

# PROJET VEHICULE AUTONOME ET INTERACTIF

*Équipe Télécommande / Partie : Contrôler et Informer*

*Responsables pédagogiques :*

*Sagonéro Cyril*

*Aubry Pierre*

*Intervenants :*

*Oukrat Rémi*

*Nazim Zakari Saibi*



*OFFOUGA Joris*

*SER2*

*Projet Véhicule Interactif*



# Table des matières

Présentation	1
Étude du cahier des charges	3
Recherche de solutions techniques	4
Analyse Fonctionnelle	9
1. Schéma fonctionnel niveau 1	9
2. Schéma fonctionnel niveau 2	10
3. Analyse fonctionnelle	10
4. Description des signaux	11
Analyse Structurelle	12
5. Schéma structurel	12
6. Analyse structurelle	13
7. Nomenclature	15
Étude Logicielle	16
8. Solution Logicielle de la fonction contrôleur	16
a) Fonction ADC	18
b) Mise en œuvre du programme de déplacement	19
9. Solution logicielle de la fonction affichage	23
a) Étude du contrôleur SSD1306 de l'afficheur OLED	23
i) Information SPI du SSD1306 :	24
ii) Graphic Display Data Ram (GDDRAM) :	25
iii) Validation de data ou de command vers SSD1306 :	26
iv) Commande de configuration du SSD1306 :	26
v) Mode d'adressage de la mémoire :	28
vi) Police de l'afficheur :	31
b) Fonction SPI	33
c) Fonction Timer	35
d) Mise en œuvre du programme d'affichage	37
Diagramme de Gantt	42



OFFOUGA Joris

SER2

Projet Véhicule Interactif



<i>Conclusion</i>	<i>43</i>
<i>Annexe</i>	<i>44</i>
10. <i>Logiciels utilisés</i>	<i>44</i>
11. <i>Programmes en C commentés</i>	<i>44</i>



*OFFOUGA Joris*

*SER2*

*Projet Véhicule Interactif*

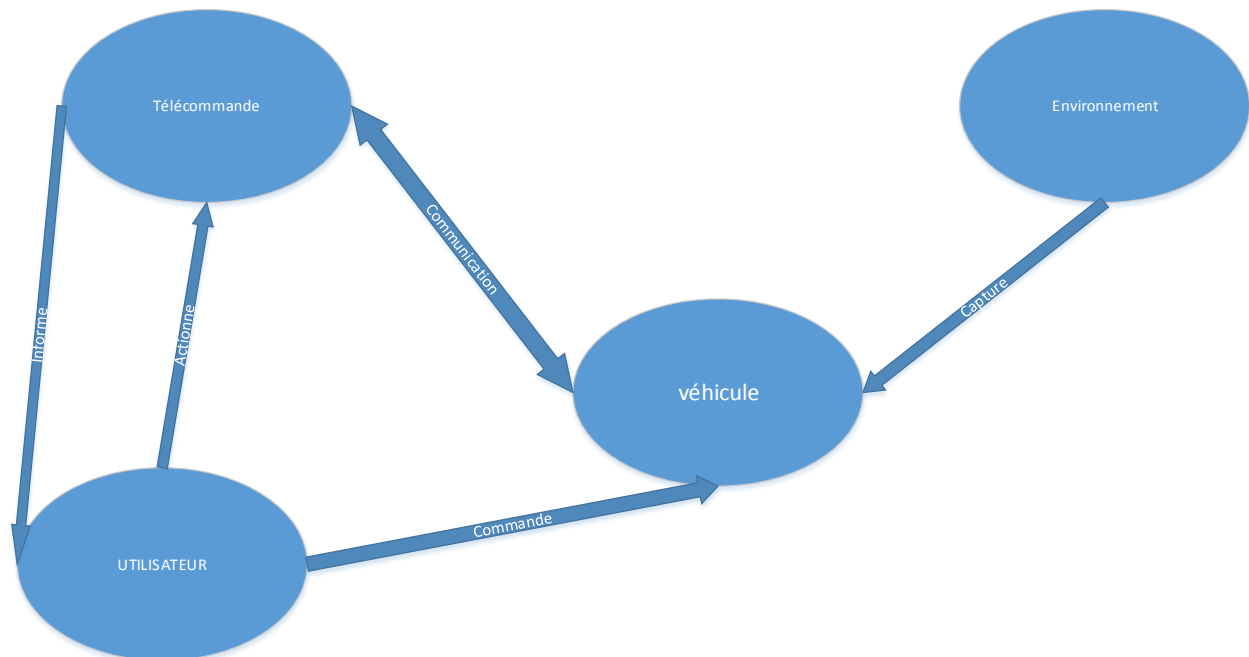


## Présentation

L'objectif du projet est de réaliser un véhicule capable d'interagir avec son environnement Il sera capable :

- D'effectuer des déplacements autonomes ou télécommandés en translation et sur 360°.
- D'éviter des obstacles.
- De transmettre son état de fonctionnement à une télécommande

Diagramme Sagittal :



Deux équipes ont été formés :

OT1 : Équipe Véhicule :

Fonctions à réaliser	En charge de la fonction
Alimentation/Informer	Rebillon Eric
Communication Radio	Elmernissi Yassine
Microcontrôleur/Gestion	Calliau Clément
Déplacement	Moulinard Jean
Environnement	Huet Samuel
Suivi de ligne	Jacquot Axel

OT2 : Équipe Télécommande

Fonctions à réaliser	En charge de la fonction
Alimentation	Renaud Pierre-Louis
Communication Radio	Elmernissi Yassine
Microcontrôleur/Gestion	Coutant Thomas
Contrôleur/Affichage	Offouga Joris

Cette partie a pour but de présenter les solutions mises en œuvre pour la partie Contrôleur et Affichage de OT2.

Une étude approfondie du cahiers des charges et des solutions mises en œuvre sera faite.



OFFOUGA Joris

SER2

Projet Véhicule Interactif



### Étude du cahier des charges

#### ■ Fonction principale : Contrôleur

Assure le contrôle du rover par l'intermédiaire d'un moyen de saisie des ordres utilisateurs.

Pour ce faire, il faut déterminer le système de saisies des ordres de l'utilisateur garantissant un déplacement à 360° du véhicule.

- Précision du cahier des charges :
  - Déplacement du véhicule en translations à savoir les axes X et Y ainsi que la possibilité d'effectuer des rotations sur l'axe Z.
  - Un système de saisies à faible consommation justifiant un fonctionnement sur batterie.
  - Commande de passage du mode autonome au mode radiocommandé.

#### ■ Fonction principale : Affichage

Assurer l'affichage des informations relatives à l'activité du rover et de la télécommande.

Pour ce faire, il faut déterminer un système d'affichage permettant de visualiser l'activité du rover et de la télécommande.

- Précision du cahier des charges :
  - Moyen d'affichage à faible consommation justifiant un fonctionnement sur batterie.
  - Encombrement minimal justifiant une prise en main unique de la télécommande.
  - Visualisation de la batterie du rover et de la télécommande ainsi que de son mode de fonctionnement.


### Recherche de solutions techniques

#### ■ Fonction principale : Contrôleur

Afin d'assurer au mieux la restitution des mouvements du rover via à la télécommande, nous allons passer en revue les différents moyens permettant la réalisation de cette fonction :

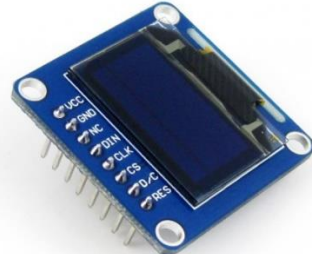
	<p><i>Mini-Joystick</i></p> <p><i>Avantages :</i> <i>Alimentation au choix</i> <i>Consommation &lt; 1mA</i> <i>Déplacement : Translation et rotation</i> <i>Interface : Analogique</i></p> <p><i>Avantages :</i> <i>Alimentation au choix</i> <i>Interface d'interprétation des valeurs par ADC</i> <i>Mise en œuvre logicielle</i></p>
	<p><i>Capteur Tactile</i></p> <p><i>Alimentation 2 à 5.5 V</i> <i>Consommation &lt; 1mA</i> <i>Déplacement : Translation et rotation</i></p> <p><i>Avantages :</i> <i>Consommation minimale</i></p> <p><i>Inconvénients :</i> <i>Informations quant à la récupération des données non communiqué.</i></p>

## Recherche de solutions techniques

	<p>Nunchuck</p> <p>Alimentation 3.3V Consommation 20mA Déplacement : Translation et rotation Interface : I2C Avantages : Récupération des données via un bus de communication Consommation minime Inconvénients : Encombrement maximale Solution inadaptée au cahier des charges</p>
---	--


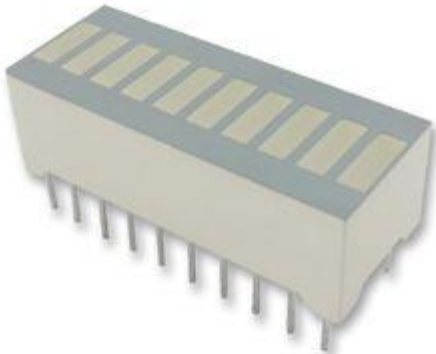
### ■ Fonction principale : Affichage

Afin d'assurer au mieux la restitution des informations du rover et de la télécommande via à celle-ci, nous allons passer en revue les différents moyens permettant la réalisation de cette fonction :

	<p>Interfaces SPI/I2C Texte bleu et jaune Alimentation : 3,3 à 5 V Résolution de 128 x 64 Angle de visibilité à 160° Consommation : 20 mA maximum Mode d'affiche : Graphique Taille de la Ram 1ko Prix : 8.23€ Avantages : Consommation minime Grand angle de vision Large gamme d'alimentation Inconvénients : Mise en œuvre logicielle</p>
---	--



## Recherche de solutions techniques

	<p><i>Afficheur LCD – 7segements :</i></p> <p><i>Alimentation : 5V</i></p> <p><i>Consommation : 20 mA par Led</i></p> <p><i>Mode d'affichage : Réflectif</i></p> <p><i>Prix : 7.66€</i></p> <p><i>Avantages :</i></p> <p><i>Mise en œuvre logicielle</i></p> <p><i>Inconvénients :</i></p> <p><i>Consommation énorme.</i></p>
	<p><i>Baragraphe LED :</i></p> <p><i>Alimentation : 2.2V</i></p> <p><i>Consommation : 20 mA par Led</i></p> <p><i>Prix : 2.60€</i></p> <p><i>Avantages :</i></p> <p><i>Mise en œuvre facile</i></p> <p><i>Inconvénients :</i></p> <p><i>Consommation énorme.</i></p> <p><i>Dans ce choix de solution, 1 par objet technique.</i></p> <p><i>Encombrement assez conséquent.</i></p>

