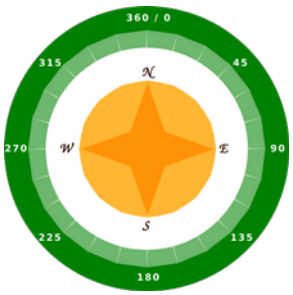
TP11 – UTILISATION DE LA FONCTION COMPAS

Le but des exercices de cette séance de travaux pratiques est de manipuler la fonction compas fournie avec l'unité de mesure inertielle présente sur la carte Sense Hat.



Le Sense Hat est équipé d'une puce IMU (Inertial Measurement Unit) qui comprend un ensemble de capteurs permettant de détecter les mouvements :

- Un gyroscope (pour détecter la position de la carte)
- Un accéléromètre (pour détecter les mouvements)
- un magnétomètre (pour détecter les champs magnétiques)

C'est le magnétomètre qui permet d'obtenir une mesure de l'orientation.

On considère que l'ouverture d'un nouveau projet dans l'environnement de développement Visual Studio Code est acquise. En cas de difficulté avec la gestion d'un projet, reprendre le TP01 à partir de l'étape 3 de la partie 1.

Éteindre correctement le système cible en fin de séance

À partir du terminal, taper la commande suivante :

```
$ sudo poweroff
```

Liste des fonctions à utiliser dans ce TP

En plus des fonctions déjà utilisées lors des TP précédents, nous allons faire appel à de nouveaux sous-programmes de la bibliothèque.

- Initialisation de l'unité de mesure inertielle.

  ```
  void senseSetIMUConfig(bool magn, bool gyro, bool accel);
  ```

- Mesure de l'orientation en degrés

  ```
  double senseGetCompass();
  ```

- Affichage de la direction cardinale sur la matrice de leds

  ```
  void senseShowMessage(string msg);
  ```

- Passage d'une chaîne de caractères en langage C++ au langage C

  ```
  // Déclaration de variables
  string direction;

  // Instructions
  senseShowMessage(direction);
  ```

- Pour faire une temporisation avec une durée en millisecondes

  ```
  sleep_for(milliseconds(duree));  // Bloque l'exécution pendant 'duree' en ms
  ```

Exercice 1 – Afficher une série de mesures de cap

L'objectif est d'écrire un programme principal qui affiche une série de 60 mesures de cap. Une mesure est appelée toutes les 500ms

Préparation (avant la séance de TP)

1. Donner la déclaration des constantes de type ENTIER :
   - **INIT_MESURES** définit le nombre de mesures « à vide »
   - **MAX_MESURES** définit le nombre de mesures utiles
2. Donner la vue externe et l'algorithme de la fonction d'initialisation **initCompas()** qui fait appel à **senseSetIMUConfig()** puis lance une série de mesures nécessaires aux calculs de la bibliothèque de gestion du capteur.
   - Le nombre de ces mesures est donné par la constante **INIT_MESURES**
   - La valeur du premier paramètre active le magnétomètre
   - La valeur du deuxième paramètre active le gyromètre pour la fusion des données
   - La valeur du troisième paramètre désactive la fonction accéléromètre
3. Donner la vue externe et l'algorithme du programme principal qui initialise le composant IMU via l'appel à **initCompas()** puis lance la série de **MAX_MESURES** toutes les 500ms à l'aide de la fonction **senseGetCompass()**.

   Chaque mesure est affichée à la console avec un format à 2 décimales.

Rechercher dans la documentation de la bibliothèque **iomanip**, les modes d'utilisation des « manipulateurs » **fixed** et **setprecision()**
Voir : Reference <iomanip> setprecision

Réalisation (en séance de TP)

1. Ouvrir une session SSH sur la carte cible. À partir du terminal, lancer la commande :

   ```
   getLab.sh tp08-ex1
   ```

   Une fois le dossier créé, aller dans le menu Fichier et sélectionner Ouvrir le dossier. Choisir le dossier **tp08-ex1**.
2. Codez en C++ le programme puis testez-le.
3. Repérer les différents points cardinaux en tournant la carte Sense Hat.
   Tester l'opération suivante et évaluer la validité des mesures.

