講師用

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

アームロボット②

（第3回/第4回テキスト）

必ず、生徒に授業日と自分の名前を記入させるようご指導をお願いいたします。

第3回授業日 2024年　月　日

第4回授業日 2024年　月　日

名前

Human ヒューマンアカデミー ジュニア STE"AMスクール

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

2024年8月授業分

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

アームロボット②

第3回

アームロボットをカシコク動かす

講師用

目　　次

0. アームロボットをカシコク動かす

0.0. 「アームロボットをカシコク動かす」でやること

0.1. 必要なもの

0.2. アームロボットの注意点

1. アームロボットのしくみと動作原理

1.0. 「ArmControl」の実行

1.1. 「ArmControl」のプログラム

1.2. リンク機構のしくみ

1.3. アームロボットの動作原理

2. サーボモーターの制御と調整

2.0. 「Servo_write」のプログラム

2.1. 「Servo_writeUs」のプログラム

2.2. 初期位置の確認と調整

2.3. 「semiAutoArm」のプログラム

3. まとめ

○　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。

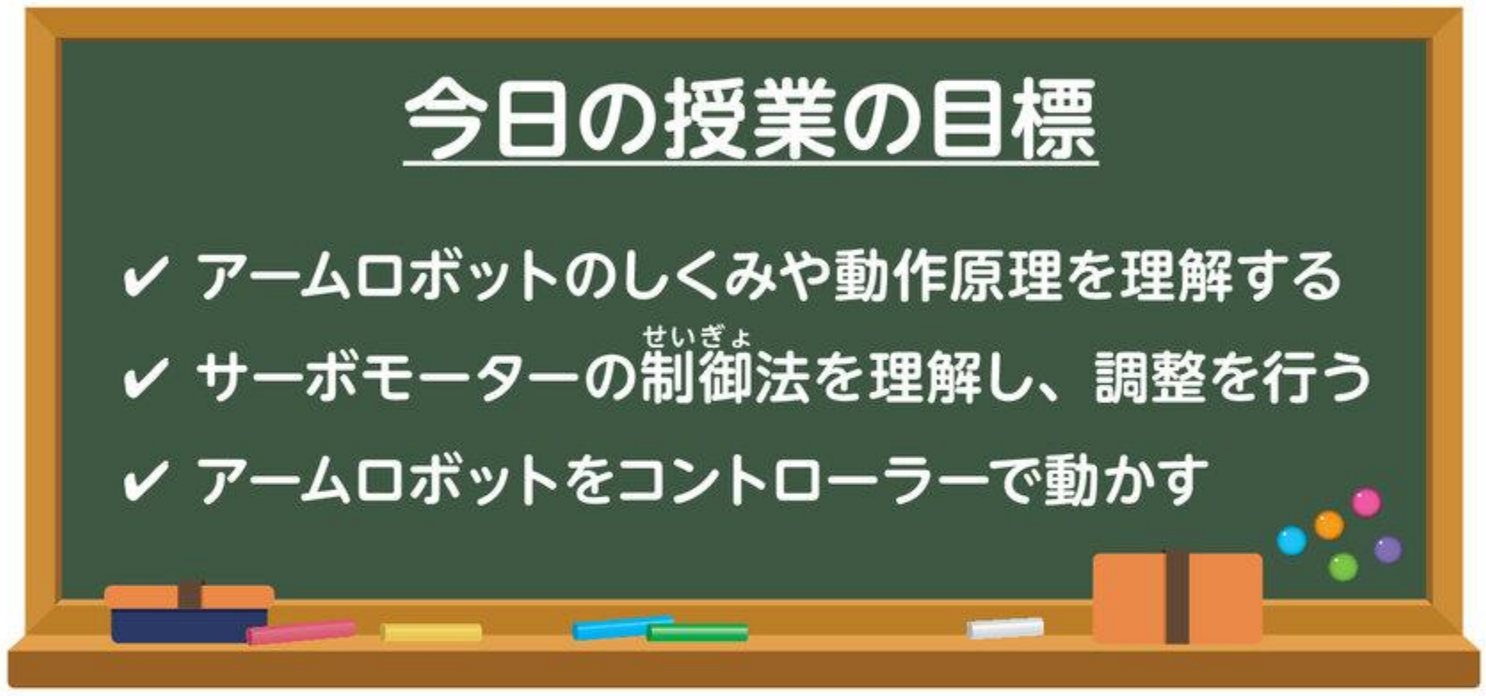
○　今回の目標を黒板に予め書いておきます。

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

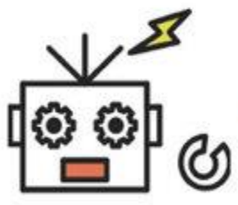
目安時間は授業時間 120 分のうち、休憩 10 分程度を取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0．アームロボットをカシコク動かす（目安10分）

0.0. 「アームロボットをカシコク動かす」でやること



今回の授業では、コントローラーを使って手動でアームロボットを動かします。
また、ロボットの最終調整をします。前回はハードウエア（ロボット）の調整をしておきましたが、今回はソフトウエア（プログラム）によって微調整を行います。コンピュータに性能を発揮させて操作性能を上げる、という流れで学んでいきます。
「問題を発見して、それをどう解決して、素晴らしい結果に結びつけるのか」、こういった問題解決力は今後のロボットの開発だけでなく、人生においてもとても役に立ちます。ロボット開発を通して、ロボット博士になるだけではなく、人生の達人にもなれるのです！
今回は解決方法として、数学の知識が必要となります。「そもそも勉強ってなんでするんだろう？」と思っている人も多いですよね？ 数学というのは、勉学の目的ではなくて、あくまで問題解決の道具にすぎないのです。説明ばかり聞いても仕方がないので、まずはやってみましょう！



