目次

1 プログラミングの準備をしよう

プログラミングを始める前に、いくつかの準備が必要です。１つ１つ確認しながら準備を進めていきましょう。

1-1 ユニットの特徴

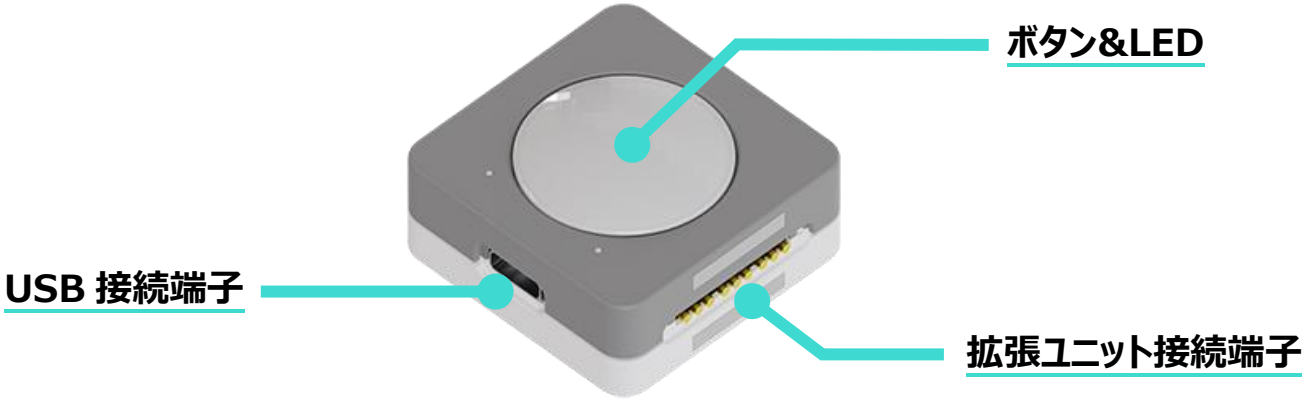
本商品はメインユニットと 3 種類の拡張ユニット（スピーカー・赤外線通信ユニット・マイク）を使ってプログラミングします。

本商品にセットされているユニットの一覧

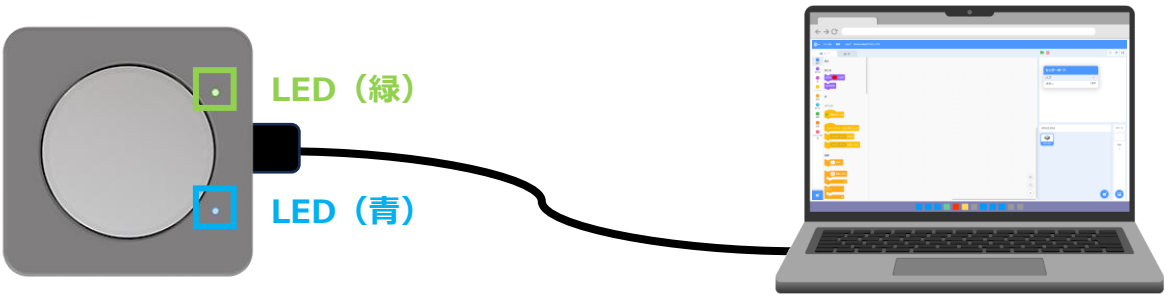
メインユニット	スピーカー
	
プログラミングで制御できる小さなコンピューターです ・ボタンと LED が付いています ・拡張ユニットを接続できます	・ブザー音や録音した音声、wav ファイルを再生します 使い方・作例 ▶ p.27
赤外線通信ユニット	**マイク**
	
・赤外線を受信したり送信したりできます ・赤外線信号を記録できます 使い方・作例 ▶ p.35	・音の大きさを計測します ・音声を録音します 使い方・作例 ▶ p.40

メインユニット

プログラミングで制御することのできる小さなコンピューターです。
中央の LED を点灯させたり、接続した拡張ユニットを操作したりすることができます。
また、中央の LED はボタンにもなっていて、ボタンを使ってプログラムを制御することができます。



メインユニットは USB Type-C ケーブルでデバイスに接続して使用します。
デバイスから電気が供給されていると、緑色の小さな LED が点灯します。
また、インターネットや他のメインユニットと接続している場合は、青色の小さな LED が点灯します。



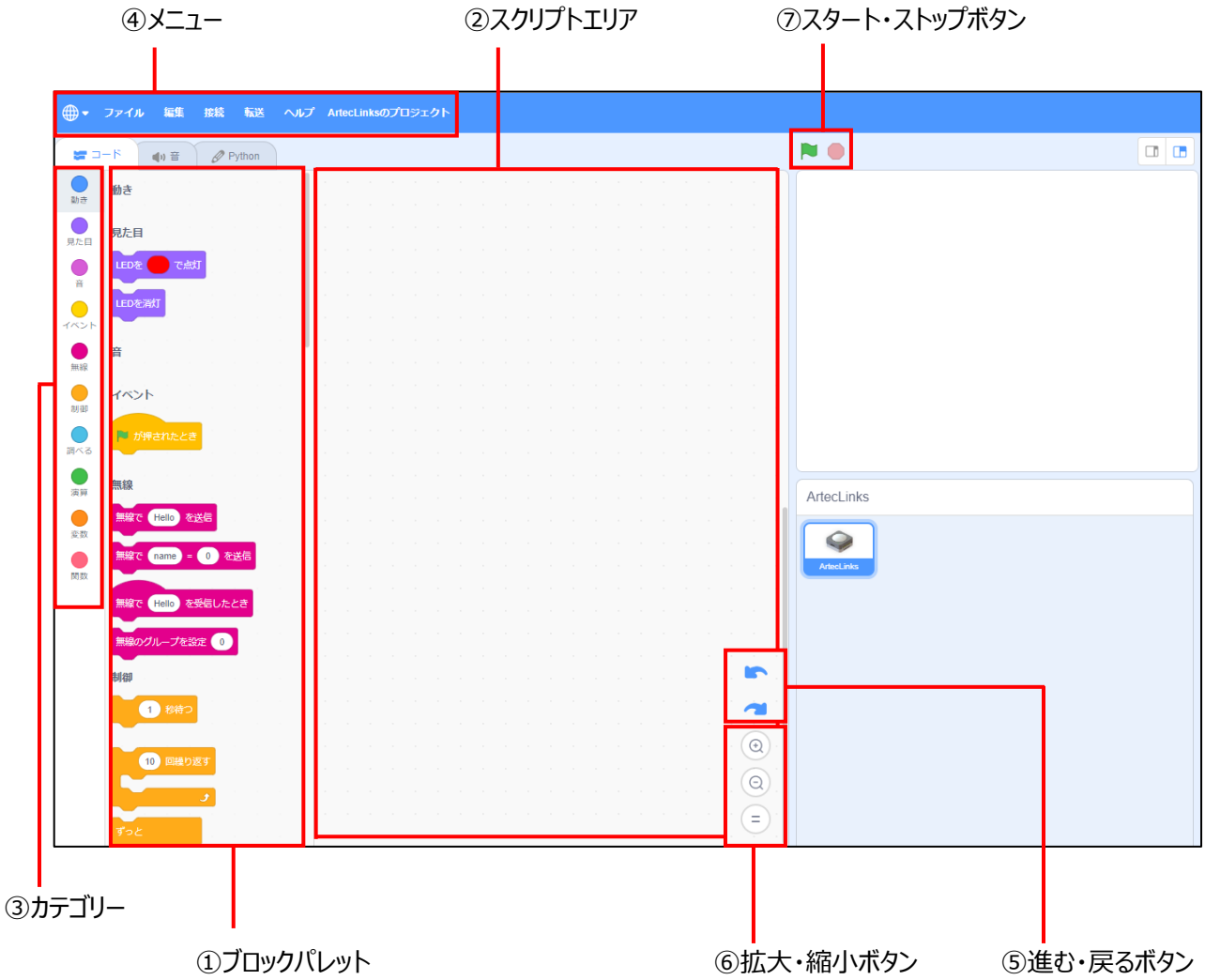
拡張ユニット

メインユニットに接続することで、様々な機能を追加することができます。
マグネットがついている面どうしをくっつけることで、接続できます。

1-2 ソフトウェアのダウンロード

プログラミングをするためには、専用のソフトウェアが必要です。
下記サイトにアクセスして、インストール版のソフトウェアをダウンロードしてください。
または、オンライン版の WEB アプリを開いて使用してください。
https://www.artec-kk.co.jp/arteclinks/software/

ソフトウェアを立ち上げると次の画面が表示されます。
画面内のそれぞれの場所と名前、役割を確認しましょう。



それぞれの役割

①ブロックパレット

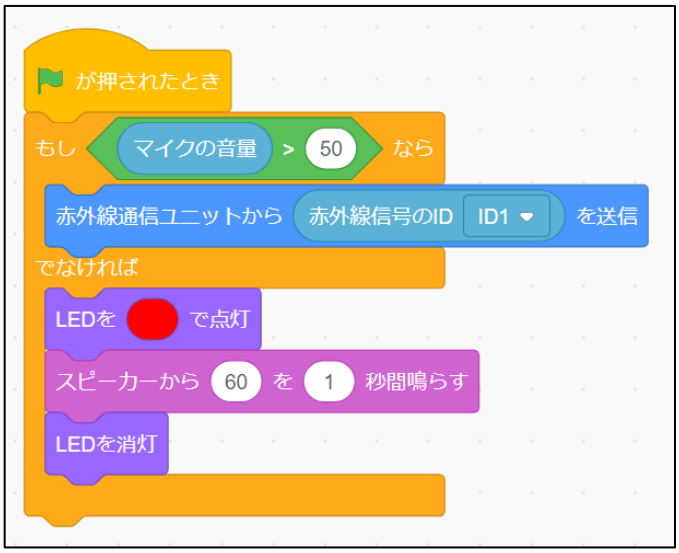
メインユニットへ伝える命令が並んでいます。
これらひとつひとつの命令のことを「ブロック」と呼びます。

②スクリプトエリア

ブロックを並べることができるエリアです。

ブロックどうしはドラッグ&ドロップで自由に繋げたり離したりすることができます。

ブロックを繋げて作成したプログラムのことを「スクリプト」と呼びます。



③カテゴリー

ブロックは、それぞれが持つ特徴によって色分けされています。

カテゴリーから使用したいブロックの種類を選択して、ブロックパレット内を移動することができます。

動き	赤外線を送信するときに使うブロック	制御	命令を実行するタイミングを決めるときに使うブロック
見た目	メインユニットの LED を制御するときに使うブロック	調べる	センサーやタイマーの値を調べるときに使うブロック
音	スピーカーから音を再生するときに使うブロック	演算	数字の大小を比較・計算するときに使うブロック
イベント	イベントが発生したときにスクリプトを開始するブロック	変数	変数・リストを使うためのブロック
無線	他のメインユニットと無線で通信するときに使うブロック	関数	関数を使うためのブロック
IoT	メインユニットを Wi-Fi に接続するときに使うブロック		

※無線ブロックは、本製品では使用しません

④メニュー

ファイルの読み込みや保存、メインユニットとの接続や転送などの機能を選択して実行します。

	ソフトウェアで表示される言語を切り替えます。「日本語(漢字)」、「にほんご(ひらがな)」、「English(英語)」の 3 種類から選択できます。
	作成したプログラムの保存や読み込みを行います。
	ブロックの表示・非表示の切り替えを行います。
	メインユニットをデバイスに接続します。
	つくったプログラムをメインユニットに転送します。
	ファームウェア(メインユニットのバージョン)の更新や、ソフトウェアのバージョンを確認できます。

⑤拡大・縮小ボタン

スクリプトエリアに並べたブロックの表示倍率を変更できます。

	1 段階拡大して表示します。		1 段階縮小して表示します。
	最初の倍率に戻して表示します。		

⑥進む、戻るボタン

スクリプトを一つ前、または一つ後の状態に戻すことができます。

	スクリプトを 1 つ前の状態に戻します。		スクリプトを 1 つ後の状態に戻します。

⑦スタート・ストップボタン

デバイスとの通信中に、スクリプトの実行・停止を行うことができます。

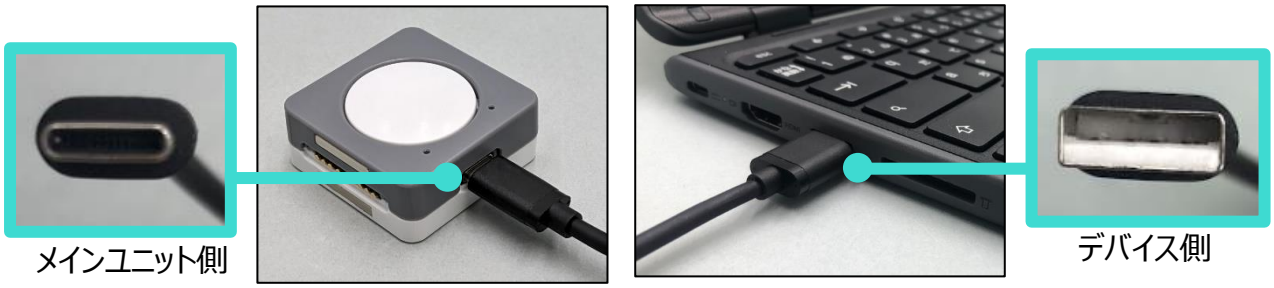
	 の下に繋がれたスクリプトを実行します。		実行しているスクリプトを停止します。

1-3 メインユニットとの接続

メインユニットをデバイスに接続しましょう。

①ソフトウェアを起動させます。

②メインユニットとパソコンを USB ケーブルで接続します。



③画面左上のメニューから「接続」をクリックします。

④「USB JTAG…」を選び、「接続」をクリックします。
画面上に「ユニットボード」が表示されたら接続完了です。

接続に失敗する場合は USB ケーブルを抜き差しして、再度②から実行して下さい。

2 プログラミングを練習しよう

2 章では、メインユニットに搭載されている LED とボタンを使って、プログラミングの基本を練習します。
また、拡張ユニットを使ったプログラムも紹介しています。

以降はメインユニットとデバイスを接続して、画面にユニットボードが表示された状態でプログラミングして下さい。

2-1 プログラミングで LED を点灯させよう

LED を点灯・消灯させよう

メインユニットの LED を点灯させるときは 

のブロックを使います。

また、消灯させるときは 

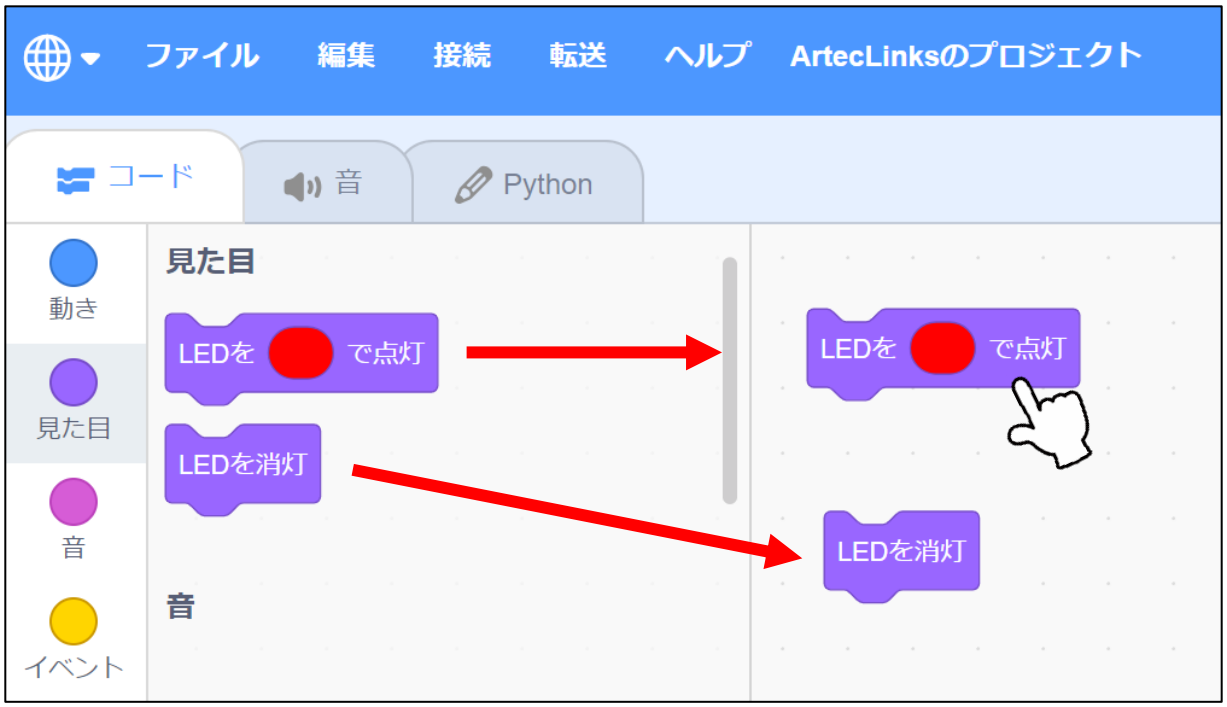
のブロックを使います。

①カテゴリーの 

にある 

と 

をスクリプトエリアにドラッグ&ドロップします。



②ブロックをクリックすると、それぞれに対応した命令が実行されます。

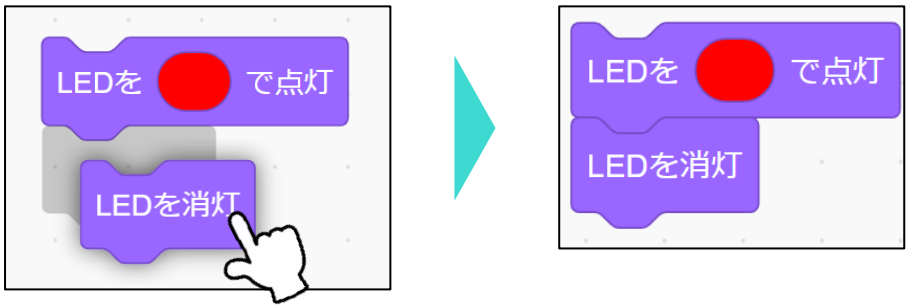
スクリプトエリアにドラッグ&ドロップしたブロックは、繋げることで上から順番に実行させることができます。

③ 

の下に 

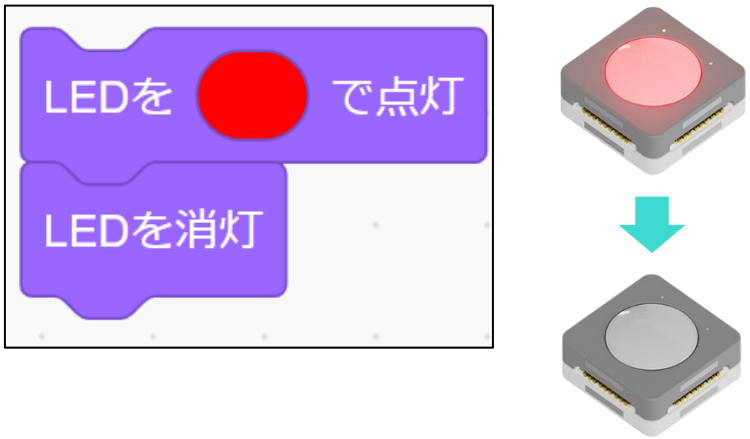
を繋げましょう。

ブロックをドラッグして繋げたいブロックの下に移動させ、灰色が表示されているときに手を離すことで
ブロックを繋げることができます。



このスクリプトを実行すると、メインユニットの LED が一瞬だけ赤色で点灯し、すぐに消灯します。

繋げられたブロックは、必ず上のブロックから順番に実行されます。



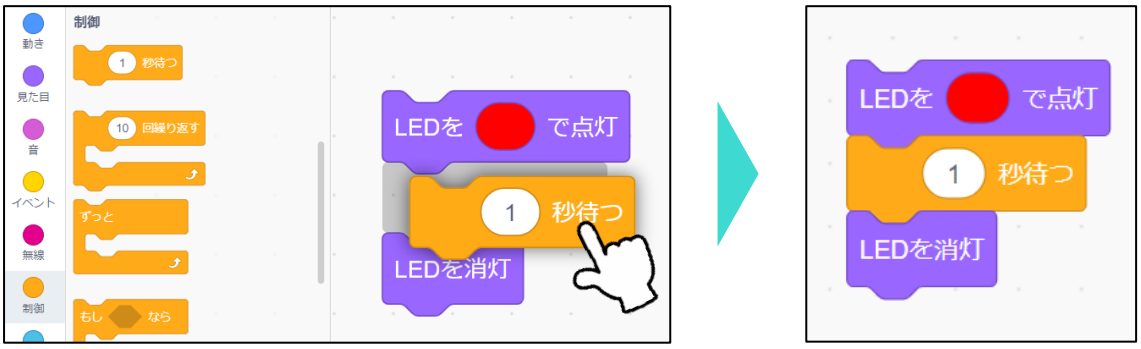
④点灯する時間を長くしたいときは 

を使ってスクリプトを一時停止させます。

カテゴリーの 

にある 

をブロックの間に挟みましょう。



この状態でスクリプトを実行すると、LED が赤色で点灯し、1 秒後に消灯します。

1 秒待つ

には好きな数値を入力して、待つ時間を変えることができます。

数値の入った白い丸をクリックして、キーボードで好きな数値を入力しましょう。

※入力する数値は必ず「半角」で入力しましょう。全角で入力した場合、スクリプトは正常に実行されません。
テキスト内では数値を入力する箇所が複数出てきますが、必ず半角で入力してください。

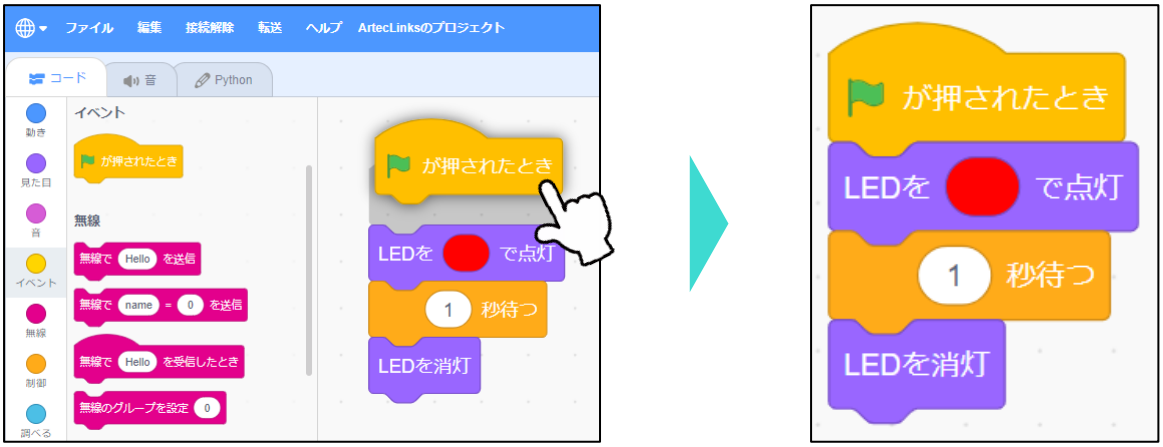
また、繋げたブロックは 

と  を使うことでも実行できます。

カテゴリーの 

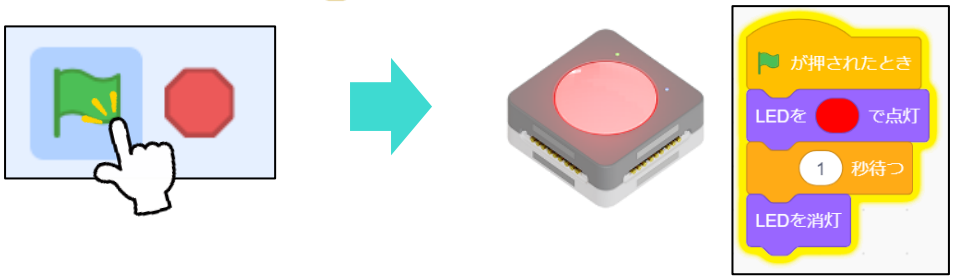
から 

をスクリプトの一番上にドラッグ&ドロップして繋げます。



この状態で  をクリックすると、 

の下に繋げられているスクリプトが実行されます。

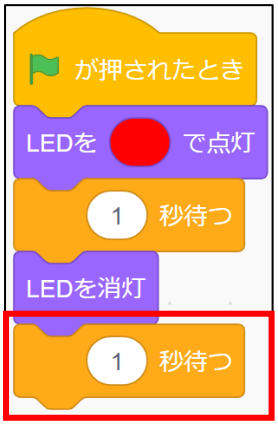


LED を点滅させよう

LED を 1 秒間隔で点滅させてみましょう。

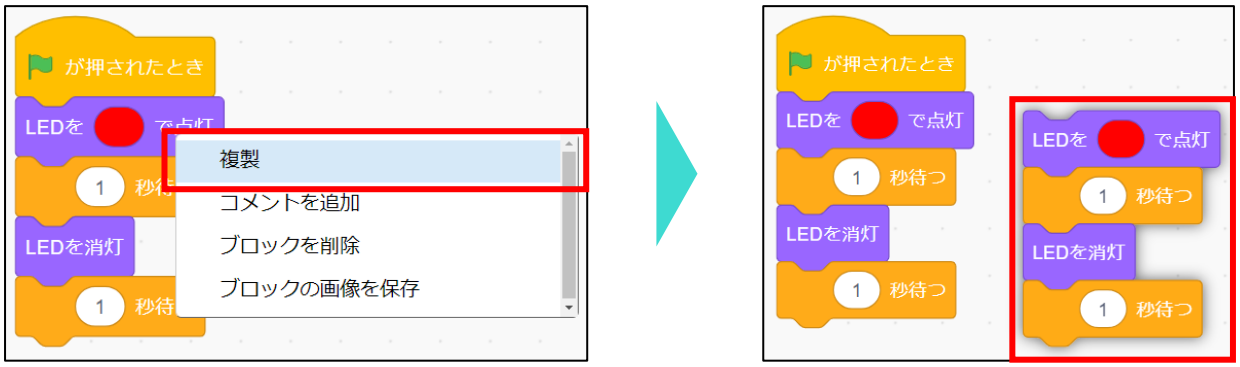
上でつくったスクリプトの下に 

を繋げると、1 回だけ LED を点滅させるスクリプトになります。

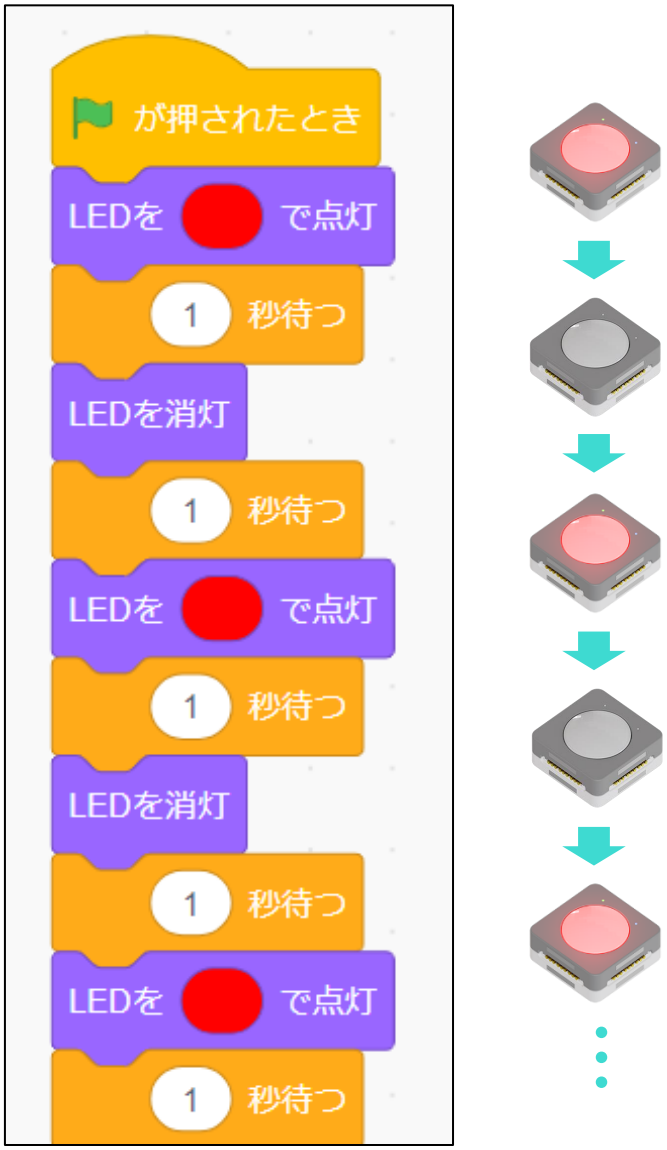


このスクリプトをもとに、何回も LED を点滅させるスクリプトをつくります。

同じブロックやスクリプトを何回も使いたい場合は「複製」を使います。
複製したいブロックやスクリプトを右クリックして「複製」をクリックすると、スクリプトエリアに同じスクリプトが追加されます。



ブロックを複製して繋げることで、何回も点滅させるプログラムがつくれます。



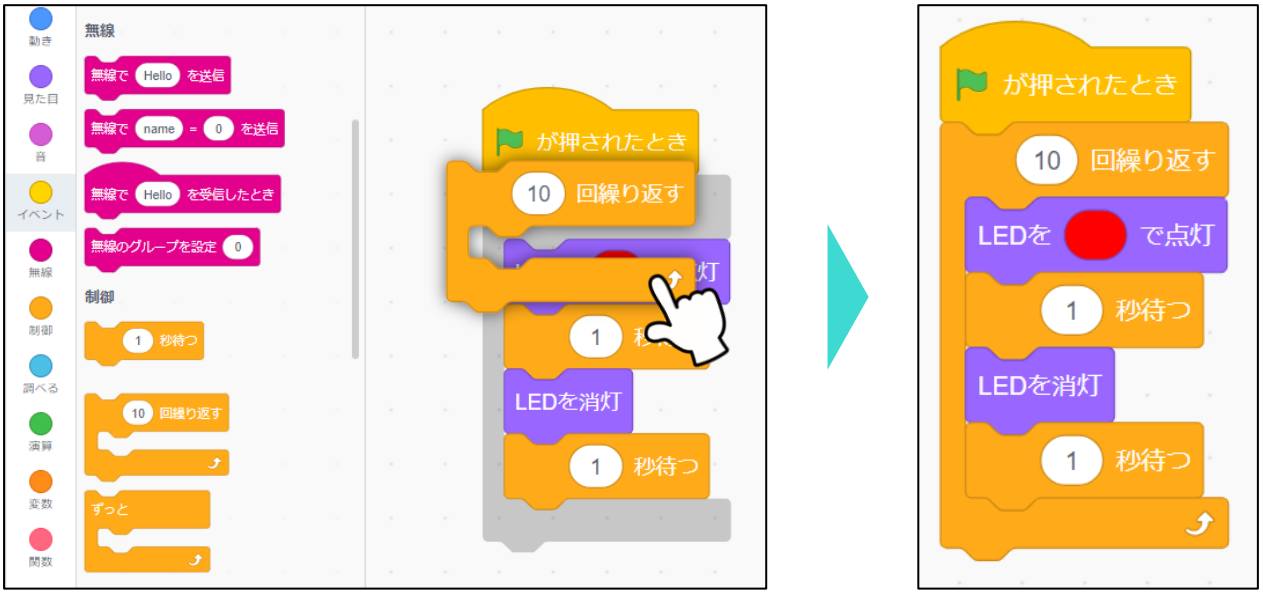
繰り返しブロックを使うと、ブロックの命令を繰り返し実行することができます。

カテゴリーの 

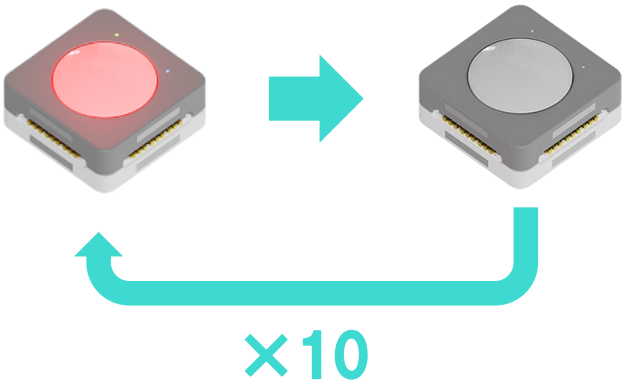
にある 

をドラッグして、 

の下で手を離します。



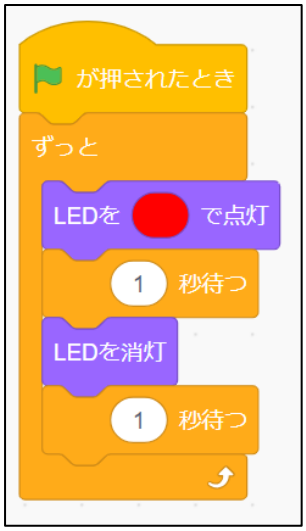
この状態でスクリプトを実行すると、LED が 10 回点滅します。

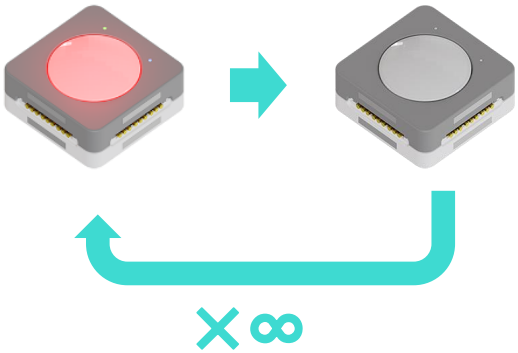


には初めから「10」が入力されていますが、
クリックして数値を入力することで、スクリプトを繰り返す回数を変えることができます。

の代わりに


を使用すると、
ブロックで囲んだスクリプトをずっと実行させることができます。

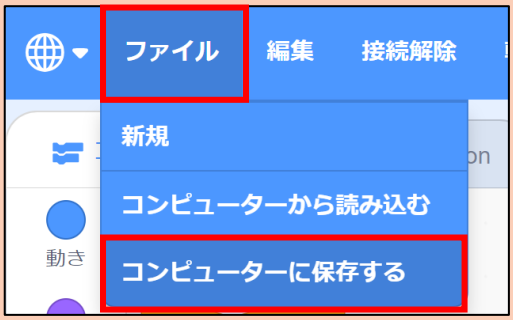




プログラムの保存と読み込み

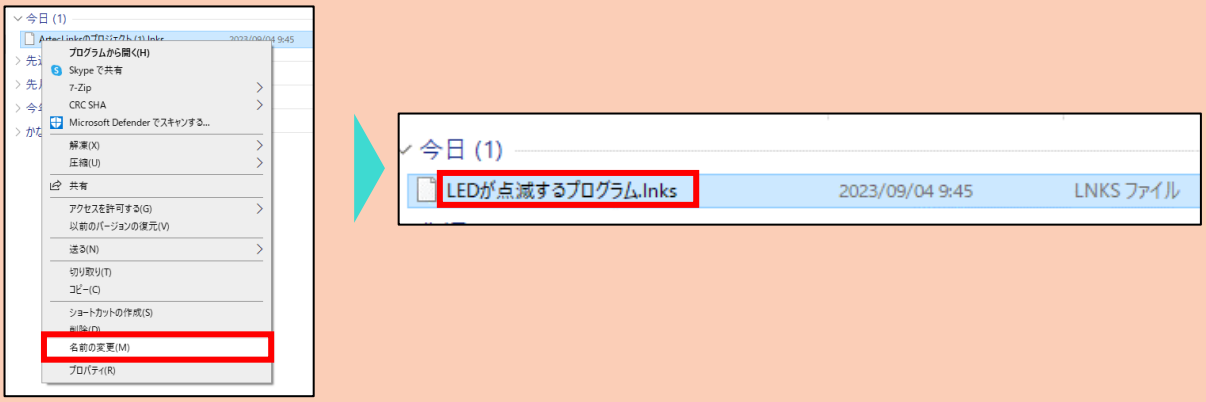
作成したプログラムはデバイスに保存することができます。

①「ファイル」から「コンピューターに保存する」を選択して、プログラムをデバイスに保存します。



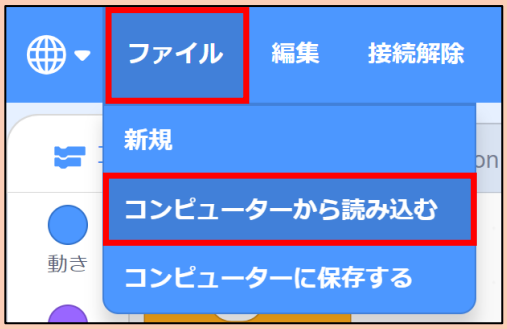
※オンライン版（Web アプリ）を使用している場合は、デバイスの「ダウンロード」フォルダに保存されます。
（Windows の場合）

②どんなプログラムをつくったのかわかるように、保存したフォルダを右クリックして、
「名前の変更(M)」を選択して名前を入力しましょう。（Windows の場合）

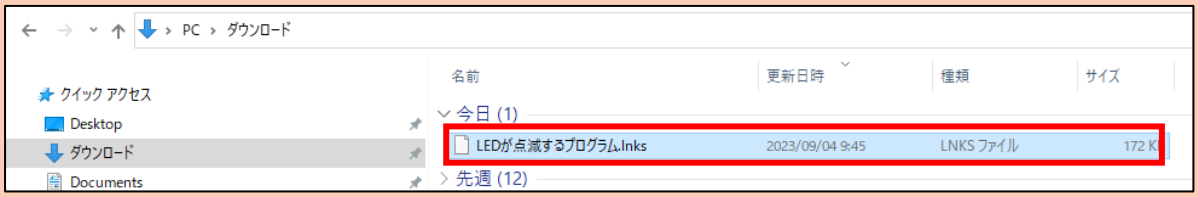


完成したプログラムだけでなく作成中のプログラムを保存しておくと、プログラムを途中から読み込んで続きを作成したり、内容を変更したりすることができます。こまめな保存を心がけましょう。

③プログラムを読み込む場合は、メニューの「ファイル」から「コンピューターから読み込む」を選択して、
デバイスのフォルダを開きます。



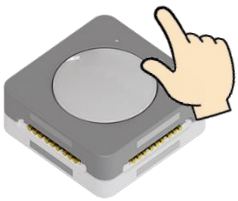
④読み込みたいファイルをダブルクリックしてプログラムを読み込みます。

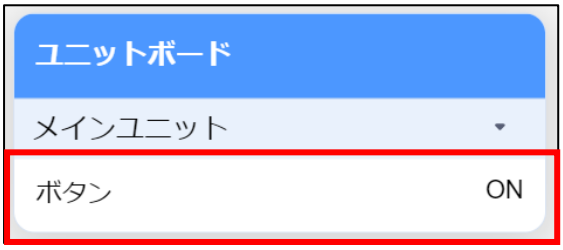


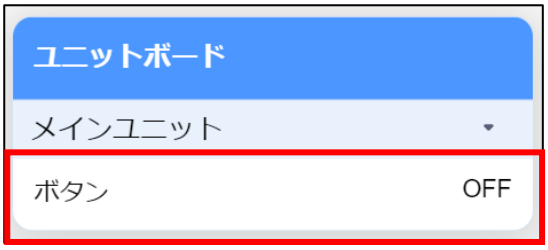
ボタンを使って LED を点灯させよう

メインユニットのボタンで LED の点灯・消灯を制御するプログラムを作成します。

メインユニットのボタンは、画面上の「ユニットボード」から押されているかどうかを判別することができます。







条件分岐ブロックを使うと、ボタンが押されているかどうかによって、異なるスクリプトを実行させることができます。

①カテゴリーの 

にある 

を 

の下にドラッグします。

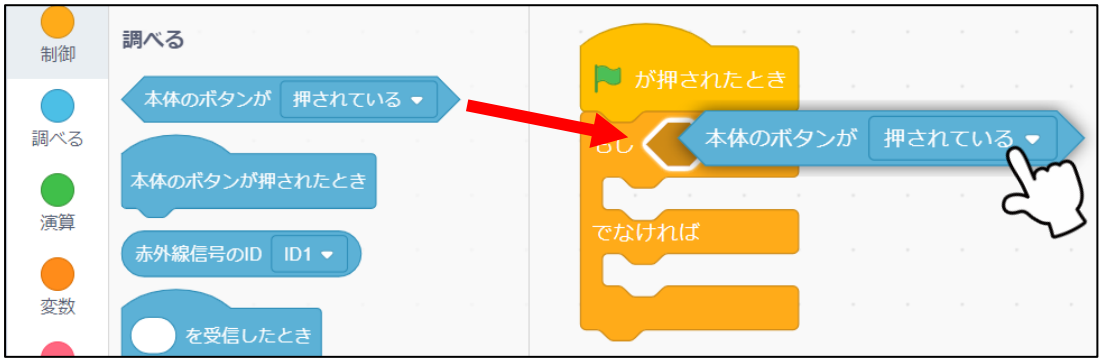
② 

の空欄に、カテゴリーの 

にある 

を

ドラッグ&ドロップします。



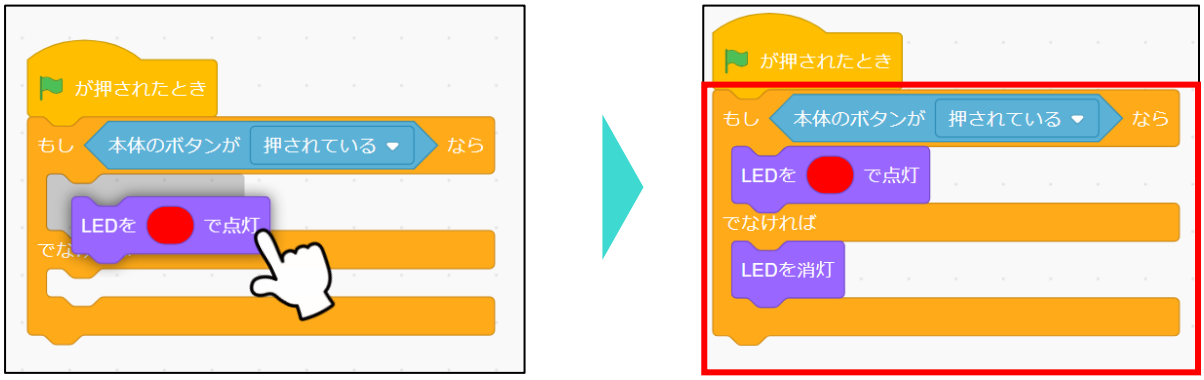
③ 

の「もし～なら」の下に 

を、

「でなければ」の下に 

をドラッグ&ドロップします。

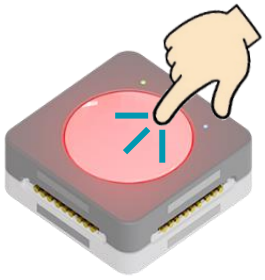


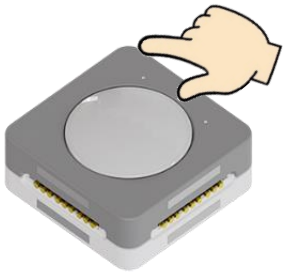
④ 

の下に 

をドラッグ&ドロップして、スクリプトを囲みます。

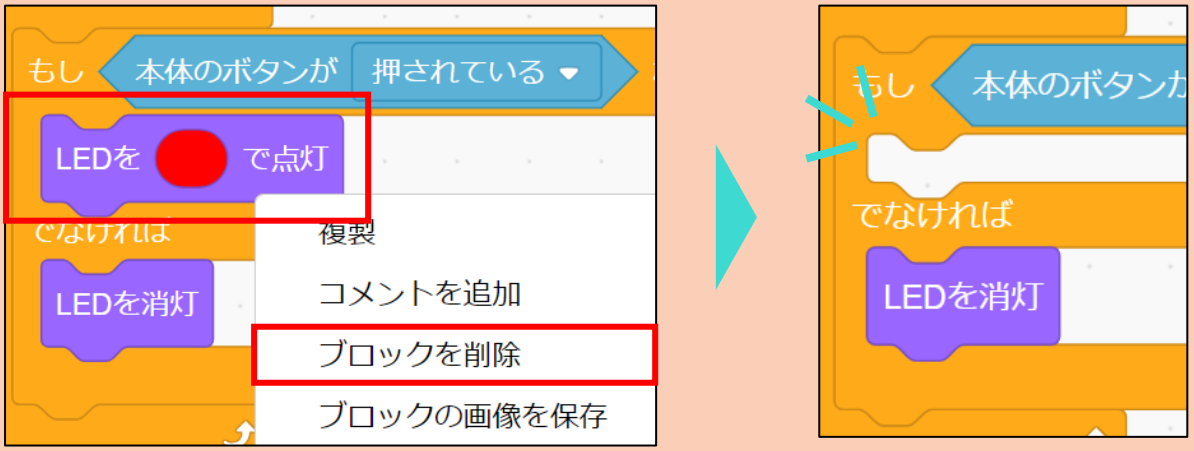
⑤スクリプトを実行します。ボタンを押すと LED が赤色で点灯し、ボタンから手を離すと消灯します。





