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

|  |
| --- |
|  |

Introduction:

 Notre TP consiste a créer 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 son passage. Le moniteur du gardien affichera donc qu'un véhicule est sorti.

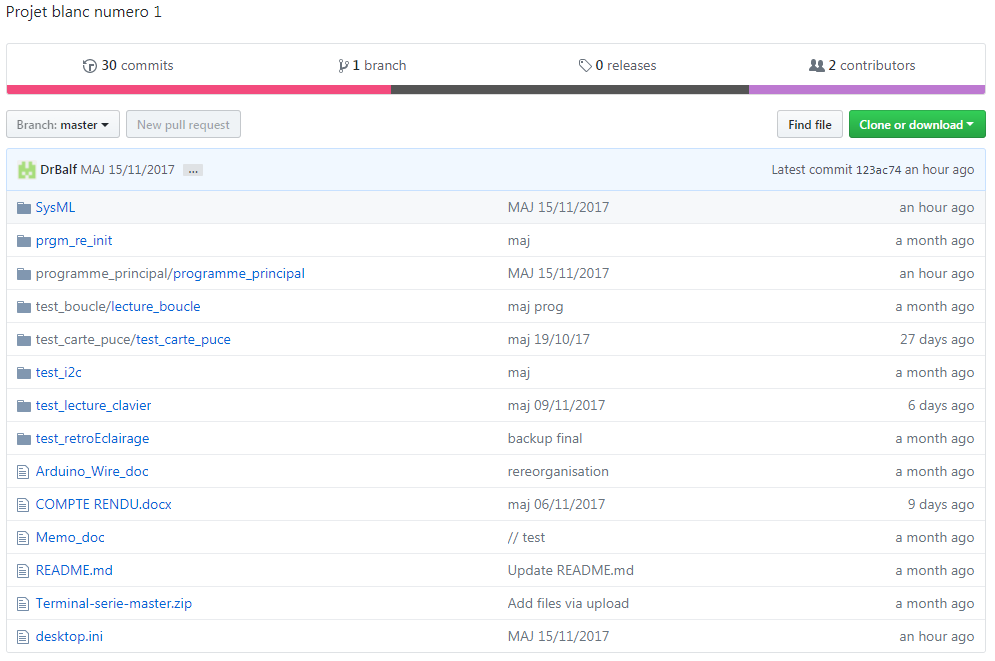
Si le véhicule se trouve sur la boucle Amont, un message de bienvenue s'affichera sur un afficheur LCD, ainsi qu'une demande de présenter une carte client dans le lecteur de carte ou un code à saisir sur le clavier situé sous l'afficheur LCD.

Si la carte ou le code ne sont pas valides l'accès au parking est donc refusé 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 s'affichera demandant au client de quitter la boucle.

Les débuts du TP :

 Avant de commencer le TP, nous nous sommes chargés de créer le "repository" sur GitHUB (c'est un service web d'hébergement et de gestion de développement de logiciels), pour protéger et faciliter le développement de notre projet.

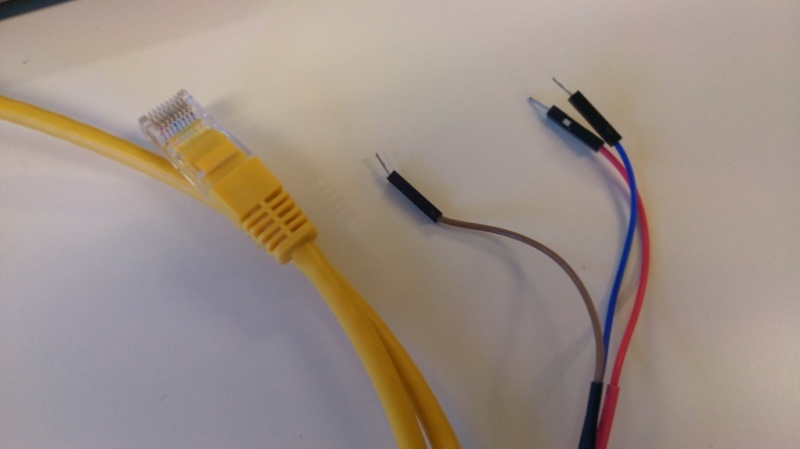
Ensuite nous nous sommes imprégnés du projet en lisant les documentations techniques et le cahier des charges , puis nous nous sommes directement réparti les tâches à fin de gagner du temps en étant organisés.

