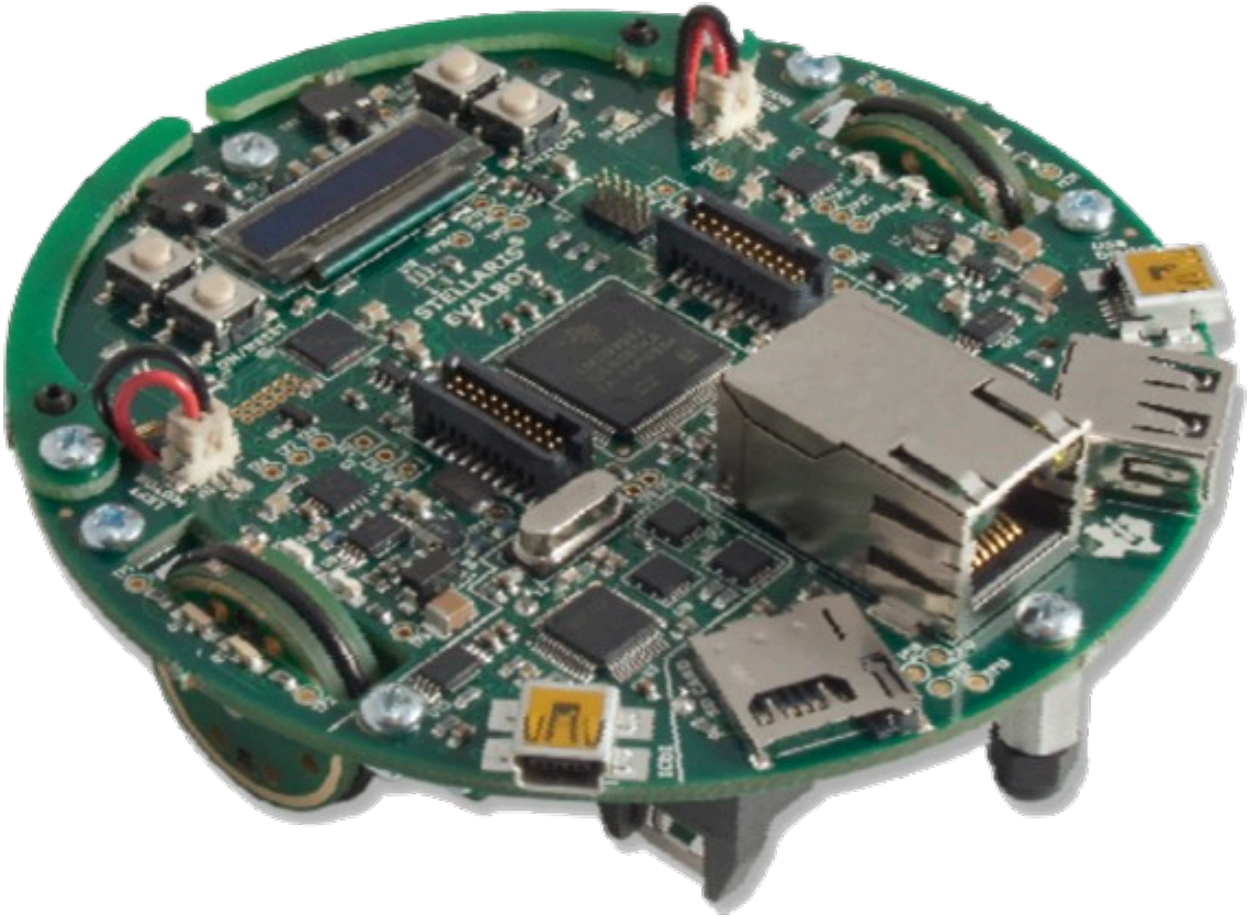
GrassGuardian

CHEN Michel
michel.chen@edu.esiee.fr

BEGHELLI Matteo
matteo.beghelli@edu.esiee.fr



Source : https://www.ti.com/lit/ml/spmu168/spmu168.pdf?ts=1703888565227

Sommaire

1. Description détaillée du projet 3

2. Organigramme et les différents scénarios 4

3. Explication des choix de configuration des GPIO utilisés, du déroulement de chaque programme et des résultats obtenus 6

