組込みとOSSとアーキテクチャ

株式会社アックス 代表取締役　竹岡 尚三

1 はじめに

組込みシステムに使用可能な大規模ソフトウェアで、アーキテクチャや細部が公開されているものは、OSS（Open Source Software）に多いので、本稿では主にOSSについて述べる。

2 人工知能、機械学習

最近、人工知能技術に注目が集まっている。とくに機械学習、深層学習（DeepLearning）が注目されている。機械学習は、深層学習だけではなく、例えば、Support Vector Machine（SVM）なども広く使用されている。

これまで、計算機は、人間が現象やデータを解析し、モデル化し、それをプログラムに書き下したものを実行していた。それに対して、機械学習は、人間が解析していない画像などでも、学習用データを多量に用意し、学習させることで、未知のデータを判定できるようになる。人間が解析やプログラミングできないものを対象に、判定などを行ってくれる人工知能は、これまで人間しかできなかった知的作業の一部を肩代わりしてくれるものと、大きく期待されている。

ニューラルネットワークのような人間の脳を模した機械学習自体は、新しいものではなく、1950年代末期に、パーセプトロンが発表され、学習＆判定能力があることが分かっていた。続いて、多層パーセプトロンが発表され、それは既にバックプロパゲーション（誤差逆伝播法）を用いて学習を行っていた。1980年代後半にもニューラルネットワーク・ブームが起こり、1990年前後には、日本でも「ニューロ家電」や「ニューロ・ファジー家電」がたくさん作られた。これは、学習結果がROM化され、当時でも低性能なシロモノ家電用マイコンで実行された。その頃には、遺伝的アルゴリズム（Genetic Algorithm）や人工生命もブームとなり、機械が自分自身で学習し、生命維持の本能を獲得する、などの研究も多かった。機械学習は、最近に、目覚ましい進歩を遂げたわけではなく、例えば、国際電気通信基礎技術研究所（ATR）脳情報通信総合研究所所長の川人光男氏は、「1980年代の第２次ニューロブームのときのことを思い出さざるを得ない（中略）今の人工知能技術の基礎はいずれも当時出てきたもので、そこから本質的な飛躍はない」という主旨の発言をしている。※1

深層学習は、最近のGPGPU（汎用GPU）の出現により、小規模なハードウェア・システムでも実用的に多数のコネクションの計算ができるようになったため、急激に注目を集めるようになった。しかし、GPUほどの演算能力のない、直近の組込みCPUで扱うには、深層学習は負荷が大きい。

深層学習が注目される直前までの機械学習は、SVMがよく使用されていた。SVMは、ARM11程度のCPUでも十分に高速に動作し、現行の組込みシステムでも、即座に実用に供せる。弊社株式会社アックスでは、SVMを組込みに使用した実例として、顔認識を行い、顔を追尾する自走ロボットを試作した。このロボットは、Raspberry Pi（初代）を使用し、CPUはARM11 700MHzである。※2

機械学習は、学習のためのデータを用意して、学習を行わせる。一般に深層学習でもSVMでも、学習にはかなりな演算能力が必要で、通常のx86 PCで行っても、かなりの時間がかかる。深層学習はSVMなどに比べると、非常に演算が多いため、GPGPUの使用が望まれる。通常の深層学習は学習結果のデータの解析が不可能、若しくは困難で、学習させるためのデータをカット・アンド・トライで選択するということが行われている。深層学習のパラメータ調整と、学習データの選択は、ノウハウの固まりである。とくに学習データの選択方法は、今後も工学的アプローチがあまりできないと思われる。SVMは、学習結果はサポート・ベクトルとして得られ、判定対象もベクトルとして表現されるので、解析は可能である。しかし、ベクトルの各要素は、学習時に自動的に決定されるので、人間にとって直感的な解析が容易なわけではない。用途によるが、一般的に、機械学習は、学習結果が96％程度よりも高い精度を持たないと、実用的ではないと言われることが多い。

OSSで公開されている機械学習の多くは、学習コマンドと、認識サブルーチンなどから構成されている。つい最近は、Pythonから呼び出せる形式のものが多い。使用するには、学習データを用意し、学習コマンドを使用して学習を行わせる。得られた学習結果を、認識サブルーチンなどに設定して、認識対象をもって認識ルーチンを呼