大切なのは成功や失敗といった結果ではなく、そのプロセスなのダ！
ワレナガラ名言！

0.1. 必要なもの

前回つくったロボットと、以下のパーツを準備しておきましょう。
アームロボットは、動かす前にサーボモーターの配線を確認しましょう。

USB ケーブル 1	コントローラー 1	AC アダプター 1
		

図 0-0　必要なもの

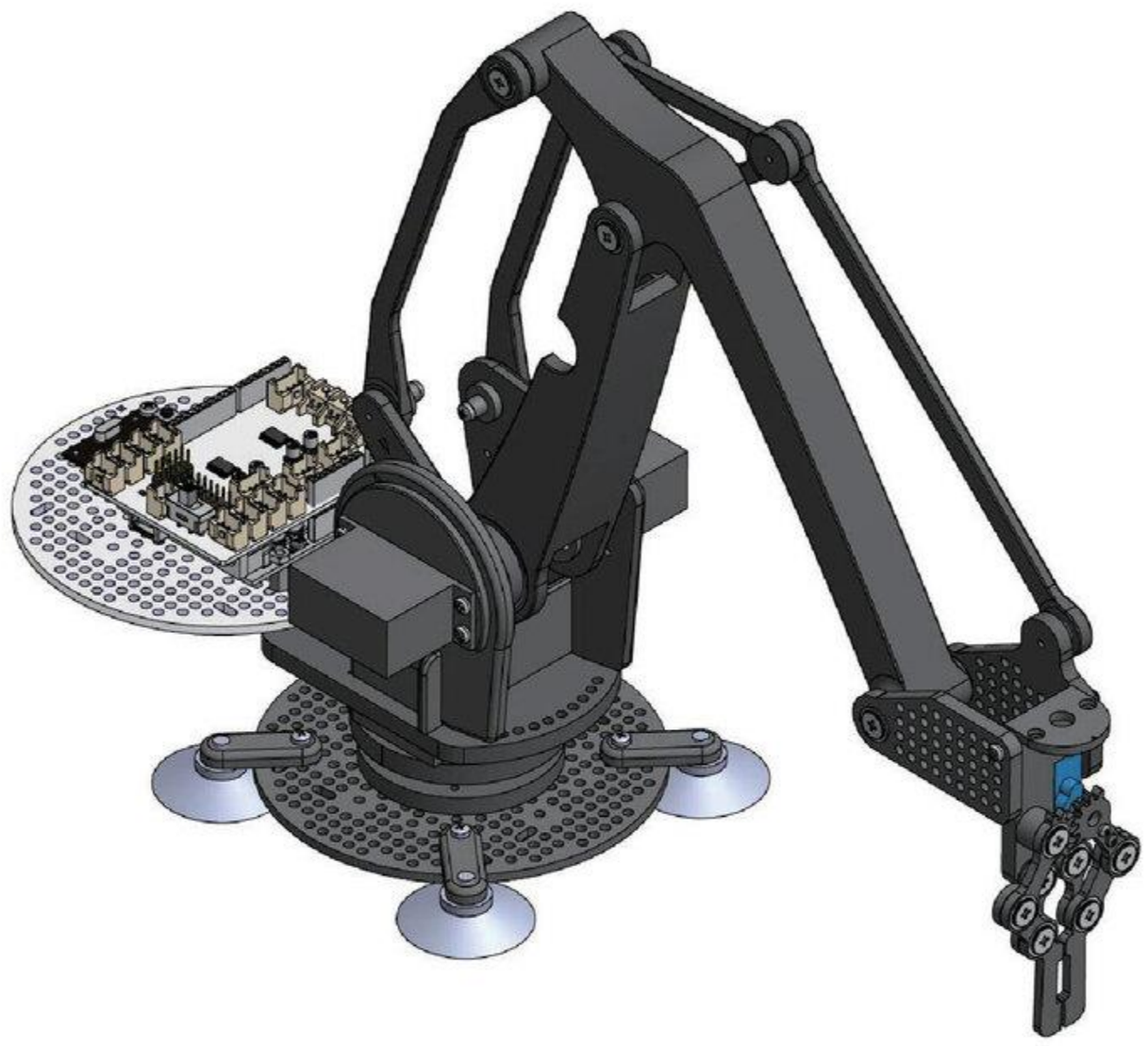


図 0-1 アームロボット本体

0.2.　アームロボットの注意点

1）AC アダプターの使い方

アームロボットは電気をたくさん消費するので、安定的に動かすために電池ボックスではなく、AC アダプターを使います。家庭用コンセントに接続した AC アダプターの電源プラグを、マイコンボードの電源ポートに差し込んで使用しましょう。

POINT

AC アダプターを使用する場合、前回書き込んだプログラムが残っていて、サーボモーターが急に動き出す場合があります。十分に注意をしましょう。

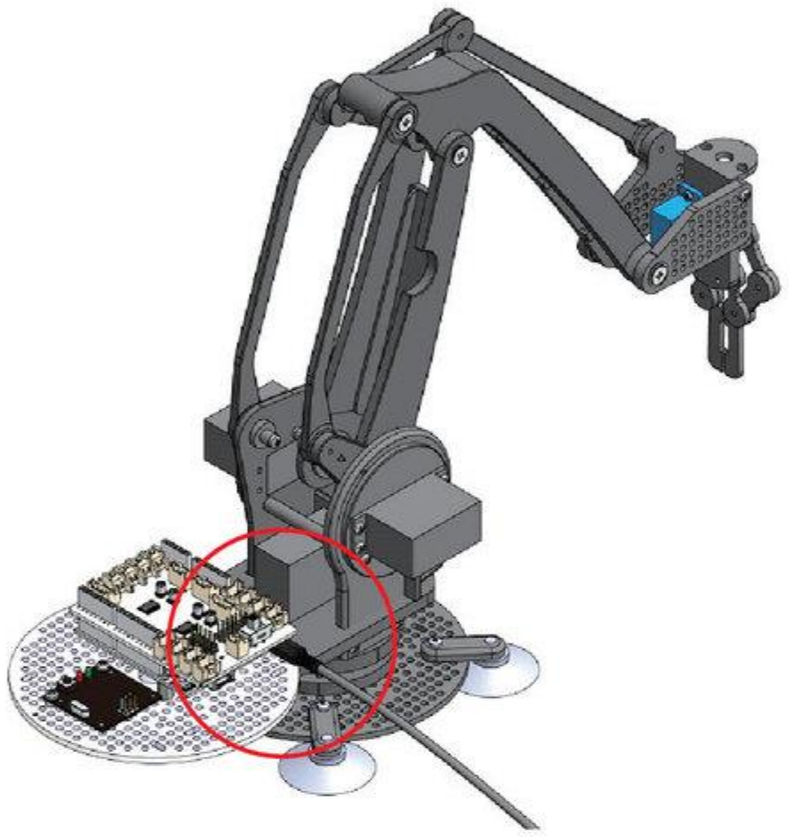


図 0-2　AC アダプターの接続

講　図 0-2 には描かれていませんが、実際にはロボプロシールドにリボンケーブル（無線受信モジュールと接続）、タッチセンサー（D3）、そして 3 個のサーボモーターが接続されています。
タッチセンサーは動作スタートの合図に使用します。

2）動作前の確認事項

アームロボットを動かすときは事前に、左側面、右側面、ベース部の各マークの位置が合うように手で調整しましょう。マークの位置が合った状態で動作を開始しないと、思っていた通りの動きにならない可能性があります。

⚠注意！

複数のサーボモーターを同時に動かすと、思わぬ動きで指を挟まれたりして怪我をする可能性があります。ロボットを持つときは、アームの背中の方を持ち、関節を持たないようにしましょう。また、プログラム実行中はアームロボットの周囲に顔を近づけると危険です。アームロボットを動かすときは、周囲を整理してパソコン等の機器にぶつからないように注意しましょう。

1．アームロボットのしくみと動作原理（目安 50 分）

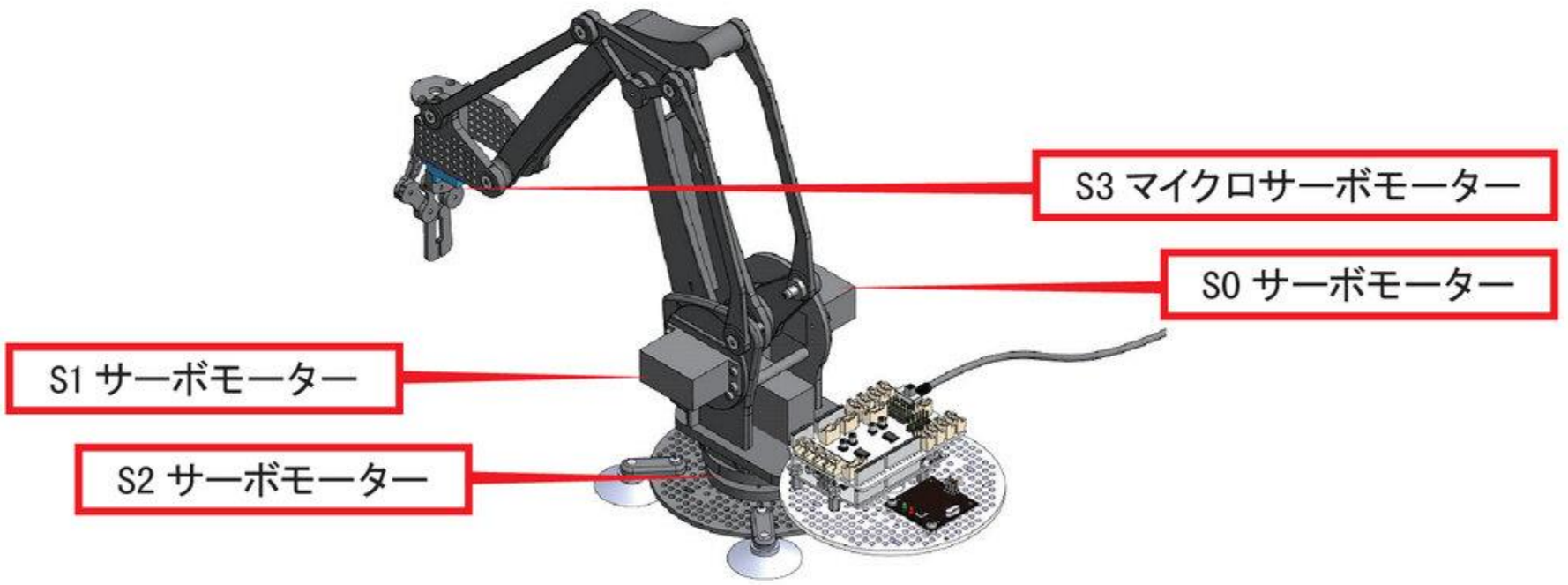
1.0.　「ArmControl」の実行

この章では、アームロボットがどうやって動いているのかをプログラムや機構（きこう）などの面から学んでいきましょう。
まずは、コントローラーとペアリングしてください。以下のプログラムを実行して動作確認をしましょう。タッチセンサーが動作開始の合図でしたね。

