ESIEE PARIS 04/12/2018

# Architecture

## Projet N°10 : CacheBot

Pierre PERRIER | Sarah CE-OUGNA

# SOMMAIRE

**Présentation du projet _____ 2**

**Fichier RK_Led_Switch_Bumper.s _____ 3**

**Fichier RK_Led_Switch_Bumper.s _____ 7**

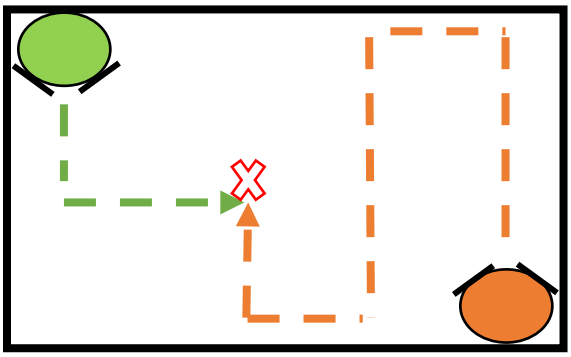
**Fichier Program.s _____ 14**

**Explications et Conclusion _____ 20**

## Description

Il s'agit d'un jeu de cache-cache entre deux robots : un des robots se cache et reste immobile, pendant que l'autre le cherche.

Dès qu'il le trouve, il s'arrête et le jeu peut recommencer : c'est à lui de se cacher et à l'autre de le trouver.



## Scénario

On délimitera une surface de jeu, par exemple un carré de 50cm2.

Il y aura deux « modes » de jeu : celui qui cherche et celui qui se cache. Les boutons switch permettront de définir quel mode de jeu chaque robot utilise.

Celui qui se cache se déplacera sur une certaine distance à l'intérieur de la surface délimitée, s'arrêtera, puis patientera.

Pendant qu'il avancera jusqu'à sa « cachette » seulement une de ses LEDs sera allumée, et lorsque qu'il aura trouvé sa « cachette », ses deux LEDs seront allumées.

Celui qui cherche se déplacera selon un pattern, par exemple : « Tout droit sur 50cm puis droite, droite, de nouveau tout droit sur 50cm puis gauche, gauche », et ainsi de suite. Lorsque ses bumpers entreront en contact avec le robot « caché », la partie sera terminée et il s'arrêtera.

Pendant sa recherche, ses LEDs clignoteront alternativement à un rythme lent, puis à un rythme rapide lorsqu'il aura gagné et donc trouvé l'autre robot.

Une fois cette partie terminée, on replacera manuellement les robots à leur position initiale.

La partie pourra alors recommencer et on déterminera de nouveau grâce aux boutons quel robot se cache et quel robot cherche.

## Code

Nous avons organisé notre programme en 3 fichiers : le fichier de configuration des moteurs fourni dans le cours (RK_Moteur.s), un fichier de configuration des LEDs, Switchs et Bumpers (RK_Led_Switch_Bumper.s), et un fichier contenant le programme main (Program.s).

