Mihai Galos,
David Navarro, Fabien Mieyeville, Ian O'Connor

Reconfiguration Dynamique dans les Réseaux de capteurs sans fil

INL – UMR 5270

Organisation de la présentation

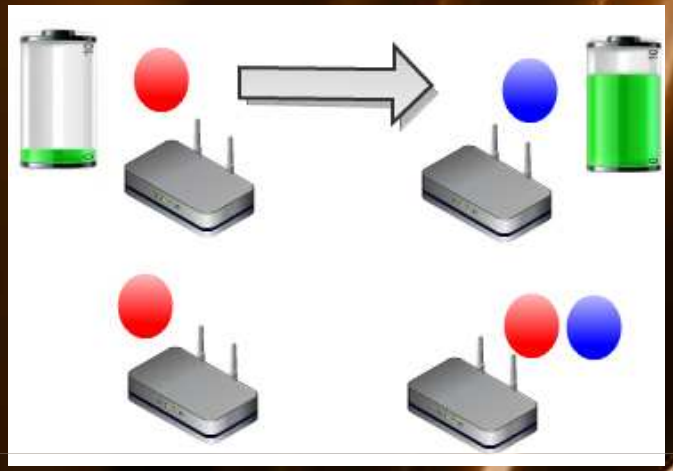
1. WSN: aperçu general
2. Reconfiguration dynamique
3. Machines Virtuelles pour WSN
4. Systèmes d'exploitation pour WSN
5. MinTax
6. Compilateur MinTax pour WSN
7. Résultats
8. Fin

WSN: apperçu general

- Qu'est-ce que c'est WSN?
- Organisation des noeuds(routage)
  - auto-organisation
  - noeuds fixes
- Noeuds: parties composantes
  - MCU
  - émetteur-récepteur radio
  - Batterie
  - (capteurs : lumière, température, etc.)

Reconfiguration Dynamique

- Qu'est-ce que c'est?



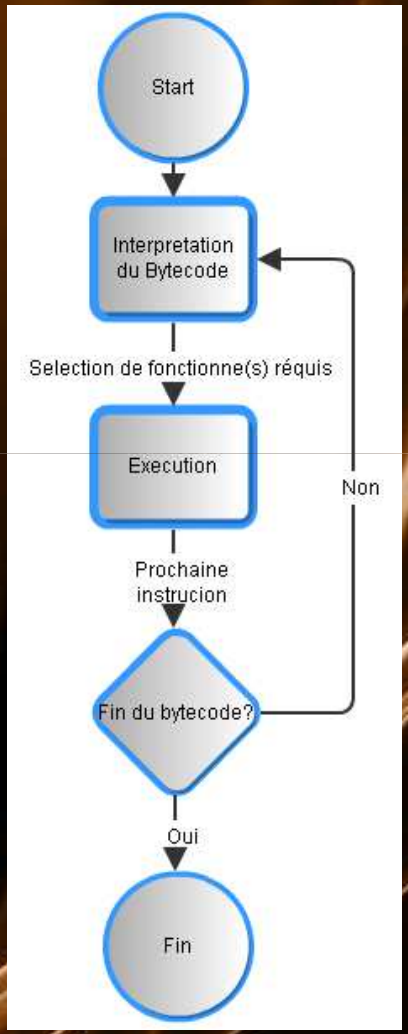
- Pourquoi a-t-on besoin de cela?
  - batterie faible
  - mis à jour du firmware
  - changement du role des nœuds
  - ajout des nouvelles noeuds

Reconfiguration Dynamique

- Solutions existantes:
  - Envoi de code objet brut(coûteux)
  - Machines Virtuelles(envoi du bytecode)
  - Systèmes d'exploitation(envoi de code objet)
- Solution proposée: compilation in-situ

Machines Virtuelles – aperçu général

- Inconvénients:
  - longue durée d'interprétation
  - coût d'énergie
  - bas niveau d'abstraction(mais plus grand que code machine)
- Avantages:
  - taille plus petite que code machine
  - support pour hétérogénéité



Machines virtuelles – exemples

- MatéVM (TinyOS)
  - lié a TinyOS
- DarjeelingVM
  - taille grande
- VMSTAR
  - support pour compilation «just in time»(JIT)
  - architecture fermée

