

技術

分野

新学習指導要領
への第一歩

やってみよう プログラミング

PROGRAMMING

目次

プログラミング教育における技術分野の役割 …… ①

Scratch 編

- ① Scratch とは …… p.2
- ② Scratch の基本を知ろう …… p.4
- ③ Scratch のいろいろな操作方法 …… p.6
- ④ Scratch でメッセージを送ろう (制作例) …… p.8
- ⑤ Scratch で機器を動かそう (制作例) …… p.14

ドリトル編

- ① ドリトルとは …… p.18
- ② ドリトルの基本操作 …… p.20
- ③ ドリトルで通信しよう (制作例) …… p.24
- ④ ドリトルで機器を動かそう (制作例) …… p.28

JavaScript 編

- ① JavaScript とは …… p.32

本書で掲載された Scratch およびドリトルのプログラムのコード、および「Challenge」コーナーのプログラム例や実習の手立てなどを、開隆堂出版 Web サイトで公開しています。

(<http://www.kairyudo.co.jp/cgh33i> より「やってみよう プログラミング」を選択してください)

プログラミング教育における技術分野の役割

1 はじめに

今回改訂された学習指導要領総則では、情報活用能力が、言語能力及び問題発見・解決能力と並んで、学習の基盤となる資質・能力として例示され、各教科等の特質を生かした教科等横断的な視点から教育課程を編成し、その育成を図っていくことが求められている。そして、特に小学校では、この資質・能力を育成する具体的な方策として、情報手段を適切に活用した学習活動の充実を図ることとともに、「プログラミングを体験しながら、コンピュータに意図した処理を行わせるために必要な論理的思考力を身に付けるための学習活動」の計画的な実施が求められている。

一方、中学校の総則には、プログラミングに関する記述はない。これは技術・家庭科技術分野というプログラミングを学習内容とする教科があるためである。もちろん、中学校の他教科等でプログラミング教育を実施することも可能であるが、今回の改訂で重視されているプログラミング教育について、技術分野は専門に行う教科として位置づけられ、大きな責任を担っていることを認識することが大切である。

2 プログラミング教育の目標

「小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）」では、各学校段階におけるプログラミング教育の目標を以下のように想定している。

<知識及び技能>

・小学校：身近な生活でコンピュータが活用されていることや、問題の解決には必要な手順があることに気付くこと。

中学校：社会におけるコンピュータの役割や影響を理解するとともに、簡単なプログラムを作成できるようにすること。【A】

高等学校：コンピュータの働きを科学的に理解するとともに、実際の問題解決にコンピュータを活用できるようにすること。

<思考力・判断力・表現力等>

・発達の段階に即して、「プログラミング的思考」（自分が意図する一連の活動を実現するために、どのような動きの組合せが必要であり、一つ一つの動

きに対応した記号を、どのように組み合わせたらいいのか、記号の組合せをどのように改善していけば、より意図した活動に近づくのか、といったことを論理的に考えていく力）を育成すること。

【B】

<学びに向かう力・人間性等>

・発達の段階に即して、コンピュータの働きを、よりよい人生や社会づくりに生かそうとする態度を涵養すること。【C】

なお、小学校理科の学習指導要領では、以下のようなプログラミングの活動が例示されているが、このように各教科等でプログラミング教育を行うのは、各教科の目標の実現のためにこの活動が必要だからである。言い換えれば、教科等でプログラミング教育を行う際には、教科等の目標とプログラミング教育の目標の両方の達成を目指すということである。

小学校学習指導要領 理科

第3 指導計画の作成と内容の取扱い

2 第2の内容の取扱い

(2) <略>掲げるプログラミングを体験しながら論理的思考力を身に付けるための学習活動を行う場合には、児童の負担に配慮しつつ、例えば第2の各学年の内容の〔第6学年〕の「A物質・エネルギー」の(4)における電気の性質や働きを利用した道具があることを捉える学習など、与えた条件に応じて動作していることを考察し、更に条件を変えることにより、動作が変化することについて考える場面で取り扱うものとする。

3 技術分野におけるプログラミング教育

(1) 基本的な考え方

先に示した考え方によれば、技術分野におけるプログラミング教育は、技術分野の目標とともにプログラミング教育の目標の達成も目指すことが大切ということになる。しかし、技術分野の内容Dの目標を確認すると、(4)まで含めれば、中学校におけるプログラミング教育の目標は技術分野の中に全て含まれることがわかる。参考に、内容Dの各項目で目指す資質・能力一覧と、プログラミング教育の目標の関係を表に示す。【 】で示された部

■内容D 情報の技術 資質・能力一覧とプログラミング教育の関係

	知識及び技能	思考力, 判断力, 表現力等	学びに向かう力, 人間性等
(1)	・情報の表現, 記録, 計算, 通信などについての科学的な原理・法則の理解 ・情報のデジタル化や処理の自動化, システム化, 情報セキュリティなどに関わる基礎的な技術の仕組みの理解	・情報の技術に込められた工夫を読み取る力 ・情報の技術の見方や考え方の気付き	・進んで情報の技術と関わり, 主体的に理解し, 技能を身に付けようとする態度
(2)	・情報通信ネットワークの構成と, 情報を利用するための基本的な仕組みの理解 ・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技能 【A】	・情報の技術の見方・考え方を働かせて, 問題を見いだして課題を設定し解決できる力 【B】	・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 ・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
(3)	・計測・制御システムの仕組みの理解 ・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技能 【A】	・情報の技術の見方・考え方を働かせて, 問題を見いだして課題を設定し解決できる力 【B】	・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 ・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
(4)	・生活や社会に果たす役割や影響に基づいた情報の技術の概念の理解 【A】	・よりよい生活や持続可能な社会の構築に向けて, 情報の技術を評価し, 適切に選択, 管理・運用したり, 新たな発想に基づいて改良, 応用したりする力	・よりよい生活や持続可能な社会の構築に向けて, 情報の技術を工夫し創造していこうとする態度 【C】

分が, プログラミング教育の目標と関係する部分である。

ただし, 先に示した理科におけるプログラミング教育はあくまでも例示であることや, 他教科等でもこの教育が行われる可能性があることから, 小学校で使用したプログラミング言語の種類や, 他教科におけるプログラミングの経験等も含めた生徒の実態を踏まえて技術分野におけるプログラミング教育を計画することが大切である。

(2) 今回の学習指導要領における変更点

今回の学習指導要領では, 従前の「プログラムによる計測・制御」が「計測・制御のプログラムによる問題の解決」と変更されている。これは, 技術によって問題を解決することを強調しているためである。

また, これ以外にも, 小学校における学習を想定し, 技術の専門性を明確に示すために, 以下のような変更をしていることを確認しておくことが大切である。

- 従来から指導している機器等に組み込まれている「計測・制御」に関するプログラミングに加えて, コンピュータやスマホ等の画面等で利用されている「コンテンツ」を新たに取り上げる。具体的には, 従前の「デジタル作品の設計・制作」ではソフトウェアを用いる例が多かったが, これをプログラミングにより学ぶことに改める。
- 小学校の学習を発展させる内容とする。具体的には, コンテンツに関しては, 小学校では音, 画像など1種類の入力情報に対し, 片方向の定められた動きをくりかえす場合が多いと考えられるが, 技術分野においては, 複数の情報を扱い, 使用者の働きかけ(入力)に

よって異なる応答(出力)を返す仕組みをもち, さらに, コンピュータ間の情報通信を処理の一部に含むプログラミングを扱う。

- 計測・制御に関しては, 小学校では, 既存の計測・制御のシステムを動作させる場合が多いと考えられるが, 技術分野では, 問題を解決するために, どのようなセンサーやアクチュエータが必要か, それをどのように組み合わせるかといった計測・制御システムの構想も学ぶこととする。

4 おわりに

現行学習指導要領では, 課題の解決策を構想する際に「フローチャート」を例示していた。今回はこの部分を「アクティビティ図のような統一モデリング言語」に変更している。これは, 解決すべき問題によっては「並列処理」や「割り込み」といったフローチャートでは表現しにくい処理が必要となる可能性があること。加えて, このようなことを容易に実現できるプログラミング言語も普及してきているためである。

このようなことも含めて, 技術分野だからこそできる専門性のあるプログラミング教育を行うためには, 教師が新たな内容について指導するために必要な知識等を身に付けることが必要であり, 最近のさまざまなプログラミング言語がどのような機能をもち, 操作性がどのようにかわっているのかを知ることがその第一歩である。本冊子が, 皆様のこれからのプログラミング教育実践に少しでも役立つことを期待している。

(上野 耕史 文部科学省初等中等教育局教育課程課教科調査官)

1 Scratchとは

1 Scratch について

Scratch は、MIT（マサチューセッツ工科大学）が開発したビジュアル型のプログラミング言語です。ブロックを用いたビジュアル型のプログラミング言語が、文字入力を中心とした言語よりも優れている点として、主に以下の2点が挙げられます。

- プログラムで使用できる命令がブロックとして画面の中に全て表示されているので、何ができるのか画面を見るだけで把握できる。
- ブロック同士はマウス操作でつなぎ、文法的に間違っている場合はブロックをつなげることができないので、プログラミング言語の文法が分からなくてもその形状で何がつながるのか把握できる。

そのため生徒たちは、自分がプログラムで表現したい・実現したいことを考えることに専念できます。

Scratch は子ども向け言語というわけではなく、複雑で高度なプログラムも作ることができます。2018年1月現在、全世界で2357万人がコミュニティサイトにユーザ登録している人気の言語です。またScratchには、プログラムを作るときに必要なサンプル画像や効果音などが最初から豊富に用意されている上に、自分で画像を描く、編集する、カメラで撮影する、音を録音する機能があるので、メディアを統合的に扱いやすくなっています。

2 Scratch 1.4 を準備しよう

Scratchには、いくつかのバージョンがあります。バージョンによって得意なことが違い（表1）、バージョンが新しい方が優れているということではありません。ネットワークを利用した双方向性のあるコンテンツを作成するにはScratch 1.4というバージョンを使います。Scratch 1.4は

https://scratch.mit.edu/scratch_1.4/

にアクセスし、Windowsであれば図1のリンクからダウンロードします。解凍すればコンピュータにインストールしなくても使えます。

ScratchのWebページが日本語でない場合は、ページ下部にあるドロップダウンリストから「日本語」を選択します。



図1 Scratch 1.4 のダウンロード

表1 Scratchのバージョンの主な違い

機能	Scratch 1.4	Scratch 2.0
作成したプログラム同士でネットワーク通信する機能（Meshと言います）	○	×
ダウンロードせずWebだけでプログラムを作成すること	×	○*
音の録音や編集	○	○
画像作成や編集	○	○

* 2018年1月現在はFlashが必要ですが、Flash無しでも動くScratch 3.0が公開される予定です。

3 Scratch 1.4 の画面構成と主な機能

Scratch でプログラムをするキャラクター等はスプライトと呼ばれます。起動直後はサンプルとしてネコのスプライトが表示されます。Scratch では、必要なスプライトを用意し、各スプライトをどのように動作させるか、ブロックを組み合わせることでプログラミングを行います。図2では、「④スプライトリスト」からネコが選択されています。ここで選択されたスプライトに対して、「カテゴリー」で選ばれた「①ブロックパレット」から「②スクリプトエリア」にブロックをドラッグ&ドロップしてスクリプトを作成している様子です。プログラムの実行結果は、「③ステージ」で見ることができます。「メニューバー」では、ファイルの保存、読み込み、スプライトの拡大縮小、全画面表示などを行うことができます。



図2 Scratch 1.4 の画面構成

Scratch 1.4 の画面構成

①ブロックパレット

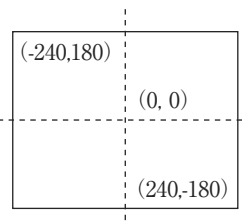
10 歩動かす や **90 度に向ける** など、Scratch に用意されている命令をブロックと呼びます。ブロックはさまざまなカテゴリーに分類されています。

②スクリプトエリア

プログラムを編集する領域です。上部には「スクリプト」「コスチューム」「音」の3つのタブがあります。

③ステージ

プログラムの動作を確認する領域です。ステージの広さは480ピクセル×360ピクセルで、右図のような座標です。



④スプライトリスト

使用するスプライトの追加や削除ができます。

2 Scratchの基本を知ろう

1 スプライトを動かしてみよう

Scratch でプログラミングを行うための基本的な操作や用語について学んでいきましょう。

1.1 Scratch の基本操作

① スクリプト

Scratch ではプログラムのことを「スクリプト（英語で「脚本」という意味）」と呼びます。脚本の通りに「ステージ（舞台）」上で「スプライト（演者）」が動きます。また、スプライトの見た目は「コスチューム（衣装）」で指定するといったように、Scratch は演劇にたとえられます。スクリプトは、ブロックを積み重ねて作ります。

② ブロックのつなげ方

元々あるブロックの下に、つなげたいブロックを白く光るところまでドラッグ&ドロップします。

③ ブロックのはずし方

1段目のブロックから2段目のブロックをはずしたときは、2段目のブロックをドラッグします。そのとき、2段目から下のブロックも一緒にはずれます。

④ ブロックの消し方

消したい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。また、消したいブロックの上で右クリックをすることで表示されるメニューの中にある「削除」を選択して消すこともできます。

⑤ スクリプトの開始の合図

🚩をクリックすることで、「🚩がクリックされたとき」のブロックがついたスクリプトが動き出します。

⑥ スクリプトを止める合図

●をクリックすることで、実行中のすべてのスクリプトが止まります。

⑦ 「ずっと」のブロック

「ずっと」のブロックは、ブロックの間に挟まれたスクリプトを永久に繰り返し行うためのものです。

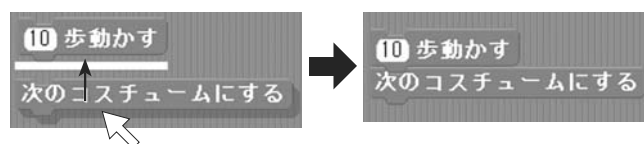
Scratch には、回数を設定できるものなど、いくつかのブロックが用意されています。

