http://irem.univ-reunion.fr/spip.php?article776

Compter en binaire et algo à gogo avec le robot Thymio

- Lycée et post-bac
- Informatique et sciences du numérique
-

Publication date: lundi 10 novembre 2014

Le robot Thymio permet de découvrir la programmation de manière très progressive, grâce à une interface de programmation visuelle, couplée à une interface textuelle.

[http://irem.univ-reunion.fr/local/cache-vignettes/L367xH363/2013-12-12-athymioii-82bfb.png] Le robot Thymio II est produit et vendu par l'association Mobsya, et, pour environ une centaine d'euros, on peut le commander ici.
Ce petit robot permet une réelle approche scientifique de la robotique pour nos élèves, possédant de nombreux capteurs (capteurs infrarouge, capteur de température, micro, capteurs mesurant les 3 composantes de l'accélération, 5 boutons on-off) et de nombreux actuateurs (nombreuses leds dont une led RGB qui permet donc d'appréhender le mélange des couleurs, un moteur sur chaque roue, un haut-parleur). L'apprentissage de sa programmation se fait de manière très progressive, grâce à une interface de programmation visuelle, qui crée les lignes de codes correspondantes, ce qui permet d'affiner ensuite le code.

Son site de présentation - et de référence d'ailleurs -, permet assez rapidement de réaliser le champ d'applications potentielles qu'il vous sera possible de réaliser grâce à ce petit robot.

Au lycée Antoine Roussin, nous en avons commandé 5 afin de pouvoir les utiliser dès la rentrée 2013 en spécialité ISN en Terminale S.
Un projet Bac a d'ailleurs été réalisé sur Thymio. Les élèves ont réalisé un suivi de ligne à l'aide du robot. Une vidéo est visible ici.

[http://irem.univ-reunion.fr/local/cache-vignettes/L300xH231/dessin-thymio-300-d447c.jpg] Vous pouvez voir ce schéma pour avoir une liste complète de ses capteurs et actuateurs.
Thymio permet une approche multi-disciplinaire de la robotique, et c'est ce qui nous a réellement séduits, mon collègue de physique et moi-même.

Et, cerise sur le gâteau, Thymio II est un projet open source et open hardware !

Nous venons d'en commander 6 de plus, afin de les utiliser dès la seconde...

Balbutiements

Au démarrage, le Thymio possède un certain nombre de comportements usines.

Au début de l'utilisation du Thymio avec les élèves, nous leur avons demandé d'illustrer au maximum de commentaires les programmes qu'ils lisaient, et d'écrire dans la mesure du possible des algorithmes décrivant les comportements du robot.
PNG

Voici un exemple de programme du Thymio parlant anglais illustré de commentaires :
PNG
La deuxième page de commentaires se trouve ici.

Je voudrais maintenant simplement illustrer le tutoriel de prise en mains du robot.

Ballets en couleur

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH165/couleursrgb400-de3fe.png]
Grâce à la led RGB, on pourra commencer à mélanger les couleurs.

On pourra aussi faire des programmes de base qui consistent à tester divers comportements du Thymio.
Il peut vous suivre comme un petit chien, ou au contraire Thymio vous fuira comme la peste.
Il peut éviter de tomber de la table, etc...

Premier ballet...
... ou *Première implémentation du Si... Alors... , avec activation des moteurs*.

[http://irem.univ-reunion.fr/local/cache-vignettes/L88xH400/couleursetmarche200-2-52c0c.png]	[http://irem.univ-reunion.fr/local/cache-vignettes/L188xH400/couleursetmarche-codesource-2-fa000.png]

Le code source sur l'image ci-dessus est automatiquement généré, dès que l'on active des couples d'événements-action dans l'interface de programmation visuelle (VPL). Il peut bien évidemment être nettoyé.
[http://irem.univ-reunion.fr/local/cache-vignettes/L241xH400/couleursetmarche-codesourcepropre-35e69.png]

En activant le mode avancé de programmation visuelle du Thymio, on a accès à un minuteur, et à des boutons d'état (qui activent en fait des leds). Cela va nous permettre d'améliorer le programme précédent et d'amorcer un ballet du Thymio qui devient petit à petit un petit robot danseur...

Première boucle infinie
Faisons tout d'abord réaliser une boucle infinie au robot, rendue possible grâce au timer (premier exercice sur les boucles du tutoriel) :

```
1.
    var state[4] = [0,0,0,0]
2.
    var new_state[4] = [0,0,0,0]
3.
    timer.period[0] = 0
4.
    call sound.system(-1)
5.
    call leds.top(0,0,0)
6.
    call leds.bottom.left(0,0,0)
7.
    call leds.bottom.right(0,0,0)
8.
    call leds.circle(0,0,0,0,0,0,0,0)
9.
    sub display_state
10.
    call leds.circle(0,state[1]*32,0,state[3]*32,0,state[2]*32,0,state[0]*32)
11.
    onevent buttons
12.
    if state[0] == 0 then
13.
    motor.left.target = -500
14.
    motor.right.target = 0
15.
    timer.period[0] = 2000
16.
    end
17.
    if state[0] == 1 then
18.
    motor.left.target = 0
19.
    motor.right.target = -500
20.
    timer.period[0] = 3000
21.
    end
22.
    call math.copy(state, new_state)
23.
    callsub display_state
24.
    onevent timer0
25.
    timer.period[0] = 0
26.
    if state[0] == 0 then
```

```
27.
    new_state[0] = 1
28.
    end
29.
    if state[0] == 1 then
30.
    new_state[0] = 0
31.
    end
32.
    call math.copy(state, new_state)
33.
    callsub display_state
```

