

クラシックサイズマイクロマウス  
**Pi:Co Classic 3 操作説明書**  
『本体組立完成后 まとめ』

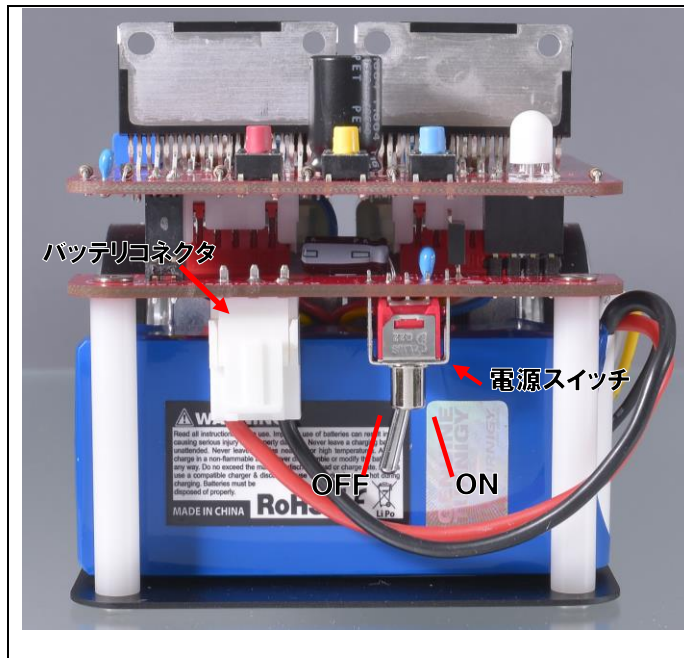
## 目次

1 各部位の説明 .....	3
1-1 電源スイッチ .....	3
1-2 ファームウェア書込みスイッチ .....	3
1-3 モード選択スイッチ .....	4
1-4 モード表示用 LED .....	4
1-4-1 実行モード .....	4
1-4-2 調整モード .....	5
1-5 その他 .....	7
1-5-1 バッテリを取り付ける .....	8
2 プログラムを書き込む .....	9
3 実際に迷路を走らせる為の調整 .....	11
3-1 ターミナルエミュレータの準備 .....	11
3-2 センサ調整 .....	12
3-2-1 センサの角度調整 .....	12
3-2-2 センサ用パラメータの設定 .....	14
3-3 物理的な調整 .....	18
3-3-1 ゲイン数について .....	18
3-3-2 走行距離の調整(タイヤ直径チェック) .....	19
3-3-3 旋回の角度調整(トレッド幅チェック) .....	21
4 迷路を走らせてみる .....	23
5 その他 .....	24
5-1 本体を置く位置 .....	24
5-2 ゴール地点の数え方 .....	24
5-3 タイヤのごみを取る .....	25

## 1 各部位の説明

Pi:Co Classic3 本体（以下本体）の各部の名称と役割、使用方法は次のようになります。

### 1-1 電源スイッチ

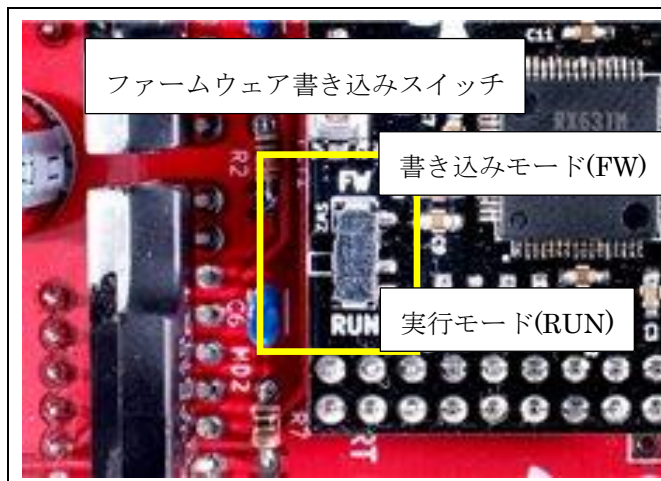


本体の電源 ON/OFF 用スイッチです。

●左に入れると OFF、右に入れると ON になります。

●バッテリーをつなげてから操作してください。

### 1-2 ファームウェア書き込みスイッチ



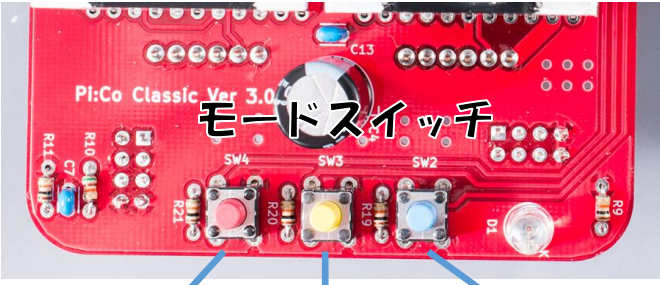
ビルドしたプログラムをマイコンに書き込むモードにしたり、実行モードにしたりするスイッチです。

●プログラムを書き込むときは書き込みモード（FW）に入れます。

●調整や、走らせる時は実行モード（RUN）に入れます。

書き込みの手順については「2 プログラムを書き込む」を参考にしてください。

## 1-3 モード選択スイッチ

 <p>モード選択スイッチ (送り)</p> <p>モード選択スイッチ (戻り)</p> <p>モード決定スイッチ</p>	<p>モードを選択するスイッチです。</p> <ul style="list-style-type: none"><li>●モード選択スイッチ(送り) を1つ押すとモードが1つ進みます。</li><li>●戻りたい時はモード選択スイッチ(戻り)を押します。</li><li>●実行するモードが決まったらモード決定スイッチを押し、実行します。</li><li>●モードには実行モードと調整モードがあります。(4)モード表示用LEDに点灯時の様子を記載しますので参考にしてください。</li></ul>
--	---

