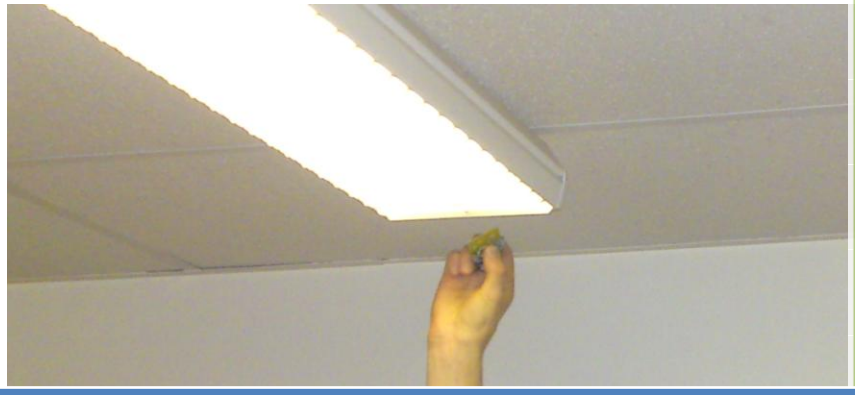
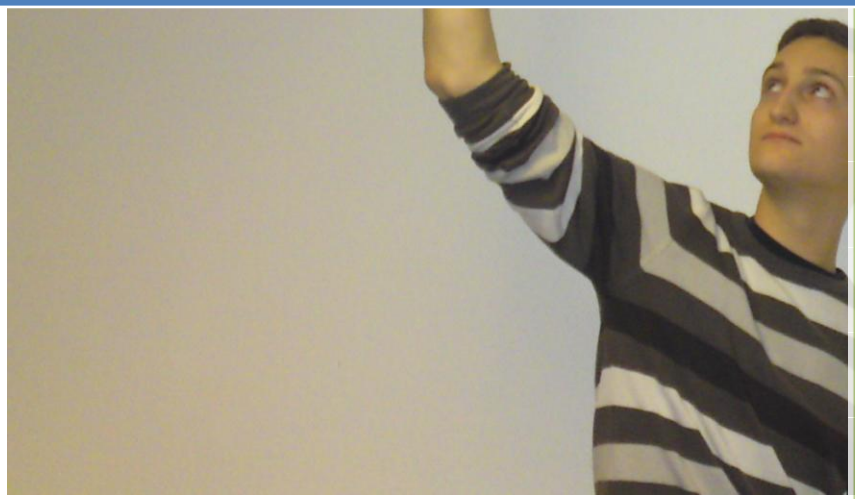
2010



LE MODULE LUMIERE



Nicolas SALSE – Maxime LEFEBVRE

IUT de Ville d'Avray

11/04/2010

Sommaire

I. Présentation et objectifs

1. Objectifs

Le but de ce projet est de fabriquer une carte d'acquisition de lumière. Cette dernière devra permettre, après avoir été placée à proximité d'une source de lumière, de relever à intervalles réguliers les données concernant l'évolution de la luminosité dans une salle sur plusieurs mois. On cherche ainsi, en récupérant sur un ordinateur les informations enregistrées, à déterminer si il existe ou non un quelconque défaut de surconsommation.

2. Enjeux

L'enjeu principal ici est l'économie d'énergie, non seulement dans l'environnement étudié, mais aussi pour la carte d'acquisition. En effet, pour pouvoir nous donner des informations précises pendant plusieurs mois, notre maquette devra consommer le minimum d'énergie possible et n'effectuer que les taches nécessaires au bon déroulement de l'étude.

3. Principe de fonctionnement

Un composant photosensible capte la lumière ambiante et renvoie une tension liée à l'intensité lumineuse sur un microcontrôleur. Après conversion analogique/numérique, l'µC est chargé de stocker toutes les informations relatives à l'acquisition dans une mémoire externe, la récupération des données se faisant à l'appui sur un bouton. Le système est soumis à une tension de 3V fournie par une pile. La communication avec l'ordinateur se fera via le port série ou un port USB.

