Fok Frédéric
Van Wambeke Nicolas

INSA Toulouse
3RT – groupe B

Nicolas Van Wambeke

Digitally signed by Nicolas Van Wambeke
DN: CN = Nicolas Van Wambeke, SN = Van Wambeke, G = Nicolas
Reason: I am the co-author of this document
Location: Toulouse, FRANCE
Date: 2005.07.16 23:20:36 +01'00'

BE TELECOMMUNICATIONS

TRANSMISSIONS DE DONNEES NUMERIQUES PAR FAISCEAU LASER A L'AIR LIBRE

DEUXIEMME PARTIE

SOMMAIRE

I. INTRODUCTION

La première partie de notre compte rendu de Bureau d'étude s'achève sur l'élaboration du démodulateur dans la chaîne de réception. Nous avons depuis, avancé dans la réalisation du projet et afin d'arriver à avoir une chaîne d'émission totalement fonctionnelle et une chaîne de réception presque entièrement terminée.

Nous verrons dans cette partie du compte rendu le détail des étapes d'élaboration de la chaîne d'émission que nous n'avions pas encore commencé ainsi que les éléments nécessaires à la complétion de notre chaîne de réception. C'est-à-dire la mise en place d'un filtre passe bas afin de terminer la démodulation, d'un trigger de Schmidt afin de régénérer le signal en sortie du démodulateur ainsi que la mise en place d'un convertisseur courant tension permettant d'attaquer le montage récepteur en tension bien que la diode de réception délivre elle un courant variable.

II. Mise en place de la chaîne d'émission

A. Présentation

La chaîne d'émission constitue l'ensemble des éléments qui seront placés dans les boîtiers avec lesquels les joueurs pourront 'tirer' l'un sur l'autre. L'identité du joueur qui tire est véhiculée par une modulation de l'intensité lumineuse émise par le joueur.

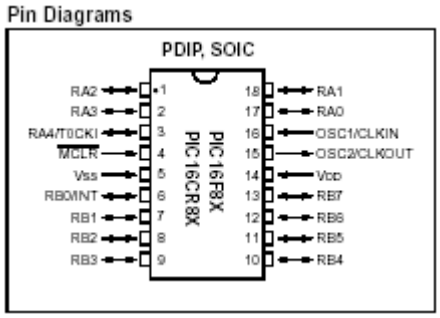
Afin de générer le signal de commande de la diode, nous avons utilisé un microcontrôleur commercialisé par la compagnie MicroChip. Il s'agit du PIC 16F84A. Ce microcontrôleur présente l'avantage d'avoir un jeu d'instruction assez simple tout en permettant de faire beaucoup de choses. Bien que programmable en C, nous l'avons programmé en Assembleur directement afin de garder une maîtrise totale sur le déroulement des différentes instructions. En effet, la génération d'un signal de commande demandait ici que nous puissions avoir une bonne maîtrise du temps.

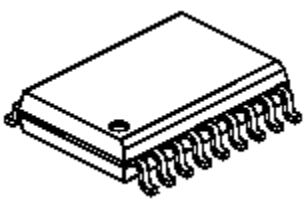
B. Programmation du PIC 16F84A

1. Préambule:

Nous avons d'abord pris connaissance au cours d'une demi-scéance du langage et de ces spécificités. Puis nous avons commencé à manipuler le langage avec un petit programme de boucle infinie.

Pour la programmation du pic, il nous a donc été nécessaire de connaître sa structure interne à savoir les registres, mémoires, ports d'entrées sorties et leurs possibilités. L'accès à la mémoire et aux ports nous d'abord été un peu flou mais l'expérimentation nous a rapidement permis d'identifier les façons correctes de procéder. Nous avons utilisé le logiciel MPLAB IDE comme interface de développement pour le codage de nos programmes. Nous avons notamment apprécié la fonction de simulation qui nous a permis de bien cerner les erreurs avant même de tester sur la cible.





Le PIC que nous avons utilisé possède un jeu d'instruction réduit de 35 instructions. On s'est renseigné et il en existe en fait 37 mais les instructions OPTION et TRIS sont déconseillés par le constructeur car dans la série 16 il existe d'autres microcontrôleurs qui n'utilisent pas ces deux instructions. Pour notre utilisation en TP, cela n'a pas d'influence.

2. Premier Programme : Temporisation

Afin de pouvoir compter le temps qui s'écoule et de se donner la possibilité de 'perdre du temps' dans nos programmes, il était nécessaire de créer un programme qui prenait comme argument d'entrée une valeur et qui rendait la main au programme appelant après un temps déterminé par la valeur passée en paramètres.

Le listing de ce programme est présent en annexe 1. Le passage de paramètres se fait par l'intermédiaire du registre W, lorsque l'on appelle le programme de temporisation, celui-ci lit la valeur présente dans le registre W et effectue une attente passive en décrémentant un compteur dont la valeur est calculée pour que le temps attendu soit exactement égal à la valeur contenue dans W lors de l'appel.

Nous allons détailler, étape par étape la réalisation de ce programme.

```
;-----------------------------------------
;-- déclaration pour l'initialisation
;-----------------------------------------
      processor   16f84a            ; obligatoire!
      radix       dec               ; base de numération par défaut
      include     "regs84.inc"      ; inclusion de déclaration utile
```

Tout d'abord, il est nécessaire de spécifier un certain nombre de directives d'assemblage. On spécifie quel processeur on utilise (:processor 16f84a), le codage par défaut utilisée (radix dec) et include «regs84.inc» permettant de donner des noms aux adresses des registres... (ex: status), ce qui est bien plus pratique pour le codage!!

Puis, nous commençons le codage du corps du programme par la déclaration des espaces mémoires réservés aux variables.

```
;-----------------------------------------
;-- constante et variable diverses ... ------
;-----------------------------------------

R     equ 0x0C    ; adresse de la variable
Vinit equ D'80'   ; valeur à 10
```

Remarque : Dans le listing fourni en Annexe 1, nous utilisons l'instruction :

```
movlw       Vinit
```

où Vinit est une constante de valeur 80. Cette instruction ne sera pas présente dans notre procedure finale de temporisation appelable par un programme quelconque, en effet, le placement de la valeur de l'attente dans le registre W sera a la charge du programme appelant.

Pour réaliser une attente, il nous a fallu tenir compte des temps mis par le micro processeur pour exécuter une instruction. Nous avons vu que certaines instructions prenaient un temps qui était conditionné par le résultat d'un test ce qui a rendu le codage plus compliqué. De plus, la boucle minimale d'attente avec décrémentation d'un compteur dure trois instructions. Ceci n'est pas commode à manipuler et nous avons préféré une boucle durant quatre cycles d'horloge. En effet, le nombre de tours à effectuer dans la boucle est donc, à quelques instructions près, égal au résultat de la division par 4 de la valeur contenue dans W. Une division par quatre correspond à deux décalages à droite ce qui rend le codage très aisé.

Notre boucle est donc la suivante (durée 4 instructions) :

```
boucle
      nop
      decfsz      R, 1
      goto  boucle
```

En plus d'effectuer la division de la valeur contenue dans W par 4, il faut tenir compte de la présence d'une retenue lors de cette division. Comme notre division par 4 est en fait deux divisions successives par 2, il faut deux fois tenir compte de la présence où non d'une retenue.
Si après la première division il y a présence d'une retenue, il faudrait théoriquement perdre un temps d'horloge et aucun sinon. Rien que le fait de tester la présence d'une retenue prends déjà un cycle d'horloge, il est donc impossible de ne perdre qu'un temps lors de la présence d'une retenue et aucun sinon. Nous avons donc décidé de procéder de la façon suivante :

```
      btfsc status, 0   ; Si on a pas eu de retenue (nombre paire)
      goto  boom        ; On perds 2 sinon on perds 3 temps


boom  bcf   status,0    ; On remets le bit de Carry du STATUS à 0
```

Si il n'y a pas de retenue, on passe directement à l'instruction d'étiquette boom, sinon, on effectue un goto (2 cycles) vers l'instruction d'étiquette boom. On arrive ainsi a avoir deux temps si le nombre est par et 3 s'il est impair.

La seconde division est traitée de façon analogue mais avec des temps d'attente résultants deux fois supérieurs et donc tous pairs car la présence d'une retenue lors de la seconde division n'a plus rien à avoir avec le fait que le nombre de départ soit pair ou impair.

Ensuite, une série d'instructions de décrémentation permettent de synchroniser le programme afin que la valeur résultat corresponde au nombre de tours à effectuer dans la boucle pour que l'attente totale soit correctement temporisée.

Nous avons par la suite utilisé notre programme de temporisation, notamment pour l'attente au niveau bas lors de l'émission du code et nous avons pu constater qu'il fonctionnait parfaitement.

