21/12/2015

Rapport de mini-projet microprocesseur

Système d'alarme anti-intrusion

Promotion de Gennes

Année 2015 - 2016

MURET Guillaume
LADRANGE Benoit

Table des matières

1 Cahier des charges

Notre projet consiste en la réalisation d'un système anti-intrusion. Lors d'une détection de présence ou d'une manipulation du boitier de l'alarme, notre système devra envoyer un message par le biais d'une communication UART afin d'annoncer une intrusion puis, déclencher une caméra. Une fois la caméra lancée, l'utilisateur pourra visionner l'acquisition de la vidéo surveillance sur un petit écran LCD présent dans le système.
Prendre une photo en appuyant sur un bouton qui sera ensuite enregistrée sur une carte SD est également une fonctionnalité souhaitée.

Nous attendons une bonne transition entre les différents périphériques grâce à une conception logicielle que nous aurons définie rigoureusement au préalable. Le système devra également entrer en interruption lors d'appui sur un bouton ou lors d'une détection de présence.

Les enjeux majeurs de ce projet seront de créer un système autonome et sans disfonctionnement.

2 Manuel d'utilisation

Une fois le dispositif mis sous tension, il suffit d'attendre qu'un mouvement soit détecté par le capteur de présence ou qu'un individu manipule la carte pour déclencher le système. L'envoi d'un message, l'activation de la caméra et la prise de photo seront des évènements gérés par notre microcontrôleur.

3 Choix des composants

Le premier composant au cœur de notre système et indispensable à la réalisation de notre projet est la carte STM32F4-discovery. Ce microcontrôleur est responsable de la gestion des périphériques externes, des interruptions et de toutes les fonctions et procédures générées par notre code source.

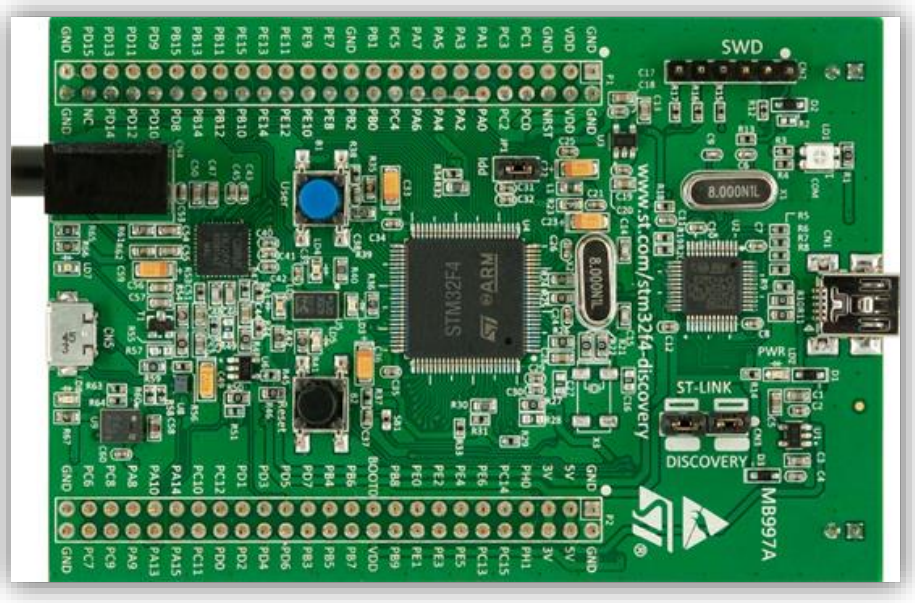


Figure 1: Carte STM32F4-discovery

Le second composant indispensable à la réalisation de notre projet est la carte DM-STF4BB qui vient s'emboiter avec la carte stm32f4-discovery (composant du dessus). De nombreux périphériques externes peuvent s'ajouter à cette carte comme une caméra, un écran LCD, une carte micro SD ainsi que des communications UART ou ETHERNET.