3.1. GPIO utilisés et ports utilisés 6

3.2. Déroulement de chaque programme et les résultats obtenus 7

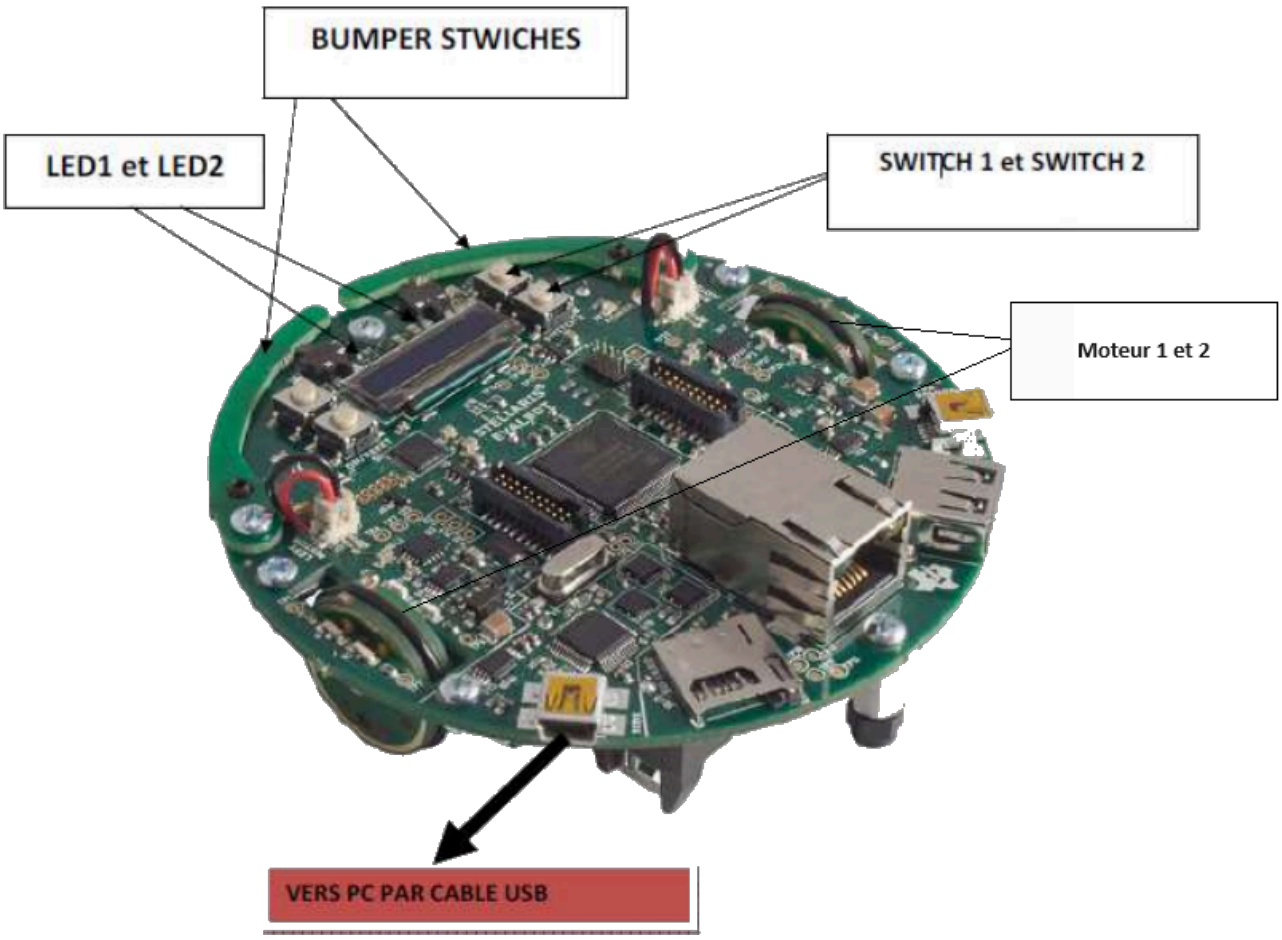
3.2.1. Import et constantes 7

3.2.2. Programme principal 10

1. Description détaillée du projet

Dans le cadre de la programmation d'une carte EvalBot, nous avons dû développer un programme pour simuler le comportement d'un robot tondeuse, en incluant l'utilisation de certains composants : les Leds , les Bumpers, les Switches et les moteurs

