ドローン&ロボット、機械学習OSSの紹介と、OSSの品質についてのアプローチ

2016年11月16日

OSS活用WG/技術本部副本部長/
技術高度化委員長
竹岡尚三 (株)アックス/Tier IV

今、OSS
(オープンソース・ソフトウェア)
なのか?

自動運転もOSS!「Autoware」

- 名古屋大学 加藤真平先生 日本で最も進んだ自動運転 研究
- 名古屋大学 開発 自動運転ソフトウェア「Autoware」サポート
- 名古屋大学ら、開発済み自動運転システム一式をオープンソース化…加藤准教授「時間をジャンプ」
  - Autowareは、オープンソース・ソフトウェアとして無償配布されている

http://response.jp/article/2015/08/26/258648.html

大人のOSS使用は、すでに十数年の歴史あり

■1980年代中,後期～
- Ciscoは、BSD UNIXベースのルータ

■1990年代末期～
- Linuxが、家電各社(SONY,パナソニック,シャープなど)にて採用
- 半導体企業も独自CPUに、Linuxを移植

■ちょっと前～
- Android, Linux, BSDが無ければ、IT機器を作るのが大変
  ※JASA会員企業 いわく:
  お客様の指定で、OSS採用を行わねばならない!

■ナウ
- ロボット用 OSSが、続々と出現中
- 機械学習OSSも、たくさんあり!

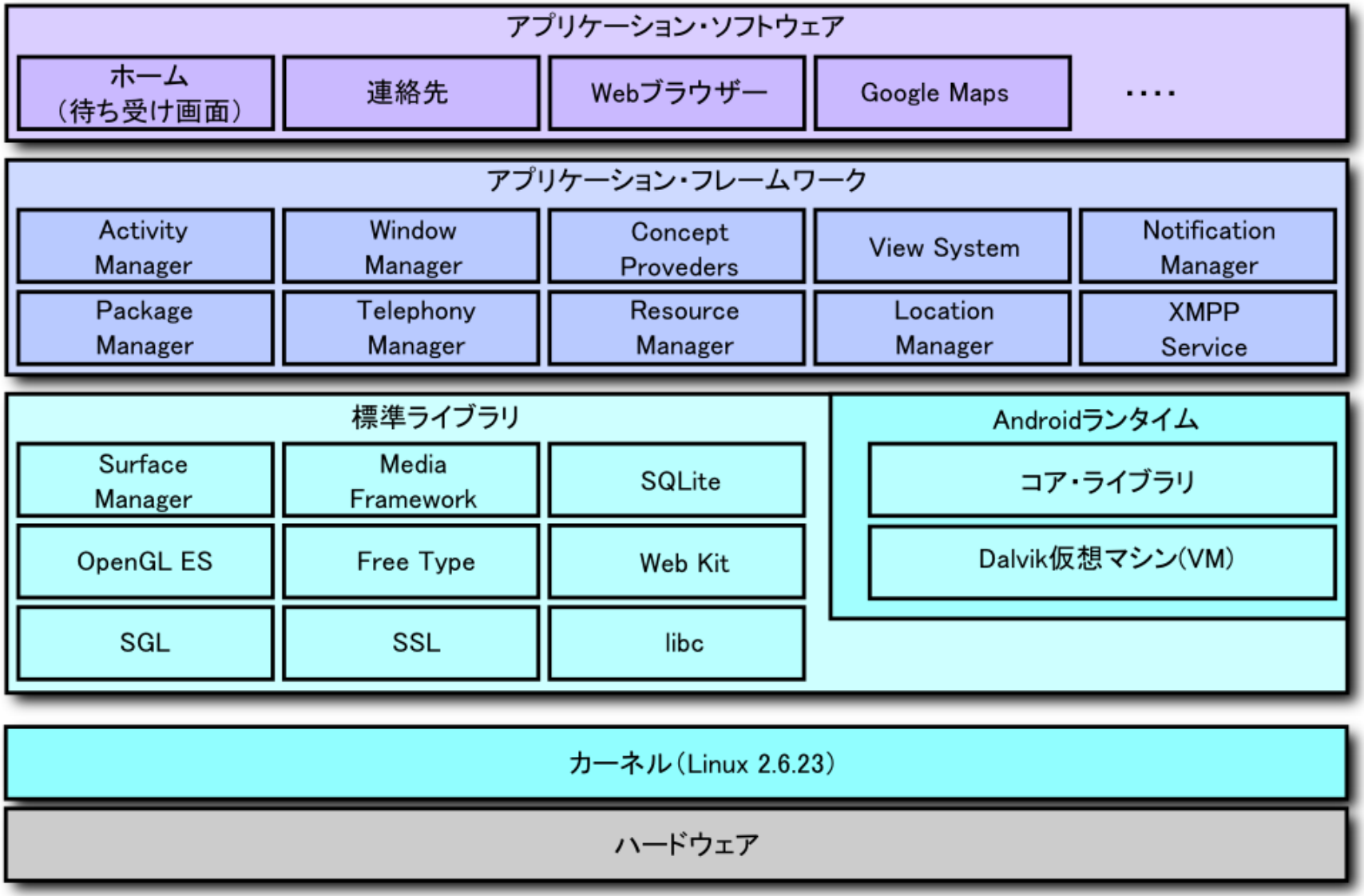
IoT,M2Mもオープンソース技術

- Arduino
  - オープンソース・ソフトウェア (OSS)
    ＋
  - オープンソース・ハードウェア (OSHW)
  - OSS
    - 開発環境 IDE
    - ライブラリ: 膨大な量のライブラリ
    - OSSで、世界中のみんなが開発
  - OSHW
    - 回路図
    - 基板レイアウト
  - お手軽試作にバッチリ
    - ハードウェアは、アマチュア品質だが…
- Raspberry Pi
  - Linux搭載、ハードウェア情報公開

OSSで、巨大ソフトウェア・スタック

- OS(Linux, BSD, OpenSolaris)
- インターネット接続
- Windowシステム(窓、描画管理)
- 文字レンダリング
- 3Dグラフィックス
- 仮名漢字変換
- ブラウザ、メーラ
- マルチメディア(音楽、動画)の記録/再生
- プログラミング言語 Ruby,Perl,PHP…
- オフィススイート

Androidのアーキテクチャ



- 蓬田宏樹、他著「Androidの野望」日経エレクトロニクス 2007年12月17日号 p.47-69 より引用

OSSの本当の利点

■「無料だから嬉しい」とか言ってると、負ける

■特定企業のOSとは違い、ソースがあるので、
理解し、独自の改良が可能

■デファクト・スタンダードがOSSなら、
特定ベンダに囲い込まれない

■自分の都合でシステムをリリースできる
- 特定のソフトウェアのリリース(バージョンアップ)に引っ張られない
  - 独自に品質を上げたソフトウェアを、自由に維持できる

OSSって面倒くさい !?

- ライセンス問題
  - ソースコード公開義務(!?)
  - 誤解が多い
- 品質問題
  - 適当に作られた無料のソフトウェアの品質は?
    - どこの誰だか、知らない人が作ったものでしょ…
    - タダのものがそんなに良ければ、プロは商売あがったりだよ
      - タダのものが、かなり良い、ということは、非常に頻繁にある
- 品質の押さえ方は、大事
- 実は、タダ乗りも多い
  - タダ乗りは、悪いことではない
- OSS共通の問題をみんなで考えよう

→ JASA　OSS活用委員会

OSSのせいで、仕事が減る?!

- これまで
  - 受託して,仕様を起こし、一品づつ作っていた
  - ライブラリ/モジュール(サブルーチン)を作って販売していた
- しかし、現在
    - JPEG デコード・ルーチンの開発を委託する顧客がいるだろうか?

      →OSSに乗るしかない
- 顧客がOSS前提で仕様を決めている
  - サービス主導の時代→OSSで素早くシステム構築

- OSSを活用する世界をみんなで考えよう

  → JASA　OSS活用委員会

OSSに乗るしかない!

