1. nxtOSEK特有の表現

* タスクの組み方

nxtOSEKの場合、通常のC言語での書き方とは異なり、main関数を用いることが出来ない。よって、タスクが一つしかないとしてもタスクの宣言を行う必要がある。

nxtOSEKはリアルタイムOSであるため、設定を行えばプログラム上でループを組むことなくタスクを周期的に動作させることが可能である（RTOSがタスクを決められた周期毎に呼び出す）。

タスクの定義及び設定はプログラムファイルとは別にOILファイル上に記述することで行う。OILファイル上でタスクの名前、優先度、実行周期を決定する。

タスクの処理内容はプログラムファイル上に記述する（詳しくは別ファイル参照）。

よって、OILファイルとプログラムファイルを合わせる事で、タスクのリアルタイム処理を実現することができる。

* 各センサの利用方法

各センサにはそれぞれ固有の関数（API）が用意されており、利用したいタイミングでそれらを呼び出すことによって各センサを利用することができる。例えば、光センサの値を取得したい場合は、プログラム内にecrobot\_get\_light\_sensor(NXT\_PORT)と記述すれば良い。この際、引数にセンサが接続されているポート番号を指定する必要がある。

* 特殊なAPI

nxtOSEKでは必要最小限のAPIを提供しているため、特殊なnxtOSEK特有の機能を持つAPIは無い。しかし、一定時間システム停止機能を提供するsystick\_wait\_ms()など、leJOS-NXJで提供されている機能を一般化したものも存在する。

* バランサーについて

バランサーについては、nxtOSEKが提供している倒立振子制御APIがあるため、それを利用すれば良い。使用する場合はそのAPIを4msで呼び出す必要がある。バランサーで使用しているパラメータは変更は可能だが、特に変更する必要は無かった。

* 開発環境

前提としてC言語がコンパイルできる環境。cygwin、GNUARM、nxtOSEKが必要。

* フック関数の存在

nxtOSEKでは、フック関数を利用することができるため、プログラム起動時の動作とプログラム終了時の動作（センサの起動、終了など）をまとめて記述することができる。

1. leJOS-NXJ特有の表現

* タスクの組み方

タスクを組む場合は、Runnableインターフェースを実装（implements）する必要がある。Runnableインターフェースではrunメソッドが定義されており、これを実装した物がタスクとなる。よって、各クラスにつきタスクを一つ定義できることになる。

また、runメソッドはインスタンスを通して呼び出すのでは無く、threadクラスのインスタンスを引数にrunメソッドを実装したクラスを指定して呼び出すことで生成し、startメソッドを呼び出すことでタスクが開始する。例えば、

Thread tracerThread = new Thread(tracer);

tracerThread.start();

と記述することで、tracerクラス内のrunメソッドがタスクとして動き出す。また、Threadオブジェクトを生成するためには、Threadクラスを継承する必要があり、タスクを持つ側とタスクを生成する側のそれぞれに特別な記述が必要となる。

ただし、leJOS-NXJは周期的な動作を提供していないため、while文を利用して繰り返し処理を実装することで擬似的に周期的な動作を実現する必要がある。

複数のrunメソッドを同時に動作させることも可能であるため、並列処理を実現することはできる。

* 各センサの利用方法

各センサはクラスとして表現されており、それらをインスタンス化することによってそれぞれのインスタンスを独立したものとして使用することができる。各クラス毎にメソッドが定められており、それらをAPIとして使用することができる。例えば、光センサの値を取得する場合は次のようになる。

private LightSensor light = new LightSensor(SensorPort.S3);

light.readNormalizedValue();

センサ類は値の取得が目的であるため、インスタンスの生成に関してはあまり考える必要が無いが、値を保持するモータ類やジャイロセンサに関しては、インスタンスを保持する場所について考慮しておく必要がある。

* 特殊なAPI

nxtOSEKは最低限の機能のみを提供しているが、leJOS-NXJはそれに加えてある程度応用的な機能も提供している。例えば、nxtOSEKでは光センサの取得値は0から1024の値に限定されていたが、leJOS-NXJではそれに加えてパーセンテージや最大値、最小値の取得も行うことが可能となっている。また、センサに対していくつかのモードを設定することも可能（例えば超音波センサに対しては、（後述）。

タイマーについては、タイマータスクを作成してその中で実装する必要がある。

* バランサーについて

leJOS-NXJでは標準の倒立制御APIは提供されていない。よって、自力で倒立制御APIを作成する、誰かが作成したAPIを利用する、倒立制御をしないなどでこの問題を解決する必要がある。

