SN - Epreuve de projet Enseignement spécifique à la spécialité

Session 2016

Etablissement : Lycée Alphonse BENOIT
L'ISLE-SUR-LA-SORGUE (84)

Système numérique

DOSSIER

Projet : RPI URBACO

Groupe du projet	
Nombre d'étudiant : 4	
Étudiant N°1	Thomas VINCENT
Étudiant N°2	Alexandre CARTEGNIE
Étudiant N°3	Matéo BLETTERY
Étudiant N°4	Damien GALLEGO

REMERCIEMENTS

J'adresse mes remerciements aux personnes qui nous ont aidées dans la réalisation de ce projet.

En premier lieu, nous remercions M.Hortolland, M.Defrance, M.Chastel, professeurs au lycée Alphonse Benoît. Il nous ont guidé dans notre travail et nous ont aidés à trouver des solutions pour avancer.

Nous remercions aussi L'entreprise URBACO, plus particulièrement M.ROVAI et M.BUSSI qui nous ont aidés en nous fournissant des données précises sur le marché des bornes et aussi en nous donnant un avis professionnel.

Table des matières

1 Evaluation final

Tâche		Compétence		Critères d'évaluation	Etudiant			
					N°1 page	N°2 page	N°3 page	N°4 page
T7.2	Produire un prototype logiciel et/ou matériel. A/P	C3.9	Valider une fonction du système à partir d'une maquette réelle.	• Le prototype est fonctionnel. • Le dossier de conception et de fabrication est rédigé. • Le dossier de recette est rédigé.	135	73 82	51 57	100 109
		C4.2	Adapter et/ou configurer un matériel.					
		C4.3	Adapter et/ou configurer une structure logicielle.					
		C4.4	Développer un module logiciel.					
		C4.6	Produire les documents de fabrication d'un sous-ensemble.					
		C4.7	Documenter une réalisation matérielle /logicielle.					
T7.3	Valider le prototype. A	C3.5	Contribuer à la définition des éléments de recette au regard des contraintes du cahier des charges.	• Le compte rendu des tests est mis en corrélation avec le cahier de recette.			43 47	107
		C4.5	Tester et valider un module logiciel et matériel.					
T8.1	Définir une organisation ou un processus de maintenance préventive	C2.1	Maintenir les informations.	• Le dossier de maintenance préventive est adapté au contrat commercial.	21			
T8.2	Définir une organisation ou un processus de maintenance curative.	C2.1	Maintenir les informations.	• Le document définissant la procédure et les moyens d'intervention et de suivi en cas de défaillance du produit est établi.	19			

T9.1	Finaliser le cahier de recette.	C3.1	Analyser un cahier des charges.	• Le cahier de recette est rédigé.	135	82	57	109
		C3.5	Contribuer à la définition des éléments de recette au regard des contraintes du cahier des charges.					
		C4.5	Tester et valider un module logiciel et matériel.					
T9.2	Installer un système ou un service.	C2.5	Travailler en équipe.	• Le système ou le service est en production. • Les documentations client (mise en route, exploitation et maintenance) sont finalisées.	22			
T10.3	Exécuter et/ou planifier les Tâches professionnelles de MCO.	C2.3	Organiser et/ou respecter la planification d'un projet.	• Un mode de contournement est mis en place. • Le temps de rétablissement en mode nominal est le plus rapide possible. • Un rapport d'intervention est rédigé. • Le *roadbook* est enrichi.	16			
T10.4	Proposer des solutions d'amélioration du système ou du service.	C3.6	Recenser les solutions existantes répondant au cahier des charges.	• L'efficience du système est améliorée. • Le taux de pannes ou d'erreurs est diminué. • Des correctifs sont proposés. •	*			
T11.3	Assurer la formation du client.	C2.2	Formaliser l'expression d'un besoin.	• Le client est capable de ... (en fonction du cahier des charges). • Le client (document d'évaluation de la formation et du formateur) est • satisfait.	*			
		C2.5	Travailler en équipe.					
		C2.3	Organiser et/ou respecter la planification d'un projet.					
		C2.5	Travailler en équipe.					

T12.1	Organiser le travail de l'équipe.	C2.3	Organiser et/ou respecter la planification d'un projet.	• Les compétences humaines sont en adéquation avec les Tâches professionnelles et les objectifs. • Le bilan des actions menées est mis à jour. • Les objectifs l'équipe sont définis.	*			
		C2.4	Assumer le rôle total ou partiel de chef de projet.					
		C2.5	Travailler en équipe.					
T12.2	Animer une équipe.	C2.1	Maintenir les informations.	• Les délais du projet sont respectés. • Les problèmes et les conflits sont gérés. • Les comptes rendus (pour les revues de projet) sont rédigés et présentés.	32	60	84	117F
		C2.3	Organiser et/ou respecter la planification d'un projet.					
		C2.5	Travailler en équipe.					

2 PARTIE COMMUNE

Aujourd'hui, dans la plupart des cas, stationner en centre-ville nécessite de trouver la disponibilité d'une place de parking pour une durée limitée. Une fois stationné, l'usager doit alors s'acquitter du paiement du stationnement grâce à l'horodateur automatique qui délivrera le ticket spécifiant l'heure maximale de stationnement.

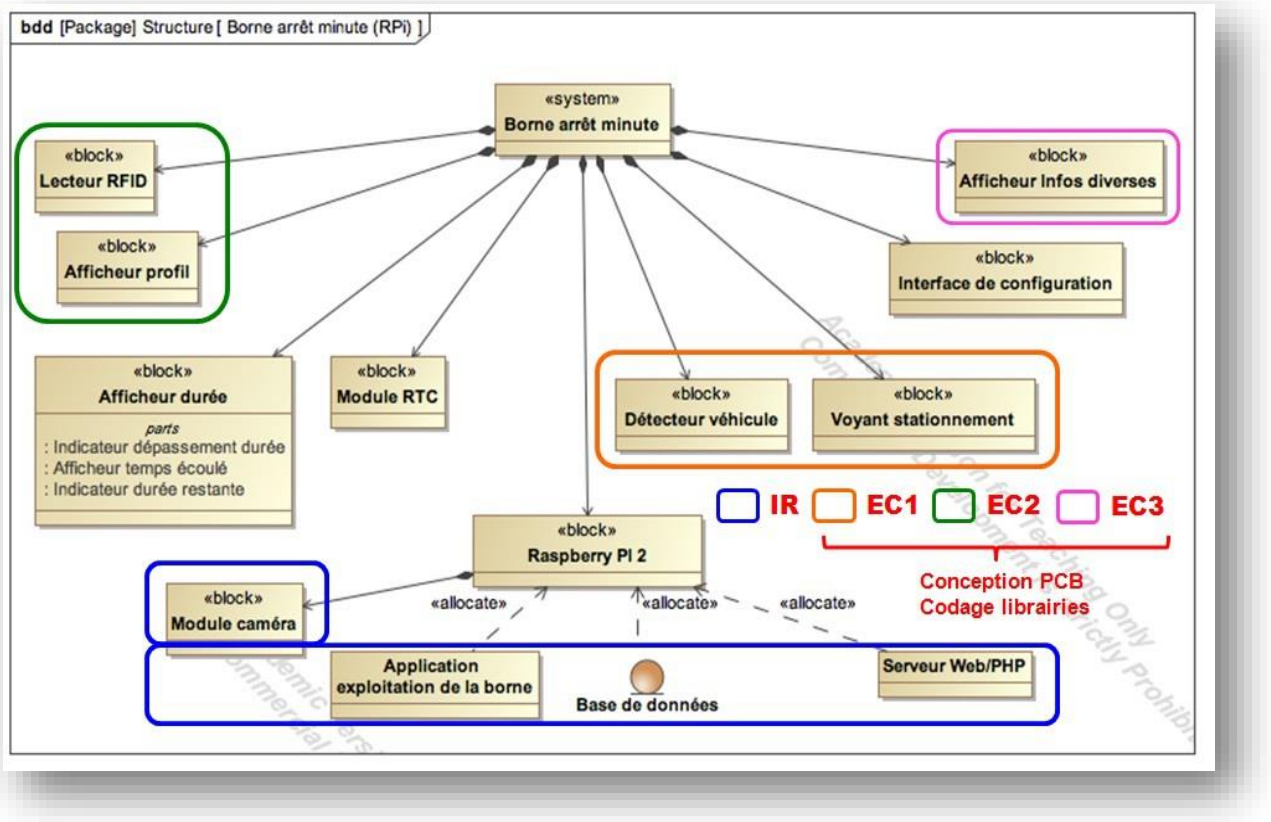
D'autre dispositifs, gratuits sur une période, obligent que l'automobiliste dispose du « disque de stationnement » où sera indiqué l'heure d'arrivée sur la place de parking. Les dispositifs « arrêt-minute » permettent de restreindre en temps le stationnement sur certaines places ciblées à proximité de commerces de services rapides tels que les pharmacies, boulangeries, tabacs... tout en indiquant à l'automobiliste s'il est ou non en dépassement de temps autorisé.

URBACO désire développer un système "Arrêt Minute" spécifique, et suivant ces propre caractéristiques.

Aujourd'hui, ce produit existe dans le catalogue URBACO. Ce type de matériel est couteux, et les temps de câblage sont importants.

Afin de réduire ces coûts et faciliter la mise en œuvre, URBACO a décidé d'opter pour une carte électronique dédiée.

2.1 Répartition du travail



Chaque rectangle désigne une tache spécifique à chaque étudiant concerné par le projet URBACO RPI.

Objectifs :

Etudiant IR : Concevoir/Coder/Tester le Firmware d'exploitation de la borne.

Concevoir/Coder/Tester un site Web pour supervision par client léger.

Etudiant EC1 : Détecter un véhicule et mettre en service le voyant de stationnement.

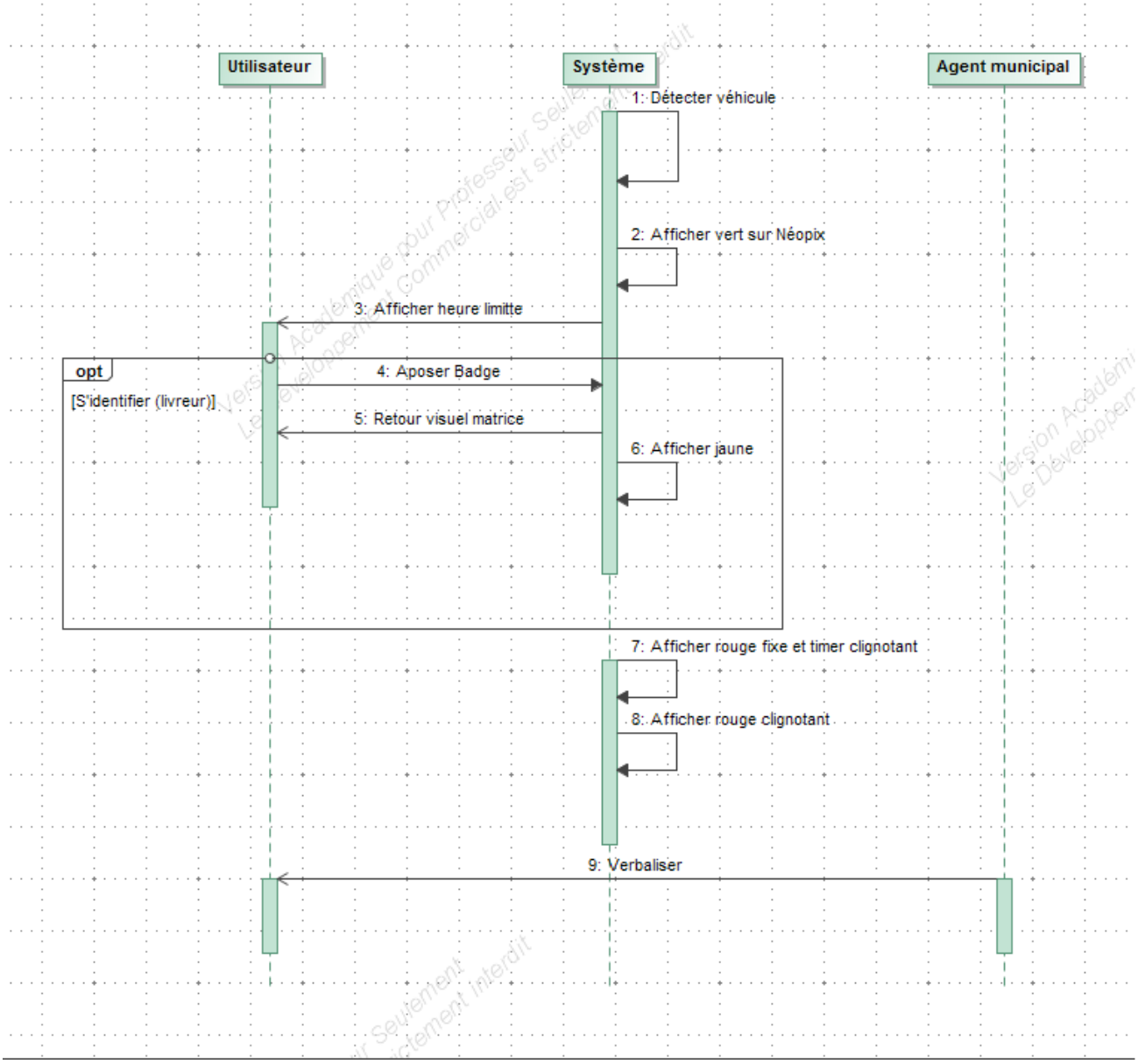
Etudiant EC2 : A partir de la présentation d'un badge RFID, afficher le profil de l'utilisateur

Etudiant EC3 : Afficher des informations de type« informations municipales ».

Intégrer les différents constituant matériels et logiciels (partie EC) du projet.

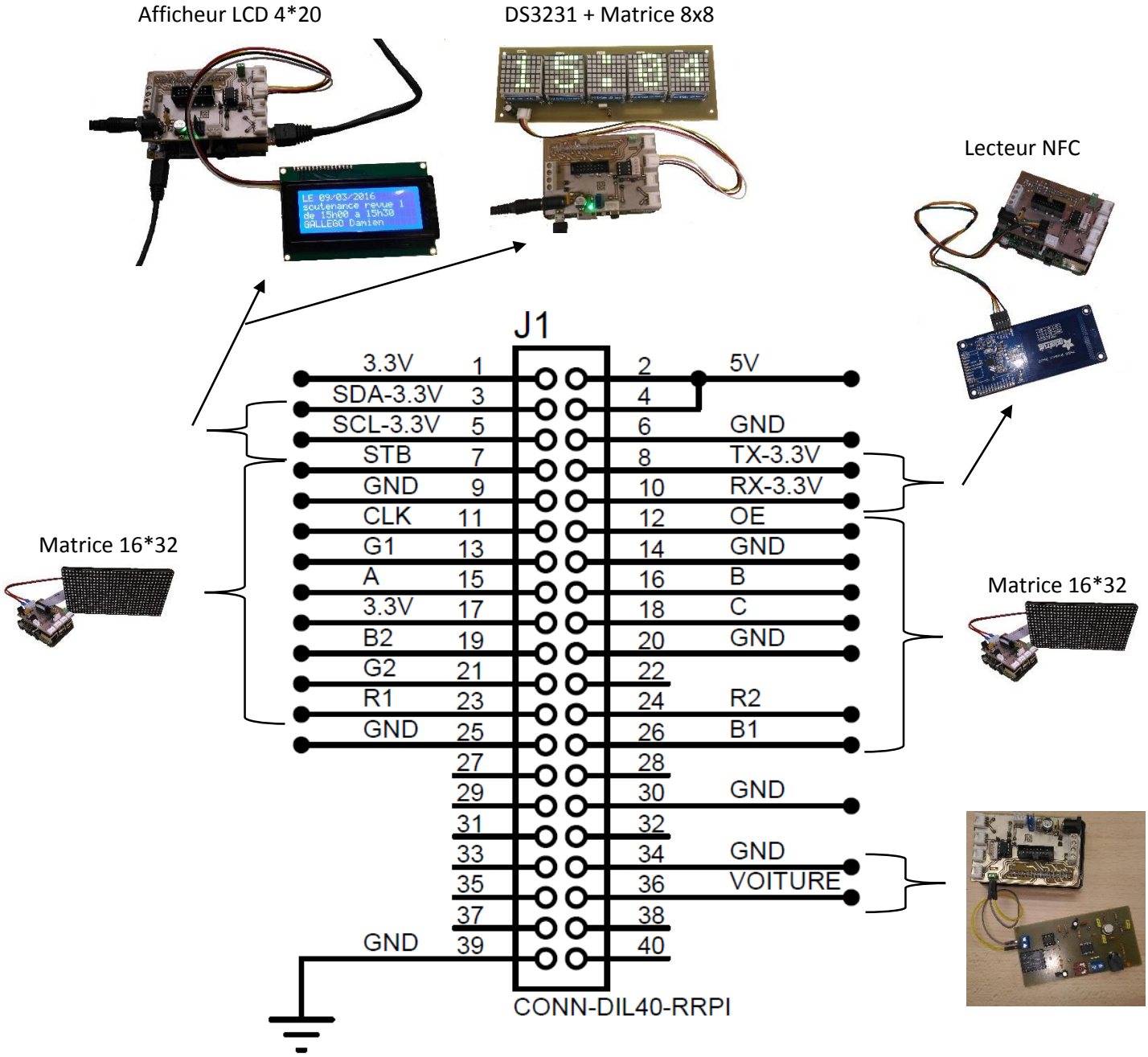
2.2 Diagramme SYSML

Le diagramme suivant représente le scenario typique de la borne en fonctionnement.



2.3 Occupation du GPIO

Voici le schéma d'occupation du GPIO, plus précisément l'occupation des broches de la Raspberry.
Au total nous utilisons 28 broches I/O sur les x disponible.

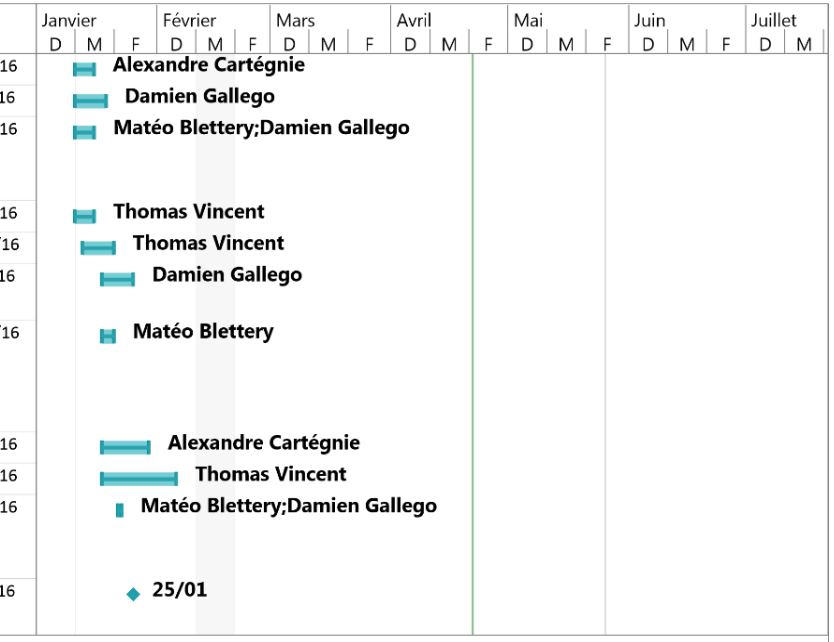


3 Fiche de réunion

Feuille de réunion		***Réf.*** FMCxxxxxxx
Projet	***Réalisateur (s)***	***Date***
URBACO Raspberry Pi		17.02.2016
Phase		
Réunion permettant de connaitre l'avancement du projet et les problèmes rencontré.		
Détail de la réunion		***Etudiant***
		GALLEGO
		Etat
Vérification de fonctionnement des led neopixel sur raspberry pi		OK
Test afficheur I2C 4*20 avec arduino		OK
Test carte rfid sur raspberry		OK
Test lib python led ws2812b		OK
Creation lib pour led ws2812b sur RPI avec wiringPi (ça fonctionne par intermittence)		BUG
		BLETTERY
		Etat
Test carte rfid sur Arduino		OK
Test carte RFID sur raspberry		OK
Test matrice led 16*20 sur raspberry		OK
		CARTEGNIE
		Etat
Test led neopixel Raspberry		OK
Test de la carte de détection (fréquence de détection trop haute par rapport a la réalité des calculs)		BUG puis OK
		VICENT
		Etat
Chaine cross compilation QT PC → RPI		BUG
Suivi tuto, Site PHP Architecture MVC et intégration SILEX et Moteur de Template		EN COURS
Installation lighttpd, MySQL, module PHP		OK

4 Diagramme de Gantt

N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin
1		Essaie Néopixel	3 jours	Lun 11/01/16	Ven 15/01/16
2		Afficheur I2C 4x20 test	3,57 jours	Lun 11/01/16	Lun 18/01/16
3		Recherche de doc,librairie,câblage et mise en fonctionnement RFID	3 jours	Lun 11/01/16	Ven 15/01/16
4		Installation de Jessie Lite	3 jours	Lun 11/01/16	Ven 15/01/16
5		Recherche de Tutoriel sur a	3,57 jours	Mer 13/01/16	Mer 20/01/16
6		Création librairie afficheur I2C	3,57 jours	Lun 18/01/16	Lun 25/01/16
7		Recherche de doc,librairie,câblage et mise en fonctionnement Afficheur 16x32	1,86 jours	Lun 18/01/16	Mer 20/01/16
8		Carte de détection des véh	5,29 jours	Lun 18/01/16	Ven 29/01/16
9		Cross-Compiation	7,57 jours	Lun 18/01/16	Ven 05/02/16
10		Recherche de doc,librairie,câblage et premier test de simultanéité des	1 jour	Ven 22/01/16	Ven 22/01/16
11		test de modification librairie matrice (afficheur)	0 jour	Lun 25/01/16	Lun 25/01/16



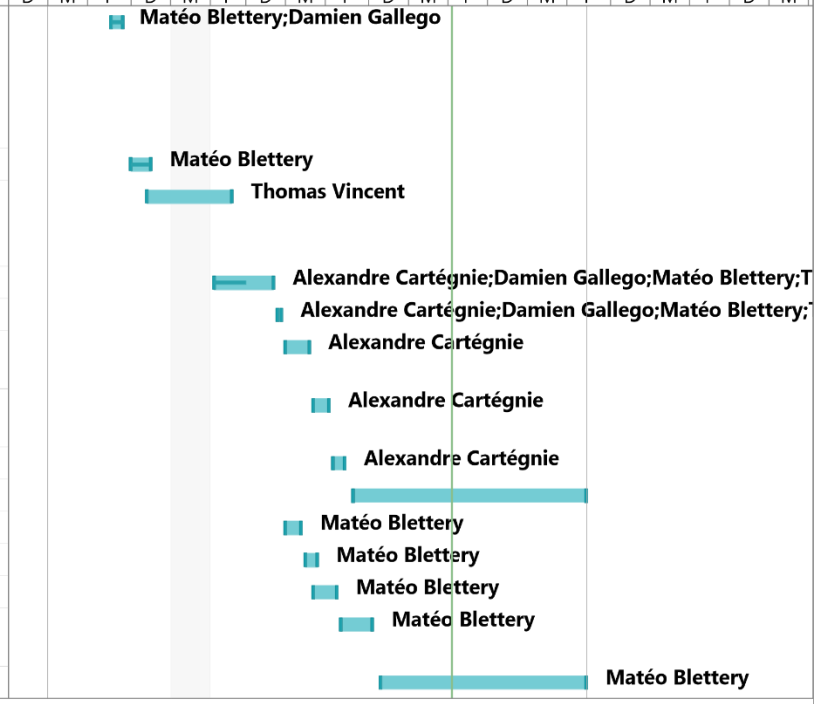
Projet : Gantt Urbaco
Date : Ven 22/04/16

Tâche	Récapitulatif inactif	Tâches externes
Fractionnement	Tâche manuelle	Jalons externes
Jalon	Durée uniquement	Échéance
Récapitulative	Report récapitulatif manuel	Avancement
Récapitulatif du projet	Récapitulatif manuel	Progression manuelle
Tâche inactive	Début uniquement	
Jalon inactif	Fin uniquement	

Page 1

Epreuve E6.2 Projet URBACO Raspberry PI 2016

N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin
12		Recherche de doc,librairie,câblage et mise en fonctionnement lecteur NFC + 2eme test de simultanéité	2,43 jours	Mer 27/01/16	Ven 29/01/16
13		Séance QT	3 jours	Lun 01/02/16	Ven 05/02/16
14		Tutoriel Openclassrooms Architecture php professionnelle	2 jours	Ven 05/02/16	Ven 26/02/16
15		Revue de projet	5,86 jours	Lun 22/02/16	Lun 07/03/16
16		Revue de Projet oral	1,29 jours	Mer 09/03/16	Mer 09/03/16
17		Test Néopix avec AT Tiny	2,14 jours	Ven 11/03/16	Mer 16/03/16
18		Création schéma de routage	1,57 jours	Ven 18/03/16	Lun 21/03/16
19		Réalisation d'une carte	2,43 jours	Mer 23/03/16	Ven 25/03/16
20		Assemblage finale	15 jours	Lun 28/03/16	Mer 25/05/16
21		Finalisation application QT	1,57 jours	Ven 11/03/16	Lun 14/03/16
22		Création interface carte	2,43 jours	Mer 16/03/16	Ven 18/03/16
23		Création librairie	2,14 jours	Ven 18/03/16	Mer 23/03/16
24		Création schéma Solidwork	3,29 jours	Ven 25/03/16	Ven 01/04/16
25		Assemblage Finale	13,29 jours	Lun 04/04/16	Mer 25/05/16



Projet : Gantt Urbaco
Date : Ven 22/04/16

Tâche	Récapitulatif inactif	Tâches externes	
Fractionnement	Tâche manuelle	Jalons externes	
Jalon	Durée uniquement	Échéance	
Récapitulative	Report récapitulatif manuel	Avancement	
Récapitulatif du projet	Récapitulatif manuel	Progression manuelle	
Tâche inactive	Début uniquement		
Jalon inactif	Fin uniquement		

Page 2

5 Fiche de consommation de la borne

Composant (qté)		**Consommation Min* (mA)**	**Consommation Max** (mA)**
NEOPIXELS (61)		130	1000
Carte détection véhicule		20	50
*Matrice 16*32*		92	660
Lecteur NFC		73	83
Matrice 8x8 (5)	Vert ou Rouge	0.002	165
	Orange (Vert + Rouge)	0.002	330
Shield	Seule	10.6	***
ATiny85		***	10
Afficheur LCD 4x20		12	42.2
RPI		300	500
Total		1132	2800

*Consommation min basé sur les modules éteint mais les drivers alimenter

**Consommation maximale basé sur le fonctionnement normale de la borne arrêt minute

6 Fiche de maintenance curative

Anomalie	Cause possible	Actions correctives	Validé/ défaillant
Néopixels ne s'allument pas	Mauvais raccordement	Connexion des fils (vérifier la résistance interne des leds)	
	Défaillance de l'ATtiny85	Renvoyer au SAV (problème logiciel)	
Le véhicule n'est pas détecté	Aucun véhicule ne stationne	Placer un véhicule et vérifier s'il est détecté	
	Panne de l'alimentation électrique	Vérifier le transformateur secteur et le raccordement électrique de la carte	
	Mauvaise connexion de la spire de détection	Replacer les fils dans leur bornier	
	Le seuil de comparaison est défaillant	Procéder au réglage du système grâce au potentiomètre.	
L'afficheur ne s'allume pas	Afficheur non connecté	Connecter l'afficheur	
	Base de données vide	Remplire la base de données pour essai	
Toute la borne ne fonctionne pas	Vérifier si la LED verte est éclairé sur la carte mère	Changer l'alimentation de la carte mère	
	Vérifier si la LED rouge de la RPI est éclairé	Changer l'alimentation de la RPI	
	Si après avoir changé l'alimentation de la RPI rien ne fonctionne	Changer la RPI	

La lecture du badge ne fonctionne pas	Vérifier si la led rouge est bien allumé sur le lecteur PN532	Changer l'alimentation
	Mauvais câblage	Vérifier le câblage du lecteur PN532 nfc, modifier s'il n'est pas correct
	Badge dysfonctionnel	Utiliser un badge administrateur pour vérifier la lecture
Le profil utilisateur ne s'affiche pas	Panne de l'alimentation électrique	Changer l'alimentation
	Mauvais câblage	Vérifier le câblage, le modifier s'il n'est pas correct
	Problème avec le lecteur PN532 nfc, Profil non récupéré car badge non lu	Se référer aux pannes potentielles du lecteur PN532 nfc, ci-dessus.

7 Fiche de maintenance préventive

Les valeurs indiquées dans le tableau ci-dessous sont à titre indicatif, il ne s'agit en aucun cas de référence

Composant	Durée de vie moyenne	Date de mise en service	Date de remplacement conseillée	Date de prochain remplacement
NEOPIXEL	100 000 Heures (10 ans)	18/07/2016	Juillet 2026	
ATtiny85	5 ans	18/07/2016	Juillet 2021	
Carte de détection véhicule	10 ans	18/07/2016	Juillet 2026	
Raspberry Pi2	1 an*	18/07/2016	Juillet 2017	
Bloc alimentation RPI	1an*	18/07/2016	Juillet 2017	
Afficheur LED RGB 16*32	10ans	18/07/2016	Juillet 2026	
Shield**	A vie	***	***	
Afficheur LCD 4*20	6 ans	18/07/2016	Juillet 2022	
Lecteur NFC	A vie	***	***	

8 Fiche de mise en service

Pour pouvoir profiter des fonctionnalités de la borne Arrêt Minute URBACO Raspberry Pi, Veuillez suivre les instructions ci-dessous lors de la première mise en service.

-Alimenter la borne au secteur 230V-50Hz

-Se connecter au wifi ***« Urbaco Rpi »***

-Se connecter au site WEB en utilisant login et Mot de passe fournis

-Accéder au menu ***« paramétrage »***

-Cliquer sur le bouton ***« Initialisation de la borne »***

La borne est à présent fonctionnelle et prête à être utilisée.

9 Journal de bord commun au projet URBACO RPI

Date	*Objet*
13.01.2016	▪ **GALLEGO** Test LED NEOPIXEL + programmation en python pour les NEOPIXEL Pris connaissance afficheur 4x20 I2C ▪ **BLETTERY** Recherche pour le module rfid test module rfid arduino uno commencer a installé les lib pour rfid ▪ **CARTEGNIE** Recherche pour neopixel et instalation des libs ▪ **VINCENT** Installation de Jessie lite recherche de tutoriel sur adminLTE,bootstrap 3,Jquery
14.01.2016	▪ **GALLEGO** Test de l'afficheur I2C DFROBOT sur Arduino = ok Chercher et trouver lib pour afficheur sur RPI en python Test de l'afficheur sur RPI en 3.3V fonctionnement = non Test de l'afficheur sur RPI en 5v avec level shifter de Castello fonctionnement = oui
15.01.2016	▪ **GALLEGO** Etude de la trame envoyer par le maitre I2C a l'esclave après beaucoup d'heure et de recherche d'aide des profs un résultat et sortie. L'adresse du composant est envoyée en claire 0x20 les commandes passe par une fonction qui fait un & 0xF0 avec le code commande puis reprend le code commande et le décale de 4 bits et fait un & 0xF0 se qui nous donne 2 mots de 8 bits en suis le premier mot passe par une autre fonction un | LCD_BACKLIGHT cette valeur vos soit 0x00 si off et 0x08 si allumer donc aux valeurs précédentes on ajoute 8. Les codes commande une fois décrypter sont retrouver identique Les caractères ASCII une fois décrypter on retrouve le code HEX mais avec +1 ce qui pour le moment mes inconnu. ▪ **BLETTERY**

	essais RFID sur rasperry avec la librairie “rfid-master” (librairie commune à arduino et rasperry). comme sur arduino, certaines broches sur rpi sont dédié a la spi. ▪ **CARTEGNIE** fin d’installation libs pour neopix, création de 3 programmes distincts pour rouge vert et eteint. ▪ **VINCENT** Suivi du tutoriel https://wiki.qt.io/RaspberryPi2EGLFS pour la cross compilation Erreur au moment du make (étape 7)
16.01.2016	▪ **GALLEGO** Recherche pour le capteur RFID rc522 Après des échecs avec des librairies en C j’ai trouvé une librairie en python qui fonctionne avec l'exemple fournie. Des modifications a ce code son a apporter pour pouvoir écrire aussi sur les badges a moi que l'écriture des badges se face seulement quand le détenteur du badge s’inscrit pour en avoir un. Sur le capteur RFID rc522 se trouve une broche d'interruption. des essais sont à prévoir pour vérifier si cette broche s'active ou non quand un badge passe devant car si cette broche s’active ou se désactive quand un badge passe devant on peut l’utiliser pour dire à la rpi que un badge veut être badge ce qui nous permettrai à ce moment-là d'appeler le script python. Il suffirait que dans notre programme final ont détecté un front descendant ou un front montant.
17.01.2016	▪ **GALLEGO** Recherches de différents connecteurs pour le shield car plusieurs bus de donnée.
18.01.2016	▪ **GALLEGO** - Décodage de trame I2C afficheur 4*20 - Explication de Mr HOROLAND sur le décodage de trame ▪ **BLETTERY** Essais RFID avec librairie “pi-rc522-master” fonctionne sur Raspberry de Damien Gallego mais sur aucunes autres. ▪ **VINCENT** installation d’une chaine de cross -compilation QT : 5% Recherche de solution pour le problème du tutoriel étape 7. Problème réseaux dans la salle
20.01.2016	▪ **GALLEGO** Début de la création d’une librairie en C++ POO pour l’utilisation de l’afficheur 4*20

	▪ **BLETTERY** Essais RFID problemes encore inconnu Essais sur Matrice, cablage (ok), Librairie trouvée. ▪ **CARTEGNIE** Mise en place et essai de la carte de détection véhicule, aide sur le câblage matrice
22.01.2016	▪ **GALLEGO** Suite de la création de la librairie pour l'afficheur I2C 4*20 avec wiringPiI2C ▪ **CARTEGNIE** Suite et fin des tests sur la carte de détection essayent de comprendre les calculs • BLETTERY Essais et analyse d'une trame pour la carte RFID
25.01.2016	▪ GALLEGO Fin de la création de la librairie et essais, essai afficheur ne fonctionne pas comme il faut bug au niveau de l'affichage. ▪ VINCENT Mise en place de la chaîne de cross-compilation 25 % • BLETTERY Mise en marche de la matrice essai avec image PNG.
27.01.2016	▪ GALLEGO Recherche du défaut dans ma librairie en croyant que sa viens de mes delay je l'ai augmentés. Essai de nouveau, décevant rien changé. • BLETTERY Test de la matrice et du lecteur RFID, un conflit survient… Orientation du choix vers de la NFC.
29.01.2016	▪ BLETTERY Essai pour modifier les caractéristiques de la librairie pour pouvoir utiliser La RFID sans succès, le choix de la NFC est donc avéré.
01.02.2016	▪ **GALLEGO** BTS Blanc n°1 ▪ CARTEGNIE BTS Blanc n°1 ▪ BLETTERY BTS Blanc n°1 ▪ VINCENT BTS Blanc n°1
03.02.2016	▪ VINCENT

	Mise en place de la chaîne de cross-compilation 60 % • BLETTERY Test de la libraire NFC, celle-ci est opérationnelle UID Badge est récupérable. Recherche sur le câblage d'un récepteur NFC , librairies et câblage du récepteur.
05.02.2016	▪ BLETTERY Création d'une application QT permettant de se déplacer dans l'arborescence de la Rpi.
22.02.2016	▪ GALLEGO Je me suis renseigner sur la fonction delay et il fait mètre un 'int' et pas un 'float' comme j'avais fait j'ai don modifier. Nouvelle essai concluent cette fois sa fonctionne comme il faut. • B LETTERY Récupération de l'UID sur SALEAE . pour la carte nfc
24.02.2016	▪ **GALLEGO** Absent ▪ **BLETTERY** réussi à afficher une image en fonction du code d'un badge ▪ **CARTEGNIE** Mise en place du dossier pour la première revue de projet ▪ **VINCENT** suivi tutoriel OpenClasseroom sur l'architecture PHP pro et résolution de problèmes divers Mise en place de la chaîne de cross-compilation 60 %
26.02.2016	▪ **GALLEGO** Edition du dossier de revue 1 ▪ **BLETTERY** Edition du dossier de revue 1
29.02.2016	▪ **GALLEGO** Edition du dossier de revue 1 ▪ **BLETTERY** Edition du dossier de revue 1
02.03.2016	▪ **GALLEGO** Finalisation de l'édition revue 1 et début d'édition power point revue 1 ▪ **BLETTERY** Finalisation de l'édition revue 1 et début d'édition power point revue 1
04.03.2016	▪ **GALLEGO**

	Assemblage et remise des dossiers de revue 1 ▪ **BLETTERY** Assemblage et remise des dossiers de revue 1 ▪ **CARTEGNIE** Assemblage et remise des dossiers de revue 1 ▪ **VINCENT** Assemblage et remise des dossiers de revue 1
07.03.2016	▪ GALLEGO Finalisation du power point de revue 1 ▪ VINCENT mise en commun des librairies des EC et début du code de l'application finale 5 %
09.03.2016	▪ **GALLEGO** Soutenance revue 1, préparation à l'orale ▪ CARTEGNIE Soutenance revue 1, préparation à l'orale ▪ BLETTERY Soutenance revue 1, préparation à l'orale ▪ VINCENT Soutenance revue 1, préparation à l'orale
11.03.2016	▪ **GALLEGO** Création d'emprunts ISIS et ARES pour le Shield ▪ CARTEGNIE Mise en place de la communication entre 2 cartes Arduino grâce au protocole I2C essai avec néo pixels sur la carte « esclave ». Modélisation du schéma de la carte de détection sur Proteus avec essais de simulation… problèmes de simulation. ▪ BLETTERY Amélioration Application QT
14.03.2016	▪ **GALLEGO** Préparation orale anglais et éco-gestion ▪ CARTEGNIE Mise en place et modification du schéma sur proteus partie oscillateur et Convertisseur F/U Le schéma est simulable sans problèmes. ▪ **VINCENT** Création du site en partenariat avec Corentin 5 %

	▪ BLETTERY Amélioration Application QT
16.03.2016	▪ **GALLEGO** Début du schéma sur ISIS ▪ CARTEGNIE Préparation au routage en traçant un schéma structurel sur proteus. ▪ BLETTERY Amélioration Application QT
18.03.2016	▪ **GALLEGO** Finalisation du schéma sur ISIS ▪ CARTEGNIE Correction du schéma structurel ▪ BLETTERY Recherche si possibilité de modification carte NFC possible
21.03.2016	▪ **GALLEGO** Début de la conception du typon sur ARES ▪ CARTEGNIE Début du routage ▪ BLETTERY Amélioration Application QT
23.03.2016	▪ **GALLEGO** Finalisation du typon sur ARES Envoie des fichiers schéma et typon a Mr HORTOLAND ▪ CARTEGNIE Routage, Soudure de la carte comprenant 61 Néopixels et tests sous arduino. L'ensemble fonctionne parfaitement. ▪ VINCENT Création d'une application de démonstration simulant la borne
25.03.2016	▪ **GALLEGO** Correction du schéma et du typon suite au compte rendu de Mr HORTOLAND ▪ CARTEGNIE Routage de la carte correction des erreurs de routages, apport de modifications ▪ VINCENT Création du site en partenariat avec Corentin 20% ▪ BLETTERY Création d'une borne sous solidworks

