

# Manuel technique de Speed Reflex

Par Lucas TÉCHER, Martin TABAKA et Mathis MAILLOT

Mai 2024

## I Introduction

Ce manuel technique décrit en détail le fonctionnement de Speed Reflex, un jeu de rythme utilisant deux cartes microbit. Nous examinerons le code source, les détails techniques de la communication radio, la gestion du son et la prise en compte des actions des joueurs.

## II Émetteur : emetor.py

### II.i Imports et configuration initiale

---

```
1 from microbit import *
2 import random
3 import radio
4 import music
```

---

- `microbit` : Bibliothèque pour contrôler les fonctionnalités du microbit.
- `random` : Pour générer des séquences d'actions aléatoires.
- `radio` : Pour la communication entre les deux microbits.
- `music` : Pour gérer les sons joués par les microbits.

### II.ii Fonctions principales

- `generate_random_sequence(length)`  
Génère une séquence d'actions aléatoires sans répétition consécutive.

---

```
1 def generate_random_sequence(length):
2     instructions = ["up", "down", "left", "right", "0", "1", "2", "A", "B"]
3     sequence = []
4     prev_action = ""
5
6     for _ in range(length):
7         action = random.choice(instructions)
8         while prev_action == action:
9             action = random.choice(instructions)
10
11         sequence.append(action)
12         prev_action = action
13     return sequence
```

---

— `display_action(action)`  
Affiche l'action actuelle sur l'écran du microbit.

---

```
1  def display_action(action):
2      hash_table = {
3          "up": Image.ARROW_S,
4          "down": Image.ARROW_N,
5          "left": Image.ARROW_W,
6          "right": Image.ARROW_E,
7      }
8      if action not in hash_table.keys():
9          display.show(action)
10     else:
11         display.show(hash_table[action])
```

---

— `game_over()`  
Affiche et joue une animation et un son de défaite.

---

```
1  def game_over():
2      music.play(music.POWER_DOWN, wait=False)
3      display.scroll("GAME OVER")
4      ending = []
5      for k in range(6, 0, -1):
6          ending.append(Image.ANGRY.shift_up(k))
7      for k in range(0, 6):
8          ending.append(Image.ANGRY.shift_down(k))
9
10     display.show(ending, delay=100, loop=True)
```

---

— `win_game()`  
Affiche et joue une animation et son de victoire.

---

```
1  def win_game():
2      music.play(music.RINGTONE, wait=False)
3      radio.send("win")
4      display.scroll("WIN")
5      ending = []
6      for k in range(6, 0, -1):
7          ending.append(Image.HAPPY.shift_up(k))
8      for k in range(0, 6):
9          ending.append(Image.HAPPY.shift_down(k))
10
11     display.show(ending, delay=100, loop=True)
```

---

### II.iii Boucle principale

Initialise le jeu et gère la séquence des actions.

---

```
1  startup = False
2  music.play(music.PRELUDE, wait=False, loop=True)
```

```

3
4 while not startup:
5     for clock in Image.ALL_CLOCKS:
6         if not pin_logo.is_touched():
7             display.show(clock)
8             sleep(100)
9         else:
10            startup = True
11
12 radio.send("startup")
13 music.play(music.POWER_UP)
14
15 flash_animation = [Image().invert() * (i/9) for i in range(9, -1, -1)]
16 display.show(flash_animation, delay=100)
17 sequence = generate_random_sequence(15)
18
19 incoming = radio.receive()
20 while not incoming == "ready":
21     incoming = radio.receive()
22
23 music.play(music.NYAN, wait=False, loop=True)
24 for action in sequence:
25     display_action(action)
26     radio.send(action)
27     while True:
28         status_receive = radio.receive()
29         if status_receive == "next":
30             break
31         elif status_receive == "loose":
32             game_over()
33
34 win_game()

```

---

### III Récepteur : receptor.py

#### III.i Imports et configuration initiale

```

1 from microbit import *
2 import radio
3 import music

```

---

#### III.ii Fonctions principales

— game\_over()  
Gère l'animation et le son de défaite.

```

1 def game_over():
2     music.stop()
3     radio.send("loose")

```

---

---

```

4         display.show(Image.SAD)
5         sleep(500)
6         music.play(music.POWER_DOWN, wait=False, pin=None)

```

---

— win\_game()  
Gère l'animation et le son de victoire.

