ENSIAME

Université de Valenciennes et du Hainaut-Cambrésis

SMARTWATCH

Ingénierie microcontrôleur

Projet:

Conception d'une montre connectée

Achbah Fatima Zahra
Camille Fournet
Aideen Hervé

Table des matières

Introduction

Le projet SmartWatch, requis dans le cours des microcontrôleurs, consiste à créer une montre connectée. Cette montre devra comporter plusieurs composants proposés par notre professeur. Elle devra afficher l'heure, afficher le nombre de pas que l'on a effectué le matin ainsi que ceux effectués l'après-midi. De plus celle-ci doit pouvoir également nous indiquer le nombre de calories dépensées. L'affichage doit être sur un écran LCD. La montre doit également communiquer des informations à travers un module Bluetooth.

Dans ce document vous trouverez les différentes étapes suivies pour accomplir ce projet.

Chapitre 1

Cahier des charges

La montre doit avoir 4 modes :

- **Mode Horloge:** Ce mode doit donner l'heure sur 6 digits « .. : .. : .. »
  Une mise à l'heure : Manuelle : 3 boutons et programmé sur PC
  Précision : 1H/mois
- **Mode de mesure des pas :** Matin et après-midi.
- **Mode mesure des calories :** poids, taille, âge, sexe.
- **Mode connexion PC**

Il faut aussi respecter la liste des composants imposé par le constructeur :

- Accéléromètre : ADXL345
- BLE312
- Afficheur 162C
- Microcontrôleur : C8051F912

Chapitre 2
Composants

[1]

1. Microcontrôleur C8051F912

Le microcontrôleur sous notre disposition est un MicroC fabriqué par SiliconLabs.

Ce microcontrôleur à un microprocesseur 8051 et un Interface de débogage intégrée au système, à pleine vitesse et non instructive (sur puce). Un CAN de 10 bits, une référence de courant programmable 6bits et un oscillateur interne programmable de 24,5 MHz avec technologie à spectre étalé. Il a une mémoire Flash sur puce de 8ko, 4ko, ou 2ko. Ainsi, il a une mémoire vivante RAM de 512 octets. Il a une interface série SMBus / 12C, UART améliorée et SPI. [1]

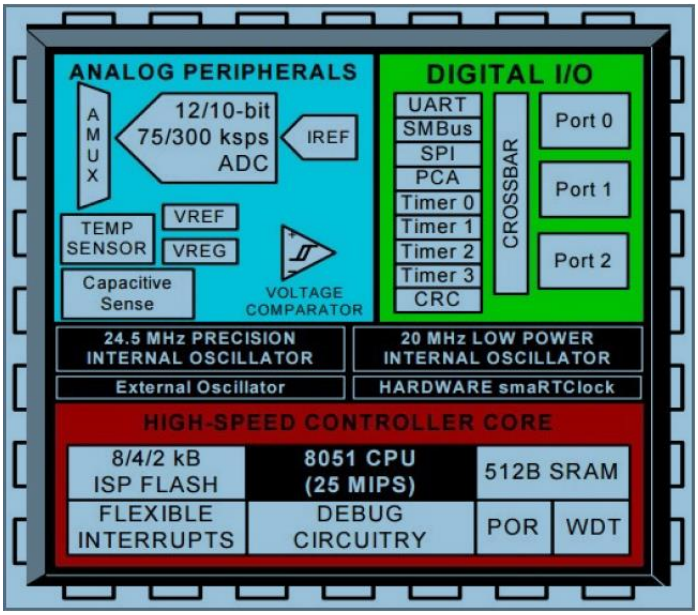


Figure 1: Schéma descriptif du μC

Les Micros C8051F99* ont 4 Timers 0,1,2,3 qui peuvent compter sous différents modes :

Timer 0 : Peut fonctionner sous 4 modes, un compteur de 13-bit ou uncompteur de 16-bit ou deux compteur de 8-bit ou un seul compteur de 8 bitsavec ou sans Auto-Reload.
Timer 1 : Peut fonctionner sous 3 modes, un compteur de 13-bit ou uncompteur de 16-bit ou un compteur de 8 bits avec ou sans Auto-Reload.

