Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Domaine Sciences de l'ingénieur
Rte du Rawyl 47
CH- 1950 Sion 2
Tél. +41 27 606 85 11
Fax +41 27 606 85 75
info@hevs.ch

www.hevs.ch

Filière Systèmes industriels

Orientation Power & Control

Diplôme 2010

Johan Walthert

Gestion des stores électriques en fonction de la lumière et de la présence

Professeur Fariba Bützberger

Expert André Rotzetta

Sion, le 12 juillet 2010

Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Gestion des stores électriques en fonction de la lumière et de la présence

Diplômant/e Johan Walthert

Objectif du projet

Le but du projet est d'optimiser la consommation d'énergie de chauffage ou de climatisation en profitant au mieux des apports naturels par un pilotage intelligent des stores électriques.

Méthodes | Expériences | Résultats

Un système de gestion des stores électriques d'une pièce habitable en fonction du rayonnement solaire et de la présence dans la pièce a été conçu et réalisé.

Le système proposé est composé de deux modules électroniques émettant par Radio Frequency les signaux provenant des capteurs de présence à l'intérieur et de l'ensoleillement à l'extérieur, ainsi que d'un module de réception des signaux RF agissant directement sur le moteur électrique des stores. Un dernier module avec affichage LCD et clavier joue le rôle d'interface IHM permettant aux utilisateurs d'introduire leurs besoins et souhaits.

3 modes de fonctionnement sont programmés : Manuel, Astro et Save Energy.

Le système est exploitable toute l'année. Le but étant de limiter l'apport de chaleur du soleil en été et de la favoriser en hiver.

Le prototype développé a été testé et validé au niveau du fonctionnement des différents modes ainsi que de la consommation électrique. Selon les résultats des travaux précédents, le système permet de diminuer la consommation d'énergie. Mais aucunes mesures n'étant réalisées, par manque de temps, le gain ne peut pas être chiffré pour notre cas.

Travail de diplôme | édition 2010 |

Filière
Systèmes industriels

Domaine d'application
Power and control

Professeur responsable
Dr Fariba Bützberger
mof@hevs.ch

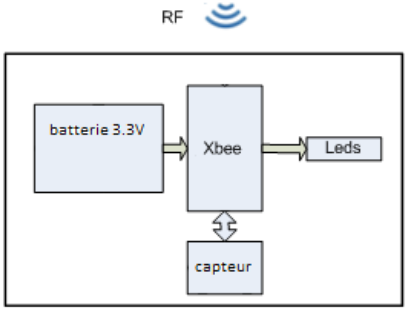


Schéma bloc du module émetteur à capteur (capteur de présence pour le module intérieur et capteur de rayonnement pour le module extérieur).

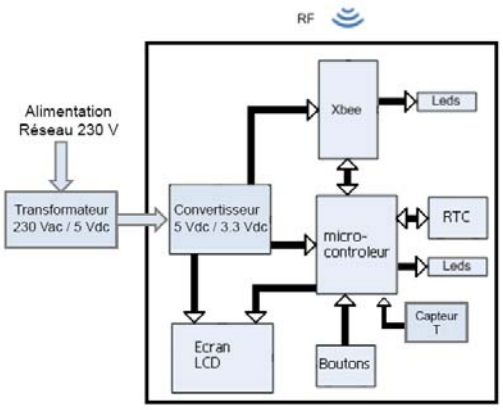


Schéma bloc du module central avec interface IHM.

HES-SO Valais
Route du Rawyl 47
1950 Sion

Tél. 027 606 85 11
URL www.hevs.ch

Sommaire

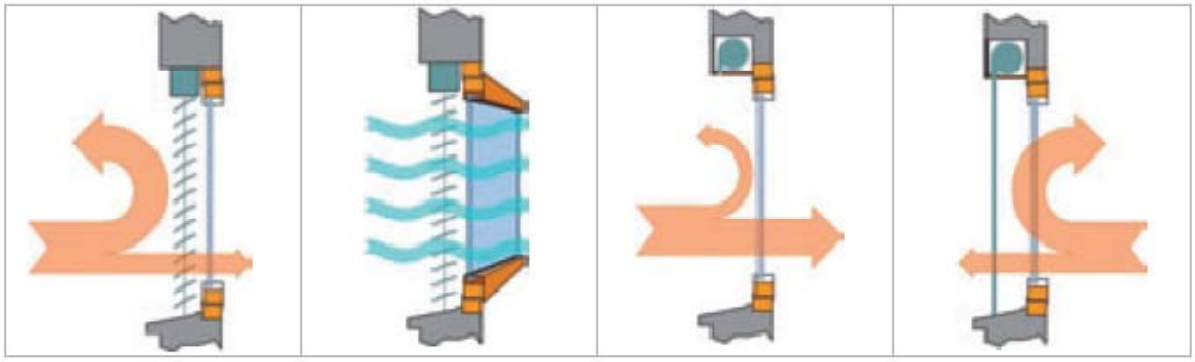
1 Description générale 3
1.1 Présentation du projet 3
1.2 But du projet 4
1.3 Stratégie de commande des stores 5
2 Communication RF 6
2.1 Caractéristiques générales des modules 6
2.1.1 Particularité 7
2.2 Communication Xbee 8
2.3 Trames API 10
2.4 Tests et programmations des modules 11
2.5 Paramétrage à l'aide de X-CTU 11
2.5.1 Tests de communication 13
3 Programmation du microcontrôleur 15
3.1 Oscillateur interne 16
3.1.1 Registre OSCCON, oscillator control register 17
3.2 Port d'entrées, sorties 18
3.2.1 Port A 18
3.2.2 Port B 18
3.2.3 Port C 18
3.3 Communication SPI 19
3.4 Communication UART 20
3.4.1 Registre TXSTA, Transmit Status and Control Register 21
3.4.2 Registre RCSTA, Receive Status and Control Register 22
3.4.3 Registre BAUDCON, Baud rate Control Register 23
3.4.4 Exemple de définition du Baud Rate 24
3.4.5 Erreur de Baud Rate 25
3.5 Communication I2C 26
3.5.1 Registre SSPCON1, MSSP Control Register (I2C mode) 27
3.5.2 Registre SSPCON2, MSSP Control Register (I2C mode) 28
3.6 Interruption 29
3.6.1 Registre INTCON, Interrupt Control Register 30

3.6.2 Registre PIE1, Periferal Interrupt Enable Register 1 31
3.6.3 Evénements 32
4 Design et fonctionnalités des cartes 33
4.1 Carte Commande moteur 33
4.2 Cartes émettrices 35
4.2.1 Carte présence 36
4.2.2 Carte rayonnement 37
4.3 Carte Command principale 38
5 Consommation énergétique et autonomie 39
6 Logique de commande et de gestion 40
6.1 Mode manuel 40
6.2 Mode automatique Astro 40
6.3 Mode automatique Save Energy 41
7 Affichage, menu et paramètres 42
8 Tests et résultats 42
9 Améliorations 42
10 Conclusions 43
11 Remerciements 43
12 Bibliographie 44
13 Annexes 44

1 Description générale

1.1 Présentation du projet

L'utilisation et la construction de bâtiment utilisent aujourd'hui environ 40% de l'énergie produite. Une installation de protection solaire automatisée selon des critères énergétiques peut réduire, selon l'orientation d'une pièce, la consommation d'énergie de 10%.



1 2 3 4

Figure 1 - Illustrations du rayonnement solaire

1. Durant la belle saison un store baissé permet de se protéger contre la surchauffe de la pièce. Ce qui permet de diminuer l'utilisation d'un système de climatisation.
2. L'ouverture du store durant les nuits estivales consent à une éventuelle aération naturelle ce qui permet un refroidissement de la pièce.
3. Le rayonnement solaire à disposition l'hiver permet de chauffer naturellement la pièce exposée.
4. Les nuits hivernales, un store baissé permet de mieux conserver la chaleur à l'intérieur.

Un store intelligent rend donc dynamique le rayonnement solaire ce qui diminue l'utilisation de moyens annexes pour le chauffage et la climatisation. De plus, la lumière du jour est utilisée au maximum réduisant la consommation des éclairages artificiels.

1.2 But du projet

L'utilisation de capteurs de rayonnement et de présence permet de développer l'intelligence d'un store tout en permettant aux utilisateurs de la pièce de commander les stores selon leurs besoins. La gestion se fait donc pièce par pièce.

Le but du projet est donc de concevoir et de réaliser un système intelligent de gestion des stores électriques d'une pièce habitable en fonction du rayonnement solaire et d'une présence dans ladite pièce. Deux cartes électroniques, Es et Ep, de faible consommation doivent transmettre les signaux provenant des capteurs à une carte de commande (C) qui commandera les stores via des relais sur la carte de commande moteur (R). La communication entre les cartes sera réalisée à l'aide de signaux Radio Frequency(RF). L'utilisateur pourra toutefois régler les stores selon ses envies à tout moment à l'aide l'interface homme machine présente sur la commande principale.

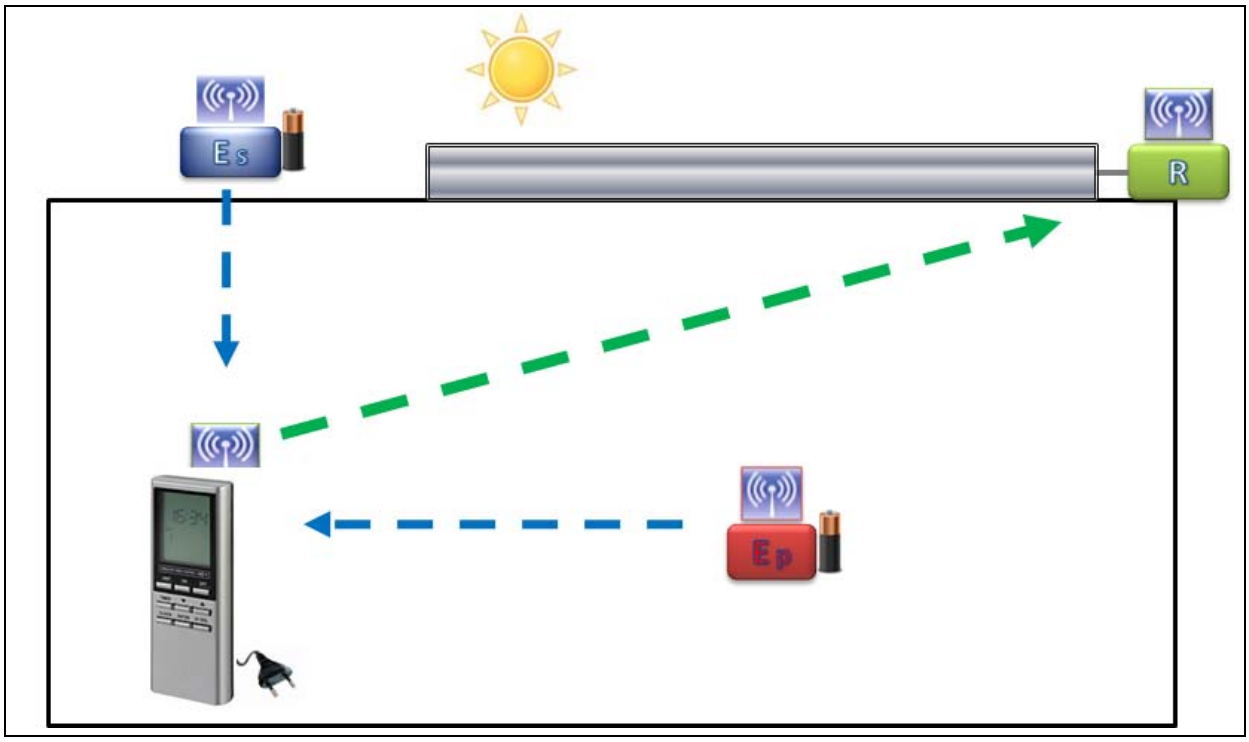


Figure 2 - schéma de principe du projet

Ce système, exploitable toute l'année, peut s'appliquer aussi bien aux maisons bien protégées thermiquement qu'aux vieux bâtiments. Les coûts d'implantation et d'exploitation n'engendrent que peu de frais étant donné que la communication est sans fil.

Ce projet est une suite d'un projet financé par la réserve stratégique de la HES-SO, OpenBat ou optimisation énergétique du bâtiment. Le projet store intelligent est lui financé par l'OFEN (office fédérale de l'énergie) pour 2009 et 2010.

1.3 Stratégie de commande des stores

En cas d'absence dans la pièce, les stores doivent donc être commandés de manière à optimiser la consommation énergétique. Il existe deux modes de fonctionnement automatique. Le mode save énergy qui ajuste les stores en fonction de la saison et du rayonnement. La saison est définie à l'aide d'une horloge temps réel sur la commande principale et le rayonnement grâce à une cellule solaire. Et le mode Astro qui agit selon l'heure et la date ainsi que d'un horaire de référence fixe par mois.

Pour une question de confort, l'utilisateur ne doit pas être dérangé par des mouvements automatiques des stores. C'est pourquoi le système passe en mode manuel durant une présence. Sa présence est déterminée à l'aide d'un capteur de mouvement ou une action sur la commande principale.

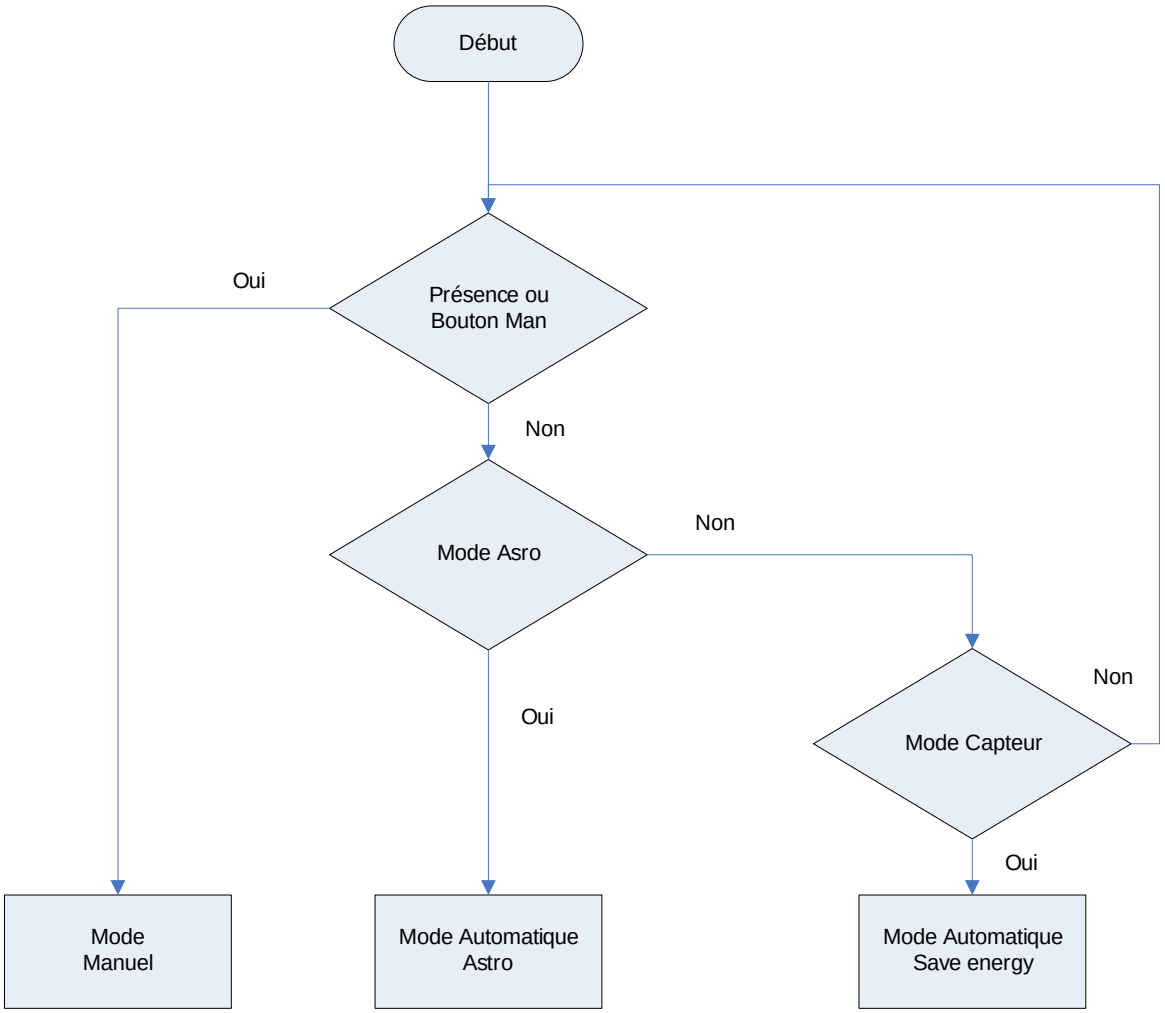


Figure 3 - Structogramme général de l'application

2 Communication RF

On utilise les modules "Xbee séries 2 OEM RF" sont des transceivers radio spécialement conçus pour la réalisation de systèmes de communication au sein de réseaux de capteurs sans fil utilisant le protocole Zigbee. De petites dimensions, ces modules se distinguent par leur grande simplicité d'utilisation, leur faible consommation en mode veille et leur coût très compétitif.

Figure 4 - module Xbee séries 2 OEM RF

2.1 Caractéristiques générales des modules

Les modules XBee présentent une puissance de transmission RF de 2 mW. La portée intérieure peut atteindre au maximum 40 mètres suivant la nature des obstacles. Le débit RF est de 250 Kbps alors que le débit à l'interface est au maximum de 230,4 Kbps.

Leurs caractéristiques avancées en matière de gestion de réseaux leurs permettent de communiquer en broadcast, multipoints ou point à point sur près de 65'000 adresses différentes. Selon leur place et leur fonction dans le réseau, ils peuvent être paramétrés différemment:

En coordinateur: Sous ce mode, un module peut gérer plusieurs sous-réseaux mais ne peut pas être utilisé en sleep mode, il sera donc utilisé pour le module de la carte réceptrice.

En Router: Ce mode lui permet de gérer un seul sous-réseau et de se mettre en sleep-mode. Il peut donc avoir des modules dans son réseau et communiquer avec d'autres router et le coordinateur du réseau global.

En End Device: Mode définissant le module comme étant en bout de chaîne. Ce mode est utilisé pour les Xbee se trouvant dans un réseau mais ne devant pas gérer d'autres modules. Cette configuration, utilisée pour les cartes émettrices, accepte également le sleep mode.

Pouvant être alimentés entre 2,1 et 3,6V, ils consomment environ 10mA en étant réveillé, un pic de 40mA lors des transmissions RF et moins d' 1uA en sleep mode selon le data sheet qui se trouve sur le cd annexe 8 au rapport.

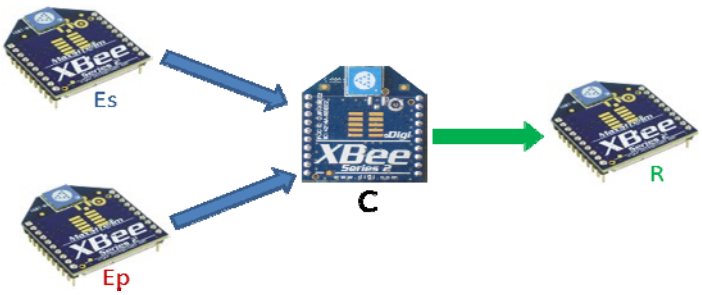


Figure 5 - mise en réseau des module RF

2.1.1 Particularité

Le système est prévu pour fonctionner avec un microcontrôleur et non seul. Dans notre cas nous avons éliminé le microcontrôleur ce qui implique un trafique radio plus conséquent et donc une gestion de ce trafique est nécessaire.

On constate par mesures que le délai entre l'envoi et la réception de l'acquittement est de 109 ms pour un paquet.

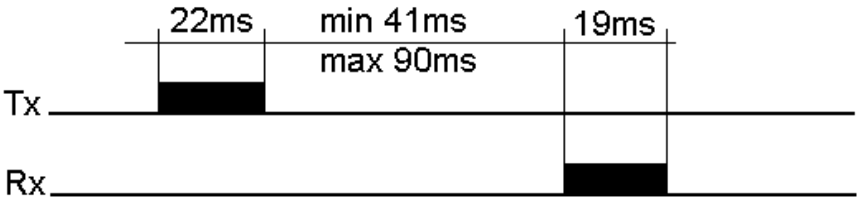


Figure 6 - mesure de l'envoi d'une trame

En envoyant 2 paquets le délai passe a 135ms l'on peut donc envoyer 2 paquets à la suite mais pas plus sinon il y a perte de données.

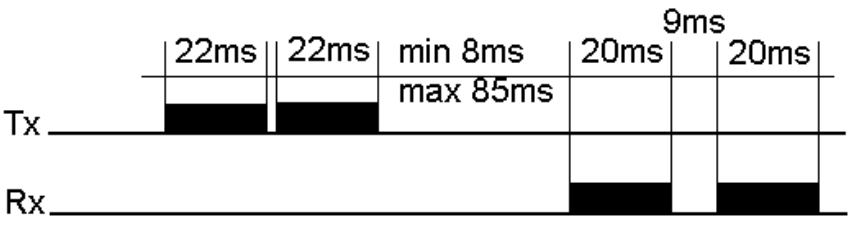


Figure 7 - mesure de l'envoi de 2 trames consécutives

2.2 Communication Xbee

Le module XBee peut être connecté à n'importe quel circuit présentant une liaison UART pour autant que les niveaux de tension ne doivent pas être adaptés.

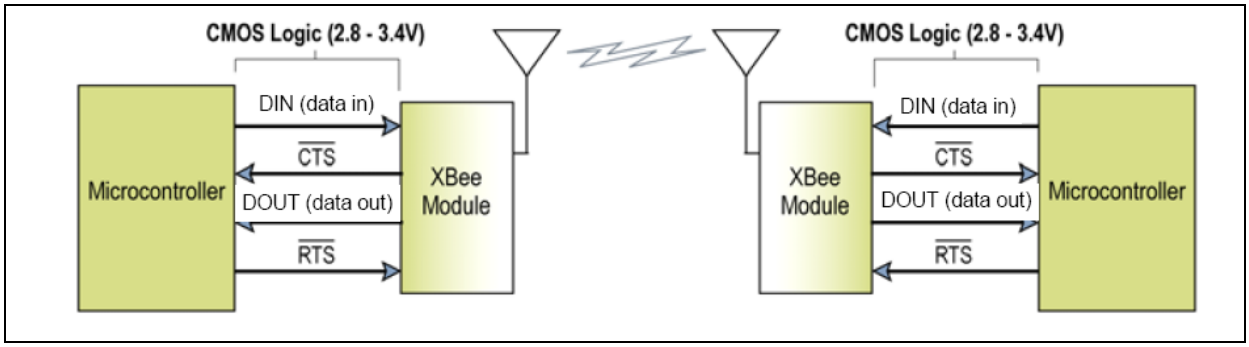


Figure 8 - illustration de la communication via Xbee

La communication est de type asynchrone entre le module et l'UART d'un microcontrôleur et se fait selon la figure 8. Chaque paquet est constitué d'un bit start de niveau bas, suivi de 8 bits de données avec le bit de poids faible en premier, et enfin un bit stop de niveau haut. La synchronisation des données et la vérification de la parité sont assurées par le module UART.

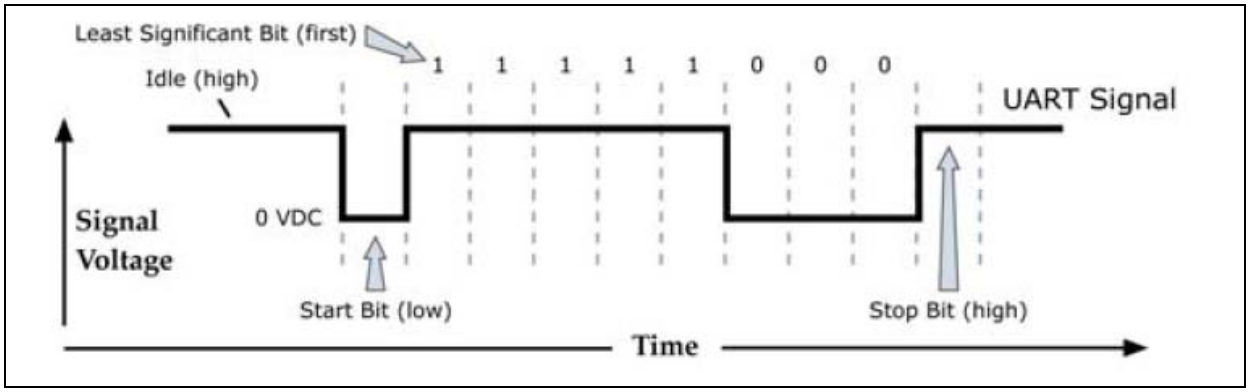


Figure 9 - trame transmise par le module Xbee

Etant connecté à un circuit présentant une liaison série asynchrone, le module utilise un buffer pour stocker les données transmises par l'UART via la pin DIN. Ce flux de données est contrôlé par le signal CTS. Lorsque le buffer ne dispose que d'un espace libre de 138 bits, CTS est mis à 1 afin de signaler au circuit d'arrêter l'envoi de données. Il est remis à 0 lorsque le buffer dispose de 276 bits d'espace mémoire libre.

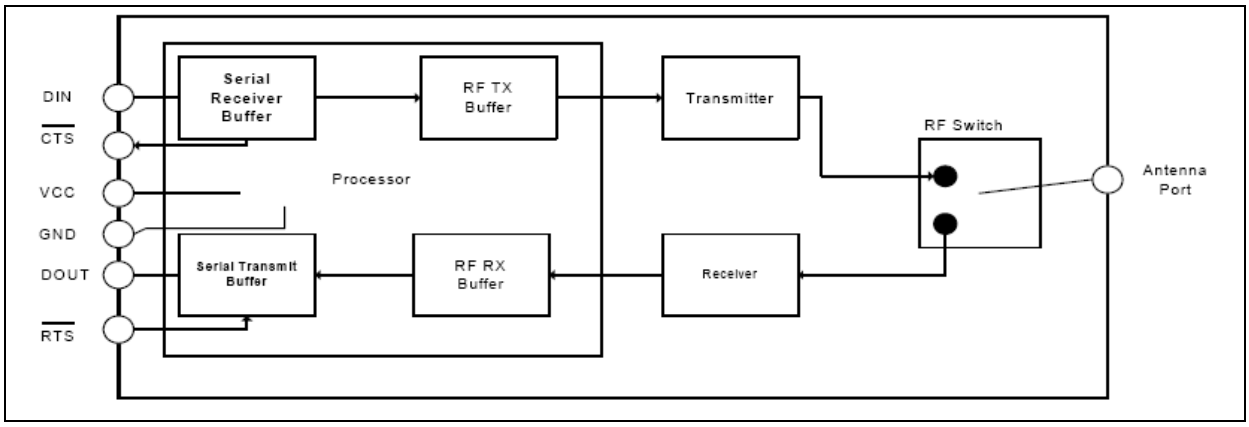


Figure 10 - structure interne des modules

Par ailleurs, un autre buffer est utilisé pour stocker les données reçues via RF. Lorsque ce buffer atteint sa capacité maximale, toute donnée envoyée par voie RF est perdue. Ce flux est contrôlé par le signal RTS. Lorsqu'il est au niveau haut, les données restent stockées dans le buffer. Elles ne sont transmises via la pin DOUT que lorsqu'il est au niveau bas. Il est à noter que lors d'une réception RF, le module Xbee transmet directement la trame reçue au buffer de sortie.

Remarque

Les signaux RTS et CTS ne sont pas gérés par le microcontrôleur dans cette application.

2.3 Trames API

Les modules Xbee peuvent communiquer soit au travers de commandes AT, soit au travers de trames API (Application Programming Interface). Ces trames permettent de transmettre des données ainsi que des messages permettant de vérifier la réussite de la communication. Le type de transmission API est donc préférable.

Les trames API, transmises en Big Endian, se présentent sous la forme suivante:

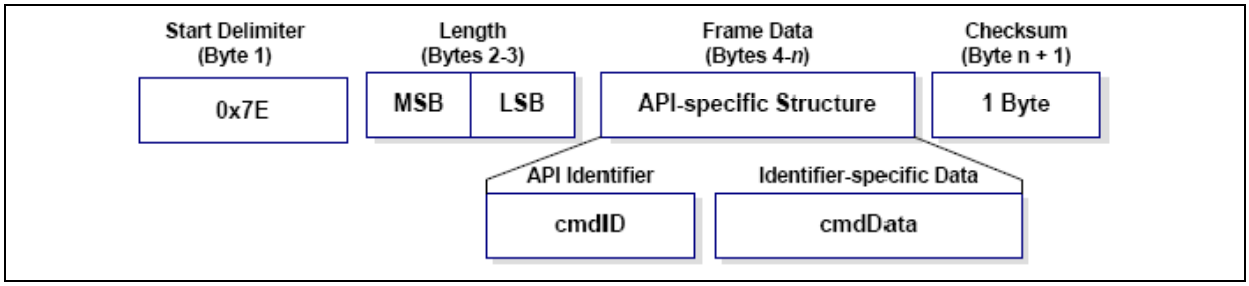


Figure 11 - structure d'une trame API

Start Delimiter	:	Byte permettant à l'utilisateur de définir qu'il s'agit d'une trame API
Length	:	Bytes définissant la longueur totale de la trame sans le Checksum
API Identifier	:	Byte définissant quel type de commande contient la trame
cmdData	:	Données spécifiques selon la commande désirée
Checksum	:	Byte permettant de contrôler la bonne réception de la trame
		Le Checksum équivaut au dernier byte de la soustraction
		0xFF-(somme des bytes compris dans Frame Data)

Les commandes et les types de trames possibles sont détaillées dans le data sheet du module Xbee et se trouvent également dur le cd en annexe 8 du rapport.

2.4 Tests et programmations des modules

Afin de faire des tests de communication série et de programmer facilement les modules, Digi met à disposition un programme, X-CTU, téléchargeable gratuitement sur leur site. Ce soft permet de transmettre des trames API ou des commandes AT à un module Xbee se trouvant sur une platine de test via le port série ou USB du PC. Il offre également la possibilité de programmer les modules de manière simple.

2.5 Paramétrage à l'aide de X-CTU

Une fois le module Xbee sur la platine de test, le programme peut-être lancé. Il s'ouvre sur la fenêtre suivante:

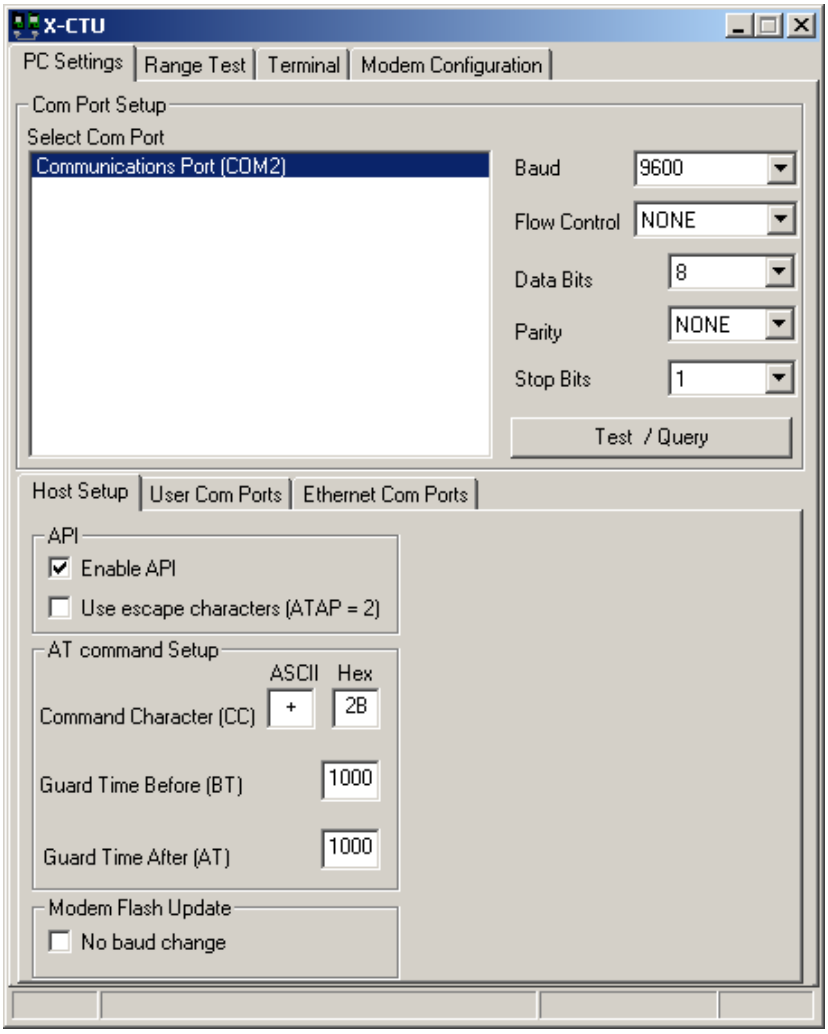


Figure 12 - PC Settings X-CTU

Les paramètres Flow Control, Data Bits, Parity et Stop Bits doivent prendre les valeurs indiquées sur la fenêtre. La vitesse, 9600 bauds pour cette application, et le mode de transmission peuvent être modifiés selon les besoins. Si les informations transmises sont de type API il faut cocher la case "Enable API".

Une fois les paramètres entrés, un test de communication peut-être effectué ce qui ouvre la fenêtre suivante:

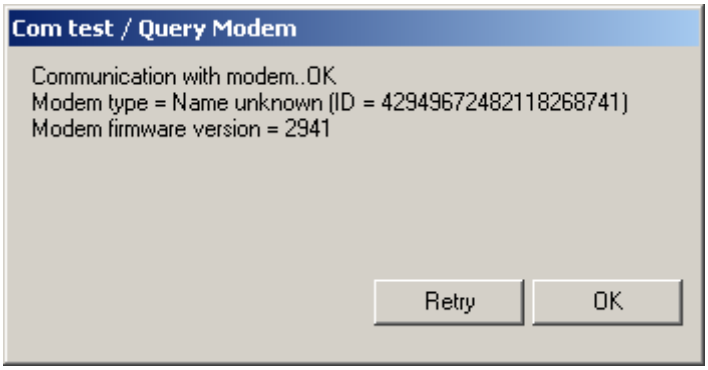


Figure 13 - Fenêtre Test X-CTU

Après avoir cliqué sur ok, l'accès aux paramètres du module se fait en sélectionnant l'onglet Modem Configuration. Un click sur read permet d'afficher la configuration du Xbee se trouvant sur la carte de test.

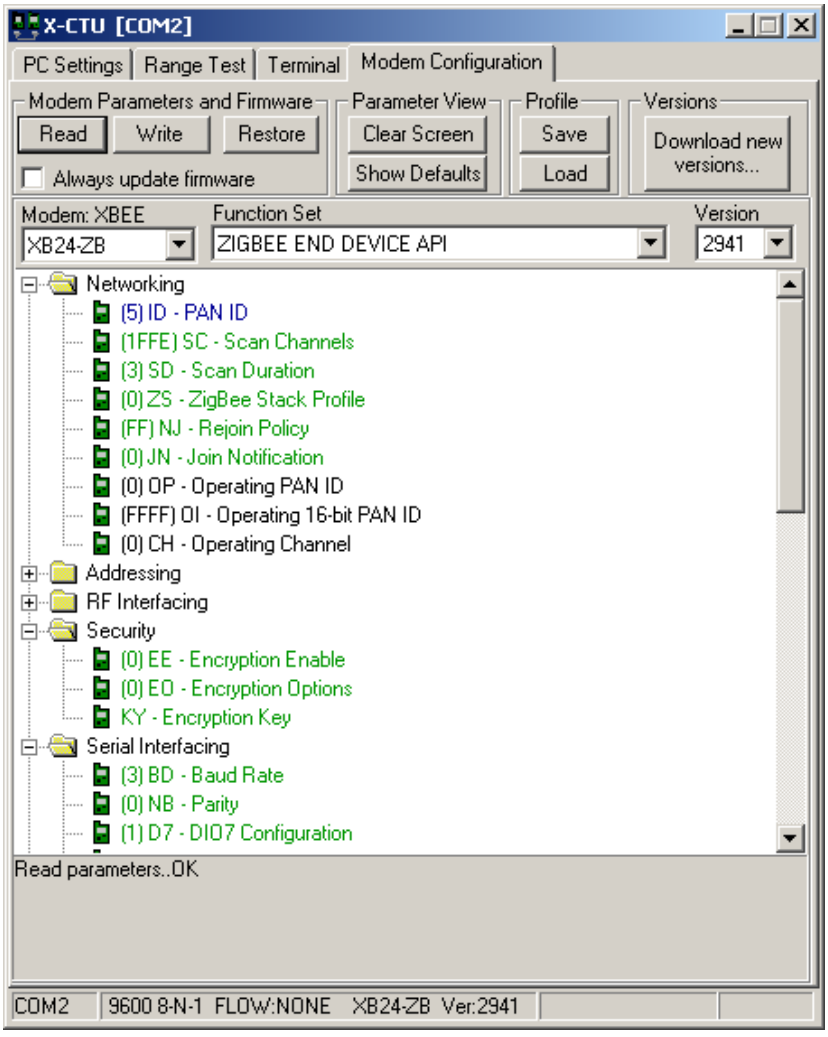


Figure 14 - Configuration du module X-CTU

Pour paramétrer le module en coordinateur ou en end device, il suffit de le sélectionner dans la liste se trouvant dans la barre de recherche Function Set.

Les paramètres importants pour l'application qui nous intéressent sont:

ID - PAN ID	Ce nombre doit être commun à tous les modules se trouvant dans un même réseau.
BD - Baud Rate	Définition de la vitesse de communication série du module.
SM -Sleep Mode	Définition du fonctionnement du sleep mode. Ce paramètre doit être mis à '1' pour les modules End Device. Le coordinateur n'a pas ce paramètre dans sa configuration.

Les autres paramètres n'ont pas besoin d'être changés. Une fois la configuration établie, le module peut-être programmé en cliquant sur write. Le Xbee peut ensuite être retiré de la plaque **après avoir coupé l'alimentation**.

2.5.1 Tests de communication

Afin de tester la communication entre le PC et les XBee ou entre les XBee, le logiciel XbeeCom, développé par la HES-SO valais permet d'envoyer des trames entrées manuellement. Il faut tout d'abord choisir le baud rate qui est de 9600 dans notre cas puis le port de communication.

- L'onglet API frame permet de choisir le type de trame à envoyer :

AT Commande	permet l'envoi de la trame sur le module Xbee qui est relié au port com. du PC.
Remote AT Commande	permet l'envoi d'une trame sur un module Xbee par liaison radio à l'aide du module qui est relié au port com.

  Les autres paramètres de API frame ne sont pas utilisés.

