第 27 回つやまロボットコンテスト

走(はし)って！積(つ)んで！

陣取(じんと)りロボコン

小中学生の部

マイクロビット(micro:bit)及(およ)び送受信機(そうじゅしんき)

プログラム説明書(せつめいしょ)

V1.10　　令和 4 年 8 月 17 日

バージョン	修正日付	変更内容など
V1.00	2021/08/30	
V1.10	2022/08/17	１．V2.21 についての記述を追加。 ２．(3) 説明画像の 1 枚目が現在と異なるため更新。 ２．(4) 書込途中でケーブルを抜かないよう注意書き追加。 ２．(7) 説明画像変更。 ６．(2) 左車輪モーターのリード線の色を赤黒入れ替え。

内容

1. マイクロビットについて

マイクロビット（micro:bit）は、イギリスの国営放送（BBC）が皆さんのような小中学生にコンピューターの事を学んでいただくために開発したものです。

税別 2,000 円（2022 年 8 月現在は 2,420 円）のコンピューターでありながら、たくさんの機能を持っています。

更に、2020 年 11 月に同価格で、性能がアップした V2（バージョン 2）が発売されました。

現在は、半導体不足に対応するため、一部の部品を変更した V2.21 になり、価格も少しアップしています。

主な機能は以下の通りです。

・A・B　2 つのスイッチ入力
・G（加速度）センサー（ゆさぶり検出など）
・傾きセンサー（3 方向）
・明るさセンター
・方位センサー
・簡易温度センサー
・5×5 マトリックス LED
・簡易スピーカー（V2 以降のみ）
・簡易マイク（V2 以降のみ）
・タッチセンサー（V2 以降のみ）
・デジタル入力またはデジタル出力またはアナログ出力として利用可能な I/O×16（その内、6 はアナログ入力可能）
・無線通信機能
・シリアル通信（PC とのシリアル通信、拡張）
・USB インターフェイス
・その他の機能を拡張できるカードエッジコネクター

以上のような豊富な機能をパソコン上でブロックを並べていくだけでプログラミングでき、USBケーブルでつなぐだけで専用の書き込み装置などなくプログラムの書き込みができ、書き込んだ後は PC から外して動かすことができます。

本書では、マイクロビットでのプログラミングの仕方、書き込み方法、皆さんへ配布時に予め書き込んであるプログラムの説明を行います。

2. マイクロビットのプログラミング環境

(1) インターネットブラウザー

マイクロビットのプログラミングを行うには開発用プログラムや書き込み用プログラムをインストールする必要はありません。ブラウザーで専用のホームページを開くだけで利用できます。

ブラウザーは以下のものが利用可能です。(Windows、Mac、Linux、または Chrome OS)

- Edge バージョン 14 以降
- Internet Explorer バージョン 10 以降（マイクロソフト社非推奨＝使わない方が良い）
- Chrome バージョン 22 以降
- Firefox バージョン 16 以降
- Safari バージョン 6 以降（Mac）
- Mobile Safari on iOS 6 以降 for iPad, iPhone, iPod Touch

※ 本書では、Windows の Chrome を使った画面を表示しています。

(2) URL

ご利用のブラウザーで以下の URL を開きます。

「 https://makecode.microbit.org/ 」

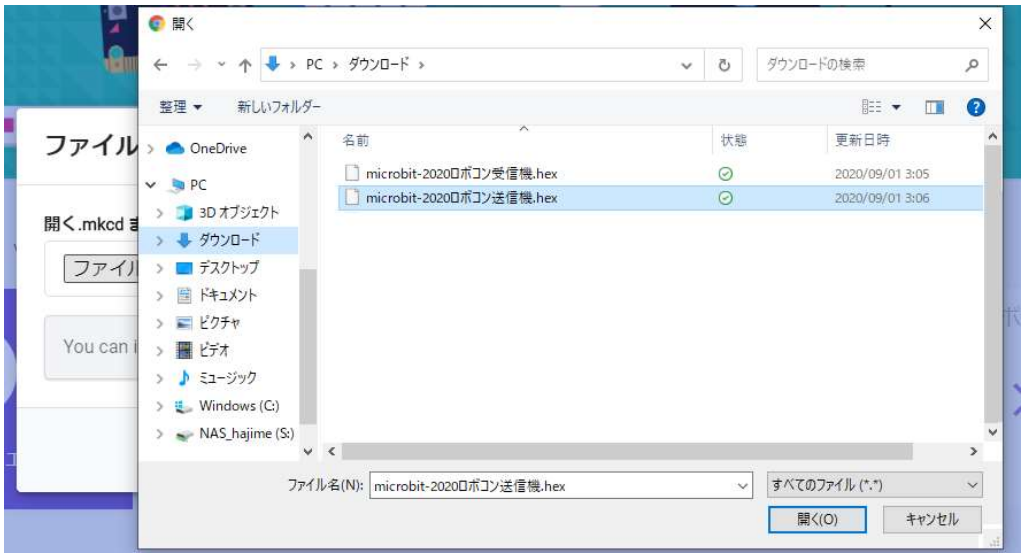
(3) ファイルの読み込み

前ページの画面で「読み込む」ボタンをクリックします。

以下の様な表示になりますので、「ファイルを読み込む...」をクリックします。

「ファイルを選択」をクリックします。

予めダウンロードしておいたプログラムを選んで「開く」をクリックします。

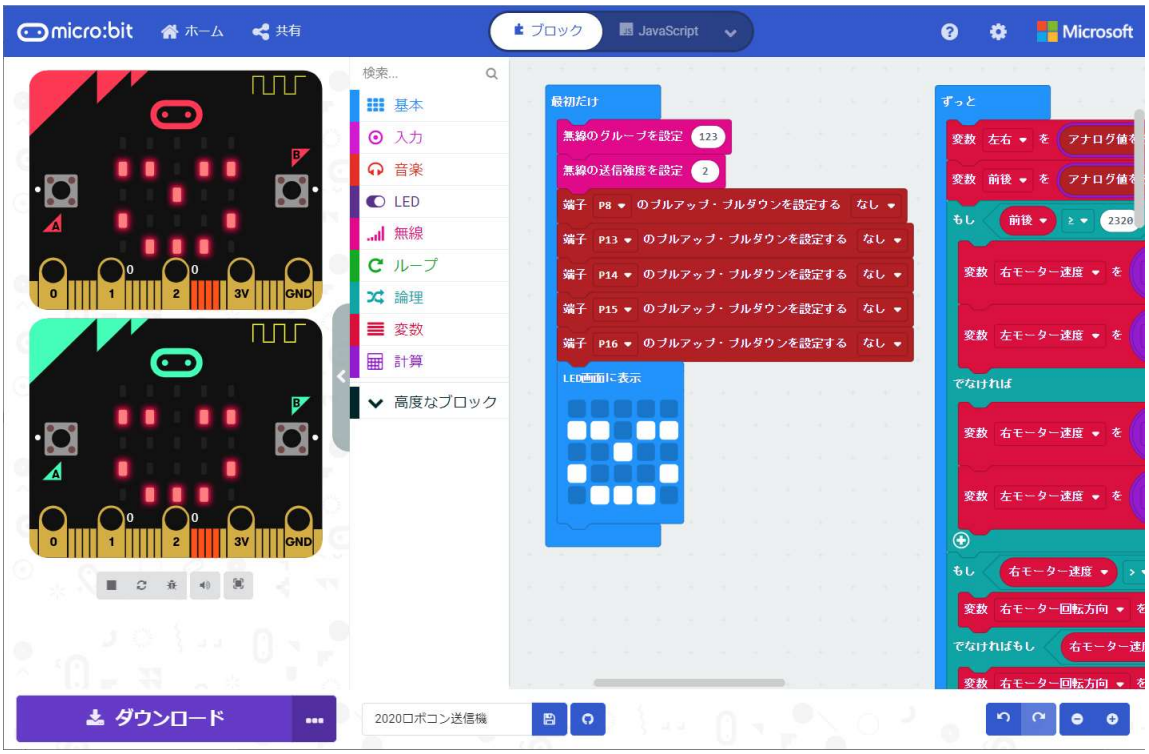


「つづける」をクリックします。

ファイルからの読込が完了すると、以下の様なプログラム編集画面になります。

プログラムは一部しか見えていませんが、右側のコマンドのブロックが無い、グレーの部分をマウスでドラッグすると、他の部分も見ることができます。

また、右下に［(－) (＋)］のようなアイコンがありますが、(－)をクリックするとブロック全体が小さくなって全体を見渡せるようになります。小さくしすぎると文字が読めなくなりますので、詳しく見たいときは(＋)をクリックして、大きくします。



