Capucine BOURGADE ISE-OC

Sarah GUILLOTEL

Nolwenn OLLIVIER

**Rapport de Projet : la Serre Connectée**

***Bus de Communication / Méthodologie et Outils de Gestion Qualité Logicielle***

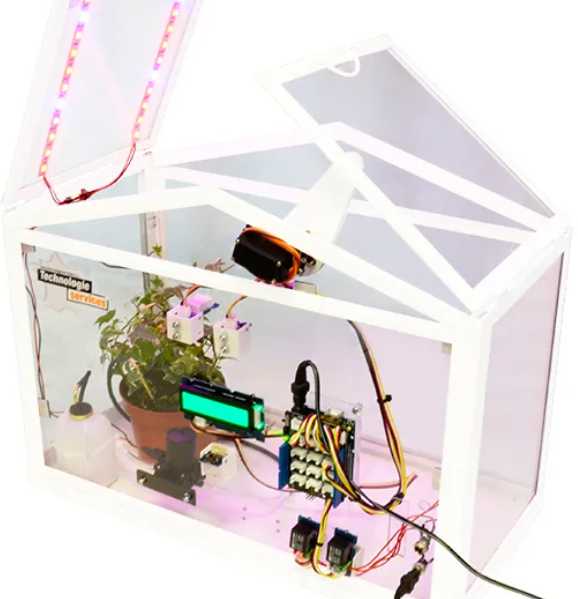




Table des matières

[I) Cahier des charges : 3](#_Toc68104743)

[II) Exigences : 3](#_Toc68104744)

[III) Schéma de principe : 3](#_Toc68104745)

[IV) Architecture générale (multi couches – électronique/mécanique/software) : 4](#_Toc68104746)

[V) Architecture détaillée (fonctions prototypes – software) : 4](#_Toc68104747)

[VI) Tests Unitaires 6](#_Toc68104748)

[a) Fonctions concernant le servo-moteur : 6](#_Toc68104749)

[b) Fonction concernant le capteur UV : 8](#_Toc68104750)

[c) Fonction concernant le capteur hydrométrie et de température : 9](#_Toc68104751)

[d) Fonction concernant le module Bluetooth : 11](#_Toc68104752)

[VII) Test d’intégration 13](#_Toc68104753)

[VIII) LDRA 14](#_Toc68104754)

[a) Servomoteur 15](#_Toc68104755)

[b) Capteur UV 15](#_Toc68104756)

[c) Bluetooth 16](#_Toc68104757)

[d) Slave 17](#_Toc68104758)

[e) Master 18](#_Toc68104759)

[f) Lien GITHUB : 19](#_Toc68104760)

# Cahier des charges :

Le client souhaite modéliser une serre connectée à l’aide de deux cartes MSP430 (2553/2553) connectées entre elles via un bus SPI. Les commandes sont fournies au système selon une interface de commande Putty (via UART) ou à l’aide d’un système de commande Bluetooth (choix à définir avec le client).

Un servo-moteur est utilisé pour gérer l’ouverture de la serre via un signal PWM. La serre est également équipée d’un capteur hydrométrique (communication I²C), et d’un capteur UV (signal analogique transmis par UART). Le servo moteur se situe sur la carte « Maître », tandis que le capteur UV et hydrométrique se situent sur la carte « Esclave ».

Il est souhaité par le client qu’en fonction des valeurs reçues par le capteur hydrométrique, la servo-moteur réagisse en fonction (cf. II – Exigences). Pour des raisons que nous aborderons dans la partie XXX, il nous a été impossible de mettre en œuvre la communication I²C, par conséquent la simulation des valeurs de température et d’humidité sera faite via l’interface de commande par l’utilisateur.

# Exigences :

Les exigences du client sont les suivantes :

* Communication UART entre MSP430G2553 et l’invite de commandes
* Communication SPI entre les deux MSP4302553
* Communication I²C entre le Capteur HD et le MSP4302553 dit « Esclave »
* Seuils de température définis comme suit :
  + - Inférieur à 10°C : Trop bas, cela déclenche la fermeture de la porte de la serre
    - Supérieur à 30°C : Trop haut, cela déclenche l’ouverture de la porte de la serre
* Seuils d’humidité définis comme suit :
  + - Inférieur à 20% : Trop bas
    - Supérieur à 90% : Trop haut
* Pour les seuils de température et d’humidité, une LED s’allume quand le seuil est dépassé, et un message s’affiche sur l’invite de commande pour prévenir l’utilisateur.

# Schéma de principe :

SPI

MSP430G2553

MSP430G2553

Entrée (ligne de commandes ou Bluetooth)

Capteur HD

Capteur UV

Capteur Servo-moteur

Signal Analogique

PWM

(UART)

I²C

Figure 1 - Architecture système de la serre

# Architecture générale (multi couches – électronique/mécanique/software) :

Electronique : MSP430G2553

MSP430G2553

Servo-moteur

Capteur hydrométrique

Capteur UV

Module Bluetooth

Mécanique :

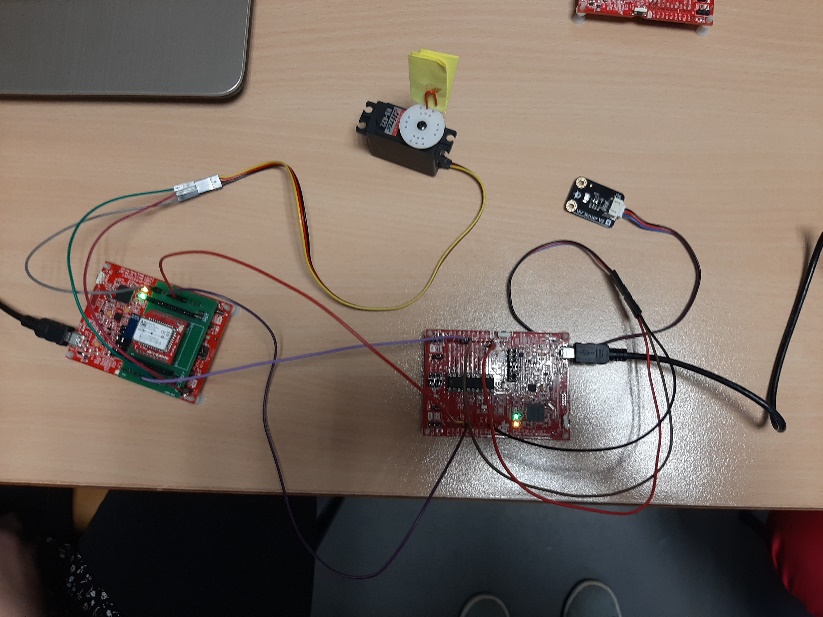


Figure : Montage serre connectée

Software :

*GEN\_001* : Gestion de l’ouverture de la serre via l’utilisation du servo-moteur par PWM.

*GEN\_002* : Gestion de la température par I²C

*GEN\_003* : Gestion de l’humidité par I²C

*GEN\_004* : Récupération des données du capteur UV par signal analogique.

