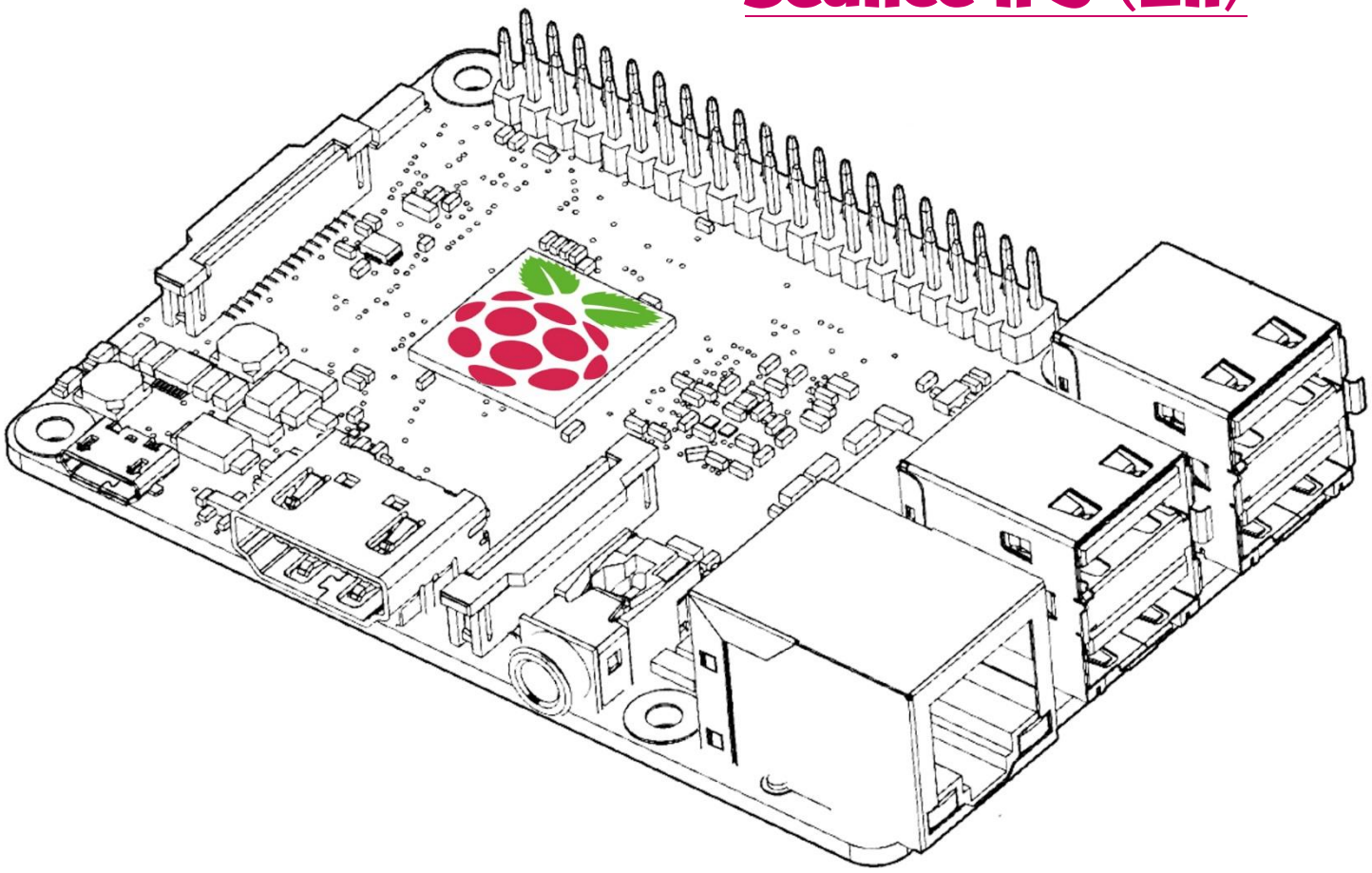
AT31 - Module IPE

IUT RENNES

Séance n°3 (2h)



Utilisation de la carte d'extension 'Sense Hat'

Michaël BOTTIN

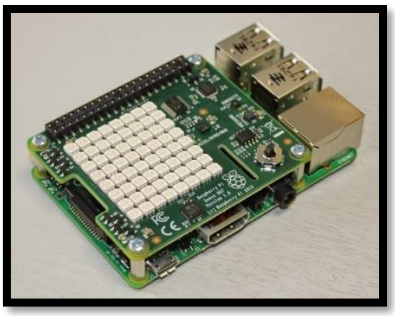
Objectifs de la séance :

- Découvrir l'utilisation d'une carte d'extension commercialisée
- Utiliser une bibliothèque de très haut niveau disponible avec la carte
- Découvrir de nouveaux capteurs