Systèmes d'exploitation

- Il existe des dizaines de SO
  - TinyOS(pas RTOS, pas préemptif)
  - MantisOS
  - NanoRK(addition dynamique de nouvelles modules)
- Problème:
  - les nouvelles fonctionnalitées sont toujours en code objet
  - pas de / faible support pour hétérogénéité materielle / hétérogénéité des SO

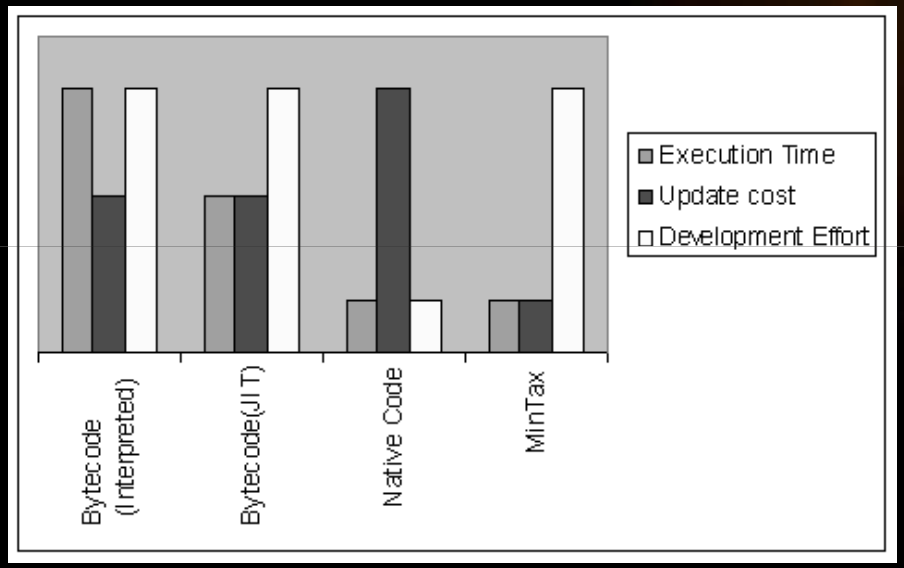
Compilation In-Situ

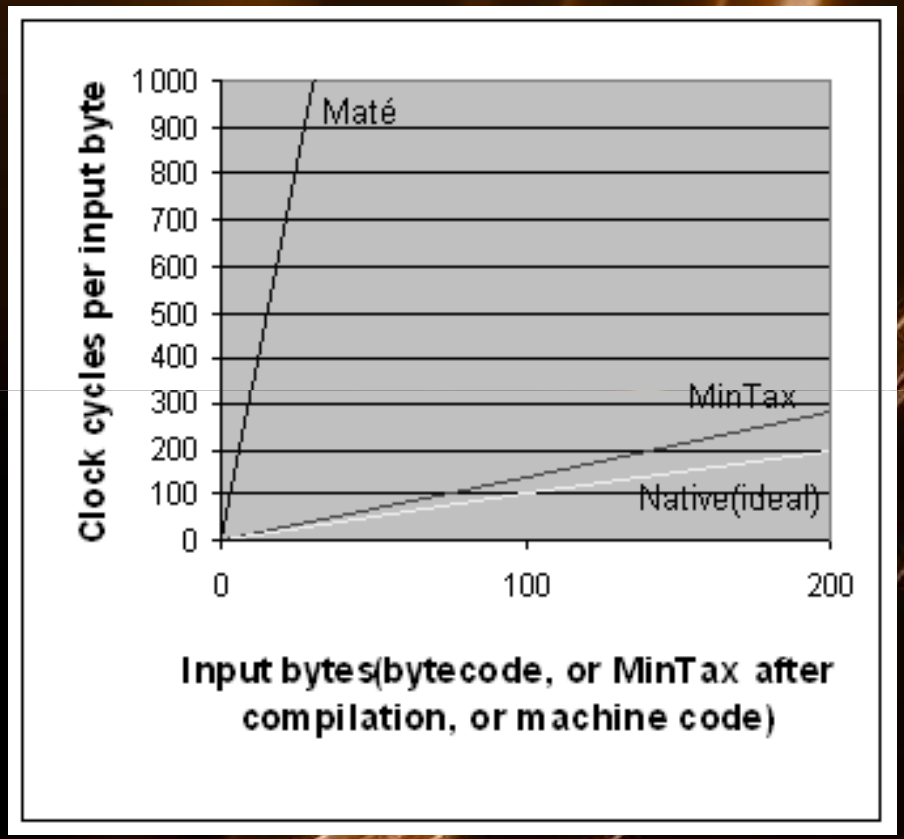
- Nécessité d'un langage haut-niveau: MinTax
- Le Compilateur reste sur le noeud; pas sur un PC
- Dès qu'il finit, une écriture de flash sera effectué avec la nouvelle fonctionnalité

