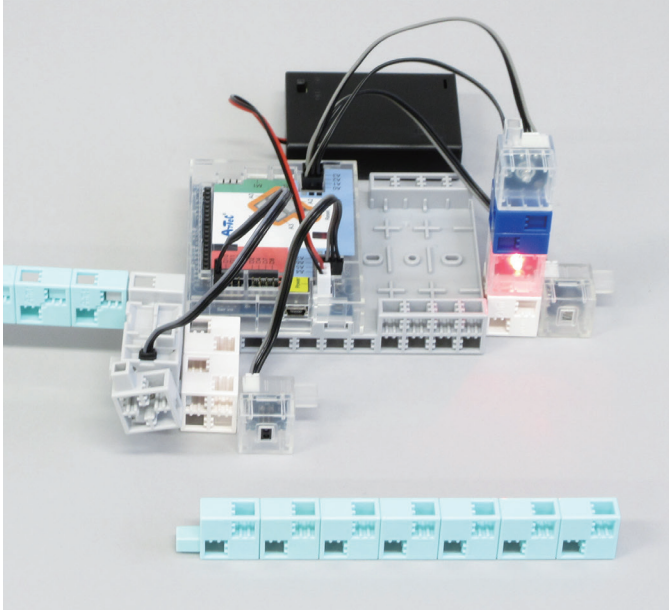
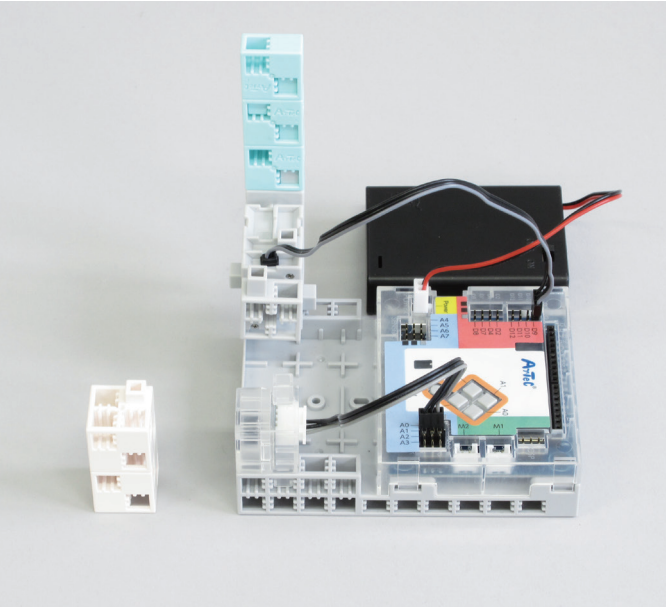
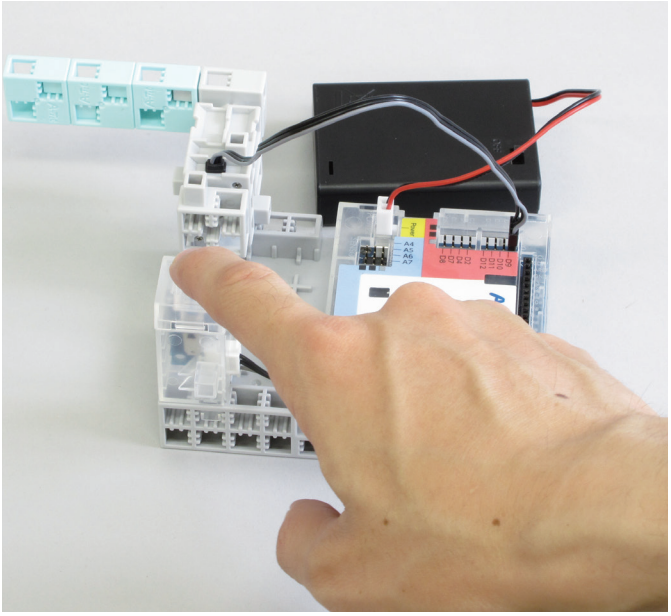
計測と制御キットC

プログラムによる自動ゲートの制御

教員用







目 次

部品名などの記載について

教科書では「センサ」「モータ」と表記されていますが、本テキストではソフトウェアの表記に合わせるため「センサー」「モーター」※1と表記しています。

※1：JIS 規格ではどちらでも誤りではないとされています。

第1章
身近なプログラミング

＜学習内容＞

- 身近な計測・制御システム
- プログラミング環境の基本操作

1．身の回りの計測・制御

私たちの身の回りにある電気製品は、計測・制御の仕組みを使って自動的に様々な仕事をしています。決まった動作を繰り返したり、外部の情報を元に柔軟に対応したりすることができるのは、コンピューターやセンサーが使われているからです。このような計測・制御システムは、センサー／コンピューター／アクチュエータの三つの要素から構成されています。

エアコンの例を見てみましょう。

例 エアコン

センサー

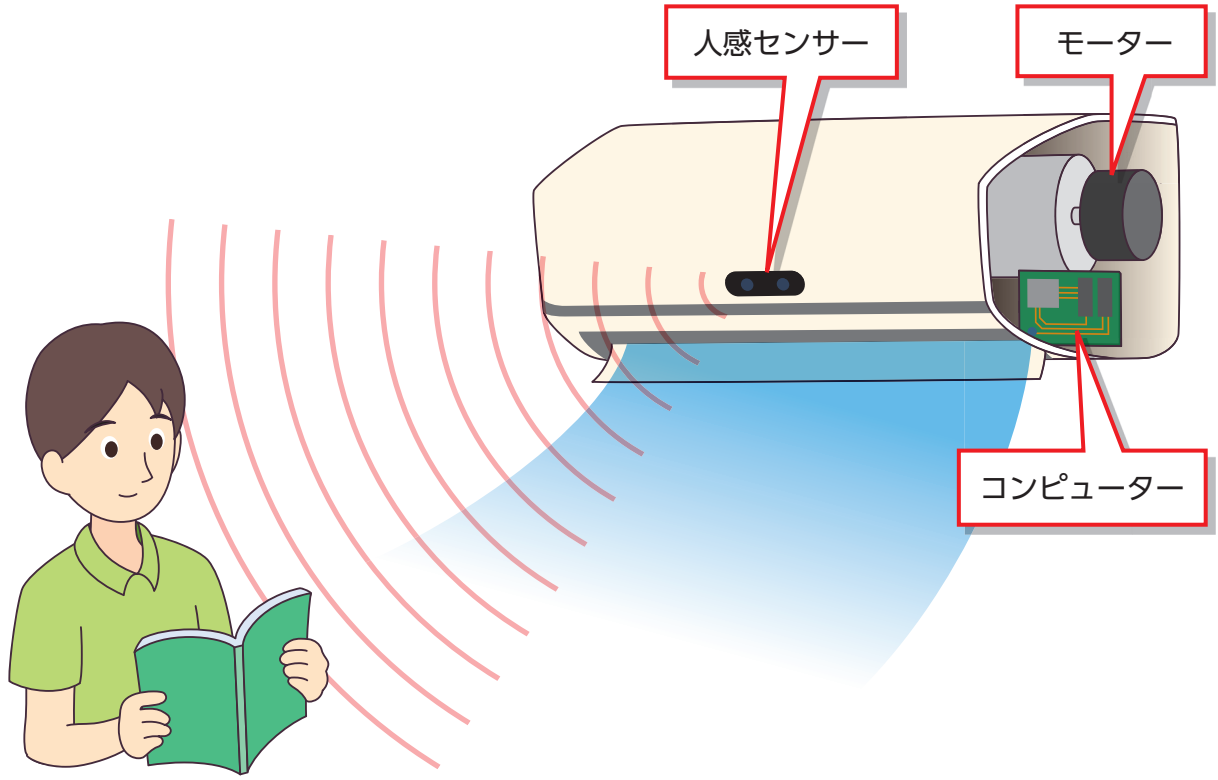
人感センサー（赤外線センサー）・温感センサーで人の位置や体温・室内の温度を計測する。

コンピューター

センサーで感知・計測した情報を処理・判断する。それを元に温度や風力を設定しモーターへ指示を出す。

アクチュエータ

コンピューターからの指令を受けてモーターの回転数を変え温度の調整をする。



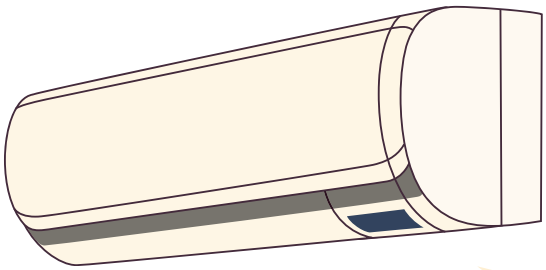
身近な製品でどういうものがコンピューターやセンサーを活用しているか探してみましょう。

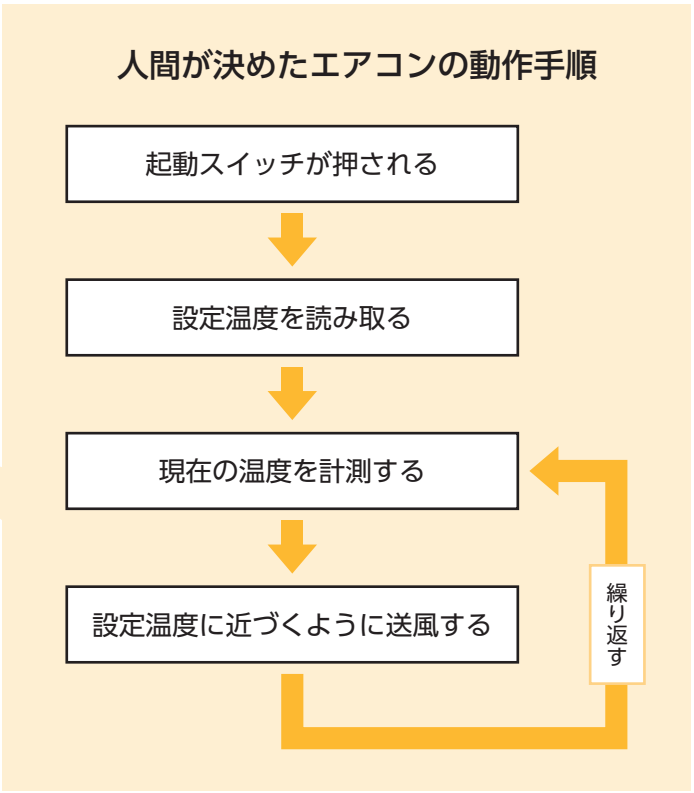
製品名	センサーで計測しているもの	計測結果からコンピューターが行う動作
風呂給湯器	水量	給湯する / 給湯を止める
お掃除ロボ	壁や障害物の有無	壁や障害物を避けるように動く

2．プログラミングとは

エアコンの例のように、コンピューターが使われている機械はセンサーから得た情報をもとに、あらかじめ人間が決めた手順通りにアクチュエータを動かします。

例 **エアコンを動かすとき**





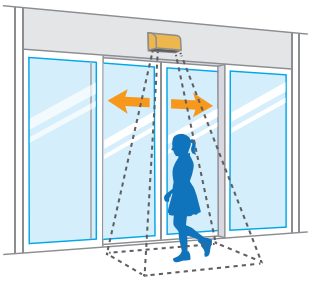
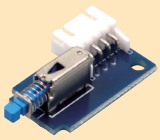
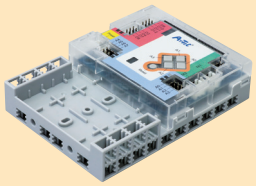
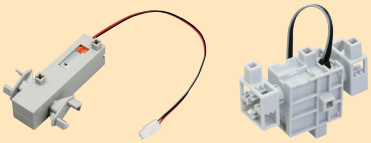
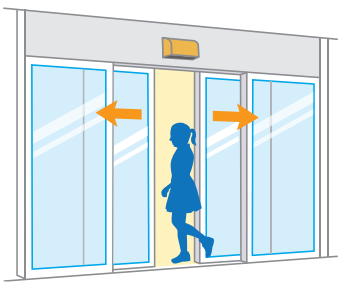
このようにコンピューターの動作手順を示したものをプログラムと言います。プログラムは人間が話すときに使う言葉とは違う、特別な言葉を使って表します。この言葉をプログラミング言語といい、プログラミング言語でプログラムを書くことをプログラミングと言います。

```
f ifNil: [
    "Turn setting laguage"
    self setLanguage: ltmp.
    ↑ self].

" Head and Footer file "
" Default English. "
headCode ← StandardFileStream new open: 'code\head←en.txt' forWrite: false.
footCode ← StandardFileStream new open: 'code\foot←en.txt' forWrite: false.

((ltmp = 'ja') | (ltmp = 'ja←HIRA')) ifTrue: [
    Transcript show: '---------- Trans to Japan ----------'; cr.
    headCode ← StandardFileStream new open: 'code\head.txt' forWrite: false.
    footCode ← StandardFileStream new open: 'code\foot.txt' forWrite: false.
]
```

▲プログラミング言語で書かれたプログラムの画面

3．計測・制御システムと ArtecRobo パーツの対応

ArtecRobo のパーツは以下のようなコンピューターによる計測・制御システムで使われる各要素に対応しています。

	自動ドアの動作の流れ	押しボタン式信号の動作の流れ
		
センサー 周囲の情報を計測する	 **赤外線フォトリフレクタ** 物体の有無を調べる	 **タッチセンサー** 押されたことを調べる
コンピューター センサーで得た情報を判断しアクチュエータに動作命令を出す	 **Studuino(スタディーノ) 基板**	※ Studuino（スタディーノ）基板は、計測システムを学習するために開発された Arduino 互換のマイコンボードです。
アクチュエータ コンピューターからの命令に従って動作する	 **DC モーター　サーボモーター** 電気を回転する力に変える	 **LED ライト** 電気を光に変える  **ブザー** 電気を音に変える
		

※アクチュエータとはエネルギーを動きに変えるものを指すため、動きのない LED やブザーはアクチュエータに含まれません。

4．プログラミング環境

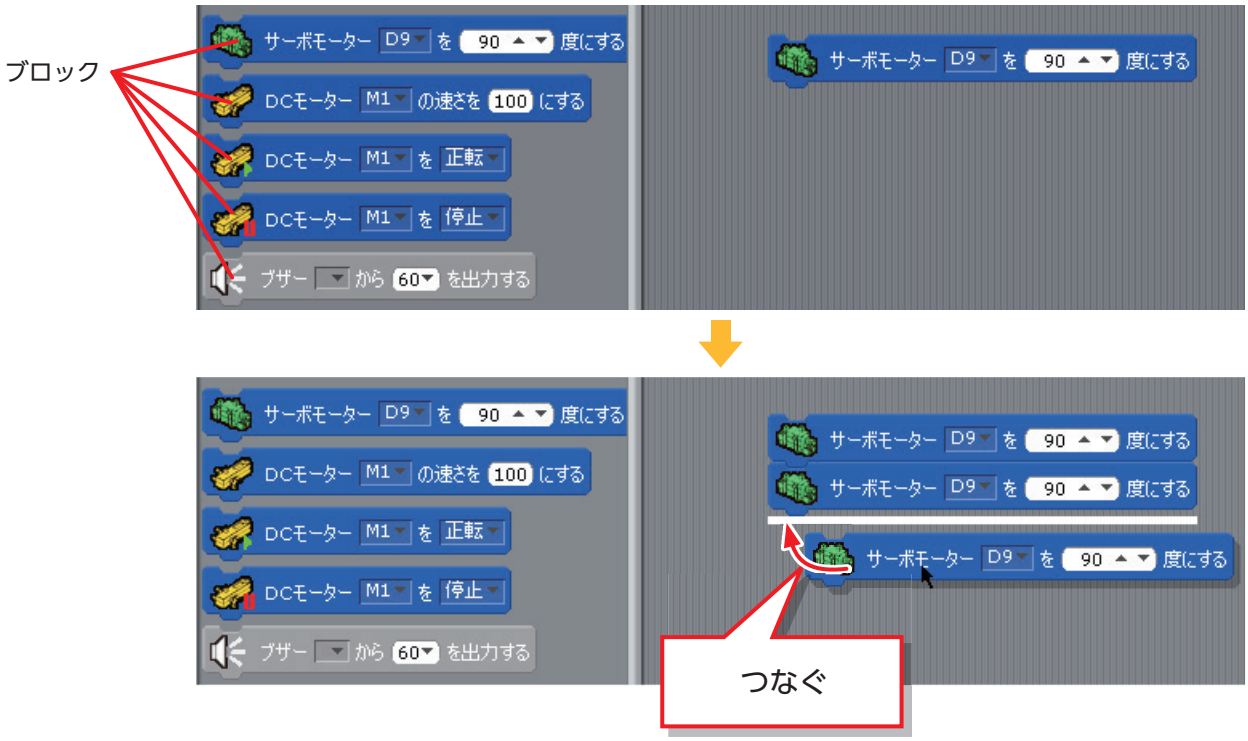
この授業では、文字の代わりにブロックのような絵をつないでコンピューターへの命令をプログラミングできる「ビジュアルプログラミング言語」を使います。

①ソフトウェアの立ち上げ

②操作方法

◆ プログラムの作成

ブロックパレットにある命令をおもちゃのブロックのようにつなぐことでプログラムをつくります。この命令のひとつひとつを「ブロック」と呼びます。



◆ ブロックの削除

削除するブロックをブロックパレットにドラッグ＆ドロップします。

カテゴリーの種類について

カテゴリーは「動き」「制御」「調べる」「演算」「変数」の５種類があります。それぞれのアイコンをクリックすることで、カテゴリーを選択することができます。

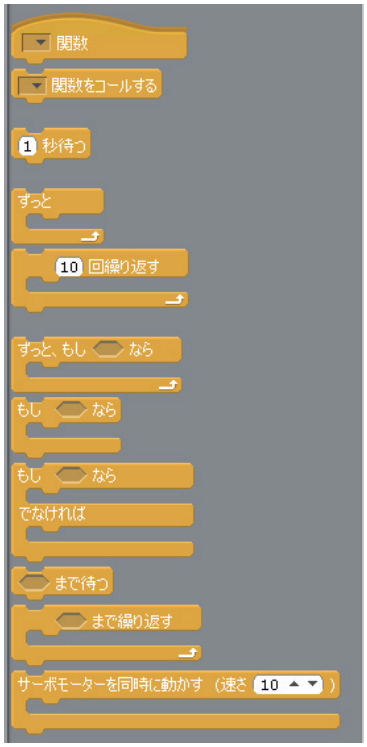
◆ 動き

アクチュエータの動きを命令します。

※灰色のブロックは使用できません。
後述の設定変更で使用可能になります。

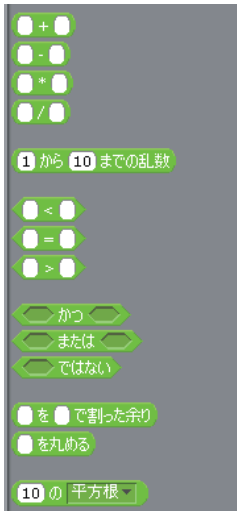
◆ 制御

命令の実行順を制御します。



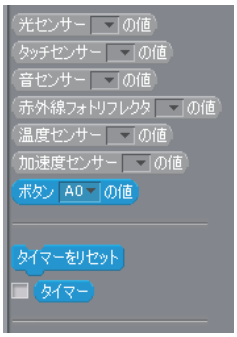
◆ 演算

計算の命令を出したり
条件式を作成します。



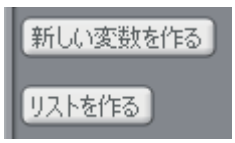
◆ 調べる

センサー上を指定して
周りの情報を調べます。



◆ 変数

数値情報の記録に使います。
（今回は使いません）



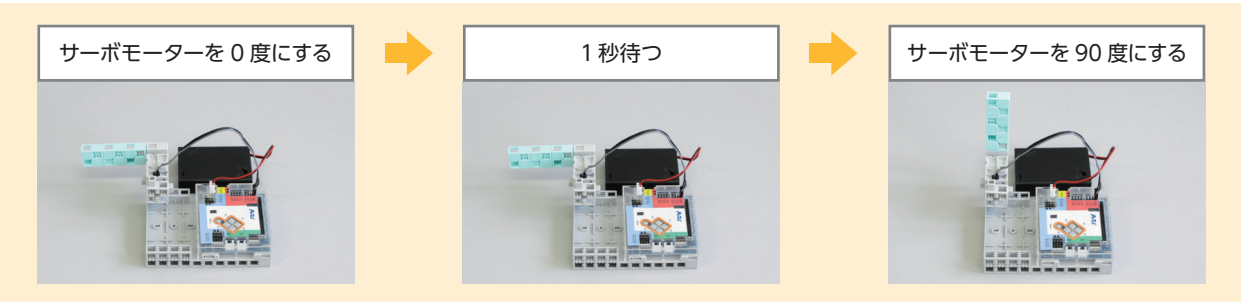
第2章
順次処理とフローチャート

＜学習内容＞

- 順次処理
- フローチャート

1．順次処理のプログラム

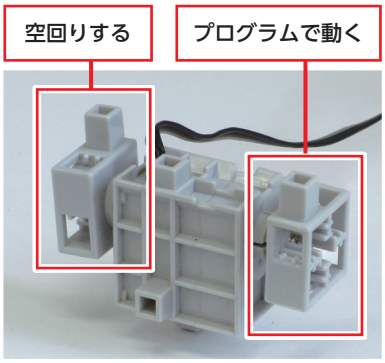
命令を順番通りに行う処理のことを「順次処理」といいます。サーボモーターを使ってゲートをつくる中で、順次処理のプログラムを学びましょう。



サーボモーターについて

・プログラムで回る角度を決めて動かすモーターです。
・回る角度は0度から180度の間で指定します。
・手でゆっくり回したときに重たく感じる側がプログラムで動きます。

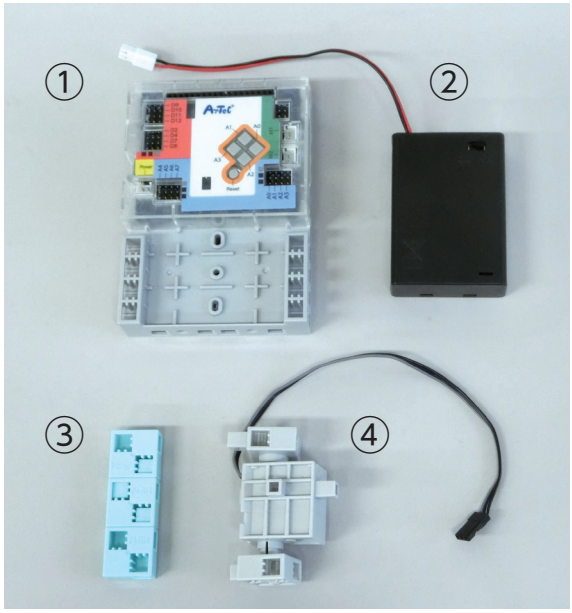
※無理やり回したり、はげしく動かしたりせずに、ゆっくりと手で回すようにしてください。



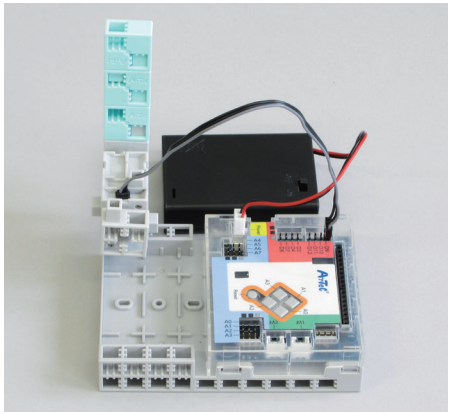
①組み立て

59 ページの組み立て説明を見て、ゲートを組み立てましょう。

組み立て準備

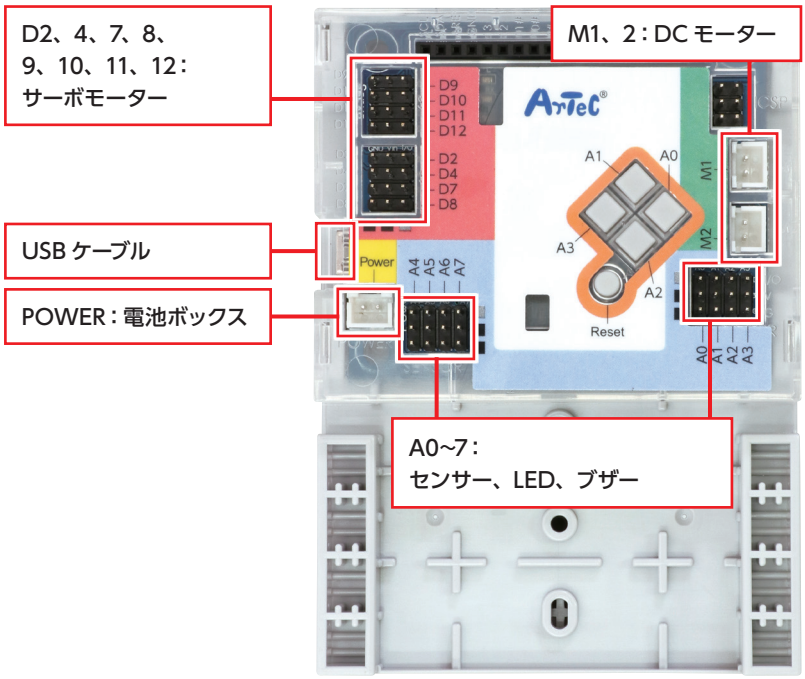


① Studuino（スタディーノ）基板 ……1
② 電池ボックス（電池入） ……1
③ ハーフブロック　薄水……3
④ サーボモーター ……1



Studuino（スタディーノ）基板に接続できるパーツ

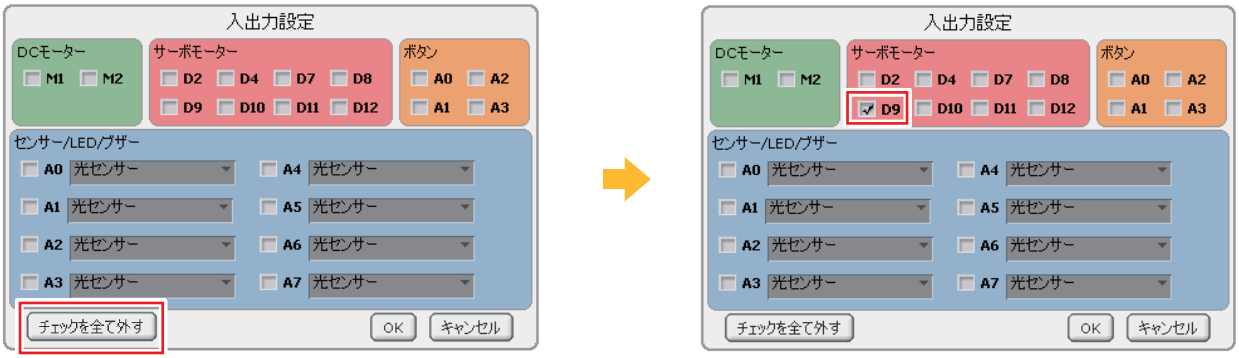
Studuino（スタディーノ）基板にセンサーやアクチュエータをつなぎ、プログラムを書き込むことで簡単に計測・制御システムをつくることができます。パーツによってコネクタをつなぐ場所が決められているので注意しましょう。



