GEII

Rapport de projet d'étude et réalisation

Carte de développement 68HC11

BOUVRY Colin
SIBILLE Julien

Année 2002-2003

SOMMAIRE

1. SUJET..... Page 1

2. PROGRAMMES ..... Page 2

2.1. PROGRAMME TESTANT LA DETECTION DE PISTE..... Page 2

2.2. PROGRAMME POUR TROUVER LES VALEURS DE COMPENSATIONS POUR QUE LE ROBOT AVANCE TOUT DROIT ..... Page 2

2.3. PROGRAMME PERMETTANT DE TESTER LA CARTE HACHEUR..... Page 3

2.4. PREMIERE VERSION D'UN SUIVIT DE PISTE ..... Page 4

2.5. PROGRAMME DE CONTOURNEMENT D'OBSTACLE..... Page 6

2.6. PROGRAMME GENERALE..... Page 10

3. UTILISATION DU CONNECTEUR D ..... Page 21

4. PROBLEME POSE PAR L'ARRET AVEC REDEMARRAGE..... Page 21

5. ALIMENTATION DES LAMPES..... Page 22

6. CONCLUSION..... Page 23

1. SUJET

Le but de ce sujet est de poursuivre la réalisation et la mise au point du robot développé l'année dernière et ayant participé au concours robotique. Basée sur l'utilisation d'une carte mère à base de microcontrôleur 68HC11, ce robot possède déjà certaines fonctions comme la reconnaissance de piste, la commande moteur ou l'arrêt de piste et est parfaitement autonome. Toutefois, certaines fonctions sont à améliorer et d'autres, comme la gestion d'obstacles sont à introduire pour cette nouvelle version. Il s'agit donc de développer un robot mobile répondant au règlement actuel et prenant en compte ses éventuelles modifications pour l'année à venir en améliorant ou en développant de nouvelles cartes électroniques et en réalisant les programmes assembleur correspondants.

Tout d'abord, nous avons appris l'assembleur du 68HC11.
Ensuite, nous avons étudié les cartes présentes.
Enfin, nous avons créé des programmes utilisant l'ensemble des cartes. Nous nous sommes aperçus que l'équipe de l'an denier avait un bon suivit de piste très rapide et nous l'avons repris.

Nous avons apporté au robot :

- Un réglage de l'intensité lumineuse pour une meilleure détection de la piste
- Un contournement d'obstacle
- Un démarrage sans piste
- Une gestion de la priorité à droite
- Une gestion du ralentissement du robot
- Une gestion de l'arrêt définitif du robot
- Une simplification du programme

2. PROGRAMMES

2.1. PROGRAMME TESTANT LA DETECTION DE PISTE

Ce programme sert à vérifier la détection de la piste. En effet, il est bon de savoir que le microcontrôleur reçoit bien les bonnes informations. Pour cela, on lit les informations sur la carte de détection de piste et on envoie ces informations sur une plaque « Labdec » permettant de les visualiser avec des leds. Il est ainsi facile de voir si la détection est bien prise en compte par le microcontrôleur en comparant les valeurs des leds sur la carte de détection de piste avec celle qui sont sur la plaque « Labdec ».

Programme detecte.a11

```
porta  equ   $1000  ; adresse du port a
ddra   equ   $1001  ; registre de direction du port a
porte  equ   $100A ; adresse du port e

       org   $8000  ; adresse du début de l'EEPROM
       ldab  #$FF   ; configuration du
       stab  ddra   ; port a en sortie
boucle ldaa  porte  ; charge les données du port e
       staa  porta  ; envoie des données sur port a
       bra   boucle ; boucle infinie
       end
```

2.2. PROGRAMME POUR TROUVER LES VALEURS DE COMPENSATIONS POUR QUE LE ROBOT AVANCE TOUT DROIT

Ce programme sert à rechercher les valeurs de compensations qu'il faut envoyer sur la carte hacheur pour que le robot avance le plus droit possible.
On a constaté que les moteurs ne tournent pas à la même vitesse pour les mêmes signaux de commande. Par exemple, si on envoie $60 sur le port a et le complément de $60 sur le port b (collecteurs ouverts), les moteurs ne tournent pas à la même vitesse et le robot dévira de sa trajectoire. On peut ainsi avec ce programme chercher les bonnes valeurs de compensation en tâtonnant.

