メカトロニクス基礎実験/知能機械工学基礎実験
3. ロボットの基礎

1. 実験機器

1.1 ロボットアーム RV-2SD

本実験では，三菱電機製のロボット RV-2SD を使用する．本ロボットは，産業用のロボットを教育用途に小型化した製品である (図 1) ．

図 1　ロボットアーム RV-2SD

このロボットは「垂直多関節ロボット」あるいは「6 軸多関節ロボット」と呼ばれる機構をしており，6 つの回転関節を持つ．各関節は接地部分に近い順に第 1，第 2，……第 6 関節と呼ばれるほか，それぞれに固有の呼称も存在する．以下では第 1 関節から順に説明していく．

第 1 関節，つまりロボットの接地部分から一つ目の関節は，ウエスト関節とも呼ばれる (図 2) ．この関節は人間の腰関節に相当する．

図 2　ウエスト関節（第 1 関節）

第 2 関節，つまり設置部分から二つ目の関節は，ショルダー関節とも呼ばれる (図 3) ．人間の肩関節に相当する．

図 3　ショルダー関節（第 2 関節）

第 3 関節はエルボー関節とも呼ばれる (図 4) ．人間の肘関節に相当する．

図 4　エルボー関節（第 3 関節）

第 4 関節はツイスト関節とも呼ばれる (図 5) ．人間の肘の捻りに相当する運動を行う．

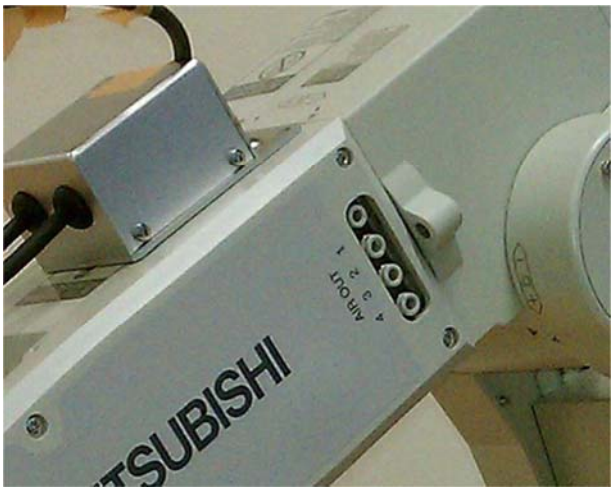

図 5　ツイスト関節（第 4 関節）

第 5 関節はリストピッチ関節と呼ばれる (図 6) . 人間の手首関節のピッチ運動に相当する運動を行う.

図 6　リストピッチ関節（第 5 関節）

第 6 関節はリストロール関節とも呼ばれる (図 7) . 人間の手首関節の回転方向に相当する運動を行う.

図 7　リストロール関節（第 6 関節）

以上の 6 関節の回転を組み合わせることで，ロボットアームの動作を行なっている.

各関節の駆動には AC サーボモータが使用されている. AC サーボモータと DC サーボモータの構造を図 8 に示す. AC サーボモータは入力信号に交流電圧を使用するため，制御が難しいが， DC サーボモータと違いブラシレスでメンテナンスフリーであることから，ロボットアームなどに採用される場面が増えてきている.

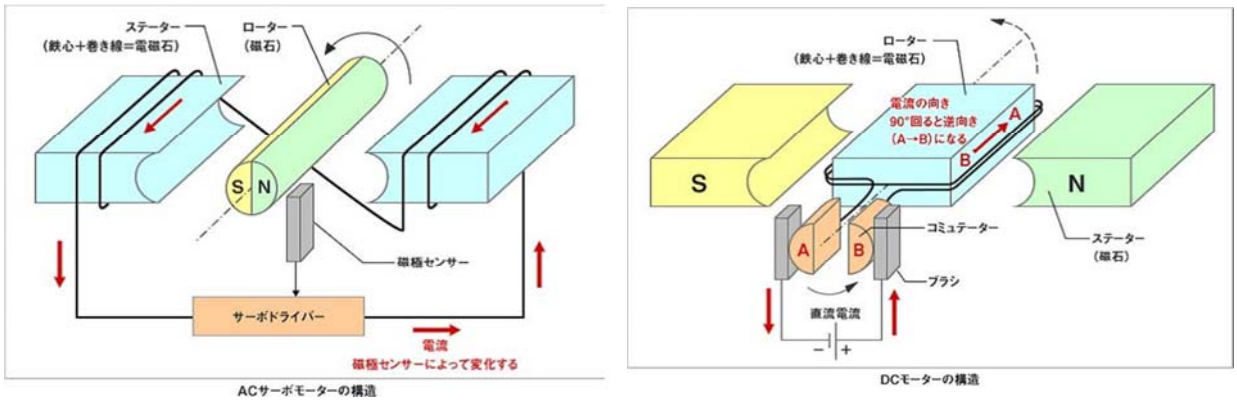

図 8　サーボモータの構造 [1]