②入出力設定

入出力設定で Studuino（スタディーノ）基板のどの場所にどのパーツをつないでいるかを登録します。

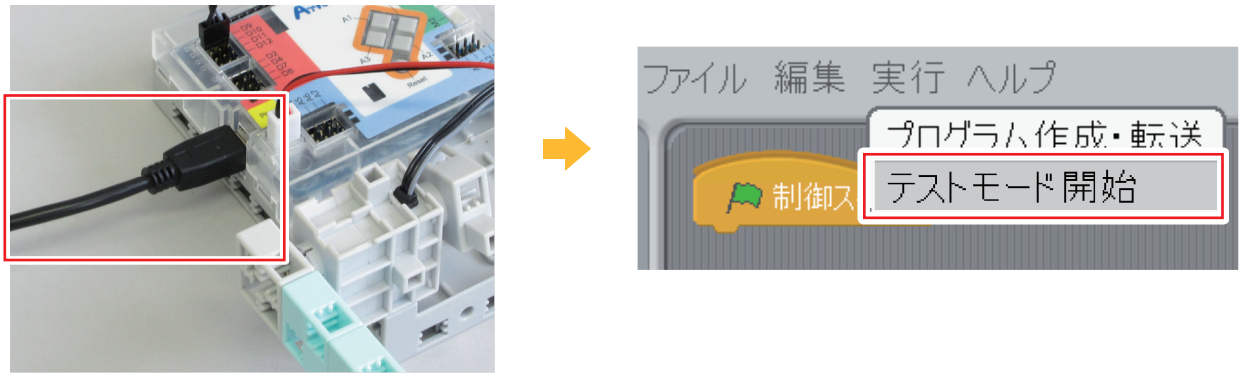
D9 につないだサーボモーターを登録します。一度全てのチェックを外してから、サーボモーターの D9 にチェックを入れて「OK」をクリックしましょう。



Studuino（スタディーノ）基板はつないだパーツを自動で認識できないので、新しくパーツをつないだり、パーツをつなぎかえたりしたときは必ず入出力設定を行うようにしましょう。

③テストモード

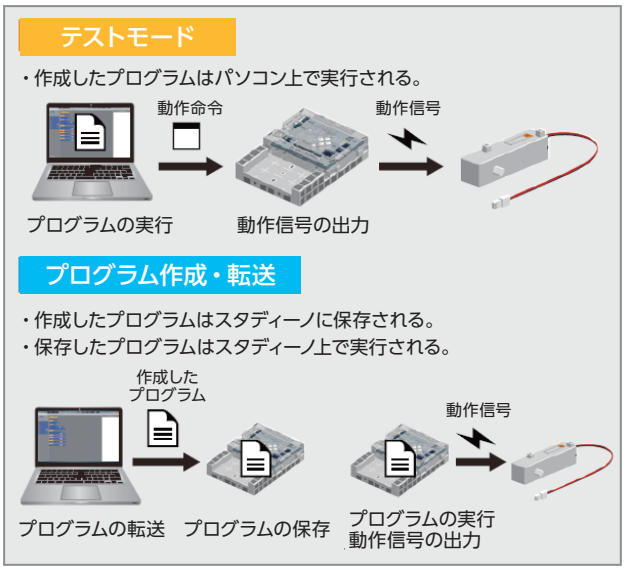
テストモードはパソコンと Studuino（スタディーノ）基板が常に通信している状態にする機能で、動作を確認しながらプログラムをつくることができます。USB ケーブルでコンピューターと Studuino(スタディーノ) 基板をつないで、テストモードにしましょう。



USB ケーブルの接続時はパソコンから電力が供給されるので、電池ボックスのスイッチをオンにしなくてもブザーや LED を動かすことができます。ただし、DC モーターやサーボモーターはパソコンからの電力が使用できないため、必ず電池ボックスを使うことに注意しましょう。

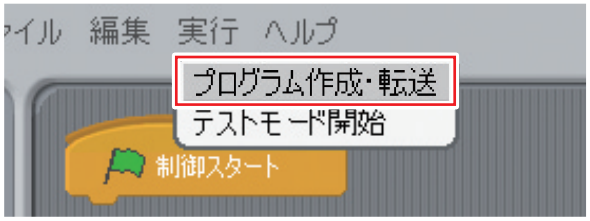
コンピューターと Studuino（スタディーノ）基板を離して使いたいとき

プログラムを実行させる方法には「テストモード」のほかに、「プログラム作成・転送」があります。「プログラム作成・転送」は発表などでコンピューターと Studuino（スタディーノ）基板を離して使いたいときに行います。



◆ プログラムの転送方法

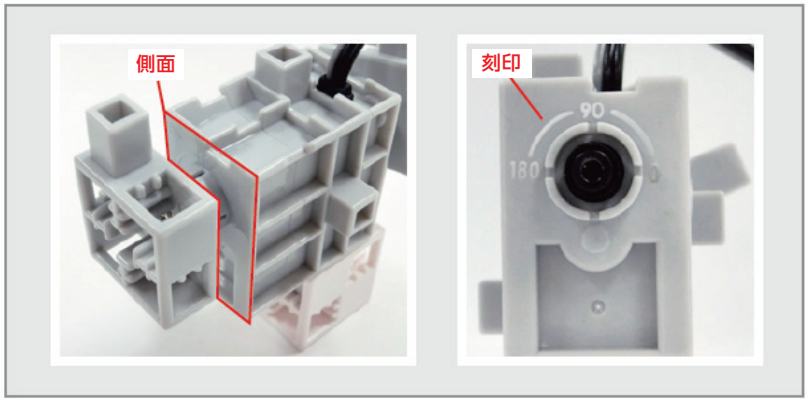
作成したプログラムを「制御スタート」につないでから「プログラム作成・転送」をクリックします。



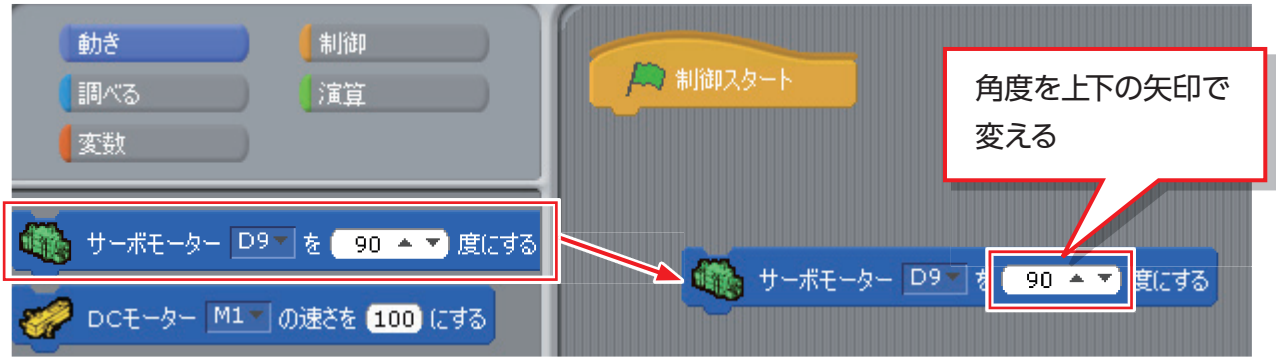
プログラム作成・転送は、プログラムをつくるたびに転送を行う必要がありますが、USB ケーブルを外してもプログラムを実行できたり、プログラムの実行や読み込みが早くなるというメリットもあります。

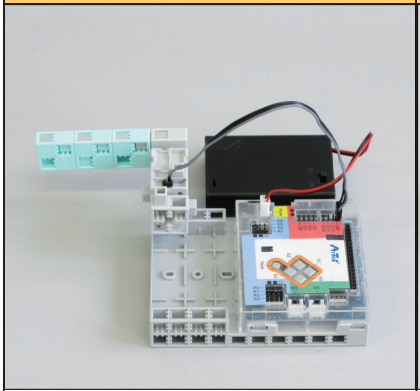
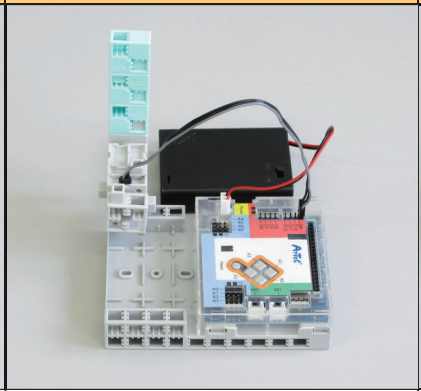
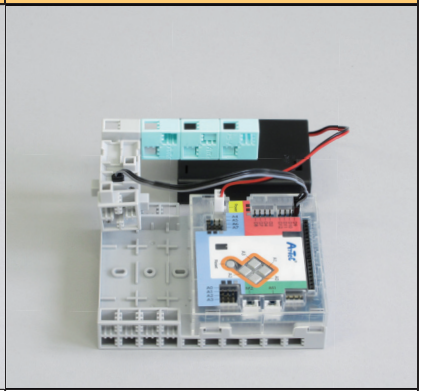
④サーボモーターの動作確認

サーボモーターを動かして、プログラムと動作の関係を調べましょう。サーボモーターの角度は、側面の刻印で確認することができます。



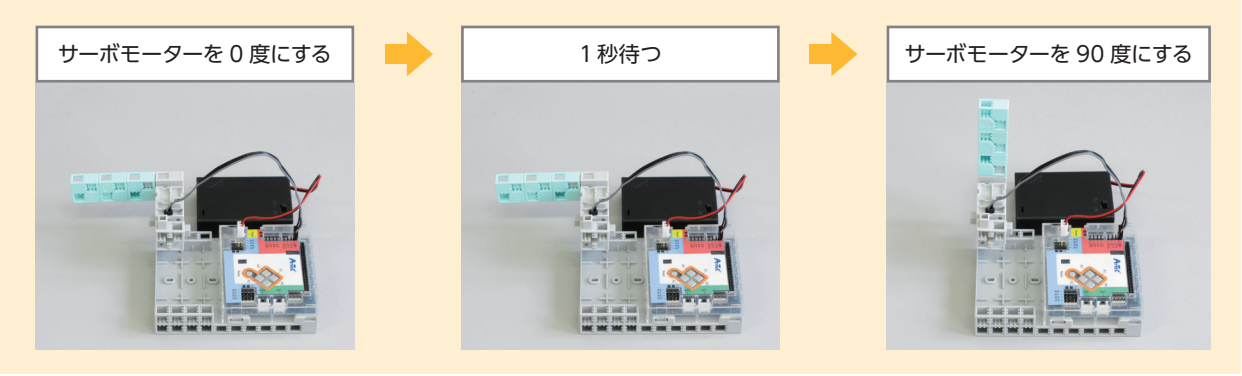
サーボモーターのブロックをスクリプトエリアに並べましょう。電池ボックスのスイッチをオンにして、サーボモーターの角度を上下の矢印で変え、ゲートの位置とサーボモーターの角度の関係を調べましょう。



ゲートが左のとき	ゲートが真ん中のとき	ゲートが右のとき
		
0 度	90 度	180 度

⑤プログラミング

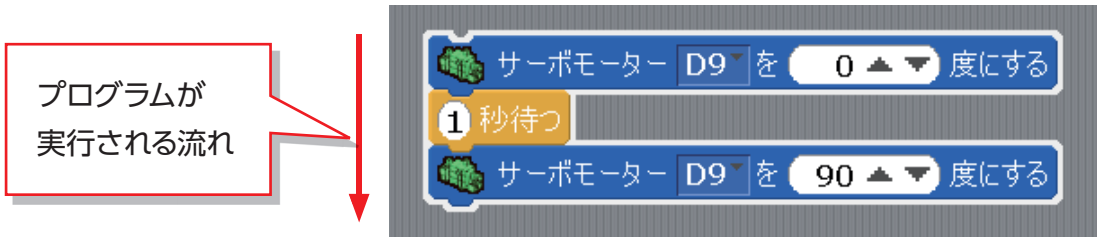
サーボモーターが 0 度→ 90 度と動くようにプログラムをつくりましょう。



サーボモーターを 0 度に動かすプログラムをつくりましょう。数値はキーボードでも、数字をクリックしたときに表示されるパッドでも入力可能です。

つぎに、制御カテゴリーにある「1秒待つ」のブロックを先ほどのブロックの下につなぎましょう。

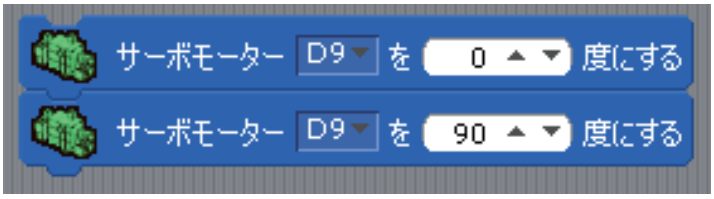
最後に、サーボモーターを９０度に動かすブロックをつなぎます。つないだブロックをクリックしてプログラムを実行しましょう。プログラムの実行中は実行されているブロックが白枠で囲われます。



このように上から順番にプログラムが処理されることを「順次処理」といいます。

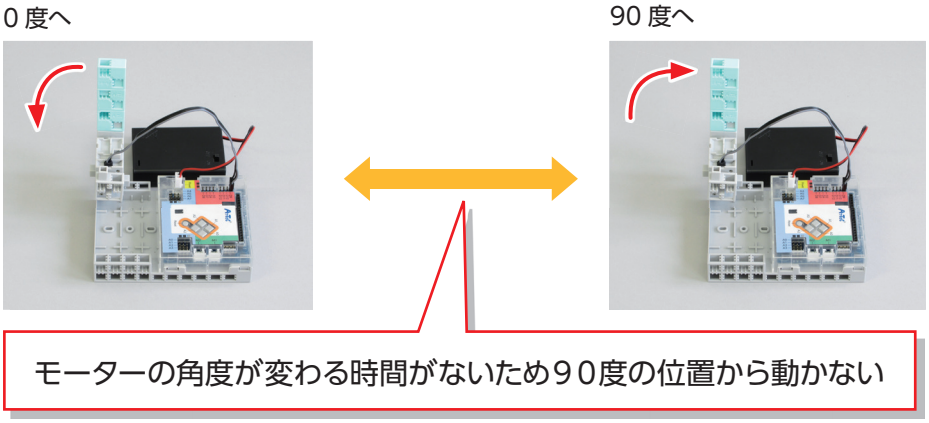
「1秒待つ」のブロックを入れないとどうなるのか？

以下のプログラムで動作をさせた場合、サーボモーターは９０度からほぼ動きません。

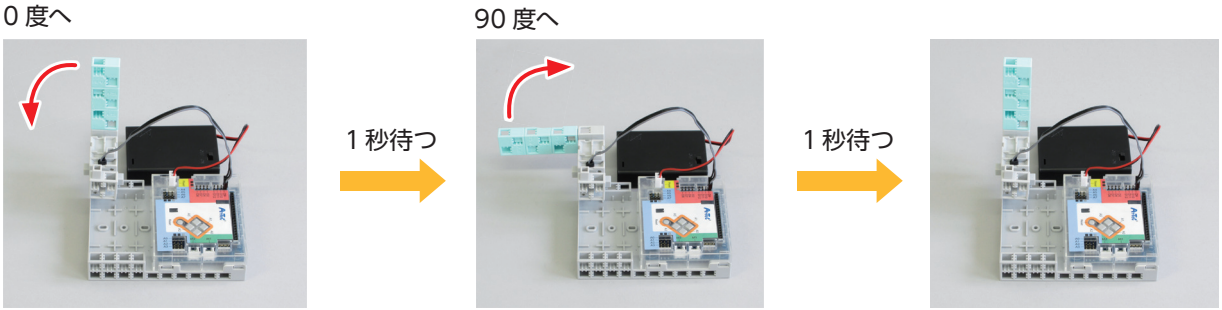


これはコンピューターがプログラムを非常に高速で処理していることが原因です。「サーボモーターを0度にする」命令の直後に、「サーボモーターを９０度にする」命令が行われてしまうため、想定している動作になりません。

「1秒待つ」を入れない場合

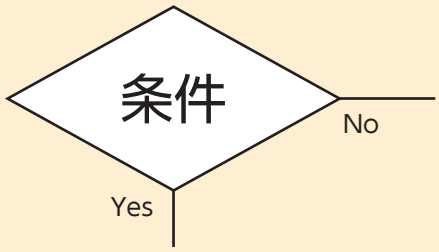


「1秒待つ」を入れる場合

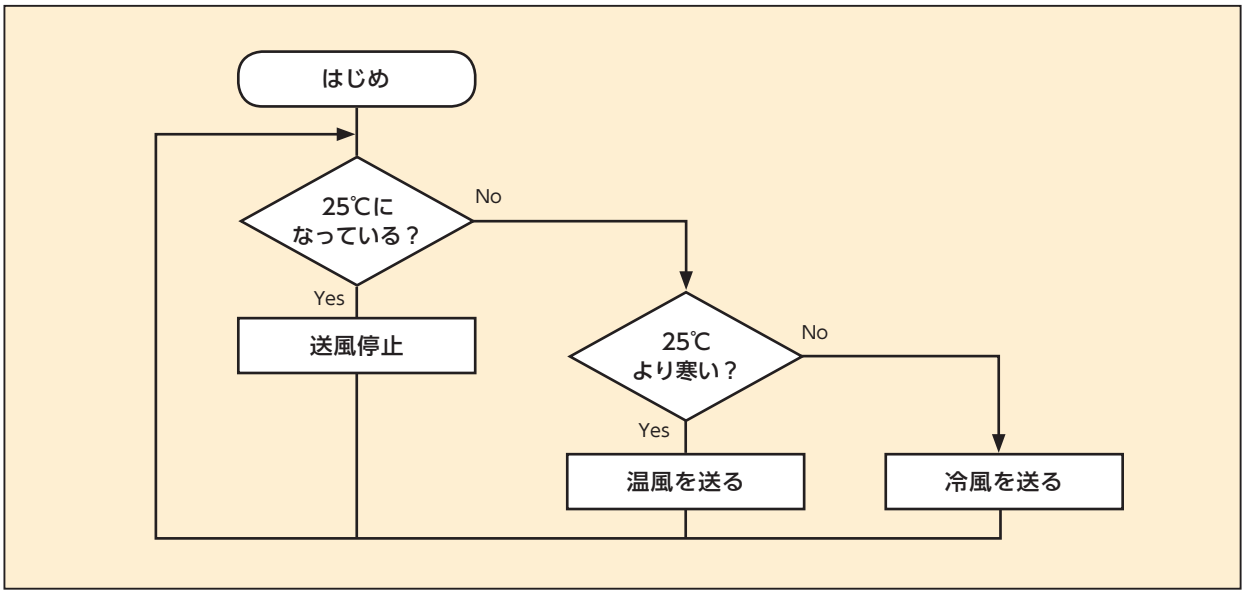


2．フローチャート

処理の手順をあらかじめ整理しておくとプログラムがつくりやすくなります。手順を整理したり考えをまとめる方法の1つとして、「フローチャート」という図がよく用いられます。

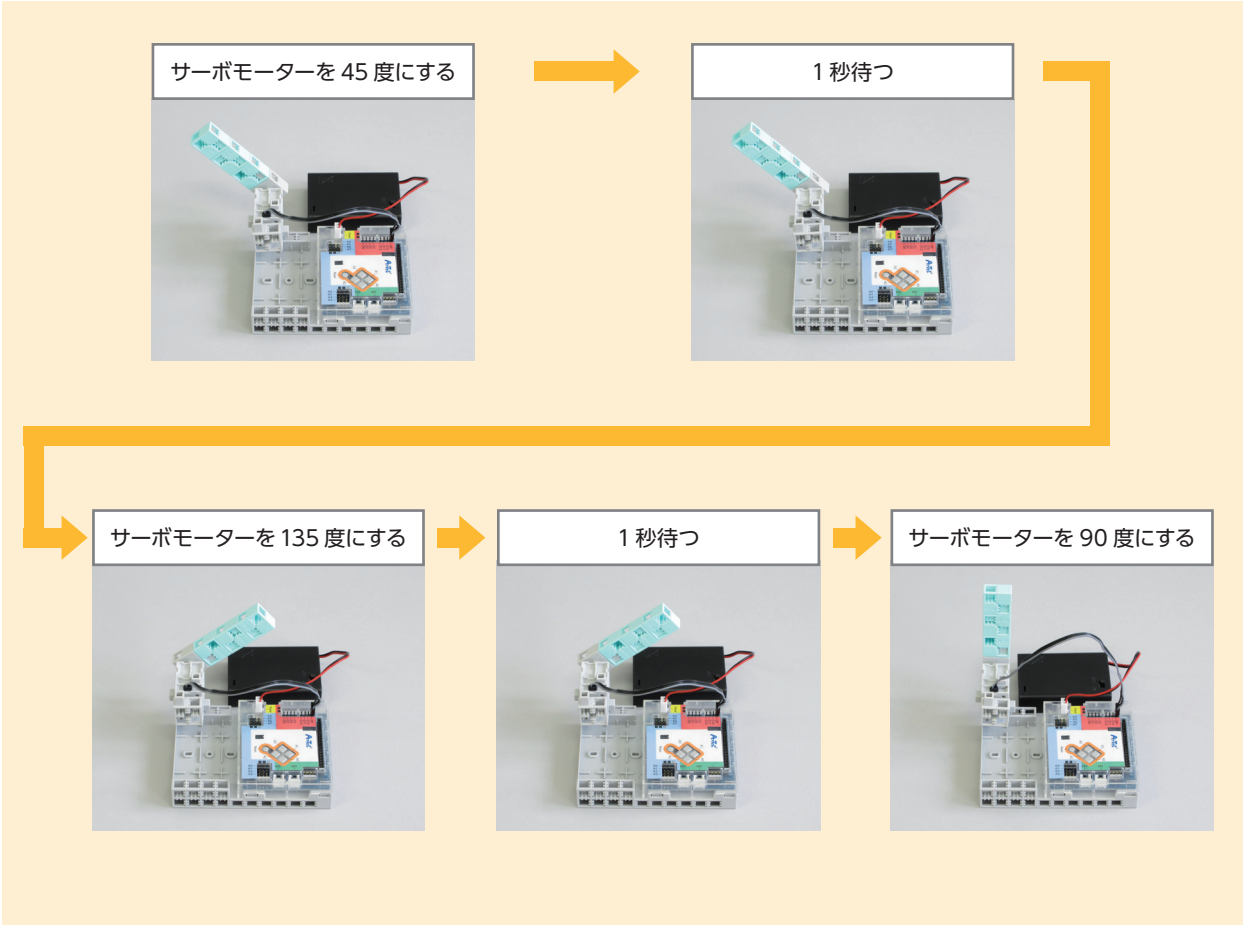
代表的なフローチャート記号	
	処理の開始・終了
	一般的な処理
	繰り返し処理の開始
	繰り返し処理の終了
	条件で処理を分ける

