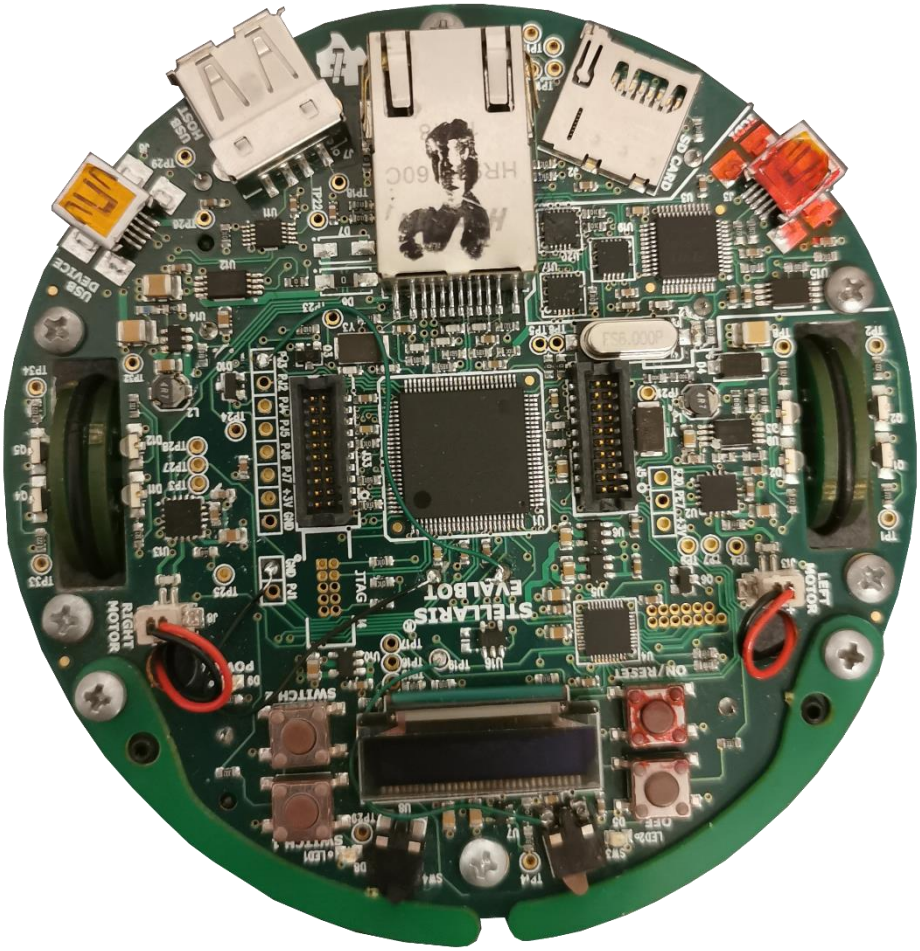
Projet Stellaris EVALBOT

Jeu séquences lumineuses



DESRIAUX Lucie & NOËL Maëva

2020-2021

ESIEE - E3FI

Table des matières

I - Description du projet

1) Concept

Le concept du projet est de faire un jeu à l'aide des leds et des bumpers. Le but de ce jeu est de faire clignoter les deux LEDs de devant en séquence. Le joueur doit alors reproduire la séquence qu'il vient de voir à l'aide des bumpers. Si le joueur gagne, le robot effectue une "danse de la joie" en tournant sur lui-même et en faisant clignoter les LEDs. S'il perd, le robot fait clignoter les deux LEDs trois fois d'affilée simultanément et rapidement.

2) Scénarios

Lorsque l'utilisateur allume l'EVALBOT, les deux LEDs clignotent pour signifier le début du jeu. Ensuite, les deux LEDs au niveau des bumpers vont clignoter alternativement de manière à créer une séquence, plus longue de un clignotement à chaque fois. Au départ, on verra ainsi une LED s'allumer, le joueur appuiera sur le bumper correspondant. S'il a appuyé correctement, la LED s'allumera à nouveau, suivie d'un deuxième allumage ou de l'allumage de l'autre LED. Le jeu continue jusqu'à ce que le joueur se trompe ou qu'il ait reproduit intégralement la séquence sans erreurs.

Scénario 1

- Le joueur se trompe lors d'une des répétitions de la séquence.
- Le joueur perd.
- Les LEDs clignotent simultanément 3 fois pour signifier une erreur.
- Le joueur appuie sur le bouton "on" de l'EVALBOT pour relancer le jeu.