び出せば、結果が得られる。認識をサブルーチン呼び出し形式ではなく、バッチ処理で行えるコマンドが用意されていることも多い。機械学習の内容など知らなくとも、機械学習を利用することは可能で、OSSの機械学習を利用することは、かなり容易である。

機械学習は、原理上、学習したことのないデータを入力された場合、何を出力するか分からない。認識率が99％であっても、１％の出力がとんでもない誤りの場合がある。そのため、力の大きな装置などへ機械学習を組み込む場合には、誤った出力が、機器を動かしてしまわないように、機械学習ではない機構（例えば、伝統的な通常プログラムによる動作抑止ルーチン）による安全策を講じておく必要がある。

機械学習は、いわゆる人工知能で、ロボットの頭脳として、非技術者には、何でもできると思われているようである。しかしながら、機械学習は画像認識などには力を発揮するが、不向きな用途もある。弊社では、ルール・ベースの推論システムと機械学習を組み合わせたハイブリッドなAI「ごまめ」を開発し、使用している。

3 最近の組込みアーキテクチャ

旧来からの組込みシステムは、リアルタイム・モニタや小さなOSカーネルに、いわゆるミドルウェアなどの層を積んでいく形式が多かった。例えば、スマートフォンOSであるAndroidなどは、Linuxカーネルに、ミドルウェアの層を積んでいく形になっている。

だが、最近の大規模な組込みシステムでは、少し様相が変わってきた。例えば、1980年代より「ソフトウェア・バス」というような言葉で言われたものが、現実になってきている。ここで言う「ソフトウェア・バス」とはソフトウェアのモジュール同士をつなぐ役目だけを果たすミドルウェアの一種であり、部品となるソフトウェア・モジュールには、インターフェースの切り口を提供する。各ソフトウェア・モジュールは共通のインターフェースを使用して、機能を提供する。名称は変わりつつも、ソフトウェア・バス的な構想は、何度も現れては、消えていった。

その理由の幾つかは、ソフトウェア・バス自身がメモリを消費する、モジュール間の通信コストが有意な負荷となる、（旧来の）組込みシステムでは、さほど大きなソフトウェア・モジュールが多くはなかったハードウェア進化が早く、ソフトウェア資産の陳腐化が早過ぎたなどであろうと思われる。

しかし、現在は、ハードウェア資源（CPU性能、メモリ）が潤沢になり、同時に大規模なソフトウェアが増え、その資産を活かしながら、新しいソフトウェア・モジュールを加えていくようにしたいという時代になってきた。

また、情報処理アーキテクチャとしては、前処理-処理-後処理のように、情報/データの流れとして捉え、前処理や後処理は既存のものを使用し、ここでいう「処理」部分に、自分の作成したモジュールを差し込みたい、という要請がある。

とくに、ロボット分野では、それが成功しており、「ロボット・ミドルウェア」と呼ばれる、国立研究開発法人 産業技術総合研究所（産総研）のRTM（Robotics Technology Middleware）や、ROS（Robot Operating System）が多くのシステムで採用されている。

4 ロボット・ミドルウェア

ロボット・ミドルウェアは、ロボットの内部で、各ソフトウェア・モジュールをつなぐためのフレームワークであり、次のような要請がある。※3

- コンポーネントをつなぐ
- コンポーネントの独立性を高めさせる
- オーバーヘッドなし
- 存在を感じさせない

産総研のRTM（OSS版はOpenRTM-aist）は、オブジェクト通信の組込み版であり、RTMを使用する各モジュールは、RTC（RT Component）と呼ばれる。RTC間のインターフェースが、OMG（Object Management Group）において2008年に国際規格となり公開されている［図１］。