Timer 2 : Peut fonctionner sous 3 modes, un compteur de 16-bit ou un compteur de 8-bit avec ou sans Auto-Reload, et le mode comparateur 0.
Timer 3 : Peut fonctionner sous 3 modes, un compteur de 16-bit ou uncompteur de 8-bit avec ou sans Auto-Reload, et le mode capture horloge externe.
Dans la suite on utilisera les Timers pour réaliser l'Horloge. On aura besoin d'un Timer 16-bits avec Auto-Reload. Le choix du mode Auto-Reload est fait car il peut entrer dans une boucle qui compte jusqu'à une valeur fixée par l'utilisateur et déclenche une interruption et après elle recommence le comptage de cette valeur en parallèle avec l'exécution du programme. Entre ces 4 Timers on choisit le Timer 2 (ou 3).

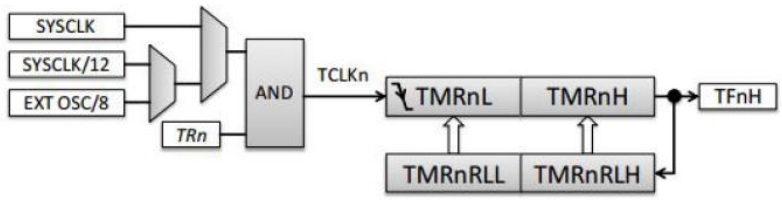


Figure 2: Timer 2(ou 3) mode Auto-Reload *n : nb de bits

UARTest une interface asynchrone universel Récepteur / émetteur, il sert à faire communiquer le MicroC avec des dispositifs distants. Les dispositifs de communication ont un TX et une broche d'entrée du récepteur RX. L'interface UART est symétrique donc n'importe quel côté peut envoyer des données de façon asynchrone. Donc on a un transfert de données bidirectionnel. La figure suivante représente les possibilités de connexion. On utilisera ce type de communication avec le module BluE. [1]

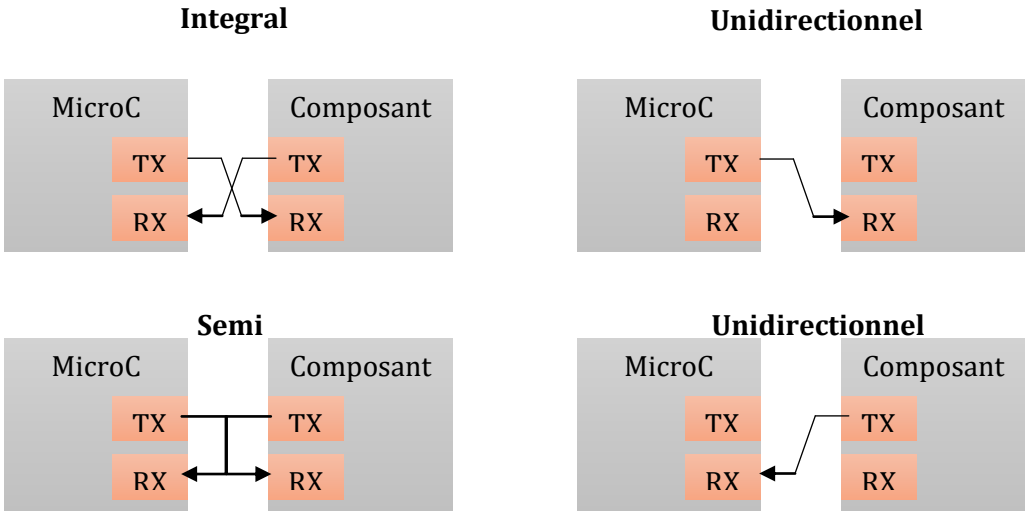


Figure 3: Types de Transmission UART

SPI est une interface périphérique série. L'interface SPI est utilisée pour connecter des circuits intégrés-capteurs. SPI utilise un fil pour les données sortantes et un autre pour les données d'entrées. Il existe aussi un 3eme fil qui est piloté par le maître qui sert à faire la synchronisation du transfert en fournissant un signal Clock. On utilisera ce type de communication avec ADXL345.