①

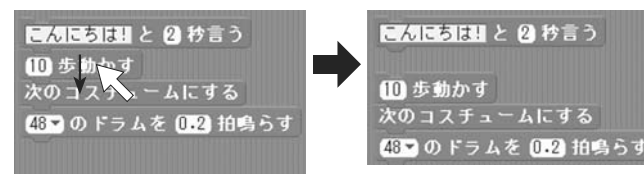


「スクリプト」と呼ぶ

② ブロックのつなげ方



③ ブロックのはずし方

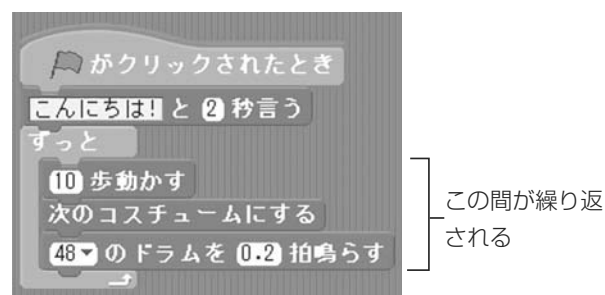


④ ブロックの消し方



ブロックパレットへドラッグ&ドロップする

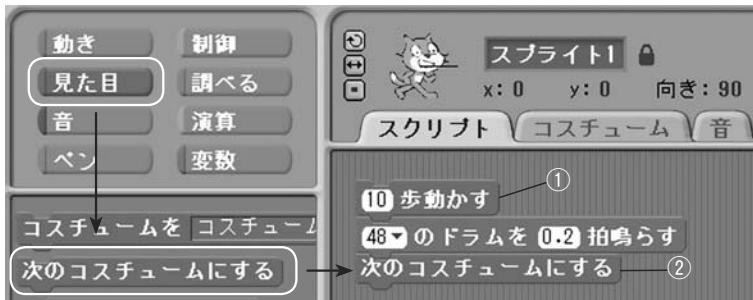
⑦



この間が繰り返される

1.2 Scratchでプログラミングしてみよう（スプライトをダンスさせる）

①, ②



① 「動き」のブロックパレットにある「10歩動かす」のブロックをスクリプトエリアに移動させます。

② ①の手順と同様に「音」「見た目」のブロックパレットから、ブロックを移動させつなげます。

※ブロックパレットとブロックの色を参照すると見つけやすくなります。

③



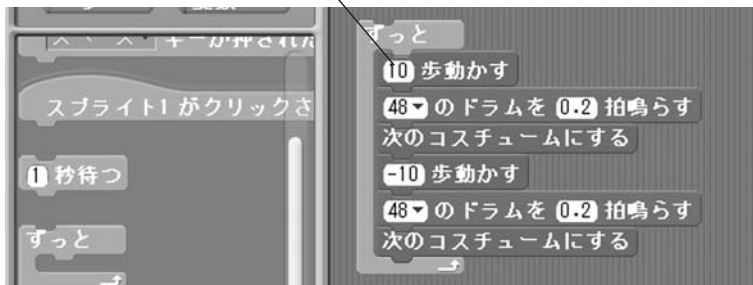
「-10」に書き換える

③ 同様の操作でスクリプトの続きを作ります。「10歩動かす」のブロックの数字をクリックし、半角で「-10」と入力します。


④ 「制御」のブロックパレットから、「ずっと」のブロックを移動し、③で作成したスクリプトの全体を挟むように追加します。



⑤ 「見た目」のブロックパレットにある「こんにちは！と2秒言う」のブロックを、スクリプトエリアに置いた「ずっと」の上につなげます。

④

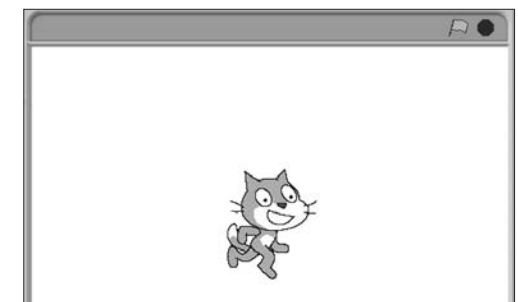
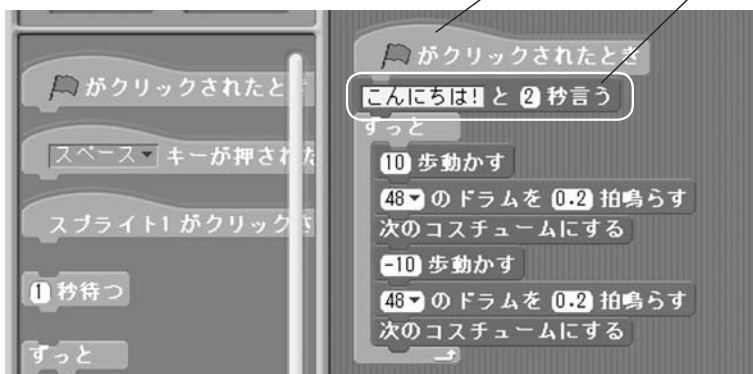


一番上のブロックに合わせると全体が挟まれる

⑥ 「制御」の中の「がクリックされたとき」のブロックをつなげます。

◇ をクリックしてプログラムが動作するか確認します。このプログラムはずっと動き続けます。止めたいときは、をクリックします。

⑤, ⑥



Challenge

新しい音を読み込んだり、録音したり、新しいコスチュームにカメラを使ってコマ撮りしたりすることで、マルチメディアの学習を考えてみましょう。

3 Scratchのいろいろな操作方法

1 変数と条件分岐を使う

ネコをクリックすると、クリックされた総回数をネコがセリフとして言い、スペースキーを押すと回数がリセットされて0に戻るプログラムを作りましょう。

ここでは、クリックされた回数を表示する簡単なプログラムをつくります。クリックされると回数が1つずつ増える単純なものですが、コンピュータは前の数字を記憶させないと、1を足した次の数字を求めることができません。こうした一時的に情報を記憶させておくもののことを「変数」と言います。



- ① 最初に、スプライトリストにあるネコをクリックして選択します。

変数を作るには、オレンジの「変数」カテゴリから「新しい変数を作る」を押します。変数は、いくつでも作ることができますが、名前をつけて識別できるようにします。今回は「kaisu」にしましょう。

通常は「すべてのスプライト用」として作成します。

①



- ② 次に、目的の動きができるようにスクリプトを作っていきます。

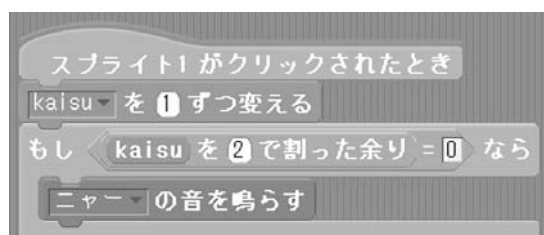
紫の「～と言う」というブロックの中に「kaisu」と名づけられた丸いオレンジの変数ブロックを入れることで、変数の値を言わせることができます。

変数は、プログラムを作る上ですぐに必要となる概念です。Scratchでは、変数に日本語で名前をつけることもできますが、後で説明するMeshでネットワークを利用する場合に、日本語が正しく表示されませんので、半角の英数字で名前をつけるようにしましょう。

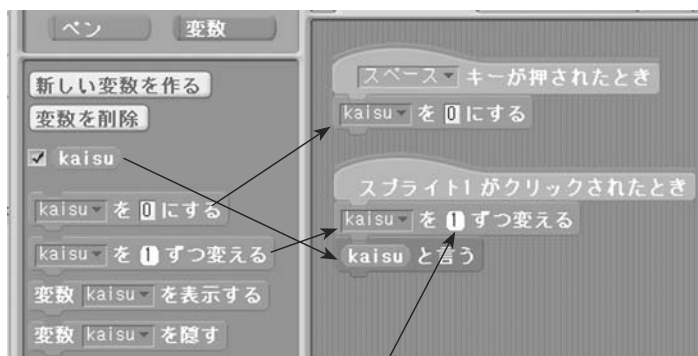
Challenge

偶数の時にネコが鳴くようにしましょう。下図のように、「～もし～なら」ブロックの中に3つのブロックを入れて条件を指定します。

もしこのプログラムが正しく動作しない場合は、Scratch 1.4のダウンロード方法を再度確認してみましょう。



②



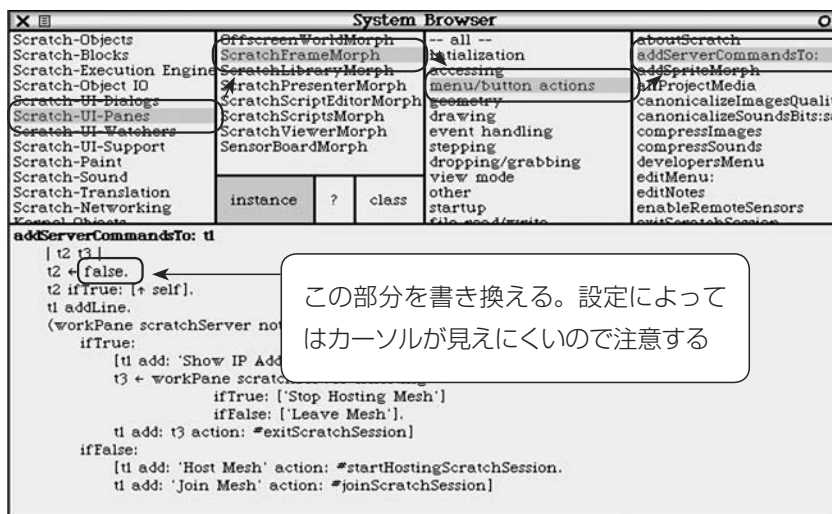
「～を～ずつ変える」というブロックは、もともと「1」と書かれています。Scratchでは、白い窓の部分には、自分で書き換えや、変数や計算するブロックを入れることができます。

2 Mesh でネットワークを利用した通信をしよう

Scratch 1.4 には、さまざまな拡張機能があります。新学習指導要領に対応した授業をするために、Scratch の Mesh という機能を使えるようにしましょう。これは最初に一度設定するだけでよく、以下の手順で一度作成した「Scratch.image」のファイルを、他の PC のものの上書きするだけで、Mesh が使えるようになります。Mesh を使うと、文字を送信したり、他のコンピュータのスプライトを動かしたり、対戦したりするような、ネットワークを利用した双方向性のあるプログラムを簡単に作成することができます。

Mesh を使えるようにすると、コンピュータ室などのネットワークに接続されている複数のコンピュータ同士で、変数の共有や、「～を送る」ブロックを使用して、離れているコンピュータへの指示などができます。

④



※ Mesh を使ってプログラムを作るときは、設定を終えた Scratch で、Mesh ネットワークに参加させます。情報をやりとりしたい生徒たち（クラス全員でも構いませんし、班ごとでも構いません）でグループを作ったら、その中の 1 名は Scratch で、シフトキーを押しながら、メニューバーの「共有」をクリックします。すると下図のメニューが出るので、「Host Mesh」を選びます。



すると画面に IP アドレスが表示されるのでメモしておきます。次に、他の Scratch を使っているグループメンバーは、自分の PC の Scratch で同様にシフトキーを押しながら「共有」をクリックし、今度は「Join Mesh」を選び、先ほどメモした IP アドレスを入力します。そうすると、それらの Scratch 同士でネットワークを利用して情報交換できるようになります。

① Scratch の左上にある、Scratch のロゴの R の文字をシフトキーを押しながらクリックします。

② 「turn fill screen off」を選びます。

③ Scratch ウィンドウの下部と右側に表れた灰色の部分をクリックしてメニューを表示させ、「open...」→「browser」を選び左図のシステムブラウザを表示させます。

④ システムブラウザの左側から順に「Scratch-UI-Panes」→「ScratchFrameMorph」→「menu/button actions」→「addServerCommandsTo:」と順にクリックし、「t2 ← true.」の箇所を「t2 ← false.」に変更します。「. (ピリオド)」は消さないようにしましょう。

⑤ システムブラウザ上で右クリック → 「accept」を選び、開いたウィンドウにイニシャルを入れて、「accept」を押します（イニシャルは、編集の識別として入れるだけです）。

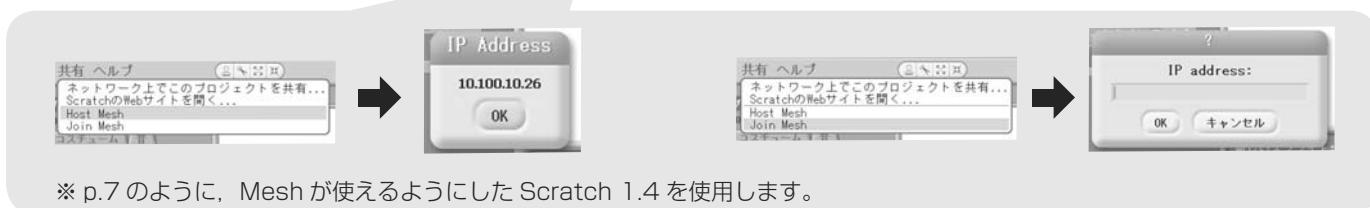
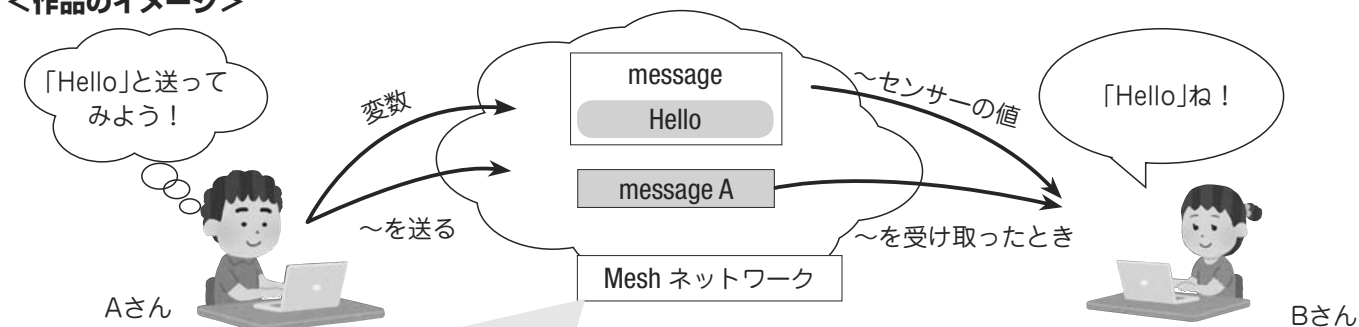
⑥ 左上の × を押し、システムブラウザを閉じて、再度 Scratch のロゴの R の文字を、シフトキーを押しながらクリックし、メニューから「save image for end-user」→ 「Yes」を選ぶと、Mesh が使える設定になった「Scratch.image」が作成されます。

4 Scratchでメッセージを送ろう

1 ScratchのMesh機能を活用して、メッセージを送ってみよう！

ScratchのMesh機能を活用すると、同じネットワークに接続されたPCで実行されているScratch同士で通信を行うことができます。ここでは、AさんとBさんで変数を共有し、Aさんが作ったスクリプトからBさんが作ったスクリプトに対してメッセージを送る作品を作ってみましょう。

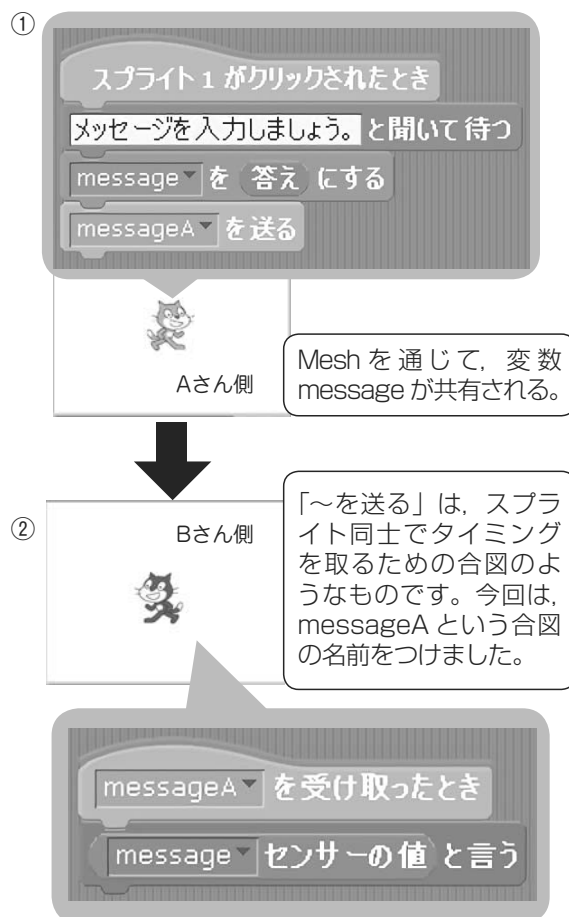
<作品のイメージ>



Point

- 現状、Meshで送ることができる文字列は半角英数字のみです。
- Aから送信されたことを把握するために、「送る」と「受け取る」を使用します。共有された変数と違って、こちらは自分で入力しなければなりません。打ち間違えないように注意しましょう。

- ① 送信者はスプライト1（今回はネコ）のキャラクターをクリックすることで、任意の文字列を入力できます。入力された文字列は、「答え」という変数に入れます。「答え」という変数は、Meshで内容を共有することのできない特別なものなので、別の変数（今回はAからのメッセージという意味で、messageAとつけました）を作成します。
- ② 共有された変数の値を取得するには、「～センサーの値」を使います。～の部分には、Mesh上で共有されている変数名が表示されます。