Lien du code du projet réalisé : https://github.com/michel-ch/ASM-EvalBot

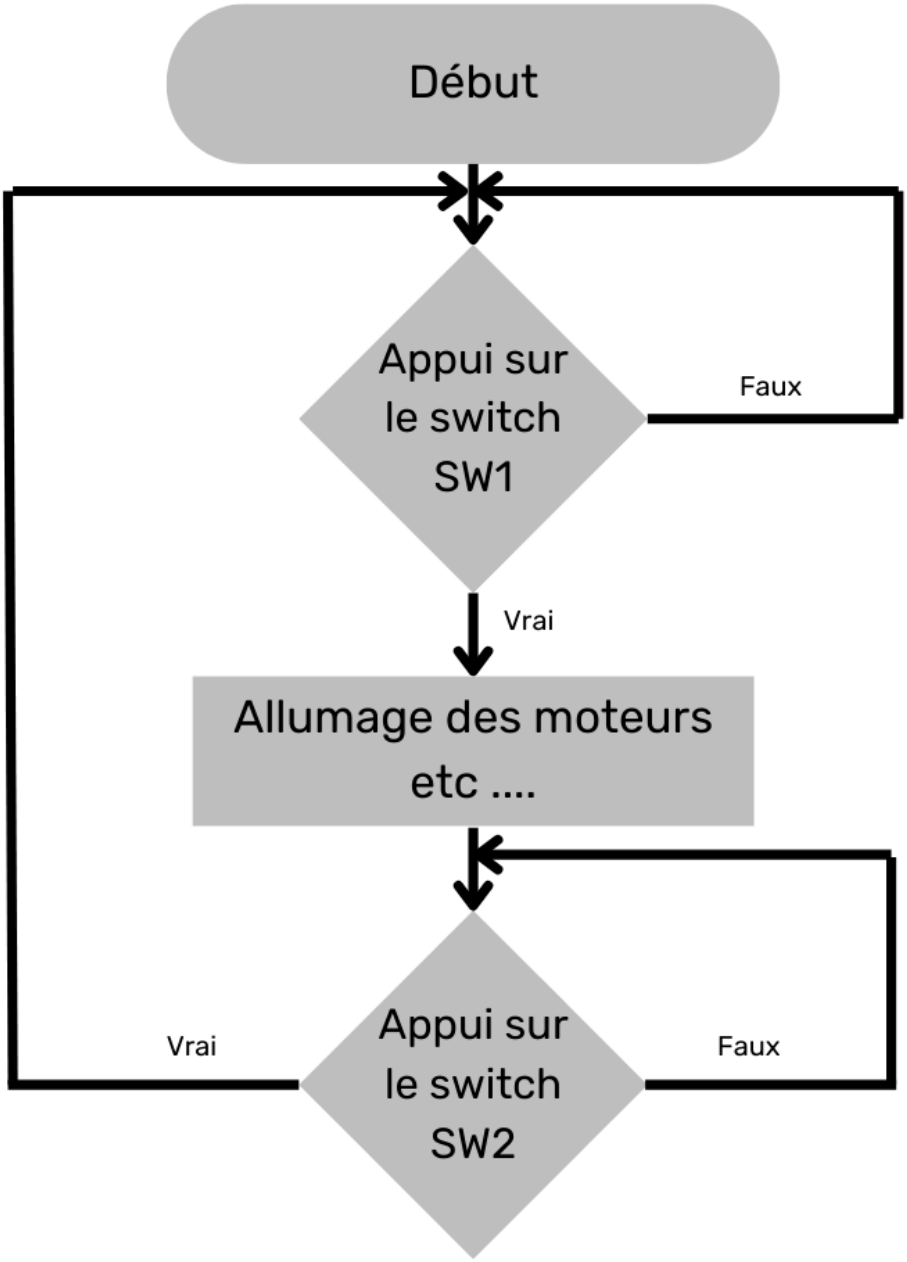


Source : https://blackboard.esiee.fr/bbcswebdav/pid-155650-dt-content-rid-930330_1/xid-930330_1