ブロックの削除

間違って並べてしまったブロックは、ブロックを右クリックして「ブロックを削除」をクリックすることで削除できます。



また、条件分岐をするブロックは 

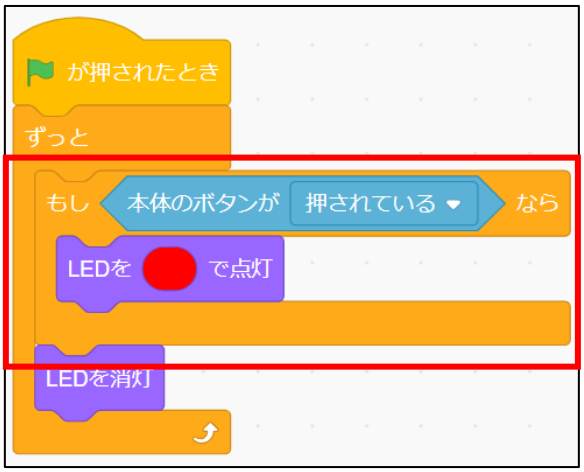
の他にも 

があります。

この 2 つのブロックの違いは何でしょうか？

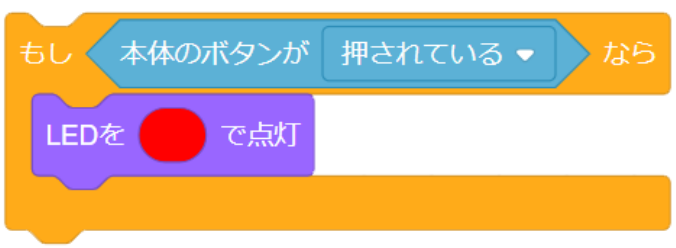
④で作成したスクリプトと次のスクリプトを比較してみましょう。

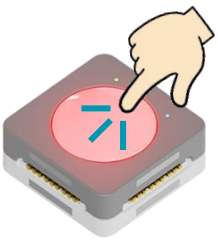
次のスクリプトも「本体のボタンが押されている」ときに LED を赤色で点灯させるスクリプトです。



プログラムは上から順番に実行されるため、このスクリプトでは、

ボタンが押されているときに点灯した後、消灯するブロックが順番に実行されます。





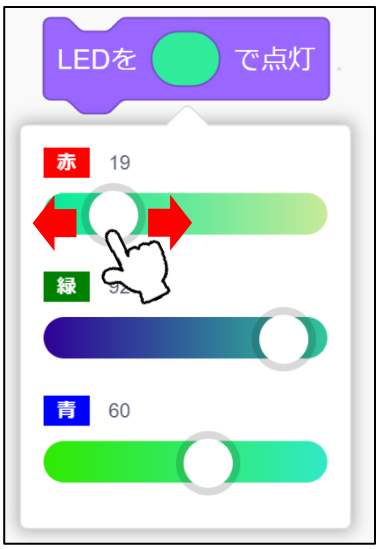
そのため、このスクリプトを実行してボタンを押すと、LED が高速で赤く点滅してしまいます。

ブロックを並べる順番に注意しながら、つくりたいプログラムによって 2 種類のブロックを使い分けましょう。

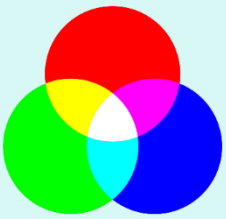
LED を様々な色で点灯させよう

LED が点灯する色は、プログラミングをして変更することができます。

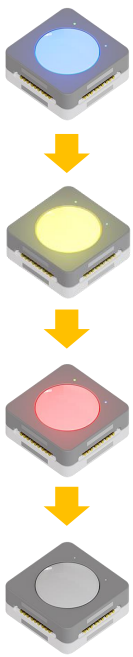
の赤色の部分をクリックして、表示された白丸を左右にドラッグすると、点灯させる色を変更できます。

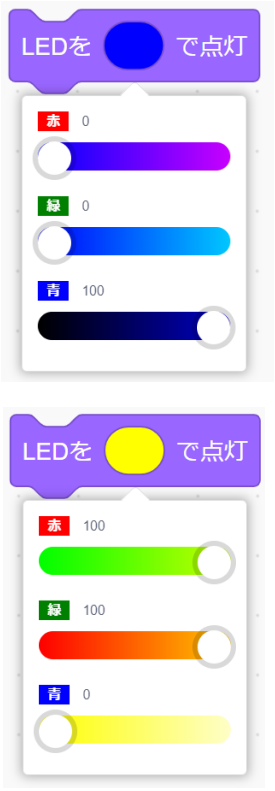


アーテックリンクスでは、赤・緑・青の３色(光の三原色)の光の強さを 0～100 で制御して混ぜ合わせることで、様々な色を表現しています。
このように光を混ぜ合わせて色を表現することを「加色混合」といいます。



赤・緑・青のそれぞれの光の強さを変更して、LED を様々な色に点灯させるプログラムをつくってみましょう。





点灯させる色は、光の強さを直接入力することでも変更できます。

メニューの「編集」から「ブロックの表示・非表示」を選択し、「LED RGB 数値指定」をクリックすると、
カテゴリーの 

に 

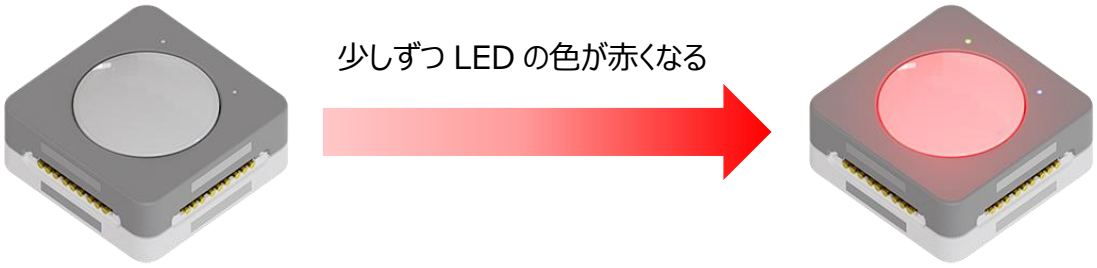
が追加されます。

このブロックの空欄に赤・緑・青のそれぞれの光の強さを入力することで、好きな色で点灯させることができます。

LED をグラデーションで点灯させよう

LED をグラデーションで点灯させましょう。

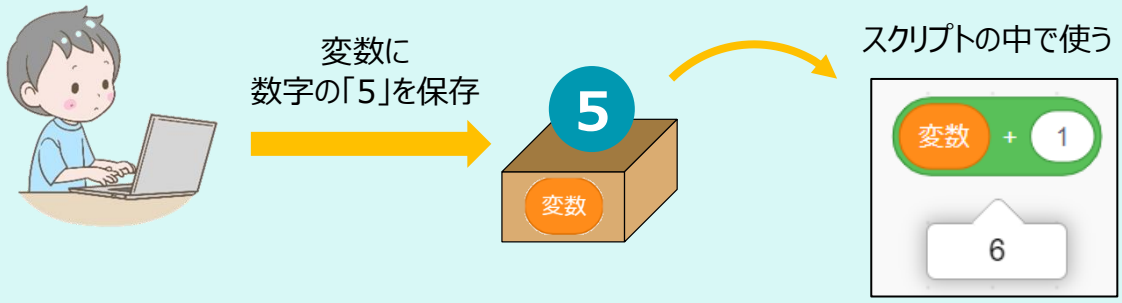
今回作成するスクリプトでは、「変数」を使います。



変数とは

変数とは、数字や文字を記憶できる「箱」のことです。

必要なデータを変数に保存しておくと、スクリプトの実行中に保存されているデータを使うことができます。

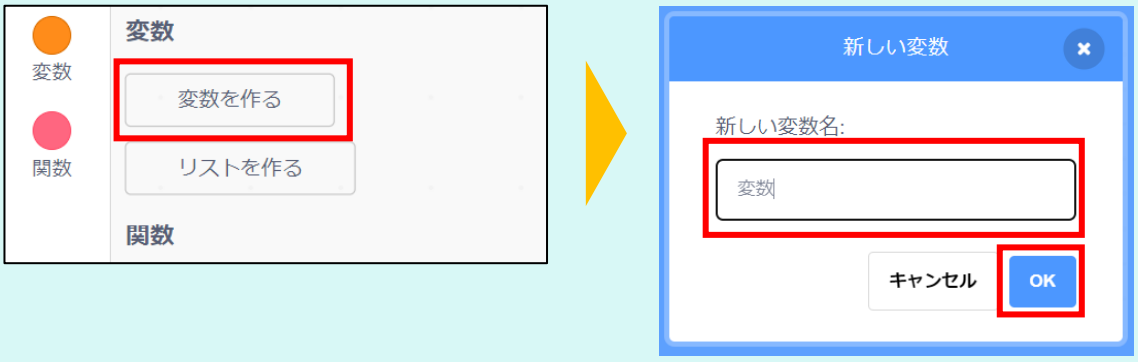


＜変数の使い方＞

①カテゴリーの 

から「変数を作る」を選択し、名前を入力して「OK」をクリックします。

(名前は後から変更できます。)

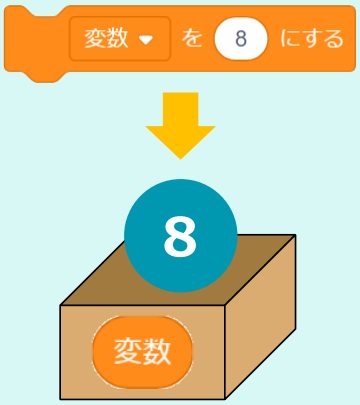


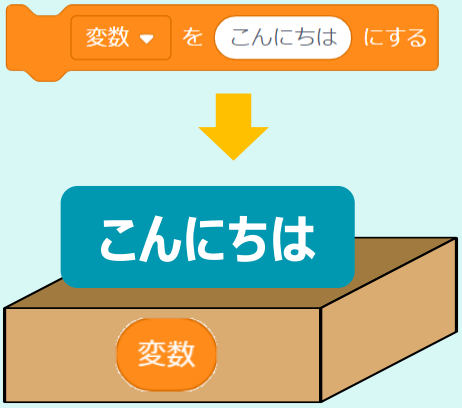
②ブロックパレットに作成した変数のブロックが表示されていることを確認しましょう。

③変数にデータを保存したいときは、 

の空欄に数値や文字を入力して実行します。

※保存をするごとに、それまで保存されていたデータは上書きされるため、注意しましょう。





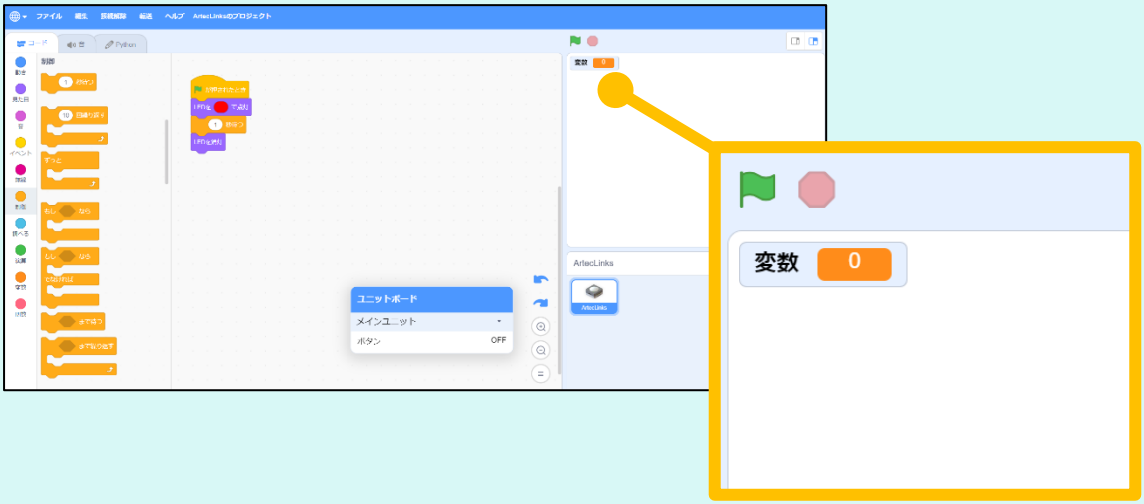
また、 

を実行することでも変更できます。

保存されている数値

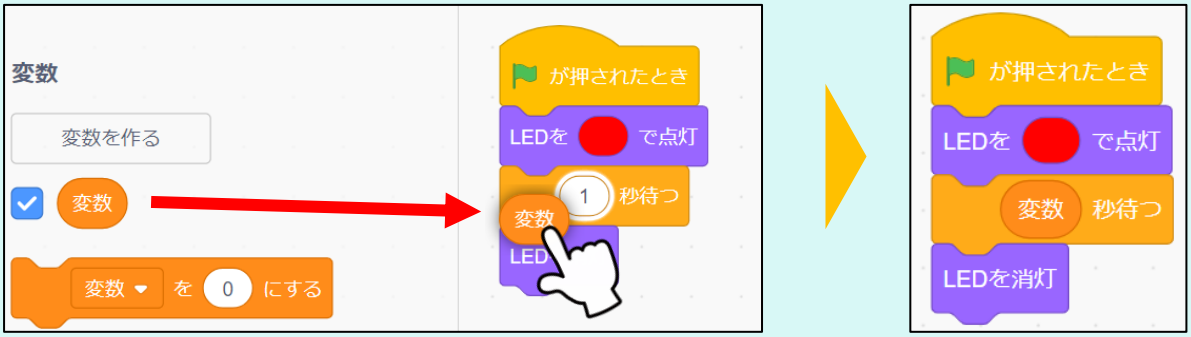
新しく保存される数値

④変数に保存されているデータは、画面右上に表示されています。
変数には初めから「0」が保存されています。



※画面右上に表示されている変数の表示・非表示を切り替えたい場合は、次のブロックを実行する、または
ブロックパレットの☑をクリックして切り替えましょう。

⑤変数に保存されているデータをスクリプトで使用したいときは 変数 を使います。

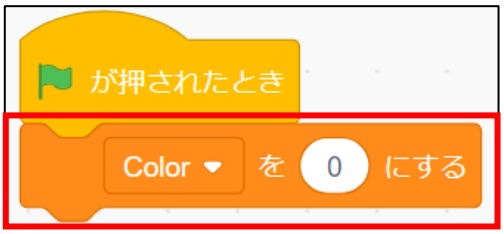


それでは、LED をグラデーションで点灯させるスクリプトをつくりましょう。

①変数「Color」を作成します。

※変数の名前によって、プログラムの動作が変わることはありません。
何のデータを保存しているか、わかりやすい名前を付けましょう。

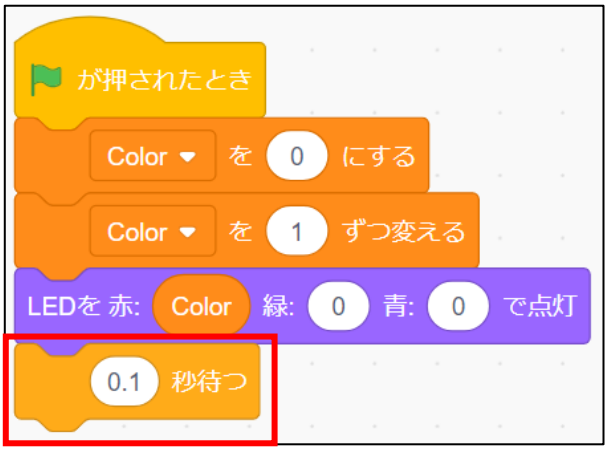
②変数「Color」を「0」にします。



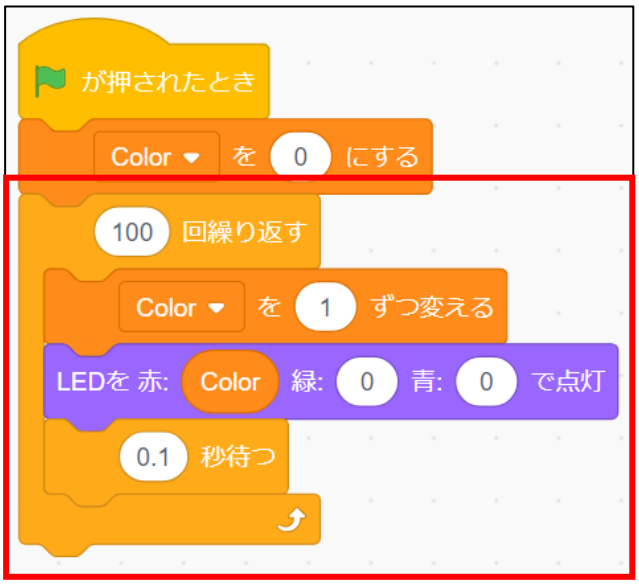
③変数「Color」を「1」ずつ変えます。

④LED の赤色の光の強さを、変数「Color」の値にして点灯させます。

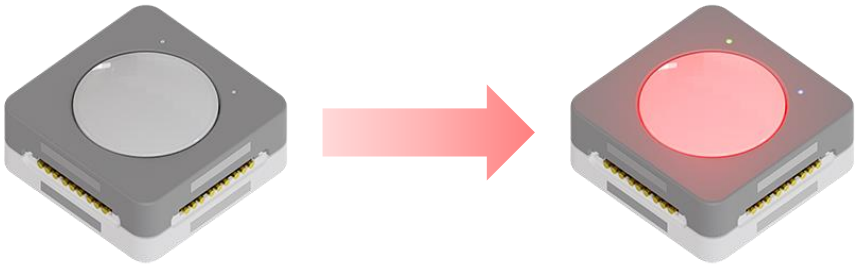
⑤LED の色を少しずつ変化させるために、0.1 秒待ちます。



⑥③～⑤でつくったスクリプトを「100 回」繰り返します。



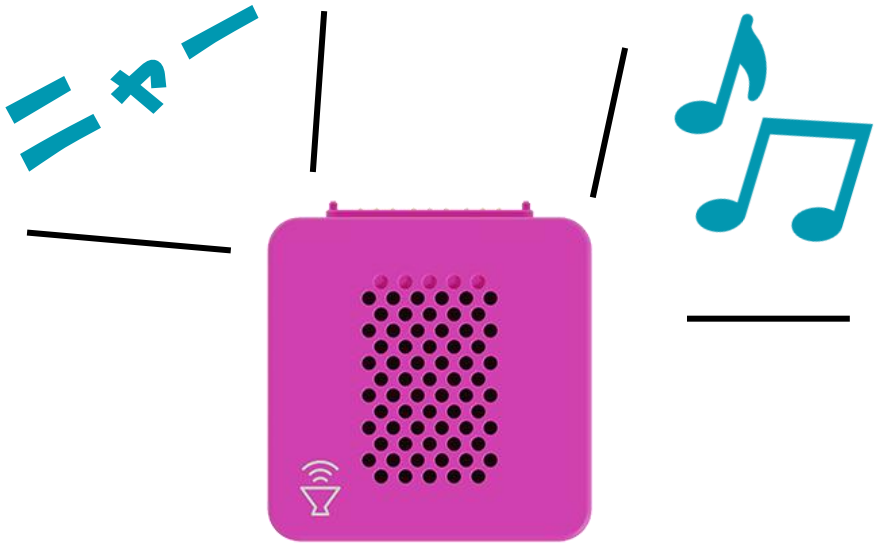
スクリプトを実行すると、点灯している LED の色が少しずつ赤く変化します。



赤：0　　**赤：100**

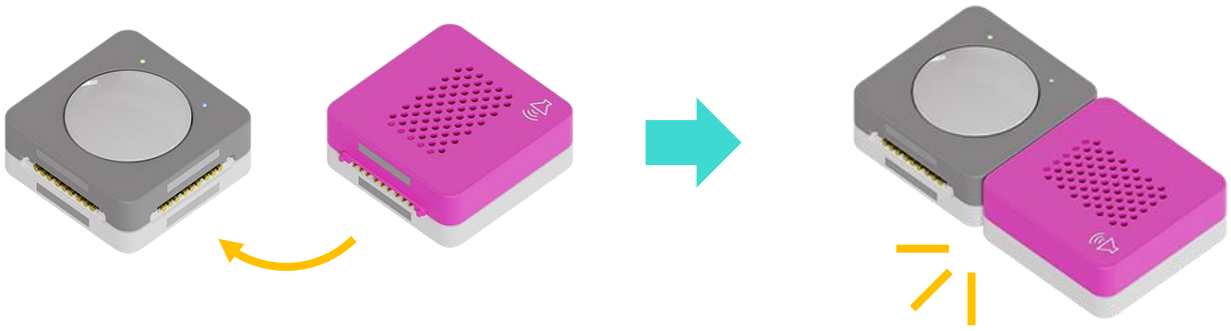
2-2 スピーカーから音を再生しよう

メインユニットに拡張ユニットを接続すると、様々な機能を使うことできます。
今回は、メインユニットにスピーカーを接続して、好きな音を鳴らすプログラムを作成しましょう。

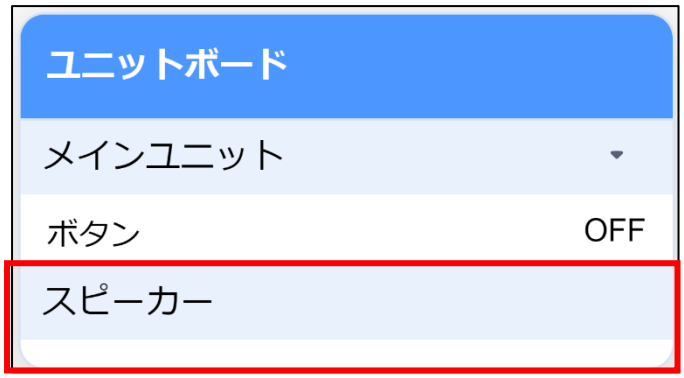


スピーカーをメインユニットに接続しよう

①スピーカーをメインユニットに接続します。(どの位置に接続しても問題ありません)



②ユニットボードに「スピーカー」が追加されます。



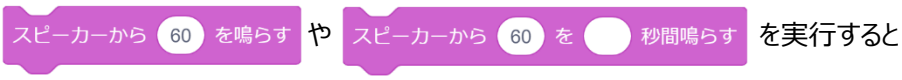
ユニットボードに「スピーカー」が表示されない場合は一度拡張ユニットを取り外し、再度接続してください。

③カテゴリーの 

にスピーカー用のブロックが追加されます。

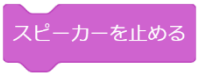
※スピーカー用のブロックが表示されない場合は、画面左上の「編集」から「ブロックの表示・非表示」を選び、
「スピーカーブロック」をクリックしましょう。

スピーカーから音を鳴らそう



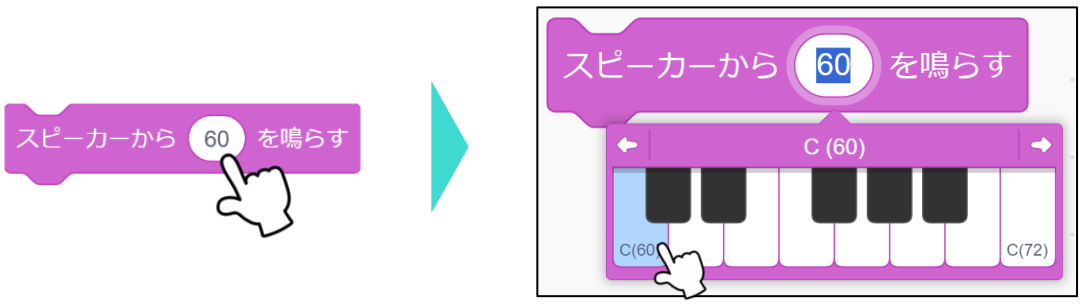
を実行すると

スピーカーからブザー音を鳴らすことができます。

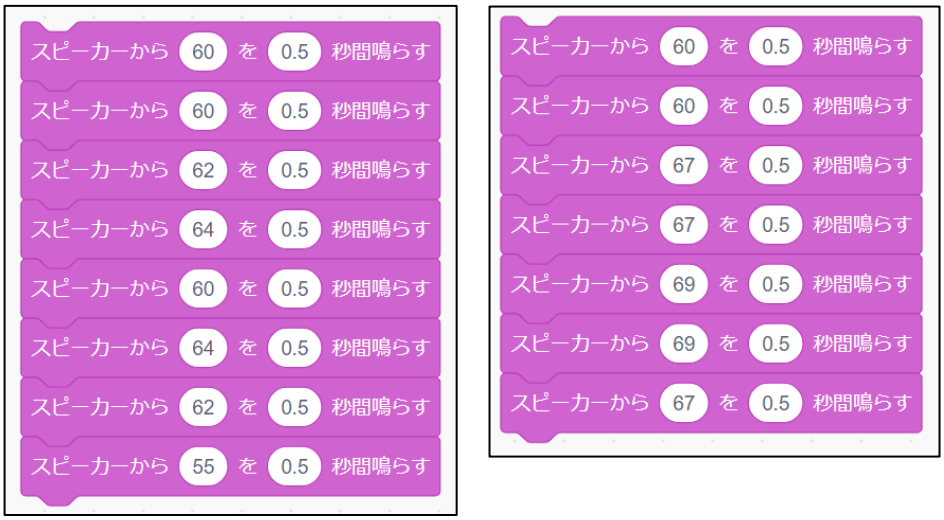
また、 

を実行すると、音を止めることができます。

デフォルトの設定では「60(ド)」が鳴るようになっています。
「60」をクリックすると鍵盤が表示され、鳴らしたい音の鍵盤をクリックすると鳴らす音を変更することができます。



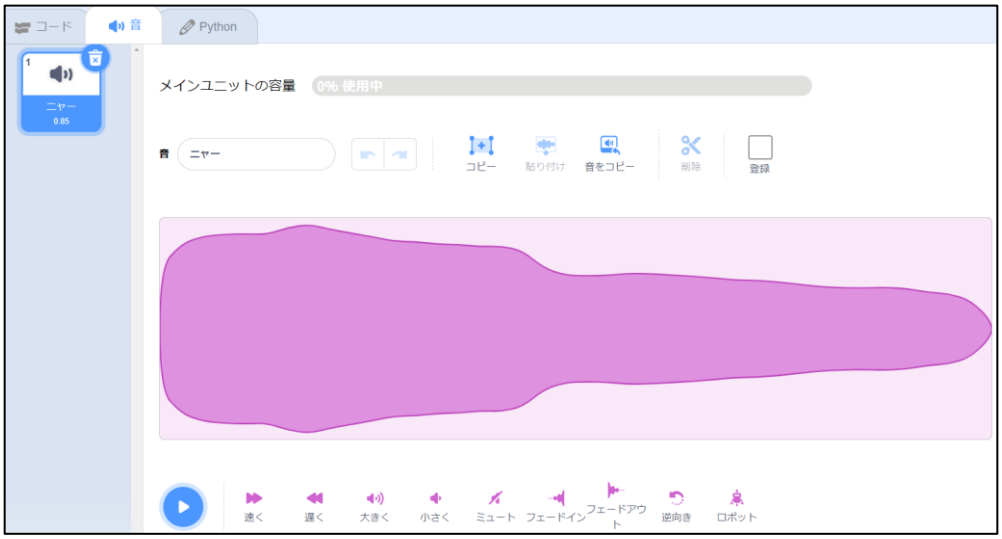
ブロックを順番に並べて実行することで、様々な楽曲を演奏することができます。



ネコの鳴き声を鳴らそう

ソフトウェア上には、様々な音データが用意されています。
これらの音データはスピーカーから再生することができます。

①ブロックパレットの上にある「音」をクリックします。

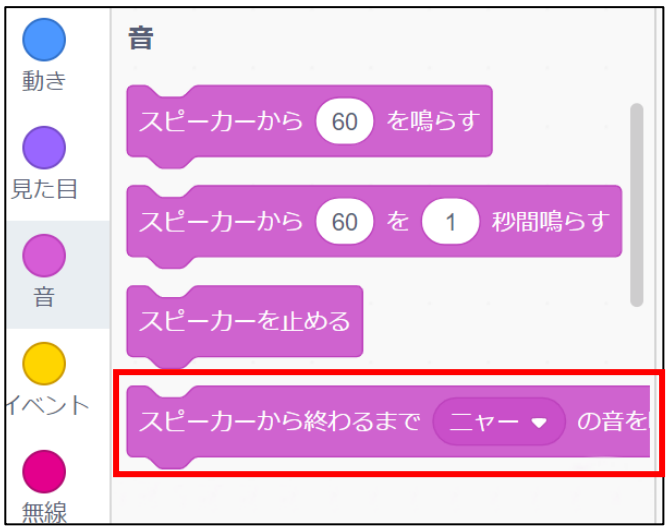


②スピーカーから再生したい音データを選び、「登録」をクリックします。

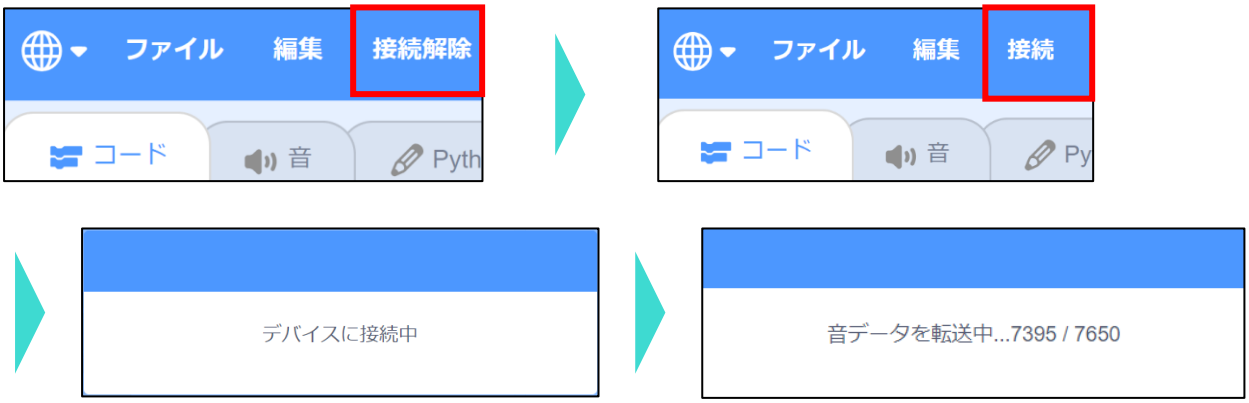
③音データを登録すると、カテゴリーの 

に新しく 

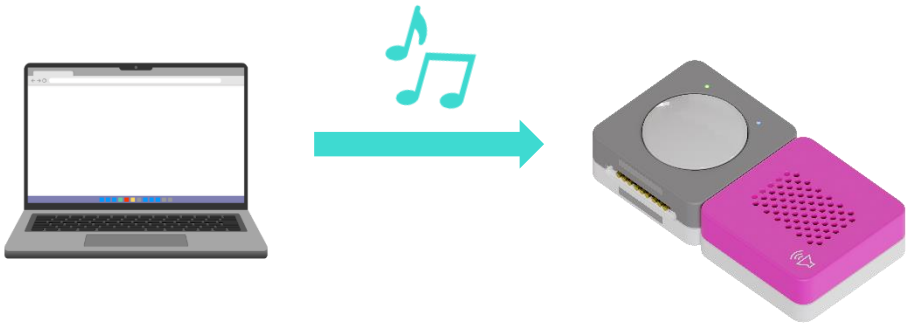
が追加されます。



④画面左上の「接続解除」をクリックして一度接続を解除し、「接続」をクリックして再度接続しましょう。



この操作によってソフトウェア上の音データがメインユニットに転送され、スピーカーから音データを再生できるようになります。



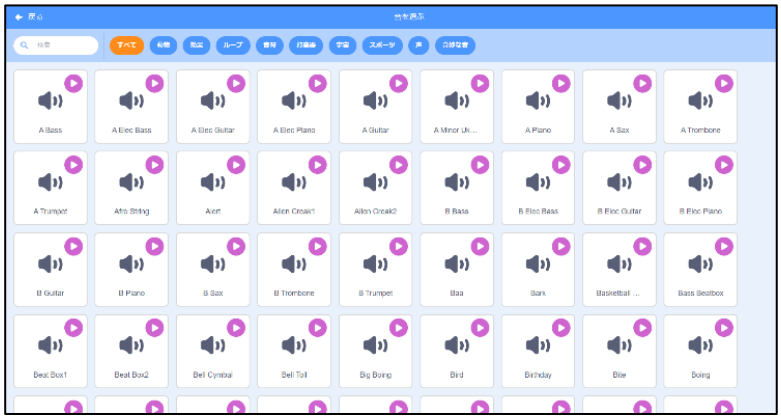
⑤ 

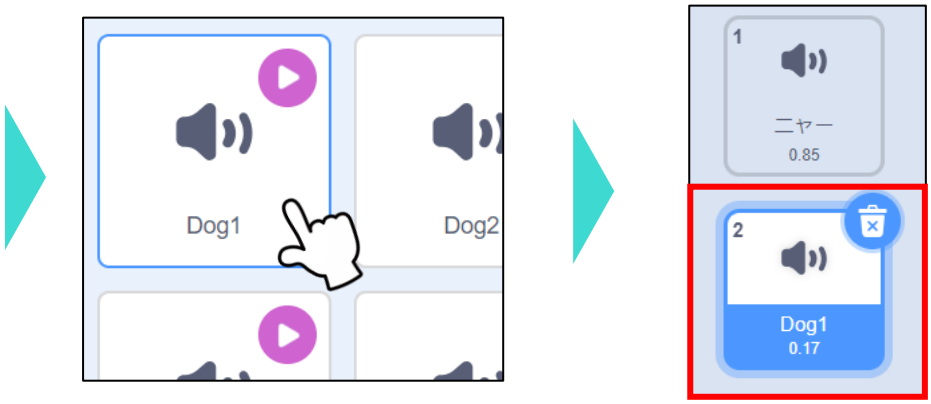
を実行すると、スピーカーからネコの鳴き声が再生されます。

様々な音データを追加しよう

最初から追加されている音データはネコの鳴き声だけですが、その他の音データを追加して再生することができます。

①画面左下の  から「音を選ぶ」をクリックして一覧を表示し、好きな音データをクリックして追加します。



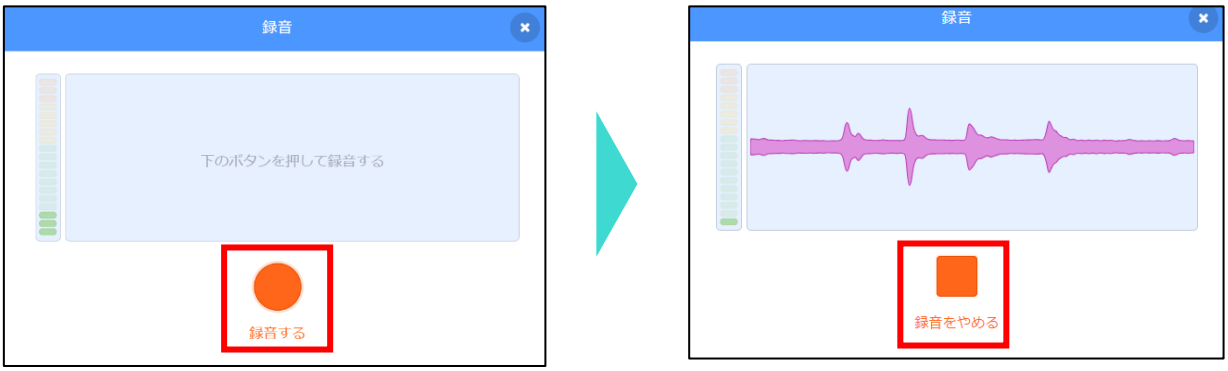


②音声を録音して再生することもできます。画面左下の  から「録音する」をクリックします。

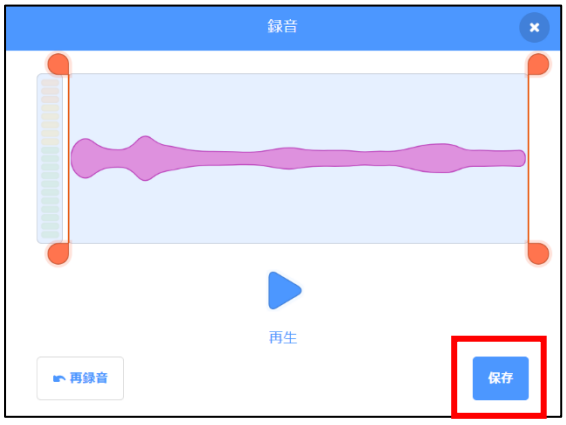
③ 

をクリックすると録音が開始され、 

をクリックするまで録音することができます。

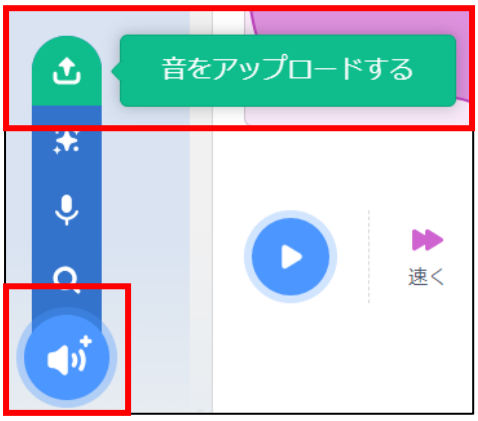


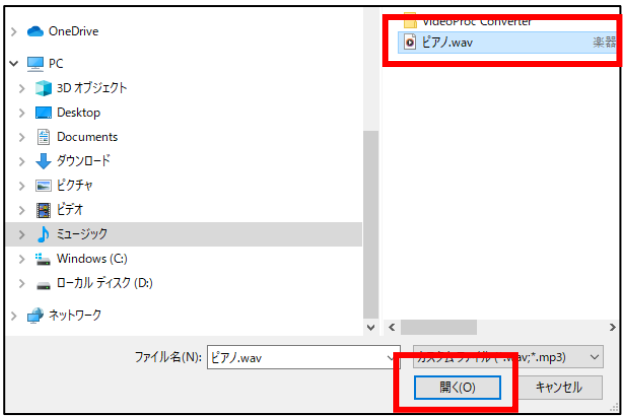
④オレンジ色の線を左右にドラッグして保存したい音の範囲を選択し、「保存」をクリックすると追加できます。



⑤ソフトウェア上の音データだけでなく、デバイスに保存している wav ファイルを追加して再生することもできます。

画面左下の  から「音をアップロードする」をクリックしてフォルダを開き、追加したいファイルを開くと追加できます。





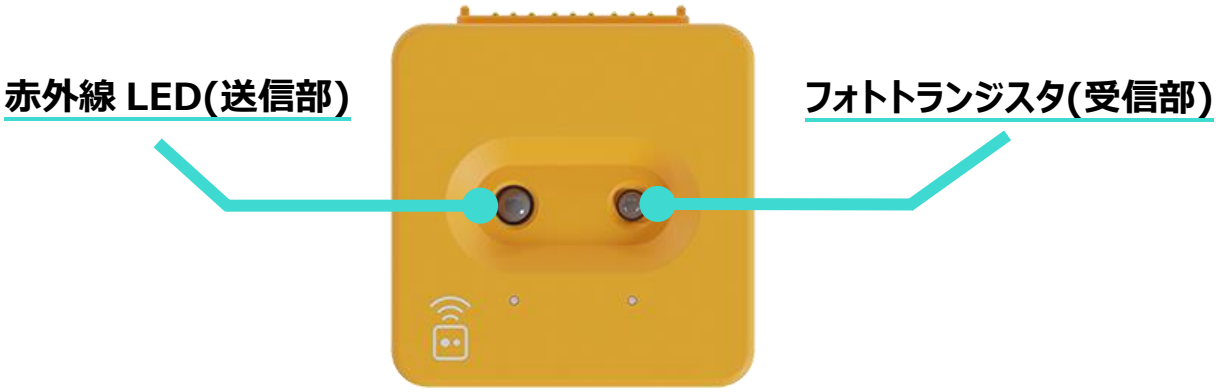
音声ファイル（.wav）は 300KB まで登録することができます。

3 音声で家電制御

3 章では、赤外線通信ユニットとマイクを用いて、音声で家電を操作するプログラムを紹介します。

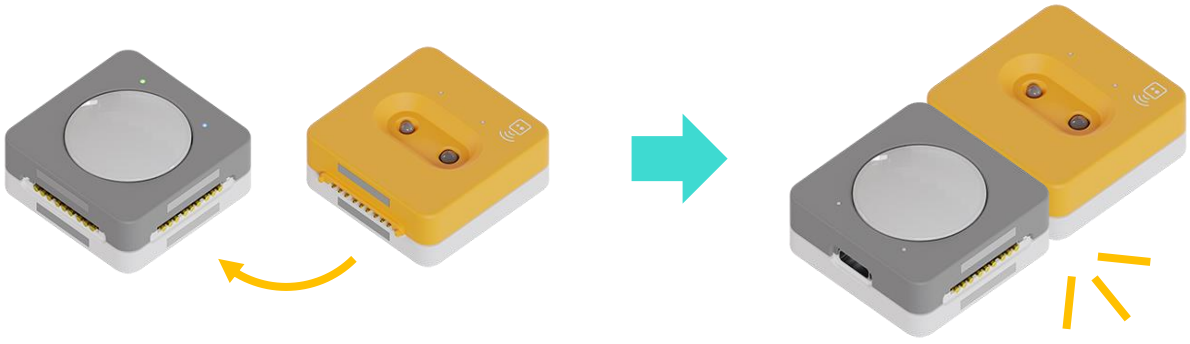
3-1 赤外線通信ユニットで家電を動かそう

赤外線通信ユニットを接続すると、赤外線を送受信することができます。
家電のリモコンなどから送信される赤外線を、フォトトランジスタ（受信部）で受信して、記録することができます。
記録した赤外線信号は、好きなタイミングで赤外線 LED（送信部）から送信することができます。

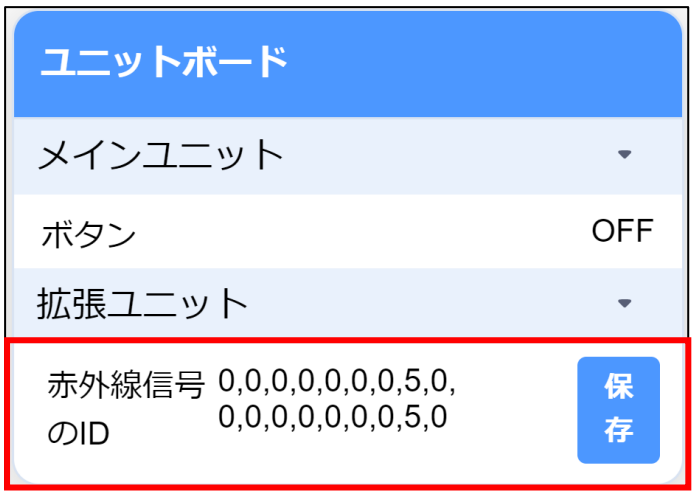


赤外線通信ユニットを接続しよう

①赤外線通信ユニットをメインユニットに接続します。(どの位置に接続しても問題ありません)



