

Académie Aix-Marseille

B.T.S SN 2017

Projet :

MARITON

Lycée Alphonse Benoit
L'Isle sur Sorgue

Etudiants :

Doux Christopher

Benkheddouma Jordan

Galatioto Enzo

Contassot Thomas



Sommaire

Partie Commune	2
I/ L'entreprise	2
II/ Le projet.....	2
III/ Cahier des charges.....	2
IV /Diagramme d'Activité et Carte Mentale.....	3
V/ Partie Personnelle Doux Christopher	4
VI/ Partie personnelle de Benkheddouma Jordan	29
VII/ Partie personnelle de Galatioto Enzo	1
VIII/ Partie personnelle de Contassot Thomas.....	13

Partie Commune



I/ L'entreprise

Mariton est une entreprise spécialisée dans la fabrication de stores et de moustiquaires sur mesure. Elle est localisée à Saint Chamas dans les Bouches du Rhône (13).

Depuis plus de 60 ans, Mariton habille aussi bien les fenêtres des maisons individuelles que celles des entreprises, des administrations et des collectivités.

Innovante, performante et forte d'un savoir-faire maîtrisé, Mariton propose des produits de qualité, bien pensés et adaptés à chaque configuration : stores vénitiens, stores à bandes verticales, stores enrouleurs intérieurs ou extérieurs, stores plissés, panneaux japonais, vélums de véranda, moustiquaires pour fenêtre, moustiquaires pour porte, Sécurit'Air...

II/ Le projet

Le but du projet est de réaliser un banc de test, aussi appelé "cycleur", afin de tester sur plusieurs milliers de cycles des stores ayant 2 types de motorisations : soit 24V continu, soit 230V alternatif.

Un cycle correspond à une descente du store suivie d'une temporisation de 2 minutes (par défaut) puis d'une remontée suivie d'une temporisation de 3 minutes (par défaut). Les temporisations permettent au moteur de refroidir.

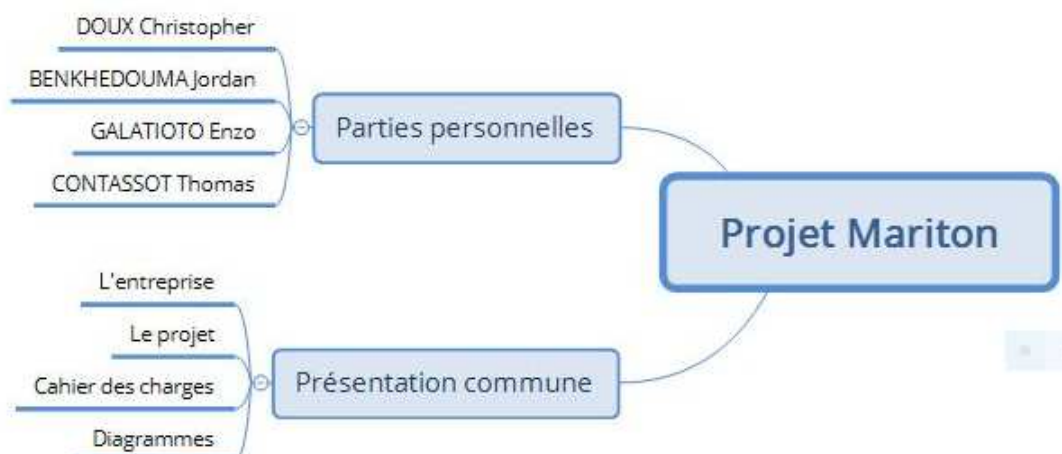
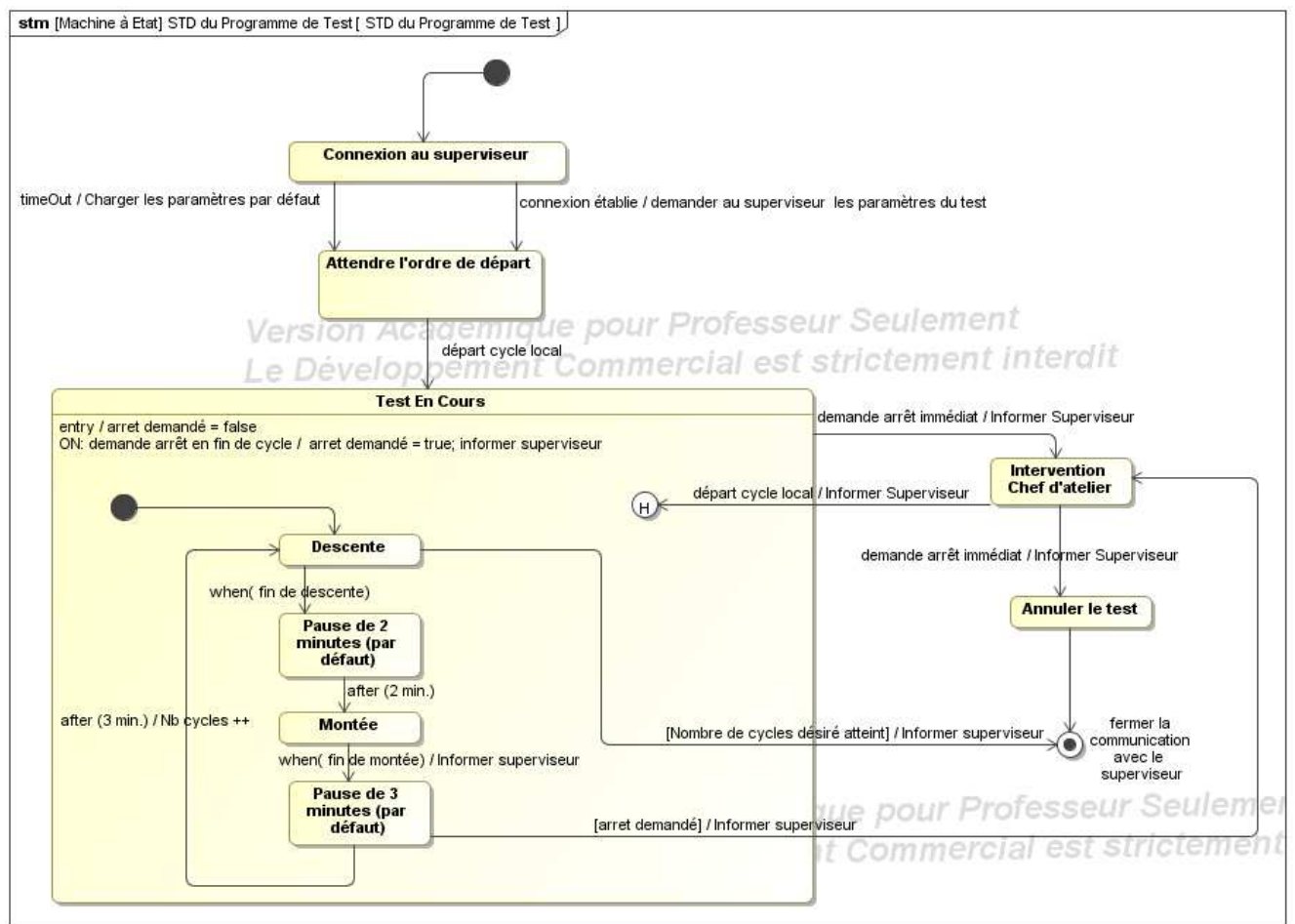
III/ Cahier des charges

Il s'agit de prolonger les activités de la société dans le domaine de la domotique et d'acquérir une autonomie dans la fourniture de matériels et logiciels nécessaires pour l'intégration aux produits existants. Dans ce projet il s'agit plus précisément de créer un banc de test des stores.

Pour cela nous avons à notre disposition le matériel suivant :

- *. Une carte Raspberry Pi,
- *. Une carte Arduino,
- *. Un store,
- *. Et une imprimante.

IV /Diagramme d'Activité et Carte Mentale



V/ Partie Personnelle Doux Christopher

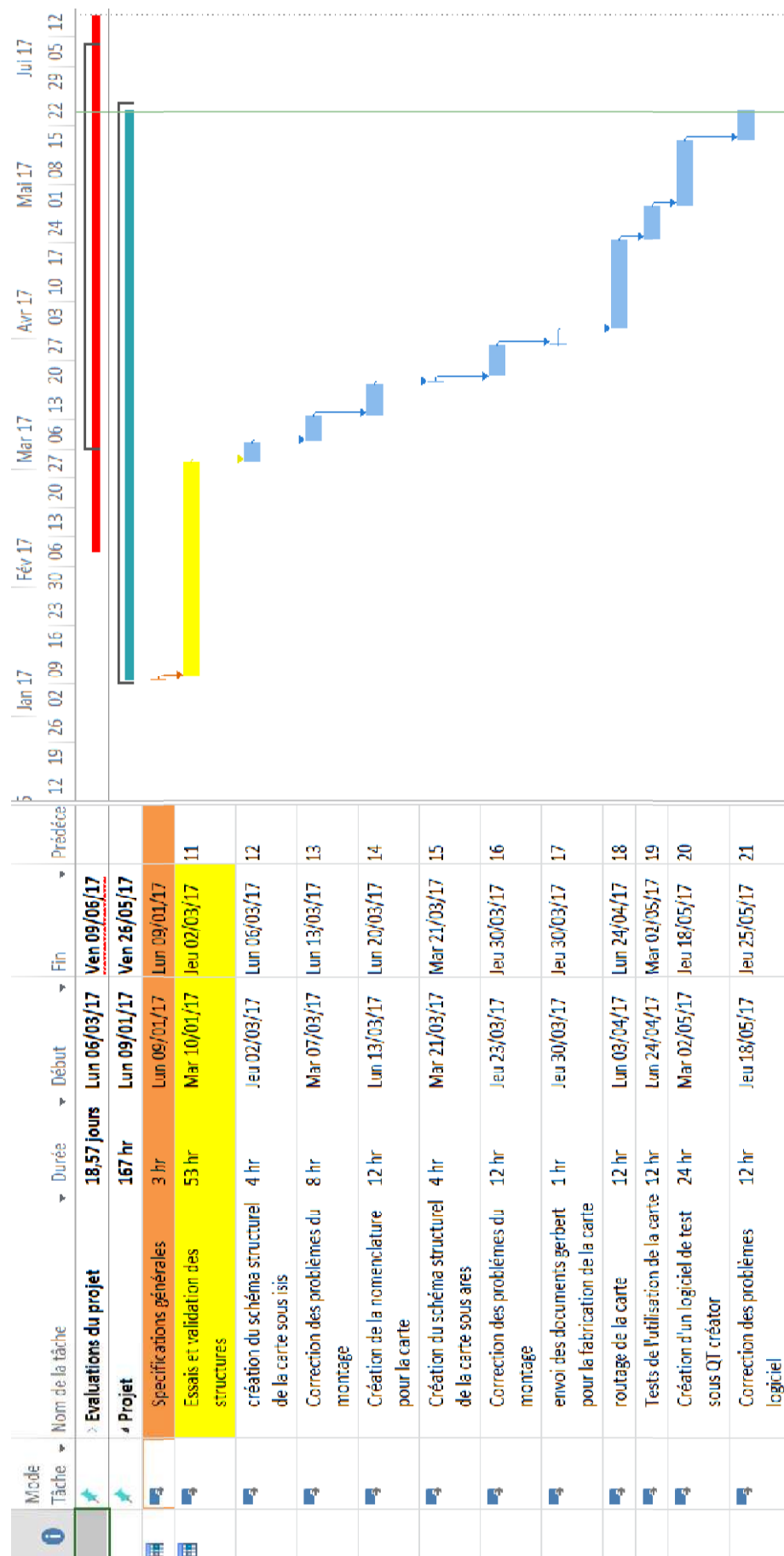
1/ INTRODUCTION

Le travail en tant que ec2 consistera principalement à la création d'une carte sur la Raspberry pi permettant la gestion d'un store et l'information du personnel entourant le matériel (informations visuelles).

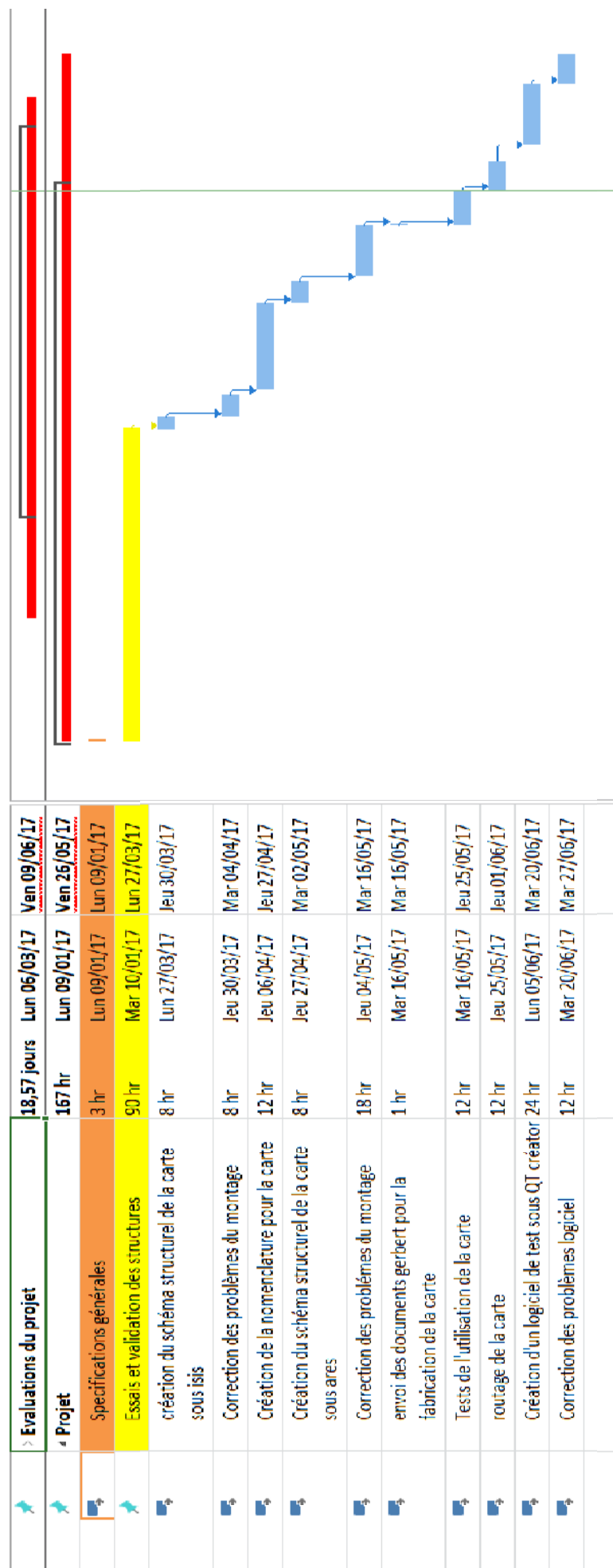
La carte sera utilisé pour :

1. connecter en liaison TOR:
 - les relais pour l'alimentation du store
 - des boutons poussoirs pour l'arrêt d'urgence
 - des deepSwitchs
2. connecter en liaison i2c:
 - un afficheur lcd
 - une horloge en temps réelle
 - un ATtiny85 pour la gestion des LEDs d'information

