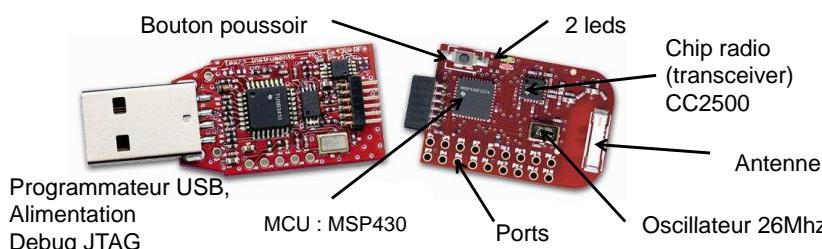


# Wireless sensor network (Réseau de capteurs sans fil)

C. Belleudy  
(belleudy@unice.fr)  
University Nice – Cote d'Azur



## Case study : kit TI Z430-RF2500



### MCU - MSP430:

- Fréquence nominale : max 16 Mhz – 4 fréquences, 4 tensions (1,8V – 3,6V)
- 5 modes repos : LPM0..LMP4

### Transceiver CC2500 - Transmission (TX)

- output power : programmable from -30dbm to 0dbm
- Modulation,
- Data rate,
- Period ...

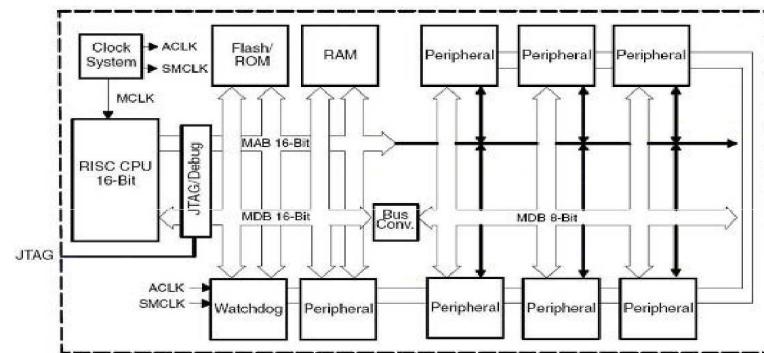
3

## Plan

1. Fonctionnalité de base de la plateforme EZ430
2. Programmation du MSP 430
3. Environnement de programmation
4. Communication entre deux noeuds
5. Conception d'une application liée à la gestion de la température

2

## Cas particulier : EZ430-2500



- Micicontroleur : 16-bit 16MHz RISC
- 32kB ROM, 1kB RAM
- 2 Timers, USCI, 10-bit ADCs
- 16 registres
- Clock tick (programmable) : 1mhz (1μs)
- 1 instruction est exécutée à chaque cycle (fetch de la mémoire ROM)

4

## Quelques points intéressants

- Programmation => mise en ROM du programme.
  - Exécution à partir de la première adresse du programme
  - Programmation de la radio (à configurer en début du programme)
  - Basée sur des interruptions :
    - Réception d'un signal
    - Gestion de l'interruption: le programme initial est dérouté par une fonction définie par l'ISR (Interrupt Routine Service)
- Dans notre cas, trois types d'interruption :
- Interne :
    - Pression sur un bouton (BP : bouton poussoir)
    - TIMER
  - Externe
    - Réception d'un packet,
- Timer : Registre sur 16 bits qui s'incrémente à chaque cycle (1Mhz), Test de la valeur du registre (programmable) => interruption.

5

## Plan

1. Fonctionnalité de base de la plateforme EZ430
2. **Programmation du MSP 430**
3. Environnement de programmation
4. Communication entre deux noeuds
5. Conception d'une application

7

## Les entrées/sorties du MSP430

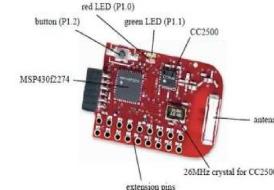
Le MSP430 possède 40 pins :

- fonctions analogiques : 4 pins pour alimenter la maquette + 2 pour des tests d'usine + 2 pour l'oscillateur externe;
- fonctions numériques : 32 pins => 4 ports de 8 pins