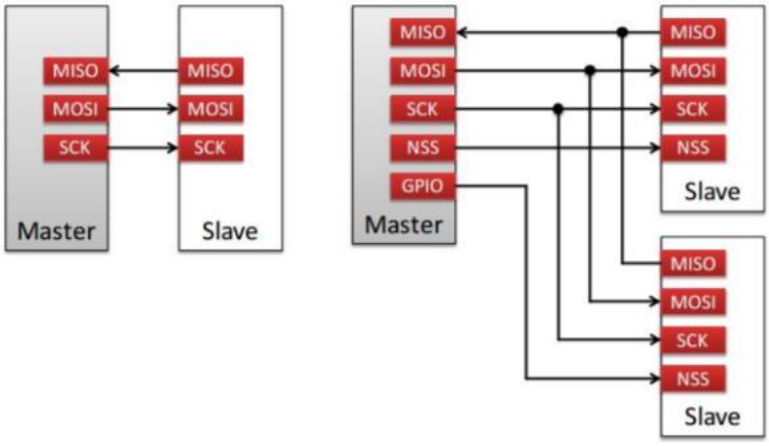


Figure 4: Connections Master Slave

2. Module Bluetooth BLE112

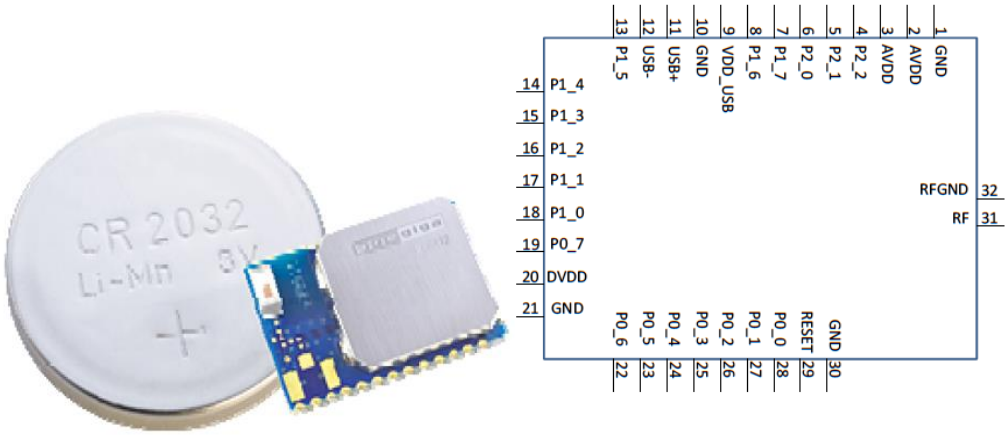


Figure 5: BLE112: Description des Pins

Le **BLE112** a un seul mode de communication UART et il est de basse énergie, donc il plus convenable pour une montre connectée. Le rôle de ce composant est de faire la communication entre le téléphone (Application Android) et la montre.

