
RAPPORT DE PROJET
PROJET : EasySCOPE

RÉSUMÉ – Ce rapport de projet a pour but de décrire les objectifs du projet EasySCOPE, de faire le point sur les fonctionnalités développées, de présenter comment a été conçu le projet final.

Jules Soufli-Jumel, Eric Chen, Quentin Gavoille

Nous attestons que ce travail est original, qu'il est le fruit d'un travail commun au trinôme et qu'il a été rédigé de manière autonome.

Paris, le 19/11/2022

Table des matières

I. Objectifs	p. 3
II. Glossaire	p. 4
A. Acronymes	p. 4
III. L'équipe	p. 5
A. Présentation de l'équipe	p. 5
B. Organisation de l'équipe	p. 5
C. Diagramme de GANTT	p. 6
IV. Contexte et problématique	p. 7
A. Contexte	p. 7
B. Problématique	p. 7
C. Spécifications techniques	p. 7
V. Conception	p. 8
A. Architecture fonctionnelle	p. 8
B. Architecture matérielle	p. 9
C. Architecture logicielle	p. 11
VI. Développement	p. 13
A. Menu	p. 13
B. Affichage 7-segment	
p. 15	
C. Oscilloscope et trigger	
p. 18	
D. Signal rectangulaire	
p. 19	
VII. Tests et validation	p. 21
A. Menu	p. 21
B. Affichage 7-segment	
p. 22	
C. Oscilloscope et trigger	
p. 23	
D. Signal rectangulaire	
p. 26	
VIII. Bilan	p. 27
A. État d'avancement	p. 27
B. Pertinence de la solution technique	p. 27
C. Bilan sur le travail d'équipe	p. 27
IX. Sources	p. 28

I. Objectifs

Ce rapport de projet est un compte-rendu final de notre projet EasySCOPE. Il a pour rôle de décrire la problématique de départ du projet, quelles sont les attentes, les besoins du cahier des charges. Ensuite il permet également de comprendre ce que l'équipe a réalisé, les fonctionnalités développées, la conception du projet et éventuellement ce qui n'a pas pu être fait. Grâce à ce document, le lecteur pourra comprendre comment l'équipe de travail a développé une solution pour répondre à la problématique initiale.

Dans ce rapport de projet, le lecteur va pouvoir trouver en premier lieu différentes aides (glossaire, acronymes) afin de mieux comprendre le vocabulaire lié à ce projet de microcontrôleur. Ensuite, il pourra y trouver une présentation de l'équipe, une explication de comment a été réparti le travail ainsi qu'une section contexte et problématique de ce projet. Après cela, il y aura les parties de conception et de développement, puis les tests et validation. Enfin, ce rapport sera conclu par différents bilans, les sources et les annexes.

II. Glossaire

A. Acronymes

Acronyme	Signification
MCU	Microcontrôleur. C'est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires, unités périphériques et interfaces d'entrées-sorties.
ADC	Le convertisseur analogique-numérique (ADC en anglais) est un dispositif électronique permettant de convertir une grandeur analogique en entrée en une valeur numérique codée sur plusieurs bits.
GPIO	Ce sont les ports d'entrées et de sorties d'un composant électronique.
PWM	Pulse Width Modulation (modulation de largeur d'impulsion en français) est une technique dont le but est de générer un signal pseudo analogique à partir d'un environnement numérique ou analogique
GLCD	Graphical Liquid Crystal Displays. Ecran intégré sur la EasyPIC sur lequel on peut afficher du texte, des images ou dans notre cas des signaux.
GBF	Générateur basse fréquence. C'est un appareil qui permet de délivrer un signal avec la fréquence désirée sous forme de sinusoïdes, de créneaux, ou de triangles. On peut également régler différents paramètres (fréquence, rapport cyclique, amplitude,...). Ce signal peut être observé grâce à un oscilloscope.

III. L'équipe

A. Présentation de l'équipe

Pour la réalisation du projet EasySCOPE, nous avons gardé la même équipe que lors des TP de microcontrôleur. Elle se compose de trois membres : Eric Chen, Jules Souffi-Jumel et Quentin Gavoille.



Nos compétences acquises en langage C, en microcontrôleur et en électronique au cours de nos 4 années d'études à l'ECE nous ont permis de mener à bien ce projet. De plus, la curiosité de certains, l'assiduité et la rigueur d'autres, font de notre groupe une équipe de travail complémentaire.

B. Organisation de l'équipe

Avec l'objectif de réaliser un projet fonctionnel dans le temps imparti nous avons dû faire preuve d'organisation et nous nous sommes répartis le travail de la manière suivante.

Tâches	Eric	Jules	Quentin
Visuels écran d'accueil	X	X	X
Affichage 7-segment displays		X	X
Menu principal/choix mode			X
Affichage oscilloscope mode 1	X	X	
Gestion des interruptions/boutons		X	
Signal rectangulaire mode 2	X		
Documentation technique	X	X	X

C. Diagramme de Gantt

Le Diagramme de GANTT décrit notre utilisation du temps alloué pour le projet.

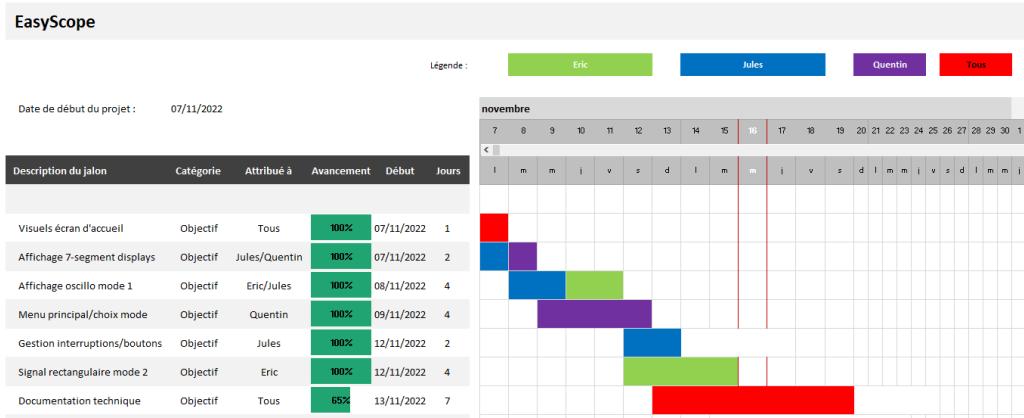


Figure 1: Diagramme de Gantt

Nous avons commencé à travailler sur le projet le lundi 7 novembre. Nous avions un exemple de code en C sur boostcamp qui utilisait l'écran GLCD. Nous avons donc étudié ce code puis nous avons débuté notre propre programme. On a implémenté les visuels pour les écrans d'accueil tous ensemble. Les deux premiers jours, nous avons également réussi à afficher la valeur de la tension analogique en entrée sur les 7-segment displays.

Pour le reste du projet, nous nous sommes répartis les tâches pour avancer plus rapidement. Nous avons formé deux groupes pour effectuer deux tâches de même durée (4 jours environ) en parallèle : l'affichage de l'oscilloscope pour le mode oscilloscope, l'affichage du menu principal pour le choix du mode et le mode signal rectangulaire.

Les deux dernières tâches de développement ont été réalisées à partir du 12 novembre. En simultané, nous avons commencé à rédiger le rapport de projet le dimanche 13.

Nous avons consacré le temps restant aux tests sur oscilloscope et avec GBF.

IV. Contexte et problématique

A. Contexte

Un oscilloscope est un instrument de mesure destiné à visualiser un signal électrique, le plus souvent variable au cours du temps. Le premier oscilloscope électromécanique été inventé en 1893, il se dote en 1932 d'un écran cathodique avant que Tektronix ne lance en 1946 le premier oscilloscope analogique à balayage déclenché. Dans les années 1980, le développement des oscilloscopes numériques s'accélère. Ces instruments intègrent des convertisseurs analogiques-numériques qui échantillonnent les signaux à une fréquence déterminée pour afficher les échantillons numériques à l'écran.

Dans le cadre de notre cours de microcontrôleur en majeure Systèmes Embarqués, nous devions réaliser un oscilloscope ainsi qu'un générateur de signal rectangulaire. Ce projet nous a permis de mettre en application les connaissances, sur les microcontrôleurs, vues lors de ce semestre.

B. Problématique

Le but de ce projet est la réalisation d'un oscilloscope et d'un générateur de signaux en utilisant la carte EasyPIC board v7 avec son écran GLCD, les 7-segment displays, le PIC18F4550 et les ports d'I/O de la carte.

Le projet se décompose en différentes parties que nous appellerons modules. Le premier est le menu, avec l'affichage de début, avec le titre, les noms et le menu avec le GLDC. Puis il y a ensuite le multimètre 7-segment. A partir d'une certaine tension analogique en entrée, nous devons utiliser l'ADC de la carte pour la convertir en une valeur analogique puis l'afficher sur les 4 7-segment displays avec une conversion entre 0 et 5V. Le troisième module est l'oscilloscope : à l'aide du GLCD, nous devons afficher le signal d'entrée ainsi que ses caractéristiques. Il faut aussi configurer un bouton pour pouvoir choisir entre deux modes : running et trigger. Enfin, le dernier module est le générateur de signaux rectangulaires. Pour information, nous avons ajouté un autre module (qui n'est pas dans le cahier des charges) qui correspond à un menu de sélection. Celui-ci sera présenté avant les trois autres dans ce rapport.