Tableau de comparaison

	Coût d'énergie?	Usage Flash?	Adapté aux besoins?
Code Machine + SO	- - -	+ + +	- -
Machine Virtuelle	+	+	+
Compilation In-Situ	+ + +	+ (+)	+ +

Comparaison(Continuation)





MinTax

- Qu'est-ce que c'est MinTax?
- Langage «strongly typed»
- Noms des fonctions, variables, constantes sont au pire 2 characteres
- Chaque clause se delimite des autres par un delimitateur (par défaut « ; »)
- Support pour boucles «while, for» et conditions«if, switch-case»
  - Chaque clause précisé au-dessus se termine par un fin de clause
  - Des instructions avant celui n'ont pas besoin du delimitateur
  - Clauses imbriquées
- Support pour structures, support partiel pour pointeurs

MinTax

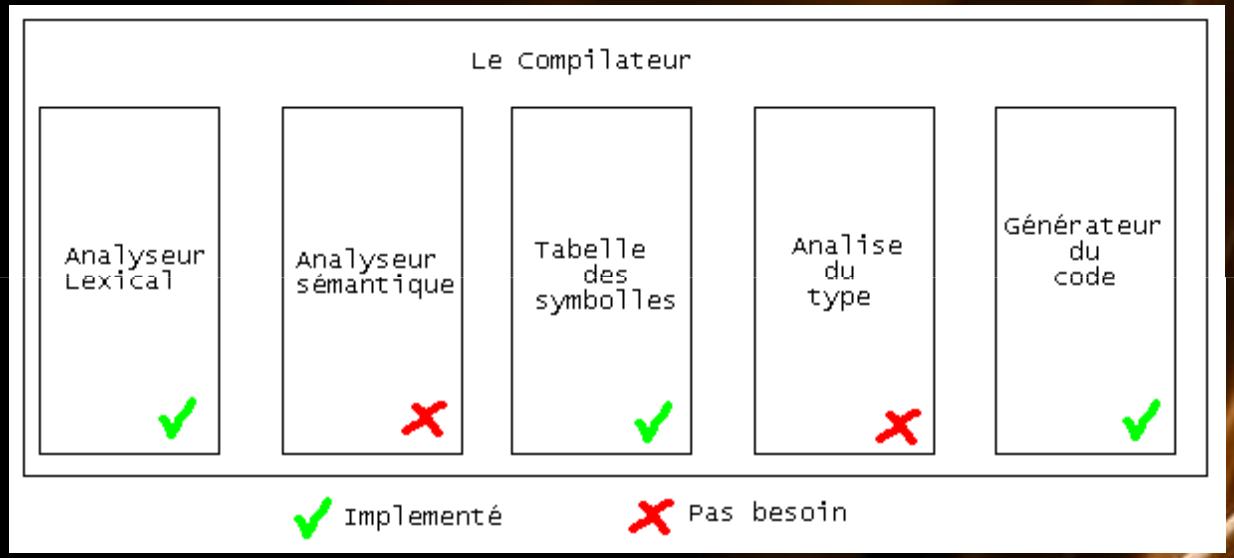
- Support pour différents types des données: s/uint8 , s/uint16
- Opérations sans signe égal

- 1 seul caractére pour appeler une fonction
  - Une fonction peut avoir des paramètres formables de maximum 4 octets
- Le fin d'une fonction peut retourner une valeur
  - Maximum 4 octets pour le retour

