

# 基盤構築特論

## Lesson1. Getting Started with TinyOS

岩手県立大学大学院 ソフトウェア情報学研究科

分散システム学講座 及川一樹

## ちゅ〜い

---

- ▶ この文章は2008年10月15日現在の  
TinyOS Tutorials Lesson 1 Getting Started with TinyOSを  
元にした資料です  
[http://docs.tinyos.net/index.php/Getting\\_Started\\_with\\_TinyOS](http://docs.tinyos.net/index.php/Getting_Started_with_TinyOS)
- ▶ 内容が正しいかどうかに関しては責任を持ちません。  
参考程度にどうぞ〜

# 1 Introduction

---

- ▶ Lesson1では以下の内容を行う
  - ▶ TinyOSプログラムをコンパイルし、moteにインストールする方法
  - ▶ コンポーネントモデルの基本概念と記法の紹介
  - ▶ TinyOSのソースコードドキュメントの生成と読み方

## 2 Compiling and Installing (1)

---

- ▶ “Blink” という単純なサンプルを利用
  - ▶ moteハードウェアを持っていない場合は、TOSSIMというシミュレータ向けにコンパイル
- ▶ TinyOSでは多機能で拡張的なmakeシステムを採用している
  - ▶ 簡単に新しいプラットフォームやコンパイルオプションを追加できる
  - ▶ `tinynos-2.x/support/make` に定義されている

## 2 Compiling and Installing (2)

---

- ▶ 正しく環境が構築されているか確認
  - ▶ `$ tos-check-env`
  - ▶ 警告やエラーが出た場合は内容を元に対処
    - ▶ tinyos-helpメーリングリストに質問を投稿
    - ▶ アーカイブから検索するのも有用
  - ▶ `tos-check-env`コマンドが無いと言われた場合
    - ▶ インストール説明書を参考にTinyOS toolsをインストール
    - ▶ RPMをインストールした場合 → `/usr/bin`や`/usr/local/bin`を探す
    - ▶ CVSからダウンロードした場合
      - `tinyos-2.x/tools/tinyos`に入った後
        - `$ configure`
        - `$ make`
        - `$ make install` (※Linux環境ではスーパーバイザで実行する必要あり)

## 2 Compiling and Installing (3)

---

- ▶ TinyOSのビルドシステムが有効になっているか確認
  - ▶ MAKERULES環境変数をチェック
    - ▶ `$ printenv MAKERULES`
  - ▶ 通常は以下のパスが出力される
    - ▶ `/opt/tinyos-2.x/support/make/Makerules`
- ▶ もしMAKERULES環境変数が設定されていない場合は手動で設定

## 2 Compiling and Installing (4)

- ▶ make [platform] でTinyOSアプリケーションをコンパイル
- ▶ Blinkサンプルをmica2向けにコンパイルする方法
  - ▶ apps/Blink ディレクトリに移動
  - ▶ `$ make mica2`
    - ▶ シミュレーションを行う場合は  
`$ make mica2 sim`

```
dark /root/src/tinyos-2.x/apps/Blink -4-> make telosb
mkdir -p build/telosb
    compiling BlinkAppC to a telosb binary
ncc -o build/telosb/main.exe -Os -O -mdisable-hwmul -Wall -Wshadow
-DDEF_TOS_AM_GROUP=0x7d -wnesc-all -target=telosb -fnesc-cfile=build/telosb/app.c
-board= BlinkAppC.nc -lm
    compiled BlinkAppC to build/telosb/main.exe
        2782 bytes in ROM
        61 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
writing TOS image
```

## 2.1 Making sure you're invoking the right version of the nesC compiler

- ▶ 以下のエラーが出る場合、古いバージョンのnescコンパイラを呼び出している

```
BlinkAppC.nc:46: syntax error before `new`  
make: *** [exe0] Error 1
```

- ▶ TinyOS 1.xからのアップグレードなどを行うと、容易に発生する
- ▶ `ncc --version` を実行し、バージョンを確認する
- ▶ バージョンが意図したもので無かった場合
  - ▶ `which`コマンドでパスを確認して、古いnccをリネーム
  - ▶ 同様の処理をnescに 대해서 も行う

## 2.2 Installing on a mica-family mote (1)

---

- ▶ プログラムをmoteにダウンロード
  - ▶ moteをプログラミングボードの上に置き接続する
  - ▶ 電源をプログラムボードに接続するかノードから直接供給
    - ▶ 電源が供給される → PWDと書かれている緑色のLEDが点灯
    - ▶ moteの電源を利用 → moteのスイッチをONにする
  - ▶ プログラミングボードのスイッチは通常はOFF
    - ▶ moteにプログラミングする際に問題が起こったらONにし、プログラミングが終わったらスイッチをOFFに戻す
  - ▶ パススルーDB9シリアルケーブルを利用し、9-pinのコネクタをコンピュータのシリアルポートに接続する
    - ▶ シリアルポートを持っていない場合は、容易に入手できるDB9シリアル-USB変換ケーブルを利用する

