|  |  |
| --- | --- |
| Référence du document  Document d'architecture general.docx | Type de document : |
| Qualification  V1 | Auteur(s) : A.SANNIER  A.PERRIN |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modifications apportées à ce document : | | | | |
| **Version** | **Date** | **Auteur** | **Action** | **Modifications** |
| V1 | 10 avril 2020 | A.SANNIER  A.PERRIN | Création |  |



Table des matières :

[1. INTRODUCTION : 3](#_Toc37432346)

[1.1 Contexte : 3](#_Toc37432347)

[1.2 Présentation de l’équipe : 3](#_Toc37432348)

[2. PRESENTATION DU PROJET : 3](#_Toc37432349)

[2.1 Expression du besoin : 3](#_Toc37432350)

[2.2 Exigences logiciels 3](#_Toc37432351)

[2.3 Solutions techniques 4](#_Toc37432352)

[3. Architecture Logiciel 5](#_Toc37432353)

[4. Architecture Logiciel détaillée 6](#_Toc37432354)

[5. Test unitaire : 8](#_Toc37432355)

# INTRODUCTION :

## Contexte :

Nous avons dû réaliser ce projet dans un cadre pédagogique. Dans la dominante ISE OC que nous avons choisi, nous avons consacré les heures de bus de communication ainsi que les heures de MOQL pour le réaliser.

N’ayant pas pu nous rendre dans les salles TP prévues à cet effet pour cause de fermeture de l’école, nous avons dû réaliser ce projet en Visio conférence, chacun chez nous, avec le matériel à notre disposition.

## Présentation de l’équipe :

Alexandre SANNIER : Employé par Transdev ROUEN en tant que Agent de maitrise dans l’informatique industriel et l’électronique

Arthur PERRIN : Apprenti à Matra Electronique et médaille d’or au JO en roulade de rapidité.

Nous sommes actuellement tous les deux en deuxième année du cycle ingénieur à l’esigelec, en alternance.

# PRESENTATION DU PROJET :

## Expression du besoin :

Nous devons pouvoir contrôler un robot à distance afin que ce dernier puisse se déplacer un espace délimité. Ce dernier doit pouvoir détecter les obstacles, à l’aide d’un capteur pouvant pivoter, les signaler et s’arrêter au besoin.

## Exigences logiciels

* **SYS\_0010**:

Nom : CommandeRobot

Description : Le SAMBOT doit répondre aux commandes envoyées par l’utilisateur.

* **SYS\_0020 :**

Nom : DetectionObstacle

Description : Le SAMBOT doit pouvoir détecter un obstacle et s’arrêter lors de cette détection. Il doit également envoyé une notification à l’utilisateur de cette détection.

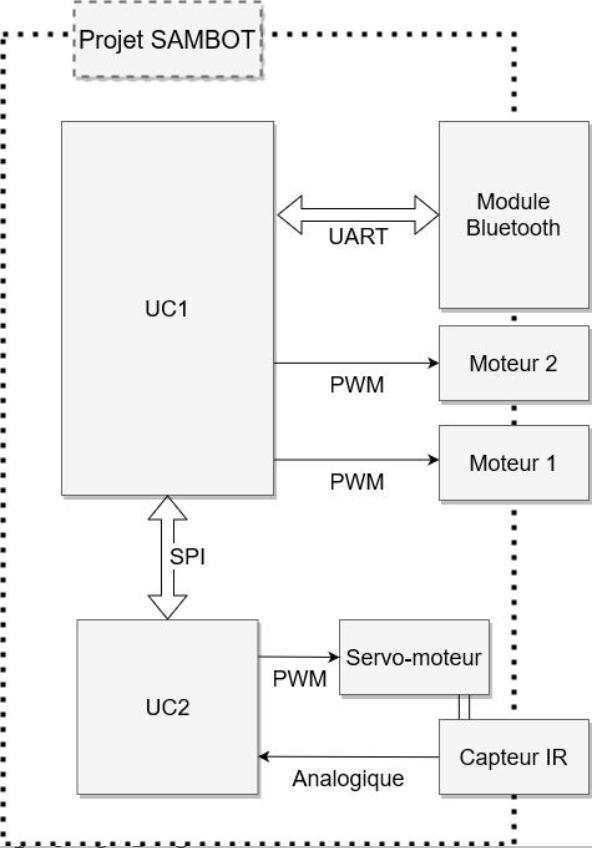
Le capteur de distance doit pouvoir balayer son champ de vision à 180°

* **SYS\_0030 :**

Nom : InterfaceBluetooth

Description : Les commandes doivent pouvoir être envoyées via une interface Bluetooth ainsi que les notifications d’obstacles.