- L'onglet 64-bit address permet de choisir l'adresse de destination de la trame
- L'onglet AT commande permet de choisir le type de données à transmettre. La liste détaillée des commandes se trouve dans le datasheet du module Xbee.
- L'onglet Frame data nous affiche la trame qui est envoyée par le programme en Hexadécimale.
- L'onglet receive affiche toutes les trames reçues par le module Xbee relié au port com.

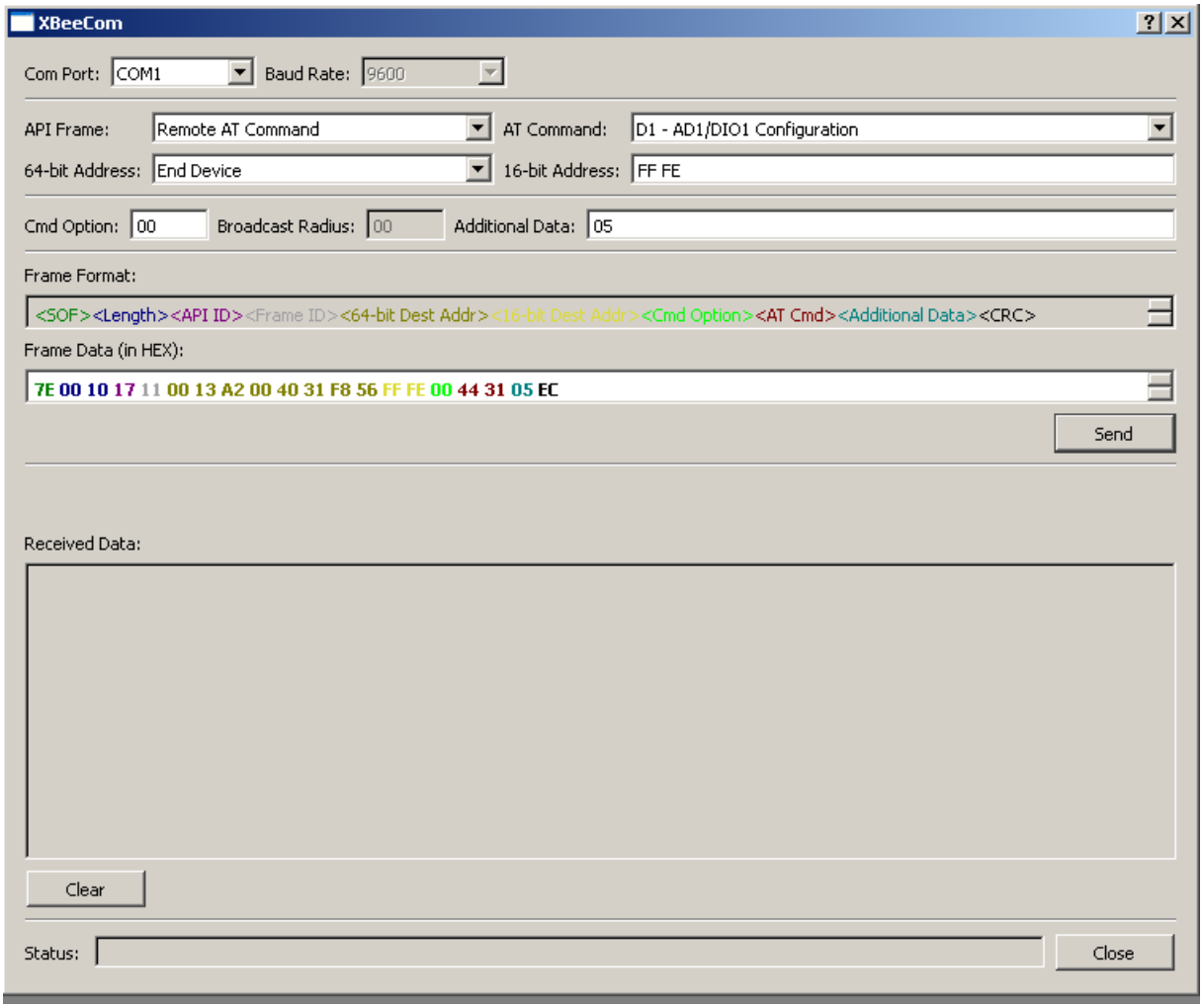


Figure 15 - Exemple de trame envoyée avec XBeeCom

3 Programmation du microcontrôleur

Afin de communiquer entre les modules Xbee et les différents périphériques présents, un microcontrôleur de type PIC est utilisé. Ce type de contrôleur correspond très bien au cahier des charges du projet; il peut être mis en "sleep mode" afin d'optimiser la consommation, la plus part possède un module UART (Universal Asynchronous Receiver Transmitter) et il peut également travailler en 3,3V ce qui évite une adaptation des signaux transmis par les modules Xbee. Tout le système fonctionne sans microcontrôleur sauf la commande principale qui gère le toute avec un seul microcontrôleur

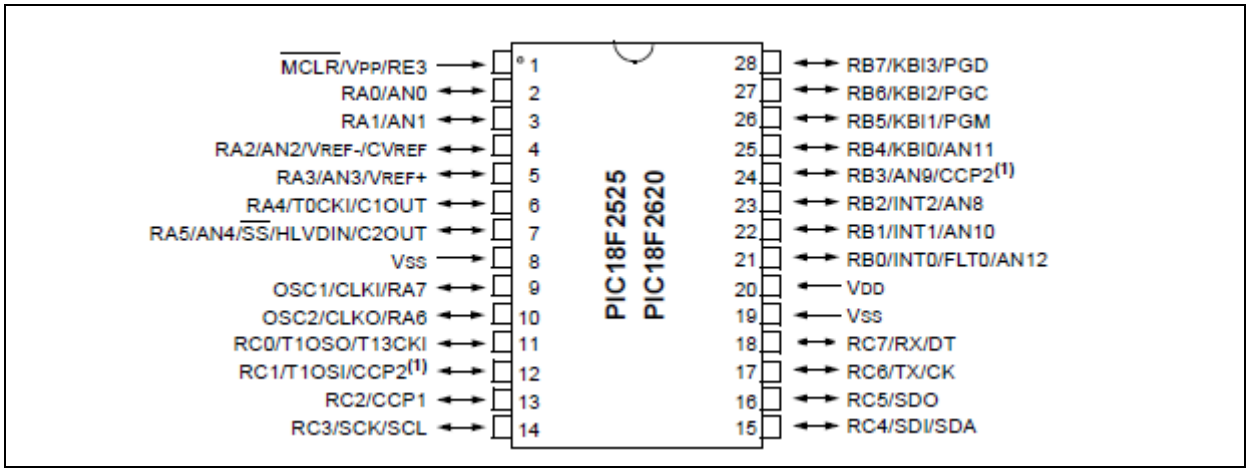


Figure 16 - diagramme des pins du PIC18F2525

Cette partie du rapport est une approche générale du microcontrôleur utilisé, le PIC18F2525. Elle permet d'appréhender l'UART, l'I2C, le SPI et les interruptions du contrôleur. Les codes écrits pour les différentes étapes ainsi que les paramètres des différents registres utilisés sont expliqués dans les commentaires des codes qui se trouvent en annexe n°4. Une explication plus détaillée des paramètres est faite ci-dessous.

Il est à noter que le microcontrôleur de la carte commande principale peut être reprogrammée à tout moment à l'aide d'un module MPLAB-ICD2 disponible dans l'enceinte de l'école.

Pour une étude plus approfondie du microcontrôleur la lecture du data sheet est nécessaire. Ce document se trouve en intégralité sur le cd annexe 8 du rapport.

3.1 Oscillateur interne

Il existe plusieurs façons de configurer l'oscillateur. En externe par un circuit RC, avec un quartz et autres, mais aussi en interne. C'est cette dernière solution qui a été choisie afin de diminuer le nombre de composants. Cet oscillateur comme le schéma ci-dessous le démontre permet de travailler à des fréquences entre 31 KHz et 8MHz.

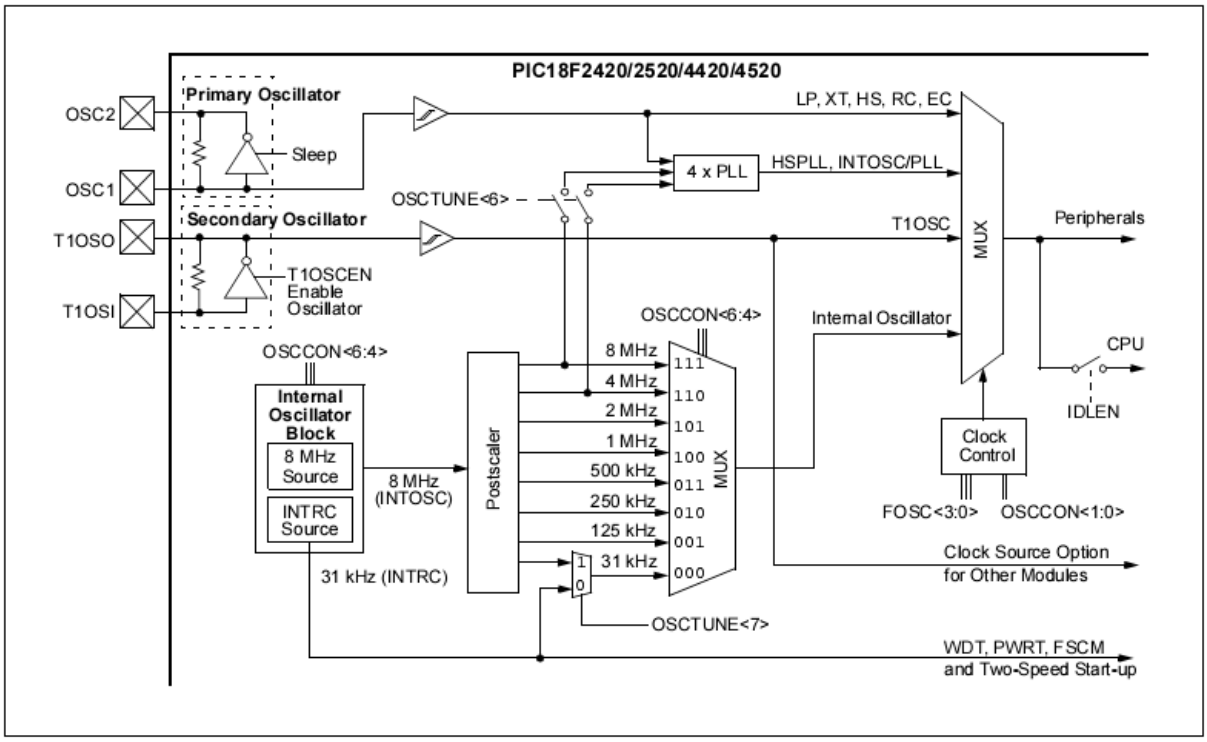


Figure 17 – schémas de principe de l'oscillateur

3.1.1 Registre OSCCON, oscillator control register

Ce registre permet de sélectionner la fréquence de l'oscillateur. Il est configuré avec la valeur 0X70 ce qui sélectionne la fréquence de 8MHz.

REGISTER 2-2: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0	R/W-1	R/W-0	R/W-0	R(1)	R-0	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **IDLEN:** Idle Enable bit
1 = Device enters an Idle mode on SLEEP instruction
0 = Device enters Sleep mode on SLEEP instruction

bit 6-4 **IRCF<2:0>:** Internal Oscillator Frequency Select bits
111 = 8 MHz (INTOSC drives clock directly)
110 = 4 MHz
101 = 2 MHz
100 = 1 MHz(3)
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz (from either INTOSC/256 or INTRC directly)(2)

bit 3 **OSTS:** Oscillator Start-up Timer Time-out Status bit(1)
1 = Oscillator Start-up Timer (OST) time-out has expired; primary oscillator is running
0 = Oscillator Start-up Timer (OST) time-out is running; primary oscillator is not ready

bit 2 **IOFS:** INTOSC Frequency Stable bit
1 = INTOSC frequency is stable
0 = INTOSC frequency is not stable

bit 1-0 **SCS<1:0>:** System Clock Select bits
1x = Internal oscillator block
01 = Secondary (Timer1) oscillator
00 = Primary oscillator

Note 1: Reset state depends on state of the IESO Configuration bit.
2: Source selected by the INTSRC bit (OSCTUNE<7>), see text.
3: Default output frequency of INTOSC on Reset.

Figure 18 - registre RCCON du pic

3.2 Port d'entrées, sorties

Les ports d'entrées, sorties sont configurés à l'aide des registres TRIS(A,B,C) pour la direction du signal (entrée / sortie) et PORT(A,B,C) pour leur valeur initial. Les particularités spécifiques à chaque port sont reprisent ci-dessous.

3.2.1 Port A

Le port A est configuré en entrée afin de recevoir les signaux des boutons de commande.

Ces signaux sont de type numérique il faut donc configurer les registres suivants :

ADCON1 qui définit si le signal est de type numérique ou analogique.

CMCON qui défini le mode de fonctionnement des comparateurs. Dans notre cas ils sont désactivés.

3.2.2 Port B

Le port B est configuré en sortie. Il envoie les signaux pour l'affichage sur l'écran ainsi que la Led rouge qui signal un problème.

3.2.3 Port C

Le port C est dédié à la communication avec l'horloge temps réel (I2C) et avec le module XBee(UART). Sa configuration est décrite plus en détaille dans les chapitres « UART » et « I2C ». il reçoit également le signal du capteur de température.

3.3 Communication SPI

La communication avec l'écran s'effectue par liaison sérielle. Le port SPI étant partagé avec l'I2C, il a donc fallu utiliser d'autres pins en réalisant la liaison SPI par une routine software qui fonctionne selon le graphe suivant. Avec la particularité de traiter un byte à la fois. (CS1 passe au niveau haut après le bit D0)

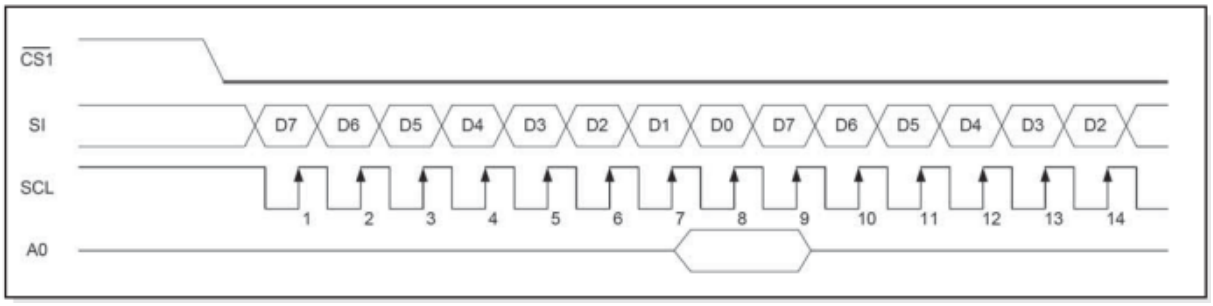


Figure 19 - diagramme de communication SPI

Etant donné que la communication avec l'écran est unidirectionnelle, asynchrone et l'acquisition des données s'effectue au flanc montant de l'horloge, la routine développée ci-dessious ne s'occupe que de l'envoi des données. Pour plus d'informations le data sheet de l'écran est disponible en annexe 8 sur le cd.

```
void envoi_SPI( char type_spi, char n_spi )
{   char z = 0;
    // donner le signal de transmition
    CS1=0;
    // choix du type de donnée 1= afficher 0= commande
    A0 = type_spi;
    asm("nop");
    //déclément du bit de donnée de 7 à 0
    for (z =0;z<8;z++)
    {
        SCL= 1;
        if ((n_spi & mask) == 0x80)
        {
            si = 1;
        }
        else
        {
            si = 0;
        }
        n_spi = n_spi << 1;
        SCL=0;
        asm ("nop");
    }
    SCL=1;
    asm("nop");//attente avant arret
    CS1=1;
}
```

Figure 20 - Code de la routine d'envoi SPI

3.4 Communication UART

Lorsque aucune information n'est transmise, l'entrée de l'UART, pin 17 RC6/Tx/CK, reste au niveau haut. Si le module Xbee transmet une trame, le start bit (niveau bas) définit le début de la réception. Les 8 bits de données qui suivent sont enregistrés dans le registre RCREG. Une fois la réception complète, le flag RCIF du registre PIR 1 est mis à '1'. A cet instant la valeur de RCREG correspond au byte transmit par le module Xbee. Une fois ce byte lu, RCIF est automatiquement remis à '0'. Une répétition de la lecture du registre RCREG permet d'enregistrer la trame API complète transmise par le Xbee.

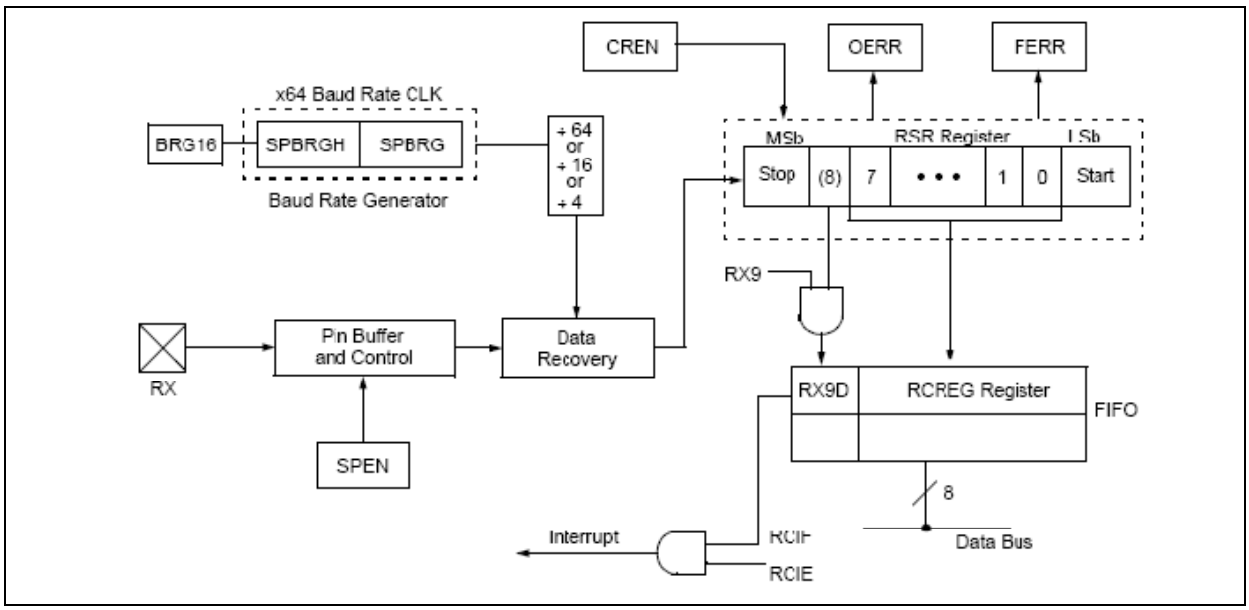


Figure 21 - schéma block de la réception sur l'UART

Lorsque le PIC veut transmettre des informations, les bytes sont chargés dans le registre TXREG. Une fois ce registre plein, le byte à transmettre est transmit automatiquement dans le registre TSR ce qui set le flag TXIF du registre PIR1. Cette opération réalisée, les données sont transmises sur la liaison série et le flag TMRT du registre TXSTA est mis à '1' dès que le registre est vide. La répétition de cette opération permet l'envoi de la trame API complète.

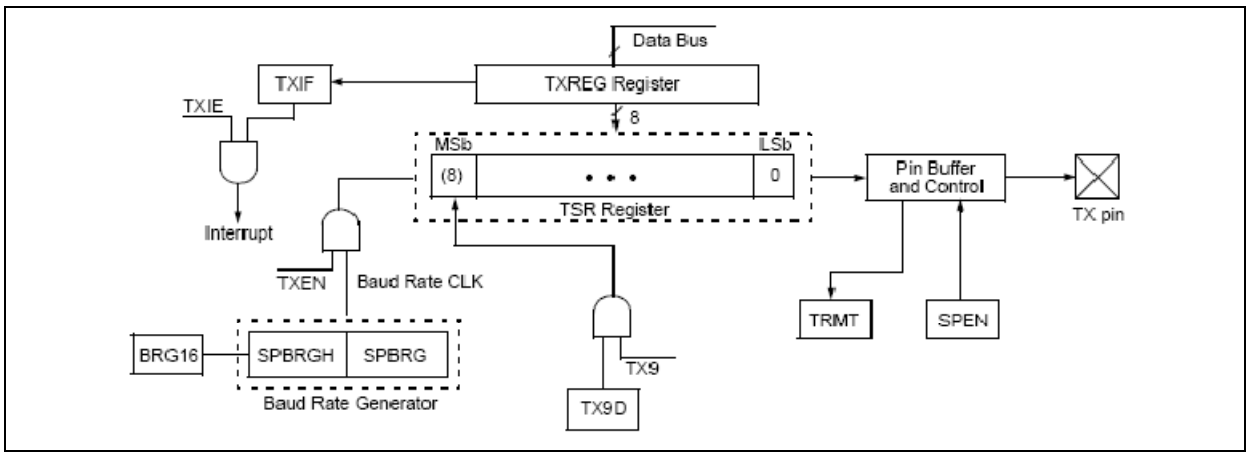


Figure 22 - schéma block de la transmission sur l'UART

La transmission entre le Xbee et le PIC ce faisant via l'UART, il est nécessaire de paramétrer les différents registres du microcontrôleur de façon à pouvoir utiliser cette liaison correctement. Trois registres sont nécessaires au paramétrage de la transmission série qui nous intéresse, TXREG, TXSTA et BAUDCON.

3.4.1 Registre TXSTA, Transmit Status and Control Register

Ce registre doit avoir la valeur 0x24. Cela définira une liaison asynchrone sur 8 bits de données. La mise à '1' de BRGH permet une communication à une vitesse "élevée".

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-Bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed
Synchronous mode:
Don't care.

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th Bit of Transmit Data
Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode.

Figure 23 - registre TXSTA du PIC

3.4.2 Registre RCSTA, Receive Status and Control Register

Ce registre contient les paramètres de la réception. Il prend la valeur 0x90 ce qui permet d'activer le port série et la réception.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-Bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-Bit (RX9 = 0):
Don't care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be cleared by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit, CREN)
0 = No overrun error

bit 0 **RX9D:** 9th Bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

Figure 24 - registre RCSTA du PIC

3.4.3 Registre BAUDCON, Baud rate Control Register

Ce registre prend la valeur 0x08 de manière à paramétrer le baud rate sur 16 bits. Cette option permet de diminuer l'erreur de baud rate par rapport à un paramétrage sur 8 bits.

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **ABDOVF:** Auto-Baud Acquisition Rollover Status bit
1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
0 = No BRG rollover has occurred

bit 6 **RCIDL:** Receive Operation Idle Status bit
1 = Receive operation is Idle
0 = Receive operation is active

bit 5 **RXDTP:** Data/Receive Polarity Select bit
Asynchronous mode:
1 = Receive data (RX) is inverted (active-low)
0 = Receive data (RX) is not inverted (active-high)
Synchronous mode:
1 = Data (DT) is inverted (active-low)
0 = Data (DT) is not inverted (active-high)

bit 4 **TXCKP:** Clock and Data Polarity Select bit
Asynchronous mode:
1 = Idle state for transmit (TX) is a low level
0 = Idle state for transmit (TX) is a high level
Synchronous mode:
1 = Idle state for clock (CK) is a high level
0 = Idle state for clock (CK) is a low level

bit 3 **BRG16:** 16-Bit Baud Rate Register Enable bit
1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored

bit 2 **Unimplemented:** Read as '0'

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RX pin not monitored or rising edge detected
Synchronous mode:
Unused in this mode.

bit 0 **ABDEN:** Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
0 = Baud rate measurement disabled or completed
Synchronous mode:
Unused in this mode.

Figure 25 - registre BAUDCON du PIC

3.4.4 Exemple de définition du Baud Rate

Une fois les registres correctement paramétrés, le baud rate peut être défini à l'aide des bytes SPBRG et SPBRGH.

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-Bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-Bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-Bit/Asynchronous	
0	1	1	16-Bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-Bit/Synchronous	
1	1	x	16-Bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

Figure 26 - Formules de calcul du baud rate

La communication étant asynchrone sur 8 bits, une transmission rapide étant souhaitée et le paramétrage du baud rate définit sur 16 bits la formule de calcul est:

$$BaudRate = \frac{Fosc}{4 \cdot (n+1)}$$

En fixant le baud rate à 9600 et la fréquence du PIC à 16MHz la valeur de 'n' peut-être calculée.

$$n = \frac{Fosc}{4 \cdot BaudRate} - 1 = \frac{8 \cdot 10^6}{4 \cdot 9600} - 1 = 208.333$$

La valeur 208, 0x00D0, peut être mise dans les bytes SPBRGH et SPBRG de manière à définir le baud rate.

SPBRGH = 0x00

SPBRG = 0xD0

3.4.5 Erreur de Baud Rate

La valeur de 'n' n'étant pas un nombre entier, une erreur de baud est à prendre en compte.

$$BaudRateRéel = \frac{8 \cdot 10^6}{4 \cdot (208+1)} = 9569{,}37$$

$$Erreur = \frac{9569{,}37 - 9600}{9600} = 0.0031 \rightarrow 0.31\%$$

Cette erreur très faible est tout à fait acceptable et ne risque pas de compromettre la communication série.

3.5 Communication I2C

Le port I2C permet la liaison avec l'horloge temps réel. Tout les paramètres de liaisons sont dans le data sheet en annexe 8. Les registres qui gèrent cette communication sont SSPCON1, SSPCON2 ainsi que quelques flags paramètres.

SSPADD gestion du baude rate par la formule suivante

$$Fi2c = \frac{Fosc}{(4*(SSPADD+1))}$$

SSPIF flag des interruptions

BCLIF flag des colisions

SMP slew rate (400KHz)

FIGURE 17-16: MSSP BLOCK DIAGRAM (I²C™ MASTER MODE)

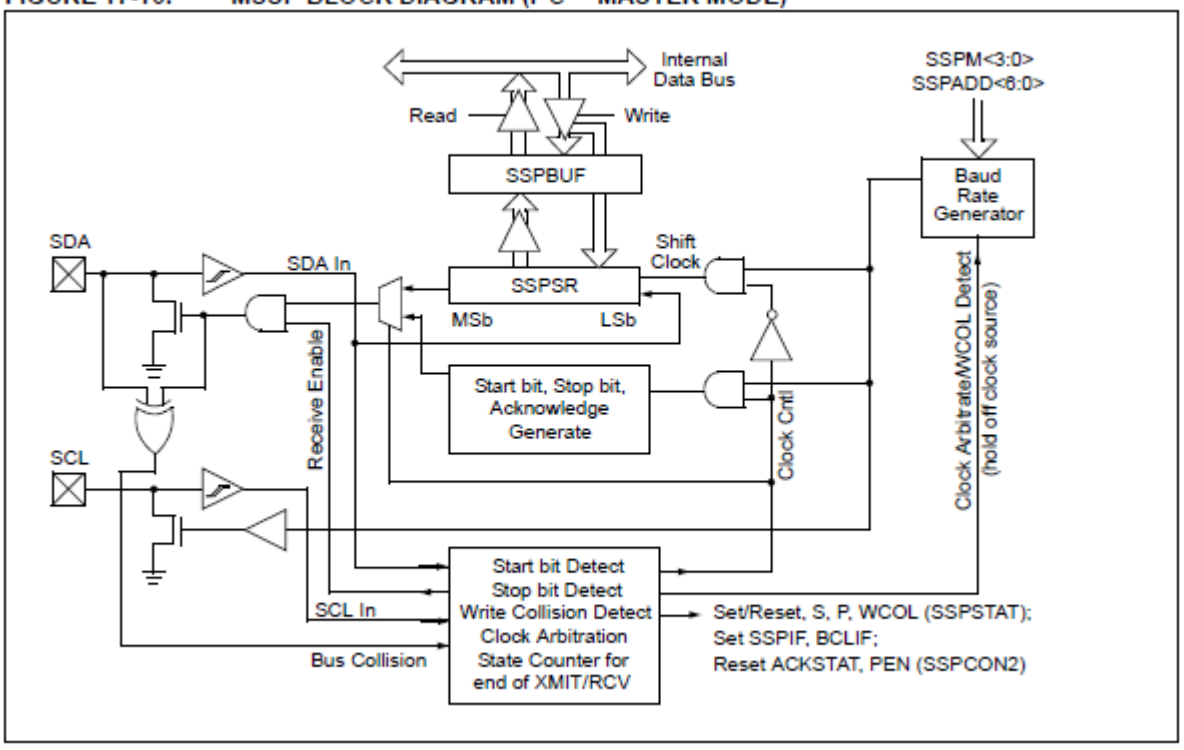


Figure 27 - liaison I2C

SSPADD étant un chiffre entier (19) il n'y a pas d'erreur de baud rate.

3.5.1 Registre SSPCON1, MSSP Control Register (I2C mode)

Ce registre permet de configurer la communication I2C en master avec la valeur 0x38.

REGISTER 17-4: SSPCON1: MSSP CONTROL REGISTER 1 (I²C™ MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN(1)	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **WCOL:** Write Collision Detect bit
In Master Transmit mode:
1 = A write to the SSPBUF register was attempted while the I²C conditions were not valid for a transmission to be started (must be cleared in software)
0 = No collision
In Slave Transmit mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision
In Receive mode (Master or Slave modes):
This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit
In Receive mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared in software)
0 = No overflow
In Transmit mode:
This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(1)
1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCK Release Control bit
In Slave mode:
1 = Releases clock
0 = Holds clock low (clock stretch), used to ensure data setup time
In Master mode:
Unused in this mode.

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits(2)
1111 = I²C Slave mode, 10-bit address with Start and Stop bit interrupts enabled
1110 = I²C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1011 = I²C Firmware Controlled Master mode (Slave Idle)
1000 = I²C Master mode, clock = FOSC/(4 * (SSPADD + 1))
0111 = I²C Slave mode, 10-bit address
0110 = I²C Slave mode, 7-bit address
Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

Note 1: When enabled, the SDA and SCL pins must be properly configured as inputs or outputs.

Figure 28 - registre SSPCON1 du pic

3.5.2 Registre SSPCON2, MSSP Control Register (I2C mode)

Ce registre contient les flags pour le fonctionnement de la communication.

REGISTER 17-5: SSPCON2: MSSP CONTROL REGISTER 2 (I^2C™ MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT(2)	ACKEN(1)	RCEN(1)	PEN(1)	RSEN(1)	SEN(1)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **GCEN:** General Call Enable bit (Slave mode only)
1 = Enables interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled.

bit 6 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)(2)
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)(1)
1 = Initiates Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence Idle

bit 3 **RCEN:** Receive Enable bit (Master mode only)(1)
1 = Enables Receive mode for I^2C
0 = Receive Idle

bit 2 **PEN:** Stop Condition Enable bit (Master mode only)(1)
1 = Initiates Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enable bit (Master mode only)(1)
1 = Initiates Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enable/Stretch Enable bit(1)
In Master mode:
1 = Initiates Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle
In Slave mode:
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is disabled

Note 1: For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I^2C module is not in the Idle mode, these bits may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).
2: Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

Figure 29 - registre SSPCON1 du pic

3.6 Interruption

Les interruptions permettent de ne pas louper un événement important. Les interruptions sont gérées par les registre INTCON et PIE1 pour les flags qui doivent générer une interruption ainsi que leurs activations.

FIGURE 9-1: PIC18 INTERRUPT LOGIC

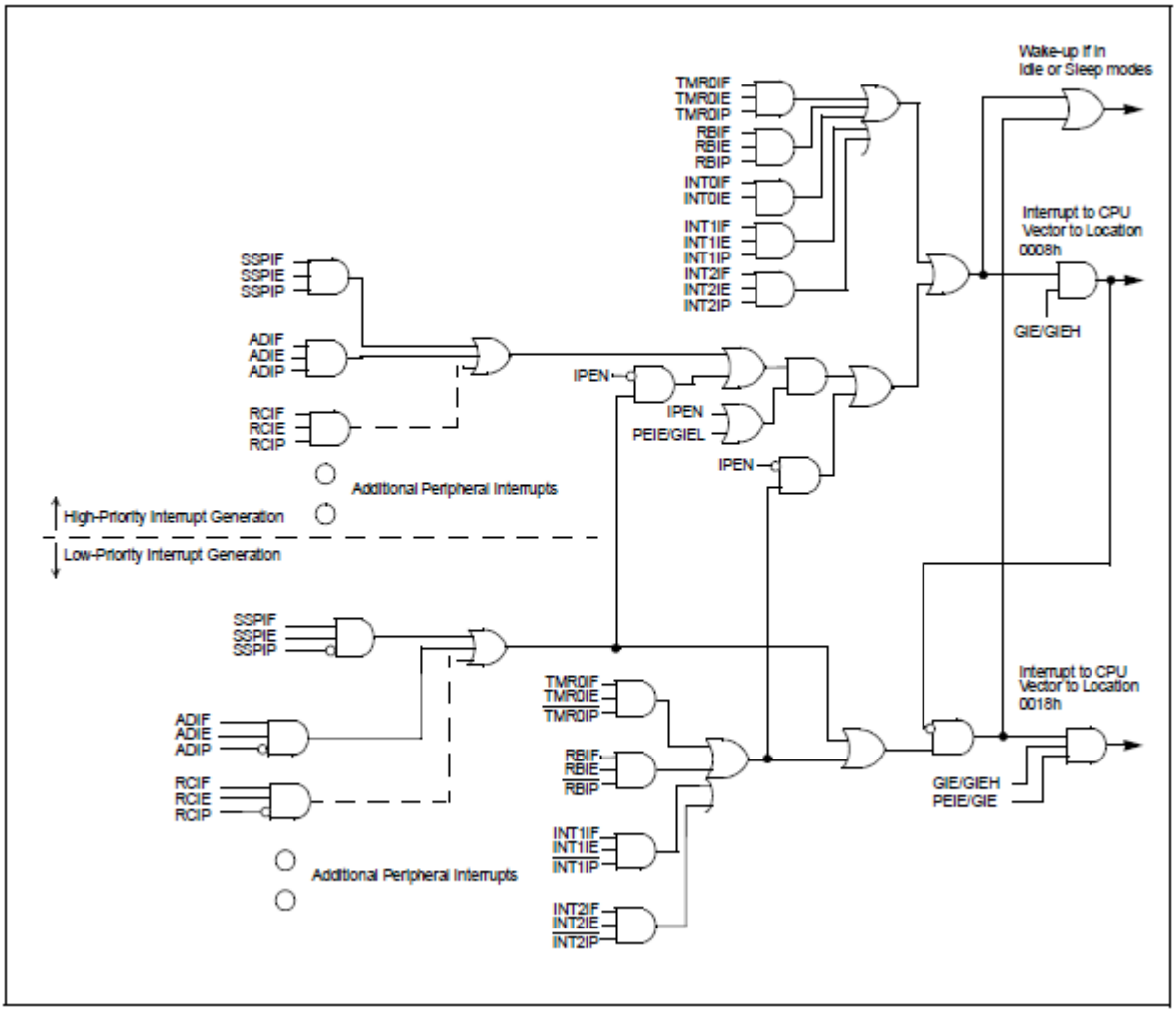


Figure 30 - logique d'interruption

3.6.1 Registre INTCON, Interrupt Control Register

Ce registre contient les flags pour l'activation des interruptions. Ainsi que GIE et PEIE qui gèrent l'activation et la désactivation générale.

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high-priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low-priority peripheral interrupts
0 = Disables all low-priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit(1)
1 = At least one of the RB<7:4> pins changed state (must be cleared in software)
0 = None of the RB<7:4> pins have changed state

Note 1: A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

Figure 31 - registre INTCON du pic

3.6.2 Registre PIE1, Periferal Interrupt Enable Register 1

Ce registre contient les flags pour l'activation des interruptions.

REGISTER 9-6: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit 6 **ADIE:** A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE:** EUSART Receive Interrupt Enable bit
1 = Enables the EUSART receive interrupt
0 = Disables the EUSART receive interrupt

bit 4 **TXIE:** EUSART Transmit Interrupt Enable bit
1 = Enables the EUSART transmit interrupt
0 = Disables the EUSART transmit interrupt

bit 3 **SSPIE:** Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Note 1: This bit is unimplemented on 28-pin devices and will read as '0'.

Figure 32 - registre PIE1 du pic

3.6.3 Evénements

Il existe plusieurs événements qui demande des interruptions ils sont décrit ci-dessous :

Réception de données du module XBee

Il est important de ne pas louper un paquet qui arriverait sur le bus UART. Cela impliquerait un retard dans l'échantillonnage du système. La réception de données sur ce bus génère donc une interruption par le flag RCIF qui écrit toutes les trames reçues dans un rolling buffer.

Réception de données de l'horloge temps réel

La réception des données provenant de l'horloge temps réel est importante mais pas vitale. Une erreur lors de la réception impliquerait un décalage pendant une période de 60 secondes. Ce qui est tout à fait admissible.

Timer 0

Le Timer 0 est utilisé pour obtenir un signal d'horloge de 0.5 Hz. A chaque passage dans l'interruption le système peut ainsi incrémenter des variables et gérer des temporisations avec la seconde comme unité.

4 Design et fonctionnalités des cartes

Les structogrammes complets, les codes commentés, les schématiques, les schémas d'implantations, les schémas mécaniques des boîtiers, la liste des composants des quatre cartes ainsi que les protocoles de tests se trouvent en annexe 1 a 6 du rapport.

4.1 Carte Commande moteur

La carte Cmd_moteur permet de commander les stores en fonction des trames reçues par la carte commande principale. Elle est alimentée via un transformateur Traco Power 230V/3,3V. Cette solution évite l'utilisation de régulateur de tension.

Les informations sont réceptionnées par un module Xbee qui commande des relais dont les contacts activent la commande du moteur des stores dans un sens ou dans l'autre. Les bobines de ces relais SMD monostables fonctionnent dès 2,25V.