# RK_Led_Switch_Bumper.s

```
; PERRIER Pierre, CE-OUGNA Sarah - ESIEE Paris
; 12/2018 - Evalbot (Cortex M3 de Texas Instrument)
; Projet - CacheBot (Jeu de cache-cache entre deux robots)

; This register controls the clock gating logic in normal Run mode
SYSCTL_PERIPH_GPIO EQU      0x400FE108  ; SYSCTL_RCGC2_R (p291 datasheet
de lm3s9b92.pdf)

; The GPIODATA register is the data register
GPIO_PORTF_BASE        EQU        0x40025000  ; GPIO Port F (APB) base:
0x4002.5000 (p416 datasheet de lm3s9B92.pdf)

; The GPIODATA register is the data register
GPIO_PORTD_BASE        EQU        0x40007000       ; GPIO Port D (APB)
base: 0x4000.7000 (p416 datasheet de lm3s9B92.pdf)

; The GPIODATA register is the data register
GPIO_PORTE_BASE        EQU        0x40024000  ; GPIO Port E (APB) base:
0x4002.5000 (p416 datasheet de lm3s9B92.pdf)

; configure the corresponding pin to be an output
; all GPIO pins are inputs by default
GPIO_O_DIR             EQU   0x00000400  ; GPIO Direction (p417 datasheet
de lm3s9B92.pdf)

; The GPIODR2R register is the 2-mA drive control register
; By default, all GPIO pins have 2-mA drive.
GPIO_O_DR2R            EQU   0x00000500  ; GPIO 2-mA Drive Select (p428
datasheet de lm3s9B92.pdf)

; Digital enable register
; To use the pin as a digital input or output, the corresponding GPIODEN
bit must be set.
GPIO_O_DEN             EQU   0x0000051C  ; GPIO Digital Enable (p437
datasheet de lm3s9B92.pdf)

; Pul_up
GPIO_I_PUR             EQU   0x00000510  ; GPIO Digital Enable (p437
datasheet de lm3s9B92.pdf)

; Broches select
BROCHE4_5              EQU        0x30       ; led1 & led2 sur broche 4
et 5
BROCHE4                EQU        0x10       ; led1
BROCHE5                EQU        0x20       ; led2

BROCHE6_7              EQU   0xC0       ; boutons poussoirs 1&2
BROCHE6                      EQU   0x40       ; bouton poussoirs 1
BROCHE7                      EQU   0x80       ; bouton poussoirs 2
```

```
BROCHE0_1               EQU   0x03        ; bumpers 1&2
BROCHE0                       EQU   0x01        ; bumper 1
BROCHE1                       EQU   0x02        ; bumper 2

            AREA    |.text|, CODE, READONLY
            ENTRY

            ; Export des fonctions contenues dans le fichier
            EXPORT LED_SWITCH_INIT
            EXPORT ALLUME_DROITE
            EXPORT ALLUME_GAUCHE
            EXPORT ETEINT_DROITE
            EXPORT ETEINT_GAUCHE

; Fonction d'initialisation (configuration des LED, Switchs et Bumpers)
LED_SWITCH_INIT

            ; ;; Enable the Port E, F & D peripheral clock           (p291
datasheet de lm3s9B96.pdf)
            ; ;;
            ldr r8, = SYSCTL_PERIPH_GPIO                  ;; RCGC2
         mov r0, #0x00000038                              ;; Enable clock
sur GPIO D et F où sont branchés les leds (0x28 == 0b111000)
            ; ;;

(GPIO::FEDCBA)
        str r0, [r8]

            ; ;; "There must be a delay of 3 system clocks before any GPIO
reg. access  (p413 datasheet de lm3s9B92.pdf)
            nop                                                          ;;
tres tres important....
            nop
            nop                                                          ;;
pas necessaire en simu ou en debbug step by step...

            ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^CONFIGURATION LED

        ldr r8, = GPIO_PORTF_BASE+GPIO_O_DIR    ;; 1 Pin du portF en
sortie (broche 4 : 00010000)
        ldr r0, = BROCHE4_5
        str r0, [r8]

            ldr r8, = GPIO_PORTF_BASE+GPIO_O_DEN    ;; Enable Digital
Function
        ldr r0, = BROCHE4_5
        str r0, [r8]

            ldr r8, = GPIO_PORTF_BASE+GPIO_O_DR2R   ;; Choix de
l'intensité de sortie (2mA)
        ldr r0, = BROCHE4_5
        str r0, [r8]
```

```
           mov r2, #0x000                                ;; pour eteindre
LED

           ; allumer la led broche 4 (BROCHE4_5)
           mov r3, #BROCHE4_5          ;; Allume LED1&2 portF broche 4&5
: 00110000

           ldr r8, = GPIO_PORTF_BASE + (BROCHE4<<2)  ; @data Register =
@base + (mask<<2) ==> LED 1
           ldr r9, = GPIO_PORTF_BASE + (BROCHE5<<2)  ; @data Register =
@base + (mask<<2) ==> LED 2

           ;vvvvvvvvvvvvvvvvvvvvvvvvFin configuration LED

           ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^CONFIGURATION SWITCH

           ldr r7, = GPIO_PORTD_BASE+GPIO_I_PUR    ;; Pul_up
        ldr r0, = BROCHE6_7
        str r0, [r7]

           ldr r7, = GPIO_PORTD_BASE+GPIO_O_DEN    ;; Enable Digital
Function
        ldr r0, = BROCHE6_7
        str r0, [r7]

           ldr r7, = GPIO_PORTD_BASE + (BROCHE6<<2)  ;; @data Register =
@base + (mask<<2) ==> Switcher 1
           ldr r10, = GPIO_PORTD_BASE + (BROCHE7<<2)  ;; @data Register =
@base + (mask<<2) ==> Switcher 2
           ;vvvvvvvvvvvvvvvvvvvvvvvvFin configuration SWITCH

           ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^CONFIGURATION BUMPER

           ldr r5, = GPIO_PORTE_BASE+GPIO_I_PUR    ;; Pul_up
        ldr r0, = BROCHE0_1
        str r0, [r5]

           ldr r5, = GPIO_PORTE_BASE+GPIO_O_DEN    ;; Enable Digital
Function
        ldr r0, = BROCHE0_1
        str r0, [r5]

           ldr r5, = GPIO_PORTE_BASE + (BROCHE0<<2)  ;; @data Register =
@base + (mask<<2) ==> BUMPER 1
           ldr r4, = GPIO_PORTE_BASE + (BROCHE1<<2)  ;; @data Register =
@base + (mask<<2) ==> BUMPER 2
           ;vvvvvvvvvvvvvvvvvvvvvvvvFin configuration BUMPER

           BX    LR


; Fonction pour allumer la LED Droite
ALLUME_DROITE
```

```
            str r3, [r8]                                        ; Allume
LED1 portF broche 4 : 00010000 (contenu de r3)
            BX    LR


; Fonction pour éteindre la LED Droite
ETEINT_DROITE
            str r2, [r8]
            BX LR


; Fonction pour allumer la LED Gauche
ALLUME_GAUCHE

            str r3, [r9]                                        ; Allume
LED2 portF broche 5 : 00100000 (contenu de r3)
            BX    LR


; Fonction pour éteindre la LED Gauche
ETEINT_GAUCHE
            str r2, [r9]
            BX LR

            END
```