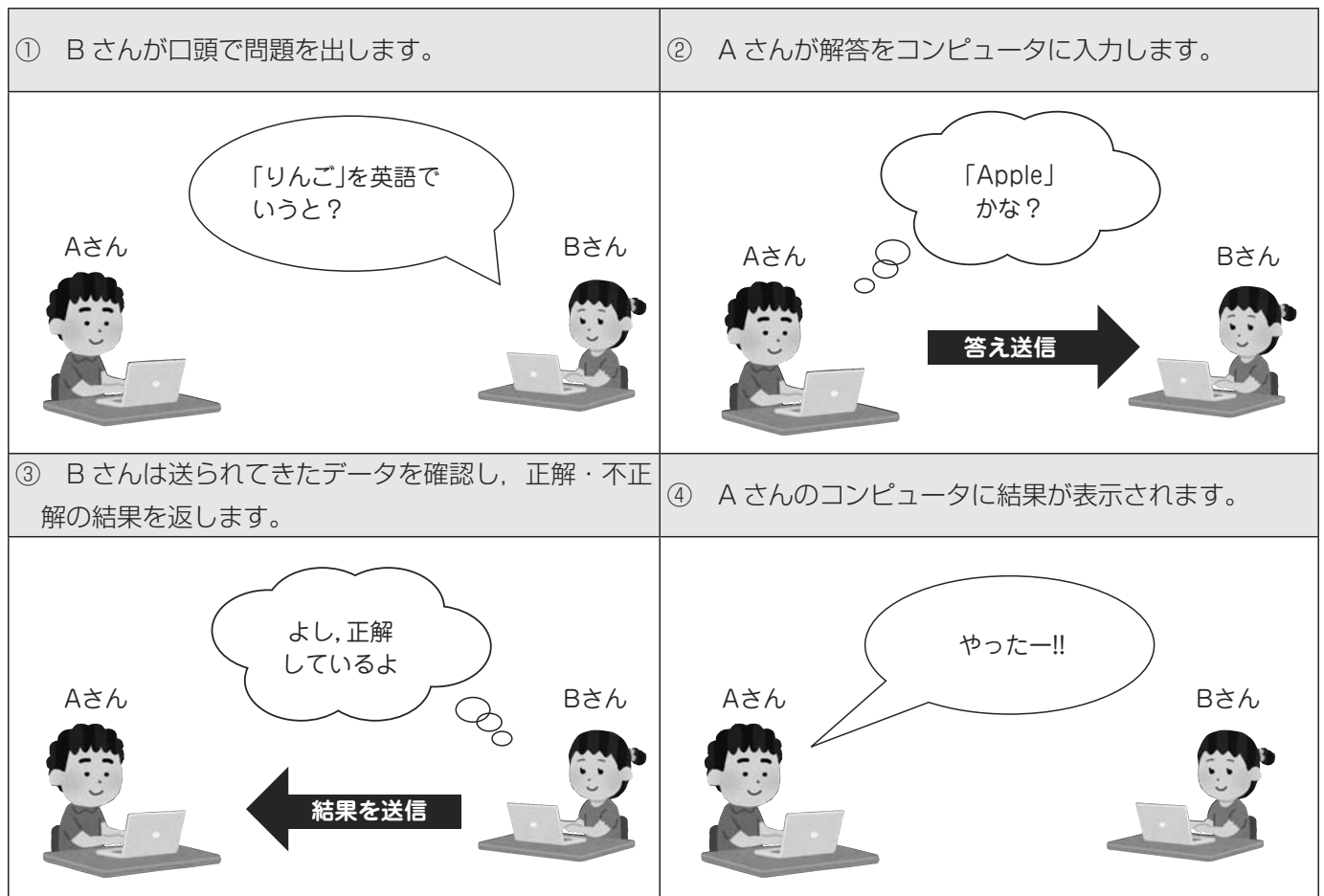
Challenge

- ・ 送信側と受信側が交代をして、どちらのスクリプトも作成し、より理解を深めてみましょう。
- ・ 現在の作品では送信側から受信側への一方的な通信となっています。「双方向」にメッセージをやり取りするにはどうしたらよいか？（p.9から解説しています）

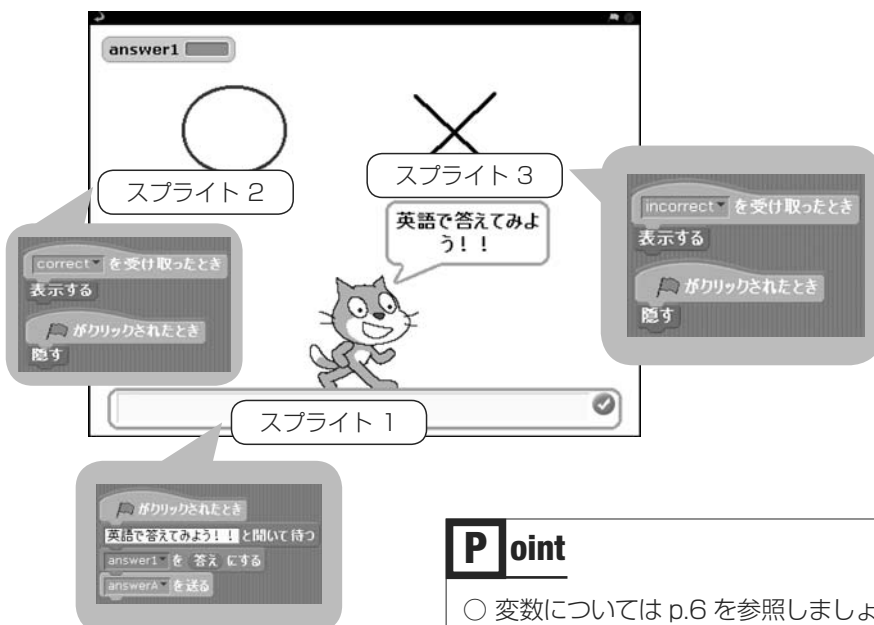
2 Scratch の Mesh 機能を活用して、クイズを作ってみよう！

p.8 の作品をさらに発展させて、英単語クイズを作ってみましょう。ペアで問題を出題し、解答者と、解答に正解・不正解のフィードバックを返す人の作品を作ってみましょう。

<作品のイメージ>



① Aさんの画面イメージとプログラム



① スプライト1では、スクリプトの実行者が任意の文字列を入力し、変数(ここでは「answer1」)に格納し、answer1の内容が変更されたというメッセージをBさんのScratchへ送ります。

② スプライト2と3は<作品のイメージ>の④でAさんのコンピュータの画面に正解か不正解かを表示するためのものです。スクリプトを実行し、①～③までは表示せず、④で「○」か「×」のどちらかを表示します。

Point

- 変数については p.6 を参照しましょう。
- 正解・不正解の表示はデジタル画像の編集として生徒に工夫させることが考えられます。また、画像で表示するだけでなく、音も鳴らす等の工夫も考えられます。

③ Aさんが入力した文字列を2秒間表示し、その後正解であれば「y」、不正解であれば「n」の入力を促すスクリプトです（ここでは仮に「y」、「n」としていますが、別の英数字を使うことも可能です）。

④ 条件分岐である「もし～なら」を使って、正解・不正解によってAさんに返す結果を変えています。

Point

○「answer1 センサーの値」というブロックは、最初「スライダーの値」と表示されています。「▼」をクリックして、自分で変更しましょう。

○条件分岐についてはp.6を参照しましょう。

③ Bさんの画面イメージとスクリプト

2秒後

```

answerA を受け取ったとき
answer1 センサーの値と 2 秒言う
正解の時はyを、間違っているときはnのキーを押してください!! と聞いて待つ
もし 答え = y なら
  correct を送る
もし 答え = n なら
  incorrect を送る
  
```

④

⑤ Bさんの入力結果によって、Aさんの画面表示が変わります。

正解の場合は「correct」
不正解の場合は「incorrect」

という文字列でスプライト2、スプライト3のどちらを表示するかを決めています。（p.9のスプライト2と3のプログラムを参照。）

⑤ Aさんの画面イメージ

正解時

不正解時

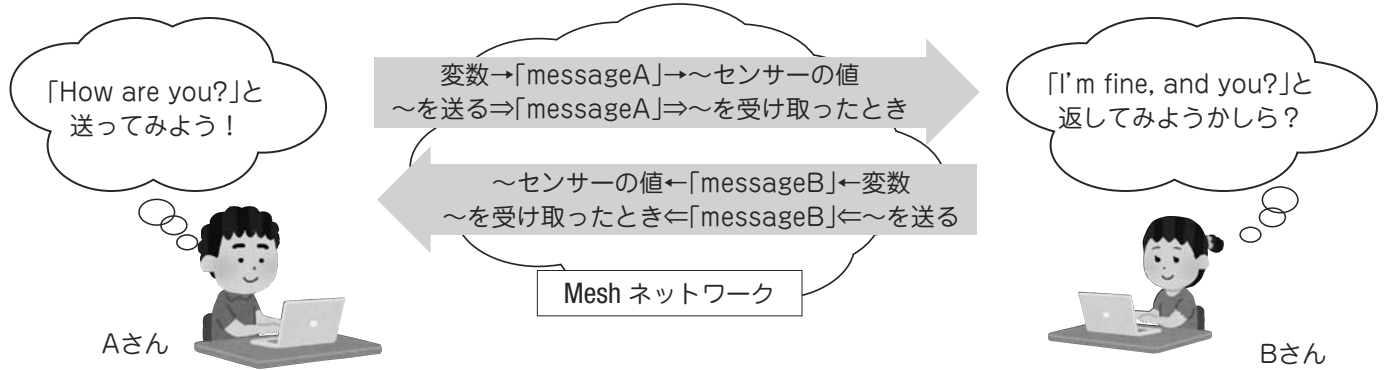
Challenge

- ・繰り返しスクリプトを実行することを前提に、正解した数をカウントして表示する等の工夫をしてみよう。
- ・現在はBさんが口頭で問題を出していますが、Bさんの問題もコンピュータで入力する作品にするにはどうしたらよいか？（p.11から解説しています）

3 双方向性のチャットを作ってみよう！（発展）

p.9～10の作品をさらに発展させてみましょう。前はBさんが口頭で問題を出していましたが、ここではBさんからのメッセージとし、口頭ではなくコンピュータで通信をして、AさんとBさんで何度もメッセージのやり取りができるような作品を作ってみましょう。

<プログラムのイメージ>



① Aさんの画面イメージとプログラム



① Aさんの画面では、ネコのスプライトをクリックすると、メッセージを入力することができます。

Point

○ 変数については p.6 を参照しましょう。また、現在は をクリックすることでスクリプトが始まりますが、 をクリックの代わりにキーボードのキーを割り当てるなど、使いやすさを考えて工夫させてみましょう。

② Bさんの画面イメージとプログラム



② Bさんの画面では、Aさんから送られてきたメッセージが表示され、チョウのスプライトをクリックすることで、返事を入力することができます。

※スプライトは、新たに作成するたびに「スプライト●」の●の部分の数字が増えた名前が自動的につけられます。スプライトが増えてくると識別しにくくなるので、分かりやすい名前をつけるとよいでしょう。今回は、「butterfly」と「cat」という名前をつけています。

Point

○ メッセージはAさんとBさんのどちらからでも送ることができます。

Point

○ 作品が完成して、お互いに会話が可能になったら、送信するメッセージが相手にとって適切なメッセージかどうか、情報モラルでの指導事項と組み合わせて考えさせてみましょう。

③ Aさんの画面に、Bさんから送られてきた文字が表示されます。その後、会話を続ける場合はネコのスプライトをクリックします。

③ Aさんの画面イメージ

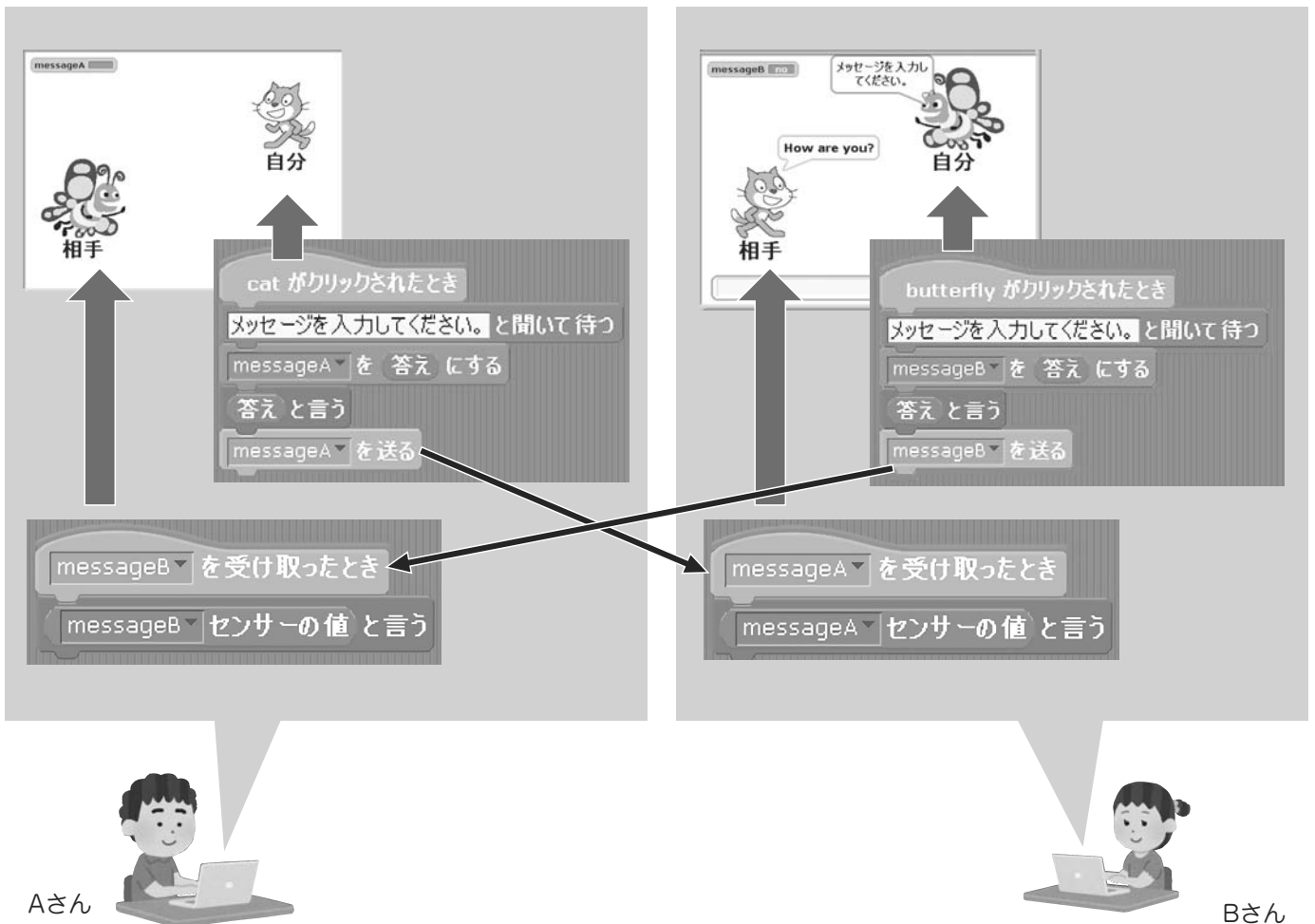


Challenge

- ・会話の履歴が残るにはどうしたらよいか？
- ・メッセージが送られてきたことを相手に確実に通知するにはどうしたらよいか？
- ・p9～10のクイズの作品と組み合わせて、お互いに問題を出し合い、結果を返す作品を作ってみよう！

4 双方向性のチャットのまとめ

今までの作品をまとめて、各画面のスプライトのスク립トが、他のスク립トとどう関係しているのかを図示しながら授業を行うことでより理解が深まることが考えられます。しかし、この作品では、最低限の機能だけなので不自由なことがたくさんあります。メッセージが届いたことをより確実に知らせる方法や、チャットをやり過ぎないように使用時間の制限をかける機能や、安全性を高めるため使用時にパスワードを入れるようにするなど、メディアを複合させるようにしてさらに工夫を加えてみましょう。

