冗長性利用モジュール

1. はじめに

冗長性利用モジュールで提供したいモーション機能は下記のとおりです.

[1] CP モーション機能（冗長軸の動きは自動判定）
[2] 使用する冗長軸を指定した CP モーション機能（肘のみ，肘＋腰など）
[3] HOLD モーション機能（自アーム動作により腰軸，移動軸が動いても他アームの位置・姿勢を維持する）
[4] HOLD モーション機能（制御点の位置・姿勢を維持したまま，指定した冗長軸を動作させる，肘や腰の入れ替えに使用）
[5] CP モーション＋冗長軸モーション機能（制御点と冗長軸を同時に動作させる）

冗長性利用モジュールでは上記のモーション機能を実現するために，下記の解法アルゴリズムを実装しています.

- ノルム最小解冗長性解法アルゴリズム
- 冗長軸指定冗長性解法アルゴリズム

上記の解法アルゴリズムはどちらも片腕のみの解法と双腕同時解法を実装しています. 各モーション機能はサービスポートから目的に応じて簡単に切り替えることが可能です.

冗長性利用モジュールはロボットの移動方法については未実装です．ロボットアームの各関節を目的角度に実際に駆動するのは，サンプルシミュレーションの場合は関節移動モジュール，実機のロボットの場合はモータコントローラが行うことを想定しています.

2. RTC 構成

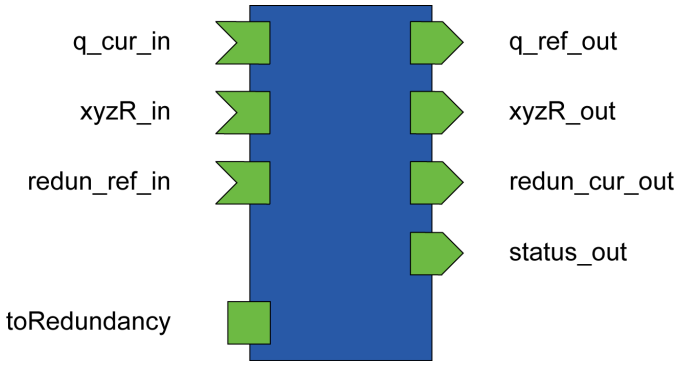

図 1　冗長性利用モジュール（RedundancySolutionCore）

Data InPort

ポート名	データ型	説明	
q_cur_in	TimedDoubleSeq	仮想モデルの現在関節角度・位置 格納順 : 移動 x→移動 y→移動 θ→腰軸→右アーム→左アーム→首軸	
		[0]	移動 x [mm]
		[1]	移動 y [mm]
		[2]	移動 θ [degree]
		[3]	腰軸 rx [degree]（現状ダミー）
		[4]	腰軸 ry [degree]
		[5]	腰軸 rz [degree]（現状ダミー）
		[6] ～ [12]	右アーム J1～J7 [degree]
		[13] ～ [19]	左アーム J1～J7 [degree]
		[20] ～ [22]	首軸 J1～J3 [degree]
xyzR_in	TimedDoubleSeq	アーム手先の目標位置・回転行列 格納順：右アーム→左アーム	
		[0]	右アーム x [mm]
		[1]	右アーム y [mm]
		[2]	右アーム z [mm]
		[3] ～ [11]	右アーム回転行列 (3×3)
		[12]	左アーム x [mm]
		[13]	左アーム y [mm]
		[14]	左アーム z [mm]

		[15] ～ [23]	左アーム回転行列 (3×3)
redun_ref_in	TimedDoubleSeq	冗長軸の目標位置	
		[0]	移動 x [mm]
		[1]	移動 y [mm]
		[2]	移動 θ [degree]
		[3]	腰軸 rx [degree]（現状ダミー）
		[4]	腰軸 ry [degree]
		[5]	腰軸 rz [degree]（現状ダミー）
		[6]	右アーム肘角 [degree]
		[7]	左アーム肘角 [degree]

Data OutPort

ポート名	データ型	説明	
q_ref_out	TimedDoubleSeq	仮想モデルの目標関節角度・位置 格納順は q_cur_in に同じ	
xyzR_out	TimedDoubleSeq	アーム手先の現在位置・回転行列 格納順は xyzR_in に同じ	
redun_cur_out	TimedDoubleSeq	冗長軸の現在位置，格納順は redun_ref_in に同じ	
status_out	TimedDoubleSeq	ステータス情報	
		[0]	逆運動学の解が収束したか否か
		[1]	条件数（最小特異値／最大特異値）

ServicePort

サービス名	データ型	説明
toRedundancy	RedundancyConfig	上位（汎用 RTC）からの指令値

toRedundancy

オペレーション名	説明
resetOriginalFrame	原点フレームを移動部の現在位置でリセット
selectMode	制御対象とするアームの指定, および逆アームの HOLD 指定, 冗長軸指定モードか否か.
selectRedundancy	冗長軸の指定 （肘軸，腰軸（rx，ry，rz 軸），移動（x，y，θ 軸））
setControlPointOffset	制御点のオフセット量を指定（位置，姿勢）
setRedundantAxesLimit	冗長軸の可動範囲を指定 （肘軸，腰軸（rx，ry，rz 軸），移動（x，y，θ 軸））

オペレーション仕様

・ resetOriginalFrame

機能：

原点フレームを移動部の現在位置でリセットする.