関節角度を測定するために，各関節に絶対位置エンコーダ（アブソリュートエンコーダ）が搭載されている．絶対位置エンコーダは相対位置エンコーダ（通常のエンコーダ）と比較すると高価であるが，原点出しが不要であるため，作業開始時の準備段階を減らすことができる．また，関節駆動に対して，ロータリーエンコーダによってフィードバック制御を行なっている．各関節にはブレーキが掛かるようになっており，電源を切ると関節が動かないようになっている．

このような大きな機械を動かす場合，モータで直接駆動しようとすると非常に大きなものを採用しなければ，必要となるトルクを得ることは難しい．そのため，減速機を用いるのが一般的である．特にロボットのような空間的制約のある機器に対して多く利用されている減速機がハーモニックドライブである(図 9)．ハーモニックドライブは歯車を利用した減速機よりも少ないスペースで高減速比を実現することが可能である．

図 9　ハーモニックドライブ [2]

1.2 電動ハンド

ロボットアームに TAIYO 製の電動ハンド ESG1-FT-2840 (図 10) を装着して対象物の把持を行う．このハンドは 2 爪の平行グリッパによって把持を行い，グリッパはモータとボールねじを利用して駆動している．また，ハンドには絶対位置エンコーダではなく相対位置エンコーダが搭載されているため，ハンドはアームと異なり，電源を入れるたびに原点出しの準備動作が必要となる．

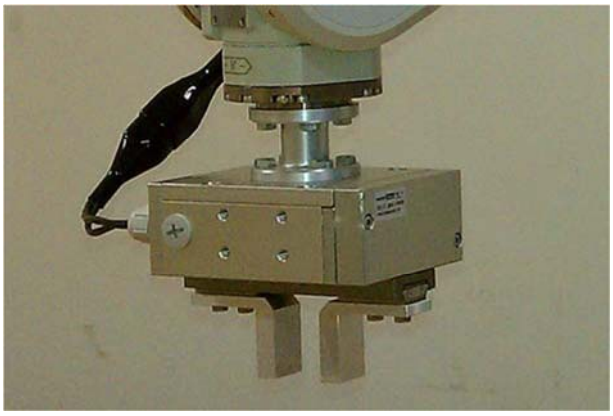

図 10　電動ハンド ESG1-FT-2840

1.3 コントローラ

コントローラはロボットの制御に使用するプログラムを実行し，制御時に必要な各種計算（運動学計算や逆運動学計算など）を行う装置である (図 11) ．本実験で利用するコントローラは TCP/IP 通信でパーソナルコンピュータ（PC）に接続されており，PC との間でプログラムの送受信を行い，プログラムの選択・実行などを PC 側から行うことができる．

図 11　コントローラ

1.4 ティーチングボックス

ティーチングボックス（ティーチングペンダント）は，ロボットアームに座標を教示するための装置である (図 12) ．本実験で使用するものは高機能で，教示の他に電動ハンドの制御や外部入力のデバッグ，プログラムの編集や実行も可能である．

図 12　ティーチングボックス

1.5 直線補間と関節補間

ロボットの動作には直線補間と関節補間の 2 つがある．直線補間とは，実空間（手先位置のxyz軸と，手先姿勢を表すロール・ピッチ・ヨーの 3 軸をまとめたもので表される）においてロボットの手先を直線的に移動するものである．一方，関節補間とは，関節角空間（6 つの関節それぞれの回転角を軸とするような 6 次元空間で表される）の上でロボットを直線的に動かすものである．

直線補間動作は，ユーザにとっては実際の手先軌道を想像しやすいが，直線補間動作のロボットの軌道は関節角空間上で見ると迂回した曲線経路となっており，関節補間動作に比べると高速性に劣る．また直線補間動作は，始点座標と終点座標の与え方によっては，ロボットの実行可能な動作経路が生成できない場合や，ハンドとロボット本体が自己干渉するような経路が生成される場合もある．

関節補間動作は，高速な動作に適しているが，実空間上でのロボットの手先軌道は直線ではなく曲線となるため，ユーザにとってはロボットの動作を想像しにくい．特に現実のロボットには各関節に関節角限界があるため，始点座標と終点座標の与え方によっては，関節補間動作によってユーザがまったく予想しなかったような軌道が生成されることもある．

2 つの動作は適宜使い分けられる．例えば部品の組み付けや部品のピッキングなどでは，水平作業面に対して手先を鉛直方向に直線的に動かすことが必要であるため，直線補間動作が用いられる．また障害物を避けながら手先を動かすことが必要な場合（例えば自動車ボディ内部のスプレー塗装のため，ロボットの手先を自動車の窓から車内に入れるような場合）にも直線補間動作が用いられる．障害物がなくアームを高速に動かしたい場合（例えば右側に伸ばしていたアームを素早く左側へと振るような場合）には，関節補間動作の方が適している．

