

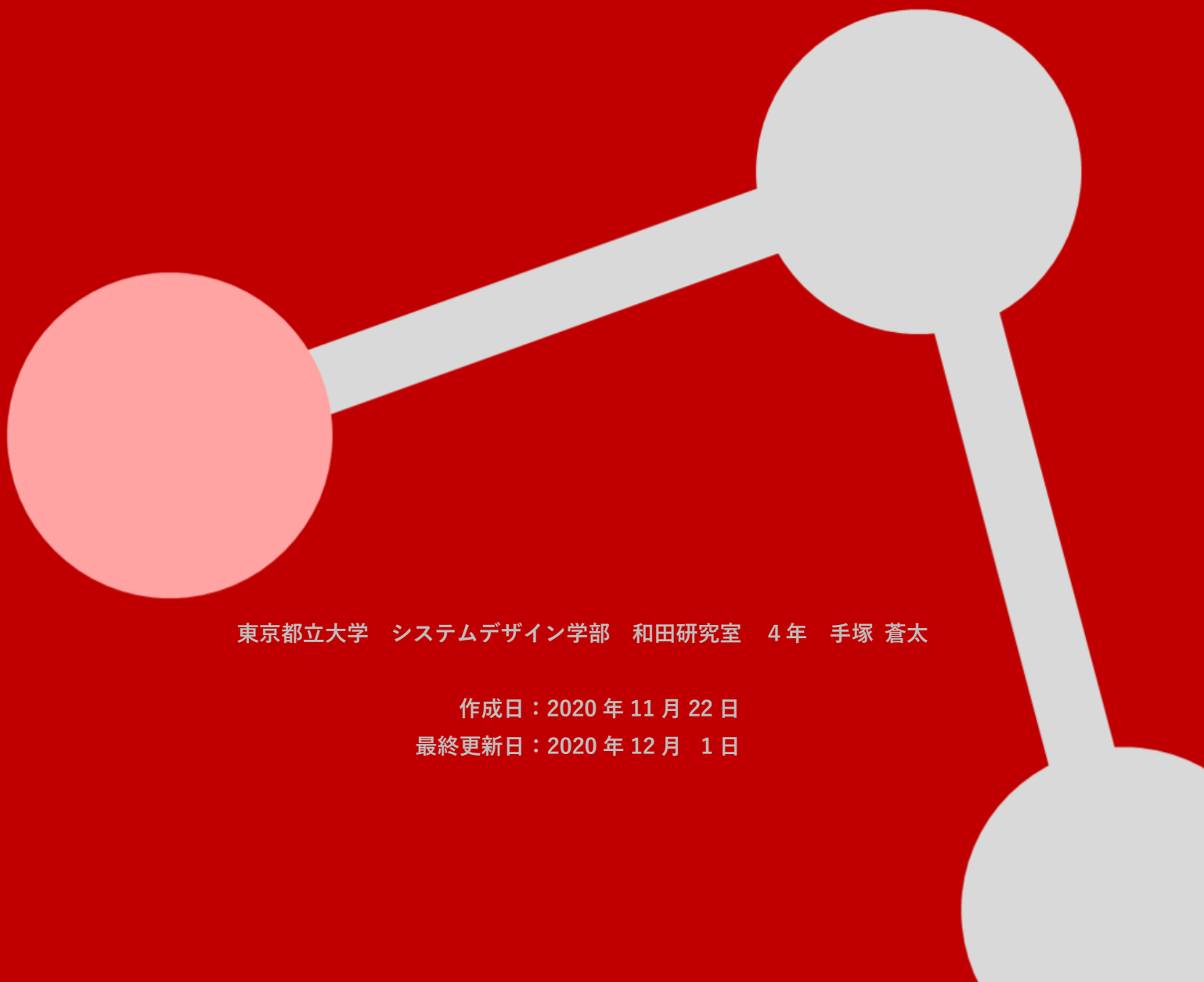
# シリアルリンクロボットアームを 直感的に操作可能な RTC 群

ユーザーマニュアル ver 1.5

東京都立大学 システムデザイン学部 和田研究室 4年 手塚 蒼太

作成日：2020 年 11 月 22 日

最終更新日：2020 年 12 月 1 日



# 目次

1. はじめに .....	2
2. 開発したシステム.....	3
2.1 開発環境.....	3
2.2 開発した RTC .....	3
2.2.1 RobotArmGUI .....	4
2.2.2 RobotArmCUI .....	6
2.2.3 RobotArmAutomaticController.....	6
2.2.4 AccelerationController.....	7
2.2.5 SerialOutAngle .....	7
2.3 開発したハードウェア .....	8
3. システムの準備 .....	12
3.1 RTC の準備 .....	12
3.2 マイコンの準備 .....	15
3.3 システムの起動方法.....	17
4. 操作方法 .....	21
4.1 GUI を用いた操作 .....	21
4.1.1 上視点から操作.....	21
4.1.2 横視点から操作.....	23
4.2 CUI を用いた操作 .....	24
4.3 自動で動作 .....	25
Q&A .....	27
参考 .....	30
改訂履歴.....	30

## 1. はじめに

ロボットアームを製作・制御しようとする場合、アーム本体の製作については製品化されている組み立てキットを利用すれば簡単に行うことができる。また近年は工業用のものと比べて小型で扱いやすいラジコン用サーボモータ（以下、「サーボモータ」という）が安価に入手できるようになった。これを利用して組み立てキットを使わなくとも手軽にロボットアームを設計・製作できる。しかし、ここで障壁となるのがアームの各関節の制御である。アームを思い通りに動かすには各関節のサーボモータを同時に適当な角度に動かす必要がある。キット製品は各関節のサーボモータの角度をジョイスティックやポテンショメータで直接指定することが多い。そこで、ロボットアームの手先位置を指定することで直観的な操作ができる RT コンポーネント群を開発した。組み立てキットの販売も多いシリアルリンク型の 3 軸垂直多関節ロボットアームを対象とし、2 次元表示の GUI で操作できるほか、キーボードから手先座標を直接入力する機能や自動制御の機能を備える。今までミドルウェアを扱ったことがない方でも気軽に手を出せるような直感的に使えるシステムを目指した。

## 2. 開発したシステム

### 2.1 開発環境

- Windows 10 64bit
- OpenRTM-aist 1.2.1
- Visual Studio 2018 (C++)
- Python 3.7.7
- CMake 3.16.6
- Arduino IDE 1.8.5

### 2.2 開発した RTC

ロボットアームを制御するにあたり、五つのコンポーネントを製作した。システムの内部ブロック図を図1に示す。RobotArmGUI コンポーネントに他コンポーネントを接続することでシステムを構成する。

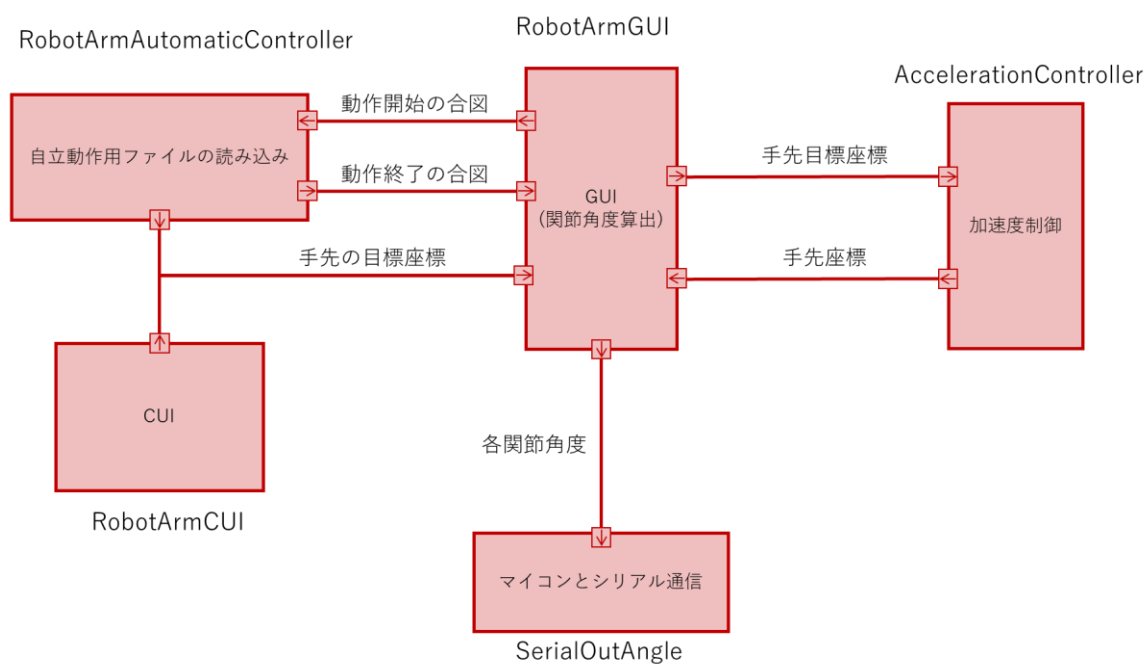
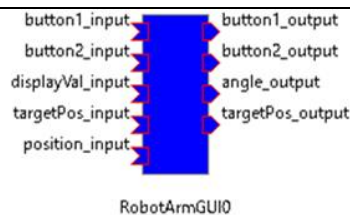


