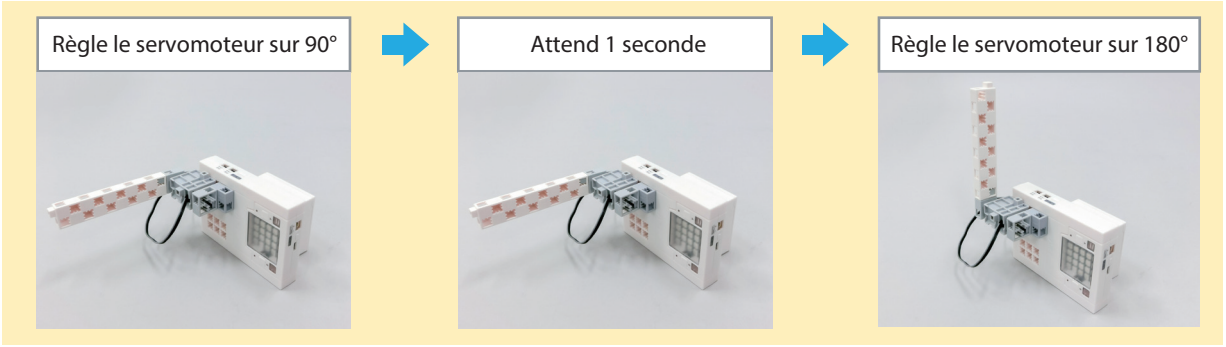
```
1: from pyatcrobo2.parts import Servomotor
2:
3: sv13 = Servomotor('P13')
4: sv13.set_angle(90)
```

La ligne 4 de ce programme utilise la méthode **set_angle()** pour régler l'angle de l'objet **sv13** que nous avons créé à la ligne 3 à une valeur de **90** degrés. Modifier ce nombre va changer l'angle du servomoteur. Teste différentes valeurs d'angle pour voir comment cela va modifier la position de la barrière.

Barrière fermée	Barrière ouverte
	
90°	**180°**

1.3 Programmer un servomoteur

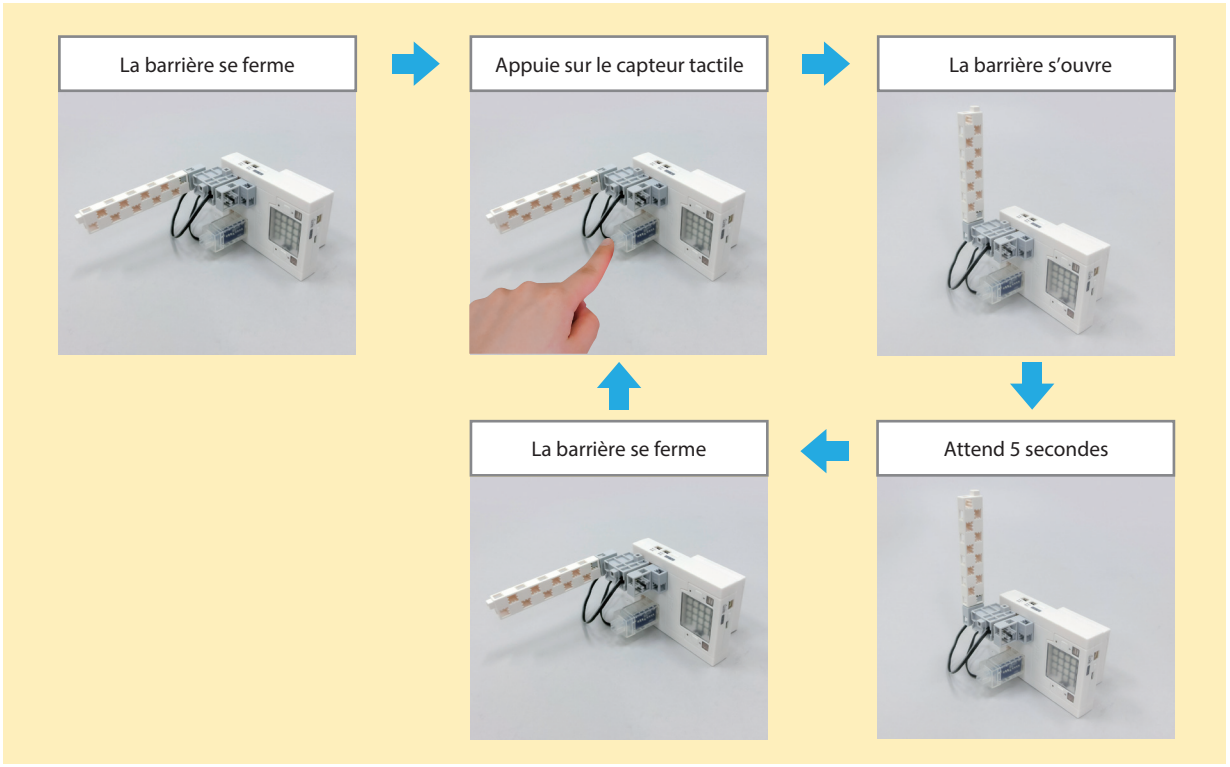
Fabriquons un programme qui fait tourner un servomoteur d'une position de 90 degrés à une position de 180 degrés.




```
1: from pyatcrobo2.parts import Servomotor
2: import time
3:
4: sv13 = Servomotor('P13')
5: sv13.set_angle(90)
6: time.sleep(1)
7: sv13.set_angle(180)
```

2. Création d'une barrière pousse-bouton

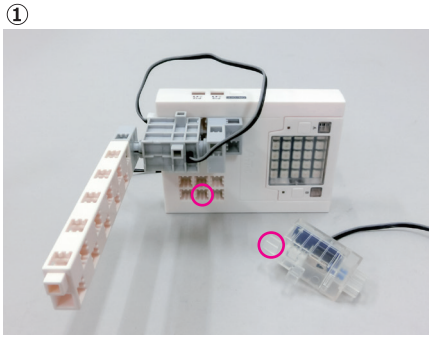
Ajoutons un capteur tactile pour créer une barrière contrôlée par un bouton.

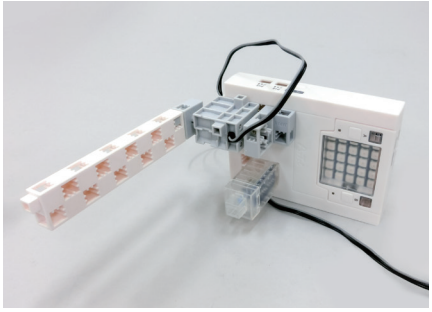


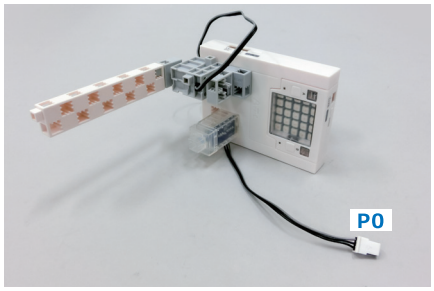
2.1 Construire une barrière pousse-bouton

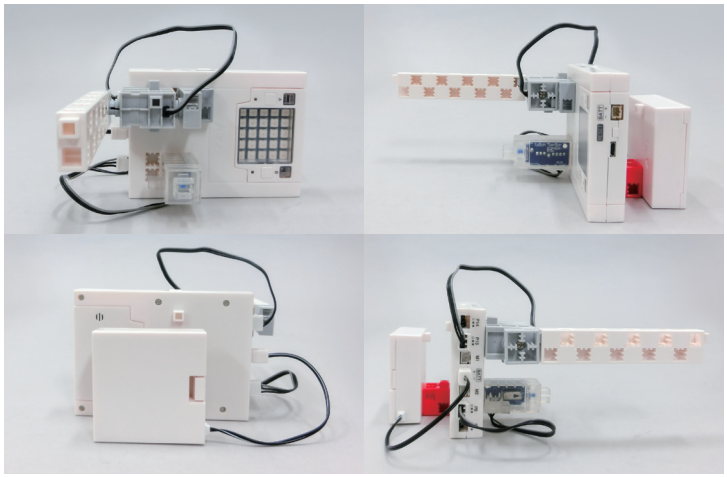
Ajoute un capteur tactile à ta barrière pour en faire une barrière pousse-bouton.