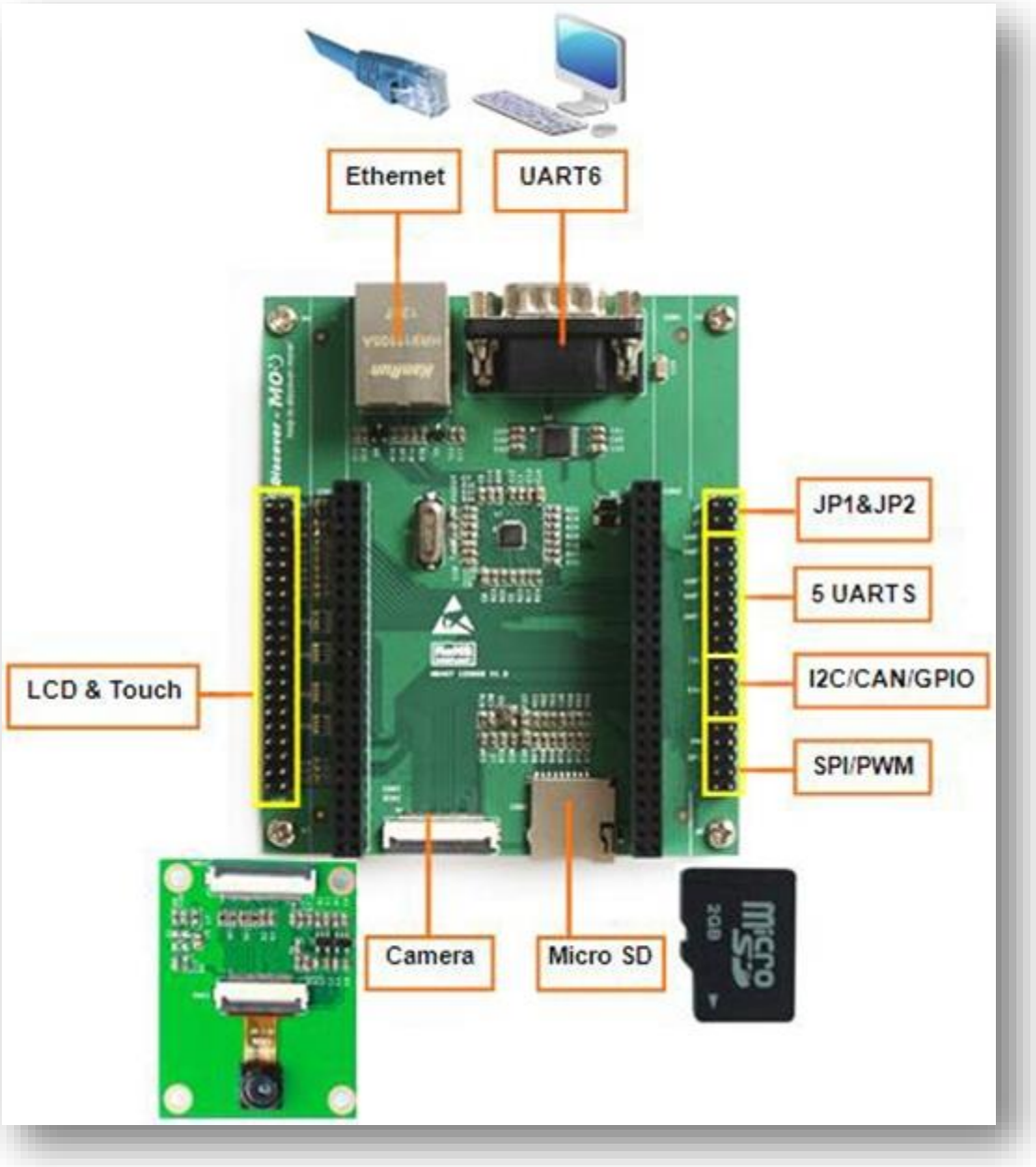


Figure 2: Carte DM-STF4BB

Un autre composant utilisé dans notre système est l'écran LCD. Celui-ci permet d'afficher ce que filme la caméra en temps réel. Même si cet écran est tactile, nous n'utilisons pas cette fonctionnalité du composant. L'écran LCD est le DM-LCD35RT.

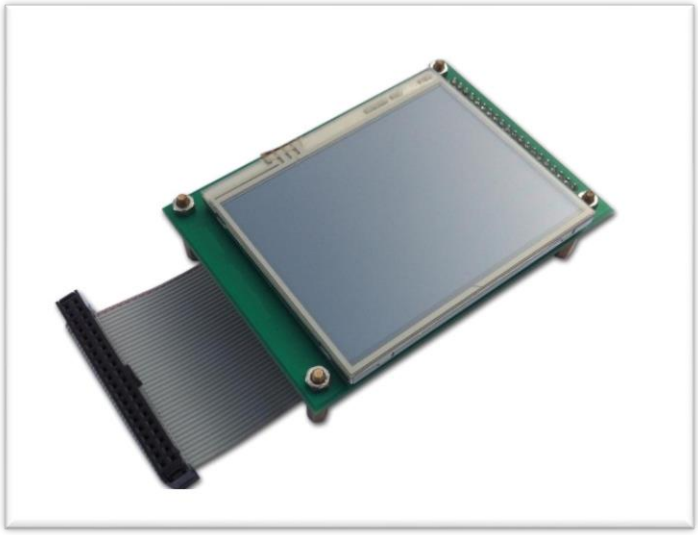


Figure 3 : Ecran LCD DM-LCD35RT

Nous avons utilisé la caméra fournie par l'ESEO : la DM-CAM130. Celle-ci sert à filmer et affiche ce son acquisition sur le LCD en direct.