## 2.2 Installing on a mica-family mote (2)

- ▶ `$ make mica2 reinstall mib510,serialport`
  - ▶ `serialport` の所にはシリアルポートのデバイス名を入れる
  - ▶ Windowsの場合
    - ▶ COMn → `/dev/ttySn-1`
  - ▶ Linuxの場合
    - ▶ 通常のシリアルポート → `/dev/ttySn`
    - ▶ USBシリアルケーブル → `/dev/ttyUSBn` , `/dev/usb/lp/n`
    - ▶ 書き込み権限が必要なので、権限がない場合は付与する  
`$ chmod 666 serialport`

```
cp build/mica2/main.srec build/mica2/main.srec.out
installing mica2 binary using mib510
uisp -dprog=mib510 -dserial=/dev/ttyUSB1 --wr_fuse_h=0xd9 -dpart=ATmega128
--wr_fuse_e=ff --erase --upload if=build/mica2/main.srec.out
Firmware Version: 2.1
Atmel AVR ATmega128 is found.
Uploading: flash
Fuse High Byte set to 0xd9
Fuse Extended Byte set to 0xff
rm -f build/mica2/main.exe.out build/mica2/main.srec.out
```

## 3 Installation options

---

- ▶ プログラムをテスト
  - ▶ moteをプログラミングボードから取り外し、電源を入れる
  - ▶ 4Hzでカウンターがインクリメントする様子が3つのLEDで表示されれば成功
- ▶ 各種コマンドの説明
  - ▶ reinstall: 現在のコンパイル済みバイナリをインストールする
  - ▶ clean: コンパイル済みバイナリファイルをすべて削除する
  - ▶ install: プログラムを再コンパイルし、インストールする
- ▶ ネットワークではmoteに一意的な識別子が必要となる
  - ▶ コンパイル時には1つの識別子が割り当てられている
  - ▶ 別の識別子を利用したい場合は、インストール時に指定できる
    - `$ make telosb install.5`
    - ▶ インストールされたノードには識別子5が与えられる

## 4 Components and Interfaces (1)

---

- ▶ Blinkがどのように動いているのか
  - ▶ TinyOSと同じように nesC で記述されている
- ▶ nesC
  - ▶ C言語にコンポーネントと並行性の為の機能を追加した言語
  - ▶ nesCアプリケーションは1つ以上のコンポーネントの集合やそれらの接続から成り立つ
    - ▶ コンポーネントは2つのスコープを持つ
      - インターフェイスの名前などの仕様
      - 上記の実装
    - ▶ コンポーネントはインターフェイスを提供し、利用する
      - 提供されるインターフェイスはコンポーネントが提供する機能を表す
      - 利用するインターフェイスはコンポーネントの実装においてジョブを実行するために必要な機能を表している

## 4 Components and Interfaces (2)

---

- ▶ インターフェイスは双方向性を持つ
  - ▶ インターフェイスはコマンド群とイベント群の仕様を定める
    - ▶ コマンド: インターフェイス提供者が実装すべき機能
    - ▶ イベント: インターフェイス利用者が実装すべき機能
  - ▶ コンポーネントがコマンドを呼び出す為には、イベントを実装しなければならない
  - ▶ 1つのコンポーネントは以下の利用や提供が可能
    - ▶ 複数のインターフェイス
    - ▶ 同一のインターフェイスの複数のインスタンス
- ▶ コンポーネントが提供・利用するインターフェイスの集合は、コンポーネントの署名と見なされている

## 4.1 Configurations and Modules

---

- ▶ コンポーネントには以下の2種類がある
  - ▶ モジュール
    - ▶ 1つ以上のインターフェイスの実装
  - ▶ コンフィギュレーション
    - ▶ 同時に使う他のコンポーネントをまとめる
    - ▶ インターフェイス同士を接続する
- ▶ すべてのnesCアプリケーションはコンポーネントを接続するトップレベルのコンフィギュレーションで記述されている

## 5 Blink: An Example Application (1)

---

- ▶ 3つのLEDでカウンターを表示している
  - ▶ 実際はLED0~LED2はそれぞれ0.25Hz, 0.5Hz, 1Hzでオン・オフを繰り返している
  - ▶ 2秒でインクリメントする0~7までの2進カウンタと同じ効果
- ▶ Blinkは2つのコンポーネントから成る
  - ▶ "BlinkC.nc"というモジュール
  - ▶ "BlinkAppC.nc"というコンフィギュレーション
- ▶ トップレベルコンフィギュレーションファイルが必要とする
  - ▶ この場合は BlinkAppC.nc (通常、自身の名前にちなんで名付けられる)
  - ▶ BlinkC.ncと他のコンポーネントを結びつけるのに利用される
- ▶ BlinkC.nc は実装を提供