2. ロボット言語

RV-2SD で用いられるロボット言語の「MELFA-BASIC V」について簡単に説明する．MELFA-BASIC は，古典的なコンピュータ言語の「BASIC」に，ロボットの位置や動作に関連した変数・制御命令・ハンド命令・入出力制御命令などを追加して拡張したものである．

2.1 変数型と特別な文字

変数型には数値型，文字列型，位置型，関節型がある．数値型（整数，実数）と文字列型は一般のプログラミング言語にも存在する変数型であるが, 位置型と関節型は MELFA-BASIC などロボット言語に特有の変数型である．

位置型変数は 6 次元ベクトルであり，ロボットの手先（ハンドの先端ではなく，ハンドを取り付ける手首の部分）の 3 次元の実空間における位置・姿勢を $(x, y, z, \theta, \phi, \psi)$ という形式で表す．x, y, z が位置を，θ, ϕ, ψ が姿勢をそれぞれ意味する．

関節型変数も位置型変数と同様に 6 次元のベクトルである．ただし関節型変数では，6 つの回転関節のそれぞれの角度 $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ によってロボットの姿勢を表現する．

MELFA-BASIC では，変数の型宣言をする必要はなく，変数名の最初の文字によってその変数の型が自動判別される．まず変数名の最初に数字は使用できない（例：1a, 2X などの変数名は不可）．P ではじまる変数名（例：P1, P2, PSTART など）は位置型変数，J ではじまる変数名（例：J3, J4, J-FINISH など）は関節型変数，C ではじまり $ で終わる変数名（例：C1$, C2$, C_MESSAGE$など）は文字列型変数としてそれぞれ扱われる．これら以外の文字ではじまる変数はすべて数値型変数と見なされるが，数値型変数が整数型なのか実数型なのかは，その変数に最初に代入された値によって自動判別される．

2.2 演算子（比較，四則演算など）

MELFA-BASIC には“=”（等号または代入），“<>”（不等号≠），“>=”（不等号≧）などの比較演算子や，四則演算の“+”（足し算），“-”（引き算），“*”（掛け算×），“/”（割り算÷）などの演算子が定義されている．また“Mod”は剰余を求める演算子で，整数同士の割り算をした時の余りを求めることができる．

```
例　1　M1=7                 ’変数 M1 に 7 を代入
　　2　 M1=M1+2                 ’M1 + 2の値を求め，その値を変数 M1 に上書きする
　　3　 M2=7-6 / 2              ’変数 M2 に 7 − (6 ÷ 2) の値を代入
　　4　 M3=M1*M2 / 6            ’変数 M3 に (M1 × M2) ÷ 6 の値を代入
　　5　 M4=M3　Mod　4           ’変数 M4 に，変数 M3 を 4 で割った余りを代入
```

上記のステップ（行番号）1～5 を実行した後の各変数の値は，M1 = 9，M2 = 4，M3 = 6，M4 = 2 となる．

2.3 基本命令

Wait：条件が成立するまでプログラムをその行で待機させる

```
例　1　Wait ＜数値変数＞=＜数値定数＞
```

For ～ Next：繰り返し

```
例　1　MSUM=0                   ’変数 MSUM を 0 に初期化
　　2　For M1=1 To 5            ’数値変数 M1 を 1 から 5 まで+1 ずつ増加させながら反復
　　3　　MSUM=MSUM+M1           ’数値変数 MSUM に M1 の値を足す
　　4　Next M1                  ’「For M1=……」の行に戻る
```

上記の繰り返しを終えた時，MSUM の値は 15 となる．

If ～ Then ～ Else ～：条件分岐

```
例　1　If M1<=5 Then            ’数値変数 M1 の値が 5 以下ならば
　　2　　M2=M1                  ’変数 M2 に M1 を代入する
　　3　Else                     ’M1 が 5 より大きい場合は
　　4　　M2=M1*0.5              ’M2 にはM1 × 0.5を代入する
　　5　Endif                    ’IF 文おわり
```

「～でない場合」の処理を指定する必要がなければ，3 行目と 4 行目は省略可能.

GoTo：無条件分岐

```
例　1　GoTo　*LBL
　　　　⋮
　　　　⋮
　　8　*LBL
```

指定先（対応するラベルのついた 8 行目）へ無条件で分岐する．

※MELFA-BASIC では GoTo などの飛び先にステップ番号（行番号）を指定できないため，飛び先のステップ番号の後にラベルを付ける必要がある．ラベルの最初には文字「*」が必要.

