Micro:bit lampadaire intelligent

TP SNT
« Lampadaire intelligent »

Introduction

Capacités attendues

- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs, l'IHM et les actions des actionneurs dans des systèmes courants.
- Réaliser une IHM simple d'un objet connecté.
- Écrire des programmes simples d'acquisition de données ou de commande d'un actionneur.

Selon L'ADEME (Agence de l'Environnement et de la Maîtrise de l'Energie) l'éclairage public représente 41% des consommations d'électricité des collectivités territoriales, 16% de leurs consommations toutes énergies confondues et 31% de leur facture d'électricité.
Le potentiel de réduction des consommations est énorme : Plus de la moitié du parc de luminaires urbains est obsolète, près de 40% des luminaires en services ont plus de 20 ans. Pour les collectivités un investissement dans des technologies novatrices permettrait une réduction des coûts de dépenses énergétiques, d'exploitation et de maintenance. www.ademe.fr Éclairage public : un gisement d'économies d'énergie

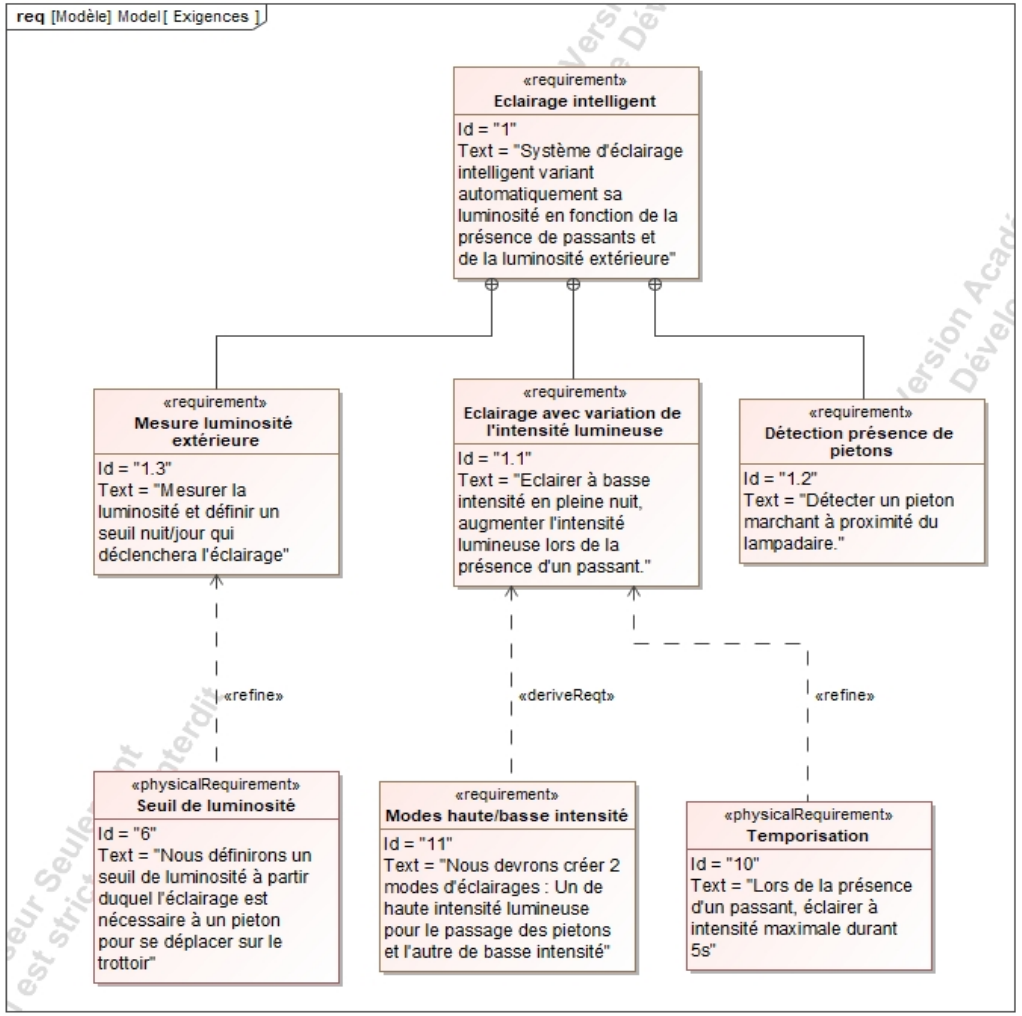
Le lampadaire Kara

La société Toulousaine ***Kawantech*** a inventé un lampadaire intelligent appelé ***Kara***, ce dernier éclaire à son maximum seulement quand c'est nécessaire : Quand il fait nuit et lors du passage d'un piéton à proximité. Le système *Kara* s'installe sur un lampadaire LED existant et le transforme en lampadaire intelligent.
Selon *Kawantech* : « Les capteurs Kara fonctionnent en réseau. Installés sur chaque luminaire d'une zone, ils analysent les objets qui se déplacent dans la rue et communiquent entre eux pour optimiser l'éclairage public. Le calculateur de Kara, qui pilote la puissance des LEDs, peut discerner une voiture, un piéton ou simplement un mouvement de branche d'arbre. » www.kawantech.com