# RK_Moteur.s

```
; PERRIER Pierre, CE-OUGNA Sarah - ESIEE Paris
; 12/2018 - Evalbot (Cortex M3 de Texas Instrument)
; Projet - CacheBot (Jeu de cache-cache entre deux robots)

;Les pages se réfèrent au datasheet lm3s9b92.pdf

;Cablage :
;pin 10/PD0/PWM0 => input PWM du pont en H DRV8801RT
;pin 11/PD1/PWM1 => input Phase_R  du pont en H DRV8801RT
;pin 12/PD2       => input SlowDecay commune aux 2 ponts en H
;pin 98/PD5       => input Enable 12v du conv DC/DC
;pin 86/PH0/PWM2 => input PWM du 2nd pont en H
;pin 85/PH1/PWM3 => input Phase du 2nd pont en H

;; Hexa corresponding values to pin numbers
GPIO_0          EQU         0x1
GPIO_1          EQU         0x2
GPIO_2          EQU         0x4
GPIO_5          EQU         0x20

;; pour enable clock    0x400FE000
SYSCTL_RCGC0    EQU         0x400FE100       ;SYSCTL_RCGC0: offset 0x100
(p271 datasheet de lm3s9b92.pdf)
SYSCTL_RCGC2    EQU         0x400FE108       ;SYSCTL_RCGC2: offset 0x108
(p291 datasheet de lm3s9b92.pdf)

;; General-Purpose Input/Outputs (GPIO) configuration
PORTD_BASE      EQU         0x40007000
GPIODATA_D      EQU         PORTD_BASE
GPIODIR_D       EQU         PORTD_BASE+0x00000400
GPIODR2R_D      EQU         PORTD_BASE+0x00000500
GPIODEN_D       EQU         PORTD_BASE+0x0000051C
GPIOPCTL_D      EQU         PORTD_BASE+0x0000052C ; GPIO Port Control
(GPIOPCTL), offset 0x52C; p444
GPIOAFSEL_D     EQU         PORTD_BASE+0x00000420 ; GPIO Alternate
Function Select (GPIOAFSEL), offset 0x420; p426

PORTH_BASE      EQU         0x40027000
GPIODATA_H      EQU         PORTH_BASE
GPIODIR_H       EQU         PORTH_BASE+0x00000400
GPIODR2R_H      EQU         PORTH_BASE+0x00000500
GPIODEN_H       EQU         PORTH_BASE+0x0000051C
GPIOPCTL_H      EQU         PORTH_BASE+0x0000052C ; GPIO Port Control
(GPIOPCTL), offset 0x52C; p444
GPIOAFSEL_H     EQU         PORTH_BASE+0x00000420 ; GPIO Alternate
Function Select (GPIOAFSEL), offset 0x420; p426

;; Pulse Width Modulator (PWM) configuration
```

```
PWM_BASE          EQU         0x040028000         ;BASE des Block PWM
p.1138
PWMENABLE         EQU         PWM_BASE+0x008      ; p1145

;Block PWM0 pour sorties PWM0 et PWM1 (moteur 1)
PWM0CTL                 EQU         PWM_BASE+0x040 ;p1167
PWM0LOAD          EQU         PWM_BASE+0x050
PWM0CMPA          EQU         PWM_BASE+0x058
PWM0CMPB          EQU         PWM_BASE+0x05C
PWM0GENA          EQU         PWM_BASE+0x060
PWM0GENB          EQU         PWM_BASE+0x064

;Block PWM1 pour sorties PWM1 et PWM2 (moteur 2)
PWM1CTL                 EQU         PWM_BASE+0x080
PWM1LOAD          EQU         PWM_BASE+0x090
PWM1CMPA          EQU         PWM_BASE+0x098
PWM1CMPB          EQU         PWM_BASE+0x09C
PWM1GENA          EQU         PWM_BASE+0x0A0
PWM1GENB          EQU         PWM_BASE+0x0A4


VITESSE                 EQU         0x142 ; Valeures plus petites => Vitesse
plus rapide exemple 0x192
                                                ; Valeures plus grandes =>
Vitesse moins rapide exemple 0x1B2


            AREA    |.text|, CODE, READONLY
            ENTRY

            ;; The EXPORT command specifies that a symbol can be accessed
by other shared objects or executables.
            EXPORT      MOTEUR_INIT
            EXPORT      MOTEUR_DROIT_ON