プログラムの書き込み
RoboticsProfessorCourse2 ＞ ArmRobot2 ＞ ArmControl

やってみよう！

コントローラーのアナログスティックで操作（そうさ）をしたとき、どの操作（そうさ）でどのサーボモーターが動いているか観察しよう。
そして、下のコントローラーの図の空欄（くうらん）にサーボモーターが接続されているコネクターの記号（S0 ～ S3）を書き込（こ）もう。



左スティック上下	左スティック左右	右スティック上下	右スティック左右
			
S1	S3	S0	S2

1.1. 「ArmControl」のプログラム

どのスティックで、どのサーボモーターが制御されているかがわかりましたね。
では、「ArmControl」のプログラムをのぞいてみましょう。４つのパートに分けて見ていきます。

1）#define（プログラム中の定数に名前をつける）

#define はプログラムで使われる定数（固定して変化しない値）に名前をつけています。そうすることで、４個のサーボモーターの原点位置と、それぞれの可動域の角度を決めているのです。
サーボモーターの可動域は **180 度**です。今回のアームロボットのアーム部は、以下のように設定しています。

プログラム「Arm Control」より抜粋

```
#define ORIGIN 90            // 原点位置
#define SERVOR_MIN 50        // サーボモーター右 最小角度50
#define SERVOR_MAX 100       // サーボモーター右 最大角度100
#define SERVOL_MIN 50        // サーボモーター左 最小角度50
#define SERVOL_MAX 140       // サーボモーター左 最大角度140
#define SERVOROT_MIN 0       // サーボモーターベース 最小角度0
#define SERVOROT_MAX 180     // サーボモーターベース 最大角度180
#define SERVOGRIP_MIN 55     // マイクロサーボモーター 最小角度55
#define SERVOGRIP_MAX 125    // マイクロサーボモーター 最大角度125
```

サーボモーターの制御角度を、下の図と照らし合わせてチェックしてみてください。

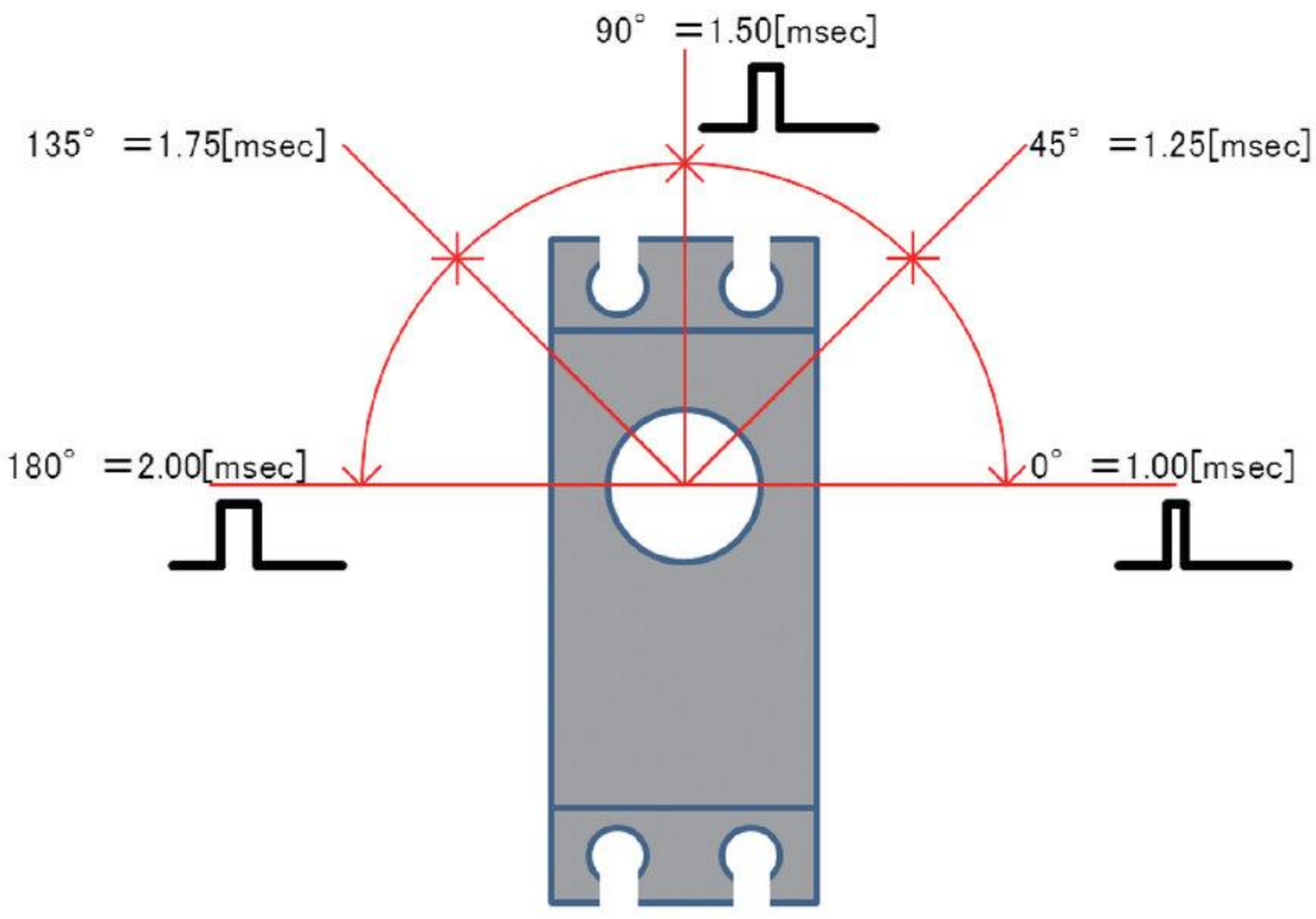


図 1-0　サーボモーターの制御

2）VarSpeedServo

VarSpeedServo は、各サーボモーターに名前をつけています。そうすることで、プログラム上で区別ができ、わかりやすくしているわけです。

講　varSpeedServo.h を使うと、サーボモーターを任意の速度で制御できます。

プログラム「ArmControl」より抜粋（ばっすい）

```
#include <RPlib.h>
#include <VarSpeedServo.h>
#include <PS2X_lib.h>
(中略)
VarSpeedServo servoR;     // サーボモーターを使うときのオマジナイ（servoRと名づける）
VarSpeedServo servoL;     // サーボモーターを使うときのオマジナイ（servoLと名づける）
VarSpeedServo servoROT;   // サーボモーターを使うときのオマジナイ（servoROTと名づける）
VarSpeedServo servoGRIP; // サーボモーターを使うときのオマジナイ（servoGRIPと名づける）
```

