2016年 SWEST18（セッションs2b）セミナー資料

「オープンソースを用いた ドローンの自律制御ソフトウェア技術」

ファイブソリューションズ（株） 礒部正幸

アジェンダ

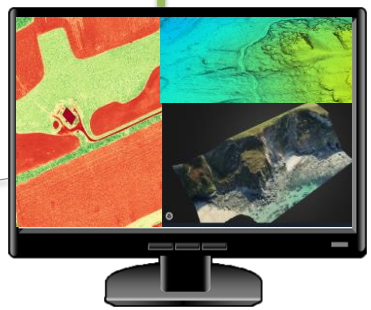
- ドローン関連ソフトウェアの種類
- ROSを始めとするOSS、およびドローンへの応用
- 開発している基盤ソフトのアーキテクチャ

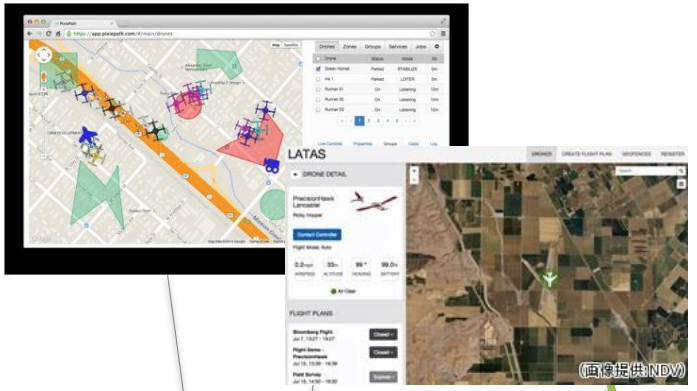
ドローンとソフトウェア

クラウド

②知能制御
（アドバンスコントロール）

③データ解析クラウド



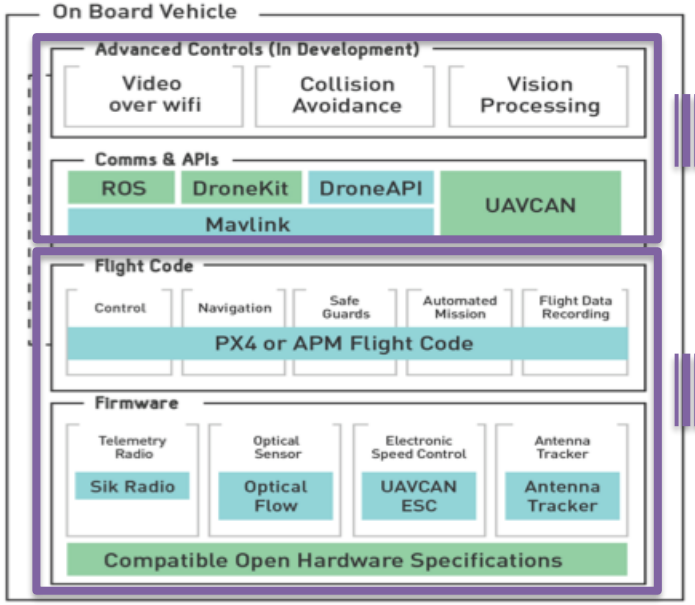


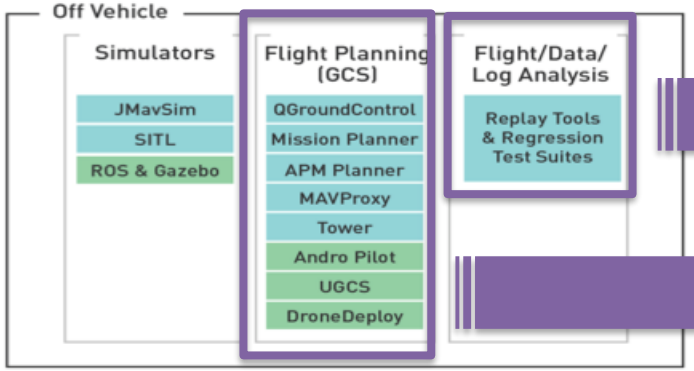
①飛行制御
（フライトコントロール）

④管制/モニタリング

（参考）Dronecode

- オープンソースコミュニティによるドローン向けソフトウェアスタック
- スタック全体で前スライドの①、②、③、④をカバー（順次実装されている）
- （後述する）ROSとの連携がサポートされており、AI部分のカスタマイズが可能





②知能制御
（ROS接続、および Dronecode独自のアドバンス制御）

①飛行制御
（対応センサーとファームウェア、
および 飛行制御アルゴリズム）

③データ解析クラウド
（データの転送と蓄積.
Dronecodeはこの部分のコミットメントは限定的.）

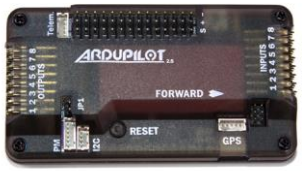
④管制/モニタリング
（飛行計画、遠隔制御、etc）

①飛行制御（フライトコントロール）

(1) Dronecodeを組込んだ単体の完成品FC

pixmini

(3DR製)pixhawk



ardupilot

・GPS,加速度,ジャイロといった制御に必要なセンサー類が内蔵されている。
・制御ソフトはインストール済み。
I/Oピンがわかりやすく引き出されており、コネクタで機体につなぐだけで飛行制御が可能。
・制御パラメータはI/Oピンへの電圧で入力。

(2) Dronecodeを組み込んだFC拡張ボード

(ラズパイ用拡張ボード)NAVIO+

・拡張ボードには制御に必要なセンサー類が搭載されており、マザーボードと機体それぞれにI/Oピンで接続。
・FCのソフト自体はCPU(マザーボード)にインストール。
・制御パラメータはFCソフトの設定 or I/Oピンへの電圧入力。

(3) ドローンメーカ製の（クローズドソース）FC

(DJI製)A2

(Airware製)FlightCore

・制御に必要なセンサー類、制御ソフト、機体と接続するコネクタ等が全て組み込まれ完成したモジュールとなっている。
・制御パラメータなどはあまりいじれず、対応機体も同メーカのものに限られる。

