スタートガイド

# QEC-M-090T x AZD-KRED x OVR6048K1-V

EtherCAT制御 + HMI統合

iCOP × Orientalmotor

Date: 2025.11.17


QEC-M-090T
(HMI + EtherCAT Mデバイス)

EtherCAT®

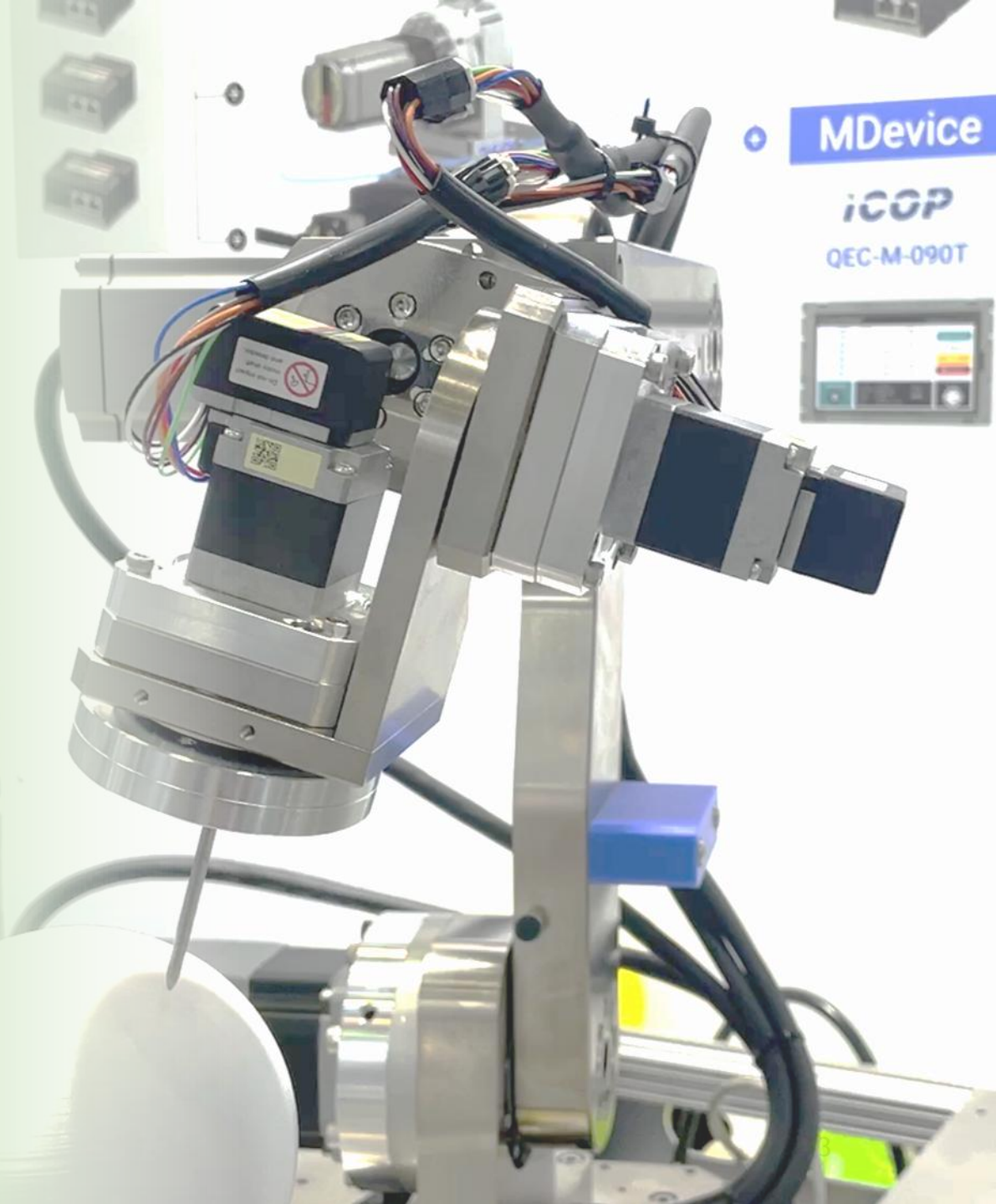
OVR6048K1-V
(6軸ロボット, AZシリーズ)

# 改訂履歴

| 記述日 | バージョン | 備考 |
| --- | --- | --- |
| **2025/10/10** | Version1.0 | New release. |
| **2025/10/30** | Version1.1 | API用の軌道表示ツール(キネマティクスビューア)のパラメータ使用方法を追加 |
| **2025/11/17** | Version1.2 | • 用語を変更: “マスタ/スレーブ” → “Mデバイス / Subデバイス”<br>• タイプミスの修正 スライド59 (“down” → “done”)<br>• 86Duino IDE 501+ 紹介の追加 スライド5 |

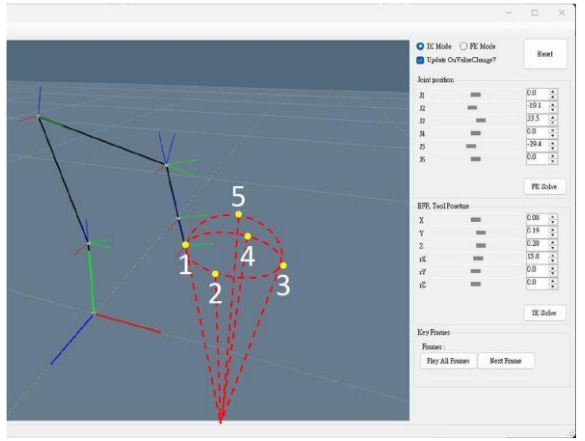
# 1.プロジェクト紹介

部品表と要件



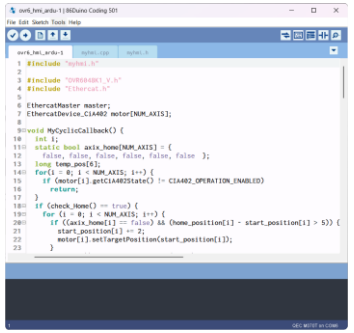
# プロジェクト紹介

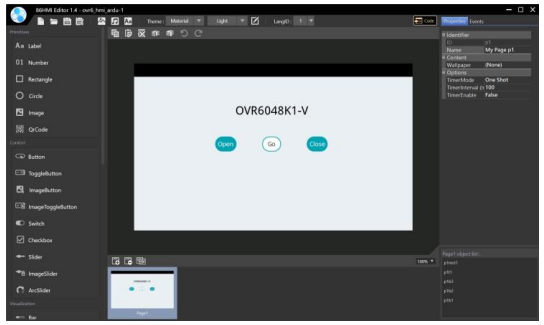
**OVR6048K1-V**のための完全な開発フローを、**QEC-M-090T**（EtherCAT MDevice + HMI）と6台の**AZD-KRED**ドライバを用いて構築しました。このシステムは、**DC同期**による**ECAT_SYNC**で動作し、配線と初期パラメータテンプレート、原点復帰と安全機能、関節空間での初心者向け**運動解析テスト**、そして作業空間での「**球状周回**」デモを網羅しています。



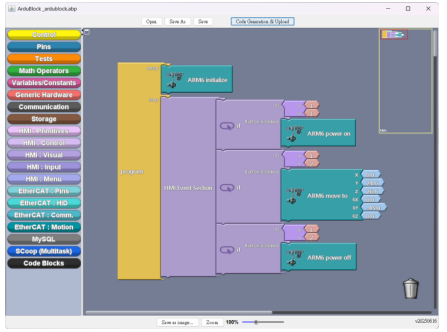
運動学ビューア

また、**86HMI**に結び付けられた**ArduBlock**ブロックというローコードパスも提供しているため、ユーザーは **86Duino**でモーションのプロトタイプ作成、PDO の監視、結果の検証を迅速に行うことができます。


86Duino



86HMI

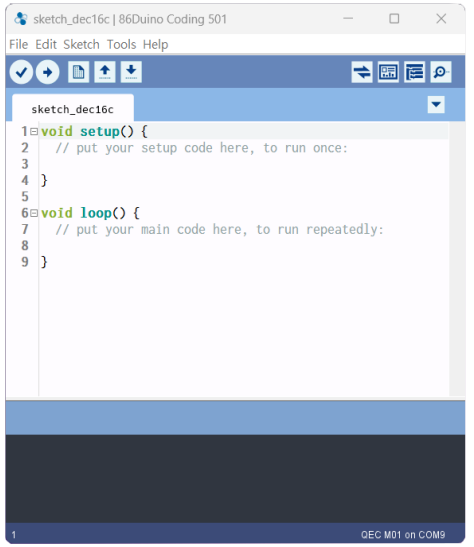

ArduBlock

# ソフトウエア：86Duino IDE 501+

**86Duino IDE（統合開発環境）**は、コードの記述と86Duinoボードへのアップロードを簡単にします。Windows、Mac OS X、およびLinux上で動作し、環境は Javaで書かれており、Arduino IDE、Processing、DJGPP、その他のオープンソースソフトウェアをベースにしています。


86Duino IDE 501+ Logo

QEC.TW website




86Duino IDE 501+

## 86duino最新版のアップデート内容

新メジャーリリースの86Duino IDE 501+ は、より高速でより強力になりました。サードパーティ製 EtherCAT サブデバイスへの対応範囲が拡大され、追加の EthercatDevice クラスが導入されました。これにはCiA-402 規格に準拠するあらゆる EtherCAT サーボドライブを制御するために設計された汎用 CiA-402 サブデバイスクラスも含まれます。

* 備考: **Oriental Motor 6-axis OVR6048K1-V arm** 向けには特別な86Duino IDEビルドを提供します（モデルヘルパーとサンプルを含みます）

# 部品表と要件

ハードウェア

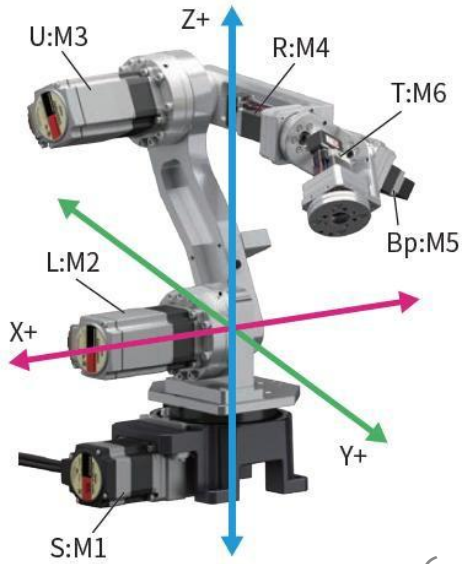
- **EtherCAT Mデバイス** : QEC-M-090T
- **EtherCAT ドライブ** : AZD-KRED x6
- **ロボットアーム & メカニクス** : OVR6048K1-V (モーターは下記の通り)
  - S:  M1:  DGB130R36-AZAKL
  - L:  M2:  AZM66MKH + CSG-20-100-2UH-LW-SP-B
  - U:  M3:  AZM66MKH + CSG-20-100-2UH-LW-SP-B
  - R:  M4:  AZM26AK + CSF-8-50-2UP-SP-A
  - Bp: M5:  AZM24AK + CSF-8-50-2UP-SP-A
  - T:  M6:  AZM24AK + CSF-8-50-2UP-SP-A

Software

- **86Duino IDE 501プレビュー**
  **(オリエンタルモーター社ロボット・アーム・バージョン)**

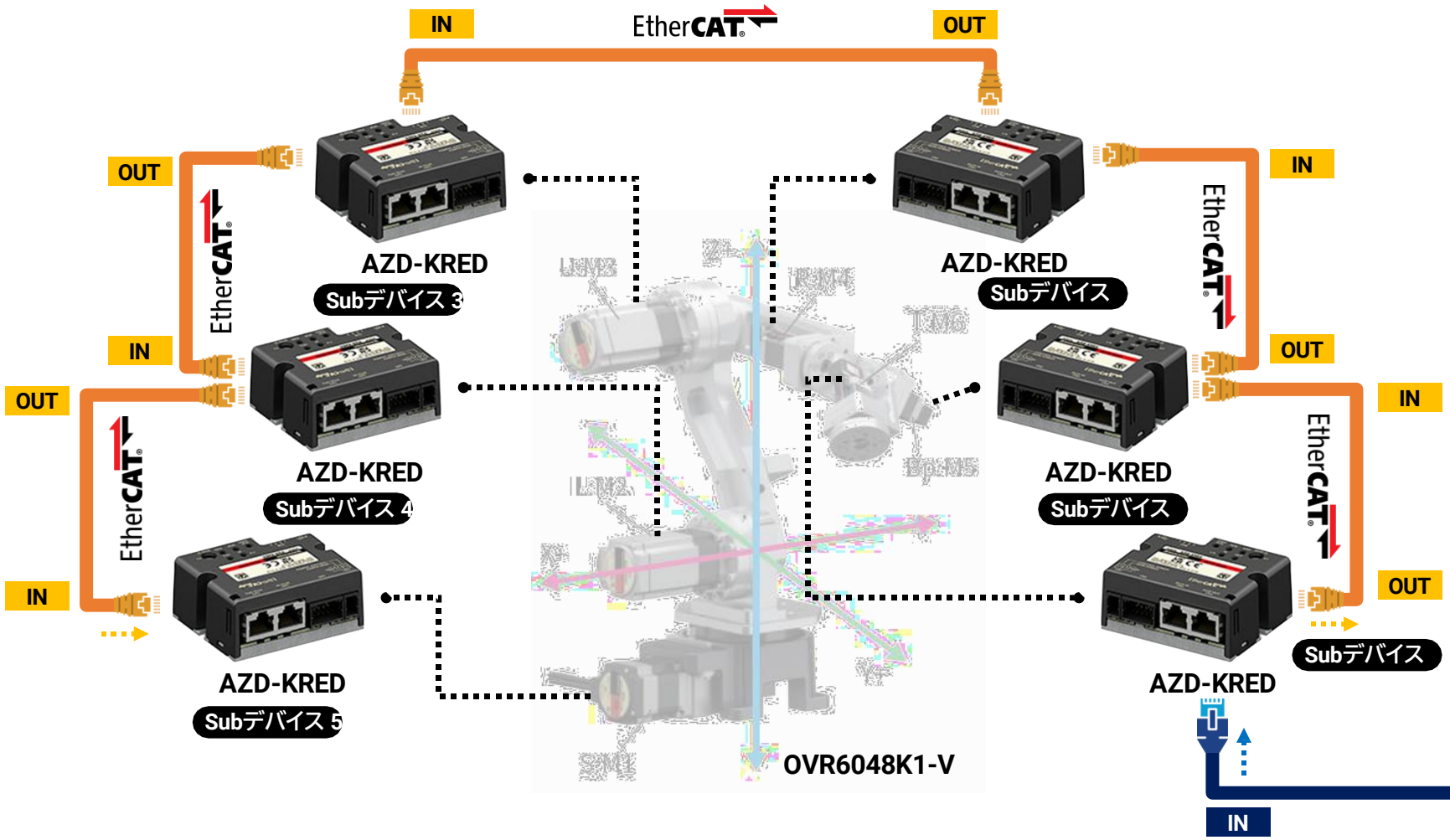
QEC-M-090T

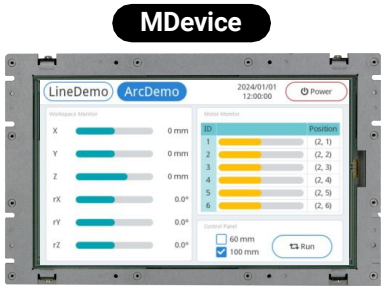

AZD-KRED



OVR6048K1-V

# 配線図




## ノード順序

- **S: M1:** SubDevice 5
- **L: M2:** SubDevice 4
- **U: M3:** SubDevice 3
- **R: M4:** SubDevice 2
- **Bp: M5:** SubDevice 1
- **T: M6:** SubDevice 0

* 位置に基づくアドレス指定(ケーブルの順序によって固定)、スワップしないでください。

# Mデバイス概要｜QEC-M-090T

- **QEC-M-090T**の仕様、配線、機能、安全性については、**必ず当社の公式マニュアルに従ってください:** https://www.qec.tw/product/qec-m-090t/
- このプロジェクトでは、**QEC-M-090T**(EtherCAT Mデバイス・オープン・フレーム、9インチ LCD) を使用します。
- 製品の特長:
  - **リアルタイムデュアルコアCPU**搭載のEtherCAT Mデバイス
  - **86Duino** IDE(統合開発環境)搭載
  - 抵抗膜式タッチスクリーン搭載9インチTFT LCD（800×480）
  - HMI用HDMI出力付き高信頼性SLC eMMCストレージ
  - 産業用動作温度範囲： -20～+70℃
  - **完全なEtherCATスタック**、PDO/SDO、CiA-402（PP/CSP/CSV）、診断機能およびログ機能搭載
  - 電圧、温度、システムステータス監視機能搭載

# ロボットアーム概要｜OVR6048K1-V

- アームの仕様 (寸法、ペイロード、リーチ、制限) については、**常にオリエンタルモーター社の公式資料に従ってください。**
- 製品ページとマニュアル（CAD、取り付け、安全に関する注意事項）を参照してください:
  **OVR 6軸多関節ロボット**
  https://www.orientalmotor.com.tw/tw-zh/products/robots-controllers/ovr-6-axis-articulated-robot
- 統合前に以下の項目をご確認ください:
  - **ペイロード / リーチ**
  - **関節制限** (J1 – J6, 機械的及びソフトウエア制限)
  - **原点/ゼロ定義**
  - **ギア比 & 単位スケーリング**(角度 ↔ パルス、速度・加速度単位)
  - **ツール・エンド・インターフェース**(フランジ仕様、有効I/O/配線)
  - **安全セクション** (非常停止、立入禁止ゾーン、配線ルーティング)

* 注: この資料はQEC統合ワークフローを示すものであり、公式仕様ではありません。
必ずOEMマニュアルを参照してください。

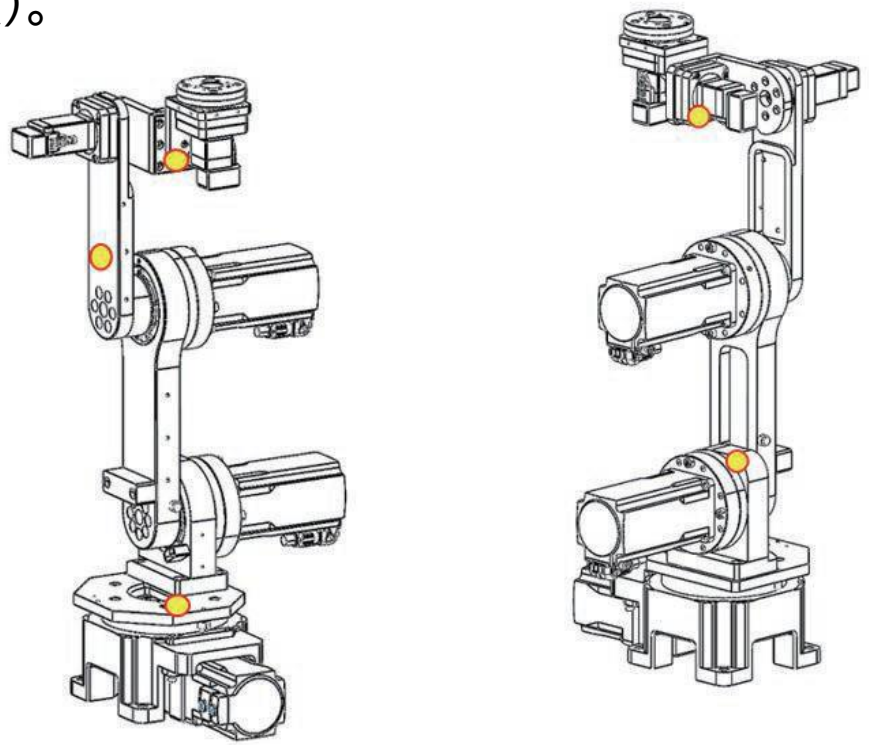
# ドライバ概要｜AZD-KRED

- AZ-miniドライバの仕様、配線、機能、安全性については、**必ずオリエンタルモーター社の公式マニュアル（AZ-miniドライバ）**に従ってください。
  https://www.orientalmotor.com.tw/tw-zh/products/alpha-step/az-mini-driver
- 本プロジェクトは、**AZD-KRED (EtherCAT)**を使います。
- 統合前に以下の点を確認してください：
  - **モデルとインタフェース** (AZ-mini AZD-KRED EtherCAT)
  - **定格電力**および接地/シールド
  - モーター/ブレーキの互換性とコネクタ
  - **ABZO絶対エンコーダ**のゼロ点設定およびバックアップ方法
  - **単位と分解能** (電子ギア、パルス/回転、速度/加速度の単位とリミット)
  - **原点復帰パラメータ** (方式、速度、オフセット、位置決め完了条件)
  - 環境と取付け (温度/振動、ケーブル長、発熱)

*注: このデッキに推奨されるドライバ：AZD-KRED（EtherCAT）

# 初期姿勢

- **目的：** 座標系、軌道生成、およびAPIにおける参照基準として、全関節の標準的な基準姿勢（ゼロ姿勢）を確立します。
- **参照情報：** メーカー仕様書によると、本モデルにはゼロ姿勢を機械的に設定するための位置決めピン穴（φ4h7）が5箇所設けられています（ハイライトされた位置を参照）。
- **いつ:** 初めて使用する前に、およびメンテナンスやドライバの交換後に実行します。
- **手順（推奨）**
    1. 手動（またはティーチ）モードに切り替え、低速で目標姿勢に近づけてください。
    2. リンク機構の整合性を確認するため、5箇所の位置決めピン穴にピンを挿入してください。
    3. ±0.1°の許容範囲を許可します（設定可能）。
    4. 安全（重要）： 停止中または動作中は、ドライバのFREE信号を入力しないでください。モータ電流と保持力が遮断されるため、ロボットが重力の影響で姿勢を崩し、落下する可能性があります



