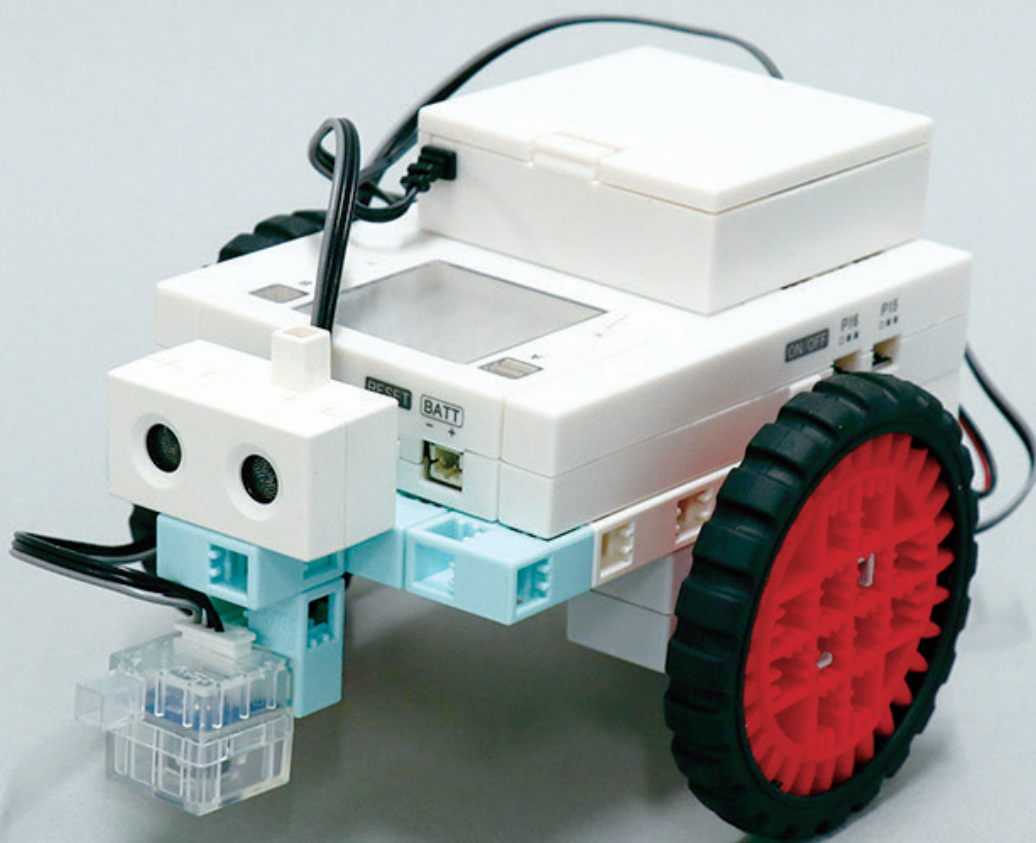


情報の技術 スタンダードパッケージ

計測と制御編

教員用



0. 問題を見いだし課題を設定する（計測・制御による問題解決）

中学校学習指導要領（平成29年告示）解説 技術・家庭編(p.53)より

- 情報の技術の見方・考え方を働かせて、問題を見い出して課題を設定し解決する力を育成
- 安全・適切なプログラムの制作、動作の確認及びデバッグ等ができるようにする
- こうした活動を通して、自分なりの新しい考え方や捉え方によって、解決策を構想しようとする態度の育成
- 自らの問題解決とその過程を振り返り、よりよいものとなるよう改善・修正しようとする態度の育成

（中学校学習指導要領（平成29年告示）解説 技術・家庭編：p.19）

生活や社会における事象を、技術との関わり視点で捉え、社会からの要求、安全性、環境負荷や経済性などに着目して技術を最適化すること。

（中学校学習指導要領（平成29年告示）解説 技術・家庭編：p.48）

生活や社会における事象を、情報の技術との関わり視点で捉え、社会からの要求、使用時の安全性、システム、経済性、情報の倫理やセキュリティなどに着目し、情報の表現、記録、計算、通信の特性等にも配慮し、情報のデジタル化や処理の自動化、システム化による処理の方法等を最適化すること。

「(1) 生活や社会を支える情報の技術」で気付いた技術の見方・考え方から、問題を見い出して、課題を設定する流れが大切になります。例えば、生活をサポートするロボットで考えてみましょう。

解決したい問題には、「大きい問題」と「小さい問題」があります。大きい問題は、根本的に計測・制御システムとして解決したい問題です。掃除の労力負担、企業などでの清掃要員の経費増大等といった問題です。小さい問題は、システムを構成する要素や機能を実現していく過程で、新たに発見される問題です。そのため「大きな問題」の解決のために「概念設計」という設計を行います。小学校でのプログラミング的思考を活かして、意図することを実現するためには、どのような機能が必要か、それらをどのように接続・連携させればよいのか、という全体構成を設計します。次に、それらの要素や機能を具体的にどのようにプログラミングするのかという設計を行います。これを「詳細設計」と言います。しかし、実際にはScratchのようなブロック言語は、それ自体が設計を図示しているくらいのわかりやすさがあるので、アクティビティ図などのUMLで流れを簡潔に整理して、すぐにプログラミングさせることも考えられます。

- ①前進するロボットを動かしてみても不便に感じた点（問題）から改善点（課題）を話し合わせましょう。生徒たちが最初に気づく問題は、「思いついた」ということが多いことでしょう。生徒のそうした「単なる思いつき」の発言であっても、それが技術の見方・考え方に照らして価値付けすることで、生徒たちは、「そうか、そのように考えればよいのか」という考え方を学びます。「障害物にぶつかってしまう」という問題は、安全性に着目したから気づいたと言えます。同じ結果を得られるのであれば、たくさんセンサーを付けるよりも1つのセンサーで実現した方が経済性が高いといえます。プログラムも無駄が多くて長いものでは、見にくくわかりにくいと保守・メンテナンス・改良もしにくいでしょう。多少極端ですが、無駄が多いことでディスクの使用領域をたくさん使ったり、通信するときのデータ量の増大につながれば経済性も悪くなるでしょう。
- ②その上で、この段階では、社会からの要求としてどのように動くことがまずは基本機能として必要なのかを考えさせます。次に、解決したい問題に対して、問題が生じる場面において、どのように状況が変化するのか、それをどのようにセンシング（計測）すればよいのか考えさせます。例えば、「手をかざしたらLEDをつけたい」という要求に対して、「手をかざしセンサー」というものは存在しません。手をかざすという状況によって、「影ができる」という現象を捉えるために「光センサー」が使えるのではないかと、対象物との距離を計測できるセンサー（超音波センサー、赤外線フォトリフレクタ）を使えば、手をかざすことで手までの距離が変わることを利用できるのではないかとという原理をもとにした計測への理解を深められます。
- ③そして、それによってどのように出力をコントロール（制御）すればよいのかを考えさせます。そのためには、どのような処理が必要になるのかをイメージさせます。例えば、明るさセンサーを使って、常に明るさを調べておいて、「暗くなったら（明るさセンサーの値が、ある値よりも小さくなったら）、モーターを止めよう」のように、入力→処理→出力、という中での信号の流れがわかるようにしておくことが大切です。技術の見方・考え方で、求められる機能を課題として挙げさせましょう。
- ④生徒が設定した課題は、実際に生活や社会で実際に利用されている方法で解決することが必ずしも重要なのではなく、目的に適したプログラミング言語を用いて、生徒自身がアイデアを出し、工夫し創造しながら問題を解決するプロセスを大切にしましょう。
- ⑤その際、そもそもの目的を達成するために、複数のセンサーやアクチュエータ（LEDも含む）をどのように構成するのか、「システム」として全体を捉えさせるようにしましょう。
- ⑥本指導書では、情報の技術の見方・考え方を意識しながら、アクティビティ図や実際のプログラム例を示し、各機能をプログラミングする際のコツや、改良のヒントを紹介しています。参考例として、開隆堂の教科書で紹介されている実習例をもとに、**次ページ目次**のサンプルを扱っています。

目次

導 入	身近なプログラミング	6ページ
-----	------------	------

基本学習	センサー・アクチュエータを活用したプログラムによる計測・制御の学習を行います。
------	---

テーマ1	LEDの制御 / ブザーの制御	13ページ
------	-----------------	-------

メインユニットのフルカラー LED やブザーを活用して、信号機の制御やセンサーライトを再現しながらプログラムによる制御の基本を学習します。

応用学習	指導要領に則った課題解決実習を行います。
------	----------------------

テーマ1	机の上の小さなゴミがある、技術室の床に金属の屑がある	31ページ
------	----------------------------	-------

0. 組み立て	32ページ
---------	-------

● 技術の見方・考え方:社会からの要求

1. 直進だけでは回収範囲が狭い → 方向転換する機能	35ページ
-----------------------------	-------

● 技術の見方・考え方:安全性の観点、システム

2. 物に触れても動き続ける → 接触検知機能	37ページ
-------------------------	-------

3. 机の端から落ちてしまう → 落下防止機能	39ページ
-------------------------	-------

4. 前方に壁があっても突っ込んでしまう → 壁の検知機能	41ページ
-------------------------------	-------

テーマ2	就寝時などに、身の回りから離れた場所にある貴重品が盗まれたりしないかが心配	43ページ
------	---------------------------------------	-------

0. 組み立て	44ページ
---------	-------

● 技術の見方・考え方:社会からの要求

1. 人が常時監視することができない → 状況の変化を検知する機能	46ページ
-----------------------------------	-------

● 技術の見方・考え方:安全性の観点、システム

2. より確実に守りたい → 複数のセンサーを関連させて検知	48ページ
--------------------------------	-------

サーボモーターを使用した作例の組み立て	50ページ
---------------------	-------

→ 離れた場所にも通知	52ページ
-------------	-------

→ 動きで通知	54ページ
---------	-------

テーマ3	運動不足解消のためのエクササイズを楽しく継続したい	56ページ
------	---------------------------	-------

0. 組み立て	57ページ
---------	-------

● 技術の見方・考え方:社会からの要求

1. エクササイズを楽しみたい → ゲーミフィケーションの要素	59ページ
---------------------------------	-------

● 技術の見方・考え方:システム

2. 色々なエクササイズをしたい → システムとして構成する	61ページ
--------------------------------	-------

注意事項

端末と接続状態でのご使用时

Studuino:bit にプログラムを転送させずに、端末と Studuino:bit を USB もしくは Bluetooth で「接続」した状態で端末からプログラムを実行する場合、プログラムを Studuino:bit に転送して実行した場合にくらべて、プログラムの処理に遅延が生じる場合があります。この遅延はアプリ上でのプログラムの処理速度および端末と Studuino:bit 間の通信により発生します。センサーの値に応じて自動車を制御するといった、処理の遅延が動作に影響するようなプログラムを作成する場合は、プログラムを転送させて動作確認を行ってください。

乾電池のご使用时

乾電池が消耗してくると、基板にリセットがかかりやすくなります。とくに DC モーターを回転させる瞬間に、最も電圧降下が激しくなり、基板にリセットがかかって意図しない動作をしたり、Bluetooth の接続が切れたりすることがあります。また、電源を入れたまま保管してしまうと、待機電力により意図せず電池の消耗が起こることがあります。保管時は電池ボックスのコネクタを抜いておくようにしてください。

iPad/Android/Chromebook用Bluetooth通信対応のアプリご使用时

Bluetooth 通信対応アプリにおいては、USB での端末と Studuino:bit の通信に対応していないため、以下の点に注意いただく必要がございます。あらかじめご理解ください。

①ファームウェア更新について

ファームウェア更新時、端末と Studuino:bit が Wi-Fi 接続されます。
Studuino:bit は Wi-Fi による更新時に下記の状態で作動します。

Studuino:bit 内アクセスポイント SSID: Stu:bit0192837465
Studuino:bit 内 WEB サーバー IP: 192.168.4.1

端末の管理システム等により Wi-Fi の接続先やサーバーのアクセス先に制限がかかっている場合は、ファームウェア更新が正常に行えません。

アクセスを許可していただくか、Windows 用ソフトウェアを用いて USB ケーブル経由でのファームウェア更新を実施してください。

②Bluetooth接続時の機能制限について

右記の無線ブロックについてはプログラムを Studuino:bit へ転送することで機能しますが、端末と Studuino:bit が Bluetooth 接続されている状態では機能しません。



ArtecRobo2.0サポートガイドはこちら

https://www.artec-kk.co.jp/artecrobo2/pdf/jp/artecrobo2_supportguide.pdf

作成上の注意

ロボットモードとキャラクターモードでは、プログラムの挙動や使用できる機能が異なります。

ロボットモード	キャラクターモード
使用できるスプライトはArtecRobo2.0のみ	色々使用できる
Studuino:Bit本体にプログラムを転送（書き込み）して、電池ボックスをつなげれば単体で動く	パソコンとの常時USB有線接続 iPadの場合は常時Bluetoothでの接続
変数、リスト、ブロック定義に日本語は使用できない	制約無し
Pythonに変換したプログラムを見られる	見られない
イベントで「メッセージを送る」が使えない	使える
通信させたとき、負の数は送信できない	負の数も送信できる

導入

身近なプログラミング

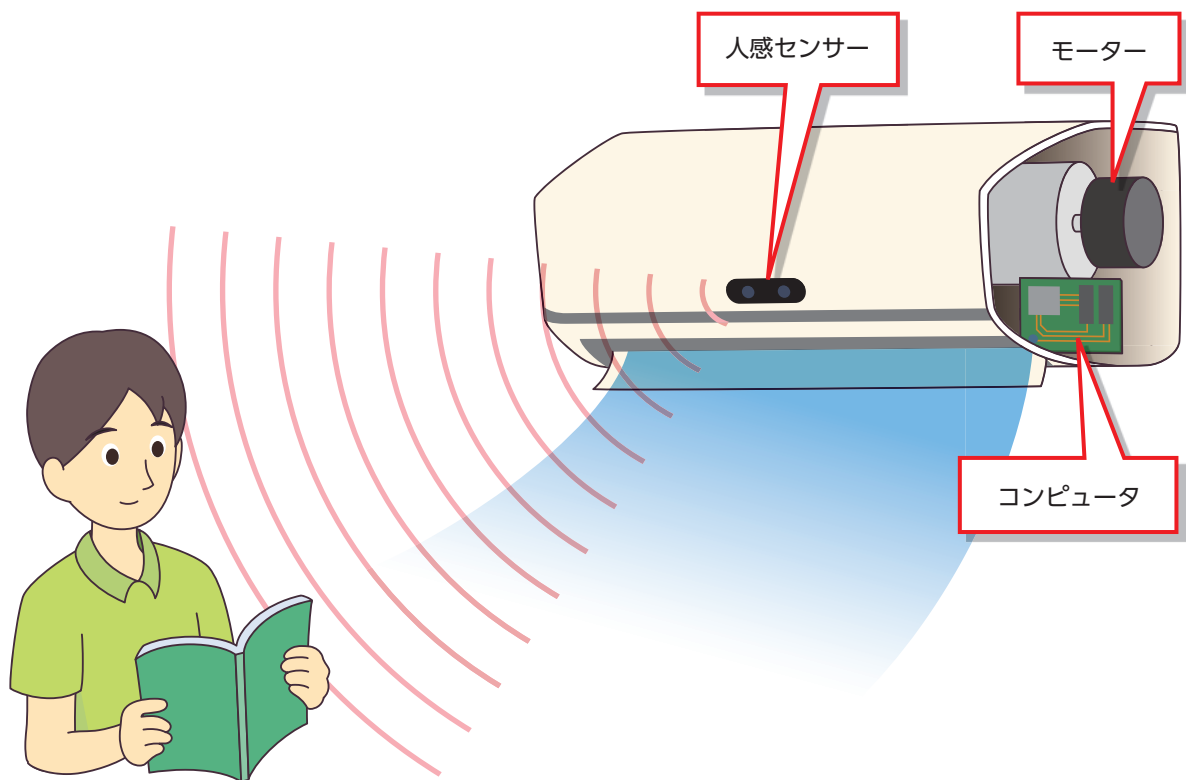
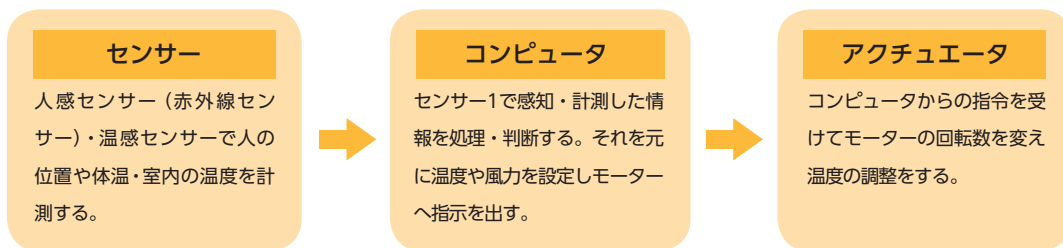
<学習内容>

- 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は電気製品が、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／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



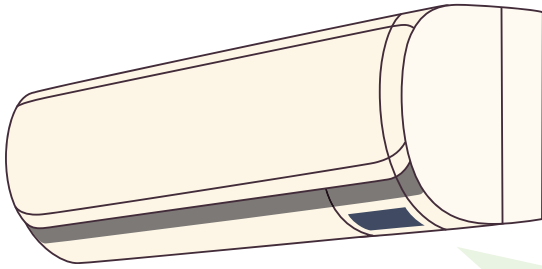
身近な製品でどういうものがコンピュータやセンサーを活用しているか探してみましょう。

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

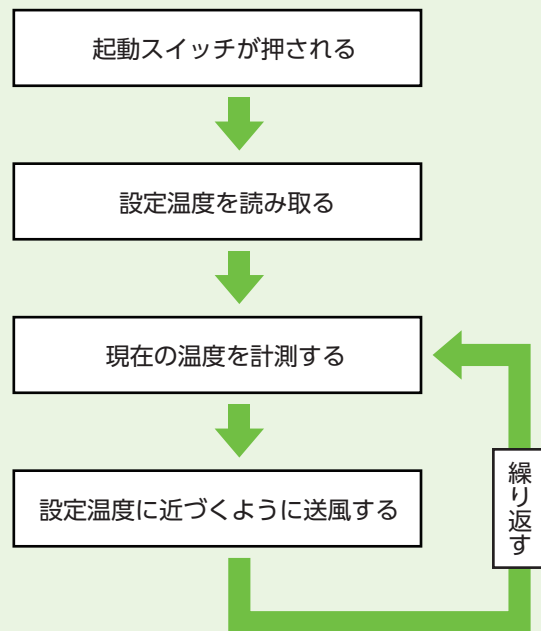
2. プログラミングとは

エアコンの例のように、コンピュータが使われている機械はセンサ得た情報を元に、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



人間が決めたエアコンの動作手順



このようにコンピュータ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。

```
from pystubit import *
import machine

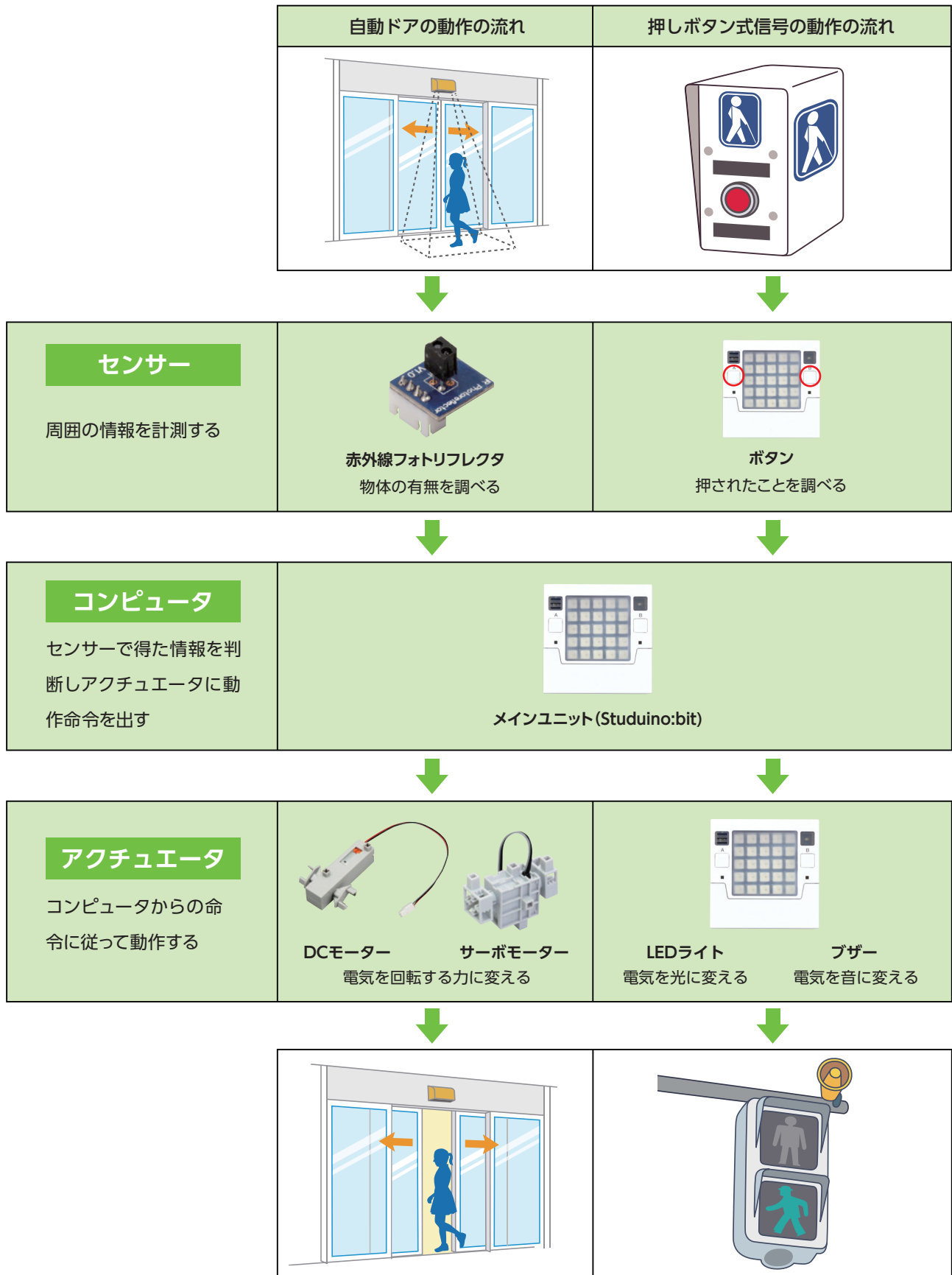
if lightsensor.get_value() > 300:
    xon = False

while True:
    if button_a.is_pressed() is True:
        bcount = bcount+1
        if bcount == 4:
            bcount = 1
    elif button_b.is_pressed() is True:
        bcount = bcount-1
        if bcount == 0:
```

