Étude des transmissions radio avec TVM et Xbee

Pierre-Yves Lucas*
Lab-STICC/WSN

23 février 2012

1 Protocole de mesure

La motivation est le contrôle temporel et énergétique lors de transmissions radio 2.4 GHz sur des matériels XBee et MSP430. Cette note sera développée pour mieux couvrir le MSP430 et les systèmes PSoC Cypress.

1.1 Le matériel

On utilise une carte Arduino ATMEGA 2560, comportant un microcontrôleur Atmel AVR 2560, et un module Xbee. Le module Xbee se place sur une carte d'alimentation intermédiaire (shield), couplée sur la carte Arduino : la communication entre le MCU et XBee est faite par un lien série (UART).

Le montage est alimenté par des accumulateurs en série, qui fournissent une tension continue de 6 V. Une résistance de 1 Ω est placée en série dans le montage. En mesurant la tension à ses bornes avec un oscilloscope, on obtient la valeur instantanée de l'intensité du courant. Ce montage permet de visualiser les variations de consommation d'énergie du microcontrôleur et du module Xbee. De cette façon, on peut connaître quelles sont les périodes d'activité et de veille du MCU, et les charges induites par la radio.

La mesure de l'intensité :

$$U = R.I, avec R = 1\Omega$$

.

1.2 Le logiciel

Un interpréteur Transputer Virtual Machine (TVM) [1] exécute un programme Occam chargé sur la carte. Le programme exécute en boucle une commande d'émission de paquet au module Xbee.

Le code chargé d'écrire le paquet sur l'UART est une primitive C que l'on a développé dans l'interpréteur. Elle assure actuellement seulement l'émission des paquets.

On a fait varier les paramètres suivants :

1. la durée entre deux itérations (11 ms minimum),
2. la taille du paquet émis (100 octets maximum),
3. le débit de données de l'UART (250 000 bauds maxi).

*avec Eloi Keita et Mahamadou Traoré

1. `http://concurrency.cc`

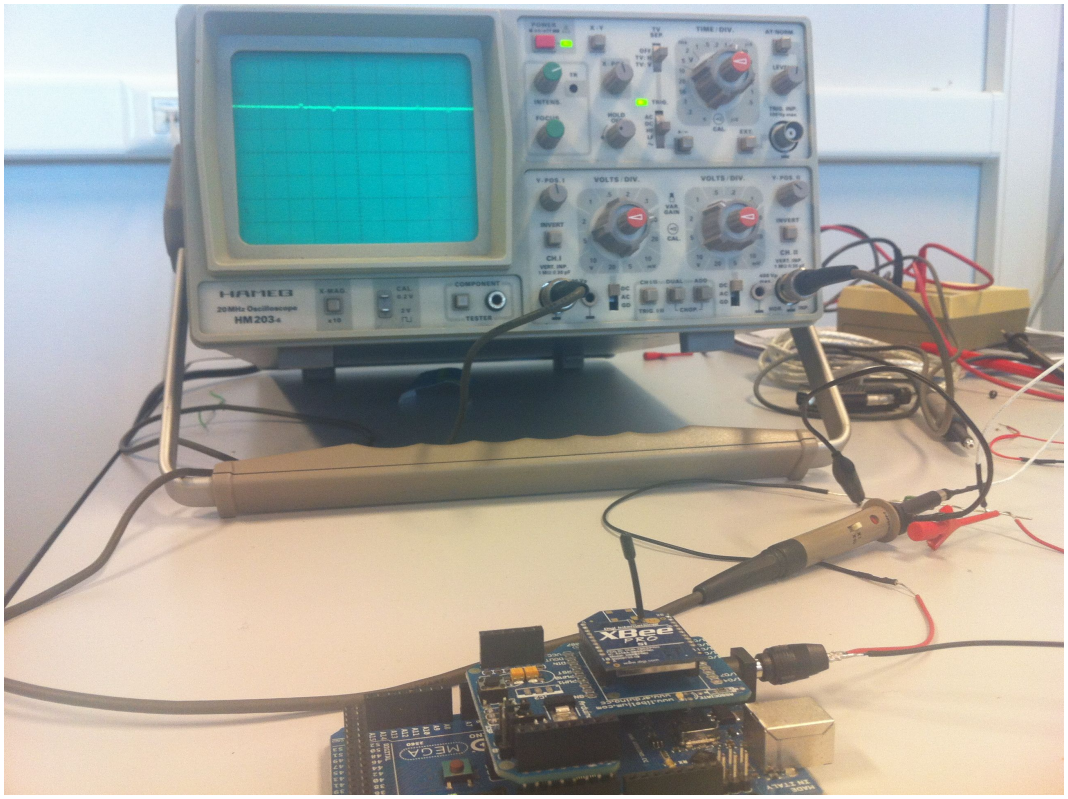

FIGURE 1 – Le montage expérimental côté Arduino : on repère le module XBee,

Dans le programme Occam, un chronomètre capture les estampilles temporelles encadrant l'appel de la procédure d'émission (Timer Occam).