[http://irem.univ-reunion.fr/local/cache-vignettes/L168xH400/boucleinfinie-tourneg2s-tourned3s-200px-18f34.png]

Deuxième ballet

Et voici un début de programme pour un joli ballet :

[http://irem.univ-reunion.fr/local/cache-vignettes/L83xH400/balletencouleurpartie1-200px-02761.png]	

1.

var state[4] = [0,0,0,0]

```
2.
    var new_state[4] = [0,0,0,0]
3.
    timer.period[0] = 0
4.
    call sound.system(-1)
5.
    call leds.top(0,0,0)
6.
    call leds.bottom.left(0,0,0)
7.
    call leds.bottom.right(0,0,0)
8.
    call leds.circle(0,0,0,0,0,0,0,0)
9.
    sub display_state
10.
    call leds.circle(0,state[1]*32,0,state[3]*32,0,state[2]*32,0,state[0]*32)
11.
    onevent buttons
12.
    if button.center == 1 then
13.
    call leds.top(0,0,0)
14.
    call leds.bottom.left(0,0,0)
15.
    call leds.bottom.right(0,0,0)
16.
    motor.left.target = 0
17.
    motor.right.target = 0
18.
    end
19.
    if button.forward == 1 and state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3] ==
    0 then
20.
    timer.period[0] = 1000
21.
    motor.left.target = 500
22.
    motor.right.target = 500
23.
    call leds.top(0,32,0)
24.
    end
25.
    if state[0] == 1 and state[1] == 1 and state[2] == 1 and state[3] == 1 then
26.
    timer.period[0] = 1000
27.
```

```
    call leds.top(0,0,0)
28.
    call leds.bottom.left(0,0,0)
29.
    call leds.bottom.right(0,0,0)
30.
    motor.left.target = 0
31.
    motor.right.target = 0
32.
    end
33.
    if button.backward == 1 and state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3]
   == 0 then
34.
    timer.period[0] = 1000
35.
    motor.left.target = -500
36.
    motor.right.target = -500
37.
    call leds.top(0,0,32)
38.
    end
39.
    if button.right == 1 and state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3] == 0
  then
40.
    timer.period[0] = 1000
41.
    motor.left.target = 500
42.
    motor.right.target = 0
43.
    call leds.bottom.left(0,32,16)
44.
    call leds.bottom.right(0,32,16)
45.
    end
46.
    if button.left == 1 and state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3] == 0
  then
47.
    timer.period[0] = 1000
48.
    motor.left.target = 0
49.
    motor.right.target = 500
50.
    call leds.bottom.left(0,16,32)
51.
    call leds.bottom.right(0,16,32)
```

```
52.
        end
53.
        call math.copy(state, new_state)
54.
        callsub display_state
55.
        onevent timer0
56.
        timer.period[0] = 0
57.
        if state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3] == 0 then
58.
        new_state[0] = 1
59.
        new_state[1] = 1
60.
        new_state[2] = 1
61.
        new_state[3] = 1
62.
        end
63.
        if state[0] == 1 and state[1] == 1 and state[2] == 1 and state[3] == 1 then
64.
        new_state[0] = 0
65.
        new_state[1] = 0
66.
        new_state[2] = 0
67.
        new_state[3] = 0
68.
        end
69.
        call math.copy(state, new_state)
70.
        callsub display_state
```

Ici, l'intervention de l'humain est nécessaire, mais on peut très rapidement améliorer ce programme pour qu'une intervention humaine ne soit plus nécessaire, en mixant par exemple les deux programmes précédents...
Le ballet complet se trouve dans le port-folio de cet article car, seule une direction est implémentée sur l'image

ci-dessus.

Exercice : Créer un comportement du robot correspondant à cette image puis simplifier le code source au maximum, car l'interface VPL crée des redondances dans le code, à chaque fois par exemple que l'on teste la même condition (plusieurs **if** pour la même condition, c'est inutile, il faudra tout mettre dans le même bloc).

Compter avec le Thymio

[http://irem.univ-reunion.fr/local/cache-vignettes/L200xH199/les16etatsduthymio-66799.png]
Ce qui m'intéresse ici, c'est de compter avec Thymio. Le Thymio possède un cercle de 8 leds sur sa face supérieure. Avec 4 leds seulement, on peut décrire 16 états différents, et donc compter en binaire (led éteinte = 0, led allumée = 1) en base 16.