図1 SysML ブロック図

ライセンスは GPL とする。

## 2.2.1 RobotArmGUI

表 1 RobotArmGUI

		
InPort		
名前	データ型	説明
button1_input	RTC::TimedBoolean	ボタン1を操作
button2_input	RTC::TimedBoolean	ボタン2を操作
displayVal_input	RTC::TimedLong	任意の表示値を入力
targetPos_input	RTC::TimedPoint3D	目標手先位置[m]
position_input	RTC::TimedPoint3D	手先位置[m]
OutPort		
名前	データ型	説明
button1_output	RTC::TimedLong	ボタン1の状態
button2_output	RTC::TimedLong	ボタン2の状態
angle_output	RTC::TimedDoubleSeq	各関節角度[rad]
targetPos_output	RTC::TimedPoint3D	手先目標位置[m]
コンフィギュレーションパラメータ		
名前	デフォルト値	説明
upperArmLength	300	上腕長さ[mm]
lowerArmLength	300	下腕長さ[mm]
outerLimitRadius	500	動作範囲上限半径[mm]
innerLimitRadius	100	動作範囲下限半径[mm]
showSideView	1	横視点の有無(0 or 1)
showCoordinates	1	座標表示の有無(0 or 1)
showCoordinateAxes	1	座標軸の有無(0 or 1)
auxiliaryLineInterval	45	補助線の間隔[°]
auxiliaryArcInterval	100	補助円弧の間隔[mm]
wheelRate	10	ホイール1段ごとのZ方向変位
switchWheelDirection	0	ホイール方向逆転(0 or 1)
accentColor_R	255	アクセントカラー (赤)
accentColor_G	0	アクセントカラー (緑)
accentColor_B	0	アクセントカラー (青)
backgroundBrightness	0	背景明るさ
arcBrightness	50	円弧の明るさ
custom1_name	Auto	ボタン1の名前
custom1_pressName	Running	ボタン1押下時の名前
custom1_choicesNum	3	ボタン1送信値の数
custom1_alterateOperation	1	ボタン1をオルタネイト動作(0 or 1)
custom1_linkWithRightClick	0	ボタン1を右クリックと連動(0 or 1)
custom2_name	none	ボタン2の名前
custom2_pressName	pressName	ボタン2押下時の名前
custom2_choicesNum	1	ボタン2送信値の数
custom2_alterateOperation	0	ボタン2をオルタネイト動作(0 or 1)
custom2_linkWithRightClick	0	ボタン2を右クリックと連動(0 or 1)

GUI を作り出すコンポーネント。描画には DX ライブラリを利用している。逆運動学による各関節角度の計算もこのコンポーネントで行う。ロボットアームを上から見た様子を簡易的に表示し、手先の目標位置をマウスでドラッグすることで  $x, y$  方向の移動を、マウスホイールを回転することで  $z$  方向の移動を行う (図 2 上部)。横からの視点 (回転軸の動きを無視した視点) も表示可能で、上からの視点と同様に手先の目標位置をドラッグして移動できる (図 2 下部)。コンフィギュレーションパラメータを変更することで補助線の間隔や操作部の色、ボタン数などを設定できる (図 3)。

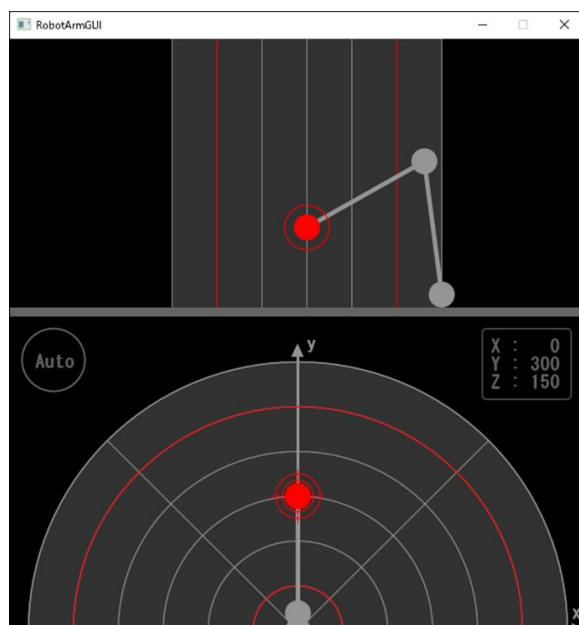


図 2 デフォルト値の GUI

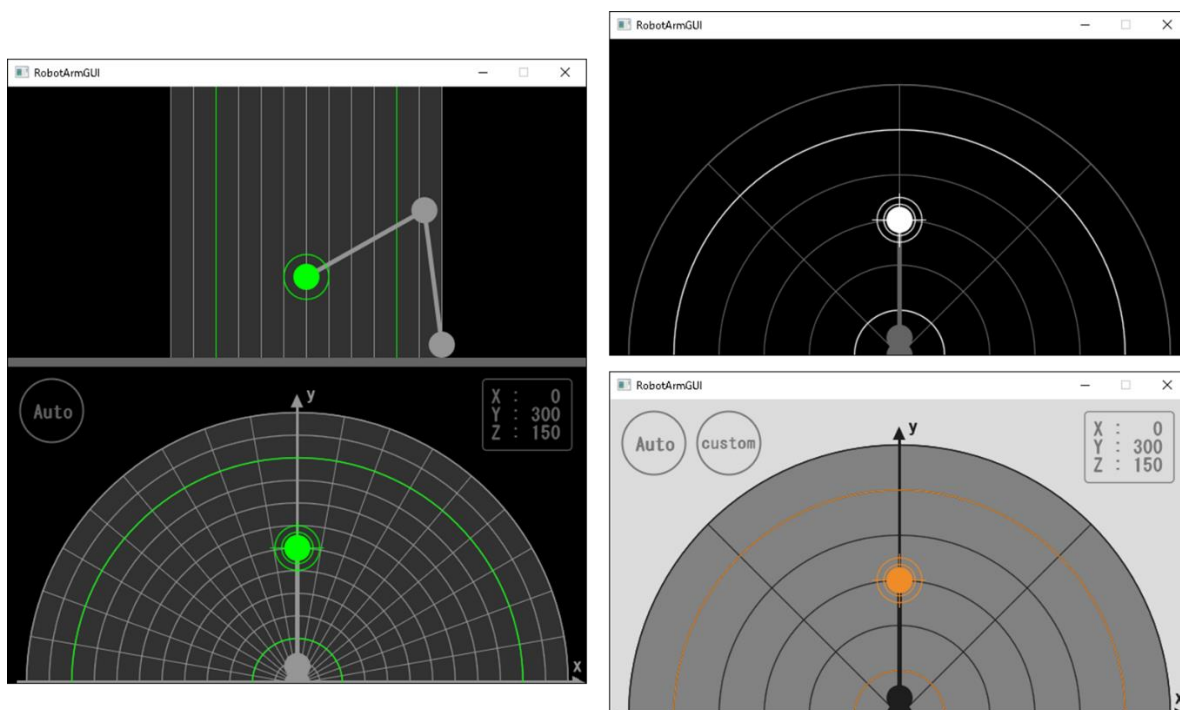



図 3 カスタマイズした GUI

また、自分で表示名や操作方法、送信値を任意に指定できる「Custom ボタン」を備える (図 3 右下)。

### 2.2.2 RobotArmCUI

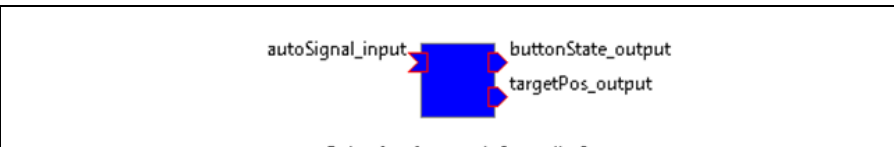
表 4 RobotArmCUI

 RobotArmCUI0		
OutPort		
名前	データ型	説明
targetPos_output	RTC::TimedPoint3D	手先目標座標[m]

CUI により手先目標位置の指定を行う。

### 2.2.3 RobotArmAutomaticController


表 5 RobotArmAutomaticController

 RobotArmAutomaticController0		
InPort		
名前	データ型	説明
autoSignal_input	RTC::TimedLong	ファイル選択用信号
OutPort		
名前	データ型	説明
buttonState_output	RTC::TimedBoolean	ボタン操作
targetPos_output	RTC::TimedPoint3D	手先目標座標[m]