2. Organigramme et les différents scénarios

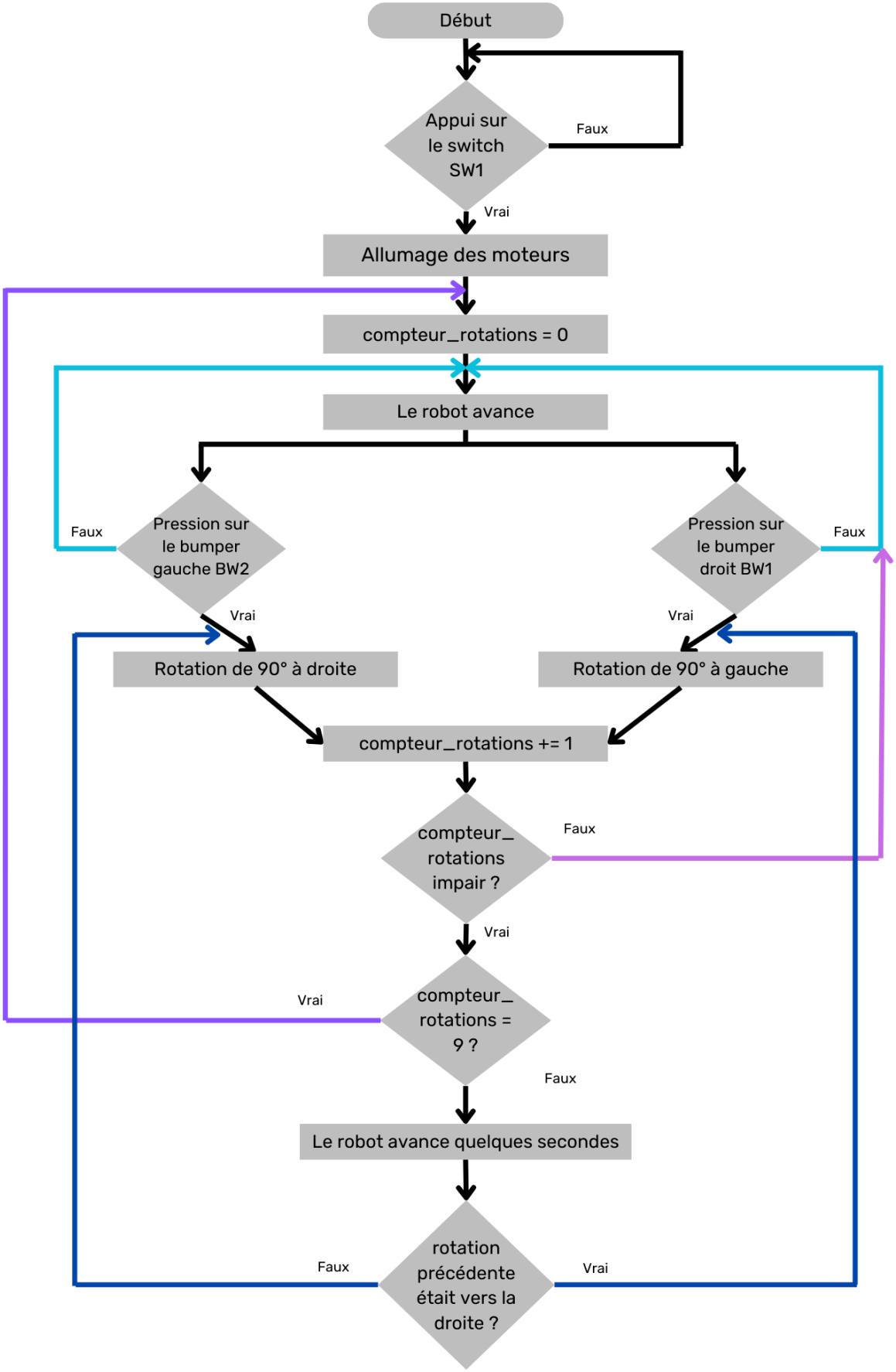
Ces schémas representent le fonction du robot et les différents scénarios.

Actions faites à n'importe quel moment du programme par le robot :



fait par BEGHELLI Matteo

Organigramme complet de l'ensemble du programme du robot :



fait par BEGHELLI Matteo

3. Explication des choix de configuration des GPIO utilisés, du déroulement de chaque programme et des résultats obtenus

3.1. GPIO utilisés et ports utilisés

Les GPIO sont définis dans la documentation ainsi que son utilisation avec les ports. D'après celle-ci :

Les Leds sont sur le port F(`GPIO_PORTF_BASE  EQU  0x40025000` *ligne 12*), sur la broche 4 pour la Led 1 et sur la broche 5 pour la Led 2.

```
; PORT F : selection des leds 1 et 2, BROCHE 4 et 5 du PORT F

PIN4      EQU  0x10

PIN5      EQU  0x20

PIN45     EQU  0x30
```

ligne 52

Les Switches sont sur le port D (`GPIO_PORTD_BASE  EQU  0x40007000` *ligne 15*)sur la broche 6 pour le Switch 1 et sur la broche 7 pour le Switch 2.

```
; PORT D : selection du SW1 et 2 ,BROCHE 6 et 7 du PORT D

PIN6      EQU   0x40

PIN7      EQU  0x80

PIN67     EQU  0xC0
```

ligne 44

Les Bumpers sont sur le port E (`GPIO_PORTE_BASE  EQU  0x40024000` *ligne 18*), sur la broche 0 pour le Bumper Droit et sur la broche 1 pour le Bumper Gauche.

```
; PORT E : selection du BW1 et 2 ,BROCHE 0 et 1 du PORT E

PIN0      EQU  0x01

PIN1      EQU  0x02

PIN01     EQU  0x03
```

ligne 60

Les PIN regroupant deux PIN permettent de pouvoir exploiter les deux broches d'un coup.

Pour les Leds, nous avons utilisé les GPIO :

GPIO_DIR (Direction) : Ce GPIO sert à configurer la broche positionnée en sortie, par défaut toutes les broches sont en entrée. (C'est à dire pouvoir lire la valeur de ce GPIO.)

