ロボット博士養成講座

ロボティクスプロフェッサーコース

二足歩行ロボット①

第2回

二足歩行ロボットの組み立て②

講 師 用

目　　次

○　授業開始にあたって

授業のはじめは、着席させ、大きな声であいさつしてから始めます。

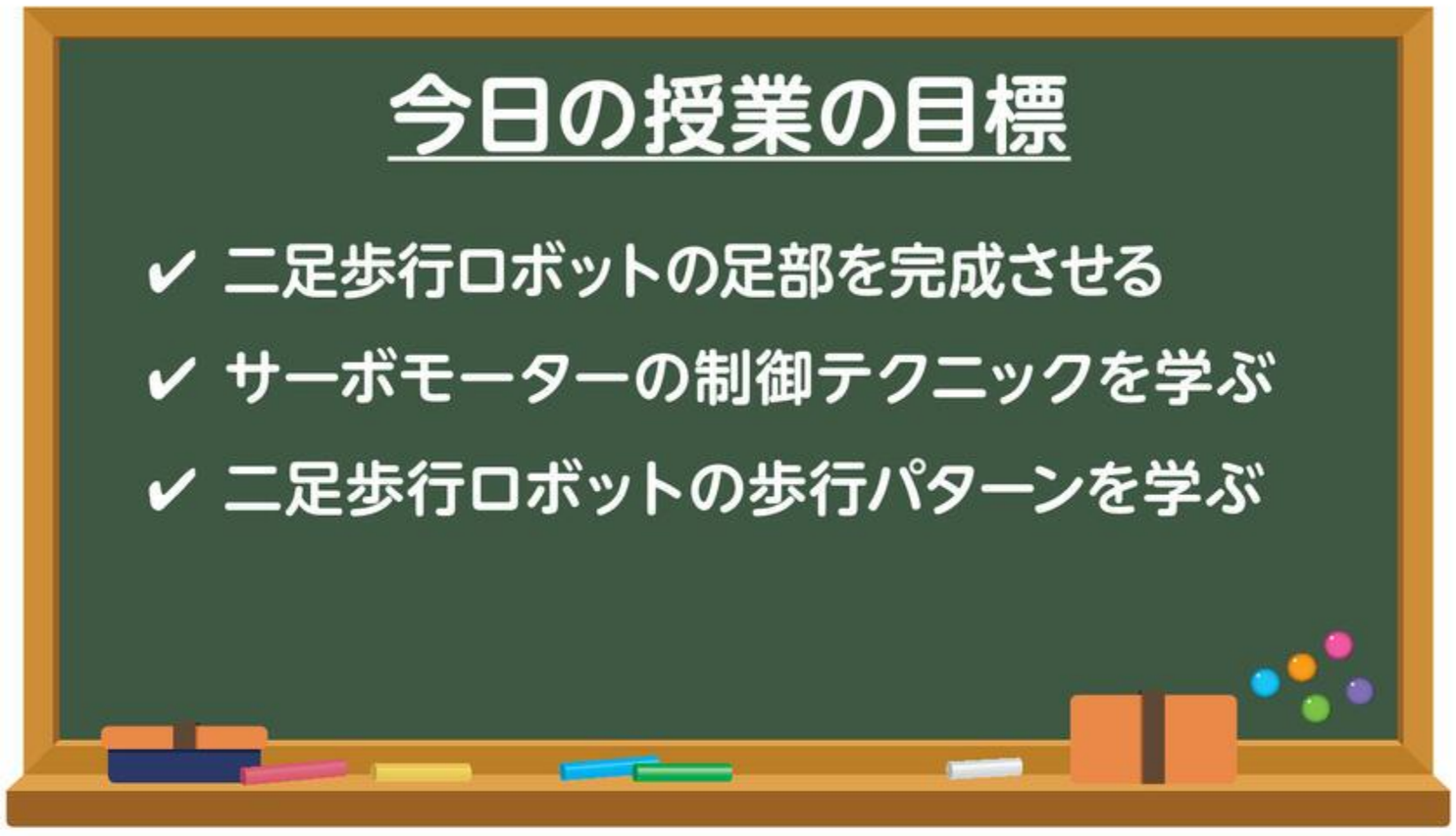
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（授業の目標を明確化することは大変重要なことですので、生徒によく理解させます）

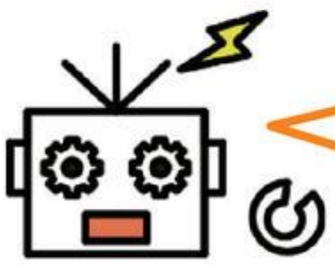
目安時間は授業時間 120 分のうち、休憩 10 分程度取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0.　二足歩行ロボットの組み立て②（目安５分）

0.0.　「二足歩行ロボットの組み立て②」でやること



第1回では、「サーボモーターの仕組み」と「原点位置合わせ」について勉強しました。
そして、サーボモーターを使うときに取り込む「Servoライブラリ」の内容を見て、サーボモーターに与えられるパルス信号の処理を確認しました。
プログラムを構成するライブラリを見ていくと、モーターやセンサーがどんな制御信号を受けて動いているのかを紐解いていくことができますが、関数がたくさん出てきて、慣れないうちは理解が難しいですね。
第2回では、ロボットの組み立てをすすめて、ロボットの下半身部分を完成させましょう。
また、コントローラーで操作しながら、二足歩行ロボットの歩行パターンをマスターしましょう。



サーボモーターの制御をマスターして、今日もまた一歩前進！

0.1.　必要なもの

今回使用するパーツは、第1回の「必要なもの」の残りです。
第1回のテキストを確認してください。

１．　組み立て（目安 40 分）

1.0.　足の組み立て

＜組み立て手順①＞

C-4パーツ（×2）とオイレスブッシュ（×4）を使い、図のように組み立ててください。
C-4 パーツは、どちらが右足、左足になるのか図をよく見て組み立てましょう。

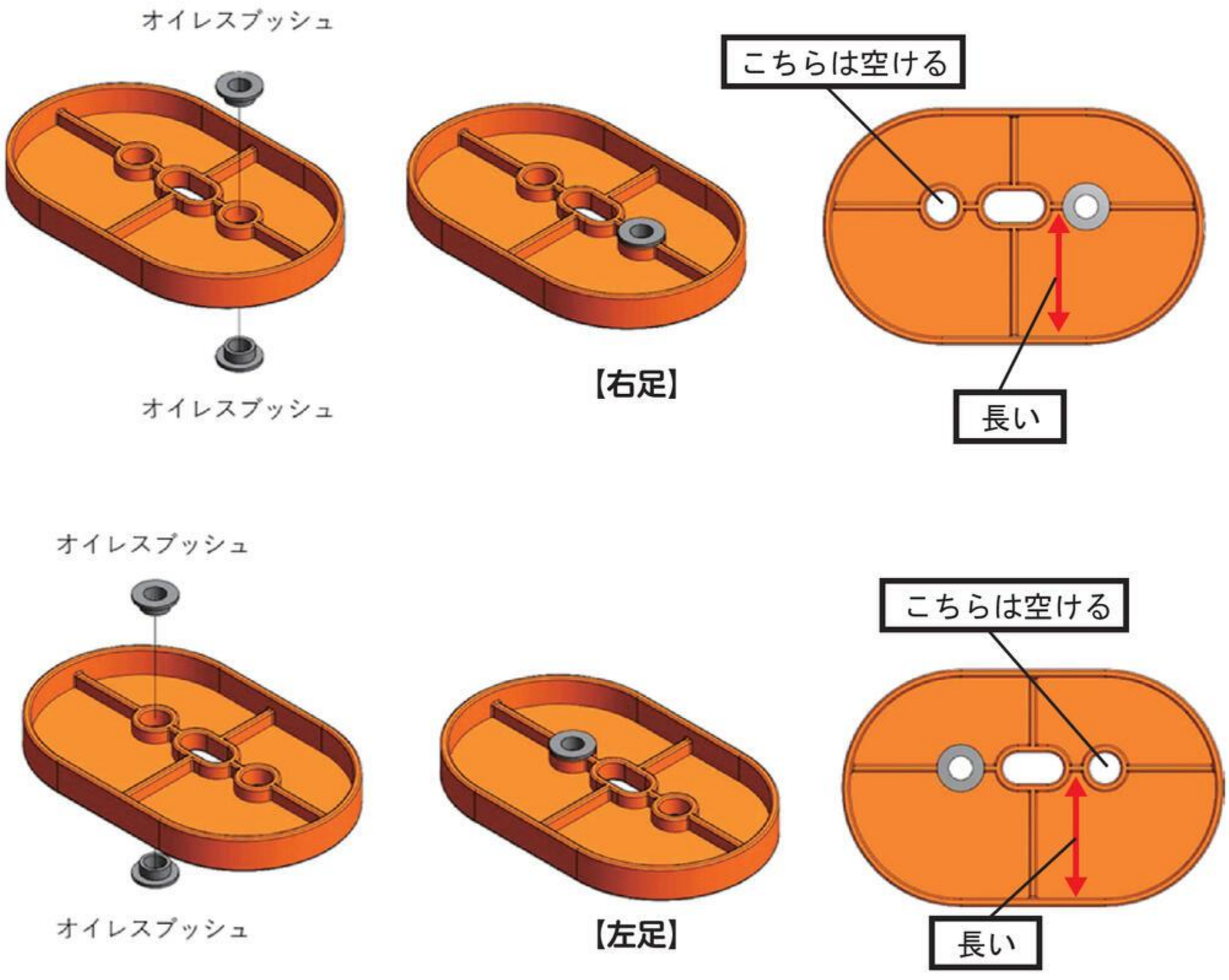


図 1-0　C-4 パーツ

次に、前回組み立てた足関節部分のサーボモーター（× 2）を用意してください。
サーボモーターは念のため原点位置調整ができているか確認をしましょう。

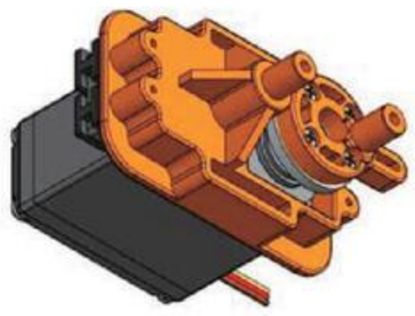


図 1-1　足関節

＜組み立て手順②＞

図のように、足関節になるC-2パーツの長穴にC-3パーツをさし込んでください。同じものを二組つくりましょう。また、C-3パーツは、さし込みの向き（サーボモーター側から入れる）に注意して組み立てましょう。