手先位置を記述したテキストファイルを読み込み、手先目標位置を順に移動させることでロボットアームを自動で動作させる。ファイルは3つまで読み込める。

## 2.2.4 AccelerationController


表 2 AccelerationController

 <p>targetPos_input → position_output</p> <p>AccelerationController0</p>		
InPort		
名前	データ型	説明
targetPos_input	RTC::TimedPoint3D	目標手先位置[m]
OutPort		
名前	データ型	説明
position_output	RTC::TimedPoint3D	手先位置[m]
コンフィギュレーションパラメータ		
名前	デフォルト値	説明
accel	20	加速度[mm/F <sup>2</sup> ]
maxSpeed	0.2	最大速度[mm/F]

目標手先位置への追従速度を調節するコンポーネント。ロボットアームの手先目標位置を受け取り、手先位置を出力する。これにより、目標手先位置を素早く移動させてもアームの手先位置は滑らかに移動する。

## 2.2.5 SerialOutAngle

表 3 SerialOutAngle

 <p>angle_input</p> <p>SerialOutAngle0</p>		
InPort		
名前	データ型	説明
angle_input	RTC::TimedDoubleSeq	各関節座標[rad]
コンフィギュレーションパラメータ		
名前	デフォルト値	説明
portName	COM1	ポート名

各関節座標を受け取り、シリアル通信でマイコンに送信する。マイコン側ではそれぞれのサーボモータを writeMicroseconds 関数を使用して 544～2400 の範囲（約 0.1 度単位）で動かすが、シリアル通信では一度に 1byte 以上の値は送ることができないため値を分けて送信している（図 4）。



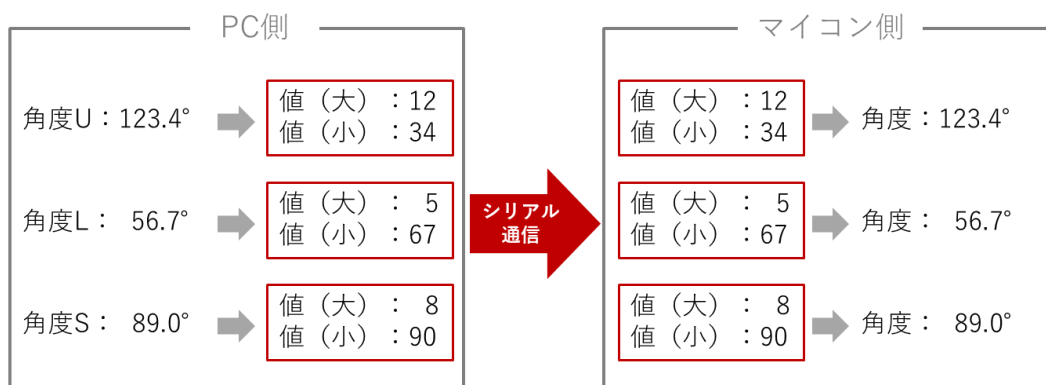


図4 SerialOutAngle コンポーネントによる PC-マイコン間データやり取りのイメージ

## 2.3 開発したハードウェア



図5 設計したロボットアームの3Dモデル

開発した RTC 群のテストを行うため、小型のロボットアームの製作を行った。3DCAD を用いて設計し、3D プリンタとレーザー加工機を用いて部品を製作した（図 5）（図 6）（図 7）。仕様を表 5 に示す。リンク部分は厚さ 5mm のアクリル板製で土台部分に 3D プリンタで出力した PLA 製のパーツを使用している。3つのサーボモータ（DS3218）を土台部分に集中して設置し、複数の平行リンク機構を用いて各関節へ動力を伝達する。サーボモータの制御には arduino を用いる。また、旋回用サーボモータへの負荷を軽減するためクローム鋼球と PLA パーツ、厚さ 2mm のアクリル板を使用して製作した簡易的なスラスト軸受機構を備えている（図 8）。



図 6 製作したロボットアームの部品

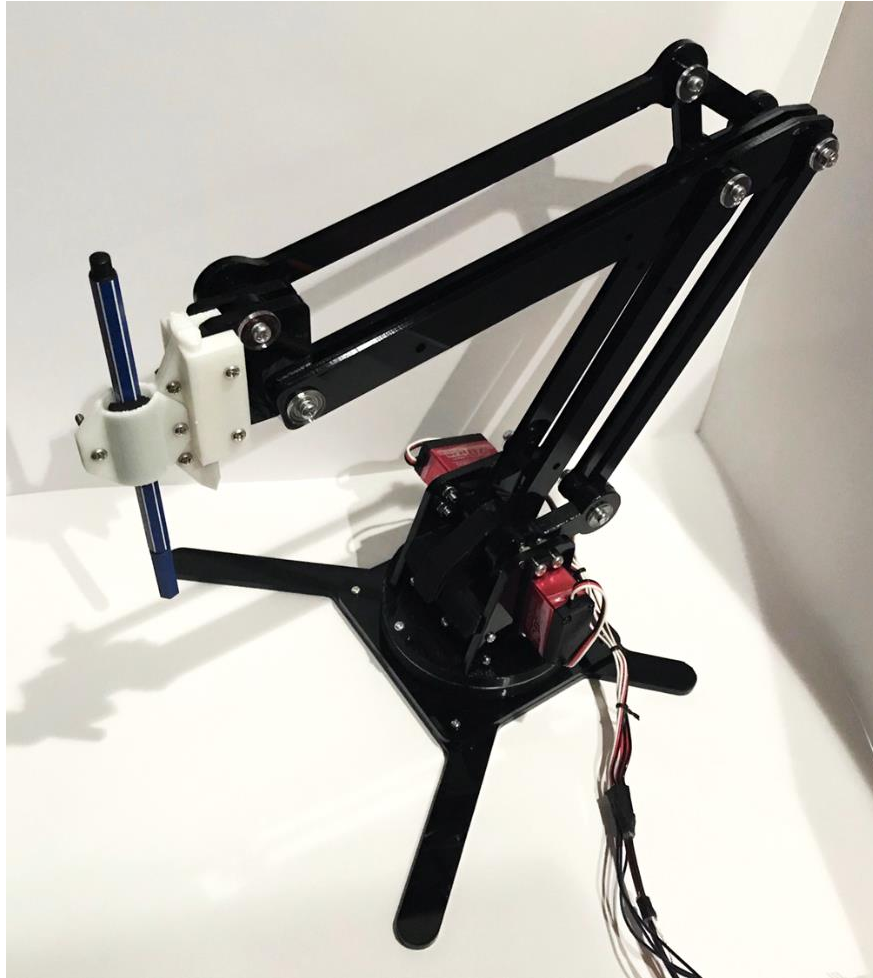


図7 製作したロボットアーム

表5 Robot arm specifications

種類	垂直多関節
軸数	3 軸
上腕長さ	300mm
下上長さ	300mm
可搬重量	200g
総重量	943g
電源仕様	DC5V 3A

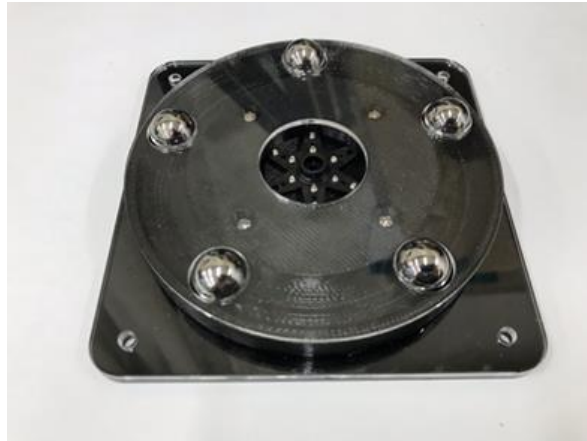


図8 簡易スラストベアリング

また、動作確認用に手先部分には筆記用部を装着できるソケットパーツを設けた（図9）。付け根部分には受動動作するスライド機構を備えており、筆記具に大きな負荷がかかることを防ぐ構造となっている。



図9 筆記具用ソケット

## 3. システムの準備

### 3.1 RTC の準備

#### (1) DX ライブラリを配置する

GUI の描画には DX ライブラリ を使用するため、わかりやすい場所（C ドライブ直下など）に配置しておく。

#### (2) ビルドする

CMake と Visual Studio を使った手順を以下に示す。

※「RobotArmGUI」と「RobotArmCUI」は名前が似ているので注意。

##### (2-1) 「RobotArmGUI」以外のビルド

- ① CMake を起動する。
- ② 「RobotArmCUI」ディレクトリ内の「CMakeLists.txt」を CMake 上にドラッグ&ドロップする (図 10)。