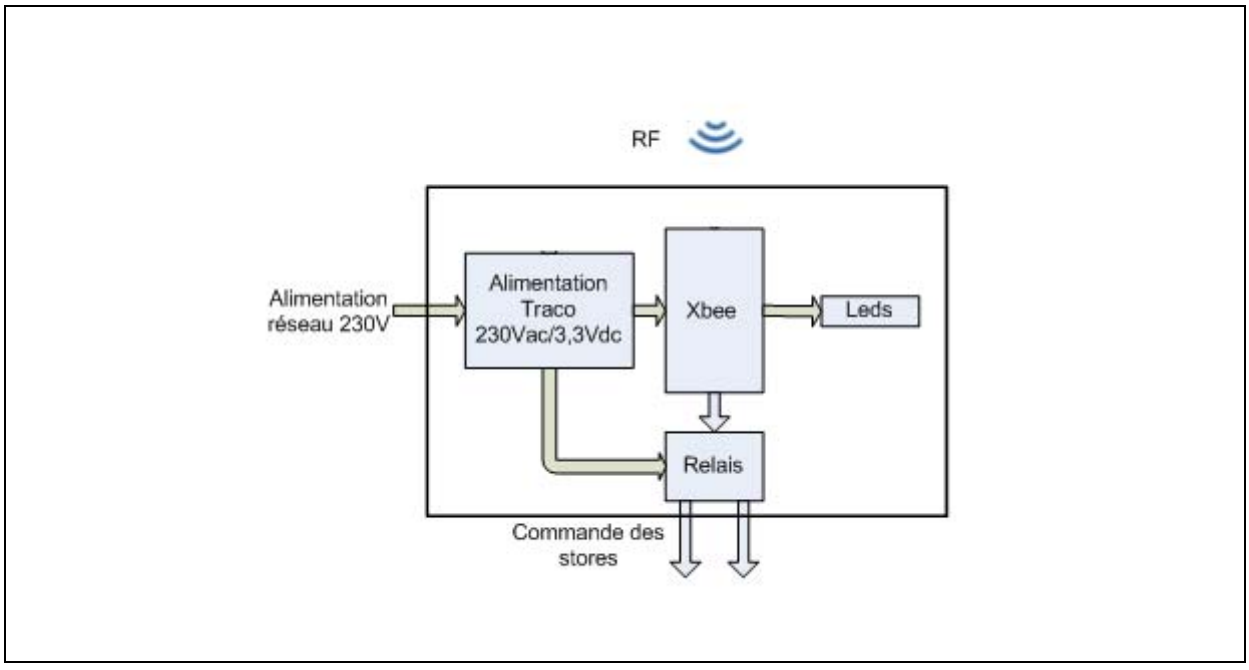


Figure 33 - schéma block de la carte réceptrice (R)

Une led verte permet de connaître l'état du module XBee. Elle clignote quand le module n'est pas en mode sleep et reste éteinte en mode sleep. Elle peut être désactivée à l'aide d'un jumper

Des problèmes de communication pouvant survenir entre les modules Xbee, un bouton poussoir de reset est aussi présent sur la carte. De plus un bouton poussoir permet de vérifier la liaison radio avec le coordinateur.(fonction comissioning)

Figure 34 – carte réceptrice

La consommation étant moins importante sur cette carte étant donné que l'alimentation provient du réseau, le module XBee n'entre jamais en sleep mode. Le module XBee est configuré comme end device avec l'activation et désactivation du mode sleep par un signal externe et non des timer comme les autres cartes. Cela permet ainsi d'être réactif aux commandes radio de la Cmd_principale. Cela permet aussi de ne pas charger le réseau RF par les données de réveil de ce module.

La liste des composants avec les prix se trouve en annexe 6

4.2 Cartes émettrices

Les cartes émettrices transmettent à la carte commande principale l'état de leur capteur respectif via un module Xbee. Toutes deux sont alimentées par deux piles 1,5V ayant une capacité totale de 2,4 Ah. La consommation est optimisée par l'utilisation de composants adaptés au niveau de tension des piles et par une mise en « sleep mode » des modules Xbee lorsqu'aucune trame n'est transmise.

Figure 35 - photos des cartes émettrices (Ep, Es)

Les deux cartes permettent de transmettre l'état des capteurs ainsi que le niveau de tension de la pile. Ce qui permet de savoir si celui-ci s'approche du niveau minimal permettant à la carte de fonctionner correctement.

L'élimination du microcontrôleur implique une consommation énergétique plus faible en mode sleep mais un trafique plus important des données RF. Car le module XBee attend ses ordres d'activation et désactivation de l'alimentation ou de l'envoi de la valeur du capteur par RF au-lieu d'être commandé par le microcontrôleur.

Tout comme la carte Command moteur, elles sont équipées d'un bouton de reset, d'un bouton de liaison RF et d'un jumper permettant de couper l'alimentation. Les deux cartes possèdent une led verte pour signaler l'état du module XBee. Elle clignote quand le module est réveillé et reste éteinte en mode sleep. Elles peuvent être désactivées à l'aide d'un jumper.

4.2.1 Carte présence

Le capteur de présence consommant en stand by 170uA, la coupure de son alimentation permet une économie d'énergie non négligeable. Lorsque le capteur n'est pas utilisé, le module XBee coupe son alimentation.

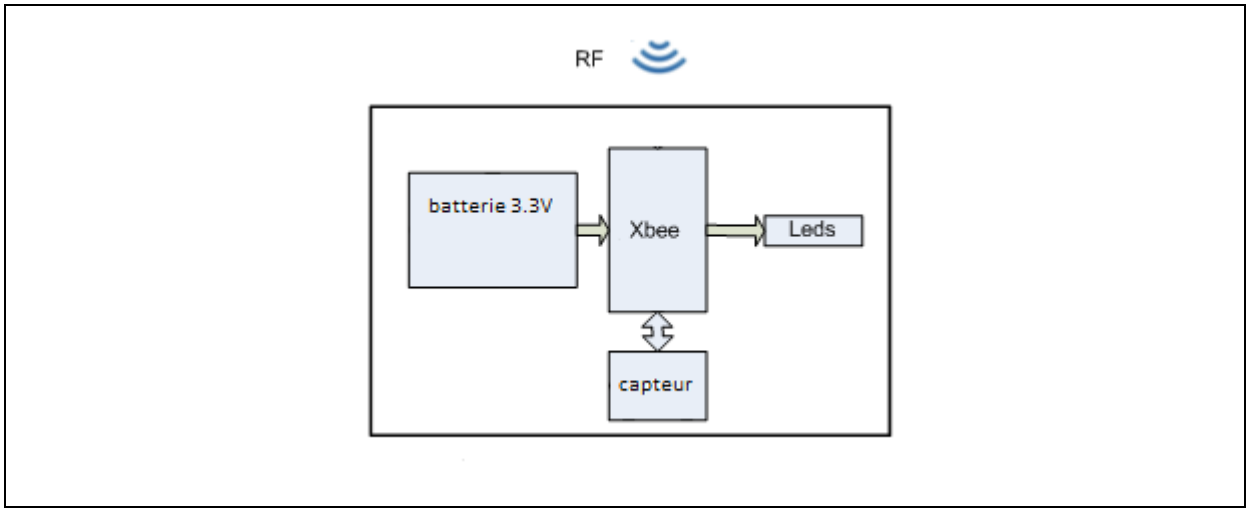


Figure 36 - schéma block de la carte présence

La carte est en mode sleep la plus part du temps. Elle se réveille chaque 15 min et transmet sur ordre la valeur de son capteur à la commande principale. Le temps de transmission varie en fonction du capteur et de ses besoins. Dans notre cas le capteur est stable après un temps d'alimentation de 30 secondes selon le datasheet, mais par mesures et test 10 secondes suffisent à stabiliser le capteur. Ce point augmente la consommation par la prolongation du temps d'éveil du module XBee.

La liste des composants avec les prix se trouve en annexe 6. Le data sheet du capteur se trouve en annexe 2.

4.2.2 Carte rayonnement

La carte rayonnement nous informe de l'intensité lumineuse à l'extérieur de la pièce, bâtiment ou autres. La mesure s'effectue par une photodiode qui converti le rayonnement dans le spectre de 350 a 820 nm en une tension électrique. Cette tension est amplifiée afin de correspondre aux niveaux min et max de l'entrée analogique de module XBee. Il permet ainsi de choisir la plage de mesure désirée. Chaque 15 minutes le système sort du mode sleep. Par communication RF avec la Command principale le système transmet sur demande la valeur mesurée.

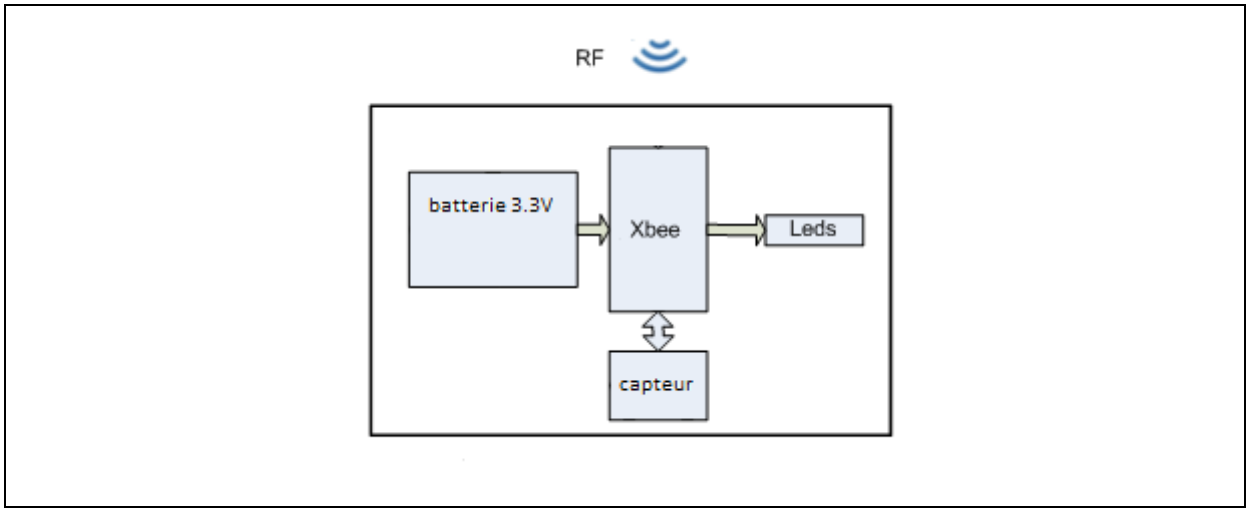


Figure 37 - schéma block de la carte rayonnement (Es)

Afin d'économiser l'énergie, l'alimentation la photodiode est coupée par le XBee lors du mode sleep.

La liste des composants avec les prix se trouve en annexe 6. Le data sheet du capteur se trouve en annexe 1.

4.3 Carte Command principale

Cette carte est le centre du système. Elle se compose d'un module XBee pour la communication, d'une horloge temps réel, d'un capteur de température, d'une led verte pour l'état du module XBee, d'une led rouge pour signaler un problème à l'utilisateur, d'un écran pour l'affichage et de boutons pour l'IHM.

A cause de sa consommation élevée, l'alimentation par batteries a dû être écartée. La carte est donc alimentée par le secteur. Un chip permet de stabiliser la tension d'alimentation à 3.3V et 350mA max avec une tension d'entrée entre 2.7V et 16.5V.

Sa consommation n'étant plus un point vital, le rétro-éclairage de l'écran a été ajouté pour le confort.

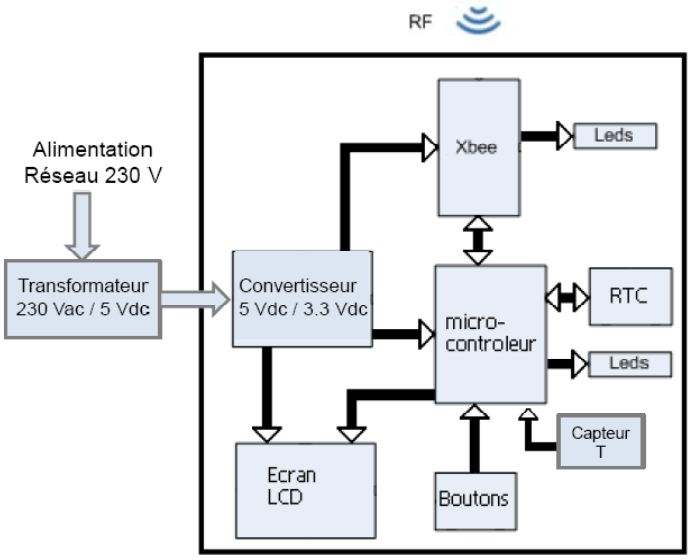


Figure 38 - Schémas de principe Command Principale

La liste des composants avec les prix se trouve en annexe 6.

5 Consommation énergétique et autonomie

Selon la figure 38 avec une consommation à 100mA, et sachant que la tension de fonctionnement peut descendre jusqu'a 2.1V, on déduit que l'on peut soutirer 95% des ressources de la pile.

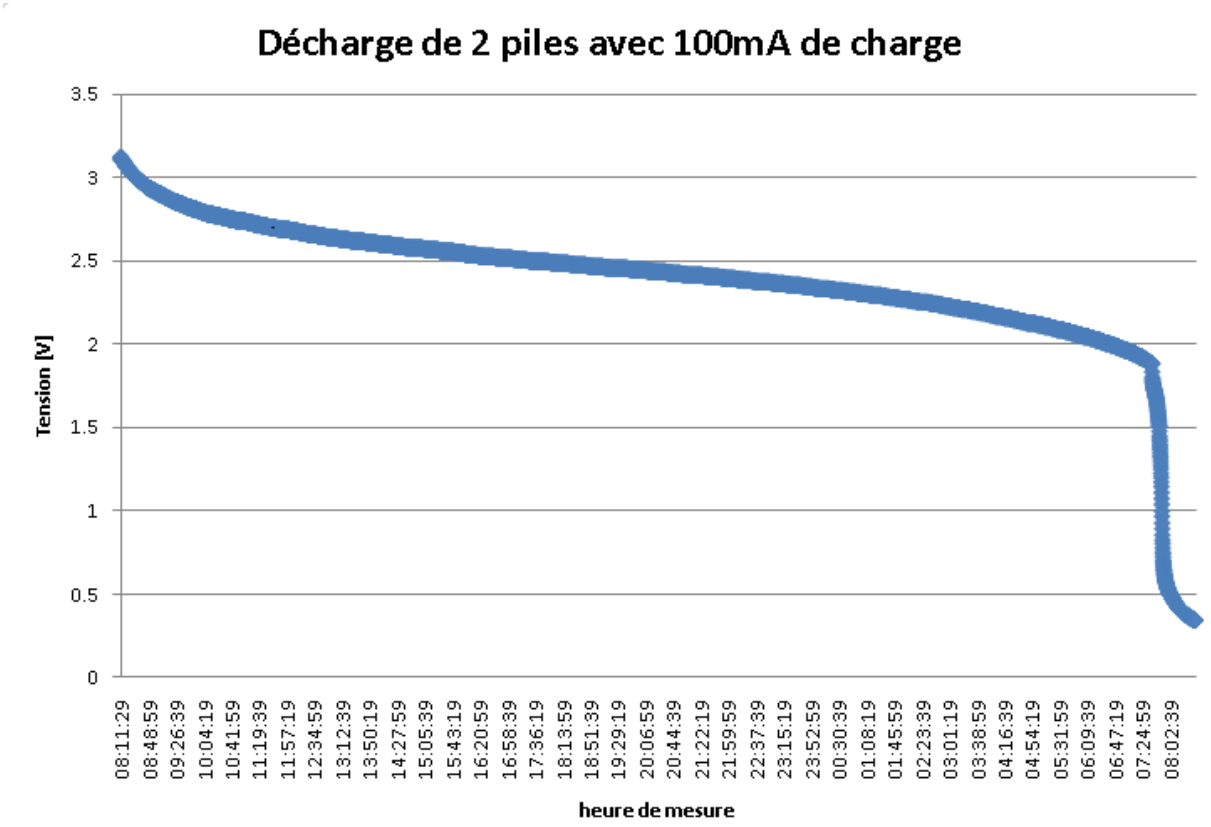


Figure 39 - graphique de décharge de 2 piles

Le calcul d'autonomie, avec l'estimation de 9 secondes d'écoute et 1 seconde d'envoie de données RF, se fait comme suite

$$(\ 10mA * 9sec + 40mA * 1sec + 30uA * 890\ sec) * 4 = 0.624mAs$$

Pour une année la consommation du système rapporté à la seconde est de :

$$0.624 * 24\ heures * 365\ jours = 5490mAs$$

La charge des 2 piles est de 2400mAh = 10080mAs avec le rendement de 95% selon le graphe ci-dessus cela donne 9576mAs.

L'autonomie de la carte est donc de

$$\frac{9576 * 12}{5490} = 20.9\ mois$$

Carte	consommation	durée	autonomie max
Carte rayonnement	40mA	1 sec	
	10mA	9 sec	
	30uA	890 sec	20.9 mois

Carte	consommation	durée	autonomie max
Carte présence	40mA	1 sec	
	10mA	9 sec	
	100uA	890 sec	15 mois

Command moteur	40mA sous 3.3V	connecté au réseau 230
Command principale	40 a 100mA sous 3.3V	connecté au réseau 230

6 Logique de commande et de gestion

La logique de commande et gestion du système est expliquée en détaille dans le cahier des charges en annexe 7. Les grandes lignes du fonctionnement sont décrites ci-dessous.

6.1 Mode manuel

Le mode manuel est le plus primaire. Il donne la commande à l'utilisateur. Seul les boutons monter, descendre et stop sont activés. Il est signalé sur l'écran par « mode manuel ».

Pour passer en mode manuel il faut soit, activer le bouton « man / auto » ou lorsqu'une présence est détectée le système commute automatiquement.

Pour sortir du mode manuel il faut soit désactiver le bouton « man / auto » ou lorsque le bouton est sur auto et que aucune présence n'est détectée.

6.2 Mode automatique Astro

Le mode Astro est l'un des deux modes automatiques. Une plage horaire fixe est définie pour chaque mois de l'année. Durant la journée les stores sont ouverts et fermés durant la nuit. Le paramètre « fonctionnement été » permet d'inverser l'état des stores durant l'été (juin à aout).

Plages horaires

Mois	Début	Fin
Janvier	9h00	17h00
Février	8h20	18h00
Mars	7h40	19h00
Avril	7h00	20h00
Mai	6h20	21h00
Juin	5h40	22h00
Juillet	5h40	22h00
Aout	6h20	21h00
Septembre	7h00	20h00
Octobre	7h40	19h00
Novembre	8h20	18h00
Décembre	9h00	17h00

Avec le paramètre «pos. Intermédiaire » est possible de donner une position intermédiaire au store en fixant une temporisation sur la descente et montée du store mais cette option n'a pas été implémentée dans cette version.

6.3 Mode automatique Save Energy

Le mode Save Energy est le deuxième mode automatique. Chaque 15 minutes une mesure de luminosité est effectuée ainsi qu'une mesure de présence dans la pièce. Le système réagit en fonction de ces mesures afin de laisser entrer la chaleur et de la conserver au mieux en hiver, inversement en été. En cas de présence la commande manuelle est prioritaire.

Algorithme appliqué pendant l'hiver

IF Soleil **AND** Pièce non-occupée → Monter automatiquement les stores

IF Couvert **AND** Pièce non-occupée → Descendre automatiquement les stores

IF Pièce occupée → Commande manuelle des stores

Algorithme appliqué pendant l'été

IF Soleil **AND** Pièce non-occupée → Descendre automatiquement les stores

IF Couvert **AND** Pièce non-occupée → Monter automatiquement les stores

IF Pièce occupée → Commande manuelle des stores

L'utilisateur peut choisir le seuil de détection de la luminosité entre 0 et 100%. Il peut également choisir un nombre de mesures au dessus du seuil entre 1 et 3 avant d'agir(plus de détail dans le manuel utilisateur annexe 5). Cela permet d'introduire une temporisation et éviter des mouvements trop fréquents.

7 Affichage, menu et paramètres

Un manuel utilisateur est disponible en annexe 5 il reprend dans les détails l'affichage principal, la composition du menu ainsi qu'une explication de chaque paramètre avec son chemin d'accès.

8 Tests et résultats

Tous les tests réalisés sur les cartes ou le programme sont en annexe 1 à 4 dans les protocoles de tests. Ils comportent des tests hardware, les configurations des modules XBee, les tests de fonctionnement et de consommation.

Dans l'ensemble tout fonctionne correctement. La communication entre les cartes est stable et bien détectée par les modules XBee. Il y a tout de même une erreur sur la mesure de température qui est instable.

Par manque de temps, aucun test de rendement et de gain énergétique à long terme n'a été réalisé.

Les tests ont étés réalisés sur une maquette. Touts les temporisations et les paramètres des modules ont donc été configurés pour fonctionner avec cette maquette. Le comportement du système correspond au cahier des charges avec l'adaptation des temporisations.

9 Améliorations

Affichage sur l'écran : il faudrait travailler à une fréquence supérieure à 8 MHz afin de diminuer le temps de travail et ainsi ne plus avoir de rafraichissement visible sur l'écran. Cela implique une horloge externe.

Ajout de paramètres : par manque de temps et afin de ne pas surcharger le menu déjà très lent des paramètres utiles ont été supprimés.

+- 120 minutes pour ajuster l'heure en mode astro

Position intermédiaire des stores en mode astro

Paramètres : en cas de coupure de courant les paramètres choisis par l'utilisateur sont perdus. Il faudrait les sauvegarder dans la mémoire EEPROM afin qu'ils ne soient pas perdus. La solution se trouve dans le HTSOFT en annexe 8 sous EEPROM.

Command principale : le système principal pourrait être relié à une super capa ou un grand condensateur afin de ne plus être sensible aux microcoupures du réseau.

Écran : connecter l'entrée reset de l'écran au microcontrôleur afin de garantir qu'à chaque démarrage l'écran ne soit pas bloqué.

Programme : il serait intéressant de connaître la position du store afin d'éviter de multiplier les commandes lors du mode automatique. Le simple ajout de deux variables bloquant la répétition des commandes suffirait.

Command moteur : prévoir une version alternative afin de piloter le moteur directement avec des relais qui supportent le 230VAC

Liaison RF : il serait intéressant de gérer les flags de saturation de la communication UART. Cela améliorerait la fiabilité du système.

10 Conclusions

Par le développement de ce nouveau prototype le système est plus facile a prendre en main. L'interface utilisateur permet une meilleur compréhension et utilisation du produit par l'utilisateur.

Le design des composants a été amélioré. Le changement des piles est plus facile. Des paramètres permettent de personnaliser le fonctionnement et de l'adapter a ses besoins, envies.

Le système développé est plus petit, moins gourmand en énergie pour les capteurs. Il remplit les fonctionnalités : manuelle, Astro et Save enery selon le cahier des charges. L'interface homme machine est rudimentaire mais fonctionne très bien.

Dans l'ensemble les tests de fonctionnement sont satisfaisant bien qu'ils n'aient pas étés effectué a long terme.

Sion, le 12.07.2010

Johan Walthert

11 Remerciements

Je tiens à remercier toutes les personnes qui on pu m'aider en répondant à mes questions ou en m'orientant vers une ou l'autre solution. A savoir

C. Costa pour les routines de l'écran

P. Sartoretti pour les logiciels de programmation

C. Bianchi pour les questions sur la programmation

C. Truffer pour le suivit du travail et la maquette de démonstration

Ainsi que F. Bützberger pour le suivit global du travail.

12 Bibliographie

Documentation sur le compilateur HT Soft(en annexe sur le CD)

Documentation DIGI XBee (en annexe sur le CD)

Rapport du travail de diplôme de M Bolometti : Réglage par pièce (gestion du chauffage et des stores).2007

Rapport du travail de diplôme de L Fontannaz : gestion des stores électrique en fonction de la lumière et de la présence 2009.

13 Annexes

Annexe 1 : carte rayonnement

- Structogramme
- Protocole de test
- Schémas électrique
- Schémas PCB
- Schémas mécanique
- Data sheet capteur

Annexe 2 : carte présence

Annexe 3 : carte cmd_moteur

Annexe 4 : carte cmd_principale

Annexe 5 : Manuel utilisateur

Annexe 6 : liste matériel

Annexe 7 : Cahier des charges

Annexe 8 : Annexe CD

- Data sheet
- Programme
- Plan mécanique
- Rapport
- Protocole de test
- PCB
- Logiciels pour XBee

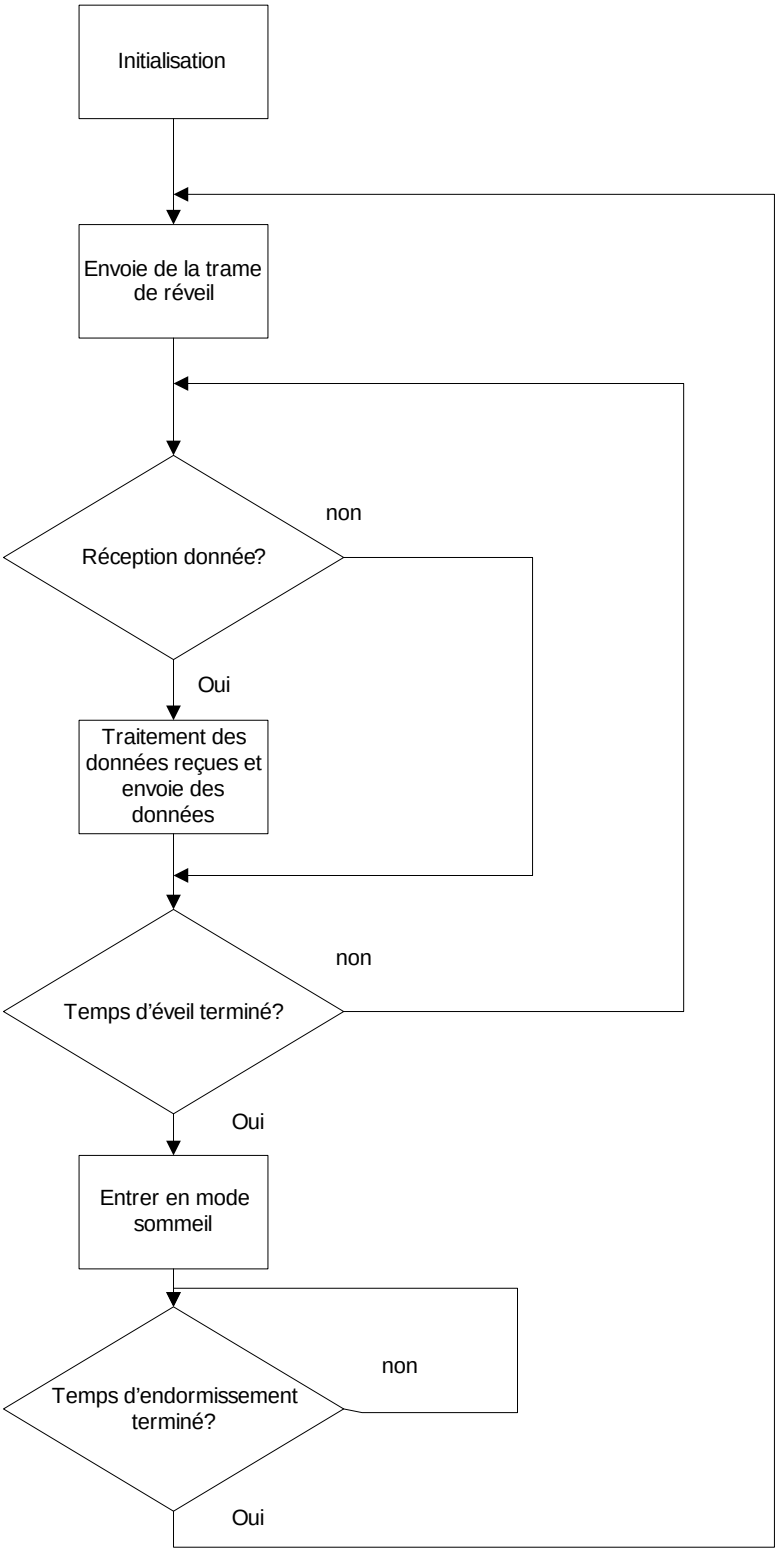
Annexe n°1

Carte rayonnement

Contenu

- Structogramme
- Protocol de test
- Schémas électrique
- Schémas PCB et plan d’implantation
- Schémas mécanique
- Data sheet capteur

Structogramme carte de rayonnement



Protocole de test

Carte rayonnement

Test hardware (sans XBee)

Vérification visuelle des soudures et pistes : _____

Vérification avec la sonnette

Cour circuit GND, VCC _____

Cour circuit GND les autres pistes _____

Cour circuit VCC les autres pistes _____

Alimenter en 3.3V

Pine n°1 alimentation 3.3V _____

Le bouton reset fonctionne correctement Pull down (activé =0V) _____

En alimentant (3V) la pine n°15 la led s'allume (! jumper) _____

Alimenter en 3V PT3 avec la photodiode cachée PT1=0V _____

Alimenter en 3V PT3 avec la photodiode éclairée PT1=2V _____

Alimenter en 3V PT3 avec la luminosité qui varie PT1 varie 0-2V _____

Test softwear

Configuration superviseur

Commande	Valeur	Description	remarque
DL	00	Adresse basse su destinataire	Pas de destinataire uniquement récepteur
DH	00	Adresse haute du destinataire	Pas de destinataire uniquement récepteur
D5	01	DIO5 Configuration	DIO5 Assiciated Indication LED

Configuration end devise

Commande	Valeur	Description	remarque
DL	XX	Adresse basse su destinataire	Le coordinateur peut etre atteint avec l'adresse 0000
DH	XX	Adresse haute du destinataire	Le coordinateur peut etre atteint avec l'adresse 0000
SM	04	Sleep mode	Activation du sleep mode 0 desabled, 04 enabled
SP	01F4 = 5sec	Sleep period	Période de sleep 20-AF0 20 = 320ms AF0 =28s
SN	02	Nbr. of sleep period	Nombre de période sleep 1-FFFF (65535)
ST	4E20 = 20sec	Time Before sleep	Délais avant sleep 1-FFFF (65.5sec)
SO	06	Sleep option	00 pas activé 02 toujours se réveiller pour le ST Time 04 sleep le temps complet SN*SP 06 sleep le temps complet SN*SP+ toujours se réveiller pour le ST Time
JN	01	Join Notification	Envoie d'une notification au autres modules lore de l'enclenchement ou de l'association a un réseau
PR	1FF0	Pull Up resistor	Résistence pull up desactivée pour l'entrée digitale DIO1
D0	01	AD Configuration	Entrée digitale configurée comme commissioning bouton
D1	04	AD Configuration	00 désactivée 03 activée comme entrée 04 sortie digitale a 0
D2	02	AD Configuration	00 desactivée 02 analog input
D4	00	AD Configuration	00 désactivée 04 sortie digitale a 0 05 sortie digitale a 1
D5	01	DIO5 associated	
V+	08F0	Niveau Vcc	Défini le seuil de détection du niveau de tension faible pour l'envoie d'une alerte. 2.66V

1 Le XBee s'endors et se réveille selon le cycle _____ _____

2 La led d'association clignote lors de l'éveil et reste éteinte lors du sommeil _____

3 Lors du réveil par réception de la trame _____

le capteur nous retourne sa valeur _____

3.1 capteur caché trame _____ _____

3.2 capteur semi éclairé trame _____ _____

3.3 capteur éclairé trame _____ _____

4 avec le bouton reset le module se réveil _____

5 par l'activation (0V) du bouton commissioning le module se réveil _____

Consommation

Les mesures sont effectuées à l'oscilloscope

Courant consommé en mode sommeil _____

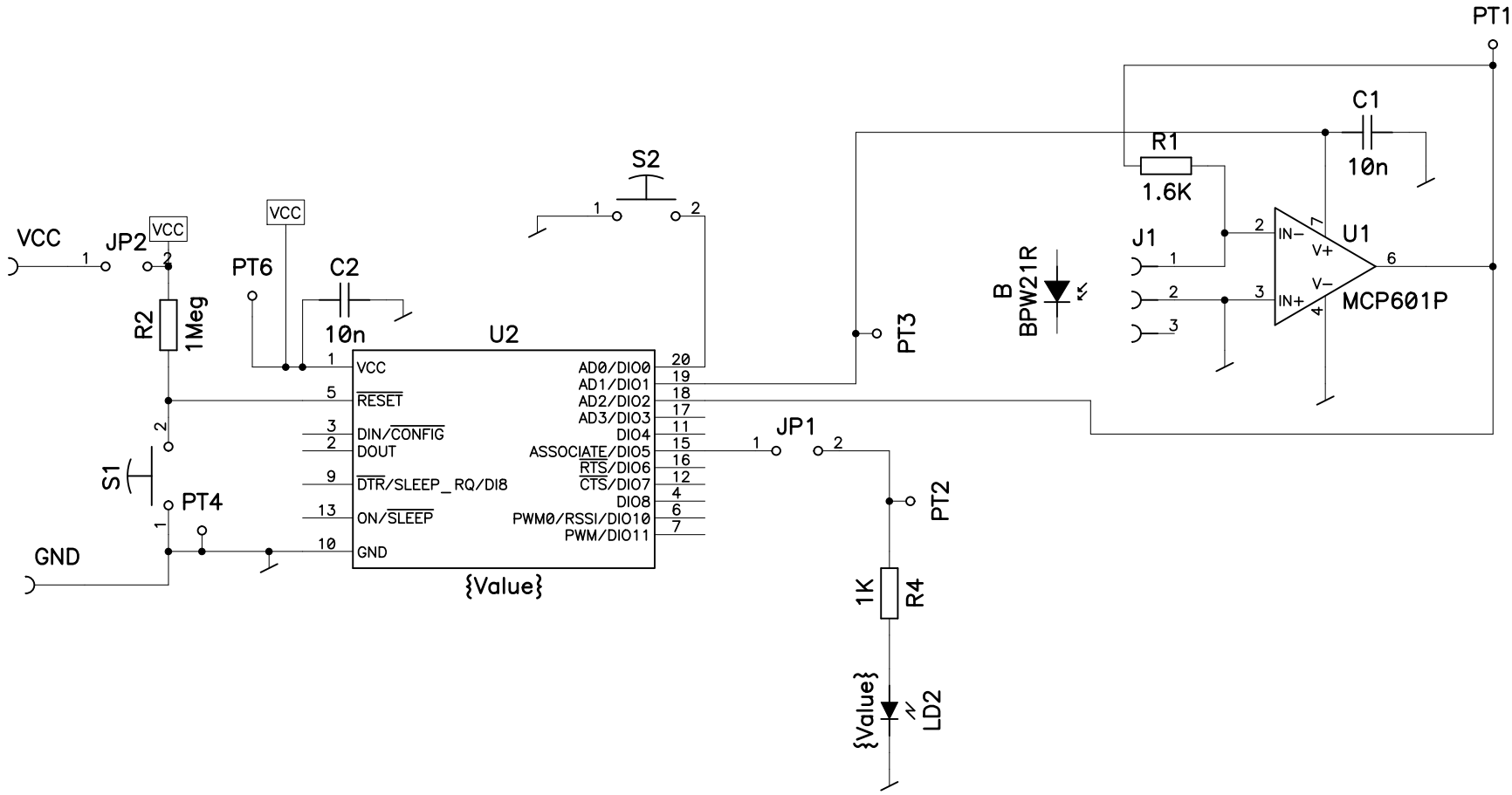
Courant consommé en mode éveil _____

Courant consommé pendant l'envoi d'une trame _____

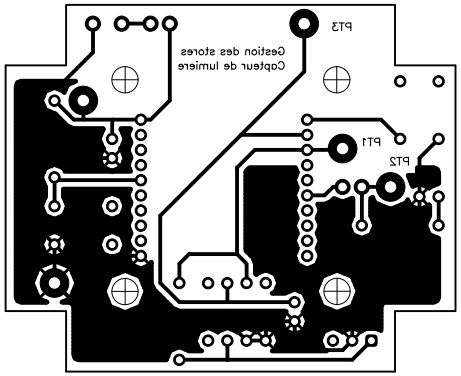
Test ok

Sion, le _____

Walthert Johan

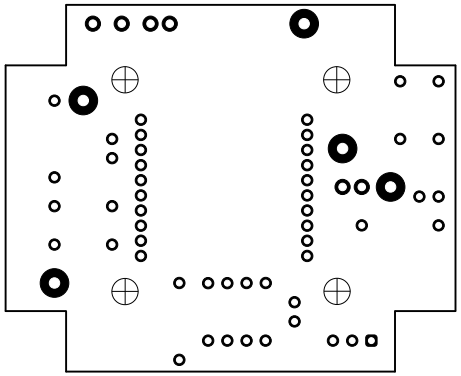


Gestion des stores electriques Carte rayonnement Sheet1	DES	07.06.10Waltherj
	REV	V1.1
HAUTE ECOLE VALAISANNE	1/3	{Path} Capteur_de_lumiere.sch



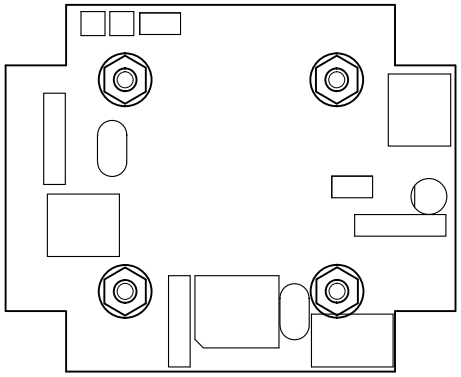
Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.60	+	2	Yes
0.80	X	51	Yes
1.00	Y	6	Yes
1.40	T	5	Yes
3.50	⧖	4	Yes

Gestion des stores electriques Carte rayonnement	DES	17.06.2010
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ capteur_lumiere.pcb	



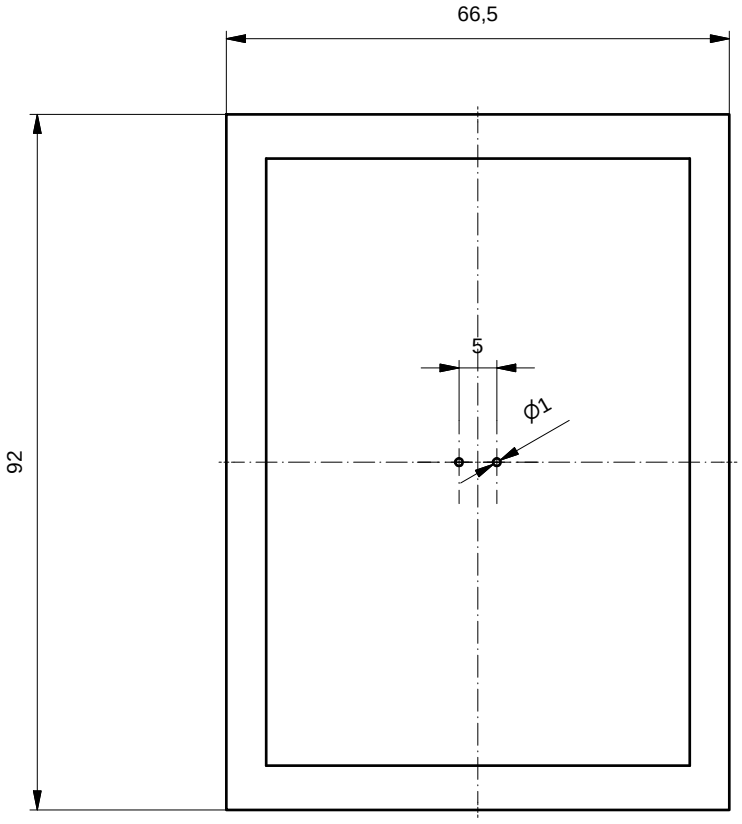
Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.60	+	2	Yes
0.80	X	51	Yes
1.00	Y	6	Yes
1.40	⊤	5	Yes
3.50	⧖	4	Yes

Gestion des stores electriques Carte rayonnement	DES	17.06.2010
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ capteur_lumiere.pcb	



Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.60	+	2	Yes
0.80	X	51	Yes
1.00	Y	6	Yes
1.40	⊤	5	Yes
3.50	⧖	4	Yes

Gestion des stores electriques Carte rayonnement	DES	17.06.2010
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ capteur_lumiere.pcb	



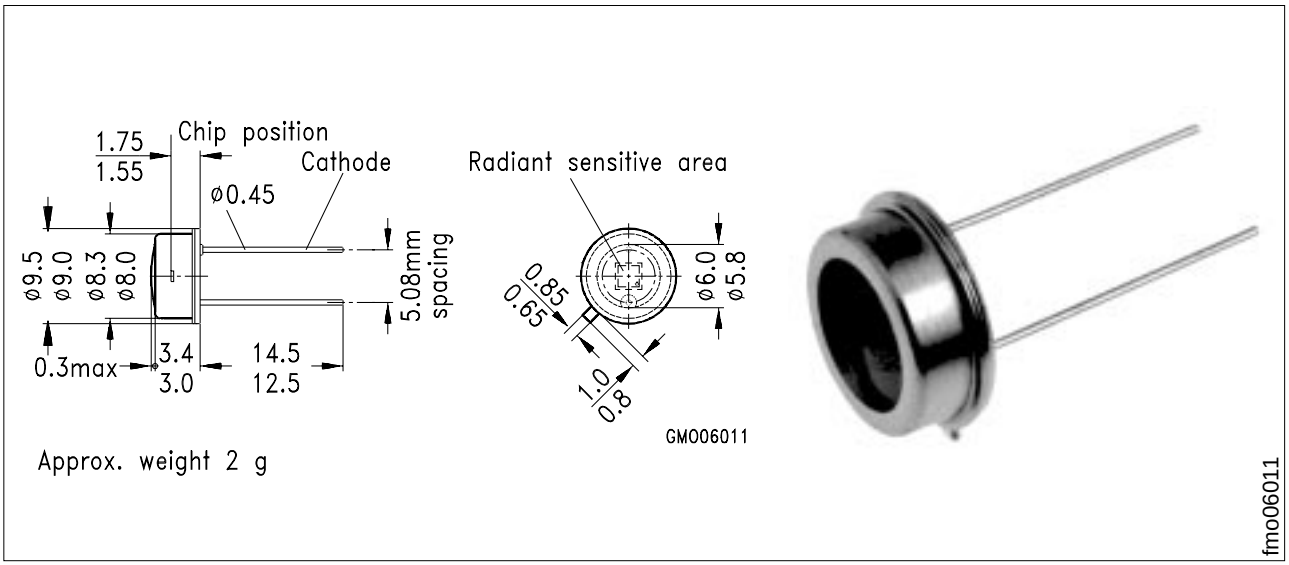
	Dessiné Gezeichnet	waltherj	07.06.2010	Echelle Massstab
	Contrôlé Geprüft			1:1

Fichier
Datei Y:\waltherj\plan_mecanique_TD_johan_walthert\Capteur_lumiere.idw

Hes·so VALAIS WALLIS

SIEMENS

Silizium-Fotodiode für den sichtbaren Spektralbereich — BPW 21
Silicon Photodiode for the visible spectral range



Maße in mm, wenn nicht anders angegeben/Dimensions in mm, unless otherwise specified.