Scénario 2

- Le joueur reproduit entièrement la séquence sans erreur.
- Il gagne.
- L'EVALBOT effectue une "danse de la joie" : il fait clignoter ses leds puis il tourne sur lui-même.
- Le joueur appuie sur le bouton "on" pour relancer le jeu.

II - Fonctionnement

1) Théorie

Notre projet contient 4 fichiers de code. Il contient notamment un fichier led.s qui contient des méthodes agissant sur les LEDs, un fichier moteur.s contenant une fonction liée aux moteurs et un fichier de configuration des moteurs qui a été mis à notre disposition durant l'unité. Le quatrième et dernier fichier est notre main.s, le fichier qui rassemble tout notre code principal et contient le déroulement du jeu.
Nous avons découpé notre code afin de pouvoir importer certaines méthodes définies dans d'autres fichiers, et donc d'améliorer la lisibilité et la réutilisation de code.

Dans notre programme, nous définissons un tableau qui correspond à la séquence. Ce tableau nous sera utile pour savoir quelle LED allumer lors du déroulement du jeu. Nous avons à disposition plusieurs fonctions importées, pour faire clignoter les LED et activer les moteurs. Notre code contient par ailleurs plusieurs étiquettes qui nous permettent de gérer les différents scénarios lors du jeu. Le détail du fonctionnement de ces étiquettes est décrit dans la partie suivante et dans le code au travers des commentaires.

2) Code

Fichier moteurs.s

Comme dit plus tôt, le fichier moteur.s, relié au fichier config_moteur.s pour sa configuration, contient les méthodes relatives aux moteurs. Dans ce fichier, il y a une méthode : DANSE. Cette méthode permet de faire tourner le robot sur lui-même.

Initialisation, importations

La première partie du fichier consiste à l'initialisation des variables globales ainsi qu'à l'import des fonctions nécessaires contenues dans config_moteur.s, et l'exportation des méthodes créées.

```
;; Evalbot (Cortex M3 de Texas Instrument)
;; Projet architecture des ordinateurs
;; Maëva NOËL et Lucie DESRIAUX - E3FI 2I
;; Jeu de séquence

        AREA    |.text|, CODE, READONLY
        ENTRY
        EXPORT  DANSE

        ; IMPORT
        IMPORT  MOTEUR_INIT                 ; initialise les moteurs (configure les pwms + GPIO)

        IMPORT  MOTEUR_DROIT_ON             ; activer le moteur droit
        IMPORT  MOTEUR_DROIT_OFF            ; déactiver le moteur droit
        IMPORT  MOTEUR_DROIT_ARRIERE        ; moteur droit tourne vers l'arrière

        IMPORT  MOTEUR_GAUCHE_ON            ; activer le moteur gauche
        IMPORT  MOTEUR_GAUCHE_OFF           ; déactiver le moteur gauche
        IMPORT  MOTEUR_GAUCHE_AVANT         ; moteur gauche tourne vers l'avant

; Durée pour la danse du robot

DUREE               EQU     0x20FFFFD
```

Figure 1 : Exportation de la méthode contenue dans le fichier, importation des méthodes de config_moteur.s, initialisation des variables globales.

Méthode DANSE

La seconde partie de notre code contient la méthode DANSE. Comme dit précédemment, cette méthode permet au robot de tourner sur lui-même.
L'étiquette WAIT correspond à un timer afin de limiter la rotation du robot.

```
; Méthode pour que le robot effectue un tour sur lui-même
DANSE
        MOV r2, #0
        MOV r3, #500
        ; Configure les moteurs
        BL  MOTEUR_INIT

        ; Activer les deux moteurs droit et gauche
        BL  MOTEUR_DROIT_ON
        BL  MOTEUR_GAUCHE_ON

        ; Rotation à droite de l'Evalbot
        BL  MOTEUR_DROIT_ARRIERE
        BL  MOTEUR_GAUCHE_AVANT
        BL  WAIT

; Boucle d'attente
WAIT    LDR r1,=DUREE
wait1   SUBS r1, #1
        BNE wait1

; Arrêt des moteurs
        BL  MOTEUR_DROIT_OFF
        BL  MOTEUR_GAUCHE_OFF

        BX  LR

        END
```

Figure 2 : Méthode DANSE du fichier moteur.s

Fichier led.s

Le fichier led.s contient plusieurs méthodes permettant de faire clignoter les LEDs comme nous le voulons.

