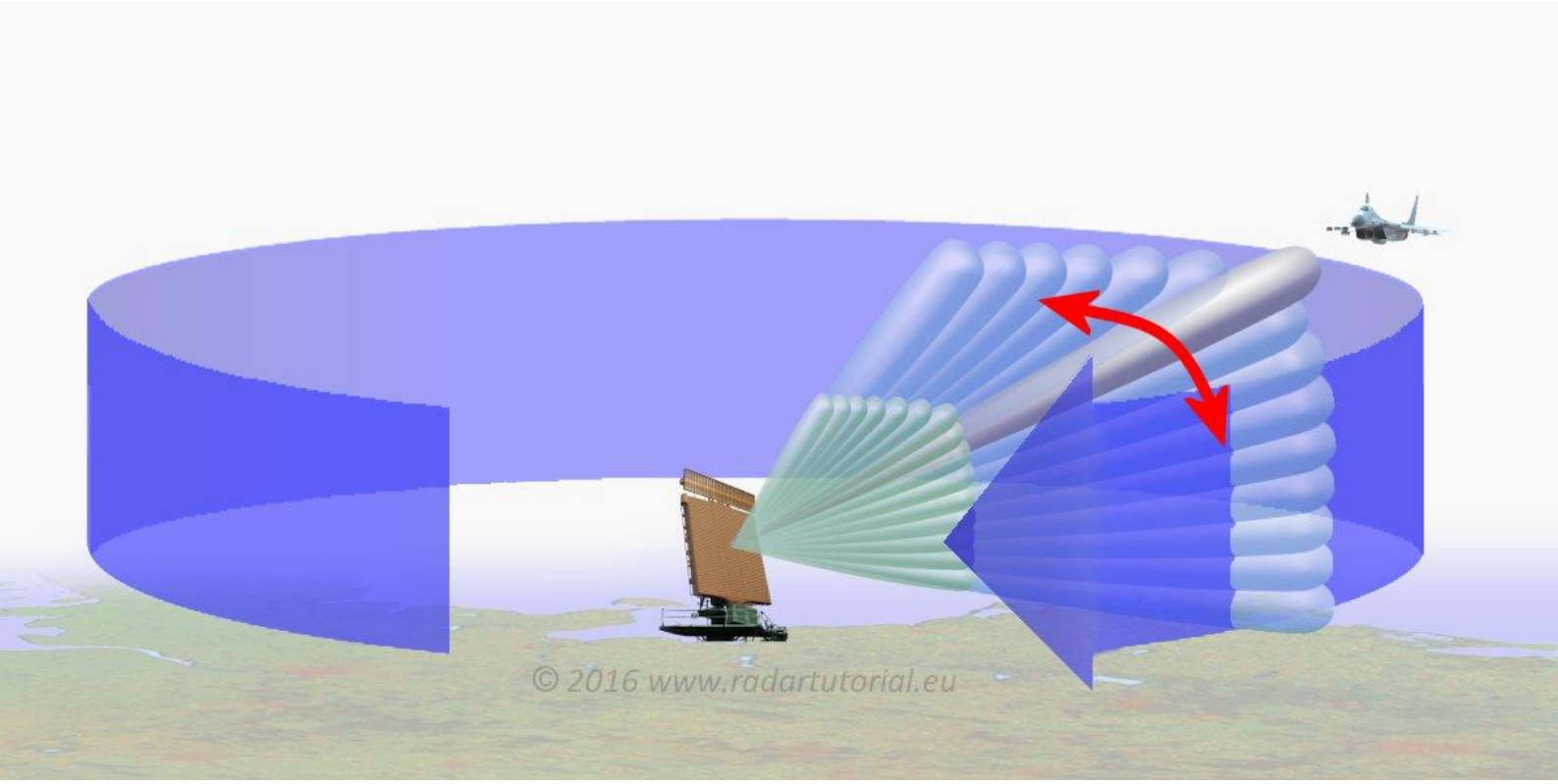
Compte rendu Mini-Projet

Radar 2D sur SoC-FPGA

Raimbaud Rio
12 janvier 2022

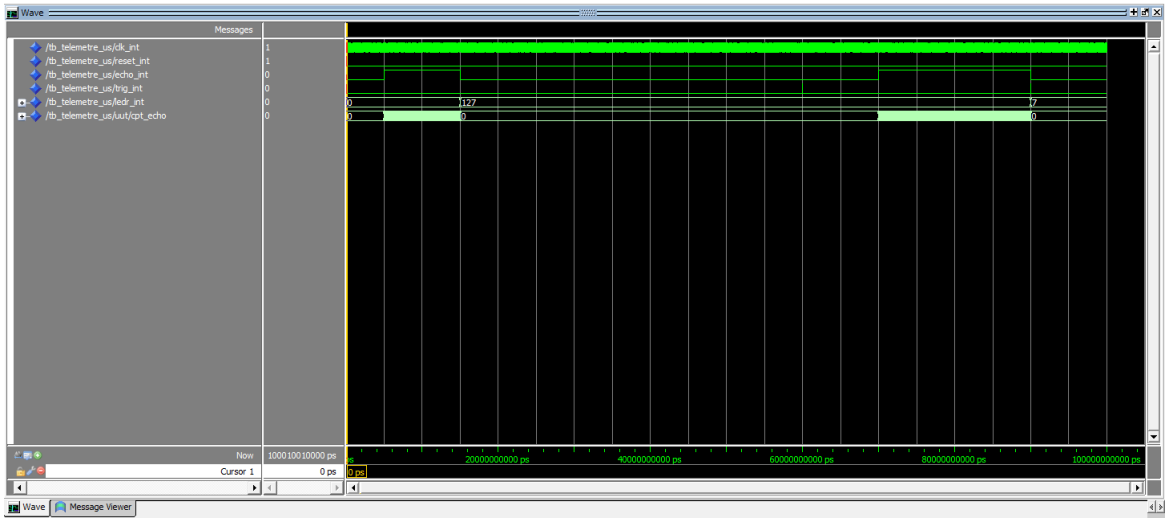


IP télémètre ultrason HC SR04

Simulation de l'IP (Modelsim)

La première étape a été de décrire le composant HC_SR04 en VHDL avec les informations sur le fonctionnement du capteur afin de lire sa valeur correctement. Il faut pour cela utiliser les signaux « trig » et « echo » du capteur.
Le test de simulation sur Modelsim donne ceci et est expliqué en dessous.

Capture Modelsim :



Le signal « echo » vaut '1' tout au long de l'aller-retour de l'onde émise.
Plus exactement, lors de l'émission (i.e « trig » = '1'), « echo » est à l'état '1' jusqu'à ce que cette onde soit retournée vers le module HC-SR04. A cet instant, « echo » valant '0', « cpt_echo » arrête de s'incrémenter, et sa valeur finale sera proportionnelle au temps de parcours de l'onde.

La relation entre vitesse, distance et temps étant connue de tous, il est aisément possible de retrouver la distance nous séparant de l'obstacle rencontrée par cette même onde.
En fonction de cette distance, on construit un vecteur de bits LEDR qui, pour chaque dizaine de centimètres déterminé par le capteur, allumera une LED rouge de notre FPGA.