## 1-4 モード表示用 LED

ビット1～4のLEDでモードを表示します。

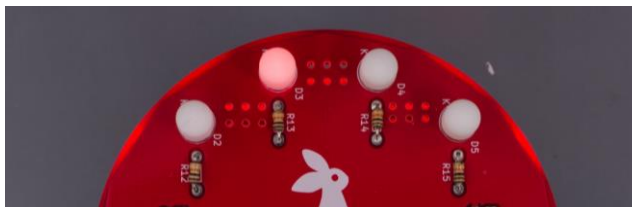
表示できるのは実行モード1～15、調整モード1～7です。

### 1-4-1 実行モード

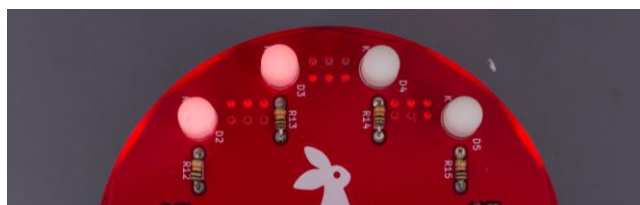
モード1



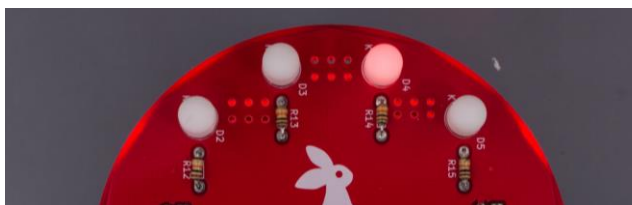
モード2



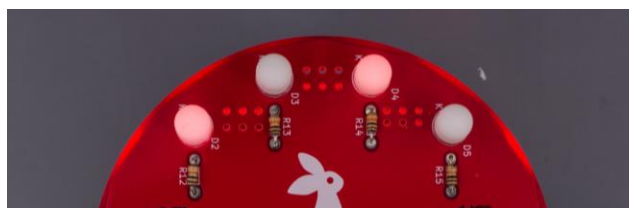
モード3



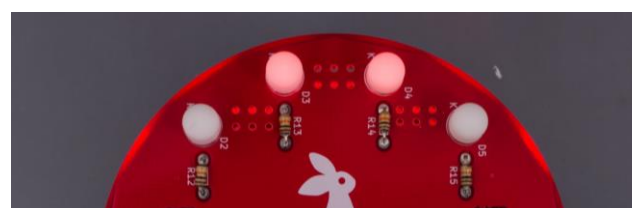
モード4



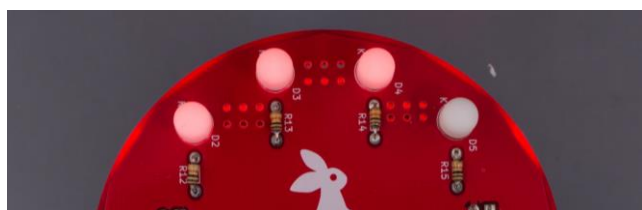
モード5



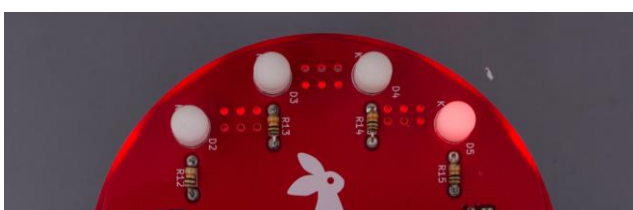
モード6



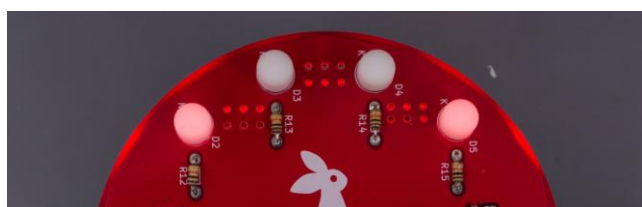
モード7



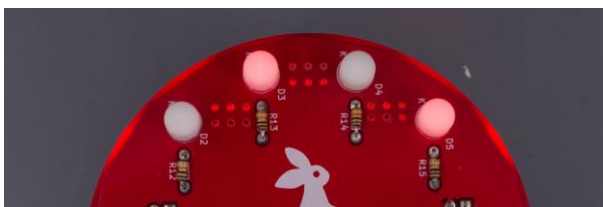
モード8



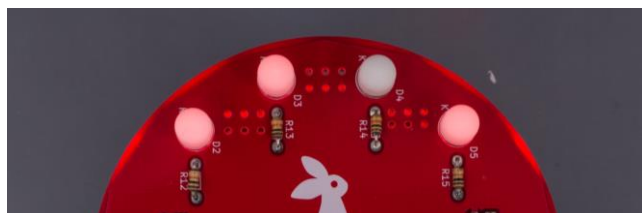
モード9



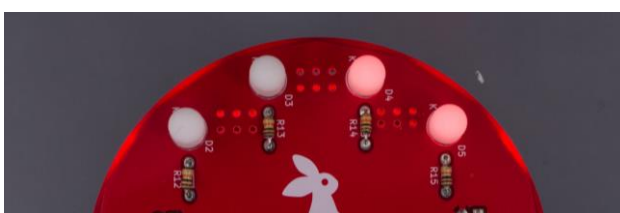
モード10



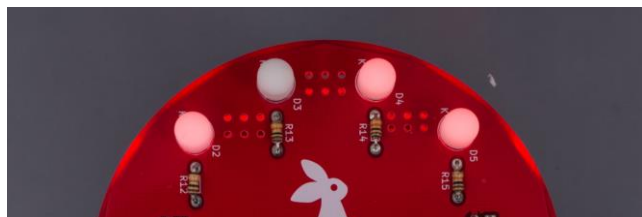
モード11



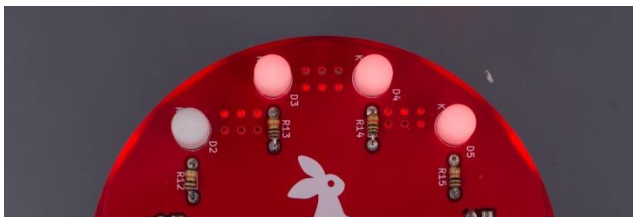
モード12



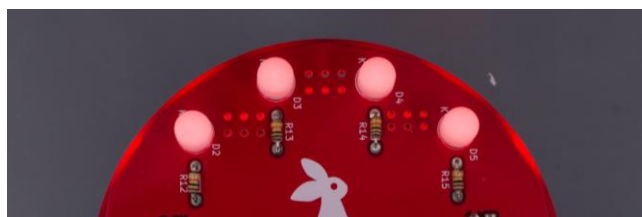
モード13



モード14

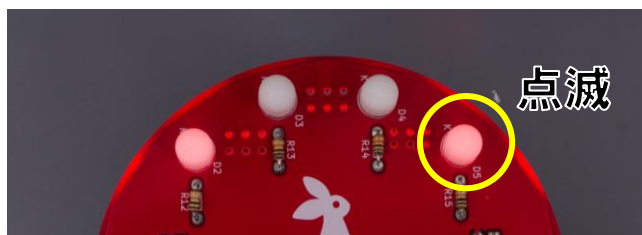


モード15

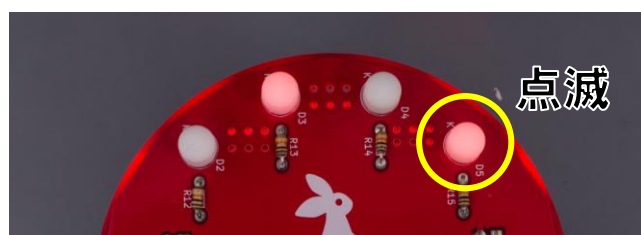


## 1-4-2 調整モード

モード1



モード2



モード3



モード4



モード5



モード6

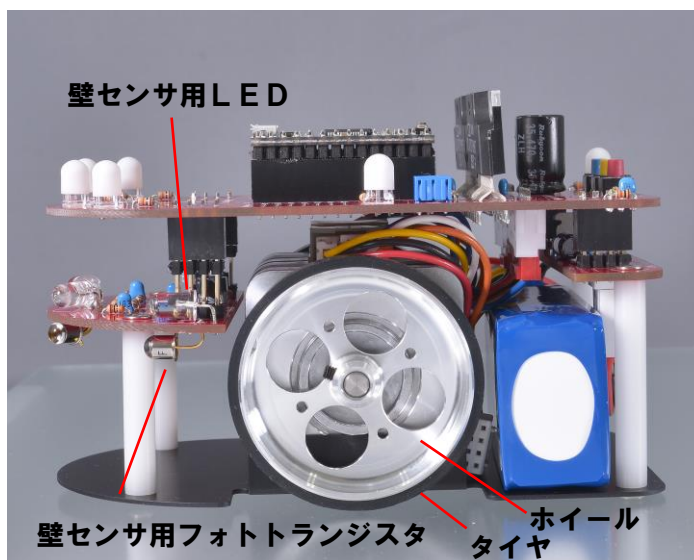
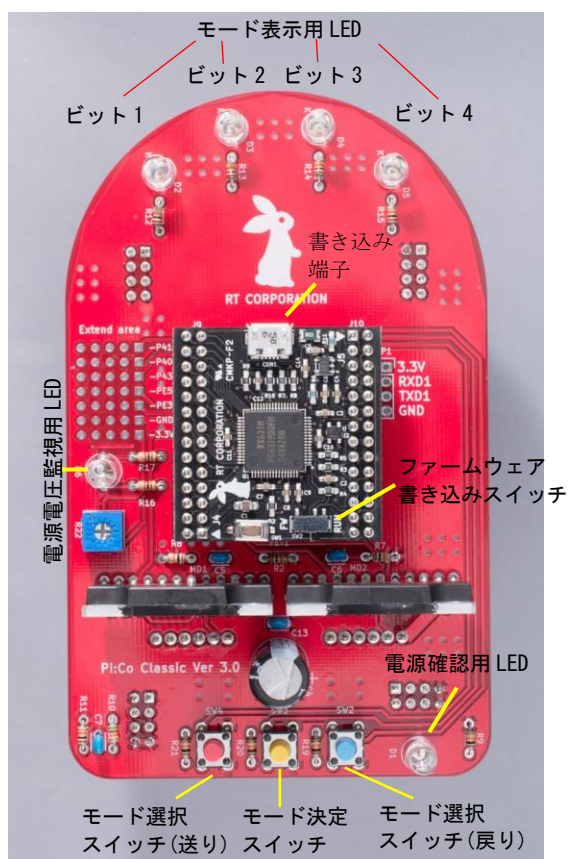