C. Spécifications techniques

Pour ce projet, nous devions utiliser la carte EasyPIC v7 pour générer différents signaux, et l'écran GLCD pour les afficher ainsi que certaines informations comme pourrait le faire un oscilloscope. Pour la programmation, nous avons décidé de coder en langage C sur MPLAB X. Nous utilisons mikroProg pour "téléverser" le code .hex vers la board. Nous serons également amenés à utiliser un GDF et un oscilloscope classique pour tester le bon fonctionnement des fonctionnalités que nous développons.

V. Conception

A. Architecture fonctionnelle

Les différentes fonctionnalités à implémenter sont :

- Afficher la tension d'entrée sur les 7-segment displays
 - Afficher la courbe et ses caractéristiques sur l'écran GLCD
 - Pouvoir réaliser des actions via interruptions externes
 - Créer un générateur de signal rectangulaire et l'afficher sur l'écran

Considérons le diagramme fonctionnel suivant :

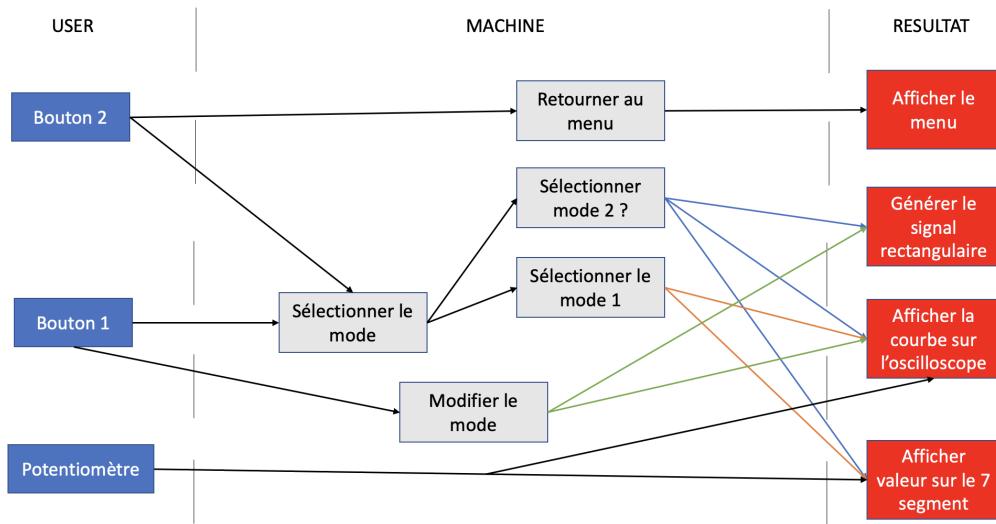


Figure 2: Schématique architecture fonctionnelle

Comme nous pouvons le voir sur le schéma ci-dessus, nous avons trois entrées utilisateur. Le potentiomètre, pour régler la valeur de la tension analogique d'entrée, ainsi que les valeurs de fréquence et de rapport cyclique (dans le mode signal rectangulaire). Les deux boutons 1 et 2 pour les actions.

Il y a différents traitements dans notre programme. Dans le menu principal, les deux boutons permettent de sélectionner l'un des deux modes. Le premier pour le choix du mode et le second pour la validation. Lorsque l'on est dans le mode oscilloscope, on peut choisir de passer en mode running ou trigger (choix avec le bouton 1), ou de revenir en arrière avec le deuxième bouton. Dans le mode signal rectangulaire, on peut choisir de sélectionner la fréquence ou le rapport cyclique (bouton 1) et de faire une modification avec le potentiomètre. On a la possibilité de revenir au menu principal en appuyant sur le bouton 2.

Finalement, nous avons quatre résultats possibles en sortie. On affiche le menu lorsque l'on démarre le programme ou si l'on retourne au menu via le bouton 2. On affiche la valeur de la tension d'entrée sur les 7-segment displays. On affiche la courbe sur l'oscilloscope si on a sélectionné le mode 1. On génère le signal rectangulaire et on affiche la courbe sur l'oscilloscope si on a sélectionné le mode 2.

B. Architecture matérielle

Pour réaliser ce projet, nous utilisons l'easypic board sur laquelle nous trouvons les différents composants dont nous avons besoin pour le projet. Considérons ce schéma proteus qui recense les différents composants utilisés et les liens entre eux :

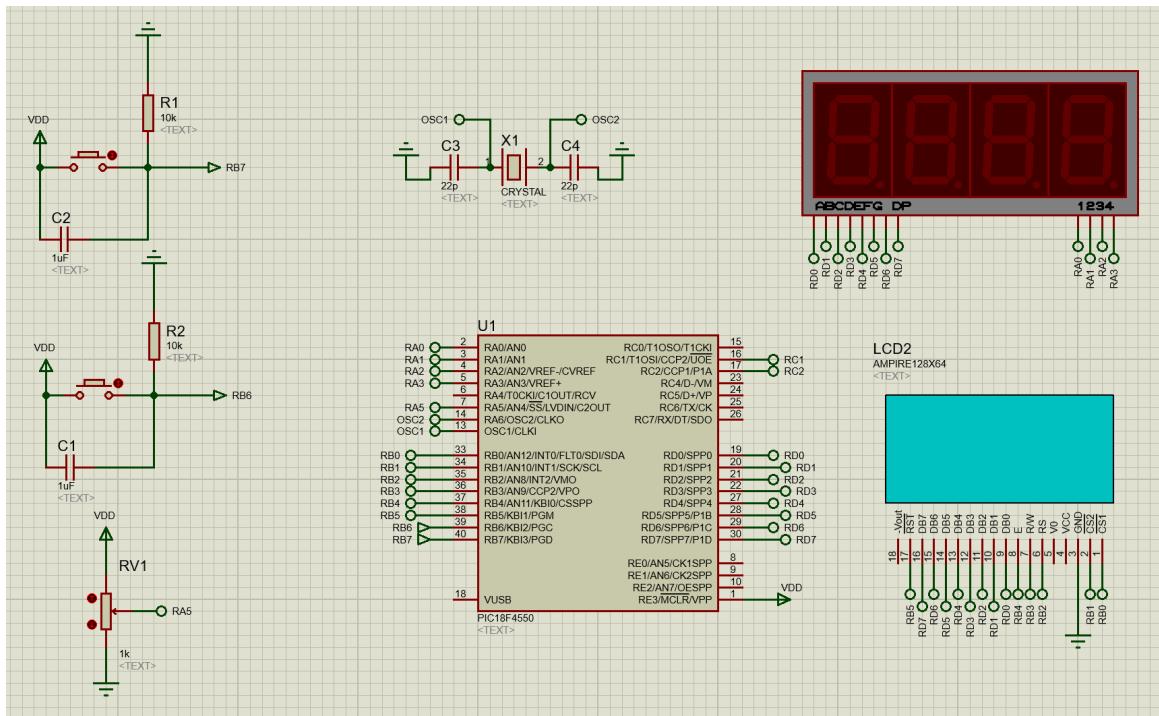


Figure 3: Schématique architecture hardware

De gauche à droite et de haut en bas, décrivons l'architecture matérielle de notre projet. On a d'abord le schéma de câblage pour les boutons reliés à RB7 et RB6 qui vont nous servir pour la réalisation des différentes actions. Ces deux boutons sont montés en pull down. Leur circuit se compose d'une alimentation VDD (5V), un résistance de 10k Ohm en pull down, un condensateur de découplage de 1uF et le ground.

On a, en dessous des deux boutons, le potentiomètre qui est une résistance variable permettant de laisser passer plus ou moins de courant et donc de réguler la tension que nous envoyons à l'entrée AN4. On l'utilise pour transmettre une tension analogique plus ou moins grande en entrée de l'ADC.

Ensuite, le milieu de notre schématique proteus représente la PIC18F4550 qui est le microcontrôleur que nous utilisons pour ce projet. Nous pouvons voir aussi le circuit, régissant l'horloge, composé d'un quartz et de deux condensateurs de 22pF. C'est une clock de fréquence 8 MHz qui gère les différents timers du PIC18F.

Enfin, la partie à droite de la photo montre les composants pour l'affichage des actions réalisées. On a premièrement les 4 7-segment displays, dont les ports sont reliés à PORTD et qui utilisent 4 ports en bits de sélection de RA3 à RA0. En dessous, l'écran GLCD dont les ports sont reliés à PORTD, et qui utilise 6 ports de RB5 à RB0 pour le mode OUTPUT.

Plus généralement, on utilise des labels pour une meilleure visibilité du câblage de la schématique sur Proteus.

Voici un tableau récapitulatif de l'ensemble des composants utilisés dans le cadre de ce projet.

