

# Stduino

## ブロックプログラミング環境

---

### 入門ガイド 後編

【DC モーター サーボモーター】



本資料は、Stduino(スタディーノ)プログラミング環境のブロックプログラミング環境のチュートリアルになります。Stduino プログラミング環境の変更に伴い、加筆・修正が加えられる可能性があります。

## 目次

5. DC モーター .....	1
5.1. DC モーターの制御.....	1
5.1.1. Studuino 基板と DC モーターの接続 .....	1
5.1.2. 入出力ポート情報の設定 .....	2
5.1.3. プログラミング.....	3
5.2. 車の制御.....	6
5.2.1. プログラミング.....	6
5.2.2. DC モーター校正 .....	13
5.3. 加速度センサーを使用した車の制御 .....	15
5.3.1. Studuino 基板と加速度センサーの接続 .....	16
5.3.2. 入出力ポート情報の設定 .....	16
5.3.3. 加速度センサーの動作確認 .....	17
5.3.4. プログラミング.....	18
6. サーボモーターを動かす .....	28
6.1. サーボモーターの角度校正.....	28
6.1.1. サーボモーターの駆動軸角度の調整.....	28
6.1.2. Studuino 基板とサーボモーターの接続 .....	29
6.1.3. 入出力ポート情報の設定 .....	29
6.1.4. プログラミング.....	31
6.2. サーボモーター3つでアームロボをつくる.....	33
6.2.1. ロボットの組み立て .....	33
6.2.2. 入出力ポート情報の設定 .....	35
6.2.3. プログラミング.....	35

## 5. DC モーター

### 5.1. DC モーターの制御

DC モーターを制御する基本的なプログラムを通して、DC モーターの動かし方と設定を学びます。

#### 5.1.1. Studuino 基板と DC モーターの接続

以下の手順で車を組み立てます。

(1) DC モーターに以下のようにタイヤを取り付けます。

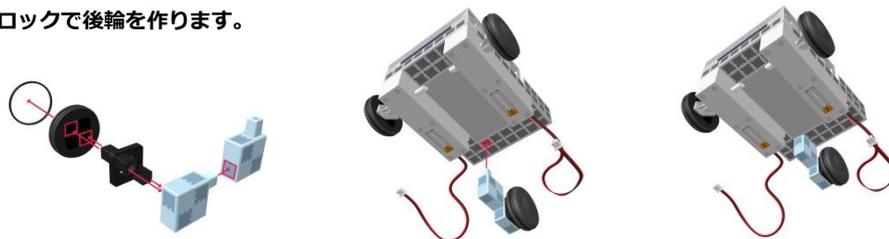
※左右対称に2つ作る。



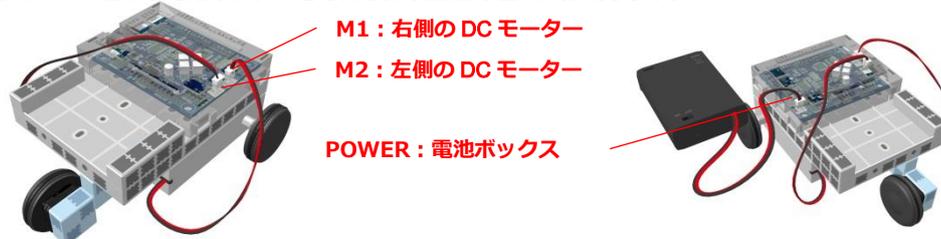
(2) DC モーターを基板台座の裏面に取り付けます。



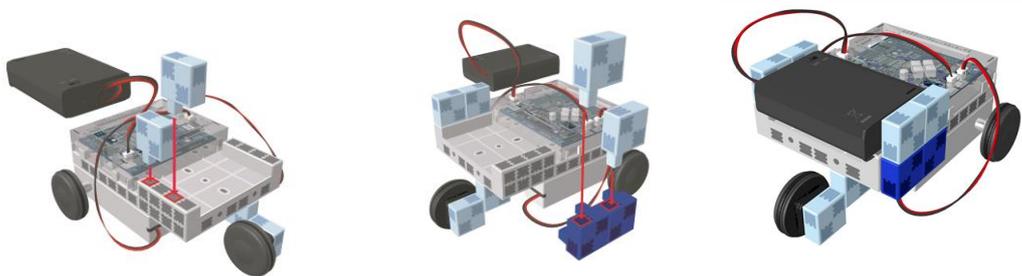
(3) ブロックで後輪を作ります。



(4) DC モーターおよび電池ボックスをそれぞれ Studuino 基板に接続します。



(5) 電池ボックスを基板台座に固定します。



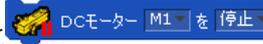
### 5.1.2. 入出力ポート情報の設定

ブロックプログラミング環境に Studuino 基板のポート情報を設定します。ブロックプログラミング環境のメニューバーの「編集」から「入出力設定...」を選択して、入出力設定ダイアログボックスを開きます。入出力設定ダイアログボックスの「DC モーター」エリアの M1、M2 がチェックされていることを確認してください。

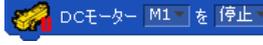


### 5.1.3. プログラミング

M1 に取り付けられた DC モーターを制御するプログラムを作成し、テストモードで動作を確認しながら DC モーターの設定と動きを確認します。

- ① 「動き」パレットから  ブロックと  ブロックと  ブロックをドラッグし、それぞれくっつけます。



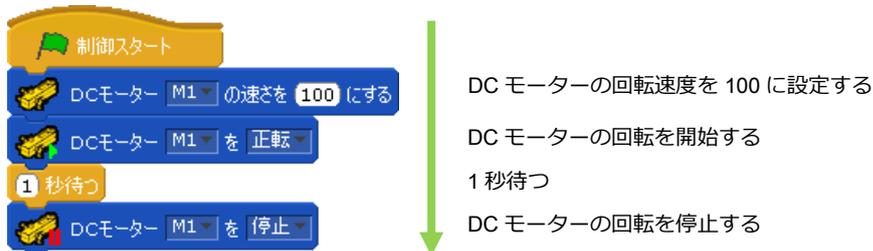
- ② 「制御」パレットから  ブロックを  ブロックと  ブロックの間にくっつけます。



- ③  ブロックにくっつけます。



以上で、M1 に接続した DC モーターを回転するプログラムができました。



④ DC モーターの動きを確かめます。Studuino 基板と PC が USB コードで接続されていることを確認し、メニューバーの「実行」から「テストモード開始」を選択してください。

⑤ テストモードに移行したら、をクリックしてください。



M1 に接続した DC モーターが回転して、1 秒後に停止します。

⑥ の速さを 100 にするブロックの値を 50 に設定し、をクリックしてください。



DC モーターの回転速度が遅くなります。

- ⑦ DCモーター M1 の速度を 50 にする ブロックの値を 100 に戻し、DCモーター M1 を 正転 を「逆転」に設定し、旗をクリックしてください。



DC モーターの回転方向が逆になります。

- ⑧ DCモーター M1 を 逆転 ブロックの値を「正転」に戻し、DCモーター M1 を 停止 を「解放」に設定し、旗をクリックしてください。



DC モーターが惰性で停止します。

※DC モーターの回転速度は 100 が最大となります。100 以上の数値を入力してもそれ以上速度は上がりません。

※DC モーターの回転速度を一定以上低く設定すると、DC モーターが回転しません。

設定数値は DC モーターへ流す電流の量を表しているため、DC モーターを動かすためには、DC モーターが回転するために必要な最低限の電流を流す必要があります。

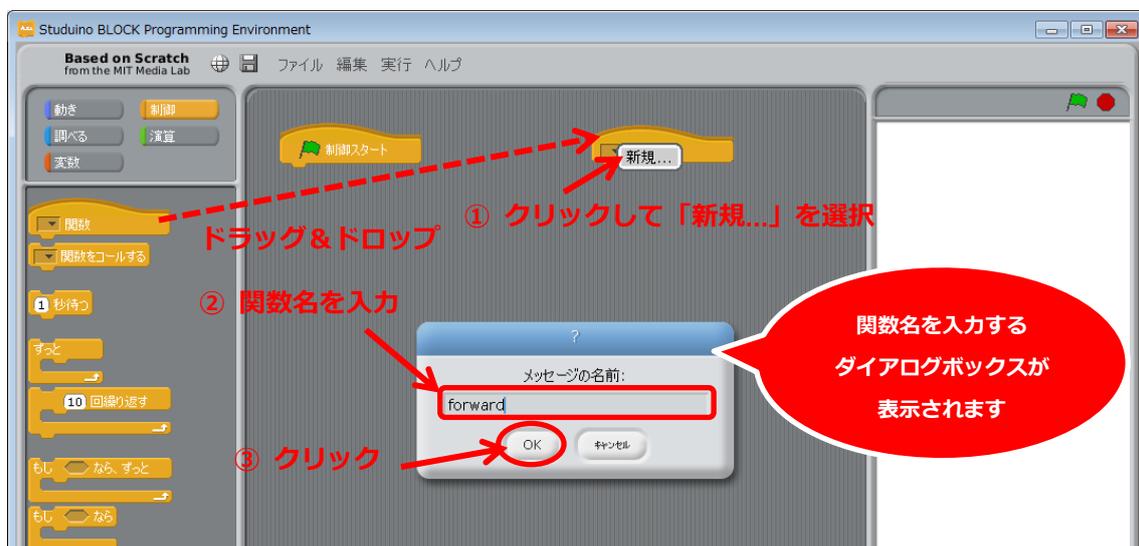
## 5.2. 車の制御

DC モーター 2つを使用して、車を前進、後退、回転するプログラムを作成します。

### 5.2.1. プログラミング

最初に車の前進プログラムを作成します。

- ① 「制御」パレットから「関数」ブロックをドラッグし、ブロックの▼をクリックし、「新規...」を選択します。ここでは表示されたダイアログボックスに関数名「forward」を設定し、OK をクリックします。



- ② 「動き」パレットから「DCモーター M1 の速さを 100 にする」ブロックと「DCモーター M1 を正転」ブロックと「DCモーター M1 を停止」ブロックをドラッグし、「関数」ブロックとそれをくっつけます。



- ③ 「制御」パレットから「1秒待つ」ブロックを「DCモーター M1」を「正転」ブロックと「DCモーター M1」を「停止」ブロックの間にくっつけます。



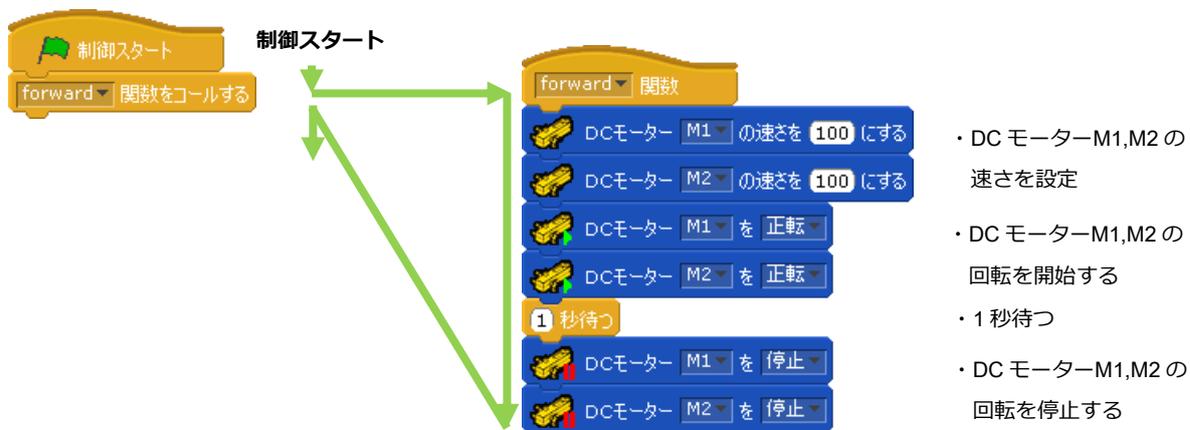
- ④ 「DCモーター M1」の速さを「100」にするブロックと「DCモーター M1」を「正転」ブロックと「DCモーター M1」を「停止」ブロックの M1 を M2 に変更します。