モード7



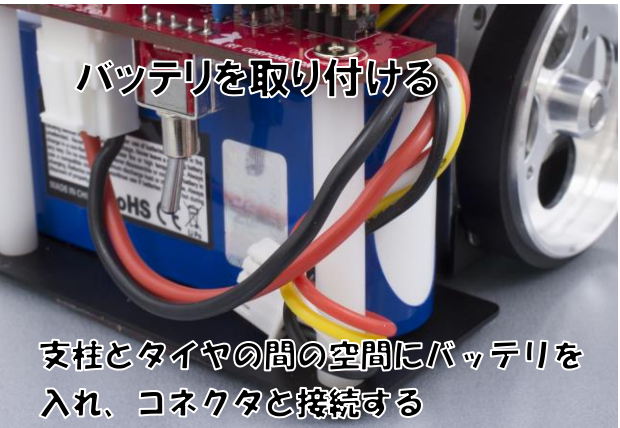



## 1-5 その他



<p>電源電圧監視用 LED</p>  <p>FULL → EMPTY</p>	<p>バッテリーの電圧を監視する LED です。</p> <ul style="list-style-type: none"> <li>●FULL 状態の水色から電圧が低くなるにしたがって赤くなっていきます。LED キャップをつけると色がわかりやすくなります。(写真は白の LED キャップをつけてあります)</li> <li>●MAX で 12.6V です。11V 以下になる前に充電しましょう。9V 以下になると再充電ができなくなってしまいます。</li> </ul> <p>※LED の極性を指定と逆に付けると上記とは反対に FULL 状態で赤く光り電圧が低くなると青く光りますが、動作には問題ありません。</p>
<p>マイコン電源確認用 LED</p>	<p>マイコンへの通電が確認できる LED です。</p>
<p>電源確認用 LED</p>	<p>電源スイッチが ON になると点灯します。</p>
<p>壁センサ用 LED</p>	<p>壁の有無、距離を調べる為に前壁、横壁に向けて赤色発光します。</p>
<p>壁センサ用フォトランジスタ</p>	<p>壁に反射した壁センサ用 LED の光を受け、壁の有無、距離を感知します。</p>
<p>ホイール</p>	<p>左右対称になるよう、モータドライバに取り付けます。</p>
<p>タイヤ</p>	<p>ホイールに取り付けるゴム製のタイヤです。両面テープで取り付けるとずれません。</p>

## 1-5-1 バッテリを取り付ける

 <p><b>バッテリーを取り付ける</b></p> <p>支柱とタイヤの間の空間にバッテリーを入れ、コネクタと接続する</p>	<p>バッテリーを本体に取り付けます。</p> <ul style="list-style-type: none"><li>●電源基板側の支柱とタイヤの間の空間にバッテリーを入れ、電源基板のコネクタと接続します。</li></ul> <p>写真のように充電用コネクタを巻き込んでつけると走行中邪魔になりません。</p> <ul style="list-style-type: none"><li>●本体のメンテナンスなどで基板をはずすときや、使用を終了するときはコネクタをはずしてください。ショートや放電の原因になります。</li></ul>
 <p><b>バッテリーの止め方 (例 1)</b></p> <p>約 20mm</p> <p>支柱と電池の幅(約 20mm)の所にストッパをつける</p>	<p>バッテリーを安定させます。</p> <ul style="list-style-type: none"><li>●本体にバッテリーを取り付けたとき、そのままにしておくとバッテリーがタイヤに当たってまっすぐ走れなくなることがあるので当たらないようにします。</li><li>●たとえば支柱とタイヤの間、バッテリーの厚み分のところにストッパーをつけ、これ以上タイヤ方向に動かないようにします。</li></ul>
 <p><b>バッテリーの止め方 (例 2)</b></p> <p>養生テープなどのはがしやすいテープでバッテリー(ケーブルのない方)と支柱を固定する</p>	<ul style="list-style-type: none"><li>●養生テープなどのはがしやすいテープでバッテリーと支柱を固定します。</li></ul> <p>このほかにもやりやすい方法でバッテリーを固定してみてください。</p>

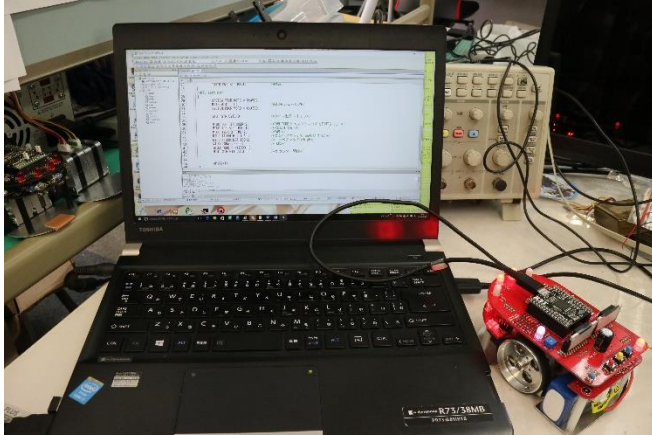


## 2 プログラムを書き込む

プログラムを書き込まないことにはマウスは動きません。

まずはプログラムを書き込みましょう。サンプルプログラム各 STEP と同じ書き込み方です。

### 準備



### 用意するもの

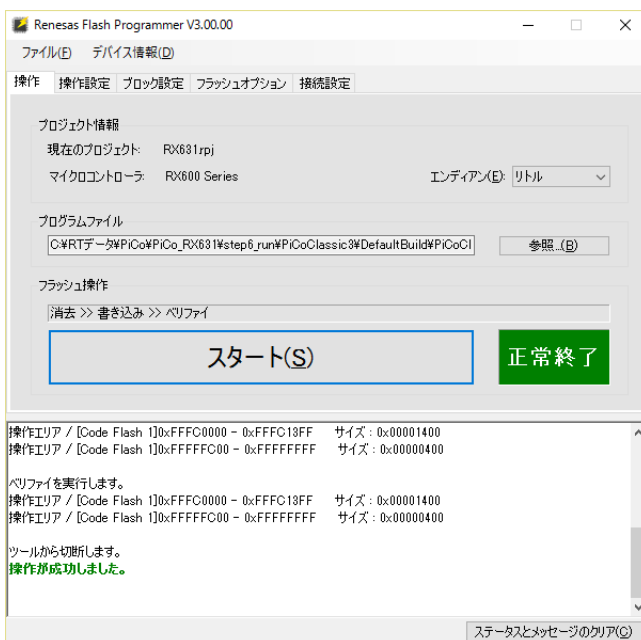
本体

パソコン(以下 PC) (RFP と GS+に使いたいプログラムを表示しておく)

USBmicroB ケーブル(以下 USB ケーブル)