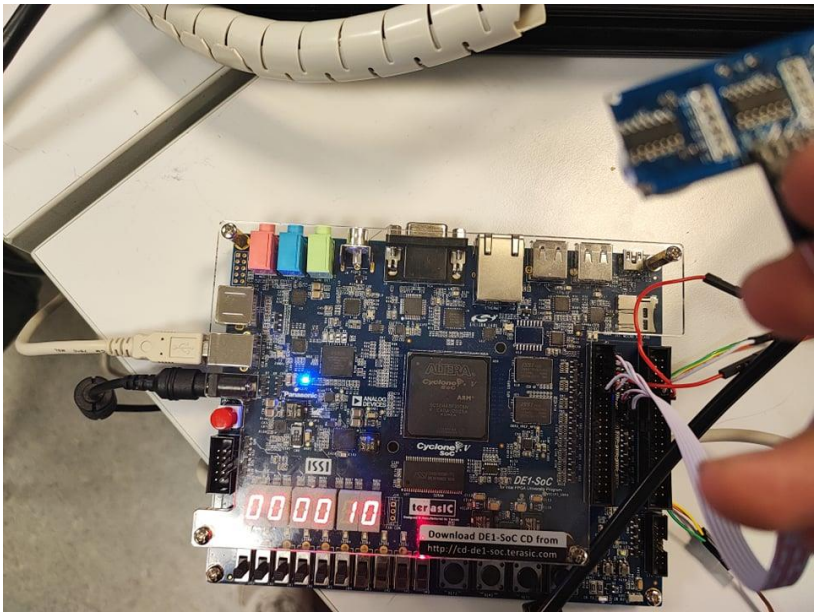
Le pallier de 10cm est arbitraire, mais il permet d'avoir une assez bonne approximation de la distance mesurée.

Synthèse de l'IP (Quartus)

La seconde étape consistait en la synthèse de notre IP précédemment créée et simulée. Les résultats de la simulation étaient ceux escomptés, et nous n'avons utilisé aucun outil quelconque fourni par le langage de design hardware VHDL impossible à synthétiser.

Nous avons de solides raisons de penser que si la synthèse pose un problème, le code source VHDL réalisé à l'étape précédente ne sera en rien responsable.
La synthèse en photo donne ceci et est expliqué en dessous.

Photos sur la carte SoC (la table se trouve exactement à 10cm), l'affichage sept segments est un ajout de notre part et n'était pas alors demandé à cet instant.



Situé à exactement 10 cm de la table, le capteur nous renvoie une valeur correcte de cette distance, qui nous permet d'allumer une seule LED rouge d'une part, ainsi que d'afficher cette valeur « 10 » sur nos afficheurs sept segments.
Le système est fonctionnel.

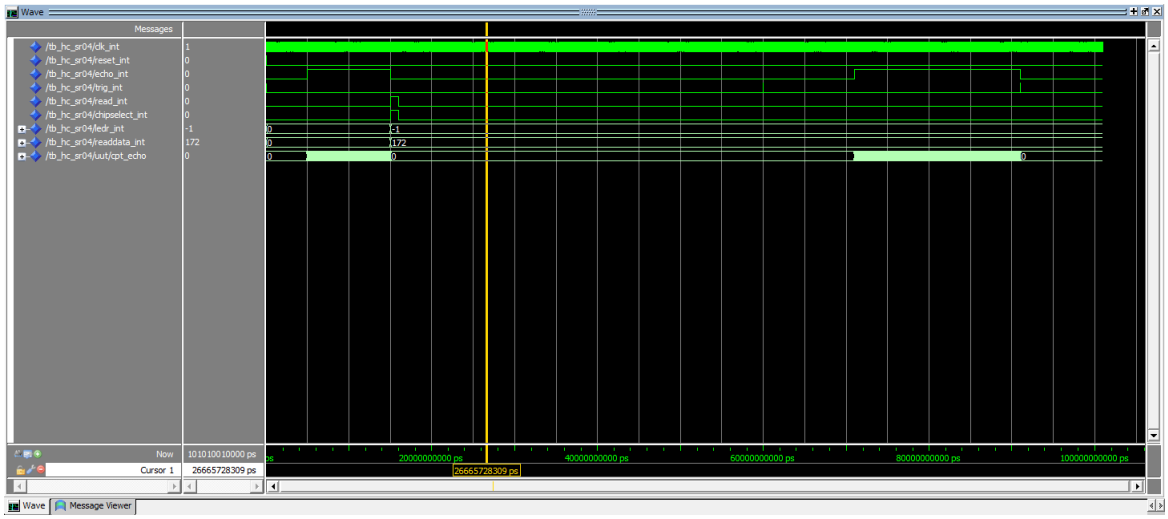
IP Avalon télémètre ultrason HC SR04 dans QSYS