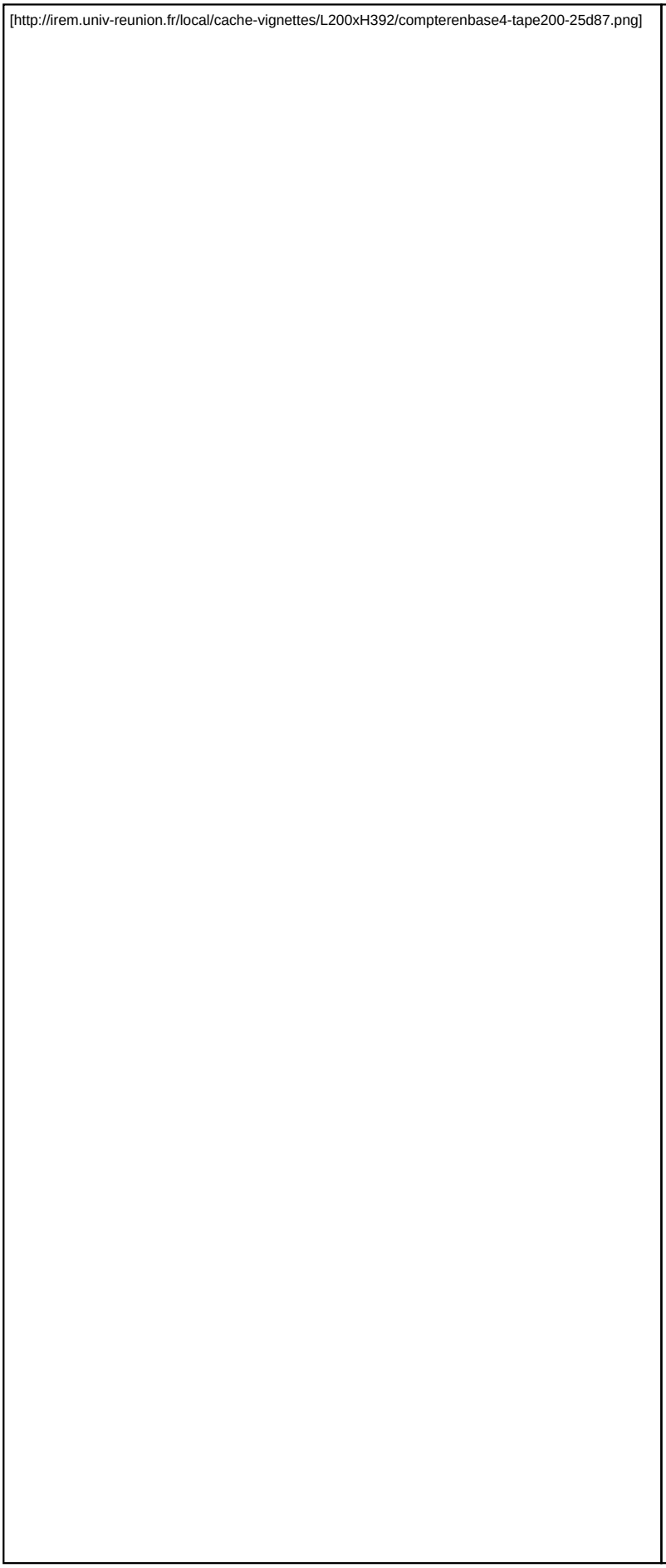
Compter en unaire modulo 2

On va afficher le reste de la division euclidienne d'un nombre par 2.
On choisit une led. Elle sera éteinte (blanc) si on tape dans les mains un nombre pair de fois et allumée (orange) si on tape dans les mains un nombre impair de fois.

[http://irem.univ-reunion.fr/local/cache-vignettes/L200xH229/compterenbase2-tape200-c4c19.png]	var state[4] = [0,0,0,0]
	var new_state[4] = [0,0,0,0]
	mic.threshold = 250
	call sound.system(-1)
	call leds.top(0,0,0)
	call leds.bottom.left(0,0,0)
	call leds.bottom.right(0,0,0)
	call leds.circle(0,0,0,0,0,0,0,0)
	sub display_state
	call leds.circle(0,state[1]*32,0,state[3]*32,0,state[2]*32,0,state[0]*32)
	onevent mic
	if state[0] == 0 then
	new_state[0] = 1
	end
	if state[0] == 1 then
	new_state[0] = 0
	end
	call math.copy(state, new_state)
	callsub display_state
	onevent buttons
	if button.center == 1 then
	new_state[0] = 0
	end
	call math.copy(state, new_state)
	callsub display_state

