Plateforme d’atterrissage d’un module spatial miniature

Projet de Printemps 2019 - Rapport Final

1. Cahier des Charges V2

Mise en contexte

Lors de l'atterrissage d'une fusée, il est utile pour les équipes au sol de savoir quand est-ce qu'elles peuvent s'approcher afin d'éviter les accidents. Nous modélisons ce scénario.

Lors de la phase d'approche d'un objet nous détectons sa position et affichons son état. Puis, une fois au sol, sa température est enregistrée et celui-ci est refroidi à une température choisie avant de donner le feu vert aux techniciens.

Modifications Ajouts

Mise en oeuvre

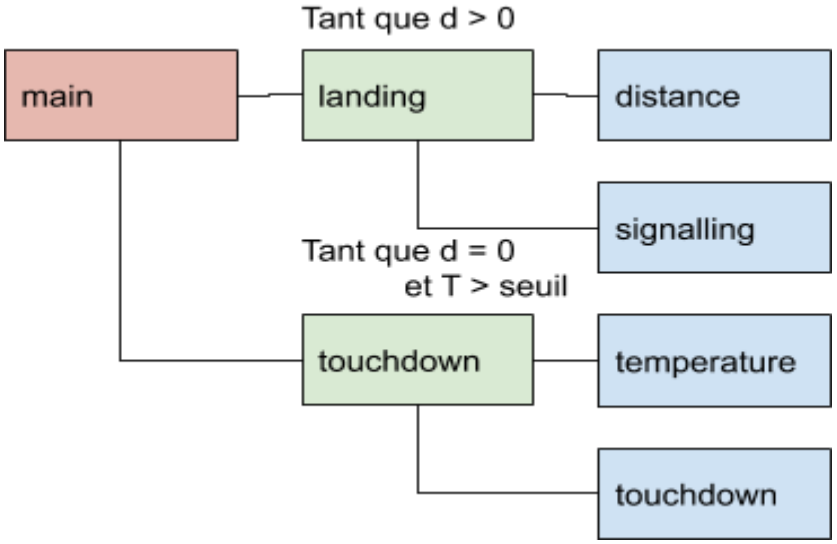
Utilisation des périphériques de la carte STK-300

- **Capteur de distance GP2Y0A21YK0F** DDRF **:** Détection de la position du module, le résultat est utilisé pour modifier l'affichage de la matrice de LED. Utilisable sur une distance de 10 à 80 cm.
- **Matrice de LED WS2812b** DDRE **:** L'affichage est modulé en fonction de la proximité du module comme détecté par le capteur de distance.
- **Capteur de température DS18B20** DDRB **:** Détection de la température de l'objet une fois au sol. Précision de la mesure tronquée à un entier sur 8-bits. Pas d'utilité pour la fonction alarme disponible pour ce projet. Communication 1-Wire.
- **Moteur JST X27** DDRD **:** Utilisé pour simuler le refroidissement de la zone, tel un ventilateur. Mieux adapté qu'un servomoteur car il risque de devoir tourner sur de longues durées.
- **Boutons** (intégrés) DDRD **:** Changement manuel de la valeur de température acceptable au degré près. PD0 pour l'incrémentation et PD1 pour la décrémentation.
- **LCD 2x16 Hitachi 44780 U :** Affichage de la température actuelle et du seuil choisi via les boutons sur la ligne du haut ainsi que d'un message informant sur l'état des opérations sur la ligne du bas.

Tout le processus est automatisé. Le programme tournera en permanence dans une boucle correspondant soit à l'atterrissage, soit au refroidissement du module spatial. Les boutons PD0 et PD1 qui servent à interagir avec l'utilisateur pour fixer le seuil de température feront donc office d'interruptions durant la phase de mise en place.

Le bouton PD3 sera utilisé comme reset général pour pouvoir facilement redémarrer le processus pour un deuxième atterrissage.

