

Compte rendu Projet Microcontrôleur

Baclet Anne-Charline Biau-Guillaumet Marie
Griet Thibaud Guerry Camille

Juin 2016

Table des matières

I Périphériques	4
1 Écran tactile	4
1.1 Introduction	4
1.2 Le contrôleur écran SSD1289	4
1.2.1 Le fonctionnement	4
1.2.2 L'implémentation	5
1.3 Le contrôleur tactile XPT2046	7
1.3.1 Le fonctionnement	7
1.3.2 L'implémentation	7
2 DéTECTEUR ultra-son	8
2.1 Introduction	8
2.2 Ports mis à disposition	8
2.3 Génération d'ultrasons	9
2.4 Récupération de l'écho	10
3 Bipper	12
3.1 Introduction	12
3.2 Génération du signal carré	13
3.2.1 Problème de l'attente active	13
3.2.2 Implémentation à l'aide de flags	13
3.3 Test en simulation	13
4 Émetteur-recepteur intrarouge	14
4.1 Introduction	14
4.2 Emission de trames	14
4.3 Test en simulation	16
4.4 Test sur la carte	17
4.5 Réception	17
II Application d'ouverture à distance d'un verrou	20
5 Introduction	20
6 Ordonnancement des tâches	20
7 Fonctionnement de l'application	20
7.1 Saisie du code sur l'écran tactile	20
7.2 Ouverture par passages devant le module ultra-son	21

Introduction

Notre objectif est de réaliser une application qui permet l'ouverture d'un verrou à distance. Un écran tactile permet la saisie d'un code à quatre chiffres afin d'ouvrir le verrou. Il est aussi possible de le déverrouiller en réalisant une séquence précise de passages devant un émetteur-récepteur ultra-son.

Première partie

Périphériques

1 Écran tactile

1.1 Introduction

Le déverrouillage du verrou peut s'effectuer par saisie d'un code sur écran tactile. La programmation s'effectue principalement en deux étapes : une étape pour paramétriser l'écran et une deuxième pour pouvoir récupérer l'appui d'un utilisateur sur l'écran tactile.

1.2 Le contrôleur écran SSD1289

1.2.1 Le fonctionnement

Le contrôleur écran SSD1289 envoie des données grâce à un bus parallèle 8 bits.

P2.0	P2[0]	D0
P2.1	P2[1]	D1
P2.2	P2[2]	D2
P2.3	P2[3]	D3
P2.4	P2[4]	D4
P2.5	P2[5]	D5
P2.6	P2[6]	D6
P2.7	P2[7]	D7
P1.25	P1[25]	EN
P1.24	P1[24]	DIR
P1.23	P1[23]	LE
P0.21	RD1	RD
P0.20		RS
P0.23		WR
P0.22		CS

FIGURE 1 – Table des PINOUT utilisés par le contrôleur écran

Les données sont contenues dans les pins D0-D7 qui sont en Input/Ouput.

DC est un signal qui permet de choisir si on envoie des données ou une commande et de sélectionner le registre à utiliser. Il est nommé RC dans les fichiers. R/W permet de choisir si l'on est en mode lecture ou écriture. GDDRRAM est une appellation qui fait référence à la mémoire graphique. IR est le registre qui

contient le numéro du registre auquel on veut accéder. AC est le registre qui contient l'adresse où l'on veut écrire sur l'écran. Il s'incrémente lors du passage d'un pixel à un autre.

L'écran de 32" a pour dimension 240x320 pixels et les couleurs sont gérées en RGB.

1.2.2 L'implémentation

Les bibliothèques mises à disposition afin de programmer l'affichage du clavier 10 touches sont :

- ili_lcd_general pour le paramétrage de l'écran
- english_16x8 contenant les pixels afin de dessiner un caractère (16x8 pixels)
- lcd_api pour l'écriture sur l'écran

Le clavier implémenté possède les chiffres de 0 à 9 ainsi qu'une touche de validation du code et une touche d'effacement arrière.

L'initialisation de l'écran s'effectue par l'intermédiaire d'une fonction initialisation_ecran qui initialise les variables nécessaires à la programmation, utilise la fonction de bibliothèque lcd_Initializtion pour configurer le contrôleur écran et les pins puis effectue l'affichage numérique grâce à la fonction affichage. Cette dernière affiche un fond de couleur Cyan sur l'écran puis écrit les différentes touches et enfin trace les traits permettant de délimiter les touches. L'affichage se fait par l'intermédiaire de deux fonctions de bibliothèque nommées LCD_write_english et LCD_write_english_string.