3. Afficheur 162C

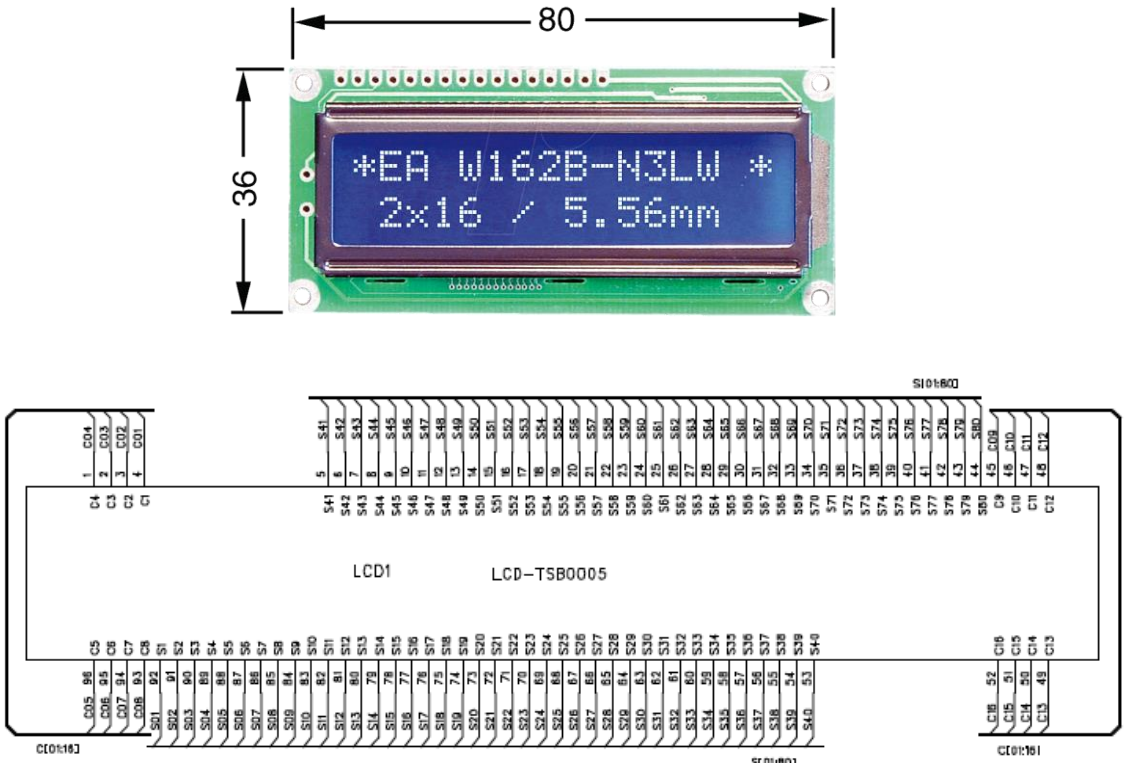
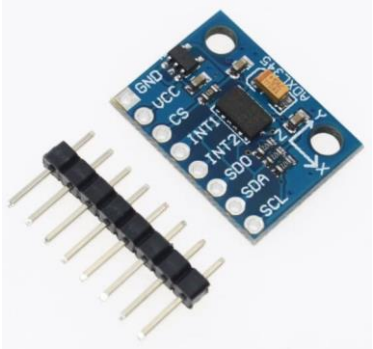