MinTax : un exemple

```
1.  aUk{    // fonction a, variable k de type unsigned int16
2.  ^i=A2;// new uint8 i; i=adc_read(PINA2)
3.  ^j=8; // new uint8 j; j=8
4.  ^u=i; // new uint8u = i;
5.   Wi<99 // while i<99
6.        Wj>0 // while j>0
7.            Pi,j; //PWM duty cycle=i, period=j
8.            d@k; // appel fontionne de retard
9.            j-   // decrement j
10.       # // fin du while j>0
11.  i+    // increment i
12.#      // fin du while i<99
13.}u;     // fin de la fonction a, valeur de retour: u
```

Compilateur pour MinTax



Analyseur Lexical

- Généré (éxpressions réguliers, re2c)
- Fusioné avec la logique pour construire la tabelle des symboles
- Extrêmement optimisé(taille reduite grace aux sauts internes)
- Responsable de classifier le fichier d'entrée en:
  - Identificateurs (noms de symboles)
  - Mots clés
  - Délimitateurs
  - Signes de punctuation
  - Operateurs
- Son sortie: Table lexicale + Table des symboles

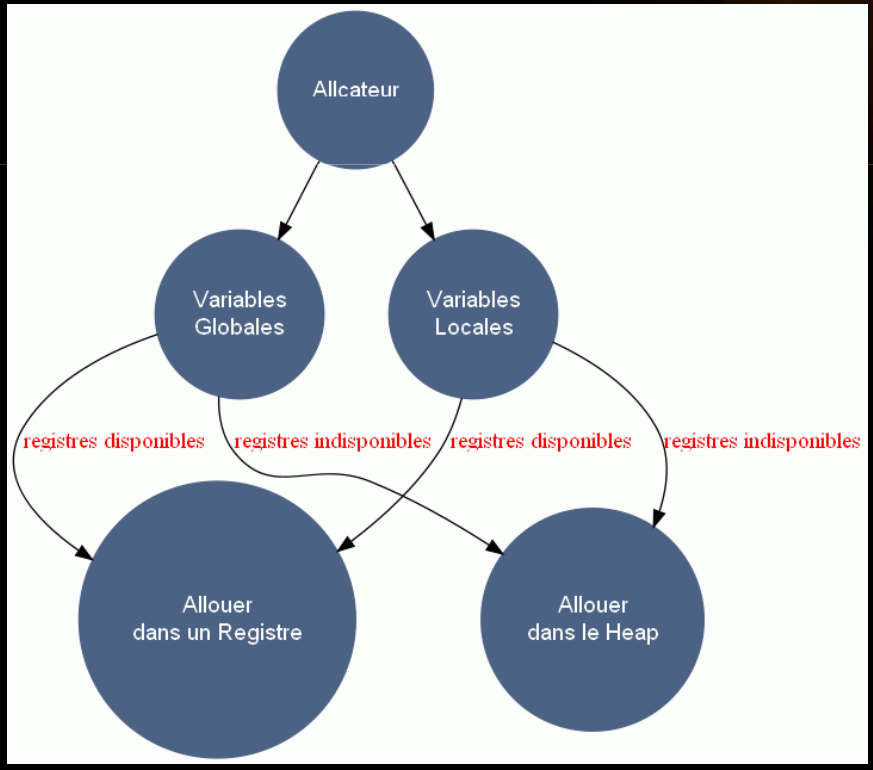
Tabelle des Symboles

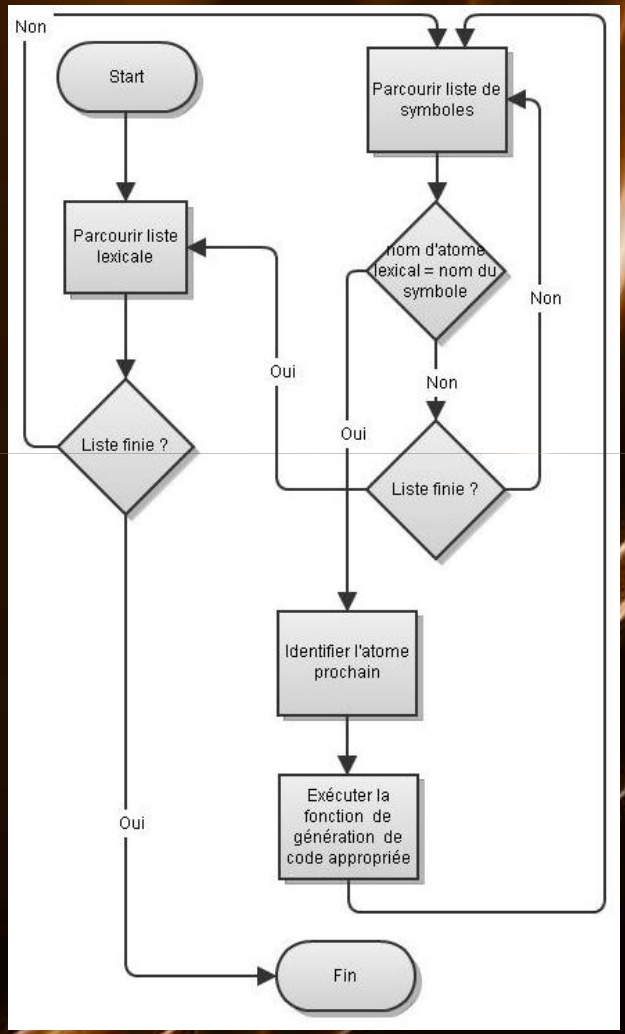
- Qu'est-ce c'est et à quoi cela sert?
  - Savoir le type du symbole et la ligne où il est référencé
  - Savoir la portée du symbole
- Nombre limité d'éléments
  - Maximum 50 symbols, max 10 références / symbole

