第4章

センサ・データの取得からノード間の送受信までをプログラミング

センサ・ネット専用OSと拡張C言語による開発事例

ワイヤレス・センサ・ネットワーク編

宮本 哲

ここでは，センサ・ネットワークのソフトウェアの開発事例を紹介する．センサのデータを取得し，ノード間の送受信を行うためのプログラミングを取り上げる．センサ・ネットワークに特化して開発されたOSと拡張C言語を使用する．なお，ここで紹介したソース・コードは，本誌付属のCD-ROMに収録されている．（編集部）

ワイヤレス・センサ・ネットワークは，センサ・ノードを自由に配置できるという特徴を持ちます．あらゆる場所に簡単に配置できるようにセンサ・ノードは小型でなければなりません．また，ほとんどの場合，センサ・ノードは電池で駆動することが求められます．これらの要件から，センサ・ノードに使われるマイクロプロセッサや無線通信モジュール，センサは小型で低消費電力のものでなければなりません．

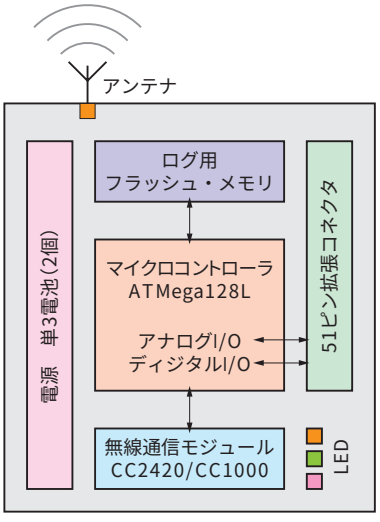
センサ・ノードが低消費電力で小型であるということは，マイクロプロセッサの性能は低く，メモリなどの資源も非常に限られるということです．低消費電力処理に加えて，ソフトウェア開発ではこういった制限も考慮しなければなりません．具体的には，以下のようなことがソフトウェアに求められます．

- プログラム・サイズが小さい
- メモリを消費しない
- むだな処理を避ける
- マイクロプロセッサをなるべくスリープさせる．無線通信モジュールやセンサは使わないときにOFFにする

現時点では，ワイヤレス・センサ・ネットワークの応用分野として考えられているものは多岐にわたり，一つのマスマーケットが見つかっているわけではありません．したがって，さまざまな用途に対応できるようにセンサ・ノードにつけるセンサは取り替え可能であり，センサ・ノードのプログラムは変更が容易であることが望ましいといえます．

本稿では，これらの要件を満たすワイヤレス・センサ・ネットワークの開発事例を紹介します．具体的には，米国Crossbow Technology社のセンサ・ネットワーク向け無線モジュール「MOTE」とセンサ・ネットワーク用OS「TinyOS」を例に挙げて，おもにソフトウェア開発について解説します．

（a）外観（MICAz）



（b）構成図

図1
本稿で使用する無線モジュール
（a）は，IEEE 802.15.4準拠の無線モジュール「MOTE MICAz」の外観である．（b）に構成を示す．51ピン拡張コネクタを介してセンサ・ボードを接続する．また，ログ用のフラッシュ・メモリを備えている．電源については単3電池2個で駆動する．

1 センサ・ネットワークの構成要素

最初に，本稿で取り上げるセンサ・ネットワーク・システムを構築するためのハードウェア，ソフトウェアについて説明します．

● ハードウェア――無線モジュールとセンサ・ボード

図1に，無線モジュール「MOTE」の概要を示します(1), (2)．本無線モジュール単体ではセンサを備えていません．拡張用の51ピン・コネクタを介して，さまざまなセンサ・ボードを接続して使います．

プログラムを書き込んだり，センサのデータをパソコンへ送信するには，インターフェース・ボードを用います．このインターフェース・ボードはシリアル・インターフェースを備えており，これをパソコンにつないで使います．このほかEthernetを備えたボードもあります．

主要なハードウェアとしては，上記のもの以外にゲートウェイ（Stargate）があります．これは，CPUに米国Intel社のXScaleを使い，その上でLinuxを動作させています．そのほか，シリアル・ポート，カード・インターフェース（PCMCIAやCompactFlash），Ethernetポート，USBポート，拡張用51ピン・コネクタを備えています．ただし，このゲートウェイは電池駆動ではありません．