Wesentliche Merkmale

- Speziell geeignet für Anwendungen im Bereich von 350 nm bis 820 nm
- Angepaßt an die Augenempfindlichkeit (V_λ)
- Hermetisch dichte Metallbauform (ähnlich TO-5)

Anwendungen

- Belichtungsmesser für Tageslicht
- Für Kunstlicht mit hoher Farbtemperatur in der Fotografie und Farbanalyse

Features

- Especially suitable for applications from 350 nm to 820 nm
- Adapted to human eye sensitivity (V_λ)
- Hermetically sealed metal package (similar to TO-5)

Applications

- Exposure meter for daylight
- For artificial light of high color temperature in photographic fields and color analysis

Typ Type	Bestellnummer Ordering Code
BPW 21	Q62702-P885

Grenzwerte
Maximum Ratings

Bezeichnung **Description**	**Symbol** **Symbol**	**Wert** **Value**	**Einheit** **Unit**
Betriebs- und Lagertemperatur Operating and storage temperature range	T_{op}; T_{stg}	– 40 ... + 80	°C
Löttemperatur (Lötstelle 2 mm vom Gehäuse entfernt bei Lötzeit $t \leq 3$ s) Soldering temperature in 2 mm distance from case bottom ($t \leq 3$ s)	T_S	235	°C
Sperrspannung Reverse voltage	V_R	10	V
Verlustleistung, T_A = 25 °C Total power dissipation	P_{tot}	250	mW

Kennwerte (T_A = 25 °C, Normlicht A, T = 2856 K)
Characteristics (T_A = 25 °C, standard light A, T = 2856 K)

Bezeichnung **Description**	**Symbol** **Symbol**	**Wert** **Value**	**Einheit** **Unit**
Fotoempfindlichkeit, V_R = 5 V Spectral sensitivity	S	10 (≥ 5.5)	nA/lx
Wellenlänge der max. Fotoempfindlichkeit Wavelength of max. sensitivity	$\lambda_{S\,max}$	550	nm
Spektraler Bereich der Fotoempfindlichkeit S = 10 % von S_{max} Spectral range of sensitivity S = 10 % of S_{max}	λ	350 ... 820	nm
Bestrahlungsempfindliche Fläche Radiant sensitive area	A	7.34	mm^2
Abmessung der bestrahlungsempfindlichen Fläche Dimensions of radiant sensitive area	$L \times B$ $L \times W$	2.73 × 2.73	mm × mm
Abstand Chipoberfläche zu Gehäuseoberfläche Distance chip front to case surface	H	1.9 ... 2.3	mm
Halbwinkel Half angle	φ	± 55	Grad deg.

Kennwerte (T_A = 25 °C, Normlicht A, T = 2856 K)
Characteristics (T_A = 25 °C, standard light A, T = 2856 K) (cont'd)

Bezeichnung **Description**	**Symbol** **Symbol**	**Wert** **Value**	**Einheit** **Unit**
Dunkelstrom Dark current V_R = 5 V V_R = 10 mV	 I_R I_R	 2 (≤ 30) 8 (≤ 200)	 nA pA
Spektrale Fotoempfindlichkeit, λ = 550 nm Spectral sensitivity	S_λ	0.34	A/W
Quantenausbeute, λ = 550 nm Quantum yield	η	0.80	Electrons Photon
Leerlaufspannung, E_v = 1000 lx Open-circuit voltage	V_O	400 (≥ 320)	mV
Kurzschlußstrom, E_v = 1000 lx Short-circuit current	I_{SC}	10	µA
Anstiegs- und Abfallzeit des Fotostromes Rise and fall time of the photocurrent R_L = 1 kΩ; V_R = 5 V; λ = 550 nm; I_p = 10 µA	t_r, t_f	1.5	µs
Durchlaßspannung, I_F = 100 mA, E = 0 Forward voltage	V_F	1.2	V
Kapazität, V_R = 0 V, f = 1 MHz, E = 0 Capacitance	C_0	580	pF
Temperaturkoeffizient von V_O Temperature coefficient of V_O	TC_V	– 2.6	mV/K
Temperaturkoeffizient von I_{SC} Temperature coefficient of I_{SC}	TC_I	– 0.05	%/K
Rauschäquivalente Strahlungsleistung Noise equivalent power V_R = 5 V, λ = 550 nm	NEP	7.2×10^{-14}	$\frac{W}{\sqrt{Hz}}$
Nachweisgrenze, V_R = 5 V, λ = 550 nm Detection limit	D*	1×10^{12}	$\frac{cm \cdot \sqrt{Hz}}{W}$

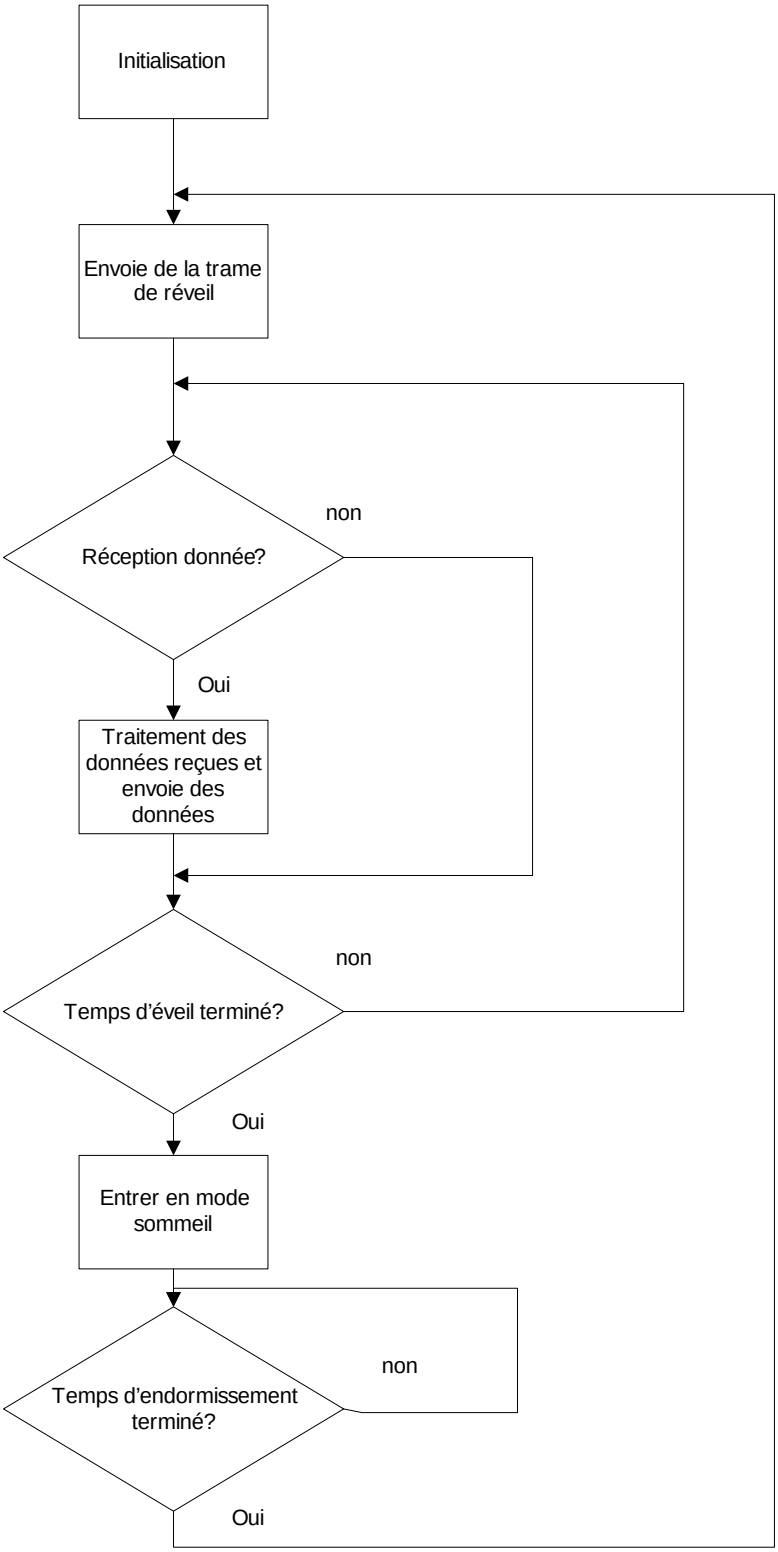
Annexe n°2

Carte présence

Contenu

- Structogramme
- Protocol de test
- Schémas électrique
- Schémas PCB et plan d’implantation
- Schémas mécanique
- Data sheet capteur

Structogramme carte de presence



Protocole de test

Capteur de présence

Test hardware (sans XBee)

Vérification visuelle des soudures et pistes : _____

Vérification avec la sonnette

Cour circuit GND, VCC _____

Cour circuit GND les autres pistes _____

Cour circuit VCC les autres pistes _____

Alimenter en 3.3V

Pine n°1 alimentation 3.3V _____

Le bouton reset fonctionne correctement Pull down _____

En alimentant (3V) la pine n°15 la led s'allume (! jumper) _____

Alimenter en 3V Pine n°19, capteur de mouvement couvert PT1=0V (!tempo) _____

Alimenter en 3V Pine n°19, capteur de mouvement découvert PT1=3V (!tempo) _____

Test softwear

Configuration superviseur

Commande	Valeur	Description	remarque
DL	00	Adresse basse su destinataire	Pas de destinataire uniquement récepteur
DH	00	Adresse haute du destinataire	Pas de destinataire uniquement récepteur
D5	01	DIO5 Configuration	DIO5 Assiciated Indication LED

Configuration end devise

Commande	Valeur	Description	remarque
DL	XX	Adresse basse su destinataire	Le coordinateur peut etre atteint avec l'adresse 0000
DH	XX	Adresse haute du destinataire	Le coordinateur peut etre atteint avec l'adresse 0000
SM	04	Sleep mode	Activation du sleep mode 0 desabled, 04 enabled
SP	01F4 = 5 sec	Sleep period	Période de sleep 20-AF0 20 = 320ms AF0 =28s
SN	04	Nbr. of sleep period	Nombre de période sleep 1-FFFF (65535)
ST	9C40 = 40 sec	Time Before sleep	Délais avant sleep 1-FFFF (65.5sec)
SO	06	Sleep option	00 pas activé 02 toujours se réveiller pour le ST Time 04 sleep le temps complet SN*SP 06 sleep le temps complet SN*SP+ toujours se réveiller pour le ST Time
JN	01	Join Notification	Envoie d'une notification au autres modules lors de l'enclenchement ou de l'association a un réseau
PR	1FF0	Pull Up resistor	Résistence pull up desactivée pour l'entrée digitale DIO1
D0	01	AD Configuration	Entrée digitale configurée comme commissioning bouton
D1	04	AD Configuration	00 désactivée 03 activée comme entrée 04 sortie digitale a 0
D2	03	AD Configuration	00 desactivée 02 analog input 03 activée comme entrée
D4	00	AD Configuration	00 désactivée 04 sortie digitale a 0 05 sortie digitale a 1
D5	01	DIO5 associated	
V+	955	Niveau Vcc	Défini le seuil de détection du niveau de tension faible pour l'envoie d'une alerte. 2.66V

1 Le XBee s'endors et se réveil selon le cycle _____ _____

2 La led d'association clignote lors de l'éveille et reste éteinte lors du sommeil _____

3 Lors du réveil par réception de la trame _____

le capteur nous retourne sa valeur _____

3.1 capteur caché trame _____ _____

3.2 capteur découvert trame _____ _____

4 avec le bouton reset le module se réveil _____

5 par l'activation (0V) du bouton commissioning le module se réveil _____

Consommation

Les mesures sont effectuées à l'oscilloscope

Courant consommé en mode sommeil _____

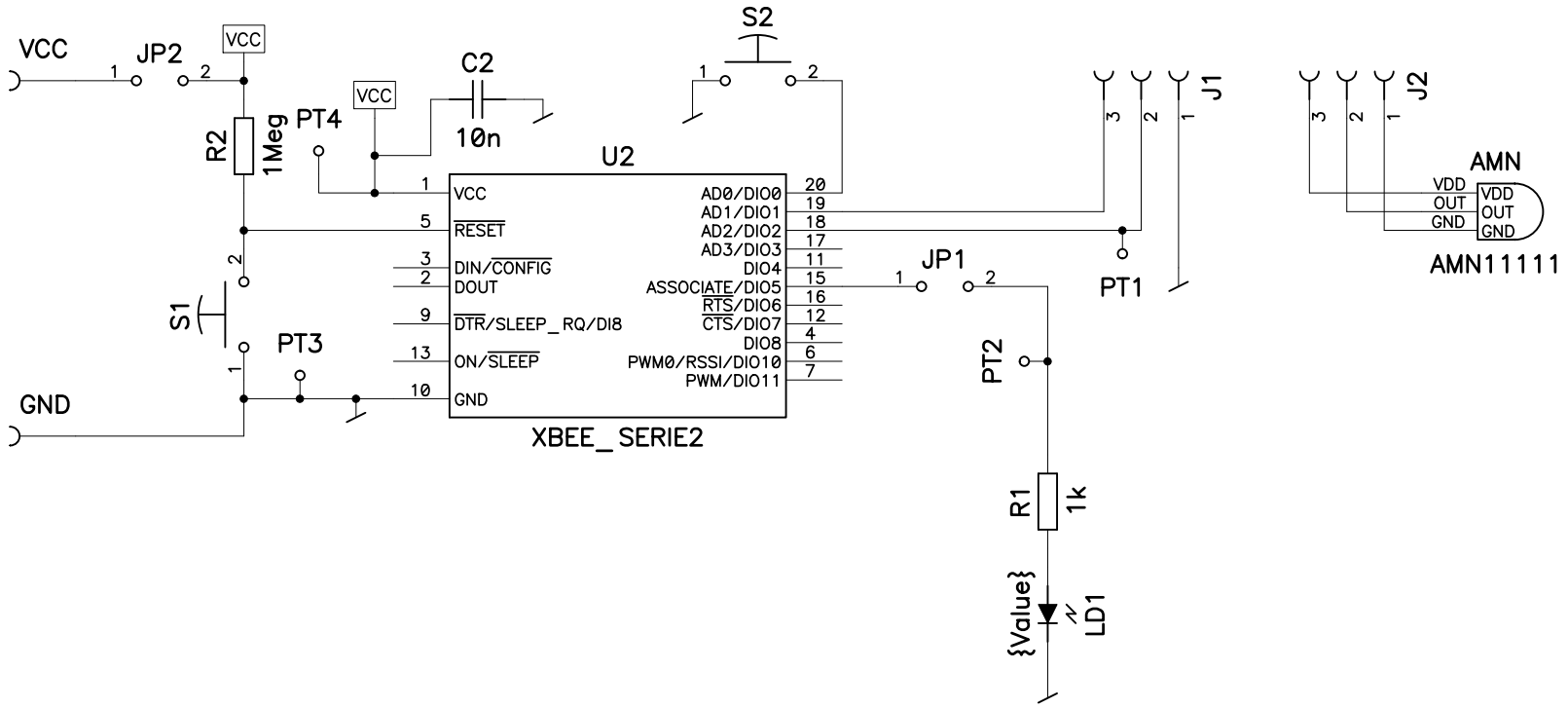
Courant consommé en mode éveil _____

Courant consommé pendant l'envoi d'une trame _____

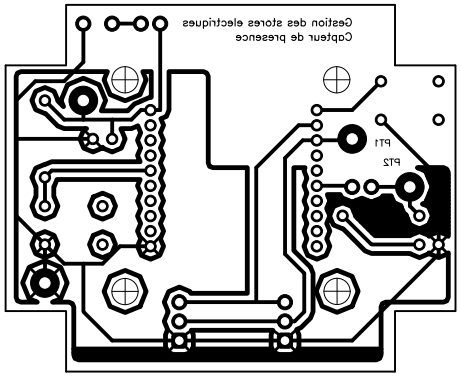
Test ok

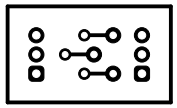
Sion, le _____

Walthert Johan



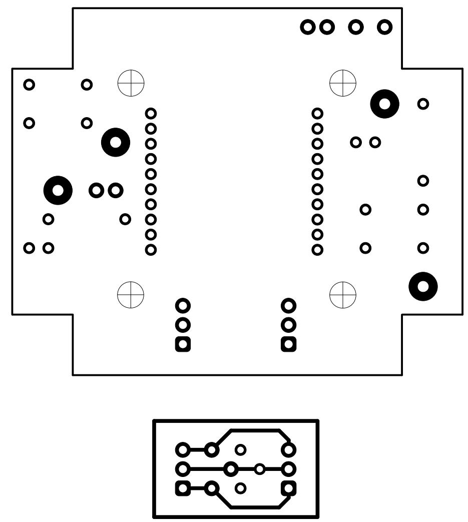
Gestion des stores electriques Capteur de presence Sheet1	DES	07.06.10Waltherj
	REV	V1.1
HAUTE ECOLE VALAISANNE	1/3	{Path} Capteur_de_presence.sch





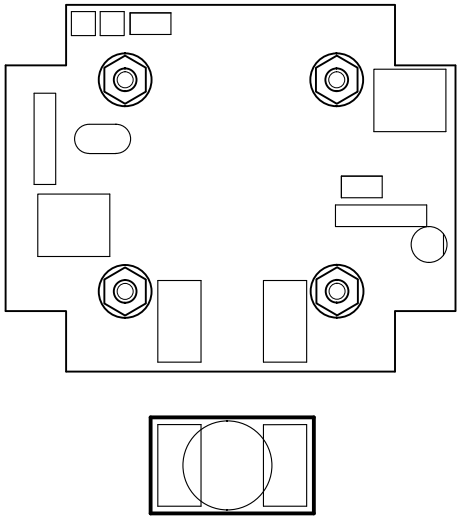
Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.80	+	39	Yes
0.90	X	3	Yes
1.00	Y	18	Yes
1.40	⊤	4	Yes
3.50	⧖	4	Yes

Gestion des stores electriques capteur de presence	DES	07.06.2010
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ capteur_ presence.pcb	



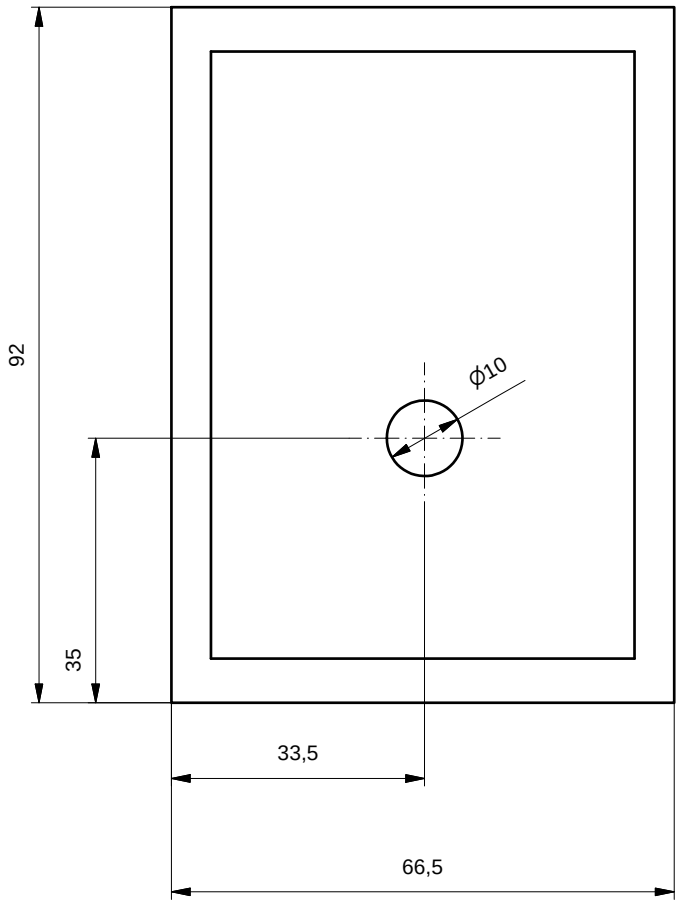
Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.80	+	39	Yes
0.90	X	3	Yes
1.00	Y	18	Yes
1.40	⊤	4	Yes
3.50	⧖	4	Yes

Gestion des stores electriques capteur de presence	DES	07.06.2010
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ capteur_presence.pcb	



Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.80	+	39	Yes
0.90	X	3	Yes
1.00	Y	18	Yes
1.40	⊤	4	Yes
3.50	⧖	4	Yes

Gestion des stores electriques capteur de presence	DES	07.06.2010
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ capteur_ presence.pcb	



le perçage est à effectuer du côté où la face est ouverte

	Dessiné Gezeichnet	waltherj	07.06.2010	Echelle Massstab
	Contrôlé Geprüft			1:1

Fichier
Datei Y:\waltherj\plan_mecanique_TD_johan_walthert\Capteur_presence.idw

Hes·so VALAIS WALLIS

Panasonic
ideas for life

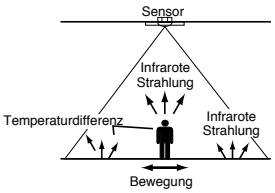
BEWEGUNGSSENSOR (PASSIV INFRAROT)

NaPiOn

SUBMINIATURISIERTER BEWEGUNGSSENSOR

Was bedeutet „passiver Infrarottyp“?

Diese Art von Sensor erkennt Veränderungen der infraroten Strahlung (also Wärmestrahlung), die sich aufgrund von Bewegungen von Personen (oder Objekten) ergibt, also eine zeitliche Veränderung des Temperaturgradienten im Messfeld. Aufgrund der immer vorhandenen Körper(-wärme)strahlung eignet sich dieser Sensor bestens zur Detektion von Personen.



Applikationen

Bewegungssensoren werden häufig zur Energieeinsparung bei z.B. Klimaanlagen, Lichtquellen, Monitoren und Ventilatoren eingesetzt. Ebenso dienen sie zur Raum- bzw. Außenbereichüberwachung. Die Serie NaPiOn zeichnet sich neben der Zuverlässigkeit auch durch die kleine Bauweise aus. Eine nahezu unsichtbare Integration ist möglich.

Besonderheiten

Weltweit kleinster Bewegungssensor mit integriertem Verstärker

Äußerst kompakt und daher auch zum direkten Bestücken auf Platinen geeignet.

Erhältlich in zwei Linsenfarben

Durch die ultraminiaturisierte Bauform und den beiden Linsenfarben (weiß und anthrazit) kann der Sensor unauffällig integriert werden.

Lieferbar mit digitalem oder analogem Ausgang

Integrierter Verstärker

Mit Hilfe des eingebauten Verstärkers kann der Sensor direkt an weitere Schaltungen oder µController angeschlossen werden.

Detektiert kleinste Bewegungen

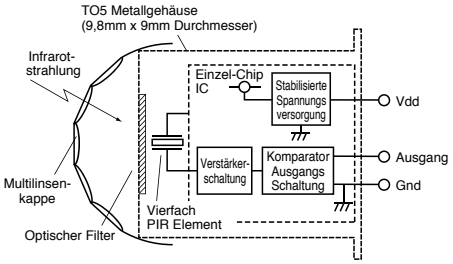
Auch kleinste Körperbewegungen werden erkannt. Der Typ für kleine Bewegungen registriert bereits Handbewegungen ab 20 cm.

Erhöhte Störunempfindlichkeit

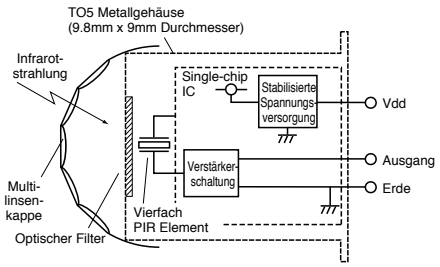
Das TP5-Gehäuse ermöglicht eine um mehr als zweimal höhere Störunempfindlichkeit als herkömmliche Typen.

Blockdiagramme

Digitalversion



Analogversion



Allgemeine Eigenschaften

max. Spannungsversorgung		-0,3 bis 7 V DC
Umgebungstemperatur	**Betrieb**	-20 bis 60 °C (ohne Eis- und Kondensbildung)
	Lagerung	-20 bis 70°C

Hinweis:
• Alle Angaben für Umgebungstemperatur von 25°C und der empfohlenen Betriebsspannung.

Elektrische Eigenschaften

Digitalversion

			Symbol	Wert		Hinweise
				Standardtyp	Stromspartyp	
empf. Betriebsspannung		**minimal typisch maximal**	**Vdd**	3,0 V DC - 6,0 V DC	2,2 V DC - 3,0 V DC	
Stromverbrauch (Standby) (*1)		**typisch maximal**	**Iw**	170 µA 300 µA	46 µA 60 µA	Iout = 0 mA
Ausgang (bei Detektion)	**Strom**	**maximal**	**Iout**	100 µA		Vout ≥ Vdd - 0,5 V
	Spannung	**minimal maximal**	**Vout**	Vdd - 0,5 V -		Offen, wenn kein Objekt erkannt wird
Einschaltzeit (Einschwingzeit)		**typisch maximal**	**Twu**	7 s 30 s		

Hinweis:
(*1) Der Stromverbrauch errechnet sich aus dem Verbrauch im Standbymodus und dem Ausgangsstrom.

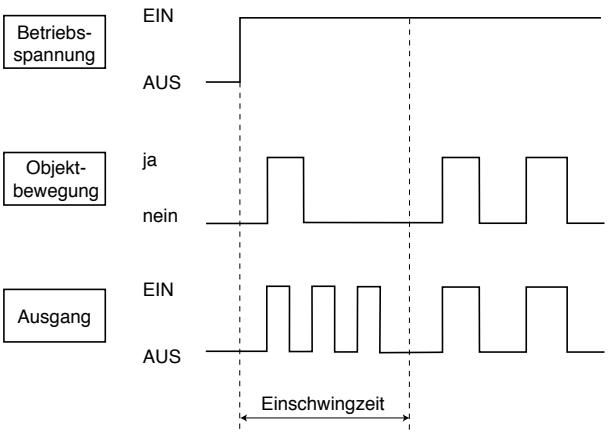
Analogversion

			Symbol	Wert	Hinweise
empf. Betriebsspannung		**minimal maximal**	**Vdd**	4,5 V DC 5,5 V DC	
Stromverbrauch (*1)		**typisch maximal**	**Iw**	170 µA 300 µA	Iout = 0 mA
Ausgang	**Strom**	**maximal**	**Iout**	50 µA	Vout ≥ Vdd - 0,5 V
	Spannung (ohne Detektion)	**minimal typisch maximal**	**Voff**	2,3 V 2,5 V 2,7 V	Spannungsbereich, wenn kein Objekt erkannt wird
	Spannung (bei Detektion)	**minimal typisch maximal**	**Vout**	0 V 2,5 V Vdd	
	Rauschen	**typisch maximal**	**Vn**	130 mVpp 300 mVpp	Spannungsrauschen im Betriebszustand
Einschaltzeit (Einschwingzeit)		**maximal**	**Twu**	45 s	

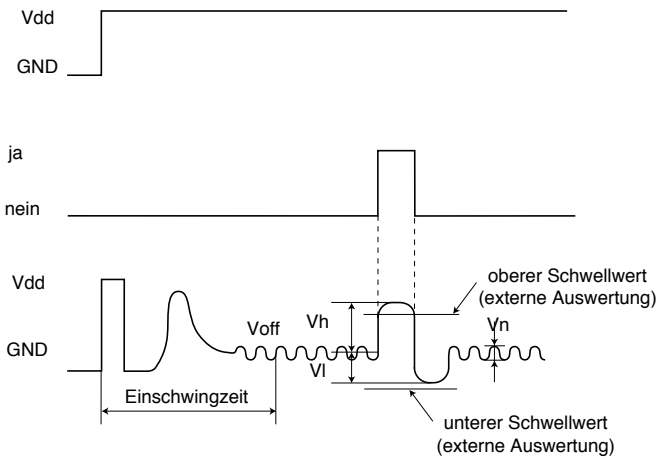
Hinweis:
• Soll der Analogtyp wie die Digitalversion betrieben werden, müssen die Schwellwerte bei Vdd ± 0,45 V gelegt werden (z.B. > 2,95V oder < 2.05V).

Zeitdiagramm

Digitalversion



Analogversion



Hinweis:
• Während der Einschwingzeit hat der Ausgang (analog oder digital) keinen definierten Zustand (30s bei der Digital- und 45s bei der Analogversion)

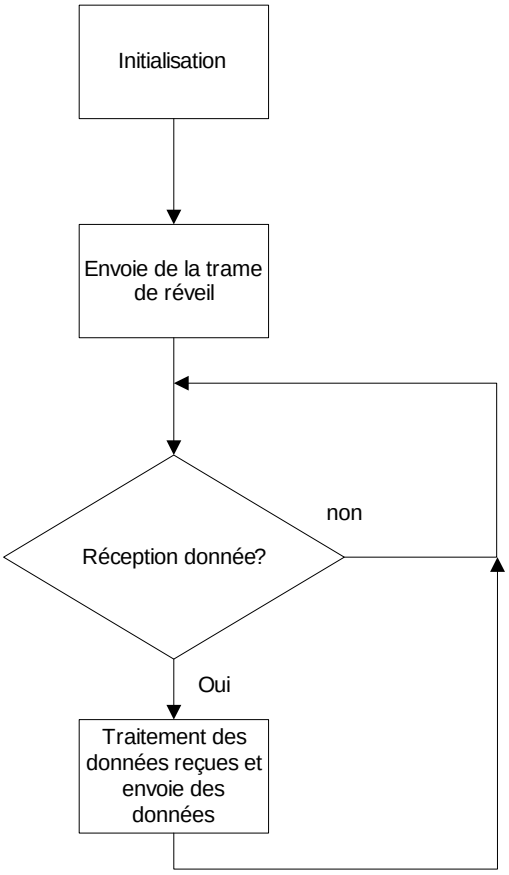
Annexe n°3

Carte cmd_moteur

Contenu

- Structogramme
- Protocol de test
- Schémas électrique
- Schémas PCB et plan d'implantation
- Schémas mécanique

Structogramme carte Cmd_moteur



Protocole de test

Commande moteur

Test hardware (sans XBee)

Vérification visuelle des soudures et pistes : _____

Vérification avec la sonnette

Cour circuit GND, VCC _____

Cour circuit GND les autres pistes _____

Cour circuit VCC les autres pistes _____

Alimenter en 3.3V (transformateur) _____

Pine n°1 alimentation 3.3V _____

Le bouton reset fonctionne correctement Pull down _____

En alimentant (3V) la pine n°15 la led s'allume (! jumper) _____

Alimenter en 3V PT1, le relais de montée tire _____

Alimenter en 3V PT2, le relais de descente tire _____

Test softwear

Configuration superviseur

Commande	Valeur	Description	remarque
DL	00	Adresse basse su destinataire	Pas de destinataire uniquement récepteur
DH	00	Adresse haute du destinataire	Pas de destinataire uniquement récepteur
D5	01	DIO5 Configuration	DIO5 Assiciated Indication LED

Configuration end device

Commande	Valeur	Description	remarque
DL	XX	Adresse basse su destinataire	Le coordinateur peut etre atteint avec l'adresse 0000
DH	XX	Adresse haute du destinataire	Le coordinateur peut etre atteint avec l'adresse 0000
SM	01	Sleep mode	Activation du sleep mode 0 desabled, 01 entrée sleep DI08
SP	XX	Sleep period	Période de sleep 20-AF0 20 = 320ms AF0 =28s
SN	XX	Nbr. of sleep period	Nombre de période sleep 1-FFFF (65535)
ST	XX	Time Before sleep	Délais avant sleep 1-FFFF (65.5sec)
SO	00	Sleep option	00 pas activé 02 toujours se réveiller pour le ST Time 04 sleep le temps complet SN*SP 06 sleep le temps complet SN*SP+ toujours se réveiller pour le ST Time
JN	01	Join Notification	Envoie d'une notification au autres modules lore de l'enclenchement ou de l'association a un réseau
PR	1FF0	Pull Up resistor	Résistence pull up desactivée pour l'entrée digitale DIO1
D0	01	AD Configuration	Entrée digitale configurée comme commissioning bouton
D1	04	AD Configuration	00 désactivée 03 activée comme entrée 04 sortie digitale a 0
D2	04	AD Configuration	00 desactivée 02 analog input 04 sortie digitale a 0
D4	00	AD Configuration	00 désactivée 04 sortie digitale a 0 05 sortie digitale a 1
D5	01	DIO5 associated	

1 Le XBee reset réveillé _____

2 La led d'association clignote en permanence _____

3 par réception de la trame _____

le relais « monter » est activé _____

4 par réception de la trame _____

le relais « descendre » est activé _____

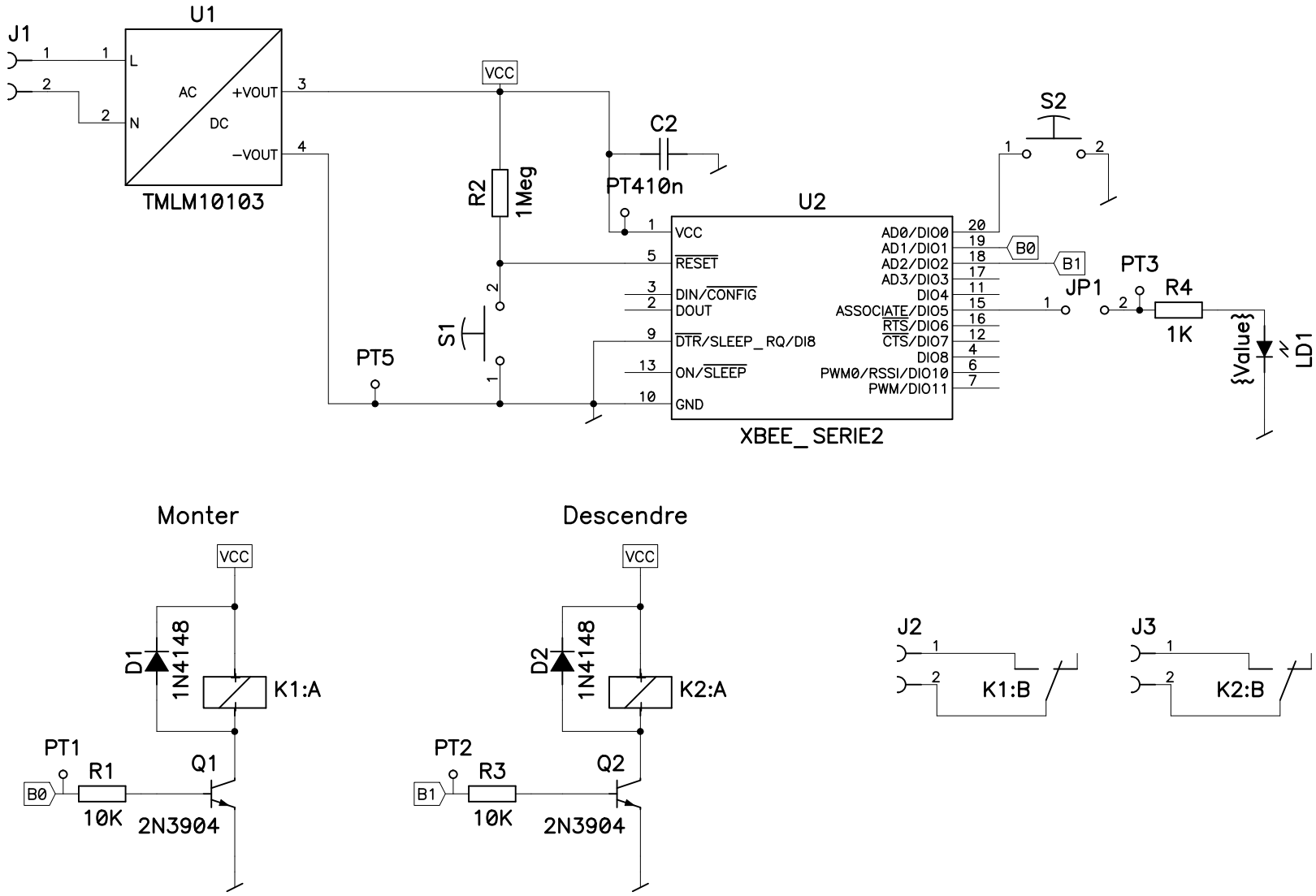
5 avec le bouton reset le module se réveil _____