Visualisez la vidéo Youtube : Kara Smart Lighting sensor

Diagramme des exigences

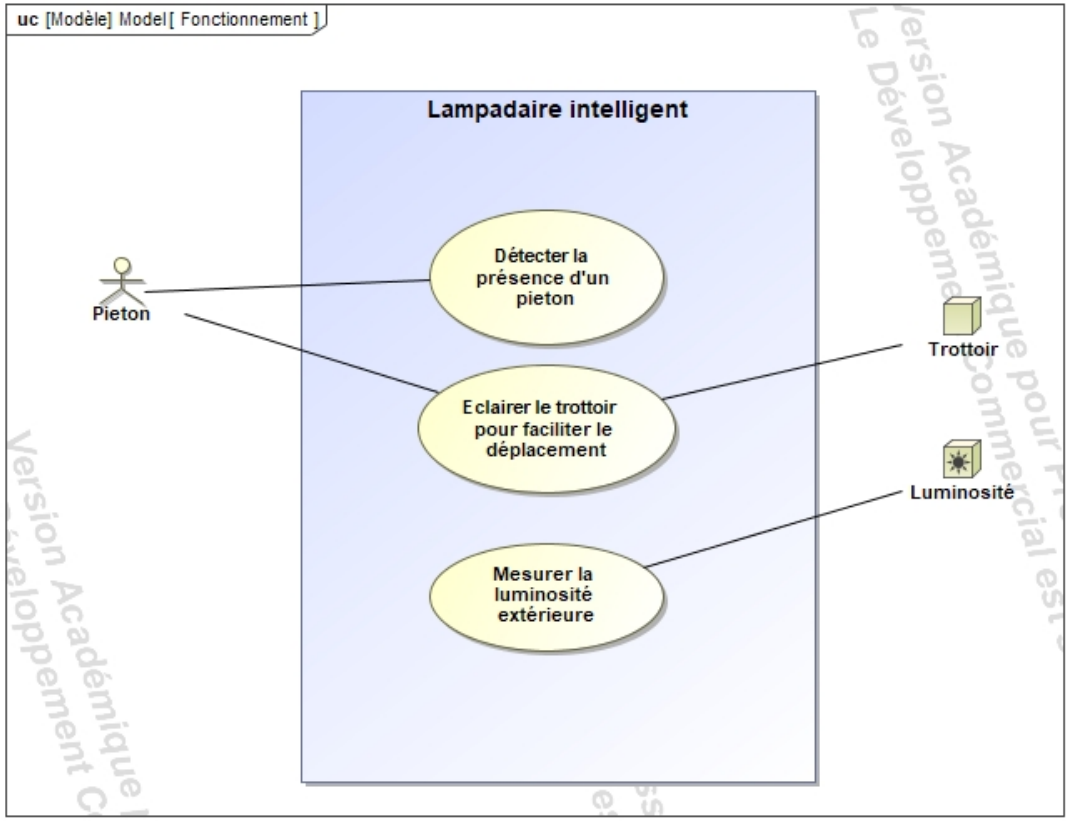
Nous proposons de réaliser notre propre système d'éclairage intelligent inspiré du lampadaire Kara, dont nous donnons les exigences ci-dessous, chaque bloc représente un critère obligatoire au système.



1. Sur le diagramme des exigences entourez en **vert** les exigences liées à la **mesure de la luminosité**
2. Entourez en **rouge** les exigences liées à l'**éclairage**
3. Puis en **bleu** celles associées à la **détection des piétons**

Diagramme des cas d’utilisation

Le diagramme des cas d’utilisation présente les interactions entre le système défini par le cadre violet et l’environnement extérieur : Piéton, Trottoir, Luminosité.

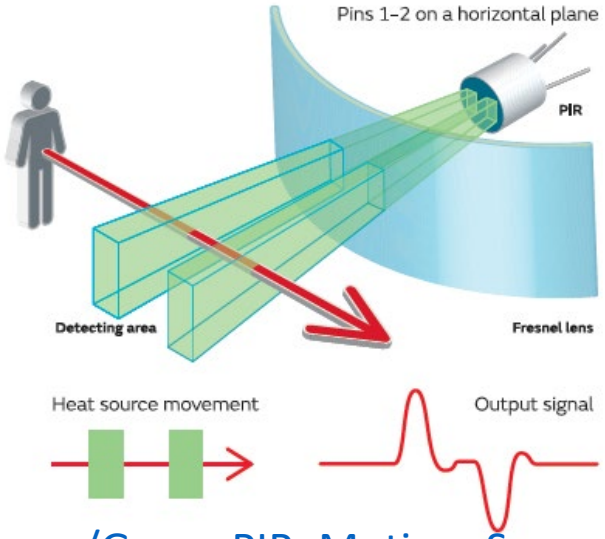


1. Sur le diagramme des cas d’utilisations entourez en **vert** les cas d’utilisations liées à la **mesure de la luminosité**
2. Entourez en **rouge** les cas d’utilisations liées à l’**éclairage**
3. Puis en **bleu** ceux associées à la **détection des piétons**

Présentation des capteurs utilisés

Capteur de présence PIR
PIR pour Passive Infra Red ou capteur infrarouge passif. Ce type de capteur détecte les radiations infra-rouges (chaleur) émises par tous les êtres vivants (humains, animaux)
Quand une personne passe devant une zone de détection ses radiations **infrarouges** sont détectées par le capteur et transformées en un signal électrique.

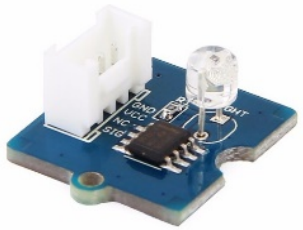
Variable associée = presence_pieton
presence_pieton =1 quand détection d'une personne
sinon presence_pieton =0
Câblage choisi sur la grove shield => pin 1

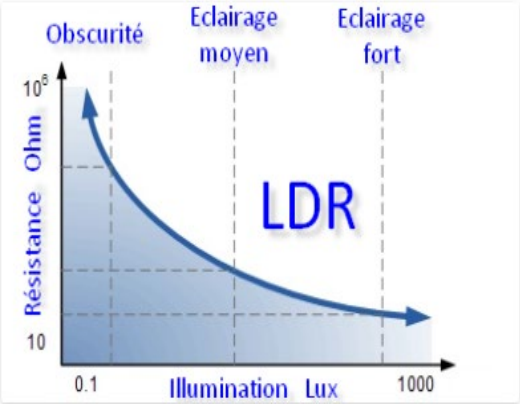


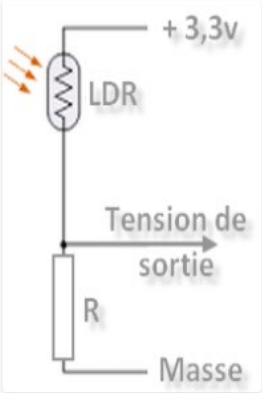
http://wiki.seeedstudio.com/Grove-PIR_Motion_Sensor/

Capteur de luminosité LDR
LDR pour Light Dependent Resistor ou résistance dépendant de la lumière. Ce composant électronique a un de ses paramètres physiques appelé résistivité qui varie en fonction de la quantité de lumière qu'il reçoit.
Dans l'obscurité, la résistance d'une LDR est proche de 1 MΩ. Avec un éclairage intense, la résistance chute fortement (quelques KΩ). Un pont diviseur permet de récupérer une tension qui sera directement le reflet de la lumière arrivant sur la LDR.