II. Choix du matériel

Dans cette partie, nous nous attacherons à expliquer pourquoi nous avons choisi les composants que nous avons utilisés pour notre maquette finale.

1. Choix de la cellule photosensible

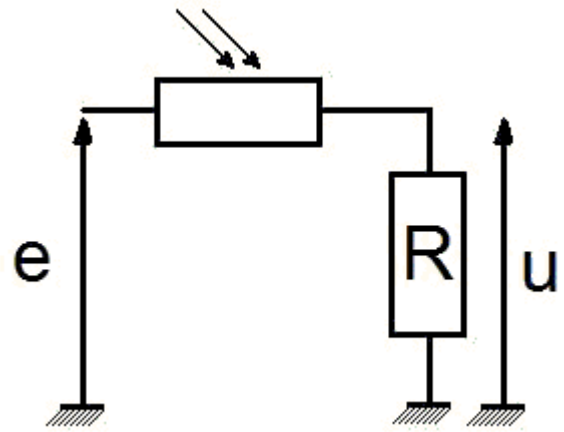
Dans cette étude, nous avons choisi d'axer nos recherches sur deux composants: la photorésistance et le phototransistor. Nous souhaitons donc évaluer leur comportement en

fonction de la quantité de lumière qu'ils reçoivent. Pour cela, nous avons utilisé un luxmètre afin de mesurer l'intensité lumineuse d'une lampe, et avons monté différentes plaquettes de test qui nous ont permis d'obtenir les résultats suivants.

a) Photorésistance

Principe: sa valeur ohmique diminue en même temps que la luminosité augmente.

Schéma de test:

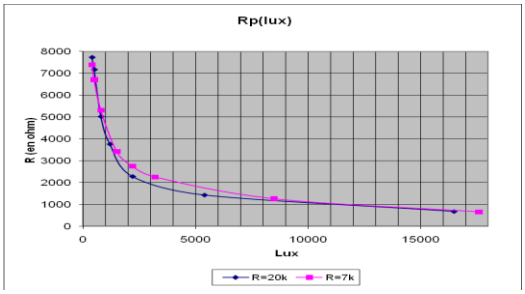


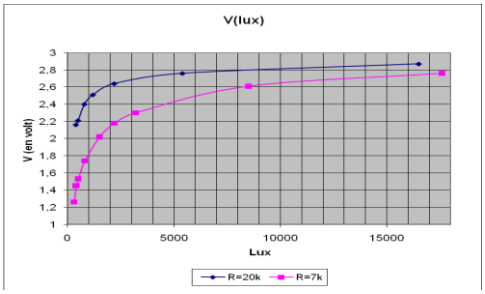
Nous relevons dans le tableau suivant les valeurs de tension en sortie afin de caractériser le comportement de notre composant.

Photoristance 19m51 : caractéristique **Lampe loupe LL622TA**

R=20k					
D(sol)	**D(réelle)**	**Lux**	**V(lux)**	**Rp(theo)**	**Rp(lux)**
7	2,5	16500	2,87	905,92	670
14	9,5	5400	2,76	1739,13	1420
23,5	19	2200	2,64	2727,27	2270
35,5	31	1200	2,51	3904,38	3750
45	40,5	800	2,4	5000,00	5010
59	54,5	500	2,21	7149,32	7160
67	62,5	400	2,16	7777,78	7720
		0	0,22		

R=7k					
D(sol)	D(réelle)	Lux	V(lux)	Rp(theo)	Rp(lux)
7	2,5	17600	2,76	608,70	650
11	6,5	8500	2,61	1045,98	1250
19	14,5	3200	2,3	2130,43	2250
24	19,5	2200	2,18	2633,03	2740
32	27,5	1500	2,02	3396,04	3420
46	41,5	800	1,74	5068,97	5300
57	52,5	500	1,53	6725,49	6700
61	56,5	400	1,45	7482,76	7380
304	299,5	300	1,26	9666,67	
		0	0,08		