バッテリー

### プログラムを書き込む

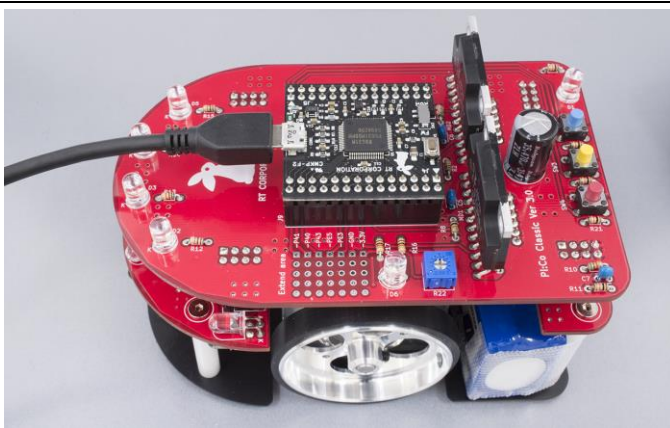


●PC の GS+にサンプルプログラム各 STEP を表示、必要ならば必要箇所に数値を入力します。

●入力したら F7 キーを押す、もしくは「PiCoClassic3 をビルド」でビルドします。

●ビルドできたら RFP を表示し、ビルドしたプログラムを参照します。違うプログラムを選ばないように気をつけましょう。

●PC と本体の書き込み端子を USB ケーブルでつなぎます。



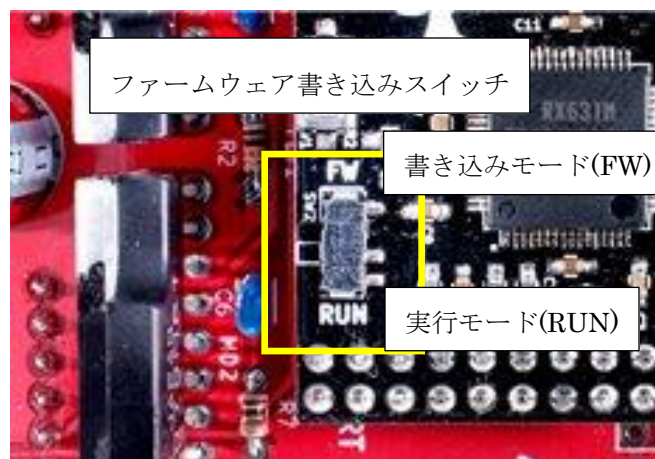
●本体のファームウェア書き込みスイッチを書込みモード (FW) にします。

●本体の電源スイッチを ON

●RFP のスタートボタンを押し、プログラムを書き込みます。

●書き込みが終了したら本体の電源を切り、USB ケーブルをはずします。

## 動かしてみる



●本体のファームウェア書き込みスイッチを実行モード (RUN) にします。

●電源スイッチを ON

(モード表示用 LED モード 1 状態、マイコン電源確認用 LED、電源確認用 LED、電源電圧監視用 LED が点灯します)

●モード選択スイッチでモードを選び、モード決定スイッチで実行します。

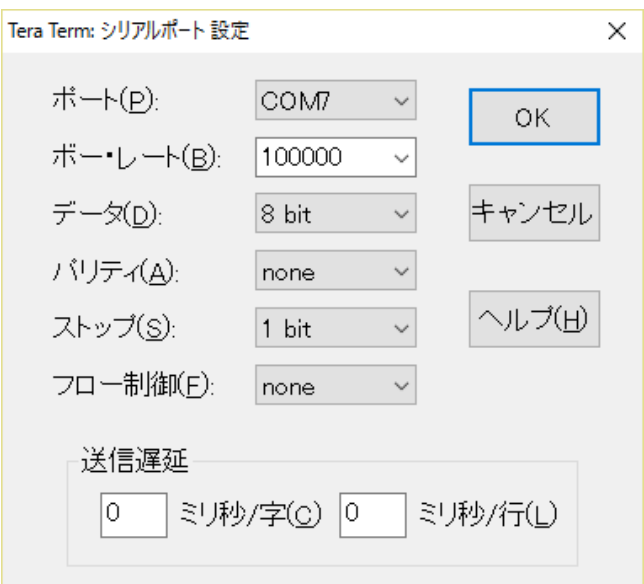
### 3 実際に迷路を走らせる為の調整

プログラムを入れただけではまだ上手く走れません。安定した走行をさせる為に、センサやターン角度などの調整をします。

本取扱説明書ではこれ以降、サンプルプログラムは Step8\_running\_maze を使用します。

#### 3-1 ターミナルエミュレータの準備

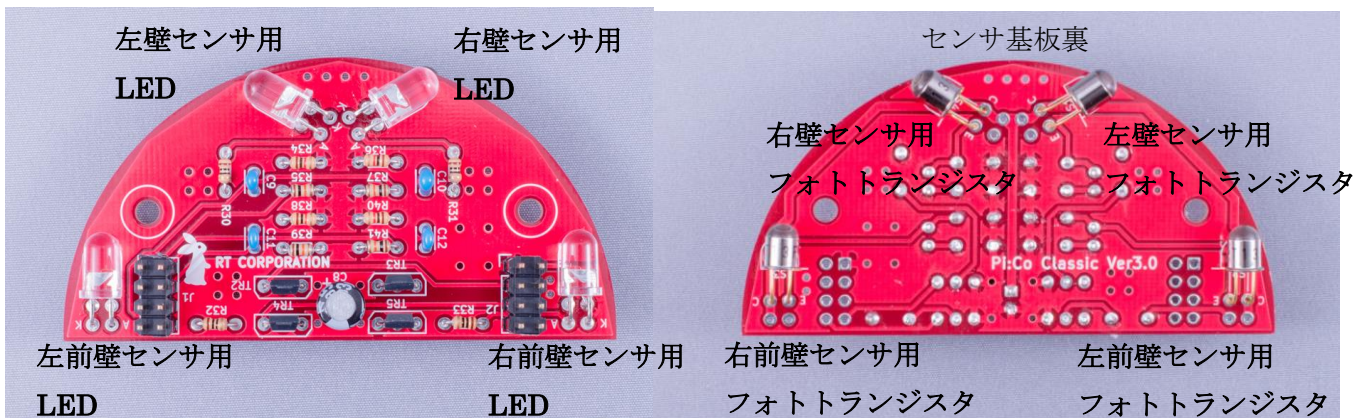
センサの値を取る為にターミナルエミュレータを使用します。ここではフリーソフトの Tera Term を使用しています。使いやすいターミナルエミュレータを使ってください。

ターミナルエミュレータシリアルポート設定	
	<ul style="list-style-type: none"><li>●ターミナルエミュレータを用意します。ダウンロード等して、PC にインストールしてください。</li><li>●PC と本体を USB ケーブルで接続します。</li><li>●ターミナルエミュレータを起動し、ケーブルをさしている com ポート番号に設定してください。ボーレートは任意です。</li><li>●Tera Term の場合、メニューバ「ファイル」の「新しい接続」で com ポートを設定できます。</li></ul> <p>設定の保存はメニューバ「設定」の「設定の保存」で行います。</p> <p>保存しておけば次回からは設定の必要はありません。</p>

## 3-2 センサ調整

### 3-2-1 センサの角度調整

壁の有無、距離は、壁センサ用 LED からの反射光をフォトトランジスタで受けることで計測しますので、上下、横から見て受光できる角度に調整してください。

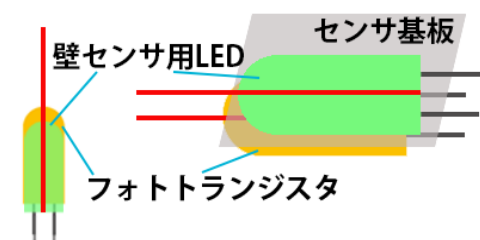


横から見た角度



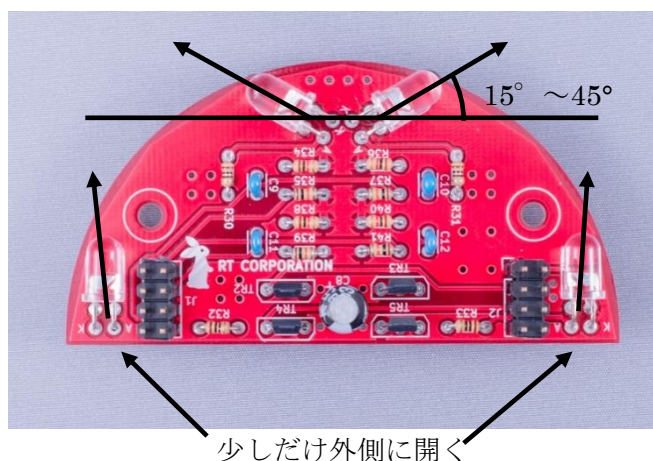
図のように受光しやすいよう、壁に向かって少々角度をつけてください。

上下から見た角度



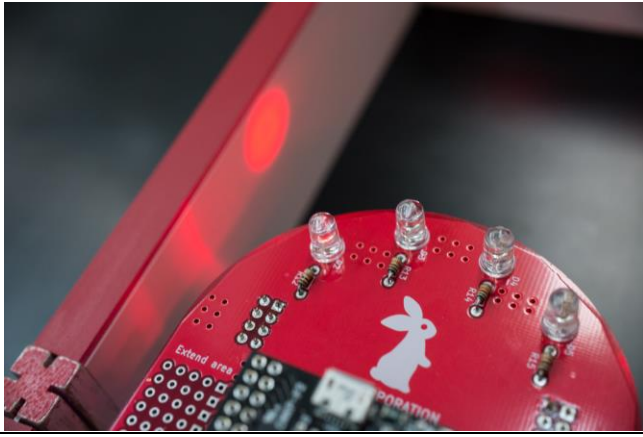
壁センサ用 LED とフォトトランジスタの向いている方向（光軸）が平行になるようにしてください。