Compter en unaire modulo base 4

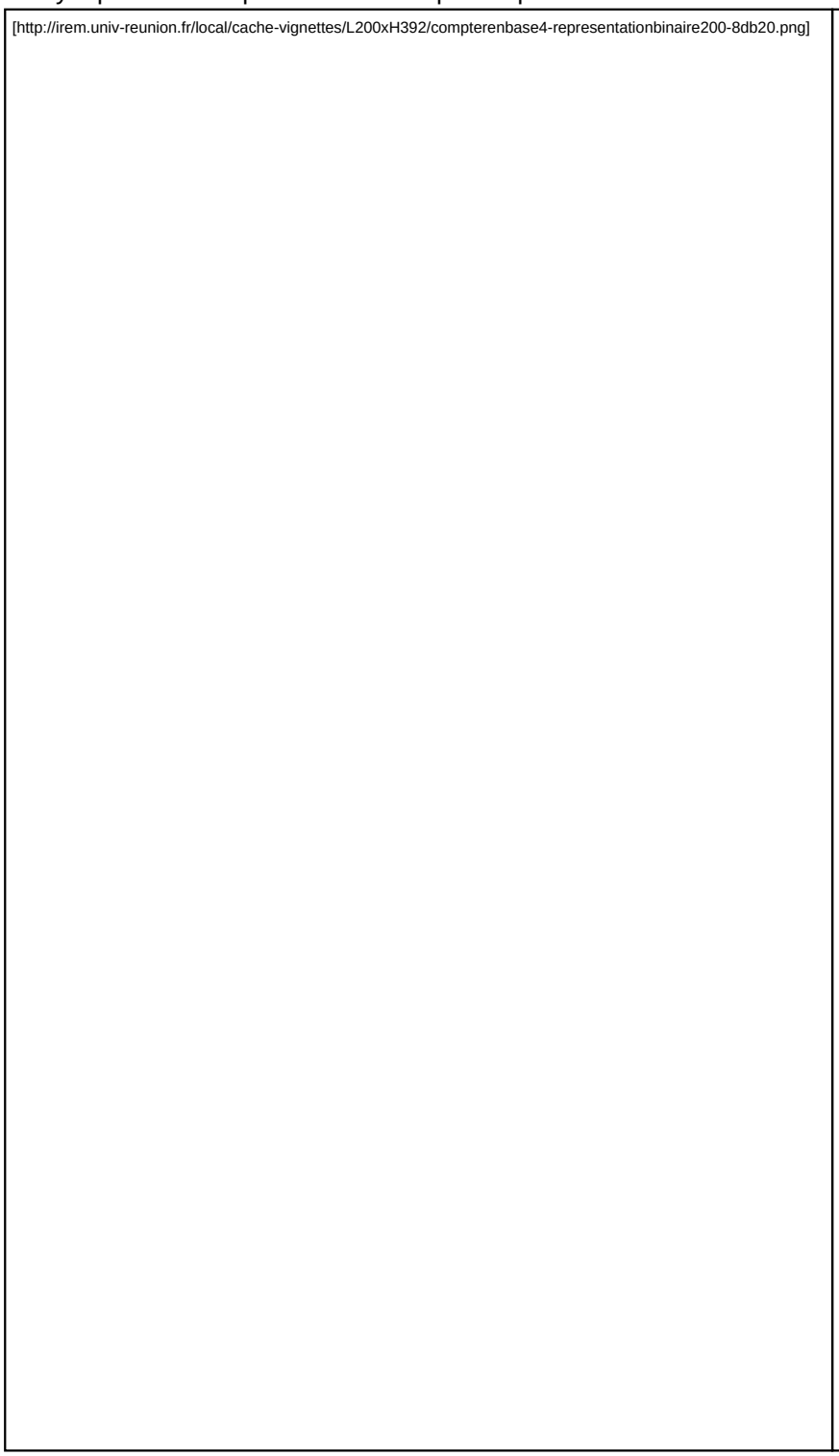
On va afficher le reste de la division euclidienne d'un nombre par 4.
Il faut donc falloir choisir 3 leds car il y a quatre états possibles : 0, 1, 2, 3.



```
var state[4] = [0,0,0,0]
var new_state[4] = [0,0,0,0]
mic.threshold = 250
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call leds.circle(0,0,0,0,0,0,0,0)
sub display_state
 call leds.circle(0,state[1]*32,0,state[3]*32,0,state[2]*32,0,state[0]*32)
onevent buttons
 if button.center == 1 then
 new_state[0] = 0
 new_state[1] = 0
 new_state[2] = 0
 new_state[3] = 0
 end
 call math.copy(state, new_state)
 callsub display_state
onevent mic
 if state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3] == 0 then
 new_state[0] = 0
 new_state[1] = 1
 new_state[2] = 0
 new_state[3] = 0
 end
 if state[0] == 0 and state[1] == 1 and state[2] == 0 and state[3] == 0 then
 new_state[0] = 0
 new_state[1] = 1
 new_state[2] = 0
 new_state[3] = 1
 end
 if state[0] == 0 and state[1] == 1 and state[2] == 0 and state[3] == 1 then
 new_state[0] = 0
 new_state[1] = 1
 new_state[2] = 1
 new_state[3] = 1
 end
 if state[0] == 0 and state[1] == 1 and state[2] == 1 and state[3] == 1 then
 new_state[0] = 0
 new_state[1] = 0
 new_state[2] = 0
 new_state[3] = 0
 end
 call math.copy(state, new_state)
 callsub display_state
```

Compter en base 4 en représentation binaire

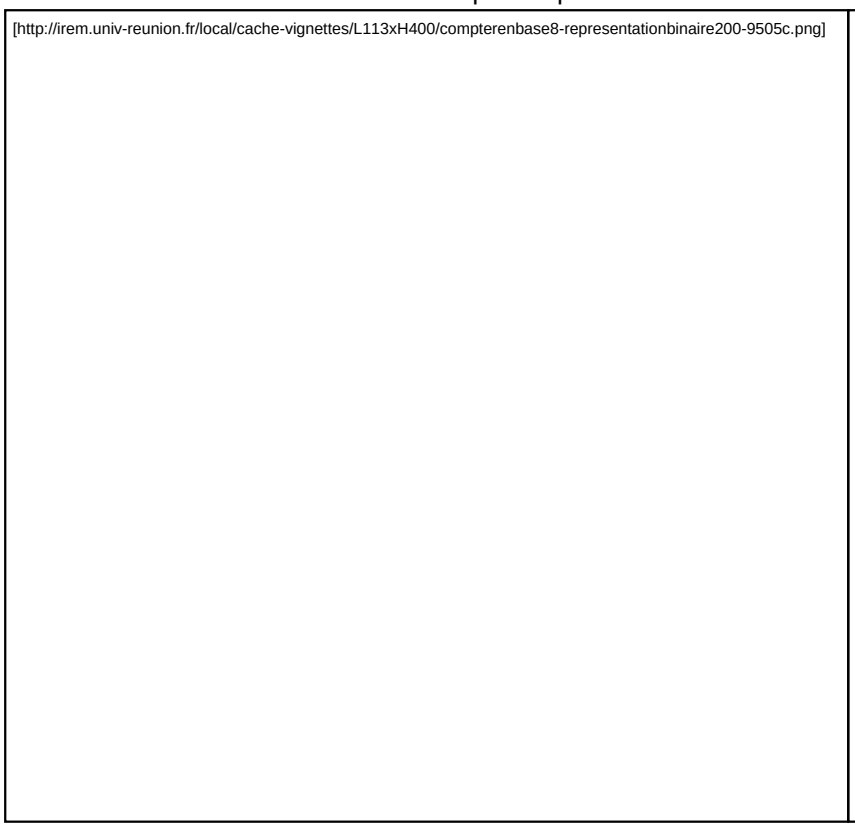
Il n'y a plus besoin que de deux leds pour représenter les nombres de 0 à 3 en base 2.



```
var state[4] = [0,0,0,0]
var new_state[4] = [0,0,0,0]
mic.threshold = 250
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call leds.circle(0,0,0,0,0,0,0,0)
sub display_state
 call leds.circle(0,state[1]*32,0,state[3]*32,0,state[2]*32,0,state[0]*32)
onevent buttons
 if button.center == 1 then
 new_state[0] = 0
 new_state[1] = 0
 end
 call math.copy(state, new_state)
 callsub display_state
onevent mic
 if state[0] == 0 and state[1] == 0 then
 new_state[0] = 0
 new_state[1] = 1
 end
 if state[0] == 0 and state[1] == 1 then
 new_state[0] = 1
 new_state[1] = 0
 end
 if state[0] == 1 and state[1] == 0 then
 new_state[0] = 1
 new_state[1] = 1
 end
 if state[0] == 1 and state[1] == 1 then
 new_state[0] = 0
 new_state[1] = 0
 end
 call math.copy(state, new_state)
 callsub display_state
```