①









2.2 Les valeurs du capteur tactile

Voici un programme qui te permet de créer un objet pour contrôler des capteurs tactiles :

```
1: from pyatcrobo2.parts import TouchSensor
2:
3: capteur0 = TouchSensor('P0')
```

À la ligne 1, la commande **from pyatcrobo2.parts import TouchSensor** te permet d'utiliser la classe **TouchSensor** du module **pyatcrobo2.parts** de la bibliothèque ArtecRobo 2.0 (remarque : le nom de la classe est logique car *touch sensor* veut dire *capteur tactile* en anglais).
Tu peux consulter l'**Annexe B** pour en savoir plus sur la classe **TouchSensor**.

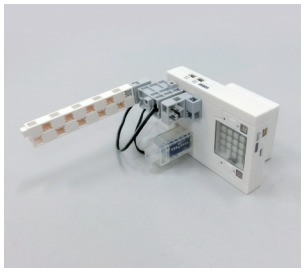
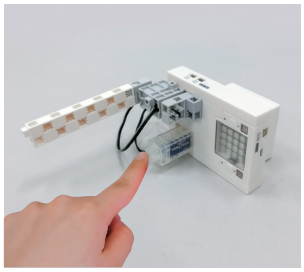
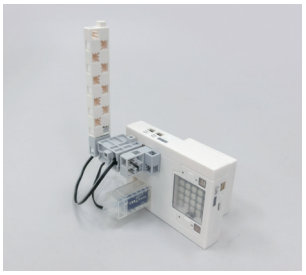
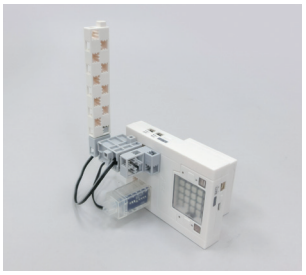
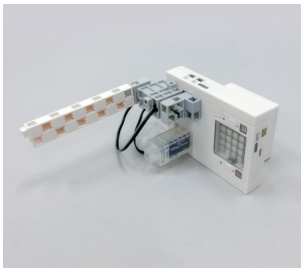
La ligne 3 permet de créer un objet que l'on appelle ici **capteur0** (mais on peut l'appeler comme on veut !) que tu peux utiliser pour contrôler le capteur tactile branché sur le port P0 de l'unité d'extension du robot.

Utiliser la méthode **get_value()** de la classe TouchSensor sur l'objet **capteur0** que l'on vient de créer (il suffira pour cela d'écrire **capteur0. get_value()**) permet de récupérer la valeur actuelle de cet objet. Cette valeur vaut soit 0, soit 1 : elle t'indique si le capteur tactile est actuellement pressé ou non, comme tu peux le voir sur le tableau ci-dessous.

Valeur de l'objet quand le capteur tactile a été pressé	Valeur de l'objet quand le capteur tactile N'a PAS été pressé
0	1

2.3 Comment fonctionne la barrière ?

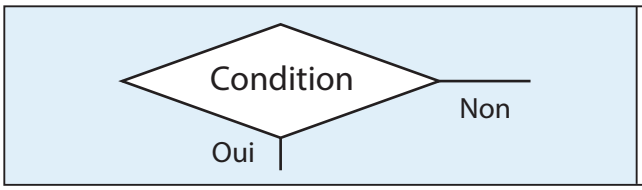
Pour savoir quelles actions nous devrons programmer, décomposons le fonctionnement de la barrière pousse-bouton en plusieurs étapes.

Séquence des étapes	Actions	Programme
①	La barrière se ferme 	Règle le servomoteur P13 à 90°
②	Appuie sur le capteur tactile 	La valeur du capteur tactile P0 = 0
③	La barrière s'ouvre 	Règle le servomoteur P13 à 180°
④	Attend 5 secondes 	Attend 5 secondes
⑤	La barrière se ferme 	Règle le servomoteur P13 à 90°

Répète ②-⑤ en boucle

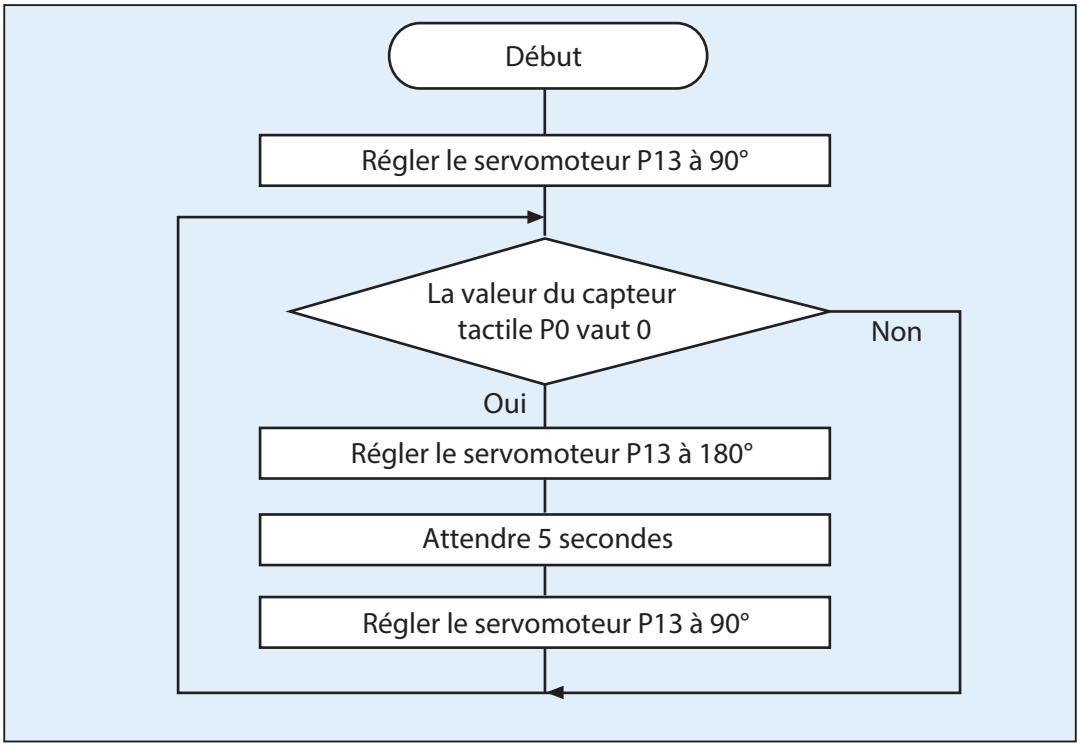
2.4 Construire un organigramme

Utilise un symbole comme celui présenté ci-dessous pour dessiner un branchement conditionnel dans un organigramme de programmation :



Divise le programme en deux branches suivant la condition.