28.03.2016	▪ **GALLEGO** Finalisation des corrections du typon sur ARES Finalisation de la présentation du schéma sur ISIS Crée un Word avec le schéma
30.03.2016	▪ **GALLEGO** BTS Blanc n°2 ▪ BLETTERY Création d'une librairie
01.04.2016	▪ **GALLEGO** Souder le reste de ma carte car grande partie fait a la maison Cheker les composant reçus Crée un Word avec entête et pied de page et sommaire auto ▪ CARTEGNIE Fabrication de la carte ▪ VINCENT Création d'un application de démonstration simulant la borne ▪ BLETTERY Librairie terminée
18.04.2016	▪ **GALLEGO** Finalisation de l'évolution de la librairie I2C_Afficheur_4x20 avec le rajout de méthodes « connecxionBDD » et « checkAffichage » permettant d'afficher le texte se trouvant dans la base de donnée. ▪ CARTEGNIE Perçage et soudage des composants premiers tests de la carte ▪ BLETTERY Annimation et modélisation solidworks terminé
20.04.2016	▪ **GALLEGO** Finalisation du dossier de revue 2 ▪ CARTEGNIE Production de documents pour la soutenance n°2 ▪ VINCENT Finalisation de l'application de démonstration

	▪ BLETTERY Dossier Revue n°2
22.04.2016	▪ **GALLEGO** Assemblage et remise des dossiers de revue 2 ▪ **BLETTERY** Assemblage et remise des dossiers de revue 2 ▪ **CARTEGNIE** Assemblage et remise des dossiers de revue 2 ▪ **VINCENT** Assemblage et remise des dossiers de revue 2
27.04.2016	▪ **TOUS** Soutenance revue 2, préparation à l'orale
09.05.2016	▪ **GALLEGO** Orale + écrit anglais ▪ **BLETTERY** Orale + écrit anglais ▪ **CARTEGNIE** Orale + écrit anglais ▪ **VINCENT** Orale + écrit anglais
12.05.2016	▪ **TOUS** Visite de la société URBACO
13.05.2016	▪ **GALLEGO** Matinée banalisée ▪ **CARTEGNIE** Matinée banalisée ▪ **VINCENT** Matinée banalisée ▪ **BLETTERY** Matinée banalisée
16.05.2016	▪ **BLETTERY** **Création de diagrammes**
18.05.2016	▪ **GALLEGO** Construction d'un diagramme de block pour le Shield • BLETTERY Les diagrammes sont terminés
20.05.2016	▪ **GALLEGO** Création avec Alexandre CARTEGNIE d'une librairie permettant de faire fonctionner les LED NEOPIXEL sur l'ATtiny85 via la Raspberry PI.
26.05.2016	▪ **GALLEGO** Remise du dossier de revue final

	▪ **BLETTERY** Remise du dossier de revue final ▪ **CARTEGNIE** Remise du dossier de revue final ▪ **VINCENT** Remise du dossier de revue final

	**Epreuve E6.2 Projet URBACO Raspberry PI 2016**	

Projet URBACO RPI

BTS Systèmes Numériques

Matéo BLETTERY

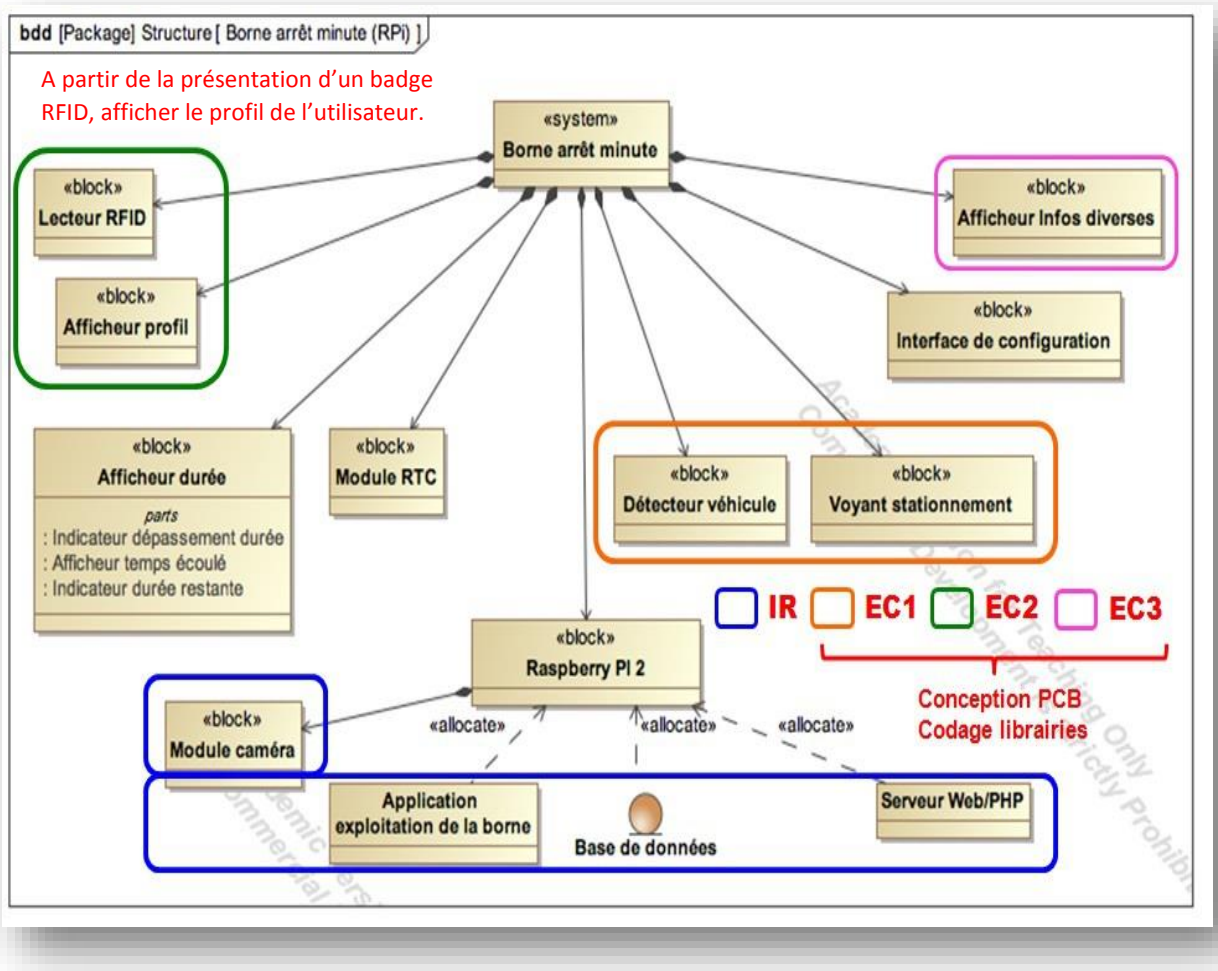
Candidat : Matéo BLETTERY

10 INTRODUCTION

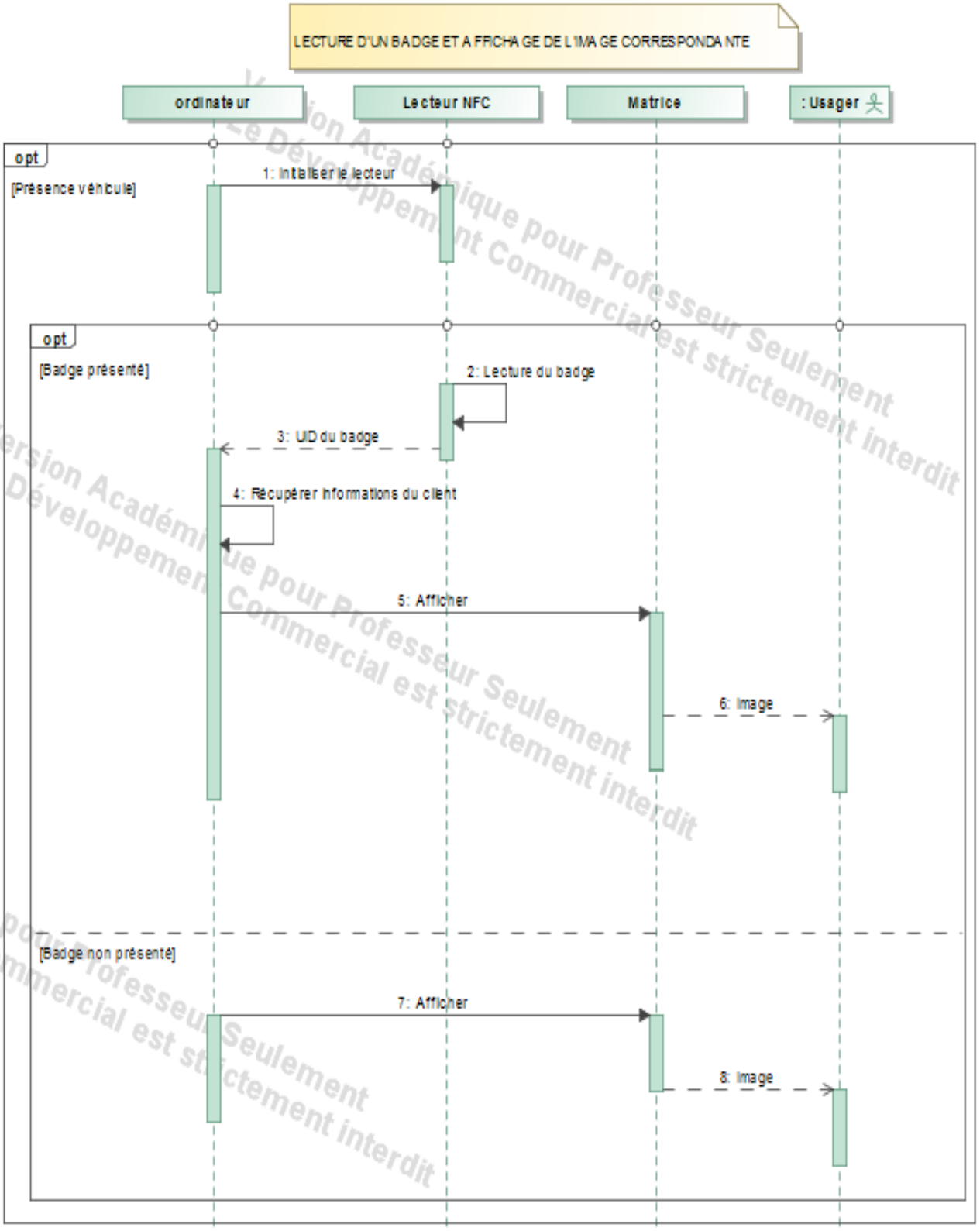
Dans le cadre de ma formation en BTS SN_EC au lycée Alphonse Benoît, je réalise durant ma deuxième année de BTS, un projet industriel. Ceci me permet de mettre en pratique une partie des connaissances théoriques apprises durant ma formation.

Ce projet technique, en collaboration avec Deux autres étudiants de la section (Alexandre Cartegnie, Damien Gallego) et d'un étudiant de la section IR (Thomas Vincent), consiste en la reprise d'un système existant. Le projet a pour but de développer une solution pour un dispositif de stationnement automobile de type « arrêt-minute » pour l'entreprise Urbaco.

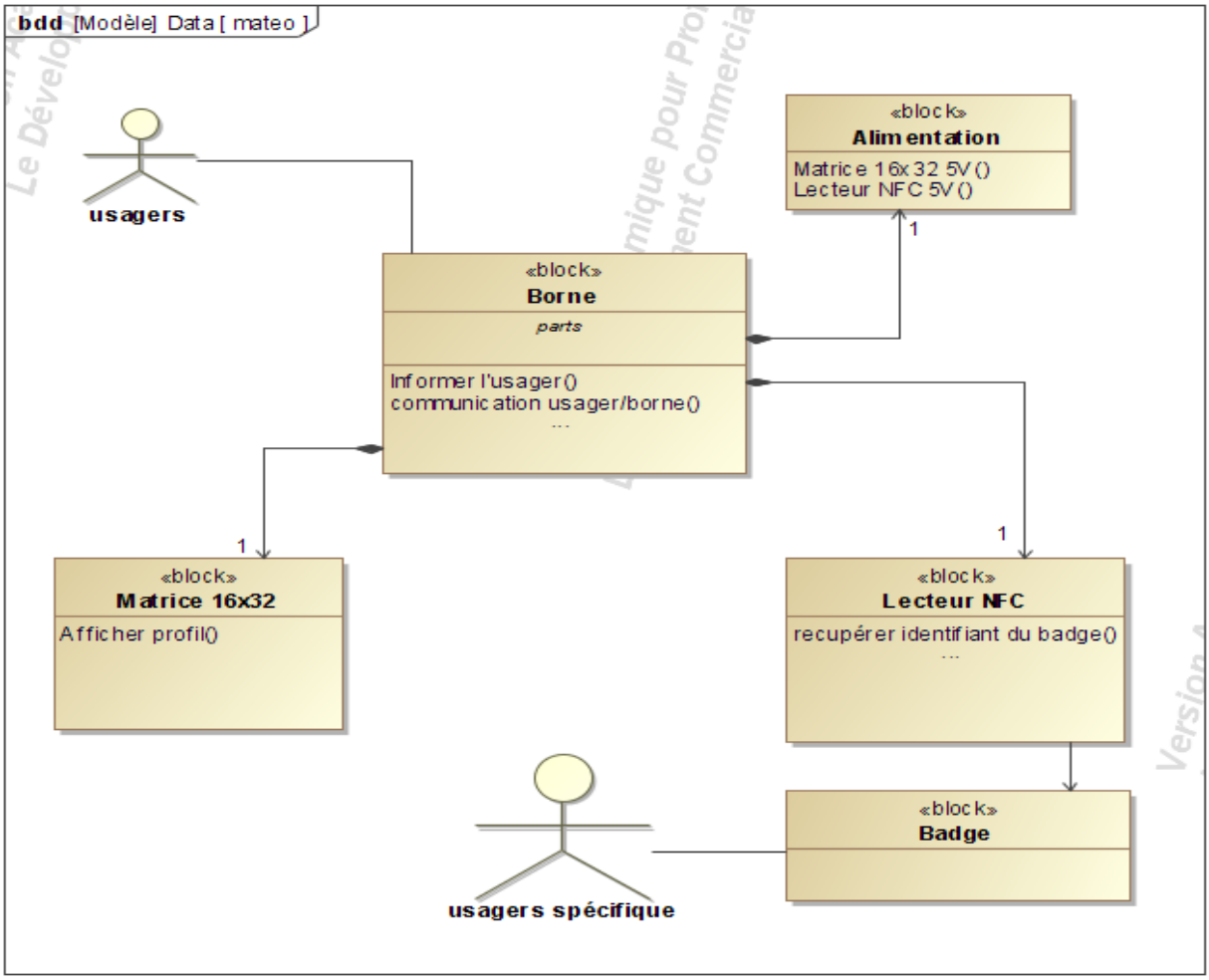
En ce qui me concerne, j'ai été chargé de travailler sur la partie : « A partir de la présentation d'un badge RFID, afficher le profil de l'utilisateur. ».

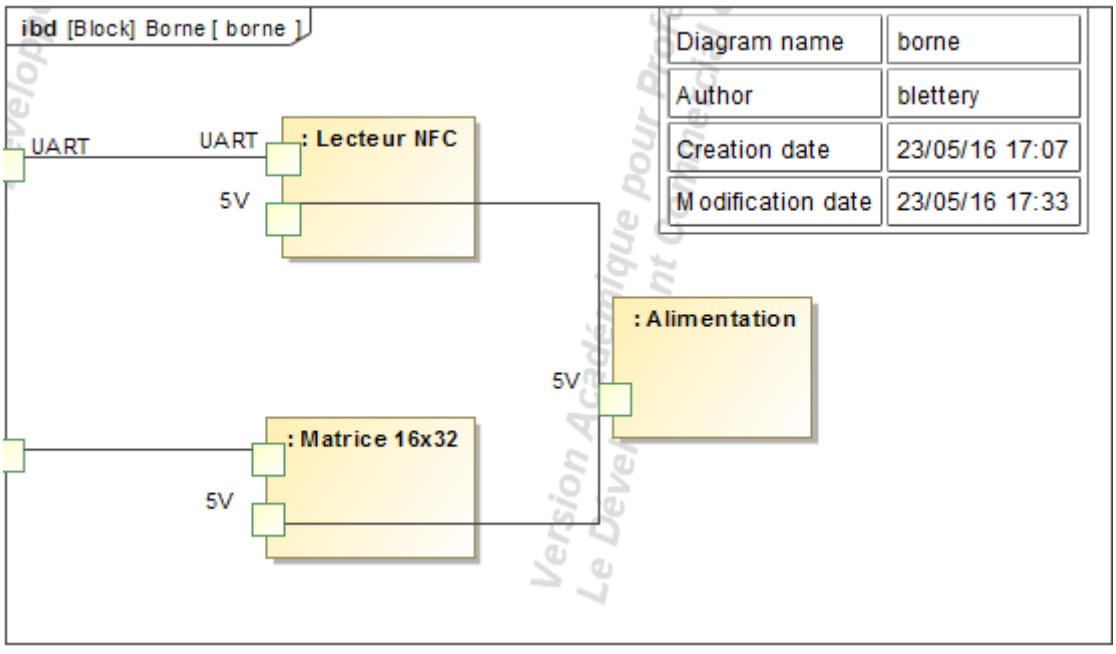


10.1 Diagramme de Séquence



10.2 Diagramme de bloc interne

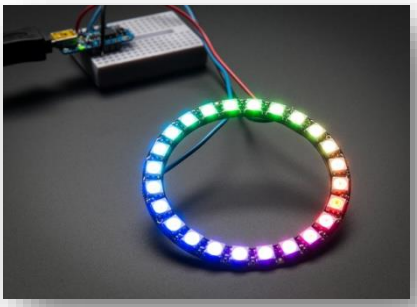


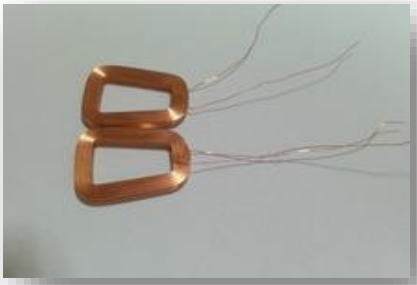


10.3 Deux groupes sur le même projet

Le projet URBACO est l'objet de deux groupes de travail. L'un travaille sur le support Raspberry (Etudiants : Damien Gallego, Alexandre Cartegnie, Thomas Vincent et moi) et l'autre travaille sur le support Arduino yun mini (Etudiants : Amélie Legoy, Pierre Reynaud, Thomas Castello et Corentin Silvestre). La borne étant déjà existante dans la réalité, le but de notre projet est principalement de reproduire cette borne avec des moyens moins couteux. Nous avons à disposition deux supports moins chers que le module siemens déjà utilisé et en commercialisation. Le lycée Alphonse Benoît a donc choisit de recréer cette borne sur les deux supports, cela faisait trop de travail pour un seul groupe. Le prototype final sera l'ensemble du travail des huit étudiants, car certains des composants du projet sont traités en coopération entre les deux groupes.

10.4 Différences et similarités entre les deux groupes

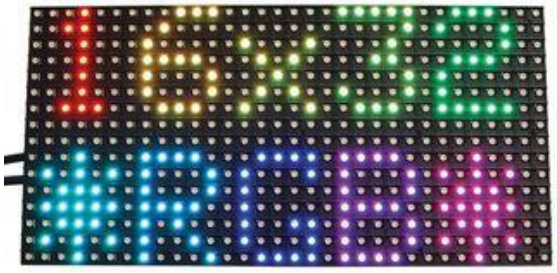




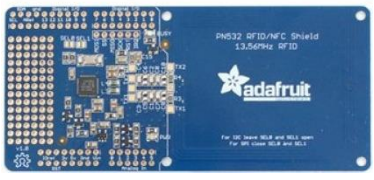
Alexandre et Pierre, ont la même partie, mais sur supports différents. Tous les deux s'occupent de faire fonctionner le cercle à led Neopixels et la détection véhicule.

Damien et Thomas s'occupent tous les deux de faire fonctionner l'afficheur lcd 4x20. Ils s'occupent aussi d'assembler l'ensemble du « Hardware ».

Amélie est seule sur sa partie, cependant elle doit être tout de même fonctionnelle sur les deux supports. Cette partie est le fonctionnement des afficheurs 8x8



Je suis seul sur ma partie aussi, celle–ci est le fonctionnement du lecteur NFC et de la matrice 16x32



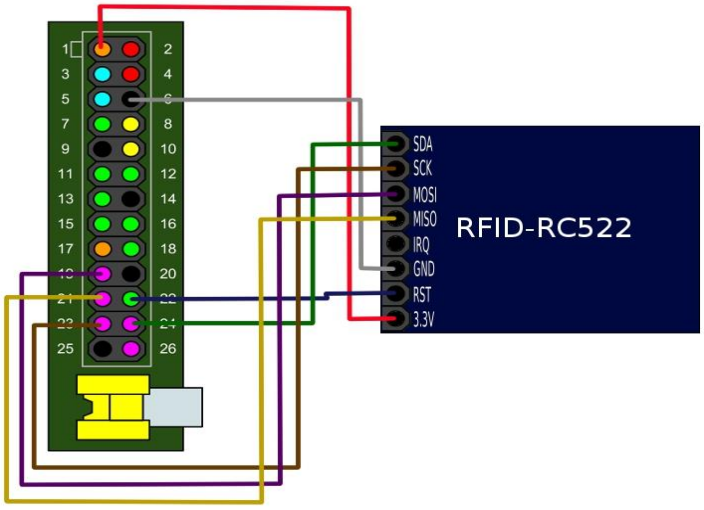
11 TRAVAIL ACCOMPLI

11.1 Mise en fonctionnement du Lecteur RFID

Une grande partie de mon projet est de pouvoir récupérer et utiliser les données des cartes RFID. J'ai donc naturellement commencé par faire fonctionner mon lecteur **RC522** Rfid.

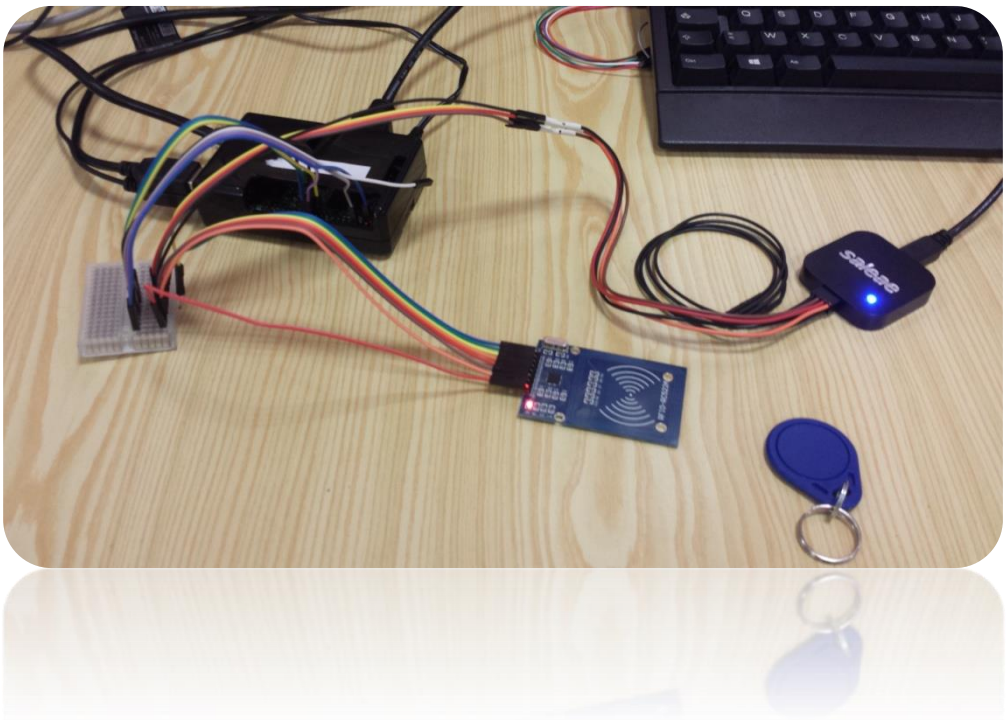
Mon travail a débuté par la recherche d'une librairie permettant de récupérer l'UID (« nom ») des cartes RFID sur Raspberry. Apres quelques recherches, une librairie semblait toute indiquée (RPI-RC522-master). Il suffisait ensuite d'installer les composantes nécessaires et de lancer le programme d'exemple.

Mon travail suivant fût de câbler le lecteur à la Raspberry, pour celui-ci, aucun problème : un schéma de câblage est donné sur internet quand on recherche RPI et RC522.



Avec tout ceci, j'ai réussi à récupérer les données des cartes RFID.

Il me fallait ensuite décoder la trame récupérée par le lecteur pour isoler L'UID des cartes. J'ai pour cela utilisé un analyseur logique « SALEAE » que j'ai branché sur les broches de sortie du lecteur, pour récupérer la trame.



L'UID récupéré est **: UID: # Calling card_auth on UID [85, 157, 46, 0, 230]**

Vous trouverez la trame en annexe.

Voici le décodage **:**

(MOSI = master out, slave in

MISO = master in, slave out)

SPI	MOSI: 8	MISO: 85
SPI	MOSI: 85	MISO: 18
SPI	MOSI: 8	MISO: 157
SPI	MOSI: 157	MISO: 18
SPI	MOSI: 8	MISO: 46
SPI	MOSI: 46	MISO: 18
SPI	MOSI: 8	MISO: 0
SPI	MOSI: 0	MISO: 18
SPI	MOSI: 8	MISO: 230

Mon idée était d'utiliser cet UID pour assigner un rôle aux badges. L'UID est propre à chaque carte, aucune n'aura le même.

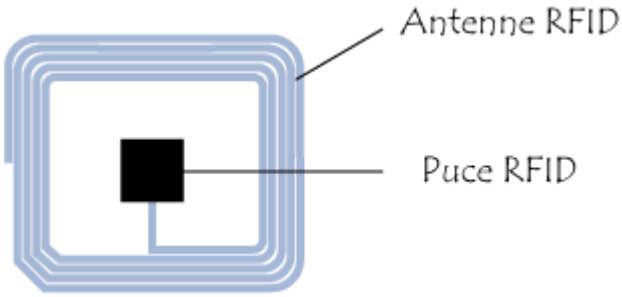
Exemple : Le badge [85, 157, 46, 0, 230] est assigné au rôle « camion », à chaque fois que ce badge sera scanné, le scénario « camion » se lancera.

11.2 Qu'est-ce que la RFID

Un système RFID comprend 3 composants essentiels :

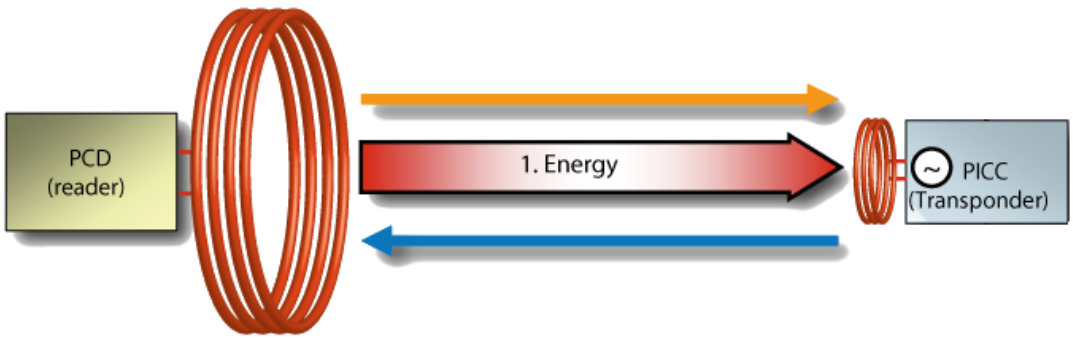
Une puce :

Une antenne :



Un lecteur :

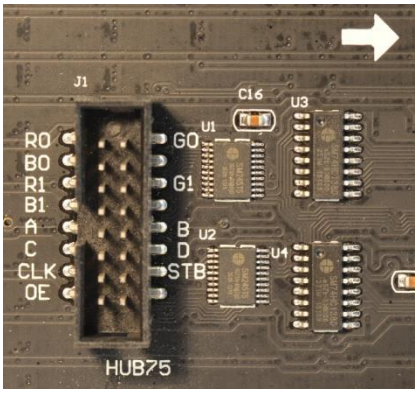
Grâce à son antenne, la puce communique avec le lecteur qui transmet les informations recueillies à un ordinateur où elles sont enregistrées dans une base de données. A l'inverse, l'ordinateur peut enregistrer des informations dans la puce via le lecteur, qui fonctionne également comme un émetteur.



11.3 Mise en fonctionnement de la Matrice 16x32

Une autre partie très importante de mon projet est de pouvoir, en fonction de l'UID des badges, afficher l'image correspondante à un statut client.
En effet, des personnes auront des droits particuliers de stationnement (Handicapés, Livreurs et mêmes commerçants). Pour cela j'ai à disposition une matrice Adafruit 16x32.

De même que pour le lecteur, j'ai commencé par rechercher une librairie me permettant de mettre en fonctionnement ma matrice. La librairie choisie est : RPI-RGB-LED-MATRIX-MASTER. Plus simple que la précédente il suffit de câbler la matrice et de lancer le programme d'exemple, avec quelques indications données par le créateur de cette libraire, on peut piloter une matrice 8x8, 16x16, 16x32 et 32x32.



Connection	Pin	Pin	Connection
-	1	2	-
💧 [3] G1	3	4	-
💧 [3] B1	5	6	GND 😊💥💧
😊💥💧 strobe	7	8	[3] R1 💧
-	9	10	-
😊💥💧 clock	11	12	OE- 😊💥💧
😊 [1] G1	13	14	-
😊💥💧 A	15	16	B 😊💥💧
-	17	18	C 😊💥💧
😊 [1] B2	19	20	-
😊 [1] G2	21	22	D 😊💥💧 (for 32 row matrix, 1:16)
😊 [1] R1	23	24	[1] R2 😊
-	25	26	[1] B1 😊
-	27	28	-
💥 [2] G1	29	30	-
💥 [2] B1	31	32	[2] R1 💥
💥 [2] G2	33	34	-
💥 [2] R2	35	36	[3] G2 💧
💧 [3] R2	37	38	[2] B2 💥
-	39	40	[3] B2 💧

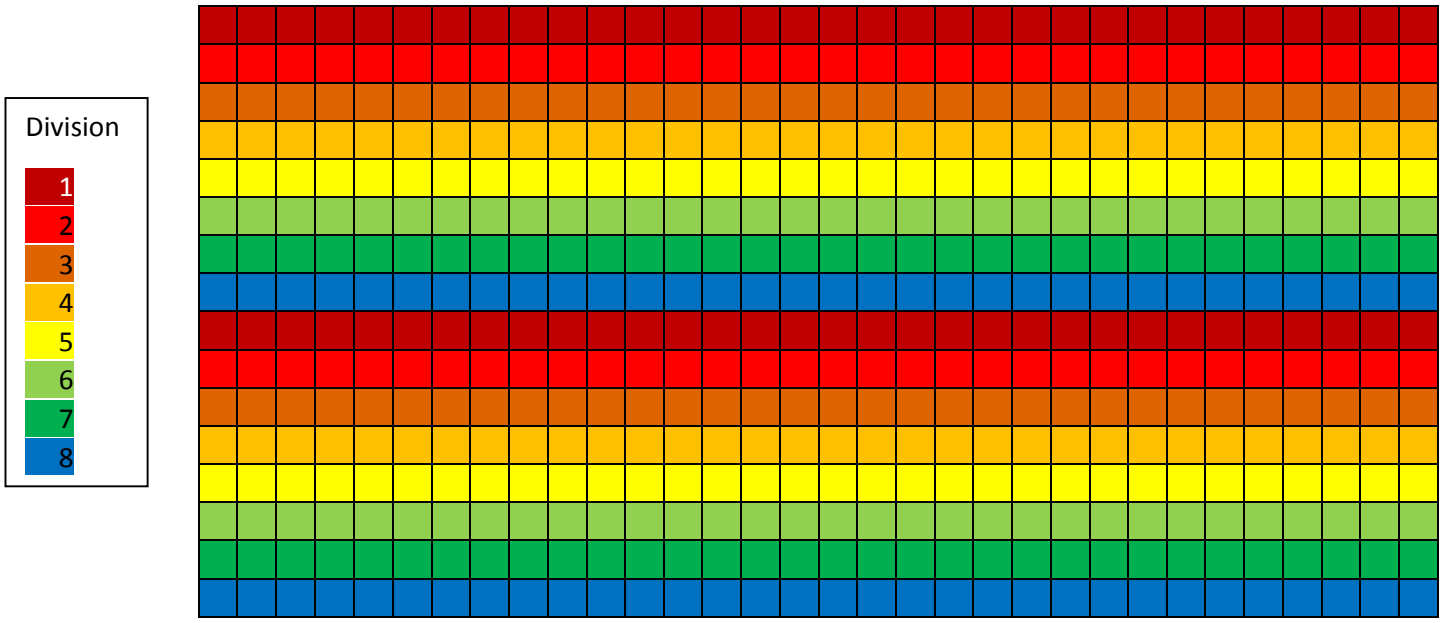
Voici les câblages possibles. Dans notre cas nous allons utiliser le câblage « smiley » car apres quelques tests je me suis aperçu que les autres câblages servent à additionner plusieurs matrices.

Un autre point très important, la librairie permet d'afficher des images sur la Matrice (ex : déssiné sur paint).

11.4 Généralité sur la matrice 16x32 Adafruit

La première chose à noter est qu'il y a 512 LED RGB dans une matrice 16x32. On **ne peut pas allumer les 512 à la fois.** L'une des raisons est que cela exigerait beaucoup de courant. (Schéma consommation du courant en annexe p55)

Au lieu de cela, la matrice est divisée en 8 sections / bandes entrelacées. La première section est «ligne»1 et «ligne» 9 (32 x 2 LED RGB = 64 RGB LED), la seconde est la 2ème et 10ème ligne, etc… jusqu'à la dernière section qui est la 7ème et la 16ème ligne. La raison pour laquelle ils le font de cette façon est que les lignes sont entrelacées (mieux pour le rendu visuel, car le rafraichissement est moins visible).



Sur la matrice en elle-même, nous avons beaucoup de broches :

-Pour commencer, chaque couleur représente deux broches (R0, R1|G0, G1|B0, B1)

-4 broches GND (Ground), pour une meilleure performance

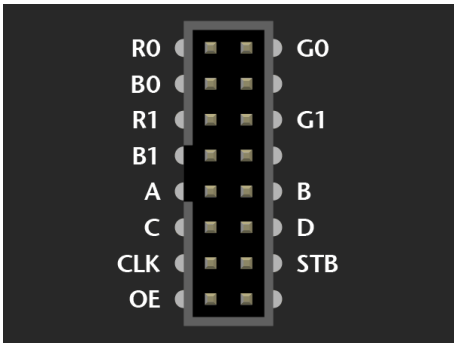
-4 broches A,B,C,D (La broche D n'est pas utilisée pour un fonctionnement en 16x32 mais pour un 32x32, on utilise donc les A B et C , soit $2^3 = 8$. les 8 divisions précédemment expliquées)

-Une broche STB (Stroboscopique)

- Une broche CLK (Clock)

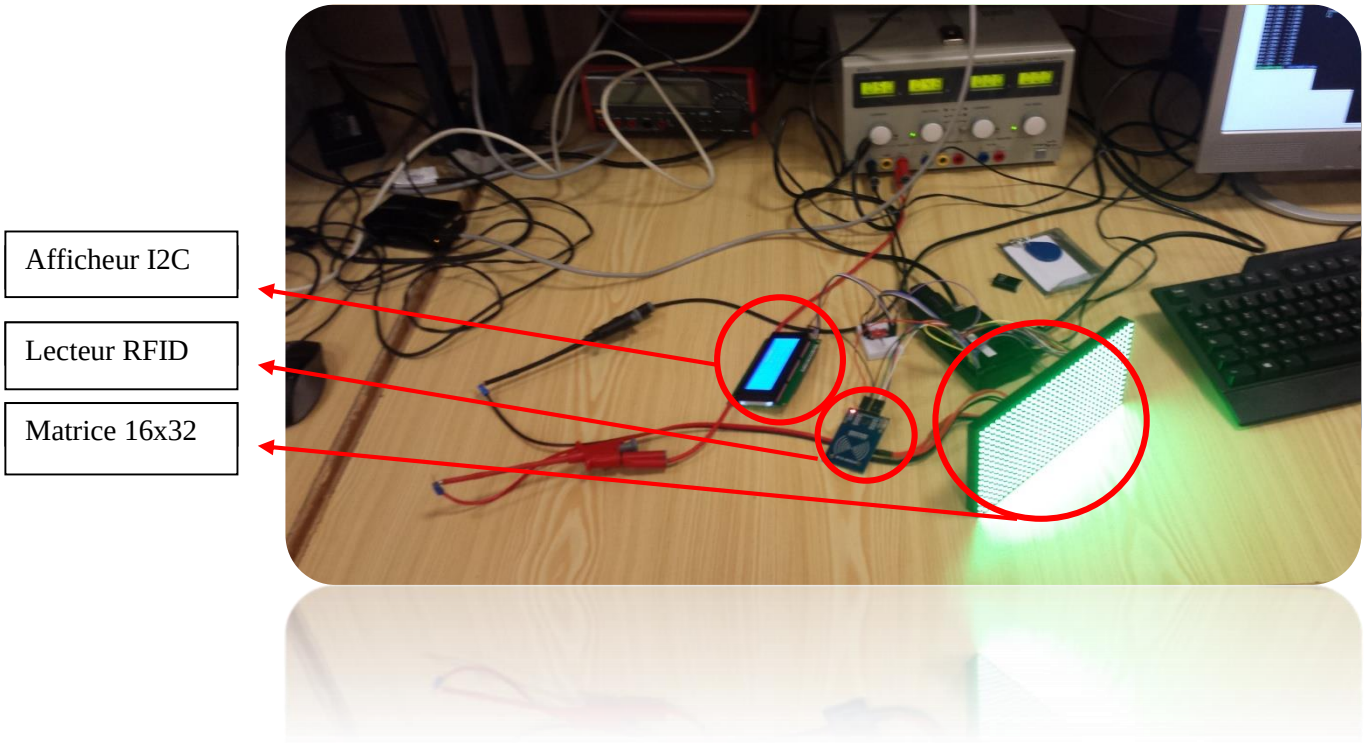
-Une broche OE (OUTPUT/ENABLE)

Permet de « chainer » plusieurs matrices.