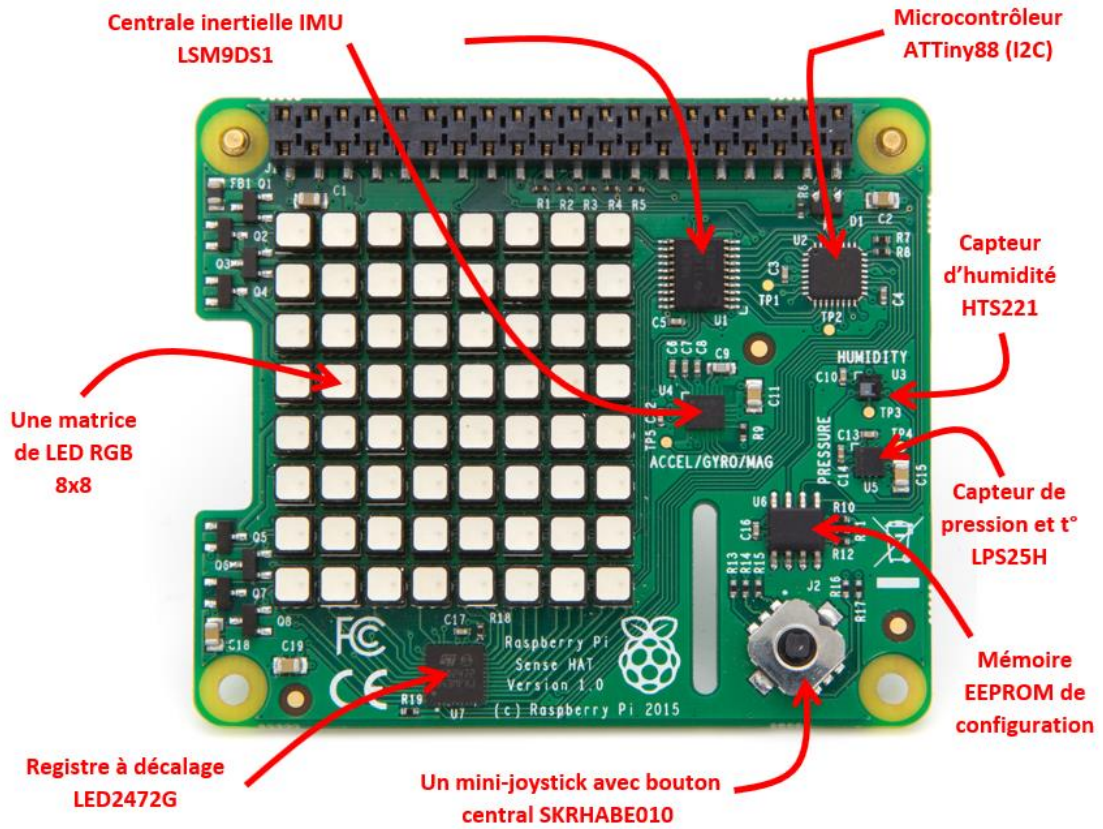
Présentation de la carte d'extension 'Sense HAT' :

NE PAS ALIMENTER TOUT DE SUITE !!

Nous allons utiliser au cours de ce TP une carte d'extension pour la Raspberry pi sortie en septembre 2015.



Qu'est-ce que l'on retrouve sur cette carte d'extension :



Chacune des 256 LEDs de la matrice dispose de trois broches indépendantes R, G et B permettant d'obtenir par combinaison toutes les couleurs souhaitées.
Les LEDs sont câblées pilotées par un registre à décalage **LED2472G**, lui-même commandé par un microcontrôleur **Atmel ATTiny88**. Un autre composant, le registre **74AHCT245** participe également au pilotage.
L'ensemble est commandé via le bus I2C du microcontrôleur (un bus de communication au même titre que la liaison série mais un peu plus compliqué à piloter). L'adresse I2C du microcontrôleur est **0x46**.

Le mini-joystick **SKRHABE010** est également associé au microcontrôleur et possède donc la même adresse I2C.

Le capteur de pression **LPS25H** est directement connecté au bus I2C, son adresse est **0x5C**. Sa plage de mesure s'étend de 60 à 1260 hPa.
Le capteur d'humidité et de température **HTS221** est lui aussi directement connecté au bus I2C. Son adresse est **0x5F**. Sa plage pour l'humidité est de 0 à 100% avec une précision de 4,5% et pour la température de -40° à +120° avec une précision maximale de 0,5°C.
La centrale inertielle IMU de référence **LSM9DS1** contient un accéléromètre 3 axes, un gyroscope 3 axes et un magnétomètre 3 axes. Ce composant est connecté sur le bus I2C. L'adresse de l'accéléromètre/gyroscope est **0x6A** tandis que le magnétomètre est sur **0x1C**.
L'accéléromètre dispose d'une plage ±2/4/8/16 g
Le gyroscope dispose d'une plage angulaire ± 245/500/2000 dps (degrees per second)
Le magnétomètre dispose d'une plage ± 4/8/12/16 gauss

J'ai évoqué que le bus I2C qui permet d'accéder aux différents composants de la carte est relativement complexe en compréhension et en mise en œuvre. Toutefois ici, on va utiliser une librairie d'un niveau si élevée que le bus I2C deviendra totalement transparent, tout comme le pilotage de la matrice de LED via le registre à décalage.