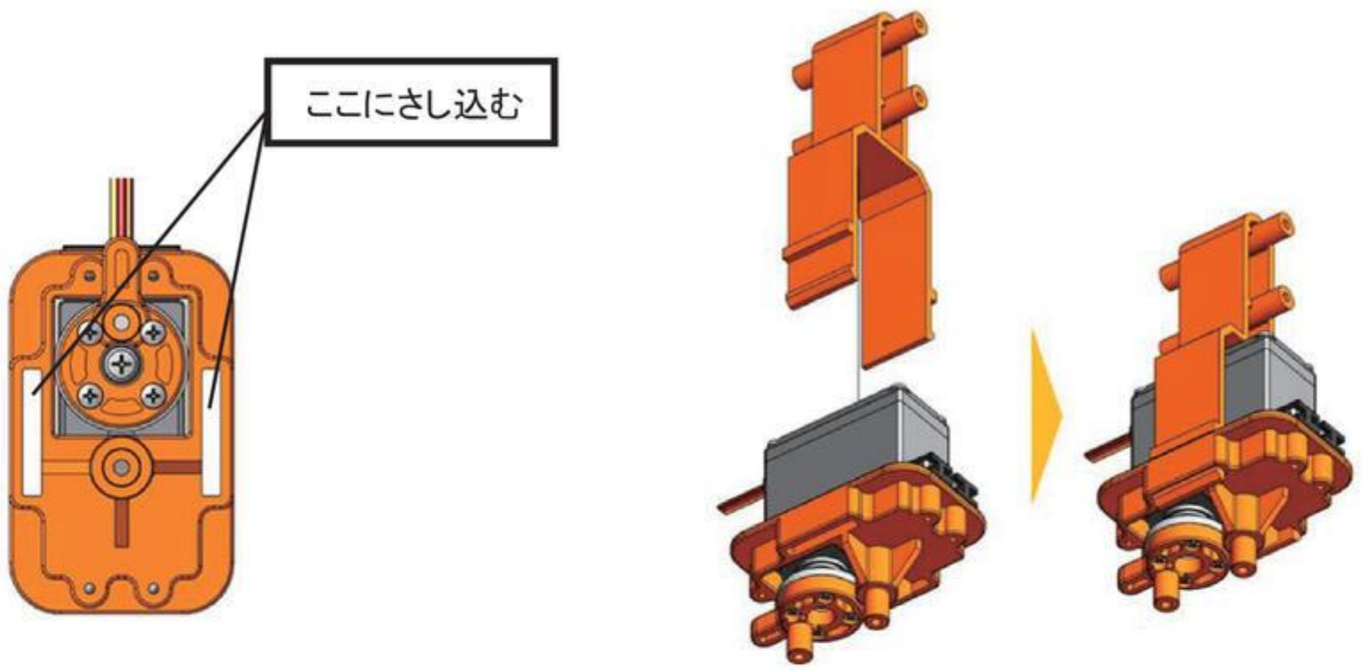


図 1-2　足の組み立て

＜組み立て手順③＞

次に、**図 1-2** で組み立てたものに、オイレスブッシュ（× 2)、M3L6 フラットヘッドビス(× 2)を使って、C-4 パーツを組み付けましょう。図のようにオイレスブッシュは C-4 パーツの楕円形の穴に両面から挟み込みように取り付けて、M3L6 フラットヘッドビスで C-1 パーツの凸部に組み付けてください。

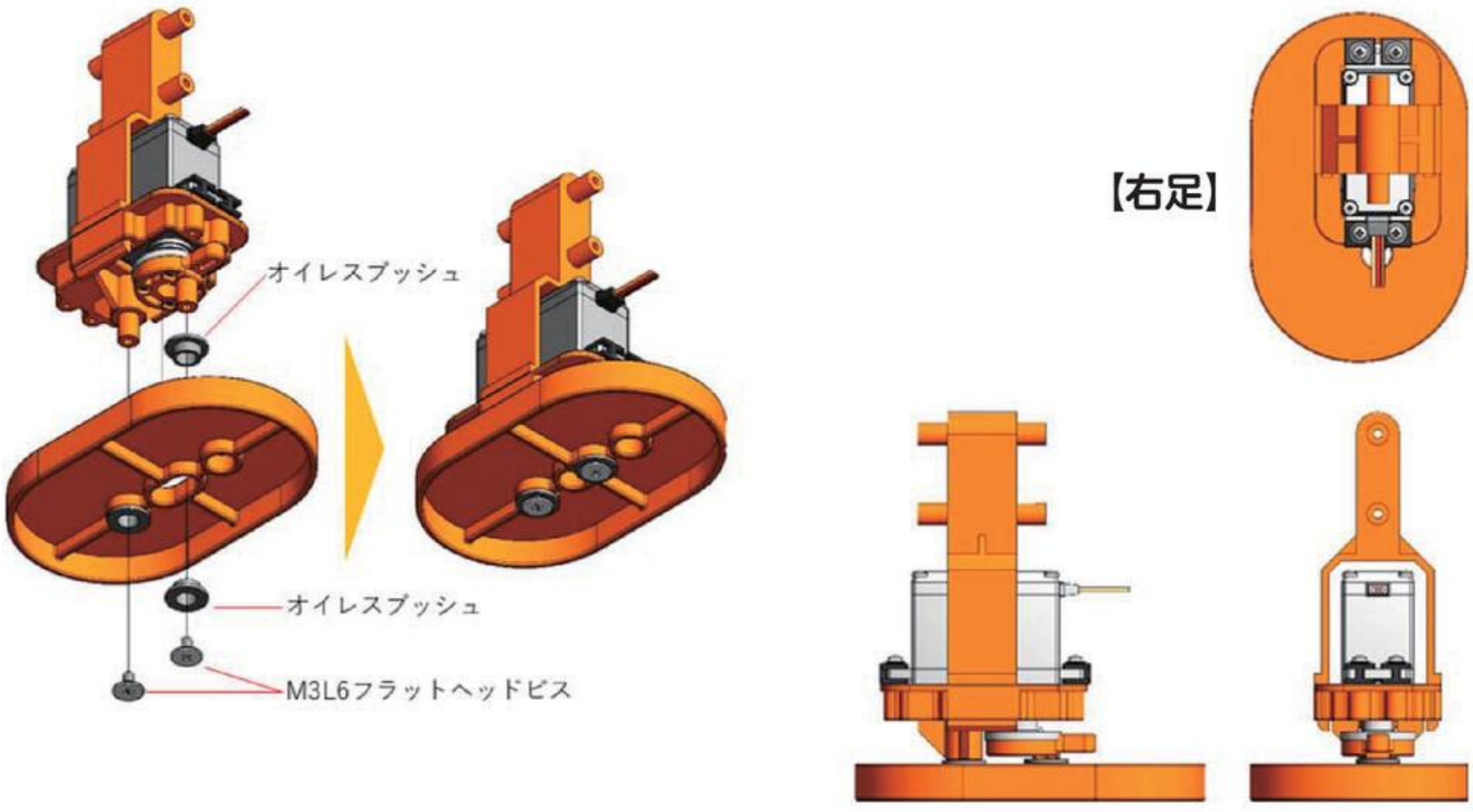


図1-3　C-4パーツの組み付け（右足）

＜組み立て手順④＞

図 1-3 で組み立てたのと同じように、オイレスブッシュ（× 2）、M3L6 フラットヘッドビス（× 2）を使って、C-4 パーツを組み付けましょう。図のようにオイレスブッシュは C-4 パーツの楕円形の穴に両面から挟み込みように取り付けて、M3L6 フラットヘッドビスで C-1 パーツに組み付けてください。

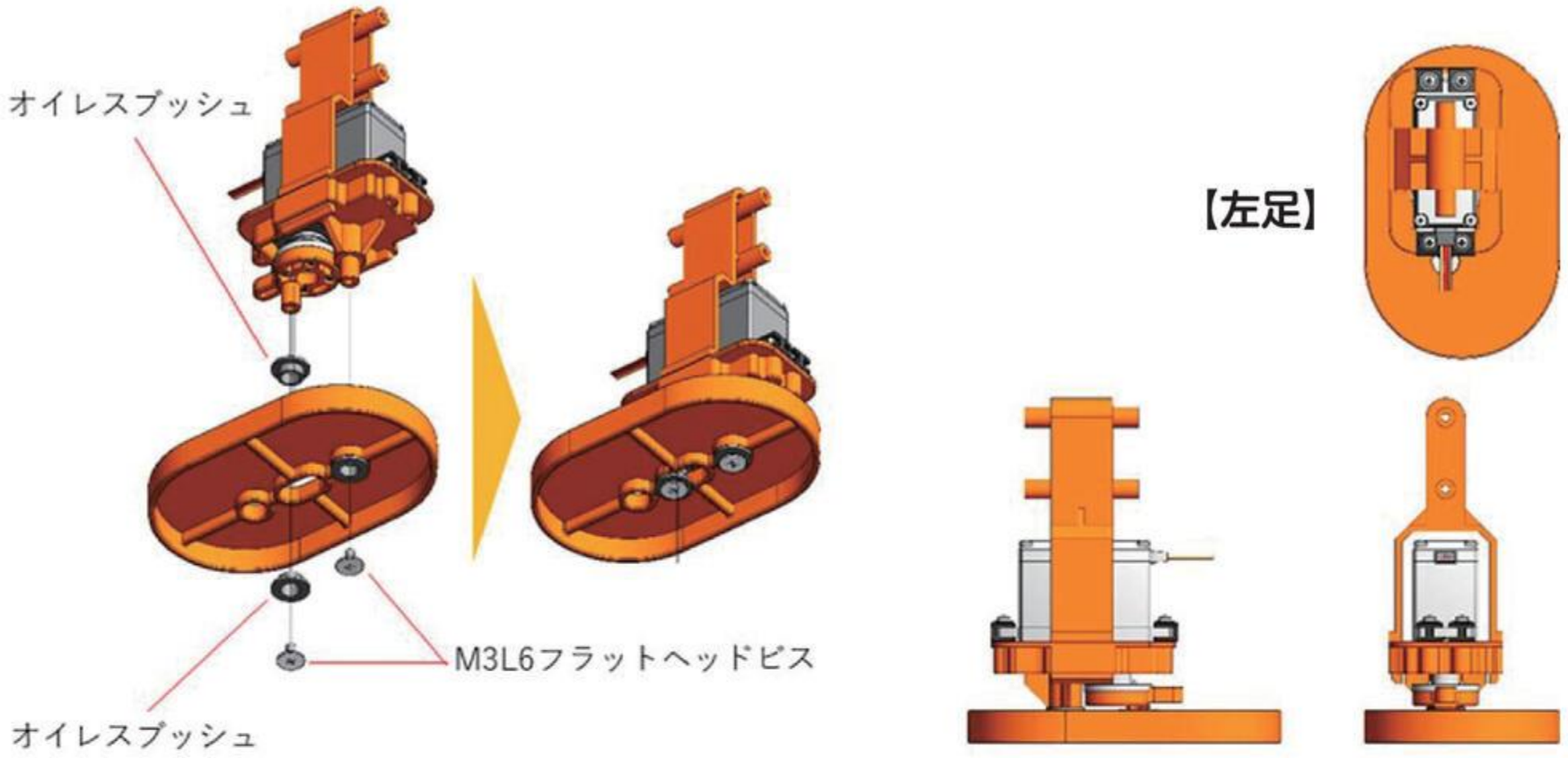


図1-4　C-4パーツの組み付け（左足）

＜組み立て手順⑤＞

次に、両足を連結させるために使用する B-3 パーツにオイレスブッシュ（× 4）を図のように取り付けてください。B-3 パーツとオイレスブッシュに隙間ができないようにしっかり取り付けてください。