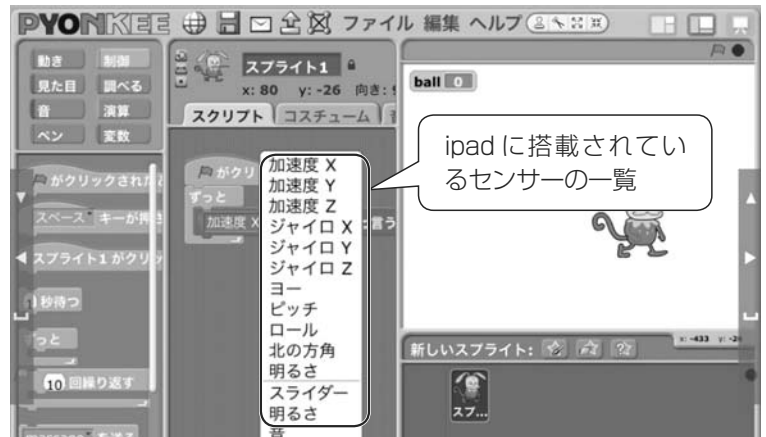


5 その他の活用例

5.1 Pyonkee (ピョンキー)

これまで説明した作品は、iPad 向けに開発されている「PYONKEE」というアプリでも実行できます。名前や登場するスプライトは多少異なりますが、スクリプトの作り方やブロックの種類など、Scratch 1.4 との共通点も多く、同じように操作できます。また、設定を変更することなく「Mesh」を用いた通信も可能です。

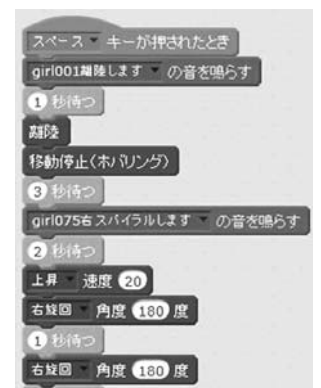
Pyonkee を使えば、iPad に搭載されているさまざまなセンサー（照度センサーや、加速度センサーなど）の値を取得（計測）して利用することができます。これらのセンサーの値を「Mesh」の機能を用いて通信することで、PC だけではできなかったさまざまな計測を用いた双方向性のコンテンツを簡単に制作することができます。



5.2 Scratch でドローンの制御

Scratch は直感的に扱いやすく、操作性が優れています。そのような長を活かして、Scratch でさまざまな連携ができるようになってきました。右図は拡張した Scratch でドローンを飛ばすスクリプトの一部です。

出典：「スクラッチ言語拡張によるミニドローン・プログラミング飛行」 車輪の再発見
<http://www.rediscovery.co.jp/?p=1125>



5.3 Scratch でのプログラミング経験を他の言語に活かす

Scratch のようなビジュアル型のプログラミング言語は、他にもさまざまな種類があります。例えば、MOONBlock（ムーンブロック）（図1）、やプログラミン（図2）、ビケットなどが挙げられます。また App Inventor を使えば、ブロックをつなげることで動く Android アプリの制作も可能です。

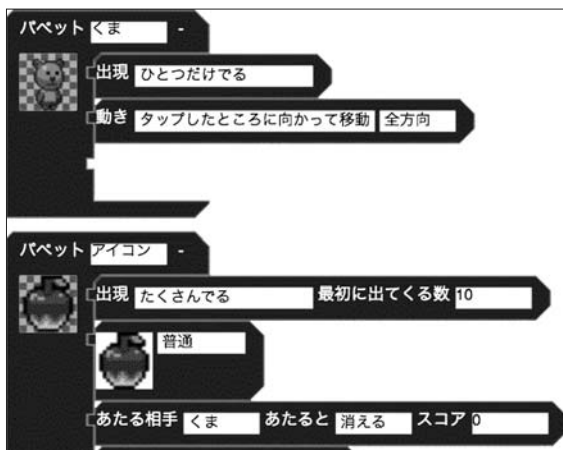


図1 MOONBlock の例



図2 プログラミンの例


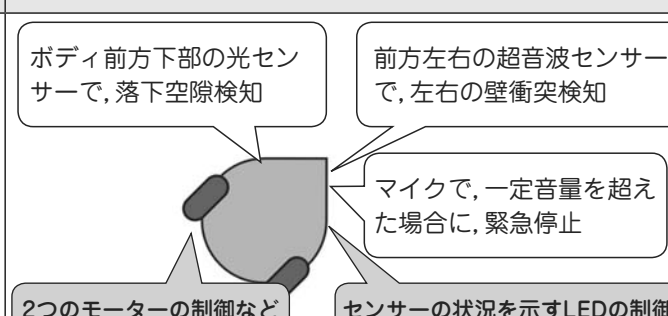
出典：「MOONBlock とは」 秋葉原リサーチセンター <http://www.moonblock.jp/docs/>

「子供向けウェブサイト『プログラミン』について」 文部科学省 http://www.mext.go.jp/b_menu/houdou/25/10/attach/1340677.htm

5 Scratchで機器を動かそう

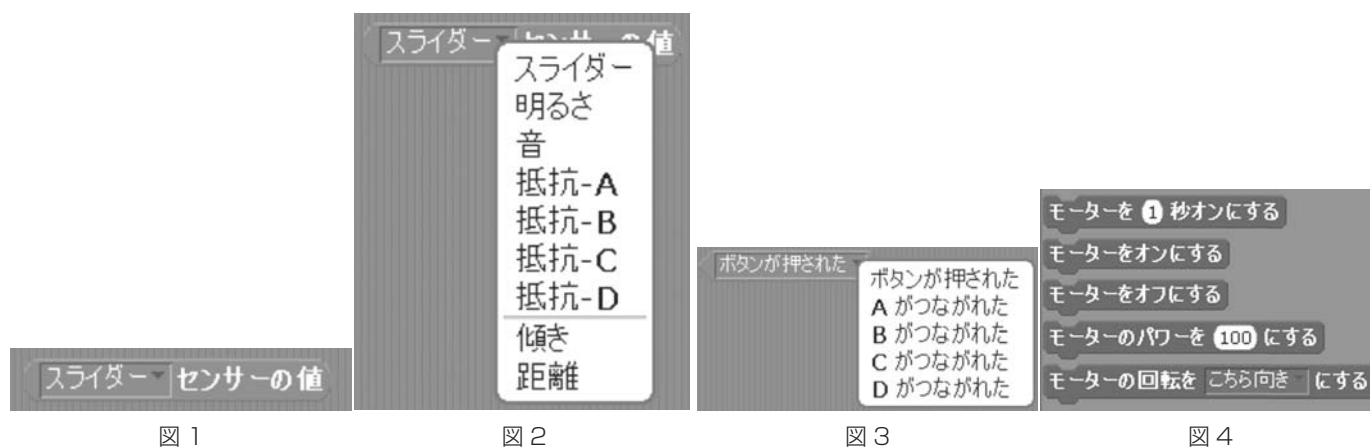
1 システムを構想しよう

新学習指導要領の計測・制御では、「システムを構想」という点が大きく変わりました。IT用語辞典バイナリでは、システムとは「複数の要素が体系的に構成され、相互に影響しながら、全体として一定の機能を果たす何物かのこと」と書かれています。つまり、要素が複数あること、それらが相互に・有機的に影響しながら、目的とする機能を実現して問題を解決することが求められていると言えます。従来の計測と制御では、単一のセンサーと単一のアクチュエータでよかったです。システムを構想するためには、複数のセンサーや複数のアクチュエータなどを扱って問題を解決する必要があります。

これまでのイメージ（自動走行ロボを例に）	これからのイメージ（自動走行ロボを例に）
目的地まで自動走行する	目的地まで安心・安全に自動走行する
 <p>前方左右の超音波センサーで、左右の壁衝突検知</p> <p>2つのモーターの制御など</p>	 <p>ボディ前方下部の光センサーで、落下空隙検知</p> <p>前方左右の超音波センサーで、左右の壁衝突検知</p> <p>マイクで、一定音量を超えた場合に、緊急停止</p> <p>2つのモーターの制御など</p> <p>センサーの状況を示すLEDの制御</p>

Scratch は汎用的なプログラミング言語なので、多くの計測・制御できるハードウェアが対応しています。Scratch を拡張して独自のブロックが用意されているものもありますが、Scratch には、予め図1～4のような命令が使えるようになっています。Mesh ネットワークを使わないのであれば、Scratch 2.0でも使用できます。また、Scratch と同じ操作感でプログラミングができる Studuino を使用してもよいでしょう。

例えば、「～センサーの値」ブロック（図1）では、図2や図3のようなセンシングが可能です。またモーターの制御であれば図4のようなブロックを使うことができます。

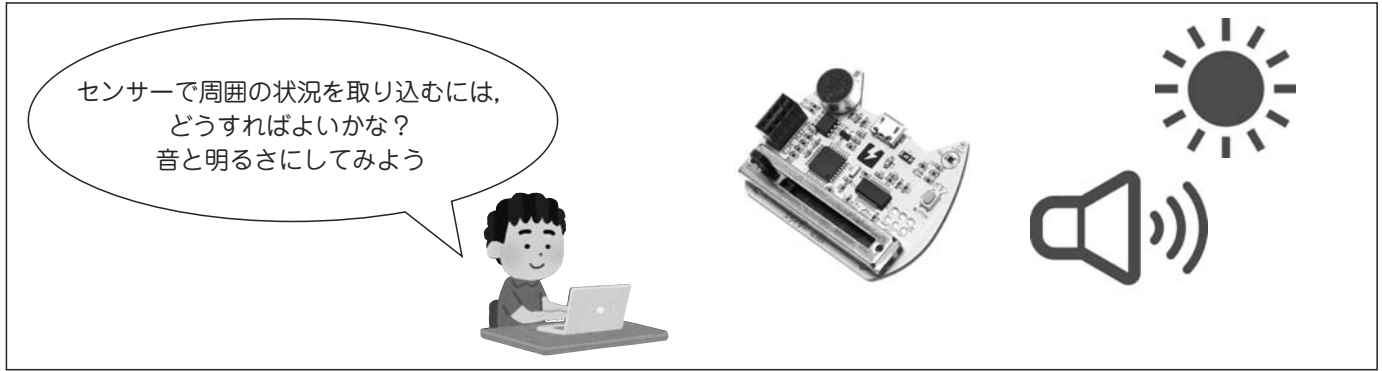


Scratch とすぐに接続して計測・制御が可能なハードウェア例（詳細な使い方も載っています）

- ・「nekoboard2」 <https://www.switch-science.com/catalog/2700/>
- ・「なのぼ〜ど AG」 http://tiisai.dip.jp/?page_id=935

2 計測・制御の準備をしよう

<計測・制御システムへの第一歩>



ここでは Scratch と「neko board2」(スイッチサイエンス社) を使用してセンサーを活用する方法を紹介します。Scratch で動作させられるものは、「Studuino」(アーテック社)、「なのぼ〜ど AG」(ちっちゃいものくらぶ)、「Scratch for Arduino」などがありますが、基本的には同様なスクリプトで動作します。

初めは手順を理解することが大変に思うかも知れませんが、それぞれの手順には情報の技術としての原理的な意味や、技術の基本的な仕組みと関係があります。実習を体験だけで終わらせないためにも、手順と作業の必要性にも触れるとよいでしょう。

③ -1)



① USB ケーブルで、neko board2 をパソコンと接続する

② Scratch を起動する

③ -2)

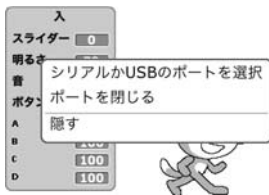


③ ScratchBoard 監視板を表示させる

ScratchBoard 監視板の表示させるためには、以下の手順が必要です。

- 1) ブロックパネル「調べる」をクリックする。
- 2) ブロックパレット下部「○○センサーの値」を、右クリックし、「ScratchBoard 監視板を表示」をクリックする。
- 3) ステージに表示された監視板を右クリックし、「シリアルか USB のポートを選択」をクリックする。
- 4) Windows なら COM 1 から順にクリックする。

③ -3)



④ neko board2 の値を Scratch に表示する

④, ⑤

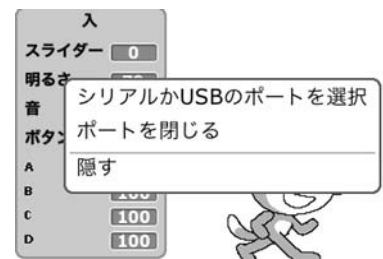


⑤ センサーの値をスプライトに表示させる

スプライトにセンサーの値を表示させるためには、「見た目」の「○○を言う」の○○に「調べる」の「○○センサーの値」を入れます。

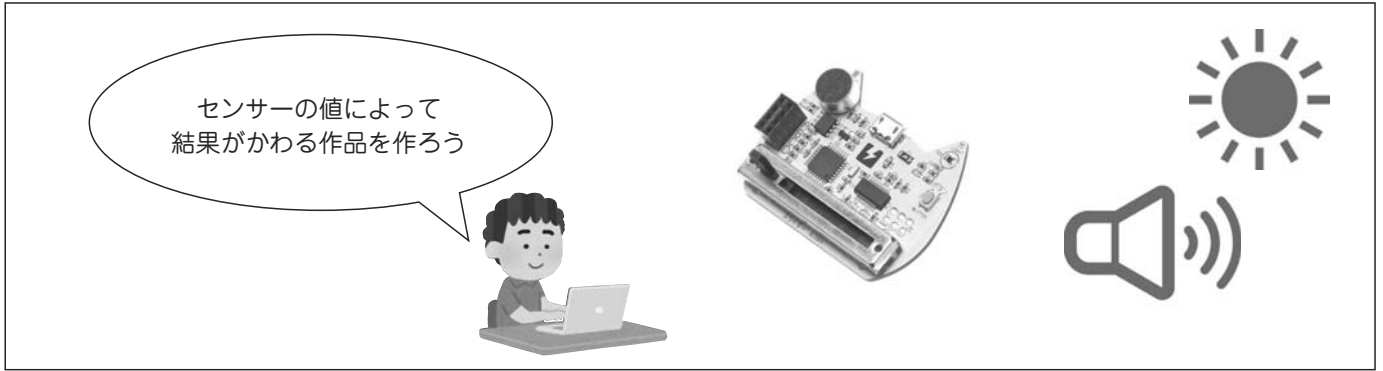
Point

- ボードの裏面に金属が当たらないように注意しましょう。
- Windows の COM ポートはデバイスマネージャーで調べることが出来ます。
- 監視板起動中は neko board2 等を取り外さないで下さい。終了時は、ステージ上に表示されている監視板を右クリックし、「ポートを閉じる」をクリックして下さい。
- ポートを閉じた後に、Scratch を終了させて下さい。
- スマホなどの充電専用ケーブルでは動作しません。



