

PIECEプログラミングモジュール 編



ようこそ、プログラミングの世界へ

私たちの身の回りにはエアコンや炊飯器、スマートフォンや自動販売機や信号機など、便利な機械や大事な機械の多くには、センサーやボタンなどの入力装置、モーターやLEDなどの出力装置と、コンピューターが使われています。コンピューターは入力の情報をもとに出力装置を制御して動かしています。

コンピューターは超高速で動いて疲れないのですが、コンピューターはプログラムによって命令された通りしか動きません。機械に希望する動きをやってもらいたいときは、その動作を達成するまでの手順を細かく伝えなければいけません。

この手順を考えて、コンピューターに分かる言葉で伝えることがプログラミングです。プログラミングで、センサーやLEDを自由に制御できるようになると、もっと便利な装置がつけたり、もっと安全に動かせる機械がつけられるようになります。

PIECEプログラミングモジュールの中にも小さなコンピューターが使われていて、プログラムで動かすことができます。

PIECEでプログラミングを体験して自分だけの装置を発明してみましょう！

1. PaletteIDEを使おう



PaletteIDE を準備しよう Windows版

※ここではPaletteIDEのWindows版について説明しています。

※MacOS版の説明は、4ページ

以下のウェブサイトからWindows版のPaletteIDEをダウンロードします。

<https://www.elekit.co.jp/software/SW-1017>

ダウンロードしたファイルはzip形式で圧縮されていますので展開します。

展開すると「palette_V**1_5_0**」という名前のフォルダができていることを確認しておきましょう。

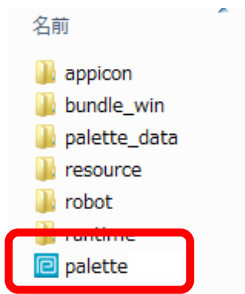
※ v以下の数字の部分はバージョンにより異なります。

※ 圧縮されたファイルを展開するためにダブルクリックしてもパソコンの設定によっては、フォルダーの中身を表示するだけで展開されない場合があります。確実に展開するために、**圧縮されたファイルの上で右クリックして「すべてを展開」**を選んでください。

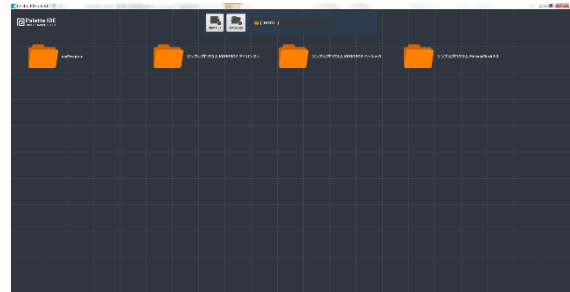


PaletteIDEを起動する Windows版

ダウンロード後に展開したフォルダーの中にある「palette」をダブルクリックして「PaletteIDE」を起動します。



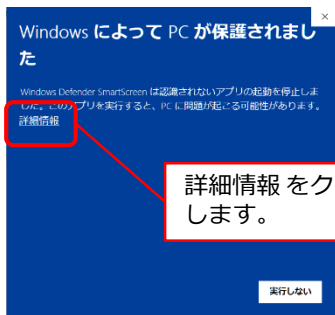
ダブルクリックすると



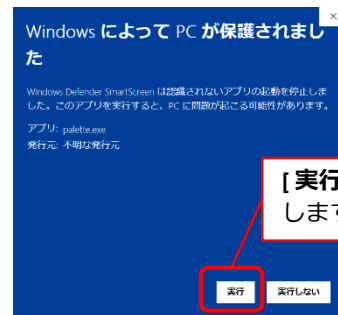
PaletteIDEが起動し、プログラム
ム一覧画面が表示されます

はじめて「PaletteIDE」を起動するときに、不明な発行元といったメッセージが出る場合があります。その場合は以下のように操作してください。

Windows10の場合



詳細情報をクリック
します。



[実行]をクリック
します。



PaletteIDE を準備しよう MacOS版

※ここではPaletteIDEのMacOS版について説明しています。

※Windows版の説明は、2ページ

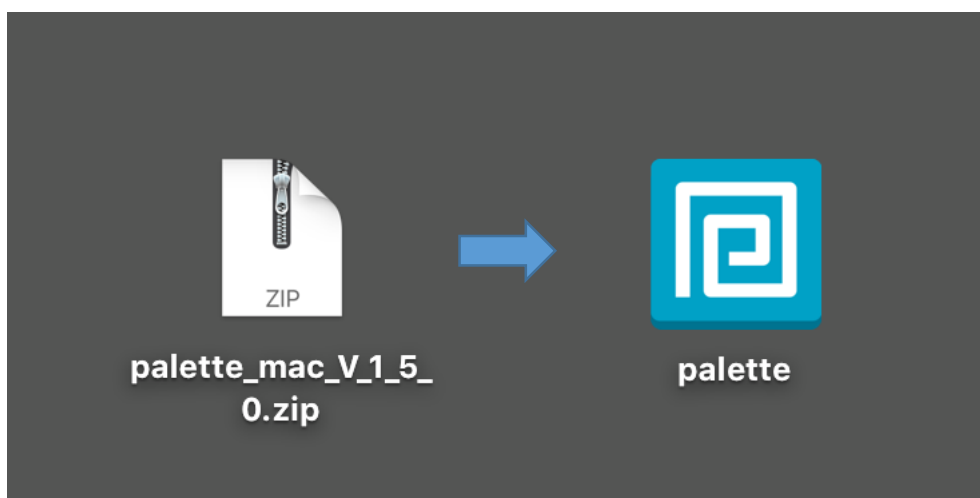
以下のウェブサイトからMacOS版のPaletteIDEをダウンロードします。

<https://www.elekit.co.jp/software/SW-1017>

ダウンロードしたファイルはzip形式で圧縮されていますので展開します。

※Safariでzip形式のファイルをダウンロードした場合、自動的に展開されます (デフォルト設定のとき)。

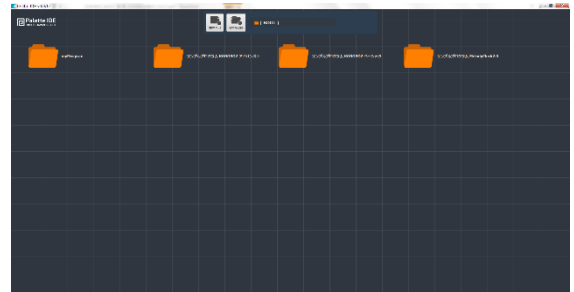
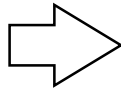
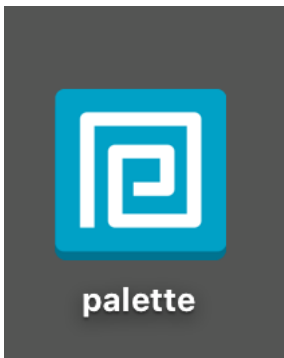
展開すると「palette」という名前のアプリケーションができあがります。





PaletteIDEを起動する MacOS版

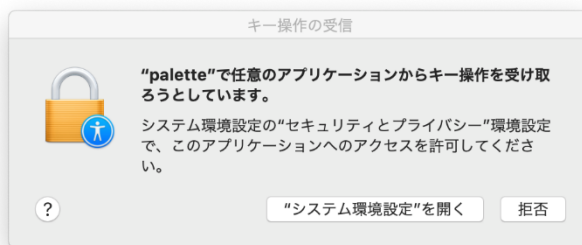
展開してできあがった「palette」をダブルタップして「PaletteIDE」を起動します。



PaletteIDEが起動し、プログラム一覧画面が表示されます

MacOS 10.15 Catalinaで使用する場合の注意

MacOS 10.15 Catalinaでは、PaletteIDEの初回起動時に、次のようなダイアログが出ます。



「システム環境設定を開く」を押して設定画面に進み、下図のように、Paletteの入力監視を許可(チェックボックスをON)してください。



この設定を行わないと、ロボットへのプログラム転送ができません。

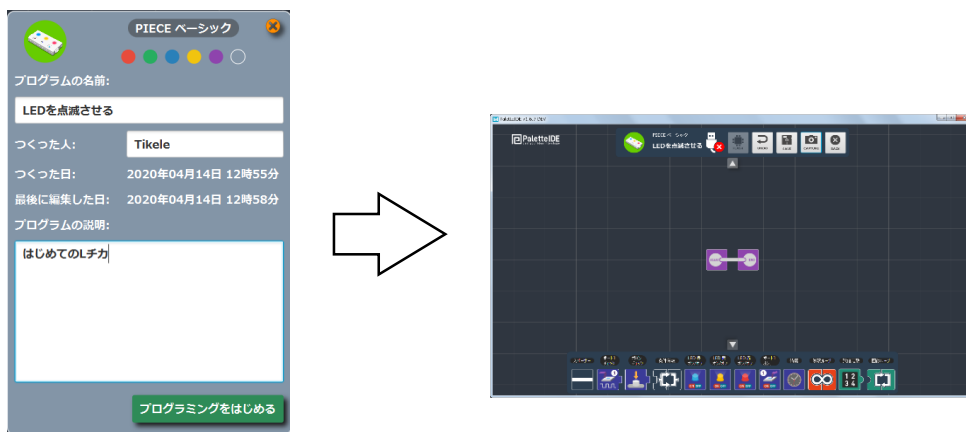
また、MacOS 10.15 Catalina以前のMacOSでは、この設定は必要ありません。

プログラムの新規作成

プログラムを新しく作るときは、画面上部の「NEW FILE」のボタンをクリックし、選択ダイアログで**PIECE ベーシック**を選びます。



選択ダイアログのOKをクリックすると**プログラム詳細ダイアログ**が出てきます。
ここでは、プログラムの名前やつくった人の名前を入力します。「プログラムの説明」のところには、どんなプログラムを作りたいかなど、プログラムの内容が分かるようなメモを書くと良いでしょう。