②ユニットボードに「赤外線信号の ID」が追加されます。



ユニットボードに「赤外線信号の ID」が表示されない場合は一度拡張ユニットを取り外し、再度接続してください。

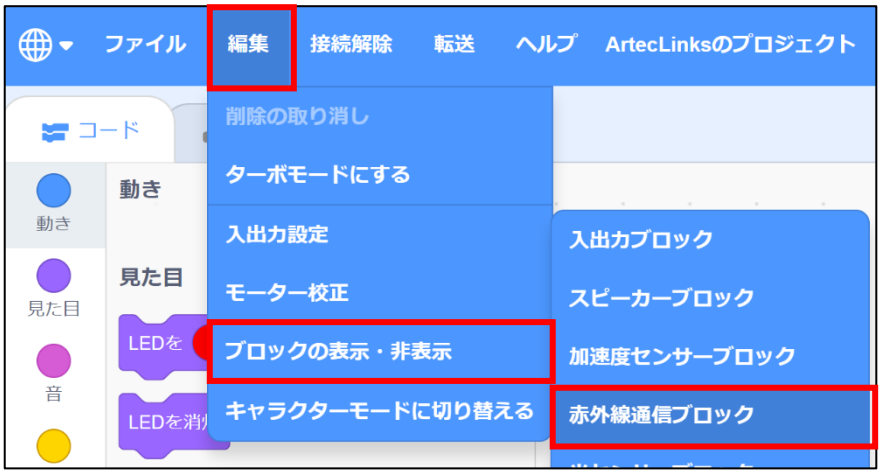
③カテゴリーの 動き に 

が、

カテゴリーの 調べる に 

が追加されていることを確認します。

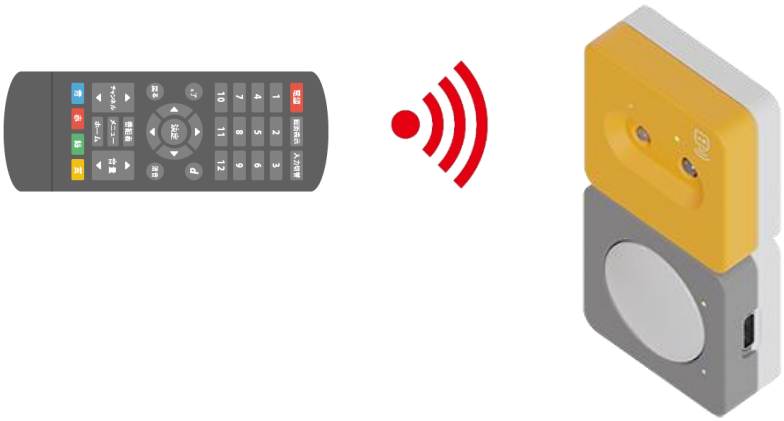
※ブロックが表示されない場合は、画面左上の「編集」から「ブロックの表示・非表示」を選び、
「赤外線通信ブロック」をクリックしましょう。



赤外線信号を登録しよう

家電のリモコンの赤外線をメインユニットに記憶させて、アーテックリンクスで家電を操作できるようにしましょう。

①家電のリモコンを赤外線通信ユニットの受信部に向け、登録したい赤外線信号のボタンを押しましょう。



②正常に赤外線を受信すると、ユニットボードに受信した赤外線信号の ID が表示されます。
「保存」を押して、赤外線信号の ID をメインユニットに保存しましょう。

③名前（例では「テレビの電源」）を付けて「登録」を押すと、ID を登録できます。
登録が完了すると画面に信号と名前が表示されます。

④登録が完了したら  をクリックして登録画面を閉じます。

⑤ 

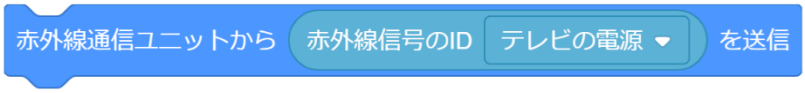
の選択肢に登録した信号が表示されていれば登録完了です。

⑥赤外線を送信したいときは 

の空欄に 

を挿入して

実行すると、送信部から指定した赤外線が送信されます。



※ 

は、あらかじめ記録されているサンプルの赤外線信号です。
赤外線信号の ID1~3 を送信しても、家電を操作することはできません。

赤外線リモコンをつくろう

メインユニットのボタンで家電を操作するプログラムを作成します。

①本体のボタンが押されたときに、赤外線通信ユニットから赤外線を送信するスクリプトを作成します。

②操作したい家電に赤外線リモコンの送信部を向けてメインユニットのボタンを押すと、登録した赤外線が送信されて家電を操作することができます。

3-2 マイクが計測した音で家電を動かそう

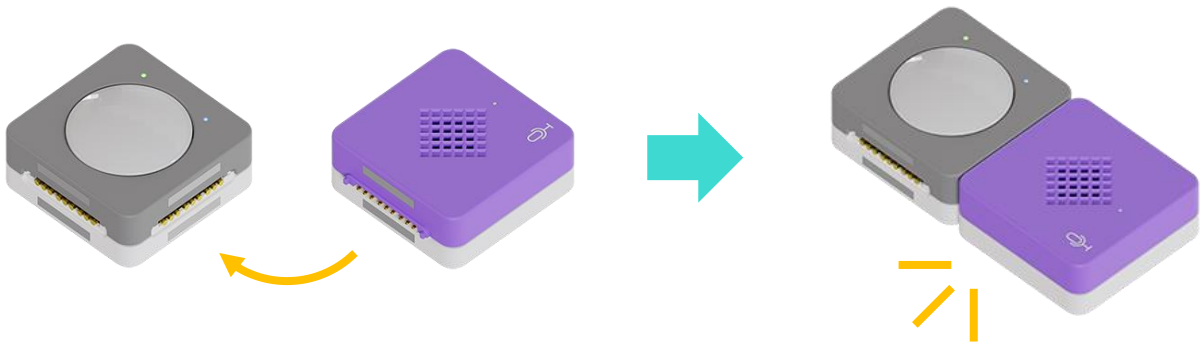
マイクを接続すると、周囲の音量を計測することができます。
音量は 0～100 の数値で表され、音量が大きいほど数値も大きくなります。

マイクで測定した音量をプログラムで分析して、音が鳴った回数によって異なる家電を操作するプログラムをつくります。

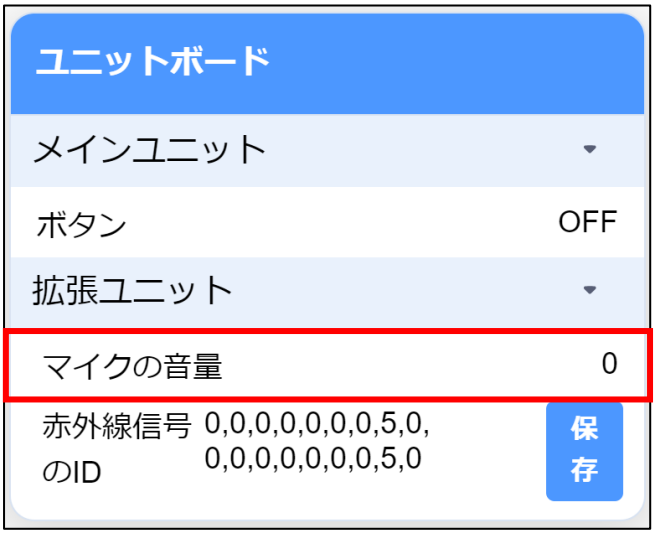
マイクを接続しよう

メインユニットにマイクを接続して、音量を計測できる状態にしましょう。

①マイクをメインユニットに接続します。(どの位置に接続しても問題ありません)



②ユニットボードに「マイクの音量」が追加されていることを確認します。



ユニットボードに「マイクの音量」が表示されない場合は一度拡張ユニットを取り外し、再度接続してください。

③カテゴリーの 

に 

が追加されていることを確認します。

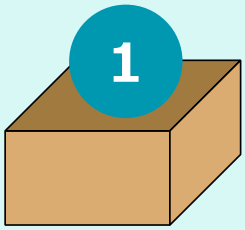
※ブロックが表示されない場合は、画面左上の「編集」から「ブロックの表示・非表示」を選び、
「マイクブロック」をクリックしましょう。

リストに音量データを保存しよう

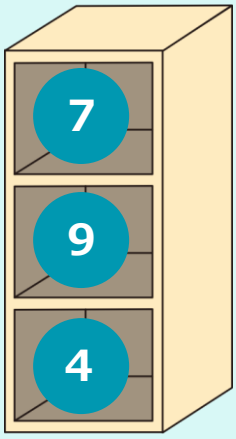
マイクをつかって音量のデータを保存してみましょう。
今回は新しく「リスト」を作成して、音量のデータを保存します。

リストとは

数値や文字などのデータを保存できる「棚」です。
変数は 1 つのデータしか保存できませんが、リストは複数のデータを保存することができます。



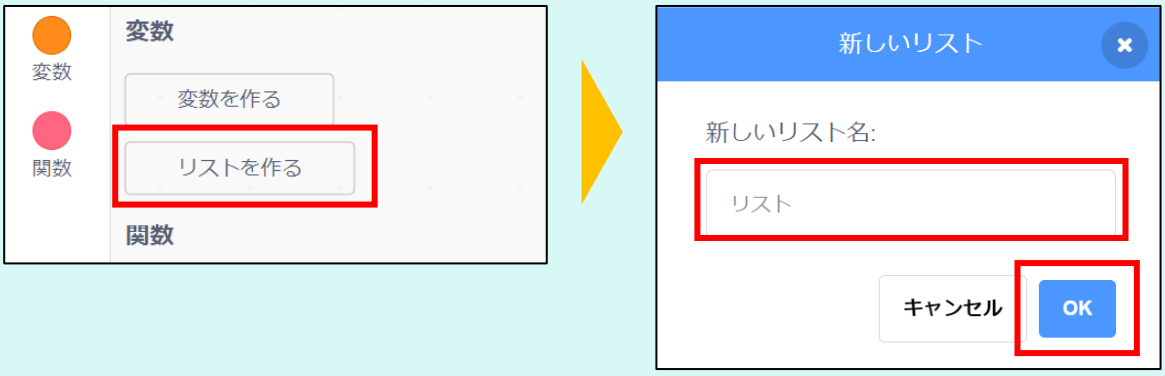
変数



リスト

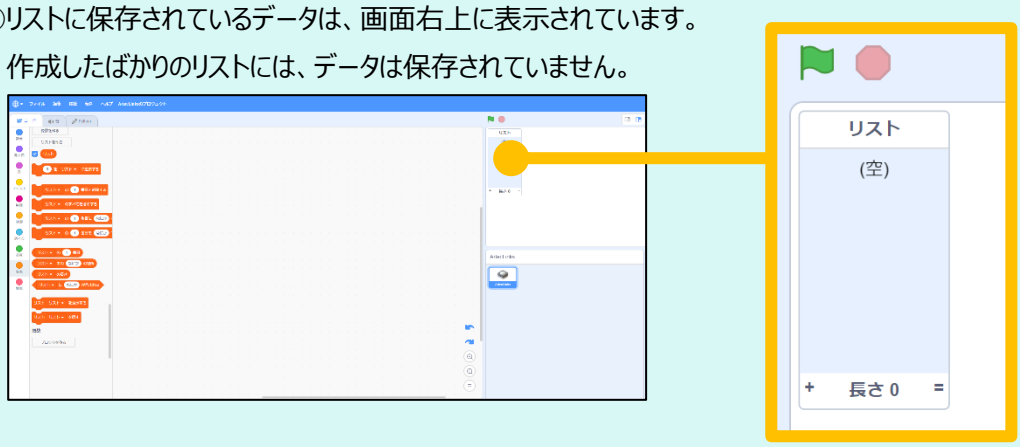
①カテゴリーの 

から「リストを作る」を選択し、名前を入力して「OK」をクリックして作成します。



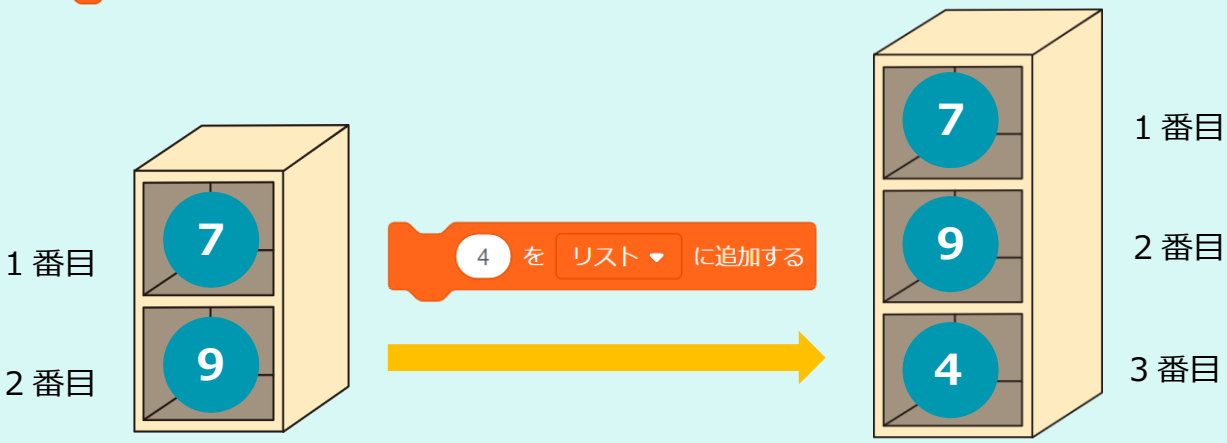
②ブロックパレットに「リスト」のブロックが表示されていることを確認しましょう。

③リストに保存されているデータは、画面右上に表示されています。
作成したばかりのリストには、データは保存されていません。

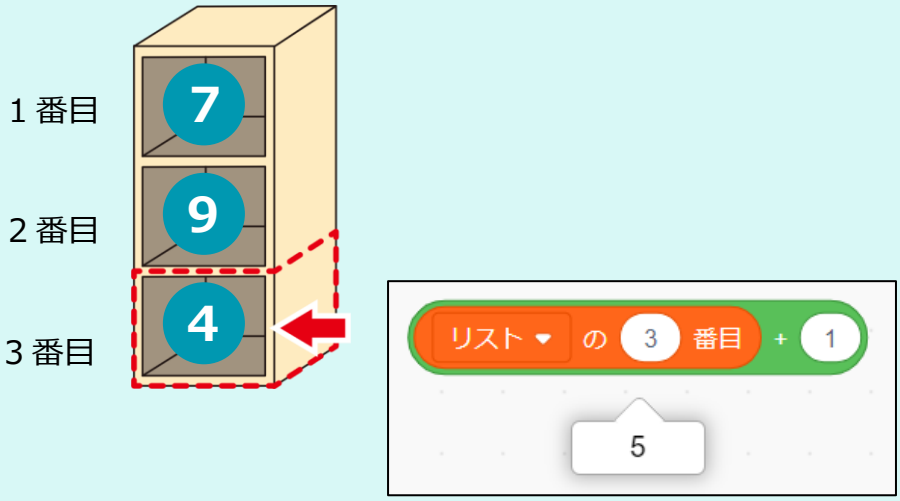


④ 

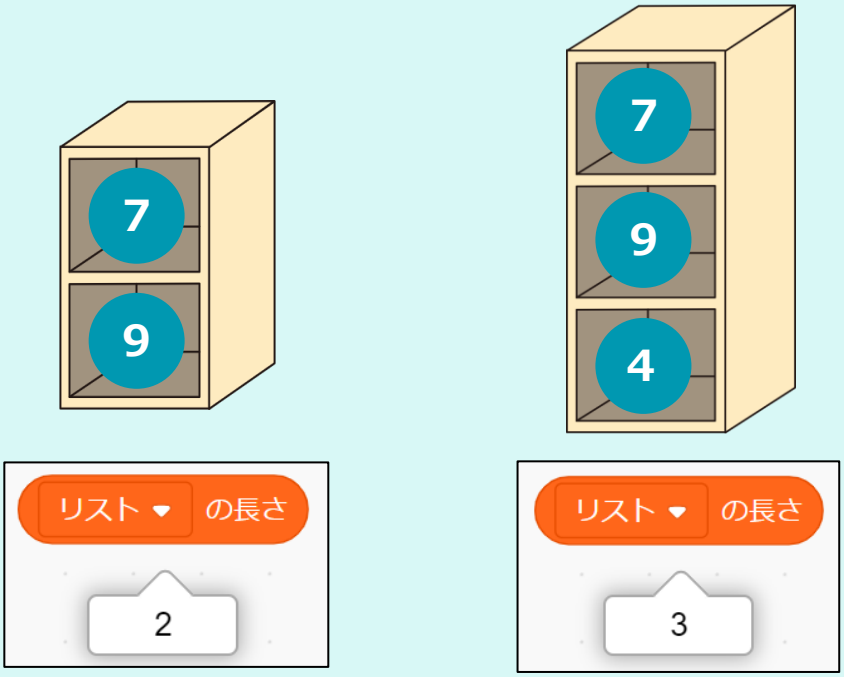
を実行すると、リストにデータを追加することができます。



リストに保存されたデータは、それぞれ 1 番目、2 番目、3 番目と自動で番号が割り振られます。
その番号をブロックで指名すると、必要なデータだけを抜き出して利用することができます。



「リスト」ブロックの多くは変数のブロックと同じように使えますが、リストの特徴的な使い方として、
プログラム内でリストの長さ（リストに保存されているデータの総数）を使用することができます。

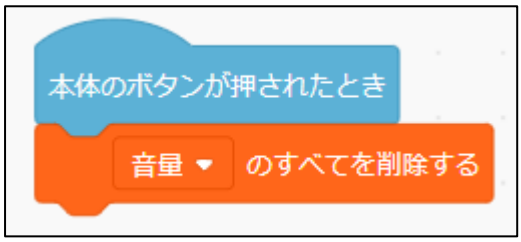


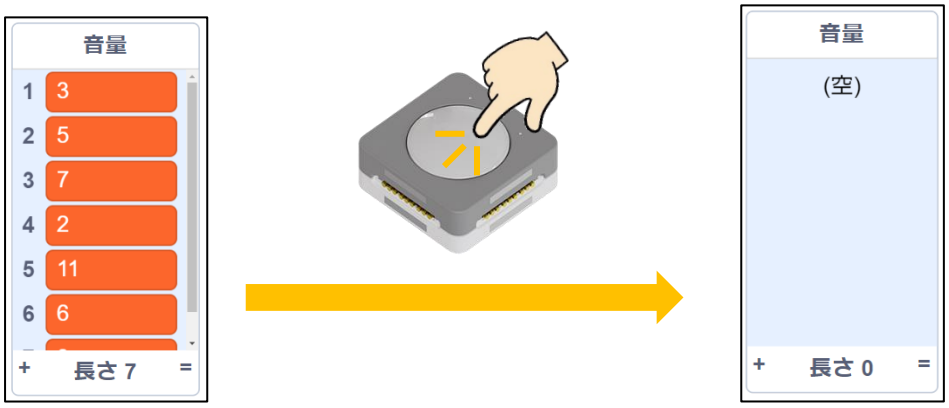
また、リストの指定した番号のデータを、指定したデータで置き換えることもできます。

それでは、音量のデータをリストに保存するスクリプトを作成しましょう。

①リスト「音量」を作成します。

②ボタンが押されたときにリスト「音量」をすべて削除します。





③マイクの音量をリスト「音量」に 0.05 秒間隔で 20 回追加します。

④カテゴリーの 演算 にある ＞ の左の空欄に マイクの音量 を、右の空欄に 10 を入力して、
カテゴリーの 制御 にある まで待つ の空欄にドラッグします。

このスクリプトが実行されると、マイクが音を検知する（マイクの音量が 10 より大きくなったとき）まで、次のブロックの実行を待ちます。
そのため、マイクが音を検知してすぐに次のブロックを実行することができます。

⑤④で作成したスクリプトを、③で作成したスクリプトの繰り返しブロックの前に挿入します。
また、ボタンを押した音で誤作動しないように 秒待つ を追加します。

このスクリプトが実行されることで、マイクが音を検知したとき（マイクの音量が 10 より大きくなったとき）から、1 秒間（0.05 秒×20 回）の音の大きさを保存することができます。

⑥ボタンを押した後に、リストに音量を追加する準備ができているか確認するために、

が実行される前に LED を点灯し、音量を追加し終わったら LED を消灯します。

⑦スクリプトを実行し、ボタンを押して数回拍手してみましょう。

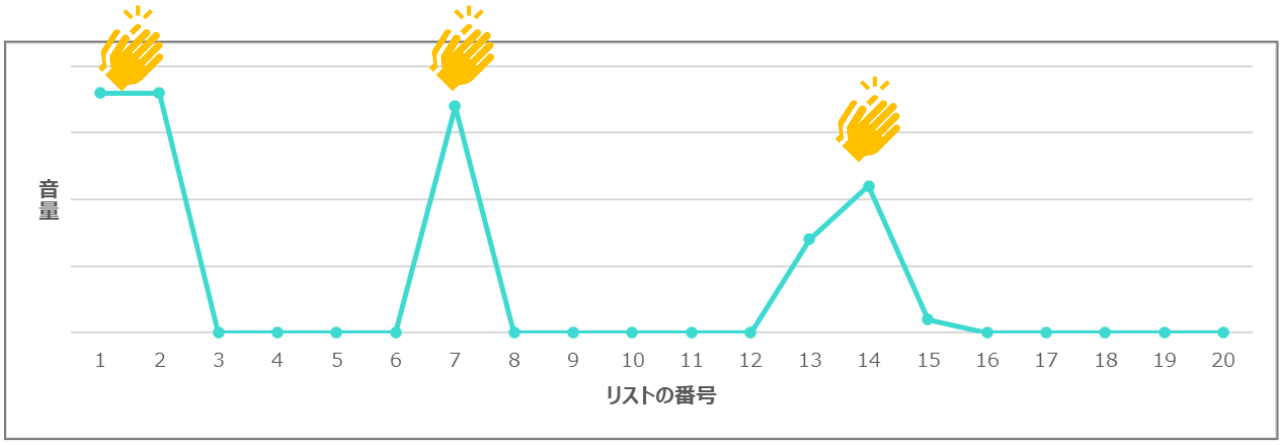
LED が点灯している間に検知した拍手の音量が、リスト「音量」に保存されます。

音が鳴った回数を数えよう

保存した音量のデータをもとに、1 秒間に音が鳴った回数を数えるプログラムを作成します。
次の例を参考にプログラムの仕組みを考えてみましょう。

例）3 回拍手したときに保存された音量データ

リストの番号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
音量	**18**	**18**	**0**	**0**	**0**	**0**	**17**	**0**	**0**	**0**	**0**	**0**	**7**	**11**	**1**	**0**	**0**	**0**	**0**	**0**



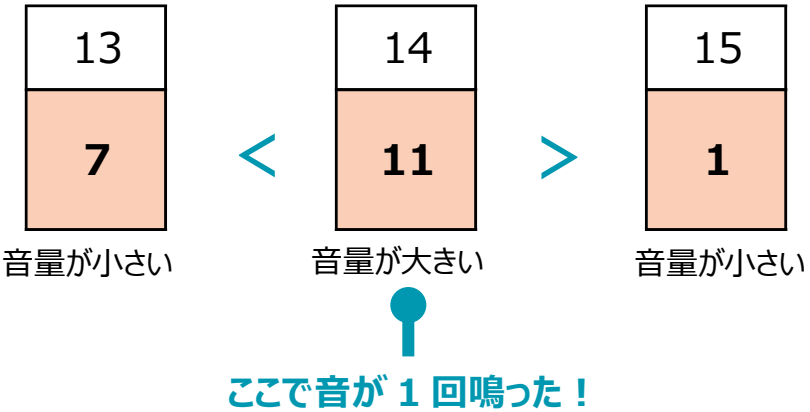
この例では測定時間中に 3 回拍手を行いましたが、
音の鳴り方によっては、音量のデータが複数のリストの番号にまたがって保存されています。

リストの番号	1	2
音量	**18**	**18**

リストの番号	13	14	15
音量	**7**	**11**	**1**

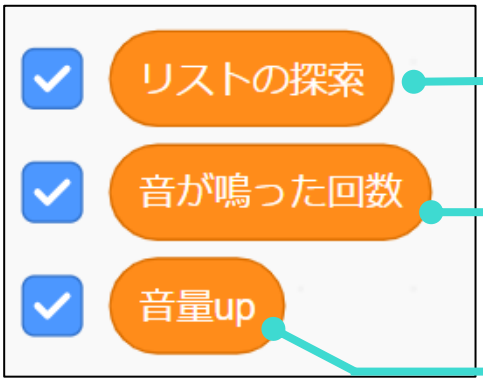
このような音量のデータから音が鳴っている回数を計測するためには、リストの前後のデータを比較して
リストの 1 つ前より音量が大きい　&　リストの 1 つ後より音量が大きい　場合に、
音が 1 回鳴ったと判断されるようにします。

例）リストの番号：14



それでは、この考えをスクリプトで表現してみましょう。

①変数「音量 up」、「リストの探索」、「音が鳴った回数」を作成します。

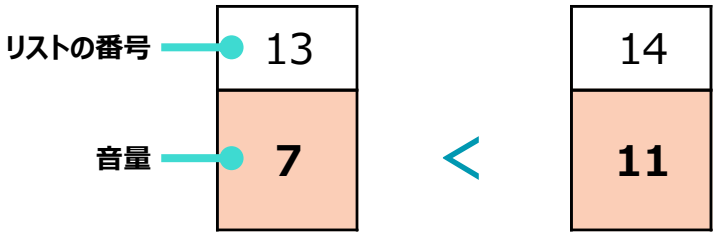


リストの番号を指定して、指定した番号で音が鳴ったかどうかを調べます。

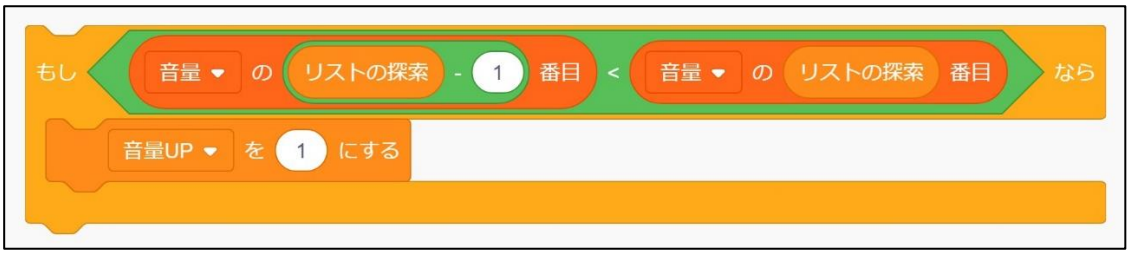
音が鳴った回数を数えます。

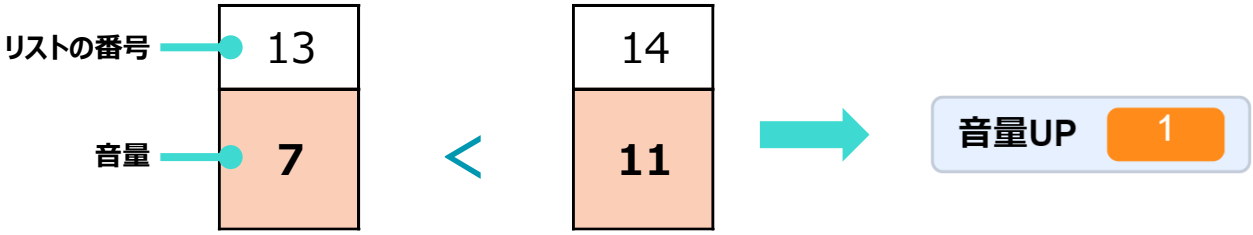
『リストの 1 つ前より音量が大きい』場合に「1」を入力します。
そうでない場合は「0」を入力して、プログラム上で区別します。

②「リストの探索」番目に保存されている音量が、1 つ前の番号の音量より大きいか調べるためのスクリプトは次のようになります。

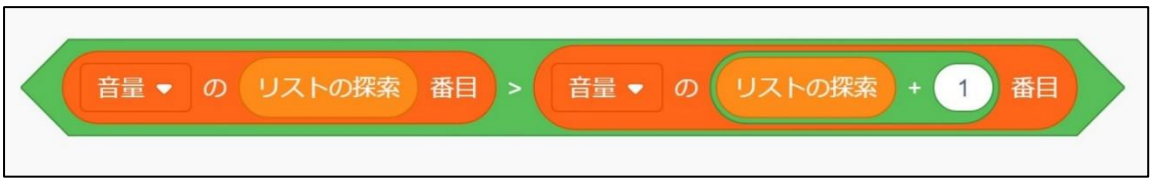


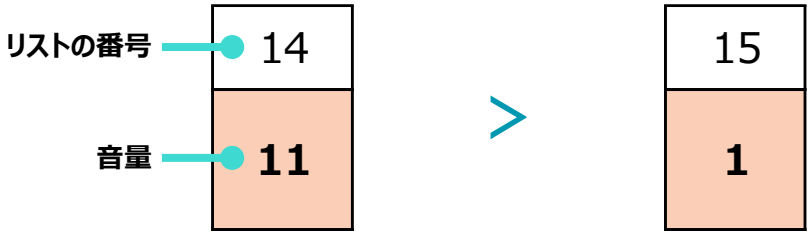
③リストの 1 つ前の番号より音量が大きいとき、変数「音量 up」を 1 にします。



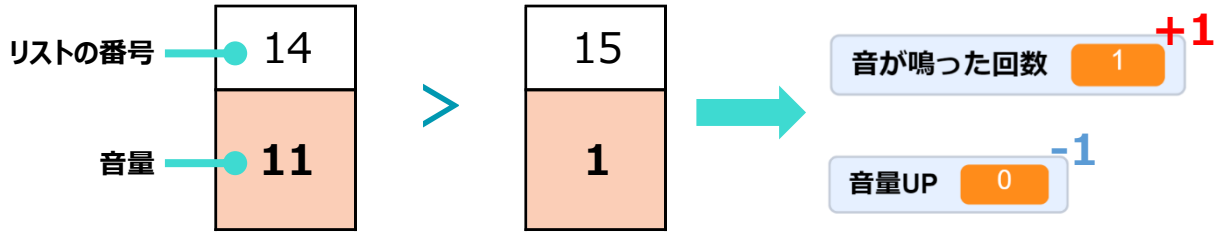


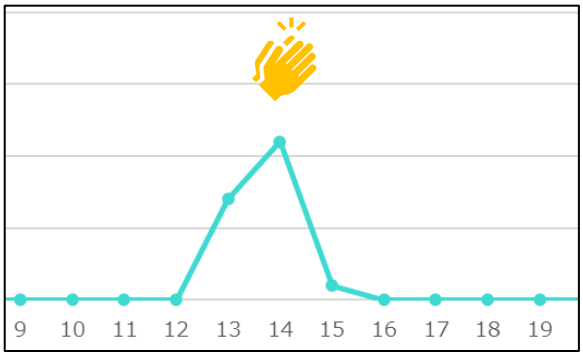
④「リストの探索」番目に保存されている音量が、1 つ後の番号の音量より大きいか調べるためのスクリプトは次のようになります。



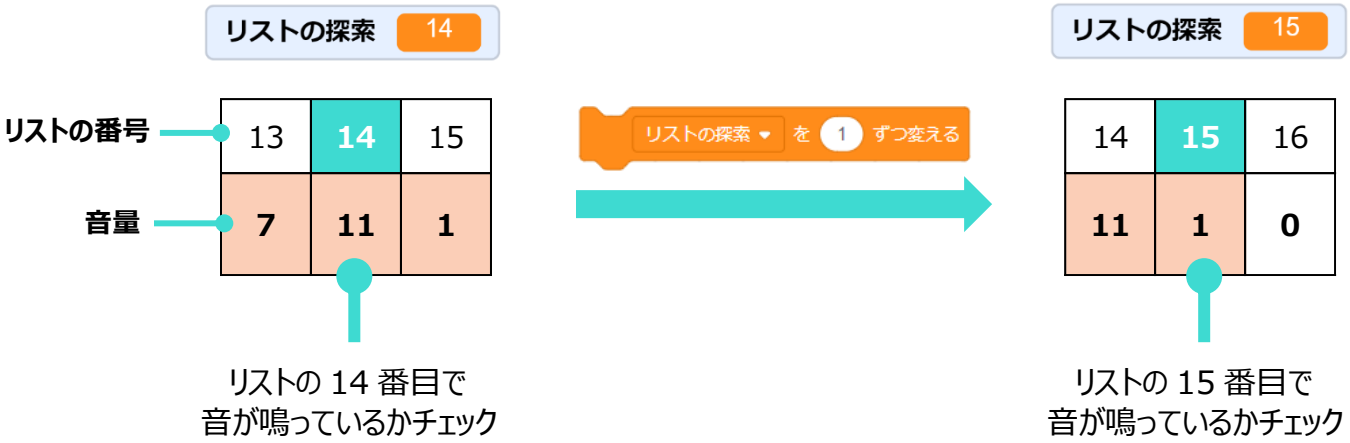


⑤リストの 1 つ後の番号のデータより大きく、1 つ前の番号のデータより大きい（変数「音量 up」が 1）とき、変数「音が鳴った回数」に 1 を足して、回数を数えます。
また、リストの 1 つ前の番号より音量は小さくなったため、変数「音量 up」は 0 に戻します。





⑥音が鳴ったかどうか判別した次は、リストの次の番号に保存されているデータを判別するために、変数「リストの探索」に「1」を足します。

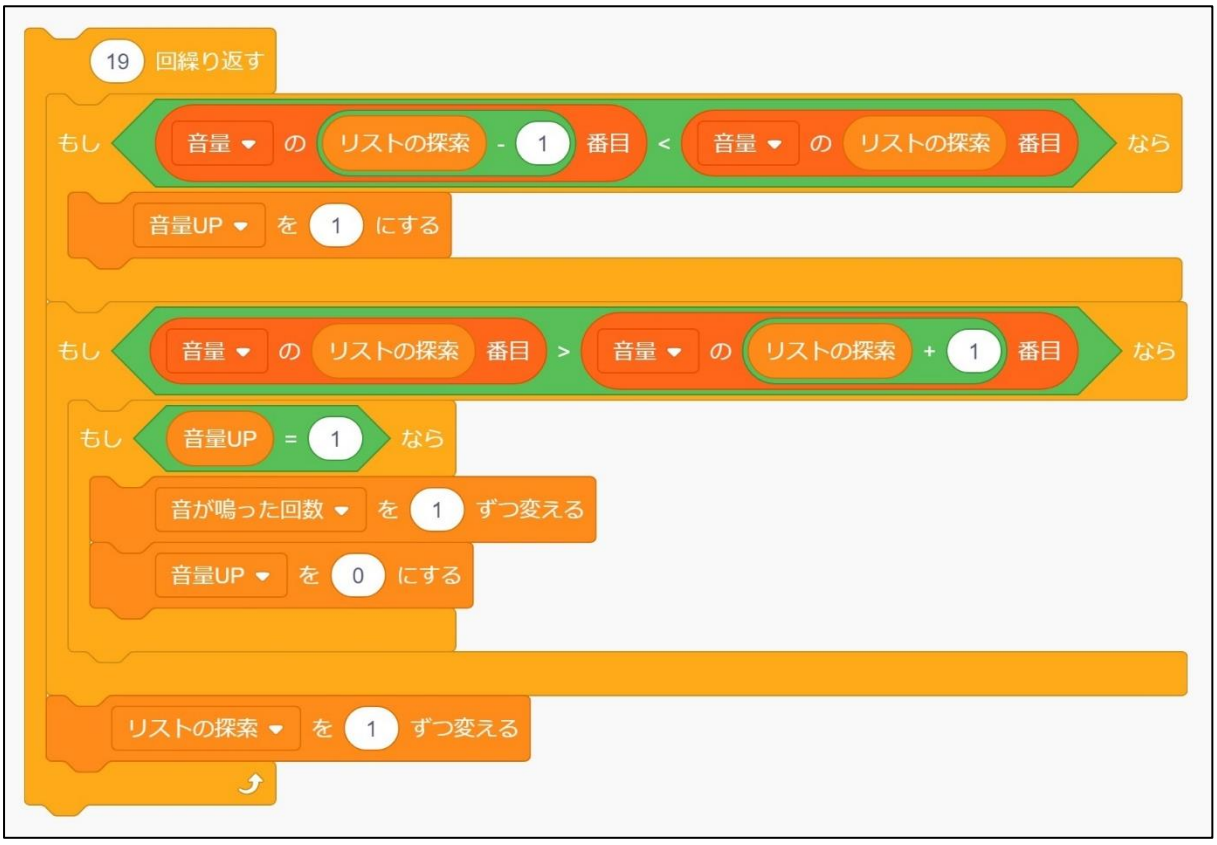


リストの 14 番目で
音が鳴っているかチェック

リストの 15 番目で
音が鳴っているかチェック

⑦すべてのリストの番号で音が鳴ったか確認するために、②～⑥で作成したスクリプトを 19 回繰り返します。

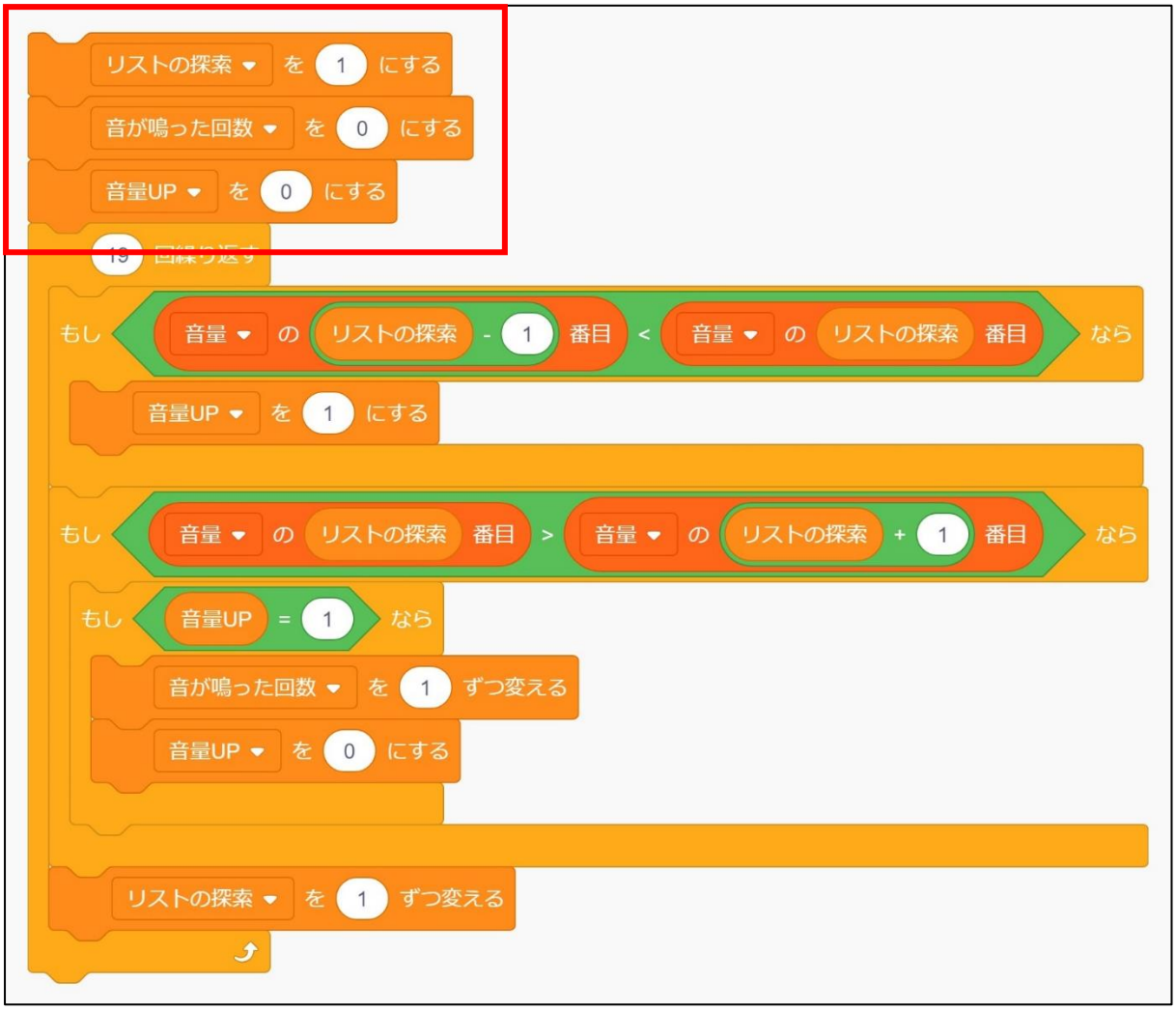
※繰り返し回数が 20 回だと、存在しない 21 番目のリストを探し続けて実行が終わらないことがあります。
繰り返す回数はリストの長さより「1」だけ少ない数を入力してください。



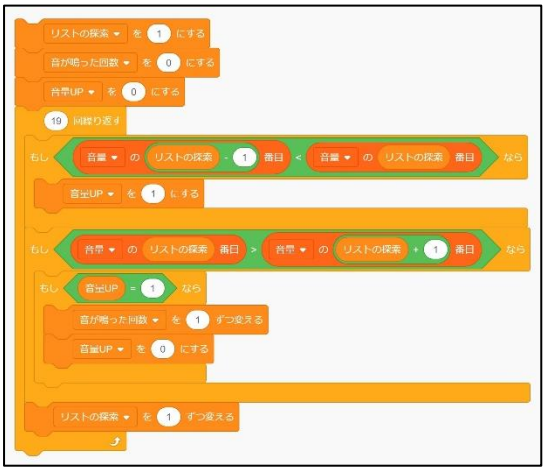
⑧プログラムが実行されたとき、最初に変数に保存されている数値を設定します。

変数「音量 up」「音が鳴った回数」に 0 を入力します。

変数「リストの探索」には、リストの番号が 1 から始まるため、1 を入力します。



このスクリプトが実行されると、1 秒間に鳴った音の回数が変数「音が鳴った回数」に保存されます。



音が鳴った回数で異なる赤外線を送信しよう

音が鳴った回数によって異なる赤外線を送信するスクリプトを作成して、拍手の回数によって異なる家電を操作しましょう。

制御したい家電の赤外線信号は、p.37 を参考にして、あらかじめ登録して下さい。
※後から赤外線信号を追加することも可能です。

①拍手の回数が 1 回（変数「音が鳴った回数」が 1）のときに、「テレビの電源」の赤外線を送信します。

②同じように、変数「音が鳴った回数」が 2 のとき、3 のときに異なる赤外線を送信します。

③どの赤外線を送信しているか見た目で判別できるように、それぞれ異なる色で LED を点灯させます。
また、赤外線送信後も LED が点灯したままにならないように、赤外線を送信した後に時間を空けて消灯させます。

