ユーザーズマニュアル

鳥瞰視点カメラロボット

（ドローン）

発行日 2018 年 3 月 28 日
公立大学法人会津大学
株式会社東日本計算センター

目　次

1．はじめに

1.1．鳥瞰視点カメラロボットとは

鳥瞰視点カメラロボット（以下ドローン）は、無人航空機の事を指し、本書は 4 発ローターのマルチコプターを用いたシステムのユーザーマニュアルです。

本システムは 2015～2017 年度 会津大学 ロボットバレー創出推進事業の一環として開発したものです。

1.2．用語集

本書で用いる用語を一覧に示します。

表 1-1 用語一覧

用語	読み	説明
DroneKit	ドローンキット	3D Robotics 製ドローン開発プラットフォーム。OSS ライセンスは Apache License 2.0。
ArduPilot	アルデュパイロット	ドローンだけではなく、飛行機や Rover にも活用可能なオープンソフトウェア。OSS ライセンスは GPLv3。
Armed	アームド	ドローン飛行可能状態
disarm	ディスアーム	ドローン飛行不可状態
RTL	アールティーエル	フライトモードの一種。 ドローン離陸場所に帰還する。
Guided	ガイデッド	フライトモードの一種。 指定した位置に移動。
Stabilize	スタビライズ	フライトモードの一種。 GPS 等のセンサーを使用して自動的に姿勢制御をせずに手動操作する。
Loiter	ロイター	フライトモードの一種。 GPS などのセンサーを使用して自動的に姿勢制御する。

1.3. 略語集

本書で用いる略語を一覧に示します。

表 1-2 略語一覧

略語	説明
FC	フライトコントローラー
RPi	Raspberry Pi
PS4	PlayStation4
RTL	Return To Launch

1.4. 動作環境

ドローン及び地上局の動作環境一覧は以下のとおりです。

1.4.1. ドローン

表 1-3 ドローン環境一覧

環境		バージョン	補足
OS	Raspbian	Base on XXX[1]	XXX：OS 名(Jessie 等)
CPU	900MHz quad-core ARM Cortex-A7 CPU	-	-
メモリ	1GB	-	-
ストレージ	microSD 16GB	-	1.5 使用機器 No.11 参照
RT ミドルウェア	OpenRTM-aist-Python	1.1.0	-
依存ライブラリ	DroneKit-Python	2.9.0	-
	ArduPilot	3.4	-
	OpenCV	2.4.1	ロボット USB カメラ RTC を使用する場合に必要
	Picamera	1.13	ロボット RPi カメラ RTC を使用する場合に必要

[1] Navio+, Navio2(フライトコントローラー)製造元 EMLID 社によるカスタマイズ
https://docs.emlid.com/navio/common/ardupilot/configuring-raspberry-pi/

1.4.2. 地上局（PC）

表 1-4 地上局環境一覧

環境		バージョン	補足
OS	Ubuntu	14.04 LTS	-
CPU	Core i5-2450M CPU @ 2.50GHz × 4	-	-
メモリ	4GB	-	-
ストレージ	HDD 750GB	-	-
RT ミドルウェア	OpenRTM-aist-Python	1.1.0	-
	OpenRTM-aist	1.1.1	-
依存ライブラリ	kivy	1.9.1	-
	-	-	-

1.5. 使用機器

表 1-5 使用機器一覧

No	機器名称	個数	補足
1	Lenovo G570	1	ホスト OS：Ubuntu 14.04 LTS
2	DIY Quad Kit	1	3D Robotics 社製 DIY ドローン。 生産・販売終了モデル。
3	PRi	1	ドローンに搭載。Element14 製。
4	Navio+	1	EMLID 社製 FC
5	GPS Antenna	1	GPS/GNSS アンテナ MCX コネクター。 Navio+の ANT 端子に接続。
6	プロポ送受信機セット	1	ドローンの操縦装置(Futaba 製 T14SG)
7	SBUS to PPM Decoder	1	FrSKY 製。Navio+で上記プロポ使用するのに必要。
8	DUAL SHOCK 4	1	旧型(CUH-ZCT1J)の PS4 コントローラー
9	Wi-Fi ドングル	1	IODATA 製 WN-G300UA
10	屋内 Wi-Fi ルーター	1	Buffalo 製 WXR-2533DHP
11	microSD カード	1	Radius 製 RP-MSU16X
12	ドローンバッテリー	1	KyPOM 製。4 セル, 4200mAh, 14.8V
13	バッテリー充電器	1	HYPERION 製 EOS0606i AC/DC
14	USB カメラ	1	Logicool 製 C930e

1.6. 関連資料

表 1-6 資料一覧

No	資料名	パス(RTC-Library-FUKUSHIMA)
1	機能仕様書_DroneIO	https://rtc-fukushima.jp/component/1126/
2	機能仕様書_DroneViewer	https://rtc-fukushima.jp/component/1031/
3	機能仕様書_DroneController	https://rtc-fukushima.jp/component/1175/
4	RTC 概要_RTC_GameController	https://rtc-fukushima.jp/component/1129/
5	機能仕様書_DroneMapViewer	https://rtc-fukushima.jp/component/1776/
6	機能仕様書_RobotUSBCam	https://rtc-fukushima.jp/component/1531/
7	機能仕様書_RobotRPiCam	https://rtc-fukushima.jp/component/2421/
8	機能仕様書_RobotImageUploader	https://rtc-fukushima.jp/component/2420/

2．本システムでできること

①. 離着陸

離陸は高度 10m まで上昇し、着陸はドローン起動位置への帰還のみです。離着陸中の他操作は出来ません。

②. 移動

現座標に対して（移動速度 × ターゲット経緯度算出係数）分の経緯度差分[2]を加味したターゲット座標を設定し移動します。なお、移動モードの切り替え（東西南北移動<->機体の向きに対しての前後左右移動）が可能です。初期設定は東西南北移動となります。

ターゲット速度の初期設定は 3m/s です。1～10m/s(1m/s 刻み)まで変更可能です。

③. 高度上昇/下降

ホームポジションの高度から 2～15m(1m 刻み)まで変更可能です。高度下降による着陸は出来ません。

④. 旋回

機体の向きに対して右回りまたは左回りに旋回します。

⑤. センサー情報表示

専用ビューアにセンサー情報等を表示します。詳細は 3.2 で後述します。

⑥. 撮影

撮影間隔ごとに撮影を行い画像と画像付加データを保存します。撮影間隔の初期設定値は 1s です。0.1s 以上の値で変更可能です。他にも変更可能な項目があります。詳細は表 7-3 を参照ください。

撮影した画像および付加データは、sftp で取り出すか Ubuntu がインストールされている PC に SD カードを挿入して取り出すことができます。付加データのフォーマットは CSV 形式で、各項目は表 2-1 のとおりです。

[2] 10m あたりの緯度差分: 0.000137 度、10m あたりの経度差分: 0.000108 度（会津での使用を想定）

表 2-1　ロボット USB カメラ(またはロボット RPi カメラ)付加データフォーマット

項目	説明
ファイル名	保存される画像のファイル名
フォルダパス	画像を保存しているフォルダパス
サイズ幅	画像幅のピクセル数
サイズ高さ	画像高さのピクセル数
ロボット識別子	使用したロボットの識別子
カメラ No	使用したカメラの識別番号
緯度	撮影時の緯度(DEG 形式)
経度	撮影時の経度(DEG 形式)
高さ	撮影時の高さ(単位:m)
方角	撮影時の向き(DEG 形式)
ピッチ角	撮影時のピッチ角(単位:rad)
ロール角	撮影時のロール角(単位:rad)
年月日	撮影時の年月日
時間	撮影時の時刻(時分秒ミリ秒)
撮影計画名	撮影の計画名(最大:半角 30 文字)

3．本システムのユーザーインターフェースについて

3.1．ドローンコントローラー仕様

本システムでは、PS4 コントローラーによる操作が基本ですが、従来通りプロポでの操作も可能です。

3.1.1．PS4 コントローラー仕様詳細

コントローラーのキー仕様は下表のとおりです。

キーが押下されている限り動作は継続します。相反する操作(例：高度上昇/下降を同時に要求)をした場合は何もしません。

なお、Drone Controller RTC の動作周期は 100 ミリ秒であるため、**動作周期以下でのキー状態変化検知は行いません**（例：離陸指示する場合には R1 キーを 100 ミリ秒間押し続けます)。

表 3-1 キーマッピング一覧

キー名称	動作
十字キー上	移動(北または機体の向きに対して前方)
十字キー下	移動(南または機体の向きに対して後方)
十字キー右	旋回(右回り)
十字キー左	旋回(左回り)
□	移動(西または機体の向きに対して左方向)
○	移動(東または機体の向きに対して右方向)
△	高度上昇
×	高度下降
R1	離陸
R2	帰還
L1	ターゲット速度変更(+1m/s)
L2	ターゲット速度変更(-1m/s)
OPTIONS	移動モード切り替え
上記以外	－