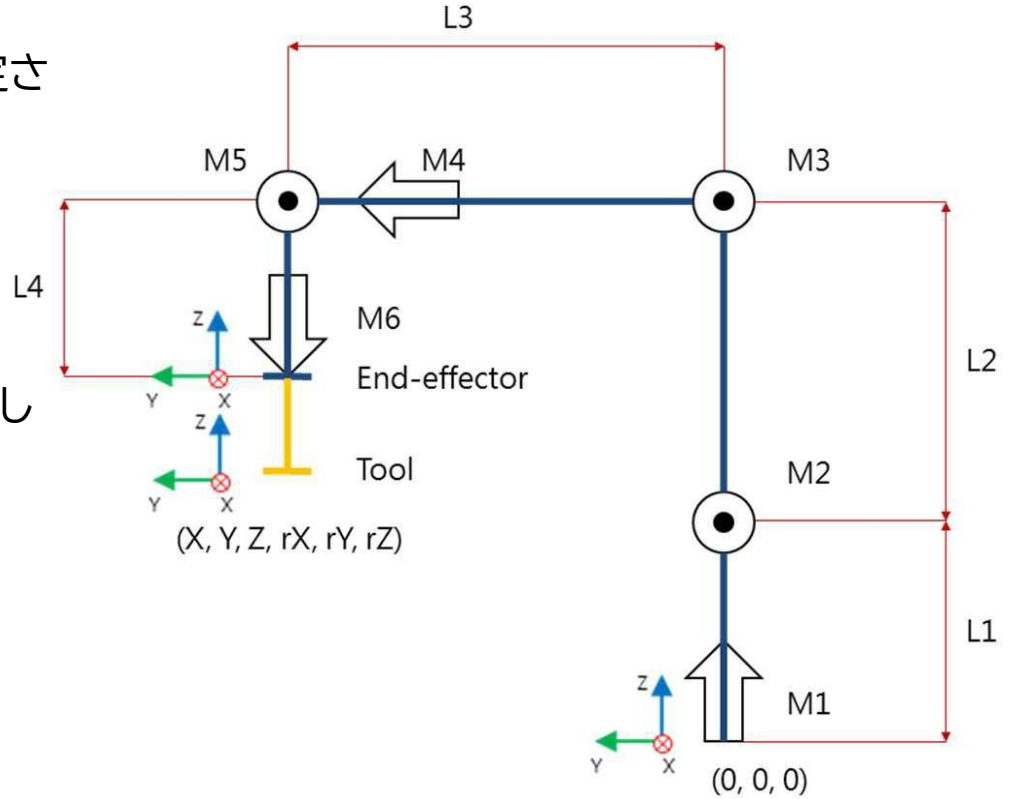
OVR6048K1 – 基準となる姿勢

* 注意: 正確な位置/制限については、OEM マニュアルに従ってください。

# ゼロ姿勢と座標系の取り決め

## ゼロ姿勢

- **基準座標の原点**は、ジョイントM1の中心における（0,0,0）に設定されます。
- J1～J6 を 0° に設定した**標準姿勢**がグローバル基準となります。
- **外的（エクストリンシック）回転順序：**ワールド座標系の X 軸 → Y 軸 → Z 軸の順に回転します。
- **合成行列：**$R\text{world} = Rz(rZ) \cdot Ry(rY) \cdot Rx(rX)$（$X \rightarrow Y \rightarrow Z$の順に回転して構成されます）




ゼロ姿勢・二次元図

OVR6048K1
ゼロ姿勢正面図1

OVR6048K1
ゼロ姿勢正面図2

＊注意: 正確な位置/制限については、OEM マニュアルに従ってください。

# 各軸の分解能

- **AZD-KRED (ドライバ)のデフォルト分解能は、10,000です。**

- OVR6048K1-V軸モデルは次のとおりです（添付画像を参照）

| 軸 | モーター/減速機 | 総比率 | ステップ/回転 | ステップ/角度 | 角度/ステップ |
|---|---|---|---|---|---|
| M1 | DGB130R36-AZAKL | 36 | 360,000 | 1,000.0 | 0.0010° |
| M2 | AZM66MKH + CSG20-100 | 100[1] | 1,000,000 | 2,777.78 | 0.00036° |
| M3 | AZM66MKH + CSG20-100 | 100[1] | 1,000,000 | 2,777.78 | 0.00036° |
| M4 | AZM26AK + CSF8-50 | 50 | 500,000 | 1,388.89 | 0.00072° |
| M5 | AZM24AK + CSF8-50 | 50 | 500,000 | 1,388.89 | 0.00072° |
| M6 | AZM24AK + CSF8-50 | 50 | 500,000 | 1,388.89 | 0.00072° |

| OVR6048K1-V |
|---|
| DGB130R36-AZAKL |
| AZM66MKH-CSG20-100 |
| AZM66MKH-CSG20-100 |
| AZM26AK-CSF8-50 |
| AZM24AK-CSF8-50 |
| AZM24AK-CSF8-50 |

# 2. 軌道表示ツール

**OVR6048K1版**

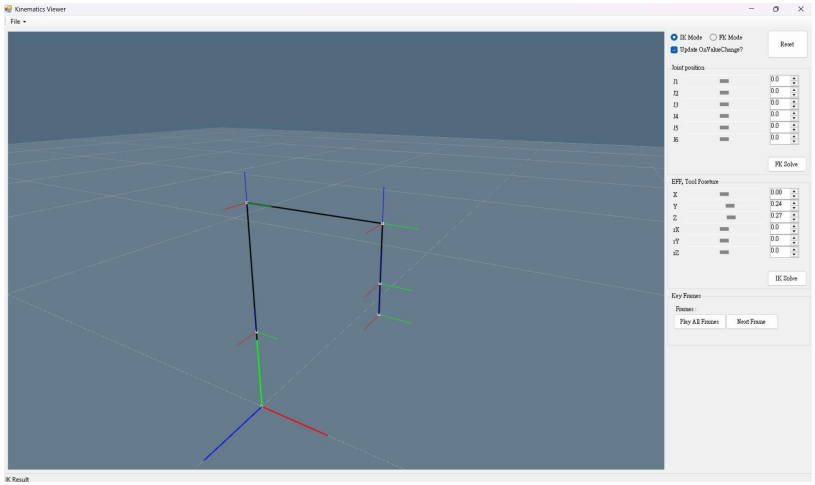
# 軌道表示ツール紹介 - 1

## 何か?

- 軽量な3Dシミュレータで、OVR6048K1のリンク長、ゼロ姿勢、軸定義をプリセットします。
- FK(順運動学)／IK(逆運動学)に対応し、姿勢と可達性をリアルタイムに可視化します。

## なぜ、使うか

- 実機ロボットを動作させる**前に**、ポーズ、経路、特異点を検証します。
- ウェイポイントとタイミングを生成し、それらをQECプログラム／HMIへ渡します。




軌道表示ビューア

# 軌道表示ツール紹介 - 2

## 特徴

- モード：IK / FK 切り替え、オプションで即時再計算のための **OnValueChange** を更新します。
- ポーズ I/O：**J1～J6** または **TCP (X、Y、Z、rX、rY、rZ)** を編集し、どちらかの方法で解決します。
- キーフレーム：**次のフレーム**または**全フレーム**再生でステップで実行します。
- パスのインポート：testpath.txt を読み込み、フルモーションのタイムラインをシミュレートします。

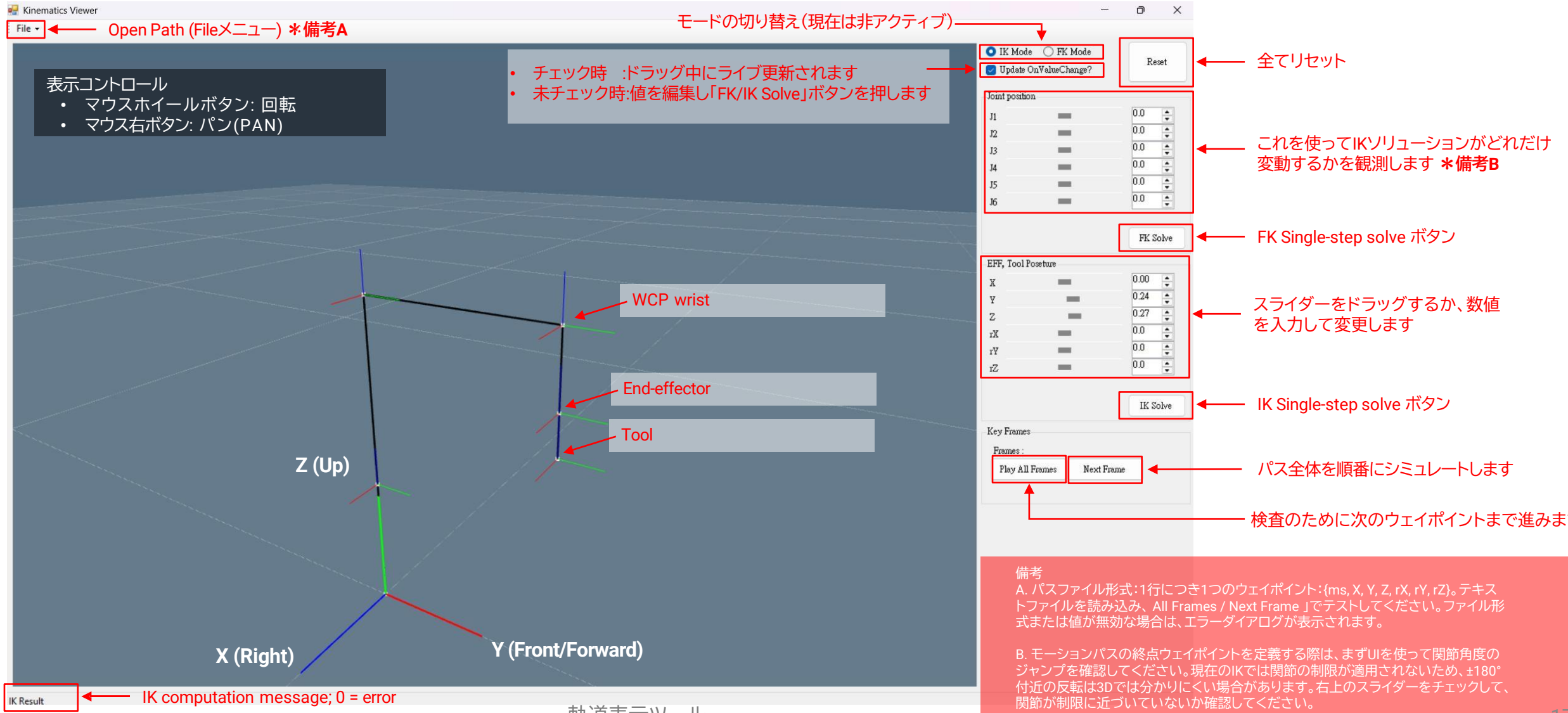
## 規約

- 外部回転順序：X → Y → Z（**ワールド座標系**に基づく）
- ゼロ姿勢はOVR6048K1基準に合わせて調整済みです。単位：**位置はmまたはmm**、角度は**度（deg）**。

## クイック操作

- マウス操作：　左ドラッグで回転、中央ドラッグでパン（移動）、ホイールでズーム。
- アクション：　**FKソルブ / IKソルブ、リセット。**

# 軌道表示ツール UI機能



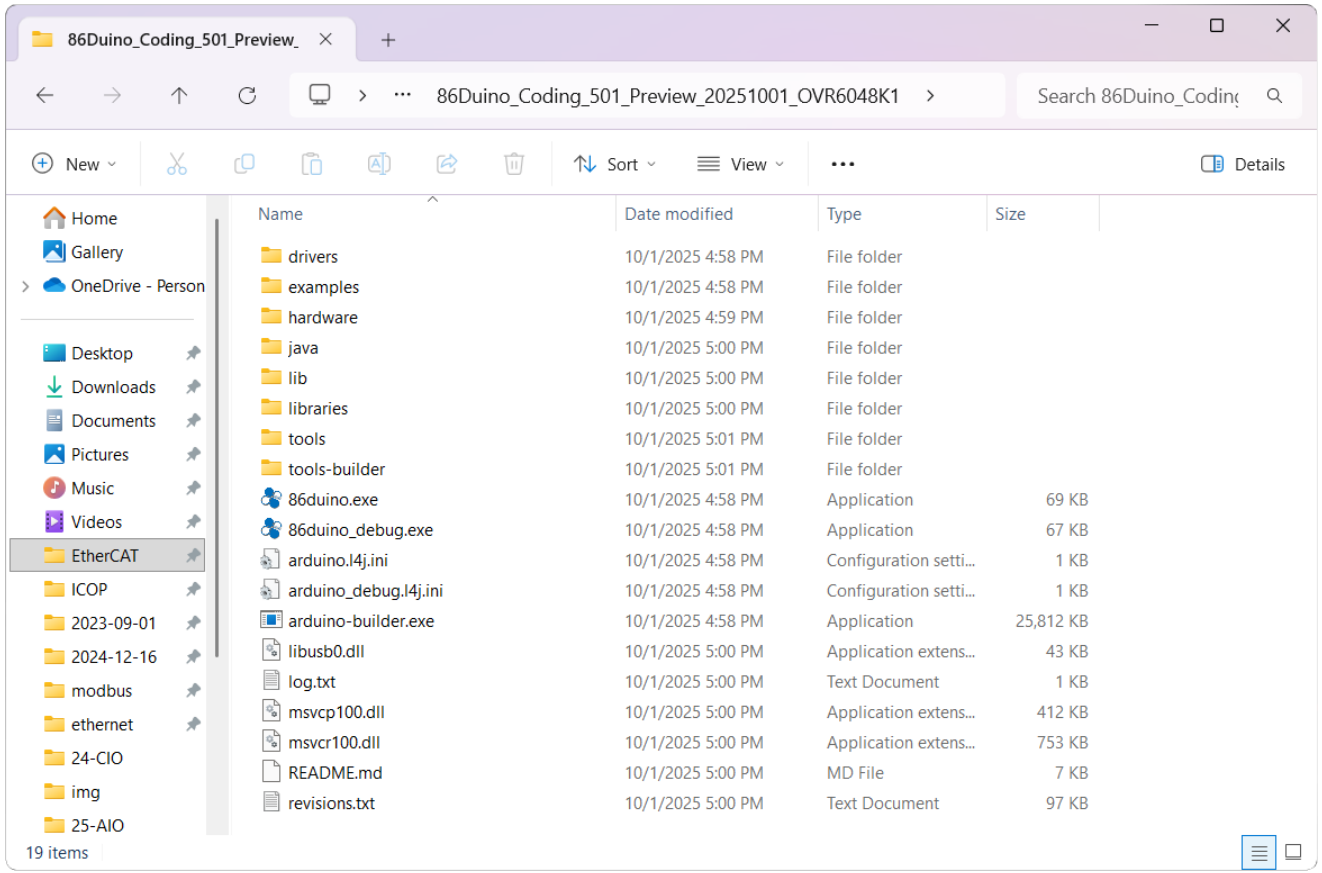

備考
A. パスファイル形式：1行につき1つのウェイポイント：{ms, X, Y, Z, rX, rY, rZ}。テキストファイルを読み込み、All Frames / Next Frame 」でテストしてください。ファイル形式または値が無効な場合は、エラーダイアログが表示されます。

B. モーションパスの終点ウェイポイントを定義する際は、まずUIを使って関節角度のジャンプを確認してください。現在のIKでは関節の制限が適用されないため、±180°付近の反転は3Dでは分かりにくい場合があります。右上のスライダーをチェックして、関節が制限に近づいていないか確認してください。

# 軌道表示ツールの起動 - 1

## ステップ1 – IDEフォルダを開く

- 86Duino IDE ディレクトリ (例: …¥86Duino_Coding_501_Preview_20251001_OVR6048K1) に移動します。

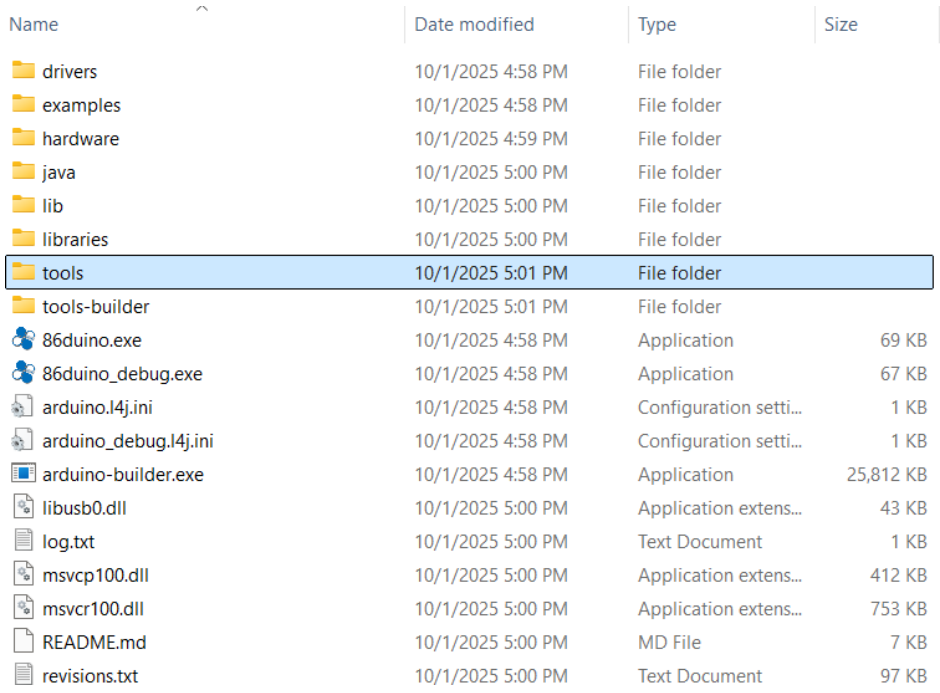



軌道表示ツールのファイルパス

# 軌道表示ツールの起動 - 2

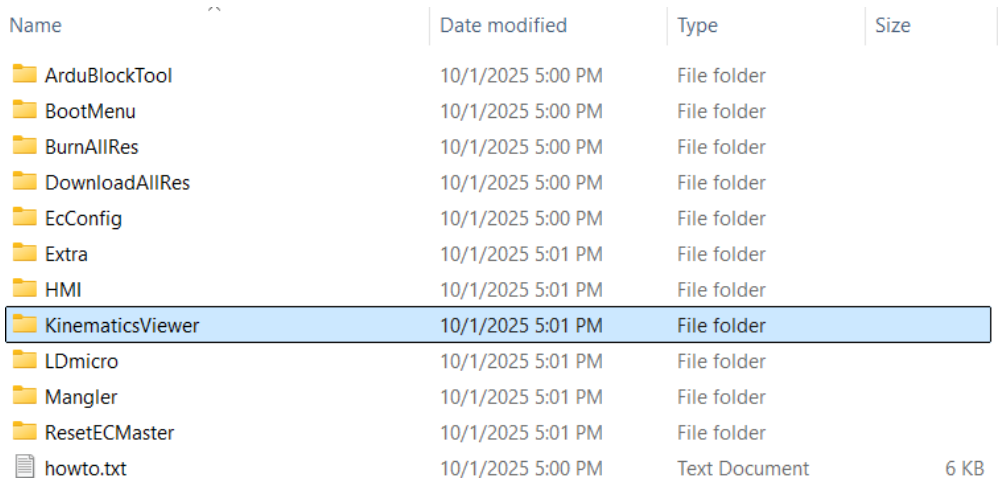
## ステップ2 – ツールに移動する

- 開く: tool → KinematicsViewer



| Name | Date modified | Type | Size |
|---|---|---|---|
| drivers | 10/1/2025 4:58 PM | File folder | |
| examples | 10/1/2025 4:58 PM | File folder | |
| hardware | 10/1/2025 4:59 PM | File folder | |
| java | 10/1/2025 5:00 PM | File folder | |
| lib | 10/1/2025 5:00 PM | File folder | |
| libraries | 10/1/2025 5:00 PM | File folder | |
| tools | 10/1/2025 5:01 PM | File folder | |
| tools-builder | 10/1/2025 5:01 PM | File folder | |
| 86duino.exe | 10/1/2025 4:58 PM | Application | 69 KB |
| 86duino_debug.exe | 10/1/2025 4:58 PM | Application | 67 KB |
| arduino.l4j.ini | 10/1/2025 4:58 PM | Configuration setti... | 1 KB |
| arduino_debug.l4j.ini | 10/1/2025 4:58 PM | Configuration setti... | 1 KB |
| arduino-builder.exe | 10/1/2025 4:58 PM | Application | 25,812 KB |
| libusb0.dll | 10/1/2025 5:00 PM | Application extens... | 43 KB |
| log.txt | 10/1/2025 5:00 PM | Text Document | 1 KB |
| msvcp100.dll | 10/1/2025 5:00 PM | Application extens... | 412 KB |
| msvcr100.dll | 10/1/2025 5:00 PM | Application extens... | 753 KB |
| README.md | 10/1/2025 5:00 PM | MD File | 7 KB |
| revisions.txt | 10/1/2025 5:00 PM | Text Document | 97 KB |

軌道表示ツールのファイルパス 2



| Name | Date modified | Type | Size |
|---|---|---|---|
| ArduBlockTool | 10/1/2025 5:00 PM | File folder | |
| BootMenu | 10/1/2025 5:00 PM | File folder | |
| BurnAllRes | 10/1/2025 5:00 PM | File folder | |
| DownloadAllRes | 10/1/2025 5:00 PM | File folder | |
| EcConfig | 10/1/2025 5:00 PM | File folder | |
| Extra | 10/1/2025 5:01 PM | File folder | |
| HMI | 10/1/2025 5:01 PM | File folder | |
| KinematicsViewer | 10/1/2025 5:01 PM | File folder | |
| LDmicro | 10/1/2025 5:01 PM | File folder | |
| Mangler | 10/1/2025 5:01 PM | File folder | |
| ResetECMaster | 10/1/2025 5:01 PM | File folder | |
| howto.txt | 10/1/2025 5:00 PM | Text Document | 6 KB |