Initialisations, importations et exportations

Comme le fichier précédent, led.s est séparé en plusieurs parties. La première partie consiste donc à l'initialisation des variables globales et l'exportation des méthodes créées.

```
;; Evalbot (Cortex M3 de Texas Instrument)
;; Projet architecture des ordinateurs
;; Maëva NOËL et Lucie DESRIAUX - E3FI 2I
;; Jeu de séquence

; Logique de synchronisation d'horloge en mode run normal

SYSCTL_PERIPH_GPIOF EQU     0x400FE108

; GPIO Port F (APB) base: 0x4002.5000

GPIO_PORTF_BASE     EQU     0x40025000

; GPIO Direction

GPIO_O_DIR          EQU     0x00000400

; Variateur GPIO 2-mA

GPIO_O_DR2R         EQU     0x00000500

; Registre d'activation numérique
; Activation numérique GPIO

GPIO_O_DEN          EQU     0x0000051C

; Port - LED 1 et 2

PORT45              EQU     0x30

; Port - LED 1

PORT4               EQU     0x10

; Port - LED 2

PORT5               EQU     0x20
; Fréquence de clignotement

DUREE                EQU     0x000DFFFF


        AREA    |.text|, CODE, READONLY
        ENTRY

        ; Export
        EXPORT  CLIGNOTE2
        EXPORT  CLIGNOTE3
        EXPORT  CLIGNOTEG
        EXPORT  CLIGNOTED
```

Figure 3 : Exportation des méthodes contenue dans le fichier, initialisation des variables globales

Méthode CLIGNOTE2

Nous avons ensuite toutes nos méthodes. La première méthode, CLIGNOTE2, permet de faire clignoter simultanément nos deux LEDs à l'avant de l'EVALBOT deux fois de suite.

```
54  ; Méthode : faire clignoter deux fois les deux LEDs simultanément
55  CLIGNOTE2
56  ; Activer l'horloge périphérique du port F en définissant le bit 5 (0x20 == 0b100000)
57
58          LDR r3, = SYSCTL_PERIPH_GPIOF
59          MOV r8, #0x00000020
60
61  ; Délai de 3 horloges système
62
63          NOP
64          NOP
65          NOP
66
67  ; Configuration LED
68
69          LDR r3, = GPIO_PORTF_BASE+GPIO_O_DIR
70          LDR r8, = PORT45
71          STR r8, [r3]
72
73          LDR r3, = GPIO_PORTF_BASE+GPIO_O_DEN    ; Activer la fonction numérique
74          LDR r8, = PORT45
75          STR r8, [r3]
76
77          LDR r3, = GPIO_PORTF_BASE+GPIO_O_DR2R   ; Choix de l'intensité de sortie (2mA)
78          LDR r8, = PORT45
79          STR r8, [r3]
80          MOV r10, #0x000                         ; Eteindre LED
81
82  ; Allumer les LEDs (PORT45)
83
84          MOV r11, #PORT45
85          LDR r3, =GPIO_PORTF_BASE + (PORT45<<2)
86
87  ; Fin configuration LED
88
90  ; Boucle : 2 clignotements
91          MOV r4, #0
92
93  loopC2      CMP r4, #2
94              BEQ finC2           ; Boucle terminée
95              ADD r4, #1
96              STR r10, [r3]       ; Eteint LED car r10 = 0x00
97              LDR r9, = DUREE     ; Pour la durée de la boucle d'attente
98
99  wait1C2     SUBS r9, #1
100             BNE wait1C2
101
102             STR r11, [r3]       ; Allume LED
103             LDR r9, = DUREE     ; Pour la durée de la boucle d'attente
104
105 wait2C2     SUBS r9, #1
106             BNE wait2C2
107
108             B loopC2
109 finC2
110             STR r10, [r3]
111             BX LR
```