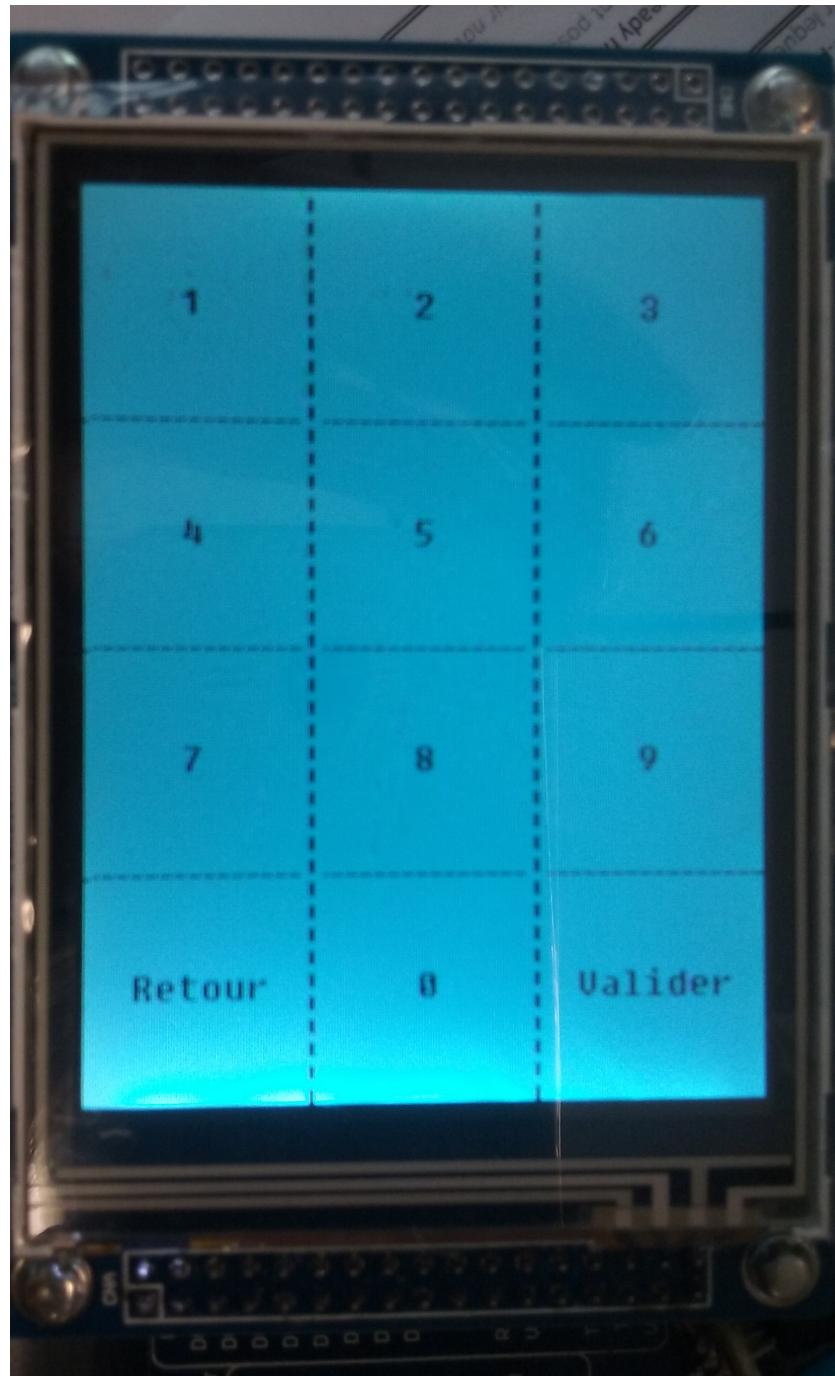


FIGURE 2 – Affichage sur l'écran tactile

1.3 Le contrôleur tactile XPT2046

1.3.1 Le fonctionnement

Le contrôleur tactile XPT2046 envoie des données grâce à un bus SPI.

P0.6		T_CS
P0.9		T_DI
P0.8		T_DO
P0.7	SCK1	T_SCK
P0.19		T_IRQ

FIGURE 3 – Table des PINOUT utilisés par le contrôleur tactile

Lorsque l'on utilise un bus SPI, afin de pouvoir recevoir des données il faut auparavant en émettre. Les échanges se font de la manière suivante : 8 bits sont émis puis après un top d'horloge on en reçoit 12. Ces 12 bits seront reçus en 2 fois, ce qui explique les décalages nécessaires.

T_CS est géré par le GPIO. Quand il est à 0, le bus SPI gère les données, et quand il est à 1, il ne se passe plus rien (ni envoi ni réception de données). T_DI, T_DO et T_SCK sont utilisés pour gérer le bus SPI. La fonction qui nous intéresse est la fonction 2 car elle permet de récupérer les coordonnées x et y de l'appui sur l'écran tactile. T_IRQ, qui permet de gérer l'interruption, ne sera pas utilisé.

Pour notre application, il faut mettre le bit SER/DFR à 0 ; c'est le mode de référence. S est le bit de start. Il faut le mettre à 1 pour commencer le dialogue.

	A2	A1	A0
x	0	0	1
y	1	0	1

Le bit A permet de sélectionner la coordonnée à récupérer. Pour avoir un codage sur 12 bits, on met le bit MODE à 0. Les bits PD1-PD0 sont à 00, ce qui correspond au mode "power-down between conversions", qui permet d'utiliser moins d'énergie.

1.3.2 L'implémentation

Les bibliothèques mises à disposition afin de programmer la récupération des entrées utilisateur sont :

- touch_panel pour le paramétrage du tactile et la récupération des coordonnées d'appui
- isr_touch pour programmer l'interruption (non utilisé ici)

Tout ce qui concerne l'écran sera exécuté dans le main du programme, sans aucune interruption, car les fonctions de bibliothèque contiennent des primitives bloquantes.

L'interface implémentée permet à l'utilisateur de rentrer un code sur l'écran. Le code doit être rentré sous 15 secondes (durée pendant laquelle le bipper est actif). Si le délai est dépassé, l'écran et ses variables se remettent à zéro et l'utilisateur peut à nouveau entrer un code. L'utilisateur doit appuyer sur le bouton valider à la fin de la saisie. Un code ne peut être rentré si le bouton poussoir de l'ultrason est enclenché.

L'initialisation du tactile s'effectue par l'intermédiaire d'une fonction initialisation_ecran qui initialise les variables nécessaires à la programmation et utilise la fonction de bibliothèque touch_init pour configurer le contrôleur tactile et les pins. On utilise le timer 0 afin de ne récupérer les appuis que toutes les 0,05 secondes. Une durée plus courte surchargerait le main et serait inutile car un appui utilisateur est une action relativement longue.

Si un appui est détecté, alors on récupère la donnée de l'appui grâce à la fonction update_password. La fonction update_password permet de stocker dans la variable tableau password le chiffre entré à l'écran. S'il s'agit de la touche "retour", alors on décrémente la variable qui nous sert de curseur pour entrer les chiffres dans le tableau password. S'il s'agit de la touche "valider", alors la fonction valider s'exécute et on affiche soit un message de déverrouillage (si le code entré est '1234') soit un message d'erreur et on réinitialise l'écran et ses variables. Si le code est correct alors on active l'infrarouge pour interagir avec le verrou.