            EXPORT  MOTEUR_DROIT_OFF
            EXPORT  MOTEUR_DROIT_AVANT
            EXPORT  MOTEUR_DROIT_ARRIERE
            EXPORT  MOTEUR_DROIT_INVERSE
            EXPORT      MOTEUR_GAUCHE_ON
            EXPORT  MOTEUR_GAUCHE_OFF
            EXPORT  MOTEUR_GAUCHE_AVANT
            EXPORT  MOTEUR_GAUCHE_ARRIERE
            EXPORT  MOTEUR_GAUCHE_INVERSE


MOTEUR_INIT
            ldr r6, = SYSCTL_RCGC0
            ldr   r0, [R6]
          ORR r0, r0, #0x00100000  ;;bit 20 = PWM recoit clock: ON (p271)
          str r0, [r6]

      ;ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_1);PWM clock is processor clock
/1
      ;Je ne fais rien car par defaut = OK!!
```

```
      ;*(int *) (0x400FE060)= *(int *)(0x400FE060)...;

      ;RCGC2 :  Enable port D GPIO(p291 ) car Moteur Droit sur port D
            ldr r6, = SYSCTL_RCGC2
            ldr   r0, [R6]
        ORR r0, r0, #0x08  ;; Enable port D GPIO
        str r0, [r6]

      ;MOT2 : RCGC2 :  Enable port H GPIO  (2eme moteurs)
            ldr r6, = SYSCTL_RCGC2
            ldr   r0, [R6]
        ORR r0, r0, #0x80  ;; Enable port H GPIO
        str r0, [r6]

            nop
            nop
            nop

      ;;Pin muxing pour PWM, port D, reg. GPIOPCTL(p444), 4bits de
PCM0=0001<=>PWM (voir p1261)
      ;;il faut mettre 1 pour avoir PD0=PWM0 et PD1=PWM1
            ldr r6, = GPIOPCTL_D
            ;ldr  r0, [R6]     ;;   *(int *)(0x40007000+0x0000052C)=1;
        ;ORR     r0, r0, #0x01 ;; Port D, pin 1 = PWM
            mov   r0, #0x01
        str r0, [r6]

      ;;MOT2 : Pin muxing pour PWM, port H, reg. GPIOPCTL(p444), 4bits de
PCM0=0001<=>PWM (voir p1261)
      ;;il faut mettre mux = 2 pour avoir PH0=PWM2 et PH1=PWM3
            ldr r6, = GPIOPCTL_H
            mov   r0, #0x02
        str r0, [r6]

      ;;Alternate Function Select (p 426), PD0 utilise alernate fonction
(PWM au dessus)
      ;;donc PD0 = 1
            ldr r6, = GPIOAFSEL_D
            ldr   r0, [R6]      ;*(int *)(0x40007000+0x00000420)= *(int
*)(0x40007000+0x00000420) | 0x00000001;
        ORR r0, r0, #0x01 ;
        str r0, [r6]

      ;;MOT2 : Alternate Function Select (p 426), PH0 utilise PWM donc
Alternate funct
      ;;donc PH0 = 1
            ldr r6, = GPIOAFSEL_H
            ldr   r0, [R6]      ;*(int *)(0x40007000+0x00000420)= *(int
*)(0x40007000+0x00000420) | 0x00000001;
        ORR r0, r0, #0x01 ;
        str r0, [r6]

      ;;-----------PWM0 pour moteur 1 connecté à PD0
      ;;PWM0 produit PWM0 et PWM1 output
```

```
      ;;Config Modes PWM0 + mode GenA + mode GenB
            ldr r6, = PWM0CTL
            mov   r0, #2           ;Mode up-down-up-down, pas synchro
        str r0, [r6]

            ldr r6, =PWM0GENA ;en decomptage, qd comparateurA = compteur
=> sortie pwmA=0
                                    ;en comptage croissant, qd comparateurA
= compteur => sortie pwmA=1
            mov   r0,   #0x0B0      ;0B0=10110000 => ACTCMPBD=00 (B
down:rien), ACTCMPBU=00(B up rien)
            str r0, [r6]     ;ACTCMPAD=10 (A down:pwmA low), ACTCMPAU=11
(A up:pwmA high) , ACTLOAD=00,ACTZERO=00

            ldr r6, =PWM0GENB;en comptage croissant, qd comparateurB =
compteur => sortie pwmA=1
            mov   r0,   #0x0B00     ;en decomptage, qd comparateurB =