Dessinons un organigramme pour la séquence d'actions décrite dans la section **2.3**.



Cet organigramme n'a pas de cellule « fin » parce que le programme ne s'arrête jamais tant que le capteur tactile continue d'être pressé.

2.5 Programmer la barrière pousse-bouton

Utilise ton organigramme de la section **2.4** pour écrire ton programme.

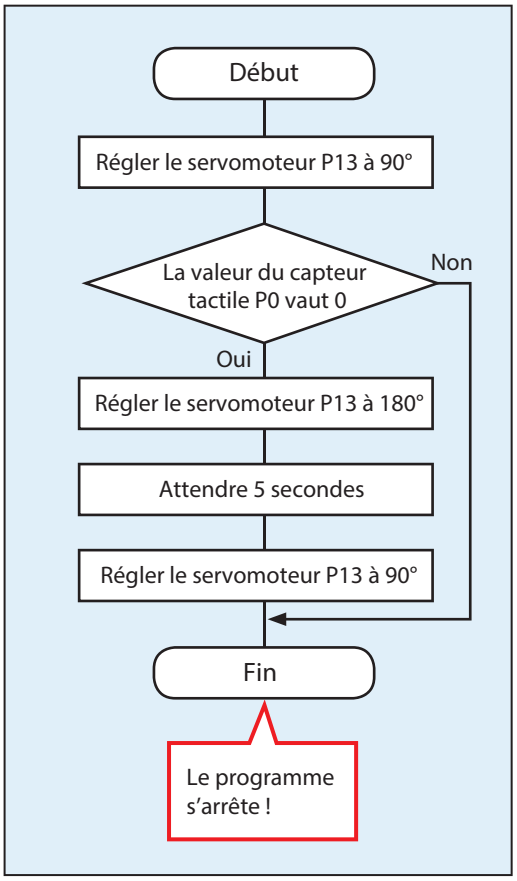
```
from pyatcrobo2.parts import Servomotor, TouchSensor
import time

sv13 = Servomotor('P13')
capteur0 = TouchSensor('P0')

sv13.set_angle(90)

while True:
    if capteur0.get_value() == 0:
        sv13.set_angle(180)
        time.sleep(5)
        sv13.set_angle(90)
```

Qu'arrive-t-il si l'on n'utilise pas une boucle **while** ? Regardons l'organigramme et le programme avec la boucle **while** en moins.

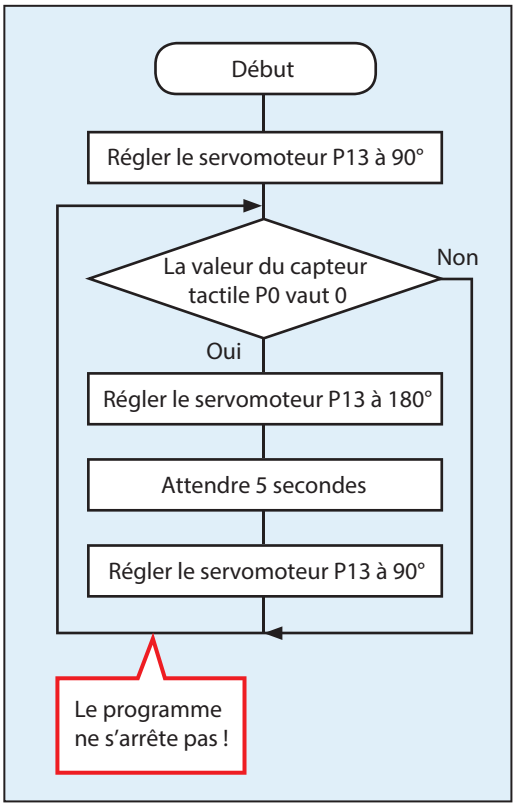



```
1: from pyatcrobo2.parts import Servomotor, TouchSensor
2: import time
3:
4: sv13 = Servomotor('P13')
5: capteur0 = TouchSensor('P0')
6:
7: sv13.set_angle(90)
8:

10:     sv13.set_angle(180)
11:     time.sleep(5)
12:     sv13.set_angle(90)
```

Si tu lances ton programme tel quel, la barrière ne fera rien du tout quand tu appuieras sur le capteur tactile. Cela est dû à l'extrême rapidité avec laquelle l'unité centrale exécute ton programme - le programme aura fini de tourner presque immédiatement après que tu l'as lancé !

C'est pourquoi tu as besoin d'une boucle **while** pour faire en sorte que le programme n'arrête jamais de vérifier si le capteur tactile a été pressé. Le programme fait alors bien ce qu'on voulait qu'il fasse.

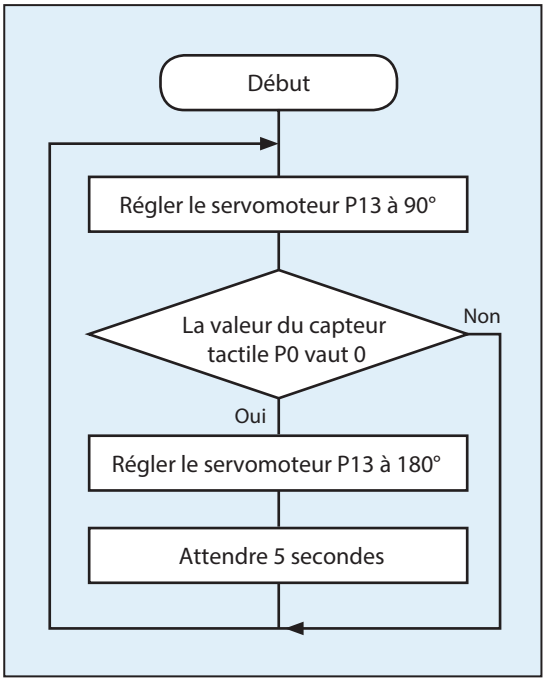



```
1: from pyatcrobo2.parts import Servomotor, TouchSensor
2: import time
3:
4: sv13 = Servomotor('P13')
5: capteur0 = TouchSensor('P0')
6:
7: sv13.set_angle(90)
8:
9:   while True:
10:     if capteur0.get_value() == 0:
11:         sv13.set_angle(180)
12:         time.sleep(5)
13:         sv13.set_angle(90)
```

Les ordinateurs ont pour mission de faire exactement ce que le programme leur dit. Les petites erreurs dans les programmes amènent donc souvent les ordinateurs à faire une chose qu'on ne voulait pas ! Planifier tes programmes avec un organigramme au préalable peut t'aider à éviter des problèmes comme celui-ci.