センサ基板全体



左右の壁センサ用 LED とフォトトランジスタは真横ではなく、前壁に対して  $15^{\circ} \sim 45^{\circ}$  位傾けます。  
前壁センサ用 LED とフォトトランジスタは真正面ではなく、ほんの少し外側に開きます。これは LED の光が広がる為です。内側に狭めるのはやめましょう。

## 光の位置



壁に当たる赤い光が壁の上にはみ出さないように調整してください。

角度が決まったら樹脂などで固めておく touch した拍子などにずれるということがなくなります。



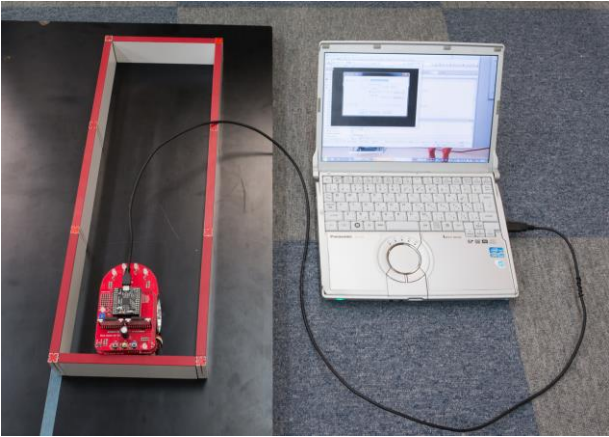
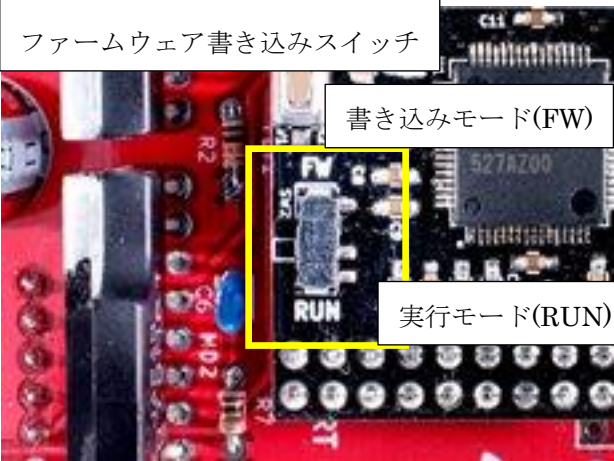
### 3-2-2 センサ用パラメータの設定

センサ用パラメータを設定します。サンプルプログラムの parameters.h に入力します。  
壁の有無を決める値を「壁の閾値（しきいち）」と言います。

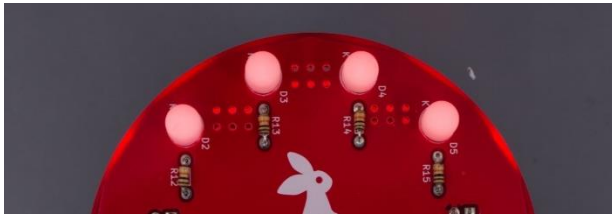
各種センサ用変数は以下のようにになっています。

- REF\_SEN\_R：右センサの目標値（本体が迷路の真ん中を走る為の右センサの目標値）
- REF\_SEN\_L：左センサの目標値（本体が迷路の真ん中を走る為の左センサの目標値）
- TH\_SEN\_R：右センサの閾値（本体が右壁から一番離れたときの右センサの値）
- TH\_SEN\_L：左センサの閾値（本体が左壁から一番離れたときの左センサの値）
- TH\_SEN\_FR：右前センサの閾値（本体が前壁を認識するときの右前センサの値）
- TH\_SEN\_FL：左前センサの閾値（本体が前壁を認識するときの左前センサの値）

各種センサ用変数の設定方法

<p>準備</p> 	<p>本体、PC、USB ケーブル、迷路</p> <p>●PC と本体を USB でつなぎ、ターミナルエミュレータを起動します。</p>
<p>ファームウェア書き込みスイッチ</p>  <p>書き込みモード(FW)</p> <p>実行モード(RUN)</p>	<p>●本体のファームウェア書き込みスイッチを実行モード (RUN) にしてから電源を ON にします。</p>

全点灯させてから

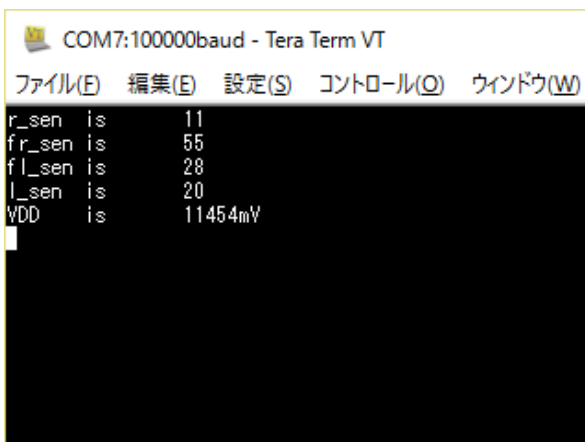


調整モード1へ



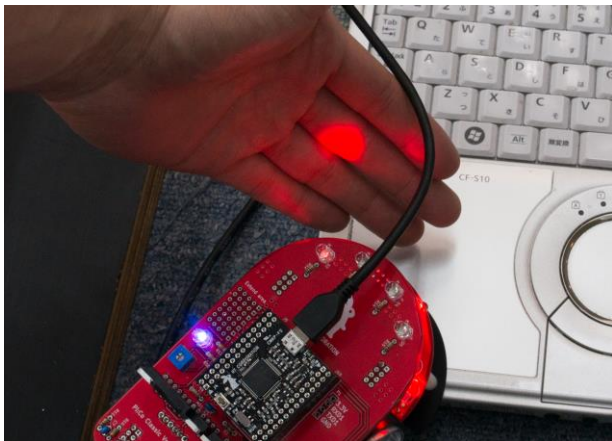
- モード選択スイッチでモード15 (LED 全点灯)を選び、モード決定スイッチで決定  
→調整モード1になります。  
(ビット1点灯、ビット4点滅)
- 調整モード1のまま、決定  
→ターミナルエミュレータにセンサの値が表示されます。

センサの値



●画面のセンサの値

r_sen	右センサ
fr_sen	右前センサ
fl_sen	左前センサ
l_sen	左センサ


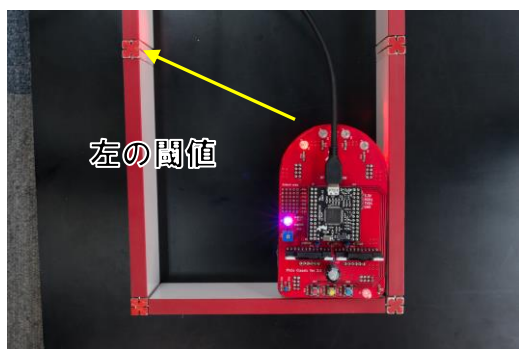


- 手などをかざしてみて、画面のセンサの値が変化するか確認しましょう。

## マウスの姿勢制御用目標値を決める (REF\_SEN\_R・REF\_SEN\_L)

	<p>マウスが迷路の真ん中を走るように設定します。</p> <p>●ターミナルエミュレータにセンサの値が表示された状態で、本体を迷路の真ん中に、横壁と平行に置きます。</p>
<pre> 30 31 //センサ関連パラメータ 32 #define WAITLOOP_SLED 600 33 34 #define REF_SEN_R 242 35 #define REF_SEN_L 216 36 37 //センサの値をメモする 38 39 40 #define TH_SEN_FR 125-15 41 #define TH_SEN_FL 121-15 42 </pre> <p><b>真ん中に置いたときの値</b></p>	<p>●ターミナルエミュレータに表示されている右センサ (r_sen is ○○) の値を REF_SEN_R に、左センサ (l_sen is ○○) の値を REF_SEN_L に入力します。</p> <p>センサの値をメモしておいてから最後にまとめて入力してもいいでしょう。</p>