Une caractéristique importante des cartes officielles 'HAT' est la possibilité d'embarquer une mémoire EEPROM I2C qui communiquera avec la Raspberry dès la mise sous tension. Cette mémoire permettra notamment de fournir la configuration des différentes broches GPIO de la Raspberry pour qu'elles soient utilisables par la carte d'extension. C'est notamment le cas pour la Sense Hat. Vous pourrez le vérifier rapidement à la mise sous tension puisque celle-ci affichera déjà un mode arc-en-ciel sur la matrice alors qu'aucune programmation n'aura été effectuée.

Vous pouvez dès à présent installer votre carte Raspberry avec ses périphériques (clavier, souris, câble d'écran, câble Ethernet), sa carte µSD. Vérifier que vous êtes bien hors tension, et installer alors la carte d'extension Sense Hat sur la Raspberry si ce n'est pas déjà fait.

Vous pouvez alors alimenter la Raspberry avec son adaptateur secteur. Vous devez voir le voyant d'alimentation de la carte d'extension s'allumer.

Remarque : pour information, cette carte est vendue à environ 40€, ce qui est le prix de la Raspberry !! Bon nombre de cartes d'extension, même parfois très simples, sont vendues assez cher. Si celle-ci concentre beaucoup de technologie dans un espace restreint, pas mal d'autres cartes se résument à des composants standards. Il devient alors beaucoup plus intéressant de la réaliser soi-même avec les compétences acquises à l'IUT… 😉

Récupération des fichiers utiles pour cette séance :

Pour cela, tapez la commande suivante dans le terminal :
$ git clone https://github.com/MbottinIUT/IUT_IPE_TP3.git
[rem : il n'y a pas d'espace dans ce lien, juste des '_']

Une fois la tâche effectuée, vous devriez retrouver, en ouvrant un explorateur de fichiers, un dossier '**IUT_IPE_TP3**' sous '**/home/pi**'.

Recommandation importante concernant les programmes à écrire :

On utilisera comme d'habitude l'environnement de développement '**Thonny**' et vous veillerez à avoir un fichier par exercice demandé.

On va utiliser pour ces exercices une librairie python de très haut niveau dédiée à l'utilisation de cette carte.
Vous trouverez la documentation (in english !!) de cette librairie à l'adresse suivante :
https://pythonhosted.org/sense-hat/api/
Chaque méthode de la classe 'Sense hat' est bien détaillée et accompagnée d'un exemple.
Ceci devrait vous permettre d'écrire vos programmes avec une certaine autonomie !!
Je vous conseille de laisser cette page web ouverte tout au long de cette séance.

Cette librairie est installée par défaut dans l'OS. Toutefois, il est toujours intéressant de la maintenir à jour (quelques bugs peuvent avoir été corrigés récemment).
Veuillez donc taper la commande suivante dans un terminal :
$ sudo apt-get install sense-hat
Cette librairie contient en fait une classe de haut niveau nous fournissant des méthodes et des attributs pour accéder/modifier aux informations fournies par les différents composants de la carte.

Voici à peu près la structure que vous devez adopter pour les programmes de cette séance :

```
# TITRE DE L'EXERCICE

# Importation des modules nécessaires

# Instanciation des objets nécessaires

# Initialisation des variables

# -------------------------
# -- PROGRAMME PRINCIPAL --
# -------------------------
# Boucle infinie
while True :
    try :
        # Code de l'exercice
    except KeyboardInterrupt :
        # Effacement de la matrice
        sense.clear()
        print ("Au revoir...")
        # Quitte la boucle infinie
        break
```

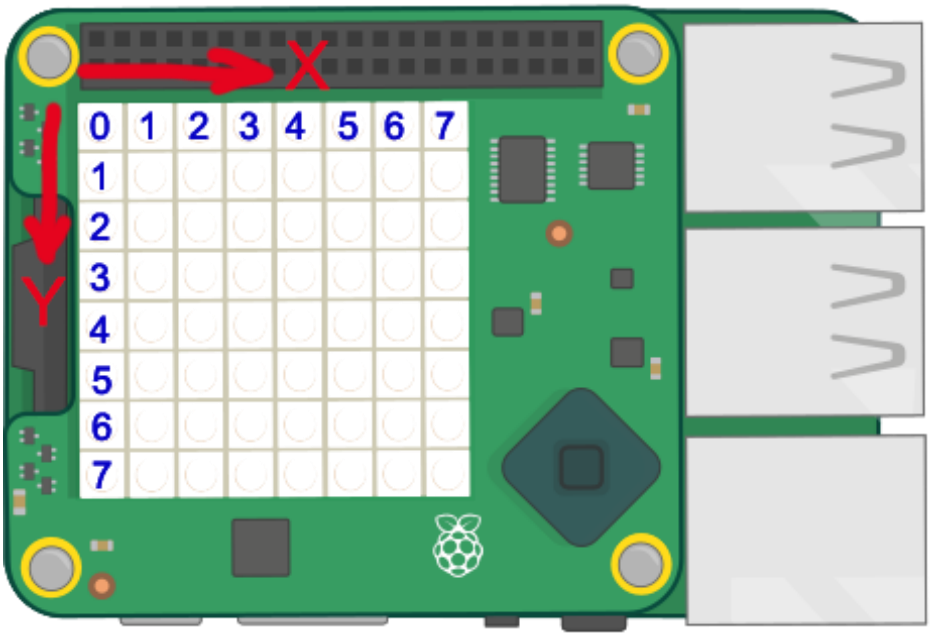
EXERCICE 1 : 'Ex01.py'

