Université François-Rablelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

Projet tuteuré

Battery Management System for Lithium Batteries

LEBRUN Alexandre
REY Kevin
2ème Année – Groupe Q1
Promotion 2010-2012

Enseignants :
LEQUEU Thierry
GLIKSOHN Charles

Table des matières

Introduction

Un B.M.S.[1], est, comme son nom l'indique, un système permettant la gestion des batteries, et plus précisément, dans notre étude, des batteries de technologie lithium-ion.

Il permet à l'utilisateur une utilisation plus aisée de la batterie et d'avoir en temps réel différentes informations sur celle-ci. En effet, une batterie lithium-ion est assez fragile, et pour que celle-ci ait une durée de vie maximale et ne soit pas un danger, certains paramètres doivent être surveillés.

Un premier paramètre à surveiller est bien entendu la tension, ces batteries ne supportent pas les décharges profondes, elles ne doivent pas être déchargées de plus de 15 % de leur charge maximale. Au contraire, une surcharge n'est pas très grave, mais cela induit une élévation de la température à cause de la dissipation de l'énergie et la température est justement l'autre paramètre à surveiller. Une température trop élevée peut même conduire jusqu'à l'explosion de celle-ci. Pour un fonctionnement optimal, la batterie doit être utilisée à moins de 40 °C avec peu de variation de température et à l'abri du soleil.

Ce système s'occupe donc de sécuriser la batterie sur ces différents risques. Au travers de notre projet, nous concevrons une maquette simulant une batterie et un programme gérant cette maquette à la manière d'un B.M.S. dans le but que celui-ci soit adapté plus tard au kart électrique.

Notre étude débutera par la présentation du cahier des charges et du planning. Puis nous verrons la conception de la maquette simulant la batterie avant de terminer sur la programmation du microcontrôleur qui s'occupera de gérer la maquette.

1 Batterie Management System

1. Cahier des charges

1.1. Présentation générale du projet

Le B.M.S. est un système permettant la surveillance de la charge et décharge d'une batterie. Ce système permettra de gérer la batterie du kart électrique afin de prévenir le conducteur de l'état en temps réel de la batterie et ainsi savoir s'il est nécessaire de la recharger. Il surveillera aussi la charge de celle-ci pour empêcher toute surcharge. De plus, le B.M.S. mesurera la température de la batterie afin de prévenir en cas de surchauffe de celle-ci.

1.2. Contraintes

Le système devra être capable de mesurer les différentes tensions à l'aide d'un microcontrôleur ATmega8535, limité à 5V par entrée, et la mesure devra être la plus précise possible. La mesure de tension utilisera les 8 entrées du port analogique du microcontrôleur, il faudra mesurer la température de la batterie et les différentes informations devront être affichées sur un écran LCD.

1.3. Solutions envisagées

Pour débuter, nous reprendrons un ancien projet traitant du même sujet. La maquette étant en mauvais état, nous la referons et en profiterons pour l'améliorer afin de mieux assurer sa charge et ainsi éviter l'explosion des condensateurs.

La mesure des tensions se fera avec les entrées analogiques du microcontrôleur et la mesure de température via une liaison I^2C[2]. Nous rechercherons un matériel compatible tel que le LM75 qui sera étudié plus tard si cette solution est retenue.

2 Inter Integrated Circuit

2. Planning

Illustration 1: Planning prévisionnel et réel [7]

Comme nous pouvons le constater, la conception de la carte a pris plus de temps que nous l'espérions, ce qui nous a conduit à un retard sur l'ensemble du projet. Nous nous étions laissé une semaine pour parer à un éventuel souci inattendu ce qui nous a permis de minimiser l'effet de ce retard.

Nous observons aussi que la programmation a été aussi longue et qu'elle a débuté en même temps que la mise à jour du programme pour l'I²C. Cela s'explique par le fait qu'étant donné le retard pris, nous avons préféré traiter la surveillance des tensions et de la température en même temps pour gagner un peu de temps, les tests se sont donc, par conséquent, réalisés en même temps eux aussi.

3. Conception de la plaque de test

3.1. Étude théorique de la maquette

Nous avions au début une maquette à disposition permettant de simuler la batterie du Kart électrique, mais celle-ci était inutilisable (des composants étaient manquants et la carte était en mauvais état). Nous avons donc préféré la refaire en ajoutant une protection supplémentaire, des diodes Zener permettant de limiter la tension aux bornes de chaque condensateur.

3.1.1. Les cellules de la batterie

La carte que nous avons faite, visible en annexe, s'inspire directement de la carte que nous avions au début, c'est une carte sur laquelle sont disposés 8 condensateurs représentant 8 accumulateurs d'une batterie.

Dans une batterie lithium-ion, chaque élément a une tension nominale de 3,6 V ou 3,7 V. Dans notre carte les condensateurs sont limités à 2,3 V, mais cela ne changera que très peu de chose dans le programme, nous expliquerons pourquoi après avec la mesure de tension.

Le choix des condensateurs est dû au fait que nous avons besoin d'une charge et d'une décharge rapide, mais visible à l'échelle humaine, pour notre projet. Ils ne pourraient pas remplacer des accumulateurs dans un cas réel.

Nos condensateurs ont une capacité de 10 Farads, nous verrons ce que cela implique sur leur décharge au cours du temps à cause du reste du montage un peu plus loin. Comme nous pouvons le constater, ce ne sont pas des condensateurs standards, ce sont en fait des supercondensateurs, car ils ont une capacité importante.

Illustration 2: Un supercondensateur [8]

3.1.2. La mesure de tension

Pour la mesure de tension aux bornes des condensateurs, nous avons besoin d'abaisser la tension, car l'ATmega8535 peut lire des tensions analogiques jusqu'à 5 V. Une solution simple pour faire cela est l'utilisation des ponts diviseurs de tension. Le microcontrôleur peut adapter la valeur retournée par le convertisseur analogique-numérique. Nous utiliserons une comparaison par rapport à une tension de référence interne (qu'il génère lui-même) ce qui nous laisse le choix entre 5 V et 2,56 V. Pour avoir un maximum de précision nous avons choisi 2,56 V.

Nous avons récemment dit que le programme ne sera que très peu influencé par la tension des éléments qu'il mesurera, c'est en effet dû au fait qu'on adapte la tension lue par le microcontrôleur. De ce fait, seul l'affichage de la tension sera à modifier pour une batterie.