3.2. ビューア

3.2.1. ドローンビューア

以下にドローンビューアおよび各表示項目について説明します。Quick タブは簡易画面(初期表示画面)、Detail タブを選択すると各種情報を表示する詳細画面となります。

図 3-1 ドローンビューア(簡易画面)

図 3-1 の(1)～(4)の詳細は以下のとおり。

表 3-2 簡易画面内の各種メーターとエラーメッセージ詳細

No.	名称	説明
1	ドローン高度	ドローンの現在の高度
2	3 軸ジャイロ	上段: Yaw, 下段: Roll(中央), Pitch(両端)
3	加速度	進行方向の速度(km/h と m/sec 表示) メーター最大時速は 72km/h
4	エラーメッセージ	ドローンからのエラーメッセージを表示 (表 8-1 参照)

エラーメッセージの一例として、地上局との通信が途絶した場合は、以下のメッセージを表示します。

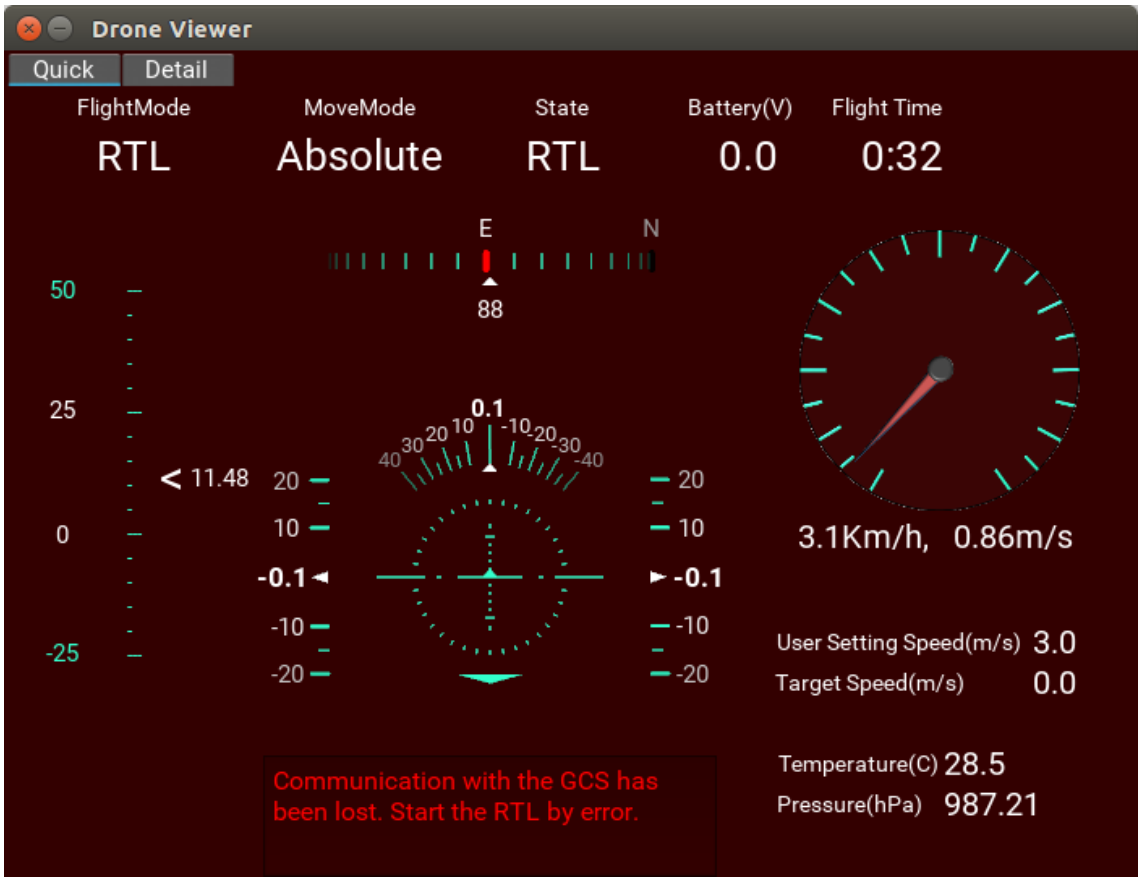


図 3 - 2 通信途絶時のビューア(簡易画面)