宣言：

ReturnID resetOriginalFrame() ;

・ selectMode

機能：

動作モードを指定する.

宣言：

ReturnID selectMod (

in short mode,

in RightLeft type,

in Boolean isHoldOtherArm

) ;

引数：

名前	説明
mode	動作モードの指定 0：CP モーションモード 制御点を指定位置に移動，冗長軸の動作自動判定 1：冗長軸指定モード 冗長軸の動作を指令値として出力（redun_ref_in ポート） 制御点は維持 2：CP モーション&冗長軸指令モード 制御点および冗長軸の両方を指定位置に移動
type	対象が右アームか左アームか，両方のアームかを指定する RightLeft 列挙型の値
isHoldOtherArm	type で指定したアームと反対のアームの動作の種類を指定 *true*：反対のアーム制御点の位置・姿勢を維持 *false*：反対のアームはモーション制御しない

・ selectRedundancy

機能：

動作モードを指定する.

宣言：

ReturnID selectRedundancy (

in RedundantAxesMask redundancy

) ;

引数：

名前	説明
redundancy	使用可能な冗長軸を指定する RedundantAxesMask 構造体型の値

・ setControlPointOffset

機能：

制御点のオフセット量を指定する（位置，姿勢）.

宣言：

ReturnID setControlPointOffset (

in RightLeft type,

in HgMatrix offset

) ;

引数：

名前	説明
type	右アームか左アームか，両方のアームかを指定する RightLeft 列挙型の値
offset	動作目標位置を相対値で指定する HgMatrix 構造体型の値 単位は並進[mm]，回転要素は無次元

・ setRedundantAxesLimit

機能：

冗長軸の可動範囲を指定する.

宣言：

ReturnID setRedundantAxesLimit (

in RedundantLimit limit

) ;

引数：

名前	説明
limit	冗長軸の可動範囲を指定する RedundantLimit 構造体型の値

HgMatrix

概要：

同次変換行列（Homogeneous Transform Matrix）の情報を有する構造体.

定義：

typedef double HgMatrix [3][4]；

備考：

同次変換行列 4×4 の第 4 行を省略した 3×4 の行列．座標系は右手系.

RedundantAxesMask

概要：

冗長軸の指定を有する構造体.

定義：

struct *RedundantAxesMask {*

boolean rightElbow;

boolean leftElbow;

boolean IumbarRx;

boolean IumbarRy;

boolean IumbarRz;

boolean vehicleX;

boolean vehicleY;

boolean vehicleTheta;

};

LimitValue

概要；

上下の制限値を有する構造体.

定義：

struct LimitValue {

double upper;

double lower;

};

RedundantLimit

概要：

冗長軸のリミット値情報を有する構造体.

定義：

```
struct RedundantLimit {
        LimitValue rightElbow;
        LimitValue leftElbow;
        LimitValue IumbarRx;
        LimitValue IumbarRy;
        LimitValue IumbarRz;
        LimitValue vehicleX;
        LimitValue vehicleY;
        LimitValue vehicleTheta;
};
```

ReturnID

概要：

リターン情報を有する構造体.

定義：

```
struct ReturnID {
        long id;
        string comment;
};
```

RightLeft

概要：

アームの左右の種別を示す列挙型.

定義：

```
enum RightLeft {
        RIGHT;
        LEFT;
        DUAL
};
```

3. モーション機能と使用するデータポート，サービスポート

[1] CP モーション機能（冗長軸の動きは自動判定）
[2] 使用する冗長軸を指定した CP モーション機能（肘のみ，肘＋腰など）
[3] HOLD モーション機能（自アーム動作により腰軸，移動軸が動いても他アームの位置・姿勢を維持する）
[4] HOLD モーション機能（制御点の位置・姿勢を維持したまま，指定した冗長軸を動作させる．肘や腰の入れ替えに使用）
[5] CP モーション＋冗長軸モーション機能（制御点と冗長軸を同時に動作させる）

