|  |
| --- |
| Lycée raymond queneau |
| COMPTE RENDU |
| TP Commande Barrière |
|  |
| **Genest Axel - Chauveau Aurélien** |
|  |

|  |
| --- |
|  |

Introduction:

Notre TP consiste a créé un système pour une barrière automatique. Il lira les boucles aval et amont afin de détecter ou non la présence d'un véhicule. Si le véhicule se situe sur la boucle aval, la barrière s'ouvrira pour laisser sortir le véhicule et se refermera après sont passage. Le moniteur du gardien affichera donc qu'un véhicule est sortie. Si le véhicule se trouve sur la boucle amont, un message de bienvenue s'affichera sur un afficheur LCD, ainsi qu'une demande de présenter une carte client dans le lecteur de carte ou un code à inscrire sur le clavier situé sous l'afficheur LCD. Si la carte ou le code ne sont pas valide l'accès au parking est donc refuser et un message demandant au client de quitter la boucle s'affiche. Si la carte ou le code est validé la barrière s'ouvre et laisse donc l'accès au parking avant de se refermer après le passage du véhicule. Si l'accès a été validé mais que le véhicule ne rentre pas dans le parking la barrière se refermera au bout de 30sec et un message demandant au client de quitter la boucle s'affichera.

Photo de la barrière.

L'I2C dans notre TP :

Dans se TP nous utilisons le bus I2C afin de communiquer entre la carte Arduino et le système. Nous avons pour cela du adapter un câble RJ45 pour pouvoir communiquer en i2C. Nous avons isoler 4 fils : le fil blanc/bleu utiliser pour la masse (GND), le fil bleu pour l'horloge(SCL) et les fils blanc/orange et marron pour les données(SDA). Nous nous retrouvons donc avec les 3 fils typique de l'i2C qui sont donc le SDA (Serial DAta), le SCL (Sérial CLock), et le GND (GrouND). Le Sérial DAta permet de communiquer les données, ses dernières son synchronisé via les pulsations d'horloge envoyer par le Sérial CLock. La synchronisation des donnée est importante elle permet une bonne compréhension entre les deux systèmes.

photo du câble

Au début du TP nous devions, pour communiquer avec la barrières, brancher notre adaptateur sur notre port RJ45 du réseaux local (le port 134) et ensuite, dans la baie de brassage, relier notre port (le port 134) au port RJ45 de la barrières (le port 114) et ainsi communiquer. Mais nous étions 5 groupes sur se même TP et donc cela demander une certaine organisation et de nombreux aller-retour à la baie de brassage pour se remettre en communication avec le système. L'avantage c'est que l'on ne pouvais pas communiquer à plusieurs en même temps sur le système.

photo de la baie de brassage

Par la suite une pieuvre a été installée, elle permet de mettre un port choisie pour chaque groupe en relation direct avec le port de la barrière. Se qui nous évitas de faire des allers retours à la bais de brassage pour remettre les port en communication . L'inconvénient c'est que cela demander beaucoup plus de rigueur au niveau du passage des commande sur la barrière entre chaque groupe car il fallait absolument éviter de communiquer en même temps sur le systèmes.

photo de la pieuvre

Une fois que nous avions installé la pieuvre nous nous somme rendu compte que le système de la barrières n'exécutait plus correctement les programmes qui auparavant fonctionnais parfaitement sans la pieuvre. Nous avons donc réfléchis beaucoup de temps avant de nous rendre compte que la pieuvre provoquais des perturbations magnétiques dans le fils qui communiquais avec la barrière et que la distance qui nous séparais de la barrière en câblage était à présent aussi à prendre en compte. Pour palier à se problème nous avons donc réduit la vitesse de transmission via la bibliothèque i2C dans notre programme. En passant un certain bit à 1 nous avons divisée la vitesse de transmission du CPU de la carte Arduino se qui à par conséquence réduit l'effets de la distance et des perturbations magnétiques sur les données transmise. N'ayant plus de problème de communication nous avons pus continuer notre TP.

photo bibliothèque i2C

Les débuts du TP :