ドローンビューアの詳細画面は以下のとおりです。各項目の詳細は次頁の表を参照ください。

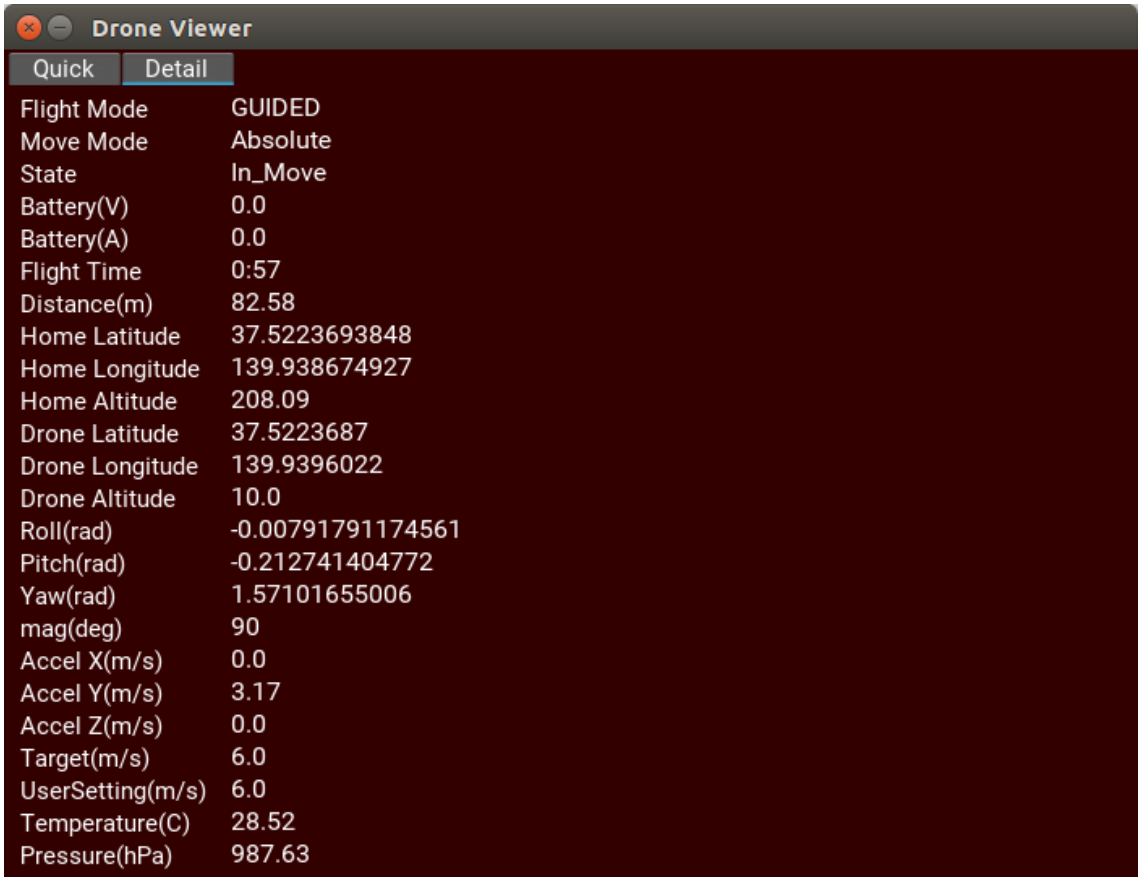


図 3 - 3 ドローンビューア詳細画面

表 3-3 ドローンビューア表示項目一覧

<table>
<tr><th colspan="2">項目</th><th>単位</th><th>説明</th></tr>
<tr><td colspan="2">Flight Mode</td><td>-</td><td>現在のフライトモードを表示</td></tr>
<tr><td colspan="2">Move Mode</td><td>-</td><td>現在の移動モードを表示
<table>
<tr><th>モード</th><th>意味</th></tr>
<tr><td>Absolute</td><td>東西南北移動</td></tr>
<tr><td>Relative</td><td>機体の向きに対する移動</td></tr>
</table></td></tr>
<tr><td colspan="2">State</td><td>-</td><td>ドローン(I/O RTC)の内部モードを表示
<table>
<tr><th>モード</th><th>意味</th></tr>
<tr><td>Disarm</td><td>ドローン飛行不可能状態</td></tr>
<tr><td>Armed</td><td>ドローン飛行可能状態</td></tr>
<tr><td>Takeoff</td><td>離陸中</td></tr>
<tr><td>RTL</td><td>帰還中</td></tr>
<tr><td>In_Move</td><td>移動中</td></tr>
<tr><td>Alt_CHG</td><td>高度変更中</td></tr>
<tr><td>Hovering</td><td>ホバリング中</td></tr>
<tr><td>Yaw_CHG</td><td>旋回中</td></tr>
</table></td></tr>
<tr><td colspan="2" rowspan="2">Battery</td><td>V</td><td>バッテリーの電圧値</td></tr>
<tr><td>A</td><td>バッテリーの電流値[3]</td></tr>
<tr><td colspan="2">Flight Time</td><td>分秒</td><td>フライト時間</td></tr>
<tr><td colspan="2">Distance</td><td>m</td><td>ホームポジション(ドローン起動位置)とカレントポジション(ドローン)の距離</td></tr>
<tr><td rowspan="3">Home
Drone</td><td>lat.</td><td rowspan="2">10 進</td><td rowspan="3">ホームポジションとカレントポジションの緯度/経度/高度。ホームポジションの高度は海抜高度、カレントポジションの高度はホームポジションの高度からの相対値。</td></tr>
<tr><td>lon.</td></tr>
<tr><td>alt.</td><td>m</td></tr>
<tr><td colspan="2">Roll, Pitch, Yaw</td><td>rad.</td><td>3 軸ジャイロセンサー</td></tr>
<tr><td colspan="2">mag</td><td>deg.</td><td>磁気コンパス(=機体の向き)
北を 0° とし時計回りに角度を表示。</td></tr>
<tr><td colspan="2">Accel</td><td>m/sec</td><td>3 軸加速度センサー</td></tr>
<tr><td colspan="2">Target Speed</td><td>m/sec</td><td>ターゲット速度</td></tr>
<tr><td colspan="2">UserSetting</td><td>m/sec</td><td>ユーザーがコントローラーにて指定した速度</td></tr>
</table>

[3] 本システム構成(RPi2&Navio+)では取得できません。

Temperature	C	気温
Pressure	hPa	気圧

3.2.2. ドローンマップビューア

ドローンの位置情報を地図上(Open Street Map)に表示します。以下では機首の向きに対して前進指示をした場合のマップビューアになります。指示をした際に現在位置からターゲット位置までのガイドラインを表示します。

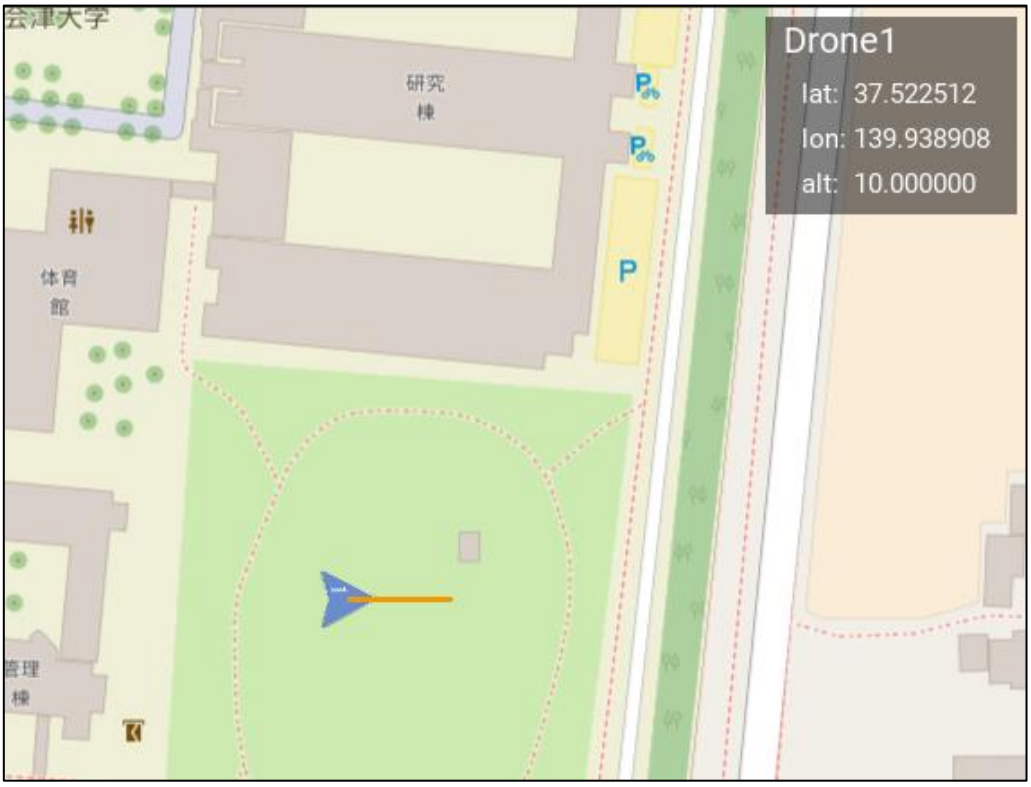


図 3-4 ドローンマップビューア

4．システム配置図

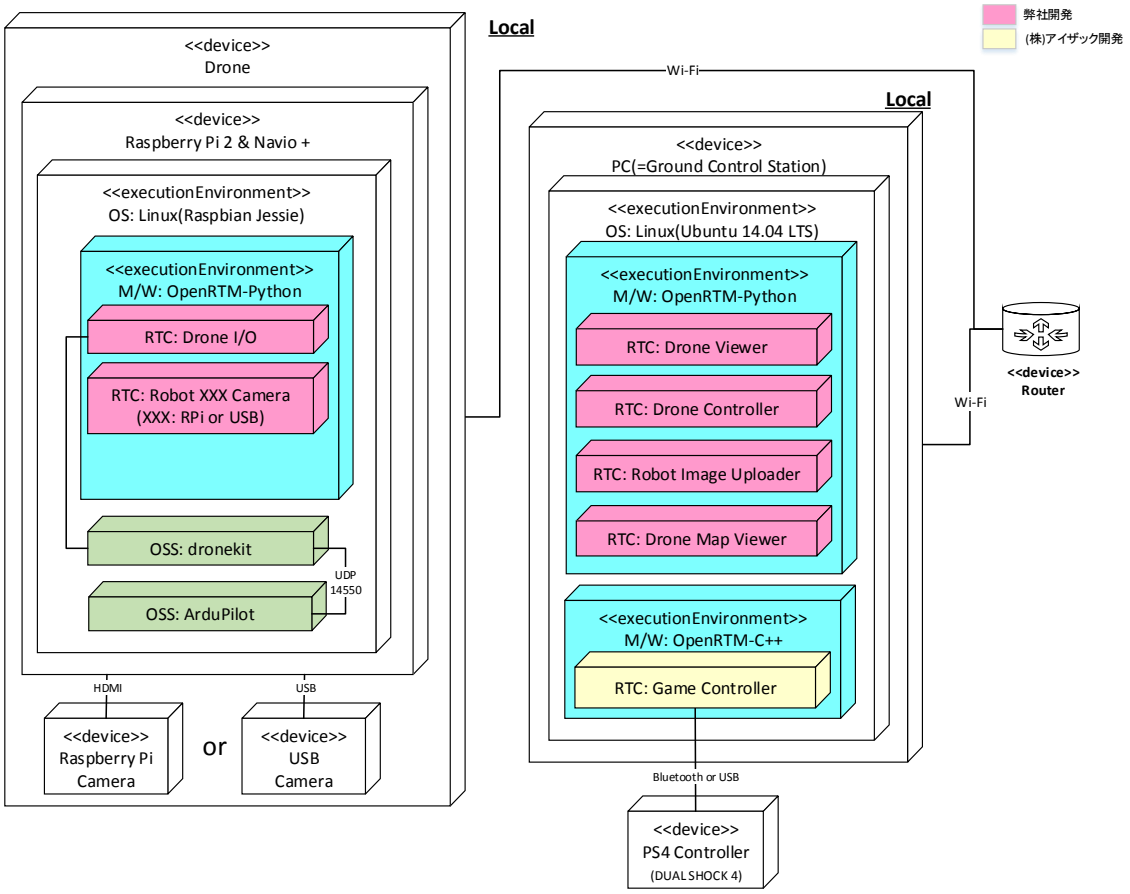


図　4 - 1　システム配置図

以下は実際のシステム構成一例（PS4 コントローラーは無線接続）です。

図　4 - 2　実際のシステム構成一例

5．システム構成

表 5-1 システム構成一覧

ディレクトリ名	ファイル名	説明
DroneIO	DroneIO.py	実行ファイル
	drone_distance_cal.py	2 点間距離算出ファイル
	DroneDataType.idl	独自型 IDL ファイル
	DroneDataType_idl.py	IDL ファイルを基に生成された python ファイル
	DroneIO.conf	コンフィギュレーションファイル
	rtc.conf	
	RTC.xml	プロファイル
DroneController	DroneController.py	実行ファイル
	target_gps.py	ターゲット GPS 算出
	drone_distance_cal.py	2 点間距離算出ファイル
	DroneDataType.idl	独自型 IDL ファイル
	DroneDataType_idl.py	IDL ファイルを基に生成された python ファイル
	DroneController.conf	コンフィギュレーションファイル
	rtc.conf	
	RTC.xml	プロファイル
DroneViewer	DroneViewer.py	実行ファイル
	drone_viewer.py	ビューア本体ファイル
	drone_viewer.kv	画面生成ファイル
	drone_distance_cal.py	2 点間距離算出ファイル
	DroneDataType.idl	独自型 IDL ファイル
	DroneDataType_idl.py	IDL ファイルを基に生成された python ファイル
	DroneViewer.conf	コンフィギュレーションファイル
	rtc.conf	
	RTC.xml	プロファイル