*GEN\_005* : Envoi des commandes via Bluetooth ou interpréteur de commandes (Putty)

*GEN\_006* : Gestion des différents ports et bus de communications

# Architecture détaillée (fonctions prototypes – software) :

*DET\_001* : Recevoir les commandes

Description : Le micro-processeur doit être capable de recevoir les commandes de l’interpréteur Putty (ou Bluetooth) via une communication UART

Couverture : GEN\_005

Module : MSP430G2553

*DET\_002* : Interpréter et répartir/distribuer les commandes

Description : Le micro-processeur doit être capable d’interpréter les commandes pour pouvoir les distribuer sur les ports de communication des éléments associés.

Couverture : GEN\_006

Module : MSP430G2553

*DET\_003* : Fonction température haute

Description : Le capteur température doit détecter un niveau de température supérieur à 30°C. Cela doit déclencher l’allumage d’une led rouge et actionner l’ouverture de la porte de la serre via le servo-moteur. Cela doit déclencher l’envoi d’un message sur l’interpréteur de commande.

Couverture : GEN\_002

Module : MSP430G2553 / MSP430G2231

*DET\_004* : Fonction température basse

Description : Le capteur température doit détecter un niveau de température inférieur à 10°C. Cela doit déclencher l’allumage d’une led rouge et actionner la fermeture de la porte de la serre via le servo-moteur. Cela doit déclencher l’envoi d’un message sur l’interpréteur de commande.

Couverture : GEN\_002

Module : MSP430G2553 / MSP430G2231

*DET\_005* : Fonction température correcte

Description : Le capteur température doit détecter un niveau de température compris dans une plage entre 20°C et 30°C. Cela doit déclencher l’allumage d’une led verte. Cela doit déclencher l’envoi d’un message sur l’interpréteur de commande.

Couverture : GEN\_002

Module : MSP430G2553 / MSP430G2231

*DET\_006* : Fonction humidité haute

Description : Le capteur d’humidité doit détecter un niveau d’humidité supérieur à 90%. Cela doit déclencher l’envoi d’un message sur l’interpréteur de commande.

Couverture : GEN\_003

Module : MSP430G2553

*DET\_007* : Fonction humidité basse

Description : Le capteur d’humidité doit détecter un niveau d’humidité inférieur à 20%. Cela doit déclencher l’envoi d’un message sur l’interpréteur de commande.

Couverture : GEN\_003

Module : MSP430G2553

*DET\_008* : Fonction humidité correcte

Description : Le capteur d’humidité doit détecter un niveau d’humidité compris dans une plage entre 20% et 90%. Cela doit déclencher l’envoi d’un message sur l’interpréteur de commande.

Couverture : GEN\_003

Module : MSP430G2553

*DET\_009* : Récupération des données UV

Description : Le capteur UV doit être capable de récupérer les données UV de l’environnement ambiant de la serre et d’afficher sur l’interpréteur de commandes les valeurs obtenues avec un message d’information associé.

Couverture : GEN\_004

Module : MSP430G2553

*DET\_010* : Fonction ouverture de la serre via le servo-moteur

Description : Le servo-moteur doit pivoter dans le sens horaire lorsque la température perçue est supérieure au seuil défini par le client pour ouvrir la porte de la serre.

Couverture : GEN\_001

Module : MSP430G2231

*DET\_011* : Fonction fermeture de la serre via le servo-moteur

Description : Le servo-moteur doit pivoter dans le sens antihoraire lorsque la température perçue est inférieure au seuil défini par le client pour fermer la porte de la serre.

Couverture : GEN\_001

Module : MSP430G2231

*DET\_012* : Communication I²C

Description : La liaison entre le capteur hydrométrique et le MSP430G2553 doit être effectuée à l’aide d’une communication I²C.

Couverture : GEN\_002 / GEN\_003 / GEN\_006

Module : MSP430G2553

*DET\_013* : Communication SPI

Description : La liaison entre les deux MSP430G2553 doit être effectuée à l’aide d’une communication SPI.

Couverture : GEN\_006

Module : MSP430G2553

*DET\_013* : Communication UART

Description : La liaison entre l’entrée (interpréteur de commande ou module Bluetooth) et le MSP430G2553 doit être effectuée à l’aide d’une communication UART.

Couverture : GEN\_005 / GEN\_006

Module : MSP430G2553

# Tests Unitaires

## Fonctions concernant le servo-moteur :

Afin de tester le Servo moteur, nous l’avons branché sur le MSP43G2553, puis nous avons également branché ce microcontrôleur à un PC à l’aide d’un câble USB afin de communiquer avec le Servo moteur via l’interpréteur de commande PuTTY.

Après avoir configurer l’interpréteur PuTTY (configuration du port USB de communication, ici COM4) ainsi que télécharger le programme à tester sur le MSP430G2553, nous avons pu lancer notre script de test et évaluer les comportements du Servo Moteur obtenus.

Via l’interface PuTTY, en tapant la commande ‘H’ on obtient de l’aide sur les commandes possibles et en tapant la commande ‘O’ ou ‘F’, on peut activer le Servo Moteur qui ouvrira ou fermera la porte de la serre connectée.

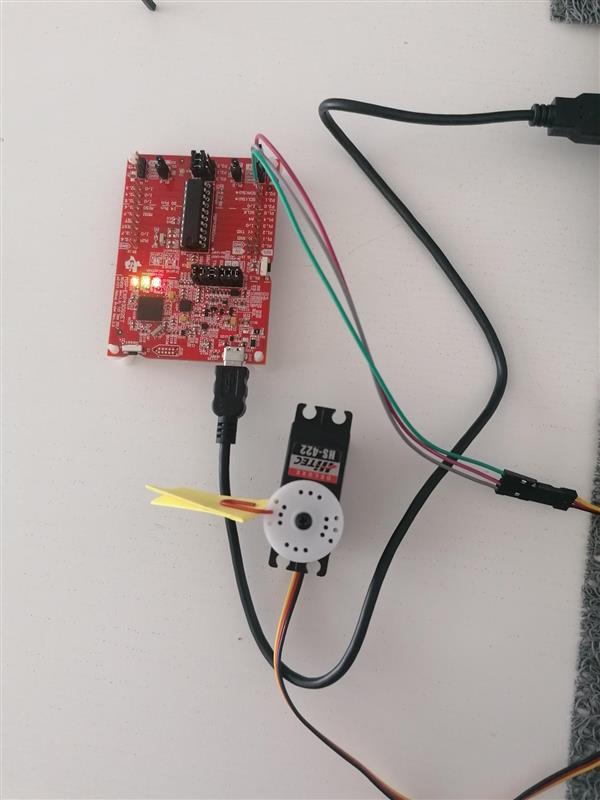
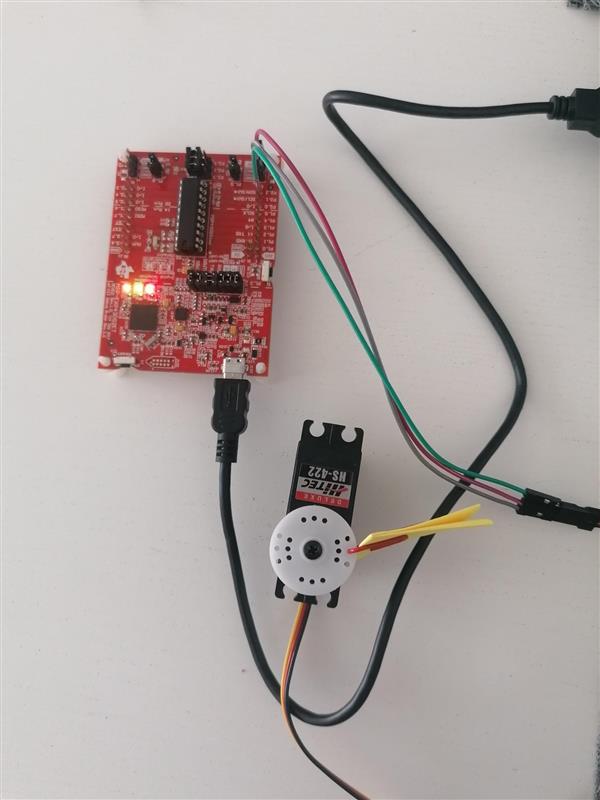