6 par l'activation (0V) du bouton commissioning le module se réveil _____

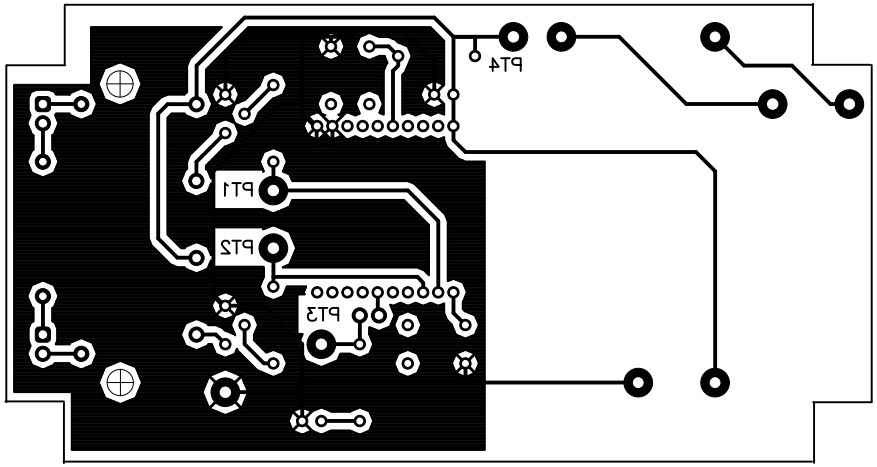
Test ok

Sion, le _____

Walthert Johan



Gestion des stores electriques commande moteur Sheet1	DES	06.04.10Waltherj
	REV	V1.1
HAUTE ECOLE VALAISANNE	1/3	{Path} cmd_moteur.sch

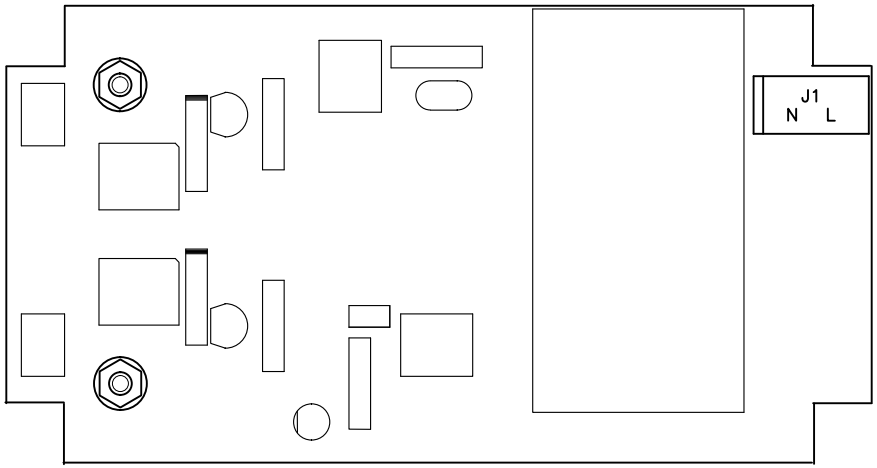


Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.80	+	46	Yes
0.90	X	4	Yes
1.00	Y	10	Yes
1.30	⊤	4	Yes
1.40	⧖	7	Yes
3.50	⋈	2	Yes

Gestion des stores electriques	DES	Date
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ cmd_ moteur.PCB	

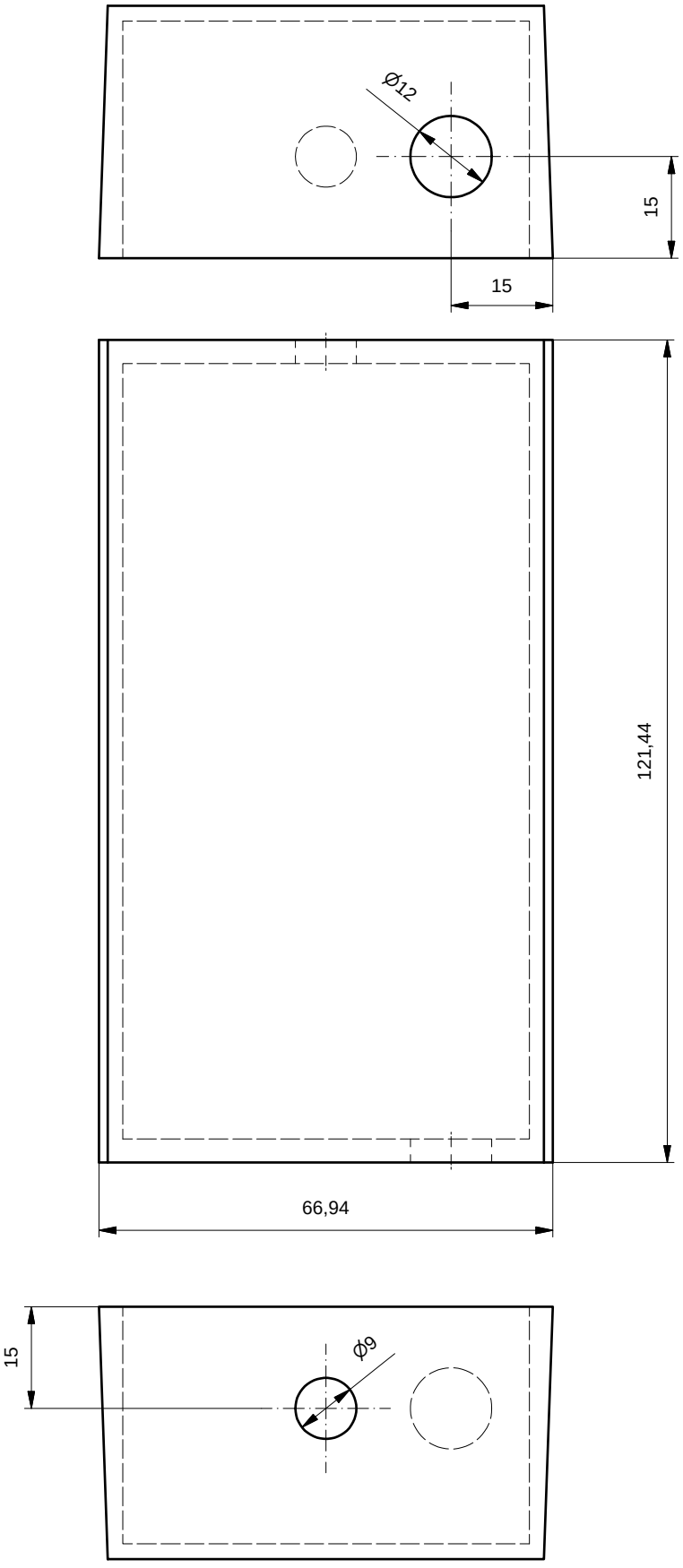
Drill table				
Hole	Dia (mm)	Symbol	Quantity	Plated
M	3.2	+	48	yes
N	2.9	×	4	yes
P	1.0	Y	16	yes
R	1.3	+	4	yes
S	1.4	⊠	7	yes
T	2.3	⊠	2	yes

Gestion des erreurs des moteurs	DES	date
	VER	1.14
HAUTE ECOLE VALAISANNE	PCB\ cmd_moteur_PCB	



Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.80	+	46	Yes
0.90	X	4	Yes
1.00	Y	10	Yes
1.30	⊤	4	Yes
1.40	⧖	7	Yes
3.50	⋈	2	Yes

Gestion des stores electriques	DES	Date
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ cmd_ moteur.PCB	



	Dessiné Gezeichnet	waltherj	07.06.2010	Echelle Massstab
	Contrôlé Geprüft			1:1

Fichier
Datei Y:\waltherj\plan_mecanique_TD_johan_walthert\Cmd_moteur.idw

Hes·so VALAIS WALLIS

Annexe n°4

Carte cmd_principal

Contenu

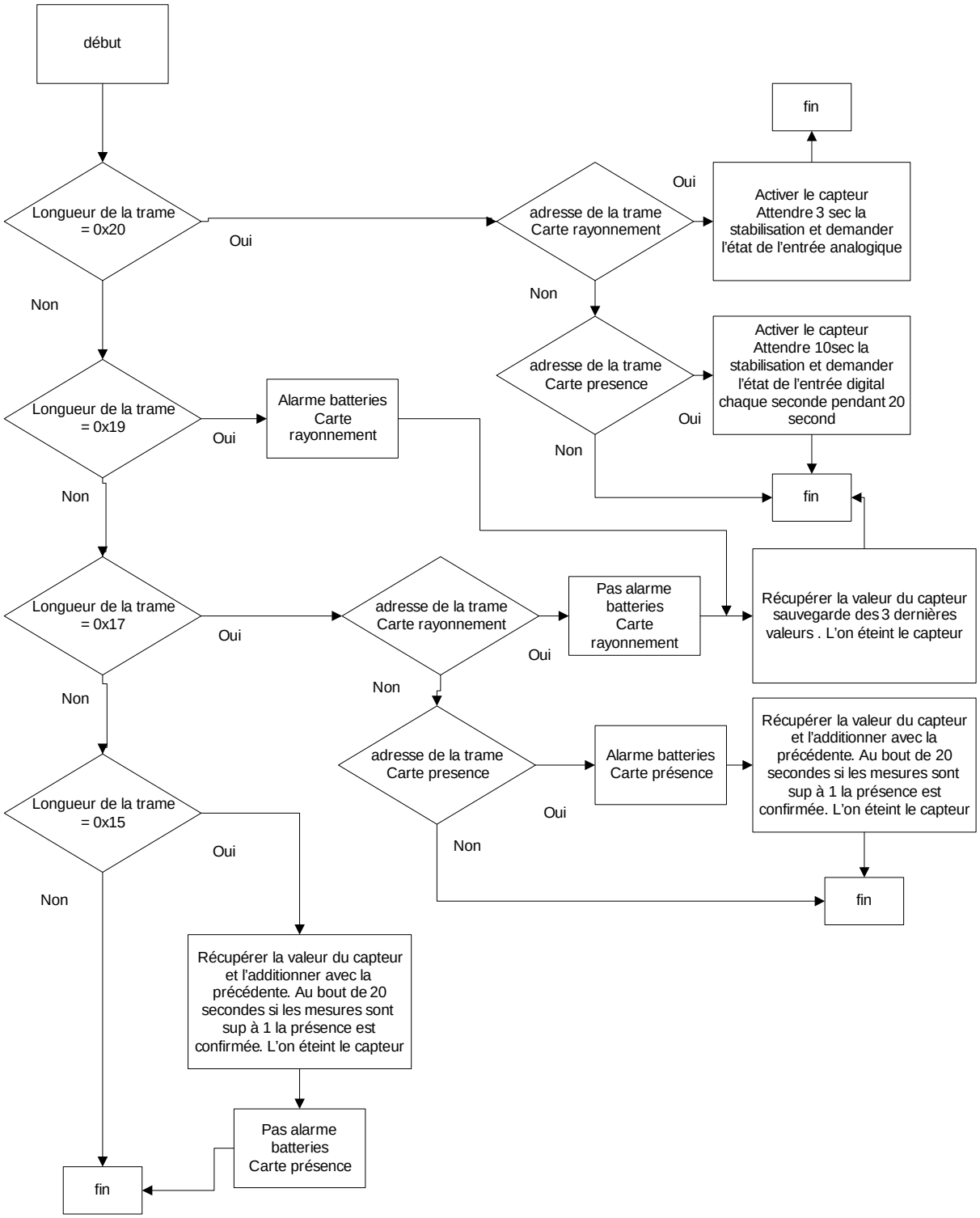
- Structogramme
- Protocol de test
- Schémas électrique
- Schémas PCB et plan d'implantation
- Schémas mécanique
- Code

Structogramme traitement paquets radio

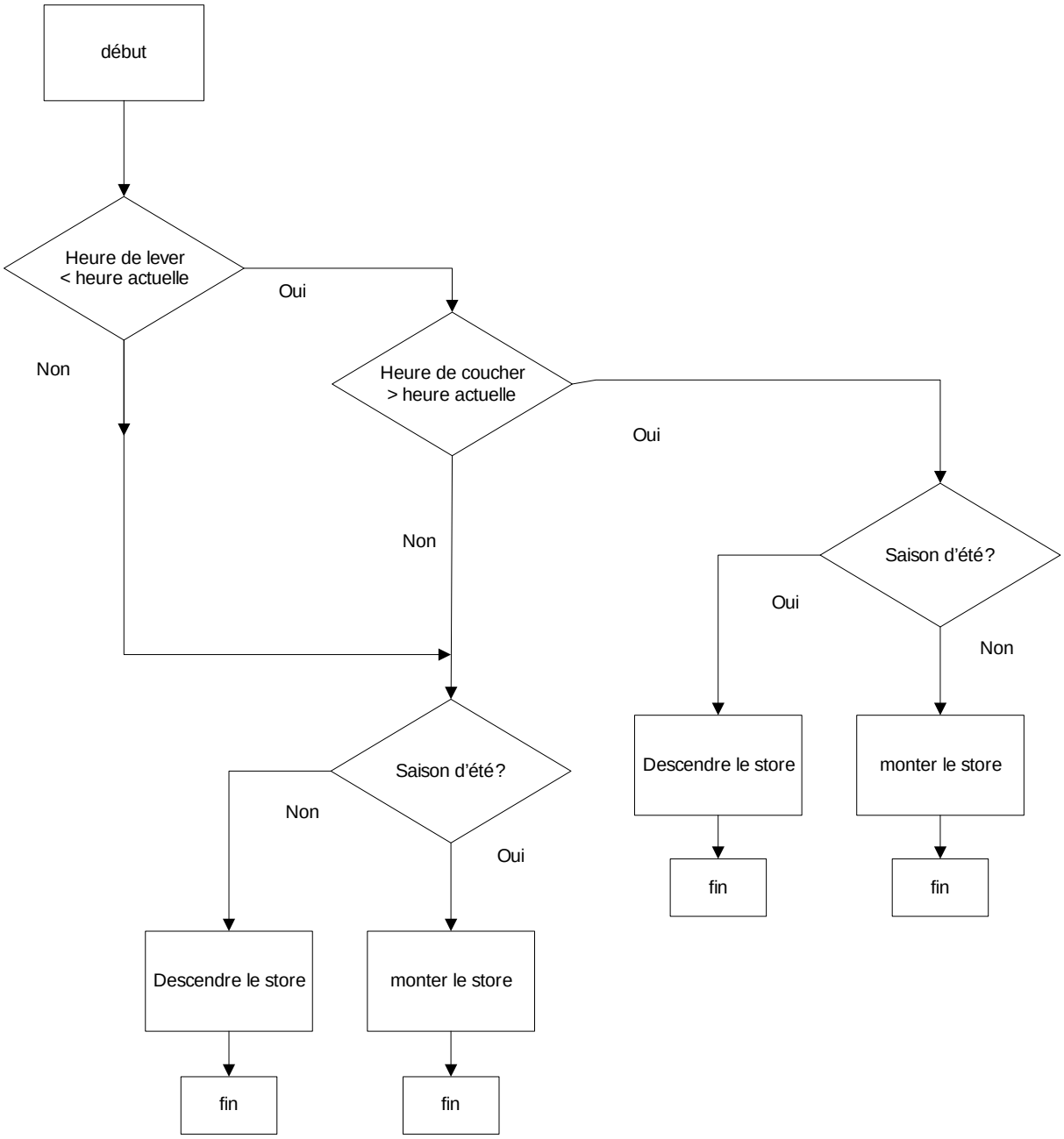
Par réception d'un paquet radio l'interruption du module uart enregistre le paquet reçu dans un rolling buffer de 10 lignes.

L'envoi de trame se fait également par le module uart avec un tableau de références pour les paquets à envoyer.

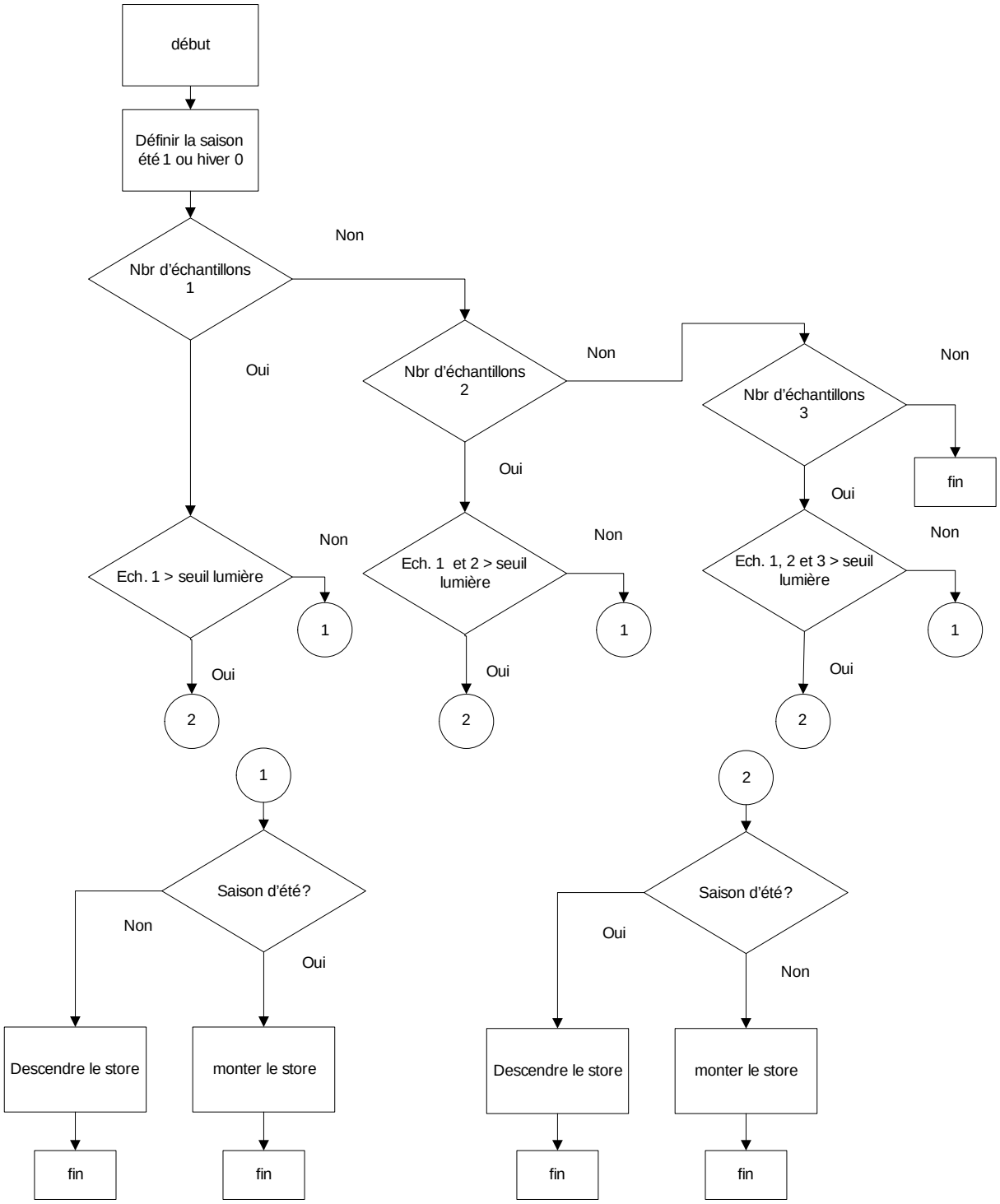
Le traitement des paquets reçus s'effectue quand les pointeurs de lecture et d'écriture ne sont pas identiques. Selon le schéma si dessous



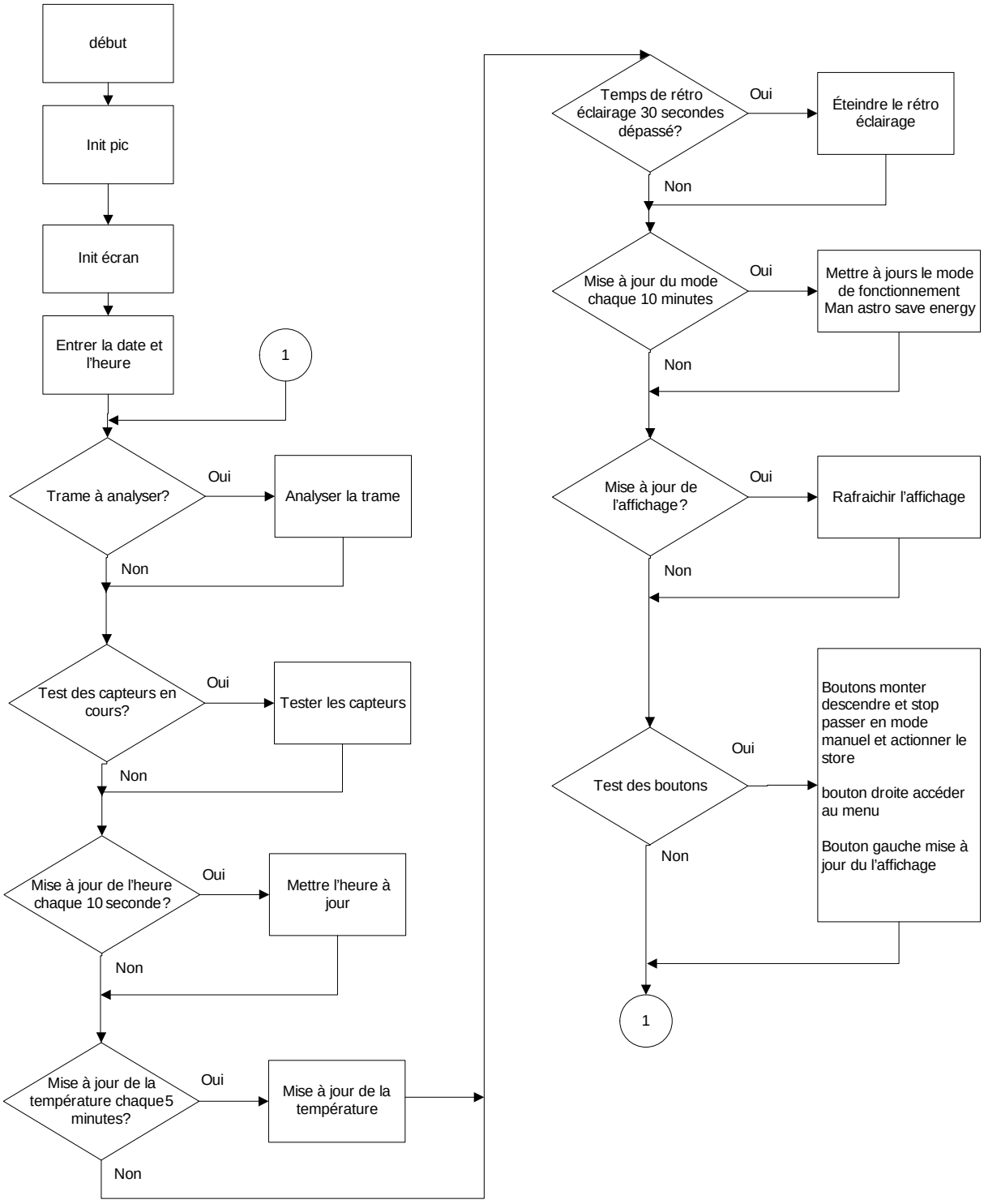
Structogramme mode astro



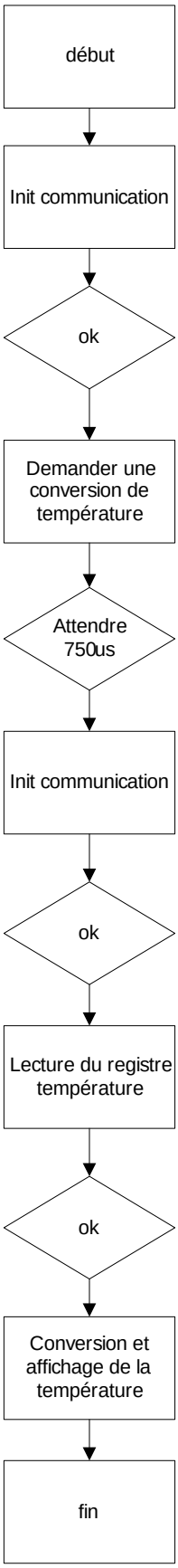
Structogramme mode save energy



Structogramme boucle principale

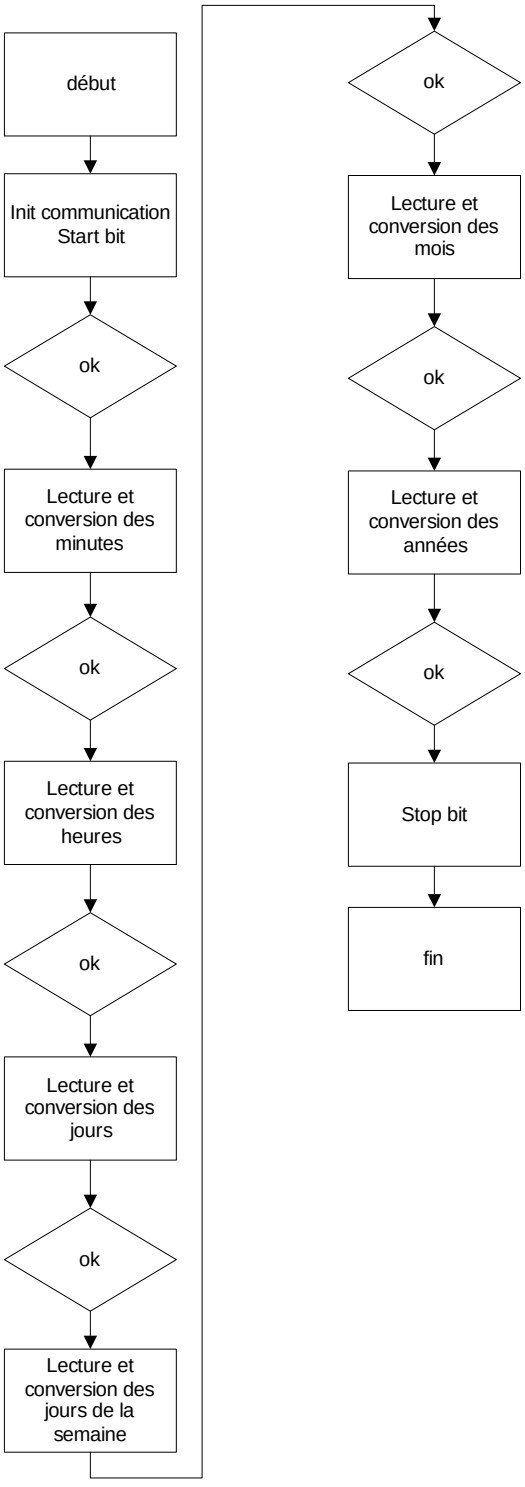


Structogramme température

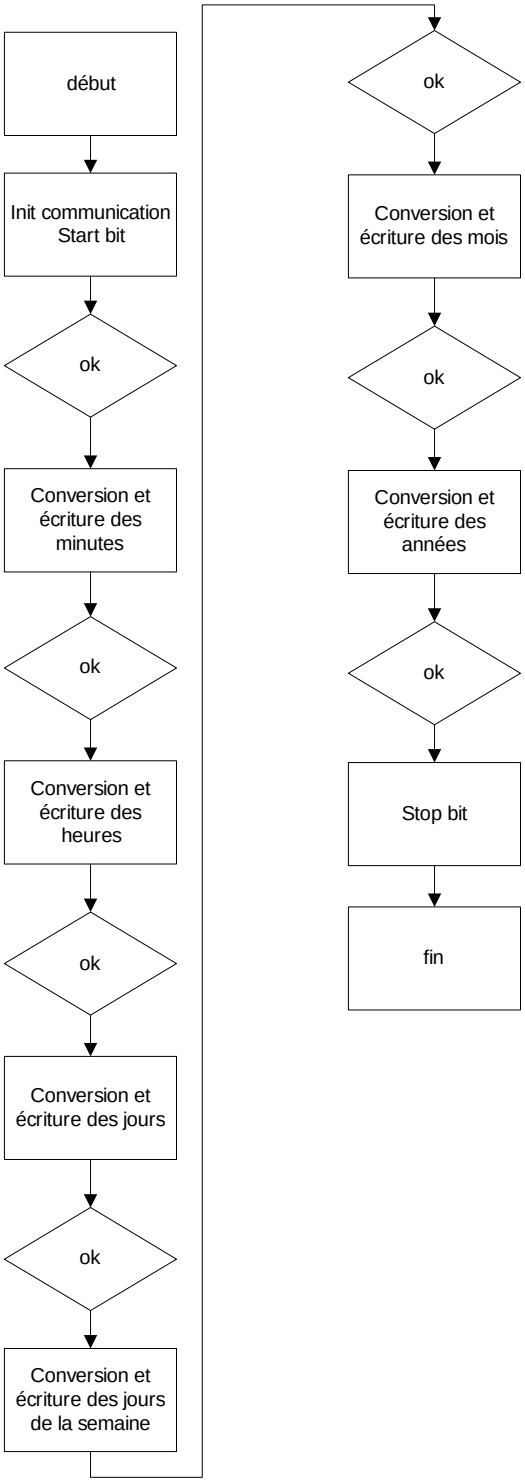


Structogramme horloge temps réelle

Lecture



écriture



Protocole de test

Commande principale

Test hardware (sans les composants)

Vérification visuelle des soudures et pistes : _____

Vérification avec la sonnette

Cour circuit GND, VCC _____

Cour circuit GND les autres pistes _____

Cour circuit VCC les autres pistes _____

Alimenter en 3.3V (Vin varie entre 3V et 12V) _____

Test hardware (avec les composants)

Configuration XBee superviseur

Commande	Valeur	Description	remarque
DL	00	Adresse basse su destinataire	Pas de destinataire uniquement récepteur
DH	00	Adresse haute du destinataire	Pas de destinataire uniquement récepteur
D5	01	DIO5 Configuration	DIO5 Assiciated Indication LED

1 Le XBee reset réveillé _____

2 La led d'association clignote en permanence _____

3 La liaison UART entre le module XBee et le uC fonctionne _____

4 La liaison SPI avec l'écran fonctionne _____

5 La liaison I2C avec l'horloge temps réel fonctionne _____

6 Les signaux des boutons arrivent correctement sur le uC _____

7 La led rouge est pilotable par la sortie du uC _____

8 Le rétro éclairage de l'écran est pilotable par la sortie du uC _____

Test softwear

Mode manuel

Par pression sur le bouton Up une trame est envoyée par RF _____

Par pression sur le bouton Down une trame est envoyée par RF _____

Par pression sur le bouton Ok une trame est envoyée par RF _____

Par liaison avec la carte Cmd_moteur les actions sur les boutons Up, Down,

stop activent les relais monter et descendre en fonction de l'action désirée. _____

L'affichage sur l'écran est correcte (en entrant et sortant du mode) _____

Mode automatique ASTRO

Le paramètre fonctionnement été est modifiable _____

Le paramètre position intermédiaire est modifiable _____

Si une présence est détectée le système passe en manuel _____

A l'aide d'un break point en modifiant l'heure et la date on passe par toutes les cases du tableau des heures de références. Avec la fonction normal _____

A l'aide d'un break point en modifiant l'heure et la date on passe par toutes les cases du tableau des heures de références. Avec la fonction inverse _____

Par liaison avec la carte Cmd_moteur en modifiant la date et l'heure, et en lançant la fonction astro

les relais « monter » et « descendre » en fonction de l'action désirée sont activés.

_____

Mode automatique Save Energie

Le paramètre seuil de luminosité est modifiable _____

Le paramètre nombre d'échantillons est modifiable _____

Si une présence est détectée le système passe en manuel _____

La présence n'est plus détectée le système revient en save energy _____

En hiver quand le seuil de luminosité est atteint le store monte, quand ce seuil est plus bas le store descend. _____

En été quand le seuil de luminosité est atteint le store descend, quand ce seuil est plus bas le store monte. _____

Le nombre d'échantillons permet de réaliser une temporisation au mouvement _____

Menu

Tous les paramètres sont modifiables _____

Le menu est accessible depuis la page principale _____

La navigation dans le menu est possible dans tous les sens _____

Erreurs

La led rouge est allumée quand une erreur est détectée _____

L'erreur batterie faible carte présence est détectée _____

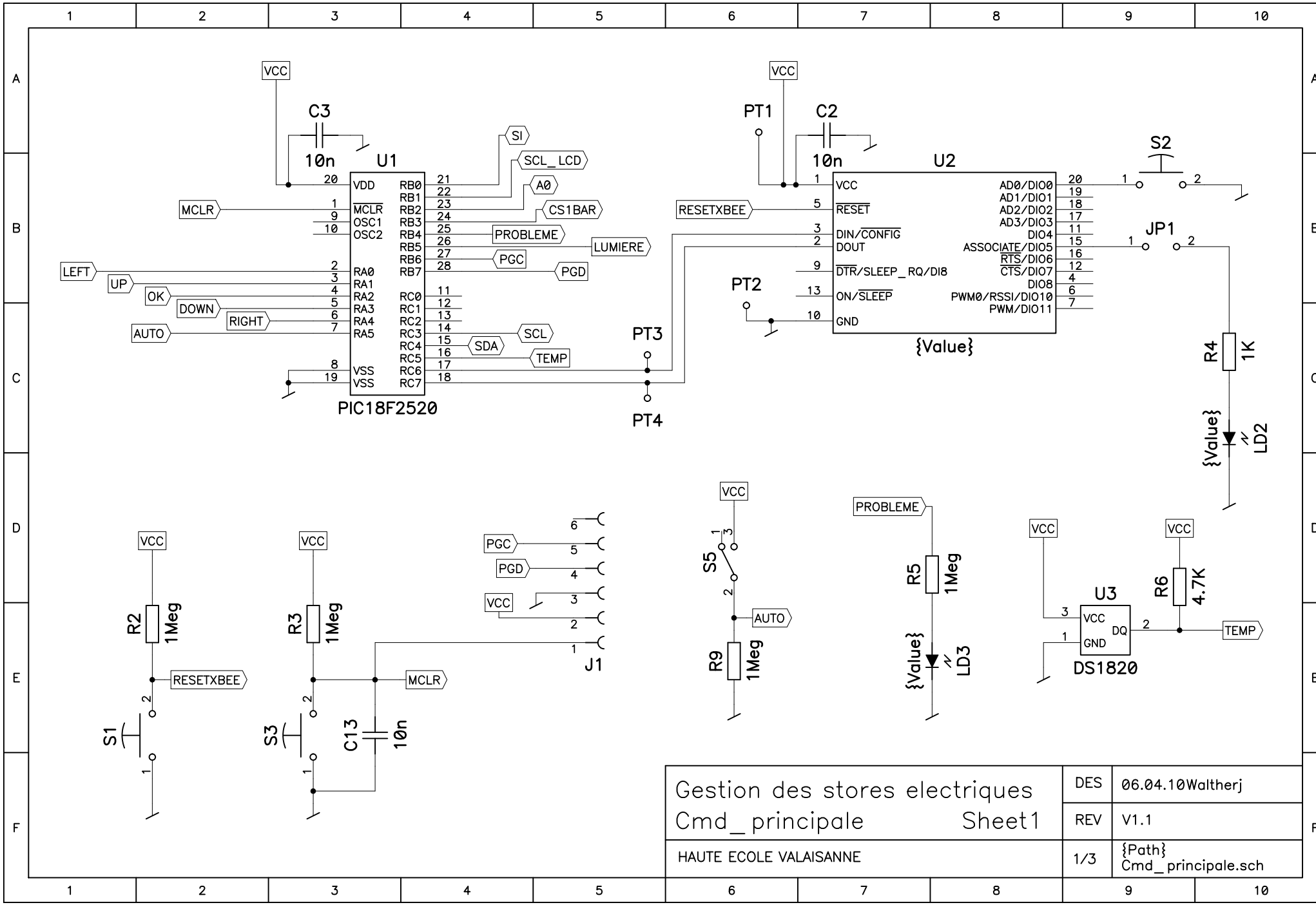
L'erreur batterie faible carte rayonnement est détectée _____

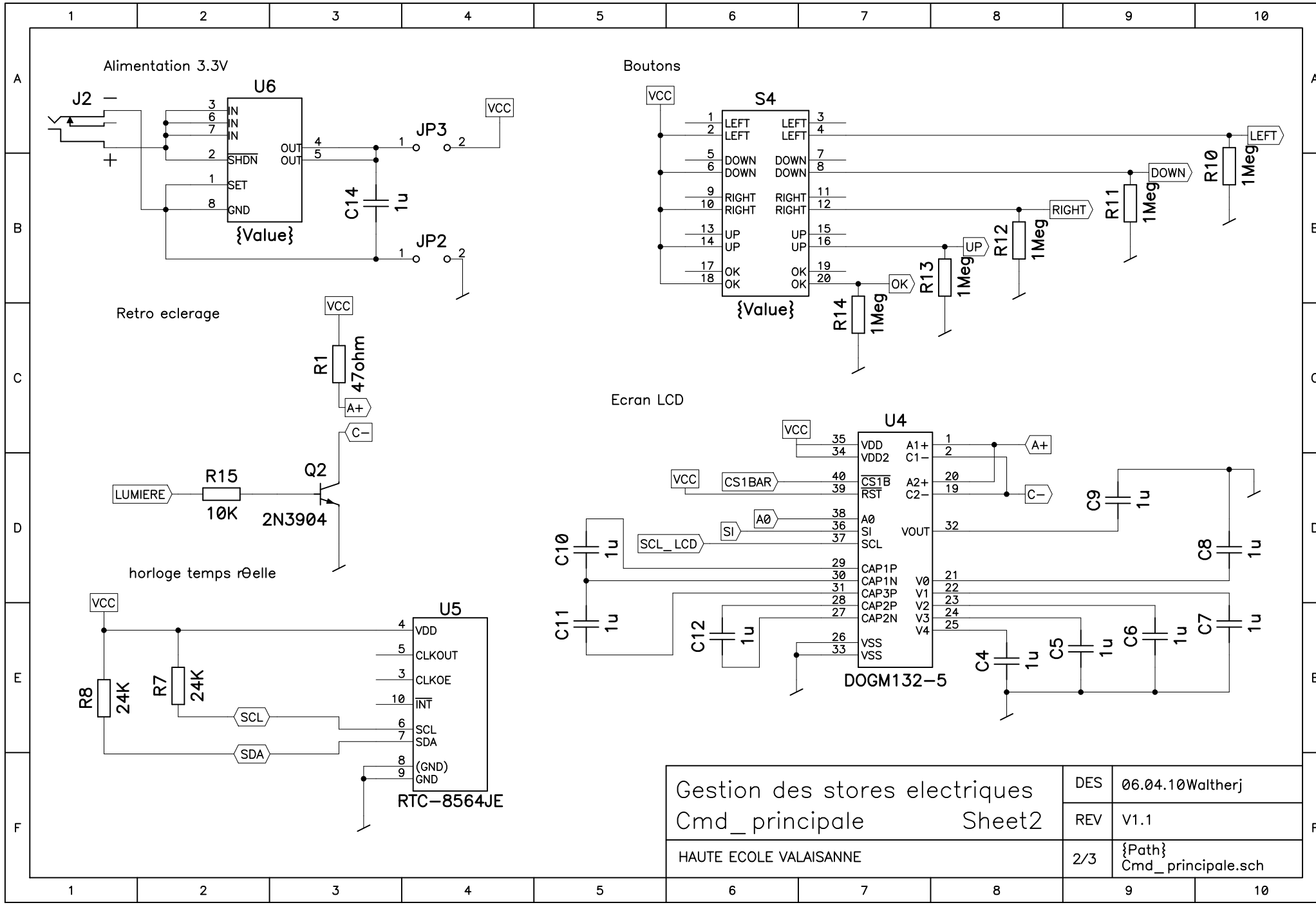
L'erreur de communication avec la Cmd_moteur est détectée _____

