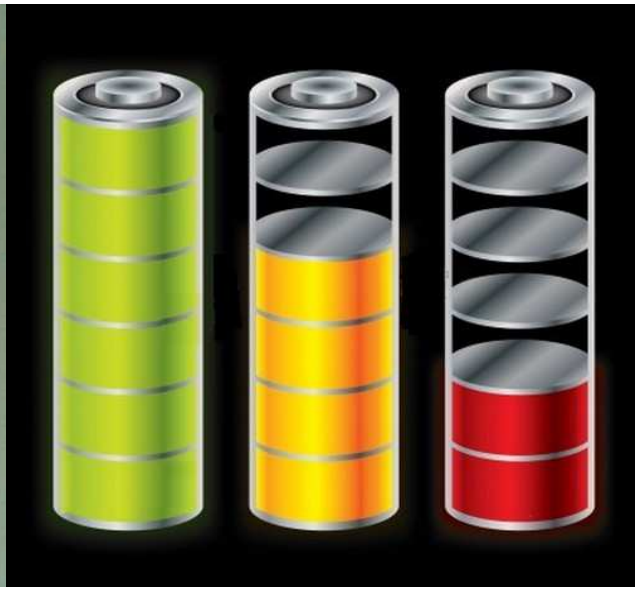
BECB : BOITIER D'ETAT DE CHARGE DE BATTERIE



PROJET TUTORE
2013 / 2014

Etudiants :

Arnaud MEYER
Arthur WILHELM

Tuteur :

Christophe LOMBARD

SOMMAIRE

Table des matières

I) INTRODUCTION

1) Objectifs :

Produire une maquette permettant la surveillance d'une batterie ainsi que la visualisation de son état de charge.
Maîtriser les échanges de données par les réseaux LIN.

2) Description du projet :

Le projet réside dans l'analyse, le développement et la validation d'un système de surveillance de charge de batterie connecté à un calculateur par réseau LIN.

Ce type de dispositif existe sur les véhicules type Peugeot 508 et porte le nom de BECB. Il faut chercher une structure matérielle à base de microcontrôleur permettant de quantifier et mesurer un état de charge de batterie au plomb.

L'information sera transmise sur un réseau de terrain (LIN).

3) Remerciements :

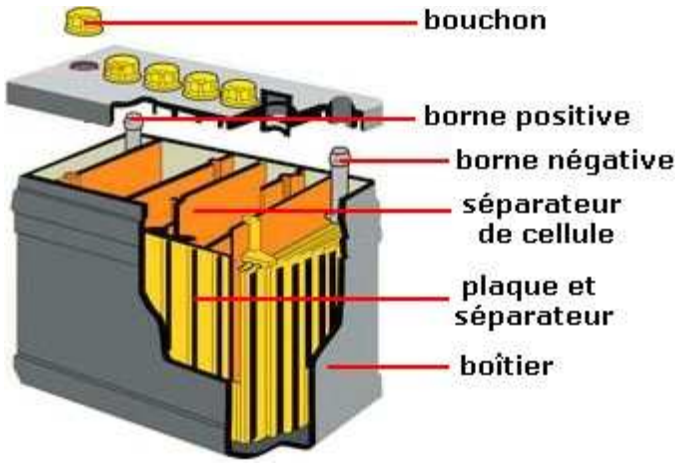
Nous tenons à remercier l'ensemble du personnel enseignant qui nous ont accompagnés et aidés tout au long de notre projet.

Les informations précieuses qu'ils nous ont données, nous ont ainsi permis de poursuivre ce projet dans les meilleures conditions.

II) LA BATTERIE AU PLOMB

1) Historique :

L'accumulateur au plomb a été inventé en 1859 par le français Gaston Planté. Il a été en effet le premier à avoir mis au point la première batterie rechargeable. À l'origine, les accumulateurs étaient situés dans des cuves en verre. Par la suite, on a systématisé l'emploi des cuves en plastique.



2) Caractéristiques techniques :

La tension nominale qui dépend du nombre d'éléments, la tension nominale U est égale au nombre d'éléments multiplié par 2,1 V.
Une batterie bien chargée a une tension supérieure à 2,1 x 6 = 12,6 V
Une batterie déchargée ou en mauvais état a une tension inférieure à 1,8 x 6 = 10,8 V

État de charge	Tension batterie	50 %	12,06V
100 %	12,7V	40 %	11,9V
90 %	12,5V	30 %	11,75V
80 %	12,42V	20 %	11,58V
70 %	12,32V	10 %	11,31V
60 %	12,2V	0 %	10,5V

La capacité de stockage, notée Q, représente la quantité d'énergie disponible. Elle s'exprime en ampère-heure.

Énergie spécifique : 20-40 Wh/Kg
Densité d'énergie volumique : 40-100 Wh/L
Durée de vie : 4 à 5 ans
Nombre de charges : 500 à 1200 cycles

3) Performances :

La batterie au plomb est celle qui a la plus mauvaise énergie massique 35 Wh/kg, après la batterie Nickel-Fer. Cependant, elle est encore très utilisée car elle est la moins coûteuse et peut fournir un courant de grande intensité, utile pour le démarrage des véhicules automobiles.

4) Utilisation :

Cette batterie sert à alimenter les composants électriques des véhicules à moteur à explosion, particulièrement le démarreur électrique, alimentée par un alternateur.

Les voitures électriques ne se sont toujours pas imposées du fait du mauvais rapport masse/énergie des batteries, bien que le rendement d'un moteur électrique soit exceptionnel.
(85% pour en moteur électrique contre seulement 30% pour un moteur thermique).

5) Causes de dégradation :

Les principales causes de dégradation des batteries sont :
- la sulfatation
- la décharge complète
- le cyclage
- l'oxydation des électrodes
- l'oxydation des bornes

6) Influence de la température :

La capacité réelle d'une batterie diminue avec la température : c'est ce qui explique que les démarrages des véhicules sont plus difficiles lorsque la température est très basse.

La durée de vie d'une batterie est indiquée par le fabriquant pour une température ambiante de 20°C.
Cette durée de vie est réduite de moitié pour une élévation de température de 10°C.

Température	0°C	10°C	15°C	20°C	25°C	30°C
Capacité	80%	92%	95%	100%	103%	105%

Le coefficient de température de la batterie est de 1,2mV/°C.

Sources :
http://garnero.michel.free.fr/Docs2/batterie_plomb

III) BECB

Boîtier d'Etat de Charge Batterie

1) Présentation :

Le boîtier d'état de charge batterie est un composant essentiel du système électronique de gestion énergétique.

Implanté dans la cavité de la borne (-) de la batterie, il enregistre dynamiquement et avec la plus grande précision les valeurs relatives à la batterie, comme le courant, la tension et la température.

À partir des valeurs mesurées, le capteur détermine l'état actuel et prévisible de la batterie puis transmet cette information au calculateur habitacle via le réseau multiplexé LIN pour affiner l'activation du mode économie d'énergie.

Les informations état batterie sont disponibles et diffusées sur demande du calculateur habitacle dans les phases suivantes :

- réveil du calculateur habitacle;
- réveil partiel ou total du réseau CAN moteur;
- réveil des réseaux CAN confort et CAN carrosserie.

Mode délestage :

Véhicule roulant, lorsque le niveau d'énergie restant dans la batterie est faible, le délestage neutralise temporairement certaines fonctions, telles que l'air conditionné, le dégivrage de la lunette arrière...
Les fonctions neutralisées sont automatiquement réactivés dès que les conditions le permettent.

Le BECB mesure la température de la batterie et
estime son état de charge. C'est en mesurant sa tension au repos et l'intensité
qu'elle consomme ou distribue, qu'il réalise cette opération. Il est un élément clé pour
la fonction de gestion de l'alternateur piloté.

2) Pourquoi et comment mesurer l'état de charge d'une batterie ?

Sans dispositif pour indiquer précisément l'état de charge, c'est comme si vous conduisiez une voiture sans jauge d'essence …

Une mesure de tension seule pour estimer l'état de charge d'une batterie peut être trompeuse.
En effet, une batterie sulfatée peut présenter une tension élevée alors que sa capacité effective peut être très limitée.
De plus, l'estimation de l'état de charge sera plus précise si la batterie est laissée au repos quelque temps (24 h après une charge, une heure après une décharge) avant la mesure de la tension.

L'utilisation d'un pèse-acide permet d'obtenir des mesures très précises de l'état de charge. La relation entre température de la batterie, densité de l'acide et état de charge s'obtient au moyen d'abaques.
Cependant la mesure avec un pèse acide est ponctuelle et non continue.

IV) Calcul du SOC (état de charge) et ses enjeux

1) Le calcul du SOC :

Le calcul du SOC se décompose en deux phases :
_ La phase dynamique : la batterie est sollicitée en courant (hors équilibre électrochimique)
_ La phase repos (mode veille) : la batterie est au repos (courant nul) elle est alors en phase de relaxation pour converger vers un état d'équilibre électrochimique. Pour chacune de ces phases, la méthode d'estimation du SOC est différente. C'est en phase de repos et après un temps suffisamment long (3 à 4h) qu'on atteint la meilleure estimation du SOC de la batterie : c'est l'étape de recalage qui permet d'effacer les dérives précédentes liées principalement aux imprécisions de mesures et aux erreurs de rendement de charge lors du comptage Ah.