Composant	Description
PIC18F4550	C'est notre microcontrôleur, de la famille des PIC. C'est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires, unités périphériques et interfaces d'entrées-sorties. Nous utilisons ses portes I/O dont voici le détail.
PORTA	<RA3:0> Gère le 7-segment en mode OUTPUT ; <RA5/AN4> entrée analogique en mode INPUT. On y lit la tension reçue et l'analysons à l'aide de l'ADC du PIC.
PORTB	<RB5:0> Gère le GLCD display en mode OUTPUT ; <RB7:6> entrée analogique en mode INPUT.
PORTC	<RC1> Utilisé pour signal rectangulaire en mode OUTPUT;
PORTD	<RD7:0> Gère les bus de données entre le 7seg et le PIC et entre le GLCD display et le PIC. Mode OUTPUT.
PORTE	<RE3> Gère le reset (Actif LOW). Mode INPUT.
Bouton pull down RB6/RB7	Relié à RB6 et RB7 respectivement en mode INPUT. Condensateur de découplage (1uF). Résistance en pull down (10k). Utilisés pour réaliser des actions via interruption.
GLCD	Graphical Liquid Crystal Displays. Reçoit des bus de données de PORTD (OUTPUT), affichage géré par <RB5:0> (OUTPUT).
7-segment Display	Reçoit des bus de données de PORTD (OUTPUT), bit de sélection via <RA3:0> (OUTPUT)
Potentiomètre (0-1k)	Résistance variable laissant passer plus ou moins de tension vers RA5/AN4 (INPUT). Utilisé pour envoyer une tension analogique vers l'ADC.
Clock 8MHz	Condensateurs de 22p, gère les timers du PIC

C. Architecture logicielle

Vous pouvez voir ci-dessous l'algorigramme du code qui explique son fonctionnement :

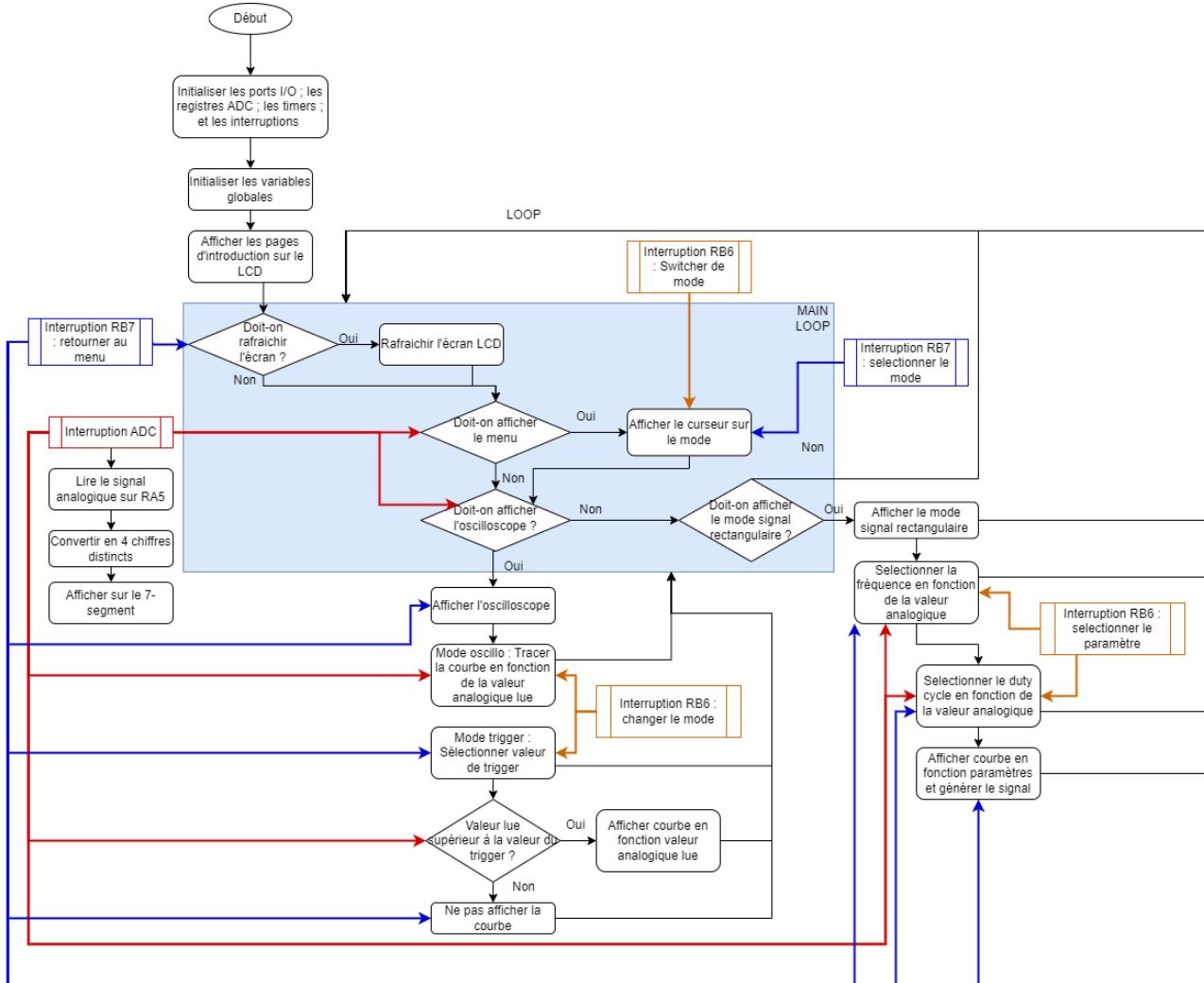


Figure 4: Algorigramme du code

Lorsque le programme se lance, on initialise les ports entrées/sorties de la PIC18F, les registres pour la conversion analogique-numérique, les timers et les interruptions. On initialise également les variables globales. Pour terminer, on affiche sur l'écran GLCD les visuels d'introduction, c'est-à-dire le titre du projet, un dessin de signal et le nom des membres du groupe.

Après cela, on rentre dans la loop du main. On réalise ensuite différents tests pour savoir si l'on doit rafraîchir l'écran puis si l'on doit afficher le menu. Si oui, on arrive sur l'écran du menu principal. On a le choix entre 2 possibilités : l'oscilloscope et le signal rectangulaire. Si on clique sur le bouton RB6, on lève une interruption qui permet de changer entre les 2 modes (on a un petit curseur en dessous du mode sélectionné). Dans ce menu, on a aussi une interruption sur RB7 pour valider le choix du mode et une sur ADC. Cette dernière est levée lorsque l'ADC overflow. Dans ce cas, on lit la valeur analogique sur RA5 et on fait la conversion pour pouvoir l'afficher sur les 4 7-segment displays.

Une fois que nous avons choisi le mode dans lequel nous voulons aller, on rafraîchit à nouveau l'écran et on affiche l'écran correspondant à notre sélection.

Si on est dans le mode oscilloscope, on affiche l'oscilloscope ainsi que la courbe du signal en fonction de la valeur analogique lue en entrée. On peut également choisir, par interruption externe sur RB6, entre un mode running (affichage continu du signal) et un mode trigger (on affiche le signal unique-

ment lorsque sa valeur atteint une certaine valeur). Dès que la courbe atteint le côté droit de l'écran GLCD (screen full), le signal recommence de la gauche et efface progressivement le tracer de la courbe précédente. Nous utilisons toujours ici l'interruption sur ADC. Dans cette partie du programme, l'interruption sur le bouton RB7 permet de revenir au menu principal à n'importe quel moment. En mode Trigger, nous proposons dans un premier temps à l'utilisateur de saisir la hauteur de la valeur de trigger, puis nous n'affichons la courbe qu'une seule fois, dès que celle-ci dépasse la valeur de trigger.

Si on est dans le mode générateur de signal rectangulaire, on affiche le visuel correspondant. On génère un signal rectangulaire avec des paramètres modifiés en fonction de la valeur analogique lue par interruption sur ADC. On sélectionne la fréquence et le duty cycle de la courbe. L'interruption sur RB6 permet de sélectionner l'un de ces deux paramètres. Ensuite, on modifie la valeur du paramètre avec le potentiomètre. On affiche finalement la courbe du signal rectangulaire correspondant, ainsi que les paramètres choisis sur l'écran GLCD. Comme précédemment, l'interruption sur le bouton RB7 permet de revenir au menu principal à n'importe quel moment.

VI. Développement

Dans cette partie, nous allons vous présenter le développement de chaque module avec des explications sur les choix techniques des composants électroniques, leur fonctionnement, ainsi que les calculs théoriques. Néanmoins, les explications porteront principalement sur le concept clef derrière la conception du projet.

A. Menu

En branchant notre carte EasyPIC v7, nous sommes tout de suite accueillis par un écran d'accueil, un écran avec le titre du projet et un écran avec nos noms :

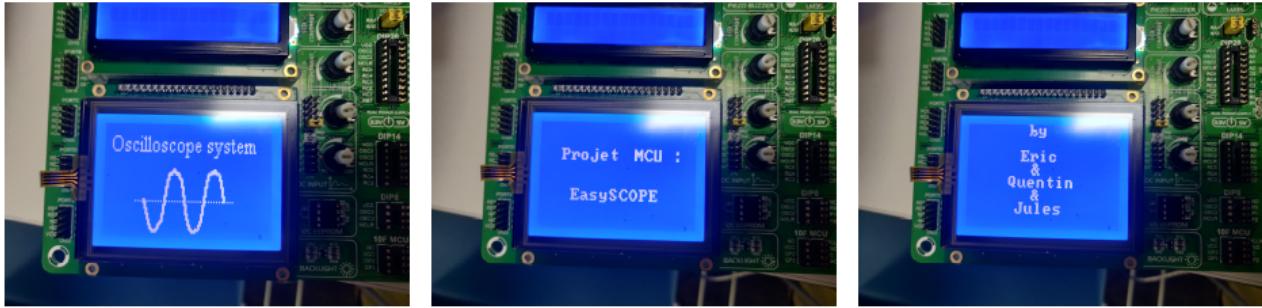


Figure 5: GLCD menu accueil

Suite à cela nous arrivons directement au menu avec, comme nous pouvons le voir, un choix entre 1 : l'Oscilloscope et 2 : le Signal Rectangulaire.



Figure 6: GLCD menu choix

Nous allons pouvoir basculer entre le mode 1 et le mode 2 avec une interruption sur le bouton **RB6** et confirmer notre choix avec une interruption sur le bouton **RB7** (cf. Tests et validation). Pour activer et mettre en fonctionnement ces boutons, il faut tout d'abord considérer les boutons **RB6** et **RB7** comme des INPUT et pour cela nous allons mettre "1" dans les bits 6 et 7 de **TRISB**. Puis dans **INTCON**, nous allons activer toutes les interruptions en mettant "1" dans le bit 7 qui est **GIE** et activer toutes les priorités hautes des interruptions si **IPEN** est égale à "1" ce qui est le cas. De plus, nous allons activer toutes les interruptions périphériques en mettant "1" dans le bit 6 qui est **PEIE** et activer toutes les priorités basses des interruptions périphériques si **IPEN** est égale à "1" ce qui est toujours le cas, et enfin activer les interruptions de changement sur le port **RB** en mettant "1" dans le bit 3 qui est **RBIE**.