Programme tdroit.a11

```
portb  equ   $1060
ddra   equ   $1001
portd  equ   $1008
ddrd   equ   $1009

       org   $8000  ; adresse du début de l'EEPROM
       ldaa  #$FF   ; configuration du
       staa  ddra   ; port A en sortie
       clra         ; on met $00 dans A pour configurer
       staa  ddrd   ; le port D en entrée
```

```
start   ldaa    #$60    ; on charge la valeur $60 dans l'accumulateur A
        staa    porta   ; on envoie la valeur de l'accumulateur A sur le port a
        ldab    #$5a    ; on charge la valeur $5a dans l'accumulateur B
        comb            ; on le complémente car le port B est à collecteurs ouvert
        stab    portb   ; on envoie la valeur de l'accumulateur B sur le port b
        bra     start   ; on boucle pour que le robot avance

        end
```

2.3. PROGRAMME PERMETTANT DE TESTER LA CARTE HACHEUR

Ce programme sert à vérifier la carte hacheur et de savoir si la valeur de compensation est toujours la même pour différentes vitesses.
Il sert à vérifier en cas de panne ou est le problème. On peut ainsi faire tous les tests possibles en regardant sur les points tests de la carte hacheur.

On incrémente la vitesse toutes les 0.2 secondes. Les moteurs s'arrêtent à une vitesse fixée.

Programme tdroitva.a11 :

```
porta   equ     $1000
portb   equ     $1060
ddra    equ     $1001
portd   equ     $1008
ddrd    equ     $1009

        org     $8000           ; adresse du début de l'EEPROM
        ldaa    #$FF            ; configuration du
        staa    ddra            ; port A en sortie
        clra                    ; on met $00 dans A pour configurer
        staa    ddrd            ; le port D en entrée

; avancement du robot
                                ; pour envoyer $FF sur le port A, il faut charger $FF
                                ; pour envoyer $FF sur le port B, il faut charger $00 à cause des
                                sorties à collecteurs ouverts

        ldaa    #$01            ; vitesse du départ initial
start   jsr     compmot         ; va au sous programme de compensation

; temporisation d'environ 0.2 seconde

        ldx     #$FFFF          ; on charge $FFFF dans l'accumulateur x( 16 bits)
tempo   dex                     ; on décrémente de un l'accumulateur x
        bne     tempo           ; boucle tant que la valeur $0000 n'a pas été atteinte


; varation de la vitesse
```

```
        inca                ; on incrémente l'accumulateur a de un pour augmenter la
                            vitesse
        cmpa    #$F3        ; on compare l'accumulateur a à $F3
        beq     arret       ; on sort de la boucle quand $F3 est atteint et on va au sous-
programme                   arrêt
        bra     start       ; boucle tant que la valeur $F3 n'a pas été atteinte
; arrêt du robot


arret   ldaa    #$00        ; valeur pour arrêter le port b $00 et le port a $FF
        staa    portb
        coma
        staa    porta
        bra     arret       ; boucle infinie pour un arrêt définitif

compmot                     ;compensation du moteur
        staa    porta
        adda    #$05
        coma
        staa    portb
        coma
        suba    #$05
        rts
        end
```

On a remarqué pour différentes vitesses que la valeur de compensation change.

2.4. PREMIERE VERSION D'UN SUIVIT DE PISTE

Ce programme permet au robot de suivre la piste.
Il sera utilisé dans le programme général pour le ralentissement.
Tout d'abord, on lit la valeur détectée de la piste, ensuite on la compare à la valeur centrale $18 (hexadécimale) pour laquelle le robot avance tout droit. Ceci, dans le but de savoir si le robot a dévié à droite ou à gauche de la piste.
Valeur centrale de référence sur 8 bits :

0	0	0	1	1	0	0	0

Enfin, on agit sur les moteurs du robot de façons différentes suivant la déviation gauche ou droite pour qu'il soit de nouveau centré. Le programme se répète tant la valeur détectée est différente de $00.

Programme lent.a11 :

```
porta   equ     $1000
portb   equ     $1060
ddra    equ     $1001
portd   equ     $8
ddrd    equ     $1009
```