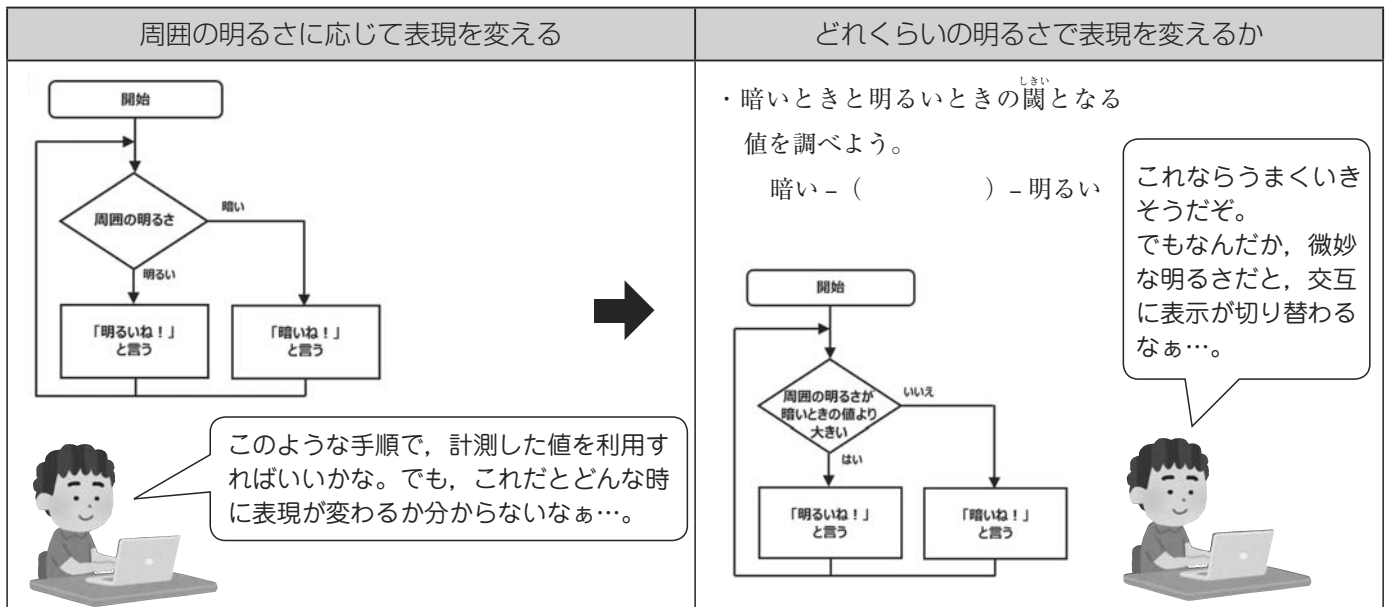
3 計測・制御の構想・制作をしよう

<計測・制御するイメージ>



Scratch とセンサーの接続が出来たら、センサーの値を利用していきましょう。明るさによってスプライトの動き方を変化させるために、具体的な課題を設定して、情報の処理の手順とスクリプトを考えていきます。

<情報の処理の手順のイメージ：フローチャートでの設計例>



①センサーの値がどのようなときに、どのような動きをさせるか、手順を考える

スプライトでセンサーの値によって表現が変わるようにします（目的によって最適な閾値を設定します）。

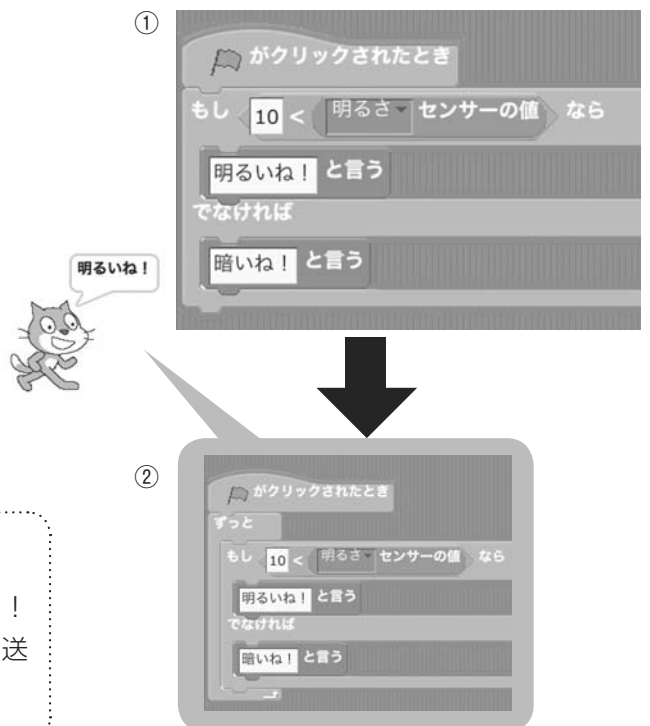
②手順に従ってスクリプトを制作する

明るさセンサーの値を常に確認させるため、「ずっと」動くようにブロックを追加します。

③考えた手順通りにスプライトが動いているか確認する（デバッグ）

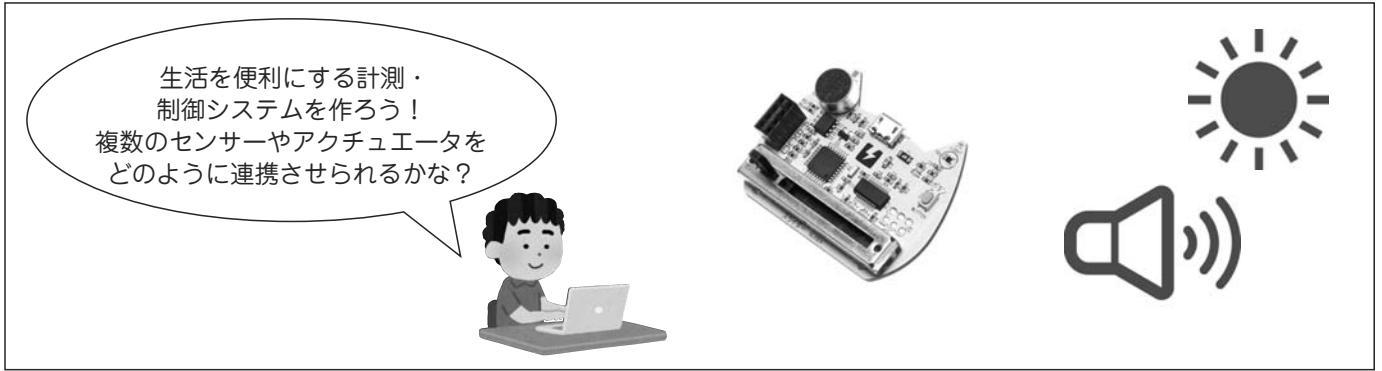
Challenge

- ・微かな明るさのときでも表現が安定するようにしてみよう！
- ・Mesh を利用して、センサーの値を違うコンピュータに送れるようにしてみよう。



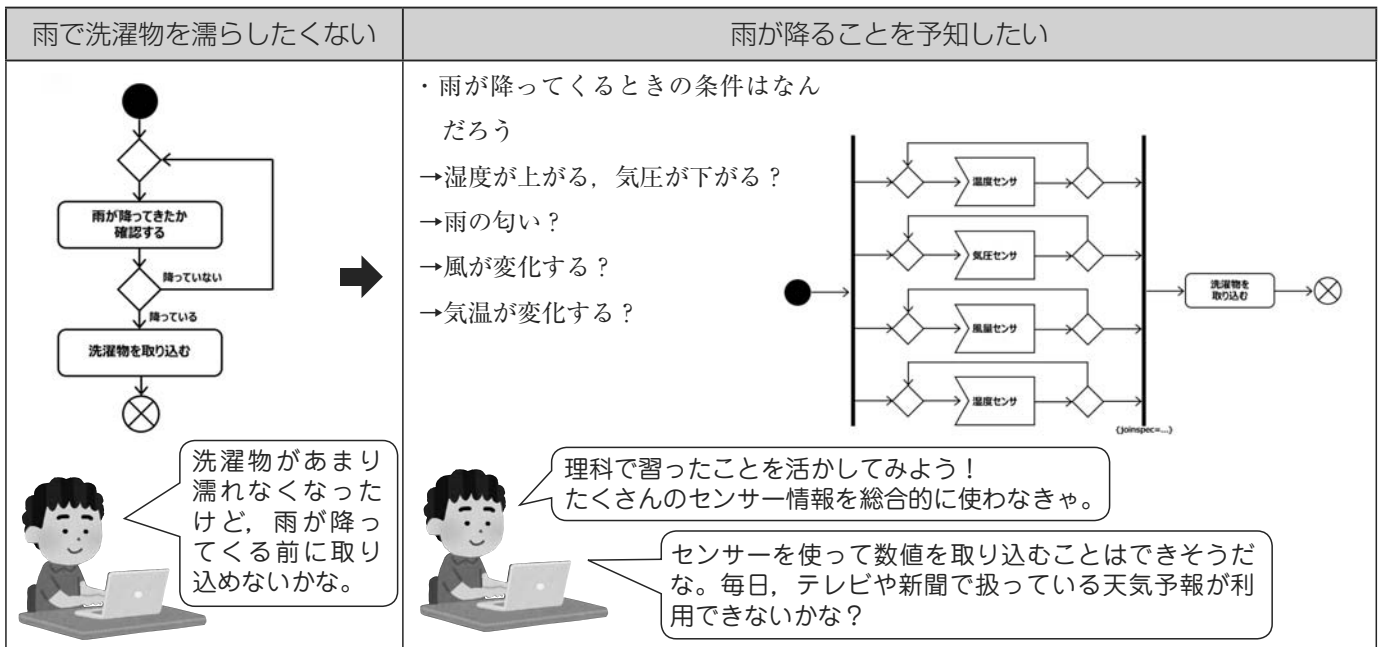
4 計測・制御システムを構想・制作しよう

<計測・制御システムのイメージ>

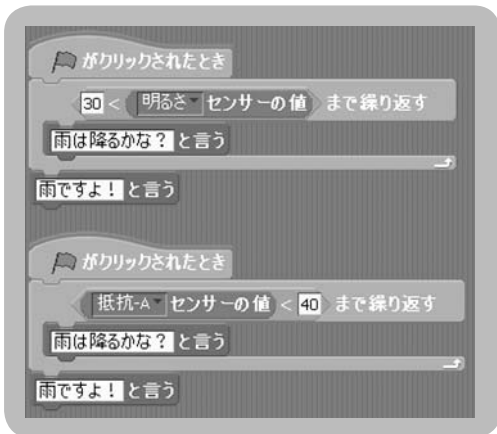


いよいよ計測・制御システムを構想していきます。ここでは、「雨が降ってきたら洗濯物を自動的に取り込むシステム」を例に構想・設計していきましょう。

<情報の処理の手順のイメージ：アクティビティ図での設計例>



複数のセンサーを同時に利用する例（並列処理）



同じスクリプトエリアに必要なだけスクリプトを書く



①システムの目的を考え、センサーやアクチュエータを選ぶ
目的に従って、スプライトごとにセンサーの値によってアクチュエータが動くようにします。センサーとアクチュエータを別々に作っておくと、デバッグしやすくなります。

②設計をもとにスクリプトを制作する

③設計通りにスプライトが動いているか確認する（デバッグ）

Challenge

・複数のセンサーを使って目的の動作をさせてみよう。

【更なる発展例】

・Raspberry PiにPythonをインターフェースとして、Scratchを使うことでインターネット上の天気予報を取得することが出来ます。

① ドリトルとは

1 ドリトルについて

ドリトルは、初心者でもプログラミングが学びやすいように設計された教育用プログラミング言語で、大阪電気通信大学の兼宗進教授により開発されました。教育用ですが、オブジェクト指向という考え方が取り入れられています。

ドリトルのプログラムは、テキストを入力しながら作成していきます。ブロック型のプログラミングとは異なり、キーボードから命令を直接打ち込む必要がありますが、以下のような特長があります。

- 命令が日本語を中心に構成されているため、初心者でもわかりやすい。
- オブジェクト指向を取り入れているため、社会におけるプログラミングの主流である、オブジェクト指向によるプログラミングを学ぶことができる。

2 オブジェクトとドリトルについて

オブジェクト指向は、現在のプログラミングの主流になっています。オブジェクトを作り、それに命令を与えて実行する方法です。プログラミングを習った経験がある人でもイメージしにくいかもしれません。そこで、オブジェクトを作るということを、ドリトルではどのようにプログラムで実現しているのか説明します。ドリトルには、あらかじめ何種類かのオブジェクトが用意されています。その代表的なものが、タートルオブジェクトです。例えばドリトルのプログラムで

```
「かめた=タートル!作る。」
```

という命令があります。この命令は、タートルという名前のオブジェクトを作り、それに「かめた」という名前をつけるという意味です。ドリトルではこの「かめた」に命令を与えてさまざまな図形やアニメーションのプログラムを作成します。

このように、オブジェクトを作るということは、ドリトルで用意されているオブジェクトを複製して命名するという手順をとります。本文では、オブジェクトのことをよりわかりやすくするために「部品」と表現しています。部品を作り、構成していくという考え方は、ものづくりの作業とどこか似ていないでしょうか。

ドリトルでは、タートルオブジェクト以外にも、タイマーオブジェクトやボタンオブジェクト、制御命令用のオブジェクト等が用意されています。それらを部品と考え、自分で使えるようにカスタマイズしながらプログラムを構成していくことは、技術・家庭技術分野における設計の学習にもつながる考え方です。さらには、自分が作ったプログラムをオブジェクト（部品）として考えると、他者と部品を共有しながらプログラムを作っていくことも可能になります。

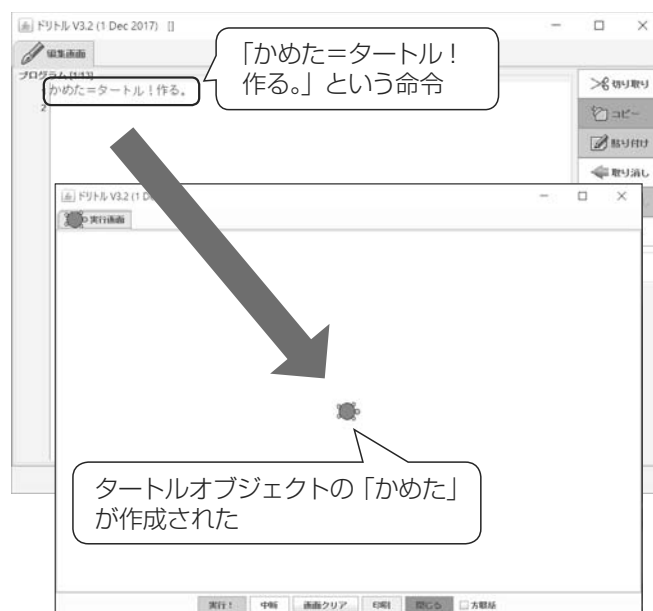


図1 ドリトルのプログラムのイメージ

3 ドリトルを準備しよう

ドリトルは、オンラインで使う方法と、ダウンロードして使う方法があります。本書では、ダウンロードして使う方法を解説します。

<http://dolittle.eplang.jp/download>

にアクセスし、Windowsであれば図2のリンクからダウンロードします。解凍すればインストールせずに使えます。

解凍したフォルダにある、「dolittle.bat」をダブルクリックすると、ドリトルが起動します。

ドリトルはWindows用、Mac用、Linux用、Raspberry Pi用があります。また、開発版もありますが、ここでは正式版のWindows用を例に進めます。



図2 ドリトルのダウンロード



図3 ドリトルの実行

4 ドリトルの画面構成と主な機能

ドリトルは、編集画面と実行画面で構成されています。編集画面では、主に日本語と数字を使ってプログラムを作成します。編集画面の「実行！」ボタンを押すと、実行画面に結果が表示されます。