▲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ArtecRobo2.0パーツの対応

ArtecRobo2.0のパーツは以下のようなコンピュータによる計測・制御システムで使われる各要素に対応しています。



※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

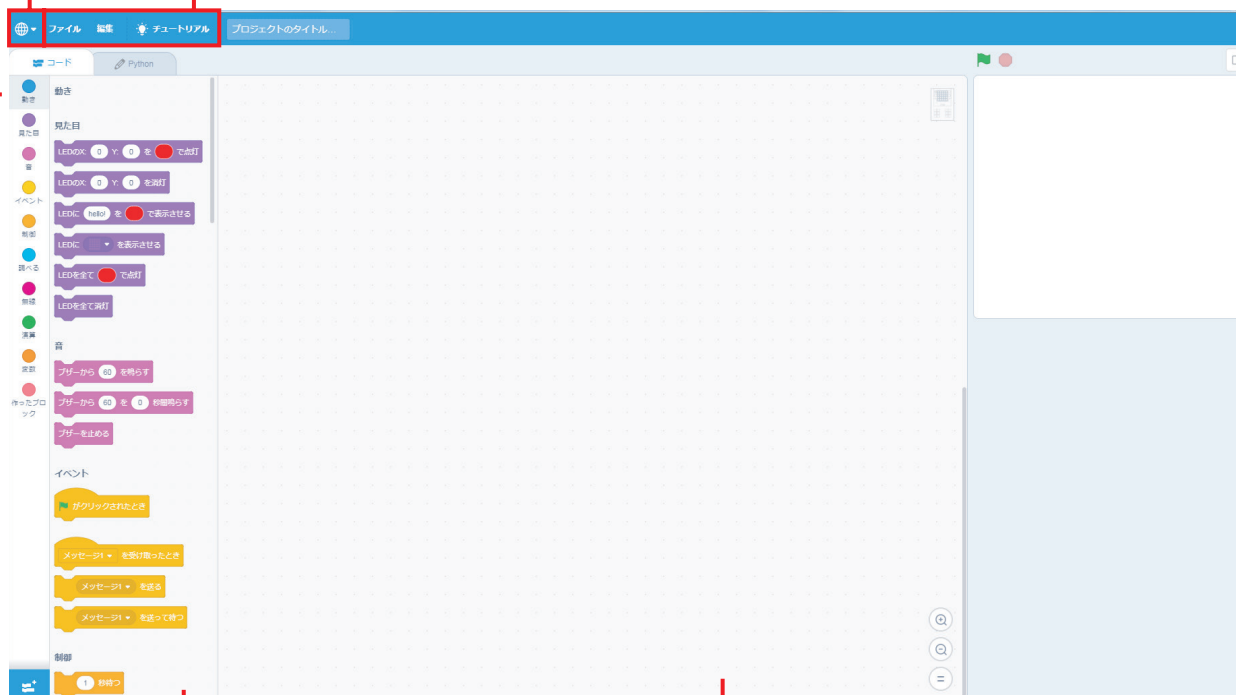
ロボットモードを選択してください。



カテゴリ：命令の種類を選ぶことができます

言語の選択

メニュー



ブロックパレット：
センサーやアクチュエータへの命令が表示されます

スクリプトエリア：
命令をつないでプログラムをつくることができます

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れます。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」「見た目」「音」「イベント」「制御」「調べる」「無線」「演算」「変数」「関数」の10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	DC モーターやサーボモーターなど、アクチュエータの動きを命令するブロックです。起動時、ブロックは表示されていません。後述の「入出力設定」で使用可能になります。
 見た目	LEDの点灯・消灯の命令をするブロックです。
 音	ブザーを鳴らす命令をするブロックです。
 イベント	プログラムスタートを行うブロックです。
 制御	命令の実行順を制御するブロックです。
 調べる	センサーを指定して周りの情報を調べるブロックです。
 無線	無線で送受信を行うブロックです。
 演算	計算の命令を出したり、条件式を作成するブロックです。
 変数	数値情報の記録をする変数やリストブロックを作成します。
 関数	関数としてブロックを作成します。

基本学習 テーマ1

<学習内容>

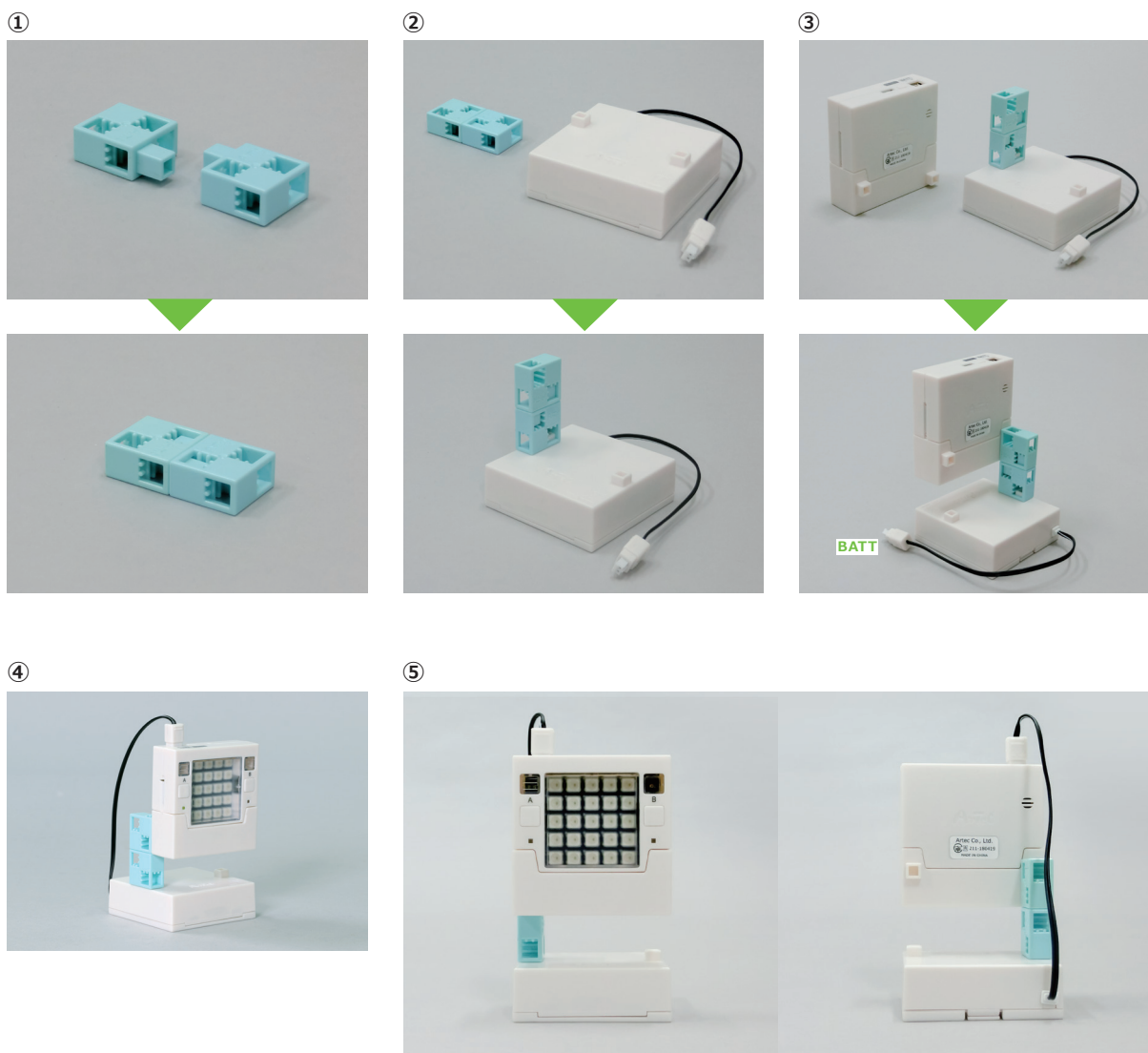
- LED の制御 / ブザーの制御
- 順次処理
- 繰り返し処理
- 条件分岐処理

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。LEDを使った信号機をつくる中で、順次処理のプログラムを学びましょう。



①「信号機」の組み立て



② コンピュータとの通信

Windows/Mac の場合

USB ケーブルによる通信

- ① コンピュータとメインユニット (Studuino:bit) を USB ケーブルで接続します。



- ② 「編集」から接続を選択します。



Android/iPad/Chromebook の場合

Bluetooth による通信

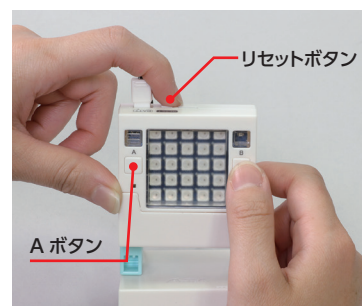
- ① 「編集」から接続を選択します。



- ② 表示されるメッセージに従い、メインユニット (Studuino:bit) の B ボタンを押しながらリセットボタンをおしてください。



- ③ メインユニット (Studuino:bit) の LED に表示された点灯パターンと同じデバイスを選択してください。



右図のようなセンサーボードが表示されると、通信が正常に行われています。

Bluetooth による通信時は通信ランプが青に点灯します。

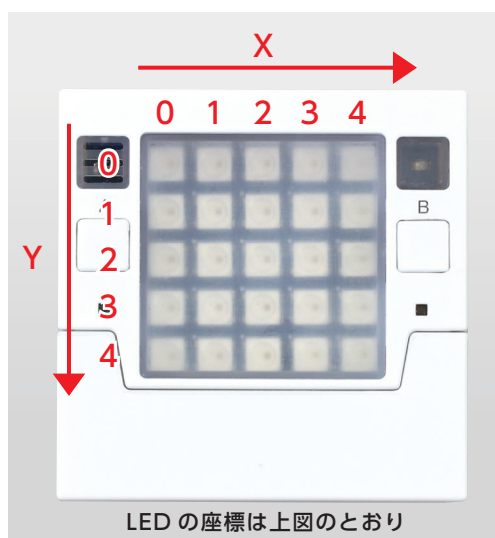


通信ランプ

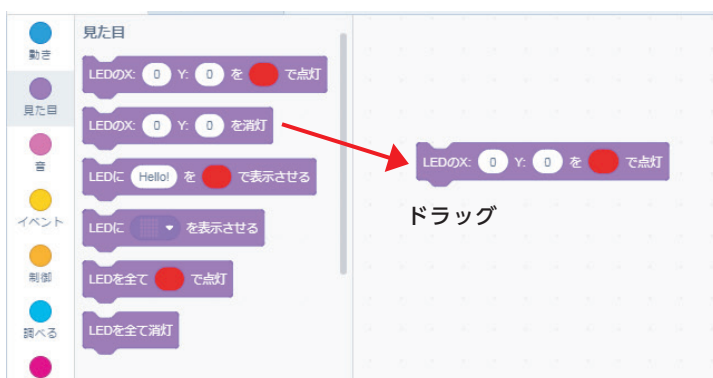
センサーボード	
Studuino:bit	
ボタンA	1
ボタンB	1
光センサー	6
温度センサー	136.7
加速度センサー X	0.05
加速度センサー Y	-0.17
加速度センサー Z	0.98
ジャイロセンサー X	0
ジャイロセンサー Y	-2
ジャイロセンサー Z	2
磁気センサー X	94
磁気センサー Y	-76
磁気センサー Z	-92

③LED を点灯させるプログラム

LED を光らせる命令を出すには次のブロックをつかいます。



ブロックをドラッグして光らせたい LED の座標と色を指定します。



LEDのX: 0 Y: 0 を 赤 で点灯 をクリックしてLED が点灯することを確認してください。

やってみよう!

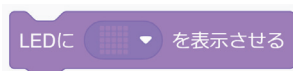
その他のブロックについてもどのようにLED が点灯するか同様に確認してください。



…………指定した座標の LED を消灯します。



…………指定した文字をスクロール表示します。(半角英数字)



…………全ての LED の点灯色を個別に指定します。



…………全ての LED の点灯色を同時に指定します。

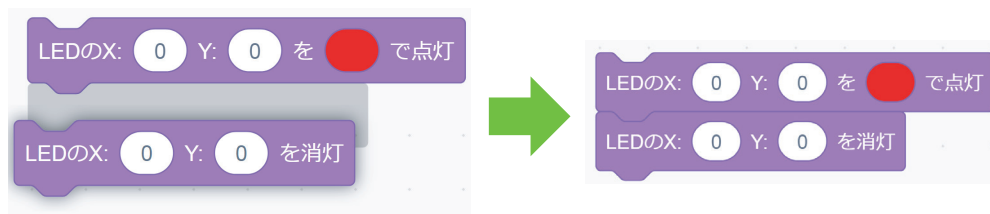


…………全ての LED を消灯します。

④LED を 1 秒間だけ点灯させるプログラム

LEDのX: 0 Y: 0 を ● で点灯 のブロックと LEDのX: 0 Y: 0 を消灯 を順番につなげましょう。
このようにブロックをつなげると、上から順番に命令が送られて、プログラムが実行されます。

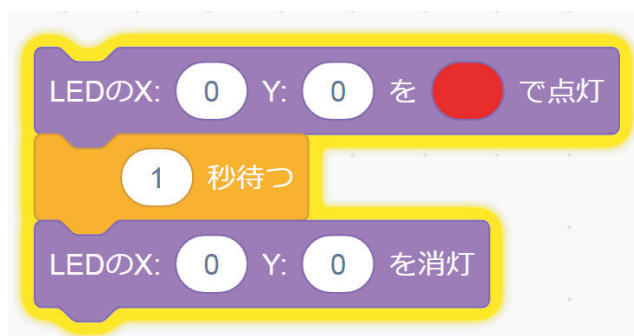
ドラッグしたブロックを影ができたところではなすと、ブロックどうしをつなぐことができます。



1 秒待つ ブロックを LEDのX: 0 Y: 0 を ● で点灯 と LEDのX: 0 Y: 0 を消灯 の間につなぎましょう。

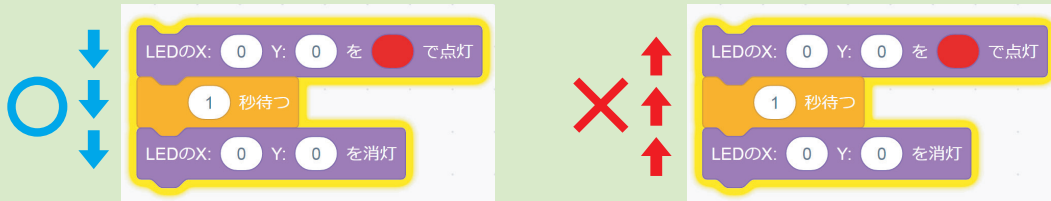


並べたブロックをクリックするとプログラムが転送されて実行されます。
LED が 1 秒間光って消えることを確認しましょう。

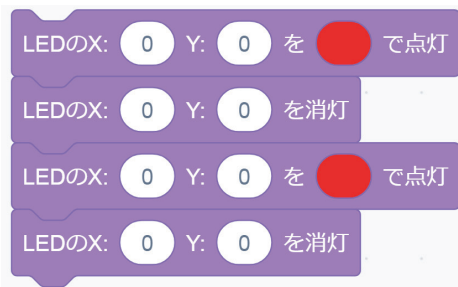


1 秒待つ の数値を変えることで、LED の点灯時間を設定することができます。
また、時間は少数点も設定できるので、自由に変えて動かしてみましょう。
※コンピュータ内の通信処理速度により設定した時間より少し長い時間点灯する場合があります。

プログラムは必ず上から順番に実行されます。



「1秒待つ」のブロックを入れないとどうなるのか？



左のプログラムで動作をさせた場合、LED はほとんど点灯しません。これはコンピュータ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。つまり、「LED 点灯」の命令の直後に、「LED 消灯」の命令が行われてしまうため、LED を点灯している時間がありません。※一瞬点灯する場合がありますが、これはコンピュータとの通信の際のわずかな遅延によるものです。

⑤接続を解除する

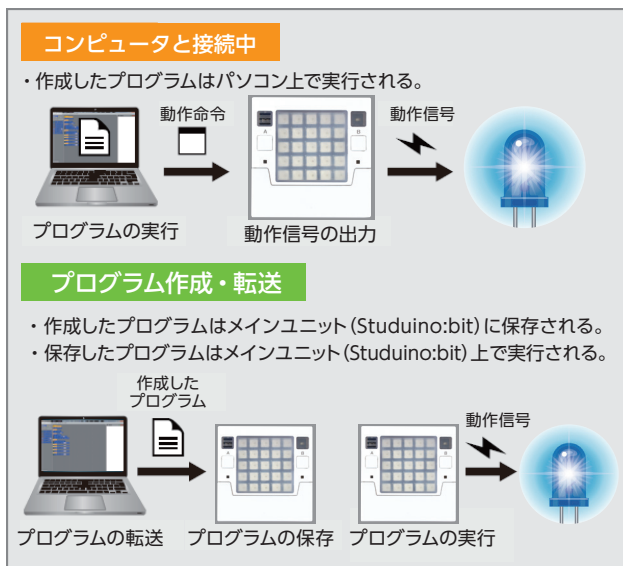
プログラムの動作が確認できたら、接続を解除しましょう。画面上方の編集より、接続解除を選びます。



⑥プログラムを転送する

ここまではコンピュータとメインユニット (Studuino:bit) が通信し、コンピュータからの命令でプログラムを動かしていました。

作成したプログラムをメインユニット (Studuino:bit) に転送することで、メインユニット (Studuino:bit) のなかにプログラムが保存され、コンピュータから切り離してもプログラムを実行することができるようになります。



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 ケーブルや Bluetooth の通信を切断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。

◆ プログラムの転送方法

①作成したプログラムの一番上に **がクリックされたとき** をつなぎます。



②編集から「転送」を選びます。



Android/iPad/Chromebook の場合

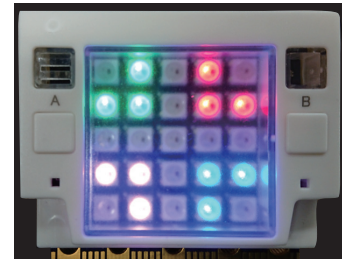
Bluetooth による通信



①表示されるメッセージに従い、メインユニット (Studuino:bit) の B ボタンを押しながらリセットボタンを押してください。



②メインユニット (Studuino:bit) の LED に表示された点灯パターンと同じデバイスを選択してください。



③プログラムを転送する先を指定してください。

※0～9まで選択できます。



④転送が完了すると、自動的にプログラムが実行されます。

再度動かしたい場合はリセットボタンを押してください。

複数のプログラムを転送した場合の実行方法

複数のプログラムを 0 ～ 9 の番号を指定して転送することができます。

通常は最後に転送したプログラムが実行されますが、以下の方法で番号を指定して転送済みのプログラムを実行することができます。



①電源を接続する(通常起動)

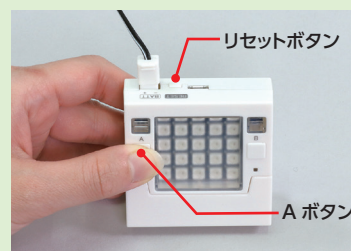
- (1) 電池ボックスもしくは USB ケーブルをメインユニットに接続します。
- (2) 電源ランプ(緑)が点灯し、電源が自動的に ON になり起動します。

通常起動時は前に実行したプログラムが自動的に実行されます。



②プログラム選択モードへの移行

- (1) A ボタンを押したまま、リセットボタンを押してください。
リセットボタンを押すと再起動のため一時的に電源ランプが消灯します。
- (2) リセットボタンを離して 3 秒後に電源ランプが再点灯したら、A ボタンを離します。



- (3) 緑の LED で 0 が表示されるとプログラム選択モードへ移行しています。



③プログラムの選択・起動

- (1) プログラム選択モード時に A ボタンを押すと、表示が 0・1・2・・・と切り替わります。9 まで切り替わると 0 に戻ります。
- (2) 選択した番号で B ボタンを押すと、各番号に割り振られたプログラムが実行されます。