- 顧客が、OSS前提で仕様を決めている
  - サービス主導の時代→OSSで素早くシステム構築
  - Windows+ブラウザで、できていることは、できて欲しい
- 高度なロボットを、すぐに作りたい
- 人工知能(AI)が入った機器をすぐに作りたい

- OSSを活用する世界をみんなで考えよう

→JASA　OSS活用委員会

GPL

■ GPL
- ソースコードは、そのバイナリを持っている人には、公開されるべき。という思想
- 改変したソースコードも、公開されなければならない
  ―独自に改良したコードも公開しなければならない
  ―(組み込みの世界では、厳しいと言えるだろう)
- GPLのソフトウェアにリンクしたソフトウェアのソースコードも、公開しなければならない
  ―独自に作ったソフトウェアも、GPLソフトウェアにリンクしたら、自動的に公開義務が生ずる
  ―(極めて厳しい、と言えるだろう)

■ LGPL
- ライブラリのために作られたライセンス
- LGPLライブラリは、ダイナミック・リンクであれば、リンクを行っても、他のソフトウェアに、GPL(LGPL)が伝播することは無い
- 小さな組み込みシステムでは、ダイナミック・リンクが現実的ではないかも
- → 結局、小規模 組み込みシステムでは厳しい、と言えるかも

■ GPL v3問題
- ソースコードを寄付した人が持っている特許が、そのソースコードに含まれていた場合、当該 特許を無償で無制限に許諾しなければならない
- 大企業の多くが、GPL v3のソフトウェアの開発には関わっていない
  ―Linuxは、GPLv2 維持宣言をしている(2014年秋 現在)

GPL:よくある誤謬

■ソースはインターネットで配布しなければならない
- 嘘です

■ソースを誰にでも上げなければいけない
- 嘘です
- バイナリを持っている人にだけ、ソース入手の権利があります

■誰とライセンス契約をすればいいのか
- する必要はありません

■GPL違反をするとストールマンに訴えられる
- 嘘です
- 日本では、プログラムの著作権者しか訴える権利がありません
- ストールマンが怒る可能性はありますが、それには法的な意味はありません

BSD/MIT風ライセンス

■BSDライセンス
- MITライセンスは、BSDと同じ
- 何をしても、ソースコードの開示義務は無い
- 古いBSDライセンスは「宣伝条項」があったが…
  ―宣伝条項: すべての広告にライセンス表示を入れる
- 現在のBSDライセンスは、「宣伝条項」は無い

■Apacheライセンス
- Apache Licenseのコードが使われていることを知らせる文言を入れる。という義務がある程度
- 何をしても、ソースコードの開示義務は無い
- ライセンスを変更してもよい
- ライセンスされたファイルに元々ある著作権と特許権の記述はそのまま保持
- Apache Licenseのコード中に特許を含んでいる場合、特許を無償で無限に許諾しなければならない

ライセンスの知識は必要

■正しい知識があれば、問題は出ない
- GPLであっても、ソフトウェア作成者は、
- みんなに使って欲しいと考えている
  → ユーザが困るようなことはしていない(はず)

■ズルは止めよう
- GPLのコードの断片を、独自開発のソフトウェアに混ぜると、公開義務が生じる
  ーBSDライセンスのものは、ライセンス表示をソースコードに入れれば、問題無い

ロボット用
OSS

ロボットもOSS時代

■英語版Wikipedia「Open-source robotics」の項

http://en.wikipedia.org/wiki/Open-source_robotics

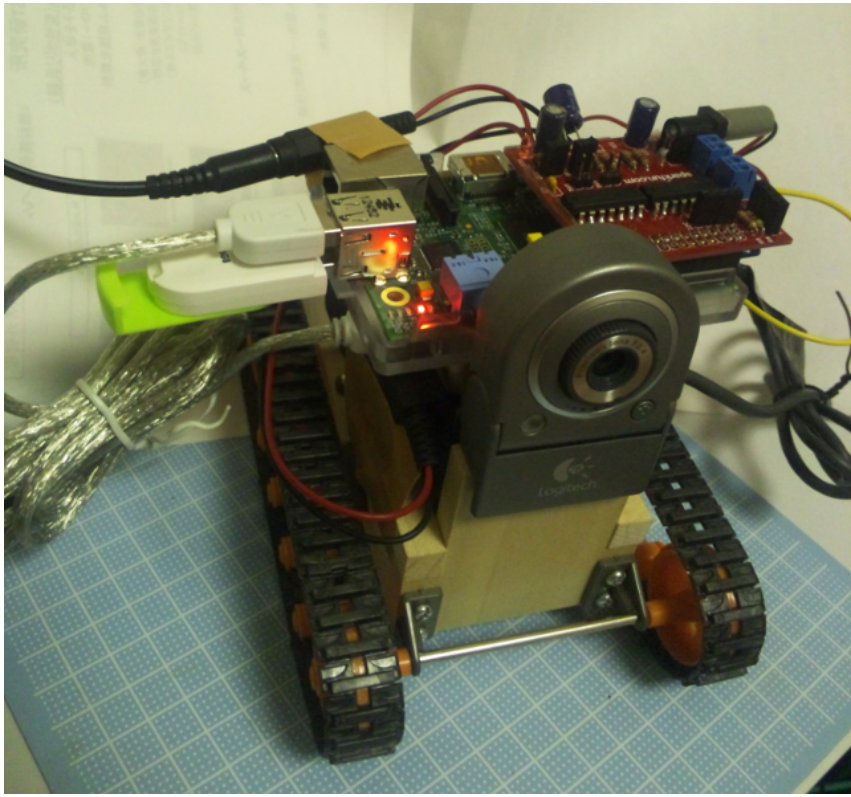
■フル・ロボット・プロジェクト32個

■ソフトウェアのみのプロジェクト 28 個

■オープン・ハードウェア・プロジェクト:3個

OSSロボット例 (たけおか謹製)

- 人間の顔を見つけたら、追尾する
  - 顔の位置を判断して、右へ行くか、左へ行くかを決める
- Linux+OpenCV+OpenEL
- OpenCVは顔認識などを含む
- 総合的な画像処理ライブラリ
- DCモータの制御は**OpenEL**
- OpenELはJASAなどが推進している
- ロボット用の下位ハードウェアの
- 抽象化層の規格
- ハードウェア
  - RaspberryPi
    ーARM11@700MHz
    ー512MBytesRAM
  - USBカメラ
  - DCモータ



OpenCV

■画像処理のミドルウェア

・Intel, Willow Garage が開発

・いろんなものが入っている

・極めて便利

ー画像処理 (Image Processing)

ー構造解析 (Structural Analysis)

ーモーション解析と物体追跡 (Motion Analysis and Object Tracking)

ーパターン認識 (Pattern Recognition)

ーカメラキャリブレーションと3次元再構成 (Camera Calibration and 3D Reconstruction)

ー機械学習

ーユーザインタフェース

OpenCV(機械学習)による顔認識

■OpenCV
- 一般的画像処理
- 画像認識
  - ―学習器
  - ―認識器
- カメラ入力部

■USBカメラを接続
- ロジクール QCAM-E2500, 30万画素
- Logicool 861200
- カメラ入力

■顔認識
- 学習済みデータが用意されている
- 簡単に使用できる
- ロボットの目玉にでも

「たけおか opencv raspberrypi」でググる

自動運転
ミドルウェア
OSS

自動運転もOSS!「Autoware」

- 名古屋大学 加藤真平先生のAutoware
  - 日本で最も自動運転の研究が進んでいる
  - 名古屋 守山市で、公道を自動運転走行
  - 自動車会社もスポンサー
- http://www.pdsl.jp/fot/autoware/
- 愛知県知事が名古屋大学自動運転カーに試乗
- http://www.aisantec.co.jp/ir/library/as20150618.pdf
- 名古屋大学 開発 自動運転ソフトウェア「Autoware」サポート
- 名古屋大学ら、開発済み自動運転システム一式をオープンソース化…加藤准教授「時間をジャンプ」
  - Autowareは、オープンソース・ソフトウェアとして無償配布されている
- http://response.jp/article/2015/08/26/258648.html
- http://www.pdsl.jp/%E6%97%A5%E6%9C%AC%E8%AA%9E%E3%83%88%E3%83%83%E3%83%97/
- http://news.mynavi.jp/series/coolchips18_auto_car/003/

自動運転OSS Autoware

基本機能

- 3次元自己位置推定
- 3次元地図生成
- ナビアプリ
- 経路生成
- 経路追従（0～60km/h）
- 交差点右左折／一旦停止
- 自動停止
- 自動駐車
- 車両認識
- 歩行者認識
- レーン認識
- 標識認識
- 路上サイン認識
- 信号認識
- 移動体追跡
- レーンチェンジ