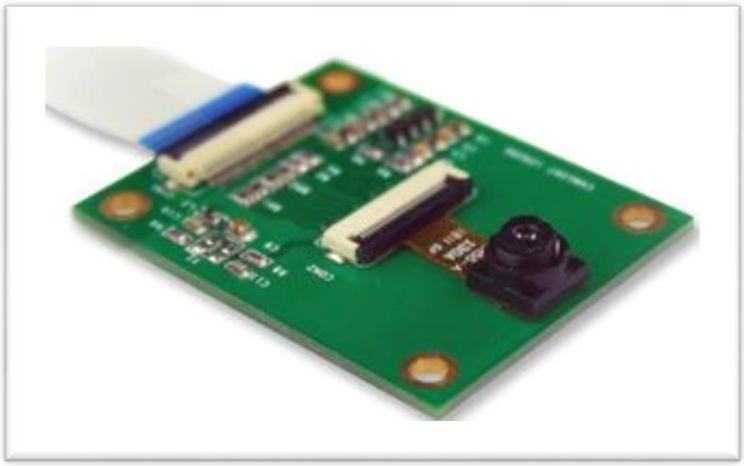


Figure 4: Caméra DM-CAM130

Le détecteur de présence est le « digital infrared motion sensor (SKU:SEN0018) ». Celui-ci déclenche un niveau logique haut sur une sortie spécifique lorsqu'une présence est détectée.

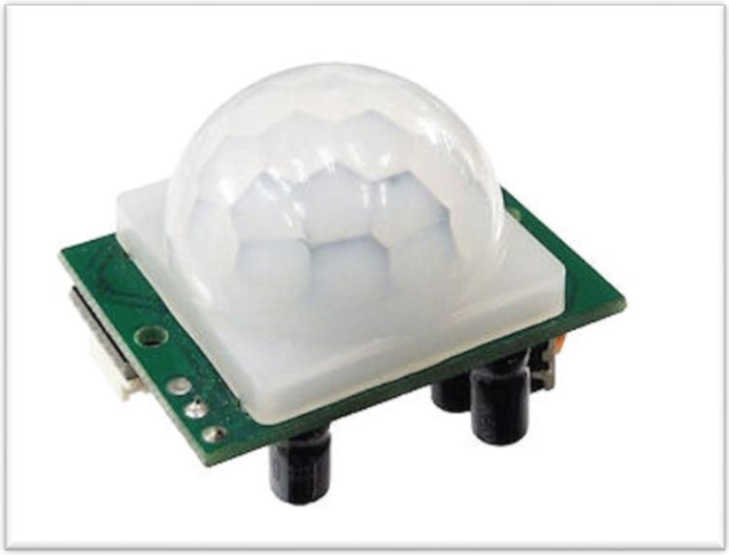


Figure 5: Digital infrared motion sensor (SKU:SEN0018)

Enfin, nous avons utilisé un câble USB/UART qui nous permet de manipuler l'UART pour envoyer et recevoir des caractères sur la liaison série RS232 de la carte STM32F4-discovery. Ces caractères seront envoyés sur notre PC à travers le câble USB.

Figure 6: Câble USB/UART

4 Démarche de l'élaboration du système anti-intrusion

Avant de commencer notre projet, nous avons réfléchis aux différentes étapes clefs pour faciliter la programmation de notre système.
Dans un premier temps, nous avons défini notre système comme suit, à l'aide de la machine à état générale :