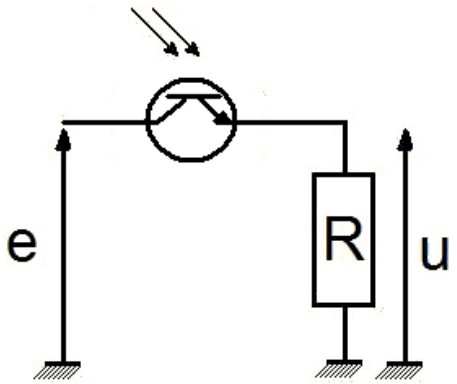


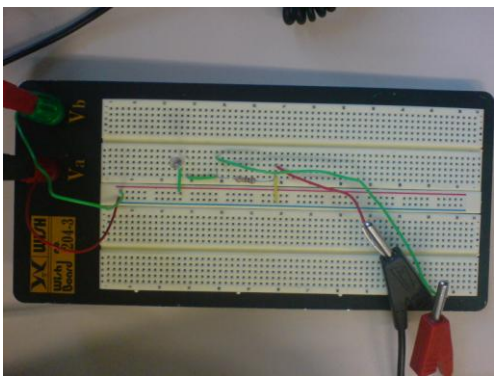
Sur nos 2 tests, nous constatons que la caractéristique de la photorésistance est bien définie. De plus, on remarque que le fait de prendre une résistance plus grande pour le pont diviseur fait saturer plus rapidement la tension de sortie.

b) Phototransistor

Principe: son courant de sortie au en même temps que la luminosité.

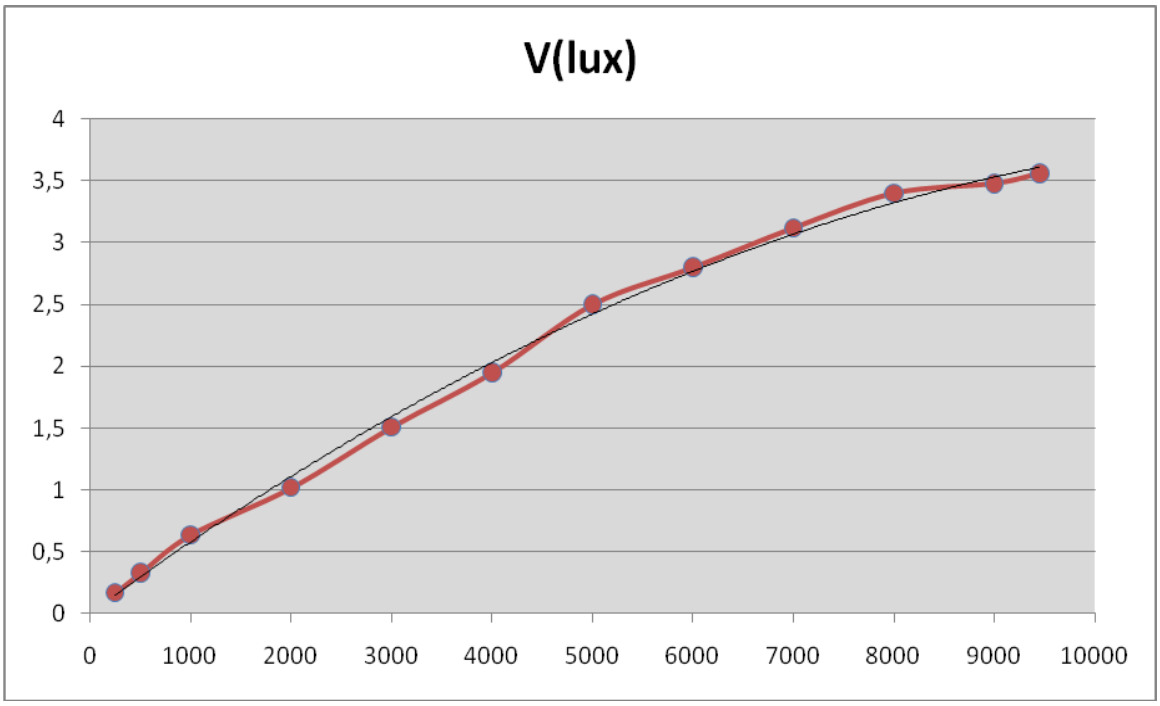
Schéma de test:





Caractéristiques photodiode BPW96B

R = 1kHz		
Lux	**V(lux)**	**i en A**
9450	3,56	3,56E-03
9000	3,48	3,48E-03
8000	3,4	3,40E-03
7000	3,12	3,12E-03
6000	2,8	2,80E-03
5000	2,5	2,50E-03
4000	1,95	1,95E-03
3000	1,51	1,51E-03
2000	1,02	1,02E-03
1000	0,64	6,40E-04
500	0,33	3,30E-04
250	0,17	1,70E-04



Nous décidons donc ici de choisir le phototransistor comme capteur de lumière. En effet, nous avons choisi dans notre cahier des charges e nous baser sur des seuils lumineux afin de déterminer l'état de la source lumineuse. Ainsi, on peut par exemple choisir de considérer la source allumée si le niveau lumineux dépasse 3000 lux. Or la plage d'évolution de la tension de sortie avant saturation est plus grande avec le phototransistor. Les seuils seront donc d'autant plus faciles à définir. Nous prenons en complément une résistance de 1k.

Nous mettons donc en place le schéma précédent dans la partie analogique de notre maquette.

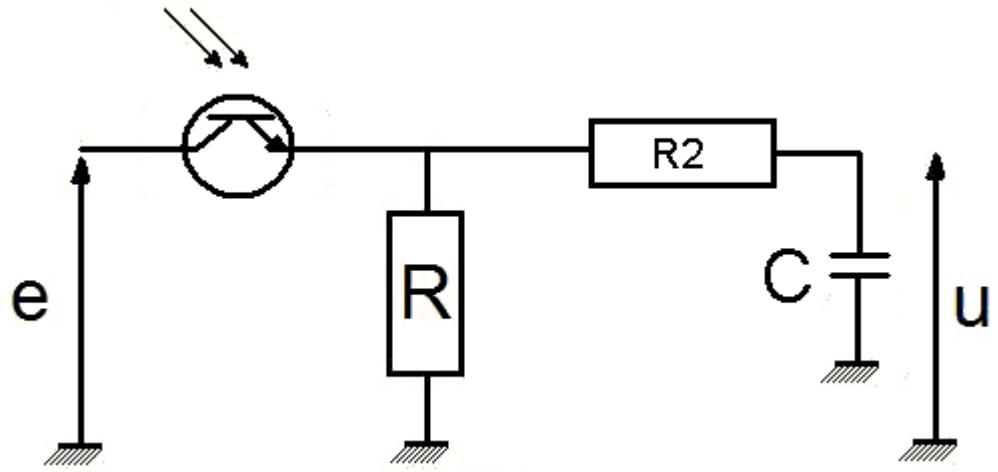
2. Filtre du 1er ordre

Après avoir testé notre système capteur résistance, nous nous somme aperçu que notre signal était perturbé par un signal sinusoïdal de fréquence 100Hz qui est lié à la fréquence lumineuse du courant de secteur. Il nous fallait donc mettre en place un filtre de 1er ordre dans le but de « couper » toutes fréquences supérieures ou égales à 100Hz.

Pour être large, nous décidons donc de mettre en place un filtre de fréquence propre de 50 Hz environ, soit une résistance de 180k et une capacité de 100nF en série. En effet,

$R*C = 180k * 100n = 0{,}018 \rightarrow \omega_0 = 1 / (R*C) = 55Hz$.

Soit le schéma suivant :



La tension analogique u doit maintenant être convertie en numérique afin de pouvoir traiter les informations reçues à l'aide d'un microcontrôleur.

3. Microcontrôleur et mémoire externe

Cahier des charges de l'µC :

- CAN
- liaison série
- protocole SPI
- alimentation en 3,3V

L'ATmega16 fourni de base sur les kits STK500 nous offre ces caractéristiques, mais ont beaucoup d'autres fonctionnalités qui ne nous sont pas utiles. Nous choisissons donc de prendre un

microcontrôleur de la même famille mais de gamme moins élevée. Dans notre cas, l'ATmega168P convient très bien.