- ⑤ 「開放をコールする」ブロックをドラッグして「制御スタート」ブロックとくっつけます。



以上で、車を前進させるプログラムができました。



⑥ 車の動きを確かめます。Studuino 基板と PC が USB コードで接続されていることを確認し、メニューバーの「実行」から「テストモード開始」を選択してください。

⑦ テストモードに移行したら、をクリックしてください。



車が前進して、1 秒後に停止します。

⑧ をクリックしてテストモードを終了させてください。



次に、車の後退プログラムを作成します。

- ⑨ **forward** 関数 ブロック上で右クリックして、ブロックを複製します。



- ⑩ 複製したブロックの **forward** 関数 の▼をクリックし、「新規...」を選択します。表示されたダイアログボックスに関数名「backward」を設定し、OK をクリックします。



- ⑪ backward 関数の  ブロックと  ブロックを「逆転」に変更します。



- ⑫  ブロックの▼をクリックし、「backward」を選択します。



以上で、車を後退させるプログラムができました。

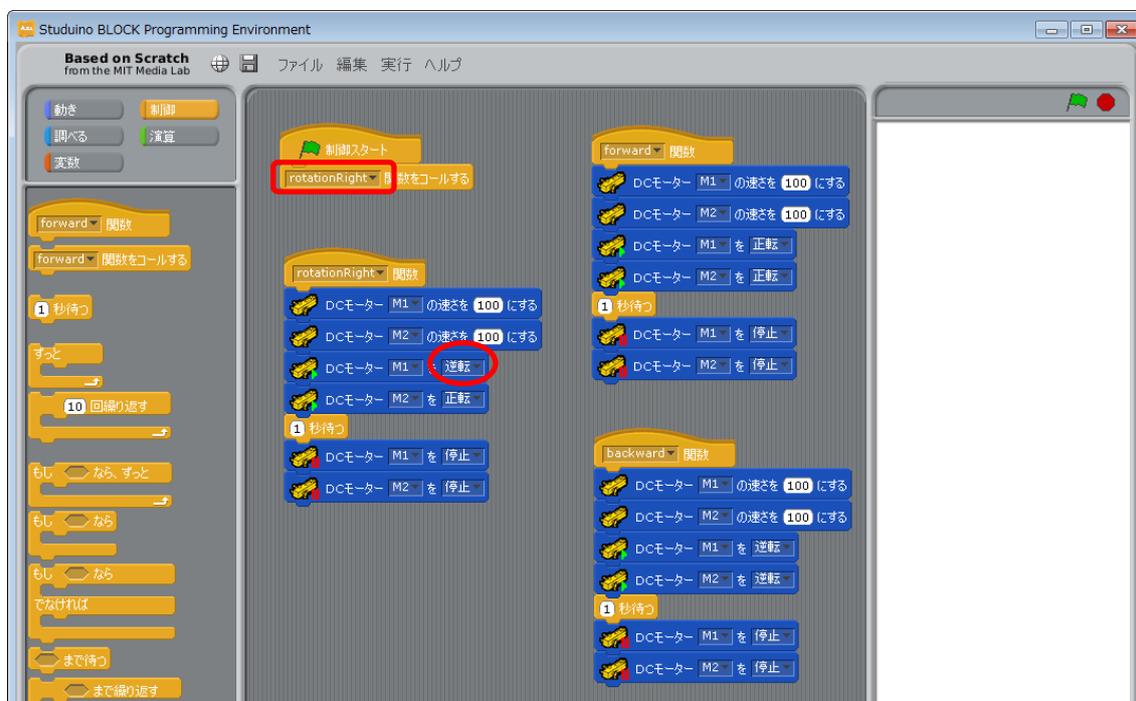
手順の⑥、⑦を行い、車が後退して、1秒後に停止することを確認してください。

次に、車を右に回転するプログラムを作成します。

⑬ 手順の⑨、⑩を行い、rotationRight 関数を作成してください。



⑭ rotationRight 関数の 1 番目の DCモーター M1 を 正転 ブロックを「逆転」に変更し、backward 関数をコールする ブロックの▼をクリックし、「rotationRight」を選択します。



以上で、車を右に回転させるプログラムができました。

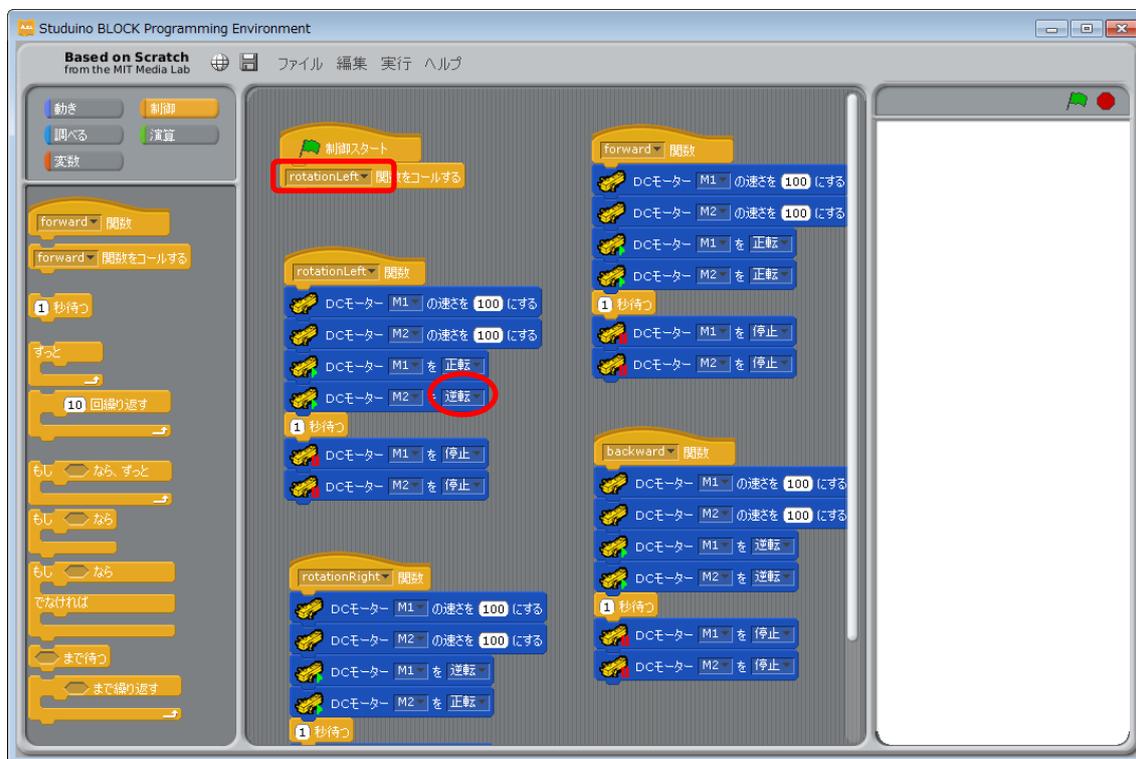
手順の⑥、⑦を行い、車が右に回転して、1 秒後に停止することを確認してください。

次に、車を左に回転するプログラムを作成します。

⑮ 手順の⑨、⑩を行い、rotationLeft 関数を作成してください。



⑯ rotationLeft 関数の 2 番目の DCモーター M1 を正転 ブロックを「逆転」に変更し、rotationRight 関数をコールする ブロックの▼をクリックし、「rotationLeft」を選択します。



以上で、車を左に回転させるプログラムができました。

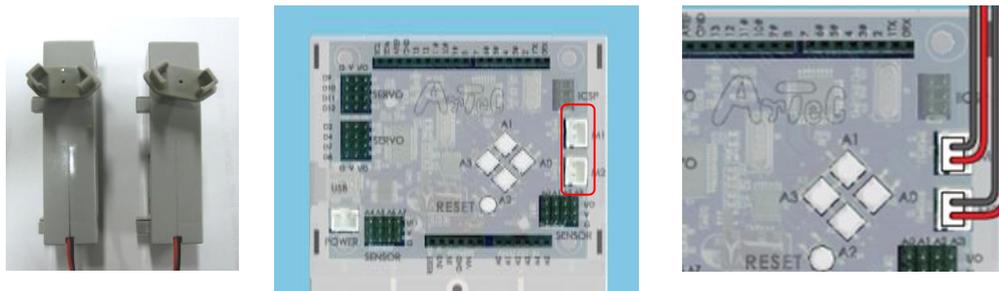
手順の⑥、⑦を行い、車が左に回転して、1 秒後に停止することを確認してください。

## 5.2.2. DC モーター校正

DC モーターは個体差により同じ設定でも回転速度のズレが生じます。DC モーター 2 つで移動する車を作った時、前進の際にまっすぐ走らない場合、DC モーター校正によりこのズレを補正する必要があります。