Pour chaque appui, les coordonnées utilisées correspondent à la moyenne de 10 relevés auquels on a enlevé le minima et le maxima. Cela permet d'obtenir des coordonnées plus précises que si l'on effectuait simplement un unique relevé.

2 DéTECTEUR ultra-son

2.1 Introduction

L'ultrason HC-SR04 est un appareil permettant d'émettre des ultrasons de l'ordre de 40 kHz et d'en capturer le retour (appelé ici écho). L'émission nécessite cependant quelques conditions pour bien fonctionner. En effet, les ultrasons ne peuvent être créés que lorsqu'une tension de 5V, pendant une durée de l'ordre d'une dizaine de microsecondes, apparaît sur la borne «Trigger». De plus chaque émission doit être espacée de 100ms de la précédente afin de laisser un temps convenable pour que l'écho puisse revenir. Le signal alors reçu devra être traité pour que le verrou s'ouvre à 3 passages de main. Pour finir, le fonctionnement de l'ensemble doit s'effectuer que lorsque le bouton poussoir est appuyé.

2.2 Ports mis à disposition

Afin de réaliser la programmation du module, trois ports devaient être configurés :

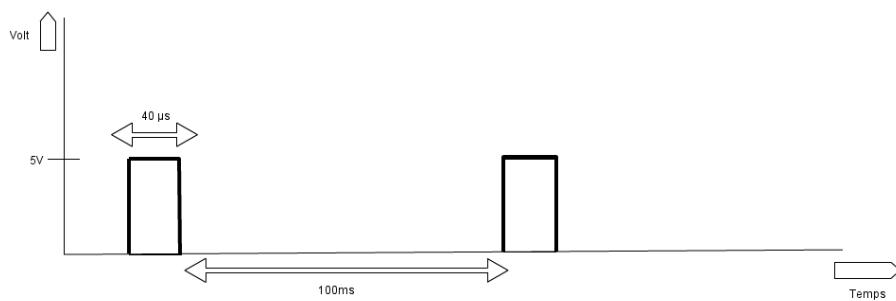
- le port GPIO 0.24 qui est relié à la broche «Trigger» de l'ultrason
- le port GPIO 0.25 qui est relié à la broche «Echo» de l'ultrason
- le port GPIO 0.26 qui est relié au bouton poussoir de l'appareil

2.3 Génération d'ultrasons

Dans cette partie, nous allons nous intéresser au port 0.24 qui permet de commander l'émission. Cependant, l'activation du port nécessite une configuration du port 0.26 afin que, lorsque le bouton poussoir est en position basse, le trigger se lance (et lorsqu'il est en position haute, il ne se passe plus rien).

Voilà pourquoi le port 0.26 est en pulldown. Un flag nommé bouton_appui annonce si ou ou non quelqu'un appuie dessus. Ce flag est essentiel pour tout le reste du code. C'est grâce à lui que toutes les autres procédures se lancent ou s'arrêtent.

Si ce flag vaut 1 alors la broche du port 0.24 se retrouve alimentée. A ce moment là, le but est de configurer le port comme suit :



Il faut donc manipuler le GPIO afin d'obtenir un tel signal. Cependant, obtenir un front montant de quelques microsecondes est assez délicat. Voilà pourquoi il était préférable d'émettre une tension pendant environ une quarantaine de microsecondes plutôt que 10 comme il était stipulé dans le cahier des charges.

Une base de temps s'incrémentant tous les $2 \mu\text{s}$ va être utilisée pour permettre la réalisation de ce trigger.

Il est à noter que dans cette partie du code, aucune interruption GPIO est utilisée. Deux flags signalant la fin des $40 \mu\text{s}$ et la fin des 100ms permettent de mettre à jour le pin à 1 ou 0 sur la broche 0.24.

Une fois une période passée, les compteurs se réinitialisent et redémarrent. De ce fait nous obtenons le signal suivant :

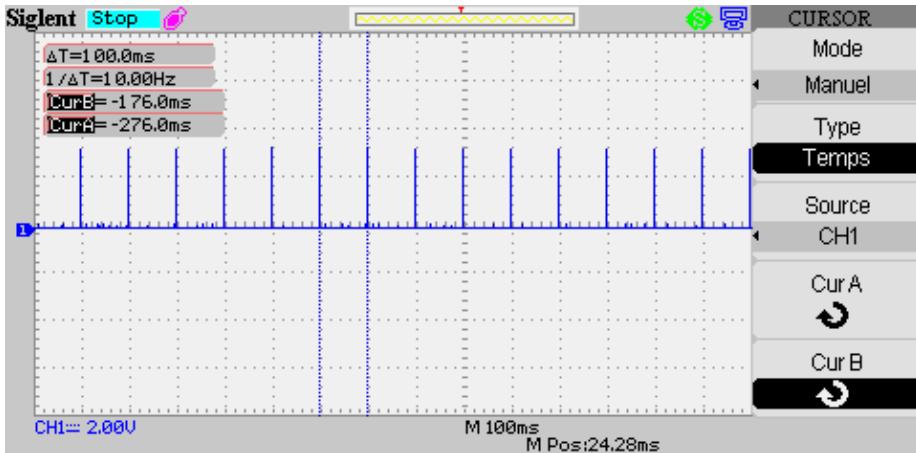


FIGURE 4 – Signal sur le port GPIO 0.24

Si nous zoomons davantage sur une pulsation, nous obtenons le signal ci-dessous :