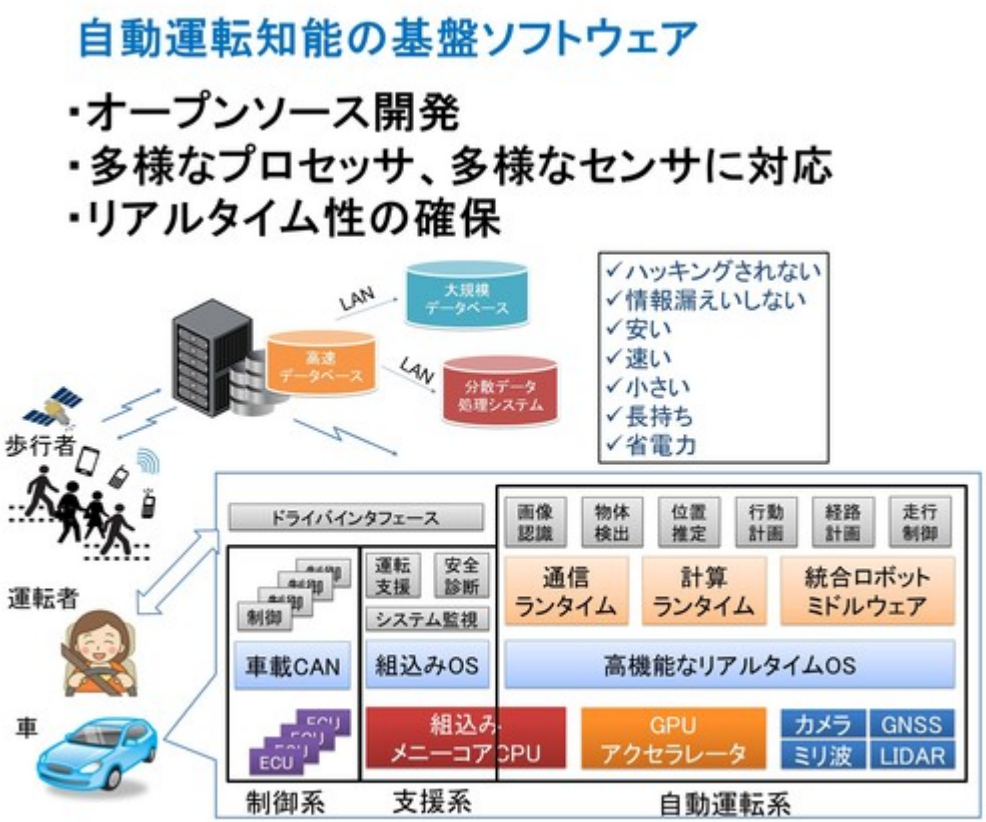
http://www.pdsl.jp/fot/autoware/ から引用

自動運転OSS Autoware 構成・特徴

- ZMP社のロボカー製品を車両として利用可能
- アイサンテクノロジー社の高精度3次元地図を利用可能
- 測位衛星技術社、Javad社のGNSS測位機を利用可能
- 北陽電機社、Velodyne社の3次元スキャナを利用可能
- インクリメントP社のAndroid端末用ナビアプリを利用可能
- インテル社のチップほか, Linuxが稼働する環境で利用可能
- NVIDIA社のGPGPU技術（GeForce, Tegra）を利用可能
- イーソル社の車載向け組込みシステムに対応
- インターネット上のデータベース接続などもサポート
- TierIV/アックスが有償サポート提供

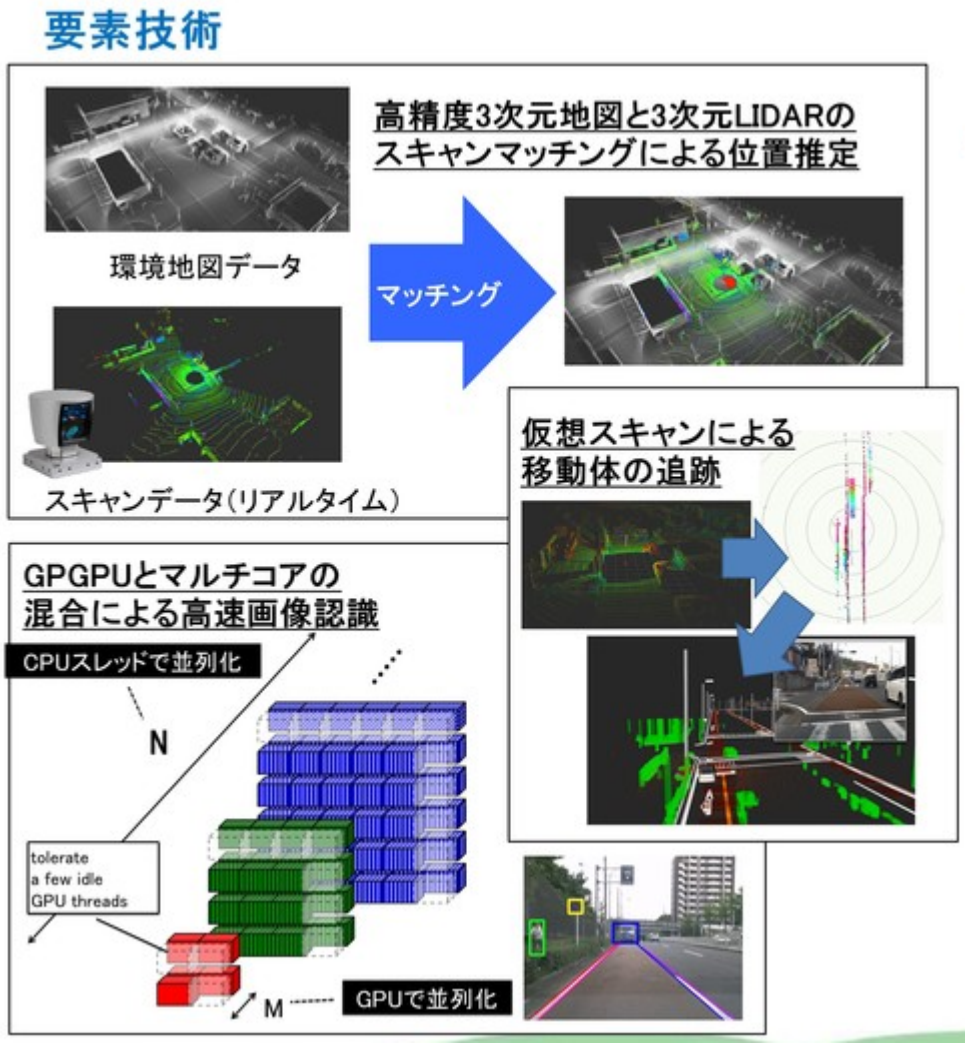
http://www.pdsl.jp/fot/autoware/ から引用。一部、付加

自動運転OSS Autoware 構成・特徴



http://www.pdsl.jp/fot/autoware/ から引用

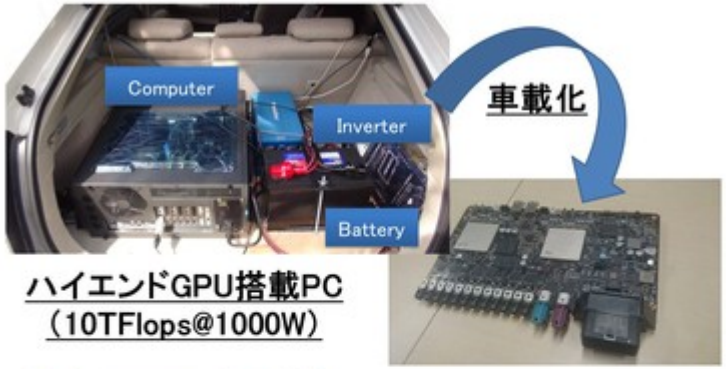
自動運転OSS Autoware



http://www.pdsl.jp/fot/autoware/ から引用

自動運転OSS Autoware

コンピューティングハードウェア



ハイエンドGPU搭載PC
（10TFlops@1000W）

組込みGPU搭載Drive PX
（1TFlops@10W）

全方位カメラ用FPGA

移動体データ解析用
Hadoop PCクラスタ
&
ディープラーニング用
超並列GPGPUクラスタ

車両および環境センサ

・リファレンスカー
・多様なセンサを搭載して性能検証

Velodyne HDL-64/32
(3D LIDAR)

Point Grey Ladybug 5
(Camera)

JAVAD RTK-GNSS
(GNSS/GPS)