11.5 Premier test d'assemblage

Une fois ces tâches accomplies, il fallait tester le fonctionnement en simultané de nos appareils. Nous avons commencé par brancher en simultané La Matrice 16x32 (connexion parallèle), le lecteur RFID (connexion SPI), l'afficheur (connexion I2C) de l'étudiant Damien Gallego et enfin la connexion UART que l'on teste avec un programme d'exemple sur Raspberry.



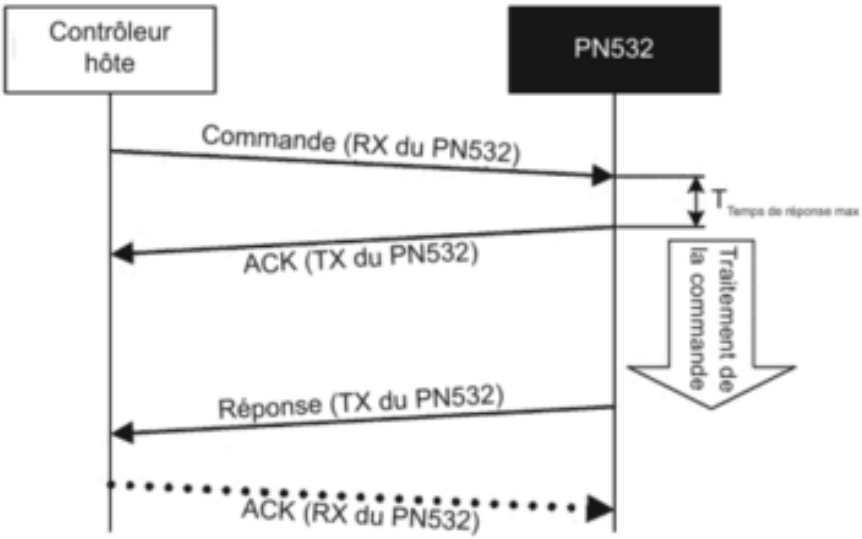
Tout fonctionnait sauf le lecteur Rfid. Le souci, étant la librairie RPI-LED-MATRIX qui prenait la main sur les broches SPI de la carte Raspberry. Plus clairement, Les broches utilisées par le lecteur rfid sont indisponible car utilisées en priorité par la librairie de la matrice, celle-ci étant faite pour combiner plusieurs matrices. Nous avons donc pensé modifier cette librairie pour permettre aux broches SPI de fonctionner normalement. Cependant une autre idée plus simple et plus fonctionnelle s'est présentée à nous, nous avons commandé un nouveau Lecteur de carte, celui-ci est non pas en RFID mais en NFC et utilise seulement 2 fils (connexion UART). Avec les tests de simultanéité précédente nous savions que cette connexion était compatible avec les autres appareils.

11.6 Fonctionnement du Lecteur NFC PN532

La communication en champ proche est une technologie de communication sans contact. Les dispositifs NFC (Near Field Communication) peuvent facilement inter opérer avec d'autres protocoles sans fil, y compris La RFID (identification par radiofréquences). Souvent utilisés pour le contrôle des accès bâtiments etc…

La carte PN532 est dotée des composants matériels nécessaires à la mise en œuvre des protocoles de communication UART, SPI et I2c. Nous utilisons La communication UART, elle est prise en charge directement par la Raspberry. Pendant le premier test d'assemblage, nous avions testé la communication UART, de plus les broche RX, TX de la Raspberry ne sont pas utilisées.

La communication de donnée NFC est un échange qui implique non seulement des commandes d'envoi et de réception mais également des accusés de réception positif(ACK) et négatif(NACK).Ce fonctionnement est comparable à celui du protocole TCP c'est en cela qu'il garantit des communications sans erreurs.



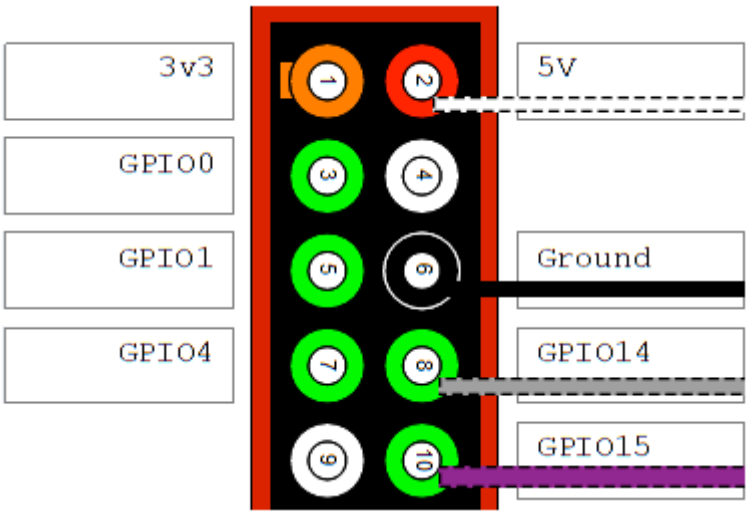
Représentation graphique de l'échange de données entre la Raspberry et le contrôleur PN532

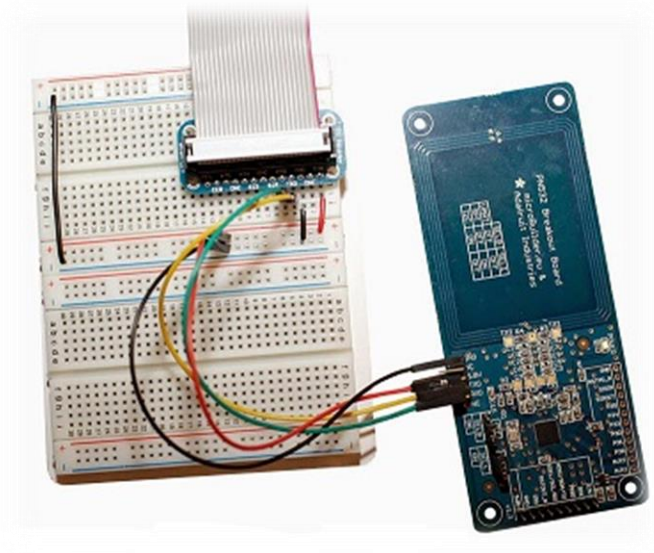
La bibliothèque libnfc prend en charge les communications de données entre le lecteur PN532 et la Raspberry. Le micro logiciel du lecteur envoie des données vers la Raspberry lorsqu'il détecte un tag RFID.

Le lecteur NFC PN532 est muni d'un régulateur de tension (MIC5225) qui permet de passer d'une alimentation en entrée de 5V à une alimentation de 3.3V. Le MIC5225 est un régulateur « low dropout », c'est-à-dire qu'il peut y avoir une faible différence de tension entre Vout et Vin.

11.7 Mise en œuvre du Lecteur NFC PN532

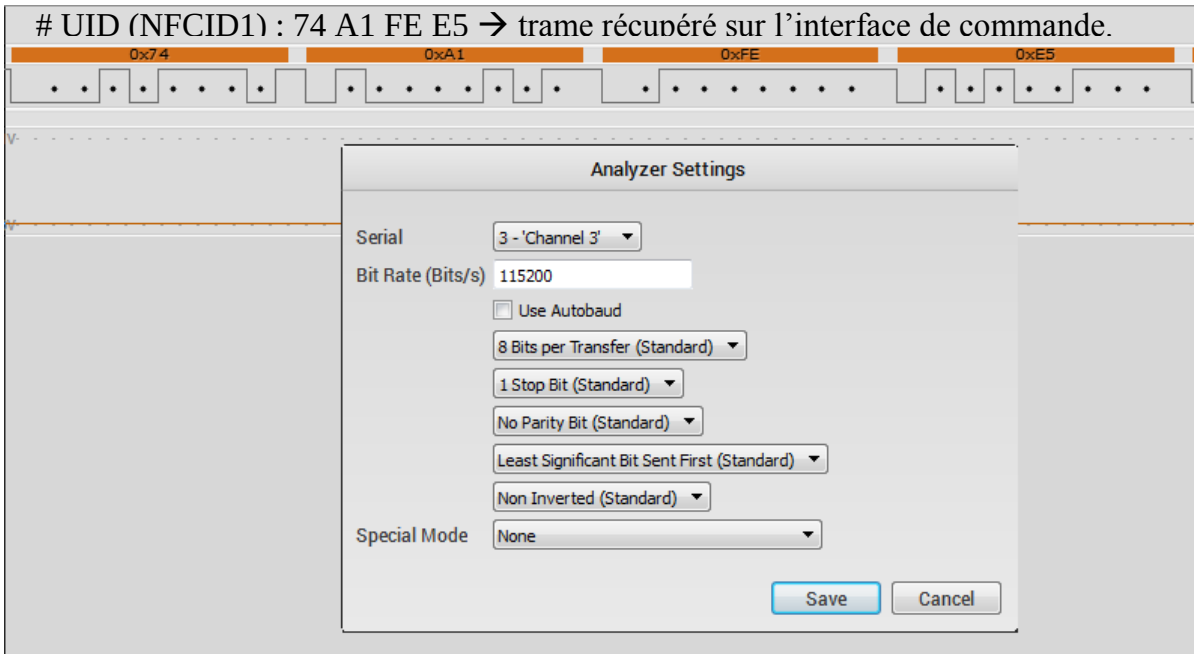
Comme pour le lecteur RFID, mon premier pas fût de rechercher une librairie fonctionnelle, celle-ci trouvée (libnfc-master) et installé (https://learn.adafruit.com/adafruit-nfc-rfid-on-raspberry-pi/building-libnfc), il suffisait de câbler et de la tester (voir annexe p56). Ce lecteur fonctionne en UART le câblage est d'autant plus simple qu'il y a Trois fils à brancher sur la Raspberry.





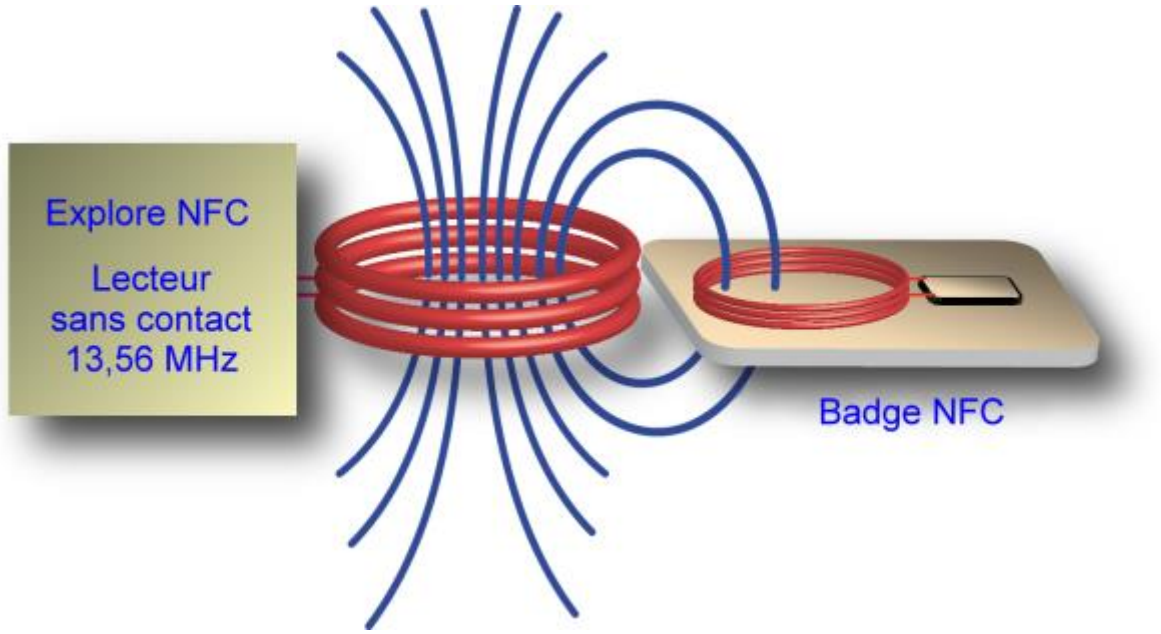
Il me fallait ensuite décoder la trame récupérée par le lecteur pour isoler L'UID des cartes. J'ai pour cela utilisé un analyseur logique « SALEAE » (un analyseur logique permet de décoder la trame directement) que j'ai branché sur les broches de sortie du lecteur, et j'ai récupéré la trame.

➔ Récupération de l'UID sur SALEAE



11.8 Qu'est-ce que la NFC ?

Tout comme la RFID, La lecture NFC se fait par émission d'une onde électromagnétique par le lecteur. Envoyée sur l'antenne du lecteur, un signal à 13,56 MHz provoque l'émission de cette onde. L'énergie est transmise au badge (appelé aussi TAG) situé à proximité.



Le badge comporte lui aussi une antenne (voir ci-dessus la photo d'un badge NFC autocollant) qui va collecter cette énergie. Cette énergie sert à alimenter le circuit intégré (le petit point noir) situé sur le badge. Une fois alimenté il devient actif. Le lecteur module l'onde à 13,56 MHz pour transmettre une demande de lecture au badge, puis continue d'envoyer un signal à 13,56 MHz non modulé pour alimenter le badge NFC. Le TAG peut alors répondre en faisant varier le courant qu'il consomme sur son antenne. Ces variations sont transmises au lecteur qui en déduit la réponse du badge. Autour de la fréquence porteuse de 13.56MHz, on va moduler deux sous-porteuses (à 14.0475 MHz et à 12.7125, donc une largeur de bande = 847.5 KHz). Ce sont ces sous-porteuses qui serviront à transporter l'information.

11.9 Second test d'assemblage

Cette fois ci nous avons branché : La Matrice 16x32 (connexion parallèle), le lecteur NFC (connexion UART), les néopixels (de l'étudiant Alexandre Cartegnie) /L'afficheur (connexion I2C).

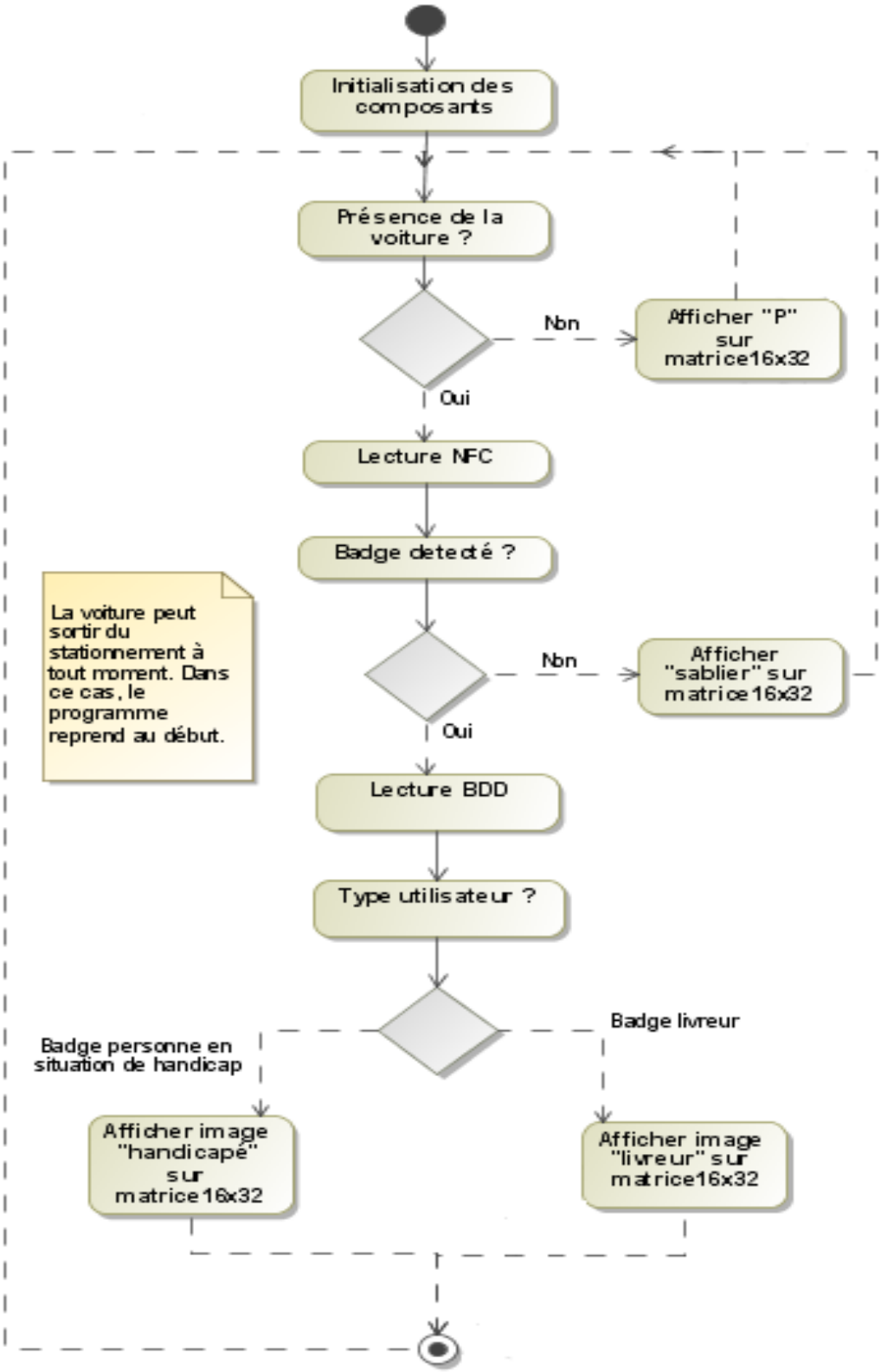
Nous avons choisi d'utiliser le rond néopixel, car cela nous a permis de faire deux tests en un. Plus précisément, nous avons pu tester la simultanéité du lecteur nfc et de la matrice16x32, mais aussi tester le fonctionnement du rond néopixel en communication I2C. Cette fois ci Tout fonctionne correctement, aucunes latences.

11.10 Application QT

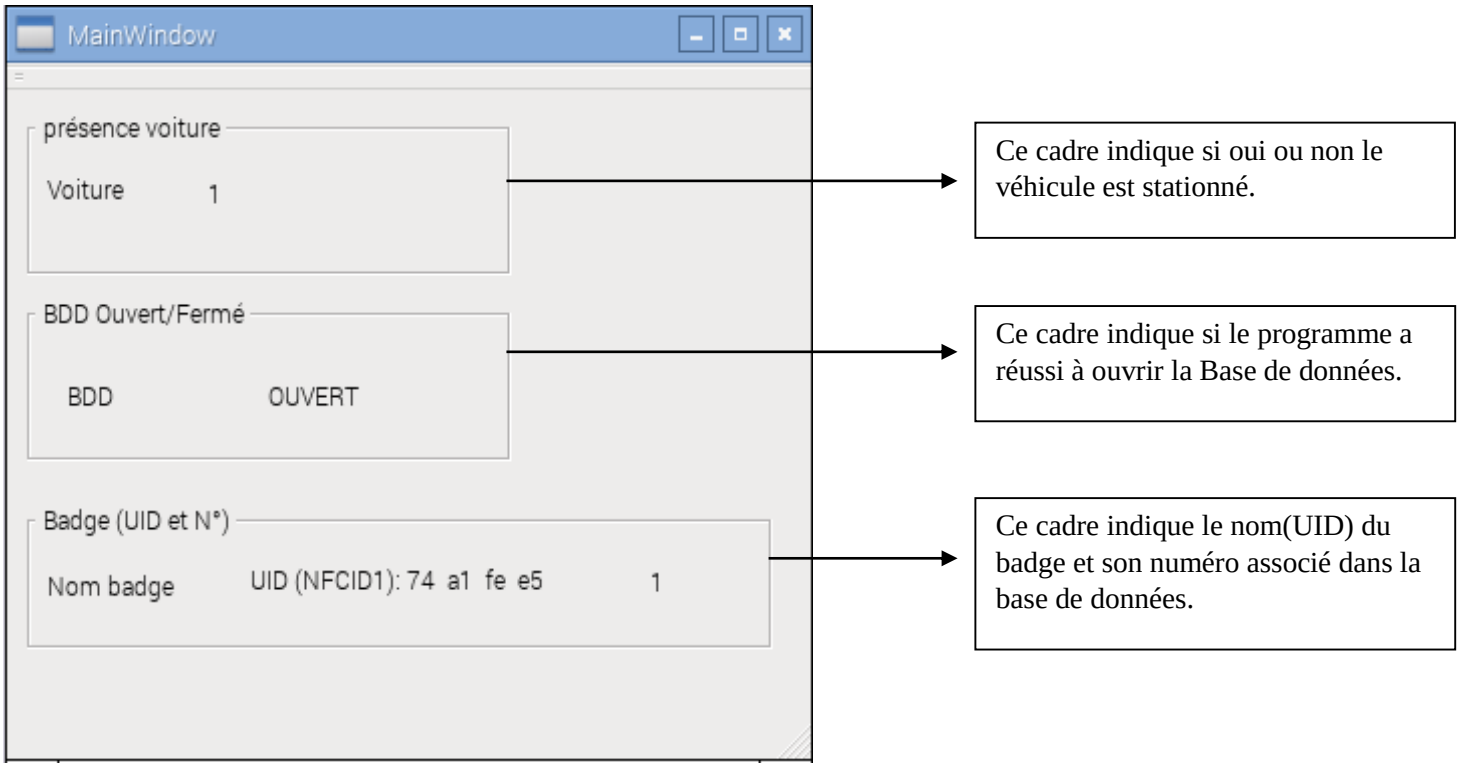
La matrice et le lecteur sont maintenant fonctionnels.

Je me suis donc penché sur une des parties principales de mon projet qui consiste en une application permettant d'afficher les images associées aux badges automatiquement.

11.10.1 Diagramme d'activité



11.10.2 Application



-J'ai commencé par lancer le programme permettant de lire le badge à partir de l'application.

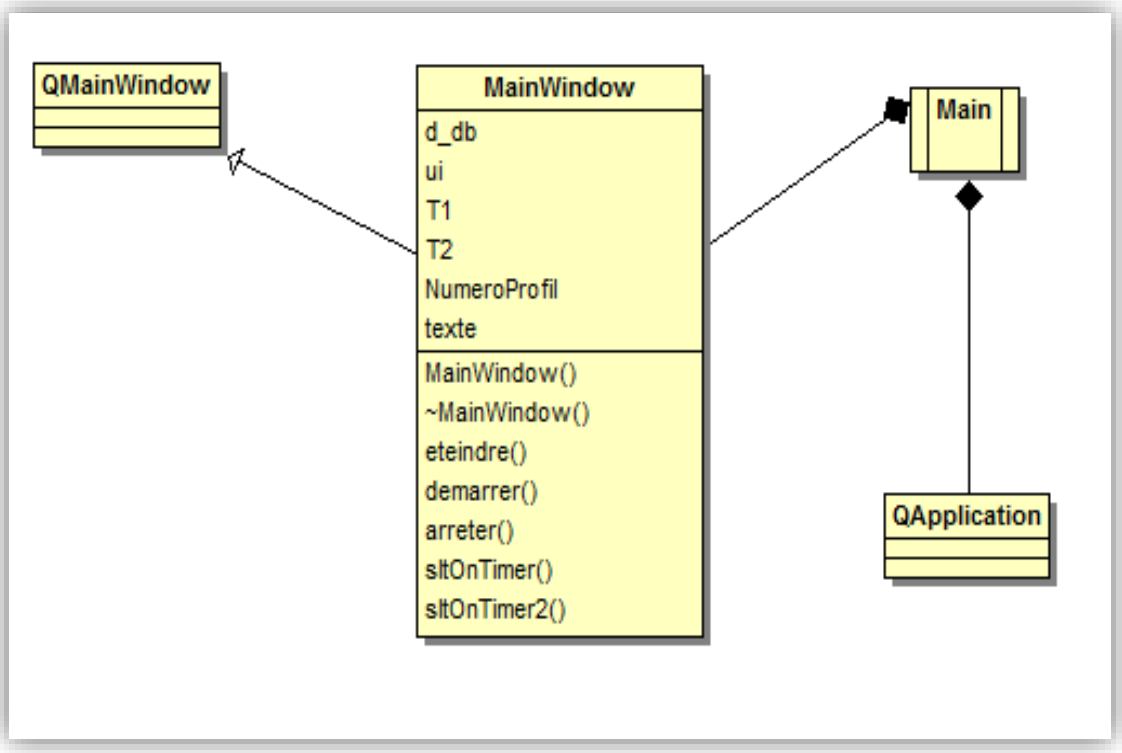
-J'ai ensuite réussi à afficher les images sur la Matrice à partir de l'application.

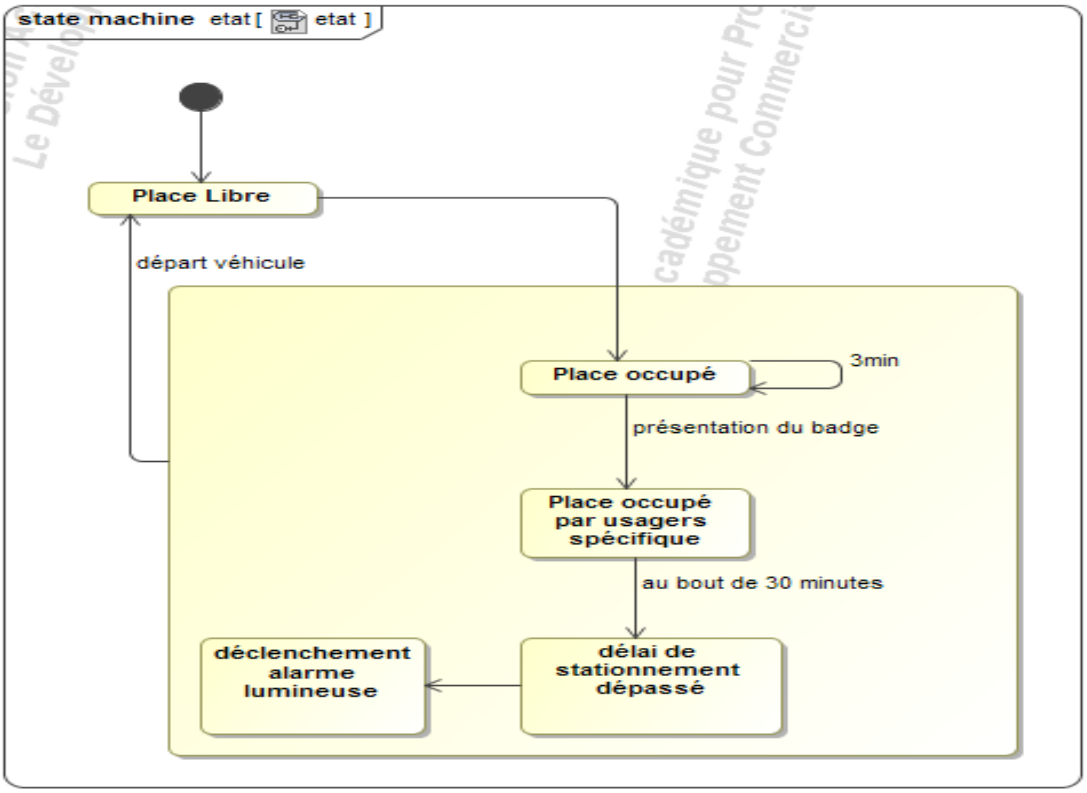
-Puis je me suis attelé à afficher une image à partir de L'UID récupéré par le lecteur.

-Le projet doit pouvoir fonctionner dans la réalité, donc il faudra pouvoir stocker les UID dans une BDD, Thomas Vincent s'en est occupé. J'ai donc entrepris de lire cette BDD à partir de l'application.

-Enfin, l'élément déclencheur qui active mon application est la lecture de l'état d'une broche (Voiture stationnée ou non) à partir des constructions d'Alexandre Cartegnie et de Pierre Reynaud qui s'occupent de la détection véhicule.

11.10.3 Diagramme de classe et état



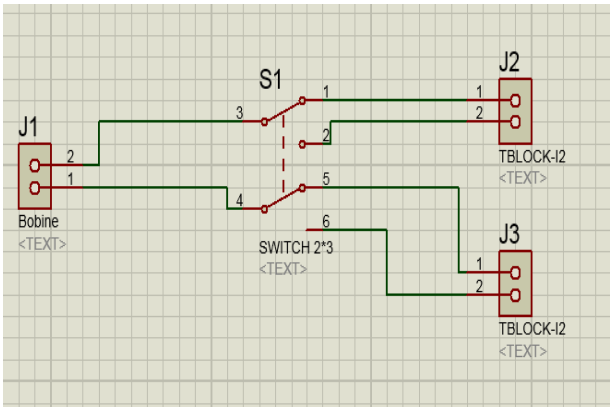


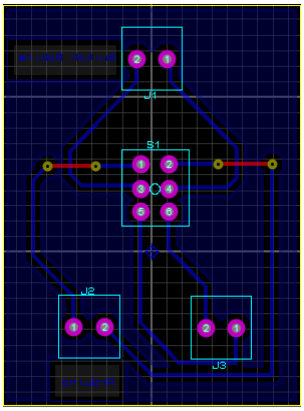
11.11 Librairie

Différentes application peuvent utiliser le même type de fonctions : calculs, gestion de fichiers, affichage, etc... Cela augmente la taille des fichiers sources, parfois énormément, et surtout, si l'on fait une modification, il faut la reporter dans tous les programmes si l'on n'utilise pas de librairie.
Tout d'abord qu'est-ce qu'une librairie. C'est une collection de fonctions, prêtes à être utilisées par des programmes.

Pour alléger mon programme, le rendre plus lisible et utilisable par Thomas Vincent, j'ai créé une librairie qui assemble toutes ses fonctions. Vous la trouverez en annexe p58.

11.12 Construction





Dans le cahier des charges, je n'ai pas de construction à faire. Cependant, pour permettre de « switcher » entre arduino et RPI nous avons décidé de créer une carte.

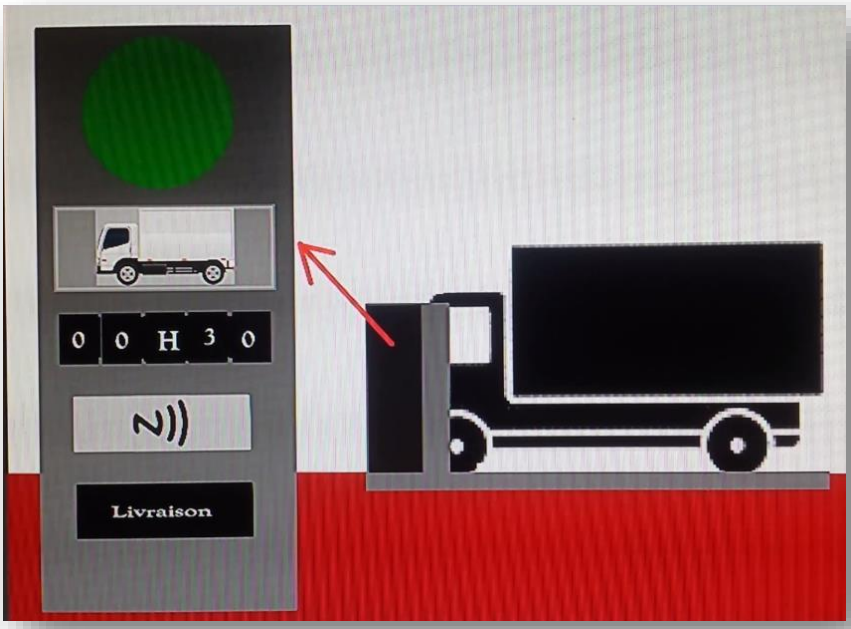
Les composants utilisés sont les suivants :

- Un interrupteur à bascule 2RT

-Trois Borniers à deux contacts au pas de 5.08 mm

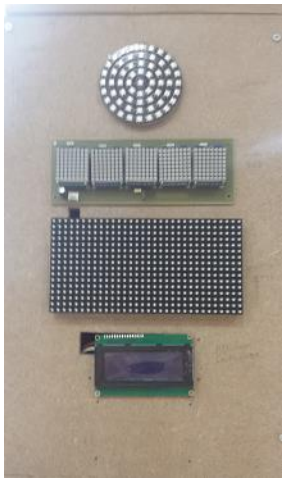
11.13 Animation 2D du scénario « Livraison »

Pour mieux comprendre et faire comprendre le but de notre projet, j'ai aussi créé une animation pour le scénario « livraison » avec le logiciel OpenToonZ. Cette animation permet de voir clairement le fonctionnement de la borne en fonction du stationnement d'un livreur.

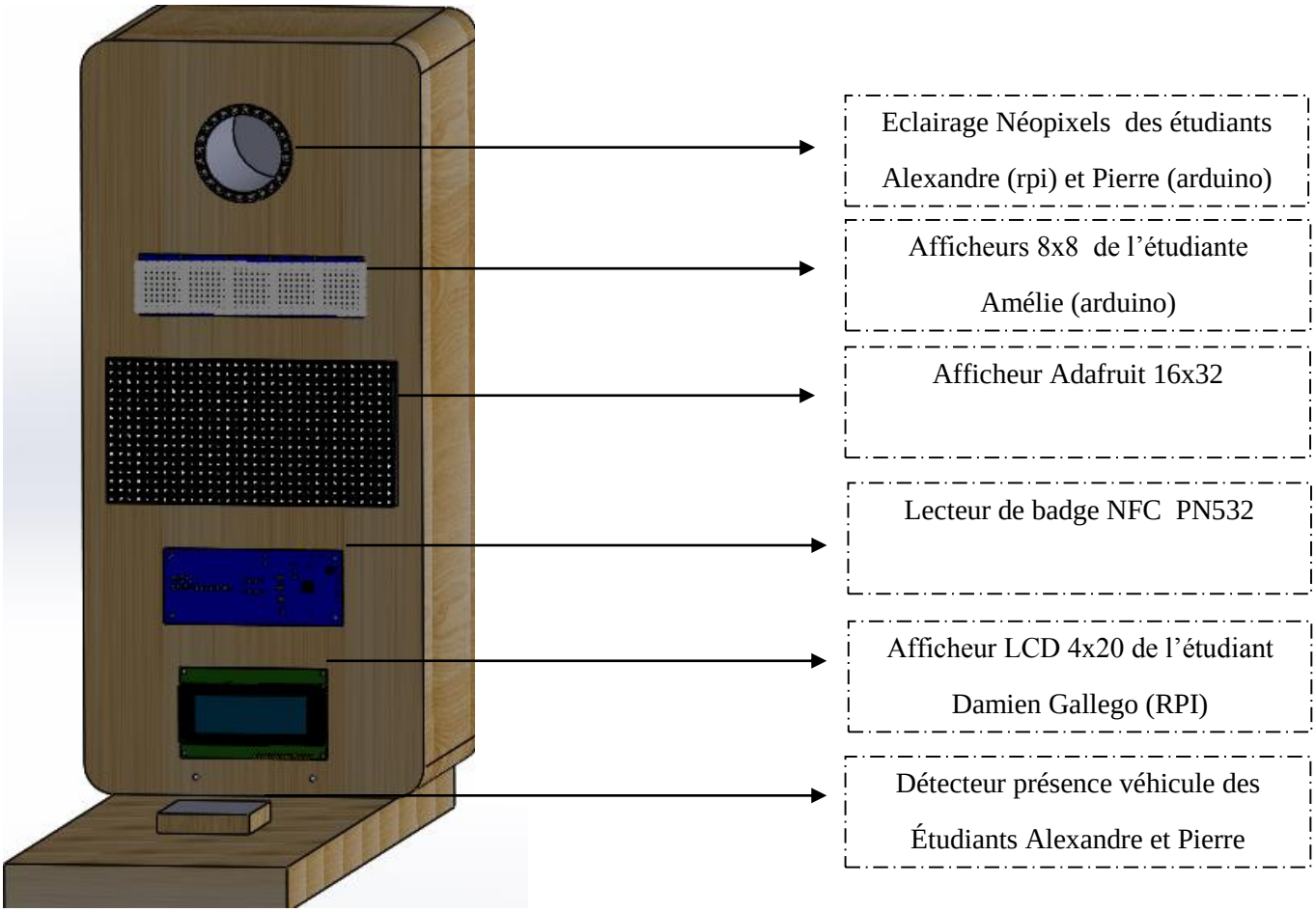


11.14 Prototype de la borne physique

Avec l'aide de la modélisation 3D de la borne, nous sommes allé commander les planches (assez fine, pour permettre au lecteur nfc de capter le badge à travers) qui nous permettront de créer physiquement la borne.



11.15 Borne sous SOLIDWORKS



Pour avoir un premier aperçu, j'ai avec le logiciel SOLIDWORKS créé la borne physique virtuellement. Cette modélisation nous a permis d'avoir une vue d'ensemble sur les éléments des projets, et de pouvoir déterminer les dimensions nécessaire à la création physique de la borne.

11.16 Améliorations possibles

Durant le projet nous somme allé voir l'entreprise URBACO qui nous a donné son avis sur le projet et de ce fait nous a permis de penser à des améliorations possible du projet ci-dessous la liste des amélioration possible :

- Créer une base de données sur les badges nfc.

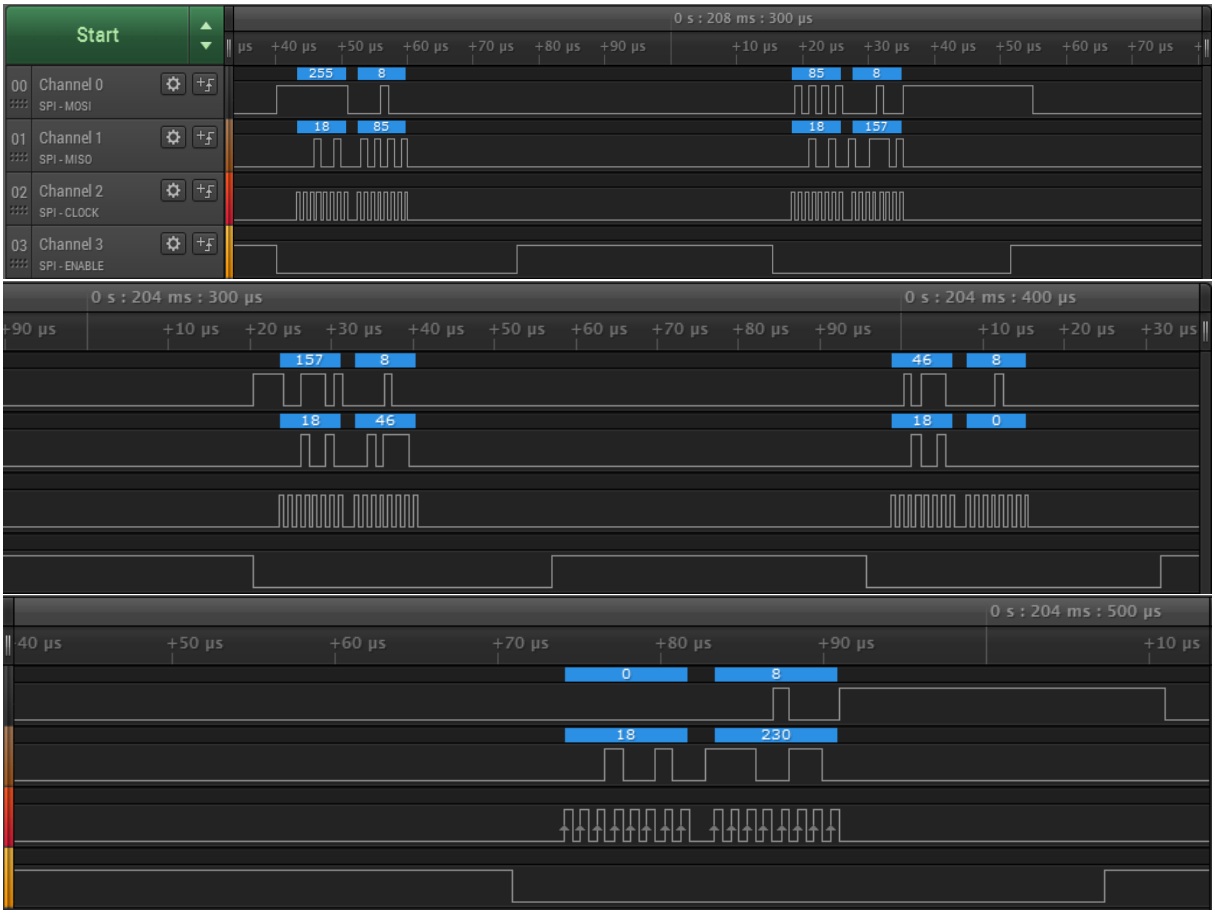
- Ajout d'une marge d'erreur pour le temps de stationnement.