GPIO_DEN (DigitalEnable) : Ce GPIO sert à activer la fonction numérique permettant la conversion des données numériques (valeurs binaires) en données analogiques (intensité).

GPIO_DR2R (2 mA) : Ce GPIO sert à fixer l'intensité de courant électrique à véhiculer par la broche positionné à 2 mA ; d'autres registres similaires sont également disponibles pour 4 ou 8 mA.

Pour les Bumpers et les Switches, nous avons utilisé les GPIO :

GPIO_DEN (DigitalEnable) : Ce GPIO sert à activer la fonction numérique permettant la conversion des données numériques (valeurs binaires) en données analogiques (intensité).

GPIO_PUR (PulUp) : Ce GPIO par défaut les boutons poussoirs sont reliés à la masse, par la suite l'activation de la résistance pul_up doit se faire en logiciel, par la configuration de ce registre

Source : Documentation Technique Documentation Technique 2 *(avec cartographie des composants page 16)*

3.2. Déroulement de chaque programme et les résultats obtenus

3.2.1. Import et constantes

Afin de pouvoir utiliser le robot EvalBot, nous avons commencé le programme par définir les constantes que nous allons utiliser tout au long de l'execution du programme.

Définition du registre qui contrôle la logique de gestion de l'horloge en mode d'exécution normal :

```
; Ce registre contrôle la logique de gestion de l'horloge en mode d'exécution normal
SYSCTL_RCGC2_R (page 291 du document lm3s9B92.pdf)

SYSCTL_PERIPH_GPIO EQU   0x400FE108
```

ligne 7

Définition des Ports utilisés :

```
; Adresse de base du PORT F
GPIO_PORTF_BASE  EQU  0x40025000

; Adresse de base du PORT D
GPIO_PORTD_BASE  EQU  0x40007000

; Adresse de base du PORT E
GPIO_PORTE_BASE  EQU  0x40024000
```

ligne 11

Définition des GPIO utilisés:

```
; Direction des GPIO (page 417 du document lm3s9B92.pdf)

GPIO_O_DIR    EQU 0x400

; Le registre GPIODR2R contrôle la commande de conduite de 2 mA

; Par défaut, toutes les broches GPIO ont une commande de 2 mA.

; Sélection de la commande 2 mA pour les GPIO - page 428 du document lm3s9B92.pdf

GPIO_O_DR2R    EQU 0x500

; Registre d'activation numérique

; Pour utiliser la broche en tant qu'entrée ou sortie numérique, le bit GPIODEN
correspondant doit être activé.

; Activation numérique des GPIO - page 437 du document lm3s9B92.pdf

GPIO_O_DEN    EQU 0x51C

; Registre pour activer les switchs et les bumpers en logiciel - par défaut, ils sont
reliés à la masse donc inactifs

GPIO_PUR      EQU 0x510
```

ligne 20

Définition des Pins utilisés :

```
; PORT D : selection du SW1 et 2 ,BROCHE 6 et 7 du PORT D

PIN6      EQU    0x40

PIN7      EQU  0x80

PIN67     EQU  0xC0
```

```
; PORT F : selection des leds 1 et 2, BROCHE 4 et 5 du PORT F

PIN4      EQU  0x10

PIN5      EQU  0x20

PIN45     EQU  0x30

; PORT E : selection du BW1 et 2 ,BROCHE 0 et 1 du PORT E

PIN0      EQU  0x01

PIN1      EQU  0x02

PIN01     EQU  0x03
```

ligne 44

Définition de la durée de clignottement :

```
; définit la fréquence de clignotement
DUREE_CLIGNOTEMENT  EQU    0x0015FFFF
```

ligne 68

Définition de la durée d'un avancement :

```
; définit la durée pendant laquelle le robot doit avancer après une rotation
DUREE_AVANCE    EQU    0x008FFFFF
```

ligne 71

Import des configurations des moteurs

```
      ENTRY
      EXPORT   __main

      ;; La commande IMPIN spécifie qu'un symbole est défini dans un objet partagé
lors de l'exécution.
      IMPORT   MOTEUR_INIT             ; initialise les moteurs (configure les
pwms + GPIO)

      IMPORT   MOTEUR_DROIT_ON           ; activer le moteur droit
      IMPORT  MOTEUR_DROIT_OFF         ; d activer le moteur droit
      IMPORT  MOTEUR_DROIT_AVANT         ; moteur droit tourne vers l'avant
      IMPORT  MOTEUR_DROIT_ARRIERE      ; moteur droit tourne vers l'arri re
      IMPORT  MOTEUR_DROIT_INVERSE      ; inverse le sens de rotation du moteur
droit

      IMPORT   MOTEUR_GAUCHE_ON         ; activer le moteur gauche
      IMPORT  MOTEUR_GAUCHE_OFF        ; d activer le moteur gauche
```

```
      IMPORT  MOTEUR_GAUCHE_AVANT         ; moteur gauche tourne vers l'avant
      IMPORT  MOTEUR_GAUCHE_ARRIERE      ; moteur gauche tourne vers l'arri re
      IMPORT  MOTEUR_GAUCHE_INVERSE      ; inverse le sens de rotation du moteur
gauche
```

ligne 74

3.2.2. Programme principal

Cette partie du programme est la fonction principale qui va manipuler les registres disponibles afin de répondre au besoin de ce projet.