Figure 4 : Code de la méthode CLIGNOTE2

Méthode CLIGNOTE3

La seconde méthode, CLIGNOTE3, permet de faire clignoter simultanément nos deux LEDs à l'avant de l'EVALBOT trois fois de suite.

```
; Méthode : faire clignoter trois fois les deux LEDs simultanément
CLIGNOTE3
; Activer l'horloge périphérique du port F en définissant le bit 5 (0x20 == 0b100000)

        LDR r3, = SYSCTL_PERIPH_GPIOF
        MOV r8, #0x00000020

; Délai de 3 horloges système

        NOP
        NOP
        NOP

; Configuration LED

        LDR r3, = GPIO_PORTF_BASE+GPIO_O_DIR
        LDR r8, = PORT45
        STR r8, [r3]

        LDR r3, = GPIO_PORTF_BASE+GPIO_O_DEN    ; Activer la fonction numérique
        LDR r8, = PORT45
        STR r8, [r3]

        LDR r3, = GPIO_PORTF_BASE+GPIO_O_DR2R   ; Choix de l'intensité de sortie (2mA)
        LDR r8, = PORT45
        STR r8, [r3]
        MOV r10, #0x000                         ; Eteindre LED

; Allumer les LEDs (PORT45)

        MOV r11, #PORT45
        LDR r3, =GPIO_PORTF_BASE + (PORT45<<2)

; Fin configuration LED
; Boucle : 3 clignotements
        MOV r4, #0

loopC3      CMP r4, #3              ; Boucle terminée
            BEQ finC3
            ADD r4, #1
            STR r10, [r3]           ; Eteint LED car r10 = 0x00
            LDR r9, = DUREE         ; Pour la durée de la boucle d'attente

wait1C3     SUBS r9, #1
            BNE wait1C3

            STR r11, [r3]           ; Allume LED
            LDR r9, = DUREE         ; Pour la durée de la boucle d'attente

wait2C3     SUBS r9, #1
            BNE wait2C3

            B loopC3
finC3
            STR r10, [r3]
            BX LR
```

Figure 5 : Code de la méthode CLIGNOTE3

Méthode CLIGNOTED

La troisième méthode, CLIGNOTED, permet de faire clignoter une seule fois la LED avant droite de l'EVALBOT.

```
; Méthode : faire clignoter LED droite une fois
CLIGNOTED
; Activer l'horloge périphérique du port F en définissant le bit 5 (0x20 == 0b100000)

        LDR r3, = SYSCTL_PERIPH_GPIOF
        MOV r8, #0x00000020

;  Délai de 3 horloges système

        NOP
        NOP
        NOP

; Configuration LED

        LDR r3, = GPIO_PORTF_BASE+GPIO_O_DIR
        LDR r8, = PORT4
        STR r8, [r3]

        LDR r3, = GPIO_PORTF_BASE+GPIO_O_DEN    ; Activer la fonction numérique
        LDR r8, = PORT4
        STR r8, [r3]

        LDR r3, = GPIO_PORTF_BASE+GPIO_O_DR2R   ; Choix de l'intensité de sortie (2mA)
        LDR r8, = PORT4
        STR r8, [r3]
        MOV r10, #0x000                         ; Eteindre LED

; Allumer la LED (PORT4)

        MOV r11, #PORT4
        LDR r3, =GPIO_PORTF_BASE + (PORT4<<2)

; Fin configuration LED
; Clignotement LED

            STR r10, [r3]           ; Eteint LED car r10 = 0x00
            LDR r9, = DUREE         ; Pour la durée de la boucle d'attente

wait1CD     SUBS r9, #1
            BNE wait1CD

            STR r11, [r3]           ; Allume LED
            LDR r9, = DUREE         ; Pour la durée de la boucle d'attente

wait2CD     SUBS r9, #1
            BNE wait2CD

            STR r10, [r3]
            BX LR
```

Figure 6 : Code de la méthode CLIGNOTED

Méthode CLIGNOTEG

La quatrième méthode, CLIGNOTEG, est presque la même que CLIGNOTED, elle permet de faire clignoter une seule fois la LED avant gauche de l'EVALBOT.

```
; Méthode : faire clignoter LED gauche une fois
CLIGNOTEG
; Activer l'horloge périphérique du port F en définissant le bit 5 (0x20 == 0b100000)

        LDR r3, = SYSCTL_PERIPH_GPIOF
        MOV r8, #0x00000020

; Délai de 3 horloges système

        NOP
        NOP
        NOP

; Configuration LED

        LDR r3, = GPIO_PORTF_BASE+GPIO_O_DIR
        LDR r8, = PORT5
        STR r8, [r3]

        LDR r3, = GPIO_PORTF_BASE+GPIO_O_DEN    ; Activer la fonction numérique
        LDR r8, = PORT5
        STR r8, [r3]

        LDR r3, = GPIO_PORTF_BASE+GPIO_O_DR2R   ; Choix de l'intensité de sortie (2mA)
        LDR r8, = PORT5
        STR r8, [r3]
        MOV r10, #0x000                         ; Eteindre LED

; Allumer la LED (PORT5)

        MOV r11, #PORT5
        LDR r3, =GPIO_PORTF_BASE + (PORT5<<2)

; Fin configuration LED
; clignotement LED

            STR r10, [r3]           ; Eteint LED car r10 = 0x00
            LDR r9, = DUREE         ; Pour la durée de la boucle d'attente

wait1CG     SUBS r9, #1
            BNE wait1CG

            STR r11, [r3]           ; Allume LED
            LDR r9, = DUREE         ; Pour la durée de la boucle d'attente

wait2CG     SUBS r9, #1
            BNE wait2CG

            STR r10, [r3]
            BX LR

    END
```