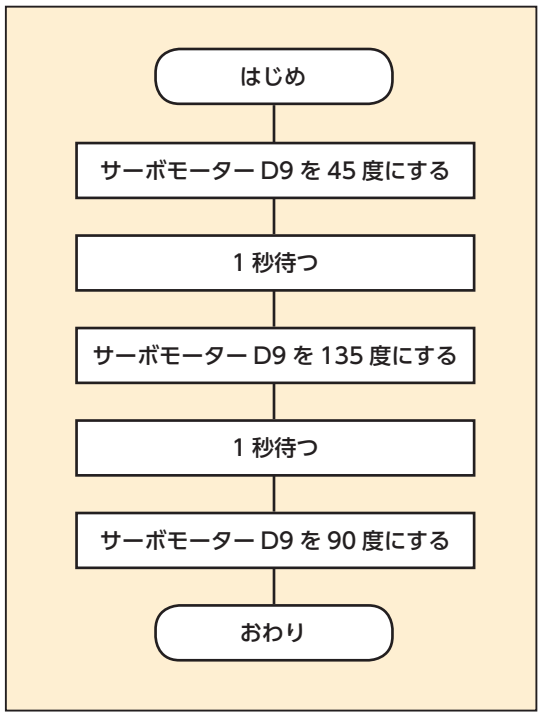
例 **室温を25℃に保つエアコンの動作の手順を表したフローチャート**



練習課題

次の動作をフローチャートを使ってまとめたあと、プログラムをつくりましょう。





第3章
繰り返すプログラム

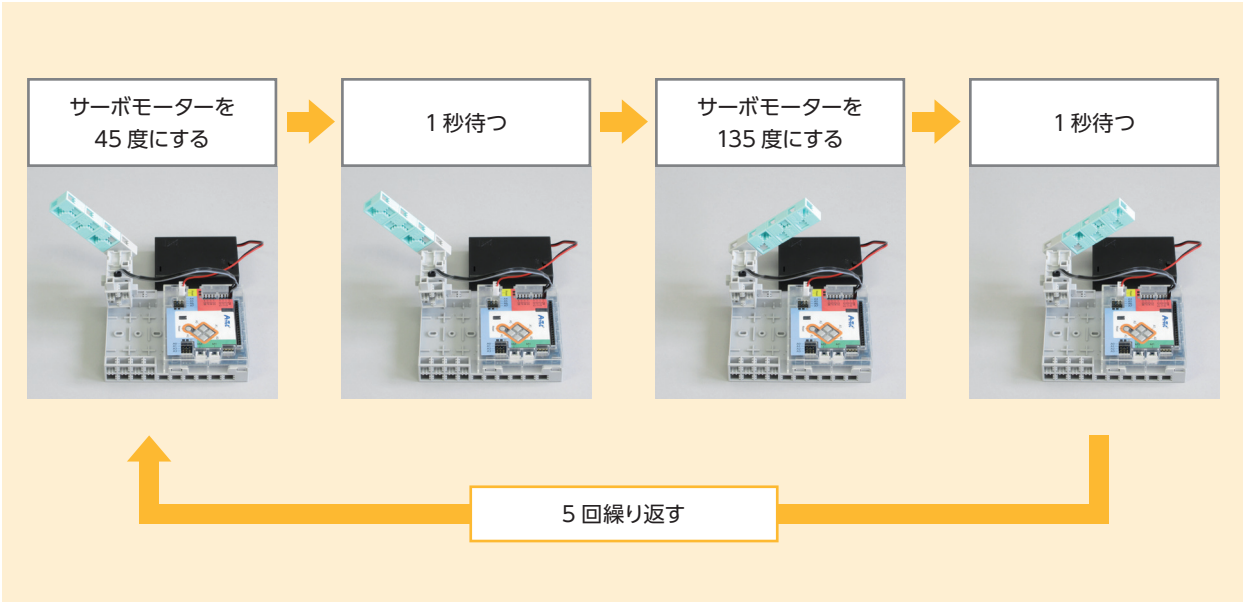
＜学習内容＞

- 繰り返し

1. 繰り返しのプログラム

サーボモーターが４５度と１３５度に交互に５回動くプログラムをつくりましょう。

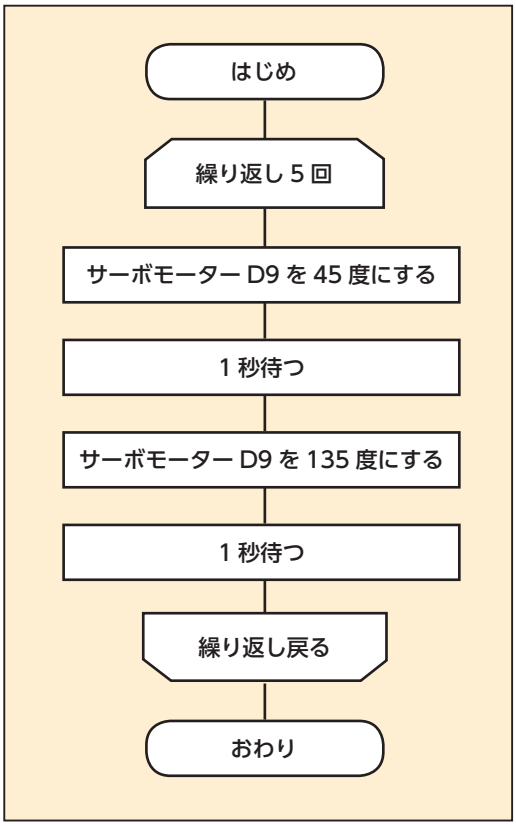
第２章と同じゲートを使いましょう。



①フローチャート

同じ動作を複数回繰り返す場合、以下の記号を使います。動作を表すフローチャートをまとめましょう。

繰り返し○回	繰り返し処理の開始
繰り返し戻る	繰り返し処理の終了



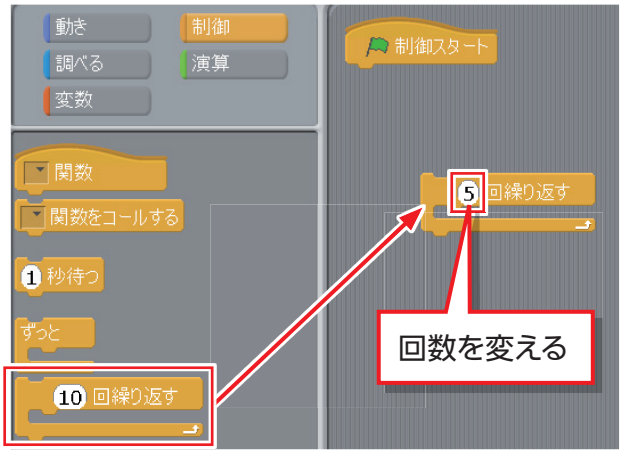
②プログラム作成

まとめたフローチャートをもとにプログラムをつくりましょう。

同じ動きを複数回繰り返す場合、 

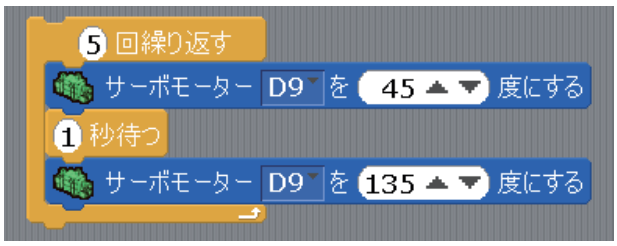
を利用すると簡単にプログラムをつくることができます。

に囲まれたブロックが指定した回数だけ繰り返し実行されます。

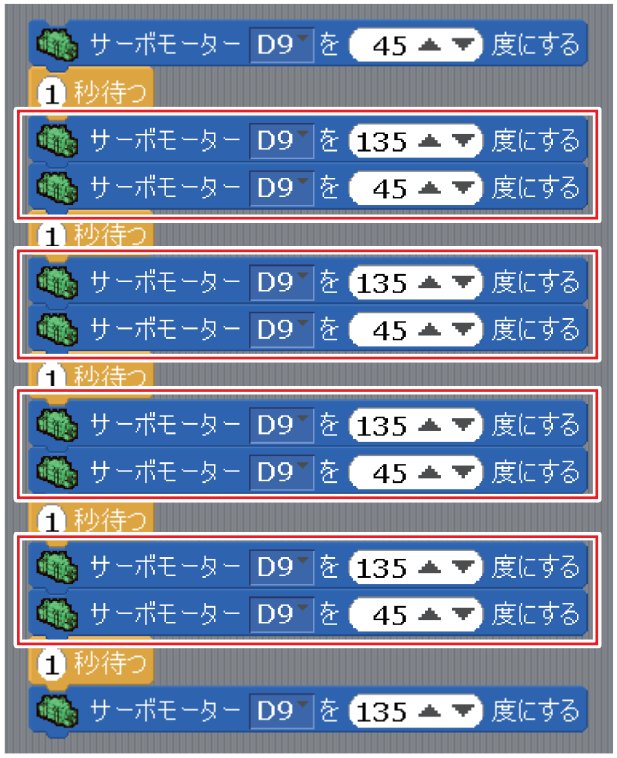


③動作確認と修正

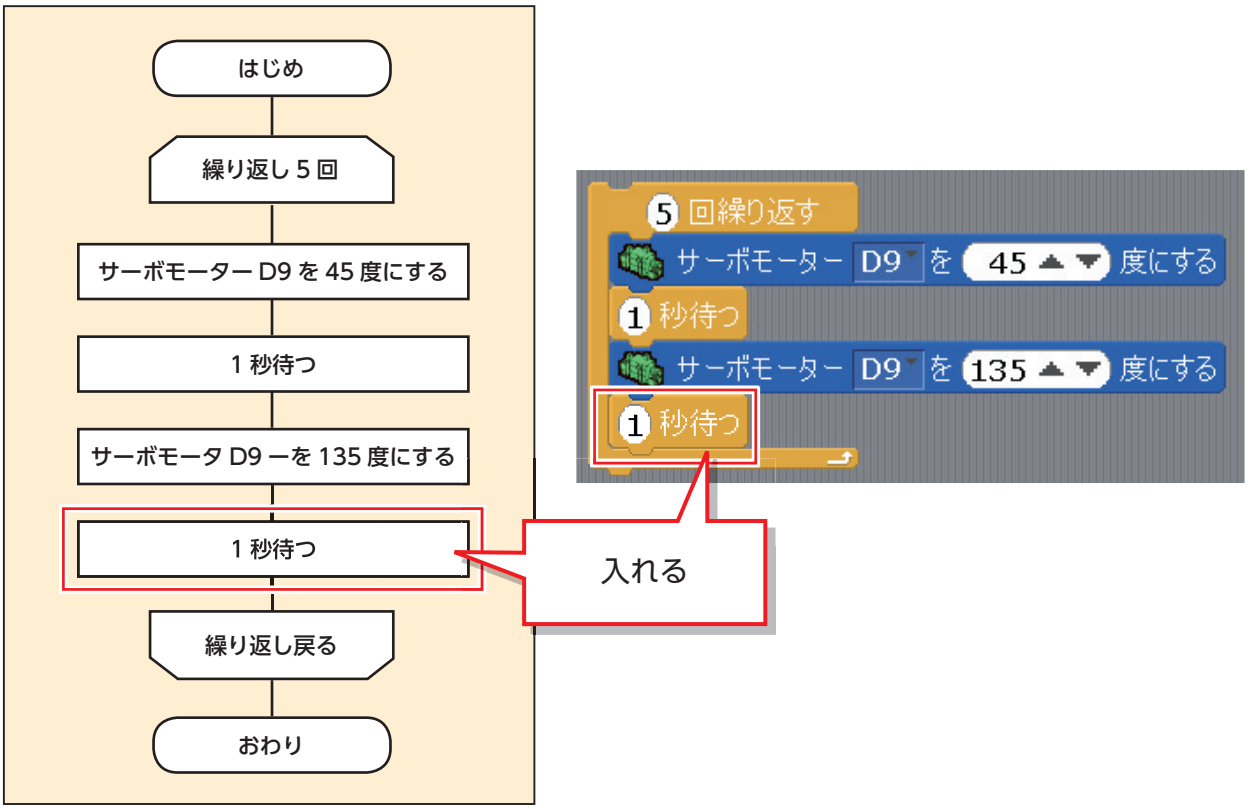
完成したプログラムを動かすと、想定通りに動かないことがあります。そのときのプログラムは以下のようになっているのではないでしょうか。



このプログラムを繰り返しブロックを使わずに書いてみると、赤枠部分で 135 度にするブロックの直後に 45 度にするブロックが存在しており、135 度にする時間がないことがわかります。



コンピューターはプログラムを組んだ通りに動きます。そのため、プログラムをつくる前にフローチャートを使って動作を予測し、間違った動きをする恐れがないかを確認することが大切です。

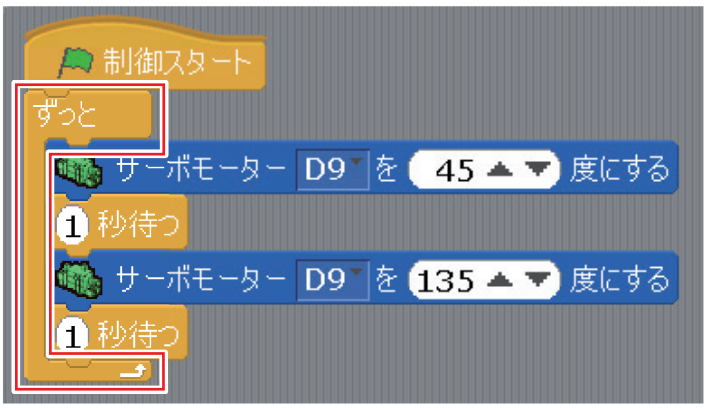


④無限に繰り返すプログラム

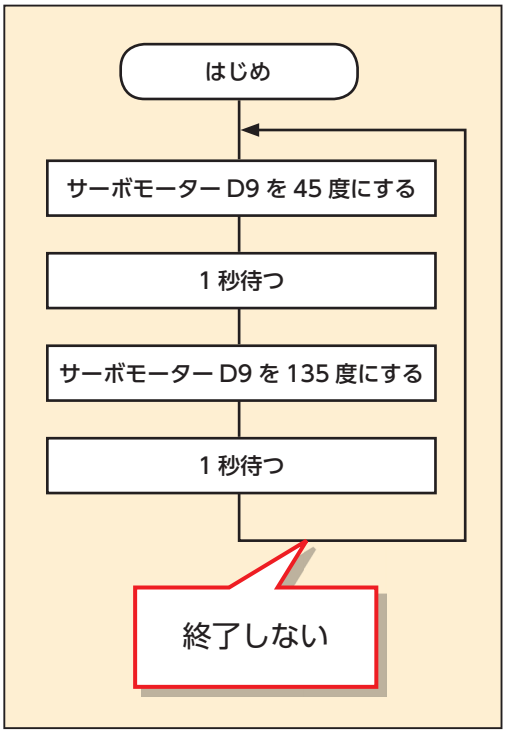
同じ動きをずっと繰り返させたい場合は、 

を利用します。

「5回繰り返す」を「ずっと」のブロックに変更しましょう。



このとき、フローチャートは右のように「終了」がなくなり、繰り返し続けることを矢印で表します。



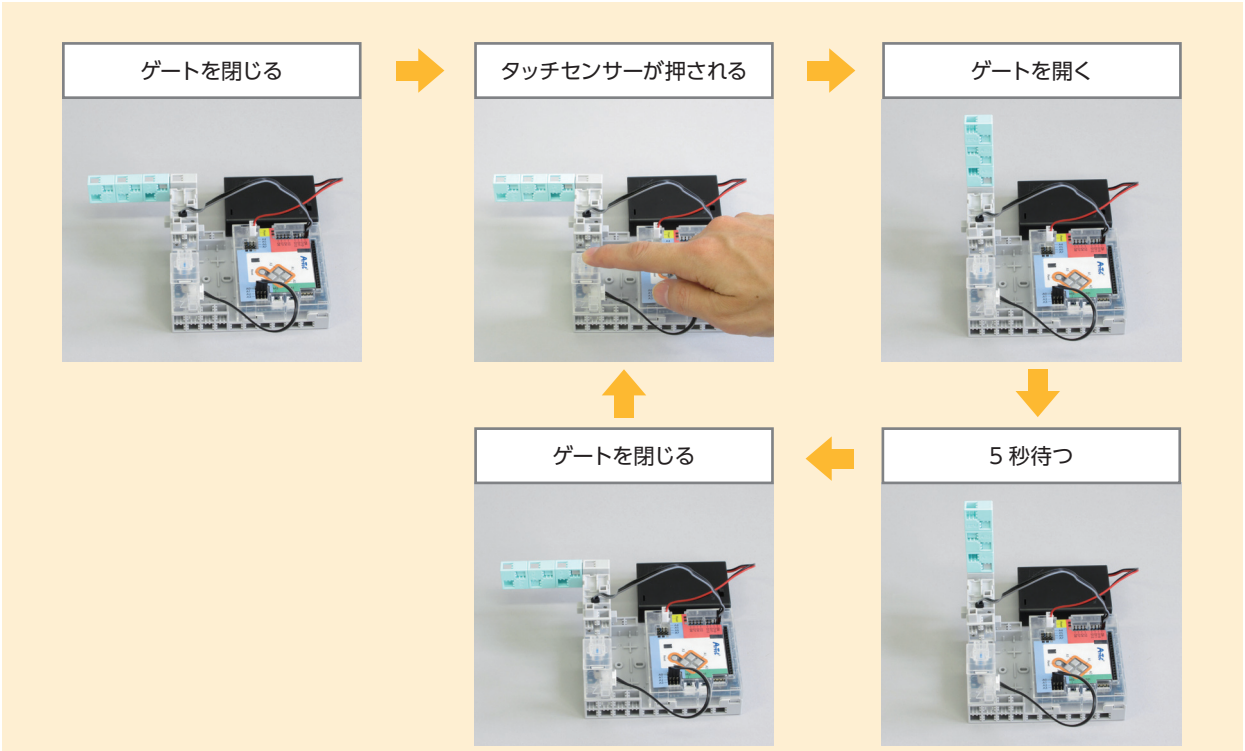
第4章
条件分岐のプログラム

＜学習内容＞

- 条件分岐
- タッチセンサー

1. 条件分岐のプログラム

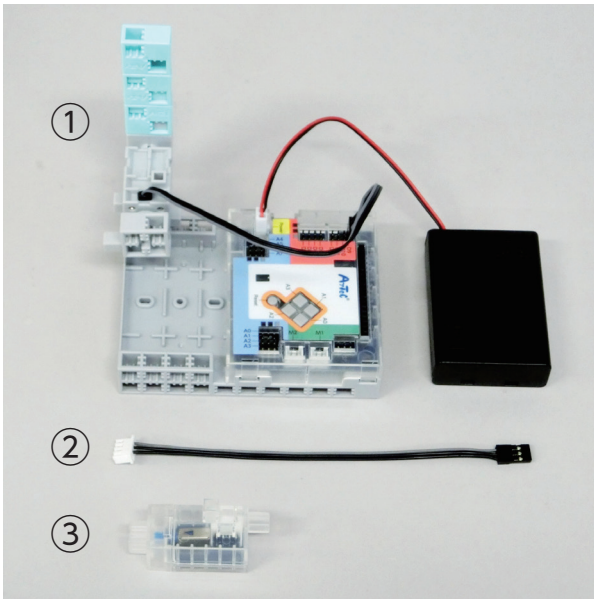
タッチセンサーを用いて、押しボタン式のゲートをつくりましょう。



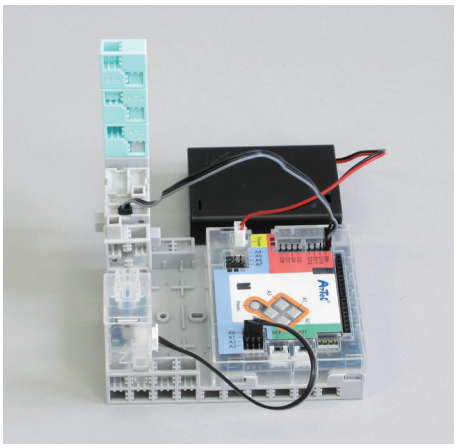
①組み立て

60 ページの組み立て説明を見て、ゲートにタッチセンサーを取り付けた押しボタン式ゲートを組み立てましょう。

組み立て準備



① ゲート（組立済）……1
② センサーコード……1
③ タッチセンサー……1



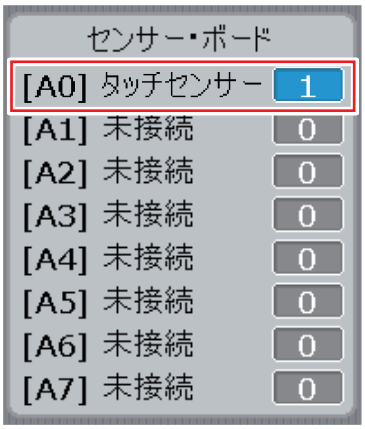
②入出力設定

A0 にタッチセンサーが追加されたので、A0 にチェックをつけて「タッチセンサー」を選択しましょう。

③タッチセンサーの数値確認

センサーの情報は数値で表されます。テストモードを実行し、タッチセンサーが押されているときと押されていないときの数値の変化を「センサー・ボード」で確認しましょう。

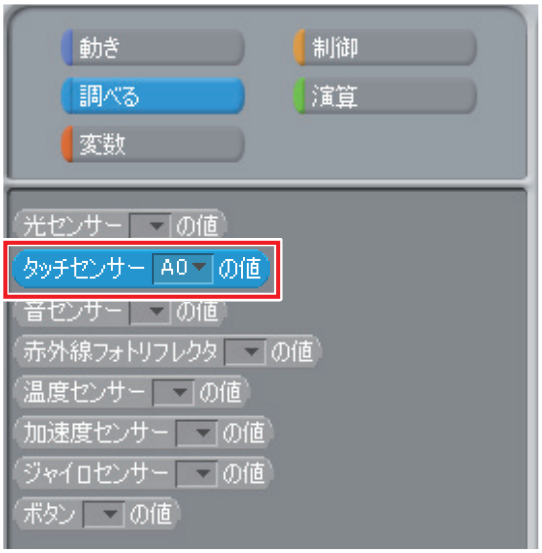
「センサー・ボード」はテストモード実行中に表示され、センサーの値がリアルタイムで確認できます。



タッチセンサーが 押されているときの数値	タッチセンサーが 押されていないときの数値
0	1

タッチセンサーの数値は「調べる」カテゴリーの

で知ることができます。

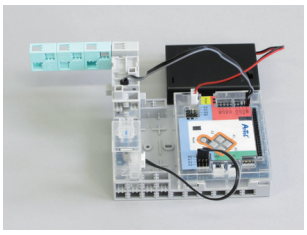
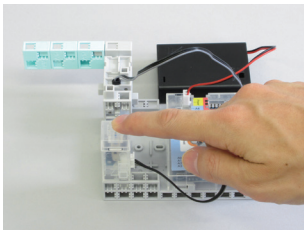
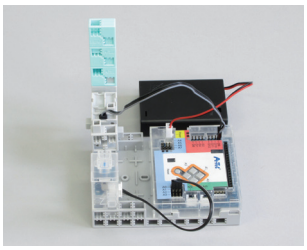
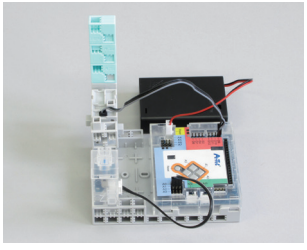
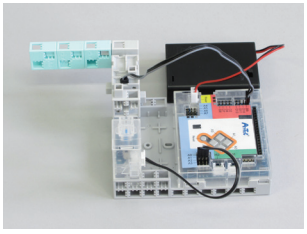


また、 

をクリックすると、右の画像のようなメッセージ形式で現在の数値を知ることができます。

④動作の整理

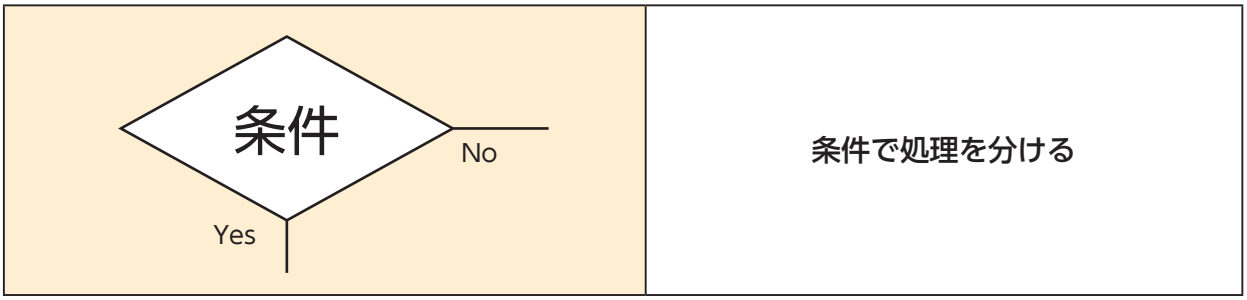
動作とプログラムの関係を整理しましょう。

順番	動作	プログラム
①	ゲートを閉じる 	サーボモーター D9 を 0 度にする
②	タッチセンサーが押される 	タッチセンサー A0 の値 ＝ 0
③	ゲートを開く 	サーボモーター D9 を 90 度にする
④	5秒待つ 	5 秒待つ
⑤	ゲートを閉じる 	サーボモーター D9 を 0 度にする