### ① Studuino 基板の DC モーター用コネクタに DC モーターを接続します。

DC モーターを 2 つ並べ、回転速度の違いを確認できるようにします。



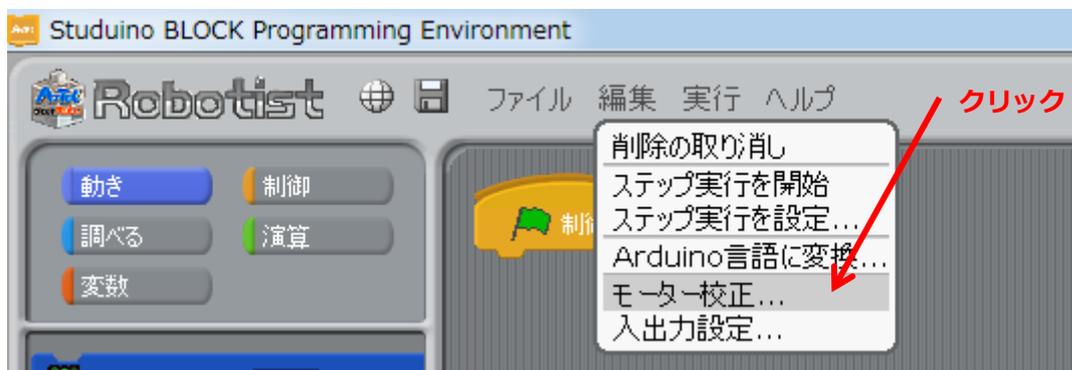
### ② 入出力設定を行います。

入出力設定画面で DC モーターの M1、M2 にチェックを入れます。



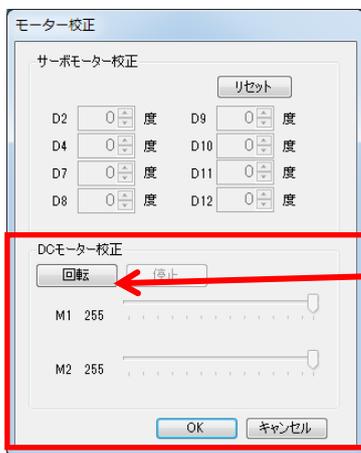
### ③ メニューバーの「編集」から「モーター校正...」を選択します。

基板に USB ケーブルが接続されていることを確認してください。



モーター校正を選択すると、下図のようなウインドウが開きます。ここでは、下側のDCモーター校正を使用します。

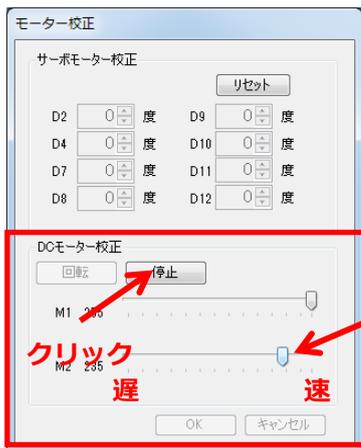
④ 回転ボタンを押し、モーターを回転させます。



回転ボタンをクリックすると、2つのモーターが最大速度で回転し始めます。

クリック

⑤ スライダーで速度の調整を行います。



回転速度の速い方のスライダーを調整し、もう一方と回転がそろるようにします。

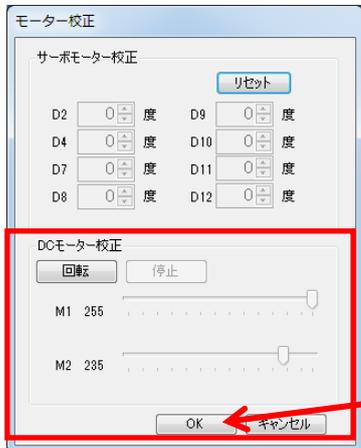
完了したら、停止ボタンをクリックし、回転を止めます。

スライダーで速度を調整

クリック  
遅

速

⑥ OK ボタンをクリックし、校正を完了します。



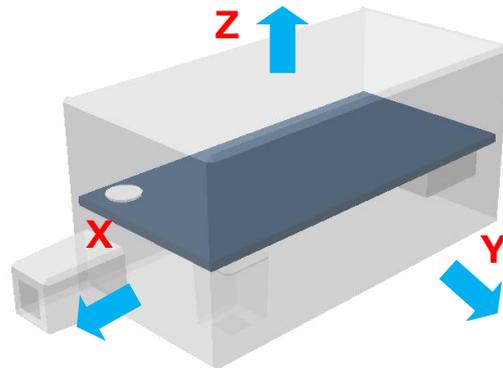
OK ボタンをクリックし、設定を反映させます。

クリック

### 5.3. 加速度センサーを使用した車の制御

5.2 車の制御で作成したプログラムを使って、加速度センサーを利用した車の制御プログラムを作成します。

加速度センサーとは、加速度（速度が一定時間あたりになしてどれだけ変化したか）を計測するセンサーであり、X,Y,Zのそれぞれの方向の加速度を読み取ります。



また、加速度センサーは下図のようにセンサーの向きを変えると、それぞれの方向の値が変化します。

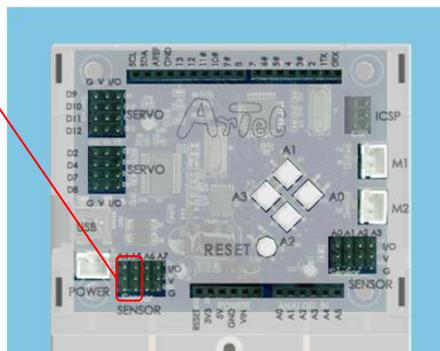
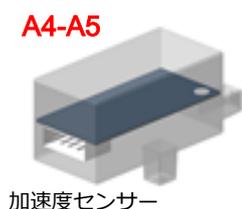


加速度センサーを固定した状態でも加速度が0にならない方向があるのは、重力加速度<sup>※</sup>を検出しているためです。この性質を利用して、地面に対してどのくらい傾いているかを知ることができます。

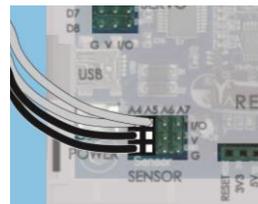
※重力加速度とは重力の向きに常に働く加速度です。

### 5.3.1. Studuino 基板と加速度センサーの接続

Studuino 基板のセンサー/LED/ブザー用コネクタの A4、A5 に加速度センサーを取り付けます。



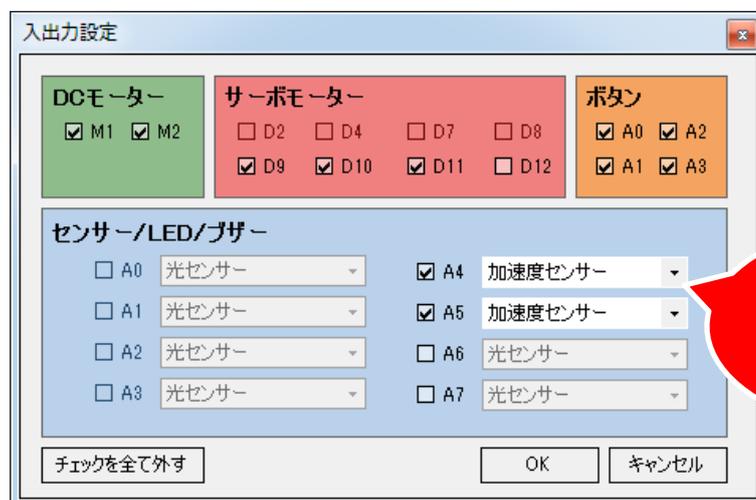
加速度センサーは4本線のコードを使用して、A4 と A5 にまたがって接続する必要があります。



センサー接続コードの向きに注意！  
上側に灰色のコードがくるように。

### 5.3.2. 入出力ポート情報の設定

ブロックプログラミング環境に Studuino 基板のポート情報を設定します。ブロックプログラミング環境のメニューバーの「編集」から「入出力設定...」を選択して、入出力設定ダイアログボックスを開きます。入出力設定ダイアログボックスの「センサー/LED/ブザー」エリアの A4、A5 をチェックして加速度センサーを選択して下さい。



A4、A5 をチェックし  
加速度センサーを  
選択してください

### 5.3.3. 加速度センサーの動作確認

加速度センサーの値をチェックします。Studuino 基板と PC を USB コードで接続し、ブロックプログラミング環境のメニューバーの「実行」から「テストモード開始」を選択してください。テストモードが開始されるとセンサー・ボードが表示され、「[A4/A5] 加速度センサー」が表示されます。

- ・ 加速度センサーの値は、0~100 で表します。
- ・ X 方向（前ページの図参照）を水平に置いた場合、加速度センサー（X）の値は 50 になります。
- ・ X 方向が上向きになるように傾けるにつれて、加速度センサー（X）の値は増え、垂直になったとき、75 となります。
- ・ 反対に、X 方向が下向きになるように傾けるにつれて、加速度センサー（X）の値は減り、垂直になったとき、25 となります。
- ・ 重力加速度以上の加速度を与えたとき（加速度センサーを X 方向に動かしたとき）は、さらに値の変化は大きくなります。

※Y、Z 方向についても同じです。

センサー・ボード	
[A0] ボタン	1
[A1] ボタン	1
[A2] ボタン	1
[A3] ボタン	1
[A4/A5] 加速度センサー(X)	45
[A4/A5] 加速度センサー(Y)	73
[A4/A5] 加速度センサー(Z)	58
[A6] 未接続	0
[A7] 未接続	0

傾きを変えたり  
動かしたりすることで  
値が変化します

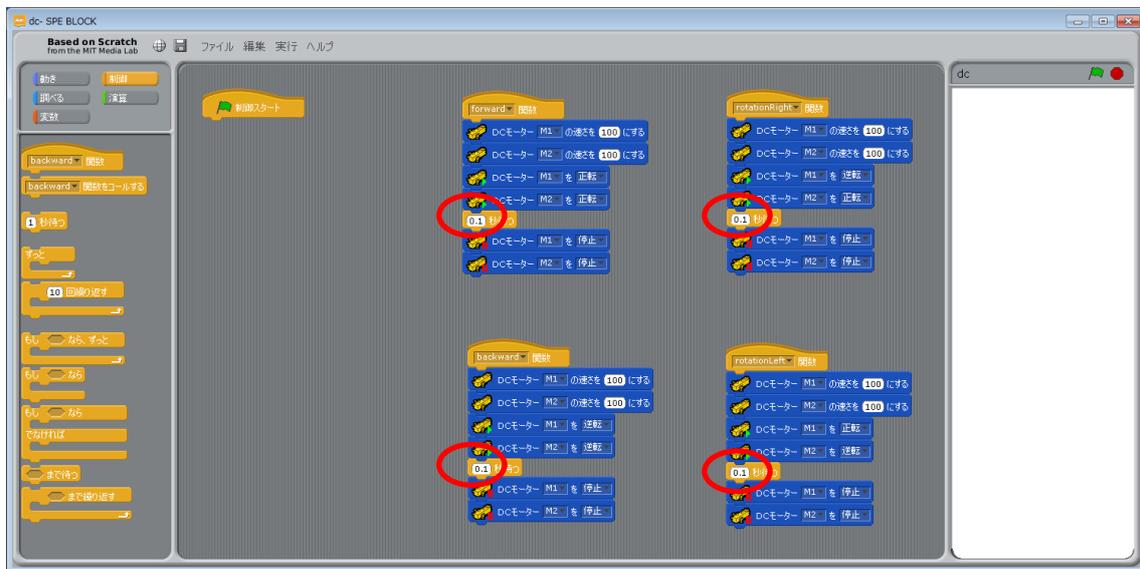
確認ができれば、メニューバーの「実行」から「テストモード終了」を選択し、テストモードを終了してください。

### 5.3.4. プログラミング

5.2.1 で作成した車を前進・後退・回転するプログラムを使用します。



① **1 秒待つ** ブロックの設定を「0.1」に変更します。



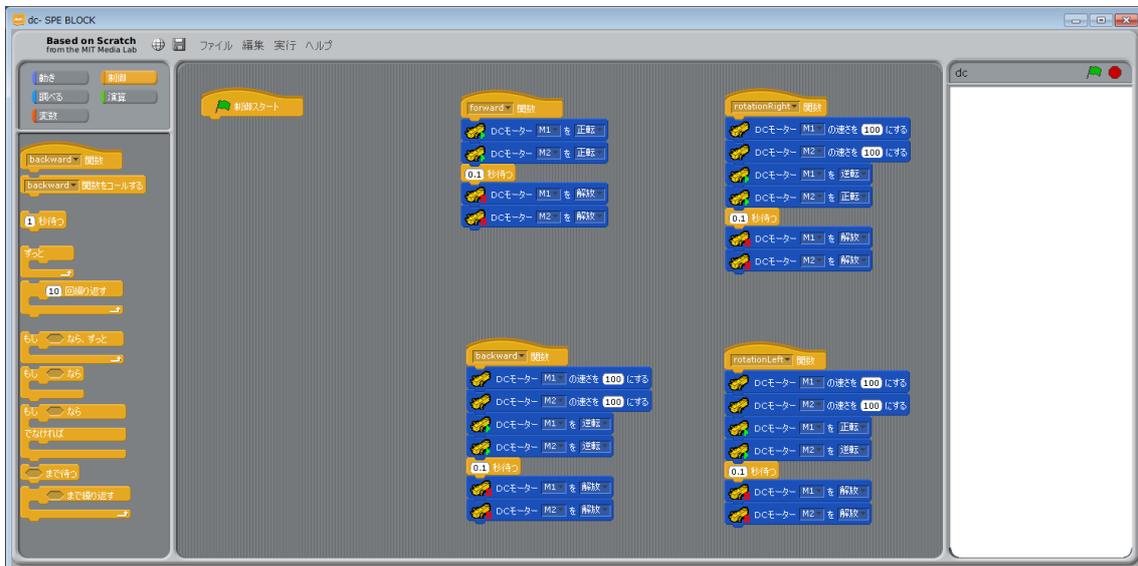
- ②  ブロックと  ブロックの設定を「解放」に変更します。



- ③ forward 関数の  ブロックと  ブロックを切り離し、 をブロックパレットにドラッグして削除します。



- ④ forward 関数と DCモーター M1 を 正転 ブロックをくっつけます。



- ⑤ 他の関数も手順③、④と同様の手順で DCモーター M2 の速さを 100 にする を削除します。



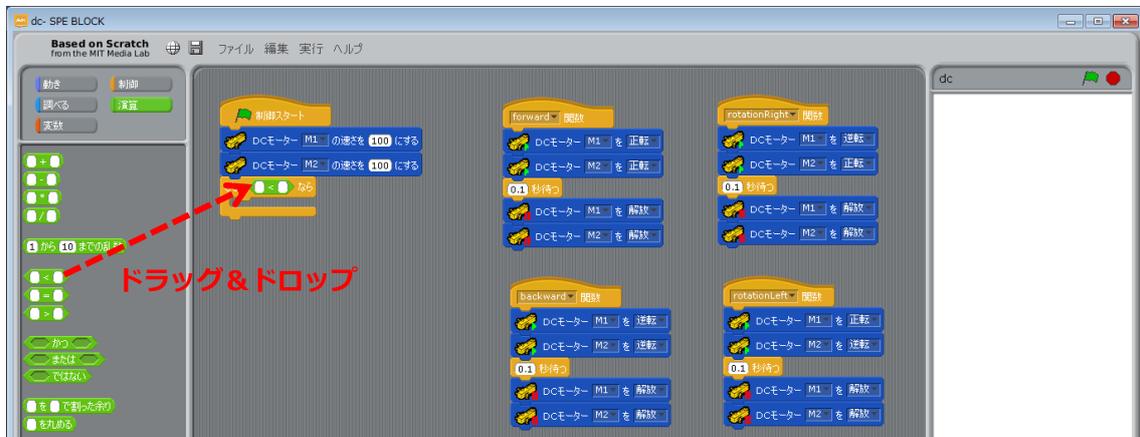
- ⑥ 「動き」パレットから  DCモーター M1 の速さを 100 にする ブロックを 2 つドラッグし、  
 制御スタート ブロックにくっつけ、2 番目のブロックの設定を「M2」に設定します。