3）attach( )

attach() は、名前をつけたそれぞれのサーボモーターに、接続するコネクター（ピン番号）を割り当てています。

こうすることで、サーボモーターとコネクターのペアが整理され、正しく動くわけですね。

プログラム「ArmControl」より抜粋（ばっすい）

```
servoR.attach(S0, SERVO_MG995_MIN, SERVO_MG995_MAX);
                                        // servoRと名づけたモーターをS0に接続
delay(500);
servoL.attach(S1, SERVO_MG995_MIN, SERVO_MG995_MAX);
                                        // servoLと名づけたモーターをS1に接続
delay(500);
servoROT.attach(S2, SERVO_MG995_MIN, SERVO_MG995_MAX);
                                        // servoROTと名づけたモーターをS2に接続
delay(500);
servoGRIP.attach(S3, SERVO_SG90_MIN, SERVO_SG90_MAX);
                                        // servoGRIPと名づけたモーターをS3に接続
}
```

4）int

「int」は、「integer（整数）」の略です。名前の通り、整数を代入できます。-32,768 ～ 32,767 までの値を入れることができます。ここでは、コントローラーを操作するために、制御に使う値を決めています。

プログラム「ArmControl」より抜粋

```
int rud = 0, lud = 0, rlr = 0, llr = 0;
                                        // アナログスティックRLの上下左右
int angR = 90, angL = 90, angROT = 90, angGRIP = 90;
                                        // それぞれのサーボモーターの角度
```

1.2. リンク機構のしくみ

今回のアームロボットには四節リンクが３つ隠れています。それらのリンク機構が動かしやすさをアシストしています。実際に、リンク機構のしくみを図で見てみましょう。

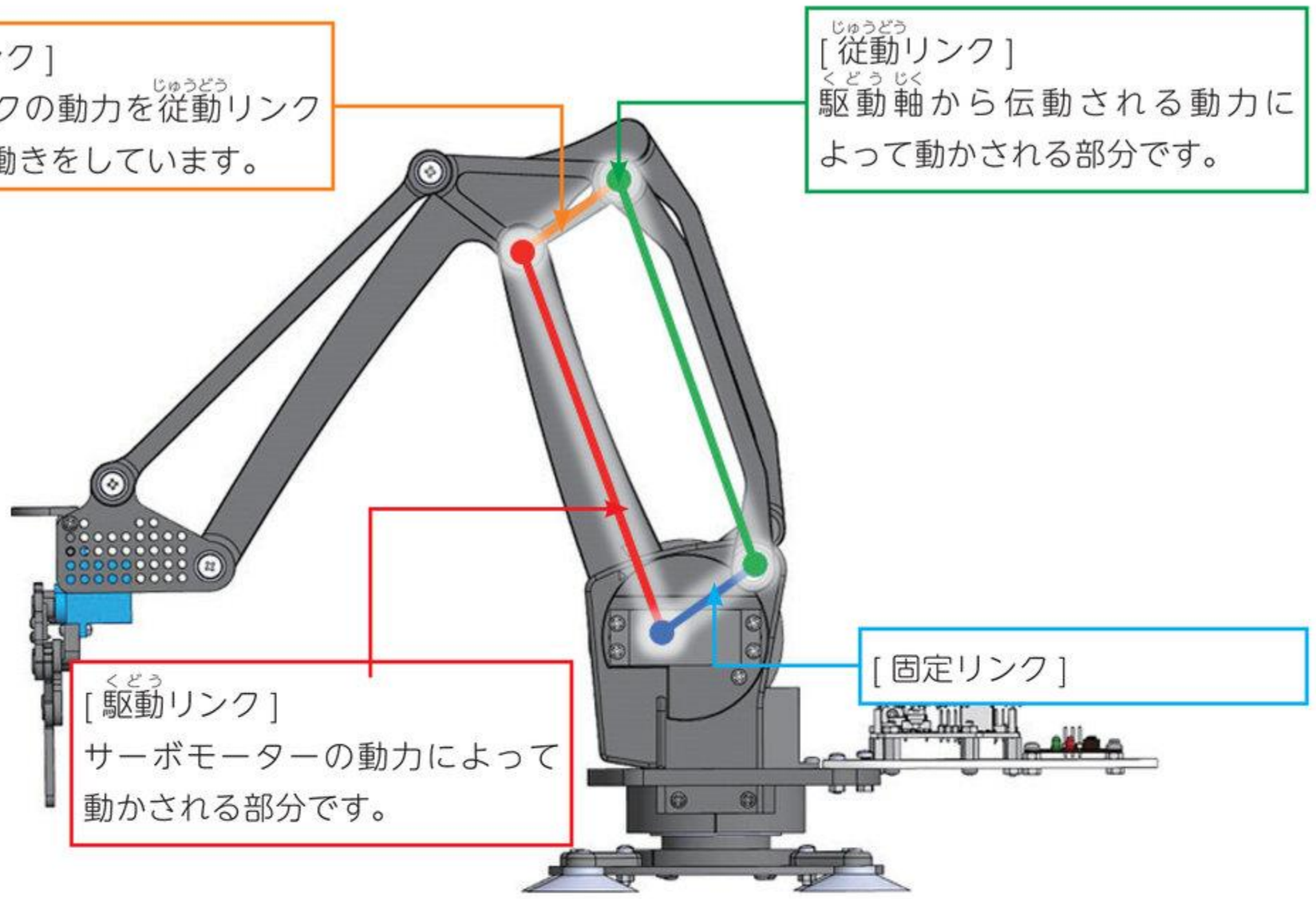
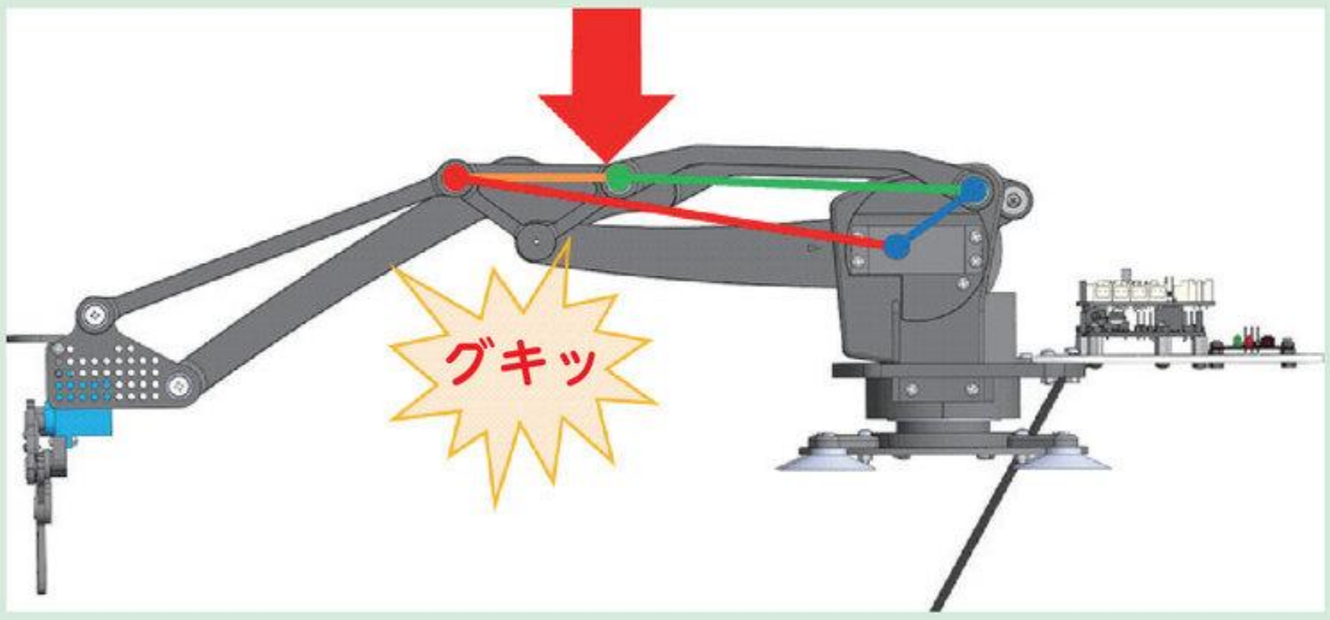