## Solutions techniques



|  |  |  |
| --- | --- | --- |
|  | Arthur PERRIN | Alexandre SANNIER |
| UC1 | Atemga328p (atmel studio + avr-gcc) | MSP2553 |
| UC2 | Attiny85 (atmel studio + avr-gcc) | MSP2231 |
| Servo-moteur | SG-90 9g | SM-S2309S |
| Capteur IR | Similaire ESIGELEC | POTENTIOMETRE |
| 2 Moteurs | 4 LEDs (2PWMs+ 2 Sens) | OSCILOSCOPE |
| Module Bluetooth | HM-10 BLE contrefait | RN42 |

Dans un souci de clarté et d’optimisation du travail, nous n’étudierons que le programme d’Alexandre SANNIER sur les MSP430. Le code d’Arthur PERRIN est cependant disponible et commenté sur github.

# Architecture Logiciel

Nous allons ici lister exigences architectural logiciels. Elles seront en accord avec les exigences logicielles citées plus haut.

Elles seront nommées HLR\_XXXX

* HLR\_0100

Nom : MouvementRobot

Description : Lorsqu’une commande est passée le robot doit réagir en fonction de cette commande.

Couvre : SYS\_0010

Module : board (MSP2553)

* HLR\_0200

Nom : InterfaceUtilisateur

Description : Permet de passer une commande au robot via une interface utilisateur distante.

Couvre : SYS\_0020

Module : UART, main (MSP 2553)

* HLR\_0300

Nom : MouvementCapteur

Description : Permet de commander la direction du capteur d’obstacle.

Couvre : SYS\_0010

Module : board ( MSP2231)

* HLR\_0400

Nom : DetectionObstacle

Description : Permet de détecter un obstacle situé à une distance donnée du capteur et arreter les moteurs en cas d’obstacle

Couvre : SYS\_0020

Module : main ( MSP2553)

* HLR\_0500

Nom : EnvoiNotification

Description : Envoi des notifications à l’utilisateur (distance, obstacle)

Couvre : SYS\_0020, SYS\_0030

Module : main  (MSP2553)

* HLR\_0600

Nom : CommunicationUtilisateur

Description : Toutes les communications avec l’utilisateur doivent se faire par Bluetooth

Couvre : SYS\_0030

Module : UART , main (MSP2553)

# Architecture Logiciel détaillée

Nous allons ici lister exigences architectural détaillées logiciels. Elles seront en accord avec les exigences architectural citées plus haut.

Elles seront nommées DDR\_XXXX

* DDR\_0100

Nom : Mouvement Robot

Description : Le robot doit bouger dans la direction passée en paramètre. Si aucune direction il s’arrete

Couvre : HLR\_0100

Fonction : void mvtRobot(int PuissanceA, int PuissanceB)

Test : TU\_0100

* DDR\_0200

Nom : Sens Robot

Description : Définition du sens du robot

Couvre : HLR\_0100

Fonction : void robotSens(int sens)

* DDR\_0300

Nom : Stop Robot

Description : Le robot doit s’arrêter sur commande

Couvre : HLR\_0100

Fonction : void robotStop()

Test : TU\_0200

* DDR\_0400

Nom : Init Timer PWM

Description : Permet de configurer les timer pour les PWM (2231)

Couvre : HLR\_0100

Fonction : void init\_TimerMot()

* DDR\_0500

Nom : envoi un char

Description : Permet d’envoyer un char via L UART du msp 2553

Couvre : HLR\_0200

Fonction : void send\_uart(char)