各モーション動作時に使用するオペレーション（サービスポート），データポートについて表に示す．オペレーションについては，〇はモーションに関連する情報，×は指定されても関係しない情報である．データポートについては，〇が入力値を読み込み，および出力するデータであり，×は読み出さないデータである．

type を左/右（LEFT/RIGHT）とした場合

モーション機能 / ポート / オペレーション		[1]	[2]	[3]	[4]	[5]
selectMode	mode（動作モード）	〇 モード 0	〇 モード 0	〇 モード 0	〇 モード 1	〇 モード 2
	type（左/右）	〇	〇	〇	〇	〇
	isHoldOtherArm（逆アームの維持）	〇	〇	〇 （true）	×	×
selectRedundauncy (使用する冗長軸の指定)		〇 （すべて true）	〇	〇	×	×
selectRedundantAxes（冗長軸の可動範囲指定）		〇	〇	〇	×	×
q_cur_in		〇	〇	〇	〇	〇
xyzR_in		〇	〇	〇	〇	〇
redun_ref_in		×	×	×	〇	〇
q_cur_out		〇	〇	〇	〇	〇
xyzR_out t		〇	〇	〇	〇	〇
redun_cur_out		〇	〇	〇	〇	〇

type を両腕（DUAL）とした場合

モーション機能 / ポート / オペレーション		[1]	[2]	[3]	[4]	[5]
selectMode	mode（動作モード）	○ モード 0	○ モード 0		○ モード 1	○ モード 2
	type（両腕）	○	○		○	○
	isHoldOtherArm（逆アームの維持）	×	×		×	×
selectRedundauncy (使用する冗長軸の指定)		○ （すべて true）	○		×	×
selectRedundantAxes（冗長軸の可動範囲指定）		○	○		×	×
q_cur_in		○	○		○	○
xyzR_in		○	○		○	○
redun_ref_in		×	×		○	○
q_cur_out		○	○		○	○
xyzR_out		○	○		○	○
redun_cur_out		○	○		○	○

4. パッケージ内容

冗長性利用モジュールの利用例として，本パッケージでは OpenHRP に付属の GrxUI を利用したサンプルシミュレーションを行うモジュール群も同封しています.

4．1．必要環境

下記の環境でサンプルシミュレーションの動作確認を行っています.

- Ubuntu 10.04 LTS
  - http://www.ubuntu.com/
- OpenHRP3 version 3.0.8
  - http://www.openrtp.jp/openhrp3/jp/
- OpenRTM-aist-1.0.0-RELEASE
  - http://www.openrtm.org/openrtm/ja/node/849/
- Ruby 1.8.7
  - http://www.ruby-lang.org/ja/

（注意点）

ruby は上記の Web サイトからソースパッケージをダウンロードしてインストールする方法の他に，Ubuntu 10.04 では apt-get でもインストールすることが可能です．しかし，Web サイトから行った場合，ruby は /usr/local/bin/ にインストールされるのに対して，apt-get を用いた場合では /usr/bin/ に ruby がインストールされます. したがって apt-get でインストールした場合，本パッケージの redundancy_solution_package/rtc/module_connector/ruby/ にある kill_all_module.rb と startup_module.rb の一行目の記述を下記のように修正してください.

（修正前）#!/usr/local/bin/ruby　→　（修正後）#!/usr/bin/ruby

4．2．ディレクトリ構成

下記のようなディレクトリ構造になっています.

/ redundancy_solution_package/

|-model/	ロボットモデルの VRML ファイル
|-project/	GrxUI のプロジェクトファイル
|-shell/	GrxUI の起動用シェルなど
|-rtc/	rtc モジュール群
|	
|-SmartPal_controller_HG/	SmartPal5 のコントローラ
|-redundancy_solution_core/	冗長性利用モジュール
|-planner/	手先位置指示モジュール
|-trajectory_calculation/	軌道補間モジュール
|	
|-module_connector/	モジュール群の起動&ポート接続&アクティベートをまとめて実行

4．3．コンパイル手順

(1) 環境設定

redundancy_solution_package/に移動し，Make.rules の下記の記述を修正してください.

OPENHRP3_HOME = (使用している OpenHRP3 のトップへの絶対パス)

(2) モジュール群のコンパイル

redundancy_solution_package/に移動し, make コマンドを実行してコンパイルしてください.

```
$ cd redundancy_solution_package /
$ make
```