---

```

1     def win_game():
2         music.stop()
3         display.show(Image.HEART)
4         sleep(500)
5         music.play(music.RINGTONE, wait=False, pin=None)

```

---

### III.iii Boucle principale

Gère la réception des actions et l'interaction du joueur.

---

```

1 incoming = radio.receive()
2 while not incoming == "startup":
3     incoming = radio.receive()
4     display.show(Image.ASLEEP)
5
6 music.play(music.POWER_UP, pin=None)
7 flash_animation = [Image().invert() * (i/9) for i in range(9, -1, -1)]
8 display.show(flash_animation, delay=100)
9 radio.send("ready")
10
11 timer_duration = 3 * 1000
12 running = True
13 music.play(music.NYAN, wait=False, loop=True, pin=None)
14 while running:
15     display.clear()
16     incoming = radio.receive()
17
18     if incoming == "win":
19         win_game()
20     elif incoming != None:
21         action_done = False
22         timer_start = running_time()
23         while not action_done:
24             spent_time = running_time() - timer_start
25             left_time = timer_duration - spent_time
26
27             display.show(int(left_time / 1000) + 1, wait=False)
28             if left_time <= 0:
29                 game_over()
30                 running = False
31                 break
32
33             if accelerometer.is_gesture("up") and incoming == "up":
34                 action_done = True

```

```

35         elif accelerometer.is_gesture("down") and incoming == "down":
36             action_done = True
37         elif accelerometer.is_gesture("left") and incoming == "left":
38             action_done = True
39         elif accelerometer.is_gesture("right") and incoming == "right":
40             action_done = True
41         elif button_a.is_pressed() and incoming == "A":
42             action_done = True
43         elif button_b.is_pressed() and incoming == "B":
44             action_done = True
45         elif pin0.is_touched() and incoming == "0":
46             action_done = True
47         elif pin1.is_touched() and incoming == "1":
48             action_done = True
49         elif pin2.is_touched() and incoming == "2":
50             action_done = True
51
52     if action_done:
53         display.show(Image.HAPPY)
54
55     sleep(500)
56     radio.send("next")

```

---

## IV Détails techniques

1. Communication radio :
  - La communication radio entre les microbits utilise le module `radio`.
  - L'émetteur envoie des actions à effectuer et le récepteur envoie des statuts ("next" ou "loose").
2. Gestion du son :
  - Les sons sont gérés par le module `music`.
  - Différents sons sont utilisés pour les événements (démarrage, victoire, défaite).
3. Prise en compte des actions :
  - Le récepteur détecte les actions à l'aide de capteurs intégrés : accéléromètre pour les gestes, boutons A et B, et pins tactiles.
4. Séquence et timing :
  - Une séquence aléatoire d'actions est générée pour chaque partie par la fonction `generate_random_sequence(length)` et grâce au module `random`.
  - Un minuteur est utilisé pour imposer une limite de temps pour chaque action. La fonction `running_time()` est utilisée pour suivre le temps écoulé.

## V Code Source

### V.i Émetteur : `emeter.py`

---

```

1 from microbit import *
2 import random
3 import radio

```

```

4  import music
5
6  def generate_random_sequence(length):
7      instructions = ["up", "down", "left", "right", "0", "1", "2", "A", "B"]
8      sequence = []
9      prev_action = ""
10
11     for _ in range(length):
12         action = random.choice(instructions)
13         while prev_action == action:
14             action = random.choice(instructions)
15
16         sequence.append(action)
17         prev_action = action
18
19     return sequence
20
21 def display_action(action):
22     hash_table = {
23         "up": Image.ARROW_S,
24         "down": Image.ARROW_N,
25         "left": Image.ARROW_W,
26         "right": Image.ARROW_E,
27     }
28
29     if action not in hash_table.keys():
30         display.show(action)
31     else:
32         display.show(hash_table[action])
33
34 def game_over():
35     music.play(music.POWER_DOWN, wait=False)
36     display.scroll("GAME OVER")
37     ending = []
38     for k in range(6, 0, -1):
39         ending.append(Image.ANGRY.shift_up(k))
40     for k in range(0, 6):
41         ending.append(Image.ANGRY.shift_down(k))
42
43     display.show(ending, delay=100, loop=True)
44
45 def win_game():
46     music.play(music.RINGTONE, wait=False)
47     radio.send("win")
48     display.scroll("WIN")
49     ending = []
50     for k in range(6, 0, -1):
51         ending.append(Image.HAPPY.shift_up(k))
52     for k in range(0, 6):
53         ending.append(Image.HAPPY.shift_down(k))
54