3. Programme d'émission en bande de base

Après avoir programmé le programme de temporisation, nous avons attaqué la programmation des broches de sortie du PIC. Nous avons procédé dans un premier temps à la réalisation d'une suite infinie de "zéro" et de "un". Nous avons donc utilisé notre programme de temporisation précédent pour permettre de garder soit le niveau haut soit le niveau bas pendant une durée déterminée à l'avance et qui est donnée comme paramètre à notre temporisateur.

Le listing du programme sinus est disponible en Annexe 2.

Voici la boucle infinie qui réalise une suite de « zéro » et de « un »:

```
      main
            call confentr
      sinus
            call un
            call zero
            goto sinus
```

Remarque : pour coder les "zéro" et "un", il est important de configurer la broche du PIC sur laquelle nous allons délivrer les bits en sortie.

```
confentr
            bsf   STATUS, RP0 ; passage en bank1 pour la config
            bcf   TRISA, 1    ; bit configuré en sortie
            bcf   STATUS, RP0 ; repassage en bank0
            return
```

Le "un" est codé comme suit :

```
un
      bsf   PORTA, 1    ; mise en place d'un 1 en sortie
      movlw 166
      call  temporis
      bcf   PORTA, 1
      movlw 79
      call  temporis
      return
```

Le "un" qui est codé ici correspond à notre un de codage à savoir que le "un" est définie comme étant un signal qui dure 167 microseconde au niveau haut et 83 microseconde au niveau bas, soit une durée de 250 microsecondes.

Suite à la programmation de ce programme, nous avons enregistré le code dans la mémoire du PIC grâce au logiciel ICProg.

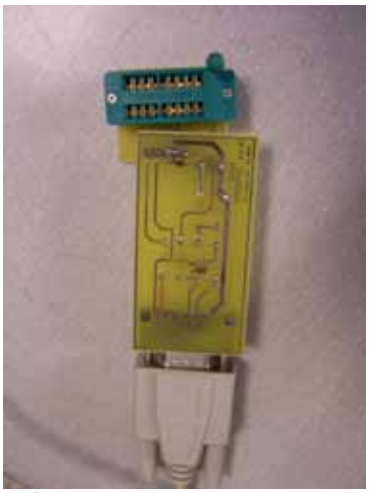


Fig. 1 : Programmateur

Puis nous avons testé à l'oscilloscope le signal que rendait le PIC à la sortie de la patte que nous avions configurée. Nous avons pu constaté que avions bien le signal attendu et nous avons vérifié les temps au niveau haut et bas du "1".

4. Programme d'émission du signal modulé

Nous avons amélioré notre programme d'émission dans le but d'obtenir une modulation avec porteuse à 100Khz et de coder notre code joueur. Nous avons choisi, en accord avec les autres binômes, le code suivant: "0101"

Le listing complet du programme d'émission modulée est donné en Annexe 3

Ce codage est réalisé comme suit :

```
code        ; programme principal
      call zero
      call un
      call zero
      call un
      call un
      call zero
      call un
      call zero
      call pause
      goto code
```

On remarque la présence, en plus de l'appel des programmes zéros et un, un appel à un programme nommé pause. Il s'agit en fait ici du programme permettant d'attendre deux millisecondes entre chaque envoi du code.

A des fins de simplifications et pour pouvoir utiliser le PIC afin de réaliser la modulation, nous allons coder des alternances uniquement positives pour le signal modulé, en effet, il est impossible de placer en sortie du PIC une tension négative.

Le code du programme d'émission d'un zéro est le suivant :
(Vzero correspond au nombre de périodes de la porteuse pendant le niveau haut, ici 8).

```
zero
      movlw       Vzero
      movwf T

Boucle0     bsf   PORTA, 1    ; mise en place d'un 1 en sortie
            nop
            nop
            nop
            nop
            bcf   PORTA, 1
            nop
            decfsz      T,1
            goto  Boucle0

            bsf   PORTA, 1    ; -|
            movlw       162   ;  |- permet d'attendre 167 µs
                              ;  |  au niveau bas
            call  temporis    ;  |
            return            ; -|
```

On remarque que Boucle0 produit en fait un signal à variation rapide de période 10µs qui correspond à la porteuse à 100 Khz.

Comme auparavant, on a ensuite testé notre code en entrant les données dans le PIC. Nous avons alors regardé à l'oscilloscope notre signal. Nous avons pu vérifier que notre programme était correct en visualisant si les contraintes temporelles étaient respectées.

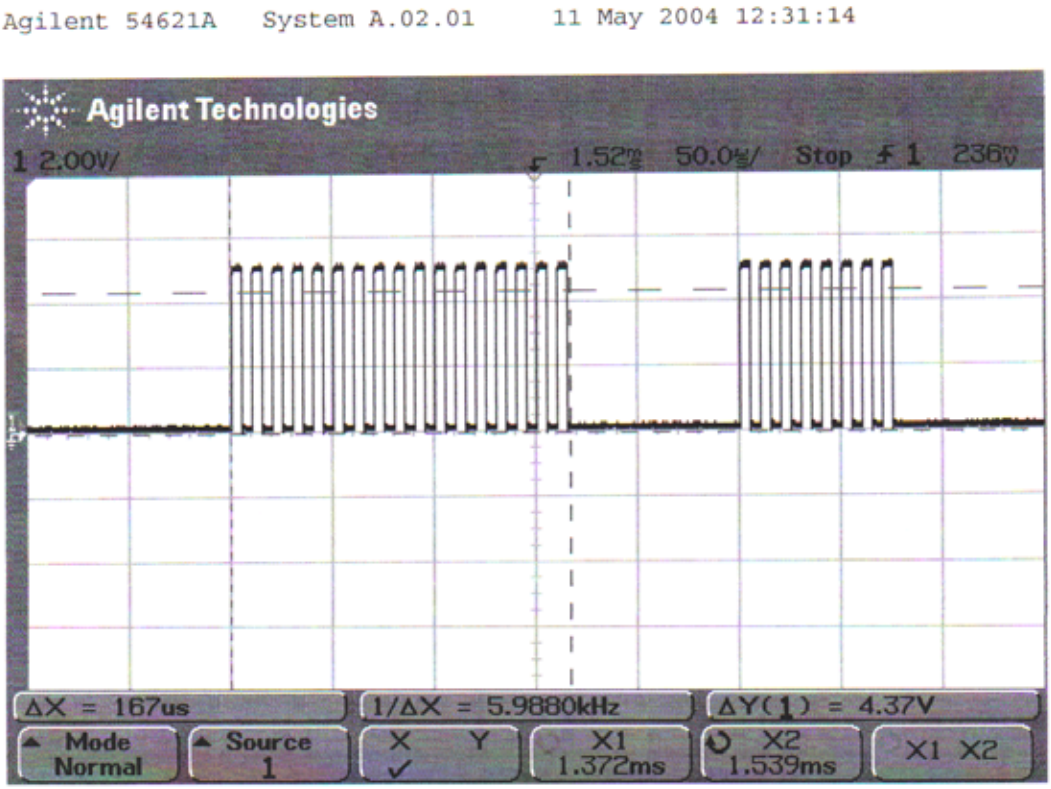


Fig. 2 : 167µs au niveau haut modulé pour un « un ».