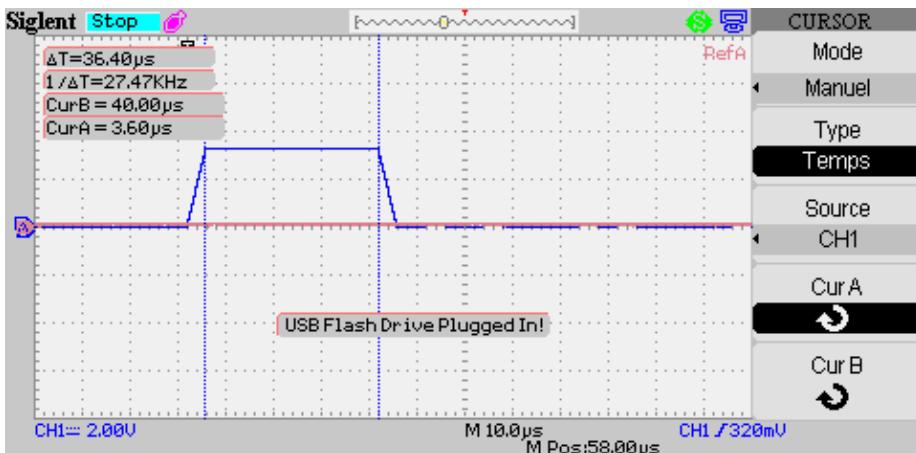


FIGURE 5 – Agrandissement du signal sur une pulsation

Il est à remarquer que nous n'obtenons pas exactement la période de $40\ \mu\text{s}$ souhaitée. Cela démontre bien qu'il est assez compliqué de programmer précisément une période en terme de microsecondes.

2.4 Récupération de l'écho

La génération de Trigger permet alors au HC-SR04 d'envoyer automatiquement des ultrasons et également d'en récupérer l'écho. Bien que la récupération de l'écho se fasse automatiquement et ne nécessite pas de configurer le port 0.25 (seulement le paramétrier en entrée), il est nécessaire désormais d'analyser ce retour dans le but de savoir si oui ou non les ultrasons ont rencontré un obstacle à une distance définie et si cet obstacle réapparaît plusieurs fois.

Le signal récupéré sur le port 0.25 présente plusieurs front montants espacés d'une même période. La représentation de la distance de l'objet par rapport à l'appareil apparaît alors comme un élargissement entre fronts montant et descendant si l'objet est éloigné ou comme un rétrécissement si l'objet est près.

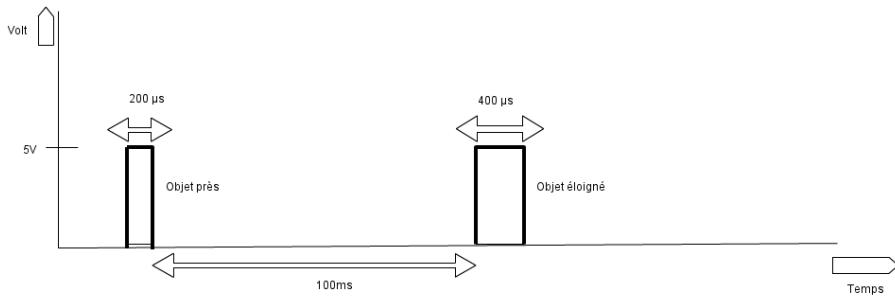


FIGURE 6 – Exemple d'écho reçu sur le port0.25

Pour déverrouiller le verrou, 3 passages de main devant l'écho suffise. Il faut donc établir une distance à laquelle la main devra passer devant l'appareil. Pour cela, j'ai choisi arbitrairement 10 cm. Seulement, il est extrêmement difficile voire impossible de passer sa main trois fois à exactement 10 cm du HC-SR04. De ce fait, une marge de 2cm de part et d'autre est essentiel pour permettre le déverrouillage. L'utilisateur devra alors passer sa main dans un intervalle compris entre 8 et 12cm.

En terme de temps, cela correspond à $t = \frac{340 \times 8 \times 10^{-3}}{2}$ Soit pour 8cm cela donne $406 \mu\text{s}$ et pour 12 cm on a $754 \mu\text{s}$.

Pour un objet se situant à environ 10 cm, on obtient sur l'oscilloscope le signal suivant :

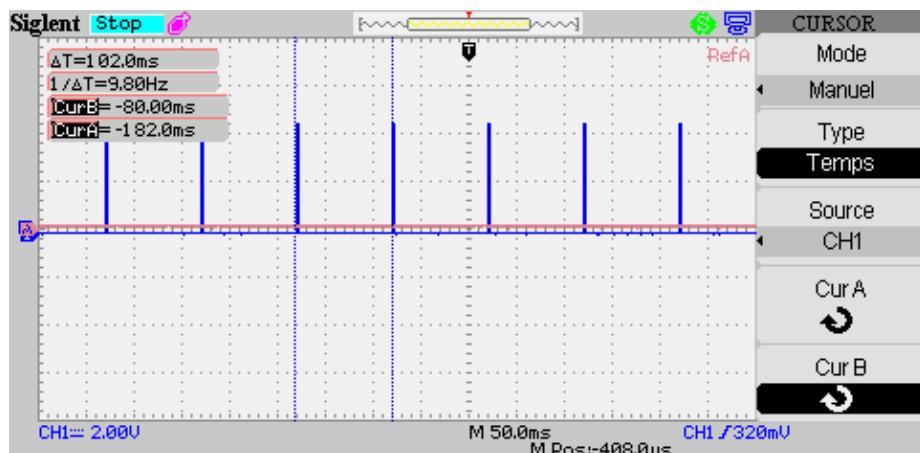


FIGURE 7 – Signal de l'écho récupéré sur le port0.25

Si l'on zoom sur une pulsation, on obtient le signal suivant :