軌道表示ツールのファイルパス 3

# 軌道表示ツールの起動 - 3

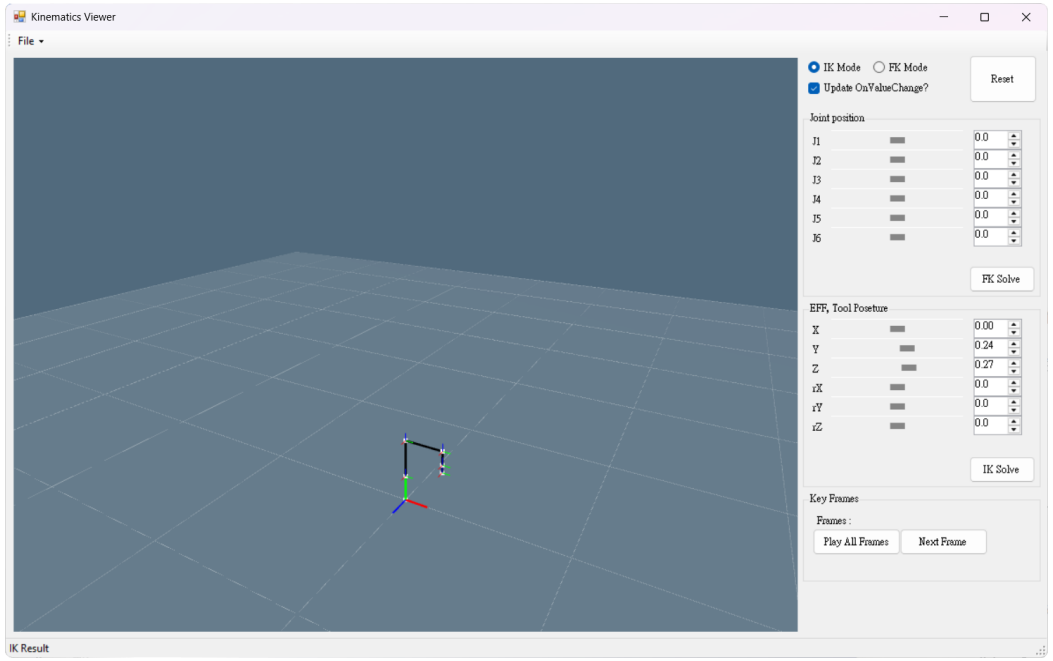
## ステップ3 – 表示ツールを実行する

- **KinematicsViewer.exe**.をダブルクリックします。

| Name | Date modified | Type | Size |
|---|---|---|---|
| all.txt | 10/1/2025 5:01 PM | Text Document | 12 KB |
| all2.txt | 10/1/2025 5:01 PM | Text Document | 12 KB |
| BulletSharp.dll | 10/1/2025 5:01 PM | Application extens... | 2,101 KB |
| glew32.dll | 10/1/2025 5:01 PM | Application extens... | 383 KB |
| glut32.dll | 10/1/2025 5:01 PM | Application extens... | 232 KB |
| KinematicsViewer.exe | 10/1/2025 5:01 PM | Application | 2,082 KB |
| step1_CW_xz_x_-ry.txt | 10/1/2025 5:01 PM | Text Document | 3 KB |
| step2_CW_xy_-x_rx-y.txt | 10/1/2025 5:01 PM | Text Document | 2 KB |
| step3_CW_xy_y_r-x-y.txt | 10/1/2025 5:01 PM | Text Document | 2 KB |
| step4_CW_xy_x_r-xy.txt | 10/1/2025 5:01 PM | Text Document | 2 KB |
| step5_CCW_yz_-y_-rx.txt | 10/1/2025 5:01 PM | Text Document | 3 KB |
| step6_CW_xy_y_-rxy.txt | 10/1/2025 5:01 PM | Text Document | 2 KB |
| testPath.txt | 10/1/2025 5:01 PM | Text Document | 1 KB |
| testPath_withError.txt | 10/1/2025 5:01 PM | Text Document | 1 KB |

軌道表示ツールのファイルパス 4

軌道表示ツール2

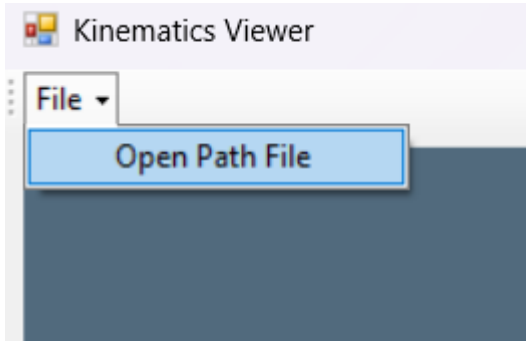



実行ファイルとともにいくつかのサンプルの .txt ファイルが表示されます。

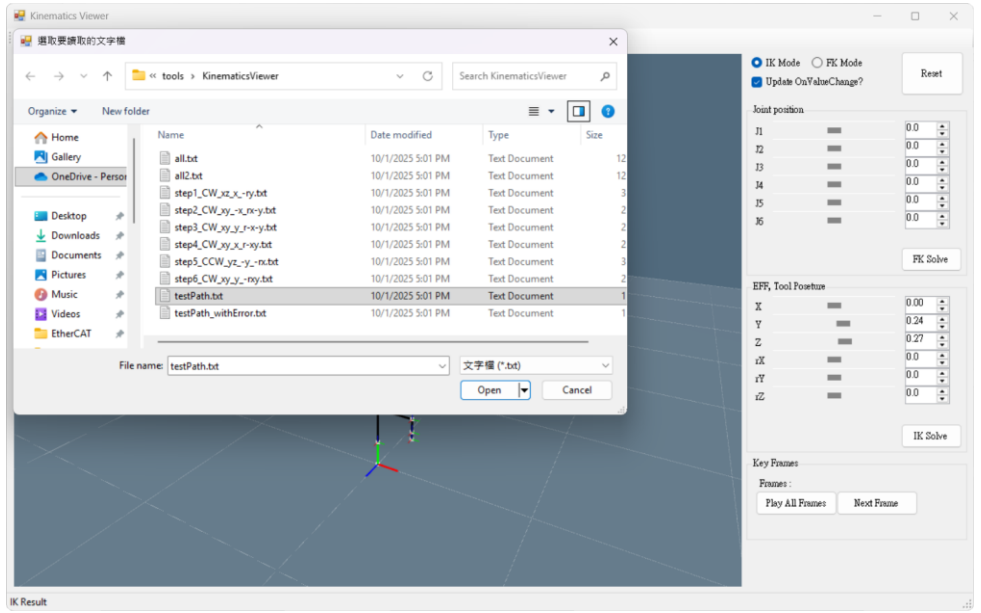
# 軌道表示ツールの起動 - 4

## ステップ4 – モーションパスを読み込む

- 表示ツールにて: **File → Import/Open Path... → testPath.txt**を選択します。
  *(または、all.txt、all2.txt、step1_...txt などの他のサンプルファイルを選択します。)*
- 軌道をシミュレートするには、**Play All Frames / Next Frame** を使います。

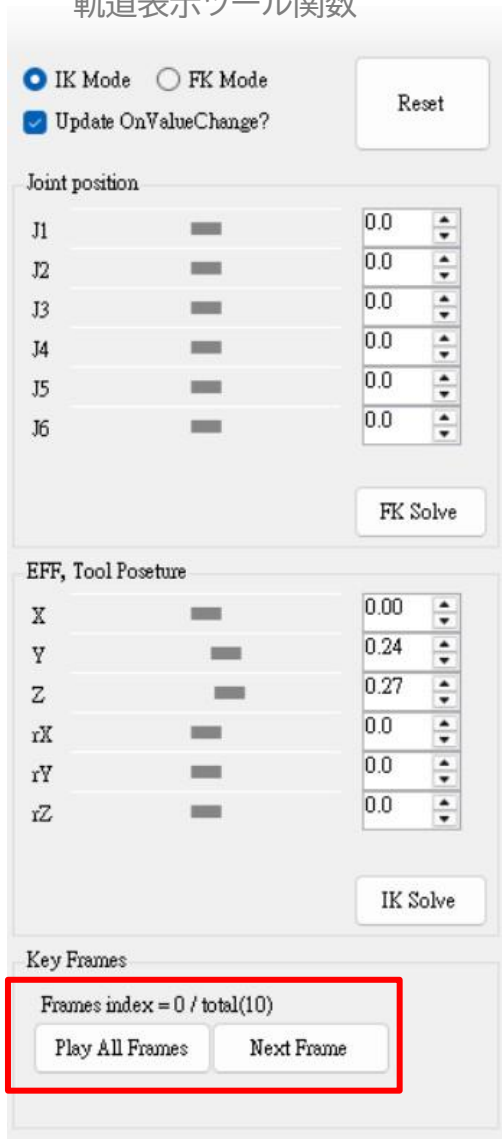



軌道表示ツール– ファイルオープン 1




軌道表示ツール– ファイルオープン 2

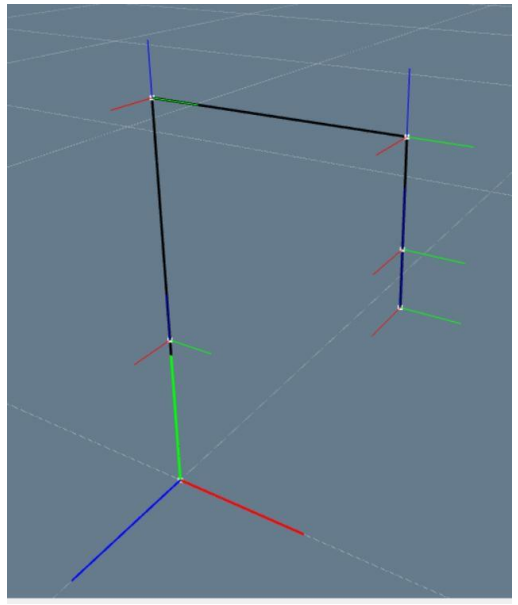
軌道表示ツール関数



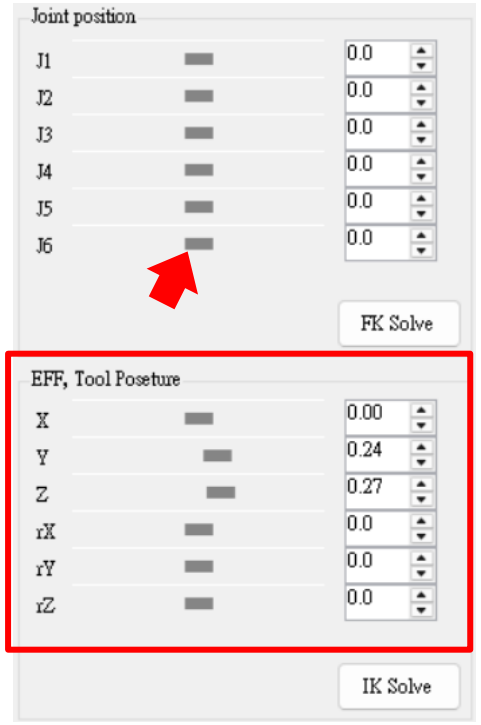
# 軌道表示ツールの起動 - 5

## ステップ5 – 値を変更する

ビューアでは、「 **Joint position** 」または「 **EFF, Tool　Poseture** 」エリアのバーまたは入力値をドラッグすることで、ビューウィンドウ上でロボットの動作をシミュレートできます。シミュレーション終了後は、「 **EFF, Tool Poseture** 」エリアの値と対応するAPI関数への入力値を使用して、軌道表示ツールと同じ位置にアームを制御できます。



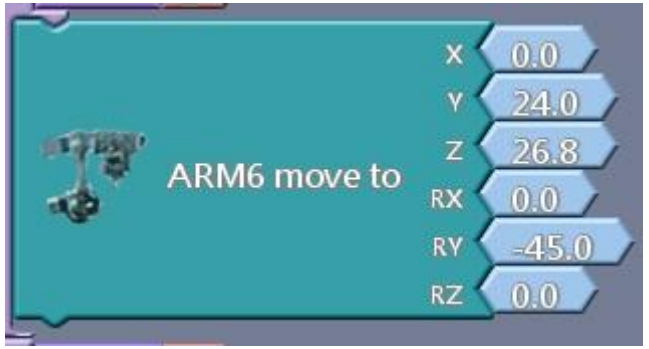
ビューワウインドウ




運動制御ビューラ関数

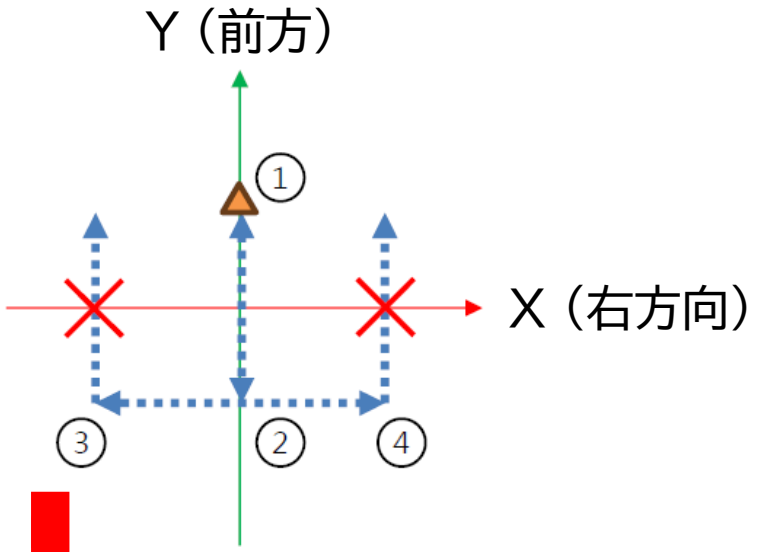
```
if (Hmi.buttonClicked(p1b4)) {
  Set_Des_Position(0.0, 24.0, 26.8, 0.0, -45.0, 0.0);
  prepare_move();
}
if (Hmi.buttonClicked(p1b5)) {
  Set_Des_Position(0.0, 24.0, 26.8, 0.0, -20.0, 0.0);
  prepare_move();
}
if (Hmi.buttonClicked(p1b6)) {
  Set_Des_Position(0.0, 24.0, 26.8, 0.0, 20.0, 0.0);
  prepare_move();
}
if (Hmi.buttonClicked(p1b7)) {
  Set_Des_Position(0.0, 24.0, 26.8, 0.0, 45.0, 0.0);
  prepare_move();
}
```