Compter en base 8 en représentation binaire

On aura cette fois besoin de trois leds pour représenter les nombres de 0 à 7 en base 2.



```
var state[4] = [0,0,0,0]
var new_state[4] = [0,0,0,0]
mic.threshold = 250
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call leds.circle(0,0,0,0,0,0,0,0)
sub display_state
 call leds.circle(0,state[1]*32,0,state[3]*32,0,state[2]*32,0,state[0]*32)
onevent buttons
 if button.center == 1 then
 new_state[0] = 0
 new_state[1] = 0
 new_state[2] = 0
 new_state[3] = 0
 end
 call math.copy(state, new_state)
 callsub display_state
onevent mic
 if state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3] == 0 then
 new_state[0] = 0
 new_state[1] = 1
 new_state[2] = 0
 new_state[3] = 0
 end
 if state[0] == 0 and state[1] == 1 and state[2] == 0 and state[3] == 0 then
 new_state[0] = 1
 new_state[1] = 0
 new_state[2] = 0
 new_state[3] = 0
 end
 if state[0] == 1 and state[1] == 0 and state[2] == 0 and state[3] == 0 then
 new_state[0] = 1
 new_state[1] = 1
 new_state[2] = 0
 new_state[3] = 0
 end
 if state[0] == 1 and state[1] == 1 and state[2] == 0 and state[3] == 0 then
 new_state[0] = 0
 new_state[1] = 0
 new_state[2] = 1
 new_state[3] = 0
 end
 if state[0] == 0 and state[1] == 0 and state[2] == 1 and state[3] == 0 then
 new_state[0] = 0
 new_state[1] = 1
 new_state[2] = 1
 new_state[3] = 0
```

```
end
if state[0] == 0 and state[1] == 1 and state[2] == 1 and state[3] == 0 then
new_state[0] = 1
new_state[1] = 0
new_state[2] = 1
new_state[3] = 0
end
if state[0] == 1 and state[1] == 0 and state[2] == 1 and state[3] == 0 then
new_state[0] = 1
new_state[1] = 1
new_state[2] = 1
new_state[3] = 0
end
if state[0] == 1 and state[1] == 1 and state[2] == 1 and state[3] == 0 then
new_state[0] = 0
new_state[1] = 0
new_state[2] = 0
new_state[3] = 0
end
call math.copy(state, new_state)
callsub display_state
```

Compter en base 16 en représentation binaire !

On aura cette fois besoin de quatre leds pour représenter les nombres de 0 à 15 en base 2.

[http://irem.univ-reunion.fr/local/cache-vignettes/L58xH400/compterenbase16-representationbinaire200-71dc8.png]	

```
var state[4] = [0,0,0,0]
var new_state[4] = [0,0,0,0]
mic.threshold = 250
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call leds.circle(0,0,0,0,0,0,0,0)
sub display_state
call leds.circle(0,state[1]*32,0,state[3]*32,0,state[2]*32,0,state[0]*32)
onevent buttons
if button.center == 1 then
new_state[0] = 0
new_state[1] = 0
new_state[2] = 0
new_state[3] = 0
end
call math.copy(state, new_state)
```

```
callsub display_state
onevent mic
if state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3] == 0 then
new_state[0] = 0
new_state[1] = 1
new_state[2] = 0
new_state[3] = 0
end
if state[0] == 0 and state[1] == 1 and state[2] == 0 and state[3] == 0 then
new_state[0] = 1
new_state[1] = 0
new_state[2] = 0
new_state[3] = 0
end
if state[0] == 1 and state[1] == 0 and state[2] == 0 and state[3] == 0 then
new_state[0] = 1
new_state[1] = 1
new_state[2] = 0
new_state[3] = 0
end
if state[0] == 1 and state[1] == 1 and state[2] == 0 and state[3] == 0 then
new_state[0] = 0
new_state[1] = 0
new_state[2] = 1
new_state[3] = 0
end
if state[0] == 0 and state[1] == 0 and state[2] == 1 and state[3] == 0 then
new_state[0] = 0
new_state[1] = 1
new_state[2] = 1
new_state[3] = 0
end
if state[0] == 0 and state[1] == 1 and state[2] == 1 and state[3] == 0 then
new_state[0] = 1
new_state[1] = 0
new_state[2] = 1
new_state[3] = 0
end
if state[0] == 1 and state[1] == 0 and state[2] == 1 and state[3] == 0 then
new_state[0] = 1
new_state[1] = 1
new_state[2] = 1
new_state[3] = 0
end
if state[0] == 1 and state[1] == 1 and state[2] == 1 and state[3] == 0 then
new_state[0] = 0
new_state[1] = 0
new_state[2] = 0
new_state[3] = 1
end
if state[3] == 1 then
new_state[1] = 1
```