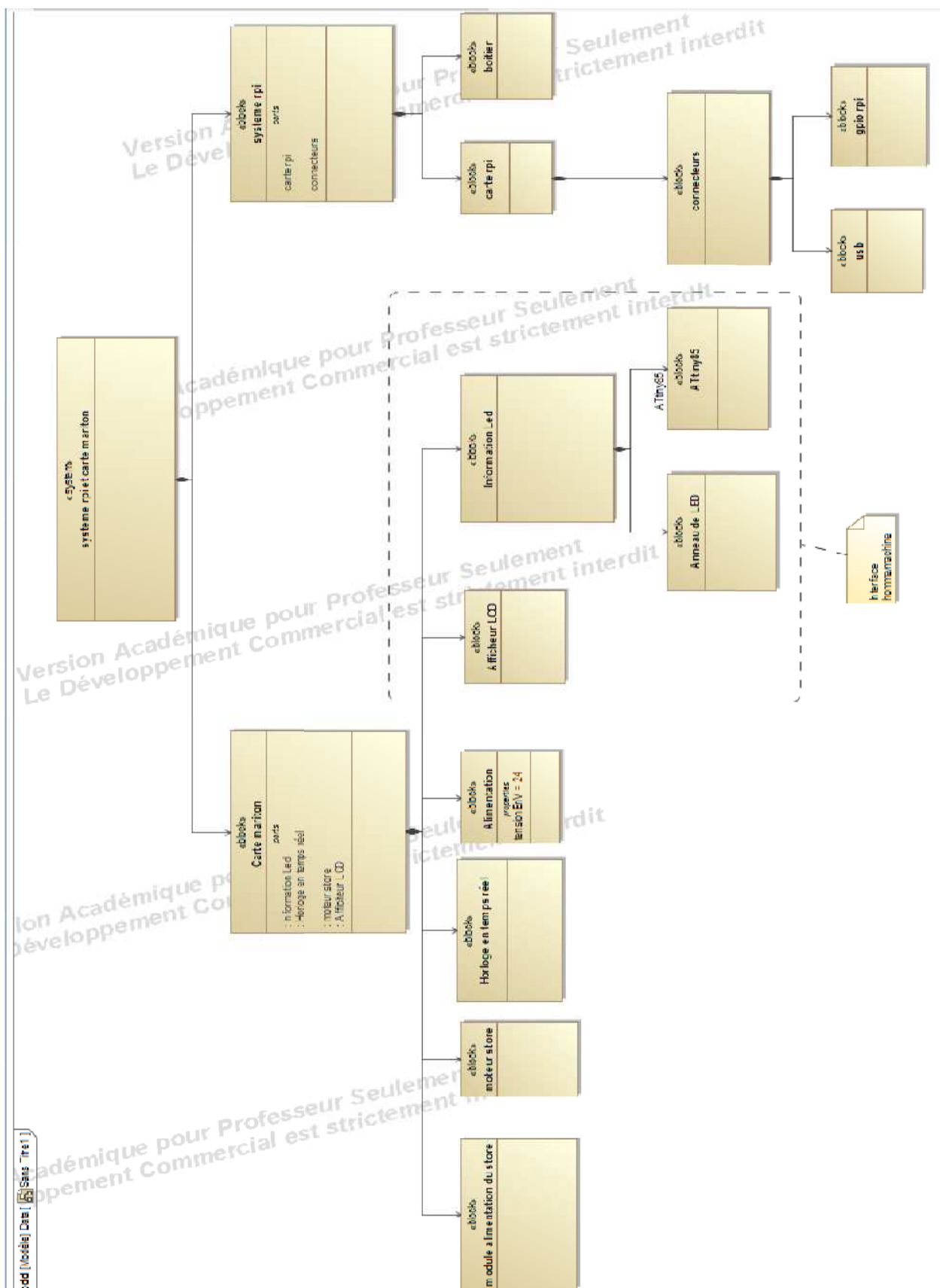
2/ PLANIFICATION PREVISIONNELLE

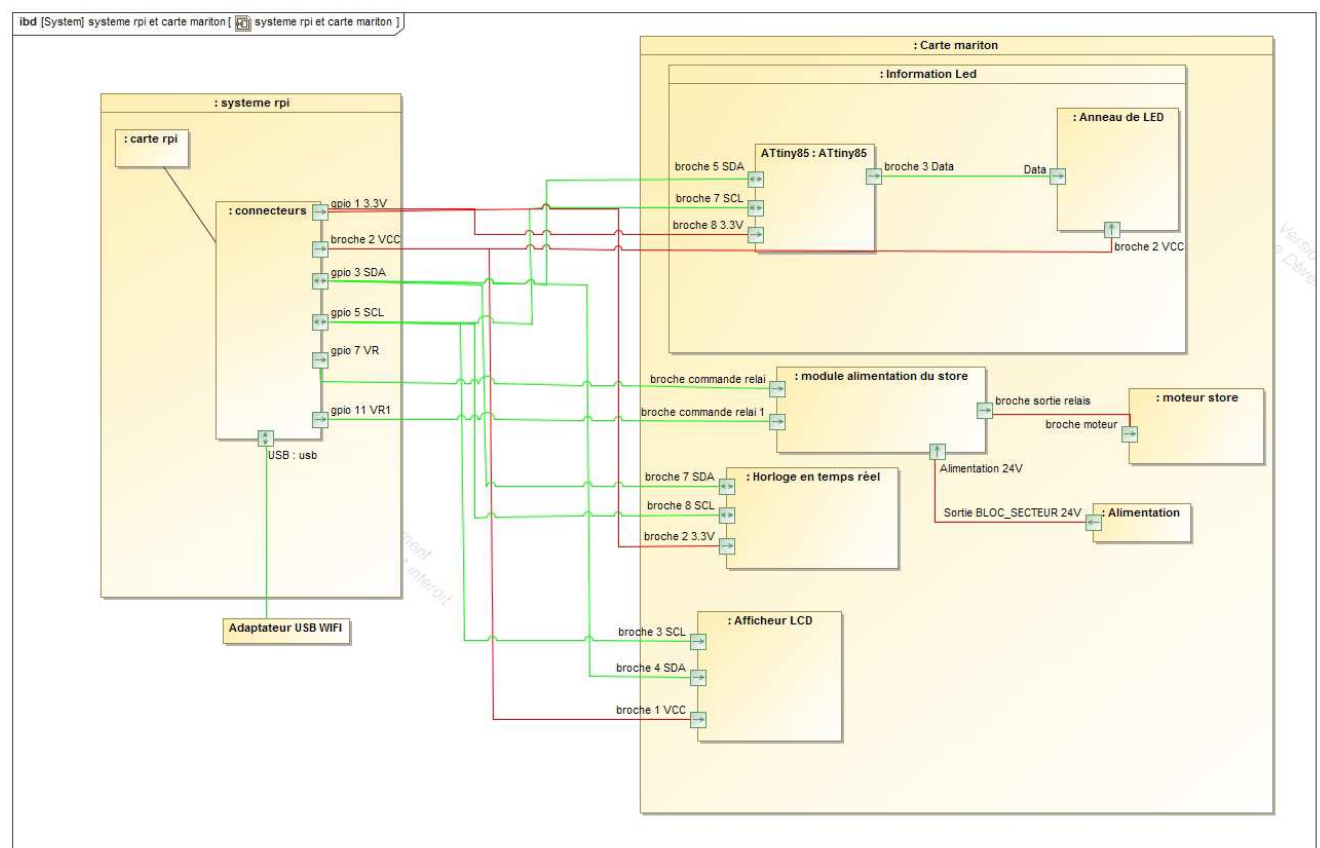
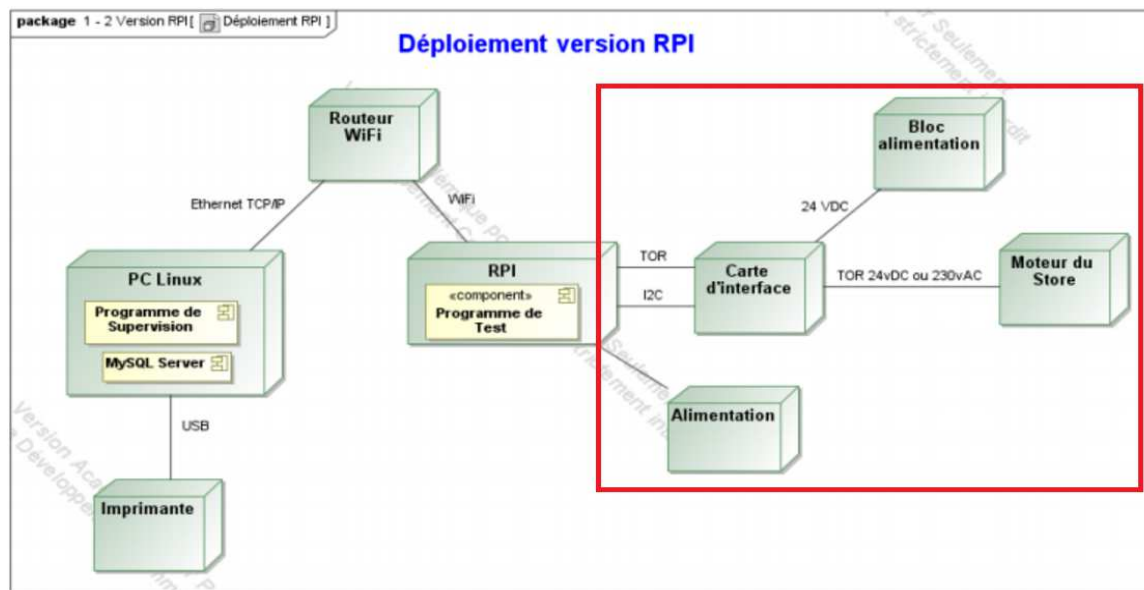


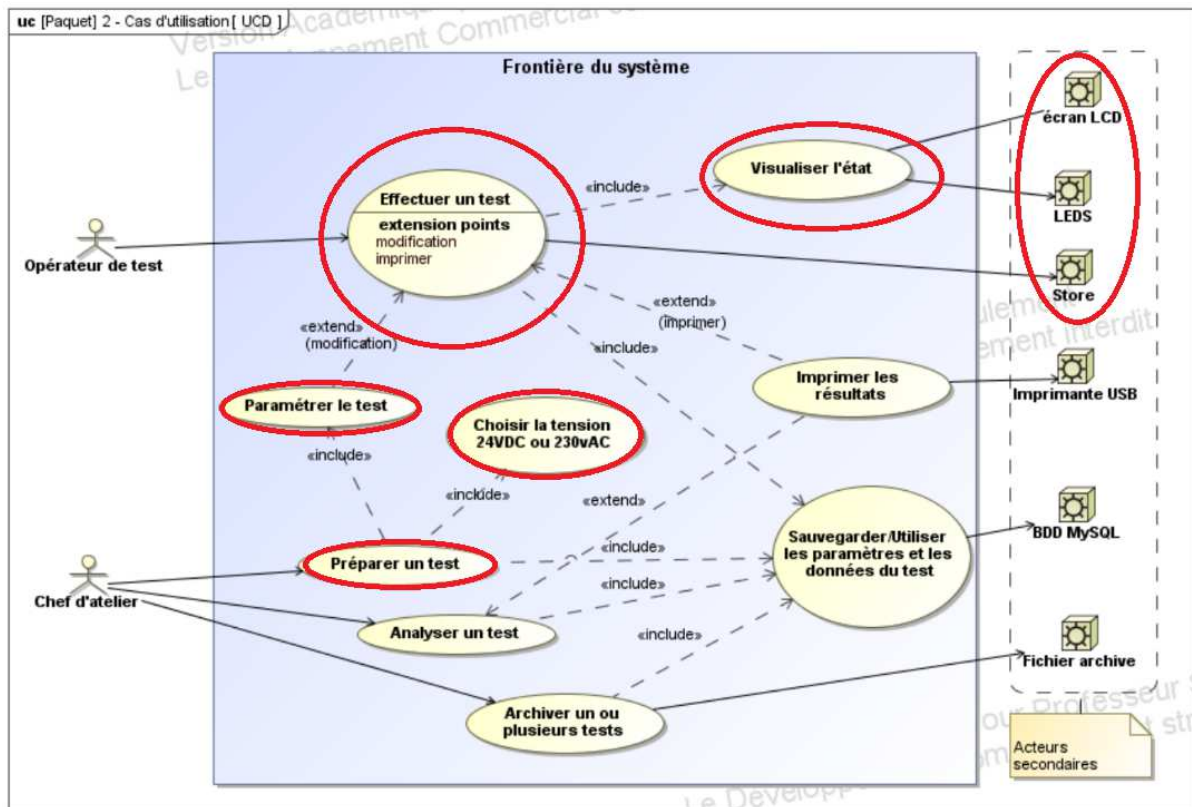
3/ PLANIFICATION REELLE



4/ DIAGRAMMES







5/ MISE EN ŒUVRE DU CONNECTEUR WIFI SUR LA RASPBERRY PI

La mise en œuvre de l'adaptateur est passé tout d'abord par l'installation du pilote, on j'ai rencontré un problème de taille aucun pilote ne pouvait être installé. L'adaptateur wifi était déjà par défaut un des matériels utilisable sur la raspberry, le seul problème était donc de savoir comment l'activer, ce qui fut rapide à trouver, il a suffi en rentrant quelques commandes dans une console pour savoir si le matériel a été identifié puis d'ajouter dans le fichier de configuration du wlan0 donc de la connexion au réseau sans fil. Pour finir il a fallu paramétrer la connexion au wifi de l'adaptateur ce que peut ce fait en modifiant dans le fichier de ses paramètres de connexion et modifier son SSID (le nom du réseau) puis la clé de connexion (clé PSK) (qui doit obligatoirement être présente). Sa connexion est en USB.



Aide pour l'installation de l'adaptateur USB Wifi :

Pas besoin d'installer un logiciel l'adaptateur USB Wifi est déjà par défaut reconnu par la RPI, Il faut simplement l'activer

Premièrement il faut vérifier si il est détecté dans une console faire un `lsusb` et vérifier la présence de Edimax Technology Co,

Puis faire un `lsmod` dans la même console pour vérifier si le pilote est chargé dans la même console et vérifier la présence du module `8192cu` de poids `485042` utilisé par `0`,

Puis faire un `iw config` pour vérifier si l'adaptateur `wlan0` est présent.

Ouvrir le fichier interfaces via la console

```
sudo nano /etc/network/interfaces.conf
```

Et vérifier la présence des ses lignes permettant la configuration du `wlan0`

```
auto wlan0
```

```
allow-hotplug wlan0
```

```
iface wlan0 inet manual
```

```
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
```

Après modification de ce fichier faire la commande

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Pour créer ou modifier le fichier de configuration de la connexion du `wlan0`

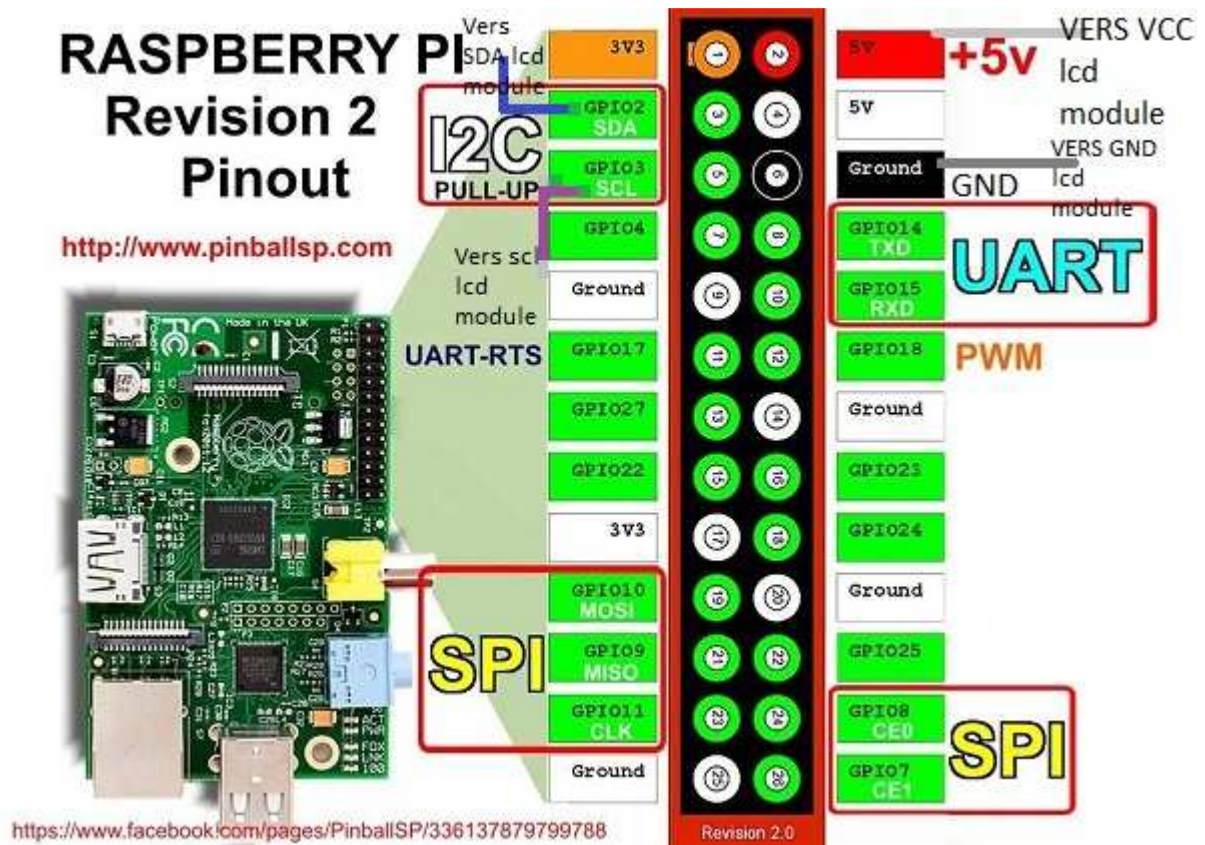
```
network={  
    ssid="_SSID_"  
    proto=RSN  
    key_mgmt=WPA-PSK  
    pairwise=CCMP TKIP  
    group=CCMP TKIP  
    psk="_WPA_KEY_"  
}
```

