TP07 – TABLEAU À UNE DIMENSION

Le but des exercices de cette séance de travaux pratiques est de réaliser un module de « punching-table » ou vous pourrez taper sur la table plusieurs fois avec un total de 8 essais, réaliser un classement des essais et un affichage sur les leds du SenseHat. Ces exercices vous permettront aussi de vous familiariser avec le codage en C++ des éléments suivants :

- Définition et appel de sous-programmes
- Utilisation de tableau à une dimension

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

- Pour allumer le pixel aux coordonnées (x ,y) de la couleur formée par les valeurs de R, G et B :

```
senseSetRGBpixel(int x, int y, uint8_t R, uint8_t G, uint8_t B)
```

- Pour faire une temporisation de duree millisecondes

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

- Pour récupérer les données de l'accéléromètre

```
senseGetAccelG(double &x, double &y, double &z)
```

  Les paramètres x, y et z sont des nombres réels au format double représentation.

Exercice 1 – Créer un histogramme à leds

L'objectif est d'écrire un sous-programme afficheHistogramme et un programme principal qui allume les colonnes de leds du SenseHat sous forme d'un histogramme représentant les valeurs contenues dans un tableau **tabVal** de 8 valeurs strictement comprises entre 0 et 7 qui seront saisies dans la console.

Préparation (avant la séance de TP)

1. Donner la déclaration de la constante **MAX** qui définit le nombre de valeurs stockées dans le tableau **tabVal**. La valeur de la constante est 8. Elle est visible des sous-programmes et du programme principal.
2. Donner la vue externe et l'algorithme du sous-programme **saisirValeurs()** qui demande à l'utilisateur les valeurs comprises entre 0 et 7 à stocker dans chaque élément du tableau **tabLues** passé en paramètre formel.
   Le code du sous-programme doit vérifier que l'utilisateur saisit bien une valeur comprise entre 0 et 7.
3. Donner la vue externe et l'algorithme du sous-programme **afficherHistogramme()** qui allume pour chaque colonne de la matrice de pixels, le nombre de leds correspondant à chaque valeur du tableau **tabAff** passé en paramètre formel.
4. Faire l'algorithme de ce programme qui appelle les deux sous-programmes.

Réalisation (en séance de TP)

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

   ```
   getLab.sh tp7-ex1
   ```

   Une fois le dossier créé, aller dans le menu Fichier et sélectionner Ouvrir le dossier. Choisir le dossier tp7-ex1.
2. Codez en C++ le sous-programme **saisirValeurs()** et appelez le dans un premier programme principal qui affiche les valeurs du tableau **tabVal** directement à la console.
3. Codez en C++ le sous-programme **afficherHistogramme()** qui remplace l'affichage des valeurs à la console par l'affichage sur les leds.
4. Testez et faites valider.

Exercice 2 – Calculer et enregistrer la norme minimale du vecteur accélération

L'objectif est de réaliser un détecteur de pics maximal d'accélération pour un jeu de « Punching-table » dans une fête foraine. Pour cela il faudra lire les trois sorties (x, y et z en G) de l'accéléromètre du SenseHat et enregistrer la valeur minimale de la norme Euclidienne (norme=$\sqrt{x^2+y^2+z^2}$) de l'accélération réalisée sur 100 mesures en imposant un délai minimal de 100ms entre chaque mesure.

Préparation (avant la séance de TP)

1. Donnez la déclaration de la constante **MAX_NORME_ACCEL** dont la valeur initiale est fixée à 10G. La valeur de cette constante sert à initialiser la recherche du minimum de la norme de l'accélération.
2. Donnez la vue externe et l'algorithme du sous-programme **calculNormeAccel()** qui effectue le calcul de la norme donné ci-dessus à partir des 3 paramètres formels **xa**, **ya** et **za**. Les paramètres sont nécessairement de type double. Le résultat du calcul est retourné au programme appelant.
3. Donnez la vue externe et l'algorithme du sous-programme **minNormeAccel()** qui renvoie la valeur minimale de la comparaison entre les deux paramètres formels **norme** et **min**.