Figure 3: Ouverture et Fermeture de la Serre

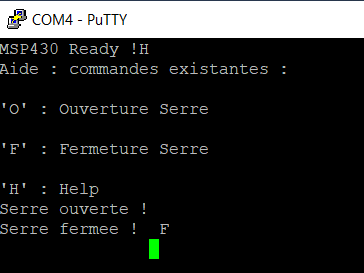


Figure 4: Interface Putty pour l'affichage des commandes réalisées

*TU DET\_010* : Fonction ouverture de la serre via le servo-moteur

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY | Coche |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY | Coche |  |
| Dans PuTTY taper la commande ‘O’ | Ouverture porte serre | Coche |  |

Figure 5 - Tests ouverture porte serre

*TU DET\_011* : Fonction fermeture de la serre via le servo-moteur

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY | Coche |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY | Coche |  |
| Dans PuTTY taper la commande ‘F’ | Fermeture porte serre | Coche |  |

Figure 6 - Test fermeture porte serre

## Fonction concernant le capteur UV :

*TU DET\_009* : Récupération des données UV

1. TU DET 009\_1 : Avec obstacle devant le capteur
2. TU DET 009\_2 : Sans obstacle devant le capteur

Pour réaliser le test unitaire du capteur UV, nous l’avons branché sur la MSP430G2553, que nous avons elle-même branché via USB au PC (cf. figure 1). Pour avoir une interface via PuTTY, il faut se mettre sur le port correspondant à la communication UART, ce que l’on retrouve dans le gestionnaire de périphériques. Dans notre cas et avec ce PC, il s’agit du COM8. Après avoir lancé PuTTY, il suffit d’aller dans CCS pour charger le programme dans la carte et le lancer. En tapant la commande ‘H’ on obtient de l’aide sur les commandes possibles, et en tapant ‘U’ on reçoit l’indice UV actuel (cf. figure 2). Dans notre cas un reçoit l’indice UV 4, ce qui est cohérent au vu des conditions (dans une salle avec peu de fenêtres, lumière de dos).

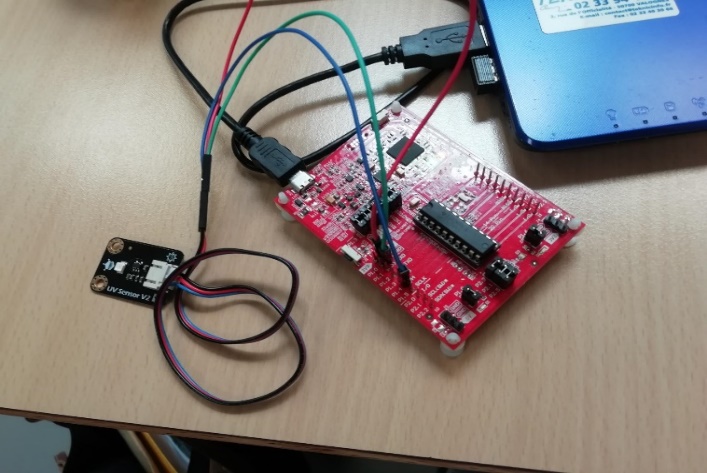


Figure 7 - Montage test unitaire capteur UV



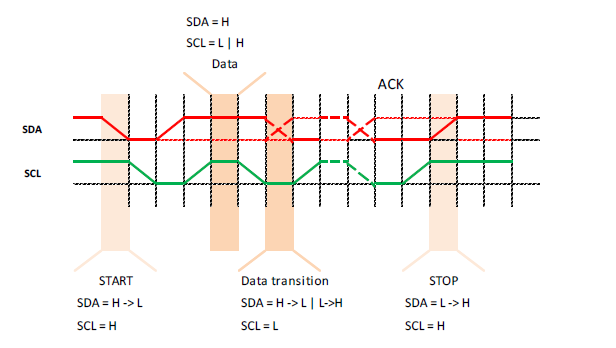
Figure 8 - Interface Putty pour l'affichage de l'indice UV

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY | Coche |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY | Coche |  |
| Dans PuTTY taper la commande ‘H’ | Affichage de l’aide | Coche |  |
| Dans PuTTY taper la commande ‘U’ | Affichage de l’indice UV actuel | Coche |  |

Figure 9 - Test fonctionnement capteur UV

## Fonction concernant le capteur hydrométrie et de température :

Comme indiqué plus tôt, nous n’avons pas été capable de mettre en œuvre la communication I²C. En effet le capteur hydrométrique nécessite une communication I²C, pour cela nous avons donc dû remplacer la carte esclave qui était un MSP430G2231 à l’origine par une MSP430G2553 car celle d’origine ne permettait pas la liaison I²C et SPI à la fois, contrairement au MSP430G2553 qui nous le permet.

La communication I²C se définit de la manière suivante :

* Un bit de START
* La trame de communication
* Un bit de STOP

N’ayant pas réussi à simuler le bit de STOP à envoyer au capteur afin de stopper la communication, et ne possédant pas de librairies fonctionnelles sur MSP430G2553, nous avons choisi de mettre de côté cette partie afin de nous concentrer d’avantage sur les autres parties de notre projet.