Figure 7 : Code de la méthode CLIGNOTEG

Fichier main.s

Le fichier main.s contient notre code principal. C'est dans ce fichier que nous appelons toutes nos méthodes créées précédemment.

Initialisation, importations et exportations

La première partie du fichier consiste à l'initialisation des variables globales, ainsi qu'à l'exportation du main et l'importation des méthodes utilisées.

```
1  ;; Evalbot (Cortex M3 de Texas Instrument)
2  ;; Projet architecture des ordinateurs
3  ;; Maëva NOËL et Lucie DESRIAUX - E3FI 2I
4  ;; Jeu de séquence
5
6          AREA    |.text|, CODE, READONLY
7
8  ; Logique de synchronisation d'horloge en mode run normal
9  SYSCTL_PERIPH_GPIOF EQU     0x400FE108
10
11 ; GPIO Port F (APB) base: 0x4002.5000
12 GPIO_PORTE_BASE     EQU     0x40024000
13 GPIO_PUR            EQU     0x510
14
15 ; Registre d'activation numérique
16 ; Activation numérique GPIO
17 GPIO_O_DEN          EQU     0x0000051C
18
19 ; PORT E : sélection du BUMPER GAUCHE, LIGNE 0 du Port E
20 PORT0               EQU     0x01
21
22 ; PORT E : sélection du BUMPER DROIT, LIGNE 1 du Port E
23 PORT1               EQU     0x02
24
25 ; Sélection des deux bumpers
26 PORT01               EQU     0x03
27
28 ; Taille du tableau de séquence
29 TAILLE              EQU     0x5
30
31         ENTRY
32         EXPORT  __main
33
34         ; Import des fonctions liées aux LEDs et au moteur
35         IMPORT  CLIGNOTE2
36         IMPORT  CLIGNOTE3
37         IMPORT  CLIGNOTEG
38         IMPORT  CLIGNOTED
39         IMPORT  DANSE
```

Figure 8 : Code du fichier main.s : initialisations, import et export

Méthode main : activation des entrées

Nous passons ensuite au code de la méthode main. Tout d'abord, nous activons les différentes entrées.

```
44  ; Activer GPIOF
45              LDR r6, = SYSCTL_PERIPH_GPIOF
46              MOV r0, #0x00000030
47              STR r0, [r6]
48
49  ; Délai de 3 horloges système
50              NOP
51              NOP
52              NOP
53
54  ; Acriver fonction digitale - Port E
55              LDR r7, =GPIO_PORTE_BASE+GPIO_O_DEN
56              LDR r6, =PORT01
57              STR r6, [r7]
58
59  ; Activer le registre des bumpers, Port E
60              LDR r7, = GPIO_PORTE_BASE+GPIO_PUR
61              LDR r6, = PORT01
62              STR r6, [r7]
```

Figure 9 : Code pour activer les entrées

Méthode main : commencement du jeu et initialisation des variables locales

Ici, nous faisons l'initialisation des variables locales. Nous appelons également la méthode CLIGNOTE2 pour avertir l'utilisateur que le jeu commence.

```
65  ; Clignote deux fois pour lancer le jeu
66              BL CLIGNOTE2
67
68  ; Début du jeu
69  ; Initialisation des variables utilisées
70  init
71              MOV r0,#0
72              LDR r1,=TAILLE
73              LDR r2,=SEQUENCE
74              MOV r4,#0
75              LDRB r5,SEQUENCE
76              B loop
```

Figure 10 : Code initialisation

Méthode main : clignotement des leds selon le tableau contenant la séquence à afficher