- ⑦ 「制御」パレットから  もし...なら ブロックをくっつけます。



- ⑧ 「演算」パレットから  < ブロックをくっつけます。



- ⑨ 「調べる」パレットから「加速度センサー」の値ブロックを「<」ブロックの左辺にくっつけ、右辺に 40 を設定します。



- ⑩ 「制御」パレットの「関数をコールする」ブロックを「もし 加速度センサー の値 < 40 なら」ブロックの開いているところに入れ、forward 関数を選択します。



- ⑪ 「もし 加速度センサー の値 < 40 なら」ブロックで右クリックし、複製を選択します。



- ⑫ 複製したブロックを複製元のブロックにくっつけます。



- ⑬  ブロック<記号上で右クリックし、>を選択します。



- ⑭  ブロックの右辺を 60 に設定し、 ブロックで backward 関数を選択します。



- ⑮ もし **加速度センサー X の値 < 40** なら  
 ブロックで右クリックし、複製を選択します。



- ⑯ 複製したブロックを複製元のブロックにくっつけます。



- ⑰ 複製した **加速度センサー X の値** ブロックに Y を、**関数をコールする** ブロックに rotationLeft 関数と rotationRight 関数を設定します。



- ⑱  ブロックをドラッグし、 以下のブロックを囲みます。



以上で、加速度センサーを使って車を制御するプログラムができました。





DC モーターM1,M2 の速さを 100 に設定

- ・ 加速度センサーを前に傾けた場合 forward 関数(前進処理)を実行
- ・ 加速度センサーを後に傾けた場合 backward 関数(後退処理)を実行
- ・ 加速度センサーを左に傾けた場合 rotationLeft 関数(左回転処理)を実行
- ・ 加速度センサーを右に傾けた場合 rotationRight 関数(右回転処理)を実行

⑰ 加速度センサーの傾きと車の動きを確かめます。Studuino と PC が USB コードで接続されていることを確認し、電池ボックスを ON にして、メニューバーの「実行」から「テストモード開始」を選択してください。

⑱ テストモードに移行したら、をクリックしてください。



加速度センサーの傾きに反応して車が動きます。

- ⑱ テストモードで動作が確認できたら、メニューバーの「実行」から「プログラム作成・転送」を選択してください。

※テストモードでは、一つ一つのプログラムの処理に時間がかかるため、車はスムーズに動きません。プログラム転送後はスムーズに動くようになります。

## 6. サーボモーターを動かす

ここでは、サーボモーターを使用する基本的なプログラムの作成を通して、サーボモーターの動かし方、変数ブロックと演算ブロックの使い方を説明します。

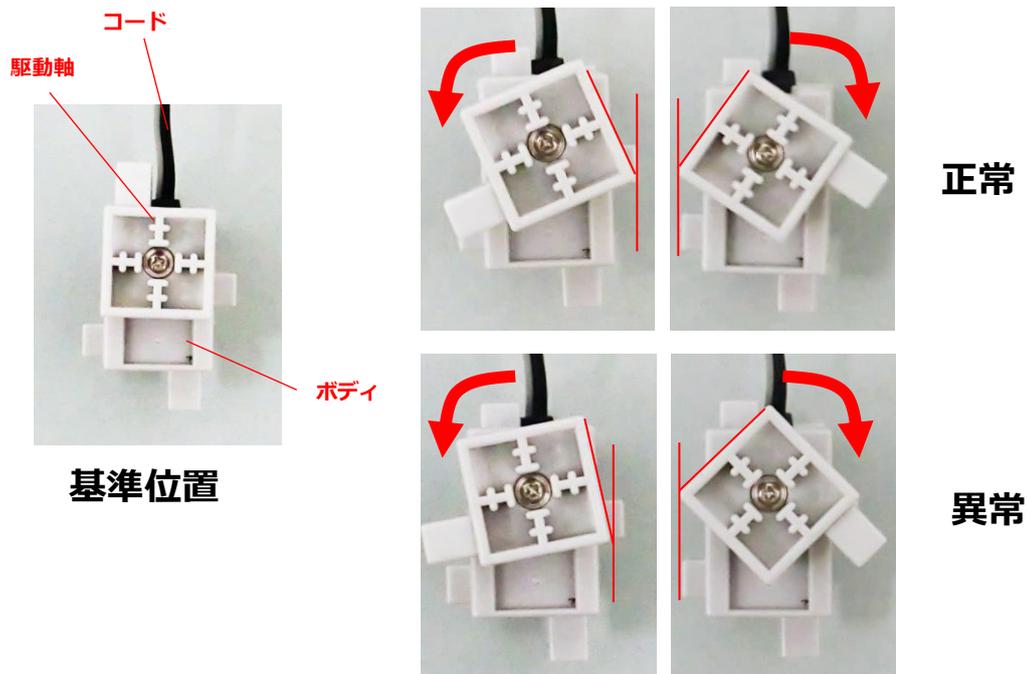
### 6.1. サーボモーターの角度校正

サーボモーターは個体差により同じ角度設定にしても数度のズレが生じます。サーボモーター角度校正により、このズレを補正する必要があります。

#### 6.1.1. サーボモーターの駆動軸角度の調整

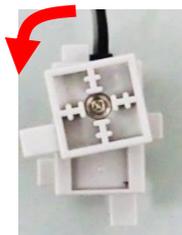
サーボモーターを取り付ける前に、サーボモーターの駆動軸が正常に取り付けられているか、以下の方法で確認してください。

駆動軸を基準位置の状態から、左右に止まる位置まで回転させたときのボディとの角度が、左右で大きく異なる場合は、駆動軸のブロック部分がスリップして正常な位置からずれてしまっています。



これは駆動軸に大きな力がかかった時、内部のギヤが破損しないように、一定の力でブロック部分のみがスリップして回転することで、内部のギヤの破損を防ぐ仕組みになっているため起こります。

駆動軸のブロック部分をずれている方向と反対方向に手で回転させると、カチッと音が鳴りズレが元に戻ります。



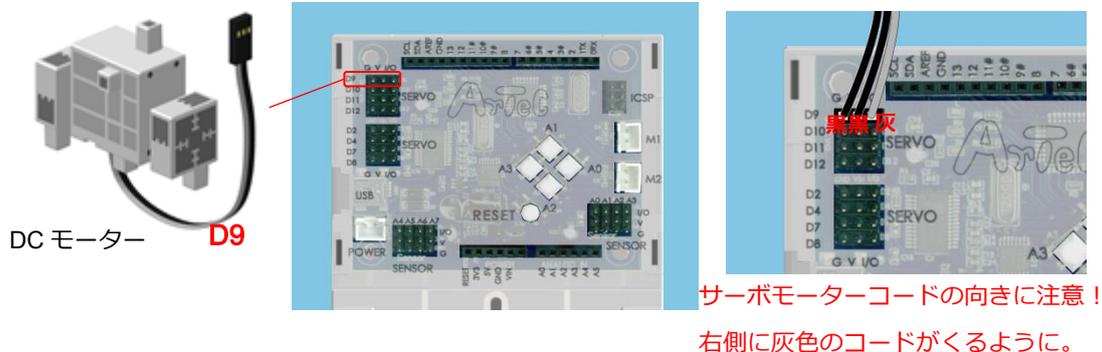
※むやみにスリップさせないでください。

サーボモーターの劣化、破損の原因となります。

※微小な角度のズレは、後のソフトウェアでの角度校正で補正できます。

### 6.1.2. Studuino 基板とサーボモーターの接続

Studuino 基板のサーボモーター用コネクタの D9 にサーボモーターを接続します。



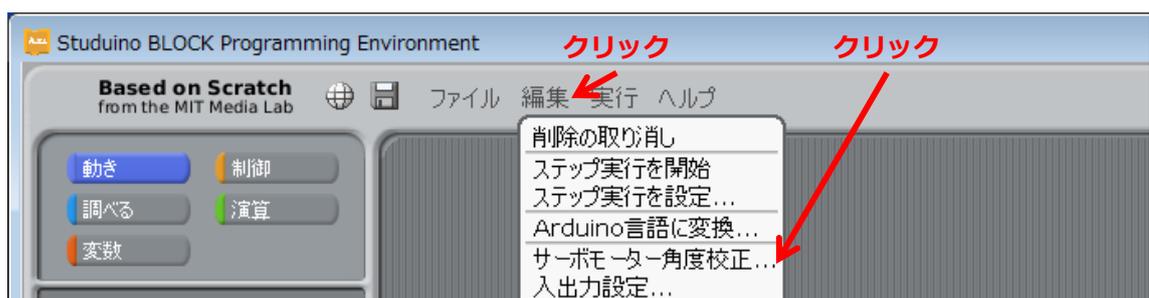
### 6.1.3. 入出力ポート情報の設定

ブロックプログラミング環境に Studuino 基板のポート情報を設定します。ブロックプログラミング環境のメニューバーの「編集」から「入出力設定...」を選択して、入出力設定ダイアログボックスを開きます。入出力設定ダイアログボックスの「サーボモーター」エリアで、D9 のみチェックしてください。



- ① メニューバーの「編集」を選択し、表示されたウィンドウから「サーボモーター角度校正」を選択します。

基板に USB ケーブルが接続されていることを確認してください。



サーボモーター角度校正を選択すると、接続されているサーボモーターが全て 90 度の位置で固定されると同時に、自動的にテストモードとなり、下図のようなウインドウが開きます。

※基板に電池ボックスをつなぎ、電源を ON にした状態で確認してください。

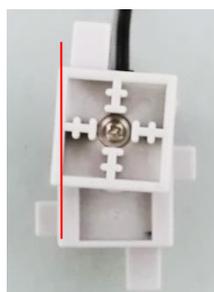


90 度の位置

駆動軸とボディが平行

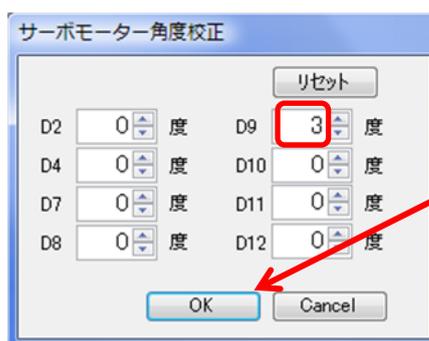


- ② この時、90 度の位置から少しずれている場合は、サーボモーター角度校正画面に数値を入力することで、サーボモーターの角度を微調整することができます。90 度の位置になるように数値を入力してください。