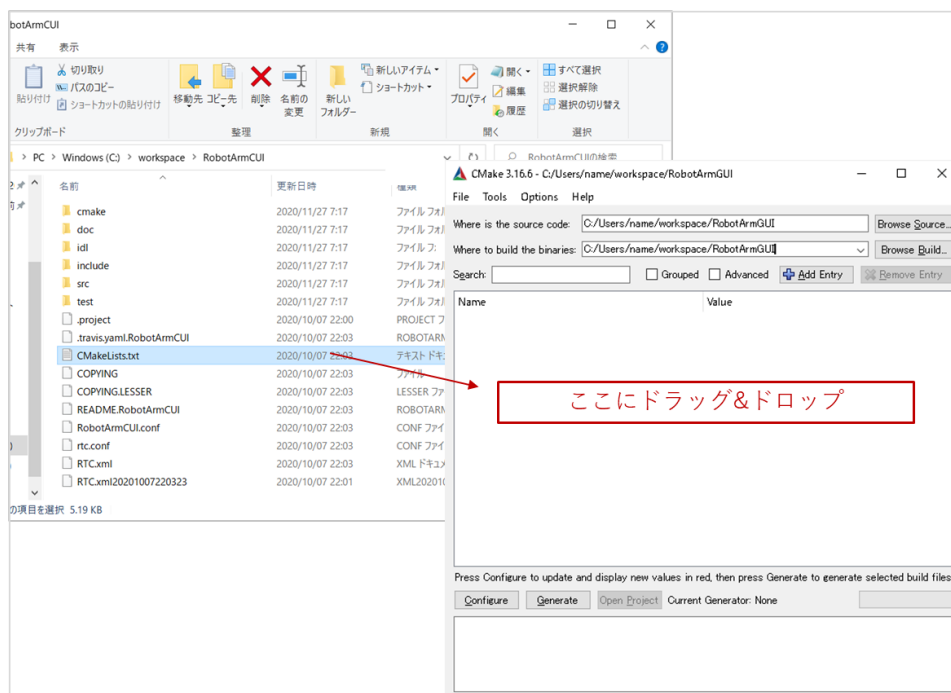


図 10 CMake での作業

- ③ 生成ファイルを入れる build ディレクトリを追加する (図 11)。

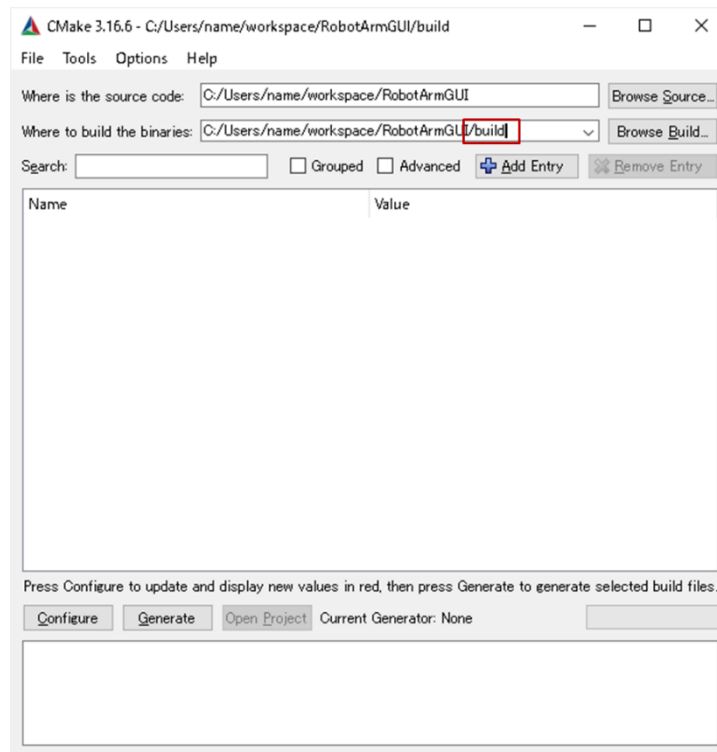


図 11 build ディレクトリの追加

- ④ 「Configure」を選択する。
- ⑤ 使用する Visual Studio のバージョンを選択し、「Finish」を選択する。（ディレクトリ生成のダイアログが出てきたら「Yes」を選択）
- ⑥ メッセージ「Configuring done」を確認し「Generate」を選択。「Generating done」の表示されたら完了。
- ⑦ build フォルダ内の「MyConsoleIn.sln」を開く。Visual Studio が立ち上がる。
- ⑧ ウィンドウ上部から Release ビルドを選択する（図 12）。

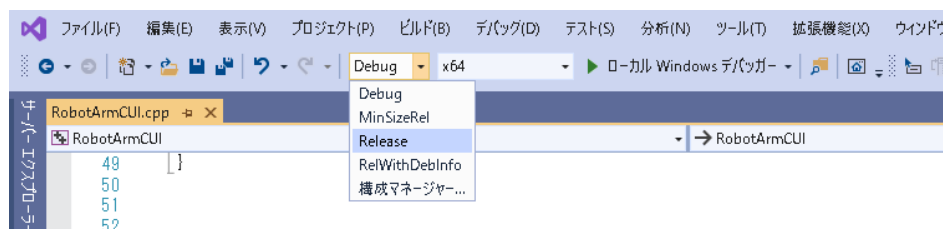


図 12 Release ビルド

- ⑨ ビルドすると RobotArmCUI¥Build¥src¥Release に実行ファイルが作成される。
- ⑩ 「RobotArmAutomaticController」, 「AccelerationController」, 「SerialOutAngle」でも②～⑧と同様の手順を繰り返して設定する。

## (2-2) 「RobotArmGUI」のビルド

- ① (2-1)の①～⑤までと同様の作業を行う。
- ② DXLIB\_DIRS の欄にDXライブラリのパッケージ内に入っている「プロジェクトに追加すべきファイル\_VC用」ディレクトリのパスを入力する(図13)。(パスの例→ c:\DxLib\_VC\プロジェクトに追加すべきファイル\_VC用)

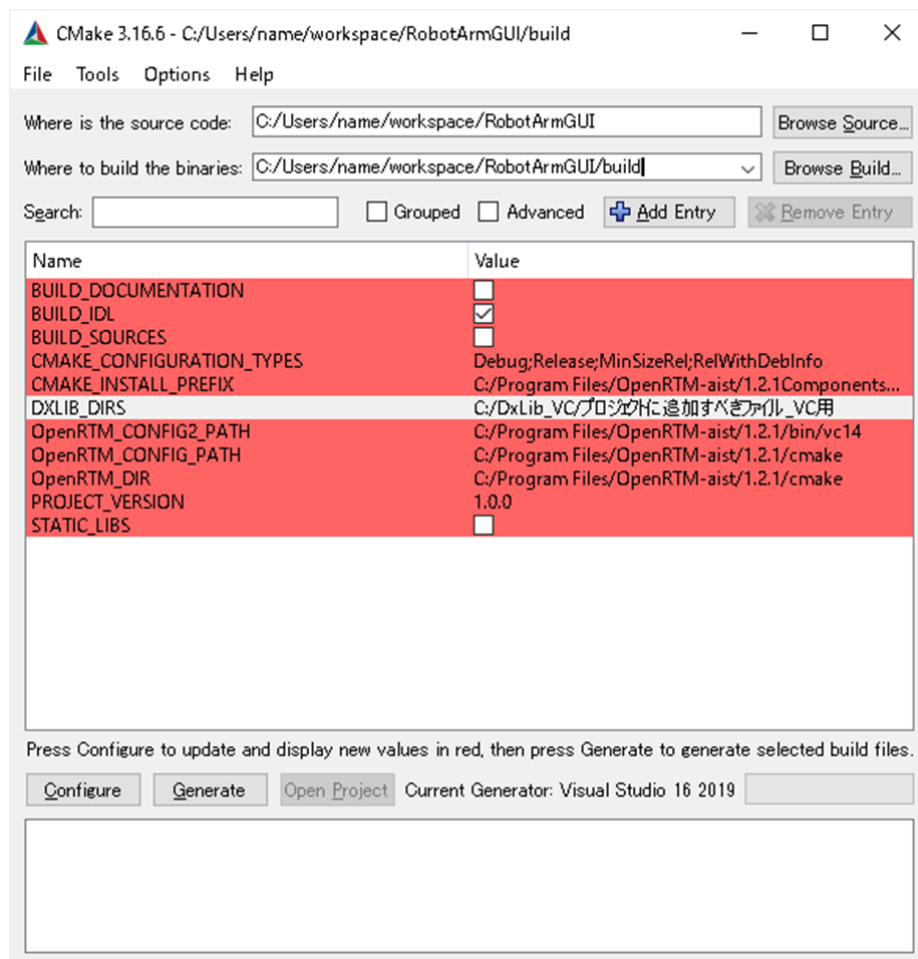


図13 DXライブラリの設定

- ③ 「Generate」を選択。「Generating done」の表示されたら完了。
- ④ buildフォルダ内の「MyConsoleIn.sln」を開く。Visual Studioが立ち上がる。
- ⑤ ウィンドウ上部からReleaseビルドを選択する(図12)。
- ⑥ ビルドすると RobotArmGUI¥Build¥src¥Release に実行ファイルが作成される。



(3) RobotoArmAutomaticController のファイルを準備

- ① 「RobotoArmAutomaticController」ディレクトリ内の「data1.dat」, 「data2dat」, 「data3.dat」を実行ファイルがある Release ディレクトリ内に移動する.
- ② ファイルを好きなように書き換える. (「4.3 自動で動作」を参照)

## 3.2 マイコンの準備

Arduino を使う場合の手順をいかに示す.