Kinematics Example Code

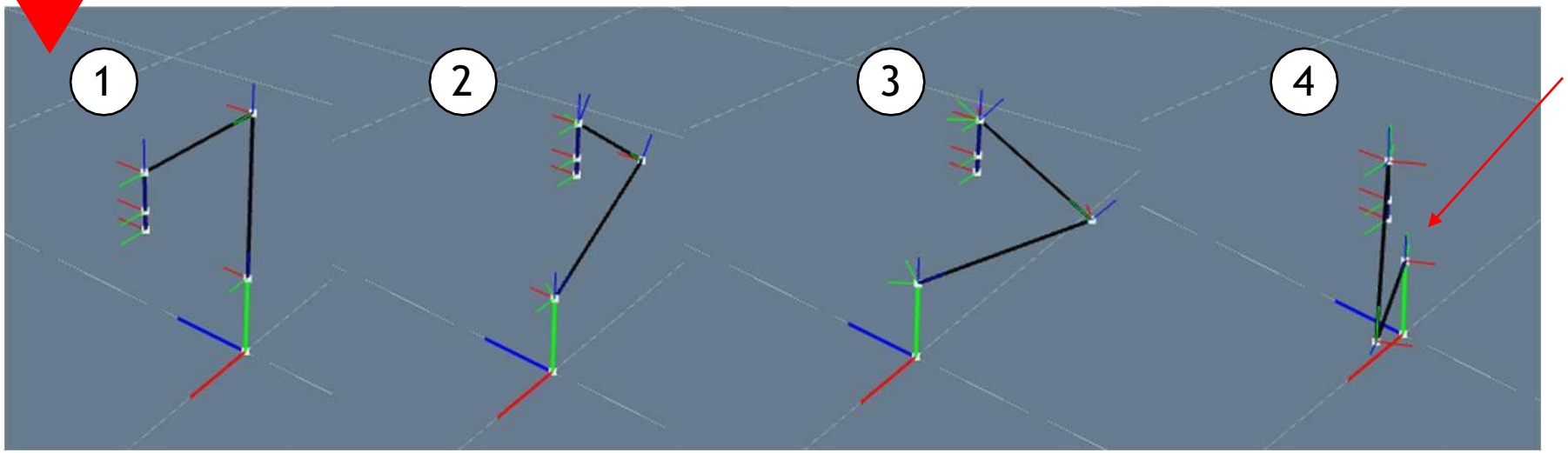



ArduBlock：ロボットアームブロックの移動

# 避けるべき経路計画- 1

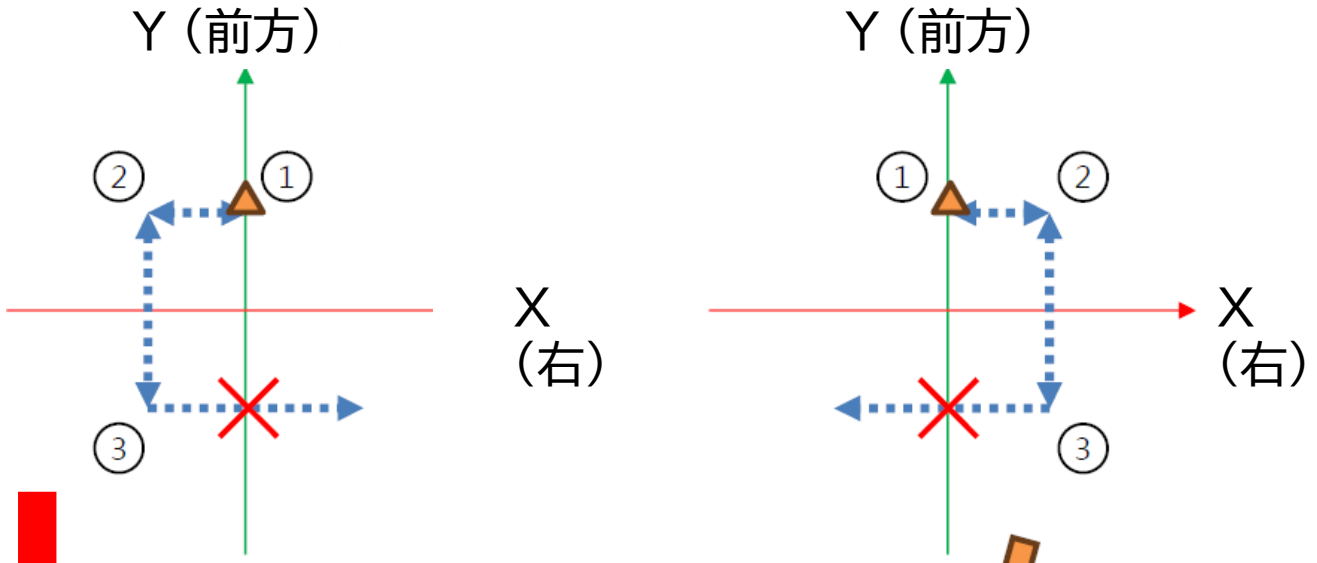



*注意：一部の動作経路では、ベースが回転限界付近まで動く可能性があります。X印が付けられた象限をまたぐ経路は避け、元の経路の象限に戻す必要があります。

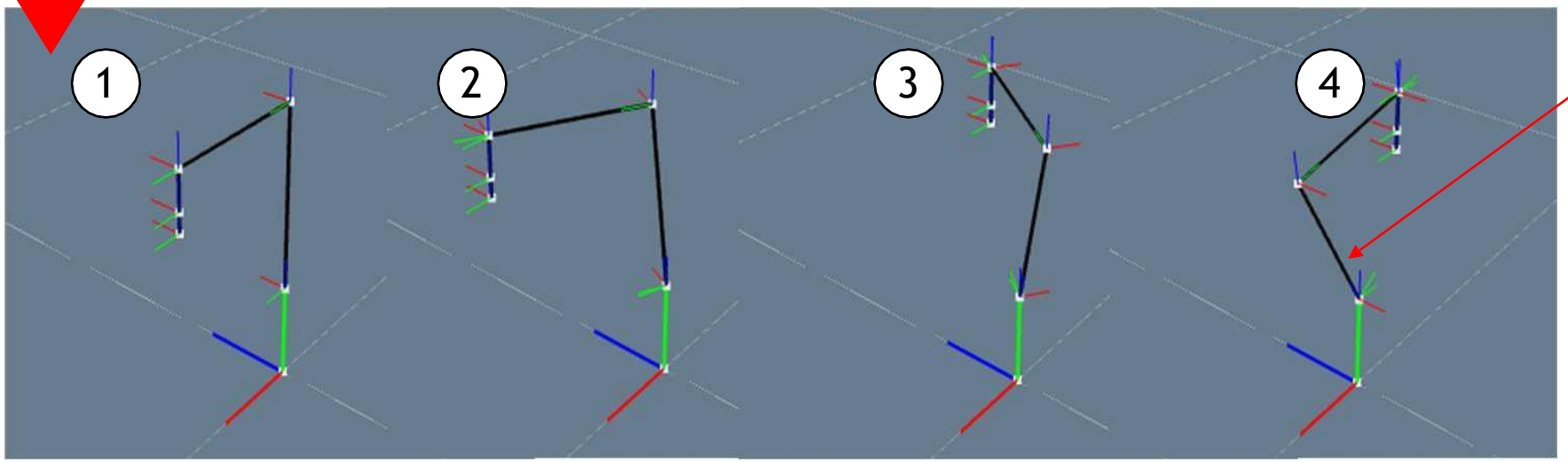



運動制御が経路を回避する-1

# 避けるべき経路計画- 2

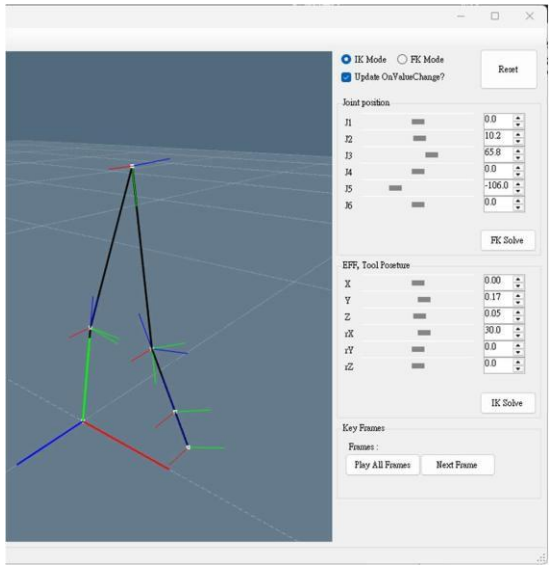



*注意:一部の動作経路では、ベースが回転限界付近まで動く可能性があります。X印が付けられた象限をまたぐ経路は避け、元の経路の象限に戻す必要があります。

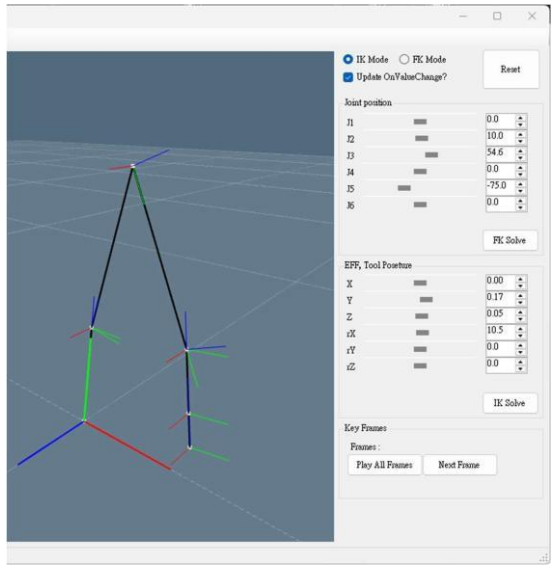



運動制御が経路を回避する - 2

# 避けるべき経路計画- 3




運動制御が経路を回避する- 3




運動制御が経路を回避する- 4

手首の回転を確認してください。X=0平面では、**ジョイント5**は-90度に近い位置で問題ありません。

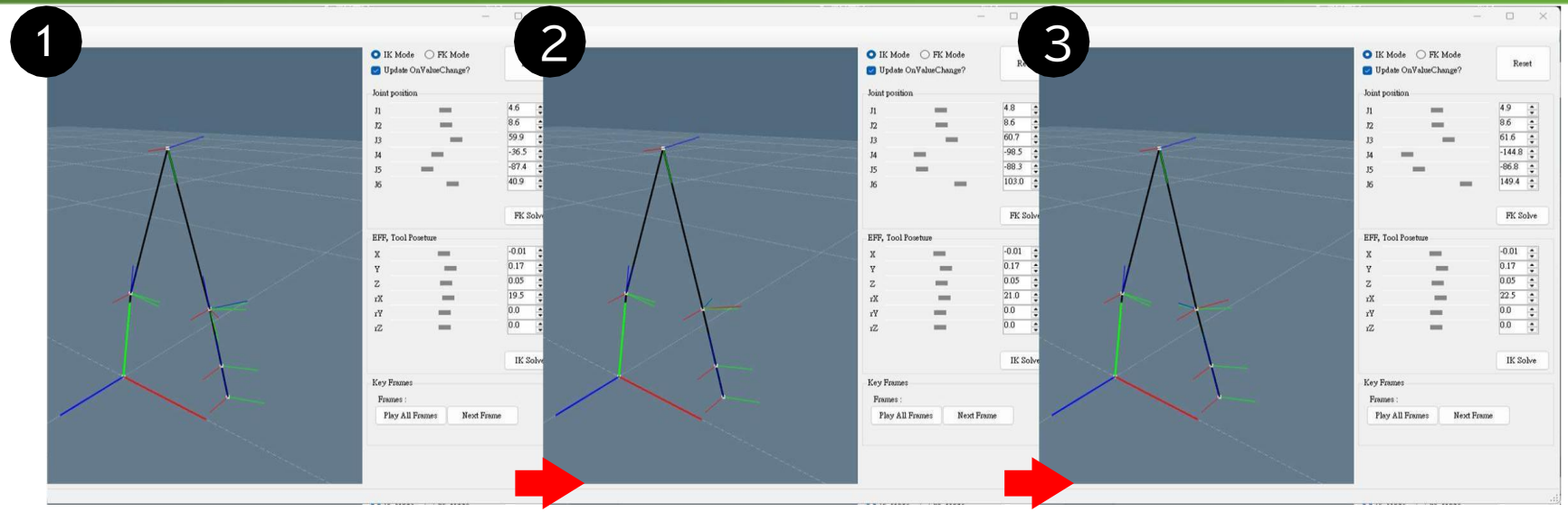
ただし、X が X=0 平面からわずかにずれている場合 (たとえば、X=-0.01)、J5を +/-90 度に近づけると、Theta 4はジッタを減し、変動を最小限に抑えるためTheta 4/6は端に向かって移動します。

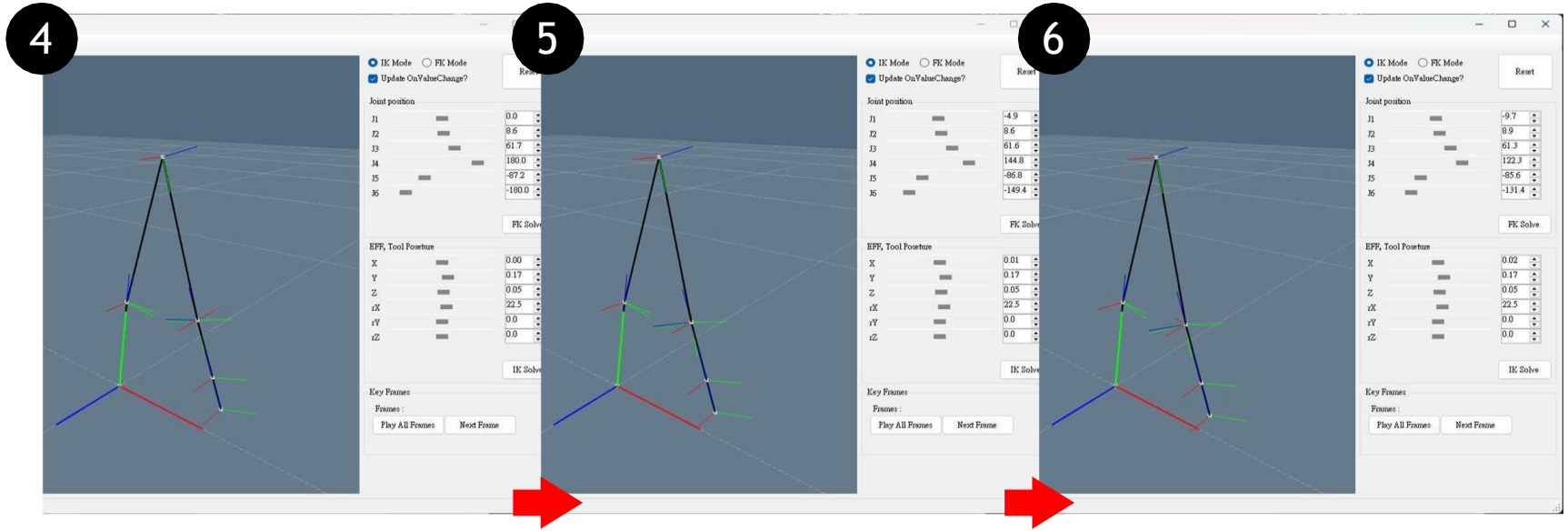
この時点で、終了位置Xが象限をまたいで移動すると(例えば、X=-0.01 0.01)、Theta 4とTheta 6は+/-180度移動します。したがって、J5が+/-90度に近い場合は、同様の動きをしないように注意してください。



(次ページに続く)

# 避けるべき経路計画- 4




運動制御が経路を回避する- 5

# 3. 使い方？

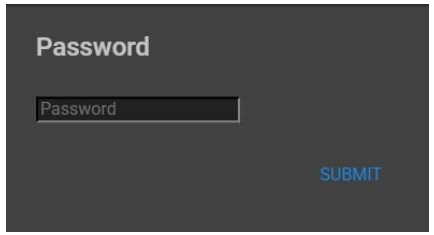
APIの紹介、コーディング方法、グラフィクツール

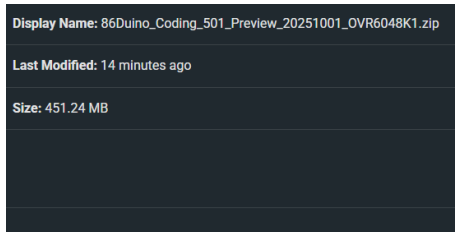
# ソフトウェアの概要とダウンロード

**オリエンタルモーター社ロボットアーム(6軸)**向けに、**86Duino IDEの特別バージョン**をリリースしました。
OVR6048K1-V特有のモーション、運動制御、ビューアをサポートしています。

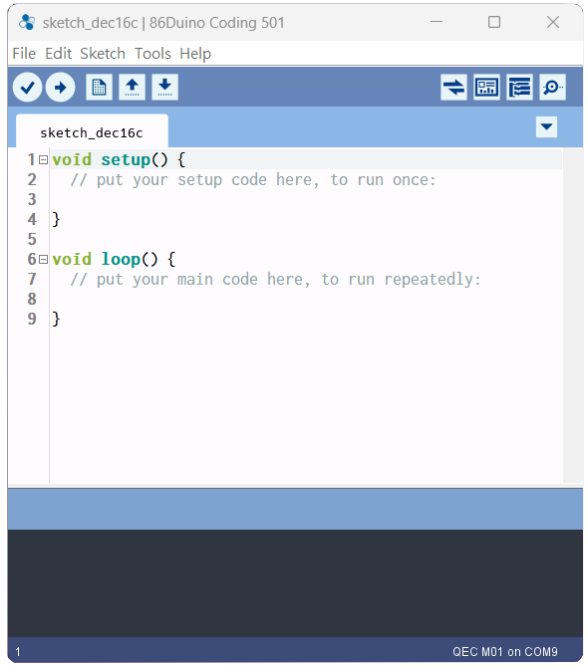
ここから86duino IDEはダウンロードできます：https://ftp.dmp.com.tw/share/Ch_rBIbF

- パスワード：icop
- ファイル名：86Duino_Coding_501_Preview_20251001_OVR6048K1.zip







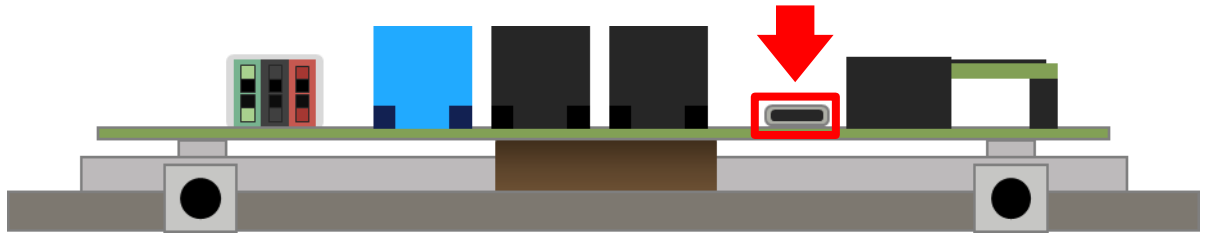

86Duino IDE 501の新しい主リリースは、これまで以上に高速で強力になりました！ 今回のバージョンでは、より幅広いサードパーティ製 EtherCAT Subデバイスに対応し、さらに EthercatDeviceクラスが追加されました。中でも、CiA402規格に準拠した任意のEtherCATサーボドライブを制御できる汎用のCiA 402 EtherCAT Subデバイスクラスが導入されています。

- **EtherCATライブラリAPIユーザーマニュアル：**
  https://www.qec.tw/ethercat/api/ethercat-library-api-user-manual/

# 開発環境のセットアップ- 1

環境を設定するには、以下の手順に従います:

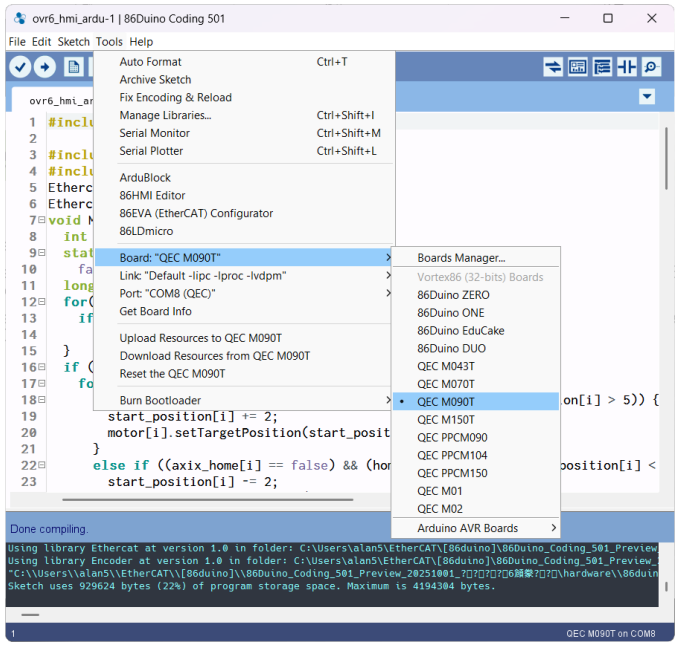
1. QEC-M-090TをUSB Type-C - USBケーブルでPCに接続します（86Duino IDEがインストールされたPC）。
2. QECの電源を入れます。
3. PC で「**Device Manager**」（Win+X を押した後のメニューで選択）ー＞「**Ports (COM & LPT)**」を開きます。「**Prolific PL2303GC USB Serial COM Port (COMx)**」が検出されているはずです。検出されない場合は、必要なドライバをインストールする必要があります。
（Windows PL2303ドライバは、ここからダウンロードできます。）



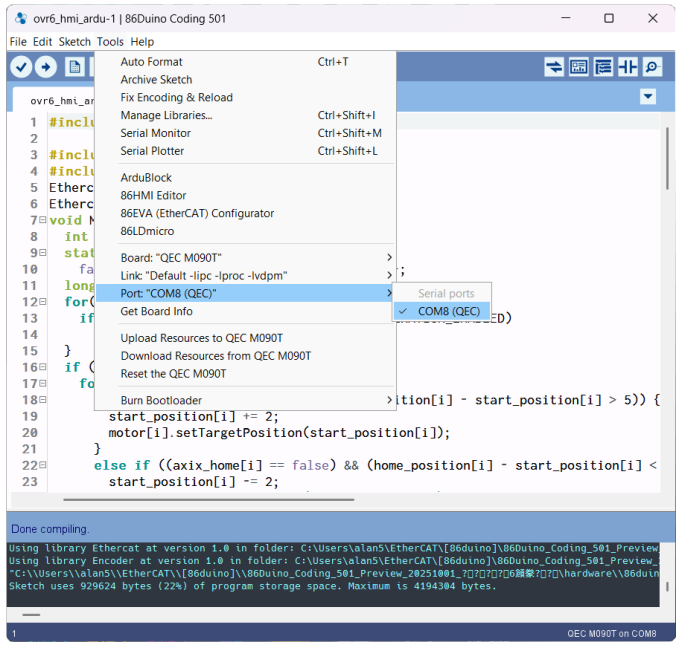
QEC-M オープンフレームボード側面図

# 開発環境のセットアップ- 2

4. 86Duino IDEを開きます。
5. 適切なボードを選択してください： IDEのメニューで、「 **Tools** 」>「 **Board** 」>「 **QEC-M090T** 」（または使用しているQEC Mデバイスモデル）を選択します。
6. ポートの選択： IDEメニューで「 **Tools** 」>「 **Port** 」を選択し、QEC Mデバイスに接続するUSBポート（この場合はCOM8（QEC））を選択します。




Tool - Board



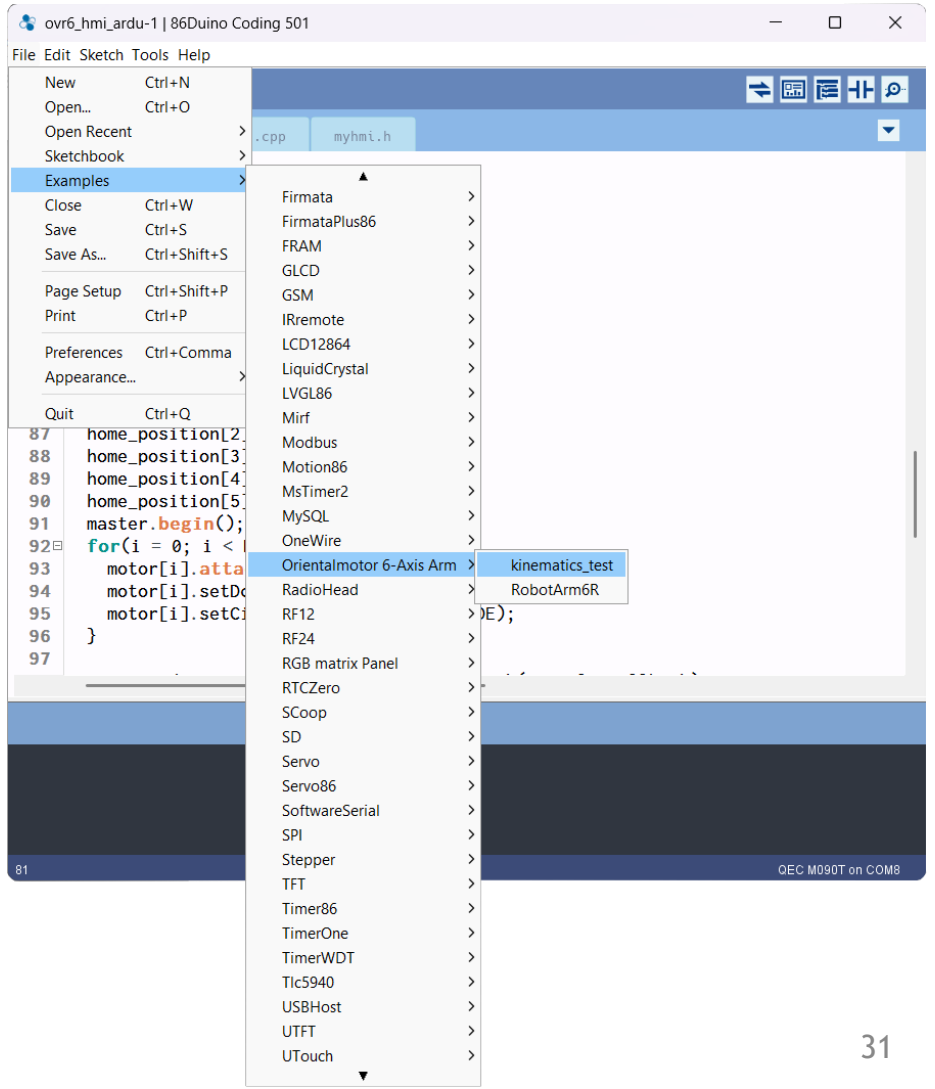

Tool - Port

# 例：運動制御テスト - 1

## ステップ1ーサンプルを開く

- 「 Tools ▸ Board 」で「 **QEC-M090T** 」を選択し、適切なポートを選択すると、サンプルセットが表示されます。
- 「 File ▸ Examples ▸ Orientalmotor 6-Axis Arm ▸kinematics_test 」を開きます。

86Duinoの例



# 例：運動制御テスト – 2

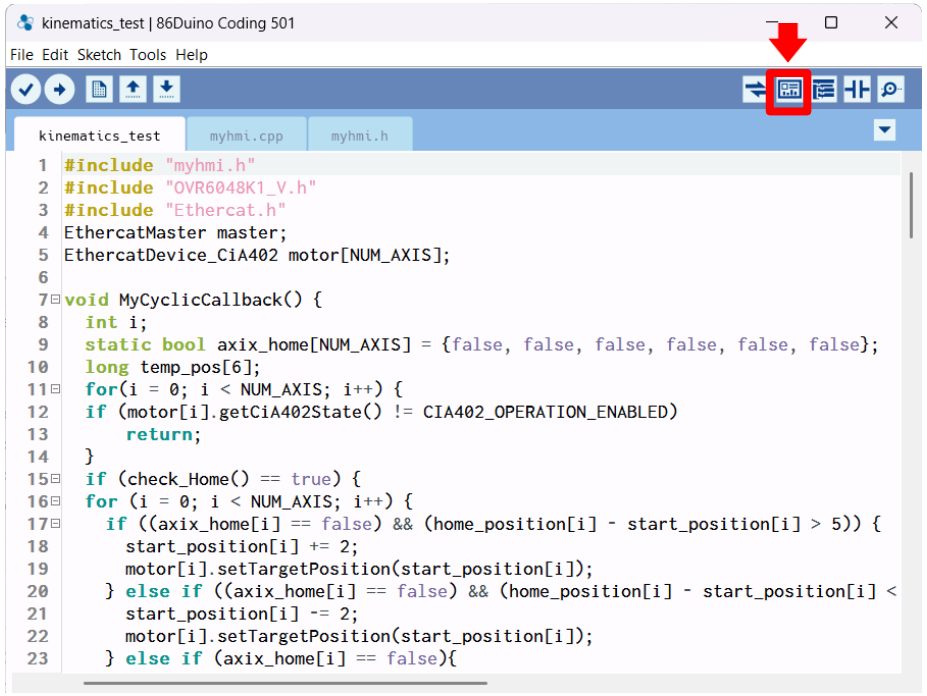
## ステップ2 - コードを表示してHMIを開く

- kinematics_test を開くとスケッチが読み込まれます（右図）。下記の3つのライブラリを使用します：
  - **myhmi.h** – HMIヘルパ
  - **OVR6048K1_V.h** – ロボット・モーション・ヘルパ
  - **Ethercat.h** – EtherCATプロトコル
- この例では、次の設定を使用します：
  - EtherCATサイクルタイム：**250マイクロ秒(µs)。**
  - EtherCATモード： **ECAT_SYNC。**
  - 分散クロック(DC): **Enable (250 µs)。**
  - CiA-402モード： **CSP (周期同期位置)。**

```
91   master.begin();
92   for(i = 0; i < NUM_AXIS; i++) {
93     motor[i].attach(i, master);
94     motor[i].setDc(cycle_time*10);
95     motor[i].setCiA402Mode(CIA402_CSP_MODE);
96   }
97
98   error_code = master.attachRTCyclicCallback(MyCyclicCallback);
99   if (error_code != 0) {
100   printf("error : %d\n", error_code);
101  }
102  master.start(cycle_time*10, ECAT_SYNC); // cycle_time = 25000; // nanosecond
```

