機能仕様書

Lamp Recognition DB

Ver.1.0.0

発行日 2024 年 3 月 31 日

公立大学法人会津大学

株式会社東日本計算センター

改版履歴

Ver	改版日	内容
0.8.0	2023/10/27	新規作成
	2023/11/15	データ構成、RTC 設定ファイル修正
1.0.0	2024/3/31	発行

目次

1. はじめに

本アプリケーションはアクアクルー株式会社開発の「LampDataSub」に DB 関連処理を追加したものとなる。

1.1. 対象読者

本書は RDR(Robot Data Repository)上で動作する遠隔 IoRT システムデータの Subscribe 処理ついて記述した文章であり、OpenRTM-aist や DB に関する知識があることが望ましい。

1.2. 開発環境及び使用機器

開発環境を以下に記載する。

表 1-1 開発環境

言語・環境		バージョン	補足
OS	Ubuntu	20.04 LTS	-
CPU	Intel(R) Core(TM) i7 or Intel(R) Xeon CPU	-	-
開発言語	Python	3.8	-
ミドルウェア	MongoDB	4.4 系	-
	PostgreSQL	14 系	-
	OpenRTM-aist Python	1.2.2	-
依存ライブラリ	pymongo	4.0 系	-
	psycopg2-binary	2.9.3	-
	paho-mqtt	1.6 系	-
	OpenRTM_aist_paho_mqtt_interface	0.6.2	-

1.3. 前提事項/注意事項

本アプリケーション使用にあたっての前提ならびに注意事項を下表に示す。

表 1-2 前提ならびに注意事項

前提事項	(1) 対象となる DB サービスが起動していること (2) MQTT サービスが起動していること
注意事項	無し

2. 構成、静的事項

2.1. モジュール名

本アプリケーション名(RT コンポーネント名)は「Lamp Recognition DB」とする。

2.2. 機能概要

遠隔 IoRT システムから配信されるランプ状態トピックを取得し、指定 DB に書込み要求を行う[図 2-1]。

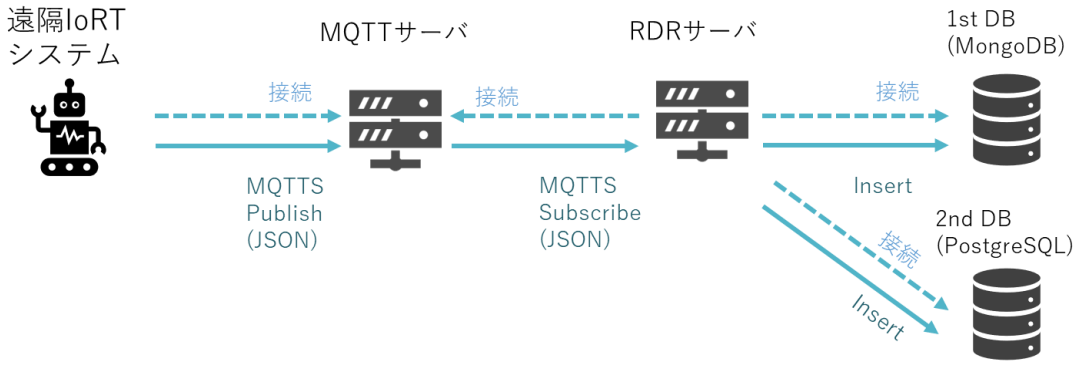

図 2-1 遠隔 IoRT システムとの連携概要

本アプリケーションが Subscribe するトピックを表 2-1、各 Callback 処理を表 2-2、受信したデータを格納する DB テーブル各種を表 2-3～表 2-4、図 2-2 に示す。

表 2-1 Subscribe するトピック一覧

トピック名	説明
robot_project/aquacrew/lamp	ランプ状態通知

表 2-2 各 Callback 処理概要

Callback 名	処理概要
onActivated	DB 接続(※) ※本アプリケーション実行前に、実際の接続先と DB 名/DB テーブル名を指定する(2.5 にて後述)
onExecute	受信したデータを DB に書込み要求
onDeactivated onError onReset onFinalize	DB 切断 onError により Error 状態に遷移した場合のみ、RTC の Reset 処理を行い、Inactive 状態にする。その後に Active にする場合は、手動で RTC を Activate する必要がある。