図 1-1　アームロボット左側面から見たリンク機構

豆知識

リンクロボットでも習いましたが、アームロボットのリンク機構も図のように特定の方向へこれ以上移動できない姿勢があります。矢印の方向に物理的に押しこめず、それ以上動かせないポイントを「特異点」といいます。



1.3.　アームロボットの動作原理

アームロボットは、サーボモーターの回転を利用して動作します。
つまり、こちらが指定できるのは「どのサーボモーターを、何度回転させるか」という点だけですね。
しかし、イメージしてみてほしいのですが、基本的にアームロボットは「30cm 前の物をつかむ」など、距離（長さ）を使って命令する場面のほうが多いのではないでしょうか。

この問題を解消するのに、「数学」が大活躍するのです。
三角形には、３つの辺の「長さ」と、どれか１つの角の「角度」のうち、３つがわかれば残り１つも求められる、というきまりがあります。
図 1-2 の通りアームロボットの中には三角形のリンクがありますが、そのうち２つの辺はパーツの長さのまま固定です。その間の角度を何度にするか指定すれば、自動的に残り１つの辺の長さが決まります。つまり、「角度」を決めるだけで、アームを伸ばす「距離」を指定できる、というわけです！

マトリクス LED を扱ったとき、「特定の点だけ点灯する」「2 けたの数字になるよう点灯する」「指定した文字になるよう点灯する」といった命令を使い分けていたことを覚えていますか。
アームロボットもこれと同様「モーターの回転角度を指定する」「アームの移動距離を指定する」「アームの位置を指定する」といった命令があるので、状況に応じていちばん適したものを選ぶことができます。

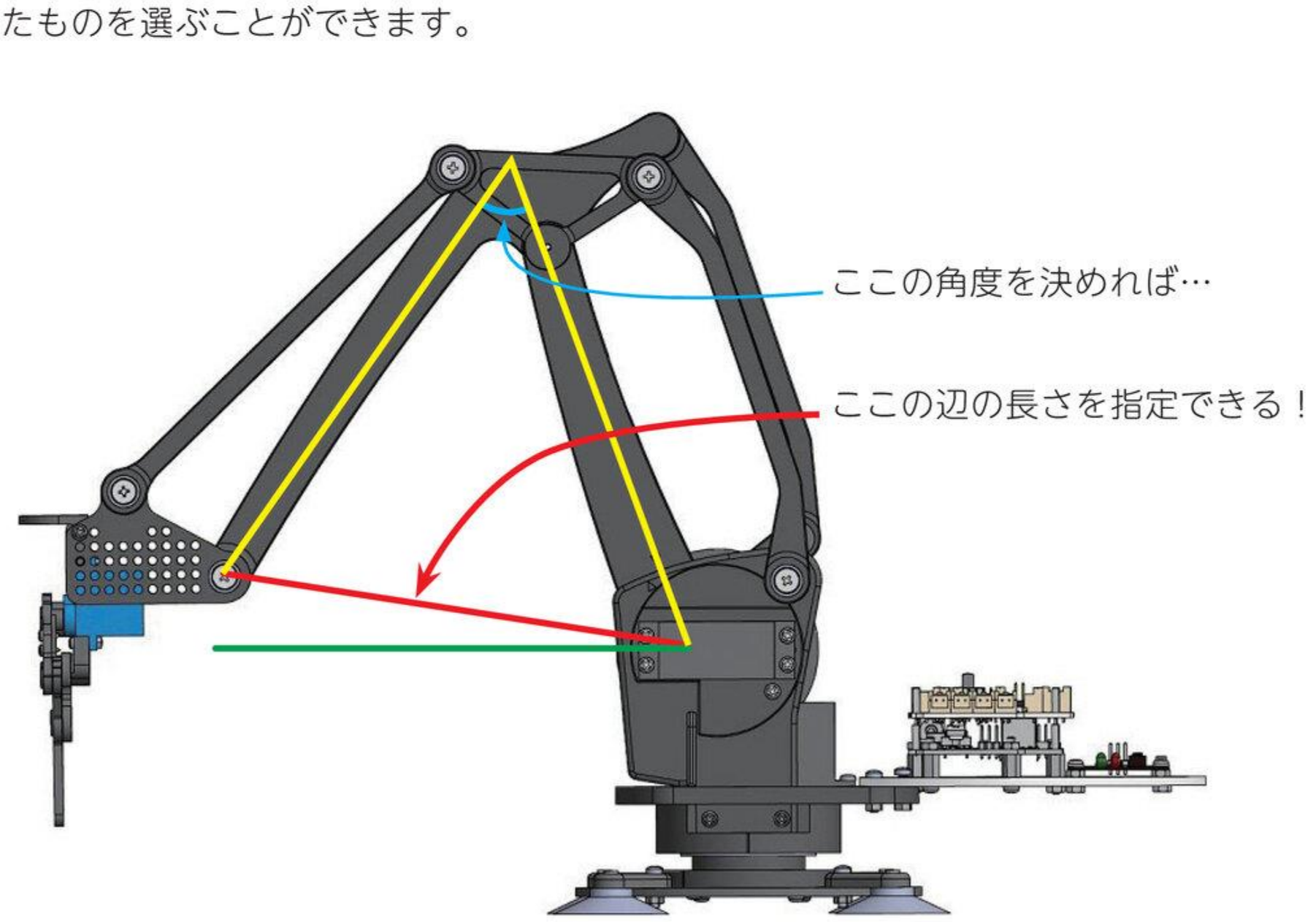
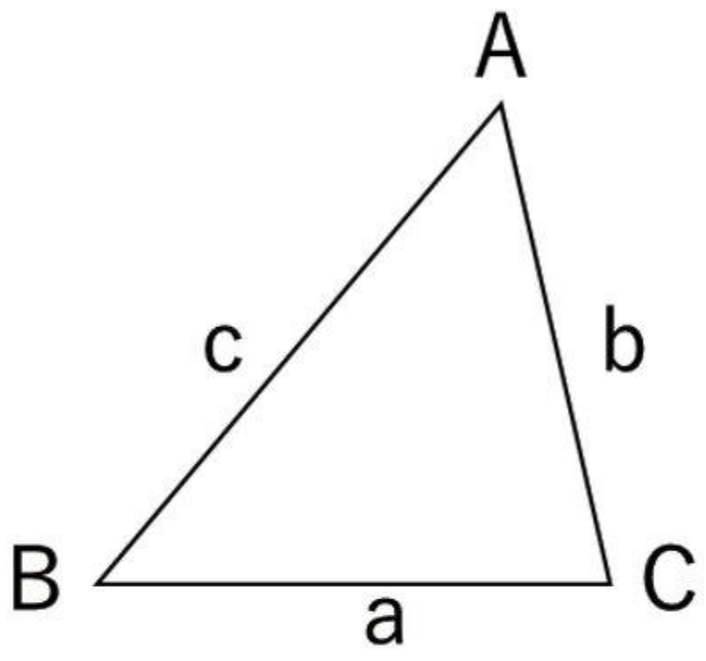


図 1-2　アームロボットのしくみ

コラム　「余弦定理」と「三角関数」

アームロボットの姿勢制御には「三角形の３つの辺と１つの角の大きさのうち、３つがわかれば残り１つも求めることができる」というきまりを使っています。
このきまりは、数学の世界で「余弦定理」とよばれていて、下のような公式があります。
式そのものは高校数学で学ぶ内容なので、いまは全く意味がわからなくて大丈夫ですが、とにかく「数学の複雑な公式も、実はちゃんと世の中で活用されている」ということは知っておいてほしいのです。
知っている命令をうまく組み合わせて新たな動作を生み出していくプログラミングと同じように、数学も「すでに存在する知識（公式やきまり）を、いかに活用して未知の問題に役立てるか」を考える学問なのです。「こんな公式が何の役に立つの？」という疑問の答えを探すのも、数学の大事な要素と言えるでしょう。



$$\cos A = \frac{b^2 + c^2 - a^2}{2bc}$$

余弦定理

コラム　アームロボットの活用

●パワーショベル

みなさんが思いつく実用化されているロボットアームとは何でしょうか？ すぐに思いつくのは、パワーショベルかもしれません。俗にいうショベルカーのことです。
ロボットといっても実際は職人さんが手動で関節の操作をします。ちょうど、今回のロボットでも行ったマニュアルで関節を一個一個動かすようなものです。
一方、経験を積まなくても動かせるようにロボット化されたパワーショベルも存在します。
また、もっとロボットらしい双腕のパワーショベルも存在します。