Schéma fonctionnel



Sous-routines	Description
Main	Chargée de l'exécution ordonnée du programme et de l'interaction utilisateur via LCD.
landing (sous-routine de main) distance > 0	Tourne pendant la phase d'atterrissage du module spatial. Chargée de l'analyse de la position du module et de la gestion de l'affichage sur la matrice de LED.
distance (sous-routine de landing) PINF	Détecte la position approx. du module pour la signaler via la matrice de LED.
signalling (sous-routine de landing) PORTE	Signale par un schéma évolutif, la proximité du module spatial.
nb_led (inclue dans signalling)	Change le nombre de LED allumées selon le palier atteint par le module.
color_led (inclue dans signalling)	Change la couleur des LED selon la distance du module.
touchdown (sous-routine de main) température > seuil et distance = 0	Tourne une fois que le module spatial a atterri. Chargée du refroidissement, elle analyse les données du capteur de température, fait tourner le moteur si nécessaire et gère l'affichage LCD.
temperature (sous-routine de touchdown) PINB	Détecte la température de l'objet et si elle n'est pas dans les normes choisies par l'utilisateur, lance la procédure de refroidissement.
cooling (sous-routine de touchdown) PORTD	Fait tourner le moteur pour simuler l'activation d'un système de refroidissement.

La deuxième ligne du panneau LCD est réservée à l'affichage d'un message utilisateur au cas où cela s'avèrerait utile pour l'une des sous-routines main.

2. Manuel d'utilisation

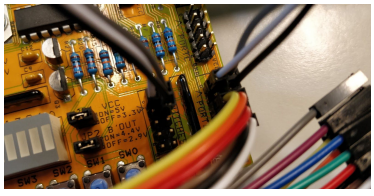
2.1 Mise en place

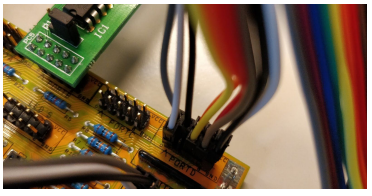
Le tableau suivant décrit les différentes connexions des périphériques.

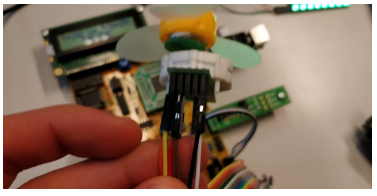
Module	PORT	Périphérique
M4	F	Capteur de Distance
Matrice de LED	E (Pin 1 + Vcc/Gnd)	---
M5	B	Capteur de Température
Boutons intégrés	D (Pins 0, 1, 3)	---
M1	D (Pins 4, 5, 6, 7 + Vcc/Gnd)	Moteur pas à pas
LCD	---	---

Schéma explicatif plus concret pour le partage du PORTD :

Boutons intégrés		PORTD		M1	
0	1	0	1	0	1
2	3	2	3	2	3
4	5	4	5	4	5
6	7	6	7	6	7
GND	Vcc	GND	Vcc	GND	Vcc







2.2 Utilisation

Au lancement du programme utilisez les boutons PD0(+) et PD1(-) pour fixer la température maximale que vous jugez acceptable, affichée en haut à droite sur le LCD, vous pouvez maintenir le bouton appuyé. Vous avez 7s pour le faire, le programme va ensuite démarrer automatiquement. Le reste est automatisé.

Durant l'atterrissage la position du module spatial vous est communiquée par la matrice de LED et une fois au sol vous verrez la température en temps réel en haut à gauche sur le

LCD. Le système de refroidissement est activé si la température actuelle est supérieure au maximum choisi. Une fois le module refroidit un message s'affiche et les équipes au sol peuvent s'approcher.

A tout moment vous pouvez recommencer à la case départ en pressant le bouton PD3.

