TEXT'IOGS

Transmission filaire d'un message texte par modulation de fréquence

Ahcène BENSALAMA
Aloïs FOURNIER
François OLLITRAULT
Yohan BLAZY

INSTITUT D'OPTIQUE Graduate School
Première Année

28 mai 2018

Notice d'utilisation

Emetteur

Dès que l'émetteur est alimenté, il émet un signal à intervalle de temps régulier correspondant au message à transmettre. Pour modifier le message à transmettre il faut connecter le microcontrôleur à un ordinateur via le boitier MicroChip et uploader le code de base en modifiant ayant la liste L en tête de programme. La liste L accepte une séquence binaire correspondant au code des caractères ASCII. La conversion ASCII vers binaire n'est pas prise en compte par le programme.

Alimentation

Le montage doit être alimenté avec une tension continue +5V / 0V sur le bord de la plaquette (ROUGE: +5V, NOIR : 0V).

Récepteur

Le récepteur est composé de deux blocs à relier entre eux. Le premier se charge de la démodulation du signal reçu et le second, de la restitution du message sur un écran LCD.

Bloc démodulation

Le bloc démodulation se charge de recevoir le signal émis par l'émetteur. Il faut ainsi brancher la sortir de l'émetteur sur l'entrée du bloc démodulation (cf. Schéma du montage).

Alimentation

Le montage doit être alimenté en +5V / -5V au niveau des lignes équipotentielles bordant la plaquette (rouge : +5V, bleu : -5V, noir : GND)

Attention — Le bloc démodulation utilise deux filtres passe-bas pour lesquels il faut fixer la fréquence de coupure. Pour ce faire, ces filtres doivent recevoir un signal **créneau d'amplitude pic à pic 5V, centré en +2,5V**. Pour le filtre A, la fréquence du signal doit être de 2000kHz, et pour le filtre B, elle doit être de 200kHz.

Bloc de restitution du message

Ce second bloc est connecté aux deux sorties des détecteurs de crête et à un écran LCD. (cf Schéma électrique).

Alimentation

Seul le microcontrôleur doit être alimenté en 0V / 5V. De plus, les voies 0 et 1 doivent respectivement être branchées aux voies 0 et 1 de sortie du bloc de démodulation.

Planning et répartition des tâches

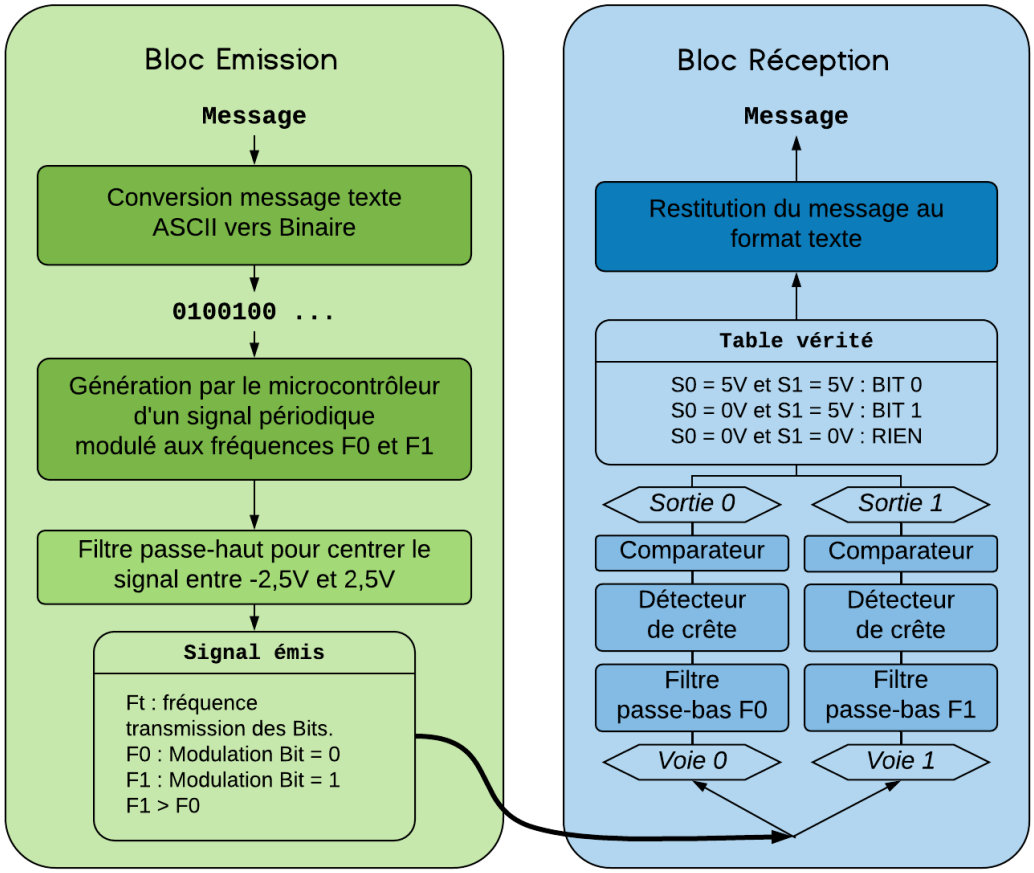
Dans un premier temps, le travail a été divisé entre les quatre membres de la manière suivante :