Test : TU\_0300

* DDR\_0600

Nom : Init UART

Description : Permet de configurer le MSP 2553 pour communication UART

Couvre : HLR\_0200

Fonction : void init\_uart()

Test : TU\_0400

* DDR\_0700

Nom : envoi une chaine de char

Description : Permet d’envoyer une chaine de char via L UART du msp 2553

Couvre : HLR\_0200 , HLR\_0600

Fonction : void send\_uart\_l(const \*char)

* DDR\_0800

Nom : envoi sdl

Description : Permet d’envoyer un saut de ligne via L UART du msp 2553

Couvre : HLR\_0200 , HLR\_0400, HLR\_0600

Fonction : void send\_uart\_sdl()

* DDR\_0900

Nom : init SPI

Description : Permet d’initialiser la communication SPI (MSP2553)

Couvre : HLR\_0300, HLR\_0400

Fonction : void init\_SPI()

Test : TU\_0500

* DDR\_1000

Nom : init usi

Description : Permet d’initialiser la communication SPI (MSP2231)

Couvre : HLR\_0300, HLR\_0400

Fonction : void init\_usi()

Test : TU\_0600

* DDR\_1100

Nom : send SPI

Description : Permet d’envoyer une commande au MSP 2231 via le MSP2553(MSP2553)

Couvre : HLR\_0300, HLR\_0400

Fonction : void send\_SPI(char)

Test : TU\_0300

* DDR\_1200

Nom : init timer communication spi

Description : Permet de garder une communication active avec le MSP2231

Couvre : HLR\_0300, HLR\_0400

Fonction : void init\_time\_com\_SPI()

Test : TU\_0700

* DDR\_1300

Nom : envoi une notification

Description : envoi des notifications sur l’état du robot(2231 > 2553 : distance par rapport à l’obstacle, le MSP2553 définit si nécessité d’arrêter robot)

Couvre : HLR\_0500, HLR\_0600

Fonction : \_\_interrupt void USCIAB0RX\_ISR() (MSP2553)

* DDR\_1500

Nom : detecter distance obstacle

Description : renvoie la distance de robot à obstacle (MSP2231)

Couvre : HLR\_0500

Fonction : INT\_32 int detectionObstacle(void)

* DDR\_1600

Nom : command capteur

Description : Permet d’orienter le capteur

Couvre : HLR\_0300

Fonction : void commandServo(void)

* DDR\_1700

Nom : init board

Description : Permet d’initialiser la board pour les capteurs du MSP2231

Couvre : HLR\_0300 , HLR\_0400

Fonction : void init\_board(void)

# Test unitaire :

* TU\_0100

Nom : test mouvement robot

Description : Permet de vérifier que le robot réagit correctement aux commandes envoyées

Détails : soit les 4 commandes. Suivante associées à l’oscillogramme de chaque moteur.

Sur la ligne supérieure le moteur A Située à droite

Sur la ligne inférieure le moteur B située à gauche du robot

|  |  |  |  |
| --- | --- | --- | --- |
| Avancer | Tourner à Gauche | Tourner à droite | Reculer |
|  |  |  |  |

**Resultat : Les signaux envoyés sont conformes aux commandes.**

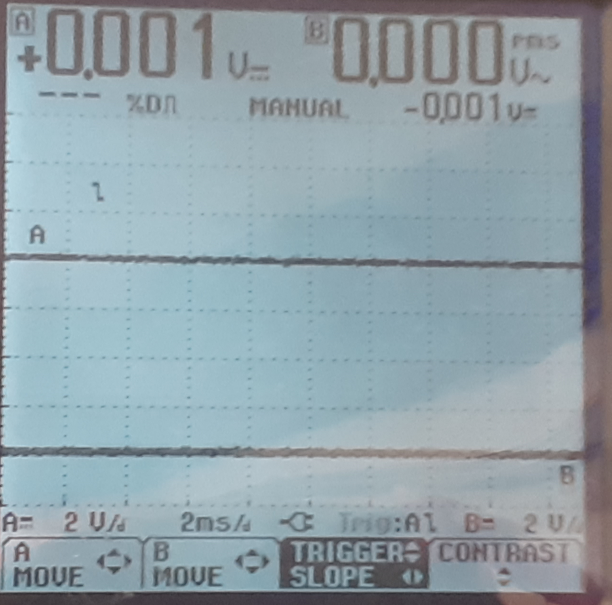
* TU\_0200

Nom : test robot stop

Description : Permet de vérifier que le robot stop ses commandes moteur lors d’une demande utilisateur..

