KXR-L4T-R ver.2
アカデミックパック

自律プログラム
マニュアル

第 1 版

近藤科学株式会社

2018 年 6 月 25 日　第 1 版

目次

はじめに

このたびは KXR-L4-T アカデミックセットをお買い求めいただき、誠にありがとうございます。本製品をご使用の前に、本マニュアルを熟読いただきますよう、お願いいたします。

本マニュアルでは PSD（位置検出素子）センサを使用し「目標に向かって近づきお辞儀をする」サンプルプログラムを説明します。なお、本マニュアルを使用するにあたりキット付属の「組立説明書」をご覧いただき「モーションの再生」まで完了してからご利用ください。

本マニュアルおよび付属品に掲載された一切の情報の流用による結果についての責任は負いかねます。

また内容は予告無く内容が変更される場合があります。ご理解の上ご使用なさいますよう、お願いいたします。

諸注意

- 本製品を濡らしたり、湿度が高い場所で使用しないでください。
- 本製品は基板上の各端子が露出しているため、ショートの危険性があります。ショートを起こしたり端子の誤接続をすると発熱/発火の恐れがありますのでご注意下さい。金属に接した状態での使用はおやめください。
- 本製品の日本国内以外での使用に関しましてはサポートいたしかねます。
- ケーブル類はプラグ部分をつかんで挿抜していただき、差し込むときは、向きを間違えないようにしてください。
- お客様によるハンダ付け不良等による不具合に関しましては保証対象外となります。
- 異常（発熱・破損・異臭など）を感じたらすぐにバッテリや電源を切るようにしてください。
- 問題が発生した場合は直ちに使用をやめ、弊社サービス部へご相談ください。

免責事項

- 本リファレンスおよび内容に関する一切の権利は近藤科学株式会社が有しますが、このリファレンスは参考資料として公開されるものです。このリファレンスを使用したときの障害や損害につきましては、近藤科学株式会社は一切保証いたしませんので、使用者の責任においてご利用ください。
- 本マニュアルに関する著作権、ロゴや一部のアイコンのデザイン、その他法的な諸権利の一切は近藤科学株式会社が有します。
- 本マニュアルには 2018 年 6 月現在における対応状況を記載しております。
- 本マニュアルの内容につきましては、予告なく変更する場合があります。
- 本マニュアルおよびをソフトウェアのあらゆる形態での不特定多数への再配布は禁止します。また著作権者に無断で販売、貸与およびリースなども出来ません。
- 近藤科学株式会社は、ソフトウェアのインストールや使用の結果によって生じたあらゆる損害に対して、一切の責任を負いません。

お問い合わせ

本製品ならびに付属品についてのお問い合わせは弊社サポート窓口までご連絡下さい。

〒116-0014　東京都荒川区東日暮里 4-17-7

近藤科学株式会社　サービス部　TEL　03-3807-7648　（サービス直通）

土日祝祭日を除く 9:00 ～ 12:00　 13:00 ～ 17:00

製品についての告知及びアップデータ等は弊社ウェブサイト http://www.kondo-robot.com に掲載されます。E-mail でのお問い合わせにつきましては、support@kondo-robot.com にて承りますが、回答までお時間を頂く場合がございます。

※Arduino 使用方法やプログラムの詳細等お問い合わせはお答えできない場合がありますので、あらかじめご了承ください。

ロボットキットの紹介

KXR シリーズ




『KXR』シリーズは、目的に応じて自由に組み換え可能なように外形を共通化した 2 種類のスペックのサーボモータ、ビス止めによって初めての方でも簡単・確実に組立可能な 30 種類以上の樹脂製フレームパーツ、ロボットのモーション(動き)作成を楽しく学べるコントロールボードとソフトウェアによって構成されています。『KXR』は、豊富に用意された同シリーズの部品を組み合わせるだけで、各セットからあらゆるロボット作例へ組み換えることができます。無線操縦機器や各種センサなどのオプションパーツも充実していますので、ユーザーオリジナルのロボット製作ですらも、容易に行えるよう設計されています。

今回使用する KXR-L4T-R は、9 個のサーボを搭載した 4 脚のカメ型ロボットです。4 点で本体を支えるため転倒することなく、目標に向かって移動することができます。パーツ点数も少ないため初心者にもお勧めのキットです。キット付属のパーツを組み替えることにより、足先を車輪にしたローバー型への換装が可能です。


ICS 機器

KXR に搭載されている KRS-3300 シリーズのサーボは、ICS 準拠のコマンドにより、角度制御や各種パラメータの設定が可能です。また、ICS 対応の受信機を接続し無線化することもできます。この説明書では、KRS サーボや受信機など ICS 対応機器をまとめて「ICS 機器」と表記する場合があります。

KXR では、KRS-3300 シリーズのサーボを使用することができます。キットに付属している KRS-3302 は、樹脂ギヤのサーボでトルクは 6.7kg・cm です。サーボはシリーズによってケースの寸法が共通化されていますので、無改造で高トルクの KRS-3304R2（13.9kg・cm）に換装することができます。

■商品情報『KRS サーボカテゴリー』https://kondo-robot.com/product-category/servomotor/krs

標準サーボ
KRS-3302 ICS

高トルクサーボ
KRS-3304R2 ICS

受信機
KRR-5FH

RCB-4mini（KXR 搭載済み）

RCB-4 は弊社 KRS サーボを搭載したロボットを動かすためのコントロールボードです。2 系統の ICS 機器用シリアル端子を搭載し、サーボを最大 35 個制御できます。また、ICS 対応の受信機を接続することで簡単に無線コントロールすることが可能です。ジャイロセンサや加速度センサなどアナログセンサを接続できる AD 端子も実装しています。モーション作成ソフト HeartToHeart4 と組み合わせることで、容易にロボットの動きを作ることができます。HeartToHeart4 について詳しくは、下記『HeartToHeart4』の項をご覧ください。

■商品情報『RCB-4mini』　http://kondo-robot.com/product/RCB-4_mini

HeartToHeart4

HeartToHeart4（以下 HTH4）は、ロボット用ビジュアルプログラミングソフトウェアです。

ほとんどの作業をマウス操作で行うことができ、プログラミングの知識がない方でも簡単にモーション作成が可能です。また、ロボットの姿勢の調整、ジャイロや加速度などオプションセンサの設定といったソフトウェアの作業をすべてカバーします。HTH4 がもつたくさんの機能を使って自由に自分だけのロボットを作り上げることができます。詳しくは下記ウェブサイトをご覧ください。

■商品情報『HeartToHeart4』　http://kondo-robot.com/product/hearttoheart4

ライブラリを使用する場合は、HTH4 の ver2.2 以降をご利用ください。それ以前のバージョンはアドレス等が違うため誤動作する原因となります。

RCB-4 内部のサーボモータの設定やロボットのモーションの作成も HTH4 を使用してください。以下の解説では、サンプルプロジェクトなど予め RCB-4 にプロジェクトが書き込まれ、各モーションが動作できる状態であることを前提に説明します。

Dual USB アダプターHS

RCB-4 と PC を接続するための USB アダプタです。コントロールボードと通信する場合は「シリアルモード（LED：緑）」で使用します。これを使用して、HTH4 で作成したモーションデータやサーボモータの設定情報を RCB-4 に書き込みます。モードを「ICS モード（LED：赤）」に切り替えることで KRS サーボや KRR-5 受信機など ICS 機器を直接 PC に接続することも可能です。

弊社各ロボットキットに付属しています。ただし、RCB-4 単品販売には付属していませんので、別途ご準備ください。

電源

電源は、キットに付属している ROBO パワーセル E タイプ 6N-800mAh(Ni-MH)、または AC アダプター（6V2A）（商品 No.03141　別売り）をご利用ください。ロボットは、サーボモータを多く使用するため、瞬間的に大電流を流すことがあります。充電式のバッテリを使用するか電源容量に余裕のある安定化電源、または AC アダプターをご用意ください。

学習用シールド＋Arduino

学習用シールド

学習用シールドは、弊社ロボット用コントロールボード RCB-4（HV/mini）や KRS サーボなど ICS 機器を Arduino UNO のシリアル端子（UART）、またはデジタル I/O 端子に接続するためのシールドです。Tx、Rx などの通信線や電源などの回路を用意する手間が省け、接続するだけで簡単に各機器の制御が可能となります。

商品情報『KXR プログラミング学習用シールドセット（Arduino 用）』https://kondo-robot.com/product/03154

●特徴

・ ICS 変換基板の回路を実装。KRS サーボや KRR-5FH 受信機など ICS 機器を制御することができます。

・ RCB-4 変換基板の回路を実装。RCB-4 のモーション再生やセンサの値を取得することができます。

・ ICS、RCB-4 の回路を同時に使用することも可能です。

・ ウェブサイトで公開している Arduino 用のライブラリを使用することが可能です。このシールドはデフォルトで SoftwareSerial に接続されていますが、はんだ付けにより HardwareSerial でもご利用いただけます。