La datasheet est fourni en annexe dans les répertoires du projet.

Cependant, la mémoire interne du microcontrôleur n'est pas assez grande pour pouvoir stocker des données de façon continue pendant plusieurs mois. Il nous faut donc une mémoire externe proportionnelle à la taille des données que l'on souhaite y stocker. Le format des données retenu est le suivant :

« Date », « heure »
Lux = « valeur numérique de luminosité »

Après de rapides calculs, nous avons déterminé qu'il nous fallait une mémoire d'un minimum de 3Mo pour pouvoir stocker nos données pendant un an avec acquisition toute les 2 minutes. Nous choisissons donc de prendre pour mémoire externe l'ATDF25. Cette mémoire nous permet de gérer la liaison SPI et de stocker toutes les données nécessaires.

Une fois de plus, la datasheet est fourni dans les répertoires du projet.

Après avoir choisi nos composants, nous avons fait fabriquer une carte de test alimentée par le kit AVR et qui nous permettait de tester notre partie analogique, mais aussi la communication avec la mémoire.

III. Problèmes liés au matériel choisi

1. Partie analogique

La résistance liée au phototransistor n'est peut être pas assez élevée pour fournir une tension de sortie suffisante aux mesures. Nous avons donc dû doubler sa valeur dans le but de pouvoir avoir des résultats cohérents.

2. Partie numérique

Nous n'avons aucun moyen de savoir si la communication avec la mémoire est possible. En effet, nous pouvons envoyer les commandes d'enregistrement et de récupération de données mais il nous est impossible de savoir si telle ou telle commande fonctionne ou pas. Nous n'avons à l'heure actuelle pas encore trouvé de solution à ce problème.

Ci-dessous, nous fournissons un exemple de trame que nous avons envoyé à la mémoire. Celle-ci prouve que l'µC envoie quelque chose mais ne prouve en aucun cas que la mémoire l'interprète correctement. Données sur 8 bits.

3. Programme et portage

Nos principales fonctions étant tirées du projet ventilateur réalisé en début de deuxième année sur ATmega16, nous devions effectuer un portage de nos programmes sur ATmega168P.

D'une part, concernant le programme en lui-même, nous avions à mettre à jour toutes les variables liées au registre dans les fonctions concernées. En effet, le portage d'un programme entre

deux microcontrôleurs de la même famille n'est pas aussi simple. Il faut faire très attention à n'utiliser que des bits qui sont reconnus par le compilateur et donc par l'ATmega168P.

D'autre part, nous avons eu des problèmes concernant la version d'AVR Studio utilisée. Cette dernière ne possédait pas les drivers nécessaires à l'exécution de notre programme. Nous avons donc du installer la dernière version d'AVR Studio mais aussi les drivers du 168P sur notre ordinateur.

4. Matériel

Le seul problème matériel que nous ayons rencontré, est à mettre en relation avec des problèmes de communication entre le microcontrôleur et l'ordinateur. Il semblerait que les cartes de liaison série soient défectueuses ce qui nous a obligé à changer d'ordinateur en cours de projet et donc de réinstaller toutes les mises à jour citées précédemment.