Dans cette partie, nous regardons si la séquence a été affichée en entier ou non. Si c'est le cas, alors le joueur a gagné, sinon nous affichons la suite de la séquence. Selon la valeur contenue dans notre tableau, nous faisons clignoter la LED droite ou gauche.

```
78  ; Boucle qui vérifie si on a atteint la fin de la séquence
79  loop
80              CMP r1, r0      ; Comparaison entre l'index actuel et la longueur de la séquence
81              BEQ win         ; Victoire car toute la séquence a été reproduite sans erreur
82              ADD r0, #1      ; Incrémentation de l'index
83              MOV r4, #0
84              B sequence
85
86  ; On compte le nombre de cases à lire dans la séquence
87  sequence    CMP r0, r4      ; Comparaison nombre de led a afficher
88              MOV r12, #0
89              BEQ bumpers     ; Si toute la séquence qui doit s'afficher a clignoté : on vérifie les bumpers
90              ; On choisit si on doit allumer la LED1 ou 2
91              LDRB r5,[r2,r4]
92              ADD r4, #1
93              CMP r5, #0
94              BEQ test1
95              BNE test2
96
97  ; Allumage LED1
98  test1       BL CLIGNOTEG
99              B sequence
100
101 ; Allumage LED2
102 test2       BL CLIGNOTED
103             B sequence
```

Figure 11 : Code clignotement des LEDs

Méthode main : Écoute entrées bumpers

Une fois les LEDs ayant fini de clignoter, nous attendons les entrées des bumpers. Si l'utilisateur a correctement reproduit la séquence correctement, on affiche la suite. Si il lui reste des bumpers à activer, alors on continue de lire les entrées bumpers.

```
105 ; Vérification des bumpers
106 bumpers     CMP r4, r12
107             BEQ loop        ; Si tous les bumpers ont été activés sans erreur : on reprend l'affichage de la séquence
108             MOV r5,r2
109             B lireEntrees   ; Sinon on lit les états des bumpers
110
111 ; Lecture de l'état du BUMPER DROIT
112 lireEntrees LDR r7,= GPIO_PORTE_BASE + (PORT0<<2)
113             LDR r9, [r7]
114             CMP r9, #0x01
115             BNE save1       ; Si le bumper droit a été activé
116
117 ; Lecture de l'état du BUMPER GAUCHE
118             LDR r7, =  GPIO_PORTE_BASE + (PORT1<<2)
119             LDR r9, [r7]
120             CMP r9, #0x02
121             BNE save0       ; Si le bumper gauche a été activé
122
123             ; Si aucun n'a été activé on relit les états
124             B lireEntrees
```

Figure 12 : code de l'écoute des entrées bumpers

Méthode main : Vérification de l'exactitude de l'entrée

Dans cette partie, nous comparons l'entrée bumper avec la valeur n du tableau contenant la séquence. Nous attendons également entre deux entrées afin de laisser le temps à l'utilisateur d'appuyer sur les bumpers pour reproduire la séquence. Si l'entrée correspond à la LED affichée, alors on retourne dans la partie d'écoute des entrées. Sinon l'utilisateur perd et nous allons dans l'étiquette *lose*.

```
137  ; On attend d'avoir l'entrée nécessaire à la vérification
138  savel
139              LDRB r5, [r2,r12]
140              ADD r12, #1
141              MOV r8, #0
142              BL WAIT
143
144  ; Si erreur on envoie vers le scénario de défaite, sinon on reprend la vérification des bumpers
145  compared    CMP r5, #1
146              BEQ bumpers
147              B lose
148
149  ; On attend d'avoir l'entrée nécessaire à la vérification
150  save0
151              LDRB r5, [r2,r12]
152              ADD r12, #1
153              MOV r8, #1
154              B WAIT
155
156  ; Si erreur on envoie vers le scénario de défaite, sinon on reprend la vérification des bumpers
157  compareg    CMP r5, #0
158              BEQ bumpers
159              B lose
```

Figure 13 : Code contenant la vérification des bumpers utilisés par l'utilisateur avec les leds affichées

Méthode main : Fin de la partie

Cette partie gère la fin du jeu. Elle contient donc l'étiquette lose qui permet d'exécuter le scénario dans le cas où l'utilisateur perd. Elle contient aussi l'étiquette win qui permet d'exécuter le scénario dans le cas où l'utilisateur gagne, et l'étiquette de fin du jeu.

```
161  ; Scénario de défaite : on fait clignoter 3 fois les LED et on va à la fin du programme
162  lose        BL CLIGNOTE3
163              B fin
164
165  ; Scénario de victoire : on fait clignoter 2 fois les LED
166  ;et le robot fait un tour sur lui même (danse de la joie), puis on va à la fin du programme
167  win         BL CLIGNOTE2
168              BL DANSE
169              B fin
170  ; Fin du programme
171  fin
```