Notation des ports : P1, P2, P3, P4

Pin Y du port Px : Px.y (y représente la position de la pin dans le port x)

Toutes les pins peuvent être programmées en entrée ou en sortie  
(Registre de 8 bits de commande)



6

## Configuration des ports d'entrée/sortie

Configuration du port :

**PxDIR.y** configure le sens de la pin Px.y;  
output : PxDIR.y=1,  
input : PxDIR.y=0;

**PxOUT.y** : écriture sur le port Px.y

**PxIN.y** : lecture du port Px.y;

**PxREN** : connexion à une résistance interne.

**PxIE** : autorisation des interruptions

**PxIES** : sélection de la transition qui provoque l'IT

Bit = 1 => passage de 1 à 0

Bit = 0 => passage de 0 à 1

**PxFG** : flag de l'IT => mis à 1 lors de la détection d'une IT

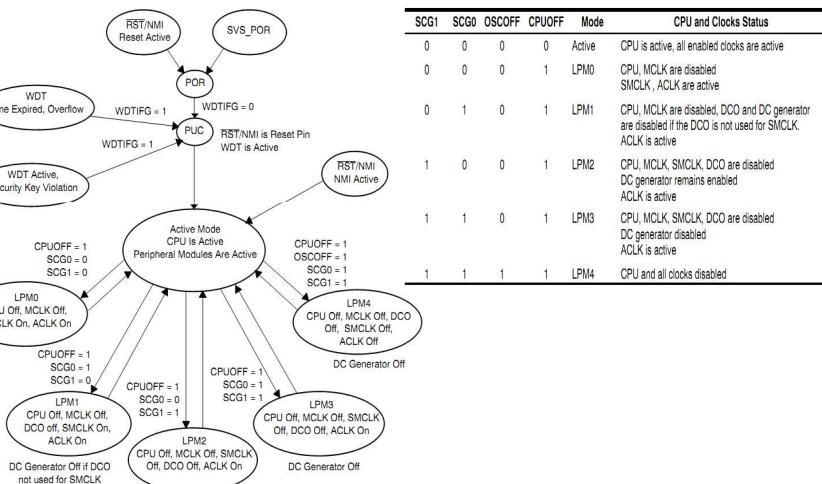
8

## Gestion des Interruptions

- Configurer les ports
  - Autorisé les interruptions
    - Indiquer le type de transition qui déclenche l'interruption
    - sur le port préciser le ou les bits qui vont déclencher l'interruption (PxIE)
    - dans le status register, GIE doit être placé à 1
- ```
_bis_SR_register(GIE);
```
- Pas d'importance sur ce nom
- Ne pas oublier de mettre à 0 le flag de l'IT sinon IT permanente

9

## Les modes repos

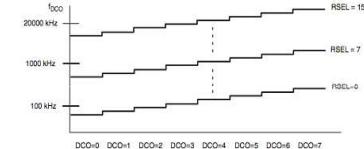


11

## Horloges et TIMER

### Horloge :

- DCOCLK - Internal High Speed Oscillator up to 16MHz
- VLOCLK - Very Low Frequency (12kHz) oscillator



### Gestion du TIMER

TACCR0 => fin de comptage

TACCR1 => début de comptage

Gestion de l'IT relative au timer

**Reset du timer : TACTL = TACLR**

**Activation de l'IT : TACCTL0 = CCIE;**

**TACTL = TASSEL\_1 + MC\_1;**

Choix de l'horloge, ici ACLK

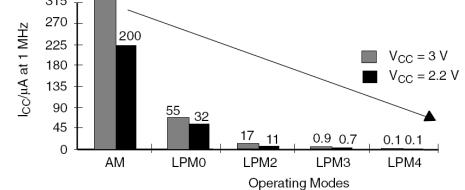
Type de comptage : up, down

10

## Les Mode repos

| CPU Operating Mode    | Clock Source  | Maximum DMA Cycle Time                 |
|-----------------------|---------------|----------------------------------------|
| Active mode           | MCLK=DCOCLK   | 4 MCLK cycles                          |
| Active mode           | MCLK=LFXT1CLK | 4 MCLK cycles                          |
| Low-power mode LPM0/1 | MCLK=DCOCLK   | 5 MCLK cycles                          |
| Low-power mode LPM3/4 | MCLK=DCOCLK   | 5 MCLK cycles + 6 $\mu$ s <sup>†</sup> |
| Low-power mode LPM0/1 | MCLK=LFXT1CLK | 5 MCLK cycles                          |
| Low-power mode LPM3   | MCLK=LFXT1CLK | 5 MCLK cycles                          |
| Low-power mode LPM4   | MCLK=LFXT1CLK | 5 MCLK cycles + 6 $\mu$ s <sup>†</sup> |