● ソフトウェア――センサ・ネットのための専用OS

無線モジュールに搭載されているソフトウェアは「TinyOS」と呼ばれるOSとそのアプリケーション・ソフトウェアです．

TinyOSはワイヤレス・センサ・ネットワーク用に開発されたOSです(3)．本OSはオープン・ソースであり，多くの開発者によって機能の追加や改良が行われています．情報交換やサポートを行うためのメーリング・リスト（http://www.tinyos.net/special/support/）や，今後の機能拡張について議論するワーキング・グループもあります．

本OSは，先に述べたワイヤレス・センサ・ネットワークのソフトウェアに求められる条件，すなわち「軽量であること」を満たすために開発されました注1．したがって，一般的なOSとは異なり，プリエンプティブ・マルチタスク（複数のタスクを強制的に切り替えるしくみ）やメモリ保護機能といったサービスは提供しません（図2）．本OSは，OSというよりもハードウェア制御やネットワーク・プロトコル・スタックなどのソフトウェア・ライブラリ注2から構成されるフレームワーク注3といったほうが正確です．このフレームワークにより，限られた資源を使って高い性能を実現することと，ソフトウェアの開発効率を上げることのバランスをうまくとっています．

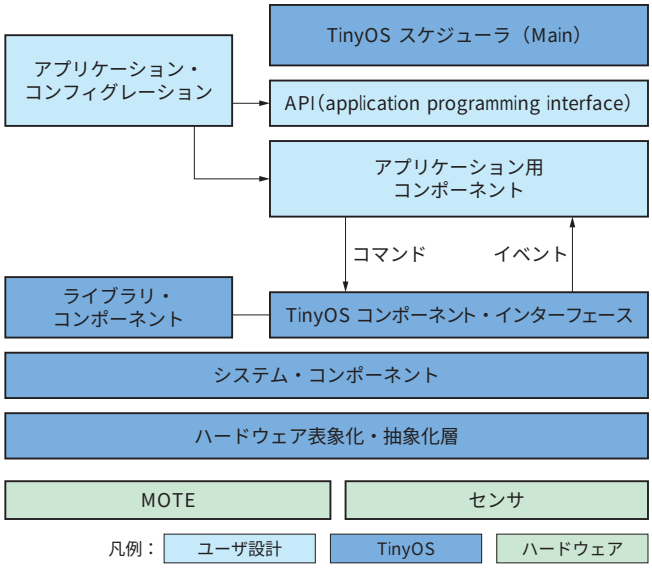


図2　TinyOSモジュール構成

TinyOSはハードウェア（MOTEやセンサ・ボード）から上へ階層構造をとっているため，モジュールの差し替えや追加が容易である．アプリケーション開発者（ユーザ）は，上位層のTinyOSコンポーネント・インターフェースを使って効率良くアプリケーション・ソフトウェアを開発できる．

● 拡張C言語によるプログラミング

TinyOSの根幹をなしているのがnesCというC言語を拡張したTinyOS用のプログラミング言語注4です．nesCにはTinyOS向けのプログラミングに対応するため，いくつかのC言語拡張が含まれています．おもな拡張としては，コンポーネント・プログラミングのための拡張（ワイヤリング）とタスク処理のための拡張の二つが挙げられます．

1）コンポーネント・プログラミング

nesCではコンポーネント（component）と呼ぶソフトウ

注1：OSなしの場合と比べてOSを実装している無線モジュール（MOTEなど）のほうが処理が重くなると指摘されることがあるが，一概にそうとは言えない．
注2：本稿では説明しないが，OAP（Over the Air Programming）というワイヤレス・ネットワークを介してセンサ・ノードのプログラムを書き換えるモジュールもある．
注3：ソフトウェア・システムを設計，実装するための枠組み．
注4：nesCのリファレンス・マニュアルは「http://www.tinyos.net/tinyos-1.x/doc/nesc/ref.pdf」から入手できる．本稿執筆時点（2005年7月）では日本語版を校正中であり，これが終わりしだいクロスボーのホームページ（http://www.smartdust.jp/）にて公開する予定．