プログラム編集画面



①ワークスペース

この場所にプログラムを作成します。

スタートとエンドのアイコンはあらかじめ配置されています。

②ドック

プログラムで使用する命令アイコンが並んでいます。

LEDをオンオフするアイコンや、プログラミングモジュールの入力や出力を制御するためのアイコン、分岐やループのアイコンなどが用意されています。

③スイッチボード

プログラムの情報を表示したり、プログラムの保存や、プログラムの書込みなどの操作をするためのアイコンが並んでいます。

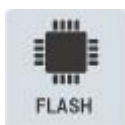


アイコンの説明 : スイッチボードのアイコン



USB チェック

プログラミングモジュールと正しく接続されているときに緑色のチェックマークが表示されます



フラッシュ

プログラムをプログラミングモジュールに書き込みます



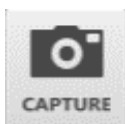
アンドウ

プログラム作成操作を1つ前の状態に戻します
グレイ表示されているときは利用できません



セーブ

作成したプログラムを保存します



キャプチャー

現在画面に表示しているプログラム部分を PNG形式の画像で保存します。背景の色は透明になります。
画像はPalette.exeが置かれているフォルダーの中に保存されます。



バック

プログラム編集画面を閉じて、プログラム一覧画面に戻ります



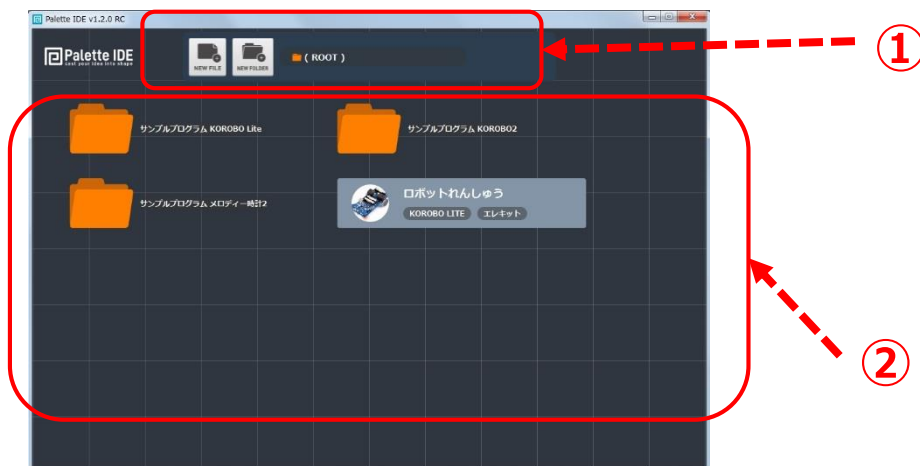
アイコンの説明 : ドックのアイコン

ドックのアイコンの詳しい説明は、「14. アイコンの説明」で説明しています。次ページ以降の「やってみよう」でプログラミングに慣れて、オリジナルのプログラムを作るときに活用してください。



プログラマー一覧画面の説明

プログラマー一覧画面



①スイッチボード

フォルダーやファイルを操作するアイコンが表示されています。



NEW FILE : 新しくプログラムを作成するときにクリックします。



NEW FOLDER : 新しくフォルダーを作成するときにクリックします。

②ワークスペース

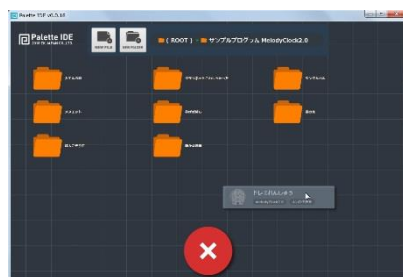
この場所に作成したプログラムやフォルダーが表示されます。



プログラムやフォルダーを削除する

削除したいプログラムやフォルダーをクリックしたまま、画面の下の方に動かし、の上で離すと削除されます。

プログラムやフォルダーを下に移動するとのアイコンが現れます。



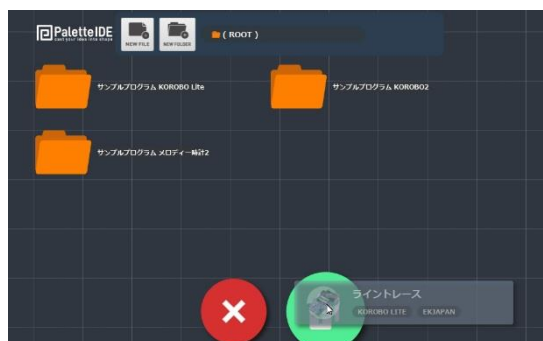


プログラムのコピー

プログラムを作ってうまく動作したときは、プログラムのファイルをコピーして保存しておくの良いかもしれません。試行錯誤しているときに「昨日のプログラムに戻したいけど上書き保存してしまっていて戻れない…」となってしまう前にコピーをしておきましょう。



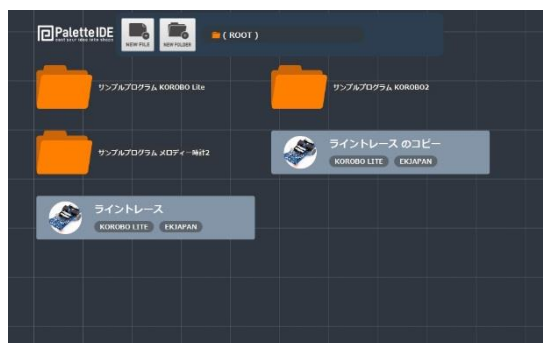
ファイルをドラッグすると画面下に
削除  とコピー  のマークが現れる



コピー  マークにファイルをドロップ
するとファイルがコピーされます



コピーするファイルの名前と説明を
記入します

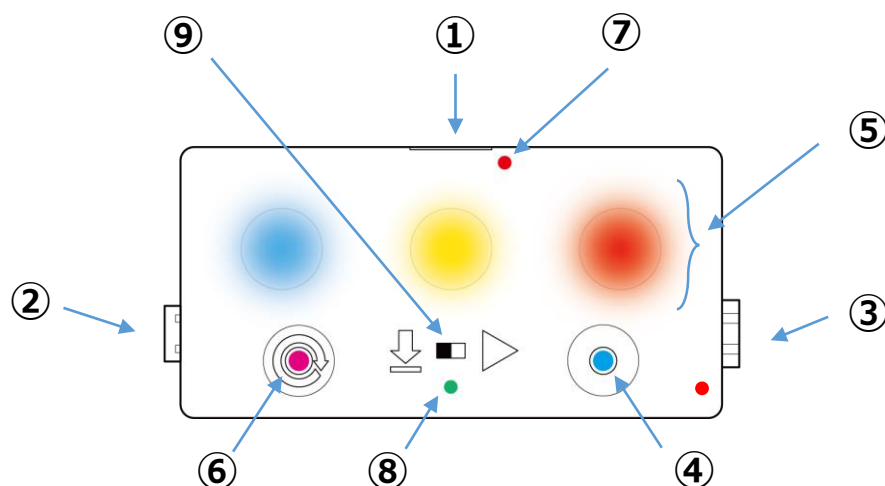


このようにファイルがコピーされます

2. プログラミングモジュールを知ろう



各部のなまえ



①	USB	付属のUSBケーブルをつなぎます。パソコンとUSBをつなぐことでプログラミングモジュールの電源になります。PaletteIDEで作成したプログラムを書き込むときにも利用されます。
②	入力1	別売のPIECEのオレンジモジュール(入力)や黄色モジュール(処理/論理)をつなぎます。
③	出力1	別売のPIECEの緑色モジュール(出力)や黄色モジュール(処理/論理)をつなぎます。プログラミングモジュールがオン信号を出している間はコネクタ近くの赤色LEDが点灯します。
④	ボタン	プログラムで自由に使用できるボタンです。
⑤	LED赤/黄/青	プログラムで自由に点灯/消灯できるLEDです。赤色、黄色、青色の3つがあります。
⑥	リセットボタン	ボタンを押すとプログラミングモジュールがリセットされ、プログラムを最初から実行します。
⑦	ステータスLED	パソコンと接続中や、リセット直後、プログラム実行が終了したときに点灯・点滅するLEDです。
⑧	電源表示LED	プログラミングモジュールに電源が接続されているときに点灯します。
⑨	モード切替スイッチ	動作モードを切り替えます。右にスライドすると「実行モード」左にすると「書き込みモード」になります。 実行モード : プログラムを実行します。 書き込みモード : PaletteIDEからプログラムを書き込みできます。

3. プログラムで動かしてみよう

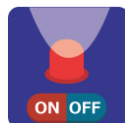


ここで使うアイコン



スタート

このアイコンからプログラムが始まります。



LED赤

プログラミングモジュールのLED赤をON/OFFします。



エンド

全ての動作を停止し、プログラムを終了します。



待機

指定された秒数だけプログラムを一時停止します。



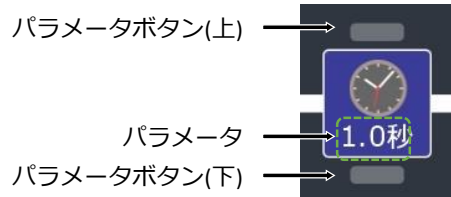
LED赤を1秒間点灯するプログラム

命令アイコンを画面下のドックからドラッグして、スタートとエンドの間の置きたい場所で離すとプログラム上にアイコンが配置されます。



下図のような「LED赤を1秒間点灯する」プログラムをつくります。

待機アイコンの秒数は1秒に設定しましょう。アイコンの上下についているパラメータボタン をクリックすると秒数が変わります。



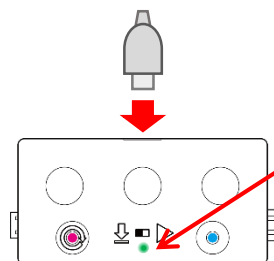
(例3-1) LED赤を1秒間点灯



プログラムを書き込もう

プログラムができれば、プログラムを書き込みましょう。

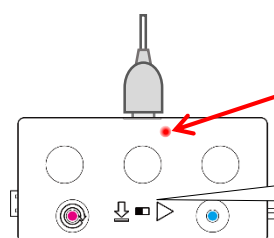
- ①プログラミングモジュールとパソコンをUSBケーブルで接続します。



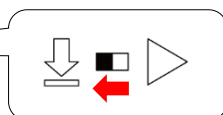
電源表示LEDが点灯します。

充電専用USBケーブルでは書き込みできません。
付属のUSBケーブルを使います。

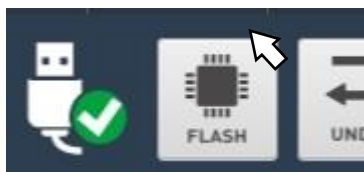
- ②プログラミングモジュールのモード切替スイッチを左側にスライドして書き込みモードにします。