   ```
   cap = 360 - senseGetCompass();
   ```
4. Faites valider.

Que peut-on conclure sur la validité des mesures en comparant celles-ci avec les résultats d'une application de smartphone de type boussole.

Voir la procédure de calibration à la fin de ce document.

Exercice 2 – Affichage de la direction cardinale

On veut compléter le programme précédent en affichant la direction cardinale à la console et sur la matrice de leds de la carte Sense Hat.

Comme l'affichage par défilement sur la matrice de leds est assez lent, on effectue la mesure de l'orientation de la carte toutes les secondes.

Voici le tableau des directions cardinales.

Direction cardinale	Angle min	Angle max
N	337,5 → 360	0 → 22,5
N.E.	22,5	67,5
E	67,5	112,5
S.E.	112,5	157,5
S	157,5	202,5
S.0.	202,5	247,5
O	247,5	292,5
N.0.	292,5	337,5

Préparation (avant la séance de TP)

Donner la vue externe et l'algorithme du programme qui affiche une série de **MAX_MESURES** en indiquant la direction cardinale de la carte Sense Hat.

Réalisation (en séance de TP)

1. À partir du terminal, lancer la commande :

   ```
   getLab.sh tp08-ex2
   ```

   Une fois le dossier créé, aller dans le menu Fichier et sélectionner Ouvrir le dossier et choisir le dossier **tp08-ex2**.
2. Copiez puis collez dans ce fichier le code du fichier précédent (**tp08-ex1**).
3. Modifier le programme principal afin d'afficher le cap et la direction cardinale à la console et/ou sur la matrice de leds avec la fonction **senseShowMessage()**.
   Supprimer la temporisation entre chaque mesure .
4. Fixer la valeur de la constante **MAX_MESURES** (60 pour une minute).
5. Comparer la validité des mesures relativement à l'exercice 1 et à celles d'une application compas de smartphone. Il est conseillé de faire faire un tour complet de la carte Sense Hat avant d'estimer la validité des mesures.
6. Testez et faites valider.

Exercice 3 – Orientation de la carte Sense Hat en fonction d'une série de caps

On veut maintenant orienter la carte en fonction d'une série de caps définis dans un tableau prédéfini.

Préparation (avant la séance de TP)

1. Donner la vue externe puis l'algorithme du sous-programme **lireCaps()** qui remplit un tableau à partir des caps saisis par l'utilisateur.
   Le nombre de directions à atteindre est défini par la constante **MAX_DIR**.
   En fonction du temps restant en séance, on pourra tester la validité de chaque saisie. Voir dernier point de la partie réalisation en séance.
2. Donner la vue externe puis l'algorithme du sous-programme **mesureCap()** qui renvoie un booléen à VRAI si le cap mesuré correspond à la direction souhaitée.

   Ce sous-programme reprend les traitements du programme principal de l'exercice précédent.

   Le prototype du sous-programme est le suivant :

   ```
   bool mesureCap(string destination)
   ```
3. Donner la vue externe du programme principal qui assure les traitements suivants :
   - Saisie de la liste des caps à atteindre avec l'appel à **lireCaps()**
   - Initialisation de la centrale inertielle.
   - Attendre un appui sur le joystick pour lancer les mesures.
   - Pour chaque direction définie dans le tableau, appeler le sous-programme **mesureCap()** tant que la direction ne correspond pas à celle définie dans le tableau.
     Une fois l'orientation voulue obtenue, on affiche le nouveau cap et on attend 2 secondes.

Réalisation (en séance de TP)

1. À partir du terminal, lancer la commande :

   ```
   getLab.sh tp08-ex3
   ```

Une fois le dossier créé, aller dans le menu Fichier et sélectionner Ouvrir le dossier et choisir le dossier **tp08-ex3.**

2. Copiez-collez les instructions utiles de l'exercice précédent**.**
3. Codez en C++ le sous-programme **lireCaps()**.
4. Faire un programme principal qui permet de tester cette fonction en affichant la liste saisie par l'utilisateur.
5. Testez et faites valider.