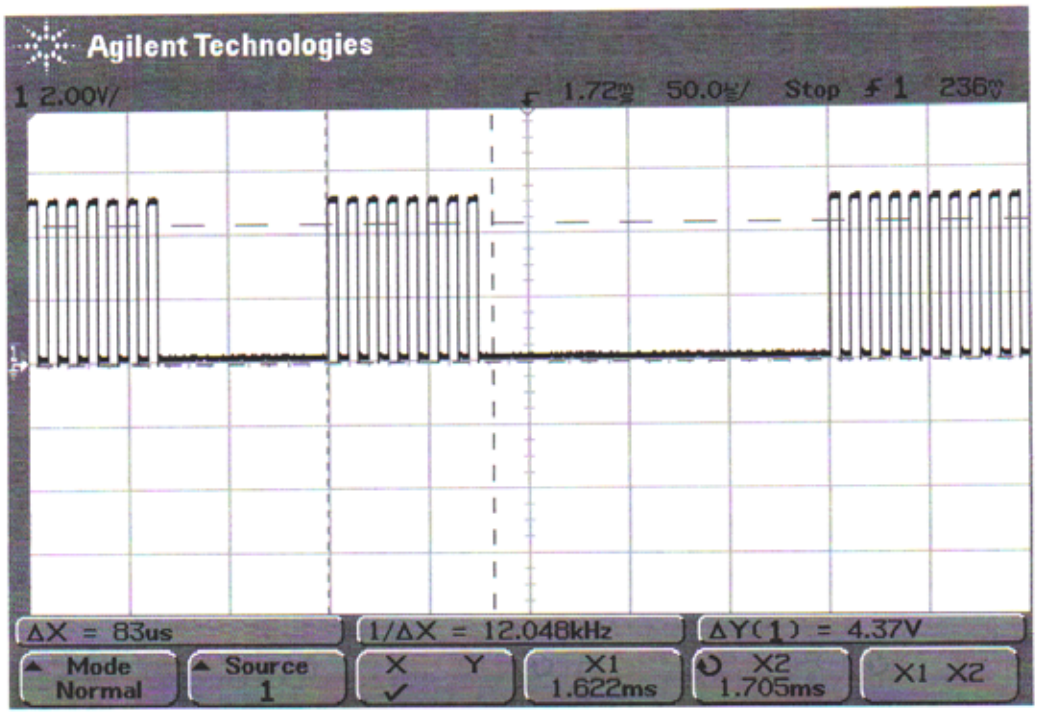
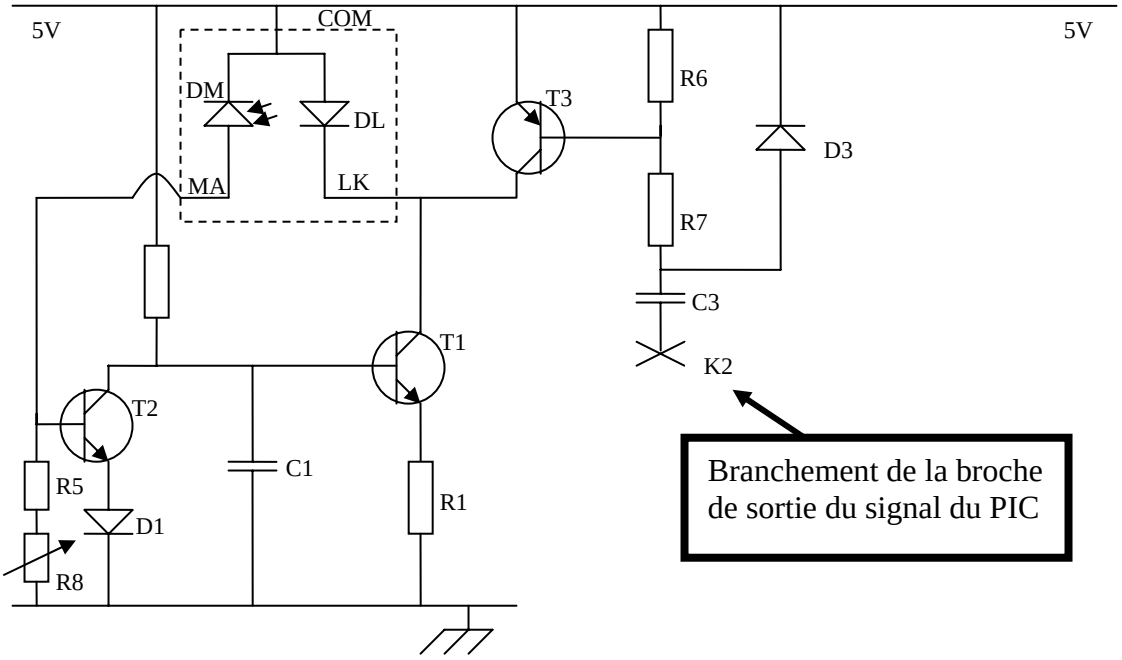


Fig. 3 : 83µs au niveau haut pour un « zéro ».

C. Le circuit d'émission

1. Schéma et Principe



Etape 1 :

- Si K2 = 5V, alors T3 est bloqué ⟨ La diode s'allume.
- Si K2 = 0V, alors T3 est passant ⟨ Aucun courant ne circule dans la diode. Mais, C3 se charge et T3 se bloque puis la diode s'allume.

Etape 2 :

On considère que T3 est bloqué (en émission). Lorsque la diode émet, un courant I circule dans le circuit pour son fonctionnement.

Rôle de C1 :

Retard des transitions et diminution des oscillations du à la régulation du circuit. C1 va donc ralentir les oscillations.

Rôle de D1 :

Limiter le bruit ⟨ il faut 1.2V pour que T2 ne soit pas bloqué. On pare donc le bruit.

Rôle de D3 :

Permet d'avoir des transitions plus franche sur la commande et d'évacuer les charges de C3.

Réglage de la puissance d'émission :

A l'équilibre (émission), on vise sur la cible et on mesure la puissance. On agit alors sur R8 pour avoir 1 mW. Ce réglage se fait expérimentalement en mesurant la puissance et on réglant le potentiomètre afin d'obtenir la puissance demandée.

2. Réalisation

Lors de la séance de réalisation du circuit d'émission, nous avons eu l'occasion d'apprendre ou de parfaire notre compétence à souder. Une erreur glissée dans le schéma du montage concernant une valeur de résistance nous a forcé à manier les différents moyens permettant de dessouder (utilisation d'une tresse à dessouder).

Le circuit imprimé qui nous a été fourni nous a permit de voir comment est réalisée concrètement l'optimisation de l'espace sur ce genre de réalisation. Afin d'avoir une idée des outils utilisés, nous avons essayé de trouver une version d'évaluation du logiciel TARGET mentionné dans le cahier de Bureau d'Etude, malheureusement, une telle version ne semble pas être distribuée par internet. Nous avons néanmoins pu trouver des équivalents, notamment le logiciel PCB (http://pcb.sf.net) utilisable sous linux mais à notre avis compliqué pour une première utilisation de ce genre de logiciels. Nous avons aussi trouvé un logiciel nommé ExpressPCB qui lui nous a semblé plus simple d'utilisation car il possède une interface graphique conviviale.