Sur la ligne supérieure le moteur A Située à droite

Sur la ligne inférieure le moteur B située à gauche du robot



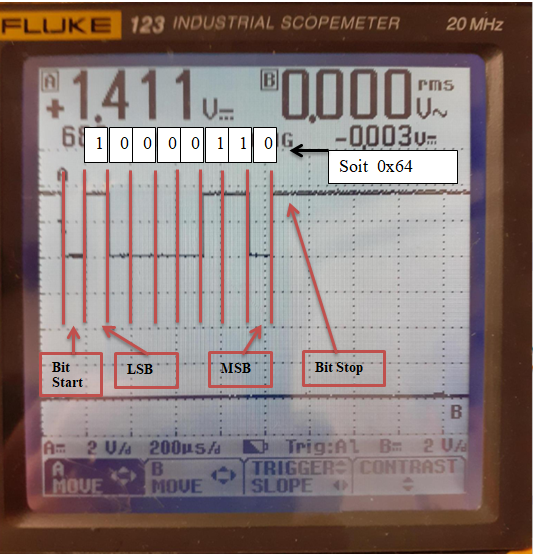
**Résultat : Le signal d’arrêt répond bien à la commande.**

* TU\_0300

Nom : test com UART

Description : Permet de vérifier que les commandes UART envoyées via le terminal sont conformes l’aide d’un oscillogramme

Détails : Soit le char ‘a’ (0x64) envoyé depuis le terminal.



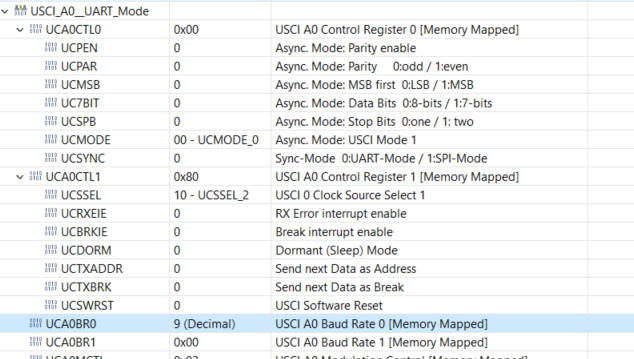
**Résultat : Le char envoyé par le terminal est bien physiquement présent sur la broche RX du MSP2553**

* TU\_0400

Nom : test initialisation de la communication des registres UART

Description : Permet de vérifier que l’état des registres de l’UART (MSP2553)

Détails : On lance le MSP et on active le debugger pour vérifier que l’état du registre correspond bien aux paramètres . ( 8bits , 1 bit de start , 1 bit de stop , Aucune parité , LSB First , 115000Baud)



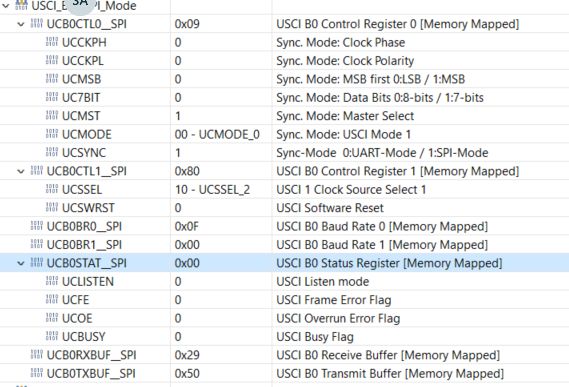
**Résultat : Les registres sont correctement configurés.**