“ユンボ”『ウィキペディア (Wikipedia): フリー百科事典』
更新日時 :2017 年 12 月 26 日 (火) 10:03
URL:http://ja.wikipedia.org/wiki/ ユンボ

●ガントリークレーン

ガントリークレーンとは港でコンテナの積み込み、積み下ろしをしている超大型のクレーンのことです。こちらは「直行座標型」という構造を持っていますが、これは身近なものでいうとクレーンゲームのように xyz 方向に動きます。ガントリークレーンは、なんと 30 トンもの荷物を上げ下げできます。

“ガントリークレーン”『ウィキペディア (Wikipedia): フリー百科事典』
更新日時 :2018 年 4 月 14 日 (土) 13:29
URL:http://ja.wikipedia.org/wiki/ ガントリークレーン

実用化されているアームロボットは精密性と安全性を向上させるために高い技術で製造されています。今回のアームロボットは、そこまでの精度や強度はありませんがロボットの機構を知るうえでは良いマシーンになるでしょう。

2. サーボモーターの制御と調整（目安 45 分）

2.0. 「Servo_write」のプログラム

この章ではサーボモーターの制御と調整の方法を学びます。
まずはサーボモーターをプログラムで制御する命令です。2 つ覚えましょう。
まず 1 つは「Serve_write」の命令です。実際にプログラムを実行して動作確認をしましょう。
動作スタートの合図は D3 に接続したタッチセンサーのスイッチです。

プログラムの書き込み
RoboticsProfessorCourse2 > ArmRobot3 > Servo_write

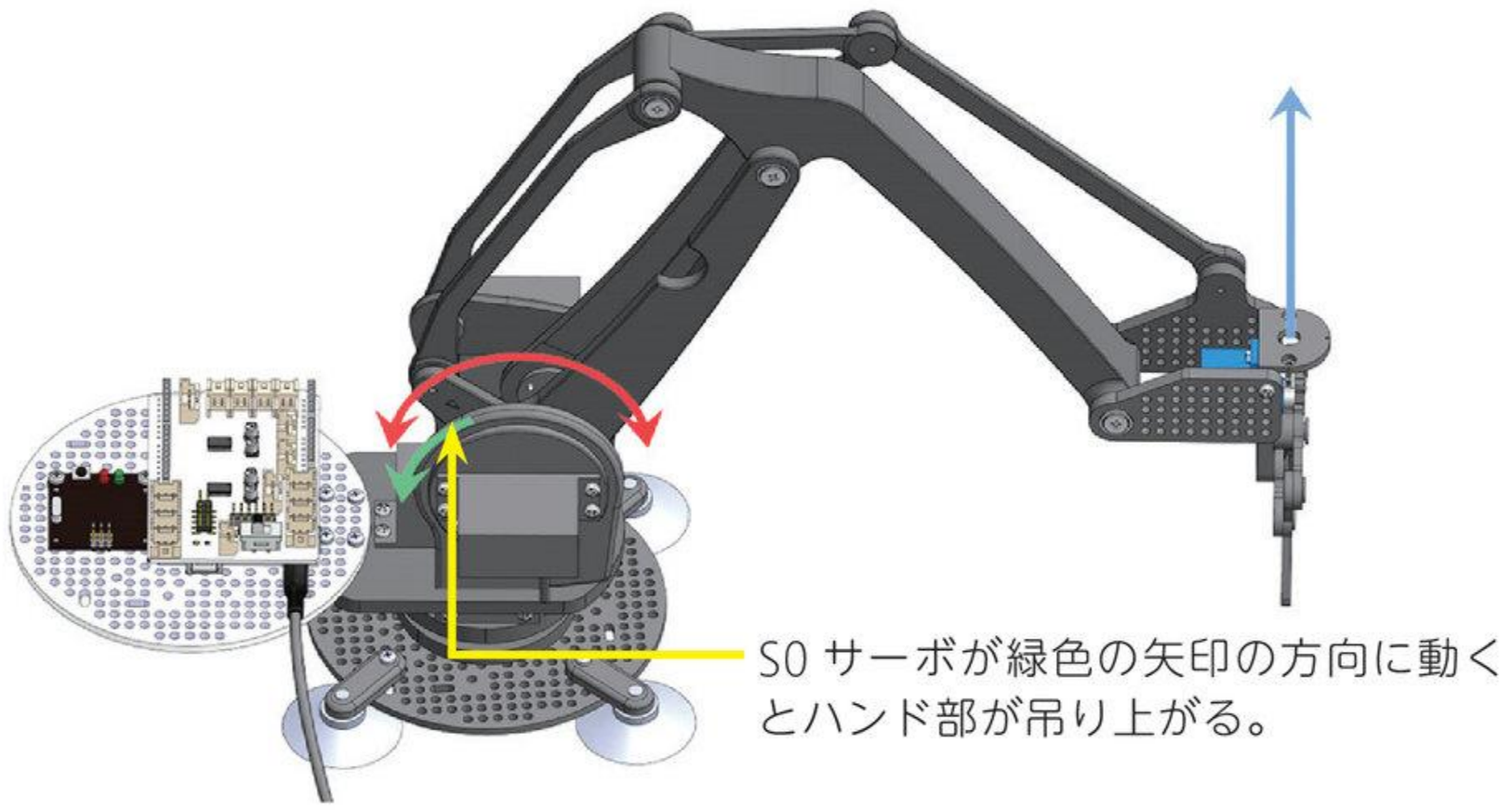


図 2-0　S0 サーボモーターの動作

実行結果：プログラムを実行後、D3 に接続したタッチセンサーを一度押すとサーボモーターが原点位置まで動く。さらにもう一度タッチセンサーを押すと S0 サーボモーターがプログラムされた範囲で動く。
このときの S0 サーボモーター（赤い矢印）の動きと、アームの動き（青い矢印）を見ると、S0 サーボモーターが緑の矢印方向に動いたときに、アームが吊り上るという相互関係がわかります。

講　サーボモーターの原点位置＝アームロボットの初期位置（▽マークの位置）となります。アームロボットの初期位置にわずかなズレがある場合は、テキスト後半で扱う初期位置の調整方法を確認のうえ調整してください。

では、ここでプログラムの中身を確認してみましょう。

プログラム「Servo_write」より抜粋

```
for(double deg = 0; deg < RANGE / 2; deg += 0.1){
    servo.write((int)deg + ORIGIN);
    delay(5);
}
```

第1回でも登場した命令ですね。

命　　令「write」
実行結果：サーボモーターの出力軸を指定の角度に回転させる
使 い 方：myservo3.write(90);
　　// myservo3と名付けたサーボモーターの出力軸の角度を90度にする

これは「角度」を指定する命令です。
for文内では移動角度の変数 deg を0.1度ずつ増やしていくため、小数の値を含むことができる変数型 double を用いています。
ただ、write の値には整数しか入れることができないため、黄色の部分で deg を整数型 int に変更する処理が入っています。
この場合、サーボモーターの軸回転が1度ずつ行われる、つまり180度の角度を1度ごとに分けて制御することになります。

豆知識

( ) の中に変数型を入れて変数名の前につけることで、変数型を変更することができます。これを型変換（キャスト）といいます。

2.1. 「Servo_writeUs」のプログラム

さきほどと同じ動作を行う、違う制御プログラムを紹介します。
では続いて、プログラム「Servo_writeUs」を開いて実行しましょう。動作のスタートの合図は、D3 に接続したタッチセンサーです。