(4) マイクロビットの接続

配布した中にある USB ケーブルの小さい方のコネクター（マイクロ B）をマイクロビットのジャックに差し込みます。台形をしていますので、向きに注意してまっすぐに差し込みます。

次に USB ケーブルの大きい方のコネクター（USB-A）をパソコンに差し込みます。

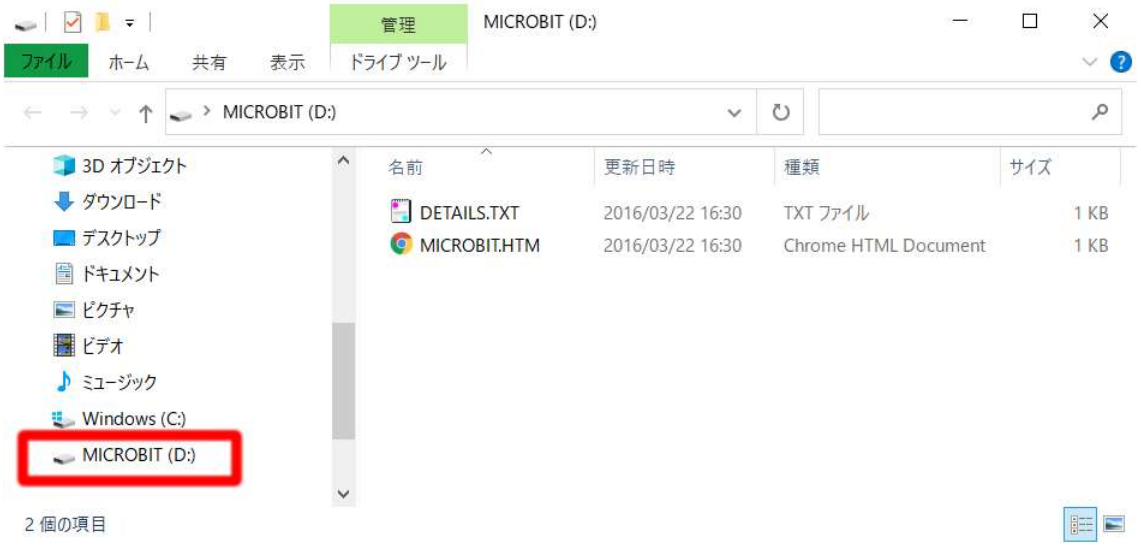
ご自宅のパソコンがネットブックやタブレットなどで、USB-A のポートが無い場合は、変換ケーブルが必要になります。どのようなケーブルを購入したらいいかわからない場合は、最後のページにある問い合わせ先までお問い合わせください。

マイクロビットをパソコンに接続すると、USB メモリーをパソコンに接続したのと同じように、メモリードライブが追加されます。

Windows のパソコンの場合で、もともと C ドライブしかない場合は、D ドライブが追加されます。ドライブ名は「 MICROBIT 」になっているので、D ドライブ以外になっていても、すぐ見つけられると思います。

作成・変更したプログラムをこのメモリードライブにコピーすると、マイクロビットが自動的に読み込んで、勝手に再起動し、プログラムが動き始めます。

このプログラムは、マイクロビットの中のフラッシュメモリーに書き込まれ、電源をオフにしても消えません。（書き込み途中でケーブルを抜くと壊れる可能性があります。）

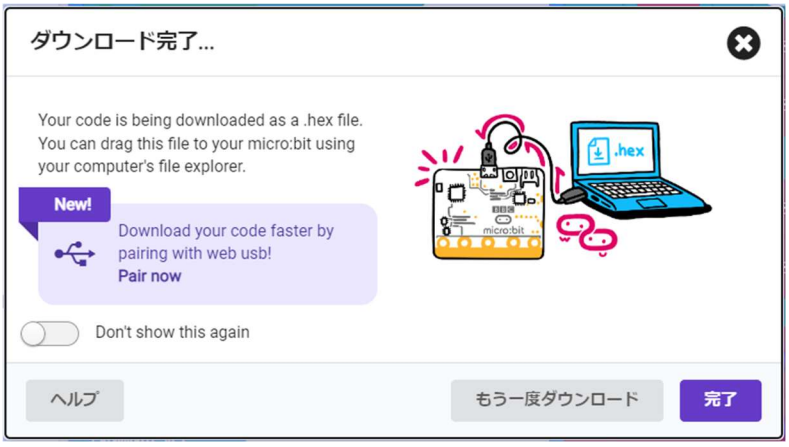


(5) プログラムの書き込み（その１：ダウンロード　＆　コピー）

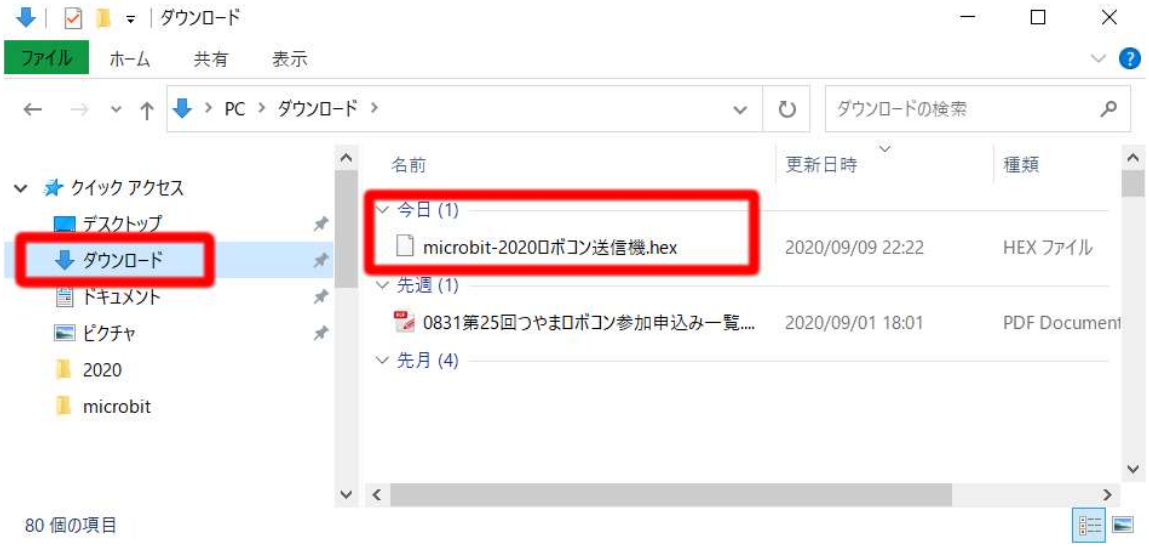
プログラムの編集画面で、左下の「ダウンロード」をクリックします。

以下の様な画面が表示されたら、ダウンロードは終わっています。

Windows パソコンの標準的な状態では、「 ダウンロード 」というところに入っています。



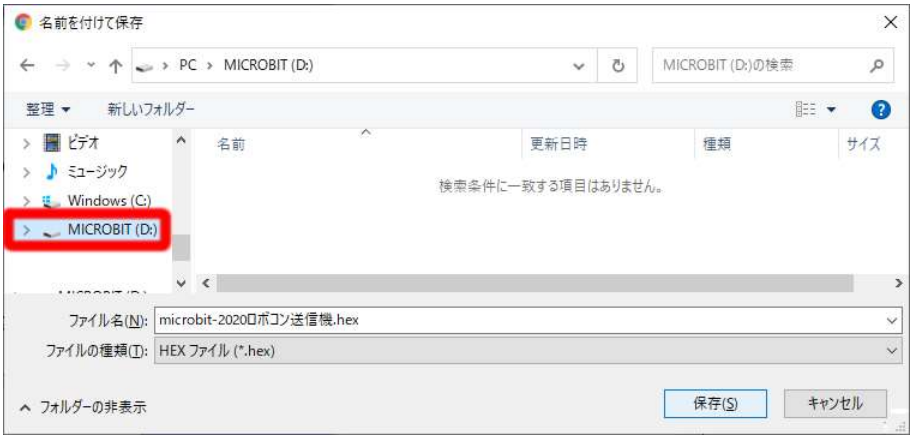
このファイルを「 MICROBIT 」の名前が付いたメモリードライブへコピーします。