## Recherche de solutions techniques



*Afficheur TFT-PROTO LCD-SPI-Parallèle-Écran tactile*

*Alimentation : 3.3V-5V pour le rétro-éclairage*

*Résolution 320x240*

*Taille en RAM : 262Ko*

*Mode d'affiche : Graphique*

*Consommation : 80 mA*

*Prix : 29,02 €*

*Avantages :*

*Consommation*

*Alimentation*

*Inconvénients :*

*Rétro-éclairage*

*Prix élevé*

*Mise en œuvre logicielle*



*Afficheur LCD Alphanumérique*

*Alimentation : 5V*

*Mode d'affichage : Transflectif*

*Interface : SPI*

*Consommation : 210 mA*

*Nombre de caractère /ligne : 20x2*

*Prix : 19,11 €*

*Angle de vue : 35°*

*Caractère accessible en RAM*

*Avantages :*

*Mise en œuvre logicielle*

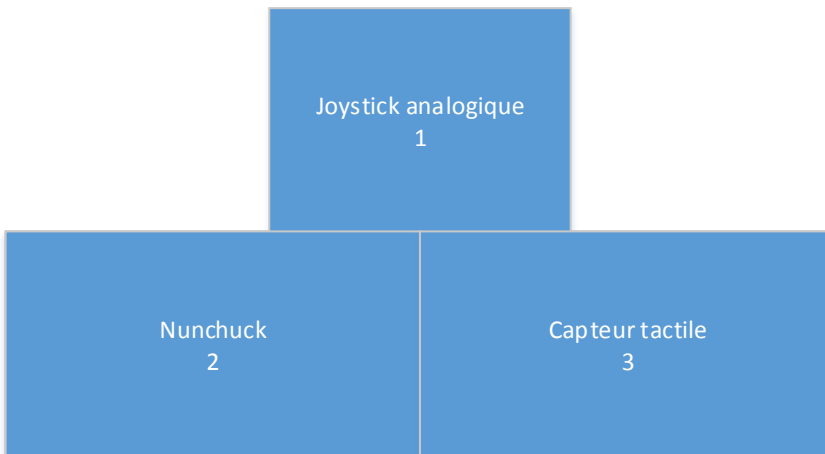
*Inconvénients :*

*Consommation*

*Taille de l'affichage*

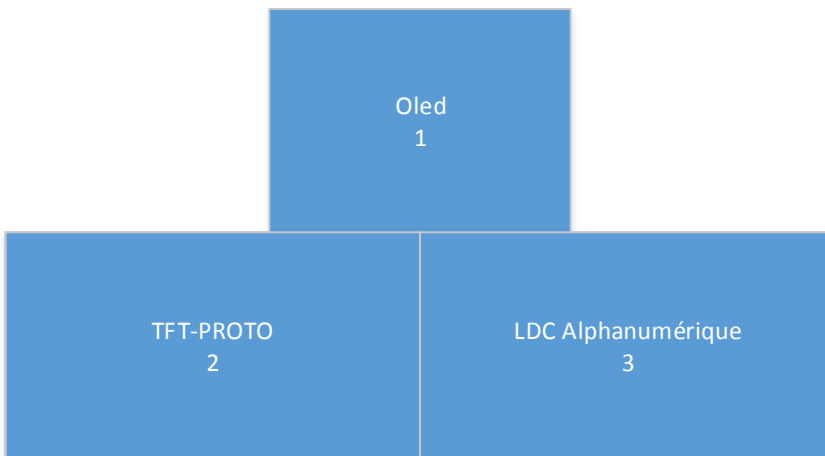
## Recherche de solutions techniques

*Podium Fonction Contrôleur :*



*La solution retenue sera un joystick analogique*

*Podium Fonction Affichage :*



*La solution retenue sera l'afficheur Oled*



OFFOUGA Joris

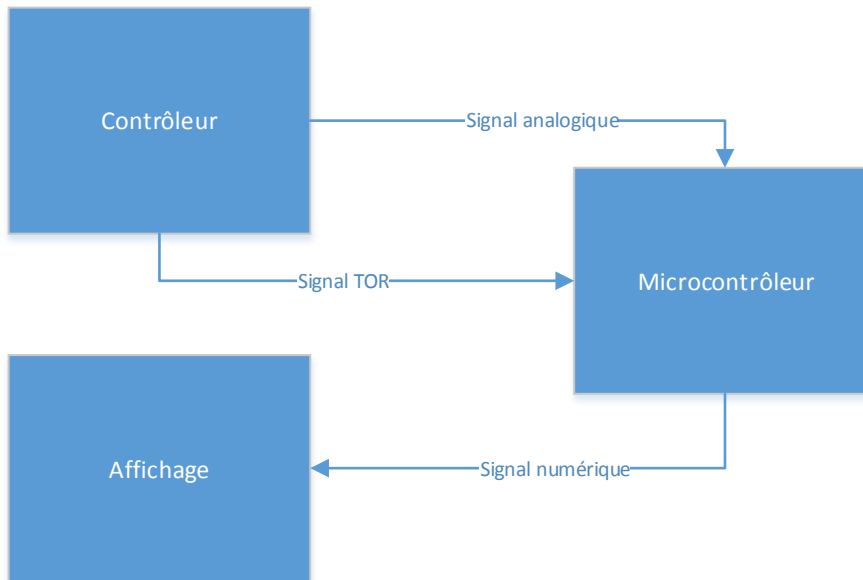
SER2

Projet Véhicule Interactif

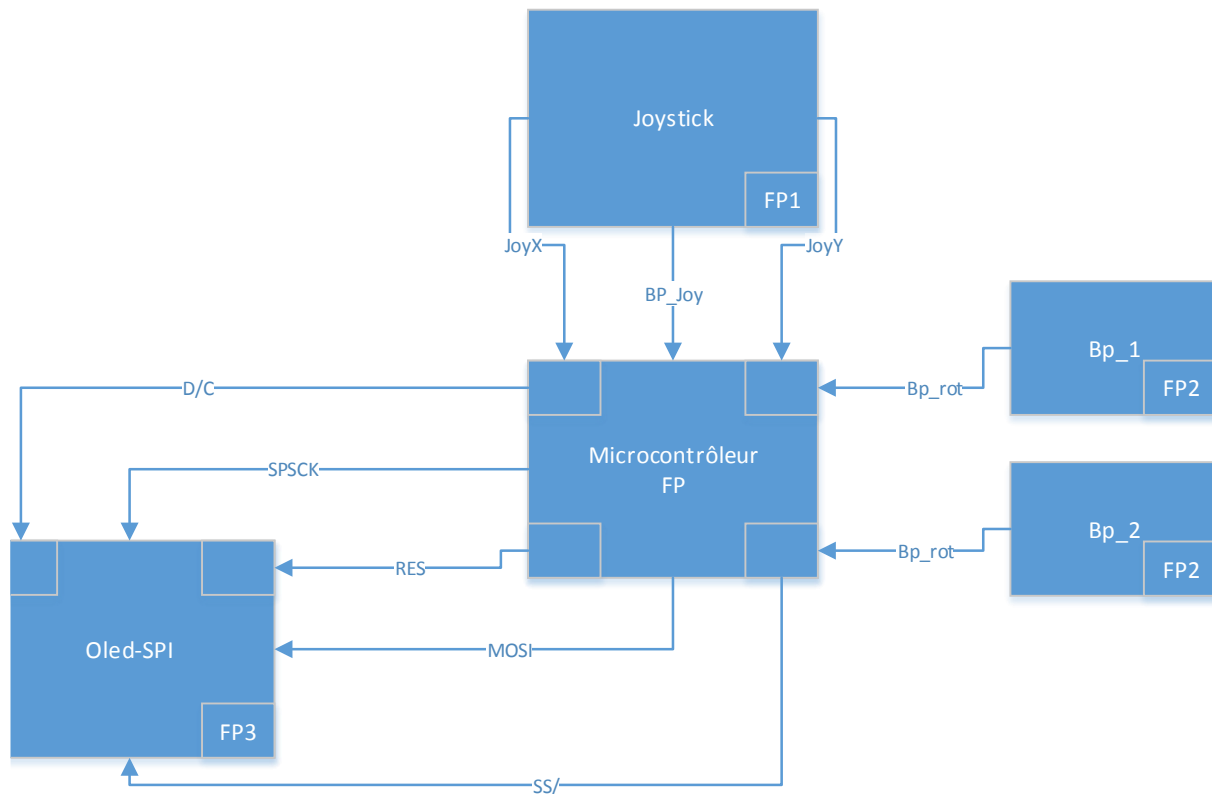


# Analyse Fonctionnelle

### 1. Schéma fonctionnel niveau 1



## 2. Schéma fonctionnel niveau 2



## 3. Analyse fonctionnelle

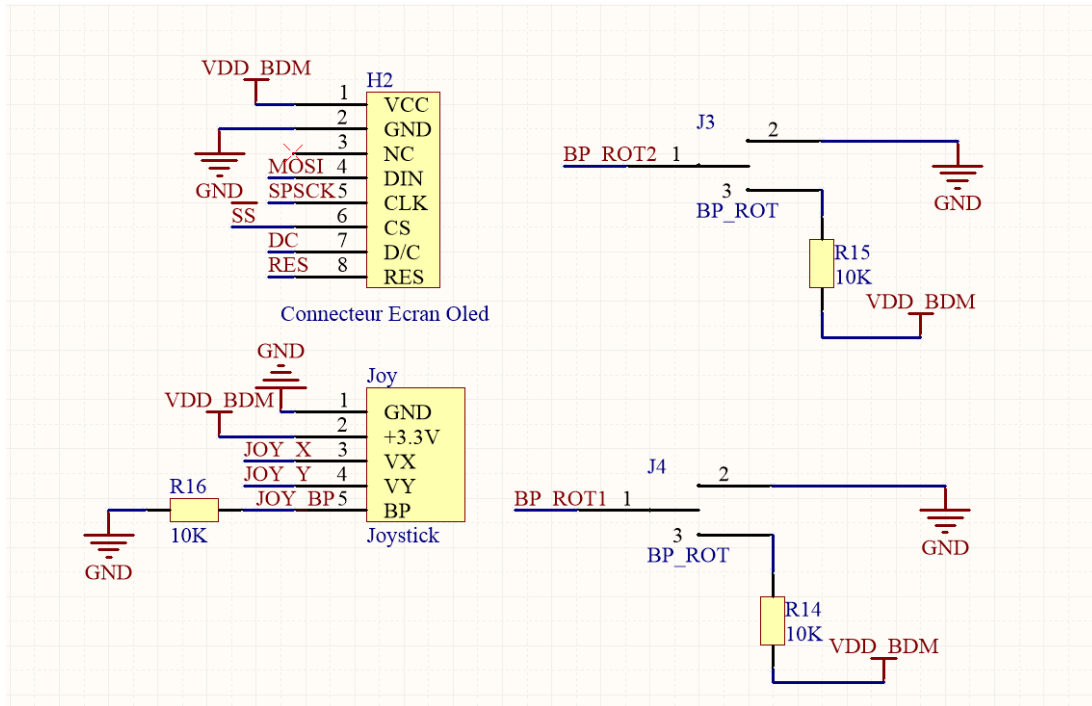
- *FP1 : Fonction permettant à l'utilisateur d'effectuer des mouvements sur l'axe X et Y. Garantissant aussi le mode de fonctionnement du véhicule.*
- *FP2 : Fonction permettant à l'utilisateur d'effectuer des rotations.*
- *FP3 : Fonction assurant l'affichage de la batterie du véhicule et de la télécommande.*
- *FP : Fonction de cadencement des programmes : Microcontrôleur MC9S08QE32 cadencé à 50MHz, sa fréquence de bus est de 25MHz*

#### 4. Description des signaux

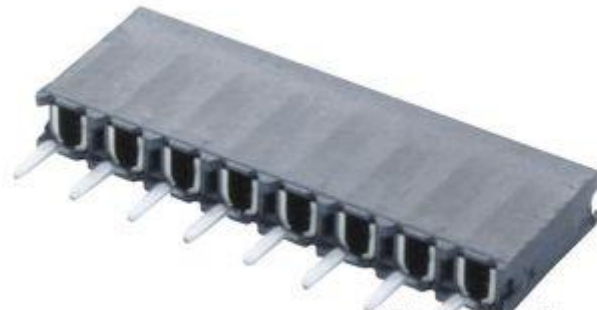
- *MOSI* : Signal carré d'envoi du maître vers l'esclave.
- *SPSCK* : Signal carré d'horloge entre le maître et l'esclave.
- *SS/* : Signal TOR sélectionnant l'esclave à qui s'adresser.
- *D/C* : Signal TOR d'envoi de commande ou de donnée vers l'afficheur.
- *RES* : Signal TOR de réinitialisation de l'afficheur.
- *BP rot* : Signal TOR permettant de réaliser des rotations
- *BP Joy* : Signal TOR changeant le mode de fonctionnement du véhicule
- *JoyX* : Signal analogique contenant a valeur de l'axe X
- *JoyY* : Signal analogique contenant a valeur de l'axe Y