Point Grey Grasshopper3

IBEO LUX 8L (3D LIDAR)

http://www.pdsl.jp/fot/autoware/ から引用

ドローン用
OSS

Dronecode

https://www.dronecode.org/

■Linux Foundationの取りまとめで

■無人飛行体 Unmanned Aerial Vehicles (UAVs)を開発

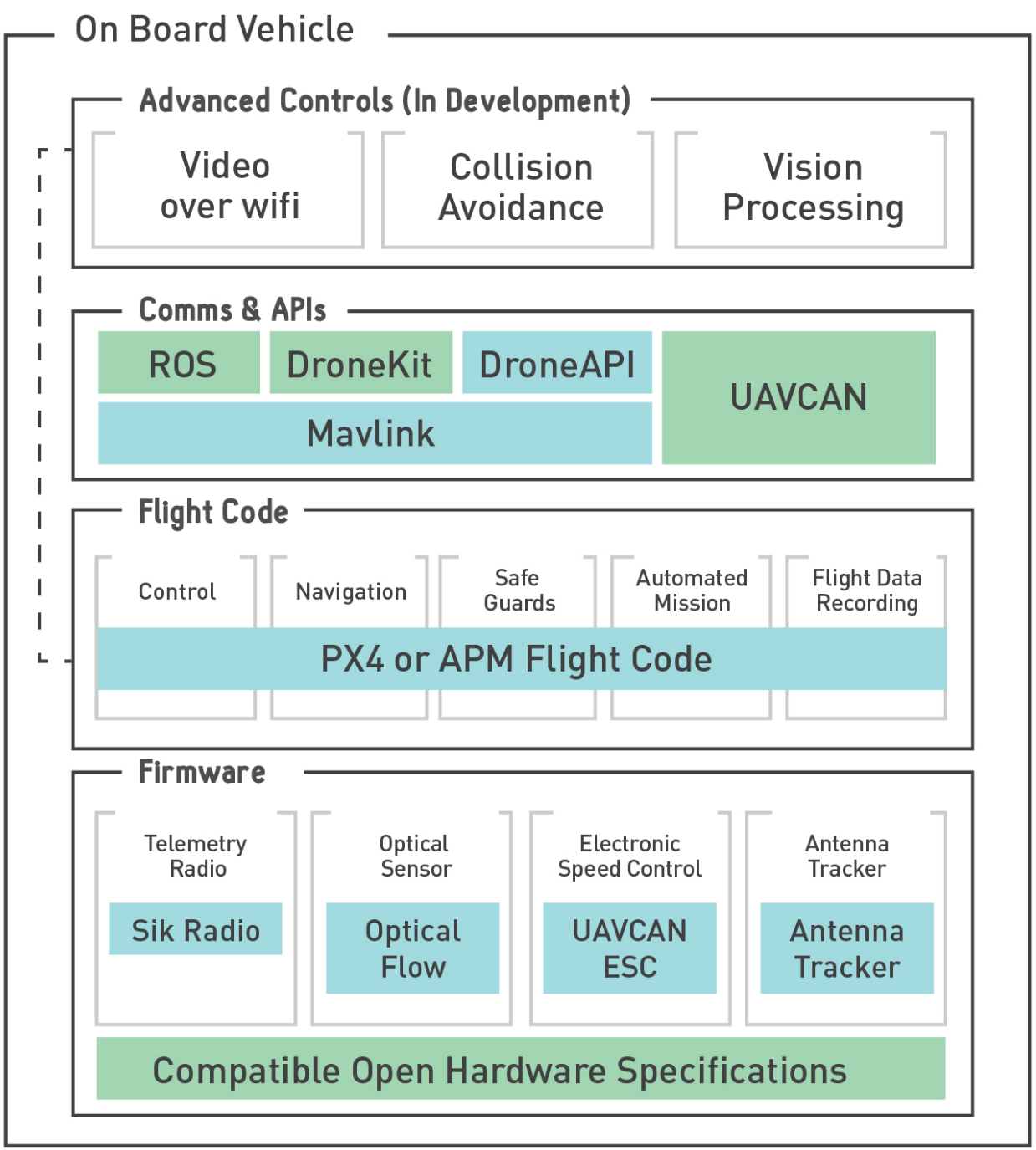
■オープンソース・ソフトウェア + オープンソース・ハードウェア

■オン・ボード(機体上)と、オフ・ボードのソフトウェアがある

- オンボード: 機体の制御、センサー制御
- オフボード: リモコン、データ収集/記録

写真は
https://www.dronecode.org/sites/dronecode/files/styles/dronecode_header/public/front_page_slides/images/drone_video_slide.png?itok=7C7IFYIP より引用

Dronecode



https://www.dronecode.org/sites/dronecode/files/pages/images/dronecode_figure1_v3_ac-01.jpg を引用、分割

Dronecode

Off Vehicle

Simulators

- JMavSim
- SITL
- ROS & Gazebo

Flight Planning (GCS)

- QGroundControl
- Mission Planner
- APM Planner
- MAVProxy
- Tower
- Andro Pilot
- UGCS
- DroneDeploy

Flight/Data/ Log Analysis

- Replay Tools & Regression Test Suites

https://www.dronecode.org/sites/dronecode/files/pages/images/dronecode_figure1_v3_ac-01.jpg を引用、分割

Dronecodeのソースコード

Dronecode は、いくつかのプロジェクトとリポジトリでできている。
ソースコードは、下記から
Ardupilot https://github.com/diydrones/ardupilot
PX4 https://github.com/PX4
Pixhawk https://github.com/Pixhawk
MAVLink https://github.com/mavlink
UAVCAN https://github.com/uavcan
ROS https://github.com/ros
Other repositories https://github.com/Dronecode

写真は http://diydrones.com/profiles/blogs/arpx4-drone-px4-ar-drone-kits より引用

機械学習
OSS

機械学習はDeep Learning だけじゃない!

■深層学習 (Deep Learning)
- 多層パーセプトロン(Perceptron)の強化形
  - パーセプトロンは、1950年代末期からある
  - バック・プロパゲーションを色んな層に
- 良い学習のためには、演算が多い
- 学習結果、判定結果は解析不能
  - 本質的に。

■サポート・ベクタ・マシン (Support Vector Machine)
- 試料をベクトル化して、比較
  - ベクトルの要素は、機械が決定する。が、人間がベクトルを見て、解析することは可能
- Deep Learningと比べて演算が少ない

■Google自動運転でDeep Learningが有名になるまでの10年間ほどは、SVMがとても多く使われていた

組み込み用 Deep Learning OSS

■組み込みで使えるものを紹介
- 独立して動作
  - クラウドとかサーバは不要
- 割と新しいもの

■でも、学習時には、GPUが必要かも…

TensorFlow (Deep Learning OSS)

- Googleが開発
- Apache 2.0ライセンス
- AndroidやiOSでも動作
- Pythonから使いやすい
- https://www.tensorflow.org/

Chainer (Deep Learning OSS)

- 日本の Preferred Infrastructure 社が開発
- ライセンス
  - 基本的に、改変、再配布など可能。詳細は下記
  - https://github.com/pfnet/chainer/blob/master/LICENSE
- Pythonから使いやすい
- X86 CPU以外で動作している情報が無い…
- http://chainer.org/

Caffe (Deep Learning OSS)

- UC BerkeleyのBerkeley Vision and Learning Center (BVLC)が開発
- 速い
- Pythonから使える
- http://caffe.berkeleyvision.org/

Support Vector Machine(SVM) OSS

- SVMは、計算が少ない
- ARMなどの組み込みCPUで十分に動作可能

SVM OSS その1

- SVM-Light
  - Thorsten Joachims @Cornell University が開発
  - Linear kernelの学習時間は短い
  - 精度はいい
  - http://www.cs.cornell.edu/People/tj/svm_light/index.html
- SVM-perf
  - Thorsten Joachims @Cornell University が開発
  - 学習時間は短い
  - 精度はいまいち
  - http://www.cs.cornell.edu/People/tj/svm_light/svm_perf.html

SVM OSS その2

- LIBSVM
  - 国立台湾大学のChih-Chung Chang と Chih-Jen Linが開発
  - 学習時間は長い
  - Linear kernelの精度は SVM-Light並
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- LIBLINEAR
  - 国立台湾大学 Machine Learning Group が開発
  - 学習時間がかなり短い
  - 精度はいい
  - http://www.csie.ntu.edu.tw/~cjlin/liblinear/
- CvSVM (OpenCV)
  - OpenCVに入っている
  - よく使われている
  - http://opencv.jp/opencv-2.1/cpp/support_vector_machines.html

ロボット用
OSS
ミドルウェア

en.Wikipedia「Open-source robotics」のOSS上位など

■NXJ
- Lego NXTロボット・キット用のOSS Javaプログラミング環境

■CLARAty
- JPL(ジェット推進研究所)で開発されている、Mars(火星)プログラムの一部として。

■ROS (Robot Operating System)
- BSDライセンス。すでに50以上のロボットで動作

■URBI
- C++の分散/組み込み コンポーネント・フレームワーク
- ＋ 並列/イベント駆動 制御スクリプト言語

■Player (robot framework)