(6) プログラムの書き込み（その 2：直接ダウンロード）

ブラウザーの設定で、「ダウンロード時に毎回保存先を選択する」という設定ができます。

この設定がしてあると、「ダウンロード」ボタンを押した時に保存先を聞かれますので、「MICROBIT」の名前のメモリードライブを指定して「保存」ボタンをクリックすれば、マイクロビットにそのまま書き込まれます。

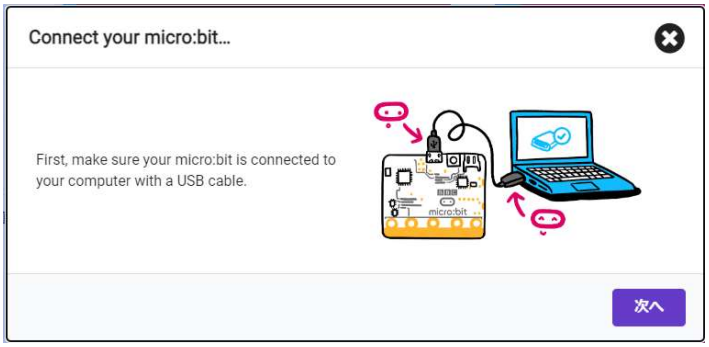


(7) プログラムの書き込み（その３：デバイス接続 & 直接書き込み）

パソコンやブラウザーの種類によってはできない場合がありますが、これが一番便利です。

左下にある「ダウンロード」ボタンの右の「…」をクリックし、その上に表示された「デバイスを接続する」をクリックします。

以下の表示が出たら、「次へ」をクリックします。



以下の表示が出たら、「 “BBC micro:bit ～～” 」という行をクリックし、右下の「接続」をクリックします。

デバイスの接続に成功したら、元の画面に戻ります。

ここまでの準備ができたら、次からは「ダウンロード」ボタンをクリックするだけで、直接マイクロビットにプログラムの書き込みが行われます。

※ 配布したマイクロビットは送信用と受信用で 2 枚ありますが、もう一方のマイクロビットを接続したときは、再度「デバイスの接続」操作が必要です。2 枚ともデバイス接続の設定がされていれば、一旦抜いて差し換えても再設定の必要はありません。

3. 送信機プログラム概要・各部名称

(1) 概要

マイクロビットのプログラムには、「最初だけ」というブロックと「ずっと」というブロック、そしてイベントブロックがあります。送信機では、「最初だけ」と「ずっと」の二つを使っています。

「最初だけ」はその名の通り、マイクロビットの電源が入った時最初だけ動く部分です。

マイクロビットの裏側にあるリセットボタンを押したり、PC からプログラムを書き込み完了したりしたあとも「最初だけ」の部分から動き始めます。

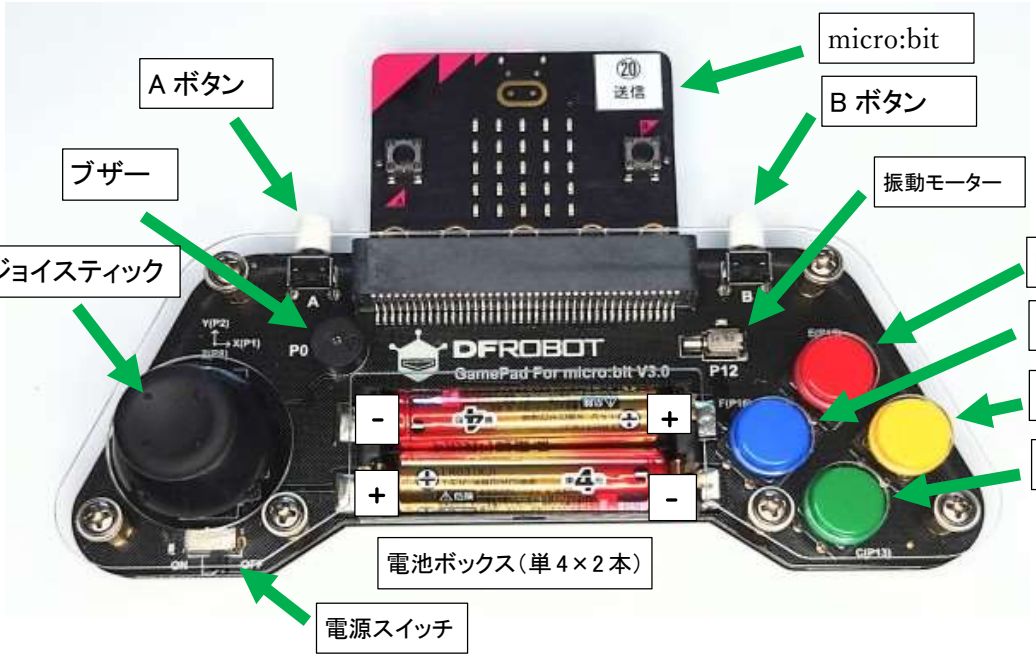
「最初だけ」の中身が最後まで動く（実行される）と、「ずっと」のブロックが動き始めます。

「ずっと」のブロックは最初から最後まで動いた後は、また先頭に戻り、「ずっと」動き続けます。

(2) 送信機各部名称

送信機は以下の様になっています。この写真の様に、リモコンのコネクターにマイクロビットを差し込んで使います。表裏を間違わないように奥まできちんと差し込みましょう。

電源スイッチが「OFF」になっているのを確認してから、単 4 の乾電池 2 本を電池ボックスへ入れます。プラスとマイナスの向きに注意しましょう。



4. 送信機プログラムの「最初だけ」の処理

(1) 全体

ここからは、配布した時点で書き込んである、送信機プログラムの説明をします。

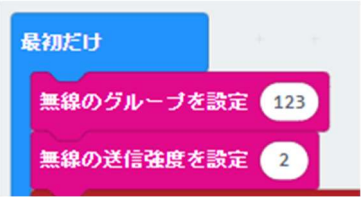
全体のプログラムは以下の様になっています。

以降で、順番に説明致します。

(2) 無線設定

「無線のグループを設定」は送信側と受信側のグループ番号を合わせることでお互いに通信が可能となります。このグループ番号は全参加チームが異なるグループ番号を使わなければ、正常にコントロールすることができません。必ず配布部品と一緒にお配りした紙に書かれた指定のグループ番号をお使いください。

「無線の送信強度を設定」は無線の電波を送信する強さを０～７で設定します。数字が大きいほど遠くまで届きます。あまり強くすると大会会場の待機ブースで調整中のチームと競技中のチームの干渉が強くなり、反応が遅くなる可能性がありますので、設定は「２」としておいてください。



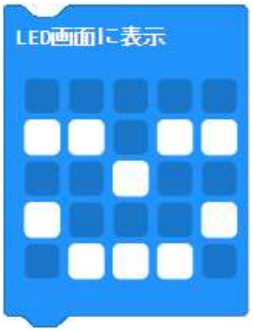
(3) デジタル入力端子への設定

「端子［P8］のプルアップ・プルダウンを設定する［なし］」は送信機用リモコンのスイッチ入力を正常に使うための初期設定です。これらを「なし」にしておかないと、デジタル入力が正常にスイッチのオンオフを判別できなくなる可能性があります。