## Analyse Structurelle

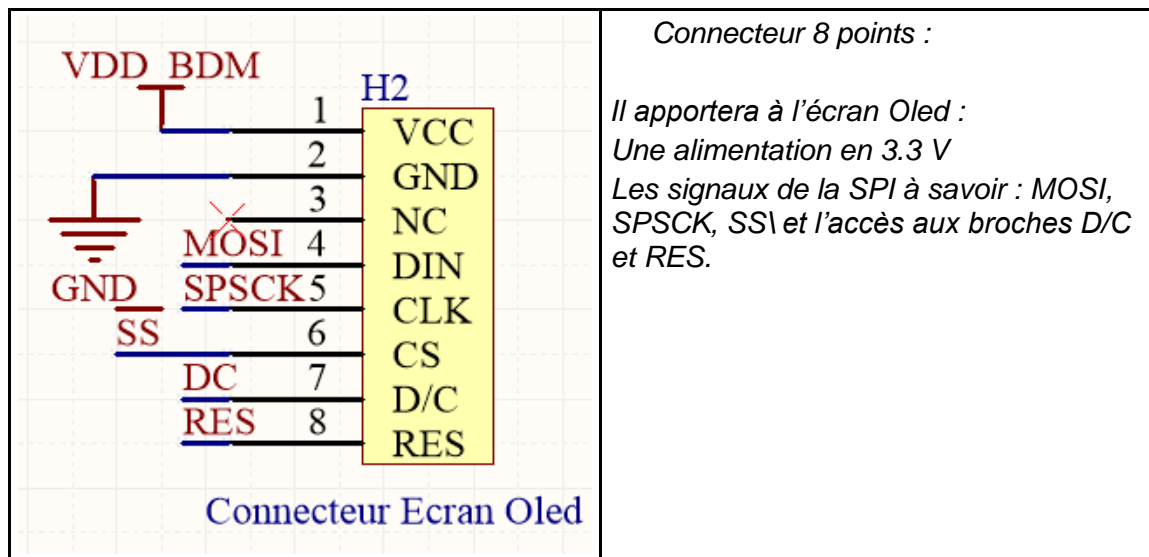
### 5. Schéma structurel



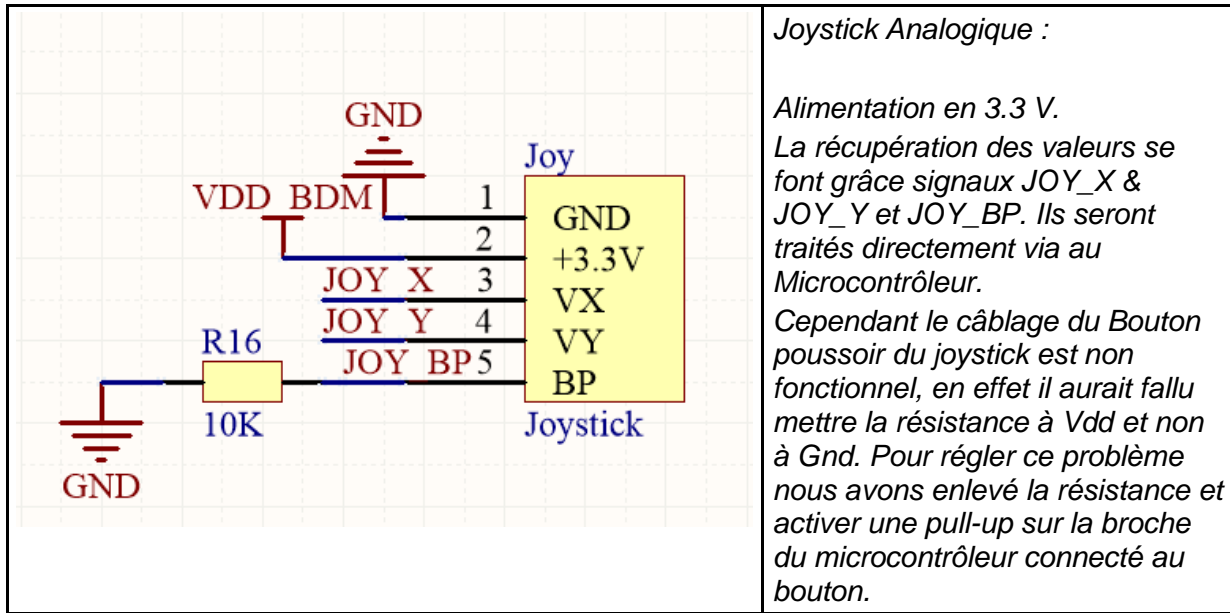
### 6. Analyse structurelle

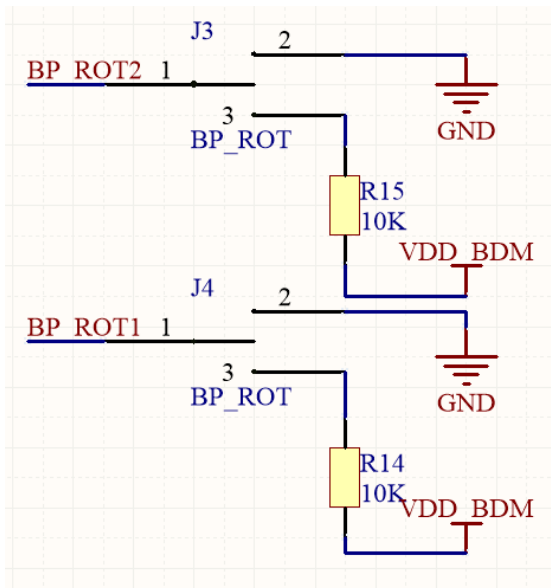


© Premier Farnell  
Copying of image is prohibited









Bouton Poussoirs :

Les deux boutons poussoirs servant à la rotation sont câblés en inverse.

## 7. Nomenclature

Description	Caractéristique	Boîtier	Fournisseur	Référence Fabricant	Code commande	Prix UHT	Quantité	Lot
Affichage OLED 0,96	SPI/I2C, 3,3 à 5 V , résolution 128*64		Robotshop	9092	RB-Wav-27	6,860 €	1	
Resistance	10kOhm, Tolérance +1%, tension 150V	0805 [2012 Metric]	Farnell	MC01W0805110K	9332391	0,002 €	3	1 de 10
Boutton Poussoir	Pouvoir de coupure : 10mA sous 24 Vcc		Farnell	623300.000.000.060	143637	5,930 €	2	
Joystick			Robotshop	IM130330001	RB-lte-55	1,610 €	1	
Connecteur	Barrette femelle au pas de 2,54 mm		Farnell	BCS-108-L-S-TE	1593463	0,949 €	1	
						Prix Total HT	21,281 €	
						Prix Total TTC	25,537 €	



OFFOUGA Joris

SER2

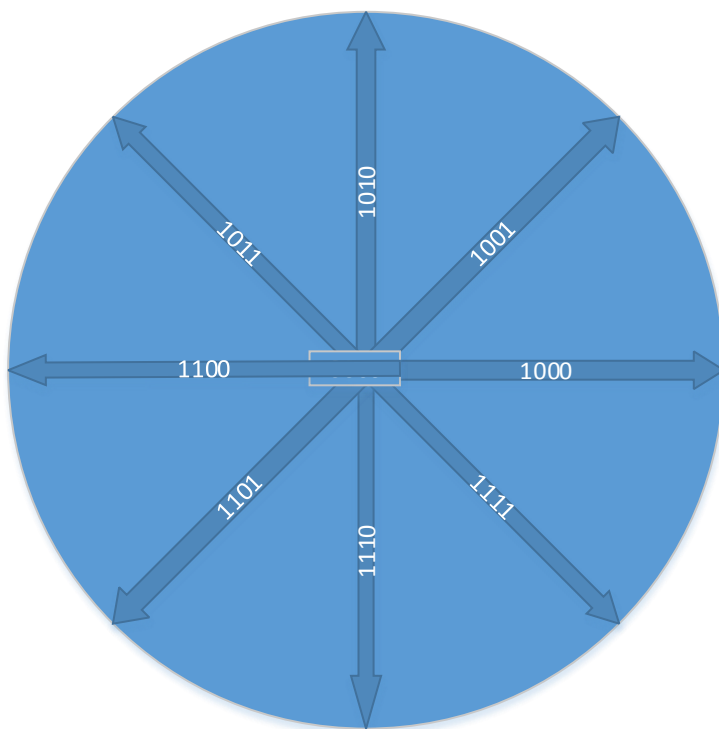
Projet Véhicule Interactif



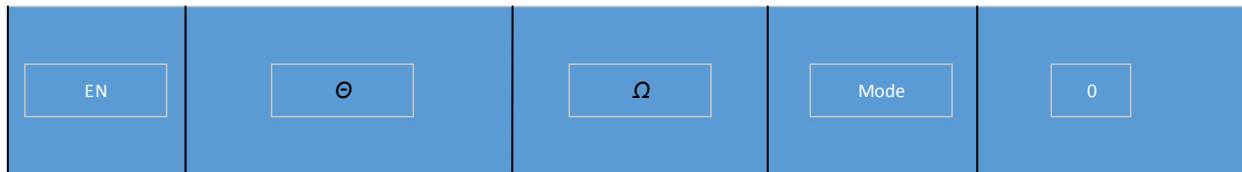
## Étude Logicielle

### 8. Solution Logicielle de la fonction contrôleur

Pour satisfaire les déplacements du rover, nous devons mettre en œuvre une solution logicielle avec l'aide du convertisseur analogique et de ports I/O du microcontrôleur de la télécommande. Afin que les déplacements du véhicule soient retranscrits via à la télécommande, nous avons convenu à l'aide d'un cercle, une table de vérité permettant de définir une série de commande à transmettre par radio au véhicule.



EN	$\Theta 0$	$\Theta 1$	$\Theta 2$	$\Omega 0$	$\Omega 1$	M	0
0	0	0	0	x	x	x	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
1	0	0	0	x	x	0	0
1	0	0	1	x	x	0	0
1	0	1	0	x	x	0	0
1	0	1	1	x	x	0	0
1	1	0	0	x	x	0	0
1	1	0	1	x	x	0	0
1	1	1	0	x	x	0	0
1	1	1	1	x	x	0	0



*EN [7] : Bit de déplacement du véhicule. A 1 active les déplacements du véhicule.*

*Θ [6 :4] : Bit de translation du véhicule. Ces trois bits constituent les déplacements en translation et en diagonale du véhicule.*

*Ω [3 :2] : Bit de rotation du véhicule. Ces deux bits définissent une rotation chacun.*

*M [1] : Bit de mode de fonctionnement du véhicule. À 0 le véhicule est en mode radiocommandé.*

*0 [0] : Bit non utilisé*

*L'utilisation de la table de vérité nous a permis de définir un octet unique permettant définir les déplacements à effectuer et connaître son mode de fonctionnement.*

*Tableau de commande du véhicule :*

0x00	Pas de déplacement
0x04	Rotation vers la gauche
0x08	Rotation vers la droite
0x80	Translation horizontale vers la droite
0x90	Translation diagonale vers la droite (+45°)
0xA0	Translation verticale vers le haut
0xB0	Translation diagonale vers la gauche (+45°)
0xC0	Translation horizontale vers la gauche
0xD0	Translation diagonale vers la gauche (-45°)
0xE0	Translation verticale vers le bas
0xF0	Translation diagonale vers la droite (-45°)

### a) Fonction ADC

Étude de registre de la fonction ADC du microcontrôleur MC9S08QE32 :

Registre ADCSC1 :



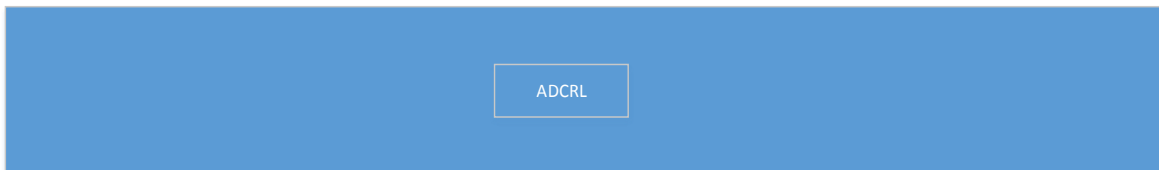
**COCO [1]** : Indicateur de fin de conversion. A 1 lorsque la conversion est terminée remise à zéro du bit par lecture du registre ADCRL.

**ADCO [1]** : Bit d'activation de la fonction ADC. A 1 active la fonction ADC

**ADCH [4 :0]** : Registre de sélection de la voie à convertir. Nous avons deux voies à convertir à savoir la voie ADP2 et ADP3.

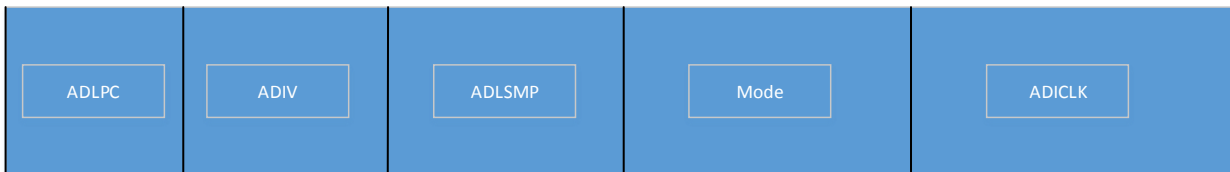
Les autres bits de ce registre sont laissés dans leur état par défaut.

Registre ADCRL :



**ADCRL** : Registre 8 bits contenant le résultat de la conversion.

Registre ADCCFG :



*ADIV [6 :5] : Configure la division de de la fréquence d'échantillonnage. A 11 divise la fréquence d'échantillonnage par 8.*

*Mode [3 :2] : Configure le mode de résolution à savoir 12/10/8 bits. A 00 configure l'ADC en 8 bits de résolution.*

*ADICLK [1 :0] : Sélectionne la source d'horloge de la fréquence d'échantillonnage. A 00 configure la fréquence de bus du microcontrôleur comme fréquence d'échantillonnage.*

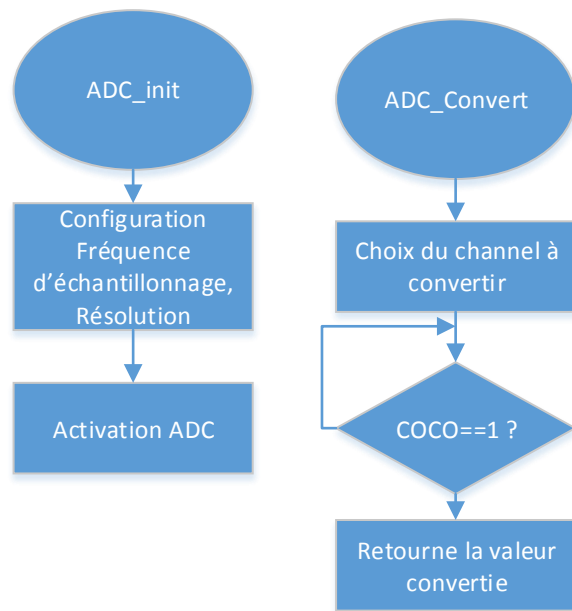
*Les autres bits de ce registre seront laissés dans leur état par défaut.*