Nous pouvons savoir si c'est une interruption bouton sur **PORTB** grâce au flag **RBIF** qui est dans le registre **INTCON**, et qui sera déclenché puis qu'il faudra remettre à 0. Dans **RCON**, nous allons activer les niveaux de priorité des interruptions en mettant "1" dans le bit 7 qui est **IPEN**, comme dit précédemment. Et dans **INTCON2**, nous allons activer les priorités des interruptions de changement

sur le port **RB** en mettant "1" dans le bit 0 qui est **RBIP**.

RB6 et **RB7** sont les deux boutons avec interruptions que nous utilisons dans ce projet. Ces interruptions s'activent sur les fronts montants et les fronts descendants des boutons **RB6** et **RB7**. Donc techniquement ils s'activent 2 fois, mais en mettant une condition "double_edge" qui s'incrémente quand nous entrons dans l'interruption du bouton en question, nous pouvons ne garder que le front montant et donc un seul appui sur le bouton. Vous pouvez voir ci-dessous l'état des registres que nous avons utilisés.

TRISB :

TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
1	1	0	0	0	0	0	0

INTCON :

GIE	PEIE	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
1	1	-	-	1	-	-	-

INTCON2 :

RBPU	INTEDG0	INTEDG1	INTEDG2	-	TMR0IP	-	RBIP
-	-	-	-	-	-	-	1

RCON :

IPEN	SBOREN	-	RI	TO	PD	POR	BOR
1	-	-	-	-	-	-	-

Notre GLCD possède une définition de 128 par 64 pixels, et ses pins sont connectées à différents ports de notre PIC18F4550. Le pin **CS1** va permettre de contrôler la partie droite de notre écran et est connecté au bit 0 du registre **LATB** qui est **LATB0**; tandis que le pin **CS2** va permettre de contrôler la partie gauche de notre écran et est connecté au bit 1 du registre **LATB** qui est **LATB1**. Le pin **RS** de l'écran va permettre de préparer l'envoi d'instructions qui est l'adressage quand sa valeur vaut "0" et elle permet aussi le transfert de données quand sa valeur vaut "1". Ce pin est connecté au bit 2 du registre **LATB** qui est **LATB2**.

Le pin **RW** va permettre d'indiquer si l'on doit lire ou écrire sur l'écran. Si **RW** est égal à "0" alors nous écrivons, et s'il est égal à "1" alors nous lisons les données. Il est connecté au bit 3 du registre **LATB** qui est **LATB3**. Le pin **E** va permettre d'enable et donc d'activer l'écran, cela va permettre d'écrire ou de lire les différentes données. Il est connecté au bit 4 du port **LATB** qui est **LATB4**. Enfin nous avons le pin **RST** qui va permettre le reset de l'écran GLCD et qui est aussi utile pour initialiser l'écran. Il est connecté au bit 5 du port **LATB** qui est **LATB5**. Voici un résumé de l'état du registre LATB.

LATB :

LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
-	-	RST	E	RW	RS	CS2	CS1

Cependant, **LATB** n'est pas le seul registre que nous utilisons. En effet, nous utilisons aussi tous les bits du registre **PORTD**, donc de **RD0** à **RD7**. Nous allons d'abord définir **TRISD** avec tous ses bits égaux à "0" si l'on considère le port D comme OUTPUT et donc on va écrire des données dans le GLCD avec le registre **LATD**. Ou alors nous pouvons définir **TRISD** avec tous ses bits égaux à "1" si l'on considère le port D comme INPUT et donc l'on va lire les données dans le GLCD avec le registre **PORTD**.

TRISD, LATD & PORTD :

TRISD	LATD	PORTD
GLCD_DATA_TRIS	WR_DATA	RD_DATA

En modifiant ces registres, nous pouvons créer différentes fonctions avec les deux plus importantes que sont glcd_WriteByte et glcd_ReadByte qui vont pouvoir lire et écrire des données. Avec ses deux fonctions, nous pouvons plot pixel par pixel grâce à la position x et y, écrire des strings sur 8 lignes, faire des droites...

B. Affichage 7-segment

Dans cette section, nous allons vous présenter le convertisseur analogique-numérique, le potentiomètre et les 4 afficheurs 7-segments.

En utilisant l'interruption sur le convertisseur analogique-numérique, nous pouvons récupérer la valeur de tension d'entrée du bit **RA5** dans le registre **PORTA**. **PORTA** est aussi connecté à un potentiomètre qui va pouvoir modifier la valeur de tension d'entrée. Après avoir récupéré cette valeur, nous allons pouvoir changer sa plage de valeur et la convertir pour pouvoir récupérer 4 digit qui pourront être affichés grâce aux 4 afficheurs 7-segments (cf. Tests et validation).

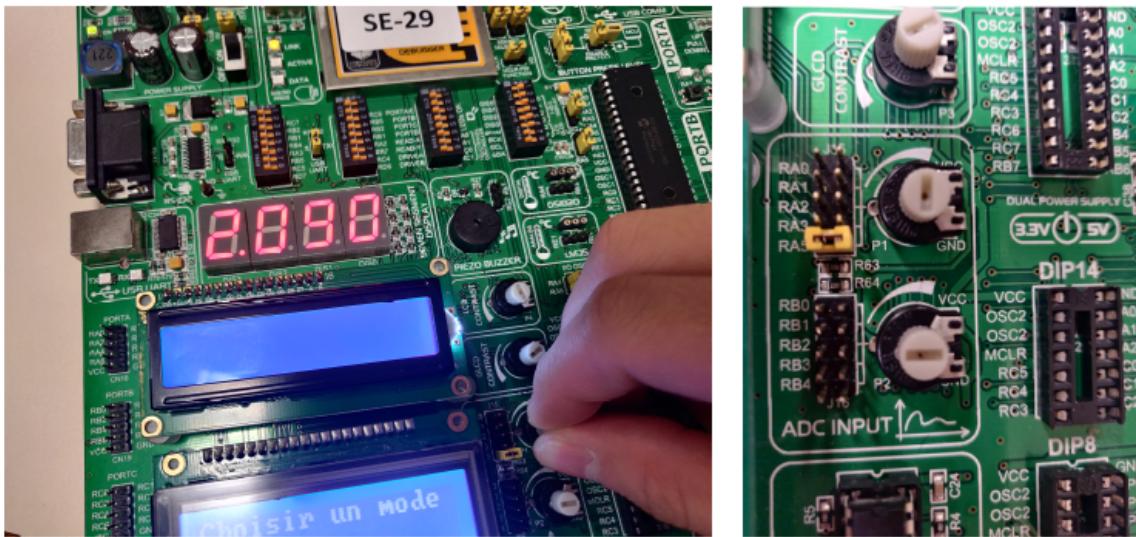


Figure 7: Convertisseur A/D

Nous allons commencer par l'interruption sur le convertisseur analogique-numérique. Dans le registre **ADCON0**, nous pouvons activer le module convertisseur A/D en mettant "1" dans le bit 0 qui est **ADON** puis nous pouvons choisir le channel analogique qui est le channel 4 qui correspond à **AN4** et qui correspond à notre **RA5** dit précédemment en mettant "0100" dans les bits 5-2 qui sont **CHS3:CHS0** (cf. Datasheet). Enfin, nous pouvons dans le main et dans une boucle, toujours mettre le convertisseur A/D en progression et ne jamais le mettre en repos en mettant "1" dans le bit 1 qui est **GO/DONE**. Dans le registre **ADCON1**, nous pouvons choisir le port de configuration du convertisseur A/D qui est dans notre cas AN4 (RA5) que l'on veut en analogique en mettant "1010" dans les bits 3-0 qui sont **PCFG3:PCFG0** (cf. Datasheet).

Enfin, nous pouvons choisir la tension de référence qui est VSS et VDD dans notre cas pour obtenir du 5V en mettant "00" dans les bits 5-4 qui sont **VCFG0** (cf. Datasheet). Dans le registre **PIE1**, nous pouvons activer les interruptions sur le convertisseur A/D en mettant "1" dans le bit 6 qui est **ADIE**. Dans le registre **IPR1**, nous pouvons mettre l'interruption de notre convertisseur A/D en basse priorité en mettant "0" dans le bit 6 qui est **ADIP**. Nous pouvons savoir si c'est une interruption du CAN grâce au flag **ADIF** qui est dans le registre **PIR1**, et qui sera déclenché puis qu'il faudra remettre à 0.

ADCON0 :

-	-	CHS3	CHS2	CHS1	CHS0	GODONE	ADON
-	-	0	1	0	0	1	1

ADCON1 :

-	-	VCFG0	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
-	-	0	0	1	0	1	0

PIE1 :

SPPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
-	1	-	-	-	-	-	-

IPR1 :

SPPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
-	0	-	-	-	-	-	-

Nous utilisons des interruptions basses pour le convertisseur A/D car nous utilisons des interruptions hautes pour les boutons **RB6** et **RB7** et nous voulons que ces boutons soient prioritaires.

Nous utilisons aussi le channel **AN4** qui correspond à **RA5** car c'est le seul disponible car nous avons besoin de **RA3:RA0** pour les 4 afficheurs 7-segments.