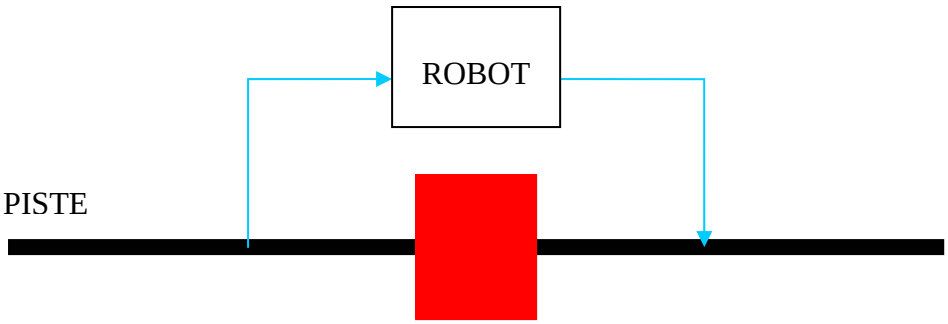
```
porte   equ     $100a
portc   equ     $61

        org     $8000           ; adresse du début de l'EEPROM
        ldaa    #$FF            ; configuration du
        staa    ddra            ; port A en sortie
        clra                    ; on met $00 dans A pour configurer
        staa    ddrd            ; le port D en entrée
        ldx     #$1000

start   ldab    porte           ; on enregistre la valeur de détection dans l'accumulateur b
        cmpb    #$18            ; on compare à $18 car c'est la valeur où le robot va tout droit
        beq     toutdroit       ; branche si #$18 = #$18 (z=1)
        cmpb    #$18
        bcc     motgauche       ; branche si < #$18 (c = 0)
        cmpb    #$18
        bcs     motdroit        ; branche si > #$18 (c = 1)
        cmpb    #$00
        beq     arret           ; branche si #$00 = #$00 (z=1)
        bra     start

motgauche
        subb    #$18            ; port e - #$18
        ldaa    #$60            ; consigne de vitesse
        staa    porta
        sba                     ; retrait de la valeur de compensation (a-b->a)
        coma                    ; complémentation du port b car celui-ci est à collecteur ouvert
        staa    portb
        bra     start

motdroit
        subb    #$18            ; porte - #$18
        ldaa    #$60            ; consigne de vitesse
        aba                     ; ajout de la valeur de compensation
        staa    porta
        sba                     ; retrait de la valeur de compensation
        coma                    ; complémentation du port b car celui-ci est à collecteur ouvert
        staa    portb
        bra     start           ; boucle tant que la valeur de détection est différente de $00

toutdroit
        ldaa    #$60
        staa    porta
        coma
        staa    portb
        bra     start

arret
        ldaa    #$00
        staa    porta
```

```
        coma
        staa    portb
        bra     arret

        end
```

Ce programme est valable seulement pour une vitesse. Le robot saccade légèrement mais le suivit de piste reste fiable.

2.5. PROGRAMME DE CONTOURNEMENT D'OBSTACLE

Ce programme permet le contournement d'un obstacle détecté sur la piste. En effet, une fois que l'ordre de la présence d'un obstacle est reçu, le robot doit être capable de sortir de la piste et la reprendre une fois l'obstacle passé.



OBSTACLE

On n'est pas obligé de tourner à 90 °, on peut tourner pour différents angles.

Programme contour.a11 :

```
porta   equ     $1000
portb   equ     $1060
ddra    equ     $1001
portd   equ     $8
ddrd    equ     $1009
porte   equ     $100a
portc   equ     $61
nbms    rmb     2               ; nbms est une variable, qui sert à enregistrer la valeur de la
                                ;temporisation sur 2 octets

        org     $8000           ; adresse du début de l'EEPROM
        ldaa    #$FF            ; configuration du
        staa    ddra            ; port A en sortie
        clra                    ; on met $00 dans A pour configurer
        staa    ddrd            ; le port D en entrée
```

```
        ldx     #$1000

; programme principal
; le programme commence quand on appuie sur le bouton reset.
; Entre le moment où l'on appuie sur le reset et la validation du moteur, il se passe un
certain temps. Or le programme commence quand on appuie sur le reset , on louperai ainsi le
début du mouvement du robot. Pour cette raison, il faut faire une petite temporisation.

start   ldx     #3000           ; on charge la valeur 3000 afin de faire une tempo de 3s
        stx     nbms            ; on met cette dernière valeur dans la ram
        jsr     temponbms       ; saut vers le sous-programme de temporisation
        jsr     contour         ; saut vers le sous-programme qui se charge du contournement
        jsr     arret           ; arrêt du robot

contour         ; sous-programme de contournement d'obstacle
        jsr     arrettemp       ; arrêt du robot pendant un temps court pour supprimer l'inertie
        jsr     droite90        ; tourne à droite
        ldx     #300            ; tempo de 300 ms
        stx     nbms
        jsr     temponbms
        jsr     arrettemp

        jsr     toutdroit       ; tout droit
        ldx     #2000           ; tempo de 2s
        stx     nbms
        jsr     temponbms
        jsr     arrettemp

        jsr     gauche90        ; tourne à gauche
        ldx     #400            ; tempo de 400ms
        stx     nbms
        jsr     temponbms
        jsr     arrettemp

        jsr     toutdroit       ; tout droit
        ldx     #5000           ; tempo de 5s
        stx     nbms
        jsr     temponbms
        jsr     arrettemp

        jsr     gauche90        ; tourne à gauche
        ldx     #300            ; tempo de 300ms
        stx     nbms
        jsr     temponbms
        jsr     arrettemp

        jsr     toutdroitpiste  ; tout droit jusqu'à la piste
        jsr     arrettemp

        jsr     droite90        ; tourne à droite
```

```
        ldx     #300
        stx     nbms
        jsr     temponbms
        jsr     arrettemp
        rts

droite90                ; sous-programme pour tourner à droite
        ldaa    #$00
        staa    porta
        ldab    #$80
        comb
        stab    portb
        rts

gauche90                ; sous-programme pour tourner à gauche
        ldaa    #$80
        staa    porta
        ldab    #$ff
        stab    portb
        rts

toutdroit               ; sous-programme pour aller tout droit
        ldaa    #$60
        staa    porta
        ldab    #$5a
        comb
        stab    portb
        rts

toutdroitpiste          ; sous-programme qui permet d'aller tout droit jusqu'à la
                        ; détection de la piste
        ldaa    #$60
        staa    porta
        ldab    #$5a
        comb
        stab    portb
        jsr     moyenneur       ; saut vers moyenneur pour éviter d'éventuelles erreurs de
                                ; détection
        cmpa    #$00            ; comparaison de la moyenne et de la valeur 0
        beq     toutdroitpiste  ; boucle tant que la moyenne est égal à 0
        rts

arrettemp               ; on arrête le robot pendant 500 ms pour annuler l'inertie
        ldaa    #$00
        staa    porta
        coma
        staa    portb
        ldx     #500
        stx     nbms
        jsr     temponbms
```