Variable associée = Lum
lum=0 quand la luminosité est totale et augmente jusqu'à 1023 en pleine lumière
Câblage choisi sur la grove shield => pin 0



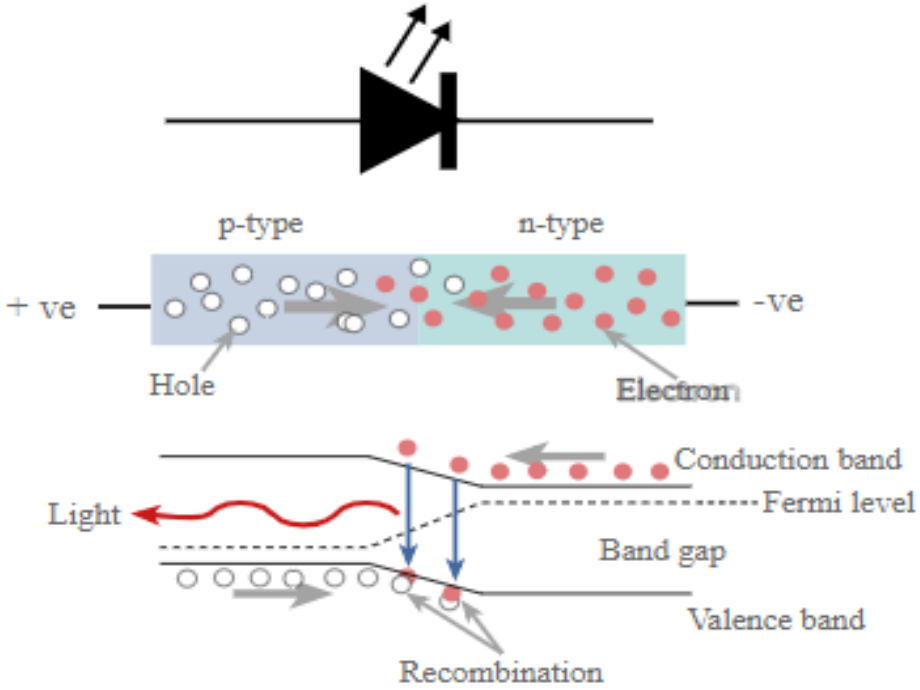




Présentation de l’actionneur utilisé

Diode Électroluminescente
Led pour Light Emitting Diode ou diode electro-luminescente. Cet actionneur transforme l’énergie électrique qu’il reçoit en énergie lumineuse.
Les leds très économes en énergie, appartiennent à la famille des semi-conducteurs. Les électrons (éléments lumineux) transitent de la zone N vers la zone P attirés par les trous qu’ils doivent remplir. Le déplacement d’énergie durant cette transition produit de la lumière.

Variable associée = alpha
alpha=1023 pour allumer la led au maximum, 0 pour l’éteindre
Câblage choisi sur la grove shield => pin 2



Bilan du cahier des charges

	Exigences	Cas d'utilisations	Matériel
Détection piétons			
Mesure luminosité			
Eclairage			

En vous aidant des diagrammes précédents remplissez le tableau.