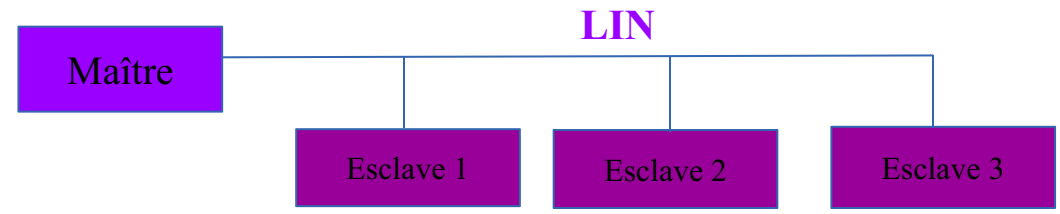
2) Algorithme de calcul :

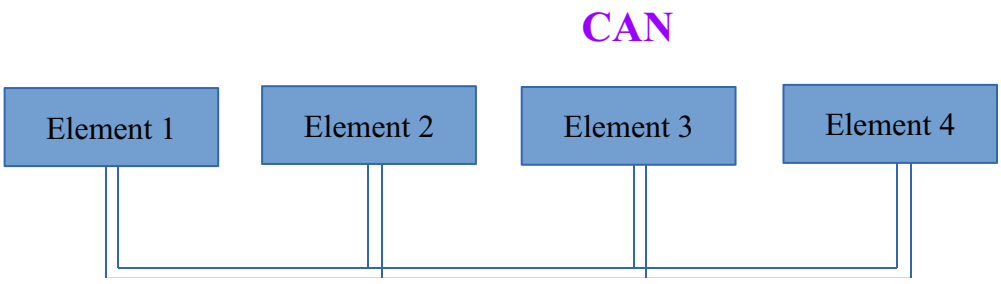
Pour estimer l'état de charge d'une batterie, nous mesurons la tension à vide, ce qui veut dire que la batterie ne fournit qu'un très faible courant (alimentation carte). Grâce à cette tension, nous estimons son état de charge. La température est prise en compte pour réajuster cette valeur dans le plan réel. La mesure du courant est utile pour connaître le courant débité par la batterie est ainsi connaître la capacité restante de celle-ci en ampère-heure.

V) LIN – CAN

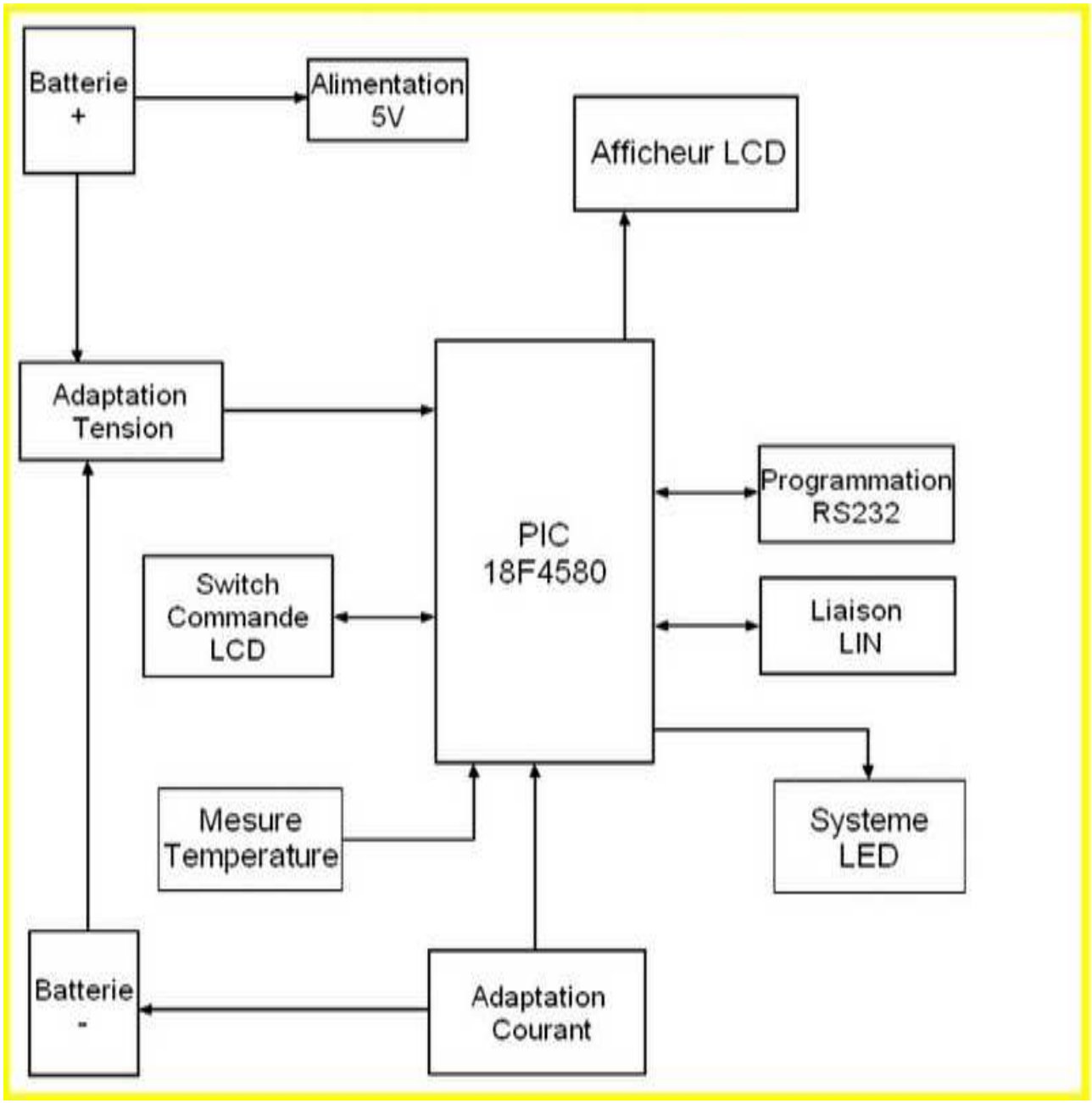
LIN	CAN
1 fil	2 fils
Le maître dirige et initialise les envoies des esclaves	Celui qui a l'adresse la plus basse envoie en premier si il y a conflit
Pas de bit de Stuffing	Bit de Stuffing
Transmission max : 19,2 Kbaud	Transmission max : 500 Kbaud en automobile
Horloge : Celui du maître	Horloge : Automatique
Nombre de liaison max : 16 esclaves + 1 maître	Nombres de liaison max : 536870912
Taille des données : 0 – 64 Octets	Taille des données : 2,4,8 + Checksum

La liaison CAN est bien plus performante que la liaison LIN. Mais une fois que le coût est pris en compte, la liaison LIN est plus avantageuse pour un réseau secondaire ou moindre.





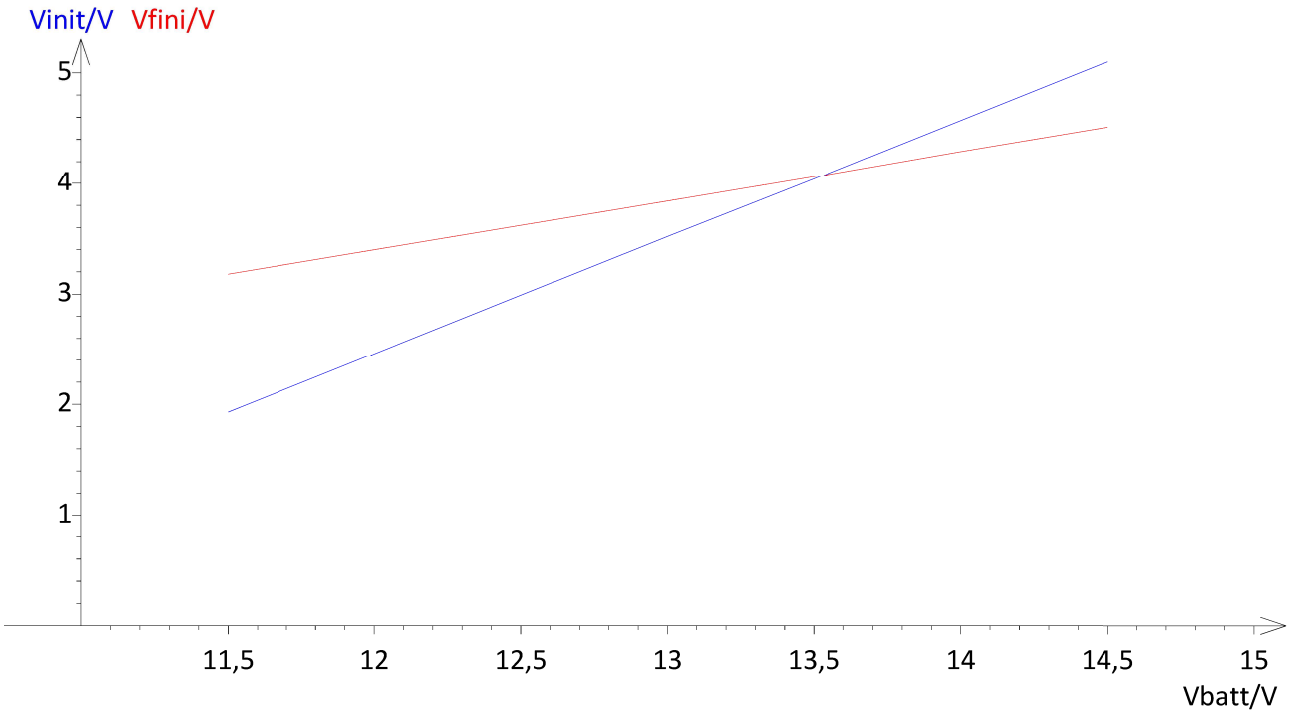
VI) SYNOPTIQUE



VII) FONCTIONS

1) Adaptation Tension :