### *b) Mise en œuvre du programme de déplacement*

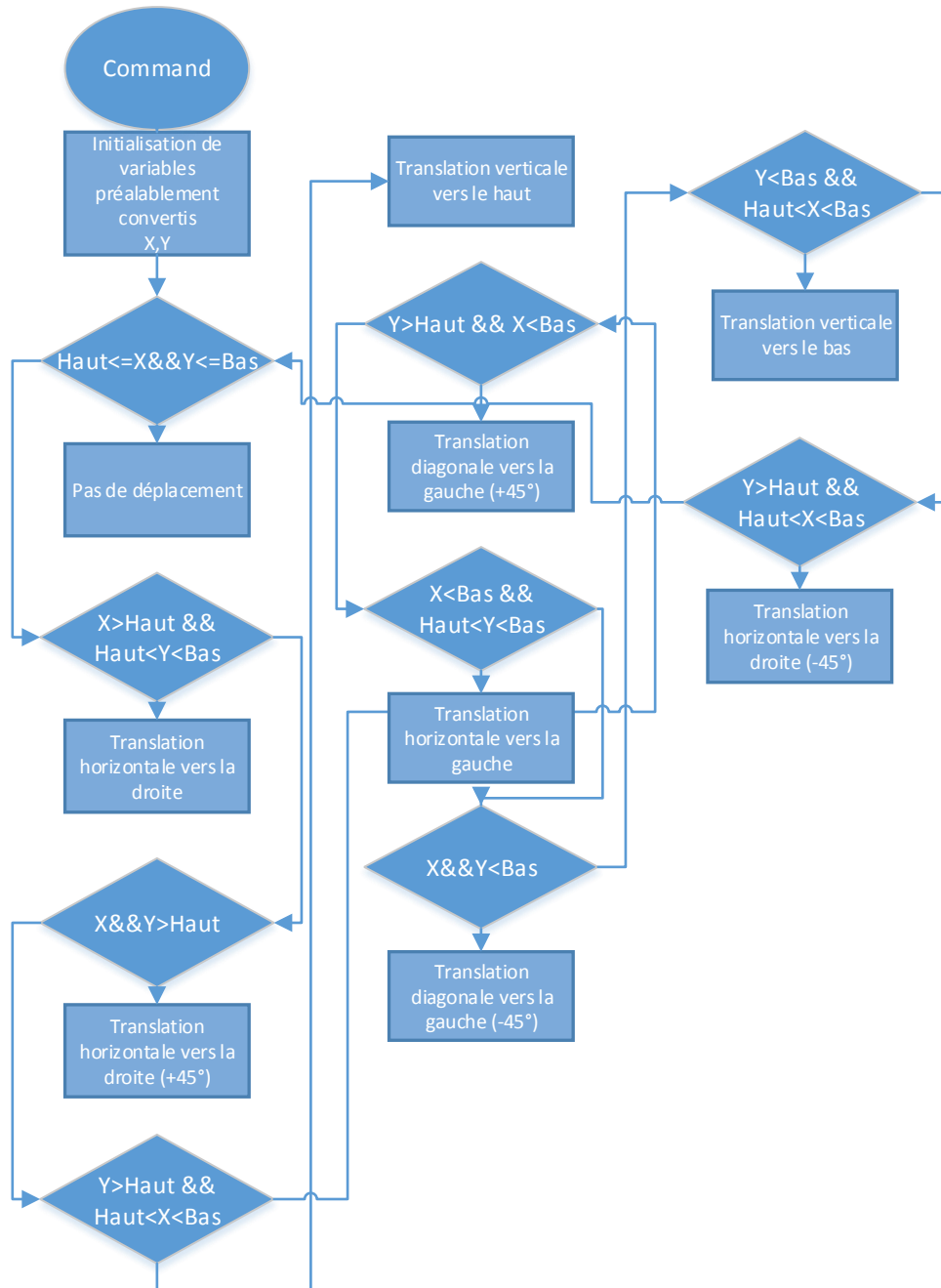
*Le véhicule se déplaçant à vitesse constante, ce programme ne prendra pas en compte une quelque conque variation de vitesse, il ne s'occupera que de définir des angles de déplacement via au tableau de commande. Nous allons donc borner les déplacements en fonction des valeurs transmises via au convertisseur analogique numérique. Nous allons fixer deux valeurs arbitraires, une limite haute et une limite basse. Chaque déplacement aura des bornes bien précises pour s'effectuer.*

*Sachant qu'un unsigned char contient un 1 octet de 0 à 255, ce sont les valeurs que prendront les deux axes du joystick et selon sa position étant donné que c'est un diviseur de tension, il possède des valeurs bien particulières. Nous utilisons 9 positions, nous bornerons le joystick de manière à ce qu'il ne soit capable que d'exécuter ses positions.*

Algorithme ADC :

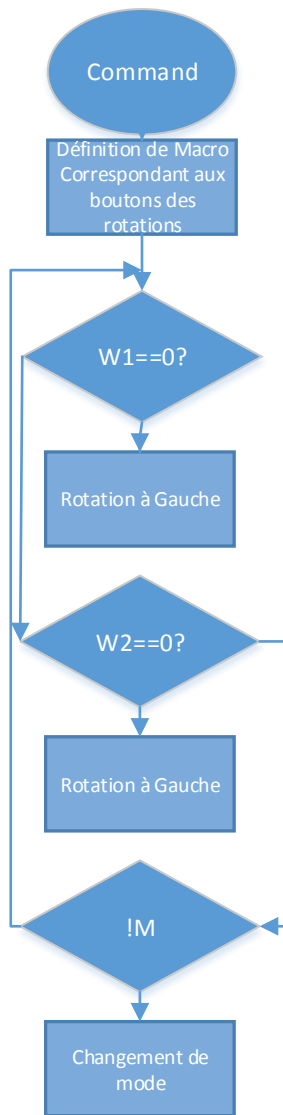


## Algorithme de déplacement :





Algorithme des boutons :



### 9. Solution logicielle de la fonction affichage

#### a) Étude du contrôleur SSD1306 de l'afficheur OLED

Nous allons tout d'abord étudier les configurations du contrôleur :

SSD1306 dispose de deux interfaces de communications : SPI & I2C

Le choix de l'interface de communication se fait à l'aide de deux jumpers au dos du module.

Configuration :

	BS1/BS0	CS	D/C	DIN	CLK
3-wire SPI	0/1	CS	0	MOSI	SCLK
4-wire SPI	0/0	CS	D/C		
I2C	1/0	0	0/1	SDA	SCL

Le mode de défaut de l'afficheur est le mode 4-wire SPI, c'est ce mode que nous allons utiliser aussi il nous faut utiliser la broche D/C (Data or Command).

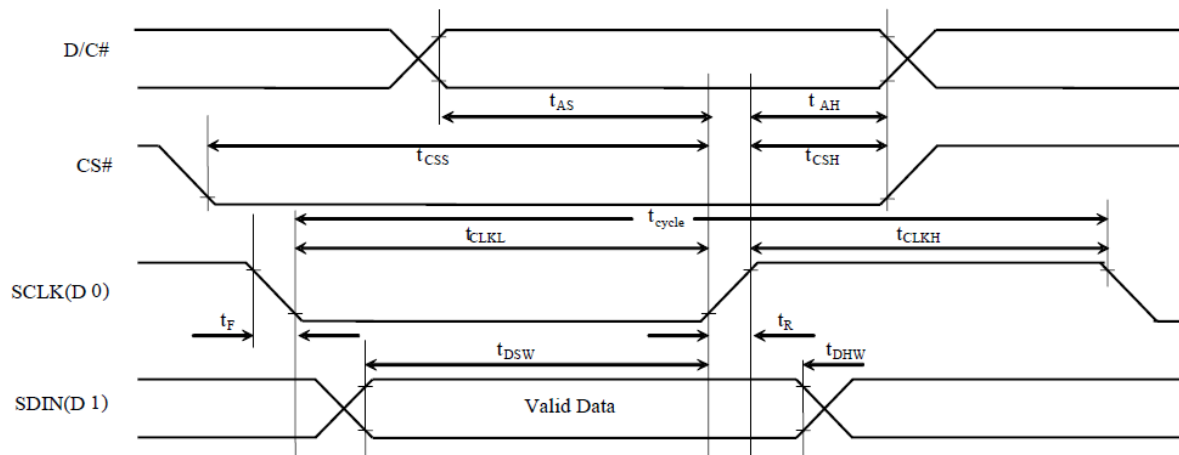
## i) Information SPI du SSD1306 :

Table 13-4 : 4-wire Serial Interface Timing Characteristics

( $V_{DD} - V_{SS} = 1.65V$  to  $3.3V$ ,  $T_A = 25^\circ C$ )

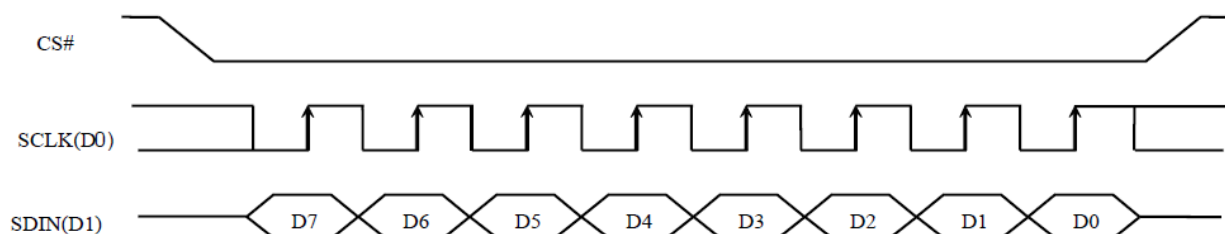
Symbol	Parameter	Min	Typ	Max	Unit
$t_{cycle}$	Clock Cycle Time	100	-	-	ns
$t_{AS}$	Address Setup Time	15	-	-	ns
$t_{AH}$	Address Hold Time	15	-	-	ns
$t_{CSS}$	Chip Select Setup Time	20	-	-	ns
$t_{CSH}$	Chip Select Hold Time	10	-	-	ns
$t_{DSW}$	Write Data Setup Time	15	-	-	ns
$t_{DHW}$	Write Data Hold Time	15	-	-	ns
$t_{CLKL}$	Clock Low Time	20	-	-	ns
$t_{CLKH}$	Clock High Time	20	-	-	ns
$t_R$	Rise Time	-	-	40	ns
$t_F$	Fall Time	-	-	40	ns

Figure 13-3 : 4-wire Serial interface characteristics



De ce document nous pouvons tirer comme information la fréquence minimum de cadencement de l'afficheur est de :  $CLkl = \frac{1}{t_{clk}} = \frac{1}{100 \times 10^{-9}} = 10MHz$

Format de la tram SPI :



Grâce au chronogramme, nous pouvons remarquer que la phase et la polarité de l'horloge sont tous deux à 1 et que la lecture des données se fait en MSB first.

ii) Graphic Display Data Ram (GDDRAM) :

Le GDDRAM est une RAM statique de type bitmap contenant le motif de bits à afficher. La taille de la RAM est de 128 x 64 bits à savoir de 1 ko de RAM et la RAM est divisée en huit pages, de PAGE0 à PAGE7, qui sont utilisées pour l'affichage matricielle.

Figure 8-13 : GDDRAM pages structure of SSD1306

PAGE0 (COM0-COM7)	Page 0	Row re-mapping PAGE0 (COM 63-COM56)
PAGE1 (COM8-COM15)	Page 1	PAGE1 (COM 55-COM48)
PAGE2 (COM16-COM23)	Page 2	PAGE2 (COM47-COM40)
PAGE3 (COM24-COM31)	Page 3	PAGE3 (COM39-COM32)
PAGE4 (COM32-COM39)	Page 4	PAGE4 (COM31-COM24)
PAGE5 (COM40-COM47)	Page 5	PAGE5 (COM23-COM16)
PAGE6 (COM48-COM55)	Page 6	PAGE6 (COM15-COM8)
PAGE7 (COM56-COM63)	Page 7	PAGE7 (COM 7-COM0)

Column re-mapping  
SEG0 -----SEG127  
SEG127 -----SEG0

Lorsqu'un octet de données est écrit dans GDDRAM, toutes les données d'image des lignes de la même page sont remplies par le pointeur d'adresse de colonne qui s'incrémente à chaque lecture de donnée.