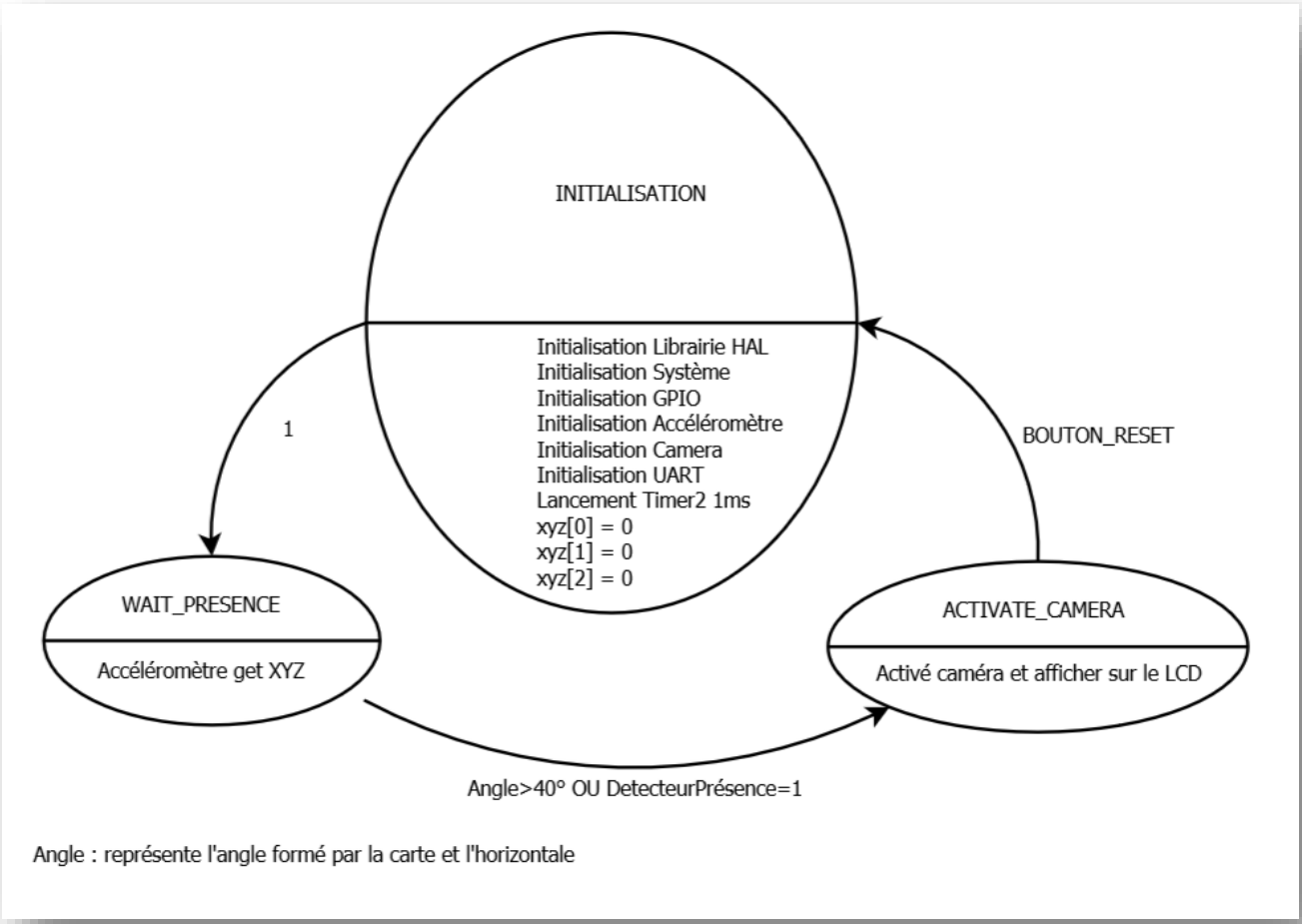


Figure 7: Machine à état : Système anti-intrusion

- **Initialisation de départ**

Avant de commencer à coder quoi que ce soit, nous devons initialiser tout ce qui doit l'être en fonction des fichiers que nous utiliserons comme la librairie HAL avec la fonction « HAL_Init » ainsi que le système avec la fonction "SYS_init ».
Dès que ces différentes instructions ont été appelées, nous avons commencé à utiliser le capteur de présence.

- **Interruption du capteur de présence**

Nous avons configuré notre première interruption reliée au capteur de présence. Sachant que le capteur de présence envoi un niveau haut lorsque celui-ci détecte une présence, nous avons dû configurer une broche de notre carte en interruption sur front montant. Nous avons regardé dans le fichier EXCEL (Ports_STM32F4) quel broche nous pouvions utiliser sans affecter d'autres broches utilisées par nos autres périphériques externes.
Dans la fonction « GPIO_Configure() » du fichier « stm32f4_gpio.c », nous avons configuré la broche PE1 en interruption sur front montant (GPIO_MODE_IT_RISING) car cette broche n'est pas utilisée lors de l'état d'attente d'une présence. Nous avons ensuite définit l'interruption

« EXTI1_IRQn », puis activé celle-ci avec : « NVIC_EnableIRQ(EXTI1_IRQn) », et pour finir, nous avons défini une priorité d'exécution d'interruption à l'aide de la fonction « HAL_NVIC_SetPriority(EXTI1_IRQn , 0, 1) ». Sans oublier de créer la procédure d'interruption « void EXTI1_IRQHandler(void) » et d'acquitter l'interruption.
Ensuite, une fois que l'interruption a été déclenchée puis acquittée, nous indiquons au programme de changer d'état et de désactiver l'interruption « NVIC_DisableIRQ(EXTI1_IRQn) » dans la procédure d'interruption pour ne plus y revenir de façon intempestive.

TEST : Pour tester notre détecteur de présence, nous avons commencé par vérifier qu'un signal était envoyé par le capteur sur la pin « Digital Signal Out » (fil vert) à l'aide de l'oscilloscope, sans oublier de connecter la broche VCC (fil rouge) au 5V du microcontrôleur et la broche GND (fil noir) à la masse.
Ensuite, pour vérifier que le programme rentrait bien en interruption, nous avons placé un breakpoint dans la fonction d'interruption « void EXTI1_IRQHandler(void) ».
Pour finir, nous avons relancé le programme pour le faire tourner en boucle dans la tâche de fond et vérifier que nous ne rentrons pas une seconde fois en interruption.
Dès que ces tests ont été validés, nous sommes passés à l'activation de la caméra.

- **Activation de la caméra**

L'activation de la caméra est un processus très simple car cette fonction est tout faite. Il faut au préalable avoir branché le LCD et la caméra sur les bons ports de la carte DM-STF4BB, faire un reset de la camera avec la fonction « Camera_reset » et lancer le timer 2 par périodes d'une milliseconde avec la fonction « TIMER2_run_1ms » car celle-ci est utilisée dans le rafraichissement périodique de la caméra.
Nous avons utilisé la fonction « Camera_statemachine(FALSE, MODE_CAMERA_TO_SRAM) ». Le booléen FALSE veut dire qu'on empêche la caméra de s'éteindre à la fin de la machine à état. Le deuxième paramètre est le mode utilisé. Nous avons choisi ce mode car il nous permet d'afficher l'acquisition de la caméra en même temps qu'un titre « CAMERA SURVEILLANCE » ainsi qu'une petite légende en dessous pour indiquer à l'utilisateur qu'il peut réinitialiser le système en appuyant sur le bouton reset (bouton noir). Ces deux petits messages ont été ajoutés dans le fichier source « demo_camera.c » dans la fonction « Camera_statemachine », dans l'état « CAMERA_ENABLE » de la ligne 151 à 153 à l'aide des fonctions « LCD_SetFont » qui gère la taille de la police de caractère et « LCD_DisplayStringLine » qui gère l'emplacement, l'affichage et la couleur du texte.