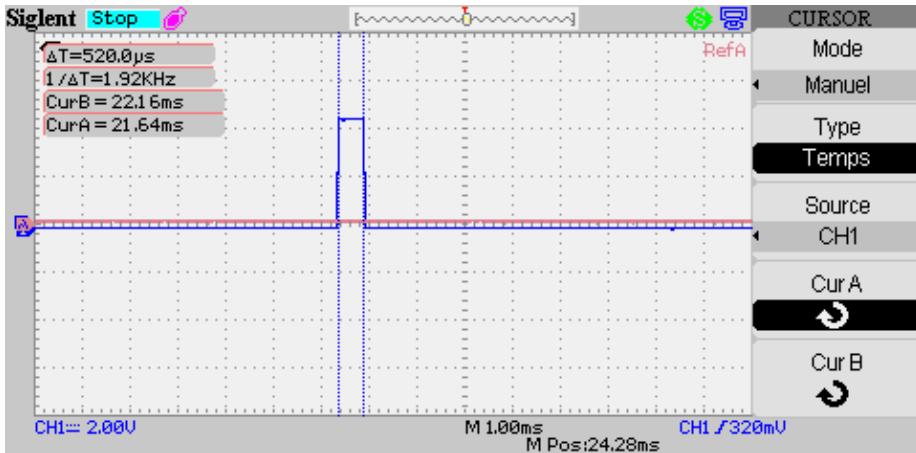


FIGURE 8 – agrandissement du signal sur une pulsation

D'après ce signal, l'objet est bien compris entre 8cm et 12cm ce qui devrait valider un passage de main.

Pour savoir combien de passages il y a eu au total, une variable open de type entier est mise à disposition. Elle s'incrémente dès que la main se trouve dans l'intervalle requis. Au bout de 3, cela est censé déverrouiller le verrou.

Néanmoins, un problème s'impose. En effet, chaque pulsation du trigger (et donc de l'écho) est seulement espacée de la suivante de 100ms, ce qui représente un temps extrêmement court si ce n'est que pour passer la main et la relever. De ce fait, avec un seul passage, le compteur open se mettra automatiquement à 3.

Il nous faut donc faire en sorte qu'il ne capte pas deux fois le même signal à la suite dans l'intervalle de validation. En effet, si on capte deux fois (ou plus) à la suite un signal dans l'intervalle de validité, cela voudrait dire que l'utilisateur n'a pas relevé la main. Mais, si le signal actuel est dans l'intervalle de validité et le signal précédent hors de ce dernier, alors cela signifierait forcément que la main s'est dégagée à un moment donné.

Voilà pourquoi une condition supplémentaire est implémentée dans la procédure de l'écho. Une variable passage recueille et stocke la valeur de l'écho précédent et la compare avec la valeur actuelle.

3 Buzzer

3.1 Introduction

Une tentative d'ouverture du verrou doit se faire en un temps limité. Afin de matérialiser cet intervalle de temps pendant lequel un utilisateur peut tenter d'ouvrir le verrou, on émet grâce à un haut-parleur une séquence de "bips" pendant 15 secondes. Passé ce délai, on considère la tentative d'ouverture comme un échec.

3.2 Génération du signal carré

3.2.1 Problème de l'attente active

Afin de produire une série de "bips" avec le haut-parleur, il faut tour à tour mettre sur la broche P1.9 un niveau 0 et un signal carré (d'une fréquence de 400 Hz environ).

Pour ce faire, la solution la plus simple est d'utiliser trois boucles "while" imbriquées. Mais ici, on ne peut utiliser cette solution car ce type de boucle est bloquant. Son utilisation empêcherait donc de réaliser d'autres tâches, notamment la saisie du code.

3.2.2 Implémentation à l'aide de flags

Afin de résoudre ce problème, utilisez différents flags commandant la durée totale de la séquence de "bips", la durée d'un bip ainsi que le durée d'une période. Ces différents flags sont mis à jour si besoin dans la routine d'interruption du Timer 0, déclenchée toutes les $2\mu s$ environ. La mise d'un 1 ou d'un 0 sur P1.9 est ensuite réalisée hors interruption par les fonctions bip() et période() en fonction des valeurs de ces flags.

3.3 Test en simulation

On peut visualiser le signal obtenu en simulation. On peut ainsi vérifier qu'on a bien une série de "bips" régulièrement espacés, d'une durée totale de 15 secondes :



FIGURE 9 – Séquence de bips de 15 secondes

En zoomant sur un des "bips", on peut vérifier qu'on a bien un signal carré d'une fréquence d'environ 400Hz :

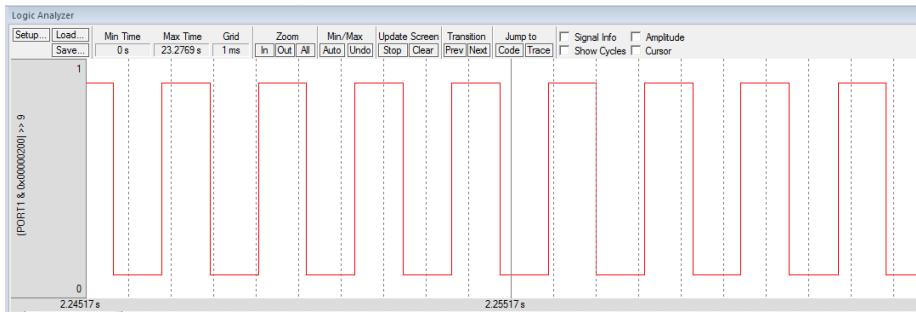


FIGURE 10 – Séquence de bips de 15 secondes

4 Émetteur-recepteur intrarouge

4.1 Introduction

La carte communique avec le verrou par liaison infrarouge. Pour établir cette communication un récepteur infrarouge TSOP4838 est utilisé. Le récepteur détecte les signaux émis à une fréquence de 38kHz et transforme le signal modulé à 38kHz en un signal logique : → une émission est transformée en «0» → une non émission est transformée en «1». L'objectif est donc d'émettre un signal modulé à 38kHz correspondant à un message qui respecte un protocole d'envoi précis et de récupérer un message respectant ce même protocole.

4.2 Emission de trames

L'envoi de trames est géré par la procédure void emission(). Cette procédure est appelée dans le main lorsque l'on désire émettre. Le message à envoyer est stocké dans une variable globale. Le Timer3 est configuré pour émettre des trames de 38kHz avec une précision de $1\mu s$. L'intervalle de temps à respecter entre chaque émission est mesuré grâce à une base de temps de $2\ \mu s$ commune à l'ensemble du projet et codée sur le timer0.