P8、P13～P16 の 5 つの入力について設定が必要です。

(4) LED 表示

「LED 画面に表示」はマイクロビットの 5×5 のマトリックスになった赤色 LED に表示を行うコマンドです。このブロック状の 5×5 の四角をクリックすると、青と白と交互に切り替わります。この白に設定された位置に対応する LED が光ります。これは自由に変えていただいても構いません。

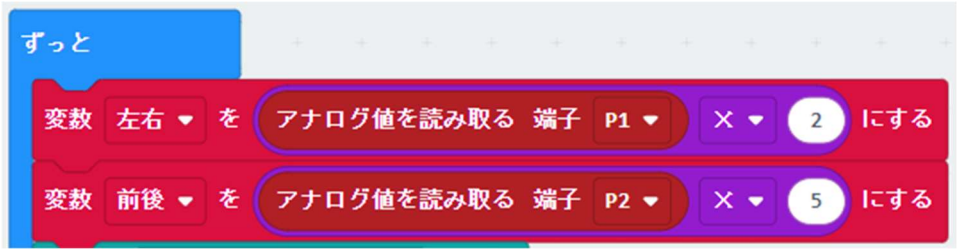


ここまでが動いた後、次の「ずっと」が動き始めます。

5. 送信機プログラムの「ずっと」の処理

(1) まず、最初の処理は以下の様になっています。

こちらをこの後、ブロックごとに分解して説明します。



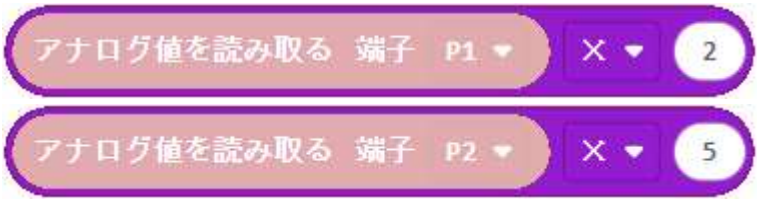
このリモコンでは、左側のジョイスティックを上下・左右に倒すと、どれくらい倒したのかを数字の大きさで入力することができるようになっています。

デジタル入力を使うと、0 か 1 のどちらかの入力となりますが、アナログ入力を使った場合、マイクロビットでは 0～1023 が入力されるようになっています。

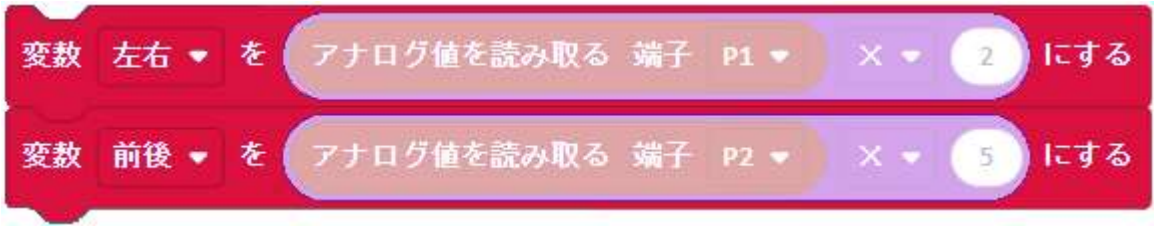
このリモコンのジョイスティックを左右に倒すと P1 端子のアナログ入力が変化します。
左へ倒すと数値は減っていき、右へ倒すと増えていきます。
左いっぱいまで倒すと、0～5 くらいになります。
手を離したときは、500～520 くらいになります。
右いっぱいまで倒すと、1015～1023 くらいになります。

このリモコンのジョイスティックを前後に倒すと P2 端子のアナログ入力が変化します。
後ろ（手前）へ倒すと数値は減っていき、前（向こう側）へ倒すと増えていきます。
手前いっぱいまで倒すと、0～5 くらいになります。
手を離したときは、500～520 くらいになります。
前方向いっぱいまで倒すと、1015～1023 くらいになります。

「アナログ値を読み取る」ブロックの周りにある紫のブロックは計算ブロックです。
ここでは、P1 から読み取った値を 2 倍に、P2 から読み取った値を 5 倍にしています。
P1 の方の掛け率が低いのは、前後に対して左右の倒れ方の影響度を少なくするためです。
このあたりが難しいと感じた場合は、適当に読み飛ばしてください。



「変数［○○］を（ □□ ）にする」というブロックは、変数に値を覚えさせるものです。
変数とは、書いたり消したりできるメモ用紙のようなもので、覚えておいた内容を後で使うためにメモしておきます。
ここでは、「左右」という名前の変数に、P1 から読み取った値の 2 倍をメモ（記憶）し、「前後」という名前の変数に、P2 から読み取った値の 5 倍をメモしています。



(2) モーター速度の計算

次の処理は以下の様になっています。こちらも分解して説明します。

この部分は、計算ブロックを 3 つ重ねて使っています。

式になおすと、「 （ （前後） − （左右） − 1536 ） ÷ 24 」となっています。

これは、右側のモーターの速度を決めるために、先程メモした変数「前後」の値が大きいほど速く、ジョイスティックを右に倒すと右に曲がりたいので、右のモーターを遅くすればいいということで、変数「左右」の値が大きくなるほど、遅くなるように引き算しています。

左右に倒したときに左右のモーターにあまり速度差がつきすぎると、コントロールが難しくなりますので、前後の値に対して、左右の値の影響度が 2/5 になる様、前のブロックで前後は 5 倍、左右は 2 倍したのです。

ジョイスティックは前後左右とも手を離したときにおよそ 512 が入力されます。

その時にモーターの速度を 0 にして停止するために、1536 を引いています。

512 × 5 − 512 × 2 − 1536 = 0

［前後］−［左右］− 1536 = -3582 ～ 3579 に変化します。そのままだと値が大きすぎるので 24 で割って、-149.25 ～ 149.125 にしています。

そして、その外側のブロックで、小数点以下を四捨五入し、整数にしています。

整数とは、1、2、3、… と指折り数えられる数の事です。

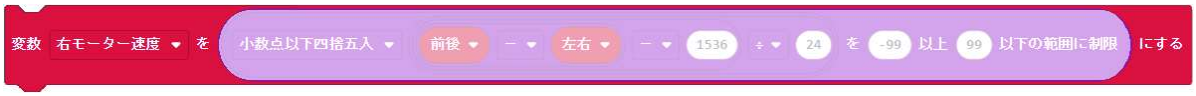
149.25 のような小数が、149 の様に小数点以下が 0 になります。

このブロックによって、ジョイスティックを左右に倒した時の数値の範囲は、-149～149 の範囲となります。

次に、上記の値に下限と上限を設けて、数値の範囲を限定するブロックです。

右モーターの速度を-99～99 の範囲にします。

上記までで計算した、-99～99 の数値を「右モーター速度」という名前の変数に記録します。



左モーターも同様に速度-99～99 の範囲になる様に計算し、「左モーター速度」等言う名前の変数に記録します。

右モーターの時と異なるのは、「前後」に「左右」の値を足しているところと、そこから引く値が異なっているところです。

ジョイスティックを左右に倒した時の左モーターの速度に対する影響は、右モーターとは逆なので、プラスマイナスが逆になっています。

また、ジョイスティックから手を離したときに前後・左右ともにおよそ 512 になりますので、

512 × 5 ＋ 512 × 2 − 3584 ＝ 0　としています。

ここまで説明したのは、以下の上半分の部分です。

緑色のブロックは、「条件分岐」というブロックです。6 角形の中の条件によって、動作が変わります。

ここでは、変数「前後」の数値が「2,320」以上なら、ここまでに説明したブロックが動作し、「2,319」以下なら、右モーターと左モーターの計算式を入れ替えています。

これは、バックしながら曲がろうとしたときに、ジョイスティックを左右に倒したのと逆に曲がってしまうのを避けるためです。

(3) モーター速度を回転方向と絶対値に変換

次にモーター速度（ -99 ～ 99 ）の数値を回転方向と速度の絶対値に変換します。

これは、モーターを動かすためのコマンドが、回転方向と速度の絶対値で指定する仕様のためです。

変数「右モーター速度」が 10 より大きければ、すなわち 11～99 ならば、変数「右モーター回転方向」に文字「F」を記憶させます。「F」は「Forward」（前進）の頭文字です。