- Alimenter la borne avec une batterie, elle sera chargée pendant la nuit avec l'éclairage publique.

- Eteindre la borne à certain moment ou juste l'affichage pour faire des économies d'énergies.

12 ANNEXE

12.1 Trame RFID

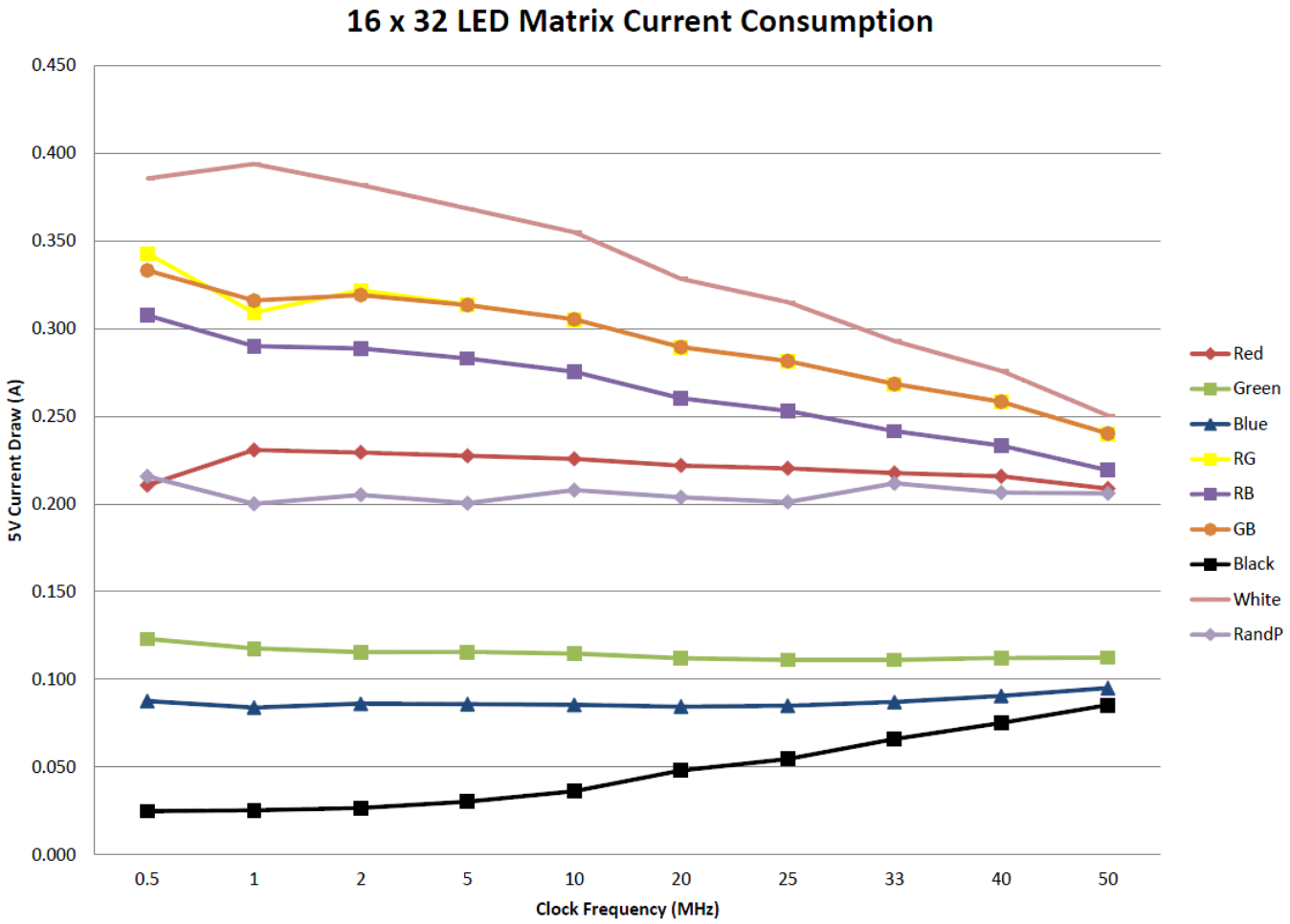


SPI	MOSI: 8	MISO: 85
SPI	MOSI: 85	MISO: 18
SPI	MOSI: 8	MISO: 157
SPI	MOSI: 157	MISO: 18
SPI	MOSI: 8	MISO: 46
SPI	MOSI: 46	MISO: 18
SPI	MOSI: 8	MISO: 0
SPI	MOSI: 0	MISO: 18
SPI	MOSI: 8	MISO: 230

Quand je passe le badge devant le lecteur, cette trame apparait sur le logiciel SALEAE.

12.2 Consommation Matrice 16x32

Consommation du courant par rapport à la fréquence de l'horloge.



12.3 Fiche recette (Lecteur NFC)

Qualification du système :	**Borne Arrêt Minute**	Code de la campagne de test :	xxx
Etudiant :	Matéo Blettery	Date :	23/05/2016

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Lecteur NFC
Objectif du scénario :	Montrer au client le bon fonctionnement du Lecteur NFC

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

-Alimentation 5V
-RaspberryPi2
-Connexion « uart » activé
-Librairie : libnfc-master téléchargée et installée
-câbler le lecteur NFC à la Raspberry

EXECUTION DU TEST

Description courte :	-Ouvrir une interface de commande et écrire « nfc-poll » -passer un badge devant le lecteur
Résultats attendus :	-les données du badge s'affichent à l'écran

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

12.4 Fiche recette (Matrice 16x32 Adafruit)

Qualification du système :	**Borne Arrêt Minute**	Code de la campagne de test :	xxx
Etudiant :	Matéo Blettery	Date :	23/05/2016

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Matrice16x32 Adafruit
Objectif du scénario :	Montrer au client le bon fonctionnement de la Matrice16x32 Adafruit

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

-Alimentation 5V
-RaspberryPi2
-Librairie : RPI-RGB-LED-MATRIX-MASTER téléchargé
-câbler la matrice à la Raspberry

EXECUTION DU TEST

Description courte :	Ouvrir une interface de commande et écrire « sudo ./led-matrix –D 1 runtext.ppm »
Résultats attendus :	-un texte s'affiche sur la matrice

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

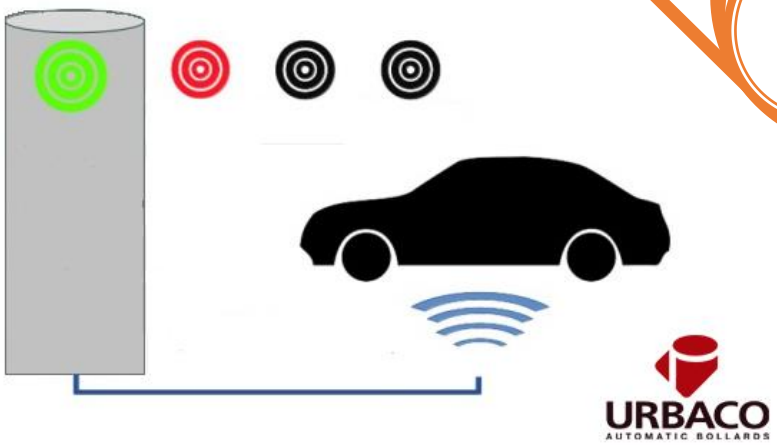
(Mettre une croix dans la case correspondante)

12.5 Librairie

```
class Matrice16x32
{
public:
    Matrice16x32(int seGarer);

    void camion();
    void handicape();
    void attendre();
    void eteindre();
    void parking();
    void nfc();
    void nobadge();
    void presenceVoiture();
    void activation();
    void connexionBDD();


    QSqlDatabase d_db;
    QString d_host , d_dbname, d_username , d_pwd;
```

C'est l'ensemble des méthodes essentiel au bon fonctionnement de ma partie du projet.

	**Epreuve E6.2 Projet URBACO Raspberry PI 2016**	



Projet URBACO RPI

BTS Systèmes Numériques

Alexandre CARTEGNIE

Candidat : Alexandre CARTEGNIE

13 Introduction

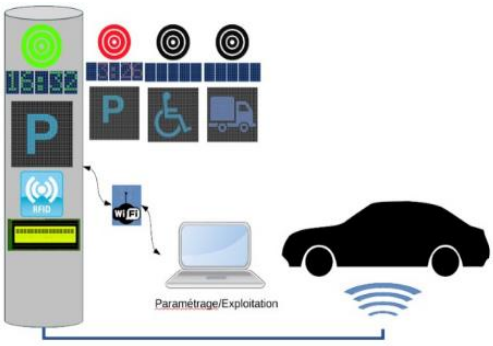
Durant la seconde année du BTS SN, il est demandé aux étudiants de réaliser un projet permettant de mettre en pratique les connaissances acquises durant les deux années de ce BTS.

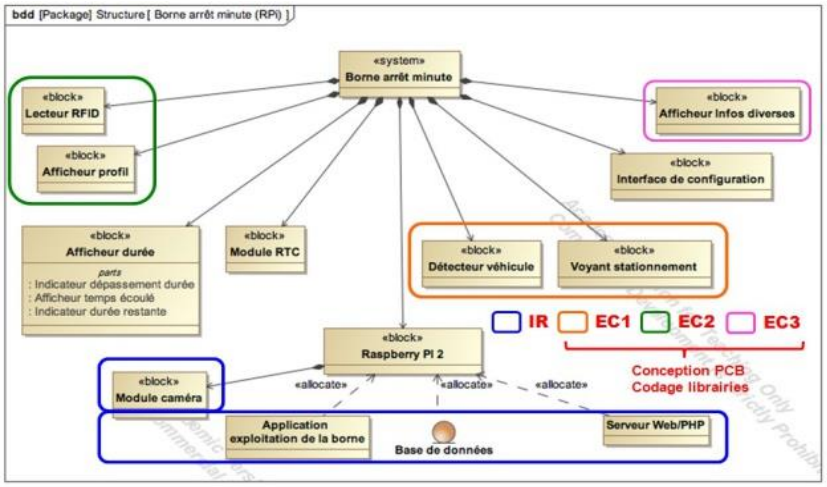
Le projet inclut ici trois étudiants d'Electronique et Communication (Matéo Blettery, Damien Gallego et Alexandre Cartegnie) et un étudiant d'Informatique et Réseaux (Thomas Vincent).

Le but de ce projet est de réaliser une borne de stationnement basée sur un modèle existant de la société URBACO basée à Entragues (84).

Les parties qui m'ont été confiées pour mener à bien ce projet sont fortement liées, il s'agit d'effectuer la signalisation de la borne pour indiquer visuellement si l'emplacement est « libre », « occupé » ou bien si le temps de stationnement est « dépassé ».

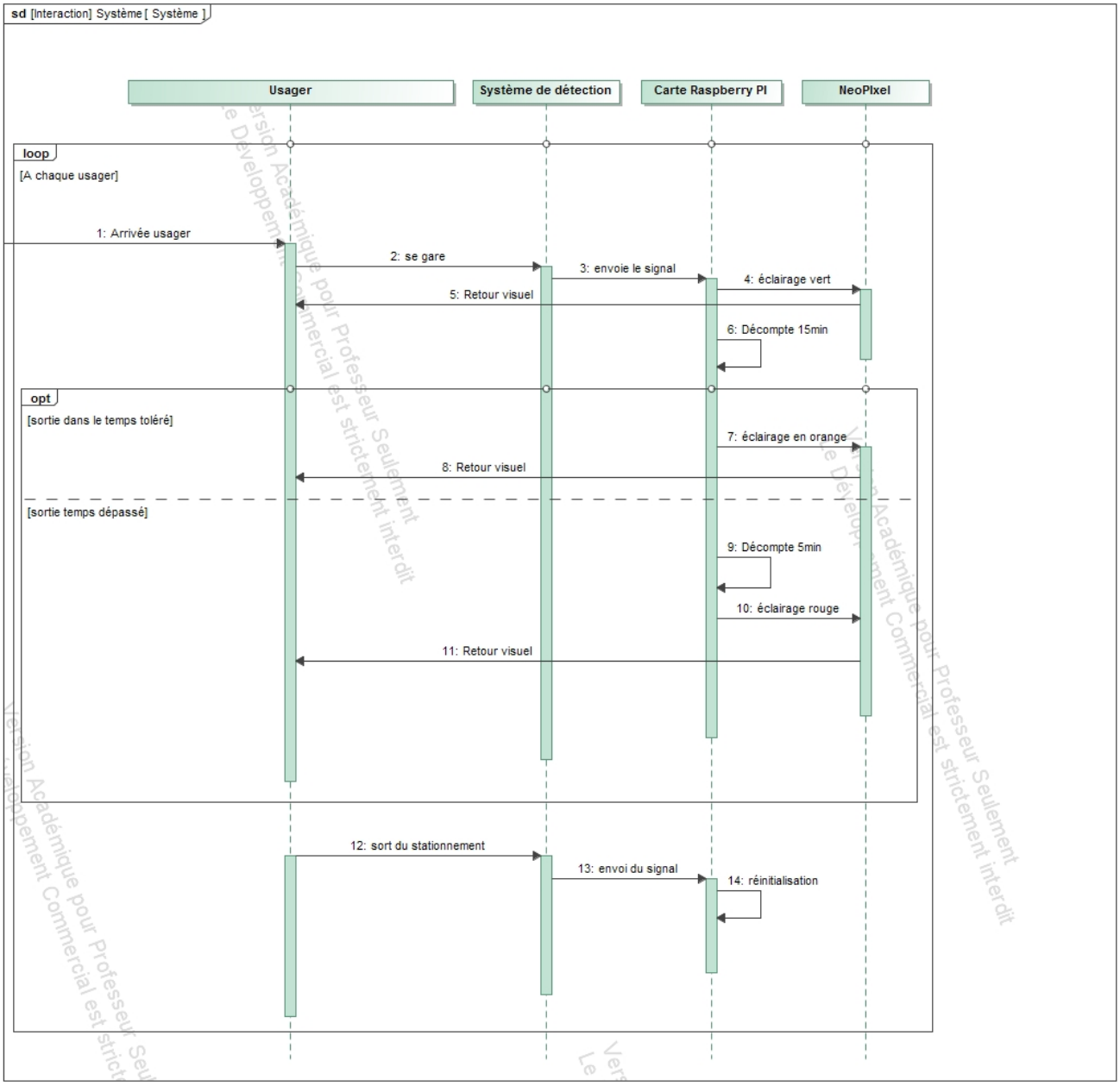
Je dois donc m'occuper de la gestion de l'éclairage des LED, mais également de la partie détection du véhicule permettant de savoir si un véhicule stationne sur l'emplacement de stationnement.

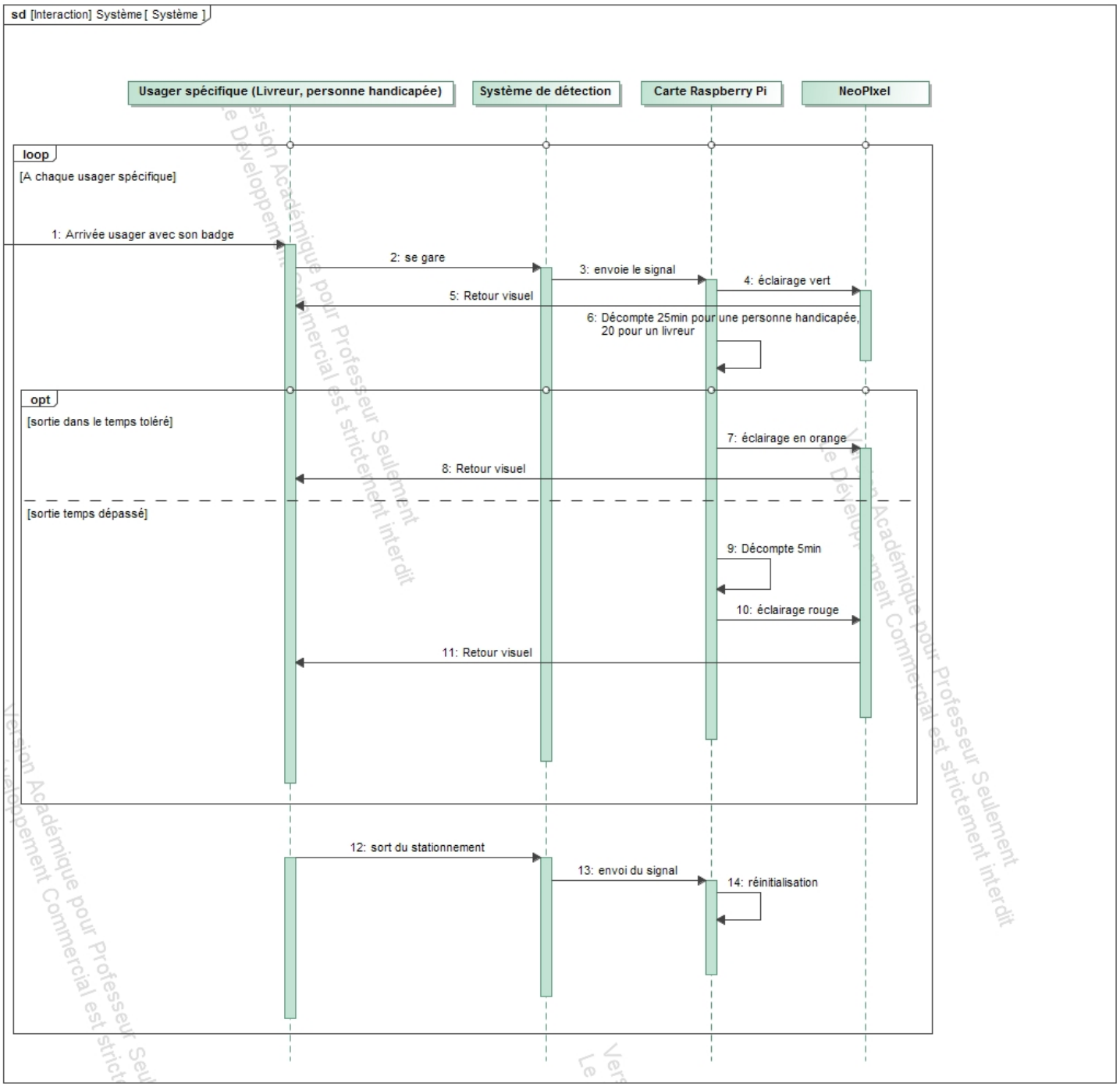




14 Visualisation SysML / diagrammes de séquences

Voici deux diagrammes de séquence présentant l'enchainement des différentes séquences en fonction de plusieurs cas de figure, le premier représentant une utilisation d'un usager « lambda », le second définit un usage d'un usager défini (livreur ou handicapé).




sd [Interaction] Système [Système]
Usager spécifique (Livreur, personne handicapée)
Système de détection
Carte Raspberry Pi
NeoPIxel
loop
[A chaque usager spécifique]
1: Arrivée usager avec son badge
2: se gare
3: envoie le signal
4: éclairage vert
5: Retour visuel
6: Décompte 25min pour une personne handicapée, 20 pour un livreur
opt
[sortie dans le temps toléré]
7: éclairage en orange
8: Retour visuel
[sortie temps dépassé]
9: Décompte 5min
10: éclairage rouge
11: Retour visuel
12: sort du stationnement
13: envoi du signal
14: réinitialisation

15 L'éclairage de stationnement

15.1 Les LED NEOPIXEL

Pour signaler la borne et avoir un aperçu rapide de l'état de stationnement des véhicules nous avons décidé avec l'ensemble de l'équipe de placer sur le haut de la borne un anneau de LEDs NEOPIXELS.

Nous avons choisi des LEDs NEOPIXELS pour plusieurs critères :

-La faible consommation d'énergie

-La forte luminosité

-La combinaison de nombreuses couleurs

-La commande sur une broche de données

-Le prix : environ 10€ pour 24 LEDs

Nous nous sommes concertés pour savoir comment gérer les différentes couleurs à attribuer pour un cycle de fonctionnement.

-Nous avons donc choisi le vert fixe lorsque la borne est libre, le vert clignotant lorsqu'un véhicule stationne sur l'emplacement.

-Lors de jours fériés, ou le dimanche l'affichage devient bleu clignotant pour signaler un stationnement gratuit.

-Pour les livraisons nous avons choisi un jaune clignotant.

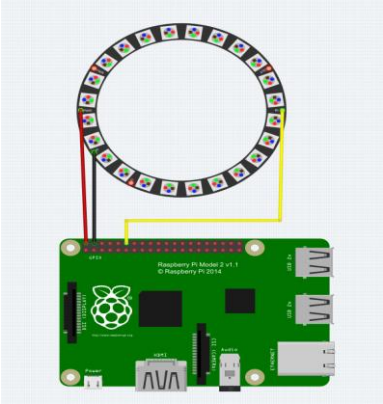
-Enfin un rouge fixe s'éclaire lorsque la limite de temps de stationnement est atteinte, qui passe ensuite en rouge tournant rapidement quand le véhicule est verbalisable par un agent.

15.2 Avancement du projet d'éclairage

Pour comprendre dans un premier temps le fonctionnement des LED WS2812B, j'ai dû effectuer quelques recherches afin de trouver une librairie compatible sur une carte Raspberry Pi2.

Une fois la librairie trouvée, il était désormais possible de faire fonctionner les LEDs grâce à un programme de test en Python fonctionnant avec 24 LEDs.

Le câblage de ces LEDs est simple, il suffit de seulement de câbler 3 fils :



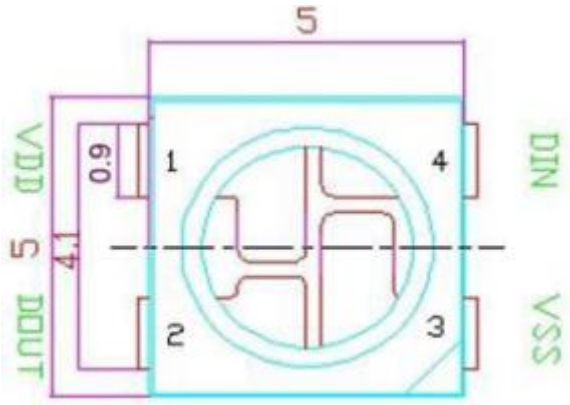
-2 fils d'alimentation 5V &GND

-1fil pour l'envoie de données

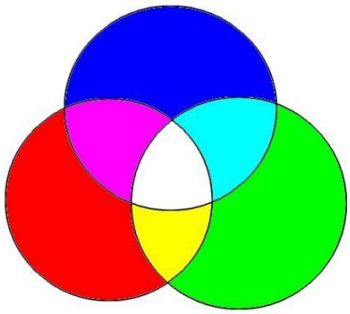
15.3 Détails sur les NEOPLIXEL

Pour la borne nous utilisons un disque de 61 LED pour augmenter la luminosité en plein jour.

Le prix du disque de 61 LED chez notre fournisseur Bangood est de environ 16$ soit 14€ environ. Chaque LED possède 4 pins de connexion, Vcc, GND, Din, Dout disposées selon le schéma suivant :

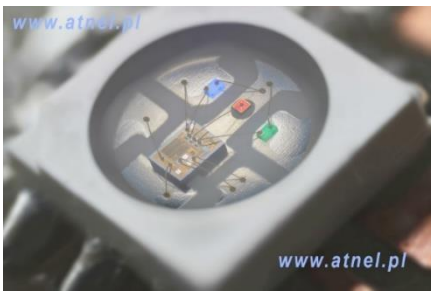


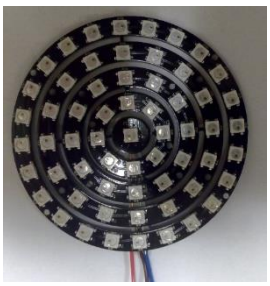
Chaque LED dispose de 3 composantes de couleurs pour pouvoir ainsi reproduire la totalité du cercle chromatique.



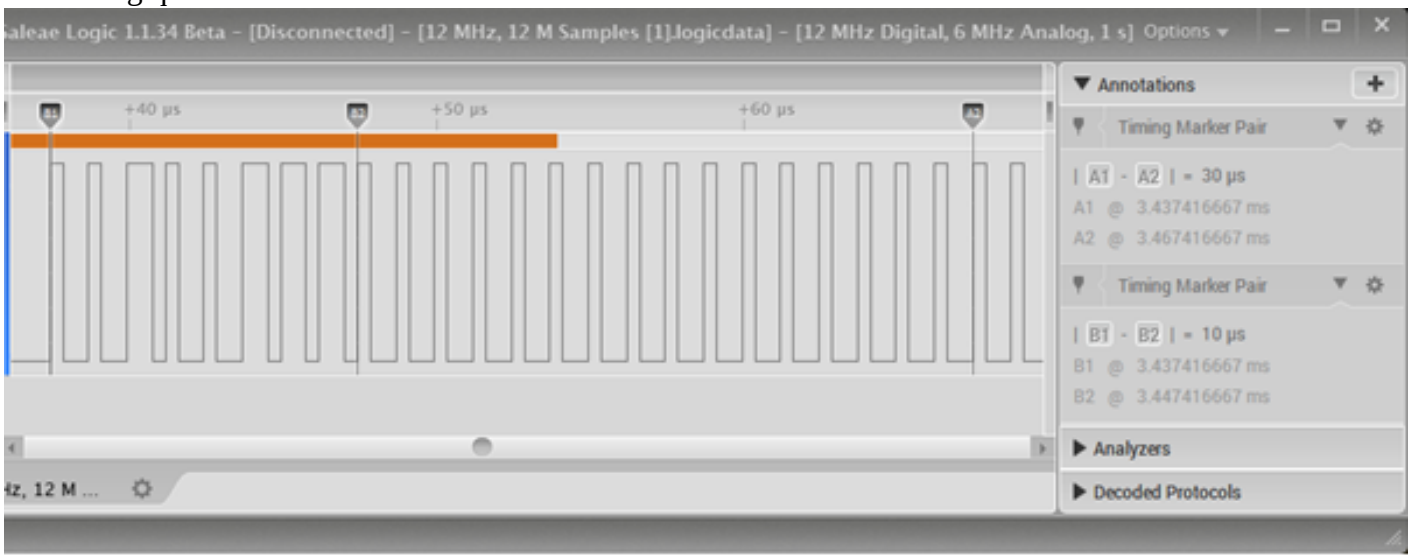
Chaque composante de couleur est codée sur un octet (0-255).

Le protocole de communication suivi par les LED est composé d'un bus de donnés suis le principe des registres à décalages.





Voici ci-dessous une capture de trame correspondant à une LED effectuée avec l'analyseur logique SALEAE.



G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

La transmission d'une trame correspondant à une LED est constituée de la manière suivante, en premier les 8 bits constituant l'intensité de la couleur verte suivi de octets pour la couleur verte et enfin bits pour la couleur bleue. La fréquence de rafraichissement est de 800 kHz.

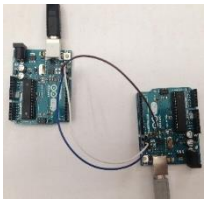
15.4 Les NEOPIXEL et L'ATtiny85

Lors du premier test d'assemblage complet de la borne, un problème est survenu, la broche PWM étant utilisée par la matrice 16*32, la Raspberry ne possédant pas d'autre sortie PWM, nous avons eu recours à une intelligence déportée de type ATtiny85 pour effectuer la gestion des LEDS néopixels.

15.4.1 Test de programmation de l'ATtiny85 avec Arduino

Ce microcontrôleur programmable 8 broches permet grâce à une communication I2C de Controller les LEDs.

Pour comprendre comment programmer le composant, j'ai dans un premier temps fait des essais de communication entre deux cartes Arduino UNO grâce à une liaison de type I2C et un programme d'exemple « I2CMaster » et « I2CSlave ».



Une fois le test effectué, j'ai installé les librairies Arduino correspondant à l'ATtiny.

Suite à cela, j'ai créé un second programme, également sous l'IDE Arduino, comprenant la librairie Néopixels Arduino dédiée à la gestion des LED Néoixels.

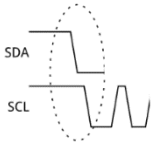
Enfin, un troisième programme rassemblant les deux programmes précédents Pour allumer les LED à partir d'une commande I2C.

15.4.2 Décodage d'une trame I2C

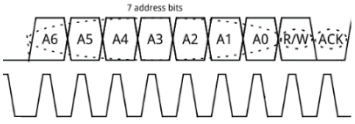
Le programme Néopixels sous Raspberry Pi fonctionne grâce à deux parties distinctes, les octets de commandes sont envoyés via BUS I2C à au composant, qui décrypte les informations pour ensuite communiquer avec les LEDs.

La trame transmise se compose :

- D'une condition de Start

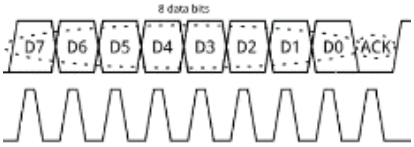


- D'une transmission de l'adresse du composant (7 bits)

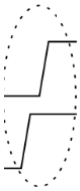


- D'un acquittement

- D'une transmission de l'information (8bits)

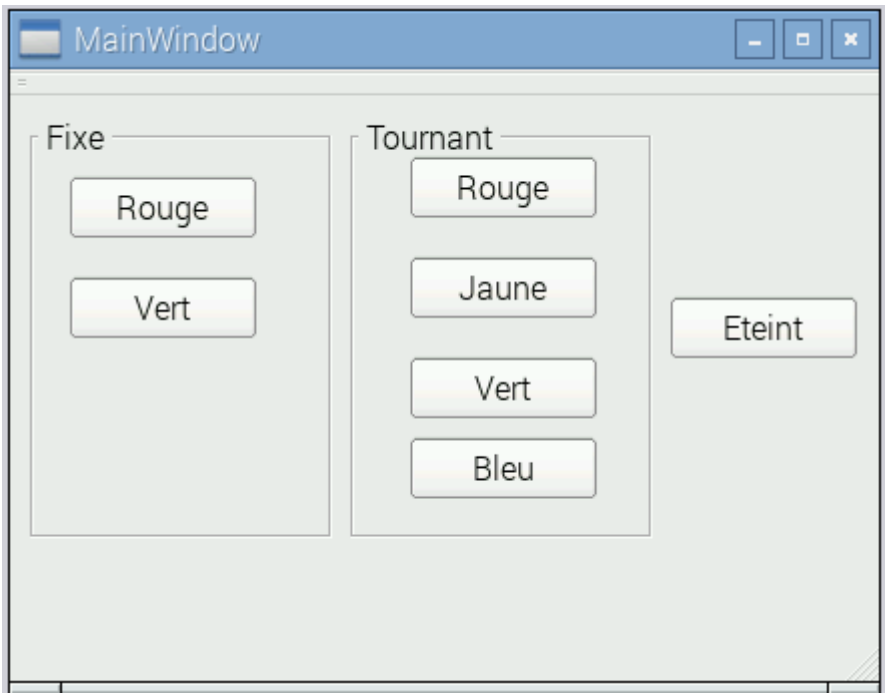


- D'une condition de stop.



15.5 Création d'un programme QT

Pour effectuer des tests simples, j'ai mis en place un programme Qt-creator, regroupant les librairies crées pour l'ATtiny ainsi que les Néopixels, ce programme dispose d'un Form permetant de faire à tous moment d'allumer les LED, de les éteindre ainsi que de les faire clignoter.



16 La détection de véhicule

Pour faire fonctionner les LED, il faut détecter la présence d'un véhicule, j'ai dû donc pour cela travailler étroitement avec Pierre Reynaud (étudiant travaillant sur le projet URBACO Arduino) pour concevoir une carte de détection.

Notre système en grande partie similaire à ceux qui composent les infrastructures routières et comporte une boucle inductive pouvant être placée sous la chaussée. Notre carte de départ faisait partie d'un projet déjà effectué par d'autres élèves d'années précédentes.

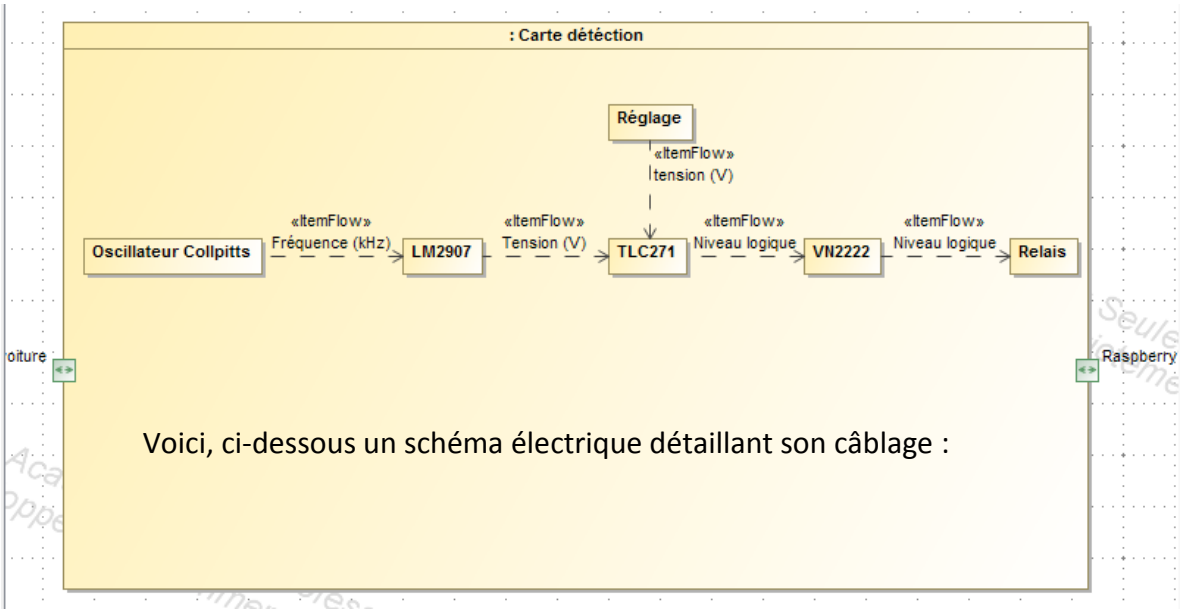
Lorsqu'un véhicule se place sur le stationnement, il fait changer la fréquence de résonnance du circuit grâce aux parties métalliques qui le composent, ce qui fait commuter un relais, qui permet à la carte Raspberry de déclencher ensuite le minuteur de stationnement ainsi que les couleurs correspondantes sur les LED Neopixels.

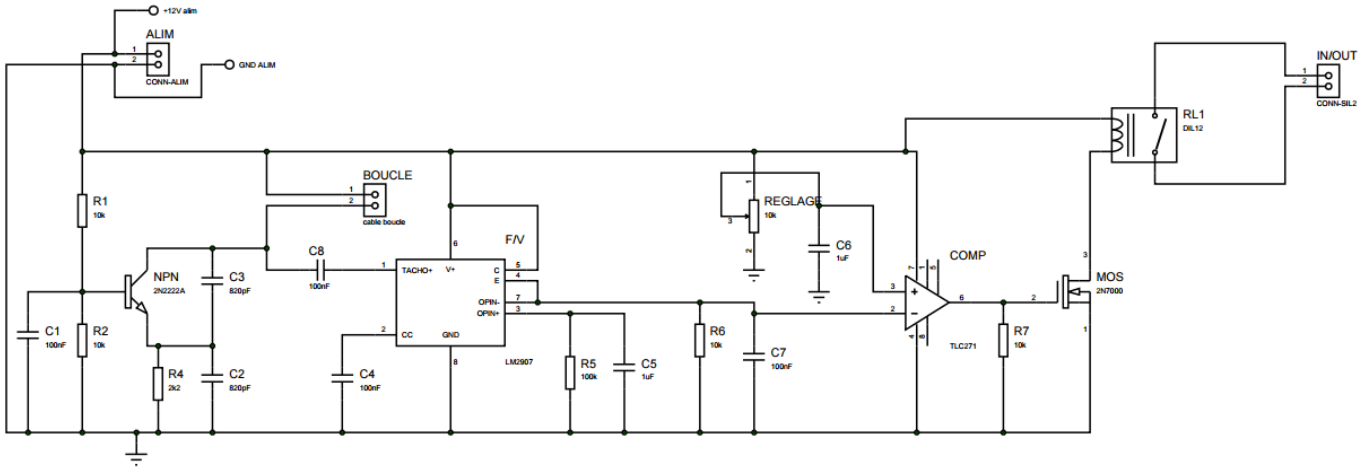
16.1 Carte d'exemple

16.1.1 Principe de fonctionnement

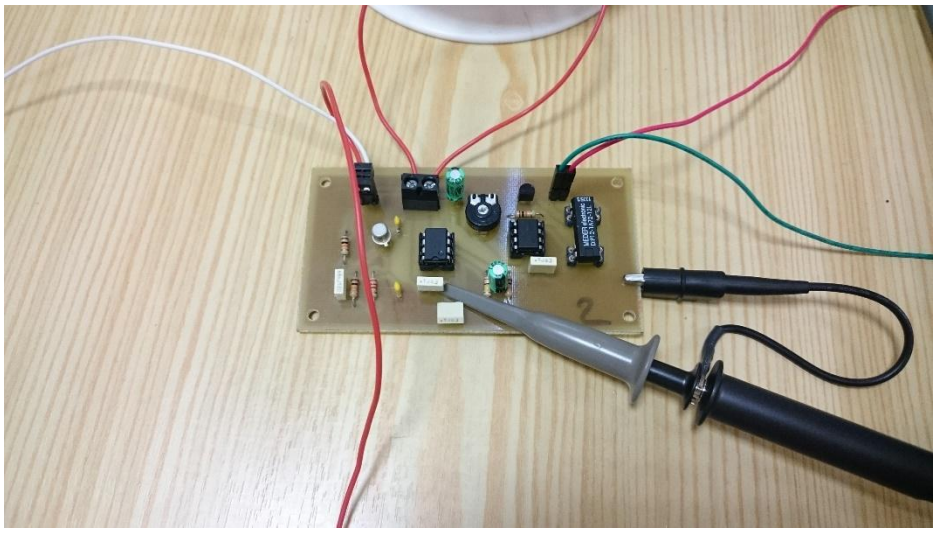
La carte est composée en première partie d'un oscillateur autonome de type Collpitts, ce montage réalisé grâce à un transistor, d'une boucle inductive et deux condensateurs permet de fournir une fréquence d'oscillation et donc un champ magnétique, ce champ pourra être perturbé par la présence d'objets métalliques (dans notre cas un véhicule). Cette fréquence est ensuite convertie en tension grâce à un LM2907, pour enfin être comparée à une tension de référence fixée à l'aide d'un potentiomètre.