- (1) [Arduino IDE](#) をインストールしておく.
- (2) 「arduino\_SerialOutAngleRTC」ディレクトリ内の「arduino\_SerialOutAngleRTC.ino」を開く.
- (3) PC と arduino を USB ケーブルで接続する.
- (4) 「ツール」→「シリアルポート」から arduino が接続されたポートを選択する (図 14).

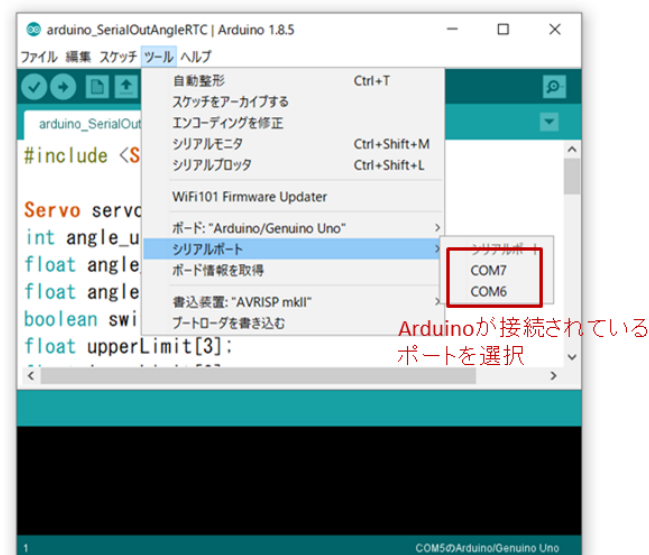


図 14 arduinoIDE\_シリアルポート選択

- (5) 「マイコンボードに書き込む」を選択 (図 15).



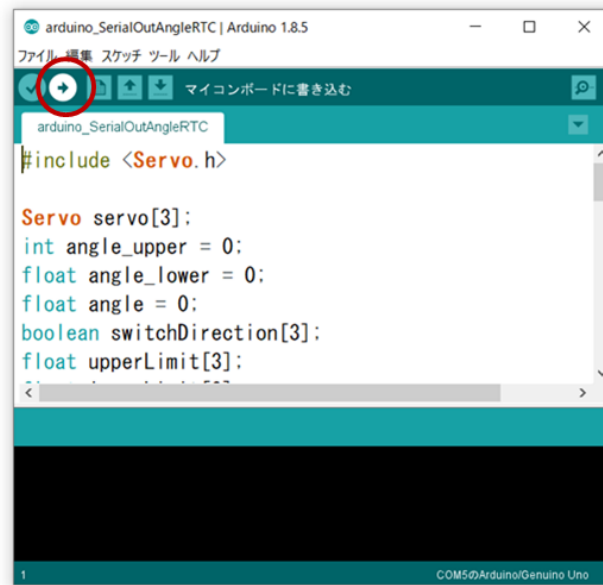


図 15 arduinoIDE\_マイコンボードに書き込む

※サーボモータの動作方向を変更したい場合は、ソースコードの「switchDirection」変数を変更する。  
サーボモータの動作範囲を変更したい場合は「upperLimit」変数, 「lowerLimit」変数を変更する。

### 3.3 システムの起動方法

#### (1) マイコンと PC を接続

マイコンと PC を USB ケーブルで接続する。接続したポート名を確認しておく（Windows の場合はデバイスマネージャー等で確認）。

#### (2) RTC を接続

- ① 「Start Naming Service」を起動する。
- ② OpenRTP を起動する。
- ③ 画面右上の「パースペクティブを開く」を選択。表示されるダイアログで「RT System Editor」を選択し、「開く」を選択（図 16）。

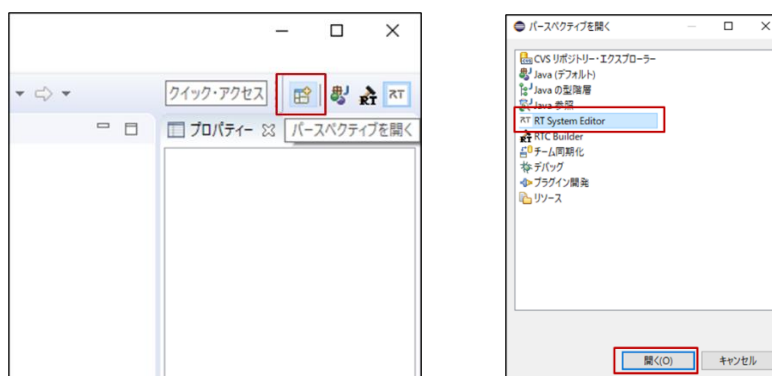


図 16 RT システムエディターを開く

④ 「3.1 RTC の準備」で作成した 5 つの実行ファイルを開く。少し待つと各コンポーネントのコンソール画面に「[コンポーネント名] ready!」と出力される。

⑤ 「NameServerView」にコンポーネントが表示されていることを確認する。折りたたまれている場合は「>」を押して展開する。RTC の名前の語尾に「0」がつく仕様。

「NameServerView」にネームサーバが現れない場合は、「ネームサーバを追加」を選択しダイアログで「localhost」を入力し「OK」を選択。

⑥ 「Open New System Editor」をクリック。右ウィンドウに「System Diagram」が表示される。

⑦ 「NameServerView」に表示された 5 つコンポーネント (RobotArmGUI, RobotArmCUI, RobotArmAutomaticController, AccelerationController, SerialOutAngle) を「System Diagram」にドラッグ&ドロップする。

⑧ 図 17 のようにコンポーネントを接続する。データポート間でドラッグ&ドロップして、表示されるダイアログは「OK」を選択する（図 18）。



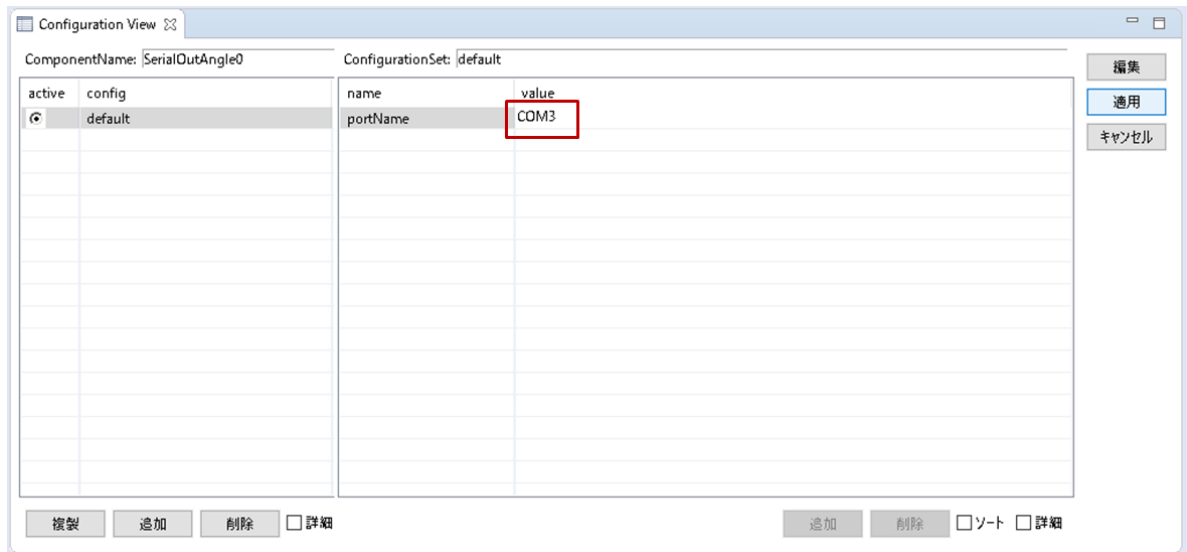


図 19 シリアルポートの設定

- ⑩ 使用するロボットアームの上腕・下腕長さがデフォルト値（上腕：300mm，下腕：300mm）でない場合は、「System Diagram」上の「RobotArmGUI」コンポーネントを選択し、「Configuration View」の「upperArmLength」と「lowerArmLength」の欄に使うアームの長さを入力する。
- ⑪ 「Activate Systems」をクリックしする。RTC が緑になったら成功（図 20）。