Grâce à notre interruptions sur le convertisseur A/D, nous pouvons récupérer la valeur du convertisseur A/D via **ADRES** qui est sur 10 bits. Nous avons donc le choix entre **ADRESH** et **ADRESL** qui sont juste des décalages de 2 vers la gauche ou vers la droite et nous allons choisir **ADRESH** car les 2 MSB du registre **ADRESH** sont beaucoup trop importants et les 2 bits de précision de **ADRESL** sont assez négligeables. Cela veut aussi dire que nous n'avons pas totalement 5V car il va manquer les 2 bits LSB de **ADRESL**.

Nous avons donc la valeur de notre convertisseur A/D qui est sur 8 bits donc entre 0 et 255. Il faudra donc changer sa plage de valeur entre 0 et 5, et pour cela nous utilisons un Quantum. Théoriquement ce Quantum est égal à :

$$quantum = \frac{5}{255} \approx 0.0196$$

Cependant en pratique et dans la réalité nous avons une tension maximale de 4.7-4.8V et pour obtenir un 5V parfait, nous avons donc décidé d'augmenter le Quantum à 0.0204919. Maintenant, nous pouvons multiplier ce Quantum à notre **ADRESH** ce qui va permettre d'obtenir des valeurs entre 0 et 5 en float.

Une fois que nous avons des valeurs entre 0 et 5 que l'on va appeler `voltage_value`, il va falloir le séparer en 4 digits pour pouvoir l'afficher sur les 4 afficheurs 7-segments. Et pour cela nous allons utiliser le fait qu'en castant un float en integer, nous allons ne récupérer que la valeur AVANT la virgule qui sera notre premier chiffre. Nous pouvons ensuite soustraire le int qui est notre premier chiffre que nous venons de récupérer à la valeur `voltage_value` et multiplier par 10 `voltage_value` pour obtenir le premier chiffre APRES la virgule qui sera notre deuxième chiffre. En répétant ce processus, nous pouvons séparer le float qui est entre 0 et 5 en 4 chiffres que nous allons pouvoir afficher sur les 4 afficheurs 7-segments.

Exemple de compréhension de la méthode :

Valeur de `voltage_value` : 2.535

Caster `voltage_value` en integer dans premier chiffre

Valeur de premier chiffre : 2

Effectuer l'opération `voltage_value = voltage_value - premier chiffre`

Valeur de `voltage_value` : 0.535

Effectuer l'opération `voltage_value = voltage_value * 10`

Valeur de `voltage_value` : 5.350

Puis on répète le calcul.

Il suffira de faire une condition pour transformer ces 4 chiffres integer en 4 valeurs hexadécimales ou binaires lisibles par le 7-segment.

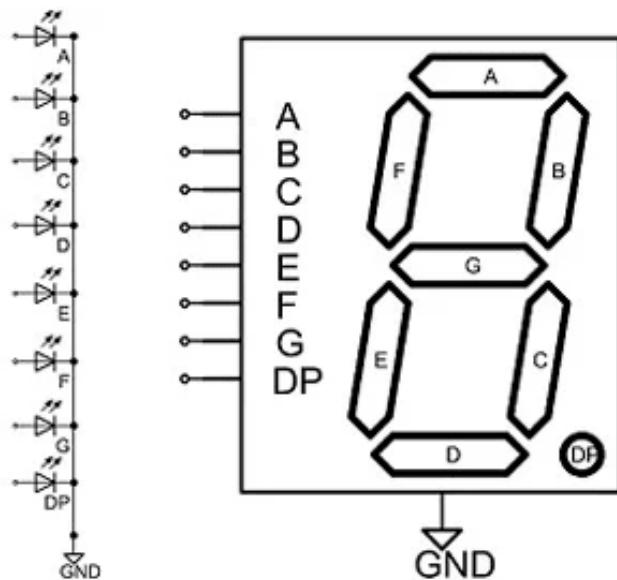


Figure 8: 7-segment display

Pour pouvoir afficher, sur un afficheur 7-segment, chaque valeur de A jusqu'à DP va être relié aux 8 bits du registre **PORTD** et cela pour chacun des 4 afficheurs 7-segments. Mais comment peut-on alors écrire différents chiffres sur plusieurs 7-segments avec que **PORTD** ? Nous pouvons alors utiliser un mélangeur pour pouvoir éteindre et allumer les afficheurs un par un et cela de manière imperceptible pour l'œil humain. Dans le registre **PORTA**, quand le bit 0 qui est **RA0** est allumé, on va pouvoir écrire pour le 7-segment **dis0** puis l'on va éteindre **RA0** et écrire sur **PORTD** avec **RA1** pour **dis1** qui est le deuxième 7-segment et ainsi de suite.

PORTD, affichage de 2 par exemple :

RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
DP	G	F	E	D	C	B	A
0	1	0	1	1	0	1	1

PORTA :

RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
-	-	-	-	0	0	0	1

C. Oscilloscope et trigger

En entrant dans l'oscilloscope, nous avons alors le mode 1 qui correspond au mode RUNNING et le mode 2 qui correspond au mode TRIGGER. Nous pouvons changer de mode avec interruption sur le bouton **RB6** et le bouton **RB7** va nous permettre de revenir au menu entre oscilloscope et signal rectangulaire. Nous pouvons aussi voir notre valeur de tension d'entrée **ADRESH** convertie entre 0 et 5V. Notre courbe va pouvoir s'afficher sur 99 pixels de longueur et 64 pixels de largeur donc 99x64.



Figure 9: Mode Running | Mode Trigger | Mode Trigger

En mode running nous avons un compteur dans la boucle d'affichage qui va s'incrémenter qui va correspondre à notre position x. Nous allons récupérer la valeur de **ADRESH** entre 0 et 255 et la convertir en la divisant par /4 pour obtenir une valeur entre 0 et 64 ce qui va correspondre à la position y. Maintenant que nous connaissons la position x et y, nous pouvons afficher la courbe. De plus, pour chaque courbe nous utilisons l'algorithme de Bresenham's qui permet de faire une courbe continue. En effet, cet algorithme permet de tracer une ligne entre 2 points peu importe que la ligne soit horizontale, verticale ou en diagonale. Elle va donc prendre 4 paramètres que sont x_1 , y_1 , x_2 et y_2 . Pour avoir une courbe continue sans trous, nous allons appliquer cet algorithme entre la valeur précédente du compteur (x_1), la valeur précédente de **ADRESH** convertie entre 0 et 64 (y_1) qui sera notre point précédent et notre point actuel qui est le compteur actuel (x_2) et la valeur de **ADRESH** actuelle convertie entre 0 et 64 (y_2). Une fois l'affichage effectué, nous pouvons dire que notre point actuel devient notre point précédent et incrémenter le compteur actuel.

Le temps ou la fréquence d'affichage va dépendre du temps d'écriture de pixel horizontal et non pas verticalement (notre courbe va de gauche à droite). Dans ce mode, nous pouvons afficher 99px en 5.40s ce qui va correspondre à :

$$\begin{aligned} \text{- pps(pixel per second)} &= 1 * 99 / 5.40 = 18.3\text{pps} \\ \text{- spp(second per pixel)} &= 1 * 5.40 / 99 = 0.054\text{spp} \end{aligned}$$

Nous pouvons aussi connecter un générateur basse fréquence pour tester notre oscilloscope (cf. Tests et validation)

En appuyant sur **RB6**, nous pouvons basculer en mode TRIGGER.

Dans le mode TRIGGER, nous pouvons choisir avec une interruption sur le bouton RB6 et le potentiomètre un seuil pour lequel lorsque notre valeur d'**ADRESH** convertie entre 0 et 5V dépasse la valeur seuil, nous pouvons afficher la courbe, sinon nous ne l'affichons pas.

Dans ce mode, nous pouvons afficher 99px en 9.20s ce qui va correspondre à :

$$\begin{aligned} \text{- pps(pixel per second)} &= 1 * 99 / 9.20 = 10.8\text{pps} \\ \text{- spp(second per pixel)} &= 1 * 9.20 / 99 = 0.093\text{spp} \end{aligned}$$

D. Signal rectangulaire

En entrant dans le mode signal rectangulaire, nous avons le mode 1 qui va permettre de modifier la fréquence de notre signal rectangulaire grâce à un potentiomètre et dans le mode 2, nous pouvons modifier le rapport cyclique. Ce signal rectangulaire est à la fois généré sur le registre **PORTC** avec le bit **RC1** et est affiché sur le GLCD. En modifiant la fréquence ou le rapport cyclique, nous modifions à la fois le signal généré et le signal affiché sur le GLCD. Nous pouvons afficher notre signal généré sur un oscilloscope et changer la fréquence et le rapport cyclique pour le voir en temps réel.