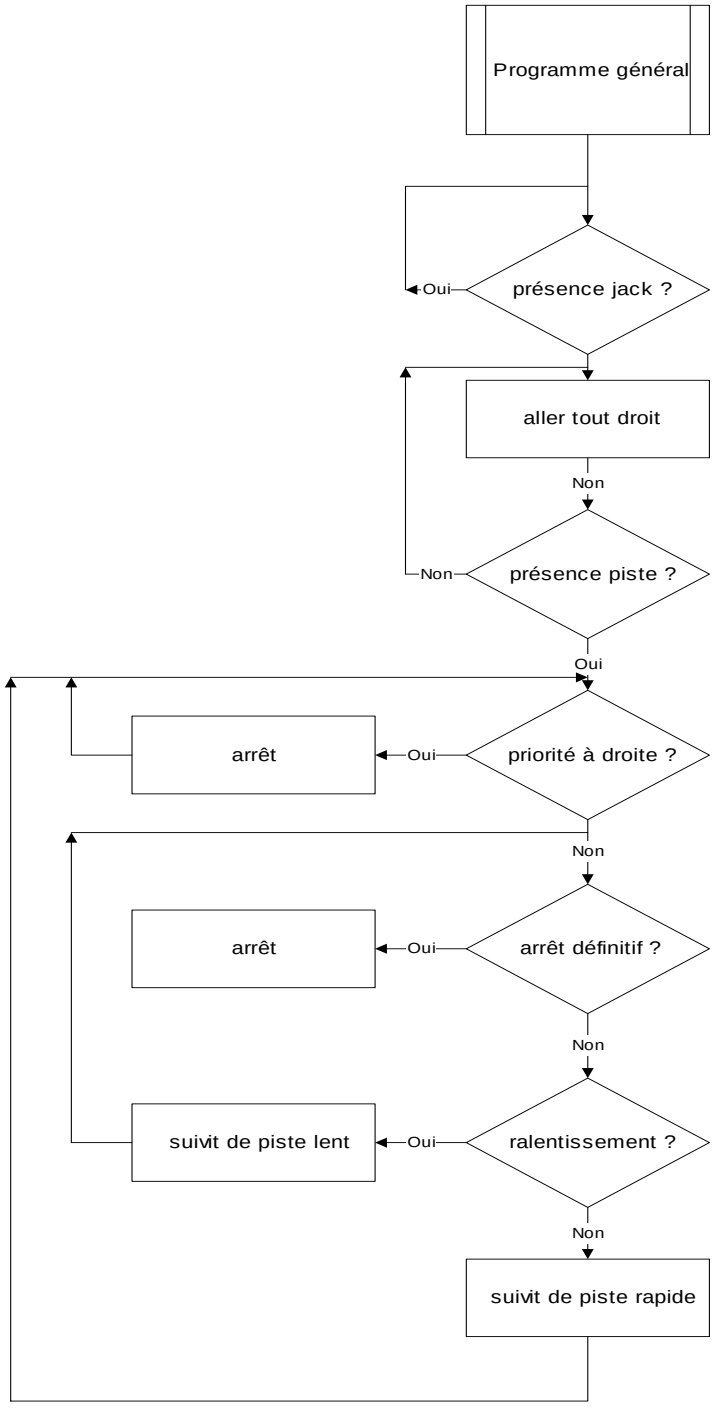
```
        rts

temponbms           ; sous-programme de temporisation réglable à la milliseconde
        ldy     nbms    ; on charge dans y la durée de la
lbl1    ldx     #500    ; on met dans x la valeur qui permet d'avoir une tempo réglable de 1 ms
lbl2    dex             ; on décrémente x
        bne     lbl2    ; on boucle tant que x est différent de 0
        dey             ; on décrémente y
        bne     lbl1    ; on boucle tant que y est différent de 0
        ldx     #1000   ; sert pour le test des bits
        rts


; sous-programme moyenneur qui enlève les états parasites
; le but est d'additionner 255 échantillons des valeurs détectées à des intervalles
; de temps différents et de diviser la somme par 255. On obtient ainsi une moyenne.
moyenneur
        ldx     #$0000
        ldaa    #$00
suite   ldab    porte
        abx             ; addition de b avec x. (b+x -> x)
        inca
        cmpa    #$ff    ; z=0 si a-ff=0
        bne     suite   ; branche si z=0
        ldd     #$00ff
        xgdx            ; échange d avec x. d <-> x
        idiv            ; d/x résultat dans x et reste dans d
                        ; La somme des valeurs détectées sur le porte
                        ; divisée par le nombre de boucle
        xgdx            ; transfert le résultat x dans l'accumulateur d, s'est à dire dans b
        tba             ; b -> a
        ldx     #1000   ; on remet la valeur original dans x
        rts

arret                   ; arrêt définitif
        ldaa    #$00
        staa    porta
        coma
        staa    portb
        bra     arret

        end
```