使用するロボットの VPML モデルは図 5 のような構成となっています．サンプルシミュレーションでは SmartPal5 のモデルを用いています．

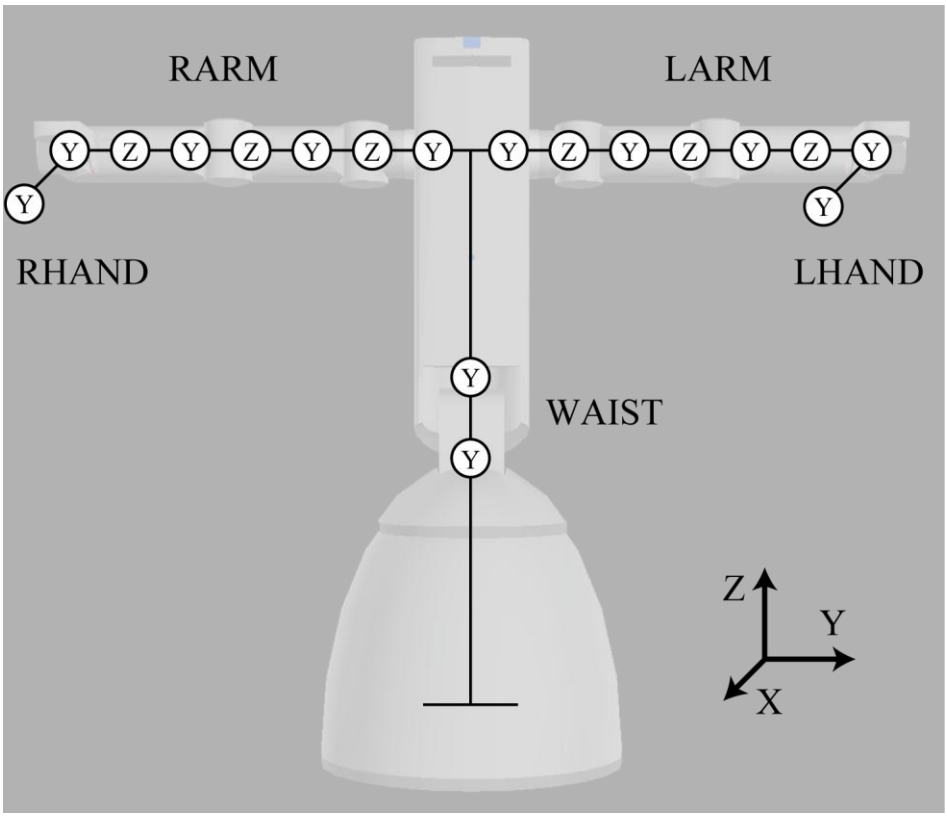

図 2　SmartPal5 モデルの構成

5. サンプルシミュレーション用の各モジュールについて

- シミュレータ（OpenHRP3）

OpenHRP3 に付属の GrxUI で起動されるシミュレータの本体です．実際には GrxUI から起動される ControllerBridge が接続されます．サンプルシミュレーションではハイゲインモードを使用して ControllerBridge と接続しています．

Data InPort

ポート名称	データ型	説明
q_ref_in	TimedDoubleSeq	シミュレータ上のロボットに設定する関節角度
dq_ref_in	TimedDoubleSeq	シミュレータ上のロボットに設定する関節角速度
ddq_ref_in	TimedDoubleSeq	シミュレータ上のロボットに設定する関節角加速度

Data OutPort

ポート名称	データ型	説明
q_cur_out	TimedDoubleSeq	シミュレータ上のロボットの現在関節角度

- 関節移動モジュール（SmartPal_controller）

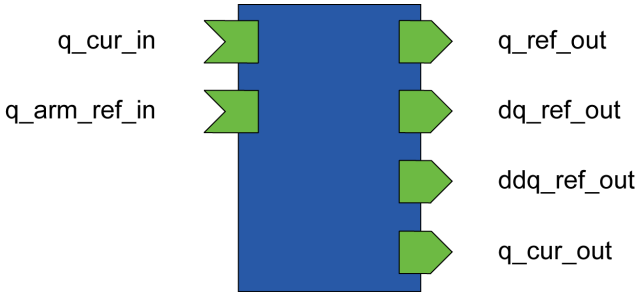

図 3　関節移動モジュール

サンプルシミュレーション用に作成した RT コンポーネントで，SmartPal5 のハイゲインコントローラです．シミュレータから入力された実モデルでの現在関節角度データを仮想モデルでのデータに変換して冗長性利用モジュールに出力し，冗長性利用モジュールから入力された仮想モデルでの目標関節角度データを実モデルでのデータに変換してシミュレータに出力します．

Data InPort

ポート名	データ型	説明	
q_cur_in	TimedDoubleSeq	実モデルの現在関節角度	
		[0]～[1]	腰軸 J1～J2 [rad]
		[2] ～ [8]	右アーム J1～J7 [rad]
		[9] ～ [15]	左アーム J1～J7 [rad]
		[16] ～ [19]	ODV の駆動軸 J1～J4 [rad]
		[20]	グリッパ [rad]
		[21]～[22]	首軸 J1～J2 [rad]
q_arm_ref_in	TimedDoubleSeq	仮想モデルの現在関節角度・位置	
		[0]	移動 x [mm]
		[1]	移動 y [mm]
		[2]	移動 θ [degree]
		[3]	腰軸 rx [degree]（現状ダミー）
		[4]	腰軸 ry [degree]
		[5]	腰軸 rz [degree]（現状ダミー）
		[6] ～ [12]	右アーム J1～J7 [degree]
		[13] ～ [19]	左アーム J1～J7 [degree]
		[20] ～ [22]	首部 J1～J3 [degree]