Simulation de l'IP (Modelsim)

Ensuite, sur Qsys, nous avons créé et intégré l'IP Télémètre. Pour cela on ajoute certains ports comme le read et chipselect en entrée, ainsi que le readdata en sortie. Ces ports supplémentaires sont significatifs de l'interface Avalon, qui permet entre autres de lire et d'écrire sur ce bus depuis un programme C (software). Les valeurs écrites sont alors interprétées par notre entité VHDL (hardware), et ses sorties alors lisibles depuis le soft en dépendront.

On teste alors dans un premier temps l'intégration de cet bus Avalon sur Modelsim : le résultat est le suivant, et est expliqué plus bas.

Capture Modelsim :



Le système utilisant le même capteur, il est strictement identique à la précédente simulation à quelques nuances près.

Les entrées « read » et « chipselect » sont mises à l'état '1' lorsque l'on cherche à lire la distance mesurée par le module HC-SR04. Dans ce cas, cette distance est écrite dans la sortie « readdata » de l'IP.

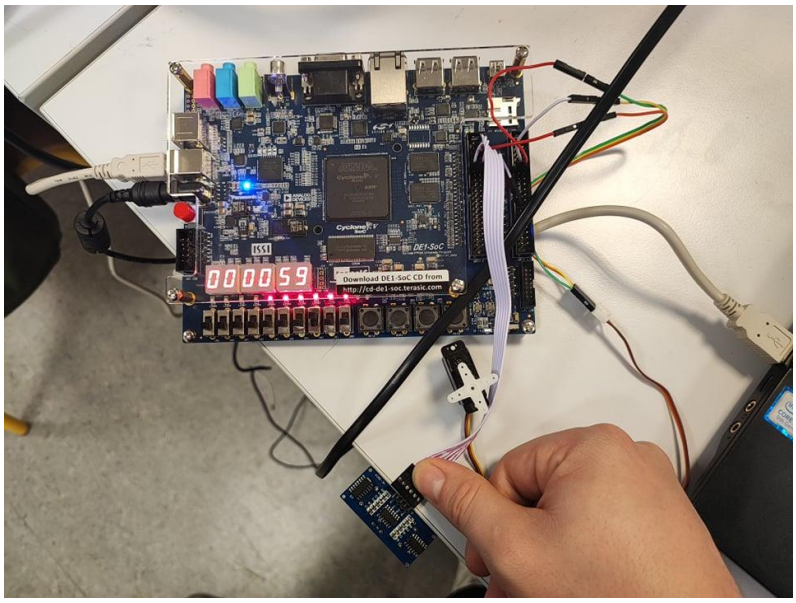
Les résultats de cette modulation sont ceux escomptées, nous pouvons passer à l'étape suivante.

Synthèse de l'IP (QSYS, Quartus et Eclipse)

Cette fameuse prochaine étape consiste encore une fois en la synthèse de notre IP, mais pas en fichier TOP-level unique comme lors de la précédente synthèse. Cette fois-ci, il fallait créer un Custom Peripheral ayant notre IP comme code source, lui associer sa clock et son reset et exporter des sorties pour relier l'ensemble de l'IP au bus Avalon, comme désiré.

Il nous a suffit de suivre les indications du premier et deuxième TP pour réaliser cette étape, et puisque la simulation de notre IP nous a confirmé le bon comportement de notre code, nous savions à nouveau que les potentielles erreurs auraient comme source les différentes suites logicielles utilisées par Quartus.
Mais finalement, nous sommes arrivés à nos fins, et après synthèse puis programmation software en C sur Eclipse, nous sommes parvenus au résultat illustré et expliqué sur la photo qui suit :

Photo sur la carte SoC (la table se trouve exactement à environ 60 cm du sol).