RTC_GameController	/include/RTC_GameController.h	ヘッダファイル
	/src/RTC_GameController.cpp	ソースファイル
	RTC_GameController.conf	コンフィギュレーションファイル
	rtc.conf	
	RTC.xml	プロファイル
DroneMapViewer	DroneMapViewer.py	実行ファイル
	map_create.py	画面生成ファイル
	DroneDataType.idl	独自型 IDL ファイル
	DroneDataType_idl.py	IDL ファイルを基に生成された python ファイル
	DroneMapViewer.conf	コンフィギュレーションファイル
	rtc.conf	
	RTC.xml	プロファイル
RobotUSBCam	RobotUSBCam.py	実行ファイル
	RobotUSBCam.conf	コンフィギュレーションファイル
	rtc.conf	
	RTC.xml	プロファイル
	sno.ini	セクション No 保持 ini ファイル
RobotRPiCam	RobotRPiCam.py	実行ファイル
	RobotRPiCam.conf	コンフィギュレーションファイル
	rtc.conf	
	RTC.xml	プロファイル
	sno.ini	セクション No 保持 ini ファイル
RobotImageUploader	RobotImageUploader.py	実行ファイル
	DroneDataType.idl	独自型 IDL ファイル
	DroneDataType_idl.py	IDL ファイルを基に生成された python ファイル
	RobotImageUploader.conf	コンフィギュレーションファイル
	rtc.conf	
	RTC.xml	プロファイル
	sno.ini	セクション No 保持 ini ファイル

6．システムの導入

6.1．インストール

RTC-Library-FUKUSHIMA から表 1-6 の RTC をダウンロードし、任意のディレクトリに配置し解凍します。なお、ドローン I/O およびロボット USB カメラ RTC(またはロボット RPi カメラ)は PRi の任意のディレクトリ(例：/home/ユーザー名/)に転送し解凍します。

6.1.1．地上局

6.1.1.1．OpenRTM-aist

C++版 1.1.1 および Python 版 1.1.0 をダウンロードページ(2018/3 現在)に従い、インストールください。

http://www.openrtm.org/openrtm/ja/content/openrtm-aist-c-111-release

http://www.openrtm.org/openrtm/ja/content/openrtm-aist-python-110-release

6.1.1.2．OpenRTP

RTC Builder および RT System Editor を含む開発ツールです。手順は以下リンク先を参照ください(2018/3 現在)。バージョンは 1.1.0-RC5 です。

http://openrtm.org/openrtm/ja/node/5778

6.1.1.3．cmake

Game Controller RTC の開発言語は C++のため、ビルド時に cmake が必要になります。

```
$ sudo apt-get update
$ sudo apt-get install cmake
```

6.1.1.4．Doxygen

Game Controller RTC のビルド時に必要になります。

```
$ sudo apt-get update
$ sudo apt-get install doxygen doxygen-gui graphviz
```

6.1.1.5. kivy

ドローンマップビューア/ドローンビューア RTC は kivy を用いて実装していますので、本パッケージ(安定版)をインストールします。python-kivy-examples はオプション(デモやサンプルコード等)ですので必要に応じてインストールください。

```
$ sudo add-apt-repository ppa:kivy-team/kivy
$ sudo apt-get update
$ sudo apt-get install python-kivy
$ sudo apt-get install python-kivy-examples
```

6.1.1.6. mapview

kivy 上での地図表示要件を実現するために必要になります。mapview は kivy-garden(kivy のアドオン)として提供されていますので、kivy-garden のインストールも必要になります。さらに mapview 必要要件モジュールとして concurrent.futures と requests のインストールも必要になります。

```
$ sudo pip install futures requests
$ sudo pip install kivy-garden
$ garden install mapview
```

6.1.1.7. ntp

外部 NTP サーバー(NICT(情報通信研究機構))と同期を取るためにインストールをします。

(1) ntp をインストールします。

```
$ sudo apt-get update
$ sudo apt-get install ntp
```

(2) NICT の NTP サーバーと同期を取るための設定をします。

```
$ sudo vi /etc/ntp.conf
```

以下のホスト名を記述後、保存して閉じます。

```
server ntp.nict.jp
```

(3) 設定内容を即時反映します。

```
$ sudo service ntp stop
$ sudo ntpdate ntp.nict.jp
$ sudo service ntp start
```

(4) 同期したことを確認するために以下のコマンドを実行します。remote 欄に「外部 NTP サーバー名」が表示されましたら同期成功を意味します。

```
$ sudo ntpq -p
```

```
remote           refid      st t when poll reach   delay   offset  jitter
==============================================
 ntp-b3.nict.go. .NICT.       1 u   59   64    3   78.679   -3.819   1.434
```

6.1.2. ドローン

DIY Quad Kit の組立およびセットアップが完了していることを前提に説明します。FC は Pixhawk(DIY Quad Kit に同梱)ではなく、Navio+を使用します。

なお、「1.3. 使用機器」と異なる機器構成の場合は、適宜読み替えが必要になります。

6.1.2.1. FC セットアップ

(1) OS インストール

a. Emlid 社の以下のリンク(2018/3 現在)から OS のイメージファイルをダウンロードします。
https://docs.emlid.com/navio/common/ardupilot/configuring-raspberry-pi/

b. OS イメージを microSD カードへ書込みます。a.のリンク先にチュートリアルがありますので、それに従います。

c. microSD カードを RPi に挿したら、USB 外付キーボード、Wi-Fi ドングル、HDMI(または HDMI→他映像信号変換)対応ディスプレイを RPi に接続し、RPi の電源を投入します。

d. SD カードの空き容量を拡張するために、以下のコマンドを実行します。実行完了後、RPi を再起動してください。

```
$ sudo raspi-config --expand-rootfs
```

(2) RPi 最大電源供給量の変更(RPi2 のみ)

今回使用する Wi-Fi ドングルおよび USB カメラの最大消費電流が PRi2 の USB ポートからの最大電源供給量(0.6A)をオーバーしてしまい、動作不安定になりますので、最大電源供給量の変更(1.2A)を行います。なお、PRi3 の場合は 1.2A が初期設定値ですので設定不要です。

```
$ sudo vi /boot/config.txt
```

txt ファイル最後尾に以下を追記し、保存して閉じます。

```
max_usb_current=1
```

(3) 時刻合わせ

RPi は Real Time Clock モジュール非搭載のため、地上局側の時刻と同期を取ります。それに必要なライブラリをインストールします。なお、インストールにて apt コマンドや wget コマンドを使用しますが、インターネット環境が必要となりますので、各自のインターネット環境に合わせて事前に設定してください。

a. ntp をインストールします。

```
$ sudo apt-get update
$ sudo apt-get install ntp
```

b. 地上局と時刻同期を行います。

```
$ sudo vi /etc/ntp.conf
```

地上局の IP アドレスを指定してください。

```
server 地上局の IP アドレス
```

c. 設定内容を即時反映します。

```
$ sudo service ntp stop
$ sudo ntpdate 地上局の IP アドレス
$ sudo service ntp start
```

d. 同期したことを確認するために以下のコマンドを実行します。

```
$ sudo ntpq -p
```

remote 欄に「*地上局のマシン名」が表示されましたら同期成功を意味します。

e. 日本時間に変換するには以下のコマンドを実行します。

```
$ sudo timedatectl set-timezone Asia/Tokyo
```

(4) ホスト名変更

PRi の初期ホスト名が「navio」になっていますので、以下の 2 つの設定ファイルにて任意のホスト名に変更します。他の機体と重複しないようにしてください。

```
$ sudo vi /etc/hosts
```

「navio50」に変更した場合の例です。記述後に保存して閉じます。

```
127.0.1.1 navio50
```

```
$ sudo vi /etc/hostname
```