*TU DET\_003* : Fonction température haute

Test avec le capteur près d’une source de chaleur

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY |  |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY |  |  |
| Dans PuTTY taper une température supérieure à 30°C | Allumage LED rouge, ouverture porte serre et affichage message d’alerte |  |  |

Figure 10 - Test température trop haute

*TU DET\_004* : Fonction température basse

Test avec le capteur auprès d’un source froide

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY |  |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY |  |  |
| Dans PuTTY taper une température inférieure à 10°C | Allumage LED rouge, fermeture porte serre et affichage message d’alerte |  |  |

Figure 11 - Test température trop basse

*TU DET\_005* : Fonction température correcte

Test à air ambiant

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY |  |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY |  |  |
| Dans PuTTY taper une température comprise entre 10°C et 30°C | Allumage LED verte et affichage message d’information |  |  |

Figure 12 - Test température correcte

*TU DET\_006* : Fonction humidité haute

Test avec le capteur auprès d’une source d’humidité (eau)

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY |  |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY |  |  |
| Dans PuTTY taper une humidité supérieure à 90% | Allumage LED rouge et affichage message d’alerte |  |  |

Figure 13 - Test humidité trop haute

*TU DET\_007* : Fonction humidité basse

Test avec le capteur près d’une source de chaleur (feu)

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY |  |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY |  |  |
| Dans PuTTY taper une humidité inférieure à 10% | Allumage LED rouge et affichage message d’alerte |  |  |

Figure 14 - Test humidité trop basse

*TU DET\_008* : Fonction humidité correcte

Test à air ambiant

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY |  |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY |  |  |
| Dans PuTTY taper une humidité comprise entre 10% et 90% | Allumage LED verte et affichage message d’information |  |  |

Figure 15 - Test humidité correcte

## Fonction concernant le module Bluetooth :

*TU DET\_009* : Fonction Bluetooth

Test avec capteur UV

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer le terminal de communication Bluetooth sur le téléphone | Ouverture de l’interface PuTTY | Coche |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY | Coche |  |
| Via le terminal bluetooth taper ‘U’ | Affichage de l’indice UV actuel | Coche |  |

Figure 16 - Test module Bluetooth avec capteur UV

Afin de tester le capteur Bluetooth, nous l’avons branché sur le MSP43G2553, puis nous avons également branché ce microcontrôleur à un PC à l’aide d’un câble USB afin de communiquer avec le capteur Bluetooth via une application smartphone. Après avoir configuré l’application smartphone (Bluetooth Terminal) ainsi que télécharger le programme à tester sur le MSP430G2553, nous avons pu lancer notre script de test et évaluer les comportements du capteur Bluetooth obtenus. Via l’application smartphone, en tapant la commande ‘H’ on obtient de l’aide sur les commandes possibles et en tapant la commande ‘O’ ou ‘F’, on peut activer le servomoteur qui ouvrira ou fermera la porte de la serre connectée.

*TU DET\_009* : Fonction Bluetooth

Test avec servomoteur

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer le terminal de communication Bluetooth sur le téléphone | Ouverture de l’interface PuTTY | Coche |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY | Coche |  |
| Via le terminal bluetooth taper ‘H’ | Affichage de l’aide | Coche |  |
| Via le terminal bluetooth taper ‘O’ | Ouverture porte de la serre | Coche |  |
| Via le terminal bluetooth taper ‘F’ | Fermeture porte de la serre | Coche |  |