Après cela redémarrer l'adaptateur wifi avec la commande

```
sudo ifup wlan0
```

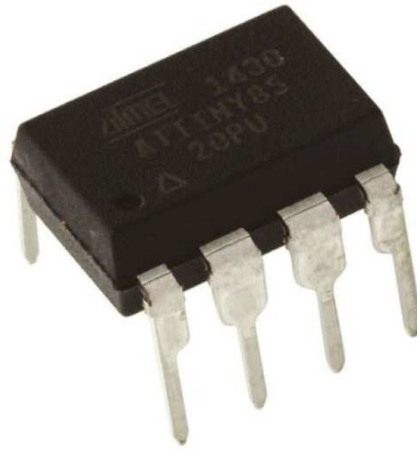
Et pour vérifier la bonne connexion du `wlan0` faire la commande

```
ifconfig wlan0
```

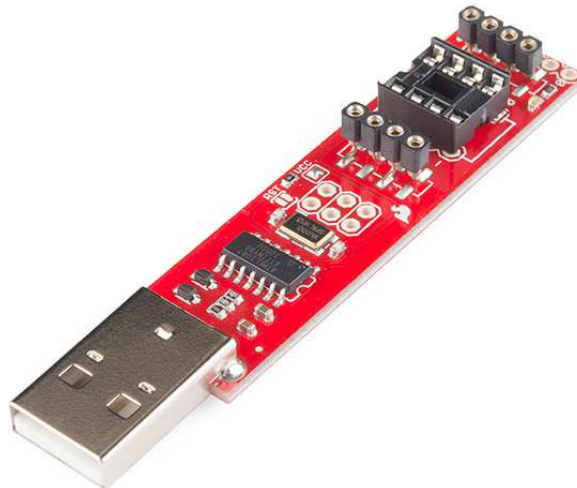



7/ MISE EN ŒUVRE DE L'ATTINY85

La mise en œuvre de l'ATtiny se fait en deux étapes, la première est le paramétrage avec arduino et la deuxième est le passage en mode esclave l'ATtiny pour que la Raspberry puisse l'utiliser en tant que telle afin de lui permettre la gestion de l'anneau de LEDs.

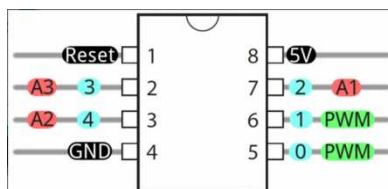


Paramétrage avec arduino



La programmation c'est donc faite avec le logiciel arduino mais sans carte arduino, avec le programmeur ci-dessus.

Brochage de l'ATtiny :



Quelques données sur l'ATtiny

Microcontrôleurs ATtiny85 tinyAVR, 8 bits :

Le microcontrôleur Atmel ATtiny85 est un circuit 8 bits basé sur l'architecture RISC améliorée AVR. Etant un circuit Atmel tinyAVR, l'ATtiny85 cumule la performance et l'efficacité de l'alimentation dans un boîtier de petite taille, parfait pour les applications dans lesquelles l'espace occupé est un élément important. En exécutant des instructions puissantes dans un seul cycle d'horloge, l'ATtiny85 atteint des cadences approchant 1 MIPS par MHz, permettant d'équilibrer la consommation d'énergie et la vitesse.

Caractéristiques du microcontrôleur

Fréquence d'utilisation : 20 MHz max.

Mémoire flash 8 ko

EEPROM 512 octets

SRAM 0,5 ko

120 instructions : la plupart des cycles d'horloge simples

32 registres d'usage général

6 GPIO

Système de débogage sur circuit debugWIRE

Programmable in-situ via le port SPI

Sources d'interruption externe et interne

Faible consommation en veille, réduction du bruit ADC et modes de mise hors tension

Circuit de reset d'alimentation amélioré

Circuit de détection de baisse de tension programmable

Oscillateur étaloné en interne

Caractéristiques du périphérique

Compteur / timer 8 bits avec générateurs d'échelle et deux canaux PWM

Compteur / timer 8 bits ultra-rapide avec générateur d'échelle distinct

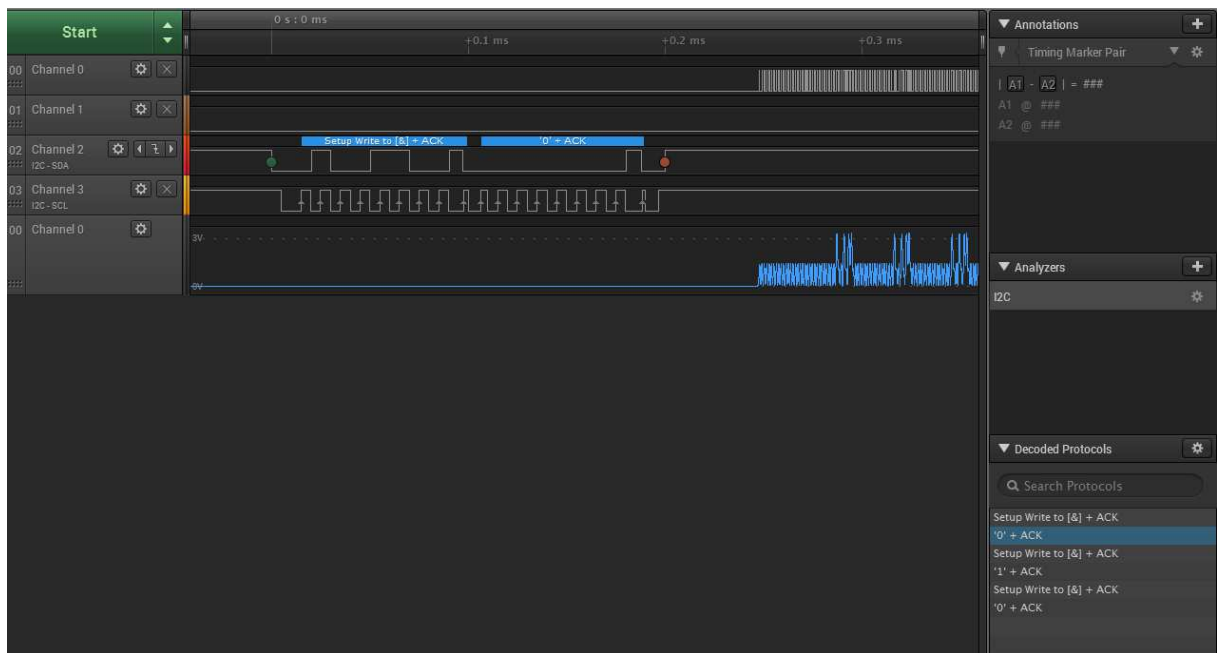
Interface série universelle (USI) avec détecteur de condition de démarrage

Convertisseur analogique-numérique (ADC) 10 bits - 4 canaux

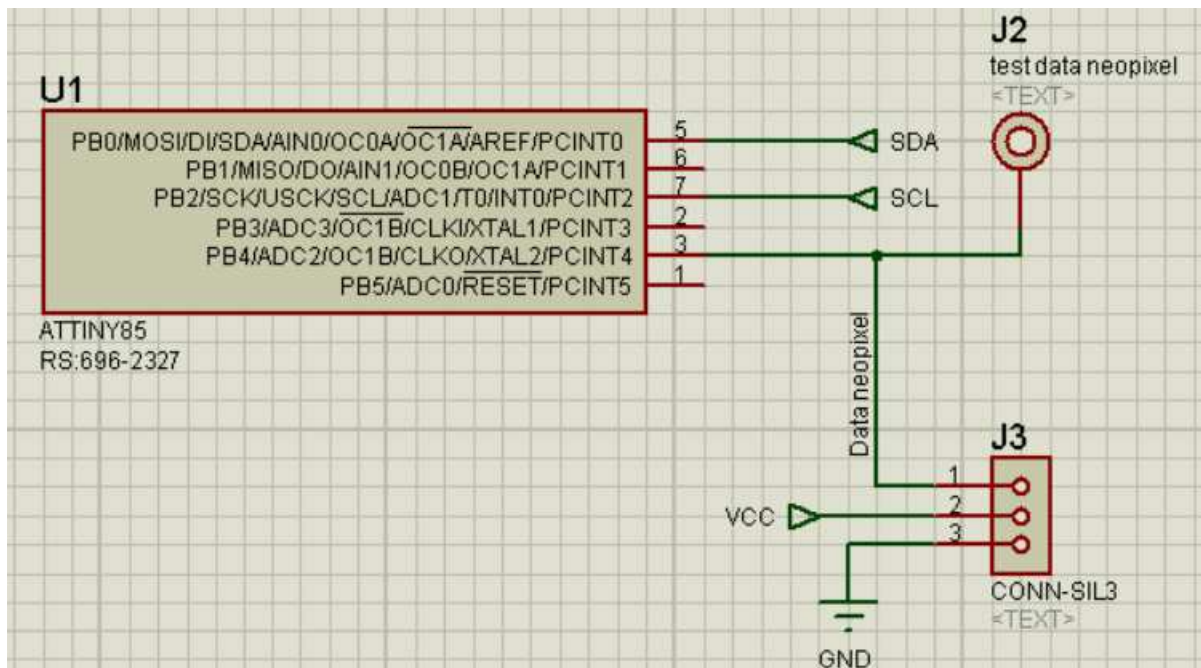
Timer Watchdog programmable avec oscillateur sur circuit séparé

Comparateur analogique intégré

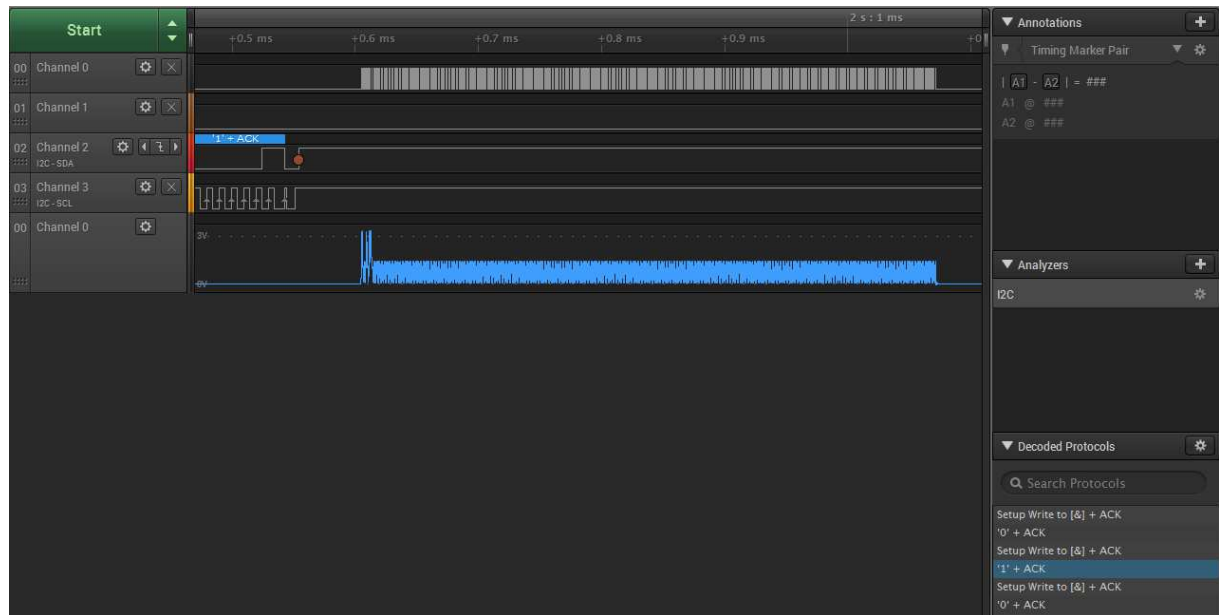
Communication de la Raspberry vers l'Attiny85



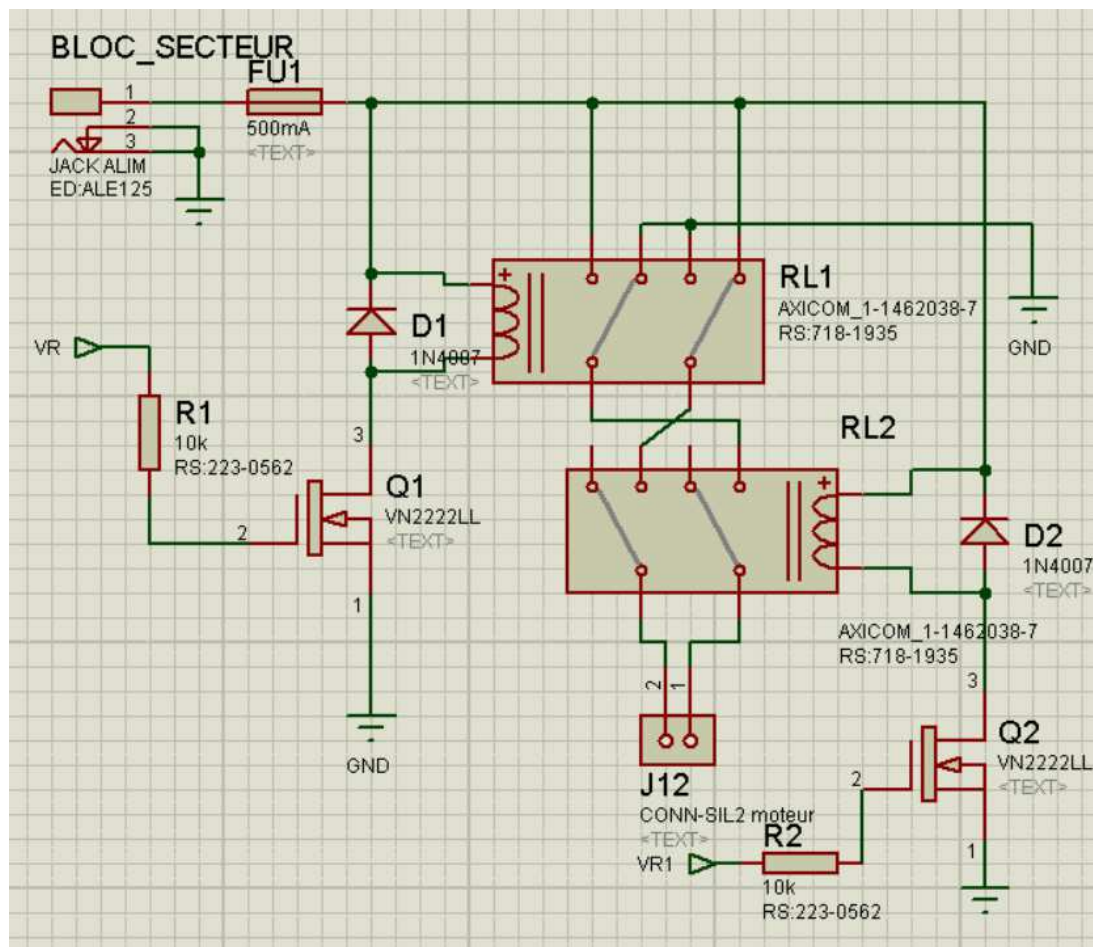
Voici le câblage de l'Attiny et de la RPI le tout comprenant l'anneau de LED (neopixel) sur Proteus



Les données de L'ATtiny vers l'anneau de LED neopixel

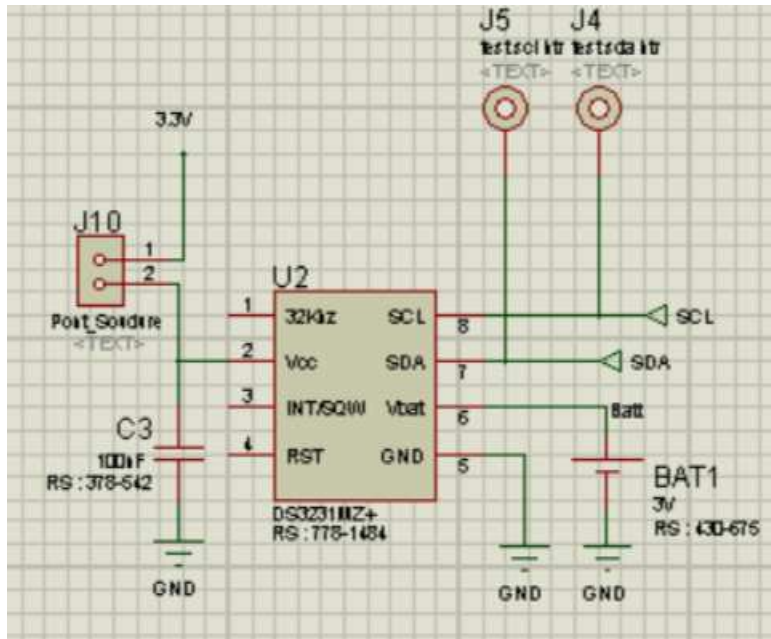


8/MONTAGE DES RELAIS ET DU MOTEUR EN 24V



Ce montage permet donc de faire tourner le moteur dans un sens ou dans l'autre et de le couper.