Objectif de l'exercice :

Pour commencer ce TP, on utiliser la matrice de LED RGB pour la remplir de pixels, chacun à une position aléatoire et d'une couleur aléatoire avec une animation qui se déroule indéfiniment.

Informations utiles :

- Pour gérer la carte, il faut commencer par importer la librairie python correspondante. C'est l'instruction '**from sensehat import SenseHat**' qui permet cela. On aura donc à l'importer dans tous les exercices de ce TP.
- De la même façon, il faut créer (instancier) un objet basé sur la classe d'objet définie dans cette librairie. L'instruction qui s'en charge se retrouvera donc dans l'ensemble de nos exercices également : '**sense = SenseHat()**'
- **IMPORTANT :** dès lors, dès que vous voulez accéder à une méthode relative à cet objet, vous utiliserez la syntaxe '**sense.methode()**' avec ou sans paramètre dans les parenthèses et en remplaçant bien évidemment le mot 'methode' par celle qui vous intéresse. Et si vous voulez accéder à un attribut de cet objet, vous utiliserez la syntaxe '**sense.attribut**' là aussi en remplaçant le mot 'attribut' par celui qui vous intéresse.
- Il est important de connaître où est l'origine de la matrice 8x8 et comment sont numérotées les lignes et les colonnes pour la suite :



Voici donc comment sont orientés les axes des coordonnées x et y.
Leurs valeurs vont de 0 à 7.

- Que ce soit pour les deux coordonnées **x** et **y** ou pour les trois composantes de la couleur **r**, **g** et **b**, vous aller devoir générer un nombre aléatoire. Pour cela vous utiliserez la méthode '**randint()**' :
  - Vous allez avoir besoin de générer des nombres aléatoires. Pour cela, vous allez utiliser la classe 'random' qu'il faut importer dans votre programme (juste après le bloc de commentaires) par la commande '**import random**'
  - Cette classe 'random' dispose de plusieurs méthodes. Celle qui nous intéresse s'appelle 'randint' et s'utilise ainsi : '**nombre = random.randint(a,b)**' . Consultez la page suivante pour savoir quoi mettre en a et b : https://docs.python.org/2/library/random.html.

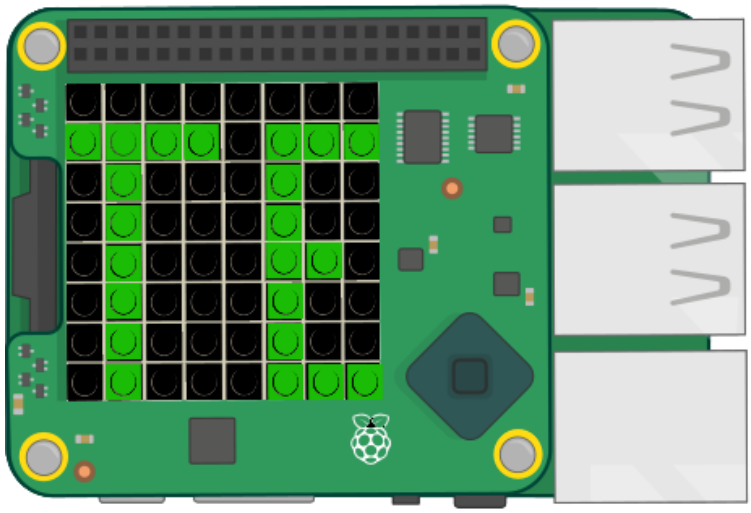
- Remarque : à noter que les 3 composantes de la couleur sont comprises entre 0 et 255 !!
- Enfin, pour allumer un pixel aux coordonnées voulues et dans la couleur voulue, on va utiliser la méthode '**set_pixel()**'. A vous de regarder dans la documentation ('Mémento Rpi' page 39) pour savoir comment l'utiliser.
- Vous pouvez ajouter une faible pause (avec '**sleep**') entre chaque pixel allumé de manière à rendre l'animation plus agréable visuellement parlant.
- Veillez toujours à effacer la matrice ('**sense.clear()**') dans l'exception '**KeyboardInterrupt**' de façon à sortir proprement du programme.