Figure 17 - Test module Bluetooth avec le servomoteur

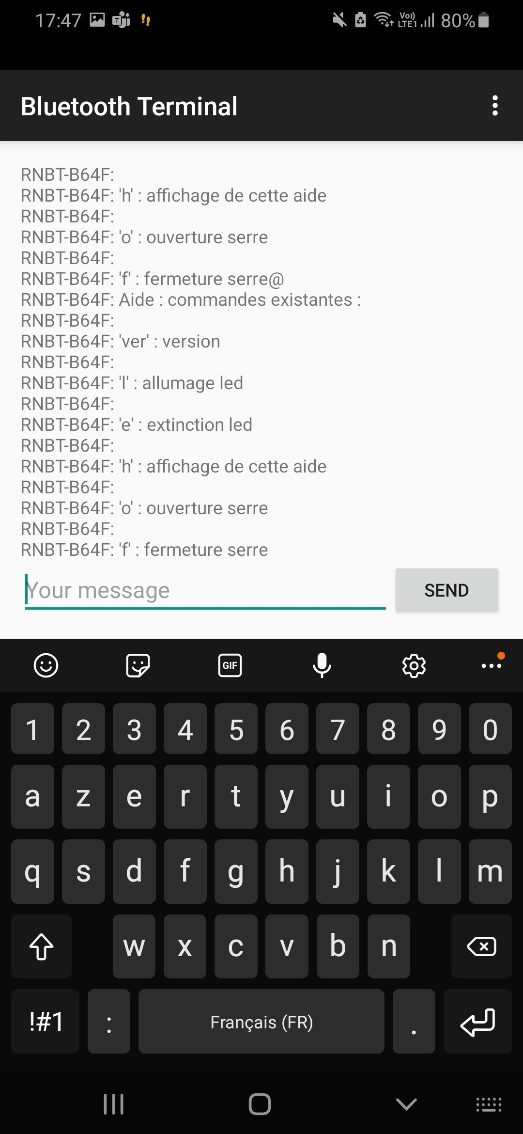


Figure - Affichage de l’aide

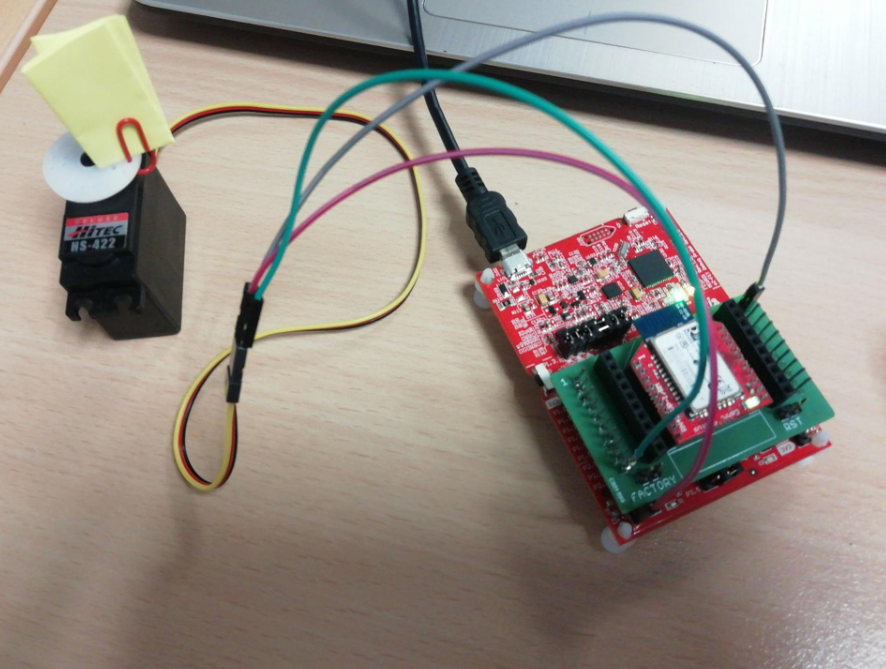
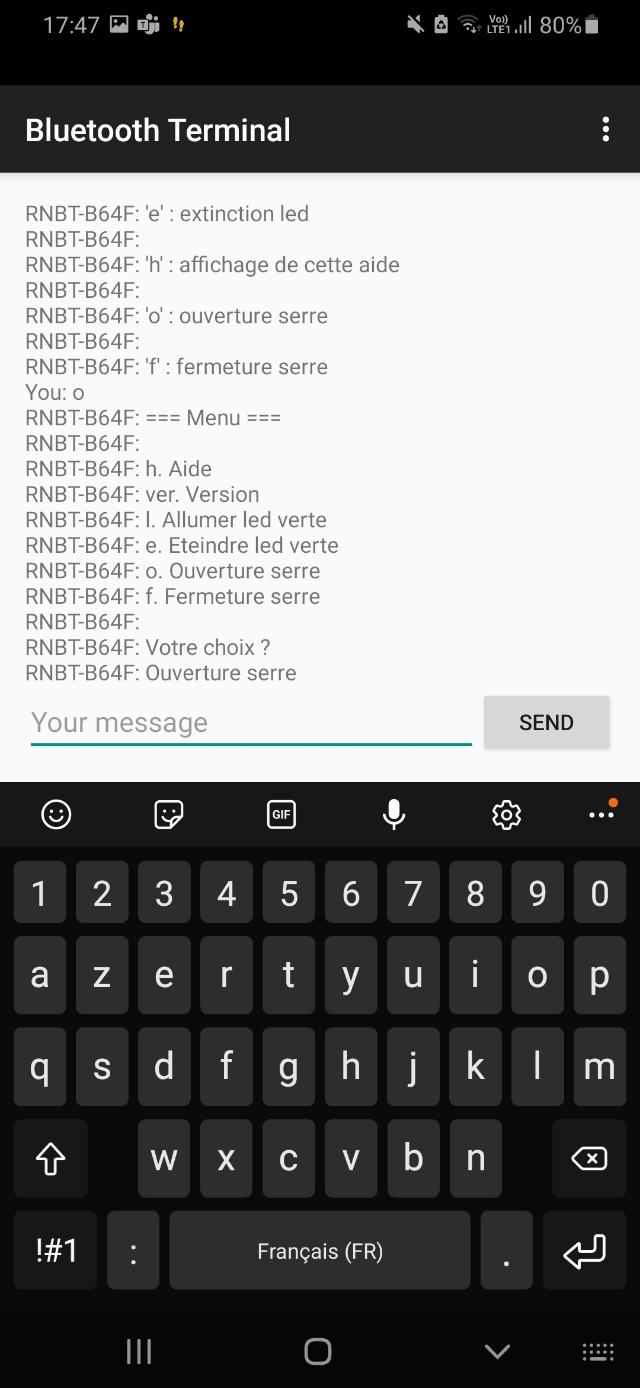
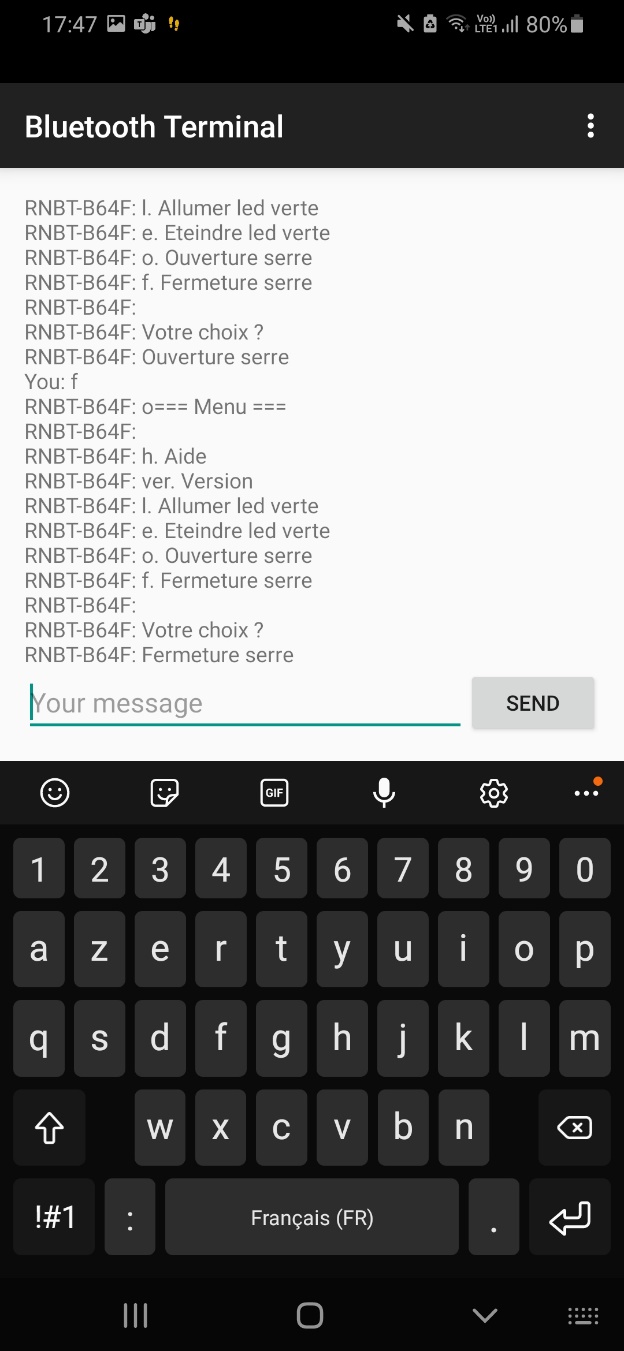
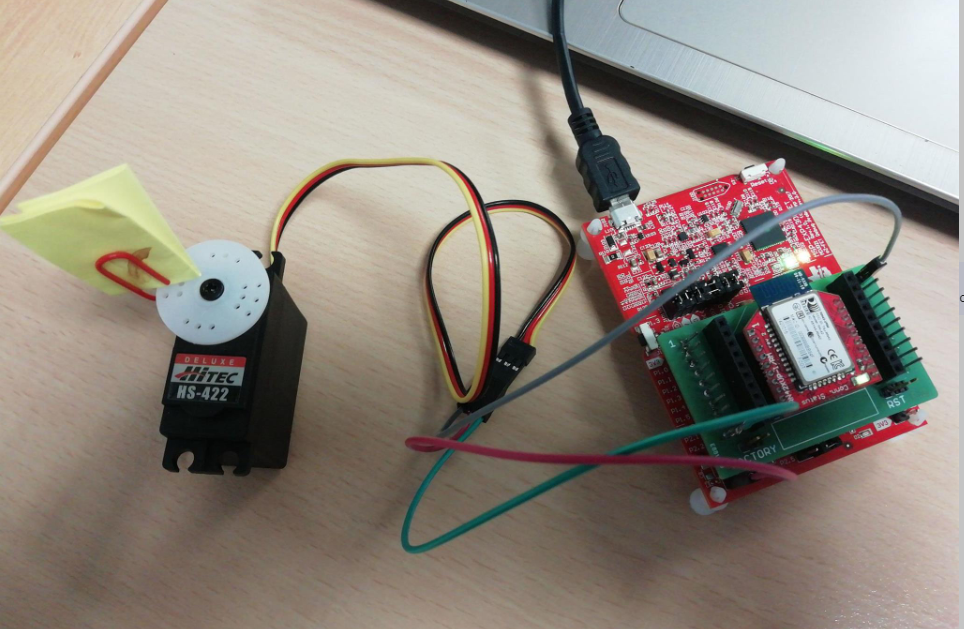
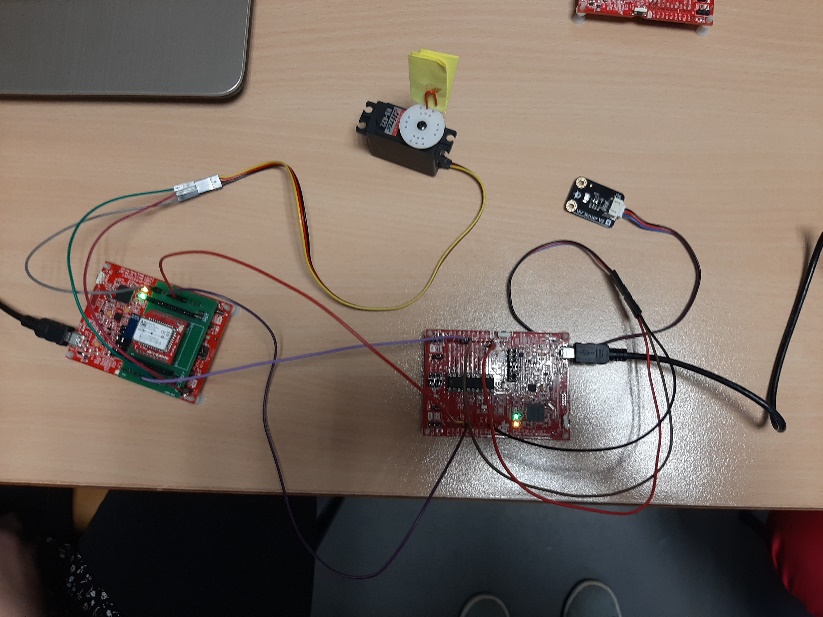
 **