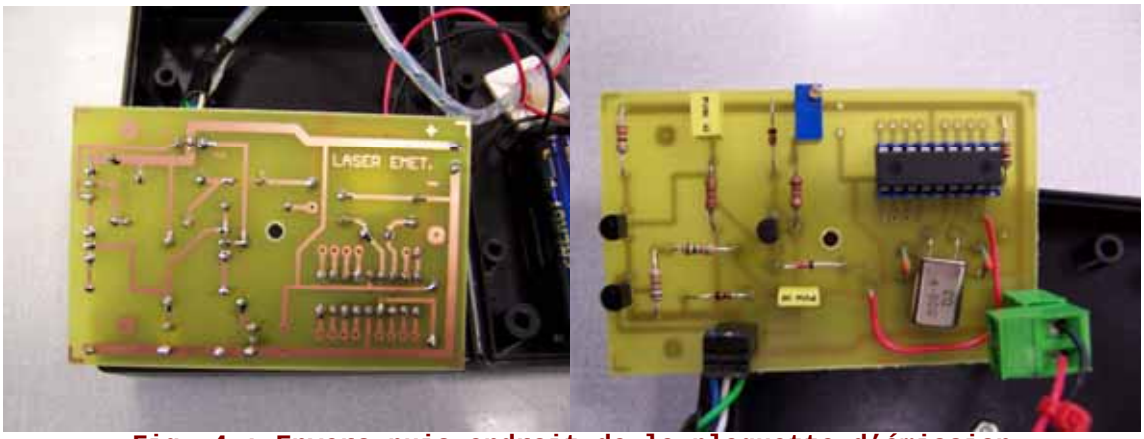


Fig. 4 : Envers puis endroit de la plaquette d'émission

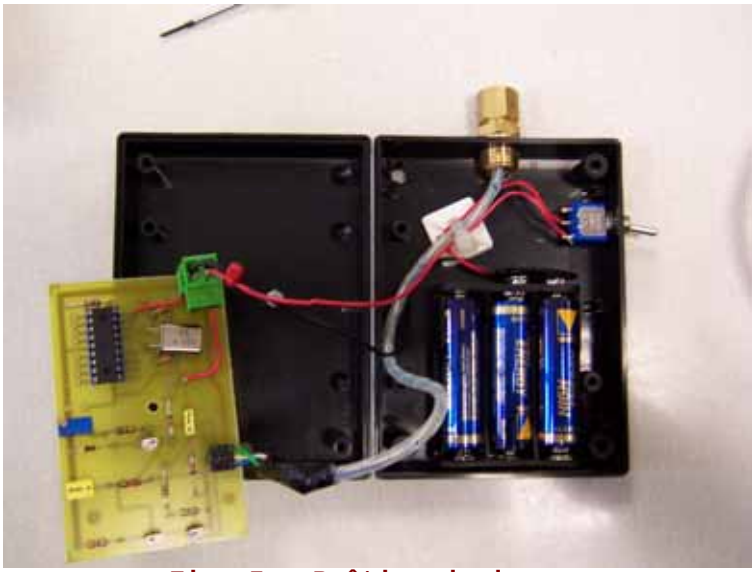
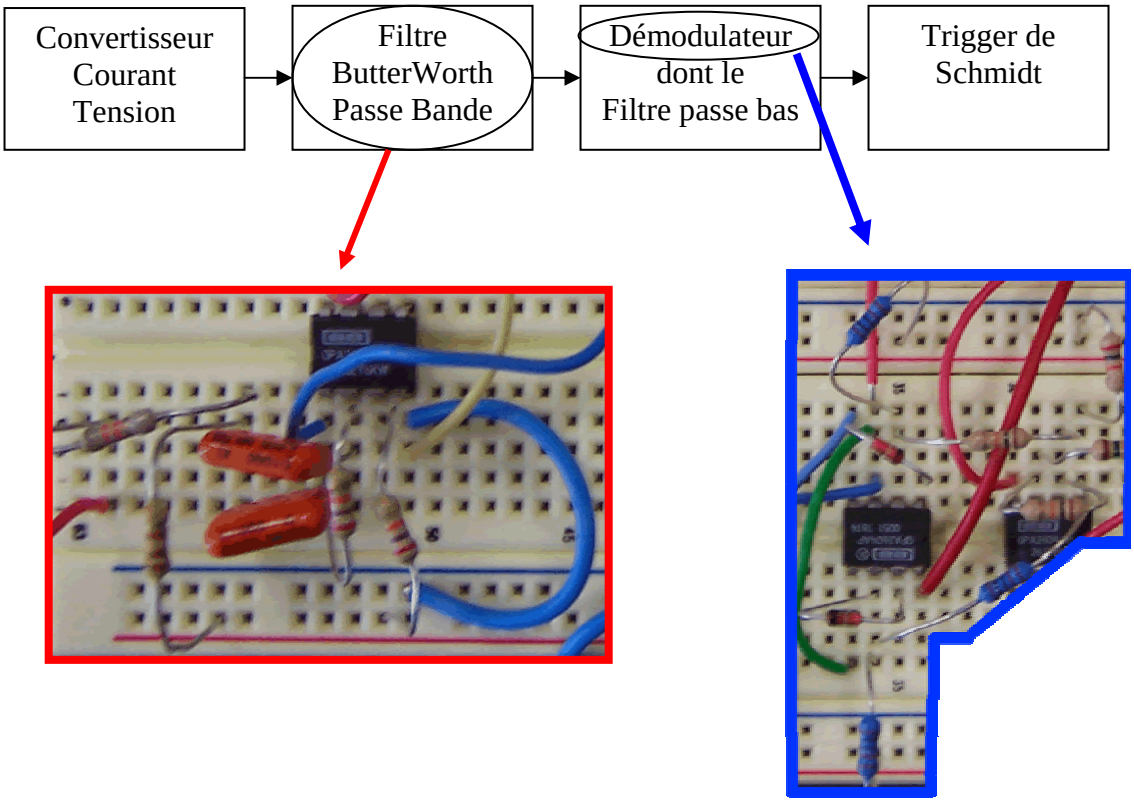


Fig. 5 : Boîtier du joueur

III. Poursuite de la chaîne de réception

Lors de la fin de la première partie du Bureau d'Etudes, nous avions réussi à compléter une bonne partie de la chaîne de traitement du signal en réception. Néanmoins, il nous restait cependant quelques éléments à concevoir comme l'illustre la figure récapitulative ci-dessous.



A. Conception du filtre RC passe bas

Une fois le signal redressé par la première partie de l'étage de démodulation, il est nécessaire de le filtrer afin de ne garder que l'enveloppe du signal qui correspond à notre signal informatif.

Nous avons donc mis en place un filtre RC (filtre passe bas) en sortie du démodulateur afin de supprimer les hautes fréquences et de pouvoir récupérer l'enveloppe de notre signal.

Le choix de la fréquence de coupure du filtre RC a été choisie aléatoirement, évidement bien inférieure à la fréquence de la porteuse (100Khz) mais pas trop faible de façon à

éviter d'obtenir une amplitude trop faible pour notre signal en sortie du filtre. Le filtre RC étant un filtre passif il n'est pas possible d'avoir une atténuation nulle. Aussi, la fréquence du signal informatif était suffisamment petite par rapport à celle de la porteuse pour qu'un filtre du premier ordre soit suffisant.

Nous avons donc choisi un condensateur de 10nF et une résistance de 1.5kΩ

$$f_c = \frac{1}{2\pi RC} = 10{,}61\ khz$$

Le signal obtenu après filtrage avait bien la forme du signal informatif que nous cherchions à retrouver, néanmoins, il ne nous était pas possible de le traiter avec un microcontrôleur car le niveau qu'il présentait était trop bas. Nous avons donc mis en place un trigger de Schmidt dans le but de régénérer notre signal informatif logique.

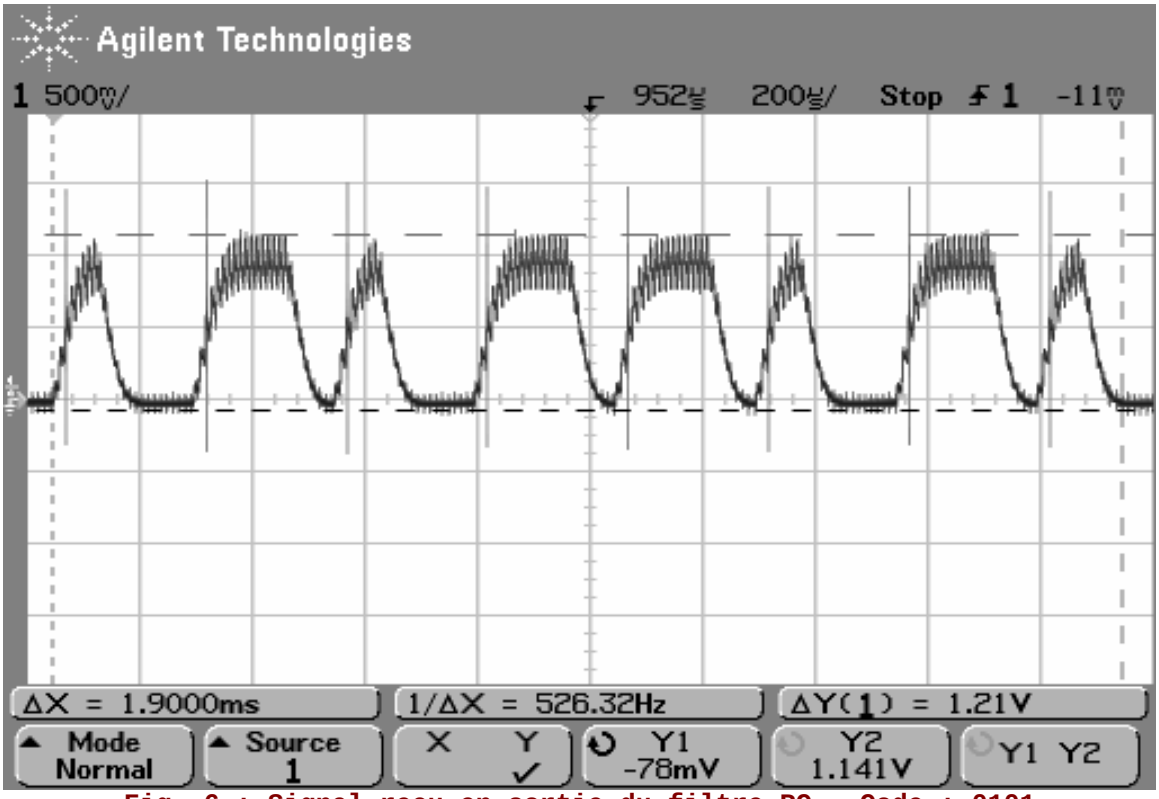
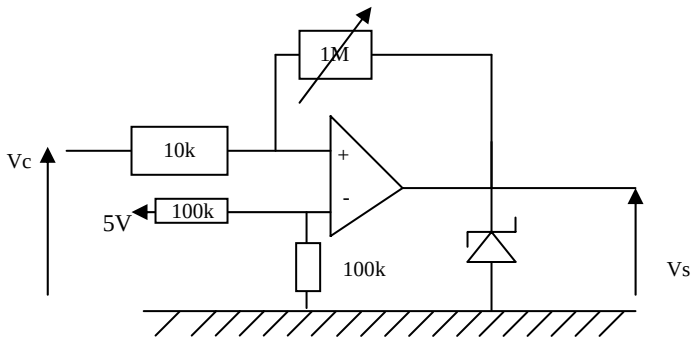


Fig. 6 : Signal reçu en sortie du filtre RC – Code : 0101

B. Mise en place du trigger de Schmidt

Après la mise en place de notre filtre RC, nous avons mis en place un trigger qui nous a permis de relever notre signal entre 0 et 5V. Ceci afin de pouvoir le fournir en entrée d'un microcontrôleur PIC afin de pouvoir effectuer le décodage de l'information contenue dans le signal reçu