「navio50」に変更した場合の例です。記述後に保存して閉じます。

```
navio50
```

(5) dronekit

a. 以下のコマンドを実行します。

```
$ sudo apt-get update
$ sudo apt-get install python-pip python-dev
$ sudo pip install dronekit
```

b. 以下のコマンドでインストールできたことを確認します。

```
$ pip show dronekit
```

(6) OpenRTM-aist

a. 一括インストールのためのシェルスクリプトが GitHub 上にありますので、PRi 上から Clone します。

```
$ git clone https://gist.github.com/ababup1192/62d0ec4da753d6997dc1.git
```

b. Clone したディレクトリに移動し、provision.sh を実行します。

```
$ sh provision.sh
```

c. 以下のコマンドでインストールできたことを確認します。

```
$ dpkg -l 'openrtm*'
```

(7) OpenCV

a. 以下のコマンドを実行します。

```
$ sudo apt-get update
$ sudo apt-get install python-numpy
$ sudo apt-get install python-opencv
```

b. 以下のコマンドでインストールできたことを確認します。

```
$ dpkg –l python-opencv
```

(8) Picamera

a. 以下のコマンドを実行します。

```
$ sudo apt-get update
$ sudo apt-get install python-picamera
```

b. 以下のコマンドでインストールできたことを確認します。

```
$ dpkg –l python-picamera
```

(9) APM(ArduPilot)インストール

以下のリンク先に従います(Overview～Autostarting on boot) (2018/3 時点)。なお、本システムの対象 vehicle は Arducopter、「Specifying launching options」にて指定する IP は自機(127.0.0.1)になります。

https://docs.emlid.com/navio/common/ardupilot/installation-and-running/[4]

(10) DHCP クライアントデーモンの停止

IP アドレス設定を自動振り分けではなく固定とする場合は、以下のコマンドで機能を停止することができます。

```
$ sudo systemctl disable dhcpcd
```

(11) Wi-Fi ドングルの設定

a. 地上局 PC の無線 LAN 機能および Wi-Fi ルーターのセットアップを事前に行ってください。手順は使用する機器や OS によって異なりますので割愛します。
b. /etc/wpa_supplicant/wpa_supplicant.conf に記載する SSID の Key の暗号化を以下のように実施します。SSID と key はお手持ちの Wi-Fi ルーターの SSID と key を入力します。

```
$ sudo sh -c “wpa_passphrase SSID Key >> /etc/wpa_supplicant/wpa_supplicant.conf”
```

[4] リンク先のドキュメントは GitHub にて管理
https://github.com/emlid/navio-docs/blob/master/docs/common/ardupilot/installation-and-running.md

c. /etc/wpa_supplicant/wpa_supplicant.conf を開き、以下の設定になっていることを確認します。下記以外の network がある場合は、コメントアウトします。また、**#psk の行は key の平文なので、必ず削除してください。**

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
   ssid="SSID"
   #psk=Key
   psk=パスフレーズ後の Key の値
}
```

d. c.に以下(太字部分)を追記します。認証方式および暗号方式は使用する Wi-Fi ドングルのスペックに合わせてください。

```
network={
   ssid="MyNetwork"
   psk=パスフレーズ後の Key の値
   key_mgmt=WPA-PSK
   proto=WPA WPA2
   priority=2
}
```

e. /etc/network/interfaces を開き、以下の設定(例:192.168.11.2)にします。なお、変更前にバックアップもしくはコメントアウトすることを推奨します。

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
address 192.168.11.2
netmask 255.255.255.0
gateway 192.168.11.1
```

```
iface default inet dhcp
```

f. PRi を再起動し、ifconfig コマンドにて、wlan0 の IP が割当されていることを確認します。

g. 8192CU[5]設定ファイル(/etc/modprobe.d/8192cu.conf)のパワーマネジメント機能を以下の方法で OFF します(デフォルト(ON)のまま使用すると、頻繁に切断が発生するようです)。変更完了後、RPi を再起動します。

```
options 8192cu rtw_power_mgnt=0
```

h. PC から RPi にリモート接続(ssh コマンド等)出来たら、設定完了です。

(12) 機体のキャリブレーションをします。以下リンク先を参考ください。
https://www.youtube.com/watch?v=DDKhaUwiX5Y

[5] 今回使用した Wi-Fi ドングルに搭載しているチップ

6.2. 起動

6.2.1. ドローン

6.2.1.1. ドローン I/O RTC

(1) ドローン本体にバッテリーを装着し、ドローン本体の電源端子に挿すことで電源を投入します。

(2) FC の LED が橙色点滅(=PRi 起動完了)になるまで待ちます。

(3) 地上局のターミナルを起動し、ssh でリモートログインします。

```
$ ssh ログイン名@ホスト名または IP アドレス
```

(4) ドローン I/O RTC のディレクトリに移動します。

```
$ cd DroneIO
```

(5) ネームサーバーを起動します。

```
$ rtm-naming
```

(6) FC の LED が緑色点滅(=GPS データ取得済み)であることを確認します。少し時間を置いても点滅しない場合は PRi の再起動を行ってください。

(7) 実行ファイルを起動します。

```
$ python DroneIO.py
```

ドローン起動場所の GPS データの取得を行いますが、3 秒以内で取得出来なかった場合は、コンソールに以下のエラーメッセージが表示されます。その場合は、RT System Editor でドローン I/O RTC を Exit してから再度実行ファイルを起動してください。

```
Waiting for home location...
Waiting for home location...
Waiting for home location...
home_location timeout. Please re-startup this RTC.
```

(8) 使用機器以外のリポバッテリーを使用する場合は、コンフィグパラメータ(表 7-1 参照)の BattCell の変更が必要になります。ただし、セル数が同じであれば、変更不要です。
バッテリー容量チェックそのものが不要である場合は、BattChk を Off にしてください。ただし、**フライト前後に必ずバッテリーチェッカーにて容量が不足していないかを確認してください。**

6.2.1.2. ロボット USB カメラ RTC(USB カメラを使用する場合)

(1) 地上局のターミナルを起動し、ssh でリモートログインします。

```
$ ssh ログイン名@ホスト名または IP アドレス
```

(2) ロボット USB カメラ RTC のディレクトリに移動します。

```
$ cd RobotUSBCam
```

(3) 実行ファイルを起動します。

```
$ python RobotUSBCam.py
```

6.2.1.3. ロボット PRi カメラ RTC(Raspberry Pi カメラを使用する場合)

(4) 地上局のターミナルを起動し、ssh でリモートログインします。

```
$ ssh ログイン名@ホスト名または IP アドレス
```

(5) ロボット RPi カメラ RTC のディレクトリに移動します。

```
$ cd RobotRPiCam
```

(6) 実行ファイルを起動します。

```
$ python RobotPRiCam.py
```

6.2.2. 地上局

6.2.2.1. 事前準備

(1) 新たにターミナルを起動し、ネームサーバーを起動します。

6.2.2.2. ドローンビューア RTC

(1) 新たにターミナルを起動し、ドローンビューア RTC のディレクトリに移動します。

(2) 実行ファイルを起動します。ビューアが表示されることを確認します。

```
$ python DroneViewer.py
```

6.2.2.3. ドローンコントローラーRTC

(1) 新たにターミナルを起動し、ドローンコントローラーRTC のディレクトリに移動します。

(2) 実行ファイルを起動します。

```
$ python DroneController.py
```

6.2.2.4. ドローンマップ RTC

(1) 新たにターミナルを起動し、ドローンマップビューア RTC のディレクトリに移動します。

(2) 実行ファイルを起動します。この時点ではまだビューアは表示されません。RTC Active 状態遷移してから 20 秒以内にドローン I/O RTC の GPS データを受信した場合に表示されます。地図表示にインターネット環境が必要ですが、一度受信した地図データはキャッシュに保存され、以降はオフライン環境でも使用可能です。

```
$ python DroneMapViewer.py
```

6.2.2.5. Game Controller RTC

手順(1)~(3)は PS4 コントローラーを**無線接続**する場合のみ必要です。

(1) Ubuntu デスクトップ画面左端のシステム設定 -> Bluetooth をクリックします。

(2) PS4 コントローラーの電源を ON（PS ボタン押下）します。ペアリング未設定の場合は、ペアリングモード（SHARE ボタンと PS ボタン長押し）に切り替えます。切り替わると、コントローラー上部のライトバーが白色点滅します。

(3) 白色点滅中に、Bluetooth 画面左下の＋ボタンをクリックすることで、デバイス一覧に「Wireless Controller」が追加されます。次回以降コントローラーを使用する場合は、PS ボタン押下のみでペアリングします。