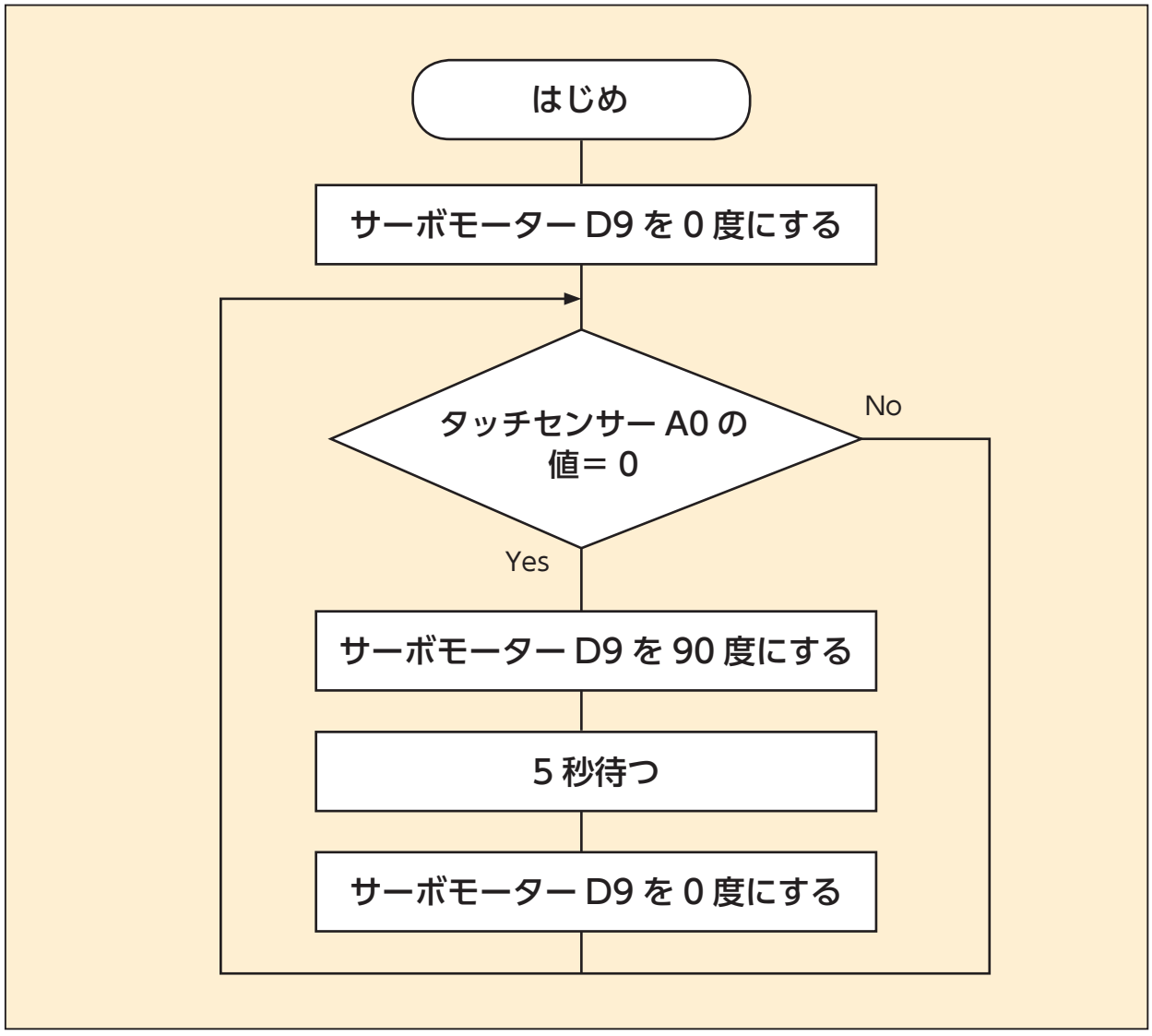
②～⑤を
ずっと繰り返す

⑤フローチャート

フローチャートで条件によって処理を分けることを表すときは以下の記号を使います。



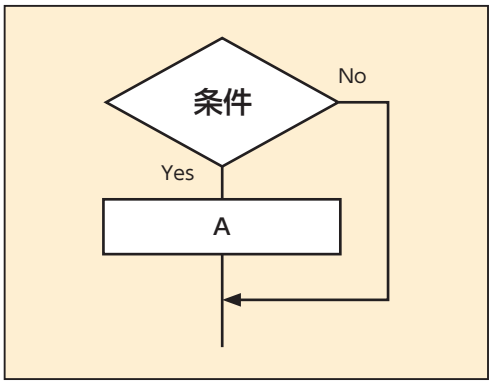
26 ページで整理した動作を参考に、フローチャートをまとめましょう。

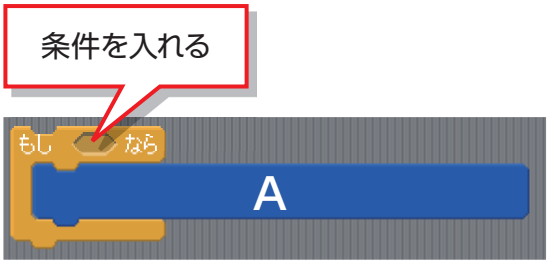


フローチャートに「終了」が存在せずに繰り返しているのは、タッチセンサーが押されているかを常に確認するためです。

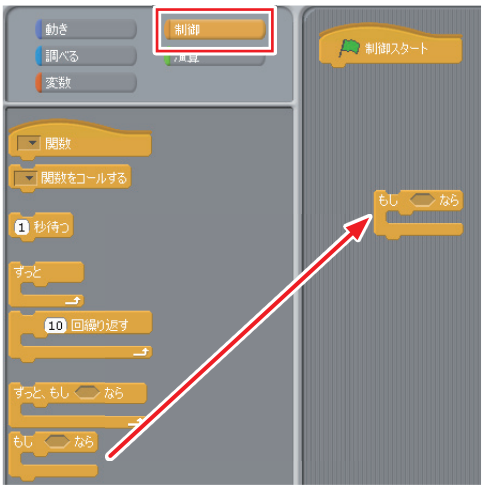
⑥プログラム作成

条件で処理を分けるフローチャートは次のプログラムと同じです。

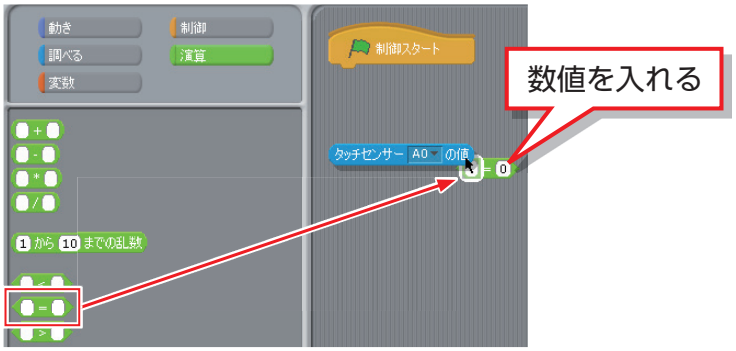




は「制御」カテゴリーにあります。



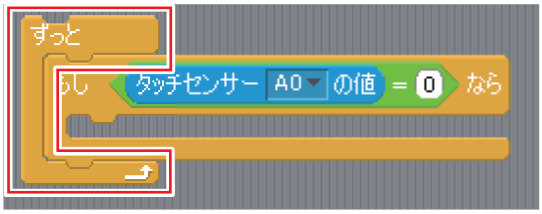
条件は「演算」カテゴリーの  と  を組み合わせてつくることができます。



作成した条件は  の空欄に入れて使うことができ、条件が成り立つときだけ囲まれたブロックが処理されるようになります。

⑦動作確認

つくったプログラムで想定通りに動作しない場合、次の点を確認して修正しましょう。

「ずっと」のブロックを使っているか	ゲートが開いたあとに 閉じるプログラムを入れているか
	

ステップ実行機能でプログラムの流れを可視化する

条件で処理を分けると、プログラムが少し複雑になります。複雑なプログラムを作成する際に間違えてしまうことは頻繁に起こることであり、その時に誤りを素早く発見し修正すること(=「デバッグ」)が大切です。

どのような流れでプログラムが動いているのかに注目することは、誤りの発見に有効です。プログラムの処理の流れを可視化する機能として、「ステップ実行機能」があります。

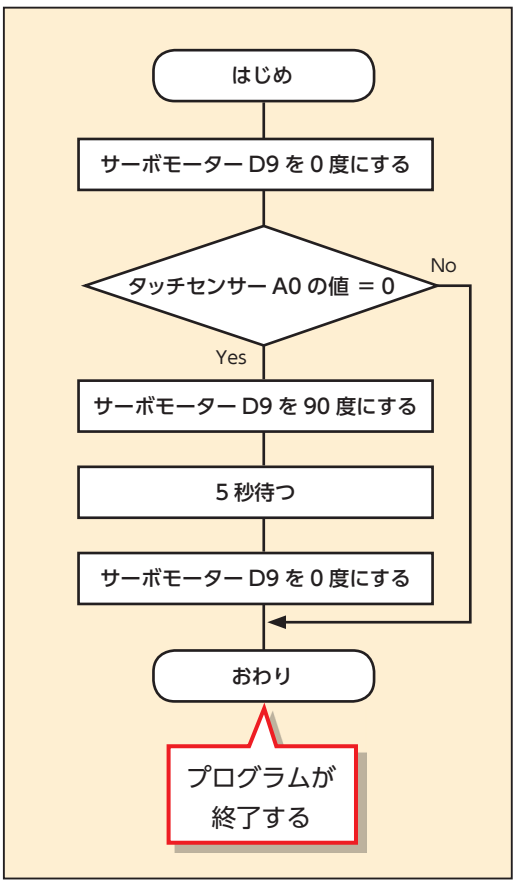
ステップ実行を開始してプログラムを実行すると、そのとき処理しているブロックが黄色く光るようになります。

デバッグを行うときは、ステップ実行機能を活用しましょう。

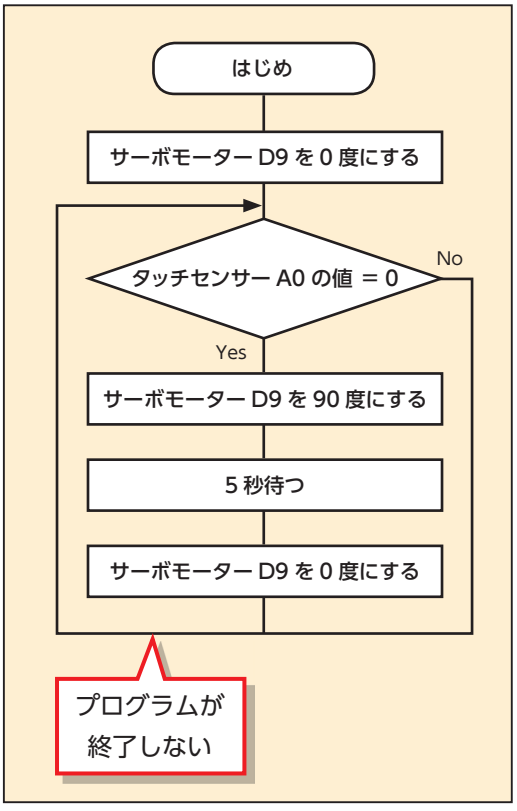
※ステップ実行時はプログラムの処理が極端に遅くなることに注意してください。

「ずっと」のブロックを入れないとどうなるのか？

「ずっと」を入れない場合のフローチャートとプログラムは以下のようになります。



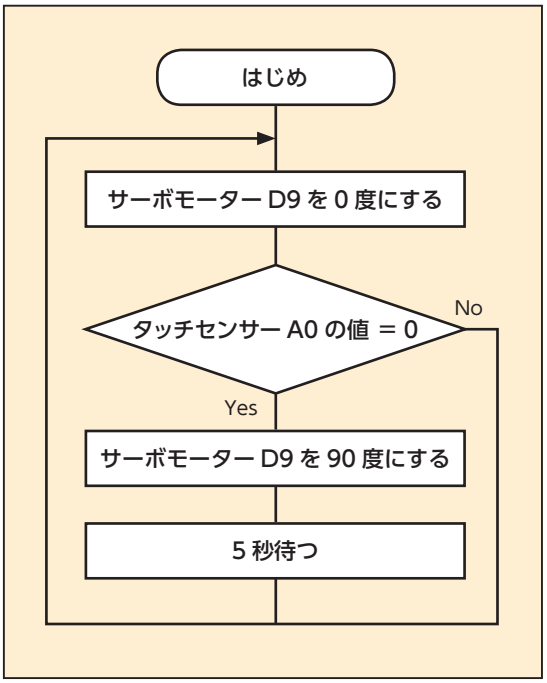
このプログラムを動作させると、タッチセンサーを押しても全く反応しなくなります。これは、プログラムが非常に高速で処理されることで、プログラムを動作させた瞬間にプログラムが終了してしまうからです。「ずっと」のブロックを入れることで、タッチセンサーの値を常に確認するようになるため想定通りの動作をするようになります。

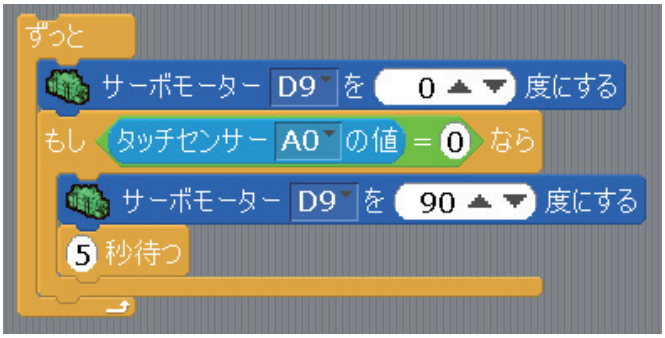


コンピューターはプログラム通りに動くので、思わぬ間違いで期待した動作と実際の動作が異なってしまうことがよくあります。フローチャートをまとめる段階で動作をよく考えることは間違いを避けることにつながります。

違うプログラムで同じ動きを実現する

以下のようなプログラムでも押しボタン式ゲートを実現することができます。





プログラムのつくり方は一つとは限りません。想定する動作が実現できればどのようなプログラムでも構いません。自分なりに考えてプログラムをつくることが大切です。

第5章
自動ゲートの製作

＜学習内容＞

- 赤外線フォトリフレクタ

1. 自動ゲートの製作

赤外線フォトリフレクタを利用して自動ゲートをつくりましょう。

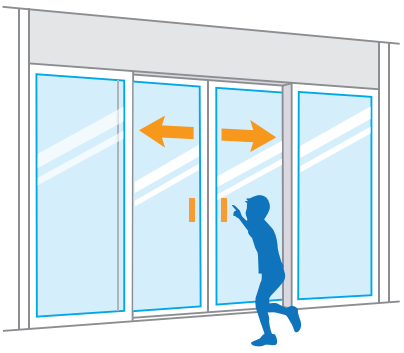
基本的な動作の流れは第４章でつくった押しボタン式ゲートと同じですが、タッチセンサーの代わりに赤外線フォトリフレクタを使用します。

自動ドアの種類

身近なところで使われている自動ドアのセンサーには様々な方式があり、代表的なものに、タッチ方式と光反射方式があります。

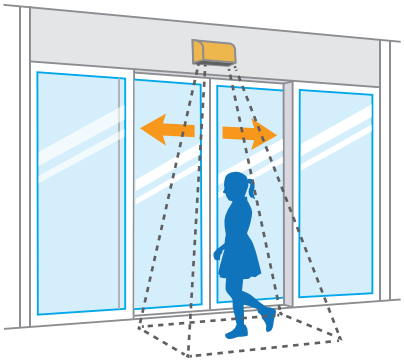
タッチ方式はタッチセンサーで実現でき、光反射方式は赤外線フォトリフレクタで実現することができます。

タッチ方式



ドアに取り付けられた
タッチスイッチを押すと開く

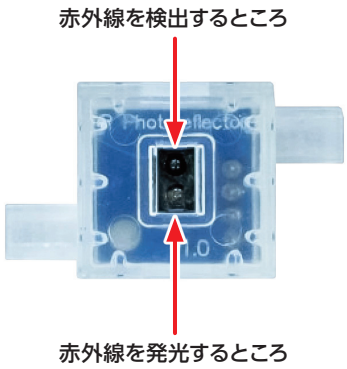
光反射方式

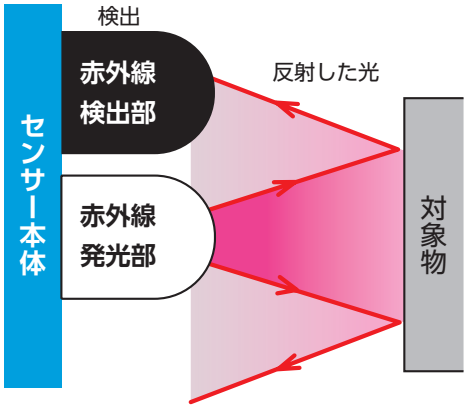


ドアの上に取り付けられたセンサーから出る
光を反射するものがあると開く

①赤外線フォトリフレクタとは

赤外線フォトリフレクタは「赤外線」という光の「反射」を利用したセンサーです。中央の2つの丸い部分は、透明な方が赤外線を発光し、黒い方は赤外線を検出します。発光部から出た赤外線が対象物にぶつかって反射してきたところを検出部で読み取ります。そのときの赤外線の強さで、対象物が存在するかどうかを知ることができます。

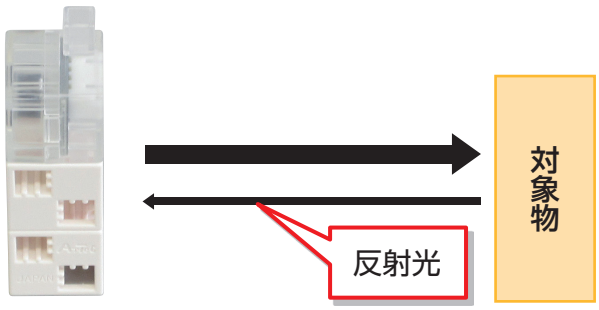




赤外線フォトリフレクタの使い方

・太陽光には赤外線が多く含まれています。太陽光が検出部に当たらないように注意してください。

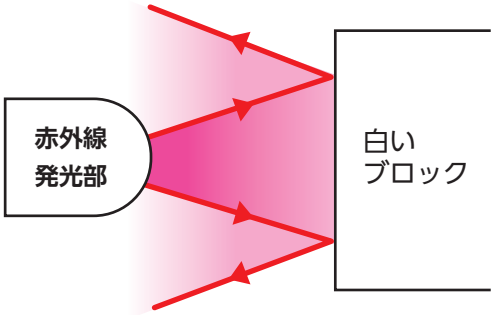
太陽光を強く検出してしまう



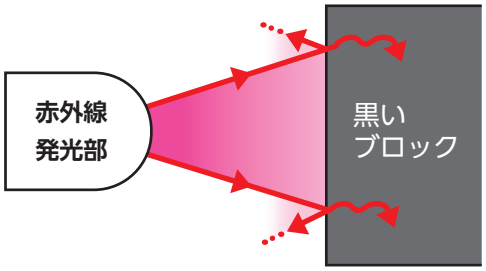
対象物で反射した光を強く検出できる

・黒系の色は赤外線を吸収しやすく反射しにくいため、他の色に比べて値が小さくなります。

反射する



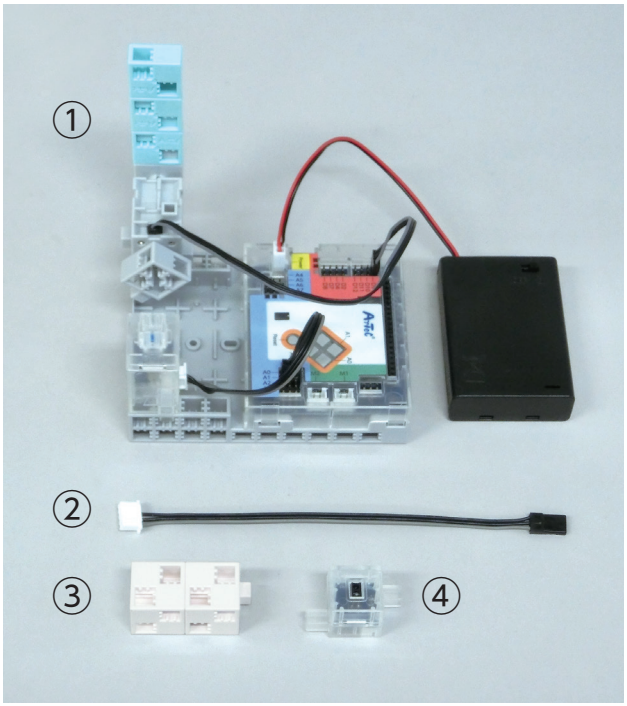
ほとんど反射せずに吸収する



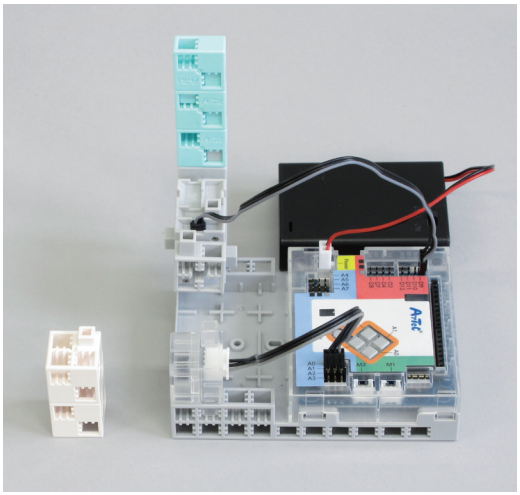
②組み立て

4章でつくった押しボタン式ゲートのタッチセンサーを赤外線フォトリフレクタに付け替えます。60 ページの組み立て説明を見て、自動ゲートを組み立てましょう。

組み立て準備

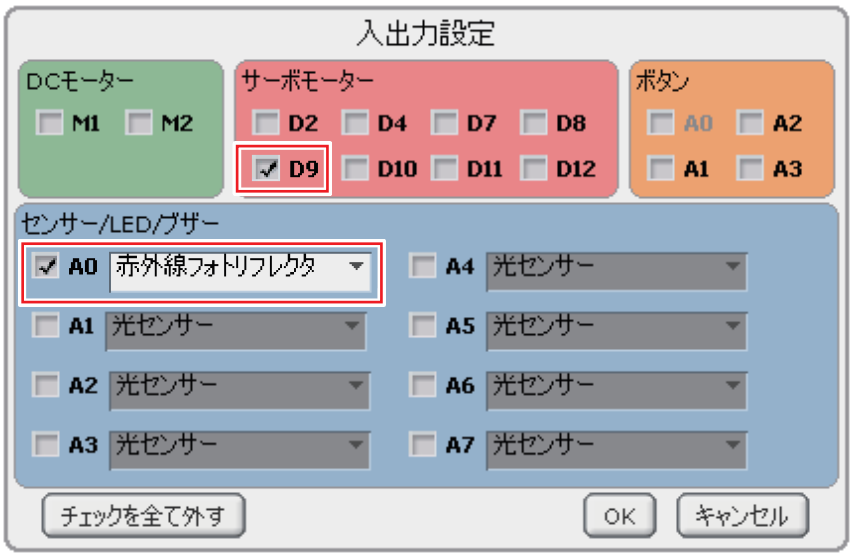


① ゲート（第4章のもの） …… 1
② センサーコード …… 1
③ 四角ブロック　白 …… 2
④ 赤外線フォトリフレクタ …… 1



③入出力設定

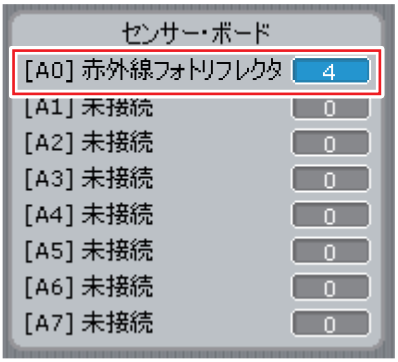
D9 にサーボモーター、A0 に赤外線フォトリフレクタとなるように設定しましょう。

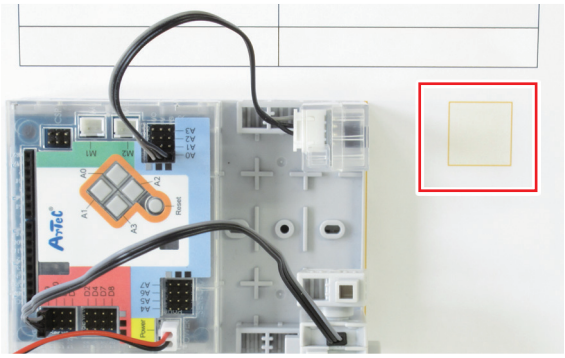
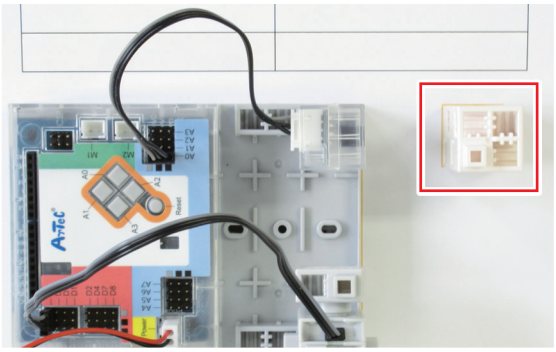


④赤外線フォトリフレクタの数値確認

自動ゲートはゲートの前に人が立ったときにセンサで感知してゲートを開きます。この「感知」の役割を赤外線フォトリフレクタで行います。テストモードにして、センサー・ボードで赤外線フォトリフレクタの値を確認しましょう。下のイラストに合わせて、Studuino（スタディーノ）基板と白いブロックを置きます。赤外線フォトリフレクタの前に白いブロックがないときとあるときの数値の変化を確認しましょう。