```
Name: dd    Type: UCHAR    Parent/Init: !    Add. Info: None                    Scope: 0, Line/End: 1
Name: dd    Type: None     Parent/Init: !    Add. Info: Function Implementation Scope: dd, Line/End: 2 , 3
Name: ee    Type: None     Parent/Init: !    Add. Info: Function Implementation Scope: ee, Line/End: 5
Name: x     Type: UCHAR    Parent/Init: !    Add. Info: Formal Parameter        Scope: ee, Line/End: 5 , 6, 7
Name: y     Type: UCHAR    Parent/Init: !    Add. Info: Formal Parameter        Scope: ee, Line/End: 5
Name: ff    Type: None     Parent/Init: !    Add. Info: Function Implementation Scope: ff, Line/End: 8
```

Le générateur du code

- Alloue les variables dans des registres ou Heap
- Donne le code machine final
- Ecrit le flash





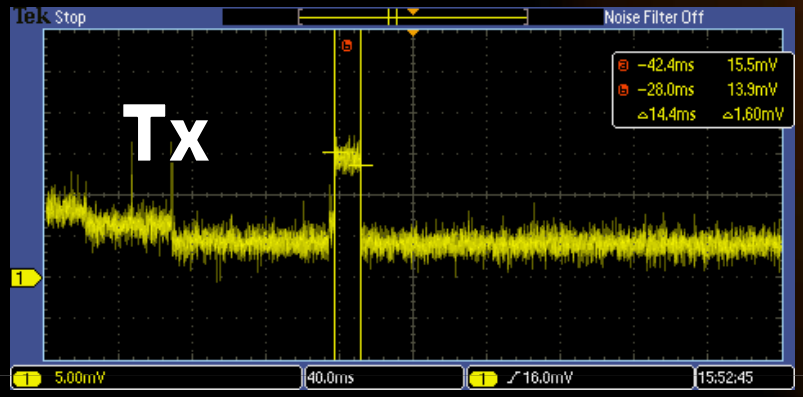
Le compilateur MinTax – résumé

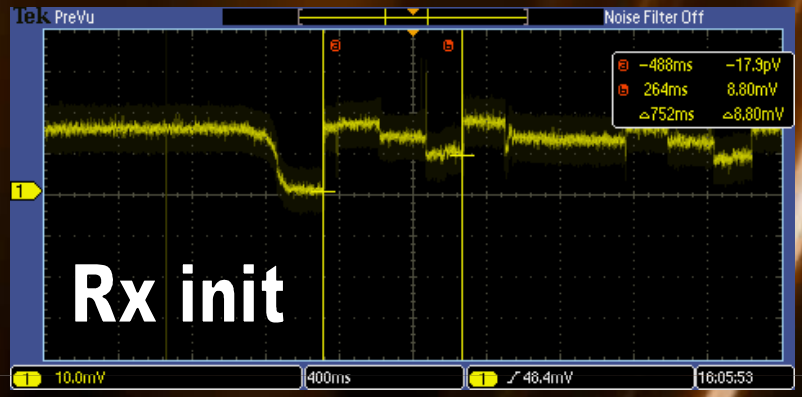
- Taille totale: 25kOctets (comparable avec une MV)
  - En utilisant optimisations de taille (-Os)
- Version AVR/PC/MSP430, version Debug / Release
- L'abstraction des symboles : changement facile de syntaxe
- Architecture materielle:
  - Atmel AVR (Crossbow Mica2 – AtMega128)
  - Texas Instruments MSP430(portage en cours) – MSP430f5438(pas d'architecture de noeud WSN pour l'instant..)