Kinematics_test.ino EtherCATに関するコード

- 86HMIアイコン（右上）をクリックして、HMIエディタを起動します。

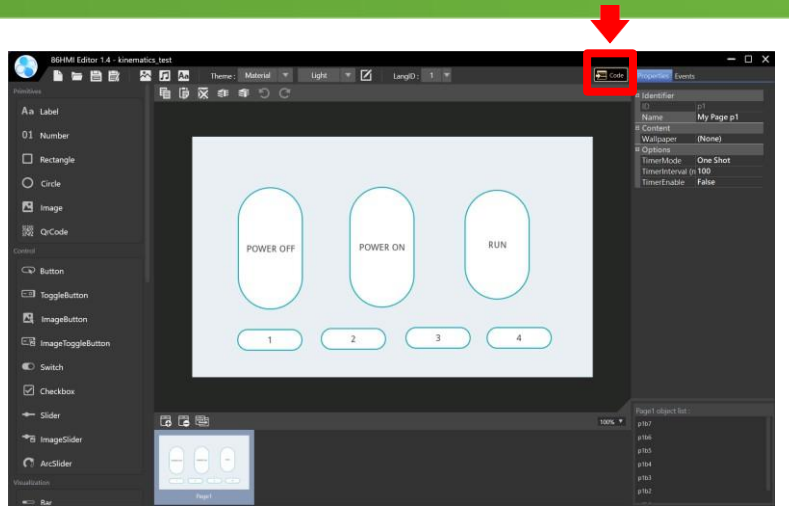



Kinematics_test.ino

# 例：運動制御テスト – 3

## ステップ3 – 86HMIレイアウトとコード生成

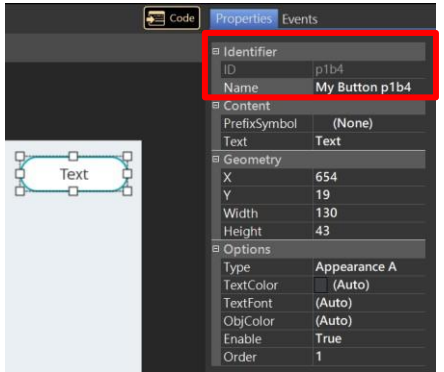
- サンプルHMIには、power_on、power_off、run、move1～move4（右図参照）の**7つのボタン**が含まれています。次のセクションでは、それぞれのボタンをコードにマッピングします。
- 「**Code**」ボタン (右上) をクリックすると、イベントスタブとincludeステートメントが生成され、メインスケッチに貼り付けられます。
- 生成後、**myhmi.h** と **myhmi.cpp** が表示されます。これらはLVGL86ライブラリのヘルパと実行プログラムを表します。

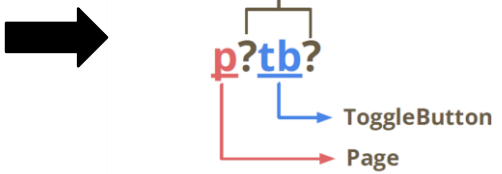



Kinematics_test HMIの例


86HMI生成ファイル

86HMI ID命名規則



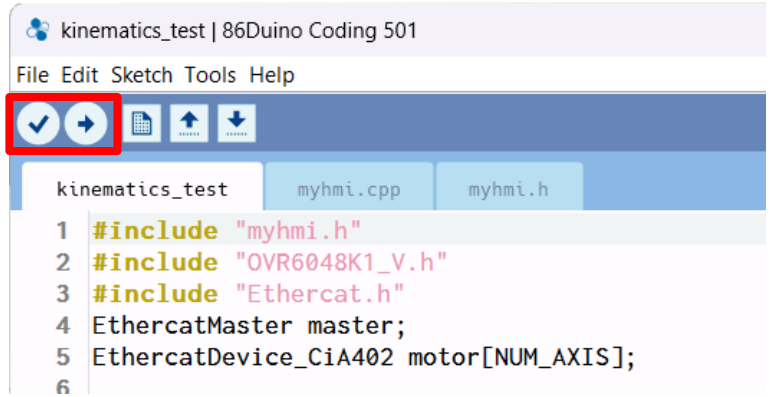



* 追加情報: 86HMIでは、すべてのオブジェクトに一意のIDがあり、命名規則は右の図に示すように、クラスシリアル番号を含むページ番号です。

# 例：運動制御テスト － 4

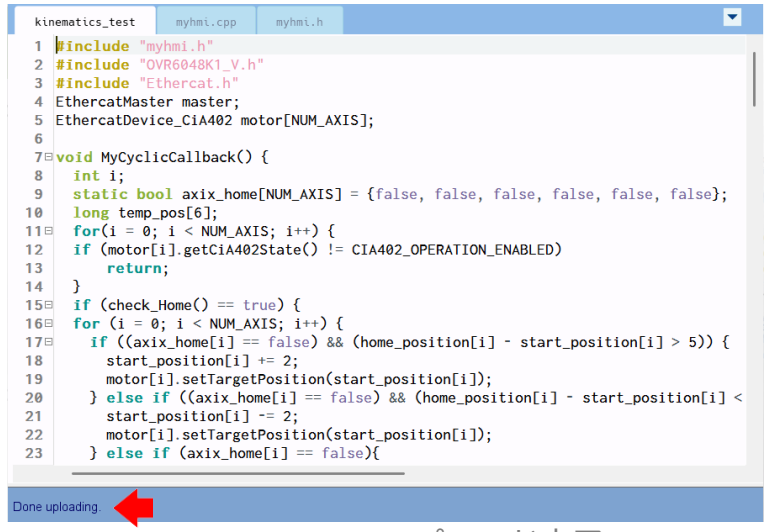
## ステップ4 － ビルドとアップロード

コードを記述したら、ツールバーの  をクリックしてコンパイルし、コンパイルが完了してエラーがないことを確認後、  をクリックしてアップロードします。




86Duinoアップロードと検証アイコン

アップロードが完了すると、下のウィンドウに「**Done uploading**」と表示されます。

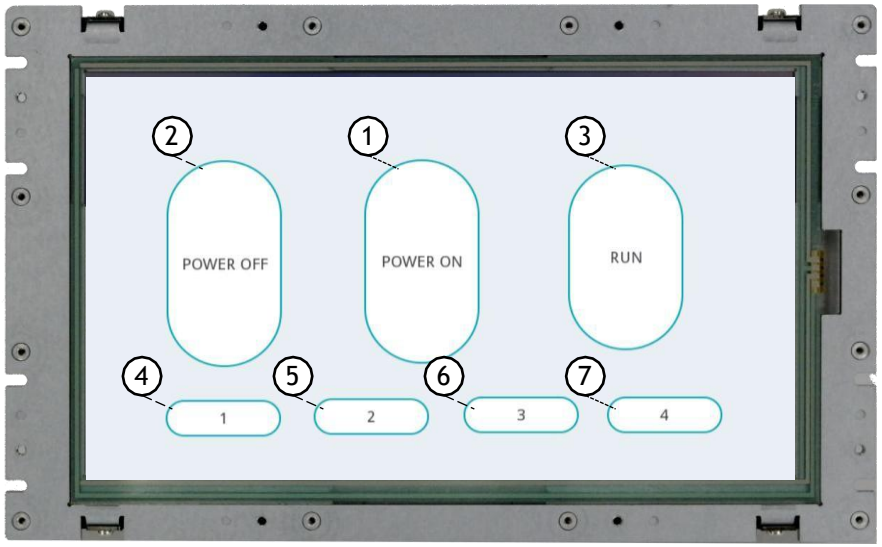



86Duinoアップロード完了

# 例：運動制御テスト – 5

## 結果

プログラムをQEC-M-090Tに正常にアップロードすると、QEC-M-090TにHMIが表示され、確認できます。




kinematics_test の例を表示するQEC-M-090T

## 各ボタンの機能

① **POWER ON (p1b1)**
```
power_on();
```

② **POWER OFF (p1b2)**
```
power_off();
```

③ **RUN (p1b3)**
```
run_demo3(5.02);
```

④ **1 (p1b4)**
```
Set_Des_Position(0.0, 24.0, 26.8, 0.0, -45.0, 0.0);
prepare_move();
```

⑤ **2 (p1b5)**
```
Set_Des_Position(0.0, 24.0, 26.8, 0.0, -20.0, 0.0);
prepare_move();
```

⑥ **3 (p1b6)**
```
Set_Des_Position(0.0, 24.0, 26.8, 0.0, 20.0, 0.0);
prepare_move();
```

⑦ **4 (p1b7)**
```
Set_Des_Position(0.0, 24.0, 26.8, 0.0, 45.0, 0.0);
prepare_move();
```

*次のページ：各機能の機能と動作の実行方法について説明します。

# 利用の流れとAPI概要 - 1

## 利用の流れ

- **セットアップ** → `begin()` ▸ `attach()` ▸ `setDc()` ▸ `setCiA402Mode()` ▸ `attachRTCyclicCallback()` ▸ `start()`
- **電源投入** → `enable()` ▸ `poweron_init()` ▸ `home()`
- **移動** → `Set_Des_Position()` ▸ `prepare_move()/ prepare_arc_move()`(コールバック経由でストリームをターゲットとする)
- **ループ** → HMIイベント

## API概要

### OVR6048K1_V.h

| API | どこで | 目的 | 典型的な使い方 |
|---|---|---|---|
| OVR6048K1_V_begin() | セットアップ | 内部バッファ/パラメータの初期化 | EtherCAT の起動前に 1 回呼び出します |
| OVR6048K1_V_loop() | ループ | 内部管理/状態 | ループの繰り返しごとに呼び出します |
| OVR6048K1_V_callback(long* pos) | 周期コールバック | 次のターゲットの作成(相対/絶対) | true が返された場合は、ターゲットを PDO に書き込みます |
| Set_Des_Position(x,y,z,rx,ry,rz) | 動作前のループ/HMI | 目標姿勢を設定す(線形や円弧軌道を実行する前に) | その後、prepare_move() または arc API を呼び出します |
| prepare_move() | ループ/HMI | 目標姿勢への直線移動の実行 | 周期コールバックを介して実行します |
| prepare_arc_move() | ループ/HMI | 円弧移動の実行(平面/中心/方向を使用) | 最初に平面/中心/方向を設定します |
| set_arc_plane(XY/XZ/YZ) | ループ/HMI | 円弧平面の選択 | 円弧の前提条件 |
| set_arc_dir(ARC_CW/ARC_CCW) | ループ/HMI | 円弧方向 | デフォルトで問題ない場合は省略可能です |
| Set_center_offset(cx,cy,cz) | ループ/HMI | 円弧中心(相対) | prepare_arc_move() の前に使用します |
| poweron_init() | 電源投入 | 有効後に内部パラメータの初期化 | 有効時に 1 回呼び出します |
| emergency_stop() | エラー/安全性 | 即時停止:再開するには再初期化が必要 | 停止時に使用します |

# 利用の流れとAPI概要 － 2

## API概要

### Ethercat.h

| API | どこで | 目的 | 典型的な使い方 |
|---|---|---|---|
| master.begin() | **セットアップ** | Subデバイスをスキャンし、Ｐｒｅ－Ｏｐに入る | 最初のEtherCAT呼び出し |
| motor[i].attach(i, master) | **セットアップ** | SubデバイスiをMデバイスに関連付ける | 各軸について |
| motor[i].setDc(cycle_ns) | **セットアップ** | DC を有効にし、周期を設定する | 例：250,000 ns (250 µs) または 1,000,000 ns |
| master.attachRTCyclicコールバック(cb) | **セットアップ** | リアルタイム周期コールバックを登録する | PDOループ |
| master.start(cycle_ns, ECAT_SYNC) | **セットアップ** | SM/FMMU/DC を設定し、OP に入る | 周期OPを開始 |
| pdoWrite()/pdoRead() | **コールバック** | プロセスデータの書き込み/読み取り | 通常はモーターAPIで抽象化されている |

### Ethercat.h > CiA402

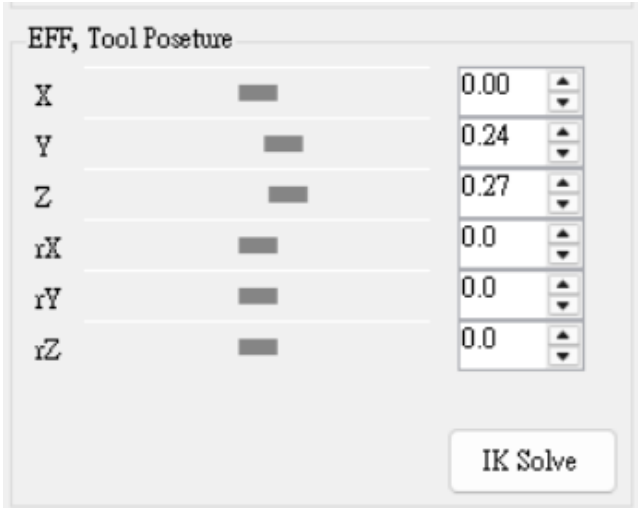
| API | どこで | 目的 | 典型的な使い方 |
|---|---|---|---|
| motor[i].setCiA402Mode(CIA402_CSP_MODE) | **セットアップ/ループ** | CSPモードを選択 | **始動前** |
| motor[i].enable() | **電源投入/ループ** | オペレーション有効モードに移行する | **OPおよび準備完了後** |
| motor[i].getCiA402State() | **コールバック/ループ** | ドライブ状態を照会 | **モーション実行前にチェックするロジック** |
| motor[i].setTargetPosition(pulse) | **コールバック/ループ** | ターゲット（PDO）に書き込み | OVRの**コールバック**/ゲート条件から |
| motor[i].getPositionActualValue() | **コールバック/ループ** | 実際の位置を読み取る | 原点復帰／モニタリング |

# APIで直接ビューア値を使用する

ユーザーは、**軌道表示ツール**から同じ姿勢の値を取得し、コード（または**ArduBlock**）に適用してアームを駆動できます。ビューアでは、スライダをドラッグするか、数値を入力して姿勢をシミュレートし、その値をAPIにコピーします。

**ETF**下のフィールド、**Tool Posture** – **(X, Y, Z, rX, rY, rZ)** –は、**1:1**で、次のようにマップされます：

```
Set_Des_Position(0.0, 24.0, 26.8, 0.0, -45.0, 0.0);
prepare_move();
```




軌道表示ツール ：EFF, Tool Poseture

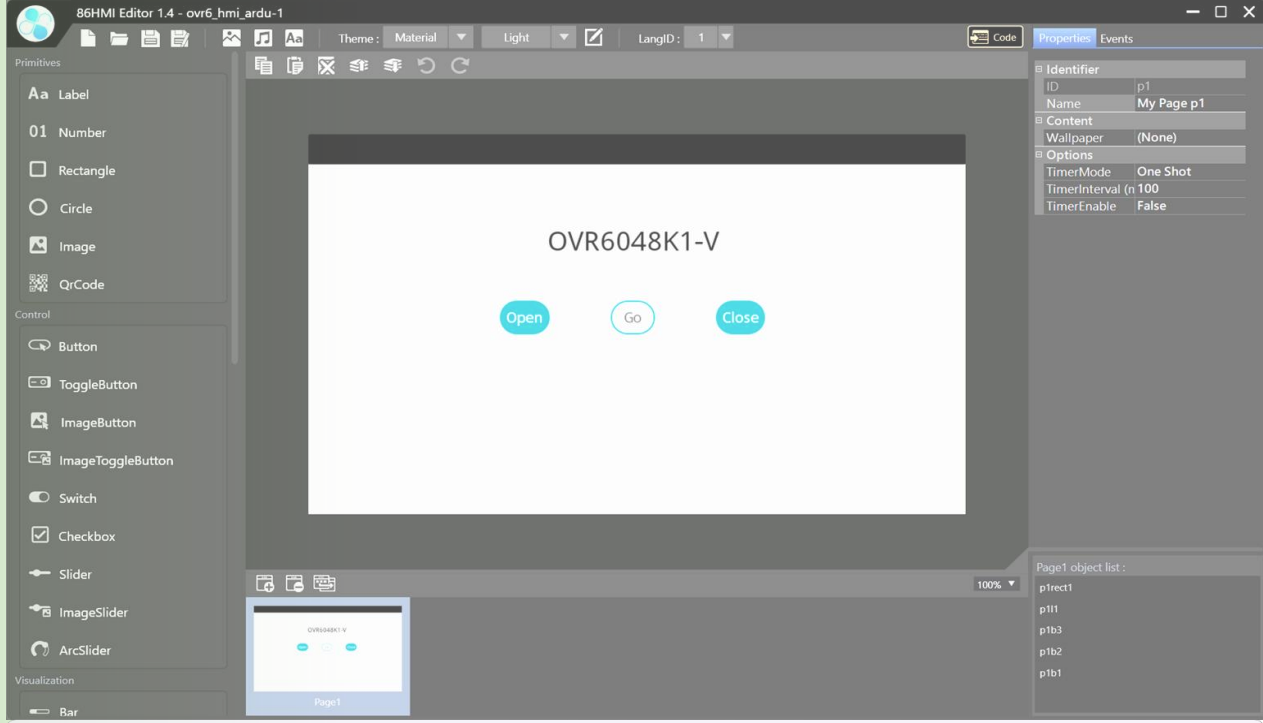
## ステップ：

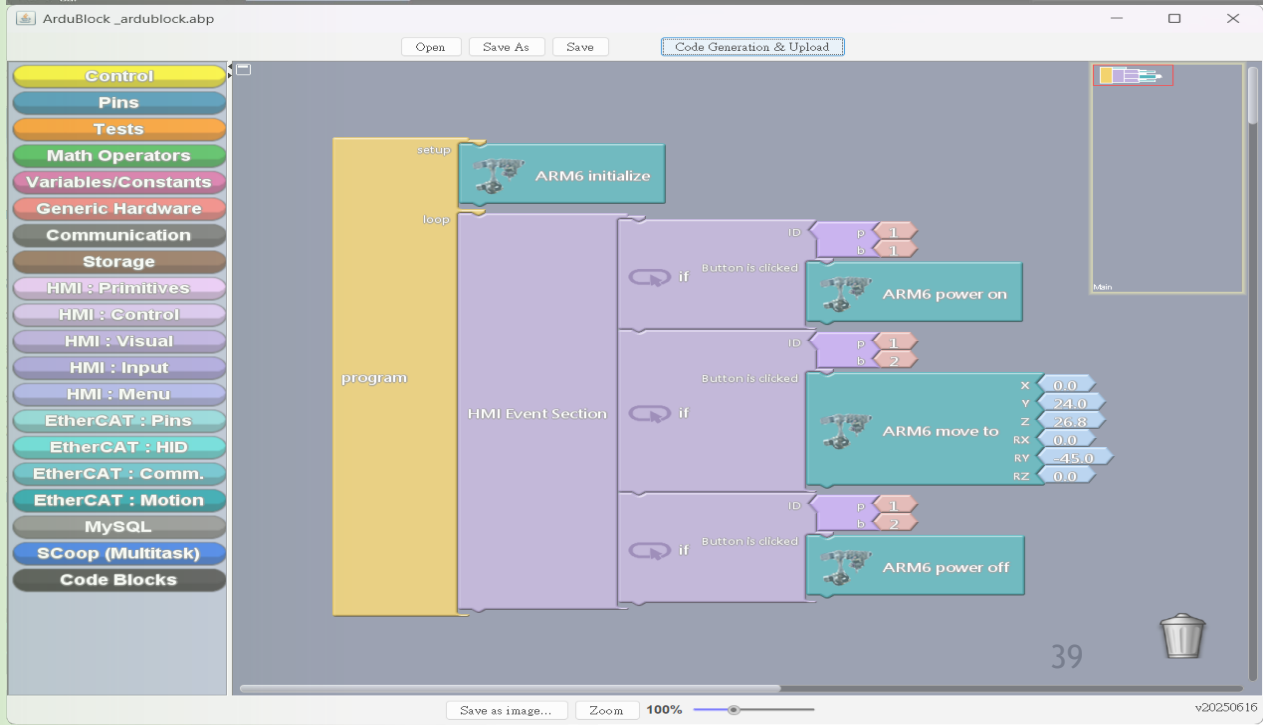
1. EFFにて，`Tool Posture, X, Y, Z, rX, rY, rZ`を設定/確認。
2. それらを1:1で、`Set_Des_Position(x, y, z, rX, rY, rZ);`にコピー。
3. `prepare_move();`を呼び出し、実行。