ステータスLEDが点灯することを確認します。



- ③FLASH(フラッシュ)ボタンをクリックしてプログラムを書き込みます。

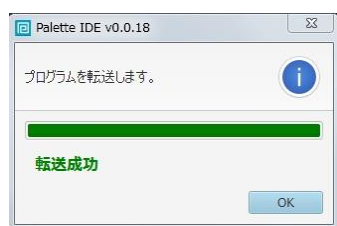


FLASHボタンが明るいときにプログラムを書き込むことができます。



FLASHボタンが暗いときはプログラムを書き込むことができません。 ケーブルの接続とモード切替スイッチの位置を確認し、**プログラミングモジュールのリセットスイッチを一回押してみます。**

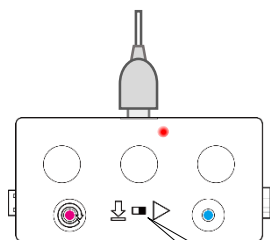
- ④画面に**転送成功**と表示されれば書き込み完了です。



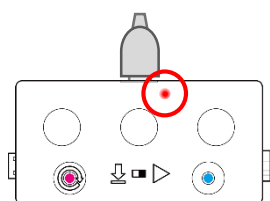
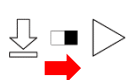


動作チェックをしよう

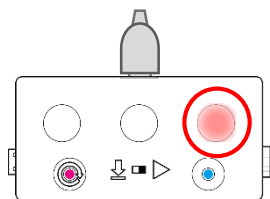
書き込んだプログラムを実行してみましょう。



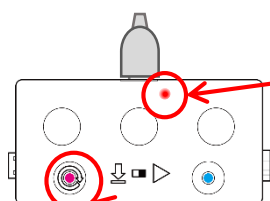
モード切替スイッチを右側に
スライドして、実行モードに
します。



実行モードにするとステータス
LED赤が数回すばやく点滅したあ
とに、プログラムを実行します。



1 秒間LED赤が点灯！



プログラムが終了すると
ステータスLED赤がゆっくり点
滅します。

もう一度プログラムを実行するときは、
リセットスイッチを一回押します。

4. 順次処理：順番に実行する

プログラムがスタートすると、命令された順番どおりに命令を実行していきます。このことを順次処理といいます。PaletteIDEでは、スタートアイコンから始まって、線でつながっているアイコンを順番に処理していきます。ここでは、PaletteIDEの順次処理をやってみましょう。



この章で使うアイコン



LED黄

プログラミングモジュールのLED黄をON/OFFします。
パラメータボタンを押してONとOFFを切り替えることができます。



LED青

プログラミングモジュールのLED青をON/OFFします。
パラメータボタンを押してONとOFFを切り替えることができます。



5 秒間LED赤を点灯する

下図のような「5秒間LED赤を点灯」するプログラムを作ります。待機アイコンのパラメータを5.0秒に設定しましょう。



パラメータボタンを
押してOFFにします

(例4-1) 5秒間LED赤を点灯



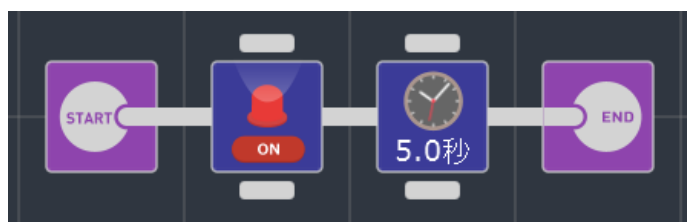
動作チェックをしよう

- ☐ START → LED赤(ON) → 待機 (5秒間) → END
の動きはうまくできましたか？



ワンポイント

下図のようなプログラムを作って、動きを確かめてみましょう。



(例4-2) LEDをOFFしないで終了

さっき作ったプログラムとは少し違い、LED赤をOFFにする命令がないのに全く同じ動きかたをします。

これは「END」アイコンが「すべての出力をOFFにしてプログラムを終了する」というアイコンなので、LED赤をわざわざOFFにしなくても勝手にOFFになってしまうためです。どちらのプログラムでも間違いではありませんが、作成するプログラムが大きく複雑になってくると、ちゃんとプログラムの中にOFFにする処理を書いておいたほうが後で見直しをするときにミスを発見しやすく、他の人が見たときにも分かりやすくなります。



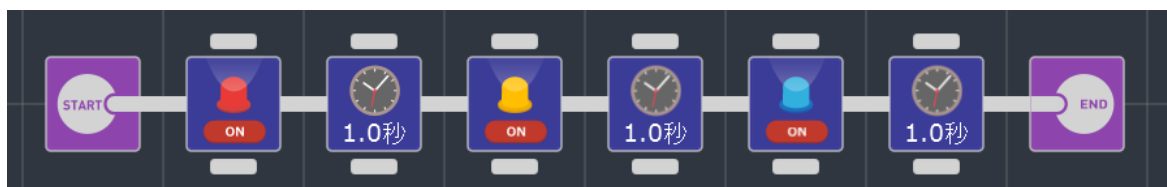
LEDを順番に点灯する

3つのLEDが順番に点灯するプログラムをつくってみましょう。

LED赤 → LED黄 → LED青 の順番で点灯させます。

順番に点灯していくように見せるために、LEDが点灯して次のLEDが点灯するまでに1秒間待つことにします。

下図のようなプログラムを作ります。待機アイコンのパラメータを1.0秒に設定しましょう。



(例4-3) LEDを順番に点灯



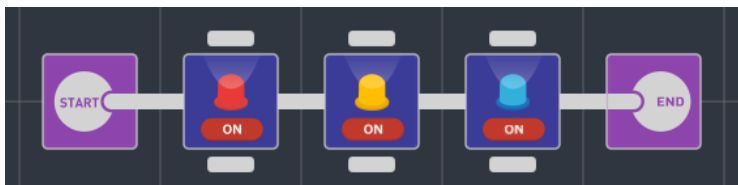
動作チェックをしよう

☐ LEDは順番に点灯しましたか？



ワンポイント

プログラムはアイコンを順番に処理していくので、下図のように待機アイコンを使わなくても、LEDは順番に点灯しそうですね。



でもプログラムを書き込んで動かしてみると、ほんの一瞬全部のLEDが点灯するだけです。その理由は、プログラムはちゃんとスタートして、LEDを点灯させるアイコンを実行しているのですが、すぐに最後のENDアイコンに到着してしまい、プログラムが終了してLEDがOFFになってしまうからです。

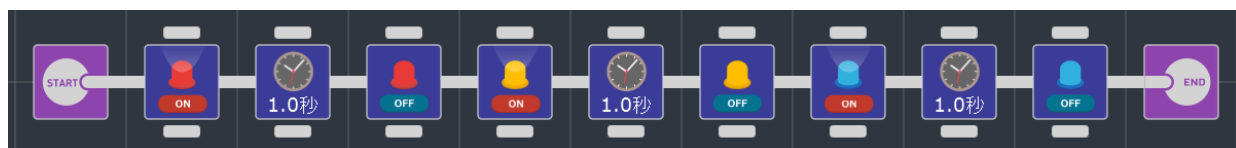
このようにプログラムに期待する動きと実際の動きが違う場合は、プログラムにどこがおかしいところがないか、考え方が間違っていないか、アイコンの動作を間違えて覚えていないかなどを確認をするようにしましょう。



LEDを1つずつ順番に点灯する

3つのLEDが順番に1つずつ点灯するプログラムをつくってみましょう。
さっき作ったプログラムでは最後に3つ全部点灯しましたが、今度は、1つ点灯させたらそのLEDを消して、次のLEDを点灯させるようにします。

下図のようなプログラムを作ります。待機アイコンのパラメータを1.0秒に設定しましょう。



(例4-4) LEDを1つずつ順番に点灯



動作チェックをしよう

- ☐ LEDが1つ点灯して消えて、次のLEDが点灯というように順番に点灯しましたか？

5. くり返し処理：同じ動きをくり返そう



同じ動きのアイコンをたくさん並べる？

例えば、(例4-4)で作った「LEDを1つずつ順番に点灯する」を5回繰り返すプログラムを作ろうとしたとき、どうすれば実現できるでしょうか？



「LEDを1つずつ順番に点灯する」を5回繰り返す

これで動かすことができます。でもこのプログラム、同じことを繰り返すだけの単純な動きなのにとっても長くて読みにくいですね。それに「繰り返す回数を100回にしてください」なんて言われたらもう大変ですね。