3. Fonctionnement

3.1 Initialisation

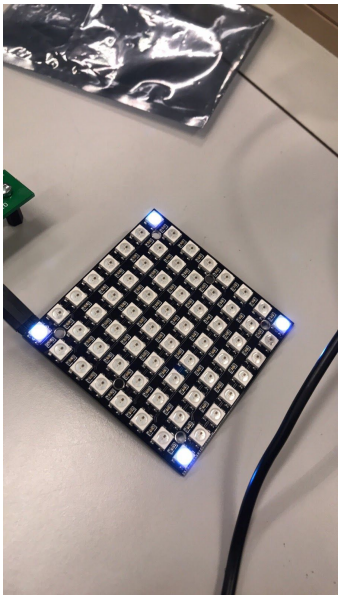
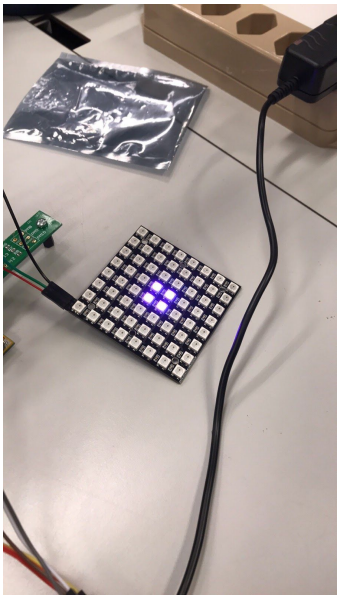
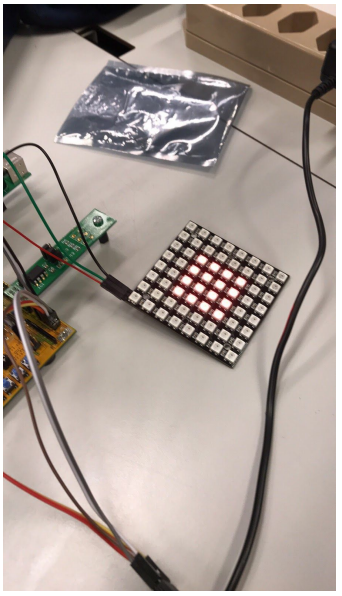
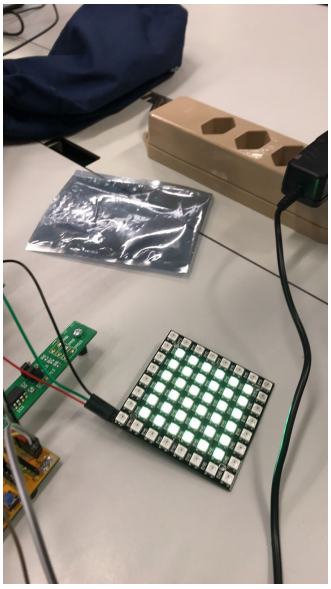
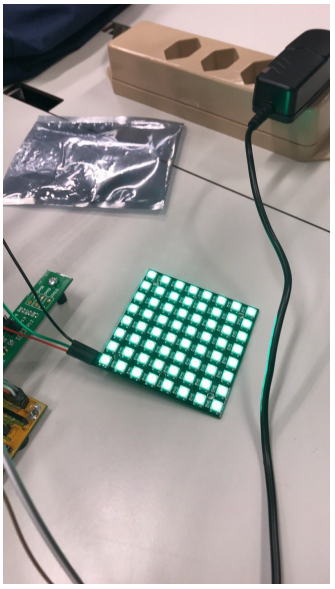
Cette partie fait usage des boutons d'interruption. Au lancement du programme, une boucle d'attente *WAIT_MS 7000* donne le temps à l'opérateur de choisir son seuil de température via des interruptions de cette boucle via les boutons PD1 et PD0. La routine de service d'interruption alors appelée incrémente ou décrémente le registre *b0* réservé à cet effet. Au début du programme *b0* est initialisé à 25°C. Ces routines sont équipées d'une boucle d'attente de *WAIT_MS 150* qui permet des interruptions contrôlables tout en gardant le bouton en question enfoncé. De plus, à chaque changement de *b0* la sous-routine *show_temp* est appelée pour actualiser les valeurs affichées sur le LCD.

Au terme des 7s, les interruptions des boutons PD1 et PD0 sont désactivée. Le reset, PD3, reste utilisable à tout moment.