②知能制御（アドバンスコントロール）

(1) Dronecodeコミュニティによるアドバンスコントロールソフト
→ スポンサーもついて開発中のようだがまだ使えるものは出てきてない状況
→ クアルコムやインテルはドローン向けのプラットフォームを提供し始めている。

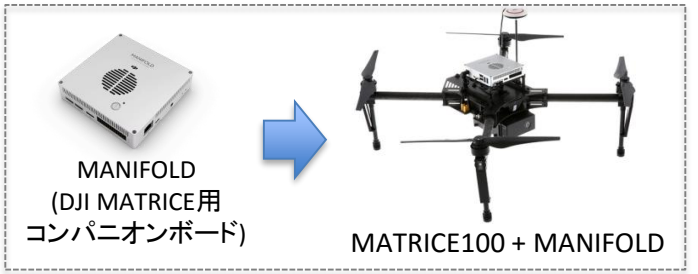
クアルコム
ドローン向けSnapdragon

(2) Dronecode + ROS
→ MAVLinkというプロトコルが実装されており、FC、ROS、
リモコン側とデータや制御コマンドをやり取りできる。
→ ROS側はMAVLink用のパッケージがOSSであり、ROSでドローンのロボティクスパッケージを
用意すれば知能制御が可能。
（画像認識、3次元環境認識、衝突回避、SLAMや自律ナビゲーションなど. 詳細は後述）

MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

(3) メーカ製ドローン + SDK
→ 最新のPhantom4では衝突回避機能、
人の自動追跡機能などを内蔵。
リモコン側からSDK越しにも利用可能。
→ カスタムソフトウェア用のプラットフォーム
「DJI MATRICE」ではNVIDIA GPUを搭載した
コンパニオンボードがオプションであり、
計算負荷のかかる画像認識処理等をオンボードで行える。

DJI Phantom4



(4) クラウドロボティクス

Rapyuta Robotics

(5) 単機能ソフトウェア部品

AUGMENTED PIXELS（オンボードSLAM）

(6) 業務用統合プラットフォーム

Airware

③データ解析クラウド

(1) 3Dモデル生成、分析、測量、検索と共有
→ 航空写真のような連結画像、正確性重視の3D地形再現、
テクスチャ込みの3D再現、農業用解析（穀物の生育状況の分析）、
土砂や構造物の体積の計算、既存のCADデータとの照合、
Webの共有リンク生成

DroneDeploy

DATAMAPPER

(2) ドローンのデータ解析用のクラウドインフラサービス（IaaS/PaaS）
→ 写真の管理、画像認識用のGPUサーバ、
画像ピクセル処理用のサーバとSDK、
クラウド上のデータに簡単にアクセスするためのモバイルGUI
さらに機能単位でソフトを売り買いできるマーケットプレイスを提供

DRONEDATA

(3) 飛行制御とクラウドデータ解析が一体となったプラットフォーム

Rapyuta Robotics

Airware

PRECISIONHAWK

(4) 大手のパブリッククラウドプラットフォームでカスタム開発
→ AWS, GCP, Azure 等が提供するIoT向け機能を組合せることでドローン向けサービスを構築可能

・データ解析クラウドについては、有力なオープンソースがあまり無く、商用のサービスが乱立しつつある。
・いわゆる「IoTクラウドプラットフォーム」と同様、ビジネス活用の側で探索が必要なため、確立したソフトウェア要件がまだ少ない。

④管制/モニタリング

（1）パブリックな飛行管制・交通整理
→ 農業、消防、インフラ点検など公共空域の飛行管制
→ 策定した飛行ガイドラインに沿って自律ドローンと通信しルールを施行
→ 商用というより行政支援アプリケーション(NASA)

（2）ローカルなドローン複数台のモニタリングと制御
→ 複数台の協調制御、ミッション管理、ジオフェンス機能、バッテリーモニタ等
→ 主にドローンのハードを含む統合プラットフォームベンダが提供

（3）機体の登録管理・行政への許認可申請代行
→ 国内ではブルーイノベーション社のSoraPass、CLUE社のDroneCloudなど

・OSS（Dronecode）では飛行計画やミッション管理の機能を搭載した管制ソフトが幾つか提供されており、それらは機体とMAVLinkプロトコルを介して通信する（コンパチブルGCS : Compatible Ground Control Stations）。
・MAVLinkをサポートした遠隔管理ソフトを独自に作ることにより、Dronecodeベースの機体の管制/モニタリング機能を実装できる。