④③で作成したスクリプトを、音量を測定するスクリプトと、音が鳴った回数を数えるスクリプトの下に繋げましょう。

この状態のスクリプトでも問題なく実行できますが、「関数」を使ってそれぞれのスクリプトを機能ごとに分類すると、プログラム全体が見やすくなります。

関数とは

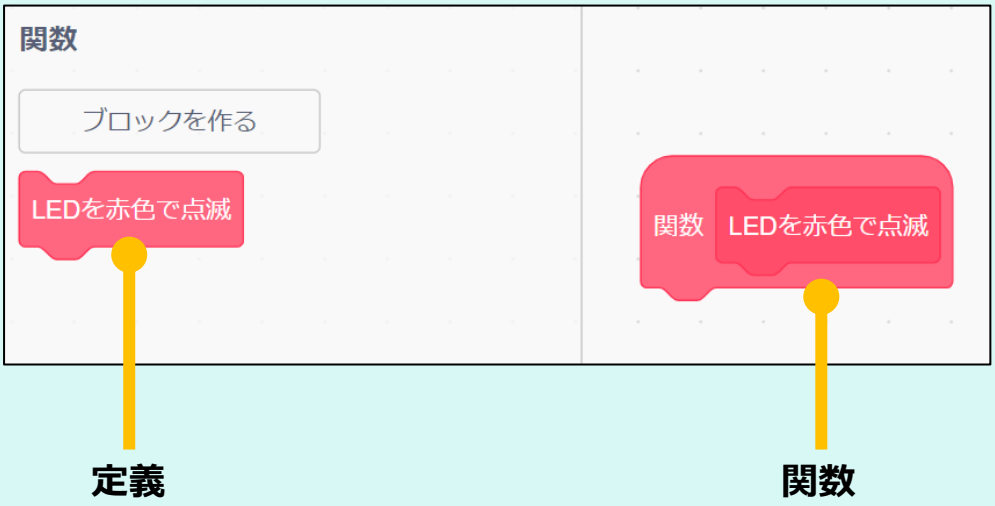
関数とは、スクリプトをまとめておける機能です。
長いスクリプトや何度も使いたいスクリプトを関数でまとめておくことで、後からプログラムを修正・変更するときにわかりやすくなります。

①カテゴリーの 

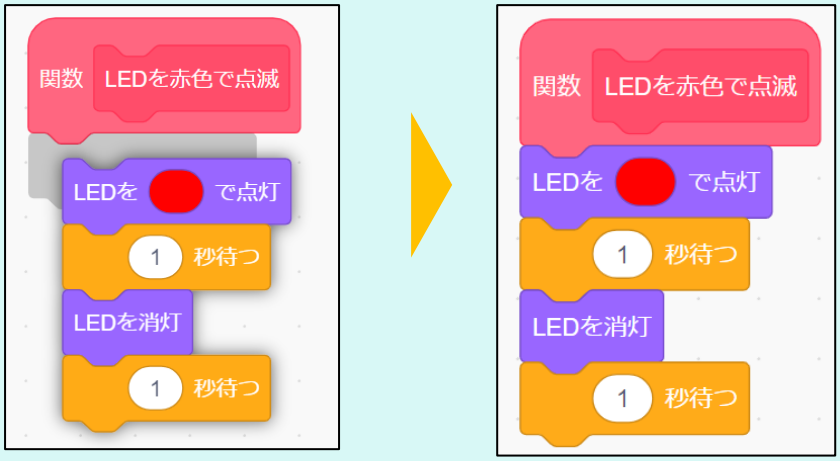
から「ブロックを作る」を選択します。

好きな名前を入力して「OK」をクリックすると関数を作成できます。

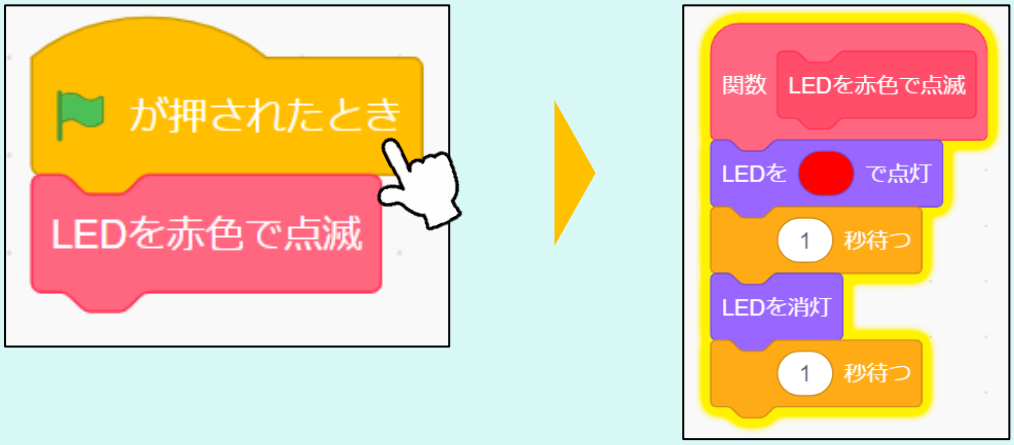
②関数を作成すると、「関数」と「定義」の 2 種類のブロックが表示されます。



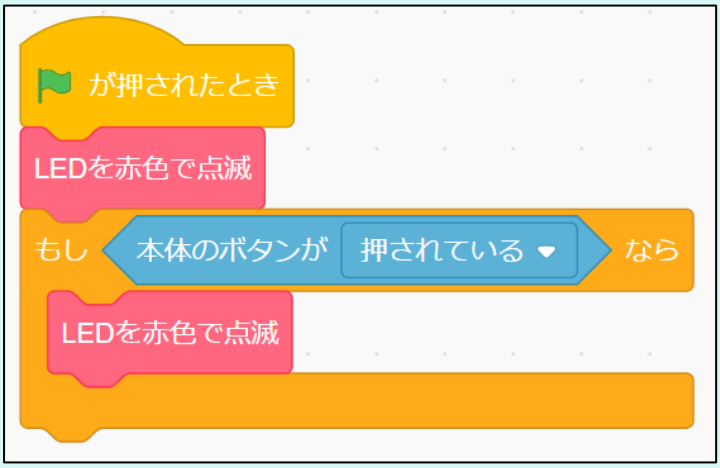
③「関数」ブロックの下にまとめたいスクリプトを繋ぎます。



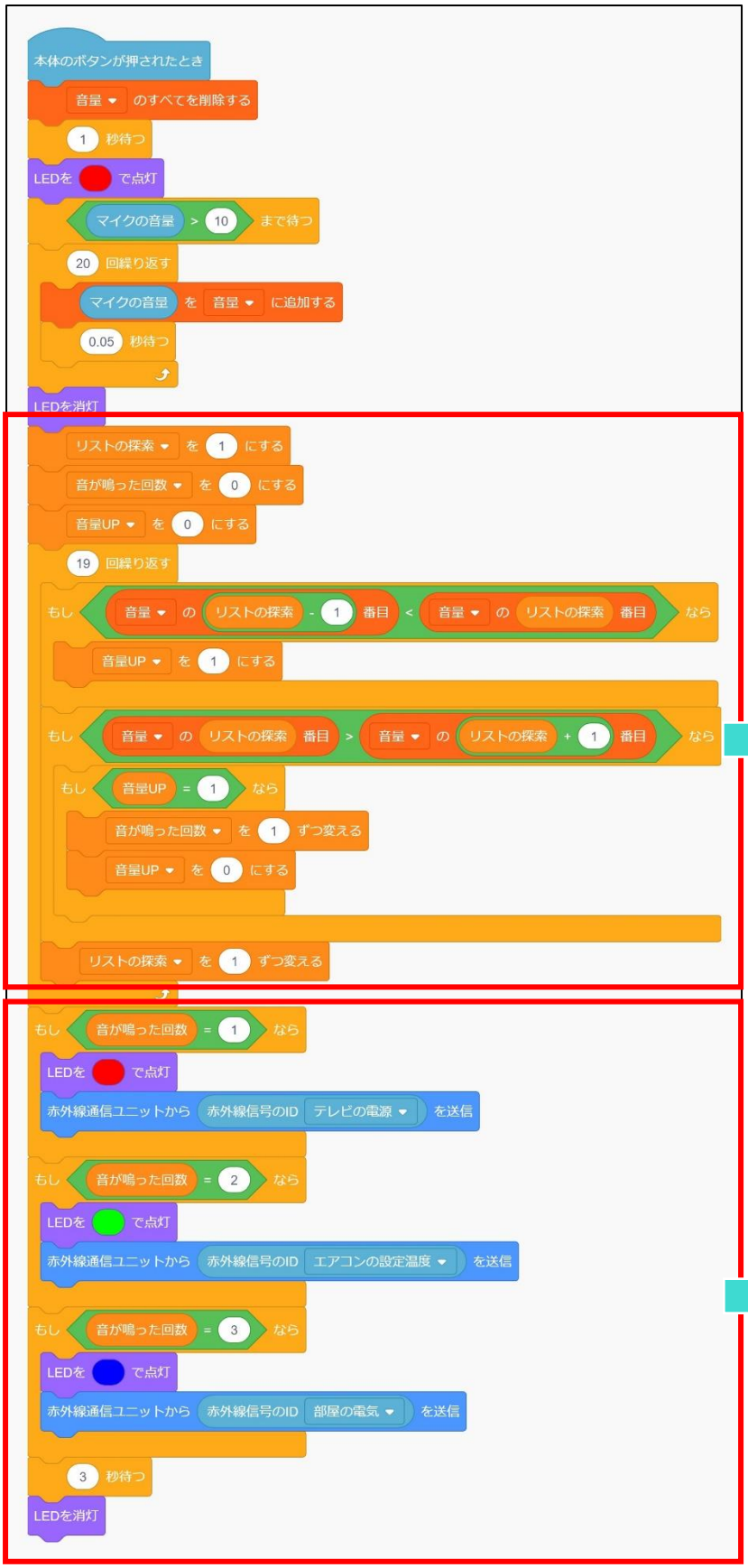
④「定義」ブロックが実行されると、「関数」ブロックの下に繋げたスクリプトが実行されます。



関数はプログラムの中で、何回でも実行することができます。



作成したスクリプトを機能ごとに関数にまとめましょう。



関数 音が鳴った回数を数える

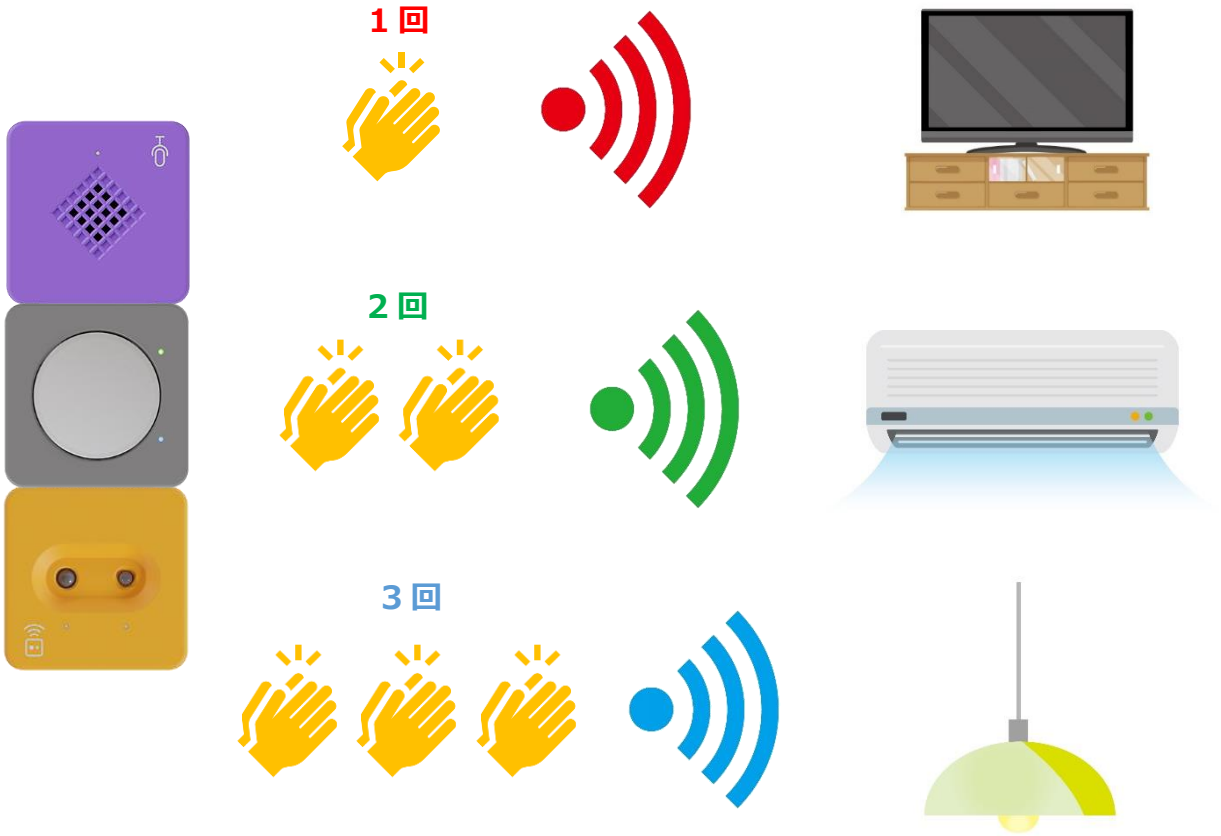
関数 赤外線を送信する

⑤関数「赤外線を送信する」を作成して、③で作成したスクリプトを下に繋ぎます。

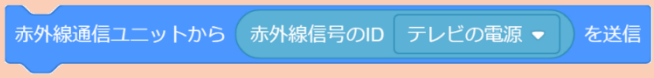
⑥関数「音が鳴った回数を数える」を作成して、p.52 で作成した音が鳴った回数を数えるスクリプトを下に繋ぎましょう。

⑦定義「音が鳴った回数を数える」と「赤外線を送信する」を、
P.47 で作成した音量を保存するスクリプトの下に接続します。

プログラムを実行して、拍手の回数によって異なる家電を制御してみましょう。



プログラムを実行してもうまく操作できない場合は、赤外線通信ユニットの送信部を制御したい家電に向け、



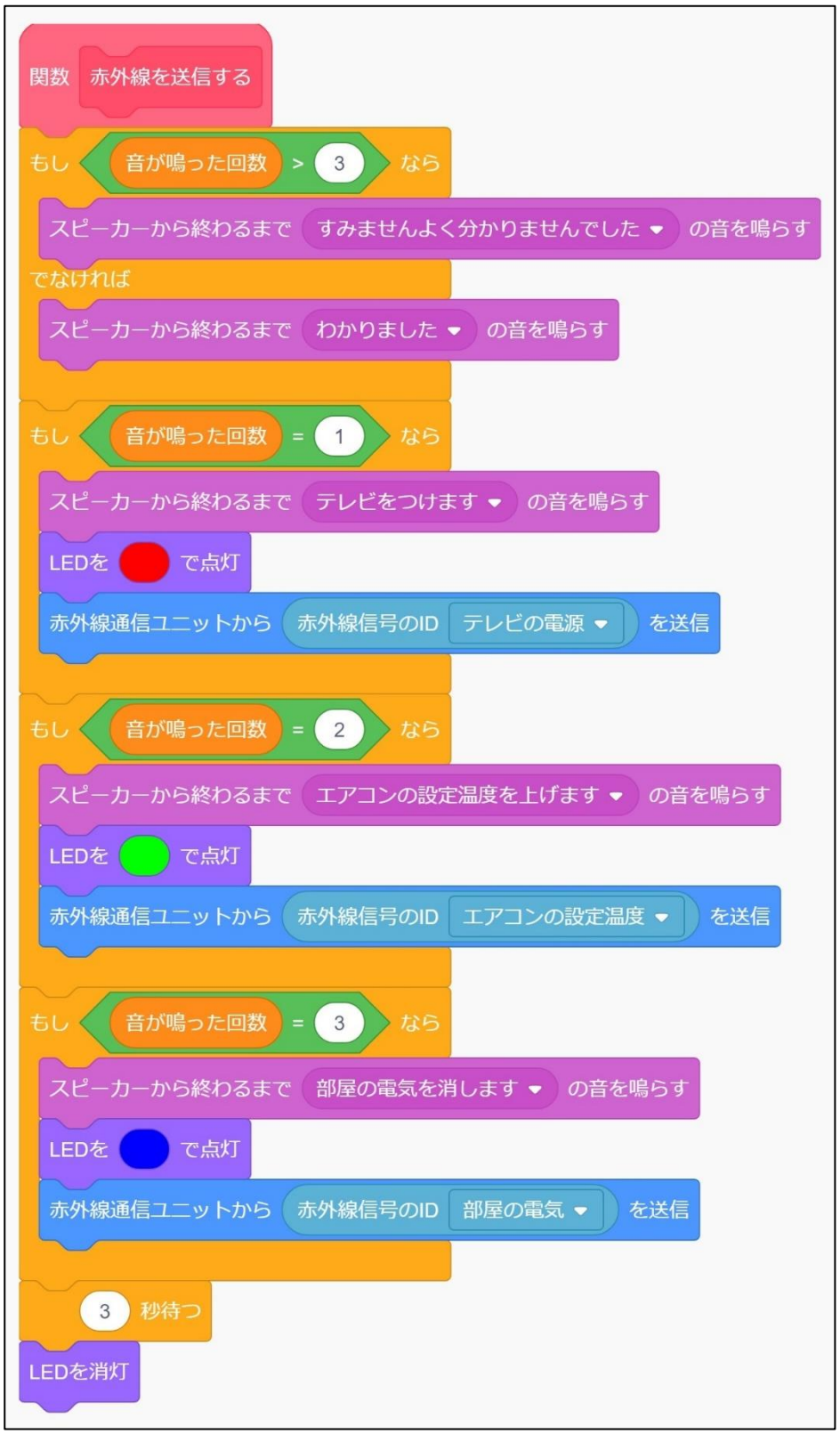
を実行して、家電を制御できるか確認してください。

制御できない場合は、赤外線が正しく登録されていない可能性があります。 → P.37 へ

スピーカーから音声を流して返事をさせよう

自分で音声を録音したり、合成音声のファイルを追加したりして、拍手した後に返事をするプログラムにアレンジできます。

関数「赤外線を送信する」のスクリプトに音を鳴らすブロックを追加することで、それぞれの赤外線を送信する前にスピーカーから返事をさせることができます。



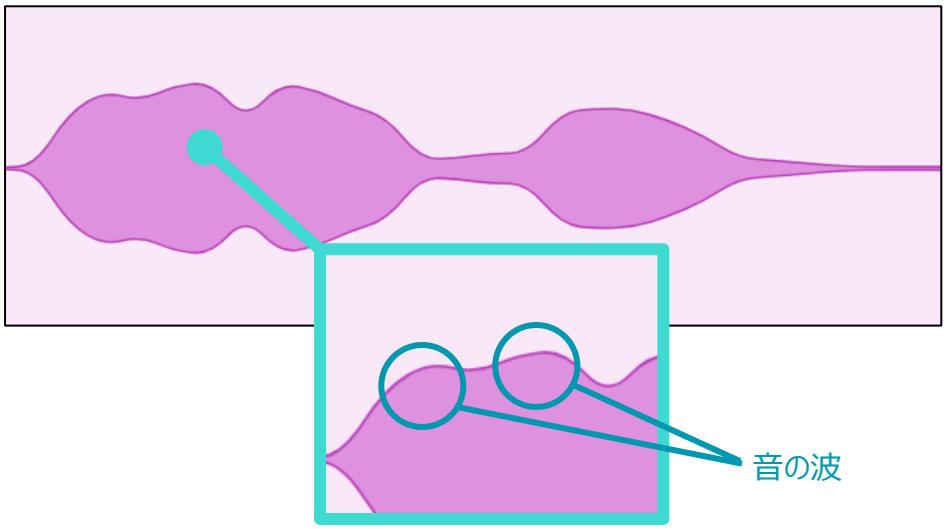
※プログラム内で使用する音声は自身の声を録音したり、音声ファイルを追加したりしましょう。

3-3 音声の違いを区別して家電を動かそう

3-2（p.52）で作成したスクリプトを用いると、簡易的に音声を区別させることができます。

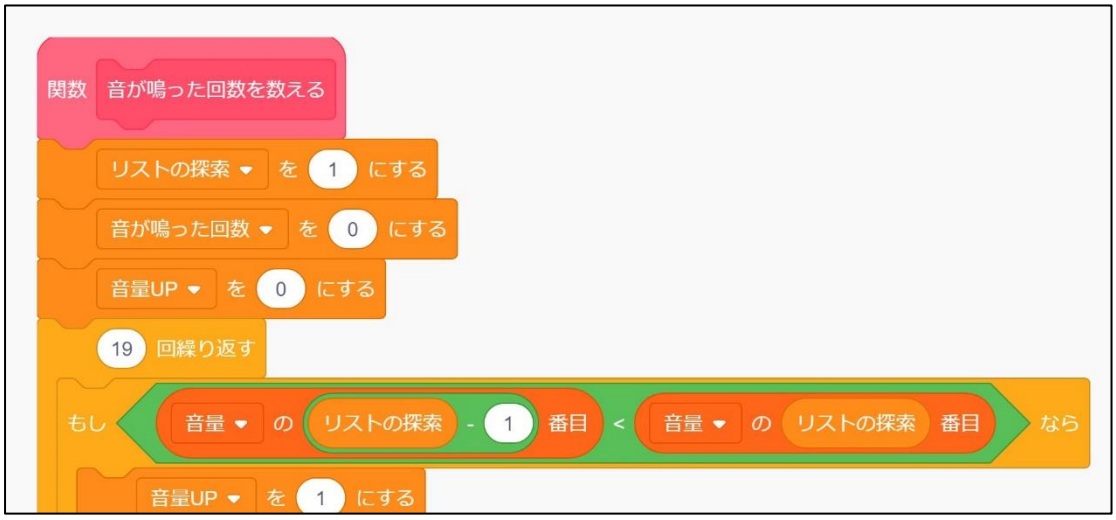
例として、音 から「テレビをつけて」という音声を録音してみましょう。

測定した音声のデータのグラフを見ると、以下のような図になっています。

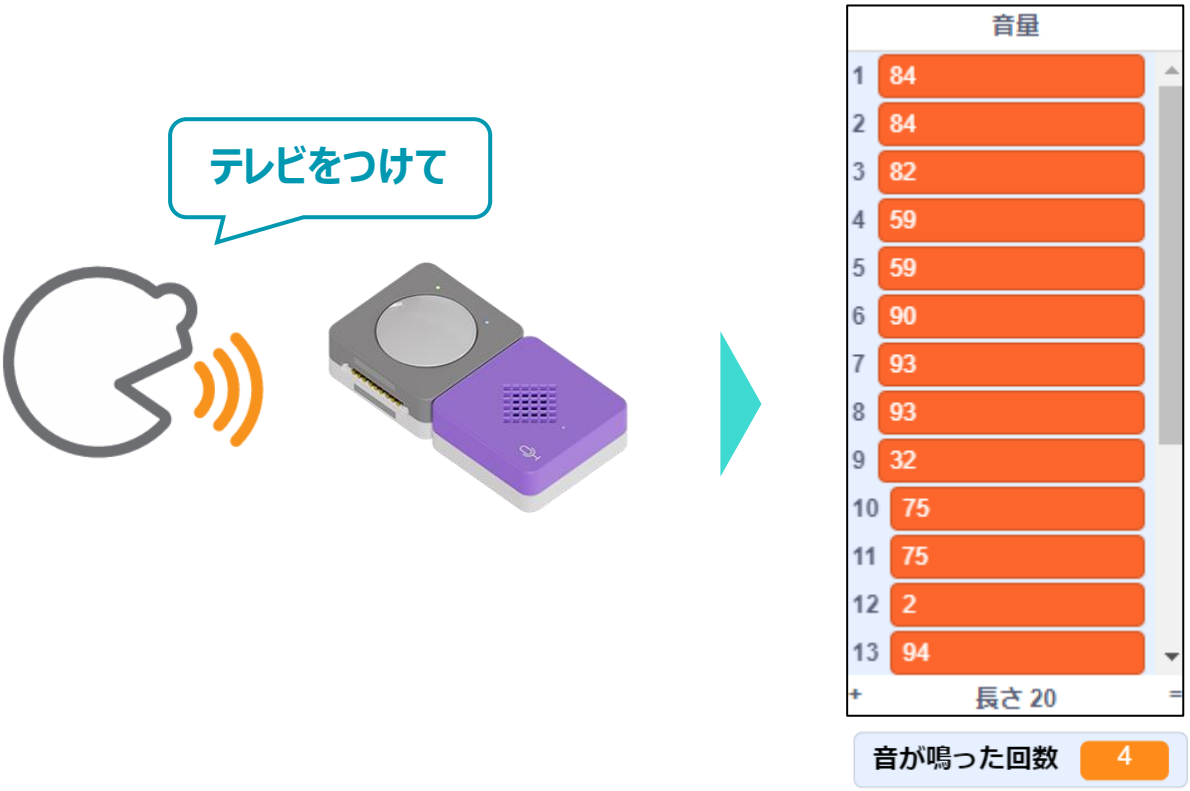


音が波のようになっていて、音量の大小は波の大きさで表されています。

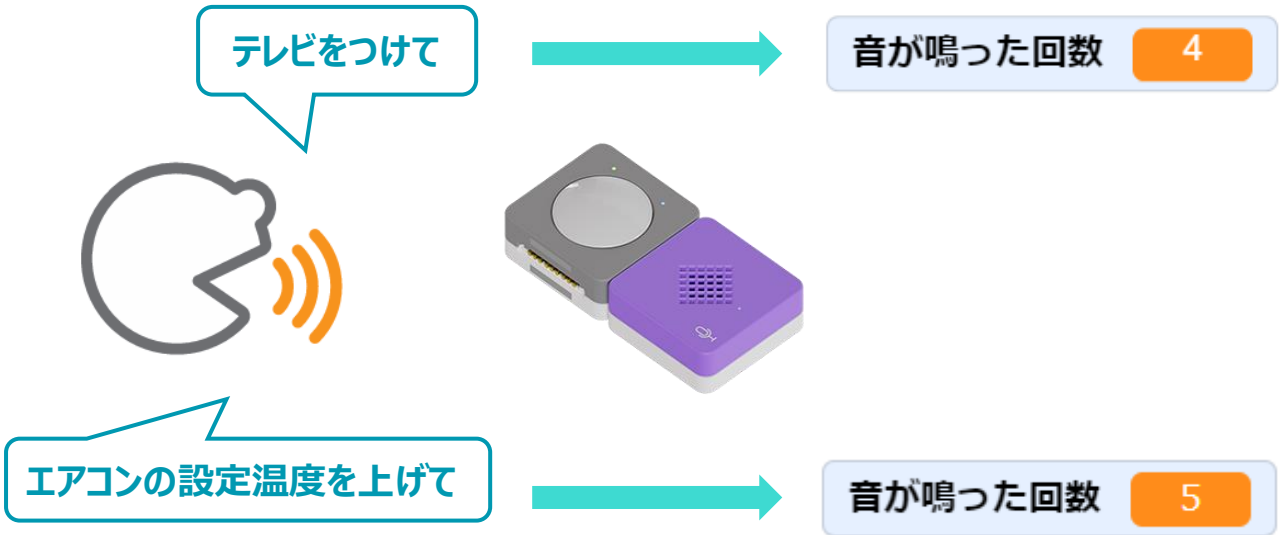
この波は音量の大小によって生まれているため、3-2（p.52）で作成した音の鳴った回数を数えるプログラムを実行すると、マイクで測定した音声が何個の波を持っているか、計測することができます。



3-2（p.52）で作成したスクリプトを実行して、マイクに向かって「テレビをつけて」と話しかけてみましょう。
リスト「音量」に「テレビをつけて」と言った音量データが追加され、音の波の数が計測されて、変数「音が鳴った回数」に記録されます。



この仕組みを利用することで、音声の長さや発音が大きく異なっている音声を区別して、音声の違いによって異なる家電を操作することができます。



音声認識の仕組み

今回作成したプログラムでは、音量をもとに音声を区別して認識していました。
実際に私たちの身の回りで使用されている音声認識では、音量の他に、音の高低（周波数）、音色（音波の形状）など複数の要素を分析することで、より正確な音声データを取得しています。

取得した音声データは、事前に分析された大量の音素（音の最小単位）のデータと比較されます。
音声データがどの音素に似ているか判断されて、音素の並びに変換されます。

並べられた音素は、事前に分析された大量の文章データをもとに単語・文章へと変換され、音声データが文章として認識できるようになります。

近年では AI（人工知能）の発達により、精度の高い音声認識が様々なサービスに利用されています。

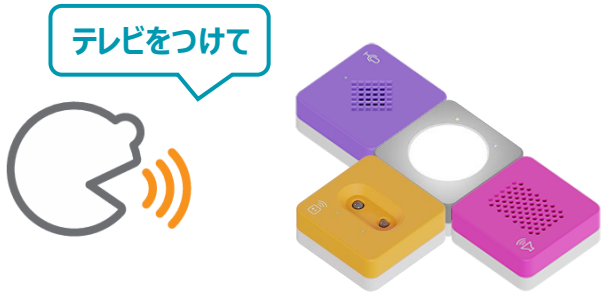
▲音声認識を用いた AI アシスタント

発展編：プログラムの実行中に 2 つのモードを切り替えよう

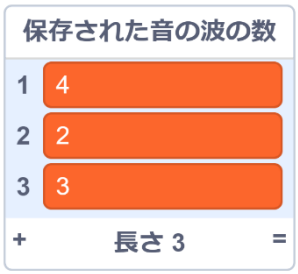
3-2（p.62）で作成した『入力された音声によって異なる家電を操作するプログラム』を改良します。
プログラムの実行中に 2 つのモードをボタンで切り替えられるように改良します。
◇録音モード：音声認識して家電を動かすための音声を登録するモード
◇音声認識モード：入力された音声の波の数を判別して家電を操作するモード

録音モード

音声認識するための基準となる音声を保存します
（今回作成するプログラムでは最大 3 つまで保存できます）

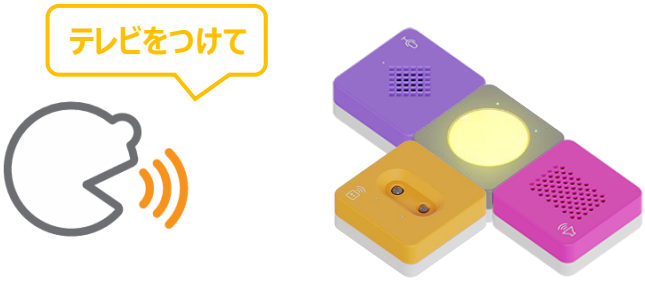


音声データをリストに保存



音声認識モード

入力した音声が録音モードで保存した音声データと
一致した場合に家電を操作します



赤外線を送信して家電を操作

サンプルプログラム　紹介ページはこちら

ここで紹介しているプログラムは、ホームページで公開しています。

以下の URL、またはボタンをクリックしてアクセスしてください。

https://www.artec-kk.co.jp/arteclinks/sample/98060/

サンプルプログラムは
ここをクリック

録音モードの仕組み

録音モードはボタンを押して操作します。

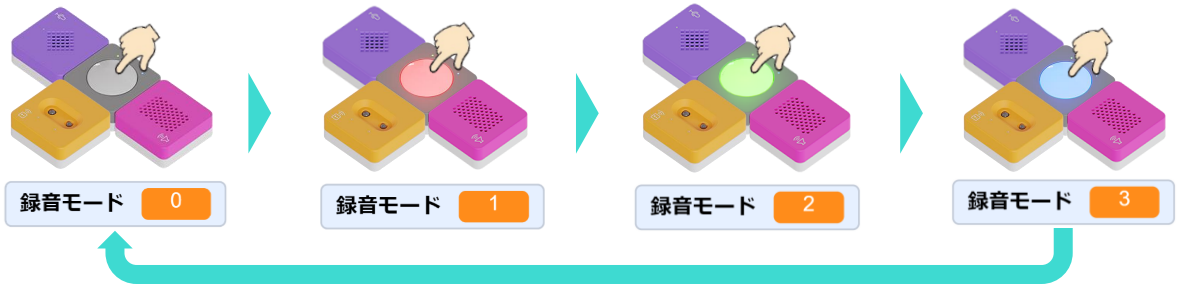
録音モードのプログラムには 2 つの機能があります。

(1)ボタンを押すと録音モードを 0→1→2→3→0…の順番で変化

(2)録音モードが 1~3 のときにボタンを長押しすると、録音開始

(1)録音モードを切り替える

ボタンを押すと、録音モードが 0→1→2→3→0…の順番で変化



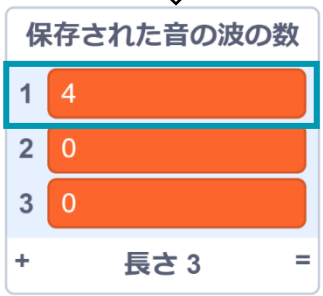
(2)録音する

録音モードが 1~3 のときに
ボタンを長押しすると録音開始

録音モード 1

録音モード 1~3 で計測した
「音の波の数」をリストの 1~3 番目に保存

録音モード 1 で録音したら
リストの 1 番目に保存



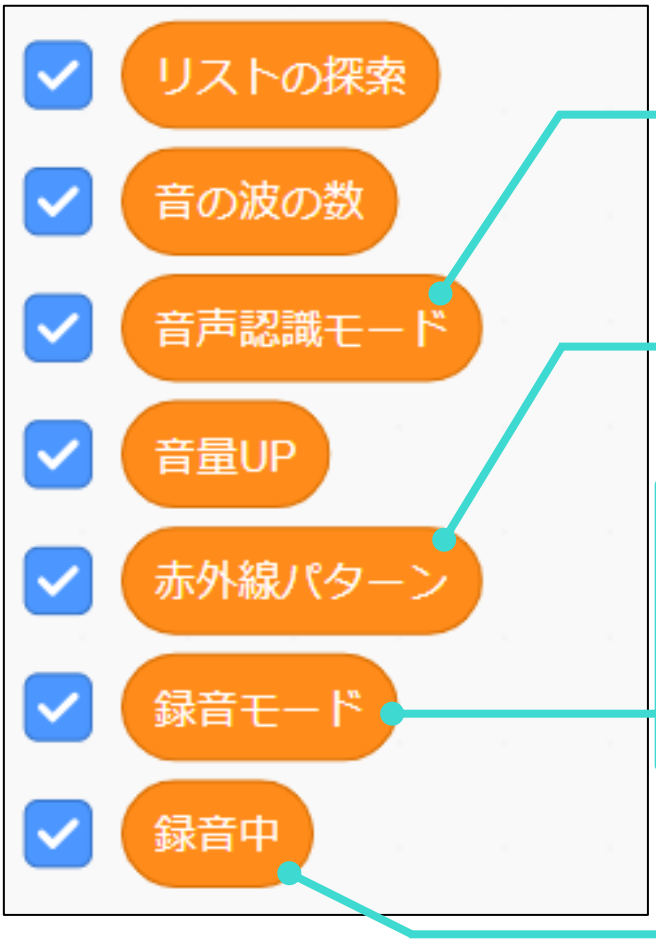
録音した音声は「音の波の数」が計測されて、リストに保存されます。

録音モードのプログラムでは、音声認識の基準となる音を登録（保存）することができます。

（音声認識の基準となる音は、何回でも録音し直すことができます。）

3-2（p.62）のスクリプトをもとに、いくつかの変数とリストを追加して、音声認識するプログラムを作成しましょう。

①変数「音声認識モード」、「赤外線パターン」、「録音モード」、「録音中」を作成します。



音声認識が可能な状態かどうか判断します。
認識できる状態　→ 1
認識できない状態 → 0

保存された数値によって、送信する赤外線の種類を変更します。

録音モードを切り替えて、複数の音声を録音します。
例）
録音モード **1**　←　テレビをつけて
録音モード **2**　←　エアコンの設定温度を上げて
録音モード **3**　←　部屋の電気を消して

録音中かどうか判断します。
録音中　　→ 1
録音中ではない → 0

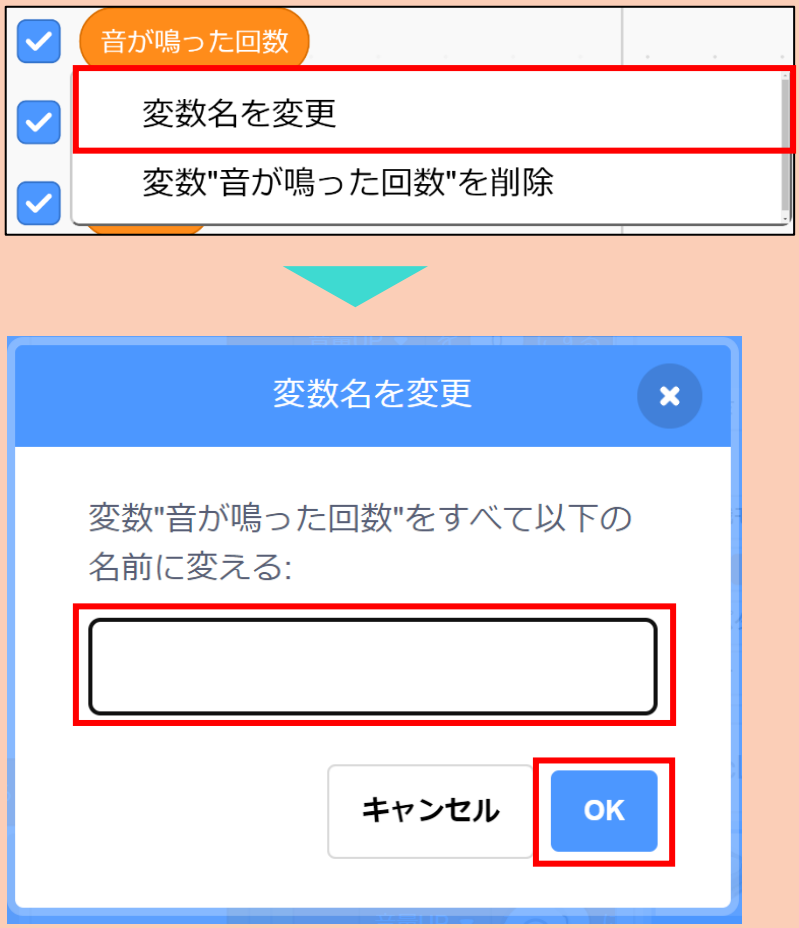
②リスト「保存された音の波の数」を作成します。

録音モード 1~3 で計測した音声の波の数をリストに保存します。
例）
録音モード **1** で録音した音声の波の数：**4**
→リストの **1** 番目に **4** を保存

③「音が鳴った回数」の名前を「音の波の数」に変更します。

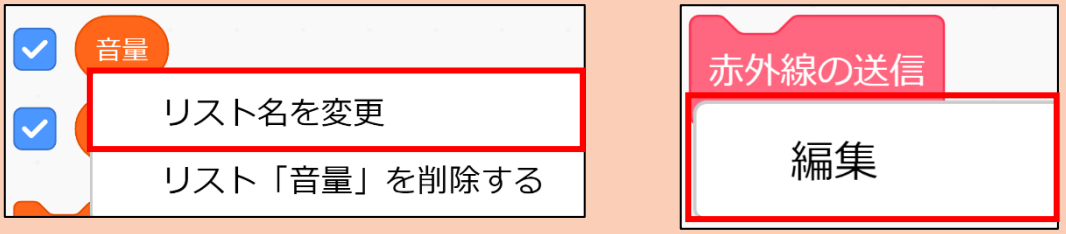
変数の名前の変更

作成した変数は、右クリックして「変数名を変更」を選択すると、名前を変更することができます。



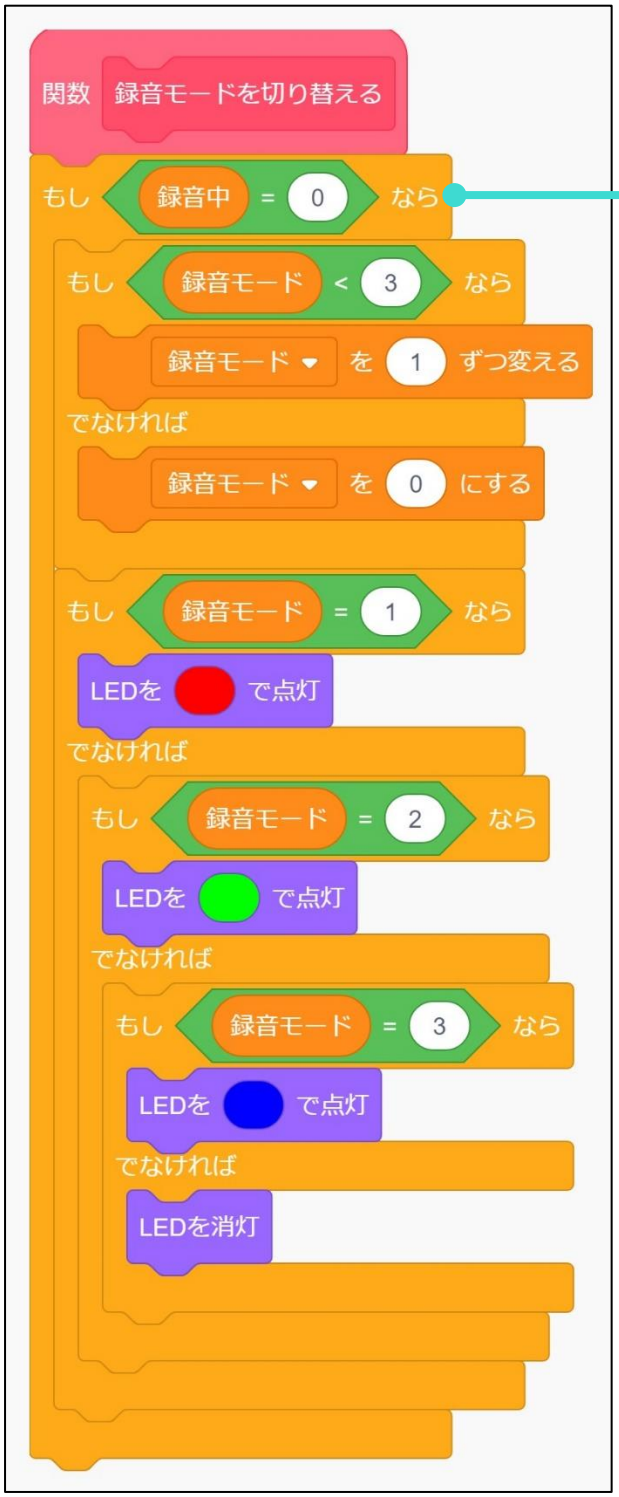
また、リストと関数についても名前を変更することができます。

リストは右クリックして「リスト名の変更」を、関数は右クリックして「編集」を選択して、名前を変更できます。



(1)録音モードを切り替えるプログラム

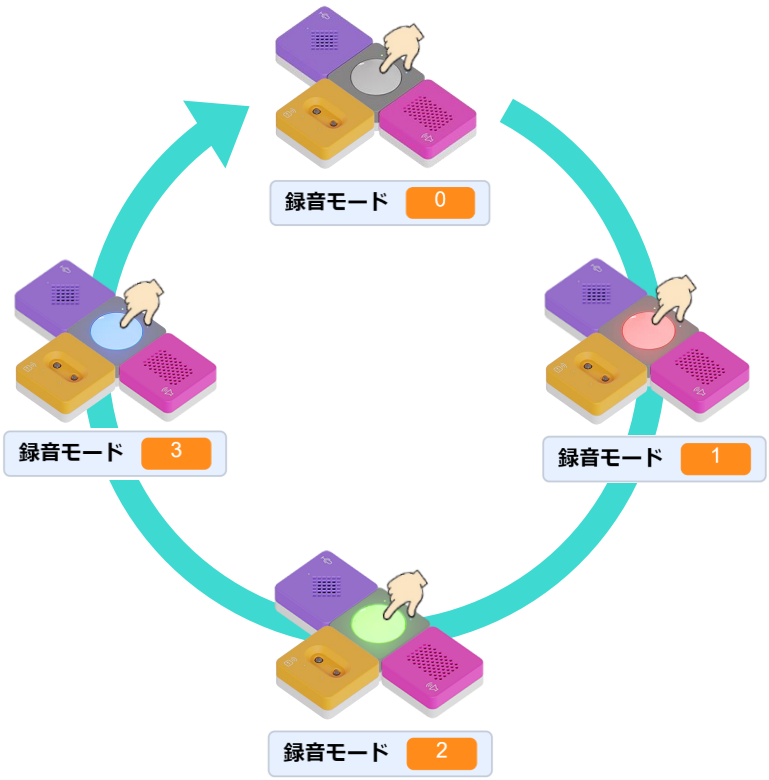
関数「録音モードを切り替える」を作成し、録音モードを切り替えるスクリプトを作成します。
変数「録音モード」が 3 未満のときは「録音モード」に 1 を足します。（0→1→2→3）
変数「録音モード」が 3 のときは 0 に戻します。（3→0）
また、「録音モード」の数値がわかるように、モードが 1 ならば赤、2 ならば緑、3 ならば青で点灯させます。