2.6. PROGRAMME GENERALE

Ce programme prend en compte :

- le démarrage avec le jack
- le départ sans piste
- le suivit de piste rapide et lent (ralentissement)
- la priorité à droite
- le contournement d'obstacle

Algorithme du programme général :



Programme general.a11 :

```
porta   equ     $1000
portb   equ     $1060
ddra    equ     $1001
portd   equ     $8
ddrd    equ     $1009
porte   equ     $100a
portc   equ     $61
nbms    rmb     2

        org     $8000           ; adresse du début de l'EEPROM
        ldaa    #$FF            ; configuration du
        staa    ddra            ; port A en sortie
        clra                    ; on met $00 dans A pour configurer
        staa    ddrd            ; le port D en entrée
        ldx     #$1000

jack    brclr   portd,x %00010000 jack      ; bit6 (PD4) pour le jack, se met à
                                            ; 1 quand on enlève le jack
        jsr     toutdroitpiste              ; saut vers un sous programme pour aller
                                            ; tout droit tant que la piste n'est pas détectée
start   ldaa    portd
        brclr   portd,x %00100000 suite1    ; bit5 (PD5) pour le redémarrage

; arret avec redémarrage
        ldaa    #$00
        staa    porta
        coma
        staa    portb
        bra     start

; test de bits
suite1
        brset   portd,x %00001000 arret     ; bit7 (PD3) pour arrêt définitif
        brset   portd,x %00000100 vitessemin ; bit 8 (PD2) pour le ralentissement quand
                                            ; obstacle
        jsr     vitessemax
        bra     start


; suivit de piste à la vitesse maximale
vitessemax
;       jsr     moyenneur   ;on charge dans A la moyenne des valeurs détectées sur le port e
        ldaa    porte       ; on choisit soit le moyenneur, soit le port e
        cmpa    #$80        ; on réalise la comparaison avec les 21 valeurs déterminées
        beq     var1        ; et on branche aux corrections appropriées pour les moteurs
        cmpa    #$C0
        beq     var2
        cmpa    #$E0
```

```
        beq     var3
        cmpa    #$40
        beq     var3
        cmpa    #$60
        beq     var4
        cmpa    #$70
        beq     var5
        cmpa    #$20
        beq     var5
        cmpa    #$30
        beq     var6
        cmpa    #$38
        beq     var7
        cmpa    #$10
        beq     var7
        cmpa    #$18
        beq     var8
        cmpa    #$1C
        beq     var9
        cmpa    #8
        beq     var9
        cmpa    #$C
        beq     var10
        cmpa    #$E
        beq     var11
        cmpa    #4
        beq     var11
        cmpa    #6
        beq     var12
        cmpa    #7
        beq     var13
        cmpa    #2
        beq     var13
        cmpa    #3
        beq     var14
        cmpa    #1
        beq     var15
        rts

var1    ldaa    #$F0    ; virage à gauche
        ldab    #$B6    ; correction maximum moteur gauche
        staa    porta
        stab    portb
        rts

var2    ldaa    #$F0    ; virage à gauche
        ldab    #$A9
        staa    porta
        stab    portb
        rts
```

```
var3    ldaa    #$E0   ; virage à gauche
        ldab    #$8B
        staa    porta
        stab    portb
        rts

var4    ldaa    #$E0   ;virage à gauche
        ldab    #$78
        staa    porta
        stab    portb
        rts

var5    ldaa    #$E0   ;virage à gauche
        ldab    #$61
        staa    porta
        stab    portb
        rts

var6    ldaa    #$E0   ;virage à gauche
        ldab    #$4B
        staa    porta
        stab    portb
        rts

var7    ldaa    #$E0   ;virage à gauche
        ldab    #$38
        staa    porta
        stab    portb
        rts

var8    ldaa    #$E0   ;droit  e0
        ldab    #$28   ;28
        staa    porta
        stab    portb
        rts

var9    ldaa    #$D0   ;virage à droite
        ldab    #$28
        staa    porta
        stab    portb
        rts

var10   ldaa    #$BD   ;virage à droite
        ldab    #$28
        staa    porta
        stab    portb
        rts