## 5 Blink: An Example Application (2)

- ▶ モジュールとコンフィギュレーションに分割する理由
  - ▶ システムデザイナーがアプリケーションを構築する場合、既存の実装を考慮する必要がない
  - ▶ 例)
    - ▶ デザイナーは設計を担当しなかったモジュールを接続することで、コンフィギュレーションを提供できる
    - ▶ 開発者は様々なアプリケーションで利用できるライブラリモジュールを提供できる
- ▶ ファイル名の命名規則
  - ▶ 強制ではないが、わかりやすくするため右の表のようなファイル名の命名規則を採用することを提案 (TEP 3)

ファイル名	ファイルタイプ
Foo.nc	インターフェイス
Foo.h	ヘッダファイル
FooC.nc	パブリックモジュール
FooP.nc	プライベートモジュール

## 6 The BlinkAppC.nc Configuration (1)

---

- ▶ nesCコンパイラはトップレベルコンフィギュレーションを含むファイルが与えられたときコンパイルを行う
  - ▶ 通常、TinyOSアプリケーションは標準のMakefileが付属
  - ▶ プラットフォームや適したオプションの選択などを行う
- ▶ `configuration BlinkAppC {`  
`}`
  - ▶ コンフィギュレーションファイルであることを示す
  - ▶ 波括弧の中にはモジュールのように、`uses`や`provides`を定義することができる
  - ▶ 重要: コンフィギュレーションはインターフェイスを利用したり提供することができる (すべてのコンフィギュレーションがトップレベルというわけではない)

## 6 The BlinkAppC.nc Configuration (2)

---

- ▶ implementationキーワードに続く波括弧の内側が実際のコンフィギュレーションの実装
  - ▶ components で始まる行では、このコンフィギュレーションが参照するコンポーネントが定義されている
    - ▶ 3つのコンポーネント: Main, BlinkC, LedsC
    - ▶ TimerMilliCのインスタンス: Timer0, Timer1, Timer2
      - as キーワードを使うことで別名(エイリアス)として定義
  - ▶ その他の行では、コンポーネントが利用するインターフェイスと他のコンポーネントが提供するインターフェイスの接続の記述から成る
    - ▶ MainC.BootやMainC.SoftwareInitインターフェイスはTinyOSのブートシーケンスの一部 (詳細はLesson 3)
      - これらへの接続はLEDやタイマを初期化するのを可能にする

# 7 The BlinkC.nc Module (1)

---

- ▶ `module BlinkC {...}`
  - ▶ BlinkCという名前のモジュール
  - ▶ 波括弧の中にはモジュールが利用・提供するインターフェイスを宣言
    - ▶ BlinkCで利用するインターフェイス
      - Timer0~Timer2: Timer<TMilli>インターフェイスのインスタンス
      - Leds
      - Boot
    - ▶ 宣言したインターフェイスのコマンドが利用可能になる
    - ▶ 宣言したインターフェイスのイベントはすべて実装する必要あり

## 7 The BlinkC.nc Module (2)

---

- ▶ BlinkAppC.ncの最後の4行の意味が明確に
  - ▶ BlinkCで利用される3つのTimer<TMilli>インターフェイスをTimerMilliCコンポーネントが提供するTimer<TMilli>インターフェイスに接続
  - ▶ 同様にBlinkCのLedsインターフェイスをLedsCコンポーネントが提供するLedsインターフェイスに接続
- ▶ nesCはインターフェイスをバインドする際に矢印を利用
  - ▶ A->B: AをBに接続
    - ▶ A: インターフェイス利用側
    - ▶ B: 提供側
  - ▶ A.a -> B.b: コンポーネントAのインターフェイスaを、コンポーネントBのインターフェイスbに接続

## 7 The BlinkC.nc Module (3)

---

- ▶ コンポーネントが同じインターフェイスの複数のインスタンスを使用または提供するとき、インターフェイスの名前付けは重要
- ▶ コンポーネントがあるインターフェイスのインスタンスを1つしか持っていない場合は、インターフェイス名を省略できる
- ▶ 例 (BlinkAppC.nc):
  - ▶ `BlinkC.Leds -> LedsC; //BlinkC.Leds -> LedsC.Ledsと同じ`
  - ▶ `BlinkC -> LedsC.Leds; //BlinkC.Leds -> LedsC.Ledsと同じ`
  - ▶ `BlinkC -> Timer0.Timer; //Compile error!`
  - ▶ `Timer0 <- BlinkC.Timer0; // BlinkC.Timer0 -> Timer0 と同じ`  
(しかし、可読性の面からほとんどの場合右矢印が利用される)