Data OutPort

ポート名	データ型	説明
q_ref_out	TimedDoubleSeq	実モデルの目標関節角度，格納順は q_cur_in に同じ
dq_ref_out	TimedDoubleSeq	実モデルの目標関節角速度 格納順は q_cur_in に同じで，単位は [rad/s]
ddq_ref_out	TimedDoubleSeq	実モデルの目標関節角加速度 格納順は q_cur_in に同じで，単位は [rad/s^2]
q_cur_out	TimedDoubleSeq	仮想モデルの現在関節角度・位置 格納順は q_arm_ref_in に同じ

- 軌道補間モジュール（TrajectoryCalculation）

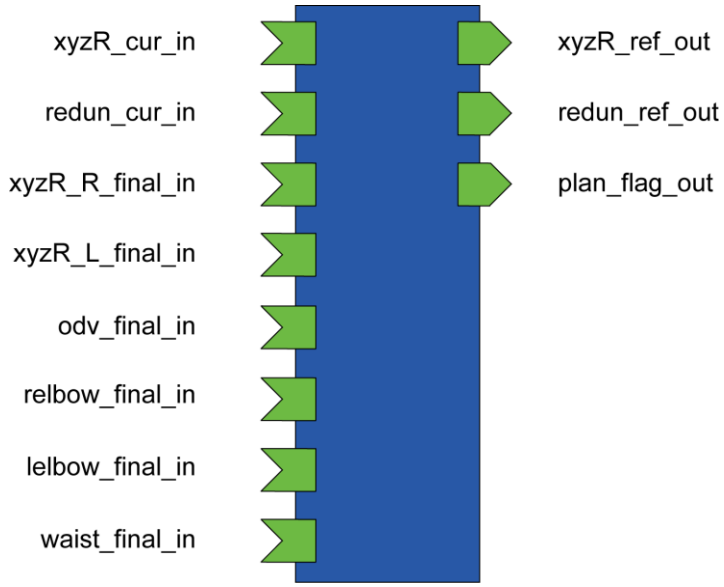

図 4　軌道補間モジュール

サンプルシミュレーション用に作成した RT コンポーネントで，冗長性利用モジュールに対してロボットアームの手先の目標位置を設定するモジュールです．手先位置指示モジュールから新たな手先の最終目標値が入力されると，手先位置・姿勢の現在値から目標値までを五次補間する手先の目標軌道を生成します．生成された軌道の現在時刻での手先の目標値を冗長性利用モジュールに出力します．

Data InPort

ポート名	データ型	説明	
xyzR_cur_in	TimedDoubleSeq	アーム手先の現在位置・回転行列	
		[0]	右アーム x [mm]
		[1]	右アーム y [mm]
		[2]	右アーム z [mm]
		[3] ～ [11]	右アーム回転行列 (3×3)
		[12]	左アーム x [mm]
		[13]	左アーム y [mm]
		[14]	左アーム z [mm]
		[15] ～ [23]	左アーム回転行列 (3×3)
redun_cur_in	TimedDoubleSeq	冗長軸の現在位置	
		[0]	移動 x [mm]

		[1]	移動 y [mm]
		[2]	移動 θ [degree]
		[3]	腰軸 rx [degree]（現状ダミー）
		[4]	腰軸 ry [degree]
		[5]	腰軸 rz [degree]（現状ダミー）
		[6]	右肘角 [degree]
		[7]	左肘角 [degree]
xyzR_R_final_in	TimedDoubleSeq	右アーム手先の最終目標位置・回転行列，時間	
		[0]	右アーム x [mm]
		[1]	右アーム y [mm]
		[2]	右アーム z [mm]
		[3] ～ [11]	右アーム回転行列 (3×3)
		[12]	補間時間 [s]
xyzR_L_final_in	TimedDoubleSeq	左アーム手先の最終目標位置・回転行列，時間	
		[0]	左アーム x [mm]
		[1]	左アーム y [mm]
		[2]	左アーム z [mm]
		[3] ～ [11]	左アーム回転行列 (3×3)
		[12]	補間時間 [s]
odv_final_in	TimedDoubleSeq	移動部の最終目標位置，時間	
		[0]	移動 x [mm]
		[1]	移動 y [mm]
		[2]	移動 θ [degree]
		[3]	補間時間 [s]
relbow_final_in	TimedDoubleSeq	右肘角の最終目標角度，時間	
		[0]	右肘角 [degree]
		[1]	補間時間 [s]
lelbow_final_in	TimedDoubleSeq	左肘角の最終目標角度，時間	
		[0]	左肘角 [degree]
		[1]	補間時間 [s]
waist_final_in	TimedDoubleSeq	腰軸の最終目標角度，時間	
		[0]	腰軸 rx [degree]（現状ダミー）
		[1]	腰軸 ry [degree]
		[2]	腰軸 rz [degree]（現状ダミー）
		[3]	補間時間 [s]