(C.f annexe diagramme de block interne)



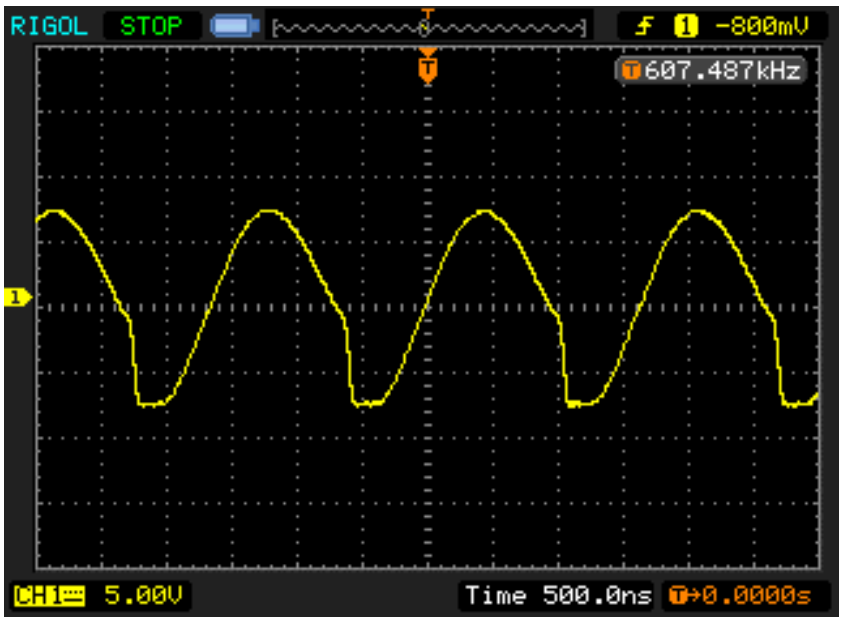


Ci-dessous voici la carte câblée lors d'un test.

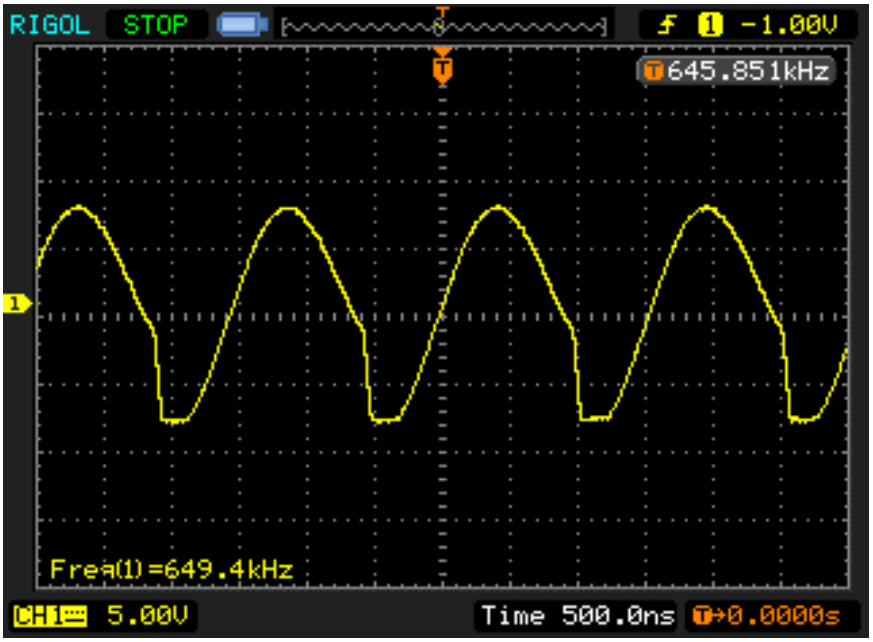


16.2 Premiers tests

La carte d'exemple fonctionne correctement, mais dans une gamme de fréquence est bien supérieure à celle utilisée en réalité pour détecter un véhicule car la plage désirée est comprise entre 50 kHz et 150 kHz.



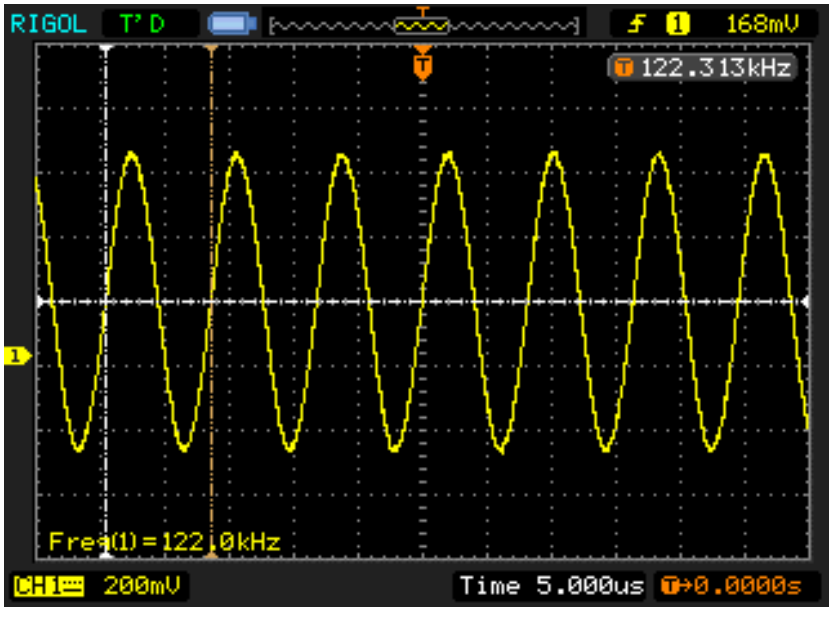
En réalisant les tests à l'oscilloscope, nous avons pu constater que l'oscillateur colpitts produisait une fréquence trop importante qui faisait saturer le convertisseur fréquence tension, la sinusoïde n'était pas propre. Nous avons dû donc rechercher la cause de ce disfonctionnement.



Nous pouvons néanmoins remarquer que la fréquence oscille bien entre deux valeurs, la fréquence la plus basse traduit la présence d'une plaque métallique.

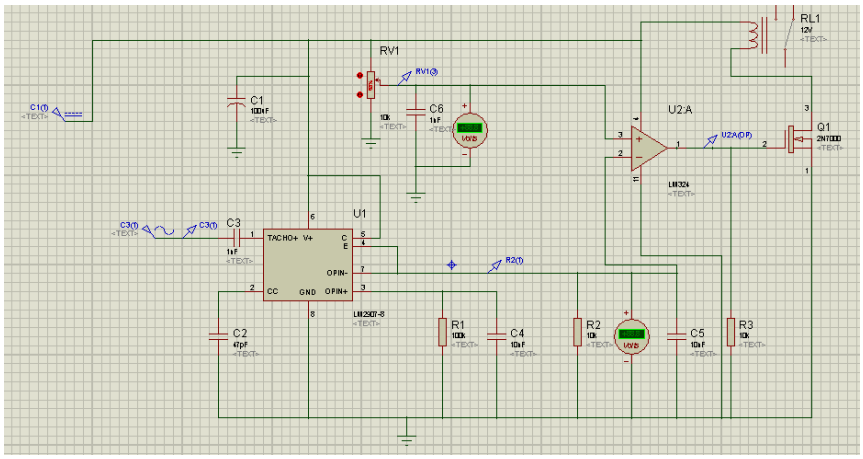
16.3 Réduction de la fréquence

Pour réduire la fréquence d'oscillation, il a fallu redimensionner la taille des condensateurs du Collpitts ainsi qu'estimer la valeur de l'inductance de la bobine. Cela e permis de stabiliser la fréquence de fonctionnement située entre 122 kHz et 12 kHz



16.4 Simulations Proteus

Dans un premier temps, en utilisant uniquement des composants simulables, j'ai pu reproduire un schéma simple pour comprendre le fonctionnement du LM-2907, convertisseur fréquence-tension, élément essentiel de la carte.

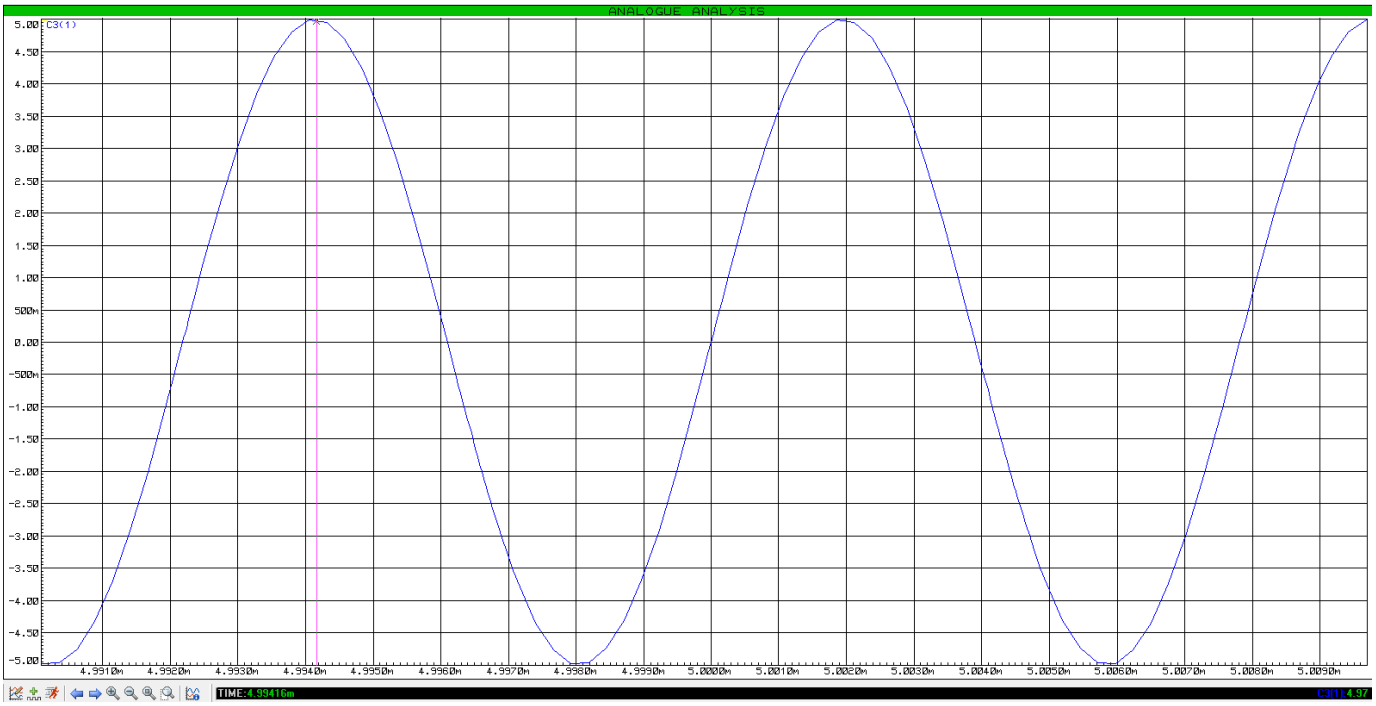


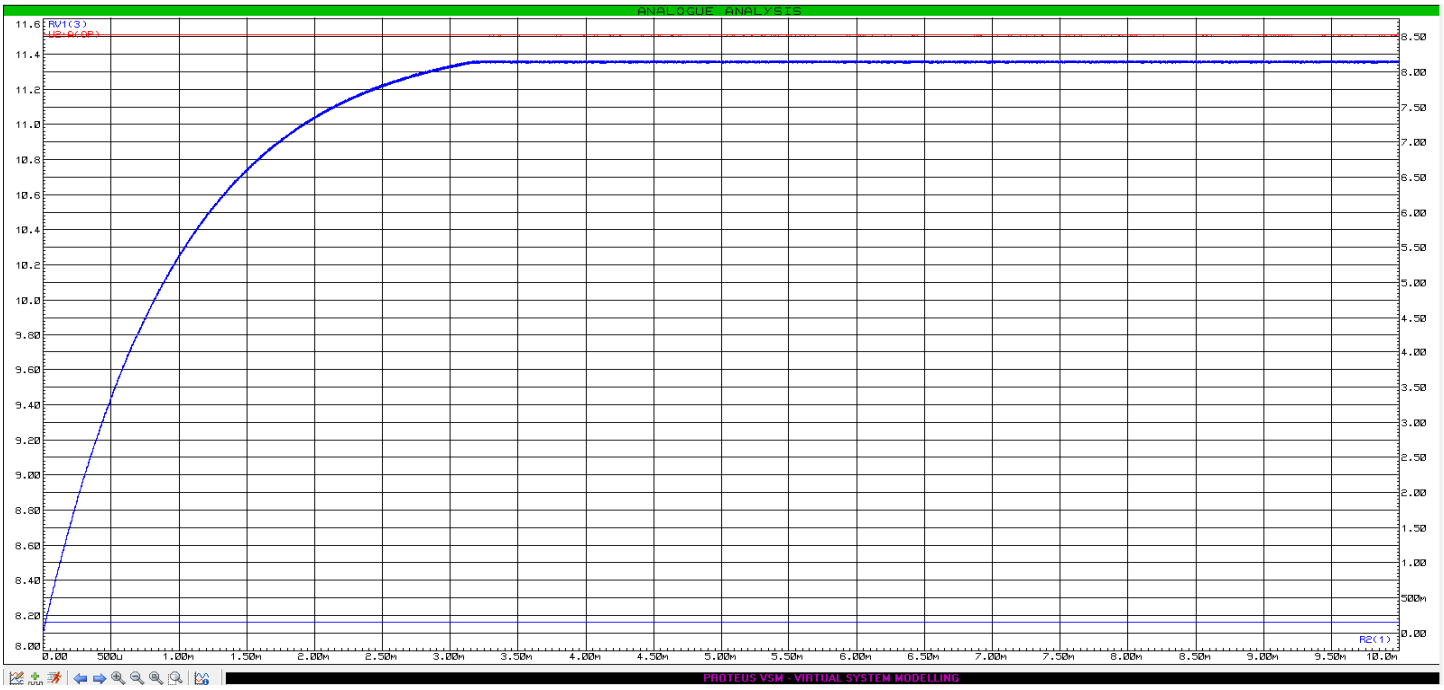
Le condensateur C5=1nF pour une simulation plus rapide, le LM324 remplace le TLC271 car il existe un modèle simulable, le relais et le potentiomètre réglable sont également des modèles simulables.

16.4.1 Fonctionnement du LM2907

Le comparateur fréquence tension que nous utilisons est un LM2907, c'est un composant 8 broches qui peut être alimenté en tension continue jusqu'à 16V. Ce composant fonctionne de manière linéaire selon l'équation suivante : VOUT = fIN × VCC × R1 × C1.

Voici ci-dessous les courbes obtenues grâce à la simulation sous Proteus ISIS.

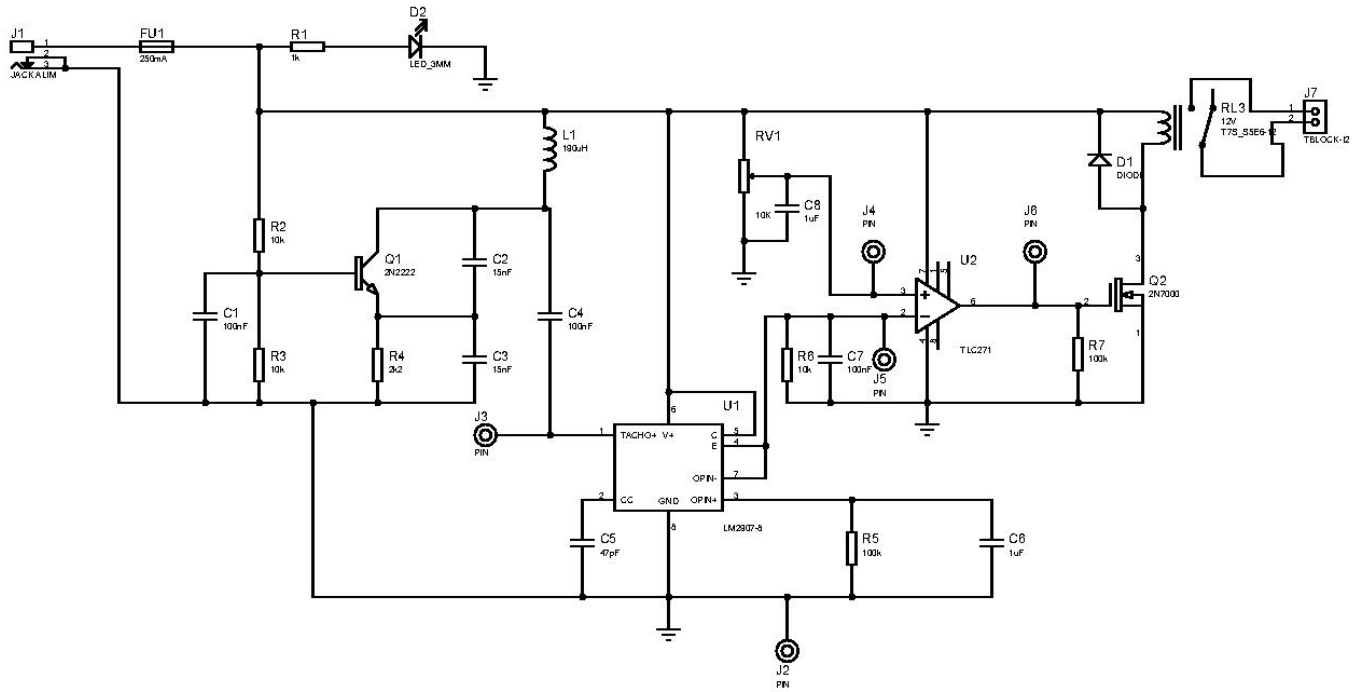




Nous pouvons remarquer que pour une fréquence de 128 kHz la tension fournie par le LM2907 est d'environ 11.3V.

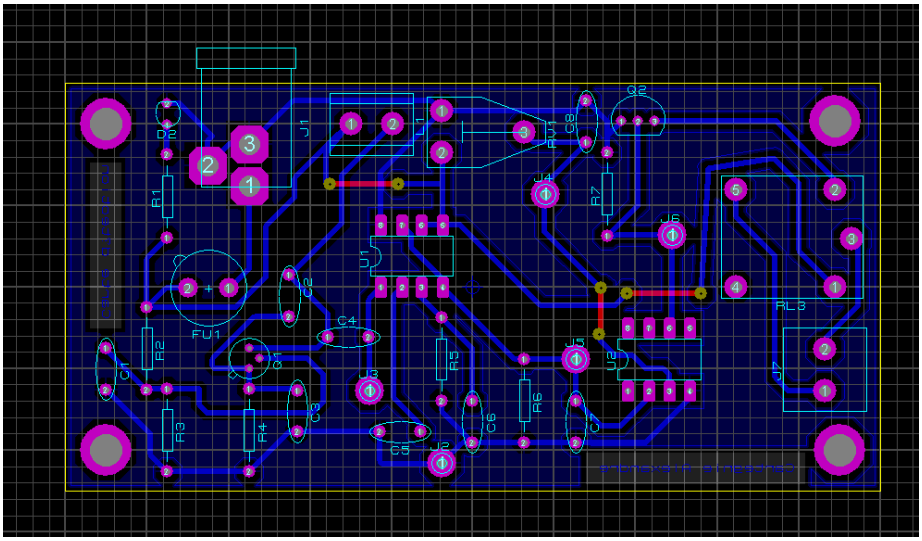
16.5 Routage et fabrication

Suite à ces essais, j'ai donc commencé le routage de la carte en commençant par le schéma de principe de la carte, reprenant cette fois des composants classiques disposant d'une empreinte à taille réelle.



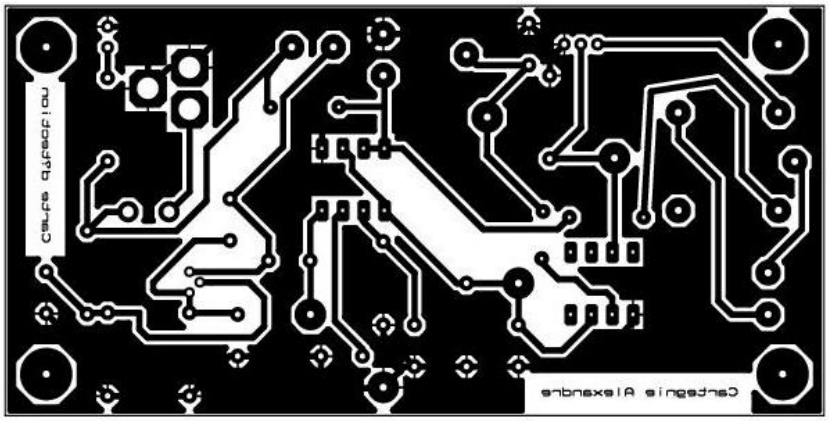
Suite au schéma de principe j'ai donc pu commencer le routage de la carte sous ARES

.

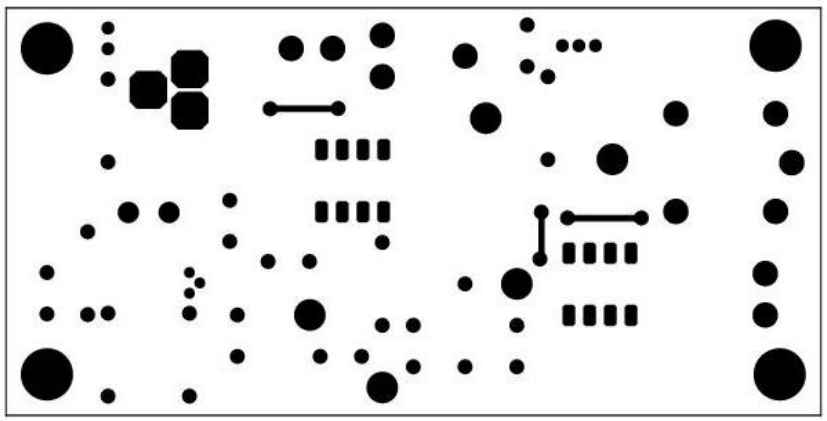


16.5.1 Fabrication

Une fois le routage effectué, j'ai donc imprimé le calque pour effectuer la fabrication de la carte au Lycée.



La carte est désormais prête, il ne reste plus qu'à la percer en s'aidant des fichiers GERBERT de perçage comme ci-dessous



On remarque la présence de 4 types de perçage, 0.8mm, 1, 3, et 4mm.

Enfin voici, ci-dessous le rendu final de la carte, dans cette configuration le LM2907 et le TLC273 ne sont pas placés pour éviter de les endommager pendant la première phase de tests.

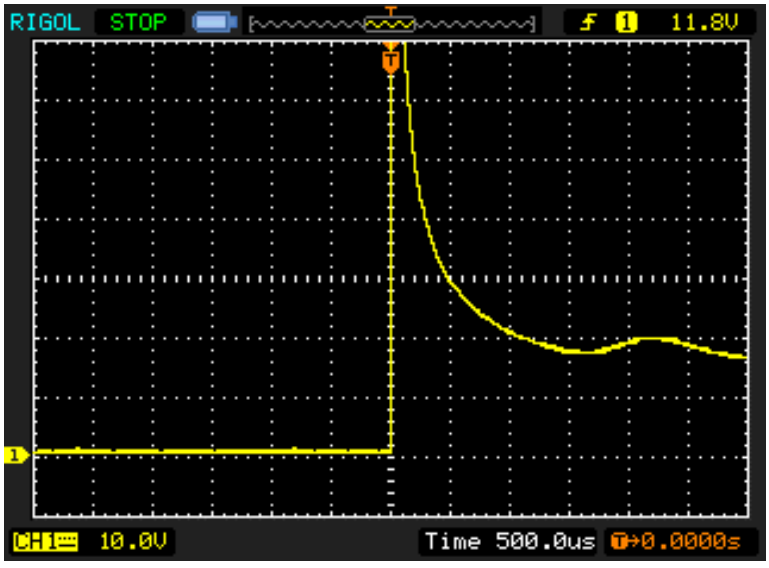
16.5.2 Coût de la carte

Grâce à la référence des composants et quelques recherches sur le site de notre fournisseur RadioSpares j'ai pu établir un cout estimatif pour la carte de détection se situant aux alentours des 10 euros. Ce qui reste relativement accessible.

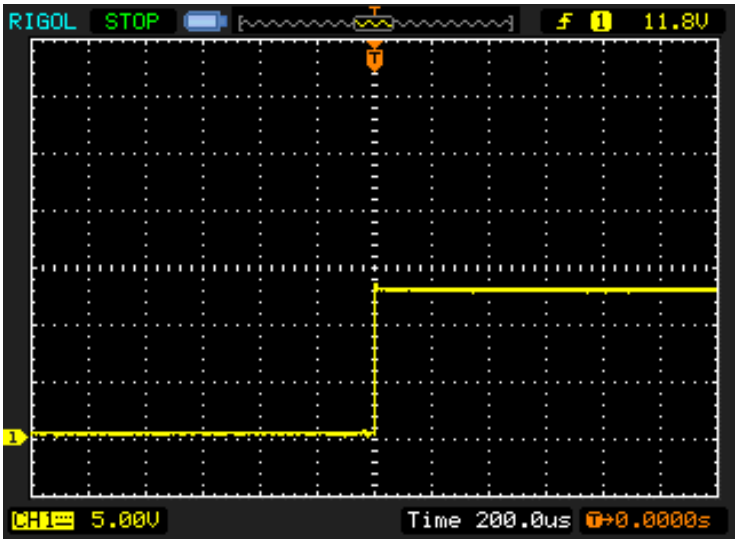
(cf annexe 81)

17 Tests de la carte finale

Lors de la phase de test, alors que la carte fonctionnait, le comparateur TLC273 dysfonctionnait régulièrement, en recherchant la documentation du relais, nous avons remarqué qu'il n'y avait pas de diode de roue libre, ce qui à chaque commutation du relais causait une surtension aux bornes du comparateur. J'ai donc retiré celui-ci pour pouvoir piloter le transistor grâce à un générateur de fréquence, en se plaçant sur la broche d'alimentation du comparateur avec un oscilloscope j'ai pu constater la cause de ce dysfonctionnement majeur



Ce pique de tension causait le dysfonctionnement de la carte, le défaut est corrigé grâce à une diode Zenner placée aux bornes de la bobine du relais, ce qui permet d'évacuer l'excédent de tension progressivement.



18 Améliorations possibles

En se renseignant sur les éléments constituants la plupart des bornes de stationnement existantes, quelques innovations importantes sont apparues :

La société MagSys utilise un module radio autonome pour détecter un véhicule et renvoyer l'information à la borne (comme montré ci-contre), ce qui évite un surcoût lors de la pose du matériel.

19 Annexes

19.1 Programme Néopixels

19.1.1 Programme inclus dans l'ATtiny85

```
#include "TinyWireS.h"                  // on inclus la lib I2C Slave pour
ATTiny
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif

#define I2C_SLAVE_ADDR  0x43            // On note l'adresse I2C
#define PIN 1
#define NUMPIXELS 61

byte byteRcvd = 0;

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB +
NEO_KHZ800);

// IMPORTANT: To reduce NeoPixel burnout risk, add 1000 uF capacitor across
// pixel power leads, add 300 - 500 Ohm resistor on first pixel's data
input
// and minimize distance between Arduino and first pixel.  Avoid connecting
// on a live circuit...if you must, connect GND first.

void setup() {
  // This is for Trinket 5V 16MHz, you can remove these three lines if you
are not using a Trinket
#if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
#endif
  // End of trinket special code
  TinyWireS.begin(I2C_SLAVE_ADDR);      // on rejoint le bus avec une
adresse d'esclave (similaire a la lib wire)

  pixels.begin();
  pixels.show(); // Initialize all pixels to 'off'
}

void vert()
{
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(0, 50, 0));
  }
  pixels.show();
}

void blink_vert()
{
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(0, 50, 0));
  }
```

```
  pixels.show();
  delay(500);
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(0, 0, 0));
  }
  pixels.show();
  delay(500);
}

void rouge()
{
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(50, 0, 0));
  }
  pixels.show();
}

void blink_rouge()
{
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(50, 0, 0));
  }
  pixels.show();
  delay(500);
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(0, 0, 0));
  }
  pixels.show();
  delay(500);
}

void bleu()
{
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(0, 0, 50));
  }
  pixels.show();
}

void blink_bleu()
{
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(0, 0, 50));
  }
  pixels.show();
  delay(500);
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(0, 0, 0));
  }
  pixels.show();
```

```
  delay(500);
}

void jaune()
{
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(50, 50, 0));
  }
  pixels.show();
}

void blink_jaune()
{
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(50, 50, 0));
  }
  pixels.show();
  delay(500);
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(0, 0, 0));
  }
  pixels.show();
  delay(500);
}

void eteint()
{
  for (int i = 0; i < NUMPIXELS; i++)
  {
    pixels.setPixelColor(i, pixels.Color(0, 0, 0));
  }
  pixels.show();
}

void loop()
{
  if (TinyWireS.available()) // si on revoit quelque chose sur le bus I2C
  {
    byteRcvd = TinyWireS.receive(); // on l'enregistre
  }
  switch (byteRcvd)
  {
    case 0x36:
      byteRcvd = 0x00;
      eteint();
      break;

    case 0x31:
      byteRcvd = 0x00;
      rouge();
      break;

    case 0x33:
      blink_rouge();
```

```
      break;

    case 0x30:
      byteRcvd = 0x00;
      vert();
      break;

    case 0x32:
      blink_vert();
      break;

    case 0x48:
      byteRcvd = 0x00;
      bleu();
      break;

    case 0x35:
      blink_bleu();
      break;

    case 0x49:
      byteRcvd = 0x00;
      jaune();
      break;

    case 0x34:
      blink_jaune();
      break;
  }
}
```

19.2 Coût de la carte

	Détection véhicule projet URBACO							Client :				
Repère	Désignation du matériel	Valeur (Référence)	Tol.±	Equivalent	Fabricant @ Fournisseur	Boîtier	Recommandation câblage	Qté	Condition	ref rapide "RS" (FA)	Prix U HT	Prix T HT
R1	Résistance couche metall 1/4W	2k Ohms	1%					1			0,220	0,220
R2	Résistance couche metall 1/4W	10k Ohms	1%	5%				1			0,220	0,220
R3	Résistance couche metall 1/4W	10K Ohms	1%	5%				1			0,220	0,220
R4	Résistance couche metall 1/4W	2,2K Ohms	1%					1			0,220	
R5	Résistance couche metall 1/4W	100K Ohms	1%					1			0,220	
R6	Résistance couche metall 1/4W	10K Ohms	1%					1			0,220	
R7	Résistance couche metall 1/4W	100 K Ohms	1%					1			0,220	0,220
LM2907	Convertisseur fréquence tension							1				0,000
RV1	Résistance variable 10 K Ohms	10 K Ohms						1			0,220	0,220
RL3	Relais 1RT	T7SS5E6-12						1			2,350	2,350
C1,	Condensateur Céramique	100 nF						1			0,142	0,142
C2; C3	Condensateur epoxy	15 nF, 50 V						2			2,400	4,800
C4	Condensateur Céramique	10 nF						1			0,142	0,142
C5	Condensateur Céramique	47pF 50V						1			0,142	
TLC271	Amplificateur opérationnel							1			0,900	0,900
C6	Condensateur chimique	1 uF						1			0,072	0,072
C7	Condensateur céramique	100 nF						1			0,142	0,142
C8	Condensatteur chimique	1 uF						1			0,072	0,072
D1	Diode témoin orange 3mm							1			0,100	
Jack Alim	jack femmelle alimentation 12 V							1			3,290	
										TOTAL HT FEUILLE		9,720

19.3 Fiche de recettes

FICHE RECETTE, QUALIFICATION DU SYSTEME

Qualification du système :	**Borne Arrêt Minute**	Code de la campagne de test :	xxx
Etudiant :	CARTEGNIE Alexandre	Date :	23/05/2016

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Carte de détection véhicule
Objectif du scénario :	Montrer au client le bon fonctionnement de la carte de détection et voir avec le client les points qu'il ne comprend pas.

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

- Une alimentation
- Plaque métallique
- Carte de détection
- Spire
- Tournevis plat
- Multimètre
-

EXECUTION DU TEST

Description courte :	Placer et connecter la bobine sur le bornier 1, connecter la carte au secteur, poser la plaque sur la bobine, vérifier la tension, enlever la plaque et vérifier une seconde fois la tension, regeler le seuil entre les deux tensions lues.
Résultats attendus :	Le relais doit commuter à l'approche de la plaque. Le niveau logique récupéré est un niveau logique « bas ».

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

FICHE RECETTE, QUALIFICATION DU SYSTEME

Qualification du système :	**Borne Arrêt Minute**	Code de la campagne de test :	xxx
Etudiant :	CARTEGNIE Alexandre	Date :	23/05/2016

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Néopixels
Objectif du scénario :	Montrer au client le bon fonctionnement de la partie Néopixels et voir avec le client les points qu'il ne comprend pas.

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

- Une alimentation
- Une carte Raspberry pi
- Le programme exécutable
- Shield principal avec ATtiny85
- Disque Néopixels

EXECUTION DU TEST

Description courte :	Connecter le shield à la carte Rpi, Connecter le LED au shield, lancer le programme éxécutable.
Résultats attendus :	Les Néopixels doivent réagir en fonction de l'ordre donné par l'éxécutable QT

BILAN

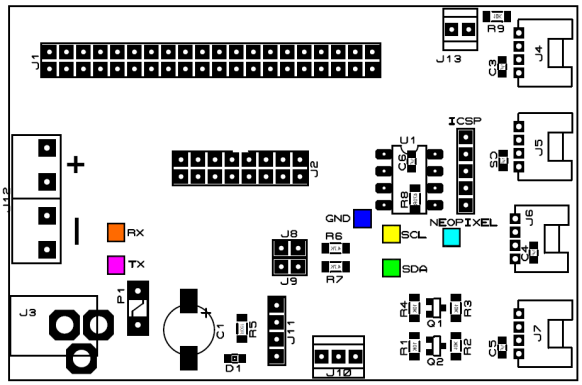
REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

	**Epreuve E6.2 Projet URBACO Raspberry PI 2016**	



Projet URBACO RPI

BTS Systèmes Numériques

Damien GALLEGO

Candidat : Damien GALLEGO

20 Le projet URBACO

20.1 La société URBACO

La société URBACO a été créée en 1986 à ENTRAIGUES-SUR-LA-SORGUE.
URBACO est aujourd'hui un acteur incontournable en Europe sur les marchés des systèmes de contrôle d'accès par bornes escamotables et du mobilier urbain.
En 2004, URBACO a rejoint le Groupe CAME, Leader Mondial sur le secteur des automatismes de portails, barrières et portes piétonnes afin d'offrir une gamme encore plus étendue de produits et de services aux marchés des collectivités locales, de l'industrie, des particuliers et des organismes de sécurité : mobilier urbain, bornes fixes, amovibles, escamotables, automatiques, portails, barrières, portes piétonnes, domotique et contrôle des accès.

20.2 URBACO Borne arrêt minute

20.2.1 Préambule

Le projet borne arrêt minute consiste en une borne statique non escamotable. Cette borne sera placée à l'abord d'une place de stationnement prisée par les citadins (ex : près d'un magasin précis).
La borne que nous sommes en train de revisiter existe déjà chez URBACO mais fonctionne avec un Automate Programmable et celui-ci est un appareil qui coute très cher, donc grâce à nos recherches il est peut être possible de créer quelque chose de similaire à un tarif moins élevé pour l'entreprise et aussi pour les Mairies qui l'achètent.

20.2.2 Borne similaire

Borne automatique à CAVAILLON de la société SIGNATURE :

- Détection de deux véhicules
- Flèches blanche s'éclairent en vert si un véhicule est présent sur la place de stationnement.
- Fleche verte clignotante quand le temps de stationnement approche de la fin du temps.
- Flash orange permettant de signaler le véhicule ayant dépassé le temps de stationnement autorisé, et donc verbalisable par un agent de police.

20.2.3 Que fait notre borne ?

Borne automatique projet URBACO

- Détection d'un véhicule
- Image du temps restant grâce aux LED placées en haut (vert fixe => temps ok, rouge fixe => temps approchant de la fin, rouge clignotant => temps de stationnement dépassé).
- Décomptage du temps restant grâce aux matrices LED.
- Affichage du type de stationnement (lambda, livreur, handicape).
- Interfaçage de l'utilisateur ayant un badge, lui permettant d'avoir son type de stationnement.
- Afficheur LCD servant à afficher des messages divers (ex : 14 juillet fête nationale).

21 Les tâches m'ayant été attribué

- Afficher des informations de type « information municipales ».
- Intégrer les différents constituants matériels et logiciels (partie EC) du projet.
- Travail conjoint avec les étudiants 2 et 3 (EC) du projet URBACO Arduino, et adaptations logicielles de leurs travaux au projet URBACO Raspberry PI.

21.1 Affichage des informations de type « infos municipales »

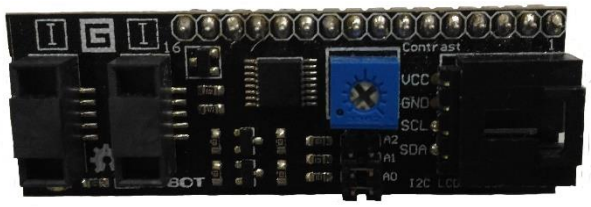
21.1.1 Description du matériel

Pour afficher les informations diverses on m'a proposé d'utiliser un afficheur LCD 4x20 couplé à un module I2C.