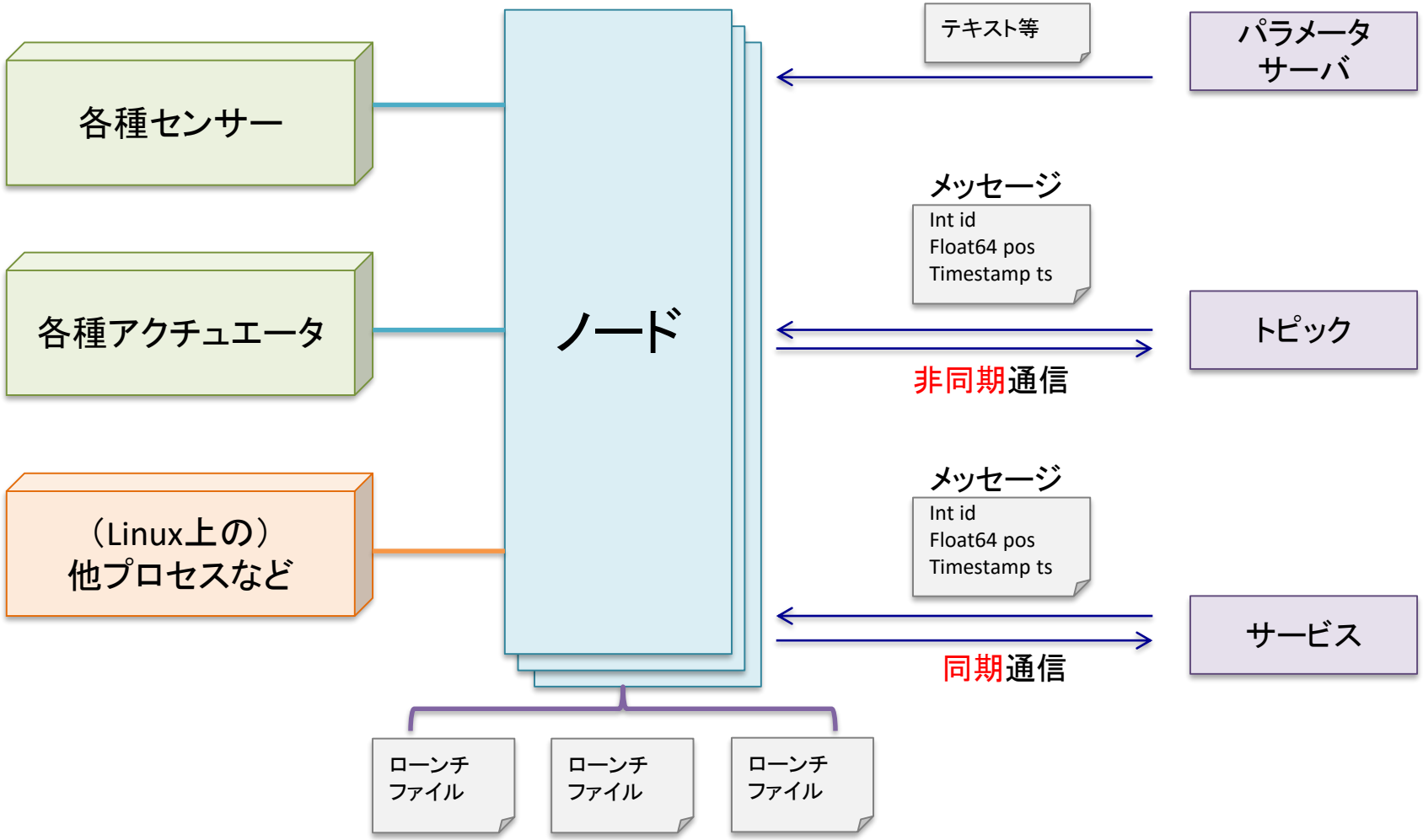
（参考）機体ハードウェアの開発について

- Dronecodeベースのドローンを自作するには、市販のパーツやモジュール（筐体、プロペラ、電源周り）を調達しFCとつなぐだけで可能。
- ただし部品選定とインテグレーションの設計は一定のノウハウを要する。（特に重量と電源のキャパシティ見積りで問題がよく生じうる）
- （本セミナーはソフト中心の内容のため、機体ハードの）詳細は個別にディスカッションしましょう！

アジェンダ

- ドローン関連ソフトウェアの種類
- ROSを始めとするOSS、およびドローンへの応用
- 開発している基盤ソフトのアーキテクチャ

ROSの概要（1）



・複数のノードがトピック・サービスを介してメッセージをやり取りしながらセンサー情報の加工、状態認識、判断・意思決定、行動計画、アクチュエーション等を行うことが出来る。
・立ち上げるノードの構成をローンチファイルとして記述して効率よく操作できる