Data OutPort

ポート名	データ型	説明
xyzR_ref_out	TimedDoubleSeq	アーム手先の現在位置・回転行列 格納順は xyzR_cur_in に同じ
redun_ref_out	TimedDoubleSeq	冗長軸の目標位置 格納順は redun_cur_in に同じ
plan_flag_out	TimedDoubleSeq	プログラム間での同期用のトリガ

- 手先位置指示モジュール（Planner）

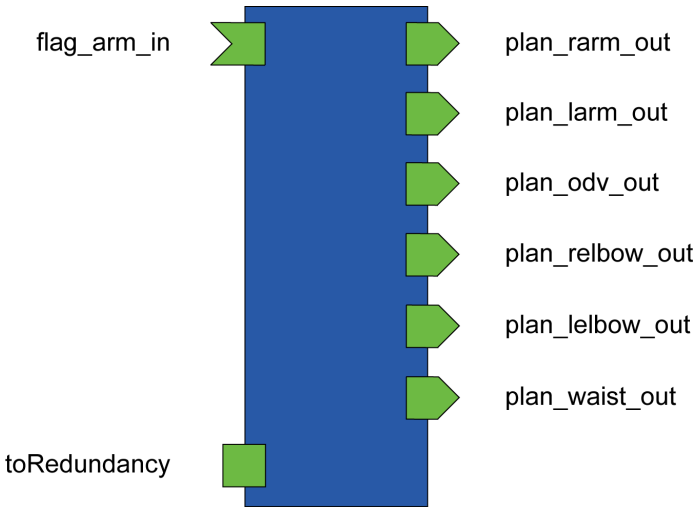

図 5　手先位置指示モジュール

サンプルシミュレーション用に作成した RT コンポーネントで，手先の最終目標位置を指定して軌道補間モジュールに出力するモジュールです．軌道補間モジュールからの入力のタイミングを利用して現在時刻を計算し，任意のタイミングで手先の最終目標位置を変更して軌道補間モジュールに出力します．またサービスポートを用いて冗長性利用モジュールに対して冗長軸指定などの要求を行います．

Data InPort

ポート名	データ型	説明
plan_flag_out	TimedDoubleSeq	プログラム間での同期用のトリガ

Data InPort

ポート名	データ型	説明	
plan_rarm_out	TimedDoubleSeq	右アーム手先の最終目標位置・回転行列，時間	
		[0]	右アーム x [mm]
		[1]	右アーム y [mm]
		[2]	右アーム z [mm]
		[3] ～ [11]	右アーム回転行列 (3×3)
		[12]	補間時間 [s]
plan_larm_out	TimedDoubleSeq	左アーム手先の最終目標位置・回転行列，時間	
		[0]	左アーム x [mm]

		[1]	左アーム y [mm]
		[2]	左アーム z [mm]
		[3] ～ [11]	左アーム回転行列 (3×3)
		[12]	補間時間 [s]
plan_odv_out	TimedDoubleSeq	移動部の最終目標位置，時間	
		[0]	移動 x [mm]
		[1]	移動 y [mm]
		[2]	移動 θ [degree]
		[3]	補間時間 [s]
plan_relbow_out	TimedDoubleSeq	右肘角の最終目標角度，時間	
		[0]	右肘角 [degree]
		[1]	補間時間 [s]
plan_lelbow_out	TimedDoubleSeq	左肘角の最終目標角度，時間	
		[0]	左肘角 [degree]
		[1]	補間時間 [s]
plan_waist_out	TimedDoubleSeq	腰軸の最終目標角度，時間	
		[0]	腰軸 rx [degree]（現状ダミー）
		[1]	腰軸 ry [degree]
		[2]	腰軸 rz [degree]（現状ダミー）
		[3]	補間時間 [s]

ServicePort

サービス名	データ型	説明
toRedundancy	RedundancyConfig	冗長性利用モジュールへの要求

5．3．RT System Editor での接続例

以下では RT System Editor 上で関節移動モジュール，冗長性利用モジュール，軌道補間モジュール，手先位置指示モジュールを接続した例です（図 6）．シミュレータはシミュレーション開始時に GrxUI から自動的に起動・接続されるので，RT System Editor 上には表示されていません．