Tous les paramètres utilisés dans les deux sous-programmes sont de type double.

4. Faire l'algorithme de ce programme.

Réalisation (en séance de TP)

1. À partir du terminal, lancer la commande :

   `getLab.sh tp7-ex2`

   Ouvrir le dossier et choisir le dossier tp7-ex2.
2. Copiez dans le programme principal le code qui mesure et affiche la norme de la mesure d'accélération du SenseHat selon les 3 composantes x, y et z.

```
sleep_for(milliseconds(100));
if (senseGetAccelG(x, y, z)) {
      cout << showpos << fixed << setprecision(6)
            << "x = " << x
            << " y = " << y
            << " z = " << z;

      normeAccel = calculNormeAccel(x, y, z);
      normeMin = minNormeAccel(normeAccel, normeMin);

      cout << "\t-> norme accélération : " << normeAccel << endl;
}
else
      cout << "Pas de mesure disponible." << endl;

cout << "La norme minimale d'accélération mesurée est : " << normeMin << endl;
```

3. Codez les deux sous-programmes appelés dans l'extrait : **calculNormeAccel()** et **minNormeAccel()**.

   Déclarez les 3 variables **x**, **y** et **z** de type réel double ainsi que les extremums de la norme d'accélération.

   Testez l'exécution du code et tapez sur la table en même temps. Que se passe-t-il ?

5. Modifiez le programme principal pour réaliser 100 lectures de l'accéléromètre du SenseHat toutes les 100ms.
   Testez et affichez dans la console les valeurs pour ces 100 mesures ainsi que la valeur minimale de la série.

6. Testez et faites valider.

Exercice 3 – Conversion linéaire de la norme en G sur 8 bits

L'objectif de cet exercice est de convertir la mesure de la norme de l'accéléromètre sur 8 bits afin de l'afficher sur les leds du SenseHat. Nous allons supposer 8 essais démarrés par l'utilisateur et à chaque nouvel essai 10 mesures de l'accélération seront effectuées toutes les 100ms.

La valeur minimale de la norme à considérer est de 0.1G à cette valeur les 8 leds d'une ligne devront être allumées. La valeur maximale de la norme est de 1, à cette valeur aucune led ne devra être allumée.

Préparation (avant la séance de TP)

1. Calculer le coefficient directeur a et l'ordonnée b à l'origine de l'équation :

   y_bits = a * acc_norm + b

   Cette équation est utilisée pour convertir la mesure de norme en G en nombre de bits à allumer.

   L'intervalle des valeurs de y_bits est [0;7]

   L'intervalle des valeurs de acc_norm est [0,1;1]

2. Donnez la vue externe puis l'algorithme du sous-programme **conversionLineaire()**

Réalisation (en séance de TP)

1. À partir du terminal, lancer la commande :

   ```
   getLab.sh tp7-ex3
   ```

   Une fois le dossier créé, aller dans le menu Fichier et sélectionner Ouvrir le dossier et choisir le dossier tp7-ex3.

2. Codez en C++ le sous-programme **conversionLineaire()**

3. Modifiez le programme principal de façon à réaliser 8 essais ; soit 8 séries de 10 mesures toutes les 100ms.

   Le minimum de la norme d'accélération est calculé pour chaque série de 10 mesures.

   Le nombre de leds à allumer est stocké dans le tableau **tabVal** pour chaque essai après conversion linéaire du minimum obtenu.

   Une fois les 8 essais réalisés, on affiche l'histogramme des valeurs stockées dans le tableau.

4. Testez et faites valider.

Exercice 4 (Bonus) – Tri du tableau des essais

Une fois le tableau des 8 essais complété dans l'ordre chronologique, on veut maintenant trier ces résultats dans l'ordre croissant. Rechercher une solution qui permette de trier les valeurs du tableau tabVal dans l'ordre croissant et de les afficher à l'aide du sous-programme **afficherHistogramme()**.