TEST : Pour tester l'activation de la caméra, nous avons appelé la fonction d'activation « Camera_statemachine » directement dans la tache de fond. Ce test fut un succès rapide car la fonction était déjà créée et fonctionnait parfaitement sans besoin de retouches. Ensuite nous avons testé l'ensemble capteur, caméra et LCD.
Une fois ces tests validés, nous avons ajouté la liaison série UART.

- **Envoi d'un message par l'UART**

L'envoi d'un message avec l'UART a été réalisé à l'aide de l'UART2 car l'UART6 possède des broches communes avec la caméra.
Pour commencer, dans l'état d'initialisation, nous avons initialisé l'UART2 en ajoutant les deux instructions suivantes « UART_init(2,UART_RECEIVE_ON_MAIN) » et « SYS_set_std_usart(USART2, USART2,USART2) ». Ces deux instructions exécutées, le programme peut communiquer à partir de l'UART2 avec une fréquence de 115200 bauds/sec et un bit de stop.
Pour communiquer entre la carte et notre PC, nous avons utilisé le logiciel Docklight qui est un logiciel gratuit de test et de simulation de signaux de type RS232, RS485/RS422, TCP/UDP. Dans notre cas, la communication est de type RS232. Avec l'aide du fichier EXCEL (Ports_STM32F4), nous avons connecté les broches du câble USB/UART aux ports mentionnés sur le fichier EXCEL : la broche de transmission TX du câble (fil blanc) est connectée à la broche PA2 de la carte, la broche de réception RX du câble (fil vert) est connectée à la broche PA3 de la carte, la broche de masse GND du câble (fil noir) est connectée à la broche GND de la carte et la broche de VCC du câble (fil rouge) est connectée à la broche 5V.

TEST : Une fois les connexions établies, nous avons testé l'envoi d'un caractère. Pour cela, l'utilisation dans la tâche de fond de la fonction « UART_putc(2,0b10101010) », qui envoie un caractère sur 8 bits avec un bit de stop à travers l'UART2, nous a servi de vérifier la bonne émission des caractères ainsi que la durée du temps d'un bit pour pouvoir synchroniser et respecter le même protocole de communication entre l'UART et notre PC. L'avantage de cette fonction est que nous n'avons pas besoin de gérer le buffer puisque cette fonction est bloquante si un caractère est déjà en cours d'envoi.
Voici ce que nous avons obtenu à l'oscilloscope :

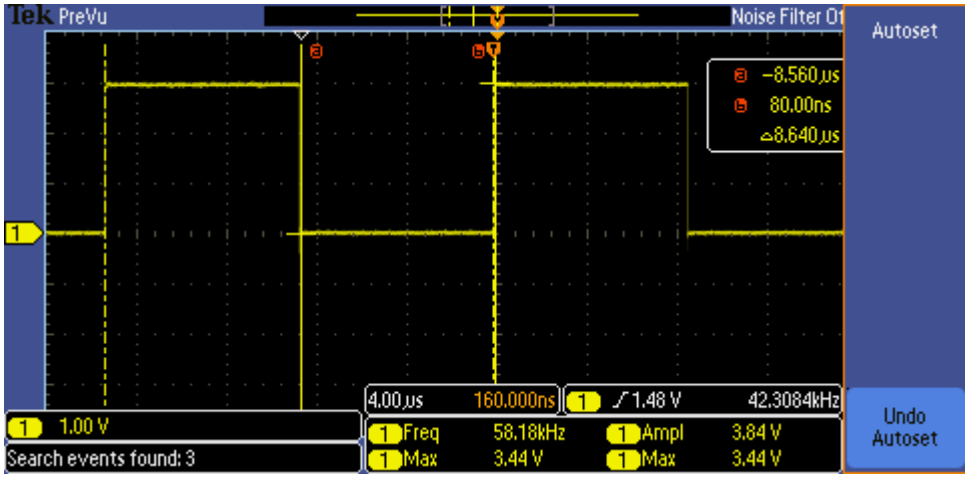


Figure 8: Signal oscilloscope d'un temps bit

On voit nettement (à l'aide des curseurs « a » et « b ») qu'un temps bits vaut $\frac{1}{8.640\mu s}$ = 115 740 bauds/sec (soit 115 200 bauds par seconde en valeur normalisée).