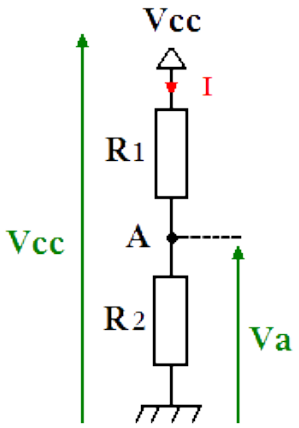


Illustration 3: Principe du pont diviseur de tension [9]

3.1.3. Les diodes Zener

Les diodes Zener sont des diodes qui permettent de limiter la tension à leurs bornes lorsque celles-ci sont branchées en inverse. Dans notre cas, brancher une diode Zener en parallèle de chaque condensateur nous aurait permis de limiter leur tension à la tension Zener et ainsi de ne pas dépasser la tension imposée par ceux-ci.

Mais nous n'avons pas pu trouver de diodes avec une tension Zener de 2,3 V. Nous avons alors trouvé une autre solution, nous avons utilisé des diodes standards montées en série.

En effet, une diode a une tension à ses bornes de 0,6 V en théorie, nous en avons monté 3 en série et ainsi, lorsque la tension dépasse 1,8V les diodes se mettent à conduire limitant ainsi la tension aux bornes des condensateurs.

Cette solution n'est pas parfaite, les tensions théoriques sont rarement les tensions que l'on mesure en pratique pour des composants standards tels qu'une diode, mais cela permettra une stabilisation de la tension des condensateurs pour limiter les risques dus aux déséquilibres des tensions causés par les résistances servant à former le pont diviseur de tension.

3.2. Réalisation

3.2.1. Calcul des résistances des ponts diviseurs de tension

Le calcul des résistances est une tâche qui peut devenir très rapidement fastidieuse, car nous sommes confrontés à un problème assez important.

Par exemple, dans notre cas, nous voulons 2,56 V en sortie du pont diviseur de tension pour tous nos condensateurs. Mais il faut calculer les résistances pour chacun d'entre eux, car la tension vue entre la masse et la borne positive du condensateur varie suivant sa position.

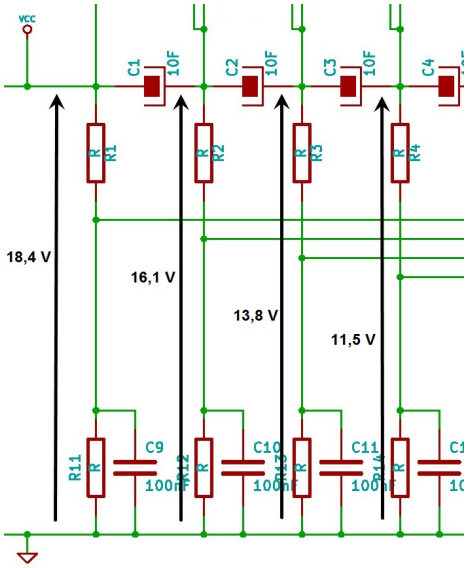


Illustration 4: Chaque condensateur a une valeur différente par rapport à la masse [7]

De plus, nous voulons que la tension de sortie se rapproche au maximum de 2,56 V, mais sans la dépasser, car c'est la tension maximale admissible par le microcontrôleur.

Nous avons alors cherché un système qui nous permettrait de calculer le meilleur couple de résistance pour avoisiner la tension maximale voulue. La réponse se trouvait dans la formule du pont diviseur de tension :

$$\frac{V_s}{V_e} = \frac{R_1}{(R_1 + R_2)}$$

Il suffit de créer une feuille de calcul capable de comparer les rapports de $\frac{V_s}{V_e}$ et de $\frac{R_1}{(R_1+R_2)}$ afin de déterminer les valeurs des résistances pour lesquelles elles sont le plus proche.