9/ MONTAGE DE L'HORLOGE EN TEMPS REELLE



Tutoriel d'utilisation de l'horloge en temps réel :

1) Activation de l'i2c sur la carte Raspberry pi

-Utilisez votre RPI en mode Super User :

```
pi@raspberrypi:~$ sudo su
```

-Editez le fichier **/etc/modules** avec l'éditeur nano :

```
root@raspberrypi:/home/pi# nano /etc/modules
```

-Ajoutez les lignes :

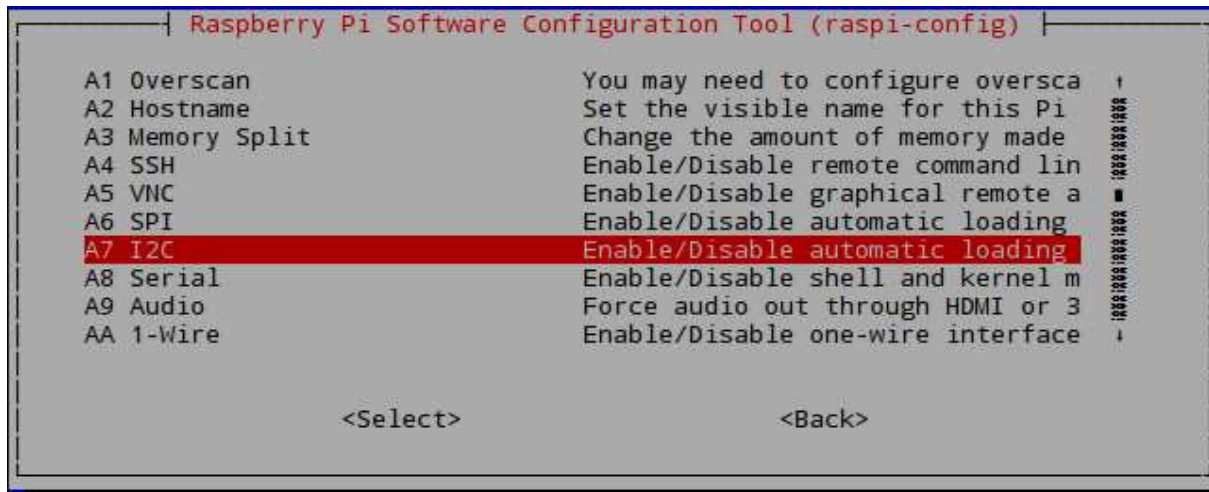
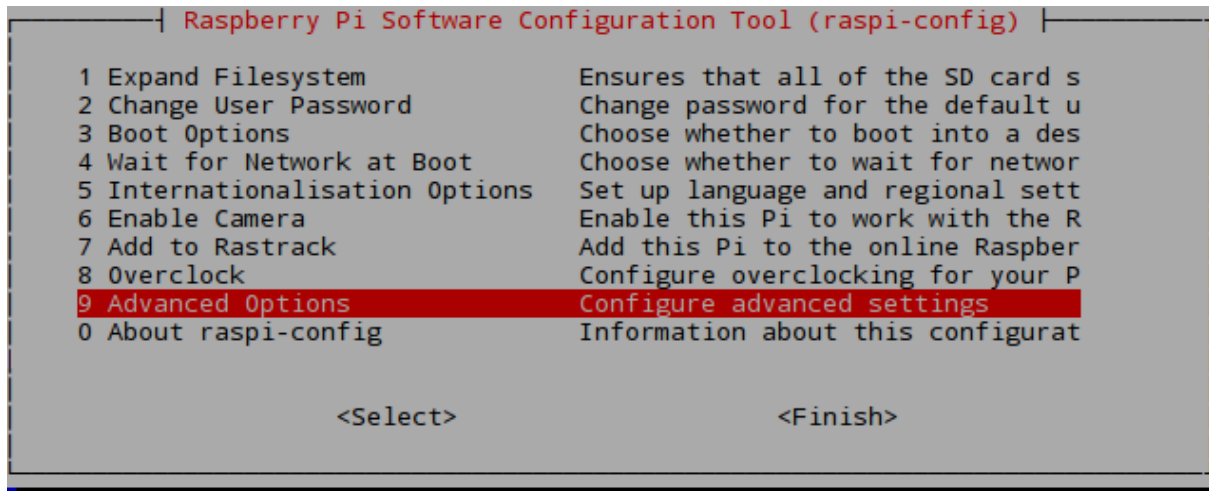
```
i2c-bcm2708
```

```
i2c-dev
```

-Sauvegardez puis lancez l'interface de configuration :

```
root@raspberrypi:/home/pi# raspi-config
```

-Entrez dans « Advanced options » et validez l'i2c



-Quittez puis redémarrez la carte :

```
root@raspberrypi:/home/pi# reboot
```

-Pour vérifier le bon fonctionnement de l'i2c sur votre carte :

```
root@raspberrypi:/home/pi# dmesg|grep i2c
```

```
root@raspberrypi:/home/pi# dmesg|grep i2c
[ 3.158266] i2c /dev entries driver
```

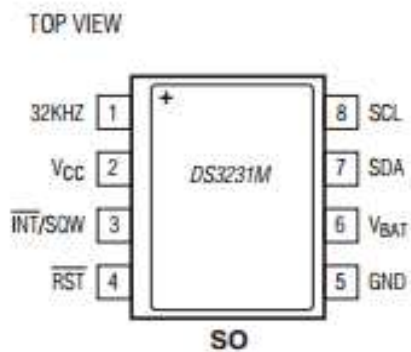
```
root@raspberrypi:/home/pi# ls /dev/i2c*
```

```
root@raspberrypi:/home/pi# ls /dev/i2c*
/dev/i2c-1
```

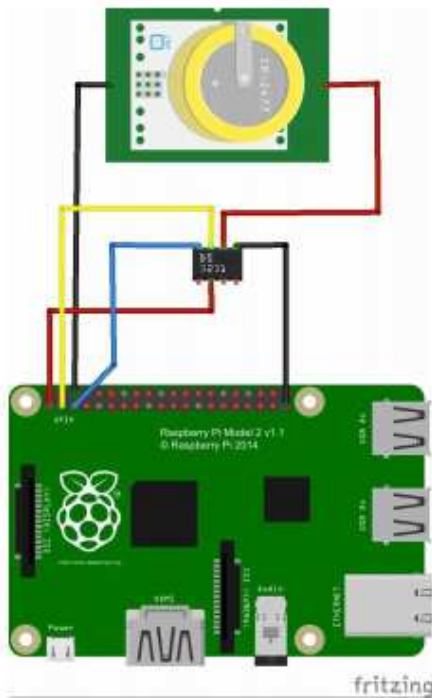
Le Bus i²c fonctionne correctement comme le montrent ces commandes.

II) Montage

Disposition des broches de la RTC : DS3231M



Pour avoir une idée du montage utilisant la RTC, voici le schéma Fritzing qui vous aidera à câbler :



Il est important de noter que sur le logiciel Fritzing la pile seule n'existe pas et est indépendante de la carte sur laquelle elle se trouve en haut du schéma. Sur votre montage, pluggez donc simplement la pile sur une breadboard.

En ce qui concerne le DS3231, ne disposant de ce composant qu'en version CMS, un typon de test a été créé afin de l'exploiter. À vous de voir si vous faites de même ou si vous utilisez le composant dans sa version non-CMS.

Broches utilisées: 1,3,5,6,25 du GPIO

III) Mise en œuvre du DS3231MZ+ :

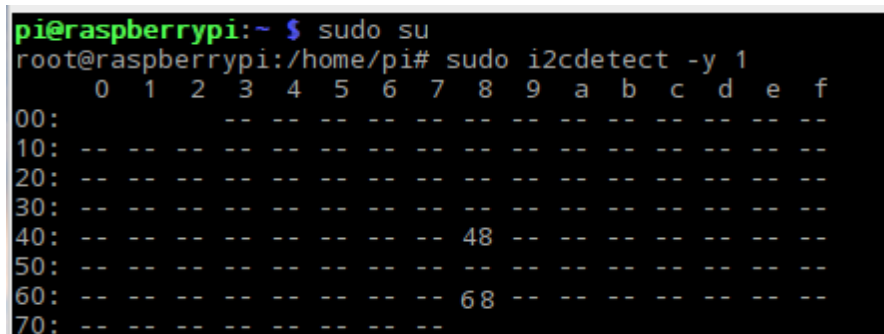
Une fois le montage réalisé, vérifier que le circuit est bien détecté :

```
root@raspberrypi:/home/pi# apt-get install i2c-tools
```

Le résultat obtenu est les installations faites sur votre RPI.

```
root@raspberrypi:/home/pi# i2cdetect -y 1
```

Permet de détecter le DS3231MZ+ et que Le Bus i2c est exploitable



```
pi@raspberrypi:~$ sudo su
root@raspberrypi:/home/pi# sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  48  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  68  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

-Le caractère ASCII 68 correspond à l'adresse du composant, c'est donc bien l'horloge temps réel DS3231MZ+ qui est détectée.

On trouve dans la documentation de la RTC l'adresse 0b1101000 ce qui correspond en hexadécimal au caractère 0x68 que nous obtenons.

-Afin de vérifier maintenant le bon fonctionnement du DS3231MZ+, suivez les instructions suivantes qui vont interroger le capteur de température interne de la RTC qui permet de retourner sa propre température car le DS3231MZ+ est aussi un capteur de température

Le résultat de la conversion est accessible avec l'adresse 0x11 sur deux octets, on a donc besoin de renseigner dans la commande le numéro du BUS (ici 1), le numéro et l'esclave (ici 0x68) et le registre à lire ou écrire (ici w pour écrire) :

```
root@raspberrypi:/home/pi# i2cget -y 1 0x68 0x11 w
```

Le résultat de cette commande est donc une valeur hexadécimale composée de deux octets. Vous pouvez essayer de faire évoluer la température du composant et relancer la commande pour vérifier que le résultat diffère.

IV) Initialisation de la RTC

Pour cela, il faut écrire la valeur 0x04 dans le registre « control » qui est le 0x0E ainsi que la valeur 0x08 dans le registre « status » qui est le 0x0F :

```
root@raspberrypi:/home/pi# i2cset -y 1 0x68 0x0E 0x04 b
```

```
root@raspberrypi:/home/pi# i2cget -y 1 0x68 0x0F b
```

0x88 (résultat de la commande)

```
root@raspberrypi:/home/pi# i2cset -y 1 0x68 0x0F 0x08 b
```

```
root@raspberrypi:/home/pi# i2cget -y 1 0x68 0x0F b 0x08
```

À présent vous pouvez lire les secondes, minutes, heures, jour, mois et année en questionnant le bon registre, registres allant de 0x01 à 0x06 et correspondant à chacune des valeurs ci-dessus dans l'ordre. Par exemple la commande :

```
root@raspberrypi:/home/pi# i2cget -y 1 0x68 0x06 b
```

Va permettre d'obtenir la valeur hexadécimale correspondant à l'année.

Vous allez maintenant pouvoir installer l'horloge temps réel à la Raspberry pour qu'elle démarre avec l'heure de celle-ci et non celle d'origine et sans internet.

Pour cela brancher la Raspberry à internet pour qu'elle soit à l'heure puis taper :

```
root@raspberrypi:/home/pi# echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

(déclaration nouvelle esclave i2c)

(Notez que dans la commande figure la référence du composant DS1307 qui est une autre horloge temps réel mais qui est compatible avec notre DS3231MZ+)

```
root@raspberrypi:/home/pi# hwclock -r (lecture de l'heure sur la RTC)
```

```
root@raspberrypi:/home/pi# date (lecture de l'heure sur la raspberry)
```

```
root@raspberrypi:/home/pi# hwclock -w (écriture de l'heure de la raspberry sur la RTC)
```

```
root@raspberrypi:/home/pi# hwclock -r (lecture de l'heure sur la RTC pour vérifier le bon changement )
```

```
root@raspberrypi:/home/pi# hwclock -r
mar. 24 janv. 2017 15:06:56 CET -0.308749 seconds
```

La RTC est désormais à l'heure et le restera même si elle est déconnectée de la Raspberry pi ou si elle n'est plus alimentée grâce à la pile

Maintenant nous allons affecter cette RTC au démarrage de la Raspberry pour que celle-ci dispose de l'heure y compris déconnecté du réseau et d'internet

Pour cela il faut changer certains fichiers de la Raspberry :

-Editez le fichier **/etc/modules** avec l'éditeur nano :

```
root@raspberrypi:/home/pi# nano /etc/modules
```

Rajoutez la ligne :

```
rtc-ds1307
```

Sauvegardez et quittez.

-Editez le fichier **/etc/rc.local** avec l'éditeur nano :

```
root@raspberrypi:/home/pi# nano /etc/rc.local
```

Avant la ligne :

```
exit 0
```

Rajoutez les lignes :

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

```
sudo hwclock -s
```

Sauvegardez et quittez.

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
sudo hwclock -s
```

Les commandes faites jusqu'à présent suffisent pour la version Jessie light de Raspberry.

Pour la version Jessie basique il faut en plus rajouter ces quelques commandes :

Donc pour finir dans le fichier **/lib/udev/hwclock-set**

```
root@raspberrypi:/home/pi# nano /lib/udev/hwclock-set
```

```
if [ -e /run/systemd/system ] ; then
```

```
exit 0
```

```
fi
```

Mettre en commentaire

```
#if [ -e /run/systemd/system ] ; then  
#exit 0  
#fi
```

Sauvegardez et quittez,

Et écrire dans le fichier **/boot/config.txt**

```
root@raspberrypi:/home/pi# nano /boot/config.txt
```

Rajouter la ligne :