プログラムの書き込み
RoboticsProfessorCourse2 > ArmRobot3 > Servo_writeUs

実行結果：「Servo_write」と同じ動きをする。
動作は同じでも制御のやり方は異なります。こちらも、プログラムの中身を確認してみましょう。

プログラム「Servo_writeUs」より抜粋

```
for(int us = 0; us < RANGE / 2; us++){
    servo.writeMicroseconds(us + ORIGIN);    // 0-180まで
    delay(5);
}
```

命　　令　「writeMicroseconds」
実行結果：サーボモーターに指定の幅のパルスを発し、回転させる
使 い 方：servo.writeMicroseconds(1500);
　　// servo と名付けたサーボモーターの出力軸を原点位置にする

先ほどの write と同様、「角度」を指定してサーボモーターを制御する命令です。
ただ、こちらは 180 度の角度を 1,856 段階に分け、約 0.097 度ずつ回転させることで、より細かい制御が可能になっています。
このアームロボットでは、サーボモーターの精度の個体差や、各パーツの微妙なかみ合わせの影響があるため、精度に大きな差が出るほどではありませんが、同じ「角度」を指定する命令だけでも状況に応じて使い分けられることは知っておきましょう。

2.2.　初期位置の確認と調整

サーボモーターの制御方法についてはもうばっちりですね。続いては、調整方法です。
アームロボットの製作のときに、サーボモーターの原点位置を定めてからモーターホーンを取り付けましたが、サーボモーターの回転軸とサーボホーンのはめ合いによっては多少の位置ずれが生じます。その位置ずれを調整するために、プログラムを修正することで解決することができます。

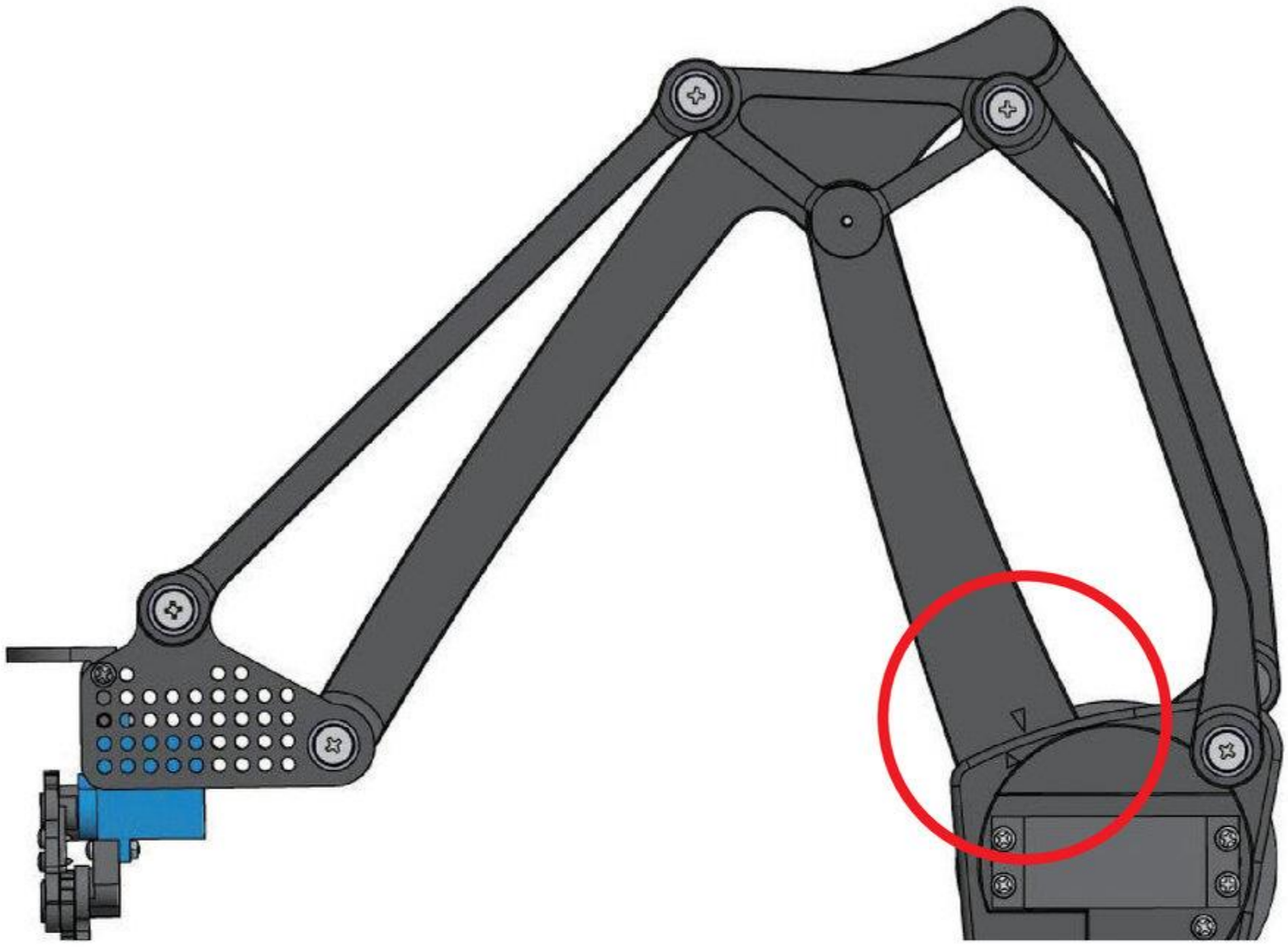


図 2-1　初期位置の確認

まずプログラム「OriginAdjust」を実行し、D3 に接続されたタッチセンサーを押します。
動作が終わったとき（停止時）に、**図 2-1** のように各マークの位置などがずれている場合、調整を行います。

プログラムの書き込み

RoboticsProfessorCourse2 > ArmRobot3 > OriginAdjust

プログラムを開いたら、左側面、右側面、ベース部の各マークなどの位置が合うように以下の黄色い部分に適切な値を入れながら、調整しましょう。また、ハンド部もグリップの角度が 180 度になるように調整しましょう。

プログラム「OriginAdjust」より抜粋（ばっすい）

```
#define ORIGIN 90                  //原点位置
#define RANGE 30
#define ADJUST_SERVO_R 0.0         // S0 はここで調整（ーで前）
#define ADJUST_SERVO_L 0.0         // S1 はここで調整（ーで前）
#define ADJUST_SERVO_ROT 0.0       // S2 はここで調整（＋で左旋回）
#define ADJUST_SERVO_HAND 0.0      // S3 はここで調整（＋でオープン）
```

POINT

プログラム「OriginAdjust」内の黄色のラインの数字は角度です。
1.0 だったら１度です。数字をマイナスにすると、逆方向に調整されます。

SERVO_R	(S0)：- の数字でアーム方向へ調整（+ は逆方向）
SERVO_L	(S1)：- の数字でアーム方向へ調整（+ は逆方向）
SERVO_ROT	(S2)：+ の数字で左旋回方向へ調整（- は逆方向）
SERVO_HAND	(S3)：+ の数字でハンドを開く（- は閉じる）