Ve max =	16,1
Vs max =	2,56
Vs/Ve =	0,15901

Coefficient multiplicatif 1

Ri\R1i	10	11	12	13	15	16	18	20	22	24	27	30	33	36	39	43	47	51	56	62	68	75	82	91
100	0,0909	0,0991	0,1071	0,115	0,1304	0,1379	0,1525	0,1667	0,1803	0,1935	0,2126	0,2308	0,2481	0,2647	0,2806	0,3007	0,3197	0,3377	0,359	0,3827	0,4048	0,4286	0,4505	0,4764
110	0,0833	0,0909	0,0984	0,1057	0,12	0,127	0,1406	0,1538	0,1667	0,1791	0,1971	0,2143	0,2308	0,2466	0,2617	0,281	0,2994	0,3168	0,3373	0,3605	0,382	0,4054	0,4271	0,4527
120	0,0769	0,084	0,0909	0,0977	0,1111	0,1176	0,1304	0,1429	0,1549	0,1667	0,1837	0,2	0,2157	0,2308	0,2453	0,2638	0,2814	0,2982	0,3182	0,3407	0,3617	0,3846	0,4059	0,4313
130	0,0714	0,078	0,0845	0,0909	0,1034	0,1096	0,1216	0,1333	0,1447	0,1558	0,172	0,1875	0,2025	0,2169	0,2308	0,2486	0,2655	0,2818	0,3011	0,3229	0,3434	0,3659	0,3868	0,4118
150	0,0625	0,0683	0,0741	0,0798	0,0909	0,0964	0,1071	0,1176	0,1279	0,1379	0,1525	0,1667	0,1803	0,1935	0,2063	0,2228	0,2386	0,2537	0,2718	0,2925	0,3119	0,3333	0,3534	0,3776
160	0,0588	0,0643	0,0698	0,0751	0,0857	0,0909	0,1011	0,1111	0,1209	0,1304	0,1444	0,1579	0,171	0,1837	0,196	0,2118	0,2271	0,2417	0,2593	0,2793	0,2982	0,3191	0,3388	0,3625
180	0,0526	0,0576	0,0625	0,0674	0,0769	0,0816	0,0909	0,1	0,1089	0,1176	0,1304	0,1429	0,1549	0,1667	0,1781	0,1928	0,207	0,2208	0,2373	0,2562	0,2742	0,2941	0,313	0,3358
200	0,0476	0,0521	0,0566	0,061	0,0698	0,0741	0,0826	0,0909	0,0991	0,1071	0,1189	0,1304	0,1416	0,1525	0,1632	0,177	0,1903	0,2032	0,2188	0,2366	0,2537	0,2727	0,2908	0,3127
220	0,0435	0,0476	0,0517	0,0558	0,0638	0,0678	0,0756	0,0833	0,0909	0,0984	0,1093	0,12	0,1304	0,1406	0,1506	0,1635	0,176	0,1882	0,2029	0,2199	0,2361	0,2542	0,2715	0,2926
240	0,04	0,0438	0,0476	0,0514	0,0588	0,0625	0,0698	0,0769	0,084	0,0909	0,1011	0,1111	0,1209	0,1304	0,1398	0,1519	0,1638	0,1753	0,1892	0,2053	0,2208	0,2381	0,2547	0,2749
270	0,0357	0,0391	0,0426	0,0459	0,0526	0,0559	0,0625	0,069	0,0753	0,0816	0,0909	0,1	0,1089	0,1176	0,1262	0,1374	0,1483	0,1589	0,1718	0,1867	0,2012	0,2174	0,233	0,2521
300	0,0323	0,0354	0,0385	0,0415	0,0476	0,0506	0,0566	0,0625	0,0683	0,0741	0,0826	0,0909	0,0991	0,1071	0,115	0,1254	0,1354	0,1453	0,1573	0,1713	0,1848	0,2	0,2147	0,2327
330	0,0294	0,0323	0,0351	0,0379	0,0435	0,0462	0,0517	0,0571	0,0625	0,0678	0,0756	0,0833	0,0909	0,0984	0,1057	0,1153	0,1247	0,1339	0,1451	0,1582	0,1709	0,1852	0,199	0,2162
360	0,027	0,0296	0,0323	0,0349	0,04	0,0426	0,0476	0,0526	0,0576	0,0625	0,0698	0,0769	0,084	0,0909	0,0977	0,1067	0,1155	0,1241	0,1346	0,1469	0,1589	0,1724	0,1855	0,2018
390	0,025	0,0274	0,0299	0,0323	0,037	0,0394	0,0441	0,0488	0,0534	0,058	0,0647	0,0714	0,078	0,0845	0,0909	0,0993	0,1076	0,1156	0,1256	0,1372	0,1485	0,1613	0,1737	0,1892
430	0,0227	0,0249	0,0271	0,0293	0,0337	0,0359	0,0402	0,0444	0,0487	0,0529	0,0591	0,0652	0,0713	0,0773	0,0832	0,0909	0,0985	0,106	0,1152	0,126	0,1365	0,1485	0,1602	0,1747
470	0,0208	0,0229	0,0249	0,0269	0,0309	0,0329	0,0369	0,0408	0,0447	0,0486	0,0543	0,06	0,0656	0,0711	0,0766	0,0838	0,0909	0,0979	0,1065	0,1165	0,1264	0,1376	0,1486	0,1622
510	0,0192	0,0211	0,023	0,0249	0,0286	0,0304	0,0341	0,0377	0,0414	0,0449	0,0503	0,0556	0,0608	0,0659	0,071	0,0778	0,0844	0,0909	0,0989	0,1084	0,1176	0,1282	0,1385	0,1514
560	0,0175	0,0193	0,021	0,0227	0,0261	0,0278	0,0311	0,0345	0,0378	0,0411	0,046	0,0508	0,0556	0,0604	0,0651	0,0713	0,0774	0,0835	0,0909	0,0997	0,1083	0,1181	0,1277	0,1398
620	0,0159	0,0174	0,019	0,0205	0,0236	0,0252	0,0282	0,0313	0,0343	0,0373	0,0417	0,0462	0,0505	0,0549	0,0592	0,0649	0,0705	0,076	0,0828	0,0909	0,0988	0,1079	0,1168	0,128
680	0,0145	0,0159	0,0173	0,0188	0,0216	0,023	0,0258	0,0286	0,0313	0,0341	0,0382	0,0423	0,0463	0,0503	0,0542	0,0595	0,0646	0,0698	0,0761	0,0836	0,0909	0,0993	0,1076	0,118
750	0,0132	0,0145	0,0157	0,017	0,0196	0,0209	0,0234	0,026	0,0285	0,031	0,0347	0,0385	0,0421	0,0458	0,0494	0,0542	0,059	0,0637	0,0695	0,0764	0,0831	0,0909	0,0986	0,1082
820	0,012	0,0132	0,0144	0,0156	0,018	0,0191	0,0215	0,0238	0,0261	0,0284	0,0319	0,0353	0,0387	0,0421	0,0454	0,0498	0,0542	0,0586	0,0639	0,0703	0,0766	0,0838	0,0909	0,0999
910	0,0109	0,0119	0,013	0,0141	0,0162	0,0173	0,0194	0,0215	0,0236	0,0257	0,0288	0,0319	0,035	0,0381	0,0411	0,0451	0,0491	0,0531	0,058	0,0638	0,0695	0,0761	0,0827	0,0909

Coefficient multiplicatif 10

Illustration 5: Exemple du tableau utilisé pour calculer les résistances [7]

Grâce à ce tableau, nous avons pu déterminer rapidement les valeurs des résistances à prendre.

Cellule	Ri retenue	R1i retenue	Vi	Vsi	erreur
C1	62	10	18,4	2,556	-0,004
C2	360	68	16,1	2,558	-0,002
C3	330	75	13,8	2,556	-0,004
C4	56	16	11,5	2,556	-0,004
C5	39	15	9,2	2,556	-0,004
C6	56	33	6,9	2,558	-0,002
C7	12	15	4,6	2,556	-0,004
C8	0	15	2,3	2,3	-0,26

Illustration 6: Tableau des résultats obtenus [7]

3.2.2. Impact des résistances sur la charge des condensateurs

Nous allons maintenant voir le principal problème causé par l'utilisation de ponts diviseurs de tension. Dans notre cas, ils nous permettent de récupérer une tension avoisinant les 2,56 V lorsque la cellule est pleinement chargée. C'est une solution assez simple à mettre en œuvre et peu coûteuse, car elle ne nécessite que 2 résistances.

Mais ce système provoque une décharge constante des condensateurs dans les résistances des ponts diviseurs de tension.

Pour réduire la puissance dissipée par les résistances des ponts il faut placer de fortes résistances afin de limiter au maximum le courant les traversants.

3.2.3. LED témoin de tension

Pour avertir de la présence d'une tension dans la batterie, nous avons ajouté une LED témoin. Une LED standard s'alimente sous une tension de 1,2V et ne doit pas recevoir plus de 20 mA sous peine de claquer.

Pour notre maquette il nous fallait donc :

$$R = \frac{U_{Total} - U_{LED}}{I} = \frac{20\,V - 1{,}2\,V}{20\,mA} = 940\ \Omega$$ soit une résistance de 1 kΩ.

Or la puissance dissipée par la résistance dépasse le quart de Watt. Une résistance standard brûlerait :

$$P = U \cdot I = 18{,}8\,V \times 20\,mA = 376\,mW > 250\,mW$$

Afin de résoudre ce problème, il faut utiliser une résistance de puissance ou augmenter la valeur de la résistance pour diminuer la puissance dissipée. Une troisième solution consiste à placer deux résistances en parallèle pour répartir le courant. Nous avons choisi d'augmenter la résistance et de mettre 2,2 kΩ.

$$P = \frac{U^2}{R} = \frac{(18{,}8\,V)^2}{2200\,\Omega} = 161\,mW < 250\,mW$$

Ainsi la LED et la résistance ne craignent pas de brûler et la luminosité de la LED est amplement suffisante pour être visible. De plus, comme la résistance est élevée, leur consommation est moindre.