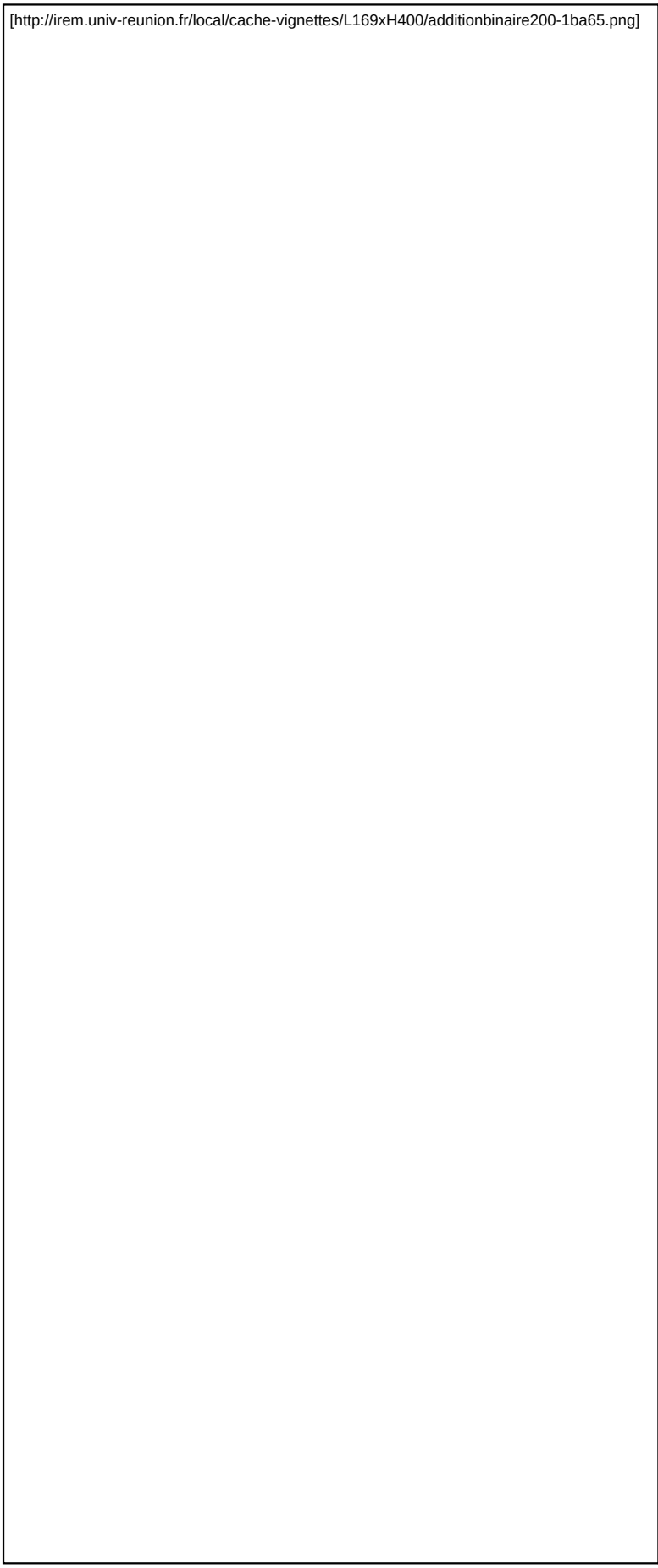
```
new_state[3] = 1
end
if state[1] == 1 and state[3] == 1 then
new_state[0] = 1
new_state[1] = 0
new_state[3] = 1
end
if state[0] == 1 and state[3] == 1 then
new_state[0] = 1
new_state[1] = 1
new_state[3] = 1
end
if state[0] == 1 and state[1] == 1 and state[3] == 1 then
new_state[2] = 1
new_state[3] = 1
end
if state[2] == 1 and state[3] == 1 then
new_state[1] = 1
new_state[2] = 1
new_state[3] = 1
end
if state[1] == 1 and state[2] == 1 and state[3] == 1 then
new_state[0] = 0
new_state[1] = 1
new_state[2] = 1
new_state[3] = 1
end
if state[0] == 0 and state[1] == 1 and state[2] == 1 and state[3] == 1 then
new_state[0] = 1
new_state[1] = 1
new_state[2] = 1
new_state[3] = 1
end
if state[0] == 1 and state[1] == 1 and state[2] == 1 and state[3] == 1 then
new_state[0] = 0
new_state[1] = 0
new_state[2] = 0
new_state[3] = 0
end
call math.copy(state, new_state)
callsub display_state
```

Exercices :

1. Pour compter en unaire modulo base 4, il suffit de trois leds. De combien de leds a-t-on besoin pour compter en base 8 ? Expliquer. Réaliser le programme.
2. La partie Â« Compter en base 16 en représentation binaire Â» ne nécessite que quatre leds pour représenter les nombres de 0 à 15 en base 2. Que peut-on, faire si on utilise huit leds ? Expliquer le principe. (pas besoin du Thymio ici...)