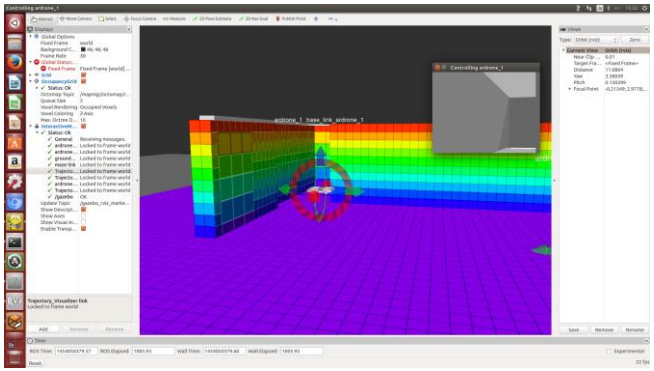
ROSの概要（2）

開発を効率化する便利な機能群

rosbag機能

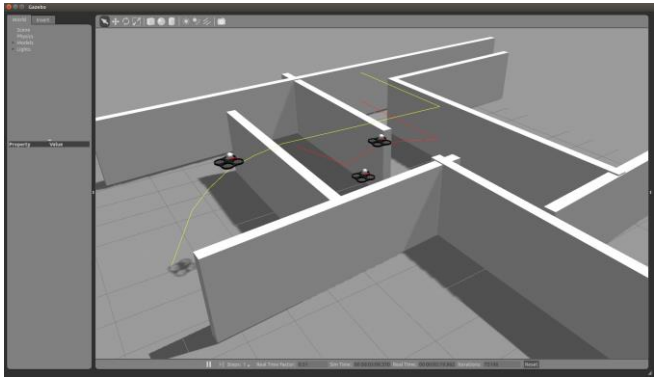
トピックに流れるメッセージの
記録・再生ツール

rviz



トピックに流れるメッセージの
可視化ツール

Gazebo



ロボット実機（センサー・アクチュエータ）
の代わりに使えるROSとシームレスに
つながるシミュレータ（物理エンジン+
3Dレンダリング）

・ほかにrqt（カメラ映像の）

ドローンに有用なROSパッケージ

ardron_autonomy

Parrot ARDroneをROSからコントロールするパッケージ

・センサー情報（カメラ、IMU、高度、GPSなど）
・制御コマンド（離着陸、左右移動、左右旋回、上下移動）

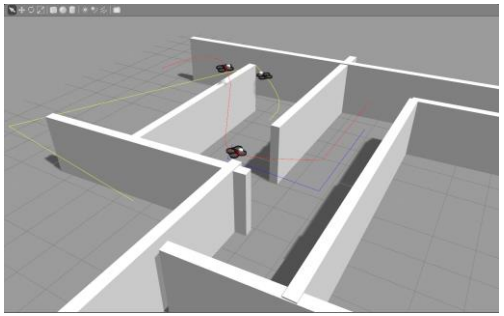
bebop_autonomy

Parrot BebopDroneをROSからコントロールするパッケージ

・センサー情報（カメラ、IMU、高度、GPSなど）
・制御コマンド（離着陸、左右移動、左右旋回、上下移動）

tum_simulator

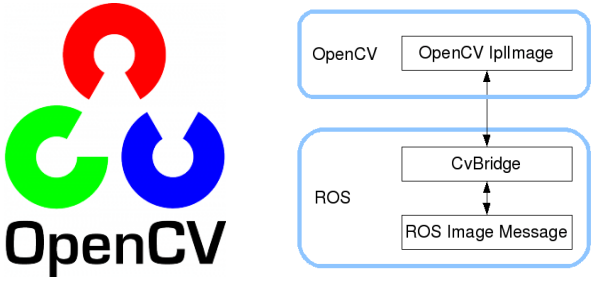
マルチコプターを想定したGazebo上のモデルプラグイン



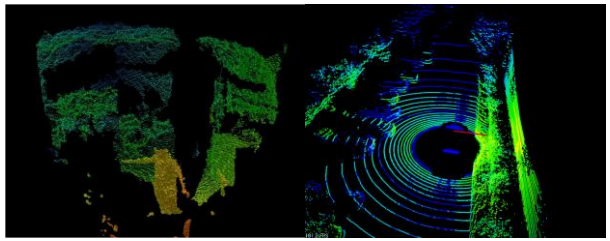
・センサー情報（カメラ、IMU、高度、GPSなど）
・制御コマンド（離着陸、左右移動、左右旋回、上下移動）

ドローン+ROSで有用なOSSライブラリ

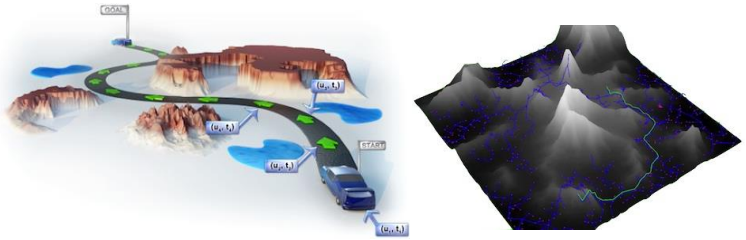
OpenCV 画像処理・画像認識



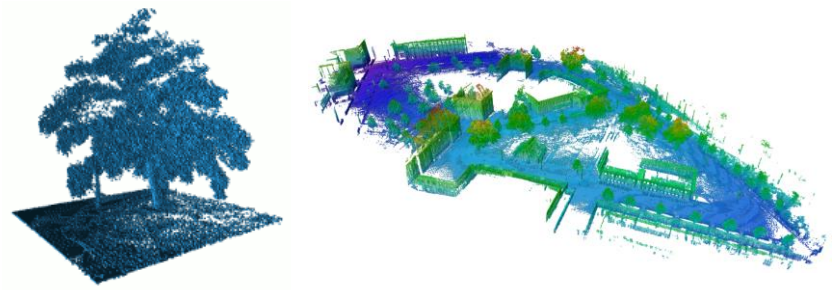
PCL 点群処理



OMPL 動作計画

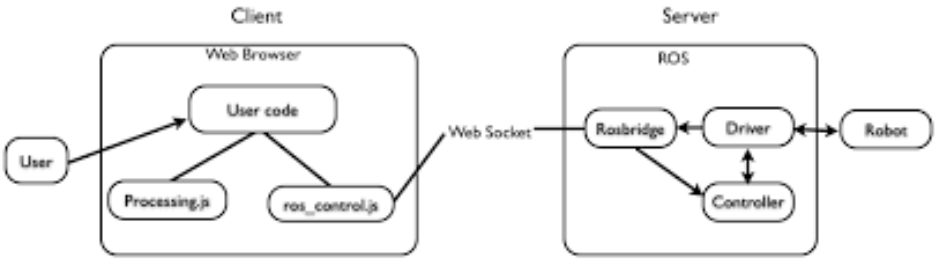


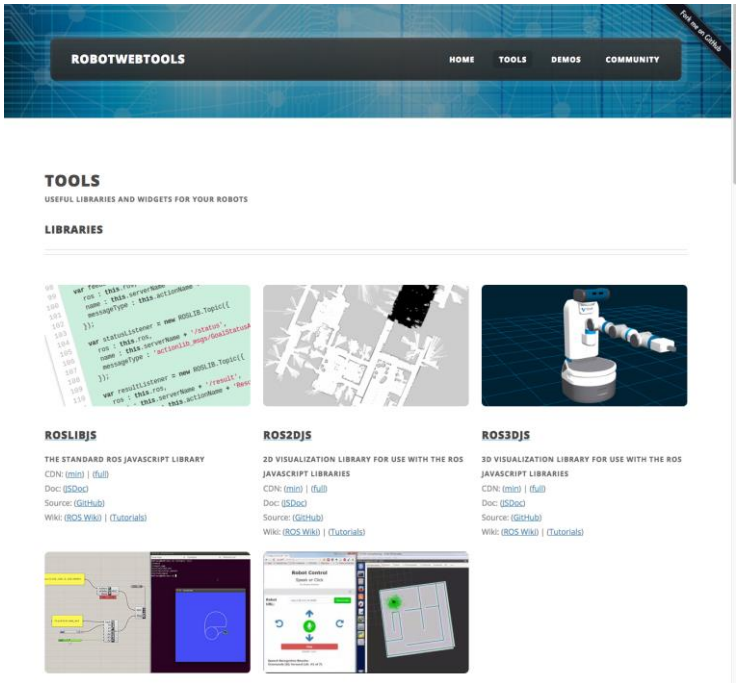
Octomap コンパクトな3次元格子表現



FCL 柔軟な衝突回避ライブラリ

ROSとWebの接続（RobotWebToolsについて）





rosbridge

・ROSメッセージをJSONにしてJavascriptとやりとりできるようにするサーバ。Websocketでブラウザ上のJavascriptと連携したり、TCPソケットで別ホストのNode.jsプロセスと連携したりできる。
・画像/映像データについては mjpeg_server で配信可能。（ただしそれなりにサーバ負荷が掛かる）