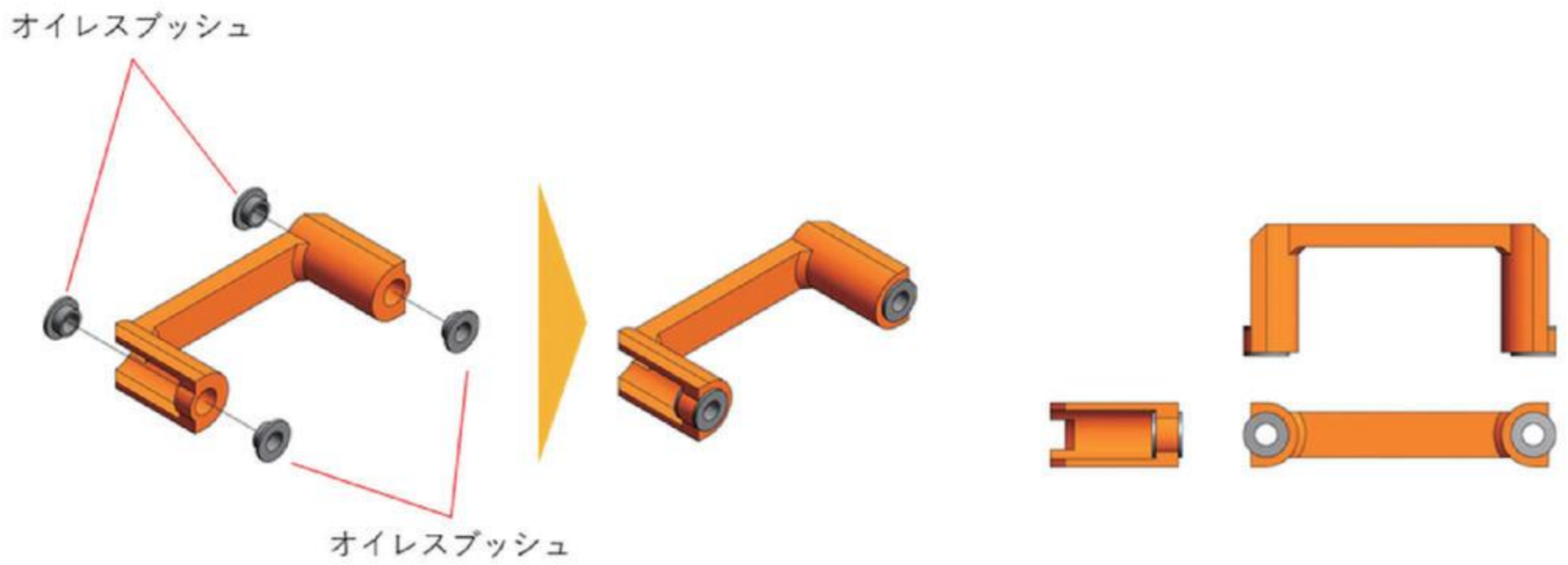


図1-5　足リンクの組み立て（B-3パーツ）

＜組み立て手順⑥＞

次に、両足を動かすために使用する B-4 パーツにオイレスブッシュ（× 4）を取り付けてください。B-4 パーツとオイレスブッシュに隙間ができないようにしっかり取り付けてください。同じものを二組つくりましょう。

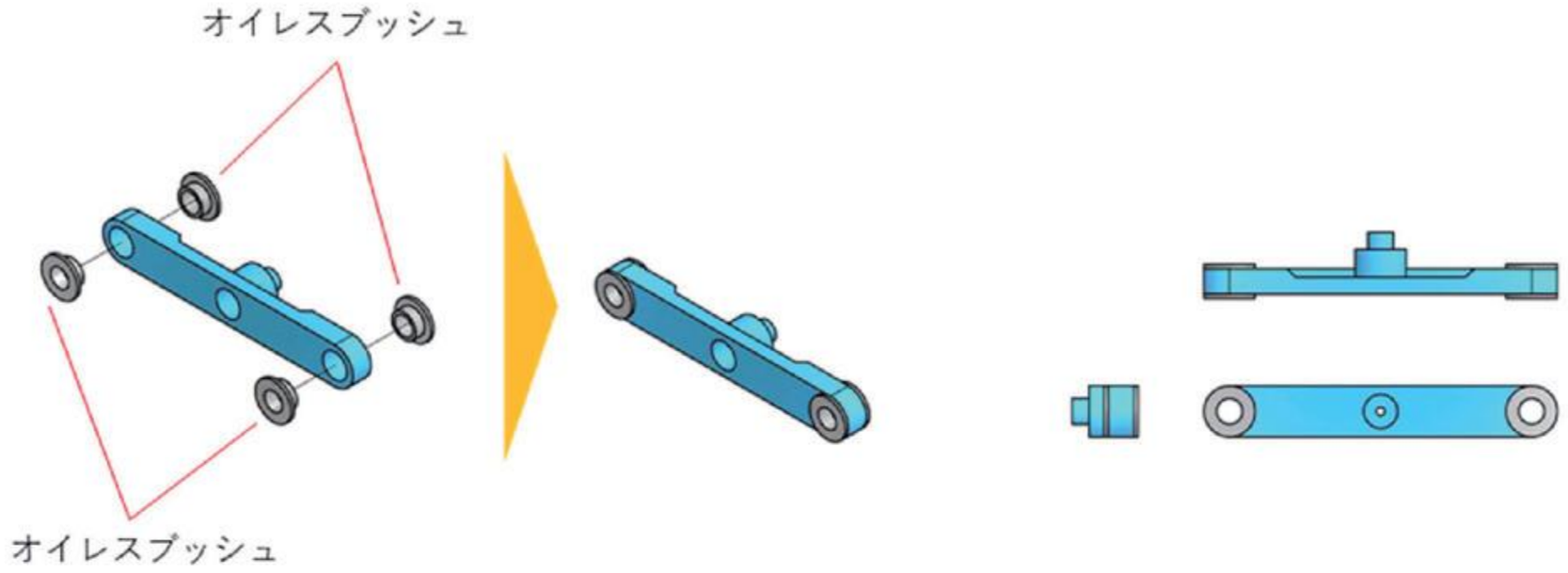


図1-6　足リンクの組み立て（B-4パーツ）

＜組み立て手順⑦＞

次に、左右の足を**図 1-5**、**図 1-6** の B-3 パーツと B-4 パーツで結合しましょう。C-3 パーツの凸型に出ている部分の下段に B-4 パーツを、上段に B-3 パーツを取り付け、図のように M3L6 フラットヘッドビス（× 4）で組み付けてください。

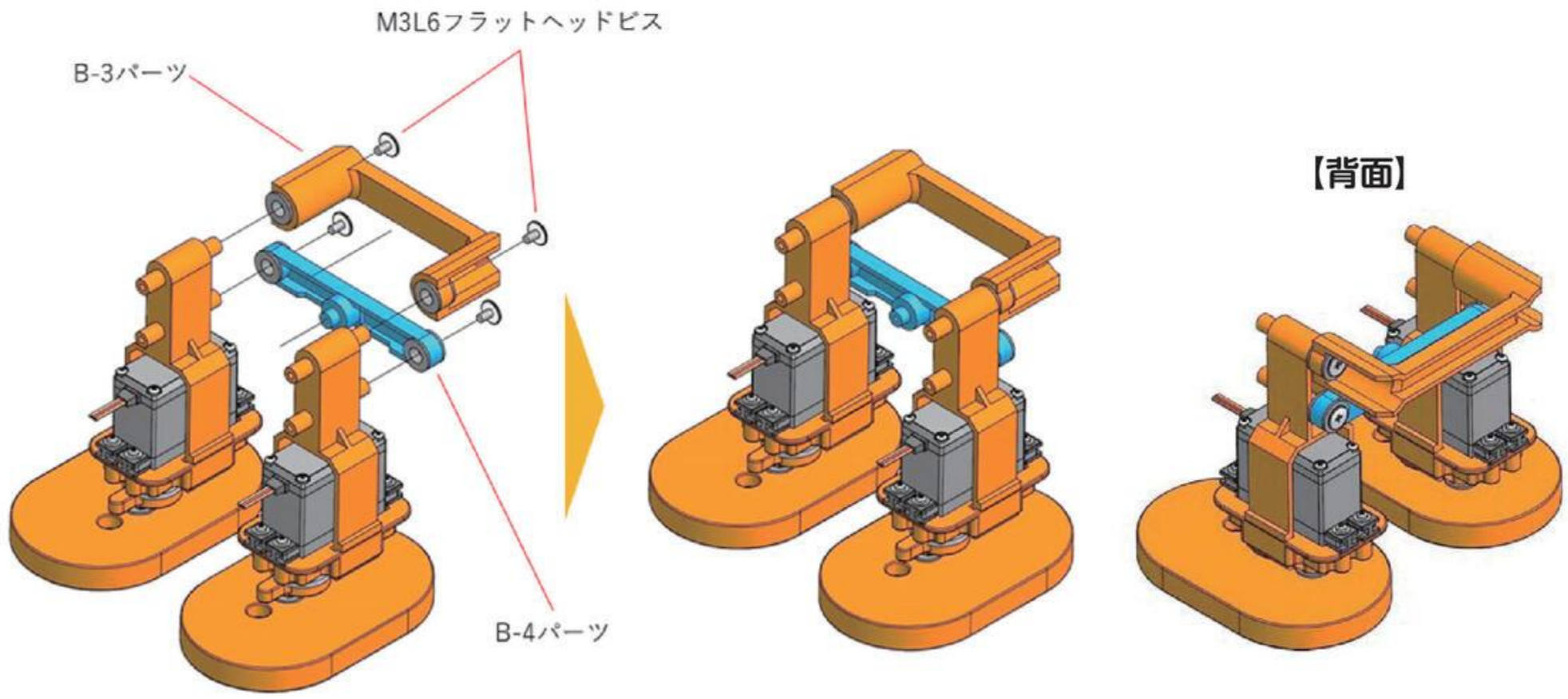


図1-7　足リンクの取り付け

＜組み立て手順⑧＞

次に、サーボモーターを取り付けたA-2パーツの丸穴の両面からオイレスブッシュ（×2）を取り付け、図のようにB-4パーツの凸型の部分に取り付けてください。さらに、図のようにもう一つのB-4パーツをC-3パーツの凸型の部分、A-2パーツの丸穴に嵌め込み、M3L6フラットヘッドビス（×2）で組み付けてください。