Ne pouvant mesurer la tension de la batterie directement sur le micro-contrôleur, nous avons utilisé un AOP permettant de réaliser une variation de tension bien plus importante qu'un simple pont diviseur.
Cette tension varie entre 1,9V pour une Batterie vide et 5V pour une charge de cette batterie.



Vinit = ((Vbatt / 9,66) – 1) x 10,2

Vfini = ((Vbatt / 4,3) – 1) x 1,9

Les différentes valeurs des résistances choisies pour l'adaptation de la tension étaient bonnes mais l'AOP saturait quand la tension était trop basse, nous avons donc dû changer quelques résistances.
Sur plaque Labdec, l'adaptation de la tension marchait parfaitement.

Nous avons gardé la courbe rouge.

2) Adaptation Courant :

Pour mesurer le courant délivré par la batterie, nous utilisons un capteur LEM.

Electrical data

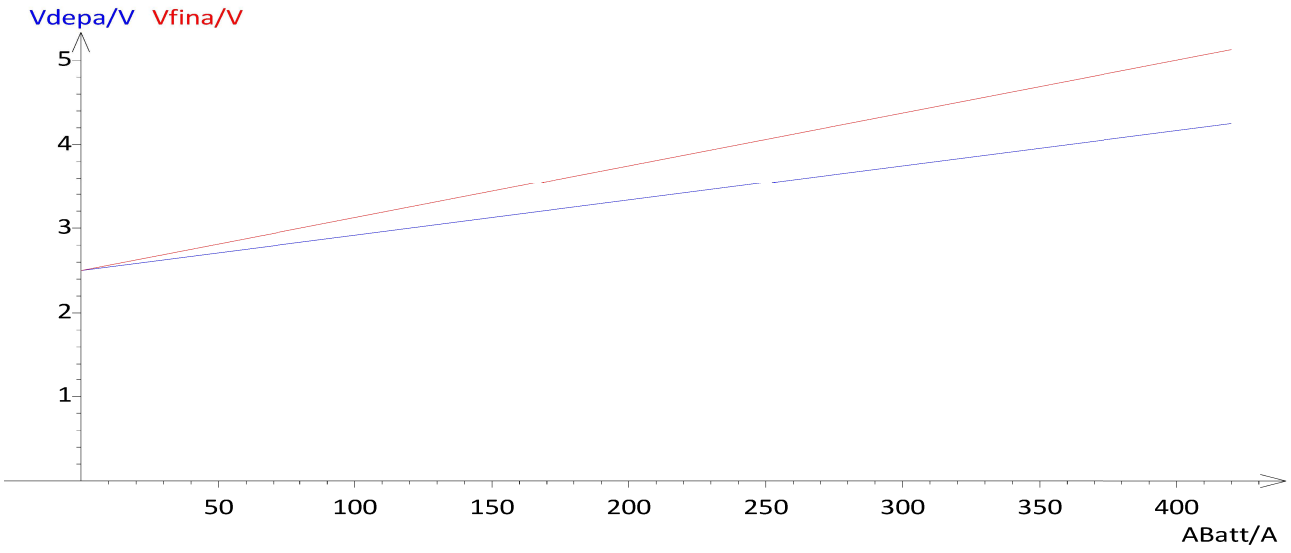
Primary nominal current rms I_{PN} (A)	Primary current, measuring range I_{PM} (A)	Type	RoHS since date code
50	± 150	**HAIS 50-P, HAIS 50-TP**[1]	45231, 46272
100	± 300	**HAIS 100-P, HAIS 100-TP**[1]	45231, 46012
150	± 450	**HAIS 150-P**	46172
200	± 600	**HAIS 200-P**	45231
400	± 600	**HAIS 400-P**	planned

V_{OUT}	Output voltage (Analog) @ I_P	$V_{REF} \pm (0.625 \cdot I_P/I_{PN})$	V
	$I_P = 0$	$V_{REF} \pm 0.025$	V
V_{REF}	Reference voltage [2] - Output voltage	2.5 ± 0.025	V
	V_{REF} Output impedance	typ. 200	Ω
	V_{REF} Load impedance	≥ 200	kΩ
R_L	Load resistance	≥ 2	kΩ
R_{OUT}	Output internal resistance	< 10	Ω
C_L	Capacitive loading	< 1	µF
V_C	Supply voltage (± 5 %)	5	V
I_C	Current consumption @ V_C = 5 V	22	mA

Nous pouvons voir dans la documentation technique du composant que la tension délivrée en sortie de ce capteur est linéaire. Ce capteur ne peut délivrer une tension supérieure à 5V.

Comme pour mesurer la tension, nous avons utilisé un AOP permettant d'augmenter la plage de mesure et ainsi d'accroître la précision.

Plus la précision est grande pour la mesure du courant et plus nous pourrons être précis sur la capacité restante de la batterie.



Vdepa = 2,5 + (0,625 x I / 150)
Vfina = 2,5 + (0,625 x I / 50)

Si le courant dépasse 400A alors la sortie du capteur LEM sature à 5V ce qui correspond au démarrage du véhicule, qui se produit très brièvement.

3) Mesure de la Température :

Température	0°C	10°C	15°C	20°C	25°C	30°C
Capacité	80%	92%	95%	100%	103%	105%

Mesurer la température ambiante proche de la batterie permet d'ajuster le SOC de départ et ainsi savoir si la batterie est vraiment déchargée ou simplement soumise à une contrainte d'ordre chimique.

4) Affichage des variables via des Dels :

Pour une visualisation approximative de l'état de charge de la batterie, nous avons mis en place un système de leds permettant de visualiser sa capacité restante.

5) Afficheur LCD :

Pour visualiser avec précision les différentes valeurs nécessaires à mesurer pour estimer l'état de charge de la batterie, nous avons mis un afficheur LCD 16*2.

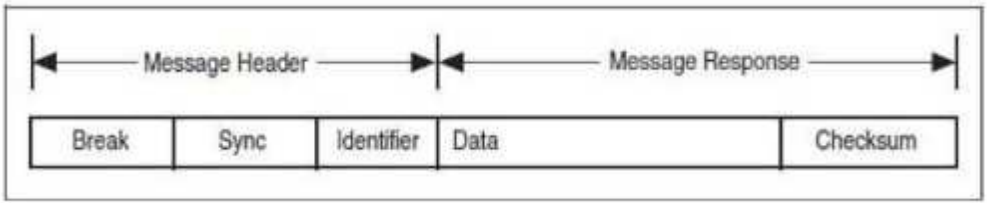
Grâce au switch, nous pouvons sélectionner 5 modes :

_ Affichage de la tension
_ Affichage du courant instantané
_ Affichage de la température
_ Affichage du SOC
_ Mode Veille

VIII) LIAISON LIN

Nous pouvons faire circuler des données sur un réseau LIN via le connecteur RS232 à l'aide du boîtier EXXOTEST.

Protocole LIN :



Constitution d'une trame :

• Champ break : 13 bits dominants minimum (état 0), permet d'initier la communication.

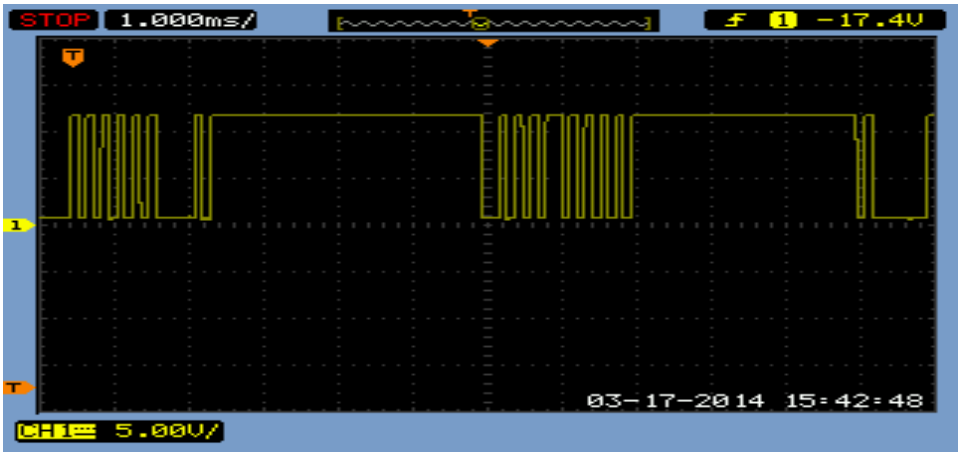
• Champ de synchronisation : un signal d'horloge est émit (un octet 0x55) pour synchroniser le maître et l'esclave.