Temps de transition entre les modes



## Quelques opérations logiques utiles

### Quelques opérateurs de base :

Fonction OU : |

Fonction ET : &

Fonction XOR : ^

Inversion : ~

Décalage : << >>

### Exemple :

Initialement => A = 0b01101001,

~A = 0b10010110

A |= 0b00000010 => A=0b01101011

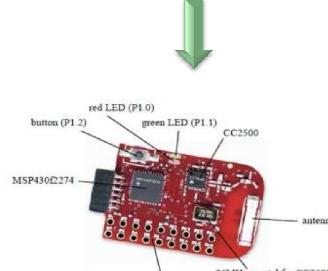
A &= 0b00001000 => A=0b01100001

A ^= 0b10001000 => A=0b11100001

A >> 2 => A=0b10100100

A << 2=> A=0b00011010

Mise à zéro/un d'un bit  
Fonction toggle ?



13

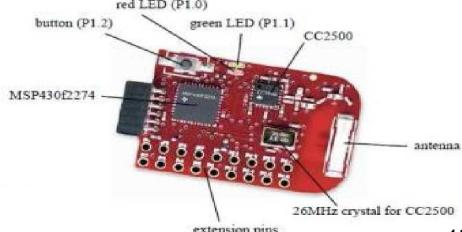
## Led\_button.c

### Exemple 2 : Action du BP & interruption

```
#include "io430.h"
#include "in430.h"
int main( void )
{
    WDTCTL = WDTPW + WDTHOLD;
    P1DIR |= 0x03;
    P1DIR &= ~0x04; Résistance interne pour les boutons poussoirs
    P1REN |= 0x04;
    P1IE |= 0x04;
    __bis_SR_register(GIE); Activation des IT sur le port 1 puis globale
    while(1);
}
```

```
#pragma vector=PORT1_VECTOR
interrupt void Port_1 (void)
{
    P1IFG &= ~ 0x04; Reset de l'IT
    P1OUT ^= 0x03;
}
```

Fonction toggle



15

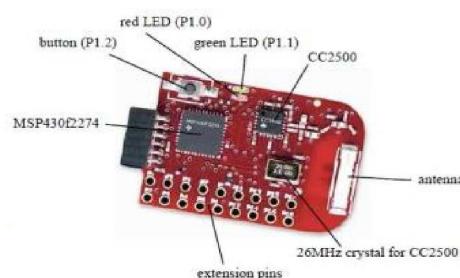
## Led\_steady.c

### Exemple 1 : allumer les leds de la carte

```
#include "io430.h"
int main( void )
{
    WDTCTL = WDTPW + WDTHOLD; Désactivation du watchdog (reset carte)
    P1DIR |= 0x03; Configuration => OUTPUT
}
```

```
P1OUT |= 0x03;
while(1);
}
```

Valeur de la sortie



14

## Led\_timer.c

### Exemple 3 : Utilisation du TIMER & IT

```
#include "io430.h"
#include "in430.h"
int main( void )
{
    WDTCTL = WDTPW + WDTHOLD;
    P1DIR |= 0x03;
    BCSCTL3 |= LFXT1S_2; Activation du timer
}
```

```
TACCTL0 = CCIE; Activation de l'IT du timer
TACCR0 = 1000;
TACTL = TASSEL_1+ MC_1; Valeur max
Mode de comptage : tick de ACK => +1
__bis_SR_register(GIE+LPM3_bit);
}
```

```
Activation des IT et mis en repos
```

```
#pragma
vector=TIMERA0_VECTOR
interrupt void Timer_A (void)
{
    P1OUT ^= 0x03;
}
```

Fonction toggle

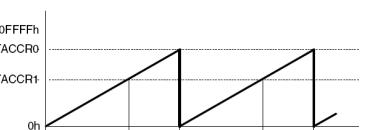
Oscillateur du TIMER (VLO) : 12 KHz

Période ?