1. Schéma et Principe

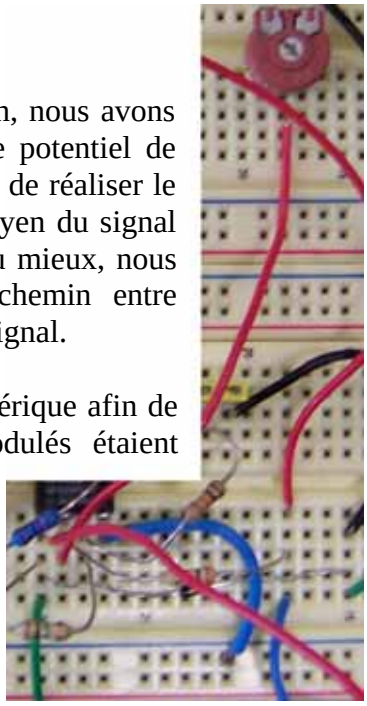


L'amplificateur étant bouclé sur l'entrée positive, le montage ici est instable en régime linéaire pour l'amplificateur. La sortie ne peut prendre que deux états correspondant aux deux seuils de saturation de l'amplificateur. La diode Zener placée en sortie permet de fixer les potentiels de sortie. En effet, lorsque l'amplificateur sera saturé en état haut, la diode zener imposera un potentiel de 5.1V pour V_s ce qui correspond au seuil de la diode. En revanche, lorsque l'amplificateur sera saturé en état haut, la diode imposera un potentiel égal à sa tension de seuil en inverse qui est de l'ordre de 0.7V
La sortie ne peut donc prendre que deux états, le « un » logique ou le « zéro » logique.

La tension en sortie dépend de la valeur de la tension différentielle. En effet, l'amplificateur est ici monté en montage amplificateur de différence. Le potentiel de l'entrée négative est ici fixé, on va donc comparer le potentiel de l'entrée positive à celui de l'entrée moins, une sortie au niveau haut signifiera que le potentiel de l'entrée positive est supérieur à celui fixé sur l'entrée négative et vice-versa.

Le seuil pour lequel une tension fournie en entrée sera interprétée comme étant de niveau haut ou de niveau bas est déterminé par réglage du potentiomètre de 1MΩ.
En pratique, nous avons effectué le réglage de façon expérimentale de façon à obtenir un signal redressé permettant de distinguer les différents symboles. (Voir Fig. 7)

2. Réalisation



Pour la mise en place de cet étage de la chaîne de réception, nous avons utilisé une source de tension afin de maintenir constant le potentiel de l'entrée négative de l'amplificateur opérationnel. Aussi, afin de réaliser le réglage du seuil, nous avons du tenir compte du niveau moyen du signal sortant du filtre RC. Afin de régler la résistance variable au mieux, nous avons essayé de trouver un réglage « moyen » à mi chemin entre l'atténuation complète du signal et l'amplification totale du signal.

Nous avons ensuite réalisé des relevés à l'oscilloscope numérique afin de vérifier que différents symboles, « un » et « zéro » modulés étaient facilement distinguable à l'œil nu et que les temps à l'état haut ainsi que à l'état bas correspondaient au mieux avec ceux que nous nous étions fixé lors du choix du codage. Tout ceci afin de faciliter ensuite le décodage du code transmis par le PIC.

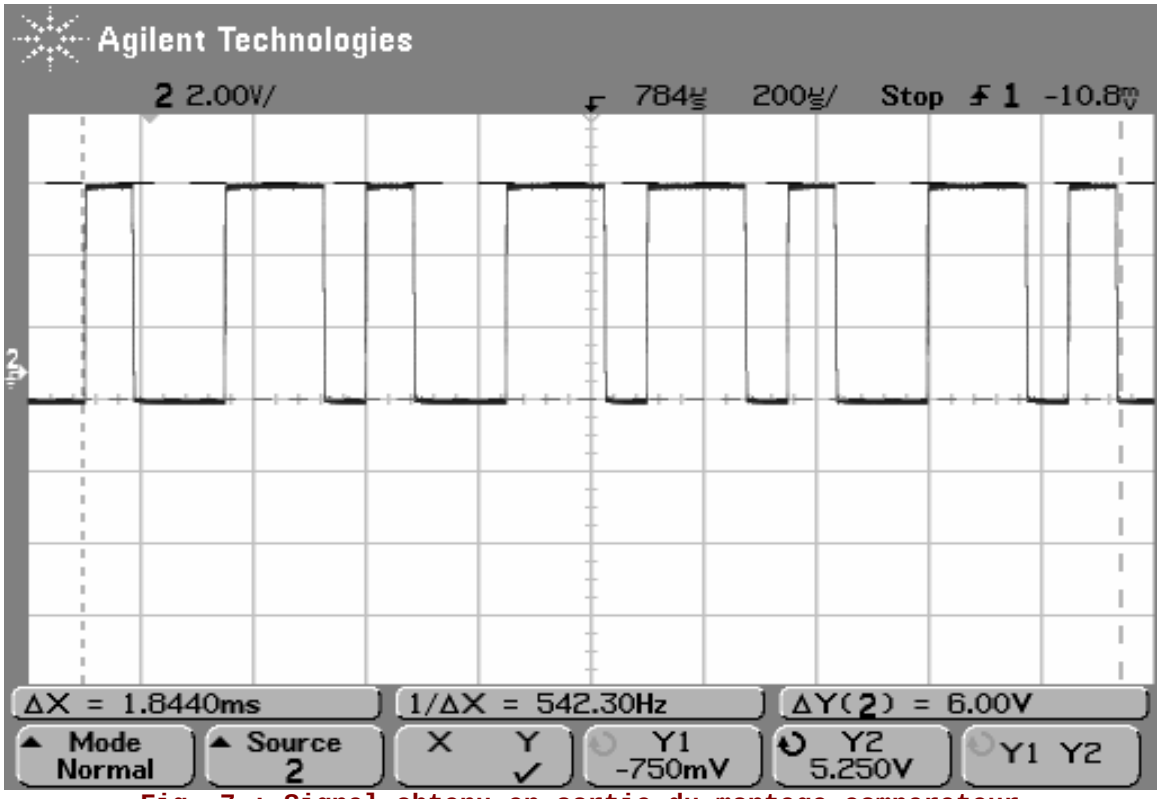


Fig. 7 : Signal obtenu en sortie du montage comparateur

On remarque sur la figure 7 que le code transmis est facilement lisible à l'œil nu, en effet, notre code joueur étant 0101 la trame est composée des huit bits : 01011010

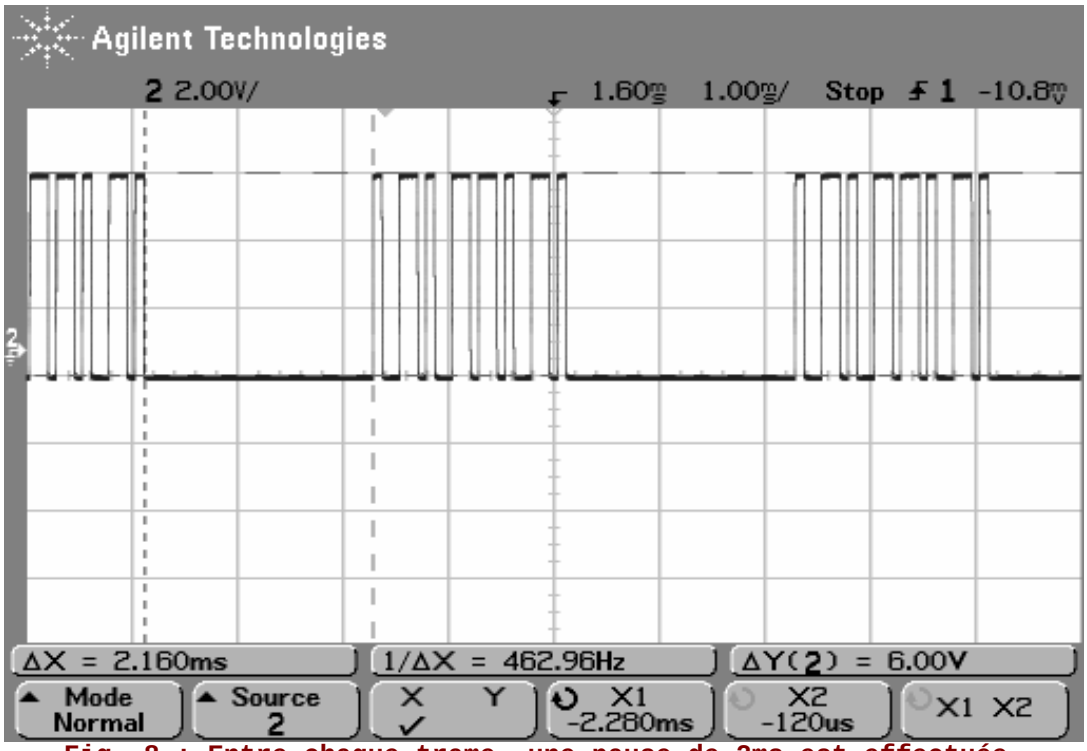


Fig. 8 : Entre chaque trame, une pause de 2ms est effectuée.