3.2 Première partie : Landing

Cette partie fait usage du capteur de distance et de la matrice de LED. Voici leur fonctionnement :
Etant donné que nous avions besoin de peu d'états d'affichage différents des LEDS nous avons choisi de faire une sous routine pour afficher chaque état. L'écriture des sous-routines est condensée à 8 lignes par l'usage de différentes macros.
Tout d'abord la matrice de LED est initialisée avec la sous routine ws2812b4_init. Ensuite le convertisseur analogique/numérique est initialisé avec le bit ADEN du registre ADCSR qui est mis à 1 et le prescaler qui est défini à clock/64. Le canal de données lues par le convertisseur: GP2_aval, est sélectionné via la macro ADMUX pour le capteur de distance. Ensuite dans le main la conversion est faite et la valeur numérique correspondant à la tension est stockée dans 2 registres, un byte bas et un byte haut.
La valeur de tension est comparée une première fois à la valeur 180 (relativement éloigné du capteur).
Si la valeur est inférieure à 180 (le module est plus loin, plus la valeur est élevée plus l'objet est proche du capteur) alors c'est l'état 0 des LED qui est affiché, soit toutes les leds éteintes sauf 4 leds blanches dans les coins. Sinon le programme passe dans une sous routine qui compare à nouveau la valeur de tension à la valeur 270. Si elle est inférieure alors l'état 1 des leds est affiché (4 leds du milieu en violet) sinon la valeur est ensuite comparée à la valeur 270. Si elle est inférieure alors l'état 2 des leds est affiché, soit 16 leds rouge centrées. La valeur de tension est comparée une dernière fois à la valeur 450 et si elle est inférieure, alors 36 leds de couleur jaune centrées sont affichées. Enfin si la valeur

est supérieure à 450, alors le module a atterri et les LEDS sont toutes allumées en vert et la partie touchdown peut être effectuée.

3.3 Deuxième partie : Touchdown

Cette partie fait usage du capteur de température et du moteur pas à pas. Voici leur fonctionnement :

1) Dans un premier temps la température ambiante est mesurée via le module M5 en utilisant la bibliothèque *"wire1.asm"* et une partie du code p.246 du polycopié du cours. La valeur flottante sur deux bits renvoyée est ensuite convertie en un entier via l'instruction *lsl* et la macro *MOVB*. La valeur est stockée dans *c0* et affichée en haut à gauche du LCD.
2) Une comparaison de *b0* et *c0* a lieu et si *c0*, la température actuelle, est supérieure à *b0*, le seuil choisi durant l'initialisation, le moteur est enclenché (simulation d'un processus de refroidissement). Dans le cas contraire, un message indique à l'opérateur que le module spatial peut être approché.
3) Le moteur est mit en mouvement en utilisant la même séquence d'instructions que proposée aux TP mais adaptée à notre branchement, en décalant les motifs de 4 bits vers la gauche, du au partage du PORTD. La durée de rotation est fixée par une boucle dans une boucle (voir sous-routine *cool_down*) de 0x08*0xFF = 2040 répétitions. Le retour au début de chaque boucle se fait via la macro *DLJNZ* (modification de DJNZ qui permet de sauter plus loin en mémoire via un *rjmp*, pas un *jmp,* le *L* signifie *Long*).
4) Nous revenons au point 2) jusqu'à ce que le module soit suffisament refroidit.

4. Conclusion

Au cours de ce projet nous avons développé un système autonome permettant de mettre à profit les enseignements du cours de microcontrôleurs. Nous avons constamment dû remettre en question la cohérence entre nos choix et le software/hardware à disposition.

La tâche était très complète puisque nous étions auteurs du cahier des charges à quelques restrictions près. Entre la conception, les datasheets et la réalisation du projet, nous avons fait face à des problèmes à tous les niveaux, ce qui est une bonne expérience pour la suite.

Nous avons essayé d'être le plus possible en accord avec notre cahier des charges mais certains aspects des périphériques que nous n'avions pas pris en compte nous ont fait changer légèrement de cap pour certaines parties.

Pour l'améliorer, nous aimerions aimé avoir plus de temps à disposition pour faire un petit montage à taille réelle de la plateforme et optimiser certaines parties du code.

Lucas Bost Filip Slezak

Date et Signatures :

5. Annexes

- main.asm
- general.asm - quelques routines et macros usuelles
- landing.asm - code de la première partie
- touchdown.asm - code de la deuxième partie