この章ではこういった単純な動作を繰り返し行うときに便利な「くり返し処理」について学んでいきます。



この章で使うアイコン



回数ループ

囲まれた部分のプログラムを数値定数で指定した回数だけ繰り返します。



数値定数

回数ループなどに使う数値を指定します。
パラメータボタンによって数値を増減できます。



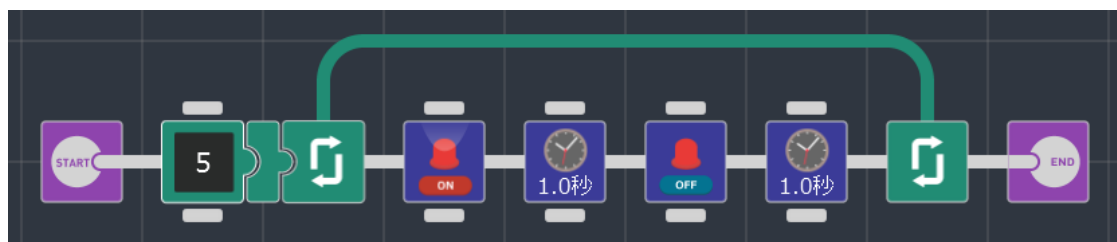
無限ループ

囲まれた部分のプログラムを無限に繰り返します。



「1秒間点灯 → 1秒間消灯」を5回くり返す

「1秒間点灯→1秒間消灯」を5回くり返すプログラムをつくります。
くり返したい動作を「回数ループ」のはじめと終わりのアイコンの間に置き、
くり返す回数を「数値定数」で設定します。数値定数アイコンのパラメータボ
タンの上をクリックして値を5にします。



(例5-1) 「1秒間点灯→1秒間消灯」を5回くり返す



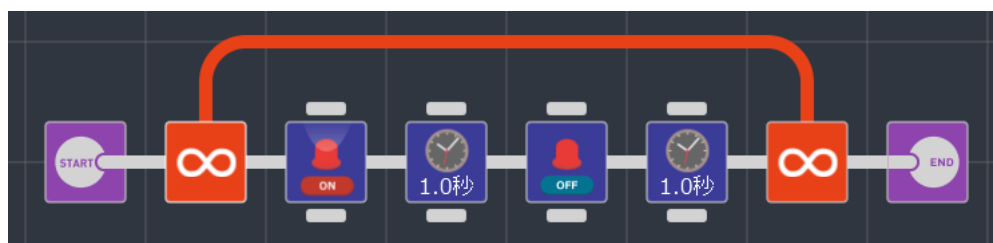
動作チェックをしよう

☐ 「LED赤(ON) →待機(1秒間)→LED赤(OFF) → 待機(1秒間)」を5回くり返して動きますか？



「1秒間点灯 → 1秒間消灯」をずっとくり返す

自動で装置を動かす場合、プログラムが終了することなく、ずっと動き続けてほしいことがほとんどです。「無限ループ」を使って、さっき作ったプログラムをずっとくり返してみましょう。



(例5-2) 無限ループ



動作チェックをしよう

☐ 「LED赤(ON) →待機(1秒間)→LED赤(OFF) → 待機(1秒間)」をずっとくり返して動きますか？

6. 条件分岐：条件によって動きをかえよう



条件で動きをかえる

人は道を歩いているときに、目の前に急にボールがとびだしてきたら、すぐに止まることができますよね。これは「ボールがとびだしてきたら」止まる、「ボールがないときは」歩く。というように条件によって動きをかえているということです。

このように条件によって動きをかえることを、条件分岐といいます。ここでは条件分岐のプログラムを学びましょう。



この章で使うアイコン



条件分岐

このアイコンの前につながっているアイコンの状態分で岐します。アイコンの状態が[○]なら分岐の○側、アイコンの状態が[×]なら分岐の×側のプログラムを実行します。



ボタン

プログラミングモジュールについているボタンの状態をチェックするときに使います。ボタンが押されているときは[○]、押されていないときは[×]になります。

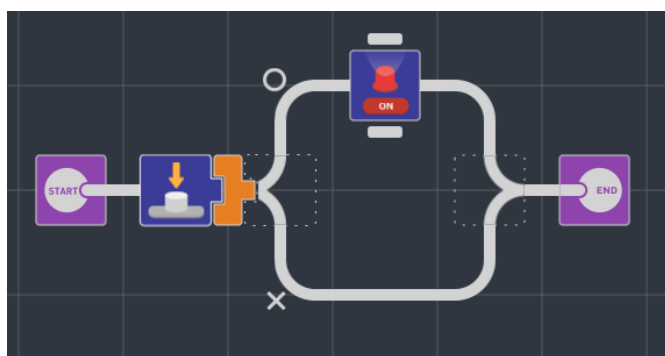


ボタンを押すとLED赤が点灯するプログラム（1）

「ボタンを押すとLED赤が点灯する」プログラムをつくります。

ボタンが押されているかどうかをチェックするためには、ボタンアイコンを使います。ボタンアイコンはプログラミングモジュールのボタンが押されていると、次につながっているアイコンに[○]を渡し、ボタンが押されていないときは[×]を渡します。

[○]と[×]で動きを変えるには条件分岐アイコンを使います。条件分岐アイコンは、前につながっているアイコンから[○]を受け取ったら上の線へ、[×]を受け取ったら下の線へ進みます。



(例6-1) ボタンを押すとLED赤が点灯(1)



動作チェックをしよう

☐ 期待した通りに動きましたか？

実はこのプログラムは**期待した通りに動きません**。電源をONにしてプログラムを実行しても、すぐにステータスLEDが点滅するだけで、ボタンを押しても何の変化もありません。なぜでしょうか。

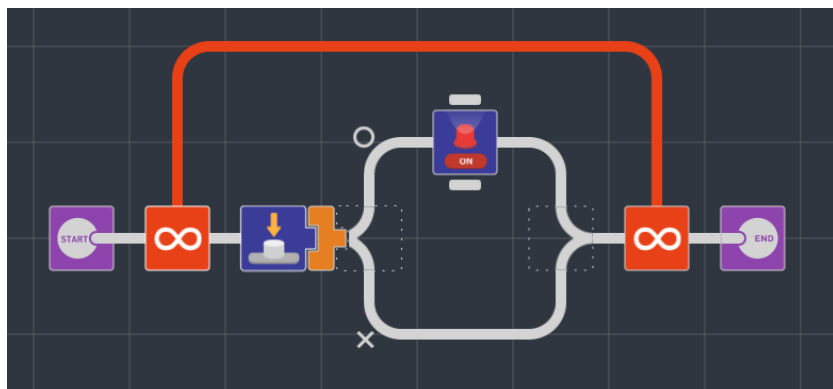
これは、プログラムがスタートすると、すぐにボタンの状態をチェックしてどちらかの道を進んで、すぐにENDに到着してしまうからです。ほんの一瞬で終わります。つまりプログラムが終わった後にボタンを押している訳です。

ずっとボタンの状態をチェックするためには、プログラムがずっと動き続けなといけません。**ずっと動くプログラム**を作るときは**無限ループ**を使います。



ボタンを押すとLED赤が点灯するプログラム（2）

「ボタンを押すとLED赤が点灯する」を無限ループで囲んだプログラムをつくれます。



(例6-2) ボタンを押すとLED赤が点灯(2)



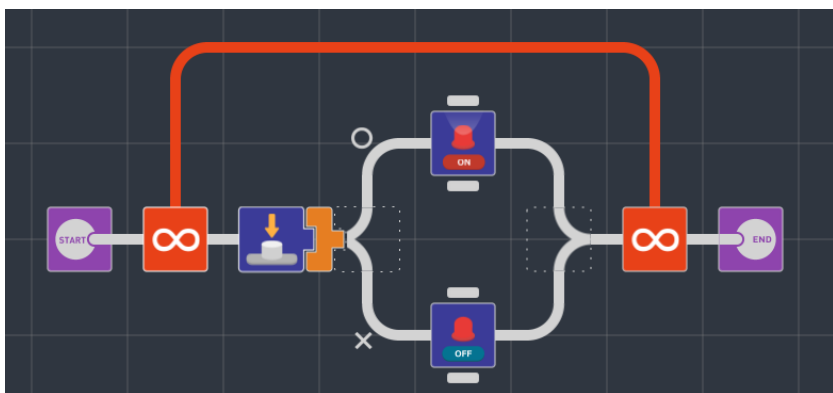
動作チェックをしよう

☐ ボタンを押すとLED赤が点灯しますか？



やってみよう

さっきのプログラムは、ボタンを押すとLED赤が点灯しましたが、今度は、ボタンを押している間はLED赤が点灯して、押していないときはLED赤が消えているプログラムを作ってみましょう。



(例6-3)ボタンを押すと点灯、離すと消灯



信号機を作ってみよう

身近にある、押しボタン式の歩行者用信号機の動きを再現してみましょう。信号機の動きをよく観察してどうやって光っているのか詳しく調べましょう。

- ・ ボタンを押すまで信号機は何色？
- ・ ボタンを押してから何秒後に青に変わる？
- ・ 青から赤に変わるときはどんな光り方をしている？



こんな信号機を作ってみましょう

- ・ 普段は(ボタンが押されていないとき)は赤色が点灯。
- ・ ボタンを押すと、「しばらくお待ちください」が光る。
- ・ ボタンを押してから3秒後に青色が点灯する。
- ・ 青色が6秒間点灯して、点滅を始める。
- ・ 青色は点滅を5回繰り返して消えてから、赤色が点灯する。

※「しばらくお待ちください」ランプはないので、
ここではLED黄を点灯させることにします。



今まで学んだ、順序処理、くり返し処理、条件分岐の組み合わせでプログラムを作ってみましょう！



考えてみよう

さっき作ったプログラムには「しばらくお待ちください」が点灯する機能を入れましたが、この機能は本当に必要なのでしょうか？ないとどうなるのでしょうか？「しばらくお待ちください」機能をなくしたプログラムを作って実際に使ってみましょう。