End：プログラムの終了

例　　End

2.4 移動命令

Mvs：直線補間で指定位置へ移動する

例　　Mvs　PGET
ワークをつかむ位置（PGET）へ直線補間移動

Mvs　PGET, -20
ワークをつかむ位置（PGET）の上空 20mm へ移動

直線補間とは，位置型変数$(x, y, z, \theta, \phi, \psi)$の空間（実空間）上で直線的にロボットの手先を移動させることを意味する．直線補間については 1.5 節，位置型変数については 2.1 節も参照のこと．

Mov：関節補間で指定位置へ移動する

例　　Mov　PGET
ワークをつかむ位置（PGET）へ関節補間移動

Mov　PGET, -20
ワークをつかむ位置（PGET）の上空 20mm へ移動

関節補間とは，関節型変数$(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$の空間（関節角空間）上で直線的にロボットの手先を移動させることを意味する．関節補間については 1.5 節，関節型変数については 2.1 節も参照のこと．

関節角空間上での直線移動は，実空間上では曲線的な移動となることに注意．

2.5 速度設定

Spd：直線補間時の速度を数値（mm/s 単位）で指定する

例　　Spd　30
速度を 30mm/s に設定する．

Dly：待ち時間の指定，及び出力信号のパルス出力時間を指定（0.01s 単位）

例　　Dly　1.0
1 秒間待つ．

2.6 割り込み処理

M_In：入力信号のオン・オフ値を返す

例　　M1%=M_In(3)
整数型変数“M1%”に入力信号 3 の値（0 か 1）を代入する．

Def Act：割り込みの条件とその処理を記述する

例　　Def Act 2, M_In(3)=1 GoTo *LBL
優先番号 2 の割り込み処理を以下のように定義する.
「入力信号 3 の値が 1 になったら，ラベル“*LBL”の行にジャンプせよ」

Act：割り込みを許可/禁止する

例　　Act 2=1
優先番号 2 の割り込み処理（“Def Act 2…”で定義された割り込み処理）を許可

例　　Act 2=0
優先番号 2 の割り込み処理を禁止

2.7 ハンド制御命令

EHOrg：ハンドの初期化と原点復帰

例　　EHOrg 1
ハンド 1 を初期化して原点復帰させる

EHclose：ハンドを閉じる

例　　EHclose 1, 40, 30
ハンド 1 を 40%の速度と 30%の力で閉じる

EHopen：ハンドを開く

例　　EHopen 1, 35, 30
ハンド 1 を 35%の速度と 30%の力で開く

ハンド開閉の例文は以下の通り.

```
1  If M_EHOrg=0 THEN          'original point return incomplete state
2      EHOrg 1                'hand 1 original point return
3      Wait M_EHOrg=1         'hand 1 original point return completion wait
4  EndIf
5  EHClose 1, 50, 30          'hand 1 close (speed=50%, force=30%)
6  Wait M_EHBusy=0            'wait for hand operation completion
7  EHOpen 1, 50, 30           'hand 1 open (speed=50%, force=30%)
8  Wait M_EHBusy=0            'wait for hand operation completion
```

3. 実験内容

3.1 パレタイズとは

パレタイズとは等間隔に整列された容器 (パレット) に対象となる物体 (ワーク) を指定した通りの順番で，適当な位置に移動させ，置いていくという作業のことをいう．日常生活で触れるものでは卵のパックなどがパレットの代表例として挙げられる (図 13) ．このような製品は当然，入っているべき個数が，入っているべき場所に収まっていることが重要となる．よって，パレタイズ作業を正確に行えるということは，製品を製造する場面で重要な要素であると言える．

図 13　卵パック

今回の実験に使用するパレットを図 14 に示す．

図 14　パレット

3.2 実験の流れ

本実験は以下のような流れでロボットアームに作業を行わせる.

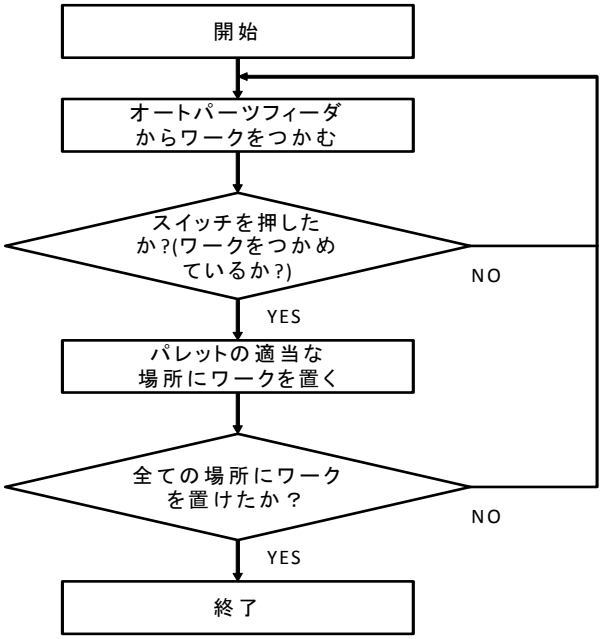

図 15　作業の流れ

オートパーツフィーダー (自動部品供給器) とは，ワークを同じ場所に送り続けることで，教示する座標の点数を減らすことができるようにする装置のことである．今回の実験に使用するものを図 16 に示す.