roslib

・rosbridgeサーバに接続するクライアント側のJSライブラリ。ブラウザ上のJS用のものとNode.js用ものとがある。
・roslibとHTML5 CanvasやWebGLを組み合わせ 2dや3dの位置情報をブラウザ上で可視化するライブラリがある。

ROSベースのシステムにWebインタフェースを備えることも比較的容易

ROSでドローンの自律制御ソフトを作る上での課題（2016年初頭時点）

重要度	内容	対処方法（例）
高	2次元平面上のロボットに存在するROSのナビゲーションスタックのようなパッケージ群が3次元のUAV向けには存在せず、作り込みがどうしても必要	SLAMや地図サーバやパスプランナなどの機能ごとに存在するライブラリを統合する
中	DronecodeとROSで開発コミュニティが割と独立していて連携が通信プロトコルくらいしかなく、オンボードとリモートで透過的に機能実装することが困難	MAVLink連携で整合的に動くソフトを開発してコントリビュートするとか
中	オンボードでROSを動作させて知能制御を行うにはまだ計算負荷が高い	ゲートウェイやクラウド側と連携させるなど
中	ドローンのメーカごとにSDKやAPIが異なっており高レイヤのロジックを共通化しようしても低レイヤの統一が難しい	典型的な機体スペックと特殊用途向けの機体スペックとでメッセージ構造を体系的に定義して吸収
低	ROSメッセージのデータフォーマットがJSONや既存のDBなどと互換性がなく、閉じたシステムになりがち	RobotWebTools（後述）を用いたり、rosbag機能を使ったバッチ処理でデータ変換

アジェンダ

- ドローン関連ソフトウェアの種類
- ROSを始めとするOSS、およびドローンへの応用
- 開発している基盤ソフトのアーキテクチャ

製品コンセプト

製品メリット
・複雑なドローンの操縦に掛かるオペレータの負担を軽減（コスト削減メリット）
・より大きなIoT統合システムに拡張可能で投資効率が良い（戦略投資メリット）

製品に課される要件
・ドローンを使った様々なサービスで求められるドローンの操縦を自動で行う
・異なるドローンの機種（ハードウェア構成）、ドローン複数台を同時に扱える
・ドローンと他のIoTデバイスを同列に扱える柔軟性を備える

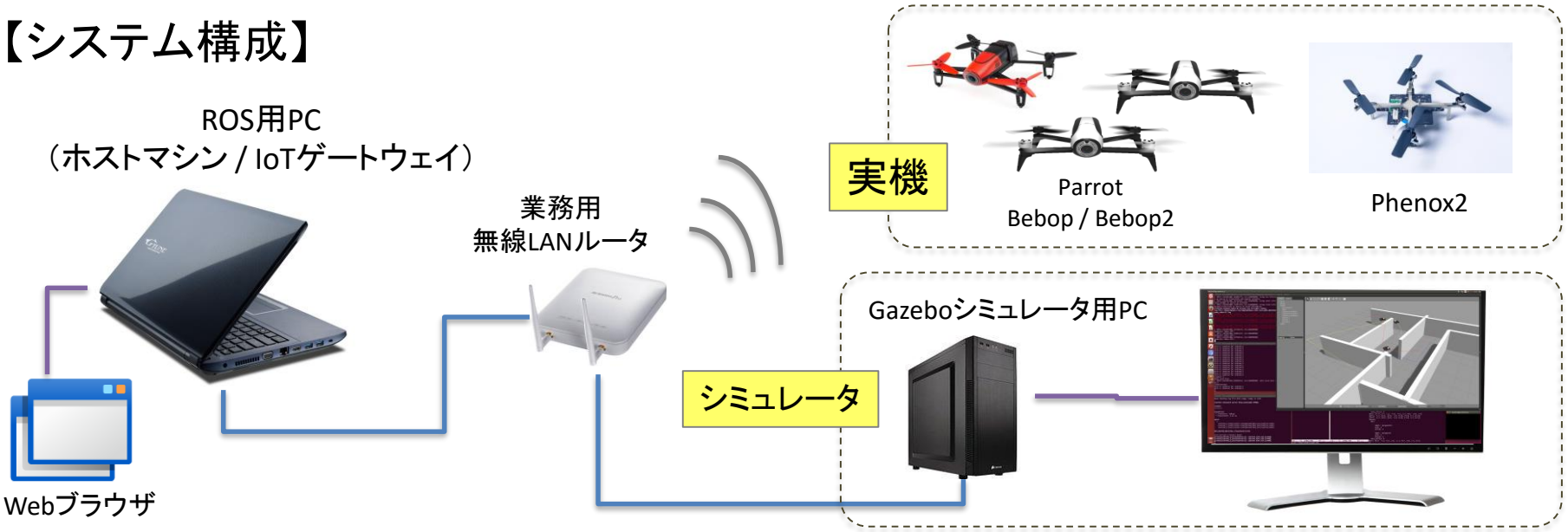
2014/11頃 着手。
その時点で可能な構成を検討した結果採用した方針：

・FCは自作しない（市販ドローンの機体 or Dronecodeベースのカスタム機体）
・知能制御と管制/モニタリングに特化し、クラウド側のデータ解析は後回し
（ただしクラウド上のデータ解析が容易に行える拡張性を意識して作る）
・まずはゲートウェイからリモートでの知能制御を行い、オンボード処理は後回し
（ただしオンボードとゲートウェイの間でなるべく移植しやすいように作る）
・ドローンだけでなくIoTデバイスも含めてシステム統合可能な設計にする
・入手性の高いParrot社のドローン（ARDRone）でまず実装。（※）