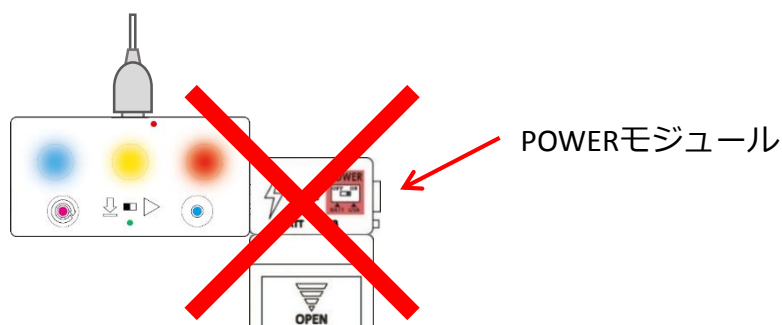
実際に使ってみると、ボタンを押したかどうか分からなくて何度もボタンを押してしまいます。使いづらいですね。一見必要なさそうな機能でも、使いやすくするためには大事な機能であることがわかりますね。

8. PIECEのモジュールつないでみよう

プログラミングモジュールの入力端子や出力端子に、PIECEのモジュールをつないでつかうことができます。ここではセットに入っている、LIGHTセンサーモジュールを入力1につないだときの使い方を説明しています。

別売の他のPIECEセットのモジュールの利用する場合は、以下の点に注意してお持ちのモジュールを接続してください。

PIECEプログラミングモジュールがUSBの電源で動いているときには、絶対に**PIECEの電源モジュール(POWERモジュール)**をつながないでください。発熱の恐れがあり、故障の原因となります。



準備しよう

プログラミングモジュールに下図のようにLIGHTセンサーモジュールをつないでください。



9. プログラミングモジュールの入力をつかってみよう



ここで使うアイコン



ポート1チェック

プログラミングモジュールの入力1に接続したPIECEモジュールの状態を調べるときに使います。

PIECEモジュールがオン信号を出しているときは[○]に、信号をだしていないときは[×]になります。



ポートとは？

ポート（PORT）とは、日本語に訳すと「港（みなと）」です。

港は、船が街に出たり入ったりするために必要な場所です。電気の世界でもこれになぞらえて、電気信号を出したり受け取ったりする端子のことをポートといいます。受け取り専用の端子のことを入力ポート、信号を出す専用の端子を出力ポートとよんだりします。

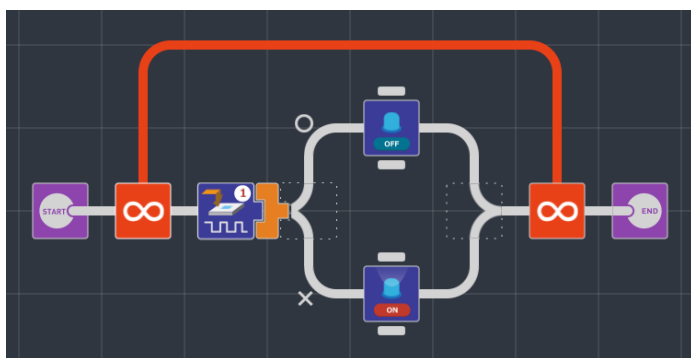


暗いときにはLED青が点灯する

「暗いときにはLED青が点灯する」プログラムを作ってみましょう。考えるときのポイントは以下の点です。

- ・ 明るさをチェックしたいので「LIGHT」センサーを使う。
- ・ “暗いとき”ということは“明るくないとき”ということ。
- ・ LIGHTセンサーは入力1につながっている。
- ・ 明るいときと暗いときで動きがかわるので、条件分岐を使う。
- ・ ずっと動き続けて欲しいので無限ループを使う。

プログラムを作成すると下図のようになります。



(例9-1)暗いときにはLED青が点灯する

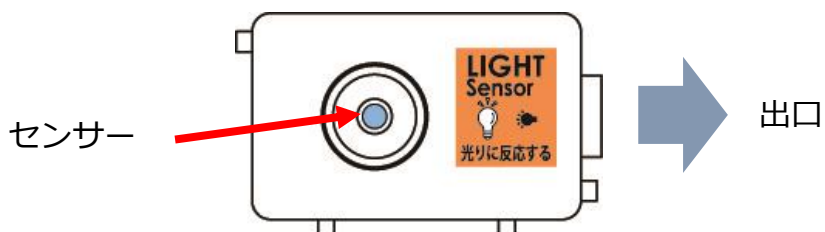


動作チェックをしよう

- ☐ センサーの周囲を暗くすると、LED青が点灯しましたか？

知っておこう：

LIGHT センサーモジュール



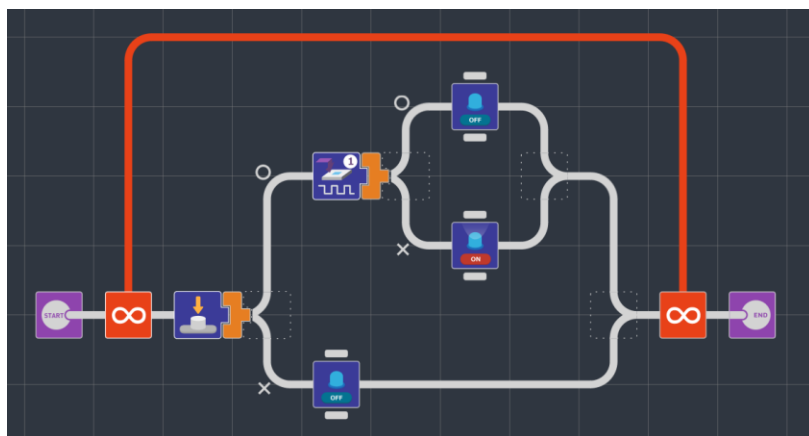
センサーが光を感じると出口からオン信号が出ます。



ボタンを押すと暗いときにだけLED青が点灯する

入力を2つ使って、条件が2つになった場合のプログラムを作ってみましょう。
考える方のポイントは以下ようになります。

- ・ ボタンが押されて、かつ、明るくないとき(暗いとき) は点灯
- ・ それ以外のときは消灯する



(例9-2) ボタンを押すと暗いときにだけLED青が点灯



動作チェックをしよう

- ☐ 暗いときにボタンを押すとLED青が点灯し、ボタンから手を離すと消灯し、明るいときはボタンを押しても、押していなくてもLED青が消灯したままになりましたか？



プログラムを観察してみよう

(例9-1)で光センサーのプログラムを作ったときは、光センサーの条件分岐で○と×の2通りだけ動きを設定しましたが、ここでは「光センサーの条件分岐」が「ボタンでの条件分岐」の中に入っています。このように条件分岐の中にまた条件分岐を入れている構造のことを「**ネスト (入れ子)**」といいます。ネストを使うことで条件分岐を3通り以上にすることができます。

10. プログラミングモジュールの出力をつかってみよう

ここでは別売の、「いろんなセンサーでライトをつけようセット」(ZZ-AC01)に用意してある出力モジュール(TORCH)を使って説明しています。

他のセットをお持ちの場合は、お手元の出力モジュールに合わせて文章を読みかえるなどしてご利用くださいますようお願いいたします。



ここで使うアイコン



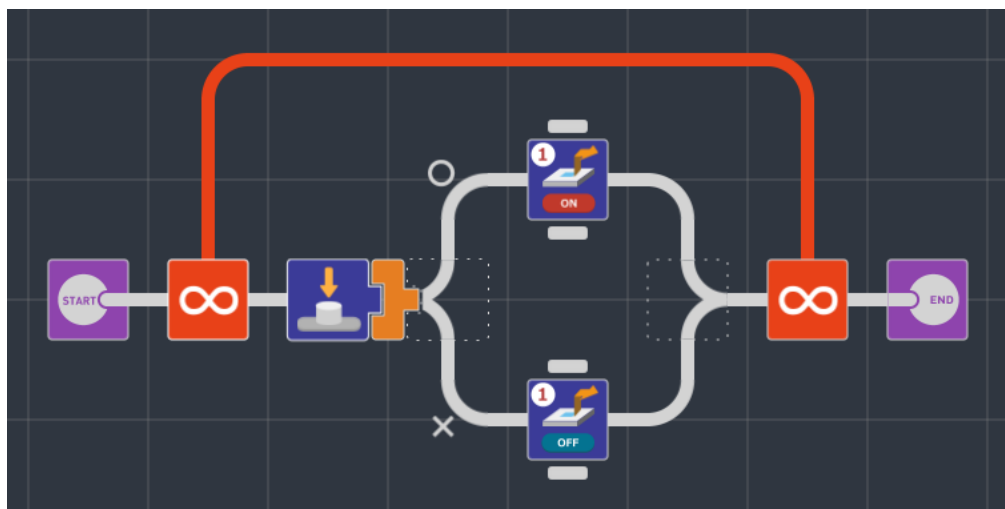
ポート1出力

プログラミングモジュールの出力1をオン／オフします。パラメータボタンを押してオンとオフを切り替えます。



ボタンを押している間TORCHが光る

プログラムを作成すると下図のようになります。



(例10-1) ボタンを押すとTORCHが光る



動作チェックをしよう

- ☐ ボタンを押すとTORCHが光り、離すと消灯しましたか？

11. いろんな装置をつくってみよう

プログラミングモジュールのLEDや入力1/出力1を、順序処理、くり返し処理、条件分岐のプログラムで動かす方法を学びました。これらを組み合わせて、身の回りにあったら便利になるもの、安全になるものを作ってみましょう。プログラムをつくって、使ってみて、使いにくくないか、もっと省エネできないかなどを考えてより便利な装置にすることに挑戦してみましょう。

玄関が暗くて危ないなあ・・・

暗くなると自動で点灯するライト

↓もっと便利にしてみる

暗くなると自動で点灯して、ボタンを押すと消灯するライト



部屋の電気を消してベッドまで行くときに暗いなあ・・・

暗くなると1分間点灯してから消えるライト

↓もっと楽しく

暗くなると1分間、ライトやLED赤黄青が賑やかに点滅する



いろんな課題をみつけて、自分の装置をつくることに挑戦してみましょう。

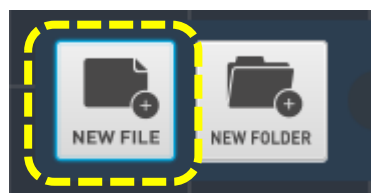
1 2. 発展

PaletteIDEにはプログラミングモジュールの使い方の幅を広げるための命令のアイコンが準備されたアドバンストモードを用意しています。

ここでは、それらの機能を紹介しています。こんな機能もあるんだということを知っておくと、これからいろんなプログラムを作成するときに、ちょうどよく便利に使えるかもしれません。