var11   ldaa    #$A7   ;virage à droite
```

```
        ldab    #$28
        staa    porta
        stab    portb
        rts

var12   ldaa    #$90    ;virage à droite
        ldab    #$28
        staa    porta
        stab    portb
        rts

var13   ldaa    #$70    ;virage à droite
        ldab    #$28
        staa    porta
        stab    portb
        rts

var14   ldaa    #$60    ;virage à droite
        ldab    #$11
        staa    porta
        stab    portb
        rts

var15   ldaa    #$50    ;virage à droite
        ldab    #$11
        staa    porta
        stab    portb
        rts

; suivit de piste à une vitesse plus lente
vitessemax2
;       jsr     moyenneur
        ldaa    porte
        cmpa    #$80
        beq     v1ar1
        cmpa    #$C0
        beq     v1ar2
        cmpa    #$E0
        beq     v1ar3
        cmpa    #$40
        beq     v1ar3
        cmpa    #$60
        beq     v1ar4
        cmpa    #$70
        beq     v1ar5
        cmpa    #$20
        beq     v1ar5
        cmpa    #$30
        beq     v1ar6
        cmpa    #$38
```

```
        beq     v1ar7
        cmpa    #$10
        beq     v1ar7
        cmpa    #$18
        beq     v1ar8
        cmpa    #$1C
        beq     v1ar9
        cmpa    #8
        beq     v1ar9
        cmpa    #$C
        beq     v1ar10
        cmpa    #$E
        beq     v1ar11
        cmpa    #4
        beq     v1ar11
        cmpa    #6
        beq     v1ar12
        cmpa    #7
        beq     v1ar13
        cmpa    #2
        beq     v1ar13
        cmpa    #3
        beq     v1ar14
        cmpa    #1
        beq     v1ar15
        rts


v1ar1   ldaa    #$D0    ;virage à gauche
        ldab    #$B0    ;correction maximum moteur gauche
        staa    porta
        stab    portb
        rts

v1ar2   ldaa    #$D0    ;virage à gauche
        ldab    #$A4
        staa    porta
        stab    portb
        rts

v1ar3   ldaa    #$C0    ;virage à gauche
        ldab    #$94
        staa    porta
        stab    portb
        rts

v1ar4   ldaa    #$C0    ;virage à gauche
        ldab    #$86
        staa    porta
        stab    portb
```

```
        rts

v1ar5   ldaa    #$C0    ;virage à gauche
        ldab    #$75
        staa    porta
        stab    portb
        rts

v1ar6   ldaa    #$C0    ;virage à gauche
        ldab    #$65
        staa    porta
        stab    portb
        rts

v1ar7   ldaa    #$C0    ;virage à gauche
        ldab    #$4A
        staa    porta
        stab    portb
        rts

v1ar8   ldaa    #$C0    ;droit
        ldab    #$44
        staa    porta
        stab    portb
        rts

v1ar9   ldaa    #$BA    ;virage à droite
        ldab    #$44
        staa    porta
        stab    portb
        rts

v1ar10  ldaa    #$A0    ;virage à droite
        ldab    #$44
        staa    porta
        stab    portb
        rts

v1ar11  ldaa    #$90    ;virage à droite
        ldab    #$44
        staa    porta
        stab    portb
        rts

v1ar12  ldaa    #$80    ;virage à droite
        ldab    #$44
        staa    porta
        stab    portb
        rts
```

```
v1ar13 ldaa   #$70   ;virage à droite
       ldab   #$44
       staa   porta
       stab   portb
       rts

v1ar14 ldaa   #$60   ;virage à droite
       ldab   #$34
       staa   porta
       stab   portb
       rts

v1ar15 ldaa   #$50   ;virage à droite
       ldab   #$34
       staa   porta
       stab   portb
       rts

arret                ; sous-programme pour l'arrêt définitif
       ldaa   #$00
       staa   porta
       coma
       staa   portb
       bra    arret

toutdroitpiste       ; sous-programme pour aller tout droit tant que la piste n'est pas
                     ; détectée
       ldaa   #$60   ; on avance tout droit à une vitesse lente
       staa   porta
       ldab   #$5a
       comb
       stab   portb
       jsr    moyenneur      ; saut vers le moyenneur pour ne pas détecter des erreurs
       cmpa   #$00           ; on compare à 0 pour savoir si la piste est détectée
       beq    toutdroitpiste ; boucle tant que la valeur est nulle
       rts