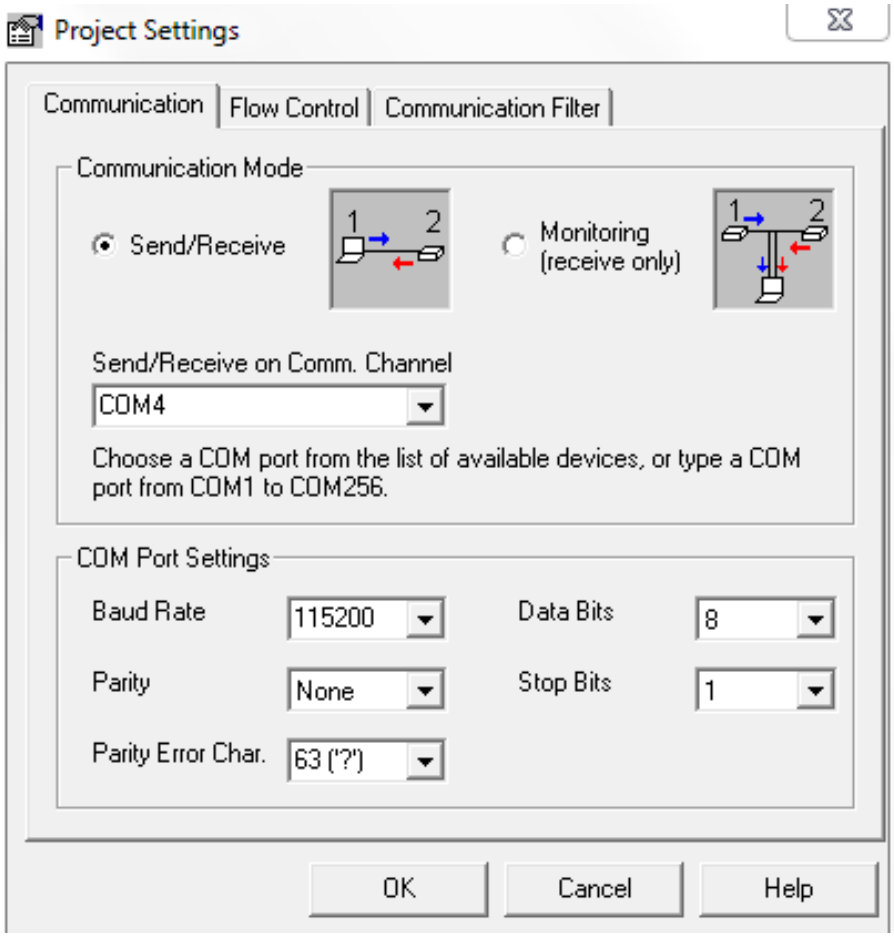


Figure 9: Fenêtre de configuration de la communication sur Docklight

Une fois que nous connaissons la valeur du temps d'un bit, nous pouvons régler la vitesse de communication sur Docklight et tester l'envoi d'un caractère ASCII qui sera interprété par le logiciel. Pour cela, nous avons encore utilisé la fonction « UART_putc(2,0x41) » dans la tâche de fond qui sera censée envoyer le caractère « a » en continue sur Docklight car 0x41 en hexadécimal correspond au caractère « a » en ASCII.
Une fois ce test vérifié, nous avons testé l'envoi d'un message à l'aide d'un « printf("Message \n") » après l'initialisation de l'UART2.
Voici ce que nous obtenons sur Docklight :

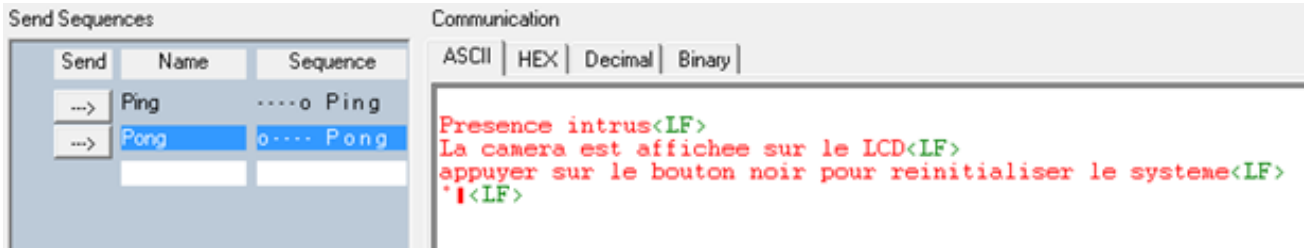


Figure 10: Message envoyé par l'UART2 après détection par le détecteur de présence

Ces tests validés, nous avons décidé d'utiliser l'accéléromètre pour informer l'utilisateur qu'un individu manipule la carte électronique.