アドバンストモードの始め方

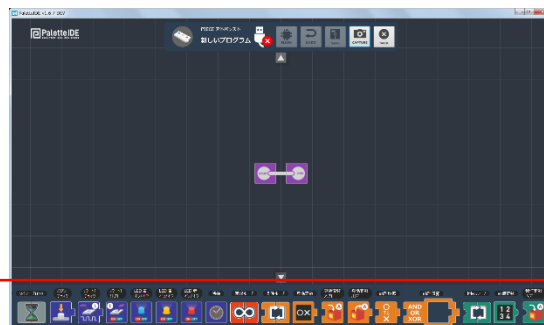
「NEW FILE」のボタンをクリック。



選択ダイアログで
「PIECE アドバンスト」
を選んで「OK」ボタンをクリック



「プログラミングをはじめる」ボタンをクリックすると、プログラム編集画面が開きます。



使えるアイコンの数が増えています



タイマーをつかう

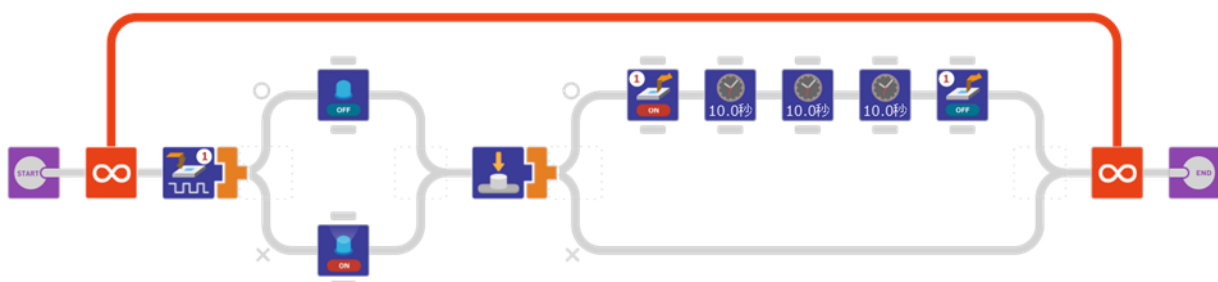
目覚まし時計のように時間をセットして、セットした時間になったら自動で決められた処理を行ってくれる機能がタイマーです。タイマーをセットしておけば、他に何かしていても、時間が来たらTORCHを点灯させるといったことができ便利です。

例えば、以下のような2つの動作をするプログラムを作ってみましょう。

①暗くなったらLED青を点灯する。

②ボタンを押したら30秒間TORCHを点灯する。

まずは、タイマーを**使わないで**プログラムを作ってみます。



(例12-1) 発展：タイマー使わない

このプログラムを実際に動かして、以下の順番でチェックしてみましょう。

- (1) LIGHTセンサー(ポート1チェック)を暗くしたり明るくしたりすると、LED青が点いたり消えたりします。
- (2) 次にボタンを押すとTORCH(ポート1出力)が点灯します。
- (3) TORCHが点灯しているときに(1)と同じようにLIGHTセンサーを暗くしたり明るくしたりしてみますが、今度はLED青は点灯したまま、または消灯したまま変化しません。
- (4) 30秒経過してTORCHが消えた後に、もう一度(1)と同じことをしてみると、今度はLED青が点いたり消えたりします。

なんだか期待している動作になっていません。

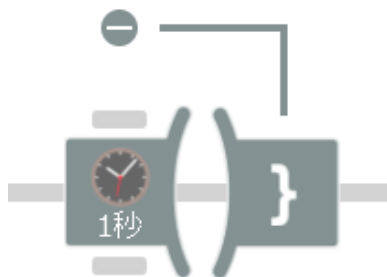
このように動作するのは、「待機アイコン」が原因です。

待機アイコンは全ての動作をそのままにして時間を数えます。つまり待機アイコン処理中は他に何もしないので、ポート1入力につないであるセンサーの状態もチェックしないのです。

時間を数えながら同時にセンサーの値もチェックしたいといったときにはタイマーを使います。

タイマーのつかい方

ドックにあるタイマーブロックアイコン（）をワークスペースに置くと下図のようなカッコで囲まれた形になります。

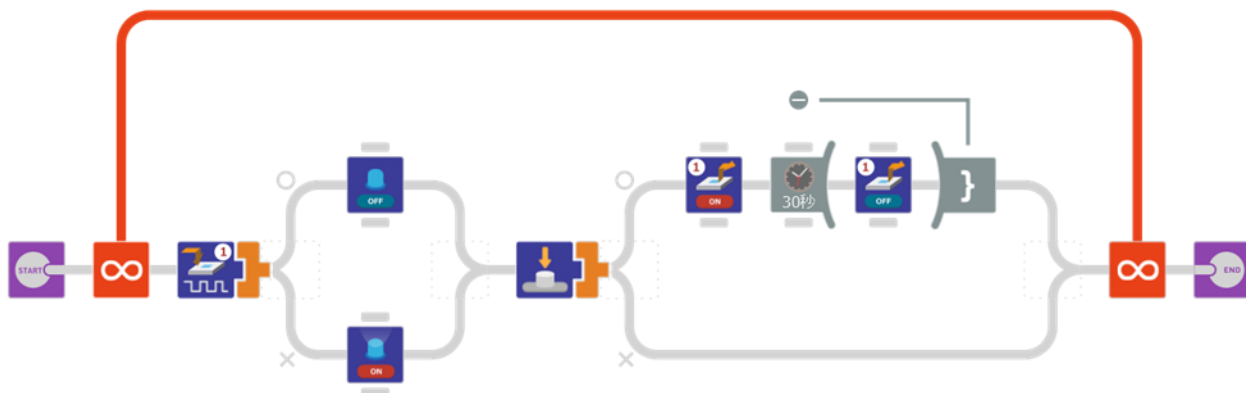


このカッコの中に時間になったら実行したいアイコンを置きます。
タイマーの時間はパラメータボタンを押して設定します。

タイマーを使ったプログラム例

さっきと同じ動作を、今度は**タイマーを使って**プログラムを作ってみましょう。

- ①暗くなったらLED青を点灯する。
- ②ボタンを押したら30秒間TORCHを点灯する。



(例12-2) 発展：タイマーを使う

今度は期待した通りに、TORCHが点灯しているときでも、明るさセンサーに反応してLED青が点いたり消えたりするようになります。

知っておこう 1 :

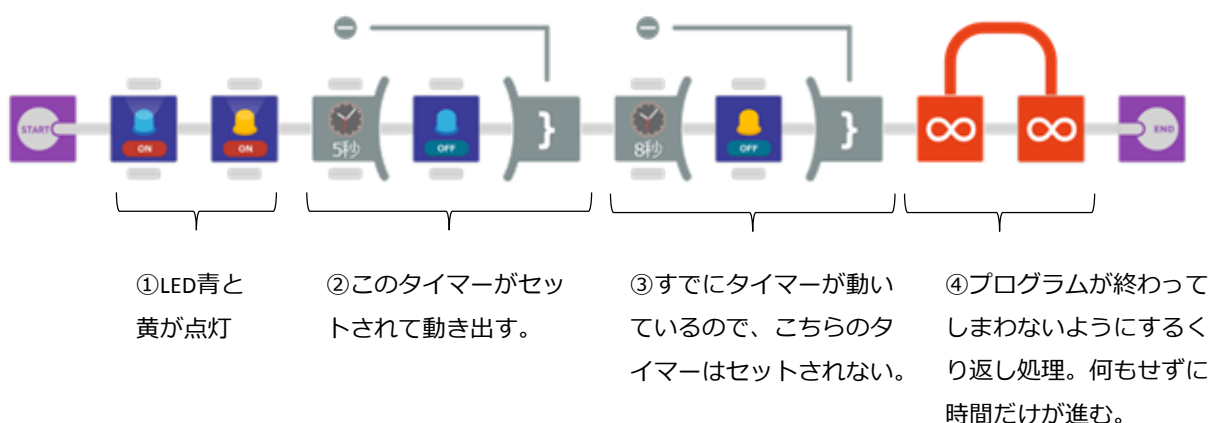
タイマーを2つ使ってしまった場合は？

タイマーがセットされて動いている間は、次のタイマーはセットされずに無視されて、プログラムの処理が進んでいきます。

例えば、

- ・ プログラムがスタートしたらすぐにLED青とLED黄色を点灯させる。
- ・ 5秒後にLED青を消す。
- ・ 8秒後にLED黄を消す。

このプログラムを以下のようにタイマーを2つ使って作成してみると期待通りに動作しません。



①プログラムがスタートすると、LED青、LED黄色が点灯する。

② 「【5秒後にLED青を消す】タイマー」がセットされて動き出す。

③ここでタイマーをセットしようとするが、すでに②のタイマーが動いているのでセットされない。

※①～③までは順番に一瞬で処理される

④ 5秒後に②のタイマーが終了してカッコの中を処理してLED青が消える。

そのあとは、③のタイマーはセットされていないので、何秒たっても変化しない。つまりLED黄は消えない。

知っておこう 2 :

タイマーの中の処理時間が長いときの注意点。

例えば、タイマーの外にある待機アイコンを処理中（つまり待機で設定した時間を待っている間）に、タイマーの中の処理をするタイミングになった場合は、タイマーの外の待機が一旦停止し、タイマーの中を優先して処理したあとに、一旦停止していた待機の処理を続けます。そのため、タイマーの中の処理に長く時間がかかったときには、タイマーの外の待機の時間が伸びたように動作することになります。