• Champ identifiant : un octet, comportant l'adresse, la longueur du message ainsi que deux bits de contrôle.

• Champ de données : Les différentes données sont envoyées dans l'ordre de l'octet 1 à 8.

• Champ de contrôle : Il s'agit d'un checksum permettant de contrôler l'ensemble des données transmises.

Nous avons pu visualiser une trame LIN, où on peut voir le champ break, le champ synchro, les données envoyées (0xAA et 0x55).



IX) PROBLEMES RENCONTRES

Nous avons rencontré plusieurs problèmes.

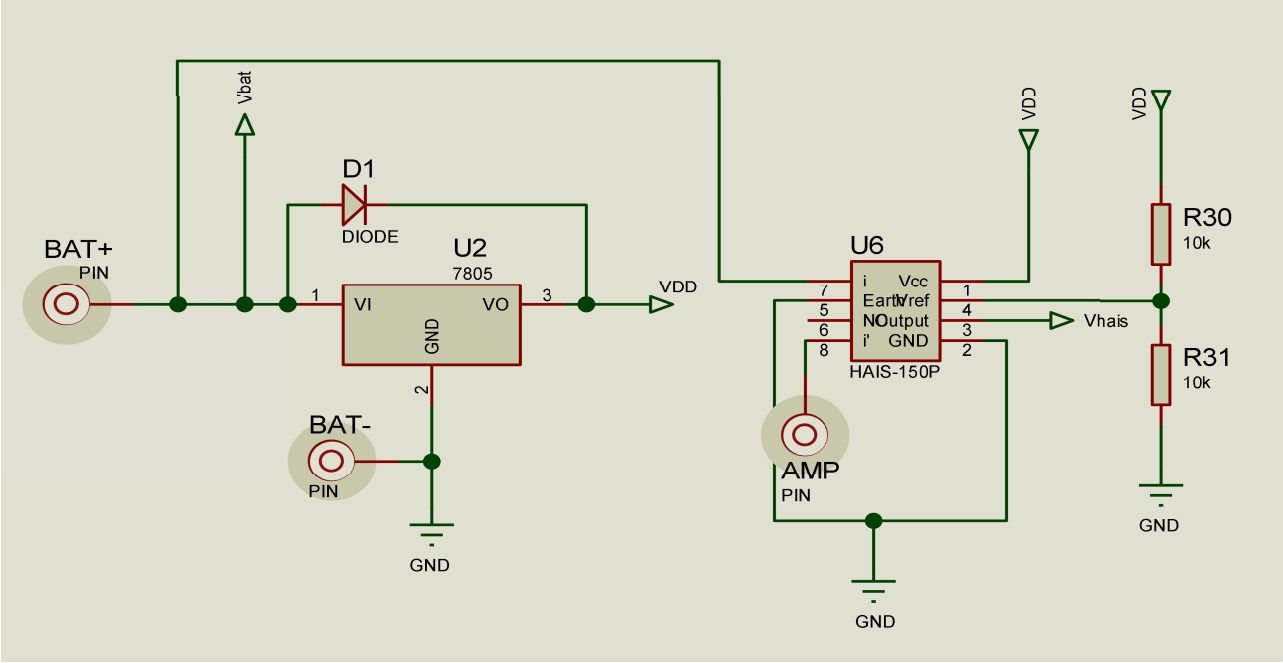
Celui qui nous a le plus ralenti est l'afficheur LCD, grâce à l'oscilloscope, nous avons pu voir qu'il y avait des conflits avec le driver LCD sous programmation C.

Nous avons pu prendre des composants de la carte des élèves en charge de ce projet l'année dernière. Nous avons pu récupérer le capteur LEM, malheureusement après installation de celui-ci sur notre carte, la tension en sortie ne correspond pas à la documentation technique.
Nous avons donc dû en commander un nouveau, la tension de sortie de ce capteur correspond à la documentation technique.

XI) ANNEXES

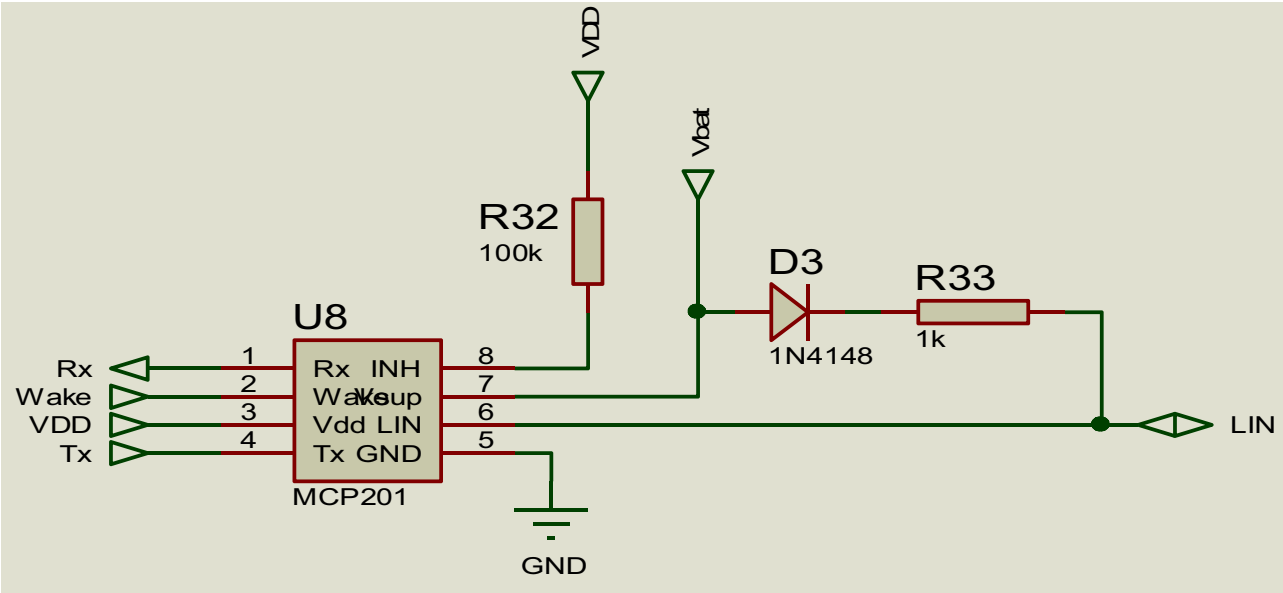
1) Sous ISIS :

Alimentation de carte avec le capteur de courant :



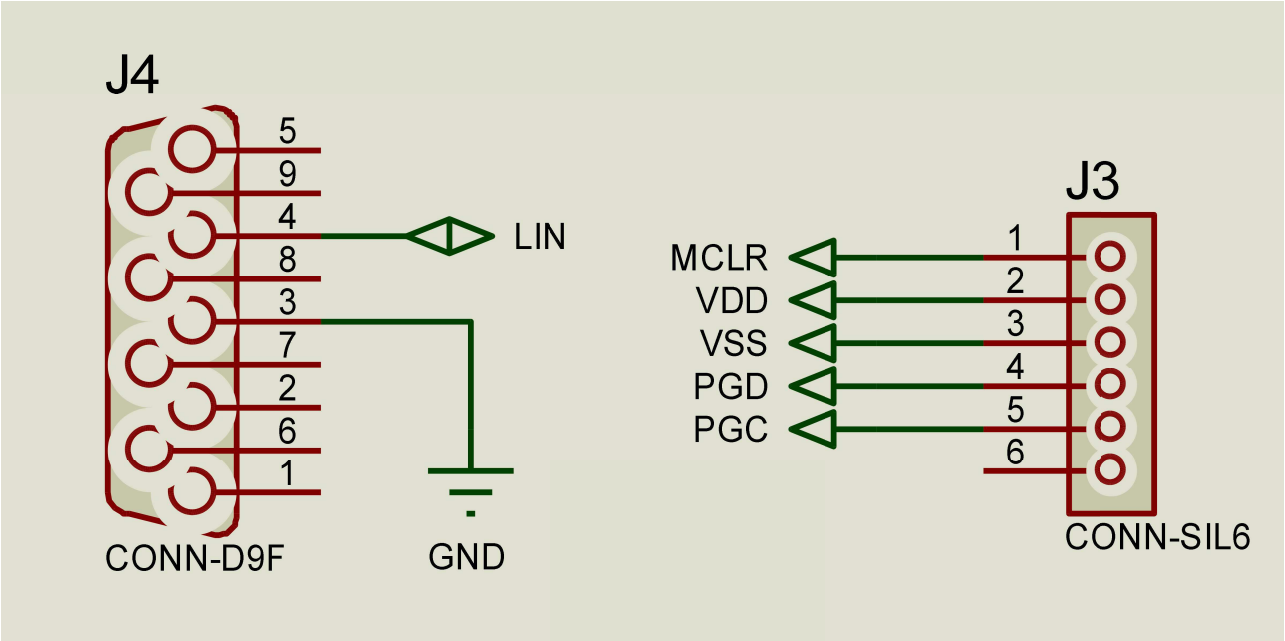
La carte est alimentée par la batterie, les 5V pour alimenter la carte sont fournis par le régulateur de tension 7805. Le bornier AMP est relié au rhéostat ou sur la partie électrique du véhicule.