L'écriture sur les afficheurs sept segments est totalement programmée en C, tout comme la lecture de la valeur mesurée par le capteur. Tout ça est rendu possible grâce à l'utilisation du bus Avalon, et de la liaison entre soft et hard qu'il permet. Satisfaits, nous pouvons maintenant attaquer la deuxième grande partie de ce mini projet d'architecture matérielle.

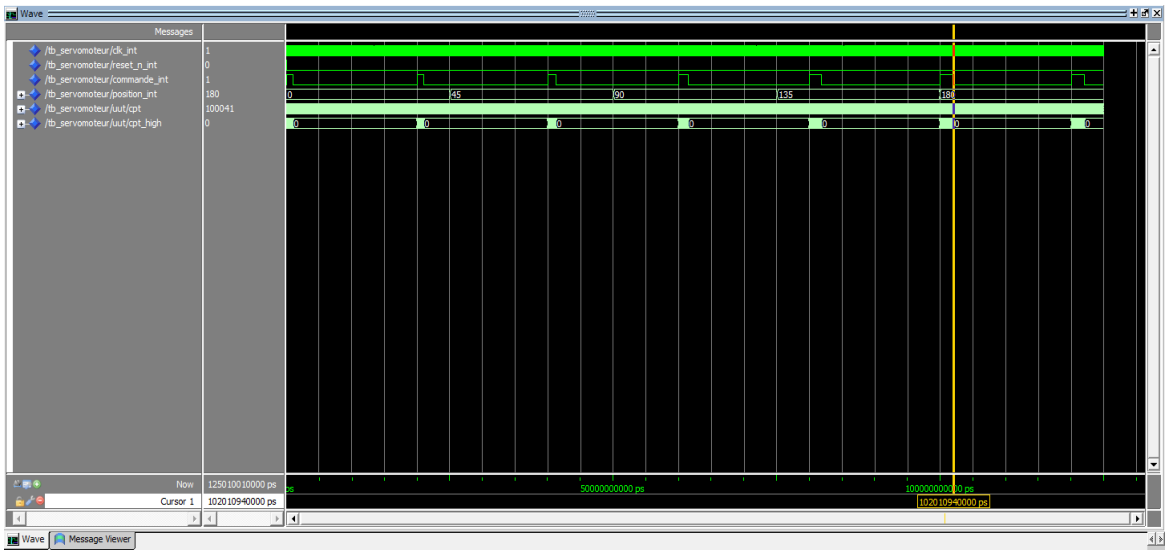
IP Servomoteur

Simulation de l'IP (Modelsim)

De façon totalement analogue à la première partie du projet, il nous faut décrire le composant en VHDL. Avec les informations données dans le sujet, nous avons compris l'importance du rapport cyclique sur la position angulaire su servomoteur, et tout l'intérêt qu'il faut porter au temps haut et bas du signal carré que doit recevoir ce composant en entrée.

Le test de simulation sur Modelsim donne ceci et est expliqué en dessous.

Capture Modelsim :



L'explication de cette simulation d'architecture sera plus rapide que la première.

Notre sortie possède deux états : un état « haut », qui implique l'écriture '1' sur la sortie « commande » du servomoteur, et un état « bas » qui résulte en une commande à l'état '0'.

En fonction de la valeur d'une entrée « position », le temps T_{haut} change et le rapport cyclique du signal de sortie s'adapte, modifiant la position du servomoteur. Nous utilisons ici aussi des compteurs pour contrôler le temps haut et le temps bas de notre entité.

Vous trouverez toute l'algorithmie et les calculs associés en de plus amples détails dans nos codes sources, mais vous remarquez bien sur la simulation un élargissement du temps haut alors que « position » passe de 0 à 180.

Synthèse de l'IP (Quartus)

Synthétiser l'IP en TOP-level n'a posé aucun problème. Nous avions connecté les signaux de sorte à contrôler l'angle en entrée du servomoteur grâce aux interrupteurs présents sur la carte.