録音中ではない（「録音中」が 0）とき
録音モードが 3 より小さい場合は値を 1 ずつ足して、
録音モードが 3 のときは値を 0 に戻します。

関数 録音モードを切り替える **が実行されると…**

ボタンを押すたびに
録音モードが切り替わる

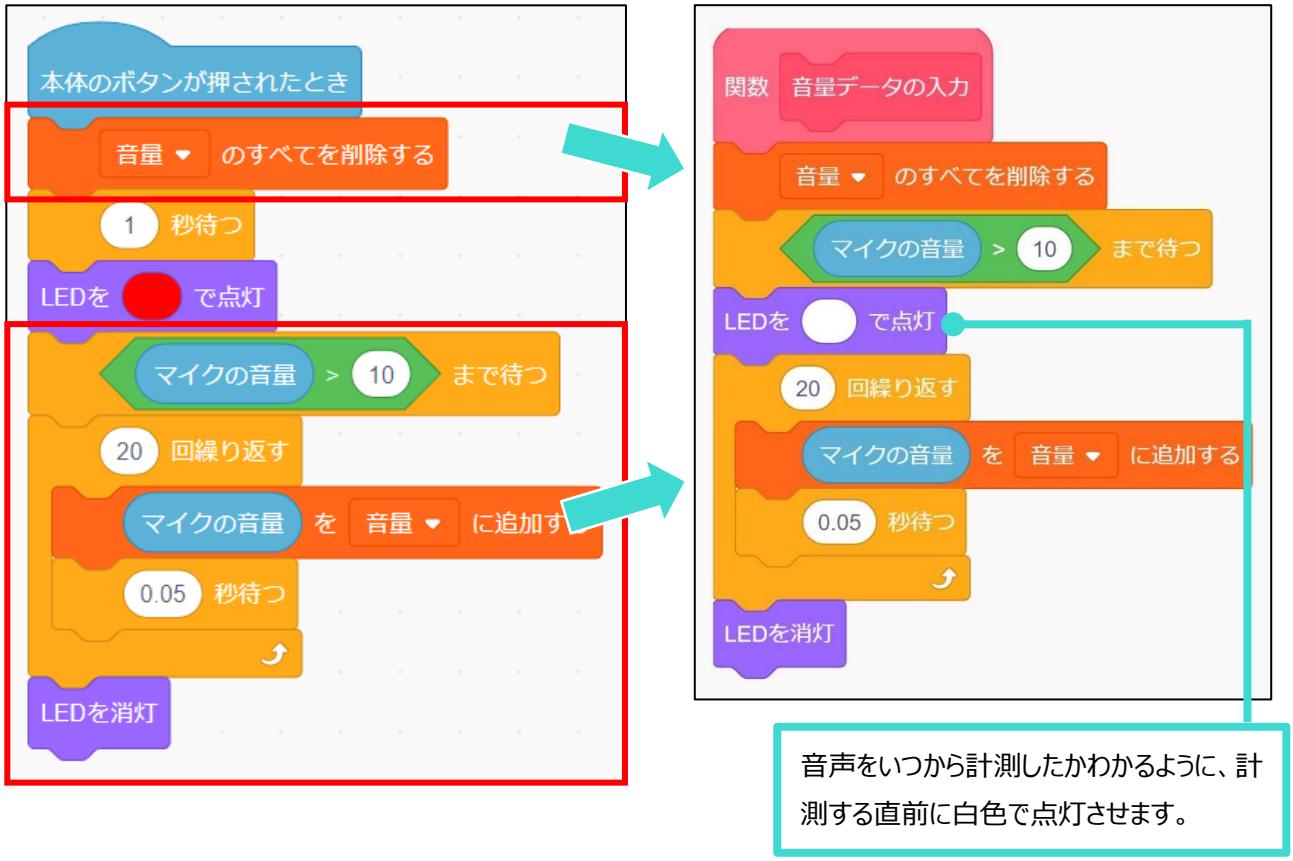


(2)録音をするプログラム

3-2（p.47）で作成したプログラムをもとに、録音をするスクリプトを作成します。

①関数「音量データの入力」を作成して、音量をリストに保存するスクリプトを作成します。

3-2（p.47）で作成した、マイクで計測した音量をリストに追加するスクリプトの一部を関数の下に繋げます。



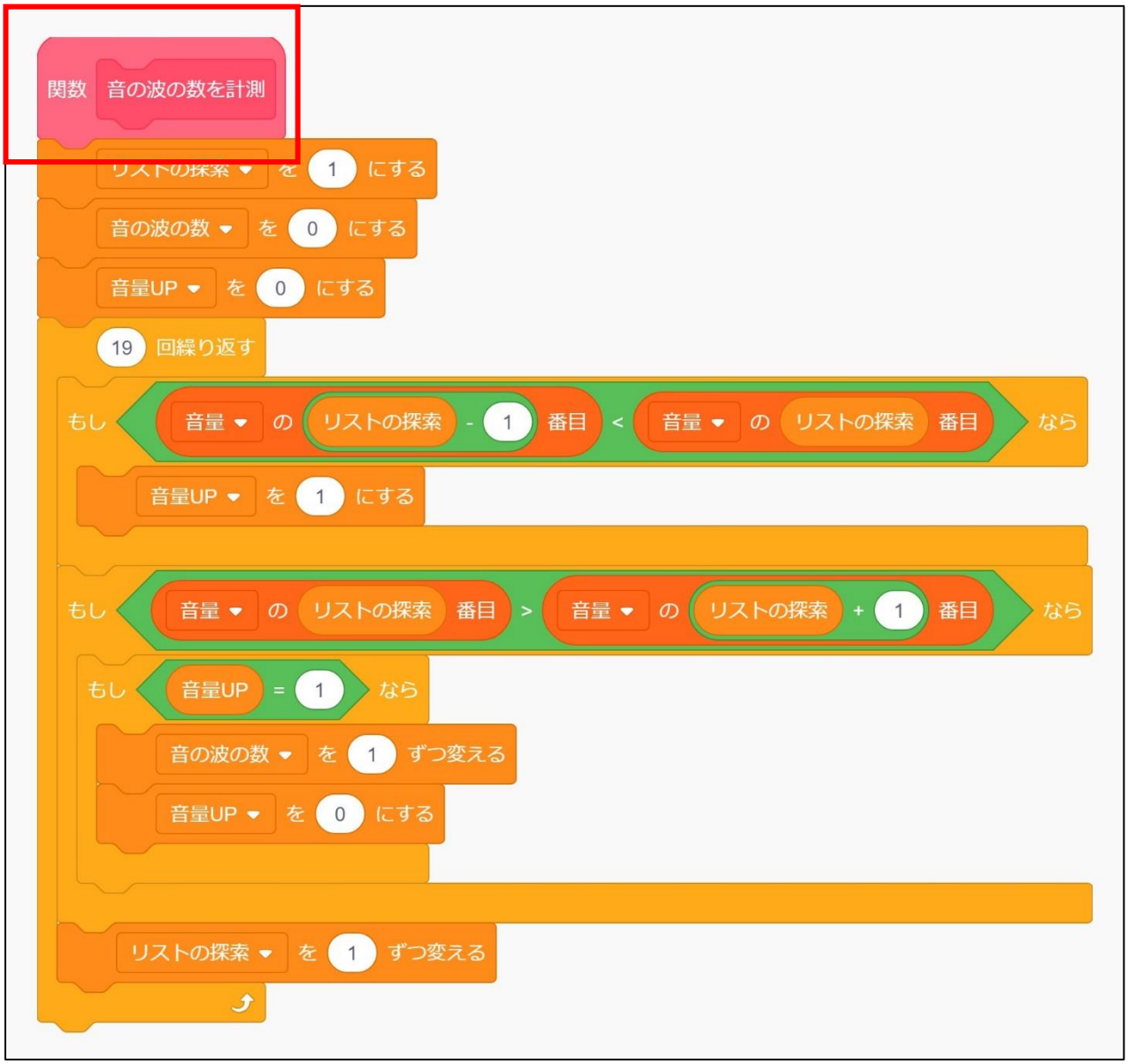
関数 音量データの入力 **が実行されると…**

マイクの音量が
リスト「音量」に追加

②入力した音の波の数を数えるプログラムを作成します。

3-2（p.59）で作成した関数「音が鳴った回数を数える」の名前を「音の波の数を計測」に変更します。

その他の部分は、そのまま使用します。

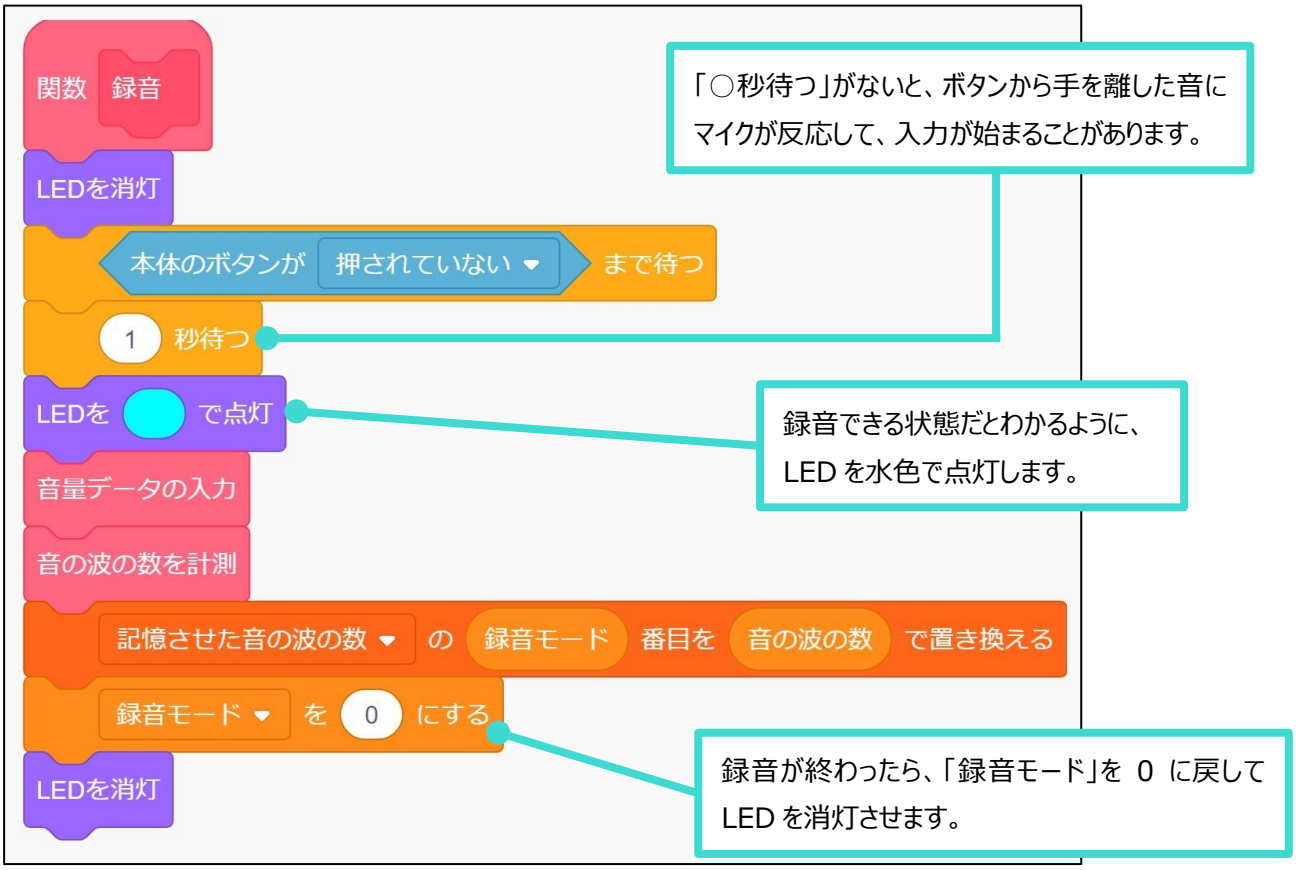


関数 音の波の数を計測 が実行されると…

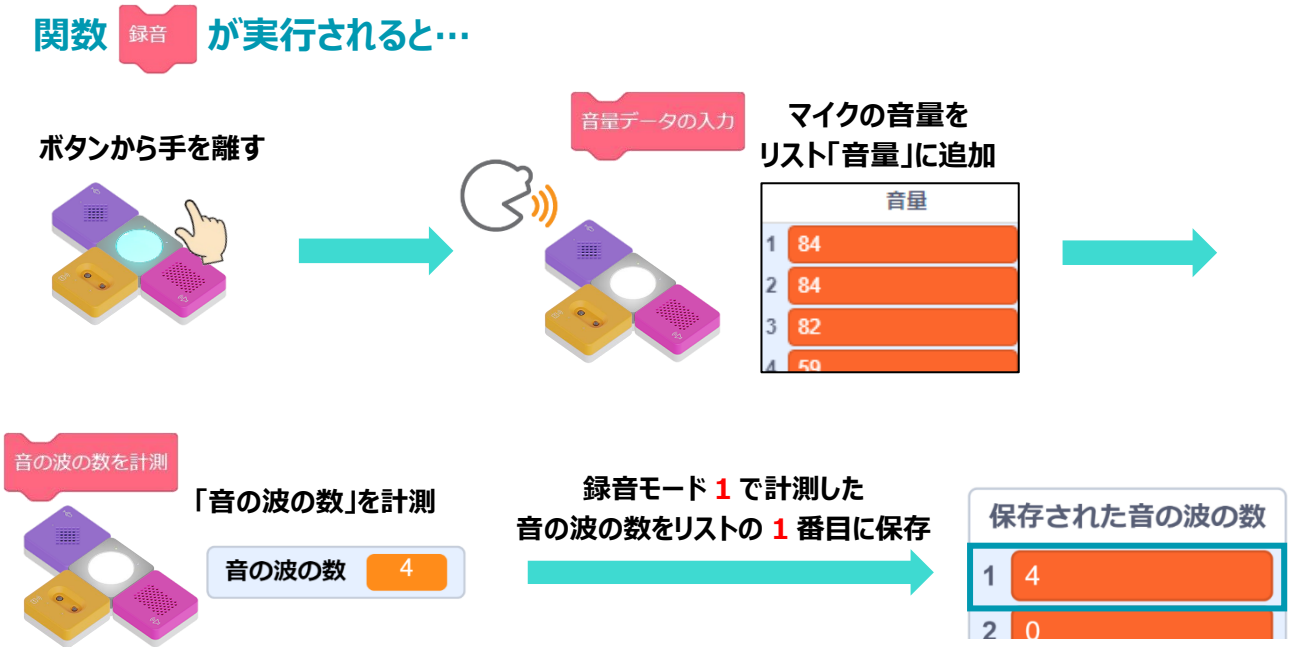
音量のデータを分析して
「音の波の数」を計測

音の波の数 4

③関数「録音」を作成し、音声を録音して波の数を計測するスクリプトを作成します。
ボタンから手を離した後に、①と②で作成した関数を実行します。



また、今回作成するプログラムでは複数の音を録音できるように、計測した「音の波の数」はリスト「保存された音の波の数」の「録音モード」番目に保存します。



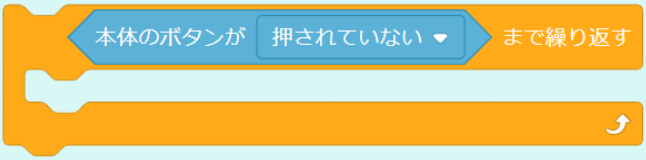
(1)と(2)のスクリプトを用いて、録音モードのプログラムを完成させよう

ボタンを押す長さによって、「(1)録音モードを切り替える」と「(2)録音する」を切り替えるスクリプトを作成します。
長押しされているときは「(2)録音する」を、短く押したときは「(1)録音モードの切り替え」を実行します。
ボタンが長押しされているかどうかは、タイマー を使って判定します。

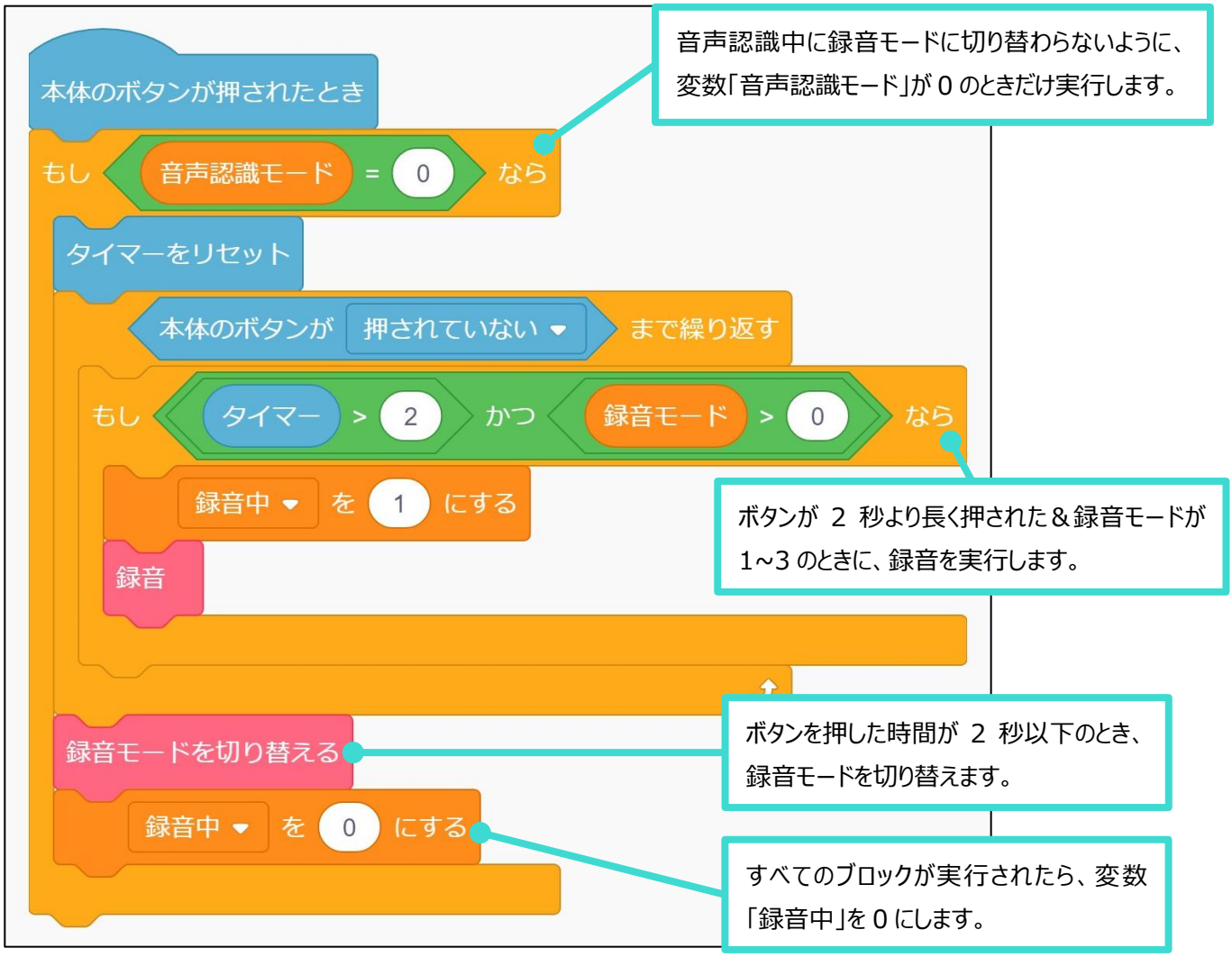
タイマーとは

タイマーとは、プログラム内で時間を計測することができる機能です。

カテゴリーの 調べる にある タイマー を使うと、
タイマーをリセット が実行されてからの時間を計測することができます。



と組み合わせると、
ボタンを長押ししたかどうかを判定できます。



音声認識モードの仕組み

録音モードのプログラムが完成したら、次は音声認識モードのプログラムを作成しましょう。
プログラムは、**(3)**音声認識の起動→**(4)**音声を入力して音声認識→**(5)**赤外線の送信、の順番で実行します。

(3)音声認識の起動

大きな音を検知したときに、音声認識を起動させる

(4)音声認識

音を入力して、録音モードで保存した音声データと比較する

保存した「音の波の数」と比較

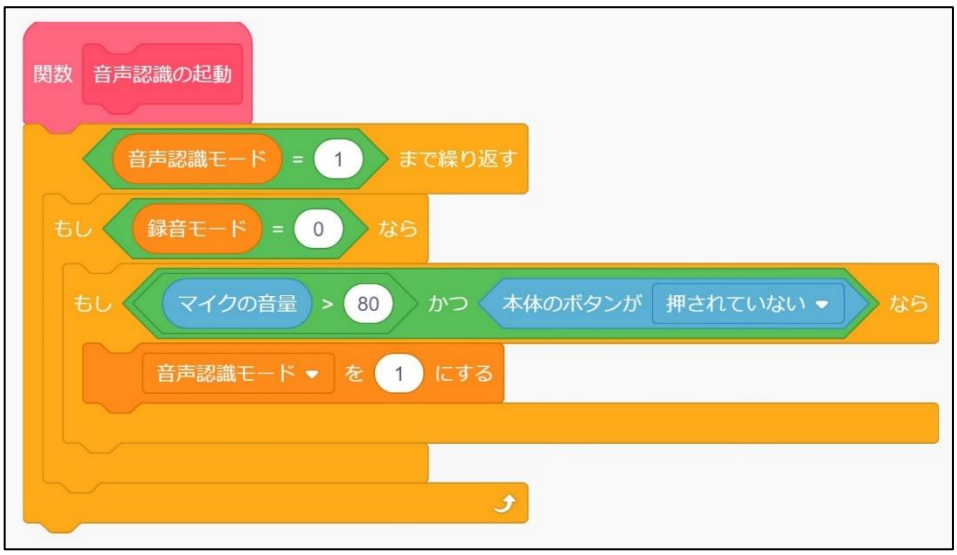
(5)赤外線の送信

入力した音声と保存された音声の「音の波の数」が一致した場合に赤外線を送信

(3)音声認識を起動させるスクリプト

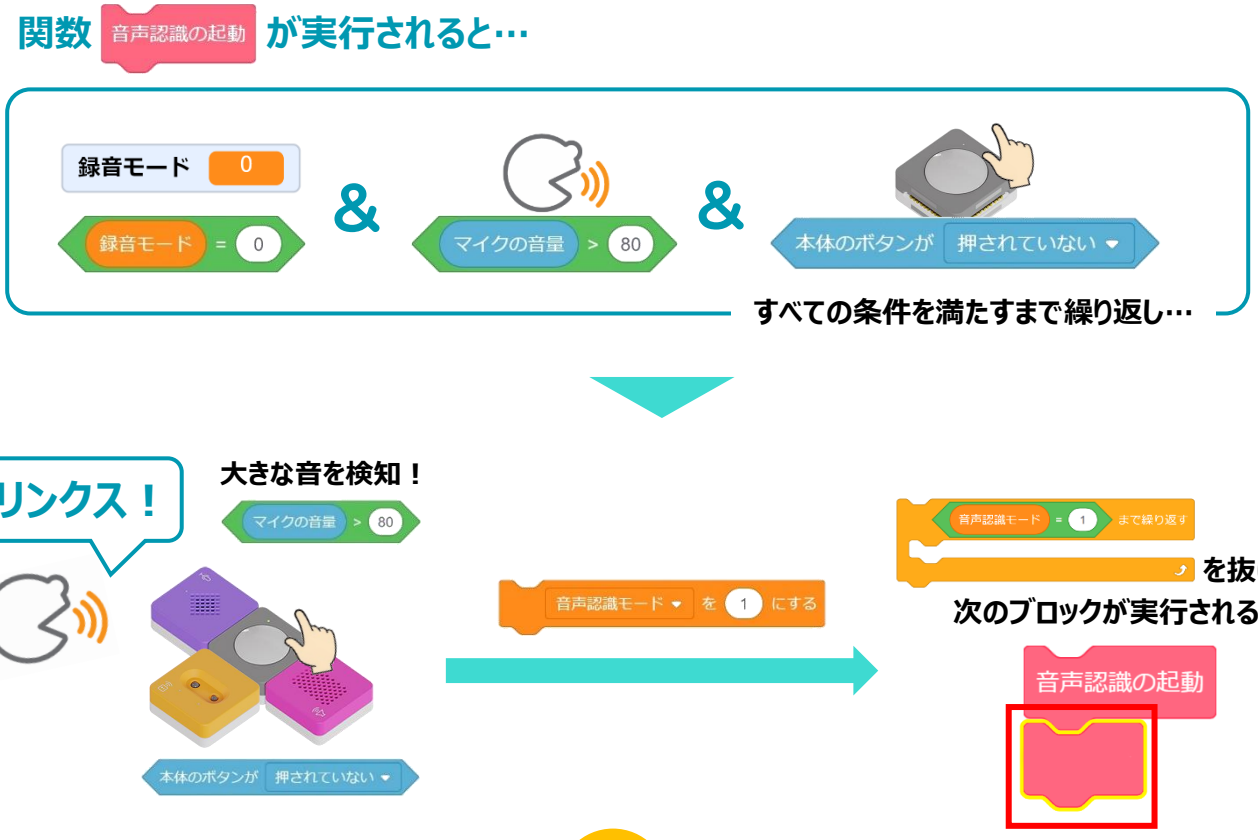
①関数「音声認識の起動」を作成します。

変数「録音モード」が 0 のときに、マイクが大きい音を検知し、かつボタンが押されていない場合に、変数「音声認識モード」を 1 にするスクリプトを作成します。



ではなく

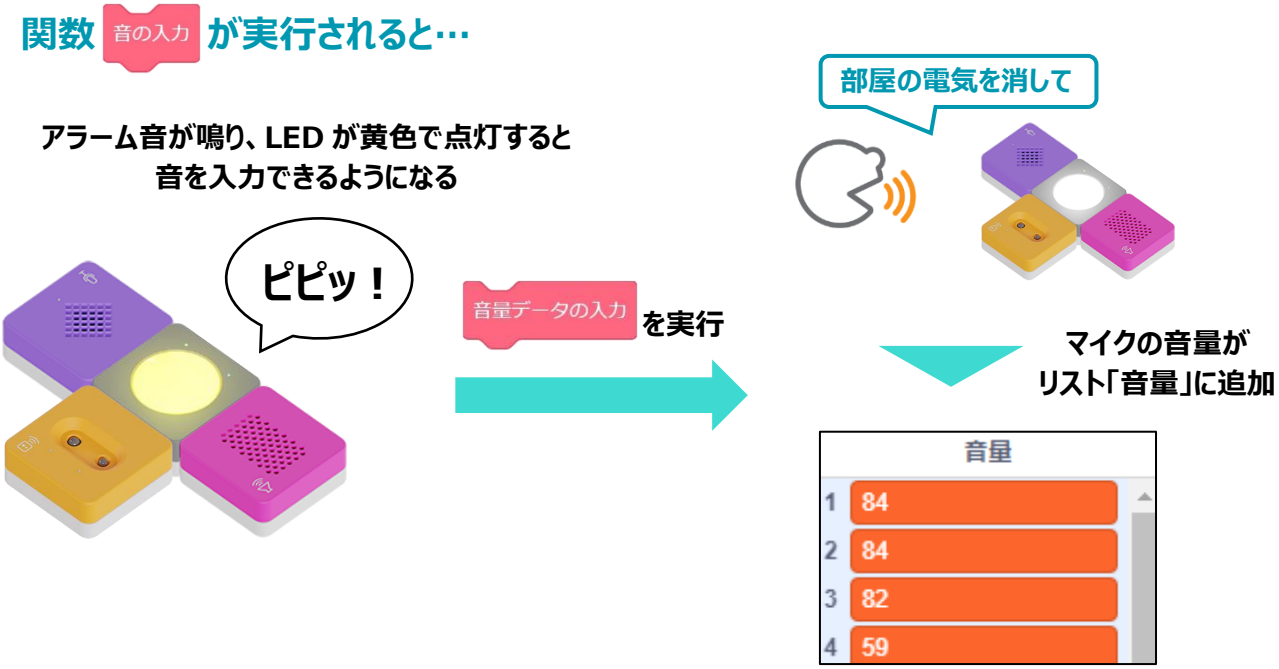
という条件が付いた繰り返しブロックを用いると、大きな音を検知して「音声認識モード」が 1 になった場合に、繰り返しから抜けられるようになっています。



(4)音を入力して音声認識をするスクリプト

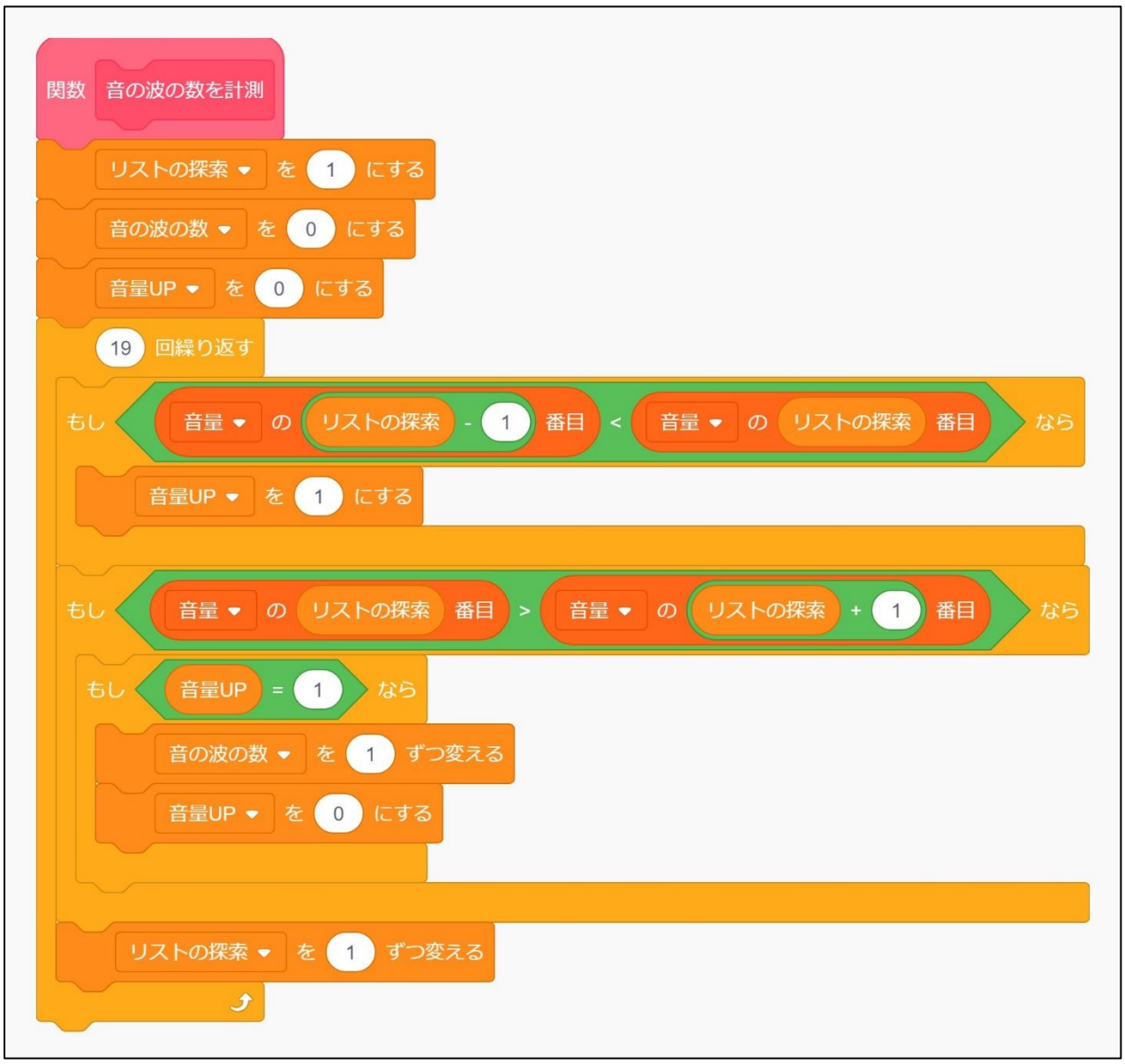
大きな音を検知した後は、入力された音声の「音の波の数」を計測します。
そして、入力した音声データと保存した音声データの「音の波の数」を比較して、どの音声と一致したかを調べます。

①音を入力するためのスクリプトを作成します。
関数「音の入力」を作成して、(2)（p.71）で作成した定義「音量データの入力」を繋げます。
また、音を入力できるようになったことがわかるように、スピーカーからアラームを再生して、LED を黄色で点灯させます。



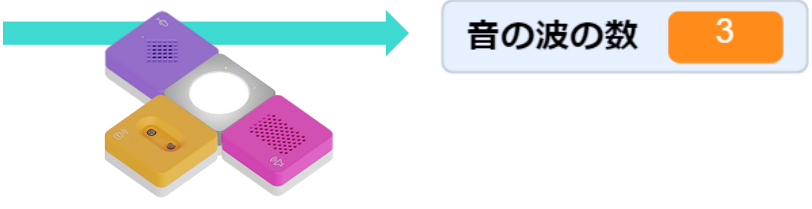
②入力した音量データを用いて、音の波の数を計測します。

入力した音の波の数を数える時は、(2)（p.72）で作成した関数「音の波の数の計測」をそのまま使用します。

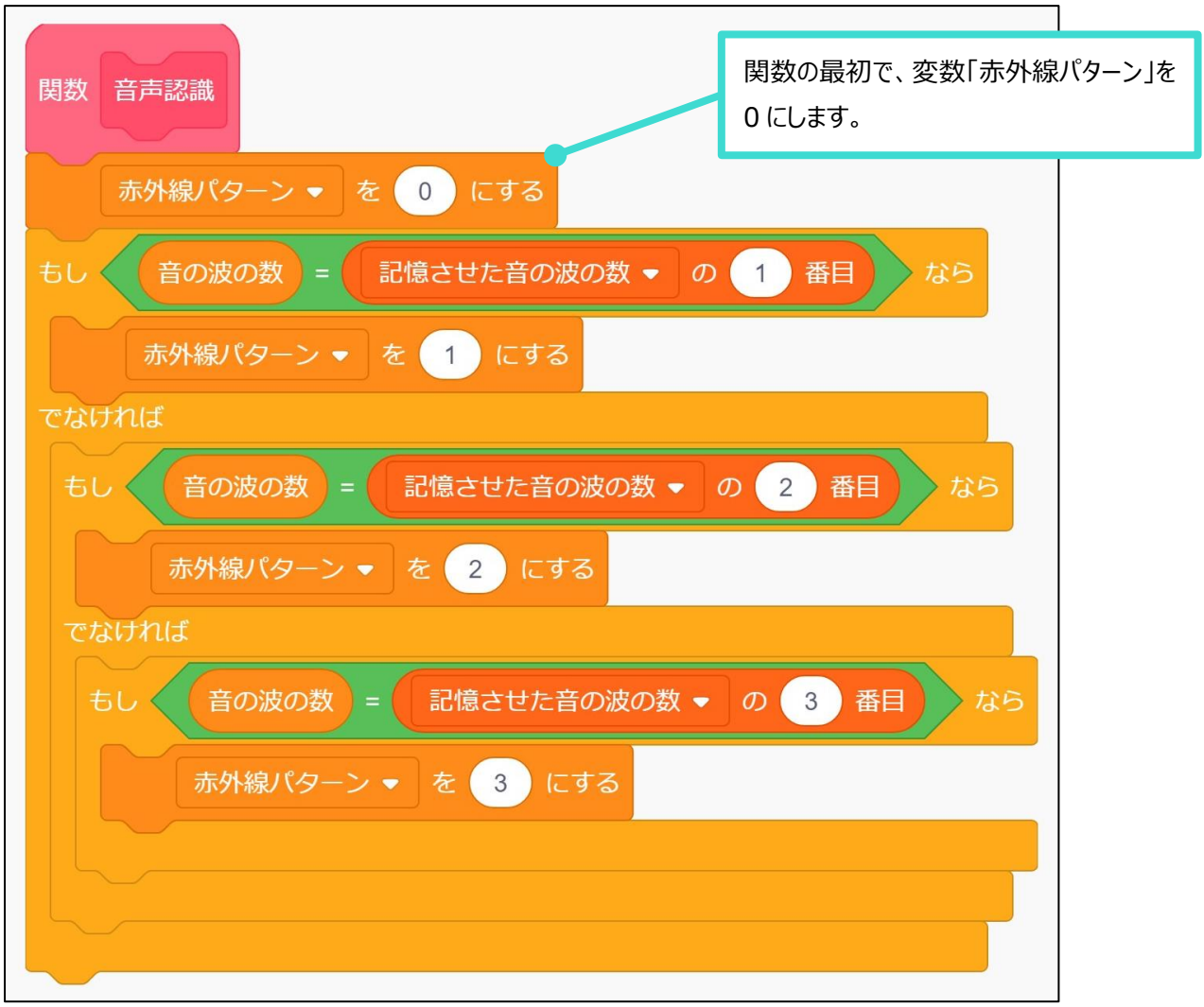


関数 音の波の数を計測 **が実行されると…**

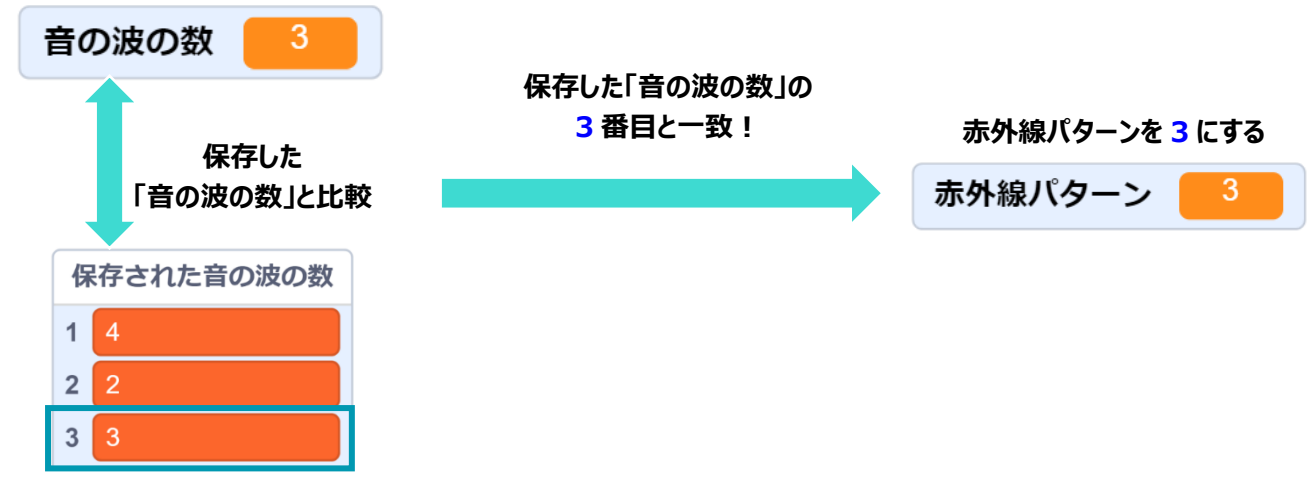
音量のデータを分析して
「音の波の数」を計測



③関数「音声認識」を作成して、音声認識をするスクリプトを実行させます。
変数「音の波の数」に保存された数値を、リスト「保存した音の波の数」の番号と順番に比較します。
「音の波の数」が一致したリストの番号を変数「赤外線パターン」に入力します。



関数 音声認識 が実行されると…



(5)赤外線を送信して家電を操作するスクリプト

3-2（p.62）で作成した関数「赤外線を送信する」を修正して作成します。
変数「赤外線パターン」の値によって異なる赤外線を送信するスクリプトに変更しましょう。

「音の波の数」が一致していない（赤外線パターン=0）ときは、赤外線信号を送信しません。

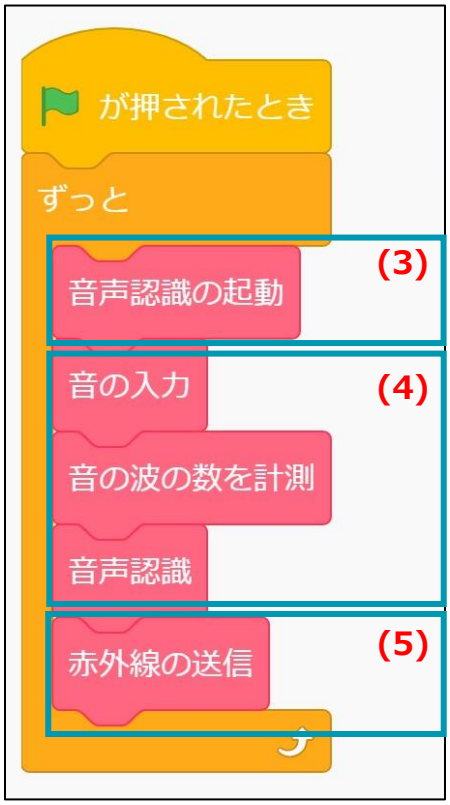
関数 赤外線を送信する が実行されると…

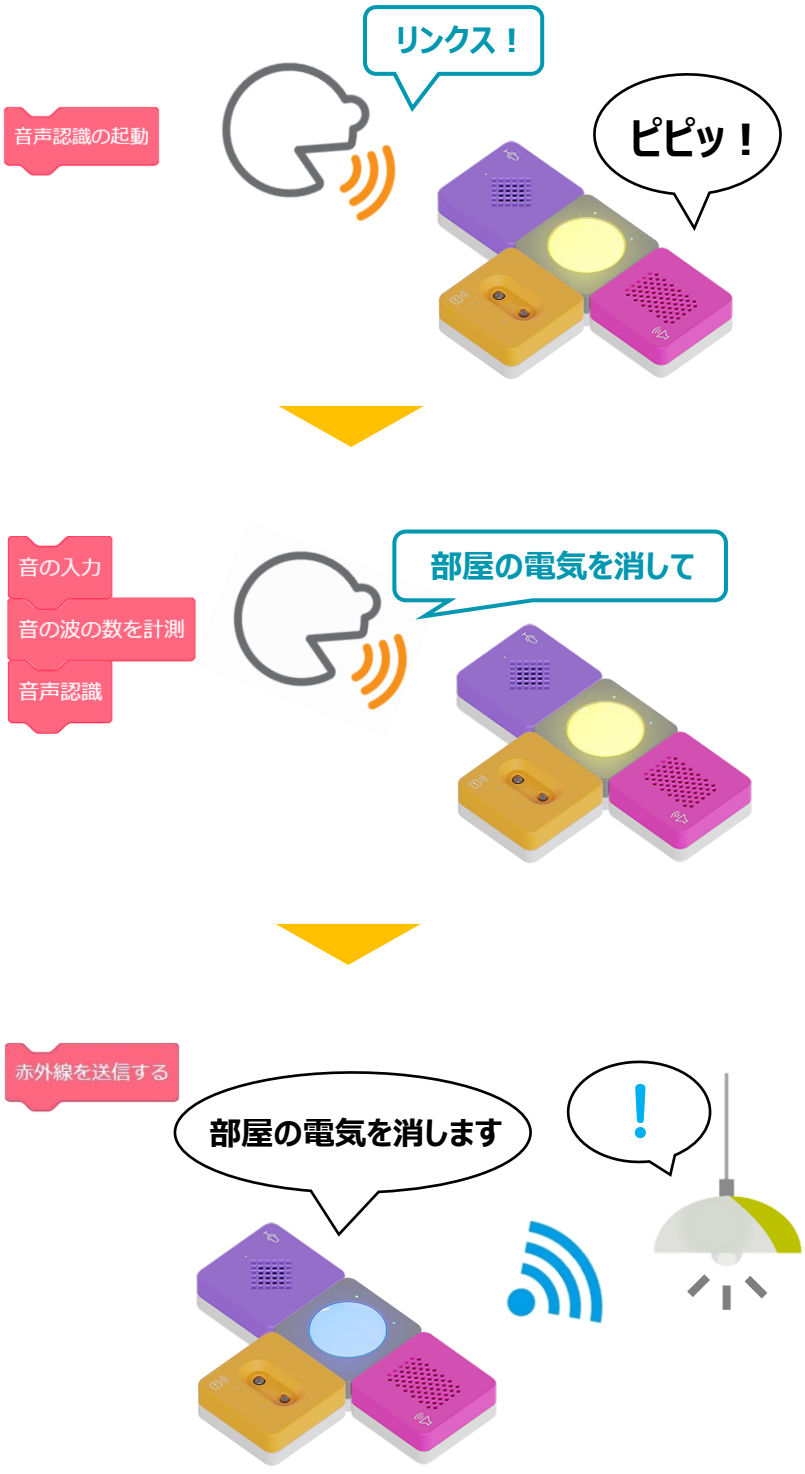
関数の最後で「音声認識モード」を 0 にリセットすると(3)に戻り、再び音声認識を開始することができます。