■OpenRTM-aist (robotics technology middleware)
- 日本の産総研が開発

※上記のいくつかは、ロボット・ミドルウェアである

ロボット用ミドルウェアとは

■フレームワーク

・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
・オーバーヘッド無し
・存在を感じさせない

参考
http://en.wikipedia.org/wiki/Robotics_middleware

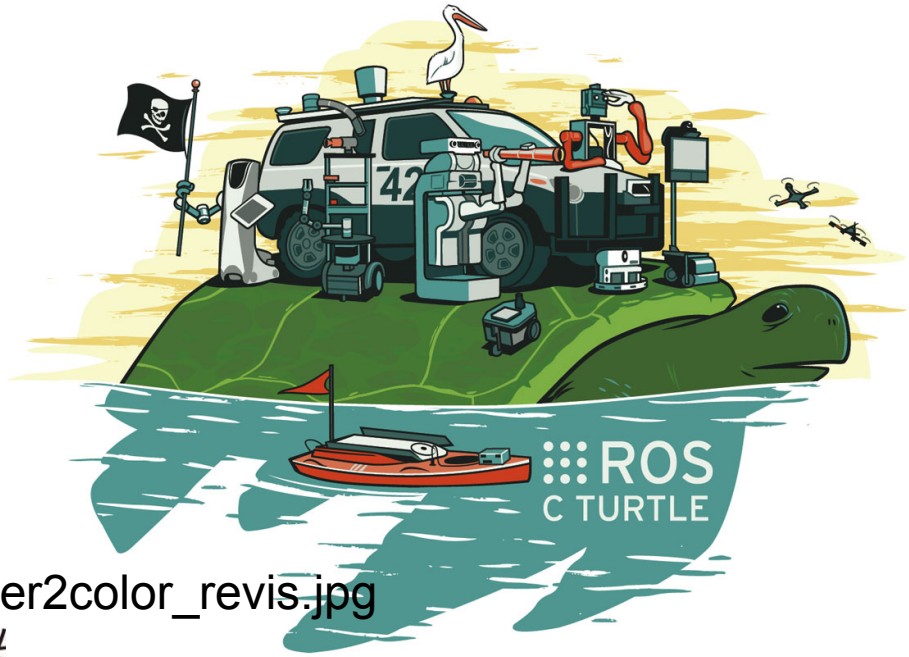
ROS

■Robot Operationg System
http://wiki.ros.org/
■ロボットのミドルウェアと、たくさんのコンポーネント
■BSDライセンス
■すでに50以上のロボットで動作
■ 2007年 the Stanford Artificial Intelligence Laboratory で開発
・Stanford AI Robot STAIR project
■2008～2013年 Willow Garageが中心に開発中
■2013年～ the Open Source Robotics Foundation

※参考
http://en.wikipedia.org/wiki/Robot_Operating_System

ROS

- データの流れに応じて、コンポーネントをつなぐ
- 自動車の自動運転でも採用
- OpenCVも含まれている
- 雑に言ってしまえば…
- ロボットを作るためのソフトウェア部品の多くが含まれている



画像は下記より引用
http://www.ros.org/news/resources/2010/poster2color_revis.jpg

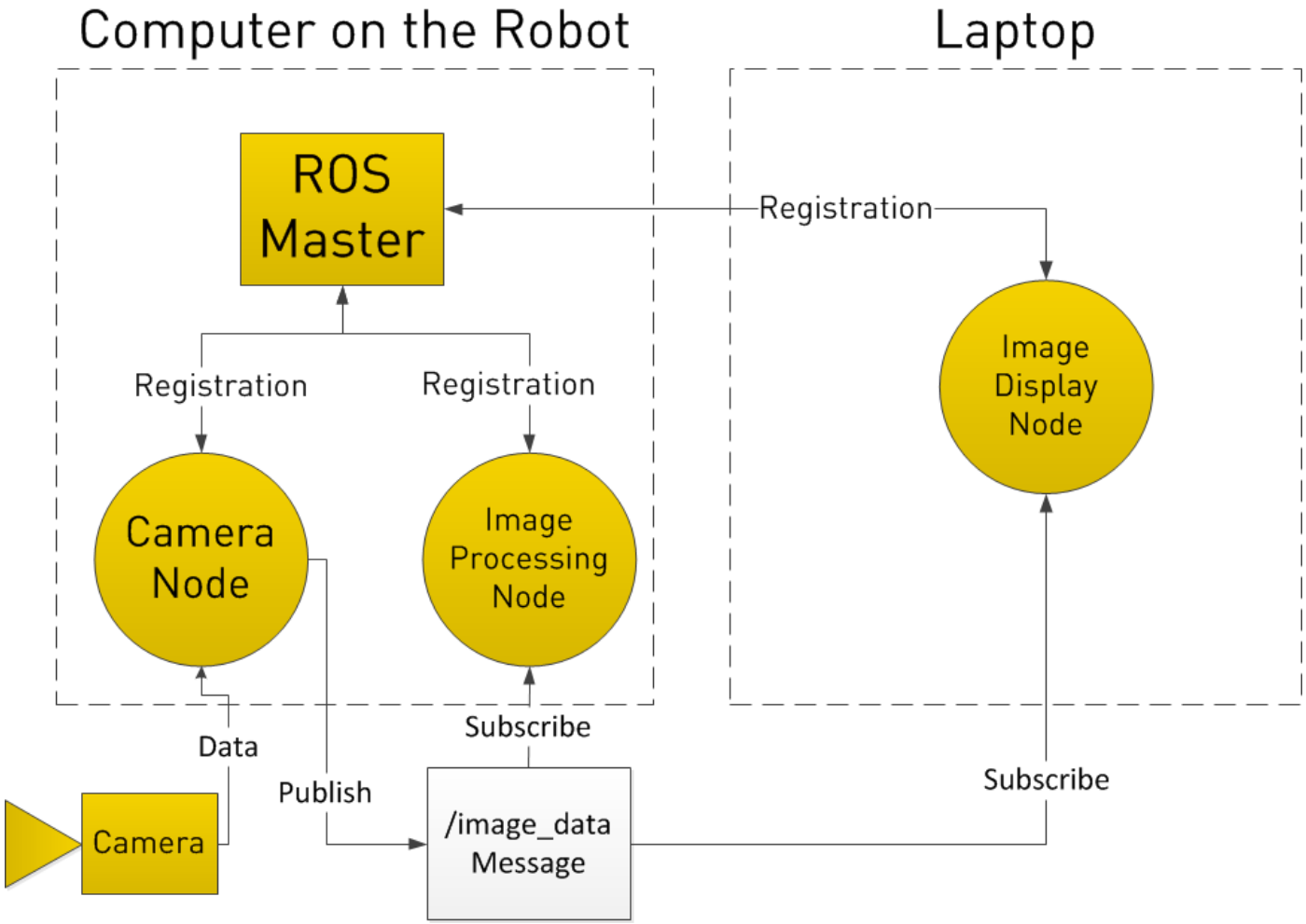
ROSパッケージの適用分野

- 知覚 (Perception)
- 物体認知 (Object Identification)
- セグメンテーションと認識 (Segmentation and recognition)
- 顔認識 (Face recognition)
- ジェスチャ認識 (Gesture recognition)
- 動作追跡 (Motion tracking)
- エゴモーション (Egomotion)
- 運動理解 (Motion understanding)
- 運動からの構造 (Structure from motion (SFM))
- ステレオビジョン:2台のカメラを通して、奥行き知覚
  - (Stereo vision: depth perception via two cameras)
- モーション (Motion)
- 移動ロボット工学 (Mobile robotics)
- 制御 (Control)
- 計画 (Planning)
- 把握 (Grasping)

※http://en.wikipedia.org/wiki/Robot_Operating_Systemから引用、和訳

ROS

■データの流れに応じて、コンポーネントをつなぐ



http://www.clearpathrobotics.com/blog/how-to-guide-ros-101/ より引用

ROS

■ROSエコ・システムのソフトウェアは3つのグループに分けられる

・(1) 言語やプラットフォームから独立したツール
　ーROSベースのソフトウェアを作ったり分散させるために使用
・(2) ROSクライアント・ライブラリ実装
　ーroscpp, rospy, roslispなど
・(3)アプリケーションに関連するコードを含んだパッケージ
　ーひとつ以上のROSクライアント・ライブラリが使用