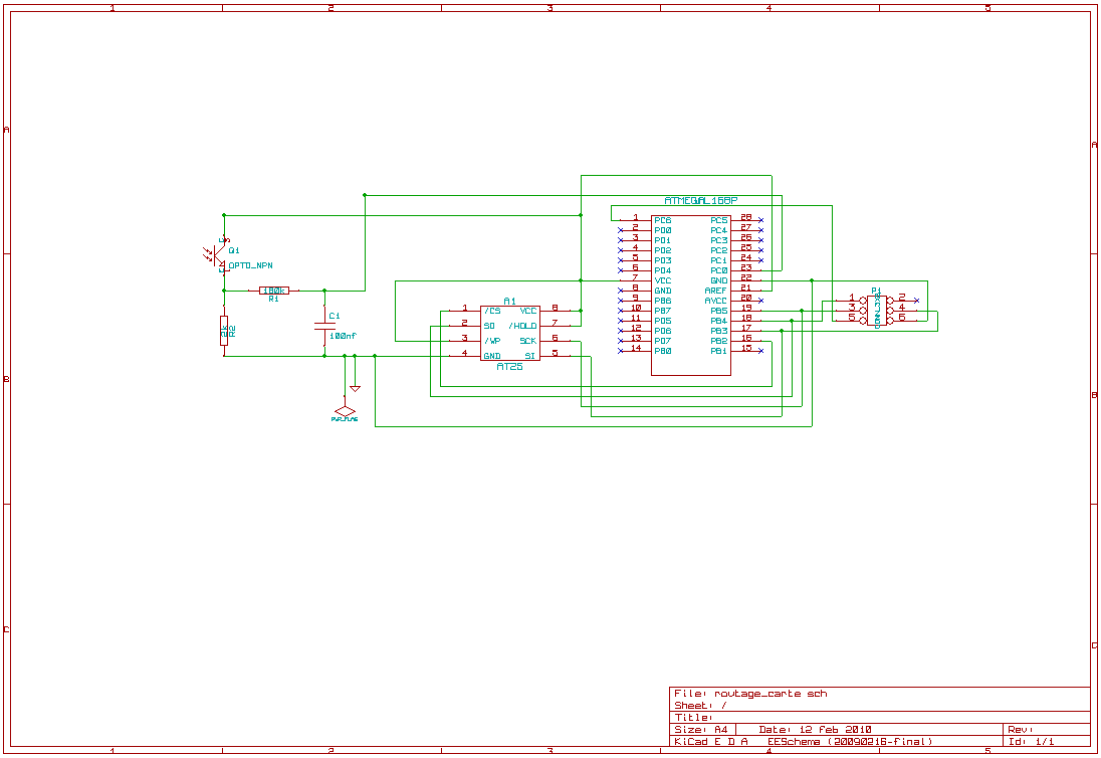
De plus, sur notre carte « finale » il nous était impossible de programmer le microcontrôleur in situ. Nous avons donc implémenté le programme sur le kit AVR puis nous avons mis notre µC sur la carte.

IV. Principe de fonctionnement

1. Carte de test sans la mémoire externe

Sachant que nous n'arrivions pas à communiquer avec la mémoire, nous avons décidé de fabriquer une carte complète mais d'écrire un programme tel que nous ne toucherons pas à la mémoire externe. Pour cela, nous avons créé nos composants puis nous avons routé la carte sous KiCad. On fournit ci-après la version schématique de notre maquette actuelle.

Schéma:



2. Théorie

De manière théorique, notre modèle doit réaliser les étapes suivantes :

tant que (1)

| se réveiller toute les minutes

| acquérir analogiquement la valeur de l'intensité lumineuse

| convertir la tension analogique en numérique

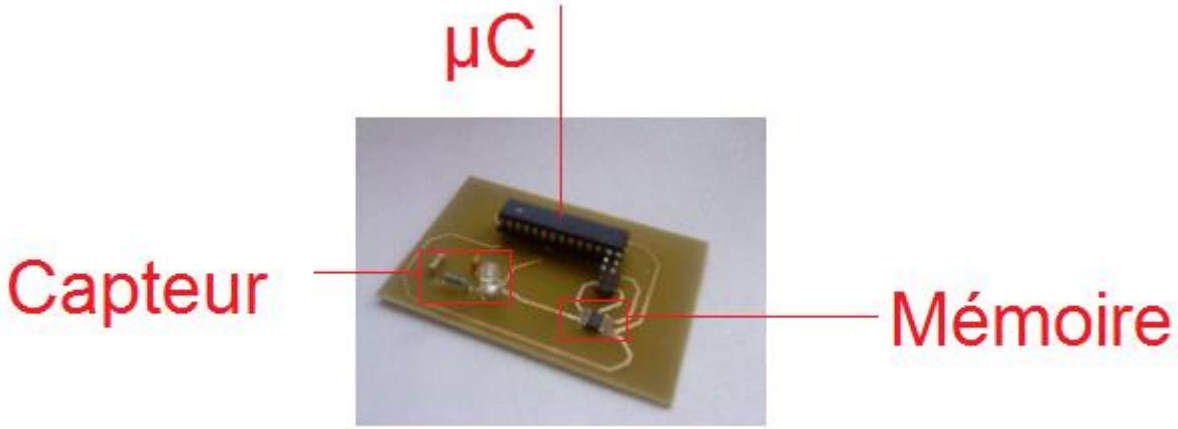
| envoyer les données au bon format à la mémoire