次回より通常起動時に③で選択したプログラムが自動で実行されます。

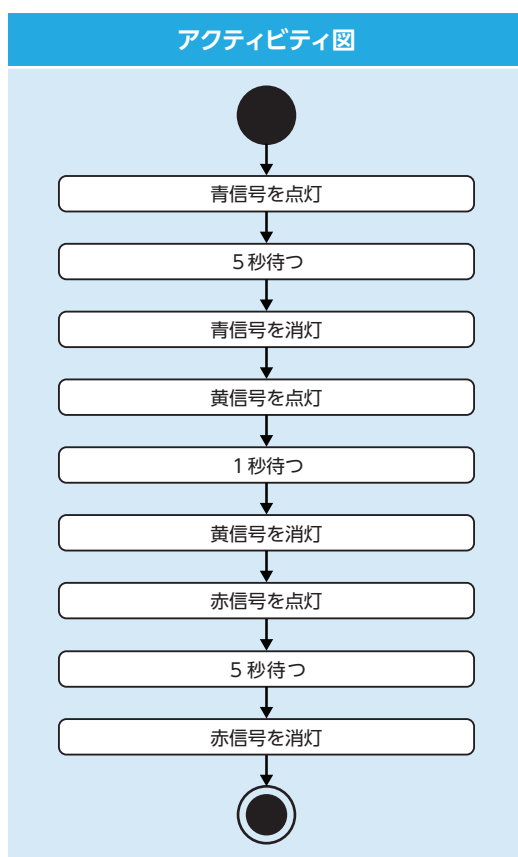


※工場出荷段階で動作確認用に各番号にはサンプルプログラムが書き込まれています。

※サンプルプログラムはソフトウェアを使用して新たなプログラムを書き込むと上書きされます。

練習課題 自動車用信号機のプログラムの作成

次の動作の手順をフローチャートにまとめたあと、プログラムをつくしましょう。



ブロックの代わりに



を使用しても同様のプログラムを作成することができます。

2. 繰り返しのプログラム

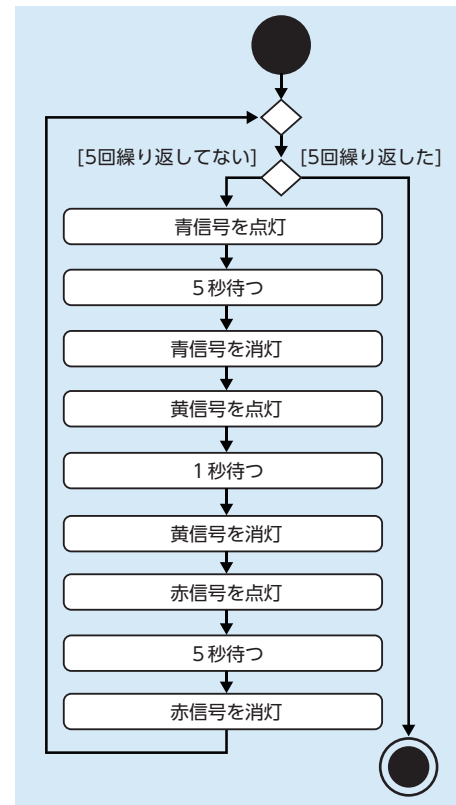
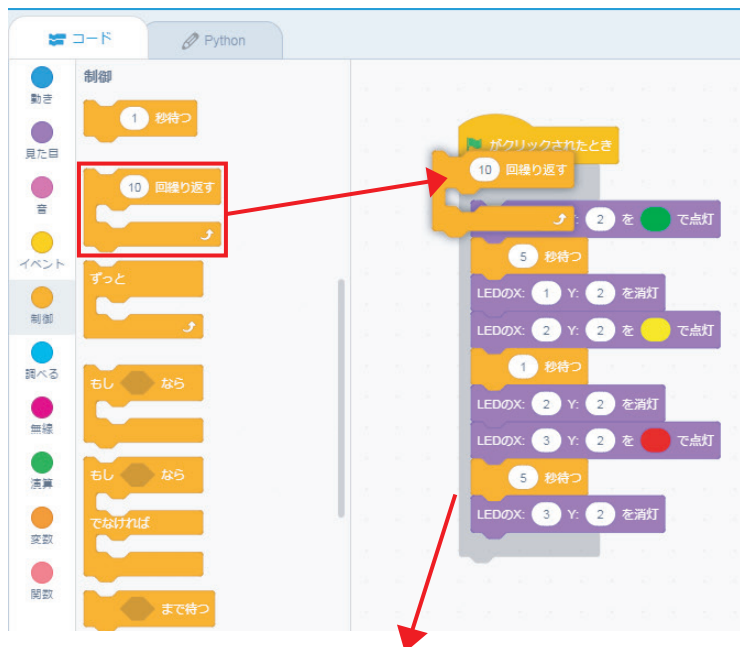
ある命令を繰り返し行う処理のことを「**繰り返し処理**」といいます。前のページで作成した信号機のプログラムは赤信号の点灯が完了すると消灯し、その後はプログラムは動きません。実際の信号機のように繰り返し動作させるには、この「**繰り返し処理**」が必要になります。

プログラムの改造（繰り返し処理の追加）

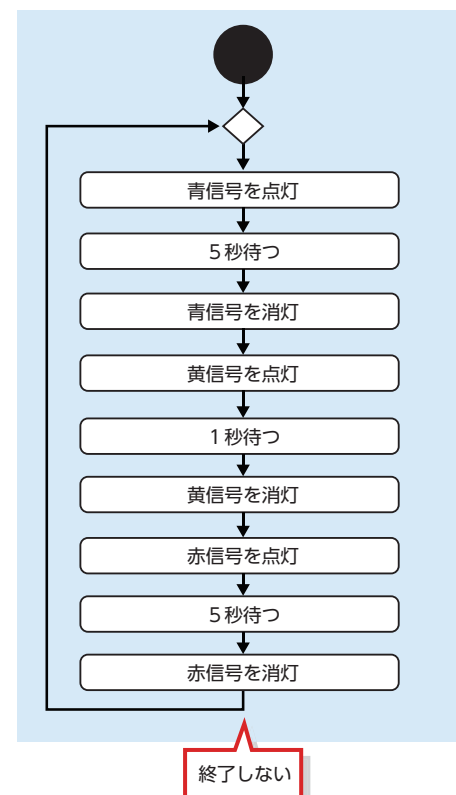
同じ動きを指定回数繰り返させたい場合は、「○回繰り返す」ブロックを、ずっと繰り返させたい場合は「ずっと」ブロックを利用します。

下図のように自動車用信号機のプログラム全体を囲むように

 または  を追加してください。

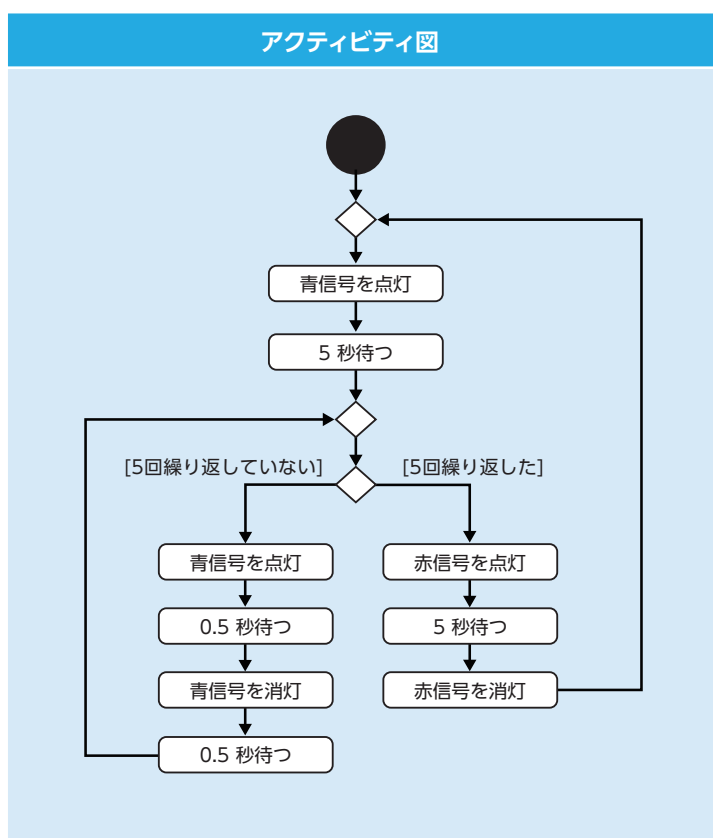
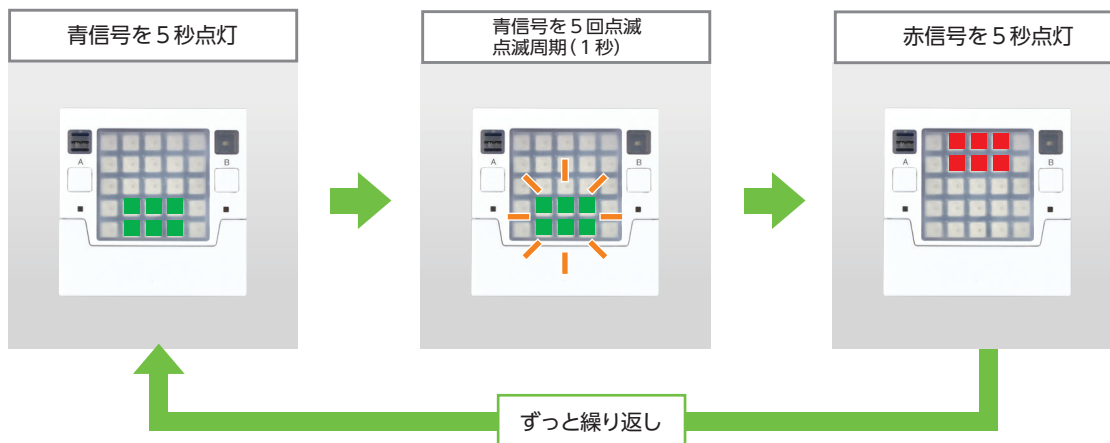


「終了」がなくなり、繰り返し続けることは下のアクティビティ図のように矢印で表すこともできます。



練習課題 歩行者用信号機のプログラムの作成

次の動作の手順をフローチャートにまとめたあと、プログラムをつくりましょう。



LEDを全て消灯 ブロックの代わりに LEDに [] を表示させる を使用しても同様のプログラムを作成することができます。

3. 条件分岐のプログラム

条件によって動作を分ける処理のことを「**条件分岐処理**」といいます。押しボタン式信号機のプログラム作成を通して条件分岐のプログラムを学びましょう。

① ボタンの数値確認

センサーボード	
Studuino:bit	
ボタンA	1
ボタンB	1
光センサー	6
温度センサー	136.7
加速度センサー X	0.05
加速度センサー Y	-0.17
加速度センサー Z	0.98
ジャイロセンサー X	0
ジャイロセンサー Y	-2
ジャイロセンサー Z	2
磁気センサー X	94
磁気センサー Y	-76
磁気センサー Z	-92

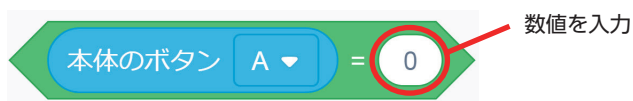
センサの情報は数値で表されます。

メインユニット (Studuino:bit) をコンピュータと接続中に表示されるセンサーボードで押しボタン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確認しましょう。





ボタンが押されている ときの数値	ボタンが押されていない ときの数値
0	1

② プログラムの作成

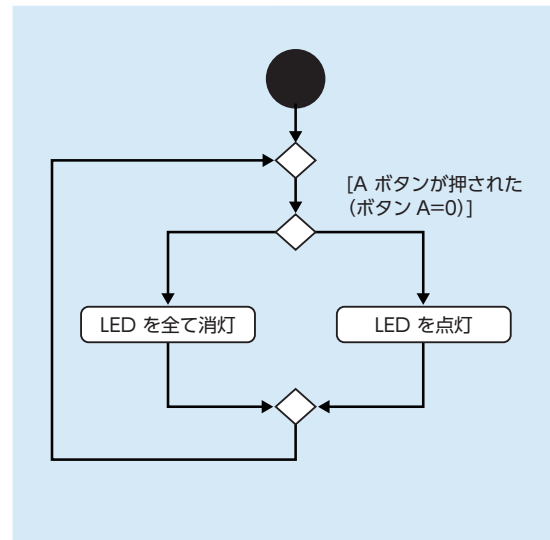
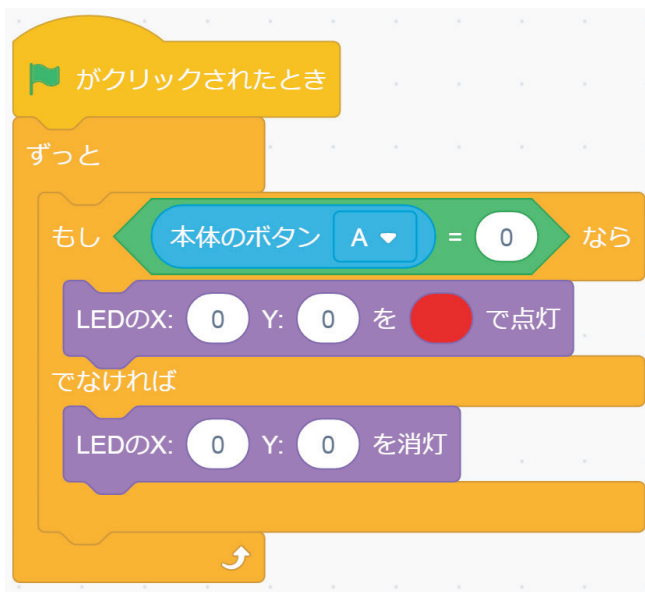
条件は  と  を組み合わせてつくることができます。



作成した条件は右図の各種制御ブロックの空欄に入れて使うことができます。

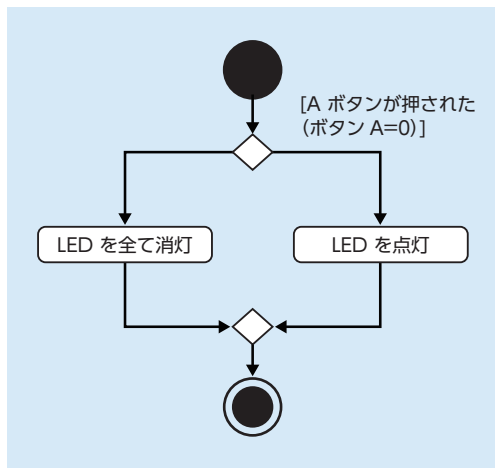
 条件が成り立つときだけ、囲まれたプログラムが処理されます。
 条件が成り立つときだけ、上段で囲まれたプログラムが処理され、 条件が成り立たないときは、下段で囲まれたプログラムが処理されます。
 条件が成り立つまで、次に続くプログラムを処理せずに待ちます。
 条件が成り立つまで、囲まれたプログラムを繰り返し処理します。

③ A ボタンを押している間 LED が点灯するプログラム



「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、A ボタン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、A ボタン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。

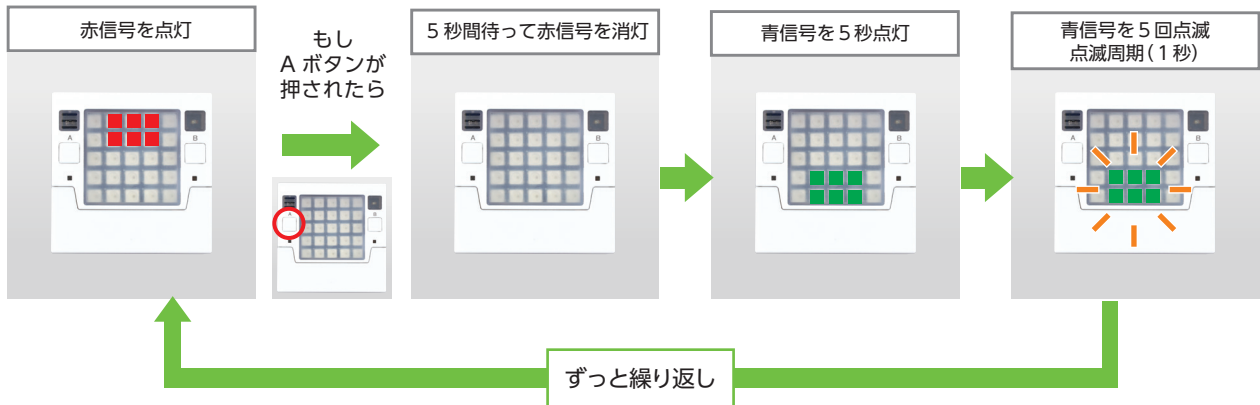


動作させた瞬間にプログラムが終了。
プログラム終了後に A ボタンを押しても反応しない。

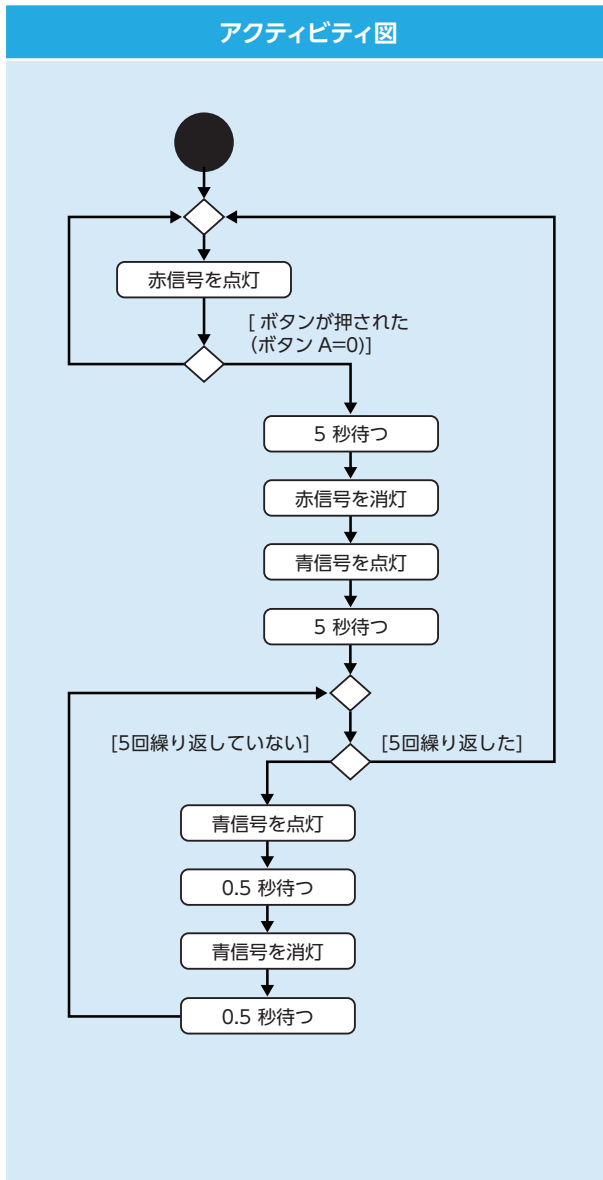
練習課題

<押しボタン式信号機のプログラム>

次の動作の手順をフローチャートにまとめたあと、プログラムをつくりましょう。



アクティビティ図



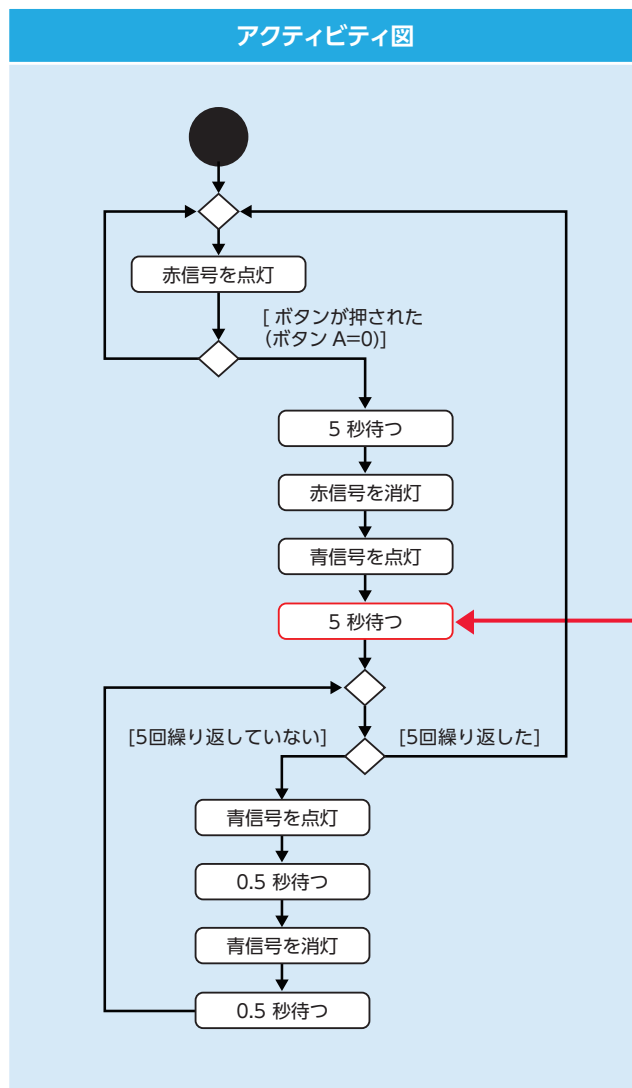
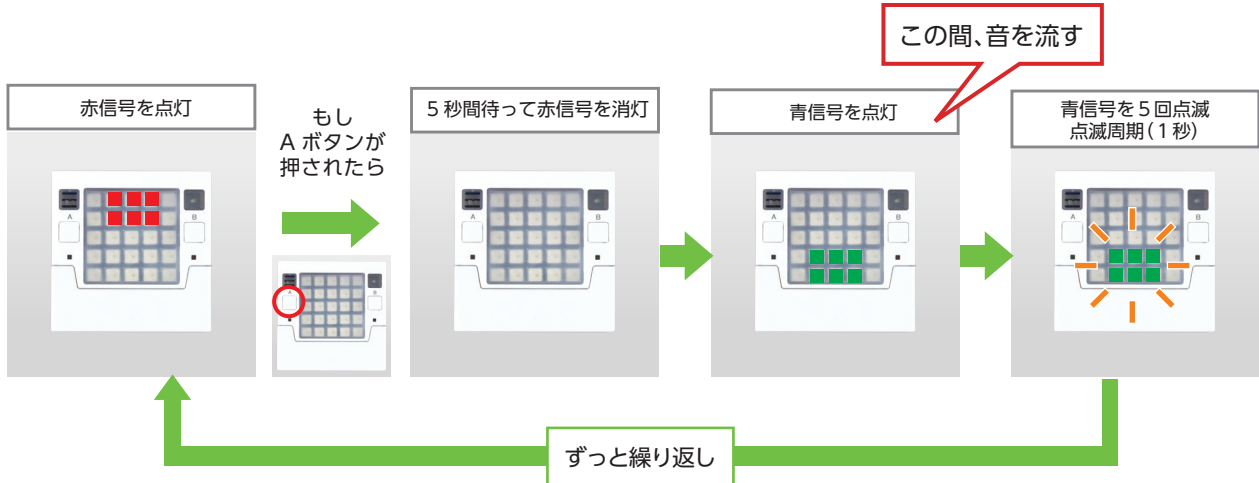
発展学習① ブザーをつかった制御