L'I2C dans notre TP :

Dans ce TP nous utilisons le bus I2C afin de communiquer entre la carte Arduino et le système. Nous avons pour cela dû 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)   
 - les fils blanc/orange et marron pour les données(SDA).

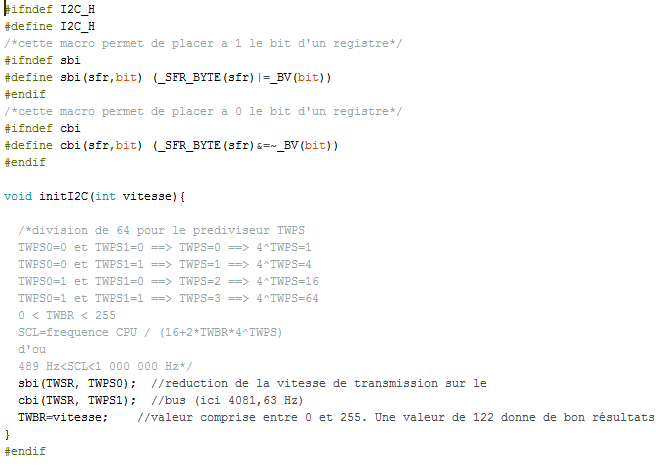
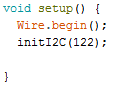
Nous nous retrouvons donc avec les 3 fils typiques de l'i2C qui sont donc le SDA (Serial DAta), le SCL (Sérial CLock), et le GND (GrouND). Le Sérial DAta permet de communiquer les données, ces dernières sont synchronisées via les pulsations d'horloge envoyées par le Sérial CLock. La synchronisation des données est importante elle permet une bonne compréhension entre les deux systèmes.

 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 ce même TP et donc cela demande 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 pouvait pas communiquer à plusieurs en même temps sur le système.

 Par la suite une pieuvre a été installée, elle permet de mettre en relation direct avec le port de la barrière un port choisi pour chaque groupe. Ce qui nous évita de faire des aller- retours à la baie de brassage pour remettre les port en communication .

L'inconvénient c'est que cela demande 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 système.

 Une fois que nous avions installé la pieuvre nous nous sommes rendu compte que le système de la barrières n'exécutait plus correctement les programmes qui auparavant fonctionnaient parfaitement sans la pieuvre.

TWBR

Nous avons donc réfléchis beaucoup de temps avant de nous rendre compte que la pieuvre provoquait des perturbations magnétiques dans le fils qui communiquent avec la barrière et que la distance qui nous sépare de la barrière en câblage était à présent aussi à prendre en compte.

Pour palier à ce problème nous avons donc réduit la vitesse de transmission via la bibliothèque i2C dans notre programme. En passant le 2ème bit de division à 1 nous avons nettement réduit la vitesse de transmission du CPU de la carte Arduino, ce qui a, par conséquent, réduit l'effet de la distance et des perturbations magnétiques sur les données transmise.

N'ayant plus de problème de communication nous avons pu continuer notre TP.

Pour notre part nous avons passé TPWS1 à 1 (**cbi** devient **sbi**) pour atteindre les 1023Hz avec le SCL.

Communiquer avec la barrière :

Nous avons commencé la communication avec la barrière, en tentant de simplement la faire s'ouvrir puis se fermer. Pour cela nous avons utilisé la bibliothèques "Wire.h" qui est intégrée à Arduino, nous n'avions plus qu'à l'inclure et à utiliser ses méthodes.

// Photos du programme de com barrière