Addition en binaire

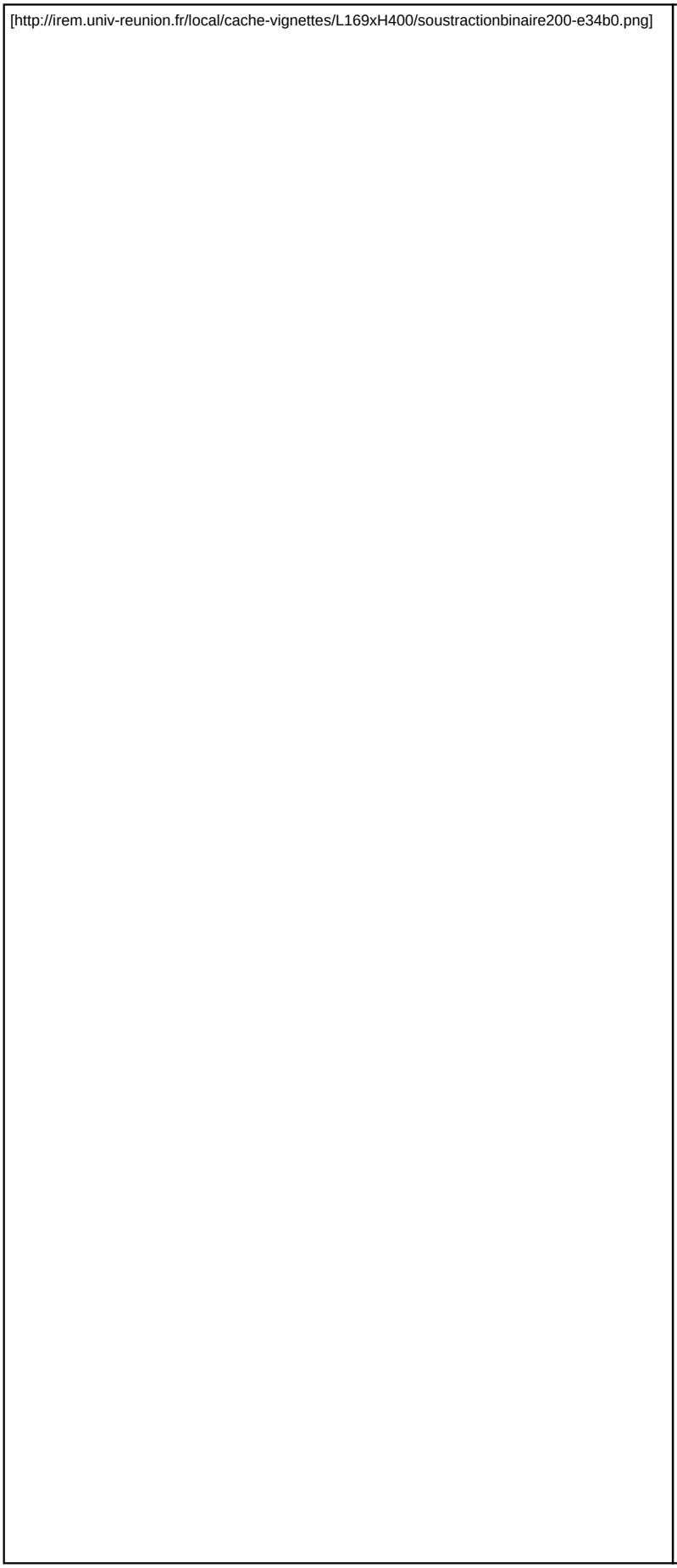
Si on ajoute 1 à un nombre en binaire qui finit par 0, ce 0 devient 1.
Si on ajoute 1 à un nombre en binaire qui finit par 1, ce 1 devient 0, et il y a un effet sur les nombres précédents..
L'arbre ci-dessous montre les transitions 0 ’ 1 et 1 ’ 0 pour l'addition.
<a href="http://irem.univ-reunion.fr/IMG/png/capture_du_2014-11-01_17_36_38.png" type="image/png">[PNG]



```
var state[4] = [0,0,0,0]
var new_state[4] = [0,0,0,0]
mic.threshold = 250
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call leds.circle(0,0,0,0,0,0,0,0)
sub display_state
 call leds.circle(0,state[1]*32,0,state[3]*32,0,state[2]*32,0,state[0]*32)
onevent buttons
 if button.center == 1 then
 new_state[0] = 0
 new_state[1] = 0
 new_state[2] = 0
 new_state[3] = 0
 end
 call math.copy(state, new_state)
 callsub display_state
onevent mic
 if state[1] == 0 then
 new_state[1] = 1
 end
 if state[0] == 0 and state[1] == 1 then
 new_state[0] = 1
 new_state[1] = 0
 end
 if state[0] == 1 and state[1] == 1 and state[2] == 0 then
 new_state[0] = 0
 new_state[1] = 0
 new_state[2] = 1
 end
 if state[0] == 1 and state[1] == 1 and state[2] == 1 and state[3] == 0 then
 new_state[0] = 0
 new_state[1] = 0
 new_state[2] = 0
 new_state[3] = 1
 end
 if state[0] == 1 and state[1] == 1 and state[2] == 1 and state[3] == 1 then
 new_state[0] = 0
 new_state[1] = 0
 new_state[2] = 0
 new_state[3] = 0
 end
 call math.copy(state, new_state)
 callsub display_state
```