Figure - Test module Bluetooth avec le servomoteur - ouverture porte serre

  
Figure - Test module Bluetooth avec le servomoteur – fermeture porte serre

# Test d’intégration

Maintenant que nous avons testé toutes nos fonctionnalités, il faut les intégrer au système final et tester si elles fonctionnent toutes ensembles. Pour cela nous avons brancher tous les composants ensemble suivant l’architecture de la figure 1, ce qui donne visuellement ceci :



*Rappel figure 2*

Voici les tests réalisés via l’interface PuTTY :

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer PuTTY en se connectant au bon port de communication | Ouverture de l’interface PuTTY | Coche |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY | Coche |  |
| Dans PuTTY taper une humidité comprise entre 10% et 90% | Allumage LED verte et affichage message d’information |  |  |
| Dans PuTTY taper une température comprise entre 10°C et 30°C | Allumage LED verte et affichage message d’information |  |  |
| Dans PuTTY taper une humidité supérieure à 90% | Allumage LED rouge et affichage message d’alerte |  |  |
| Dans PuTTY taper une température supérieure à 30°C | Allumage LED rouge, ouverture porte de la serre et affichage message d’alerte |  |  |
| Dans PuTTY taper une humidité inférieure à 10% | Allumage LED rouge et affichage message d’alerte |  |  |
| Dans PuTTY taper une température inférieure à 10°C | Allumage LED rouge, fermeture porte de la serre et affichage message d’alerte |  |  |
| Dans PuTTY, taper ‘H’ | Affichage de l’aide, des commandes disponibles |  |  |
| Dans PuTTY, taper ‘U’ | Affichage de l’indice UV | Coche |  |
| Dans PuTTY, taper ‘O’ | Ouverture de la porte de la serre | Coche |  |
| Dans PuTTY, taper ‘F’ | Fermeture de la porte de la serre | Coche |  |

Figure - Test d'intégration via PuTTY

Refaire les mêmes tests via l’interface Bluetooth :

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Résultat attendu | OK | NOK |
| Lancer le terminal de communication Bluetooth sur le téléphone | Bonne communication bluetooth | Coche |  |
| Lancement programme sous CCS | Pas d’erreur lors du lancement, affichage de « MSP430 Ready ! » dans l’interface PuTTY | Coche |  |
| Via le terminal bluetooth taper une humidité comprise entre 10% et 90% | Allumage LED verte et affichage message d’information |  |  |
| Via le terminal bluetooth taper une température comprise entre 10°C et 30°C | Allumage LED verte et affichage message d’information |  |  |
| Via le terminal bluetooth taper une humidité supérieure à 90% | Allumage LED rouge et affichage message d’alerte |  |  |
| Via le terminal bluetooth taper une température supérieure à 30°C | Allumage LED rouge, ouverture porte de la serre et affichage message d’alerte |  |  |
| Via le terminal bluetooth taper une humidité inférieure à 10% | Allumage LED rouge et affichage message d’alerte |  |  |
| Via le terminal bluetooth taper une température inférieure à 10°C | Allumage LED rouge, fermeture porte de la serre et affichage message d’alerte |  |  |
| Via le terminal bluetooth, taper ‘H’ | Affichage de l’aide, des commandes disponibles | Coche |  |
| Via le terminal bluetooth, taper ‘U’ | Affichage de l’indice UV | Coche |  |
| Via le terminal bluetooth, taper ‘O’ | Ouverture de la porte de la serre | Coche |  |
| Via le terminal bluetooth, taper ‘F’ | Fermeture de la porte de la serre | Coche |  |

Figure 22 - Test d'intégration via le module Bluetooth

# LDRA

Dans ce projet, nous utilisons un logiciel nommé LDRA mis à notre disposition pour visualiser la qualité logicielle de notre programme. Ainsi nous pouvons améliorer la qualité de notre code sans en altérer son fonctionnement. L’objectif dans un premier temps est d’améliorer chaque programme indépendamment.

## Servomoteur

Pour le programme pilotant notre servomoteur nous obtenons les résultats suivants :

Pour la partie « Code Review » : après avoir effectué les différentes modifications pour le code review nous obtenons un résultat de 100% d’acceptabilité.

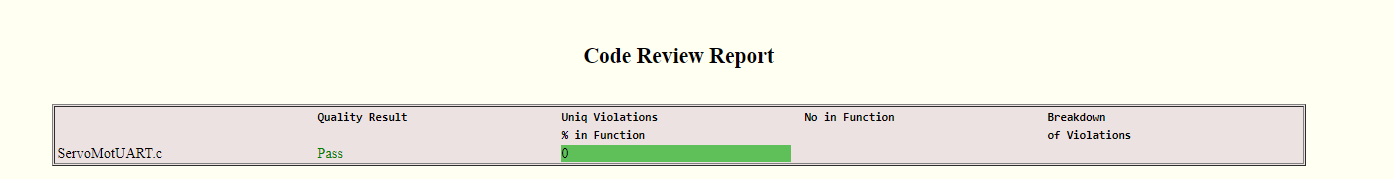


Figure -Test Manager Report\_ code review du programme pour le servomoteur

Cependant, on peut constater une erreur lors de l’analyse de notre code. Cette erreur est acceptable dans la mesure où le logiciel LDRA ne reconnait pas un de nos include (#include <stdio.h> ) alors que nous avons d’autres include. C’est une erreur due au logiciel.