- **Conception et réalisation de l'émetteur** : Ahcène et François (binôme sur la partie software et hardware)
- **Conception et réalisation du récepteur** : Aloïs (hardware pour démodulation) et Yohan (software pour restitution du message)

Lors de la mise en relation des différents blocs et pour la finalisation du montage, cette division n'était plus visible.

Ahcène/François	Aloïs	Yohan	Tout le groupe
14 mars (journée)	Réflexion sur la mise en place technique du projet et adaptation du projet initial en un projet réalisable.		
	Choix du type de modulation et caractéristiques pour la transmission		
	Premier programme pour le microcontrôleur de l'émetteur (porteuses périodiques générées par GBF)		
	Premier programme pour le microcontrôleur du récepteur		
	Travail sur le circuit de démodulation (choix des filtres et de l'architecture, premiers tests)		
28 mars	Finalisation du premier programme pour l'émetteur et montage du circuit sur plaquette		
	Finalisation du programme pour le récepteur Définition des contraintes sur le signal transmis		
	Circuit de démodulation : travail sur le détecteur de crête et comparateur		
	Révision des objectifs en raison de la contrainte de temps		
04 avril	Finalisation du circuit de démodulation et premiers tests		
	Travail sur une source externe générant les porteuses pour l'émetteur et pour les signaux périodiques pour les filtres du circuit de démodulation		
	Nouveau programme pour l'émetteur dans lequel les porteuses sont générées par le microcontrôleur		
15 mai	Finalisation du nouveau programme pour le microcontrôleur		
	Premier assemblage des blocs et série de tests		
	Identification des problèmes au niveau du bloc de démodulation et mise en place de premières corrections		
24 mai	Ajout d'un étage d'amplification entrée du bloc démodulation du récepteur		
	Nouveaux tests sur le système global et identifications des problèmes		
	Premier message transmis avec erreurs et modifications pour améliorer la restitution		
	Finalisation et « nettoyage » du montage pour tourner la vidéo de présentation		
Autres	Ecriture et tournage et montage de la vidéo de présentation		

Schéma de principe et Blocs fonctionnels



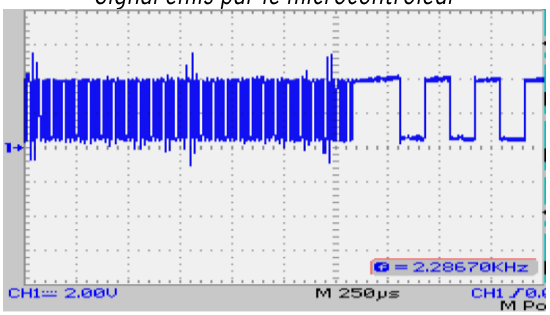
- **Microcontrôleur Emetteur**
- **Filtrage Emetteur**
- **Bloc analogique démodulation**
- **Microcontrôleur Récepteur**