Soustraction en binaire

Si on enlève 1 à un nombre en binaire qui finit par 1, ce 1 devient 0.
Si on enlève 1 à un nombre en binaire qui finit par 0, ce 1 devient 0, et il y a un effet sur les nombres précédents.
L'arbre ci-dessous montre les transitions 1 ’ 0 et 0 ’ 1 pour la soustraction.
[http://irem.univ-reunion.fr/local/cache-vignettes/L348xH400/capture_du_2014-11-01_17_37_50-6cf6e.png]



```
var state[4] = [0,0,0,0]
var new_state[4] = [0,0,0,0]
mic.threshold = 250
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call leds.circle(0,0,0,0,0,0,0,0)
sub display_state
 call leds.circle(0,state[1]*32,0,state[3]*32,0,state[2]*32,0,state[0]*32)
onevent buttons
 if button.center == 1 then
 new_state[0] = 1
 new_state[1] = 1
 new_state[2] = 1
 new_state[3] = 1
 end
 call math.copy(state, new_state)
 callsub display_state
onevent mic
 if state[1] == 1 then
 new_state[1] = 0
 end
 if state[0] == 1 and state[1] == 0 then
 new_state[0] = 0
 new_state[1] = 1
 end
 if state[0] == 0 and state[1] == 0 and state[2] == 1 then
 new_state[0] = 1
 new_state[1] = 1
 new_state[2] = 0
 end
 if state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3] == 1 then
 new_state[0] = 1
 new_state[1] = 1
 new_state[2] = 1
 new_state[3] = 0
 end
 if state[0] == 0 and state[1] == 0 and state[2] == 0 and state[3] == 0 then
 new_state[0] = 1
 new_state[1] = 1
 new_state[2] = 1
 new_state[3] = 1
 end
 call math.copy(state, new_state)
 callsub display_state
```

<dl class='spip_document_8841 spip_documents spip_documents_right' style='float:right;'> <a href="http://irem.univ-reunion.fr/IMG/zip/additionbinaireavectimer.aesl.zip" title='Zip - 611 octets' type="application/zip">[Zip - 611 octets]

En pratique, il faut affiner les programmes précédents en ajoutant un timer d'enregistrement, afin qu'il n'y ait pas de bruits parasites, ou d'écho qui perturbent le fonctionnement du programme. (Voir par exemple l'addition binaire avec timer).

Thymio et son interface de programmation, un outil idéal pour faire des Sciences

Les nombreux capteurs du Thymio laissent notre imagination sans limite. Ainsi, mon collègue de physique François Ploteau a eu l'idée de demander aux élèves des étalonnages des capteurs de distance et d'accélération.

1. Étalonnage des capteurs de distance
2. Étalonnage des capteurs d'accélération du robot Thymio
   [http://irem.univ-reunion.fr/local/cache-vignettes/L400xH300/capture_du_2014-11-02_17_36_22-400-32795.png]

Une description technique détaillée concernant les capteurs et actuateurs se trouve ici (page essentielle pour ce qui va suivre).
Pour suivre les variables liées aux capteurs, il suffit de regarder la fenêtre *Variables* de l'interface Aseba de programmation du Thymio, en laissant cochée la variable *Auto*.
[http://irem.univ-reunion.fr/local/cache-vignettes/L200xH328/capture_du_2014-11-02_17_25_05-200-d481a.png]

Ces étalonnages ont donné lieu à un recueil de données que nous avons traitées avec une superbe bibliothèque Python : la bibliothèque Pygal (je dis superbe car elle fait de magnifiques courbes...).
[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH272/mesuresdaccelerationdurobotthymio-pygal-600-26b37.png]

L'interface de programmation du robot permet aussi de voir les graphiques des valeurs enregistrées par les capteurs.

[Capteurs d'accélération]	[Capteur de température]
[Capteur de vitesse du moteur droit]	

Limite du langage Aseba, langage de programmation du robot Thymio ?

La description du langage Aseba est donnée dans sa page de référence :
« *Aseba est un langage de programmation impératif avec un seul type simple (nombres entiers signés sur 16 bits) et des tableaux. Cette simplicité permet aux développeurs sans connaissances préalables d'un système de type de programmer des comportements ; les nombres entiers étant le type de variable le plus naturel. De plus, les nombres entiers sont particulièrement adaptés aux robots à micro-contrôleurs.* »
Jusqu'ici tout va bien.
Aseba nous a aussi donné la possibilité de créer des sous-routines (mot clé **sub**) pour simplifier le code lorsqu'apparaît une même suite d'opérations à plusieurs endroits dans le code. Malheureusement, cette petite phrase donnée dans la documentation d'Aseba : « *Les sous-routines peuvent accéder à toutes les variables.* » laisse entendre que les variables du langage sont globales. C'est probablement la raison pour laquelle il n'y a pas la possibilité de créer des fonctions avec passage d'arguments. Personnellement, je trouve que cela complique l'écriture

du code.

Par contre, évidemment, l'interface de programmation visuelle, fournie en surcouche pour l'écriture des programmes sur le Thymio est un plus inestimable puisqu'il permet à des enfants de 8 ans par exemple de programmer le robot. La prise en main - devant moi - de l'interface VPL et du robot par un enfant de 8 ans n'a duré que quelques minutes. En une heure, il avait déjà réalisé par lui-même un petit comportement simple, avec marche, recul, rotation du robot, changement de couleurs selon l'événement, timer et réaction sur clappement des mains.