また、「右モーター速度」が-10 より小さければ、しなわち-99～-11 ならば、「右モーター回転

方向」に文字「R」を記憶させ、「右モーター速度」を絶対値に変換します。-99～-11 は、11～99 に変換されます。「R」は「Recession」（後退）の頭文字です。

「右モーター速度」が 10 よりも大きくはなく、また-10 より小さくもない時、すなわち-10～10 の範囲であれば、「右モーター回転方向」に「S」を記憶させ、「右モーター速度」に 0 をセットします。「S」は「Stop」（停止）の頭文字です。

ジョイスティックから手を離した状態でもぴったり 512 にはならず、個体毎に多少のばらつきがあるため、停止となる値に幅を持たせています。

左モーターについても同様の処理をします。

(4) ボタンA・ボタンBの入力

ボタンA・ボタンBは元々マイクロビットにあるボタンA・ボタンBと同じところにつながっています。そのため、ボタンが押されているか押されていないかの入力を行い、判別する専用の命令があります。以下の紫色の部分です。

そして、緑色の判断ブロックで、「もし、ボタン［A］が押されているなら、変数［ボタンA］を「1」にしています。ここで注意するのは、あとで都合がいいように、数値1ではなく、文字「1」にしています。そして、押されていなければ、変数［ボタンA］に文字「0」を記憶させています。

ボタンBについても同様にしています。

(5) ボタンCの入力

ジョイスティックの入力には、アナログ入力というコマンドを使い、ボタンA・ボタンBの入力には、専用のコマンドを使いましたが、ボタンC、ボタンD、ボタンE・ボタンFの入力には「デジタル入力」を使います。

デジタル入力を使うと、その入力端子の電圧がある電圧(A)より高いと1が、ある電圧(B)より低いと0が入力されます。電圧(A)と電圧(B)は同じではありません。電源の電圧やその他の条件によってこれらの電圧は変わります。入力端子が、もし電圧(A)と電圧(B)の間の電圧であった場合は、0が入力されるか1が入力されるかわかりません。電子回路では、そのような状態にならないように設計されています。

以下の茶色の「デジタルで読み取る 端子［P13］」というブロックがデジタル入力のコマンドです。［P13］の部分は、P0～P16まで17種類に変更ができます。

今、説明しているリモコン送信機では、ボタンCの回路がP13に接続されていますので、P13を選択しています。

ボタンCを押していない時、P13の入力は「1」となり、押した時は「0」となります。

また、ボタンA・ボタンBの時と同様に文字列にしておいた方が後で都合が良いので、P13の入力が0と等しければ（＝イコールならば)、変数「ボタンC」に文字「1」を記憶させ、P13の入力が1ならば（0と等しくなければ)、変数「ボタンC」に文字「0」を記憶させます。

(6) ボタンD・ボタンE・ボタンFの入力

これらのボタンもボタンCと同様に接続されており、それぞれ

ボタンD → P14

ボタンE → P15

ボタンF → P16

となっています。それぞれボタンと同じ名前の変数に、ボタンが押されていたら「1」を押されていなければ「0」を記憶させます。

(7) モーター速度の文字列への変換

モーターの速度については、(3)のところで数値のままでしたので、この後の無線送信処理のために文字列に変換します。

以下の処理ですが、少しややこしいので、分解して説明します。

ここでやりたいのは、変数「右モーター速度」0～99 の数値を文字列に変換することですが、無線で送信して受信側の処理の都合上、桁数を 2 桁固定にしたいともいます。数値 0 を文字列に変換すると、「0」という 1 文字になってしまい、都合が悪いので、まず「右モーター速度」に 100 を足します。値は 100～199 の範囲になります。

上記で求めた数値を文字列に変換します。3 文字の数字文字列になります。

次は、文字列の一部を切り取るコマンドブロックです。

ちょっと変に感じるかもしれませんが、マイクロビットの Makecode では、文字列の最初の文字は 0 番目ということになっています。

0 番目	1 番目	2 番目
1	2	3

ですので、「1 番目から 2 文字」を取ると、上記の表の例では、「23」となります。

「右モーター速度」が 0 の場合、100 を足して 100、文字列に変換して「1 番目から 2 文字」を取ると、「00」になります。

これらのコマンドで、モーター速度を表す文字列は、00～99 となり、必ず 2 文字になります。

こうしてできた 2 文字の数字を変数「右モーター速度文字列」に記憶させます。

左モーター速度についても同様に 2 桁の数字に変換します。

(8) 文字列をつなげる（くっつけて並べる）

送信機リモコンのジョイスティックや各ボタンを押しているかいないかという情報を、無線で受信機側に送信しますが、送信機側と受信機側でどういうデータを送ったらどういう動きをするのかということを決めておき、その決まりどおりのデータを送らなければなりません。

その決まりはこうしなければならないということはなくて、人それぞれいろいろなやり方がありますが、今回皆さんへの配布時に入っているプログラムでは、以下の様な決まりにしています。

上の段の 0～11 の数字は、0 番目から 11 番目までを表しています。全部で 12 文字です。

<table>
<tr><td>0 番目</td><td colspan="2">1</td><td>2</td><td>3</td><td colspan="2">4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td></tr>
<tr><td rowspan="2">右モーター
回転方向</td><td colspan="2" rowspan="2">右モーター
速度
（2 桁）</td><td rowspan="2">左モーター
回転方向</td><td colspan="2" rowspan="2">左モーター
速度
（2 桁）</td><td colspan="2">A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td></tr>
<tr><td colspan="6">各ボタン状態
（0:押していない、1:押している）</td></tr>
</table>

例えば、ジョイスティックを離していて、ボタン A とボタン E を押していた場合、

「 S00S00100010 」となります。

また、ジョイスティックをまっすぐ上に倒して、ボタン D を押していた場合、

「 F99F99000100 」となります。

このような文字列を作り出すのが、以下の「文字をつなげる」コマンドブロックです。

上から順に変数の中の文字列をつなげていき、つながった 12 文字を変数「送信用」に記憶させます

ここまでで回転方向やモーターの速度、ボタンの押している状態をすべて文字列にしていたのは、このためです。

(9) 無線送信

先ほど作成した、12 文字の文字列（変数「送信」の中身）を無線で送信します。

(10)鼻の点滅

無線を 1 回送信したという目印で 4 . (4)で表示したにっこりマークの花に相当する部分の LED を反転させます。1 回送信すると消え、もう 1 回送信すると光ります。

(11)時間をあける

ここまでで、ジョイスティックと各ボタンの入力処理、及び文字列に変換しての無線送信処理はすべて終わりです。「ずっと」の処理が終わると、また「ずっと」の先頭から繰り返しますが、あまり続けて無線を送信すると、受信機側で処理が追い付かなくなってうまく動かない場合がありますので、少しだけ時間を空けています。ここでは、0.1 秒＝100 ミリ秒を待って（一時停止して）います。

どうも動きがおかしいなと思うときは、200 ミリ秒に増やして試してみると良いかもしれません。

(12)改造

リモコン送信機は基本的にはこのまま使えますが、一つだけ改造するとすれば、ジョイスティックのレバーを下に押し込んだり、離したりしたことを P8 という入力端子で調べることができます。子の入力を使ってみたい場合は、プログラムの改造にチャレンジしてみましょう。