Description des blocs fonctionnels

Bloc Emission

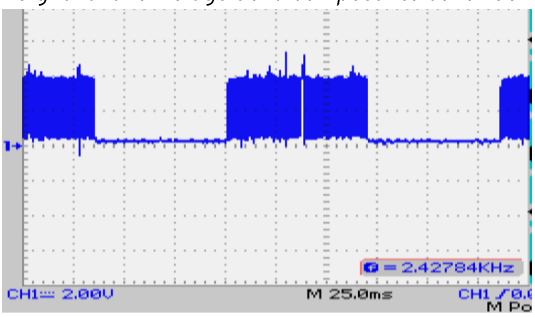
Le microcontrôleur de l'émetteur se charge de générer portant un message texte composé d'une suite de caractères au format ASCII. A chaque caractère ASCII correspond un nombre qui est codé sur 8bits en binaire. Le signal émis est rectangulaire de fréquence variable renouvelée à une fréquence **Ft = 244 Hz** et valant soit **F0 = 4,1 kHz** soit **F1 = 41 kHz** selon qu'il faille transmette un 0 ou un 1 dans le message binaire. Le message transmit est « wa ». La suite binaire correspondante est transmise à intervalle de temps régulier de 131 ms et elle est émise sur 66 ms.

Signal émis par le microcontrôleur

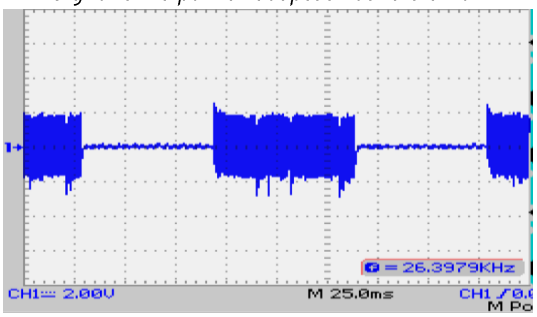


Ici on voit une transition entre les deux fréquences de modulation. On remarque un temps de latence au moment du changement de fréquence, du au microcontrôleur. En prenant Ft << F0, cela ne perturbe pas le décodage du message.

Signal avant filtrage de la composante continue

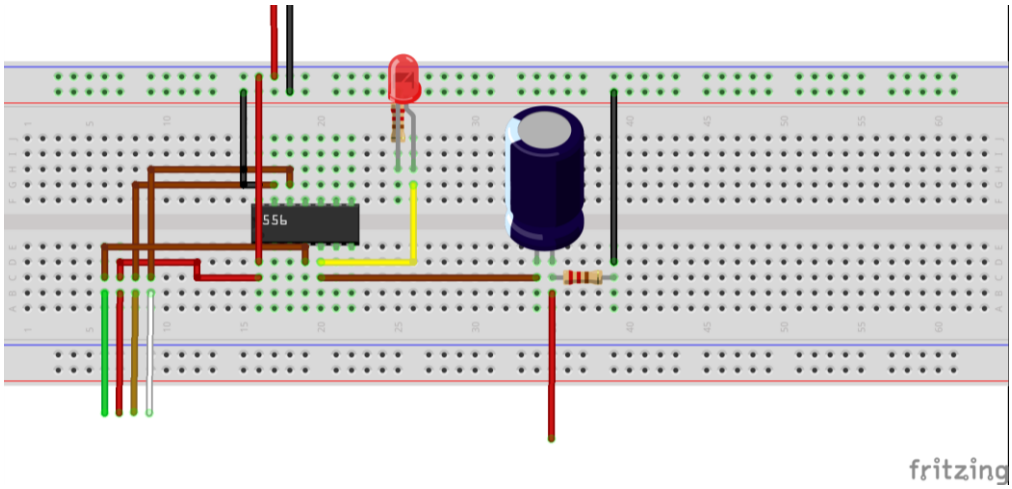


Signal émis par le récepteur centré en 0V



Shéma électrique - Bloc Emission

En haut du shéma il s'agit de l'alimentation. La sortie du microcontrôleur correspond aux fils jaune et marron. Ce signal est ensuite filtré par un filtre RC (à droite de la plaquette) en configuration passe-haut (tension mesurée aux bornes de la résistance) de fréquence de coupure 133 Hz — **R = 1200 kΩ - C = 1µF** — La fréquence de transmission ainsi que les fréquences de modulation ne sont pas affectées mais la composante continue du signal est bien filtrée. La LED rouge permet de vérifier qu'il y a bien un signal émis par le microcontrôleur.



Bloc Démodulation

Le bloc de démodulation admet en entrée un signal périodique centré en 0V. Il est conçu pour démoduler un signal modulé en fréquence avec F0 (fréquence pour le bit 0) et F1 (fréquence pour le bit 1) telles que **F1 > 10 F0**. Les fréquences de coupures des filtres passe-bas sont contrôlées par un signal périodique reçu par les deux filtres (**F0'** et **F1'**). Les fréquences de coupure des filtres sont respectivement **F0'/50** et **F1'/50**.