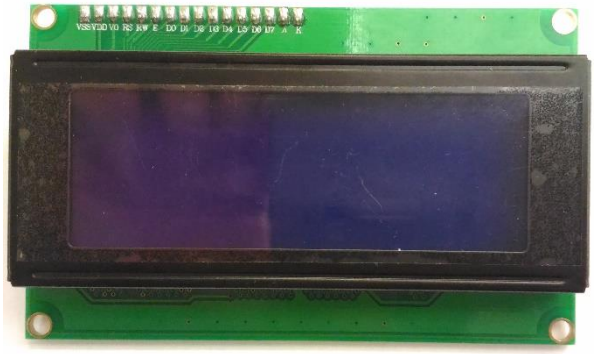
- L'afficheur LCD (Liquid Crystal Display) est un HD44780U.
- Le module I2C est associé à cette afficheur, IIC LCD Backpack basé sur un PCA8574 (Remote 8-bit I/O expander for I2C-bus with interrupt).

21.2 Photos afficheur et module I2C

Module I2C basé sur un PCA8574



Afficheur LCD HD44780U



21.3 Les recherches et essais

21.3.1 Les recherches

Une fois que j'ai récupéré l'afficheur 4x20 I2C je suis allé sur le site de DFROBOT, le fabricant.

Le nom du module est I2C/TWI LCD2004. Il inclut l'afficheur et son interface I2C.

Les caractéristiques de ce module I2C/TWI LCD2004 indiquent la disponibilité d'une librairie Arduino.

Je l'ai donc téléchargée, pour vérifier le bon fonctionnement de l'afficheur.

Par la suite, j'ai commencé à chercher une librairie fonctionnant sur Raspberry PI. Après plusieurs essais j'ai trouvé une librairie qui fonctionnait.

Pour faire fonctionner cette librairie j'ai suivi un tutoriel sur le site suivant :
http://hardware-libre.fr/2014/03/raspberry-pi-utiliser-un-lcd-4x20/

21.3.2 Suivi du tutoriel

Le tutoriel sur ce site propose :

- de configurer le bus I2C de la Raspberry PI,
- d'installer des outils pour scanner le bus I2C
- de créer trois fichiers python :
    1. **`i2c_lib.py`** : implémentation de la communication i2c en python
    2. **`lcddriver.py`** : librairie de pilotage du LCD s'appuyant sur le driver précédent
    3. **`lcdExample.py`** : exemple de mise en œuvre de la libraire

Les étapes du tutoriel sont les suivantes :

- *Création des fichiers et édition*

Dans un shell, on crée un dossier dans le répertoire personnel.

- **`mkdir afficheur_4x20_I2C`**

On entre dans le dossier créé

- **`cd afficheur_4x20_I2C`**

On crée le fichier **`i2c_lib.py`** et on l'édite :

- **`nano i2c_lib.py`**

Copier le contenu de ce fichier proposé sur le tutoriel grâce à la commande *Ctrl+c* et le coller dans le shell grâce à la commande *Ctrl+Shift+v*.

Faire de même pour les deux autres fichiers.

- **`nano lcddriver.py`**
- **`nano lcdExemple.py`**

Pour le fichier **`lcddriver.py,`** au moment de l'édition, vérifier que l'adresse du composant I2C (variable ADDRESS) est correcte et la corriger dans le cas contraire.

- Exécution du programme d'exemple
  - **`sudo python lcdExemple.py`**

Si votre configuration est correcte vous devriez avoir ceci :

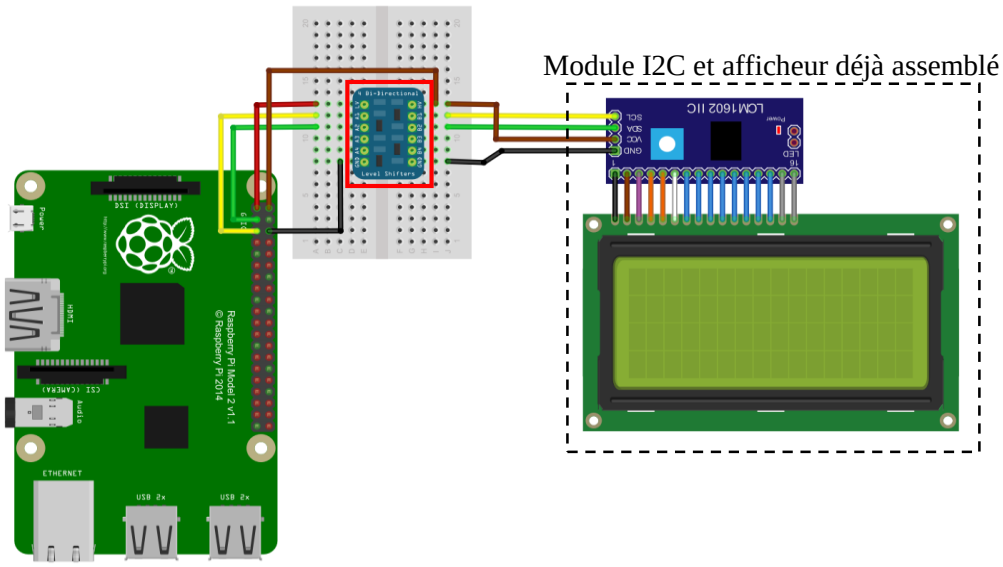
```
Hello world !
Je suis
un
Raspberry Pi !
```

21.3.3 Câblage effectué pour tester l'afficheur I2C

Comme je l'ai dit précédemment, le premier test que j'ai fait pour cet afficheur a été de le tester sur une Arduino UNO qui fonctionne en 5V.

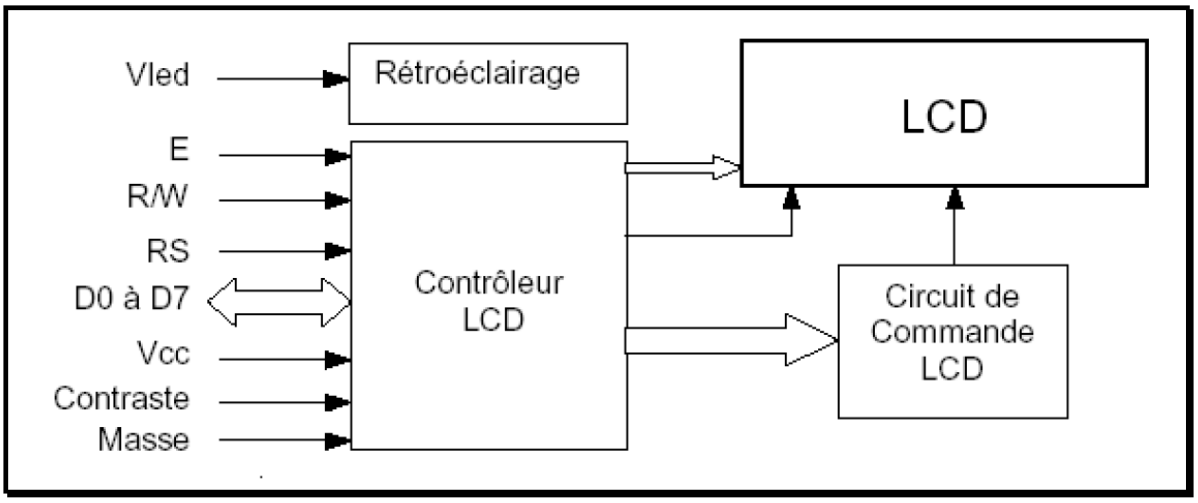
En regardant la documentation de l'afficheur, je vois qu'il peut fonctionner pour une tension de 2.7V à 5.5V et en regardant la notice du module I2C on peut lire **`supply voltage : 5V`**. Et comme l'afficheur LCD est relié à ce module I2C, le seul moyen de faire fonctionner le tout est de le faire fonctionner en 5V.

La Raspberry délivre seulement 3.3V au niveau de ses broches de données SDA et SCL, il faut que j'utilise un level shifter (encadré en rouge sur l'image ci-dessous) pour faire une adaptation de tension car l'afficheur est alimenté en 5V.



21.4 Afficheur HD44780U

21.4.1 Fonctionnement de l'afficheur



- Rôle des différentes broches de l'afficheur LCD

Vcc, Masse : alimentation de l'afficheur LCD. Il s'alimente en **0V – 5V**.

Contraste : entrée permettant de régler le contraste de l'afficheur LCD. Il faut appliquer une tension continue réglable (entre **0V** et **5V**) à l'aide d'un potentiomètre.

Vled : différence de potentiel permettant de commander le rétro éclairage.

E : entrée de validation (**ENABLE**), elle permet de valider les données sur un **front descendant**. Lorsque **E = 0** alors le bus de données est à l'état haute impédance.

RS : **Register Select** cette entrée permet d'indiquer à l'afficheur si l'on souhaite réaliser une commande (**RS = 0**) par des instructions spécifique ou écrire une donnée (envoie d'un code d'un caractère à afficher) sur le bus (**RS = 1**).

R/W : entrée de lecture (**R/W = 1**) et d'écriture (**R/W = 0**). Lorsqu'on commande l'afficheur LCD il faut se placer en écriture.

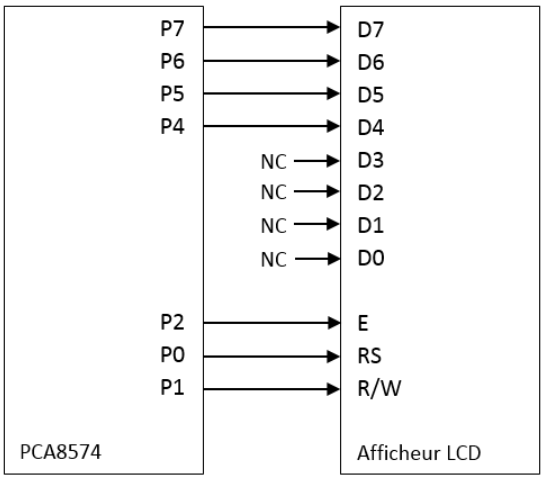
D7 à D0 : bus de données **bidirectionnel**, il permet de transférer les instructions ou les données à l'afficheur.

21.4.2 Câblage entre l'afficheur et le module I2C

Un afficheur LCD peut fonctionner de deux manières différentes.

- En mode 8 bits : câblage de D7 à D0
- En mode 4 bits : câblage de D7 à D4

Câblage entre le PCA8574 et l'afficheur LCD

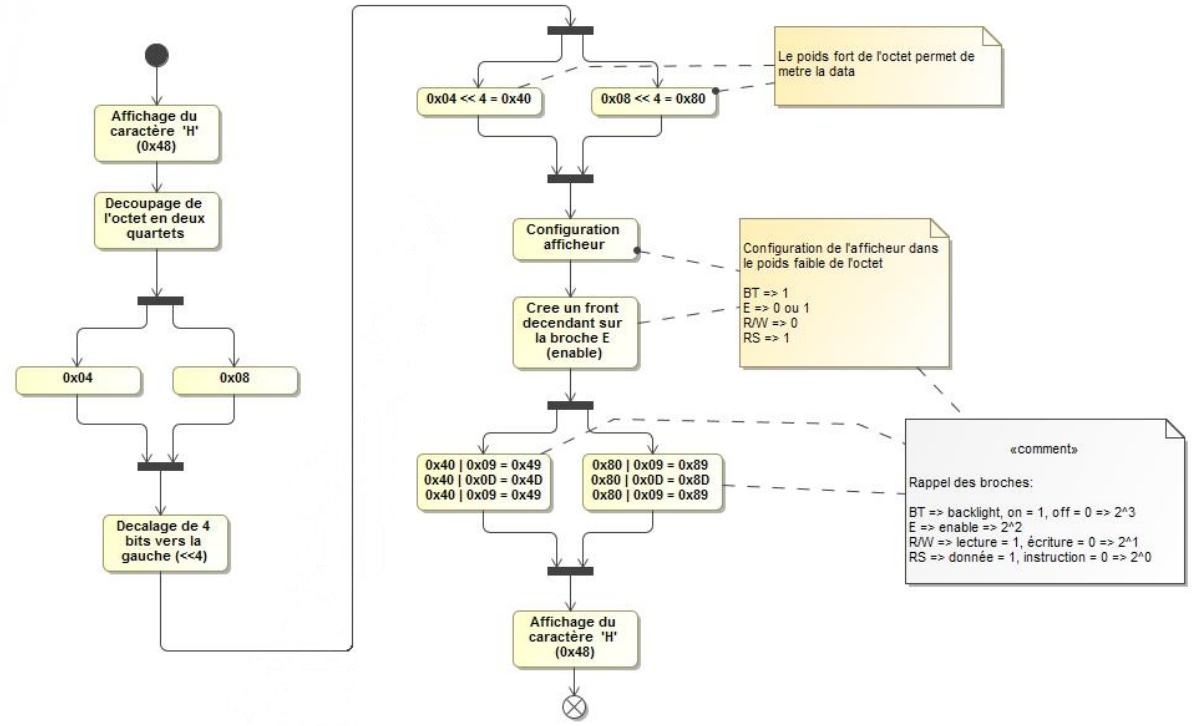


Le PCA8574 est un expander de port de 8 bits, son format est : TSSOP16 (plastic thin shrink small outline package).

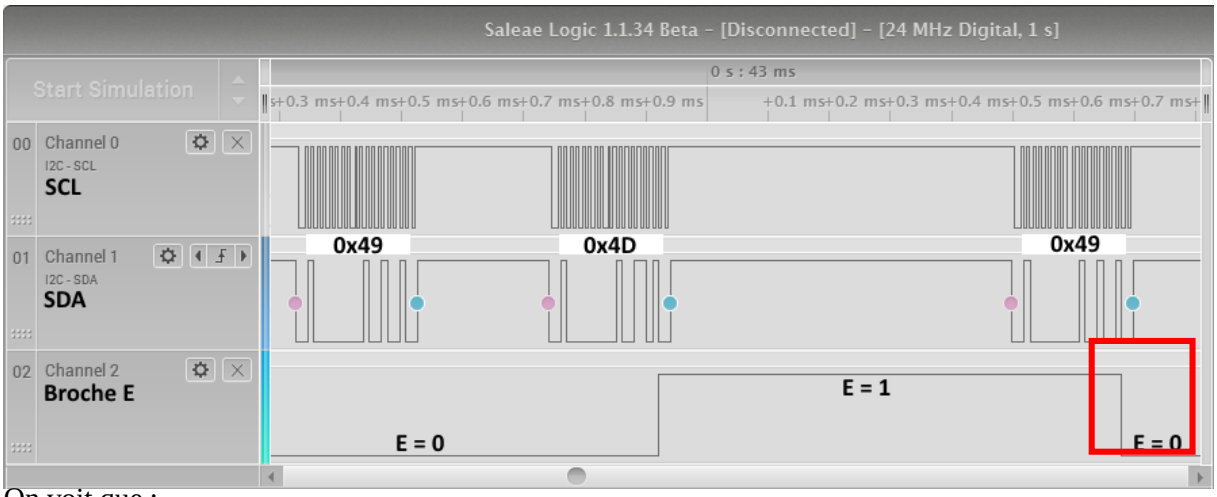
Il a 8 ports configurables en I/O, P0 à P7.

Le câblage entre le PCA8574 et le LCD est en mode 4 bits car il faut seulement 7 broches pour commander le LCD.

21.4.3 Affichage d'un caractère sur l'afficheur LCD I2C



Sur cette capture de trame on peut voir le premier octet **0x4x**

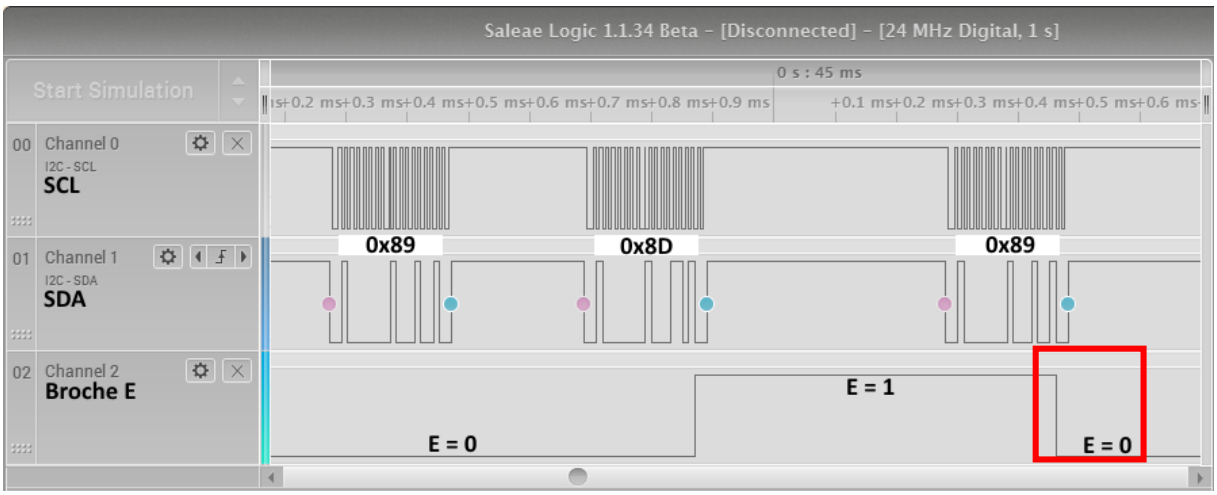


On voit que :

- 0x49 => E = 0
- 0x4D => E = 1 **E => entrée de validation, active sur un front descendant.**
- 0x49 => E = 0

Le front descendant est bien créé et donc l'afficheur peut enregistre le quartet 0x40.

Suite de la capture ou on peut voir le second octet envoyé **0x8x**



On voit que:

- 0x89 => E = 0
- 0x8D => E = 1 **E => entrée de validation, active sur un front descendant.**
- 0x89 => E = 0

Le front descendant est bien crée et donc l'afficheur peut enregistre le quartet 0x80.

L'afficheur configuré au préalable en mode 4 bit se charge du reste du travail et rassemble les deux octets reçus et les assembles pour reconstruire l'octet 0x48 => 'H'.

21.5 Raspberry, le bus I2C

21.5.1 Généralité sur le bus I2C

Le bus I2C (signifie : Inter-Integrated Circuit) crée par Philips/NXP est un bus de communication synchrone half-duplex.

Simplicité de câblage 3 fils :

- SDA (Serial Data Line) : ligne de données bidirectionnelle
- SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle
- GND masse commune

Les 2 lignes SDA et SCL sont tirées au niveau de tension V_{DD} à travers des résistances de pull-up car la sortie des différents composants se fait sur un transistor a collecteur ou drain ouvert.

Le nombre maximal d'équipements est limité par :

- le nombre d'adresses disponibles, 7 bits pour l'adresse et un bit pour définir si on écrit ou si on lit, soit 128 périphériques, en réalité moins de 128 périphériques car il y a des adresse réservées : les adresses pour les composants vont de 0x08 a 0x77
- une capacitance maximale du bus de 400pF.

Exemple d'adresse réservée :

- 0000 000**1** (utilisé pour synchroniser les périphériques lents avec les périphériques rapides)
- 0000 0110 (reset, remet tous les registres des circuits connecté à leur état initial)

Les équipements sont donc câblés sur le bus par le principe du « ET câblé », ce qui veut dire qu'en cas d'émission simultanée de deux équipements, la valeur 0 écrase la valeur 1.

On dit donc :

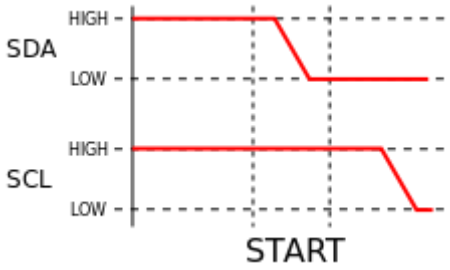
- que l'état logique « 0 » ou « LOW » est l'état « dominant »,
- que l'état logique « 1 » ou « HIGH » est l'état « récessif ».

Les différentes vitesses du bus I2C :

- « Standard Mode » ≤ 100KHz
- « Fast Mode » ≤ 400KHz
- « High-speed Mode » ≤ 3.4MHz

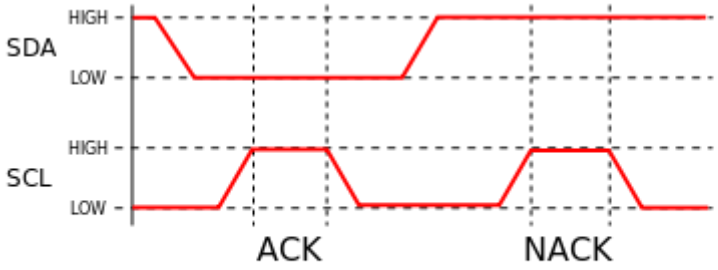
Les conditions des lignes SCL et SDA pour :

- **START**



La condition de « **START** » est caractérisée par le passage de la ligne SDA du niveau « HIGH » au niveau « LOW » pendant que la ligne « SCL » est maintenue au niveau « HIGH ».

- **ACK, NAK**



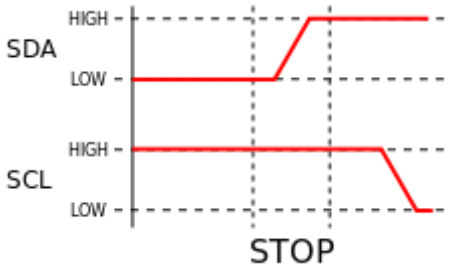
« **ACK** », en forçant la ligne SDA au niveau « LOW », pour signaler la bonne réception de l'octet, équivalent à un bit à 0

« **NACK** », en laissant la ligne SDA au niveau « HIGH », pour signaler un défaut dans la réception de l'octet, équivalent à un bit à 1

- **RESTART**

La condition de « **RESTART** » est similaire à la condition de START, à cela près que la ligne SCL doit d'abord passer du niveau « LOW » au niveau « HIGH ».

- **STOP**



La condition de « **STOP** » est caractérisée par le passage de la ligne SDA du niveau « LOW » au niveau « HIGH » pendant que la ligne SCL est maintenue au niveau « HIGH ».

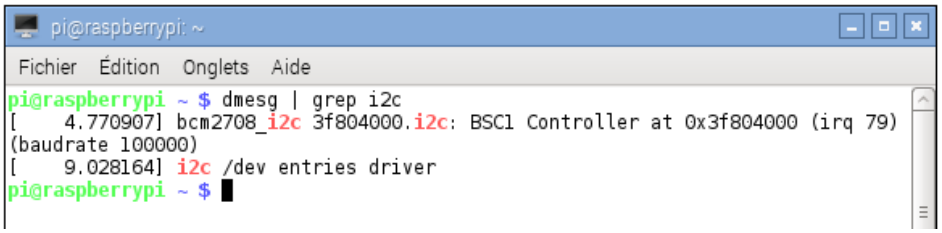
21.6 Bus I2c Raspberry PI

Connaitre la vitesse du bus I2C de la Raspberry, il existe plusieurs façon de la connaitre

On peut taper la commande suivante dans un Shell

- **`dmesg | grep i2c`** (permet de visualiser les messages du noyau se rapportant au bus I2C)

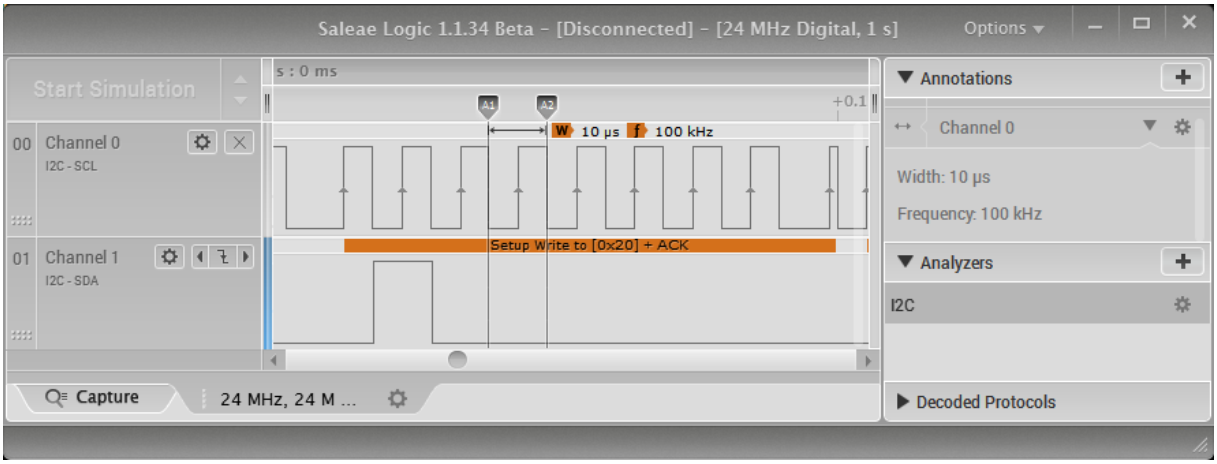
Le retour de cette commande ci-dessous



Sur la capture d'écran de cette commande on peut lire (**`baudrate 100000`**), ce qui correspond à la fréquence de la clock soit 100KHz.

Ou bien on peut aussi utiliser un oscilloscope ou un analyseur logique.

J'ai choisi de le faire avec un analyseur logique et de placer des curseurs sur la trame de la broche SCL.



Sur cette capture de trame on peut voir qu'il y a deux curseurs A1 et A2 et qu'ils sont placés sur une période de la clock. On peut lire Width : 10 µs et Frequency : 100KHz.

Pour vérifier ce qu'annonce le logiciel Logic on peut faire $\frac{1}{10\times10^{-6}} = 100KHz$

Ce qui vérifie bien la sortie de la commande précédente.

21.6.1 Configuration du bus I2C (Raspbian Wheezy)

1er étape : installer une librairie python et des outils I2C

- **sudo apt-get install python-smbus**
- **sudo apt-get install i2c-tools**

2ème étape : éditer le fichier **modules**

- **sudo nano /etc/modules**

3ème étape : ajouter les deux lignes suivantes dans le fichier **modules**

- **i2c-bcm2708** (module qui gère le contrôleur I2C)
- **i2c-dev**

4ème étape : éditer le fichier **raspi-blacklist.conf**

- **sudo nano /etc/modprobe.d/raspi-blacklist.conf**

5ème étape : ajouter un # devant **blacklist i2c-bcm2708** (# permet de mettre en commentaire la ligne)

- **#blacklist i2c-bcm2708**

6ème étape : redémarrer la RPI

- **sudo reboot** (redémarre la Raspberry)

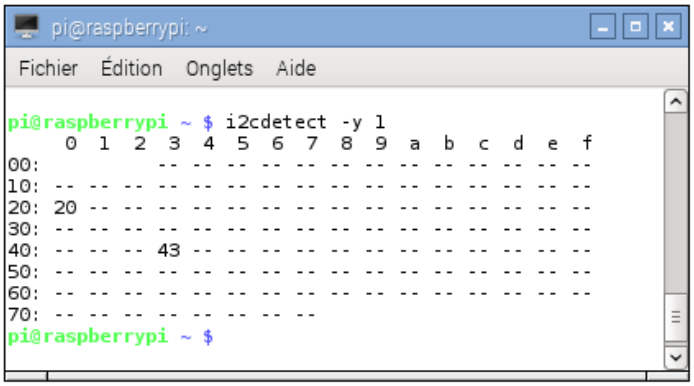
21.6.2 Tester le bus I2C

La commande suivante va permettre de scanner le bus I2C, et si il y a des périphériques I2C de connectés au bus ils n'y apparaîtront seulement que si leur adresse est comprise entre 0x03 et 0x77, commande uniquement pour Raspberry PI 2 et 3. Les adresses inferieures à 0x08 et supérieur à 0x77 sont des adresses qui ont un rôle spécifique (ex : 0x00 => broadcast).

- **i2cdetect -y 1**

La sortie de cette commande =>

Sur la capture d'écran on peut voir l'adresse 0x20 qui correspond à mon afficheur I2C et l'adresse 0x43 qui correspond à l'ATTINY85.



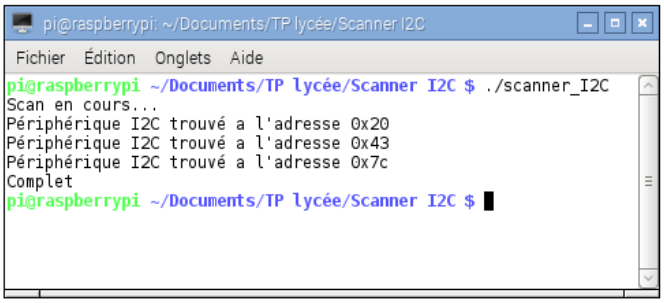
Ou pour scanner le bus I2C je peux utiliser le petit programme C++ que j'ai fait.

1er étape se déplacer dans le répertoire ou se situe l'exécutable (valable sur ma RPI)

- **cd /home/pi/Documents/Tp\ lycée/Scanner\ I2C**

2ème étape : exécuter l'exécutable (annexe page 115)

- **./scanner_I2C**



<= La sortie de cette commande

Sur la capture d'écran on peut voir l'adresse 0x20 et 0x7C qui correspond à mon afficheur I2C et l'adresse 0x43 qui correspond à l'ATTINY85.

21.6.3 Utiliser la bus I2C

Plusieurs solutions sont à notre disposition :

- Les commandes Shell
- La librairie WiringPiI2C
- La librairie Ci2c (Mr SILANUS) similaire à libi2c.c

Pour faire quelques essais j'ai utilisé les commandes Shell car elles sont rapides à utiliser. Par la suite pour mes programmes j'ai utilisé la librairie CI2C voir annexe page 113 car elle est similaire à la libraire Wire d'Arduino.

1er étape : inclure la librairie dans son programme

- `#include "Ci2c.h"`

2ème étape : créer un objet de la classe Ci2c

- **Dans une classe**

- `Ci2c *I2C; (dans le .h)`
- `I2C = new Ci2c('1', lcdAddress); (dans le .cpp)`

- **Dans un programme**

- `Ci2C I2C('1', lcdAddress); (dans .cpp)`

3ème étape : utiliser les méthodes

- `writeToI2C(unsigned char *buffer, int length);`
- `readFromI2C(unsigned char *buffer, int length);`

21.7 Création d'une librairie

Dans les consignes de « base » le code est « prioritairement » en C/C++, c'est donc pour cela que j'ai voulu créer une librairie C++ pour l'afficheur car la librairie que j'ai trouvée est en python.

Pour créer cette librairie je me suis inspiré de la librairie python en reprenant exactement le nom des méthodes. J'ai par la suite retranscrit le code python en C++ orienté objet.

Pour la communication I2C j'ai utilisé les librairies wiringPi et wiringPiI2c.

Pendant les tests, cela fonctionnait mais pas aussi bien qu'on le souhaitait. Ca dysfonctionnait au niveau de l'affichage. Un jour Mr HORTOLLAND (professeur d'électronique) regarde en partie ma librairie et me dit : « Je ne crois pas que la fonction delay(ms) de wiringPi accepte les float ». Je décide donc de remplacer mes `delay(0.1);` par `delay(1);`soit 1 mS, ceci étant fait je recompile et exécute mon programme d'exemple, cette fois ci tout fonctionnait correctement. Peu de temps après, Mr HORTOLLAND me dit qu'il existe une autre fonction `delayMicroseconds`, je remplace donc mes `delay(1)` par `delayMicroseconds(100);` Je test à nouveau est ça fonctionne.

Par la suite j'ai créé d'autres librairies pour d'autres composants en I2C et avec certains composants la librairie wiringPiI2c ne convenait pas, car à chaque fois que l'on veut écrire un octet il y a l'adresse devant. Le composant le comprenait comme une data donc ça n'allait pas. J'ai donc décidé d'utiliser la librairie Ci2c et j'ai ré-écrie ma librairie avec les méthodes de cette librairie et puis je l'ai scanné avec un analyseur logique et là c'était fonctionnel [adresse] [data] [data] … [data].

J'ai donc modifié ma librairie I2C_Afficheur_4x20 en remplacent la librairie wiringPiI2C par Ci2c.

Par la suite j'ai créé une application qui permet de rentrer du texte sur 4 lineEdit et de l'afficher sur l'afficheur et s'il n'y avait rien à écrire, j'éteins l'afficheur. J'ai donc fait évoluer ma librairie en rajoutant deux méthodes :

- `displayOn(); // allume l'afficheur`
- `displayOff(); // efface et éteint l'afficheur`

22 Commande des composants

La recherche de composant s'est faite avec les deux groupes du projet URBACO « Raspberry et Arduino » car dit dans le référentiel du projet il doit y avoir une compatibilité des modules entre les deux groupes du projet URBACO.

Connecteurs commun aux deux projets URBACO

Nous avons convenu que pour le bus I2C nous utiliserons des connecteurs GROVE 4 contacts au pas de 2 mm, pour les LED NEOPIXEL nous utiliserons des connecteurs JST 3 contacts au pas de 2.54 mm, pour le contact TOR de la détection de véhicule nous utiliserons un bornier 2 contacts au pas de 2.54 mm et pour la matrice LED RGB 16x32 nous utiliserons un connecteur HE10 2x8 contacts pour les données et pour l'alimentation de la matrice nous utiliserons des borniers 2x2 contacts au pas de 5.08 mm.

Connecteur seulement sur le projet URBACO Raspberry

Un connecteur 5 contacts avec détrompeur pour la programmation de l'ATTINY85.

Pour le bus UART nous utilisons un connecteur HE14 4 contacts au pas de 2.54mm

Picot de relevé de mesures

Des picots ont été placés sur le circuit imprimé pour permettre de faire des relevés de trames. Détails sur leur emplacement et leurs fonctions en **annexe page 106**.

Connecteur Grove 4 contacts au pas de 2mm
Utilisé pour le bus I2C

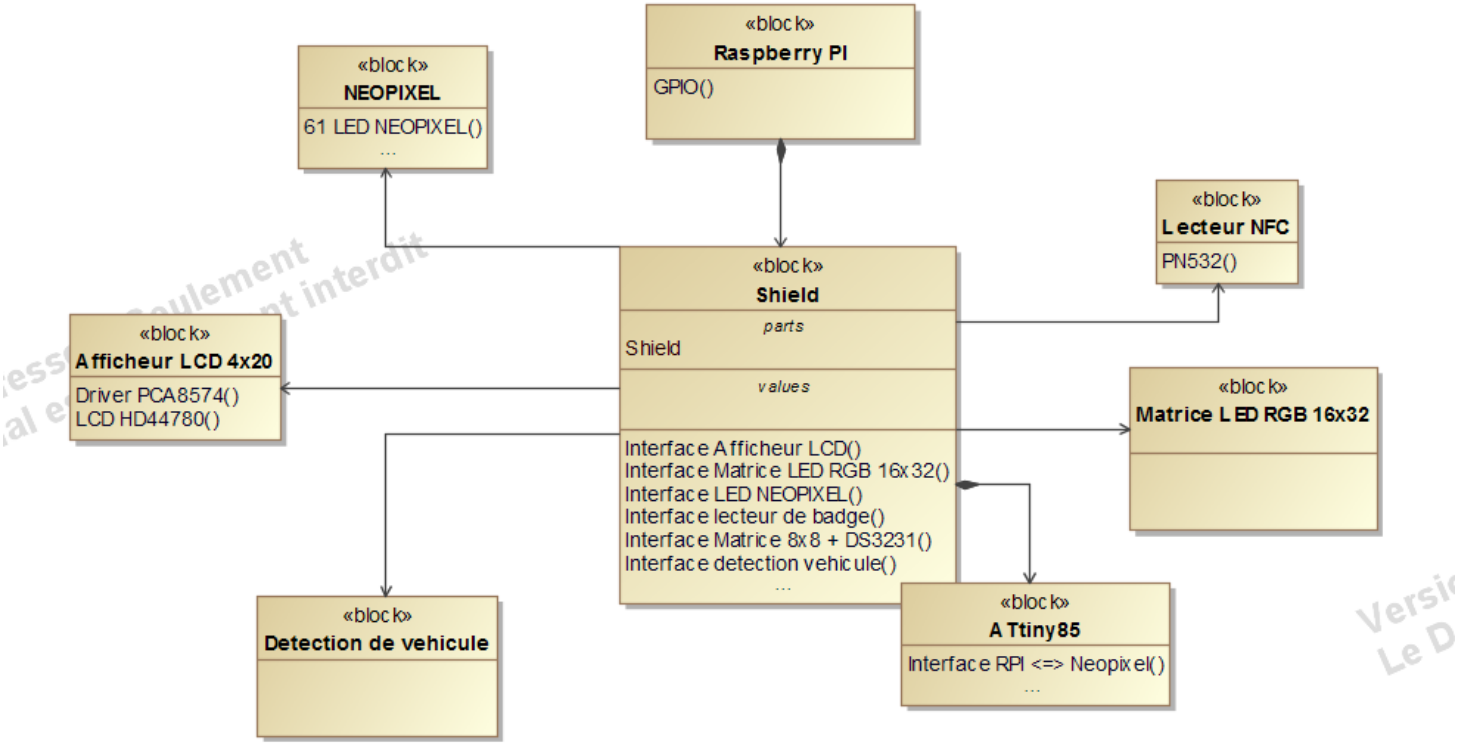
Connecteur Grove 4 contacts au pas de 2.54 mm Utilisé pour le bus UART

Connecteur Grove 4 contacts au pas de 2.54 mm Utilisé pour la matrice

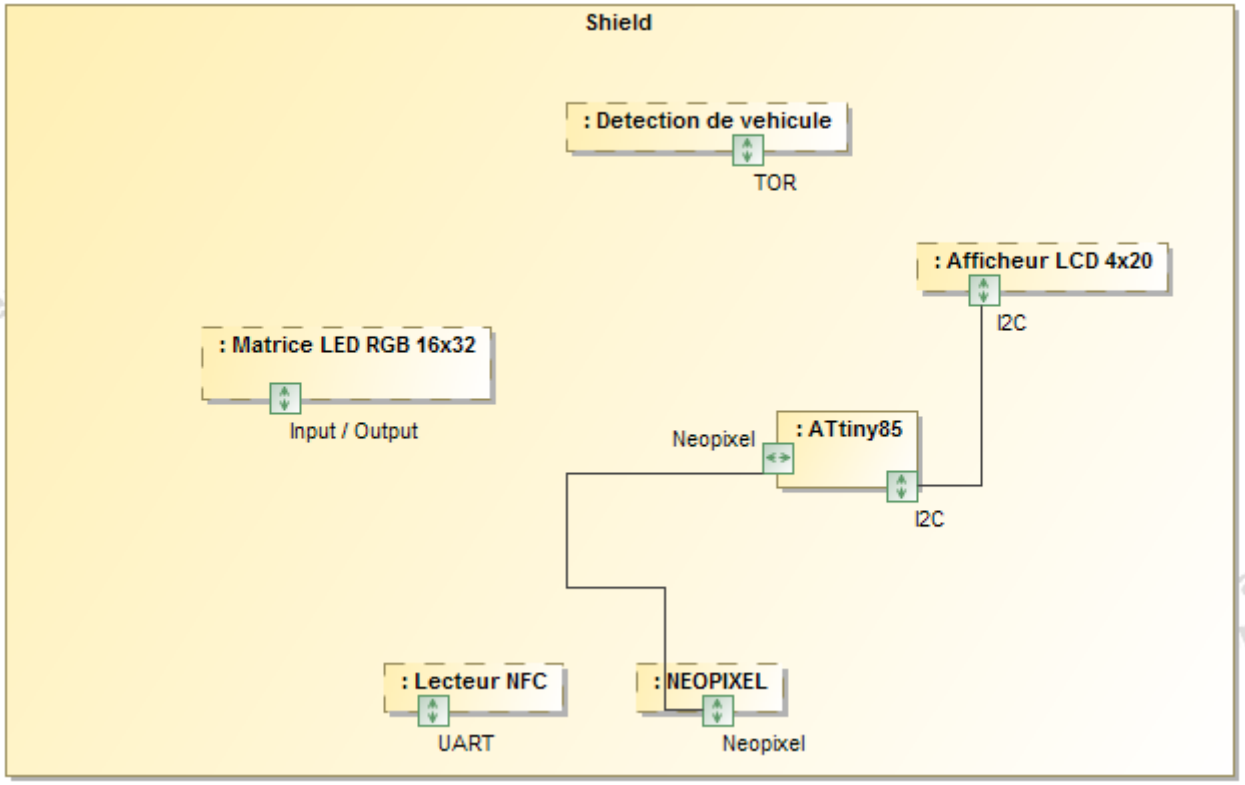
Connecteur Grove 5 contacts au pas de 2.54 mm Utilisé pour la programmation de l'ATtinny85.