太陽光を背に向けた状態で調べると正しい数値を確認しやすくなります。

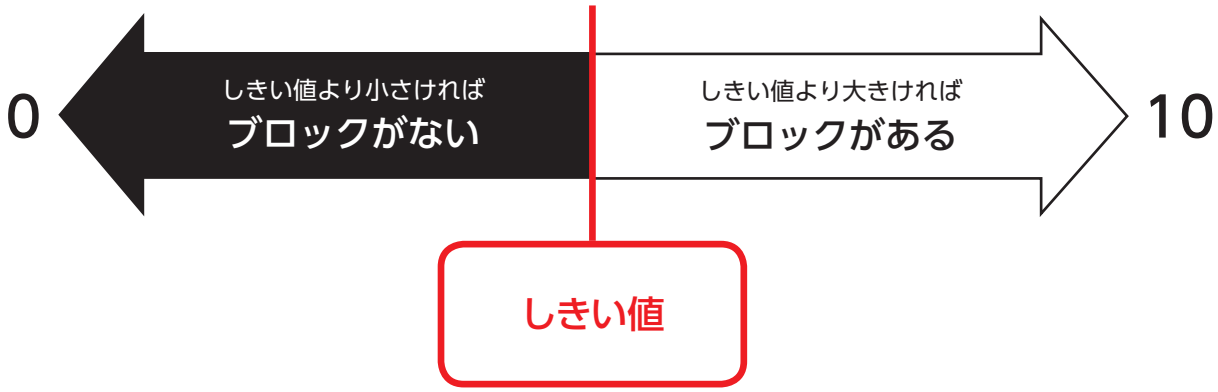


白いブロックを 置いていないときの数値	白いブロックを 置いているときの数値
	

※赤外線フォトリフレクタの個体差で同じ距離でも数値が異なる場合があります。

⑤しきい値の決定

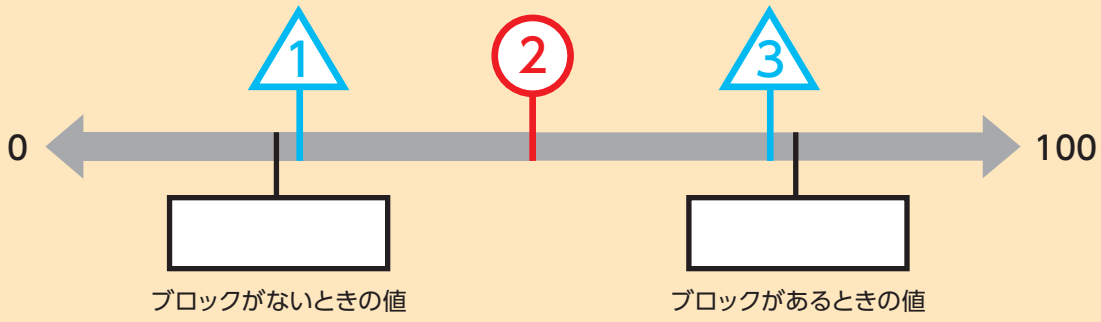
赤外線フォトリフレクタの値で白いブロックがないときとあるときを区別する場合は、２つの状態の境目となる値を決めて判断します。このような境目となる値のことを「しきい値」と言います。

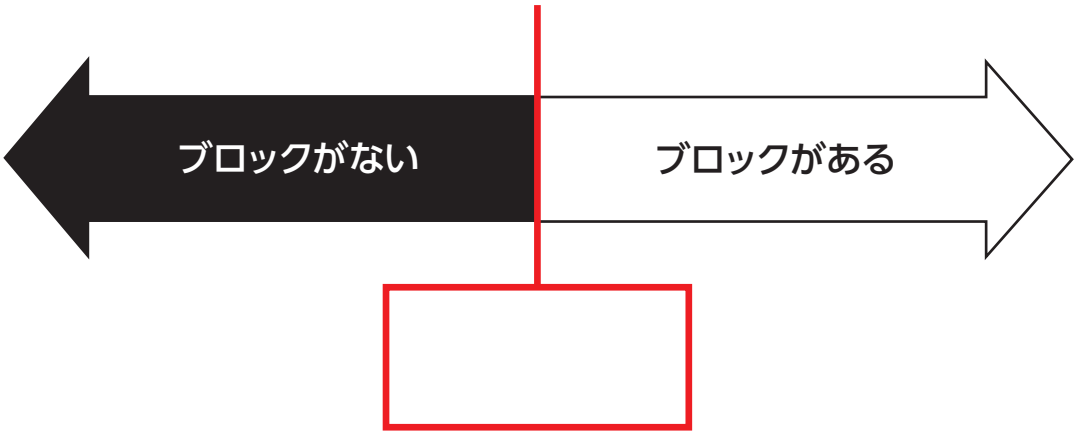


しきい値は、36 ページで調べた結果から考えます。

しきい値の決定

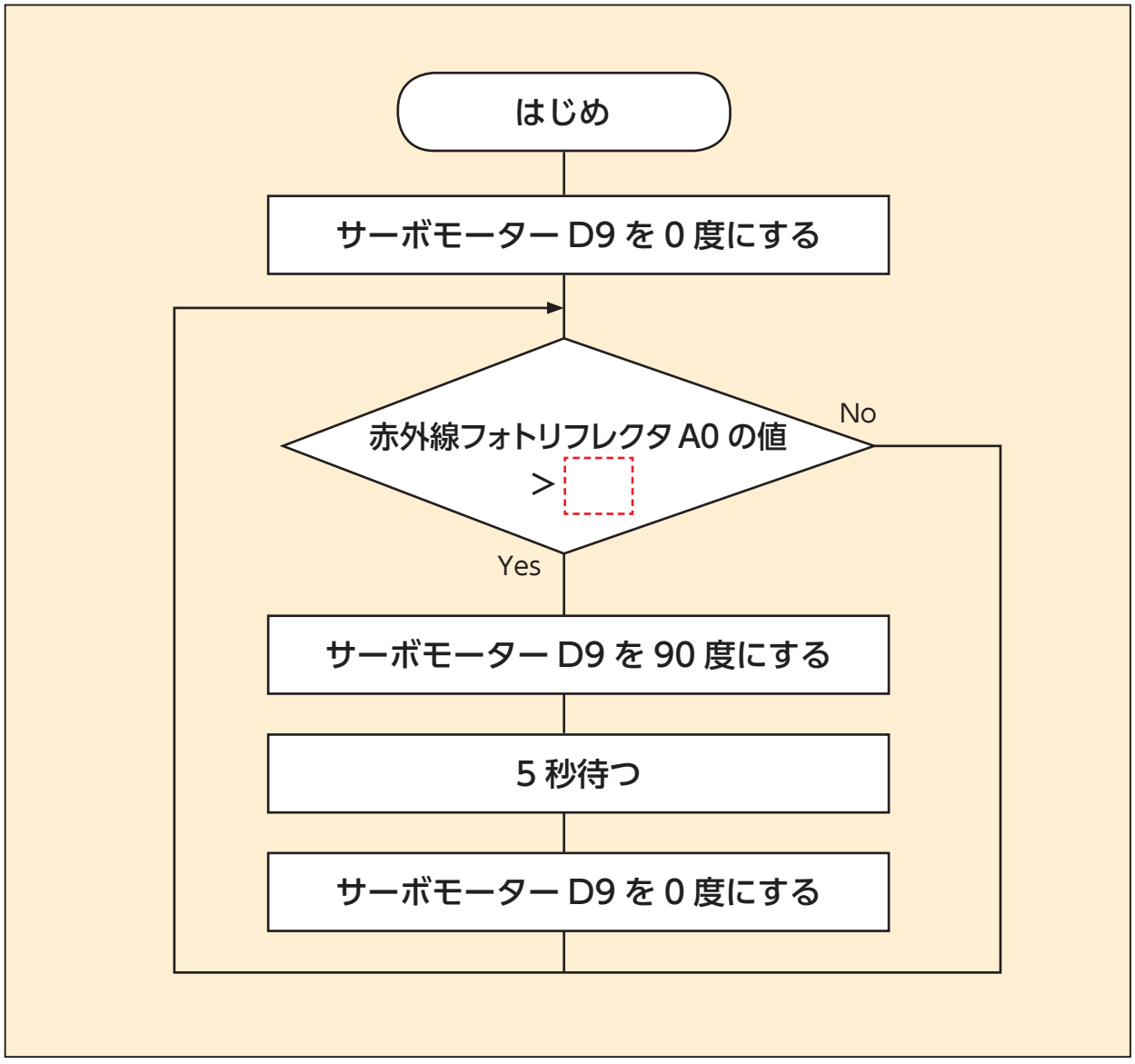
しきい値を ① や ③ のように調べたどちらかの値の近くに決めてしまうと、赤外線フォトリフレクタの値がそこから少し変わるだけで、ブロックの有無を区別してしまいます。センサーの値は周りの環境によって変わりやすいため誤って判断しないように、② のように調べた２つの値の中央あたりをしきい値として決めましょう。





⑥フローチャート

33 ページに示した動作を実現できるように手順を考えてフローチャートをまとめましょう。基本的なフローチャートの構造は 4 章でつくったものと同じで、センサーをタッチセンサーから赤外線フォトリフレクタに変更しただけです。条件の部分を決めたしきい値を利用して書き換えましょう。

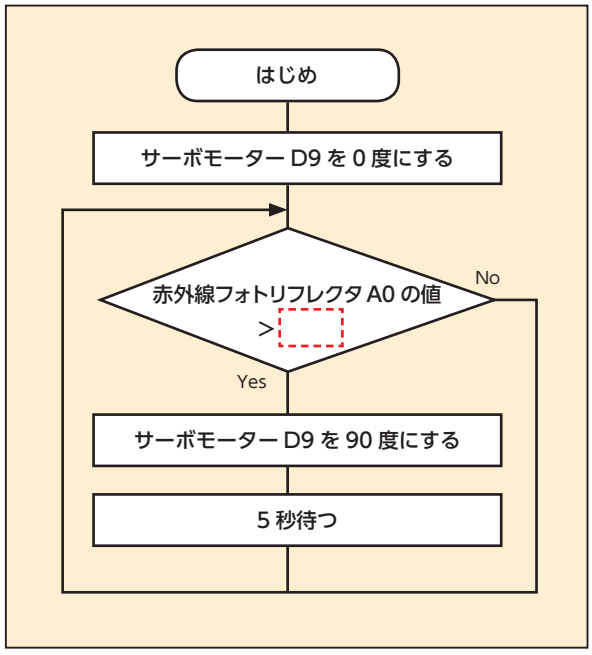


⑦プログラム作成

不等号のブロックを使うことに注意して、プログラムをつくりましょう。

自動ゲートの別解

以下のプログラムでも自動ゲートの動作が実現できます。



演習 1
踏切の製作

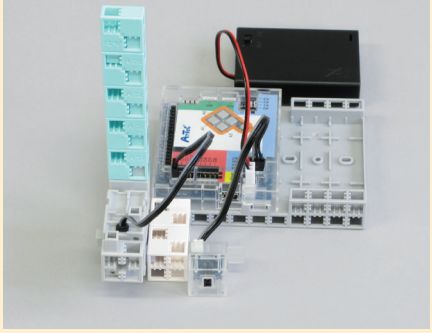
＜学習内容＞

- 踏切の製作

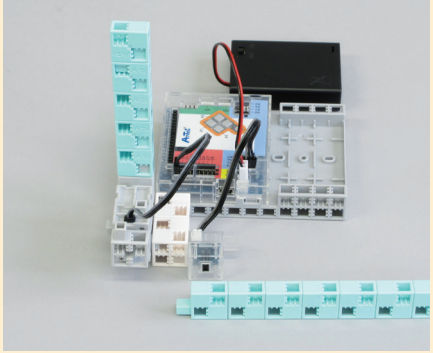
1. 踏切の製作

赤外線フォトリフレクタを使って踏切をつくりましょう。

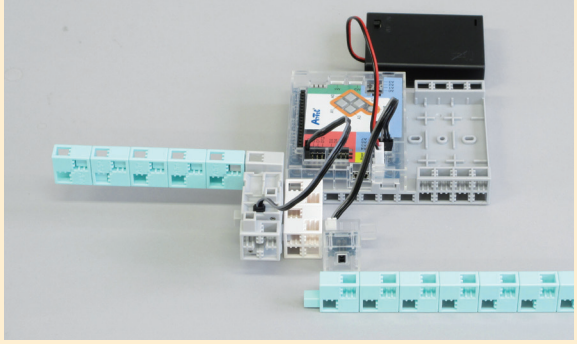
踏切を開ける



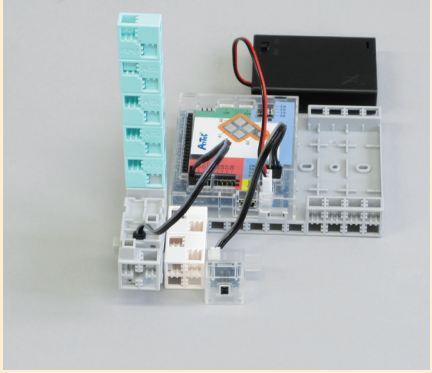
電車を感知する



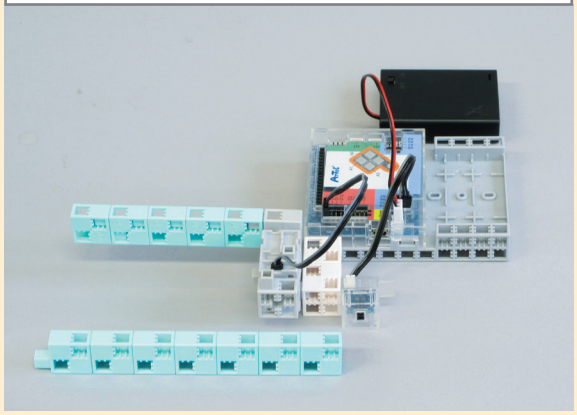
踏切を閉める



踏切を開ける



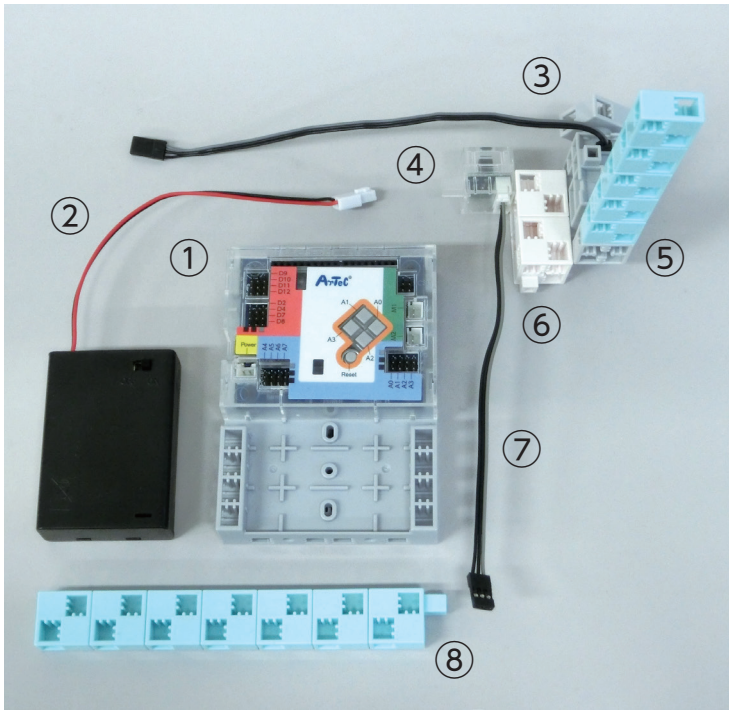
電車が通過するまで待つ



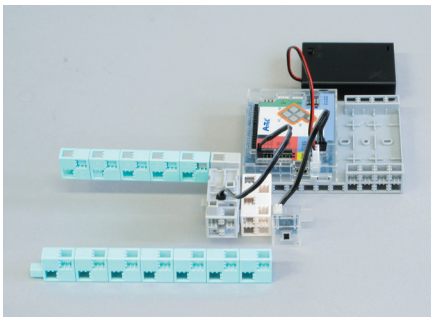
①組み立て

61 ページの組み立て説明を見て、踏切と電車を組み立てましょう。

組み立て準備

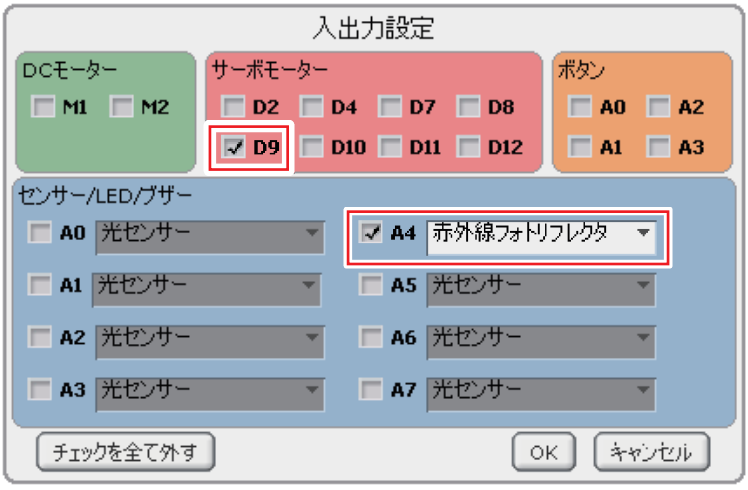


① Studuino（スタディーノ）基板…… 1
② 電池ボックス（電池入）…… 1
③ サーボモーター …… 1
④ 赤外線フォトリフレクタ …… 1
⑤ ハーフブロック　薄水…… 5
⑥ 四角ブロック　白 …… 2
⑦ センサーコード…… 1
⑧ ハーフブロック　薄水…… 7



②入出力設定

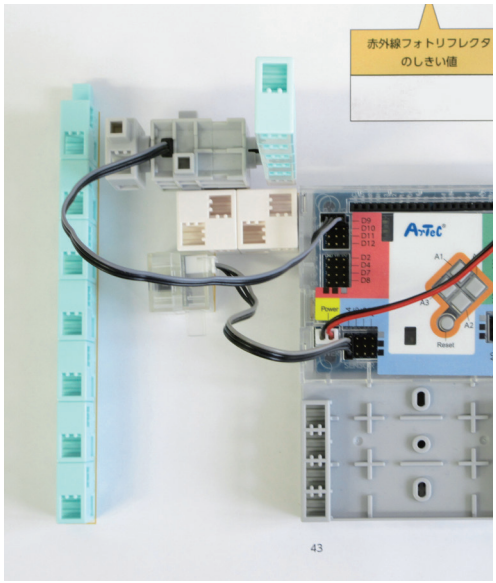
D9 にサーボモーター、A4 に赤外線フォトリフレクタとなるように設定しましょう。



※ A0 は演習 2 で使うため、ここでは赤外線フォトリフレクタを A4 に接続します。

③しきい値の決定

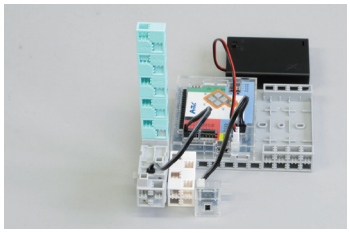
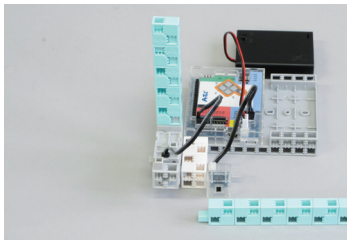
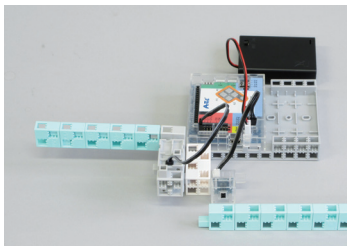
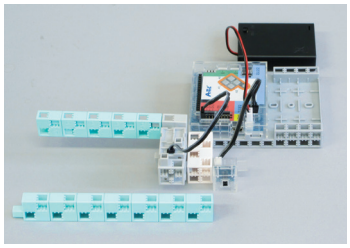
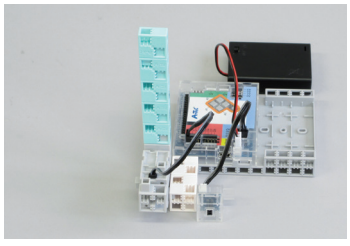
テストモードで電車が赤外線フォトリフレクタの前にいるときといないときの値を下のイラストを使って確認し、しきい値を決めましょう。



電車がいるときの値	電車がいないときの値

赤外線フォトリフレクタ のしきい値

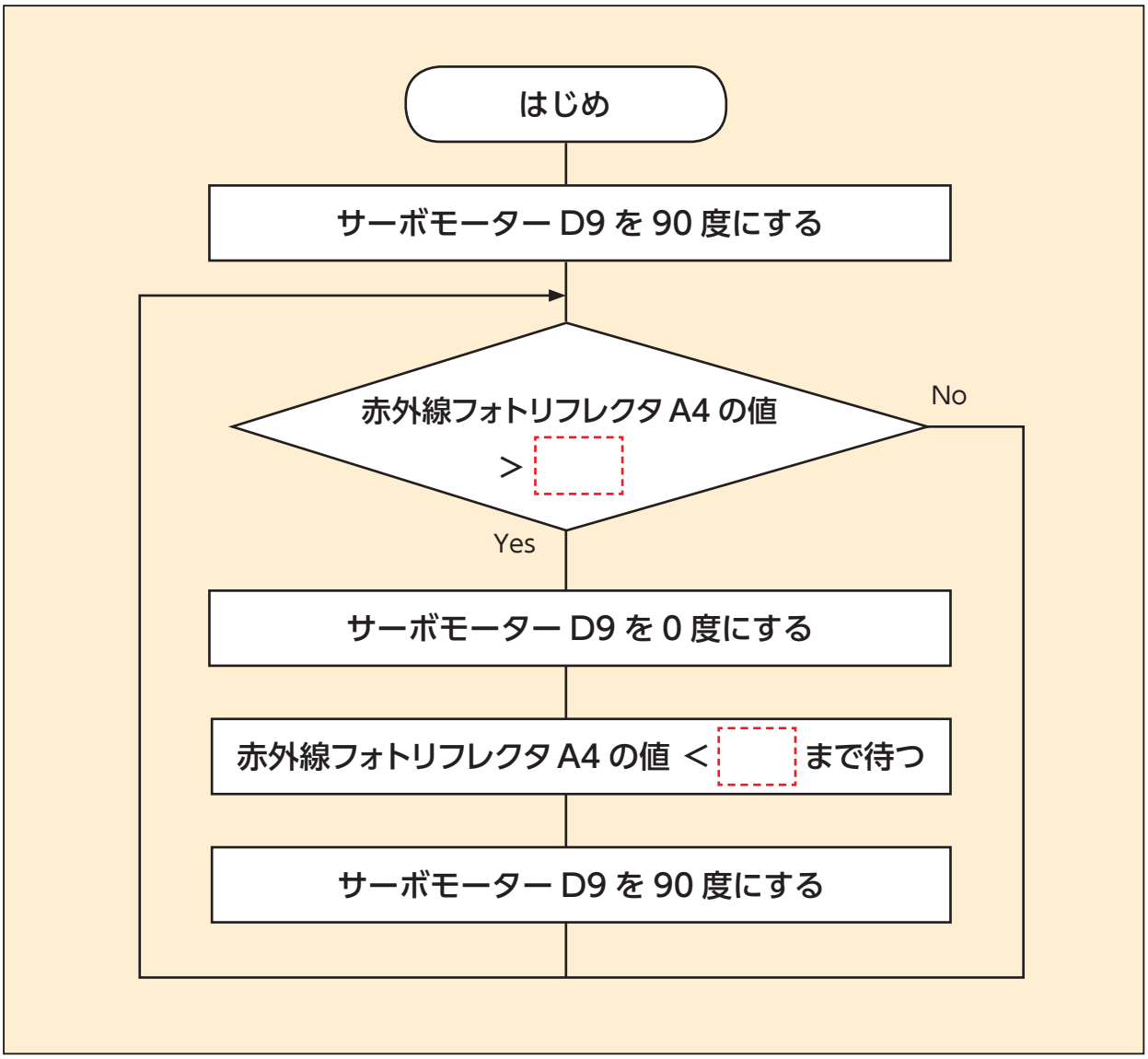
④踏切の動作の整理

順番	動作	プログラム
①	踏切を開く 	サーボモーター D9 を 90 度にする
②	電車を感知する 	赤外線フォトリフレクタ A4 の値 ＞ □
③	踏切を閉じる 	サーボモーター D9 を 0 度にする
④	電車が通過するまで待つ 	赤外線フォトリフレクタ A4 の値 ＜ □ まで待つ
⑤	踏切を開く 	サーボモーター D9 を 90 度にする

②～⑤を
ずっと繰り返す

⑤フローチャート作成

③で決めたしきい値と④で整理した動作をもとにフローチャートをまとめましょう。また、「電車が通過するまで待つ」は 赤外線フォトリフレクタの値＜ [　　] まで待つ を使いましょう。