## 左右の壁の有無を決める閾値の設定 (TH\_SEN\_R・TH\_SEN\_L)

<p>左右の閾値を設定する</p>  <p>右の閾値</p>	 <p>左の閾値</p>
<pre> 30 31 //センサ関連パラメータ 32 #define WAITLOOP_SLED 600 33 34 #define REF_SEN_R 242 35 #define REF_SEN_L 216 36 37 //センサの値をメモする 38 39 40 #define TH_SEN_FR 125-15 41 #define TH_SEN_FL 121-15 42 </pre> <p><b>左右の閾値</b></p> <pre> 37 #define TH_SEN_R 148 38 #define TH_SEN_L 142 39 40 #define TH_SEN_FR 125-15 41 #define TH_SEN_FL 121-15 42 </pre>	<p>●迷路の左端に本体を置き、ターミナルエミュレータの (r_sen is ○○) の値を TH_SEN_R に、右端に本体を置き、(l_sen is ○○) の値を TH_SEN_L に入力します。</p>

●左センサの閾値を取るときに支柱の外に出ているバッテリーのケーブルが気になる方は、バッテリーを本体の外に出して計測してください。(センサ設定を始める前の、電源を ON にする前でないとい外に出すのは難しいです。設定中にバッテリーのコネクタをはずすのはやめてください。)

## 前壁の有無を決める閾値の設定 (TH\_SEN\_FR ・ TH\_SEN\_FL)

<p>前壁の閾値を設定する</p> 	<p>マウスは壁から 1 区間前で前壁の有無を認識します。</p> <p>あまりに壁に近すぎる位置で設定すると、壁を認識できずぶつかってしまったり、反対に壁から遠すぎる位置で設定すると、本来ない所に壁があると認識してしまうことがあるので、丁寧に設定してください。</p>
<p>柱と LED が並ぶくらいを目安に</p> 	<p>●前壁の閾値をとるときの本体の位置は、壁の 1 つ手前の柱と LED が並ぶくらいを目安にしてください。</p> <p>車軸から 3 cm くらいと思ってください。</p>
<pre> 30 31 //センサ関連パラメータ 32 #define WAITLOOP_SLED 600 33 34 #define REF_SEN_R 242 35 #define DEF_SEN_T 218 36 37 <b>前壁の閾値</b> 38 #define TH_SEN_L 142 39 40 #define TH_SEN_FR 125-15 41 #define TH_SEN_FL 121-15 42 </pre>	<p>●ターミナルエミュレータの (fr_sen is ○○) の値を TH_SEN_FR に、 (fl_sen is ○○) の値を TH_SEN_FL に入力します。</p>

●値をすべて入力し終えたらビルドして本体に書き込みます。

### 3-3 物理的な調整

#### 3-3-1 ゲイン数について

マイクロマウスはセンサで取得した値から迷路の真ん中を走るように計算します。右寄りを走れば左へ、左寄りになれば右へと真ん中を目指して制御をかけます。


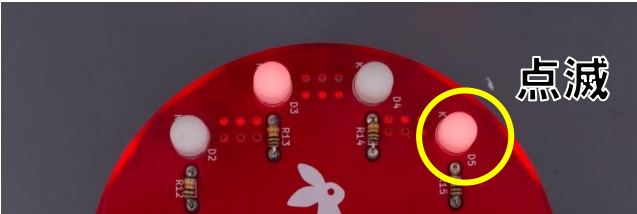
 <p>ゲイン値が高いと急な制御をする</p> <p>ゲイン値を低くすると緩やかな制御をする</p>	<p>比例制御のゲイン数の値が高いほど、姿勢を修正する動きが急になります。急なほど横壁にぶつかりやすくなるのでちょうどよい値を探しましょう。これは迷路を走っているときにわかります。走行中にくねくねと曲がって横壁にぶつかる時はゲイン数が高いときです。</p> <p>parameters.h 内の比例制御のゲイン数 CON_WALL_KP の値を調整します。強くしたいときは+、緩やかにしたいときは-にします。</p> <p>サンプルプログラムは 2.0 になっています。これを最高と考え、徐々に低くして調整していきます。</p>
 <pre>41 42 #define CON_WALL_KP (2.0) 43 44</pre> <p>ゲイン値を設定</p>	<p>●調整をする時には CON_WALL_KP (ゲイン数) を 0 にします。</p> <p>これは、調整時に制御がかかると正確な調整が出来ないからです。</p> <p>ゲイン数を 0 にしてビルドしたプログラムを本体に書き込んでから、物理的な調整をしてください。</p>

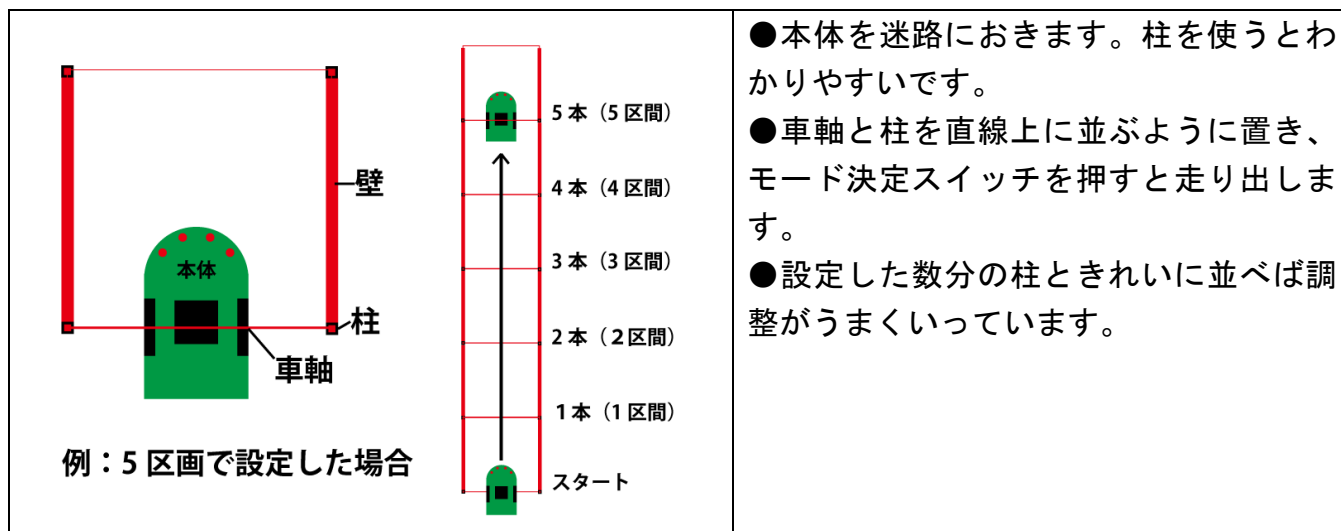
※調整が終わったら忘れず値を入れ直しましょう。直し忘れると制御せず走ってしまいます。