Bilan du cahier des charges

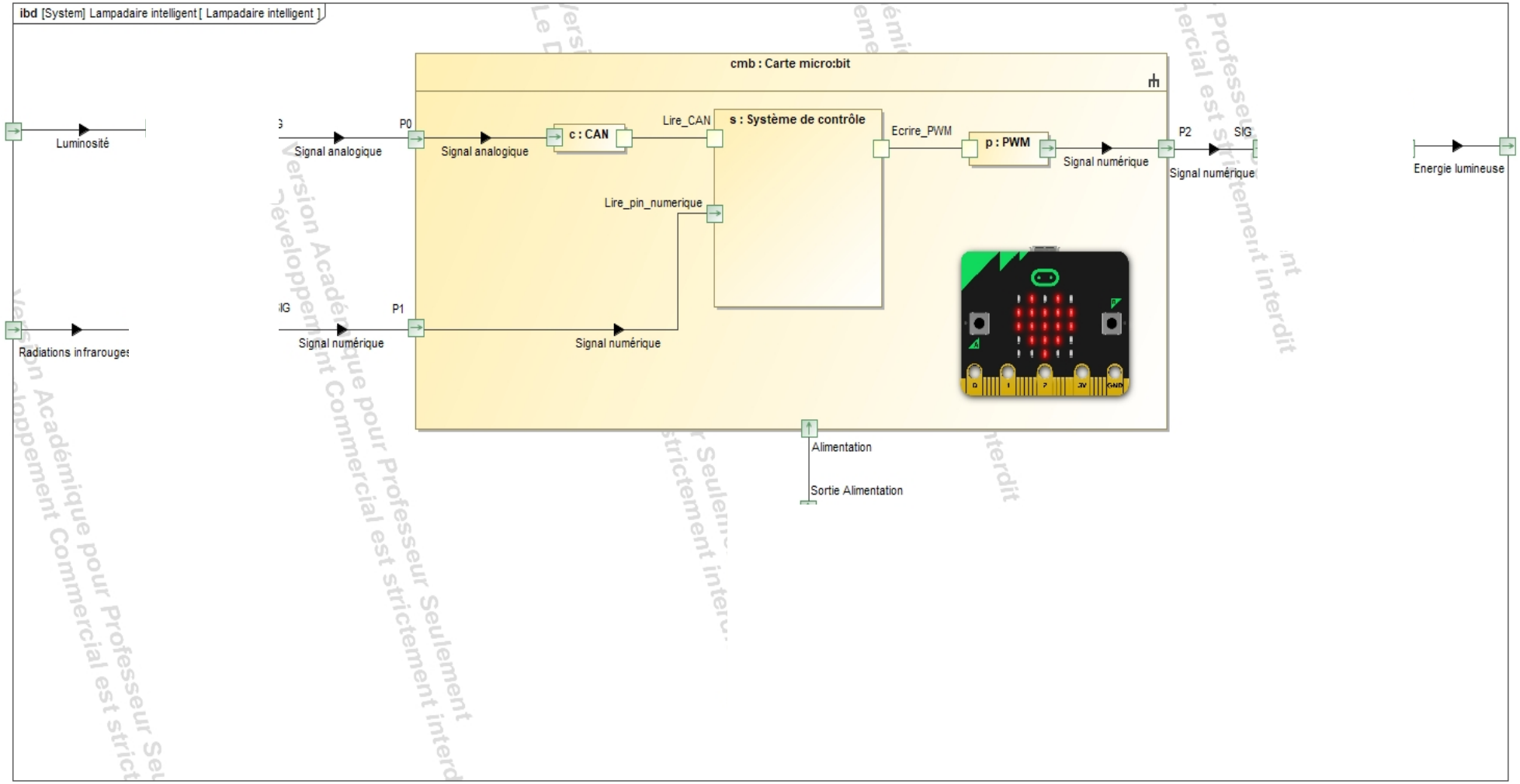
CORRECTION

En vous aidant des diagrammes précédents remplissez le tableau.

	Exigences	Cas d'utilisations	Matériel
Détection piétons	**Détection présence de piétions**	**Détecter la présence d'un piéton**	**Capteur PIR**
Mesure luminosité	**Mesure luminosité extérieure** **Seuil de luminosité**	**Mesurer la luminosité extérieure**	**Capteur LDR**
Eclairage	**Eclairage avec variation de l'intensité lumineuse** **Modes haute/basse intensité** **Temporisation**	**Eclairer le trottoir pour faciliter le déplacement**	**LED**

Chaines d’énergie et d’information

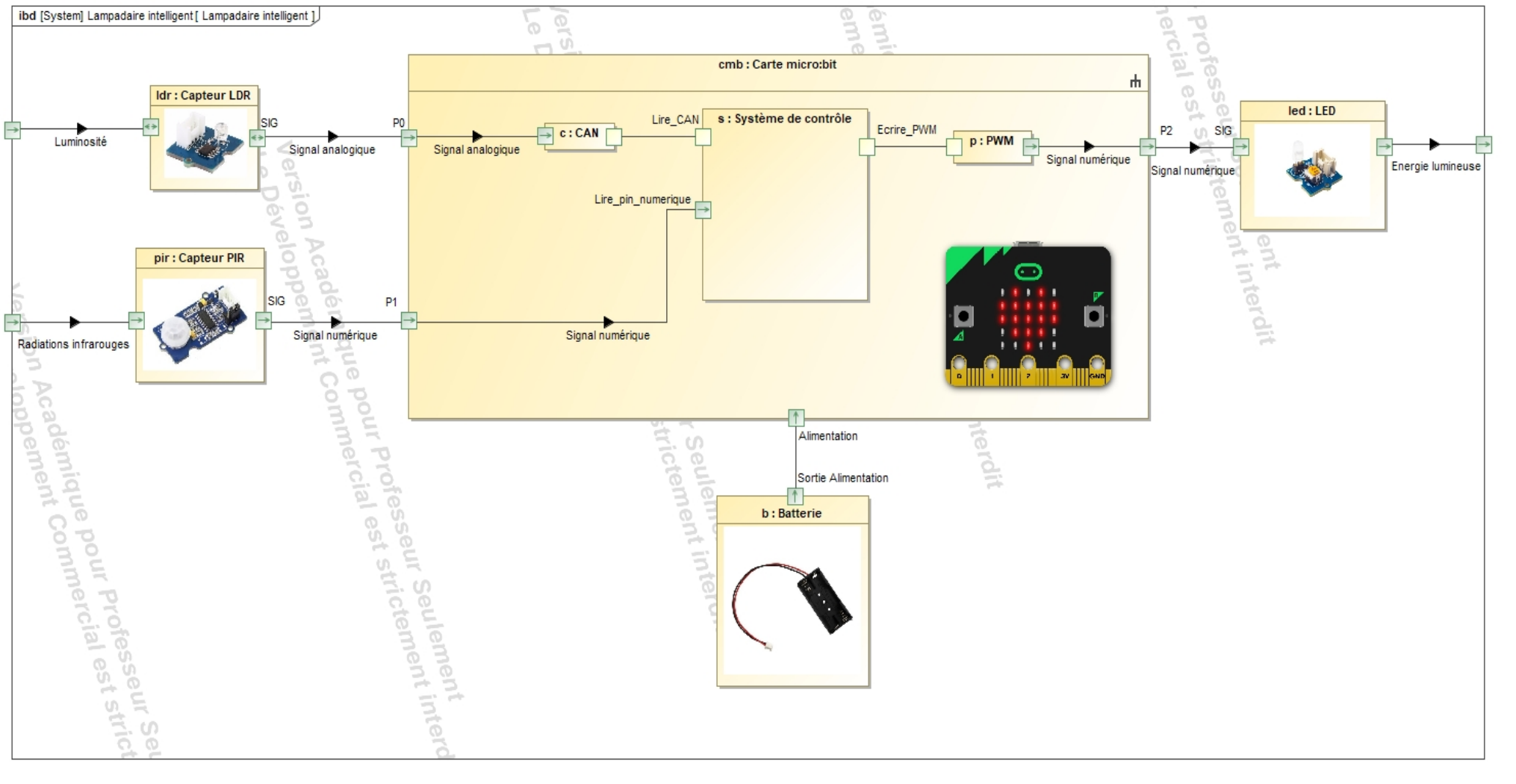
En vous aidant du cahier des charges remplissez le diagramme. Notez les flux au niveau des liens fléchés et le matériel utilisé dans les carrés blancs.