Faîtes contrôler le bon fonctionnement de cet exercice par un enseignant.

EXERCICE 2 : 'Ex02.py'

Objectif de l'exercice :

On va poursuivre l'utilisation de cette matrice en y faisant défiler horizontalement un texte saisi dans le terminal par l'utilisateur. Vous devez vous assurer que la longueur du message n'excédera pas 60 caractères !!



Informations utiles :

- Dans le terminal, le programme doit demander à l'utilisateur le message qu'il désire afficher et lui redemander un autre message si ce dernier dépasse 60 caractères. Remarque : pour l'information, on obtient la longueur d'une chaîne de caractères par la méthode '**len(chaine)**'.
- On affichera le message en jaune sur un fond bleu ou dans deux autres couleurs de votre choix pourvu qu'elles soient contrastées et que le message reste bien lisible.
- Pour l'affichage et le scrolling du texte, on va utiliser la méthode '**show_message()**' de l'objet '**sense**'. Utiliser la documentation en ligne (section '**LED matrix**') pour voir quels sont ses paramètres et comment l'utiliser.
- Tester également cette option proposée pour la rotation de l'affichage en utilisant la méthode '**sense_rotation()**'.

Faîtes contrôler le bon fonctionnement de cet exercice par un enseignant.

EXERCICE 3 : 'Ex03.py'

Objectif de l'exercice :
On va maintenant s'intéresser à un des capteurs disponibles sur la carte Sense Hat. On va ici s'intéresser à la mesure de la température.
Le but va donc être d'afficher en message défilant sur la matrice 8x8 la température mesurée par le capteur.
La couleur de fond de la matrice dépendra de la valeur de la température.

Informations utiles :

- On va utiliser, comme nous l'avons déjà fait, la méthode '**Show_message()**' pour afficher les différentes informations provenant des capteurs.
- Le capteur de température est câblé sur le bus I2C de la Raspberry, mais on dispose toujours ici d'une librairie de très haut niveau qui rend la communication sur ce bus totalement transparente pour l'utilisateur. On va utiliser la méthode '**get_temperature()**' de la librairie '**Sense Hat**'. Veuillez-vous reporter à la page web de cette librairie (section '**environmental sensors**') pour voir son utilisation.
- Pour améliorer la lecture sur la matrice 8x8, on va arrondir ces informations à un seul chiffre après la virgule. Vous allez utiliser la fonction mathématique '**round()**' dont vous trouverez la documentation officielle de Python en ligne : https://docs.python.org/fr/3/library/functions.html
- Pour l'affichage, il va falloir combiner dans une chaîne de caractères un message au texte évocateur (exemple 'temperature = ' ou 'temperature : ') et la valeur de cette température.
- Vous pouvez bien sûr utiliser à tout moment la fonction '**print**' dans le terminal pour contrôler le contenu des objets que vous manipulez.
- Pour la couleur de fond de la matrice, les seuils de température vont être un peu étonnants. En effet, le capteur de température étant situé juste au-dessus de la Raspberry, la chaleur qui se dégage de cette dernière fausse forcément la valeur obtenue. On va donc choisir :
  - Couleur bleue : température < 25°
  - Couleur magenta : température comprise entre 25° et 35°
  - Couleur rouge : température > 35°

  [Veillez à utiliser une couleur de texte compatible avec les trois couleurs de fond]
- Remarque : Essayez de comprendre pourquoi la valeur affichée ne correspond pas réellement à la température de la pièce.

Faire valider le fonctionnement de votre programme par un enseignant.

TUTO 1 : 'Tuto01.py'

Dans le prochain exercice, nous allons afficher 4 informations différentes sur la matrice. Chacune de ces informations occupera un espace identique de 4x4 pixels.

Mais avant cela, il faut vous fournir quelques informations.
Commençons par voir comment on peut afficher en une fois les 64 pixels de la matrice.
Dans un fichier, importer la librairie 'sense hat' et instancier un objet '**sense**' comme dans les 3 exercices précédents.
Puis créer deux couleurs (en mode RGB) :

- **n = (0,0,0)**
- **r = (255,0,0)**

Créez ensuite une liste de 64 valeurs en mixant les couleurs rouge et noire précédemment créées.

Cela peut ressembler à ceci :

- **CŒUR = [n,n,n,n,n,n,n,n,**
  n,n,r,n,n,n,r,n,
  n,r,r,r,n,r,r,r,
  n,r,r,r,r,r,r,r,
  n,r,r,r,r,r,r,r,
  n,n,r,r,r,r,r,n,
  n,n,n,r,r,r,n,n,
  n,n,n,n,r,n,n,n]

Utilisez alors la méthode '**set_pixels()**' (page web de la librairie 'Sense Hat' dans la section 'LED Matrix') pour afficher la totalité des pixels de la matrice définie dans l'objet '**CŒUR**'.