(3)～(5)のスクリプトを並べて、音声認識モードのプログラムを完成させよう

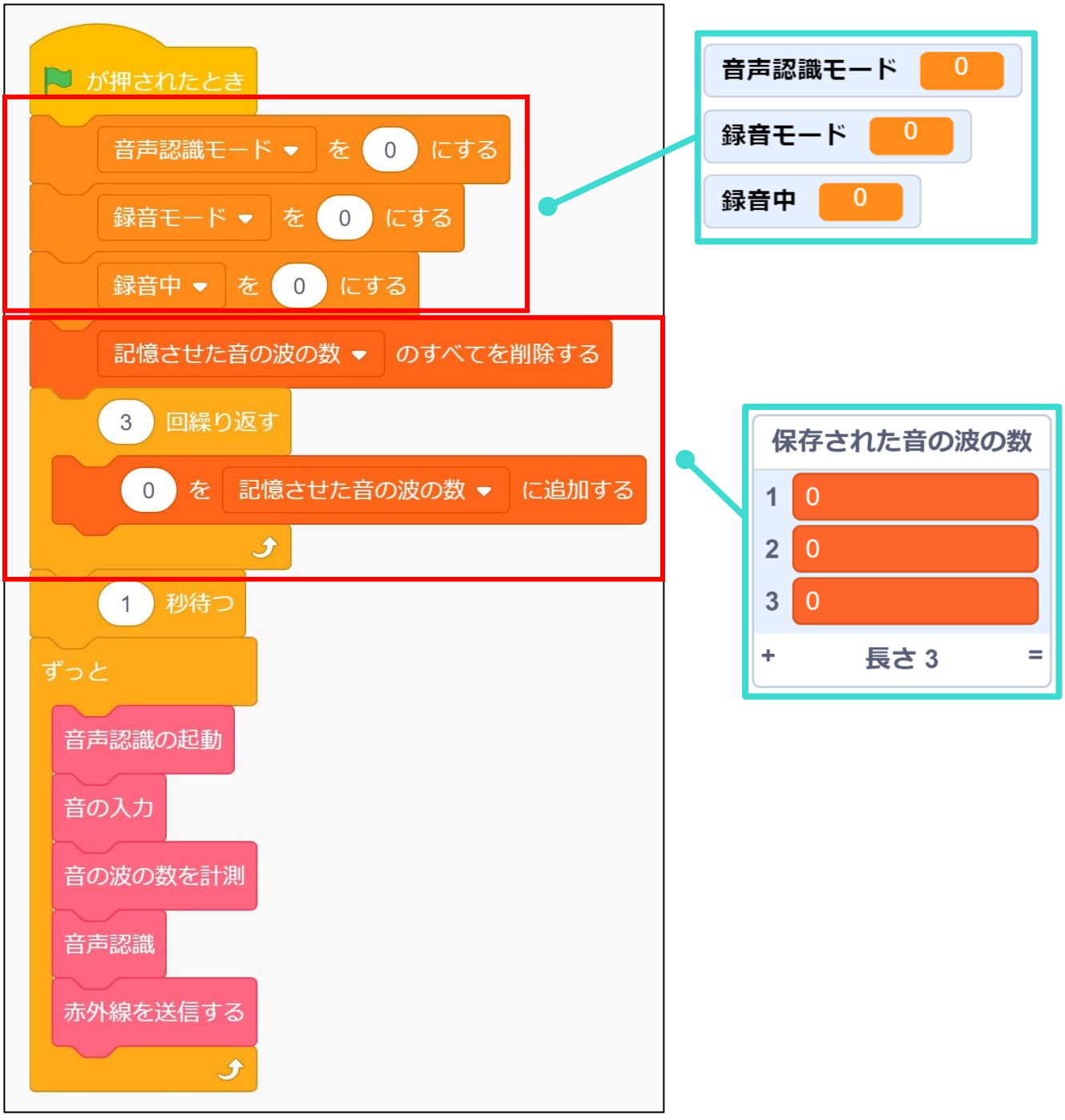
(3)～(5)で作成したスクリプトを順番に並べて、音声認識モードを実行させるスクリプトを完成させましょう。

①定義「音声認識の起動」、「音の入力」、「音の波の数を計測」、「音声認識」、「赤外線を送信する」を順番に繋げて、繰り返し実行させます。



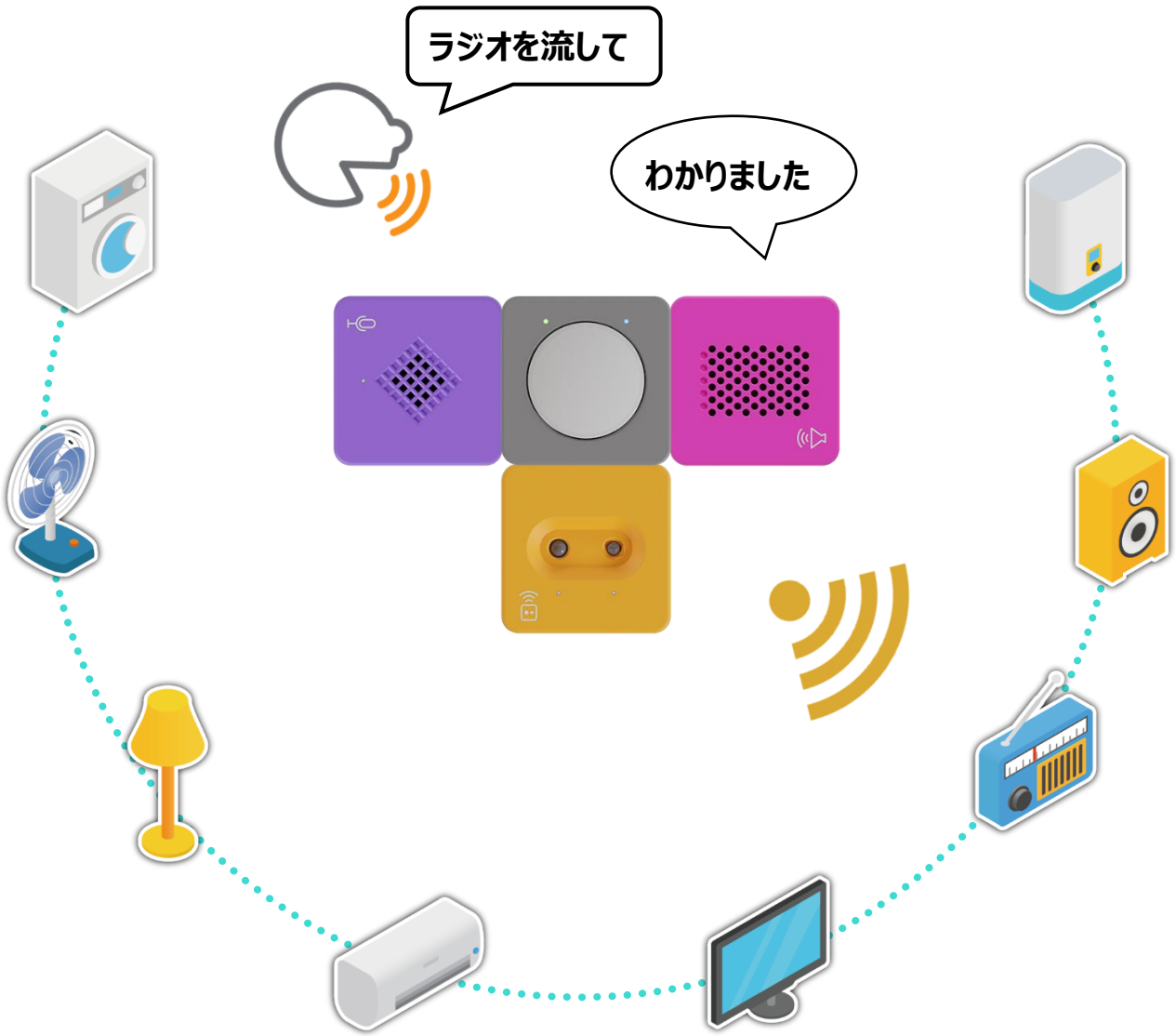


②プログラムが実行されたとき、最初に変数とリストに保存されている数値をリセットします。
変数「音声認識モード」、「録音モード」、「録音中」を 0 にします。
また、録音した音声の「音の波の数」を 3 つ保存できるように、0 が 3 つ保存されたリストを用意します。



これで音声認識モードのプログラムは完成です！
プログラムを実行して音声を入力し、家電を操作してみましょう。

例では 3 つの赤外線（テレビ、エアコン、部屋の電気）を登録したプログラムを紹介しました。
送信できる赤外線の種類・数は自由に変更できるため、生活スタイルに合ったプログラムにカスタマイズして使いましょう。



また、発展編で作成したプログラムには、スクリプトを実行するタイミングの指定やボタンの長押しなど、様々なテクニックが組み込まれています。
作成したプログラムを参考にして、この後に紹介する作品を改良したり、オリジナルのプログラムを作成したりしてみましょう。

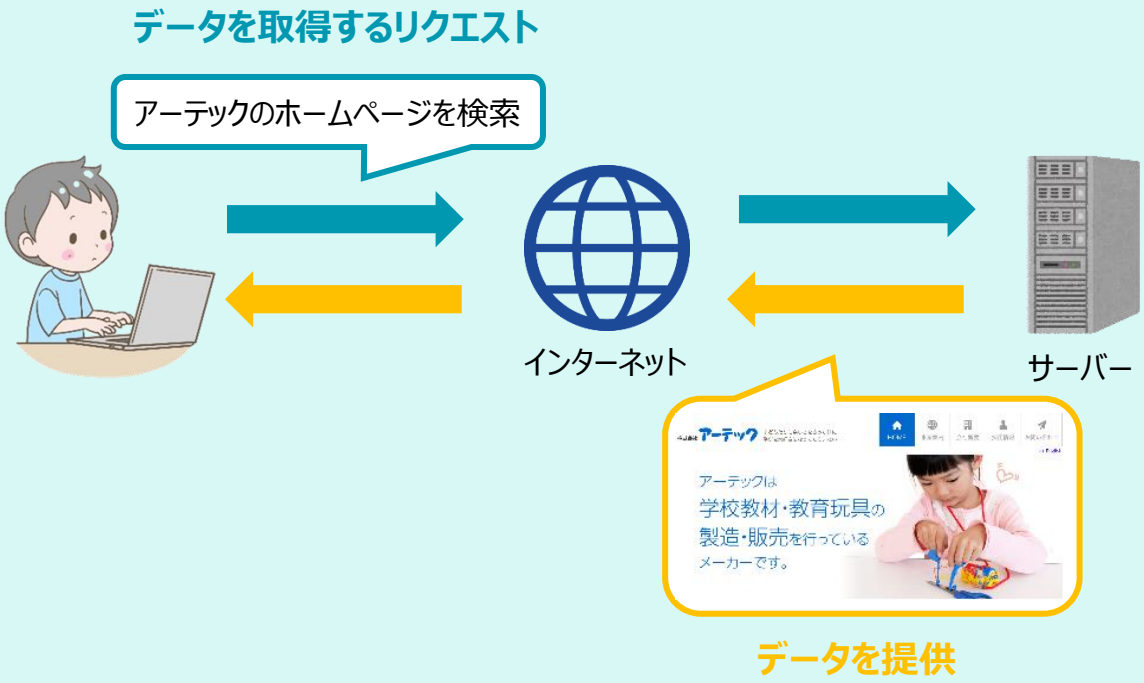
4 天気予報モニターと熱中症アラート

4 章では、メインユニットを Wi-Fi に接続して、インターネット上から取得したデータを利用するプログラムを紹介します。

メインユニットを Wi-Fi に接続すると、様々なデータを保存・提供しているサーバーにアクセスすることができ、指定したデータをサーバーから取得することができます。

サーバーとは

サーバーとは、インターネットを通じてデータやサービスを提供するコンピューターのことです。
例えば、私たちがインターネットで検索をすると、Web サイトのデータを保管しているサーバーから画像や文章のデータが提供されます。



▲サーバーから情報を取得する仕組み

サーバーには様々な種類があり、現在時刻のデータを提供するタイムサーバーや、気象情報のデータを提供する気象情報サーバーなど、リクエストを送ったサーバーによって異なるデータを取得できます。
サーバーから取得できる情報は信頼性と即時性が高いため、データを取得する先のサーバーを変更することで便利なプログラムを幅広く作成できます。

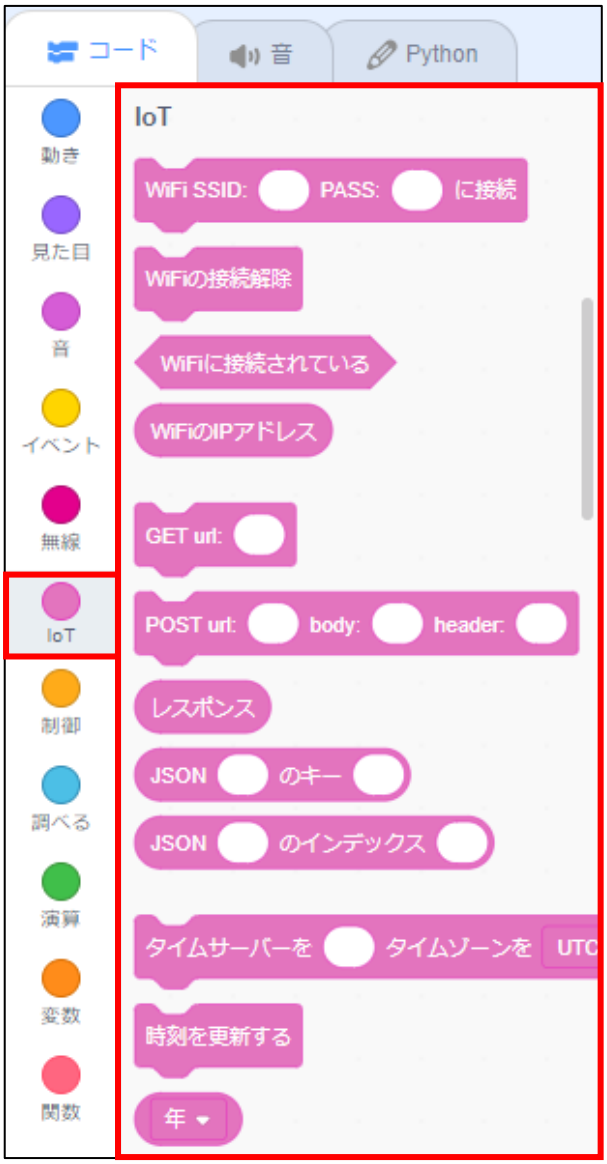
4-1 メインユニットを Wi-Fi でインターネットに接続しよう

メインユニットをインターネットに接続したり、サーバーからデータを取得したりする場合は、IoT ブロックを使います。

IoT ブロックの使い方

画面左上の「編集」から「ブロックの表示・非表示」を選択し、「IoT」をクリックします。

カテゴリーの 無線 と 制御 の間に、新しく IoT のカテゴリーとブロックが追加されます。

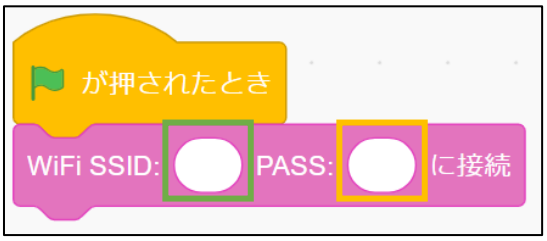


メインユニットを Wi-Fi に接続するときは、カテゴリーの 

にある 

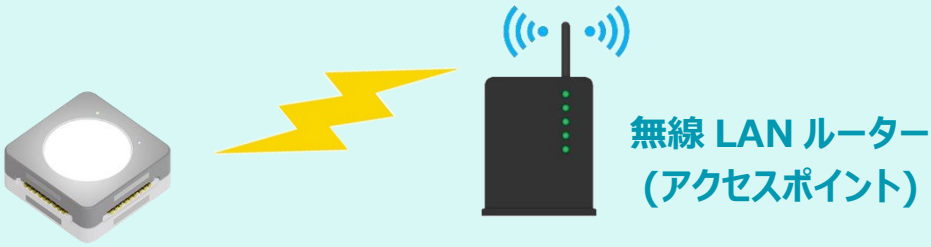
を実行します。

□ に SSID、□ にパスワードを入力して実行すると、指定した Wi-Fi に接続することができます。



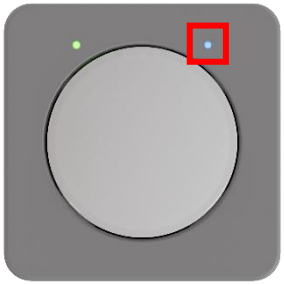
SSID とは

SSID とは接続するアクセスポイントを判別するための名前です。デバイスを Wi-Fi に接続するときは、無線 LAN ルーター等のアクセスポイントに接続する必要があります。それぞれのアクセスポイントを見分けるためにつけられた名前のことを SSID と呼びます。



アクセスポイントによっては、決まった人だけが接続できるように PASS（パスワード）の入力が求められることがあります。

メインユニットが Wi-Fi に接続されていると、メインユニットの LED(小)が青色で点灯します。



Wi-Fi への接続を解除したい場合は、 

を実行します。

LED(小)が青色で点灯しない場合は Wi-Fi に接続できていません。
SSID とパスワードの入力を間違えていないか、確認してからプログラムを再度実行してください。

4-2 サーバーから現在時刻を取得しよう

サーバーから現在時刻のデータを取得して、日付や現在時刻を表示してくれるプログラムをつくりましょう。

年	2024	月	1	日	1
時	8	分	42	秒	15

タイムサーバーから時刻を取得しよう

現在時刻を取得する場合は、時刻を提供するサーバー（タイムサーバー）から時刻情報を取得します。
タイムサーバーを設定するには、 

の空欄にサーバーの URL を入力して実行します。

①4-1（p.86）で作成したプログラムに 

を繋げて、現在時刻を取得します。
空欄にタイムサーバーの URL「**ntp2.plala.or.jp**」を入力しましょう。

② 

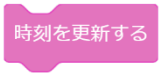
のタイムゾーンを「UTC+9」に変更します。

タイムゾーンとは

タイムゾーンとは、地球上で同じ標準時を採用している地域の集合のことです。
世界で共通の標準時として UTC（協定世界時）が用いられていますが、これはイギリスを基準としているため、経度の違いによって時差が生まれています。そこで、イギリスより東にある地域は UTC に時刻を足し、西にある地域は UTC から時刻を引くことで、各地域の標準時を表示しています。
日本では、UTC に 9 時間を足した時刻が標準時になります。同じタイムゾーンの地域には韓国やインドネシア東部などがあります。

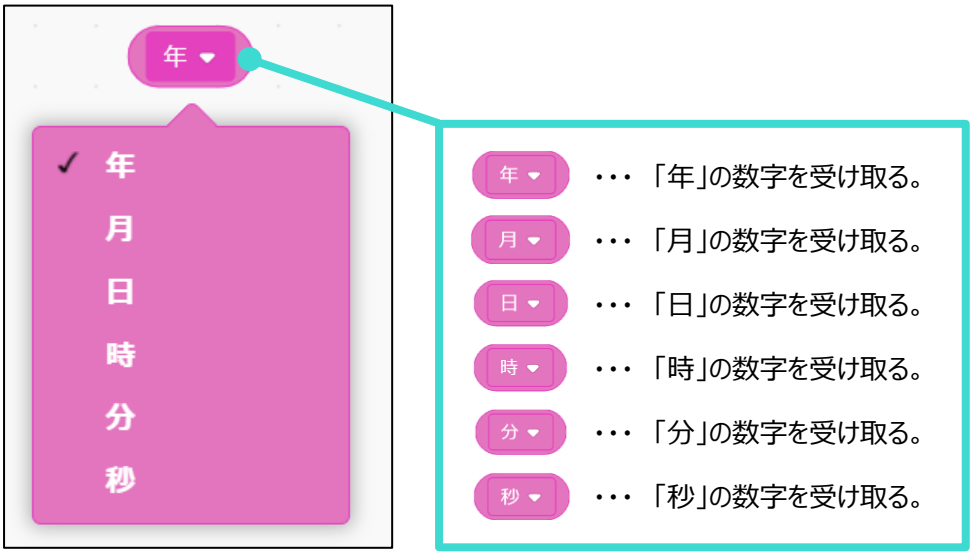
世界の地域	タイムゾーン
英国	UTC
フランス	UTC+1
インド	UTC+5:30
中国	UTC+8
シンガポール	UTC+8
韓国	UTC+9
日本	UTC+9
オーストラリア（キャンベラ）	UTC+10
リオデジャネイロ	UTC-3
アメリカ（ニューヨーク）	UTC-5
アメリカ（ロサンゼルス）	UTC-8

▲各地域のタイムゾーン

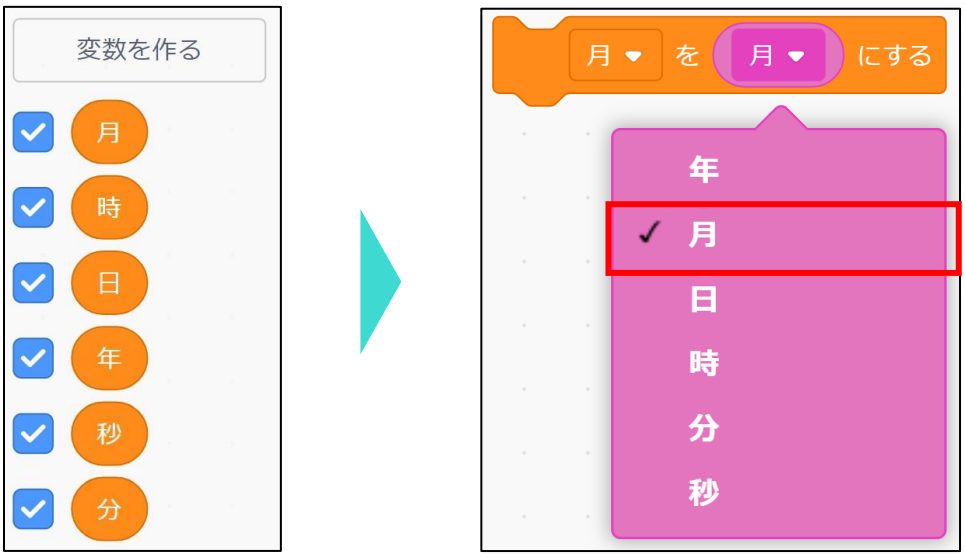
③タイムサーバーを設定して 

を実行すると、タイムサーバーから現在時刻を取得して 

にデータが保存されます。
保存されるデータは年・月・日・時・分・秒の 6 種類です。



取得したデータを表示するために、年・月・日・時・分・秒の 6 個の変数をつくり、それぞれに 年 を挿入します。



④メインユニットのボタンを押したときに時刻を更新して、作成した変数に現在時刻を保存するスクリプトを作成します。

スクリプトを実行してメインユニットのボタンを押すと、作成した変数に現在の時刻が保存されます。

現在時刻が保存されないときは、タイムサーバーの URL「**ntp2.plala.or.jp**」が正確に入力されているか確認してください。

プログラミングで目覚まし時計をつくろう

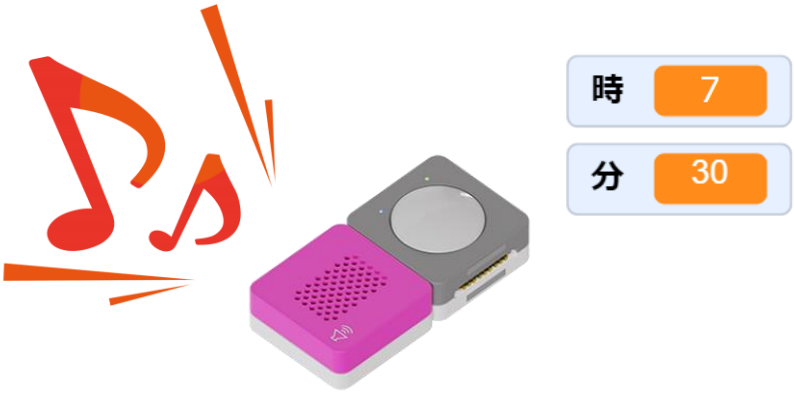
取得した現在時刻を活用し、スピーカーを接続して目覚まし時計をプログラミングしましょう。

①起床したい時刻を条件に設定して、設定した時刻（例では 7 時 30 分）になるまで現在時刻を更新し続けます。

かつ の左右に入れたブロックの条件を両方とも満たしているときだけ、繰り返しから抜けることができます。
どちらか一方だけの条件が合っていても、繰り返しから抜けることは出来ません。
例）
7 時 28 分 → ×
6 時 30 分 → ×
7 時 30 分 → ○ 繰り返しから抜けられる！

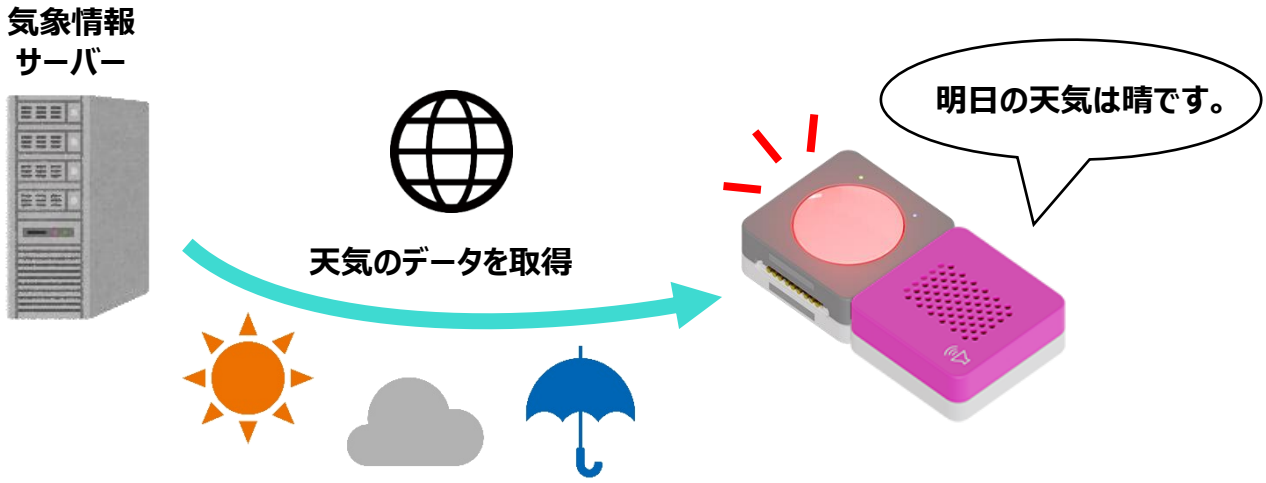
②①のスクリプトの繰り返しから抜けた後に、ボタンが押されているまでブザーを繰り返し鳴らします。
ボタンが押されたら、スピーカーを止めます。

このスクリプトを実行すると、指定した時刻になるまで現在時刻を取得し続けます。
そして、指定した時刻になるとボタンを押すまでブザーが繰り返し鳴り響きます。



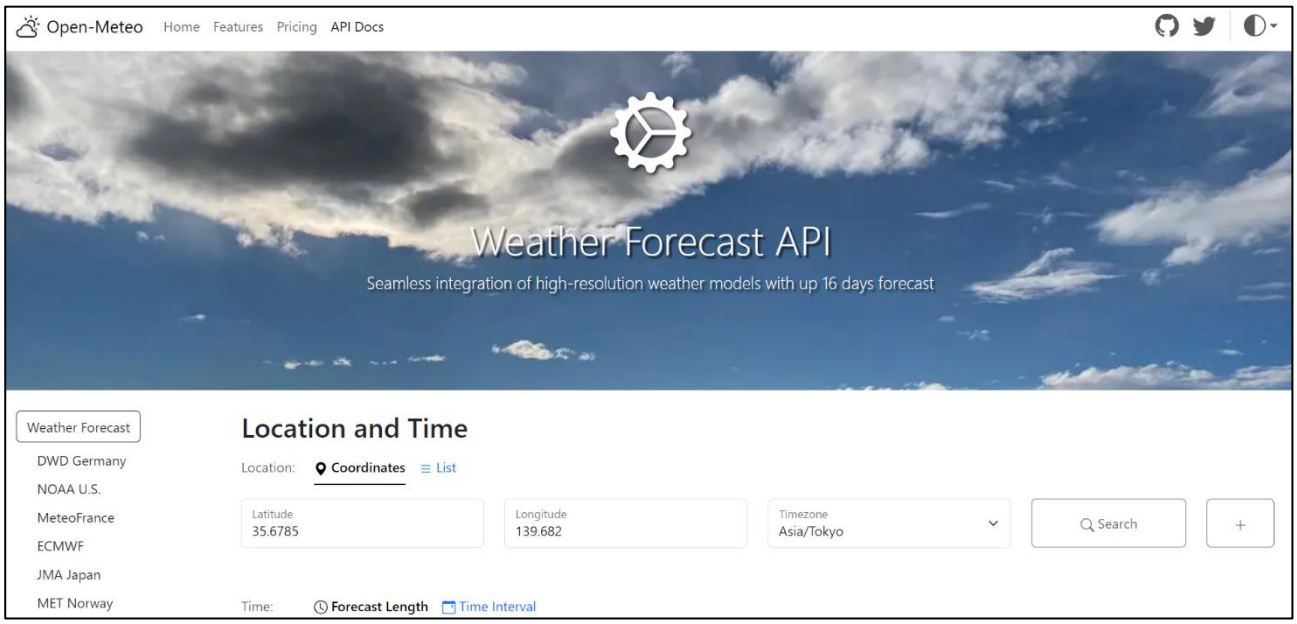
4-3 サーバーから気象情報を取得しよう

気象情報サーバーから今後の天気のデータを取得して、天気を予報するプログラムを作成しましょう。



今回作成するプログラムでは、次のサービス（Open-Meteo）を使用します。

https://open-meteo.com/en/docs



このサービスは、誰でも無料で気象情報を調べることができる WEB サービスです。

知りたい気象情報（気温や降水量、風速など）や地域を指定すると、自動で URL が作成されて、データを受け取ることができます。

気象情報を取得しよう

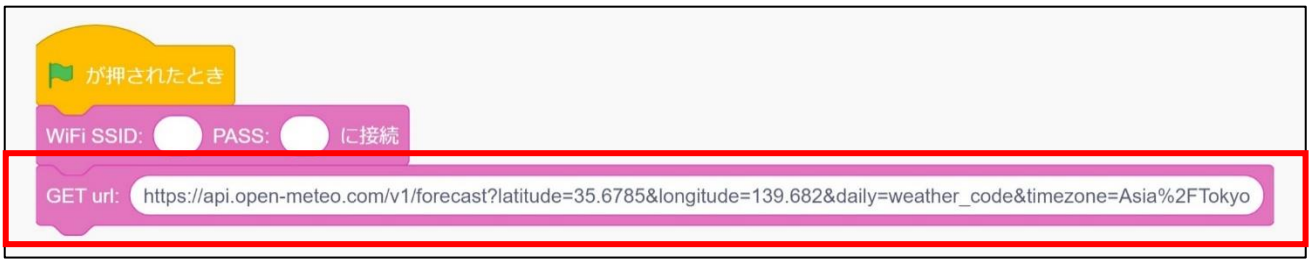
時刻以外の情報をサーバーから取得する場合は、GET url: の空欄に URL を入力して実行します。
取得したデータは レスポンス に保存されます。

試しに、下の URL から日本（東京）の 1 週間の天気を取得してみましょう。

```
https://api.open-meteo.com/v1/forecast?latitude=35.6785&longitude=139.682&daily=weather_code&timezone=Asia%2FTokyo
```

① GET url: の空欄に上記の URL をコピー＆ペーストしてメインユニットをサーバーに接続し、データを取得します。

※最初に SSID と PASS を入力して、Wi-Fi の設定をしてください。



②サーバーから保存したデータ レスポンス はそのままでは確認できないため、変数に保存して確認できるようにします。

変数「Response」を作成して、レスポンス を挿入しましょう。

スクリプトが実行されると、変数「Response」に以下の文章（テキストデータ）が保存されます。
このようなテキストデータの形式を「JSON（ジェイソン）」と呼びます。

Response

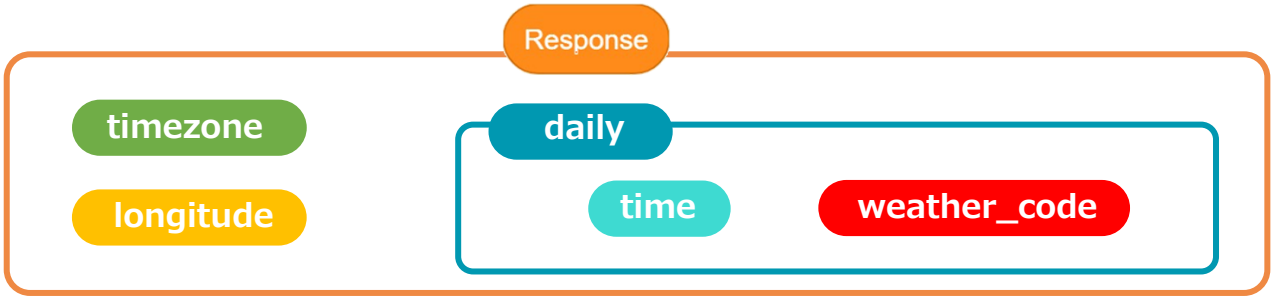
{"utc_offset_seconds": 32400, "timezone": "Asia/Tokyo", "longitude": 139.6875, "generationtime_ms": 0.04994869, "daily": {"time": ["2024-02-15", "2024-02-16", "2024-02-17", "2024-02-18", "2024-02-19", "2024-02-20", "2024-02-21"], "weather_code": [3, 53, 1, 3, 3, 61, 61]}, "elevation": 48.0, "daily_units": {"time": "iso8601", "weather_code": "wmo code"}, "latitude": 35.7, "timezone_abbreviation": "JST"}

この JSON データは内部で様々なグループにわかれています。
今回取得したデータを例にみると、変数「Response」の中にある**”daily”**というグループに、
”time”:[○○]という日付のデータと、
”weather_code”:[○○]という日付ごとの天気のデータが区分けされています。

Response

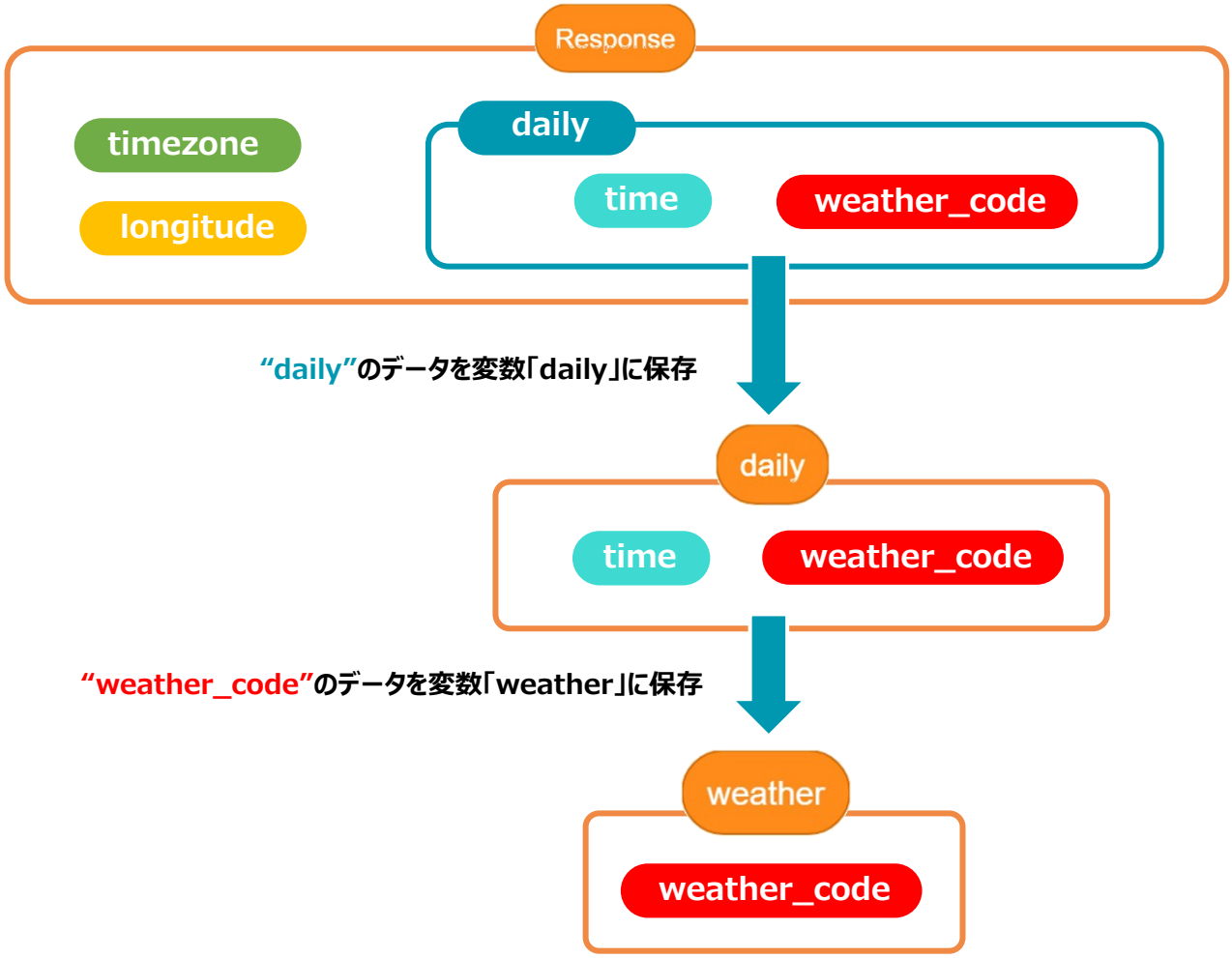
{"utc_offset_seconds": 32400, "timezone": "Asia/Tokyo", "longitude": 139.6875, "generationtime_ms": 0.04994869, "daily": {"time": ["2024-02-15", "2024-02-16", "2024-02-17", "2024-02-18", "2024-02-19", "2024-02-20", "2024-02-21"], "weather_code": [3, 53, 1, 3, 3, 61, 61]}, "elevation": 48.0, "daily_units": {"time": "iso8601", "weather_code": "wmo code"}, "latitude": 35.7, "timezone_abbreviation": "JST"}

変数「Response」に保存された JSON データには、**”daily”**の他に、東京のタイムゾーンのデータ（”timezone”）や経度のデータ（”longitude”）など、天気以外のデータも含まれています。

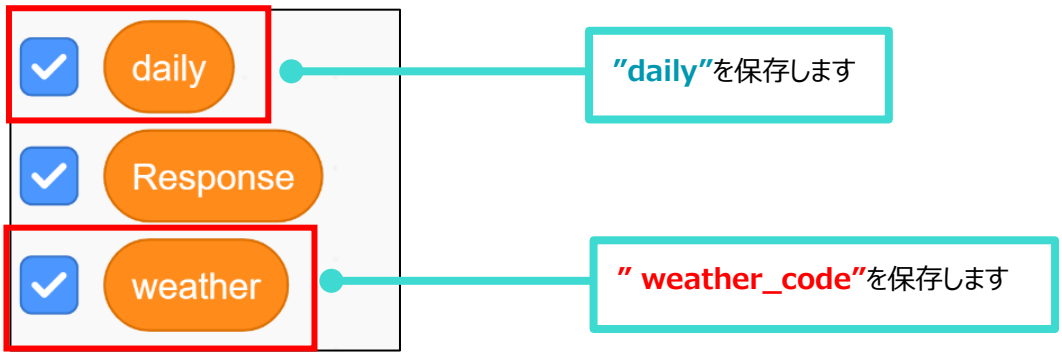


これから作成するプログラムでは**”weather_code”**のデータだけを使用するため、必要なデータだけを取り出して変数に保存しましょう。

③変数を作成して、JSON データから”weather_code”のデータだけを取り出して保存します。
”weather_code”は”daily”の中にあるため、一度”daily”のデータを取り出してから、”weather_code”を順番に取り出す必要があります。