<音響装置付信号機のプログラム>

目の不自由な人のために、信号が変わったことを音で知らせてくれる信号機があります。

これを音響装置付き信号機といいます。

押しボタン式信号機のプログラムで青信号が点灯している間、ブザーの音を鳴らすプログラムを追加して音響装置付き信号機にしてみましょう。



①ブザーの音を鳴らすプログラム

ブザーから 60 を鳴らす ……………指定した高さの音を鳴らします。

ブザーから 60 を 0 秒間鳴らす ……指定した高さの音を指定時間鳴らす。

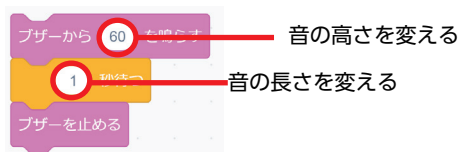
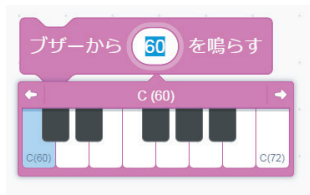
ブザーを止める ……ブザーからの音を止める。

やってみよう！

音の高さや長さを変えてみよう。

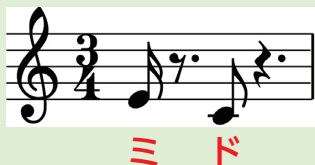
プログラム例

1 秒間音を鳴らしてとめるプログラム



②カッコウの音をつくろう

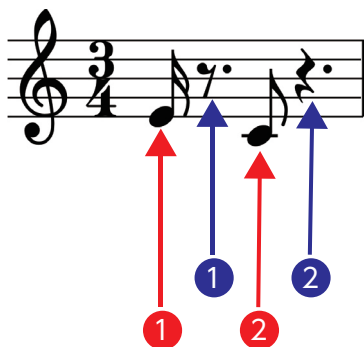
「カッコウ」の音は音符にすると
「ミ(64)・ド(60)」です。



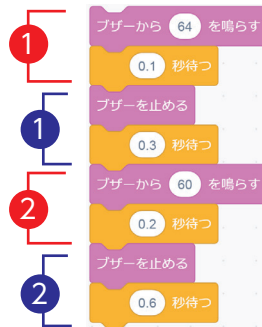
「カッコウ」の音の 長さの決め方

音楽には音の長さや音同士の間隔
などが決められた「楽譜」がありま
す。カッコウの曲にも楽譜があり、
今回のプログラムの 2 つの音と 2
回の休みの長さも実際の楽譜に合
わせて設定しています。

「カッコウ」の音の楽譜



「カッコウ」の音のプログラム



カッコウの音を 5 回繰り返す場合は、右図のように
「5 回繰り返す」ブロックで囲みます。



③青信号が点灯している間音が鳴るようにしよう

関数ブロックを作成します。

The diagram illustrates the process of creating a function block in a block-based programming environment. It starts with a sidebar on the left where the '関数' (Function) category is selected. A '関数を作る' (Create Function) button is highlighted. This leads to a '関数を作る' (Create Function) dialog box where 'buzzer' is entered as the function name. A red arrow points to the 'buzzer' name input field with the text '関数名を入力します。(ここでは buzzer とします)' (Enter the function name. (Here, we use buzzer)). Another red arrow points to the 'OK' button with the text '関数名を入力 (半角英数字)' (Enter function name (half-width alphanumeric)). A final red arrow points to the '関数 buzzer' block in the script area, which contains a sequence of blocks: '5 回繰り返す' (Repeat 5 times), 'ブザーから 64 を鳴らす' (Play sound 64 from buzzer), '0.1 秒待つ' (Wait 0.1 seconds), 'ブザーを止める' (Stop buzzer), '0.3 秒待つ' (Wait 0.3 seconds), 'ブザーから 60 を鳴らす' (Play sound 60 from buzzer), '0.2 秒待つ' (Wait 0.2 seconds), 'ブザーを止める' (Stop buzzer), and '0.6 秒待つ' (Wait 0.6 seconds). Text on the right explains that the function definition block '関数 buzzer' is added to the script area.

関数名を入力します。(ここでは buzzer とします)

関数名を入力 (半角英数字)

関数の定義ブロック「関数 buzzer」に②で作成したプログラムをつなげます。

クリック

練習課題で作成した押しボタン式信号機の青信号が点灯した後の「5 秒待つ」と、関数の呼び出しブロック「buzzer」を入れ替えます。

The diagram shows a script in the 'コード' (Code) tab. The script starts with a 'がクリックされたとき' (When clicked) event block, followed by a 'ずっと' (Forever) loop. Inside the loop, there are several blocks: 'LEDに [] を表示させる' (Show [] on LED), a conditional block 'もし 本体のボタン A = 0 なら' (If button A = 0), a '5 秒待つ' (Wait 5 seconds) block, 'LEDを全て消灯' (Turn off all LEDs), 'LEDに [] を表示させる' (Show [] on LED), a 'buzzer' function call block, '5 回繰り返す' (Repeat 5 times), 'LEDに [] を表示させる' (Show [] on LED), '0.5 秒待つ' (Wait 0.5 seconds), 'LEDを全て消灯' (Turn off all LEDs), and another '0.5 秒待つ' (Wait 0.5 seconds) block. A red arrow points from the 'buzzer' function call block in the script to the 'buzzer' function definition block in the '関数' (Function) sidebar, indicating that the function is being called.

発展学習② 光センサーをつかった制御

<センサーライトのプログラム>

周囲が暗くなったら自動で照明が点灯する、センサーライトのプログラムを考えましょう。

- ①「編集」より接続を選択しコンピュータと接続してください。

センサーボード	
Studuino:bit	
ボタンA	1
ボタンB	1
光センサー	6
温度センサー	136.7
加速度センサー X	0.05
加速度センサー Y	-0.17
加速度センサー Z	0.98
ジャイロセンサー X	0
ジャイロセンサー Y	-2
ジャイロセンサー Z	2
磁気センサー X	94
磁気センサー Y	-76
磁気センサー Z	-92



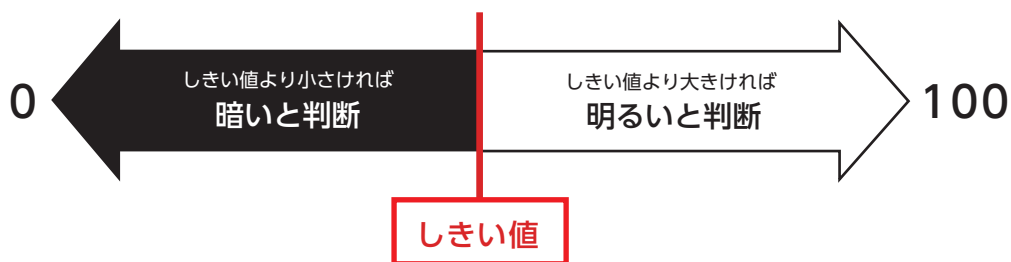
- ②センサーボードで光センサーの値を確認します。

光センサーを手で覆って暗くするとセンサーの値がどのように変化するか確認しましょう。

- ③しきい値の決定

どの程度暗くなったら LED を点灯させるかは②でしらべた光センサーの値で決めます。

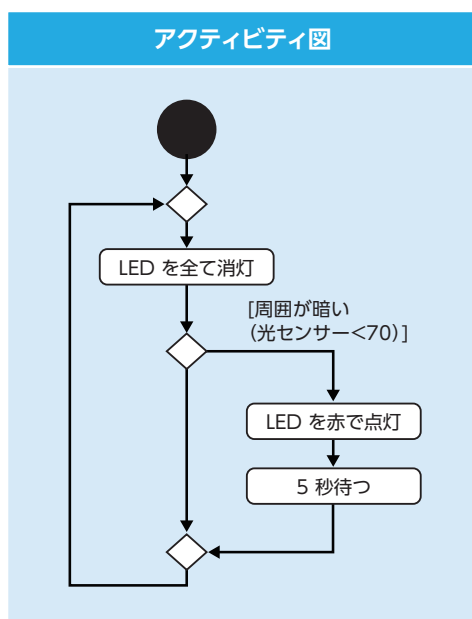
このような点灯させる条件となる値のことを「しきい値」といいます。



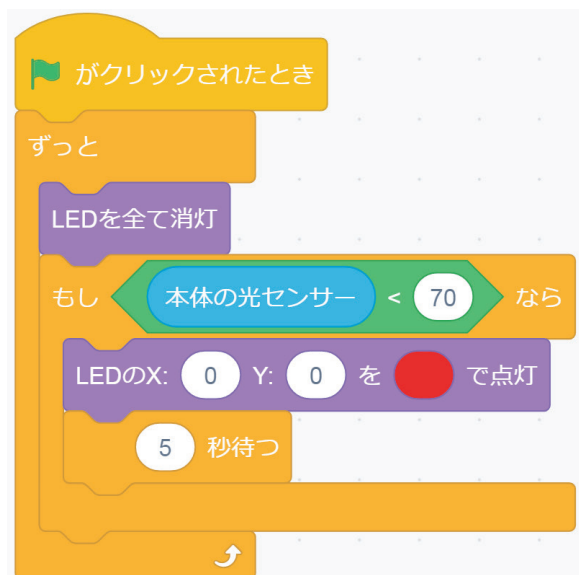
- ④光センサーの値が一定以上小さくなったら

5 秒間 LED を点灯させて消灯させるプログラムを考えましょう。

条件は と を組み合わせてつくることができます。



プログラム例



応用学習 テーマ1

金属回収ロボ

<学習内容>

0. 組み立て

技術の見方・考え方：社会からの要求

1. 直進だけでは回収範囲が狭い → 方向転換する機能

技術の見方・考え方：安全性の観点、システム

2. 物に触れても動き続ける → 接触検知機能

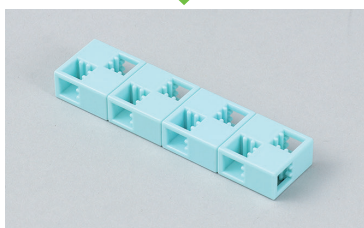
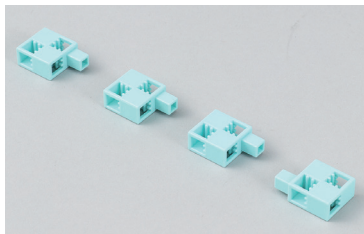
3. 机の端から落ちてしまう → 落下防止機能

4. 前方に壁があっても突っ込んでしまう → 壁の検知機能

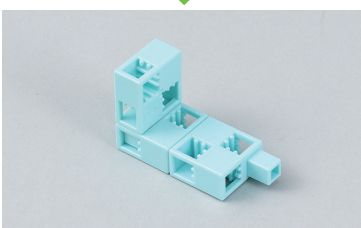
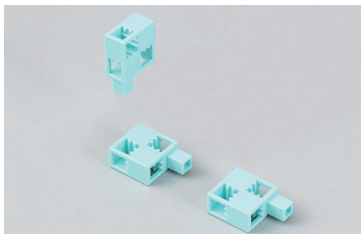
0. 金属回収ロボ

組み立て

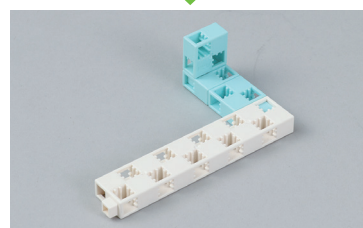
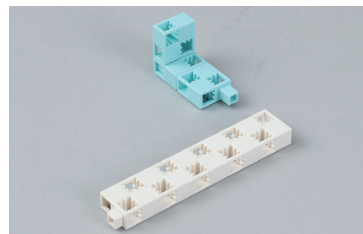
①



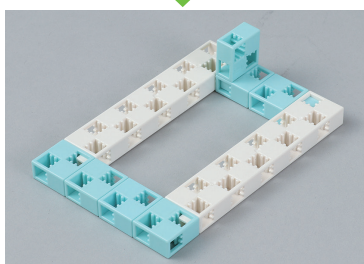
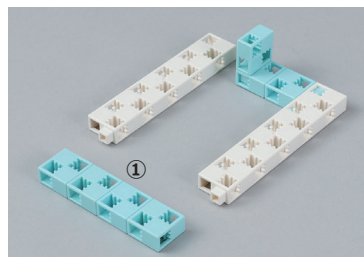
②



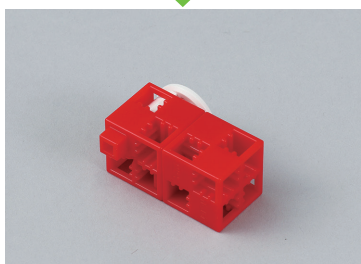
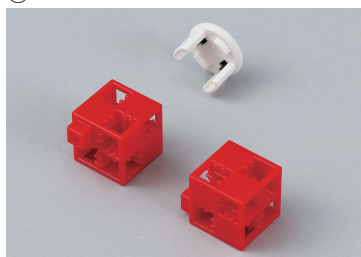
③



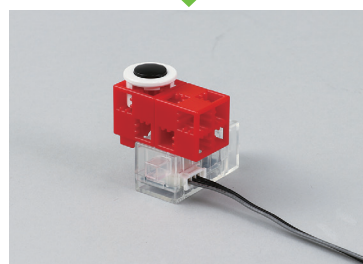
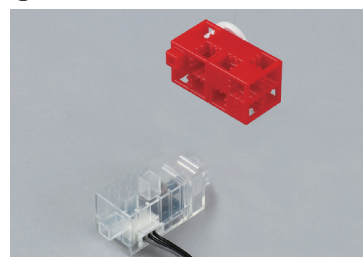
④



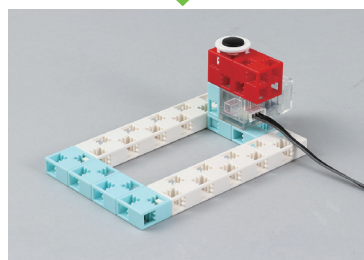
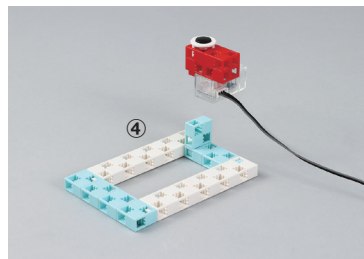
⑤



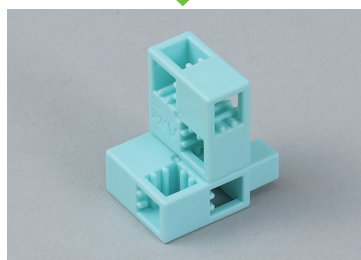
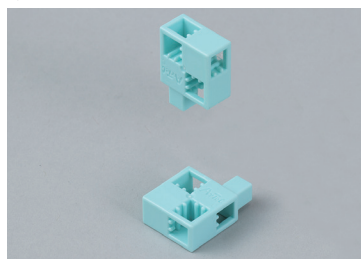
⑥



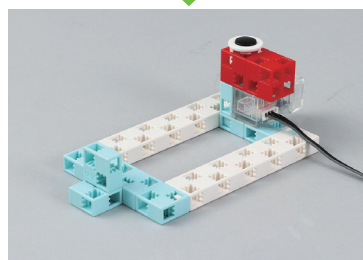
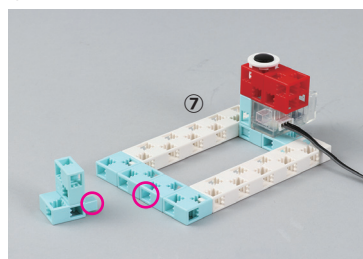
⑦



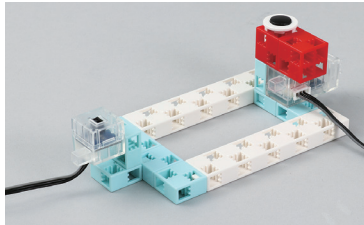
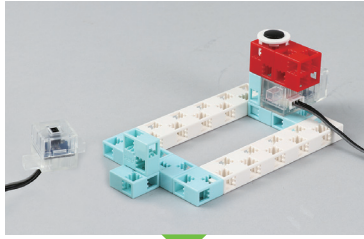
⑧



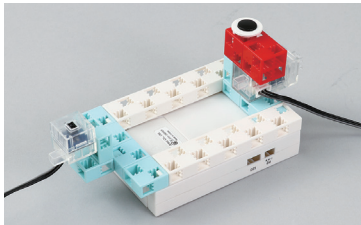
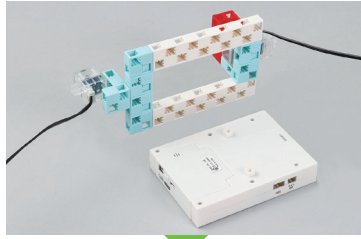
⑨



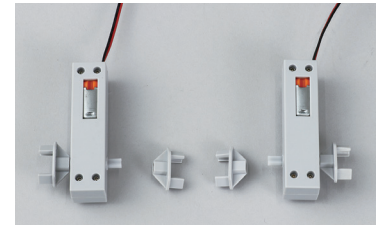
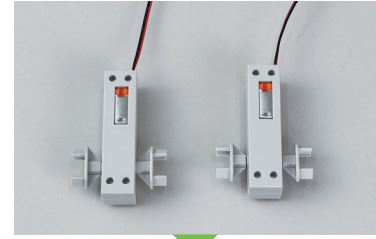
⑩



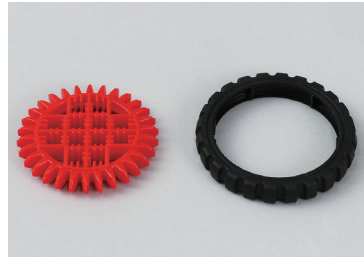
⑪



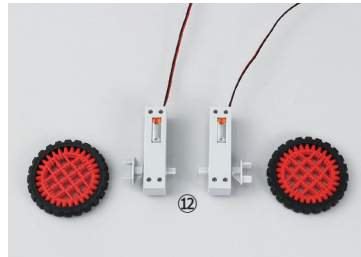
⑫



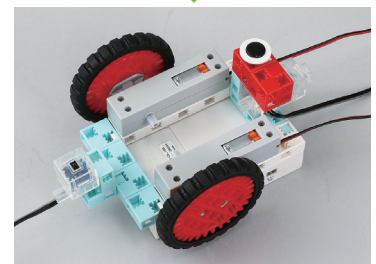
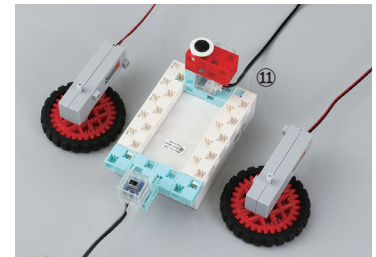
⑬ x2



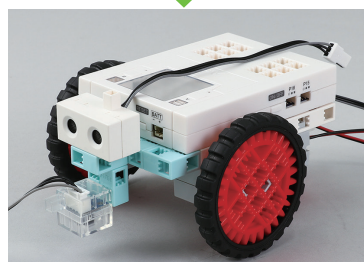
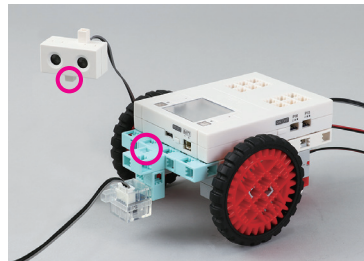
⑭