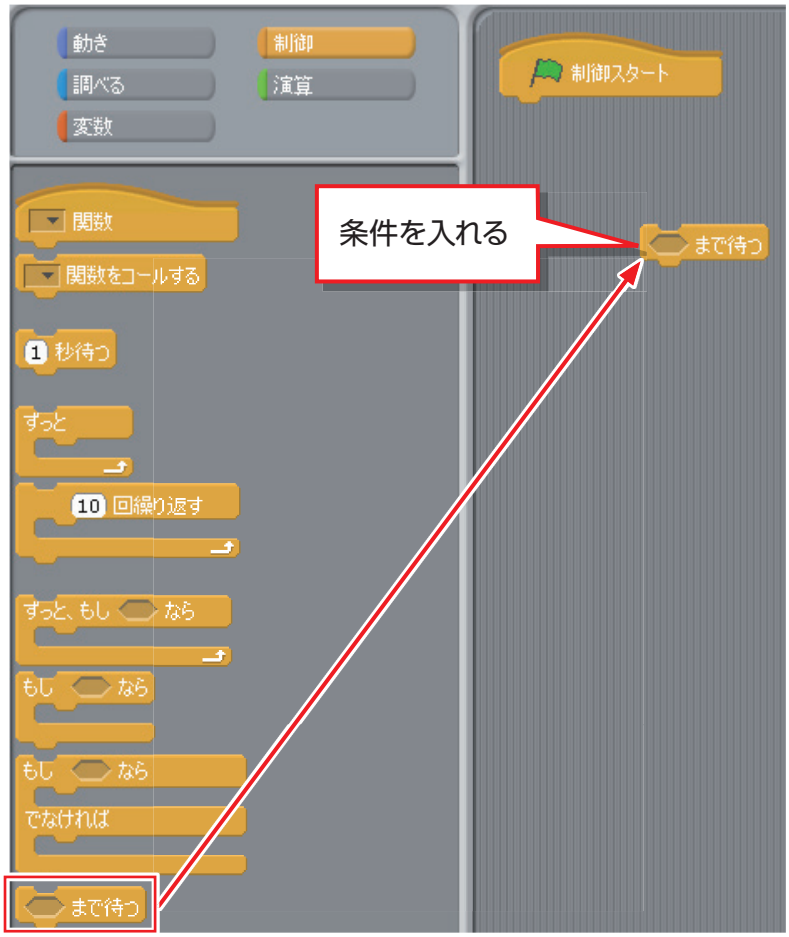


⑥プログラム作成

まとめたフローチャートをもとにプログラムをつくりましょう。 

を使うことに注意しましょう。

を使うと、指定した条件が成り立つまでブロックの処理が行われるのを止めるようになります。



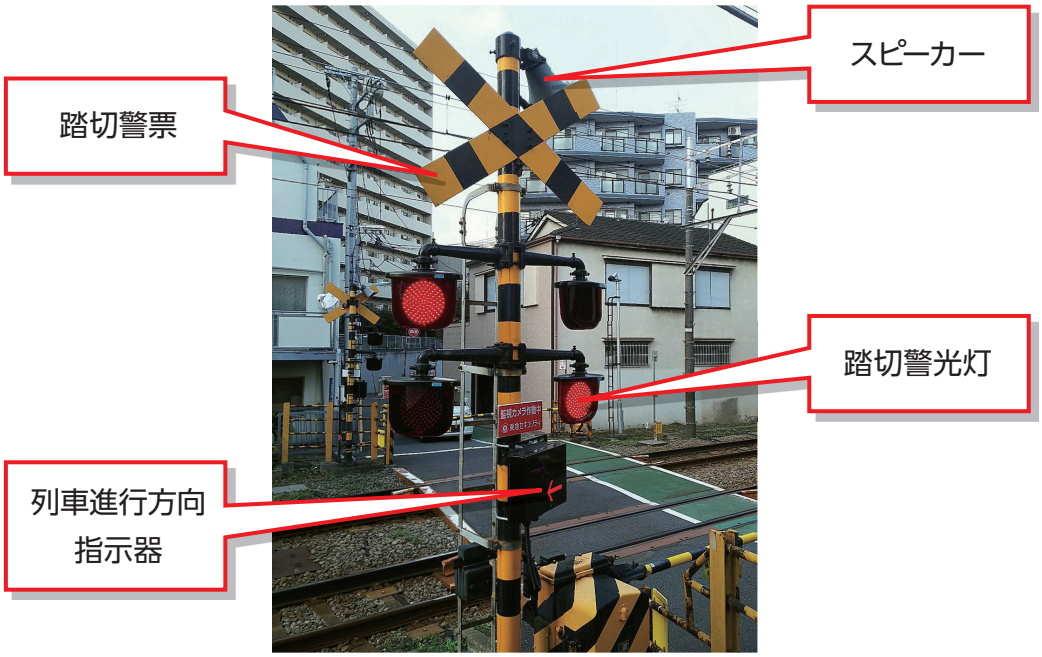
演習 2
踏切の改良

＜学習内容＞

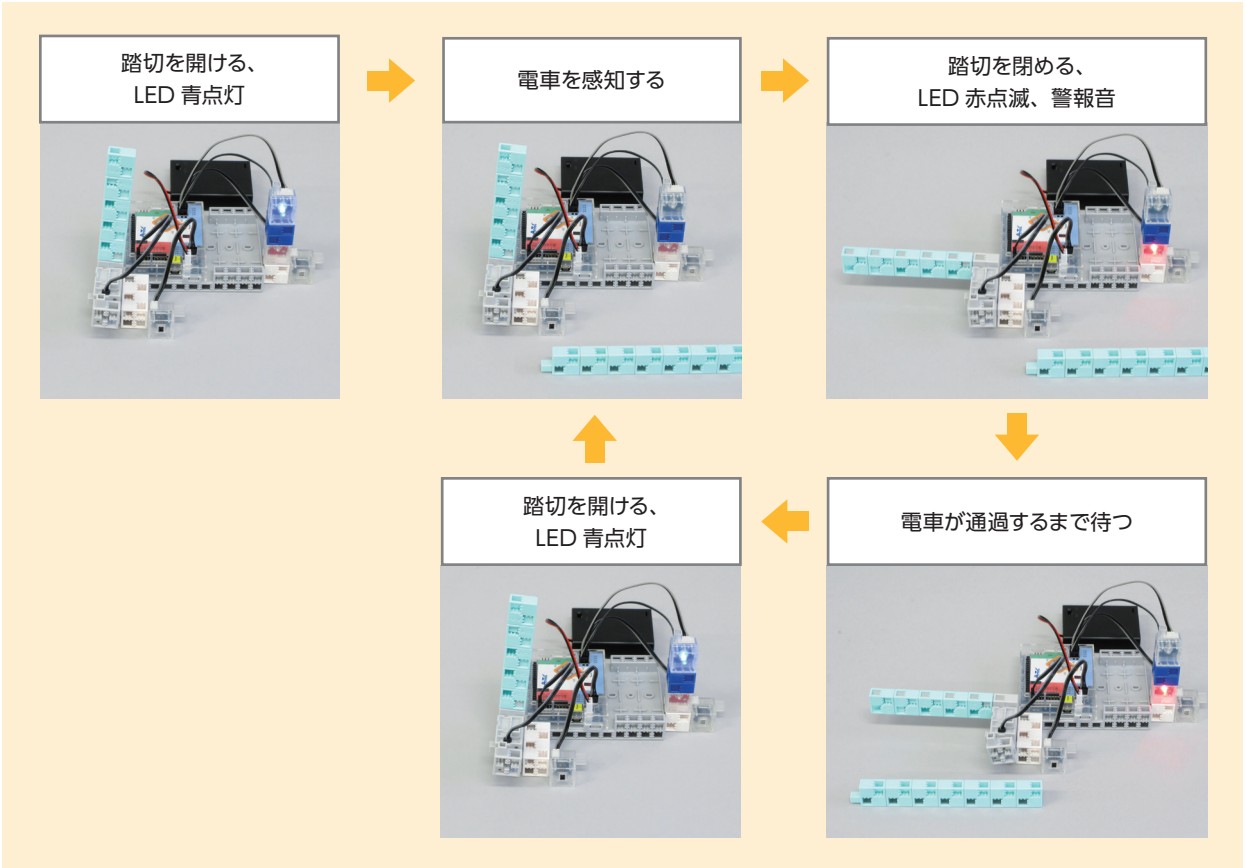
- LED
- ブザー
- 踏切の改良

1. 踏切の改良

演習 1 でつくった踏切をより事故の起こりにくいものに改良しましょう。現実の踏切はたくさんの工夫がされて、便利なものになっています。



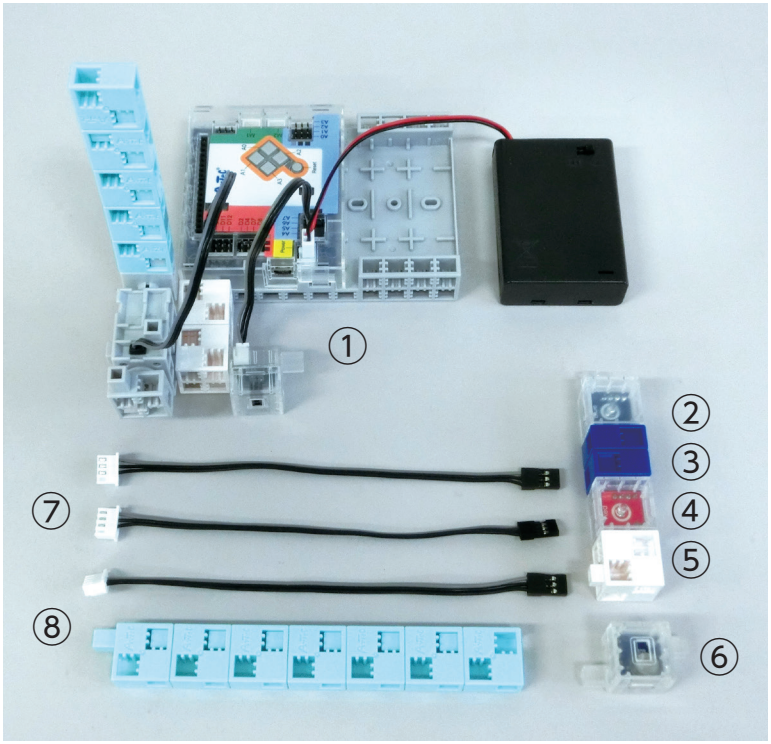
LED とブザーを取り付けて、開いているときや閉まっているときを光や音でも表しましょう。



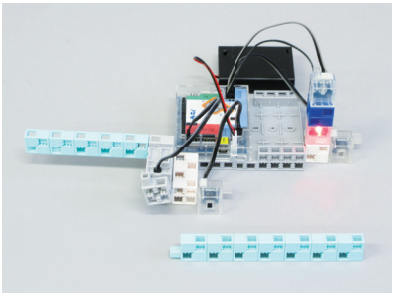
①組み立て

62 ページの組み立て説明を見て、演習 1 の①の踏切に赤と青の LED とブザーを追加しましょう。

組み立て準備



① 踏切（演習 1 のもの）…… 1
② LED 青 …… 1
③ ハーフブロック　青…… 2
④ LED 赤 …… 1
⑤ 四角ブロック　白 …… 1
⑥ ブザー …… 1
⑦ センサーコード…… 3
⑧ 電車（演習 1 のもの）…… 1

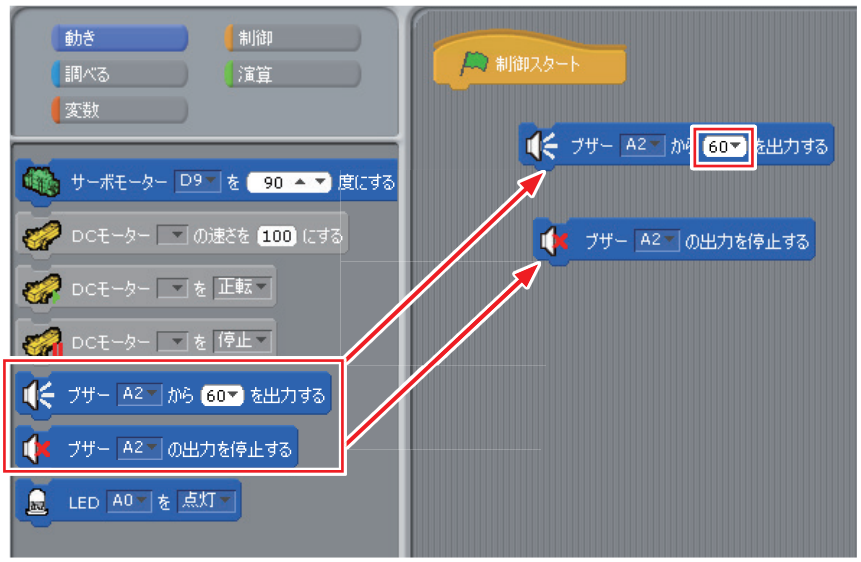


②入出力設定

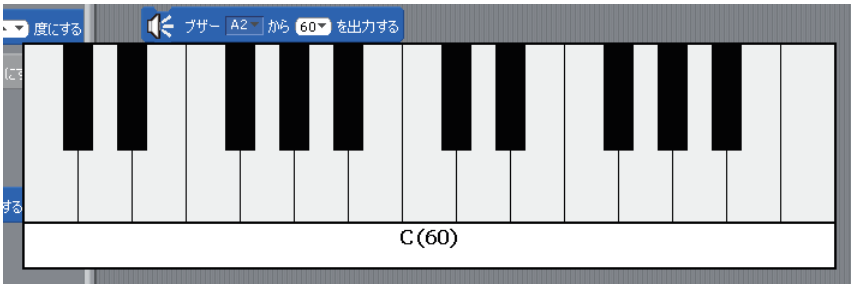
D9 にサーボモーター、A4 に赤外線フォトリフレクタ、A0 と A1 に LED、A2 にブザーとなるように設定しましょう。

③ブザーの使い方

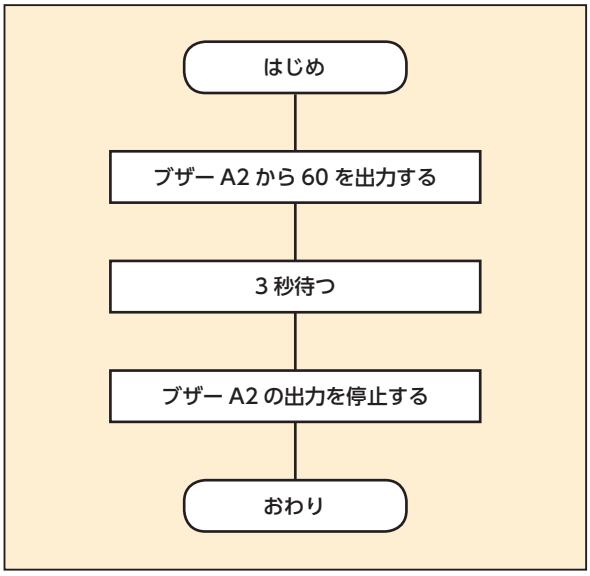
ブザーのブロックは「動き」カテゴリーにあり、出力と停止の 2 種類のブロックがあります。出力ブロックでは音の高さを決めることができます。



数字を直接変えるだけでなく、数字の横の▼をクリックしたときに表示される鍵盤でも音の高さは変更できます。

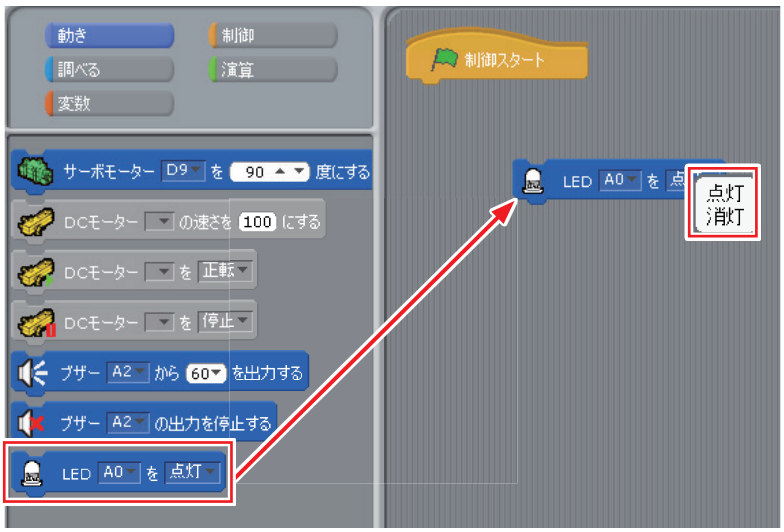


フローチャートをもとにブザーを 60（ド）の高さで3秒鳴らすプログラムをつくりましょう。

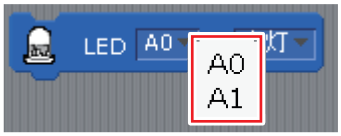


④ LED の使い方

LED のブロックは「動き」カテゴリーにあり、点灯と消灯を選ぶことができます。

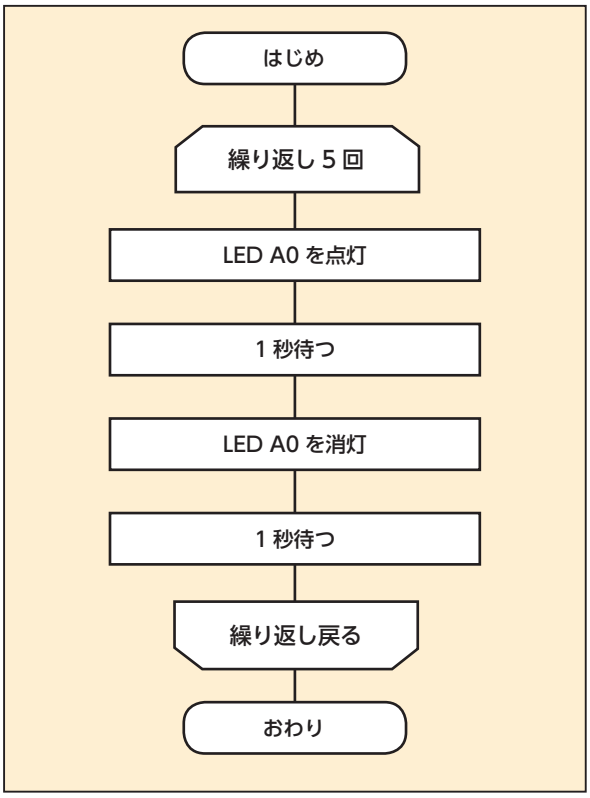


また、今回 LED は赤と青の2種類あるため、どちらを光らせるか選択する必要があります。どちらの色の LED をどのコネクターに接続したかを確認しておきましょう。



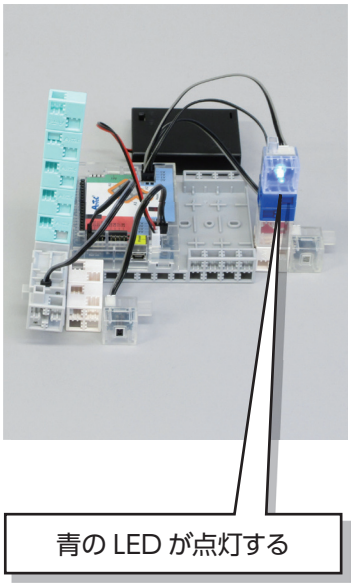
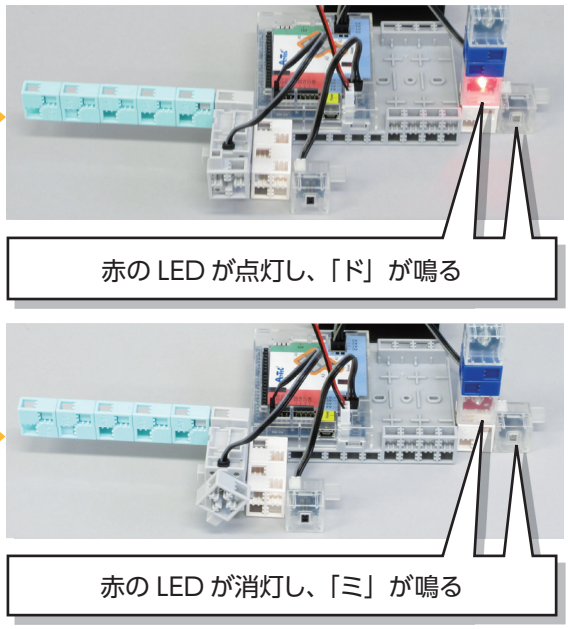
赤の LED のコネクター	青の LED のコネクター
A0	A1

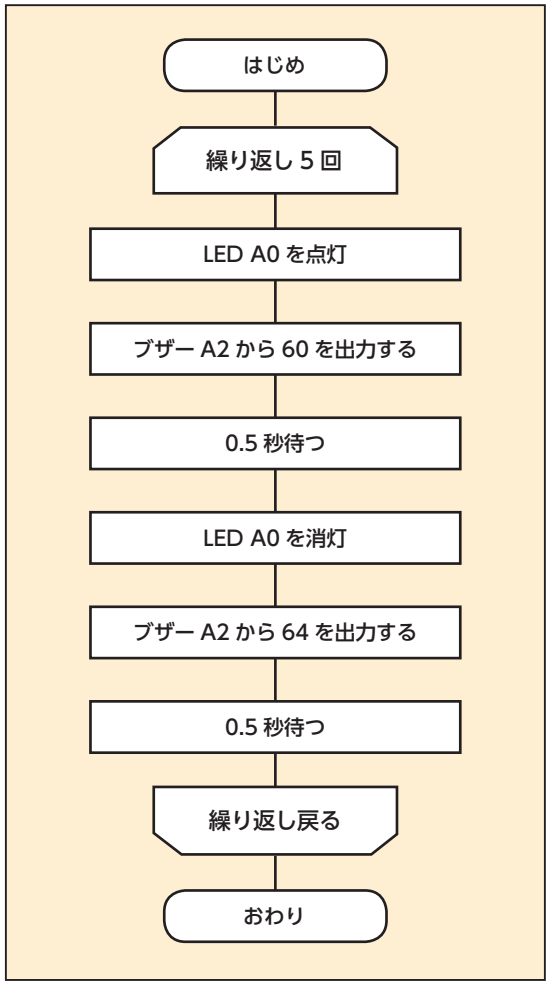
フローチャートをもとに赤の LED を5回点滅させるプログラムをつくりましょう。



⑤踏切の音と光のプログラム

今回は以下のような動作を行う踏切をつくます。

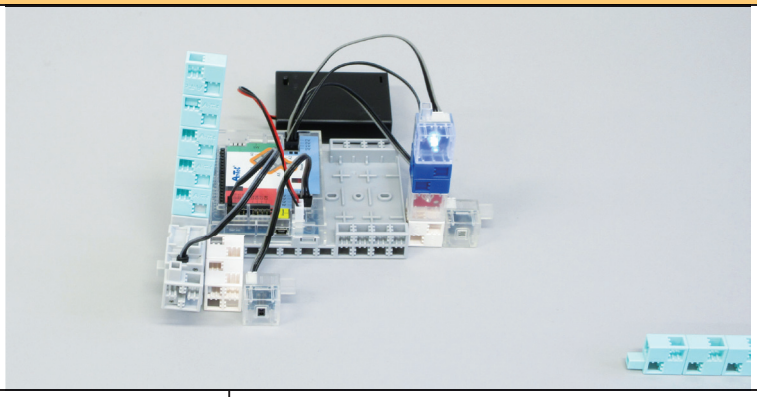
踏切の状態	踏切が開いているとき	踏切が閉まっているとき
LED とブザーの動作	 青の LED が点灯する	 赤の LED が点灯し、「ド」が鳴る 0.5 秒待つ 赤の LED が消灯し、「ミ」が鳴る

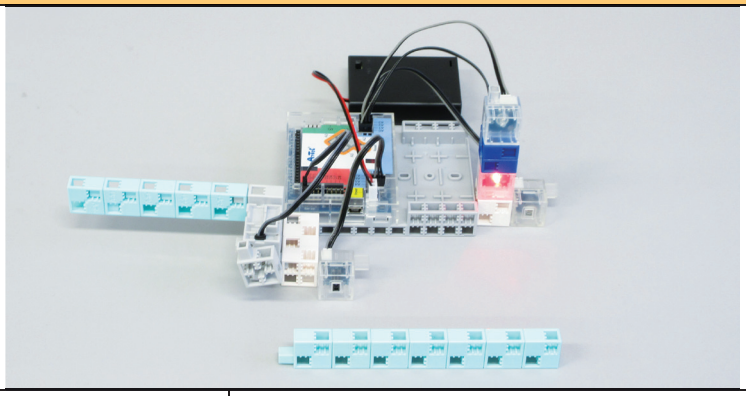


踏切が閉まっているときの動作が複雑なので整理しましょう。写真を見て閉まっているときの LED とブザーのフローチャートを作成して、プログラムをつくりましょう。「ド」は 60 の音、「ミ」は 64 の音であることに注意しましょう。

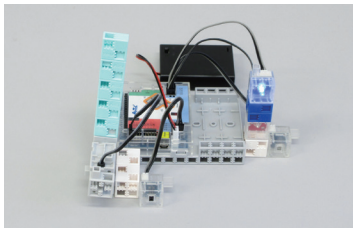
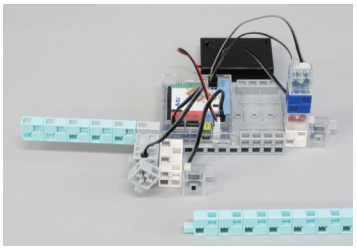
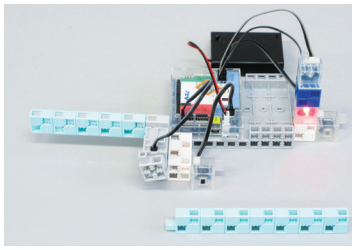
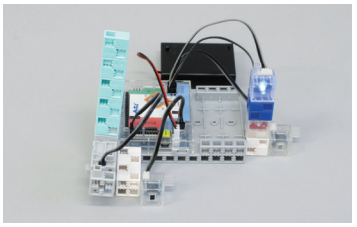
⑥動作の整理