Vous avez donc vu maintenant comment afficher en une seule instruction l'intégralité des 64 pixels pour peu que l'on ait défini une liste de 64 couleurs précédemment.

Toutefois dans le prochain exercice, les éléments à afficher seront au format 4x4 !!
Il va donc falloir à partir de liste de 16 valeurs reconstruire une liste de 64 valeurs.

La suite de ce tutoriel va vous y aider.
A la suite du programme précédent, vous pouvez copier le code suivant :

```
VIDE_2x2 = [[0,0],[0,0]]

PLEIN_2x2 = [[1,1],[1,1]]

RESULTAT_4x4 = []

for index in range(0, 2):
    RESULTAT_4x4.extend(PLEIN_2x2[index])
    RESULTAT_4x4.extend(VIDE_2x2[index])
for index in range(0, 2):
    RESULTAT_4x4.extend(VIDE_2x2[index])
    RESULTAT_4x4.extend(VIDE_2x2[index])

for index in range (0,4) :
    for index2 in range (0,4) :
        print (RESULTAT_4x4[index2 + index*4],end="")
    print("")
```

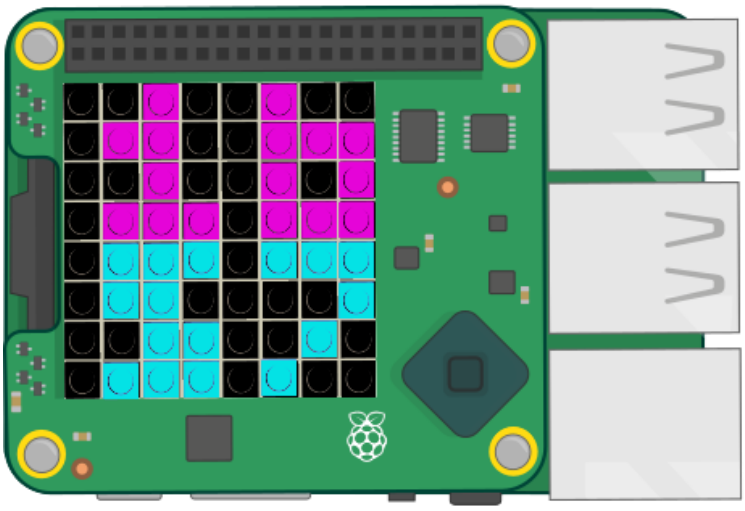
Testez-le et intervertissez les '**PLEIN_2x2**' et '**VIDE_2x2**' dans les boucles '**for**' pour bien comprendre le fonctionnement des boucles et de la méthode '**extend()**'.
Nous avons donc vu comment simplement créer une table 4x4 à partir de table 2x2 tout en respectant l'ordre des pixels imposé par la matrice.
Vous pouvez maintenant aborder l'exercice suivant en vous appuyant sur ce que vous venez de voir.

EXERCICE 4 : 'Ex04.py'

Objectif de l'exercice :

L'objectif de cet exercice est d'utiliser ce que l'on vient d'apprendre dans le tutoriel précédent pour réaliser une horloge digitale sur la matrice 8x8.
Bien sûr, dû à la faible résolution de la matrice, la définition de cette horloge ne sera pas parfaite, mais le concept est applicable à une matrice de dimensions plus grandes.

L'aspect final de cette horloge doit donc être comme ceci :



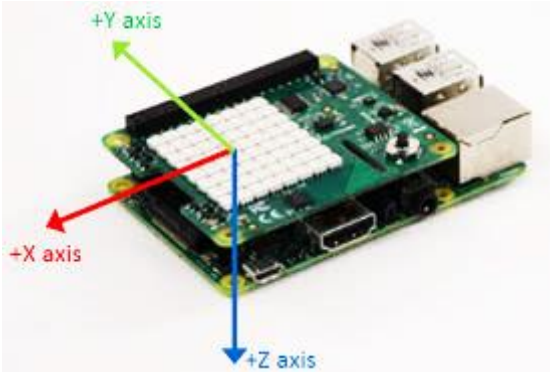
Informations utiles :

- Vous afficherez les dizaines d'heures en haut à gauche de la matrice, les unités d'heures en haut à droite, les dizaines des minutes en bas à gauche et les unités des minutes en bas à droite. Les heures sont affichées dans la couleur Magenta alors que les minutes sont dans la couleur cyan.
- Pour lire l'heure système, vous importerez la librairie 'time' et vous utiliserez les attributs '**time.localtime().tm_hour**' et '**time.localtime().tm_min**' pour récupérer la valeur des heures et des minutes.
- Pour ne pas perdre trop de temps, un squelette de code vous est fourni dans lequel l'initialisation des chiffres est déjà faite sous forme de listes au format 4x4.
- Pour accéder à la liste de 4x4 valeurs correspondant au chiffre 5 par exemple, il suffit d'écrire '**chiffres[5][index]**' avec '**index**' balayant les valeurs de 0 à 3.
- Vous utiliserez encore la méthode '**set_pixels()**' pour afficher l'intégralité des 4 chiffres sur la matrice.
- Il faut donc au préalable constituer cette liste de 64 valeurs avec la méthode '**append()**' en vous inspirant du tutoriel précédent.