調整必要

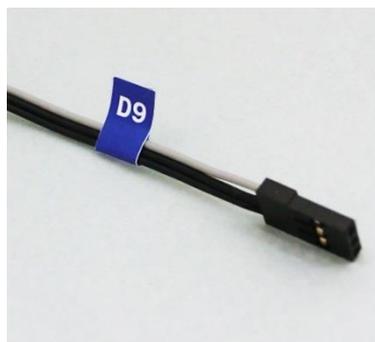
駆動軸とボディがずれている



クリック

※校正の際に取り付けたコネクタに別のサーボモーターに付け替えた場合は、再度サーボモーター角度校正が必要です。

角度校正を終えたサーボモーターには、サーボモーター用コネクタの番号と同じ番号のシールを貼りつけて、識別できるようにしてください。



## 6.1.4. プログラミング

- ① 「動き」パレットから  ブロックを  ブロックにくっつけます。



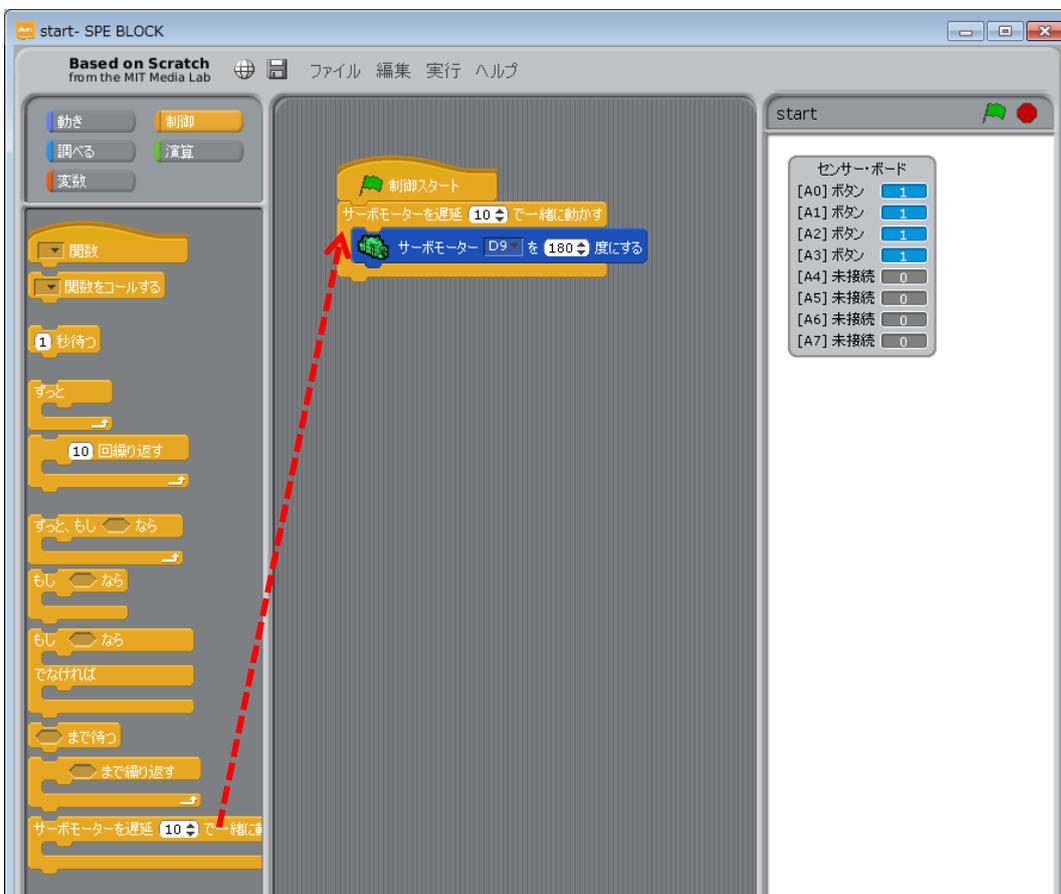
- ②  ブロックの「90」に「0」を入力します。テストモードで動きを確認します。Studuino と PC が USB コードで接続されていることを確認し、メニューバーの「実行」から「テストモード開始」を選択してください。
- ③ テストモードに移行したら、 をクリックしてください。



サーボモーターが 0 度に動くことを確認します。

- ④ 同様に  ブロックの「0」に「180」を入力し、 をクリックしてサーボモーターが 180 度に動くことを確認します。

- ⑤  ブロックで  ブロックを囲みます。



- ⑥  ブロックの  ブロックの「180」に

「0」を入力し、 をクリックしてサーボモーターが0度にゆっくり動くことを確認し

ます。  ブロックは複数のサーボモーターを囲むことができ

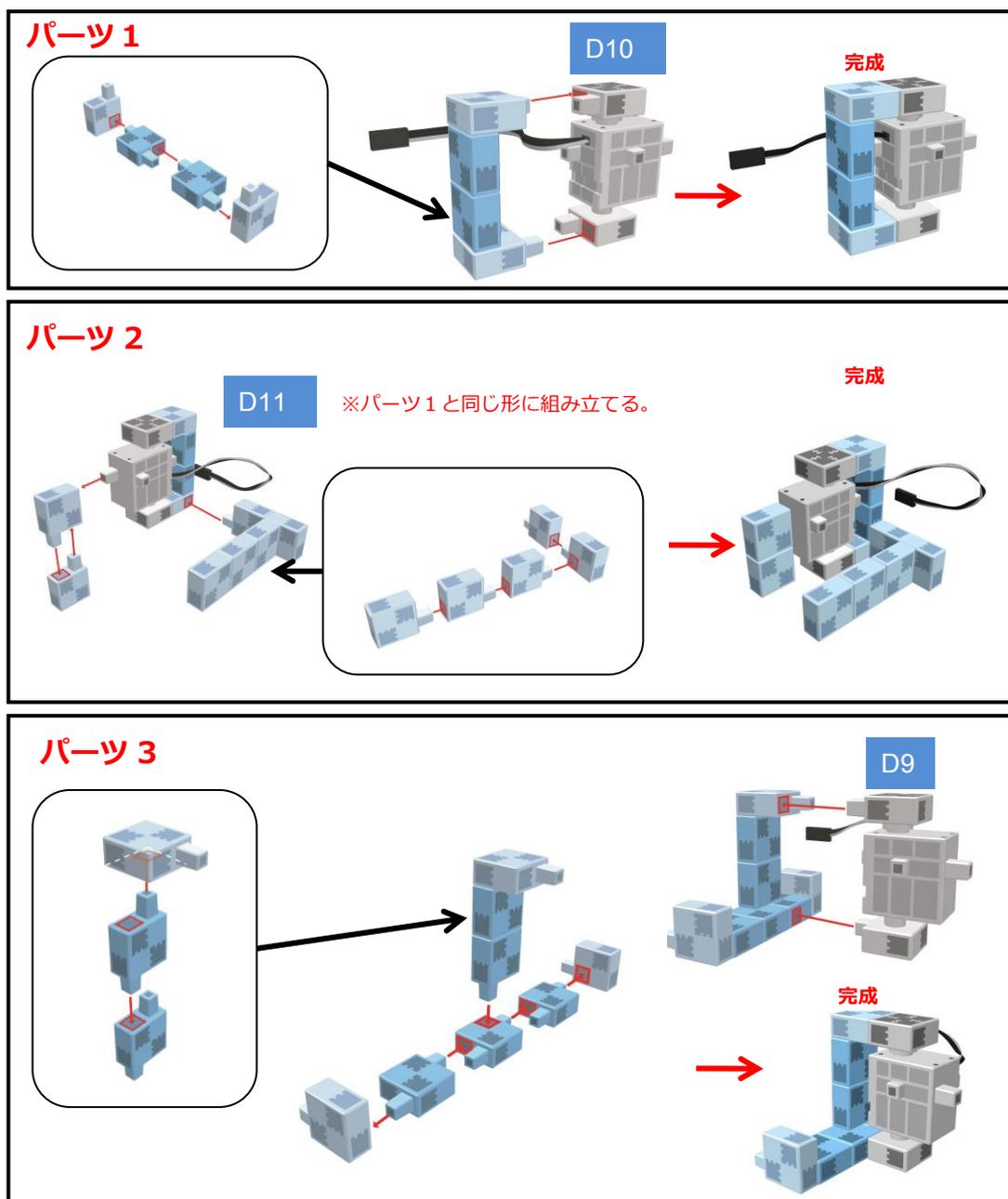
ます。このブロックで囲まれたサーボモーターブロックのサーボモーターは同時に遅延時間をつけて目的角度へ移動します。このブロックを使用してサーボモーターを制御する場合、サーボモーターの移動中は他の処理が実行できません。

## 6.2. サーボモーター3つでアームロボをつくる

サーボモーターを3つ使い、物をつかむアームロボを作成します。基板についているスイッチも使用して、ボタンを押す度に角度が変わるアームを作成します。変数ブロック、演算ブロック、ボタンプロックを用いてボタンを押すごとに状態が変わる処理を作成します。

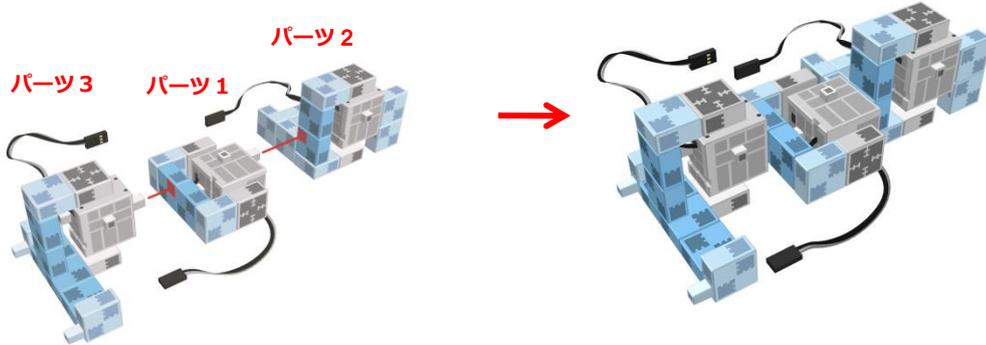
### 6.2.1. ロボットの組み立て

以下の手順でアームロボットを組み立てます。

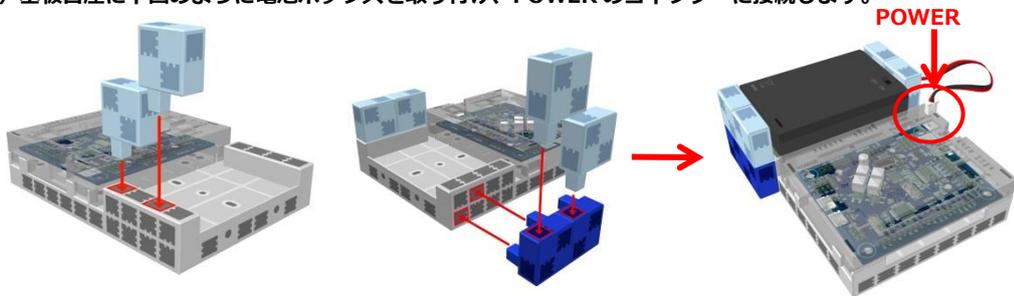


## アームの組立

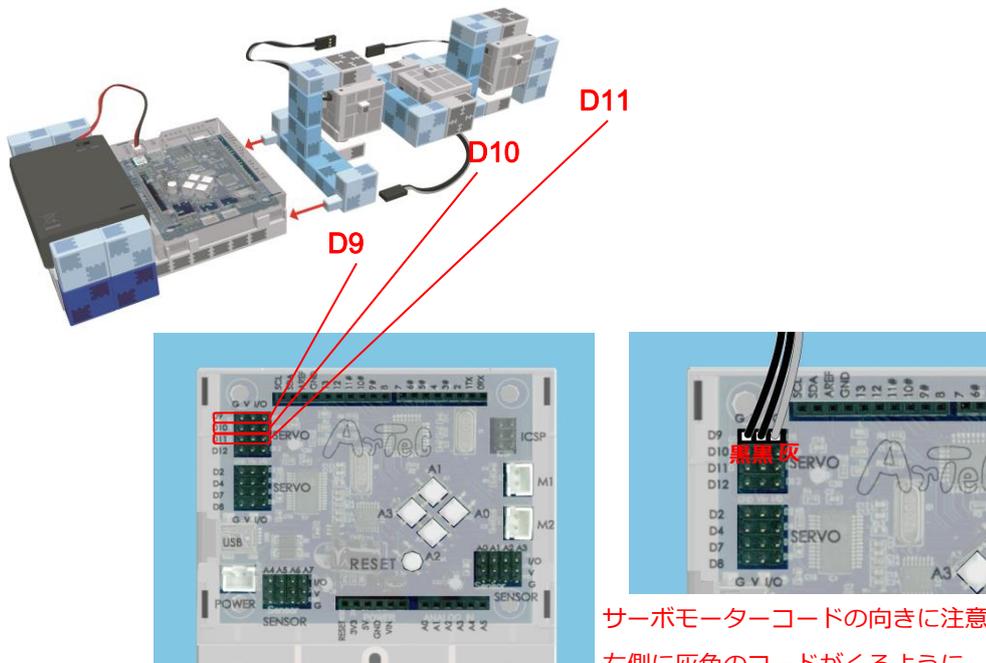
(1) パーツ 1～3 を下図のように組み立てます。



(2) 基板台座に下図のように電池ボックスを取り付け、POWER のコネクタに接続します。



(3) アームを取り付け、それぞれのサーボモーターをコネクタに接続します。



### 6.2.2. 入出力ポート情報の設定

ブロックプログラミング環境に Studuino 基板のポート情報を設定します。ブロックプログラミング環境のメニューバーの「編集」から「入出力設定...」を選択して、入出力設定ダイアログボックスを開きます。入出力設定ダイアログボックスの「サーボモーター」エリアで、D9、D10、D11 がチェックされていることを確認してください。