### 3-3-2 走行距離の調整(タイヤ直径チェック)

走る距離やターンの角度が適当なままでは、迷路を走っているうちにどんどん誤差が溜まって走行位置が大きくずれてきます。

<pre> 138 139 int exec_by_mode_adjust(int mode) 140 141 //引数modeを受け取り、そのモードに入る。 142 //戻り値が1の場合は、大本のメニューに戻る 143 switch(mode) 144 { 145     case 1: センサ設定 (センサの値を出力する) 146             view_adc(); //A/D変換の結果を8011から出力する 147             break; 148 149     case 2: タイヤ直径チェック(直線距離) 150             straight_check(9); //9区画直線を走る 151             break; 152 153     case 3: タイヤトレッド径チェック(ターン角度) 154             go_and_turn_right(); 155             break; 156 </pre>	<p>タイヤの直径をチェックします。</p> <p>タイヤのゴムが磨り減ってきたりすると、タイヤの直径も変わってきます。</p> <p>なるべく長い直線を走らせて調整します</p>
<p>準備</p> <pre> 138 139 int exec_by_mode_adjust(int mode) 140 141 //引数modeを受け取り、そのモードに入る。 142 //戻り値が1の場合は、大本のメニューに戻る 143 switch(mode) 144 { 145     case 1: センサ設定 (センサの値を出力する) 146             view_adc(); //A/D変換の結果を8011から出力する 147             break; 148 149     case 2: タイヤ直径チェック(直線距離) 150             straight_check(9); //9区画直線を走る 151             break; 152 153     case 3: タイヤトレッド径チェック(ターン角度) 154             go_and_turn_right(); 155             break; 156 </pre> <p><b>直線距離を走ります</b></p>	<p>迷路の大きさによって走れる距離が違うので、何区間を走るかをあらかじめプログラムに入力します。</p> <p>●サンプルプログラムの adjust.c case 2 に走る区間数を入力します。</p> <p>写真では ( ) 内が (9) となっているのでここでは9区間です。この数字を変え、ビルドします。</p> <p>なるべく長い直線を走らせてみます。</p>
<p>全点灯させてから</p>  <p>調整モード2へ</p> 	<p>●区間数を設定したプログラムを本体に書き込みます。</p> <p>●USB ケーブルを本体からはずし、ファームウェア書込みスイッチを実行モード (RUN) にしてから電源を ON にします。</p> <p>●モード選択スイッチ(送り)を押し、LED を全点灯させてからモード決定スイッチを押し、調整モードにします。</p> <p>●モード選択スイッチ(送り)を一つ押し、調整モード2を選びます。</p>



### ぴったり走れなかった場合

```

9
10 //物理的なパラメータ
11 #define TIRE_DIAMETER (48)
12 #define TREAD_WIDTH (64+4.23)
13 #define HALF_TREAD (0.5*TREAD_WIDTH)
14 #define TREAD_CIRCUIT (TREAD_WIDTH*PI/4)
15
16

```

●走行距離が長すぎたり短すぎたりするときは、サンプルプログラム parameters.h の TIRE\_DIAMETER で調整します。

デフォルトは 48、微妙な調整になるので小数点以下での調整になります。

進みすぎたら (48+0.1) のように値を+で大きくします。

足りない場合は (48-0.1) のように値を-で小さくします。

●入力が終わったらビルドして本体に書き込み、再度走らせて確認します。

### ●なぜ「多いのに+、少ないのに-」にするのか？

→進むべき距離に対して、実際のタイヤの直径よりも TIRE\_DIAMETER の値が小さく設定されていると、よりたくさん回転して計算上の距離を進もうとします。

しかし、実際の直径は設定よりも大きいのでたくさん回転した分、進みすぎてしまいます。ですから進みすぎたときは TIRE\_DIAMETER の値が実際のタイヤの直径よりも小さいということなので、値を+して大きくし、本来の直径にするのです。

### ●数学的にタイヤの直径を算出

仮に 5 区画走行して、10mm 足りなかったとします。プログラム上では、タイヤの直径が 48mm で 5 区画=5×180=900mm をきっちり走ったと思い込んでいます。しかし、実際は 900-10=890mm しか進んでいないためタイヤの直径がプログラム上で設定した値と違うことになります。これを数式に置き換え、実際のタイヤの直径を求めると

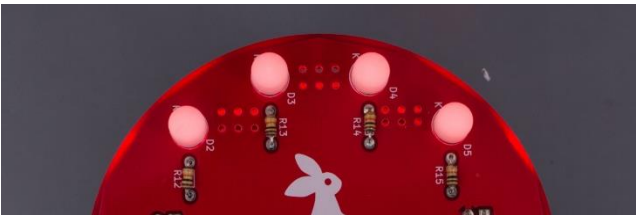

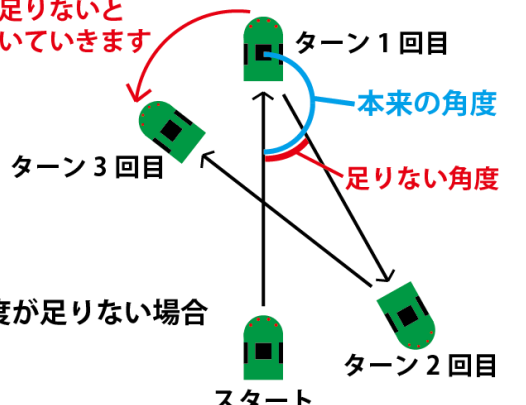
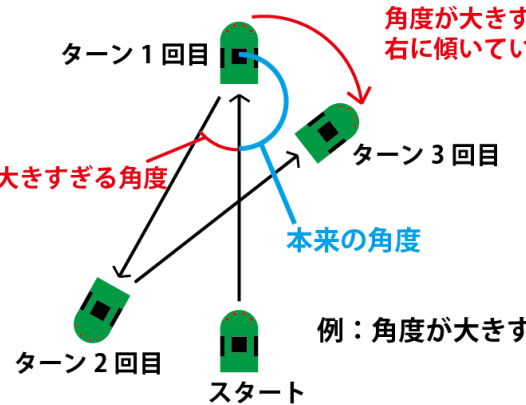
設定値    実際の値

48:900 = x :890    =>    900x = 48 × 890    =>    x = 47.47mm とタイヤの直径が求まります。

### 3-3-3 旋回の角度調整（トレッド幅チェック）

走行距離と旋回（ターン）の角度がぴったりに決まるととてもきれいに動きます。

この調整は柱や壁のない所で行います。

<pre> 138 139 int exec_by_mode_adjust(int mode) 140 { 141     //引数modeを受け取り、そのモードに入る。 142     //戻り値が1の場合は、大本のメニューに戻る 143     switch(mode) 144     { 145         case 1: センサ設定（センサの値を出力する） 146                 view_adc(); //A/D変換の結果を8011から出力する 147                 break; 148         case 2: タイヤ直径チェック(直線距離) 149                 straight_check(9); //9区画直線走る 150                 break; 151         case 3: タイヤトレッド径チェック(ターン角度) 152                 go_and_turn_right(); 153                 break; 154     } 155 } 156 </pre>	<p>ターン角度を調整します。</p> <p>このモードは2区間の往復を繰り返します。角度があっていないと繰り返すうちにどんどん斜めにずれていきます。きれいに往復できるよう調整します。</p>
<p>全点灯させてから</p>  <p>調整モード3へ</p> 	<ul style="list-style-type: none"> <li>●モード選択スイッチ（送り）を押して全点灯させてからモード決定スイッチを押し、調整モードに入ります。</li> <li>●モード選択スイッチ（送り）を二回押して調整モード3を選びます。</li> <li>●周りに壁のない状態の迷路、場所に本体を置きます。</li> <li>●モード決定スイッチを押すと2区間の往復を始めます。</li> <li>●このモードは往復したままなので、角度のチェックをしたら本体を持ち上げ、本体の電源をOFFにしてください。</li> </ul>
<p>角度が足りないと左に傾いていきます</p>  <p>例：角度が足りない場合</p>	<p>角度が大きすぎると右に傾いていきます</p>  <p>例：角度が大きすぎる場合</p>
<pre> 5 6 //物理的なパラメータ 7 #define TIRE_DIAMETER (40) 8 #define TREAD_WIDTH (62+4.23) 9 #define TREAD_CIRCUM (TREAD_WIDTH*PI/4) 10 11 </pre>	<ul style="list-style-type: none"> <li>●ずれてきた場合は、サンプルプログラムの parameters.h の TREAD_WIDTH で調整します。デフォルトは64、大きいときは(64-4)のように値を-で小さくします。小さいときは(64+4)のように値を+で大きくします。</li> <li>●微妙な調整をしたいときは(64+4.23)のように小数点以下も使用できます。</li> <li>●入力が終わったらビルドして本体に書込み、再度動かして確認します。</li> </ul>