L'angle ainsi émis correspondait à la valeur binaire de nos interrupteurs, le bit de poids faible étant le plus à droite.

(Photos de l'oscillo pour vérifier le timing ?)

(Photos du système en fonctionnement TOP-level uniquement ?)

IP Avalon Servomoteur dans QSYS

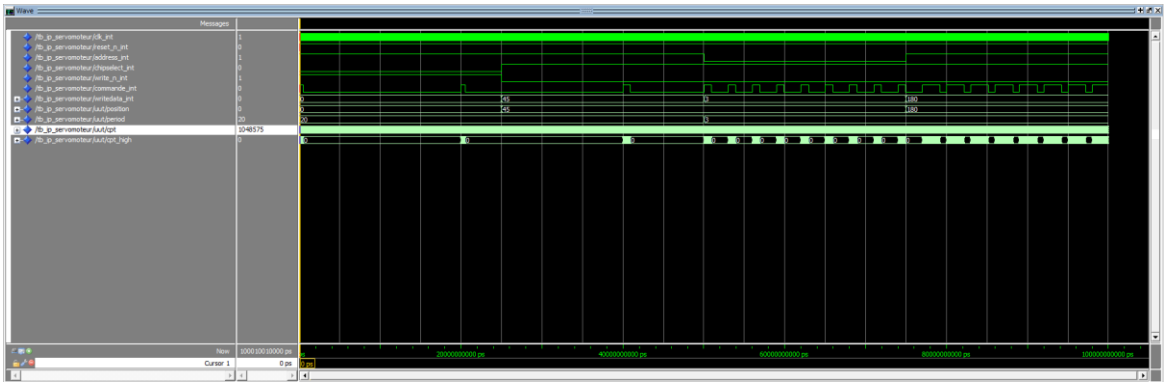
Simulation de l'IP (Modelsim)

Ensuite, tout comme la démarche précédente avec le télémètre, nous créons la nouvelle IP avec des ports supplémentaires significatifs de l'interface Avalon, tels que le port « chipselect », « address », « write_n » et « WriteData » en entrée. Un dernier port, en sortie cette fois, est le « command ».

Le port « address » est ignoré sous consigne de M.Douze.

On teste alors dans un premier temps l'intégration de cet bus Avalon sur Modelsim : le résultat est le suivant, et est expliqué plus bas.

Capture Modelsim :



A l'instar de la simulation de l'IP Avalon du HC-SR04, il n'y a rien de plus à simuler que l'écriture fonctionnelle dans l'entrée « writedata » de l'IP lorsque les signaux « chipselect » et « write » valent respectivement '1' et '0'.

Le comportement est celui attendu, le système est fonctionnel en simulation. Il reste à le synthétiser désormais.

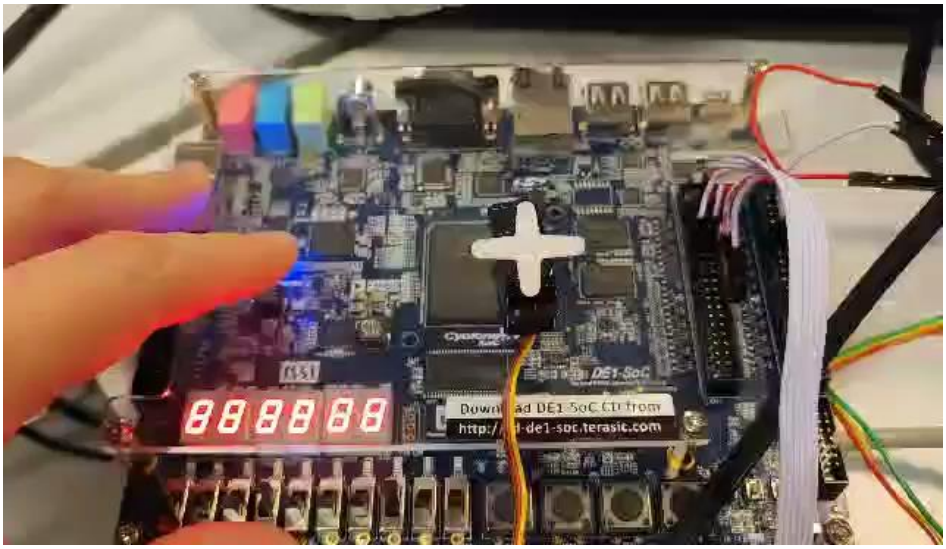
Synthèse de l'IP (QSYS, Quartus et Eclipse)