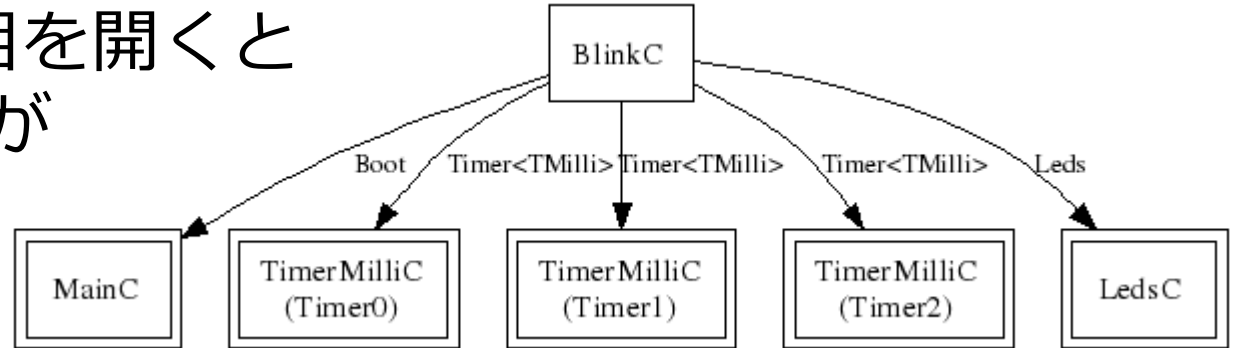
## 8 Visualizing a Component Graph (1)

---

- ▶ 丁寧に設計されたTinyOSシステムには、多くのコンフィギュレーションレイヤで構成されていることがしばしば
  - ▶ テキストエディタでモジュールの下層にたどり着くのは困難
- ▶ TinyOSおよびnesCにはnesdocと呼ばれるドキュメンテーション機能を持っている
  - ▶ nesdocはソースコードからドキュメントを自動生成するツール
  - ▶ コメントに加えて構造やコンフィギュレーションの構成なども表示
- ▶ `$ make mica2 docs` でドキュメンテーションを生成
  - ▶ `./doc/nesdoc/mica2/index.html` を開く

## 8 Visualizing a Component Graph (2)

- ▶ BlinkAppCの項目を開くと右図のような図が表示される



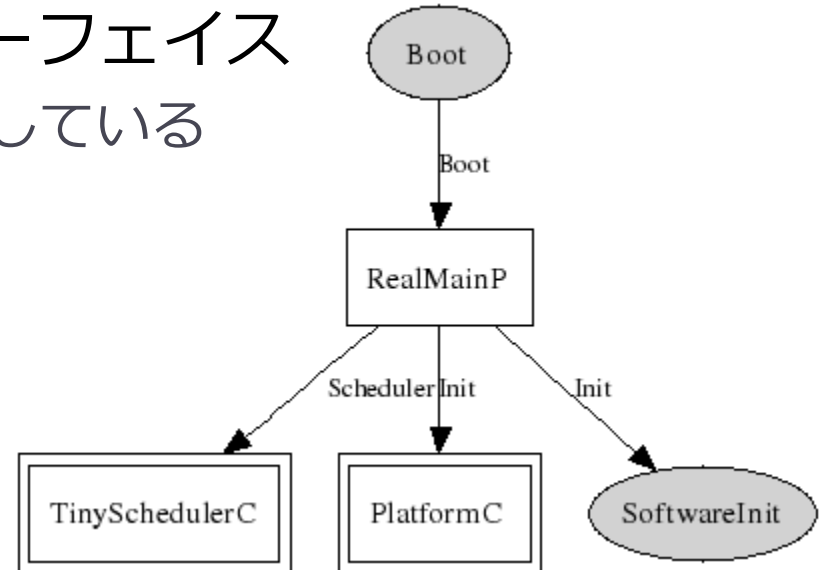
- ▶ ダイアグラムは右下の表を参考

- ▶ 線は接続を表す
- ▶ 陰付きの楕円形は提供・利用するインターフェイスを表す
- ▶ コンポーネントをクリックすることによって内部を調べることができる

	シングルトン	ジェネリック
モジュール		
コンフィギュレーション		

## 8 Visualizing a Component Graph (3)

- ▶ 陰付きの楕円は接続可能インターフェイス
  - ▶ MainCはBootとSoftwareInitを提供している
- ▶ ブートシーケンスの詳細は Lesson 3 (TEP 107)



## 9 Conclusion

---

- ▶ TinyOSのコンポーネントモデルの概念の紹介
  - ▶ コンフィギュレーション
  - ▶ モジュール
  - ▶ インターフェイス
  - ▶ 接続
- ▶ コンポーネント同士の接続によってアプリケーションが構築される方法を示した