図 6-1 Blutooth 設定画面

(4) ビルド方法、実行方法は 1.6. 関連資料の No.4 を参照ください。

6.2.3. 全 RTC 接続

(1) OpenRTP を起動し、「RT System Editor」パースペクティブに切り替えます。

(2) ネームサーバーの追加（ドローンおよび地上局）を行います。

(3) 全 RTC を System Diagram にセットします。

(4) VirtualBox 等の仮想環境上で Game Controller RTC を起動する場合、コンフィギュレーションのデバイス接続先(Port)を/dev/input/jsx(x は PS4 コントローラーに割り振られた番号)に変更します。

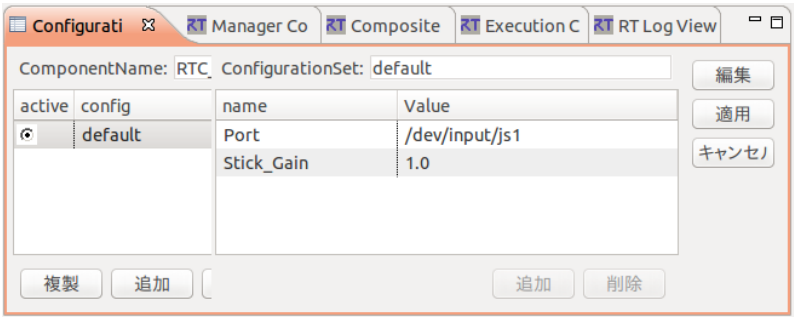


図 6-2 Game Controller RTC のコンフィギュレーションビュー

(5) 必要に応じて、各 RTC のコンフィグパラメータの設定を行います。詳細は 7.各種設定を参照ください。

(6) RTC 間のポート接続方法は以下のとおりです。

a. RTC_GameController

各ポートとも、Connector Profile の変更は以下のみ。Controller_Type ポートは本システムでは使用しません。

Dataflow Type：pull
Buffer(Output)の Buffer length：1

(a) RTC_GameController.Button -> DroneController.ButtonIn

(b) RTC_GameController.Analog -> DroneController.AnalogIn

b. DroneController

各ポートとも、Connector Profile の変更は以下のみ。

Subscription Type：new
Push Policy：new
Buffer(Input)の Buffer length：1

(a) DroneController.outTarget -> DroneIO.inTarget

(b) DroneController.outTargetSpeed -> DroneViewer.inSettingSpeed

(c) DroneController.outMoveMode -> DroneViewer.inMoveMode

c. Drone I/O

各ポートとも、Connector Profile の変更は以下のみ。

Subscription Type：new
Push Policy：new
Buffer(Input)の Buffer length：1

(a) DroneIO.outDroneCtrl -> DroneController.inDrCtrl

(b) DroneIO.outDroneMap -> DroneMapViewer.inDrone1

(c) DroneIO.outDroneViewer -> DroneViewer.inIOComposite

カメラ RTC を使用する場合は以下

(d) DroneIO.outGPSDrone -> RobotUSBCam(Or RobotPRiCam).inGPSRobot

(e) DroneIO.outGyro -> RobotUSBCam(Or RobotPRiCam).inGyro

クラウド(DB)にセンサー情報等を送信する場合は会津ラボ様開発の Adapter-RTC(adptDroneDB)を介して DB にセンサー情報を書き込みます。Connector Profile の変更は以下のみ。

Subscription Type：new
Push Policy：all

(f) DroneIO.outDroneDB -> adptDroneDB.dp_DroneDB

接続完了しますと、以下(カメラ、DB-RTC 除く)のようになります。再度ご確認ください。

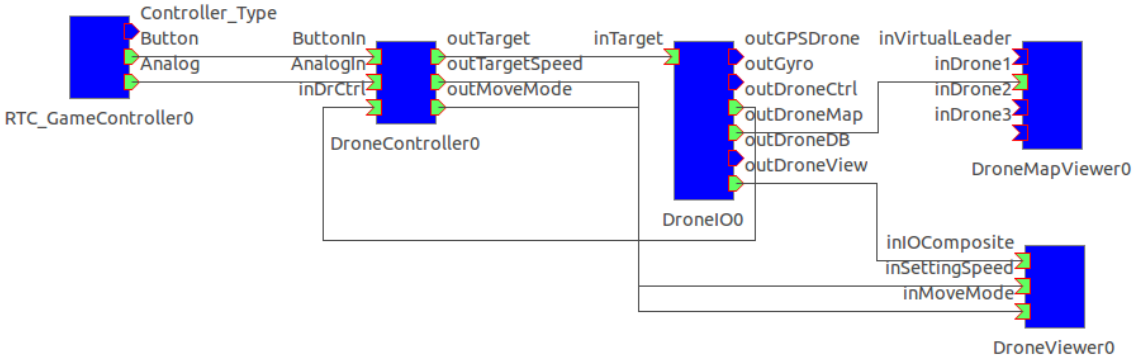


図 6 - 3 ドローンシステム接続完了時のシステムダイアグラム画面

(7) 接続完了後、システムエディタ（RTC 間の接続やコンフィギュレーション情報）を保存することを推奨します。次回以降、保存したシステムエディタの内容を読み込むことで上記(5)(6)の手順を省くことが出来ます。以下の手順で保存をします。

a. システムエディタウインドウ上でコンテキストメニューを開きます
b. Save（または Save as）を選択します
c. Profile Information が開く。Update log 以外は入力必須項目となる。入力し終えたら、OK を選択することで、保存が完了します

読み込みは以下の手順になります。

a. 読み込みに必要な RTC を起動し、ネームサーバー上に登録されていることを確認してください
b. システムエディタウインドウ上でコンテキストメニューを開きます
c. 保存内容を復元したい場合は Open and Restore、そうでない場合は Open を選択し、保存したファイル名を選択することで読み込みが完了します

(8) (6)の Adapter-RTC や DB-RTC を使用する場合は、先に DB-RTC、Adapter-RTC の順に Active し、データ受信できるようにします。

(9) 全 RTC を Activate(=All Activate)します。

以下のように、全て Active 状態に遷移しましたら、ドローンの操作が可能です。操作方法は表 3-1 を参照ください。

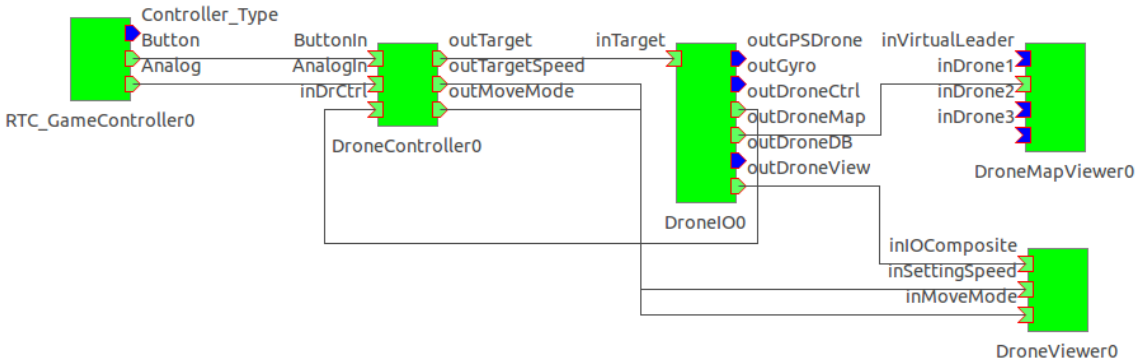


図 6 - 4 ドローンシステム 起動完了後のシステムダイアグラム画面

以下のように、ドローン I/O RTC が Active にならないことがあります。これは Activate 時に 2 秒以内に ArduPilot が Armed 状態に切り替えられなかったためです（ドローンビューアやドローン I/O のコンソールにエラーメッセージ(表 8-1 No.5 参照)を表示します)。その場合は、ドローン I/O RTC の Reset およびそれ以外の RTC を All Deactivate を行い、再度 All Activate してください。

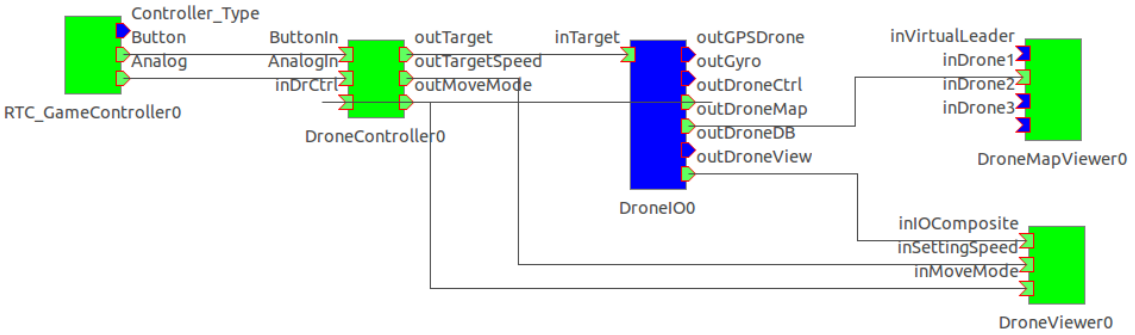


図 6 - 5 ドローン I/O RTC Activate 失敗時のシステムダイアグラム画面

6.2.4. 撮影画像ならびに画像付加データのアップロード

ドローン航行終了後、microSD に蓄積した撮影画像ならびに画像付加データをアップロード(本ロボット事業では事業用に構築したクラウド環境)する場合のみ本 RTC を起動します。なお、本 RTC を使用するユーザーは、以下環境をご用意ください。