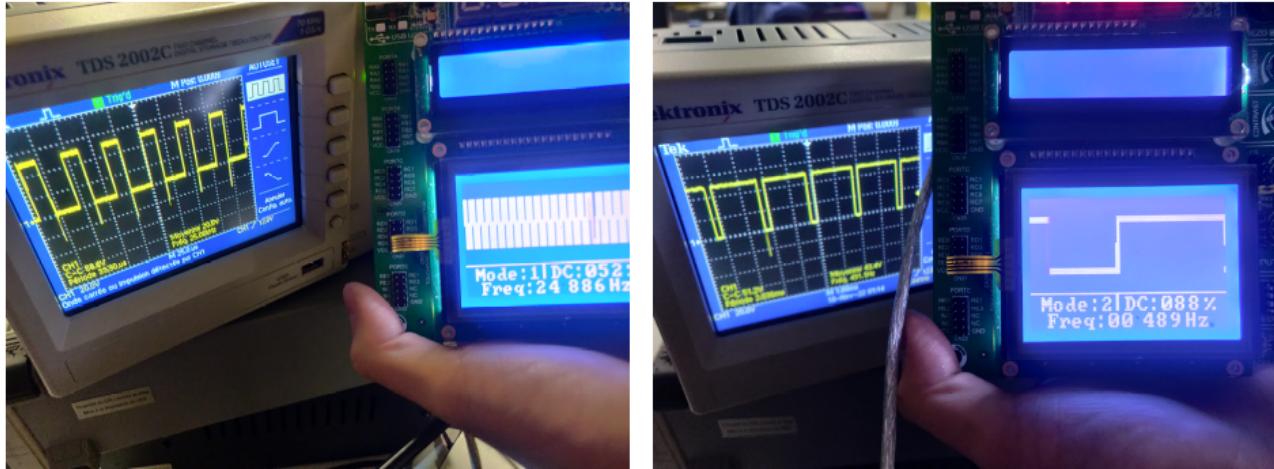


Figure 10: Signal rectangulaire

Dans le registre **TRISC**, nous pouvons mettre le bit 1 en mode OUTPUT en mettant "1" dans le bit **RC1**. Dans le registre **OSCON**, nous allons pouvoir mettre la fréquence de l'oscillateur interne à 8 MHz en mettant "111" dans les bits 6-4 qui sont **IRCF2:IRCF0**. Ensuite nous pouvons mettre l'horloge du système en oscillateur interne en mettant "10" dans les bits 1-0 qui sont **SCS1:SCS0**. Pour le CCP (capture/compare/PWM), nous allons utiliser **CCP2** qui correspond à **RC1** car **CCP1** qui correspond à **RC2** est aussi utilisé pour la luminosité de notre écran GLCD. Dans le registre **CCP2CON**, nous pouvons mettre notre CCP en mode PWM en mettant "1100" dans les bits 3-0 qui sont **CCP2M3:CCP2M0**. Dans le registre **T2CON**, nous allons pouvoir allumer le timer2 en mettant "1" dans le bit 2 qui est **TMR2ON**. Nous pouvons aussi mettre le plus grand prescaler pour avoir la plus petite fréquence possible pour que notre signal rectangulaire soit bien visible en mettant "11" dans les bits 1-0 qui sont **T2CKPS1:T2CKPS0**.

Le module CCP nous permet de générer des signaux rectangulaires en pulsant notre signal d'où le PWM. Il va allumer et éteindre très rapidement notre signal et nous allons aussi pouvoir modifier la fréquence qui correspond au nombre de motif qui se répète par seconde et le rapport cyclique qui correspond au temps où le signal est activé sur le temps total du signal. Pour modifier la fréquence, nous allons utiliser cette formule que nous allons mettre dans le registre **PR2** :

$$PR2 = \frac{Fosc}{F pwm * 4 * TimerPrescaleValue} - 1 = \frac{8000000}{F pwm * 4 * 16} - 1 = \frac{125000 - F pwm}{F pwm}$$

avec $F pwm$ qui va correspondre à notre fréquence. Pour faire varier cette fréquence, nous allons récupérer la valeur d'**ADRESH** qui est de 0 entre 255 et la convertir entre des valeurs comprises entre 490Hz et 12500Hz qui sont la valeur de fréquence minimale et maximale théorique. Ces valeurs sont obtenues en mettant $PR2 = 0$ et $PR2 = 255$ dans l'équation au-dessus et en récupérant la valeur de $F pwm$. Nous allons donc convertir notre valeur d'**ADRESH** entre 0 et 255 en 490Hz et 62500Hz qui est amplement suffisant.

Pour modifier notre rapport cyclique par rapport au potentiomètre, nous pouvons d'abord utiliser cette formule que nous allons mettre dans le registre **CCPR2L** :

$$CCPR2L = (PR2 + 1) * \left(\frac{duty\ cycle}{100} \right)$$

et de la même manière, nous pouvons récupérer la valeur d'**ADRESH** et la convertir entre 0 et 100.

TRISC :

TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
-	-	-	-	-	-	0	-

OSCCON :

IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0
0	1	1	1	0	0	1	0

CCP2CON :

-	-	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
-	-	0	0	1	1	0	0

T2CON :

-	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
-	0	0	0	0	1	1	1

Dans ce mode, nous pouvons afficher 18px en 14s ce qui va correspondre à :

- pps(pixel per second) = $1 * 128 / 14 = 9.1\text{pps}$
- spp(second per pixel) = $1 * 14 / 128 = 0.11\text{spp}$

VII. Tests et validation

A. Menu

Afin d'accéder aux différentes fonctionnalités, nous avons développé un menu qui utilise les interruptions sur RB6 et RB7. Ainsi en appuyant sur RB6 nous devrions pouvoir sélectionner les modes et avec RB7 nous devrions pouvoir valider le choix, grâce à une interruption externe. Testons cela sur proteus :

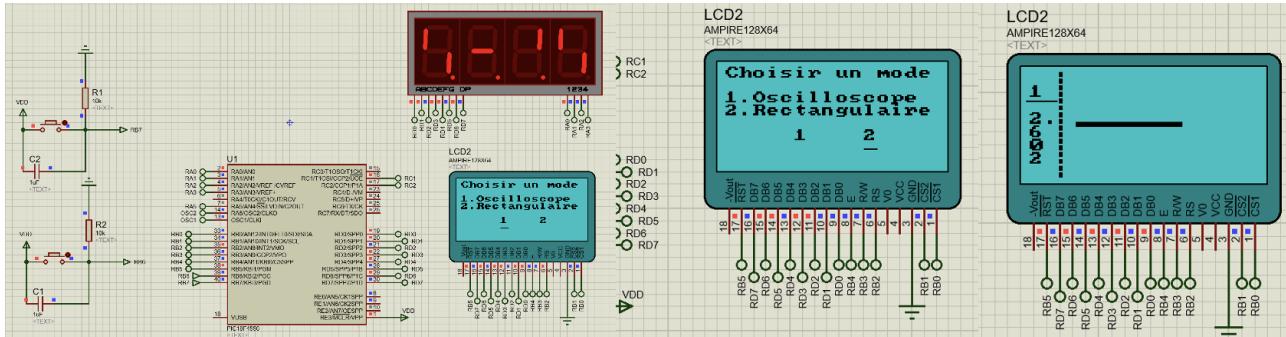


Figure 11: Simulation proteus menu

On observe bien le bon fonctionnement des fonctionnalités développées : avec RB6 nous pouvons sélectionner les modes et avec RB7 nous pouvons valider le choix. La simulation proteus est donc validée.

Testons cela en physique sur la board :



Figure 12: Simulation physique menu

On observe bien le bon fonctionnement des fonctionnalités développées comme précédemment. Nous pouvons donc valider ce module.

B. Affichage 7-segment

Dans un premier temps, nous réalisons une simple simulation proteus. Notre objectif est de voir si dans le menu, nous pouvons afficher clairement les 4 7-segment en "simultané" et qu'ils affichent une valeur comprise entre 0 et 5 V, avec une précision en mV (3 chiffres après la virgule), et si cela intervient périodiquement par interruption ADC :

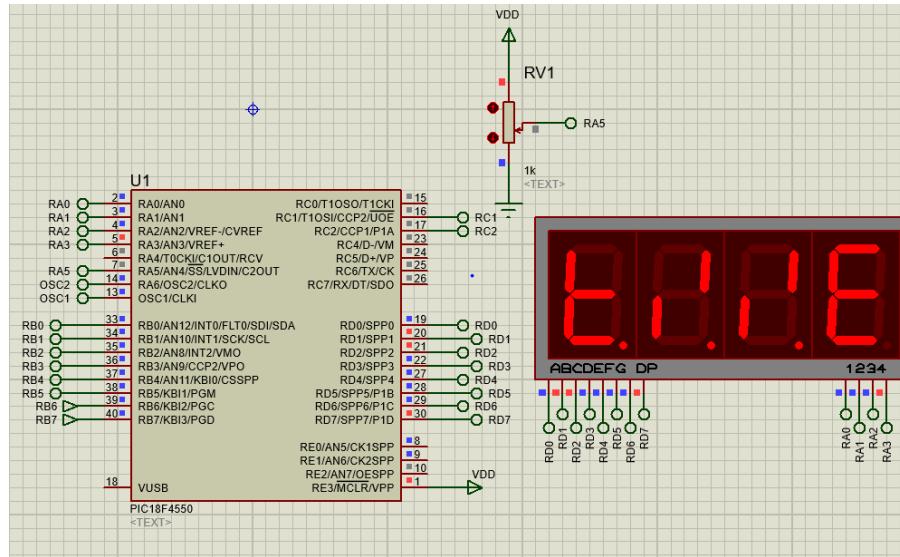


Figure 13: Simulation proteus 7-segment

Une simulation Proteus étant moins rapide qu'une simulation sur PIC, nous ne pouvons pas observer un résultat très concluant. Nous observons cependant que cela est presque fonctionnel. Passons à présent à la simulation directement sur l'easyPic board, qui devrait donner un résultat meilleur que le précédent :

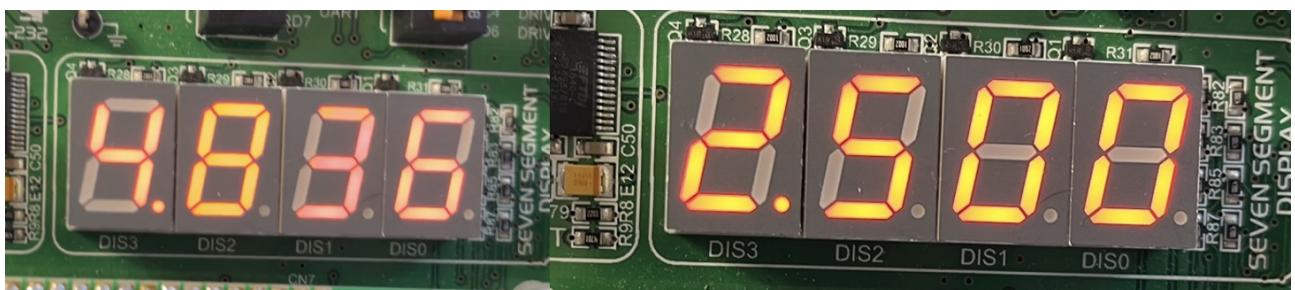


Figure 14: Simulation 7-segment physique

Comme prévu, le résultat que nous obtenons est encore plus précis que celui que nous obtenons sur Proteus. En fonction de la valeur du potentiomètre, nous observons bien la valeur de la tension analogique, en simultané, et avec une précision en mV. Enfin cette conversion s'effectue bien périodiquement par interruption sur ADC. Nous pouvons donc attester du bon fonctionnement de ce module.