モジュール名	出力ピン	モジュール名	入力ピン	入出力
SmartPal _controller	q_cur_out	Redundancy SolutionCore	q_cur_in	仮想モデルの現在関節角度
Redundancy SolutionCore	q_ref_out	SmartPal _controller	q_arm_ref_in	仮想モデルの目標関節角度
Redundancy SolutionCore	xyzR_out	Trajectory Calculation	xyzR_cur_in	アーム手先の目標位置･回転行列
Redundancy SolutionCore	redun_cur _out	Trajectory Calculation	redun_cur_in	冗長軸の現在位置
Trajectory Calculation	xyzR_ref_out	Redundancy SolutionCore	xyzR_in	アーム手先の目標位置･回転行列
Trajectory Calculation	redun_ref _out	Redundancy SolutionCore	redun_ref_in	冗長軸の目標位置
Trajectory Calculation	plan_flag_out	Planner	flag_arm_in	プログラム間での同期用のトリガ
Planner	plan_rarm _out	Trajectory Calculation	xyzR_R _final_in	右アーム手先の最終目標位置・回転行列，時間
Planner	plan_larm _out	Trajectory Calculation	xyzR_L _final_in	左アーム手先の最終目標位置・回転行列，時間
Planner	plan_odv_out	Trajectory Calculation	odv_final_in	移動部の最終目標位置, 時間
Planner	plan_relbow _out	Trajectory Calculation	relbow _final_in	右肘角の最終目標角度, 時間
Planner	plan_lelbow _out	Trajectory Calculation	lelbow _final_in	左肘角の最終目標角度, 時間
Planner	plan_waist _out	Trajectory Calculation	waist_final_in	腰軸の最終目標角度，時間