OFFOUGA Joris

SER2

Projet Véhicule Interactif



iii) Validation de data ou de command vers SSD1306 :

D/C	Description	Incrémentation d'adresse
0	Lecture commande	Non
1	Écriture donnée	Oui

iv) Commande de configuration du SSD1306 :

D/C	HEX	D7	D6	D5	D4	D3	D2	D1	D0	Command	Description
0 0	81 A[7 :0]	1 A7	0 A6	0 A5	0 A4	0 A3	0 A2	0 A1	1 A0	Définir le Contrast de l'afficheur	Double byte de commande permettant de définir le contrast, plus la valeur est élevée plus le contrast l'est. Valeur par défaut 7F
0	A4/A5	1	0	1	0	0	1	0	X0	Affichage	A4, X0=0 : L'afficheur suit le contenu de la RAM. A5, X0=1 : L'afficheur ignore le contenu de la ram
0	A6/A7	1	0	1	0	0	1	1	X0	Affichage normale/inversée	A6, X0=0 : Affichage normale (valeur par défaut) A7, X0=1 : Affichage inversée
0	AE/AF	1	0	1	0	1	1	1	X0	Affichage On/OFF	AE, X0=0 : Affichage OFF (valeur par défaut)  AF, X0=1 : Affichage ON

0	40~7F	0	1	X5	X4	X3	X2	X1	X0	Définir la ligne de démarrage de l'affichage	Définir l'enregistrement de l'affichage de la mémoire vive de 0-63 en utilisant X5X3X2X1X0.
0	A0/A1	1	0	1	0	0	0	0	X0	Réorganiser le segment	A0, X0=0 : L'adresse de colonne commence à 0 (Valeur par défaut). A1, X0=1 : L'adresse de colonne commence à 127.
0 0	A8 A[5 :0]	1 *	0 *	1 A5	0 A4	1 A3	0 A2	0 A1	0 A0	Définir le rapport multiplex	Définir le rapport MUX à N + 1 MUX N = A[5:0]: de 16MUX à 64MUX, RESET=111111 (c'est-à-dire 63, 64MUX) A[5:0] de 0 à 14 sont invalides.
0	C0/C8	1	1	0	0	X3	0	0	0	Définir la sortie COM Direction de balayage	C0h, X3= 0 : mode normal (RESET)direction à partir de COM0 à COM [N -1] C8h, X3 = 1 : mode remappé. Direction depuis COM [N-1] à COM0 Où N est le rapport Multiplex.
0 0	D3 A[5 :0]	1	1	0	1	0	0	1	1	Définir l'offset de l'affichage	Définir le décalage vertical par COM de 0 ~ 63 La valeur par défaut est 00.
0 0	DA A[5 :4]	1 0	1 0	0 A5	1 A4	1 0	0 0	1 1	0 0	Configuration des pins COM	A [4] = 0b, configuration de broche COM séquentielle A [4] = 1b (RESET), alternative COM broche

											Configuration A [5] = 0b (RESET), Désactiver COM Gauche / Droite Remappé A [5] = 1b, Activer COM Gauche / Droite remap
0 0	8D A[7 :0]	1 *	0 *	0 0	0 1	1 0	1 A2	0 0	1 0	Configuration de la pompe de charge	A2= 0 : Désactive la pompe de charge (Valeur par défaut) A2 = 1, Active la pompe de charge La pompe de charge doit être Activé par la commande suivante : 8D : Réglage de la pompe de charge 14 : Activer la pompe de charge AF : Affichage ON

v) Mode d'adressage de la mémoire :

Il existe 3 modes d'adressage mémoire différents dans SSD1306 : mode d'adressage de page, mode d'adressage horizontal et mode d'adressage vertical. Cette commande définit la voie de l'adressage de mémoire dans l'un des éléments ci-dessus

Trois modes. Dans ce cas, "COL" désigne la colonne de RAM des données d'affichage graphique.

Mode d'adressage de page :

En mode d'adressage des pages, après l'affichage, la RAM est lue / écrite, le pointeur de l'adresse de la colonne est augmenté automatiquement par 1. Si le pointeur d'adresse de colonne atteint l'adresse de la fin de colonne, le pointeur d'adresse de colonne est réinitialisé à l'adresse de début de colonne et le pointeur d'adresse de page n'est pas modifié. Il faut donc

définir une nouvelle page et les adresses des colonnes afin d'accéder au contenu de la RAM de la page suivante.

Illustration :

	COL0	COL 1	....	COL 126	COL 127
PAGE0	→	→	→	→	→
PAGE1	→	→	→	→	→
:	:	:	:	:	:
PAGE6	→	→	→	→	→
PAGE7	→	→	→	→	→

En mode lecture normale ou en écriture et en mode d'adressage de page, les étapes suivantes sont nécessaires pour définir l'emplacement du pointeur d'accès à la RAM :

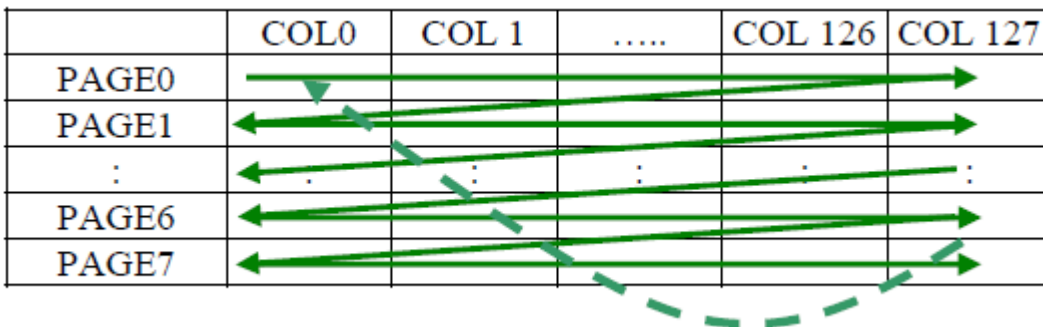
- Définir l'adresse de début de page de l'emplacement d'affichage cible par la commande 0xB0 à 0xB7.
- Définir l'adresse de la colonne de départ inférieure du pointeur par la commande 0x00 ~ 0x0F.
- Définir l'adresse de la colonne de départ supérieure du pointeur par la commande 0x10 ~ 0x1F.

Mode d'adressage horizontal & verticale :

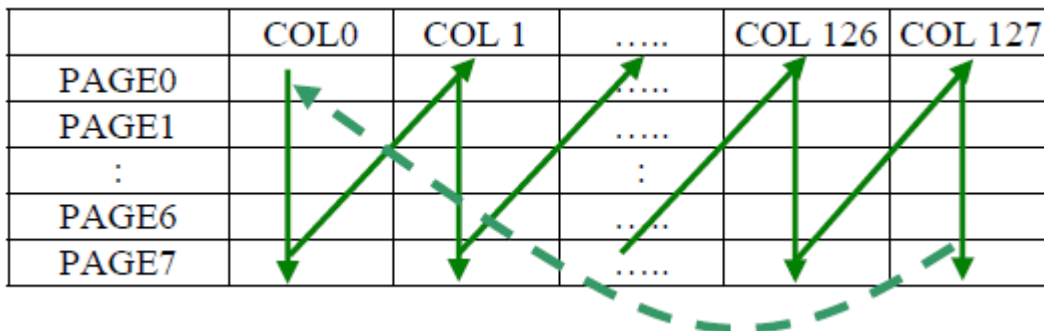
En mode d'adressage horizontal ou verticale, après l'affichage, la RAM est lue / écrite, le pointeur de l'adresse de la colonne est augmenté automatiquement par 1. Si le pointeur d'adresse de colonne atteint l'adresse de fin de colonne, le pointeur d'adresse de colonne est réinitialisé à l'adresse de début de colonne et le pointeur d'adresse de page augmente de 1. Lorsque les deux colonnes et les pointeurs d'adresse de page atteignent l'adresse de fin, les pointeurs sont réinitialisés à l'adresse de début de colonne et au début de page Adresse.



Mode d'adressage Horizontal :



Mode d'adressage Verticale :



En mode d'affichage normal d'écriture RAM ou écriture et mode d'adressage horizontal ou vertical, les étapes suivantes sont requises pour définir l'emplacement du pointeur d'accès RAM :

- Définir l'adresse de début et de fin de la colonne de l'emplacement d'affichage cible par commande 0x21.
- Définir l'adresse de début et de fin de la page de l'emplacement d'affichage cible par commande 0x22.

Pour notre mise en œuvre, nous utiliserons le mode d'adressage de page, ce mode d'adressage nous permettra d'écrire sur des zones bien précises de l'afficheur.



Le logiciel sépare la police en un tableau de caractères, un tableau renseignant le nombre de pixels correspondant à chaque caractère.

Ainsi nous pouvons générer toutes les polices connues pour l'afficheur, nous sommes partis sur le style courier car les caractères sont espacés entre eux équitablement ce qui permet une lisibilité limpide. Ne trouvant pas pratique le fait que les deux tableaux soient séparés, nous avons réuni les deux tableaux en un seul et ainsi créé cet header :

```
# * font.h
#ifndef FONT_H
#define FONT_H
/*tableau ayant en parametre la longueur du caractere et sa definition en pixels*/
typedef struct{
    const unsigned char wide;
    const unsigned char tab[7];
}lettre;

/*tableau des caracteres*/
const lettre Courier[]={
    {5,0b00000000,0b00000000}, // ' '
    {1,0b01011111}, // '!'
    {3,0b00000011,0b00000000,0b00000011}, // '"'
    {5,0b01000100,0b01111110,0b01000101,0b01111110,0b00100101}, // '#'
    {4,0b00100100,0b11010101,0b00101011,0b00010010}, // '$'
    {4,0b00001010,0b00101101,0b01011010,0b00101000}, // '%'
    {5,0b00110000,0b01001100,0b01010010,0b01110010,0b01000000}, // '&'
    {1,0b00000111}, // ''
    {2,0b00111100,0b11000011}, // '{'
    {2,0b11000011,0b00111100}, // '}'
    {5,0b00000010,0b00001010,0b00000011,0b00001010,0b00000010}, //
    {5,0b00001000,0b00001000,0b01111111,0b00001000,0b00001000}, //
    {3,0b10000000,0b11000000,0b01000000}, // ','
    {5,0b00001000,0b00001000,0b00001000,0b00001000,0b00001000}, // '-'
    {2,0b10000000,0b01000000}, // '~'
    {5,0b10000000,0b01100000,0b00011000,0b00000110,0b00000001}, // '/'
    {5,0b00111110,0b01000001,0b01000001,0b01000001,0b00111110}, // '0'
    {5,0b01000000,0b01000001,0b01111111,0b01000000,0b01000000}, // '1'
    {5,0b01100010,0b01010001,0b01001001,0b01000101,0b01000010}, // '2'
    {5,0b00100010,0b01000001,0b01001001,0b01001001,0b00110110}, // '3'
    {5,0b00011000,0b00010100,0b01010010,0b01111111,0b01010000}, // '4'
    {5,0b00100000,0b01001111,0b01001001,0b01001001,0b00110001}, // '5'
    {5,0b00111100,0b01001010,0b01001001,0b01001001,0b00110001}, // '6'
    {5,0b00000011,0b00000001,0b01100001,0b00011001,0b00000111}, // '7'
    {5,0b00110110,0b01001001,0b01001001,0b01001001,0b00110110}, // '8'
    {5,0b01000110,0b01001001,0b01001001,0b00101001,0b00011110}, // '9'
    {2,0b01000100,0b01000100}, // ':'
    {5,0b00001000,0b00001000,0b00010100,0b00010100,0b00100010}, // '>'
    {5,0b00010100,0b00010100,0b00010100,0b00010100,0b00010100}, // '='
    {5,0b00100010,0b00010100,0b00010100,0b00001000,0b00001000}, // '>'
    {4,0b00000010,0b01000001,0b01011001,0b00000110}, // '?'
    {5,0b01111110,0b01000001,0b01001001,0b01001001,0b01111110}, // '@'
    {7,0b01000000,0b01011101,0b00010011,0b01011100,0b01000000,0b01000000}, // 'A'
    {6,0b01000001,0b01111111,0b01001001,0b01001001,0b01001001,0b00110110}, // 'B'
    {5,0b00111110,0b01000001,0b01000001,0b01000001,0b00100011}, // 'C'
    {6,0b01000001,0b01111111,0b01000001,0b01000001,0b00100010,0b00011100}, // 'D'
    {5,0b01000001,0b01111111,0b01001001,0b01011101,0b01100011}, // 'E'
    {5,0b01000001,0b01111111,0b01001001,0b00011101,0b00000011}, // 'F'
    {6,0b00111110,0b01000001,0b01000001,0b01010001,0b00110011,0b00010000}, // 'G'
    {7,0b01000001,0b01111111,0b01001001,0b00001000,0b01001001,0b01111111,0b01000001}, // 'H'
    {5,0b01000001,0b01000001,0b01111111,0b01000001,0b01000001}, // 'I'
    {5,0b00110000,0b01000001,0b01000001,0b00111111,0b00000001}, // 'J'
    {7,0b01000001,0b00111111,0b01001001,0b00001000,0b00010101,0b01100011,0b01000001}, // 'K'
    {5,0b01000001,0b01111111,0b01000001,0b01000000,0b01110000}, // 'L'
    {7,0b01000001,0b01111111,0b01000111,0b00001000,0b01000111,0b01111111,0b01000001}, // 'M'
    {7,0b01000001,0b01111111,0b01000111,0b00011000,0b01100001,0b01111111,0b00000001}, // 'N'
    {5,0b00111110,0b01000001,0b01000001,0b01000001,0b00111110}, // 'O'
    {5,0b01000001,0b01111111,0b01010001,0b00010001,0b00001110}, // 'P'
    {5,0b00111110,0b01000001,0b11000001,0b11000001,0b10111110}, // 'Q'
    {6,0b01000001,0b01111111,0b01010001,0b00010001,0b00101110,0b01000000}, // 'R'
    {5,0b01100110,0b01001001,0b01001001,0b01001010,0b00110011}, // 'S'
    {5,0b00000011,0b01000001,0b01111111,0b01000001,0b00000011}, // 'T'
    {7,0b00000001,0b00111111,0b01000001,0b01000000,0b01000001,0b00111111,0b00000001}, // 'U'
    {7,0b00000001,0b00000111,0b01110001,0b01000000,0b001111001,0b00000011,0b00000001}, // 'V'
    {7,0b00000001,0b00111111,0b01000001,0b00111100,0b01000001,0b00111111,0b00000001}, // 'W'
    {7,0b01000001,0b01100011,0b00010100,0b00001000,0b00010100,0b01100011,0b01000001}, // 'X'
    {5,0b01000011,0b01010001,0b01000101,0b01111000,0b01000101,0b00000011,0b00000001}, // 'Y'
    {2,0b11111111,0b10000001}, // 'Z'
    {4,0b00000001,0b00000110,0b00111000,0b11000000}, // '['
    {2,0b10000001,0b11111111}, // ']'
    {5,0b00000100,0b00000010,0b00000001,0b00000010,0b00000100}, // '^'
    {7,0b00000000,0b00000000,0b00000000,0b00000000,0b00000000,0b00000000,0b00000000}, // '_'
    {2,0b00000001,0b00000010}, // '*'
    {6,0b01010000,0b01010100,0b01010100,0b01010100,0b01110000,0b01000000}, // 'a'