```

```

55     display.show(ending, delay=100, loop=True)
56
57     startup = False
58     music.play(music.PRELUDE, wait=False, loop=True)
59
60     while not startup:
61         for clock in Image.ALL_CLOCKS:
62             if not pin_logo.is_touched():
63                 display.show(clock)
64                 sleep(100)
65             else:
66                 startup = True
67
68     radio.send("startup")
69     music.play(music.POWER_UP)
70
71     flash_animation = [Image().invert() * (i/9) for i in range(9, -1, -1)]
72     display.show(flash_animation, delay=100)
73     sequence = generate_random_sequence(15)
74
75     incoming = radio.receive()
76     while not incoming == "ready":
77         incoming = radio.receive()
78
79     music.play(music.NYAN, wait=False, loop=True)
80     for action in sequence:
81         display_action(action)
82         radio.send(action)
83         while True:
84             status_receive = radio.receive()
85             if status_receive == "next":
86                 break
87             elif status_receive == "loose":
88                 game_over()
89
90     win_game()

```

---

## V.ii Récepteur : receptor.py

---

```

1  from microbit import *
2  import radio
3  import music
4
5  def game_over():
6      music.stop()
7      radio.send("loose")
8      display.show(Image.SAD)
9      sleep(500)
10     music.play(music.POWER_DOWN, wait=False, pin=None)
11

```

```

12 def win_game():
13     music.stop()
14     display.show(Image.HEART)
15     sleep(500)
16     music.play(music.RINGTONE, wait=False, pin=None)
17
18 incoming = radio.receive()
19 while not incoming == "startup":
20     incoming = radio.receive()
21     display.show(Image.ASLEEP)
22
23 music.play(music.POWER_UP, pin=None)
24 flash_animation = [Image().invert() * (i/9) for i in range(9, -1, -1)]
25 display.show(flash_animation, delay=100)
26 radio.send("ready")
27
28 timer_duration = 3 * 1000
29 running = True
30 music.play(music.NYAN, wait=False, loop=True, pin=None)
31 while running:
32     display.clear()
33     incoming = radio.receive()
34
35     if incoming == "win":
36         win_game()
37     elif incoming != None:
38         action_done = False
39         timer_start = running_time()
40         while not action_done:
41             spent_time = running_time() - timer_start
42             left_time = timer_duration - spent_time
43
44             display.show(int(left_time / 1000) + 1, wait=False)
45             if left_time <= 0:
46                 game_over()
47                 running = False
48                 break
49
50             if accelerometer.is_gesture("up") and incoming == "up":
51                 action_done = True
52             elif accelerometer.is_gesture("down") and incoming == "down":
53                 action_done = True
54             elif accelerometer.is_gesture("left") and incoming == "left":
55                 action_done = True
56             elif accelerometer.is_gesture("right") and incoming == "right":
57                 action_done = True
58             elif button_a.is_pressed() and incoming == "A":
59                 action_done = True
60             elif button_b.is_pressed() and incoming == "B":
61                 action_done = True
62             elif pin0.is_touched() and incoming == "0":

```



```
63         action_done = True
64     elif pin1.is_touched() and incoming == "1":
65         action_done = True
66     elif pin2.is_touched() and incoming == "2":
67         action_done = True
68
69     if action_done:
70         display.show(Image.HAPPY)
71
72     sleep(500)
73     radio.send("next")
```

---