La figure 8 ci-dessus montre une succession de trois trames reçues par notre circuit, le code étant 0101, les trames sont donc 01011010 et terminent par l'envoi d'un zéro et donc de 167µs au niveau bas, c'est pourquoi sur le relevé de l'oscilloscope, il apparaît que le temps au niveau bas entre chaque trame a pour durée 2.16ms, cela confirme que la pause dure bien 2ms.

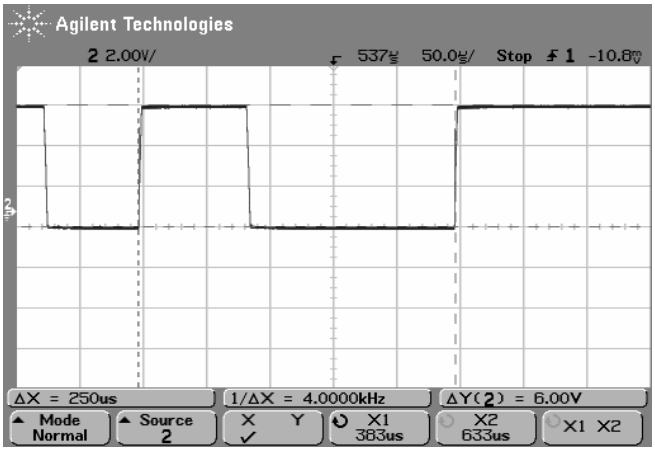


Fig. 9 : Période d'un « zéro »

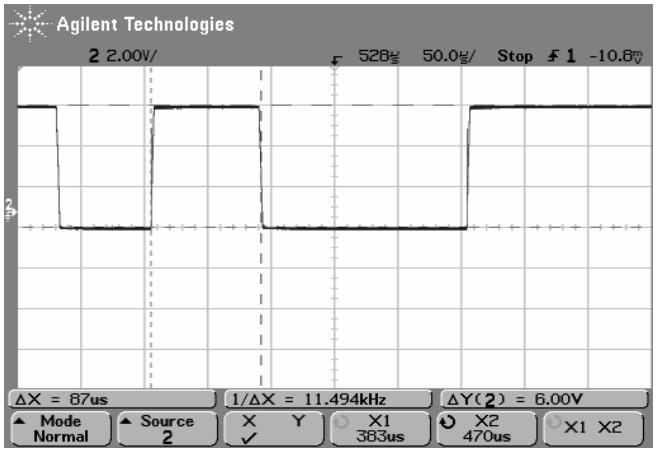


Fig. 10 : Durée niveau haut d'un « zéro »

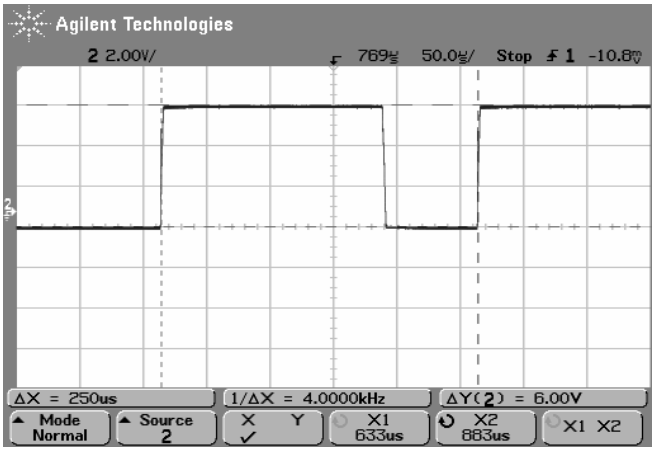


Fig. 11 : Période d'un « un »

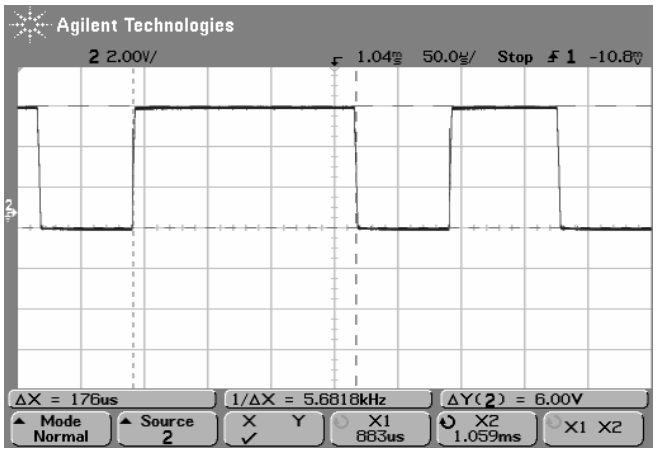
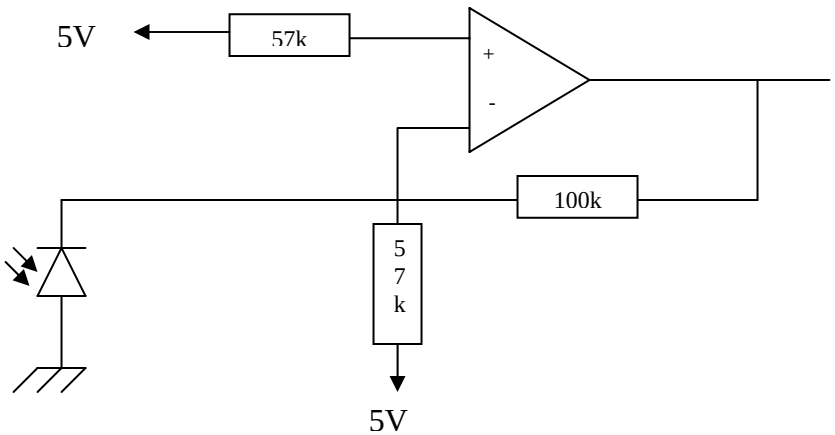


Fig. 12 : Durée niveau haut d'un « un »

C. Convertisseur Courant Tension

La diode que nous utilisons en réception délivre un courant variable en fonction de l'intensité lumineuse qu'elle capte. Notre chaîne de réception est prévue de façon à être attaquée en tension. L'information contenue dans les variations de l'intensité du courant délivrée par la diode ne peut pas directement piloter le circuit de réception. Il faut donc réaliser un circuit de conversion de ces variations de courant en variations de tension afin de pouvoir traiter le signal reçu. Ce circuit sera ensuite relié à l'entrée de notre filtre de ButterWorth.

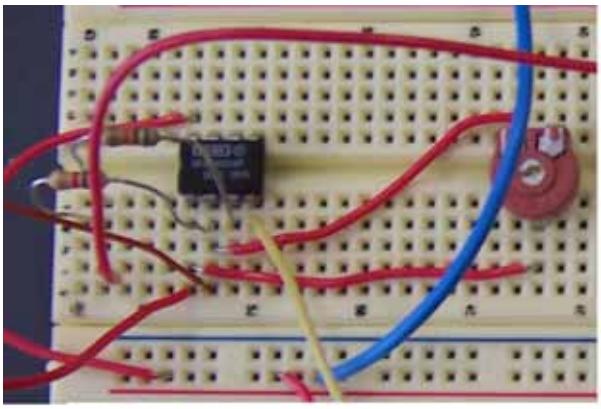
1. Schéma et Principe



Il est ici nécessaire d'élever les potentiels du convertisseur afin de pouvoir polariser correctement la diode, en effet, comme la diode doit délivrer un certain courrant, il faut

qu'elle soit dans une configuration qui rende cela possible. La tension de sortie du montage ici est une fonction des différentes résistances et du courrant d'entrée délivré par la diode.

2. Réalisation



Afin de pouvoir régler la valeur de la tension de sortie, nous avons décidé d'utiliser un potentiomètre de 100kΩ. Comme nous avions déjà utilisé une source de tension à 5V, il a été simple d'élever les potentiels afin de bien polariser la diode, Aussi, lors du branchement de la diode, il faut respecter la polarité.