編集画面

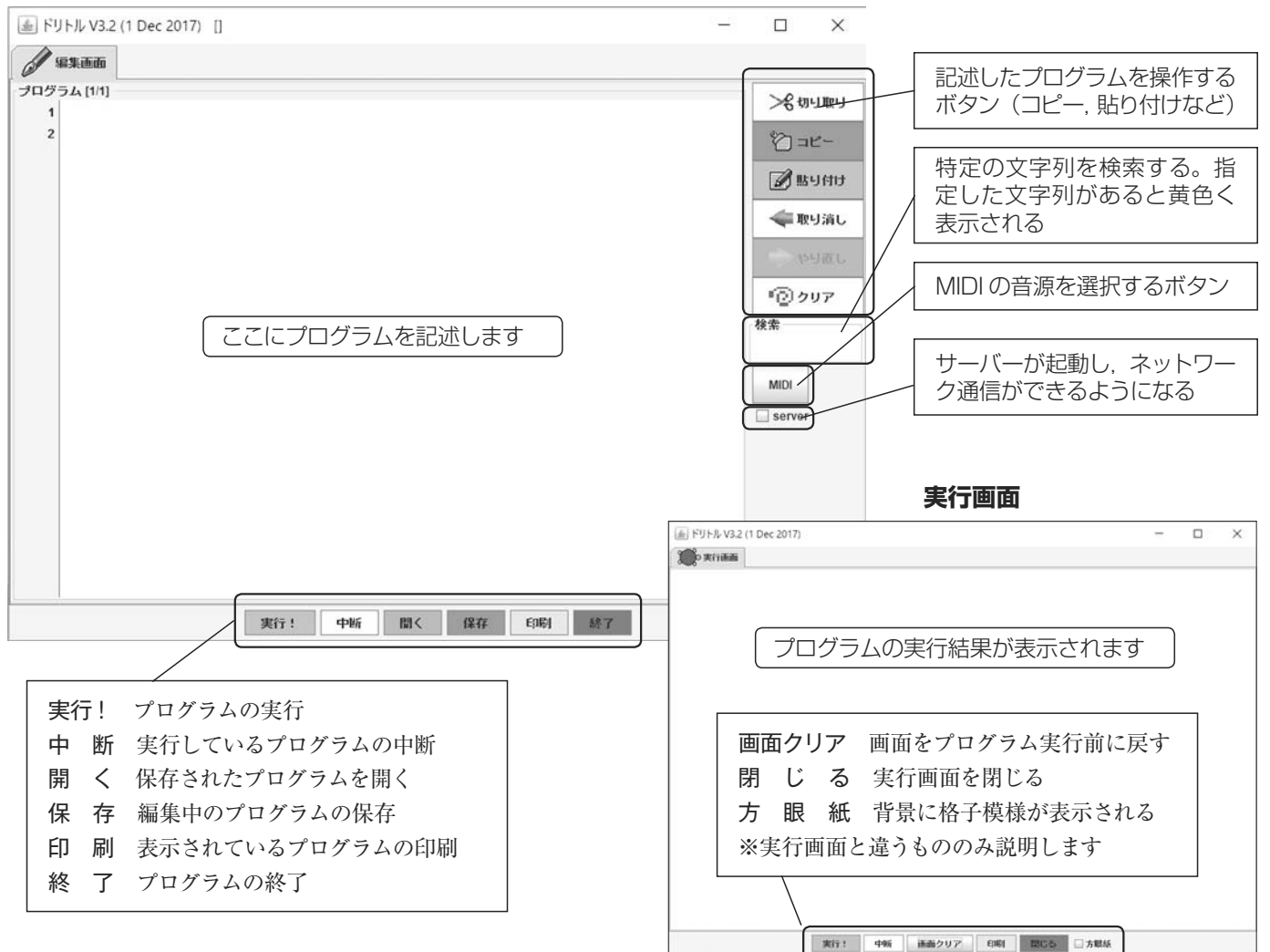
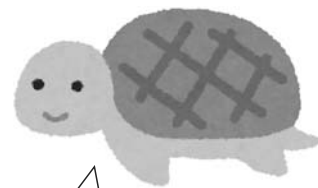


図4 ドリトルの画面構成と機能

② ドリトルの基本操作

タートルオブジェクト、アニメーション、GUI作成を含むプログラムを作成します。タートルオブジェクトが移動した跡が線になるため、星型を描画するように移動させましょう。また、移動の過程が見えるようにタイマーを設定することで、アニメーションのように動きを表示することができます。ドリトルではボタンなどのGUI作成も簡単にできます。描かれた線の色を変更するボタンを作成してみましょう。



ドリトルはカメ型のオブジェクトを動かすことが基本です。オブジェクトは変更することもできます。

1 オブジェクトの操作（図形の描画）

①オブジェクトを作る

ドリトルは、必要な部品を作りながらプログラムを作ります。

まずは「かめた」という名前のタートルオブジェクトを作成します。

①

```
1かめた=タートル!作る。
```

②オブジェクトに命令する

「かめた」を移動（歩く）して、100進んだところで144°右回転します。5回繰り返すと星型になります。

②（順次）

```
1かめた=タートル!作る。
2かめた!100 歩く。
3かめた!144 右回り。
```

実行画面



（順次）作った部品に何をさせるかを命令します。命令を並べれば、その順で実行します。

（反復）同じ命令を何度も繰り返す場合は、短くまとめることができます。

（反復）

```
1かめた=タートル!作る。
2「
3  かめた!100 歩く。
4  かめた!144 右回り。
5!5 繰り返す。
```

「」の中を5回繰り返す命令

実行画面



※同じ部品への命令は、短縮して書くことができます。

```
1かめた=タートル!作る。
2「
3  かめた!100 歩く 144 右回り。
4!5 繰り返す。
5星=かめた!図形を作る。
```

「歩く」と「右回り」を1行で記述できる

③新しく部品を作る

描いた図形を、新たな部品として使うことができます。描いた図形を「星」という部品として名づけます。

色は「黒、赤、緑、青、黄色、紫、水色、白」などがあらかじめ準備されています。好きな色を作ることも可能です。

③

```
1かめた=タートル!作る。
2「
3  かめた!100 歩く 144 右回り。
4!5 繰り返す。
5星=かめた!図形を作る。
```

実行画面



5 星=かめた！（黄色）図形を作る。

実行画面



④

```
1 かめた=タートル！作る。
2 「
3   かめた！100 歩く 144 右回り。
4 」！5 繰り返す。
5 星=かめた！（黄色）図形を作る。
6 星！-100 100 移動する。
7 星！20 右回り。
```



※図形を作る前に色を指定すると、色を塗ることができます。

④ 作った部品（図形）を使う

作った部品（図形）は移動したり複製したり色を塗り直したりできます。「星」に命令して移動してみましょう。

< 命令 >

図形の名前！（a）（b）移動する。

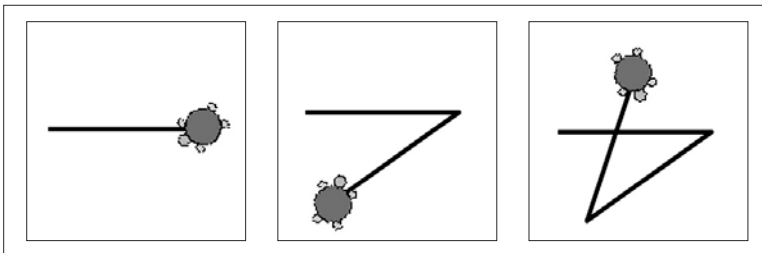
X 軸方向の移動量 a, Y 軸方向の移動量 b を設定します。図形を原点として、右方向および上方向がプラス、左方向および下方向がマイナスになります。

2 アニメーション

タートルや作った図形などの部品を一定間隔で動かすと、アニメーションを作ることができます。

①

```
1 かめた=タートル！作る。
2 時計=タイマー！作る。
3 時計！「
4   かめた！100 歩く。
5   かめた！144 右回り。
6 」実行。
```



実行画面

① タートルを動かす

タイマーを使って一定時間ごとに動かすことで、アニメーションのように見せることができます。

< 命令 >

時計！「・・・」実行。

・・・の部分に命令を書くと、一定間隔で記載した命令を実行します。

② 速度や時間、回数を変える

タイマーで動かす間隔や時間、繰り返す回数などを指定することができます。

< 命令 >

タイマーの名前！ <秒数> 間隔。

実行する命令の実行間隔を指定する。

<秒数>に設定する値の単位は秒。

②

```
1 かめた=タートル！作る。
2 時計=タイマー！作る。
3 時計！0.5秒 間隔。
4 時計！10秒 時間。
5 時計！「
```

0.5 秒毎に 10 秒間命令を実行

```
1 かめた=タートル！作る。
2 時計=タイマー！作る。
3 時計！1秒 間隔。
4 時計！5回 回数。
5 時計！「
```

1 秒毎に 5 回命令を実行

< 命令 >

タイマーの名前！ <秒数> 時間。

最初に命令を実行してから何秒間繰り返すかを指定する。

< 命令 >

タイマーの名前！ <回数> 回数。

命令を実行する回数を指定する。

③ 図形を動かす

図形を部品化している場合は、図形自体をアニメーションに利用できます。

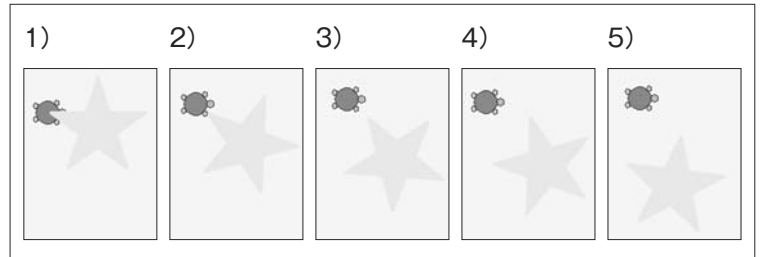
図形を「星」として部品化し、タイマーを使って回転しながら右下に移動させてみましょう。

③

```
1 かめた=タートル!作る。
2 「
3   かめた!100 歩く 144 右回り。
4 」!5 繰り返す。
5 星=かめた!(黄色)図形を作る。
6 時計=タイマー!作る。
7 時計!「
8   星!10 -10 移動する。
9   星!20 右回り。
10」実行。
```

Challenge

- ・違う形の図形を描いてみよう。
- ・図形を違う方向へ移動させてみよう。
- ・図形がゆっくり動くようにしてみよう。



3 GUI (使う人の操作に应答する)

プログラムを使う人の操作などに対して、動きの変化などで应答する機能を持ったプログラムが作れます。

① ボタンを作成する

ボタンを作成して、押すと星の色が変わるようにしてみましょう。動かす人の動作に応じて結果が変わるため、双方向性のあるコンテンツにつなげていくことも考えられます。

①

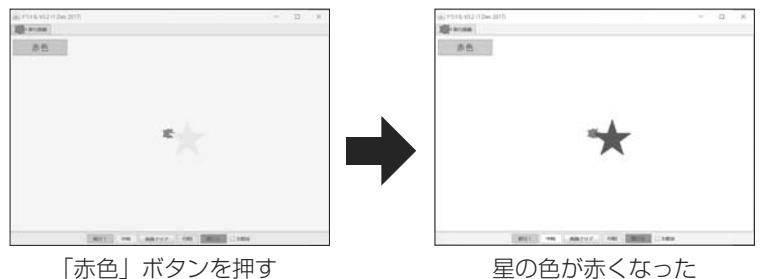
```
1 かめた=タートル!作る。
2 「
3   かめた!100 歩く 144 右回り。
4 」!5 繰り返す。
5 星=かめた!(黄色)図形を作る。
6
7 赤色ボタン=ボタン!“赤色”作る。
8 赤色ボタン:動作=「
9   星!(赤)塗る。
10」。
```

< 命令 >

ボタンの名前= ボタン!“ ” 作る。
(ボタンの名前) という名前のボタン (部品) を新しく作る。「 ” 」内はボタンのラベル。

< 命令 >

ボタンの名前:動作= 「・・・」。
・・・に命令を書くと、そのボタンを押した時に命令を実行する。



実行画面

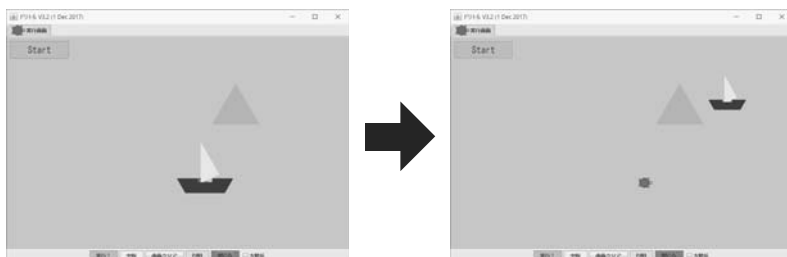
4 動く絵本の作成（応用）

タートルを動かして絵を描き，その絵を動かして，物語のあるコンテンツを作ることができます。

下記は，海の上を進むヨットをイメージしました。進むほどにヨットが小さくなっていきます。

```

① 1 画面！（水色）塗る。
   2 かめた=タートル！作る。
② 3 山=かめた！
   4     120 右回り 120 歩く
   5     120 左回り 120 歩く （緑）図形にする。
   6 山！150 150 移動する。
   7
③ 8 帆=かめた！
   9     90 左回り 100 歩く
  10     150 右回り 90 歩く 閉じる （黄色）図形を作る。
  11
④ 12 船=かめた！
  13     60 左回り 30 歩く 90 左回り 10 歩く
  14     90 右回り 50 歩く 120 右回り 40 歩く
  15     60 右回り 100 歩く 60 右回り 40 歩く
  16     120 右回り 50 歩く 90 右回り 10 歩く
  17     90 左回り 閉じる （赤）図形を作る。
  18
⑤ 19 時計=タイマー！作る。
  20 スタート=ボタン！” Start” 作る。
  21 スタート：動作=「
⑥ 22     時計！「
  23         帆！5 5 移動する。
  24         船！5 5 移動する。
  25         帆！0.99 拡大。
  26         船！0.99 拡大。
  27     」実行。
  28 」。
```



実行画面

① 画面の色の設定

海の色が背景なので，画面の色を<水色>にします。

<命令>

画面！（<色>）塗る。

画面の背景色を好きな色で塗ることが出来る。

② 山の作成

三角形の図形を緑色にし，右上に移動させます。

③ 帆の作成

黄色い三角形の図形を作成します。

<命令>

タートル！閉じる。

描き始めの位置まで線を引く。

④ 船の作成

船の形をした赤い図形を作成します。

⑤ ボタンの作成

「スタート」という名前で，「Start」というラベルのボタンを作成します。

⑥ アニメーションの設定

帆と船を右上に移動させながら，同時に少しずつ縮小していくように設定します。

<命令>

図形！ <大きさ> 拡大。

図形を<大きさ>の倍率で拡大する（1以下の値を設定すれば縮小する）。

Challenge

・海の上で波が動くような動作を追加してみよう。

③ ドリトルで通信しよう

1 ネットワークを利用した通信プログラム

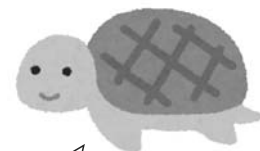
ドリトルには、ネットワークを利用して、オブジェクトを送信したり、受信したりする機能があります。送信側で作成した図形を、受信側で見られるようなやりとりができるプログラムを作成します。



送信側



受信側



送信側が作ったデータを、受信側で受け取って表示することができます。Web上のサービスも同じような仕組みです。

①準備

ドリトルで通信プログラムを作るときは、はじめにサーバーを用意します。

サーバーとしたいコンピュータでドリトルを起動し、編集画面右側の「server」にチェックを入れます。正しくサーバーが起動したときは、「server」表示の下に IP アドレスが表示されます。

(注意) コンピュータによっては、チェックをした直後にネットワーク接続を許可するかの選択画面が出ることがあります。

①



「server」ボタンにチェックを入れる

②送信プログラムの作成

○サーバーに接続する

接続するサーバーを設定します。同じコンピュータの中で通信する場合は、「localhost」とします。別のコンピュータ上のサーバーに接続するには「」の部分に接続したいサーバーの IP アドレスを記述します。(事前に教室内のコンピュータの IP アドレスを確認しておく必要があります。)

○送信用のデータ(図形)を作成する

タートル(かめた)の書いた絵から図形を作り、送信用のデータとします。

○サーバーに登録する

図形(データ)に名前をつけて、サーバーに登録します。

② ○サーバーに接続する

1サーバー!“localhost” 接続。

○送信用のデータ(図形)を作成する

4「かめた!40 歩く 120 右回り。」!3 繰り返す。
5三角=かめた!(赤) 図形を作る。

○図形に名前をつけて、サーバーに登録する

6サーバー!“sankaku” (三角) 書く。

送信プログラム(完成)