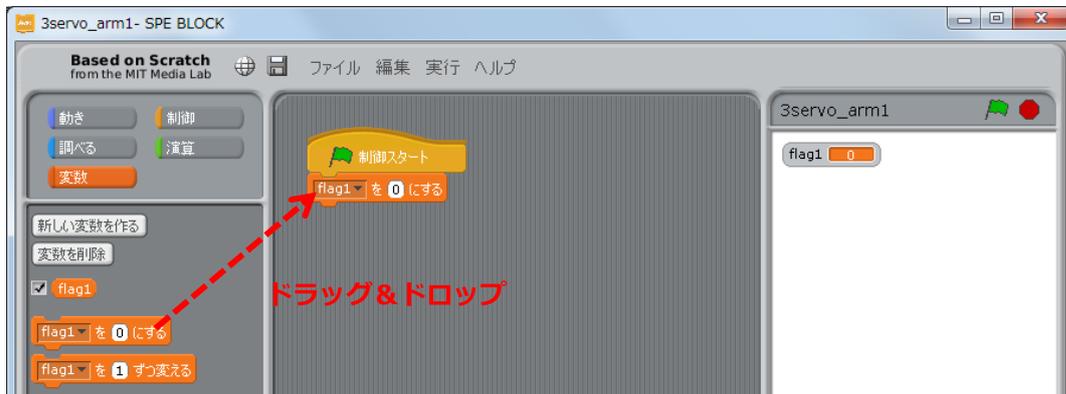
### 6.2.3. プログラミング

サーボモーター 1 つで動きを確認します。

- ① 「変数」パレットで「新しい変数を作る」をクリックし、表示される変数名設定ダイアログで変数「flag1」を作成します。

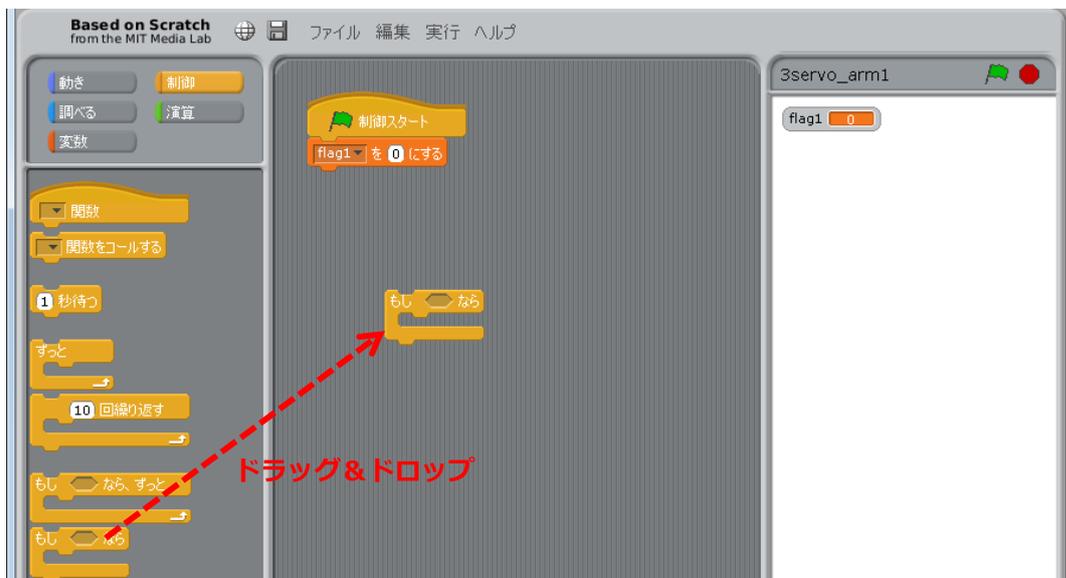


② 「変数」パレットから「flag1」を「0」にするブロックを「制御スタート」ブロックにくっつけます。



③ プッシュスイッチ A0 が押される度に flag1 の値を変える処理部分を作成します。「制御」

パレットから「もし」ブロックをドラッグします。



④ 「演算」パレットから「=」ブロックをくっつけます。



- ⑤ 「調べる」パレットから「ボタン A0 の値」ブロックを「=」ブロックの左辺にくっつけ、右辺に 0 を設定します。



- ⑥ 「変数」パレットから「flag1 を 0 にする」ブロックを「もし」ブロックの開いているところに入れます。



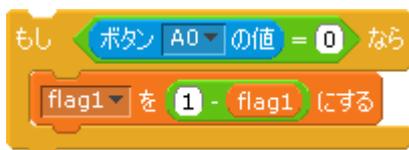
- ⑦ 「演算」パレットから「-」ブロックをくっつけます。



- ⑧ 「変数」パレットから flag1 ブロックを 0-0 ブロックの右辺にくっつけ、左辺に 1 を設定します。

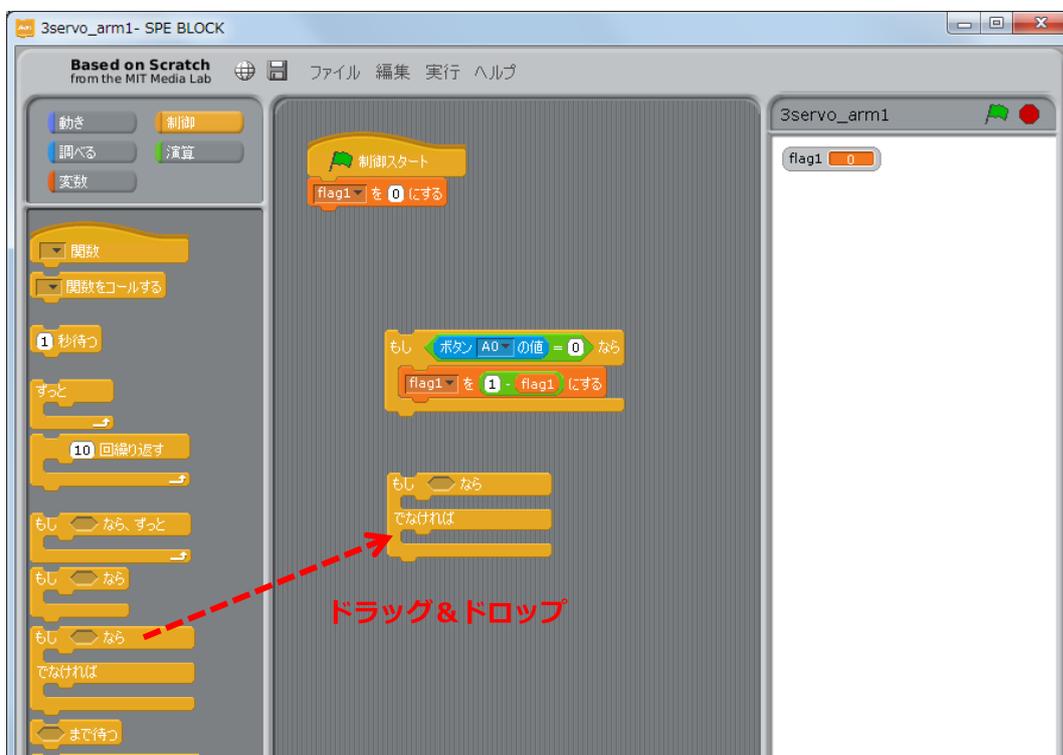


以上で、プッシュスイッチ A0 が押された場合に flag1 の値を切り替える処理部分が完成しました。このブロックでは、プッシュスイッチ A0 が押されたとき、flag1 の値が 0 であれば 1 に、1 であれば 0 になります。