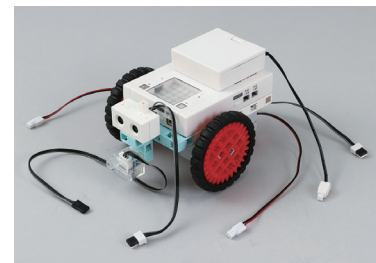
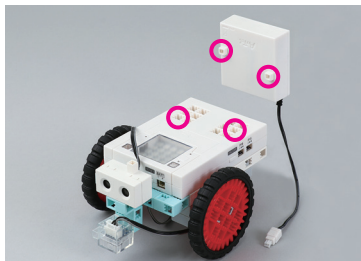
⑮



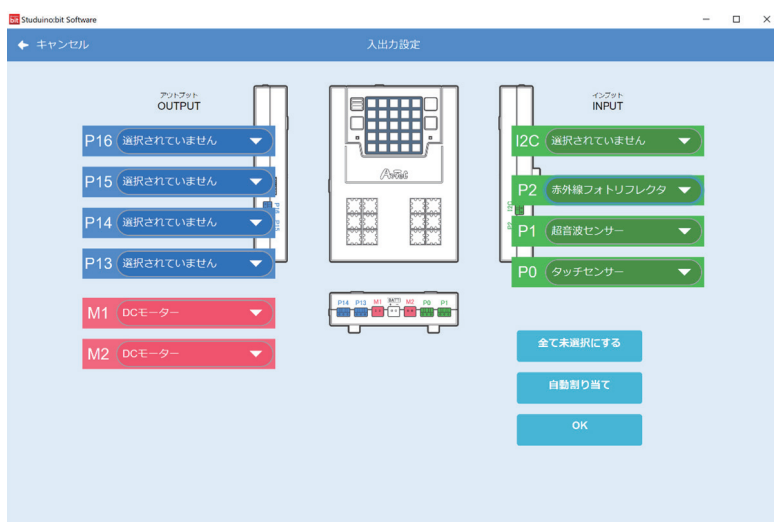
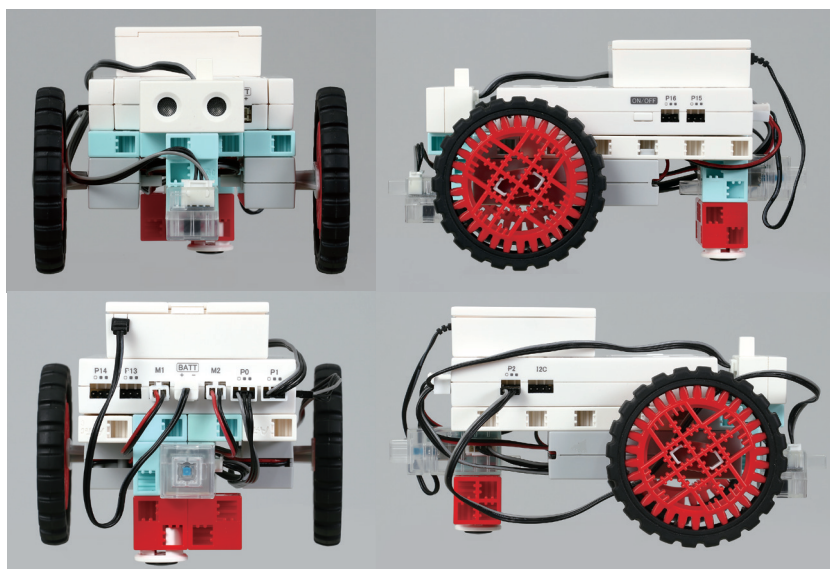
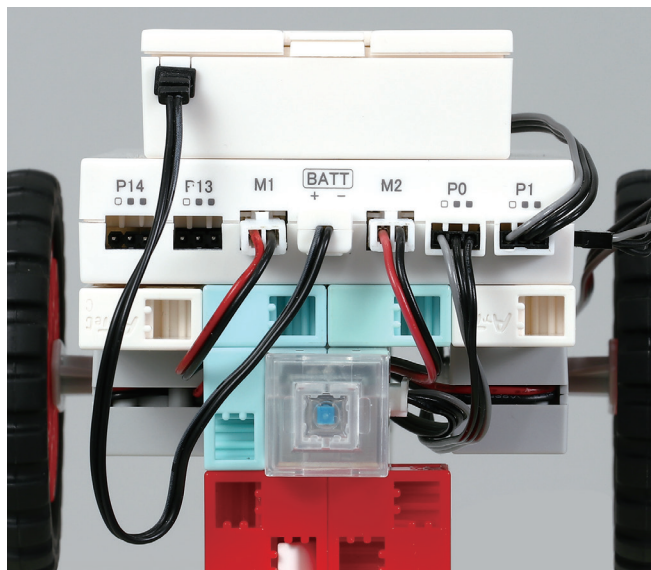
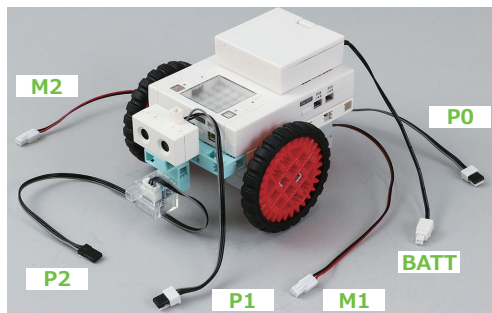
⑯



⑰

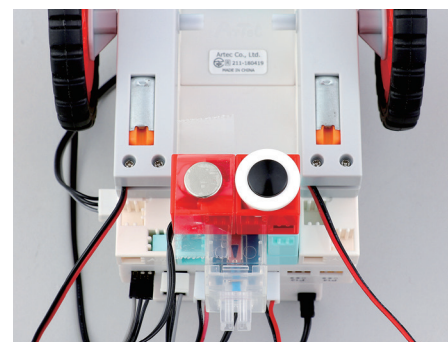


18



ネオジム磁石の取り付け

用意するもの：セロハンテープ



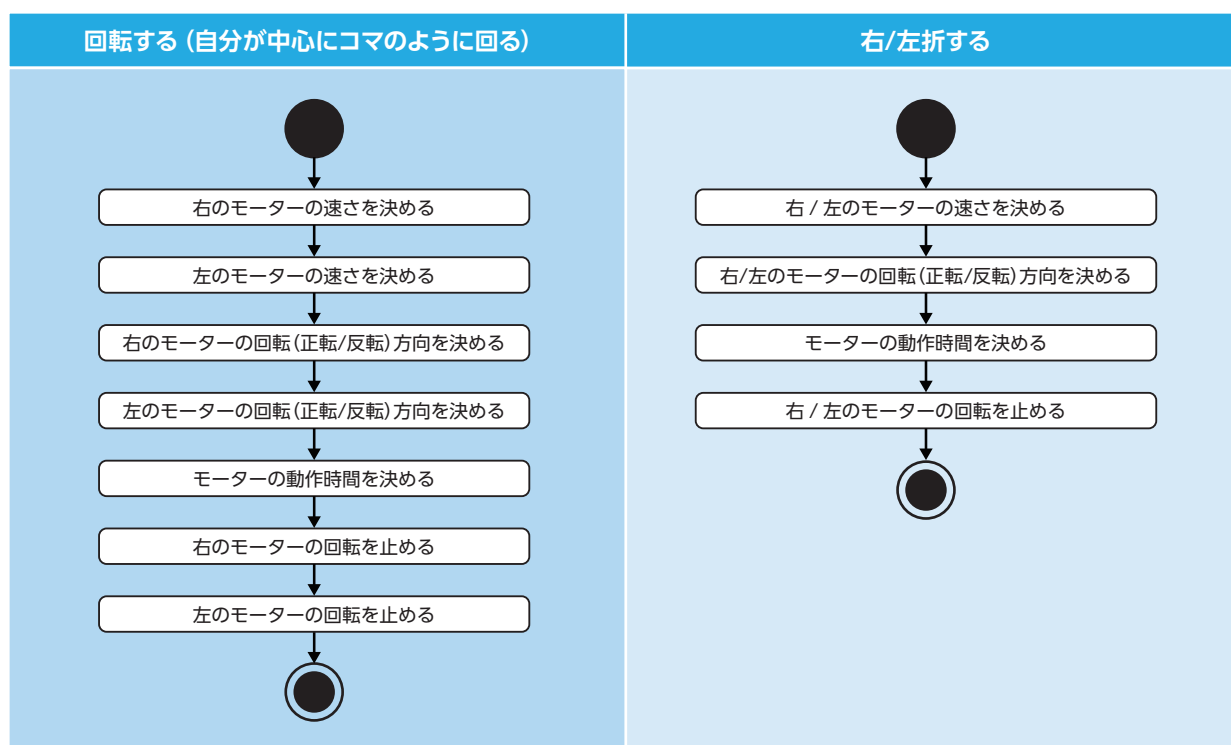
セロハンテープで固定する。

1. 直進だけでは回収範囲が狭い → 方向転換機能

プログラムについて

問題	直進しかできない。
課題の設定	回転する（自分が中心にコマのように回る）、右左折する。
処理	左右のモーターの回転方向、モーターの動作時間設定。 （向けたい方向に向くために、何秒モーターを動作させれば良いか実験して調べる。） （電池の残量が十分にあるものとし、車輪が空転しない程度の床面との摩擦があるものとする。）

アクティビティ図例

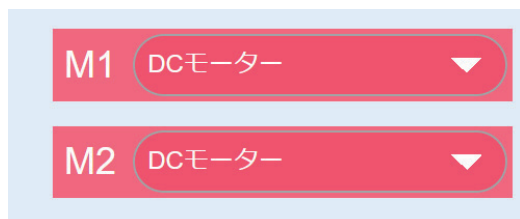


プログラム例（ロボットモード）



方向転換機能のつくり方のコツ

①編集→「入出力設定」で、M1、M2ともにDCモーターに設定変更します。



②ArtecRobo2.0のスプライトに対してプログラミングします。



③M1のモーターとM2のモーターを、左右どちらの車輪に設定するかで、モーターの正転/逆転が異なるので注意してください。

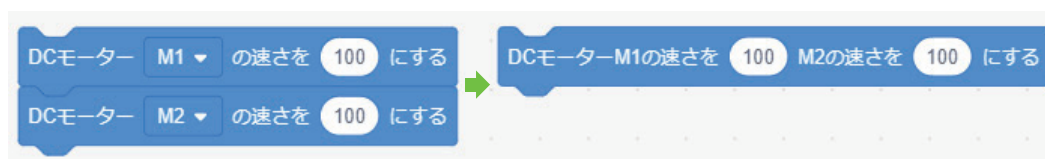
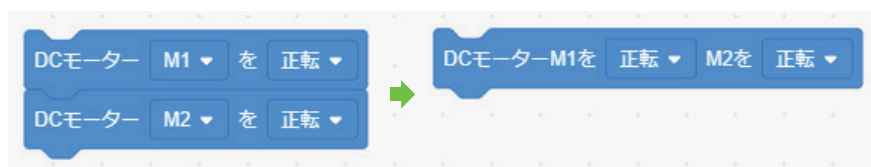
④方向転換するには、その場で回転するのか、向きを変えるのかの2つあるので目的に応じて処理を変えるようにしてください（弊社 計測・制御のプログラミングによる問題解決:p.38～42 参照 右上のQRコードからもご確認いただけます）。

<https://www.artec-kk.co.jp/dl/pfreetext/pdf/082025.pdf>

⑤本体には角度を検知する機能はないので、例えば90°回転させたい場合は、モーターを何秒動作させるか、つまりモーターを回転させたら、何秒待ってからモーターを止めるか事前に調べておく必要があります（待つ時間は、電池残量や床との摩擦、個体差等で微妙に変化します。「編集」→「モーター校正」でモーターの回転を調整することができます）。

このプログラムをさらに以下のように改良することもできます。

サンプルでは1つ1つの処理を書きましたが、1つのブロックで2つのモーターの動作を変更することもできます。このブロックを使えば、2つのモーターの動作を同時に変更することができます。

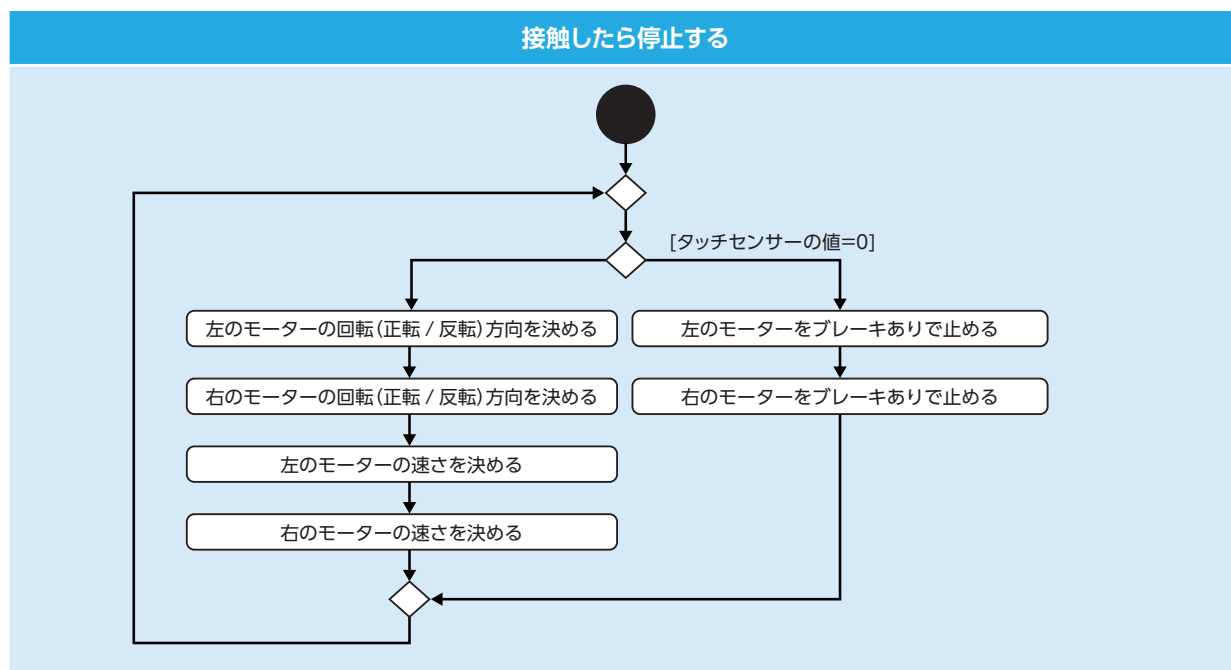


2. 物に触れても動き続ける → 接触検知機能

プログラムについて

問題	接触しているに関わらず動こうとする。
課題の設定	タッチセンサー（スイッチ）を使用して、導通の状態を利用して検知する。
処理	導通しているかどうかを調べて条件分岐させる。

アクティビティ図例



プログラム例（ロボットモード）

接触したら停止するプログラムのいくつかの例

①

③

②

④

タッチセンサーは通常状態は値が1で、導通状態（タッチした状態）で値が0になります。①は全て「もし」ですが、どちらかの状態だけに着目して「でなければ」にしたものが②です。

接触検知機能のつくり方のコツ

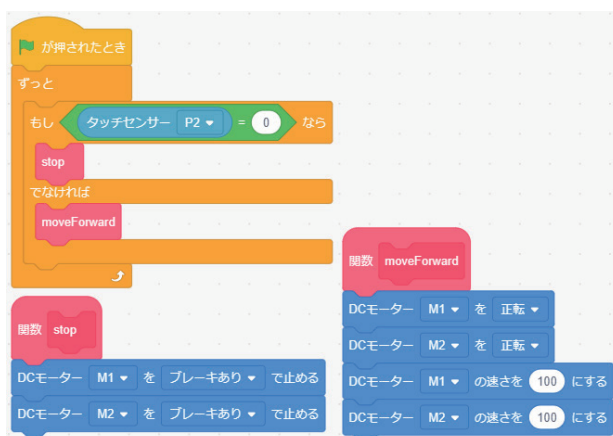
- ①アーテックロボ2.0と接続し「編集」→「入出力設定」を選び、どのコネクタに何のセンサーやアクチュエータを接続しているのかを設定します。P2では超音波センサーが使えないので注意してください。
- ②アーテックロボ2.0と接続し「編集」→「接続」をすることで、「センサーボード」を表示すると、接続されている現在の各種センサーの値をリアルタイムに知ることができます。
- ③タッチセンサーはタッチしている状態か否かの2種類の状態しかありません。それぞれの状態の時に値がどのようになっているのかを、実際に調べておきましょう。
- ④タッチセンサーは、導通しているかどうかで判別しているため、3端子のマイクロスイッチでも代替できます。

センサーボード	
Studuino:bit	
ボタンA	1
ボタンB	1
光センサー	21
温度センサー	37.82
モーションセンサー	▲
ArtecRobo2.0	
[P0] 赤外線フォトリフレクタ	0
[P1] 超音波センサー	0.0
[P2] タッチセンサー	1

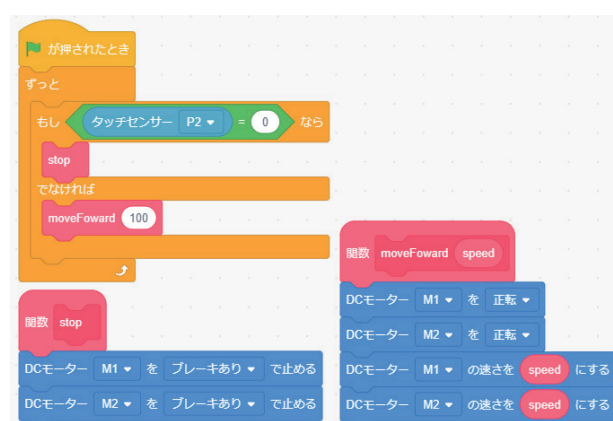
このプログラムをさらに以下のように改良することもできます。

プログラムを書いていると、機能ごとのまとまりに気が付くはずですが、例えば、「速さ100で前進」させるためには、モーターの左右の回転方向を決めて、モーターの左右の速度を決めるという処理のまとまりがあります。

まとまりを見つけることができたなら、そのまとまり毎に関数をつくって書くと、メインのプログラムをすっきりと見やすくすることができます。見やすいプログラムは、エラーの発見にも役立ちます。



関数には引数(ひきすう)を設定することで、上記の関数のスピードを変えることができます。下記では、speedという引数を追加して、速度を変化させられる関数をつくっています。ロボットモードでは関数名に日本語は使えません。

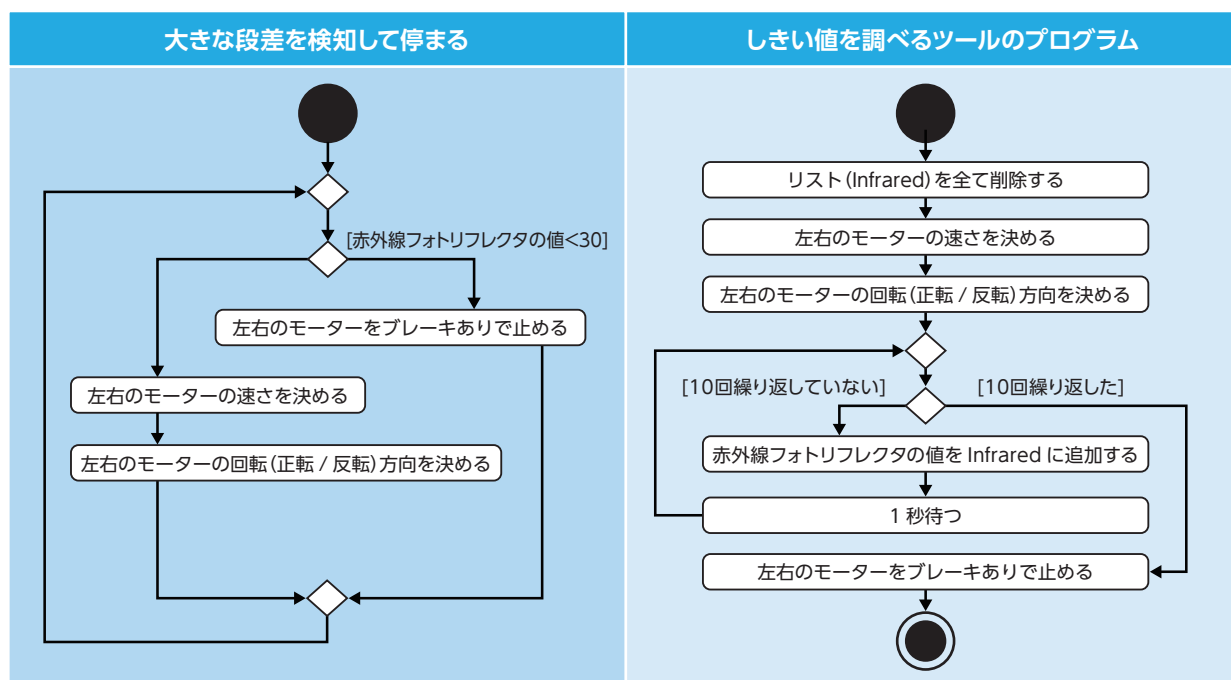


3. 机の端から落ちてしまう → 落下防止機能

プログラムについて

問題	前方の路面の状況（段差の大小、明暗）がわからない。
課題の設定	赤外線フォトリフレクタを使用して、赤外線反射の状態を利用して検知する。
処理	赤外線フォトリフレクタの値を調べて条件分岐させる。 (段差や明るさが変化する状況を再現して実験し、しきい値を決める。) (赤外線フォトリフレクタは環境光の影響を受けるので、誤差も考慮しておく。)