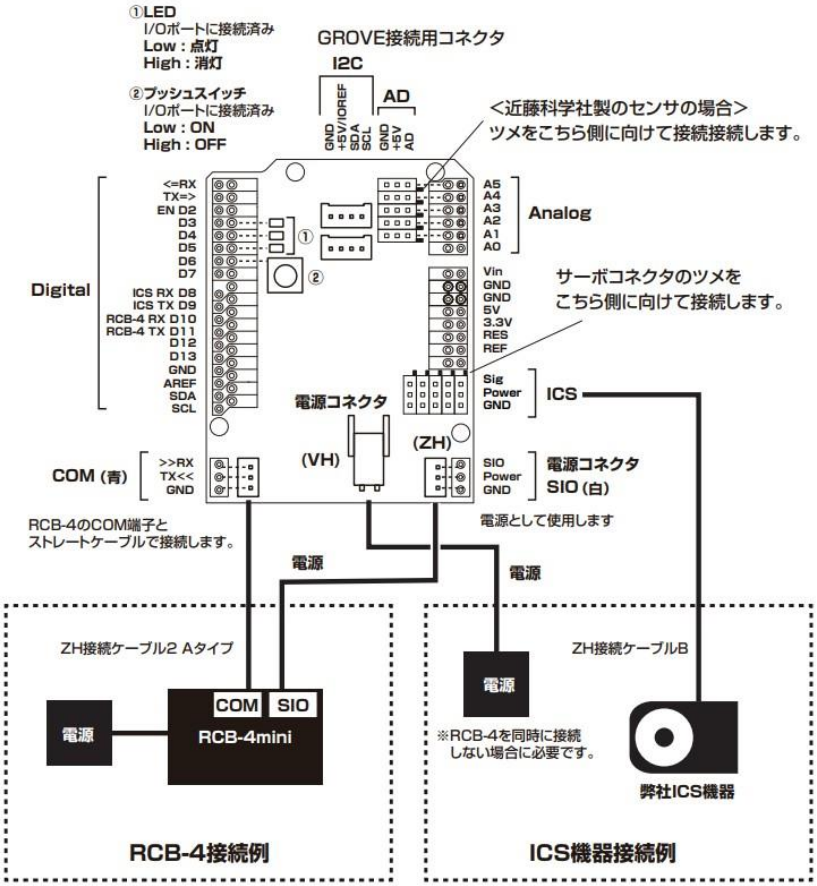
・ プログラミング学習で定番の LED 制御を行うため、3 つの LED を予め実装しました。タクトスイッチも 1 つ装備しています。

・ Arduino の AD 端子に接続されたヘッダピンを実装しています。

・ I2C 用に Seeed 社（GROVE）で採用しているコネクタを実装しました。

・ すべての端子をはんだ付け済みですので、買ってすぐにプログラミングを始められます。

●基板構成



製品の構成

ロボットと Arduino などマイコンボードとの間に学習用シールドを中継することにより、回路を自作することなく手軽に通信を行うことができます。シールドは、Arduino UNO に合わせたサイズになっています。また、端子ははんだ付け済みですのでそのまま使用できます。





Arduino について

Arduino はオープンソースハードウェアで、対応したマイコンボードが一般販売されています。

本製品は、Arduino UNO のデジタル I/O 端子を疑似的にシリアル端子として使用した SoftwareSerial で各機器と通信します。また、はんだ付けによるジャンパにより Arduino のシリアル端子(UART、HardwareSerial)を利用することも可能です。

●HardwareSerial と SoftwareSerial

Arduino などマイコンボードに実装されているシリアル端子(UART)を使用して通信することを HardwareSerial と呼びます。通常は HardwareSerial を使用することで安定した通信を行いますが、マイコンによってはシリアル端子が別の用途に使用されていたり、複数の機器と個別にシリアル通信が必要な場合あります。その場合、デジタル I/O 端子をソフトウェア上でシリアル端子のように使用します。この方法を SoftwareSerial と呼びます。SoftwareSerial は、本来ボードが持っている機能ではないため、不安定な場合があり通信が途切れる可能性があります。動作を安定させたいときは、HardwareSerial を使用することをお勧めします。

開発環境

開発環境は、Arduino IDE を用います。下記のアドレスの公式ウェブサイトからダウンロードしてご利用ください。

https://www.Arduino.cc/en/Main/Software

PC の対応等は上記公式ウェブサイトにてご確認ください。

本マニュアル作成時、弊社にて確認している Arduino IDE のバージョンは 1.8.5 です。（2018 年 6 月現在）

RCB-4 Library for Arduino

RCB-4 を Arduino から動かすためのライブラリおよびサンプルプログラムです。モーションを再生するための関数や、RCB-4 に接続したセンサの値を取得する関数などをご用意しました。詳細はこちらからライブラリに同梱しているマニュアルをご参照ください。

使用方法は後述の『Arduino ライブラリインポート』を参照にしてください。

ICS Library for Arduino

ICS 機器を Arduino から動かすためのライブラリおよびサンプルプログラムです。KRS サーボに角度を指定できる関数や、パラメータの変更などの各機能に合わせた関数をご用意しました。また、受信機 KRR-5FH で受信した送信機 KRC-5FH からのボタンデータを取得する関数もご利用できます。詳細はこちらからライブラリに同梱しているマニュアルをご参照ください。

使用方法は後述の『Arduino ライブラリインポート』を参照にしてください。

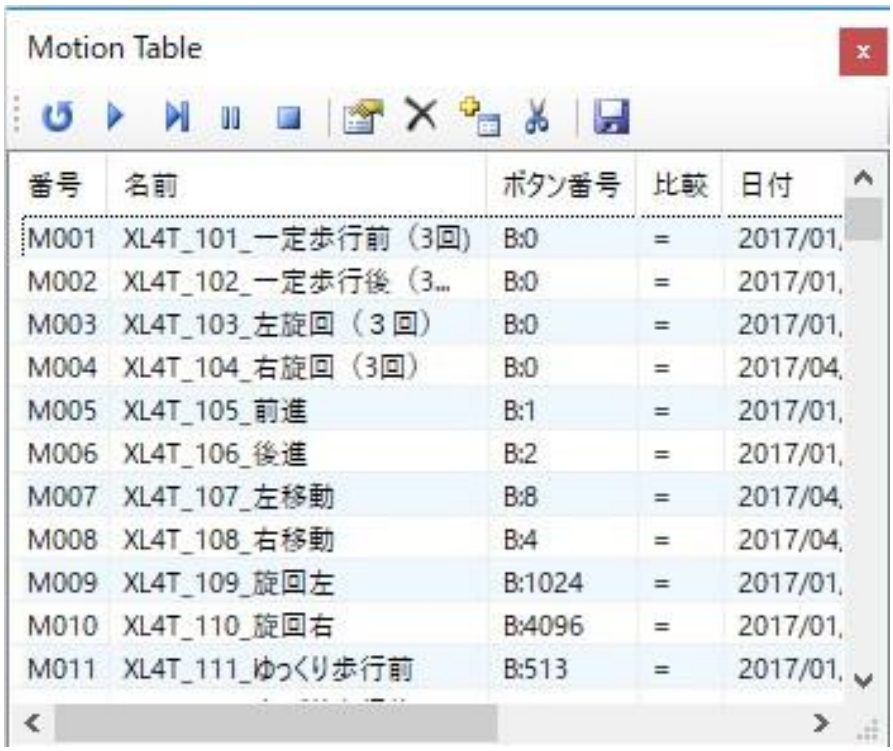
ロボットの動作確認

ロボットに書き込まれているプロジェクトについて改めて確認します。プロジェクトは、サンプルプロジェクトの**「Hello_KXR-L4T(V1.1)」**を書き込んでください。組立説明書 P.39 からを参考にトリム調整を行い、モーションが正常に再生できるかを確認してください。

プログラムでは、このプロジェクトに登録されている「モーション一覧ウィンドウ」の番号でモーションを指定します。リスト左の「M001」がモーション番号です。「M001」でしたらプログラム上では「1」と指定します。

■プロジェクトとは

プロジェクトとは、モーションデータやトリム、サーボの個数などロボットに必要な情報がひとまとめになったデータ群です。これをロボットにまとめて書き込むことで、ロボットの姿勢や動作を登録することができます。



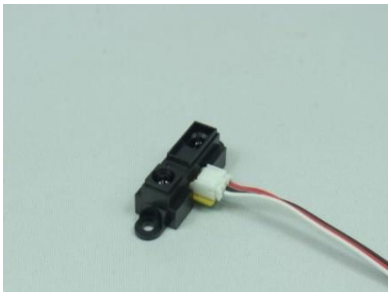
搭載方法

センサベースの組み立て

まずは、ロボットの本体に搭載する PSD センサを KXR のセンサベースに固定します。キットに付属している下記のパーツを用意してください。

・ボトムアーム 3300-20　・センサベース　・PSD センサ　・PSD センサ用接続ケーブル

・M2.6-10 ビス　×1　・2.6-4 フラットヘッドビス　×3



1)　PSD センサ用接続ケーブルを PSD センサのコネクタに接続します。

※接続向きにご注意ください

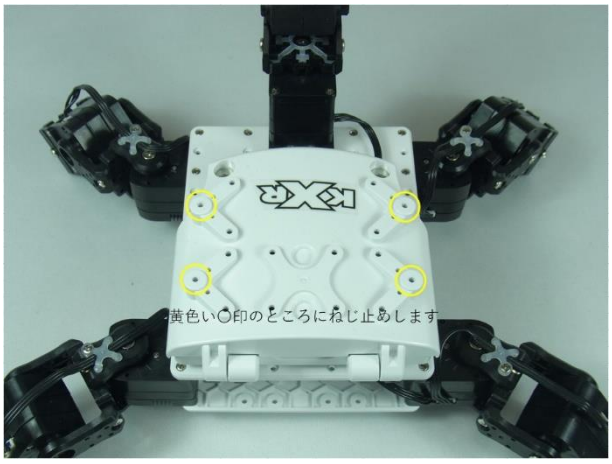
2)　センサベースにボトムアーム 3300-20 を固定します。ビスは 2.6-4 フラットヘッドビスを使用します。