* TU\_0500

Nom : test initialisation de la communication des registres SPI

Description : Permet de vérifier que l’état des registres est conforme pour l USI (MSP 2553)

Détails : On lance le MSP et on active le debugger pour vérifier que l’état du registre correspond bien aux paramètres. (Master Mode, Mode 0, Mode synchrone, Phase 0, Polarité 0 , LSB First , 8 bits )



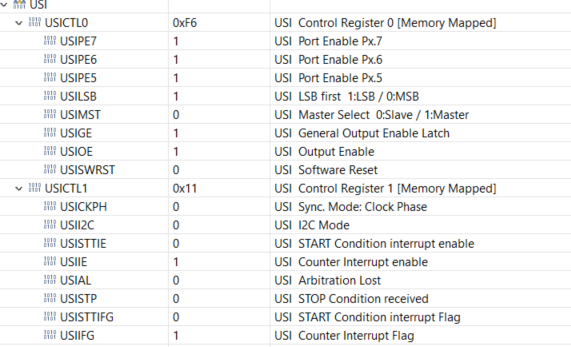
**Résultat : Les registres sont correctement configurés.**

* TU\_0600

Nom : test initialisation de la communication des registres USI

Description : Permet de vérifier que l’état des registres est conforme pour l USI (MSP 22231)

Détails : On lance le MSP et on active le debugger pour vérifier que l’état du registre correspond bien aux paramètres. (Slave Mode, LSB First , 8 bits, Clock externe )



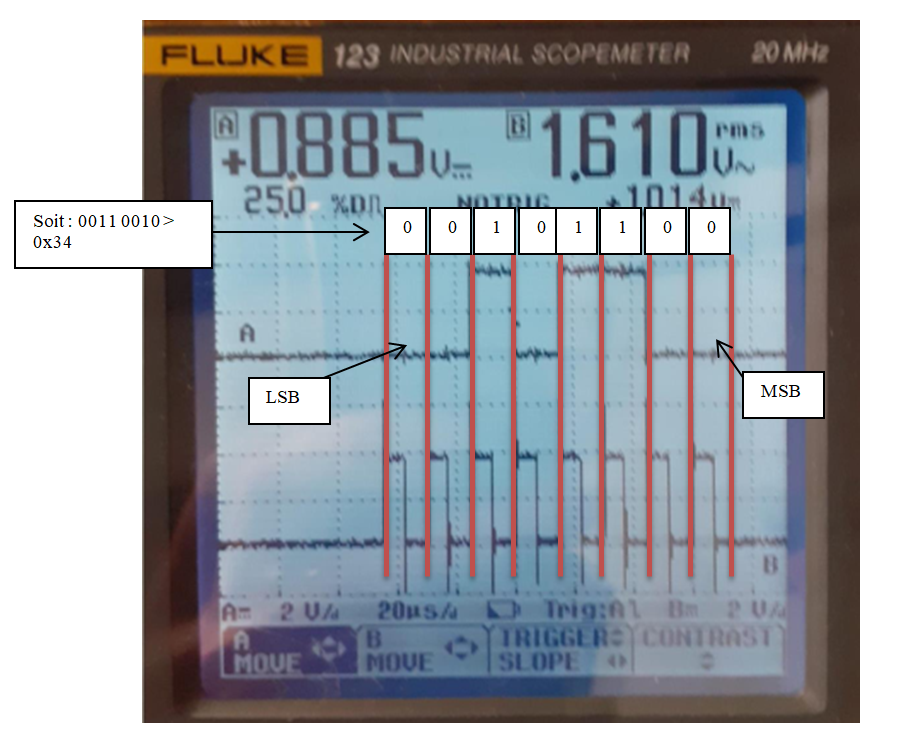
**Résultat : Les registres sont correctement configurés.**

* TU\_0700

Nom : test com SPI

Description : Permet de vérifier que les commandes SPI sont conformes avec le code, à l’aide d’un oscillogramme

Détails : Soit le char 0x34 envoyé sur la broche MOSI (P1.7) du MSP.



**Résultat : Le char envoyé par le code est bien physiquement présent sur les lignes de communication des microP**