| se mettre en veille

A l'appui sur un bouton, la mémoire doit nous renvoyer les valeurs enregistrées.

3. Pratique

Au final, nous avons pu faire fabriquer une carte qui nous permet d'enregistrer 25 valeurs puis de nous renvoyer sur l'hyperterminal ces valeurs enregistrées dans la mémoire interne du microcontrôleur.



Nous avons aussi rajouter en dernière minute 3 pins qui nous permettent de vérifier le bon fonctionnement de notre carte : 2 pins à brancher sur les port LED qui indique l'état (émission ou enregistrement) du programme et une pin à brancher sur le TxD du kit AVR afin de lire les données renvoyées sur l'hyperterminal.

En fait, suite aux différents problèmes que nous avons rencontrés, nous n'avons pas pu traiter le fonctionnement de notre circuit-imprimé avec la mémoire. Nous fournissons donc le code du programme de test de la maquette

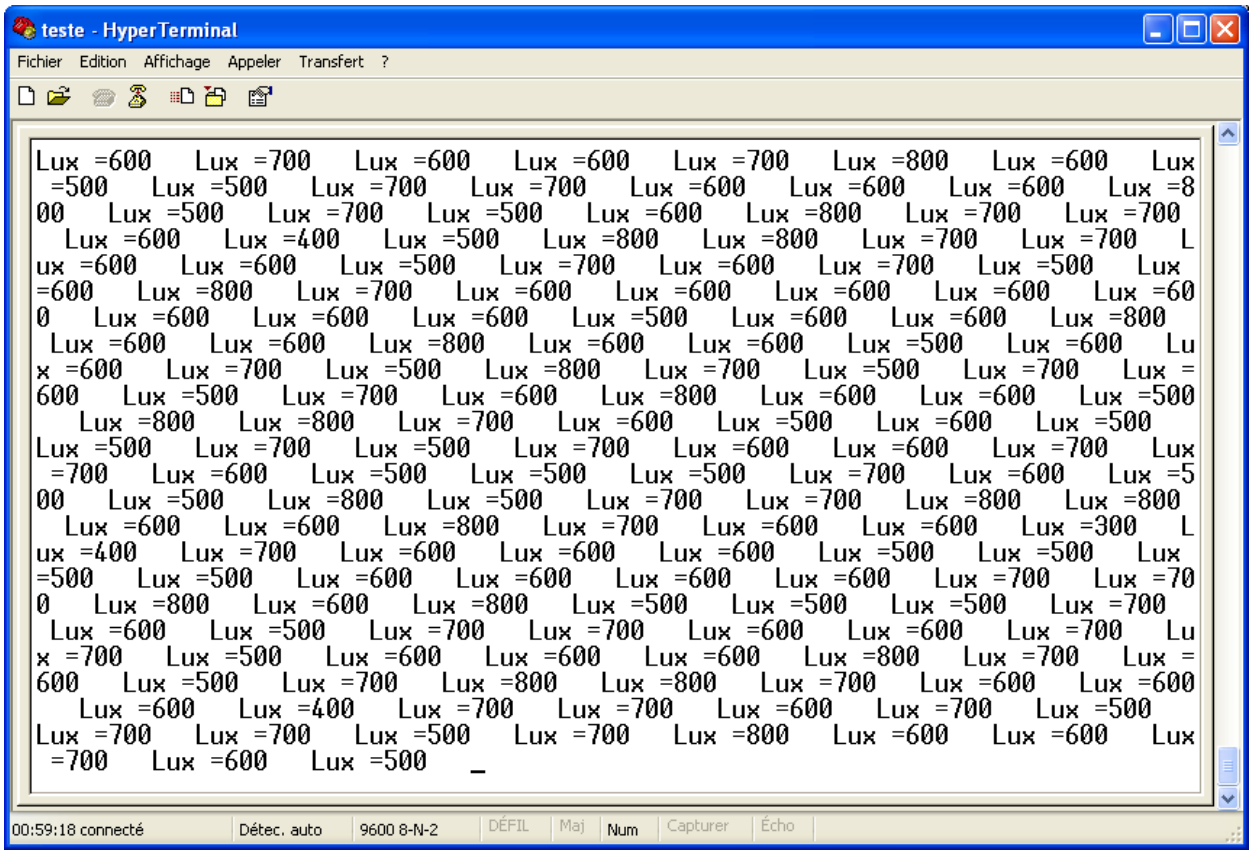
```
while(1){

    //programme de test de la liaison série
    PORTD=(1<<PORTD3)&(1<<PORTD2);

    for(i=0;i<25;i++){ //25 enregistrements
        PORTD=PORTD|(1<<PORTD2);;
        for(j=0;j<=5;j++){
            _delay_ms(100);
        }
        conversion=convertir()*100;
        itoa(conversion, char_conv[i], 10);
        PORTD=PORTD&~(1<<PORTD2);
        for(j=0;j<=5;j++){
            _delay_ms(100);
        }
    }
```

```
    for(i=0;i<25;i++){ //puis 25 émission vers hyperterminal
        PORTD=PORTD|(1<<PORTD3);
        for(j=0;j<=5;j++){
            _delay_ms(1);
        }
        emettre_chaine("Lux =");
        emettre_chaine(char_conv[i]);
        emettre_chaine("    ");
        PORTD=PORTD&~(1<<PORTD3);
        for(j=0;j<=5;j++){
            _delay_ms(100);
        }
    }
}
```