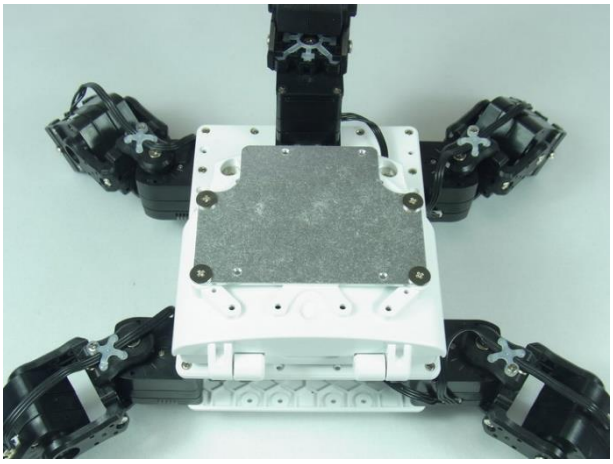
3)　センサベースに PSD センサを固定します。こちらも 2.6-4 フラットヘッドビスを使用します。

以上でセンサベースの組み立ては完了です。

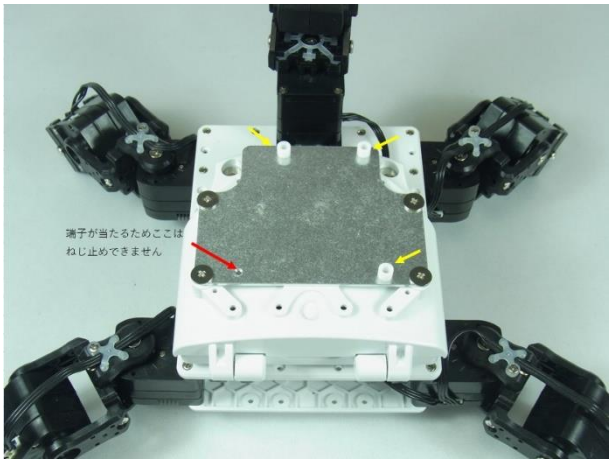
学習用シールドの搭載と配線

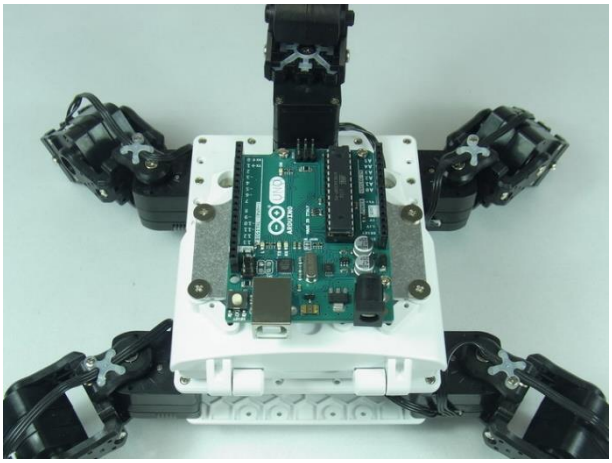
1)　バックパックの丸印のところに 2.6-4 フラットヘッドビスで板金を固定します。



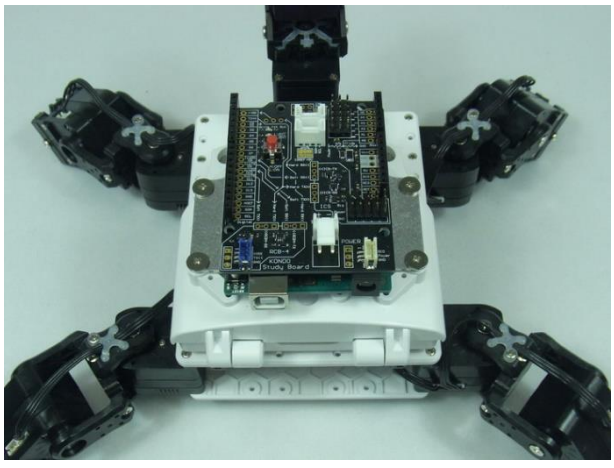


2)　スペーサーを挟み、Arduino を M2.6-8 ビスで板金に固定します。

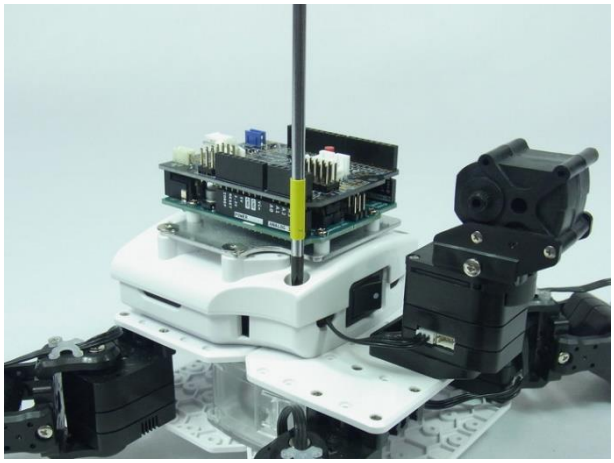




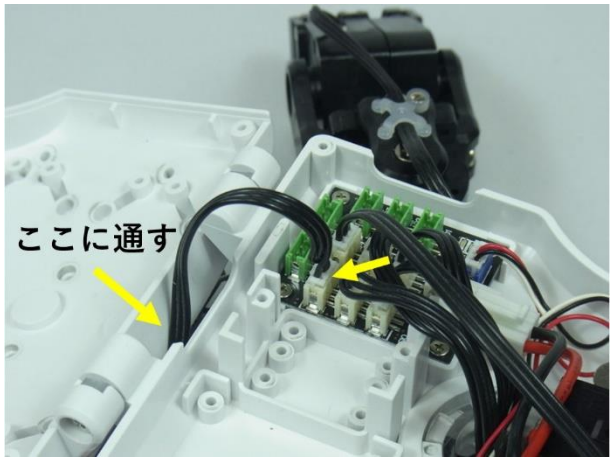
3)　Arduino UNO に学習用シールド接続します。※向きに注意してください。