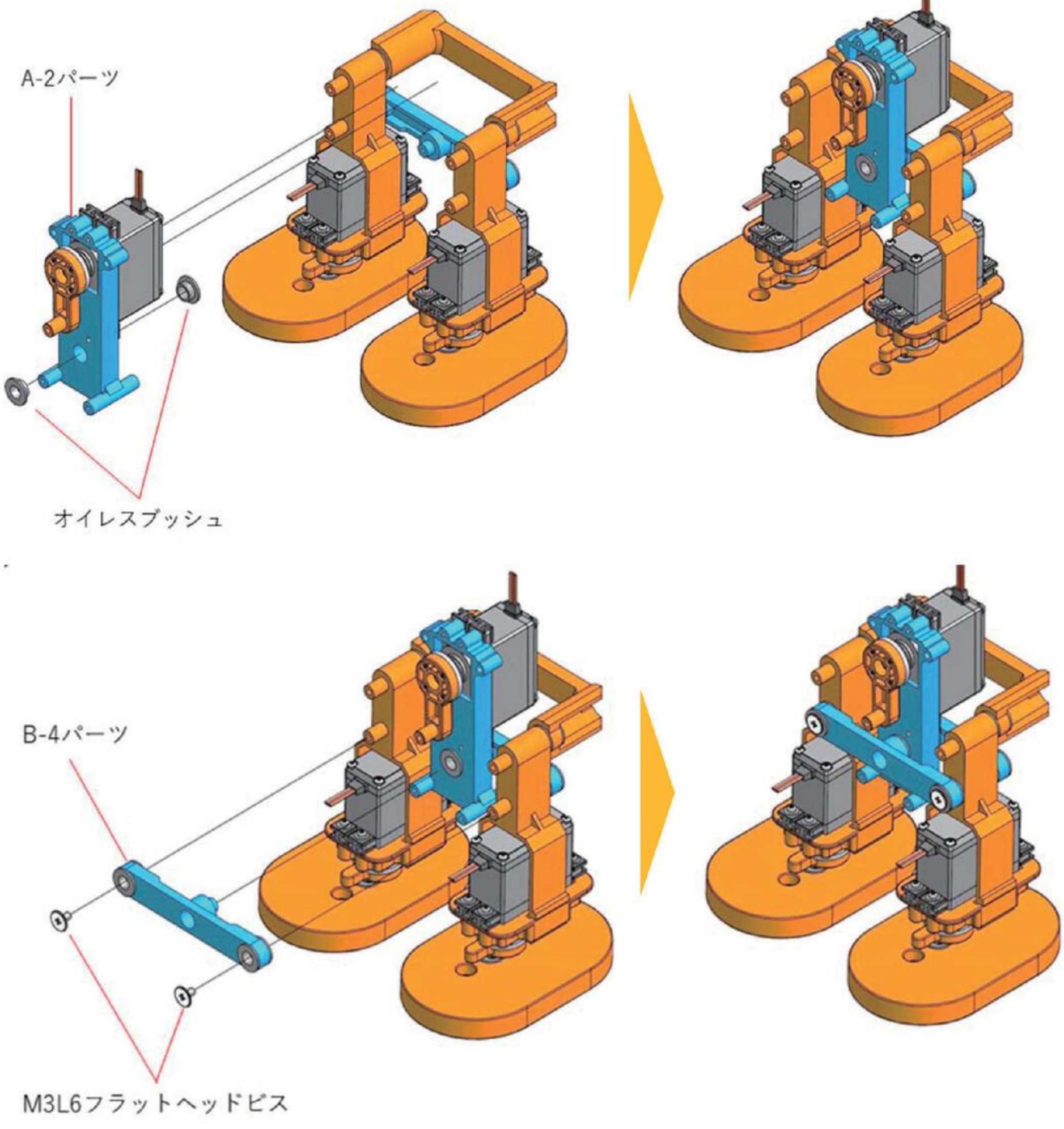


図1-8　足部サーボモーターの取り付け

＜組み立て手順⑨＞

次に、サーボモーターの動力を伝える役割をするリンクパーツを組み立てましょう。図のように B-2 パーツにオイレスブッシュ（× 4）を取り付けてください。

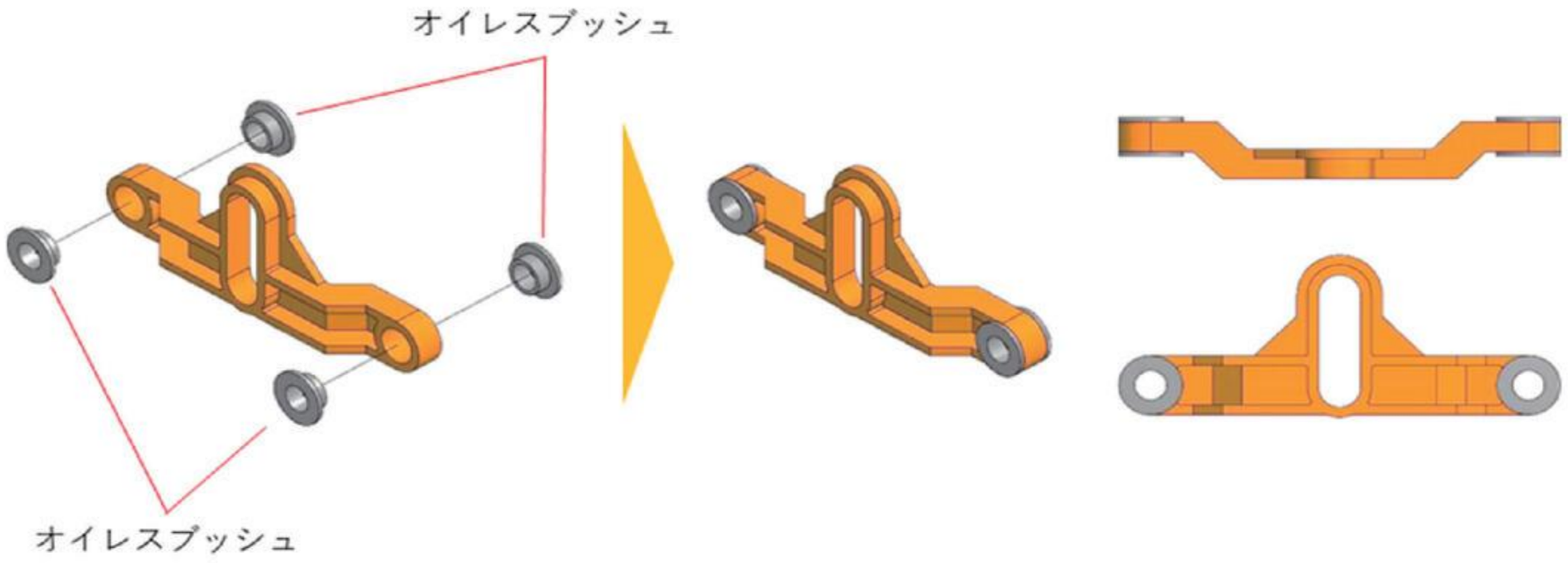


図1-9　足リンクの組み立て（B-2）

＜組み立て手順⑩＞

次に、B-2 パーツの楕円形の穴の両面からオイレスブッシュ（× 2）を取り付け、図のように C-3 パーツの凸型の部分、B-1 パーツの凸型の部分に嵌め込み、オイレスブッシュの上から M3L6 フラットヘッドビス（× 3）で組み付けて下さい。

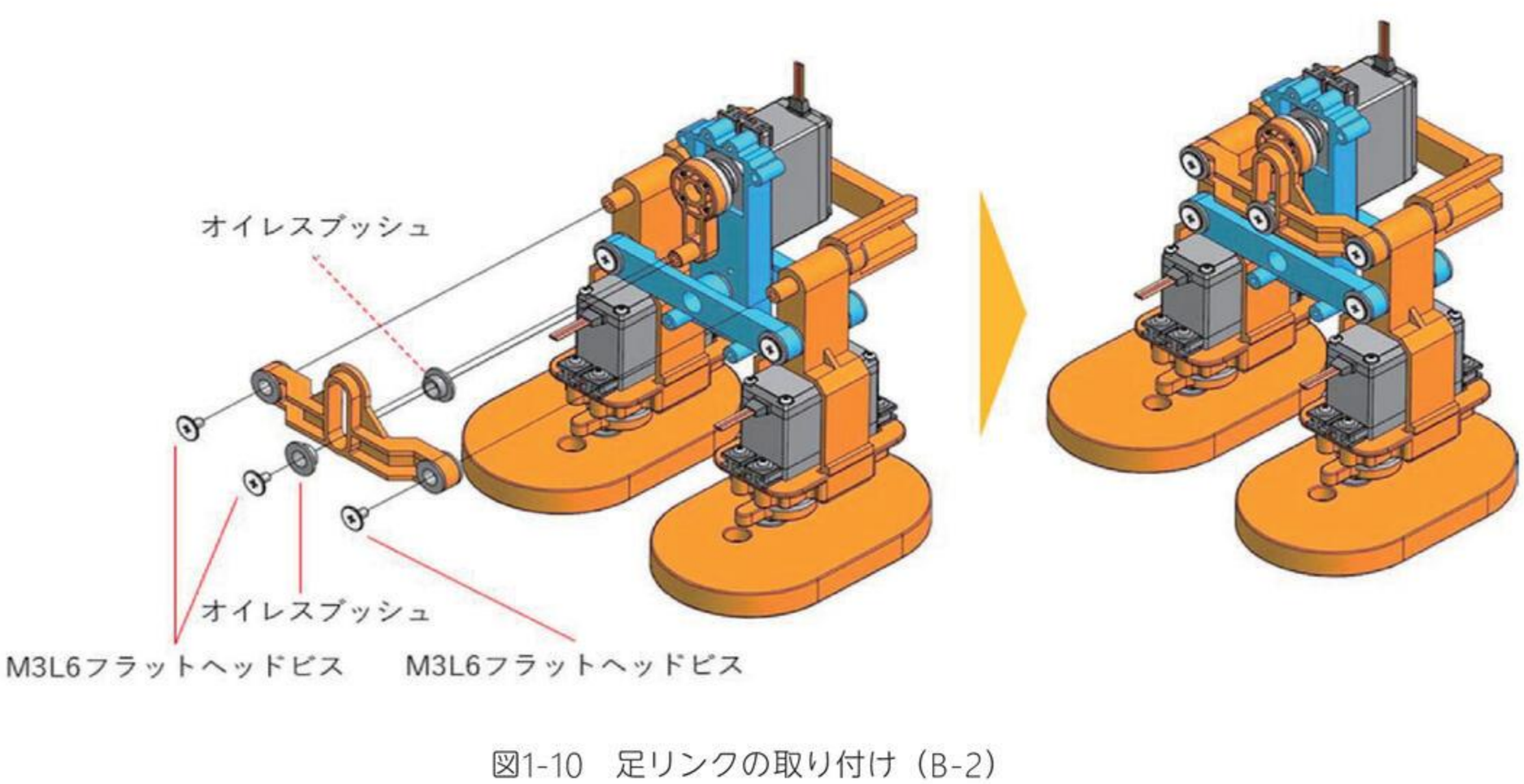


図1-10　足リンクの取り付け（B-2）

講　B-2 パーツはご購入頂いた時期によって青色のものもあります。
形状は同じであるため、作業に問題はありません。

1.1.　ボディの組み立てと配線

＜組み立て手順①＞

次に、マイコンボード、ロボプロシールドのユニット（ボディ部）に、マトリクス LED シールド、マトリクス LED を取り付け、足部と連結しましょう。ボディ部の A-1 パーツを足部の A-2 パーツのネジ穴に M3L10 タッピングネジ（B）（× 2）を使って図のように組み付けてください。