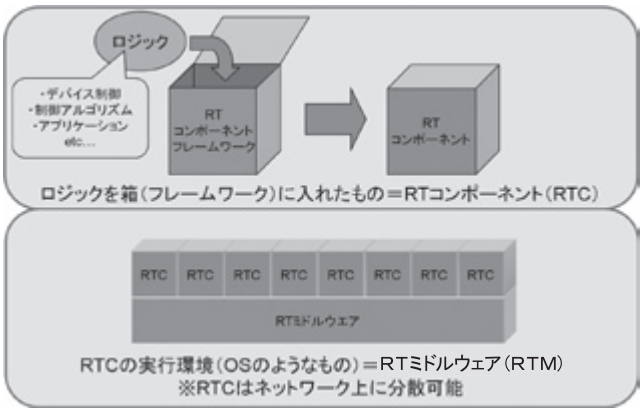

図１　RTコンポーネントとRTミドルウェア※4

ROSは、Google社の自動運転に採用され、著名になったロボット・ミドルウェアである。

ROS自体は、モジュール間の通信を行うソフトウェアであるが、ROSパッケージとして配布されている。

ソフトウェアには、ロボット開発に有用な大規模ソフトウェアがたくさん含まれている。

RTM、ROSとも、存在を感じさせない通信ミドルウェアとなっており、ユーザは、ソフトウェア・モジュールをつなぎ合わせて、所望のシステムが構築できるようになっている。RTMであれば、カメラなどの視覚モジュール、マイクが入った聴覚モジュール、モータなどの運動モジュールを組み合わせてロボットを構成する［図２］。

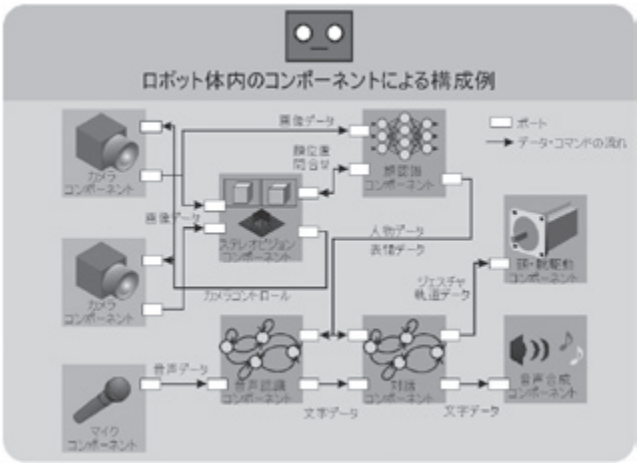

図２　RTコンポーネントによる構成例※4

ROSは、データの処理の流れに合わせて、ソフトウェア・モジュールをつなぎ合わせるというアーキテクチャとなっている。ROSには、知覚、物体認知、セグメンテーションと認識、顔認識、ジェスチャ認識、動作追跡、ステレオビジョン：２台のカメラを通して、奥行き知覚など数多くのソフトウェア・モジュールが存在している［図３］。

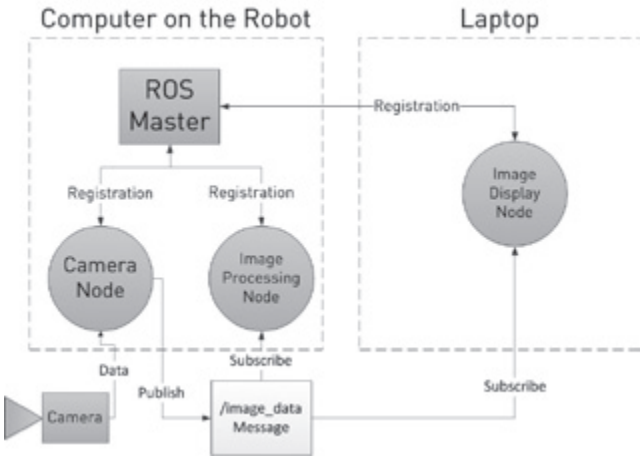

図３　ROS：データの流れに応じて、コンポーネントをつなぐ※5

また、それより低層ではOpenELというハードウェア抽象化層が組込みシステム技術協会（JASA）などを中心として開発され、同時に国際標準化の活動を行っている［図４］。OpenELにはOSS実装も存在し、配布される予定である。