- **Scrutation avec l'accéléromètre**

Avec l'aide de la mission 2 réalisée avant notre mini-projet, nous avons utilisé l'accéléromètre. Cependant, celui-ci possède des ports en commun avec le LCD. Pour pallier ce problème, nous avons appelé la fonction « DEMO_ACCELERO_SetPorts(PORTS_FOR_ACCELERO) » car nous n'utilisons pas le LCD pendant la scrutation avec l'accéléromètre. De plus, nous avons appelé la fonction « BSP_ACCELERO_Init » pour initialiser l'accéléromètre dans l'état « INITIALISATION » de la machine à états. La procédure en scrutation s'appelle « acceleroCarte » et est appelée dans l'état « WAIT_PRESENCE ». Dans cette procédure, on utilise l'accesseur « BSP_ACCELERO_GetXYZ » qui récupère les trois composantes de l'accéléromètre. Nous ne nous intéressons qu'à la composante suivant Z, c'est-à-dire celle qui repère l'inclinaison de l'ensemble de la carte.
Si la carte est à l'horizontale avec les leds et les boutons dirigées vers le haut, la composante Z tendra vers la valeur 1024. Tandis que si l'utilisateur bouge la carte, plus l'inclinaison sera forte, plus la composante Z diminuera. Si la carte est retournée à l'horizontale, c'est-à-dire avec les leds et les boutons dirigées vers le bas, la composante Z tendra vers la valeur -1024.
Dans notre cas, si la carte a une inclinaison de plus de 40° par rapport à l'horizontale, (la valeur de Z inférieure à 600) notre système change d'état et active la caméra. Dans cette condition, nous devons désactiver l'interruption du détecteur de présence avec « NVIC_DisableIRQ(EXTI1_IRQn) » pour ne plus rentrer en interruption avec le capteur de présence étant donné que la caméra fonctionne déjà. De plus, nous devons configurer les ports pour le LCD avec la fonction « DEMO_ACCELERO_SetPorts(PORTS_FOR_LCD) » pour rendre la main au LCD. Nous avons ajouté un petit message à l'utilisateur par l'UART2 pour l'informer d'une présence externe.

TEST : Confiant de notre code et voulant tester les nouvelles fonctionnalités apportées à notre système, nous avons décidé de vérifier directement l'ensemble de notre système avec le détecteur de présence. Nous avons placé un breakpoint dans la condition sur la composante suivant Z et après une inclinaison de la carte d'environ 40°, le système se stoppe dans la condition. Après réactivation de notre système, la caméra s'active et ces messages s'affichent sur Docklight :

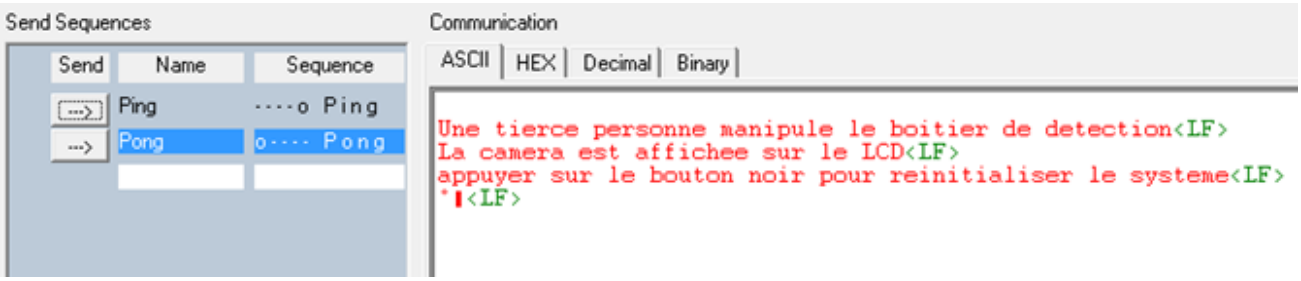


Figure 11: Message envoyé par l'UART2 après détection par l'accéléromètre

L'amélioration de notre système avec l'accéléromètre est ainsi validée.

5 Structure du programme

Procédures et fonctions crées ou modifiées		
Fichier source	**Fonction/Procédure**	**Description**
main.c	void EXTI1_IRQHandler(void)	Procédure d'interruption sur front montant sur la broche PE1
	void acceleroCarte()	Procédure en scrutation pour l'accéléromètre
	void machineAetats_capteur(void)	Machine à état du système "anti-intrusion"
	int main(void)	Main du programme avec tâche de fond et appel de la machine à état
demo_with_lcd.c	void initialisation(void)	Initialisation du système, librairie HAL, accéléromètre, UART2 et caméra
	void display_Camera_On_LCD(void)	Appel à la machine à état de la caméra
stm32f4_gpio.c	void GPIO_Configure(void)	Configuration des ports pour la broche PE1 en interruption sur front montant et la priorité de l'IT
demo_camera.c	running_e Camera_statemachine ()	Machine à état de la caméra

Tableau 1: Procédures et fonctions crées ou modifiées

6 Tests

Voici les différents tests réalisés durant les séances de mini-projet détaillés dans la section 4. Nous les résumons à travers ce tableau :

Tableau des tests		
Test	**Description**	**Validation**
Interruption capteur de présence	Vérifier si notre système rentre une seule fois en interruption en utilisant un breakpoint dans la fonction d'interruption.	Validé
Activation de la caméra	Vérifier l'activation de la caméra au bon moment, c'est-à-dire dès qu'une présence a été détectée, et l'affichage sur le LCD.	Validé
Envoi d'un message par l'UART	Vérifier la manipulation de l'UART pour envoyer des messages sur la liaison série RS232.	Validé
Scrutation avec l'accéléromètre	Vérifier après une inclinaison d'environ 40° de la carte (suivant la composante Z), le système active la caméra.	Validé