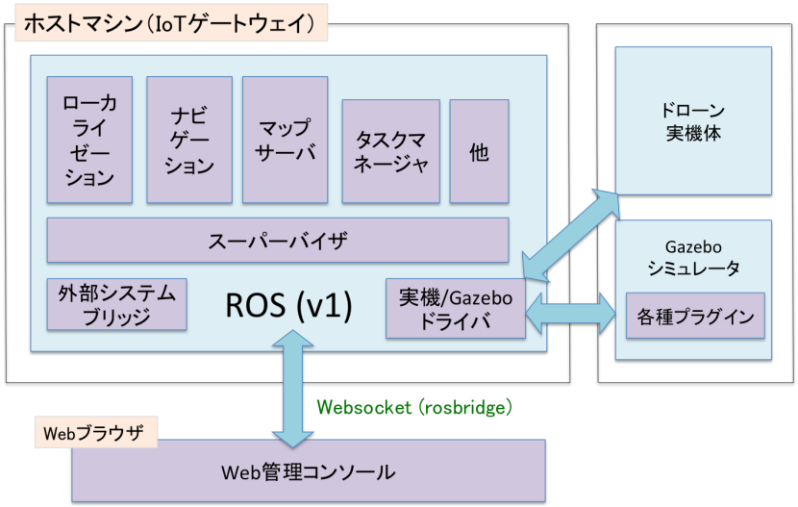
現在、Bebop, Bebop2, とフェノクス2に対応済み。

システム構成及びソフトウェアアーキテクチャ

【システム構成】



【ソフトウェアアーキテクチャ】



- ホストマシンにてROSが稼働
- 既存及び独自開発のROSノードを組合せて知能制御及び管制/モニタリング機能を実現
- 実機（異なる機種）/Gazeboシミュレータで機種依存性を吸収するドライバを作成
- Gazeboシミュレータではセンサアクチュエータ、環境オブジェクト等のプラグインを作り本体の機能検証を効率化
- ROSに接続するWeb管理コンソールを提供

サブシステム一覧

サブシステム名	役割
スーパバイザ	本基盤ソフトのメインノード。機体情報をDBで管理し、ドローンに名前をつけてROS上の名前空間上で機体やデバイスの個体名の一位性を保証する。複数台の協調タスク（ミッション）を担い、内部的にミッション管理とリソーススケジューリングの機能を持つ。またWeb管理コンソールへのファサードとなる。
タスクマネージャ	SVが管理するミッションの中で、ドローン1台ごとのタスクを管理する。ローカライゼーション、ナビゲーション、マップサーバ、ドライバなどと通信しイベント処理やタイミングの制御を行ってタスクを遂行することに責任を持つ。
ローカライゼーション SLAM	ドライバに接続し、センサー情報を統合して当該ドローンの自己位置と姿勢を推定する。SLAMモードではGPSがない状態で環境上の画像特徴点から自己位置推定を行う。
ナビゲーション	マップサーバから環境情報をもらいタスクマネージャが指定する目標地点までの飛行計画を行い、ローカライゼーションから現在位置情報を受け取りながらドライバにコマンドを送り目的地までの飛行に責任を持つ。
マップサーバ	Octomapによる3次元マップと、その上のランドマーク情報（SLAMの特徴点やマーカー）を管理する。これにより作成済みのマップを読み込んだり、獲得した環境情報をドローン複数台で共有したりといったことが可能になる。
論理ドライバ	画像のフィルタなどセンサーデータの前処理やデバイスとのコネクションのハートビートなど、より上位層で共通となる処理でデバイスに近い部分を行う。
物理ドライバ	ドローンの異なるメーカや異機種の間の差異を吸収し、上位層で共通のメッセージでのやりとりを可能にする。またドローン以外のデバイスもここで扱えるようにする。
Web管理コンソール	ROS側にroslibで接続し、Webブラウザ上から簡単に基盤ソフトの機能を利用できるようにする。
Gazeboシミュレータ	ドローンのセンサーやアクチュエータの構成を模したプラグインをいくつか用意し、基盤ソフトの実装を変えずに異なる機体ハードのロジック検証を可能にしている。

ソースツリーのルートディレクトリ

demo	added navigation demo file
drivers	updated bebop driver
localization	localization default fps changed from 5 to 20
mapmgr	merged with master
navigation	Merge remote-tracking branch 'origin/loc_raw_exhibition' into exhibit...
ros_drone_msgs	add localization
simulator	added wind plugin, and tuned nozzle plugin
speech	some fix
supervisor	added phenox driver and phenox configuration, and ardrone configuration
taskmgr	scheduler now works and subsystem status specification changed
tools	doing real drone verification
webcon	adapted to offline
.gitignore	1st verify
.gitmodules	added bebop driver
CMakeLists.txt	first commit
README.md	revised readme

トピック例：スーパーバイザ

About topics

/supervisor/Device/Status

- Contains information of all devices.
  - All devices that are managed in drone brain are called "devices"
  - Topic configuration, device status can all be obtained by this topic

/supervisor/Subsystem/Status

- Contains information of all subsystems.
  - for example, you can know whether localization is enabled.

/supervisor/Task/Status

- Feedback information of operating tasks

/supervisor/Task/Results

- Results of the tasks that are finished

About services

/supervisor/Device/Status

- Returns the status of devices when requested. The contents are the same as the message in /supervisor/Device/Status
  - 'Names' is an array of display names. Enter the display names of the devices you want to know about.
  - if 'Names' is empty, it will return all the status of the devices registered.

/supervisor/Device/Register

- Used to register and unregister devices

/supervisor/Subsystem/Status

- Return the status of subsystems when requested. The contents are the same as the message in /supervisor/Subsystem/Status
  - for example, you can know whether localization is enabled.