Figure 6: Afficheur 162C (2*16) description des Pins

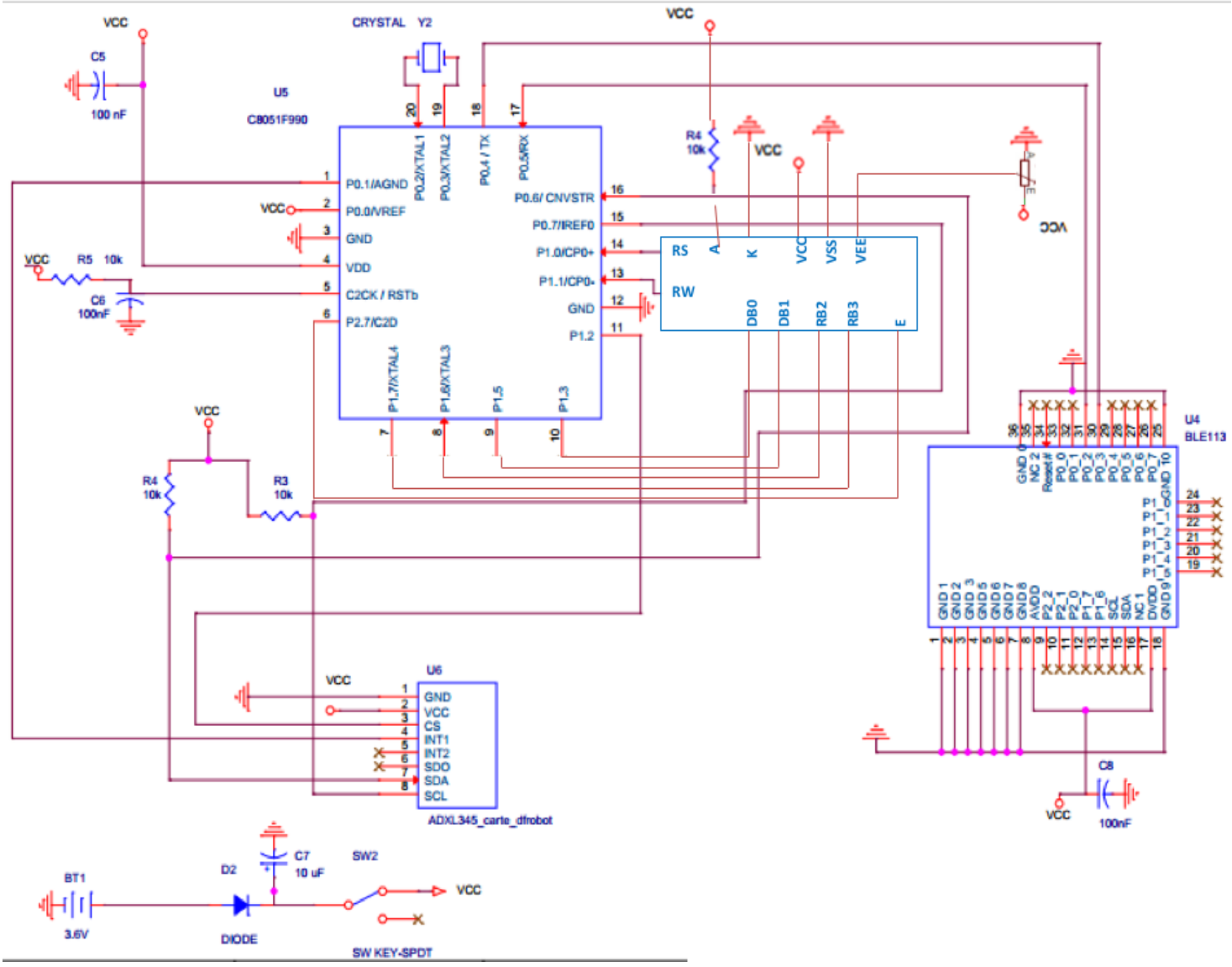
L'afficheur est composé de 2 lignes, chaque ligne à 16 cases (16 bits). On utilisera la communication I2C.

4. Accéléromètre ADXL345



Ce composant est un capteur qui permet de mesurer l'accélération linaire. ADXL345 est un capteur et il sera fixé sur notre Smart Watch. Il peut communiquer sous deux modes I2C et SPI. Or I2C est utilisé par 162C donc on utilisera la communication SPI.

5. Schéma électronique :



Chapitre 3

Programmation

Pour ce type de projet les langages de programmation les plus adaptés sont C/C++, Arduino, Assembleur. Les langages C/C++, Arduino(C+ Java) sont des langages de haut niveau, donc sont plus facile à programmer mais ils ne sont pas proche de la machine (µC), le langage Assembleur étant un langage de bas niveau, il est donc très proche de la machine mais difficile à programmer avec.

Nous avons choisi le langage assembleur en dépit de sa difficulté pour mieux comprendre notre machine et la programmer d'une manière ultime.

Nous avons plusieurs composants et donc plusieurs modes, pour cette raison on a divisé le programme principale en plusieurs sous programmes traités indépendamment.