タイマーの中では、時間のかかる長い処理を書かないようにしましょう。

（時間が伸びる例）：タイマーのカッコの中に待機を使ったときの場合



①プログラムがスタートすると、LED青、LED黄色が点灯する。

② [3秒]タイマーがセットされて動き出す。

③つづけて待機(8秒)の処理を始める。

※①～③までは一瞬で処理されます

④8秒待っている間に(③の処理をしている間に)②のタイマーが終了するので、タイマーのカッコの中の処理を始める。

⑤タイマーのカッコの中に待機(4秒)があるので、③の待機を一旦停止しておいて待機(4秒)の処理を行う。

⑥待機(4秒)の処理が終わるとLED青を消してタイマーの処理から抜ける。

⑦タイマーの処理が終わったので、一旦停止していた待機が再び動き出す。

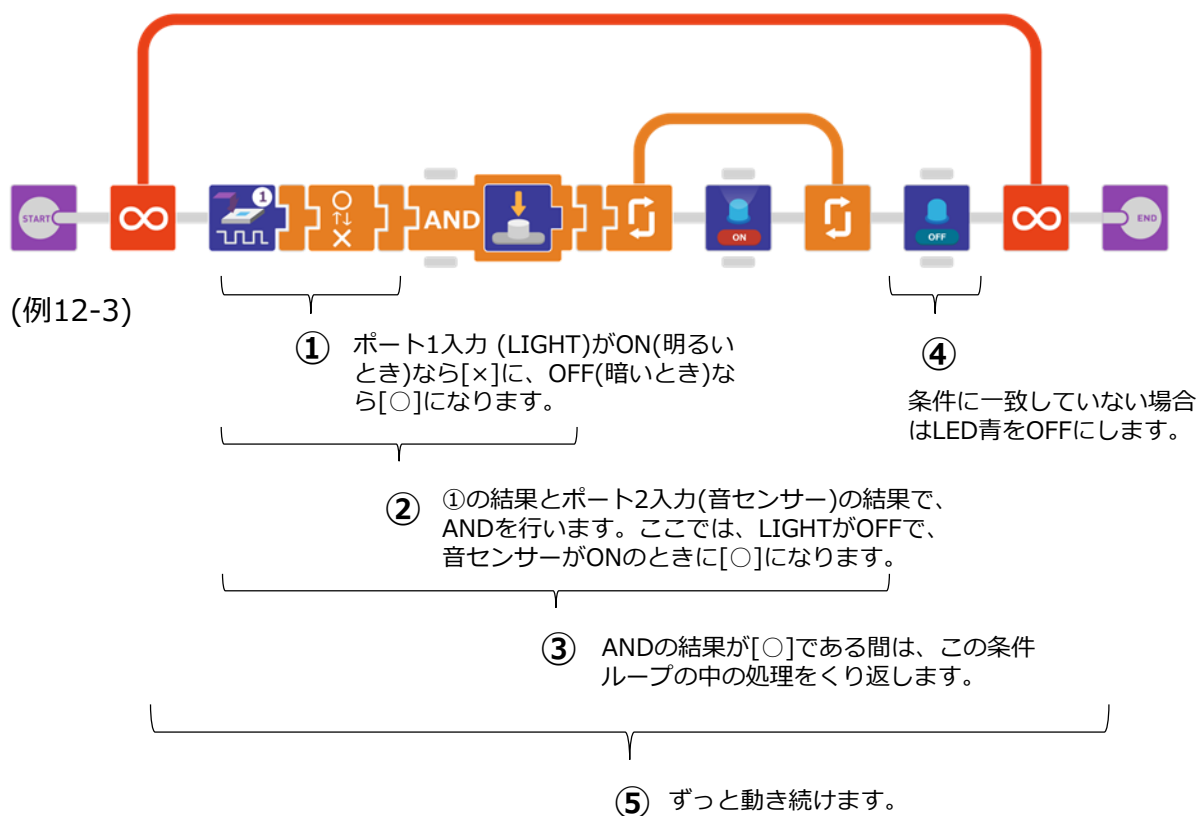
⑧待機(8秒)が終わって、LED黄が消える。

以上のように待機(8秒)が一旦停止してタイマーの中の待機(4秒)を実行するので、LED黄が消えるまでに合計の12秒かかることになります。



論理演算アイコンをつかう

「ボタンを押すと暗いときにだけLED青が点灯」の動作をさせるために、(例9-2)では条件分岐を使ってプログラムをつくりましたが、この動作は下図のように「論理演算」アイコンを使って同じ動きをするプログラムを作成することができます。論理演算アイコンを使うとプログラムがスッキリ見える場合もあります。このように同じ動作をさせる方法はひとつではなく、いろんな方法があることを知っておきましょう。



ここで使う論理演算アイコン



論理反転

左につながっているアイコンから[○]を受け取ると右につながっているアイコンに[X]を渡し、[X]を受け取ると[○]を渡します。



論理演算

[論理演算]アイコンの左側と内側の真偽値で論理演算を行った結果を、[○]または[X]で右のアイコンに渡します。論理演算の種類(AND, OR, XOR)をパラメータボタンで選択できます。

※AND, OR, XORについては14.アイコンの説明を見てください。

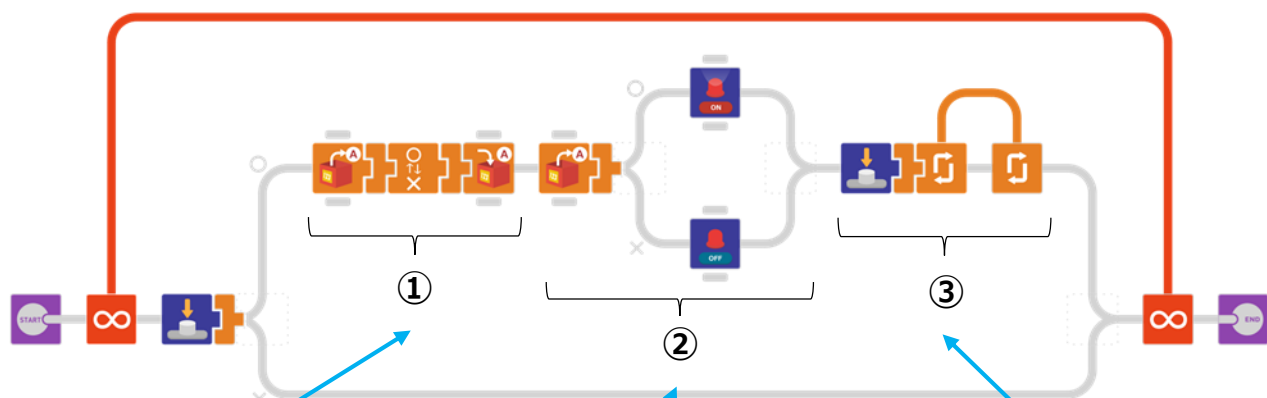


変数アイコンをつかう

「変数」をつかうと、もっといろいろな動きのプログラムができるようになります。変数はある一瞬の信号の状態を記憶できるので、途中で信号の状態が変化しても影響しないようなプログラムをつくることができます。

例えばボタンを押すごとにLEDをON→OFF→ON→OFF・・・と動くプログラムを作成するときに必要になります。

下図のようなプログラムを作成して動きを確認してみましょう。



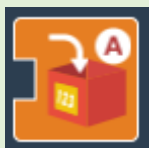
ボタンが押されたら、変数Aの真偽値を、反対にします。
[○]だったら[×]に、
[×]だったら[○]になります。

変数Aの値によって分岐します。
○だったらLEDをONして、
×だったらOFFにします。

ボタンから手が離れるのを待ってから、プログラムを進めます。
信号がOFFになるまで待ちたいときに使うプログラムのテクニックです。

(例12-4)

ここで使う変数アイコン



真偽変数 入力

真偽値を受け取り、パラメータボタンで指定した真偽変数(A, B, C, D, E)に格納します。



真偽変数 出力

パラメータボタンで指定した真偽変数(A, B, C, D, E)から真偽値を取り出して、右のアイコンに渡します。なお各真偽変数の初期値は[×]です。



条件ループ

真偽値を受け取り、[○]の場合は、[条件ループ]内の処理を実行しループの最後に到達すると、[条件ループ]の先頭に戻りもう一度真偽値を受け取ります。[×]の場合は、[条件ループ]末尾の次のアイコンに移動します。

13. 覚えると便利な操作テクニック



数値定数の値、待機の時間を効率よく設定する

[数値定数]アイコンは最小 -100、最大100の範囲内の値を、また、[待機]アイコンは最小0秒、最大10秒の範囲内の時間を設定することができます。アイコンの上下にあるパラメータボタンをクリックして値を設定することもできますが、それぞれのアイコンのパラメータボタンをドラッグすることで目盛りが表示され、マウスを上下に動かすことで値や時間を変えることができます。



※ - (マイナス) のときはパラメータボタン(下)をドラッグする



※ 待機アイコンのパラメータボタン(下)はドラッグしても目盛は表示されません。



マウスホイールでできること

マウスホイールを使うことでワークスペースに表示されているプログラムの拡大・縮小をすることができます。これはプログラム一覧画面・プログラム編集画面のどちらでも行うことができます。また、プログラム編集画面下部のドックにマウスポインタを合わせた状態でマウスホイールの上下操作をすると、ドックを水平方向にスクロールすることができます。