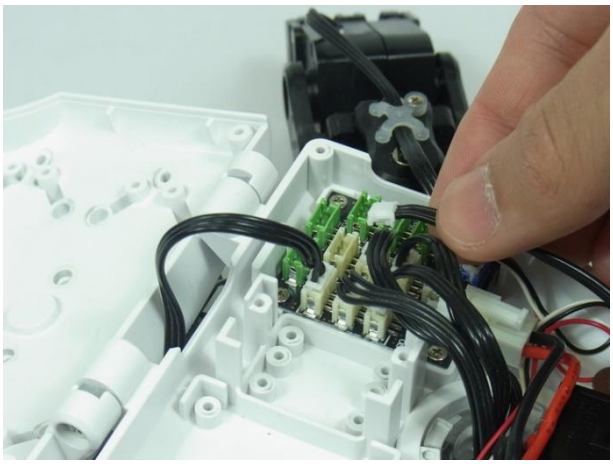
4） バックパックの 2.6-4 ビスを外し、バックパックを開きます。

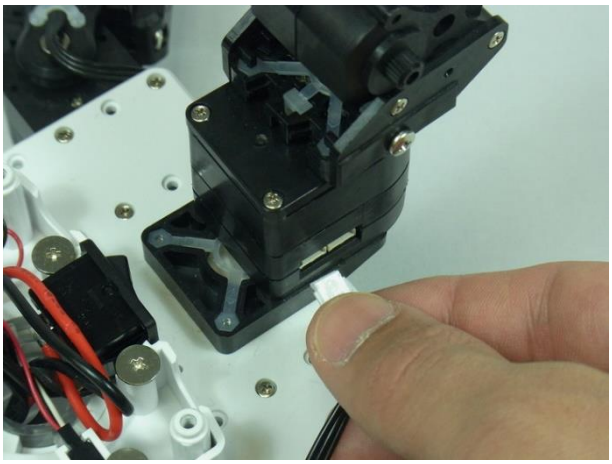


5） RCB-4mini の空いている SIO 端子に ZH 接続ケーブル 2 を接続し、バックパックカバーの隙間に通します。

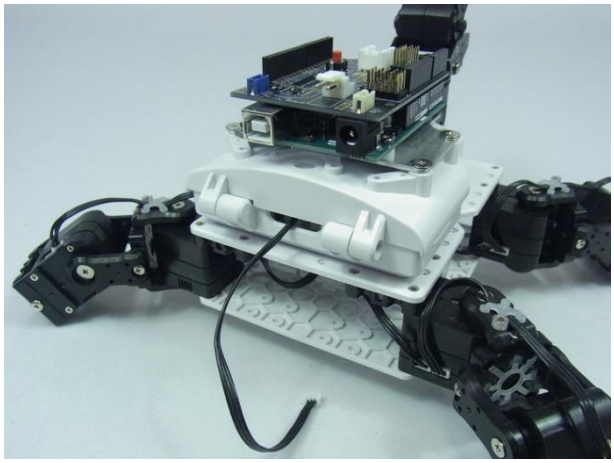


6） SIO7 からコネクタを抜き、首の ID0 のサーボからケーブルを外します。

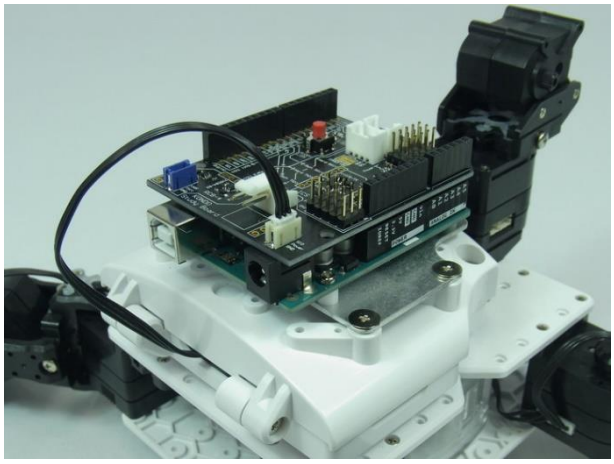




7） バックパックを閉じて 2.6-4 ビスで固定します。ケーブルが噛まないように注意してください。

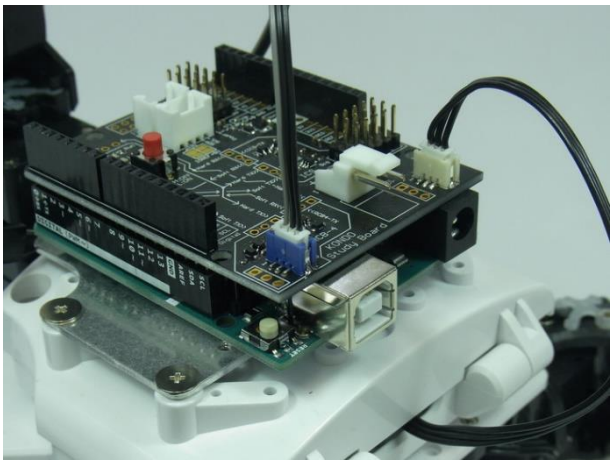


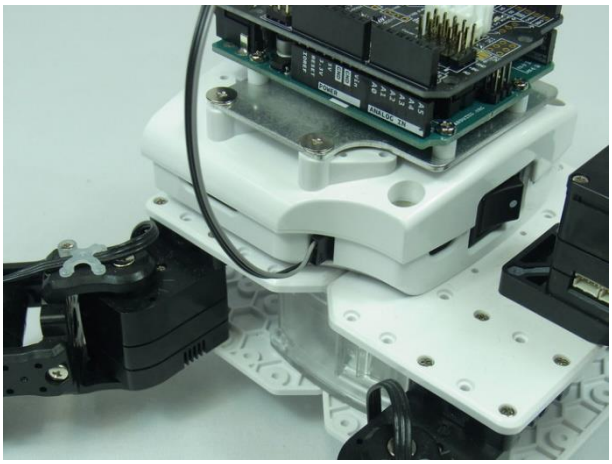
8） バックパックから出ているケーブルを学習用シールドの電源コネクタに接続します。



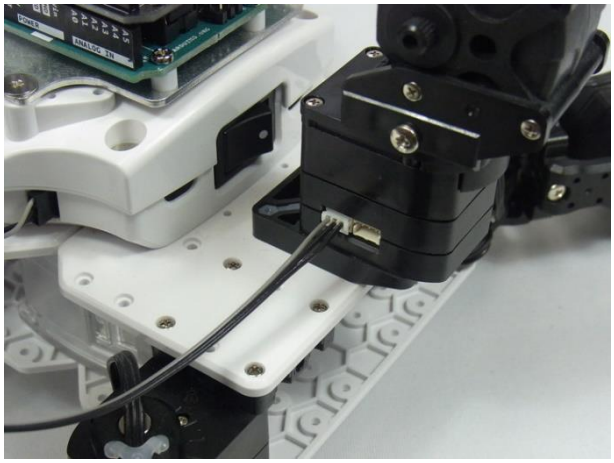
9） ZH 接続ケーブル B を学習用シールドの青い COM 端子に接続します。反対側をバックパックの COM 端子に接続します。

ZH 接続ケーブル B（ZH⇔サーボコネクタ）（200mm）

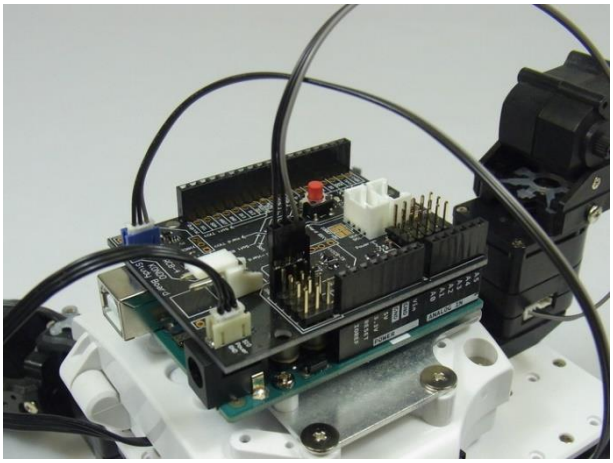


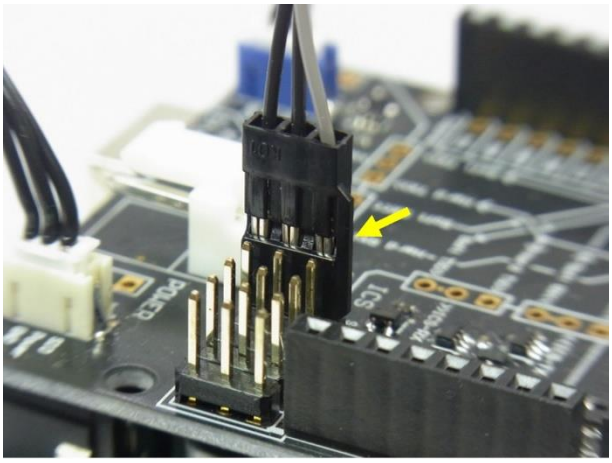


10）首の ID0 のサーボにもう一本の ZH 接続ケーブル B を接続します。

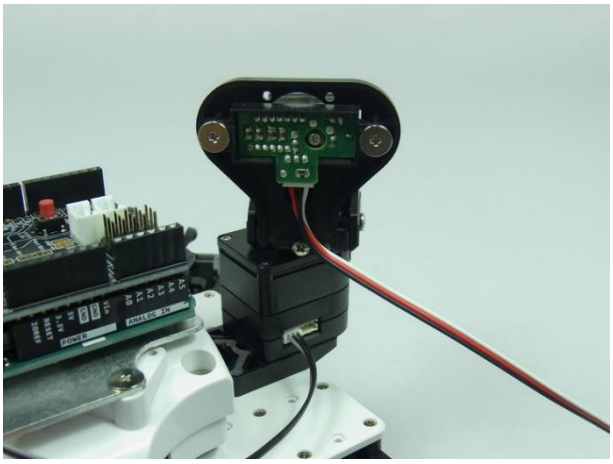


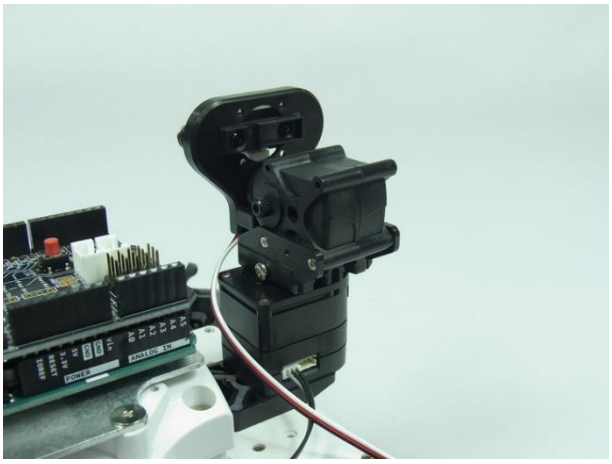
11）反対側を学習用シールドの ICS 端子にケーブルを接続します。端子が 5 ポートありますが、どこに接続しても構いません。爪の向きが画像の黄色い矢印側になるように注意してください。



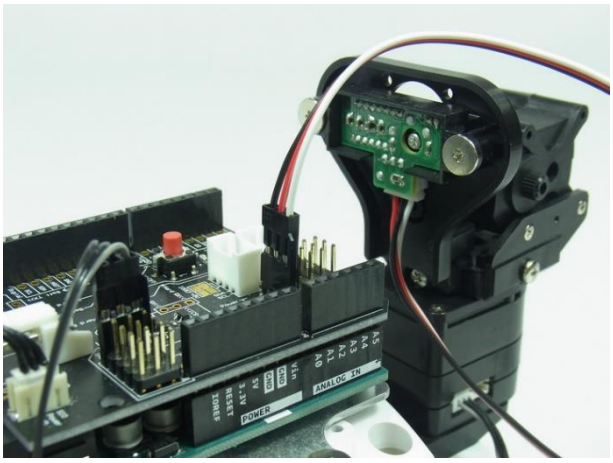


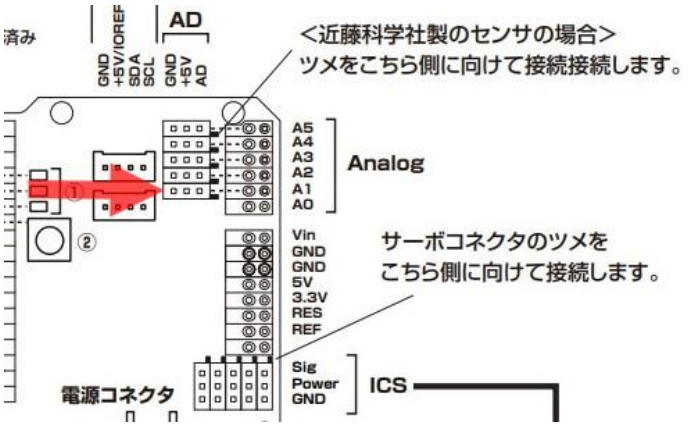
12）センサベースをロボットの頭部に固定します。センサベースのボトムアームを M2.6-10 ビスでジョイントベースに固定します。ダミーサーボの裏側（後頭部）に固定するようにしてください。頭部が正面を向いた時にセンサも正面を向いていることを確認してください。