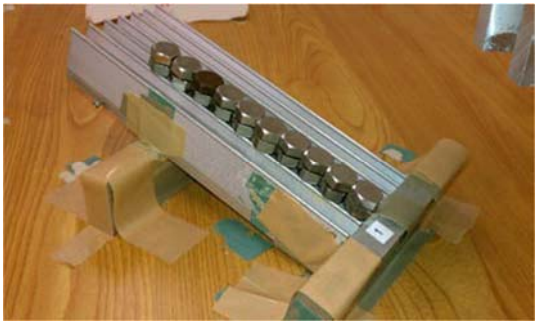

図 16　オートパーツフィーダー

今回はワークの把持ができているか確認するためのデバッグを，図 17 に示すスイッチを利用することで行う.

図 17　スイッチ

4. 実験手順

4.1 教示

ロボットにパレタイズ作業を行わせるために，まずティーチングボックス（TB）を用いて，各ポジションの教示を行わなければならない．

本実験で使用する装置は，プログラムごとにポジションの名前を設定し，各ポジションに対して教示を行う．よって，教示を行う前にプログラムを用意し，TB でそのプログラムを開く必要がある．今回の実験では，事前に用意した基本的なパレタイズ作業を行うプログラム”sample2.prg”に対して教示を行なっていく．

［注意事項］

1) ロボットに殴られないように！
   * 可能な限りロボットの可動範囲に入らない
   * どうしても動作範囲に入る場合，ロボットの動きに細心の注意を払うこと
2) ロボットやスイッチを壊さないように！
   * 壁やテーブルにぶつけないこと
   * スイッチを押すとき，少しずつ下ろす
3) 事前に TB およびコントローラの緊急停止ボタンを確認しておき，ロボットを動かすときはいつでも押せるようにすること

I. 教示作業を行うための準備

まずコントローラの電源を入れ，MODE 切替スイッチを”MANUAL”にする．次に TB の”[TB ENABLE]スイッチ”を押し，点灯する状態にする．“[EXE] キー”を押し，メインメニューが表示されたら，カーソルを”1. 管理・編集”に合わせ，”[EXE] キー”を押して選択する．”SAMPLE2”にカーソルを合わせ，“編集”を選択する．なお，画面下部に表示されるメニュー (ファンクションメニュー) を選択する際は，“F1～4”キーを使用する．プログラムが開けたら，“FUNCTION キー”を 2 回押してファンクションメニューを切り替え，“切替”を選択する．これで，“＜位置＞モード”が開かれるので，教示したいポイントまでアームを移動させ，ファンクションメニューを一回切り替えて，“教示”を選択することで教示が行える．

II. ロボットアームの動かし方

TB 背面の “イネーブルスイッチ” を押しながら，“[SERVO] キー”を押しサーボモータの電源を入れる．なお，“イネーブルスイッチ”は 3 段階スイッチになっていて，離しているときと，強く押し込んだときにサーボの電源がオフになるようになっている．次に“[JOG] キー”を押し，ジョグモードに切替える．ジョグモードは初期状態では関節動作モードになっているので，直行動作モードに切替える．この状態で“[ジョグ操作] キー”を操作することでロボットアームを動かすことができる．

III. ハンドの動かし方

“[HAND] キー”を押すことでハンド操作モードとなる．“原点”を原点のチェックボックスにチェックが入るまで長押しし初期化を行う．初期化が終われば“ハンド開”，“ハンド閉”でハンドを操作することができる．

IV. 教示の位置

ポジション 1：ボルトをつかむ位置

① TB でボルトをつかめる位置までロボットのハンドを動かす．そのとき，ボルトを引き上げる時のアームの姿勢を考慮して，位置を決める．
   注意：ハンドの中心で，ボルトがつかめるように位置を決める．
② ハンドを開いたままポジションを教示する．

③　ハンドを閉じ，ボルトをつかむ.

④　持ち上げる.

ポジション 3：スイッチを押す位置

①　ボルトをつかんだ状態で，ボタンスイッチを押せるような位置を決める．ボタンスイッチが押されるとLED が点灯されるので，それを確認しながら行う.

②　ポジションを記憶させる.

③　ボルトを上に回避させる.

ポジション 2：ボルトを落とす位置

①　パレットにある 10 個の穴の位置をロボットに教えるためには, 4 隅の上でボルトを落とせる位置(ポジション番号 20, 21, 22, 23)を教示すればよい. 教示する際, ハンドの高さはパレットの縁よりも少し高くする.