Chaines d’énergie et d’information

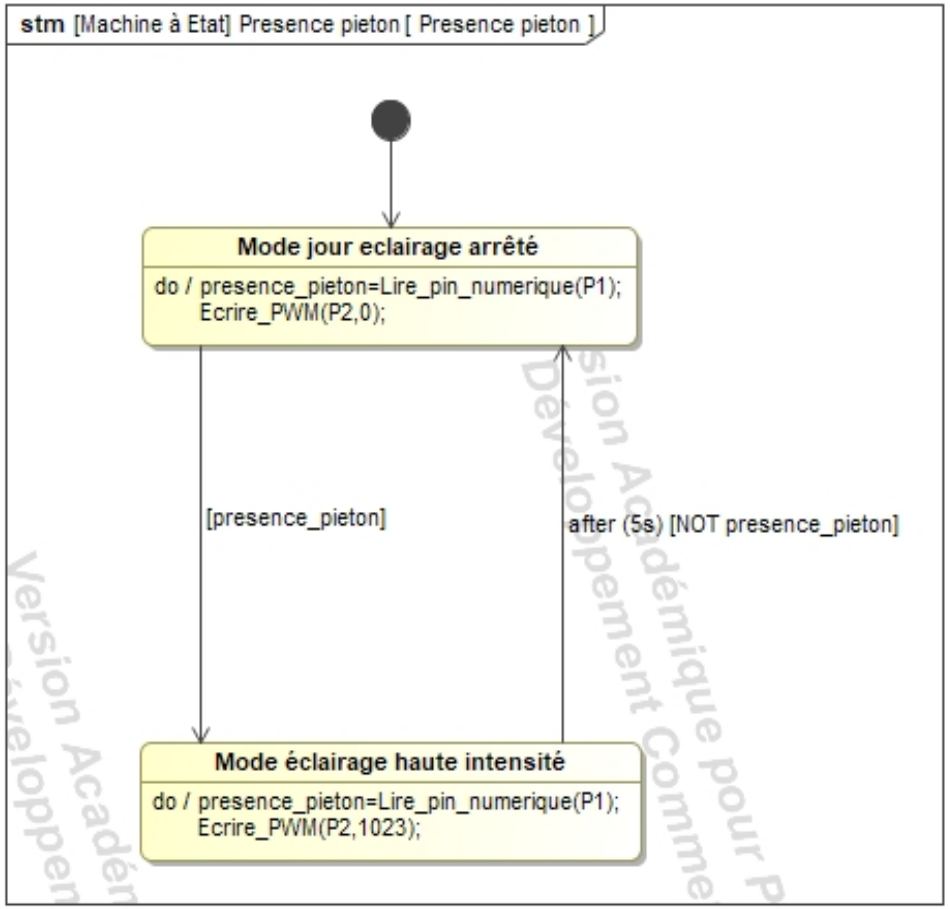
CORRECTION

En vous aidant du cahier des charges remplissez le diagramme. Notez les flux au niveau des liens fléchés et le matériel utilisé dans les carrés blancs.



1) Programme d’éclairage en fonction de la présence des piétons

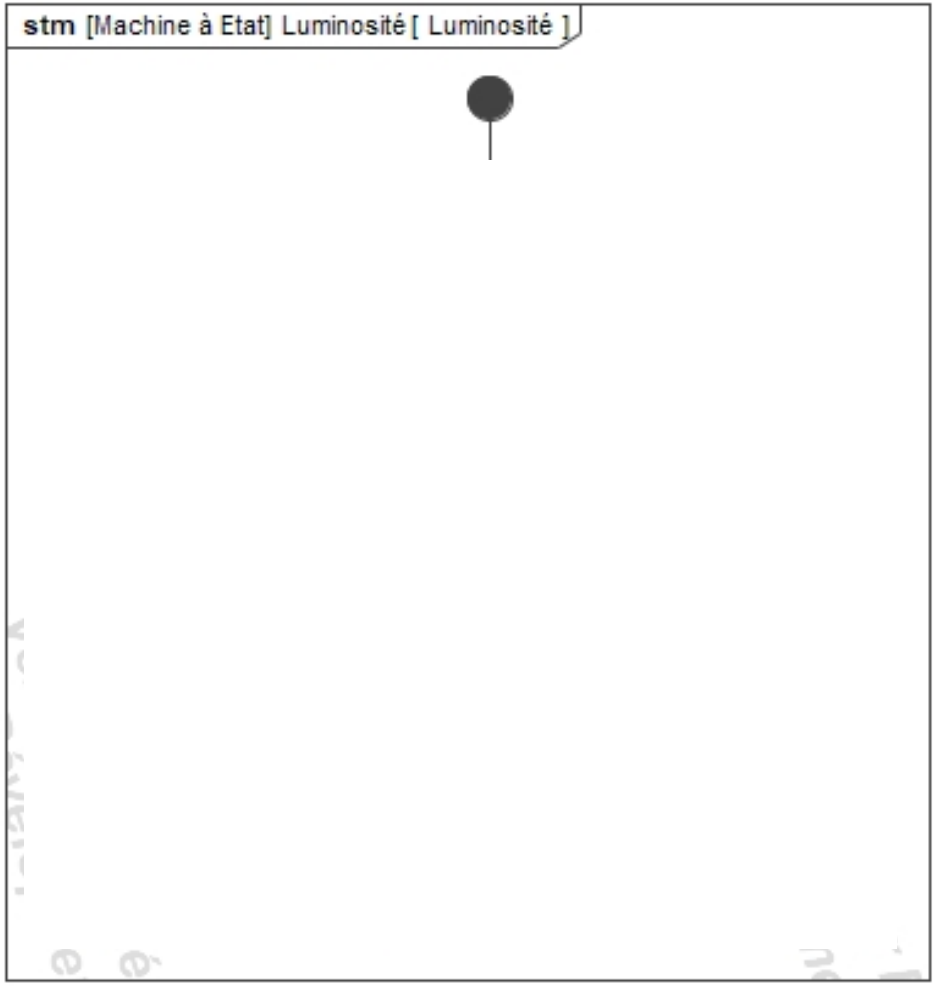
Dans un premier temps nous souhaitons réaliser le programme qui **allume la led lors de la détection d’un piéton**.
Nous donnons la machine à état (MAE), montrant le comportement d’un tel système ci-dessous.
Le système démarre dans l’état **Mode jour éclairage arrêté**, si une **présence piéton** est avérée il passe dans l’état **Mode éclairage haute intensité**.
Après **5s et sans présence piéton** il retourne en **Mode jour éclairage arrêté.**



1. Ouvrez après l’avoir récupéré le fichier python **presence_pietons_ELEVE.py**
2. En vous aidant de l’étude précédente ainsi que de la **fiche méthode microbit et micropython** commencez à réaliser le code de cette MAE.
3. Une fois terminé, **câblez et téléversez** le code sur la micro:bit.
   Attention ! Alimentez par le shield Grove, sinon le PIR ne fonctionnera pas
1. Appelez l’enseignant pour vérifier.