```

```

{6,0b01000001, 0b01111111, 0b01001000, 0b01000100, 0b01000100, 0b00111000}, // 'b'
{5,0b00111000, 0b01000100, 0b01000100, 0b01000100, 0b01001100}, // 'c'
{6,0b00111000, 0b01000100, 0b01000100, 0b01001001, 0b01111111, 0b01000000}, // 'd'
{5,0b00111000, 0b01010100, 0b01010100, 0b01010100, 0b01011000}, // 'e'
{4,0b01000100, 0b01111110, 0b01000101, 0b01000101}, // 'f'
{6,0b00111000, 0b01000100, 0b01000100, 0b01001000, 0b11111100, 0b00000100}, // 'g'
{7,0b01000001, 0b01111111, 0b01001000, 0b00000100, 0b01000100, 0b01111000, 0b01000000}, // 'h'
{5,0b01000100, 0b01000100, 0b01111101, 0b01000000, 0b01000000}, // 'i'
{4,0b00000100, 0b00000100, 0b00000101, 0b11111100}, // 'j'
{6,0b01000001, 0b01111111, 0b00010000, 0b01010100, 0b01101100, 0b01000100}, // 'k'
{5,0b01000000, 0b01000001, 0b01111111, 0b01000000, 0b01000000}, // 'l'
{7,0b01000100, 0b01111100, 0b01000100, 0b01111000, 0b01000100, 0b01111000, 0b01000000}, // 'm'
{7,0b01000100, 0b01111100, 0b01001000, 0b00000100, 0b01000100, 0b01111000, 0b01000000}, // 'n'
{5,0b00111000, 0b01000100, 0b01000100, 0b01000100, 0b00111000}, // 'o'
{6,0b00000100, 0b11111100, 0b01001000, 0b01000100, 0b01000100, 0b00111000}, // 'p'
{6,0b00111000, 0b01000100, 0b01000100, 0b01000100, 0b11111100, 0b00000100}, // 'q'
{5,0b01000100, 0b01111100, 0b01001000, 0b01000100, 0b00000100}, // 'r'
{5,0b01001000, 0b01010100, 0b01010100, 0b01010100, 0b00100100}, // 's'
{6,0b00000100, 0b00111110, 0b01000100, 0b01000100, 0b01000100, 0b00100000}, // 't'
{7,0b00000100, 0b00111100, 0b01000000, 0b01000000, 0b00100100, 0b01111100, 0b01000000}, // 'u'
{7,0b00000100, 0b00011100, 0b01100100, 0b01000000, 0b00110100, 0b00001100, 0b00000100}, // 'v'
{7,0b00000100, 0b00111100, 0b01000100, 0b00110000, 0b01000100, 0b00111100, 0b00000100}, // 'w'
{5,0b01000100, 0b01101100, 0b00010000, 0b01101100, 0b01000100}, // 'x'
{7,0b00000100, 0b00001100, 0b00110100, 0b11000000, 0b00110100, 0b00001100, 0b00000100}, // 'y'
{5,0b01001100, 0b01100100, 0b01010100, 0b01001100, 0b01100100}, // 'z'
{3,0b00010000, 0b11111110, 0b00000001}, // '{'
{1,0b11111111}, // '|'
{3,0b00000001, 0b11111110, 0b00010000}, // '}'
{5,0b00010000, 0b00001000, 0b00011000, 0b00010000, 0b00001000}, // '~'
};
#endif /* FONT_H */

```

## b) Fonction SPI

Étude de la fonction SPI du microcontrôleur MC9S08QE32 pour la mise en œuvre de l'afficheur :

Registre SPIC1 :



SPE [6] : A 1 active la fonction SPI

MSTR [4] : A 1 configure le microcontrôleur en maître

CPOL [3] : A 1 Configure l'état de repos de l'horloge sera à l'état haut

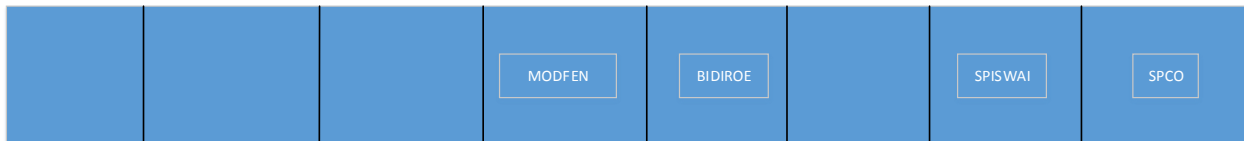
CPHA [2] : A 1 la synchronisation des données se fait sur le front du milieu

**SSOE [3]** : Bit est utilisé en combinaison avec le bit **MODFEN** du registre **SPIC2** pour définir le mode de fonctionnement du chip select (**SS/**), nous le mettrons à 1.

**LSBE [1]** : A 0 le transfert de données commencera par le **MSB**.

Les autres bits de ce registre seront laissés dans leur état par défaut.

Registre **SPIC2** :



**MODFEN [4]** : Pour configurer un fonctionnement automatique du chip select (**SS/**), nous mettrons ce bit à 1.

Les autres bits de ce registre seront laissés dans leur état par défaut.

Registre **SPIBR** :

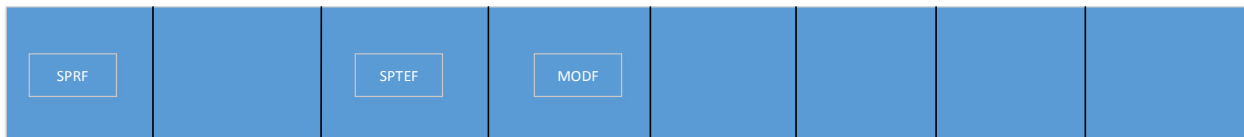


**SPPR [6 :4]** : Ces trois bits servent à une première pré division de la fréquence de bus du microcontrôleur pour l'horloge de la SPI. Ces bits seront à 0.

**SPR [3 :0]** : Ces quatre bits servent à diviser l'horloge de la SPI, par défaut l'horloge est divisé par deux.

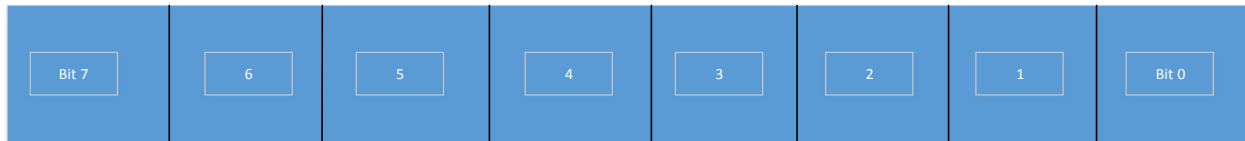
$$SPSCLK = \frac{25.165824MHz}{2} = 12.582912MHz$$

Registre **SPIS** :



**SPTEF** : Indicateur de fin de transmission, à 1 lorsque le buffer de transmission est plein, remise à zéro par lecture du registre **SPIS** lorsque **SPTEF** est à 1 suivi d'une écriture dans **SPID**.

Registre SPID :



SPID [7:0] : Registre d'émission et de transmission de la SPI.

### c) Fonction Timer

En plus de la fonction SPI du microcontrôleur à mettre en œuvre, il faudra mettre en œuvre des interruptions Timer pour l'affiche des batteries

Registre TPMxSC :



TOF [7] : Indicateur de débordement du compteur, à 1 lorsque le compteur dépasse sa capacité de comptage, remise à zéro par lecture de TPMxSC lorsque ce bit est à 1.

TOIE [6] : A 1 ce bit active une interruption lorsque le bit TOF est à 1.

CPWMS [5] : À 0 ce bit permet de choisir trois fonctions du timer (Output compare, input capture et PWM), il faut utiliser une combinaison avec 4 autres bits du registre TPMxCnSC.

CLKS [4:3] : A 01 ces deux bits sélectionne l'horloge de bus comme horloge du compteur.

PS [2:0] : A 111 divise l'horloge du compteur par 128.

Nous pouvons calculer alors le nombre de timerticks :

$$\text{Timerticks} = \frac{128}{25165824} = 5.08\mu\text{s}$$

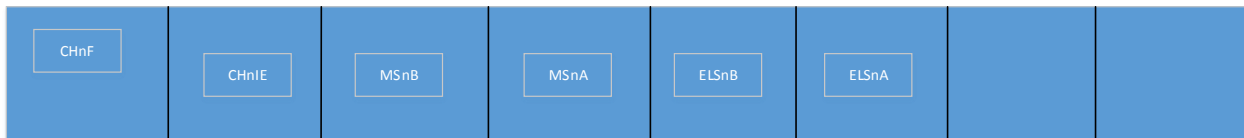
Registre TPMxMOD :



Double registre 8bits contenant le nombre de timerticks correspondant à de la demi période à compter, remise à zéro du registre lorsque que le bit TOF est remis à zéro.

$$TPMxMOD = \frac{100ms}{5.08us} = 19685$$

Registre TPMxCnSC :



MSnB : MSnA [5 :4] : A 01, configure le timer en Ouput compare.

ELSnB : ELSnA [3 :2] : A 00, configure l'Output compare en compteur logicielle.

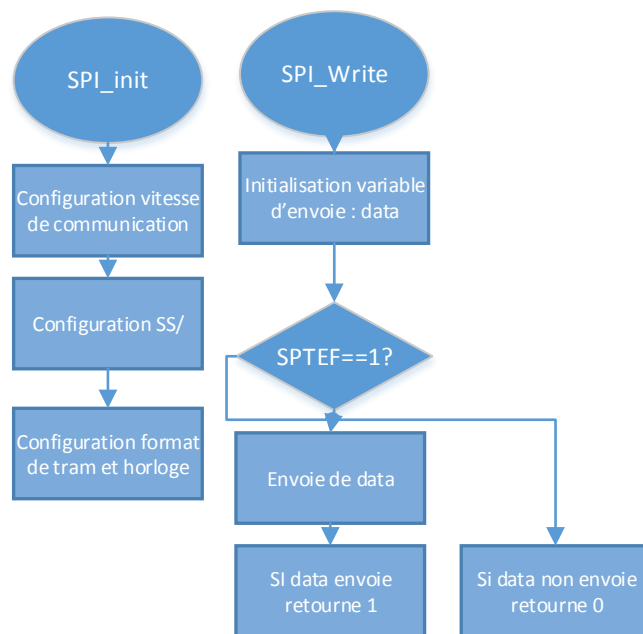
Les autres bits de ce registre seront laissés dans leur état par défaut.

### d) Mise en œuvre du programme d'affichage

Pour afficher un caractère, nous avons défini un tableau comprenant les caractères ' ' jusqu'au caractère '~' en hexadécimale les caractères font de 0x20 à 0x7E. Pour rechercher un caractère à afficher, nous allons donc faire une recherche dans le tableau en soustrayant la valeur du caractère à 0x20 pour trouver le caractère et ainsi que le nombre de pixels à utiliser pour l'afficher.