moyenneur                    ; sous programme moyenneur
       ldx    #$0000
       ldaa   #$00
suite  ldab   porte
       abx                   ; addition de b avec x. b+x -> x
       inca
       cmpa   #$ff           ; z=0 si a-ff=0
       bne    suite          ; branche si z=0
       ldd    #$00ff
       xgdx                  ; échange d avec x. d <-> x
       idiv                  ; d/x résultat dans x et reste dans d
                             ; La somme des valeurs détectées sur le porte
                             ; divisée par le nombre de boucle
```

```
        xgdx                ; transfert le résultat x dans le d, s'est à dire dans b
        tba                 ; b -> a
        ldx     $1000       ; on remet la valeur original dans x
        rts

; sous programme de ralentissement qui suit la piste selon la première méthode
vitessemin
        ldab    porte
        cmpb    #$18
        beq     toutdroit   ; branche si #$18 = #$18 (z=1)
        cmpb    #$18
        bcc     motgauche   ; branche si < #$18 (c = 0)
        cmpb    #$18
        bcs     motdroit    ; branche si > #$18 (c = 1)
        cmpb    #$00
        beq     arret       ; branche si #$00 = #$00 (z=1)
        brset   portd,x %00001000 arret          ; bit7 (PD3) pour arrêt définitif
        brset   portd,x %00000100 vitessemin     ; bit 8 (PD2) pour le ralentissement
        rts

motgauche
        subb    #$18        ;porte - #$18
        ldaa    #$60        ; consigne de vitesse
        staa    porta
        sba                 ; retrait de la valeur de compensation
        coma                ; complémentation du port b car celui ci est à collecteur ouvert
        staa    portb
        rts

motdroit
        subb    #$18        ;porte - #$18
        ldaa    #$60        ; consigne de vitesse
        aba                 ; ajout de la valeur de compensation
        staa    porta
        sba                 ; retrait de la valeur de compensation
        coma                ; complémentation du port b car celui ci est à collecteur ouvert
        staa    portb
        rts

toutdroit
        ldaa    #$60
        staa    porta
        ldab    #$5a
        comb
        stab    portb
        rts
```

```
temponbms                ; sous programme de temporisation réglable à la milliseconde
        ldy     nbms
lbl1    ldx     #500
lbl2    dex
        bne     lbl2
        dey
        bne     lbl1
        ldx     #1000
        rts

arrettemp                ; sous programme d'arrêt temporisé pour supprimer l'inertie
        ldaa    #$00
        staa    porta
        coma
        staa    portb
        ldx     #500
        stx     nbms
        jsr     temponbms
        rts

droite90                 ; sous-programme pour tourner à droite
        ldaa    #$00
        staa    porta
        ldab    #$80
        comb
        stab    portb
        rts

gauche90                 ; sous-programme pour tourner à gauche
        ldaa    #$80
        staa    porta
        ldab    #$ff
        stab    portb
        rts

contour         ; sous-programme de contournement d'obstacle
        jsr     arrettemp       ; arrêt du robot pendant un temps court pour supprimer l'inertie
        jsr     droite90        ; tourne à droite
        ldx     #300            ; tempo de 300 ms
        stx     nbms
        jsr     temponbms
        jsr     arrettemp

        jsr     toutdroit       ; tout droit
        ldx     #2000           ; tempo de 2s
        stx     nbms
        jsr     temponbms
        jsr     arrettemp

        jsr     gauche90        ; tourne à gauche
```

```
ldx     #400            ; tempo de 400ms
stx     nbms
jsr     temponbms
jsr     arrettemp

jsr     toutdroit       ; tout droit
ldx     #5000           ; tempo de 5s
stx     nbms
jsr     temponbms
jsr     arrettemp

jsr     gauche90        ; tourne à gauche
ldx     #300            ; tempo de 300ms
stx     nbms
jsr     temponbms
jsr     arrettemp

jsr     toutdroitpiste  ; tout droit jusqu'à la piste
jsr     arrettemp

jsr     droite90        ; tourne à droite
ldx     #300
stx     nbms
jsr     temponbms
jsr     arrettemp
rts
end
```