6. 受信機各部名称・接続方法

(1) 各部名称

マイクロビット用コネクターには、受信機用のマイクロビットを差し込んで使います。

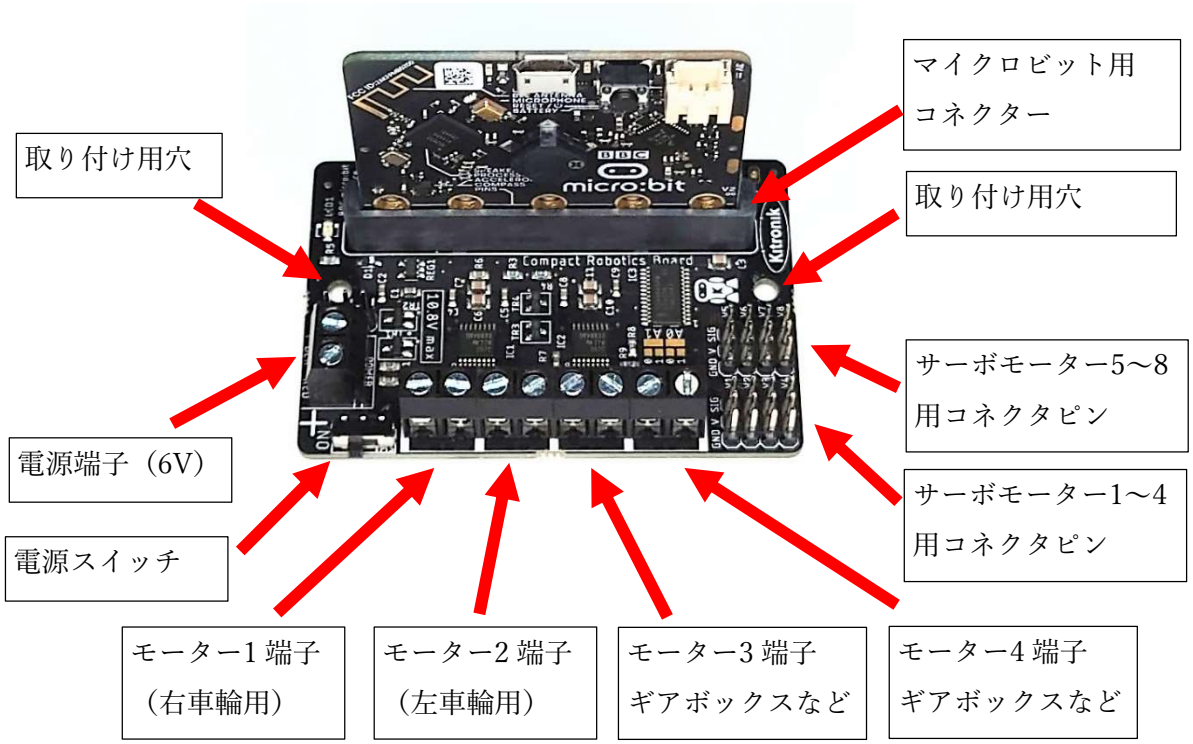
5×5 の LED が付いている側が、モーター1～4 の端子とは反対側になるように差し込みます。

表裏を間違わないように奥まできちんと差し込みましょう。

電源とモーターの端子は、小さめのマイナスドライバーでネジを緩めてから、リード線を差し込んで抜けないようにねじを締めます。あまり強く締めすぎると、リード線が切れたり、端子が壊れたりしますので、適度な力で締めましょう。

2 つの穴に配布した中にある、白色の樹脂製六角スペーサーをねじ止めします。

スペーサーの反対側をロボット本体にねじ止めするか、接着剤などを利用してください。

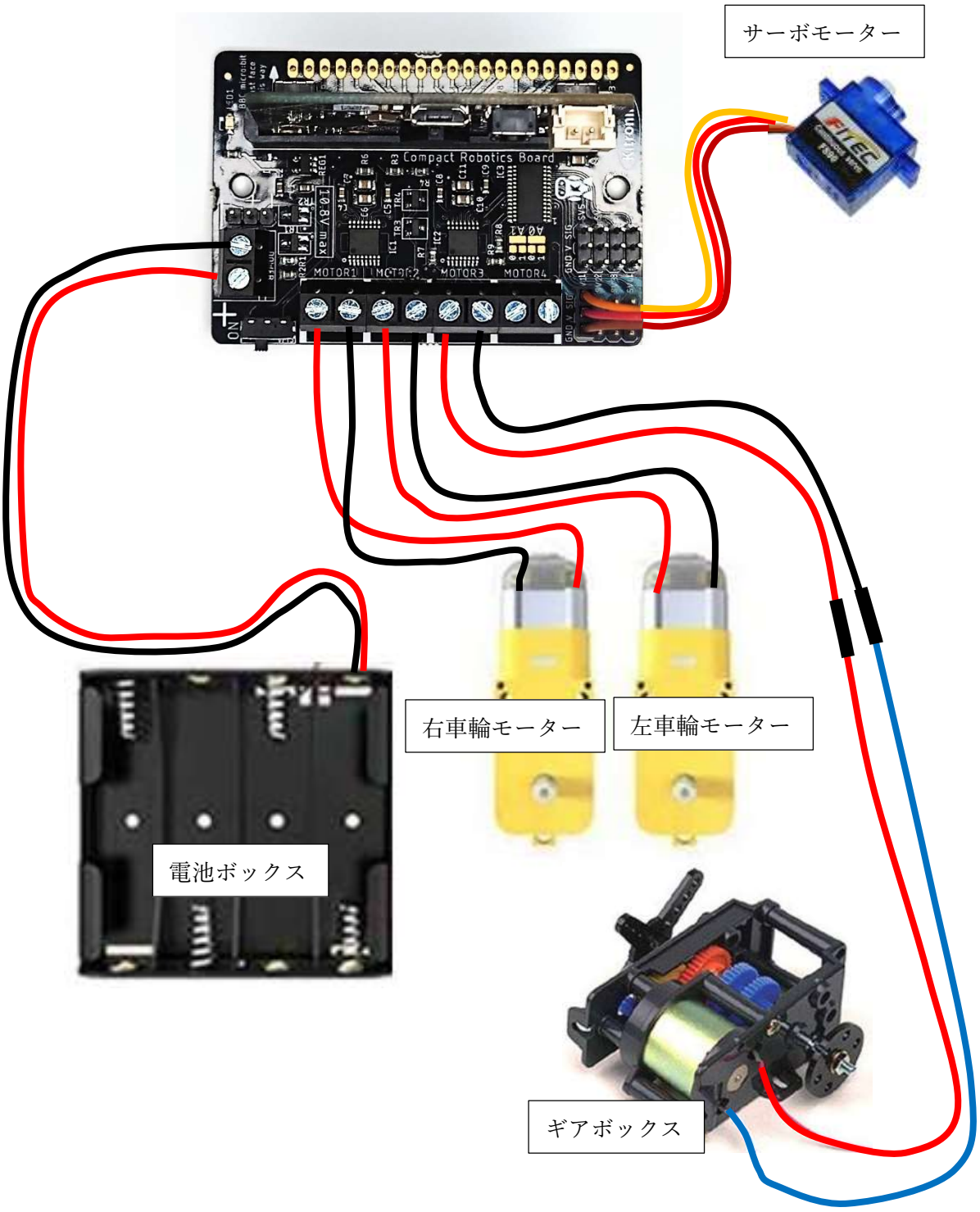


(2) 接続

以下の様に接続します。

電源スイッチの部分は半田付けが必要です。初めて半田付けを行う場合は、必ず経験者の方に指導していただくか、半田付けをお願いしましょう。

ギアボックスやサーボモーターも取り付け位置によってはリード線の延長が必要と思いますので、延長線をつなぐ部分も半田付けをして、絶縁しましょう。



7. 受信機プログラムの「最初だけ」の処理

受信機プログラムの「最初だけ」について、順に説明します。

(1) 全体

全体は以下の様になっています。

以降で順に説明します。

(2) 無線設定

4．(2)の送信機の無線設定と全く同じです。同じにしておかないとお互いに通信はできません。

(3) すべてのモーターの停止

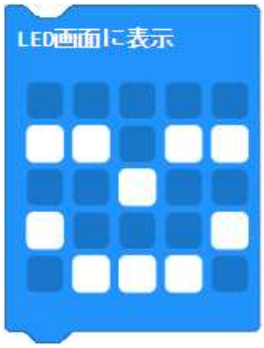
このコマンドブロックは、今回使っているモータードライバー基板のための専用コマンドブロックの中のひとつです。