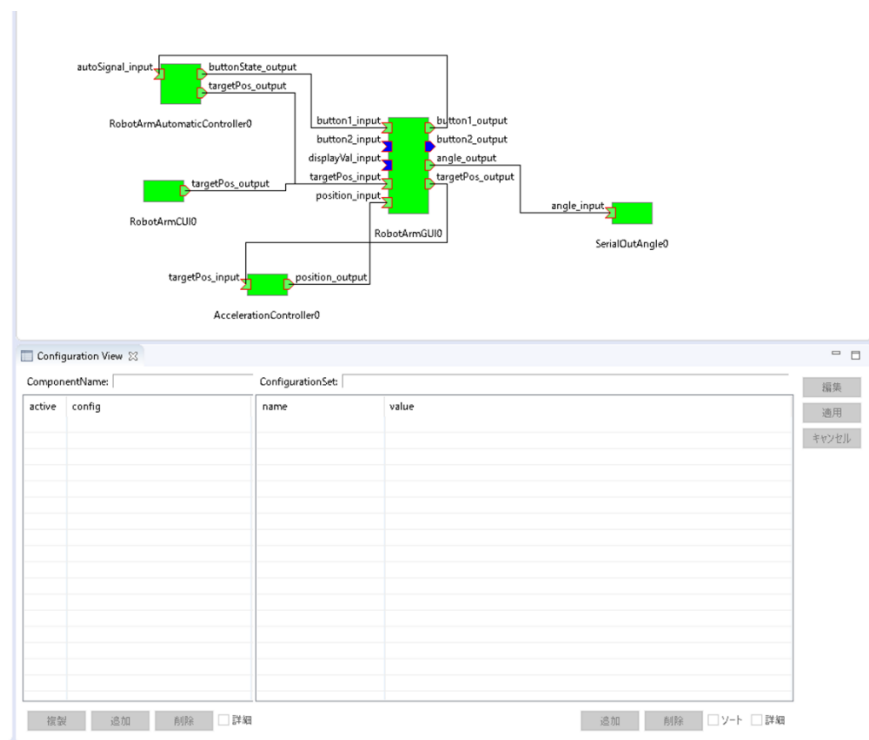


図 20 アクティベート完了

- (4) ロボットアームの電源を入れる.  
※各関節が初期角度まで動くので注意する.

**【GUI のみを使用する場合】**

GUI による操作さえできれば良いという場合は,「RobotArmCUI」と「RobotArmAutomaticController」を使わずにその他 3 つのコンポーネントのみを接続して使うことも可能. その場合は「RobotArmGUI」の「Configuration View」から「showAutoButton」を 0 に設定する.

## 4. 操作方法

### 4.1 GUI を用いた操作

基本的な操作は RobotArmGUI ウィンドウから行う。

#### 4.1.1 上視点から操作

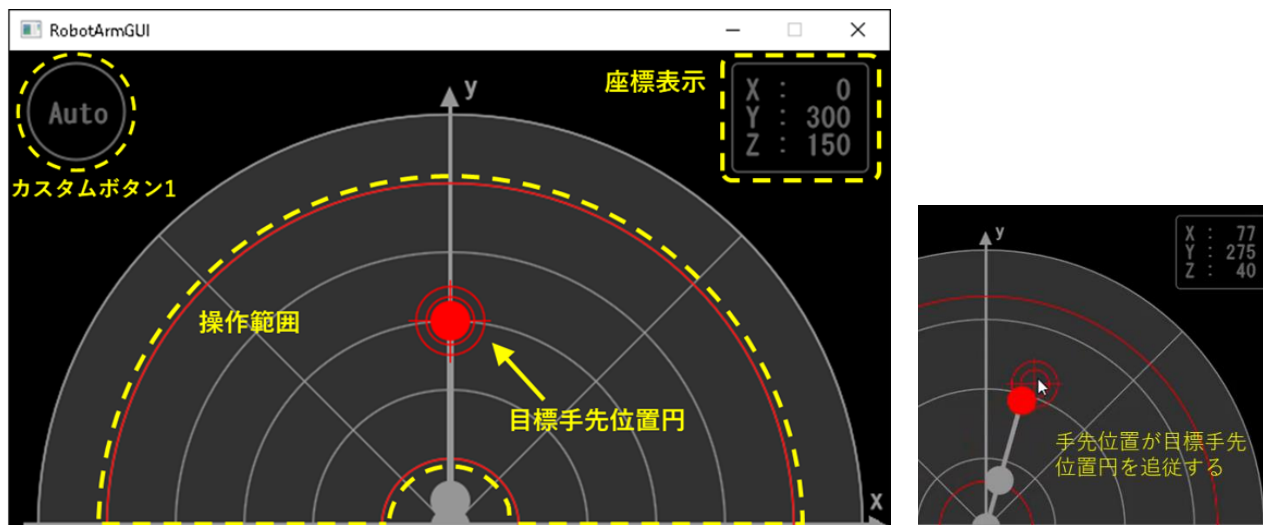


図 21 GUI の上からの視点

ロボットアームを真上から見た視点での操作について説明する。操作範囲内には目標手先位置円とアームを上から見た簡易モデルが描画されている（図 21）。目標手先位置円を動かすことアームで手先位置は目標手先位置円を追従するように動く。目標手先位置円の移動は xy 方向はマウスでドラッグすることで行い、z 方向への移動はマウスホイールを回転することで行う。左上部分はカスタムボタン表示領域となっており、コンフィギュレーションパラメータを変更することでボタンの数や任意の名前、動作を設定可能。右上部には現在の手先位置がロボットアームの根元を原点とした座標が表示される。z 方向の位置については目標手先位置円の内部にある円の大きさで大まかに確認できる（図 22）。

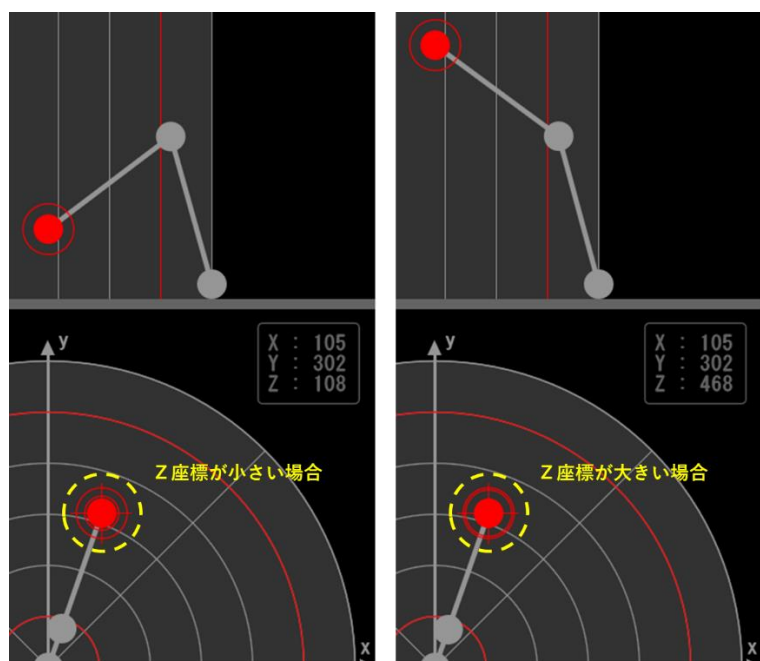


図 22 z 座標の大まかな確認方法

### 【自動位置合わせモード】

手先位置を一定間隔で移動させたい場合は自動位置合わせモードを使うと良い。Shift キーを押している間、このモードが有効になる。目標手先位置円をドラッグすると、補助線の交点のうち最も近い点に点が描画される（図 23 左）。ドラッグが終了するとその点の上に目標手先位置円が移動する（図 23 右）。

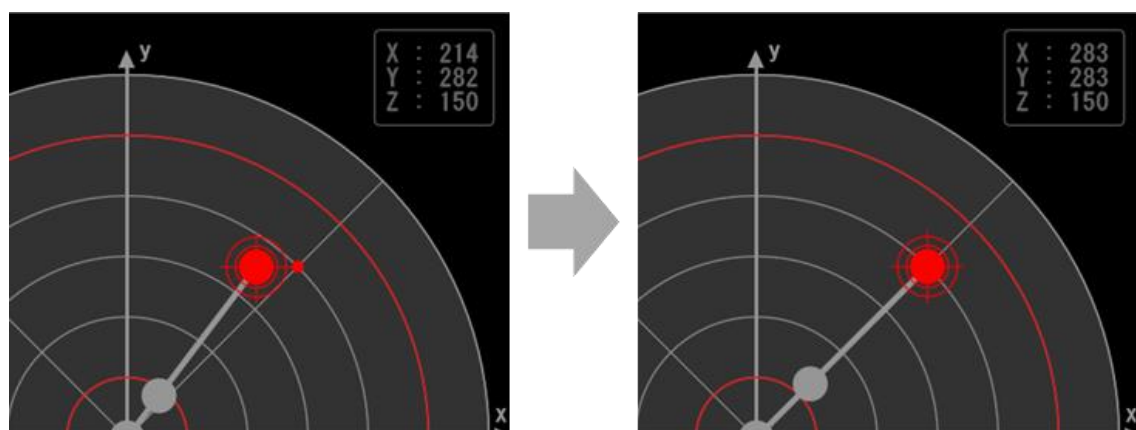


図 23 自動位置合わせモード

### 【任意の整数値を表示】

任意の数値（他コンポーネント内のデータ等）を GUI 上に表示することができる。「RobotArmGUI」コンポーネントの「displayVal\_input」データポートに整数値を入力することで、GUI 上に表示される。データポートに入力があるときのみ表示される。