Théoriquement, la LED devait avertir l'utilisateur de la présence d'une tension dans la batterie. Après réflexion, cette LED n'est pas utile, car la batterie alimentera toujours la LED et elle sera constamment allumée.

Il aurait été préférable de placer cette LED plus haut afin de, par exemple, montrer que la batterie est en charge.

3.2.4. Nomenclature

Nous allons maintenant vous présenter la liste des composants que nous avons utilisés ainsi que leur prix afin de pouvoir estimer le coût de notre projet.

Composants	Quantité	Prix à l'unité (€)
Afficheur LCD 16*4	1	16,25
ATmega8535	1	4,99
capteur de température LM75	1	1,96
Connecteur HE10 20 broches	1	1,88
Bornes d'alimentation	2	1
LED	1	0,2
Condensateur 10F / 2,3V	8	4,34
Résistance	18	0,02
Diode 1N4148	24	0,11
Condensateur 100nF	8	0,12
Prix total		65,96

Illustration 7: Nomenclature [7]

Ce prix est approximatif et risque d'être plus élevé dans la réalité, car tous les composants étaient à disposition. De plus, nous ne comptons pas la carte sur laquelle était l'ATmega8535.

4. Programmation de l'ATmega8535

4.1. Présentation des différents éléments utilisés

Comme vu précédemment, nous avons utilisé plusieurs composants et nous avons mis en œuvre un bus I²C pour communiquer avec l'un d'entre eux. Nous allons donc maintenant vous présenter ces différents éléments.

4.1.1. L'ATmega8535

Illustration 8: Plaque de programmation et de test de l'ATmega8535 [4]

L'ATmega8535 est le microcontrôleur que nous allons programmer afin qu'il puisse gérer la batterie.

Dans ce projet, l'ATmega8535 est déjà installé sur une carte. Il suffit de brancher l'ATmega8535 au bus du PC et de compiler le programme à l'aide d'un logiciel comme AVR studio ou CodeVisionAVR. Une fois la compilation terminée, le programme est chargé dans le microcontrôleur et est exécuté.

4.1.2. L'afficheur LCD

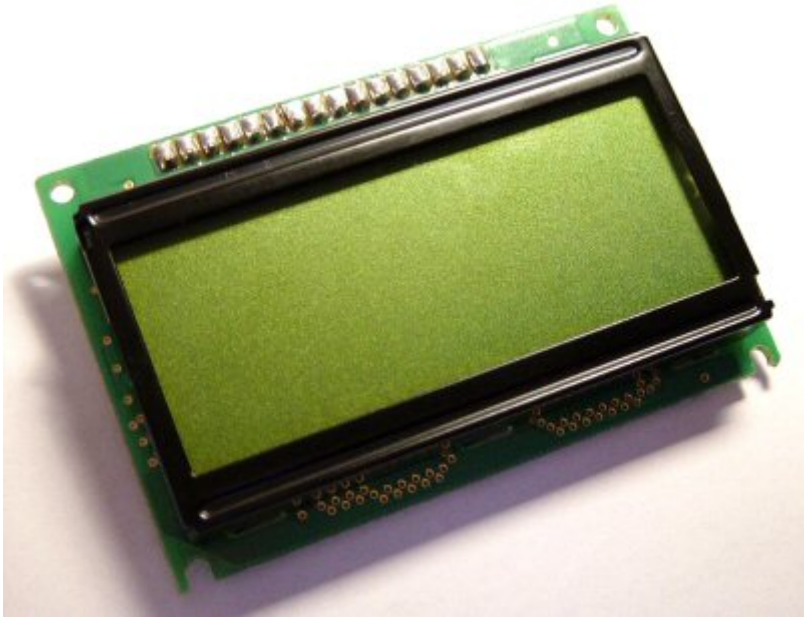


Illustration 9: Afficheur LCD [4]

Pour pouvoir communiquer avec l'utilisateur et l'informer des tensions et de la température, il nous a fallu utiliser un écran. Pour cela, nous avons gardé l'afficheur fourni avec le kit de programmation de l'ATmega8535. Cet afficheur a une dimension de 4*16 caractères, c'est-à-dire 4 lignes de 16 caractères chacune. Cela nous permettra d'afficher de manière lisible toutes les informations nécessaires.

4.1.3. Le LM75

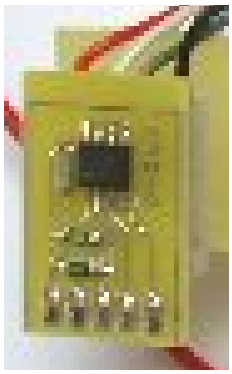


Illustration 10: Le LM75 [4]

Le LM75 est un composant CMS[3], cela signifie que ses broches sont soudées directement sur le dessus de la plaque sans la traverser. Ce circuit intégré permet de mesurer la température et de la restituer à un autre composant en communiquant via un bus I²C que nous présenterons après.

3 CMS : Composant Monté en Surface

4.1.4. L'I²C

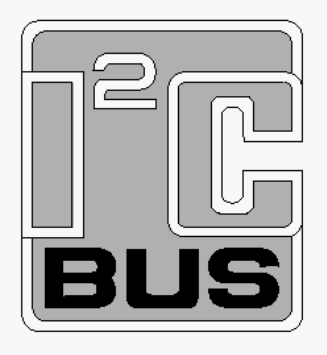


Illustration 11: Logo de l'I²C [10]

L'I²C est un moyen de faire communiquer des composants entre eux. Pour pouvoir raccorder des composants en I²C, ils doivent être munis d'une interface adéquate.

Un bus I²C se compose de 3 fils :

- Un premier servant à diffuser les données que l'on nomme SDA ;
- Un deuxième permettant de transmettre le signal d'horloge nommé SCL ;
- Et le dernier est un fils de référence relié à la masse.

Un bus I²C est donc un moyen simple et peu coûteux de relier des composants entre eux. Mais ce bus ne permet pas de brancher un nombre important d'appareils, par exemple avec le LM75, nous sommes limités à 8.

4.2. Le programme

Nous allons maintenant expliquer les lignes les plus importantes du programme ainsi que son fonctionnement global.