Test ok

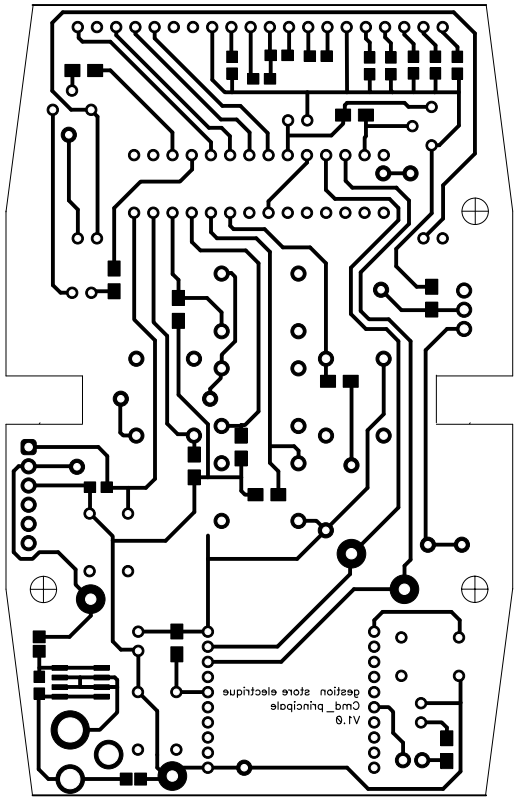
Sion, le _____

Walthert Johan


VCC
C3
10n
U1
SI
SCL_LCD
A0
CS1BAR
PROBLEME
LUMIERE
PGC
PGD
SCL
SDA
TEMP
MCLR
LEFT
UP
OK
DOWN
RIGHT
AUTO
VDD
MCLR
OSC1
OSC2
RA0
RA1
RA2
RA3
RA4
RA5
RB0
RB1
RB2
RB3
RB4
RB5
RB6
RB7
RC0
RC1
RC2
RC3
RC4
RC5
RC6
RC7
VSS
VSS
PIC18F2520
PT1
C2
U2
RESETXBEE
DIN/CONFIG
DOUT
DTR/SLEEP_RQ/DI8
ON/SLEEP
GND
AD0/DIO0
AD1/DIO1
AD2/DIO2
AD3/DIO3
DIO4
ASSOCIATE/DIO5
RTS/DIO6
CTS/DIO7
DIO8
PWM0/RSSI/DIO10
PWM/DIO11
{Value}
PT2
PT3
PT4
S2
JP1
R4
1K
{Value}
LD2
R2
1Meg
RESETXBEE
S1
R3
1Meg
MCLR
S3
C13
10n
J1
S5
AUTO
R9
1Meg
R5
1Meg
{Value}
LD3
U3
DQ
DS1820
R6
4.7K
Gestion des stores electriques
Cmd_principale
Sheet1
HAUTE ECOLE VALAISANNE
DES
06.04.10Waltherj
REV
V1.1
1/3
{Path}
Cmd_principale.sch



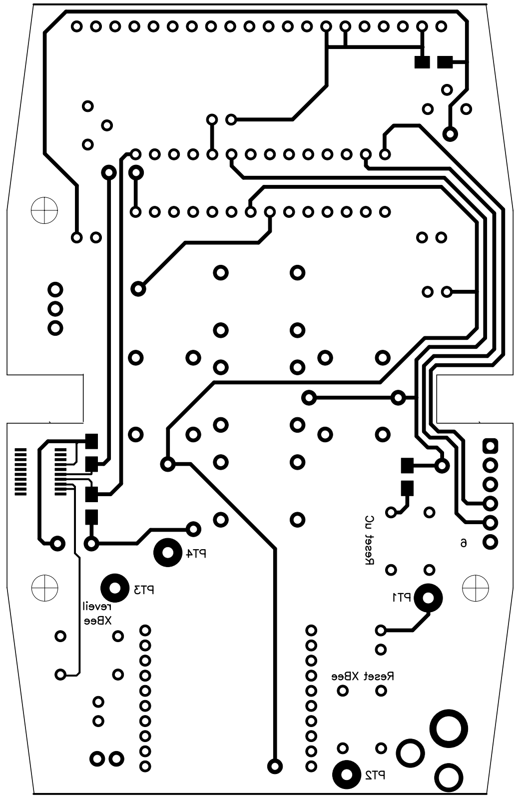
Gestion des stores electriques Cmd_principale Sheet2	DES: 06.04.10Waltherj REV: V1.1
HAUTE ECOLE VALAISANNE	2/3 {Path} Cmd_principale.sch



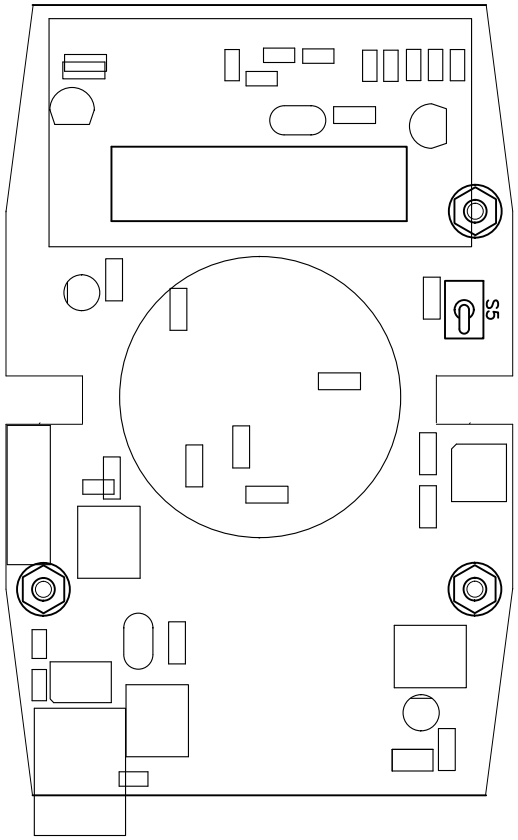
Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.80	+	70	Yes
0.81	×	28	Yes
0.90	Y	12	Yes
1.00	⊤	28	Yes
1.10		3	Yes
1.40	⧖	4	Yes
2.60	⋈	2	Yes
3.20	⊓	1	Yes
3.50	◈	3	Yes

Gestion des stores electriques Capteur de lumiere	DES	06.04.2010
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ Cmd_ principale_ lum.pcb	

Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.80	+	70	Yes
0.81	X	28	Yes
0.90	Y	12	Yes
1.00	⊤	28	Yes
1.10		3	Yes
1.40	⧖	4	Yes
2.60	⋈	2	Yes
3.20	⊡	1	Yes
3.50	◇	3	Yes

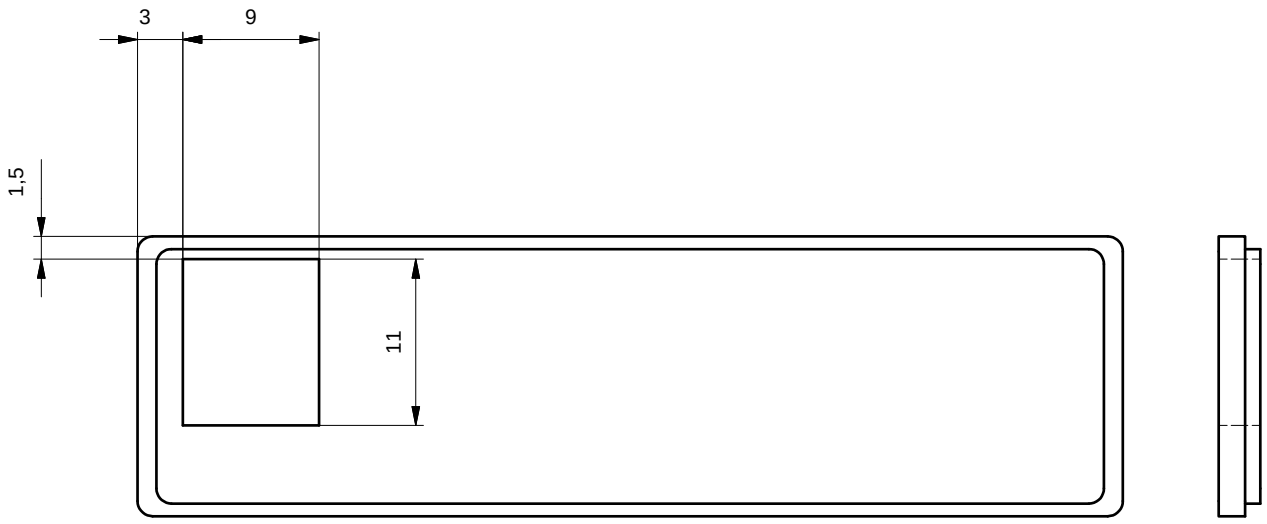


Gestion des stores electriques Capteur de lumiere	DES	06.04.2010
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ Cmd_principale_lum.pcb	



Drill Table			
Hole Dia (mm)	Symbol	Quantity	Plated
0.80	+	70	Yes
0.81	X	28	Yes
0.90	Y	12	Yes
1.00	⊤	28	Yes
1.10		3	Yes
1.40	⧖	4	Yes
2.60	⋈	2	Yes
3.20	⊡	1	Yes
3.50	◈	3	Yes

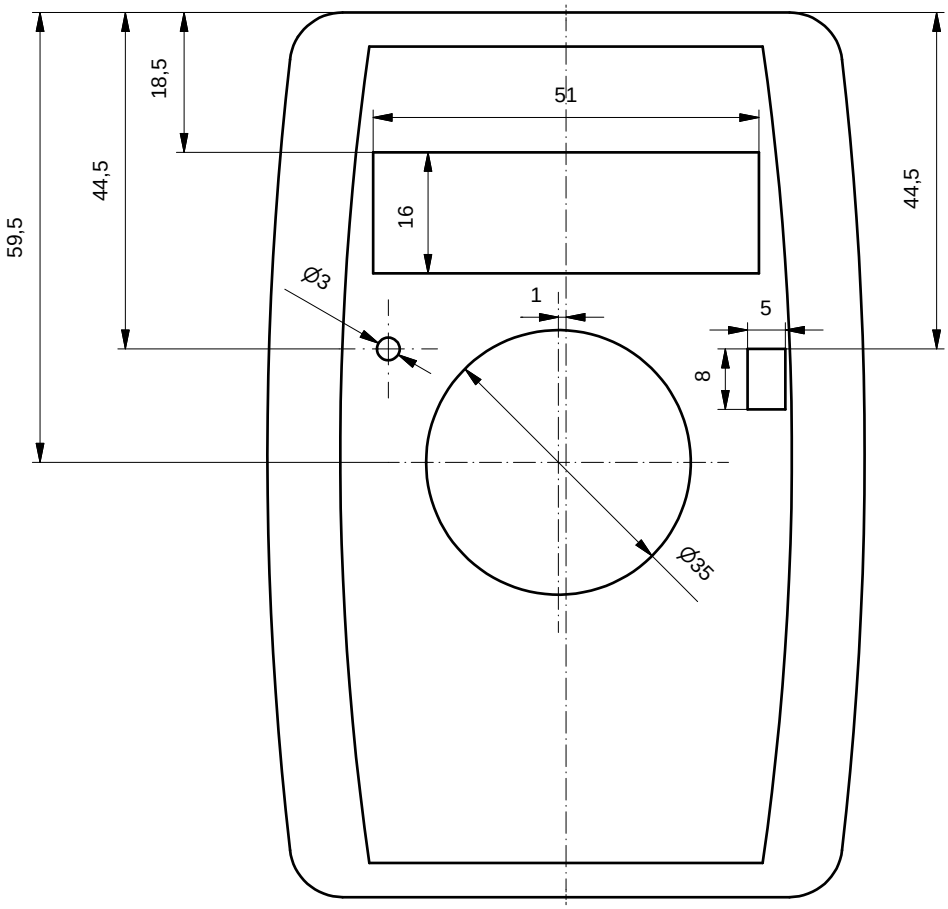
Gestion des stores electriques Capteur de lumiere	DES	06.04.2010
	REV	V1.1
HAUTE ECOLE VALAISANNE	PCB/ Cmd_ principale_ lum.pcb	



	Dessiné Gezeichnet	waltherj	07.06.2010	Echelle Massstab
	Contrôlé Geprüft			1:1

Fichier
Datei Y:\waltherj\plan_mecanique_TD_johan_walthert\fond_cmd_principale.idw

Hes·so VALAIS WALLIS



de ce côté ci la face latérale est ouverte

	Dessiné Gezeichnet	waltherj	08.06.2010	Echelle Massstab
	Contrôlé Geprüft			1:1

Fichier
Datei Y:\waltherj\plan_mecanique_TD_johan_walthert\face_cmd_principale.idw