プログラム開始時に勝手にモーターが動いてしまわないように、すべてのモーターをオフにするというコマンドです。

すべての（all）出力（outputs）をオフにする（turn off）

(4) LED 表示

4．(4)と全く同じです。この表示は自由に変えても構いません。



ここまでが動いたら、受信機の「ずっと」の処理に移ります。

と、言いたいところですが、受信機のプログラムには「ずっと」の処理がありません。

無いのですが、実際にはあって、中身が空なので、何もしないで「ずっと」ぐるぐる回っています。↓ こんな感じ

受信機では、「ずっと」の処理がない代わりに、ボタンを押した時、無線で文字列を受信した時（送信機から文字列が送られてきた時）だけ動くというのがあります。

以下で説明します。

8. 受信機プログラムの「ボタン A が押されたとき」の処理

送信機では、「ずっと」のループの中で、「ボタン A が押されているかな？」というのを調べて、押されているかいないかで処理を分けていましたが、以下の様に「ボタン A が押されたとき」というブロックを使うと、ボタン A が押されたときに自動的に処理を呼び出してくれます。

送信機では、ジョイスティックやたくさんのボタンの入力を調べて、まとめて受信機へ送信するということをしていたため、一つ一つ入力して調べる方が都合が良かったのですが、受信機では「ボタン A が押されたとき」、「ボタン B が押されたとき」、「無線で受信したとき」という風に、何かが起こったときに処理を動かすという方が便利です。

ここでは、ボタン A とボタン B をモーターのテストに使っています。

以下のブロックを分解して説明します。

(1) 「ボタン A が押されたとき」ブロック

ボタン A が押されると（実は押して、離したとき）、このブロックが動きます。ブロックの中に挟んだコマンドが順番に動いて、最後までいくと終わってまた待ち状態になります。

(2) 「モーター回転指示」ブロック

モーター（Motor）の［1］番を前（Forward）方向（direction）へ、スピード（speed）を 50 で回転させます。

モーター番号は 1～4 まで選べます。それぞれ、6．(1)のモーター1 からモーター4 に対応しています。

回転方向は前方向（Forward）と、逆方向（Reverse）のどちらか、スピードは 0～100 が選べます。

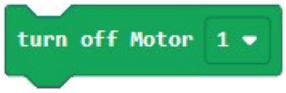
このコマンドを動かすと、モーターは止めるまで、あるいは違うスピードのコマンドを動かすまで、同じスピードで回り続けます。

(3) 「一時停止（ミリ秒)」ブロック

受信機の「ずっと」のループの最後にも同じブロックが出てきましたが、ここではテストのためにモーターを 1 秒間回転させて、1 秒（＝1,000 ミリ秒）後に止めます。

(4) 「モーター停止指示」ブロック

モーター1 を停止（turn off）させます。



(5) ここまでを合わせて

ここまでを合わせて説明すると、ボタン A を押して離したとき、モーター1 を 1 秒間だけ前方向に回して、1 秒後に停める。ということです。

モーター1 は右モーターにつなぎます。

組み立てたら、まず送信機を使わずにボタン A を押してみて、左モーターが回ったり、右モーターが逆転（バックする方に回る）したり、まったく動かなかったりしたときは、配線をチェックします。

9. 受信機プログラムの「ボタン B が押されたとき」の処理

ボタン B が押されたときもほぼ同じ処理です。

モーター2 を前方向に回転させ、1 秒後に停止させます。

左モーターが前方向にちゃんと回るのを確認するためのものです。

10. 受信機プログラムの「無線で受信した時」の処理

無線で文字列を受信したときに動く処理です。大きすぎるので全体図は省略しますが、分解して順に説明します。

(1) 「無線で受信したとき」

このコマンドは正確には、「無線で文字列を受信したとき」というコマンドです。

マイクロビットには、他にも似たような、「無線で数値を受信したとき」、「無線で名前と値を受信したとき」というブロックがありますが、今回はジョイスティックや複数のボタンの状態を一度にまとめて送りたかったので、文字列送信を使っています。

無線で受け取った文字列は、同時に変数名が「receivedString」（受信された文字列）という変数に入れられ、このブロックの中の処理で参照できます。

(2) 右モーター回転方向の取得

送信機から送信された文字列は、5．(8)で説明したとおり、12 文字の文字列で、最初の文字が右モーターの回転方向を表す 1 文字、そして文字の番号は先頭が 0 番目なので、5．(7)でも使った、「文字列の一部を切り取る」コマンドで、変数「receivedString」の先頭の 1 文字を取り出して、変数「右モーター回転方向」へ記憶させます。「F」または「R」または「S」です。

(3) 右モーター速度の取得

右モーターの速度は、1 番目と 2 番目の 2 文字でしたので、2 文字を取り出します。

モーターのスピードは数値で指定しますので、文字列から数値に変換します。

モーターのスピードは 0～100 で指定しますが、受信したスピードの文字列は 00～99 でした。0 はそもそも停止なので、1 を足して 1～100 にします。

こうしてできた 1～99 の数値を変数「右モーター速度」に記憶させます。

(4) 左モーター回転方向の取得

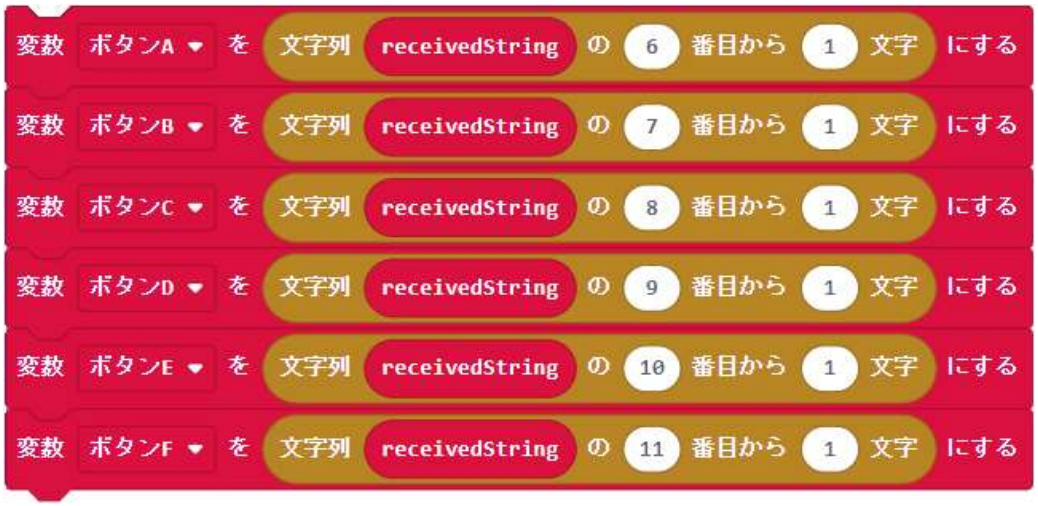
右モーター回転方向と同様に（0 番目から数えて）3 番目の文字を 1 文字取り出して、変数「左モーター回転方向」に記憶させます。

(5) 左モーター速度

右モーター速度と同様に 4 番目から 2 文字を取り出して、数値に変換し、1 を足してから変数「左モーター速度」に記憶させます。

(6) ボタン A～ボタン F の状態の取得

順番に並んだ、各ボタンの状態を表す文字を 1 文字ずつ取り出して、「ボタン A」～「ボタン F」の変数に記憶させます。



(7) 右モーターの処理

皆さんも段々慣れてきたと思いますので、ここでは、右モーターの処理を一気に説明します。

変数「右モーター回転方向」の中身が、文字「F」だったなら、モーター1 を前方向に、変数「右モーター速度」のスピードで回転させます。

文字「F」ではなくて、文字「R」だったなら、モーター1 を逆方向に、変数「右モーター速度」のスピードで回転させます。

文字「F」でも、文字「R」でもなかったなら、モーター1 を停止させます。

このように「もし～なら」ブロックは、「でなければもし」を続けて使うことができます。「でなければもし」は 2 つでも 3 つでも、10 個でも使えます。

ここを見てわかるように、モーターを停止させるのは、「S」でなくても「F」と「R」以外なら何が来ても停止します。ただ、わかりやすくするために Stop の「S」を使っています。