Nous devons une fois encore synthétiser de notre IP, non pas en fichier TOP-level unique mais avec un Custom Peripheral ayant notre IP comme code source (lui associer sa clock et son reset comme toujours, et exporter des sorties pour relier l'ensemble de l'IP au bus Avalon).

La démarche à suivre était celle déjà suivie précédemment, lors de la synthèse de notre IP Avalon HC-SR04.

Après avoir effectué l'interface de liaison des signaux, la synthèse puis la programmation software en C sur Eclipse, nous sommes parvenus au résultat suivant :

Vidéo du servo moteur en fonctionnement : entrée binaire sur 8 bits correspondant à un angle en degré.



La lecture des valeurs des interrupteurs (switch) ainsi que l'écriture de cette valeur binaire en entrée du servomoteur ont été totalement programmées et réalisées en C.

Le bus Avalon joue toujours ce rôle d'intermédiaire dans l'échange d'informations entre soft et hard.

Désormais il nous faut combiner nos deux grandes parties en un projet solide : il s'agit de la dernière question.

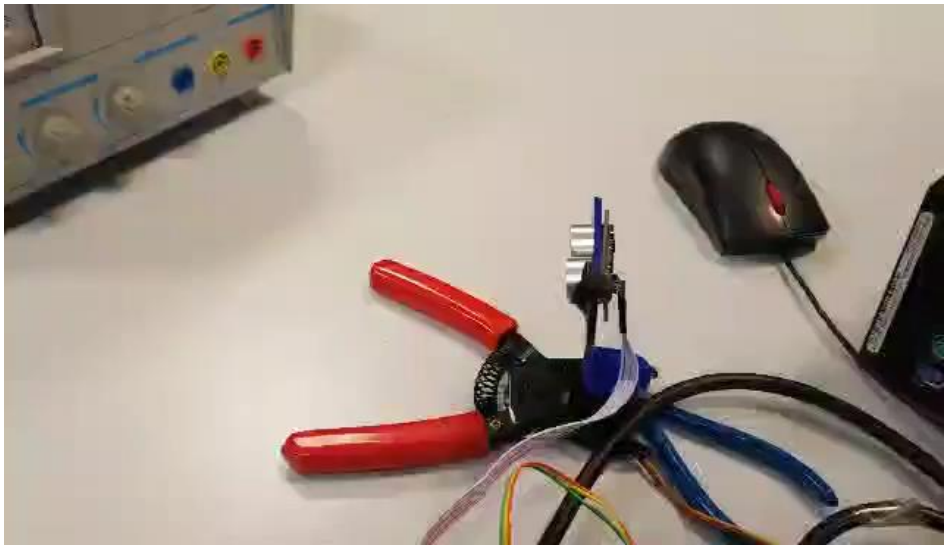
Affichage des obstacles

Synthèse du système (QSYS, Quartus et Eclipse)

Pour la dernière partie il faut créer une boucle qui fait varier les valeurs d'angles degrés par degrés en entrée du servomoteur afin d'afficher en temps réel la distance mesurée par le télémètre (qui a été placé au-dessus du servo moteur) sur une vue globale du champ autour du radar sur 180°.

On affiche les distances lues sur les afficheurs sept segments ainsi que dans la console d'Eclipse à chaque nouvelle variation d'un degré du servomoteur.

Voici une vidéo ou l'on voit bien le système en fonctionnement, affichant la distance en centimètres sur les afficheurs 7 segments.



Annexes

Codes vhdl des différentes entités.