Hes·so VALAIS WALLIS

```
//*********************************************************************
//* Fonction    :   TD_Store_elec()                                   *
//* Date        :   juillet 2010                                      *
//* Auteur      :   Johan Walthert                                    *
//* version     :   V1.0                                              *
//* Description :   programme principale pour le microcontroleur de la  *
//                  carte Cmd_principale. Ce programme s'occupe de la   *
//                  gestion des communications ainsi que des protocoles *
//                  de commande du store.                             *
//*********************************************************************

#include <pic18.h>

#include "DOGM132A.h"
//#include "main.h"
#include "string.h"
#include "stdlib.h"
#include "stdio.h"



//*****************************************
// déclaration des entree
//*****************************************
#define Up      RA1
#define Down    RA3
#define Left    RA0
#define Right   RA4
#define Ok      RA2
#define Man     RA5

#define DS_BUS  RC5
#define DS_DIR  TRISC5

//*****************************************
// déclaration des constentes
//*****************************************
#define mask    0x80

// trame a envoyer pour la communication RF
const unsigned char Trame_envoie [15][20] ={
    { 0x7E, 0x00, 0x10, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0x5A, 0xFF, 0xFE,
0x00, 0x44, 0x31, 0x05, 0xC7},   // 0 lum D1 activer
    { 0x7E, 0x00, 0x10, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0x5A, 0xFF, 0xFE,
0x00, 0x44, 0x31, 0x04, 0xC8},   // 1 lum D1 desactiver
    { 0x7E, 0x00, 0x0F, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0x5A, 0xFF, 0xFE,
0x00, 0x49, 0x53, 0xA5, 0x00},   // 2 lum IS etat entrée
    { 0x7E, 0x00, 0x0F, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0x5A, 0xFF, 0xFE,
0x00, 0x41, 0x43, 0xBD, 0x00},   // 3 lum apply changes
    { 0x7E, 0x00, 0x0F, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0x5A, 0xFF, 0xFE,
0x00, 0x41, 0x43, 0xBD, 0x00},   // 4 lum reserve pour pile
    { 0x7E, 0x00, 0x10, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0x79, 0xFF, 0xFE,
0x00, 0x44, 0x31, 0x05, 0xA8},   // 5 pers D1 activer
    { 0x7E, 0x00, 0x10, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0x79, 0xFF, 0xFE,
0x00, 0x44, 0x31, 0x04, 0xA9},   // 6 pres D1 descaviver
    { 0x7E, 0x00, 0x0F, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0x79, 0xFF, 0xFE,
0x00, 0x49, 0x53, 0x86, 0x00},   // 7 pres IS etat entrée
    { 0x7E, 0x00, 0x0F, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0x79, 0xFF, 0xFE,
0x00, 0x41, 0x43, 0x9E, 0x00},   // 8 pres apply changes
    { 0x7E, 0x00, 0x0F, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x33, 0x17, 0xA5, 0xFF, 0xFE,
0x00, 0x41, 0x43, 0xBD, 0x00},   // 9 reserve pour pile
    { 0x7E, 0x00, 0x10, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x31, 0xF8, 0x56, 0xFF, 0xFE,
0x00, 0x44, 0x31, 0x05, 0xEC},   // 10 mot monter activer
    { 0x7E, 0x00, 0x10, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x31, 0xF8, 0x56, 0xFF, 0xFE,
0x00, 0x44, 0x31, 0x04, 0xED},   // 11 mot monter descativer
    { 0x7E, 0x00, 0x10, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x31, 0xF8, 0x56, 0xFF, 0xFE,
0x00, 0x44, 0x32, 0x05, 0xEB},   // 12 mot descendre ativer
    { 0x7E, 0x00, 0x10, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x31, 0xF8, 0x56, 0xFF, 0xFE,
0x00, 0x44, 0x32, 0x04, 0xEC},   // 13 mot descendre descativer
    { 0x7E, 0x00, 0x0F, 0x17, 0x11, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x31, 0xF8, 0x56, 0xFF, 0xFE,
0x00, 0x41, 0x43, 0xE2, 0x00}    // 14 mot apply changes
};

//*****************************************
// déclaration des variables
//*****************************************
```

```
char mode = 1;                      // mode de fonctionnement 1 man 2 astro 3 save energie
char mode_travail = 2;              // mémorisation du mode auto
char fonct_normal = 0;              // choix astro foonctionnement ete 0 normal 1 inverse
char pos_inter   = 0;               // choix de la pos. interm. des stores en s
char echant_lum  = 1;               //defini le nbr de mesure avant de lancer un mouvement entre
1 et 3
char seuil_lum   = 50;              // défini le seuil de det. lumiere entre 0 et 100 %
int annee        = 2010;
char mois        = 1;
char jour        = 1;               // date du jour
char jours       = 1;               // jour de la sem lundi mardi ...
char heure       = 0;
char minute      = 0;
char temperature = 0;

char DS_Data     = 0x00;            //  buffer pour l'envoi de données au capteur de température
char temperatureData[10];           //  buffer pour les données recues du capteur de température
char signe;                         // signe de la température
char capteur_ok   = 0;              // indique que le système est en attente d'une réponce du
capteur le lumiere
char Presence_ok  = 0;              // indique que le système est en attente d'une réponce du
capteur de presence
char Presence2_ok = 0;              // confirme la réception d'un akc sur une mesure de rpésence
int Tempo_lum     = 0;              // temps de stabilisation de la photo diode (temps défini
plus bas)
int Tempo_pres    = 40;             // temps de stabilisation du capteur de mouvement(temps
défini plus bas)
int Tempo_temperature  = 0;         // tempo entre 2 mesure de température
char Tempo_lum_ecran = 0;           // tempo pour le rétro éclairage
char echantillon_pres = 0;          // contrôle que l'échantillonage de présence se fait au max
chaques secondes
char Tempon_1;                      // tempon pour l'analyse de la présence
char Tempon_2;

char Alarme_bat_pres = 0;           // indique un niveau de bat faible
char Alarme_bat_lum  = 0;

char seuil_detect;                  // seuil de lumière mis a l'échelle 00 / FF
char maj_affichage = 0;             // autorise la mise a jour l'affichage
char maj_heure = 0;                 // autorise la mise a jour de l'heure
char bond   = 0;                    // variable anti rebond pour les boutons
char monter_ok = 1 ;                // contrôle pour bloquer l'envoie de tramme a répétition
char descendre_ok = 1 ;             // contrôle pour bloquer l'envoie de tramme a répétition
char registreRTC;                   // registre actif de l'horloge temps réelle

char lumiere3 ;                     // mémorisation de la valeur du capteur lumière -30min
char lumiere2 ;                     // mémorisation de la valeur du capteur lumière -15min
char lumiere1 ;                     // etat de la valeur du capteur lumière


char Presence       = 0;            // indique que la présence est détectée
char Lumiere        = 0;            // indique que la dèrnière mesure est supérieur au seuil
//char Data             = 0;        // données RTC
char  ByteNumber    = 0;            // numéro du bit dans le tableau pour la réception uart
char ByteNumberEnvoi    = 0;        // numéro du bit dans le tableau pour l'envoi sur l'uart
char  ByteMax       = 20;           // valeur max de la trame recue uart
char coucou[21]= "salut";           // valeur témoin pour le test de l'ecran
char Trame_recue_buff [10][40];     // buffer de réception des données RF
char CntWrite = 0;                  // pointeur d'écriture pour le buffer
char CntRead = 0;                   // pointeur de lecture pour le buffer

//*****************************************
// déclaration des sorties
//*****************************************

#define Led_Rouge   RB4
#define retroLcd    RB5
//sorties SPI
#define si          RB0
#define SCL         RB1
#define A0          RB2
#define CS1         RB3
```

```
//**********************************************************************
//* Fonction    :   Init_PIC()                                         *
//* Description :   permet d'initialiser le microcontroleur avec les   *
//                  parametres de communicartion et de fonctionnement  *
//**********************************************************************
void init_PIC()
{
    //configuration de l'oscillateur interne
    OSCCON  = 0b01110000;

    // configurer port A comme entrée
    TRISA   = 0b00111111;
    CMCON   = 0b00000111;   //desactive le comparateur
    ADCON1  = 0b00001111;   //défini port A comme entrée digital
    PORTA   = 0b00000000;

    // configurer port B sorties 0-3 SPI 4 led rouge, 5-7 entree prog
    TRISB   = 0b11000000;
    PORTB   = 0b00010101;

    // Configuration port C UART I2C Temperature
    TRISC   = 0b11111000;
    PORTC   = 0b00000000;

    // configuration de l'i2c
    TRISC3 =1;//SDA et SCL configure en entree
    TRISC4=1;
    SSPCON1 = 0x38;
    SSPCON2 = 0x00;

    // vitesse de transmission
    SSPADD = 19; //Fi2c = Fosc/(4*(SSPADD+1)) = 100KHz
    CKE = 0;    //niveau d'entree
    SMP = 1;    //slew rate activer 0 = 400KHz
    SSPIF = 0;  //clear SSPIF interrupt flag
    BCLIF = 0;  //clear colision flag

    // configuration des interruptions
    INTCON  = 0b11100000;
    PIE1    = 0b00100000;

    // configuration USART pour la communication avec le XBEE
    TXSTA   = 0x24;
    BAUDCON = 0X08;
    RCSTA   = 0x90;
    SPBRGH  = 0x00;
    SPBRG   = 0xD0; //D0;
    CREN    = 1;    // activation de la réception sur l'UART

    // timer 0 clock de 1 seconde
    T0CON = 0x84;
    TMR0H = 0x0B;
    TMR0L = 0xDB;
}
//**********************************************************************
//* Fonction    :   param_origine()                                    *
//* Description :   rétablire les parametres d'origines du système     *
//**********************************************************************
void param_origine()
{
    mode = 1;   // mode de fonctionnement 1 man 2 astro 3 save energie
    mode_travail = 2;   // mémorisation du mode auto
    fonct_normal = 0;   // choix astro foonctionnement ete 0 normal 1 inverse
    pos_inter    = 0;   // choix de la pos. interm. des stores en s
    echant_lum   = 1;   //defini le nbr de mesure avant de lancer un mouvement entre 1 et 3
    seuil_lum    = 50;  // défini le seuil de det. lumiere entre 0 et 100 %
    annee        = 2010;
    mois         = 1;
    jour         = 1;   // date du jour
    jours        = 1;   // jour de la sem lundi mardi ...
    heure        = 0;
    minute       = 0;
    temperature  = 0;
}
//**********************************************************************
//* Fonction    :   delay()                                            *
//* Description :   implémentaion des délais en us et ms pour le 130   *
```

```
//**********************************************************************
void Delay130m()
{
    int time = 15000;
    while (--time>0)
    {
        asm("nop");
    }
}
void Delay480()
{
    int time = 50;
    while (--time>0)
    {
        asm("nop");
    }
}
void Delay750()
{
    int time = 78;
    while (--time>0)
    {
        asm("nop");
    }
}
void Delay100()
{
    int time = 11;
    while (--time>0)
    {
        asm("nop");
    }
}

//**********************************************************************
//* Fonction    :   ds_init()                                          *
//* Description :   initialisation de la communication avec le capteur  *
//*                 de temperature                                     *
//**********************************************************************
void ds_init()
{
    char DS_PRESENCE = 0;
    char DS_ERREUR   = 0;

    DS_BUS = 1;
    DS_DIR = 0;
    DS_BUS = 0;
    Delay480();
    DS_BUS = 1;
    DS_DIR = 1;
    Delay100();
    DS_PRESENCE = DS_BUS;
    if(DS_PRESENCE == 1)
    {
        DS_ERREUR = 1 ;
    }else
    {
        DS_ERREUR = 0 ;
    }
    Delay480();
}
//**********************************************************************
//* Fonction    :   ds_write()                                         *
//* Description :   ecriture dans le registre du capteur de temperature *
//**********************************************************************
void ds_write(char write_data)
{
    int count = 8;
    temperature = 0;
    for(count = 8; count>0; count--)
    {
        DS_Data = write_data & 0x01;
        DS_DIR = 0;
        DS_BUS = 0;
        asm("nop");
        DS_BUS = DS_Data;
        Delay100();
```

```
        DS_BUS = 1;
        write_data = write_data>>1;
        asm("nop");
    }
}

//***********************************************************************
//* Fonction    :   ds_write()                                          *
//* Description :   lecture du registre du capteur de temperature       *
//***********************************************************************

void readScratch()
{   int i = 9;
    int count = 8;
    int val = 1;
    ds_write(0xBE);
    for(i = 9; i>0;i--)
    {
        temperatureData[i]=0;
        for(count = 8;count>0;count--)
        {
            DS_DIR = 0;
            DS_BUS = 0;
            asm("nop");
            DS_DIR = 1;
            asm("nop");
            asm("nop");
            asm("nop");
            DS_Data = DS_BUS;
            temperatureData[i] = temperatureData[i] + DS_Data*val;
            val = val*2;
            asm("nop");
        }
        asm("nop");
        asm("nop");
    }
    asm("nop");
    ds_init();
    temperature = temperatureData[9]*0.5;
    signe = temperatureData[8];
    //remain = temperatureData[3]   // permet d'augmenter la résolution
    //perc = temperatureData[2]     // mais pas utile dans note cas
}
//***********************************************************************
//* Fonction    :   readProbe()                                         *
//* Description :   lecture de la température. Attention des lectures   *
//*                 trops raprochées faussent la mesure. T = 1 min      *
//***********************************************************************
void readProbe()
{
    ds_init();
    ds_write(0xCC); // skip Rom
    ds_write(0x44); // demander une convertion de temperature
    Delay750();
    ds_init();
    ds_write(0xCC); // skip Rom
    readScratch();
}

//***********************************************************************
//* Fonction    :   i2cWaitForIdle()                                    *
//* Description :   attente pour l'activation et controle le l'écriture *
//*                 sur le bus I2C pour l'horloge temps réelle          *
//***********************************************************************
void i2cWaitForIdle()
{
    while((SSPCON2 & 0x1F)|RW);
}
//***********************************************************************
//* Fonction    :   i2cStart()                                          *
//* Description :   donne un start bit sur le bus I2C                   *
//***********************************************************************
void i2cStart()
{
    i2cWaitForIdle();
    SEN = 1;
    while(SEN);
```

```
}
//**********************************************************************
//* Fonction    :   i2cRepStart()                                      *
//* Description :   repete le start bit sur le bus I2C                 *
//**********************************************************************
void i2cRepStart()
{
    i2cWaitForIdle();
    RSEN = 1;
    while (RSEN);
}
//**********************************************************************
//* Fonction    :   i2cStop()                                          *
//* Description :   donner le stop bit sur le bus I2C                  *
//**********************************************************************
void i2cStop()
{
    i2cWaitForIdle();
    PEN = 1;
    while (PEN);
}
//**********************************************************************
//* Fonction    :   i2cRead()                                          *
//* Description :   nous retourne les données sur le bus I2C sous la   *
//*                 forme d'un char                                    *
//**********************************************************************
unsigned char i2cRead(unsigned char ack)
{
    unsigned char data;
    i2cWaitForIdle();
    RCEN = 1;
    i2cWaitForIdle();
    data = SSPBUF;
    i2cWaitForIdle();
    if (ack)
    {
        ACKDT = 0;
    }
    else
    {
        ACKDT = 1;
    }
    ACKEN = 1;
    return(data);
}
//**********************************************************************
//* Fonction    :   i2cWrite()                                         *
//* Description :   écriture des données sur le bus et retourne la     *
//*                 valeur 1 si la transmission est bonne              *
//**********************************************************************
unsigned char i2cWrite(unsigned char data)
{
    i2cWaitForIdle();
    SSPBUF = data;
    return(!ACKSTAT); // retourne 1 si la transmission est ok
}
//**********************************************************************
//* Fonction    :   writeRTC()                                         *
//* Description :   écriture de données année jusque a minutes         *
//*                 dans l'horloge temps réelle                        *
//**********************************************************************
void writeRTC()
{
    i2cStart();
    i2cWrite(0xA2);
    i2cWrite(0x03);//registre minutes
    i2cWrite(((minute/10)<<4) + minute%10);// registre minute
    i2cWrite(((heure/10)<<4) + heure%10);// registre
    i2cWrite(((jour/10)<<4) + jour%10);// registre
    i2cWrite(((jours/10)<<4) + jours%10);// registre
    i2cWrite(((mois/10)<<4) + mois%10);// registre
    i2cWrite((((annee-2000)/10)<<4) + (annee-2000)%10);// registre
    i2cStop();
}
//**********************************************************************
//* Fonction    :   readRTC()                                          *
//* Description :   lecture de données année jusque a minutes          *
```

```
//*                 depuis l'horloge temps réelle                      *
//*********************************************************************
void readRTC()
{
    i2cStart();
    i2cWrite(0xA2);// adresse RTC en mode write
    i2cWrite(0x03);//registre minutes
    i2cRepStart();
    i2cWrite(0xA3);// adresse RTC en mode read
    registreRTC = i2cRead(1);
    minute = (registreRTC & 0x0F) + ((registreRTC & 0x70)>>4)*10;
    registreRTC = i2cRead(1);
    heure = (registreRTC & 0x0F) + ((registreRTC & 0x30)>>4)*10;
    registreRTC = i2cRead(1);
    jour = (registreRTC & 0x0F) + ((registreRTC & 0x30)>>4)*10;
    registreRTC = i2cRead(1);
    jours = (registreRTC & 0x07);
    registreRTC = i2cRead(1);
    mois = (registreRTC & 0x0F) + ((registreRTC & 0x1)>>4)*10;
    registreRTC = i2cRead(1);
    annee = ((registreRTC & 0x0F) + (registreRTC>>4)*10)+2000;
    i2cStop();
}
//*********************************************************************
//* Fonction    :   UART_Envoi()                                      *
//* Description :   Cette fonction permet d'écrire sur l'UART afin de   *
//*                 transmettre une trame au module Xbee qui la fera    *
//*                 suivre par transmission RF. La trame à envoyée est  *
//*                 définie dans le tableau Trame_envoie. Quand le byte *
//*                 est transmis sur l'UART, TXIF et TRMT sont mis à 1. *
//*********************************************************************

void UART_Envoi(char max_uart, char ligne)
{
    ByteNumberEnvoi = 0;
    do
    {
        TXREG = Trame_envoie [ligne][ByteNumberEnvoi];
        asm("nop");
        asm("nop");
        asm("nop");

        while(TXIF==0);

        asm("nop");
        asm("nop");
        asm("nop");

        while(TRMT == 0);

        ByteNumberEnvoi++;
    }
    while( ByteNumberEnvoi<max_uart);
}

//*********************************************************************
//* Fonction    :   Trame_analyse()                                   *
//* Description :   analyse et lancement des procédures  lors de la     *
//*                 reception d'une trame RF. 0x20 trame de réveil      *
//*                 15 et 17 mesure presence avec 17 alarme bat         *
//*                 17 et 19 mesure lumière avec 19 alarme bat          *
//*********************************************************************
void Trame_analyse()
{
    // trame signalant le réveil du système
    if (Trame_recue_buff[CntRead][2] == 0x20)
    {
        // origine de la trame, présence ou rayonnement
        // lumière
        if(Trame_recue_buff[CntRead][9] == 0x33 && Trame_recue_buff[CntRead][10] == 0x17 &&
Trame_recue_buff[CntRead][11] == 0x5A)
        {
            //lancer procédure aquisition valeur lumière
            UART_Envoi(20,0);       // trame pour activer la sortie D1
            UART_Envoi(19,3);       // appliquer les modifications
            // initialiser la temporisation pour la stabilisation du capteur
            Tempo_lum = 0;
```

```
            capteur_ok = 1;
        }
        // presence
        else if(Trame_recue_buff[CntRead][9] == 0x33 &&Trame_recue_buff[CntRead][10] == 0x17 &&
Trame_recue_buff[CntRead][11] == 0x79)
        {
            //lancer procédure aquisition valeur présence
            UART_Envoi(20,5);       // trame pour activer la sortie D1
            UART_Envoi(19,8);       // appliquer les modifications
            // initialiser la temporisation pour la stabilisation du capteur
            Tempo_pres = 0;
            Presence_ok = 1;
            Presence2_ok = 1;
        }
    }
    // une trame de data lumiere (commande envoyée IS) ou data présence avec bat faible
    else if (Trame_recue_buff[CntRead][2] == 0x17)
    {
        // trame donées lumière
        if(Trame_recue_buff[CntRead][10] == 0x33 && Trame_recue_buff[CntRead][11] == 0x17 &&
Trame_recue_buff[CntRead][12] == 0x5A)
        {
            //récupérer et archiver la valeur du capteur
            lumiere3 = lumiere2;
            lumiere2 = lumiere1;
            lumiere1 = Trame_recue_buff[CntRead][25];

            if (lumiere1>= seuil_detect)
            {
                Lumiere =1;
            }
            else
            {
                Lumiere =0 ;
            }
            //aquisition ok eteindre l'alimentation
            UART_Envoi(20,1);       //lumiere eteindre sortie 1
            UART_Envoi(19,3);       // lumiere apply changes
            Alarme_bat_lum= 0;
        }
        // trame capteur de présence avec niveau de pile faible
        if(Trame_recue_buff[CntRead][10] == 0x33 &&Trame_recue_buff[CntRead][11] == 0x17 &&
Trame_recue_buff[CntRead][12] == 0x79 )
        {
            //récupérer et convertire la valeur du capteur
            Tempon_1 = Trame_recue_buff[CntRead][23];
            Tempon_2 = Tempon_2 + (Tempon_1>>2);
            Presence2_ok = 1;
            if (Tempo_pres >= 20)
            {
                UART_Envoi(20,6);   // presence eteindre sortie D1
                UART_Envoi(19,8);   // presence apply changes
                Presence_ok = 0;
                //test de présence a la fin de l'échantillonage
                if (Tempon_2 >1)
                {   Presence = 1;
                    Tempon_2 = 0;
                }
                else
                {
                    Presence = 0;
                    Tempon_2 = 0;
                }
            }
            Alarme_bat_pres= 1;
        }
    }// trame capteur de lumière avec niveau des batteries faible
    else if (Trame_recue_buff[CntRead][2] == 0x19)
    {
        if(Trame_recue_buff[CntRead][10] == 0x33 && Trame_recue_buff[CntRead][11] == 0x17 &&
Trame_recue_buff[CntRead][12] == 0x5A)
        {
            //récupérer et archiver la valeur du capteur
            lumiere3 = lumiere2;
            lumiere2 = lumiere1;
            lumiere1 = Trame_recue_buff[CntRead][25];
```

```
            if (lumiere1>= seuil_detect)
            {
                Lumiere =1;
            }
            else
            {
                Lumiere =0 ;
            }
            //aquisition ok eteindre l'alimentation
            UART_Envoi(20,1);//lumiere eteindre sortie 1
            UART_Envoi(19,3); // lumiere apply changes
            Alarme_bat_lum = 1;
        }
    }
    // trame de données presence
    else if (Trame_recue_buff[CntRead][2] == 0x15)
    {   if(Trame_recue_buff[CntRead][10] == 0x33 &&Trame_recue_buff[CntRead][11] == 0x17 &&
Trame_recue_buff[CntRead][12] == 0x79 )
        {
            //récupérer et convertire la valeur du capteur
            Tempon_1 = Trame_recue_buff[CntRead][23];
            Tempon_2 = Tempon_2 + (Tempon_1>>2);
            Presence2_ok = 1;
            //Presence = 1;
            if (Tempo_pres >= 20)
            {
                UART_Envoi(20,6);// presence eteindre sortie D1
                UART_Envoi(19,8); // presence apply changes
                Presence_ok = 0;
                //test de présence a la fin de l'échantillonage
                if (Tempon_2 >1)
                {   Presence = 1;
                    Tempon_2 = 0;
                }else
                {
                    Presence = 0;
                    Tempon_2 = 0;
                }
            }
            Alarme_bat_pres = 0;
        }
    }
}
//*********************************************************************
//* Fonction    :   interrupt inter()                                 *
//* Description :   gestion d'une interrruption en cas de reception     *
//                  d'un paquet RF et gestion de la pulsion a 0.5Hz     *
//*********************************************************************
static void interrupt inter()
{
    if (RCIF==1)
    {
        // RCIF est mis à 0 lors de la lecture du registre RCREG
        Trame_recue_buff[CntWrite][ByteNumber] = RCREG;
        //détection du début de trame et retour au début de ligne
        if(Trame_recue_buff[CntWrite][ByteNumber]== 0x7E)
        {
            ByteNumber = 0;
        }
        ByteNumber++;
        // défini la longueur de la trame entrante
        if( ByteNumber == 3 )
        {
            ByteMax = Trame_recue_buff[CntWrite][2]+4;
        }
        // signal la fin d'une trame entrante pret pour analyse
        if (ByteNumber>ByteMax-1)
        {
            CntWrite++;
            ByteNumber = 0;
            if (CntWrite > 9)
            {
                CntWrite =0;
            }
        }
    }
    // interrupt du timer 0
```

```
    if (TMR0IF == 1 )
    {   TMR0H = 0x0B;
        TMR0L = 0xDB;
        Tempo_lum++;
        Tempo_pres++;
        Tempo_lum_ecran++;
        Tempo_temperature++;
        maj_heure++;
        echantillon_pres = 1;
        bond = 0;   // antirebond pour les boutons de commande

        // traitement des alarmes
        //*********************************************************************
        if (Alarme_bat_pres == 1 || Alarme_bat_pres == 1)
        {
            if (Led_Rouge == 1)
            {
                Led_Rouge = 0;
            } else
            {
                Led_Rouge = 1;
            }
        }
        else
        {
            Led_Rouge = 0;
        }
        TMR0IF = 0;
    }
}
//**********************************************************************
//* Fonction    :   envoi_SPI()                                        *
//* Description :   Cette fonction permet d'écrire sur le SPI afin de   *
//*                 transmettre une trame a l'écran. elle prend en      *
//*                 parametre le type de trame 0 = commande 1 = afficher*
//*                 et les 8 bit a transmettre                         *
//**********************************************************************
void envoi_SPI( char type_spi, char n_spi )
{   char z = 0;
    // donner le signal de transmition
    CS1=0;
    // choix du type de donnée 1= afficher 0= commande
    A0 = type_spi;
    asm("nop");
    //déclément du bit de donnée de 7 à 0
    for (z =0;z<8;z++)
    {
        SCL= 1;
        if ((n_spi & mask) == 0x80)
        {
            si = 1;
        }
        else
        {
            si = 0;
        }
        n_spi = n_spi << 1;
        SCL=0;
        asm ("nop");
    }
    SCL=1;
    asm("nop");//attente avant arret
    CS1=1;
}
//**********************************************************************
//* Fonction    :   stop_store()                                       *
//* Description :   Cette fonction établi le cycle pour l'action        *
////                "stoper le store". il faut choisir la trame a       *
////                envoyer ainsi que l'impulsion de commande du relais *
//**********************************************************************
void stop_store()
{
        UART_Envoi(20,10);      // activer cmd_mot sortie 2
        UART_Envoi(20,12);      // activer cmd_mot sortie 2
        Delay130m();            // delais pour éviter le chevauchement des trames
        UART_Envoi(19,14);      // cmd_mot apply changes
        //attente pour l'impulsion
```

```
        Delay130m();
        UART_Envoi(20,11);      // desactiver cmd_mot sortie 2
        UART_Envoi(20,13);      // desactiver cmd_mot sortie 2
        Delay130m();            // delais pour éviter le chevauchement des trames
        UART_Envoi(19,14);      // cmd_mot apply changes
}
//*********************************************************************
//* Fonction    :   monter_store()                                    *
//* Description :   Cette fonction établi le cycle pour l'action      *
//                  "monter le store".                                *
//*********************************************************************
void monter_store()
{
        stop_store();
        Delay130m();
        UART_Envoi(20,10);      // activer cmd_mot sortie 1
        UART_Envoi(19,14);      // cmd_mot apply changes
}
//*********************************************************************
//* Fonction    :   descendre_store()                                 *
//* Description :   Cette fonction établi le cycle pour l'action      *
//                  "descendre le store".                             *
//*********************************************************************
void descendre_store()
{
        stop_store();
        Delay130m();
        UART_Envoi(20,12);      // activer cmd_mot sortie 2
        UART_Envoi(19,14);      // cmd_mot apply changes
}

//*********************************************************************
//* Fonction    :   aff_normal()                                      *
//* Description :   afficher l'écran normal avec la date, l'heure     *
//*                 le mode, les capteurs, temperature, l'accés au menu *
//*********************************************************************
void aff_normal()
{
            char n;// variable pour la compilation du sprintf

            n = sprintf(coucou,"%d.%d.%d  %2d:%2d",jour, mois, annee, heure, minute);
            lcd_send_string(coucou,8,6);
            if(mode == 1)
            {
                strcpy(coucou,"Mode manuel      ");
            }
            else if(mode == 2)
            {
                strcpy(coucou,"Mode Astro       ");
            }
            else if (mode == 3)
            {
                strcpy(coucou,"Mode Save Energie");
            }
            lcd_send_string(coucou,24,19);
            strcpy(coucou,"Capteur");
            lcd_send_string(coucou,0,31);
            if(Presence == 1)
            {
                lcd_send_ASCII(0x7E,45,31);
            }
            else
            {
                lcd_send_ASCII(0x20,45,31);
            }
            if(Lumiere == 1)
            {
                lcd_send_ASCII(0x7F,56,31);
            }
            else
            {
                lcd_send_ASCII(0x20,56,31);
            }
            temperature = 20; // forcer la température pour la présentation car elle ne
fonctionne pas
            n = sprintf(coucou,"%d°c",temperature);
            lcd_send_string(coucou,67,31);
```

```
            strcpy(coucou,"Menu >");
            lcd_send_string(coucou,96,31);
}
//*********************************************************************
//* Fonction    :   menu_a1_txt()                                     *
//* Description :   sous menu pour la modification de la variable     *
//*                 on lui donne la valeur actuelle, le titre, un text  *
//*                 pour identifier l'action et les 2 texts a afficher  *
//*                 il nous retourne la valeur 0 ou 1 relatif au choix  *
//*                 du text pos0 ou pos 1                             *
//*********************************************************************
char menu_a1_txt(char valeur, char titre[20], char txt[10],char pos0[15],char pos1[15])
{
    signed char nouv_val = valeur;
    char mouve = 1;
    //unsigned char tab;
    lcd_clear();
    //affichage du texte fixe
    lcd_send_string(titre,0,7);
    lcd_send_string(txt,12,19);
    lcd_send_ASCII(0x81,120,27);        // triangle vers le bas
    lcd_send_ASCII(0x80,120,10);        // triangle vers le haut
    while (Ok!=1)
    {
        if (mouve==1)
        {
            if (nouv_val == 0)
            {
                lcd_send_string(pos0,60,19);
                mouve=0;
            }
            else if (nouv_val == 1)
            {
                lcd_send_string(pos1,60,19);
                mouve=0;
            }
        }
        if(Down == 1 && mouve == 0)
        {
            nouv_val--;
            //effacer l'ancienne valeur;
            strcpy(coucou, "               ");
            lcd_send_string(coucou,60,19);
            mouve=1;

        }
        else if (Up == 1 && mouve == 0)
        {
            nouv_val++;
            //effacer l'ancienne valeur;
            strcpy(coucou, "               ");
            lcd_send_string(coucou,60,19);
            mouve=1;
        }
        //teste des bornes min et max
        if (nouv_val > 1)
        {
            nouv_val = 0;
        }
        else if (nouv_val < 0)
        {
            nouv_val = 1;
        }
    }
    return nouv_val;
}
//*********************************************************************
//* Fonction    :   menu_a1()                                         *
//* Description :   sous menu pour la modification de la variable     *
//*                 on lui donne la valeur actuelle, le  titre, un text *
//*                 avant la variable, l'unité, les valeur max et min   *
//*                 il nous retourne la nouvelle valeur selectionnée    *
//*********************************************************************
int menu_a1(int valeur, char titre[20], char txt[10], char unit[5], int max, int min)
{
    int nouv_val = valeur;
    char mouve = 1;
```

```
    char n;//pour la compilation pas d'utilité
    lcd_clear();
    //affichage du texte fixe
    lcd_send_string(titre,0,7);
    lcd_send_string(txt,12,19);
    lcd_send_string(unit,103,19);

    lcd_send_ASCII(0x81,120,27);    // triangle vers le bas
    lcd_send_ASCII(0x80,120,10);    // triangle vers le haut
    while (Ok!=1)
    {
        if (mouve==1)
        {
            n = sprintf (coucou,"%4d", nouv_val);
            lcd_send_string(coucou,78,19);
            mouve = 0;
        }
        if(Down == 1 && mouve == 0)
        {
            nouv_val--;
            //effacer l'ancienne valeur;
            strcpy(coucou, "   ");
            lcd_send_string(coucou,78,19);
            mouve = 1;

        }
        else if (Up == 1 && mouve == 0)
        {
            nouv_val++;
            //effacer l'ancienne valeur;
            strcpy(coucou, "   ");
            lcd_send_string(coucou,78,19);
            mouve = 1;
        }
        //teste des bornes min et max
        if (nouv_val > max)
        {
            nouv_val = max;
        }
        else if (nouv_val < min)
        {
            nouv_val = min;
        }
    }
    return nouv_val;
}
//*************************************************************************
//* Fonction    :   menu_a2()                                             *
//* Description :   sous menu  a deux parametres. on lui fourni le titre *
//*                 du sous menu et les 2 choix possible. il nous         *
//*                 retourne 0 pour  retour en arrière ou la position de *
//*                 la flèche 1 pour le premier choix et 2 pour le second*
//*************************************************************************
 int menu_a2( char  titre[20], char param1[20], char param2[20] )
{
    int fleche = 1;
    int mouve = 1;
    lcd_clear();
    //affichage du texte du menu
    lcd_send_string(titre,0,7);
    lcd_send_string(param1,12,15);
    lcd_send_string(param2,12,23);
    // affichage de la fleche indicatrice
    do
    {   strcpy(coucou,"->");
        if(mouve==1)
        {
            switch (fleche)
            {
                case 1 :
                    {
                        lcd_send_string(coucou,0,15);
                        mouve=0;
                        Delay130m();
                        break;
                    }
                case 2 :
```

```
                    {
                        lcd_send_string(coucou,0,23);
                        mouve=0;
                        Delay130m();
                        break;
                    }
            }
        }
        //incremetn et decrement selon les bouton Up et Down
        if(Down == 1)
        {
            strcpy(coucou,"  ");
            lcd_send_string(coucou,0,7*(fleche+1)+fleche);
            fleche++;
            mouve=1;
        }
        else if (Up == 1)
        {
            strcpy(coucou,"  ");
            lcd_send_string(coucou,0,7*(fleche+1)+fleche);
            fleche--;
            mouve=1;
        }
        //teste des bornes min et max
        if (fleche > 2)
        {
            fleche = 1;
        }
        else if (fleche < 1)
        {
            fleche = 2;
        }
        if(Left==1)
        {fleche = 0;}
    }while(Ok!=1 && fleche != 0);
    return fleche;
}
//*********************************************************************
//* Fonction    :   menu_a3()                                         *
//* Description :   sous menu  a trois parametres. on lui fourni le titre*
//*                 du sous menu et les 3 choix possible. il nous     *
//*                 retourne 0 pour  retour en arrière ou la position de *
//*                 la flèche 1 pour le premier choix et 2 pour le second*
//*                 3 pour le troisième                               *
//*********************************************************************
 char menu_a3( char  titre[20], char param1[20], char param2[20], char param3[20] )
{
    char fleche = 1;
    char mouve = 1;
    lcd_clear();
    //affichage du texte du menu
    lcd_send_string(titre,0,7);
    lcd_send_string(param1,12,15);
    lcd_send_string(param2,12,23);
    lcd_send_string(param3,12,31);
    // affichage de la fleche indicatrice
    do
    {   strcpy(coucou,"->");
        if(mouve==1)
        {
            switch (fleche)
            {
                case 1 :
                    {
                        lcd_send_string(coucou,0,15);
                        mouve=0;
                        Delay130m();
                        break;
                    }
                case 2 :
                    {
                        lcd_send_string(coucou,0,23);
                        mouve=0;
                        Delay130m();
                        break;
                    }
                case 3 :
```

```
                    {
                        lcd_send_string(coucou,0,31);
                        mouve=0;
                        Delay130m();
                        break;
                    }
                }
            }
            //incremetn et decrement selon les bouton Up et Down
            if(Down == 1 && mouve == 0)
            {
                strcpy(coucou,"  ");
                lcd_send_string(coucou,0,7*(fleche+1)+fleche);
                fleche++;
                mouve=1;
            }
            else if (Up == 1 && mouve == 0)
            {
                strcpy(coucou,"  ");
                lcd_send_string(coucou,0,7*(fleche+1)+fleche);
                fleche--;
                mouve=1;
            }
            if(Left==1)
            {fleche = 0;}
            //teste des bornes min et max
            if (fleche > 3)
            {
                fleche = 1;
            }
            else if (fleche < 1)
            {
                fleche = 3;
            }
            if(Left==1)
            {fleche = 0;}
        }while(Ok!=1 && fleche != 0);
        //affichage page suivante
        return fleche;
}

//*********************************************************************
//* Fonction    :   menu_date()                                       *
//* Description :   programme modifier la date et l'heure             *
//*********************************************************************
void menu_date()
{
    //int val_init = 0;
    char titrea[20], txta[10], unita[5];
    strcpy(unita,"");
    lcd_clear();
    strcpy(titrea,"date et heure");
    strcpy(txta,"annee");
    annee = menu_a1(annee, titrea, txta, unita, 2100,2010);

    lcd_clear();
    strcpy(titrea,"date et heure");
    strcpy(txta,"mois");
    mois = menu_a1(mois, titrea, txta, unita, 12,1);

    lcd_clear();
    strcpy(titrea,"date et heure");
    strcpy(txta,"jour");
    jour = menu_a1(jour, titrea, txta, unita, 31,1);

    lcd_clear();
    strcpy(titrea,"date et heure");
    strcpy(txta,"jour sem");
    jours = menu_a1(jours, titrea, txta, unita, 7,1);

    lcd_clear();
    strcpy(titrea,"date et heure");
    strcpy(txta,"heure");
    heure = menu_a1(heure, titrea, txta, unita, 23,0);

    lcd_clear();
    strcpy(titrea,"date et heure");
```

```
    strcpy(txta,"minute");
    minute = menu_a1(minute, titrea, txta, unita, 59,0);
}
//*********************************************************************
//* Fonction    :   menu1()                                           *
//* Description :   programme pour l'affichage du menu                *
//*********************************************************************
void menu_0()
{   char titrea[21],param1a[20],param2a[20],param3a[20],txta[10],unita[5];
    char pos0a[15];
    char pos1a[15];
    char choix = 0;
    menu:
    lcd_clear();
    //affichage du texte du menu
    strcpy(titrea,"Menu");
    strcpy(param1a,"choix Astro/Save Energie");
    strcpy(param2a,"parametres");
    strcpy(param3a,"reglages");
    choix = menu_a3(titrea, param1a, param2a,param3a);
    // affichage page suivante
    switch (choix)
        {   case 0: //page precedente
            {
                asm("nop");
                asm("nop");
                break;
            }
            case 1: //page choix mode astro save energie
            {
                strcpy(titrea,"choix Astro/Save Energie");
                strcpy(txta,"mode");
                strcpy(pos0a,"Astro");
                strcpy(pos1a,"Save Energie");
                mode_travail = menu_a1_txt(mode_travail, titrea, txta, pos0a, pos1a)+2;
                mode = mode_travail;
                break;
            }
            case 2 :// page parametre
            {
                parametre:
                strcpy(titrea,"Parametre");
                strcpy(param1a,"param. Astro");
                strcpy(param2a,"param. Save Energie");
                choix = menu_a2(titrea, param1a, param2a);
                switch (choix)
                {
                    case 0: //page precedente
                    {
                        asm("nop");
                        goto menu;
                        break;
                    }
                    case 1: //page param mode astro
                    {
                        strcpy(titrea,"param. mode astro");
                        strcpy(param1a,"fonct. ete");
                        strcpy(param2a,"pos. intermediaire");
                        choix = menu_a2(titrea, param1a, param2a);

                        switch (choix)
                        {   case 0: //page principale
                            {
                                asm("nop");
                                goto parametre;
                                break;
                            }
                            case 1: //page commutation ete hiver fonction inverse d'ouverture
                            {
                                strcpy(titrea,"fonctionnement ete");
                                strcpy(txta,"fonct.");
                                strcpy(pos0a,"Normal");
                                strcpy(pos1a,"Inverse");
                                fonct_normal = menu_a1_txt(fonct_normal, titrea, txta, pos0a,
pos1a);

                                break;
```

```
                }
                case 2: //page offset position intermediaire
                {
                    strcpy(titrea,"pos. intermediaire");
                    strcpy(txta,"tempo.");
                    strcpy(unita,"sec");
                    pos_inter = menu_a1(pos_inter, titrea, txta, unita,100,0);
                    break;
                }
            }
            break;
        }
        case 2:
        {   strcpy(titrea,"param. save energie");
            strcpy(param1a,"echantillonage lum.");
            strcpy(param2a,"seuil det. lumiere");
            choix = menu_a2(titrea, param1a, param2a);

            switch (choix)
            {   case 0: //page principale
                {
                asm("nop");
                goto parametre;
                break;
                }
                case 1: //page Toff lumiere
                {
                    strcpy(titrea,"echantillonage lum.");
                    strcpy(txta,"nbr d'ech.");
                    strcpy(unita," ");
                    echant_lum = menu_a1(echant_lum, titrea, txta, unita,3,1);
                    break;
                }
                case 2: //page seuil lumiere
                {
                    strcpy(titrea,"seuil det. lumiere");
                    strcpy(txta,"seuil det.");
                    strcpy(unita,"%");
                    seuil_lum = menu_a1(seuil_lum, titrea, txta, unita,100,0);
                    // mise a l'échelle du seuil de détection
                    seuil_detect = (int)(seuil_lum * 2.55);
                    break;
                }
            }
            break;
        }
    }
    break;
}
case 3 : // page réglage
{
    strcpy(titrea,"reglage");
    strcpy(param1a,"date et heure");
    strcpy(param2a,"alarme presente");
    strcpy(param3a,"param. d'origine");
    choix = menu_a3(titrea, param1a, param2a, param3a);
    switch (choix)
            {   case 0: //page principale
                {
                    asm("nop");
                    goto menu;
                    break;
                }
                case 1: //page date et heure
                {
                    menu_date();
                    writeRTC();
                    break;
                }
                case 2: //page retablir la config d'origine
                {
                    lcd_clear();
                    strcpy(titrea,"alarme presente");
                    lcd_send_string(titrea,0,7);
                    if (Alarme_bat_pres == 1)
                    {
                        strcpy(titrea,"bat faible presence");
```

```
                                lcd_send_string(titrea,6,15);
                            }
                            if (Alarme_bat_lum == 1)
                            {
                                strcpy(titrea,"bat faible lumiere");
                                lcd_send_string(titrea,6,23);
                            }
                            if (Alarme_bat_lum == 0 && Alarme_bat_pres == 0)
                            {
                                strcpy(titrea,"pas d'alarme");
                                lcd_send_string(titrea,6,15);
                            }
                            strcpy(titrea,"Ok");
                            lcd_send_string(titrea,120,31);
                            while(Ok!=1);
                            break;
                        }
                        case 3: //page retablir la config d'origine
                        {   //int val_init = 0;
                            lcd_clear();
                            //affichage du texte fixe
                            strcpy(titrea,"param. d'origine");
                            lcd_send_string(titrea,0,7);
                            strcpy(titrea,"voulez vous retablir");
                            lcd_send_string(titrea,0,15);
                            strcpy(titrea,"les param. d'origine");
                            lcd_send_string(titrea,0,23);
                            strcpy(titrea,"Ok");
                            lcd_send_string(titrea,120,31);
                            strcpy(titrea,"< retour");
                            lcd_send_string(titrea,0,31);
                            choix = 0;
                            while(choix!=1)
                            {
                                if (Up == 1)
                                {
                                    param_origine();
                                    choix = 1;
                                }
                                else if(Left == 1)
                                {choix = 1;}
                            }
                            break;
                            }
                        }
                    break;
                }
            }
        lcd_clear();
        Tempo_lum_ecran = 0;
}
//*********************************************************************
//* Fonction    :   mode_astro()                                      *
//* Description :   le mode astro permet de gérer les commandes en    *
//*                 fonction d'une heure fixe prédéfinie pour le mois *
//*********************************************************************
void mode_astro()
{   char heure_lever,heure_coucher,min_lever,min_coucher;
    char ete_hiver = 0;
    int heure_lever_test,heure_coucher_test, heure_test;
    // choix des heures en fonction du mois
    switch(mois)
    {
        case 1: // mois de janvier
            {
                heure_lever = 9;
                min_lever = 0;

                heure_coucher = 17;
                min_coucher = 0;

                ete_hiver = 0;// mode normale
                break;
            }
        case 2: // mois de février
            {
                heure_lever = 8;
```

```
        min_lever = 20;

        heure_coucher = 18;
        min_coucher = 0;

        ete_hiver = 0;// mode normale
        break;
    }
case 3: // mois de mars
    {
        heure_lever = 7;
        min_lever = 40;

        heure_coucher = 19;
        min_coucher = 0;

        ete_hiver = 0;// mode normale
        break;
    }
case 4: // mois de avril
    {
        heure_lever = 7;
        min_lever = 0;

        heure_coucher = 20;
        min_coucher = 0;

        ete_hiver = 0;// mode normale
        break;
    }
case 5: // mois de mai
    {
        heure_lever = 6;
        min_lever = 20;

        heure_coucher = 21;
        min_coucher = 0;

        ete_hiver = 0;// mode normale
        break;
    }
case 6: // mois de juin
    {
        heure_lever = 5;
        min_lever = 40;

        heure_coucher = 22;
        min_coucher = 0;

        ete_hiver = fonct_normal;// mode normale ou inverse
        break;
    }
case 7: // mois de juillet
    {
        heure_lever = 5;
        min_lever = 40;

        heure_coucher = 22;
        min_coucher = 0;

        ete_hiver = fonct_normal;// mode normale ou inverse
        break;
    }
case 8: // mois de aout
    {
        heure_lever = 6;
        min_lever = 20;

        heure_coucher = 21;
        min_coucher = 0;

        ete_hiver = fonct_normal;// mode normale ou inverse
        break;
    }
case 9: // mois de septembre
    {
        heure_lever = 7;
```

```
            min_lever = 0;

            heure_coucher = 20;
            min_coucher = 0;

            ete_hiver = 0;// mode normale
            break;
        }
    case 10:    // mois de octobre
        {
            heure_lever = 7;
            min_lever = 40;

            heure_coucher = 19;
            min_coucher = 0;

            ete_hiver = 0;// mode normale
            break;
        }
    case 11:    // mois de novembre
        {
            heure_lever = 8;
            min_lever = 20;

            heure_coucher = 18;
            min_coucher = 0;

            ete_hiver = 0;// mode normale
            break;
        }
    case 12:    // mois de decembre
        {
            heure_lever = 8;
            min_lever = 0;

            heure_coucher = 17;
            min_coucher = 0;

            ete_hiver = 0;// mode normale
            break;
        }
}
//concatenation des heures pour les tests
heure_lever_test = heure_lever;
heure_lever_test = heure_lever_test<<8;
heure_lever_test = heure_lever_test + min_lever;

heure_coucher_test = heure_coucher;
heure_coucher_test = heure_coucher_test<<8;
heure_coucher_test = heure_coucher_test + min_coucher;

heure_test = heure;
heure_test = heure_test<<8;
heure_test = heure_test + minute;

//test de la position dans la journée et envoie des commandes
    if (heure_test>= heure_lever_test)
    {
        if (heure_test <= heure_coucher_test)
        {
            if (ete_hiver == 0)
            {
                monter_store();
            }
            else if (ete_hiver == 1)
            {
                descendre_store();
            }
        }
        else
        {   // période ou l'heure est plus haute que celle de test
            if (ete_hiver == 0)
            {
                descendre_store();
            }
            else
            {
```

```
                    monter_store();
                }
            }
        }
        if(heure_test < heure_lever_test)
        {   // période ou l'heure est plus basse que celle de test
            if (ete_hiver == 0)
            {
                descendre_store();
            }
            else
            {
                monter_store();
            }
        }
}
//*********************************************************************
//* Fonction    :   mode_save_energie()                               *
//* Description :   le mode save energie permet de gérer les commandes  *
//*                 en fonction de la luminosité extérieur            *
//*********************************************************************
void mode_save_energie()
{
    char saison;
    if (mois == 6 || mois == 7 || mois == 8)
    {
        saison = 1;         // mode été
    }
    else
    {
        saison =0;          //mode hiver
    }
    // mise a l'échelle du seuil de détection
    //seuil_detect = seuil_lum * 0xFF/100;
    // action en fonction du nombre d'échantillons
    switch (echant_lum)
    {
        case 1 :
        {
            if (lumiere1 >= seuil_detect)
            {
                if (saison == 0)
                {
                    monter_store();
                }else
                {
                    descendre_store();
                }
            }else
            {
                if (saison == 1)
                {
                    monter_store();
                }else
                {
                    descendre_store();
                }
            }
        break;
        }
        case 2 :
        {
            if (lumiere1 >= seuil_detect && lumiere2 >= seuil_detect)
            {
                if (saison == 0)
                {
                    monter_store();
                }else
                {
                    descendre_store();
                }
            }else
            {
                if (saison == 1)
                {
                    monter_store();
                }else
```

```
                {
                    descendre_store();
                }
            }
        break;
        }
        case 3 :
        {
            if (lumiere1 >= seuil_detect && lumiere2 >= seuil_detect && lumiere3 >= seuil_detect)
            {
                if (saison == 0)
                {
                    monter_store();
                }else
                {
                    descendre_store();
                }
            }else
            {
                if (saison == 1)
                {
                    monter_store();
                }else
                {
                    descendre_store();
                }
            }
        break;
        }
    }
}

//**********************************************************************
//* Fonction    :   main()                                             *
//* Description :   programme principale                               *
//**********************************************************************
void main()
{
    char flanc_auto_man = 1;
    char minute_old;
    char maj_mode = 0;
    init_PIC();
    lcd_init();
    RB5 =1;             //rétro éclairage
    menu_date();
    writeRTC();
    lcd_clear();
    maj_affichage =1;

    // mise a l'échelle du seuil de détection
    seuil_detect = (int)(seuil_lum * 2.55);
    while(1)
    {
        // test pour lancer la procédure d'analyse lors de la réception de trames
        //**********************************************************************
        if (CntRead + 1 != CntWrite)
        {
            if (CntRead +1 == 10 && CntWrite == 0)
            {
                asm("nop");
            }
            else
            {
                CntRead++;
                if (CntRead>9)
                {
                    CntRead =0;
                }
                Trame_analyse();
            }
        }
        // traitement des paquets radio pour les mesures
        //**********************************************************************
        if (Tempo_lum >= 4 && capteur_ok == 1)
        {
        // demande de l'état des entrées
            UART_Envoi(19,2);// lum etat des entrées
```

```
            capteur_ok = 0;
        }
        // attendre que le capteur soit stable et echantilloner pendant 20 secondes
        if (Tempo_pres >= 10 && Tempo_pres <= 20 && Presence_ok == 1 && Presence2_ok == 1 &&
echantillon_pres == 1)
        {
            echantillon_pres = 0;
            Presence2_ok = 0;
            // demande de mesure sur presence
            UART_Envoi(19,7);
        }

        // rafraichissement de la page principale chaque minute
        //*****************************************************
        if (maj_heure >= 10)
        {
            minute_old = minute;
            //mise a jour de la date et heur
            readRTC();
            if (minute_old != minute)
            {
                maj_affichage = 1;
            }
            maj_heure =0;
        }

        // eteindre le rétro éclérage aprés 30 secondes
        //*****************************************************
        if (Tempo_lum_ecran > 30 && RB5 == 1)
            {
                RB5 =0;
            }
        // mesure de température chaque 5 min
        //*****************************************************
        if ( Tempo_temperature > 300 && RB5 == 1)
            {
                readProbe();            // lecture de la température
                Tempo_temperature = 0;
                maj_mode = 1;
            }

        // détermination du mode de fonctionnement a executer
        //*****************************************************
        if (maj_mode == 1)
        {
            if (Presence == 1 || Man == 1)
            {
                mode = 1;
            }
            else
            {
            mode = mode_travail;
            }
            if (mode == 2)
            {
                mode_astro();
            }
            else if (mode == 3)
            {
                mode_save_energie();
            }
            maj_mode = 0;
            maj_affichage = 1;
        }
        // rafraichissement de la page principale
        //*****************************************************
        if (maj_affichage == 1)
        {
            // mise a jours de la température

            aff_normal();
            maj_affichage = 0;
        }

        // test des boutons
        //*****************************************************
        if (Left == 1 && bond == 0 )
```

```
{
    //aff_normal();
    readProbe();
    Tempo_lum_ecran = 0;
    RB5 =1;                 // rétro éclairage
    maj_affichage = 1;
    maj_mode = 1;
    bond=1;

}
if (Ok == 1 && bond == 0)
{
    mode = 1; // l'utilisateur demande une commande prioritaire
    if(mode == 1 )
    {
        stop_store();
        monter_ok       = 1;
        descendre_ok    = 1;
    }
    Tempo_lum_ecran = 0;
    RB5 =1;                 // rétro éclairage
    Led_Rouge = 0;
    bond = 1;
    maj_affichage = 1;
}
if (Up == 1 && bond == 0 )
{
    mode = 1; // l'utilisateur demande une commande prioritaire
    if (mode == 1 && monter_ok)
    {
        monter_store();
        monter_ok = 0;      // éviter l'envoi multiple
        //descendre_ok = 0;
    }
    Tempo_lum_ecran = 0;
    RB5 =1;                 // rétro éclairage
    bond = 1;
    maj_affichage = 1;
}
if (Down == 1 && bond == 0)
{
    mode = 1; // l'utilisateur demande une commande prioritaire
    if (mode == 1 && descendre_ok==1)
    {
        descendre_store();
        //monter_ok = 0;
        descendre_ok    = 0;// éviter l'envoi multiple
    }
    Tempo_lum_ecran = 0;
    RB5 =1;                 // rétro éclairage
    bond = 1;
    maj_affichage = 1;
}
if (Right == 1 && bond == 0)
{
    Tempo_lum_ecran = 0;
    RB5 =1;                 // rétro éclairage
    menu_0();
    bond = 1;
    maj_affichage = 1;
}
if (Man == 1 && flanc_auto_man == 0)
{
    mode = 1; // mode de travail en manuel
    maj_affichage = 1;
    flanc_auto_man = 1;
    Tempo_lum_ecran = 0;
    RB5 =1;                 // rétro éclairage
}
if (Man == 0 && flanc_auto_man == 1) // détection de passage man -> auto
{
    mode = mode_travail; // mode de travail en manuel
    maj_affichage = 1;
    flanc_auto_man = 0;
    Tempo_lum_ecran = 0;
    //maj_mode = 1;
    RB5 =1;                 // rétro éclairage
```

```
        }
    }
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//
//   DOGM 128*64 Graphical LCD Library
//*********************************************************************************
#include "DOGM132A.h"
#include "main.h"
//*********************************************************************************
//FrontLookup[] ASCII characters table
//*********************************************************************************
   const unsigned char FontLookup [0x62][6] ={
    { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },   // espace
    { 0x00, 0x00, 0x2f, 0x00, 0x00, 0x00 },   // !  0x01
    { 0x00, 0x07, 0x00, 0x07, 0x00, 0x00 },   // "  0x02
    { 0x14, 0x7f, 0x14, 0x7f, 0x14, 0x00 },   // #  0x03
    { 0x24, 0x2a, 0x7f, 0x2a, 0x12, 0x00 },   // $  0x04
    { 0x23, 0x13, 0x08, 0x64, 0x62, 0x00 },   // %  0x05
    { 0x36, 0x49, 0x55, 0x22, 0x50, 0x00 },   // &  x06
    { 0x00, 0x05, 0x03, 0x00, 0x00, 0x00 },   // '  07
    { 0x00, 0x1c, 0x22, 0x41, 0x00, 0x00 },   // (  08
    { 0x00, 0x41, 0x22, 0x1c, 0x00, 0x00 },   // )  09
    { 0x14, 0x08, 0x3E, 0x08, 0x14, 0x00 },   // *  0A
    { 0x08, 0x08, 0x3E, 0x08, 0x08, 0x00 },   // +  0B
    { 0x00, 0x00, 0x50, 0x30, 0x00, 0x00 },   // ,  0C
    { 0x10, 0x10, 0x10, 0x10, 0x10, 0x00 },   // -  0D
    { 0x00, 0x60, 0x60, 0x00, 0x00, 0x00 },   // .  0E
    { 0x20, 0x10, 0x08, 0x04, 0x02, 0x00 },   // /  0F
    { 0x3E, 0x51, 0x49, 0x45, 0x3E, 0x00 },   // 0  10
    { 0x00, 0x42, 0x7F, 0x40, 0x00, 0x00 },   // 1  11
    { 0x42, 0x61, 0x51, 0x49, 0x46, 0x00 },   // 2  12
    { 0x21, 0x41, 0x45, 0x4B, 0x31, 0x00 },   // 3  13
    { 0x18, 0x14, 0x12, 0x7F, 0x10, 0x00 },   // 4  14
    { 0x27, 0x45, 0x45, 0x45, 0x39, 0x00 },   // 5  15
    { 0x3C, 0x4A, 0x49, 0x49, 0x30, 0x00 },   // 6  16
    { 0x01, 0x71, 0x09, 0x05, 0x03, 0x00 },   // 7  17
    { 0x36, 0x49, 0x49, 0x49, 0x36, 0x00 },   // 8  18
    { 0x06, 0x49, 0x49, 0x29, 0x1E, 0x00 },   // 9  19
    { 0x00, 0x36, 0x36, 0x00, 0x00, 0x00 },   // :  1A
    { 0x00, 0x56, 0x36, 0x00, 0x00, 0x00 },   // ;  1B
    { 0x08, 0x14, 0x22, 0x41, 0x00, 0x00 },   // <  1C
    { 0x14, 0x14, 0x14, 0x14, 0x14, 0x00 },   // =  1D
    { 0x00, 0x41, 0x22, 0x14, 0x08, 0x00 },   // >  1E
    { 0x02, 0x01, 0x51, 0x09, 0x06, 0x00 },   // ?  1F
    { 0x32, 0x49, 0x59, 0x51, 0x3E, 0x00 },   // @  20
    { 0x7E, 0x11, 0x11, 0x11, 0x7E, 0x00 },   // A  21
    { 0x7F, 0x49, 0x49, 0x49, 0x36, 0x00 },   // B  22
    { 0x3E, 0x41, 0x41, 0x41, 0x22, 0x00 },   // C  23
    { 0x7F, 0x41, 0x41, 0x22, 0x1C, 0x00 },   // D  24
    { 0x7F, 0x49, 0x49, 0x49, 0x41, 0x00 },   // E  25
    { 0x7F, 0x09, 0x09, 0x09, 0x01, 0x00 },   // F  26
    { 0x3E, 0x41, 0x49, 0x49, 0x7A, 0x00 },   // G  27
    { 0x7F, 0x08, 0x08, 0x08, 0x7F, 0x00 },   // H  28
    { 0x00, 0x41, 0x7F, 0x41, 0x00, 0x00 },   // I  29
    { 0x20, 0x40, 0x41, 0x3F, 0x01, 0x00 },   // J  2A
    { 0x7F, 0x08, 0x14, 0x22, 0x41, 0x00 },   // K  2B
    { 0x7F, 0x40, 0x40, 0x40, 0x40, 0x00 },   // L  2C
    { 0x7F, 0x02, 0x0C, 0x02, 0x7F, 0x00 },   // M  2D
    { 0x7F, 0x04, 0x08, 0x10, 0x7F, 0x00 },   // N  2E
    { 0x3E, 0x41, 0x41, 0x41, 0x3E, 0x00 },   // O  2F
    { 0x7F, 0x09, 0x09, 0x09, 0x06, 0x00 },   // P  30
    { 0x3E, 0x41, 0x51, 0x21, 0x5E, 0x00 },   // Q  31
    { 0x7F, 0x09, 0x19, 0x29, 0x46, 0x00 },   // R  32
    { 0x46, 0x49, 0x49, 0x49, 0x31, 0x00 },   // S  33
    { 0x01, 0x01, 0x7F, 0x01, 0x01, 0x00 },   // T  34
    { 0x3F, 0x40, 0x40, 0x40, 0x3F, 0x00 },   // U  35
    { 0x1F, 0x20, 0x40, 0x20, 0x1F, 0x00 },   // V  36
    { 0x3F, 0x40, 0x38, 0x40, 0x3F, 0x00 },   // W  37
    { 0x63, 0x14, 0x08, 0x14, 0x63, 0x00 },   // X  38
    { 0x07, 0x08, 0x70, 0x08, 0x07, 0x00 },   // Y  39
    { 0x61, 0x51, 0x49, 0x45, 0x43, 0x00 },   // Z   3A
    { 0x00, 0x7F, 0x41, 0x41, 0x00, 0x00 },   // [   3B
    { 0x55, 0x2A, 0x55, 0x2A, 0x55, 0x00 },   // backslash  3C
    { 0x00, 0x41, 0x41, 0x7F, 0x00, 0x00 },   // ]   3D
    { 0x0E, 0x11, 0x11, 0x0E, 0x00, 0x00 },   // ^4,2,1,2,4--> oCelsius
    { 0x40, 0x40, 0x40, 0x40, 0x40, 0x00 },   // _   3F
    { 0x00, 0x01, 0x02, 0x04, 0x00, 0x00 },   // '   40
    { 0x20, 0x54, 0x54, 0x54, 0x78, 0x00 },   // a   41
```

```
    { 0x7F, 0x48, 0x44, 0x44, 0x38, 0x00 },   // b  42
    { 0x38, 0x44, 0x44, 0x44, 0x20, 0x00 },   // c  43
    { 0x38, 0x44, 0x44, 0x48, 0x7F, 0x00 },   // d  44
    { 0x38, 0x54, 0x54, 0x54, 0x18, 0x00 },   // e  45
    { 0x08, 0x7E, 0x09, 0x01, 0x02, 0x00 },   // f  46
    { 0x0C, 0x52, 0x52, 0x52, 0x3E, 0x00 },   // g  47
    { 0x7F, 0x08, 0x04, 0x04, 0x78, 0x00 },   // h  48
    { 0x00, 0x44, 0x7D, 0x40, 0x00, 0x00 },   // i  49
    { 0x20, 0x40, 0x44, 0x3D, 0x00, 0x00 },   // j  4A
    { 0x7F, 0x10, 0x28, 0x44, 0x00, 0x00 },   // k  4B
    { 0x00, 0x41, 0x7F, 0x40, 0x00, 0x00 },   // l  4C
    { 0x7C, 0x04, 0x18, 0x04, 0x78, 0x00 },   // m  4D
    { 0x7C, 0x08, 0x04, 0x04, 0x78, 0x00 },   // n  4E
    { 0x38, 0x44, 0x44, 0x44, 0x38, 0x00 },   // o  4F
    { 0x7C, 0x14, 0x14, 0x14, 0x08, 0x00 },   // p  50
    { 0x08, 0x14, 0x14, 0x18, 0x7C, 0x00 },   // q  51
    { 0x7C, 0x08, 0x04, 0x04, 0x08, 0x00 },   // r  52
    { 0x48, 0x54, 0x54, 0x54, 0x20, 0x00 },   // s  53
    { 0x04, 0x3F, 0x44, 0x40, 0x20, 0x00 },   // t  54
    { 0x3C, 0x40, 0x40, 0x20, 0x7C, 0x00 },   // u  55
    { 0x1C, 0x20, 0x40, 0x20, 0x1C, 0x00 },   // v  56
    { 0x3C, 0x40, 0x30, 0x40, 0x3C, 0x00 },   // w  57
    { 0x44, 0x28, 0x10, 0x28, 0x44, 0x00 },   // x  58
    { 0x0C, 0x50, 0x50, 0x50, 0x3C, 0x00 },   // y  59
    { 0x44, 0x64, 0x54, 0x4C, 0x44, 0x00 },   // z  5A
    { 0x08, 0x3E, 0x41, 0x41, 0x00, 0x00 },   // {   5B
    { 0x00, 0x00, 0x7F, 0x00, 0x00, 0x00 },   // |   5C
    { 0x00, 0x41, 0x41, 0x3E, 0x08, 0x00 },    // }  5D
    { 0x10, 0x48, 0x3B, 0x3B, 0x48, 0x10 },    // icone  presence  5E
    { 0x2B, 0x03, 0x10, 0x45, 0x00, 0x09 },    //  icone lumiere  5F
    { 0x04, 0x06, 0x07, 0x06, 0x04, 0x00 },    //  fleche haut  60
    { 0x10, 0x30, 0x70, 0x30, 0x10, 0x00 }    //  fleche bas  61
  };
//*****************************************************************************************
//FrontLookup[] ASCII characters table petit
//*****************************************************************************************

   const unsigned char FontSmall [0x5F][3] ={


    { 0x00, 0x00, 0x00},   // espace                          ** pas fait...
    { 0x00, 0x17, 0x00},   // !  0x01
    { 0x03, 0x00, 0x03},   // "  0x02
    { 0x1F, 0x0A, 0x1F},   // #  0x03
    { 0x24, 0x2a, 0x7f},   // $  0x04 **
    { 0x09, 0x04, 0x12},   // %  0x05
    { 0x36, 0x49, 0x55},   // &  x06 **
    { 0x00, 0x03, 0x00},   // '  07
    { 0x04, 0x0A, 0x11},   // (  08
    { 0x11, 0x0A, 0x04},   // )  09
    { 0x14, 0x08, 0x3E},   // *  0A **
    { 0x04, 0x1F, 0x04},   // +  0B
    { 0x00, 0x10, 0x00},   // ,  0C
    { 0x04, 0x04, 0x04},   // -  0D
    { 0x00, 0x10, 0x00},   // .  0E
    { 0x20, 0x10, 0x08},   // /  0F **
    { 0x1F, 0x11, 0x1F},    // 0  10
    { 0x00, 0x1F, 0x00},    // 1  11
    { 0x1D, 0x15, 0x17},    // 2  12
    { 0x11, 0x15, 0x1F},    // 3  13
    { 0x0C, 0x0A, 0x1F},    // 4  14
    { 0x17, 0x15, 0x1D},    // 5  15
    { 0x1F, 0x15, 0x1C},    // 6  16
    { 0x03, 0x01, 0x1F},    // 7  17
    { 0x1F, 0x15, 0x1F},    // 8  18
    { 0x17, 0x15, 0x1F},    // 9  19
    { 0x00, 0x0A, 0x00},   // :  1A
    { 0x00, 0x56, 0x36},   // ;  1B **
    { 0x04, 0x0A, 0x11},   // <  1C
    { 0x0A, 0x0A, 0x0A},   // =  1D
    { 0x11, 0x0A, 0x04},   // >  1E
    { 0x02, 0x01, 0x51},   // ?  1F **
    { 0x32, 0x49, 0x59},   // @  20 **
    { 0x1F, 0x05, 0x1F},   // A  21
    { 0x1F, 0x15, 0x1F},   // B  22
    { 0x1F, 0x11, 0x11},   // C  23
    { 0x1F, 0x11, 0x0E},   // D  24
```

```
    { 0x1F, 0x15, 0x11},   // E  25
    { 0x1F, 0x05, 0x01},   // F  26
    { 0x1F, 0x15, 0x1D},   // G  27
    { 0x1F, 0x04, 0x1F},   // H  28
    { 0x00, 0x1F, 0x00},   // I  29
    { 0x18, 0x10, 0x1F},   // J  2A
    { 0x1F, 0x04, 0x1B},   // K  2B
    { 0x1F, 0x10, 0x10},   // L  2C
    { 0x7F, 0x02, 0x1F},   // M  2D
    { 0x7F, 0x04, 0x08},   // N  2E **
    { 0x1F, 0x11, 0x1F},   // O  2F
    { 0x1F, 0x05, 0x07},   // P  30
    { 0x3E, 0x41, 0x51},   // Q  31 **
    { 0x1F, 0x0D, 0x17},   // R  32
    { 0x17, 0x15, 0x1D},   // S  33
    { 0x01, 0x1F, 0x01},   // T  34
    { 0x1F, 0x10, 0x1F},   // U  35
    { 0x0F, 0x10, 0x0F},   // V  36
    { 0x3F, 0x40, 0x38},   // W  37 **
    { 0x1B, 0x04, 0x1B},   // X  38
    { 0x03, 0x1C, 0x03},   // Y  39
    { 0x19, 0x15, 0x13},   // Z 3A
    { 0x1F, 0x11, 0x00},   // [ 3B
    { 0x55, 0x2A, 0x55},   // \ 3C  **
    { 0x00, 0x11, 0x1F},   // ] 3D
    { 0x07, 0x05, 0x07},   // ^4,2,1,2,4--> oCelsius
    { 0x10, 0x10, 0x10},   // _ 3F
    { 0x00, 0x03, 0x00},   // ' 40
    { 0x20, 0x54, 0x54},   // a 41  **
    { 0x7F, 0x48, 0x44},   // b  42 **
    { 0x38, 0x44, 0x44},   // c  43 **
    { 0x38, 0x44, 0x44},   // d  44 **
    { 0x38, 0x54, 0x54},   // e  45 **
    { 0x08, 0x7E, 0x09},   // f  46 **
    { 0x0C, 0x52, 0x52},   // g  47 **
    { 0x7F, 0x08, 0x04},   // h  48 **
    { 0x00, 0x44, 0x7D},   // i  49 **
    { 0x20, 0x40, 0x44},   // j  4A **
    { 0x7F, 0x10, 0x28},   // k  4B **
    { 0x00, 0x41, 0x7F},   // l  4C **
    { 0x7C, 0x04, 0x18},   // m  4D **
    { 0x7C, 0x08, 0x04},   // n  4E **
    { 0x38, 0x44, 0x44},   // o  4F **
    { 0x7C, 0x14, 0x14},   // p  50 **
    { 0x08, 0x14, 0x14},   // q  51 **
    { 0x7C, 0x08, 0x04},   // r  52 **
    { 0x48, 0x54, 0x54},   // s  53 **
    { 0x04, 0x3F, 0x44},   // t  54 **
    { 0x3C, 0x40, 0x40},   // u  55 **
    { 0x1C, 0x20, 0x40},   // v  56 **
    { 0x3C, 0x40, 0x30},   // w  57 **
    { 0x44, 0x28, 0x10},   // x  58 **
    { 0x0C, 0x50, 0x50},   // y  59 **
    { 0x44, 0x64, 0x54},   // z  5A **
    { 0x04, 0x0E, 0x1F},   // 5B flèche a gauche
    { 0x00, 0x00, 0x07},   // | 5C degré celsius
    { 0x05, 0x07, 0x00},// 5D degré celsius suite

    };

/*
   { 0x1F, 0x11, 0x1F},// 0  0x00
   { 0x00, 0x1F, 0x00},// 1  0x01
   { 0x1D, 0x15, 0x17},// 2  0x02
   { 0x15, 0x15, 0x1F},// 3  0x03
   { 0x0C, 0x0A, 0x1F},// 4  0x04
   { 0x17, 0x15, 0x1D},// 5  0x05
   { 0x1F, 0x15, 0x1C},// 6  0x06
   { 0x03, 0x01, 0x1F},// 7  0x07
   { 0x1F, 0x15, 0x1F},// 8  0x08
   { 0x17, 0x15, 0x1F},// 9  0x09
   { 0x1F, 0x1F, 0x1F},// 10 A
   { 0x1F, 0x1F, 0x1F},// 11 B
   { 0x1F, 0x11, 0x11},// 12 C
   { 0x00, 0x03, 0x03},// 13 degrés
   { 0x19, 0x04, 0x13},// 14 %
   { 0x1F, 0x04, 0x1C},// 15 h
```

```
   { 0x04, 0x0E, 0x1F},// 16 flèche a gauche
   { 0x1F, 0x0E, 0x04},// 16 flèche a droite
*/



char string[21]= "hello2" ;//speichert einen ASCII string


//-------------------------------------------------------------------------------------------

unsigned int LCD_RAM[4][132];//virtuelles LCD RAM


//*****************************************************************************
//lcd_clear_RAM()
//clear the virtual LCD_RAM
//-----------------------------------------------------------------------------

void lcd_clear_RAM()
{

    unsigned int x=0; //4 * 1 Byte
    unsigned int y=0; //132 * (8 * 1 Byte)

    for(x=0;x<4;x++)
    {
      for(y=0;y<132;y++)
       {
         LCD_RAM[x][y]=0x00;
       }
   }
}


//*****************************************************************************
//lcd_curseur(int pages, int curs)
//setzt den Curseur des Displays in die gewünschte Page und Spalte.
//Mit dem Parameter 'pages' gibt man dem Display die Page an, mit 'curs'
//wird die Spalte gespeichert.
//-----------------------------------------------------------------------------
void lcd_curseur(int pages,int curs)
{
   int curseurH=0;
   int curseurL=0;

   pages=pages+0xB0;
   envoi_SPI(0, pages);
   curseurL=(curs&0x0F)+0x00;
   curseurH=((curs>>4)&0x0F) +0x10;

   envoi_SPI(0, curseurH);
   envoi_SPI(0, curseurL);
}

//*****************************************************************************
//lcd_clear()
//Clear display
//-----------------------------------------------------------------------------
void lcd_clear()
{
   int   y=0;
   int   x=0;

   lcd_clear_RAM();

   for(x=0;x<4;x++)
   {
      lcd_curseur(x,0);

      for(y=0;y<132;y++)
      {
         envoi_SPI(1, 0);
      }
   }
}
```

```
//*****************************************************************************
//lcd_conv_coord_y(int cor_y)
//Place a pixel in a Byte
//
//-----------------------------------------------------------------------------
int lcd_conv_coord_y(int cor_y)
{

   int data=0b00000001;

   data=data<<cor_y;

   return data;

}

//*****************************************************************************
//lcd_pixel(int page, int co_x, int co_y, int clr)
//With this feature is selected finally to the pixel LCD display
//cleverly and is responsible for the function lcd_send_pixel been implemented
//At the function, each pixel is written in a virtual RAM
//The storage is necessary to display 8bit
//Sent need to be something to write. It is therefore important to
//know what pixels in the same column and Page already been described,
//to put it together with the new pixel on the screen again to send.
//With 'page' is the function of the page you are notified.
//'co_x' contains the column in which is written. But serves in this
//Function is only to the virtual RAM to describe.
//'co_y' contains the line of the page to describe pixels

//-----------------------------------------------------------------------------

void lcd_pixel(int page,int co_x,int co_y,int clr){

         if(clr==0){
            if((co_y&LCD_RAM[page][co_x])==0){
               co_y=LCD_RAM[page][co_x];
            }//endIF
            else{
               co_y=co_y^LCD_RAM[page][co_x];
            }//endELSE

         }//endIF
         else{
            if(co_y==(co_y & LCD_RAM[page][co_x])){
               co_y=LCD_RAM[page][co_x];
            }//endIF
            else{
               co_y=co_y+LCD_RAM[page][co_x];
            }//endELSE
         }//endELSE

         LCD_RAM[page][co_x]=co_y;
         envoi_SPI(1, co_y);
}



//*****************************************************************************
//lcd_send_pixel(int coord_x, int coord_y, int clear)
//converts the coordinates' coord_x 'and' coord_y 'so to parameters that
//lcd_pixel function can interpret correctly.
//The function receive 'clear' parameter that indicate if a pixel muss be written
//or deleted.
//-----------------------------------------------------------------------------
void lcd_send_pixel(int coord_x,int coord_y,int clear){

   int block;

/*
   coord_x=127-coord_x;                //Permet de mettre l'écran dans l'autre sens
   coord_y=63-coord_y;
*/
   if(coord_y<0x08) block=0;

   if(0x07<coord_y && coord_y<0x10) block=1;
```

```
   if(0x0F<coord_y && coord_y<0x18) block=2;

   if(0x17<coord_y && coord_y<0x20) block=3;

   if(0x1F<coord_y && coord_y<0x28) block=4;

   if(0x27<coord_y && coord_y<0x30)  block=5;

   if(0x2F<coord_y && coord_y<0x38)  block=6;

   if(0x37<coord_y && coord_y<0x40)  block=7;


   lcd_curseur(block,coord_x);
   coord_y=lcd_conv_coord_y(coord_y-(block*8));
   lcd_pixel(block,coord_x,coord_y,clear);
   coord_y=0;
}






//*****************************************************************************
//lcd_init()
//Init the LCD-Display
//-----------------------------------------------------------------------------
void lcd_init() {
      //Reset display
      //output_low(LCD_RESET);
     // delay_us(100);
      //output_high(LCD_RESET);
     // delay_us(100);
      //Pulse on CS, LCD SPI can now receive data
//          LCD_CS=1;
      //delay_us(100);
//      int k=0;

      envoi_SPI(0, 0X40);   //Display start line 0
      envoi_SPI(0, 0XA1);   //ADC reverse
      envoi_SPI(0, 0XC0);   //Normal COM0-COM63
      envoi_SPI(0, 0XA6);   //Display normal
      envoi_SPI(0, 0XA2);   //Set bias 1/9
      envoi_SPI(0, 0X2F);    //Booster,Regulator and Follower on
      envoi_SPI(0, 0XF8);   //Set internal Booster to 4x
      envoi_SPI(0, 0X00);   //Set internal Booster to 4x
      envoi_SPI(0, 0X23);   //_
      envoi_SPI(0, 0X81);   // |=>Contrast Set
      envoi_SPI(0, 0X1F);   //-
      envoi_SPI(0, 0XAC);   //No indicator
      envoi_SPI(0, 0X00);   //No indicator
      envoi_SPI(0, 0XAF);   //Display on

      lcd_clear();


}//endSUB
//*****************************************************************************
//lcd_send_ASCII(unsigned char ascii,int x_coord,int y_coord)
//-----------------------------------------------------------------------------
void lcd_send_ASCII(unsigned char ascii,int x_coord,int y_coord)
{


   int o;//for-loop
   int t;//for-loop
   int code; //stores a column of ASCII character
   int y_start;//saves the initial value of y_koords
   int q=7;//counter for a single pixel column
   int decode;
    int decode1;
    int decode2;
    int decode3;
```

```
   q=7;
   y_start=y_coord;

   //6x 8bits can write a column
   for(o=0;o<6;o++)
   {

      //The first column is loaded in 'code'
      code = FontLookup[(ascii-0x20)][o];
      code = code*2;// pour décaler le code ascii de 1 pixel vers le bas
      //a column ( 8 bits ) will be written
      for(t=0;t<8;t++)
      { decode=1;
        decode1= decode<<q;
        decode2 = code & decode1;
        decode3 = decode2>>q;

         if(decode3==1)
         {
            lcd_send_pixel(x_coord,y_coord,1);
         }

         else
         {
            lcd_send_pixel(x_coord,y_coord,0);
         }
         q--;
         y_coord--;
      }
      q=7;
      y_coord=y_start;
      x_coord++;
   }
}

//*****************************************************************************
//lcd_send_small font
//*****************************************************************************

void lcd_send_ASCII_small(unsigned char ascii,int x_coord,int y_coord)
{

   int o;//for-loop
   int t;//for-loop
   int code; //stores a column of ASCII character
   int y_start;//saves the initial value of y_koords
   int q=7;//counter for a single pixel column
   int decode ; //test du bite dans la variable code


   q=4;
   y_start=y_coord;

   //3x 5bits can write a column
   for(o=0;o<3;o++)
   {

      code=FontSmall[(ascii-0x20)][o];
      for(t=0;t<5;t++)
      { decode=1;
        decode= decode<<q;
        decode = code & decode;
        decode = decode>>q;
         if(decode==1)
        {
            lcd_send_pixel(x_coord,y_coord,1);
        }
         else
        {
            lcd_send_pixel(x_coord,y_coord,0);
        }
         q--;
         y_coord--;
      }
      q=4;
      y_coord=y_start;
      x_coord++;
```

```
   }
}

//***************************************************************************
//lcd_send_string(int c_x, c_y)
//---------------------------------------------------------------------------
void lcd_send_string( char text[21],int c_x,int c_y){


     int p;

      for(p=0;p<21;p++){

         if(text[p]!=0x00){
            lcd_send_ASCII(text[p],c_x,c_y);
            c_x=c_x+6;
         }//endIF

         else{
            p=21;
         }//endELSE

         if(c_x>127){
            p=21;
         }//endIF
      }//endFOR
}//endSUB


//***************************************************************************
//lcd_send_string(int c_x, c_y)
//---------------------------------------------------------------------------
void lcd_send_string_small(char text[21], int c_x,int c_y)
{
     int p;

      for(p=0;p<21;p++)
      {

         if(text[p]!=0x00)
         {
            //lcd_send_ASCII_small(string[p]-0x30,c_x,c_y);
            lcd_send_ASCII_small(text[p],c_x,c_y);
            c_x=c_x+4;
         }
         else p=21;

       }
}//endSUB


//***************************************************************************
//lcd_send_horizontal
//draw a horizontal line
void lcd_send_horizontal(unsigned int x_start,unsigned int x_end,unsigned int y_fix){

   unsigned int x_achse;
      unsigned int speicher;


      if(x_start>x_end){
         speicher=x_start;
         x_start=x_end;
         x_end=speicher;
      }
      for(x_achse=x_start;x_achse<(x_end+1);x_achse++){
         lcd_send_pixel(x_achse,y_fix,1);
      }

}

//***************************************************************************
//lcd_send_horizontal
//draw a horizontal line
void lcd_clear_horizontal(unsigned int x_start,unsigned int x_end,unsigned int y_fix){
```

```
   unsigned int x_achse;
   unsigned int speicher;



      if(x_start>x_end){
         speicher=x_start;
         x_start=x_end;
         x_end=speicher;
      }
      for(x_achse=x_start;x_achse<(x_end+1);x_achse++){
         lcd_send_pixel(x_achse,y_fix,0);
      }

}

//*****************************************************************************
//lcd_send_vertical
//draw a vertical line
void lcd_send_vertical(unsigned int y_start,unsigned int y_end, unsigned x_fix){

   unsigned int y_achse;
   unsigned int speicher;

      if(y_start>y_end){
         speicher=y_start;
         y_start=y_end;
         y_end=speicher;
      }
      for(y_achse=y_start;y_achse<(y_end+1);y_achse++){
         lcd_send_pixel(x_fix,y_achse,1);
      }
}

//lcd_send_vertical
//draw a vertical line
void lcd_clear_vertical(unsigned int y_start,unsigned int y_end, unsigned x_fix){

      unsigned int y_achse;
      unsigned int speicher;

      if(y_start>y_end){
         speicher=y_start;
         y_start=y_end;
         y_end=speicher;
      }
      for(y_achse=y_start;y_achse<(y_end+1);y_achse++){
         lcd_send_pixel(x_fix,y_achse,0);
      }
}
```