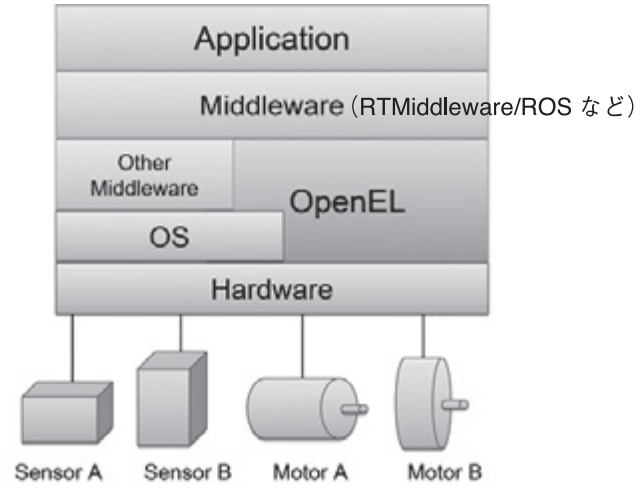

図４　OpenELの位置付け※6

5　自動運転、ROS

自動運転は、ロボットの一種である。株式会社アックスは、日本で最も先進的な研究の一つである加藤真平先生（東京大学/名古屋大学）のプロジェクトにかかわっている。また、名古屋大学発ベンチャーの株式会社ティアフォーは、そのプロジェクトの成果物であるAutowareの発展を推進している。

自動運転技術には多数の要素がある。Autowareは、Googleの自動運転と同様にROSを元にして、ROSのコンポーネントを開発することで、システムを作り上げている［図５］。

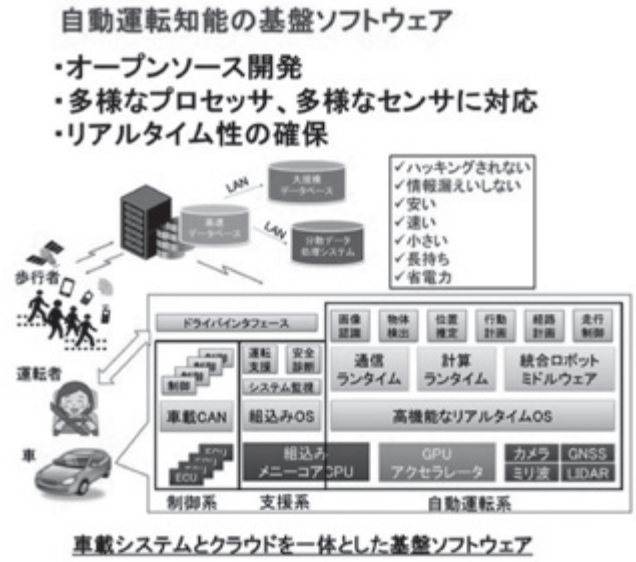

図５　自動運転OSS Autoware 構成・特徴※7

ROSには、そもそも、ロボット開発に必要な、大規模なミドルウェアやコンポーネントが相当数そろっている。そ

の中には、画像認識、立体視、ジェスチャー理解、立体構造理解などを容易に実現できるコンポーネントなどが含まれている。ROSのコンポーネントは、データフローに基づいて組み合わせるようになっている。独自に開発したコンポーネントも、既存のコンポーネントと組み合わせて、一連のデータフロー処理の中に容易に組み込める。

自動運転には、三次元地図、自車がどこに居るかを認識する、信号を認識するなどの技術が必要である。それらに加えて、路上の障害物や、実時間で道路を認識することで、安全に走行することができる。

6 ドローン

最近、ドローンについても注目が集まっている。OSS界では、LinuxFoundationが、ドローンに関するプロジェクトを取りまとめて“Dronecode”に集約した。

Dronecodeの回りでは、ハードウェアもデファクト・スタンダード化が進んでおり、基板のサイズやコネクタの位置が標準化され、ドローンの機体に数種類の基板を交換して搭載することが可能になっている。

Dronecodeは、機体に搭載する（オンボード）のソフトウェア［図６］と、遠隔操作する側（オフボード）のソフトウェア［図７］に大きく分かれている。

オンボードのソフトウェアは、実時間性が高く、回転翼を制御し、バランスを取ってドローンを飛ばすソフトウェアと、Wi-Fiからの指令を受けるソフトウェアなどから成っている。