●値を変えて動かしてみた後、再度動かしたときに同じ値でも違う動きをすることもあります。  
数回同じ値で動かしてみて、動きを確認してください。

納得するまで調整しましょう。

●タイヤの表面に付いた「ほこり」をこまめにとって調整してください。

「5 その他 5-3 タイヤのごみを取る」を参照

●数学的にトレッド幅を算出

タイヤの直径と同じように一回の回転ではなく、連続して回転させ角度の差分から実際のトレッド幅を算出します。仮に 360 度回転(90 度ターン 4 回分)し 10 度足りないとします。プログラム上ではトレッド 64mm で 4 回きっちり回ったと思い込んでいるので、先ほどの直線のとおりと同じように求めると

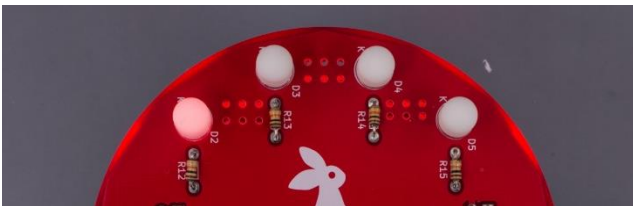
設定値	実際の値
$\frac{360}{360} \cdot 64\pi$	$\frac{350}{360} \cdot x\pi \rightarrow x = \frac{36}{35} \cdot 64 = 65.83mm$

## 4 迷路を走らせてみる

調整が終わったら迷路を走らせてみましょう。

<pre> 119 void exec_by_mode(int mode) //モード番号に従って実行する 120 { 121     MOT_POWER_ON; 122     wait_ms(1000); 123 124     switch(mode) 125     { 126     case 1: //左手法 127         search_left_hand(); 128         break; 129 130     case 2: //足立法で行って帰ってくる 131         mypos.x = mypos.y = 0; //座標を初期化 132         mypos.dir = north; //方向を初期化 133         search_adachi(GOAL_X, GOAL_Y); //ゴールまで足立法 134         rotate(right, 2); //ゴールしたら180度回転する 135         mypos.dir = (mypos.dir+6) % 4; //方向を更新 136         goal_appeal(); //ゴールしたことをアピール 137         search_adachi(0, 0); //スタート地点まで足立法で帰ってくる 138         rotate(right, 2); //帰ってきたら180度回転 139         break; 140 141     case 3: //最短走行 142         mypos.x = mypos.y = 0; //座標を初期化 143         mypos.dir = north; //方向を初期化 144         accel = 2.0; //加速度を設定 145         fast_run(GOAL_X, GOAL_Y); //ゴールまで最短走行 146         mypos.dir = (mypos.dir+6) % 4; //方向を更新 147         rotate(right, 2); //ゴールしたら180度回転 148         goal_appeal(); //ゴールしたことをアピール 149         fast_run(0, 0); //スタート地点まで最短走行 150         rotate(right, 2); //帰ってきたら180度回転 151         break; 152 153         //以下モード15以外は空き。自分でいろいろ作ってみてください。 154 155     case 4: break; 156 157     } 158 </pre>	<p>サンプルプログラムには3種のモードが入っています。</p> <p>サンプルプログラム PicoClassic3.c 行番119あたりからモードに関しての記述があります。</p> <p>●ファームウェア書込みスイッチを実行モード (RUN) に入れてから電源スイッチを ON にします。</p>
---	---

### 4-1 左手法

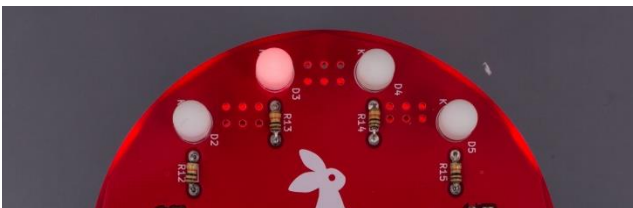


case 1 に左手法が入っています。本体ではモード 1 を選びます。

●電源を ON にした時点でモード 1 になっているので、そのままモード決定スイッチを押します。

このプログラムは探索していません。スタートもゴールも認識せず、ひたすら左壁に沿って走り続けます。

### 4-2 足立法



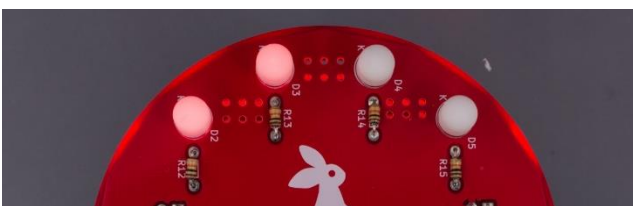
case 2 に足立法が入っています。本体ではモード 2 を選びます。

●モード選択スイッチでモード 2 を選択し、モード決定スイッチを押します。

こちらはゴールしてアピールした後、再度足立法でスタート地点へ戻ってきます。

戻ってきた後、決定ボタンを押すと足立法で重ね探索ができます。

### 4-3 最短走行



足立法での探索が終わったら最短走行を試してみましょう。Case 3 には最短走行が入っています。本体ではモード 3 を選びます。

●モード選択スイッチでモード 3 を選択し、モード決定スイッチを押します。

探索した道順の中から最短経路を見つけて走ります。

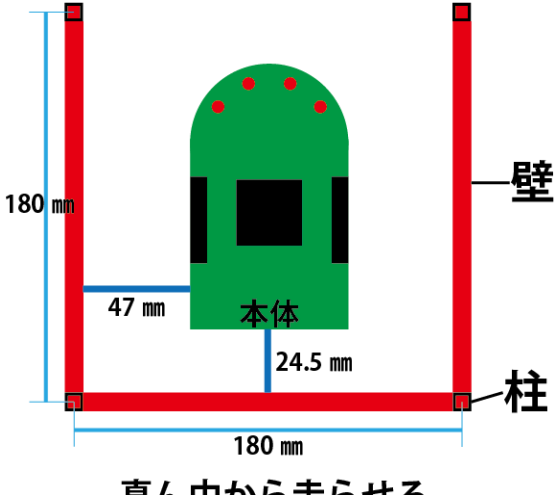
●Case4 からはあきになっています。

プログラムを書き換えたり、追加したり、自由に楽しんでください。



## 5 その他

### 5-1 本体を置く位置

 <p>180 mm</p> <p>47 mm</p> <p>24.5 mm</p> <p>180 mm</p> <p>壁</p> <p>柱</p> <p>本体</p> <p>真ん中から走らせる</p>	<p>●本体は調整のとき以外は区画の真ん中から走らせます。</p> <p>図のように横 47 mm、後ろ 24.5mm のあたりに置いて走らせてください。</p>
--	---

### 5-2 ゴール地点の数え方

<p><b>START(=0) から数える</b></p> <table border="1"><tr><td>3</td><td></td><td></td><td></td><td>GOAL</td></tr><tr><td>2</td><td></td><td></td><td>GOAL 座標 (x=4)</td><td>(y=3)</td></tr><tr><td>1</td><td></td><td></td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr></table> <p>Y 軸</p> <p>START</p> <p>X 軸</p>	3				GOAL	2			GOAL 座標 (x=4)	(y=3)	1					0	1	2	3	4	<p>●ゴール地点を決めなければマウスは走りません。</p> <p>スタートを 0 として数えます。X 軸は 0 から横に 1、2、3、4 と数えます。同じように Y 軸も 0 から縦に数えます。</p> <p>この図の例ではゴール座標は X=4, Y=3 になります。</p>
3				GOAL																	
2			GOAL 座標 (x=4)	(y=3)																	
1																					
0	1	2	3	4																	
<pre>32 //迷路関連パラメータ 33 34 #define GOAL_X 3 //ゴール座標(x) 35 #define GOAL_Y 3 //ゴール座標(y) 36</pre>	<p>●ゴール座標を parameters.h の迷路関連パラメータに入力します。</p> <p>数字だけ変えます。( ) はいりません。</p>																				

### 5-3 タイヤのごみを取る



●マウスを走らせていると、タイヤにほこりなどが付いてスリップの原因になったりします。

まめに取り除くようにします。

養生テープやカーペットクリーナでタイヤを転がすように動かし、ほこりを取ります。