図 18　TB のボタン配置

4.2 実行

I.　制御用プログラム“RT　ToolBox2”を起動する.

図 19　デスクトップ画面

II.　ワークスペース“test3”を開く.

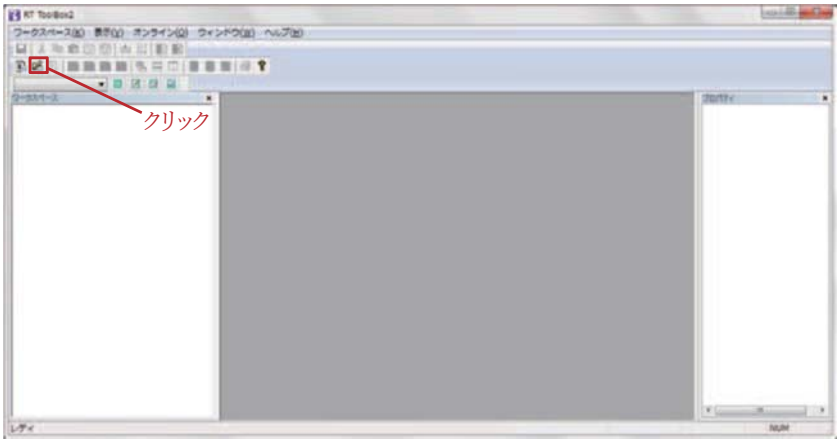

図 20　アプリケーション画面

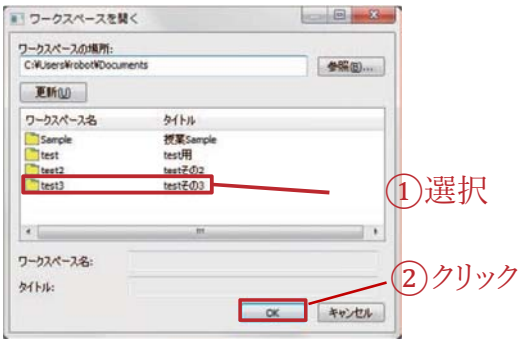

図 21　プロジェクト選択画面

III. コントローラのモードが“AUTOMATIC”になっているか確認し，オンラインモードに切替える．

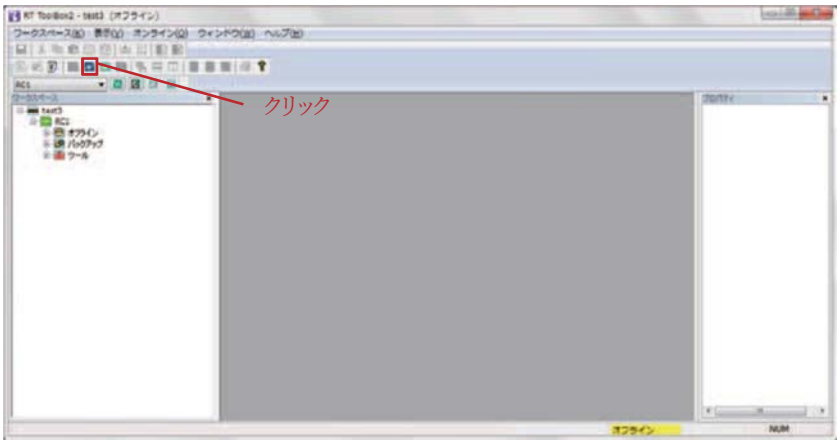

図 22　アプリケーション画面

IV. オンラインのディレクトリ（コントローラ内のデータ）を開く．

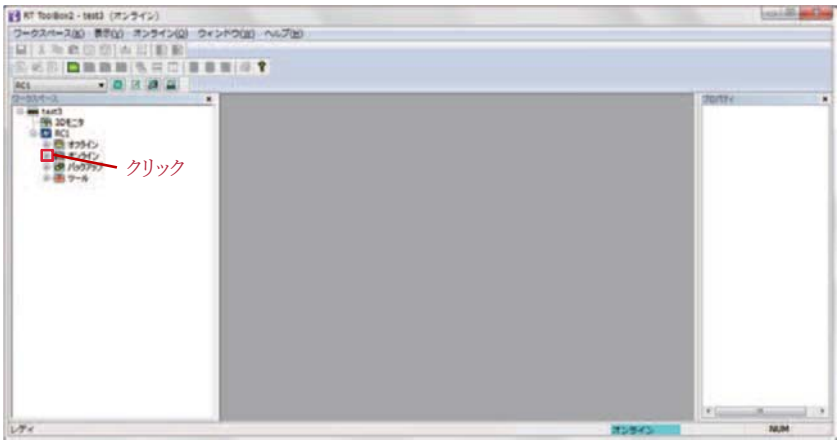

図 23　アプリケーション画面

V. 実行するプログラム上で右クリックし，“デバッグ状態で開く”を選択する．