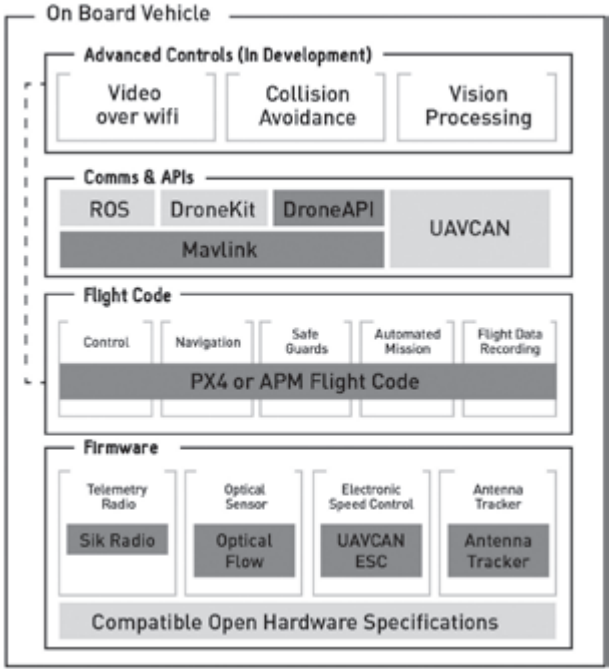

図６　Dronecodeのオンボード・ソフトウェア※8

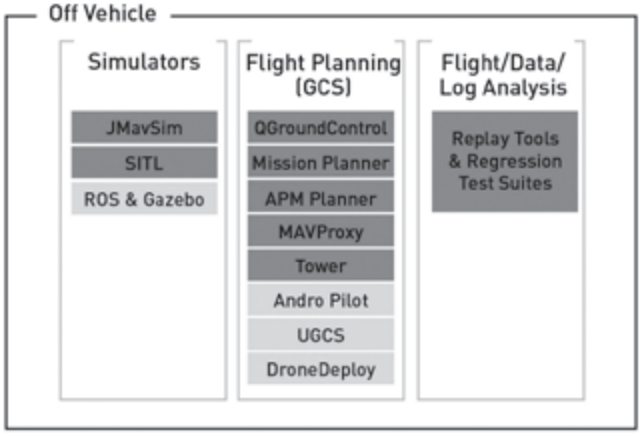

図７　Dronecodeのオフボード・ソフトウェア※8

オフボードのソフトウェアは、Linuxなどで構成されており、スティックなどの入力装置を読み取り、Wi-Fiなどを経由して、ドローンへ送信する。

日本の通常のドローンは、法令により、目視可能な範囲しか飛行できない。よって、通信方式はWi-Fiが主流である。

7 まとめ

このように、現在、注目を集めている技術の多くは大規模なOSSとして公開されており、それを元に試作を行い、技術検証を行うことは、容易で有意義である。また、用途によっては、既存OSSをそのまま組み込んで実用製品にできる。

しかし、生命にかかわるようなクリティカルな分野では、OSSの品質をどう捉えるべきか、など、既存のOSSを実際の製品の中に組み込んでいくことには、まだまだ、取り組むべき問題が多い。

それらの問題を踏まえた上で、組込みOSSを大いに活用していき、また、日本からOSSを発信していきたい。

【脚注】

※1　引用元：http://pc.watch.impress.co.jp/docs/column/kyokai/723879.html
※2　日経Linux 2014年１月号, “Raspberry Piで作る、顔認識でモジモジ近づく自走ロボット”, 竹岡尚三
※3　引用元:http://en.wikipedia.org/wiki/Robotics_middleware（左記より引用し和訳）
※4　引用元:http://www.openrtm.org/openrtm/ja/content/rt%E3%83%9F%E3%83%89%E3%83%AB%E3%82%A6%E3%82%A8%E3%82%A2%E3%81%A8%E3%81%AF%EF%BC%9F-0
※5　引用元：http://www.clearpathrobotics.com/blog/how-to-guide-ros-101/
※6　引用元：http://jasa.or.jp/openel/Main_Page/ja
※7　引用元：http://www.pdsl.jp/fot/autoware/
※8　引用元：http://www.dronecode.org/sites/dronecode/files/pages/images/dronecode_figure1_v3_ac-01.jpg（左記より引用し分割）