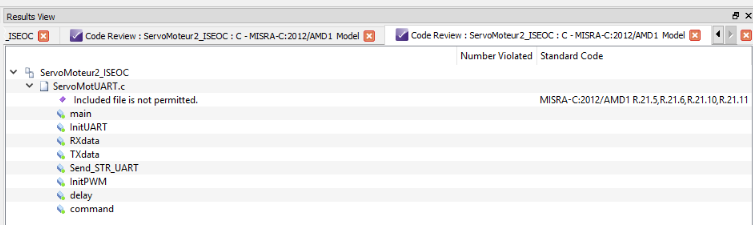


Figure – Code review du programme pour le servomoteur\_ erreur à corriger

Pour la partie « Quality Review » : après avoir effectué les différentes modifications pour le code review nous obtenons un résultat de 100% d’acceptabilité pour la testabilité et la maintenabilité. Par ailleurs, nous obtenons de bons résultats pour la clarté et les différentes métriques. Cependant toutes les modifications n’ont pu être terminées.

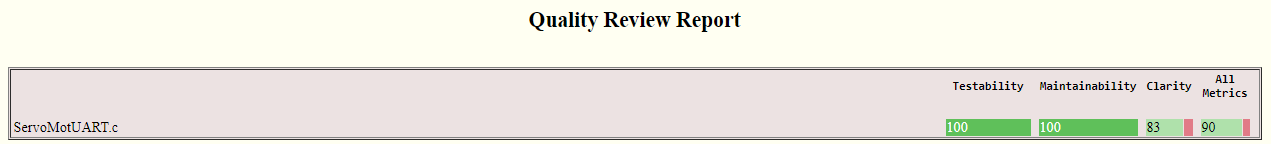


Figure – Test Manager Report\_ Quality review du programme pour le servomoteur

## Capteur UV

Pour le programme associé à ce module nous obtenons les résultats suivants:

Pour la partie « Code Review » : après avoir effectué une majeure partie des modifications, il nous reste encore quelques erreurs à corriger. C’est pourquoi nous n’obtenons pas un résultat de 100% d’acceptabilité.

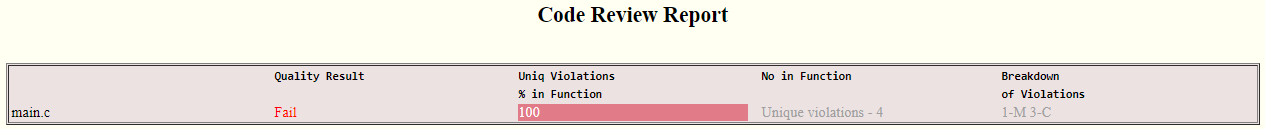


Figure - Test Manager Report\_ code review du programme pour le capteur UV

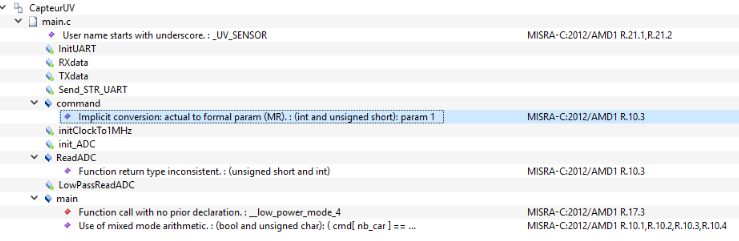


Figure – Code review du programme pour le capteur UV\_ erreurs à corriger

Pour la partie « Quality Review » : après avoir effectué les différentes modifications pour le code review nous obtenons un résultat de 100% d’acceptabilité pour la testabilité et la maintenabilité. Par ailleurs, nous obtenons de bons résultats pour la clarté et les différentes métriques. Cependant toutes les modifications n’ont pu être terminées.

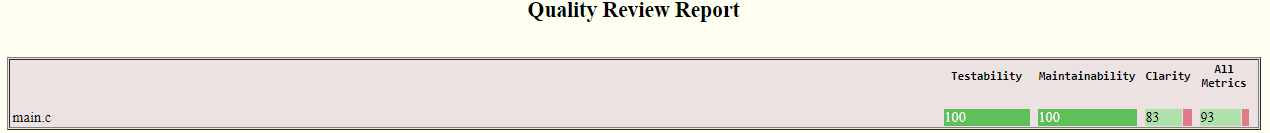


Figure – Test Manager Report\_ Quality review du programme pour le capteur UV

## Bluetooth

Pour le programme associé à ce module nous obtenons les résultats suivants:

Pour la partie « Code Review » : après avoir effectué une majeure partie des modifications, il nous reste encore quelques erreurs à corriger. C’est pourquoi nous n’obtenons pas un résultat de 100% d’acceptabilité.

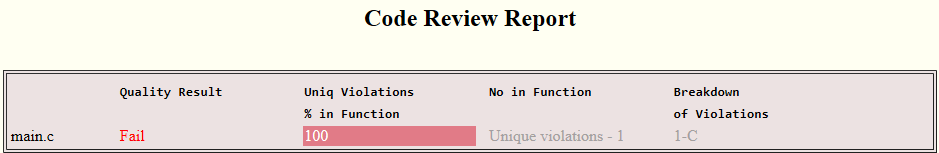


Figure – Test Manager Report\_ Code review du programme pour le module Bluetooth

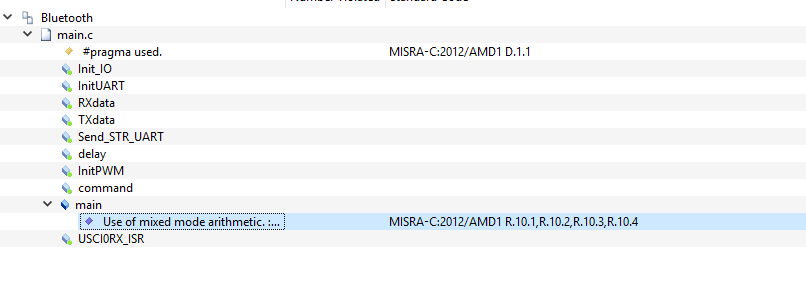




Figure – Code review du programme pour le module Bluetooth \_ erreurs à corriger

L’erreur est un problème de conversion entre deux types

Pour la partie « Quality Review » : après avoir effectué les différentes modifications pour le code review nous obtenons un résultat de 100% d’acceptabilité pour la testabilité et la maintenabilité. Par ailleurs, nous obtenons de bons résultats pour la clarté et les différentes métriques. Cependant toutes les modifications n’ont pu être terminées.

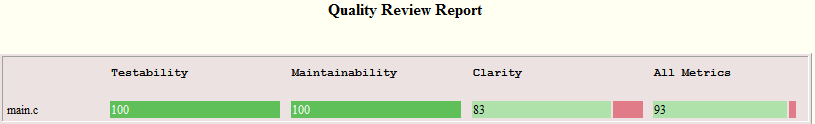


Figure – Test Manager Report\_ Quality review du programme pour le module Bluetooth

## Slave

Le code review n’est pas à 100% car il reste une erreur que l’on peut voir sur la figure 28.