その後、軌道表示ツールと同じ位置に腕を制御できます。

# 4. はじめましょう - 1

ローコードの例：86HMI + ArduBlock

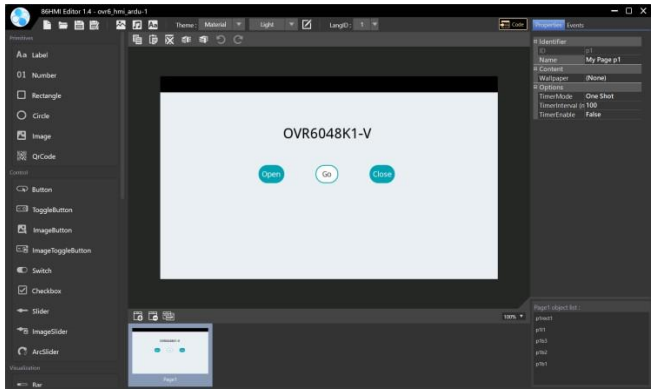


# 86HMI + ArduBlock – 概要 1

## 86HMI – UIデザイナ(HMI)

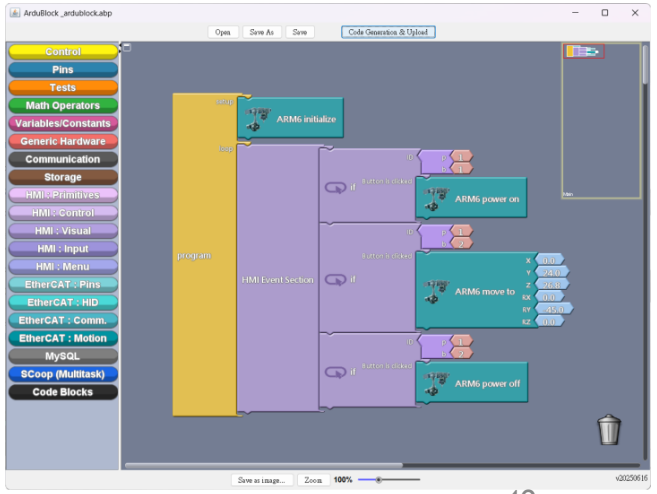
- **概要:** LVGL86ライブラリをベースにしたドラッグ・アンド・ドロップ式のスクリーンエディタです。
- **作成するもの:** ボタン、ラベルなど、固有のIDにバインドされたオブジェクト（例：p1b1、p1b2）。
- **生成するもの:** スケッチに貼り付けるC/C++ UIスタブ。イベントは**自動生成されるHMIイベントセクション**で処理されます。
- **重要性:** オペレータがパネルから直接モーション（電源のオン/オフ、プリセットの移動、デモの実行）をトリガーできます。



86HMI エディタ

## ArduBlock – モーションロジック (ブロック)

- **概要:** 86Duino IDEをベースにしたブロックベースのエディタです。
- **作成するもの:** **HMI**、**EtherCAT**、**モーション**用のブロックを使用した高レベルロジックです。
- **生成するもの:** 指定されたAPI(例: `power_on()`、 `Set_Des_Position()`、`prepare_move()`)を呼び出す86Duino コードです。



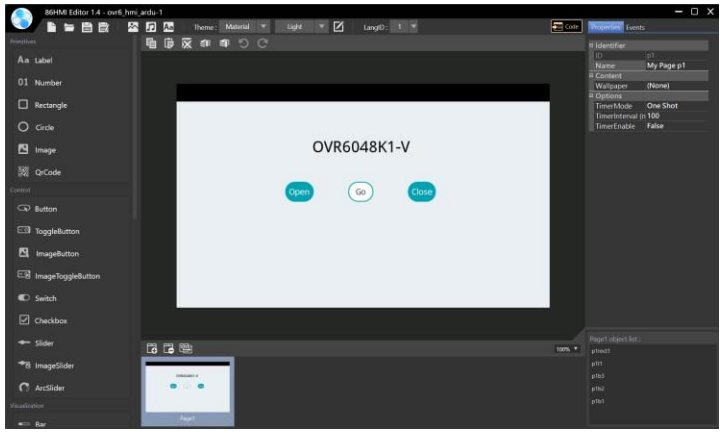
ArduBlock

*注: ARM6クラスブロックはOVR6048K1-Vプロジェクト専用です

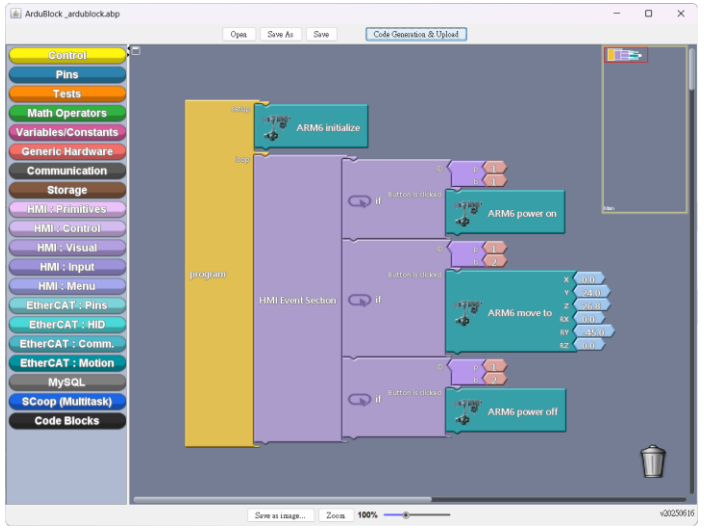
# 86HMI + ArduBlock – 概要 2

## なぜ両方？

- **86HMI**は、操作インタフェースを設計します。
- **ArduBlock**は、これらのUIイベントをロボットの動作に関連付けます。
- **これらの組み合わせは**、ボタン→API呼び出し→EtherCATモーションというローコードパスを実現します。




86HMIエディタ




ArduBlock

OVR6048K1

# 例：ローコードによる運動制御 - 1

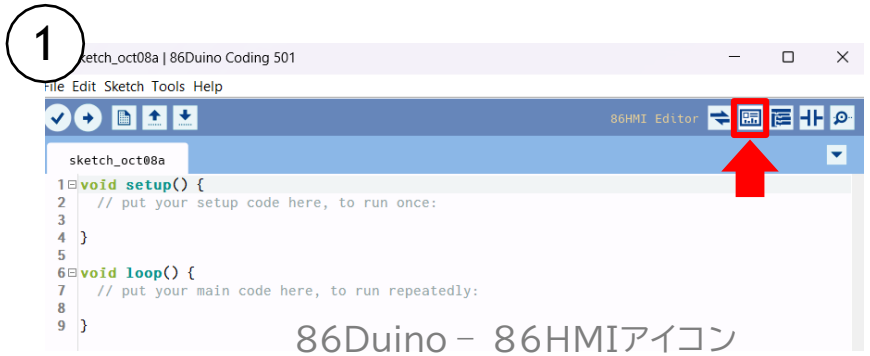
## ステップ1 - 新しいプロジェクトを開いてHMIを設計する（86HMI）

環境設定については、例：運動制御テストを参照してください。

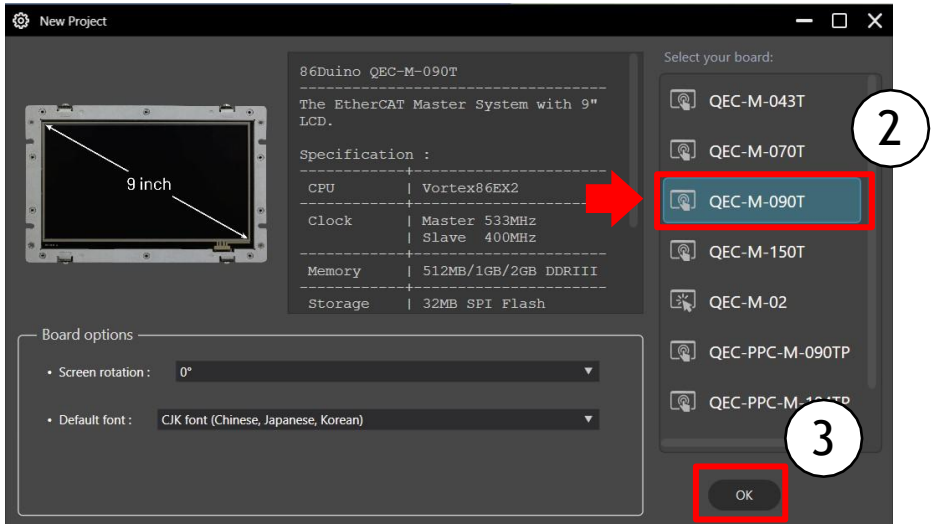
① 86HMIエディタを開きます。

② **QEC-M-090T**を選択します。

③ 「**OK**」ボタンをクリックすると、エディタパネルが表示されます。




86Duino – 86HMIアイコン




86HMI エディタ

# 例：ローコードによる運動制御 - 2

## ステップ2 - 3つのボタンを作成

最初のページに、「 **Open** 」、「 **Go** 」、「 **Close** 」の3つのボタンウィジェットを作成します。

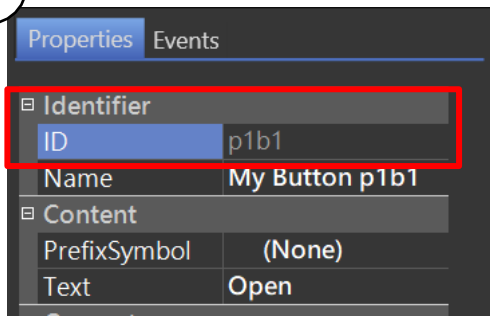
① 左側のメニューで「 **Button** 」をダブルクリック、また描画エリアまでドラッグします。

② 3つのボタンを複製します。

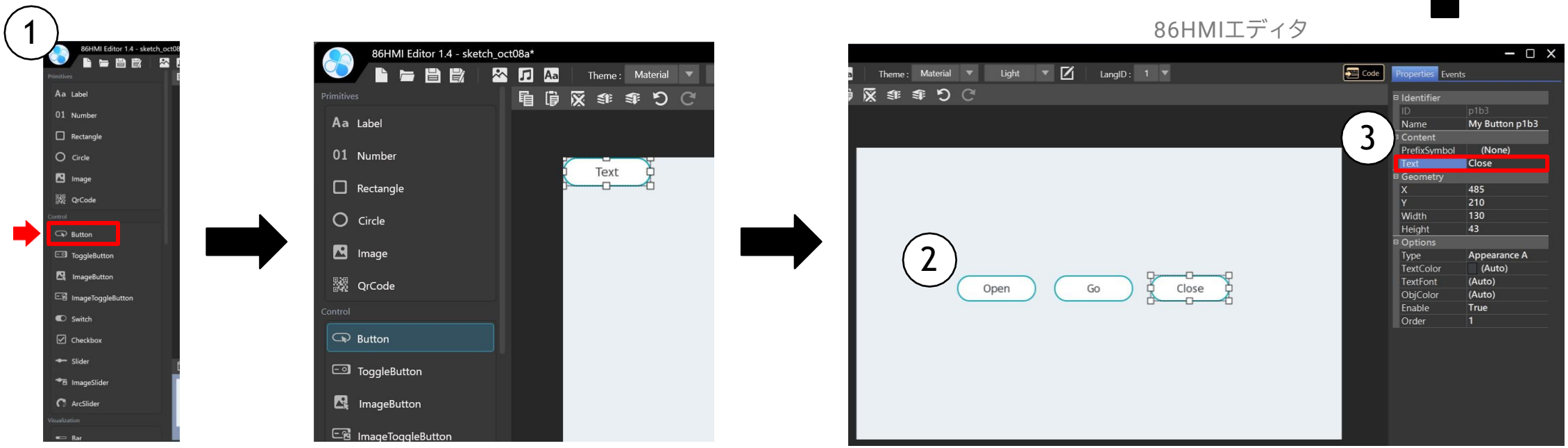
③ 右側の**Properties**メニューでテキストを「 **Open** 」、「 **Go** 」、「 **Close** 」に変更します。

④ 各オブジェクトのIDを確認します。(例：p1b1、p1b2、p1b3)

4 86HMI プロパティ



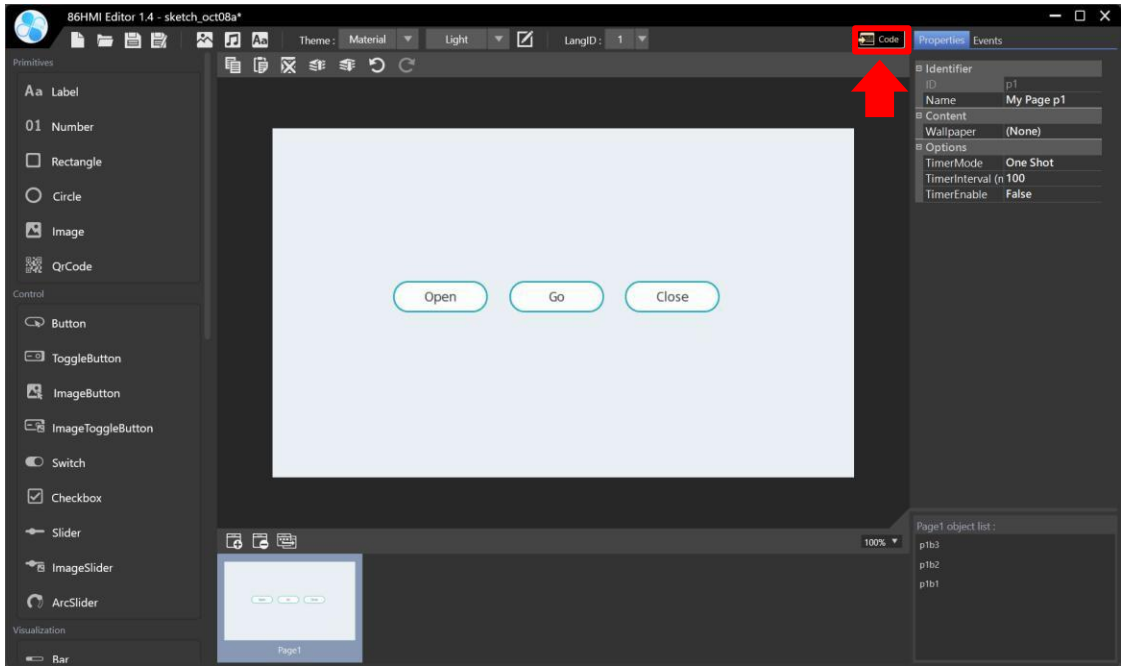

86HMIエディタ



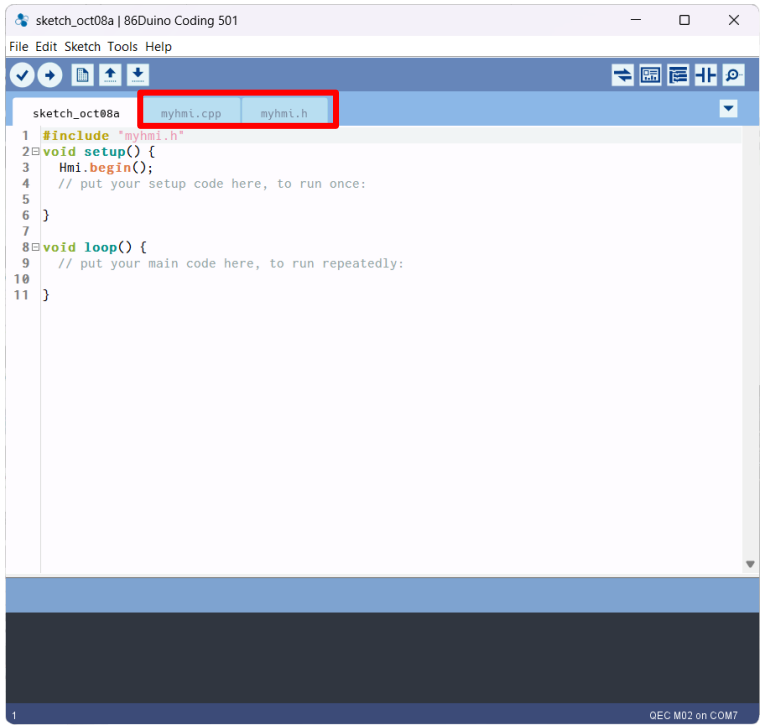
# 例：ローコードによる運動制御 - 3

## ステップ3 – 86HMIコード生成

「**Code**」(右上)をクリックしてUIコードを生成し、スケッチにコピーします。




86HMIエディタ



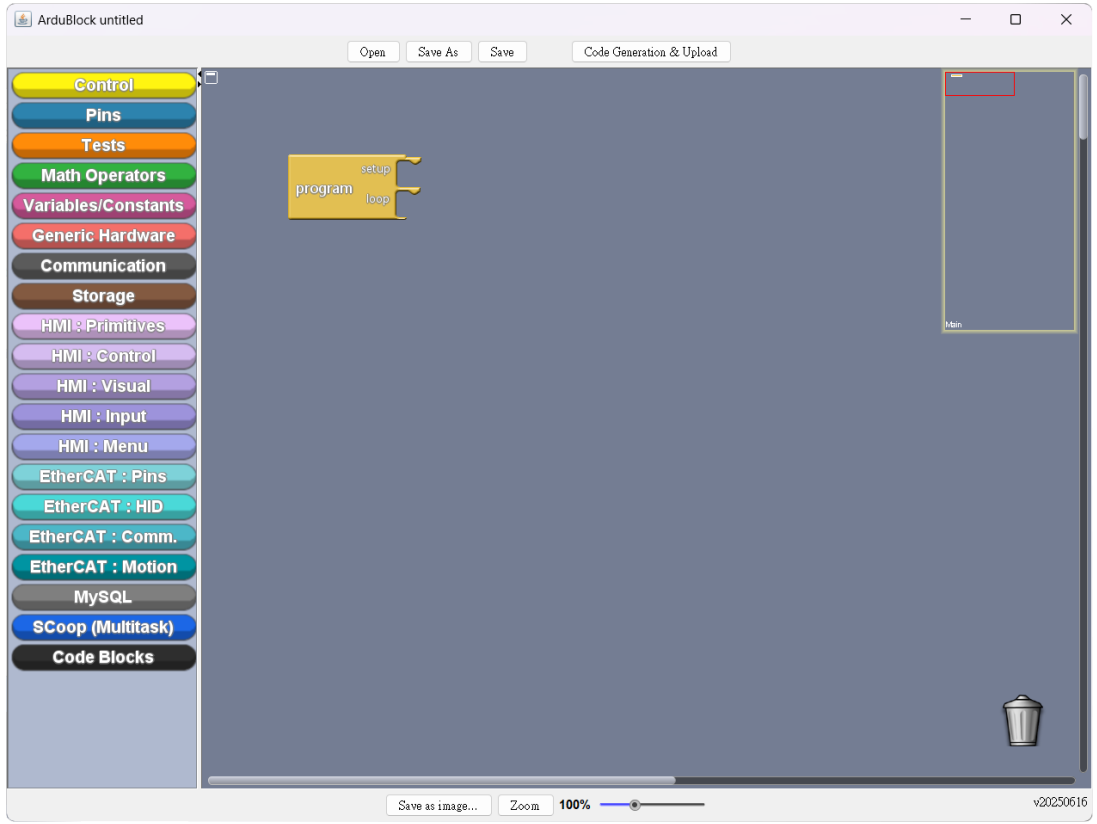

86Duino – 86HMIファイル

# 例：ローコードによる運動制御 - 4

## ステップ4 – ArduBlockを開く

右上のアイコンからArduBlockを開きます。


86Duino – ArduBlockアイコン



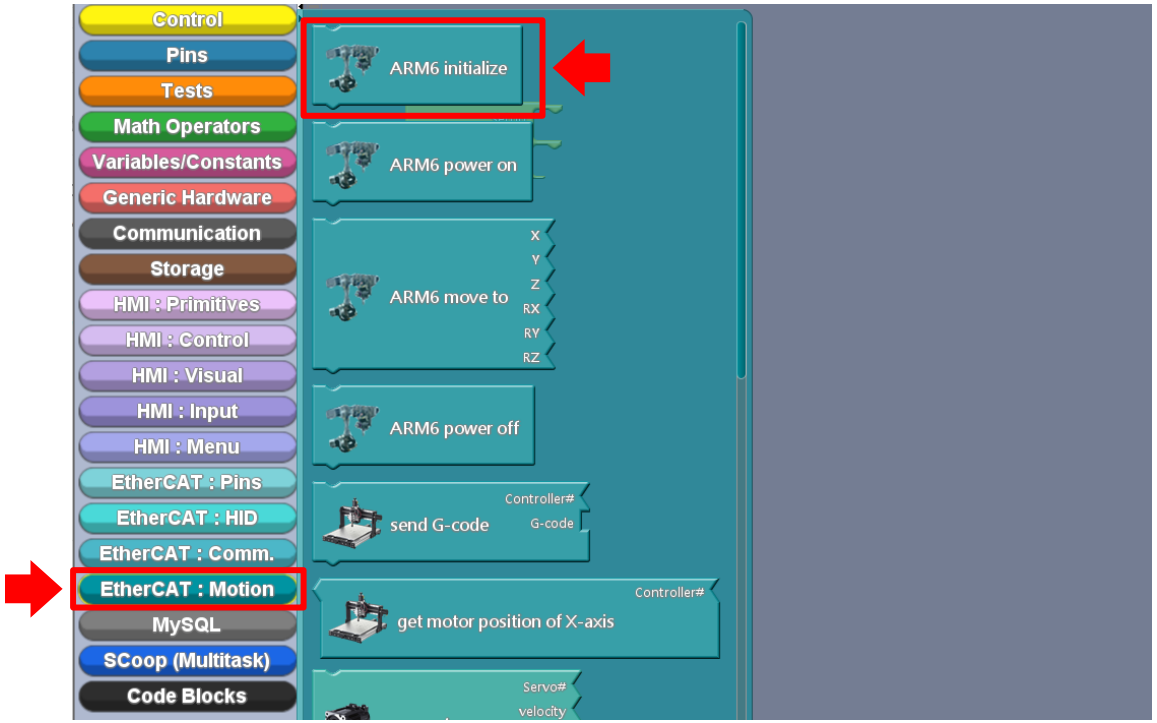

ArduBlock

# 例: ローコードによる運動制御 - 5

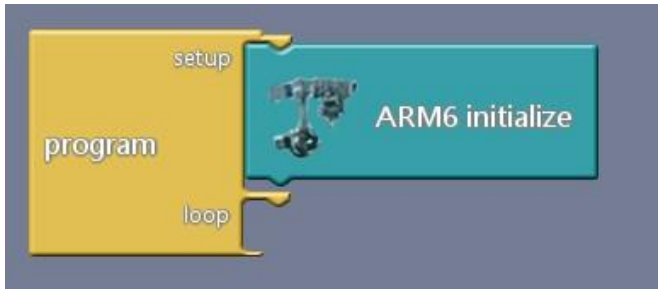
## ステップ5 – セットアップにARM6初期化ブロックを配置する

まず、**オリエンタルモーター社のドライバパラメータを初期化**する必要があります。これらのパラメータは、既に「**ARM6初期化**」というブロックにパッケージ化されており、「**EtherCAT: Motion**」クラスに含まれています。