Les estampilles récupérées à l'itération i sont copiées dans le paquet suivant qui est transmis à l'itération $i+1$.

On récupère les paquets radio avec un deuxième module Xbee, installé sur un module USB sur une station Linux. Les estampilles sont stockées dans un fichier, et l'on calcule les valeurs suivantes, pour chaque itération :

1. la durée de l'écriture sur le lien série (transmission radio)
2. l'intervalle entre deux itérations.

Pendant l'exécution du programme, on surveille la courbe de l'intensité affichée à l'oscilloscope.

2 Les résultats

Deux séries de mesures sont faites :

1. pour une vitesse de l'UART constante (9600 bauds), on fait varier la taille du paquet (21, 32, 64, 96, 100 octets de *payload*),
2. pour une taille de paquet constante, on fait varier la vitesse de l'UART entre l'Arduino et le Xbee.

Il n'y a pas de pause dans le programme entre chaque itération.
Les mesures sont données dans le tableau suivant :

Taille paquet (octet)	Vitesse UART (bds)	Copie vers UART (ms)	Délai entre transmission (ms)	Nb paquet/s	Débit UART (o/s)	Débit sans fil (o/s)
21	9600	32	11	31,25	656,25	488,37
32	9600	45	11	22,22	711,11	571,43
64	9600	81	11	12,35	790,12	696,65
96	9600	118	11	8,47	813,56	744,19
100	9600	123	11	8,13	813,01	746,27

Taille paquet (octet)	Vitesse UART (bds)	Copie vers UART (ms)	Délai entre transmission (ms)	Nb paquet/s	Débit UART (o/s)	Débit sans fil (o/s)
96	9600	118	11	8,47	813,56	744,19
96	19200	60	11	16,67	1600	1352,11
96	38400	30	11	33,33	3200	2341,46
96	57600	20	11	50,00	4800	3096,77
96	115200	10	11	100,00	9600	4571,43
96	250000	5	11	200,00	19200	6000,00

2.1 Consommation

Tension (V)	Sans Xbee	Avec XBee	En émission
7,29		150	340
7,34	80		
6,16	55		
6,09		120	260

Le débit de l'UART est de 9600 bds, chaque paquet fait 32 octets. Les valeurs idiquées sont l'intensité, exprimée en milliapères.

3 Interprétation

3.1 Délai entre chaque transmission

On observe que le délai entre deux émission consécutives est constant, indépendant de la taille du paquet et de la vitesse de l'UART. Cela correspond à l'exécution du code Occam qui prépare le paquet suivant avec les estampilles.

3.2 Durée de copie vers l'UART et débit effectif

En connaissant la durée de la copie d'un paquet, on déduit le nombre de paquets émis par seconde (colonne 5). On calcule également le débit réel de l'UART, qui est inférieur au débit attendu.

3.3 Débit net du lien radio

Finalement, la dernière colonne montre le débit net du lien radio. L'ordre de grandeur est de 6 ko/s (48kbits/s), avec une grande taille de paquet et la vitesse de l'UART la plus élevée. On compare cela au débit du protocole 802.15.4 : 250 kbits/s.