1サーバー!“localhost” 接続。
2かめた=タートル!作る。
3
4「かめた!40 歩く 120 右回り。」!3 繰り返す。
5三角=かめた!(赤) 図形を作る。
6サーバー!“sankaku” (三角) 書く。
7
8「かめた!40 歩く 90 右回り。」!4 繰り返す。
9四角=かめた!(青) 図形を作る。
10サーバー!“shikaku” (四角) 書く。

③ 〇名前入力用のエリアと読み込み実行用のボタンの作成

```
3 名前入力エリア=フィールド!作る。  
4 呼び出しボタン=ボタン!“呼び出し” 作る。
```

〇サーバーからデータを取得する

```
6 呼び出しボタン:動作=「  
7 名前=名前入力エリア!読む。  
8 受信内容=サーバー!(名前)読む。  
9」。
```

発展

```
6 名前=“未設定”。 //★  
7 呼び出しボタン:動作=「 //★  
8 「名前!=“未設定”」! なら「 //★  
9 受信内容!消える。 //★  
10 」実行。  
11  
12 名前=名前入力エリア!読む。  
13 受信内容=サーバー!(名前)読む。  
14」。
```

受信プログラム (完成)

```
1 サーバー!“localhost” 接続。  
2  
3 名前入力エリア=フィールド!作る。  
4 呼び出しボタン=ボタン!“呼び出し” 作る。  
5  
6 名前=“未設定”。  
7 呼び出しボタン:動作=「  
8 「名前!=“未設定”」! なら「  
9 受信内容!消える。  
10 」実行。  
11  
12 名前=名前入力エリア!読む。  
13 受信内容=サーバー!(名前)読む。  
14」。
```

「sankaku」を呼び出した結果

③受信プログラムの作成

〇受信画面用の部品を用意する

名前を指定して、データを読み出すために、名前入力用のエリアと読み込みを実行するためのボタンを準備します。

〇サーバーからデータを取得する

入力エリアに記述した名前を取得し、その名前を持つデータをサーバーから取得します。

※発展

★が追加部分です。受信ボタンを押したときに、前の表示内容を消すようにしています。

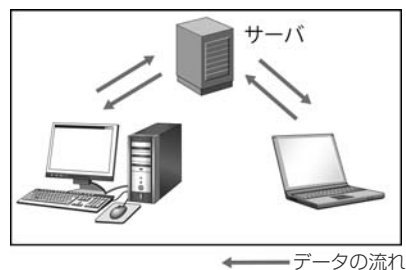
“受信内容!消える。”だけでは、初めてボタンを押したときに消す図形がないためにエラーが起こります。それを回避するために名前が未設定でないときだけ、図形を消す処理を行うようにします。

〇受信プログラムの完成

②の送信プログラムでは、「sankaku」と「shikaku」という名前で、データを登録しています。

受信プログラムでは、名前を変更して「呼び出し」ボタンを押すことで、異なるデータを表示することができます。

たくさんの人とデータをやりとりするためには、どのようなルールが必要かなどを考えてみるとよいでしょう。



このプログラム (“localhost”ではなく、IPアドレスを指定している場合)では、受信者と送信者の間で直接通信が行われているように見えますが、実際にはサーバーを介して情報のやりとりが行われています。プログラムを作る過程で、通信を実現するための仕組みや技術について考えるきっかけになります。

2 ネットワークを利用した通信プログラムの応用

図形だけではなく、文字や数値、ドリトルの部品（オブジェクト）も相手に送信することができます。

例えば、作ったタートルの部品に、絵を描くための命令を追加してから送り、受信したときにその命令を実行することで、複雑な絵を描くこともできます。

① 送信プログラムの作成

※ p.24 の送信プログラムを修正

「かめた」というタートルの部品に、「絵を描く」という命令を追加してから、「kameta」という名前でサーバーに送るプログラムを作成します。

帆と船を描く一連の命令を「絵を描く」という命令にまとめています。

Point

○ ドリトルでは、部品（オブジェクト）にオリジナルの命令を追加することができます。
オブジェクト：<命令の名前>=「・・・」。
「 」の中に書いたプログラムを、<命令の名前>に書いた名前の命令として追加できます。
このプログラムでは、帆と船を書くプログラムを「かめた」という部品（オブジェクト）の「絵を描く」という命令として追加しました。
追加した新しい命令は、すでに存在する命令と同じように利用できます。
プログラム中では「かめた！絵を描く。」がこれに当たります。

② 受信プログラムの作成

※ p.25 の受信プログラムを修正

サーバーから「kameta」というプログラムを呼び出します。呼び出したプログラム「kameta」に入っている「絵を描く」という命令を実行します。

ここでの「自分」とは、作った命令を実行するオブジェクト（部品）を指しています。

②のプログラムの19行目なら「かめた」になりますし、(月)のプログラムの14行目ならば「受信内容」というオブジェクトを指します。

②（送信側）で「かめた」という名前だったオブジェクトは、(月)（受信側）で受け取った時には「受信内容」という名前に変わっています。そのためオブジェクトの名前を書かずに「自分」という書き方にする必要がありました。

①

```
1 サーバー！"localhost" 接続。
2 かめた=タートル！作る。
3
4 かめた：絵を描く=「
5 帆=自分！
6   90 左回り 100 歩く
7   150 右回り 90 歩く 閉じる
8   (黄色) 図形を作る。
9
10 船=自分！
11   60 左回り 30 歩く 90 左回り 10 歩く
12   90 右回り 50 歩く 120 右回り 40 歩く
13   60 右回り 100 歩く 60 右回り 40 歩く
14   120 右回り 50 歩く 90 右回り 10 歩く
15   90 左回り 閉じる
16   (赤) 図形を作る。
17 」。
18
19 かめた！絵を描く。
20
21 サーバー！"kameta" (かめた) 書く。
```

実行画面



受信プログラムにも、データを受信したら、追加されている命令を実行するように修正します。

②

```
7 呼び出しボタン：動作=「
8   「名前!="未設定"」！なら「
9     受信内容！消える。
10  」実行。
11
12 名前=名前入力エリア！読む。
13 受信内容=サーバー！（名前）読む。
14 受信内容！絵を描く。
15 」。

```

実行画面



○イタズラプログラム（送信プログラムを修正）

```
かめた：絵を描く＝「  
システム！終了。  
」。
```

「絵を描く」というのはあくまで命令の名前なので、絵を描く動作をするとは限りません。一方で受信側からはどのような内容なのかわからないため、不利益な動作を実行してしまう可能性があります。ウイルス付きのファイルを例に、受信者が実行してしまうといとも簡単にこのようなことができてしまうということを説明する投げかけが考えられます。

注意！イタズラプログラム

このプログラムでは、「絵を描く」命令の内容を変更するといたずらができてしまいます。

この命令を受け取った側のプログラムは、受信と同時に命令を実行します。

すると、絵を描くのではなくドリトルを終了しようとしてしまいます。

コンピュータウイルスやセキュリティの学習につなげたり、このようなプログラムへの対策などの検討につなげたりすることが考えられます。

参考 ドリトルのいろいろな命令

ドリトルにはいろいろな命令が用意されています。これらの命令を使うことで、多様なプログラムを実現することができます。

タートルの動き

○中心に戻る…画面の中心に戻ります。

(例) かめた！ 中心に戻る。

○線の太さ…線の太さを変更します（初期は3）。

(例) かめた！ 6 線の太さ。

○円…円を描きます（数値は半径）。

(例) かめた！ 100 円。

※正の数は右回り、負の数は左回り。

○変身する…タートルの見た目を変更します。

(例) かめた！ “画像ファイル名” 変身する。

※画像ファイルはドリトル本体と同じフォルダに入れる必要があります。

※使える画像形式は、jpg、png、gifです。

※car、star、appleなど、予め用意されているものもあります（後に「png」を付けます）。

色

画面や図形などの色が指定できます。黒、赤、緑、青、黄色、紫、水色、が標準ですが、色を指定したり混ぜたりすることができます。

○作る…RGBの値を指定して色を作ります。

(例) 灰色 = 色! 128 128 128 作る。

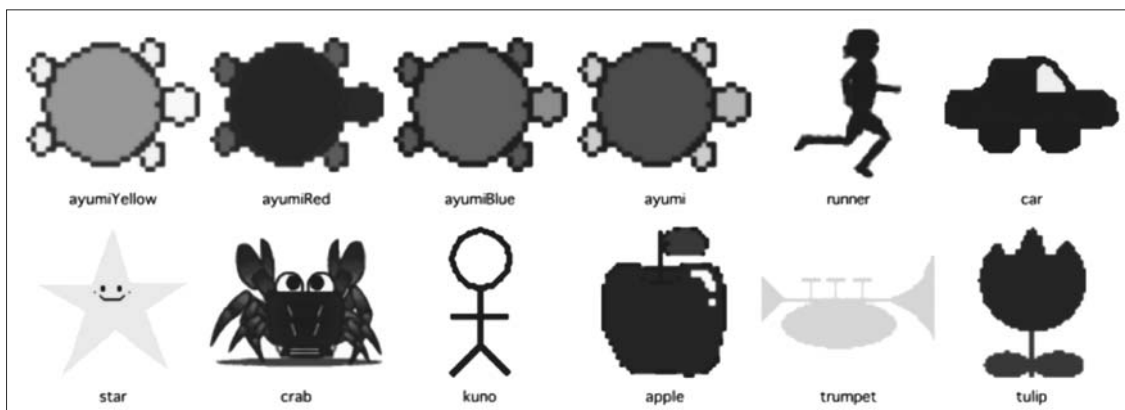
○混ぜる…色を混ぜます。

(例) オレンジ = 光! (赤) (黄) 混ぜる。

※「光!」だと加法混色に、「絵具!」だと減法混色になります。

○半透明にする…色を半透明にします。

(例) 空 = 青! 半透明にする。



タートルに利用できる画像の例

④ ドリトルで機器を動かそう

ここでは、Arduino 互換基板である、Studuino（アーテック社）を用いて、センサーによる計測結果を利用して LED やモーター等を制御するプログラムを作成します。

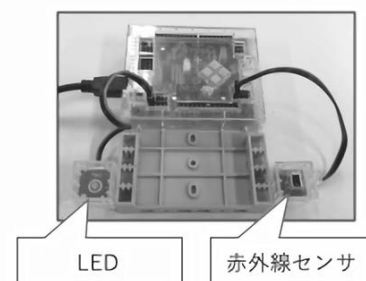
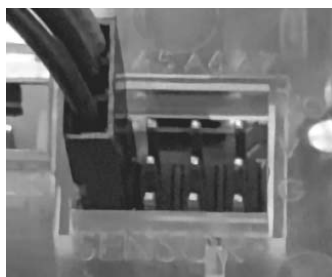
1 Studuino を動かす（LED 点灯）

ここでは Studuino を利用した計測・制御プログラムの書き方（ルール）と、実際に利用するまでの手順を確認します。

① 必要な部品を制御基板に接続する

目的を実現するためのセンサーやアクチュエータなどの部品を、プログラムによって制御を行う対象（ここでは Studuino）に接続します。接続する端子には名前がついています。プログラムを書くときは、この名前を使って接続した部品に命令を与えます。この例では LED を A4 端子に接続しています。

①



LED

赤外線センサ

② プログラムを記述する

プログラムを例に従って記述します。例は、制御対象である Studuino に LED を使うことを宣言し、LED の点灯と消灯を 1 秒ごとに繰り返すプログラムです。

②

<プログラムの手順>

- (1) Studuino 基板を使う事を指示。
- (2) プログラム実行時、初めに一回だけ実行するプログラムを指示。センサーやアクチュエータの初期設定を行う。
- (3) 動作の主体となるプログラムを指示。
- (4) 作ったプログラムを基板に転送することを指示。

③ プログラムを転送&実行する

パソコンと Studuino を接続してから、ドリトルの「実行！」ボタンを押します。

転送の確認画面が出てきたら「はい」ボタンを押し、しばらく待ちます。パソコンから基板にプログラムが正しく転送されると「転送完了」の表示が現れ、しばらくすると自動的に作ったプログラムが実行されます。

```

1 システム!“ studuino” 使う。 //・・・(1)
2 最初に実行=「 //・・・(2)
3 ST!“ A4” デジタルLED。
4 」。
5 繰り返し実行=「 //・・・(3)
6 ST!“ A4” 1 書く。
7 ST!1000 待つ。
8 ST!“ A4” 0 書く。
9 ST!1000 待つ。
10 」。
11 ST!転送。 //・・・(4)

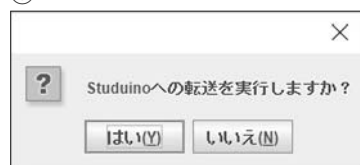
```

Point

- プログラムは、リセットボタンを押したり電源を入れ直したりすることで、いつでも実行できます。
- 「転送完了」が表示されずエラーが出る場合は、プログラムに誤りがないか、ケーブルが正しく接続されているかなどを確認してください。

- ST!“A4” デジタルLED：A4 端子に LED を接続し利用することを指示
- ST!“A4” 1 書く：A4 端子に接続 LED を点灯
「0 書く」で消灯
- ST!1000 待つ：次の命令までに 1000ms 待機する

③



2 センサーを使ってLEDの点灯を制御する (Lチカ)

LEDの点灯制御を通して、センサーの利用の方法や具体的なプログラムの書き方を確認します。ここでは、赤外線センサーの前に手をかざすと、LEDが点灯するプログラムを作ります。

①

```
2 最初に実行 = 「
3   ST!" A4" デジタルLED。
4   ST!" A0" 赤外線センサー。
5」。
```

○ ST!" A0" 赤外線センサー：A0端子に赤外線センサーを接続し利用することを指示

②

```
6 繰り返し実行 = 「
7   「(ST!" A0" 読む) > 100」! なら「
8     ST!" A4" 1 書く。
9   」そうでなければ「
10    ST!" A4" 0 書く。
11  」実行。
12」。
```

「手をかざすとLEDを点灯(そうでないときは消灯)」という条件に応じた分岐を実現している部分が下記の記述の箇所です。

```
「***」!なら「…」そうでなければ「###」実行。
```

「***」に分岐する条件を書きます。条件を満たしているときに実行する内容を・・・の中に書き、条件を満たさないときの内容は「###」に書きます。

プログラムを分岐するための条件の書き方は下記の通りです。例えば、センサーからの計測値が30である場合に分岐をしたいときは条件(・・・)の部分に「(ST!"A0"読む) == 30」を書きます。否定の場合(そうでなければ)に記述することがない場合は省略して「***」!なら「…」実行。」と書くことも可能です。

***に書いた条件が成立しているときに・・・に記述した命令を実行する。

- A == B : A と B が同じとき実行。
- A != B : A と B が異なるとき実行。
- A < B : A が B がより小さいとき実行。
- A > B : A が B がより大きいとき実行。
- A <= B : A が B 以上のとき実行。
- A >= B : A が B 以下のとき実行。

Point

○ Studuino へのプログラムの書き方の詳細や、本書で紹介していない部品を使ったプログラミングの方法などが、ドリトル公式ページのマニュアルの中で紹介されています(「使い方の説明を見よう」の項目からマニュアルを確認できます。マニュアルの「Studuino と通信しよう」の項目を参照)。

<http://dolittle.eplang.jp/>

① 使う部品を追加する

Studuino に、新たに使用する部品である「赤外線センサー」を A0 端子に接続して利用することを指示します。どの部品を使うかの指示は「最初に実行」の中に書きます。

② センサーの計測結果を利用して、LEDを制御する

「赤外線センサー」は光(赤外線)の強さを計測するセンサーです。Studuino の赤外線センサーは、センサー自身が赤外線を出しており、その光が物体に反射して戻ってきた量を計測することができます。反射量が多いと計測値は大きく、反射量が少ないと計測値も少なくなります。センサーの前に手をかざしたときは反射量が多くなるので、計測値は大きくなります。

③ プログラムを転送&実行する

パソコンと Studuino を接続してから、ドリトルの「実行！」ボタンを押します。

転送の確認画面が出てきたら「はい」ボタンを押し、しばらく待ちます。パソコンから基板にプログラムが正しく転送されると「転送完了」の表示が現れ、しばらくすると自動的に作ったプログラムが実行されます。