アクティビティ図例



プログラム例（ロボットモード）



しきい値を調べるツールのプログラムでは、「Infrared」（英語で赤外線という意味）というリストをつくって、1秒毎に赤外線フォトリフレクタの値を入れて数値を調べています。こうしてリストに入れたデータは、リストボックスの上で右クリックすると、データをダウンロードして他のアプリで再利用することができます。また1行に1ずつデータを書いたテキストファイルであれば、リストに読み込ませることもできます。



落下防止機能のつくり方のコツ

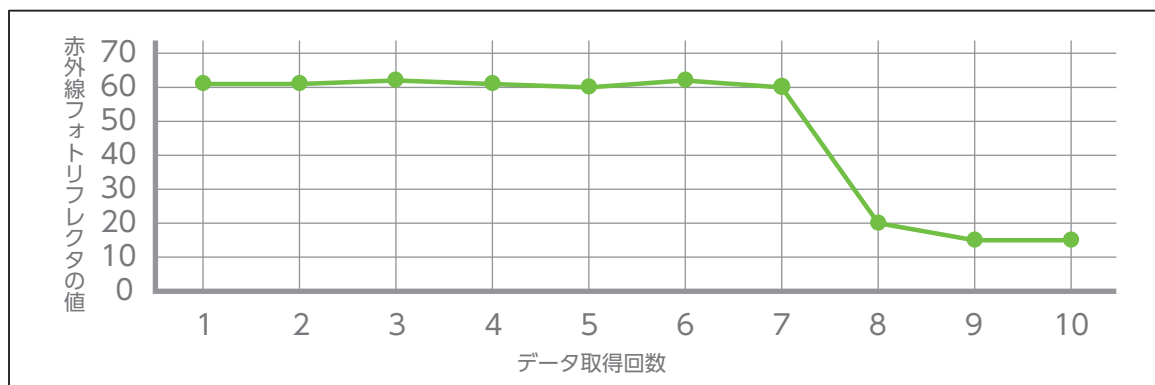
①編集→「入出力設定」で、P0～P2のどれかを赤外線フォトリフレクタに設定変更します。

P0 赤外線フォトリフレクタ ▼

②ArtecRobo2.0のスプライトに対してプログラミングします。



③調べたい個数分繰り返しの回数を設定し、実際に走行させ数値を調べてグラフを書きます。下のグラフはその例です。赤外線フォトリフレクタの値は細かい変動が多いので、こうしたグラフから誤差の影響を受けないしきい値を決定します。



このプログラムをさらに以下のように改良することもできます。

上記ではグラフを書いて大きく変化する箇所を目視で決定しましたが、よく見てみると段差がないときの値はほぼ60前後の値であることがわかります。そこで、段差がないときのデータで平均を求めて、そこから、例えば「その平均より5よりも大きい変化があったら」というプログラムを書けば、簡易的な自動学習機能をつくれます(余裕があれば、5などの固定の値ではなく、段差がないときのデータから標準偏差を使うアルゴリズムの方が確実です。数学との教科横断も考えられます)。

下記のプログラムはAボタンで平らな状態の10回の平均を「Flat_value」に入れて、緑の旗が押されたら、その平均よりも5小さくなったら、大きな段差があると判断して停まるようになっています。

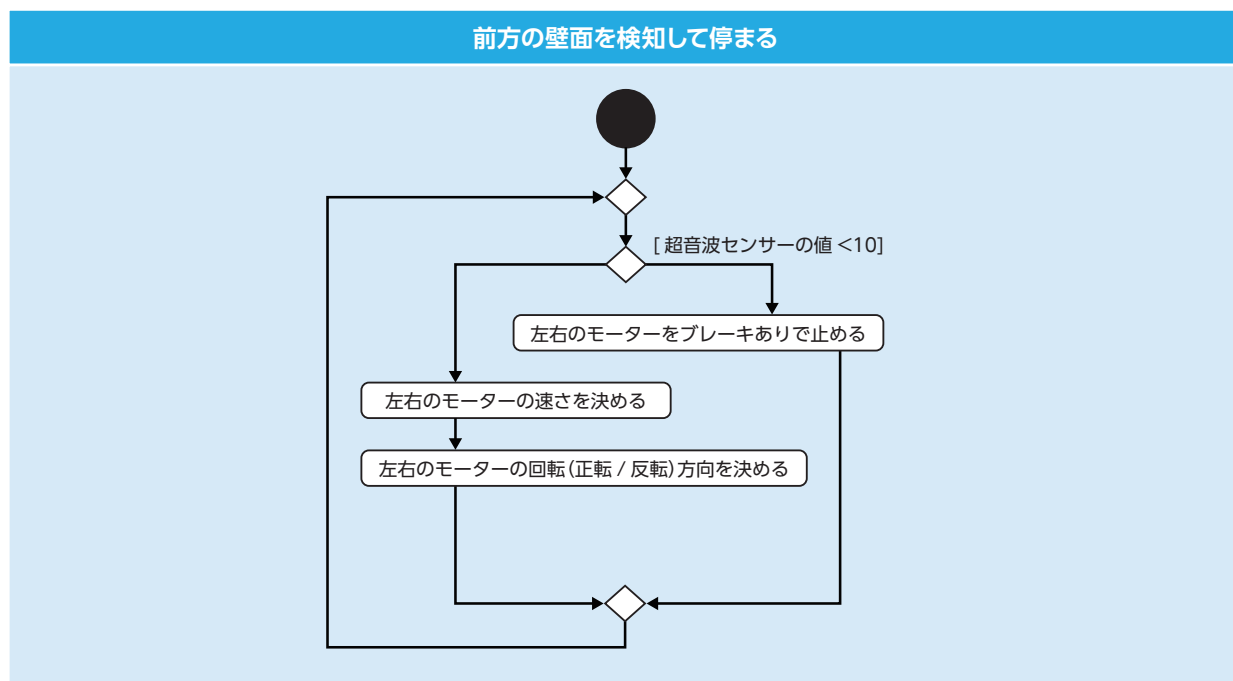


4. 前方に壁があっても突っ込んでしまう → 壁の検知機能

プログラムについて

問題	前方の壁面の状況（障害物の有無）が分らない。
課題の設定	超音波センサーを使用して、超音波の反射の状態を利用して検知する。
処理	超音波センサーの値を調べて条件分岐させる。 （障害物との距離を再現して実験し、しきい値を決める。）

アクティビティ図例



プログラム例（ロボットモード）

前方壁面10cmを検知して停まる

センサー値の設定

超音波センサー値の単位

☒ 距離 (cm) ☐ 時間 (ミリ秒)

キャンセル OK

超音波センサーでは「センサー値の設定」メニューより計測する値を距離か時間かを設定することができます。

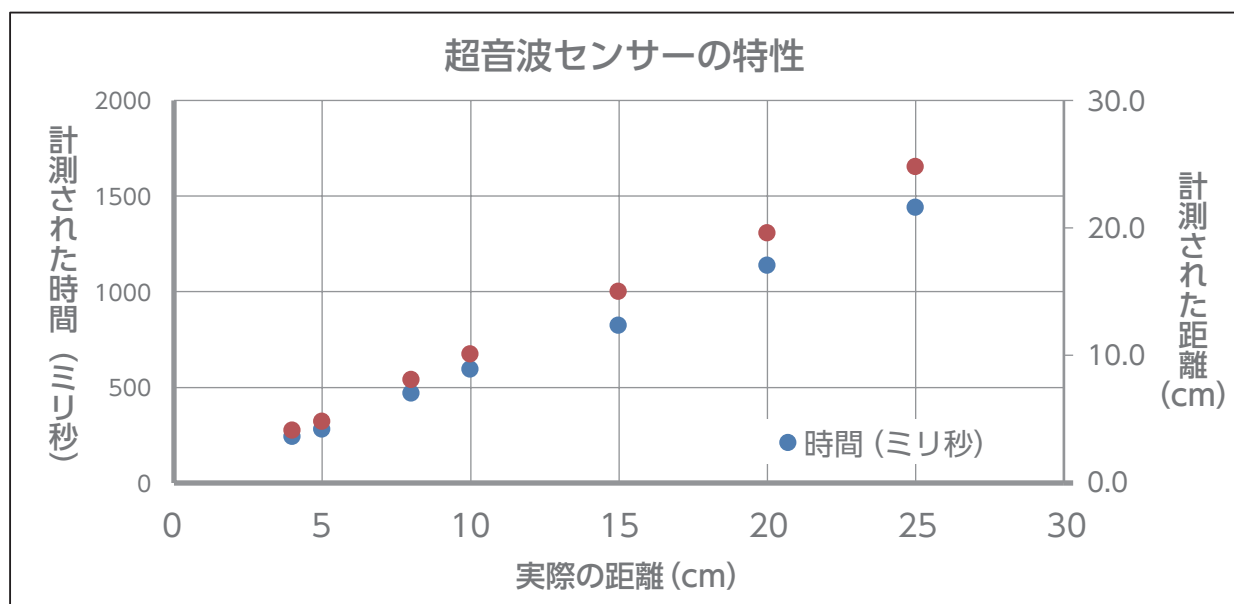
衝突防止機能の作り方のコツ

①編集→「入出力設定」で、P0～P1のどれかを超音波センサーに設定変更します(ver1.4.2時点)。P1 超音波センサー

②ArtecRobo2.0の spriteに対してプログラミングします。



③超音波センサーでは赤外線フォトリフレクタほど値の変動がありませんが、壁面の向きによって値が変わるので注意が必要です。下のグラフは実際の距離とセンサーの値2種類を計測した例です。



(要注意) このプログラムは、こうするとうまくいきません。

有線で接続し「緑の旗」をクリックすると、障害物がしきい値よりも小さい距離に来ると止まるのですが、一定間隔で一瞬移動してしまいます。

「転送」でプログラムを書き込んだ場合は、障害物が近づいても停止しません。



応用学習 テーマ2

自宅のセキュリティ対策

<学習内容>

0. 組み立て

技術の見方・考え方：社会からの要求

1. 人が常時監視することができない → 状況の変化を検知する機能

技術の見方・考え方：安全性の観点、システム

2. より確実に守りたい → 複数のセンサーを関連させて検知

- ・サーボモーターを使用した作例の組み立て

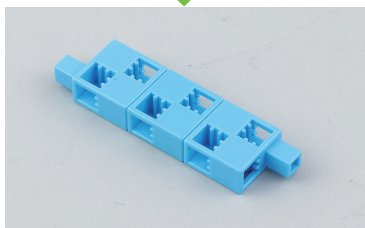
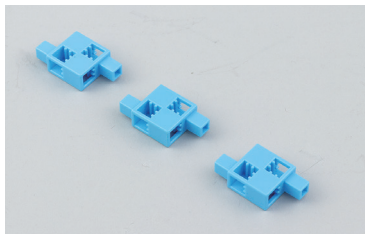
- 離れた場所にも通知

- 動きで通知

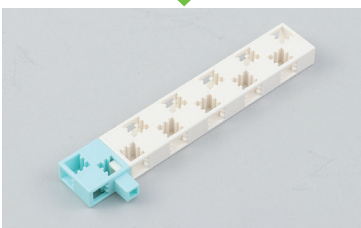
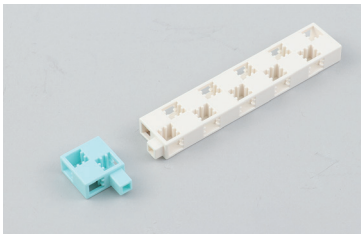
0. 自宅のセキュリティ対策

組み立て

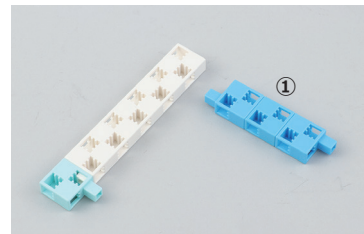
①



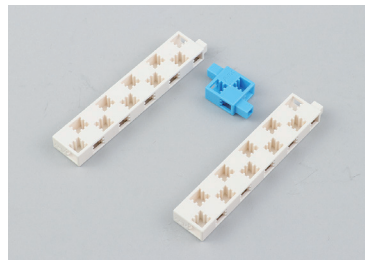
②



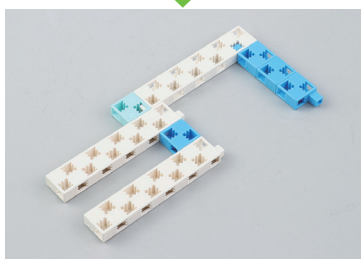
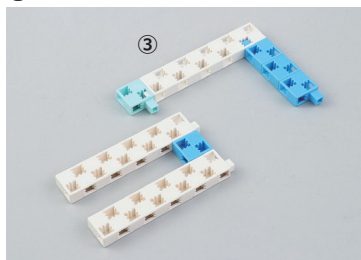
③



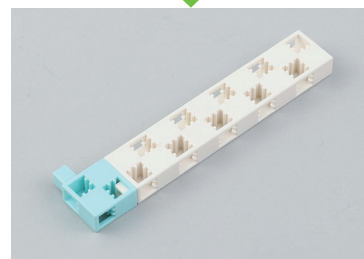
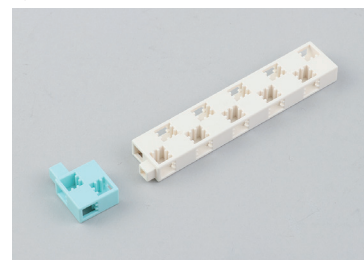
④



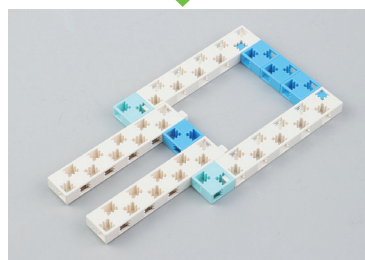
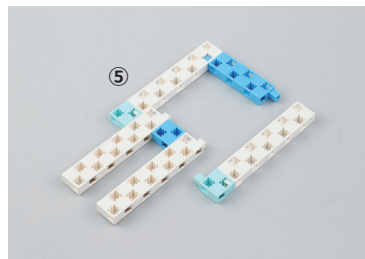
⑤



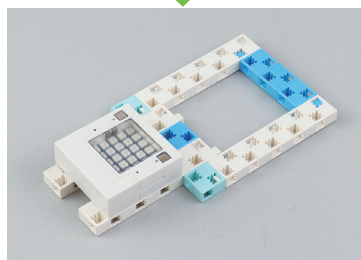
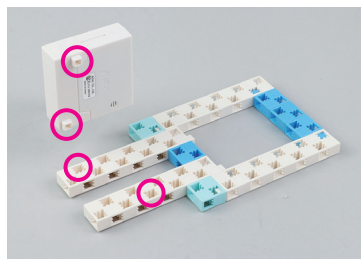
⑥



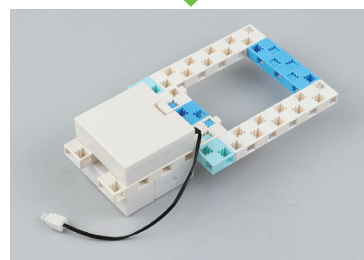
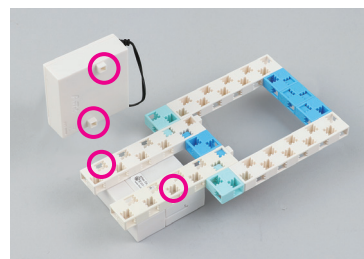
⑦



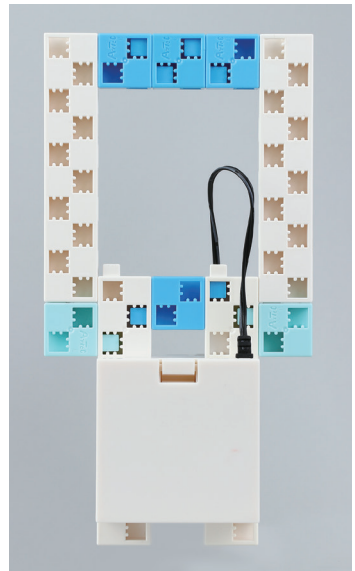
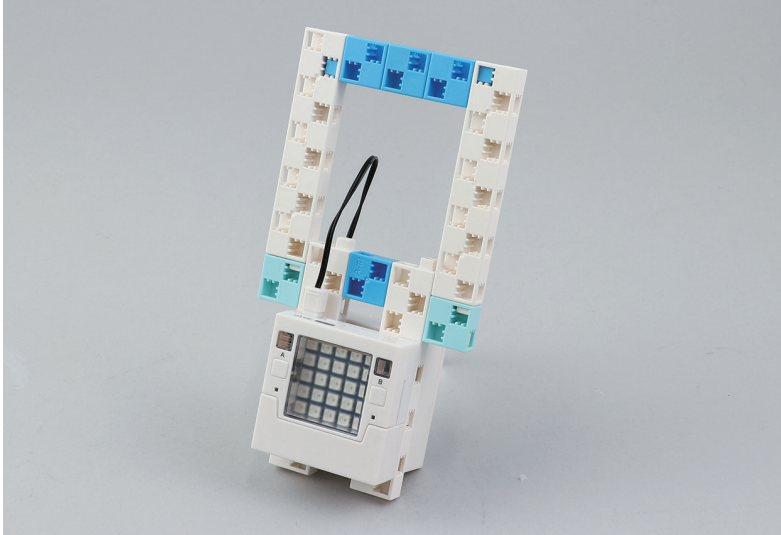
⑧



⑨



10

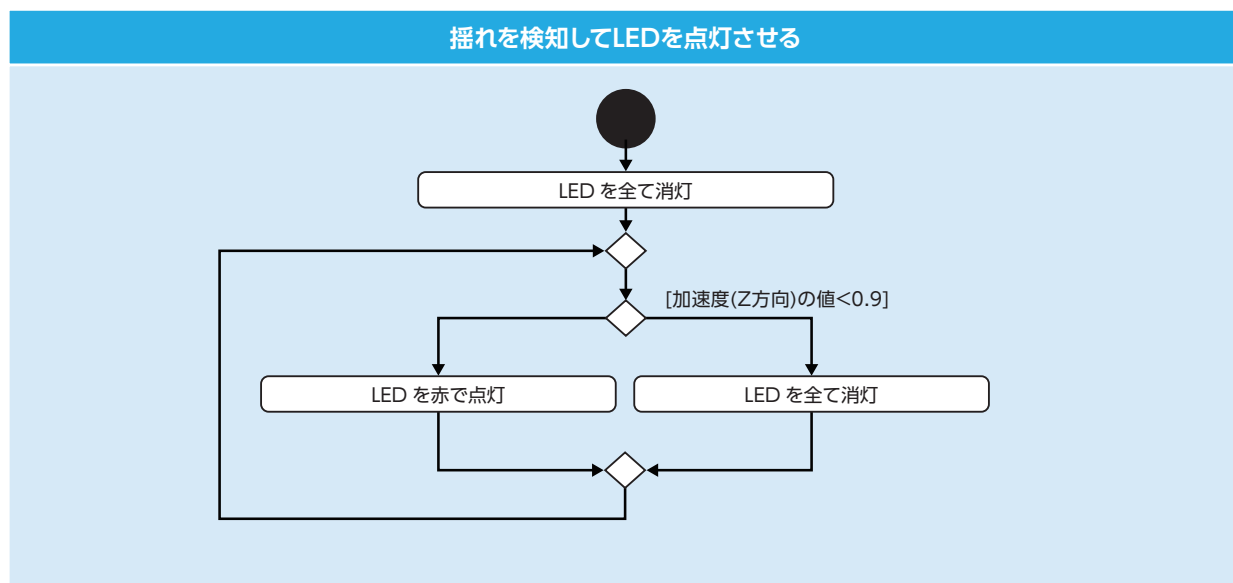


1. 人が常時監視することができない → 状況の変化を検知する機能

プログラムについて

問題	人が常時監視できない場所に異常が生じても気が付けない。
課題の設定	異常の状態（揺れる）に反応するセンサーの値を調べて通知する。
処理	生じる異常の状態を検知できるセンサーを検討する。 しきい値を調整する。

アクティビティ図例



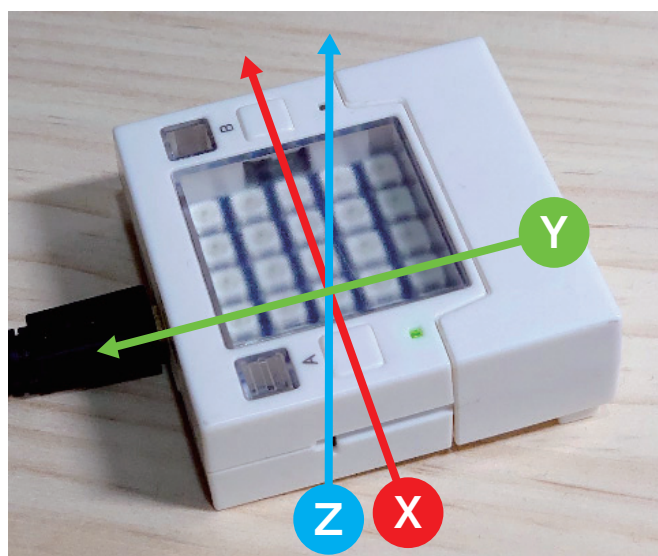
プログラム例（ロボットモード）



- 今回は、LED面の方向に動いたことを検知するようしきい値を設定しています。
- 加速度センサーは、移動した方向によって正負が異なるので、本体の向きなども注意してください。