※http://en.wikipedia.org/wiki/Robot_Operating_Systemから引用、和訳

RTミドルウェア

■産総研などが開発しているロボット用ミドルウェア
- RTコンポーネントは、OMGにて、国際標準化

■RT (Robot Technology/Robotic Technology)とは
- ロボット機能要素

■RTミドルウエアとは
- 様々な機能要素をモジュール化 (RTコンポーネント)
- RTCを、ソフトウエア的に統合するためのプラットフォーム

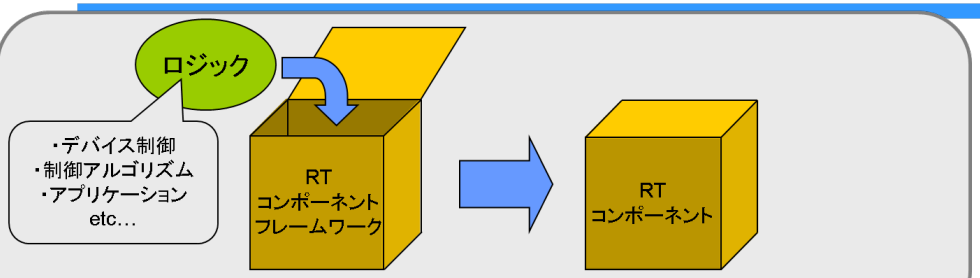
■RTミドルウエアの目的
- 仕様をオープンにする
- 様々な実装同士が相互接続できるようにする
- オープンアーキテクチャのプラットフォームを確立する

■通信はCORBAベース

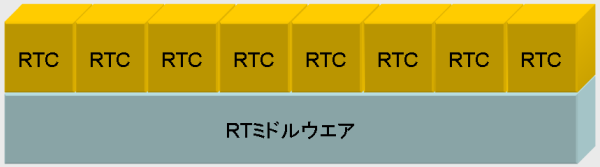
■「OpenRTM-aist」は、RTミドルウェアの産総研による実現
- ライセンスは、LGPLおよび産総研と個別に契約するライセンスのデュアルライセンス方式

画像引用元,参考 http://www.openrtm.org/openrtm/ja/node/161#toc5
http://www.openrtm.org/openrtm/ja/content/openrtm-aist-official-website

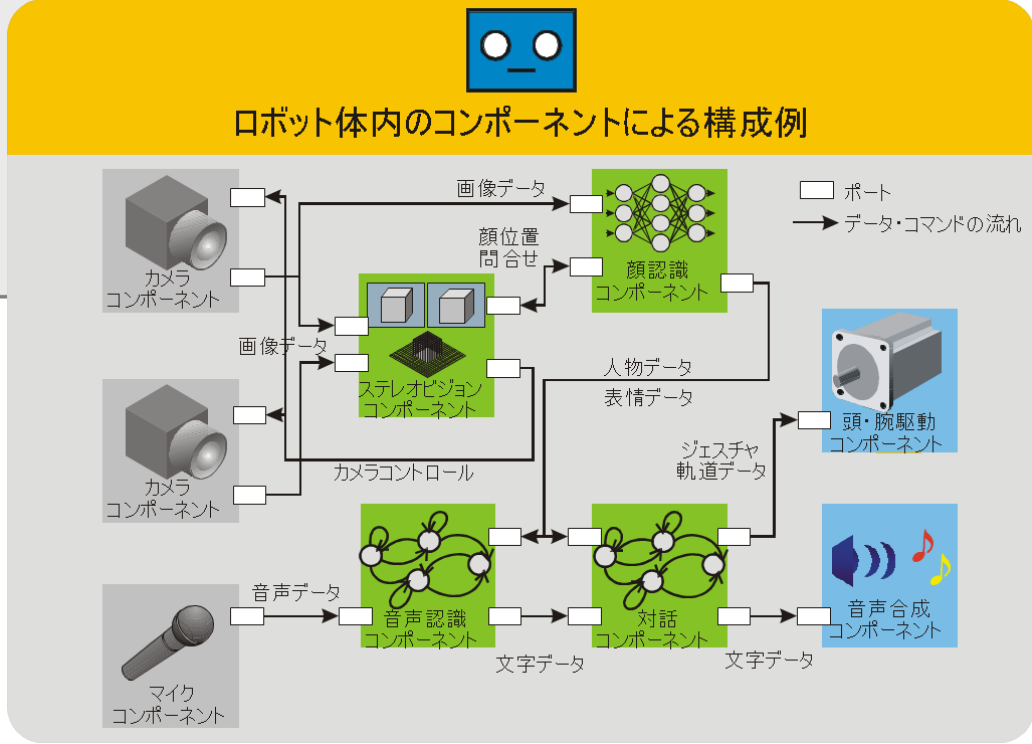
RTミドルウエア



ロジックを箱（フレームワーク）に入れたもの＝RTコンポーネント（RTC）

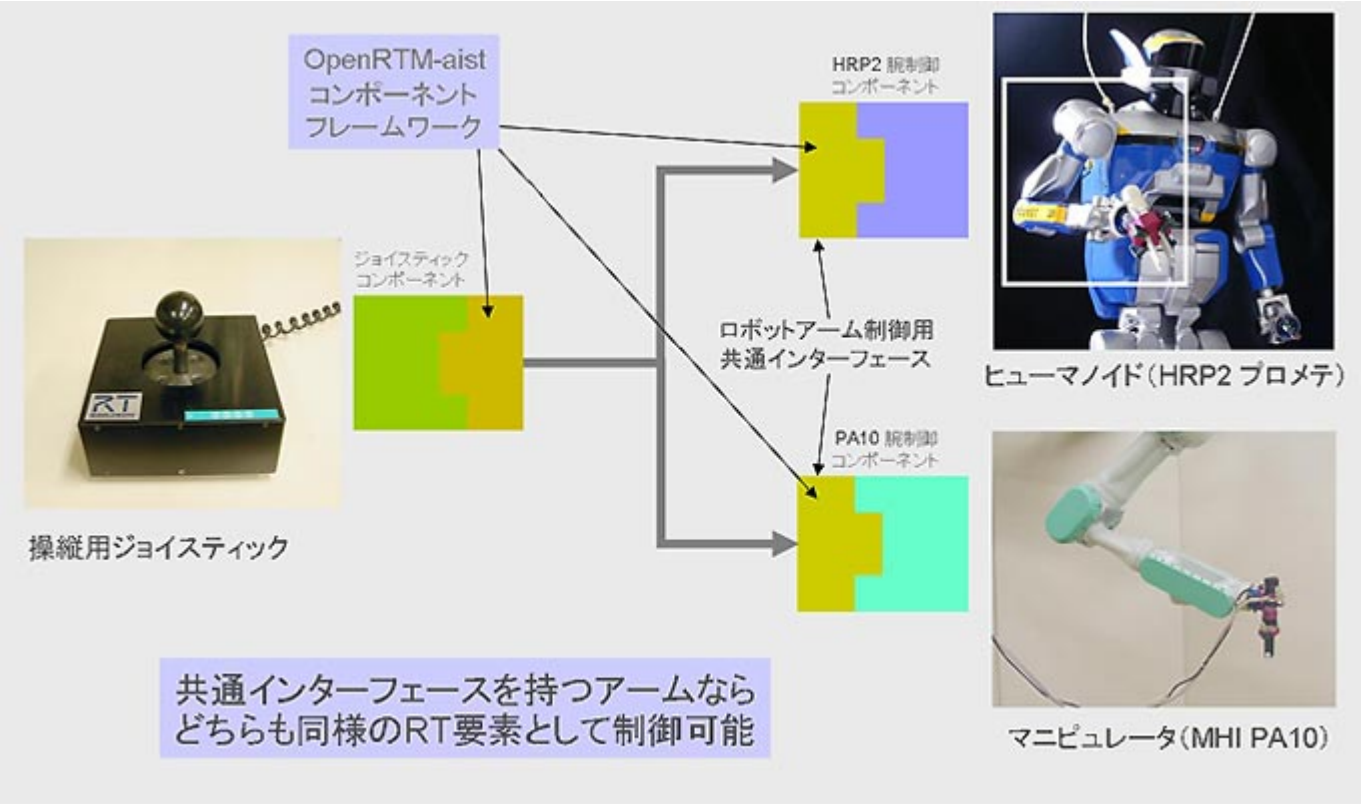


RTCの実行環境（OSのようなもの）＝RTミドルウエア（RTM）
※RTCはネットワーク上に分散可能



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 よりコピー

RTミドルウェア



引用元
http://www.aist.go.jp/aist_j/press_release/pr2005/pr20050224/pr20050224.html

RTC(RTコンポーネント) OMG標準

■ RTM / OpenRTM-aist は
- コンポーネントモデル
- そのインターフェー ス
- である「RTC」が OMG 国際標準

■ OMG (Object Management Group)
- 1989年に設立されたソフトウエア標準化団体
- CORBA (Common Object Request Broker Architecture)分散オブジェクトミドルウエア:
- UML (Unified Modeling Language) ソフトウェアモデリング言語
- などを策定･管理している組織