図 24 GUI 上に任意の値を表示

### 4.1.2 横視点から操作

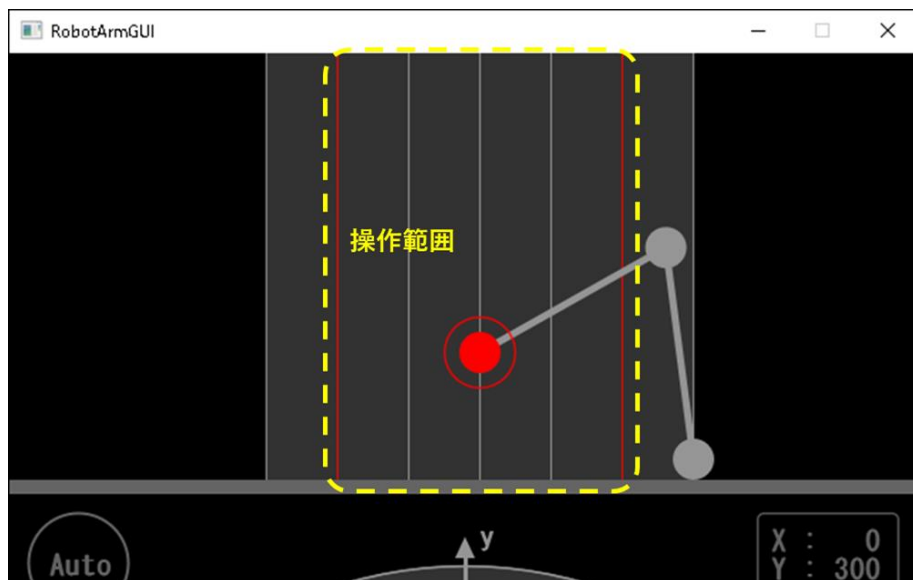


図 24 GUI の横からの視点

ロボットアームの旋回方向の軸の動きを無視して真横から見た視点での操作について説明する。横からの視点と同様に、操作範囲内には目標手先位置円とアームの簡易モデルが描画されている（図 24）。マウスでドラッグすることでアームを横から見た際の目標手先位置を移動できる。



## 4.2 CUI を用いた操作

ロボットアームを正確な位置に動かしたいは CUI で手先座標を直接打ち込むことができる。

「RobotArmCUI」コンポーネントのコンソールウィンドウを開くと「Please input target coordinates:」と表示されているので任意の手先座標を x, y, z, の順で入力する (図 25)。座標はロボットアームの根元を原点とし, 単位[mm]で入力する。エンターキーを押すと GUI 上の手先目標位置円が入力した座標に移動する。操作範囲外を入力した場合は操作範囲内で最も入力座標に近い座標に移動する。

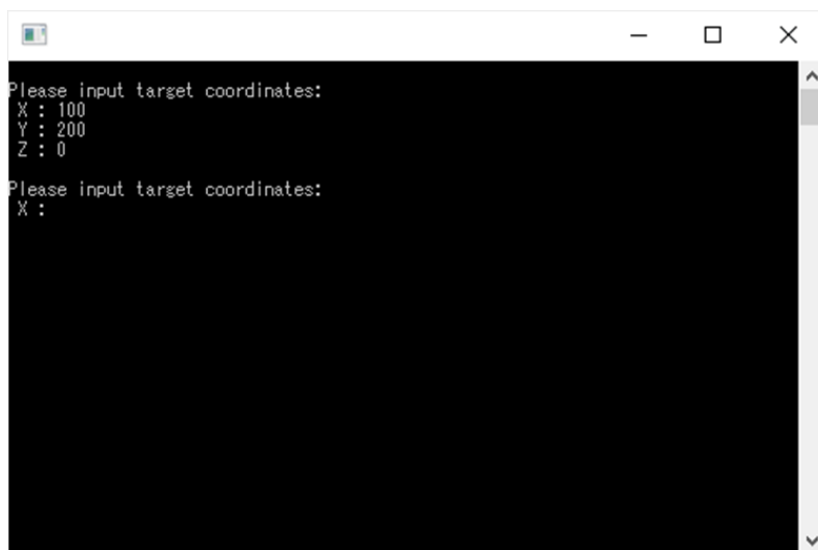


図 25 CUI での座標入力の様子

また, 座標を途中まで入力したがキャンセルしたい場合は数値以外 (アルファベットや「-」以外の記号) を入力してエンターキーを押すことで入力をキャンセルできる (図 26)。

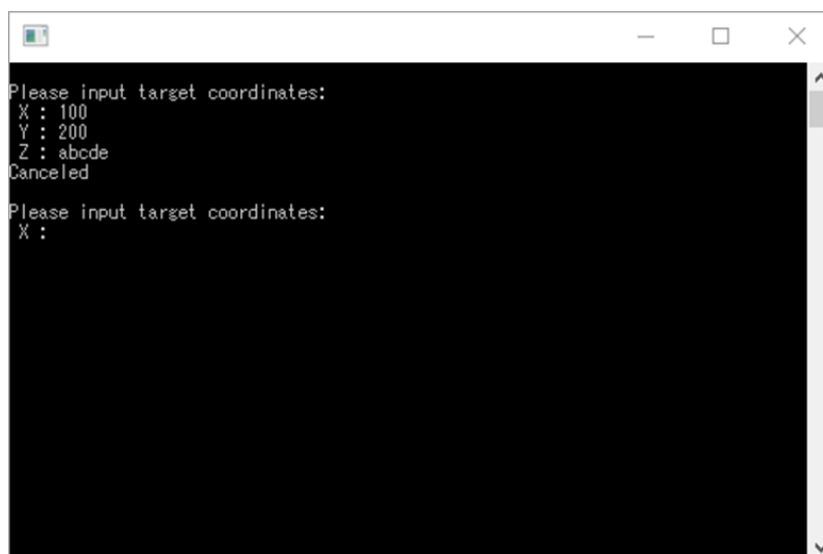


図 26 CUI の入力キャンセルの様子

### 4.3 自動で動作

テキストファイルを読み込ませることでロボットアームに決められた動作を行わせることができる。

「RobotArmAutomaticController」ディレクトリ内の「data1.data」, 「data2.data」, 「data3.data」に動作用データを3種類まで保存できる。テキストファイルの1行目には動作間隔を, それ以降の行には目標座標をスペースで区切って書き込む (図 27)。



図 27 自動制御用テキストファイルの例

動作データは「RobotArmGUI」ウィンドウの「Auto」ボタン（カスタムボタン 1）から行う（図 28）。画面左側の「Auto」と書かれた円の上にマウスポインタを移動させるとボタン下部に数字が表示されデータ選択モードに移行する。その状態でマウスホイールを回すことで使用するデータを選ぶことができる。データを選びクリックすると「Running」に表示が切り替わり、選んだファイルに書き込まれた座標が指定した間隔で読み込まれ手先目標座標が移動する。最後の座標まで移動し終わると表示が「Auto」に戻る。また、動作が途中で再度「Auto」ボタンをクリックすることで自動動作をキャンセルできる。

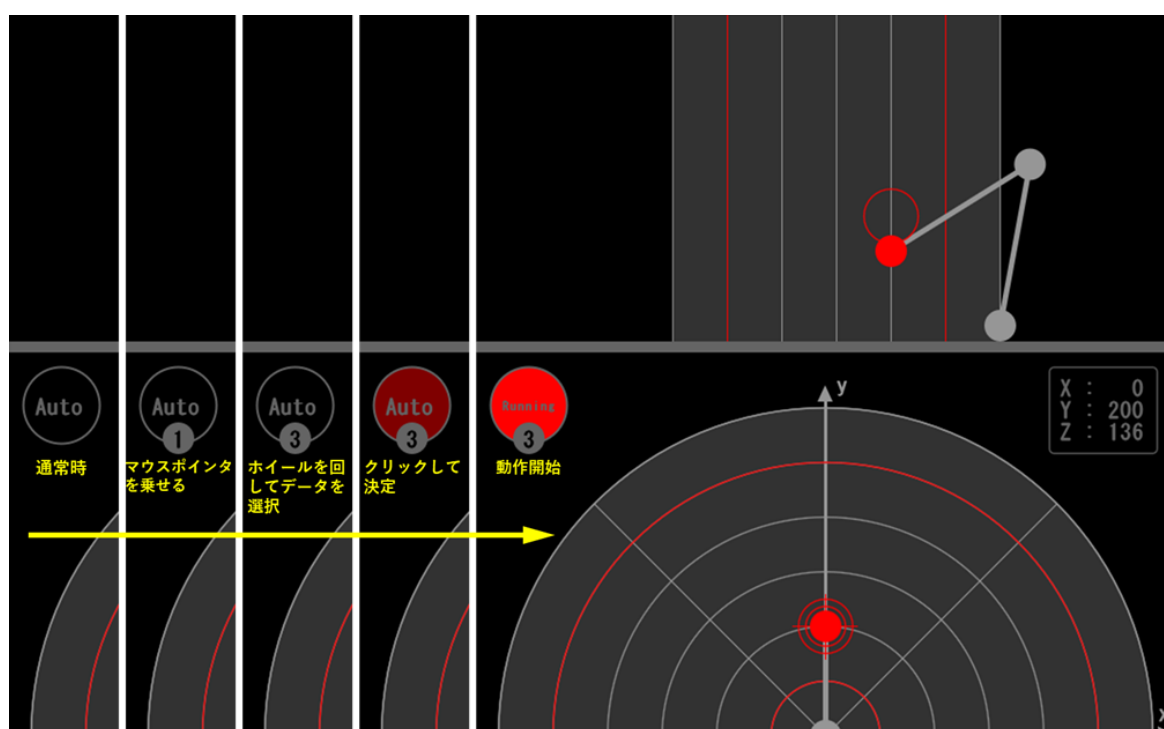


図 28 Auto ボタンの挙動

※「RobotArmGUI」コンポーネントのカスタムボタン 1 と 2 は全く同じ機能を持っているが、コンフィギュレーションパラメータのデフォルト値のみ異なる。「RobotArmAutomaticController」コンポーネントをつなげることを想定し、カスタムボタン 1 はデフォルトでは以下のように設定されている。