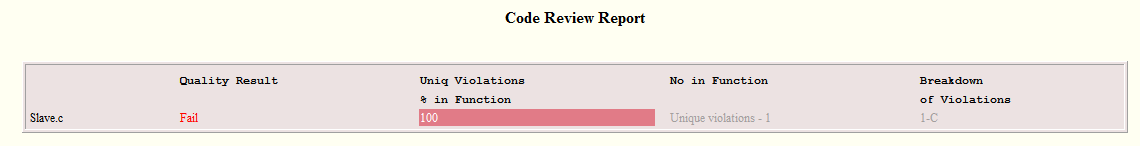


Figure - Test manager report - Code review pour le Slave

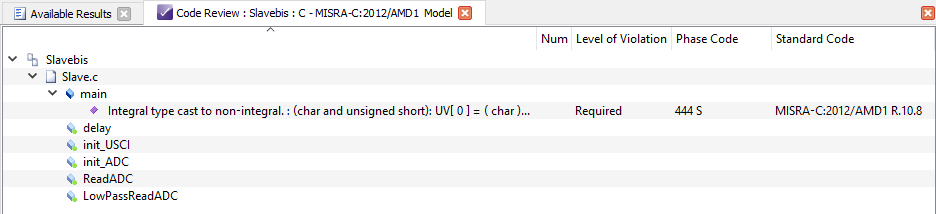


Figure 33 - Erreur restante Code review Slave



Figure - Lignes code pour erreur (cf figure 28)

Nous avons choisi de laisser de côté cette erreur car caster un int en char est exactement ce que nous voulons. Nous recevons le int (qui doit être traité précédemment en tant que tel) que nous souhaitons traiter en tant que caractère par la suite, cela nécessite donc un cast.

Pour la partie quality review, nous n’avons pas un code à 100% pour « Clarity » et « All Matrice » car un manque de temps ne nous a pas permis de terminer, en revanche « Testability » et « Maintainabilty » sont à 100%.

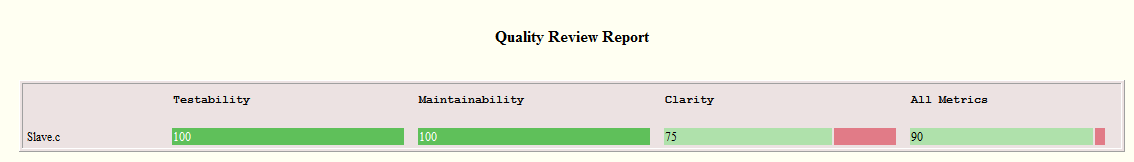


Figure 35 - Test Manager Report \_ Quality Review du programme Slave

## Master

Nous avons réalisé une analyse sur le script correspondant au master via le logiciel LDRA, et nous avons obtenu les résultats suivants.

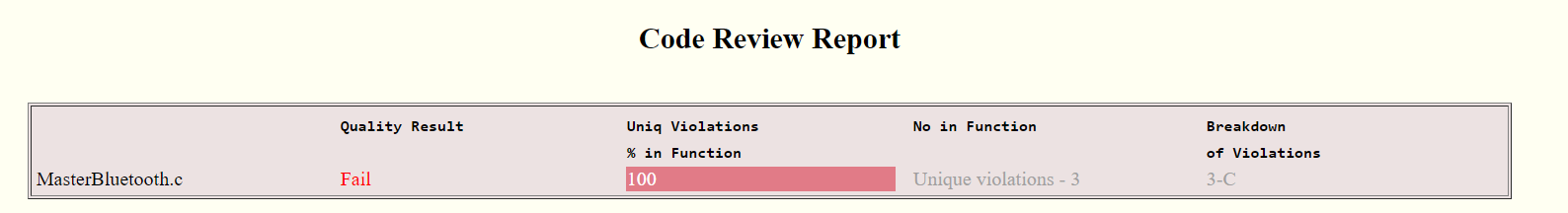


Figure : Test Manager Report\_ Code review du programme Master (avec Bluetooth)

Le code review de notre programme a été réaliser et les dernières erreurs obtenues (ci-dessous) correspondent à des erreurs de cast entre deux types différents à l’intérieur d’une fonction. Or, le cast (UCHAR)\*cmd correspond à l’attendu de la fonction qui appelle cette variable.

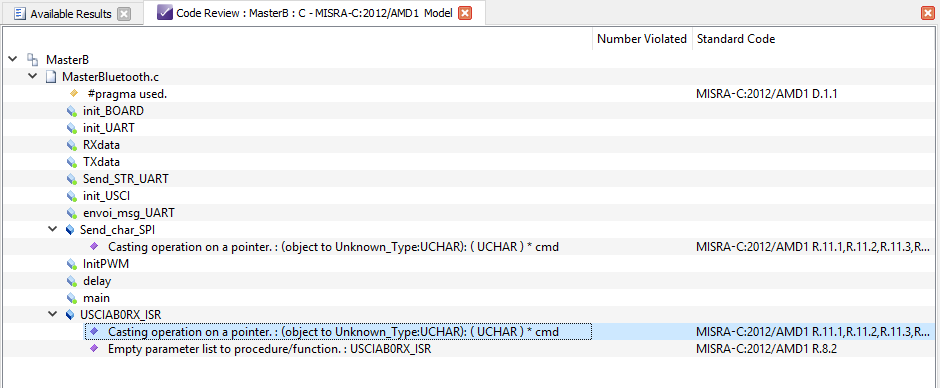


Figure : Code Review du programme Master(avec Bluetooth) \_ erreurs à corriger

Concernant la partie « Quality Review » du manager Report, celui-ci indique une qualité de code définie à 100% pour le « Testability » et « Maintainabilty ». Cependant, nous n’avons pas pu terminer la correction du quality review afin d’arriver à une qualité de 100% pour les 2 autres critères soit « Clarity » et « All Matrice ».

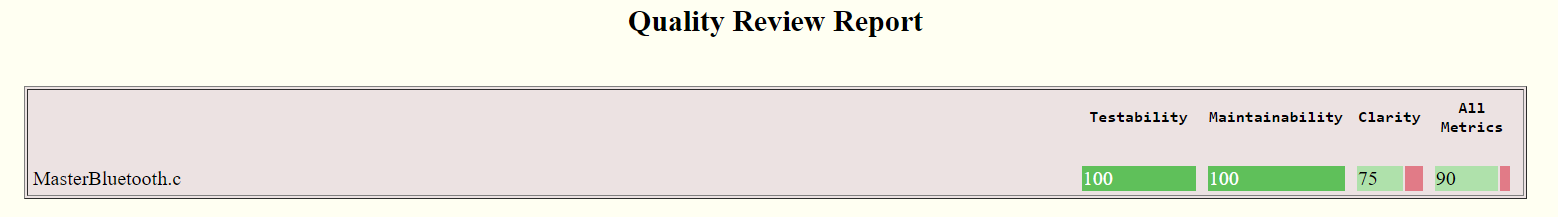


Figure - Test Manager Report \_ Quality Review pour le programme Master (avec Bluetooth)

# Lien GITHUB :

<https://github.com/Sarahguill/Projet.git>