Activation de l'IT du timer

Valeur max

Mode de comptage : tick de ACK => +1

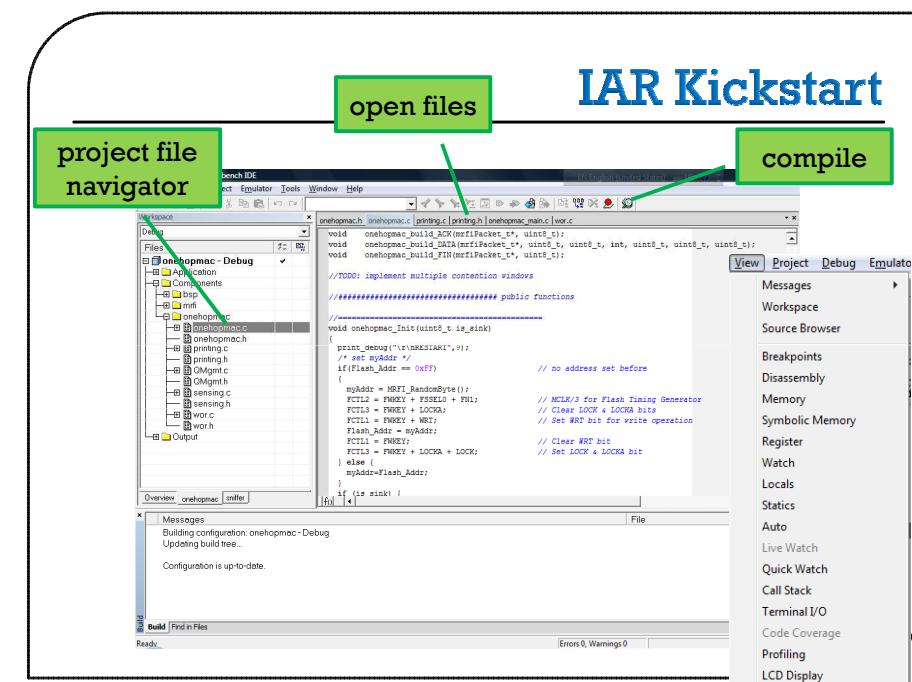


16

# Plan

1. Fonctionnalité de base de la plateforme EZ430
2. Programmation du MSP 430
3. Environnement de programmation
4. Communication entre deux noeuds
5. Conception d'une application liée à la gestion de la température

17



# IAR Kickstart

- ➊ Travailler dans le workspace donné en exemple : lab\_ezwsn.eww
- ➋ Un workspace contient plusieurs projets : Nom\_Projet.ewp

Un projet comprend Trois parties :

- Application => programme applicatif
- Components => drivers de la carte : MSP, sensing, RF ...
- Output

➌ Création d'un nouveau projet :

Afin de ne pas perdre le lien avec tous les drivers

- se rendre dans le répertoire TI\_WSN/source\_code/iar-v4.11
- Faire une copie projet souhaité (Copy Projet.ewp) puis le renommer (Projet\_v2.ewp)
- Insérer le nouveau projet dans le workspace : project/add existing project (sélectionner Projet\_v2.ewp)

19

- ➊ Un projet est lié au programme C qui sera téléversé dans le noeud: Afin de ne pas modifier les programmes d'exemples : dans le répertoire C\_files, copier/coller le fichier file.c, le renommer file\_v2.c puis project/add file selectionner file\_v2.c, puis remove ancien fichier (file.c).
- ➋ Pour rendre un projet actif => clique droit, set as active (ou cliquer deux fois sur ce projet)  
*=> attention, les commandes de compilation, téléchargement ... sont réalisées pour le projet actif.*
- ➌ Compilation (Build pour un nouveau projet)
- ➍ Programmation et debugging sur le MOTE qui est connecté à la clé USB
- ➎ Possibilité de positionner des breakpoints