2) Programme d’éclairage en fonction de la luminosité

Dans un second temps nous souhaitons réaliser le programme qui **allume la led lors de la mesure d’une luminosité supérieure au seuil lum_min=512**.

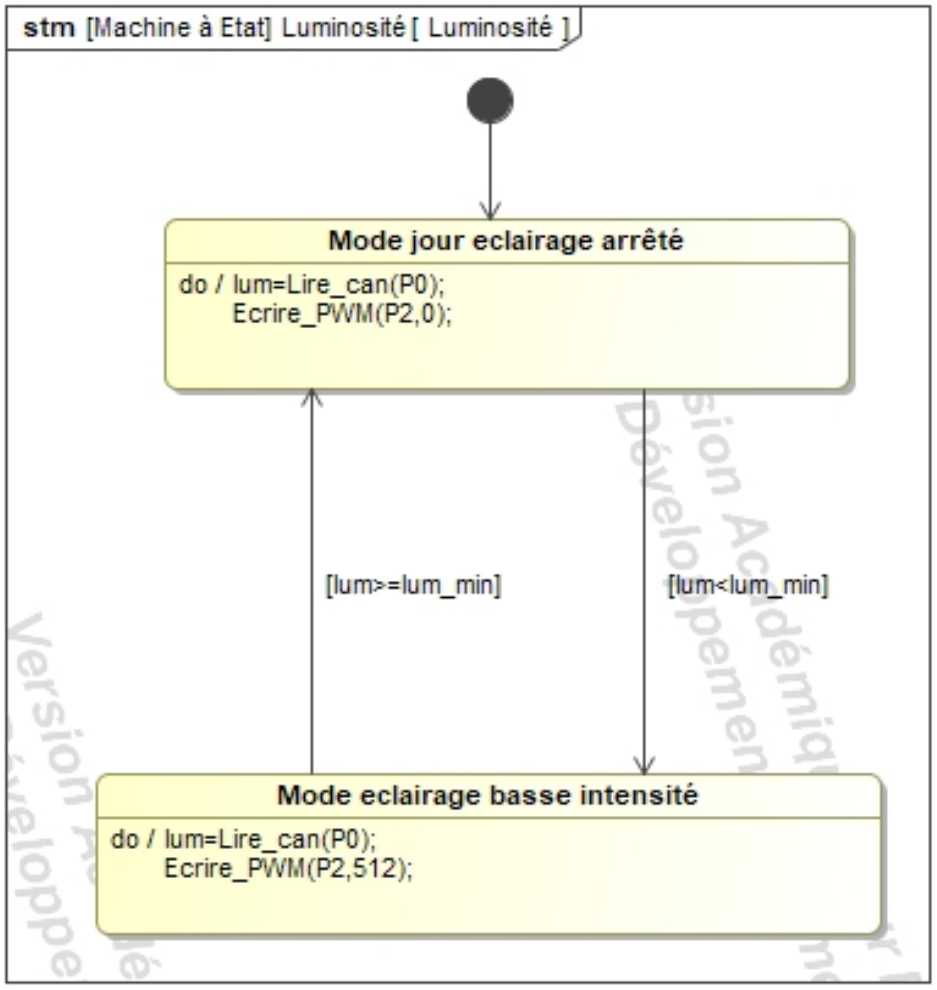


1. Réalisez entièrement la MAE de l’éclairage en fonction de la luminosité
2. Ouvrez après l’avoir récupéré le fichier python **lum_ELEVE.py**
3. En vous aidant de l’étude précédente ainsi que de la **fiche méthode microbit et micropython** commencez à réaliser le code de cette MAE.
4. Une fois terminé, **câblez et téléversez** le code sur la micro:bit.
5. Appelez l’enseignant pour vérifier.

2) Programme d'éclairage en fonction de la luminosité

CORRECTION

Dans un second temps nous souhaitons réaliser le programme qui **allume la led lors de la mesure d'une luminosité supérieure au seuil lum_min=512**.

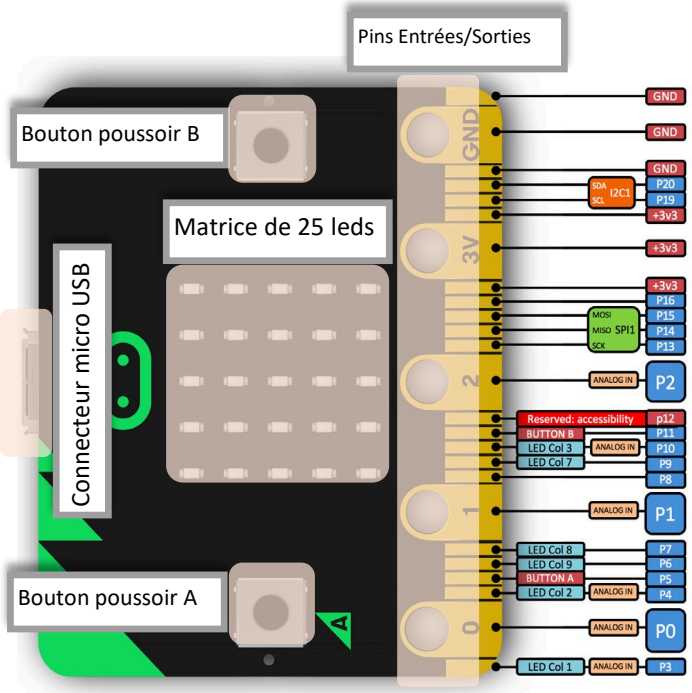


1. Réalisez entièrement la MAE de l'éclairage en fonction de la luminosité
2. Ouvrez après l'avoir récupéré le fichier python **lum_ELEVE.py**
3. En vous aidant de l'étude précédente ainsi que de la **fiche méthode microbit et micropython** commencez à réaliser le code de cette MAE.
4. Une fois terminé, **câblez et téléversez** le code sur la micro:bit.
5. Appelez l'enseignant pour vérifier.