Faire valider le fonctionnement de votre programme par un enseignant.

EXERCICE 5 : 'Ex05.py'

Objectif de l'exercice :
On va maintenant explorer un autre capteur de cette carte 'Sense Hat'. Il s'agit de l'accéléromètre.
Comme je l'ai évoqué dans l'introduction, il s'agit d'un accéléromètre 3 axes. Il est intéressant de savoir comment ces derniers sont orientés sur la carte :



Le but de cet exercice est de récupérer et d'afficher en continu simplement les valeurs brutes ('**raw**' en anglais !!) des 3 axes de l'accéléromètre (valeur entre -1.0 et +1.0) dans le terminal.

Informations utiles :

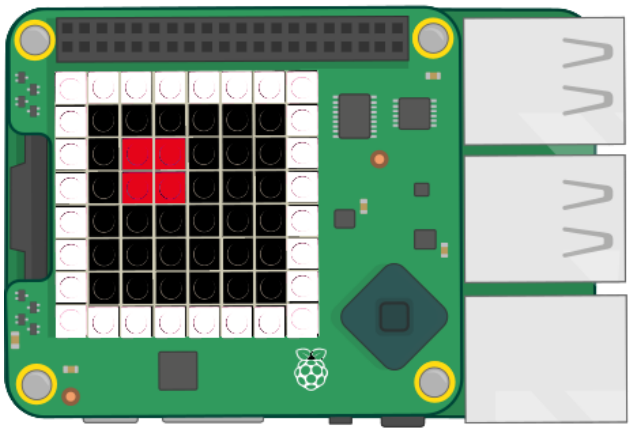
- Consulter la page web de la librairie (section '**IMU Sensor**') pour déterminer la méthode à utiliser dans cet exercice.
- La méthode utilisée renvoie un objet complet contenant les 3 informations. Cet objet est appelé un dictionnaire en Python. Pour pouvoir extraire chaque valeur d'axe dans une variable dédiée (x/y/z), il faut une petite opération. Commencer par l'afficher via un '**print**' dans votre programme. Aidez-vous ensuite de la page suivante pour isoler chaque donnée dans une variable : https://python-django.dev/page-apprendre-dictionnaire-python
- Affichez dans le terminal ces trois informations avec un message clair permettant de bien identifier chacune des trois valeurs (exemple : « x= … / y = … / z = ... »).

EXERCICE 6 : 'Ex06.py'

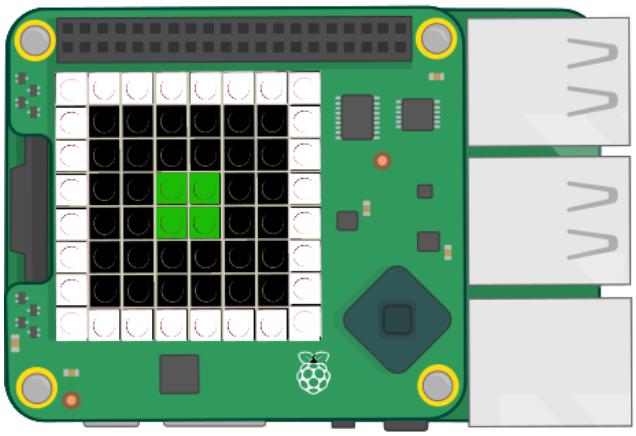
Objectif de l'exercice :
Maintenant que l'on sait comment récupérer les informations de l'accéléromètre, on va pouvoir les utiliser au travers deux exemples.

Le premier exemple, objet de cet exercice, va consister à réaliser un niveau à bulle électronique avec un affichage sur la matrice. On va se restreindre pour cet exercice aux deux axes X et Y de l'accéléromètre.

Voici à quoi doit ressembler finalement votre niveau :



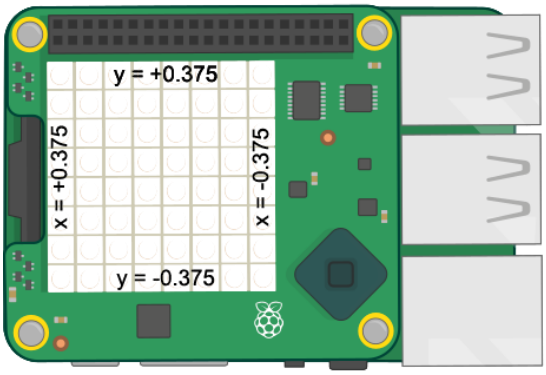
Niveau incliné



Niveau à plat (horizontal en X et Y)

Informations utiles :