4.2.1. Ordinogramme

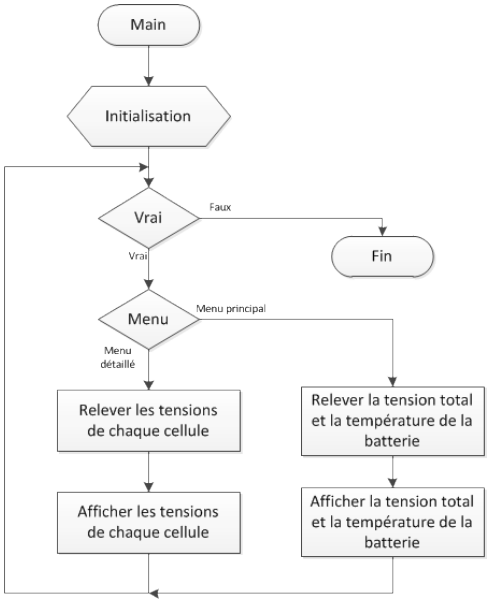


Illustration 12: Ordinogramme du programme [7]

4.2.2. Description générale

Dans le langage utilisé, le C, lorsque le microcontrôleur se lance, il exécute la fonction principale appelée « main ». C'est dans cette fonction que se trouve toute la structure du programme. Nous avons choisi de ne pas utiliser d'interruption,[4] car cela ne nous était pas indispensable.

Dans un premier temps, on appellera la fonction « Initialisation » qui permet de configurer les différents registres ainsi que les ports de l'ATmega8535. Ensuite, nous affichons un message d'accueil avant d'entrer dans la boucle infinie nécessaire au bon fonctionnement de l'ATmega8535 où se trouve le programme.

L'interface utilisateur se compose d'un bouton qui permet de changer de menu.

Le programme permet d'afficher deux menus. L'un permet d'afficher la tension totale de la batterie ainsi que sa température, c'est le menu principal. Et l'autre, affiche en détail les tensions présentent sur chacune des cellules.

Lorsque le programme détecte un problème, une surtension, une surchauffe ou un niveau de batterie faible, il avertit l'utilisateur en affichant un message sur le menu principal.

Pour permettre au programme de s'adapter à différentes configurations, nous avons choisi de définir des constantes de préprocesseur sur certaines valeurs. Ainsi, lorsque celles-ci sont modifiées, le programme est automatiquement réadapté à la configuration choisie après la compilation.

4 Une interruption est une fonction spéciale qui se lance à un événement défini. Lorsqu'elle est lancée, le reste du programme est gelé le temps de son exécution.

Conclusion

En conclusion, nous avons pu terminer notre projet dans les temps impartis. Le fait d'avoir changé de stratégie pour traiter le programme en une seule fois nous a permis de gagner un peu de temps, d'autant plus que l'I²C était très simple à programmer.

Ce projet était très intéressant, il nous a permis d'en apprendre un peu plus au sujet des batteries au lithium-ion qui sont aujourd'hui très courantes et les moyens de maximiser leurs durées de vie.

Lors des premiers tests, nous avons constaté que les mesures de tension étaient incorrectes. Elles étaient dues à une erreur lors du routage de la carte, en effet, deux des sorties des ponts diviseurs de tension n'étaient pas reliées au PORTA de l'ATmega8535. Il nous a alors fallu couper puis souder un fil relié à la bonne entrée pour chacune des deux broches concernées afin de résoudre le problème.

Maintenant, comme voulu, le programme affiche les tensions de chaque cellule ou la tension totale de la batterie ainsi que sa température. Mais les mesures de tension ne sont pas toujours très précises, ce qui n'est pas vraiment gênant, car les erreurs sont faibles malgré tout.

Résumé

Ce projet a pour but la réalisation d'une carte qui jouera le rôle de carte de test et d'un programme qui permettra de mesurer différentes données. La carte est un prototype de batterie sur lequel on effectuera des tests de mesures de tension.

On place sur la carte des condensateurs de plusieurs Farads afin de simuler le fonctionnement d'une batterie Lithium-ion où chaque condensateur représente une cellule. Le programme permet de mesurer et d'afficher la tension aux bornes de chaque cellule (ou condensateur). Pour obtenir une meilleure précision, nous avons décidé de prendre une tension de référence de 2,5V à l'intérieur de l'ATmega8535, ainsi le microprocesseur donnera une valeur plus précise des tensions. Nous utilisons des ponts diviseurs de tension afin de pouvoir mesurer la tension aux bornes de chaque condensateur aux alentours de la tension de référence. L'ATmega8535 reçoit par le bus I²C les mesures de la température. L'I²C envoie des données sous forme de trames qui, une fois traitées par le microprocesseur, permettent d'obtenir la valeur de la température.

Une fois exécuté, le programme traite les différentes données et les affiche à l'aide d'un écran LCD, cela permet à l'utilisateur de connaître l'état de la batterie et la température de celle-ci.

Afin de prévenir l'utilisateur d'un problème le programme affichera des messages d'alertes pour identifier l'origine du problème, tel qu'une surchauffe ou une surcharge de la batterie ou indiquer qu'il est nécessaire de recharger la batterie.