4 Sources de programme Occam exécuté sur Arduino

```
#INCLUDE "wiring.module"

#PRAGMA EXTERNAL "PROC C.tvmspecial.3.xbee.send.packet([]BYTE msg, VAL INT n) =
INLINE PROC xbee.send.packet([]BYTE msg, VAL INT n)
  C.tvmspecial.3.xbee.send.packet(msg, n)
:

PROC main ()
  [96]BYTE message:
  TIMER time:
  INT t1,t2,n:
  PROC PrintTime([] BYTE message, INT t1, INT t2, INT nb)
    -- dirty code to print time stamps
    BYTE ch:
    INT count:
    SEQ
      count :=t1
      IF
        count < 0
          count := count + #07FFE
        TRUE
          SKIP
      SEQ i=0 FOR 6
        SEQ
          ch := (BYTE (count REM 10))
          count := count / 10
          message[5-i] := '0' + ch
      count := t2
      IF
        count < 0
          count := count + #07FFE
        TRUE
          SKIP
      SEQ i=0 FOR 6
        SEQ
          ch := (BYTE (count REM 10))
          count := count / 10
          message[(5-i)+8] := '0' + ch
      count := nb
      IF
        count < 0
```

```
            count := count + #07FFE
          TRUE
            SKIP
        SEQ i=0 FOR 6
          SEQ
            ch := (BYTE (count REM 10))
            count := count / 10
            message[(5-i)+15] := '0' + ch
  :

  SEQ
    t2,t1,n := 0,0,0
    beginSerial (9600)
    message := ".....*t .....*t.....
*n"
    WHILE TRUE
      SEQ
        time ? t1
        xbee.send.packet (message, 96)
        time ? t2
        PrintTime(message,t1,t2,n)
        n:= n + 1
:
```

5 Mesures sur MSP430 et CC2420

5.1 Matériel utilisé

On réalise les mesures de consommation d'un micro contrôleur MSP430 et un transceiveur CC2420 installé sur une mezzanine. La carte provient de l'ENSSAT de Lannion, elle reproduit une note d'application de Texas Instrument. Un résistance de 1Ω est branchée en série dans le montage. L'oscilloscope, relié aux bornes de la résistance, affiche la tension : on déduit l'intensité du courant consommé.

5.2 Consommation pour un paquet

Le programme réalise une émission régulière d'une trame radio, sous le contrôle d'un timer.

Le MCU est relié au transceiveur par un bus SPI (Serial periphéral interface, 400 kbits/s). L'émission d'un paquet est déclenchée par une interruption, puis le MCU revient à l'état de veille. Le mode basse consommation choisi est celui où seul le processeur est arrêté (Low Power Mode 0, LPM0, cf MSP430 Family User's Guide). Il existe quatre autres modes basse consommation qui désactivent des horloges secondaires.

La table montre les consommations observée sur la résistance série, et les durées de quelques opérations : veille, transfert de données sur SPI, écoute du canal avant la transmission (phase clear channel assessment CCA), transfert d'un paquet radio, écoute et réception.

	Veille (PM0)	Activité MCU	Écoute (CCA)	Transfert données	Écoute	Réception
Intensité (mA)	2 à 12	16	68	44 à 48	48 à 52	52
Durée (ms)	-	2	< 1	10	-	10

5.3 Mesure des débits de données

On réalise en continu l'émission de trames de 32 octets, contenant un numéro d'identification. Chaque commande d'émission démarre dès que le tranceiveur CC2420 est prêt à émettre, il n'y a pas de pause dans le programme (front montant, en E).

On contrôle la réception avec le module XBee, branché en USB sur la station Linux.

On distingue sur la courbe chaque phase pour émettre un paquet.

A Ecriture de la trame sur le bus SPI

B Prise de contrôle du canal radio

C Silence avant l'émission (vérifier le protocole)