Annexe n°5

Manuel utilisateur

Contenu

Manuel utilisateur

Manuel utilisateur

Sommaire

1 Description des modes de fonctionnement 3
1.1 Mode manuel 3
1.2 Mode astro 3
1.3 Mode save energie 4
2 Affichage principal 4
3 Les commandes 4
4 Menu 5
4.1.1 Page menu 5
4.1.2 Page paramètre 5
4.1.3 Page réglage 5
5 Paramètres et réglages 6
5.1 Choix Astro/Save energie 6
5.2 Position intermédiaire (pas actif dans cette version) 6
5.3 Fonctionnement été 6
5.4 Echantillon lumière 6
5.5 Seuil lumière 6
5.6 Date et heure 7
5.7 Alarme présente 7
5.8 Param d’origine 7

1 Description des modes de fonctionnement

1.1 Mode manuel

Le mode manuel permet au système d'être utilisé comme une simple télécommande avec les options monter descendre et stop. Ce mode est actif lorsqu'une présence est détectée ou que le commutateur man / astro, save energie est en position man.

1.2 Mode astro

Le mode Astro est l'un des deux modes automatiques. Une plage horaire fixe est définie pour chaque mois de l'année. Durant la journée les stores sont ouverts et fermés durant la nuit. Le paramètre « fonctionnement été » permet d'inverser l'état des stores durant l'été (juin à aout).

Plages horaires

Mois	Début	Fin
Janvier	9h00	17h00
Février	8h20	18h00
Mars	7h40	19h00
Avril	7h00	20h00
Mai	6h20	21h00
Juin	5h40	22h00
Juillet	5h40	22h00
Aout	6h20	21h00
Septembre	7h00	20h00
Octobre	7h40	19h00
Novembre	8h20	18h00
Décembre	9h00	17h00

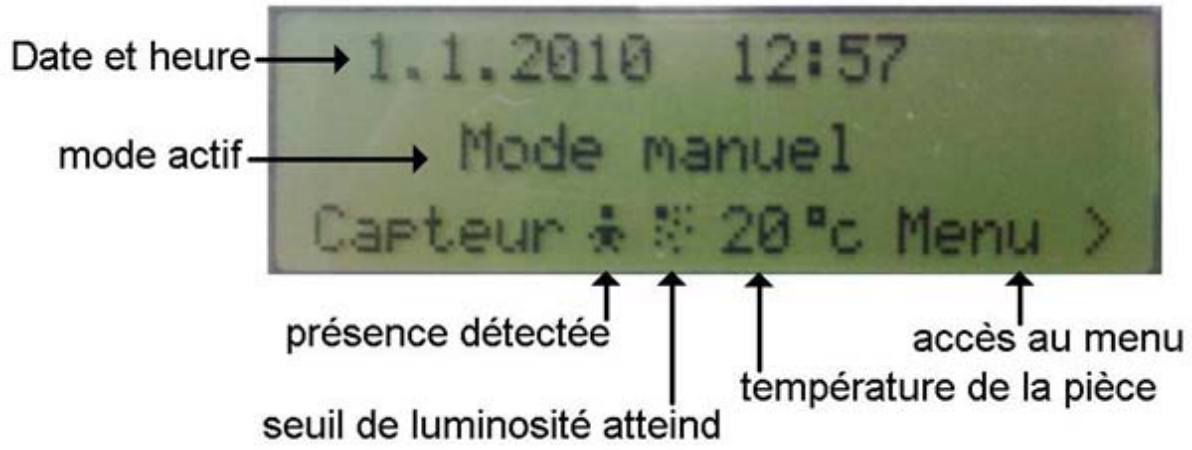
1.3 Mode save energie

Le mode Save Energie est le deuxième mode automatique. Chaque 15 minutes une mesure de luminosité est effectuée ainsi qu'une mesure de présence dans la pièce. Le système réagit en fonction de ces mesures afin de laisser entrer la chaleur et de la conserver au mieux pour l'hiver, inversement pour l'été. En cas de présence la commande manuelle est prioritaire.

L'utilisateur peut choisir le seuil de détection de la luminosité entre 0 et 100%. Il peut également choisir un nombre de mesures identiques avant d'agir entre 1 et 3. Cela permet d'introduire une temporisation et éviter des mouvements trop fréquents.

2 Affichage principal

Afin de fournir l'état du système et ses paramètres. La commande est munie d'un écran. Avec les différentes options suivantes.



Cette affichage est mis à jour automatiquement chaque 5 min il peut être mis à jour avec la flèche à gauche à tout moment lorsque cette page est affichée.

3 Les commandes

→	quand le texte menu> est affiché en bas à droite, permet d'accéder au menu.
Ok	permet de valider le choix du sous menu ou de la valeur du paramètre.
←	permet de revenir en arrière dans le menu ou de mettre à jour l'affichage principal.
Flèche haut	permet de monter la sélection.
Flèche bas	permet de descendre la sélection.

4 Menu

Un menu permet de modifier les paramètres, de régler la date et l'heure, de choisir le mode automatique et d'autres fonctions.

4.1.1 Page menu

Nous donnes accès aux sous menus suivants

- Choix astro / Save Energie
- Paramètre
- Réglage

4.1.2 Page paramètre

Nous donnes accès aux sous menus suivants

- Param. Astro
- Param. Save Energie

4.1.3 Page réglage

Nous donnes accès aux sous menus suivants

- Date et heure
- Alarme présente
- Param. d'origine

5 Paramètres et réglages

5.1 Choix Astro/Save energie

Accès menu > choix Astro/Save Energie

Description permet de choisir le type de mode automatique entre astro et save energie. (description des modes plus haut)

5.2 Position intermédiaire (pas actif dans cette version)

Accés menu > paramètres > param. Astro > pos. Intermédiaire

Description permet de donner un temps de montée ou de descente au store afin qu'il ne s'ouvre ou ne se ferme pas complètement. Ce temps est défini ente 0 et 60 secondes. Le zéro indiquant que le store s'ouvre et se ferme a 100%.

5.3 Fonctionnement été

Accès menu > paramètres > param. Astro > Fonctionnement été

Description permet de définir le mode normal ou inverse pour la commande du store durant l'été du mois de juin à fin aout.

Normal : le store s'ouvre la journée et se ferme le soir

Inverse : le store se ferme la journée et s'ouvre le soir

5.4 Echantillon lumière

Accès menu > paramètres > param. Save Energie > echantillon lumiere

Description permet de choisir le nombre de mesures, entre 1 et 3, consécutives et dépassant le seuil pour permettre l'exécution du mode save energie. Cela permet d'éviter des mouvements trop fréquents des stores.

1 : a chaque mesure le système réagit

2 : le système optimise la position du store quand 2 mesures consécutives sont supérieures au seuil choisi.

3 : le système optimise la position du store quand 3 mesures consécutives sont supérieures au seuil choisi.

5.5 Seuil lumière

Accès menu > paramètres > param. Save Energie > seuil détection

Description le seuil de détection varie entre 0 et 100% de la mesure de luminosité. Il permet d'ajuster le seuil en fonction de l'exposition du capteur.

5.6 Date et heure

Accès menu > réglage > date et heure

Description cette fonction permet de modifier la date et l'heure du système avec l'ordre chronologique suivant :

Année	année actuelle à partir de 2010
Mois	mois actuelle entre 1 et 12 le 1 étant janvier.
Jour	jour ente 1 et 31
Jour de la sem	jour de la semaine entre 1 et 7 le 1 étant lundi
Heure	entre 0 et 23 heures
Minute	entre 0 et 59 minutes

5.7 Alarme présente

Accès menu > réglage > Alarme présente

Description Donne la liste des erreurs présentes afin de les corriger

Signalisation	Type	description
Bat. faible lumière	Avertissement	Le niveau des piles du capteur d'ensoleillement est faible. Remplacer les piles.
Bat. faible présence	Avertissement	Le niveau des piles du capteur de présence est faible. Remplacer les piles.
Erreur com.	Erreur	La communication sans fils avec les capteurs ou le module de commande du store a été perdue. 1 Vérifier les piles des capteurs 2 Vérifier que le moteur du store soit bien alimenté (donner une commande manuelle) 3 Débrancher la cmd_principale et attendez 10 secondes avant de la rebrancher.
Pas d'alarme	Avertissement, erreur	Aucune erreur ou alarme n'est présente.

5.8 Param d'origine

Accès menu > réglage > param. d'origine

Description permet de rétablir les paramètres d'origine du système.

Annexe n°6

Liste matériel

Contenu

Listes des composants avec les fournisseurs, le prix par carte et au total.

Liste des composants

Carte rayonnement

quantité	déniomination	fournisseur	n°	prix	sous total
1	photo diode	distrelec	630402	13.56	13.56
1	ampli op CMOS MCP601	distrelec	644050	1.83	1.83
1	module Xbee	reselec	XB24-Z7CIT-004	21.6	21.6
2	socle pour Xbee			0.1	0.2
2	jumper	distrelec	121538	0.7	1.4
1	résistance 1M			0.1	0.1
1	résistance 1K			0.1	0.1
1	résistance 5.6K			0.1	0.1
1	condensateur découplage			0.1	0.1
1	led verte	distrelec	632031	2.37	2.37
2	bouton poussoir	distrelec	200525	1.72	3.44
1	support de pile	conrad	651043-wu	2.75	2.75
1	boitier 92X66X28	distrelec	300673	12.91	12.91
1	connecteur capteur			1	1
				total	61.46

Carte présence

quantité	déniomination	fournisseur	n°	prix	sous total
1	Capteur de mouvement	distrelec	240071	36.85	36.85
1	module Xbee	reselec	XB24-Z7CIT-004	21.6	21.6
2	socle pour Xbee			0.1	0.2
2	jumper	distrelec	121538	0.7	1.4
1	résistance 1M			0.1	0.1
1	résistance 1K			0.1	0.1
1	led verte	distrelec	632031	2.37	2.37
2	bouton poussoir	distrelec	200525	1.72	3.44
1	support de pile	conrad	651043-wu	2.75	2.75
1	boitier	distrelec	300673	12.91	12.91
				total	81.72

Carte cmd_moteur

quantité	déniomination	fournisseur	n°	prix	sous total
2	relais de commande des stores	distrelec	402415	3.77	7.54
1	module Xbee	reselec	XB24-Z7CIT-004	21.6	21.6
2	socle pour Xbee			0.1	0.2
1	jumper	distrelec	121538	0.7	0.7
1	resistance 1M			0.1	0.1
2	resistance 10K			0.1	0.2
1	résistance 1K			0.1	0.1
1	borne alimentation à vis	distrelec	140097	1.4	1.4
2	borne enfichable			1	2
2	diodes			0.1	0.2
2	transistor	distrelec	612754	0.48	0.96
1	led verte	distrelec	632031	2.37	2.37
2	bouton poussoir	distrelec	200525	1.72	3.44
1	transformateur 230-3,3V	farnell	1242632	49.2	49.2
1	boitier 120X66X40	distrelec	300678	14.63	14.63
				total	104.64

Carte principale

quantité	déniomination	fournisseur	n°	prix	sous total
1	module Xbee	reselec	XB24-Z7CIT-004	21.6	21.6
2	socle pour Xbee			0.1	0.2
1	microcontrôleur	microchip	18F2525	5	5
1	socle Uc	distrelec	651933	1.18	1.18
10	condensateur 1uF	distrelec	838361	0.41	4.1
2	condensateur 10nF	distrelec	830664	0.1	0.2
1	jumper	distrelec	121538	0.7	0.7
8	résistance 1M			0.1	0.8
1	résistance 47k			0.1	0.1
1	résistance 10K			0.1	0.1
3	résistance 4.7K			0.1	0.3
2	résistance 1K			0.1	0.2
1	transistor	distrelec	612754	0.48	0.48
1	horloge en temps réel	distrelec	643452	8.07	8.07
1	led rouge	distrelec	254450	1.29	1.29
1	led verte	distrelec	632031	2.37	2.37
1	capteur de température	farnell	9724761	9.1	9.1
1	affichage lcd graphique	distrelec	661498	23.89	23.89
3	bouton poussoir	distrelec	200525	1.72	5.16
1	bouton poussoir rond (5)	farnell	1390525	23.65	23.65
1	connecteur rond 3.3V mal	distrelec	111575	1.9	1.9
1	connecteur rond 3.3V femelle	distrelec	111588	2.3	2.3
1	boitier	distrelec	300890	19.26	19.26
1	transformateur 230-5V	distrelec	920032	36.58	36.58
1	abaisseur de tension max1658esa	distrelec	640291	9.9	9.9
1	rétro éclairage jaune pour DOGM132	distrelec	661460	6.67	6.67
				total	185.1

Prix global	**432.92 frs**

Cahier des charges de l'appareil mural

1. Introduction

L'appareil mural est destiné à commander la position du store en fonction de critères paramétrés par l'utilisateur. Un capteur de luminosité placé à l'extérieur de l'habitation ainsi qu'un capteur de présence placé à l'intérieur de l'habitation permettent d'automatiser la position du store en l'absence de l'utilisateur. Les informations de ces 2 capteurs sont transmises vers l'appareil mural qui les analyses et les transmets à la commande motorisée du store. Les échanges d'informations entre les différents appareils sont basés sur le protocole de transmission sans-fil Zigbee.

2. Interface utilisateur

L'appareil sera équipé de plusieurs touches et d'un affichage LCD permettant de le paramétrer et d'afficher les informations pertinentes à l'utilisateur.

2.1 Touches

L'appareil sera équipé des touches principales suivantes :

-. 1 touche pour descendre le store (▼)
-. 1 touche pour monter le store (▲)
-. 1 touche pour arrêter le mouvement du store (■)

-.1 commutateur Man/Aut pour la sélection du mode Manuel / Automatique

De plus, les touches secondaires suivantes permettront la configuration de l'appareil :

-. 1 touche pour afficher le menu
-. 1 touche pour défiler en bas dans le menu + incrémenter un chiffre (heure, délai, …)
-. 1 touche pour défiler en haut dans le menu + décrémenter un chiffre (heure, délai, …)
-. 1 touche pour valider une sélection

2.2 Affichage

En fonctionnement normal, un affichage LCD permettra de visualiser les informations suivantes :

-. Heure, minute
-. Jour de la semaine (1, 2, …7)
-. Etat capteur de présence
-. Etat capteur de luminosité
-. Etat de la commande du store (▲,▼)

Lors de la configuration de l'appareil, les valeurs des paramètres spécifiés dans le chapitre suivant pourront être visualisées et modifiées.

3. Configuration de l'appareil

3.1 Menu Date/Heure

Ce menu permet de régler la date (année, mois, jour) et l'heure (heure, minute) de l'appareil.
Afin d'optimiser la place sur l'affichage, seul un chiffre correspondant au jour de la semaine sera affiché.

Lundi : 1
Mardi : 2
Mercredi : 3
…
Dimanche : 7

Le passage de l'heure d'été à l'heure d'hiver, et inversement, ne sera pas réalisé automatiquement par l'appareil et devra être ajusté par l'utilisateur.

3.2 Modes de fonctionnement

On distingue 2 modes de fonctionnement :

1. Le mode manuel
2. Le mode automatique

3.2.1 Mode manuel

Ce mode est sélectionné en positionnant le commutateur Man/Aut sur Man.

Le mode manuel permet à l'utilisateur de monter, de descendre ou d'arrêter le store à l'aide des touches (▲,▼, ■).

Remarque : Les commandes du mode manuel sont prioritaires aux commandes du mode automatique.
!

3.3 Mode Automatique

Ce mode est sélectionné en positionnant le commutateur Man/Aut sur Aut.

En mode automatique, l'appareil possède 2 fonctions distinctes :

-. Fonction Astro
-. Fonction Energy Save

3.3.1 Fonction "Astro"

La fonction "Astro" adapte la montée et la descente du store aux heures quotidiennes de lever et de coucher du soleil.

Les heures de lever et de descente du store sont calculées automatiquement par l'appareil selon l'exemple ci-dessous.

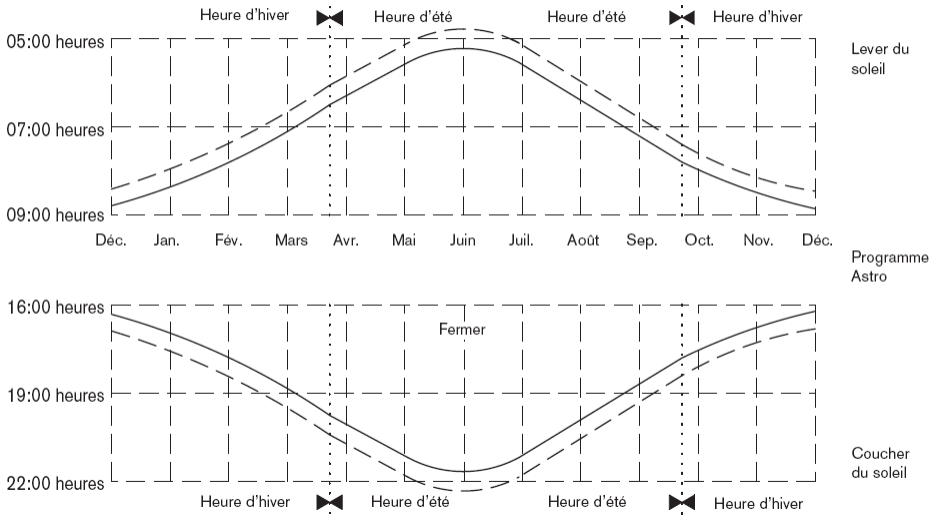


Figure 1: Fonction Astro.

Paramètres configurables :

-. Correction heures : +- 120 [min] (permet de retarder ou d'avancer le temps astronomique)
-. Position intermédiaire : x [%] (permet de positionner le store à une position finale quelconque)
x [%] est converti en un temps de descente en [sec.].
-. Angle Aération : x [°] (permet d'orienter les lamelles des stores en oblique)
x [°] est converti en un temps en [sec.] afin de positionner les lamelles avec l'angle choisi.

3.3.2 Fonction "Energy Save"

Cette fonction a pour objectif d'optimiser l'apport d'énergie solaire à l'intérieur de l'habitation pendant l'hiver et de préserver la fraîcheur de l'habitation pendant l'été.
Grâce à un capteur de présence, cette fonction ne s'active que si aucune personne ne se trouve dans la pièce.

Algorithme appliqué pendant l'hiver :

Si Ensoleillé ET Pièce non-occupée → Monter le store
Si Couvert ET Pièce non-occupée → Descendre le store

Si Pièce occupée → Aucune commande du store

Algorithme appliqué pendant l'été :

Si Ensoleillé ET Pièce non-occupée → Descendre le store
Si Couvert ET Pièce non-occupée → Monter le store

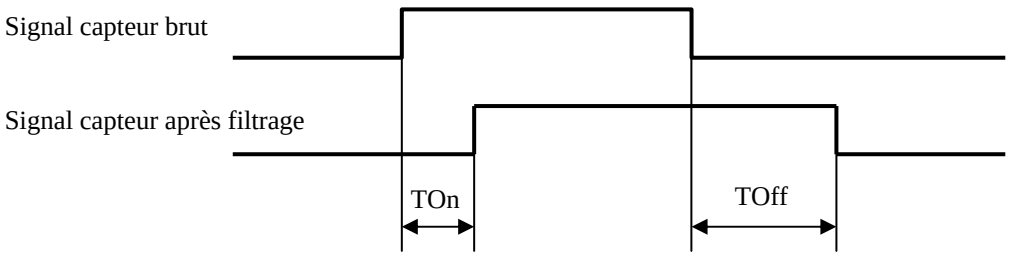
Si Pièce occupée → Aucune commande du store

Paramètres configurables :

-. Délai détecteur de présence activé (TOn) : x [sec.]
-. Délai détecteur de présence désactivé (TOff) : x [sec.]
-. Délai détecteur de luminosité activé (TOn) : x [sec.]
-. Délai détecteur de luminosité désactivé (TOff) : x [sec.]

Remarque : afin d'éviter que le store ne se déplace de manière automatique dans un délai trop court (par ex. : lors du changement de pièce de l'occupant d'une maison individuelle pendant quelques minutes) le délai du détecteur de présence désactivé sera plus important que dans le cas d'une salle de classe, de conférence, …

Principe du filtrage des capteurs de présence et d'ensoleillement :



4. Divers

-.En appuyant sur une combinaison de touches, l'appareil doit pouvoir être initialisé avec les valeurs des paramètres par défaut.