Challenge

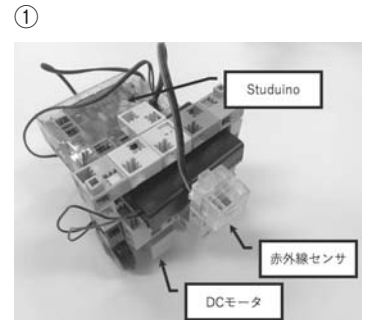
- ・一度点灯したら、しばらく点灯し続けるようにしてみよう。
- ・LED と赤外線センサを追加して、手がかざされていないときだけ追加したLEDが光るようにしてみよう。

3 応用例

同じセンサーを利用して、計測対象を検査し制御のルールを考えることでいろいろなことを自動化することができます。また、LED にモーターなどのアクチュエータを接続することで、仕事の内容を変えることもできます。

3.1 自動停止ロボット

赤外線センサーを前方に向けると、何かに接近したかを判断することができます。また、LED の代わりにモーターを制御すれば、移動ロボットを作ることができます。これを利用して、前方に何かがあることを検知したときに自動で停止するロボットを作成することができます。



① 使う部品を接続する

Studuino に、DC モーターと、A0 端子に「赤外線センサー」を接続します。

② プログラムを記述する

赤外線線の反射量が大きくなると停止し、それ以外では進むようにプログラムを作成します。

③ プログラムを転送&実行する

②

```

1 システム!" studuino" 使う。
2 最初に実行=「
3   ST!DCモーター。
4   ST!" A0" 赤外線センサー。
5 」。
6 繰り返し実行 =「
7   「(ST!" A0" 読む) >50」! なら「
8     ST!停止。
9   」そうでなければ「
10    ST!前進。
11  」実行。
12 」。
13 ST!転送。
    
```

- ST!"A0" 読む : A0 に接続したセンサーの値を取得する
- ST!停止 : DC モーターを停止する
- ST!前進 : DC モーターを前進する

3.2 ライントレーサ

センサーを床面に向けると、床の状況を判断することができます。これを利用してラインレースカーを作成することができます。ここでは、赤外線センサーを A1 に接続しました。

① 使う部品を接続する

② プログラムを記述する

黒い部分は光の反射が弱く、白い部分は光の反射が多いことを利用し、床の状況（線の上か）を判断して右折と左折を繰り返すことで線の上を走行します。

このプログラムでは、白い部分では左に進み、黒い部分では右に進みます。

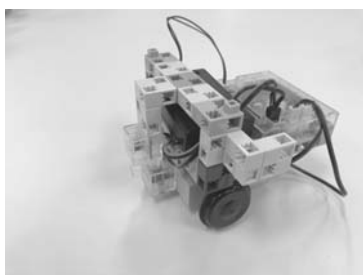
③ プログラムを転送&実行する

②

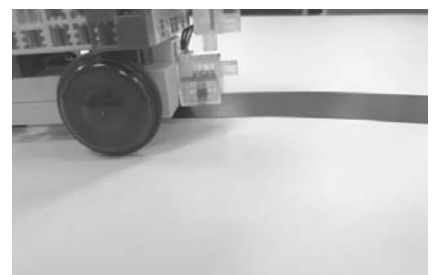
```

1 システム!" studuino" 使う。
2 最初に実行=「
3   ST!DCモーター。
4   ST!" A1" 赤外線センサー。
5 」。
6 繰り返し実行 =「
7   「(ST!" A1" 読む) >100」! なら「
8     ST!左折。
9   」そうでなければ「
10    ST!右折。
11  」実行。
12 」。
13 ST!転送。
    
```

- ST!"A1" 読む : A1 に接続したセンサーの値を取得する
- ST!左折 : DC モーターを左に進むように回転する
- ST!右折 : DC モーターを右に進むように回転する



赤外線センサーを床に向けように取りつける



ラインの内側を左右に動きながら進む

3.3 荷物運びロボット

これまでのプログラムを組み合わせ、荷物を運ぶためのアームを取りつけると、荷物を運ぶロボットを作ることができます。

このプログラムは、荷物をおろす場所の壁を検知するまで、床の線をトレースし、壁を見つけたら、荷物の乗ったアームをおろします。

②

```
1 システム!" studuino" 使う。
2 最初に実行=「
3   ST! DCモーター。
4   ST!" A0" 赤外線センサー。
5   ST!" A1" 赤外線センサー。
6   ST!" D9" サーボモーター。
7   ST!" D9" 90 書く。
8 」。
9 繰り返し実行=「
10  「(ST!" A0" 読む) > 50」! なら「
11   ST! 停止。
12   ST!" D9" 0 書く。
13  」そうでなければ「
14   ST!" D9" 90 書く。
15   「(ST!" A1" 読む) > 100」! なら「
16   ST! 左折。
17   」そうでなければ「
18   ST! 右折。
19   」実行。
20  」実行。
21 」。
22 ST! 転送。
```

○ ST!"D9" サーボモーター:
D9 端子にサーボモーターを接続し利用することを指示する
○ ST!"D9" 0 書く.:
サーボモーターの角度を指示する

<サーボモーター>

回転角度を指定することのできるモーターです。

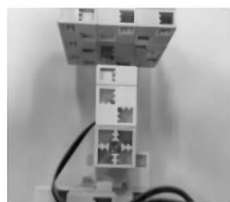
今回は、荷物の上げ下ろしができるように、アームの付け根にサーボモーターを接続しています。



サーボモーター



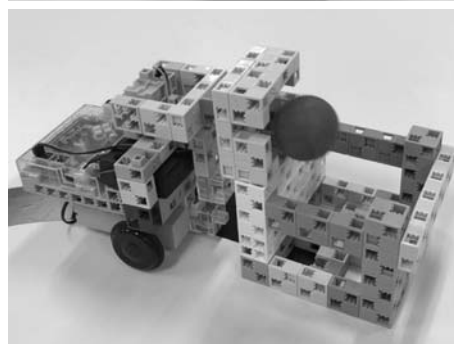
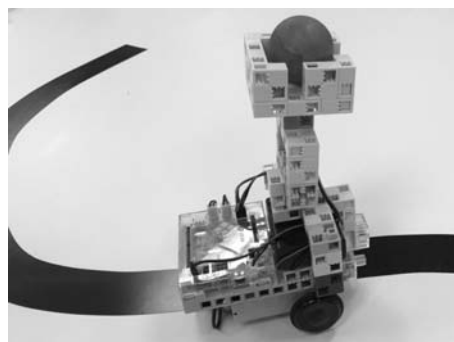
0°



90°



180°



① 使う部品を接続する

Studuino に、DC モーター、A0、A1 端子に「赤外線センサー」、D9、端子にサーボモーターを接続します。

② プログラムを記述する

プログラムの最初でサーボモーターを利用すること、90°の位置にすること（荷物を乗せるため）を設定しています。

繰り返しの中では、まず A0 端子の赤外線センサーを用いて「壁に到達したか」を判断しています。壁に達した場合にはアームをおろします。

壁に達していない場合には、アームを立てた状態（90°）に設定します。そのあと、A1 端子の赤外線センサーを使って「床の線上にいるか」を判断し、ライントレースの実施しています。

③ プログラムを転送&実行する

Challenge

- ・荷物をおろしたあと、振り返ってコースを戻るようにしてみよう。
- ・サーボモーターを利用して、障害物をどけるロボットを作ってみよう。

1 JavaScriptとは

1 JavaScriptについて

ここでは、テキスト型のプログラミング言語で、スマートフォンやパソコン等で普段よく見る Web サイト等に多く使われている JavaScript を試してみましょう。

テキスト型のプログラミング言語は、命令の打ち間違いや、命令同士のつながりのルールなど、ビジュアル型の言語に比べると覚えなければいけないことが多いですが、プログラムの中の記述を検索・置換できる、コピー&貼り付けなどプログラムの制作・編集効率が高いといった特徴があります。また、世の中で使用されているソフトウェアの多くはテキスト型の言語で開発されているため、社会とのつながりで考えた場合、中学生にはこのような文字で記述するプログラミングにも触れさせたいところです。

そこで、プロのプログラマーも使用する JavaScript を使って、テキスト型のプログラミングを体験してみましょう。

1.1 JavaScriptの特徴

- Web ブラウザ上でプログラムが実行されるので、パソコンやスマートフォン、タブレットなど多くの機種で動作させることができる。
- メモ帳さえあれば、すぐにプログラミングができる。
- チャットやアニメーション、グラフ表示など高度な機能を、ライブラリというものを使って拡張できる。などがの特徴挙げられます。

1.2 JavaScriptで作ってみよう（ウィンドウの表示）

例えば、メモ帳などのテキストエディタで図1のプログラムを入力し、ファイル名を「test.html」として保存し（図2）、そのファイルを Web ブラウザにドラッグ&ドロップしてみましょう（もしくは、図3のようにアドレスバーにファイルを保存した場所とファイル名を入力します）。

この「alert」という命令は、Web ブラウザに OK ボタンだけがあるウィンドウを表示させることができます。

今度は2行目を、「alert(5+2);」と書き換えて上書き保存後、再度ブラウザで表示させてみましょう。すると、計算結果が表示されます。

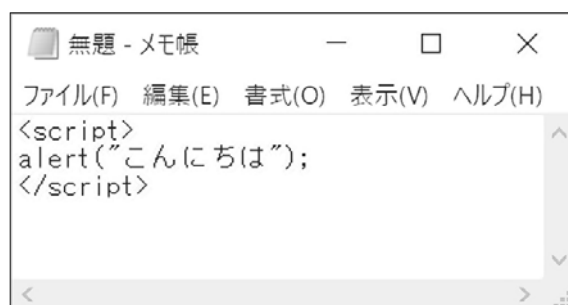


図1 JavaScriptのプログラム例



図2 HTMLでの保存



図3 実行結果

2 Makecode でブロック型と JavaScript の両方の言語を体験

JavaScript はできることが多く、さまざまところで利用される言語ですが、命令の種類やその組み合わせ方を調べる必要があるので制作に時間が掛かってしまいます。そこで、Makecode というプログラムの開発環境を使って、ブロック型で作ったプログラムがどのようにテキスト型のプログラムになるのかを見てみましょう。

ブラウザのアドレスバーに `https://makecode.microbit.org/` と入力しアクセスしましょう。

このページは、計測・制御のプログラミングとしても使用できる micro:bit (マイクロビット) のプログラムを制作するものです。画面の中のシミュレーターで実行結果を確認してみましょう。

このブロック型の言語は、Scratch とは異なりますが、一度 Scratch でプログラムを作ったことがあれば、同じようなイメージで制作できるので、すぐにプログラムを作ることができます。ビジュアル型の言語の中でも、ブロックをつなげる言語は、このようにすぐにプログラムの制作に取りかけられることが特徴ともいえます。

では、図4のようにブロックをつなげてみましょう。

ブロックをつなげ終わったら、「JavaScript」 というボタンを押してみましょう。すると図5のように、作成したプログラムが JavaScript に変換されて表示されます。

ここで書かれていることの意味を全部理解できなくても、中学生程度の英単語力があれば、ブロックで表現したことが、どのように文字で表されているのか、対比させることが可能です。

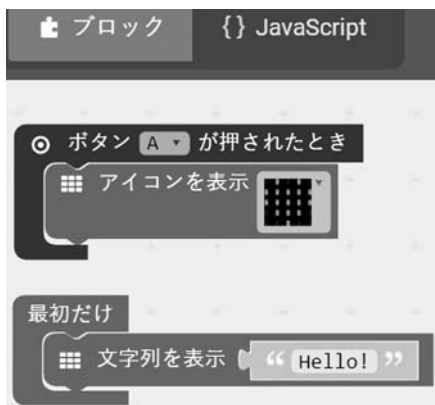


図4 ブロックでのプログラム

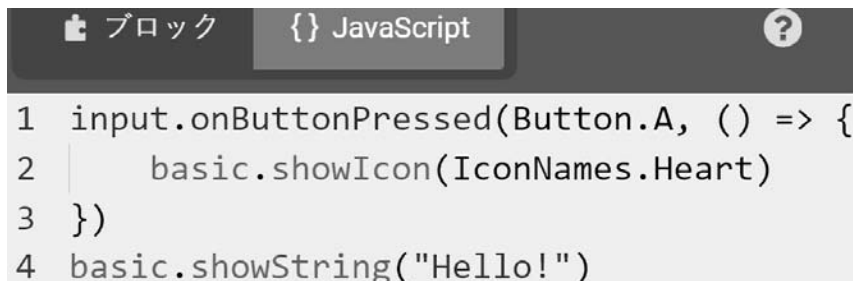


図5 ブロックでのプログラムを JavaScript で表現したもの

続けて、JavaScript の1行目から3行目をマウスでなぞってからコピー (Ctrl+c) をし、4行目の最後で Enter キーを押して表示された5行目に貼り付け (Ctrl+v) て、6行目の ShowIcon の部分を図6のように書き換えてみましょう。また4行目の「Hello!」を「Bye!」に、5行目の「Button.A」を「Button.B」に書き換えましょう。

今度は、「ブロック」のボタンを押して、ブロックによるプログラム表示に切り替えてみましょう。打ち間違いがなければ、追記・修正した通りに、ブロックが変更されて表示されます。

生徒の実態に応じて、こうしたテキスト型のプログラミングにも挑戦させると、プログラミングの概念に対する理解もさらに深まるでしょう。



図6 追記、修正した JavaScript

● 監 修

竹野 英敏 (広島工業大学)

● 執 筆

浅水 智也 (宮城教育大学附属中学校)

安藤 明伸 (宮城教育大学)

大村 基将 (大阪電気通信大学)

木村 浩之 (石巻市立牡鹿中学校)

紅林 秀治 (静岡大学)

藤原 英治 (石巻市立河南東中学校)

宮内 智 (さいたま市立大宮西中学校)

※五十音順

● 執筆協力

上野 耕史 (文部科学省)

広告

授業例で読み解く 新学習指導要領

竹野 英敏 編著

B5判/80ページ

定価:本体1,500円+税

- H29 告示の新学習指導要領における主な考え方や授業での取り上げ方を解説。
- 授業例を紹介しながら、実際にどのような授業をすればいいのか解説。



開隆堂出版株式会社

<http://www.kairyudo.co.jp/>

- 本 社 TEL 03-5684-6111
- 北海道支社 TEL 011-231-0403
- 東北支社 TEL 022-742-1213
- 名古屋支社 TEL 052-789-1741
- 大阪支社 TEL 06-6531-5782
- 九州支社 TEL 092-733-0174

AH

- 本書に掲載している商品名およびサービス名は各社の登録商標、商標、または商品名です。
 - ・ JavaScript...Oracle Corporation
 - ・ Microsoft MakeCode...Microsoft Corporation
 - ・ nekoboard2... 株式会社スイッチサイエンス <https://www.switch-science.com/>
 - ・ Pyonkee(ピョンキー)は、合同会社ソフトウメヤが、MIT メディア・ラボのScratch ソースコードライセンスにしたがって、Scratch をベースに開発した iPad 用のアプリケーションです。AppStore から無料でダウンロードできます。
<http://softumeya.com/pyonkee/ja/>
 - ・ Scratch は、MIT メディア・ラボのライフロング・キンダー・ガルテン・グループによって開発されました。 <http://scratch.mit.edu>
 - ・ Studuino... 株式会社アーテック <http://www.artec-kk.co.jp/artecrobo/edu/>
 - ・ プログラミング言語「ドリトル」... 兼宗進 <http://dolttle.eplang.jp/>

○本書に掲載されている URL は、2018 年 2 月 9 日現在に確認したものです。ページや内容は予告なく変更される場合があります。

○本書のプログラムは Windows10/7 にて動作確認を行っておりますが、全ての環境における動作を保証するものではありません。動作に関するお問い合わせにはお答えいたしかねますので、ご了承ください。