表 2-3 ランプ情報データ構成(ac_iort_ctrl_panel_lamp コレクション)

項目	説明
camera_id	カメラ ID
datetime	日時(yyyy-MM-dd HH:mm:ss)
lamp_num	ランプ番号
data	ランプ色情報(1:緑, 2:赤, 3:橙)


```
[{"camera_id":"aizu1","datetime":"2023-05-29 13:55:09","lamp_num":1,"data":1},
 {"camera_id":"aizu1","datetime":"2023-05-29 13:55:09","lamp_num":2,"data":2},
 {"camera_id":"aizu1","datetime":"2023-05-29 13:55:09","lamp_num":3,"data":2}]
```


図 2-2 データ構成一例(1 次 DB)

表 2-4 カラム情報(ac_iort_ctrl_panel_lamp テーブル)

No	論理名	物理名	型	Not Null	デフォルト
1	カメラ ID	camera_id	varchar(50)	-	-
2	日時	datetime	timestamp(6) with time zone	-	-
3	ランプ番号	num	bigint	-	-
4	ランプ色	color	bigint	-	-

2.3. 主なエラー

主なエラーを表 2-5 に示す。エラー検出した場合、本アプリケーションを終了する。

表 2-5 エラーメッセージ一覧

No	状態	エラーメッセージ
1	DB 接続エラー	"Unable to connect to NoSQL." もしくは "Unable to connect to RDB"
2	DB 書き込みエラー	"Unable to insert data to NoSQL" もしくは "Unable to insert data to RDB"

2.4. ディレクトリ構成

本アプリケーションのディレクトリ構成を図 2-3 に示す。

```
LampDataSub      -----  Lamp Recognition DB RTC のルートディレクトリ
 ├ LampDataSub.py       -----  Lamp Recognition DB RTC
 ├ mongodb.py           -----  MongoDB 操作クラス
 ├ postgres.py          -----  PostgreSQL 操作クラス
 ├ db_settings.json     -----  DB 設定ファイル
 └ rtcSub.conf          -----  RTC 設定ファイル
```

図 2-3 ディレクトリ構成

2.5. アプリケーション実行

(1) 本アプリケーションのルートディレクトリ内の db_settings.json を開く

(2) DB 設定ファイル仕様として、ファイル形式：JSON、改行コード: LF、文字コード：UTF-8 とする。パラメータを表 2-6、記述例を図 2-4 に示す。

表 2-6 DB 設定ファイル仕様

項目	型	説明
host	string	MongoDB の FQDN(もしくは IP アドレス)を指定
port	int	ポート番号を半角数字で指定
db_name	string	データベース名を指定
db_tbl	string	テーブル名を指定
db_user	string	DB ユーザー名を指定
db_pass	string	DB パスワードを指定
db_host_2nd	string	PostgreSQL の FQDN(もしくは IP アドレス)を指定
db_port_2nd	int	ポート番号を半角数字で指定
db_name_2nd	string	データベース名を指定
db_tbl_2nd	string	テーブル名を指定
db_user_2nd	string	DB ユーザー名を指定
db_pass_2nd	string	DB パスワードを指定

```
{
    "db_host":"localhost",
    "db_port":27017,
    "db_name":"your_db",
    "db_tbl":"ac_iort_ctrl_panel_lamp",
    "db_user":"your_user",
    "db_pass":"user_pass",
    "host_2nd":"localhost",
    "port_2nd ":5432,
    "db_name_2nd ":"your_db",
    "db_tbl_2nd ":"ac_iort_ctrl_panel_lamp",
    "db_user_2nd ":"your_user",
    "db_pass_2nd ":"user_pass"
}
```

図 2-4 DB 設定ファイル記述例

(3) rtcSub.conf[1]にて以下の情報を事前設定する(可読性のため改行有り)

```
# 動作周期(単位:Hz)
exec_cxt.periodic.rate: 1
# MQTT 通信モジュールへのパス
manager.modules.load_path:
/usr/local/lib/python3.8/dist-packages/OpenRTM_aist_paho_mqtt_module
# MQTT 通信モジュール名
manager.modules.preload:InPortPahoSubJsonSecure.py
# RT コンポーネントの自動 Activate 化
manager.components.preactivation:LampDataSub0
# MQTT Broker への InPort(ポート名:in)の自動接続
manager.components.preconnect:LampDataSub0.in?interface_type=mqtts_json&data_type=Timed
String&host=(ホスト名)&msport=(ポート番号)&topic=(トピック名 表 2-1 参照)&
cacert=(認証局証明書へのパス)&
cltcert=(クライアント証明書へのパス)&cltkey=(クライアント秘密鍵へのパス)
```