これをプログラムの「セットアップ」領域に配置します。




ArduBlock – EtherCAT : モーション



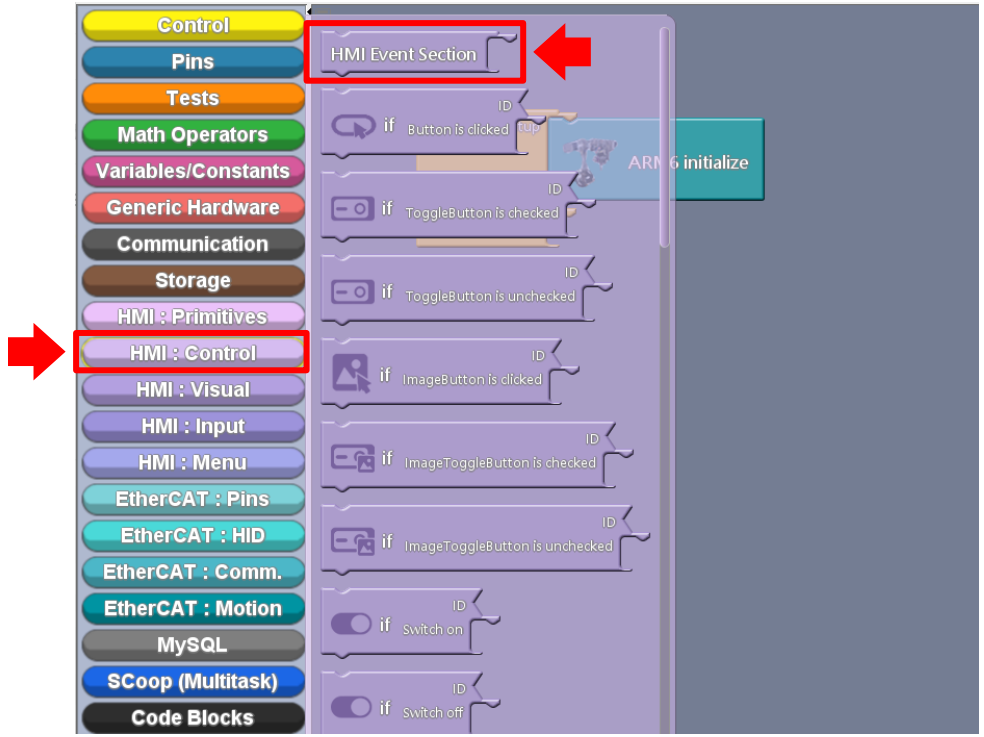

ArduBlock – ARM6初期化

# 例：ローコードによる運動制御 - 6

## ステップ6 – HMIイベントブロックをループに配置する

ロボットアームとHMIを連携させたいので、HMIイベントをループに配置する必要があります。

ボタンイベント(例： クリック、押下)は、「 **HMI : Control** 」クラスにある「 **HMI Event Section** 」というHMIイベントに関連付ける必要があります。




ArduBlock – HMI :Control

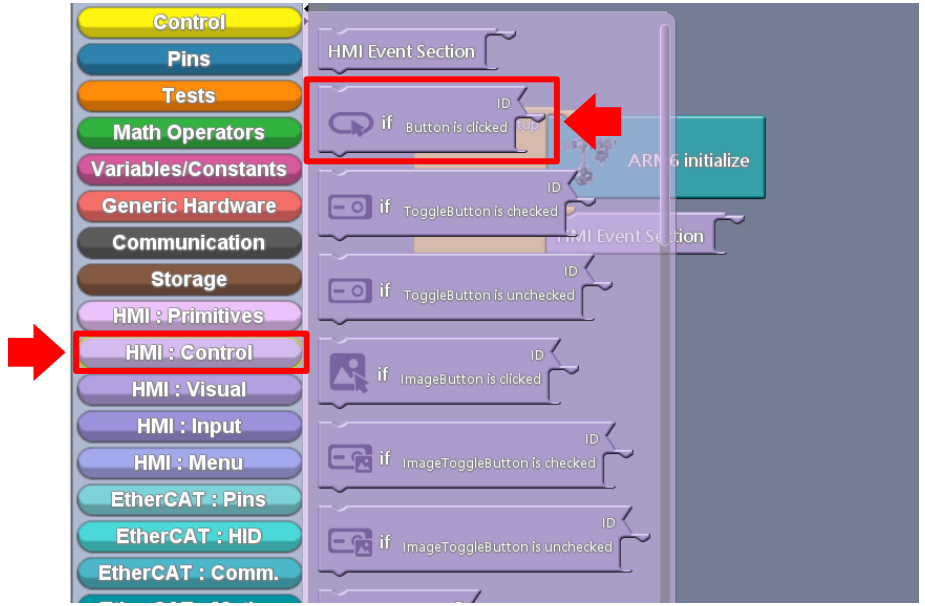

ArduBlock – HMI Event section

# 例：ローコードによる運動制御 - 7

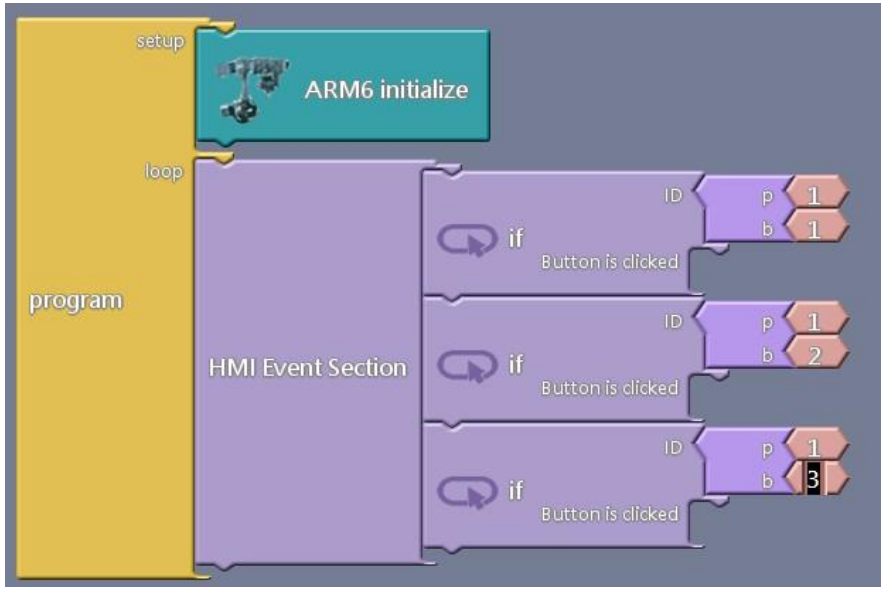
## ステップ7 – HMIイベントにボタンイベントを追加する

「 **HMI : Control** 」クラスの「 **HMI Event Section** 」に、「 **if Button is clicked** 」ブロックというボタンイベントを追加します。特定のボタンがクリックされた場合、HMIイベントは「**true**」を返します。

ボタンが3つ必要なので、クリックされたボタンのブロックを3つ複製し、**IDをp1b1、p1b2、p1b3 に変更**します。




ArduBlock – HMI :Control




ArduBlock – if Button is clicked

# 例：ローコードによる運動制御 - 8

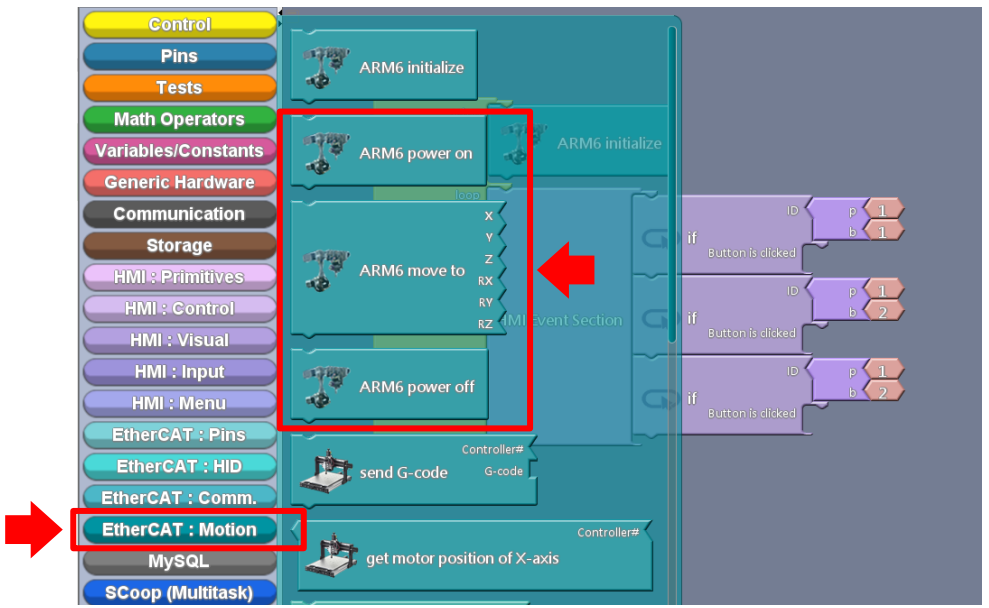
## ステップ8 – ボタンイベントにARM6の電源オン/移動/電源オフを設定する

OVR6048K1の機能は、「**EtherCAT: Motion**」クラス内の「**ARM6 power on** 」、「**ARM6 move to** 」、「**ARM6 power off** 」などのブロックに既にパッケージ化されています。

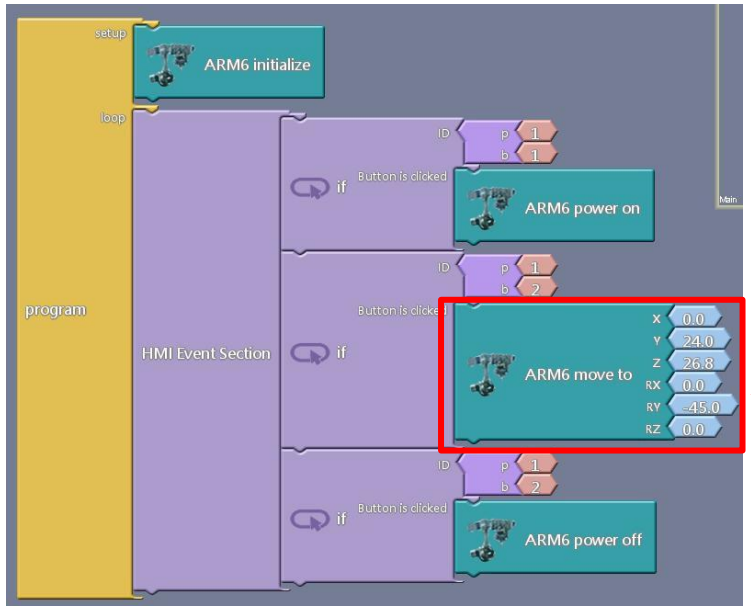
これらのOVR6048K1機能ブロックを、「**if Button is clicked** 」ブロックに以下のように設定します：

- p1b1を「**ARM6 power on** 」に、p1b2を「**ARM6 move to** 」に、p1b3を「**ARM6 power off** 」にします。

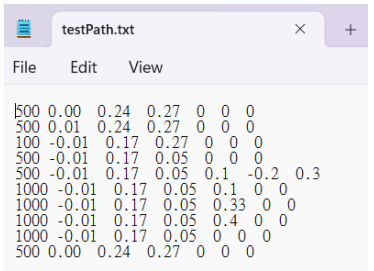
*注: 「 **ARM6 move to** 」ブロックのパラメータには、**軌道表示ツールの位置値**を使用してください。




ArduBlock – EtherCAT :Motion




ArduBlock – ARM6 move to



# 例：ローコードによる運動制御 - 9

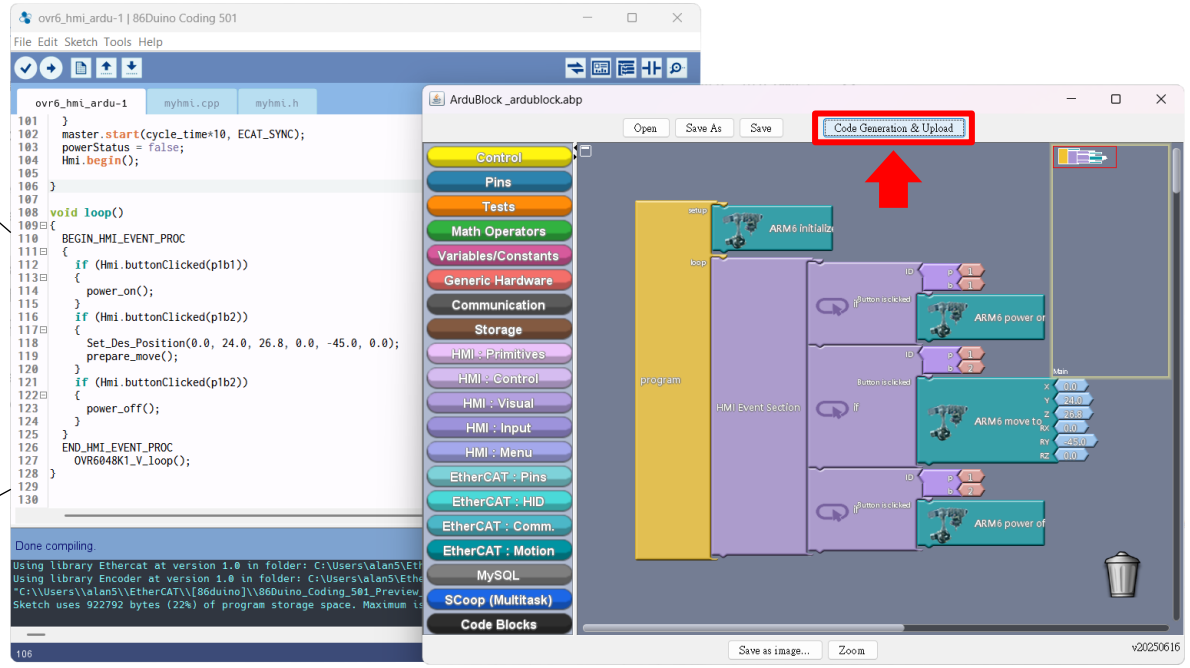
## ステップ9 – コード生成とアップロード

完了後、「 **Code Generate & Upload** 」ボタンをクリックしてコードを生成し、QEC-M-090Tにアップロードできます。

このブロックプログラミングでは、EtherCAT通信を確立し、AZ-miniドライバを設定し、HMIボタンを使用してOVR6048K1ロボットアームを設定、制御します。

```
107
108 void loop()
109 {
110   BEGIN_HMI_EVENT_PROC
111   {
112     if (Hmi.buttonClicked(p1b1))
113     {
114       power_on();
115     }
116     if (Hmi.buttonClicked(p1b2))
117     {
118       Set_Des_Position(0.0, 24.0, 26.8, 0.0, -45.0, 0.0);
119       prepare_move();
120     }
121     if (Hmi.buttonClicked(p1b2))
122     {
123       power_off();
124     }
125   }
126   END_HMI_EVENT_PROC
127     OVR6048K1_V_loop();
128 }
```

ArduBlock – 86Duino – HMIコード

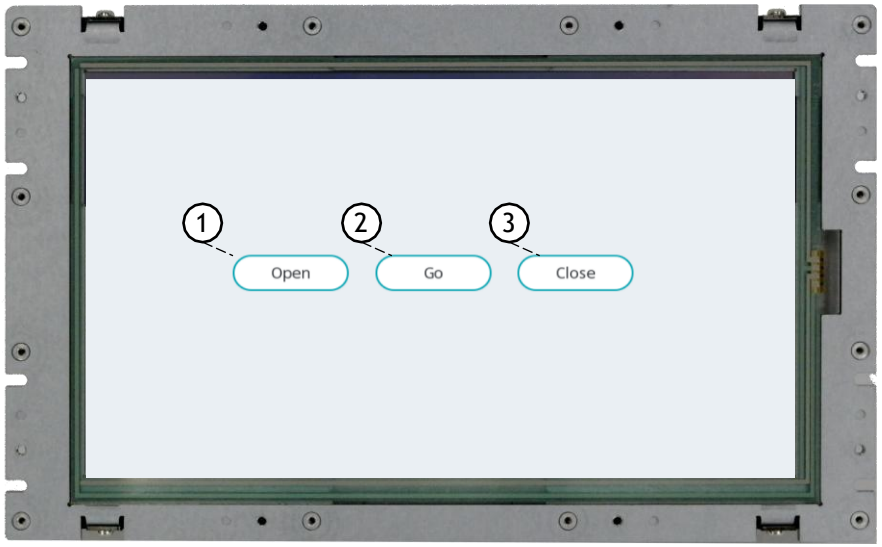



ArduBlock – 86Duino

# 例：ローコードによる運動制御 - 10

## 結果

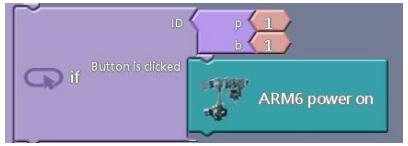
プログラムをQEC-M-090Tに正常にアップロードすると、QEC-M-090Tに表示されるHMIを確認できます。



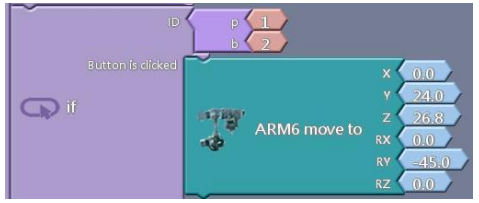

QEC-M-090T とローコードの例

## 各ボタンの機能

① **Open (p1b1)**



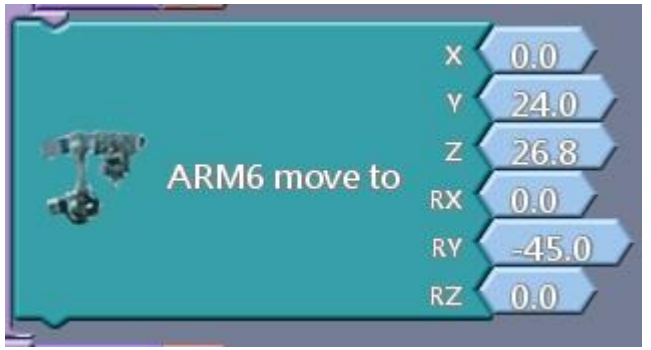

② **Go (p1b2)**




③ **Close (p1b3)**

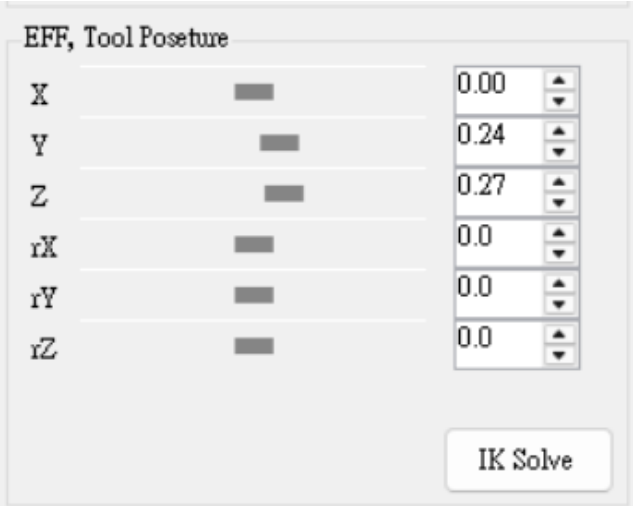

# ArduBlockでビューアの値を直接使用する

ユーザーは、**軌道表示ツールと**同じ姿勢を取得し、コード（またはArduBlock）に適用してアームを駆動できます。ビューアでは、スライダをドラッグするか、数値を入力して姿勢をシミュレートし、その値をAPIにコピーします。



ArduBlock　ロボットアーム移動ブロック

EFF の下のフィールド 、**Tool Posture** – **(X, Y, Z, rX, rY, rZ)** –は、1:1 で、次のようにマッピングされます：