Figure 14 : Code gérant la fin de partie

Méthode main : déclaration et initialisation du tableau contenant la séquence

Nous déclarons un tableau contenant les LEDs à afficher par l'EVALBOT. Si nous ajoutons des LEDs au tableau, il ne faut pas oublier d'augmenter la variable contenant la taille du tableau.

```
173     ; Initialisation de la séquence des LEDs (0 : LED gauche, 1 : LED droite)
174     AREA constants, DATA, READONLY
175 SEQUENCE DCB  0,1,0,1,1
176
177     END
```

Figure 15 : Code de déclaration / initialisation du tableau

III - Choix techniques

1) Configuration GPIO

Au niveau des GPIO, nous utilisons différents ports dans les fichiers led.s et main.s.
Dans le fichier led, nous configurons les ports : SYSCTL_PERIPH_GPIOF, GPIO_PORTF_BASE, GPIO_DIR, GPIO_DEN, GPIO_DR2R.

SYSCTL_PERIPH_GPIOF permet l'activation de l'horloge système.
GPIO_PORTF_BASE nous permet de définir le port à configurer (ici le port F, le port des LEDs).

Les trois registres suivants doivent être configurés pour les LEDs (broches de sortie) :

- GPIO_DIR nous sert à configurer la broche en sortie (au lieu d'entrée par défaut).
- GPIO_DEN sert à activer la fonction numérique, ce qui effectue la conversion entre données numériques et analogiques.
- GPIO_DR2R fixe l'intensité de courant électrique à véhiculer.

Dans le fichier main.s, nous utilisons à nouveau SYSCTL_PERIPH_GPIOF et GPIO_DEN, et nous configurons également GPIO_PORTE_BASE et GPIO_PUR.

GPIO_PORTE_BASE nous permet de définir le port à configurer (ici le port E, le port des bumpers).

GPIO_DEN et GPIO_PUR sont utilisés afin de configurer les bumpers (broches d'entrée).
GPIO_PUR permet l'activation de la résistance pul_up.
Le rôle de GPIO_DEN ne change pas.

2) Déroulement des programmes

Comme montré dans la partie *II - Fonctionnement*, nous utilisons plusieurs fichiers pour le déroulement du jeu.

Nous commençons par lire le tableau pour allumer la LED correspondante, nous attendons ensuite l'entrée du joueur et enfin nous vérifions qu'elle est correcte avant de recommencer la lecture du tableau et ainsi de suite jusqu'à la fin.

Le programme va également appeler au fur et à mesure que nous en avons besoin les méthodes de LED et du moteur. Dès qu'une erreur est commise le programme se termine. Lors de la victoire, le robot effectue sa danse de la victoire (tour sur lui-même) et le programme se termine.

3) Résultats obtenus

Nous avons rencontré quelques problèmes au cours de notre projet. Tout d'abord, notre première idée n'a pas pu aboutir en raison d'un problème matériel. Nous souhaitions utiliser l'afficheur de l'evalbot mais nous avons appris après avoir commencé que ce n'était pas possible sur les modèles qui nous ont été fournis.
Nous avons rencontré des difficultés au niveau de la danse du robot, nous souhaitions que le robot fasse un tour sur lui-même et s'arrête. Au départ le robot tournait continuellement sans jamais s'arrêter. Nous avons donc implémenté un timer pour le forcer à ne tourner que le temps d'une rotation.
Lors des tests du programme, nous avons aussi remarqué que le joueur n'avait pas le temps de reproduire la séquence que le robot considérait déjà que la séquence était fausse.
Nous avons dû, ici aussi, ajouter un timer pour laisser du temps au joueur à chaque entrée.

Notre programme est aujourd'hui fonctionnel et fait tout ce qui était prévu. Les scénarios de victoire et de défaite fonctionnent tous deux.

Avec plus de temps, nous aurions souhaité ajouter différentes séquences à reproduire, voire une génération aléatoire de séquences. Nous aurions pu aussi ajouter le fait que le jeu recommence après une défaite, pourquoi pas avec un nombre de chances défini.