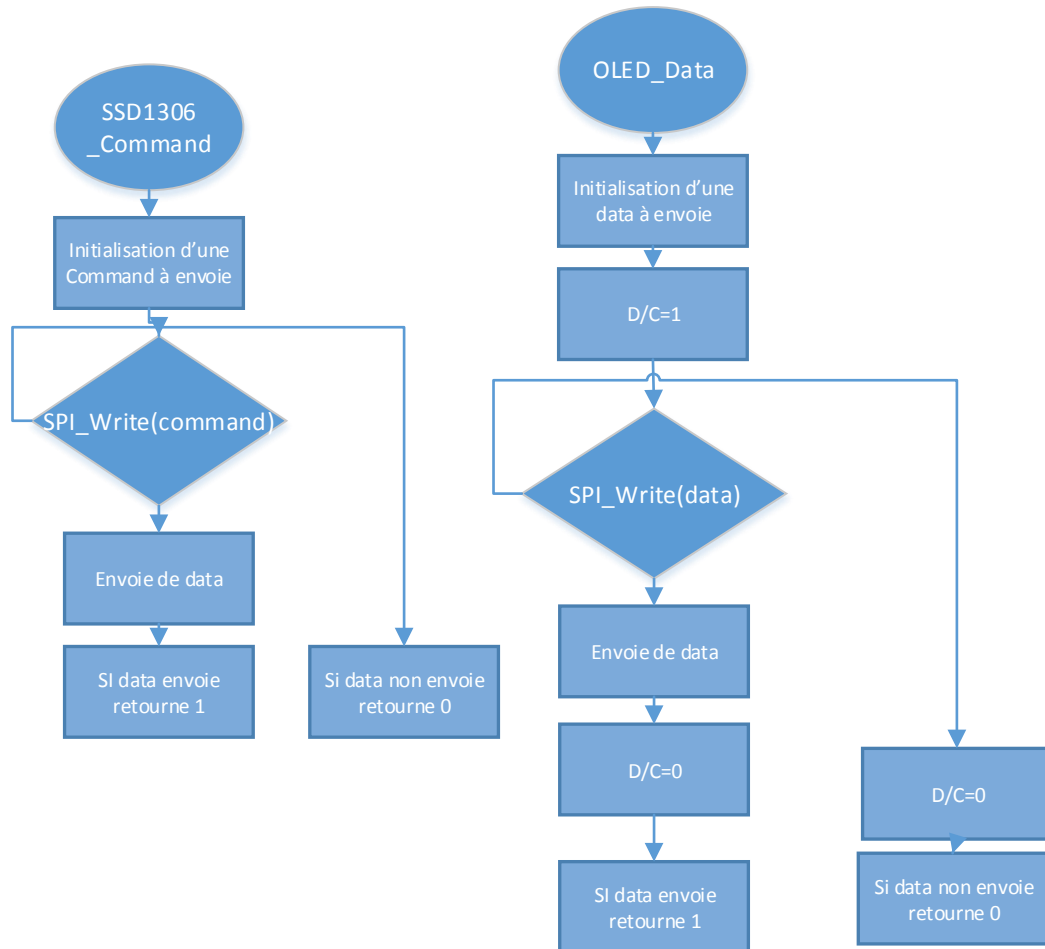
Dans le cas d'une image, il faut préalablement convertir en bitmap puis utiliser le logiciel LCD\_convert pour avoir l'expression de l'image sous forme de tableau, la taille du tableau dépendra du poids de l'image Dans le cas de notre afficheur, sa RAM ne peut contenir 1Ko, la taille du tableau de devra donc pas dépasser cette valeur.

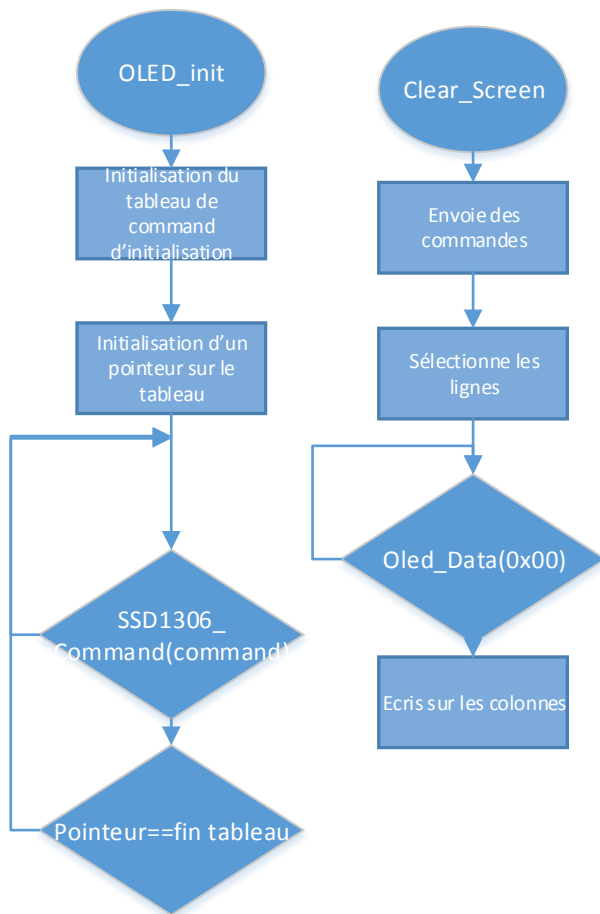
Algorithme SPI :

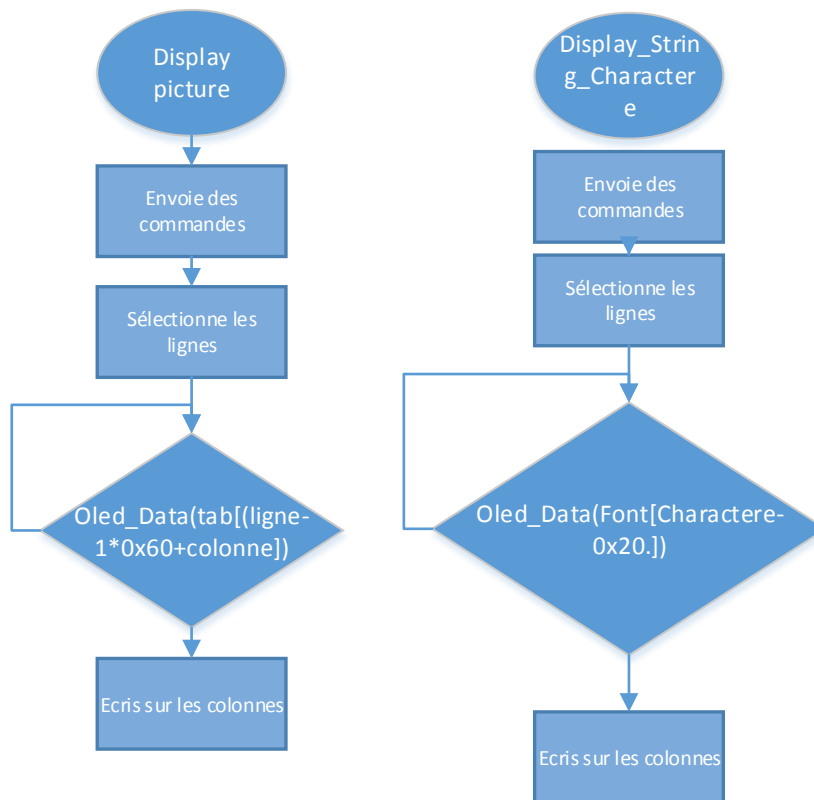


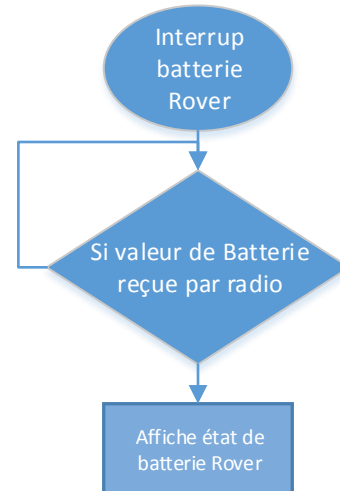
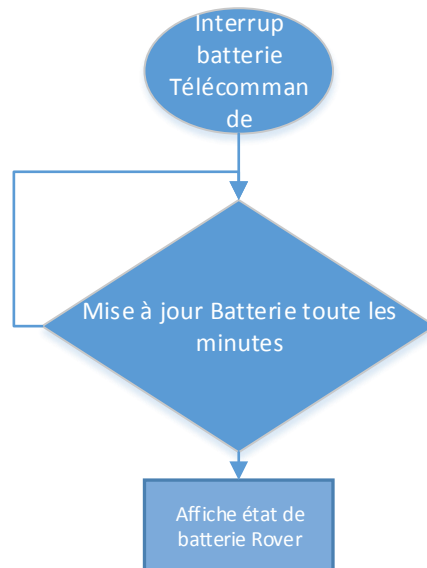
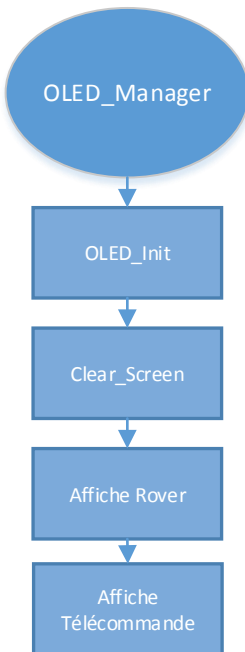


Algorithme SSD1306 :



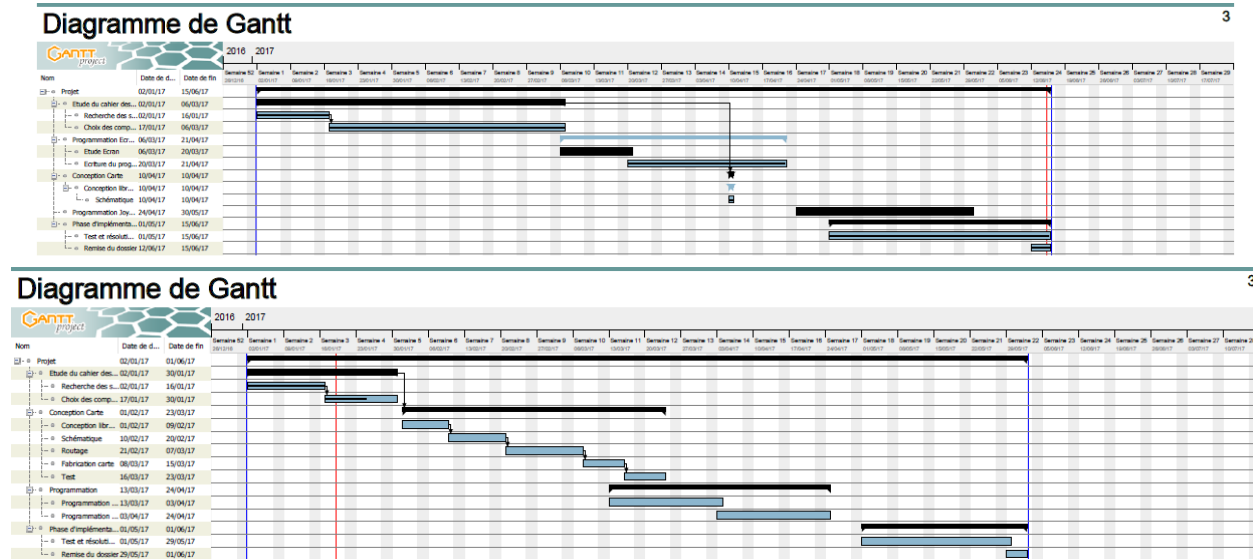






## Diagramme de Gantt

Gantt Réel ci-dessous



Gantt Prévisionnel ci-dessus

## Conclusion

---

*Ce premier projet m'a permis d'approfondir et d'utiliser les compétences que j'ai acquise en première année.*

*Le plus compliqué fut la mise en œuvre de l'afficheur, étant donné qu'il demande d'assez grosses ressources en C, j'ai reçu l'aide de notre responsable pédagogique Mr Sagonéro, qui a pris le temps de m'aider à la mise en œuvre. J'ai pu apprendre beaucoup grâce à cela, cette application m'a permis d'appréhender le C, langage de programmation appris juste en début d'année. Avec un peu plus de temps j'aurai bien aimé utiliser les autres commandes de l'écran car ils en possèdent bien plus que celle que j'ai utilisé pour ce projet.*

*La mise en œuvre du joystick que j'ai cru facile m'a posé plus de problèmes que prévus comme on peut le remarquer via à mon Gantt, je l'ai quelque peu négligé au détriment de la mise en œuvre de l'afficheur. Si je devais refaire de meilleurs choix pour ce projet je commencerais plutôt par la mise en œuvre de l'afficheur avec un autre mode d'adressage et pour le joystick on choisirait plutôt une vitesse variable pour bénéficier de sa fonction analogique. La façon dont je l'ai programmé ressemble beaucoup au mode de fonctionnement des joysticks d'une borne d'arcade.*

## Annexe

### 10. Logiciels utilisés



**CodeWarrior™**

**Altium  
Designer®**

### 11. Programmes en C commentés

SPI.c :

```
void SPI_init(void) {
    SPIBR=0x00; //vitesse de communication : fréquence de bus du µC/2
    SPIC2=0x00; // Chip select (SS/) mode automatique
    SPIC1=0x5E; // CPOL=1, CPHA=1,MSB first,µC en maitre
    PTBDD_PTBD0=1; // Mise en sortie de la broche 0 du port B * Broche DC
    PTBPE_PTPE0=1; // Active résistance pull up sur DC
    PTADD_PTADD7=1; // Mise en sortie de la broche 1 du port A * Broche RES
    PTAPE_PTAPE7=1; // Active résistance pull up sur RES
    return;
}

char SPI_Write(char data) {
    if(SPI_SPTEF) { // Si la donnée précédente a été envoyée
        SPID=data; // Envoi de la nouvelle donnée
        asm{
            NOP
        };
        return 1; // Renvoi 1
    }
    return 0; // Si la précédente donnée n'a pas été envoyée , je
renvoie 0
}
```