2.5.1 Une barrière pousse-bouton alternative

Tu peux aussi fabriquer une barrière pousse-bouton en utilisant un programme comme celui-ci :



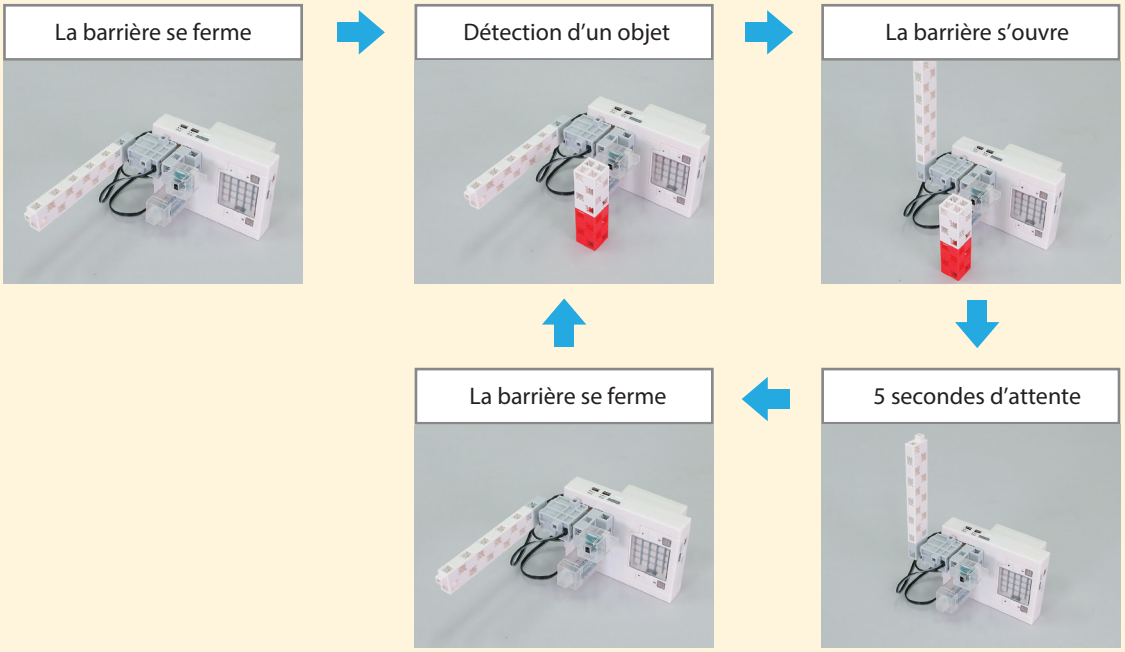

```
1: from pyatcrobo2.parts import Servomotor, TouchSensor
2: import time
3:
4: sv13 = Servomotor('P13')
5: capteur0 = TouchSensor('P0')
6:
7: while True:
8:     sv13.set_angle(90)
9:     if capteur0.get_value() == 0:
10:        sv13.set_angle(180)
11:        time.sleep(5)
```

Comme tu peux le voir, il y a plus d'une façon d'écrire un programme qui fonctionne !
Tu peux écrire ton programme comme tu veux, tant qu'il te fournit les résultats dont tu as besoin à la fin.

Réfléchis au problème toi-même pour trouver un programme qui te convient.

3. Créer une barrière activée par un réflecteur

Fabriquons une barrière qui utilise un photoréflecteur infrarouge (IR Photoreflector) pour s'ouvrir automatiquement quand il détecte une présence !



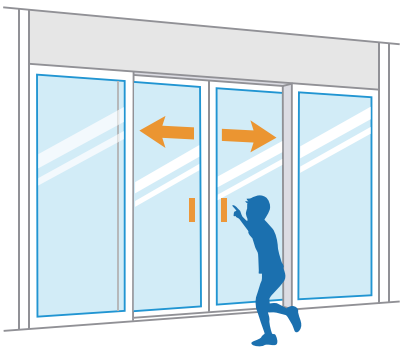
Le programme de cette barrière suit le même processus de base que la barrière pousse-bouton, mais avec un photoréflecteur infrarouge à la place d'un capteur tactile.

Différents types de portes automatiques

Tu as peut-être remarqué que les portes automatiques fonctionnent parfois de manière différente. Il y a principalement les portes dites à contact et les portes à réflecteur.

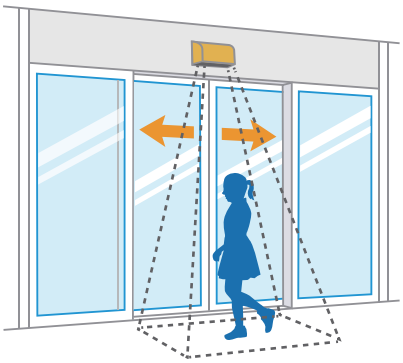
Les portes à contact utilisent des capteurs tactiles, tandis que les portes à réflecteur utilisent des photoréflecteurs infrarouges.

Portes à contact



Quand tu appuies sur le bouton, ces portes s'ouvrent directement !

Portes à réflecteur

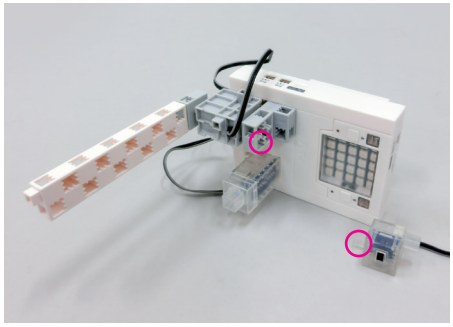


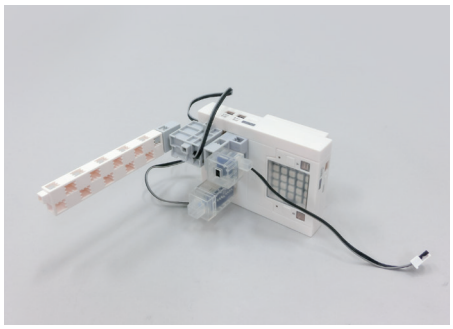
Le capteur au-dessus des portes émet de la lumière et les portes s'ouvrent quand un objet reflète cette lumière !

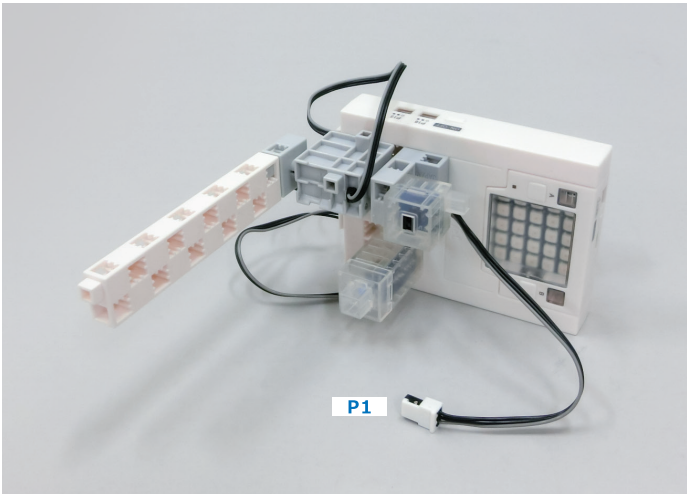
3.1 Construire une barrière activée par un réflecteur

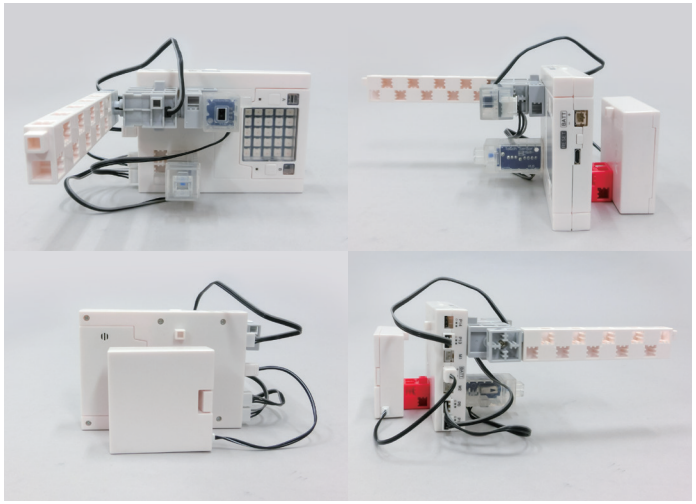
Ajoute un photoréflecteur infrarouge à ta barrière pour faire une barrière activée par un réflecteur !