Fiche méthode
Microbit et Micropython

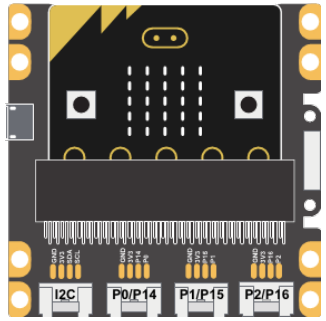
Carte micro:bit

La carte micro:bit peut être programmée de différentes manières, en langages graphiques mais aussi en lignes de codes avec Python pour les cartes à microcontrôleurs : MicroPython.
Nous utiliserons lors des TPs l'éditeur de texte Mu, pour coder et téléverser sur la micro:bit.



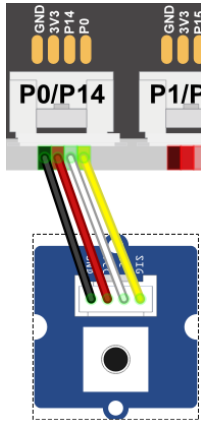
Caractéristiques

- Un processeur ARM pour exécuter votre code
- 25 leds programmables individuellement
- 2 boutons programmables
- 20 pins E/S numériques
- 6 pins d'entrée analogiques (Convertisseur Num. Analog.)
- Capteurs : Température, Accéléromètre, Boussole
- Communication : Radio, Bluetooth



Le shield Grove pour micro:bit

La carte micro:bit doit être positionnée avec la matrice de leds face à vous !



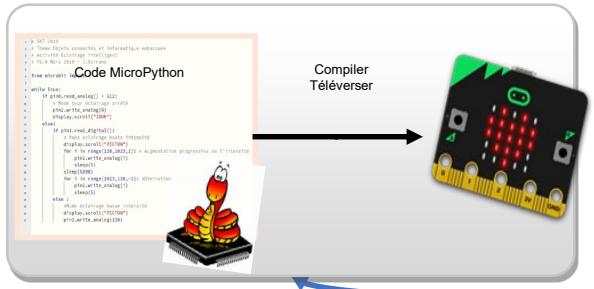
Exemple de connexion d'un module bouton poussoir avec les fils Grove :

- SIG (Signal) du BP à P0 du Sheild
- NC (Not Connected) du BP à P14 du Sheild
- Vcc (+ Alim) du BP à 3V3 du Sheild
- GND (- Alim) du BP à GND du Sheild

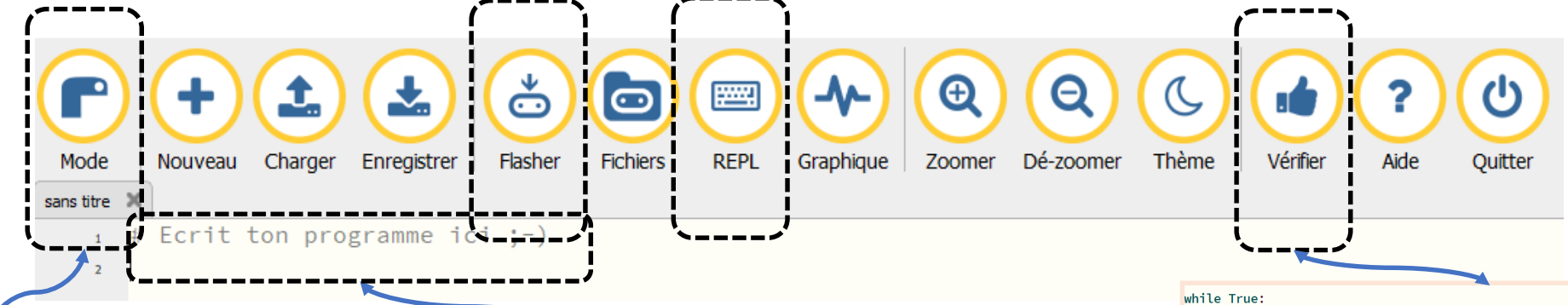
Dans ce cas le signal du bouton poussoir sera à observer sur la pin 0 du micro:bit

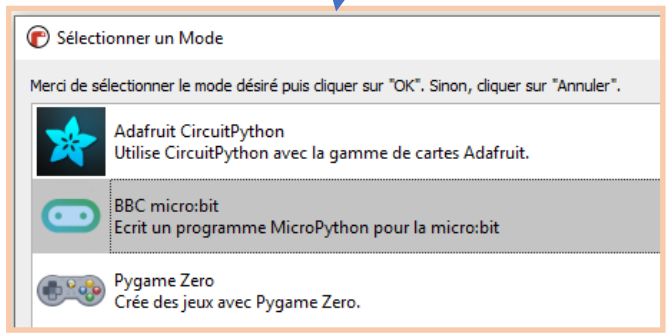
Le logiciel Mu

Mu est le logiciel utilisé en TP pour coder en Python avec la micro:bit.




```
MicroPython v1.9.2-34-gd64154c73 on 2017-09-01; m
Type "help()" for more information.
>>>
>>>
```






```
1  # SNT 2019
2  # Thème Objets connectés et inf          ée
3  # Activité Eclairage intelligen
4  # V1.0 Mars 2019 - J.Serrand
5
6  from microbit import *
7
8  while True:
9      if pin0.read_analog() > 512:
10         # Mode jour éclairage arrêté
```

Zone d'écriture du code Python

```
while True:
    if pin0.read_analog() > 512:
        # Mode jour éclairage arrêté
        pin2.write_analog(0)
        display.scroll("JOUR")
    else:
        if pin1.read_digital():
            # Mode eclairage haute intensité
            display.scroll("PIETON")
            for i in range(128,1023,1): # Augmentation progressive de l'intensité
                ↑ Missing whitespace after ','
                ↑ Missing whitespace after ','
                ↑ At least two spaces before inline comment
                ↑ Line too long (81 > 79 characters)
                pin2.write_analog(i)
                sleep(5)
            sleep(5000)
            for i in range(1023,128,-1): #Diminution progressive de l'intensité
                ↑ Missing whitespace after ','
                ↑ Missing whitespace after ','
                ↑ At least two spaces before inline comment
                ↑ Inline comment should start with '# '
                pin2.write_analog(i)
                sleep(5)
```

Vérification du code (debugger)

Explication des erreurs ou warnigs

Explication des erreurs ou warnigs

https://microbit-micropython.readthedocs.io/fr/latest/tutorials/introduction.html