SSD1306.c :

```
#include "SSD1306.h"
#include "estei.h"
#include "font.h"

static char SSD1306_Command(char );
const unsigned char *Rover="Rover";
const unsigned char *Val_batterie_Rover="100%";
const unsigned char *Val_batterie_Telecommande="100%";
const unsigned char *Telecommande="Telecommande";

union{
    unsigned char OLED_State_Machine_Byte;
    struct {
        unsigned char init          :1;
        unsigned char clear         :1;
        unsigned char text_rov      :1;
        unsigned char text_tel      :1;
        unsigned char bat_rov       :1;
        unsigned char bat_tel       :1;
        unsigned char suite         :2;
    }OLED_State_machine_bit;
}OLED_State_machine;

/*Commande d'affichage pour l'écran*/
const unsigned char print[]={0xB0,0x00,0x10};
const unsigned char print_bat[]={0xB0,0x07,0x1D};
const unsigned char print_tel[]={0xB2,0x00,0x10};
const unsigned char print_bat_tel[]={0xB2,0x07,0x1D};
```

```
/*Tableau d'initialisation du SSD1306*/
```



OFFOUGA Joris

SER2

Projet Véhicule Interactif





```

const unsigned char init_SSD1306[]={
    0xA8,0x27, //Set MUX Ratio A8h, 3Fh
    0xD3,0x00, //Set Display Offset D3h, 00h
    0x40,      //Set Display Start Line 40h
    0xA0,      //Set Segment re-map A0h/A1h
    0xC0,      //Set COM Output Scan Direction C0h/C8h
    0x81,0xff, //Set Contrast Control 81h, 7Fh
    0x8D,0x14, //Enable charge pump regulator 8Dh, 14h
    0xA4,      //Disable Entire Display On A4h
    0xA6,      //Set Normal Display A6h
    0xAF       //Display On AFh
};

/*Gestionnaire de tache de l'afficheur*/
void OLED_Manager(void) {
    switch(OLED_State_machine.OLED_State_Machine_Byte) {
        case 0:
            OLED_init();
            break;
        case 1:
            Clear_Screen();
            break;
        case 3:
            Display_String_Character(&Rover);
            //Display_Picture(estei);
            break;
        case 7:
            Display_val_bat(&Val_batterie_Rover);
            break;
        case 15:
            Display_String_Character_(&Telecommande);
            break;
        case 31:
            Display_val_bat_(&Val_batterie_Telecommande);
            break;
        default:
            break;
    }
    return;
}

```

/\* Fonction permettant l'envoi de commande unique vers l'écran \*/



OFFOUGA Joris

SER2

Projet Véhicule Interactif



```
static char SSD1306_Command(char command) {
    if(SPI_Write(command)) {
        return 1;
    }
    return 0;
}

/* Fonction permettant l'envoi de data vers l'écran */
char OLED_Data(char data) {
    DC_1;
    if(SPI_Write(data)) {
        DC_0;
        return 1;
    }
    DC_0;
    return 0;
}

/* Fonction d'initialisation de l'écran*/
void OLED_init(void) {
    static const unsigned char *init_data=&init_SSD1306[0];
    if(SSD1306_Command(*init_data)){
        if(++init_data==(&init_SSD1306[0]+sizeof(init_SSD1306))){
            OLED_State_machine.OLED_State_machine_bit.init=1;
        }
    }
    return;
}
```

```
/*Fonctionn permettant l'affichage d'une image sur l'écran */
void Display_Picture(const unsigned char tab[]){
    static unsigned char i=0,j=0;
    static unsigned char *p_cmd=&print_[0];
    static unsigned char sequence=0;
    switch(sequence){
    case 0: // ligne i
        if(SSD1306_Command(*p_cmd|i)) {
            if(++i>5){
                i=0;
                j=0;
                sequence=0;
                p_cmd=&print_[0];
            }
            p_cmd++;
            sequence=1;
        }
        break;
    case 1:
        if(SSD1306_Command(*p_cmd)) {
            if(++p_cmd==(print_[0]+sizeof(print_))) {
                p_cmd=&print_[0];
                j=0;
                sequence=2;
            }
        }
        break;
    case 2: // colonne j
        if(OLED_Data(tab[(i-1)*0x60+j])) {
            if(++j>95){
                sequence=0;
            }
        }
        break;
    default :
        break;
    }
}
```

```
/* Fonction permettant le nettoyage de l'écran */
void Clear_Screen(void) {
    static unsigned char i=0,j=0;
    static unsigned char *p_cmd=&print_[0];
    static unsigned char sequence=0;
    switch(sequence) {
    case 0: // ligne i
        if(SSD1306_Command(*p_cmd|i)) {
            if(++i==6) {
                i=0;
                OLED_State_machine.OLED_State_machine_bit.clear=1;
            }
            p_cmd++;
            sequence=1;
        }
        break;
    case 1:
        if(SSD1306_Command(*p_cmd)) {
            if(++p_cmd==(print_[0]+sizeof(print_))) {
                p_cmd=&print_[0];
                j=0;
                sequence=2;
            }
        }
        break;
    case 2: // colonne j
        if(OLED_Data(0x00)) {
            if(++j==128) {
                sequence=0;
            }
        }
        break;
    default :
        break;
    }
}
```

```

void Display_String_Character (const unsigned char **p_caractere){
    static unsigned char i=0,j=0;
    static unsigned char index_caractere=0;
    static unsigned char sequence=0;
    static unsigned char *p_cmd=&print_[0];
    switch (sequence) {
    case 0:
        if(SSD1306_Command(*p_cmd|i)){
            if(++i==6){
                i=0;
                j=0;
                sequence=0;
                p_cmd=&print_[0];
            }
            p_cmd++;
            sequence=1;
        }
        break;
    case 1:
        if(SSD1306_Command(*p_cmd)) {
            if(++p_cmd==(print_[0]+sizeof(print_))){
                p_cmd=&print_[0];
                j=0;
                sequence=3;
            }
        }
        break;
    case 3: // colonne j
        if(OLED_Data((unsigned char)Courrier[((*p_caractere)-
0x20)].tab[index_caractere])){
            if(++index_caractere==Courrier[((*p_caractere)-
0x20)].wide){
                if(OLED_Data(0x00)){
                    if(*(++(*p_caractere))=='\0'){
                        OLED_State_machine.OLED_State_machine_bit.text_rov=1;
                    }
                    index_caractere=0;
                }
            }
        }
        if(++j>95){
            sequence=0;
        }
        break;
    default :
        break;}}
void Display_val_bat(const unsigned char **val_bat){

```



OFFOUGA Joris

SER2

Projet Véhicule Interactif



```

static unsigned char i=0,j=0;
static unsigned char index_caractere=0;
static unsigned char sequence=0;
static unsigned char *p_cmd=&print_bat[0];
switch (sequence) {
case 0:
    if(SSD1306_Command(*p_cmd|i)){
        if(++i==6) {
            i=0;
            j=0;
            sequence=0;
            p_cmd=&print_bat[0];
        }
        p_cmd++;
        sequence=1;
    }
    break;
case 1:
    if(SSD1306_Command(*p_cmd)) {
        if(++p_cmd==( &print_bat[0]+sizeof(print_bat))){
            p_cmd=&print_bat[0];
            j=0;
            sequence=3;
        }
    }
    break;
case 3: // colonne j
    if(OLED_Data((unsigned char)Courrier[ ((**val_bat)-
0x20)].tab[index_caractere])){
        if(++index_caractere==Courrier[ ((**val_bat)-0x20)].wide) {
            if(OLED_Data(0x00)) {
                if(*(++(*val_bat))=='\0') {

OLED_State_machine.OLED_State_machine_bit.bat_rov=1;
                }
                index_caractere=0;
            }
        }
    }
    if(++j>95) {
        sequence=0;
    }
    break;
default :
    break;
}

void Display_String_Character_ (const unsigned char **p_caractere) {

```



OFFOUGA Joris

SER2

Projet Véhicule Interactif



```

static unsigned char i=0,j=0;
static unsigned char index_caractere=0;
static unsigned char sequence=0;
static unsigned char *p_cmd=&print_tel[0];
switch (sequence) {
case 0:
    if(SSD1306_Command(*p_cmd|i)){
        if(++i==6){
            i=0;
            j=0;
            sequence=0;
            p_cmd=&print_tel[0];
        }
        p_cmd++;
        sequence=1;
    }
    break;
case 1:
    if(SSD1306_Command(*p_cmd)) {
        if(++p_cmd==(&print_tel[0]+sizeof(print_tel))){
            p_cmd=&print_tel[0];
            j=0;
            sequence=3;
        }
    }
    break;
case 3: // colonne j
    if(OLED_Data((unsigned char)Courrier[((*p_caractere)-
0x20)].tab[index_caractere])){
        if(++index_caractere==Courrier[((*p_caractere)-
0x20)].wide){
            if(OLED_Data(0x00)){
                if(*(++(*p_caractere))=='\0'){
                    OLED_State_machine.OLED_State_machine_bit.text_tel=1;
                }
                index_caractere=0;
            }
        }
    }
    if(++j>95){
        sequence=0;
    }
    break;
default :
    break;
} }

```

```

void Display_val_bat_(const unsigned char **val_bat){
    static unsigned char i=0,j=0;
    static unsigned char index_caractere=0;
    static unsigned char sequence=0;
    static unsigned char *p_cmd=&print_bat_tel[0];
    switch (sequence) {
    case 0:
        if(SSD1306_Command(*p_cmd|i)){
            if(++i==6){
                i=0;
                j=0;
                sequence=0;
                p_cmd=&print_bat_tel[0];
            }
            p_cmd++;
            sequence=1;
        }
        break;
    case 1:
        if(SSD1306_Command(*p_cmd)) {
            if(++p_cmd==(&print_bat_tel[0]+sizeof(print_bat_tel))){
                p_cmd=&print_bat_tel[0];
                j=0;
                sequence=3;
            }
        }
        break;
    case 3: // colonne j
        if(OLED_Data((unsigned char)Courrier[((**val_bat)-
0x20)].tab[index_caractere])){
            if(++index_caractere==Courrier[((**val_bat)-0x20)].wide){
                if(OLED_Data(0x00)){
                    if(*(++(**val_bat))=='\0'){
                        OLED_State_machine.OLED_State_machine_bit.suite=1;
                    }
                    index_caractere=0;
                }
            }
        }
        if(++j>95){
            sequence=0;
        }
        break;
    default :
        break;
    }
}

```



## Command\_Joystick.c

```

#define W1 PTC1_PTC1 // Rotation gauche
#define W2 PTC2_PTC2 // Rotation droite
#define M PTA3_PTA3
#define Haut 150
#define Bas 100
#define joyX 0x08
#define joyY 0x05

/*Tableau de commande Rover*/
const unsigned char motion_control[]={
    0x00, // Pas de déplacement
    0x80, // Translation horizontale vers la droite
    0x90, // Translation diagonale vers la droite (+45°)
    0xA0, // Translation verticale vers le haut
    0xB0, // Translation diagonale vers la gauche (+45°)
    0xC0, // Translation horizontale vers la gauche
    0xD0, // Translation diagonale vers la gauche (-45°)
    0xE0, // Translation verticale vers le bas
    0xF0, // Translation diagonale vers la droite (-45°)
    0x04, // Rotation gauche
    0x08, // Rotation droite
};

void ADC_init(void) {
    ADCCFG_ADICLK=0x00; // Fréquence d'échantillonnage = Fbus
    ADCCFG_ADIV=0x06; // Fréquence d'échantillonnage = Fbus/8
    ADCSC1_ADCO=0x01; // Activation ADC
    return;
}

/*Partie ADC*/
char ADC(char Channel) {
    ADCSC1_ADCH=Channel;
    while(!ADCSC1_COCO);
    return ADCRL;
}

```

```
/*Command Joystick au Rover */
```



OFFOUGA Joris

SER2

Projet Véhicule Interactif



```

void Command(void) {
    static unsigned char sequence=0;
    unsigned char X,Y;
    X=ADC(joyX);
    Y=ADC(joyY);
    if (W1==0) {
        SCI_Send(motion_control[9]);
        return;
    }
    if (W2==0) {
        SCI_Send(motion_control[10]);
        return;
    }
    if(X>=Bas && X<=Haut && Y>=Bas && Y<=Haut){
        SCI_Send(motion_control[0]); // Pas de déplacement
    }
    else if(X>Haut && Y>Bas && Y<Haut){
        SCI_Send(motion_control[1]); // Translation horizontale vers la
droite
    }
    else if(X>Haut && Y>Haut){
        SCI_Send(motion_control[2]); // Translation diagonale vers la droite
(+45°)
    }
    else if(Y>Haut && X>Bas && X<Haut){
        SCI_Send(motion_control[3]); // Translation verticale vers le haut
    }
    else if(Y>Haut && X<Bas){
        SCI_Send(motion_control[4]); // Translation diagonale vers la gauche
(+45°)
    }
    else if(X<Bas && Y>Bas && Y<Haut){
        SCI_Send(motion_control[5]); // // Translation horizontale vers la
gauche
    }
    else if(X<Bas && Y<Bas){
        SCI_Send(motion_control[6]); // Translation diagonale vers la
gauche (-45°)
    }
    else if(X>Bas && X<Haut && Y<Bas){
        SCI_Send(motion_control[7]); // Translation verticale vers le bas
    }
    else if(X>Haut && Y<Bas){
        SCI_Send(motion_control[8]); // Translation diagonale vers la droite
(-45°)
    }
}

```



*OFFOUGA Joris*

*SER2*

*Projet Véhicule Interactif*