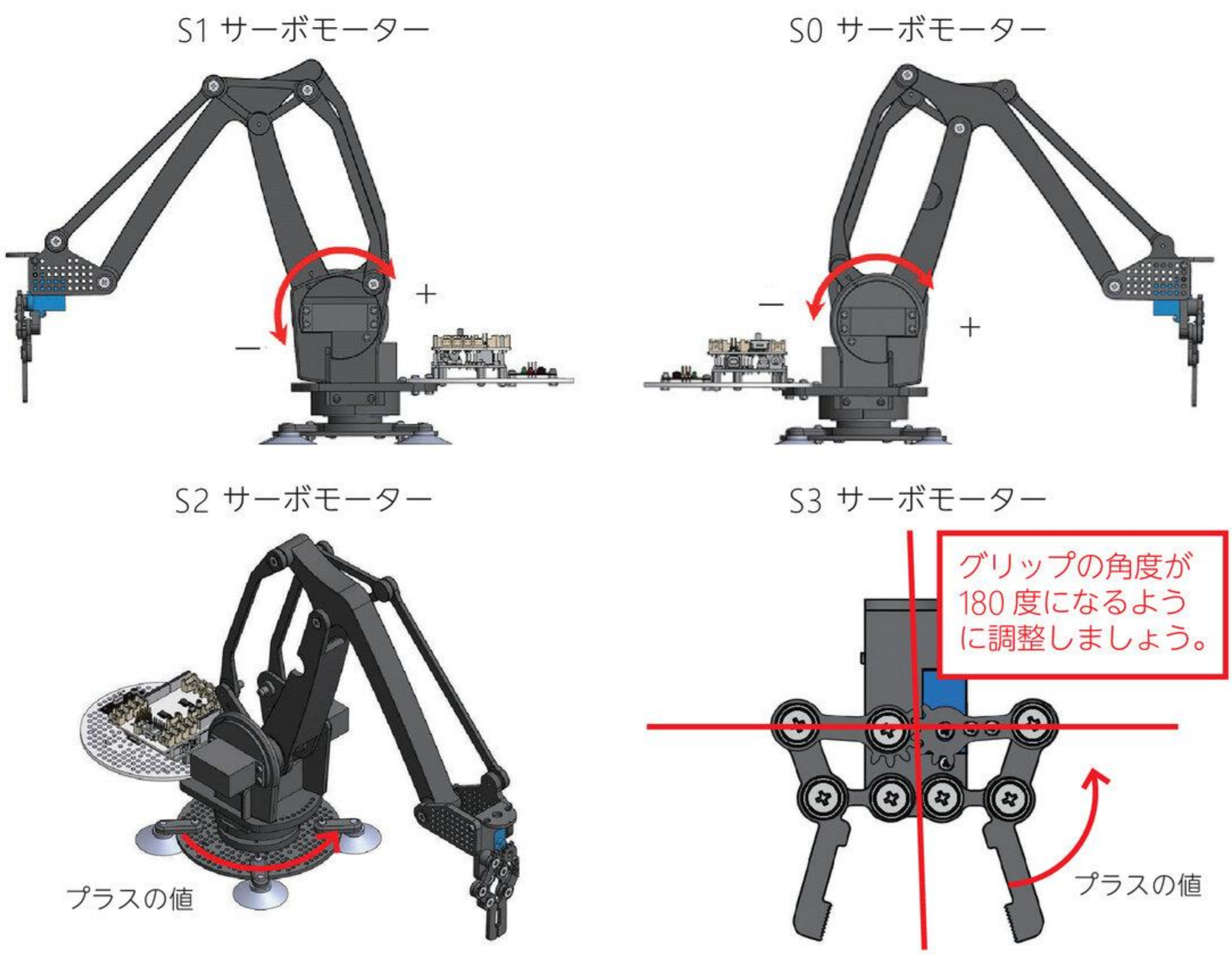


図 2-2　アームロボットの調整

各部の調整が完了したら、**表 2-0** に調整値を書きとめておきましょう。調整値は、小数点以下第一位まで設定できます。

表 2-0 サーボモーターの調整値

S0	S1	S2	S3

ここで取得した初期位置の調整値は、これ以降のプログラムで armBot.setup( ); という部分に入力すると、初期位置が補正されます。忘れずに書きとめておきましょう（調整不要な人は 0 の値のままで問題ありません）。

プログラム例

```
armBot.setup( 0.0,0.0,0.0,0.0 );
//armBoot.setup( S0 ｻｰﾎﾞ ﾓｰﾀｰ , S1 ｻｰﾎﾞ ﾓｰﾀｰ , S2 ｻｰﾎﾞ ﾓｰﾀｰ , S3 ｻｰﾎﾞ ﾓｰﾀｰ ) ;
```

2.3. 「semiAutoArm」のプログラム

「SemiAutoArm」は「ArmControl」と同じく、コントローラーを使うプログラムですが、計算値を入れて動きがスムーズになるように改良したものです。両者の詳しい違いや、「semiAutoArm」の内容については、また次回解説します。今回は操作（そうさ）を体験してみましょう。

プログラムの書き込み

RoboticsProfessorCourse2 > ArmRobot3 > semiAutoArm

図 2-3 「semiAutoArm」のコントローラーの操作（そうさ）方法

プログラム実行後、D3 のタッチセンサーを 2 回押すと操作（そうさ）が可能になります。
「ArmControl」との操作（そうさ）の違いはハンド部の動作です。ハンドは十字ボタンの上下で開閉でき、「ArmControl」に比べて少し動かしやすくなったのではないでしょうか。

POINT

ハンド部は、個々のロボットの状態や取り組む課題内容によって、開閉の度合いをさらに調整する必要があるかもしれません。そのような場合の調整方法を紹介します。
プログラム「semiAutoArm」を実行後、D3 に接続したタッチセンサーを２回押します。アームロボットの各サーボモーターが初期位置に動いたらコントローラーの十字ボタンの上ボタンを押します。そこでの状態によってプログラム中の調整値（下の黄色のライン）の数字を変更してください。
ハンド部が閉じない場合は、マイナスの値を、ハンド部が閉じてもモーターが動き続ける場合は、プラスの値を入力してください。なお、調整幅は±5.0 刻みで調整しましょう。

プログラム「semiAutoArm」より抜粋

```
void setup(){
    pinMode(D3, INPUT_PULLUP);
                        // D3に接続されたタッチセンサーを読み取る
    ps2x.config_gamepad(13, 11, 10, 12, true, true);
                        // コントローラーを使うときのオマジナイ


    while(!digitalRead(D3));
    armBot.setup(0.0, 0.0, 0.0, 0.0);
                        // 調整値（servoR,servoL,servoROT,servoGRIPの順）
```

⚠注意！
プログラム実行後、長時間操作を行わないとアームが動き出す可能性があります。
使用しないときは、AC アダプターを小まめに外しておきましょう。

やってみよう！

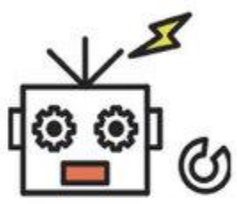
アームロボットの操作に慣れてきたら、**図 2-4** のように「机に消しゴム等をおいて、ハンドでつかみ取って移動して落とす」といった動作をやってみよう。

図 2-4 「semiAutoArm」チャレンジ

コントローラーのアナログスティックは各関節の動きに対応しています。
サーボモーターの回転を指示する形だと、アームがどう動くかいちいち考えなければならず、なかなか難しいですね！

３．まとめ（目安５分）

今回はやっと自由にアームロボットを動かすことができました！
次回は操作（そうさ）するのではなく、プログラムによって自由に動かせるようにしましょう。ペンをもたせてロボットに「図をかかせる！」ことにチャレンジをします。
非常に難易度の高いチャレンジですが、いままで身につけた調整テクニックをいかして成功させましょう！



次回は、アームロボットでお絵かきダゼ～！
ロボットとアートの融合（ゆうごう）ダネ！

講

○　以下の理解度を確認します。
・アームロボットのしくみや動作原理を理解する
・サーボモーター制御法を理解し、調整を行う
・アームロボットをコントローラーで動かす

○　次回のテーマは「アームロボットで文字や記号をかく」です。
アームロボットの残りのパーツを忘れずに持参するようにご指導ください。

≪次回必要なもの≫

次回はアームロボットのほかに以下のパーツを持ってきましょう。

ドライバー 1	USB ケーブル 1	コントローラー 1	C-9 (アームロボットパーツ) 1
			
C-10 (アームロボットパーツ) 1	**サインペン 1**	**AC アダプター 1**	**タイヤ 1**
			
M3L25 ネジ 1			
			

図 3-0　次回必要なもの