Configuration de l'horloge avec les Ports utilisés par le programme

```
; Les Ports de l'horloge sur GPIO F sont connectées aux LED, GPIO E est connecté aux
bumpers et GPIO D est connecté aux interrupteurs : 0x38 == 000111000)
; Activer l'horloge des périphériques pour les Ports F, E et D en définissant les
bits correspondants, (page 291 du document LM3S9B96.pdf), (GPIO::HGFEDCBA)

ldr r2, = SYSCTL_PERIPH_GPIO

mov r4, #0x00000038

str r4, [r2]
```

ligne 95

Ajout d'un délai de 3 cycles d'horloge système avant tout accès au registre GPIO, c'est une étape obligatoire avant toute utilisation des registres

```
; "Il doit y avoir un délai de 3 cycles d'horloge système avant tout accès au
registre GPIO (page 413 du document LM3S9B92.pdf)
; tres tres imPINant....;; pas necessaire en simu ou en debbug ;étape par étape...

   nop
   nop
   nop
```

ligne 104

Configuration des Leds avec les GPIO et Ports précédemment mentionnés

```
; CONFIGURATION LEDS

   ldr r3, = GPIO_PORTF_BASE+GPIO_O_DIR

; une broche (Pin) du PORTF en sortie (broches 4 et 5 : 00110000)

   ldr r4, = PIN45
```

```
   str r4, [r3]
; Configuration du PORT F - Enable Digital Function - PORT F
   ldr r3, = GPIO_PORTF_BASE+GPIO_O_DEN
   ldr r4, = PIN45
   str r4, [r3]
; Choix de l'intensité de sortie (2mA)
   ldr r3, = GPIO_PORTF_BASE+GPIO_O_DR2R
   ldr r4, = PIN45
   str r4, [r3]
```

ligne 125

Configuration des Bumpers et Switches avec les GPIO et Ports précédemment mentionnés

```
; Configuration Switchs
; Configuration du PORT D - Enable Digital Function - PORT D
   ldr r3, = GPIO_PORTD_BASE+GPIO_O_DEN
      ldr r4, = PIN67
      str r4, [r3]
; Activer le registre des switchs, PORT D
   ldr r3, = GPIO_PORTD_BASE+GPIO_PUR
      ldr r4, = PIN67
      str r4, [r3]
; Configuration Bumpers
; Configuration du PORT E - Enable Digital Function - PORT E
   ldr r3, = GPIO_PORTE_BASE+GPIO_O_DEN
      ldr r4, = PIN01
      str r4, [r3]
```

```
; Activer le registre des bumpers, PORT E

   ldr r3, = GPIO_PORTE_BASE+GPIO_PUR

      ldr r4, = PIN01

      str r4, [r3]
```

ligen 151

Lecture du Switch 1 au Port D Broche 6. Si il y a une pression sur le Switch 1, alors Evalbot se met à avancer en allant à allumemoteur. *ligne 205*

```
inst1
; Lecture de l'état du SW1 et rangement cet état dans r5

    ldr r8,= GPIO_PORTD_BASE + (PIN6<<2)

    ldr r11, [r8]

; Si il y a une pression sur le SW1, alors Evalbot se met à avancer, sinon il ne se
passe rien.

    cmp r11,#0x40

    bne allumemoteur

    b inst1
```

ligne 189

Allumage du moteur à partir des imports déjà fait

```
allumemoteur
   ;; BL Branchement vers un lien (sous programme)

   ; Configure les PWM + GPIO

   BL  MOTEUR_INIT

   ; Activer les deux moteurs droit et gauche

   BL  MOTEUR_DROIT_ON
   BL  MOTEUR_GAUCHE_ON
```

ligne 205

Remise à zéro de r2 qui est un compteur de rotation

```
reset_compteur_rota
```

```
   ;(re)initialise le compteur de rotation
   mov r2, #0
```