誰かが作成したAPIとしては、「銀河系はねうま」の作成した倒立振子制御 Java APIが広く使われている。このAPIはnxtOSEKと同様に利用することが可能である（オブジェクト化は必要）。

* フック関数

leJOS-NXJとしてはフック関数は用意されていないため、初期、終了時処理等はプログラム内でそれぞれ記述する必要がある。

* 開発環境

前提としてJavaがコンパイルできる環境が必要。その環境に加えてleJOS-NXJが必要。eclipseにプラグインを導入することも可能。また、ファームウェアをleJOS-NXJにする必要がある。

1. leJOS-NXJ、nxtOSEK共通の表現

* 言語が違うため一概には言えないが、プログラムの構造自体は同じもの

nxtOSEKはC言語、leJOS-NXJはJavaという大きな違いはあるものの、開発対象は同じものであるため、プログラムの記述、構造等は基本的には変わらない。

明確な違いとしては、nxtOSEKがタスク内に処理内容を記述するのに対して、leJOS-NXJは処理内容の中にタスクを記述するということである。

よって、（特殊な書き方ではあるが）nxtOSEKとleJOS-NXJでは基本的には同じモデルを題材にして同様のプログラムを作成できるのではないだろうか。

* 各センサを使用するために提供されている最小限の機能

各センサはその種類に応じた値（光センサ:輝度値、超音波センサ:距離など）を取得する機能を持つ。よって、値を取得する機能が必要となる。モータの場合は、制御信号を送信して、モータを回転させる機能が必要となる。このような最低限の機能はnxtOSKE、leJOS-NXJ共に兼ね備えており、当然といえば当然であるが各センサやモータの機能を使うための機能は提供されている。

* タスクという概念の存在

組込みシステムでは、自律した動作が基本となる。よって、ある動作を周期的に実行する機能が必要となるのだが、そのためにはタスクという概念が必要となる。そこでnxtOSKEとleJOS-NXJでは共にタスクを動作させるための構造が存在する。ただし、前述の通りタスクの実装方法は異なる。

1. nxtOSEKからleJOS-NXJに移植する場合、どのような変化が発生し得るのか？

* タスクの実行管理

nxtOSEKではタスクの起動は基本的にプログラムの実行開始と共に行われるが、leJOS-NXJでは各クラスがタスクの起動処理を行うため、タスクとクラスの関係性を事前に把握し、モデルに示す必要がある。

* センサの利用方法

nxtOSEKでは各センサの機能はAPIとしてどこからでもアクセスできるようになっているが、leJOS-NXJではAPIこそあるもののそれを利用するためには各センサ及びモータクラスのオブジェクトを生成する必要があるため、どのクラスがどのセンサ及びモータを管理するのかを把握し、モデルに示す必要がある。

また、同じセンサに対して複数のオブジェクトを生成した場合、それらはそれぞれ違った物であるため利用する場合は単一のオブジェクトを利用しなければならない可能性がある。そのためキャリブレーションなどで問題が発生する可能性が考えられる。

* 実行周期の管理

nxtOSEKではタスクの実行周期はRTOSが管理するためプログラマは１回のタスクにおける処理時間のみを考慮すればよいが、leJOS-OSEKはRTOSではないためタスクの実行周期をプログラマが管理する必要がある。

* 倒立振子制御の存在

nxtOSEKでは標準で倒立振子制御APIが提供されているため、それを利用すればすぐに倒立振子制御を行う事ができるが、leJOS-NXJでは提供していない。そのため倒立振子制御を行う場合は、外部のプログラムを利用するか開発者が自ら作成する必要がある。外部が提供しているものとしては、「銀河系はねうま」の提供している倒立振子制御APIが有名。

* 初期化をどうするべきか？

nxtOSEKではフック関数が提供されているため、初期処理で各センサのリセット等を容易に行うことができるが、leJOS-NXJではフック関数が無いため、各種初期処理について考える必要がある。対策として考えられるのは初期化関数の作成、コンストラクタの利用など。

* その他

開発環境に関しては、準備さえすればnxtOSEK、leJOS-NXJともに開発ツールが用意されているため特に問題は無い。

ただし、nxtOSEKは日本語のリファレンスが用意されているのに対し、leJOS-NXJでは英語のリファレンスしか用意されていないためある程度は英語の知識が必要となる。

その他は、基本的に言語が異なるだけと考えて良い。