6. Ajouter le code C++ du sous-programme **mesureCap()** et compléter les traitements du programme principal.
7. Il est possible d'afficher une indication de la mesure en couleur en fonction de la validité de la mesure.
   Par exemple, on peut utiliser la fonction de la bibliothèque suivante pour allumer la matrice de leds en rouge, orange ou vert en fonction des plages de mesures.
   ```
   void senseRGBClear(uint8_t R, uint8_t G, uint8_t B);
   ```
8. Tester et faites valider.
9. Coder en C++ un sous-programme **validSaisie()** qui renvoie un booléen à VRAI si la chaîne de caractères saisie par l'utilisateur dans le sous-programme **lireCaps()** correspond bien à un élément de liste des directions cardinales définie dans le tableau de l'exercice 2.
   Le prototype de ce sous-programme est :
   ```
   bool validSaisie(string direction);
   ```

Procédure de calibration de la bibliothèque RTIMU

La bibliothèque RTIMU fournit un programme appelé **RTIMULibCal** qui permet de calibrer le magnétomètre.

Depuis l'invite de commande du terminal de l'exercice en cours, on commence par exécuter les deux instructions suivantes :

```
cp -a /usr/share/librtimulib-utils/RTEllipsoidFit .
cd RTEllipsoidFit/
```

Ensuite, on lance le programme :

```
RTIMULibCal
RTIMULibCal - using RTIMULib.ini
Settings file not found. Using defaults and creating settings file
Detected LSM9DS1 at standard/standard address
Using fusion algorithm RTQF
min/max compass calibration not in use
Ellipsoid compass calibration not in use
Accel calibration not in use
LSM9DS1 init complete

Options are:

  m - calibrate magnetometer with min/max
  e - calibrate magnetometer with ellipsoid (do min/max first)
  a - calibrate accelerometers
  x - exit

Enter option:
```

Les deux options utiles dans notre cas sont 'm' puis 'e'.

À l'issue des deux étapes de calibration, le fichier **RTIMULib.ini** sera mis à jour et votre programme utilisera les paramètres définis lors de la calibration.

1. On commence par l'option 'm' du magnétomètre.
    - L'appui sur une touche déclenche la recherche des extremums du magnétomètre.
    - Il faut effectuer une série de mouvements avec le capteurs jusqu'à ce que l'affichage n'évolue plus.
    - On appuie sur la touche 's' pour sauvegarder ces premiers résultats.

```
Enter option: m

Magnetometer min/max calibration
--------------------------------
Waggle the IMU chip around, ensuring that all six axes
(+x, -x, +y, -y and +z, -z) go through their extrema.
When all extrema have been achieved, enter 's' to save, 'r' to reset
or 'x' to abort and discard the data.

Press any key to start…
```

2. On pass ensuite à l'option 'e' qui complète la première partie avec les mesures du gyromètre.
    - Ici, il faut réaliser autant de mouvements que nécessaire pour compléter une série de mesures suivant tous les axes possibles
    - Les mesures s'arrêtent dès lors que tous les axes ont été couverts par les mouvements
3. On peut quitter le programme avec la touche 'x' après avoir eu la confirmation que le fichier RTIMULib.ini a bien été mis à jour.

```
Processing ellipsoid fit data...
octave: X11 DISPLAY environment variable not set
octave: disabling GUI features
Mat size = 3
Mat size = 5358

Ellipsoid fit completed - saving data to file.
Options are:

  m - calibrate magnetometer with min/max
  e - calibrate magnetometer with ellipsoid (do min/max first)
  a - calibrate accelerometers
  x - exit

Enter option: x
RTIMULibCal exiting
```

4. Enfin on copie le fichier de calibration dans le dossier du projet de l'exercice en cours. L'instruction ci-dessous copie le fichier dans le répertoire de niveau supérieur.

```
cp RTIMULib.ini ..
```

Voici une copie d'écran de la vue dossier de Visual Studio Code qui montre que le fichier **RTIMULib.ini** est bien présent dans le dossier du projet **tp8-ex1.**