23 Fabrication

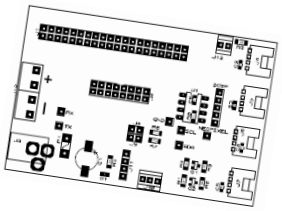
23.1 Diagramme de block



23.2 Diagramme de bloc interne



23.3 En générale



Une de mes grandes taches dans ce projet est la fabrication d'un shield permettant le regroupement des modules du projet Raspberry mais aussi ceux d'Arduino.

Les différents modules qui doivent y être connectés

Projet URBACO **Raspberry**

- L'afficheur 4x20 (I2C)
- Le lecteur NFC (UART)
- L'ATTINY85 (I2C, integer au shield)
- La matrice LED RGB 16x32
- Détection de véhicule (TOR)

Projet URBACO **Arduino**

- La carte d'Amélie (I2C, Comprenant les 5 matrices 8x8 et une DS3231)

23.4 Contrainte due liées au projet

Plusieurs bus de données (I2C, UART) donc des connecteurs différents pour les différencier.

Compatibilités des connecteurs entre le projet Raspberry et Arduino

Dimensionner les pistes en fonction du courant les traversant.

23.5 Schéma ISIS

Faire un schéma le plus claire possible, si possible en découpant les zones. Création des symboles et empreintes ISIS et ARES n'étant pas dans la base de données des logiciels.

23.6 Typon ARES

Pour la création du typon je me suis imposé que la taille de mon circuit imprimé devait être égale à la taille de la Raspberry.

Il faut faire attention à ne pas se tromper entre la couche TOP et BOTTOM, mettre des trous traversant assez grand pour pouvoir les souder par la suite.

23.7 Quelques règles pour le soudage

Une fois que le circuit imprimé a été gravé il faut vérifier qu'il n'y a pas des connexions non souhaitées.

Pour l'ordre de soudage des composants il faut commencer par les composants les plus petits et s'ils sont en CMS il faudra utiliser du flux. Le flux permet de préparer la surface avant de la souder.

Dans les deux tableaux suivant se trouve une colonne « Ordre soudage », elle indique tous simplement quels sont les composants à souder en premier et donc lesquelles sont les plus petits.

23.8 Listes des composants à souder

23.8.1 Composants face du dessus

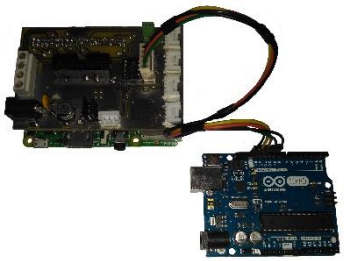
Repère	Désignation	Diamètre de pressage	Ordre soudage
J2	Connecteur male HE10 2 x 8 2.54mm	1 mm	6
J3	Jack alim Ø int : 2.1mm Ø ext : 5.5mm		6
J4, J5, J6, J7	Connecteurs GROVE 4 contacts 2mm		5
ICSP	Connecteur CI 5 contacts male 2.54mm	1 mm	9
J10	Bornier à vis 3 contacts au pas de 2.54mm		7
J11	Connecteur HE14 4 contacts au pas de 2.54mm	1 mm	6
J12	Bornier à vis 4 contacts au pas de 5.08mm	1.2 mm	8
J13	Bornier à vis 2 contacts au pas de 2.54mm		7
P1	Polyswitch Im : 1.6A Id : 3.2A	0.8 mm	
R5	Résistance 150 Ohm boitier 1206	CMS	3
D1	Led verte Vd : 3.6V boitier 0805	CMS	2
C1	Condensateur électrolytique polarise 100µF 16V	CMS	4
U1	Support PDIP 8	0.8 mm	5

23.9 Composants face du dessous

Repère	Désignation	Diamètre de pressage	Ordre soudage
R1, R2, R3, R4, R9	Résistances 10K Ohm boitier 1206	CMS	3
R6, R7	Résistances 4.7K Ohm boitier 1206	CMS	3
R8	Résistance 470 Ohm boitier 1206	CMS	3
C2, C3, C4, C5, C6	Condensateurs non polariser 1µF 16V boitier 0805	CMS	2
J1	Connecteur DIL 2 x 20 au pas de 2.54mm	0.8 mm	6
Q1, Q2	Transistors MOSFET canal N	CMS	1

24 Utilisation du Shield

24.1 Programmation de l'ATtiny85



Sur le shield on a intégré un ATTINY85 pour pouvoir faire fonctionner les LED NEOPIXEL.

L'ATTINY85 sera donc un esclave I2C pilotant les LED NEOPIXEL.

Pour éviter de devoir enlever et remettre l'ATTINY85 à chaque fois que l'on veut changer de code j'ai mis un connecteur 5 broches pour le programmer.

La programmation de l'ATTINY85 se fait par ICSP à l'aide d'une Arduino UNO ayant comme code « ArduinoISP » disponible dans les exemples de l'IDE Arduino version minimum 1.6.

Pour pouvoir programmer l'ATTINY85 il faut qu'il soit disponible dans l'onglet « outils » « type de carte » et pour cela vous pouvez suivre le tutoriel suivant : http://highlowtech.org/?p=1695

Une fois fait, pour le programmer un câble a été fait pour pouvoir connecter l'Arduino UNO au connecteur de programmation sur la Raspberry.

- Coté Raspberry

Un connecteur avec un détrompeur qui empêche les problèmes de connexion.

- Coté Arduino

5 conducteurs de couleurs différentes disponibles
- Noir => GND
- Jaune => broche 13 => SCK
- Orandel => broche 12 => MISO
- Vert => broche 11 => MOSI
- Marron => broche 10 => SS

24.2 Utilisation du câble pour l'afficheur LCD 4x20 (I2C)

Un câble a été fabriqué spécialement pour connecter l'afficheur LCD 4x20 I2C au Shield.

Ce câble comporte d'un côté un connecteur GROVE 4 contacts au pas de 2 mm et à l'autre extrémité un connecteur MOLEX 4 contacts au pas de 2.54 mm.

Conducteur jaune => SCL

Conducteur blanc => SDA

Conducteur rouge => 5V

Conducteur noir => GND

24.3 Utilisation du câble pour le lecteur NFC (UART)

Un câble a été spécialement créé pour connecter le module NFC à la Raspberry.

Le câble comporte à l'une de ses extrémités un connecteur HE14 femelle de 4 contacts au pas de 2.54 mm.

A l'autre extrémité se trouve un connecteur JWT femelle de 6 contacts au pas de 2.54 mm.

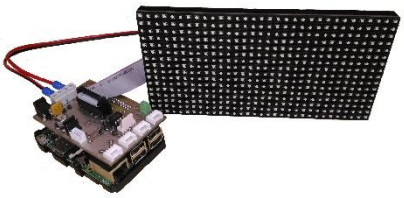
Conducteur noir => GND

Conducteur rouge => 5V

Conducteur jaune => TX

Conducteur vert => RX

24.4 Câblage de la matrice LED RGB 16x32

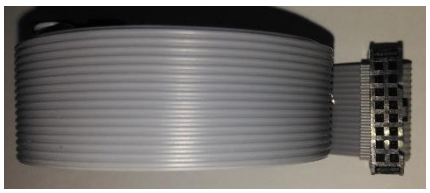


La matrice LED RGB 16x32 est connectée au Shield grâce à une nappe 2x8 contacts et un câble d'alimentation.

La nappe est faite grâce à une nappe de 16 conducteurs au pas de 1.27 mm et à ses extrémités se trouvent deux connecteurs HE10 femelles au pas de 2.54 mm.

Le câble d'alimentation est celui fourni avec la matrice LED RGB 16x32.

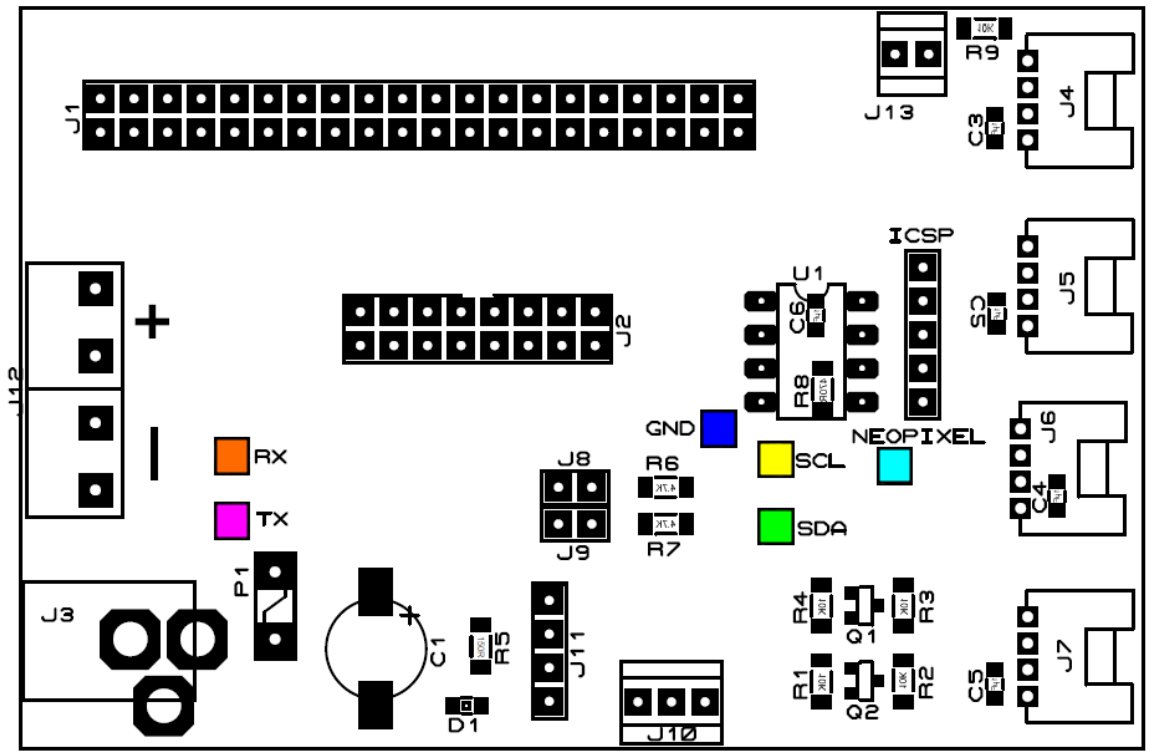
Nappe 2x8 contacts



24.5 Prise de mesures sur le Shield

Sur le Shield est prévu cinq picots pour la prise de mesure.

Ces picots permettent de connecter un oscilloscope ou bien un analyseur logique.



En rouge et rose la liaison UART

RX

TX

En jaune et vert I2C

SCL

SDA

En bleu turquoise DATA NEOPIXEL

NEOPIXEL

En bleu la MASSE (GND)

GND

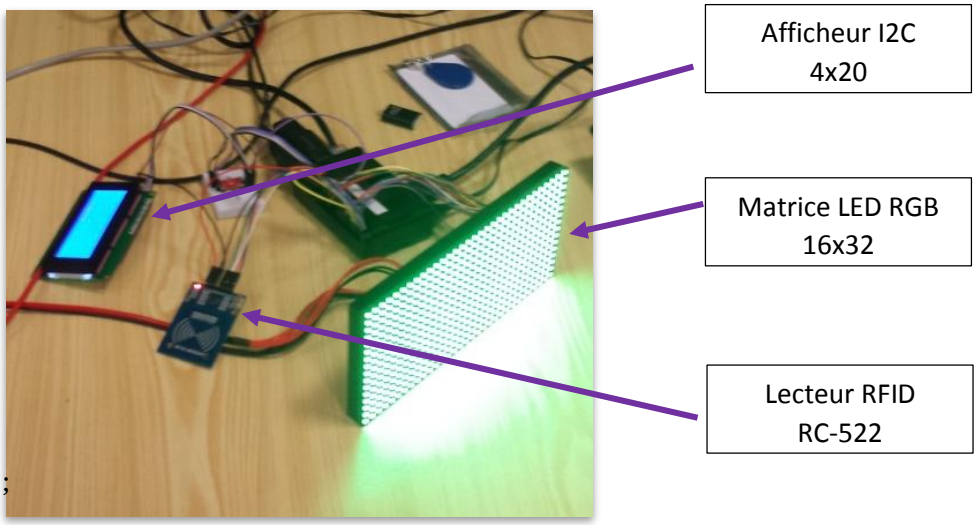
24.6 Travail conjoint au projet URBACO Raspberry PI

24.6.1 Test d'intégration software et hardware

Depuis le but du projet jusqu'à aujourd'hui nous avons fait trois tests d'intégration de la partie hardware et software.

Le premier essai d'intégration s'est fait aux alentours du 27 janvier 2016.

Il y avait la matrice LED RGB 16x32, le lecteur RC-522 RFID, et l'afficheur I2C 4x20



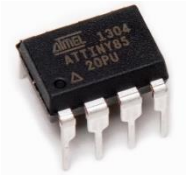
;;

Matéo a lancé l'exécution de son code pour la matrice LED RGB 16x32 ; il a eu le résultat escompté. Ensuite, il a lancé mon programme d'exemple pour l'afficheur I2C 4x20 ; là aussi, nous avons eu le résultat escompté. Il a lancé le programme permettant de lire les badges RFID et là cela n'a pas fonctionné car la matrice 16x32 utilise les broches de la SPI (logiciellement) et le lecteur RFID étant en SPI, il ne peut fonctionner. Matéo c'est donc dirigé vers une nouvelle solution : un lecteur NFC en UART.

24.6.2 Test d'intégration n°2

Le deuxième test d'intégration a été fait cette fois ci avec le lecteur NFC en UART, l'afficheur I2C 4x20, la matrice LED RGB 16x32 et les LED NEOPIXEL.

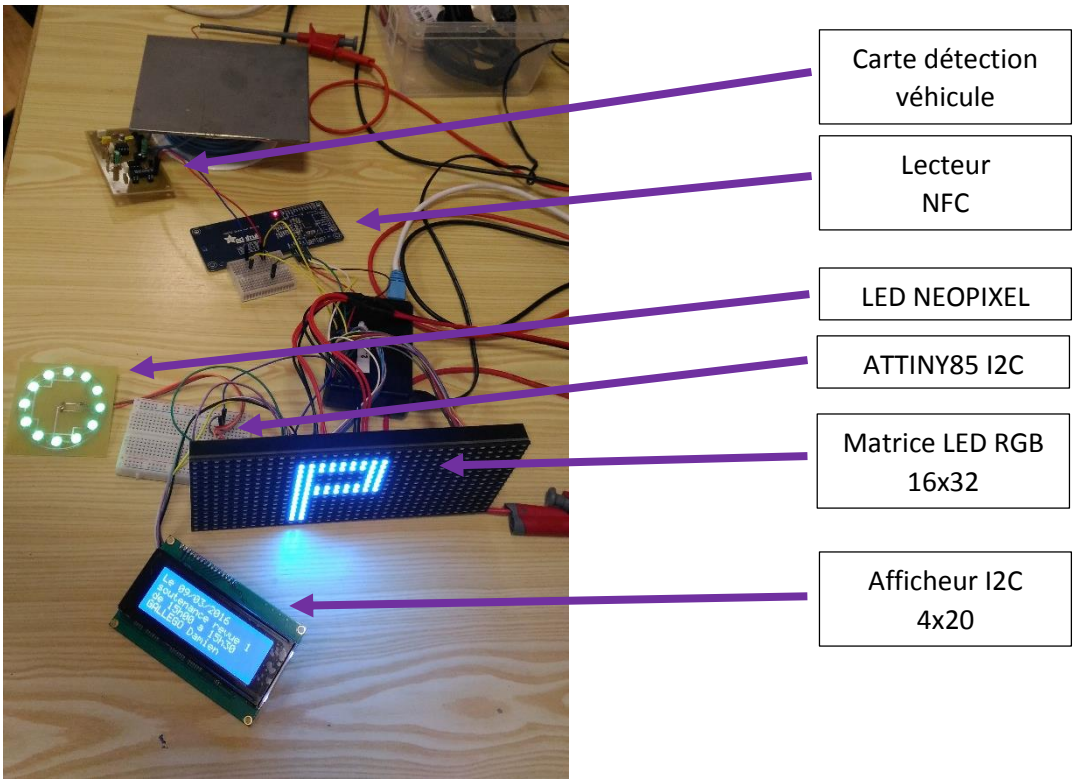
Cette fois-ci tout fonctionne hormis un problème de conflit entre l'exécution du code permettant de faire fonctionner la matrice et le code permettant de faire fonctionner les LED NEOPIXEL.



Par la suite nous avons proposé de faire fonctionner les LED NEOPIXEL grâce à un ATTINY85 en I2C slave car similaire a une Arduino, la librairie NEOPIXEL fonctionne dessus.

24.6.3 Test d'intégration n°3

Le troisième test d'intégration s'est fait avec le lecteur NFC en UART, la matrice LED RGB 16x32, le module permettant la détection de véhicule, les LED NEOPIXEL fonctionnant grâce à l'ATTINY85 en I2C et l'afficheur I2C 4x20.



Cette fois ci tout fonctionne correctement car Matéo a utilisé un autre brochage pour la matrice LED RGB 16x32. De ce fait, le bus UART est disponible et est non utilisé par la matrice.

Les LED NEOPIXEL fonctionnent grâce à l'ATTINY85 (qui est un esclave I2C) car pour le fonctionnement des LED NEOPIXEL, il a besoin de vitesse 800kHz et surtout de temps réel.

25 Fiches recette

FICHE RECETTE, QUALIFICATION DU SYSTEME

Qualification du système :	**Borne Arrêt Minute**	Code de la campagne de test :	xxx
Etudiant :	Damien GALLEGO	Date :	01/04/2016

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Affichage des informations de type « informations municipales »
Objectif du scénario :	Montrer au client le bon fonctionnement de l'affichage des informations municipales et voir avec le client les points qu'il ne comprend pas.

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

Avoir une Raspberry PI fonctionnelle, avoir le programme d'exemple de l'afficheur LCD 4x20, avoir le Shield sur lequel connecteur l'afficheur LCD et le bloc d'alimentation du Shield.

EXECUTION DU TEST

Description courte :	Brancher un cordon Ethernet sur le connecteur de la RPI le connecter à un PC ayant les logiciel adéquate, connecter le Shield au GPIO de la RPI, connecter l'afficheur LCD grâce à son câble au Shield, brancher la RPI au secteur, brancher le Shield au secteur et lancer le programme d'exemple sur la RPI, renseigner les champs vide sur l'interface graphique et cliquer sur Envoyer tous.
Résultats attendus :	Affichage des champs au préalable remplie sur l'afficheur LCD 4x20

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

FICHE RECETTE, QUALIFICATION DU SYSTEME

Qualification du système :	**Borne Arrêt Minute**	Code de la campagne de test :	xxx
Etudiant :	Damien GALLEGO	Date :	01/04/2016

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Création d'un Shield permettant le regroupement des diffèrent modules
Objectif du scénario :	Montrer au client le bon fonctionnement de l'affichage des informations municipales et voir avec le client les points qu'il ne comprend pas.

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

Avoir une Raspberry PI fonctionnelle, avoir les différent modules du projet URBACO RPI, (Afficheur LCD, Matrice RGB LED 16x32, Le lecteur NFC, Les matrice 8x8, La DS3231 et la carte de détection de véhicule) et le bloc d'alimentation secteur de la RPI et du Shield.

EXECUTION DU TEST

Description courte :	Brancher un cordon Ethernet sur le connecteur de la RPI le connecter à un PC ayant les logiciel adéquate, connecter le Shield au GPIO de la RPI, connecter les diffèrent modules citer au-dessus puis lancer les programmes d'exemple des différents modules.
Résultats attendus :	Fonctionnement des différents modules connecté au Shield.

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

26 Visite de la société URBACO

Avec les professeurs Mr HORTOLLAND et Mr DEFRANCE et les deux groupes qui travaillent sur le projet URABCO Borne arrêt minute nous somme aller visiter la société URBACO suite à l'initiative de Alexandre CARTEGNIE à prendre rendez-vous avec la société.

Nous avons rencontré Mr Alain BUSSI et Mr Daniel ROVAI. On a pu leur montraient là où nous en étions et leur expliquer ce que nous allions faire.

Nous avions aussi décidé d'aller visiter cette société car ils possèdent la borne sur laquelle nous fessons notre projet et nous voulions allais la voir pour pouvoir la comparer à celle que nous sommes en train de faire.

27 Conclusion

Le projet que l'on nous a confier cette année, le projet URBACO Borne arrêt minute sur Raspberry PI est un projet assez volumineux pour qu'il y ai trois étudiants en électronique et un étudiant en informatique.

Cela ma permit de travailler en équipe, écouter les autres, d'approfondir mes connaissance dans l'électronique et dans la programmation.

Dans ce projet il y aura eu une grande collaboration entre les différent étudiants même au travers des deux projets URBACO car il a deux projet borne arrêt minute un avec une Raspberry PI et un autre avec une Arduino Yun.

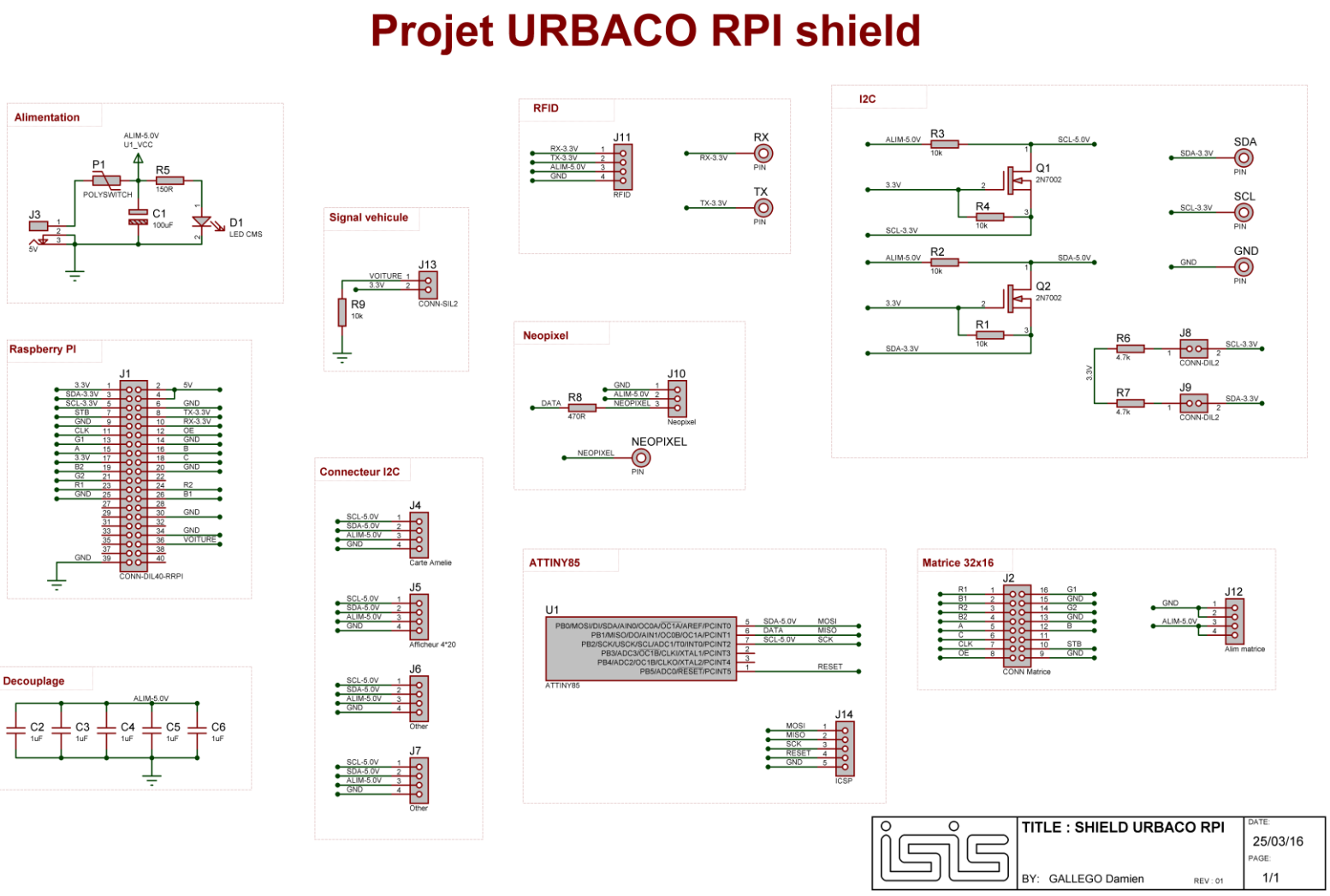
Je tiens à remercier de nouveau les professeurs ayant fait partie de ce projet qui nous on accompagner tout du long, et la société URBACO d'avoir pris du temps pour nous faire visiter l'entreprise et nous avoir donné des conseil pratique sur comment fonctionne réellement une borne.

28 Annexes

28.1 Ci2c

```
////////////////////////////////////
//Ci2c.h
////////////////////////////////////

#include <unistd.h>                     //Needed for I2C port
#include <fcntl.h>                      //Needed for I2C port
#include <sys/ioctl.h>                  //Needed for I2C port
#include <linux/i2c-dev.h>       //Needed for I2C port
#include <iostream>
#include <string.h>


class Ci2c
{
       private :

       int file_i2c;
       int addrSlave;

       public :

       Ci2c(char i2cBus, int addr);
       ~Ci2c();
       int readFromI2C(unsigned char *buffer,int length);
       int writeToI2C(unsigned char *buffer, int length);
};

////////////////////////////////////
//Ci2c.cpp
////////////////////////////////////

#include "Ci2c.h"
#include <string>

using namespace std;

Ci2c::Ci2c(char i2cBus,int addr)
{
       char filename[11] = "/dev/i2c-";
       filename[strlen(filename)] = i2cBus;
       filename[strlen(filename)] = '\0';
       file_i2c = open(filename, O_RDWR);
       addrSlave = addr;
       ioctl(file_i2c, I2C_SLAVE, addrSlave);
}

Ci2c::~Ci2c()
{}

int Ci2c::readFromI2C(unsigned char *buffer, int length)
{
       return read(file_i2c, buffer, length);
}

int Ci2c::writeToI2C(unsigned char *buffer, int length)
{

        return write(file_i2c, buffer, length);

}
```

28.2 Schéma ISIS



28.3 Scanner_I2C

```
#include <iostream>
#include "Ci2c.h"

using namespace std;

// Compilation
// g++ -o scanner_I2C scanner_I2C.cpp Ci2c.cpp

int main(void)
{
	unsigned char address = 0x01, buffer[1] = {0};
	int nDevices = 0, error;

	cout<<"Scan en cours..."<<endl;

	for(int i = 1; i < 127; i++)
	{
		Ci2c test('1', address);
		error = test.writeToI2C(buffer, 0);
		address++;

		if(!error)
		{
			cout<<"Périphérique I2C trouvé a l'adresse
0x"<<hex<<i<<endl;

			nDevices++;
		}
	}
	if(!nDevices)
	{
		cout<<"Aucun périphérique I2C trouvé"<<endl;
	}
	else
	{
		cout<<"Complet"<<endl;
	}
}
```

28.4 Commande GOTRONIC

Référence	Désignation	PU HT	PU TTC	Qté	Total HT	Total TTC
33442	Lot de 5 câbles Grove 30 cm	2,92 €	3,50 €	2	5,83 €	7,00 €
31426	Lot de 5 câbles Grove 20 cm	2,75 €	3,30 €	2	5,50 €	6,60 €
08062	Connecteur HE10F16	0,46 €	0,55 €	2	0,92 €	1,10 €
14387	Adaptateur MV530N	9,92 €	11,90 €	1	9,92 €	11,90 €
48138	Embase alim SA21A	0,50 €	0,60 €	1	0,50 €	0,60 €
31232	Connecteur Grove droit	0,13 €	0,15 €	30	3,75 €	4,50 €
31234	Connecteur Grove coudé	0,13 €	0,15 €	30	3,75 €	4,50 €
09792	Connecteur HE10BC16	0,58 €	0,70 €	2	1,17 €	1,40 €
08027	Connecteur FH220	0,92 €	1,10 €	2	1,83 €	2,20 €
09782	Connecteur HE10BD16	0,58 €	0,70 €	2	1,17 €	1,40 €

Sous-total HT	34,33 €
Sous-total TTC	41,20 €
Frais de port HT	6,67 €
Frais de port TTC	8,00 €
Montant TVA	8,20 €
Total TTC	49,20 €

www.gotronic.fr est la propriété de la société Go Tronic - SARL au capital de 8000 €
CODE APE 4759B - N° Siret 43830668000028 - N° de TVA FR80.438.306.680
Réserve de propriété : nous nous réservons la propriété des marchandises jusqu'au paiement intégral de nos factures (loi N°80335 du 12 mai 1980).
Loi n°92.1992 du 31/12/1992, un intérêt de retard égal à 1,5 fois le taux d'intérêt légal sera appliqué au montant de la facture lors d'un paiement tardif.

28.5 Commande RS-Online

Lycée A. Benoit
B.P. 118
84803 L'Isle sur la Sorgue cedex
Tél. 04 90 20 64 20
Fax 04 90 20 57 20

FOURNISSEUR

RS Composant
Fax : 0 825 345 000

BON DE COMMANDE

Référence à rappeler sur la facture et le bon de livraison

BON N° T

Cpte N°

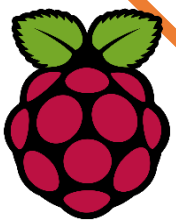
Désignation	réf. fournisseur	Qté	PUHT	PTHT
Les CMS verte Udirect 3,2V 0805	692-0922	10	0,334	3,34
Polyswitch Im 1,6A Id 3,2A	647-8550	10	0,448	4,48
Resistance CMS 150 ohm 1 % 1206	812-1792	25	0,142	3,55
Resistance CMS 10 Kohm 1 % 1206	812-1789	25	0,142	3,55
Connecteur 2*8 HE10 nape	454-2508	2	1,370	2,74
Connecteur 3 cts 2,54 mm	790-1092	5	0,780	3,9
Convertisseur TLC271BIP	661-1662	10	0,905	9,05
Mosfet VN2222LL	879-3274	25	0,292	7,3
Polyswitch Im 0,25A Id 0,5A	517-6607	10	0,401	4,01
Resistance CMS 470 ohm 5 % 1206	740-9120	50	0,017	0,85
Resistance CMS 4,7 Kohm 1 % 1206	223-2350	50	0,019	0,95
Condensateur 1µF 16V 0805	264-4450	25	0,034	0,85
Bornier 2cts 2,54mm	790-1098	5	0,552	2,76
TOTAL HT				**47,33**
			TOTAL TTC	**56,8**

IMPORTANT

Adresser vos expéditions au service de l'intendance

BTS SN
Le / /2015

	**Epreuve E6.2 Projet URBACO Raspberry PI 2016**	



Projet URBACO RPI

BTS Systèmes Numériques

Thomas VINCENT

Candidat : Thomas VINCENT

29 Introduction

Ce projet a été rendu possible grâce au partenariat avec l'entreprise Urbaco basé à Entraigues, il a pour but de créer une borne de stationnement intelligente qui à l'aide d'une Raspberry pi 2 gérera le temps de stationnement des utilisateurs.

L'entreprise URBACO dispose déjà à son catalogue d'une solution d'arrêt minute. Cependant celle-ci est à base d'automate programmable type Siemens Logo et ce type de matériel est coûteux. De plus les temps de câblage sont importants.

Afin de réduire les coûts et faciliter la mise en œuvre. URBACO a décidé d'opter pour une carte électronique dédiée en lieu et place de l'automate programmable.

Pour ce faire nous avons un budget alloué de 300 à 400€ maximum, c'est pour cette raison que l'on a pris une carte électronique Raspberry pi 2 qui sont puissante et low-cost (40€).

30 Raspberry PI 2

Sorti officiellement le 29 février 2012, le Raspberry Pi est développé par la Raspberry Pi Foundation. Il s'agit d'un ordinateur dont la taille est comparable à celle d'une carte de crédit. Il est actuellement décliné en trois modèles se différenciant par leurs composants et leurs prix. Ainsi nous avons choisi le modèle raspberry pi 2 B+ qui à la puissance nécessaire pour faire fonctionner le serveur web, le hotspot wifi et le programmes de fonctionnement de la borne.

Raspberry Pi 2:

- 900MHz quad-core ARM cortex-A7

- 1 GB RAM
- 4 USB ports
- 1 Port HDMI
- 1 port ethernet
- 1 Interface caméra
- 1 Interface d'affichage
- 1 Slot carte Micro SD
- 1 Port jack 3,5mm et composite video combiné

30.1 Serveur web embarqué (Lighttpd)

Le serveur web lighttpd, est un serveur web light qui nous permet d'économiser de la mémoire sur la raspberry. Pour installer ce serveur il a suffit de télécharger son paquet grâce à la commande « apt-get install lighttpd », une fois cela fait je lui ai ajouté les modules suivant : php5 php5-cgi php5-common php-pear php5-sqlite php5-dev.

Une fois cela fait on peut commencer à l'utiliser comme serveur web sur la raspberry. Il permet aussi de définir des serveur virtuels (cf.chapitre 3.2)

30.2 Base de données embarqué (Sqlite3)

Sqlite est est une base de données relationnelles accessible par le langage SQL et qui implémente en grande partie le standard SQL-92.

Sa particularité réside dans le fait que cette base ne repose pas sur un modèle client-serveur (comme Mysql ou PostgreSQL) mais sur un modèle local dont l'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier unique, indépendant de la plateforme.

De plus comparé aux autres moteurs de base de donées, Sqlite est très léger moins de 300 Ko ce qui en fait le moteur de base de donné le plus adapté pour sont utilisation dans un système embarqué.

30.3 Hotspot Wifi

30.3.1 Clé wifi Edimax

Conforme aux normes sans fil 802.11b/g/n jusqu'à 150Mbps

- Green Power Saving : Prise en charge du control d'energie et mise en veille automatique
- Augmente la zone de couverture par 3◎
- Assistant d'installation multilingue.
- Prise en charge 64/128-bit WEP, WPA , WPA2 et WPS-compatible.
- Prise en charge QoS-WMM, WMM-Power

L'EW-7811Un est un adaptateur nano USB, sans fil, qui offre une portée et une vitesse maximum. Cet adaptateur USB peut atteindre des taux de transfert allant jusqu'à 150Mbps lorsqu'il est connecté à un équipement WiFi 802.11n, et ce malgré son format ultra réduit. Ce composant à été choisi pour sa taille et sa compatibilité avec toutes les normes wifi mais aussi pour sa compatibilité avec le mode hotspot sur raspberry.

30.3.2 Wifi

Les caractéristiques du wifi de la clé Edimax sont les suivantes :

- Fréquence de travail : entre 2.4GHz et 2.5GHz

- Portée en intèrieur : 50m

- Portée en extèrieur = 100m

- Modulation du siganl : OFDM (Orthogonal frequency-division multiplexing)

- Longueur d'onde : 0,125m ($\lambda = c/f$)

- Taille l'antenne : 3,1 cm (taille de l'antenne = 0,25*12,5cm)

Ces calculs sont basé sur la documentation sur les antennes wifi trouvé sur le net, n'ayant rien trouvé de précis à ce sujet ces données sont possiblement erroné, mais le constructeur ne détaillant pas le type d'antenne dans sa documentation je ne peux savoir exactement le type d'antenne utilisé.

30.3.3 Mise en route du point d'accès

Pour créer un point d'accés wifi avec la clé edimax j'ai utiliser le logiciel hostapd qui permet de créer un point d'accés à partir d'une clé wifi supportant le mode AP (Access Point).

En premier j'ai configuré le fichier de configuration de hostapd il se trouve dans le dossier /etc/hostapd/hostapd.conf

Cela permet de configurer les paramètres du point d'accès tels que le mot de passe, le cryptage (wpa-psk,wpa-aes, wep) ou le ssid.

Ensuite j'ai modifié le fichier /etc/default/hostapd pour lancer le point d'accès au démarrage de la Raspberry.

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

30.4 Modules externes

30.4.1 NEOPIXEL

Le néopixel est un cercle de 61 eds fonctionnant en 5v, il est contrôlé par un Atiny qui uitilise un bus I2C. Au niveau de la borne elles permettent de signaler si la place est libre ou non, ou encore de signaler le dépassement de temps de l'usager.

Illustration 1: Néopixel 61 leds

30.4.2 Afficheur led 16x32

L'afficheur led 16x32 est une planche de 16 leds de large et de 32 leds de long, l'afficheur fonctionne en parallèle et est alimenté en 5v. Cette afficheur sert à afficher le type d'usager qui utilise la place en affichant différent logo selon l'utilisateur (livreur, handicapé, usager gratuit).

Illustration 2: Matrice16x32

30.4.3 Afficheur LCD 4x20

L'afficheur LCD 4x20 a retroeclairage led, est composer de d'un afficheur HD44780 et d'un driver I2C PCA8574, Il est alimenté en 5V. Il permet de d'afficher des messages d'information destiné aux usagers.

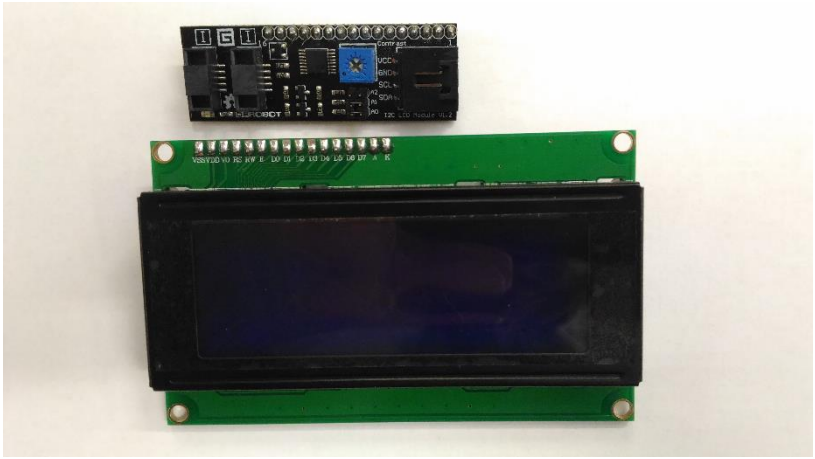


Illustration 3: Afficheur 4x20

30.4.4 Afficheur led 8x8

L'afficheur led 8x8 est composé de 5 carrés de 8 leds de long par 8 leds de large, il fonctionne lui aussi en I2C mais à été adapter pour la raspberry puisque son développement à été fait par le groupe de projet URBACO Arduino donc pour l'arduino Yun. Les afficheurs sont alimenté en 5v. Ils serviront à afficher l'heure en premier temps puis afficher le décompte du temps de stationnement de l'usager.