```
dtoverlay=i2c-rtc,ds-1307
```

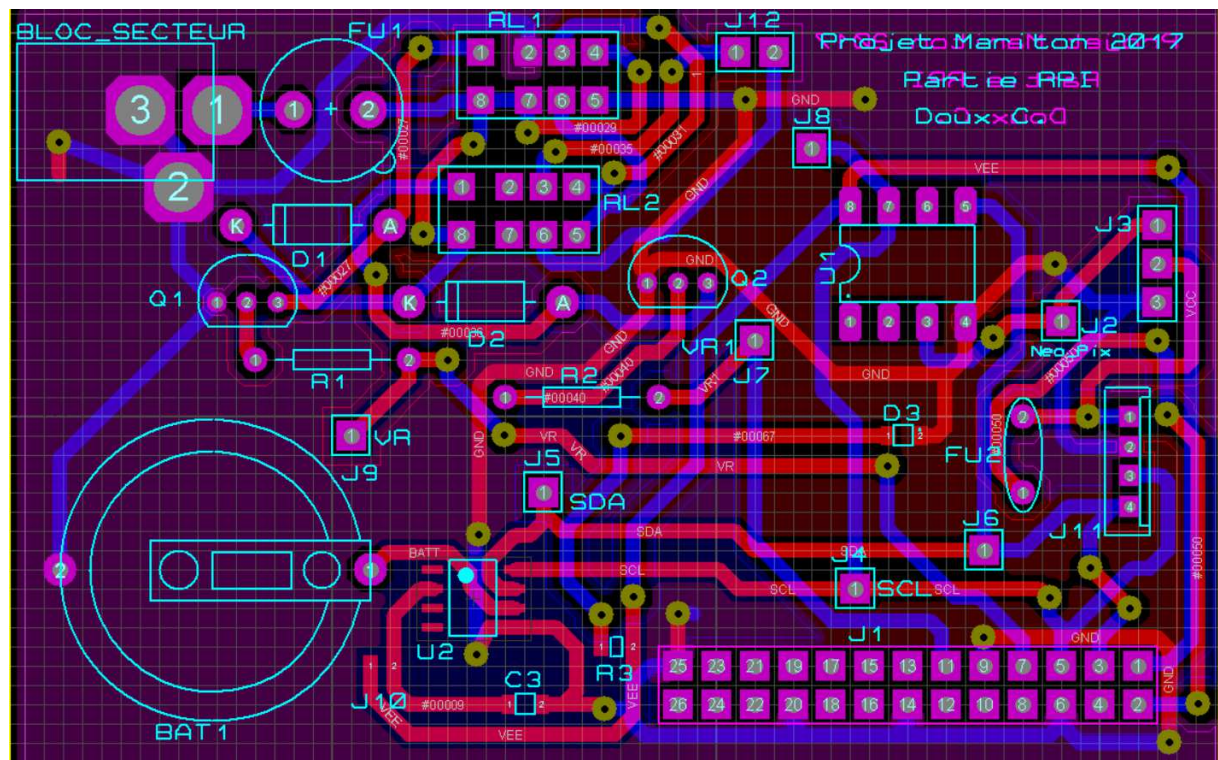
Sauvegardez et quittez.

Vous pouvez maintenant essayer :

Éteignez votre Raspberry attendez un moment et redémarrez la et toujours connecté a la RTC mais déconnecter du réseau et la Raspberry sera à la bonne heure et non à l'heure d'où elle s'est éteinte comme auparavant.

1. Fiche de rapport de lecture croisée.			Réf.FRLC-XX
Projet	Auteur	Date	
	Johann crotte	XX/XX/2017	
Produit		Durée de la réunion	
Horloge en temps réelle		X	
Lecteur(s)		Contrôleur des corrections	
Doux Christopher		X	
Liste des erreurs détectées			
Une mise a jour a était effectuée donc présence de quelques variations dans les interfaces graphiques. Mais aucune présence d’erreur majeure qui pourrait entraver le fonctionnement des composants. Ci-joint le document mis à jour pour une version plus récente de la raspberry pi.			

11/ MONTAGE FINAL SCHEMA DE FABRICATION



12/ DEPANNAGE

Problèmes :

Aucun relais en fonction : vérifier le fusible FU1.

Aucun affichage sur l'écran LCD décâbler-recâbler et redémarrer la RPI.

13/ ANNEXES

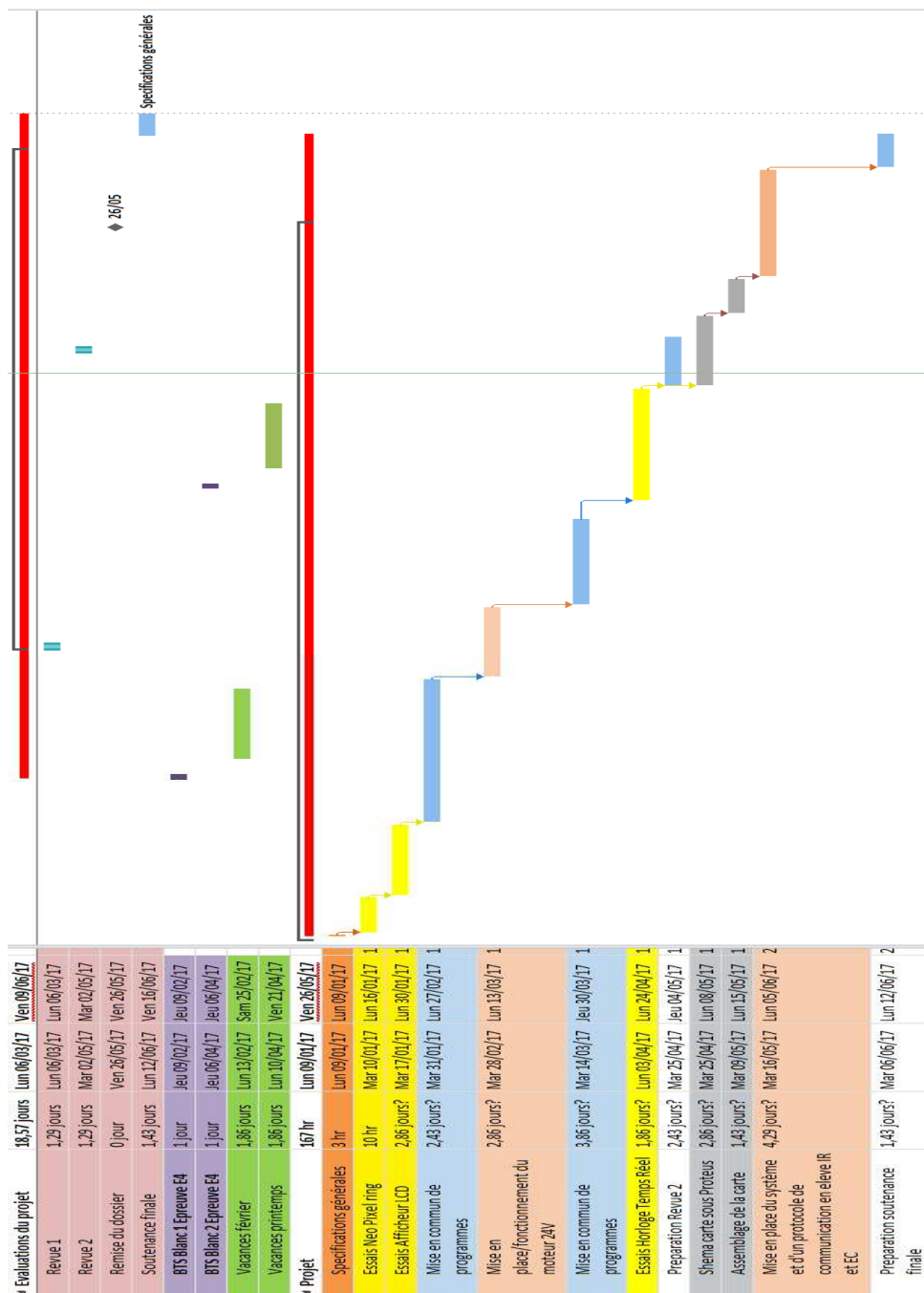
Carte rpi		Client :									
Repère	Désignation du matériel	Valeur (Référence)	Tol.±	Equivalent @ Fournisseur	Boîtier	Recommandation câblage	Qté Stock	Condition	ref rapide "RS" (FA)	Prix UHT	Prix THT
R1,R2	Résistance CMS série CRG	CRG0805F1C	1%	TE Connectivity	805		2		50 : "223-0562"	0,002	0,004
Q1,Q2	MOSFET, Canal N, 3 broches	2N2222LL-G		Microchip	TO-32		2		25 : "879-3274"	0,292	0,584
D1,D2	Diode de commutation	1N4007GPEH3/54		Vishay	DO-204AL		2		50 : "902-6474"	0,126	0,252
C3	Condensateur céramique multicouche MLCC	30032F104 MGE8	20%	Yageo	805		1		25 : "378-542"	0,070	0,070
U1	Microcontrôleur	ATTINY85-20PU		Atmel	PDIP		1		1 : "696-2327"	1,660	1,960
U2	Horloge en temps réel (RTC)	DS3231M2+		Maxim Integrated	SOTIC		1		1 : "778-1484"	6,120	6,120
RL2	Relais sans accrochage 2 RT Montage sur CI	1-1462038-7		TE Connectivity			2		1 : "718-1938"	2,670	5,340
BAT1	Pile bouton Panasonic	CR-2032HFH		Panasonic			1		1 : "430-575"	2,010	2,010
R3	Résistance CMS série CRG	CRG0805F3C	1%	TE Connectivity	805		1		50 : "223-0354"	0,036	0,036
D3	LED CMS 2.0	3LG B371		OSRAM	805		1		50 : "654-5773"	0,051	0,061

28

VI/ Partie personnelle de Benkheddouma Jordan

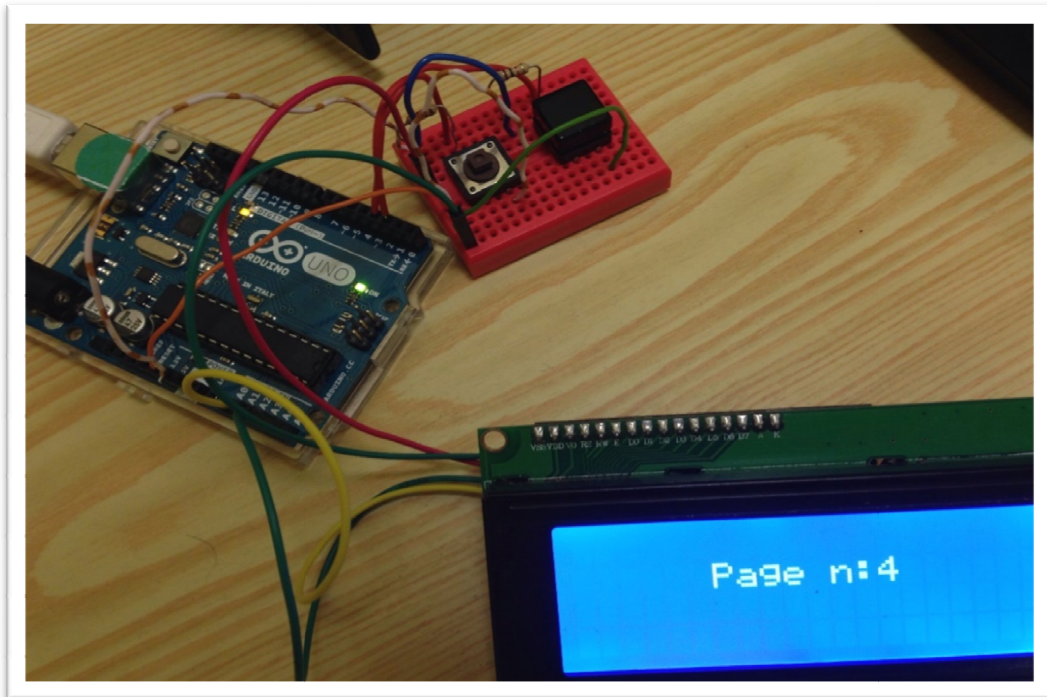
1. TACHES A EFFECTUER :

- Concevoir/Réaliser/Tester une carte d'extension pour une carte Arduino UNO.
- Concevoir/Coder/Tester la partie logicielle se trouvant sur la carte Arduino.
- Convenir avec les étudiants IR, d'un protocole d'échange entre la carte Arduino et l'ordinateur superviseur.
- Développer une application graphique permettant de vérifier le bon fonctionnement de la carte d'extension.

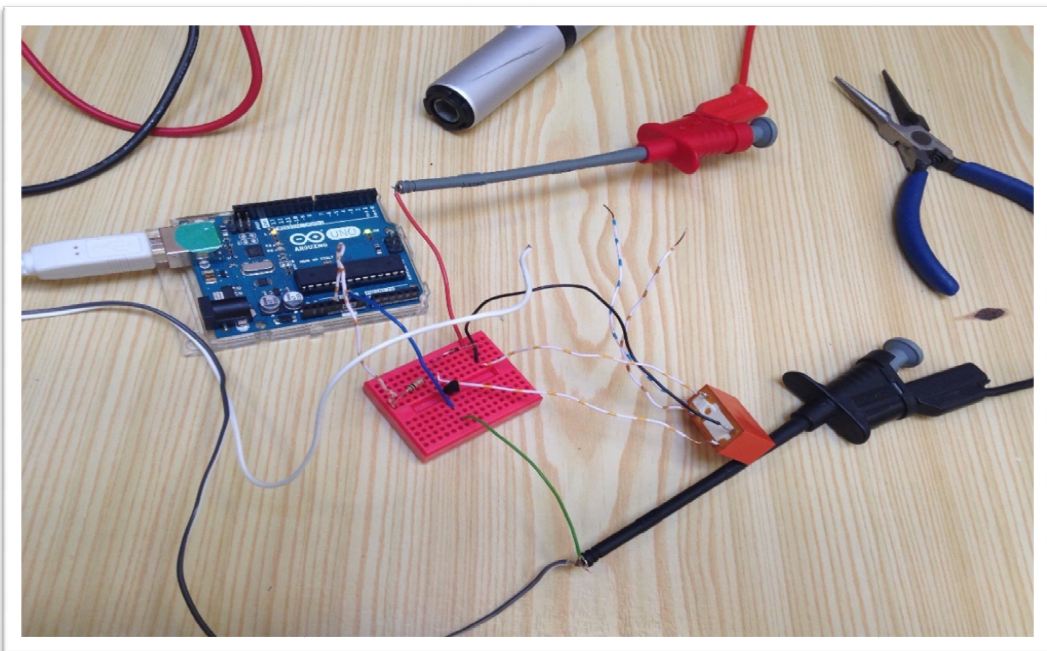


2. TRAVAIL EFFECTUE DURANT LA PERIODE PRECEDANT LA PREMIERE REVUE DE PROJET :

- Cette période a notamment été utilisée afin de se familiariser avec l'afficheur LCD et le NeoPixel Ring.
- Début de mise en œuvre d'un programme de test qui a pour but de rassembler certains des programmes déjà créés précédemment.



(Ici un programme qui a pour but de changer la page affichée en fonction de deux boutons.)



(Ici le début du montage permettant de contrôler le moteur via la carte Arduino).

3. TRAVAIL EFFECTUE DURANT LA PERIODE PRECEDANT LA DEUXIEME REVUE DE PROJET :

- Mise en place de la l'horloge temps réel (**RTC : Real Time Clock**).
- Mise en commun de l'horloge temps réel, de l'afficheur LCD et de l'anneau de LEDsNeoPixel Ring.
- Schéma de la carte sous Proteus.

4. PROBLEMES RENCONTRES :

- Problème dans la librairie de l'afficheur LCD avec Arduino.

- Solution normalement utilisée :

```
lcd.setCursor(9,0);  
lcd.print(hour);  
lcd.print(":");  
lcd.print(minute);  
lcd.print(":");  
lcd.print(second);  
delay(1000);
```

- Solution possible mais qui comporte des problèmes :

```
sprintf(Time, "%d:%d:%d", hour, minute, second);  
for (int a = 0; a < sizeof(Time); a++)  
{  
    lcd.setCursor(9+a,0);  
    lcd.print(Time[a]);  
}  
delay(1000);
```

- Solution **retenue** qui comporte des faiblesses et qui est bien plus complexe :

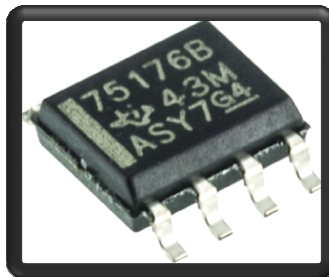
```
char Heure[] = byte hour;
char Minute[] = byte minute;
char Seconde[] = byte second;

for (int a = 0; a < sizeof(Heure); a++)
{
    lcd.setCursor(9+a,0);
    lcd.print(Heure[a]);
}
lcd.print(":");
for (int a = 0; a < sizeof(Minute); a++)
{
    lcd.setCursor(+a,0);
    lcd.print(Minute[a]);
}
```

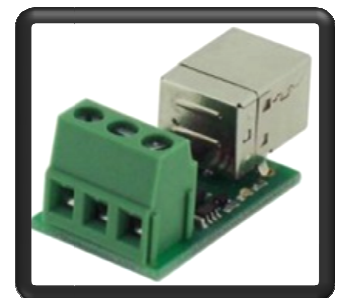
```
lcd.print(":");
for (int a = 0; a < sizeof(Seconde); a++)
{
    lcd.setCursor(+a,0);
    lcd.print(Seconde[a]);
}
delay(1000);
for (int a = 0; a < 8; a++)
{
    lcd.setCursor(9+a,0);
    lcd.print("");
}
```

5. TRAVAIL EFFECTUE AVANT LA REVUE FINALE :

- Utilisation du composant SN75176, pour la conversion RS232 / RS485.



- Achat d'un convertisseur UB/RS485 :



- Ici un programme simple qui me permet de communiquer, grâce au logiciel **TeraTerm**, entre un ordinateur et la carte Arduino.