図 2-5 RTC コンフィグ設定(MQTT 通信モジュール)

(4) ターミナルを起動して、本アプリケーションのルートディレクトリに移動し、本アプリケーションを実行する(図 2-5 は OpenRTM ワークスペース直下に配置した場合の例)
(3)を指定して実行することで MQTT Broker への接続が完了した状態へと遷移する

```
$ cd ~/(OpenRTM ワークスペース)/LampDataSub
$ python3 LampDataSub.py -f rtcSub.conf
```

図 2-6 アプリケーション実行

(5) RT System Editor にて、Lamp Recognition DB RTC が Active 状態であることが確認できる

[1] rtc.conf はコンポーネントを管理するミドルウェアのための設定ファイル

3. 振る舞い、動的事項

3.1. 処理フロー

本アプリケーションの処理フローを以下に掲載する。

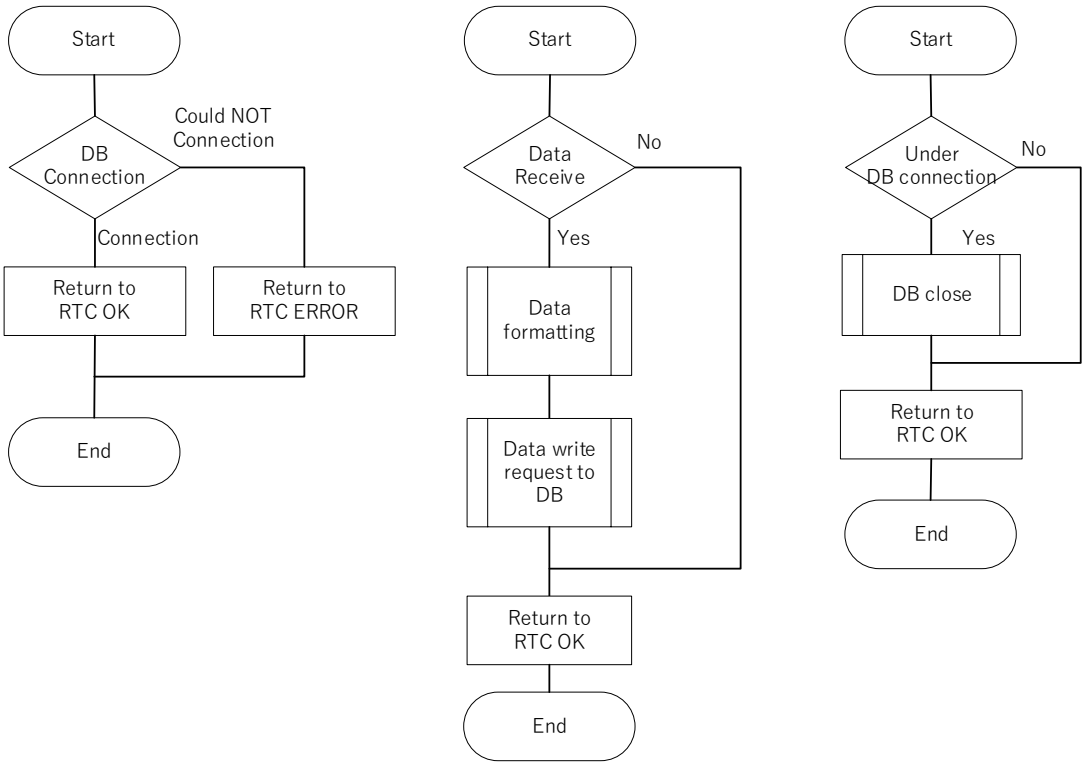

図 3-1 処理フロー(DB 接続(左) DB 書込み(中央) DB 切断処理(右))

著作権