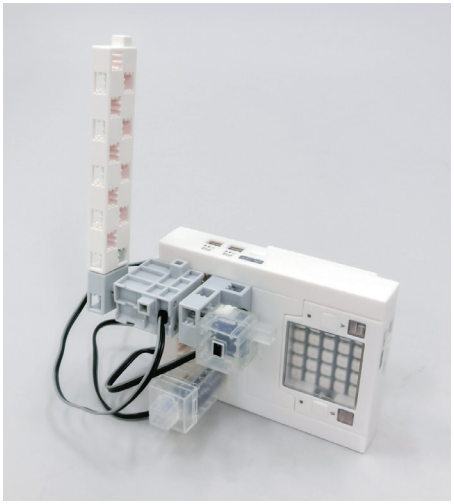
①











3.2 Tester les valeurs de ton photoréflecteur infrarouge

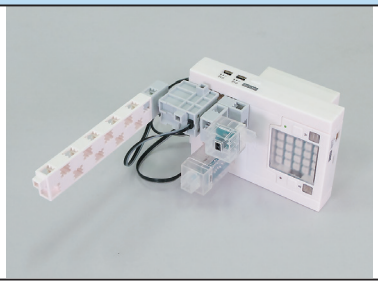
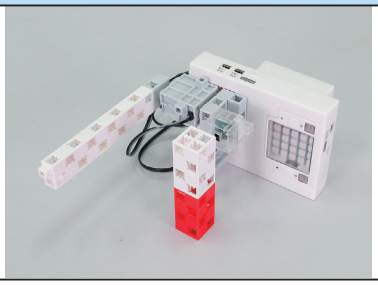
Les portes automatiques s'ouvrent quand elles détectent une présence devant elles grâce aux photoréflecteurs infrarouges.

Observons les valeurs de ton photoréflecteur infrarouge à l'aide de la classe IRPhotoReflector et en utilisant le même type de programme que celui utilisé à la section 3.3, page 49, du manuel *Les bases de la programmation - Mouvement et détection*.

```
1: from pyatcrobo2.parts import IRPhotoReflector
2: import time
3:
4: irp = IRPhotoReflector('P1')
5: while True:
6:     value = irp.get_value()
7:     print(str(value))
8:     time.sleep(1)
```

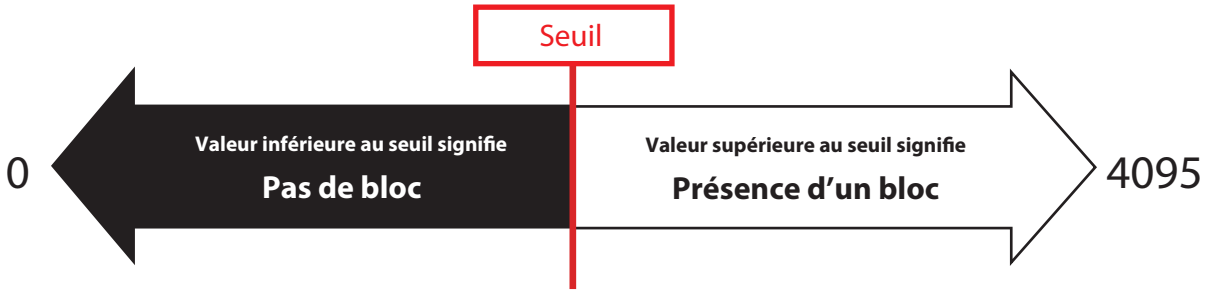
Observe la façon dont changent les valeurs de ton photoréflecteur infrarouge lorsque tu places un bloc devant lui et lorsque tu le retires.

★ Vérifie que le photoréflecteur infrarouge n'est pas exposé directement aux rayons du soleil, afin que les propres rayons infrarouges du soleil n'influencent pas tes résultats.

Valeur sans le bloc blanc	Valeur avec le bloc blanc devant
	

3.3 Trouver un seuil

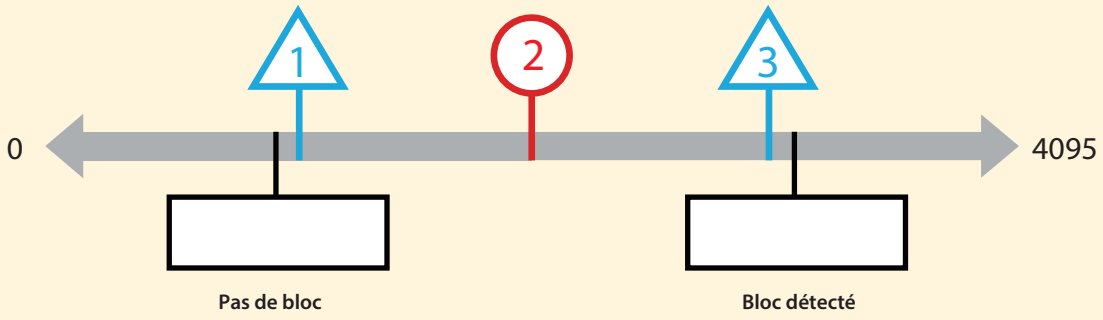
Pour que ton photoréflecteur infrarouge puisse savoir si un objet est devant lui ou non, il a besoin d'une valeur particulière qui marque la limite entre ces deux états et lui permette de faire la différence. Cette valeur est appelée un seuil.



Comment choisir un seuil ?