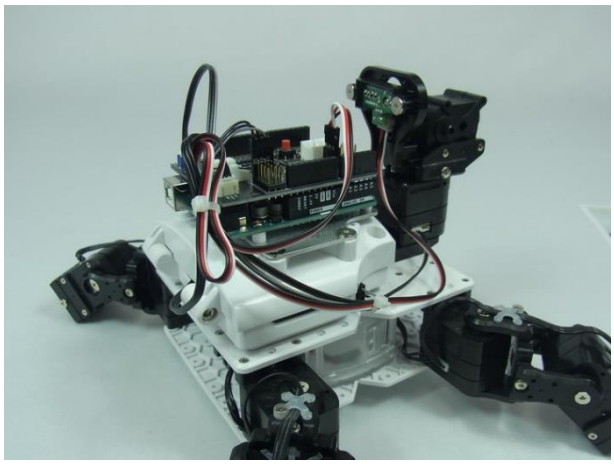


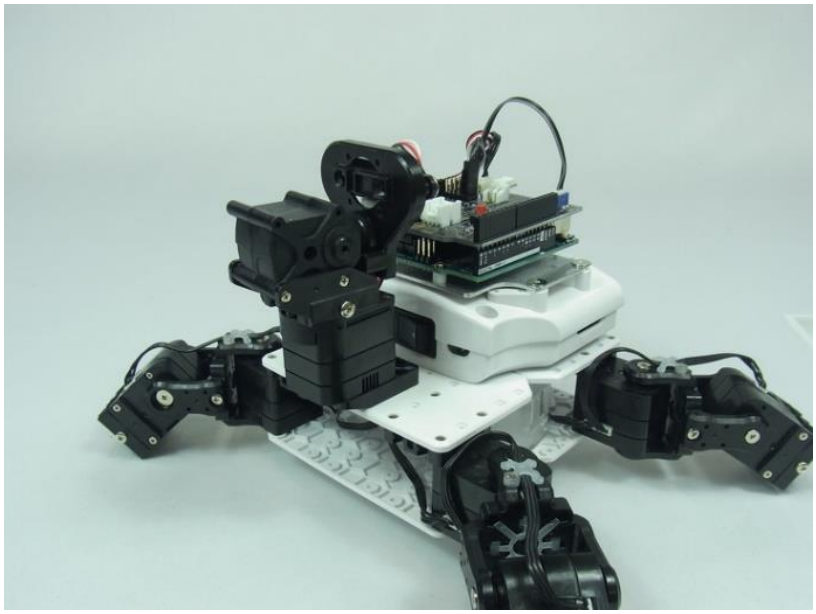
13）PSD センサの接続ケーブルを学習用シールドに接続します。Analog 端子の A1 に接続してください。ケーブルの白線がシールドの外側になるようにしてください。





14）ケーブルが引っかからないように結束バンドでまとめます。





以上で搭載と配線は完成です。

ソフトウェアの準備

Arduino ライブラリインポート

ライブラリは、弊社ウェブサイトよりダウンロード可能です。「RCB-4 用」と「ICS 用」の 2 種類が必要です。下記のアドレスよりダウンロードしてください。各ライブラリの詳細は、ダウンロードしたフォルダ内の各マニュアルを参照してください。

■RCB-4 用ライブラリ

ロボットに搭載しているコントロールボード RCB-4 と Arduino が通信するためのライブラリです。

http://kondo-robot.com/faq/rcb4-library-a1　（圧縮ファイルですべてを 1 つにまとめています）

■ICS 用ライブラリ

ロボットの関節に使用している KRS サーボなど ICS 機器と Arduino が通信するためのライブラリです。

http://kondo-robot.com/faq/ics-library-a2　（圧縮ファイルですべてを 1 つにまとめています）

下記よりインポート方法をご紹介します。

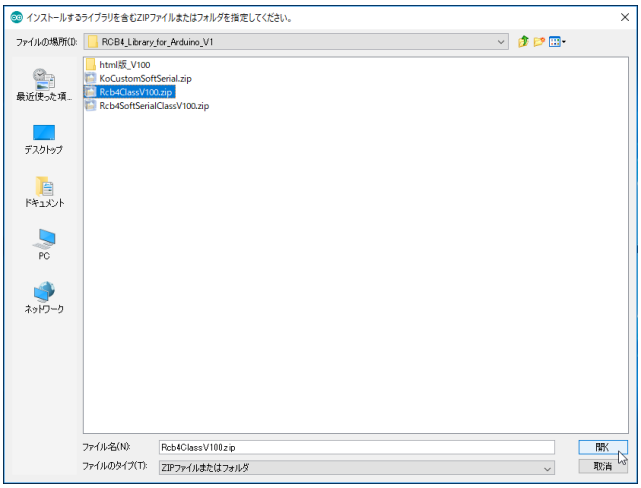
※インポート方法は Arduino　IDE のバージョン 1.8.3 を例にします。

①　弊社ウェブサイトよりライブラリ一式をダウンロードし、解凍します。生成されたフォルダ内の zip ファイルは解凍しないでください。

②　Arduino IDE を起動します

③　メニューバーの「スケッチ」　→「ライブラリをインクルード」
→「 .zip 形式のライブラリをインストール…」 を選択します

④　『インストールするライブラリを含む ZIP ファイルまたはフォルダを指定してください』となり、ファイル指定ダイアログが表示されますので、ダウンロードした zip ファイルの場所を指定します。



全ての zip ファイルをインポートします。学習用シールドは、デフォルトで SoftwareSerial に配線されているため、ライブラリも SoftwareSerial を利用しますが、HardwareSerial 内の関数も利用しますので Rcb4Class、IcsClass も忘れずにインポートしてください。

■RCB-4 の場合

・HardwareSerial の場合：Rcb4ClassVxxx.zip

・SoftwareSerial の場合：Rcb4SoftSerialClassVxxx.zip

KoCustomSoftSerialVxxx.zip　※

■ICS の場合

・HardwareSerial の場合：IcsClassVxxx.zip

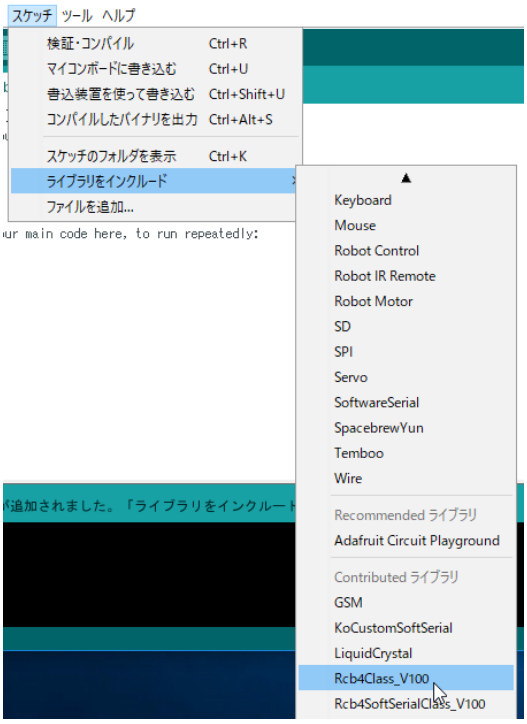
・SoftwareSerial の場合：IcsSoftSerialClassVxxx.zip

KoCustomSoftSerialVxxx.zip ※

※KoCustomSoftSerialVxxx.zip は、RCB-4、ICS どちらのフォルダにもありますが、どちらか一方がインポートされていれば問題ありません。

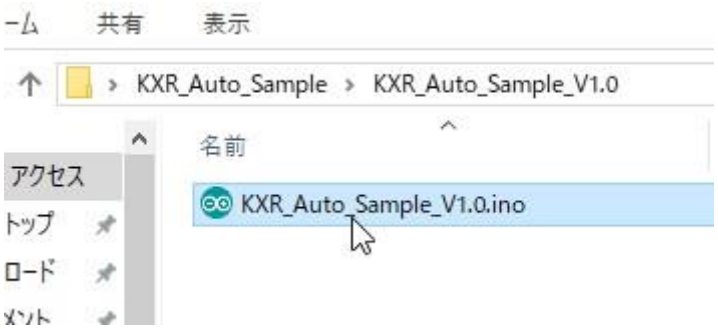
⑤ エラーでなければ、ライブラリがインポートされます。手順③で表示した「ライブラリをインクルード」リスト内に「Rcb4Class_Vxxx」または、「Rcb4SoftSerialClassVxxx」の表示がありましたら成功です。
ICS も同様に「IcsClass_Vxxx」「IcsSoftSerialClassVxxx」と表示されますので確認してください。

※右の画像は Rcb4Class のバージョン 100 をインポートした場合です



Arduino サンプルプログラム

Arduino のサンプルプロジェクトは、ダウンロードしたフォルダ内の「KXR_Auto_Sample_V1.0.ino」を使用します。ダブルクリックで Arduino IDE を起動してください。



サンプルプログラム解説

サンプルプログラム『KXR_Auto_Sample』では、PSD センサで読み取った距離データをもとに、もっとも近くにある物体に近づく、というプログラムを実行します。この手順は下記のようになっています。