On peut alors vérifier sur l'hyperterminal si notre programme fonctionne en vérifiant qu'il renvoi des valeurs cohérentes avec l'activité autour du capteur, c'est-à-dire que la valeur affichée augmente bien quand la luminosité augmente. On obtient alors un écran comme celui-ci :



Notre programme fonctionne donc correctement en utilisant la mémoire interne du microcontrôleur. Néanmoins, il reste plusieurs étapes à aborder pour terminer le projet.

V. Ce qu'il reste à faire

Malgré l'avancement du projet, il reste beaucoup de chose à faire. Dans un premier temps, nous devons commencer à nous pencher sur la communication avec la mémoire externe. En effet, il faut pouvoir s'assurer que le dialogue s'effectue bien entre le maître et l'esclave, c'est-à-dire reprendre le programme de test et remplacer l'enregistrement dans la mémoire interne par un enregistrement dans la mémoire externe.

Deuxièmement, il faut s'intéresser à la gestion des modes de veille. Pour cela, il faudra d'abord penser à alimenter le circuit par une pile de 3,3V. Ensuite nous devrons savoir à quel moment et comment interrompre le processus afin de mettre le microcontrôleur en sommeil. Ainsi on le réveillera à intervalles réguliers pour effectuer les mesures et les enregistrer. Cette gestion est importante pour que la consommation en énergie de la maquette soit réduite et que son autonomie soit élevée.

Troisièmement, il faudrait pouvoir lier notre projet au projet USB sur microcontrôleur, qui a pour but d'émuler le protocole USB sur un ATmega16. On pourrait alors créer une interface graphique nous permettrait d'afficher les résultats de nos mesures sous forme de graphique ou de tableaux. Par exemple, on pourrait demander d'afficher les valeurs de luminosité entre deux dates précises.

Pour finir il faudrait miniaturiser le projet afin de pouvoir en faire un réel module fonctionnel et fiable. Ainsi il serait plus facile de le laisser plusieurs mois dans un endroit sans craindre sa détérioration.