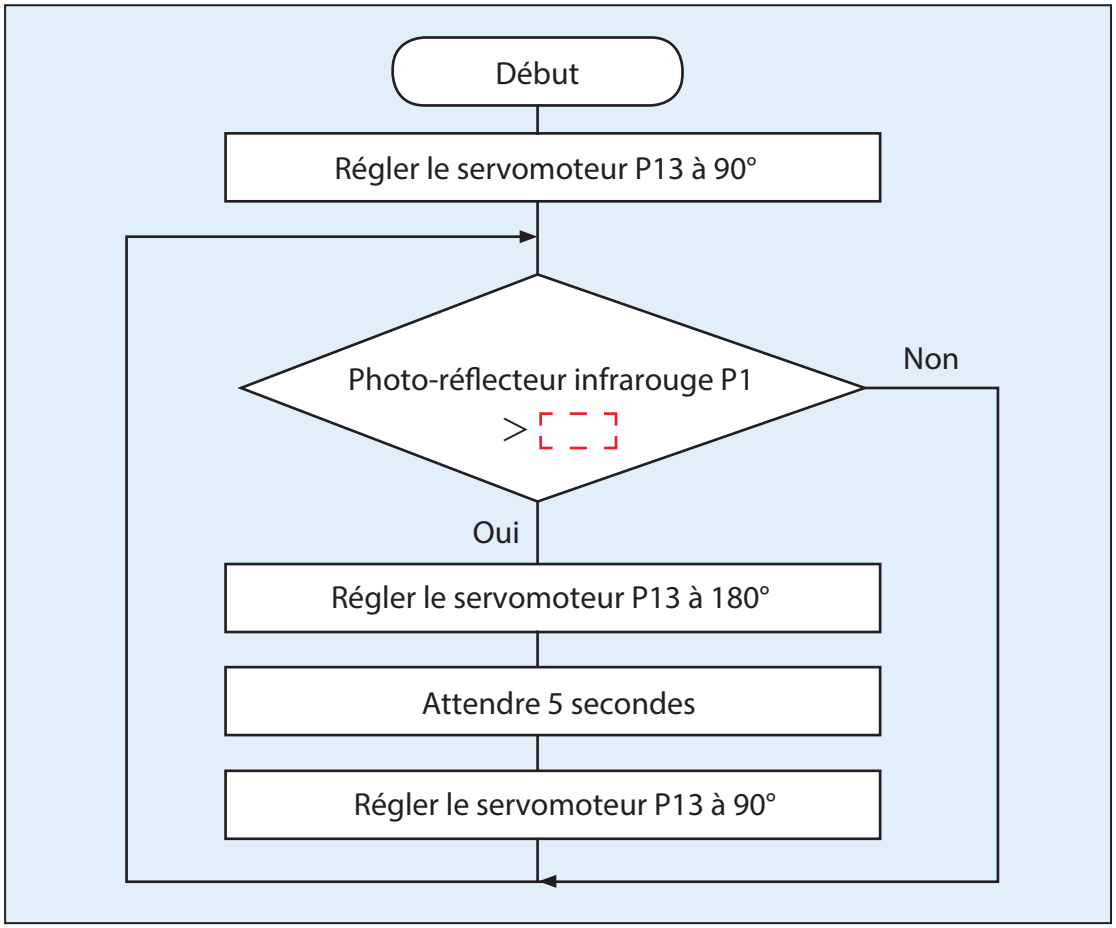
Si tu choisis pour le seuil une valeur plus proche d'une des extrémités de l'échelle que de l'autre, comme le point 1 ou 3 ci-dessous, de faibles variations sur le capteur pourraient entraîner ton programme à détecter des choses qui ne sont pas là !

Comme de légers changements dans l'environnement des capteurs peuvent facilement changer leurs valeurs, il vaut mieux choisir une valeur qui soit à mi-chemin entre les valeurs des deux états que tu essaies de différencier, comme le point 2 ci-dessous.



3.4 Élaborer un organigramme

Schématisons d'abord les étapes qui serviront à faire fonctionner ta barrière à réflecteur pour savoir exactement quelles actions programmer. Cet organigramme de programmation sera globalement le même que celui de la barrière pousse-bouton, puisque seul le capteur changera - le capteur tactile sera remplacé par un photoréflecteur infrarouge. Tu peux inscrire la valeur de seuil que tu as choisie dans l'encadré rouge de la cellule conditionnelle.



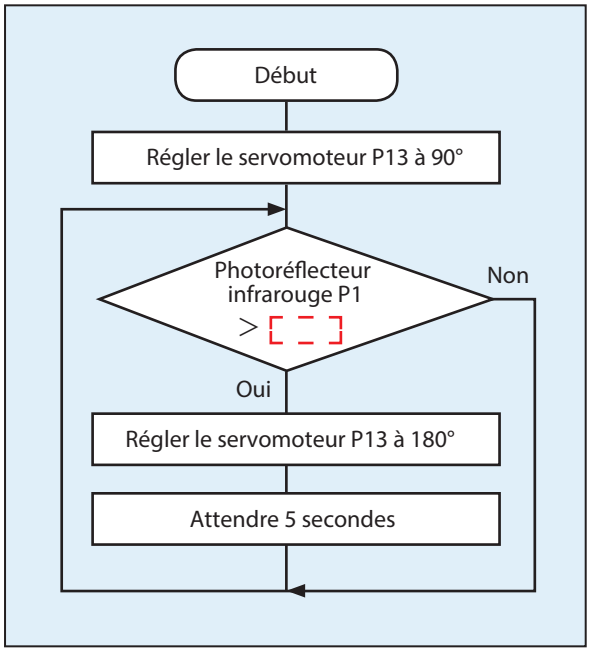
3.5 Programmons la barrière activée par un réflecteur

Utilise ton organigramme de la section **3.4** pour écrire ton programme !

```
1: from pyatcrobo2.parts import Servomotor, IRPhotoReflector
2: import time
3:
4: sv13 = Servomotor('P13')
5: irp = IRPhotoReflector('P1')
6:
7: sv13.set_angle(90)
8: while True:
9:     if irp.get_value() > [ ]:
10:         sv13.set_angle(180)
11:         time.sleep(5)
12:         sv13.set_angle(90)
```

3.5.1 Une barrière activée par un réflecteur alternative

Tu peux aussi créer une barrière activée par un réflecteur en utilisant un programme comme celui-ci :




```
1: from pyatcrobo2.parts import Servomotor, IRPhotoReflector
2: import time
3:
4: sv13 = Servomotor('P13')
5: irp = IRPhotoReflector('P1')
6:
7: while True:
8:     sv13.set_angle(90)
9:     if irp.get_value() > [ ]:
10:        sv13.set_angle(180)
11:        time.sleep(5)
```

4. Techniques avancées

Voyons, à présent, comment fabriquer une branche d'un programme basée sur plusieurs conditions et comment stocker les données dans des variables. Représente-toi la porte automatique que tu viens de faire. Quel type de problèmes pourraient se poser si tu l'utilisais ? Comment pourrais-tu l'améliorer pour résoudre ces problèmes ?

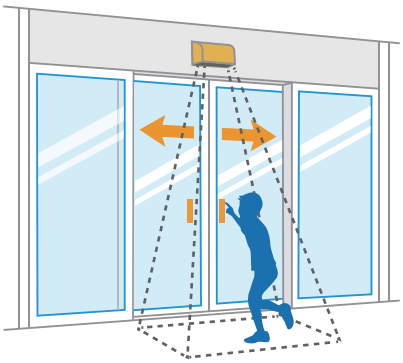
Exemple de problèmes

- La barrière pourrait s'ouvrir même si personne n'est devant, si le photoréflecteur infrarouge détecte la lumière du soleil ou d'autres objets.
- La barrière pourrait percuter quelqu'un lorsqu'elle se ferme.

Solutions

Examinons un exemple de programme basé sur des astuces que les vraies portes automatiques utilisent pour régler ces problèmes.