(8) 左モーターの処理

左モーターについても同じように処理します。

変数「左モーター回転方向」の中身が文字「F」だったなら、モーター2 を前方向に、スピード「左モーター速度」で回転させます。

変数「左モーター回転方向」の中身が文字「R」だったなら、モーター2 を逆方向に、スピード「左モーター速度」で回転させます。

「F」でも「R」でもなければ、モーター2 を停止させます。

(9) ボタン A、ボタン B の処理

ここから以降は、あくまでもサンプルですので、自由に作り変えても構いません。

変数「ボタン A」が文字「0」以外なら（つまり「1」であれば)、サーボ（Sevo）モーター1 を 0 度にします。(0 度の説明は後ほどします。)

変数「ボタン A」が文字「0」で、なおかつ変数「ボタン B」が文字「0」でなければ、サーボモーター1 を 180 度にします。

車輪を動かすモーターは電圧をかける（電池をつなぐ）と、回り続けます。プラスとマイナスを反対にすると、逆方向に回り続けます。電圧をかけない（電池を外す）と止まります。

これに対して、サーボモーターというのは、0～180 度の角度を指定すると、その角度に対応した位置まで回転し、自動的に止まります。

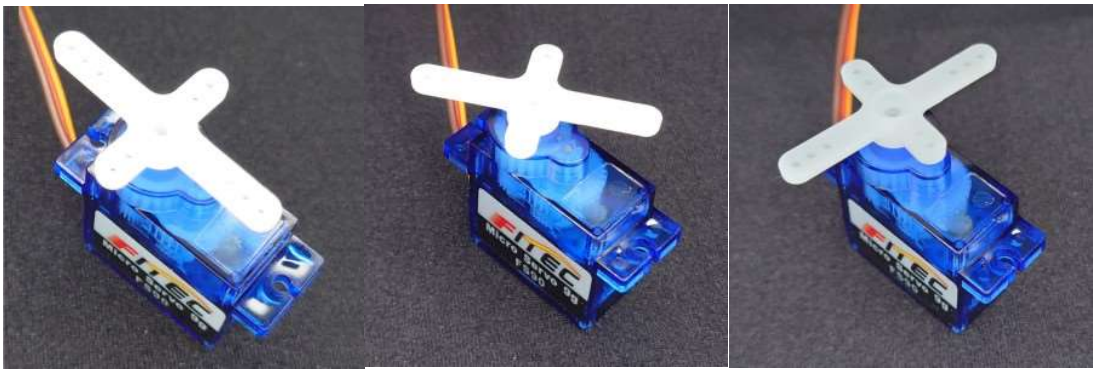
大きい角度で止まっているときに小さい角度を指定すると、自動的に逆方向に回転して角度に対応した位置まで回転すると止まります。

この指定する角度というのは、実際のサーボモーターの軸の回る角度とは異なります。

製品それぞれに多少のばらつきがありますが、私の手元にあるサーボモーターを試してみると、0～100 度付近までは動きますが、100 度近くで動かなくなり、110 度を指定しても 180 度を指定しても動かなくなります。この特徴を理解した上で使用してください。

使い方としては、発射台の向きを変えるとか、引き金を引くとか、手首を曲げたり伸ばしたりするなどが考えられます。

【動作例】　0 度　　45 度　　90 度



(10)ボタン C、ボタン E の処理

ボタン E（赤色)、ボタン C（緑色）のボタンを押すと、モーター3 を正方向、逆方向に回転さ

せることにしています。

変数「ボタン E」が文字「0」以外であれば、モーター3 を正方向にスピード 50 で回転させます。

変数「ボタン E」が文字「0」で、かつ変数「ボタン C」が文字「0」以外であれば、モーター3 を逆方向にスピード 50 で回転させます。

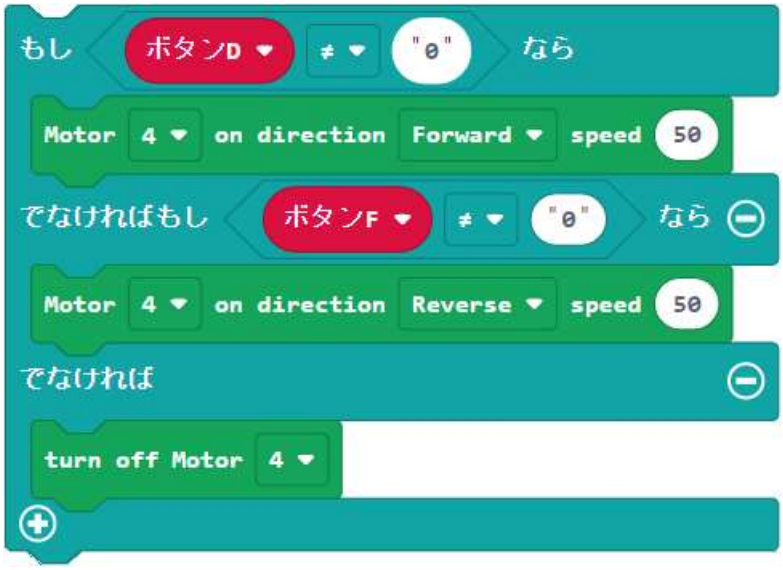
変数「ボタン E」も「ボタン C」もどちらも「0」であれば、モーター3 を停止させます。

モーターのスピードは 50 にしていますが、配布したギアボックスセットのモーターは 3V 用で、モータードライバーの電源は 1.5V の乾電池 4 本で合計 6V のため、スピード 100 で回し続けるのはモーターによくありません。スピード 50 であれば、およそ 3V の電圧がモーターにかかりますので、大丈夫です。速く回したい場合は、スピードを 50 より大きくするのではなく、ギアボックスのギア比を変えましょう。

(11)ボタン C、ボタン E の処理

ボタン C とボタン E の処理も同様です。ここでは、モーター4 の動作を割り当てています。

4 つ目のモーターは配布しておりませんので、別途追加するなどしてください。



(12)LED の点滅

これは無くてはならない処理ではありません。

送信機と同様、5×5 の LED の真ん中を無線から受信するたびに点滅させます。

送信機は、電源を入れると必ず点滅しますが、受信機は送信機の電源が入っていなかったり、無線グループ番号が間違っていたりすると点滅しません。

送信機の電源を入れて、点滅することを確認できれば無線通信が正常にできていることになります。

逆に、送信機の電源を切っても点滅していれば、他に同じグループ番号を使っている送信機が近くにあり、混信している可能性があることになります。

11. プログラム変更時の注意事項

(1) バックアップ

公開されているプログラムをダウンロードしたら、マイクロビット開発環境「メイクコード」に読み込ませてすぐに無線グループを配布した書類に書かれた番号に変更しましょう。

無線グループ番号だけ変更したら、他の部分を変更する前にバックアップを取っておきましょう。

(2) 5×5LED の表示について

5×5LED の表示には、「数を表示」、「文字列を表示」などのコマンドがありますが、これらを使用すると、指定した全桁を表示するために右から左へ文字が流れるように動きながら表示されます。そして、その表示中は次の命令に行かないため、ロボットが素早い動きができなくなってしまいます。注意しましょう。

「最初だけ」の部分に入れるのは、問題ありません。

12. お問い合わせ先

本書に記載したプログラムに関することや、ギアボックス・サーボモーターなどに関すること、はんだ付けに関すること、その他何でもご質問いただければお答えいたします。

普段は会社員のため、回答にはお時間をいただく場合もありますのでその点はご了承ください。

小学生の場合は、先生またはご両親などを通してご質問ください。

担当：　ザ・チャレンジ実行委員会　井上
電話：　090-8246-8288
メール：　hit@nomra.com
LINE：　右の QR コードから
Facebook メッセージ：　右の QR コードから
「メッセージ」をクリック

LINE

Facebook