Liaison LIN :



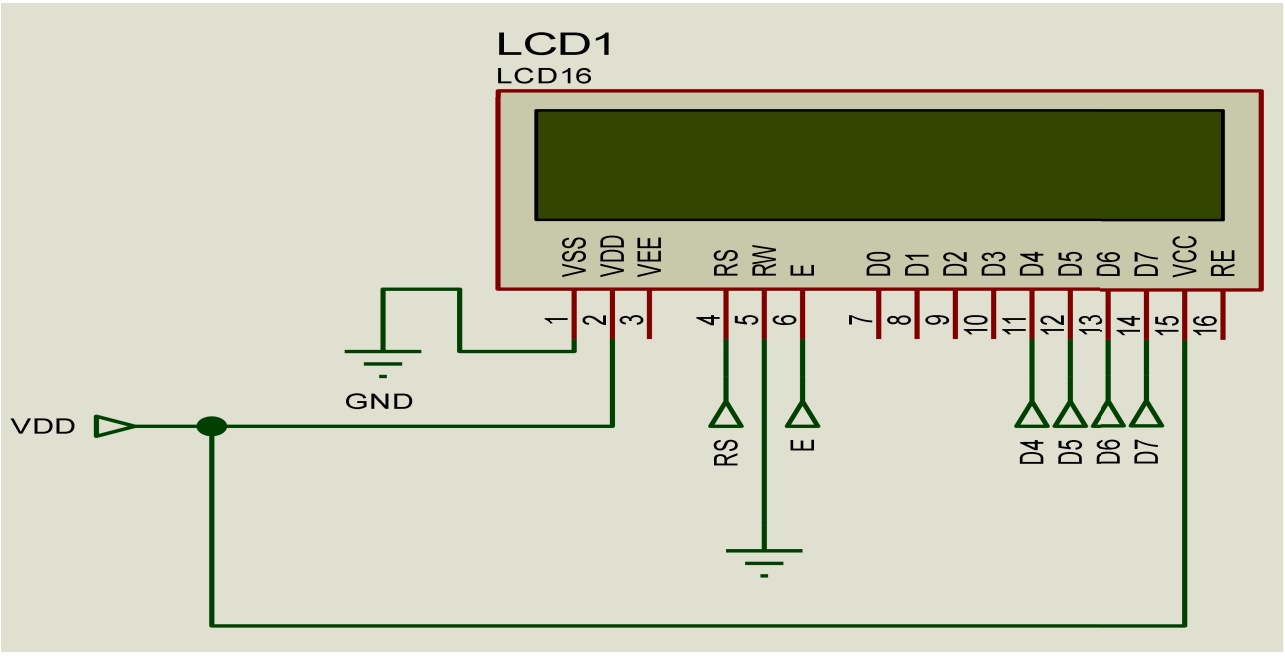
La liaison LIN ne comporte qu'un fil (voir page 8)
Nous utilisons le composant MCP201 qui est nécessaire pour l'envoie des données sur un réseau LIN.

Programmation et Port Série :



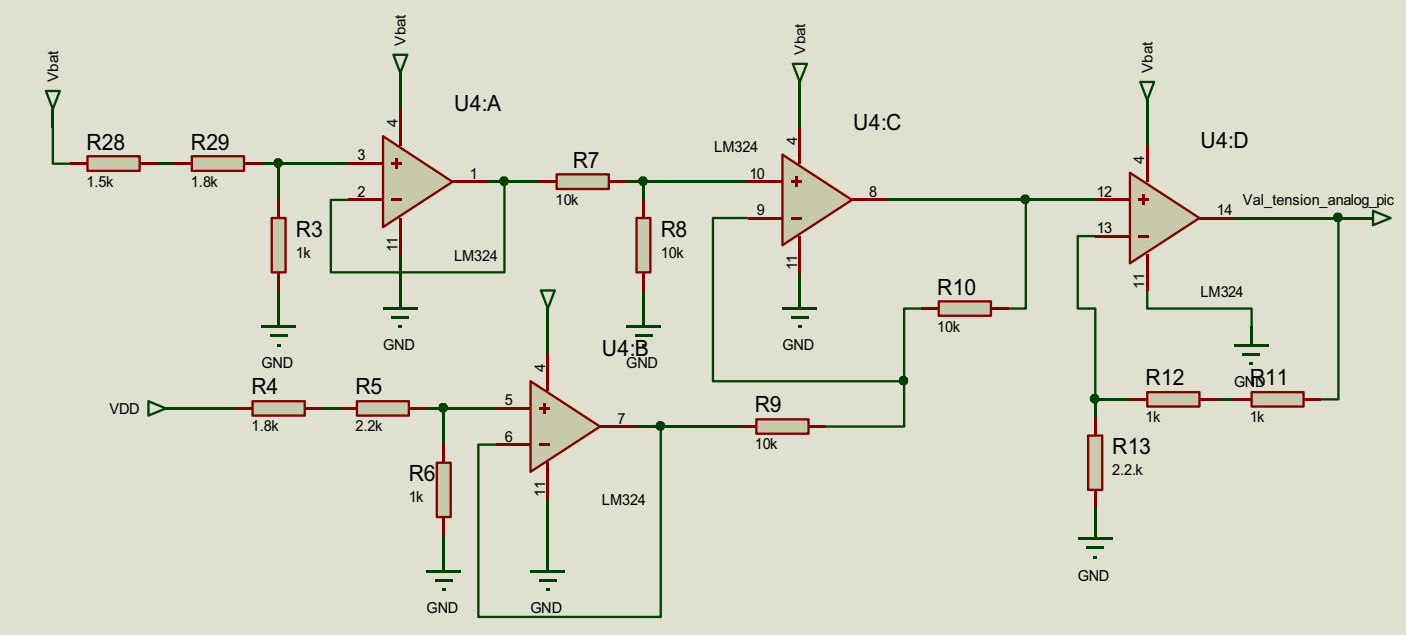
Le DB9 sert à faire circuler les données via une liaison LIN vers le boîtier USB-MUX-DIAG-II sous Exxotest.
J3 est nécessaire pour le branchement du Programmeur PICkit 3.

Afficheur LCD :



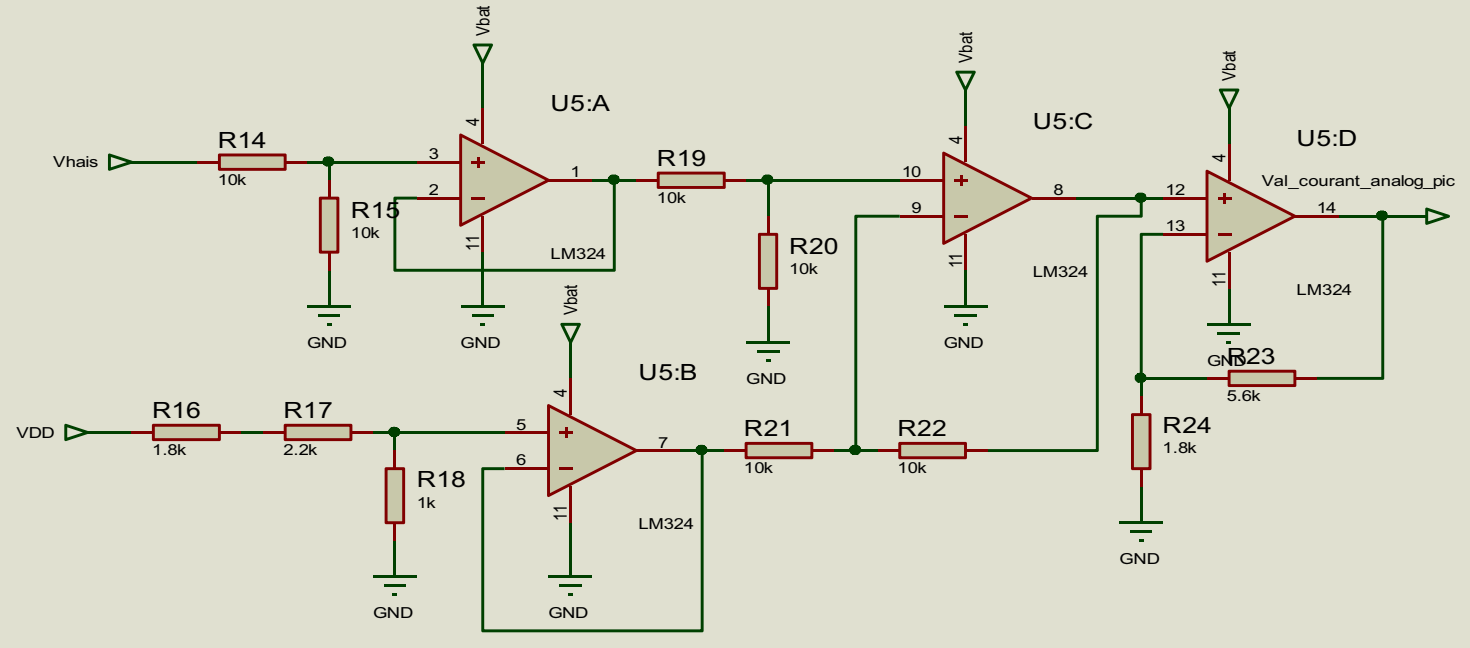
L’afficheur LCD nous permet d’afficher des données de façon précise et compréhensive.