```
#include<SoftwareSerial.h>

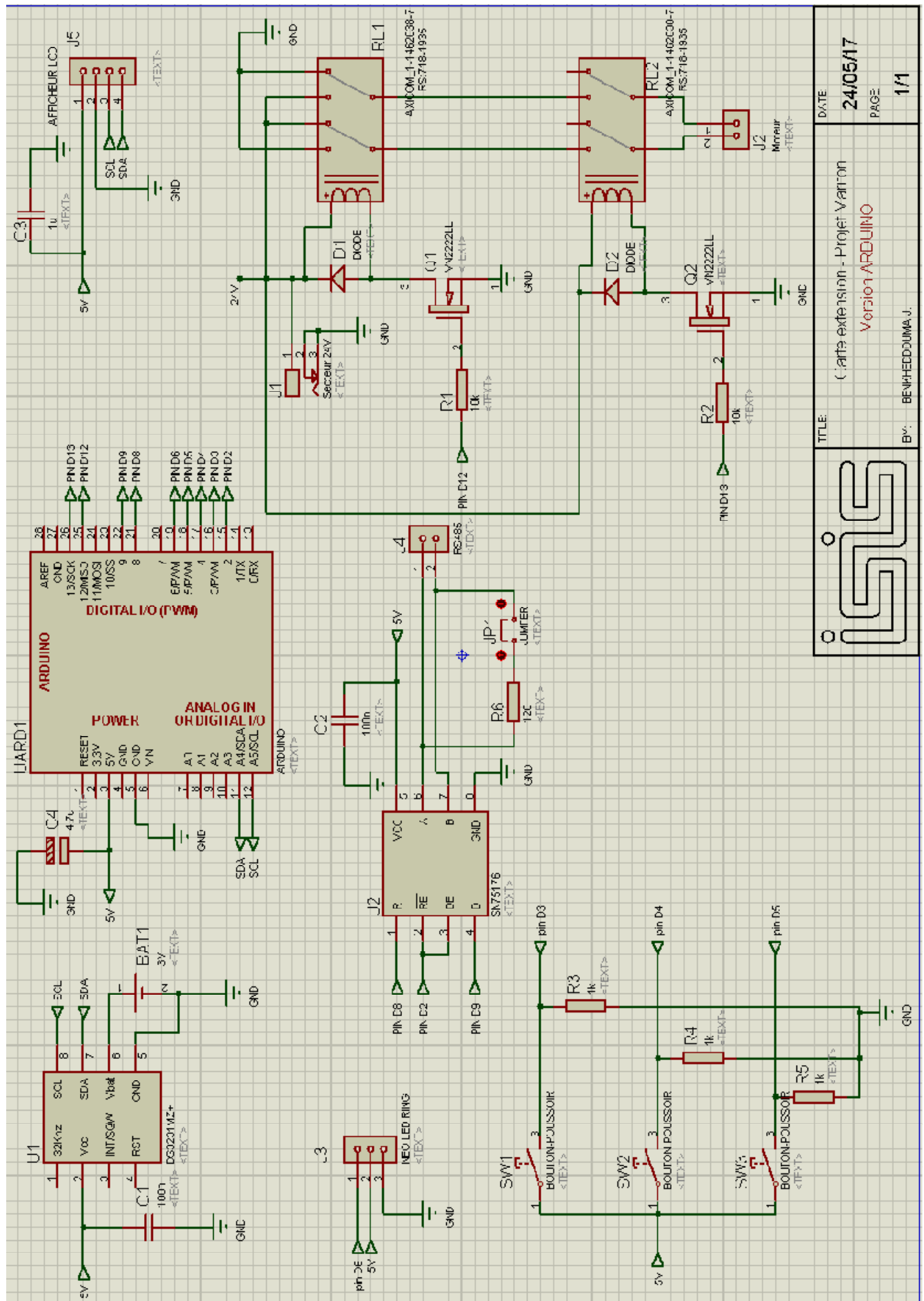
//SoftwareSerial pins
#define pinTX 4
#define pinRX 3
//MAX485 control pin
#define transmitir 2

char var;

SoftwareSerial RS485(pinRX, pinTX);

void setup() {
  //Initialize SoftwareSerial
  RS485.begin(9600);
  Serial.begin(9600);
  //Digital pins PinMode
  pinMode(transmitir, OUTPUT);
}

void loop()
{
  digitalWrite(transmitir, LOW);
  while(RS485.available() > 0)
  {
    var = RS485.read();
    Serial.print(var);
  }
}
```



TITLE: Carte extension - Projet Varron
Version ARDUINO

DATE: 24/05/17
PAGE: 1/1

BV: BEVHEDDUMA J.

VII/ Partie personnelle de Galatioto Enzo

1/ INTRODUCTION

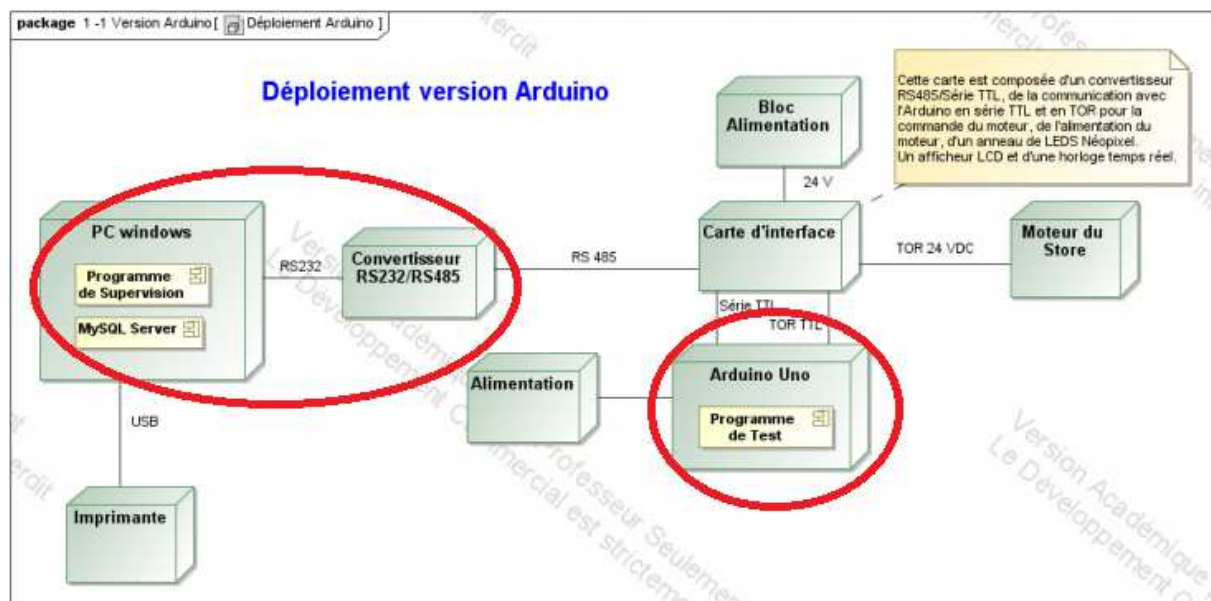
Il s'agit de prolonger les activités de la société dans le domaine de la domotique et d'acquérir une autonomie dans la fourniture de matériels et logiciels nécessaires pour l'intégration aux produits existants. Dans ce projet il s'agit plus précisément de créer un banc de test des stores.

Notre mission est de réaliser un banc de test, aussi appelé "cycleur", afin de tester sur plusieurs Milliers de cycles des stores ayant 2 types de motorisations : soit 24V continu, soit 230V alternatif. Un cycle correspond à une descente du store suivie d'une temporisation de 2 minutes (par défaut) puis d'une remontée suivie d'une temporisation de 3 minutes (par défaut). Les temporisations permettent au moteur de refroidir.

2/ MA MISSION

La version du banc d'essai sera constituée d'une carte Arduino reliée au PC par une liaison RS485, selon le diagramme ci-dessous :

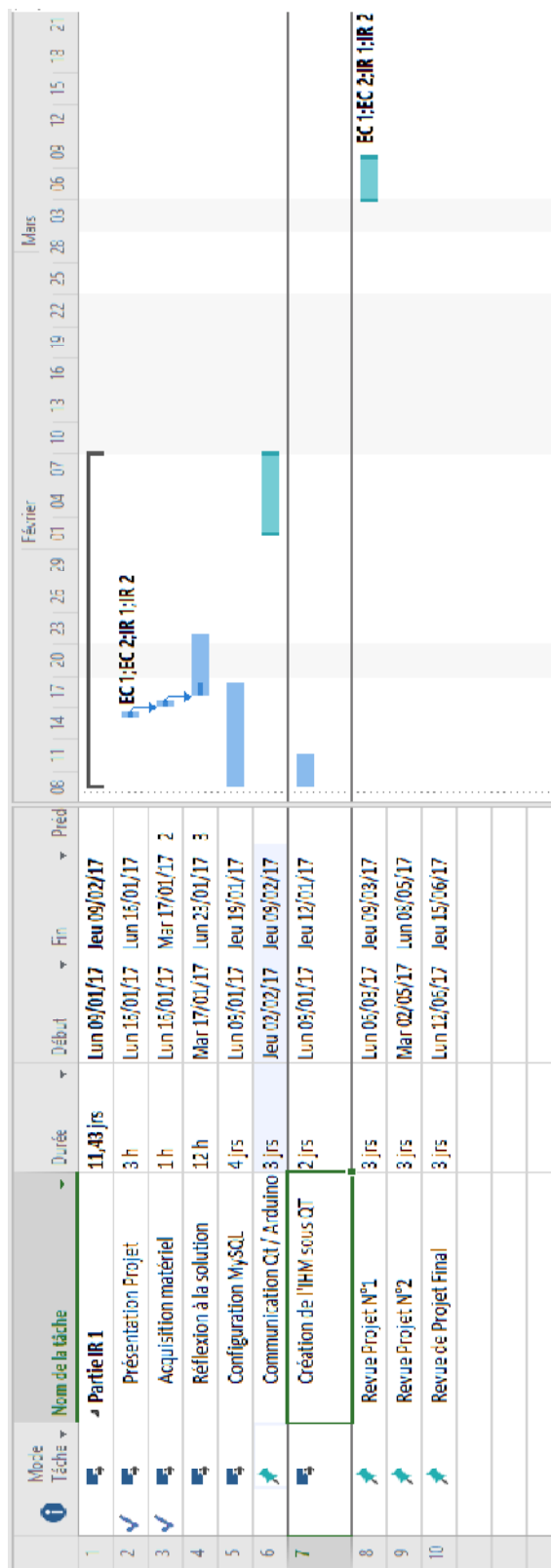
J'ai la tâche de développer les parties entourées :



Les missions abordées sont : la communication série avec QSerial entre le PC et l'arduino. L'impression ainsi que l'archivage des tests, la mise à disposition d'une interface homme machine sous Qt.

Et la supervision et le contrôle du banc de test.

3/ PLANIFICATION

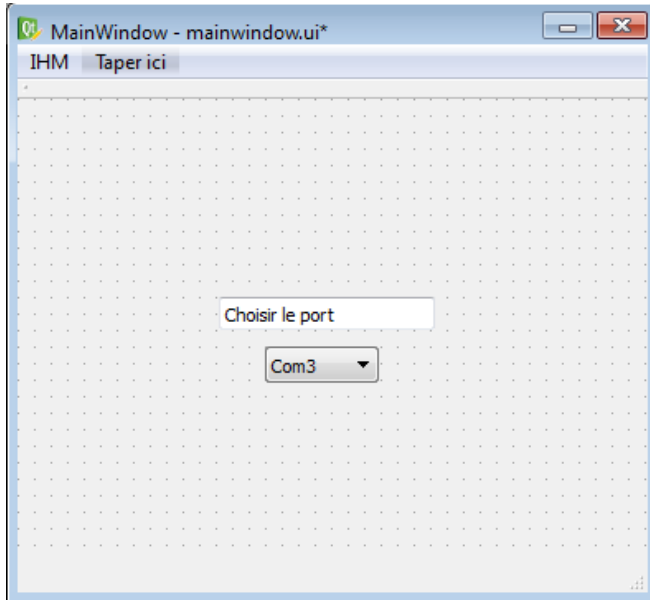


4/ REALISATION

Le module USB-RS485 utilise la puce USB FTDI FT232R pour gérer tous les protocoles USB.

Après avoir installé les pilotes et brancher le module USB-RS485 sur un port USB de rechange, j'ai du savoir à quel port COM il a été affecté.

L'USB-RS485 prend en charge des débits de données allant jusqu'à 250kbps



J'ai d'abord testé les convertisseurs grâce à une interface homme machine simplifiée avec la fonction `SerialPort.GetPortNames()` le programme saisit simplement chaque chaîne (c'est-à-dire COM3) et les boucles à placer dans la zone de liste déroulante.

5/ COMPARAISON

Pourquoi nous avons pris la décision de travailler avec du RS485 et non du RS232 ?

Le tableau suivant compare le mode de fonctionnement, le nombre total de transmetteurs et de récepteurs, la longueur maximale des câbles et la vitesse de transfert maximale.