Ce programme general.a11 est organisé contrairement au programme prfinal.a11 jouant le même rôle.

3. UTILISATION DU CONNECTEUR D

Brochage du connecteur D :

2	4	6	8	10
5 V	PG2	PD4	PD2	Tout
0 V	PG3	PD5	PD3	Rin
1	3	5	7	9

« tâbleau du document Controlboy page 5 »

Exemple :
brset portd,x %00000100 vitessemin ; bit 8 (PD2) pour le ralentissement

Teste les bits, spécifiés par 1 dans l'octet du masque, du port D ; puis effectue un branchement si tous les bits testés sont à 1.

Tableau qui définit l'octet du masque :

N° colonne	7	6	5	4	3	2	1	0
PD2	0	0	0	0	0	1	0	0
PD3	0	0	0	0	1	0	0	0
PD4	0	0	0	1	0	0	0	0
PD5	0	0	1	0	0	0	0	0

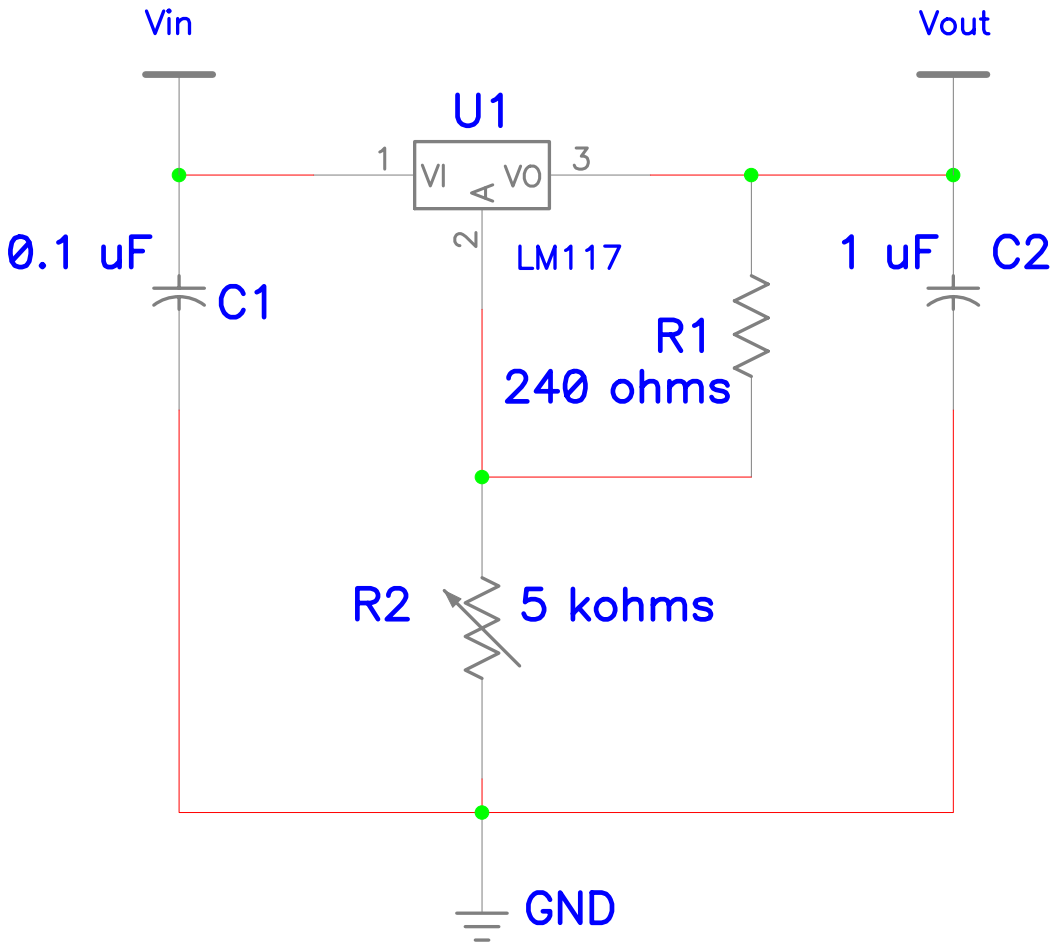
Pour tester PD2, il faut mettre un 1 à la colonne 2
Pour tester PD N, il faut mettre un 1 à la colonne N

4. PROBLEME POSE PAR L'ARRET AVEC REDEMARRAGE

Quand on reçoit un « 1 » de la priorité à droite, le robot ne s'arrête pas dans l'alignement de la piste et met un certain temps pour s'arrêter. Ceci est dû à l'inertie du robot. Pour y remédier, on propose d'utiliser une petite carte d'arrêt des moteurs qui permet de stopper l'alimentation des moteurs. Ceci contrôlé par le microcontrôleur. Cette carte doit avoir comme contrainte un bit d'arrêt et de réarmement.

5. ALIMENTATION DES LAMPES

La tension de la batterie varie au cours de son utilisation. Ce qui influe la luminosité des ampoules, plus la tension est élevée plus la luminosité sera importante et inversement, donc les valeurs détectées seront faussées. Avec ce montage, on aura une tension constante en sortie même si la tension de la batterie baisse (jusqu'à un certain seuil).
Ce montage est une alimentation stabilisée réglable, permettant un réglage précis de la tension.



6. CONCLUSION

Le projet qui devait être principale de la programmation, s'est transformé en teste de la carte de gestion d'obstacle sur la piste. Nous avons acquis un nouveau langage de programmation en assembleur. Nous avons apporté de nouvelles fonctions pour le concours robotique. Chaque année, le règlement du concours robotique évolue, il se doit de remettre à jour le robot. La carte Microcontrôleur apporte l'avantage d'être très flexible et de tester différentes méthodes de programmes afin de voir la réaction du robot.