■ RTCのインターフェース仕様
- OMG において, 産総研と米国ミドルウエアベンダ RTI (Real Time Innovations) により標準化
- RTC (Robotic Technology Component) Specification (http://www.omg.org/spec/RTC/1.0/) として2008年4月に公式リリース

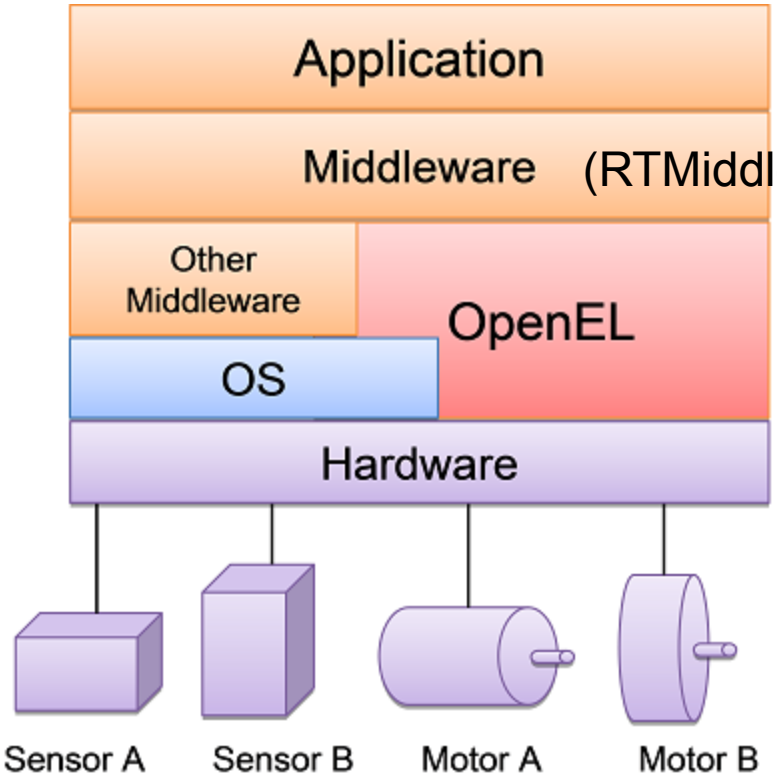
http://www.openrtm.org/openrtm/ja/content/rt%E3%83%9F%E3%83%89%E3%83%AB%E3%82%A6%E3%82%A8%E3%82%A2%E3%81%A8%E3%81%AF%EF%BC%9F-0#toc4 より引き写し

OpenEL

■JASA,産総研などが推進しているフレームワーク
■国際規格にするべく、OMGに提案中
■RTミドルウェアなどのコンポーネントの可搬性を高める
■ハードウェアに近い層を、抽象化
■OpenEL準拠で書かれたソフトウェア(RTC)は、
■ハードウェア・ドライバに依存せずに動作する

参考
http://jasa.or.jp/openel/Main_Page/ja#OpenEL.E3.81.AE.E6.AD.B4.E5.8F.B2.E3.81.A8.E6.99.AE.E5.8F.8A.E3.83.BB.E5.95.93.E7.99.BA.E6.B4.BB.E5.8B.95

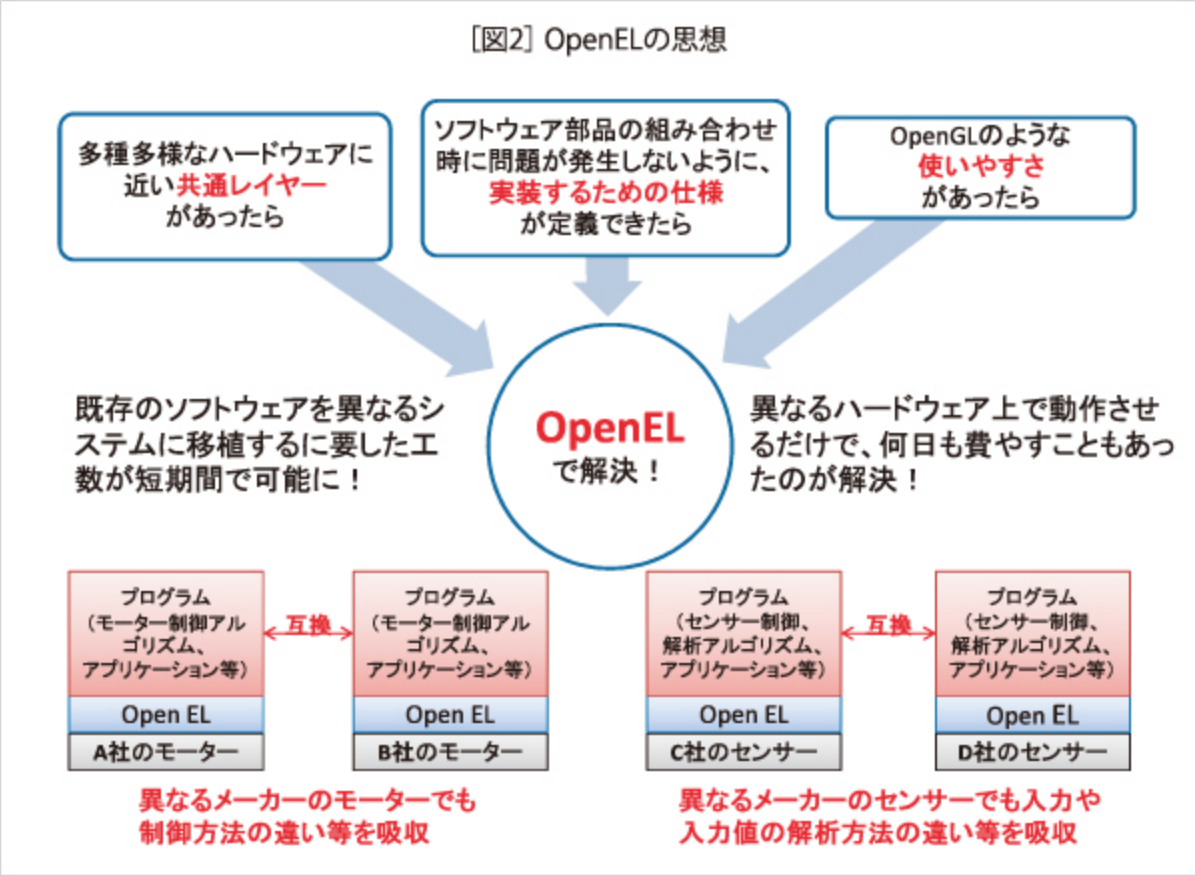
OpenEL



(RTMiddleware/ROSなど)

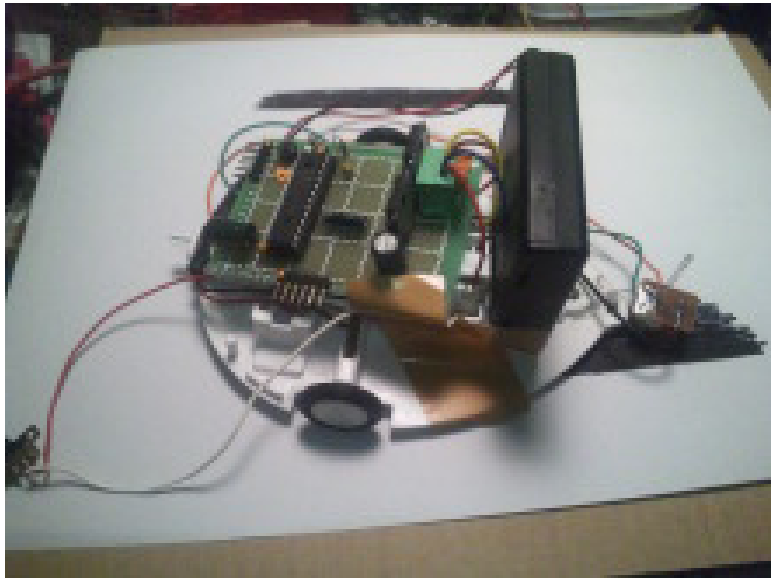
引用元 http://jasa.or.jp/openel/Main_Page/ja