① 首を旋回して周囲の距離を測る

② 読み取った距離データを距離が近い順番に並べる（ソート（解説は後述））

③ 首を旋回したときに、最も近い場所を読んだ角度に機体を向ける

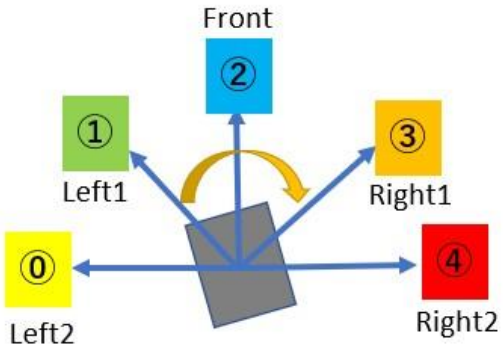
④ 直進方向に物体があったら前進モーションを再生する

⑤ ある一定の距離に近づいたら挨拶モーションを再生する

下記より、目標を探す方法とデータを並び替える方法の概要について説明します。

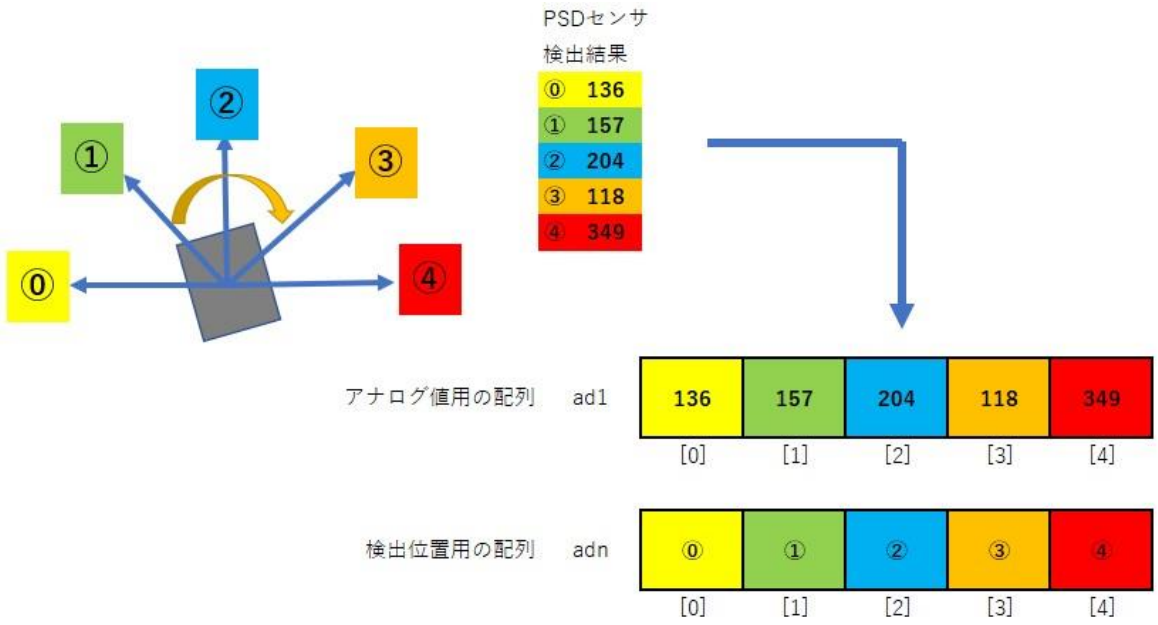
最も近距離の物体を探すには

このサンプルプログラムでは、PSD センサ（測距センサ）で距離を測り最も近いものに近づくようにプログラムされています。目標を探す方法の一つに、頭部を旋回している間に一定のタイミングで距離を読み取り、ソートして最大値を求めるという方法があります（PSD センサは、正面の一点のみを計測しますので、首を振りながら周囲の距離を測る必要があります）。PSD センサは、距離をアナログ値で出力します。これを Arduino のセンサ端子で受け取ります。PSD センサは、物体が近いほど高い値を示しますので、目標を探す場合は読み取った値の最大値を目標と判断し、その方向へ近づくプログラムを組みます。



センサーを搭載した頭部を上から見た図

上図は、ロボットが目標を探しているところを上から見た図です。0～4 の番号はアナログ値を検出するタイミングと番号です。このイメージに沿ってプログラムを組みます。

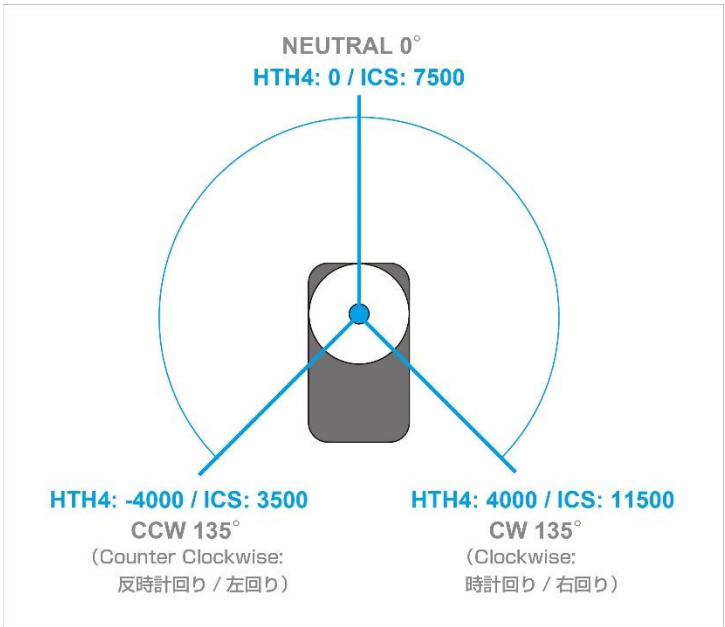


それぞれの場所で読み取ったアナログ値を配列（ad1[]）に代入します。さらに検出した位置も別の配列（adn[]）に代入しておきます。これにより、読み取った値がどのタイミングで読まれたかを判断することができます。

■サーボの動作範囲

KRS サーボをシリアルモードで回転させるときは、指令値として 3500～11500 を指定することができます。7500 を指定すると動作角の中心のニュートラル位置へ移動します。サーボの動作範囲は±135°ですので、3500 を指定するとニュートラル位置に対して-135 度（軸を正面に見て左方向）、11500 は+135 度に移動します。

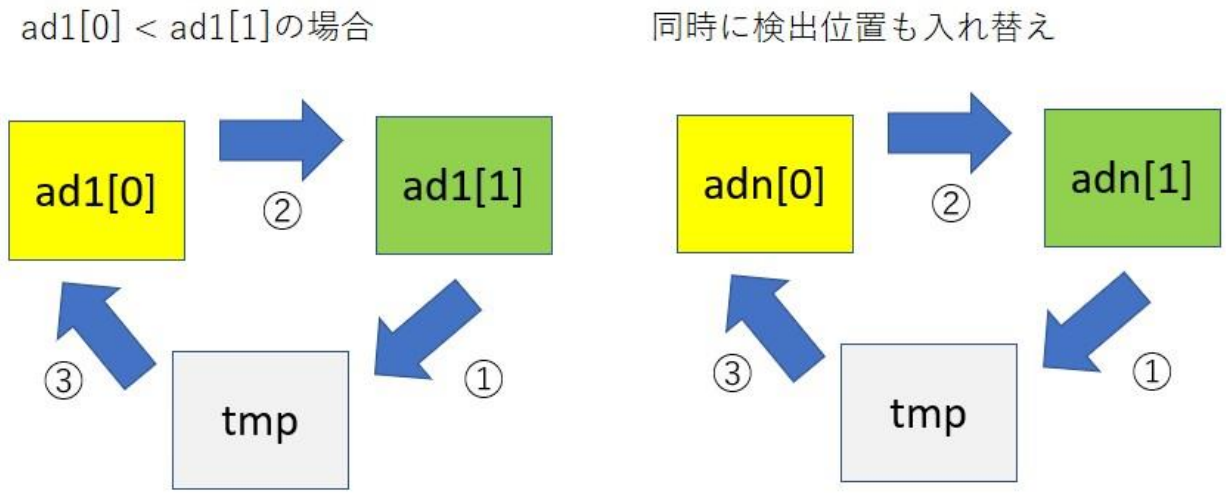
サンプルプログラムでは、首の向きを約 180 度回転させて物体の向きを探しますので、区切りが良いところで 5000～10000（±85 度）としました。この範囲を 5 分割に区切りますので、サーボの移動ステップは 1250 となります。



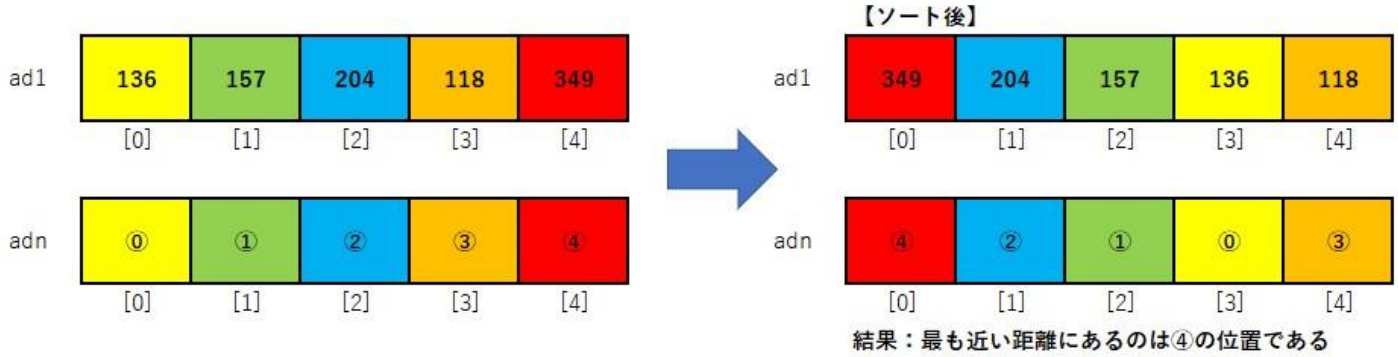
ソートとは

ソートは、値の大きい(小さい)順番に並び替えをする処理です。大きい順に並び替えた場合は、一番先頭にきたものが最大値になります。読み取った各検出位置のアナログ値をソートにかけて最大値を求め、どこに目標があるのかを割り出します。

配列に代入されたアナログ値を 2 つずつ比べます。もし、ひとつ前の値より今の値の方が大きかったら順番を入れ替えます。その際、そのまま大きい値を小さい値の配列番号に入れてしまうと、小さい値が消えてしまいます。そこで、大きい値は別の配列（tmp）に一時保存して、小さい値が移動した後に小さい値がいた配列のところにもう一度大きい値を代入します。これを繰り返すと、最も大きい値が配列の先頭に来て、最も小さな値が配列の一番後ろに来ることになります。