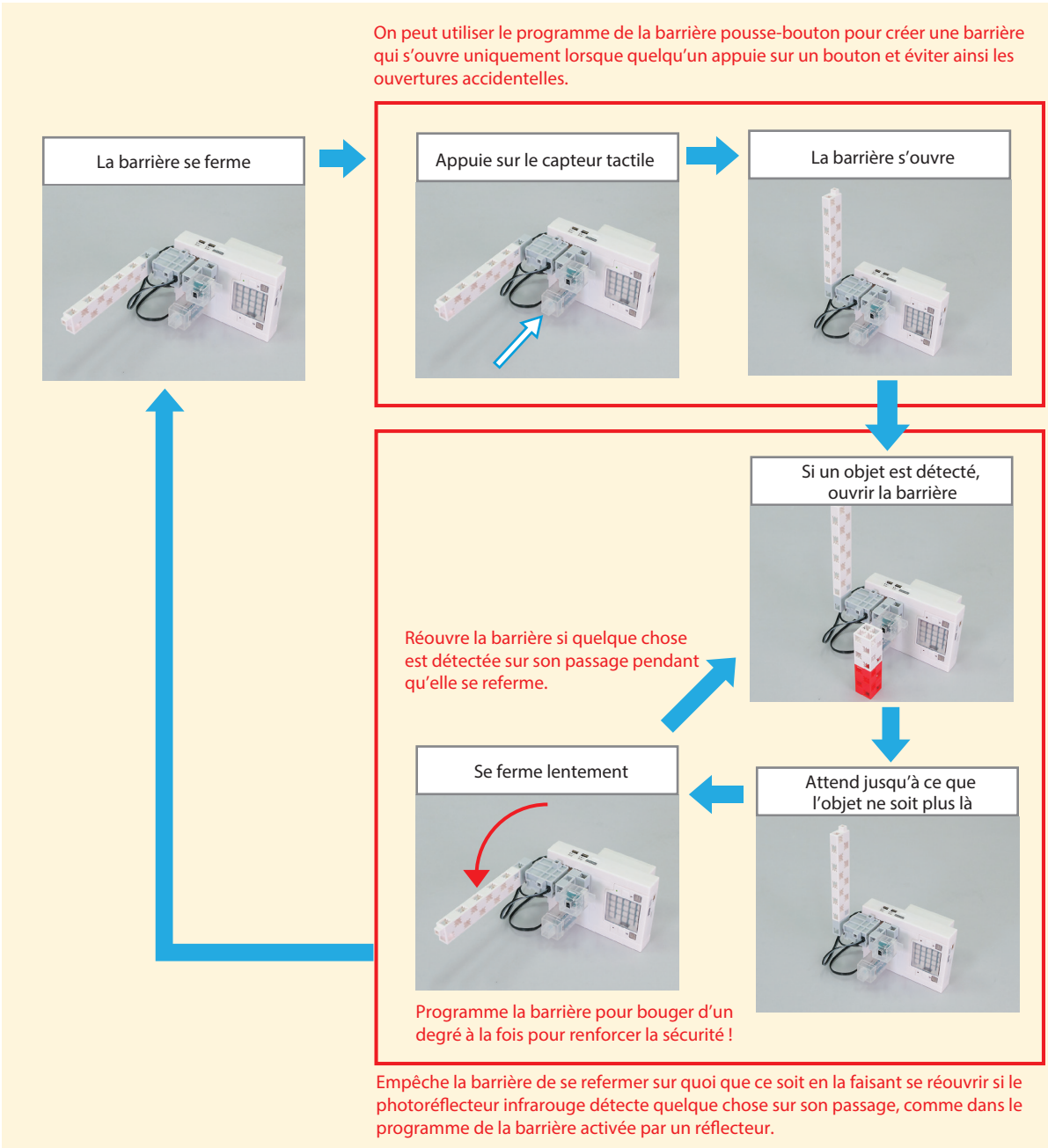
Portes à contact/réflecteur



Combine les deux types de portes pour résoudre le problème !

Mesure(s) préventive(s) : Ne s'ouvre que si quelqu'un appuie sur le bouton.

Points concernant la sécurité :
- Utilise un capteur pour détecter si des personnes sont sur le pas de la porte pour ne pas se refermer sur elles.
- Détecte les personnes en train de passer quand la porte est en train de se refermer pour la faire se réouvrir automatiquement.



Pour faire fonctionner la barrière ainsi, il faut que tu fasses en sorte que la branche du programme dépende de la position courante de la barrière !

La barrière est fermée (l'angle du servomoteur est de 90°) ⟶ Ne détecte pas les objets.

La barrière est ouverte (l'angle du servomoteur est supérieur à 90°) ⟶ Détecte les objets.

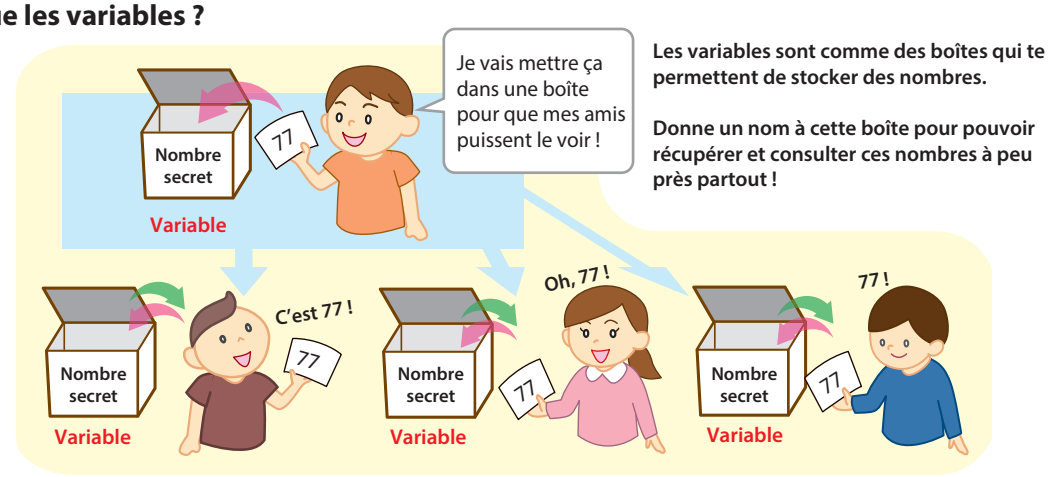
À présent, tu vas devoir utiliser une **variable** pour faire un programme dont les branches dépendent de la position de la barrière (ou plutôt de l'angle du servomoteur).

4.1 Qu'est-ce qu'une variable ?

Les variables servent à stocker des données ou des résultats de calcul pour que tu puisses les utiliser plus tard. En Python, tu peux créer une variable juste en tapant le nom de la variable suivi du symbole = et de la valeur !

Qu'est-ce que les variables ?

Visualisation des variables



Dans le cas de notre programme de barrière automatique, nous allons créer une variable que nous nommerons **barriere** comme montré ci-dessous.

N.B : on évite, en général, de mettre des accents dans les noms de variables. On peut bien sûr appeler notre variable comme on veut, même s'il vaut mieux lui donner un nom qui a un sens, pour mieux comprendre ce que l'on fait.

```
barriere = 90
if barriere == 90:
# Cette branche est exécutée si la barrière est fermée !
else:
# Cette branche est exécutée si la barrière est ouverte !
```

Grâce à cette variable dans laquelle tu enregistres la position de la barrière, tu peux utiliser une structure conditionnelle **if-else** pour diviser ton programme en deux branches et choisir celle qu'il faut utiliser suivant que la barrière est ouverte ou fermée.