名前：Auto  
 押下時の名前：Running  
 送信値の数：3  
 オルタネイト動作：1（あり）  
 右クリックと連動：0（無し）

※テキストファイルで指定した座標に目標手先位置座標が瞬間的に移動する仕様となっているため、「AccelerationController」コンポーネントの代わりに自作のコンポーネント（例えばPID 制御をするコンポーネントなど）を使用する場合は、テキストファイル内の座標は連続している必要があることに注意すること。（「AccelerationController」コンポーネントを使用している場合はアームは滑らかな動作を行うため座標同士が離れていても問題ない）

## Q&A

**Q.** このマニュアルに載っているロボットアーム以外のロボットアームも動かせますか？

**A.** RC サーボモータを使用した 3 軸又は 4 軸のシリアルリンクロボットアームであればなんでも動かします。

**Q.** GUI の文字色の設定方法は？

**A.** 文字色は背景の明るさによって自動で設定されます。

**Q.** GUI の補助線などの色の設定方法は？

**A.** 線の色は操作半円の明るさによって自動で設定されます。

**Q.** システムをアクティベートするとエラーが出ます。

**A.** OpenRTM-aist のバージョンが 1.2.1 以前の場合、図のようなエラーが出てくることがあります（図 29）。これは RobotArmGUI のアクティベートに時間がかかるためで、無視してかまいません。

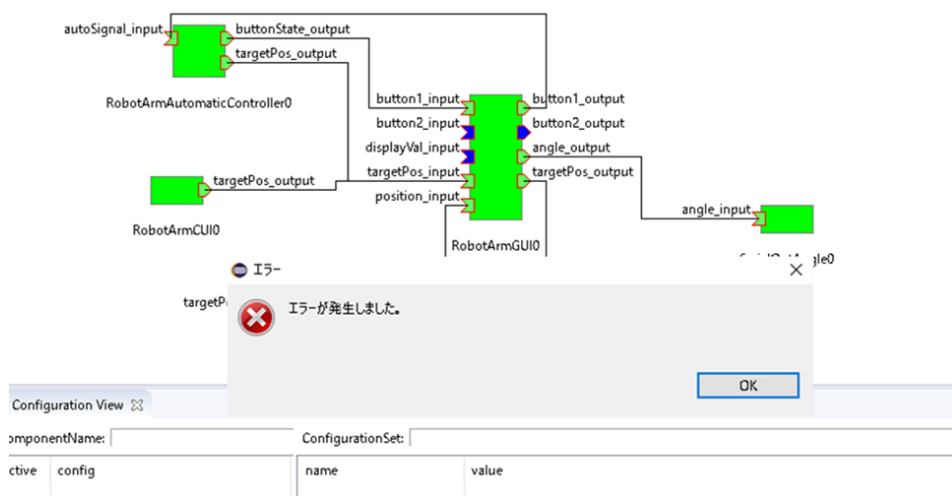


図 29 openRTP のバージョンが 1.2.1 以前の場合に表示されるエラー

**Q.** RobotArmGUI がアクティベートできません。

**A.** RobotArmAutomaticController コンポーネントからの入力待ち状態の可能性があります。RobotArmAutomaticController コンポーネントを繋げない場合は、コンフィギュレーションの showAutoButton を 0 に指定してください。

**Q.** システムをアクティベートしても RobotArmGUI コンポーネントのウィンドウが真っ黒いままです。

**A.** RobotArmGUI コンポーネントはコンソールウィンドウのほかにもう一つ GUI 用のウィンドウを作ります。GUI のウィンドウでアームを操作してください。

**Q.** 実際のロボットアームの状態と GUI に表示された状態が異なっています。

**A.** RobotArmGUI のコンフィグレーションでアームの上腕・下腕長さが実際のアームと等しく設定できているか確認してください。また、マイコン側の角度設定を確認してください。

**Q.** ロボットアームの動きが速すぎる。

**A.** AccelerationControl のコンフィグレーションで手先位置の最大移動速度を指定してください。単位は[mm/F]です。60fps の場合、1/60 秒間で移動する距離を入力してください。

**Q.** 手先位置の追従速度が設定値より遅く感じる。

**A.** RTSystemEditor で「RobotArmGUI」と「AccelerationController」コンポーネントを接続する際に、表示されたダイアログの「詳細」にチェックマークを入れ、「Buffer length」を 1 に設定してから OK を選択してください (図 30)。計 4 箇所指定します。

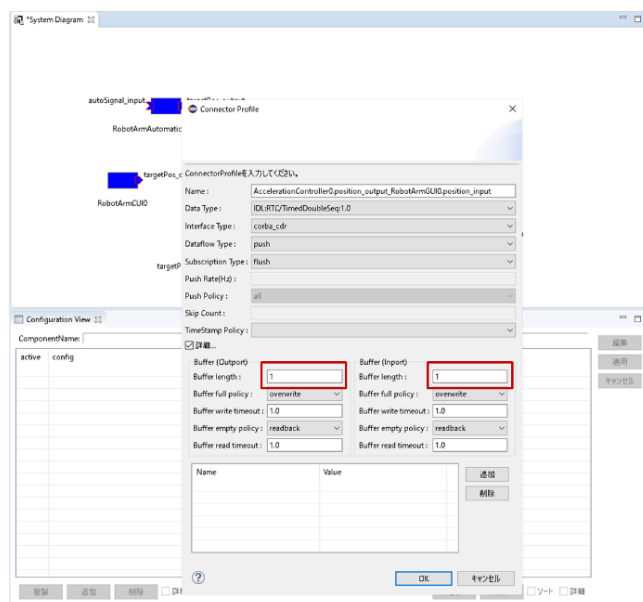


図 30 Buffer length の設定

**Q.** ホイールを回しても手先位置の Z 座標が変化しないときがあります。

**A.** マウスポインタが Auto ボタンの上にある時はマウスホイールでの Z 座標が出来ません。

**Q.** 自動制御が上手くできません。

**A.** 実行ファイルが入ったディレクトリ内にテキストファイル (data1. txt,data2. txt,data3. txt) があることを確認してください。また、テキストファイルに 1000 (1m) 以上の値を書き込むと動作しません。

**Q. Auto ボタンを押したときの挙動が、設定した動作と異なります。**

**A.** 自動制御用のファイルが RobotArmAutomaticController の実行ファイルと同じ場所にあるか確認してください。また、システムをアクティベートした後にファイルを変更しても動作に反映されません。先にファイルの変更・保存を行ってからアクティベートしてください。

**Q. カスタムボタンの名前が小さくて見えにくい。**

**A.** カスタムボタン名は文字数によって自動で決定されます。文字数を減らすとサイズは大きくなります。

**Q. ボタンが押せません。**

**A.** カスタムボタンを右クリックに同期させる設定にすると、左クリックでは押せなくなります。

**Q. 自動制御の際、各座標ごとに待機時間を変えたいです。**

**A.** RobotArtmaticController コンポーネントに各座標の待機時間を設定する機能はありません。一部の動作のみ待機時間を増やしたい場合は同じ座標を複数回連続で入力することで対応してください。

**Q. アームだけでなく、エンドエフェクタも動かしたいです。**

**A.** GUI については「Custom」ボタンを表示させることで簡単な ON/OFF の制御は可能です。ただしシリアル通信コンポーネントは 4 つ以上のデータを送信できないため、書き換える必要があります。

## 参考

[1] 広川美津雄, 井上勝雄, 岩城達也, 加島智子: “直感的インタフェースデザインの設計論の基礎的考察”, 日本感性工学会

## 改訂履歴

2020 年 11 月 22 日 ver1.0 新規作成  
2020 年 11 月 27 日 ver1.1 Q&A を追加  
2020 年 12 月 1 日 ver1.2 図を一部訂正  
2020 年 12 月 5 日 ver1.3 DX ライブラリの設定手順を変更. 表を一部訂正.  
2020 年 12 月 8 日 ver1.4 自動位置合わせモードの説明を追加. その他一部訂正.  
2020 年 12 月 13 日 ver1.5 カスタムボタンの説明, 図を変更. 数値表示機能の説明追加.