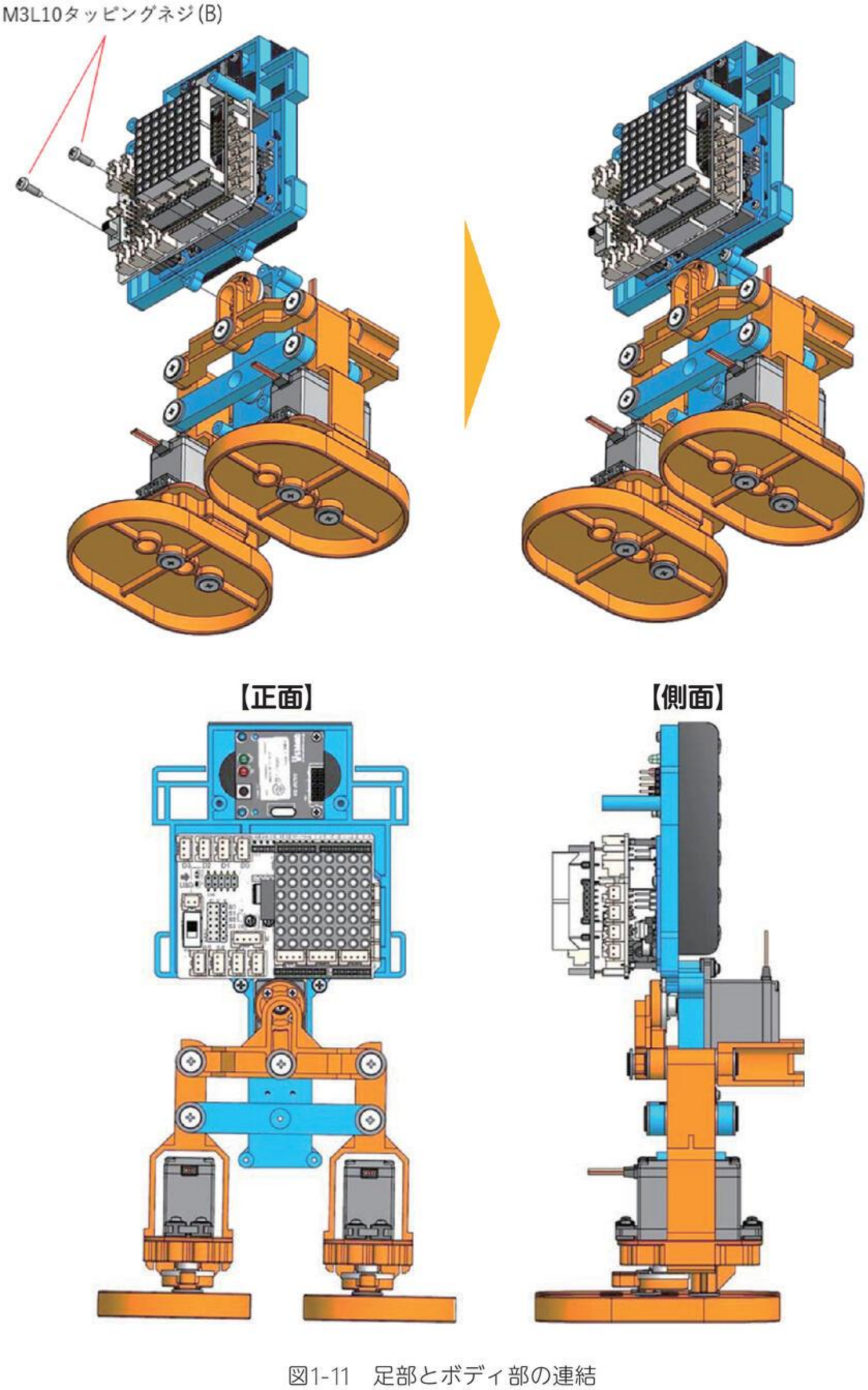


図1-11　足部とボディ部の連結

＜組み立て手順②＞

次に、足の付け根と足首に位置するサーボモーターの配線をしましょう。
写真のように、足首のサーボモーターケーブルはC-3パーツとサーボモーターの隙間から背面の方向に通してから、さらにA-1パーツのフックを通し正面に取りまとめてください。
写真は右足ですが、左足や足の付け根のサーボモーターケーブルも同じように取りまとめてください。

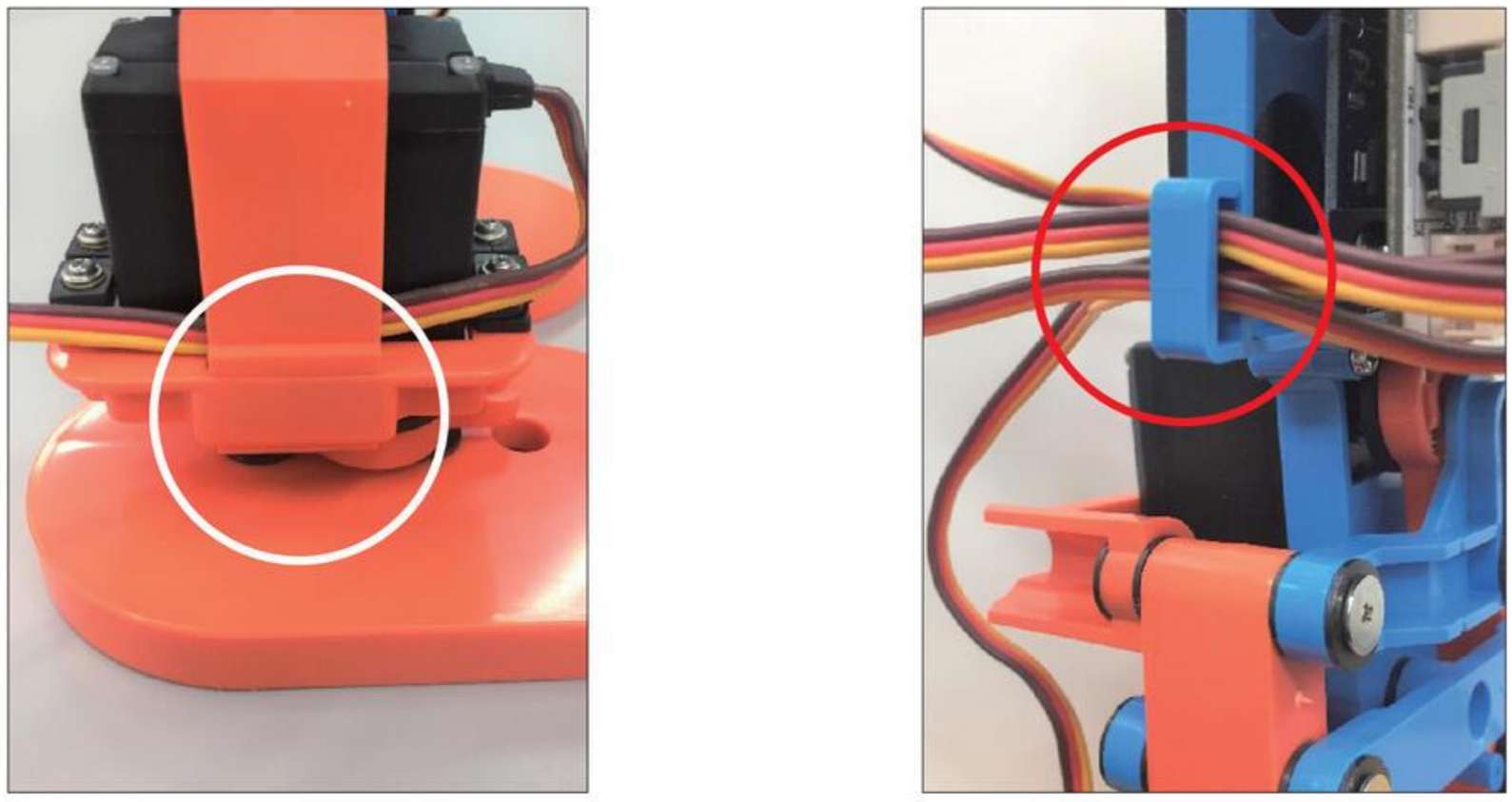


図1-12　サーボモーターケーブルの取りまとめ

＜組み立て手順③＞

次に、サーボモーターをロボプロシールドに接続しましょう。以下のように接続してください。

・左足首のサーボモーター　　ロボプロシールドの S0 端子
・右足首のサーボモーター　　ロボプロシールドの S1 端子
・足の付け根のサーボモーター　ロボプロシールドの S2 端子

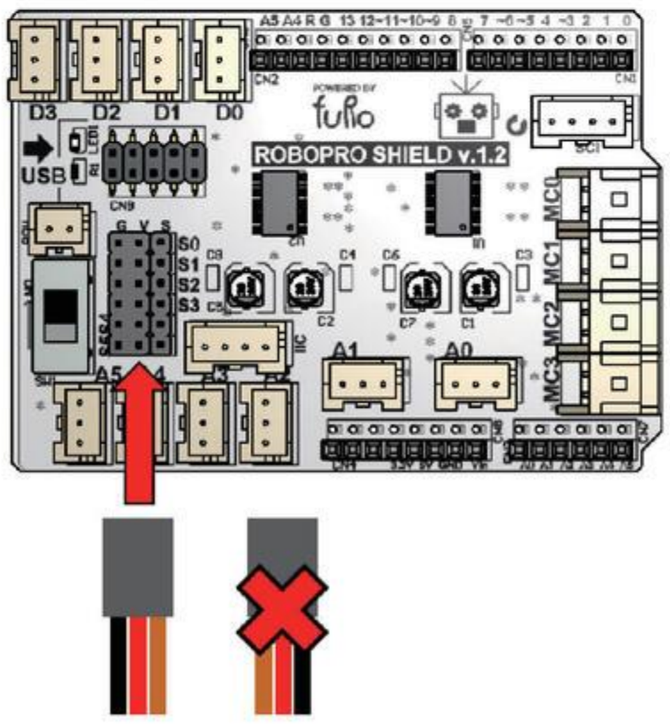


図1-13　サーボモーターコネクター接続の向き

＜組み立て手順④＞

次に、マトリクスLEDシールドのUS2にセンサーケーブルを接続して、図のようにA-1パーツの穴から背面へセンサーケーブルを通してください（センサーケーブルを接続する超音波距離(きょり)センサーは次回接続します）。

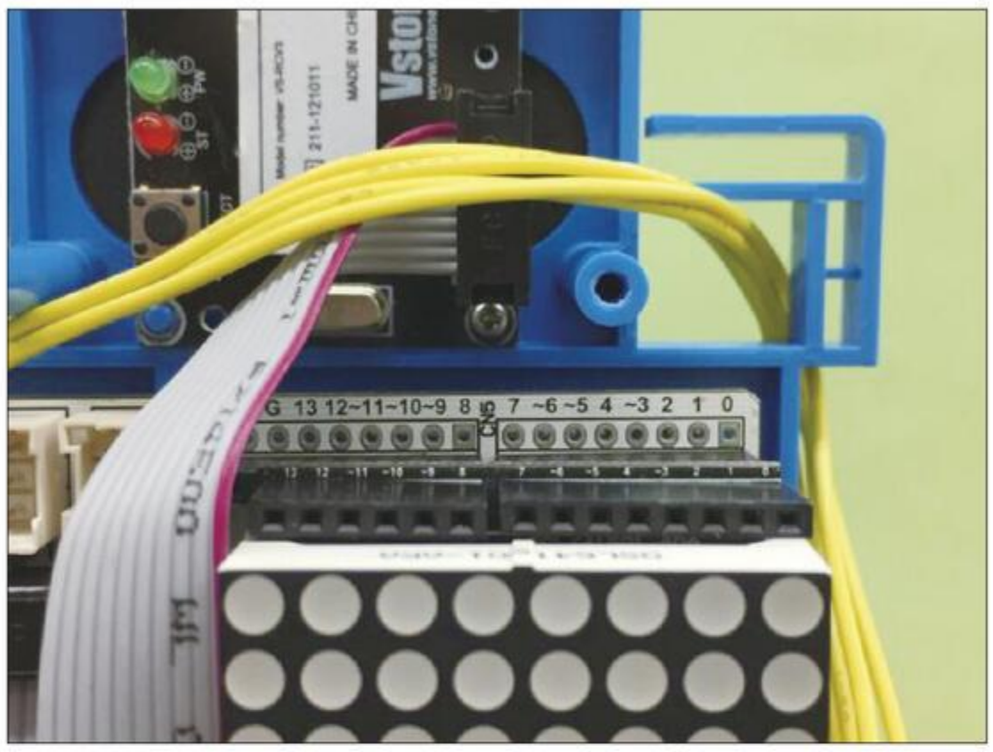


図1-14　センサーケーブルの接続

以上が今回の組み立て作業です。残ったパーツは次回まで大切に保管してください。
残りの時間は、「サーボモーターの制御」と「歩行パターン」について学びます。
複数のサーボモーターを同時に動かすためには大きな電力を持続的に供給する必要があります。
電池では電池残量により電圧値が下がると出力が低下しますので、以降は、ACアダプターとAC変換ケーブルを使用します。

図1-15　ACアダプターとAC変換ケーブル

2.　サーボモーターの制御（目安 40 分）

2.0.　サーボモーターのプログラム

第1回では、サーボモーターの原点位置合わせで2種類のプログラムを実行しました。復習の意味も含めてサーボモーターの仕様と、「ServoOrigin0/1」のプログラムに使われている関数などをまとめます。第1回で使われたプログラムは以下の表のようになります。

表2-0　第1回で使ったサンプルプログラム

サンプルプログラム	プログラム内容
BiRobot1　＞　ServoOrigin0	S0端子に接続したサーボモーターの原点位置を90度に設定する。
BiRobot1　＞　ServoOrigin1	S0端子に接続したサーボモーターの原点位置を90度に設定して、オプションとしてパルス幅の値を調整して暴走防止の処理を加える。