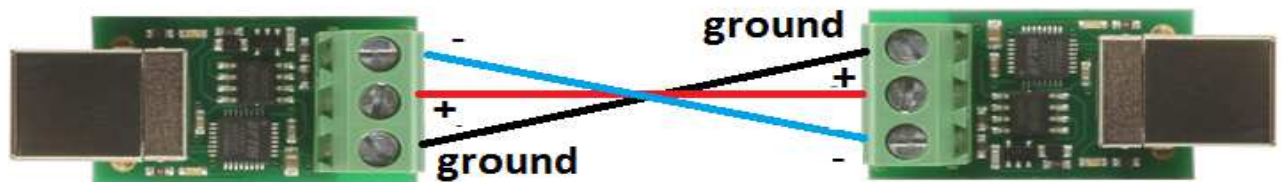
	RS232	RS485
Mode de fonctionnement	Asymétrique	Différentiel
Total de transmetteurs / récepteurs	1 transmetteur 1 récepteur	32 transmetteur 32 récepteur
Longueur maximale du câble	50 pieds	4000pieds
Vitesse maximale de transfert	160 Kbits/s peut atteindre 1 Mbit/s)	10 Mbit/s

L'interface RS-232 est l'interface série la plus répandue et est fournie comme équipement standard sur la plupart des ordinateurs de bureau compatibles avec Windows. La RS-232 ne permet qu'un transmetteur et un récepteur sur chaque ligne (voir tableau comparatif).

La RS-232 utilise aussi une méthode de transmission Full-Duplex. Certaines cartes RS-232 vendues par National Instruments supportent des vitesses de transfert allant jusqu'à 1 Mbit/s, mais la plupart des matériels sont limités à 115,2 Kbits/s.

La RS-485 est un sur ensemble de la RS-422 qui étend ses capacités.

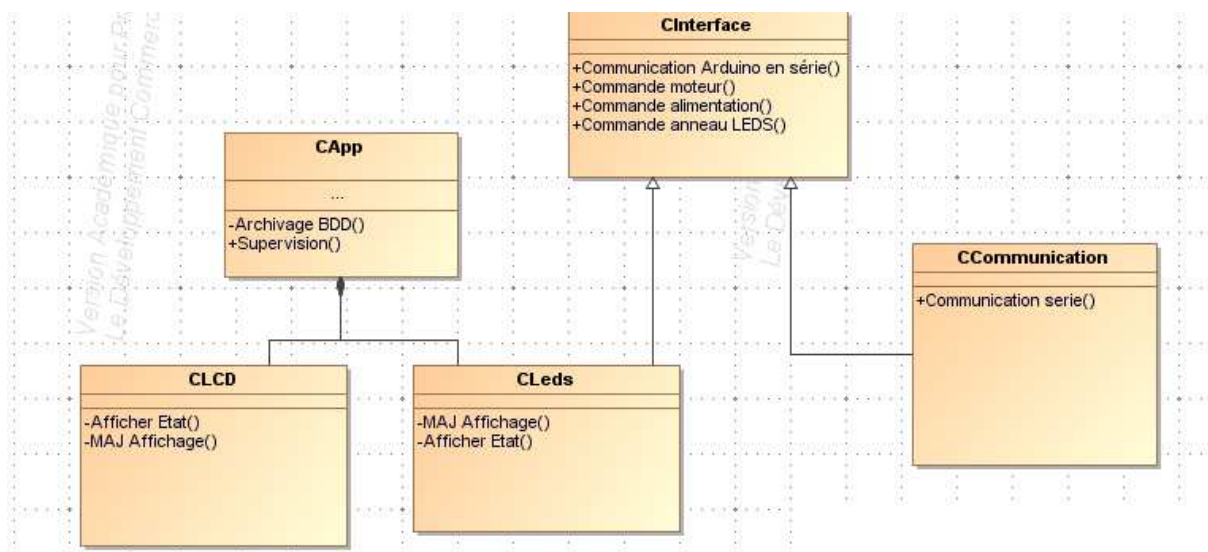
6/ BRANCHEMENTS DES CONVERTISSEUR USB-RS485



Les 2 convertisseurs USB-RS485 sont connectés entre eux grâce à 3 fils les reliant, la masse, +Ve et -Ve.

Ensuite les convertisseurs sont chacun reliés au PC.

7/ DIAGRAMME DE CLASSE



8/ PROBLEMES & SOLUTIONS

Suite à un convertisseur défectueux nous avons pris la décision de le remplacer par un convertisseur USB RS485

Nous avons pris la décision d'attribuer des paramètres par défaut au programme de test, en cas de panne ou de non réponse du superviseur (PC Windows)

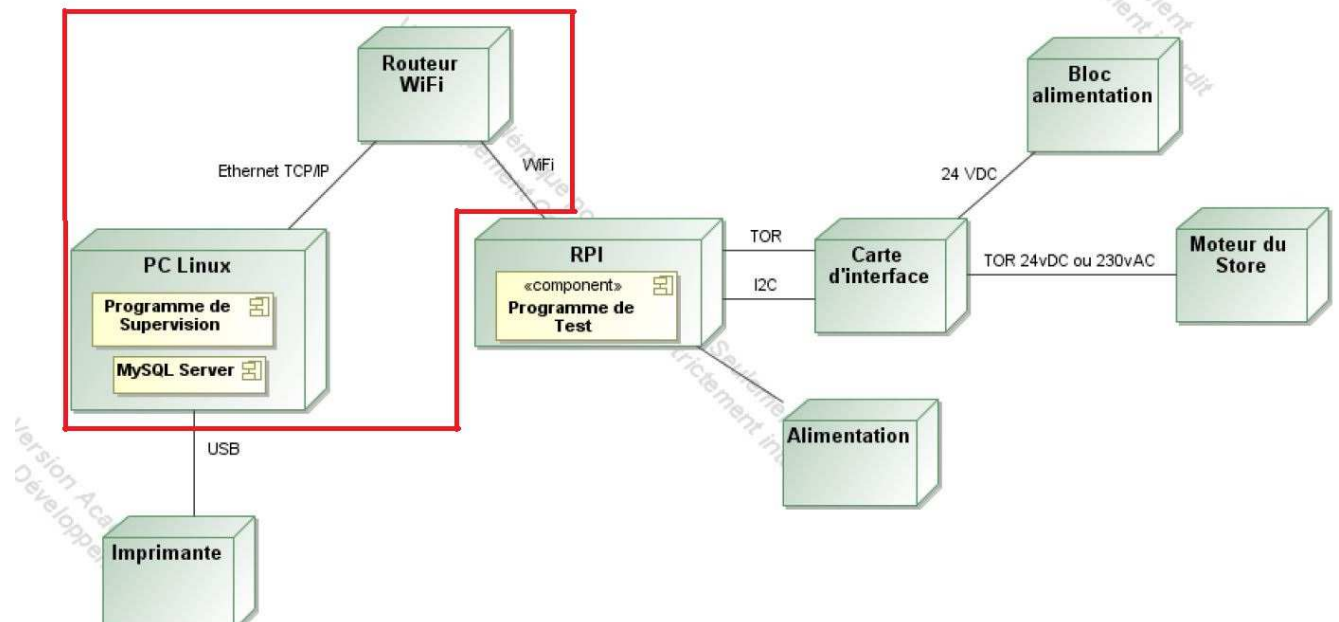
VIII/ Partie personnelle de Contassot Thomas

1/ INTRODUCTION

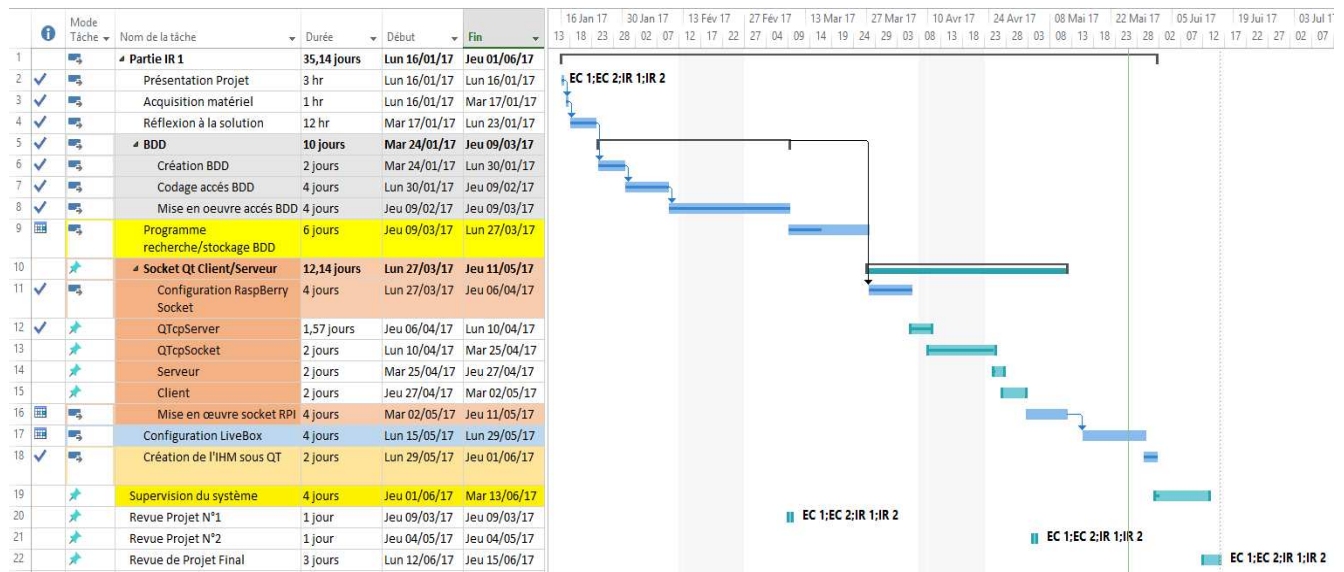
Au sein de ce projet, je suis l'IR 2, celui s'occupant de :

- . *.1. Faire communiquer la RaspBerry Pi et le PC via les sockets Qt,
- . *.2. Créer une base de données,
- . *.3. Développer un programme permettant de stocker et de rechercher des informations dans cette même base de données,
- . *.4. Développer un programme permettant de superviser et contrôler le banc de test,
- . *.5. Créer une interface homme-machine via le logiciel Qt Creator.

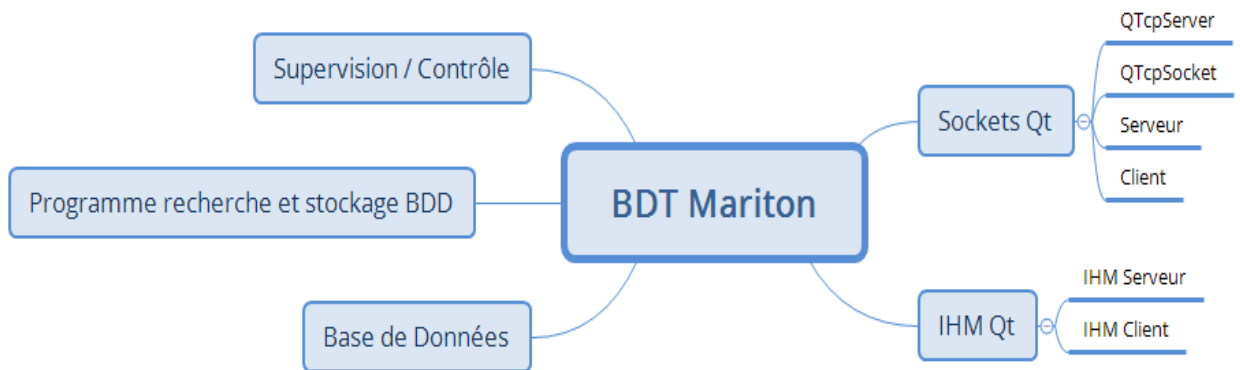
Étudiant 2	Liste des fonctions assurées par l'étudiant	Installation : Qt, MySql. Mise en œuvre : MySql. Configuration : MySql. Réalisation : Communication, impression, supervision et stockage BDD. Intégration côté RPI. Documentation : Installation, mise en service, dossier de développement.
IR 2	✓ RPI utilisée sur le banc de test. ✓ PC sous Linux pour la supervision. • Communication réseau avec les sockets Qt entre la RPI et le PC. • Création de la BDD. • Stockage et recherche des informations dans la BDD. • IHM Qt. • Supervision/contrôle.	



2/ PLANIFICATION



3/ CARTE MENTALE PERSONNELLE



4/ BASE DE DONNEES

4.1/ Modélisation de la base de données :