同時に、アナログ値を読み取った位置に対しても同じ処理をすることで、アナログ値の最大値をどこで読み取ったか、検出番号が先頭の配列に置かれることになります。



あとは、この番号にモーションを割り当て、分岐すればロボットは目標に向かって歩きだします。

KXR_Auto_Sample の詳細

このサンプルプログラムでは、これまで説明した目標を探す方法、ソートを利用して「目標に近づき挨拶をする」ための処理を実行します。下記より一行ずつ解説していきます。

```
#include <Rcb4SoftSerialClass.h>
#include <IcsSoftSerialClass.h>
```

Rcb4SoftSerialClass、IcsSoftSerialClass を使えるようにします。

```
//---モーション番号---
const byte FORWARD = 1;     //歩行（前）
const byte BACKWARD = 2;   //歩行（後）
const byte LEFTTURN = 3;    //旋回（左）
const byte RIGHTTURN = 4;   //旋回（右）
const byte GREETING = 21;   //挨拶
```

ロボットに登録したサンプルプロジェクトのモーション番号を、プログラム内で判別しやすくするためにモーション名の変数に代入します。数字は、モーション番号（「M001」など）です。

■const とは

const を使用することで、変更ができない固定の値(定数)を宣言できるようになります。例えば「FORWARD」変数へ宣言時に 1 を代入すると、プログラム中で他の数字を代入するとエラーになります。これにより、誤って変数内の数字を変更することなく、変数名で指定したモーション番号に対して正しいモーションを再生することができます。

```
//頭部サーボ検索分岐
const byte LEFT2 = 0;       //左からスタート
const byte LEFT1 = 1;
const byte FRONT = 2;       //頭部旋回角度の中心
const byte RIGHT1 = 3;
const byte RIGHT2 = 4;      //右でストップ
```

「目標を探すには」で解説した通り、ロボットの首を旋回している間に、等間隔の 5 箇所でアナログ値を読み取ります。ここでは、どこで読み取ったのかを分かりやすくするために変数に値を代入し、const で定数にしています。

```
//挨拶モーションを再生する閾値
const int GREETING_POINT = 400;

//頭部シリアルサーボの ID 番号
const byte HEAD_ID = 0;

//センサを接続した端子番号
const byte AD_CH = 1;

//シールド上のスイッチに配線されている端子番号
```

```
const byte SW_PIN = 6;
```

ここでも同様に変数に代入しています。

GREETINT_POINT は、目標にどこまで近づくかを指定する数値です。ロボットをより近づけたい、または離れたところで挨拶モーションを再生させたいときは、この数字を変更することで指示することができます。400 は約 13cm の距離です。

HEAD_ID は、ロボットの頭部に搭載した ID0 のサーボ ID です。ID を変更する場合は、この数字を書き換えます。

AD_CH は、シールドに接続した PSD センサの AD 端子番号です。

SW_PIN は、シールド上のスイッチに配線されている Arduino の端子番号です。

```
//---センサデータ保管用配列---
unsigned int ad1[5];          //検出したアナログ値
unsigned int adn[5];          //検出方向を保存
```

センサで検出した各値と、検出した位置番号を保管するための配列です。

```
//--------RCB4--------
const byte ACK_CHECK_PIN = 3;    //ACK の出力端子を指定
const byte CHECK_PIN = 4;          //通信チェックの端子を指定

const byte S_RX_PIN = 10;       //RCB 用送信端子を指定
const byte S_TX_PIN = 11;       //RCB 用送信端子を指定

const long BAUDRATE = 115200; //通信速度を指定
const int TIMEOUT_RCB = 500;   //タイムアウトの時間を指定

//--------ICS--------
const byte Sx_RX_PIN = 8;       //ICS 用送信端子を指定
const byte Sx_TX_PIN = 9;       //ICS 用受信端子を指定

const byte EN_PIN = 2;           //Enable 端子を指定
const int TIMEOUT_ICS = 100;   //タイムアウトの時間を指定

//下記の処理でクラスを使えるようにします。
Rcb4SoftSerialClass rcb4(S_RX_PIN,S_TX_PIN,BAUDRATE,TIMEOUT_RCB);   //RCB4Class の定義、softSerial 版
IcsSoftSerialClass krs(Sx_RX_PIN,Sx_TX_PIN,EN_PIN,BAUDRATE,TIMEOUT_ICS);   //ICSClass の定義、softSerial 版
```

RCB-4 クラス、ICS クラスを使用するための処理です。ここの詳細は各ライブラリのマニュアルをご参照ください。

TIMEOUT のみ RCB-4 クラスと ICS クラスで変数名が同名のため、混在を防ぐために名称を変更しました。

```
//----------------------------
//アナログ値平均
//正常な値を検出するために複数回読み取り平均化します。
//----------------------------
//int 型の関数のため戻り値の小数点以下は切り捨てます。
int ad_average(char ad_ch){
```

```
  int i;
  int adv = 0;

  for(i = 0; i < 3; i++){
    delay(10);
    adv += analogRead(ad_ch);    //3 回アナログ値を読み取り足していく
  }

  return adv/3;                  //3 回分のアナログ値を 3 で割り、戻り値とする
}
```

正常なアナログ値を取得できるように、アナログ値を複数回検出し平均を求めます。サンプルでは、3 回取得して平均するように設定しました。

```
//----------------------------
//ソート
//検出したデータを大きい順に並べ替え
//----------------------------
void bubble_sort(
  unsigned int t                 //検出回数
)
{
  int i, c;
  int tmp;

  //並べ替えるループ
  for(i = 0; i < t-1; i++){
    for(c = i+1; c < t; c++){
      if(ad1[c] > ad1[i]){

        //検出した値用の処理
        tmp = ad1[c];       //大きい値を tmp に一時保管
        ad1[c] = ad1[i];    //小さい値を大きい値の変数に代入
        ad1[i] = tmp;       //一時保管していた大きい値を代入

        //値の検出位置番号を並べ替え
        tmp = adn[c];
        adn[c] = adn[i];
        adn[i] = tmp;
      }
    }
  }
}
```

先で説明した通り、ソートで値の大きい順に入れ替えます。もし前の配列より後の配列の値が大きければ、大きい値を一度 tmp に保存し、小さい値

を大きい値が入っている配列に代入します。さらに tmp の値を小さい値が入っていた配列に代入すれば入れ替えは完了です。同じ処理を、検出した位置番号にも行うことで配列の先頭に最も大きい値を検出したときの位置番号が代入されます。この番号を頼りに Loop()内の Switch 文で再生するモーションを指定します。

```
//----------------------------
//検出エリア指定しデータ読み込み
//シリアルサーボの頭を動かしながら、データを読み込みます
//----------------------------
void search(
  byte sv_no,                  //頭部サーボの ID 番号
  byte ad_ch,                  //アナログポートの端子ナンバー
  unsigned int srt_Pos,        //検出スタート位置
  unsigned int stp_Pos,        //検出ステップ
  unsigned int t               //検索回数
)
{
  int i;
  unsigned int pos1 = 0;

  krs.setPos(sv_no, srt_Pos);        //サーボを検索開始位置に移動
  delay(500);

  for(i = 0; i < t; i++){

    pos1 = i * stp_Pos + srt_Pos;    //検出位置を指定
    krs.setPos(sv_no, pos1);         //検出位置へサーボを移動
    delay(200);

    ad1[i] = ad_average(ad_ch);   //読み取った AD 値を代入
    adn[i] = i;                   //検出位置の番号を代入
  }

  krs.setPos(sv_no, 7500);         //頭部を 7500（中央）へ移動
  delay(1000);

  bubble_sort(t);                  //データの大きい順に並べ替え
}
```

首を旋回させて物体との距離を検出する処理をここで行います。それぞれの検出位置での首の角度を指定し、上記の ad_average()で値を取得します。

pos1 = i * stp_Pos + srt_Pos;で検出位置のポジションデータを計算します。引数で受け取った stp_Pos に 1250、srt_Pos に 5000 が代入されています。最初は i が 0 ですので、0*1250+5000 でスタート位置と同じポジション 5000 になります。for 文 0～4 まで計 5 回繰り返しますのでポジションは、5000、6250、7500（中心）、8750、10000 となります。それぞれのポジションで検出した値を ad1[i]の配列に代入し、同時に読

み取った位置の番号を and[i]に代入します。

値が揃ったら bubble_sort()で最大値を求め、その検出位置を戻り値として返します。

```
void setup() {

  //--------RCB4 初期化--------
  pinMode(ACK_CHECK_PIN ,OUTPUT);       //ACK 確認を出力するピン設定
  pinMode(CHECK_PIN ,OUTPUT);           //確認を出力するピン設定
  digitalWrite(ACK_CHECK_PIN,LOW);
  rcb4.configData.word = 0xffff;        //ConfigData は間違えないように 0xFFFF で初期化
  while(rcb4.open(115200,1000) == false); //Config をとれないとモーション再生が大変なので、取得できるまでループ
  digitalWrite(ACK_CHECK_PIN,HIGH);     //OPEN できていたら ACK は返ってきている
  rcb4.checkAcknowledge();

//--------ICS 初期化--------
  krs.begin();                          //サーボモータの通信初期設定

  delay(500);
```

RCB-4 クラスと、ICS クラスの初期化処理です。ここの詳しい解説は、各ライブラリのマニュアルをご参照ください。

```
  krs.setStrc(HEAD_ID, 30);         //頭部サーボのストレッチを 30 に設定
  krs.setSpd(HEAD_ID, 60);          //頭部サーボのスピードを 60 に設定
  krs.setPos(HEAD_ID,7500);         //位置指令　頭部サーボを 7500（中央）へ
  delay(1000);
```