マウスホイールを奥に転がすと拡大



マウスホイールを手前に転がすと縮小



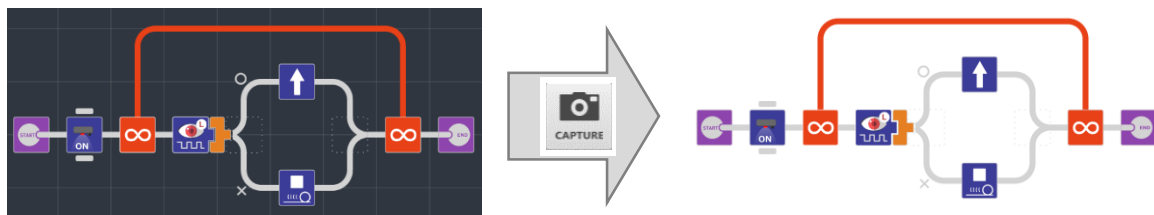
画面のスクロール・センタリング

プログラムを作成するとき、何もない(アイコンが置かれていない)位置でドラッグ操作を行うとワークスペース全体のスクロール操作ができます。また、何もない位置でダブルクリックをするとワークスペースをセンタリングします。これはプログラム一覧画面・プログラム編集画面のどちらでも行うことができます。

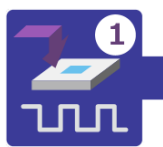


プログラムのキャプチャー

プログラム編集画面でキャプチャーアイコンを押すと、現在画面に表示しているプログラム部分をPNG形式の画像で保存します。背景は透明になります。画面のスクロールでキャプチャーしたい部分を表示させて使います。画像はPalette.exeが置かれているフォルダーに保存されます。作成したプログラムを印刷するときに便利です。



14. アイコンの説明

アイコン	アイコンの名前・説明
  	<LED赤 オン/オフ> <LED黄 オン/オフ> <LED青 オン/オフ> プログラミングモジュールのLED赤/黄/青をオン/オフします。パラメータボタンを押してオンとオフを切り替えることができます。
	<ボタン> プログラミングモジュールについているボタンの状態をチェックするときに使います。ボタンが押されているときは[○]、押されていないときは[×]になります。
	<ポート1チェック> プログラミングモジュールの入力1に接続したPIECEモジュールの状態をチェックするときに使います。PIECEモジュールがオン信号を出しているときは[○]に、信号を出していないときは[×]になります。
	<ポート1出力> プログラミングモジュールのポート出力1をオン/オフします。パラメータボタンを押してオンとオフを切り替えます。
	<スタート> このアイコンからプログラムが始まります。
	<エンド> 全ての動作を停止し、プログラムを終了します。

アイコン	アイコンの名前・説明
	<待機> パラメータによって設定した時間プログラムの処理を待機させます。 待機している間、直前の動作が継続されます。 設定できる時間は最小0秒で、0.1秒ずつ増やすことができ最大で10秒です。初期値は1秒です。
	<コードブロック> 複数のアイコンを囲んで一つのグループにするためのアイコンです。 プログラムの動作に与える影響はありません。 グループ先頭の[コードブロック]アイコンをドラッグすると、グループ全体をまとめて移動させることができます。グループ末尾の[コードブロック]アイコンをドラッグすることで、囲む範囲を変えることができます。
	<条件分岐> 真偽値を受け取り、[○]を受け取った場合は上の経路へ、[×]を受け取った場合は下の経路へ処理を進めます。 ※「条件分岐」アイコンはデータ(真偽値)を受け取るアイコンであるため、その左側には必ずデータ(真偽値)を渡すアイコンが配置されなくてはなりません。
	<スペーサー> プログラムを読みやすくするなどの目的で、アイコンとアイコンの間にスペースを挿入するためのアイコンです。プログラムの動作に与える影響はありません。
	<タイマー> タイマーに設定した時間になると、現在処理中のプログラムを一旦中断してタイマーアイコンのカッコで囲まれた部分を処理します。処理が終わったら元のプログラムに戻ります。 タイマーの時間はパラメータボタンを押して設定します。設定できる時間は0秒から1秒ずつ増やすことができ最大で100秒です。 ※プログラムはタイマーアイコンのカッコの中を優先して処理しますので、プログラム中で待機アイコンを実行しているときにタイマー割り込みが発生した場合、待機の時間が伸びたような動作になります。

アイコン	アイコンの名前・説明																																																						
	<無限ループ> [無限ループ]内の処理を実行します。ループの最後に到達した場合は、[無限ループ]の先頭に戻ります。																																																						
	<条件ループ> 真偽値を受け取り、[○]の場合は、[条件ループ]内の処理を実行しループの最後に到達すると、[条件ループ]の先頭に戻りもう一度真偽値を受け取ります。[×]の場合は、[条件ループ]末尾の次のアイコンに移動します。																																																						
	<真偽定数> パラメータボタンで設定した真偽値[○]または[×]を右のアイコンに渡します。																																																						
	<真偽変数 入力> 真偽値を受け取り、パラメータボタンで指定した真偽変数(A, B, C, D, E)に格納します。																																																						
	<真偽変数 出力> パラメータボタンで指定した真偽変数(A, B, C, D, E)から真偽値を取り出して、右のアイコンに渡します。なお各真偽変数の初期値は[×]です。																																																						
	<論理反転> 真偽値を受け取り、その値を反転させた真偽値を右のアイコンに渡します。[○]を受け取ると[×]を渡し、[×]を受け取ると[○]を渡します。																																																						
	<論理演算> [論理演算]アイコンの左側(入力1)と内側(入力2)の真偽値で論理演算を行った結果(出力)を、真偽値で右のアイコンに渡します。論理演算の種類(AND, OR, XOR)は、パラメータボタンで選択します。 <table><tr><th colspan="3">AND</th></tr><tr><th>入力 1</th><th>入力 2</th><th>出力</th></tr><tr><td>×</td><td>×</td><td>×</td></tr><tr><td>×</td><td>○</td><td>×</td></tr><tr><td>○</td><td>×</td><td>×</td></tr><tr><td>○</td><td>○</td><td>○</td></tr></table> <table><tr><th colspan="3">OR</th></tr><tr><th>入力 1</th><th>入力 2</th><th>出力</th></tr><tr><td>×</td><td>×</td><td>×</td></tr><tr><td>×</td><td>○</td><td>○</td></tr><tr><td>○</td><td>×</td><td>○</td></tr><tr><td>○</td><td>○</td><td>○</td></tr></table> <table><tr><th colspan="3">XOR</th></tr><tr><th>入力 1</th><th>入力 2</th><th>出力</th></tr><tr><td>×</td><td>×</td><td>×</td></tr><tr><td>×</td><td>○</td><td>○</td></tr><tr><td>○</td><td>×</td><td>○</td></tr><tr><td>○</td><td>○</td><td>×</td></tr></table>	AND			入力 1	入力 2	出力	×	×	×	×	○	×	○	×	×	○	○	○	OR			入力 1	入力 2	出力	×	×	×	×	○	○	○	×	○	○	○	○	XOR			入力 1	入力 2	出力	×	×	×	×	○	○	○	×	○	○	○	×
AND																																																							
入力 1	入力 2	出力																																																					
×	×	×																																																					
×	○	×																																																					
○	×	×																																																					
○	○	○																																																					
OR																																																							
入力 1	入力 2	出力																																																					
×	×	×																																																					
×	○	○																																																					
○	×	○																																																					
○	○	○																																																					
XOR																																																							
入力 1	入力 2	出力																																																					
×	×	×																																																					
×	○	○																																																					
○	×	○																																																					
○	○	×																																																					

アイコン	アイコンの名前・説明
	<回数ループ> 数値を受け取り、その回数、[回数ループ]内の処理を実行します。
	<数値定数> パラメータボタンで設定した数値を右のアイコンに渡します。パラメータボタンをクリックまたはドラッグして数値を設定します。設定できる数値は最小-100、最大100の範囲内の整数です。
	<数値変数 入力> 数値を受け取り、パラメータボタンで指定した数値変数(A, B, C, D, E)に格納します。格納できる数値の範囲は、-32768 ~ 32767の整数となります。
	<数値変数 出力> パラメータボタンで指定した数値変数(A, B, C, D, E)から数値を取り出して、右のアイコンに渡します。なお各数値変数の初期値は0です。
	<算術演算> [算術演算]アイコンの左側と内側で数値をそれぞれ受け取り、左側と内側の数値で算術演算を行った結果を、数値で右のアイコンに渡します。算術演算の種類(足し算、引き算、掛け算、割り算)は、パラメータボタンで選択します。なお割り算で内部のアイコンの数値が「0」のときは、実行時に「1」に置き換えられます。
	<正負反転> 数値を受け取り、その値の正負(プラスマイナス)を反転させた数値を右のアイコンに渡します。
	<算術比較> [算術比較]アイコンの左側と内側の数値で算術比較を行った結果を、真偽値で右のアイコンに渡します。算術比較の種類(大なり(>), 小なり(<), 等価(=))は、パラメータボタンで選択します。

15. トラブルシューティング

プログラムの転送の途中で転送失敗のメッセージが表示される。



- ・ USBケーブルが破損、変形していないか確認します。
- ・ PaletteIDEをダウンロード・展開したときにファイルの一部が破損している可能性があります。もう一度当社サイトからPaletteIDEをダウンロードして展開してみてください。また、もしもWindows標準機能を使わずに別の展開ソフトなどを使っている場合は、Windows標準機能を使って展開してみてください。

USBチェックアイコンが緑色にならず書き込めない。



- ・ 充電専用USBケーブルを誤って使っていないか確認します。プログラムの転送には付属のUSBケーブルを使います。
- ・ プログラミングモジュールのモード切替スイッチの位置を確認します。
- ・ プログラミングモジュールのリセットスイッチを押してみます。
- ・ パソコン側のUSB端子が不調である可能性があります。パソコンに別のUSB端子があれば、そちらの端子を使って確認してみます。

PIECEプログラミングモジュールのファームウェアには、ソフトウェアライブラリLUFA (<https://www.fourwalledcubicle.com/LUFA.php>)を使用しています。

PIECEプログラミングモジュールのファームウェアおよび転送用ソフトウェアのソースコードは <https://github.com/elekit-official> にて公開しています。