動作とプログラムの関係を整理しましょう。

電車が踏切を通過していないとき		
		
踏切	開く	サーボモーター D9 を 90 度にする
LED 青	点灯	LED　A1 を点灯
LED 赤	消灯	LED　A0 を消灯
ブザー	鳴らない	ブザー A2 の出力を停止する

電車が踏切を通過しているとき		
		
踏切	閉まる	サーボモーター D9 を 0 度にする
LED 青	消灯	LED　A1 を点灯
LED 赤	点滅	LED　A0 を点灯 LED　A0 を消灯 を 0.5 秒ごとに繰り返す
ブザー	点滅に合わせて鳴る	LED A0 が点灯時に「60（ド）」 LED A0 が消灯時に「64（ミ）」 で鳴る
条件	赤外線フォトリフレクタ A4 の値がしきい値を超えているとき	

まとめた動作を時系列順に並べると、以下のようになります。

踏切と電車の状況		サーボモーター	LED	ブザー
①	はじめの状態 	90 度	青点灯	鳴らない
②	近づいたとき 	0 度	青消灯	鳴らない
③	通過しているとき 	0 度	赤点滅	点灯時→ド 消灯時→ミ
④	通過し終わったとき 	90 度	青点灯	鳴らない

もし電車が踏切に近づいたら

電車が踏切を通過するまで

電車が踏切を通過するまで繰り返す

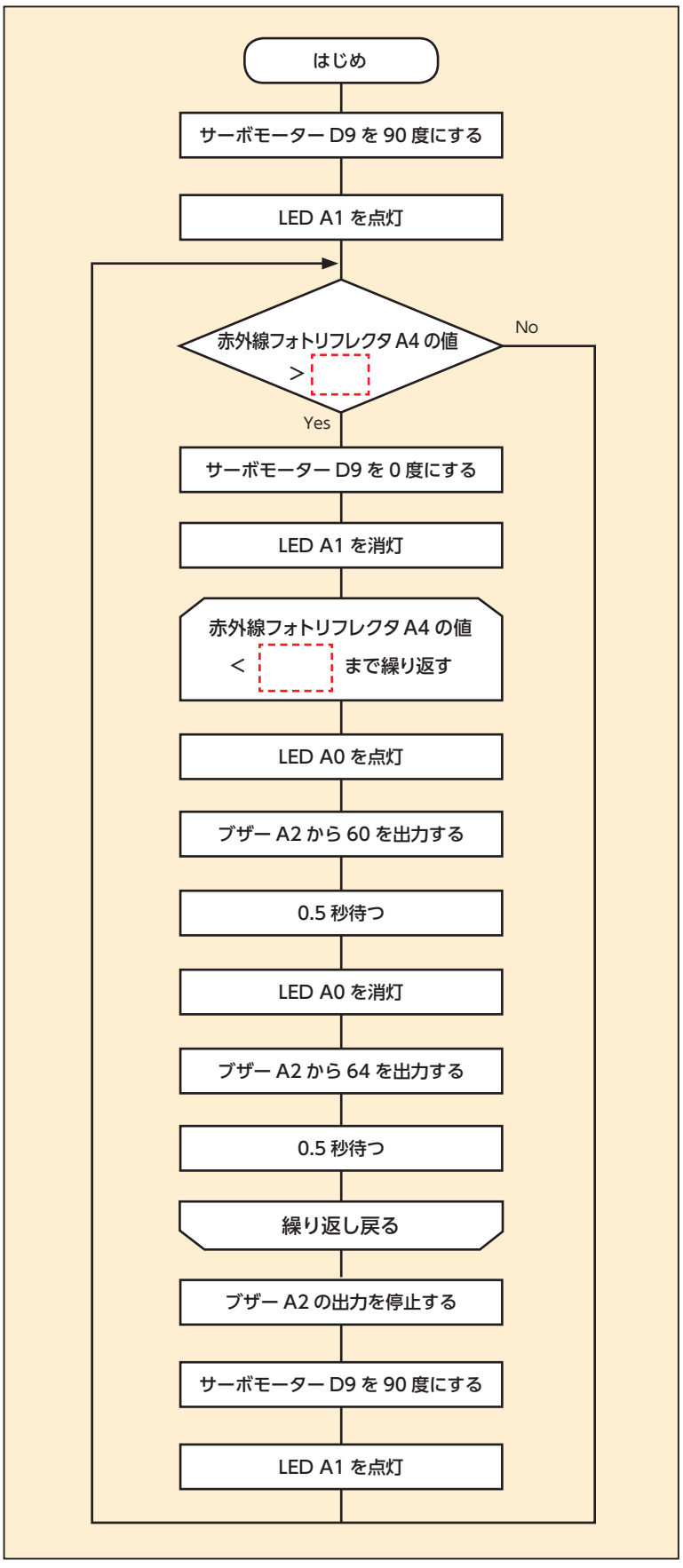
表から、赤の LED の点滅とブザーからの音は、電車が踏切を通過し終わるまで繰り返す必要があることがわかります。

「踏切を通過し終わるまで繰り返す」は、制御カテゴリーの 

を使うことで表現することができます。

⑦フローチャート作成

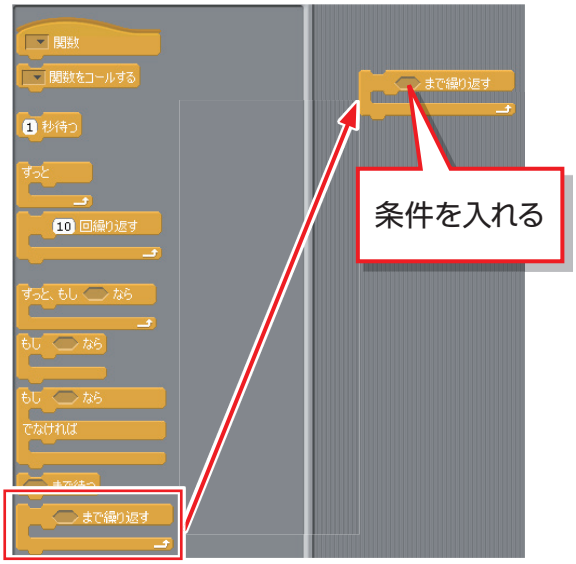
⑥でまとめた動作を参考にフローチャートを完成させましょう。



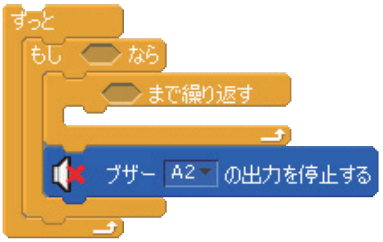
⑧プログラム作成

⑦でまとめたフローチャートをもとに、プログラムを完成させましょう。 

を使うことに注意しましょう。



うまく動かない場合、以下のことを確認しましょう。

LED のコネクターを間違えていないか	
点灯させた LED を消灯させるのを忘れていないか	
ブザーの出力を停止することを忘れていないか	
ブザーの出力を停止するブロックを 入れる場所を間違えていないか	
しきい値を使って表した条件で 不等号が逆になっていないか	

※ステップ実行機能でプログラムを実行していると、プログラムの読み込み速度が落ちるため踏切を閉じているときの挙動が遅れることがあります。

プログラムの別解

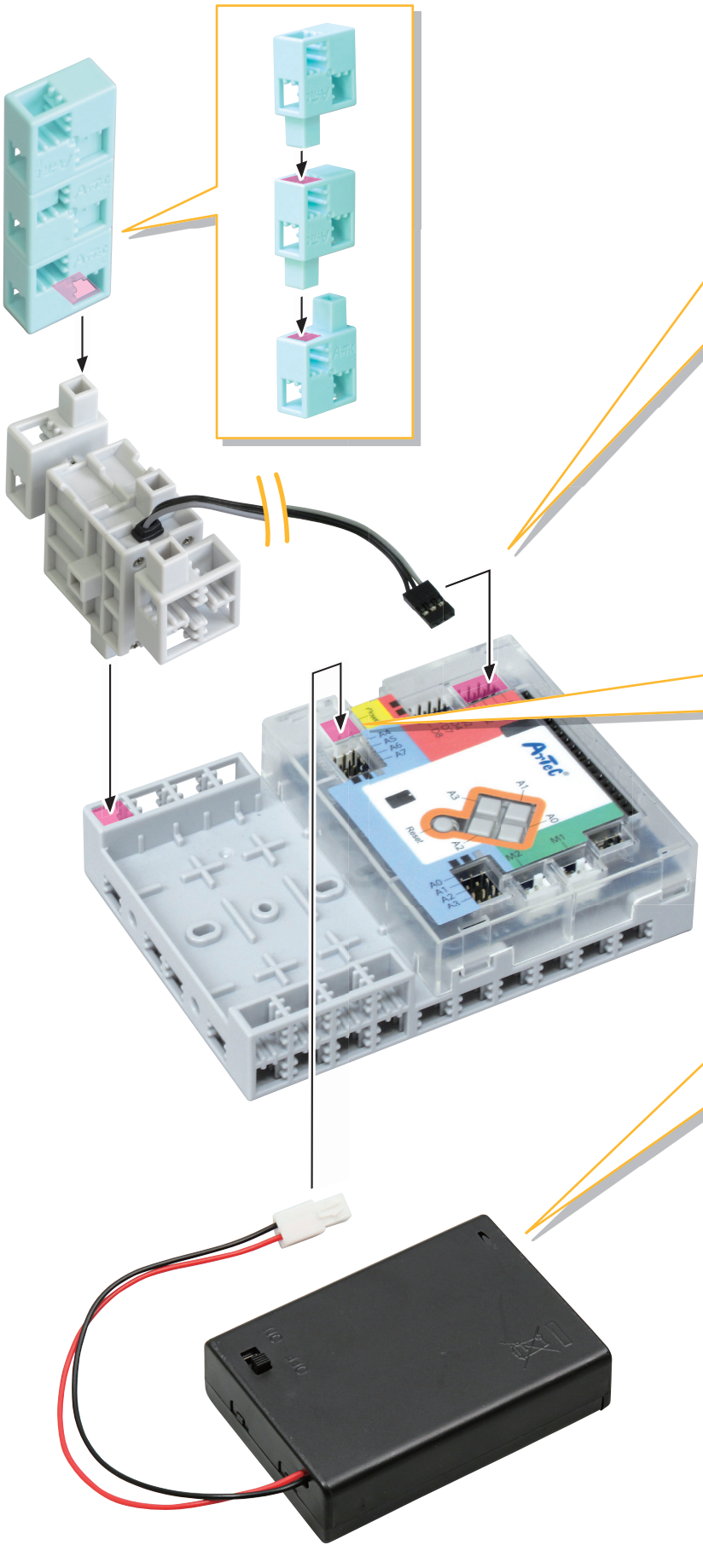
以下のプログラムでも同様の動作が実現できます。今回の改良では他にも色々なプログラムが考えられます。⑥で整理した動作が実現できていればすべて正解なので、フローチャートをよく考えて自分なりのプログラムを完成させましょう。

組み立て説明

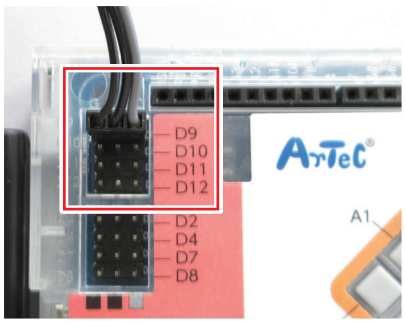
＜関連する章＞

- 第 2 章～第 5 章
- 演習問題

第 2 章 ゲートの組み立て

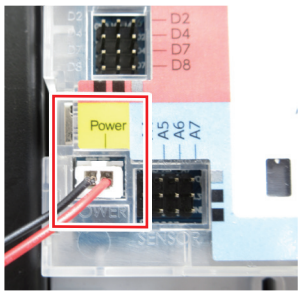


サーボモーターのコネクターを「D9」に接続する。

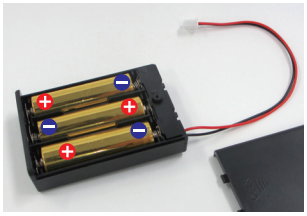


⚠ 灰色のコードの向きに注意

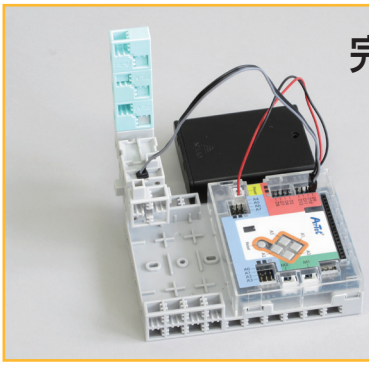
電池ボックスのコネクターを「Power」に接続する。



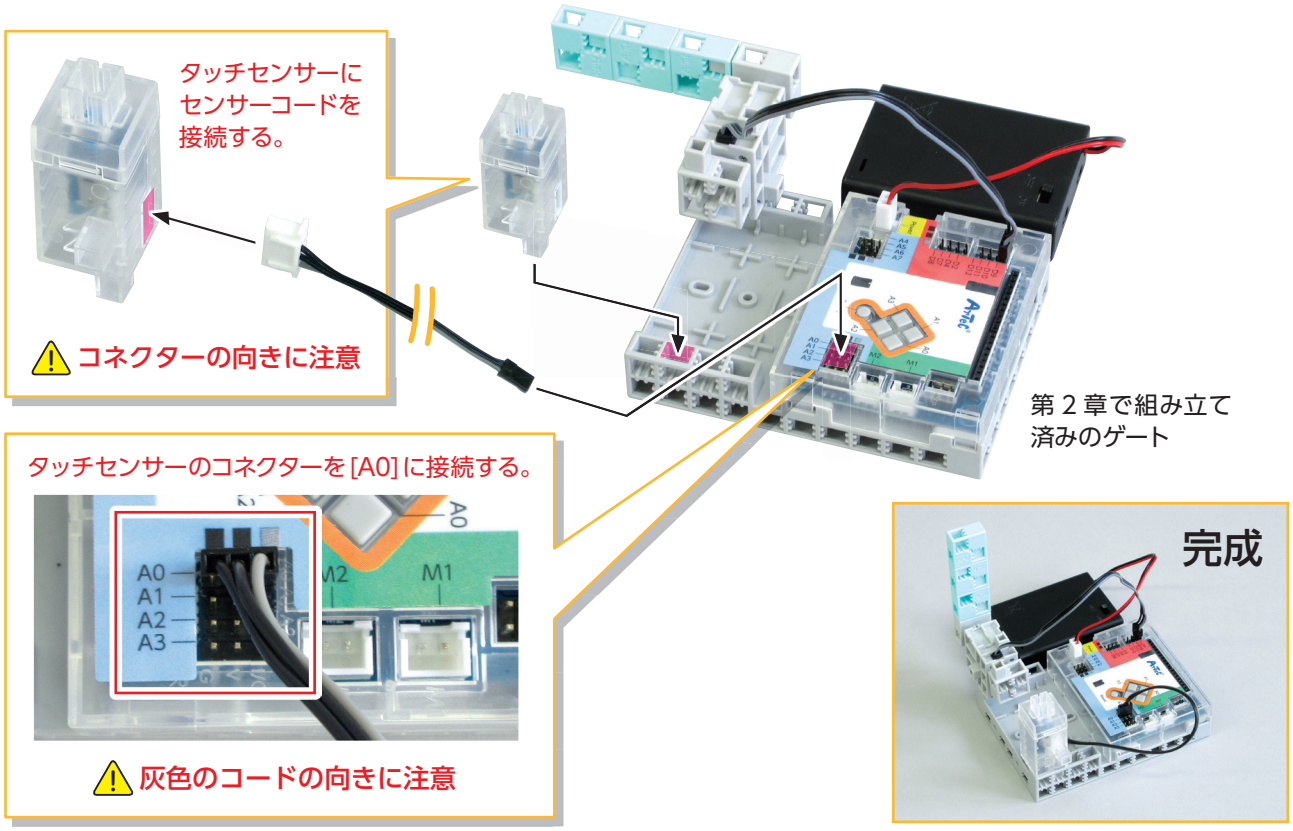
⚠ コネクターの向きに注意



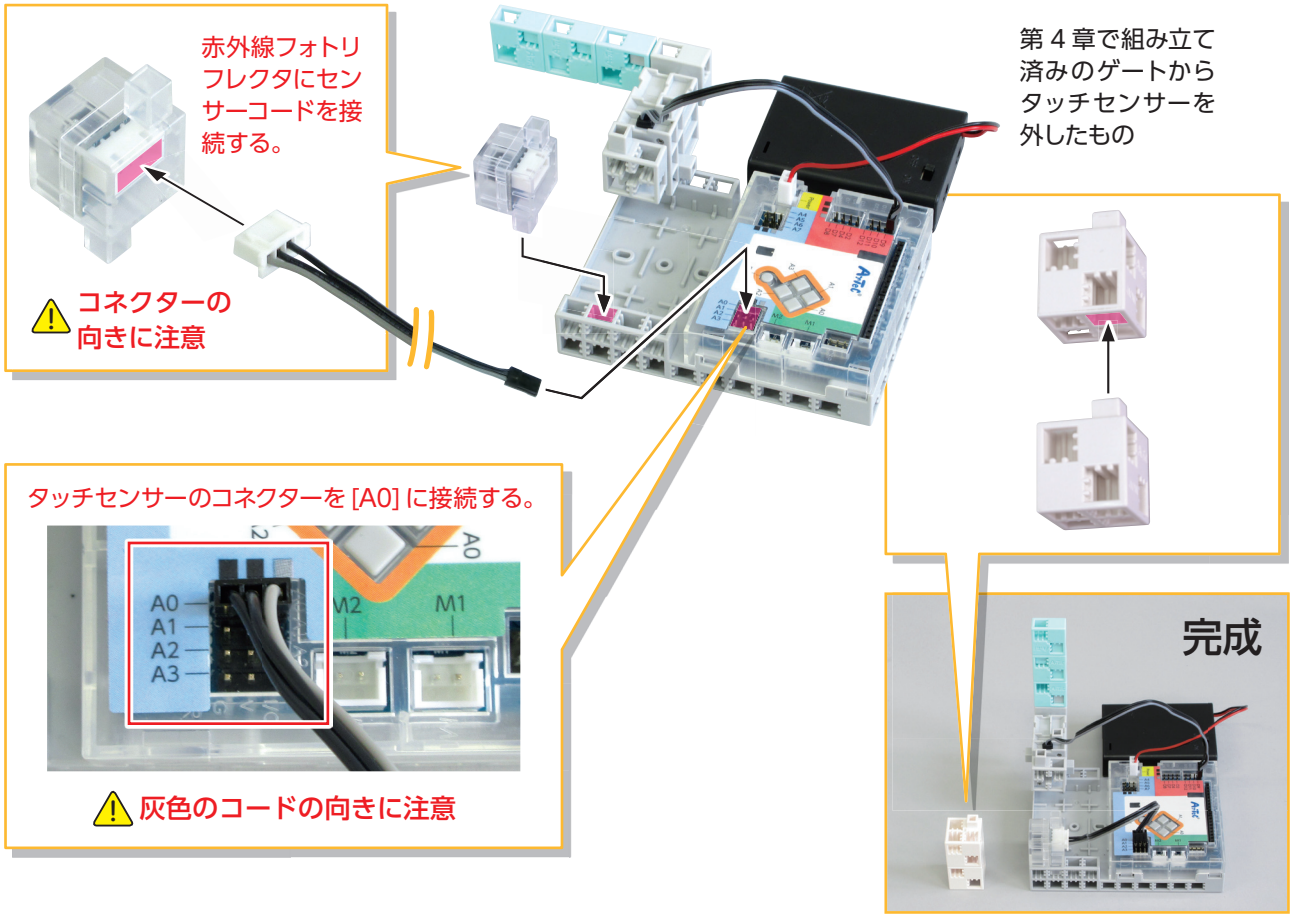
⚠ ➕プラス ➖マイナスの向きに注意



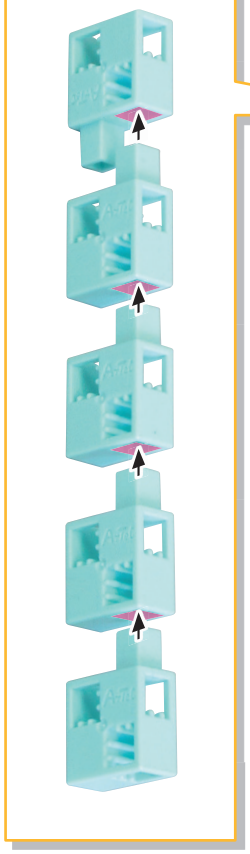
第 4 章 押しボタン式ゲートの組み立て

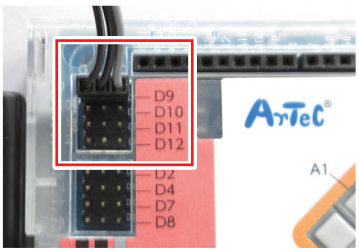


第 5 章 自動ゲートの組み立て



演習 1　踏切と電車の組み立て





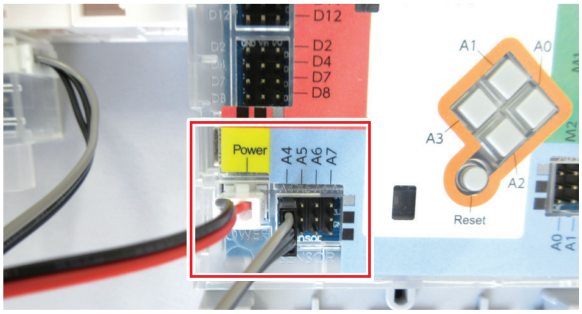
サーボモーターのコネクターを「D9」に接続する。

⚠ 灰色のコードの向きに注意

赤外線フォトリフレクタにセンサーコードを接続する。

⚠ コネクターの向きに注意

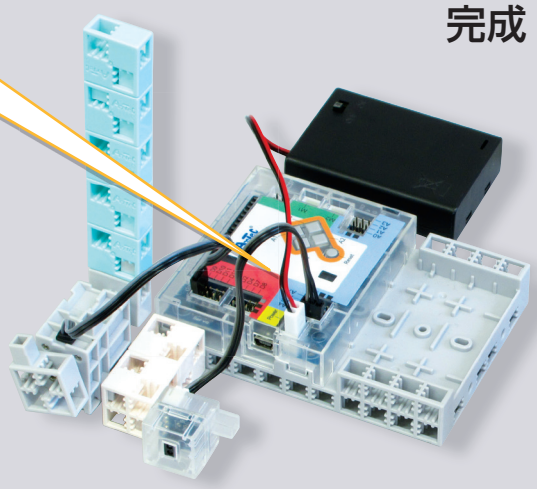
赤外線フォトリフレクタのコネクターを [A4] に電池ボックスのコネクターを [Power] に接続する。