4.2 Programme complet

```
1:  from pyatcrobo2.parts import Servomotor, TouchSensor, IRPhotoReflector
2:  import time
3:
4:  sv13 = Servomotor('P13')
5:  capteur0 = TouchSensor('P0')
6:  irp = IRPhotoReflector('P1')
7:
8:  barriere = 90
9:
10: while True:
11:     sv13.set_angle(barriere)
12:     if  barriere == 90:
13:         if  capteur0.get_value() == 0:
14:             barriere = 180
15:     else:
16:         if  irp.get_value() <  :
17:             barriere -= 1
18:             time.sleep(0.1)
19:         else:
20:             barriere = 180
21:
22:     if  barriere < 90:
23:         barriere = 90
```

- À la ligne 8, on affecte à la variable **barriere** la valeur initiale de **90**.
- À la ligne 10, le code **while True** crée une boucle infinie.
- À la ligne 11, l'utilisation d'une variable pour régler l'angle du servomoteur fait que le servomoteur tourne à l'angle qui est égal à la valeur stockée dans la variable à ce moment-là.
- Les lignes 12 et 15 mettent en place les conditions qui permettent de vérifier si la valeur de la variable **barriere** vaut **90**. Ces conditions vous permettent de créer des branches de programme séparées qui vont s'exécuter suivant que la barrière est ouverte ou fermée.
- Les lignes 12 à 14 sont exécutées si la barrière est fermée. Elles assignent à la variable **barriere** la valeur de **180** pour ouvrir la barrière si le capteur tactile a été pressé.
- Les lignes 15 à 20 sont exécutées si la barrière n'est pas fermée - qu'elle ne soit pas encore complètement fermée ou qu'elle soit ouverte. Cette partie du programme se divise à nouveau en deux branches selon que le photoréflecteur infrarouge détecte ou non quelque chose devant lui. La ligne 17 permet de diminuer la valeur de la variable **barriere** de 1 degré à la fois pour refermer lentement la barrière tant qu'aucun objet n'est détecté dans le voisinage. La ligne 17 utilise un **opérateur** d'affectation sur lequel tu peux en apprendre plus à l'**Annexe C**.
- Plus le paramètre que tu mets dans **time.sleep()** à la ligne 18 est grand (ici il vaut 0.1), plus la barrière va se refermer lentement. Si le capteur détecte quelque chose devant la barrière avant que la barrière ne se soit refermée, on assigne à la variable **barriere** la valeur de **180** pour que la barrière s'ouvre complètement à nouveau.
- Les lignes 22 et 23 permettent de s'assurer que le code de la ligne 17 n'amènera pas la variable **barriere** en dessous de **90**, en la remettant à **90** si elle descend en dessous.

Annexe A. Les méthodes de la classe Servomotor

Méthodes	Ce qu'elle fait
__init__(port)	Cette méthode est appelée un constructeur. Elle permet de créer un objet de cette classe (un tel objet est alors appelé une instance de la classe.) Le paramètre, symbolisé ici par **port**, peut être égal à **P13**, **P14**, **P15** ou **P16** et permet de spécifier que le servomoteur est branché sur le port P13, P14, P15 ou P16 de l'unité d'extension du robot.
set_angle(degre)	Le paramètre, symbolisé ici par **degre**, contrôle l'angle du servomoteur et peut être choisi entre **0** et **180**.
release()	Studuino:bit utilise un signal de MLI (Modulation de largeur d'impulsion) pour régler les angles du servomoteur. Il peut utiliser jusqu'à quatre MLI simultanément. Si tu utilises 5 ou plus parties qui nécessitent des MLI, tu ne pourras utiliser aucun des objets de la classe Servomotor que tu as créée. Tu peux résoudre ce problème en utilisant la méthode **release()** pour libérer une MLI jusqu'alors assignée à un autre objet de la classe Servomotor.

Annexe B. Les méthodes de la classe TouchSensor

Méthodes	Ce qu'elle fait
__init__(port)	Cette méthode est appelée un constructeur. Elle permet de créer un objet de cette classe (un tel objet est alors appelé une instance de la classe). Le paramètre, symbolisé ici par **port**, peut être égal à **P0**, **P1** ou **P2** et permet de spécifier que le capteur est branché sur le port P0, P1, ou P2 de l'unité d'extension du robot.
get_value()	Renvoie **0** si le capteur tactile a été pressé et **1** sinon.
is_pressed()	Renvoie **True** si le capteur tactile a été pressé et **False** sinon.

Annexe C. Opérateurs d'affectation

Si une valeur a déjà été affectée à une variable, cette valeur peut être remplacée par le résultat d'un calcul utilisant cette même variable.

Pour remplacer la valeur d'une variable, écris dans l'ordre les équations présentées dans le tableau ci-dessous. Le programme détermine d'abord la valeur du côté droit, puis utilise cette valeur dans l'opération décrite par l'opérateur d'affectation pour affecter une nouvelle valeur au côté gauche.

Côté gauche **Opérateur d'affectation** **Côté droit**

Opérateur d'affectation	Opération	Ex.	Signification
=	Affectation	a=b	Affecte la valeur b à la variable a
+=	Addition	a+=b	Affecte la valeur a+b à la variable a
-=	Soustraction	a-=b	Affecte la valeur a-b à la variable a
*=	Multiplication	a*=b	Affecte la valeur a x b à la variable a
/=	Division	a/=b	Affecte la valeur a÷b à la variable a
%=	Modulo	a%=b	Affecte la valeur du reste de a÷b à la variable a

Notes

Ecole Algora

Apprendre à coder en s'amusant

PLUS DE 36 ROBOTS DIFFÉRENTS
UN CURSUS COMPLET
POUR APPRENDRE À PROGRAMMER

2 Cursus
6-9 ans
9-14 ans

Inscris-toi sur www.algora.school

Apprendre à programmer des robots pour comprendre le monde d'aujourd'hui et de demain.

Les machines programmées, de plus en plus intelligentes, font partie intégrante de notre vie de tous les jours. Elles nous accompagnent, nous entourent et ont envahi tous les domaines de notre vie quotidienne. Maîtriser le monde, ce n'est pas les utiliser, mais avant tout comprendre comment elles fonctionnent.

Comment fonctionnent-elles ?
Selon quelle logique ? Selon quels algorithmes ?
Comment sont conçus les programmes qui leur dictent leurs actions et réactions ?

C'est ce que vous apprendrez tout au long de ces livrets d'apprentissage. Et pas seulement "en théorie" : vous allez vous-même concevoir et programmer vos propres robots : des actions simples aux plus complexes, vous apprendrez à programmer des robots amusants et originaux que vous aurez conçus vous-même. Une seule limite : votre créativité !

École Robots permet à tous de s'initier à la programmation en s'amusant, un enjeu majeur, aujourd'hui et demain.

Ecole Robots
Apprendre à coder avec les robots éducatifs
CURSUS EDUCATION NATIONALE

Pour en savoir plus : www.ecolerobots.com

Réf : LIV-EN-C-PY2 (v.1)