compteur => sortie pwmB=0
            str r0, [r6]
      ;Config Compteur, comparateur A et comparateur B
      ;;#define PWM_PERIOD (ROM_SysCtlClockGet() / 16000),
      ;;en mesure : SysCtlClockGet=0F42400h, /16=0x3E8,
      ;;on divise par 2 car moteur 6v sur alim 12v
            ldr   r6, =PWM0LOAD ;PWM0LOAD=periode/2 =0x1F4
            mov r0,     #0x1F4
            str   r0,[r6]

            ldr   r6, =PWM0CMPA ;Valeur rapport cyclique : pour 10% => 1C2h
si clock = 0F42400
            mov   r0, #VITESSE
            str   r0, [r6]

            ldr   r6, =PWM0CMPB ;PWM0CMPB recoit meme valeur. (rapport
cyclique depend de CMPA)
            mov   r0,   #0x1F4
            str   r0,   [r6]

      ;Control PWM : active PWM Generator 0 (p1167): Enable+up/down +
Enable counter debug mod
            ldr   r6, =PWM0CTL
            ldr   r0, [r6]
            ORR   r0,   r0,   #0x07
            str   r0,   [r6]

      ;;-----------PWM2 pour moteur 2 connecté à PH0
      ;;PWM1block produit PWM2 et PWM3 output
            ;;Config Modes PWM2 + mode GenA + mode GenB
            ldr r6, = PWM1CTL
            mov   r0, #2           ;Mode up-down-up-down, pas synchro
        str r0, [r6]   ;*(int *)(0x40028000+0x040)=2;

            ldr r6, =PWM1GENA ;en decomptage, qd comparateurA = compteur
=> sortie pwmA=0
```

```
                                    ;en comptage croissant, qd comparateurA
= compteur => sortie pwmA=1
            mov   r0,   #0x0B0      ;0B0=10110000 => ACTCMPBD=00 (B
down:rien), ACTCMPBU=00(B up rien)
            str r0, [r6]      ;ACTCMPAD=10 (A down:pwmA low), ACTCMPAU=11
(A up:pwmA high) , ACTLOAD=00,ACTZERO=00

            ;*(int *)(0x40028000+0x060)=0x0B0; //
            ldr r6, =PWM1GENB;*(int *)(0x40028000+0x064)=0x0B00;
            mov   r0,   #0x0B00     ;en decomptage, qd comparateurB =
compteur => sortie pwmB=0
            str r0, [r6]      ;en comptage croissant, qd comparateurB =
compteur => sortie pwmA=1
      ;Config Compteur, comparateur A et comparateur B
      ;;#define PWM_PERIOD (ROM_SysCtlClockGet() / 16000),
      ;;en mesure : SysCtlClockGet=0F42400h, /16=0x3E8,
      ;;on divise par 2 car moteur 6v sur alim 12v
            ;*(int *)(0x40028000+0x050)=0x1F4; //PWM0LOAD=periode/2 =0x1F4
            ldr   r6, =PWM1LOAD
            mov r0,     #0x1F4
            str   r0,[r6]

            ldr   r6, =PWM1CMPA ;Valeur rapport cyclique : pour 10% => 1C2h
si clock = 0F42400
            mov   r0,   #VITESSE
            str   r0, [r6]  ;*(int *)(0x40028000+0x058)=0x01C2;

            ldr   r6, =PWM1CMPB ;PWM0CMPB recoit meme valeur. (CMPA depend
du rapport cyclique)
            mov   r0,   #0x1F4      ; *(int *)(0x40028000+0x05C)=0x1F4;
            str   r0,   [r6]

      ;Control PWM : active PWM Generator 0 (p1167): Enable+up/down +
Enable counter debug mod
            ldr   r6, =PWM1CTL
            ldr   r0, [r6]    ;*(int *) (0x40028000+0x40)= *(int
*)(0x40028000+0x40) | 0x07;
            ORR   r0,   r0,   #0x07
            str   r0,   [r6]

      ;;-----Fin config des PWMs

      ;PORT D OUTPUT pin0 (pwm)=pin1(direction)=pin2(slow decay)=pin5(12v
enable)
            ldr   r6, =GPIODIR_D
            ldr   r0, [r6]
            ORR   r0,   #(GPIO_0+GPIO_1+GPIO_2+GPIO_5)