Adaptation Tension Batterie :



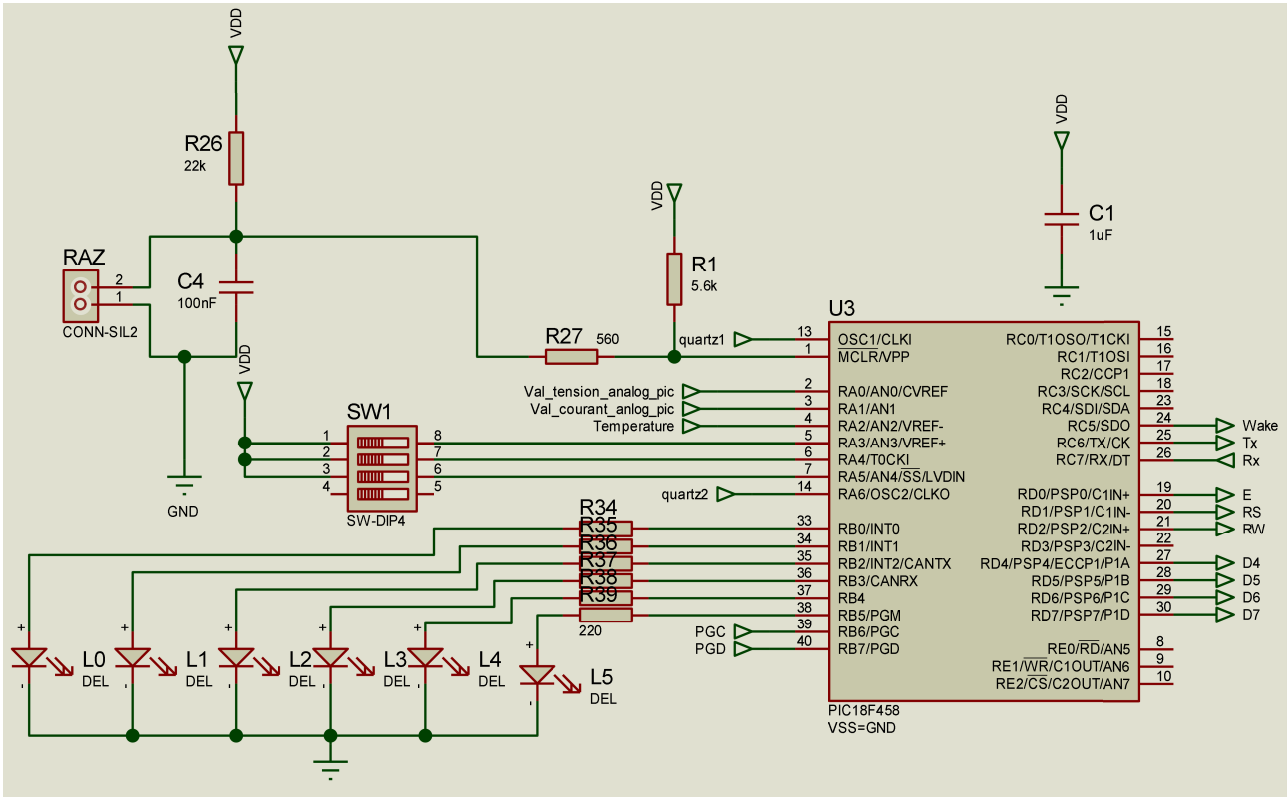
Pour adapter la tension de la batterie, nous avons utilisé un AOP avec plusieurs résistances placées de façon à réaliser des fonctions bien précises.
U4:A est un suiveur, les résistances permettent de réaliser un division par 4,3 de la tension fournie par la batterie.
U4:B est lui aussi un suiveur, ces résistances permettent d’avoir en sortie une tension égale à 1V.
U4:C est un soustracteur, ces résistances permettent de réaliser une soustraction sans pertes.
La sortie de U4:A moins la sortie de U4:C.
U4:D est un multiplieur, ces résistances permettent de réaliser une multiplication par 1,9.

Adaptation Courant Batterie :



Pour adapter la tension en sortie du capteur de courant, nous avons utilisé un AOP avec plusieurs résistances placées de façon à réaliser des fonctions bien précises.
U5:A est un suiveur, les résistances permettent de réaliser un division par 2 de la tension fournie par le capteur LEM.
U5:B est lui aussi un suiveur, ces résistances permettent d'avoir en sortie une tension égale à 1V.
U5:C est un soustracteur, ces résistances permettent de réaliser une soustraction sans pertes.
La sortie de U5:A moins la sortie de U5:C.
U5:D est un multiplieur, ces résistances permettent de réaliser une multiplication par 4,11.

PIC, Switch et Dels :



Nous utilisons un PIC18F4580 alimenté en 5V, il dispose de 4 Ports dont 1 analogique (Port A)
Les broches A0, A1 et A2 sont utilisées comme entrées analogiques.
A0 permet de recevoir la tension de l'adaptation de tension fournie par la batterie.
A1 permet de recevoir la tension de l'adaptation du courant délivré par la batterie.
A2 permet de recevoir l'image de la température.

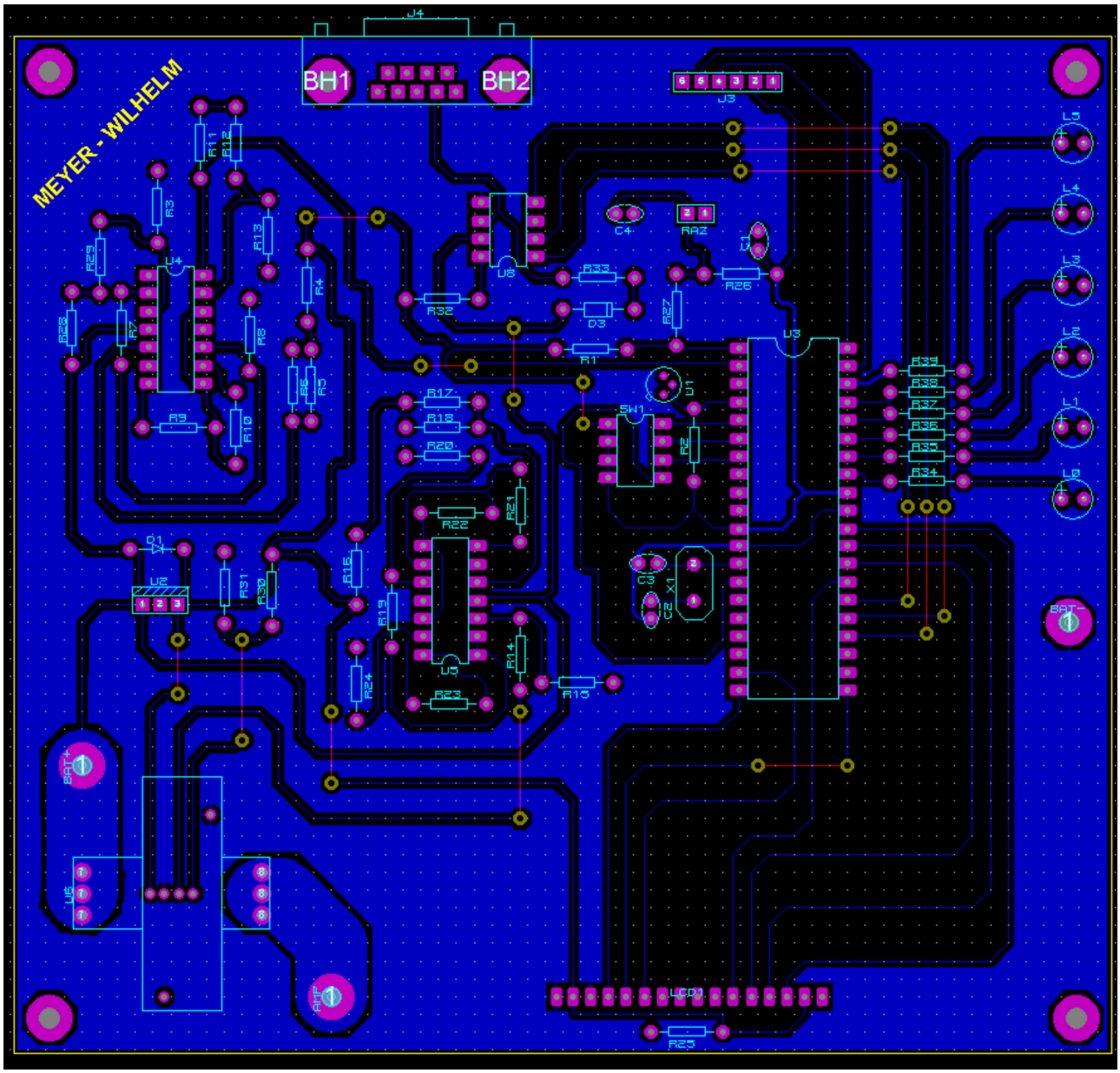
Les Switchs permettent de sélectionner un mode dans la partie programmation.