OpenEL



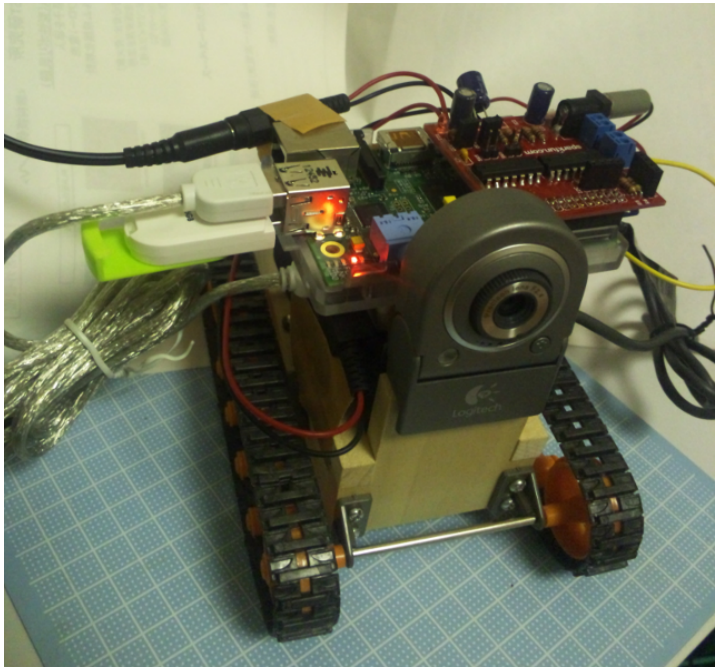
引用元 http://www.jasa.or.jp/TOP/download/bulletin-jasa/bulletin041-05.pdf

OpenEL使用 オレオレ実例



・ライントレーサ
マルチスレッドBASICインタープリタ
OS無し
OpenELでDCモータ駆動

PIC32MX250
(MIPSコア@50MHz、ROM128KB,
RAM32KB)
秋月で360円



・顔追尾ロボット
OpenCV under Linux
OpenELで DCモータ駆動

Raspi
(ARM11@700MHz, 512MBメモリ)

OSS品質評価
の
アプローチ

OSSって面倒くさい !?

- 品質問題
  - 適当に作られた無料のソフトウェアの品質は?
    - どこの誰だか、知らない人が作ったものでしょ…
    - タダのものがそんなに良ければ、プロは商売あがったりだよ
    - タダのものが、かなり良い、ということは、非常に頻繁にある
- 品質の押さえ方は、大事

OSSの品質評価の問題

- 開発手法で押さえられない
  - (導入したい)OSSは開発が終了しているから
- コードレビューにそぐわない
  - 巨大OSSだから、使用したい
    - 小さいソフトウェアなら、新たに書き下ろせる
  - 巨大OSSのコードをすべて精査するのか???
    - 通常のサイトには、その能力がない

OSSの品質評価の解決方法

- テストする
- 他者のソフトウェアを購入した場合
  - 製作者である他者が、品質を保証
    - テスト結果で、品質を示すことも多い
      - 必ずしもそうではないが…

- OSSもテストして、品質を確認

OSSのテスト方法の難点

- OSSもテストして、品質を確認したい…
- OSSのテストの難点
  - 仕様がいまひとつ明確でない
  - どこを重点的にテストすればいいか判らない
    - コードをレビューしていないから、
      難しそう&重要そうな部分が判らない
      ↓
  - 境界条件テストが難しい
  - 限界値テストは、可能だろう
    - 仕様が不明確な場合もあるが

OSSのひとつのテスト方法

- 境界条件テストが難しい
- 限界値テストは、可能だろう
  - 仕様が不明確な場合もあるが

    ↓
- いろんな引数を、対象OSSに食わせる
- 異常な振る舞いをしたら、危険
  - 危険な値が、入力されないように、外で処置

    or
  - OSSを修正

OSSのひとつのテスト方法:Fuzzテスト

- いろんな引数を、対象OSSに食わせる
- Fuzzテスト
  - ファズ（英：fuzz）（予測不可能な入力データ）を与えることで意図的に例外を発生させ、その例外の挙動を確認するという方法を用いる。ファズテストと呼ばれることもある。
  - https://ja.wikipedia.org/wiki/%E3%83%95%E3%82%A1%E3%82%B8%E3%83%B3%E3%82%B0 より引用

Fuzzテストを試行

- OSS活用WGで、fuzzテストを試行する
- 対象OSS候補
  - OpenCV :機械学習,画像処理
    - ニーズ高い
  - OpenRTM-aist (候補)
    - 規模が大きそうなので、やや躊躇している
  - OpenEL (候補)
    - JASAが中心となって策定中の規格
    - OSS版ソフトウェアを対象に行いたい

Fuzzテスト試行により

- 品質評価の基準を探る
  - Fuzzテストの結果がどの程度だと、安心だと言えるのか

    or
  - Fuzzテストの結果からは、安心だと言えないのか
- テスト・コストを探る
  - どれぐらいテストをすると
    - 問題が発現するのか
    - 十分にテストした、と言えるのか/言えないのか?

Fuzzテスト試行コストの測り方

- テスト・コストを探る
  - 「異常がすぐわかった」を 1として、
  - 他の異常が読み取れるまでのコストを、比で表す。

OpenCVにFuzzテスト試行した

- 機械学習（認識）に関する関数をテスト

- `void cvReleaseImage( IplImage** image )`
- `void cvReleaseHaarClassifierCascade( CvHaarClassifierCascade** cascade )`
- `int cvNamedWindow( const char* name, int flags=CV_WINDOW_AUTOSIZE )`
- `CvCapture* cvCreateFileCapture( const char* filename )`
- `CvCapture* cvCreateCameraCapture( int index )`
- `int cvSetCaptureProperty( CvCapture* capture, int property_id, double value )`
- `bool CascadeClassifier::load(const string& filename)`
- `CvMemStorage* cvCreateMemStorage( int block_size=0 )`
- `IplImage* cvQueryFrame( CvCapture* capture )`
- `void detectMultiScale( const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size())`
- `void cvRectangle( CvArr* img, CvPoint pt1, CvPoint pt2, CvScalar color, int thickness=1, int line_type=8, int shift=0 )`
- `void cvShowImage( const char* name, const CvArr* image )`
- `void cvReleaseMemStorage( CvMemStorage** storage )`
- `void cvReleaseCapture( CvCapture** capture )`
- `void cvDestroyWindow( const char* name )`

OpenCVにFuzzテスト試行した結果

大きな問題がありそうなもののみを、表に掲げる

関数	引数	値	結果（大きな問題がありそうなもののみ抜粋)
CvCapture* cvCreateFileCapture(const char* filename)	filename	NULL	Segmentation fault
IplImage* cvQueryFrame(CvCapture* capture)	capture	NULL	静止画使用時は黒い表示、カメラ使用時はNULLを返す
void detectMultiScale(const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size())	scaleFactor	inf	呼び出してから戻ってこない
void cvShowImage(const char* name, const CvArr* image)	name	存在しないウィンドウ名	(Hankyu+Face Detect:12853): GLib-GObject-CRITICAL **: g_object_unref: assertion 'G_IS_OBJECT (object)' failedを表示し、新たなウィンドウ生成
	image	NULL	真っ黒の表示。(Hankyu+Face Detect:16287): GLib-GObject-CRITICAL **: g_object_unref: assertion 'G_IS_OBJECT (object)' failed表示。

OpenCVにFuzzテスト試行した

- コストなどの分析は、これから
- OSS活用WGへのご参加を！
  - 一緒に、OSS品質評価問題に取り組みましょう！

2016年度実績:OSS 無料 IoTセミナー

OSSコンソーシアムCyber Physical Embedded部会 主催
JASA OSS活用WG 協力
·女子にも優しいIoTハンズ·オンセミナー
(「フレッシュマンのための IoT セミナー」)第1回
·AUG/06/2016 於:JASA会議室
·受講者：15名(満席)
·温度、湿度センサをRaspiに接続し、Webサーバへデータを送信
·女子にも優しい IoT Azureハンズ·オンセミナー
(「フレッシュマンのための IoT セミナー」)第2回
·共催 Microsoft and IoT あるじゃん
·AUG/23/2016 於: Microsoft品川本社
·受講者：約45名(満席)
·スピーカ: MS 太田さん
·Azureを使用した、エモーション分析
·Azure の使用: $25ぶんの使用料が1年間無料になるプランを使用

「ドローン&ロボット、機械学習OSSの紹介と、OSSの品質についてのアプローチ」

2016/11/16 発行

発行者　一般社団法人 組込みシステム技術協会
東京都中央区日本橋大伝馬町 6-7 住長第2ビル 3階
TEL: 03(5643)0211 FAX: 03(5643)0212
URL:http://www.jasa.or.jp/TOP/

一般社団法人 組込みシステム技術協会
Japan Embedded Systems Technology Association