状況の変化を検知する機能の作り方のコツ

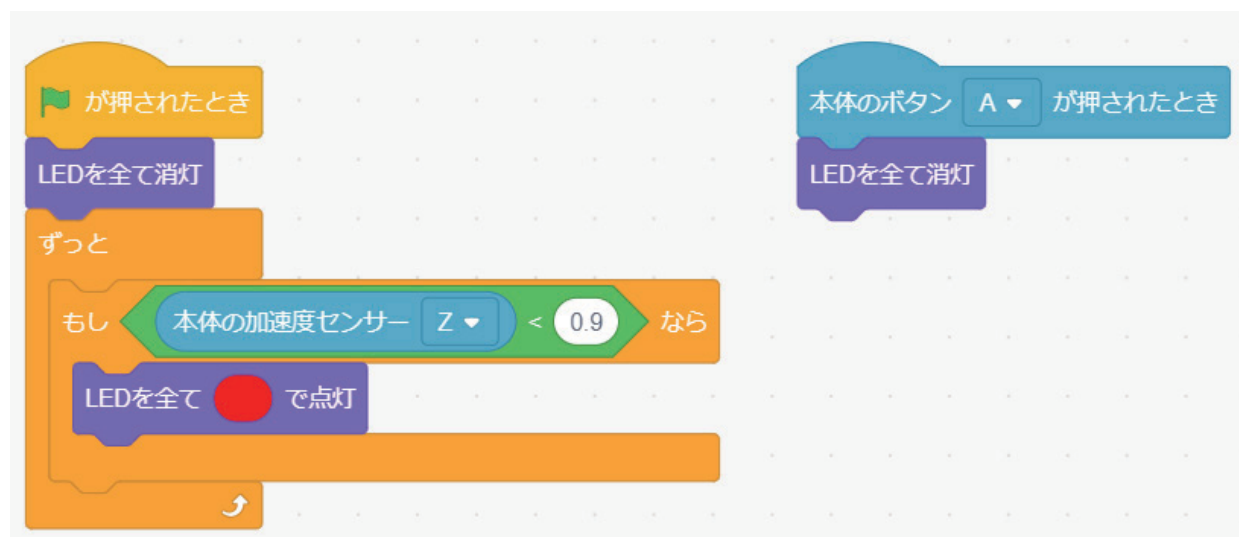
揺れを検知するとき、どの方向に動くのか、その時のセンサはどのような変化をするのかを調べます。



センサーボード	
Studuino:bit	
ボタンA	1
ボタンB	1
光センサー	35
温度センサー	38.52
モーションセンサー	▼
加速度センサー X	0.03
加速度センサー Y	-0.11
加速度センサー Z	1.06
ジャイロセンサー X	0
ジャイロセンサー Y	0
ジャイロセンサー Z	0
磁気センサー X	-3
磁気センサー Y	-14
磁気センサー Z	-55

このプログラムをさらに以下のように改良することもできます。

先のサンプルでは一瞬の加速度で点灯して消灯します。確実に気がつく・見落とさないという確実性や安全性の観点では、一度点灯したら消えずに、そのまま点灯し続けた方がよい場合があります。下記はAボタンでリセットするまで点灯したままにする例です。

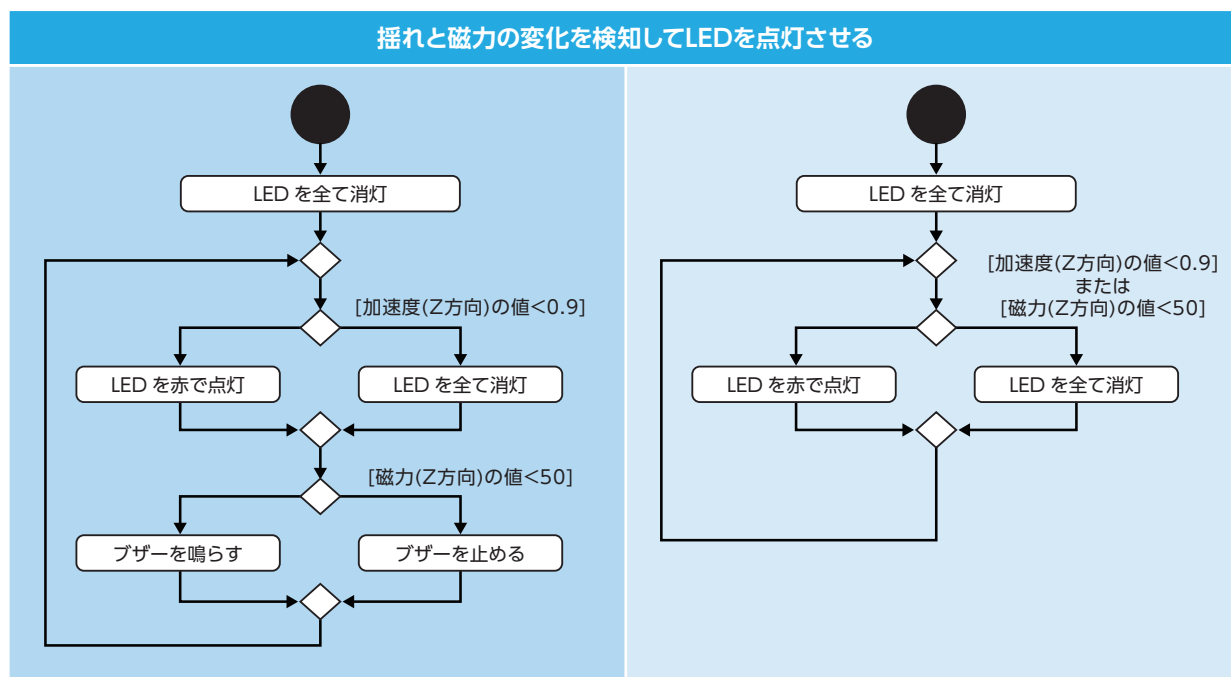


2. より確実に守りたい → 複数のセンサーを関連させて検知

プログラムについて

問題	1つのセンサーだとそれ以外の変化が生じても気づけない。
課題の設定	意図的に状況が変化するように環境を工夫する（磁力を応用する）。
処理	マグネットを固定し、アーテックロボ2.0を可動部分に設置することで、アーテックロボ2.0とマグネットの距離が変化するように設置する。 方位を知るための地磁気センサーを利用して、周辺の磁力の変化を検知する。

アクティビティ図例



プログラム例（ロボットモード）



センサー毎にLED点灯とブザーの反応とを変える場合(左)。判別する条件式を「または」でつなげて、どちらかのセンサーが反応した場合にLED点灯させる場合(右)。

ジャイロセンサーも本体の向きや磁石の向き、位置関係によって値が大きく異なります。実際に設置する場所に合わせて閾値を適切に設定してください。

複数のセンサーを関連させて検知のつくり方のコツ

- ①ここでは、単なる状況や状態変化だけでなく、変化する環境を意図的につくるようにしています。そこで今回使用するのがネオジム磁石です。写真の右側に見える丸いものがネオジム磁石です。ドアノブにぶら下げたアーテックロボ2.0との距離が変わることで、磁力の強さが変化することを検知します。
- ②加速度センサー同様に3軸あるため、磁石の向きやアーテックロボ2.0の向きによって値の変化の仕方が変わります。実際の取り付け環境で値の変化を実験して把握する必要があります。
- ③このサンプルでは、例としてZ軸の変化を用いるようにしています。



このプログラムをさらに以下のように改良することもできます。

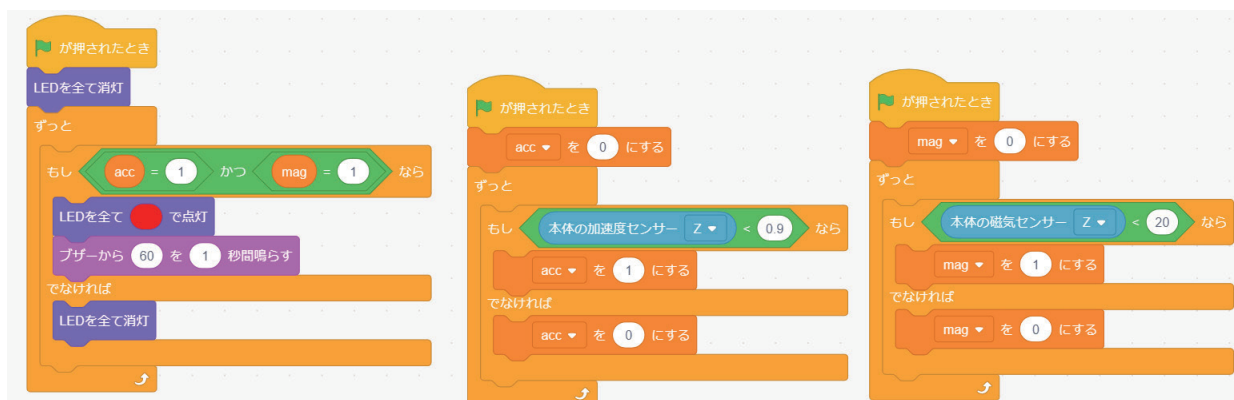
サンプルでは2つのセンサーのどちらかが反応した場合に、LEDの点灯をしています。複数の条件が揃ったときに点灯することができます。複数の条件にあうようにするためには、「かつ」(AND)という論理演算子を使います。Webの検索エンジンで、スペースで区切って複数のキーワードを入れることと同じ概念です。

また、状態を保持するための変数を作成してセンサーの値によって、その変数の中身を1か0か切り替える書き方もあります。テキストプログラミングでは、こうした2つの状態しか保存しない変数はブーリアンといって、TrueかFalseかのみを扱うものが用意されていることが多いです。

また、以下は緑の旗がクリックされると、3つのプログラムが同時に実行される並列処理になっています。これはスレッドと呼ばれるプログラミングテクニックです。プログラミングエディタで [Python](#) に切り替えると、thread (スレッド) という箇所があることで確認することができます。



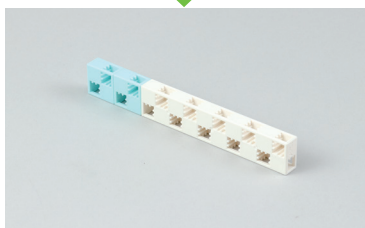
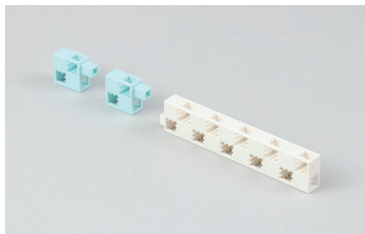
```
30- def thread_1():
31-     global _var_acc
32-     _var_acc = 0
33-     while True:
34-         if (getAccelZ() < 0.9):
35-             _var_acc = 1
36-         else:
37-             _var_acc = 0
38-
39-
40-
41- def thread_2():
42-     global _var_mag
43-     _var_mag = 0
```



2. より確実に守りたい

サーボモーターを使用した作例の組み立て

①



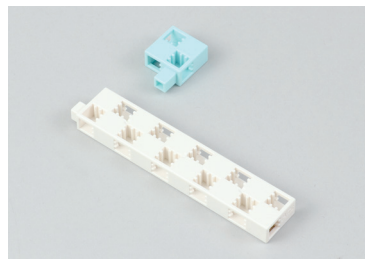
②



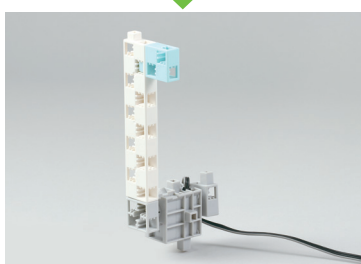
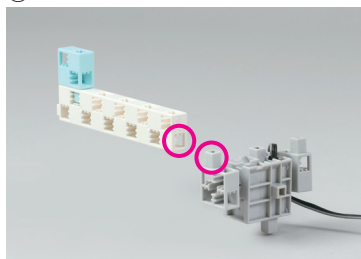
③



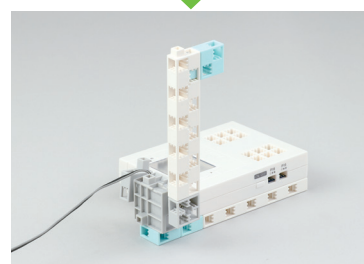
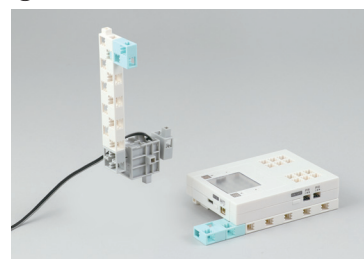
④



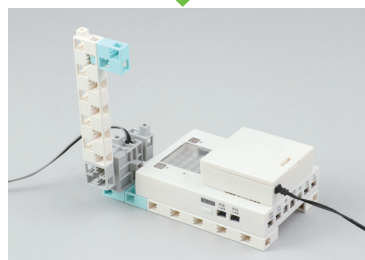
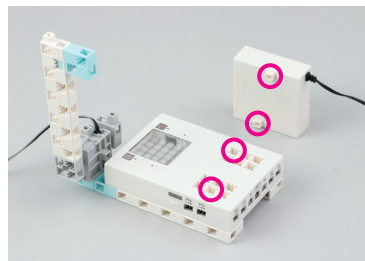
⑤



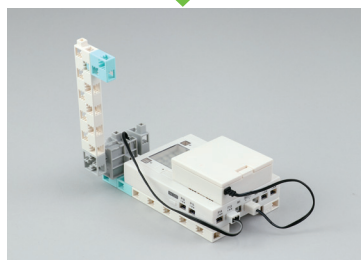
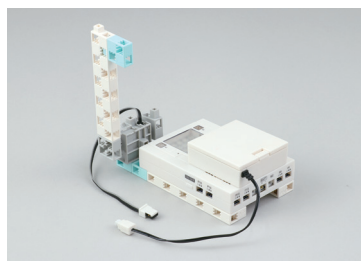
⑥



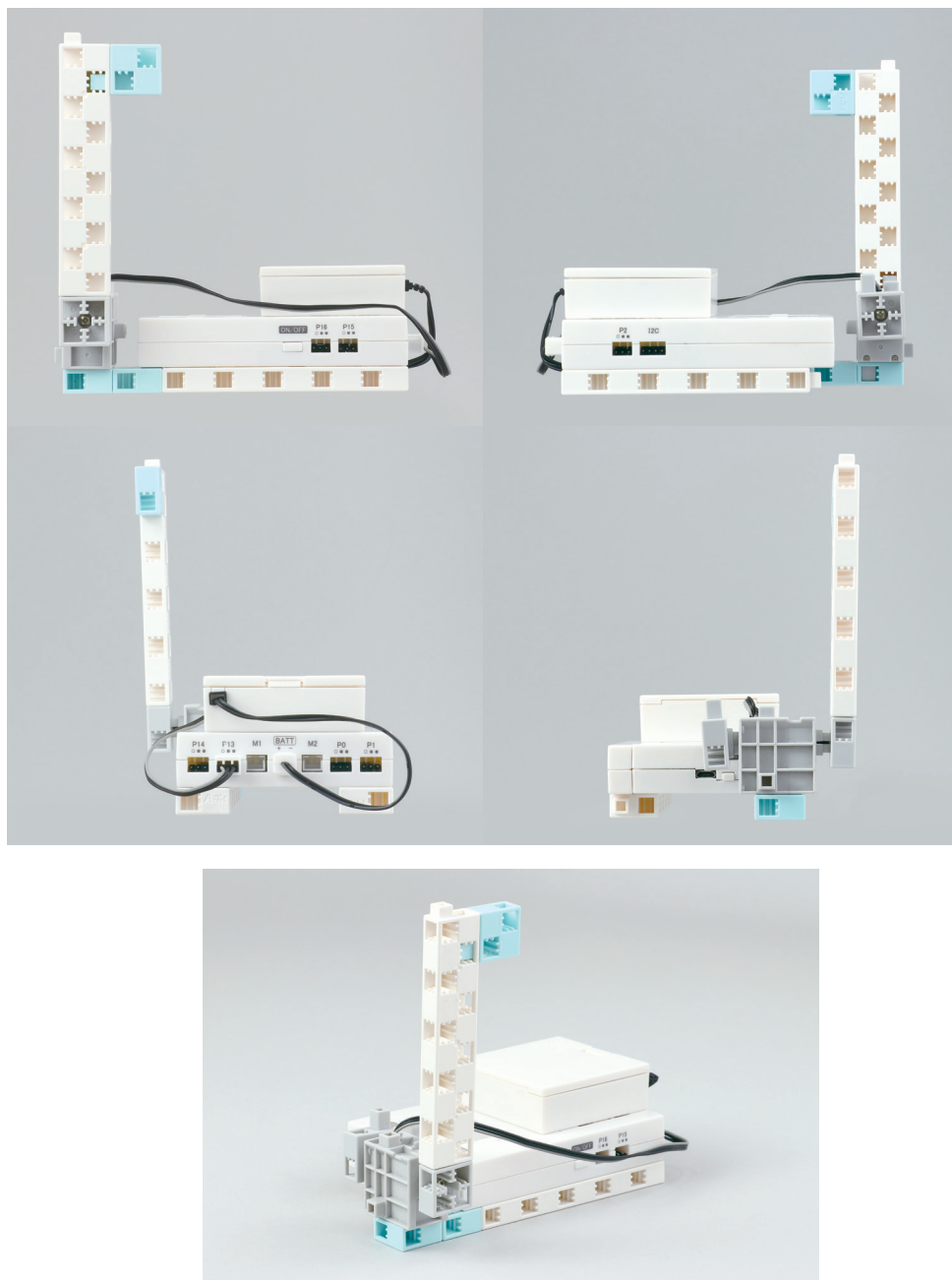
⑦



⑧



9



← キャンセル
入出力設定

**アウトプット
OUTPUT**

P16 選択されていません ▼

P15 選択されていません ▼

P14 選択されていません ▼

P13 サーボモーター ▼

M1 選択されていません ▼

M2 選択されていません ▼

**インプット
INPUT**

I2C 選択されていません ▼

P2 選択されていません ▼

P1 選択されていません ▼

P0 選択されていません ▼

全て未選択にする

自動割り当て

OK

2. より確実に守りたい → 離れた場所にも通知

※ここでは、アーテックロボ2.0を2台使用します。

プログラムについて

問題	その場にはないと通知を把握できない。
課題の設定	離れている場所に信号を送って、遠隔で通知を受信できるようにする。
処理	グループを決め、送信されたデータを色々な方法で表現する。

プログラム例（ロボットモード）

送信側	受信側

無線でのデータ送信は、以下のように書くこともできます。上記は合図を送るイメージなのに対して、下記は変数とその値を送るイメージです。

送信側2	受信側2

このプログラムをさらに以下のように改良することもできます。

キャラクターモードでつくと、ロボットモードとは違った方法でネットワークでの情報送受信、画面への変化、音など多様なメディアを統合したプログラミングができます（情報の技術スタンダードパッケージ 双方向性のあるコンテンツ編 p.5参照）。キャラクターモードでのネットワークを利用した通信では、通信したいArtecRobo2.0を同じネットワークグループに参加させます（「メニュー」→「編集」→「ネットワークに参加する」）。そうすることで、同じネットワークグループのプログラムでは、多少の遅延がありますが、リアルタイムに変数の値を共有することができます。ロボットモードとの違いは、無線のブロックがないことです。

ネットワークグループの指定

参加するグループを指定してください。[0-255]

0

キャンセル OK

送信側3 (キャラクターモード)	受信側3 (キャラクターモード)
	<p>ネットワーク上で送受信側で同じ名前となっている isOpened という変数の中身が共有される。</p>

キャラクターモードではスプライトを複数使用できるので、以下のようにメッセージを送れば、録音した声や音楽ファイルなどの音源を使用することができます。送信側のプログラムは上記と同じです。さらに、拡張機能から「音声合成」を追加すれば、パソコンのスピーカーで声を出力することができ、単なるアラート音に加えて、具体的な内容を伝えることができます。音しか聞こえない状況での安全性の機能として、こうした音声でのアラートが役に立ちます。

送信側4 (キャラクターモード)	受信側4 (キャラクターモード)	
	「音」からAlert音を選択	拡張機能で音声合成を追加

2. より確実に守りたい → 動きで通知

※ここでは、アーテックロボ2.0を2台使用します。

プログラムについて

問題	LEDが光るだけだと気がつきにくい。
課題の設定	サーボモーターで回転する角度を調整し、センサーがどの程度反応しているかを視覚的に把握しやすくする。 DCモーターを使い回転の動力として使用する。
処理	センサーの値によって、サーボモーターの角度を変える。 しきい値を超えたら、DCモーターを回転させる。

プログラム例（ロボットモード）

送信側	受信側

このサンプルでは、送信側は、1秒ごとに計測されたZ方向の磁力を送信することを基本機能としています。
(環境としては、磁石とArtecRobo2.0はドアが閉まっているときに最も値が小さく(負の数になるはず)なるようにセッティングしています)

ただし磁力の値は安定しにくく変化が大きいため、安定動作させるための工夫をしています。まずArtecRobo2.0と磁石との距離は環境次第なので、固定した値をプログラムに直接入れることができません。そのためアーテックロボ2.0のAボタンを押すことで、その時の磁気センサー(Z)の値をドアが閉じているときの値という意味で付けた「closed_value」という変数に入れています。そして、**ロボットモードでは負の数を送信できない**(p.5参照)ので、絶対値にして変数に代入しています。さらに、磁力の値は何もしなくても変動することがあるため、その変動幅をmarginという変数で、今回は仮に10としています。無線ネットワークで送信する値は、「mag」という変数名にしており、この「mag」には、計測した磁気センサーの値に「closed_value」と「margin」の値を足して、0に近い値を送信するようにしています。その後、無線で「Moved」という合図を送るという仕組みです。