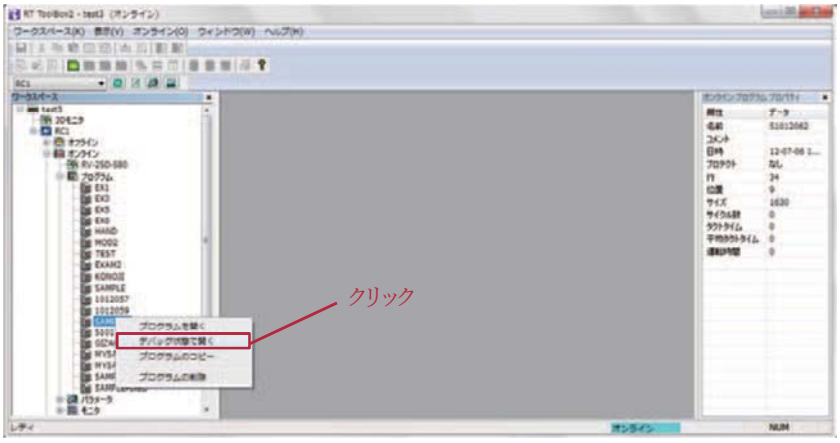

図 24　アプリケーション画面

VI. サーボの電源を入れ，プログラム実行ボタンを選択する．

図 25　デバッグ画面

4.3 プログラム

教示作業が終了したら，例として図 26 のような順序でパレタイズ作業を行う．

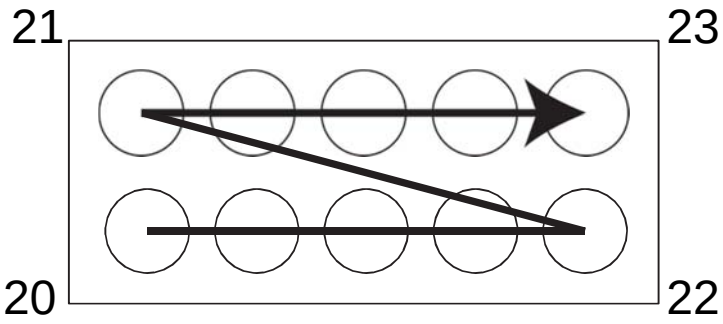

図 26　パレタイズ順序：例題

例題に使用したプログラムから，27-31 行目を消去したものを配布するので，テキストを参考にプログラムの頭から読んでいき空白を埋める．

次に，例題のプログラムを改良して，パレタイズを別の順序（図 27 のパターン 1-6 のいずれか，班ごとに指定）で行うようなプログラムを作成する．

配布されたプログラムの編集は，各自の作業用 PC 上のテキストエディタで行い，プログラムの実行はロボット制御用の PC で行う．作業用 PC とロボット制御用 PC のデータの移し替えには USB メモリを用いること．