D Emission de la trame

E Ecriture de la trame suivante

T Période de 20 ms pour émettre 32 octets : débit de 1600 octets/s

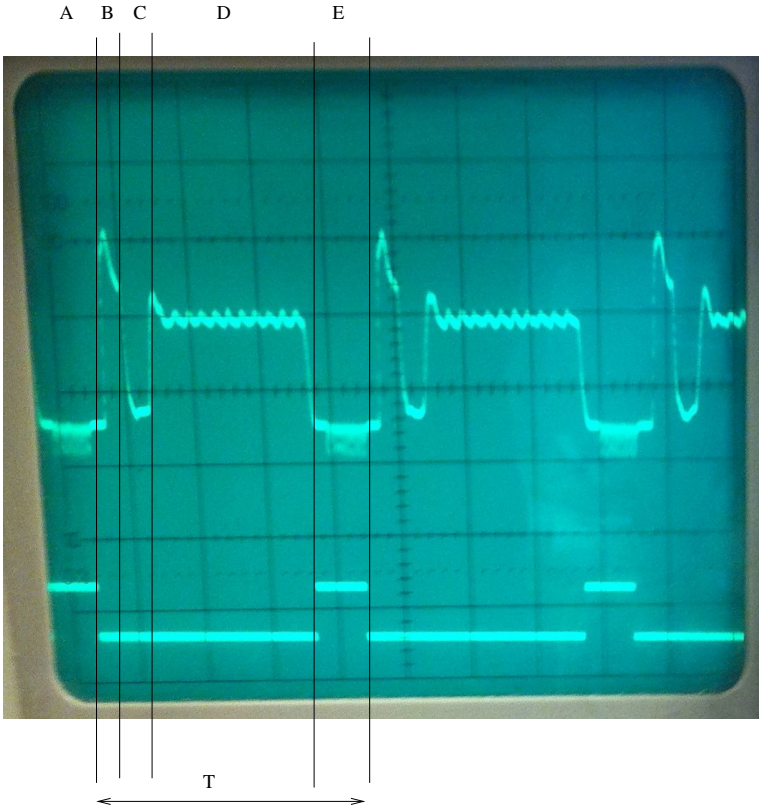

FIGURE 2 – Chronogramme d'expédition intensive sur MSP430.

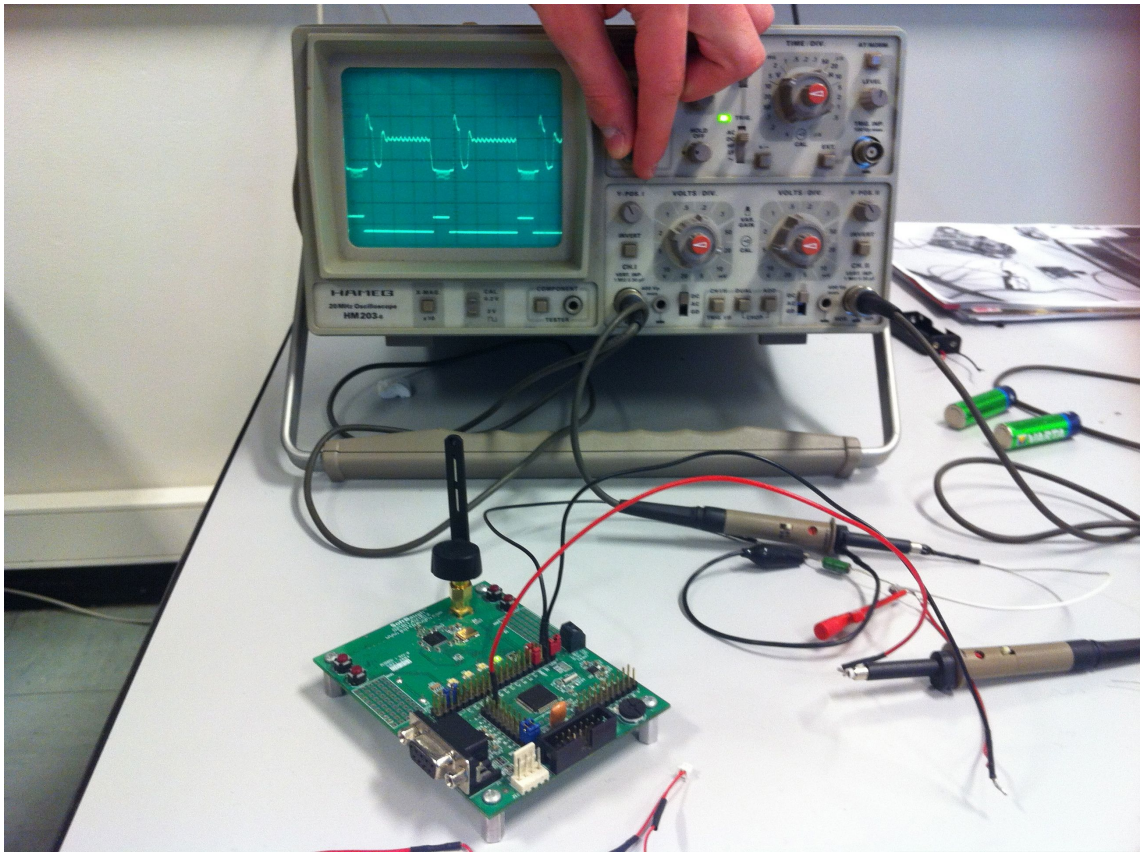

FIGURE 3 – Un systeme MSP430 - CC2420 en observation. La résistance de mesure (vert) se situe sous la sonde du haut.