ライブラリファイル「Servo.h」「Servo.cpp」で制御されている内容例と関数は以下の通りです。

POINT

サーボモーターの設定

・0 ～ 180度の範囲でプログラムにより出力軸（シャフト）の角度指令ができる。
・サーボモーターが0度のときに与えられるパルス幅は、標準544（マイクロ秒）。
・サーボモーターが180度のときに与えられるパルス幅は、標準2400（マイクロ秒）。

表2-1　サーボモーターの関数

サンプルプログラム	プログラム内容
attach(int pin) 例）servo0.attach(S0);	サーボモーターをpinのピンに接続し使う。 例）servo0と名付けたモーターをS0に接続し使う。
attach(int pin, int min, int max) 例）servo0.attach(S0, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH - 0);	サーボモーターをpinのピンに接続し、パルス幅をmin～maxの範囲に限定して使う。 例）servo0と名付けたモーターをS0に接続し、パルス幅を最小MIN_PULSE_WIDTH（544マイクロ秒）から最大MAX_PULSE_WIDTH - 0（2400マイクロ秒）に限定して使う。
write(int value) 例）servo0.write(90);	サーボモーターの出力軸の角度を指定する。 例）servo0と名付けたモーターの出力軸を90度に回転させる。

2.1.　サーボモーターの歩行制御プログラム

続いて、3 個のサーボモーターをコントローラーで操作するプログラムを実行します。
プログラムを書き込む前に、AC アダプターと AC 変換ケーブルを**図 2-0** のように接続します。

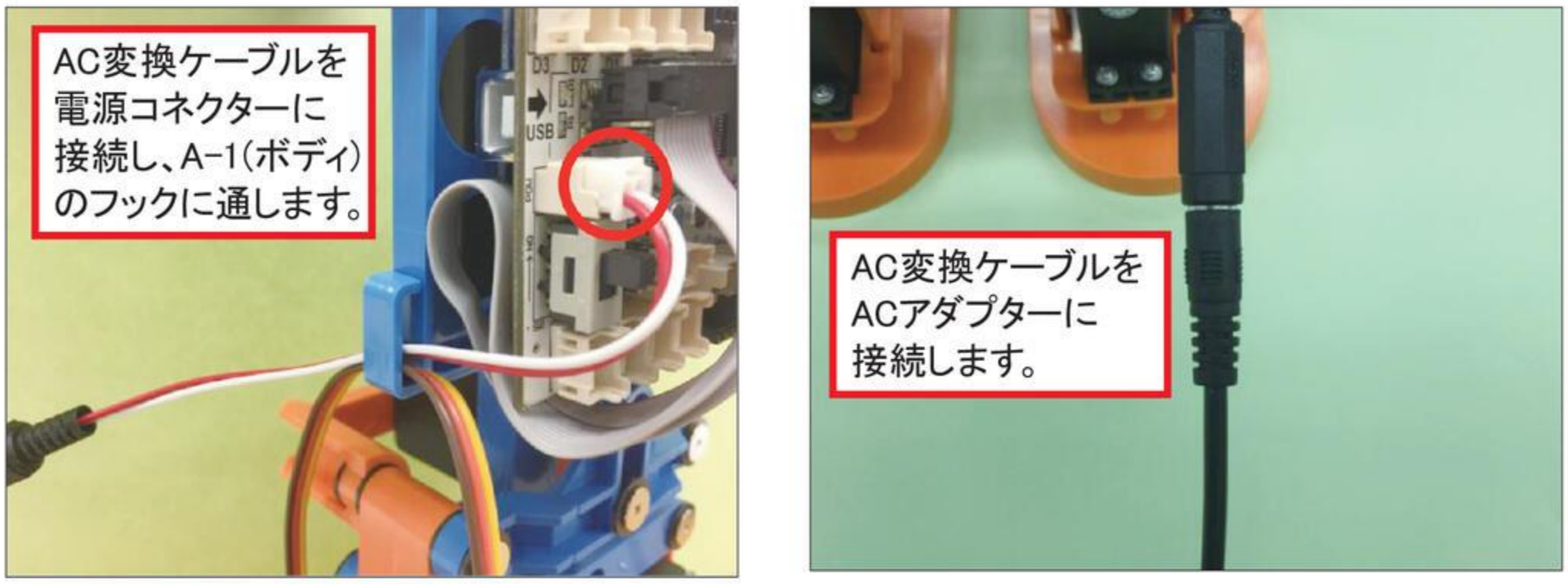


図2-0　AC変換ケーブルの接続

ACアダプターを接続したときに、ロボプロシールドの電源ランプが点灯することを確認したら、一度電源をOFFにして、以下のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse3　>　BiRobot2　>　ServoOrigin2

実行結果：左右のアナログスティックを上下すると左右足、L2・R2ボタンを押すと脚付け根のモーターがそれぞれ回転する。
今回からコントローラーを使用して二足歩行ロボットを「操縦」することを目標にしていくので、まずは「ServoOrigin0/1」から次のような変更をしています。

POINT

- 計3つのモーターを操作できるようにするため、コントローラーの読み取る部分を増やす。
- サーボモーターの消費電力を抑えるため、出力軸が一気に回転しすぎないようにする。

コントローラーとサーボモーターを使うことじたいは変わっていないので、ライブラリファイルは「ServoOrigin0/1」と同じものを取り込みます。

プログラム「ServoOrigin2」より抜粋

```
#include <RPlib.h>     //ロボプロシールドを使うためのライブラリ（設定ファイル）
#include <Servo.h>     //サーボモーターを使うためのライブラリ（設定ファイル）
#include <PS2X_lib.h> //コントローラーを使うためのライブラリ（設定ファイル）
```

#define で定数に対して名前をつけています。

プログラム「ServoOrigin2」より抜粋

```
#define    BASE_POSITION_LF        90 //BASE_POSITION_LF：左足の原点位置
#define    BASE_POSITION_RF        90 //BASE_POSITION_RF：右足の原点位置
#define    BASE_POSITION_BD        90 //BASE_POSITION_BD：ボディの原点位置
#define    DELAY_TIME             300
#define    SERVO_PERIOD            20
```

急激な動作を防ぐプログラムは、2年目のアームロボットでも「出力軸が、loop()を1回実行するごとに1度ずつしか回転しないようにする」という形で登場していました。今回も同様の処理を追加しています。

プログラム「ServoOrigin2」より抜粋

```
void  servo_update(void){
    if (currentLF < targetLF){ //もしservoLFの現在位置が目標位置より小さかったら
        currentLF++;           //現在値を+
    }
    else
    if (currentLF > targetLF)  //もしservoLFの現在位置が目標位置より大きかったら
    {
        currentLF--;           //現在値を-
    }

(中略)

    servoLF.write(currentLF);  //増減させた新しい現在位置をservoLFに反映
(中略)
}

void  PS2X_update(void){
    ps2x.read_gamepad();       //コントローラーから値を読み取る

    targetLF = ps2x.Analog(PSS_LY) * 5 / 8 + 90;
```

currentLFとtargetLFという2つの変数で左足モーターの回転を制御しています。アナログスティックの値に応じてtargetLFを変化させ、currentLFがtargetLFと同じ値になるまでモーターを回転させているわけです。currentLFが1ループにつき1ずつしか増減しないので、モーターも1度ずつしか回転していきません。ps2x.Analog(PSS_LY) * 5 / 8 + 90;とやや複雑な式になっているのは、−128～+128だったアナログ値を0～180度の角度指令値に直す処理です。前回より少ない手順で値を変換できていますね。

ただし、計算してみるとわかりますが、この式だと角度指令値の最小・最大は0や180にはなりません。10～170度の範囲におさめてゆとりをもたせているわけですね。
さて、メインループを見てみましょう。

プログラム「ServoOrigin2」より抜粋

```
void loop(){
    if (( counter % 5 ) == 0){
        PS2X_update();
    }

    servo_update();

    delay(SERVO_PERIOD);
    counter++;
}
```

なんと処理が４つしかありません。そのうえ、複雑な処理は関数でまとめられています。ロボプロで扱ってきたプログラムのなかでも、トップクラスにシンプルなメインループではないでしょうか。

まとめられている関数のうち、`PS2X_update();`はコントローラーの状況をみてモーターの目標位置を設定する処理、`servo_update();`は目標位置になるようモーターを回転させる処理を担当しています。つまり、`targetLF`は`PS2X_update();`で、`currentLF`は`servo_update();`で変更しているということになります。

`PS2X_update();`だけ if 文に含まれていますが、その条件式は`counter % 5`です。`%`はこれまでにも何度か登場しましたが、わり算をしたときの「余り」を求めるための記号です。もし変数`counter`が 13 であれば、13 ÷ 5 ＝ 2 あまり 3 なので、`counter % 5`は「3」となります。変数`counter`が 18 になっても、やはり 5 で割ればあまりは 3 なので`counter % 5`は「3」となります。

ステップアップ

プログラム「ServoOrigin2」では、`if (( counter % 5 ) == 0)`を満たすときのみ`PS2X_update();`を実行しているね。つまり、`PS2X_update();`をどのようなタイミングで実行したいということかな？

ループ5回につき1回だけ実行したい。

今回のメインループは`SERVO_PERIOD`、つまり20ミリ秒ごとに1回実行されますが、1秒間にボタンを50連打するようなペースでもない限り、コントローラーのボタンの状況はそこまで短時間では変化しません。そのため、ボタン状況の読み取りはループ5回ごと、つまり100ミリ秒ごとに行う程度で十分なわけです。

コラム　モジュロを活用する

わり算したときの余りを求めることを「剰余演算」とよびますが、これを英語では「モジュロ（modulo）」といいます。もし「13を5で割ったときのあまりを求める式」をつくるなら、Arduinoでは % という記号（剰余演算子といいます）を使い 13 % 5 とかきますが、数学の世界では「13 mod 5」とかくこともできます。ちなみに「13を5で割ったときの余りは3」というのを「13 ≡ 3 (mod 5)」とかくこともあります。

もし近くにカレンダーがあれば見てみるとわかりやすいですが、日付の値を7で割ったとき、曜日が同じ日どうしは余りが等しくなりますよね。これも剰余演算を使った例と言えます。ちなみに数学の世界には「関ヶ原の戦いがあった日は何曜日か」「自分が20歳の誕生日を迎えるのは何曜日か」といった過去・未来の曜日を求めるための公式がありますが、この公式の中にもモジュロが使われています。「ツェラーの公式」とよばれるものなので、興味がある人は調べてみるのもよいでしょう。

2.2.　サーボモーターの調整方法

続いては、微調整のやり方を確認します。
以下のプログラムで定義された３つの値で微調整できます。適切な値は、個々のロボットで微妙に違うので、ほかの人のロボットの値をコピーしても意味がありません。

プログラム「ServoOrigin2」より抜粋

```
#define    BASE_POSITION_LF        90 //BASE_POSITION_LF：左足の原点位置
#define    BASE_POSITION_RF        90 //BASE_POSITION_RF：右足の原点位置
#define    BASE_POSITION_BD        90 //BASE_POSITION_BD：ボディの原点位置
```

ロボットを後ろから見て体が右に傾いている場合は、BASE_POSITION_BD の値を小さく、左に傾いている場合は、BASE_POSITION_BD の値を大きくします。足先が右を向いている場合は、BASE_POSITION_LF、BASE_POSITION_RF の値を小さく、左を向いている場合は、BASE_POSITION_LF 、BASE_POSITION_RF の値を大きくします。
なお、今回はボディ以外はすぐに値が戻るようにプログラムが組まれているため、あまり意味がありませんが、今後のために上の調整方法を覚えておきましょう。