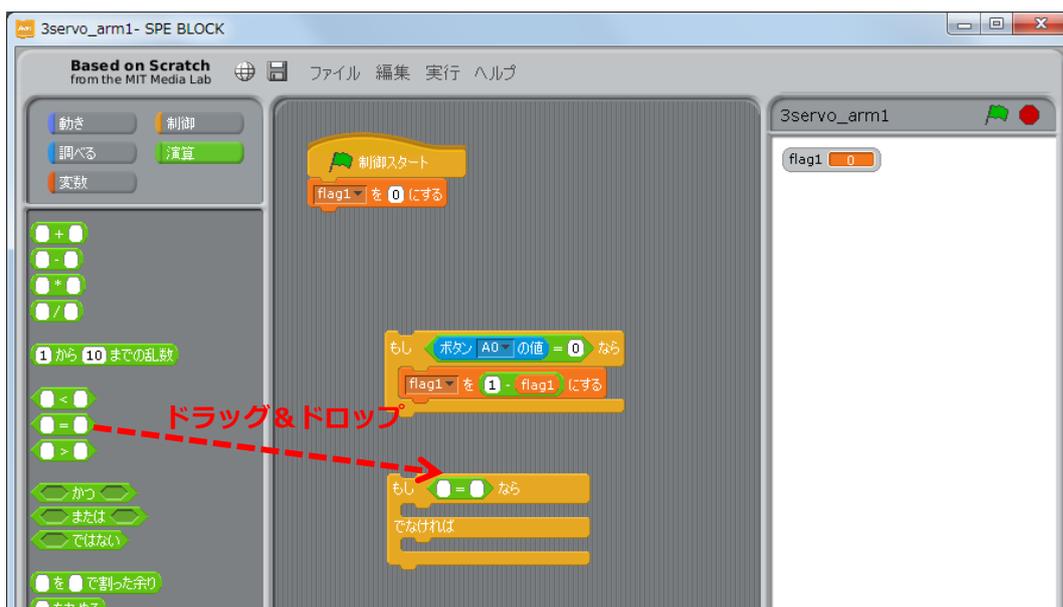


⑨ flag1 の値に応じてサーボモーターの角度を変える処理を作成します。「制御」パレット

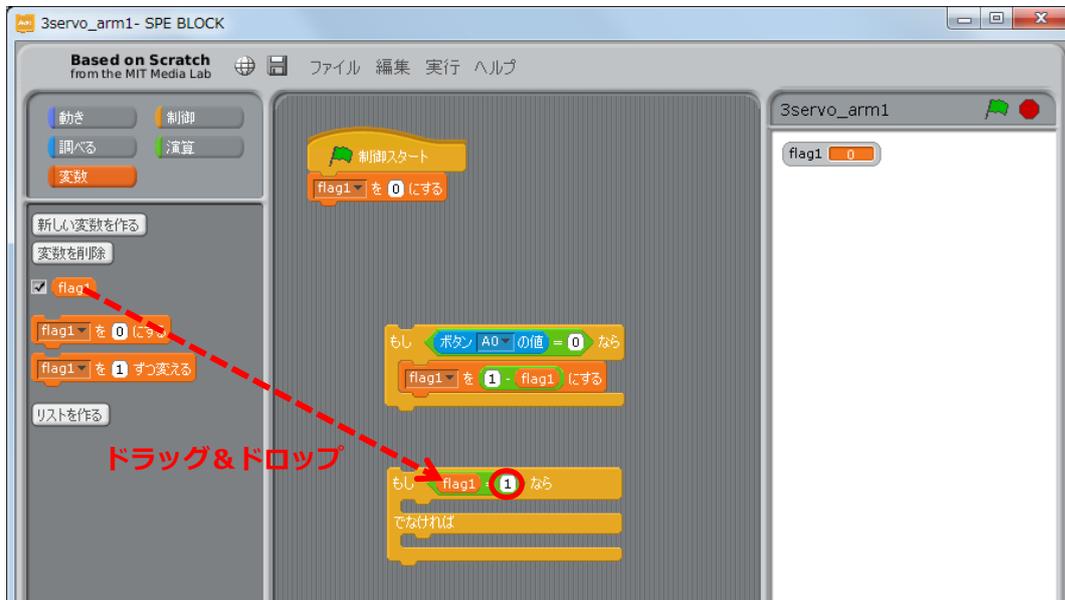
もし なら  
でなければ  
から ブロックをドラッグします。



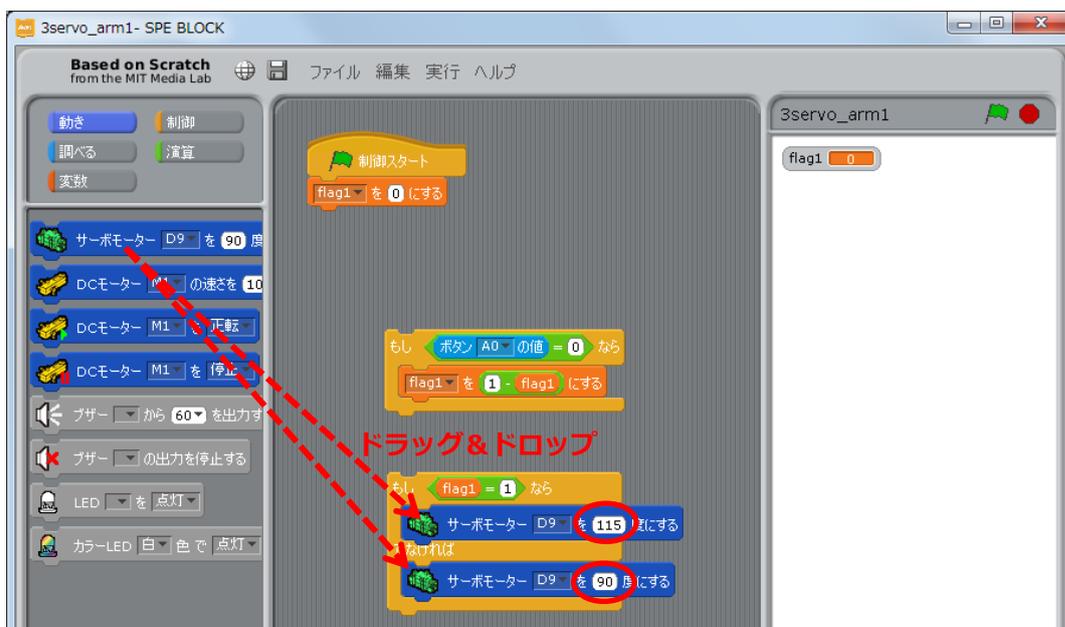
⑩ 「演算」パレットから  $0=0$  ブロックをくっつけます。



- ⑪ 「変数」パレットから flag1 ブロックを =0= ブロックの左辺にくっつけ、右辺に 1 を設定します。

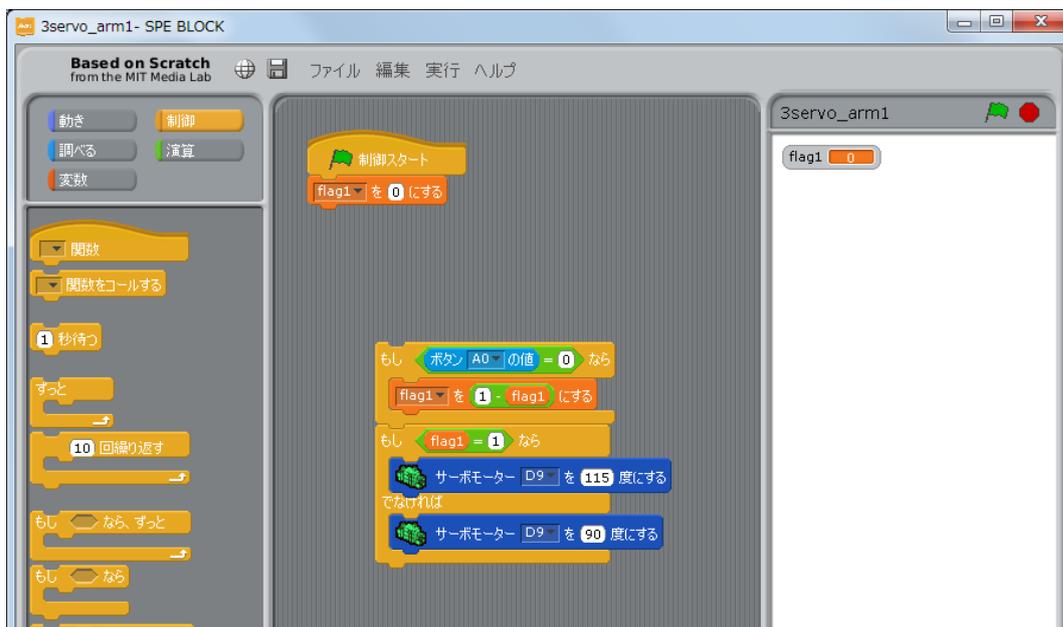


- ⑫ 「動き」パレットから サーボモーター D9 を 90 度にする ブロックをドラッグし、  
 ブロックの開いているところにくっつけます。くっつけた上下のブロックの値部分にそれぞれ 115, 90 を入力します。



以上で flag1 の値に応じてサーボモーターの角度を変える処理部分が完成しました。

⑬ ⑫で作成したブロックとくっつけます。

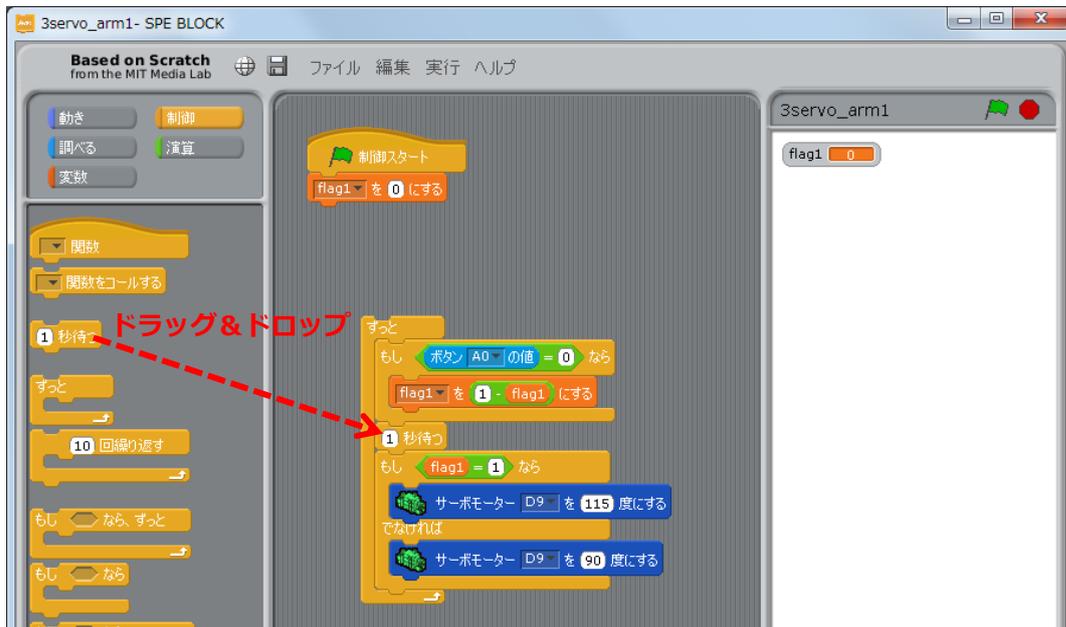


⑭ 「制御」パレットから「ずっと」ブロックをドラッグしてブロックを囲みます。

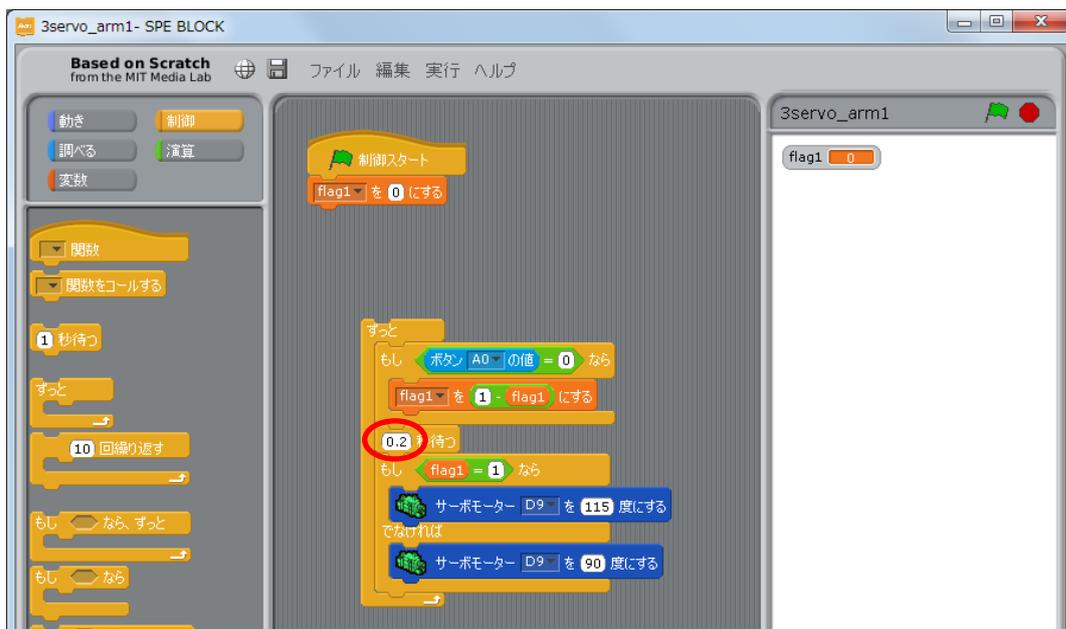


これまでのステップで、プッシュスイッチを押す度にサーボモーターの角度を変える処理が出来あがりましたが、このループの中で使われている処理は、すべて命令を与えた後即座に次のステップに移ります。そのため、一度ボタンを押している間に何度もループを回り、その都度 flag1 の値が切り替わり、違う角度が設定されてしまいます。これを回避するため、次のステップで繰り返しの中に適当な待ち時間を設けます。この例では 0.2 秒とします。