C. Oscilloscope et trigger

Dans un premier temps, nous réalisons une simple simulation proteus et allons tester le premier des deux modes : l'affichage d'une courbe (tel un oscilloscope basique). Ainsi en bougeant le potentiomètre qui régit la valeur analogique lire nous devrions voir notre courbe évoluer sur l'écran. De plus, si la courbe dépasse la largeur de l'écran, nous devrions la voir revenir au début et progressivement effacer la trace de la précédente. Enfin nous devrions également voir affiché sur le côté gauche la valeur analogique lire par l'ADC. Simulons cela :

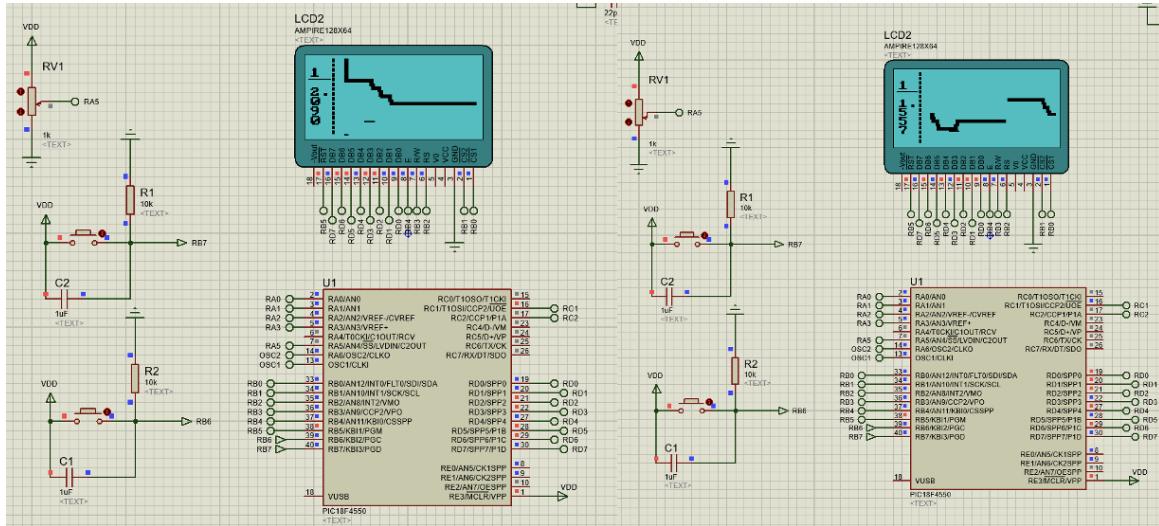


Figure 15: Simulation proteus mode 1 oscilloscope

Nous pouvons observer le résultat que nous voulions obtenir : la courbe s'affiche bien en fonction de la valeur analogique lire par le potentiomètre, et quand la courbe dépasse la largeur de l'écran, nous la voyons revenir au début et progressivement effacer la trace de la précédente. Nous pouvons également voir la valeur analogique affichée sur la gauche de l'écran. La simulation du premier mode est donc validée. Passons à la vérification physique sur la board, où nous devrions observer les mêmes résultats mais de manière plus fluide :

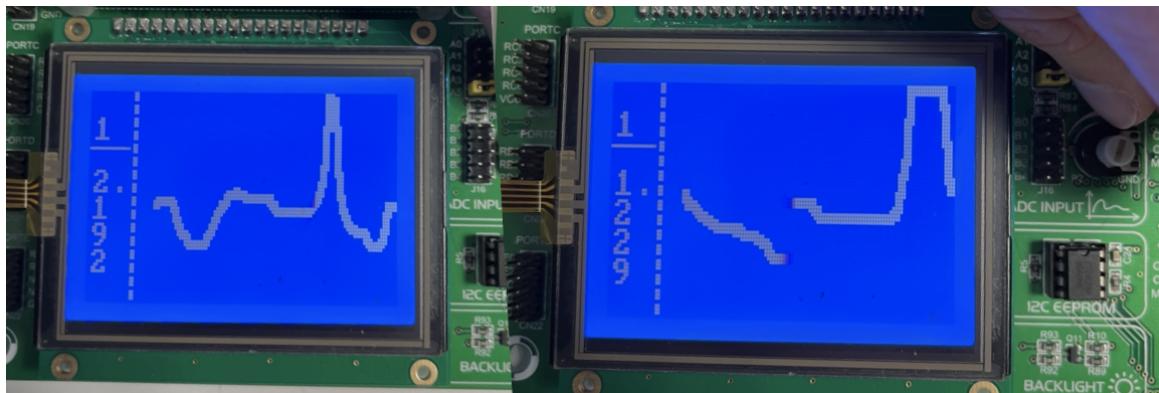


Figure 16: Simulation physique mode 1 oscilloscope

Nous pouvons observer le résultat que nous voulions obtenir, et de manière plus fluide qu'en simulation proteus. La simulation du premier mode est donc validée.

A présent, nous souhaitons également simuler ce mode en utilisant directement un GBF, relié à la pin qui lit la tension analogique. Nous devrions obtenir des résultats similaires à ceux précédemment évoqués et simulés à l'aide de notre potentiomètre. Voici notre résultat :

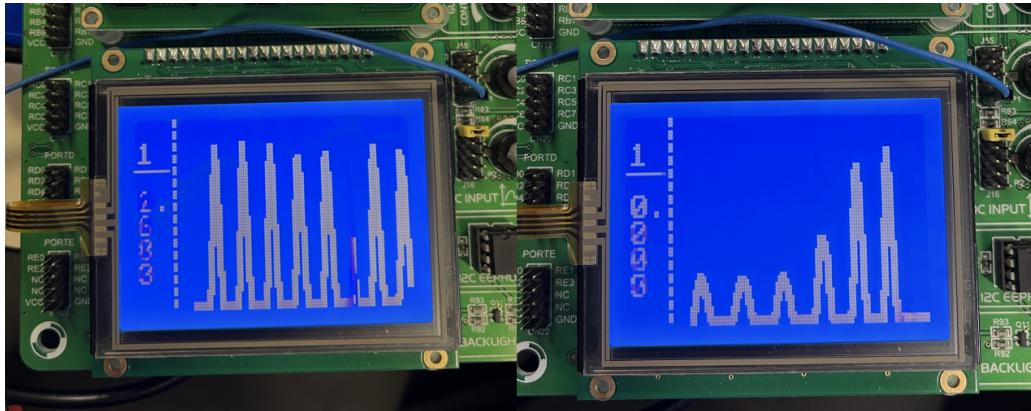


Figure 17: Simulation physique mode 1 oscilloscope avec GBF

Pour obtenir une courbe de ce style, nous avons généré un signal sinusoïdal d'1 Hz avec le GBF, relié à l'aide d'une pince crocodile et d'un fil à notre pin RA5. On retrouve bien les observations précédentes. Ce mode est donc validé.

A présent, nous voulons tester le mode trigger. En passant en mode trigger à l'aide du bouton RB6, nous devrions pouvoir sélectionner une valeur de trigger dans un premier temps à l'aide du potentiomètre, puis pouvoir valider avec RB6, et enfin dans le cas où la valeur analogique lue serait inférieure au trigger, la courbe ne devrait pas s'afficher et inversement si la valeur analogique lue serait supérieure au trigger, la courbe devrait s'afficher. Testons cela dans un premier temps sur proteus :

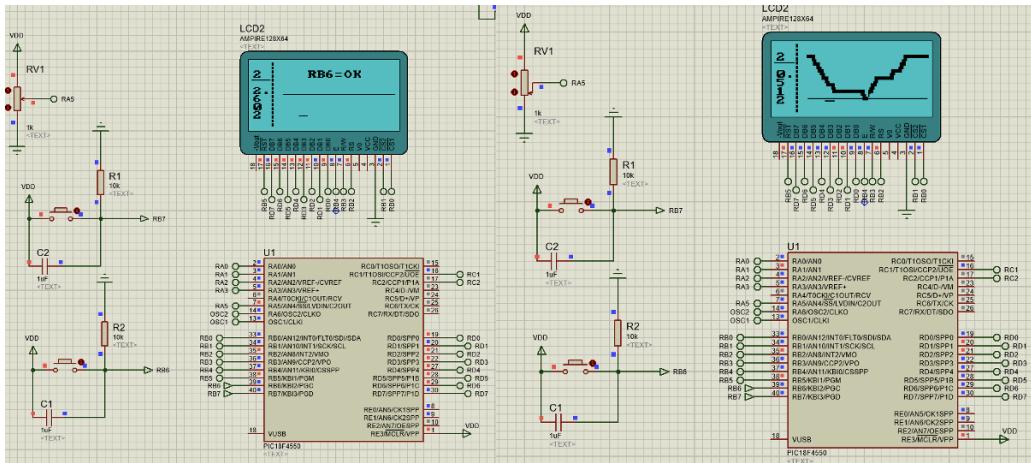


Figure 18: Simulation proteus mode 2 oscilloscope

On observe bien qu'en passant en mode trigger à l'aide du bouton RB6, nous pouvons sélectionner une valeur de trigger dans un premier temps à l'aide du potentiomètre, puis nous pouvons valider avec RB6. On observe ensuite que dans le cas où la valeur analogique lue est inférieure au trigger, la courbe ne s'affiche pas et inversement si la valeur analogique lue est supérieure au trigger, la courbe s'affiche. La simulation proteus est donc validée, passons au test en physique sur la board où nous devrions avoir les mêmes résultats mais de manière plus fluide encore.