Nous utilisons le Port B pour la visualisation de l'état de charge grâce au dels.

Le Port C n'est pas utilisé mis à part pour la transmission LIN (RC6 et RC7).

Le port D est utilisé pour contrôler l'afficheur.

2) Sous ARES :



Le typon n'a pas été modifié suite au erreurs trouvés sur la carte lors de nos essais.

3) Liste des Composants :

38 Resistors

Quantity:	References	Value
2	R1, R23	5.6k
5	R2, R4, R16, R24, R29	1.8k
6	R3, R6, R11, R12, R18, R33	1k
2	R5, R17	2.2k
12	R7-R10, R14, R15, R19-R22, R30, R31	10k
1	R13	2.2.k
1	R26	22k
1	R27	560
1	R28	1.5k
1	R32	100k
6	R34-R39	220

4 Capacitors

Quantity:	References	Value
2	C1, C4	10nF
2	C2, C3	15pF

7 Integrated Circuits

Quantity:	References	Value
1	U1	LM335
1	U2	7805
1	U3	PIC18F458
2	U4, U5	LM324
1	U6	HAIS-150P
1	U8	MCP201

2 Diodes

Quantity:	References	Value
1	D1	DIODE
1	D3	1N4148

15 Miscellaneous

Quantity:	References	Value
3	AMP, BAT+, BAT-	PIN
1	J3	CONN-SIL6
1	J4	CONN-D9F
6	L0-L5	DEL
1	LCD1	LCD16
1	RAZ	CONN-SIL2
1	SW1	SW-DIP4
1	X1	CRYSTAL

XII) PROGRAMMATION

Vu que nous avons rencontré un problème d'afficheur LCD, nous avons donc fait un programme uniquement grâce à notre système de dels.

Test Transmission LIN :

```
#include <C:\Logiciels\PICC\Devices\18F4580.h>
#fuses HS,NOWDT
#use delay (clock=20000000)                                  // quartz de 20MHz
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)               // transmission 9600 baud, déclaration cablage
#include <C:\Logiciels\PICC\Drivers\LINBUS.h>

#byte TXSTA=0xBA                          // registre du microcontrôleur gérant la transmission via USART
#bit SENDB=TXSTA.3

int8 command[2]={0xAA,0x55};              // exemple de tableau de données

void main()
{
int ident=MASTER_ADDRESS;          // MASTER_ADDRESS est une variable presente dans le driver LIN
output_high(pin_c5);               // mise à un de la broche WAKE permettant le reveil du MCP201
while(true)
{
        SENDB=1;                   // obtention du champ BREAK,
        putc(0x00);
        linbus_generate_synchfield();      // Emission du caractère d'émission ( 0x55 )
        lin_bus_transmit_data(ident,2,&command[0]);      // envoi de l'identifiant, le nombre d'octet à envoyer
                                                         // et l'envoi du tableau contenant les données
        delay_ms(50);                                    // tempo de 50ms pour une bonne visualisation
}
}
```

Programme avec afficheur LCD :

Avec la carte de tp ainsi qu'une plaque Labdec avec les AOP, résistances et les alimentations simulant la batterie et le capteur de courant. L'affichage sur l'afficheur de la carte TP fonctionnait parfaitement.

```
 #include <C:\Logiciels\PICC\Devices\18F4580.h>
#device ADC=10
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#include <C:\Logiciels\PICC\Drivers\LINBUS.h>
#include <lcd.c>

#byte TXSTA=0xFAC
#bit SENDB=TXSTA.3

unsigned int16 tension1=0,tension=0,tvide=0,courant=0,courant1=0,z=0,t1=0,temperature=0,te=0;
unsigned int s=0,a2=0,a3=0,a4=0,u=0,x=0,c=1;

#int_timer0
void tmr0()
{
set_timer0(3036);                    // initialisation du timer0
s++;                                 // incrementation de la variable s
u++;                                 // incrementation de la variable u
        if (s==21) s=1;              // au bout de 2 sec, s égal à 1
        if (u==10)                   // quand une seconde passe on fait
        {
                u=0;                         // RAZ de la variable u
                z=courant+z;                 // incrementation du la variable z par le courant
                if (z>=3600)                 // si z superieur ou égal à 3600
                {
                if      (tension>150) x=x+1;     // si tension > 150 ( en charge ) on incremente l'état de charge
                        else x=x-1;              // sinon on décremente l'état de charge
                                z=0;             // RAZ de la variable z
}       }       }
```

```
void main ()
{
      setup_adc(ADC_CLOCK_DIV_64);              // convertisseur analogique numerique plus lent mais plus precis
      setup_adc_ports(AN0_TO_AN2);              // AN0 , AN1 , AN2 sont des entrée analogiques
      lcd_init();                               // initialisation de l'afficheur LCD
      enable_interrupts(INT_TIMER0);            // activation du timer0
      enable_interrupts(GLOBAL);                // autorisation de l'utilisation des interruptions
      setup_timer_0(RTCC_INTERNAL|RTCC_DIV_8);      // diviseur de frequence par 8
      set_timer3(3036);                                 // valeur de rechargement pour 100ms

      set_adc_channel(0);           // sur l'éntrée AN0
      tension1=read_adc();          // convertisseur analogique numerique vers la variable tension1
      tvide=(tension1-400);         // cette conversion se fait une seul fois pour estimer l'état de charge à vide
      x=tvide;

set_adc_channel(2);         // sur l'entrée AN2
te=read_adc();              // convertisseur analogiqeu numerique vers la variable te
t1=(te-553);                // cette conversion se fait une seul fois pour estimer l'état de charge à vide
temperature=(t1*40)/103;
if (temperature>=10||temperature<=15) x=x*0.93;     // d'après des ressources, la température fait varier l'état de charge
if (temperature>=40||temperature<10)  x=x*0.85;     // plus y fait froid et plus l'état de charge diminue

while (true)                // boucle infinie
        {
                lcd_gotoxy(1,1);
                lcd_putc("Bat Restant :");
                lcd_gotoxy(14,1);
                printf(lcd_putc,"%3u",x);                   // affichage de la valeur de l'état de charge en %

                if (x>100)                          // si la valeur est superieur à 100%, la batterie est chargée
                {
                        lcd_gotoxy(13,1);
                        lcd_putc("FIN ");
                }
a2=input(PIN_A5);                       // lecture de l'éntrée AN3
a3=input(PIN_A4);                       // lecture de l'éntrée AN4
a4=input(PIN_A3);                       // lecture de l'éntrée AN5
```

```
set_adc_channel(0);                    // sur l'entrée AN0
 tension1=read_adc();                  // convertisseur analogique numerique vers la variable
 tension1tension=(tension1-422)*3/11;  // calcul permettant d'avoir la valeur voulue
set_adc_channel(1);                    // sur l'entrée AN1
courant1=read_adc();                   // convertisseur analogique numerique vers la variable courant1
courant=(courant1-233)*4/7;            // calcul permettant d'avoir la valeur voulue
set_adc_channel(2);                    // sur l'entrée AN2
te=read_adc();                         // convertisseur analogique numerique vers la variable te
t1=(te-553);
temperature=(t1*40)/103;               // calcul permettant d'avoir la valeur voulue

//****************************** TENSION ******************************

                if (a2==0&&a3>0&&a4==0)
                        // si le switch pour AN4 est activé
                {

while (tension>191&&s>1)                        // tant que la tension est superieur à 14.8V
                                {
        lcd_gotoxy(1,2);
        lcd_putc("Batterie Morte   ");
                                }

while (tension>=0&&tension<10&&s>1)             // tant que la tension est inferieur à 11.9V
        {
        lcd_gotoxy(1,2);
        lcd_putc("Batterie Vide   ");
        }

while (tension>=10&&tension<20&&s<40&&s>1)    // tant que la tension est comprise entre 11.9 et 12.1V
        {
             lcd_gotoxy(1,2);
             lcd_putc("Batterie Critiq  ");
         }
```

```
while (tension>=100&&tension<=150&&s>1)          // tant que la tension est comprise entre 13.2 et 14.2V
        {
            lcd_gotoxy(1,2);
            lcd_putc("Batterie Pleine  ");
        }

while (tension>150&&tension<164&&s>1)            // tant que la tension est comprise entre 14.2V et 14.4V
        {
            lcd_gotoxy(1,2);
            lcd_putc("Batt en Charge  ");
        }

while (tension>=164&&tension<190&&s>1)                  // tant que la tension est superieur à 14.4V
        {
            lcd_gotoxy(1,2);
            lcd_putc("DFT Alternateur  ");
        }

while (tension>=20&&tension<100)     // Si les cas au dessus ne sont pas validé alors on affiche sa valuer en %
        {
            lcd_gotoxy(1,2);
            printf(lcd_putc,"Tension   : %lu  ",tension);
                lcd_gotoxy(16,2);
                lcd_putc(0x25);
                    break;
                    }
                }
//*************** COURANT ***********************************

                if (a2>0&&a3==0&&a4==0)                  // si le switch pour AN4 est activé

                {
                    lcd_gotoxy(1,2);
                    lcd_putc("C instant :  ");
                    lcd_gotoxy(14,1);
                    printf(lcd_putc,"%3lu",courant);     // affichage de la valeur du courant instantané
                }
```

```
//****************Veille*****************************************

        if (a2==0&&a3==0&&a4==0)        // si tout les switchs sont desactivés alors on éteint toute visualistation
                {
                        lcd_putc("/f");
                        output_b(0);
                }

//****************LEDS*******************************************
                        if (x>=1&&x<17)
                        {
                                output_b(c);
                                c=c*2;
                                delay_ms(200);
                                if (c>32) c=1;
                        }


                        if (x>=17&&x<33)        output_b(0x01);

                        if (x>=33&&x<50)        output_b(0x03);

                        if (x>=50&&x<67)                output_b(0x07);

                        if (x>=67&&x<81)        output_b(0x0F);

                        if (x>=81&&x<95)        output_b(0x1F);

                        if (x>=95&&x<105)       output_b(0x3F);

                        if (x>=105)
                        {
                                output_b(0x2A);
                                delay_ms(200);
                                output_b(0x15);
                                delay_ms(200);
                        }
```

```
//**************Temperature*************************************************
                if (a2==0&&a3==0&&a4>0)
                {
                        lcd_gotoxy(1,2);
                        // permet l'affichage sur la ligne 1
                        printf(lcd_putc,"Temp : %2lu Degres",temperature);          // affichage des degrés
                        delay_ms(50);
                }
        }
}
```