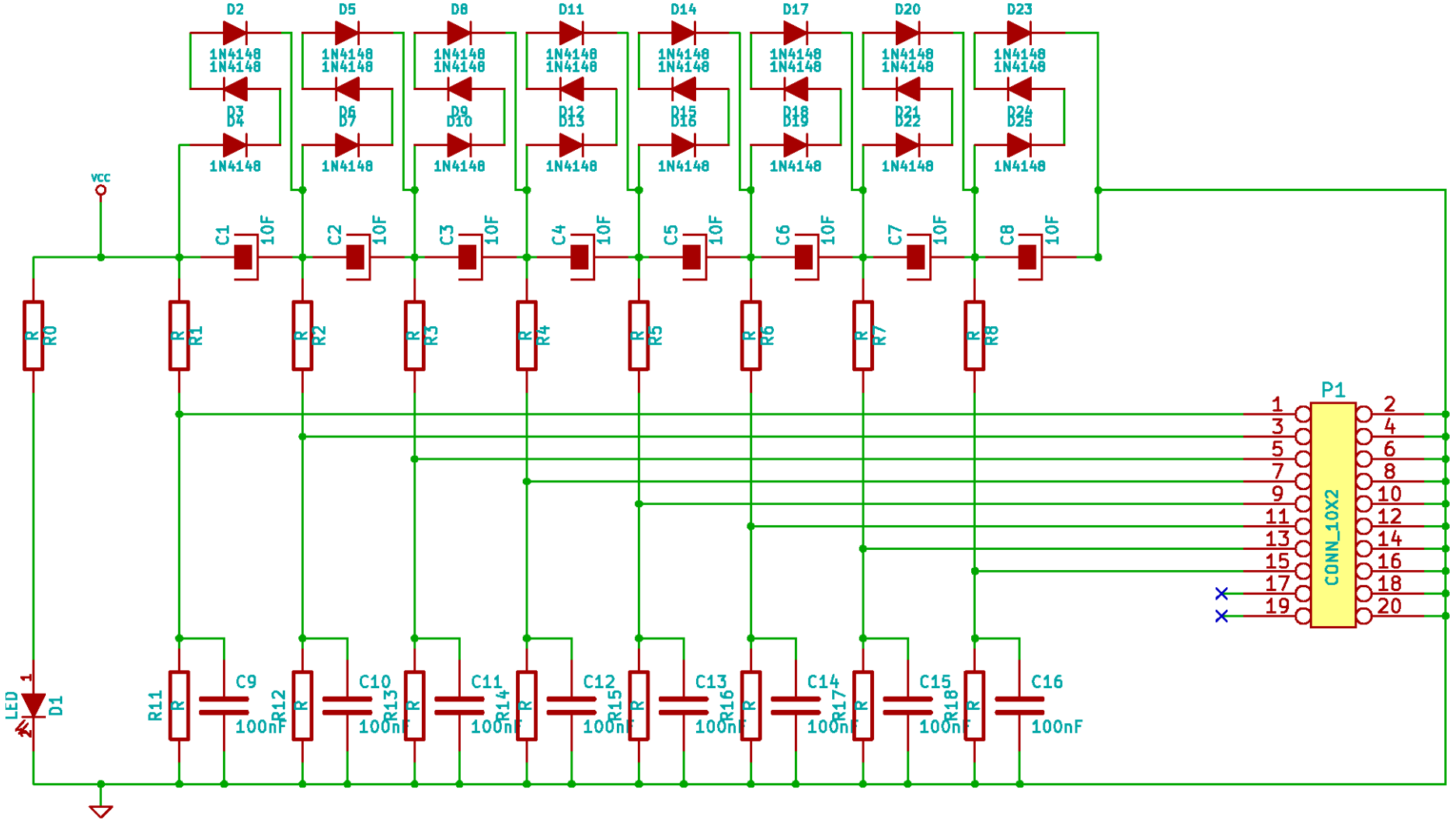
Index des illustrations

Bibliographie

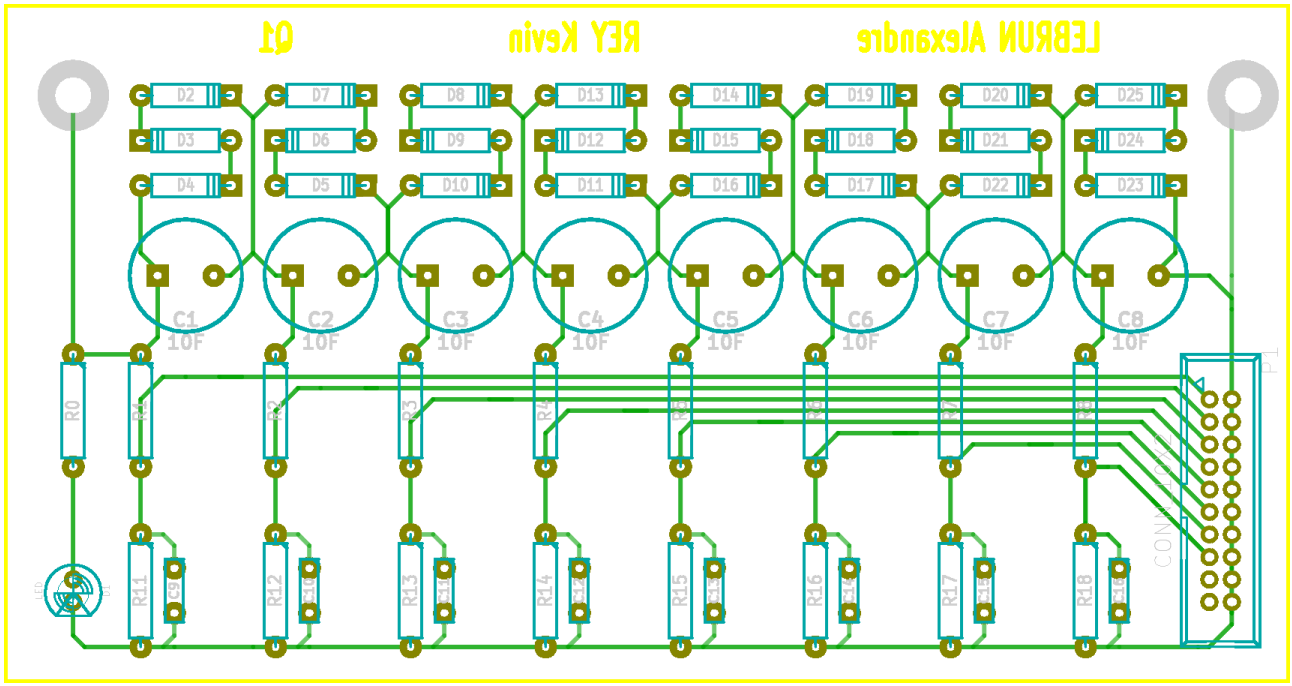
[1] **Vincent**. *Li-ion : entretenir et optimiser sa batterie*, 2011, [En ligne]. (Page consultée le 10/02/2012) <http://blog.mobileoo.fr/2011/06/22/technologie/comment-entretenir-et-optimiser-sa-batterie-li-ion/>
[2] **National Semiconductor**. *LM75*, 2005.
[3] **Atmel**. *ATmega8535(L)*, 2006.
[4] **Thierry LEQUEU**. *La documentation de Thierry LEQUEU sur OVH*, 2012, [En ligne]. (Page consultée le 01/02/2012) <http://www.thierry-lequeu.fr/>
[5] **Aurélien Jarno**. *Le bus I²C*, 2008, [En ligne]. (Page consultée le 01/02/2012) <http://www.aurel32.net/elec/i2c.php>
[6] **PoBot**. *Les batteries Li-Ion et Li-PO*, 2011, [En ligne]. (Page consultée le 10/02/2012) <http://www.pobot.org/Les-batteries-Li-Ion-et-Li-PO.html>
[7] **Kevin REY & Alexandre LEBRUN**. *Réalisations personnelles*, 2012.
[8] **Farnell**. , 2012, [En ligne]. (Page consultée le 10 mars 2012) <http://fr.farnell.com/panasonic/eechw0d106/condensateur-10f-2-3v/dp/9696695>
[9] **Tony Archambeau**. *Diviseur de Tension*, 2012, [En ligne]. (Page consultée le 10 mars 2012) <http://www.elektronique.fr/cours/diviseur-de-tension.php>
[10] **Marc BOUGET**. *Le bus I2C*, 2008, [En ligne]. (Page consultée le 24 mars 2012) <http://mbouget.perso.neuf.fr/>