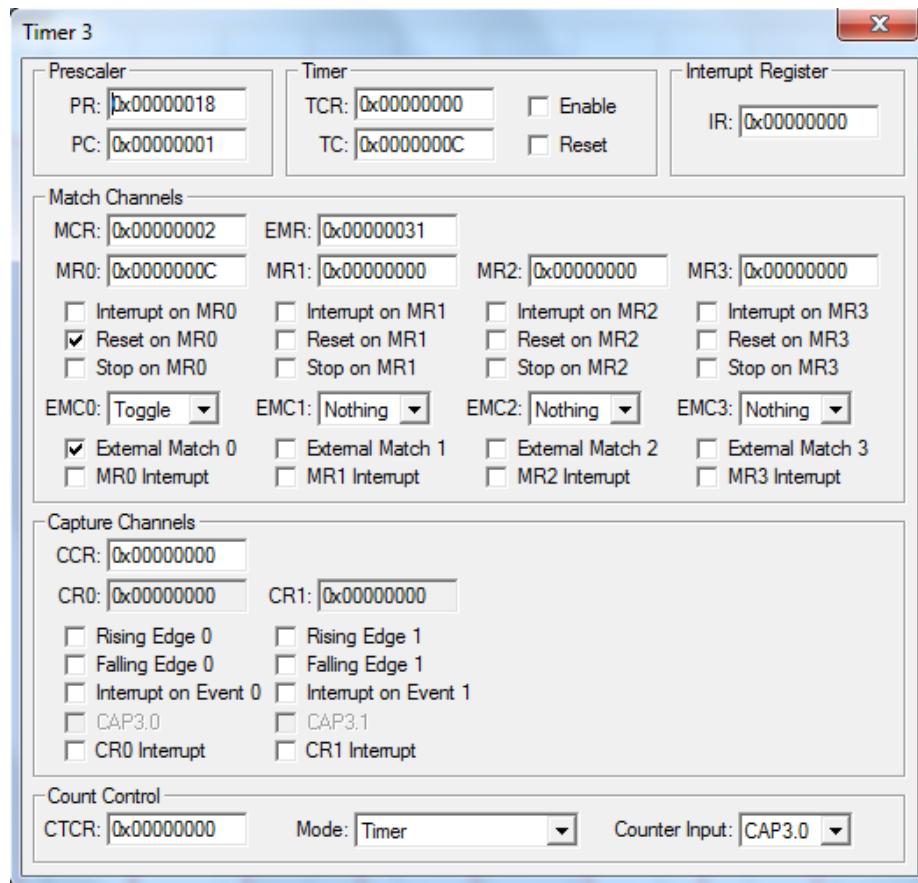


FIGURE 11 – Configuration du Timer 3

4.3 Test en simulation

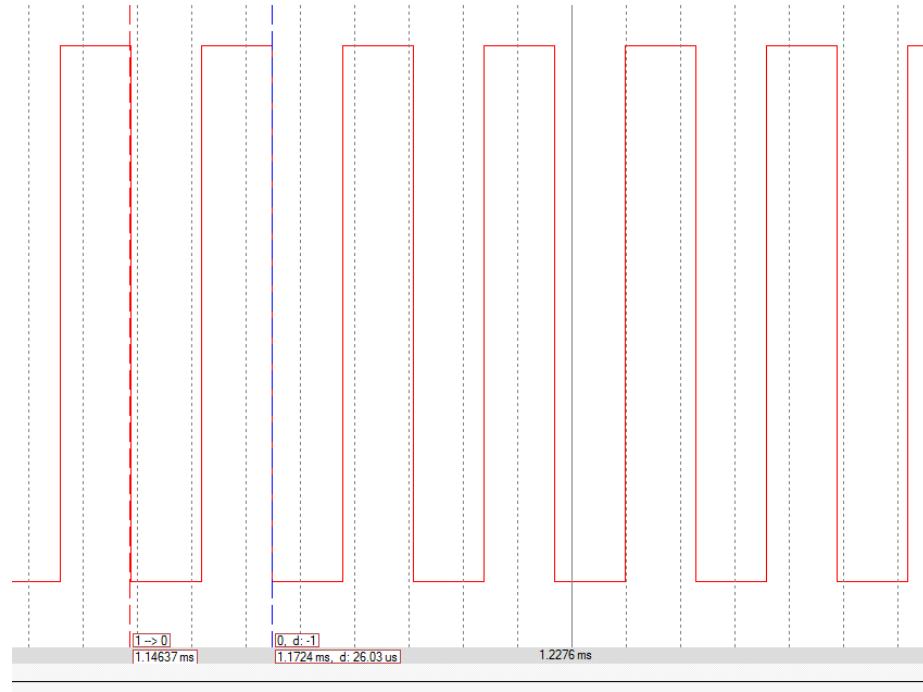


FIGURE 12 – Signal de 38kHz émis par le Timer3

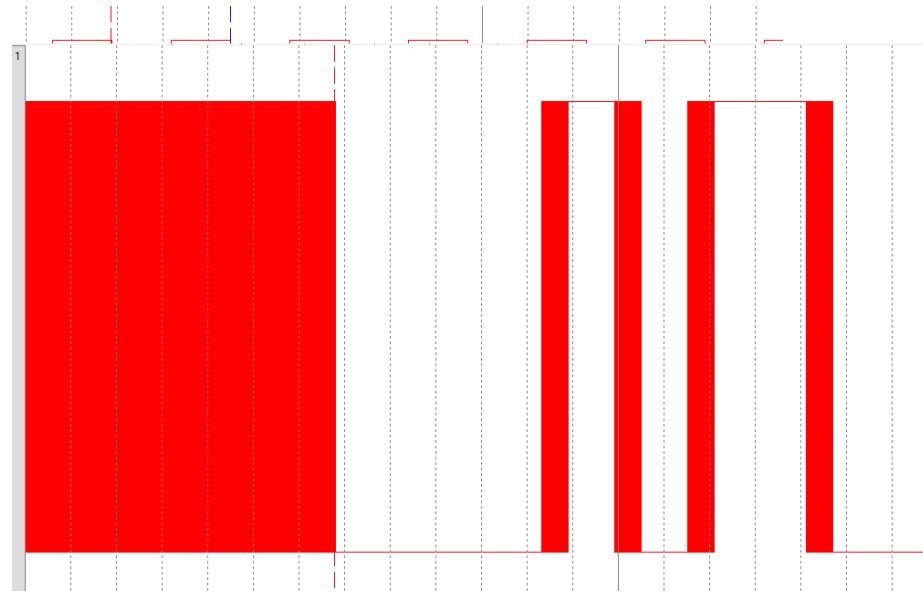


FIGURE 13 – Émission de la trame «110»

4.4 Test sur la carte

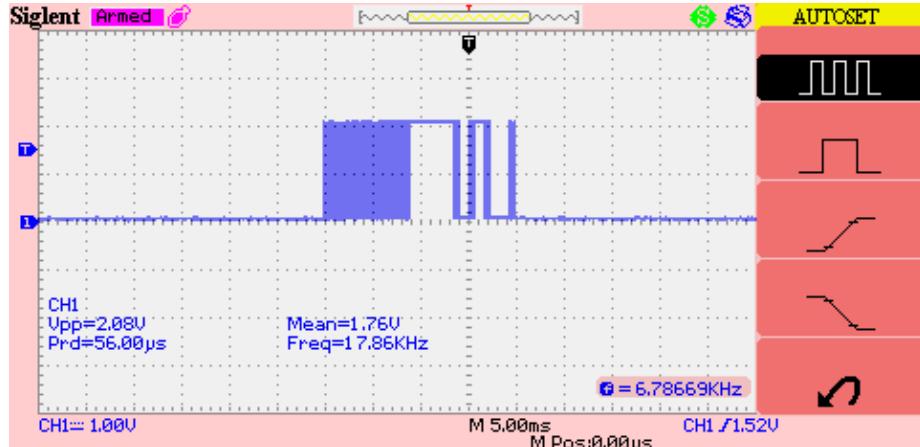


FIGURE 14 – Émission de la trame «110»

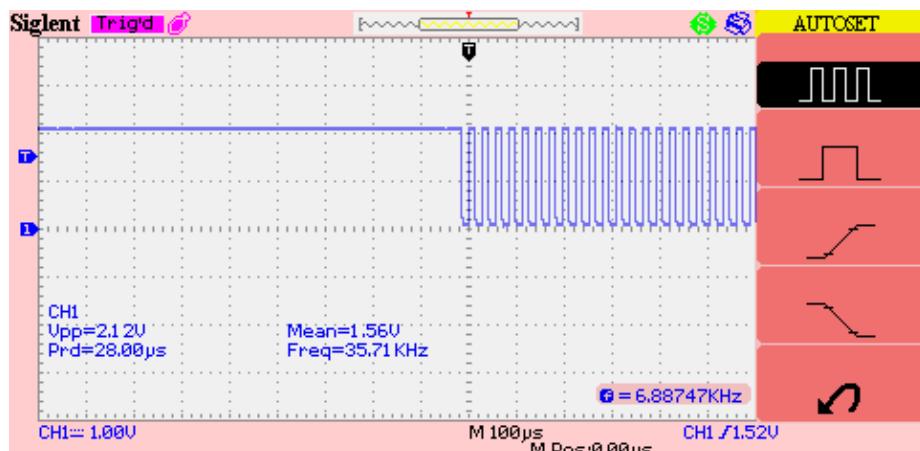


FIGURE 15 – Zoom sur une émission (fréquence de 38kHz)

4.5 Réception

Le périphérique Timer2 est configuré en mode capture et déclenche une interruption à chaque détection d'un front montant ou d'un front descendant sur la broche P0.5. Grâce à la valeur du TC, la durée de l'émission (ou de la non émission) est ensuite analysée dans l'interruption. Le message est récupéré dans la variable globale messageR. Dès que l'analyse d'un message complet a été effectuée, une LED s'allume.

Initialement il était prévu dans le protocole que la fin du message soit détectée par $40\ 000\mu\text{s}$ de non émission. Cependant la fin de cette non émission n'est pas suivie d'un front, on ne peut donc pas générer d'interruption.

Le message envoyé par le verrou a une taille fixe (3 bits dans l'exemple proposé). La fin de l'émission est donc détectée grâce à la variable globale indi-

ceMessage. Lorsque la variable indiceMessage prend la valeur 3, la réception du message est terminée.