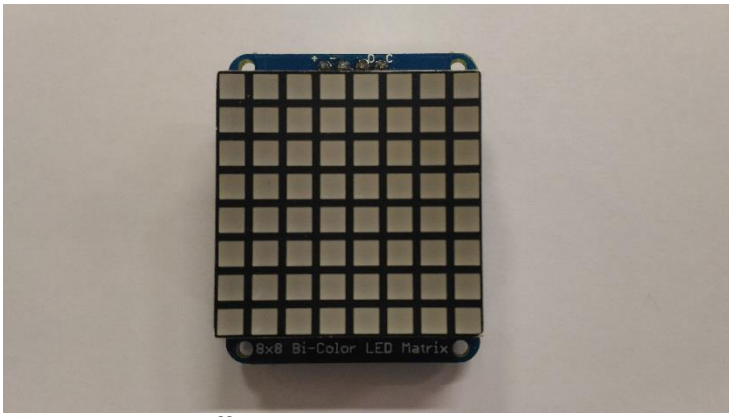


Illustration 4: Afficheur 8x8

30.4.5 Lecteur NFC

Le lecteur NFC permet de lire les badges des utilisateurs spéciaux (livreur, handicapé), le lecteur fonctionne en 5V et fonctionne en UART

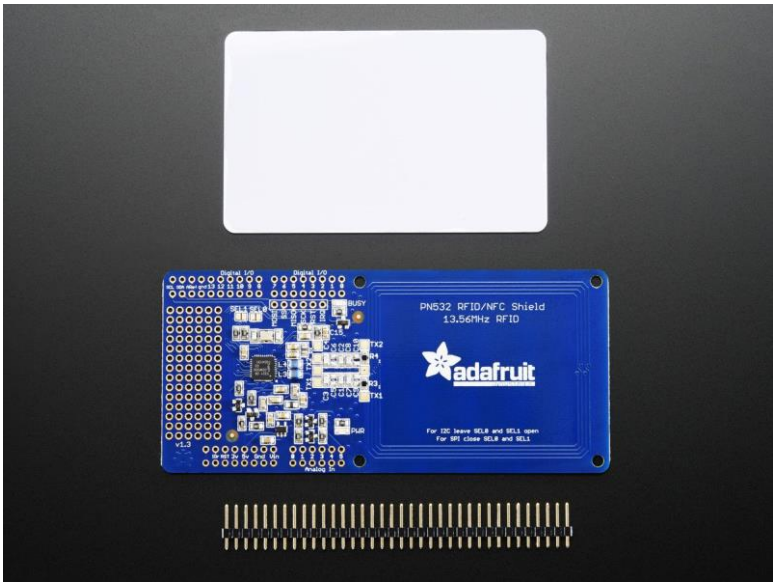


Illustration 5: Lecteur NFC Adafruit

30.4.6 Carte de détection de véhicule

La carte de détection de véhicules est une boucle à induction qui permet de détecter la présence de véhicule. Elle est reliée à la carte Raspberry, c'est un module autonome entièrement analogique.

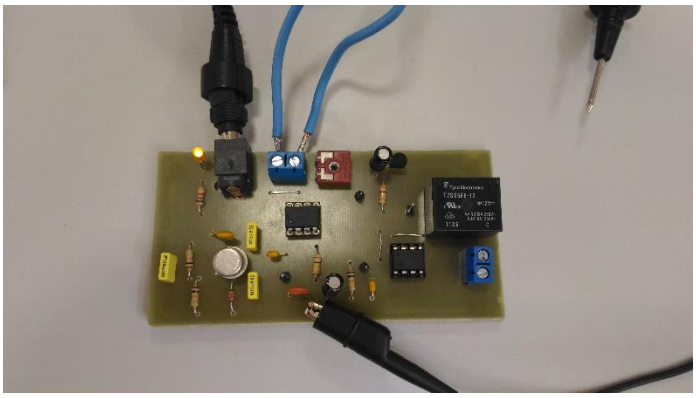


Illustration 6: Carte de détection de véhicule

31 Programme de gestion de la borne

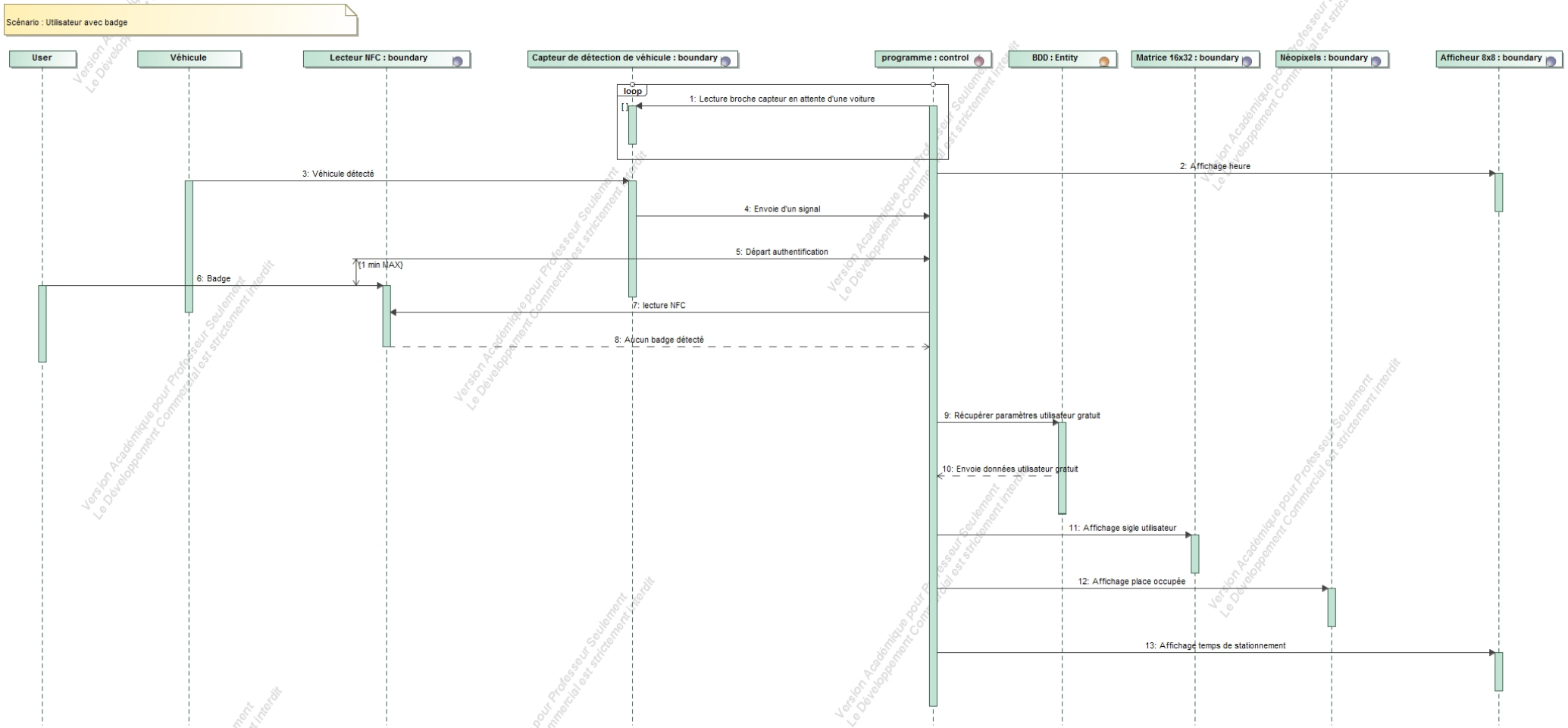
31.1 Fonctionnement interne (Diagramme de séquence)

Ces diagrammes de séquences permettent de décrire les scenarii de la borne de façon simple et compréhensible et aussi celui du programme de la borne.

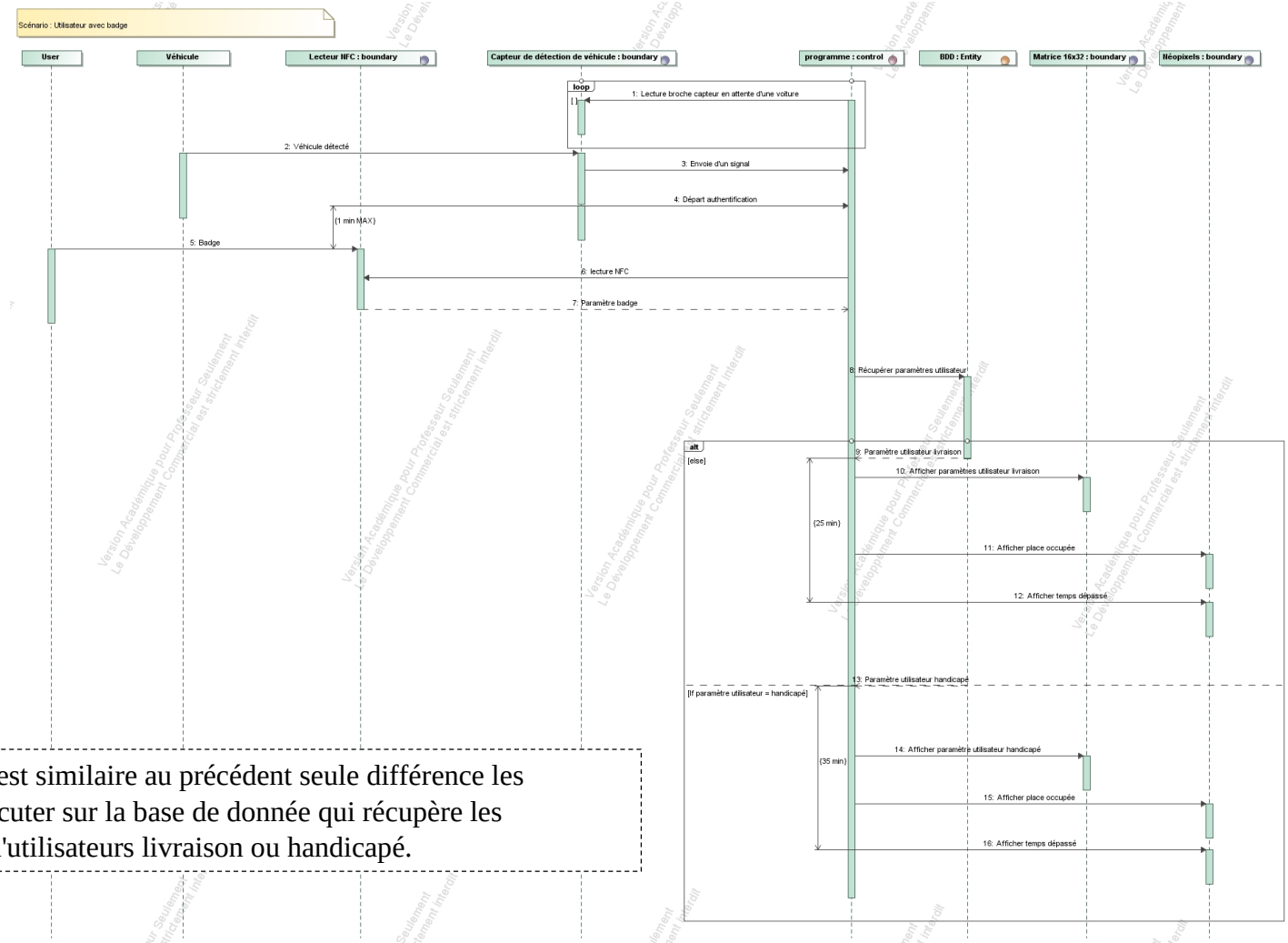
Scénario 1 utilisateur sans badge :

Dans ce scénario l'utilisateur arrive avec son véhicule qui est détecté par la carte de détection de véhicule, celle la même envoie un signal qui est reçu par le programmes, qui ensuite déclenche le lecteur de badge, celui-ci n'ayant détecté aucun badges, le programme exécute une requête dans la base de donnée pour récupérer les données de l'utilisateur sans badges (utilisateur gratuit), puis ils affichent que la place est occupée, le sigle de l'utilisateur et son temps de stationnement.

Voir page suivante.


Scénario : Utilisateur avec badge
User
Véhicule
Lecteur NFC : boundary
Capteur de détection de véhicule : boundary
programme : control
BDD : Entity
Matrice 16x32 : boundary
Néopixels : boundary
Afficheur 8x8 : boundary
loop
[ ]
1: Lecture broche capteur en attente d'une voiture
2: Affichage heure
3: Véhicule détecté
4: Envoie d'un signal
5: Départ authentification
{1 min MAX}
6: Badge
7: lecture NFC
8: Aucun badge détecté
9: Récupérer paramètres utilisateur gratuit
10: Envoie données utilisateur gratuit
11: Affichage sigle utilisateur
12: Affichage place occupée
13: Affichage temps de stationnement

Scénario 2 utilisateur avec badge :



Ce scénario est similaire au précédent seule différence les requêtes exécuter sur la base de donnée qui récupère les paramètres d'utilisateurs livraison ou handicapé.

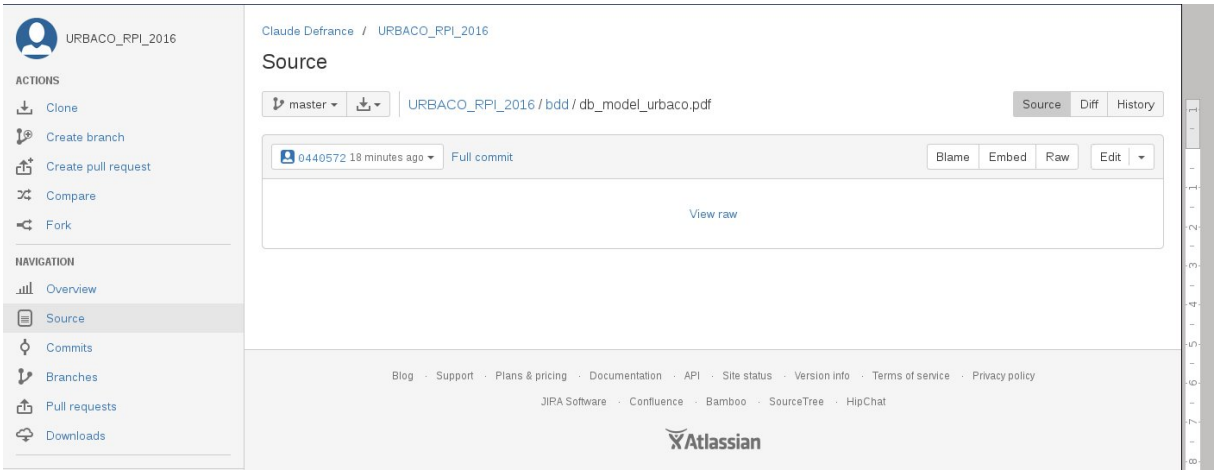
31.1.1 FSM (Finite State Machine)

Une « finite state machine » ou machine à état fini en français est un modèle mathématique de calcul, utilisé dans de nombreuses circonstances, allant de la conception de programmes informatiques et de circuits en logique séquentielle aux applications dans des protocoles de communication. Un automate fini est une construction abstraite, susceptible d'être dans un nombre fini d'*états*, un seul état à la fois, l'état où il se trouve est appelé l'« état courant ». Le passage d'un état à un autre est dirigé par un événement ou une condition ; ce passage est appelé une « transition ». Un automate particulier est défini par la liste de ses états et par les conditions des transitions.

Le programme de la borne est donc basé sur une FSM car son fonctionnement se prêtais bien à l'implémentation de la machine à état fini.

Vous pouvez accéder à mon code en suivant le lien suivant :
https://bitbucket.org/cdefrance/urbaco_rpi_2016.git

Pour télécharger le code aller dans l'onglet source puis cliquez sur le dossier que vous voulez voir et pour télécharger les fichiers cliquez sur l'un deux puis sur « view raw » pour lancer le téléchargement.



31.2 Site web embarqué

Le site web embarqué sur la Raspberry est hébergé sur le serveur lighttpd , une fois installé avec tous ses modules pour le php et sqlite, j'ai édité le fichier /etc/lighttpd/lighttpd.conf comme ci-dessous :

```
server.modules = (

#...

"mod_fastcgi",

#...

)

# At the end of the file

fastcgi.server = (

".php" => ((

"bin-path" => "/usr/bin/php5-cgi",

"socket" => "/tmp/php.socket"

))

)
```

ensuite éditer le fichier /etc/php5/cgi/php.ini comme ci-dessous :

```
cgi.fix_pathinfo = 1
date.timezone = Europe/Berlin
# http://de2.php.net/manual/de/timezones.php
```

une fois cela fait le mode cgi du serveur est activé et le serveur prêt à l'emploi.

31.2.1 Le thème Bootstrap

Bootstrap est une collection d'outils utile à la création de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. Ce thème m'a permis de développer toutes les pages du sites exception faite de la page de login que j'ai faite directement en javascript, html et css.

Pour intégrer bootstrap sur une page web il y deux méthode la première télécharger les fichier bootstrap et Jquery.js puis les mettre dans le dossier du site et les inclures directement dans vos fichier html ou php. J'ai utilisé cette méthode pour le site car il est embarqué sur la Raspberry qui ne sera pas connecté au réseaux ce qui ne permet pas la deuxième méthode ci-dessous.

Deuxième méthode : Intégrer bootstrap et Jquery à partir d'une url avec les balises suivantes :

```
<link rel="stylesheet"
href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/boots
trap.min.css">
```

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.
js"></script>
 <script
src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min
.js"></script>
```

31.2.2 Les graphique avec highcharts

Highcharts est une bibliothèque graphique écrite en HTML5 et JavaScript, offrant des graphiques complets et complexes, mais intuitifs et interactifs pour votre site ou votre application Web. Elle implémente différents types de graphiques (colonnes, lignes, aires, camemberts, nuages de points, etc.). Développée pour les mobiles, Highcharts supporte les tableaux de bord interactifs ou autonomes dans tout projet Web.

Highcharts est très simple d'utilisation, il suffit d'ajouter le script javascript et la feuille de styles css pour pouvoir l'utiliser.

Le paramétrage des graphiques se fait par l'intermédiaire d'un objet JSON , l'éventail de paramétrage est très larges (type de graphique, légende, les échelles, les labels, les axes, des évènement tels que onClick ou onMouseOver).

Cette librairie comprend aussi un module d'export qui permet d'exporter le graphqiue en image JPEG,SVG,PNG.

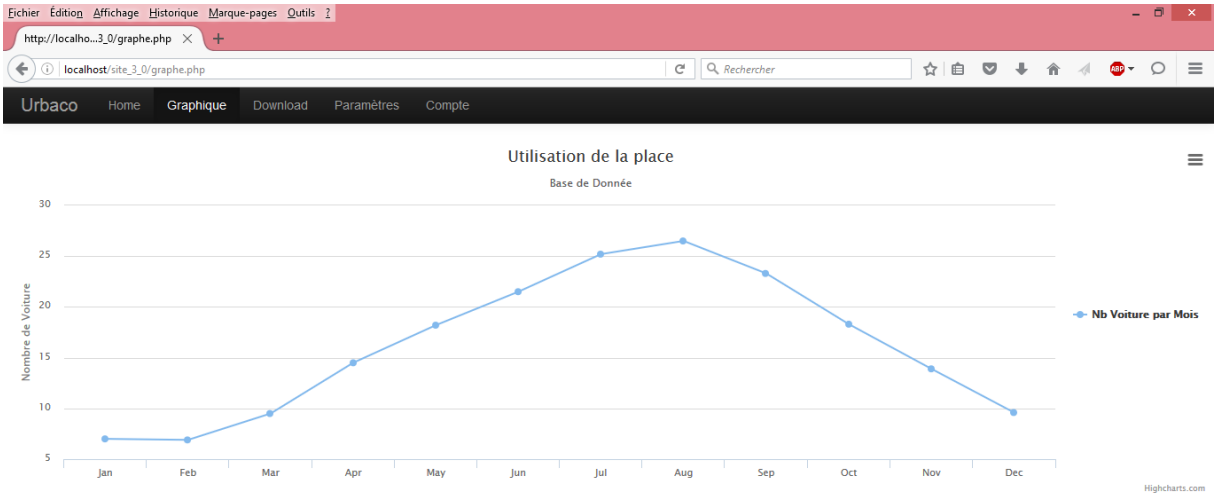


Illustration 7: Exemple de graphique highcharts

31.2.3 Le paramétrage de la borne

Cette pages de paramétrage de la borne permettra au technicien, de modifier les horaires de fonctionnement de la borne mais aussi les horaires de stationnement qui seront payant ou gratuit, ajouter des badges d'utilisateurs et modifier le temps de stationnement accordé aux différents types d'utilisateurs.

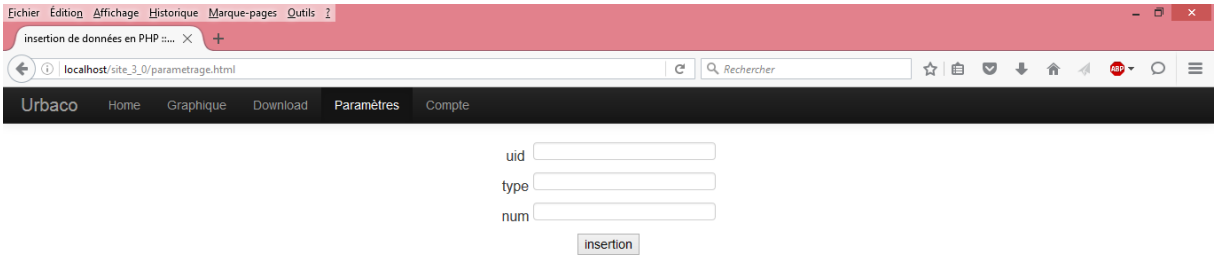


Illustration 8: Page paramétrages du site en version alpha

32 Axe d'amélioration du projet

Durant le projet nous somme allé voir l'entreprise URBACO qui nous a donné sont avis sur le projet et de ce fait nous à permis de penser à des améliorations possible du projet ci-dessous la liste des amélioration possible :

- Créer une base de données sur les badges nfc

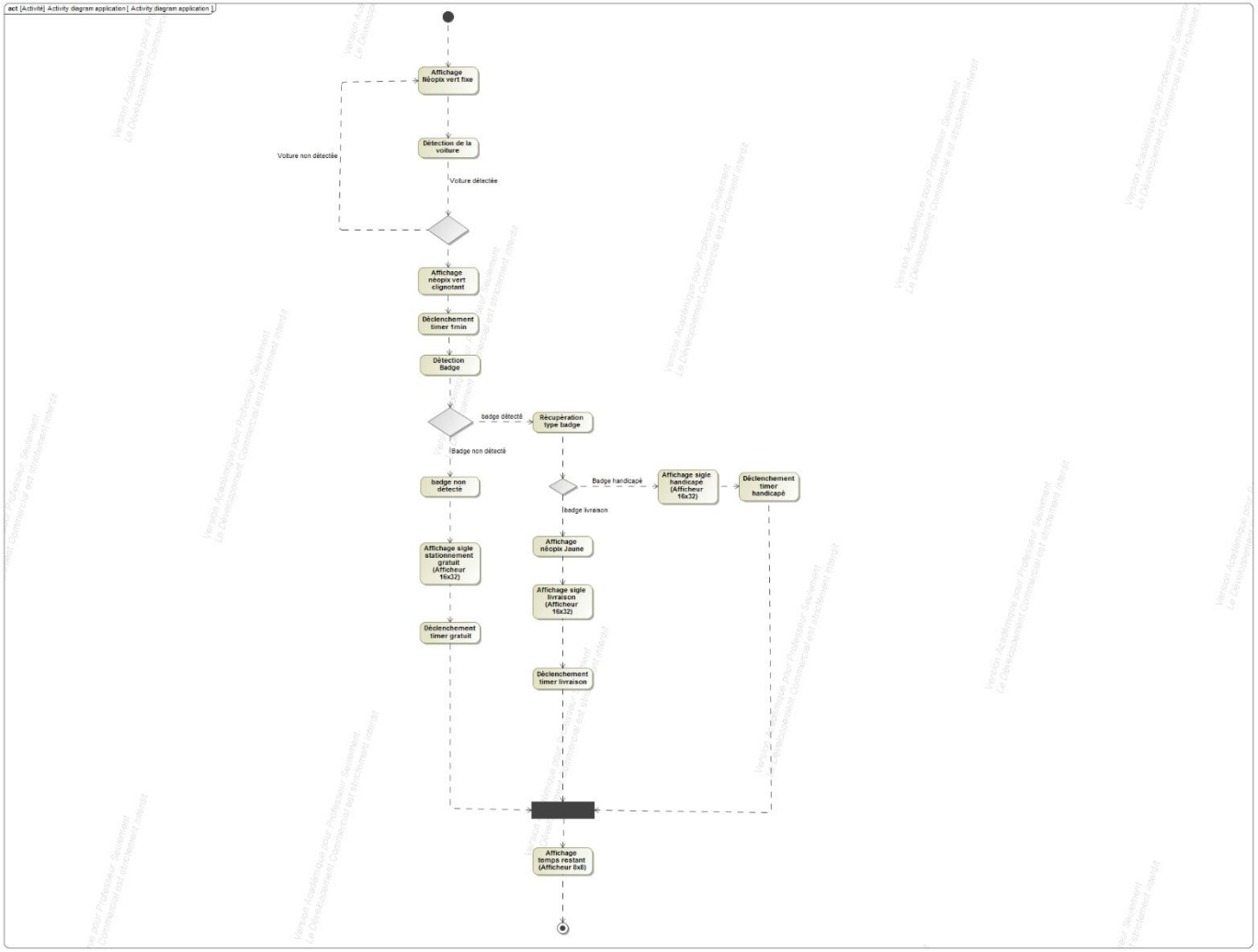
- Ajout d'une marge d'erreur pour le temps de stationnement

- connecter la borne avec des cartes sim et un abonnement MtoM (Machine to machine) et ajout du shield gsm pour raspberry.

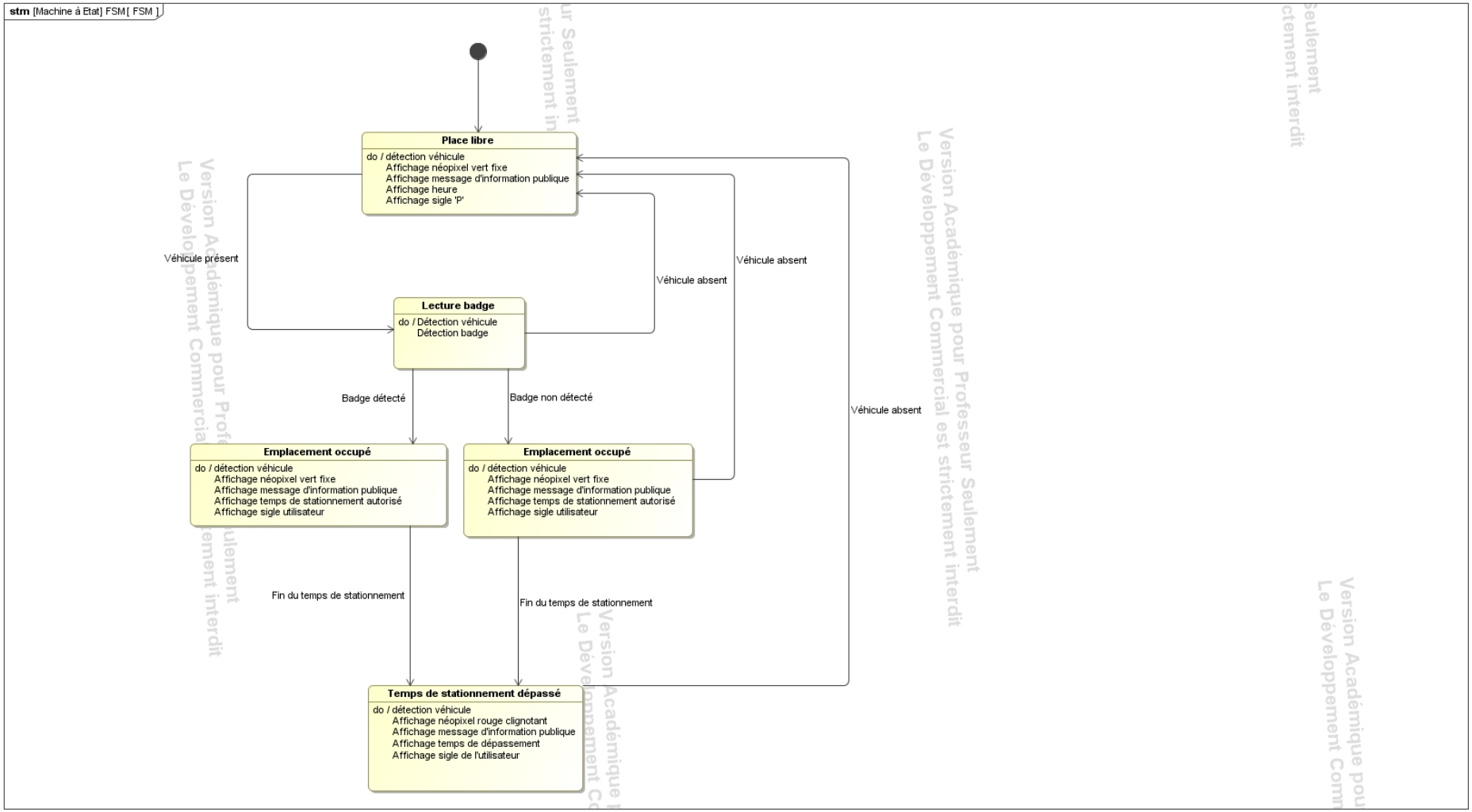
- Eteindre la borne à certain moment ou juste l'affichage pour faire des économies d'énergies.

33 Diagramme SysML

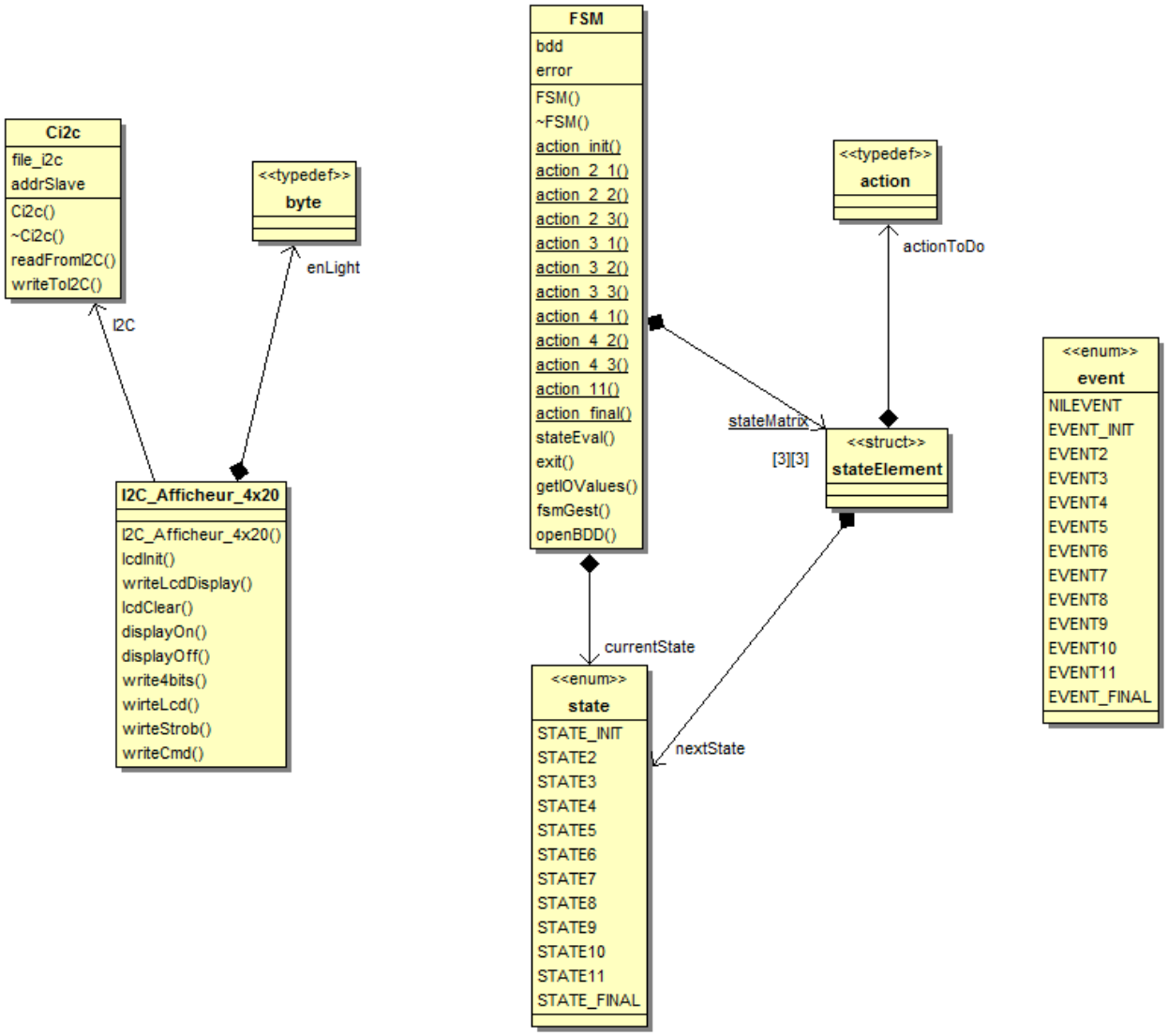
33.1 Diagrammes d'activités



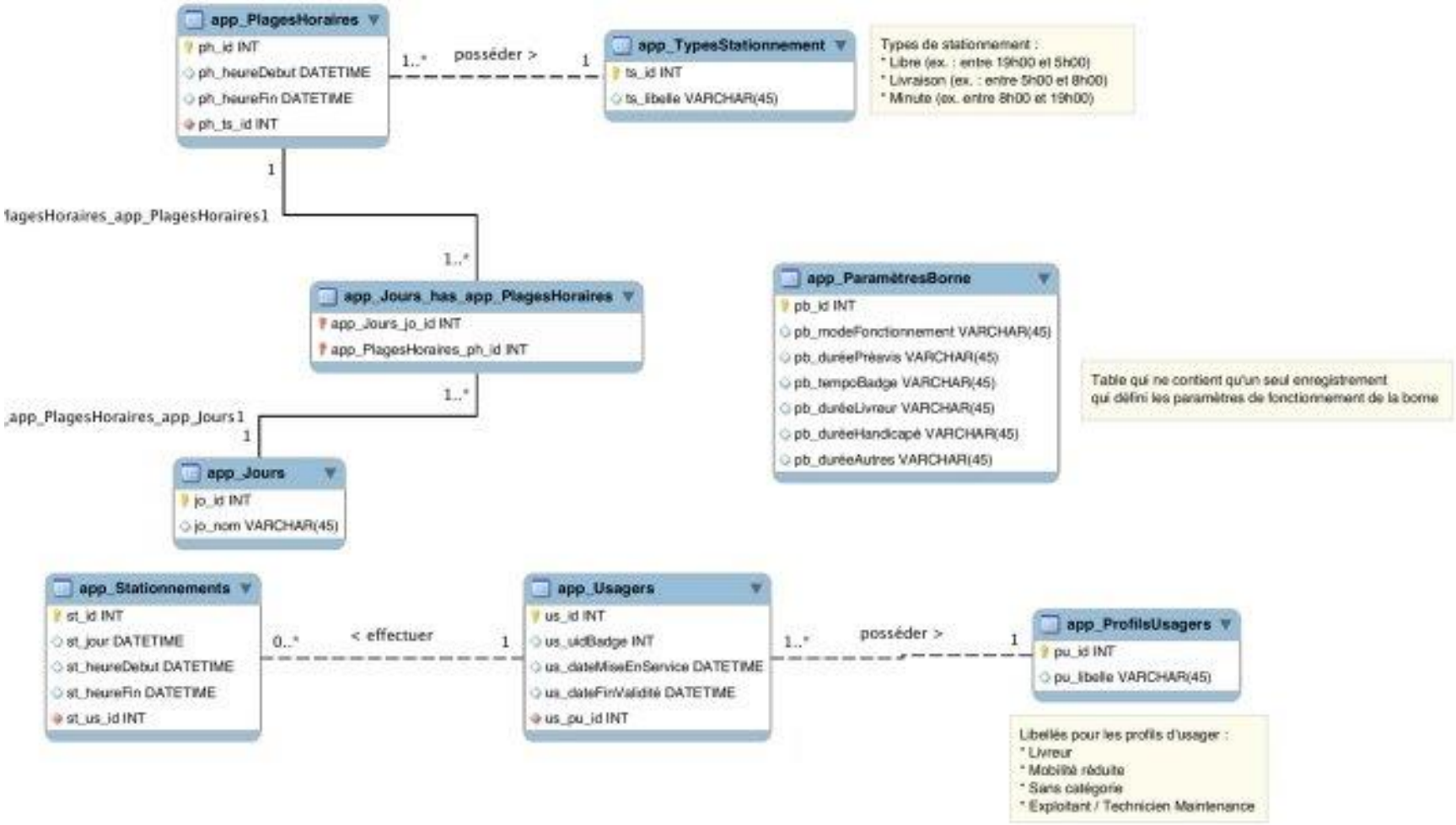
33.2 Diagrammes d'états



33.3 Diagramme de classe



34 Schéma structurel de la base de donnée



35 Annexe

35.1 Fiche recette

FICHE RECETTE, QUALIFICATION DU SYSTEME

Qualification du système :	**Borne Arrêt Minute**	Code de la campagne de test :	xxx
Etudiant :	Thomas VINCENT	Date :	26/04/2016

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Borne arrêt-minute
Objectif du scénario :	Tester le bon fonctionnement de la borne

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

- Alimenter la borne
- Avoir un badge NFC
- Avoir un véhicule ou une plaque métallique pour le simuler
- Avoir une tablette

EXECUTION DU TEST

Description courte :	Une fois la borne alimenter, mettre un véhicule sur le détecteur de véhicule puis reproduire les scénarii d'un utilisateur sans badges et d'un utilisateur avec badge.
Résultats attendus :	La borne doit avertir que la place est occupé, afficher le temps de stationnement restant, afficher le sigle de l'utilisateur et afficher un message d'information aux publiques.

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)