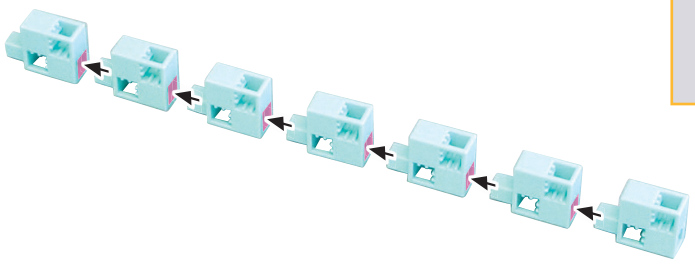


⚠ 灰色のコードの向きに注意

⚠ コネクターの向きに注意

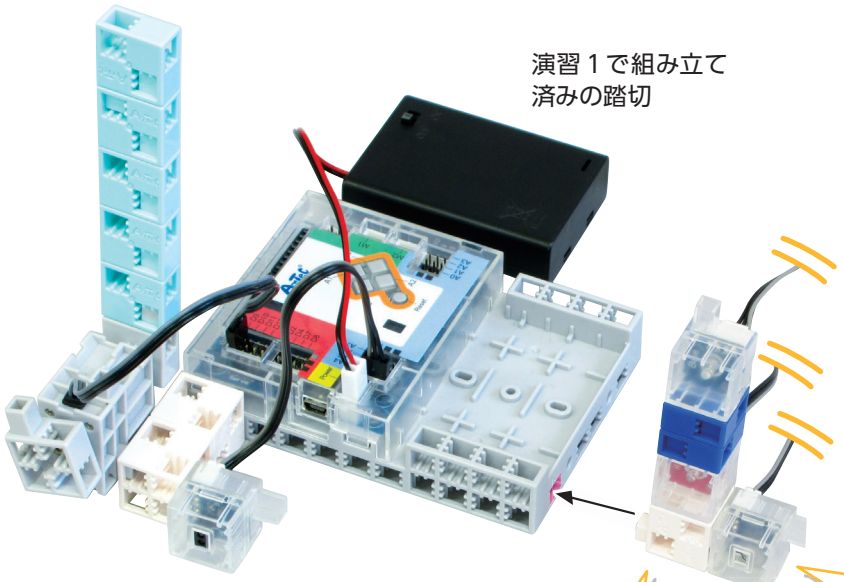
完成



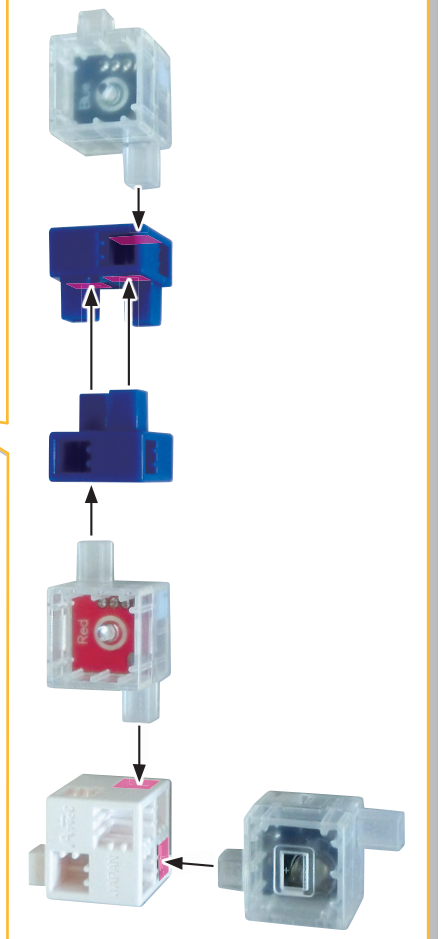


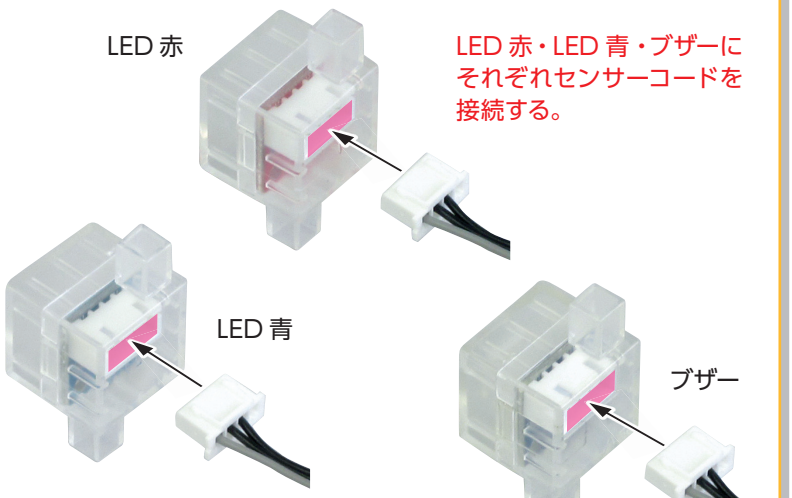
電車

演習 2　踏切の改良



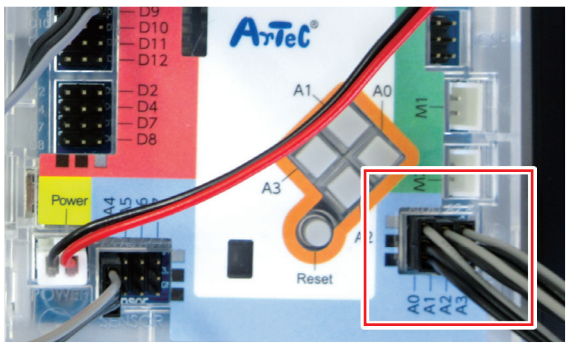
演習 1 で組み立て
済みの踏切



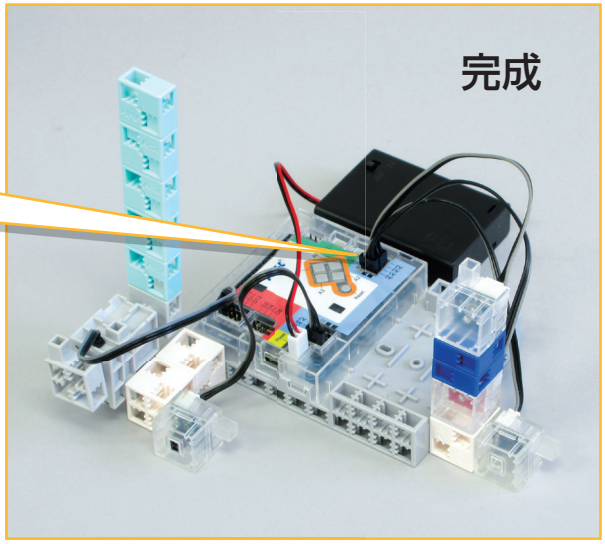


⚠ コネクターの向きに注意

LED 赤を [A0]・LED 青を [A1]・ブザーを [A2] に
接続する。



⚠ 灰色のコードの向きに注意



付録

＜掲載内容＞

- その他の操作方法
- ブロック一覧表

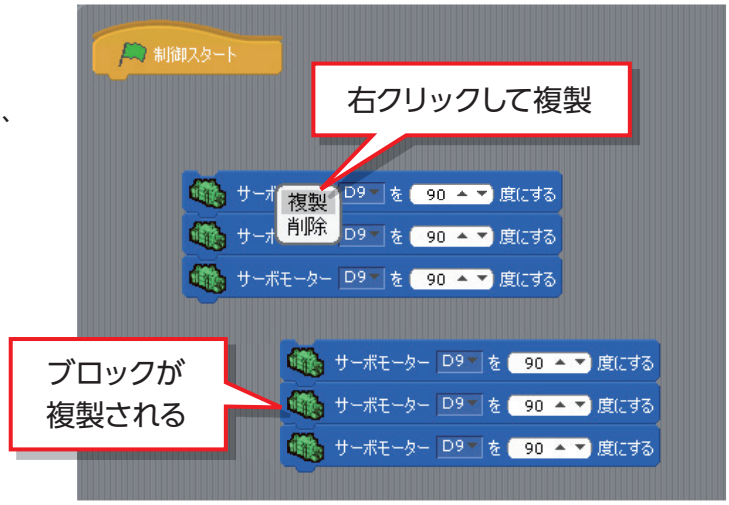
付録１　その他の操作方法

◆ ブロックの複製

つながっているブロックを複製できます。

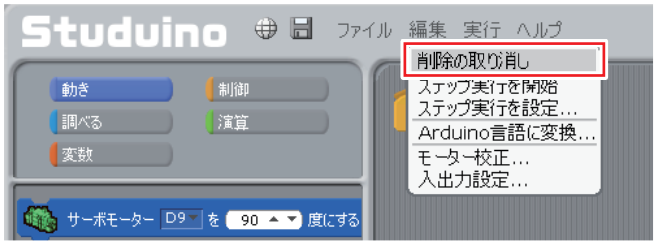
つなげたブロックの一部分だけ複製したい場合は、

その部分を抜き出して同じ操作を行います。



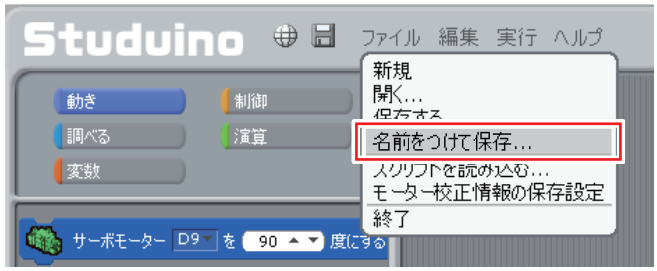
◆ ブロックの削除の取り消し

削除したブロックを戻したい場合は、「編集」から「削除の取り消し」を選択します。

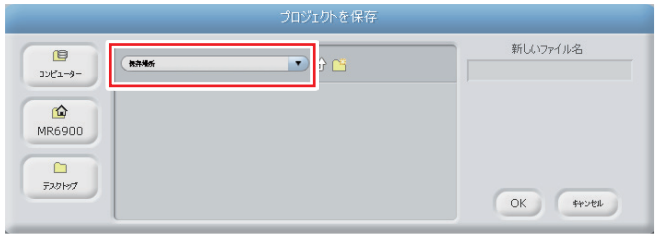


◆ プログラムの保存方法

①「ファイル」から「名前を付けて保存」をクリックします。



②保存する場所を選択します。



③ファイルに名前を付けて「OK」をクリックします。

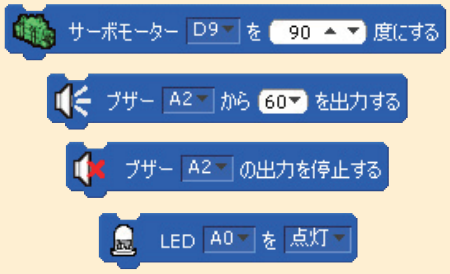
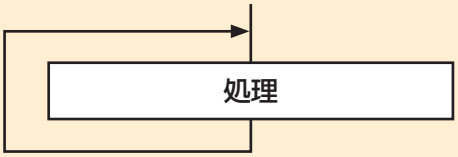
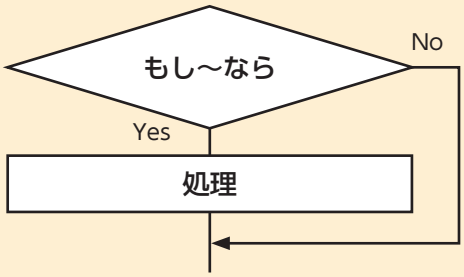
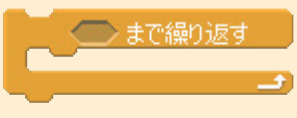
付録 2　ブロック一覧表

テキストで使うブロックに関する使い方の一覧表です。

カテゴリ	ブロック	使い方
動き		サーボモーターを指定した角度にする
		指定した高さの音を鳴らす
		音を止める
		指定した LED を点灯・消灯する
制御		指定した秒数待つ
		囲われたプログラムをずっと実行する
		囲われたプログラムを指定回数実行する
		囲われたプログラムを 条件が満たされたとき実行する
		条件が満たされるまで待つ
		条件が満たされるまで 囲われたプログラムをずっと実行する
調べる		タッチセンサーの値を調べる
		赤外線フォトリフレクタの値を調べる
演算		「小なり」で値を比較する
		「等しい」で値を比較する
		「大なり」で値を比較する

付録 3　ブロックとフローチャートの対比表

テキストで使うブロックとフローチャートの対比表です。

処理	ブロック	フローチャート
動作の処理		
待つ		
ずっと 繰り返す		
指定回数 繰り返す		
もし～なら		
～まで待つ		
～まで 繰り返す		

確認問題集

① コンピューターの動作手順を記述したものを何というでしょうか。

② プログラムを作成するときに使う特別な言葉を何というでしょうか。

③ 光、音、温度などの周囲の情報を計測する装置を何というでしょうか。

④ モーターや LED などのコンピューターからの命令に従って仕事をするものを何というでしょうか。

⑤ 社会の中で使われている製品を1つ挙げ、その中で使われているセンサーとその役割について説明しなさい。

①		②
③		④
⑤	製品	
	センサー	
	役割	

① 命令を順番通りに行う処理のことを何というでしょうか。

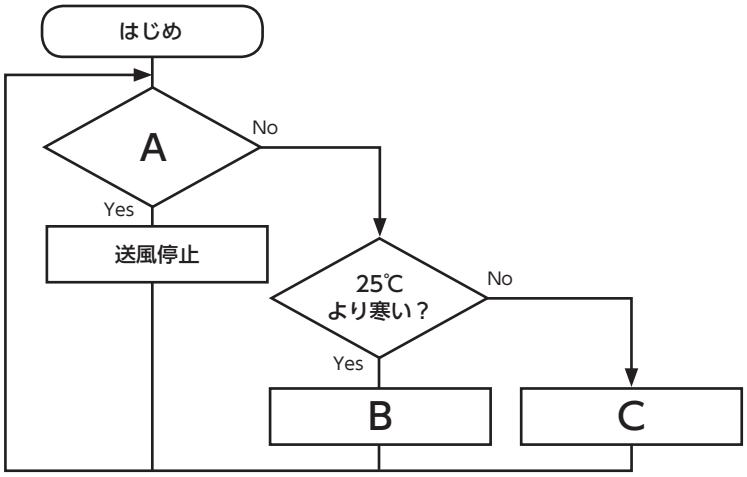
② 矢印や記号を用いて処理の手順を整理した図のことを何というでしょうか。

③ 次のA～Eの処理を表すフローチャートの記号の形をそれぞれ描きなさい。

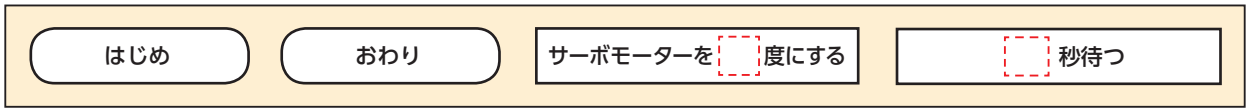
A：処理の開始と終了　　B：一般的な処理　　C：繰り返しの開始

D：繰り返しの終了　　E：条件分岐

④ 次のA～Cの空欄を埋めて室温を25℃に保つエアコンの動作を表すフローチャートを完成させなさい。



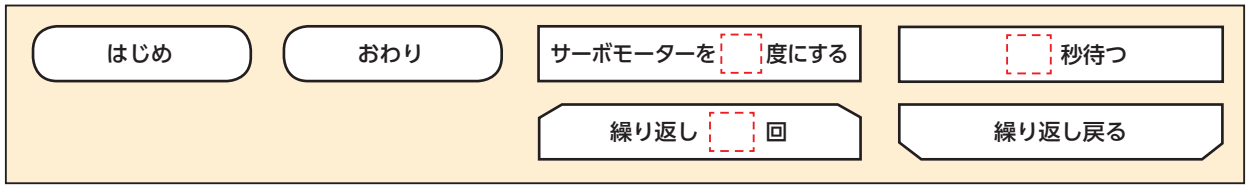
⑤ サーボモーターを1秒おきに45度→90度→135度の順番で動かすときの処理の手順を以下の記号を用いて、フローチャートで表しなさい。



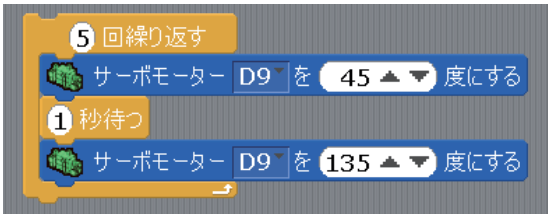
①		
②		
③	A	B
	C	D
	E	
④	A	
	B	
	C	
⑤		

① 同じ処理を回数や条件を満たすまで繰り返すことを何というでしょうか。

② サーボモーターを 1 秒おきに 45 度と 135 度に交互に 5 回動かすときの処理の手順を以下の記号を用いて、フローチャートで表しなさい。



③ サーボモーターを1秒おきに 45 度と 135 度に交互に 5 回動かしたいと考え、右のようなプログラムをつくりました。しかし、プログラムを実行すると考えた通りに動きません。その理由を正しく説明したものを選びなさい。



A：「45 度に動かす」ブロックと「135 度に動かす」ブロックの順番が逆だから

B：「ずっと」のブロックがないから

C：「135 度に動かす」ブロックの後に「1 秒待つ」ブロックがないから

②

①

③

① 条件によって動作を選択する処理のことを何というでしょうか。

② プログラムの誤りを発見し、正しく動かすために修正することを何というでしょうか。

③ ボタンが押されたときのみ５秒間開く押しボタン式ゲートの動作の手順をフローチャートにまとめます。以下の情報を参考に、フローチャートに適当な値を書き込みなさい。

	押されているとき	押されていないとき
タッチセンサーの値	0	1
	ゲートが閉じているとき	ゲートが開いているとき
サーボモーターの角度	0	90

④ ③のフローチャートをもとにプログラムを右のようにつくりました。しかし、プログラムを実行すると考えた通りに動きません。その理由を正しく説明したものを選びなさい。

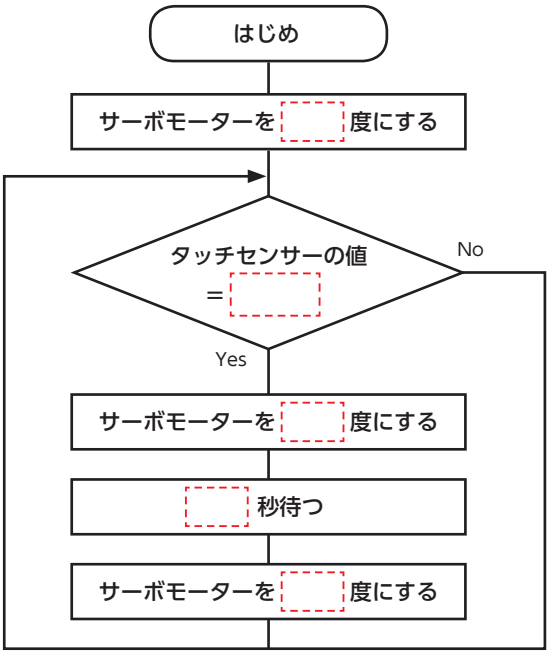
A：タッチセンサーの条件が間違っているから

B：ずっと繰り返しタッチセンサーの値を確認する処理になっていないから

C:「もし～なら」のブロックの中の「90 度に動かす」ブロックと「0 度に動かす」ブロックの順番が逆だから

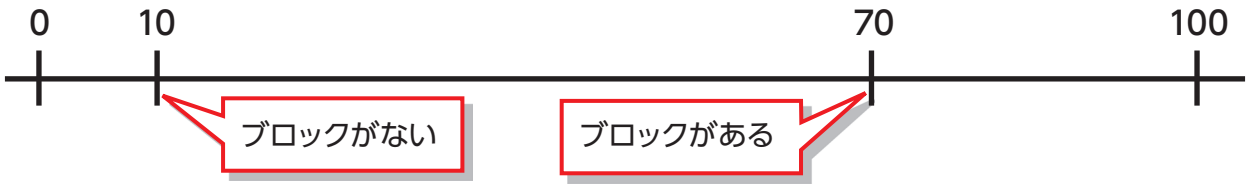
①
②
④

③



① 窓際の席で赤外線フォトリフレクタを使うことになりました。このとき、注意しなければならないことを説明しなさい。

② ブロックがあるかどうかを赤外線フォトリフレクタで判断します。ブロックがある場合の値が７０で、ない場合の値が１０のとき、しきい値はどのような値にすればいいでしょうか。適当な値を答え、また、その値を選んだ理由も述べなさい。



③ 人を感知したときのみ５秒間開くゲートの動作の手順をフローチャートにまとめます。以下の情報を参考にして、フローチャートに適当な不等号や値を書き込みなさい。

	人がいるとき	人がいないとき
赤外線フォトリフレクタの値	70	10
	ゲートが閉じているとき	ゲートが開いているとき
サーボモーターの角度	0	90

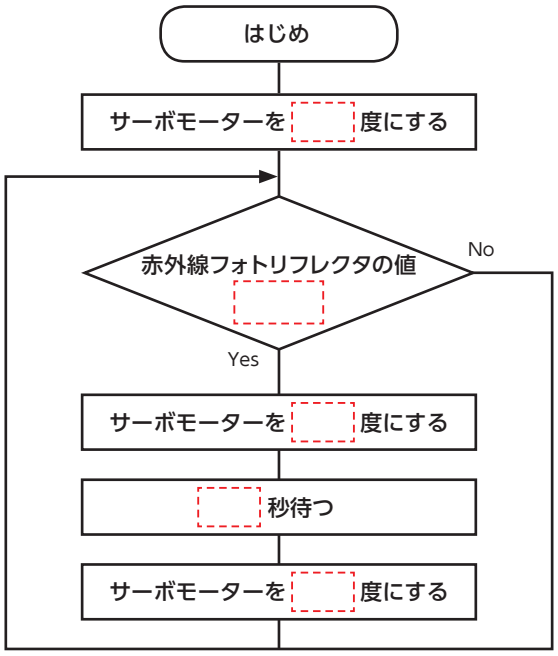
①

②

値

理由

③



解答　第 1 章　確認問題

① プログラム		② プログラミング言語
③ センサー		④ アクチュエータ
⑤（例）	製品	自動ドア
	センサー	赤外線フォトリフレクタ（人感センサー）
	役割	人がいるかどうかを判断する

解説

① プログラム・フローチャート・アルゴリズムの違いは以下のように説明できます。

手順を記述したもの	手順を図で表したもの	手順の考え方
プログラム	フローチャート	アルゴリズム

② 「プログラム言語」でも正解です。

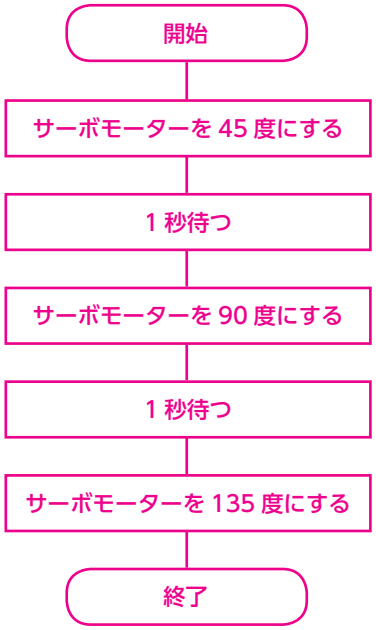
④ エネルギーを動きに変えるものだけをアクチュエータと言う場合もあります。

⑤ その他別解が多数あります。

解答　第 2 章　確認問題

①	順次処理		
②	フローチャート		
③	A 	B 	
	C 	D 	
	E 		
④	A	25℃になっているか？	
	B	温風を送る	
	C	冷風を送る	

⑤



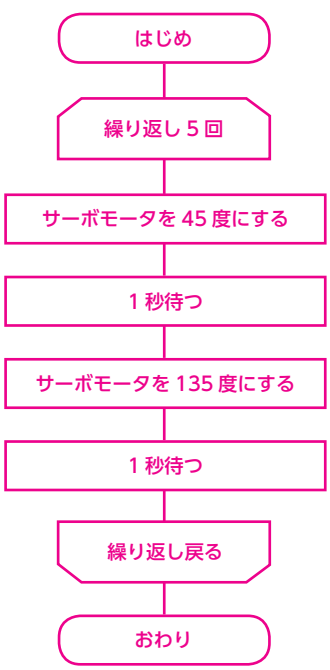
解説

③ Cで「繰り返し〇回」、Dで「繰り返し終了」と書いてあっても正解です。

解答　第３章　確認問題

①	反復
③	C

②



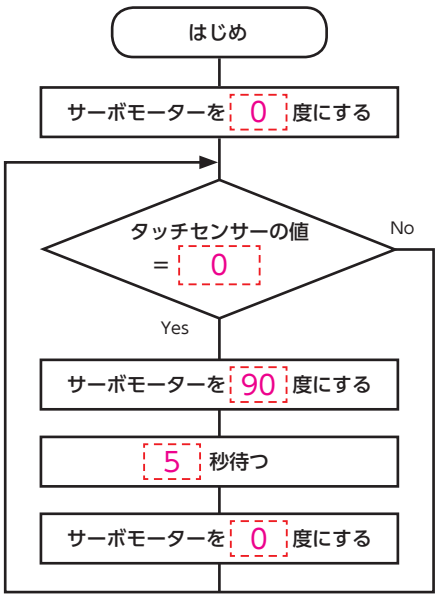
解説

② 繰り返しの記号を使わずに５回処理を書く形でもフローチャートとしては正解です。

解答　第４章　確認問題

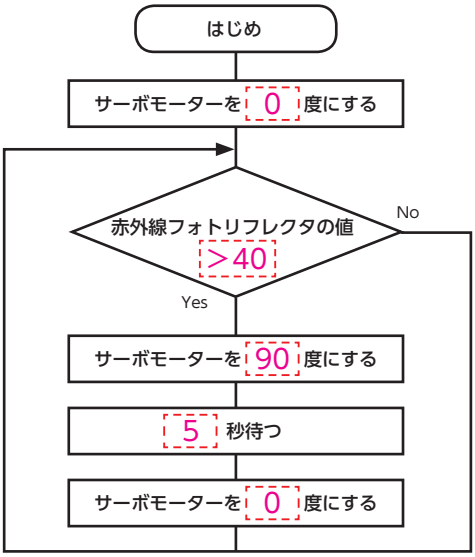
①	条件分岐
②	デバッグ
④	B

③



解説

④ プログラムは上から順番に非常に高速で処理されるため、繰り返しプログラムを動作させてタッチセンサーの値を確認させる処理にしなければ、一瞬でプログラムが終了してしまい考えた動作が実現できません。

①	赤外線フォトリフレクタを太陽光に向かって配置しないようにすること	
②	値	40
	理由	周りの環境の影響で誤作動を起こしにくくするため
③		

解説

① 赤外線フォトリフレクタの検出部を太陽光に向けない旨が書かれていれば正解です。

② 解答では、しきい値をブロックがあるときとないときの値の中間を取って 40 としています。誤作動を避けることが目的ですので、30 ～ 50 の値であれば正解として問題ありません。理由は誤作動を避けること・正確に判断することを目的にしている旨が書かれていれば正解です。

③ しきい値は②の問題と同様に考えます。ゲートが開くときは、人を感知したときであるので、「赤外線フォトリフレクタの値 ＞ しきい値」の条件を使います。

ArTeC Robo
アーテックロボ

計測と制御キット C

プログラムによる自動ゲートの制御

教員用

テキストに関するお問い合わせ

TEL：072-990-5514

E-mail：support@artec-kk.co.jp

085541 K0418