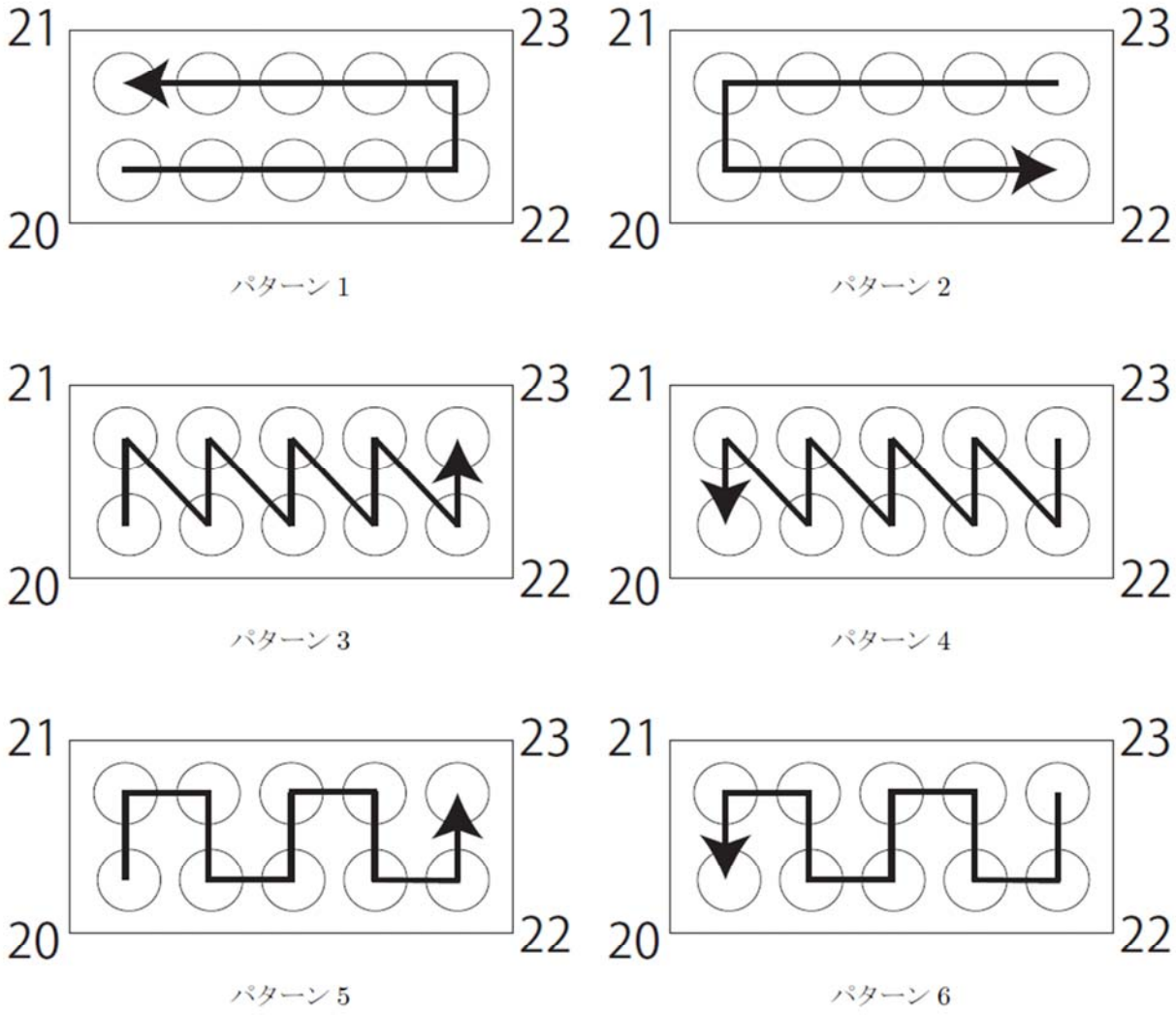

図 27　パレタイズ順序：課題

レポート課題（テキストに記載の課題ではなく，こちらを行うこと）

① 実験の目的

② 実験システム（注：ロボットおよびパレタイズ作業に必要な周辺装置）

③ 実験内容と手順

④ 各自の演習課題のパレタイズ作業順序を図で示し，作成したプログラムにその説明を 1 行ごとに記入する．

⑤ (必須課題) ロボットの位置や姿勢を数学的に求める方法として，運動学がある．運動学には，ロボットの各関節の状態（直動関節ならリンク長，回転関節なら関節角）からロボットの手先の位置・姿勢を求める「順運動学」と，ロボットの取るべき手先の位置・姿勢が与えられた時に，そのような手先位置･姿勢を実現するための各関節の状態を求める「逆運動学」とがある．一般に，順運動学よりも逆運動学のほうが難しい．その理由は，順運動学では各関節の状態からロボット手先の位置・姿勢が一意な解として求まるのに対し，逆運動学では目的とするロボットの手先位置・姿勢から各関節の状態が一意には求まらないためである．
上記の下線部について，図 32 に示すような 2 つの回転関節を持ちxy平面内で動作するロボットアームを用いて具体的にわかりやすく説明せよ．ただし必要に応じて図や式を用いてもよい．また変数なども各自で追加してよい．

⑥ (必須課題) 6 ページ 1.5 節で述べたように，直線補間動作の場合，ロボットの始点座標と終点座標の与え方によっては，ロボットが実行可能な経路を生成できなくなる場合がある．
上記の下線部について，図 32 のロボットアームを想定して実例を一つ示せ．必要に応じて図や式を用いてもよい．また変数なども各自で追加してよい（ヒント：ロボットの手先が到達可能な領域をxy平面で考えるとよい）．

⑦ (任意課題) ロボットの機構と制御法に関する基本事項をまとめよ．

⑧ (任意課題) 本実験またはロボットに関する感想を述べよ（10 行以上）．

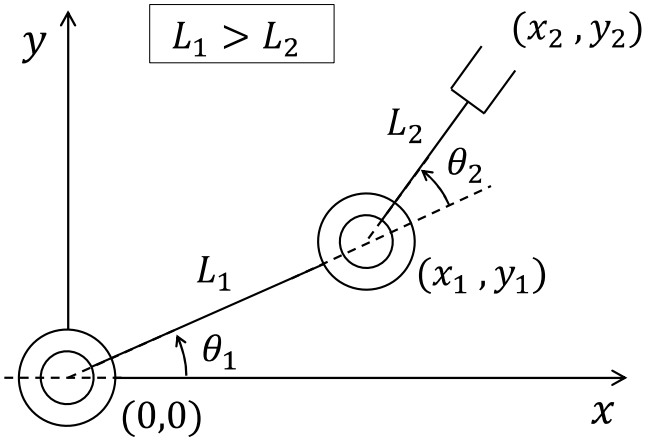

図 32　　2 自由度のロボットアーム

参考文献

[1] ヘイシンロボディスペンサー Web サイト
http://www.robo-dispenser.com/compass/compass08.html
[2] ハーモニックドライブ Web サイト
http://www.hds.co.jp/products/hd theory