En dessous d'un certain seuil sur l'amplitude du signal reçu, la démodulation ne fonctionne plus. Ce seuil **Vs** est fixé par la tension de comparaison du comparateur (comprise entre 0V et 5V et fixée par le potentiomètre C) et par le gain de l'étage d'amplification (G=2,2). Ce seuil est tel que : **Vs** = **Vpot. / 2,2**.

Les deux sorties S0 et S1 génèrent un signal en tout ou rien (+5V / 0V) en fonction du signal reçu :

Fréquences dans le signal en entrée	S0	S1
F < F1'/50	+5V	+5V
F < F0'/50	+5V	0V
Amplitude GND to pic du signal en entrée		
Ve < Vs	0V	0V

Exemple de fonctionnement avec des valeurs typiques

— Vpot = 1V donc Vs = 450mV — F0' = 2000kHz et F1' = 200kHz — Ve = 2V —

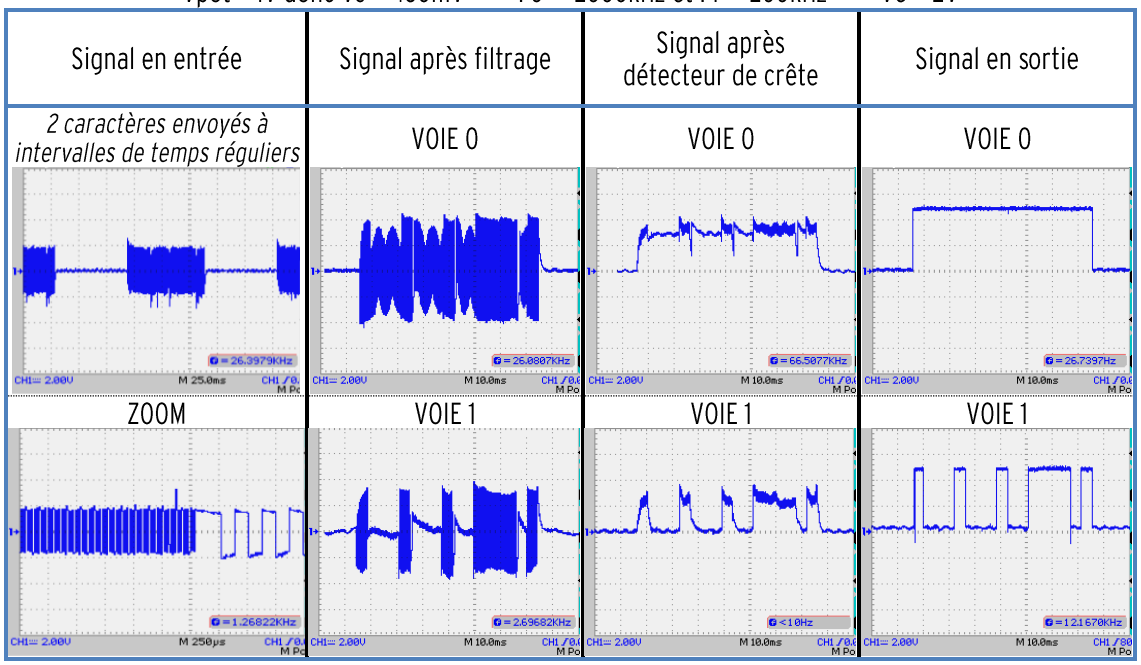
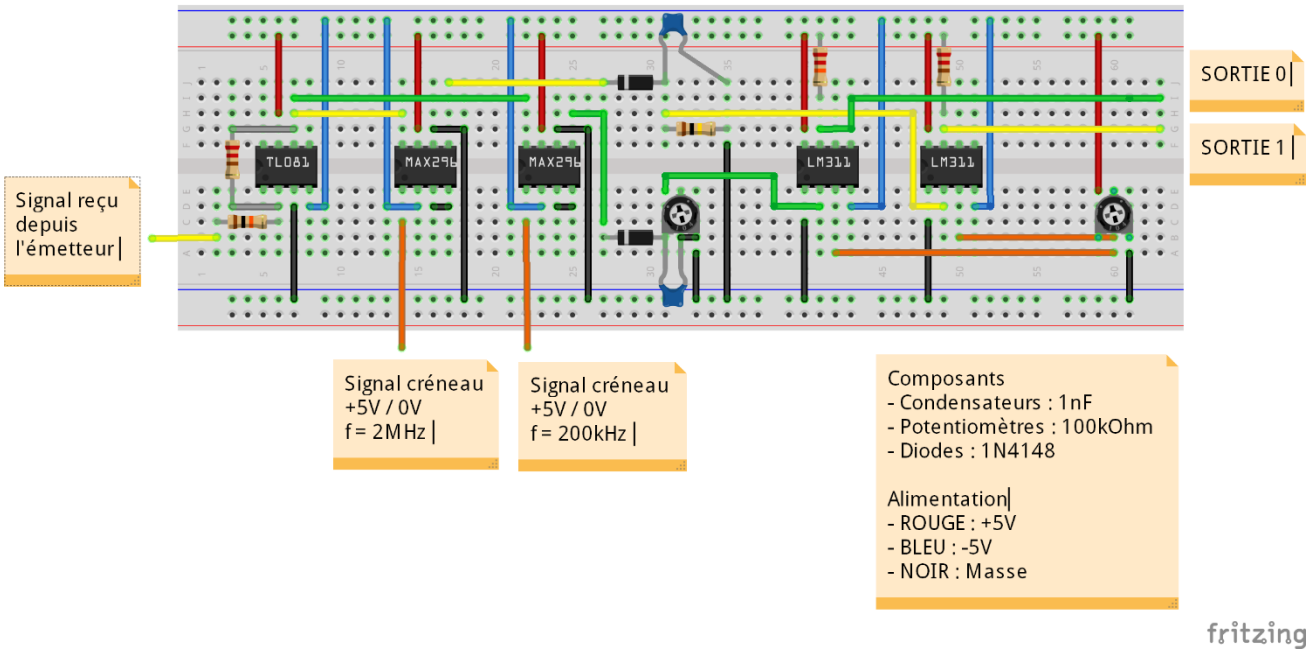