; Déclaration des variables

```
DIZMS EQU 30H
SEC EQU 31H
MINT EQU 32H
HEURE EQU 33H
PAS_MATIN_POIDS_FORT EQU 34H
PAS_MATIN_POIDS_FAIBLE EQU 35H
PAS_SOIR_POIDS_FORT  EQU 36H
PAS_SOIR_POIDS_FAIBLE EQU 37H
INT_ENABLE EQU 2EH
INT_MAP EQU 2FH
DATA_FORMAT EQU 40H
VITESSE EQU 41H
RAPIDE EQU 42H
MOYEN EQU 43H
LENT EQU 44H
K EQU 45H
Min EQU 46H
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

; Programme principal

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ORG 0H
SJMP RESET
ORGH 03H
INT_BOUTON:
        SJMP GESTION_MODE
        ORG 0BH
        RETI
ORG 13H
SJMP INT_PODOMETRE
ORG 1BH //Sécurité du programme
RETI
ORG 23H
SJMP INT_SERIAL
ORG  50H
SJMP INT_TIMER2
GESTION_MODE:
```

; Initialisation du Timer

```
MOV TMR3CN, #0000 0101 B
MOV TMR3RLL, #E0H
SETB TR3
MOV DIZMS, #00H
MOV SEC, #00H
MOV MINT, 00H
MOV HEURE, #00H
```

; Code Timer

```
ORG 0073H
PUSH A
INT_TIMER2: DJNZ DIZMS, SORTIE
MOV DIZMS, #100
```

```
INC SEC
MOV A, SEC, #00H
INC MINT
MOV A, MINT
CJNE A, #60, SORTIE
MOV MINT, #00H
INC HEURE
MOV A, HEURE
CJNE A, #24, SORTIE
MOV HEURE, #00H
SORTIE : POP A
         RETI
```

; Sous-programme Afficheur LCD

; Table ASCII

```
TABLE_ASCII : DB, #30H // Numero 0
              DB, #31H // Numero 1
              DB, #32H // Numero 2
              DB, #33H // Numero 3
              DB, #34H // Numero 4
              DB, #35H // Numero 5
              DB, #36H // Numero 6
              DB, #37H // Numero 7
              DB, #38H // Numero 8
              DB, #39H // Numero 9
              DB, #3AH // :
```

; Temps

```
TEMP_ASCII: MOV A, R0
DA A
ANL A,#F0 H
```

```
SWAP A
MOV DPTR,#TABLE_ASCII
MOV A,@A+DPTR
```

; Sous-programme Compteur des pas

```
PAS_ASCII : MOV A, R0
            DA A
            ANL A, #F0 H
            SWAP A
            MOV DPTR, #TABLE_ASCII
            MOVC A,@A+DPTR
RESET :     MOV PAS_MATIN_POIDS_FORT, #00H
            MOV PAS_MATIN_POIDS_FAIBLE, #00H
            MOV PAS_SOIR_POIDS_FORT, #00H
            MOV PAS_SOIR_POIDS_FAIBLE, #00H
            MOV INT_ENABLE, #00H
            MOV INT_MAP, #00H
            MOV DATA_FORMAT, #00H

INIT :      MOV INT_ENABLE, #01000000
            MOV INT_MAP, #10111111
            MOV DATA_FORMAT, #00000010

INT_PODOMETRE : PUSH A
                MOV A, HEURE
                SUB A, #12
                JB (PSW.1), MATIN
                SHORTJUMP SOIR

MATIN :     INC PAS_MATIN_POIDS_FAIBLE
            MOV A, PAS_MATIN_POIDS_FAIBLE
            CJNE A, #256, MATIN
            MOV PAS_MATIN_POIDS_FAIBLE, #00
```

```
                INC PAS_MATIN_POIDS_FORT
                MOV A, PAS_MATIN_POIDS_FORT
                JZ INT_PODOMETRE

SOIR :          INC PAS_SOIR_POIDS_FAIBLE
                MOV A, PAS_SOIR_POIDS_FAIBLE
                CJNE A, #256, SOIR
                MOV PAS_SOIR_POIDS_FAIBLE, #00
                INC PAS_SOIR_POIDS_FORT
                MOV A, PAS_SOIR_POIDS_FORT
                SHORTJUMP END

END :  POP A
       RETI
```