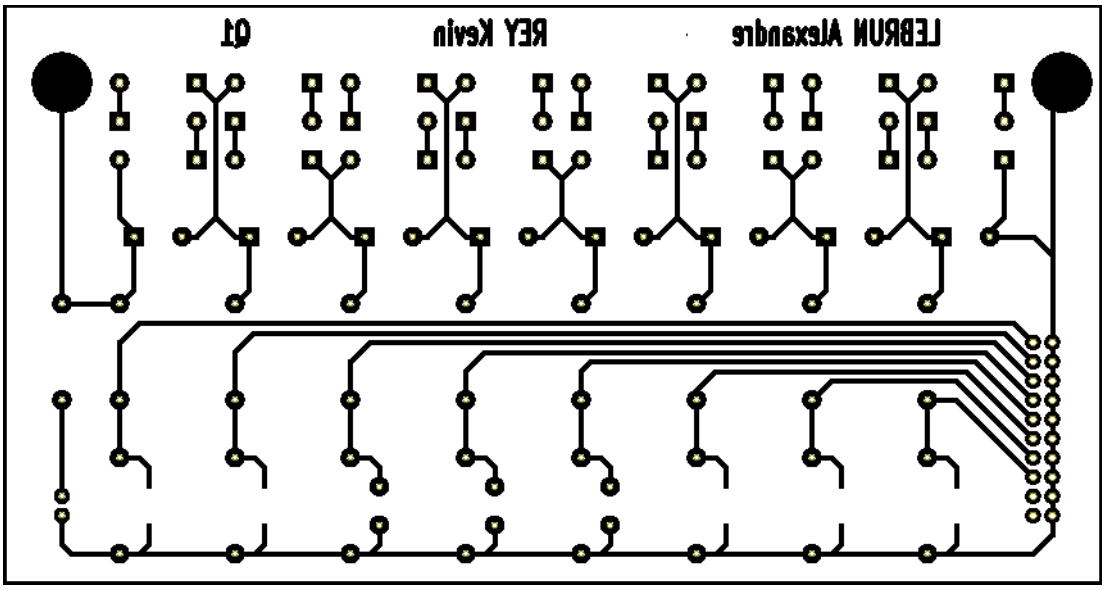
Annexes

Schéma électrique



Typon


Q1
REY Kevin
LEBRUN Alexandre


Q1
REY Kevin
LEBRUN Alexandre

Programme complet

```
/*****************************************************
Projet   : BMS
Version  : 2.0
Date     : 12/03/2012
Auteurs  : REY Kevin & LEBRUN Alexandre
Compagnie: GEII


Chip type           : ATmega8535
Clock frequency     : 8,000 MHz
*****************************************************/

// Déclaration des librairies
#include <mega8535.h>
#include <delay.h>
#include <stdlib.h>

//Définition des constantes
#define TAB_TENS_MAX {230, 230, 230, 230, 230, 230, 230, 256} //Tableau des valeurs réelles des
tensions *100 pour une mesure de 2.56V aux entrées analogiques
#define TENS_MAX 1650 //Tension maximale de la batterie *100
#define TENS_MIN 1300 //Tension minimale de la batterie *100 (demande de charge)
#define TEMP_MAX 400 //Température maximale de la batterie *10
#define PORT_TENS_MAX PORTD.0 //Broche image de l'état de surtension (actif à 0)
#define PORT_TENS_MIN PORTD.1 //Broche image de l'état de batterie faible (actif à 0)
#define PORT_TEMP_MAX PORTD.2 //Broche image de l'état de surchauffe (actif à 0)
#define PIN_TENS_MAX PIND.0
#define PIN_TENS_MIN PIND.1
#define PIN_TEMP_MAX PIND.2

// Déclarations du bus I2C
#asm
    .equ __i2c_port=0x18 ;PORTB
    .equ __sda_bit=0
    .equ __scl_bit=1
#endasm
#include <i2c.h>

// Déclaration du capteur de température LM75
#include <lm75.h>

// Déclaration du module LCD
#asm
    .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

// Déclaration des variables globales
const int TabTensMax[] = TAB_TENS_MAX; //Tableau des valeurs réelles des tensions *100 pour une
mesure de 2.56V aux entrées analogiques
int TabTens[] = {0,0,0,0,0,0,0,0};
int TempBatt = 0;
int TensBatt = 0;
int Alert[] = {1,1,1};

// Fonction d'initialisation
void Initialisation(void)
{
    // Initiation des ports
    PORTA=0x00;
    DDRA=0x00;

    PORTB=0x00;
    DDRB=0x00;

    PORTD=0x00;
    DDRD=0xFF;

    // Initialisation du CAN
    // Référence du comparateur interne à 2,56 Volts
    //ACSR=0x80; //comparateur désactivé
    ADCSRA=0x86;
    ADMUX=0xE0; //référence de comparaison interne = 2.56V et résultat de conversion alignée à
```

```
droite
    SFIOR=0x00;

    // Initialisation du I2C
    i2c_init();

    // Initialisation du LM75
    // thyst: 20 °C
    // tos: 40 °C
    // O.S. polarity: 0
    lm75_init(0,20,40,0); // adresse 0, température de déclenchement 40 °C, repasse à l'état bas en
dessous de 20 °C, actif à l'état bas

    // Initialisation du module LCD
    lcd_init(16);
    lcd_clear();

    // Autorisation des interruptions
    //SREG.7 = true;
}

int MesureTension_totale(void)
{
    ADMUX=0xE6;
    ADCSRA.6 = 1; //Lancement de la conversion
    while(ADCSRA.6); //Attente de la fin de conversion
    return ((int)ADCH)/0.139*1.953;
}

// Mesure, calcul et retourne la tension d'une cellule
int MesureTension_cV(int NumCellule) //NumCellule = [0;7]
{
    if(NumCellule<=1)
    {
        ADMUX=0xE0|NumCellule+6; //Paramètrage de l'entrée analogique à mesurer
        ADCSRA.6 = 1; //Lancement de la conversion
        while(ADCSRA.6); //Attente de la fin de conversion
    }
    else if(NumCellule>1)
    {
        ADMUX=0xE0|NumCellule-2; //Paramètrage de l'entrée analogique à mesurer
        ADCSRA.6 = 1; //Lancement de la conversion
        while(ADCSRA.6); //Attente de la fin de conversion
    }
    return ((int)ADCH*(8-NumCellule))*1.953/256.0*TabTensMax[NumCellule]; //On retourne la valeur en
10^(-2) volts (centivolts) de l'entrée
}

// Affiche "l'arrière-plan" de la sélection TensionCellule
void AffichageBaseTensionCellule(void)
{
    lcd_clear();
    lcd_putsf("1: -.-V  2: -.-V");
    lcd_gotoxy(0,1);
    lcd_putsf("3: -.-V  4: -.-V");
    lcd_gotoxy(0,2);
    lcd_putsf("5: -.-V  6: -.-V");
    lcd_gotoxy(0,3);
    lcd_putsf("7: -.-V  8: -.-V");
}

// Actualise les tensions pour la sélection TensionCellule
void AffichageTensionCellule(int *Tab)
{
    char ValTxt[6];
    signed char i;
    char j = 0;

    // Traitement du tableau
    for(i=6 ; i>=0 ; i--)
    {
        if(Tab[i] != 0) //Si la tension est supérieur à 0 (sinon cela signifie qu'il y a un problème
avec la connexion)
        {
            for(j=7 ; j>i ; j--)
            {
                Tab[i] -= Tab[j];
```

```
            }
        }
    }

    // Actualisation de l'écran LCD
    j = 0;
    for(i=0 ; i<4 ; i++)
    {
        if(Tab[j] != TabTens[j]) //On actualise l'affichage si la valeur change
        {
            TabTens[j] = Tab[j];