20

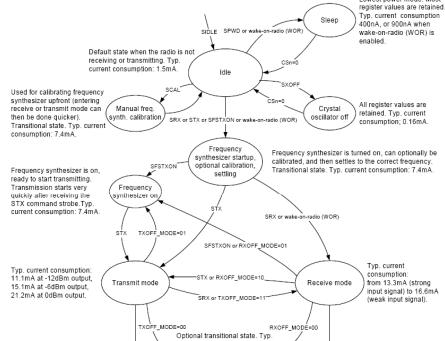
# Plan

1. Fonctionnalité de base de la plateforme EZ430
2. Programmation du MSP 430
3. Environnement de programmation
4. Communication entre deux noeuds
5. Conception d'une application

21

# Fonctionnement du RF2500

- ➊ Any frequency on the 2.4-2.485GHz band
  - Not 802.15.4-compliant
- ➋ 47 registres de configuration
  - Mise en repos ou activation de Tx/Rx
  - TXBUF, RXBUF
  - Puissance d'émission
  - fréquence du Tx
- ➌ Contrôle de la RF : machine d'état



22

# Trrx\_simple

## Programmation d'une communication

```
#include "mrfi.h"
#include "radios/family1/mrfi_spi.h" (à rajouter)
int main(void)
{
  BSP_Init();
  PIREN |= 0x04;
  P1IE |= 0x04;
  MRFI_Init();
  MRFI_WakeUp();
  MRFI_RxOn();

  bis_SR_register(GIE+LPM4_bits);

  void MRFI_RxCompleteISR() ←
  {
    P1OUT ^= 0x02;
  #pragma vector=PORT1_VECTOR
  _interrupt void Port_1(void)
  {
    P1IFG &= 0x04;
    mrfiPacket_t packet;
    packet.frame[0]=8+20;
    MRFI_Transmit(&packet, MRFI_TX_TYPE_FORCED);
    P1OUT ^= 0x01;
  }
}

Fonction appelée à la réception d'une trame
```

**Initialisation**

**Configuration de la radio**

**Activation du circuit radio**

**Mode Rx (activation des ITs)**

**2 MOTS :**  
Sous l'action du BP => envoie d'une donnée  
Le nœud qui émet allume la led rouge.  
Le nœud qui reçoit la led verte.

**Déclaration d'une trame (paquet)**

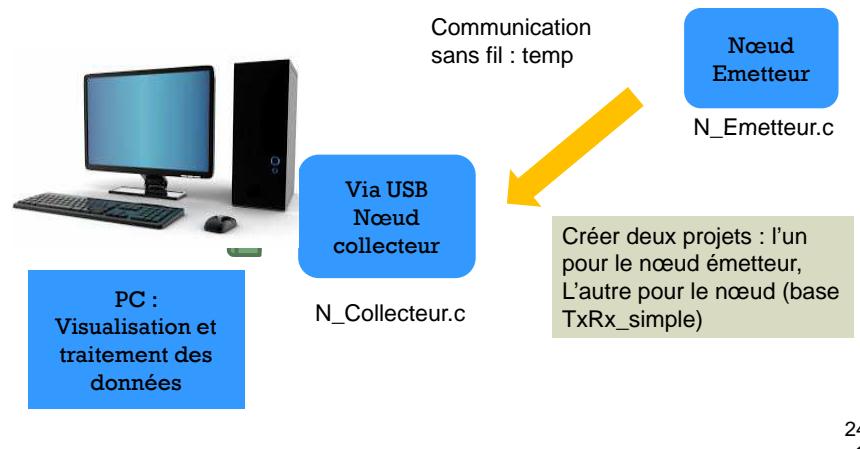
**Taille de la trame (28)**

**Copie de la trame dans la pile TX et transmission**

23

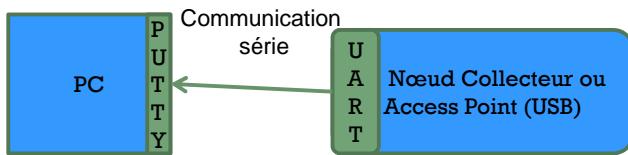
# Communication entre deux noeuds

## Architecture :



24

## Communication avec le PC et l'access Point



UART => PC

=> Output : chaîne de caractères  
TXString(output, (sizeof output))

Lecture des données avec le logiciel PUTTY, TERA TERM, python (Module pyserial) => nécessite au préalable l'initialisation de l'UART.