Boucle principale qui va faire avancer le robot.

```
loop
   ; Boucle de pilotage des 2 Moteurs (Evalbot avance)

   BL  MOTEUR_DROIT_AVANT
   BL  MOTEUR_GAUCHE_AVANT
```

ligne 222

Lecture de l'état du Bumper 1 au Port E Broche 0 et rangement cet état dans r5. Si le Bumper droit ne vaut pas 1, soit il est poussé, cela fera une rotation vers la gauche en allant à rota_gauche. *ligne 332*

```
; Lecture de l'état du BW1 et rangement cet état dans r5

    ldr r7,= GPIO_PORTE_BASE + (PIN0<<2)

    ldr r5, [r7]

; Si pression sur BW1 alors Evalbot tourne sur lui meme dans le sens triginométrique

    cmp r5,#0x01

    bne rota_gauche
```

ligne 228

Lecture de l'état du Bumper 2 au Port E Broche 1 et rangement cet état dans r5. Si le Bumper gauche ne vaut pas 2, soit il est poussé, cela fera une rotation vers la droite en allant à rota_droite. *ligne 268*

```
; Lecture de l'état du BW2 et rangement cet état dans r5

    ldr r7,=GPIO_PORTE_BASE + (PIN1<<2)

    ldr r5,[r7]

; Si pression sur BW2 alors Evalbot tourne sur lui meme dans le sens horaire

    cmp r5,#0x02

    bne rota_droite
```

ligne 240

Lecture de l'état du Switch 2 au Port D Broche 7 et rangement cet état dans r11.
Si le Switch 2 est appuyé alors on entre dans stopmoteur. *ligne 414*
Si rien n'a été appuyé alors nous recommençons la boucle en allant à loop. *ligne 222*

```
; Lecture de l'état du SW2 et rangement cet état dans r11 (et non r5 car les 2
composants doivent pouvoir etre pressés en meme temps)

    ldr r8,= GPIO_PORTD_BASE + (PIN7<<2)

    ldr r11, [r8]

; Si pression sur SW2 alors Evalbot s'arrete

    cmp r11,#0x80

    bne stopmoteur

;Si rien d'appuyé, on reste dans la boucle, Evalbot continue d'avancer

    b loop
```

ligne 252

Rangement 1 dans r4 et incrémentation du compteur de rotation puis rotation vers la droite arrière, redéfinition de r10 et r6 à 0 et allumage de la Led 1.
Lecture de l'état de la Led 1 au Port F Broche 4 et rangement cet état dans r3.

```
rota_droite

   ; définit une rotation à droite
   mov r4, #1

   ; incrémente le compteur de rotation
   add r2, #1

   BL  MOTEUR_DROIT_ARRIERE

; Allumer la led broche 4 (PIN4)

   mov r10, #0x000              ;; pour eteindre LED
   ldr r12, = PIN4             ;; Allume PINF broche 4 : 00010000
   ldr r3, = GPIO_PORTF_BASE + (PIN4<<2)    ;; @data Register = @base + (mask<<2) ==>
LED
   mov r6, #0             ;; initialise un compteur de clignotement
```

ligne 268

Clignotement de la Led 1 après rota_droite *ligne 268* pendant DUREE_CLIGNOTEMENT durée avec incrémentation du compteur r6 à chaque clignotement.
Pendant les clignotements, nous relirons l'état du Switch 2 et nous rangeons cet état dans

r11. Si le Switch 2 est appuyé alors on entre dans stopmoteur. *ligne 414*
Sinon nous continuons en rebouclant dans clin_d. *ligne 288*
Après la fin de la boucle, si la valeur du compteur de rotation (r6) est pair, alors on retourne dans loop. *ligne 222*
Sinon on va à AVANCE_X_SECONDES. *ligne 396*

```
; CLIGNOTEMENT led 1

clin_d
       str r10, [r3]          ;; Eteint LED car r2 = 0x00
       ldr r9, = DUREE_CLIGNOTEMENT   ;; pour la duree de la boucle d'attente1
(wait1_d)
    add r6, #1           ;; incrementation du compteur de clignotement
    cmp r6, #4           ;; au bout de 3 clignotements il retourne dans l'état où il
avance
    beq avance_d
    b wait1_d

avance_d              ;; retourne dans loop si le compteur de rotation est pair,
c'est à dire si la séquence avance puis rotation contraire vient de se produire
    cmp r2, #2
    beq loop
    cmp r2, #4
    beq loop
    cmp r2, #6
    beq loop
    cmp r2, #8
    beq loop

    b AVANCE_X_SECONDES       ;; autrement effectue la séquence avance puis
rotation contraire


wait1_d  subs r9, #1
       bne wait1_d

       str r12, [r3]        ;; Allume PINF broche 4 : 00010000 (contenu de r3)
       ldr r9, = DUREE_CLIGNOTEMENT  ;; pour la duree de la boucle d'attente2
(wait2_d)

wait2_d    subs r9, #1
       bne wait2_d

; Lecture de l'état du SW2 et rangement cet état dans r11

    ldr r8,= GPIO_PORTD_BASE + (PIN7<<2)

    ldr r11, [r8]

; Si pression sur SW2 alors Evalbot s'arrête
```

```
    cmp r11,#0x80

    bne stopmoteur

    b clin_d
```