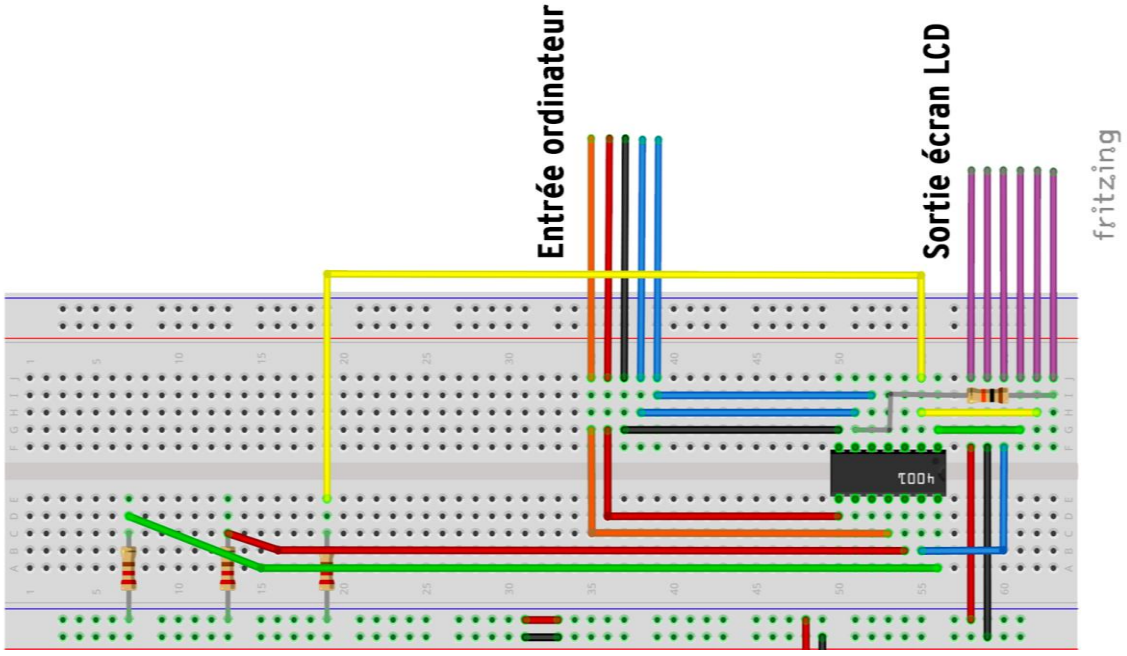
Schéma électrique — Bloc Démodulation



Bloc Restitution du message — Microcontrôleur du récepteur

Le bloc de réception admet 2 voies en entrée et est branché à un écran LCD. Le rôle de ce bloc est de recevoir la suite de bits correspondant au message texte et de la transformer en caractères à afficher sur l'écran. Pour ce faire, on utilise un algorithme dont le principe est le suivant. On contrôle de manière régulière (à une fréquence égale à la fréquence d'envoi) les deux entrées du bloc. Si la voie 1 est à 0, il n'y pas de transmission, on ne fait donc rien et on réinitialise toutes les variables. Si la voie 1 est à 1 alors on stock dans une variable la valeur de la voie 2. Une fois que l'on a stocké 8 bits dans cette variable, on réalise la conversion vers le caractère correspondant que l'on affiche sur l'écran. (cf. Code en annexe)

Schéma électrique — Bloc Restitution



Annexe A — Code C pour le microcontrôleur du bloc Emission

```
/* Le programme utilise deux horloges. La première horloge A contrôle la
période de transmission des bits. A chaque coup d'horloge A on lit le
prochain bit à transmettre. La deuxième horloge B permet de contrôler la
période d'oscillation du signal émis. Si le bit lu au moment du coup
d'horloge A est 0 (resp. 1) on modifie la période de l'horloge B pour
obtenir la bonne fréquence F0 (resp. F1).*/

#include <xc.h>
#include <stdio.h>
#include <stdint.h>

#include "./Ressources/config.h"
// le point slash Ressources signifie que l'ordinateur trouve ressources
dans le même dossier que le programme
#include "./Ressources/spi.h"

#define pr20 59 //Valeur de PR2 correspondant à f0
#define pr21 5   //valeur de PR2 correspondant à f1

int valeurF0 = 0.5 * 4 * (pr20 + 1); // configuration pour que RC = 50%.
int valeurF1 = 0.5 * 4 * (pr21 + 1); // configuration pour que RC = 50%.

int L[] =
{1,1,1,0,1,1,1,0,1,1,0,0,0,0,1,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2};
// Liste de zéros et uns qui correspond au message. Les 2 servent à
marquer une pause à la fin du message.