/supervisor/Subsystem/Control

- used to control each subsystems.
  - used to enable localization
  - used to enable manual control
  - used to register maps in map server

/supervisor/Task/Register

- used to register and unregister tasks.
  - the details of the task should also be sended
- use the task id that can be obtained from the /supervisor/Task/Status Topic to delete tasks.

トピック例：ローカライゼーション

Topic interface (for other subsystems)

external I/F

- `/localization/users` paramserver_ex.FlagChange
  - FlagChange.target should be formatted like 'ModelName/drone_name' (ex: 'BebopDrone2/bebop_1')

control input

- `/(drone_name)/cmd_vel` control input

sensor input (deterministic feedback)

- `/(drone_name)/compass` Compass

sensor input (probabilistic feedback)

- `/(drone_name)/distance` Distance sensor
- `/(drone_name)/fix` GPS
- `/(drone_name)/fix` Altitude

estimated pose output

- `/localization/(drone_name)/pose` PoseWithCovarianceStamped
- `/localization/(drone_name)/debug` PoseStamped (for debug on rviz)

スーパバイザ

仕様と実装

・rospyで実装
・ドローン毎に、機種と機体名をwebコンソールから受け取る
・登録した機体情報はバークレーDBで永続化
・サブシステムのROSノードプロセス監視
・Pythonスレッドで複数台の協調タスクミッションを実装
・空いてるドローンの自動割り当て: リソーススケジューリング機能
（タスクをグラフ構造化してマルチエージェント巡回セールスマン問題を解く）
・機体名から映像トピックやセンサートピックなどの「トピック名」を引く機能

設計実装上の課題とその対処

・ROSには何らかの実体の一意性を名前空間上で保証する仕組みが存在しない
物理デバイスに名前をつけそれを名前空間に対応づける機能がどうしても必要となり
スーパーバイザというレイヤを作った。
・launchファイルで立ち上げたROSノードをシャットダウンする処理がROSはあまりケア
してくれない → ターミナルからlaunchファイルを実行した場合と同じことを実行し
SIGTERMをターミナルに送ることでシャットダウン時の一貫性を持たせるようにした

タスクマネージャ

仕様と実装

・rospyで実装
・スーパーバイザから支持されるタスク種別ごとに、イベントの種類とハンドラのセットを用意し、タスクリクエストがくると当該ドローンに対してそのセットを登録する。
・例えば、ナビゲーションのゴール到着イベントや、ナビゲーション失敗イベント、ローカライゼーションで自己位置推定に失敗したイベント、タイマーイベント、それからユーザからのキャンセルイベントなどがあり、それらのハンドリングとしてタスクの遂行を実装している。

設計実装上の課題とその対処

・ドローンをユーザの希望通りに動かすには、ゴール地点の指定や、何らかの制御ポリシー（例えば、あの人を追いかけながら撮影する、とか）や、不測の事態への対応などといった様々な状況に対応する必要があり、統一的に扱う方法が必要
→ イベント駆動方式

ローカライゼーション

仕様と実装

・roscppで実装。ドローン1台毎に1ノード立ち上がる。Nodeletで高速化。
・粒子フィルタと無香カルマンフィルタのどちらかをlaunchファイルで選択。
・ドローンに搭載されるセンサーの種類毎に尤度計算のコードがあり、
尤度を独立とみなしてベイズフィルタ部分で掛け算することでセンサ融合。
（高度センサのように値の信頼性が高いものはそのまま状態ベクトルに
上書きできるようにもした）
・センサーの構成は、ドローンの機種毎にlaunchファイルを記述することで定義
・推定結果は自己位置姿勢（6DoF）とその誤差楕円

設計実装上の課題とその対処

・「確率ロボティクス」を参考にベイズフィルタに基づくローカライゼーションを採用したが
ドローン毎に非線形のベイズフィルタを計算せねばならず重い。
→ 対処については後述
・ゲートウェイ方式で作っているため、実機からの画像遅延や秒間フレームレート（fps）
に限界がある（頑張って20fpsとか）
→ 対処については後述

ナビゲーション

仕様と実装

・roscppで実装。
・タスクマネージャからゴール地点の情報を、マップサーバから環境情報を受け取る
・OMPLのクラスを継承してパスプランニング、FCLで衝突回避
（ドローン同士の衝突回避は本基盤ソフトに登録されているドローン同士に限定）
・簡易的なPID制御でドローンを操縦するコマンドを算出してメッセージを物理ドライバへ
・ローカライゼーションで誤差楕円が大きくなりすぎた（迷子）場合、待機したり
ナビゲーションを失敗させてタスクマネージャに処理を委ねる
・ゴール地点に到達したら成功メッセージをタスクマネージャへ

設計実装上の課題とその対処

・当初は既存のライブラリ（MoveIt!）を利用することを試みたが、MoveIt!は多関節ロボットには適しているものの空間を自由に動ける
→ OMPLを採用し、カスタムコードを追加することで実装
・マップサーバ側がoctomapでデータを持ち、ナビゲーションがパスプランや衝突回避にメッシュ情報を用いるためデータ変換が必要
→ マップサーバ側でメッシュ化する機能を実装
・デッドロック対応。複数台のドローンが「お見合い」をしてしまうケースがある。
→ 一般的な解決策まで至らず、典型的なケースを個別に実装。

マップサーバ

仕様と実装

・roscppで実装
・基本情報は3次元の占有格子。Octomapライブラリを利用。
・3dマップの読み書き機能。2次元平面濃淡画像(2.5次元)の読み書き機能。
・ランドマーク（特徴点やマーカ）の登録・削除・検索機能も実装（PCLのkd-treeを使用）
・指定された直方体領域から切り出す機能