・アップロード先に受信側 RTC を自前で開発したのを配置

・アップロード先に DB 構築ならびに画像アップロード用のディレクトリ

6.2.4.1. Robot Image Uploader RTC

(1) microSD に蓄積した撮影画像ならびに画像付加データを地上局の任意のパスにコピーをしてください。

(2) ターミナルを起動し、cd コマンドにて、ロボットイメージ Uploader RTC のディレクトリに移動します。

(3) ロボットイメージ Uploader RTC の実行ファイルを起動します。

```
$ python RobotImageUploader.py
```

(4) 受信側 RTC と接続します。本ロボット事業での受信側 RTC は Adapter-RTC(図 6-6 中央)と DB-RTC(図 6-6 右)[6]を使用しています。

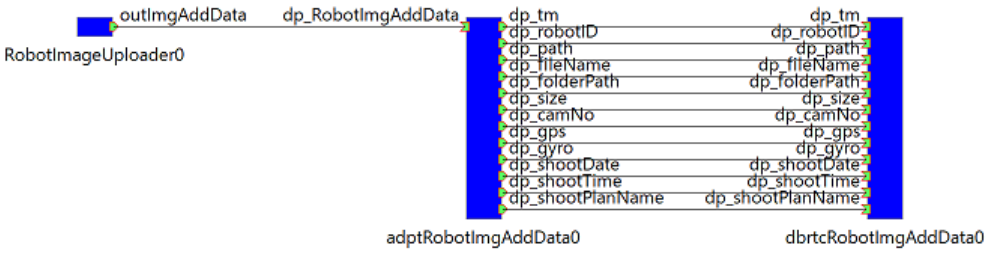


図 6-6 Robot Image Uploader RTC 起動完了後のシステムダイアグラム画面

(5) ロボットイメージ uploader RTC のコンフィグレーションの設定(表 7-7)をします。アップロード先の情報(コンフィグレーションの UserID, Hostname, UploadPath)を設定してください。コンフィグレーションの AddDataFilePath ならびに PictureFilePath は(1)で指定したパスを設定してください。

(6) 先に受信側 RTC を Active し、データ受信できるようにします。

(7) (6)完了後にロボットイメージ uploader RTC を Active にします。

[6] (株)会津ラボ開発の RTC

(8) 画像付加データ送信完了後に、ロボットイメージ uploader RTC のコンソール画面にて、アップロード先のパスワードを問われた場合は、パスワードを入力し Enter を押下してください。

```
                    -Lenovo-G570:~/DroneSystem/DBTestEnv_20171019/RobotImageU
ploader$ python RobotImageUploader.py
The authenticity of host '          ホストネーム          ' can't be established.
ECDSA key fingerprint is                                                      .
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (              /.ssh/known_
hosts).
ユーザーID@ホストネーム's password:
```

図 6 - 7 ロボットイメージ Uploader RTC コンソール画面

(9) 入力完了しますと、撮影画像を指定したパスに転送します。転送完了後にロボットイメージ uploader RTC は Inactive に遷移し処理終了となります。

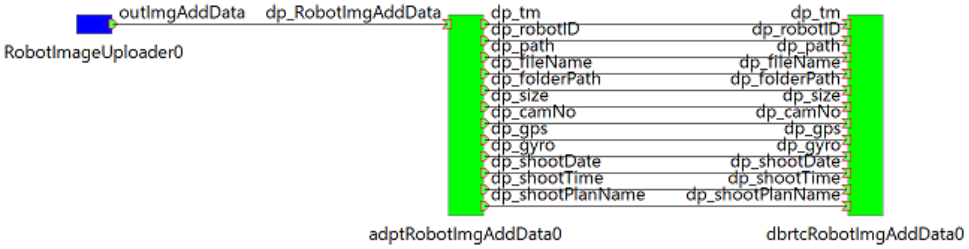


図 6 - 8 転送完了後のシステムダイアグラム画面

7．各種設定

各 RTC が持っているコンフィギュレーションによる調整機能について説明します。

7.1．ドローン I/O RTC

表 7-1 ドローン I/O RTC コンフィギュレーション一覧

名称	データ範囲	デフォルト値	説明
BattCell	2<=x<=7	4	バッテリーのセル数
RTLLowBatt	20<=x<=60	30	バッテリー低下による RTL 実施判断閾値[7]
BattChk	off, on	on	バッテリー容量チェック有無
TakeoffAlt	1.0<=x	10.0	離陸高度(m)
RTLAlt	0<=x<=8000	1500	RTL 高度(cm)
RTLAltFinal	-1<=x<=1000	0	HOME 到達後の高度(cm) 0 は自動的に着陸
WpNavSpeed	0<=x<=2000	500	水平方向移動速度(cm/s) 50cm 刻みで設定
RTLLoitTime	0<=x<=60000	5000	着陸前のホバリング時間(ミリ秒) 1000 ミリ秒刻みで設定
WpNavSpeedDn	0<=x<=500	100	Waypoint 降下速度 ここでは初期着陸速度(cm/s)として利用する 10cm 刻みで設定
LandSpeed	30<=x<=200	50	最終着陸速度(cm/s) 10cm 刻みで設定
DisarmDelay	0<=x<=127	30	自動的に Disarm に切り替えるまでの時間(秒)
SystemBehavior	single, formation	single	本 RTC がどのシステム上で振る舞うか
RobotID	－	-	ロボット識別番号 **必ずデフォルト値以外の値を設定してください**

[7] 搭載バッテリーの S(セル)数及び残電圧値から算出。例えば、4S のバッテリーで残量が 30%未満の場合の閾値は 14V(リポバッテリーは 1S あたり 3.7V。Max 値(4.2V):100%、Min(3.2V):0%)です。

7.2. ドローンコントローラーRTC

表 7-2 ドローンコントローラーRTC コンフィギュレーション一覧

名称	データ範囲	デフォルト値	説明
AltCoefficient	1<=x<=10	1	ターゲット高度算出係数
LatLonCoefficient	2<=x<=10	10	ターゲット経緯度算出係数
LatDiff10m	0.000000<x<0.100000	0.000137	10m あたりの緯度差分
LonDiff10m	0.000000<x<0.100000	0.000108	10m あたりの経度差分
MoveSpeed	1.0<=x<=10.0	3.0	ドローン移動速度
MinAlt	1.0<=x<=100.0	2.0	ドローン最小高度
MaxAlt	10.0<=x<=100.0[8]	15.0	ドローン最大高度
TurnSpeed	0<=x<=360	45	1 秒あたりのドローン旋回速度
MoveTurnSpeed	0<=x<=360	30	1 秒あたりの移動時の旋回速度
NoseLatDiff10m	0.000000<x<10.000000	※[9]	10m あたりの機首緯度差分
NoseLonDiff10m	0.0000000<x<10.0000000	※[10]	10m あたりの機首経度差分

7.3. ドローンマップビューア RTC

表 7-3 ドローンマップビューア RTC コンフィギュレーション一覧

名称	データ範囲	デフォルト値	説明
DroneNum	1<=x<=3	1	ドローン制御台数
MapZoomLevel	0<=x<=19	18	地図の縮尺レベル
VirtualLeader	off, on	off	仮想リーダー機の表示の有無
VirtualLeaderLocus	off, on	off	仮想リーダー機の軌跡表示の有無

7.4. ドローンビューア RTC

表 7-4 ドローンビューア RTC コンフィギュレーション一覧

名称	データ範囲	デフォルト値	説明
AltMaxVal	0<x<=1000	50	ドローン高度

[8] ドローン最大高度(MaxAlt)の設定値は、最小高度(MinAlt)の設定値以上の値とします。

[9] デフォルト値：9.01323x10^(-5)の「9.01323」の部分をデータ範囲とします。

[10] デフォルト値：8.9831566x10^(-5) / cos(lat/2pi)の「8.9831566」の部分をデータ範囲とします。

7.5. ロボット USB カメラ RTC

表 7-5 ロボット USB カメラ RTC コンフィギュレーション一覧

名称	データ範囲	デフォルト値	説明
Interval	0.1<=x	1.0	キャプチャ撮影間隔(単位:秒)
Width	1<=x	1280	画像幅のピクセル数
Height	1<=x	720	画像高さのピクセル数
RobotID	-	-	撮影ロボット識別子
CameraNo	-	1	撮影カメラ識別番号[11]
RootPath	-	/home/pi/	画像および付加データ保存先ルートパス
PlanName	-	TestShot	撮影計画名(半角 30 文字)