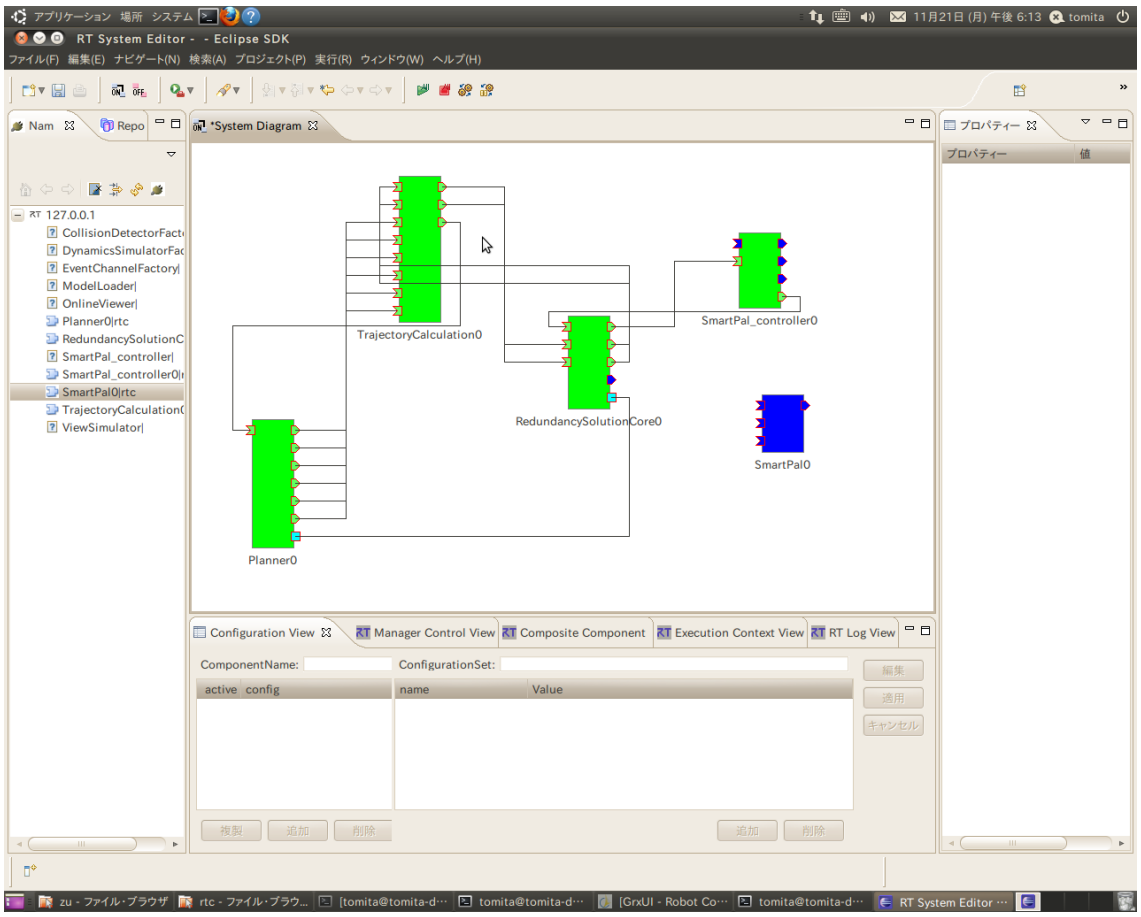

図 6　RT System Editor での接続例

6. シミュレーション手順

(1) OpenHRP3 の起動

OpenHRP3（GrxUI.sh）を起動させます．デフォルトの GrxUI.sh を使うと付属のプロジェクトを読み込む際にモデルとコントローラが正しく読み込めないため，redundancy_solution_package/shell/に移動し，コンソール上で./gui.sh コマンドを実行することで OpenHRP3 を起動してください．

```
$ cd redundancy_solution_package/shell/
$ ./gui.sh
```

シェルスクリプトの説明は redundancy_solution_package/shell/ の README.txt を参照してください．

(2) プロジェクトファイルのロード

GrxUI でプロジェクトファイルを読み込んでください．サンプルシミュレーション用のプロジェクトファイルは redundancy_solution_package/project/にある SmartPal5.xml を読み込んでください．

(3) コンポーネントの接続と活性化

redundancy_solution_package/rtc/module_connector/に移動し，make run コマンドを実行します．これで各コンポーネントの接続と活性化が行われます．module_connector のコンソール上に” --------- Redundancy on Activated ---------”と表示されたらシミュレーション開始してください（図 7）．途中で警告等が出てきますが，無視しても問題ありません．

```
$ cd redundancy_solution_package/rtc/module_connector/
$ make run
```

module_connector がうまく make されない場合は一度各コンポーネントを make してから再び試してみてください．

シミュレーションを開始すると，”Controller 'SmartPal_controller' may already exist. Restart it ?”とメッセージが出ますので [No] を選択するとシミュレーションが開始されます．再度シミュレーションを行う場合は module_connector のコンソール上で一度 make kill した後に，make run してください．

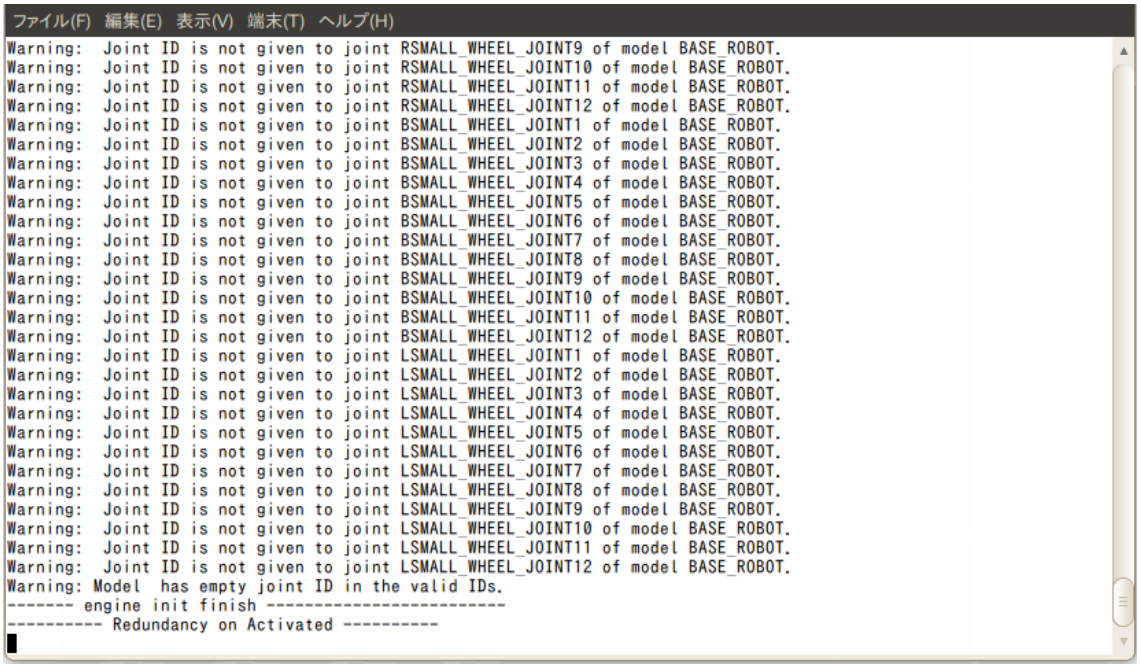

図 7　module_connector のコンソール