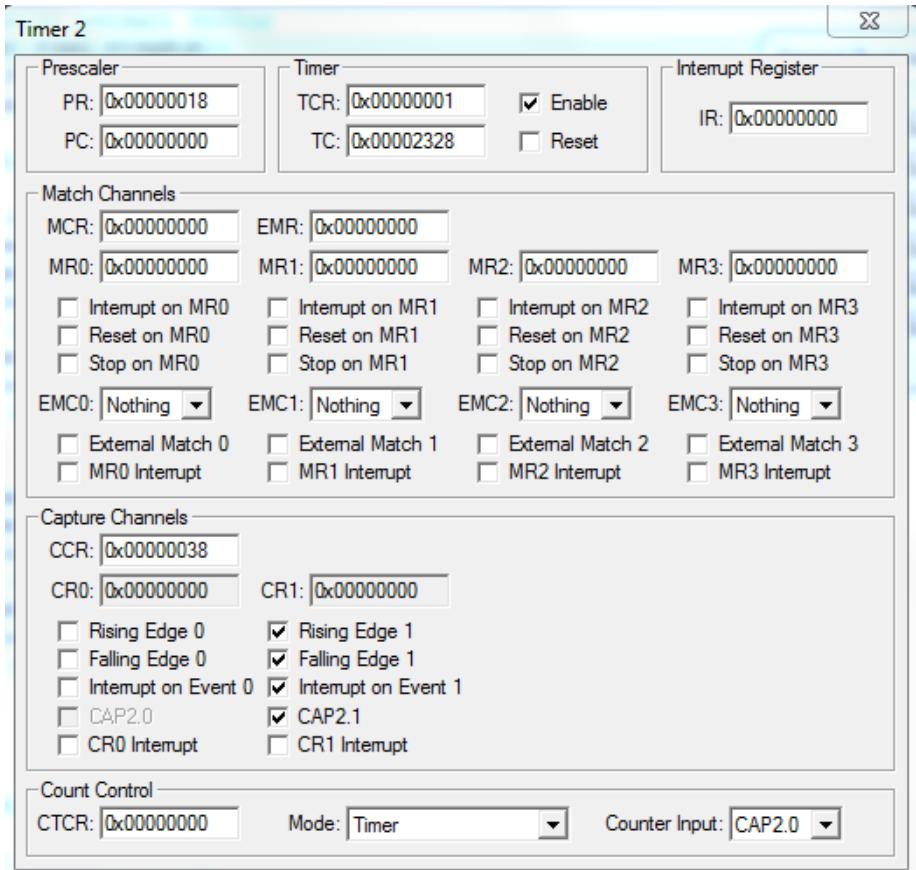


FIGURE 16 – Configuration du Timer2

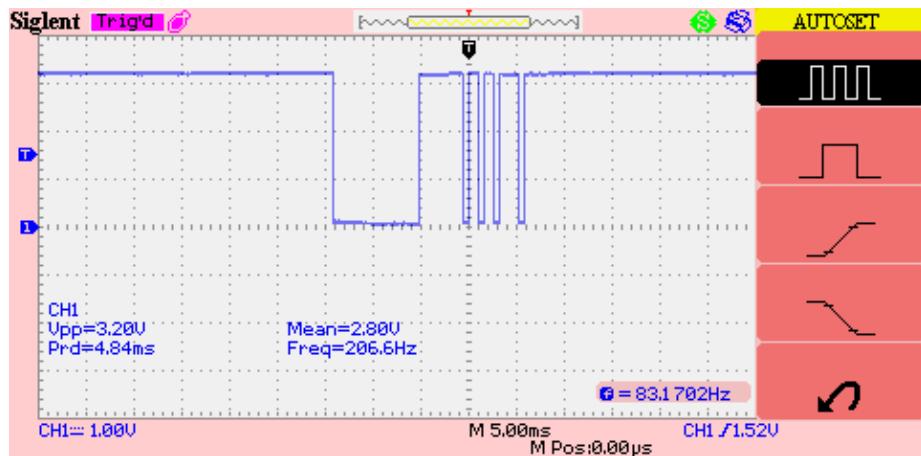


FIGURE 17 – Réception d'un message de 3bits

Deuxième partie

Application d'ouverture à distance d'un verrou

5 Introduction

Nous avons rassemblé les codes correspondant aux différents périphériques afin de réaliser notre application. Il a donc fallu définir la manière dont les différents périphériques allaient "communiquer" entre eux. Nous avons décider d'utiliser des variables globales, qui sont dans un premier temps modifiées dans les procédures correspondant à un périphérique, puis lues dans le programme principal afin de déterminer si d'autres procédures, correspondant à d'autres périphériques, doivent être lancées.

6 Ordonnancement des tâches

Les différents périphériques sont dans un premier temps initialisés. Ensuite, le programme principal ordonne les différentes tâches correspondant aux différents périphériques dans le "while(1)".

Dans un premier temps, on active si besoin les procédures correspondant au module ultra-son. Il est à noter que les autres périphériques, à l'exception du bipper, ne peuvent agir tant que le bouton poussoir activant l'ultra-son. Ainsi, on vérifie avant de lancer les différents procédures de l'écran tactile si ce bouton poussoir n'est pas appuyé en vérifiant la variable bouton_appui.

Le bipper est activé par l'écran tactile et le module ultra-son à l'aide des variables ecran_appui et bouton_appui, respectivement. Le bipper sert également à réinitialiser l'écran tactile au bout de 15 secondes si le code n'a pas été validé, avec le passage de bipActive de 1 à 0.

Enfin, l'émission infrarouge est déclenchée lorsque la tentative d'ouverture est un succès, ce qui est indiqué par le passage à 1 de la variable tsOpen pour l'écran tactile ; et le passage à 3 de la variable open pour le module ultra-son.

7 Fonctionnement de l'application

7.1 Saisie du code sur l'écran tactile

Le code peut être saisi sur l'écran tactile à tout moment, dès lors que le bouton poussoir du module ultra-son n'est pas appuyé. Le code doit être entré dans les 15 secondes suivant le premier appui sur l'écran. Ce délai est matérialisé par la séquence de "bips" émise par le haut-parleur.

Le code correct est 1234. Si ce code est rentré en moins de 15 secondes, l'écran affiche le message "Déverrouillage", et l'émission infrarouge est lancée. Si le code entré est erroné, l'écran affiche un message d'erreur avant de se réinitialiser. Si le délai s'écoule sans qu'on ait rien validé, l'écran se réinitialise, sans afficher de message particulier.

7.2 Ouverture par passages devant le module ultra-son

Pour activer l'ouverture du verrou par passages devant le module ultra-son, il faut appuyer sur le bouton poussoir situé sur ce module.

Afin d'ouvrir le verrou, la séquence à réaliser se compose de 3 passages de la main devant le module à une distance d'environ 10 cm. Le succès est indiqué par l'arrêt du bipper.

Il est à noter que le bouton poussoir doit être maintenu enfoncé pendant toute la durée de la tentative d'ouverture.