int i=0;

void initPIC(void);
void change_freq(int BIT);

void main(void) {
      initPIC();
      while(1){}
      return;
}

void initPIC(void){
      TRISCbits.TRISC5 = 0;
      //La broche RC5 (reliée à PWM1)est une sortie
      TRISCbits.TRISC4 = 0;   //La broche RC4 est config en sortie
      TRISCbits.TRISC3 = 0;   //La broche RC3 est config en sortie
      PWM1CONbits.PWM1EN=1;
      PWM1CONbits.PWM1OE=1;
      PWM1CONbits.PWM1POL=1;

      PR2=0;
      T2CONbits.T2CKPS=0b00;
      //Le facteur de predivision est mis à 0b00=1 p.153
      T2CONbits.TMR2ON=1;   // on allume le timer 2
      T2CONbits.T2OUTPS=0b0000; // postscaler est mis à 1

      OSCCONbits.IRCF = 0b1011; // 1MHz fréquence oscillateur interne
      OPTION_REGbits.PS = 0b001; // PS prédiviseur 0b001=4
```

```
	// TIMER0 permet de générer l'intervalle de temps de transmission
	des bits.
	OPTION_REGbits.T0CS = 0 ;
	// Pour lancer le timer il faut clear le TMR0CS
	OPTION_REGbits.PSA = 0;
	//PSA est mis à zéro car on passe par le prescaler
	INTCONbits.TMR0IE = 1;
	// autorise l'interruption par le timer0 quand tmr0if = 1
	INTCONbits.GIE = 1;   // autorise les interruptions
	return;
}

void change_freq(int BIT){
// Fonction qui change la fréquence de TMR2 en fonction du BIT à
transmettre
	if (BIT == 0) {
		PR2 = pr20;
	// configuration du PR2 pour que la fréquence du TMR2 soit F0.
		PWM1DCL = valeurF0 << 6;
		PWM1DCH = valeurF0 >> 2;
	}
	if (BIT == 1) {
		PR2 = pr21 ;
	// configuration du PR2 pour que la fréquence du TMR2 soit F1.
	PWM1DCL = valeurF1 << 6;
	PWM1DCH = valeurF1 >> 2;
	}
	// on rajoute le 2 pour permettre de générer des périodes de non
	transmission sans éteindre l'émetteur ceci est utile pour les tests
	de bon fonctionnement
	if (BIT == 2){
		T2CONbits.TMR2ON=0;   // on éteint le timer 2
	}
}


void interrupt prog_interruption(void){
	if(INTCONbits.TMR0IF == 1){
		change_freq(L[i]);
		i = i + 1;
	// permet d'avoir des intervalles de temps sans transmission
		if (i == 33) {
			i = 0;
			T2CONbits.TMR2ON=1; // on rallume le timer
		}
	INTCONbits.TMR0IF = 0;
	}
}
```

