CENTRE SCOLAIRE SAINTE-JULIENNE

TA 16 – Utilisation du son

Exercices Thymio – Série 4 – Enoncés

Mise en situation

Une institutrice primaire souhaite faire découvrir la programmation robotique à ses élèves, elle t'a demandé de lui fournir les solutions qui répondent à ses besoins.

Objets d'apprentissage

Appliquer	Transférer
• Appliquer les règles de syntaxe et les conventions spécifiques à un langage de programmation • Déclarer une variable en appliquant les règles et les conventions • Utiliser des fonctions prédéfinies (bibliothèque) en vue d'animer un objet réel ou virtuel • Tester la séquence d'instructions conçue • Commenter des lignes de codes	• Écrire un logigramme d'actions d'un objet réel ou virtuel intégrant structure répétitive et opérateurs logiques • Améliorer une séquence pour répondre à un besoin défini • Corriger une séquence défaillante proposée pour atteindre un but défini
Connaître	
• Expliquer la notion d'expression • Expliquer la notion d'instruction • Expliquer la notion de séquence • Caractériser les types de données • Expliquer la notion de variable • Expliquer la notion d'affectation • Différencier les opérateurs logiques dont "et", "ou", "non" • Expliquer la notion de répétition	

Tâches à accomplir:

- Dessiner sur papier l'enchainement des traitements;
- Tester et corriger avec Aseba votre solution;
- Enregistrer la solution finale.

Actuateurs et capteurs à utiliser:

Les LEDs RGB

Il y a deux LEDs RGB sur le robot pilotées ensemble, ce sont celles qui indiquent le comportement du robot. Deux autres LEDs RGB dessous sont pilotables séparément.

Activation par défaut: éteintes dans le mode Aseba.

leds.top(red, green, blue) commande les valeurs de rouge, vert et bleu respectivement, pour les LEDs du dessus.

leds.bottom.left(red, green, blue) commande les valeurs de rouge, vert et bleu respectivement, pour la LEDs du dessous à gauche.

leds.bottom.right(red, green, blue) commande les valeurs de rouge, vert et bleu respectivement, pour la LEDs du dessous à droite.

Timer (minuterie)

Thymio fournit **deux timers** défini par l'utilisateur. Un tableau de 2 valeurs, **timer.period**, permet de spécifier la période des timers:

Timer.period[0]: période du timer 0 en millisecondes

Timer.period[1]: période du timer 1 en millisecondes

Lorsque le délai expire, le timer génère un événement **timer0** respectivement **timer1**.

Boutons

Thymio possède **5 boutons capacitifs** correspondant aux flèches et au bouton central. Un tableau de 5 variables, buttons.binary, contient l'état de ces boutons (1 = appuyé, 0 = relâché):

- ***button.backward***: flèche arrière
- ***button.left***: flèche gauche
- ***button.center***: bouton central
- ***button.forward***: flèche avant
- ***button.right***: flèche droite

Thymio met à jour ce tableau à une fréquence de 20 Hz, et génère l'événement button après chaque mise à jour. En outre, pour chacune de ces touches, quand elle est appuyée ou relâchée, un événement correspondant avec le même nom est émis.

Détection de l'intensité du son

Le Thymio peut détecter lorsque le bruit ambiant est au-dessus d'une intensité donnée et émet un événement. La variable **mic.intensity** montre l'intensité du microphone en cours, tandis que la variable **mic.threshold** contient la limite de l'intensité pour l'événement. Si **mic.intensity** est au-dessus de **mic.threshold** , l'événement **mic** est généré.

Lecture de sons

Vous pouvez jouer des sons de synthèse ou du système. Lorsque Thymio fini la lecture d'un son demandée par ASEBA, il déclenche l'événement **sound.finished**. Il ne se déclenche pas un événement si la lecture est interrompue ou si un nouveau son est joué.

Son de synthèse

La fonction native **sound.freq** joue une fréquence, spécifiée en Hz, pour une certaine durée, spécifié dans 1/60 s. Une durée égale à **0 joue le son à l'infini** et une durée de **-1 arrête le son**.

Modification de l'onde primaire

La génération de sons de synthèse fonctionne par ré-échantillonnage d'une onde primaire. Par défaut, l'onde est triangulaire, mais vous pouvez définir votre propre onde à l'aide de la fonction native **sound.wave**. Cette fonction prend en entrée un tableau de 142 échantillons, avec des valeurs de -128 à 127. Ce tableau doit représenter une onde tonique de la fréquence spécifiée dans **sound.freq**. Comme Thymio joue des sons à 7812,5 Hz, ce tableau est joué entièrement à la fréquence de 7812.5/142 = ~ 55 Hz. Jouer un son de fréquences plus élevées fait sauter des échantillons dans le tableau.

Ex 1

Simuler une sirène de pompier.
Afficher la couleur « Rouge » au départ.
Afficher la couleur « Blanche » et jouer un son d'une fréquence de 700 et d'une durée illimitée quand le bouton de droite est appuyé.
Afficher la couleur « Bleue » et jouer un son d'une fréquence de 500 et d'une durée illimitée quand le bouton de gauche est appuyé.
Afficher la couleur « Verte » et stopper le son quand le bouton du centre est appuyé.
Enregistrer la solution dans un fichier nommé EX01.aesl qui se retrouve dans un dossier nommé Série4.

Ex 2

Reprendre l'Ex 1.
Utiliser un timer qui fait varier automatiquement chaque seconde les comportements lumineux et sonores.
Enregistrer la solution dans un fichier nommé EX02.aesl qui se retrouve dans un dossier nommé Série4.

Ex 3

Jouer une mélodie lorsqu'on claque dans les mains.
Adapter l'intensité de la couleur « Blanche » en fonction de la note jouée.
Enregistrer la solution dans un fichier nommé EX03.aesl qui se retrouve dans un dossier nommé Série4.

Remarques:

Utiliser le code suivant pour le calcul de l'onde sonore:

```
var wave[142]
var i
var wave_phase
var wave_intensity

for i in 0:141 do
wave_phase = (i-70)*468
call math.cos(wave_intensity, wave_phase)
wave[i] = wave_intensity/256
end
call sound.wave(wave)
```

Paramétrer la sensibilité de détection du son:

```
mic.threshold = 100
```

Définir les constantes suivantes:

```
DO   440              FA   740              SI   1040
RE   540              SOL  840              UN   35
MI   640              LA   940              DEUX 60
```

Utiliser la partition suivante (tableaux à définir également):

```
call math.copy(notes[0:27], [DO, DO, RE, DO, FA, MI, DO, DO,
RE, DO, SOL, FA, DO, DO, RE, DO, LA, FA, MI, RE, DO, DO, RE,
DO, LA, FA, SOL,FA])
call math.copy(durations[0:27], [UN, UN, UN, UN, UN, DEUX, UN,
UN, UN, UN, UN, DEUX, UN, UN, UN, UN, UN, UN, DEUX, UN, UN
, UN, UN, UN, UN, UN, UN])
```

Utiliser une variable play qui adopte 3 significations:

-1 Arrêter la musique
0 (Ré)initialiser la musique
1 Jouer la musique

A la fin d'une note jouée, il faut passer à la note suivante, il y en a 28 au total.