            lcd_gotoxy(2,i);
            ftoa(TabTens[j]/100.0, 2, ValTxt);
            lcd_puts(ValTxt);
        }
        j++;
        if(Tab[j] != TabTens[j]) //On actualise l'affichage si la valeur change
        {
            TabTens[j] = Tab[j];
            lcd_gotoxy(11,i);
            ftoa(TabTens[j]/100.0, 2, ValTxt);
            lcd_puts(ValTxt);
        }
        j++;
    }
}

// Affiche "l'arrière-plan" de la sélection Générale
void AffichageBaseGeneral(void)
{
    lcd_clear();
    lcd_putsf("Temperature :");
    lcd_gotoxy(3,1);
    lcd_putsf("-.- ßC");
    lcd_gotoxy(0,2);
    lcd_putsf("Tension batt :");
    lcd_gotoxy(3,3);
    lcd_putsf("-.-- V");
}

// Actualise les informations pour la sélection Générale
void AffichageGeneral()
{
    static int old_Temp;
    static int old_Tens;
    char ValTxt[6];

    if(old_Temp != TempBatt)
    {
        old_Temp = TempBatt;
        lcd_gotoxy(3,1);
        ftoa(TempBatt/10.0, 1, ValTxt);
        lcd_puts(ValTxt);
        lcd_putsf(" ßC ");
    }
    if(old_Tens != TensBatt)
    {
        old_Tens = TensBatt;
        lcd_gotoxy(3,3);
        ftoa(TensBatt/100.0, 2, ValTxt);
        lcd_puts(ValTxt);
        lcd_putsf(" V ");
    }
}

// Gère la surtension de la batterie
void AffSurtension(char actif)
{
    static char old;
    if(old != actif)
    {
        old = actif;
        lcd_gotoxy(0,2);
        if(actif)
        {
            lcd_putsf("Surtension !  ");
        }
```

```
        else
        {
            lcd_putsf("Tension batt : ");
        }
    }
}

// Gère le niveau de batterie faible
void AffNiveauFaible(char actif)
{
    static char old;
    if(old != actif)
    {
        old = actif;
        lcd_gotoxy(0,2);
        if(actif)
        {
            lcd_putsf("Batterie faible");
        }
        else
        {
            lcd_putsf("Tension batt : ");
        }
    }
}

// Gère les surchauffes de la batterie
void AffSurchauffe(char actif)
{
    static char old;
    if(old != actif)
    {
        old = actif;
        lcd_gotoxy(0,0);
        if(actif)
        {
            lcd_putsf("Surchauffe ! ");
        }
        else
        {
            lcd_putsf("Temperature :");
        }
    }
}

// Fonction principale
void main(void)
{
    char Menu = 0;
    char flag = 1;
    int Tab[8];
    int i;

    Initialisation();

    lcd_putsf("   Projet BMS\n      Par\nLebrun Alexandre\n   Rey Kevin");
    while(PINB.4);

    AffichageBaseGeneral();

    while (1)
    {
        // Gestion du changement de menu
        if(!PINB.4 && !flag) //Si on appuie sur le bouton
        {
            Menu = !Menu;
            if(Menu)
            {
                for(i=0 ; i<8 ; i++)
                {
                    TabTens[i] = 0;
                }
                AffichageBaseTensionCellule();
                Alert[0]=1;
                Alert[1]=1;
                Alert[2]=1;
            }
```

```
            else
            {
                TempBatt = 0;
                TensBatt = 0;
                AffichageBaseGeneral();
            }
            flag = 1;
        }
        else if(PINB.4 && flag)
        {
            flag = 0;
        }

        // Actualisation des valeurs
        TempBatt = lm75_temperature_10(0); //Mesure de la température
        TensBatt = MesureTension_totale(); //Mesure de la tension totale
        if(Menu)
        {
            for(i=0 ; i<8 ; i++)
            {
                Tab[i] = MesureTension_cV(i);
            }
            AffichageTensionCellule(Tab);
        }
        else
        {
            AffichageGeneral();
            AffSurtension(!Alert[0]);
            AffNiveauFaible(!Alert[1]);
            AffSurchauffe(!Alert[2]);
        }

        // Gestion des alertes et des sorties correspondantes
            //Surtension
        if(TensBatt >= TENS_MAX)
        {
            PORT_TENS_MAX = 0;
            Alert[0]=0;
        }
        else if(PIN_TENS_MAX == 0)
        {
            PORT_TENS_MAX = 1;
            Alert[0]=1;
        }
            //Niveau de batterie faible, demande de charge
        if(TensBatt <= TENS_MIN)
        {
            PORT_TENS_MIN = 0;
            Alert[1]=0;
        }
        else if(PIN_TENS_MIN == 0)
        {
            PORT_TENS_MIN = 1;
            Alert[1]=1;
        }
            //Surchauffe
        if(TempBatt >= TEMP_MAX)
        {
            PORT_TEMP_MAX = 0;
            Alert[2]=0;
        }
        else if(PIN_TEMP_MAX == 0)
        {
            PORT_TEMP_MAX = 1;
            Alert[2]=1;
        }
        delay_ms(500);
    }
}
```