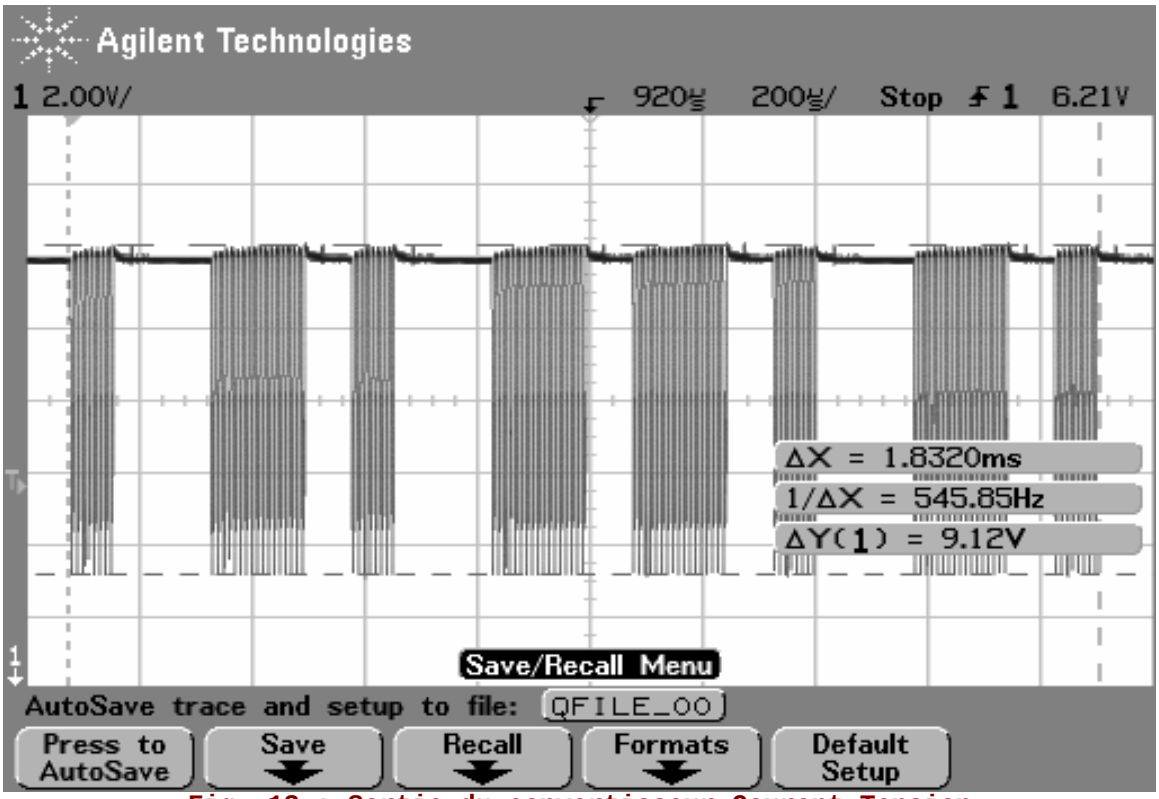


Fig. 13 : Sortie du convertisseur Courant Tension

La figure 13 ci-dessus montre la sortie obtenue lorsque l'on pointe notre émetteur sur la diode. On observe bien ici la réception de notre signal et le code est déjà visible.

D. Détails supplémentaires sur les étapes déjà réalisées

Nous avons au cours de cette deuxième moitié du Bureau d'Etude réalisé un test complet des étages réalisés dans la première partie du projet. En effet, le fait que nous disposions d'un émetteur fonctionnel et d'une chaîne de réception complète nous a permis de vérifier le bon fonctionnement de chacun de nos blocs séparément.

1. Précisions concernant le filtre de ButterWorth

Nous avons pu tester notre filtre dans des conditions de travail avec notre signal sortant du convertisseur courant tension, la figure 14 ci-dessous représente la sortie filtrée d'une trame d'information. Le fait d'avoir bien calibré notre filtre lors de sa conception nous permet d'avoir une atténuation minimale pour le signal informatif tout en conservant un bon gain dans la bande passante et donc en ne perdant pas d'information utile.

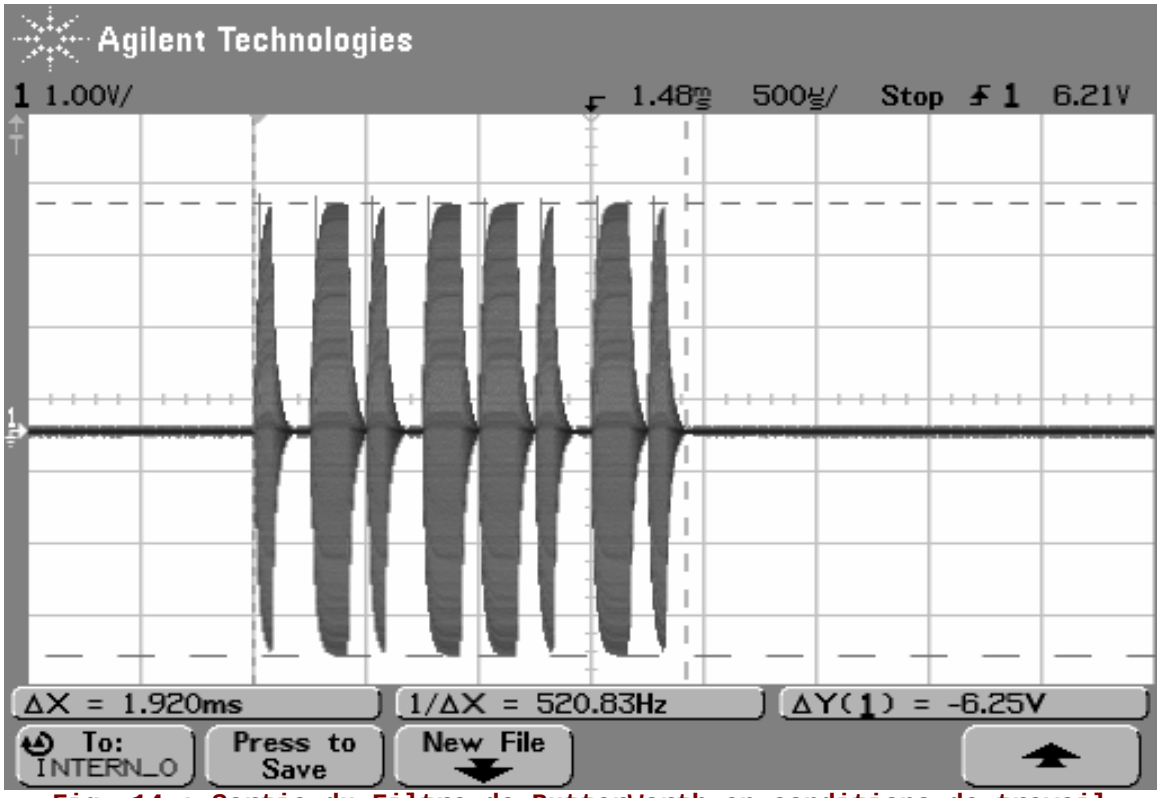


Fig. 14 : Sortie du Filtre de ButterWorth en conditions de travail

2. Précisions sur le redresseur.

Nous avons aussi pu vérifier le bon fonctionnement de notre redresseu en situation de réception d'un signal réel transmis par notre émeteur. Une fois de plus, les résultats obtenus sont satisfaisants.

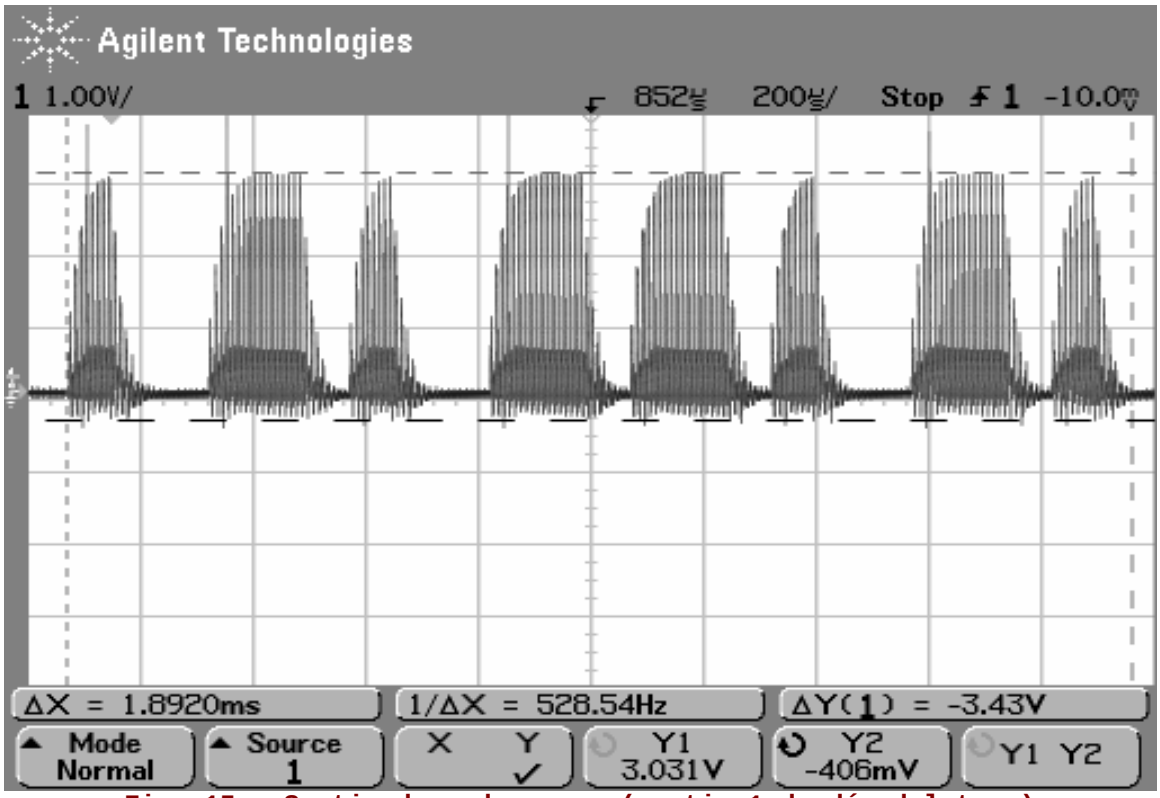


Fig. 15 : Sortie du redresseur (partie 1 du démodulateur)

E. Programmation du PIC en réception

La dernière étape de notre chaîne de réception consiste en un microcontrôleur PIC destiné à reconstituer à partir du signal en sortie du trigger de Schmidt le code du joueur qui a tiré sur la photodiode.