Programme avec système de dels :

```
#include <C:\Logiciels\PICC\Devices\18F4580.h>
#device ADC=10
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)

unsigned int16 tension1=0,tension=0,tvide=0,courant=0,courant1=0,z=0,t1=0,temperature=0,te=0;
unsigned int s=0,a2=0,a3=0,a4=0,u=0,x=0,c=1;

#int_timer0
void tmr0()
{
set_timer0(3036);                   // initialisation du timer0
s++;                                // incrémentation de la variable s
u++;                                // incrémentation de la variable u
        if (s==21) s=1;             // au bout de 2 sec, s égal à 1
        if (u==10)                  // quand une seconde passe on fait
        {
                u=0;                // RAZ de la variable u
                z=courant+z;        // incrémentation du la variable z par le courant
                if (z>=3600)        // si z supérieur ou égal à 3600
                {
                if      (tension>150) x=x+1;      // si tension > 150 ( en charge ) on incrémente l'état de charge
                        else x=x-1;               // sinon on décrémente l'état de charge
                        z=0;                // RAZ de la variable z
                }
        }
}

void main ()
{
    setup_adc(ADC_CLOCK_DIV_64);            // convertisseur analogique numérique plus lent mais plus précis
    setup_adc_ports(AN0_TO_AN2);            // AN0 , AN1 , AN2 sont des entrée analogiques

    enable_interrupts(INT_TIMER0);          // activation du timer0
    enable_interrupts(GLOBAL);              // autorisation de l'utilisation des interruptions
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_8);        // diviseur de fréquence par 8
```

```
      set_timer3(3036);                  // valeur de rechargement pour 100ms
      set_adc_channel(0);                // sur l'entrée AN0
            tension1=read_adc();         // convertisseur analogique numérique vers la variable tension1
      tvide=((tension1-402)*42)/40;      // cette conversion se fait une seul fois pour estimer l'état de charge à vide
                  x=tvide;
                  set_adc_channel(2);    // sur l'entrée AN2
                  te=read_adc();         // convertisseur analogique numérique vers la variable te
                  t1=(te-553);           // cette conversion se fait une seul fois pour estimer l'état de charge à vide
                  temperature=(t1*40)/103;
if (temperature>=10||temperature<=15) x=x*0.93;    // d'après des ressources, la température fait varier l'état de charge
if (temperature>=40||temperature<10)  x=x*0.85;    // plus y fait froid et plus l'état de charge diminue

while (true)        // boucle infinie
        {
        a2=input(PIN_A5);                  // lecture de l'entrée AN3
        a3=input(PIN_A4);                  // lecture de l'éntrée AN4
        a4=input(PIN_A3);                  // lecture de l'entrée AN5

set_adc_channel(0);                // sur l'entrée AN0
tension1=read_adc();               // convertisseur analogique numérique vers la variable tension1
tension=(tension1-650);            // calcul permettant d'avoir la valeur voulue
set_adc_channel(1);                // sur l'entrée AN1
courant1=read_adc();               // convertisseur analogique numérique vers la variable courant1
courant=(courant1-106);            // calcul permettant d'avoir la valeur voulue
set_adc_channel(2);                // sur l'entrée AN2
te=read_adc();                     // convertisseur analogique numérique vers la variable te
t1=(te-553);
temperature=(t1*40)/103;           // calcul permettant d'avoir la valeur voulue


//****************************** TENSION ****************************
                  while (a2==0&&a3>0&&a4==0)
                  {
a2=input(PIN_A5);                  // lecture de l'éntrée AN3
a3=input(PIN_A4);                  // lecture de l'éntrée AN4
a4=input(PIN_A3);                  // lecture de l'éntrée AN5
```

```
if (tension>=0&&tension<17)
                        {
                                output_toggle(PIN_B0);
                                delay_ms(100);
                                output_toggle(PIN_B1);
                                delay_ms(100);
                                output_toggle(PIN_B2);
                                delay_ms(100);
                                output_toggle(PIN_B3);
                                delay_ms(100);
                                output_toggle(PIN_B4);
                                delay_ms(100);
                                output_toggle(PIN_B5);
                                delay_ms(100);
                        }

                        if (tension>=17&&tension<33)      output_b(0x01);

                        if (tension>=33&&tension<50)      output_b(0x03);

                        if (tension>=50&&tension<67)      output_b(0x07);

                        if (tension>=67&&tension<81)      output_b(0x0F);

                        if (tension>=81&&tension<95)      output_b(0x1F);

                        if (tension>=95&&tension<105)     output_b(0x3F);

                        if (tension>=130)
                        {
                                output_b(c);
                                c=c*2;
                                delay_ms(200);
                                if (c>32) c=1;
                        }
                }
```

```
//*************** COURANT ***********************************

while (a2>0&&a3==0&&a4==0)
{
                a2=input(PIN_A5);                       // lecture de l'entrée AN3
                a3=input(PIN_A4);                       // lecture de l'entrée AN4
                a4=input(PIN_A3);                       // lecture de l'entrée AN5
                if (courant>=0&&courant<7)          output_b(0x03);
                if (courant>=7&&courant<14)         output_b(0x0F);
                if (courant>=14&&courant<=21)       output_b(0x3F);
}

//***************Veille****************************************

if (a2==0&&a3==0&&a4==0)        output_b(0);
// si tout les switchs sont désactives alors on éteint toute visualisation

//****************LEDS*****************************************
                while (a2>0&&a3>0&&a4>0)
                {
                        a2=input(PIN_A5);                       // lecture de l'entrée AN3
                        a3=input(PIN_A4);                       // lecture de l'entrée AN4
                        a4=input(PIN_A3);                       // lecture de l'entrée AN5
                        if (x>=0&&x<17)
                        {
                                output_toggle(PIN_B0);
                                delay_ms(200);
                        }

                        if (x>=17&&x<33)            output_b(0x01);

                        if (x>=33&&x<50)            output_b(0x03);

                        if (x>=50&&x<67)            output_b(0x07);

                        if (x>=67&&x<81)            output_b(0x0F);
```

```
                    if (x>=81&&x<95)        output_b(0x1F);

                    if (x>=95&&x<105)       output_b(0x3F);

                    if (x>=105)             // si la valeur est supérieur à 100%, la batterie est chargée
                    {
                        output_b(0x2A);
                        delay_ms(100);
                        output_b(0x15);
                        delay_ms(100);
                    }
                }
            }
}
```