            str   r0,[r6]
      ;Port D, 2mA les meme
            ldr   r6, =GPIODR2R_D ;
            ldr   r0, [r6]
            ORR   r0,   #(GPIO_0+GPIO_1+GPIO_2+GPIO_5)
            str   r0,[r6]
      ;Port D, Digital Enable
```

```
            ldr   r6, =GPIODEN_D ;
            ldr   r0, [r6]
            ORR   r0,   #(GPIO_0+GPIO_1+GPIO_2+GPIO_5)
            str   r0,[r6]
      ;Port D : mise à 1 de slow Decay et 12V et mise à 0 pour dir et pwm
            ldr   r6, =(GPIODATA_D+((GPIO_0+GPIO_1+GPIO_2+GPIO_5)<<2))
            mov   r0, #(GPIO_2+GPIO_5) ; #0x24
            str   r0,[r6]

      ;MOT2, PH1 pour sens moteur ouput
            ldr   r6, =GPIODIR_H
            mov   r0,   #0x03 ;
            str   r0,[r6]
      ;Port H, 2mA les meme
            ldr   r6, =GPIODR2R_H
            mov r0, #0x03
            str   r0,[r6]
      ;Port H, Digital Enable
            ldr   r6, =GPIODEN_H
            mov r0, #0x03
            str   r0,[r6]
      ;Port H : mise à 1 pour dir
            ldr   r6, =(GPIODATA_H +(GPIO_1<<2))
            mov   r0, #0x02
            str   r0,[r6]

            BX    LR    ; FIN du sous programme d'init.

;Enable PWM0 (bit 0) et PWM2 (bit 2) p1145
;Attention ici c'est les sorties PWM0 et PWM2
;qu'on controle, pas les blocks PWM0 et PWM1!!!
MOTEUR_DROIT_ON
            ;Enable sortie PWM0 (bit 0), p1145
            ldr   r6,   =PWMENABLE
            ldr r0, [r6]
            orr r0,     #0x01 ;bit 0 à 1
            str   r0,   [r6]
            BX    LR

MOTEUR_DROIT_OFF
            ldr   r6,   =PWMENABLE
            ldr r0,     [r6]
            and   r0,   #0x0E ;bit 0 à 0
            str   r0,   [r6]
            BX    LR

MOTEUR_GAUCHE_ON
            ldr   r6,   =PWMENABLE
            ldr   r0, [r6]
            orr   r0,   #0x04 ;bit 2 à 1
            str   r0,   [r6]
            BX    LR

MOTEUR_GAUCHE_OFF
```

```
            ldr   r6,   =PWMENABLE
            ldr   r0,   [r6]
            and   r0,   #0x0B ;bit 2 à 0
            str   r0,   [r6]
            BX    LR

MOTEUR_DROIT_ARRIERE
            ;Inverse Direction (GPIO_D1)
            ldr   r6, =(GPIODATA_D+(GPIO_1<<2))
            mov   r0, #0
            str   r0,[r6]
            BX    LR

MOTEUR_DROIT_AVANT
            ;Inverse Direction (GPIO_D1)
            ldr   r6, =(GPIODATA_D+(GPIO_1<<2))
            mov   r0, #2
            str   r0,[r6]
            BX    LR

MOTEUR_GAUCHE_ARRIERE
            ;Inverse Direction (GPIO_D1)
            ldr   r6, =(GPIODATA_H+(GPIO_1<<2))
            mov   r0, #2 ; contraire du moteur Droit
            str   r0,[r6]
            BX    LR

MOTEUR_GAUCHE_AVANT
            ;Inverse Direction (GPIO_D1)
            ldr   r6, =(GPIODATA_H+(GPIO_1<<2))
            mov   r0, #0
            str   r0,[r6]
            BX    LR

MOTEUR_DROIT_INVERSE
            ;Inverse Direction (GPIO_D1)
            ldr   r6, =(GPIODATA_D+(GPIO_1<<2))
            ldr   r1, [r6]
            EOR   r0, r1, #GPIO_1
            str   r0,[r6]
            BX    LR

MOTEUR_GAUCHE_INVERSE
            ;Inverse Direction (GPIO_D1)
            ldr   r6, =(GPIODATA_H+(GPIO_1<<2))
            ldr   r1, [r6]
            EOR   r0, r1, #GPIO_1
            str   r0,[r6]
            BX    LR

            END
```