; Sous-programme Calories

```
    RESET :     MOV min ,#00h
                MOV INT_ENABLE, #00h
                MOV INT_MAP, #00h
                MOV DATA_FORMAT, #00h

    INIT :      MOV INT_ENABLE, #01000000
                MOV INT_MAP, #10111111
                MOV DATA_FORMAT, #00000010

    INT_CALORIE : PUSH A        ;Conserve les données de l'accumulateur
                  PUSH B        ;Conserve les données de l'accumulateur auxiliaire
                  MOV A, VITESSE    ;Déplace dans l'accumulateur les données des vitesses
                                    récupérées dans le programme de l'accéléromètre
                  MOV B, MINUTE
                  JNZ A, #9, RAPIDE ;Quand la vitesse est supérieure à 9mk/h saut à
                                    l'étiquette rapide
                  SUB A,#4          ;On enlève 4 à la vitesse
                  JB (PSW.1), LENT  ;Si vitesse négative alors vitesse < 4km/h
```

```
                JB (PSW.0), MOYEN  ;Si vitesse supérieur à 0 → 4<v<8 km/h

RAPIDE :        INC min            ;A chaque minute où la vitesse est supérieure à
                                   9km/h, on incrémente le compteur
                MOV B, min
                DIV B,#60          ;Transforme les minutes en heures
                MOV K,#750         ;K coefficient calcul des calories
                MUL K,B            ;Multiplie le coefficient par la durée de l'effort
                MOV A,K            ;Met dans l'accumulateur la valeur calculée avant
LENT :          INC min             ;A chaque minute où la vitesse est inférieure à
                                   4km/h, on incrémente le compteur
                MOV B, min
                DIV B,#60          ;Transforme les minutes en heures
                MOV K,#200         ;K coefficient calcul des calories
                MUL K,B            ;Multiplie le coefficient par la durée de l'effort
                MOV A,K            ;Met dans l'accumulateur la valeur calculée avant
MOYEN :         INC min             ;A chaque minute où la vitesse est comprise entre
                                   4 .1 et 8km/h, on incrémente le compteur
                MOV B, min
                DIV B,#60          ;Transforme les minutes en heures
                MOV K,#200         ;K coefficient calcul des calories
                MUL K,B            ;Multiplie le coefficient par la durée de l'effort
                MOV A,K            ;Met dans l'accumulateur la valeur calculée avant
                SHORTJUMP END
END :           POP A
                POP B
                RETI
```

; Sous-programme Bluetooth

```
MOV TMOD, #20H // timer 3, mode2
MOV TH3, #-3 //9600baud
MOV SCON,#50H // 8-bit,1stop,REN active
SETB TR3
```

```
ENVOI_ HEURE:   MOV SBUF, HEURE
ICIH :          JNB TI,ICI//Attendre l'envoi dernier bit
                CLR TI
                SJMP ENVOI_MINT

ENVOI_MINT :    MOV SBUF, MINT
ICIM :          JNB TI, ICI
                CLR TI
                SJMP ENVOI_SEC

ENVOI_SEC :     MOV SBUF, SEC
ICIS :          JNB TI, ICI
                CLR TI
                SJMP ENVOI_PAS

ENVOI_PAS1 :    MOV  SBUF , PAS_MATIN
ICIP1 :         JNB TI,ICI
                CLR TI
                SJMP ENVOI_PAS2
ENVOI_PAS2 :    MOV SBUF, PAS_SOIR
ICIP2 :         JNB TI, ICI
                CLR TI
                SJMP ENVOI_HEURE
```

Bibliographie

[1] «DataSheet».

[2] B. Odant, Microcontôleurs 8051 et 8052 et mise en oeuvre, DUNOD.

[3] p. KAUFFMANN, mise en oeuvre et applications du microcontrôleur 8051, MASSON.

Annexe

Langage python des calories

```
v=Vitesses
m=Minutes
M=0
c=0 //compteur heure
k=0 //coeff de vitesse
R=0 // résultat calories
If v=<4 :
    k=200
    M=M+m
    C=(k*M)/60
If v>=9 :
    k=750
    M=M+m
    C=(k*M)/60
Else
    k=400
    M=M+m
    C=(k*M)/60
Return C
```