Code Python

Un code MicroPython pour la carte micro:bit comporte plusieurs parties.

```
1  # SNT 2019
2  # Nom du programme
3  # Rôle du programme
4  # Nom Prénom du codeur
5  # V1.0 Date
6
7  from microbit import *
8
9  foo=22
10 pi=3.14
11 chaine="Vive le SNT !"
12
13 def Ma_fonction(param1,param2):
14     ...
15     #Bloc d'instructions
16     ...
17     return resultat
18
19 while True:
20     #Mon code débute ici
```

Cartouche du code

Import de la library du micro:bit

Déclaration de variables globales

Ajout de macros

Boucle principale

Bloc d'instructions

Un code Python est divisé en blocs, les instructions de contrôles : if, while, for, etc, nécessite une indentation (tab), pour que Python comprenne qu'on souhaite coder dedans :

```
while True:
    #Je suis dans le while
    if pi==3.14:
        #Je suis dans le if
```

Code Python (suite)

Variables
Python s'occupe automatiquement du typage de la variable, lors de la première affectation :
foo=22 #foo est un entier
pi=3.14 #pi est un flottant

Types de variables
Principaux types de variables :

- int #Entier
- float #Flottant contient un nombre décimal
- str #String contient une chaîne de caractères (phrase)
- bool #Booléen contient True ou False (Vrai ou Faux)
- file #Contient un fichier

Transtypage
Python donne la possibilité d'imposer un type à une variable, exemple avec la console Python :
>> pi=3.14 #pi est un flottant
>> pi=int(pi) #On transforme pi en entier
>> pi
>> 3 #On a tronqué pi

Opérateurs
Dans une condition il faut parfois de comparer une variable avec une valeur ou encore deux variables entre-elles. Pour cela Python accepte les opérateurs suivants.

Opérateurs logiques	
or	OU logique 
and	ET logique 
not	NON logique 

Opérateurs de comparaison	
==	Egalité
!=	Différence
<	Infériorité
>	Supériorité
<=	Inférieur ou égal
>=	Supérieur ou égal

Opérateurs mathématiques	
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo (Reste de la div)
**	Puissance

Code Python (suite)

Structures conditionnelles

Si - if

```
if pi==3.14:
    #A faire si condition vraie
```

Si Sinon - if else

```
if pi==3.14:
    #A faire si condition vraie
else :
    #A faire si condition fausse
```

Imbrication des if else

```
if pi==3.14:
    #A faire si pi=3.14
elif pi==0 :
    #A faire si pi=0
elif pi==1 :
    #A faire si pi=1
else :
    #A faire si toutes les
    #conditions précédentes
    #sont fausses
```

Structures répétitives

Tant que - while

```
while foobar>3:
    #A faire tant que foobar>3
```

Pour - for

```
for cpt in range(0,10,1):
    #cpt -> variable de comptage
    #0 -> initialisation de cpt
    #10 -> compte de 0 à 9
    #1 -> Compte de 1 en 1
```

Code Micro:Python

Lire sur une pin numérique

```
valeur=pin0.read_digital()
#Stocke dans "valeur" l'état de la
#pin0 -> 0 ou 1
```

Ecrire sur une pin numérique

```
pin0.write_digital(1)
# Met la pin 0 à l'état haut
pin0.write_digital(0)
# Met la pin 0 à l'état bas
```

Utiliser le CAN

Le convertisseur analogique numérique permet de convertir une information provenant d'un capteur analogique entre 0 et 3.3V en un entier N compris entre 0 et 1023.

Veuillez bien regarder que la pin choisie soit une entrée analogique → analog in

```
N=pin0.read_analog()
    #Stocke dans N la valeur convertie
    #par le CAN [0;1023]
```

Utiliser le PWM

Le PWM (Pulse Width Modulation) ou Modulation de Largeur d'Impulsions en français permet de faire varier le rapport cyclique d'un signal carré.

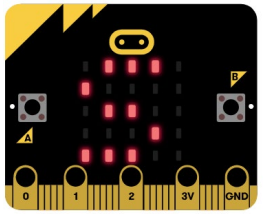
En sortie le composant connecté à la micro:bit verra une variation de la tension entre 0 et 3.3V pour un nombre alpha codé entre 0 et 1023.

```
pin0.write_analog(512)
    # Met une tension de 3.3/2=1.65V sur la pin 0
pin0.write_analog(1023)
    # Met une tension de 3.3V sur la pin 0
pin0.write_analog(0)
    # Met une tension de 0V sur la pin 0
```

Code Micro:Python (suite)

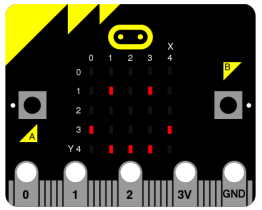
Afficher un texte sur la matrice de leds

```
display.scroll("SNT")
```



Afficher une image sur la matrice de leds

```
display.show(Image.HAPPY)
```



Une liste exhaustive des images est disponible sur le site : microbit-micropython.readthedocs.io

Utiliser les 2 boutons poussoirs

```
etat_A=microbit.button_a.is_pressed()
#etat_A=1 si bp A appuyé sinon 0

etat_A=microbit.button_a.was_pressed()
#Renvoie 1 si bp A appuyé depuis le début
#ou depuis le dernier appel de cette fonction

nb_appuis_A=microbit.button_a.get_presses()
#Renvoie le nb d'appuis sur bp A depuis le début
#ou depuis le dernier appel de cette fonction
```

Utiliser l'accéléromètre

L'accéléromètre intégré à la carte est capable de détecter certains types de mouvements préenregistrés.

up (de bas en haut)	**right** (de gauche à droite)	**freefall** (en chute libre)
down (de haut en bas)	**face up** (tourné matrice en haut)	**3g, 6g, 8g** (soumis aux accélérations du même nom)
left (de droite à gauche)	**face down** (tourné matrice en bas)	**shake** (secoué de haut en bas)

Utiliser la boussole

Pour un maximum de fiabilité il est vraiment nécessaire de calibrer la boussole

```
microbit.compass.calibrate()
# Lance le programme de calibration

angle=microbit.compass.heading()
#Renvoie l'angle par rapport au nord
#0° -> N
#180° -> S
#90° -> E
#270°-> O
```