# Program.s

```
; PERRIER Pierre, CE-OUGNA Sarah - ESIEE Paris
; 12/2018 - Evalbot (Cortex M3 de Texas Instrument)
; Projet - CacheBot (Jeu de cache-cache entre deux robots)



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

            ; Import des fonctions du fichier RK_Led_Switch_Bumper.s
            IMPORT LED_SWITCH_INIT                          ; Initialise les
LED, Switchs et Bumpers
            IMPORT ALLUME_DROITE                    ; Allume la LED Droite
            IMPORT ALLUME_GAUCHE                    ; Allume la LED Gauche
            IMPORT ETEINT_DROITE                    ; Eteint la LED Droite
            IMPORT ETEINT_GAUCHE                    ; Eteint la LED Gauche

            ; Import des fonctions du fichier RK_Moteur.s
            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_AVANT              ; moteur droit tourne
vers l'avant
            IMPORT  MOTEUR_DROIT_ARRIERE      ; moteur droit tourne vers
l'arrière
            IMPORT  MOTEUR_DROIT_INVERSE      ; inverse le sens de
rotation du moteur droit

            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
            IMPORT  MOTEUR_GAUCHE_ARRIERE           ; moteur gauche tourne
vers l'arrière
            IMPORT  MOTEUR_GAUCHE_INVERSE           ; inverse le sens de
rotation du moteur gauche


__main

            ; Configure les LED, Switchs et Bumpers
            BL  LED_SWITCH_INIT
```

```
            ; Configure les PWM + GPIO
            BL    MOTEUR_INIT


; Lis l'état du bouton poussoir 1, si le bouton est poussé, alors le
programme va à la branche "Cache",
; Sinon, le programme va à la branche "ReadState2" qui lit le bouton
poussoir 2
ReadState
            ldr r11,[r7]
            CMP r11,#0x00
            BNE ReadState2
            BL Cache


; Lis l'état du bouton poussoir 2, si le bouton est poussé, alors le
programme va à la branche "Cherche",
; Sinon, le programme retourne à la branche "ReadState" qui lit le bouton
poussoir 1
ReadState2
            ldr r11,[r10]
            CMP r11,#0x00
            BNE ReadState
            BL Cherche


; Branche du programme de celui qui se cache
Cache
            ; Allumer la LED droite
            BL ALLUME_DROITE

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

            ; Evalbot avance droit devant
            BL    MOTEUR_DROIT_AVANT
            BL    MOTEUR_GAUCHE_AVANT

            ; Avancement pendant une période (deux WAIT)
            BL    WAIT  ; BL (Branchement vers le lien WAIT); possibilité
de retour à la suite avec (BX LR)
            BL    WAIT

            ; Rotation à gauche de l'Evalbot pendant une demi-période (1
seul WAIT), il fait donc ici un quart de tour
            BL    MOTEUR_GAUCHE_ARRIERE   ; MOTEUR_GAUCHE_INVERSE
            BL    WAIT

            ; Avancement pendant une période (deux WAIT)
            BL    MOTEUR_GAUCHE_AVANT
            BL    WAIT
            BL    WAIT
```

```
            ; Désactiver les deux moteurs droit et gauche, le robot
s'arrête donc
            BL    MOTEUR_DROIT_OFF
            BL    MOTEUR_GAUCHE_OFF

            ; Allumer la LED Gauche (Les deux LEDs sont donc allumées)
            BL ALLUME_GAUCHE

            ;Fin du programme de celui qui se cache, on retourne à la
branche ReadState, qui lit l'état du Switch 1
            ; On peut ainsi recommencer la partie
            BL ReadState


; Branche du programme de celui qui cherche
Cherche
            ; Activer les deux moteurs droit et gauche
            BL    MOTEUR_DROIT_ON
            BL    MOTEUR_GAUCHE_ON

; Ici débute la boucle qui fera suivre un pattern à notre robot
loop
            ; Eteindre la LED Gauche puis allumer la LED droite
            BL ETEINT_GAUCHE
            BL ALLUME_DROITE

            ; Evalbot avance droit devant
            BL    MOTEUR_DROIT_AVANT
            BL    MOTEUR_GAUCHE_AVANT

            ; Avancement pendant deux périodes (quatre WAIT),
            ; A chaque étape de l'avancement,
            ; On vérifie s'il y a collision en lisant l'état des Bumpers
grâce à la branche ReadCollision
            BL    ReadCollision
            BL    WAIT
            BL    ReadCollision
            BL    WAIT
            BL    ReadCollision
            BL    WAIT
            BL    ReadCollision
            BL    WAIT
            BL    ReadCollision

            ; Rotation à gauche de l'Evalbot pendant une demi-période (1
seul WAIT), il fait donc ici un quart de tour
            BL    MOTEUR_GAUCHE_ARRIERE   ; MOTEUR_GAUCHE_INVERSE
            BL    WAIT

            ; Avancement du robot pendant 1 WAIT, tout en vérifiant s'il y
a collision
            BL    MOTEUR_GAUCHE_AVANT
            BL    ReadCollision
```

```
            BL    WAIT
            BL    ReadCollision

            ; De nouveau rotation à gauche pendant une demi-période (1