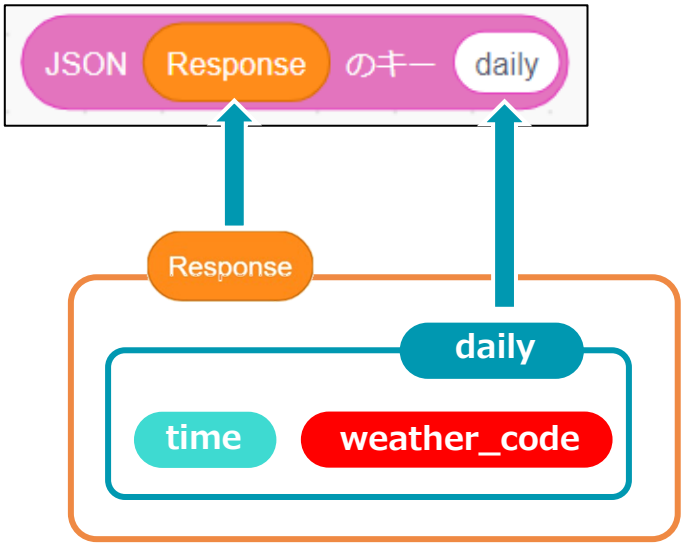


そのため、”daily”を保存する変数「daily」と、”weather_code”を保存する変数「weather」を作成します。



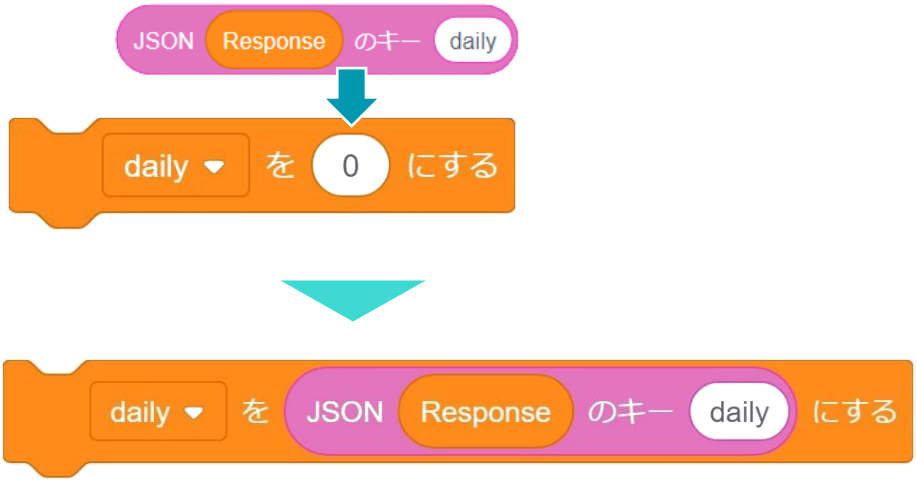
④取得したデータから必要な部分だけを選んで保存する場合は、 

を使います。
JSON には保存元となる JSON データの名前を、キーには保存したいデータのグループ名、または番号を入力します。



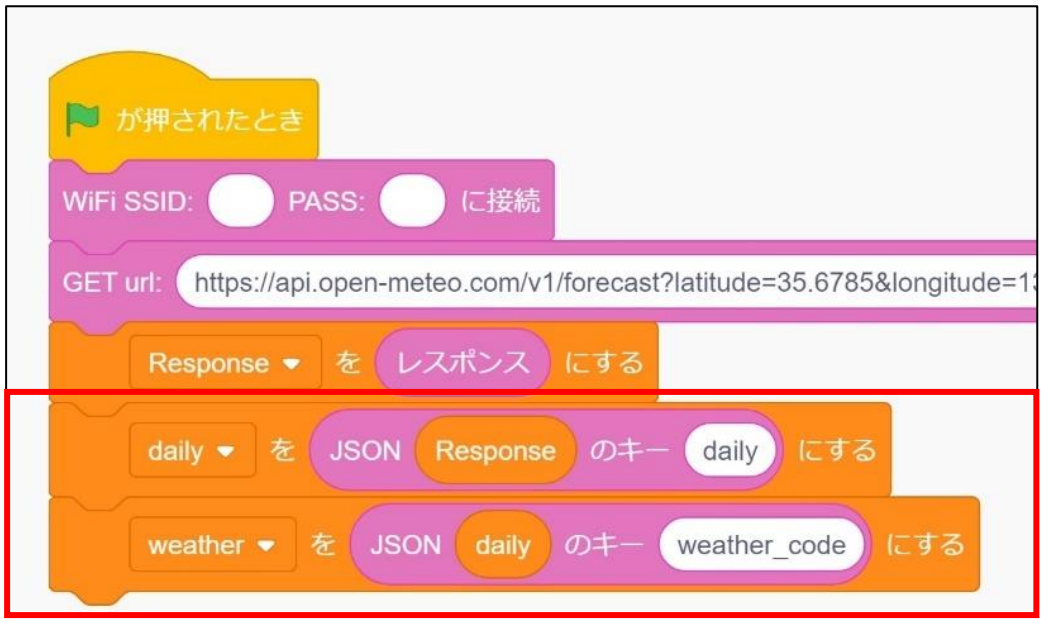
⑤ 

を変数「daily」に保存します。

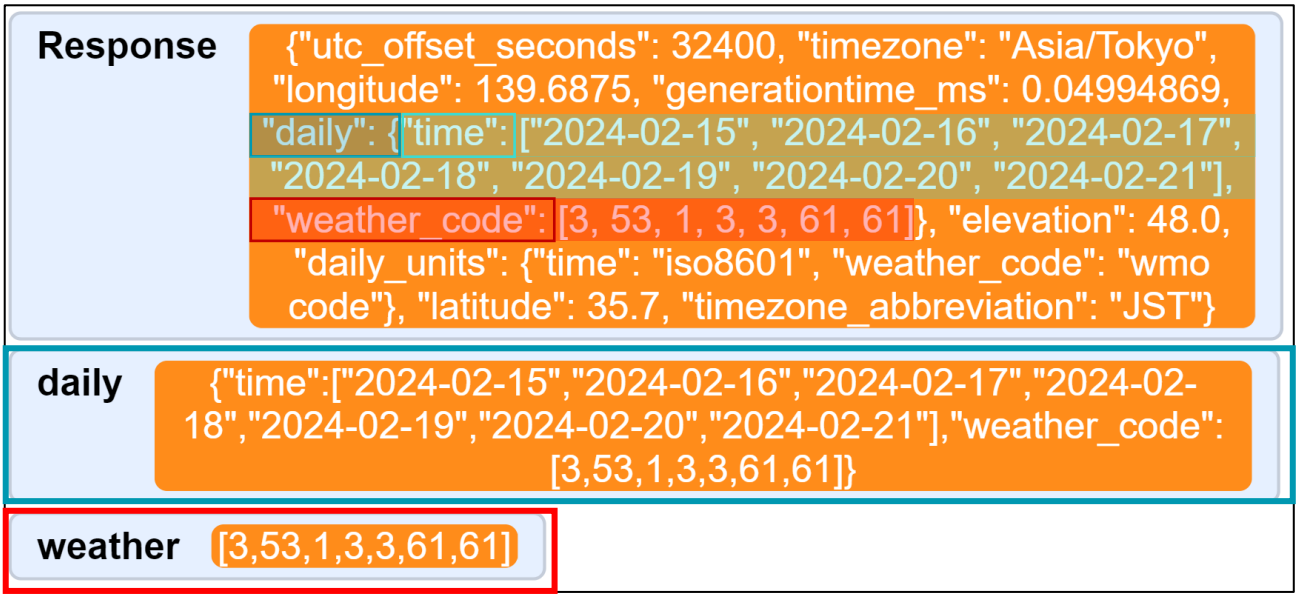


⑥同じように、変数「daily」に保存された JSON データから**"weather_code"**のデータを変数「weather」に保存するスクリプトを作成します。

⑦　⑤、⑥のスクリプトを②のスクリプトの下に繋げます。



このスクリプトを実行すると、変数「daily」に 1 週間の日付（**"time"**）と天気（**"weather_code"**）が、変数「weather」に 1 週間の天気(**"weather_code"**)が保存されます。



Weather_code(ウェザーコード)の読み取り方

ウェザーコードとは、世界気象機関（WMO:World Meteorological Organization）が定めた天気の識別番号です。すべての天気を 0~99 の番号に分類することで、世界中の誰に対しても天気を正確に伝えることができます。

Open-Meteo では、天気のデータを以下のウェザーコードで分類してデータを提供しています。

WMO Weather interpretation codes (WW)

Code	Description
0	Clear sky
1, 2, 3	Mainly clear, partly cloudy, and overcast
45, 48	Fog and depositing rime fog
51, 53, 55	Drizzle: Light, moderate, and dense intensity
56, 57	Freezing Drizzle: Light and dense intensity
61, 63, 65	Rain: Slight, moderate and heavy intensity
66, 67	Freezing Rain: Light and heavy intensity
71, 73, 75	Snow fall: Slight, moderate, and heavy intensity
77	Snow grains
80, 81, 82	Rain showers: Slight, moderate, and violent
85, 86	Snow showers slight and heavy
95 *	Thunderstorm: Slight or moderate
96, 99 *	Thunderstorm with slight and heavy hail

(*) Thunderstorm forecast with hail is only available in Central Europe

ウェザーコードのおおまかな天気の目安は以下の通りです。

晴　：0,1

曇り：2,3

雨　：51,53,55,56,57,61,63,65,66,67,80,81,82

例で取得した 1 週間のウェザーコード「3,53,1,3,3,61,61」を晴、曇り、雨の 3 種類で分類すると、

「曇り、雨、晴、曇り、曇り、雨、雨」

になります。

手順⑦では、1 週間のウェザーコードをサーバーから取得して変数に保存することができました。
しかし、変数は 1 つのデータしか保存できないため、7 日間のウェザーコードが 1 つのデータのまとまりとして変数「weather」に保存されています。

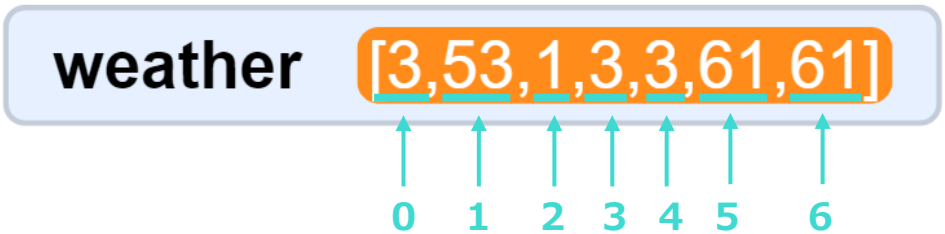
このままでは、1 日ごとのウェザーコードをプログラムで使用することができません。
データを使用するためには、ウェザーコードを 1 日分ずつ取り出して、リストに保存する必要があります。

⑧ウェザーコードを 1 日分ずつ保存するために、リスト「weather」を作成します。

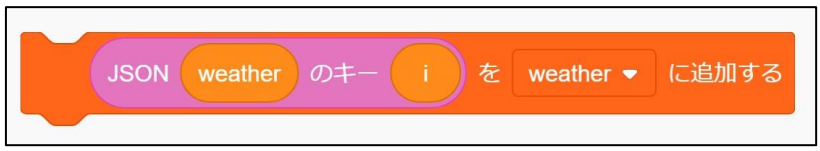
⑨データをリストの何番目に保存するかを指定するために、変数「i」を作成します。

⑩リスト「weather」に保存されているデータをすべて削除し、変数「i」を 0 にします。

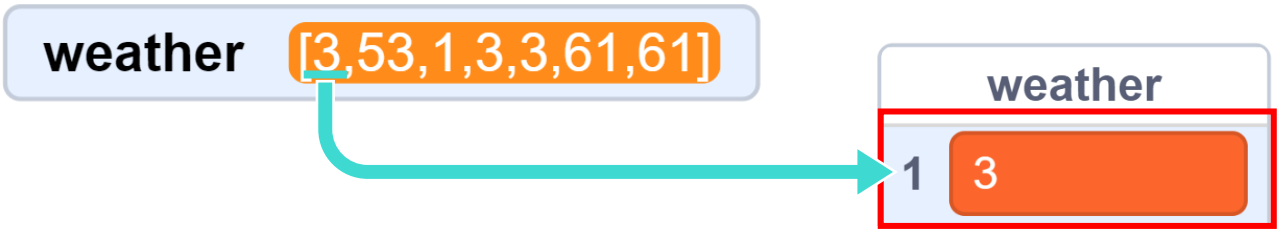
⑪変数「weather」の 0 番目から順番に、リストに追加するスクリプトをつくります。
※JSON データは左から順に 0 番目、1 番目、2 番目と数えます。



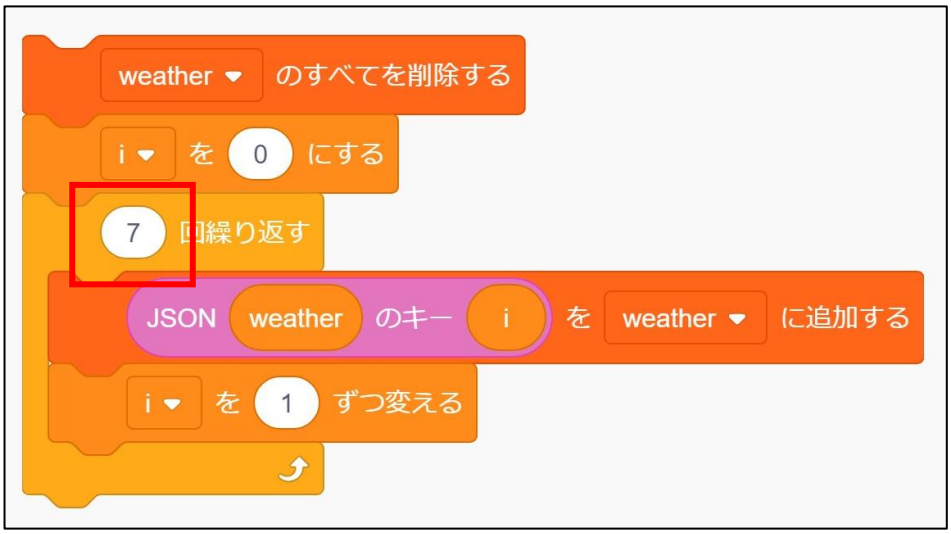
指定する JSON データを変数「weather」に、キーを変数「i」にして以下のブロックを実行すると、変数「weather」の「i」番目のデータをリストに追加することができます。



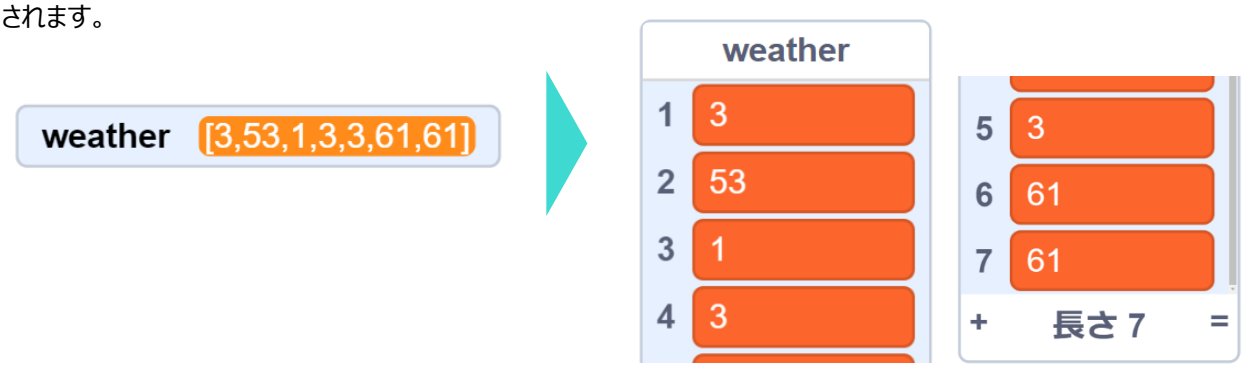
例） i 0 のとき、変数「weather」の 0 番目をリストに追加



⑫今回は 0 ～ 6 番目までの 7 つのウェザーコードが保存されているため、⑦を実行した後に変数「i」に 1 を足すスクリプトを 7 回繰り返して、リストに 7 つのデータを追加します。



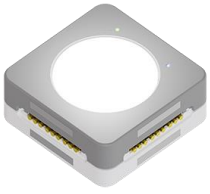
スクリプトを実行すると、変数「weather」に保存されていた 7 つのウェザーコードが、リスト「weather」に順番に追加されます。



天気予報モニターを作成しよう

リスト「weather」に保存したデータを用いて、１週間の天気を音声とLEDの光で知らせてくれるプログラムを作成します。
霧や雪など細かく区別して予報することも可能ですが、ここでは簡単に晴、曇り、雨の３種類で予報します。

晴

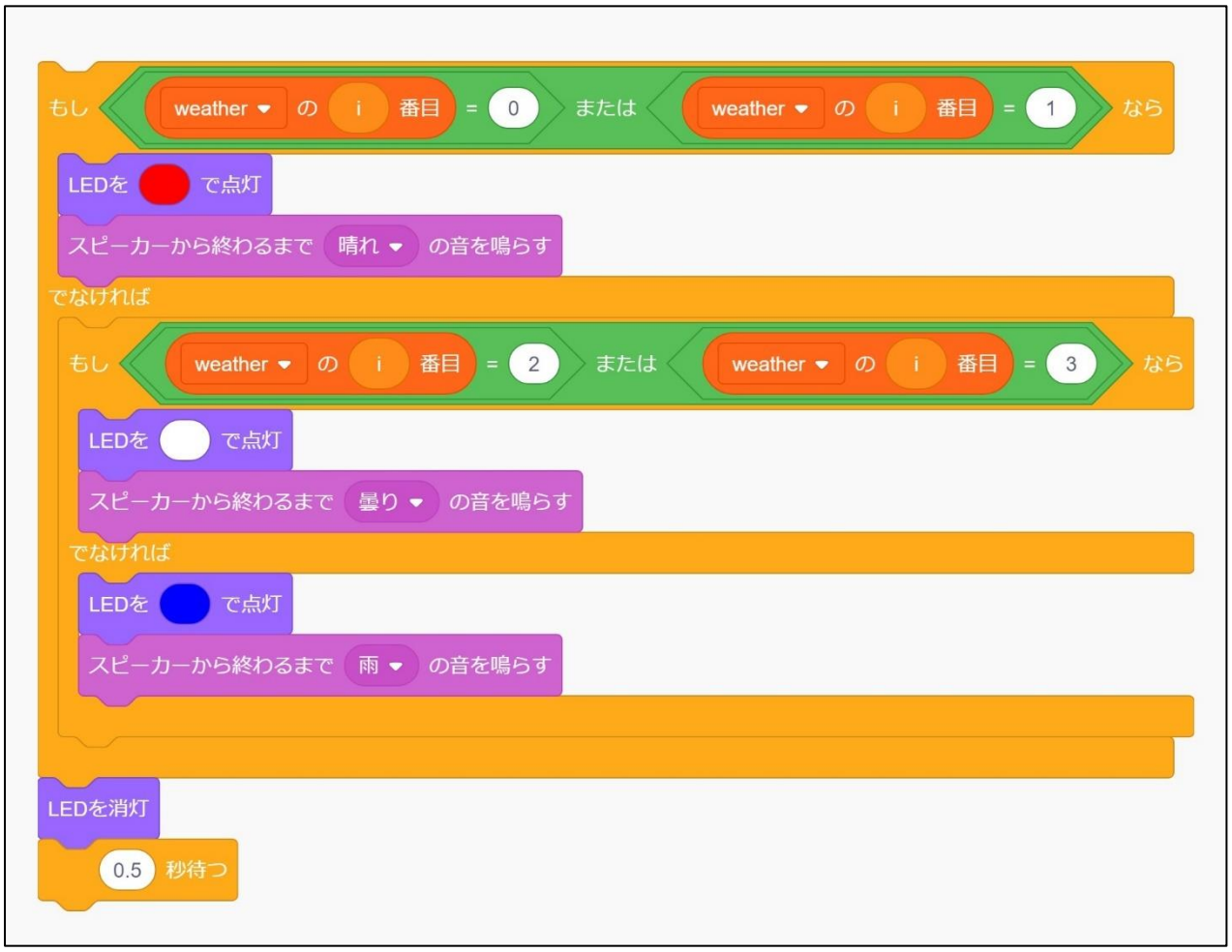


曇り

雨

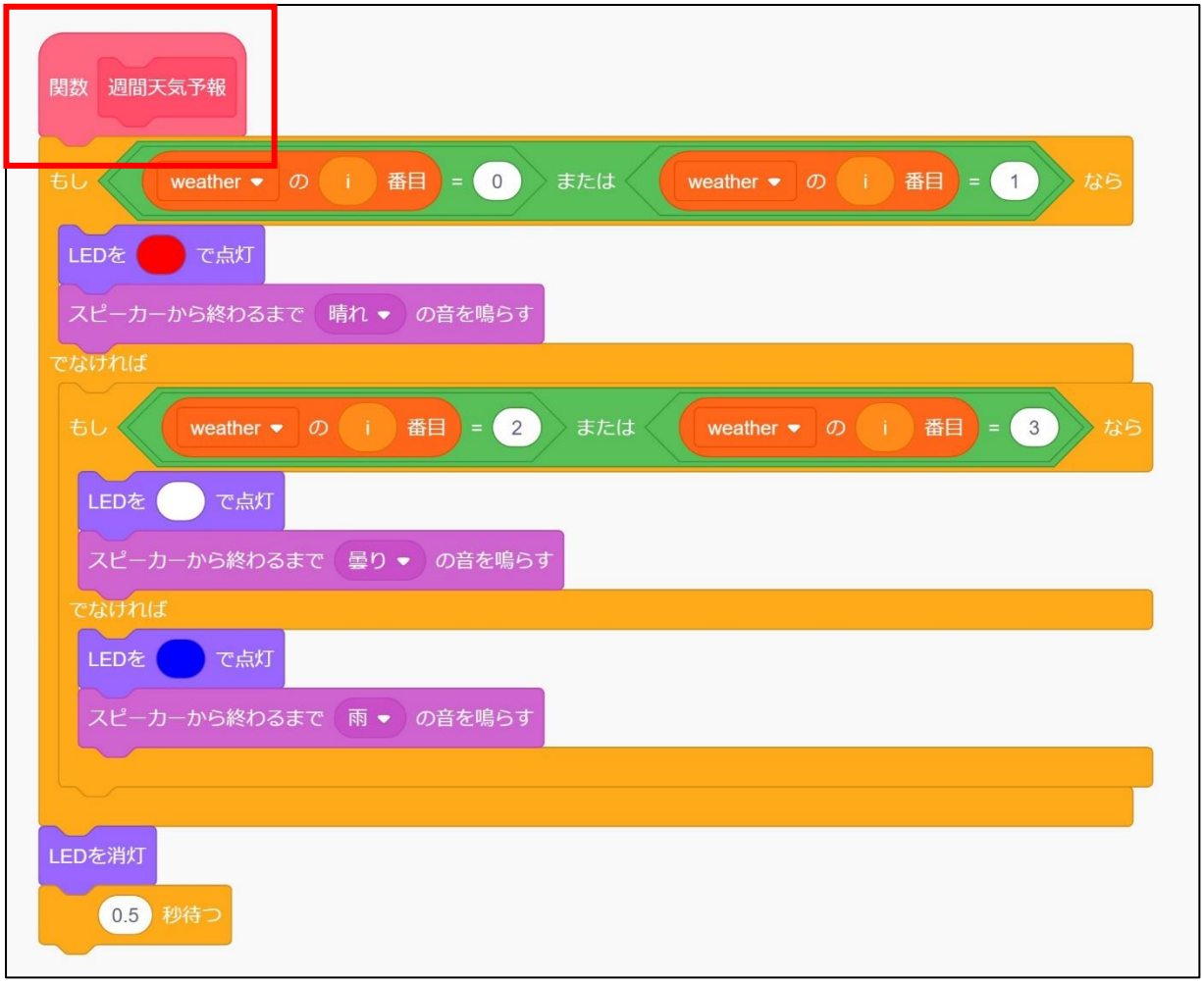
サンプルプログラムは
ここをクリック

①リスト「weather」の変数「i」番目に保存されている数値が「0」または「1」のときは赤色で点灯、「2」または「3」のときは白色で点灯、それ以外のときは青色で点灯させて、それぞれ音声を再生します。
また、LED の色を確認しやすくするために、点灯した後に一定時間待って消灯させます。



※プログラム内で使用する音声は自身の声を録音したり、音声ファイルを追加したりしましょう。

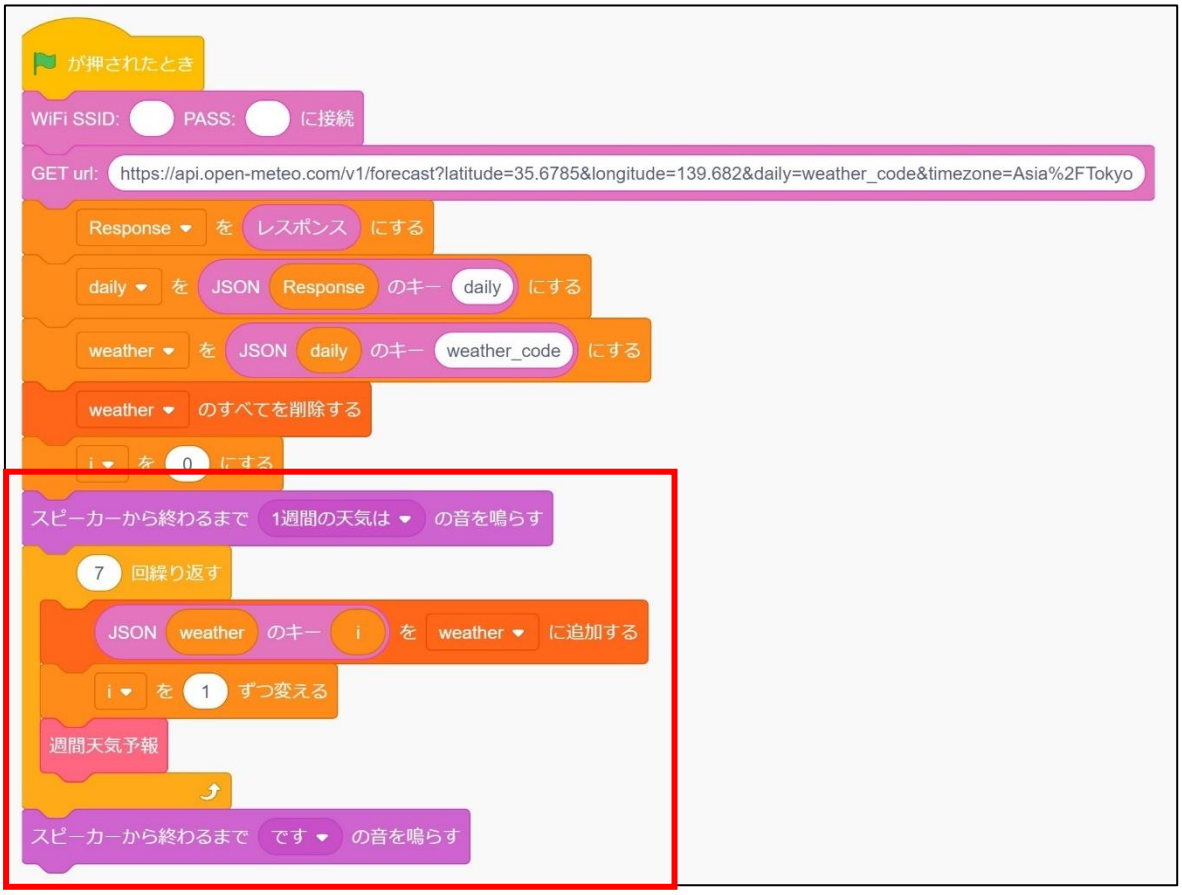
②関数「週間天気予報」を作成して、①を下に繋ぎます。



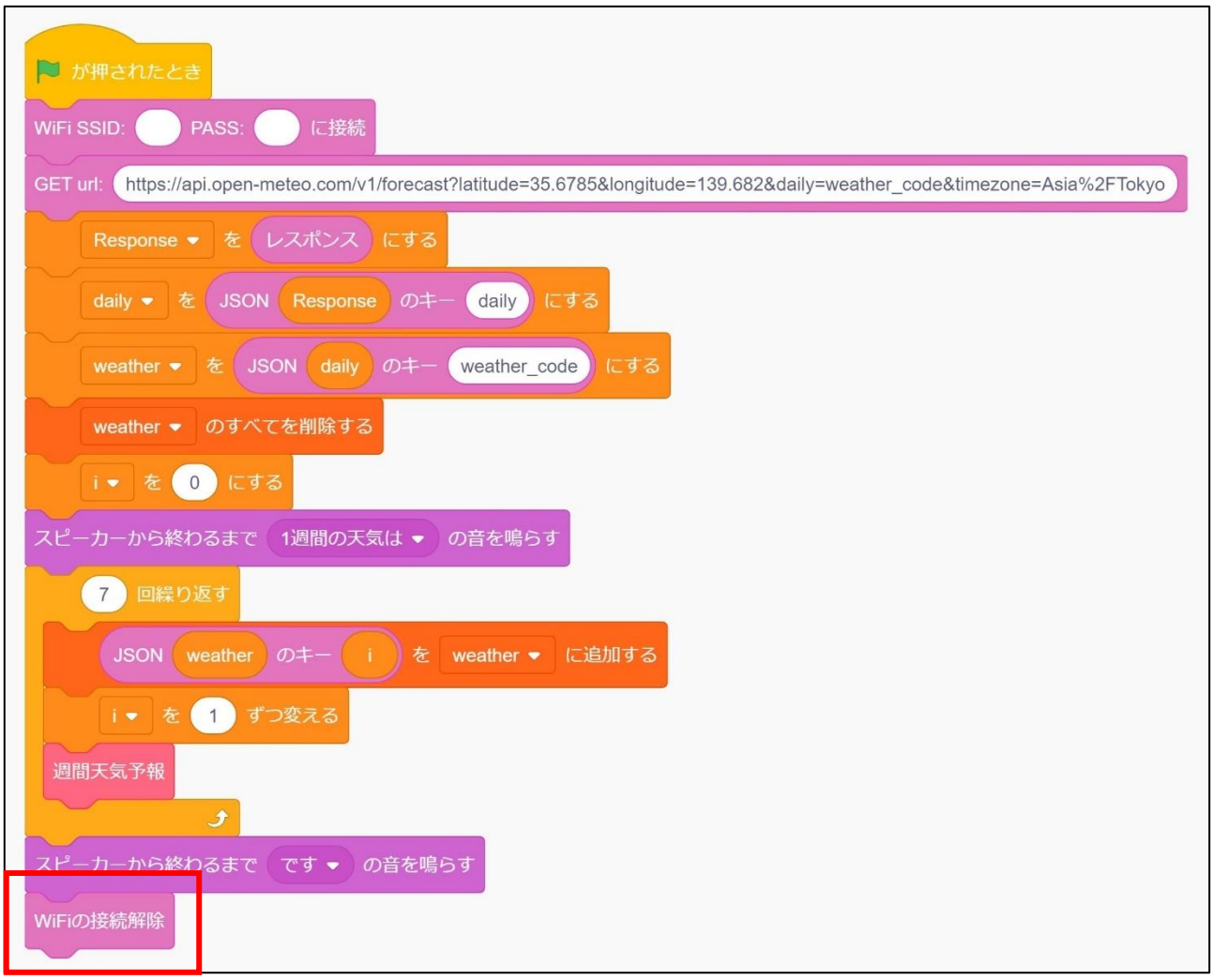
③p.100 で作成したスクリプトの「7 回繰り返す」の中に、定義「週間天気予報」を繋げます。

④天気予報の音声が自然な文章で再生されるように、「7 回繰り返す」の前後で「1 週間の天気は」と「～です」という音声を再生します。

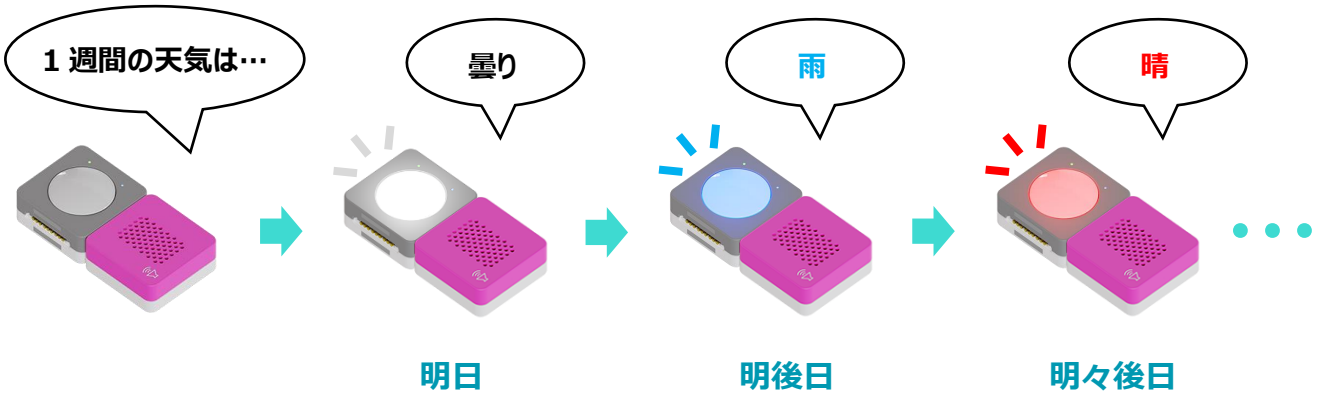
⑤④を p.97 で作成したスクリプトの下に繋げます。



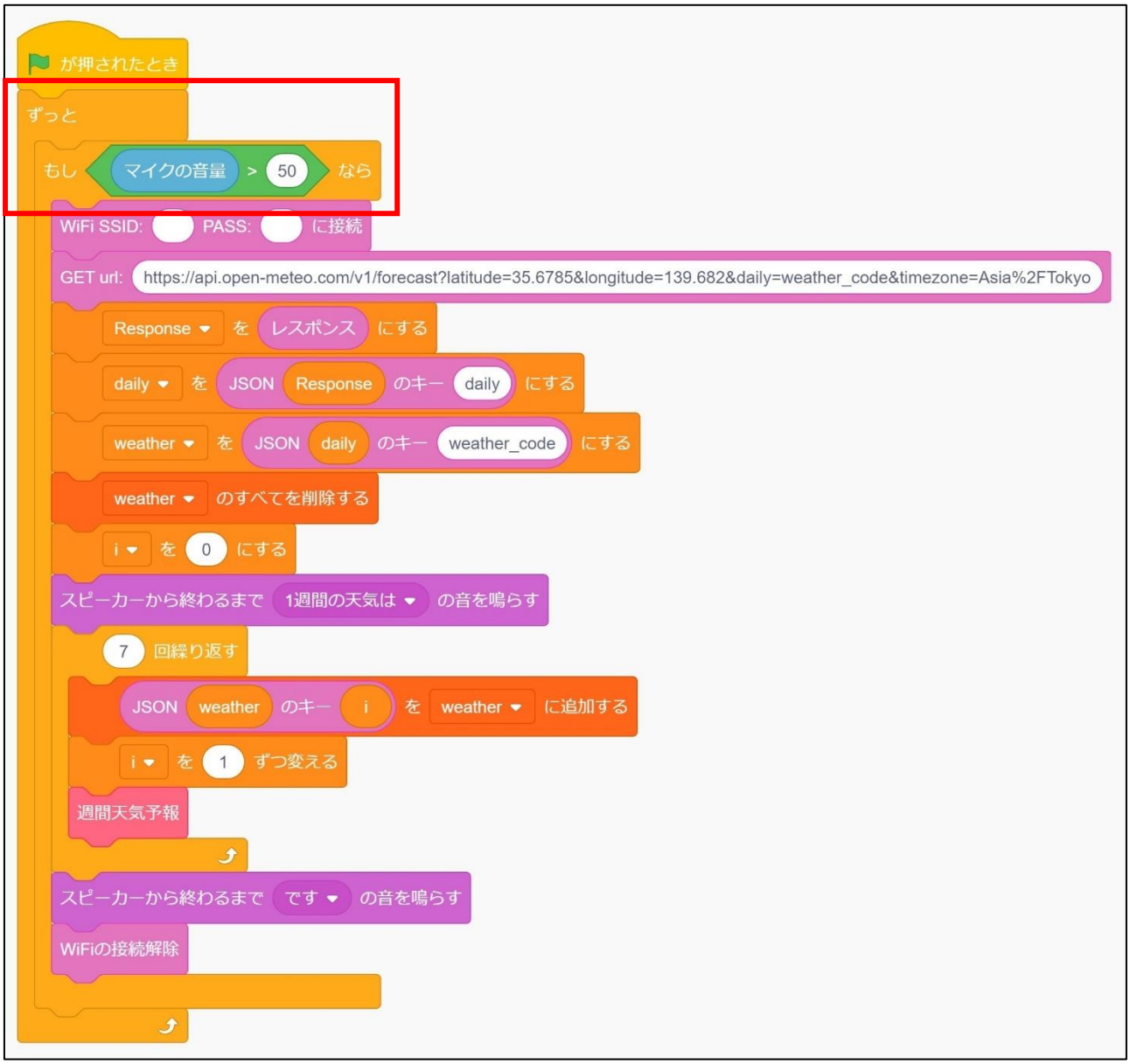
⑥スクリプトの最後に WiFiの接続解除 を実行して、Wi-Fi との接続を解除します。



スクリプトを実行すると、1 週間の天気予報を音声と LED の光で知らせてくれます。



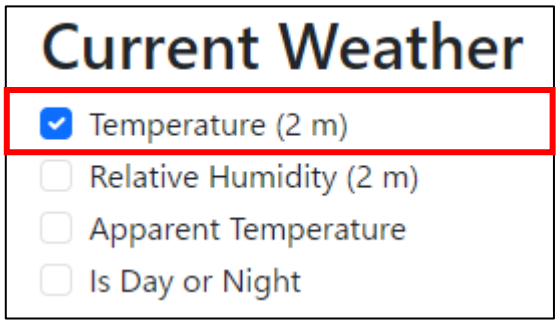
⑦メインユニットにマイクを接続して、マイクが大きい音を検知したときに天気予報を実行することもできます。



様々なデータを取得してみよう

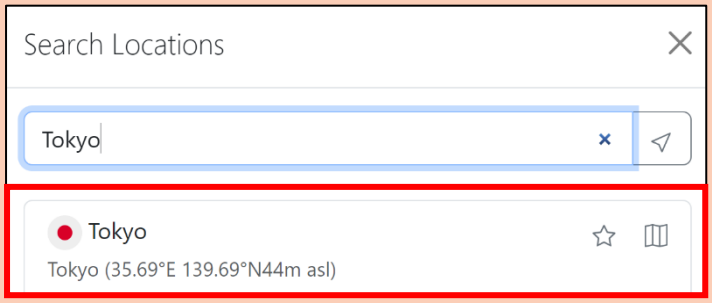
4-3 のプログラムでは 1 週間の気温を取得しましたが、Open-Meteo では取得する日付と日数を変えたり、気温の他に湿度や気圧などを取得したりすることができます。
例として、現在の気温のデータを取得してみましょう。

①Open-Meteo にアクセスして「Current Weather」の「Temperature（2m）」に□をクリックして✓を入れます。
「Temperature（2m）」以外の不要な箇所に☑が入っているときは、☑をクリックして✓を外しましょう。



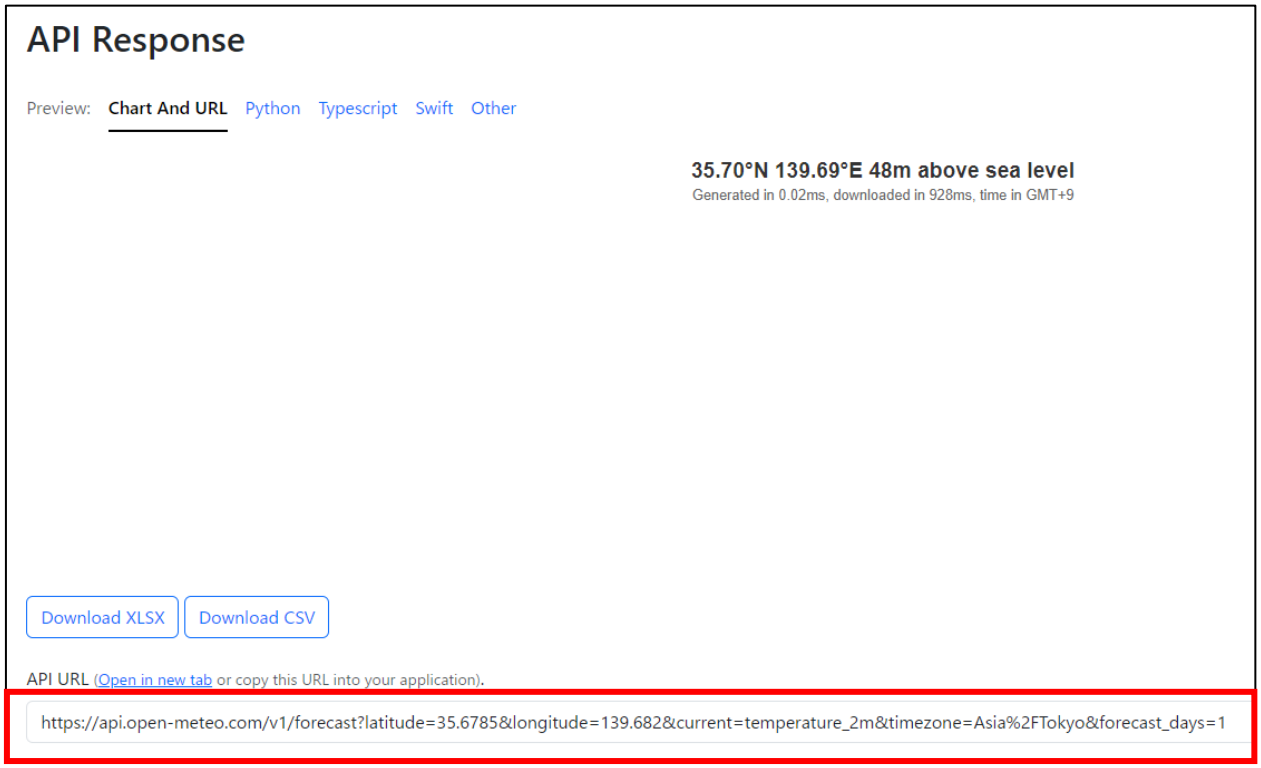
※最初の設定では、取得するデータの地域の座標(緯度と経度)はドイツの座標に設定されています。
また、タイムゾーンはイギリスの標準時が設定されているため、取得したい地域の座標・タイムゾーンを設定します。
例）東京：緯度　35.6895 、経度　139.691 、タイムゾーン　Asia/Tokyo（UTC+9）

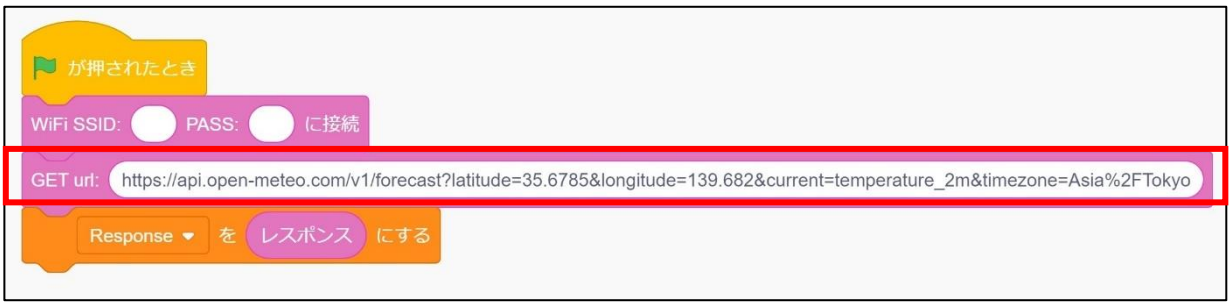
座標は「Location and Time」の「Search」から地域名を入力して選択することでも設定できます。



②取得するデータを変更すると、URL も変更されています。
ページを下にスクロールし、新しく設定された URL をコピーして 

に貼り付けましょう。





③②のスクリプトを実行すると、指定した地域における現在の気温データが、変数「Response」に保存されます。

Response

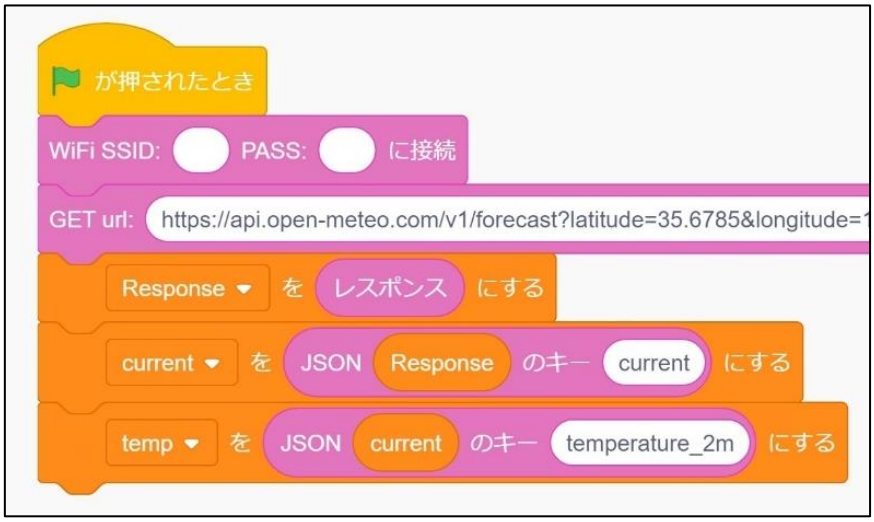
{"utc_offset_seconds": 32400, "timezone": "Asia/Tokyo", "longitude": 139.6875, "generationtime_ms": 0.01609326, "current": {"time": "2024-02-16T09:30", "temperature_2m": 7.2, "interval": 900}, "elevation": 48.0, "latitude": 35.7, "current_units": {"time": "iso8601", "temperature_2m": "°C", "interval": "seconds"}, "timezone_abbreviation": "JST"}