Prise de photo et enregistrement sur carte micro SD	Vérifier qu'une photo a été enregistrée sur la carte micro SD après appui sur le bouton bleu.	Non Validé

Tableau 2: Récapitulatif des tests

7 État d'avancement

Nous avons rempli une grande partie du cahier des charges, nous avons bien incorporé dans notre système l'accéléromètre, le détecteur de présence et la caméra diffusée sur le LCD. Les évènements de scrutation et d'interruption ont bien été testés et validés. Cependant nous ne sommes pas parvenus à prendre une photo et l'enregistrer sur la carte micro SD car celle-ci n'était pas reconnue par notre carte électronique. A ce jour, nous n'avons pas été en mesure de déterminer la nature du problème. Cependant, le système est opérationnel car notre il détecte une présence ainsi qu'une manipulation du boitier et active la caméra après une des deux détections.

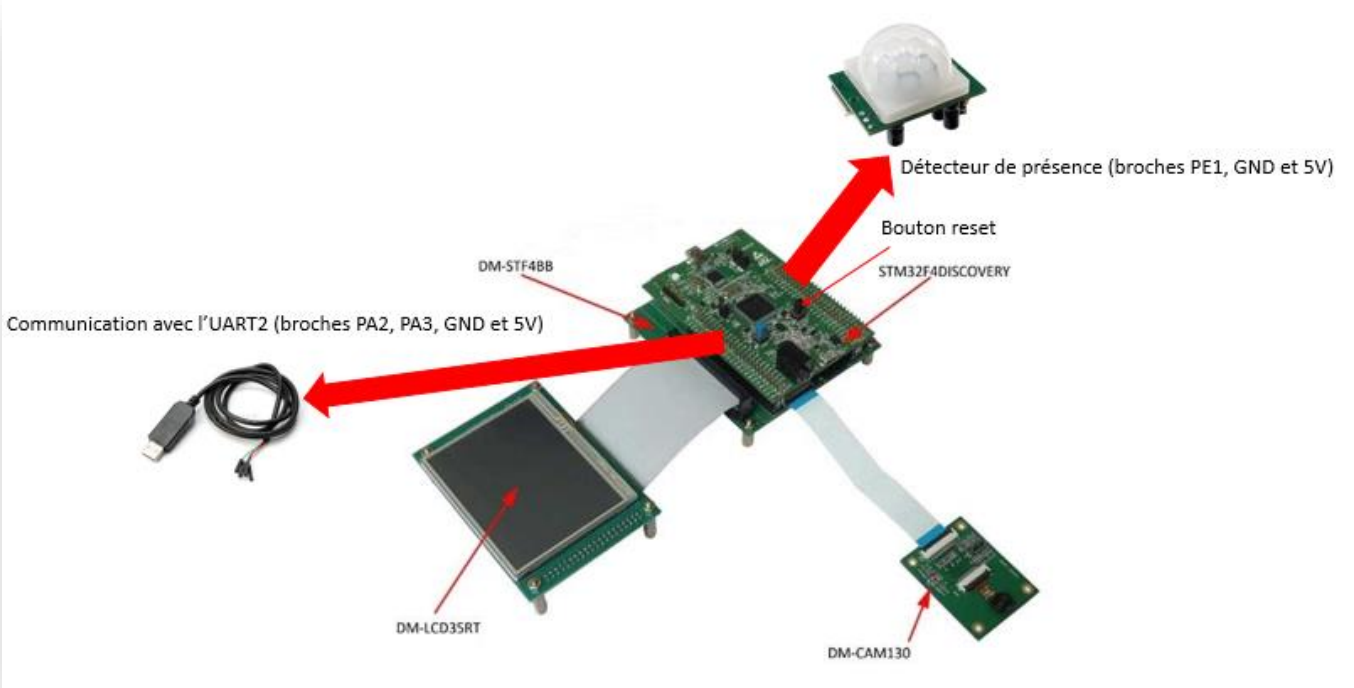
8 Conclusion

Ce projet, dans son intégralité, nous aura fourni un réel enrichissement étant donné que nous avons pu mener ce projet de A à Z avec plus ou moins d'autonomie. En effet, nous avons été totalement libres du choix du sujet et nous avons mené toutes les étapes nécessaires pour le voir aboutir.

La programmation en vue d'une implémentation sur le microprocesseur nous aura permis de mettre en pratique les connaissances acquises lors des séances de cours et de TD sur le logiciel « System Workbench for STM32 » ainsi que lors des séances de mini-projet microprocesseur de I1. . Nous avons eu quelques problèmes de conflit lors de l'utilisation de certains périphériques qui avaient des broches communes avec d'autres.

Au travers des différentes difficultés rencontrées, nous nous sommes rendus compte de plusieurs détails cruciaux qui nous aurons marqués et que nous espérons ne plus reproduire dans le futur notamment en option EOC ou SE et dans le monde du travail.

Annexe 1 : Schéma d'ensemble du système



Annexe 2 : Table des illustrations