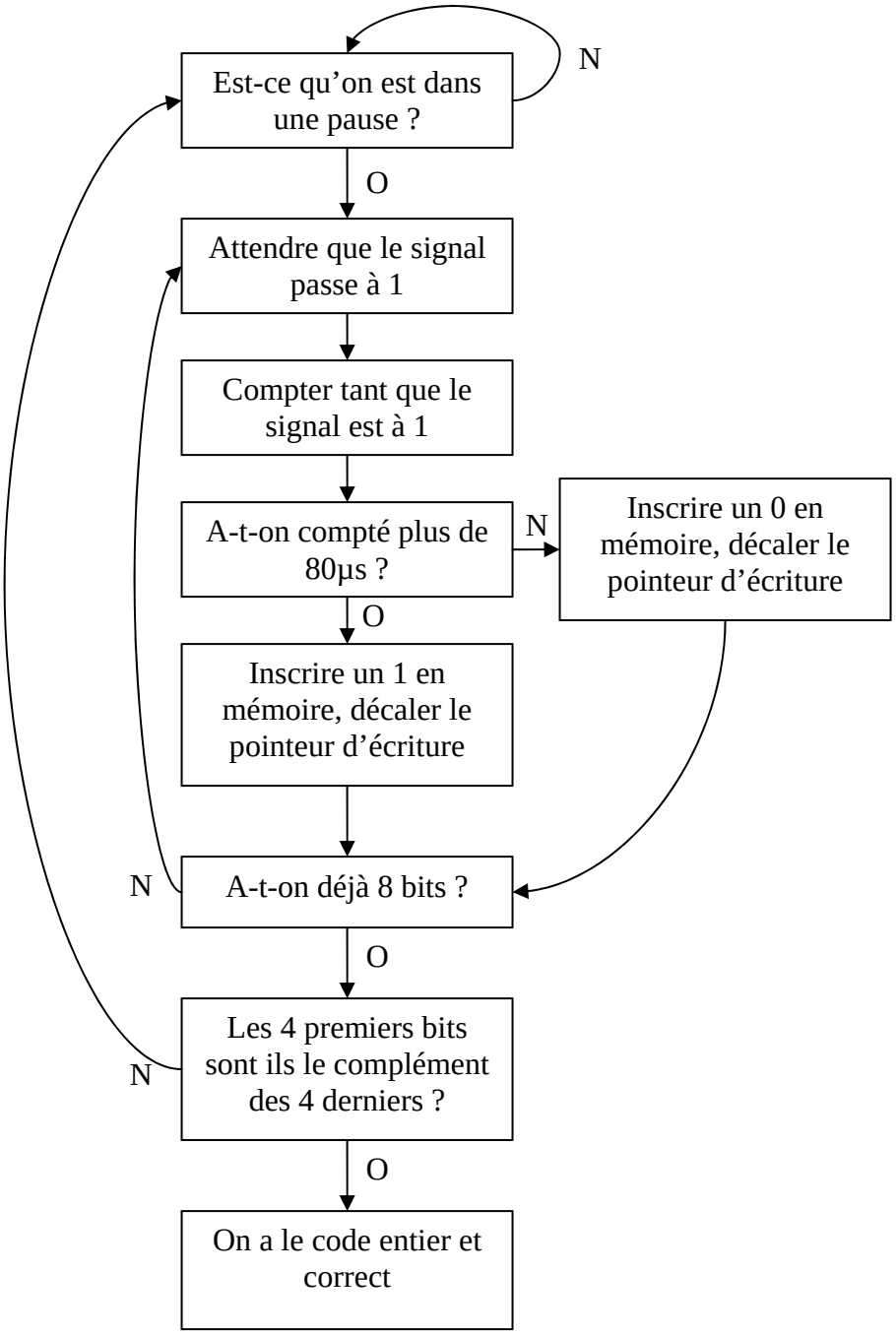
Afin de programmer ce PIC, et compte tenu du peu de temps qu'il nous restait pour réaliser ce programme, nous avons décidé de nous répartir le travail entre deux binômes.

1. Algorithme général

Le programme de réception doit pouvoir décoder la trame qu'il reçoit en entrée. Pour cela, il faut que le PIC puisse distinguer les différents symboles qu'il reçoit. Il parait donc

évident que plusieurs sous programmes peuvent être conçus afin de mieux segmenter le travail.

L'algorithme complet à mettre en œuvre est le suivant :



2. Algorithme d'attente du début du signal

Nous avons tenté d'écrire l'algorithme de détection des pauses dans un premier temps. Le test en simulation grâce à MPLAB de notre programme permettait de bien détecter périodes pendant lesquelles le signal était à zéro pendant au minimum 250µs seulement, une fois le programme chargé sur le PIC et testé dans un environnement de réception, la détection des périodes d'interruptions ne fonctionnait pas.

L'algorithme en lui-même était simple, il consistait à configurer la broche 1 du PIC en entrée et la broche 2 du PIC en sortie. Une fois cette étape de configuration terminée, on regarde le signal présent en entrée sur la broche 1, tant qu'il ne s'agit pas d'un zéro, on boucle sur l'instruction de test (c'est-à-dire qu'on ne fait rien d'autre qu'attendre que le signal soit un zéro). Une fois un zéro détecté, on déclanche une temporisation de 40µs puis on regarde à nouveau la broche en entrée, s'il y a toujours un zéro, on relance un nouvelle temporisation de 40µs, sinon, on se replace dans l'état initial d'attente de la présence d'un zéro en entrée.
Cette seconde temporisation, si on a toujours un zéro en entrée, se répète 6 fois de façon à être certain que le signal n'est pas passé à 1 pendant plus de 40µs sur une période de 280µs. Lorsque cette situation est détectée, on place sur la broche configurée en sortie un « un » pendant 100µs.

Lors des simulations, à chaque fois que notre signal d'entrée restait à zéro pendant plus de 260µs, on avait un échelon sur la broche de sortie. Notre signal de sortie présentait donc un échelon de tension toutes les 4ms et celui-ci avait une durée de 100µs.

Lorsque nous l'avons testé sur le PIC et en réception, nous n'avons pas observé le comportement que nous espérions et nous ne savons toujours pas pourquoi.

Le listing complet de l'algorithme est présenté en Annexe 4.

IV. CONCLUSION

Lors de ce Bureau d'Etude nous avons réalisé l'intégralité des chaînes d'émission et de réception d'une transmission numérique par voie optique à l'air libre. Afin de pouvoir y parvenir, nous avons du mettre en pratique autant des connaissances que nous avions acquise pendant les années précédentes (en optique et éléctronique par exemple) mais aussi des connaissances nouvelles acquises au cours de l'année en cours (notions de codage, transmission, modulation, filtrage actif, architecture de microcontrolleur, language d'assemblage, structure et fonctionnement des ordinateurs)

Le sujet possédant une version commerciale fonctionnelle est très motivant et permet d'avoir à tout moment une image du résultat escompté en tête.

Tout au long du Bureau d'Etude, nous avons été confronté à des problèmes aussi bien au niveau éléctronique qu'au niveau informatique et nous avons du apporter les solutions permettant de les surmonter. Le Bureau d'Etude nous a donc appris à utiliser de nouvelles techniques et à trouver les solutions aux problèmes liés à leur utilisation pratique.

V. ANNEXES

Annexe 1 : Programme de temporisation

Annexe 2 : Programme d'émission en bande de base

Annexe 3 : Programme d'émission du signal modulé

Annexe 4 : Programme de détection des pauses en réception.

Annexe 5 : Divers

Annexe 1 :

Programme de temporisation

Annexe 2 :

Programme d'émission en bande de base

Annexe 3 :

Programme d'émission du signal modulé

Annexe 4 :

Programme de détection des pauses en réception

Annexe 5 :

Divers