今回取得した JSON データでは、”current”というグループに
”time”(時刻)、”temperature_2m”(気温)が区分けされています。

Response

{"utc_offset_seconds": 32400, "timezone": "Asia/Tokyo", "longitude": 139.6875, "generationtime_ms": 0.01609326, "current": {"time": "2024-02-16T09:30", "temperature_2m": 7.2, "interval": 900}, "elevation": 48.0, "latitude": 35.7, "current_units": {"time": "iso8601", "temperature_2m": "°C", "interval": "seconds"}, "timezone_abbreviation": "JST"}

④変数「current」、「temp」を作成し、必要なデータだけを抽出して変数に保存しましょう。



Response

{"utc_offset_seconds": 32400, "timezone": "Asia/Tokyo", "longitude": 139.6875, "generationtime_ms": 0.02098083, "current": {"time": "2024-02-16T09:30", "temperature_2m": 7.2, "interval": 900}, "elevation": 48.0, "latitude": 35.7, "current_units": {"time": "iso8601", "temperature_2m": "°C", "interval": "seconds"}, "timezone_abbreviation": "JST"}

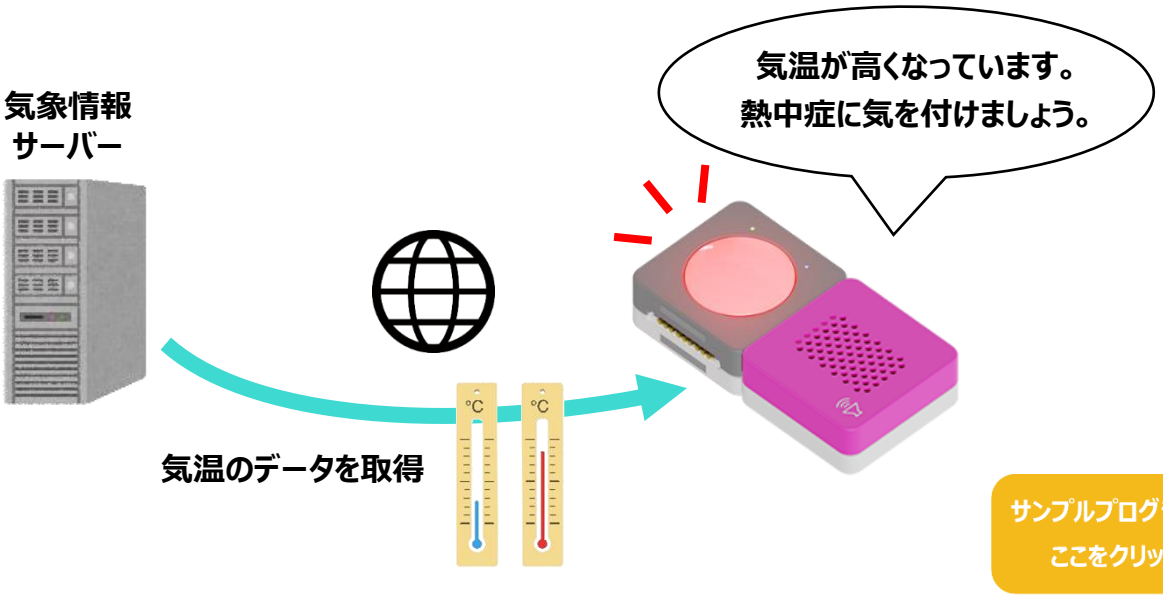
current {"time":"2024-02-16T09:30","temperature_2m":7.2,"interval":900}

temp 7.2

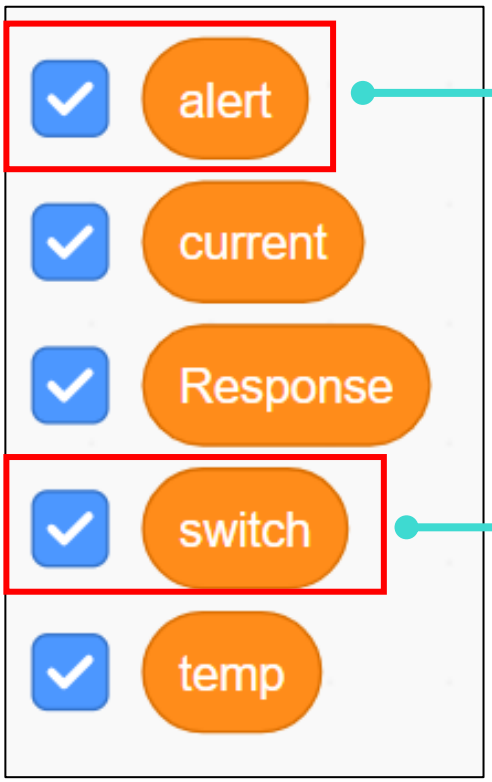
取得するデータを変更することで、様々なデータを利用したプログラムを作成できます。
次のページでは、現在の気温データを用いたプログラムを紹介しています。

熱中症アラートを作成しよう

取得した現在の気温データを利用して、熱中症アラートを作成しましょう。
気温が 30℃を超える場合に LED を赤く点灯させて、スピーカーから警告音や注意喚起のアナウンスを再生させます。



①変数「alert」と「switch」を作成します。



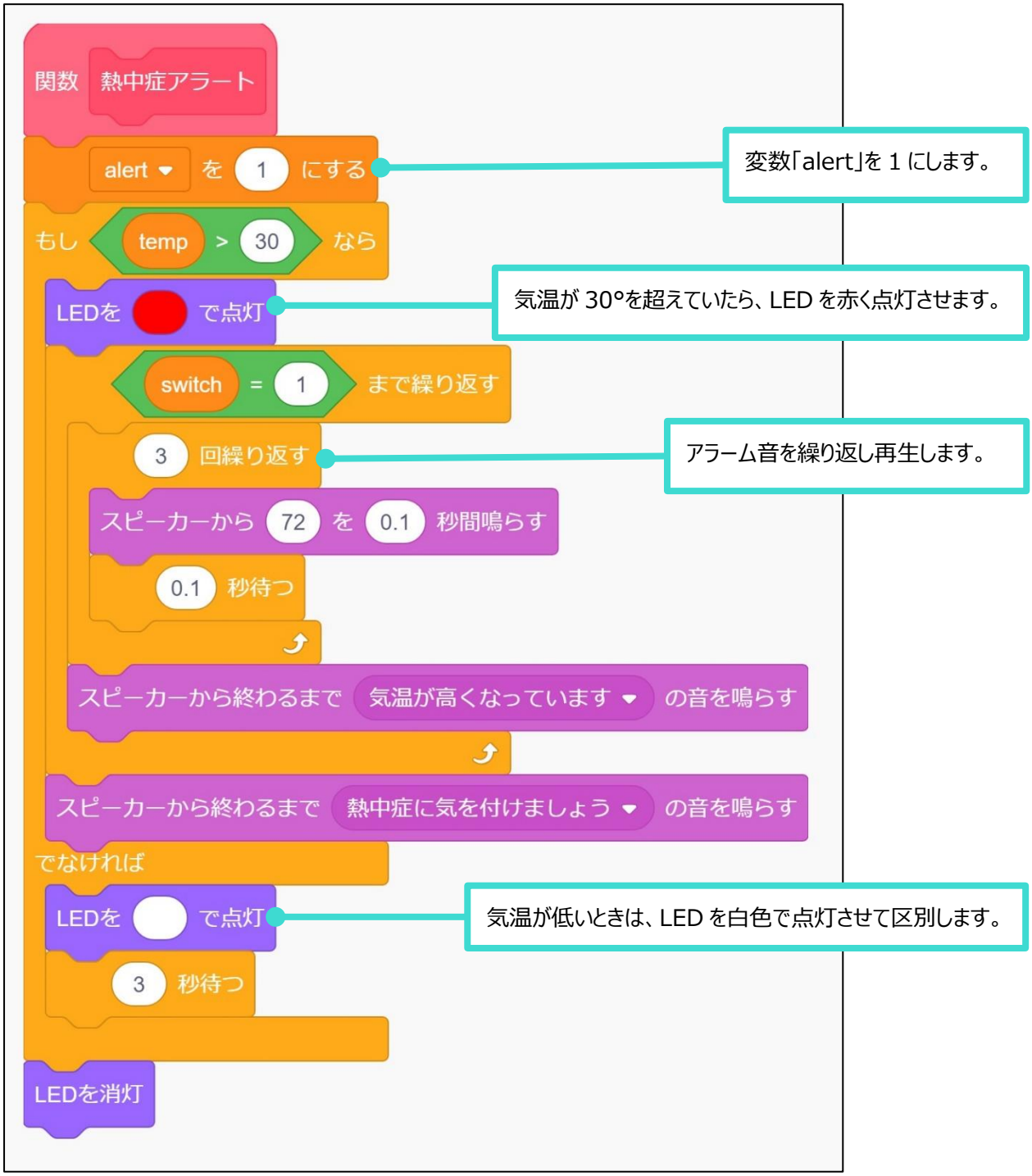
熱中症アラートが実行されているか判別します。
熱中症アラートが実行中は「1」、
実行していないときは「0」にします。

熱中症アラートが実行中にボタンが押されたか判別します。
ボタンが押されたら「1」
ボタンが押されていなければ「0」にします。

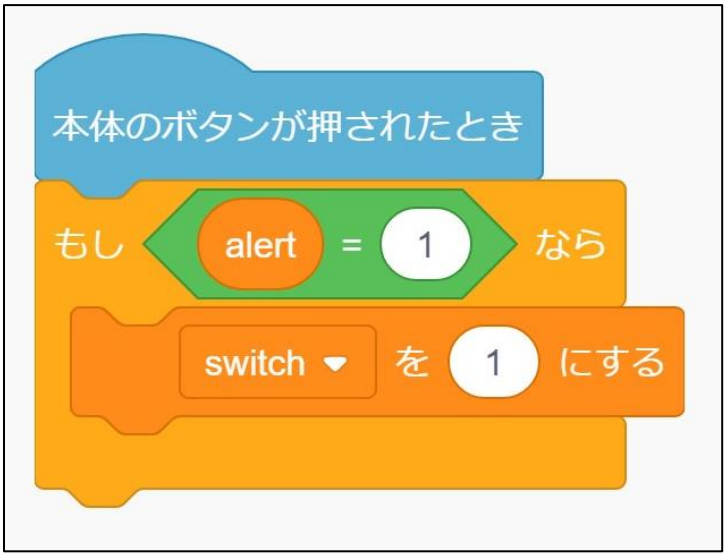
②関数「熱中症アラート」を作成します。

関数が実行されたとわかるように変数「alert」を 1 にした後、気温が 30℃を超える（変数「temp」が 30 より大きい）ときに LED を赤く点灯させます。

そして、ボタンを押す（変数「switch」が 1 になる）まで、ブザー音と警告音を繰り返し再生します。

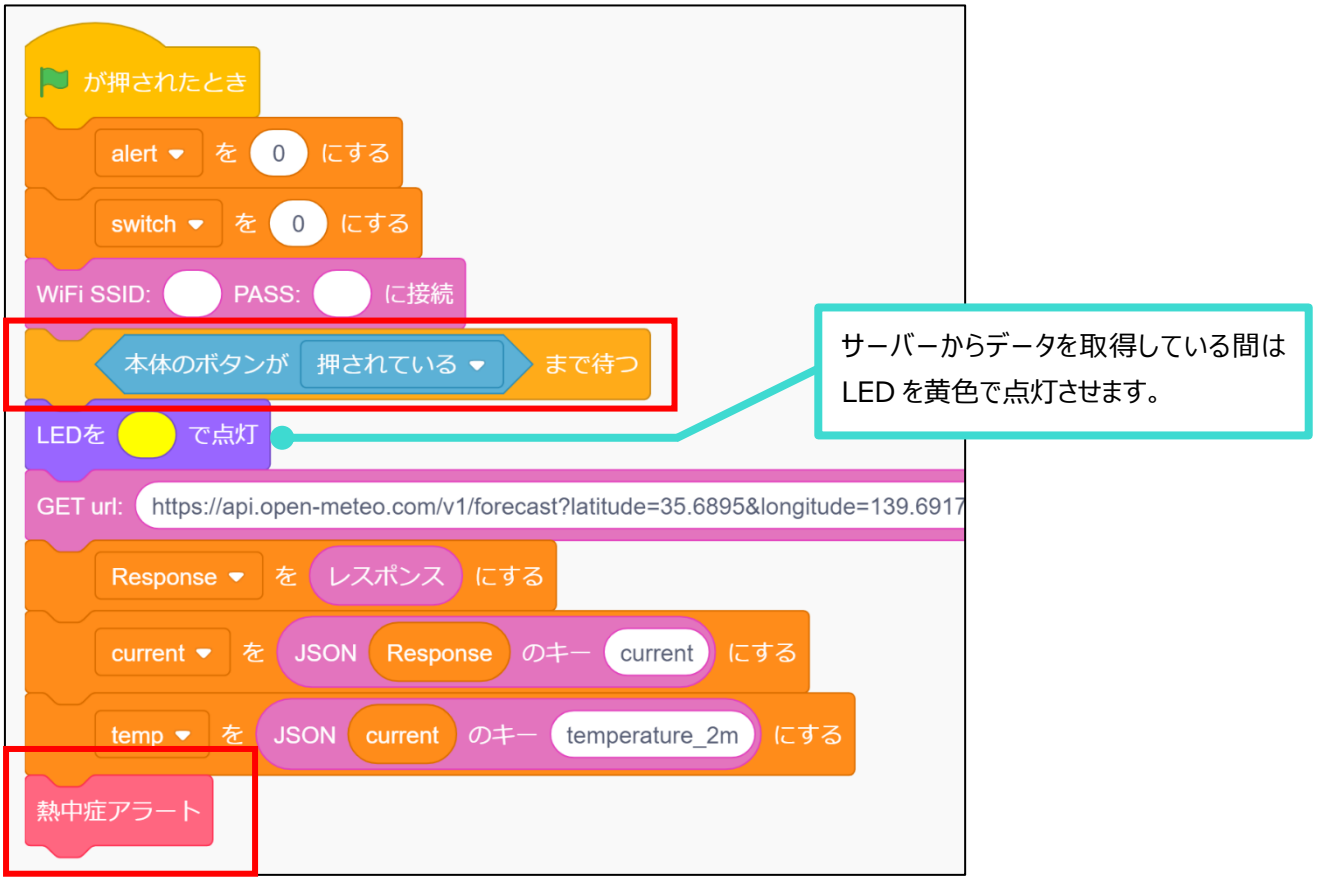


③ボタンを押したとき、関数「熱中症アラート」が実行中なら、変数「switch」を 1 にします。

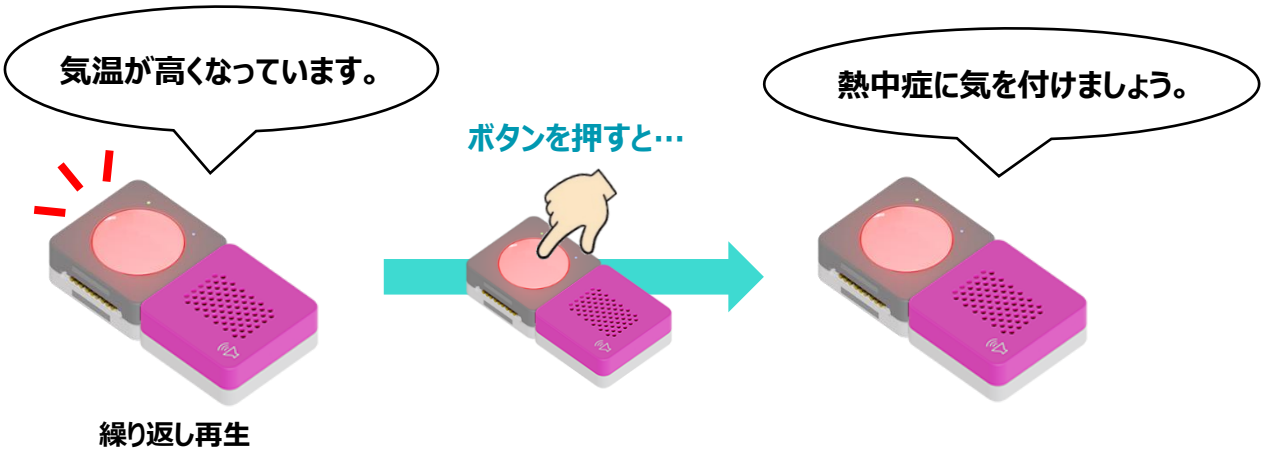


④p.108 で作成したスクリプトの下に、定義「熱中症アラート」を繋げます。
また、ボタンを押したタイミングで熱中症アラートが実行されるように、サーバーからデータを取得する前に

を入れます。



プログラムを実行すると、現在の気温が 30°を超えていた場合に熱中症アラートが実行されます。
熱中症アラートが実行されると、ボタンを押すまで警告音が繰り返し再生されます。



4 章では、タイムサーバー「ntp2.plala.or.jp」と気象情報サーバー（Open-Meteo）の 2 種類のサーバーから JSON データを取得するプログラムを紹介しました。
これらのサーバー以外にも、地震や津波の速報を提供するサーバーや、テレビの番組表を提供するサーバーなど、JSON データを提供してくれるサーバーは数多く存在しています。
自分が欲しいデータによってサーバーを選択して、便利なプログラムを作成しましょう！

サーバーによっては、事前に会員登録したり、データの取得回数によってお金を支払ったりする必要があります。
必ず使用条件を確認してから利用しましょう。

5 その他作品の紹介

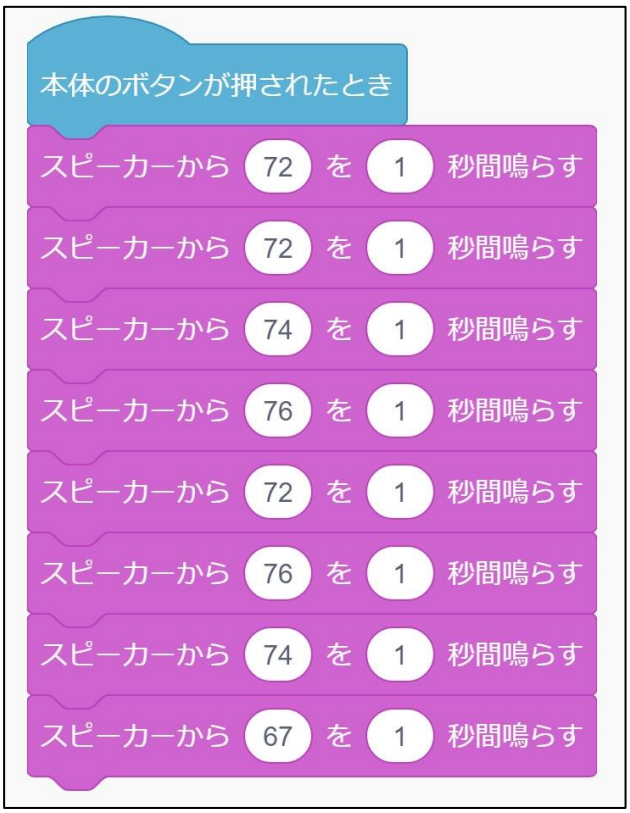
拡張ユニットの使い方によって、様々な便利な装置やシステムを作成することができます。
5 章では、いくつかの参考プログラムを紹介します。

サンプルプログラムは
ここをクリック

5-1 ミュージックプレイヤー

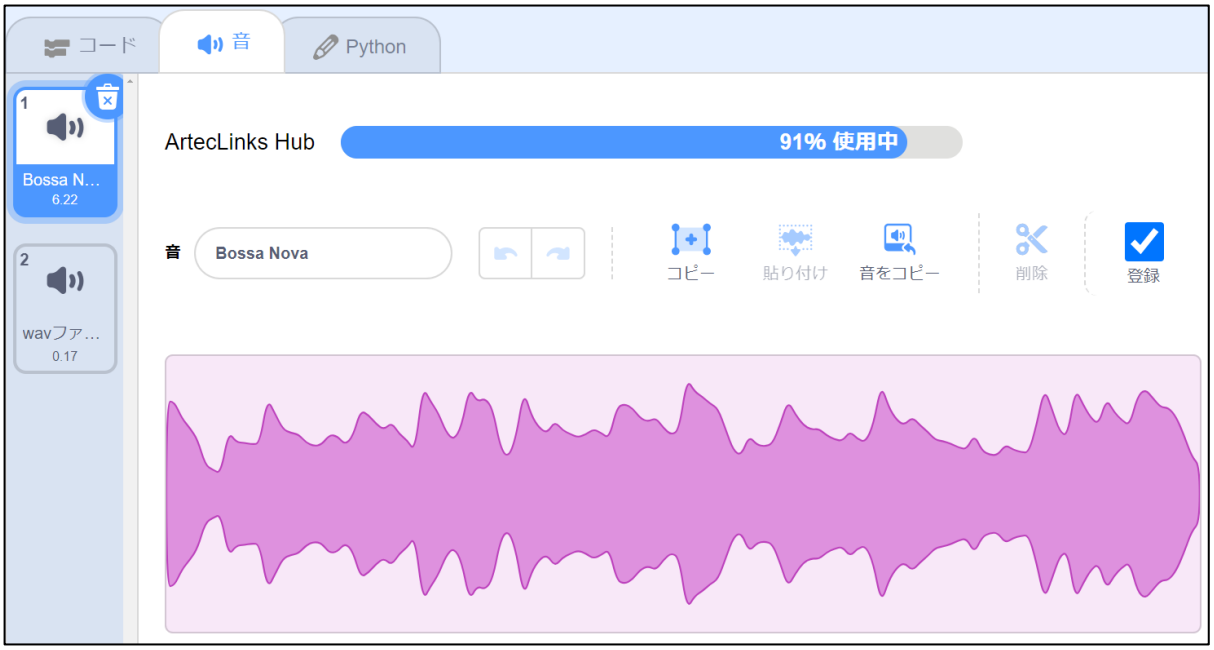
ボタンを押して好きな音楽を再生できるプログラムです。

本体のボタンが押されたとき の下に音を鳴らすブロックを自由に並べて、好きな楽曲を再生してみましょう。



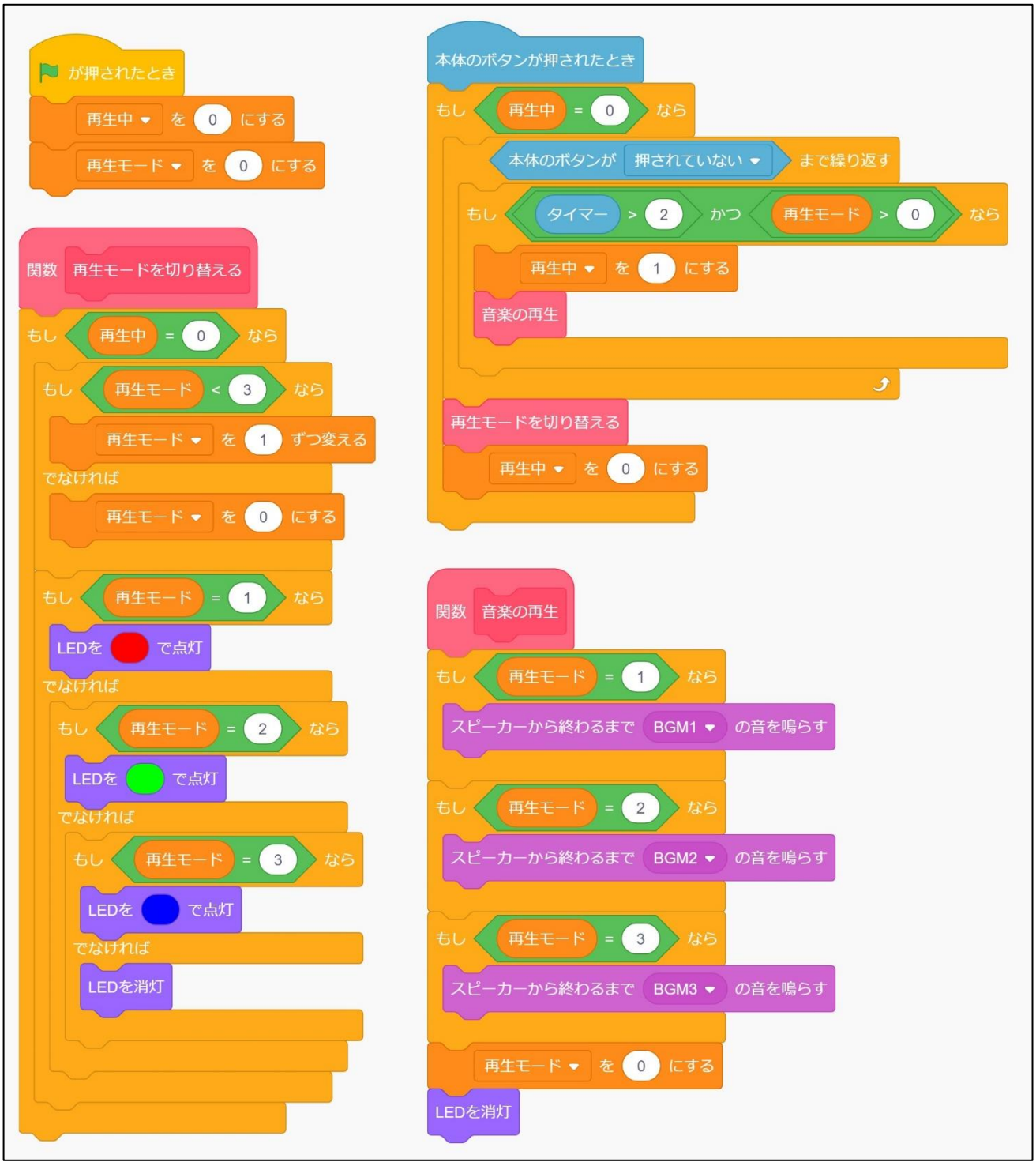
ソフトウェアに用意されている音データや wav ファイルを追加して、再生することもできます。

（音データの追加方法 → p.30）



発展編：複数の音データを切り替えて再生しよう

3-3：発展編（p.66）で作成した関数「録音モードを切り替える」を応用すると、ボタンを短く押して再生される音データを切り替え、長押して再生するプログラムにつくりかえることもできます。



5-2 ボイスレコーダー

2-2（p.27）では、 

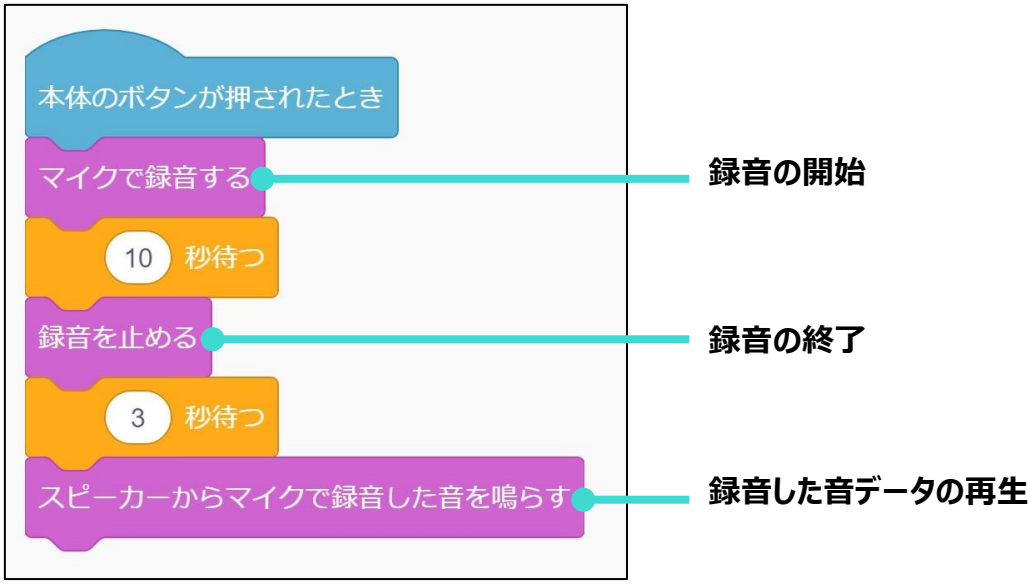
で事前に録音した音データを再生する方法を紹介しました。

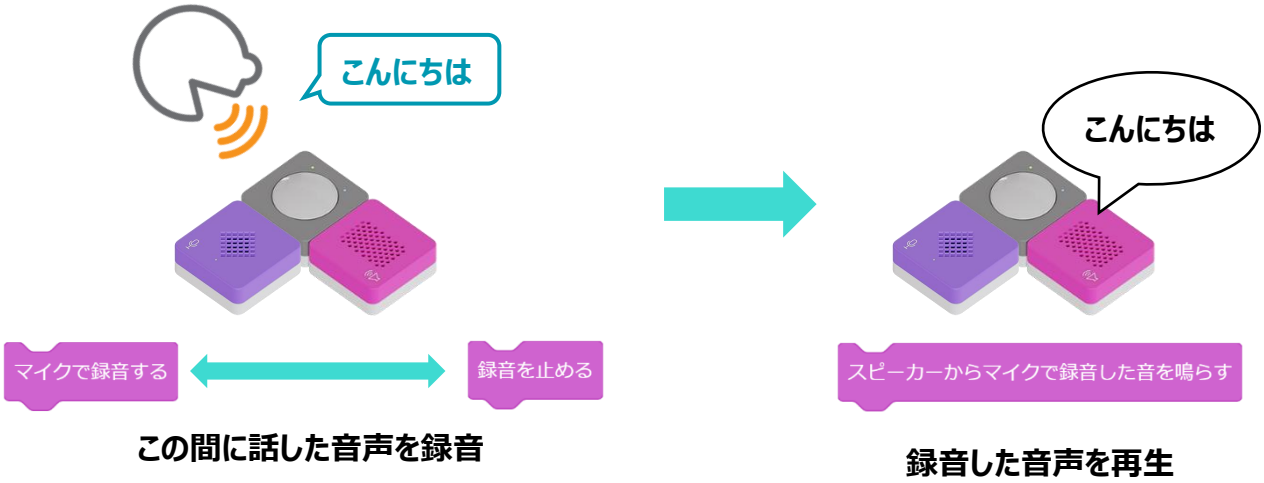
今回は事前に録音せずに、プログラムの実行中に音データを録音して再生する方法を紹介します。

マイクで録音する を実行すると、録音を止める が実行されるまでマイクで計測した音データが録音されます。

録音した音は スピーカーからマイクで録音した音を鳴らす を実行すると再生できます。

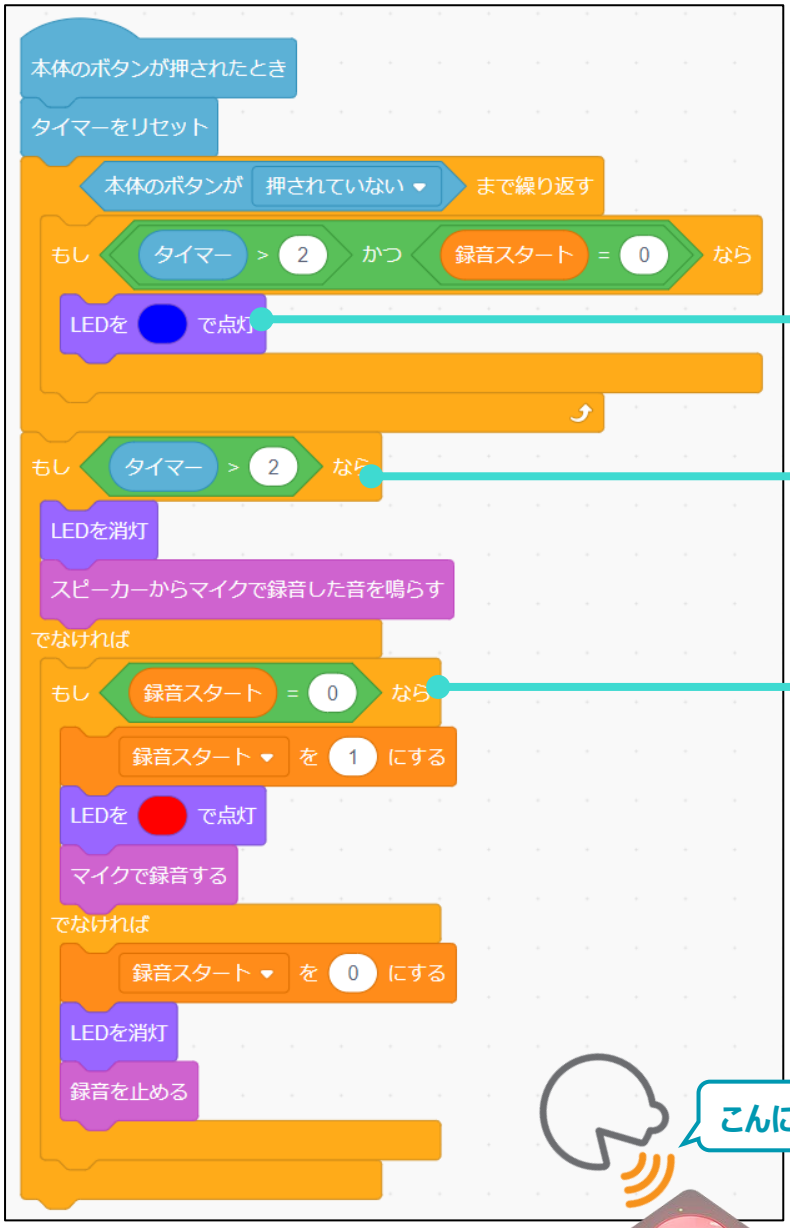
例）ボタンを押してから 10 秒間録音して、3 秒後に再生するプログラム





ボタンを長押しすると録音を開始できる、ボイスレコーダーのプログラムです。
もう一度ボタンを押すと、録音を終了します。
録音された状態でボタンを長押しすると、録音した音声を再生できます。

変数「録音スタート」の数値によって、
録音中かどうかを区別します。
0 の場合 → 録音されていない
1 の場合 → 録音中

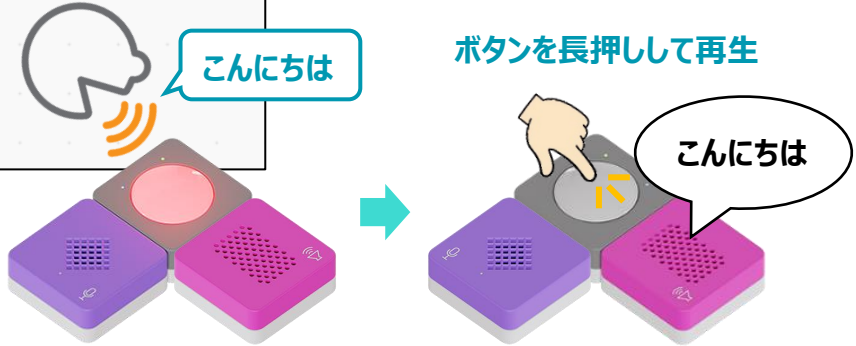


ボタンが 2 秒以上長押しされたかわかるように LED を青色で点灯させます。

ボタンが 2 秒以上押された場合、LED を消灯して録音した音を再生します。

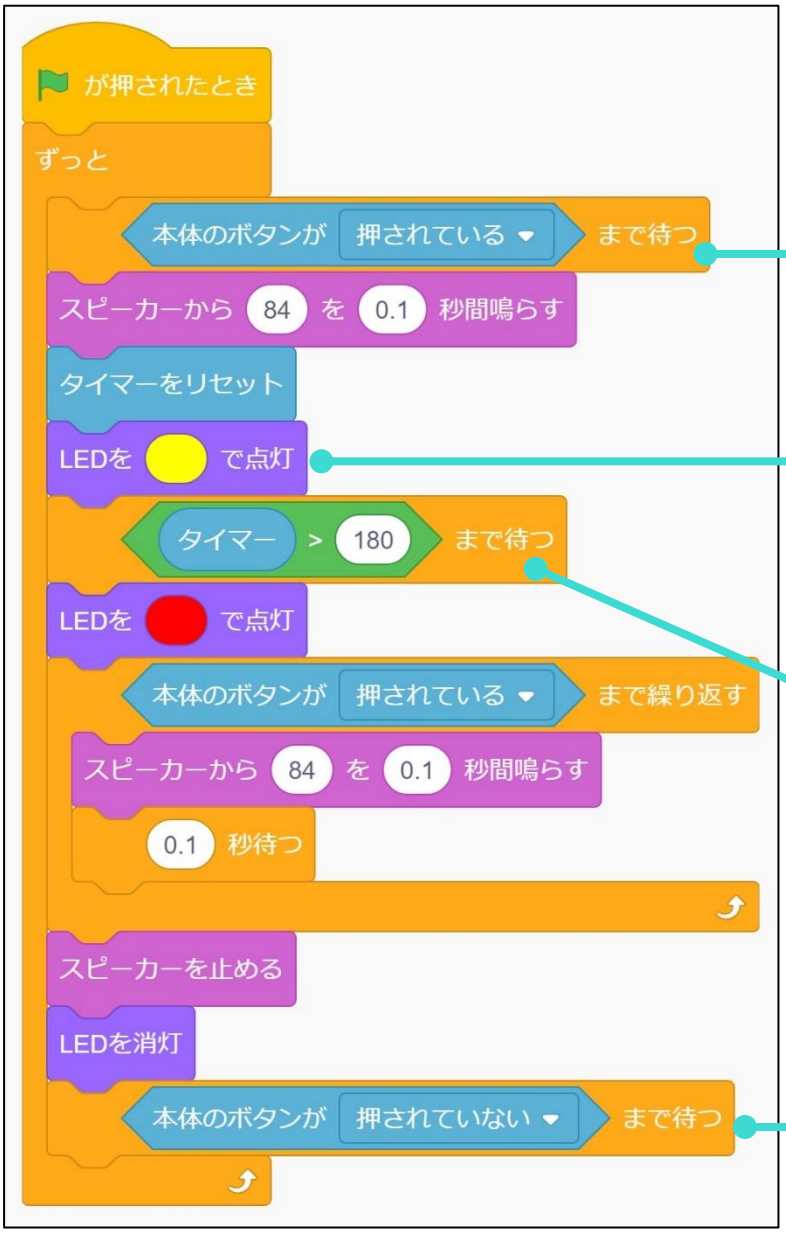
ボタンを押された時間が 2 秒以下の場合

録音中ではないとき(録音スタート＝0)
→LED を点灯して、録音を開始します。
録音中のとき(録音スタート＝0)
→LED を消灯して、録音を終了します。



5-3 カップラーメンタイマー

3 分間を計測できるカップラーメンタイマーのプログラムです。
ボタンを押してから 3 分後にアラーム音が鳴り、ボタンを押すとアラーム音が鳴りやみます。



ボタンが押されたら、タイマーをリセットして 3 分間計測します。

3 分計測中だとわかるように、計測中と計測後で、異なる色で LED を点灯させます。

3 分経過後、ボタンを押すまでアラーム音を鳴らします。

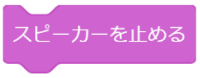
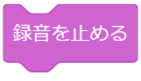
ボタンを押してアラーム音を止めた後に、再度ボタンを押すとカップラーメンタイマーを作動できます。

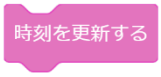
5 章で取り上げた 3 つの作例以外にも、アーテックリンクスの機能を活用すると様々なプログラムを作成することができます。

例）泥棒対策に使える防犯アラーム

自分で考えたオリジナルのプログラムを作成したり、サンプルプログラムを参考にしたりして、プログラミングを楽しみましょう！

使用したブロックの一覧

カテゴリー	ブロック	命令の内容	説明ページ
動き		指定した赤外線を送信します。	P.36
見た目		LEDを指定した色で点灯します。	P.8
		LEDをRGB数値で指定した色に点灯します。	P.21
		LEDを消灯します。	P.8
音		スピーカーから指定した音を鳴らします。	P.29
		スピーカーから指定した音を指定した時間だけ鳴らします。	P.29
		スピーカーから鳴らしている音を停止します。	P.29
		スピーカーから登録した音データを再生します。	P.31
		スピーカーからマイクで録音した音を再生します。	P.116
		マイクで音を録音します。	P.116
		録音を終了します。	P.116

カテゴリー	ブロック	命令の内容	説明ページ
イベント		が押されたとき、次のブロックの命令を実行します。	P.11
IoT		指定したWi-Fiに接続します。	P.86
		Wi-Fiの接続を解除します。	P.86
		入力したURLにあるサーバーから情報を取得します。	P.93
		サーバーから取得した情報を調べます。	P.93
		取得した情報の中で、キーに対応している情報を調べます。	P.96
		入力したURLのタイムサーバーに接続し、タイムゾーンを設定します。	P.87
		タイムサーバーから現在時刻を取得します。	P.88
		タイムサーバーから取得した年・月・日・時・分・秒の情報を調べます。	P.88
制御		指定された時間だけ、次のブロックの命令を実行するのを待ちます。	P.10
		内側のブロックの命令を指定された回数だけ繰り返し実行します。	P.13
		内側のブロックの命令をずっと繰り返し実行します。	P.14

カテゴリー	ブロック	命令の内容	説明ページ
制御		指定された条件が成り立つときだけ、内側のブロックの命令を実行します。	P.19
		指定された条件が成り立つときは上段のブロックの命令を実行し、条件が成り立たないときは下段のブロックの命令を実行します。	P.16
		指定された条件が成り立つまで、内側のブロックの命令を繰り返し実行します。	P.74
		指定された条件が成り立つまで、次のブロックの命令を実行するのを待ちます。	P.46
調べる		メインユニットのボタンが押されたとき、次のブロックの命令を実行します。	P.39
		メインユニットのボタンが押されているかどうかを調べます。	P.17
		記憶している赤外線の信号を調べます。	P.36
		マイクが計測している音量を調べます。	P.41
		計測中のタイマーの時間を調べます。	P.74
		計測中のタイマーの時間を0秒にリセットして、再び計測を開始します。	P.74
演算		左の数字が右の数字より大きいかどうかを比べます。	P.46

カテゴリー	ブロック	命令の内容	説明ページ
演算		左の数字が右の数字より小さいかどうかを比べます。	P.49
		「2つの条件がどちらも成り立つとき」という条件を設定します。	P.74
変数		変数に保存されているデータを調べます。	P.24
		変数に指定したデータを保存します。	P.23
		変数に保存されているデータを、指定された数値ずつ変更します。	P.23
		リストに保存されているデータを調べます。	P.42
		指定されたデータを、リストの最後尾に追加します。	P.43
		リストに保存したデータを、全て消去します。	P.45
		リストの指定した番号にあるデータを、指定されたデータと入れ替えます。	P.44
		リストの「指定した数値」番目に保存されているデータを調べます。	P.43
		リストに保存されているデータの総数を調べます。	P.44
関数		関数にまとめたスクリプトを実行します。	P.56

アーテックリンクス　はじめてのプログラミング　音声で家電を動かそう

2024 年 4 月 1 日　発行

【著者・編集】

株式会社アーテック 「アーテックリンクス　はじめてのプログラミング　音声で家電を動かそう」製作担当

【発行元】

株式会社アーテック

〒581-0066

大阪府八尾市北亀井町 3-2-21

【お問い合わせ】

本書の内容や機材、ソフトウェアに関するお問い合わせは下記宛先までメールにてご連絡ください

株式会社アーテック 「アーテックリンクス　はじめてのプログラミング　音声で家電を動かそう」製作担当

E-mail:support@artec-kk.co.jp

動作環境

●ハードウェア推奨環境

- 下記 OS を搭載した PC・タブレット

Windows(11/10) / macOS 10.15 以上 / ChromeOS 100 以上

- 画面サイズ 10 インチ以上

●アプリの動作環境

- WEB アプリ…Chrome/Edge 89 以上

※全ての端末での動作を保証するものではありません

45709 | K0424