ligne 286

Rangement 2 dans r4 et incrémentation du compteur de rotation puis rotation vers la gauche arrière, redéfinition de r10 et r6 à 0 et allumage de la Led 2.
Lecture de l'état de la Led 2 au Port F Broche 5 et rangement cet état dans r3.

```
rota_gauche

   ; définit une rotation à gauche
   mov r4, #2

   ; incrémente le compteur de rotations
   add r2, #1

   BL  MOTEUR_GAUCHE_ARRIERE


; Allumer la led broche 5 (PIN5)

   mov r10, #0x000              ;; pour eteindre LED
   ldr r12, = PIN5              ;; Allume PINF broche 5 : 00010000
   ldr r3, = GPIO_PORTF_BASE + (PIN5<<2)    ;; @data Register = @base + (mask<<2) ==>
LED
   mov r6, #0             ;; initialise un compteur de clignotement
```

ligne 332

Clignotement de la Led 2 après rota_gauche *ligne 332* pendant DUREE_CLIGNOTEMENT durée avec incrémentation du compteur r6 à chaque clignotement.
Pendant les clignotements, nous relirons l'état du Switch 2 et nous rangeons cet état dans r11. Si le Switch 2 est appuyé alors on entre dans stopmoteur. *ligne 414*
Sinon nous continuons en rebouclant dans clin_g. *ligne 353*
Après la fin de la boucle, si la valeur du compteur de rotation (r6) est pair, alors on retourne dans loop. *ligne 222*
Sinon on va à AVANCE_X_SECONDES. *ligne 396*

```
; CLIGNOTEMENT led 2

clin_g
       str r10, [r3]          ;; Eteint LED car r2 = 0x00
       ldr r9, = DUREE_CLIGNOTEMENT   ;; pour la duree de la boucle d'attente1
(wait1_g)
    add r6, #1          ;; incrementation du compteur de clignotement
```

```
    cmp r6, #4          ;; au bout de 3 clignotements il retourne dans l'état où  il
avance
    beq avance_g
    b wait1_g

avance_g              ;; retourne dans loop si le compteur de rotation est pair,
c'est à dire si la séquence avance puis rotation contraire vient de se produire
    cmp r2, #2
    beq loop
    cmp r2, #4
    beq loop
    cmp r2, #6
    beq loop
    cmp r2, #8
    beq loop

    b AVANCE_X_SECONDES       ;; autrement effectue la séquence avance puis
rotation contraire

wait1_g  subs r9, #1
       bne wait1_g

       str r12, [r3]        ;; Allume PINF broche 5 : 00010000 (contenu de r3)
       ldr r9, = DUREE_CLIGNOTEMENT  ;; pour la durée de la boucle d'attente2
(wait2_g)

wait2_g    subs r9, #1
       bne wait2_g

; Lecture de l'état du SW2 et rangement cet état dans r11

    ldr r8,= GPIO_PORTD_BASE + (PIN7<<2)

    ldr r11, [r8]

; Si pression sur SW2 alors Evalbot s'arrête

    cmp r11,#0x80

    bne stopmoteur

       b clin_g
```

ligne 351

Boucle faisant avancer le robot pendant DUREE_AVANCE durée. Si le robot 5 fois de suite un obstacle, le robot reprend sa route depuis reset_compteur_rota. *ligne 217*
Si r4 est égal à 1, on repart dans rota_gauche *ligne 332*, sinon on repart dans rota_droite. *ligne 268*

```
AVANCE_X_SECONDES

  ldr r9, = DUREE_AVANCE

  ; fait avancer le robot pendant un temps donné
  BL  MOTEUR_DROIT_AVANT
  BL  MOTEUR_GAUCHE_AVANT

  cmp r2, #9         ;; si le robot a rencontré 5 fois de suite un obstacle, alors on
considère que cet obstacle est un mur et le robot reprend sa route
  beq reset_compteur_rota

wait  subs r9, #1
      bne wait

   cmp r4, #1
   beq rota_gauche
   b rota_droite
```

ligne 396

Extinction du moteur gauche et droit, extinction des Leds 1 et 2 et retour au début inst1

```
stopmoteur
; Desactiver les moteurs
   BL  MOTEUR_DROIT_OFF
   BL  MOTEUR_GAUCHE_OFF

; Éteindre les leds
   str r10, [r3]

; Retour dans l'état d'attente d'appui sur SW1
   b inst1
```

ligne 414

Merci de nous avoir lu.