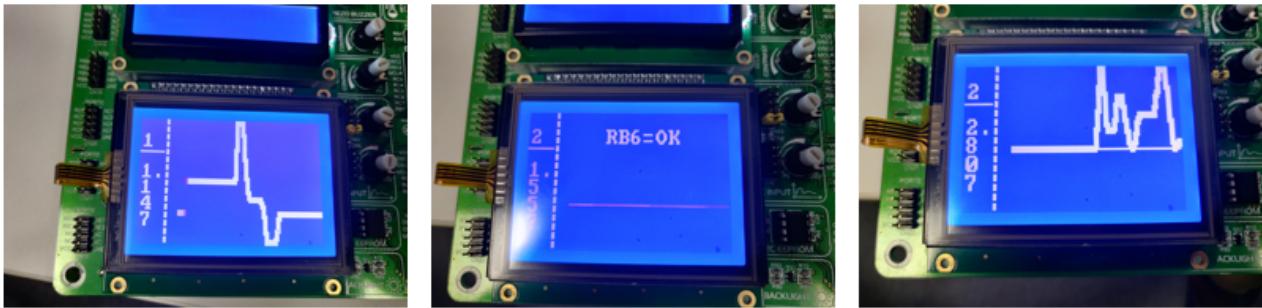


Figure 19: Simulation physique mode 2 oscilloscope

On observe bien les résultats escomptés. Enfin, si nous appuyons de nouveau sur RB6, nous devrions pouvoir changer de mode, et si nous appuyons sur RB7 nous devrions pouvoir retourner au menu à n'importe quel moment grâce à une interruption externe. Essayons cela :



Figure 20: Simulation physique interruption

On peut en effet naviguer dans le mode de cette manière. Les tests de ce module sont donc validés.

D. Signal rectangulaire

Dans ce dernier module, débutons par une simulation proteus, où nous devrions pouvoir sélectionner la fréquence avec le potentiomètre, valider avec RB6, puis sélectionner le rapport cyclique et valider avec RB6 et pouvoir observer la courbe en temps réel sur notre écran LCD. Avec une simulation sur oscilloscope, nous devrions également pouvoir observer notre courbe. Testons cela :

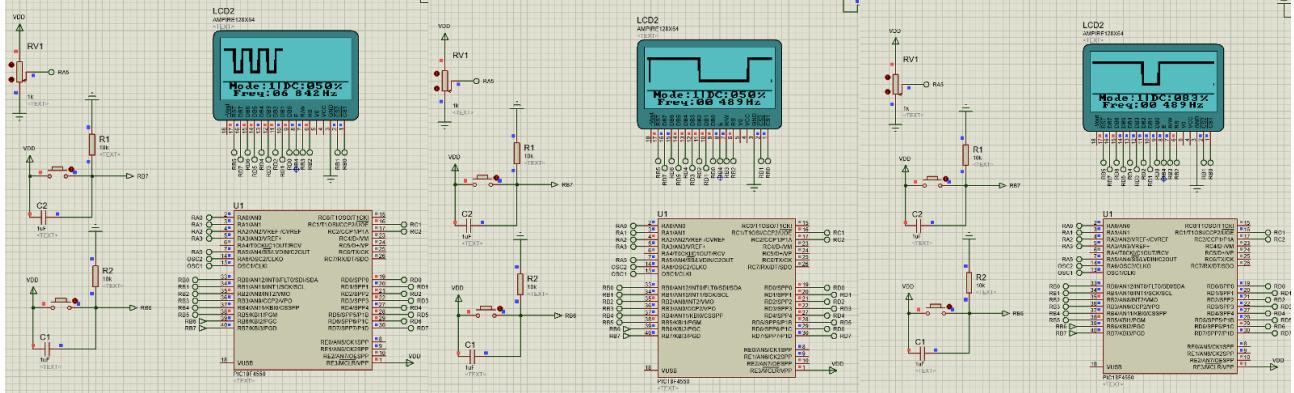


Figure 21: Simulation proteus signal rectangulaire

Nous pouvons en effet sélectionner la fréquence avec le potentiomètre, valider avec RB6, puis sélectionner le rapport cyclique avec le potentiomètre et valider avec RB6 et pouvoir observer la courbe en temps réel sur notre écran GLCD. Enfin notre courbe est bien générée et apparait comme signal rectangulaire sur l'oscilloscope. Testons cela sur notre board :



Figure 22: Simulation physique signal rectangulaire

Nous observons les mêmes résultats qu'en simulation. Testons également avec un oscilloscope :



Figure 23: Simulation physique signal rectangulaire avec oscilloscope

Nous observons les résultats escomptés, nous générions bien un signal rectangulaire et pouvons modifier son rapport cyclique et la fréquence. Le module est donc validé.

VIII. Bilan

A. État d'avancement

Comme présenté dans la partie tests et validation, les quatre modules de ce projet sont fonctionnels. Le menu principal permet de choisir entre le mode oscilloscope et le mode signal rectangulaire, puis de modifier l'affichage sur l'écran pour correspondre au choix de l'utilisateur. On affiche la valeur de la tension analogique (entre 0 et 5V) sur les 7-segment displays.

Ensuite, lorsque l'on entre dans le mode oscilloscope, on affiche la courbe du signal correspondant à la tension lue via RA5. Dans ce même mode, un bouton (RB6) nous permet de choisir entre un mode running (signal affiché en continu) et un mode trigger (signal affiché à partir d'une certaine valeur). On a aussi le bouton RB7 pour revenir au menu. Enfin, lorsque l'on rentre en mode signal rectangulaire, on peut générer un signal rectangulaire, sélectionner ses paramètres (fréquence et duty cycle) avec le bouton RB6 et les modifier avec le potentiomètre. Nous pouvons afficher la courbe du signal rectangulaire généré sur l'écran GLCD.

Nous pensons avoir répondu à la majorité des attentes du cahier des charges. Les trois modules principaux (sans compter le menu) sont fonctionnels.

B. Pertinence de la solution technique

Concernant les points à améliorer, nos 7-segment displays clignotent lorsque l'on affiche en même temps sur les 7-segment et sur le GLCD (l'oscilloscope). Nous aurions aimé obtenir un résultat plus performant pour un meilleur confort visuel. De plus, la fréquence que nous pouvons choisir lorsque l'on génère le signal rectangulaire est limitée. Par exemple pour une fréquence de 30 kHz et un duty cycle de 50 pourcent, l'écran affiche un rectangle blanc car le signal est trop rapide. Globalement, avec plus de temps et moins de projets à rendre à la fois, nous aurions sûrement implémenté quelques uns des bonus pour obtenir un design plus abouti. Nous sommes néanmoins très satisfaits du résultat actuel.

C. Bilan sur le travail d'équipe

Ayant travaillé ensemble pour de nombreux autres projets, nous avons eu une très bonne entente au sein du groupe. Nous avons réussi à travailler de manière organisée et efficace pour atteindre les objectifs attendus du cahier des charges. Ce projet nous a permis de mettre en application les compétences acquises en cours de microcontrôleur. Nous avons pu réutiliser nos connaissances en C et en électronique pour mener à bien le projet.

Pour nous organiser, nous avons réalisé de nombreuses réunions, profitant de notre temps en présentiel à l'ECE, et avons utilisé des moyens de communication tels que Discord et Messenger. Nous avons utilisé Github pour pouvoir partager nos avancées sur le code de manière plus efficace. Cependant, nous aurions pu améliorer nos temps de disponibilité en se consacrant plus tôt sur ce projet (dès les vacances), et en n'hésitant pas à poser plus de questions à l'équipe pédagogique.

C'est pourquoi pour le prochain projet, nous réaliserons sûrement des réunions de lancement plus conséquentes afin de séquencer directement le projet et tenter de le jaloner, pour mieux s'organiser dans le temps et nous tacherons de plus nous référer à l'équipe pédagogique.

IX. Sources

Voici nos sources pour le projet EasySCOPE.

- Site WEB, « How to code your first algorithm », ghost-together.medium.com :
<https://ghost-together.medium.com/how-to-code-your-first-algorithm-draw-a-line-ca121f9a1395>
- Site WEB, « PIC18F4550 et PWM », electronicwings.com :
<https://www.electronicwings.com/pic/pic18f4550-pwm>
- Site WEB, « PWM using pic », openlabpro.com :
<https://openlabpro.com/guide/pulse-width-modulation-using-pic18f4550/>
- Documents fournis dans l'onglet projet de la page microcontrôleur.
- Datasheet PIC18F4550 fournie sur Boostcamp.

X. Annexes

Documents annexes, éventuels codes (**pas de code dans le rapport**).