- Le carré de quatre pixels (rouge ou vert) représente notre 'bulle' et le cadre blanc les limites de notre niveau. Seul le carré se déplace selon l'orientation de la carte.
- La 'bulle' doit être verte lorsqu'elle est au centre (carte Sense Hat à plat) et rouge partout ailleurs.
- Pour afficher le cadre, vous avez intérêt à utiliser la méthode '**set_pixels()**' plutôt que de dessiner les pixels un par un !!
- Pour la 'bulle', il est par contre conseiller d'utiliser la méthode '**set_pixel()**' quatre fois.
- Pour augmenter la sensibilité de notre niveau lorsqu'il est proche de l'horizontal, vous allez devoir limiter les valeurs de x et y à +/- 0.375 :



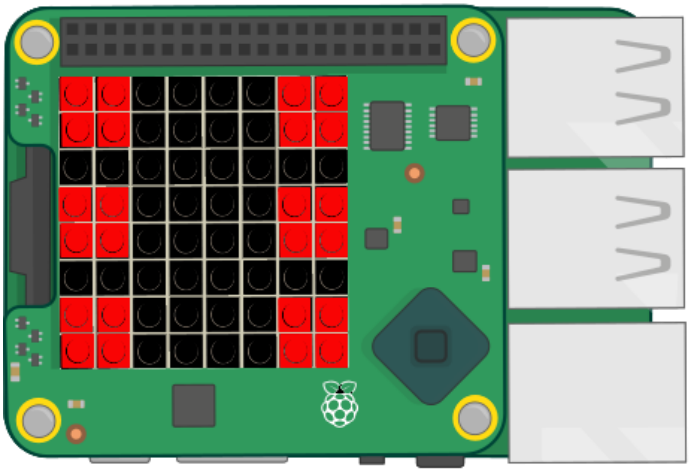
- Pour le calcul de la position de la 'bulle', vous aurez certainement besoin de convertir votre calcul en un entier. Cela se fait facilement en Python via la fonction '**int()**'.

Faire valider le fonctionnement de votre programme par un enseignant.

EXERCICE 7 : 'Ex07.py'

Objectif de l'exercice :

On va continuer à utiliser l'accéléromètre dans un second exemple. Ici, on va créer un dé électronique et pour simuler son lancer, il faudra secouer légèrement la carte 'Sense Hat' :



Informations utiles :

- Tout comme pour l'horloge, la définition des faces du dé ont été faites dans le squelette de code fourni. Elles sont écrites sous forme de listes pour être directement compatibles avec la méthode '**set_pixels()**' afin de faciliter leur affichage.
- Faire afficher un message sur la matrice au démarrage du programme comme par exemple : « secouer légèrement la carte pour lancer le dé !! ».
- Comme pour l'exercice précédent, on se limitera à l'utilisation des axes x et y de l'accéléromètre.
- Il vous faudra définir un seuil à partir duquel on pourra considérer que la carte a été secouée et effectuer vos tests en fonction de ce seuil. Ce test doit se faire sur les deux axes et dans les deux directions de chaque axe. Vous pouvez utiliser la méthode '**abs()**' pour simplifier vos tests (cf. https://docs.python.org/3/library/functions.html).
- Pour la détermination d'une face aléatoire, vous aurez besoin de la méthode '**randint()**' utilisée dans l'exercice 1.
- Une première version consiste à afficher une face du dé dès que la carte est secouée.
- Si vous avez le temps, vous pouvez mettre en œuvre une seconde version où l'on simule le roulement du dé en affichant une succession de faces aléatoires avec un temps de plus en plus long (fonction '**sleep**') entre chaque jusqu'à la stabilisation sur le résultat final du dé.

Faire valider le fonctionnement de votre programme par un enseignant.

EXERCICE 8 : 'Ex08.py'

Objectif de l'exercice :

Cet exercice ne va pas vous demander beaucoup d'effort car une seule ligne de code vous sera demandée. Il s'agit ici de créer une interface graphique qui affiche une photographie. A chaque endroit de la photo où l'on va cliquer avec la souris, on va récupérer la valeur de la couleur du pixel correspondant et afficher cette couleur sur la matrice de la carte 'Sense Hat'.

Informations utiles :

- Vous allez pouvoir ouvrir le programme '**Ex08.py**' qui est quasiment complet. Ce dernier utilise le package '**guizero**' qui permet de créer des interfaces graphiques (déjà utilisé au cours du TP précédent).

- La seule ligne de code que vous devez ajouter est celle de la ligne n°28. L'instruction à ajouter est celle qui va permettre de modifier la totalité de la couleur de la matrice. A vous de trouver laquelle (en vous aidant de la section 'LED matrix' de la page web de la librairie 'Sense Hat').
- Vous pouvez aussi lors de vos tests, changer de photographies pour vérifier que votre programme fonctionne quel que soit le fichier utilisé. Il suffit juste de modifier le chemin du fichier de la ligne n°15 (plusieurs photographies sont fournies dans le dossier '**images**' du TP3)

Pensez à faire valider le fonctionnement de votre programme avant de finir cette séance.