Initialisation UART :  
P3SEL |= 0x30;  
UCA0CTL1 = UCSSEL\_2;  
UCA0BR0 = 0x41;  
UCA0BR1 = 0x3;  
UCA0MCTL = UCBR8\_2;  
UCA0CTL1 &= ~UCSWRST;

25

## Plan

1. Fonctionnalité de base de la plateforme EZ430
2. Programmation du MSP 430
3. Environnement de programmation
4. Communication entre deux noeuds
5. Conception d'une application liée à la gestion de la température

27

## Lecture du port série en python

- Python : Anaconda/spider
- Module pyserial (conda install pyserial, se placer dans Scripts si pb)
- Exemple de lecture du port série (programme à compléter) :

*Lecture\_portcom.py :*

```
import serial as s
ser=s.Serial()
ser.baudrate = 9600
ser.port = "COM38"
ser.timeout=10
```

```
ser.open()
x=ser.readline()
print(x)
ser.close()
```

Configuration du port (9600 bauds)  
Donner le port COM associé  
Timeout : temps à partir duquel,  
une lecture sera considérée  
comme terminée si aucune data  
n'a été envoyée

Ouverture du port, lecture du port, puis  
fermeture du port

Ce programme ne permet qu'une seule lecture du port com,  
modifier le afin de faire plusieurs lectures consécutives (ne  
pas mettre de boucle infinie)

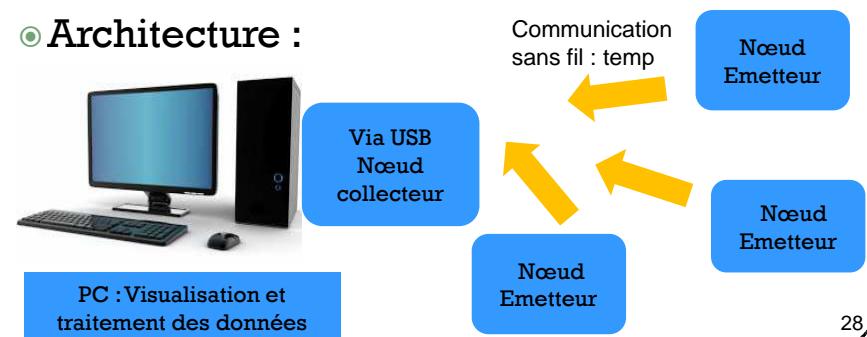
26

## Application : gestion de la température

Services :

- visualisation de l'évolution de la température,
- gestion chaudière,
- Alerte : détection feu.

Architecture :



28

## Nœud émetteur (simple\_sening): lecture temp + transmission

```
#include "mrfi.h"
#include "radios/family1/mrfi_spi.h"
__no_init volatile int tempOffset @ 0x10F4; // Temperature offset set at production

uint8_t get_adc(int input)
{ uint8_t value;
volatile long temp;
if (input==1)
{ ADC10CTL1 = INCH_10 + ADC10DIV_4; // select input, ADC10CLK/5
ADC10CTL0 = SREF_1 + ADC10SHT_3 + REFON + ADC10ON + ADC10IE +
ADC10SR; }
else
{ADC10CTL1 = INCH_11 + ADC10DIV_4; // select input, ADC10CLK/5
ADC10CTL0 = SREF_1 + ADC10SHT_2 + REFON + ADC10ON + ADC10IE +
REF2_5V; }
for( value = 240; value > 0; value-- ); // delay to allow reference to settle
ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
__bis_SR_register(CPUOFF + GIE); // LPM0 with interrupts enabled
value = ADC10MEM;
ADC10CTL0 &= ~ENC;
return value;
}
```

Fonction de lecture température,  
tension alim, ...