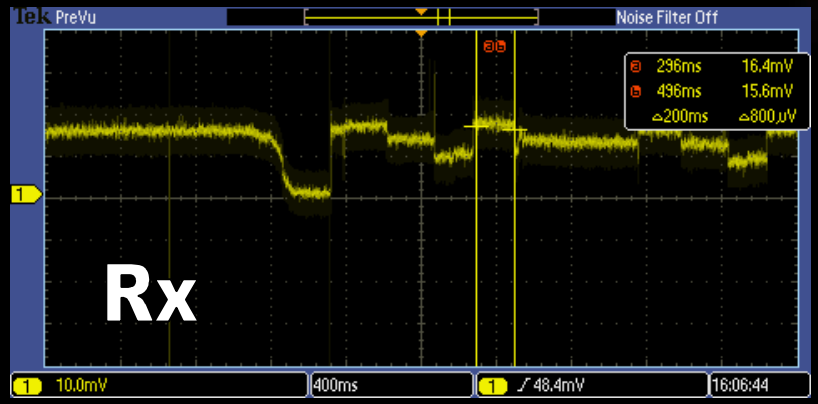
Compilateur in situ: avantages sur MV

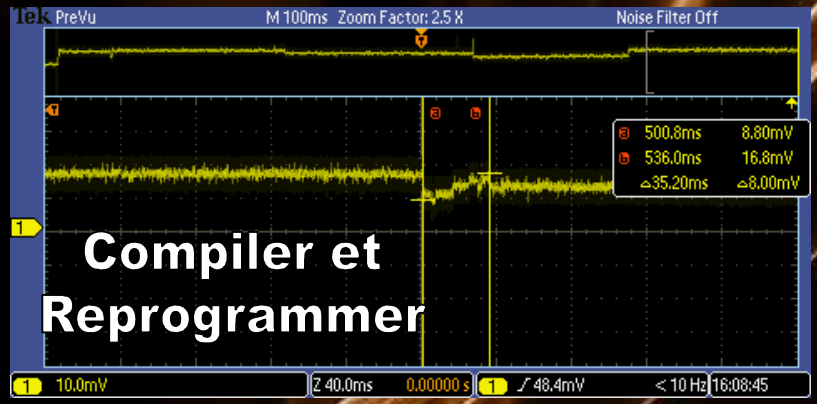
- Plus adapté aux systèmes en temps réel
- Taille réduite de données
- Plus facile de compresser
  - Bytecode: opérandes et opération dans le même octet(sequence cvasi-aléatoire) => difficile à compresser
- Consommation plus faible d'énergie
  - Le code n'est plus interprété

Résultats


Tek Stop
Noise Filter Off
−42.4ms 15.5mV
−28.0ms 13.9mV
Δ14.4ms Δ1.60mV
Tx
1
5.00mV
40.0ms
16.0mV
15:52:45


Tek PreVu
Noise Filter Off
−488ms −17.3pV
264ms 8.80mV
Δ752ms Δ8.80mV
Rx init
1
10.0mV
400ms
46.4mV
16:05:53


Tek PreVu
Noise Filter Off
236ms 16.4mV
496ms 15.6mV
Δ200ms Δ800µV
Rx
1
10.0mV
400ms
48.4mV
16:06:44


Tek PreVu
M 100ms Zoom Factor: 2.5 X
Noise Filter Off
500.8ms 8.80mV
536.0ms 16.8mV
Δ35.20ms Δ8.00mV
Compiler et
Reprogrammer
1
10.0mV
Z 40.0ms 0.00000 s
48.4mV
< 10 Hz
16:08:45