7.6. ロボット PRi カメラ RTC

表 7-6 ロボット RPi カメラ RTC コンフィギュレーション一覧

名称	データ範囲	デフォルト値	説明
Interval	0.1<=x	1.0	キャプチャ撮影間隔(単位:秒)
Width	1<=x	3200	画像幅のピクセル数
Height	1<=x	2400	画像高さのピクセル数
RobotID	-	-	撮影ロボット識別子
CameraNo	-	1	撮影カメラ識別番号
RootPath	-	/home/pi/	画像および付加データ保存先ルートパス
PlanName	-	TestShot	撮影計画名(半角 30 文字)
Rotation	0,90,180,270	0	撮影画像の回転

[11] 使用するカメラデバイス番号を事前に確認し、それに+1 した値を設定する。
(例: USB カメラを挿したときに、/dev/video1 が生成された場合のカメラデバイス番号は 1 となるので、CameraNo には 2 を設定する)

7.7. ロボットイメージ uploader RTC

表 7-7 ロボットイメージ uploader RTC コンフィギュレーション一覧

名称	データ範囲	デフォルト値	説明
UploadPath	-	/home/	撮影画像 Upload 先のパス
AddDataFilePath	-	/home/pi/	画像付加データ格納元ルートパス
PictureFilePath	-	/home/pi/	撮影画像データ格納元ルートパス
UserID	-	user	撮影画像 Upload 先のユーザーID
HostName	-	localhost	撮影画像 Upload 先のホスト名(または IP アドレス)

8. エラーメッセージ

エラー発生時、コンソール等にエラーメッセージを表示します。以下は各 RTC のエラーメッセージです。

表 8-1 ドローン I/O RTC のエラーメッセージ一覧

No.	メッセージ	説明
1	Socket error. Please re-startup this RTC.	Vehicle 接続先が存在していません。本システムでは自機固定のため発生しません。
2	Connection Timeout. Please re-startup this RTC.	接続例外エラーです。dronekit ドキュメント[12]によると、稀に発生するとのことですが、一度も発生しませんでした。
3	home_location timeout. Please re-startup this RTC.	ホームポジションのダウンロードに失敗しました。屋外では FC 搭載の GPS モジュールや GPS アンテナが故障していない限り発生しません。
4	Robot ID is not registered!	ロボット ID が登録されていません。登録してください。
5	Does not the home_location. Please re-startup this RTC.	ホームポジションが存在しないまま、RTC を Activate した場合に発生します。No.3 同様、屋外では発生しません。
6	Arm switch timeout. Please re-startup this RTC.	一定期間内に Armed 状態に切り替えられませんでした。再度 Active してください。
7	Communication with the GCS has been lost. Start the RTL by error.	ドローン<->地上局との通信が一定期間(3秒)途絶えた場合に発生します。ホームポジションに帰還します。帰還途中で通信良好となっても帰還動作し続けます。
8	Low battery detected. Start the RTL by error.	バッテリー容量が一定基準を下回った場合に発生します。ホームポジションに帰還します。

[12] http://python.dronekit.io/develop/best_practice.html#connecting

表 8-2 ドローンコントローラーRTC のエラーメッセージ一覧

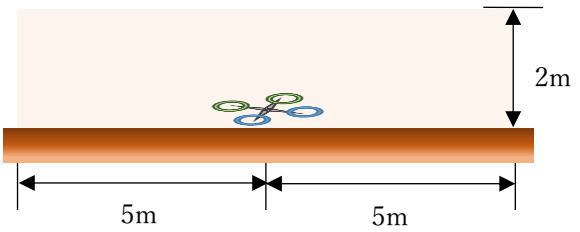
No.	メッセージ	説明
1	ERROR: xxx data is NOT available xxx: GPS または MAG	ドローン I/O RTC から位置情報や磁気コンパスデータが受信できません。ドローン I/O RTC が Active であることを確認してください。
2	WARNING: Conflicting operations are detected	相反する操作要求検知時に発生します。相反する操作要求を解除してください。
3	WARNING: Take-off operation is NOT available in the current drone position	離陸前のポジションが誤差許容範囲内(ホーム～ドローンポジション距離誤差が 5m 以内かつドローンポジションの高度誤差が 2m 以内)である場合は離陸可能ですが、これらの条件以外となった場合はすでに離陸完了とみなします。  GPS 誤差が大きいことで発生していますので、再度キャリブレーションをするか、FC や GPS アンテナそのものの交換が必要です。
4	WARNING: xxx operation is NOT available in the current drone position xxx：操作名称	ドローンポジション(ホームポジションまたはホームポジション以外)に対して、無効な操作が行われています。
5	ERROR: “port_name” is NOT connected	Game Controller RTC とのデータポート未接続時に発生します。

表 8-3 ロボットカメラ RTC のエラーメッセージ一覧

No.	メッセージ	説明
1	ERROR: (*) camera is NOT available (*)USB or RPi	USB(または RPi)カメラが認識できません。正しく接続されていることを確認してください。USB カメラ使用時はコンフィグレーションパラメータ CameraNo の値が適切かを確認してください。
2	ERROR: Creation of save destination folder failed (folder path)	保存先フォルダ作成失敗時に発生します。既に同じフォルダが存在するか、フォルダ書き込み権限があるかを確認してください。
3	WARNING: Failed to save image	画像保存失敗時に発生します。アクセス権限や空き容量が不足していないかどうか等を確認してください。
4	WARNING: Failed to save csv data	付加データ保存失敗時に発生します。アクセス権限や空き容量が不足していないかどうか等を確認してください。
5	Robot ID is not registered!	ロボット ID が登録されていません。登録してください。

9．FAQ

よくある質問を以下に示します。

表 9-1 FAQ 一覧

No.	Q：質問	A：回答
1	コントローラー操作が効きません	以下の原因が考えられます。 ・Game Controller が Active せず Game Controller のコンソールに赤文字でエラーメッセージが出ていますと、PS4 コントローラーが地上局と接続できていないことを示します。接続してもエラーが発生する場合は PS4 コントローラーの故障が考えられます。 ・RTC のデータポート接続ミス 本書に沿って接続してください。 ・PS4 コントローラー接続先アドレスミス 1.6 関連資料 No.4 を参照いただき、PS4 コントローラー接続先アドレスと Game Controller のコンフィグパラメータ(接続デバイス先)が合致していることを確認してください。 ・PS4 コントローラーの電池不足 (=ライトバーが青色点灯せず) 充電が必要です。充電中はオレンジ色にてゆっくり点滅します。 有線接続でもフライト可能ですので、USB ケーブル（USB type-A<-->microUSB type-B）を別途ご用意ください。 ・ArduPilot の状態不定 着陸失敗等により発生するようです。RPi のシャットダウンをし、電源投入し直すことで解消します。

2	All Activate で各 RTC が Active に遷移してから約 30 秒後にドローン I/O RTC が InActive になってしまいます。	ArduPilot の仕様で、Armed 状態から一定時間離陸要求が無い場合、フェールセーフ機能が作動し、disarm 状態に切り替えます。ドローン I/O RTC では Armed->disarm に切り替わったタイミングで InActive に遷移します。よって、**ドローン I/O RTC が Active に遷移したら時間内に離陸操作をしてください**。 時間を変更したい場合は、表 7-1 の DisarmDelay を参照ください。
3	帰還動作開始後に高度が 15m まで上昇しました。	ArduPilot の仕様で、15m(デフォルト値)まで上昇してから帰還動作を開始します。 帰還高度を変更したい場合は表 7-1 の RTLAlt を参照ください。
4	離陸できません	表 8-2 No.3 が原因で発生しています。
5	離陸後すぐに RTL してしまいます	秋冬または寒冷地でのフライトですと、離陸中にバッテリー電圧値が急降下することがあります。表 7-1 のデフォルト設定の場合、14V 未満で低バッテリー検出し、この事象が頻発します。 フライト前にバッテリーを少し温めてから使用するか、BattChk(表 7-1 参照)を off にすることで回避できます。 なお、離陸前や着陸後にはバッテリーチェッカーにてバッテリー残量チェックをお願い致します。
6	画像のサイズが変更した通りの値になりません	お使いのカメラが設定した値の縦横比に対応しているかどうかご確認ください。

著作権