29

## Nœud émetteur : lecture temp + transmission

```
mrfiPacket_t packet;
packet.frame[0]=8+20;
for (i=0;<16;i++) {
packet.frame[i+9]=output[i]; }
MRFI_Transmit(&packet, MRFI_TX_TYPE_FORCED);
P1OUT ^= 0x01;
for (i=0;<32000;i++)
{value2=value2+1;
 value2=value2*value2; }
}
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void) {
__bic_SR_register_on_exit(CPUOFF); // Clear CPUOFF bit from SR
P1OUT ^= 0x02;
}
void MRFI_RxCompleteISR()
{
```



Tempo

31

## Nœud émetteur : lecture temp + transmission

```
int main( void )
{ int value0,value1,value2,pointer, degC, temp, volt, i;
BSP_Init();
//Initialisation de la radio
MRFI_Init(); .... => Modifier le canal
//Initialisation de l'ADC
P2DIR |= 0x02;
//Configuration UART -port USB
P3SEL |= 0x30; // P3.4,5 = USCI_A0 TXD/RXD
...
char output[] = " \r\n";
while(1) {
P2OUT ^= 0x02;
value0 = get_adc(1);

output[0] = '0'+((value1/100)%10);
output[1] = '0'+((value1/10)%10);
output[2]='.';
output[3] = '0'+(value1%10);
output [4]='C';
}
```

```
value1 = get_adc(2);
value1 = value1/2;

output[6] = '0'+((value1/100)%10);
output[7] = '0'+((value1/10)%10);
output[8]='.';
output[9] = '0'+(value1%10);
output[10]='V';

TXString(output, (sizeof output)-1);
```

30

## Nœud récepteur (Modifier TxRx\_simple)

- Ne pas oublier d'initialiser l'UART
- Modifier la fonction de réception afin de copier la trame reçue sur le port série

```
void MRFI_RxCompleteISR()
{
int i;
mrfiPacket_t packet;
MRFI_Receive(&packet);
char output[] = " ";
for (i=1;i<27;i++) {
output[i]=packet.frame[i];
}
TXString(output, sizeof output);
TXString("\n\r", 1);
//TXString("essai\n\r", 6);
P1OUT ^= 0x02;
}
```

Copie de la trame  
reçue dans une  
chaîne de caractère  
qui sera transmise au  
PC.

32

## Programme python

### Affichage la courbe de température par nœud :

```
Modules : matplotlib et numpy  
from matplotlib import pyplot as plt  
import numpy as np  
Créer un tableau puis : plt(tab)
```

### Affichage avec information temporelle :

```
import datetime  
date = datetime.datetime.now()  
str(date)=> '2018-02-19 09:58:51. ....'  
Accès aux différents champs :  
date.day, date.hour, date.minute, date.second, ...
```

### Sauvegarde dans un fichier :

- Création d'un fichier CSV (import csv)
- Quelques fonctions utiles :  
f=open('test1.csv','w'), f.write(x), f.close()  
fichier= csv.reader(open("detail.txt", "r"), delimiter=" ")

=> Programmer l'application finale

33

### A vous de concevoir l'application de

### l'objet au service :

- finaliser la programmation des nœuds,
- adapter la programmation python afin de visualiser les courbes de température,
- Mettre en place la gestion de la chaudière+ alerte incendie

34

## Exemple d'alerte en python

### Envoie de mail sur un compte gmail

```
import smtplib  
from email.mime.text import MIMEText  
from email.mime.multipart import MIMEMultipart  
  
msg = MIMEMultipart()  
msg['From'] = 'yyyyy@gmail.com'  
msg['To'] = 'ZZZZZ@gmail.com'  
msg['Subject'] = 'Alerte Température'  
message = 'Hausse température détectée !'  
msg.attach(MIMEText(message))  
mailserver = smtplib.SMTP('smtp.gmail.com', 587)  
mailserver.ehlo()  
mailserver.starttls()  
mailserver.ehlo()  
mailserver.login('yyyyy@gmail.com', 'password')  
mailserver.sendmail('yyyyy@gmail.com', 'ZZZZZ@gmail.com',  
    msg.as_string())  
mailserver.quit()
```

35