軌道表示ツール:EFF、Tool Poseture

## ステップ：

1. EFF、Tool Postureで、X、Y、Z、rX、rY、rZ を設定/確認します。
2. それ多をARM6 move to(X, Y, Z, RX, RY, RZ)ブロックに1:1でコピーします。

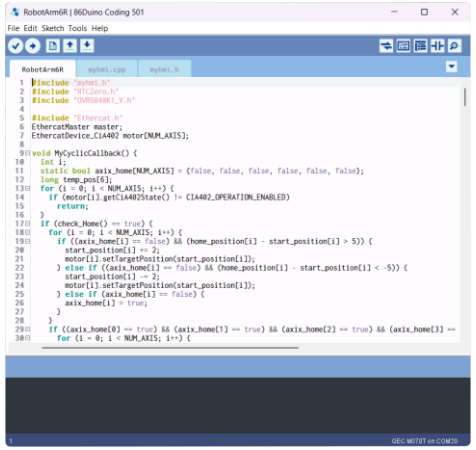
その結果、軌道表示ツールと同じ位置にアームを制御できます。

# 5. はじめましょう - 2

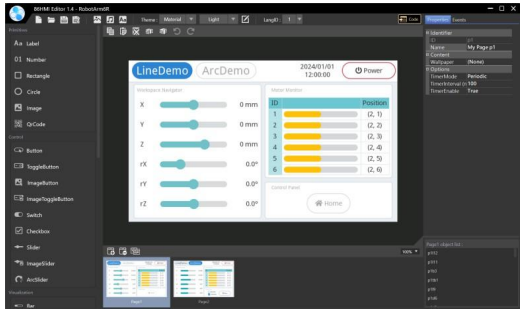
**RobotArm6R**の例

# 例：RobotArm6R– 概要

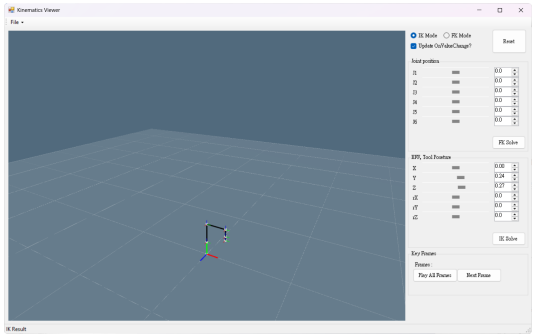
**QEC-M-090T** と **AZD-KRED × OVR6048K1-V** を組み合わせで、**ECAT_SYNC＋DC**を用いることで、**球面上**でのエンドエフェクタの滑らかな軌道を、計画からシミュレーション、ワンタッチHMI操作まで一貫して実行します。




RobotArm6R 86Duinoの例




RobotArm6R HMIの例



RobotArm6R 起動表示ツール

RobotArm6R基準姿勢

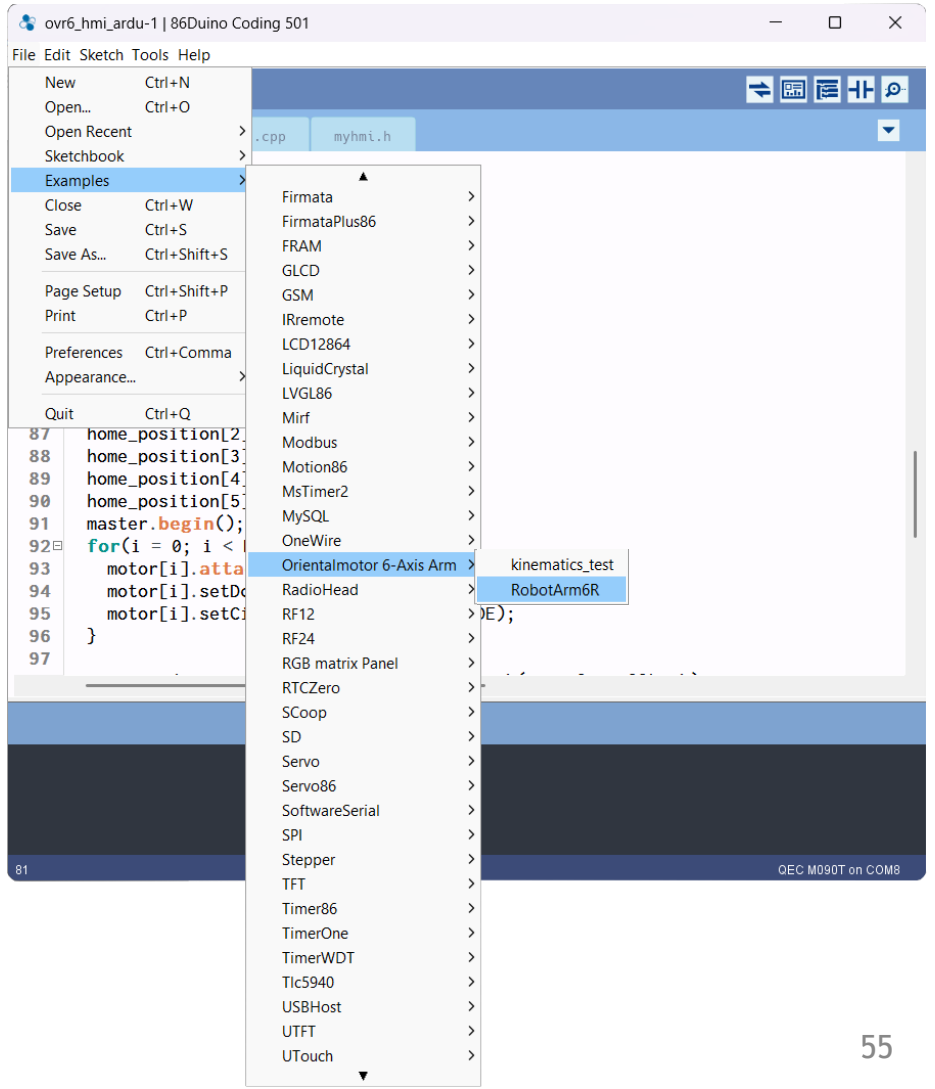

RobotArm6Rの例

# 例：RobotArm6R - 1

## ステップ1：例を開く

- 「 Tools ▸ Board 」で「**QEC-M090T**」を選択し、適切なポートを選択すると、サンプルセットが表示されます。
- “File ▸ Examples ▸ Orientalmotor 6-Axis Arm ▸RobotArm6R”を開きます。

86Duinoの例



# 例: RobotArm6R - 2

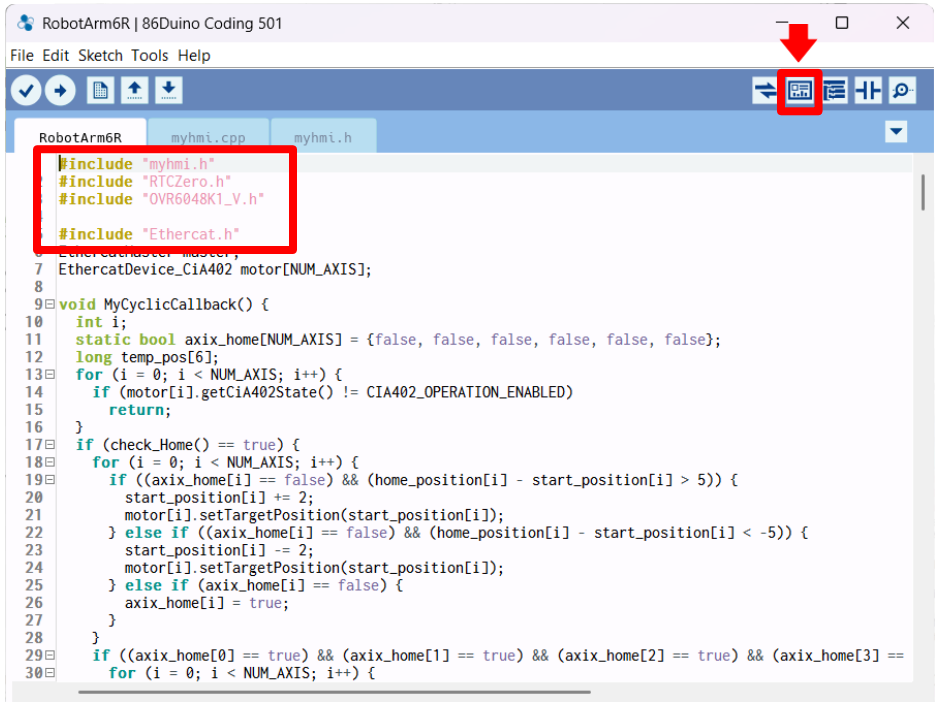
## ステップ2 - コードを表示してHMIを開く

- RobotArm6Rを開くとスケッチが読み込まれます(右)。3つのライブラリを使用します:
  - **myhmi.h** — HMIヘルパ
  - **OVR6048K1_V.h** — ロボット・モーション・ヘルパ
  - **Ethercat.h** — EtherCATプロトコル
  - **RTCZero.h** — RTCタイマライブラリ
- この例では、以下の設定を使用します:
  - EtherCATサイクルタイム:**250 マイクロ秒 (μs)**。
  - EtherCATモード: **ECAT_SYNC**。
  - 分散クロック(DC): **Enable (250 μs)**。
  - CiA-402モード: **CSP (周期同期位置)**。

```
 91  master.begin();
 92  for(i = 0; i < NUM_AXIS; i++) {
 93    motor[i].attach(i, master);
 94    motor[i].setDc(cycle_time*10);
 95    motor[i].setCiA402Mode(CIA402_CSP_MODE);
 96  }
 97
 98  error_code = master.attachRTCyclicCallback(MyCyclicCallback);
 99  if (error_code != 0) {
100   printf("error : %d\n", error_code);
101  }
102  master.start(cycle_time*10, ECAT_SYNC); // cycle_time = 25000; // nanosecond
```

EtherCATのRobotArm6R.ino コード

- 86HMIアイコン(右上)をクリックしてHMIエディタを起動します:



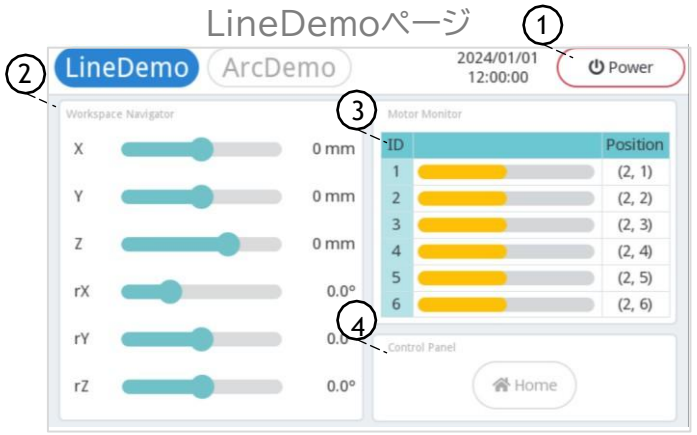

RobotArm6R.ino

# 例: RobotArm6R- 3

## ステップ3 – 86HMIレイアウト

- HMIは、**LineDemo** と **ArcDemo**の2ページがあります。

LineDemoページ




ArcDemo ページ


## 各HMIオブジェクトの機能

① **Power(電源)**
すべての軸ドライバを有効または無効にします (CiA402 の有効/無効に切り替えます)。

② **作業空間ナビゲータ**
ロボットアームの作業空間の値を、x、y、z(mm)、rx、ry、rz(°)で表示します。この領域は LineDemo ページ内で操作可能であり、ユーザーは作業空間内でアームを移動できます。ただし、事前にホーミングを行う必要があり、デモ実行中は同時に操作することはできません。

③ **モーターモニタ**
ロボットアームの位置(モータードライバから読み取った値)を表示します。また、現在の位置を示すバーも表示します。

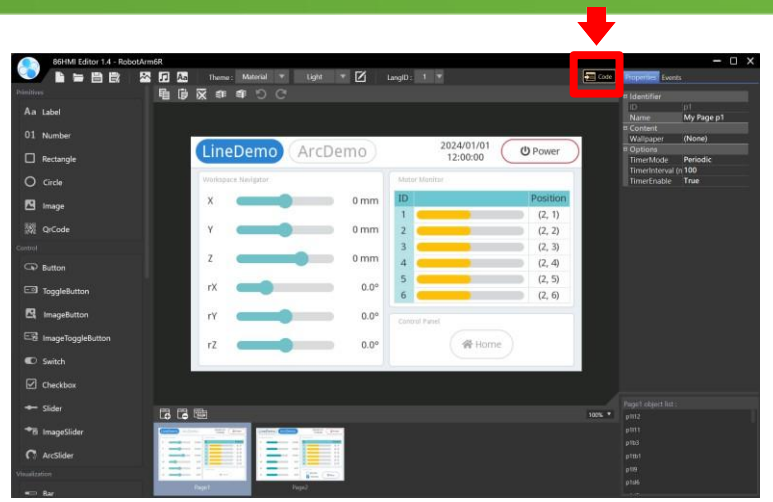
④ **コントロールパネル (Home)**
LineDemo では、この領域にホーム ボタンがあり、アーム (すべての軸) を原点位置に移動します。

⑤ **コントロールパネル (Run)**
ArcDemo では、Run ボタンと2つのラジオオプション(60mm と 10mm)がある領域です。移動オプションを選択した後に、Run ボタンをクリックすると、ロボットアームが選択した寸法の球面上を動き始めます。Run ボタンを再度クリックしない限り、動作は継続されます。正しい寸法の球体をクリックする必要があることに注意してください。これはホーミング後に行ってください。

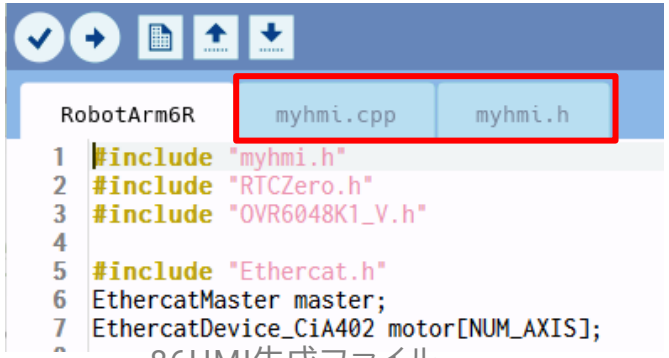
# 例: RobotArm6R- 4

## ステップ4 – 86HMIコード生成

- 「**Code**」ボタン (右上) をクリックすると、イベントスタブとincludeステートメントが生成され、メインスケッチに貼り付けられます。
- 生成後、**myhmi.h** と **myhmi.cpp** が表示されます。これらはLVGL86 ライブラリのヘルパと実行プログラムを表します。



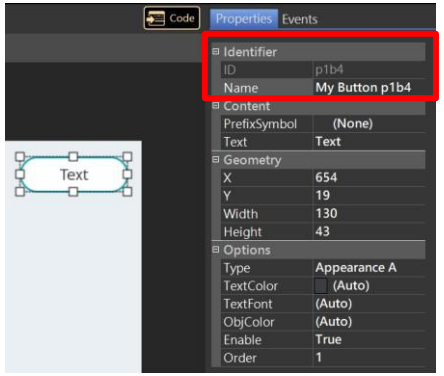

RobotArm6R HMIの例

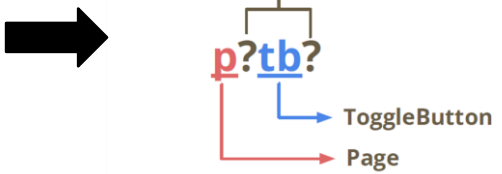



86HMI生成ファイル

* 追加情報: 86HMIでは、すべてのオブジェクトに一意の IDがあり、命名規則は右の図に示すように、クラスシリアル番号を含むページ番号です。

86HMI ID命名規則

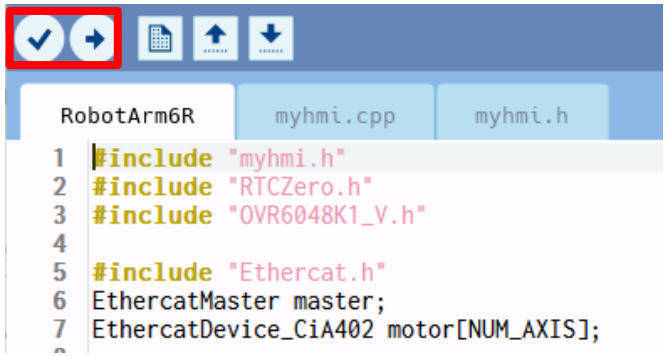


# 例: RobotArm6R - 5

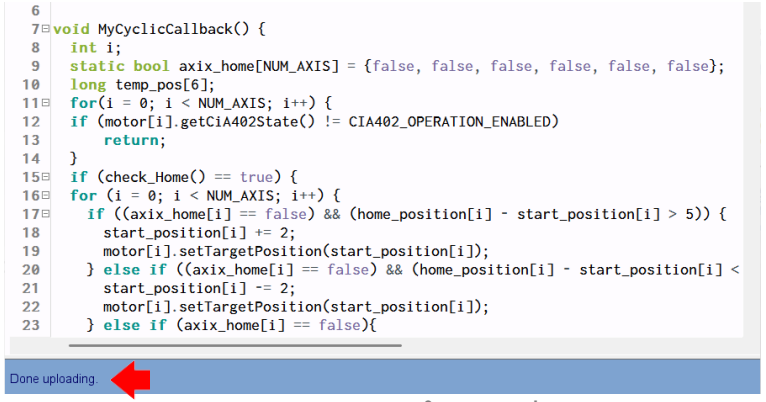
## ステップ5 - ビルドとアップロード

コードを記述したら、ツールバーの  をクリックしてコンパイルします。コンパイルが完了し、エラーがないことを確認したら、  をクリックしてアップロードします。




86Duino アップロード及び検証アイコン

アップロードが完了すると、下のウィンドウに「**Done uploading**」と表示されます。

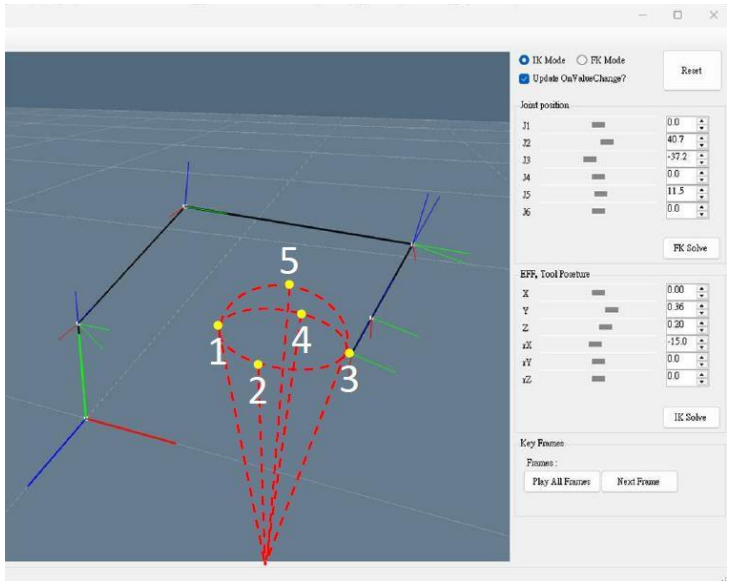



86Duinoアップロード完了

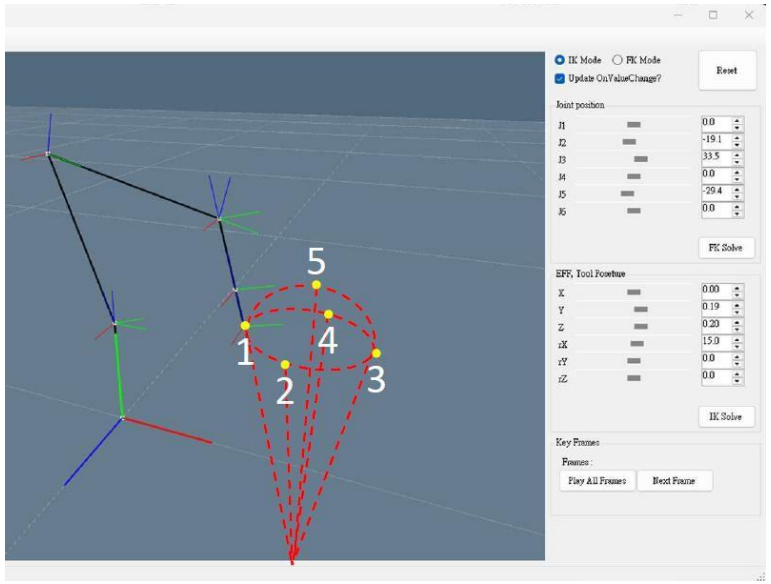
# 例: RobotArm6R の説明

RobotArm6Rの例では、指定した半径の球体の周りを動きます。デモには、半径60mmと100mmの球体の2種類があります

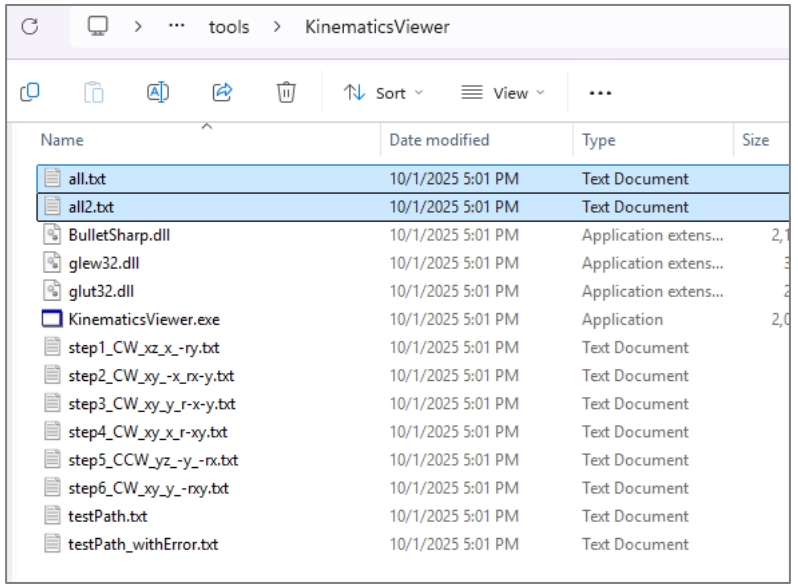
ユーザーは、起動表示ツールでパスの値を確認できます。例は **all.txt**および **all2.txt**と呼ばれます。




軌道表示ツールでのrobotArm6R の移動 -1




軌道表示ツールでのrobotArm6Rの移動 -2




robotArm6R 球面移動デモのパス

# 例: RobotArm6R – ビデオ

## 結果


（ビデオ： https://youtu.be/9-X3WKAfU1o?si=HUMSyUU-MeYdyDMh ）

# Thank you!!

さらに詳しい情報をご希望の場合は、お気軽にお問い合わせください。

**QEC Website**

**ICOP Website**

**Orientalmotor Website**