2.3.　調整値のヘッダーファイル化

さらにカシコイ調整値の引き継ぎのやり方も教えます。まず、以下のプログラムを開いてください（まだ書き込みはしません）。

プログラムの書き込み

RoboticsProfessorCourse3　>　BiRobot2　>　ServoOrigin3

このプログラムを開くと、プログラム本体である「ServoOrigin3」のタブの横に、もう一つ、AdjustParam.h というタブ（**図2-1**の赤丸の部分）が表示されます。
「Arduino IDE」では、プログラムと同じフォルダに入っているファイルは自動的に別タブで展開され、タブを選択するとファイルの中身を確認することができます。
プログラム「ServoOrigin3」は、プログラム「ServoOrigin2」では #define されているパラメータを、ヘッダーファイルとして分離したものです。さきほどの３つの数字、BASE_POSITION_LF、BASE_POSITION_RF、BASE_POSITION_BD の値はここに入力できます。以降のプログラムについては、この調整値をかいた AdjustParam.h のヘッダーファイル内で補正することで、調整値を引き継がせることができるようになるわけです。やり方の一つとして知っておきましょう。

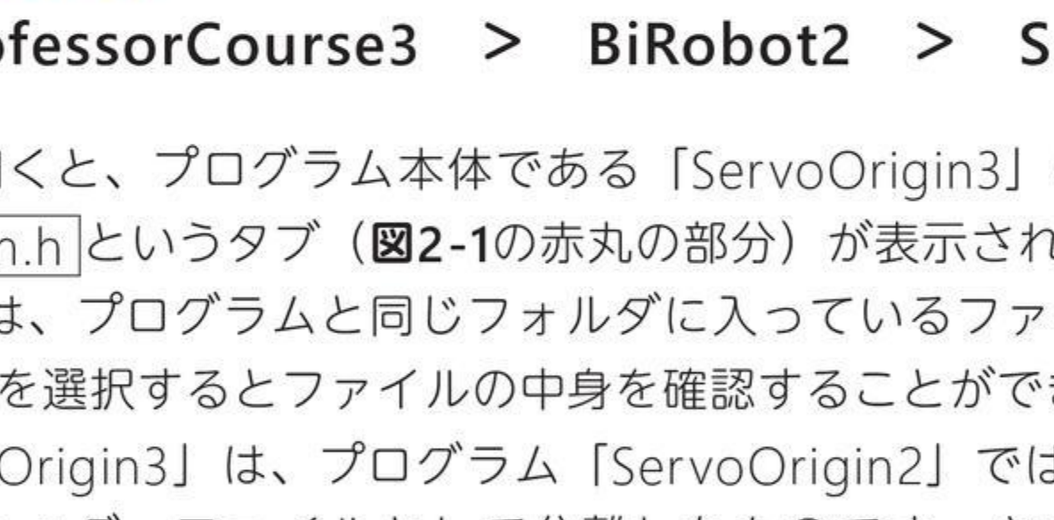


図2-1　プログラム「ServoOrigin3」

3.　歩行システム（目安 20 分）

3.0.　二足歩行ロボットを動かす

二足歩行ロボットの完成は次回になりますが、現時点でも足を動かして歩くことが可能です。ここでは、二足歩行ロボットの歩行システムを理解するために、コントローラーでサーボモーターを動かし、二足歩行ロボットの歩行システムを習得します。
コントローラーの L2/R2 ボタンを押すと、ボディのサーボモーターがゆっくりと動いて身体を傾け、僅(わず)かに左右のどちらかの足が持ち上がります。とてもゆっくりとした動きですが、この動きを速くしすぎると、勢い（慣性(かんせい)）がついてそのまま横方向に転倒してしまいます。以後、プログラムをアレンジするときは注意して行ってください。
また、周囲に壊れやすいものを置いたり、二足歩行ロボットを机の端(はし)で動かすのは危険ですので、広い机の真中や床面を利用してください。

さて、身体を左右どちらかに倒した状態で、コントローラーの左右のアナログスティックを上下に動かすと、持ち上がっている足は足先が左右に回転し、逆の接地している足は、足先が動く代わりに、逆方向に体をねじる動きが発生します。二足歩行ロボットは、たった３個のサーボモーターの動きだけで、歩行動作をすることができます。

皆さんは、いつも無意識に二足歩行をしていると思います。ここで、二足歩行という動作についてあらためて考えてみましょう。
1本の足だけに注目すると、歩行中の足には二通りあることがわかります。一つは、地面に足裏を接地させて体を支えている立脚相(りっきゃくそう)、もう一つは、足裏が宙に浮いている遊脚相(ゆうきゃくそう)です。
二足歩行は、２本の足を交互に、立脚相(りっきゃくそう)と遊脚相(ゆうきゃくそう)をくり返すことで行います。
二足歩行ロボットでは、身体を左右に傾(かたむ)けることで、左右の足を立脚(りっきゃく)、遊脚(ゆうきゃく)させることができます。立脚(りっきゃく)している方の足を動かすことで、体をねじるようにして遊脚(ゆうきゃく)している足を前に出したり、後ろに出したりすることができます。

図3-0　二足歩行ロボット

チャレンジ課題

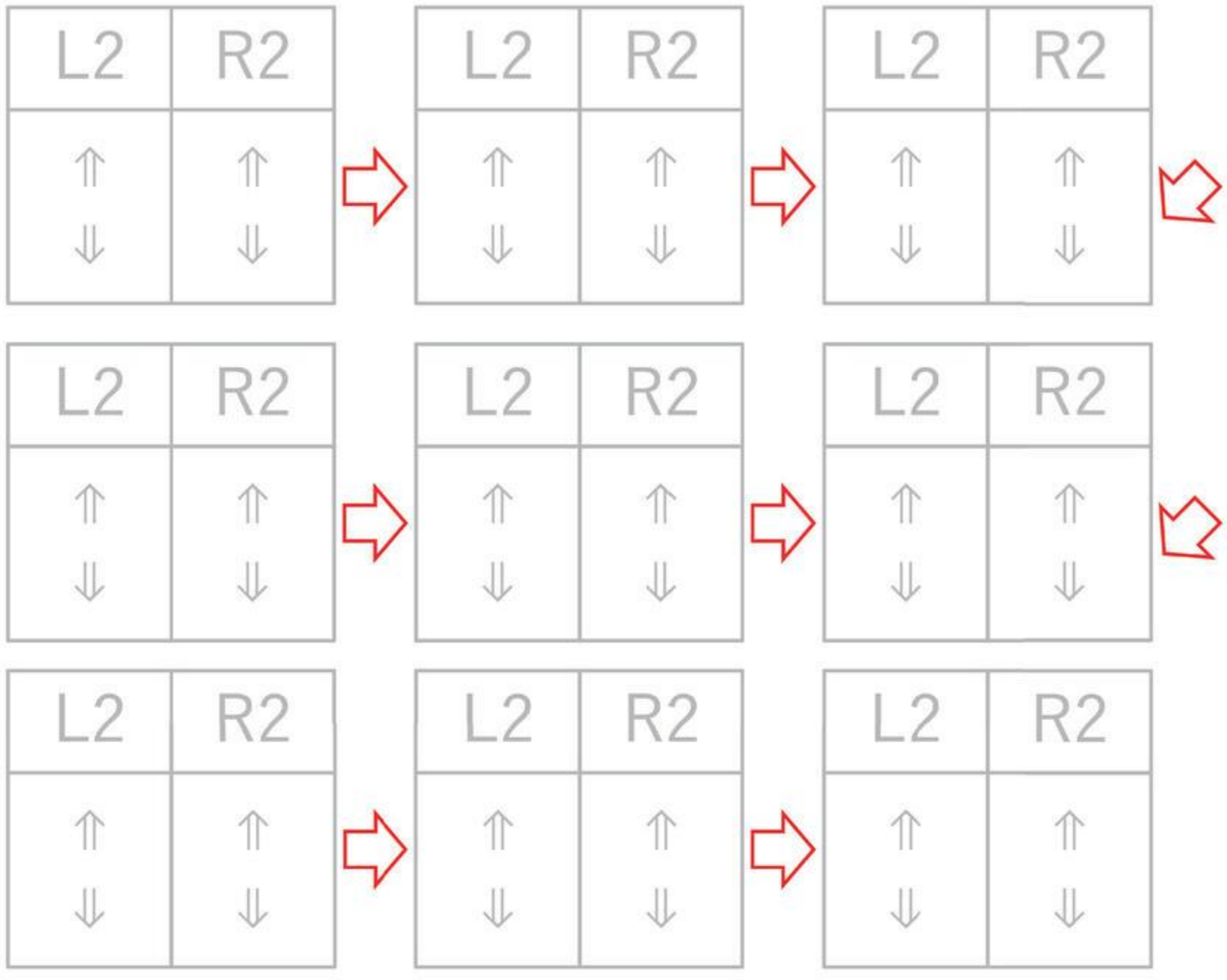
前のページの説明には、プログラム「ServoOrigin3」で二足歩行ロボットを前進させるためのヒントがかかれているよ。読み解いて、実際にロボットを前進させるためにはどのような操作が必要か確認してみよう！
手順がわかったら、次のページにメモしてみてね。できたら、後退、左右旋回（せんかい）の手順も同じように探してみよう！

講

どちらか片方の脚を浮かせた状態で、左右脚を同時に動かしてから足を下ろす、という動作をくり返すと移動できます。
左右アナログスティックを同じ方向に傾けるようにすると前進または後退、逆の方向に傾けるようにすると左右旋回になります。