seul WAIT), il refait donc un quart de tour
            BL    MOTEUR_GAUCHE_ARRIERE   ; MOTEUR_GAUCHE_INVERSE
            BL    WAIT

            ; Eteindre la LED droite puis allumer la LED gauche
            BL ETEINT_DROITE
            BL ALLUME_GAUCHE

            ; Avancement pendant deux périodes (quatre WAIT),
            ; A chaque étape de l'avancement,
            ; On vérifie s'il y a collision en lisant l'état des Bumpers
grâce à la branche ReadCollision
            BL    MOTEUR_GAUCHE_AVANT
            BL    ReadCollision
            BL    WAIT
            BL    ReadCollision
            BL    WAIT
            BL    ReadCollision
            BL    WAIT
            BL    ReadCollision
            BL    WAIT
            BL    ReadCollision

            ; Rotation à droite de l'Evalbot pendant une demi-période (1
seul WAIT), il fait donc un quart de tour
            BL    MOTEUR_DROIT_ARRIERE   ; MOTEUR_GAUCHE_INVERSE
            BL    WAIT

            ; Avancement du robot pendant 1 WAIT, tout en vérifiant s'il y
a collision
            BL    MOTEUR_DROIT_AVANT  ; MOTEUR_GAUCHE_INVERSE
            BL    ReadCollision
            BL    WAIT
            BL    ReadCollision

            ; Rotation à droite de l'Evalbot pendant une demi-période (1
seul WAIT), il refait donc un quart de tour
            BL    MOTEUR_DROIT_ARRIERE   ; MOTEUR_GAUCHE_INVERSE
            BL    WAIT

            ; Le robot vient d'effectuer un aller-retour avec un décalage
vers la gauche,
            ; Il ne lui reste plus qu'à effectuer ce même pattern de
déplacement
            ; Jusqu'à ce qu'il entre en contact avec le robot qui se cache
            BL    loop  ; Retour au début de la branche loop


; Branche vérifiant l'état des Bumpers : si un des deux est appuyé,
; Cela signifie que notre robot est entré en contact avec l'autre,
```

```
; On va alors à la branche GAGNE
; Sinon, on revient où on en était dans le code avec BX LR
ReadCollision
            ldr r11,[r5]
            CMP r11,#0x00
            BEQ GAGNE
            ldr r11,[r4]
            CMP r11,#0x00
            BEQ GAGNE
            BX LR


; Code à effectuer par le robot qui cherche une fois qu'il a trouvé
l'autre robot (donc gagné)
GAGNE
            ; Désactiver les deux moteurs droit et gauche, le robot
s'arrête donc
            BL    MOTEUR_DROIT_OFF
            BL    MOTEUR_GAUCHE_OFF

            ; Eteindre les deux LEDs (Droite et Gauche)
            BL    ETEINT_GAUCHE
            BL    ETEINT_DROITE

            ; Faire clignoter les LEDs en appelant successivement les
fonctions d'allumage et d'éteinte
            ; Elles s'allument (pendant une demi-periode WAIT2) et
s'éteignent deux fois chacune
            BL  ALLUME_DROITE
            BL    WAIT2
            BL    ETEINT_DROITE
            BL  ALLUME_GAUCHE
            BL    WAIT2
            BL    ETEINT_GAUCHE
            BL  ALLUME_DROITE
            BL    WAIT2
            BL    ETEINT_DROITE
            BL  ALLUME_GAUCHE
            BL    WAIT2
            BL    ETEINT_GAUCHE

            ; Fin du programme de celui qui cherche, on retourne à la
branche ReadState, qui lit l'état du Switch 1
            ; On peut ainsi recommencer la partie
            BL    ReadState


; Boucle d'attente pour le déplacement
WAIT  ldr r1, =0x8FFFFF
wait1 subs r1, #1
        bne wait1

            ; retour à la suite du lien de branchement
            BX    LR
```

```
; Boucle d'attente pour le clignotement des LED
WAIT2 ldr r1, =0x002FFFFF
wait2 subs r1, #1
        bne wait2

            ; retour à la suite du lien de branchement
            BX    LR

            NOP
        END
```

## Explication

Nous avons réuni les configurations des LEDs, Switchs et Bumpers par souci de simplification : les LEDs se trouvant sur le Port F, les Switchs sur le Port D et les Bumpers sur le Port E, il n'y avait pas de problème d'override.

Pour la configuration de chacun des deux Switchs, des deux LEDs et des deux Bumpers, nous avons effectué une configuration commune (en utilisant le code binaire contenant deux broches, par exemple pour les LEDs, nous avons BROCHE4_5 donc 0x30), puis nous avons séparé leur utilisation en appelant deux registres contenant respectivement une broche chacun (BROCHE4 : 0x10 et BROCHE5 : 0x20).

Nous avons délibérément appelé ce fichier de configuration avant celui de configuration des moteurs, car ce dernier contient des OR permettant de ne pas effacer les configurations faites préalablement.

## Conclusion

Le jeu se déroule correctement, et respecte bien le cahier des charges que nous avions défini au début du projet. Ce projet nous a particulièrement plu, car il a pu nous montrer une approche différente de la programmation, sur des cartes électroniques.