設計実装上の課題とその対処

・様々なサブシステムから共有して利用するため、基本情報を何にするかは重要
→ 当面必要な機能を満たせそうな3次元の占有格子を採用。
（人認識機能と組み合わせて環境に人がいることを認識したい場合には不適合
でありそうなように、これで全てをカバーできるわけではない）

（物理・論理）ドライバ

仕様と実装

・機種毎にroscppやrospyで実装
・ドローンの機種毎に実機とやりとりする方法が異なり、それを共通化する（物理ドライバ）
・また、画像データに対するフィルタやオプティカルフロー計算や特徴点の計算といった、上位のサブシステムから共通に必要となる前処理を行う。（論理ドライバ）

設計実装上の課題とその対処

・物理ドライバでは、映像遅延が生じないよう通信の仕方やバッファリングなI/O待ちをケアする必要があり、機種毎にそうした工夫を個別に行っている。
・機種Aでは1つのセンサであるものが、機種Bでは2つのセンサで実現されるもの（GPS＋高度センサ）の場合、全部そろってるAのほうに合わせる。その場合、機種Bはセンサー情報が到着するタイミングが非同期のため物理ドライバのノードの中でバッファリングして同期を取り共通のROSメッセージに詰め替えてpublishしている。

Web管理コンソール

仕様と実装

・ブラウザ上でゲートウェイのホストを指定してもらいroslibからwebsocketでROS(rosbridge_server)とやりとり。ドローンのカメラ映像はmjpeg_serverに接続して受信し表示。
・複数ブラウザも可能で、1台のゲートウェイ（ROS）に複数のPCがWebコンソールとして接続可能。

設計実装上の課題とその対処

・素朴なWebサービスのようにWebアプリケーションサーバを立ててサーバサイドでやる方式ではrosbridgeとサーバが2枚でダブルことになり設計として汚くなるため、全てクライアントサイドでプレゼンテーションレイヤの処理を行うSPA（Single Page Application）方式をとっている。そのため、通常のWebアプリなら割と簡単なドローンの登録削除一覧や永続化のような機能をROS側で実装している。
これは将来Dronecode+FlightPlannerのようなOSSと連携する機能を開発したいときに好ましくない設計になってしまっている。
・CDNで配信される静的ファイル（cssとかjsとかimgとか）は、こうしたゲートウェイ方式のフィールドアプリケーションシステムの場合はまずい。全てローカルコピーが必要。
（恥ずかしながら、実機検証に行った時に気付いた…。）

Gazeboシミュレータ / rviz

仕様と実装

・tum_simulatorをベースに、アクチュエーションのプラグイン（農薬散布を想定したノズル等）、センサープラグイン（距離センサ等）、ワールドプラグイン（風の効果等）を作成し、検証に利用
・本基盤ソフトで対応する実機と、tum_simulator上のドローンでROSメッセージを共通化している
・他に、3Dのメッシュ情報を生成するツール（瓦礫生成ツール）や、rvizとデータを同期させるプラグインなどを作成

設計実装上の課題とその対処

・tum_simulatorをGazebo+ROSで使おうとすると、現状gazebo2.5+ROS-indigoじゃないとまともに動かない。そのため使えるGazeboのバージョンが若干古くなる（最新は7.1.0）。

その他 生じた課題

・BebopDroneがデフォルトでは複数台の同時コントロールができない問題
（通常、ドローン本体が無線LANの親機になり、複数台同時操作するためには
ドローンを子機化しそれぞれ別々のIPを振る必要がある）

→ Bebopのファームウェアを書き換えると出来る
（情報必要でしたら弊社に問合せください）

・VisualSLAM機能を実装する上での課題
- ORB-SLAMを始め、VisualSLAMは多くのOSS実装がGPLになっている
（ROSパッケージはBSDライセンスであるものが多い）
→ 独自に実装することに。
- Gazebo上でのロジック検証はスムーズにいったが、BebopDrone実機で
検証すると遅延やノイズの影響でかなり精度が落ちる
（ORB-SLAM自体でも同様の現象が起きる）
→ VisualSLAMのstate-of-the-artレベルの精度の限界に加え、
そもそものゲートウェイ方式でSLAMを行うことの困難さに直面

・mjpeg_serverがコーデックとデータ転送で計算負荷を食いまくる問題
また、映像が実カメラ→ROS→mjpeg_server→canvasと経由し遅延しまくる

デモ

デモ(1）
- ドローンの登録、Gazeboシミュレータ上のドローンの登録
- 音声（julius）認識でのドローンの操作
- Webブラウザからのドローンの操作

デモ（2）
- Gazebo上でのVisualSLAMの動作デモ

デモ（3）
- 会場のSfMによる3次元地図生成+自律Navigationのデモ
（事前検証が必要であり、会場で本当にできるかは未定）

今後の予定

・市販ドローンではなくDronecodeベースで弊社基盤ソフトの
リファレンスドローンを開発し、自律制御機能を完成させる。

→ ローカライゼーション/SLAMの精度をソフトウェアだけで向上させるのは
困難であり、機体とセットで開発することに。

→ これはIoTシステム全般にいえる教訓なのではないかと思う。
ハードだけ、ソフトだけ、で課題を解くのではなくハードとソフト
の総合システムとして最もうまく課題解決可能な構成をとるのが非常に重要。

・オンボード処理、クラウドロボティクス
→ ゲートウェイからのリモートコマンド方式は遅延やコーデック負荷などが
クリティカルになるため、オンボードとゲートウェイ/クラウド で計算を
分担できるようアーキテクチャを拡張する。

・自律制御機能の強化
- アクチュエーション（農薬散布、インフラ打音点検）も含めた制御
- スポーツやエンタメで使うための画像認識と連動した制御
- クラウド側の解析に必要なデータ収集のための制御

・クラウド上のデータ解析サービスでシームレスに接続
- ドローンで収集したデータをクラウド上でシームレスに解析（主にバッチ処理）
できるようにする機能強化（すなわち、rosbagのデータライフサイクルマネジメント）

ご清聴ありがとうございました。

（是非この後も個別にディスカッションしましょう！）