頭部サーボのストレッチとスピードを変更します。頭部をポジション 7500（ニュートラル位置）へ移動させて準備完了です。

delay(1000);は、サーボが目的地（ポジション 7500）まで到着するのを待つための処理です。以降のプログラムでも、モーションやサーボの移動が完了するまで delay();を入れて次の処理を待つようにします。

```
  pinMode(SW_PIN, INPUT);               //シールドのスイッチを入力に設定
  while(digitalRead(SW_PIN) == 1){}     //スイッチが押されたら loop()スタート
}
```

電源投入後、意図しないタイミングでロボットが動き出さないように、スイッチが押されるまでその場でとどまる処理を入れました。学習用シールドには、予めスイッチが実装されていますのでこれを利用します。このスイッチは Arudino の Digital6 に接続されています。while 分を利用してこの端子の状態を読み続けます。このスイッチは、常時 HIGH の状態ですので while 分の条件の通りこの端子が HIGH（1）の場合は何もせずにここに留まります。スイッチが押されると LOW になり条件から外れますので、ループから抜け次の処理が実行されます。

```
void loop() {

  //データを検出し大きい順に並べ替える
  //(サーボ ID, ADch, 検出スタート位置, 検出ステップ, 検出回数)
```

```
search(HEAD_ID, AD_CH, SEARCH_START, SEARCH_STEP, 5);
```

上記で解説した首を旋回しながら値を検出し、最も近い場所にある物体を探すための処理を実行します。各変数の値は、プログラムの最初に代入しましたのでそちらを参照してください。

```
//検出したアナログ値の最大が閾値以上だったら挨拶モーション再生
if(ad1[0] > GREETING_POINT){
  rcb4.motionPlay(GREETING);
  delay(2000);
}
```

PSD センサは、物体に近づくほど高い値を出力しますので、ロボットが物体に近づくほど値が大きくなります。この検出した値が GREETING_POINT を超えたら挨拶モーションを再生します。

ただし、PSD センサは近づきすぎると値が反転し、距離が計測できなくなるので注意が必要です。最大値は 650（約 4cm）ですので、この手前でモーションを再生するようにしてください。

■距離と値の関係を計測する方法

目標との距離と取得する値については、予め serialPrintln();を利用して距離と値を計測しておきます。「ツール」の「シリアルモニタ」を使用することで Arduino が取得した値を表示することができます。プログラムは「スケッチ例」の「Analog」にある「AnalogInOutSerial」を使用します。初期値で「const int analogInPin = A0;」となっているところを「const int analogInPin = A1;」に変更して使用してください。プログラム書き込み後に「シリアルモニタ」を起動すると取得した値が自動で表示されます。

```
//閾値以下だったら最大値を頼りに移動
  else{
    //最も大きい値を検出した位置番号をもとにモーションを再生
    switch(adn[0]){

     case LEFT2:     //最も左が最大だったとき
      rcb4.motionPlay(LEFTTURN);  //左旋回モーション再生
      delay(2000);                    //モーションが終わるまで処理を待つ
      break;

      case LEFT1:    //中央より一つ左が最大だったとき
        rcb4.motionPlay(LEFTTURN);  //左旋回モーション再生
        delay(2000);                    //モーションが終わるまで処理を待つ
        break;

      case FRONT:    //中央が最大だったとき
        rcb4.motionPlay(FORWARD);  //前進モーション再生
        delay(2000);                    //モーションが終わるまで処理を待つ
        break;

      case RIGHT1:  //中央より一つ右が最大だったとき
        rcb4.motionPlay(RIGHTTURN);  //右旋回モーション再生
```

```
      delay(2000);                    //モーションが終わるまで処理を待つ
      break;

    case RIGHT2:  //最も右が最大だったとき
      rcb4.motionPlay(RIGHTTURN);  //右旋回モーション再生
      delay(2000);                    //モーションが終わるまで処理を待つ
      break;

    default:       //どの条件とも会わなかったとき
      digitalWrite(5, HIGH);          //シールド上の LED 点灯
      delay(1000);
      digitalWrite(5, LOW);           //シールド上の LED 消灯
      break;
  }
}
```

bubble_sort()を実行することで、検出した値の中で最も大きい値とその検出位置が配列の 0 番に代入されています。よって、adn[0]には最大値の位置が代入されていますので、この値をもとに switch 文で再生するモーションを選択します。

・adn[0] = 0～1 の場合　ロボットから左側に物体がありますので「左旋回」モーションを再生します。

・adn[0] = 2 の場合　=　直線方向に物体がありますので「前進」モーションを再生し、物体に近づきます。

・adn[0] = 3～4 の場合　=　ロボットから右側に物体がありますので「右旋回」モーションを再生します。

これを繰り返すことで、ロボットが物体の方向に向き、前進で近付きます。やがて、物体との距離が縮まりますと検出する値が大きくなりますので、GREETING_POINT を超えたところで「挨拶」モーションを再生します。

以上が、このサンプルプログラムの一連の流れになります。

default は、どの条件にも合わないときの処理です。エラーを表示するために Dijital5 に接続されたシールド上の LED が点滅する処理を実行します。

モーション再生方法について

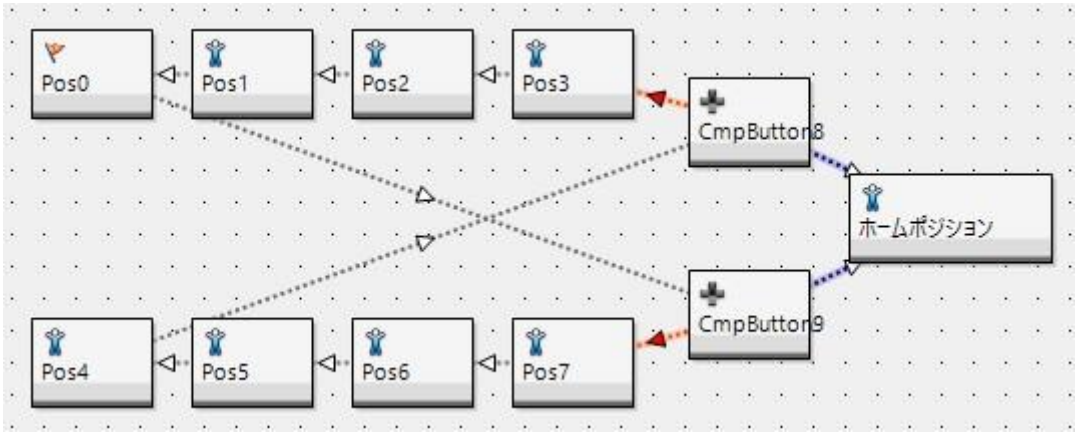
モーションの再生方法は 2 種類あります。これまでのサンプルプログラムで使用したモーション番号を指定して再生する motionPlay()と、各モーションに割り当てたボタン番号を指定して再生する setKrrButtonData()です。

motionPlay()は、指定した番号のモーションを再生するシンプルな関数ですが、予め登録したモーションの歩数しか再生できませんので自由に歩数を指定することができません。

対して setKrrButtonData()は、ボタン番号でモーションを再生しますので、無線コントローラで操作するように「ボタンを押している間再生し続ける」のと同じ処理が可能です。これにより、例えば物体を正面に見付けた時に、「閾値まで歩き続ける」などの処理が可能になります。また、旋回も同様に、「目標が目の前に来るまで旋回し続ける」という処理もできます。

■ロボットの操縦方法について

ロボットの操縦方法は、搭載するモジュールによって異なります。送信機 KRC-5FH とペアで使用する受信機 KRR-5FH をロボットに搭載することで、ボタン操作による無線コントロールが可能になります。この無線コントローラーを使用して HTH4 で設定したボタン番号に該当するモーションを再生できます。また、ボタンを押されている間は動作を続けるなどの処理が可能です。この続けるか、停止するかの分岐処理はモーションを作成する際の機能として用意されている CmpButton を使用します。例えば、歩行モーションで「ボタンが押されている間前進し続ける」処理を行うときは、次の一歩を出す前に CmpButton を配置することで、ボタンが押されていれば次の一歩を踏み出す、押されていなければ停止する、という処理ができます。先ほど説明した setKrrButtonData()では、ボタンデータを RCB-4 に送り続けることによって CmpButton による分岐を指定することができます。



setKrrButtonData()でモーション再生を指定する方法

先の解説で説明した通り、ボタン番号を指定してモーションを再生します。指定方法は、ライブラリで予め用意した予約語を指定するだけですので非常に簡単です。ただし、モーション番号で指定する motionPlay()がモーション終了とともに自動で止まるのに対して、setKrrButtonData()は停止を指示しないと止まりません。この停止の指示は KRR_BUTTON_NONE を送ることによって実行されます。

ボタン割り当てと予約語については、『Rcb4Library_forArduino_1_0.pdf』の P.31 をご参照ください。

例えば 3 秒間前進モーションの再生する場合は下記のように記入します。

```
rcb4.setKrrButtonData (rcb4.KRR_BUTTON_UP);
delay(3000);
rcb4.setKrrButtonData (rcb4.KRR_BUTTON_NONE);
```

サンプルプロジェクトの「Hello_KXR-L4T」にて、前進モーションにはボタン番号“1”が割り当てられています。これは rcb4.KRR_BUTTON_UP（0

ｘ0001）に該当しますので引数に記入します。この関数の後、delay(3000);により処理がここで留まりますのでロボットは前進し続けます。その後、KRR_CUTTON_NONE が送信されたことによりロボットは停止します。

解説は以上です。このサンプルプログラムでは目標を探して移動するための最低限な処理を行っています。工夫をすればより短時間で成功率の高いプログラムができますのでぜひチャレンジしてください。組立説明書の最後にオプション製品の一覧があります。KXR シリーズは組み合わせることで独自のロボットに改造することができます。プログラムと同時にロボット本体の改造もお勧めします。