```
CREATE TABLE IF NOT EXISTS `mydb`.`Test` (  
  `idTest` INT NOT NULL,  
  `NbCycles` INT NULL,  
  `TpsAttHaut` INT NULL,  
  `TpsAttBas` INT NULL,  
  PRIMARY KEY (`idTest`))
```



4.2/ Explications du contenu des tables :

Ma base de données se compose d'une seule table appelée Test, elle contient les différents paramètres que le système est censé récupérer afin de réaliser ses tests.

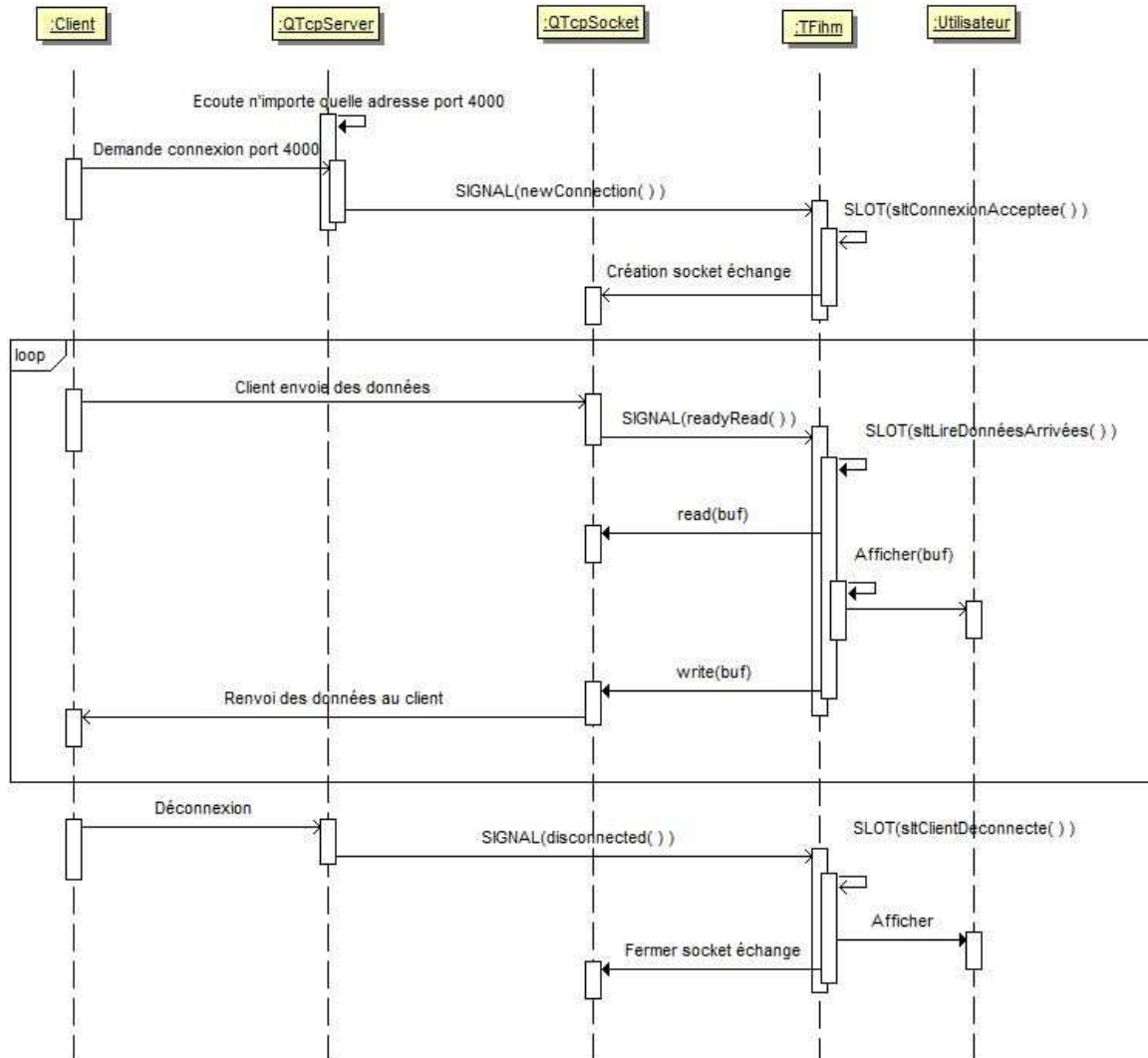
Elle contient :

- .*. NbCycles : Fixe le nombre de cycles que le système devra réaliser avant de s'arrêter.
- .*. TpsAttHaut : Fixe le temps (en seconde) d'attente lorsque le moteur à fait monter le store.
- .*. TpsAttBas : Fixe le temps (en seconde) d'attente lorsque le moteur à fait descendre le store.

5/ COMMUNICATION SOCKETS CLIENT/SERVEUR

5.1/ Diagrammes de séquence

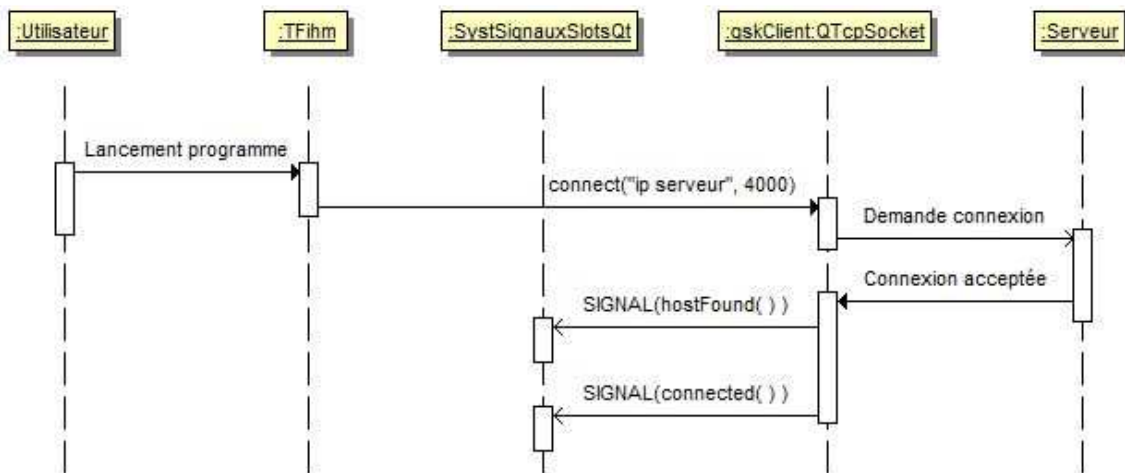
A. Voici le diagramme représentant le Serveur TCP.



Le serveur attend une connexion sur n'importe quelle adresse au port 4000, s'il en reçoit une, cela déclenche un signal de connexion et un slot pour prévenir l'utilisateur qu'il est connecté. Ensuite, une socket d'échange est créée. Via cette socket, le client envoie des données déclenchant un signal de lecture. La lecture s'effectue et l'IHM du serveur voit les données arrivées et les renvoie au client pour « confirmation ».

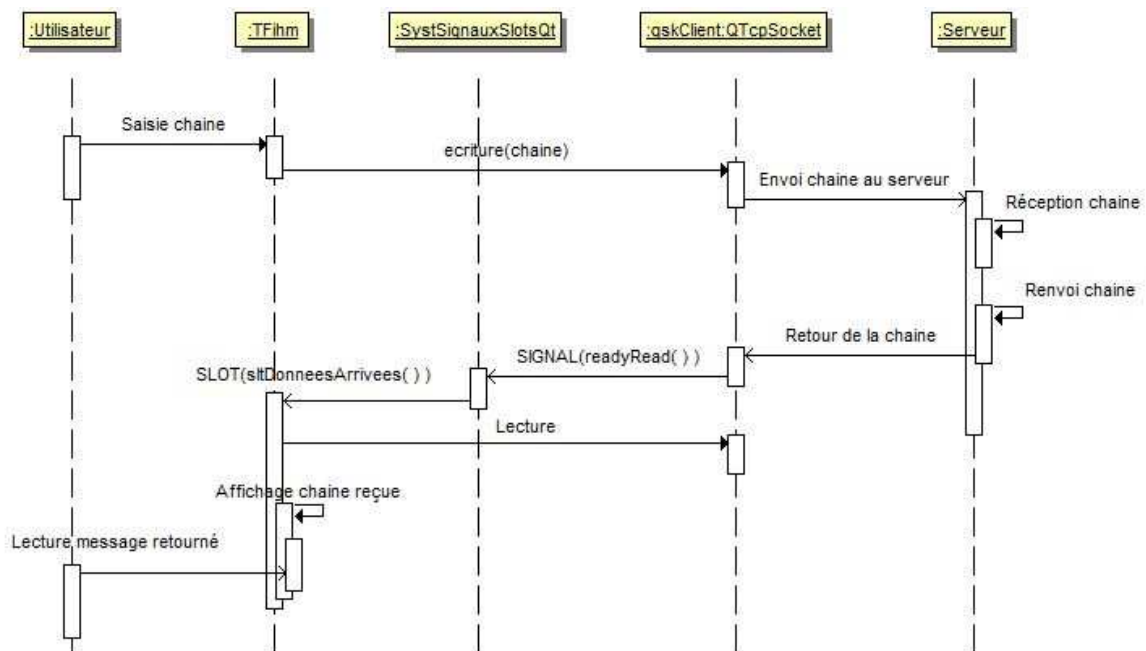
Lors de la déconnexion, un signal est émis afin que la socket d'échange soit fermée.

B. Voici le diagramme d'initialisation du Client.



Le client lance le programme et se connecte au port 4000 demandant la connexion au serveur. Une fois cette connexion établie, deux signaux sont alors émis : le premier signifiant qu'un hôte (le client) a été trouvé, et l'autre signifiant que la connexion est établie correctement.

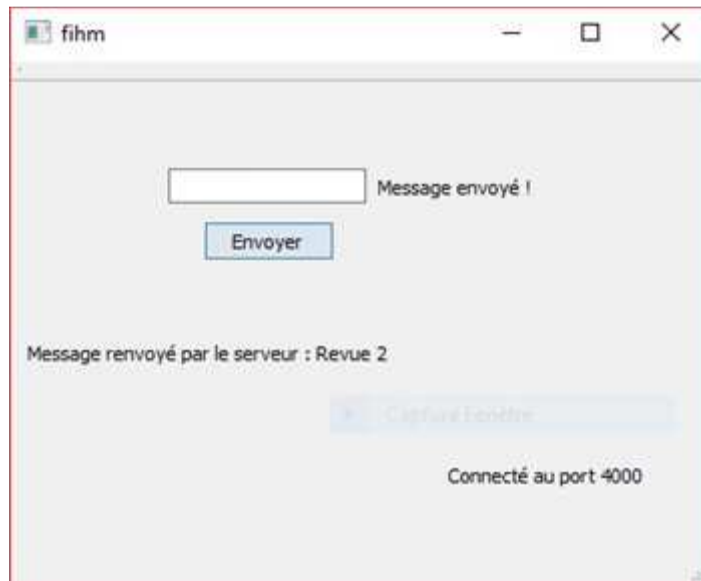
C. Voici le diagramme de communication entre le client et le serveur.



Le client écrit son texte dans un LineEdit, cette chaine de caractère sera alors envoyée au serveur. Cette chaine est reçue et aussitôt renvoyée au client via un signal. La chaine est affichée sur l'IHM du serveur ainsi que sur l'IHM du client afin de bien vérifier l'intégralité de la chaine envoyée par le client.

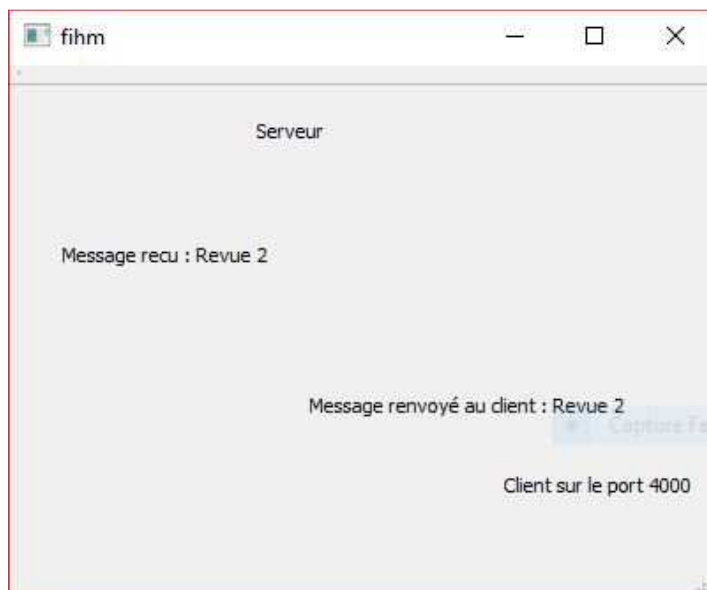
5.2/ IHM Qt du programme Socket Client-Serveur

A. Le Client



Comme expliqué précédemment le client écrit dans un QLineEdit afin d'envoyer son texte au serveur qui lui renvoie son propre texte.

B. Le Serveur



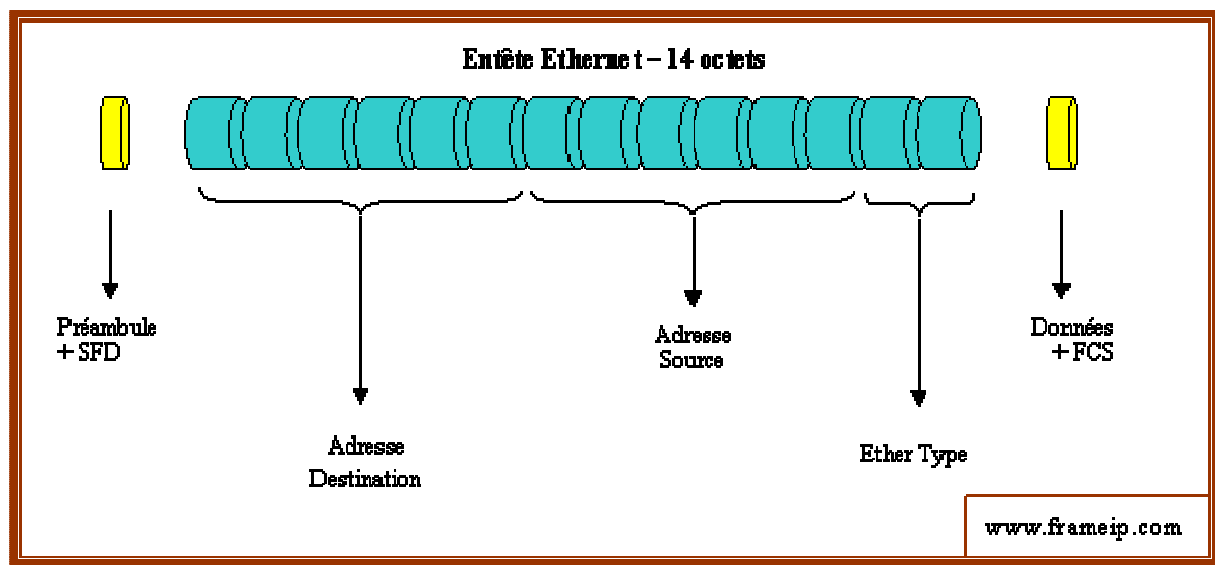
Comme expliqué précédemment le serveur reçoit ce que le client lui envoie, le lit et le renvoie au client.

6/ PROGRAMME RECHERCHE ET STOCKAGE BDD

Ce programme est destiné à permettre à la RPI de recevoir de nouveaux paramètres depuis la base de données comme le nombre de cycles ou les temps d'attentes. A ce jour, ce programme n'est pas développé, mais je sais comment le réaliser, j'ai pris du retard sur les autres parties m'empêchant pour le moment de réaliser celle-ci.

7/ PARTIE PHYSIQUE

A. Etude d'une trame Ethernet



1. Définition des différents champs

1.1 Préambule

Ce champ est codé sur 7 octets et permet de synchroniser l'envoi. Chacun des octets vaut 10101010 et cette série permet à la carte réceptrice de synchroniser son horloge.

1.2 SFD

Ce champ est codé sur 1 octet et indique à la carte réceptrice que le début de la trame va commencer. La valeur du SFD (Starting Frame Delimiter) est 10101011.

1.3 Adresse de destination

Ce champ est codé sur 6 octets et représente l'adresse MAC (Medium Access Control) de l'adaptateur destinataire. Dans le cadre d'un broadcast, l'adresse utilisée est FF-FF-FF-FF-FF-FF.

Cette adresse est ce que l'on appelle l'adresse physique d'une carte Ethernet (Hardware address). En fait cette adresse est divisée en deux parties égales :

- Les trois premiers octets désignent le constructeur. C'est l'organisation OUI (Organizationally Unique Identifier) gérée par l'IEEE, qui référence ces correspondances.

- Les trois derniers octets désignent le numéro d'identifiant de la carte, dont la valeur est laissée à l'initiative du constructeur qui possède le préfixe

1.4 - Adresse source

Ce champ est codé sur 6 octets et représente l'adresse MAC (Medium Access Control) de l'adaptateur émetteur.

Cette adresse est ce que l'on appelle l'adresse physique d'une carte Ethernet (Hardware address). En fait cette adresse est divisée en deux parties égales :

- Les trois premiers octets désignent le constructeur. C'est l'organisation OUI (Organizationally Unique Identifier) gérée par l'IEEE, qui référence ces correspondances.
- Les trois derniers octets désignent le numéro d'identifiant de la carte, dont la valeur est laissée à l'initiative du constructeur qui possède le préfixe

1.5 - Ether Type

Ce champ est codé sur 2 octets et indique le type de protocole inséré dans le champ donnée. Voici un extrait des différentes correspondances :

0x6000	-	DEC
0x0609	-	DEC
0x0600	-	XNS
0x0800	-	IPv4
0x0806	-	ARP
0x8019	-	Domain
0x8035	-	RARP
0x809B	-	AppleTalk
0x8100	-	802.1Q
0x86DD	-	IPv6

Dans le cadre d'une trame VLAN taggée 802.1Q, ce champ doit être à 0x8100 afin de spécifier l'ajout des 4 octets suivants.

1.7 - Données

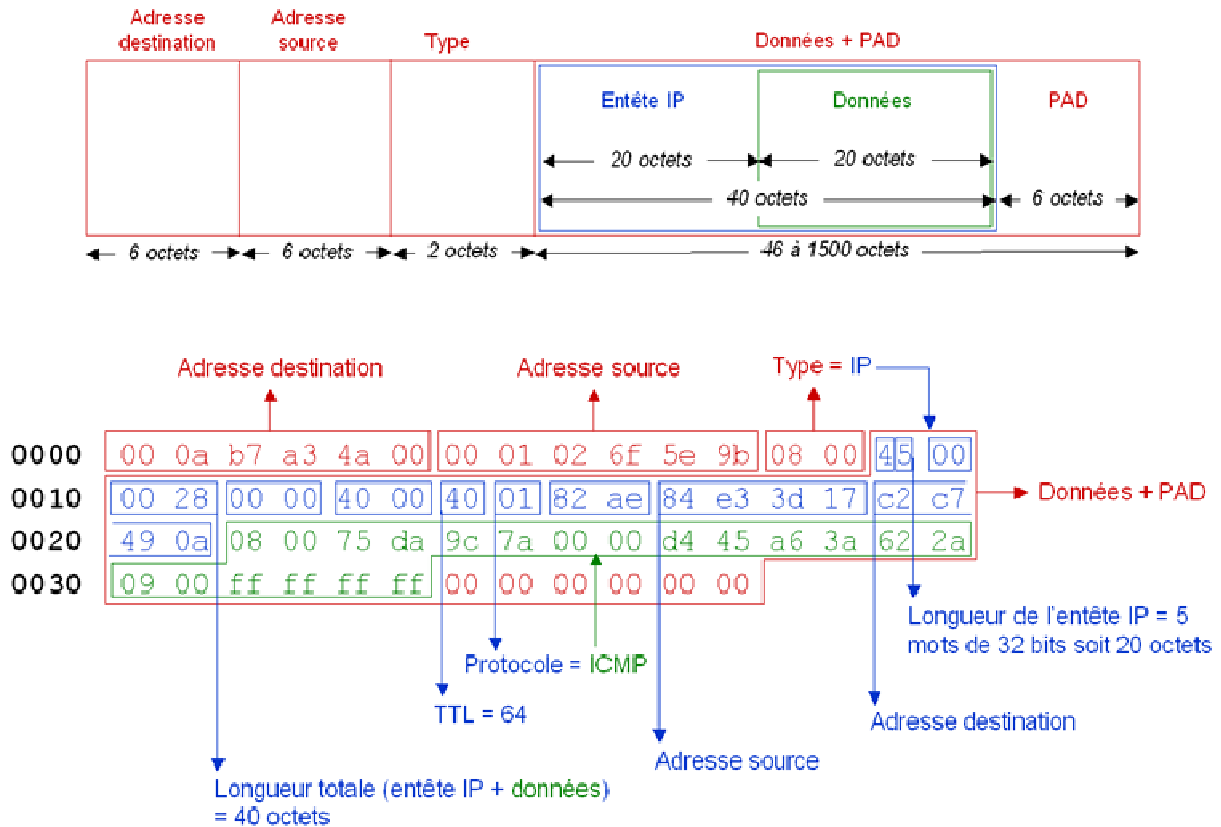
Ce champ est codé entre 46 et 1500 octets et contient les données de la couche 3. Dans le cas de TCP/IP, c'est ici que vient se loger le datagramme IP. L'unité de transfert maximale est le MTU (Maximale Transfer Unit) et sa valeur est classiquement de 1500 octets. Si la taille des données est inférieure à 46 octets, alors elle devra être complétée avec des octets de bourrage (padding) et c'est la couche réseau qui sera chargée de les éliminer.

1.8 - FCS

Ce champ est codé sur 4 octets et représente la séquence de contrôle de trame. Il permet à l'adaptateur qui réceptionnera cette trame de détecter toute erreur pouvant s'être glissée au sein de la trame.

Les erreurs binaires sont principalement créées par les variations d'affaiblissement du signal et l'induction électromagnétique parasite dans les câbles Ethernet ou les cartes d'interface. La valeur de FCS (Frame Check Sequence) est le résultat d'un calcul polynomial appelé CRC (Cyclic Redundancy Code). A la réception de la trame, la couche liaison effectue le même calcul et compare les deux résultats qui doivent être égaux afin de valider la conformité de la trame reçue.

2. Décodage trame Ethernet



- Adresse MAC de la machine ayant répondu : 00:0a:b7:a3:4a:00
- Adresse MAC de la machine ayant initiée l'échange : 00:01:02:6f:5e:9b
- Dans les deux datagrammes IP, le champ *Total Length* vaut 0x28 (40), c'est-à-dire que les deux datagrammes ne représentent que 40 octets chacun. Or le champ données de la trame Ethernet doit contenir au moins 46 octets, il y a donc un bourrage de 6 octets (à 0x00) qui a été ajouté.
- TTL de la trame 1 = 0x40 (64) ; TTL de la trame 2 = 0x3a (58), $64 - 58 = 6$ routeurs
- Le protocole encapsulé est ICMP (champ *Protocol* vaut 0x01) ; il s'agit vraisemblablement d'un message ICMP de demande d'écho, et un message ICMP de réponse d'écho, engendré par la commande ping.
- Adresse IP de la machine ayant initiée l'échange : 84.e3.3d.17 = 132.227.61.23
- Adresse IP de la machine ayant répondu : c2.c7.49.0a = 194.199.73.10

8/ CONCLUSION

Ce projet m'a beaucoup apporté en matière de connaissances et de compétences. Il m'a permis de développer un esprit d'équipe et une meilleure organisation. J'ai eu plusieurs difficultés, une mauvaise gestion du temps et des lacunes en codage qui ont résulté en de nombreux échecs sur mes différents codes. J'ai démarré ce projet par les rappels et les formations mises à notre disposition à propos des bases de données afin de bien la concevoir. Une fois la partie programmation commencée, je me suis vite perdu ne sachant pas trop comment réaliser ce que l'on me demandait au sein de ce projet, fort heureusement nous avons abordé le sujet d'une communication réseau pendant l'année scolaire.

La communication avec les autres membres de l'équipe était plutôt bonne, nos différentes réunions nous ont permis de bien savoir qui en était où et notre professeur référent assistait souvent à nos réunions afin de nous diriger vers la bonne voie et nous fournir les bons documents.