Annexe B — Code C pour le microcontrôleur du récepteur

```
54
55  /* l'interruption ci_dessous se dÈclenche ‡ une frÈquence qui est celle du timer. Le principe est que
56  l'information est reÁue ‡ cette frÈquence, ainsi pour chaque bit reÁu, on Èvalue la valeure des
57  deux entrÈes.On modofie ensuite bin, qui correspond au carractËre reÁu en binaire. Au bout de
58  nb_bit itÈrations, on a reÁu le caractËre complet, ce qui permet de le convertir et de l'afficher
59  sur un Ècran et de commencer la rÈcÈption du bit suivant*/
60
61
62  void interrupt prog_interruption(void){
63      if(INTCONbits.TMR0IF == 1){      // Chapitre 7 - registre 7-1 - p. 64
64
65          if((PORTCbits.RC5 == 0) && (PORTCbits.RC3 == 1)){
66              bin = bin << 1;
67              bin += 1;
68              i++;
69
70          }
71          if((PORTCbits.RC5 == 1) && (PORTCbits.RC3 == 1)){
72
73              bin = bin << 1;
74              i++;
75
76          }
77          if (i==nb_bit+1){
78              PORTCbits.RC1 = !PORTCbits.RC1;
79              i=1;
80
81
82              indic = 1;
83              lettre[0] = conversion_bin(bin);
84              bin = 0b00000000;
85
86
87
88          }
89          if(PORTCbits.RC3 == 0){
90              i = 1;
91              bin = 0b00000000;
92
93
94          }
95          /*bin = 0b00000000;
96          bin = bin + 119;
97          lettre[0] = conversion_bin(bin);*/
98
99          INTCONbits.TMR0IF = 0;
100     }
101 }
102 int conversion_bin(int bin){
103     return bin  ;
104 }
```

```c

#include <xc.h>
#include "../Ressources/config.h"
#include "../Ressources/lcd.h"

void initPIC(void);
int conversion_bin(int bin);
int i = 1;                //i est le nombre de bits nÈcessaire pour coder un caractËre
int bin = 0b00000000;     //bin correspond au code du caractËre en binaire
char lettre[2];
int indic = 0;
int nb_bit = 8;
int ligne =0;


void main(void) {
    lettre[1] = '\0';
    initPIC();
    initLCD_DOG();
    clearLCD();            //On initialise le timer et l'Ècran LCD
    while(1){
        if(indic == 1 ){
            PORTCbits.RC1 = 0;
            writeStrLCD(lettre,1,ligne);
            ligne++;
            indic = 0;
        }
        if (ligne == 30){
            ligne = 1;

        }
    }
    return;
}

void initPIC(void){
    ANSELA = 0;
    ANSELC = 0;
    TRISCbits.TRISC1 = 0;  //sortie test reliËe ‡ une LED
    TRISCbits.TRISC3 = 1;  //deux entrÈes pour reccevoir les deux signaux du rÈcÈpteur
    TRISCbits.TRISC5 = 1;

    OSCCONbits.IRCF = 0b1011;       //dÈfini la frÈquence du timer

    OPTION_REGbits.PS = 0b001;     // Chapitre 18 - figure 18-1 - p. 137 dÈfini la prÈdivision du timer
    OPTION_REGbits.T0CS = 0;    // Chapitre 18 - registre 18-1 - p. 139
    OPTION_REGbits.PSA = 0;

    INTCONbits.TMR0IE = 1;  // Chapitre 7 - 7-1 - p. 64
    INTCONbits.GIE = 1;     // Chapitre 7 - 7-1 - p. 64
    return;
}
```