受信側は、1秒ごとに送信される「mag」の値を元にして、ドアが開いたかどうかのしきい値(今回は150)を決めています。今回は、サーボモーターを90度回転させているだけですが、サーボモーターに旗を付けたり、何かの家電製品のスイッチを押させるなど分りやすく知らせる工夫をすることができます。

このプログラムをさらに以下のように改良することもできます。

今回のサンプルプログラムでは、磁気センサーの値が、ドアが閉まっているときは0に近く、大きくドアが開くほど(磁石とArtecRobo2.0の距離が大きくなるほど)センサーの値が大きくなるようにセッティングしています。

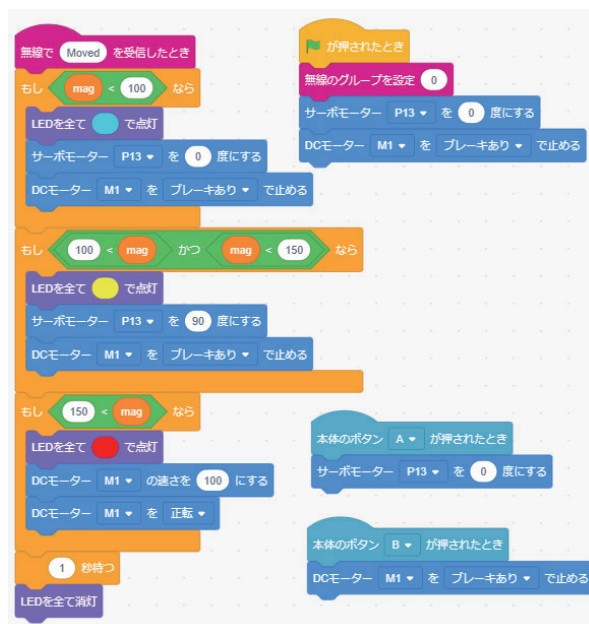
その値の揺れ幅を調べ、ドアの実際の開き具合を確認しながら、3段階に処理を分けるようにしましょう。ほぼドアが閉まっていると考えられるときには、安全を示す青のLEDを点灯させ、かつサーボモーターの角度を0度の向きに設定しています。つぎに、ドアが少し開いている状態は、magの値に幅を持たせるため、「かつ(AND)」でつないでいます。この時はLEDを黄色に点灯させ、サーボモーターを90度に動かし、視覚的な動きで注意喚起を促します。そして、ある値を超した場合は完全に開いた状態としています。



次のサンプルは、DCモーターの動作も用いて、ドアが大きく開いたときに連続した回転運動をさせています。これをプーリーにしたり、歯車の原動車として用いると、内容C エネルギー変換の技術と統合させることができます。

そして下のプログラムは、p.54の応用で磁気センサーの値を1秒ごとに10回読み取り、その平均と標準偏差を求めるツールのプログラムです。数学との教科間連携やSTEM教育的なアプローチになります。

Aボタンでデータを取得し、Bボタンで取得したデータの平均と標準偏差を計算しています。



value		average		SD	
1	40	1	-201	1	0.774596
2	35	2	-4.2	2	3.487119
3	31	3	20.3	3	1.615549
4	29	4	29.6	4	4.386342
5	27				
6	29				
7	27				
8	26				
9	26				
10	26				
+ 長さ 10 =		+ 長さ 4 =		+ 長さ 4 =	

応用学習 テーマ3

トレーニング補助装置

<学習内容>

0. 組み立て

技術の見方・考え方：社会からの要求

1. エクササイズを楽しみたい → ゲーミフィケーションの要素

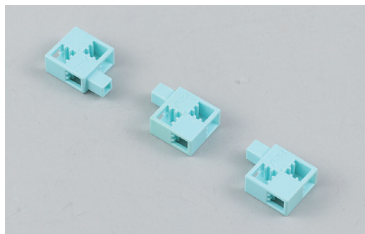
技術の見方・考え方：安全性の観点、システム

2. 色々なエクササイズをしたい → システムとして構成する

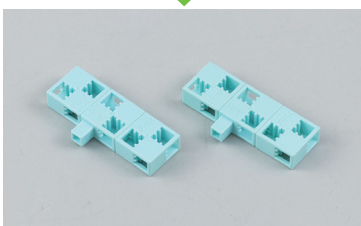
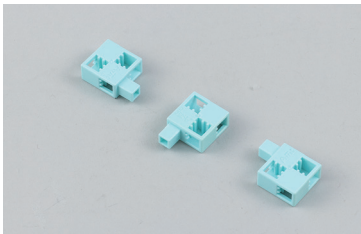
0. トレーニング補助装置

組み立て

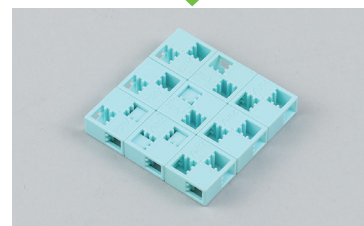
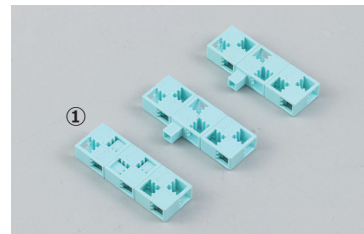
①



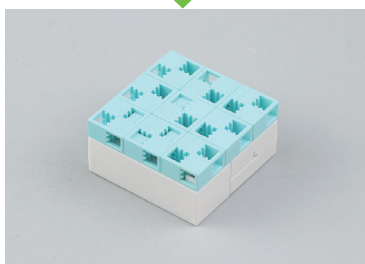
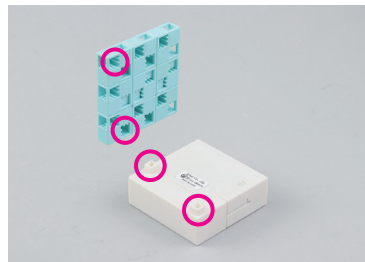
② ×2



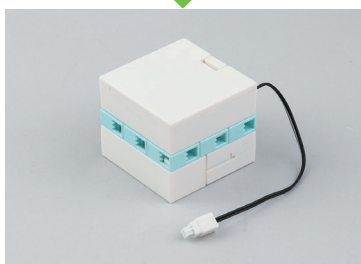
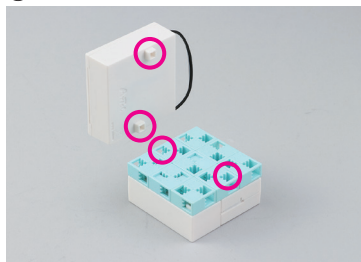
③



④



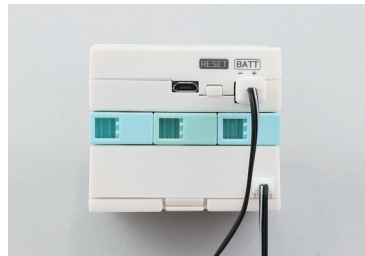
⑤



⑥



⑦

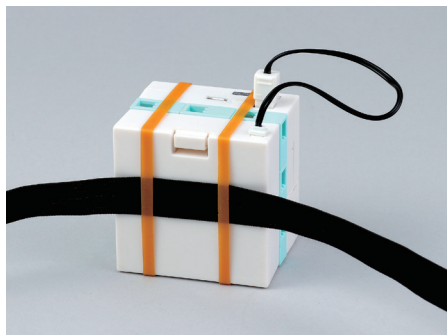
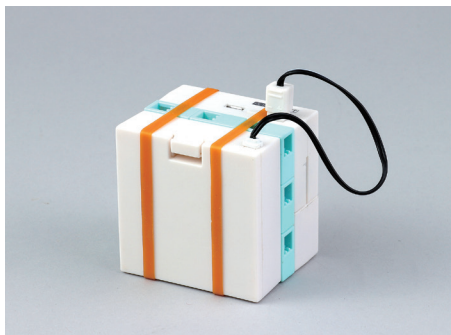


⑧



トレーニング補助装置の体への取り付け例

用意するもの：輪ゴム×2、ゴムバンドやひもなど



作例

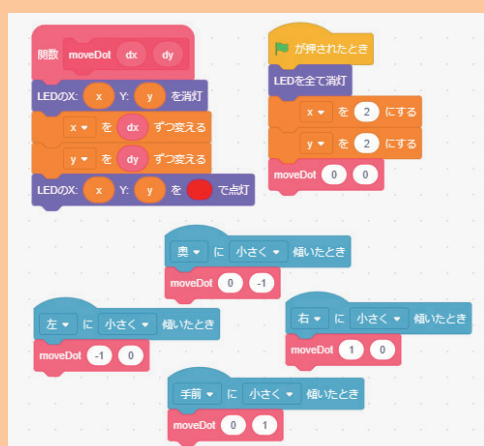
1. エクササイズを楽しみたい →ゲーミフィケーションの要素

プログラムについて

問題	自分の動作が客観的に把握できない。自分の動作が上達しているかわからない。
課題の設定	ゲーミフィケーションの要素を用いて、単調なことでも楽しめるようにする。 主観的に感じることを量で表せるようにする。
処理	身体の傾きや揺れが、どの程度理想的な状況かフィードバックさせる。 制限時間やしきい値を厳しくするなど難易度を上げるようにする。 無線通信を使用して友達と対戦できるようにする。

プログラム例（ロボットモード）

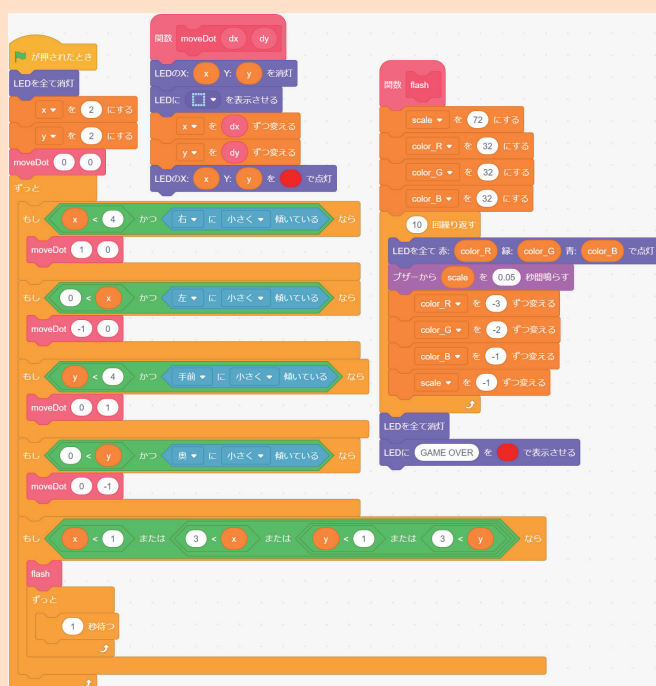
傾く毎にドットが移動



傾いている方向にドットが移動



大きく傾いたらゲームオーバーになる



- 大きく傾くことで、ドットが端に来たときに反応させる処理を末尾に追加しています。
- 繰り返しを使って変数の値を変化させ、音とLEDの色を変えています。教科書（開隆堂：p.215、東京書籍:p.202）の色のデジタル表現の原理を用いた工夫です。
- 例えば体幹を鍛えるプランクのエクササイズや両腕の水平伸ばしなどは、こうしたプログラムでStudio:bitを腕や腹筋に取り付けることで自分の身体の状態を把握することができます。
- ドットの移動が早すぎると体勢を変えるのに間に合わないため0.5秒待つようにしています（プログラムはPCに接続した状態とStudio:bitに転送して実行すると実行速度が変わることがあるので、実際に試して調整します）。

タイマー機能



制限時間や経過時間を実現するには、タイマーブロックを使用します。プログラム実行時に自動的にタイマーが進みますので、計測を始めるタイミングで「タイマーをリセット」して使います。

5秒耐えられたらクリアという目的で作成しています。クリアの表示例としてLEDを点灯させています。しかし、クリアしてもドットが移動するプログラムが動いているため、ゲームクリアにもかかわらずゲームオーバー画面が出るのが考えられます。それを回避するために、先のプログラムの一部分を下記のように変更します。



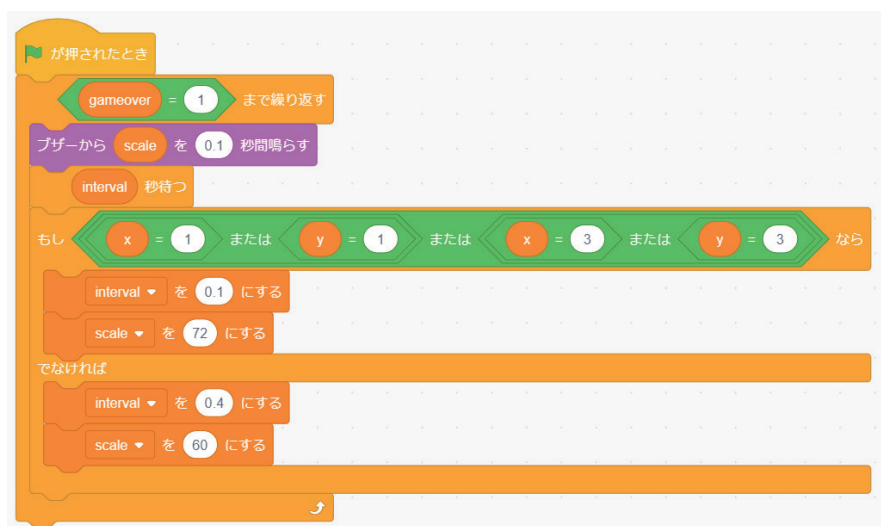
対戦機能

対戦をするためには、2つのアーテックロボ2.0を無線接続する必要があります。この場合は2台とも同じプログラムで構いません。同じ無線グループをしていて、左下図のようにゲームオーバーになったときに無線通信で「gameover」という文字列を送信するようにしておきます。先に「gameover」を受信した方が負け、ということにしています。



このプログラムをさらに以下のように改良することもできます。

先のサンプルではLED表示のみなので、身体の傾きや揺れを視覚的に把握することしかできませんでした。そのため、音のメディアを使用して把握できるようにするプログラムです。先のサンプルにこのプログラムを入ると、端に近づくと音の変化で把握できるようになります。



2. 色々なエクササイズをしたい → システムとして構成する

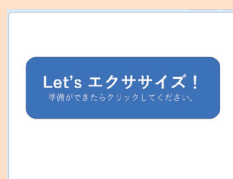
プログラムについて

処理

制限時間内に、所定のエクササイズメニューを終えられるようなチャレンジ。
 (エクササイズのメニュー毎に、負荷の大きさを変更できる。)
 (自分が何回やっているのか、回数が把握できるようにする。)
 (エクササイズで消費するカロリーの概算を表示する。)
 (制限時間は数字で出るのではなく、視覚的に時間経過と残り時間が把握できるようにする。)

プログラム例（キャラクターモード）

使用しているスプライトと画面設計



(1) スタート画面



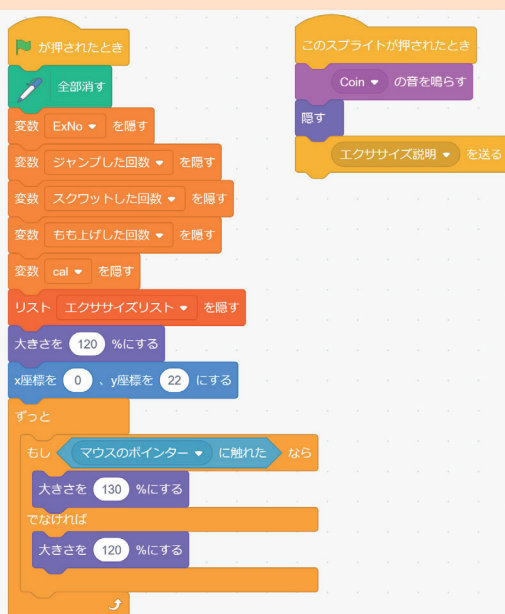
(2) 説明



(3) カウントダウン



(4) 実行画面

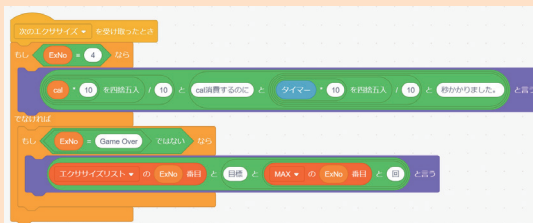
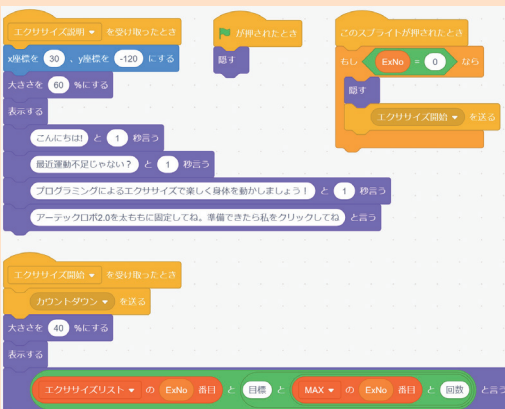


このスプライトはスタート画面を表示します。画面の表示設定を行っています。


ボタンであることを視覚的にわかりやすくするために、マウスポインタがスプライトに重なったら、押せることを示すフィードバックとして大きさを少し変化させるようにしています。

また、クリックされたことのフィードバックには、音を鳴らしています。

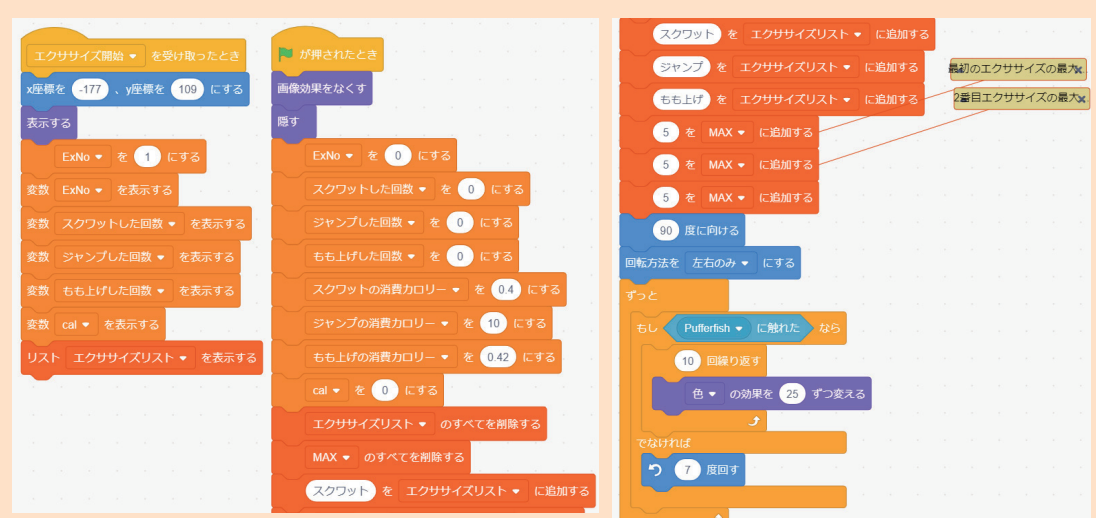
次の処理はエクササイズの説明なので、「エクササイズ説明」というメッセージを送り、「Abby」と「風船」に受け取らせます。




このスプライトでは、エクササイズをどのように行うのかという説明と、消費カロリーの計算結果、所要時間を表示しています。「ExNo」は現在何番目のエクササイズを行っているかを示しています。



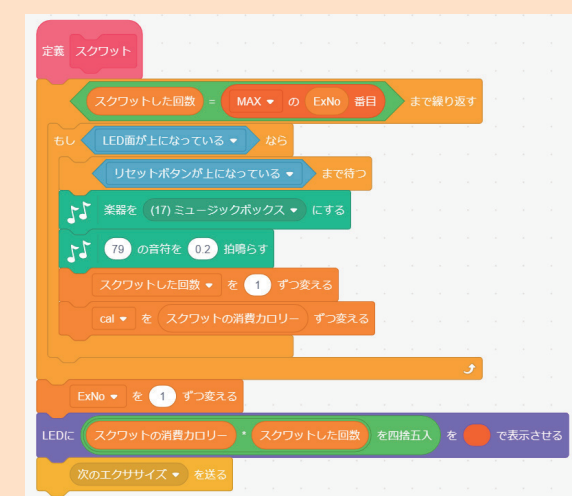
風船



風船はタイマー代わりに、変数やエクササイズメニュー、回数等の初期設定もしています。



ArtecRobo...



アーテックロボ2.0のSpriteは、本体のセンサーの状況を検知して、回数を増やしたり、正しくカウントされた時にそのことがエクササイズ中でもわかるように音を鳴らしたりしています。

左のプログラムは、スクワットの例です。

このプログラムでは、内容をわかりやすくするためにしきい値を用いず、状態を示すブロックを使用しています。

もし～なら、の中身を工夫して、LEDが上になってから、90度向きが変わったときに回数をカウントするようにしています。



Start!



カウントダウンはコスチュームを切り替えています。

このプログラムをさらに以下のように改良することもできます。

キャラクターモードでは、アーテックロボ2.0本体はコンピュータとつながっている必要があります。パソコンであればUSBケーブル接続が必要です。iPad版であれば無線でのキャラクターモード動作が可能です。パソコンで作成し、Google Drive上のフォルダにプログラムを保存して、それをiPadで読むとよいでしょう。

041776 K0821