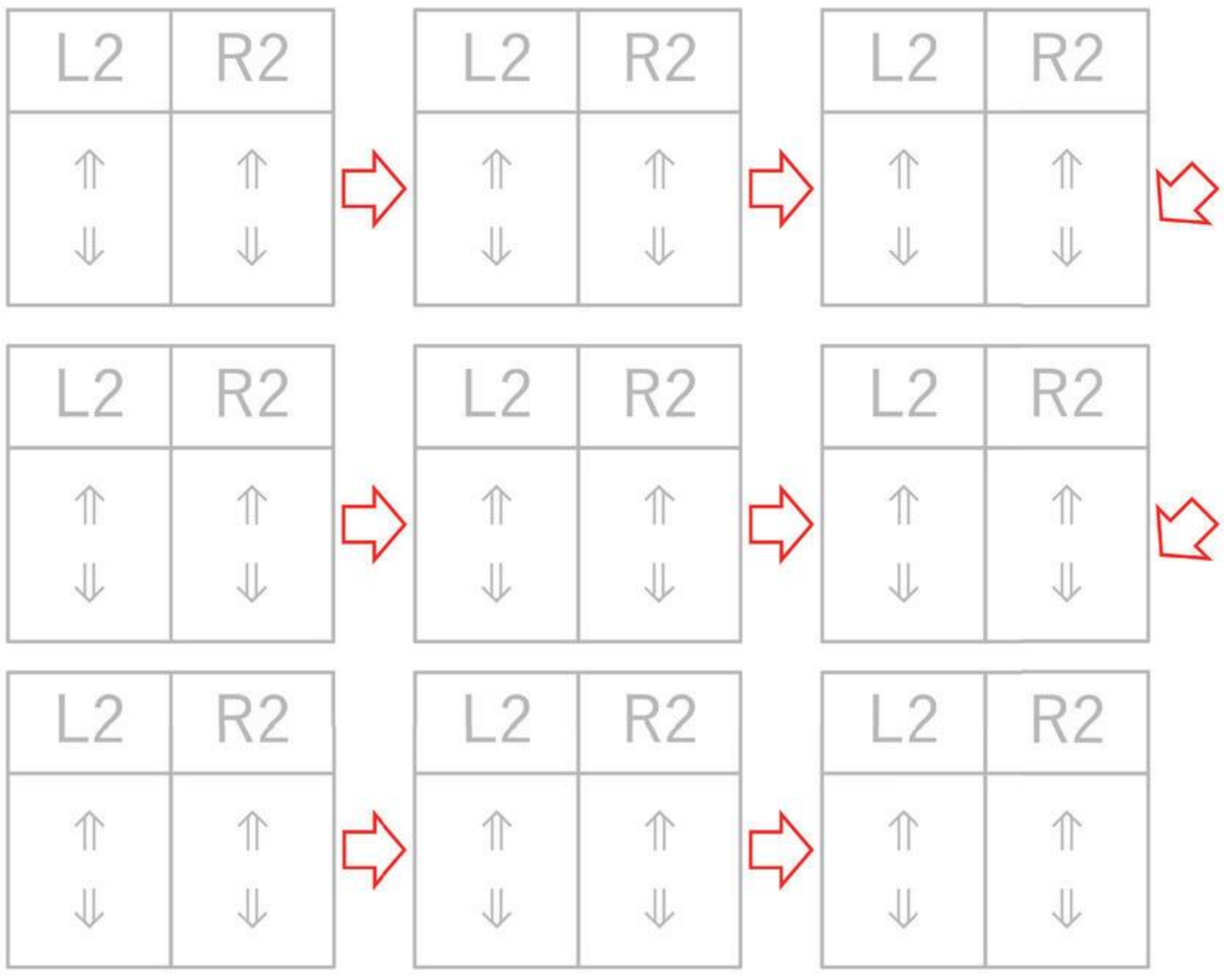
3.1. 前進の手動制御

コントローラーのボタンを操作して、どのような順番で足をどう動かすとスムーズに前進動作ができるのか、その順番と動きを見て記録してみましょう。



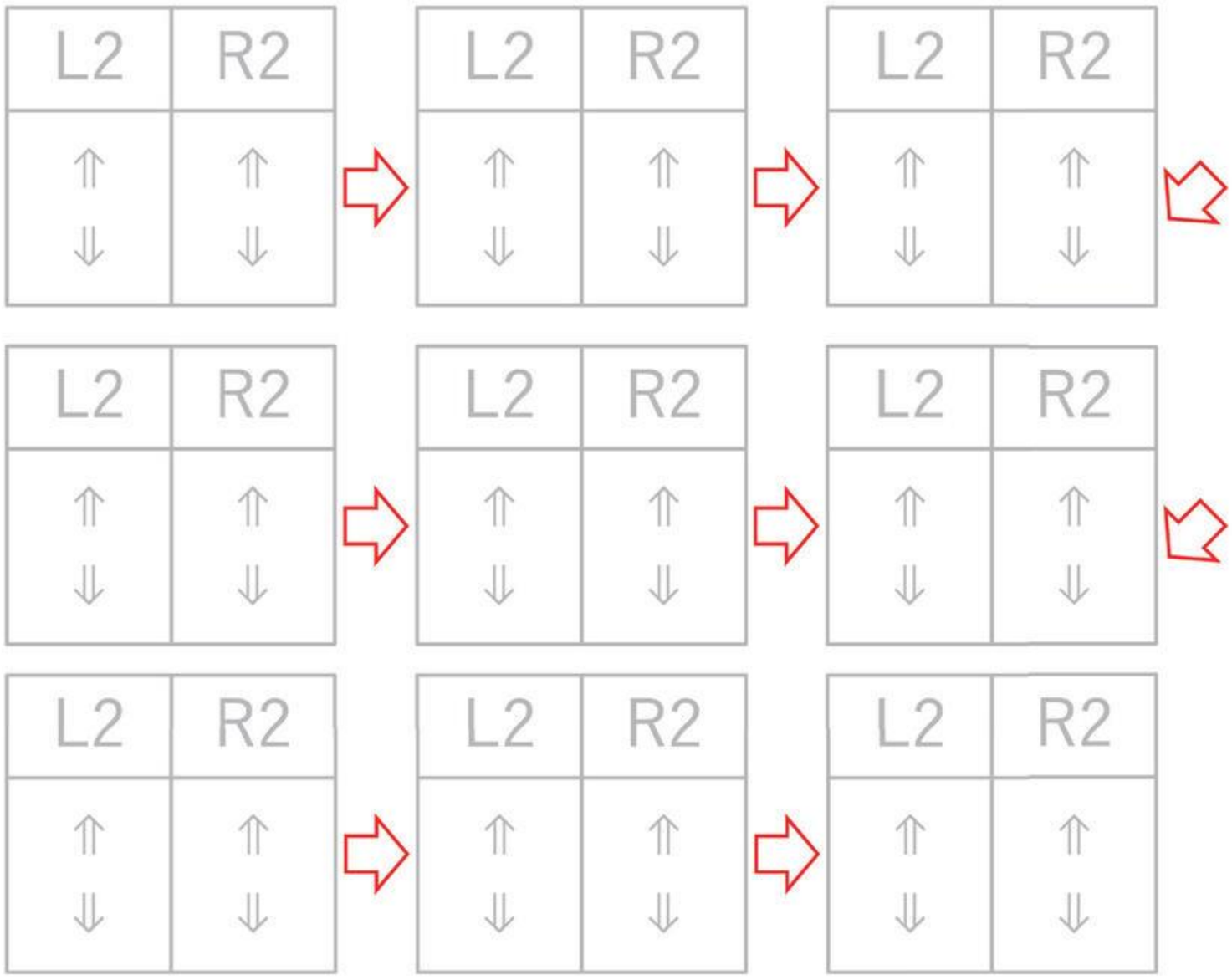
3.2. 後退の手動制御

同じように、後退動作の順番と動きを見て記録してみましょう。



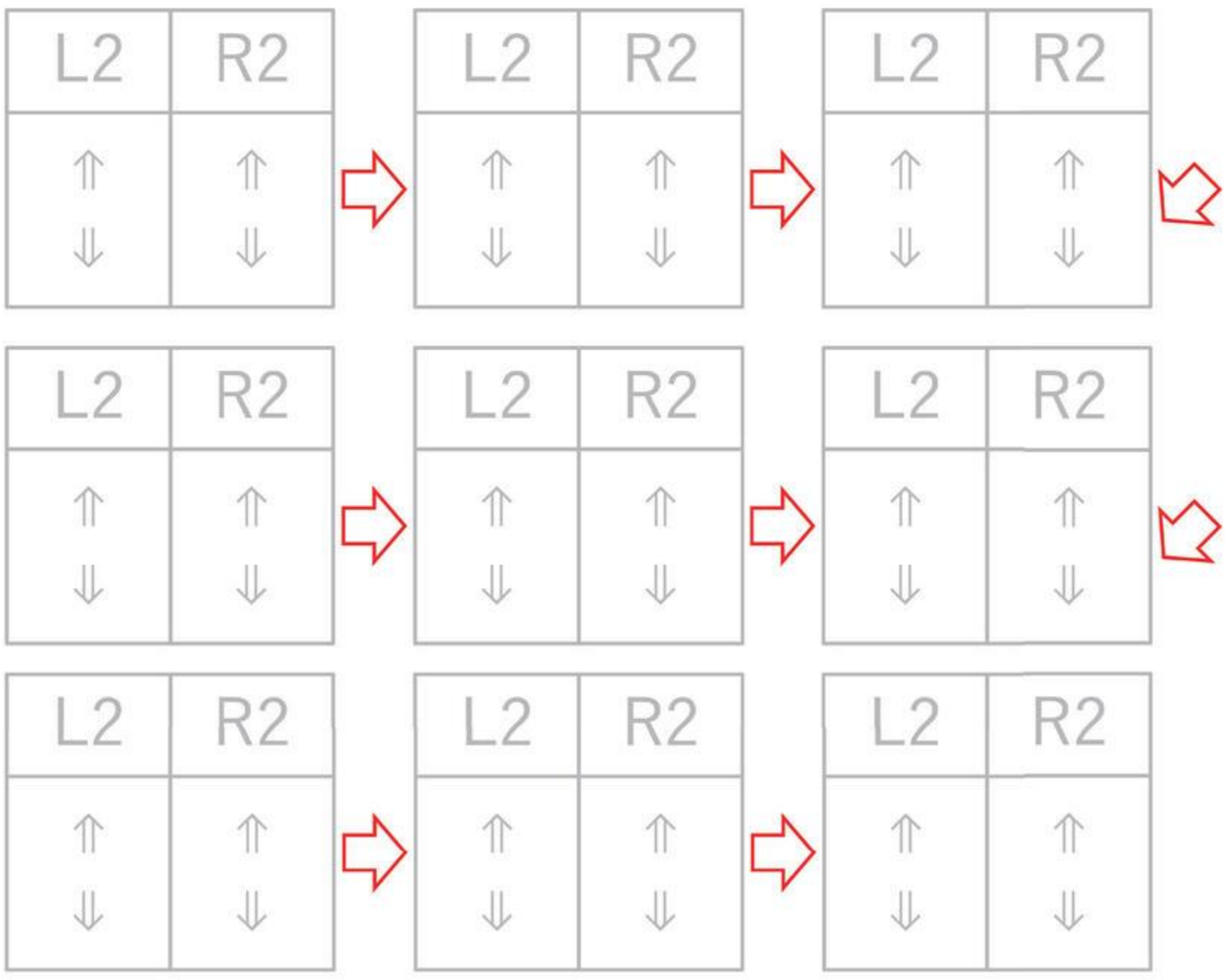
3.3.　右旋回（せんかい）の手動制御

同じように、右旋回（せんかい）動作の順番と動きを見て記録してみましょう。



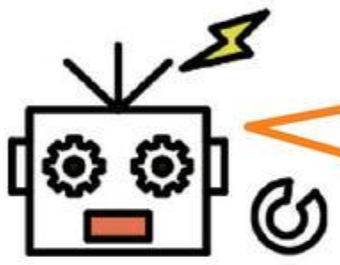
3.4.　左旋回（せんかい）の手動制御

同じように、左旋回（せんかい）動作の順番と動きを見て記録してみましょう。



4. まとめ（目安5分）

今回は、二足歩行ロボットの腰下部の組み立てを進めて、構造と歩行システムを勉強しました。
コントローラーでの操作は厄介（やっかい）ですが、この動きもパターン化してプログラミングさえすれば、簡単なボタン操作で実行させることができるようになります。
二足歩行ロボットは動くだけではなく、音や表示といったいろいろな機能を追加することを最終の目的としているため、組み立てはまだまだ中盤です。次回に備えて、残りのパーツを紛失しないように綺麗（きれい）に整理整頓（せいとん）してしまっておきましょう。
次回は、二足歩行ロボットを完成させて、マトリクスLEDやスピーカーなども使っていきます。活用するものが増えるとプログラムも徐々に長くなりますが、基本に立ち返って少しずつ読み解いていけば、きっと理解が進むと思いますよ！



手動制御はリズムが肝心だったね！

《次回必要なもの》

次回は、今回残ったパーツを持ってきてください。

講

第2回の理解度を確認してください。
・二足歩行ロボットの足部を完成させる
・二足歩行ロボットの制御テクニックを学ぶ
・二足歩行ロボットの歩行パターンを学ぶ
今回は、六脚ロボットでも行ったモーションパラメータの学習と手動制御の実験を中心に行いました。手動制御の答え合わせは次回行います。
いよいよ、第3回では二足歩行ロボットの完成に近づけます。
ACアダプターも忘れずにご準備ください。