- ⑮ 「制御」パレットから「1秒待つ」ブロックをドラッグし、図の位置にくっつけます。



- ⑯ 「1秒待つ」ブロックに0.2を設定します。



これにより、0.2秒ごとにボタンの状態を判定するようになりました。

- ⑰ ブロックを **flag1 を 0 にする** ブロックにくっつけます。



以上で、プッシュスイッチ A0 を押すごとに D9 につながれたサーボモーターの角度を 115 度、90 度と変えるプログラムが作成できました。

- ⑱ メニューバーの「実行」から「プログラム作成・転送」を選択して、実際の動きを確認しましょう

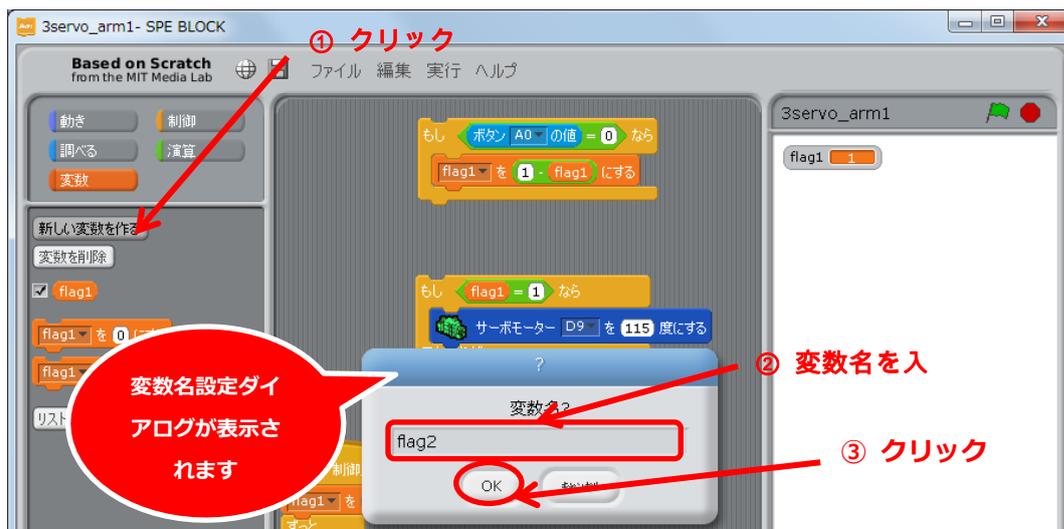
- ⑱ 残り2つのサーボモーターについても同様にプログラムを作成していきます。出来上がったプログラムのかたまりは下にずらしておき、もし「ボタン A0」の値 = 0 なら ブロックを右クリックして複製を作ります。



- ② 複製したブロックを分解し、**1 秒待つ** ブロックをコマンドパレットにドラッグ&ドロップします。



- ② 「変数」パレットで「新しい変数を作る」をクリックし、表示される変数名設定ダイアログで変数「flag2」を作成します。



- ② 複製したブロックの「flag1」を「flag2」に置き換えます。



- ⑳ **flag2** ブロックを **0-0** ブロックの右側にドラッグします。もともとあった ブロックが飛び出すので、右クリックで削除します。下側の **0-0** ブロックも同様にドラッグし、飛び出した ブロックを削除します。



- ㉑ **ボタン A0 の値** ブロックに A1 を選択します。さらに **サーボモーター D9 を 90 度にする** ブロックに D10 を選択します。



㊸ 出来上がった2つのブロックをさらに複製し、複製元とくっつけます。



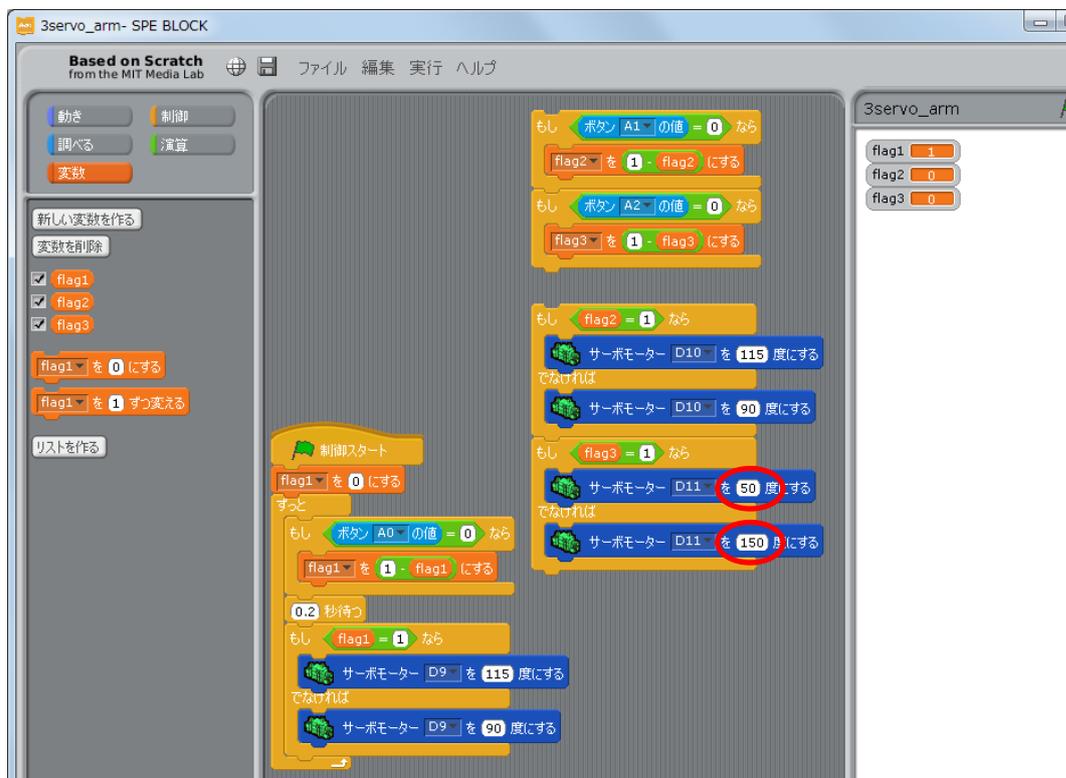
㊹ 3つ目のサーボモーターの設定をします。変数 flag3 を作成します。



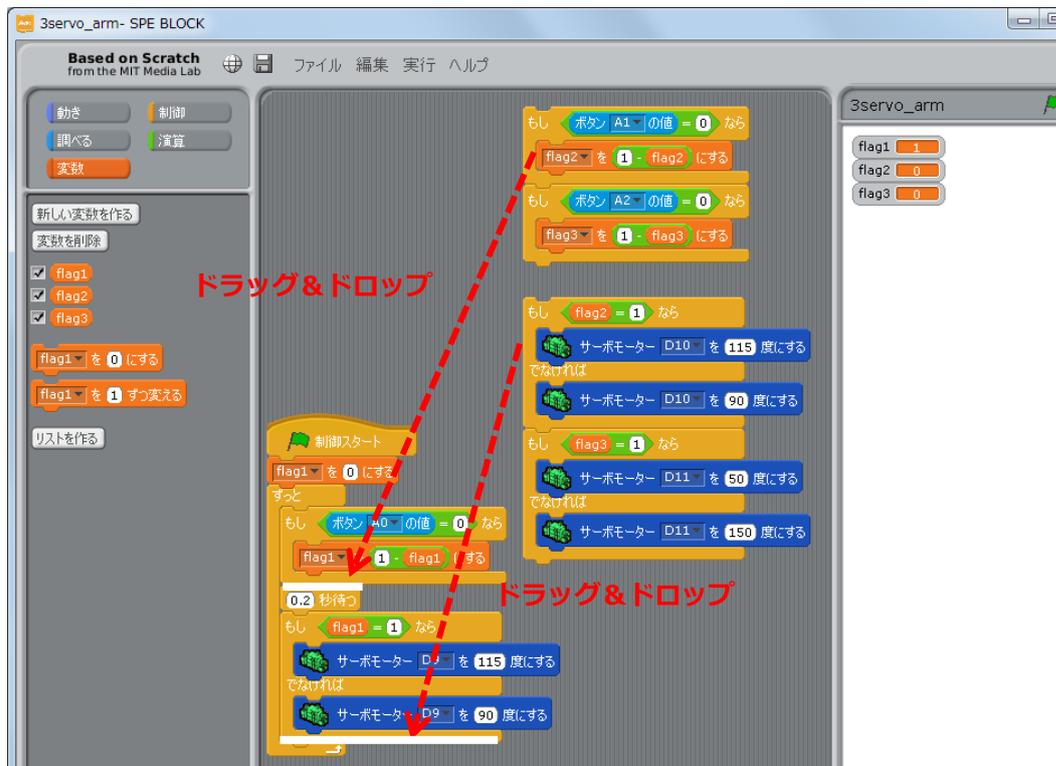
- ⑳ 2つ目の時と同様の手順で、複製したブロックの値を変えていきます。変数は flag3 を、サーボモーターは D11 を選択します。



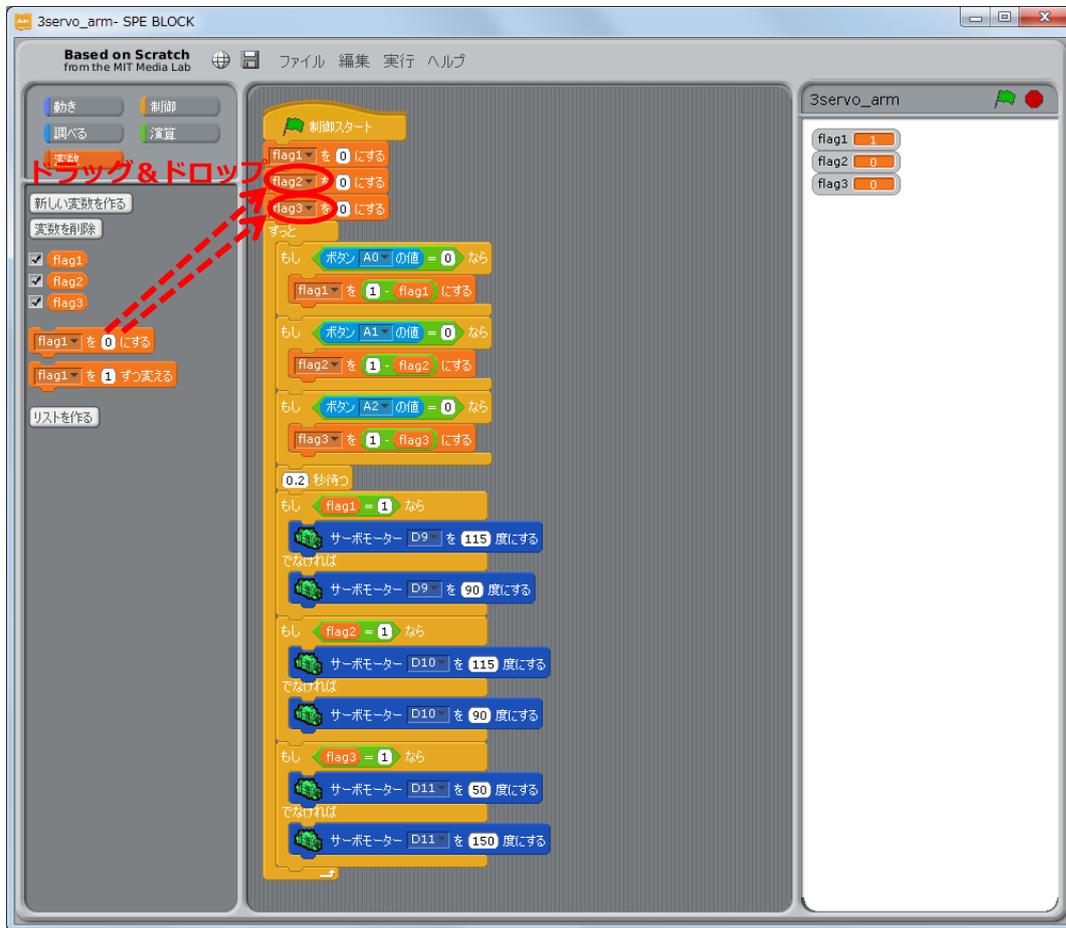
- ㉑ サーボモーターの角度を 50 度、150 度に変更します。



㊹ それぞれのブロックを白線の位置にくっつけます。



- ③⑩ flag1 を 0 にする ブロックを 2 つドラッグし、図の位置にくっつけます。それぞれ flag2, flag3 を選択します。



以上で、ボタン A0, A1, A2 を押すごとに D9, D10, D11 につながれたサーボモーターの角度を変えるアームのプログラムが作成できました。メニューバーの「実行」から「プログラム作成・転送」を選択して下さい。転送が完了すると、ボタン A0, A1, A2 を押すことにより、アームが動作します。