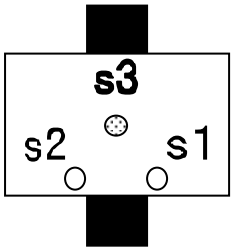
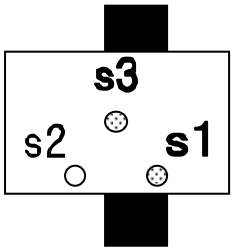
６　「ライントレース」制御に挑戦　～センサ(s1，s2)に注目～

＜基本的な考え方＞　左右２個のセンサ(s1，s2)でラインをはさむと，
「ラインから外れた場合」，左右のどちら側に外れたのかが分かる。
↓
したがって，常に「ライン中央に戻す」制御が可能となる。

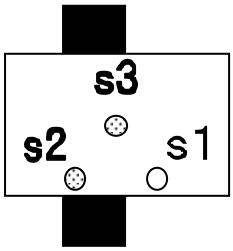
問題　２個のセンサ(s1，s2)による制御の場合，下図の**センサ入力（10 進数）**と，**モータ出力（10 進数）**はどうなる？ヒント：poke5 を実行してみよう



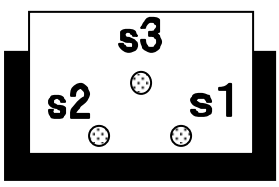
コース①（ライン中央）　　s3　s2　s1
センサ入力（ＩＮＰ）＝（1　0　0）$_2$＝＿＿＿＿
※[s1＝０，s2＝０]なので，制御データは「直進」
↓
モータ出力　＝　ＯＵＴ　＿＿＿＿



コース②（左に少しずれた状態）　s3　s2　s1
センサ入力（ＩＮＰ）＝（1　0　1）$_2$＝＿＿＿＿
※[s1＝１，s2＝０]なので，制御データは「右へ軌道修正」
↓
モータ出力　＝　ＯＵＴ　＿＿＿＿

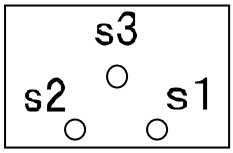


コース③（右に少しずれた状態）　s3　s2　s1
センサ入力（ＩＮＰ）＝（1　1　0）$_2$＝＿＿＿＿
※[s1＝０，s2＝１]なので，制御データは「左へ軌道修正」
↓
モータ出力　＝　ＯＵＴ　＿＿＿＿



コース④（停止位置）　　s3　s2　s1
センサ入力（ＩＮＰ）＝（1　1　1）$_2$＝＿＿＿＿
※[s1＝１，s2＝１]なので，制御データは「ブレーキ」
↓
モータ出力　＝　ＯＵＴ　＿＿＿＿

整理してまとめる	センサ入力	モータ出力	センサ入力とモータ出力の関係はどうなるか？
コース①			
コース②			
コース③			
コース④			



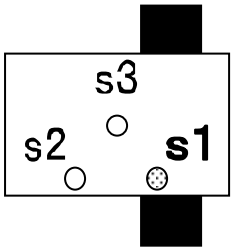
コース⑤（ラインの外）　　s3　s2　s1

センサ入力（ＩＮＰ）＝（0　0　0）$_2$＝＿＿＿＿

※[s1＝０，s2＝０]なので，制御データは「直進」

↓

モータ出力 ＝ ＯＵＴ ＿＿＿＿



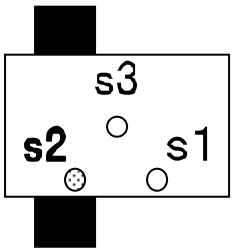
コース⑥（左にずれた状態）　　s3　s2　s1

センサ入力（ＩＮＰ）＝（0　0　1）$_2$＝＿＿＿＿

※[s1＝１，s2＝０]なので，制御データは「右へ軌道修正」

↓

モータ出力 ＝ ＯＵＴ ＿＿＿＿



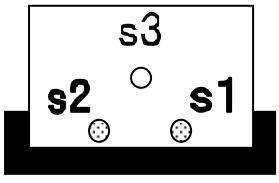
コース⑦（右にずれた状態）　　s3　s2　s1

センサ入力（ＩＮＰ）＝（0　1　0）$_2$＝＿＿＿＿

※[s1＝０，s2＝１]なので，制御データは「左へ軌道修正」

↓

モータ出力 ＝ ＯＵＴ ＿＿＿＿



コース⑧（停止位置）　　s3　s2　s1

センサ入力（ＩＮＰ）＝（0　1　1）$_2$＝＿＿＿＿

※[s1＝１，s2＝１]なので，制御データは「ブレーキ」

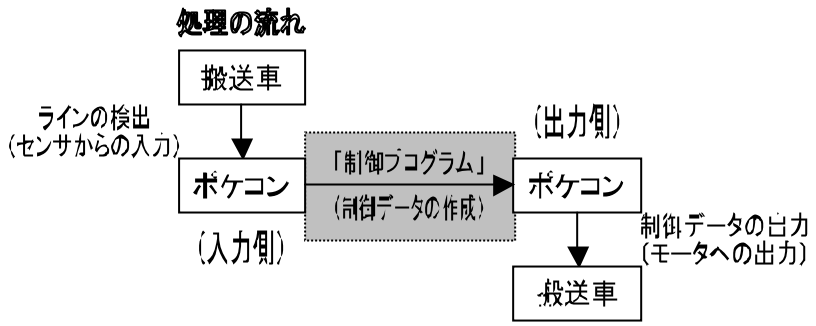
↓

モータ出力 ＝ ＯＵＴ ＿＿＿＿

整理してまとめる	センサ入力	モータ出力	センサ入力とモータ出力の関係はどうなるか？
コース⑤			
コース⑥			
コース⑦			
コース⑧			

★「ライントレース」制御するプログラムを作ろう（ＢＡＳＩＣによる制御）

＜プログラムの考え方＞



＜プログラム化＞

手順１　ラインの検出（センサからの入力）　⇨　「ＩＮＰ命令」

手順２　制御データの作成　⇨ **ＩＦ文**（もし，４より小さかったら）＋４

手順３　制御データの出力（モータへの出力）　⇨　「ＯＵＴ命令」

手順４　手順１へ戻る

問題　プログラムを作ってみよう。ヒント：poke6 を実行してみよう

手順１　10 _____

手順２　20 _____

手順３　30 _____

手順４　40　ＧＯＴＯ　10

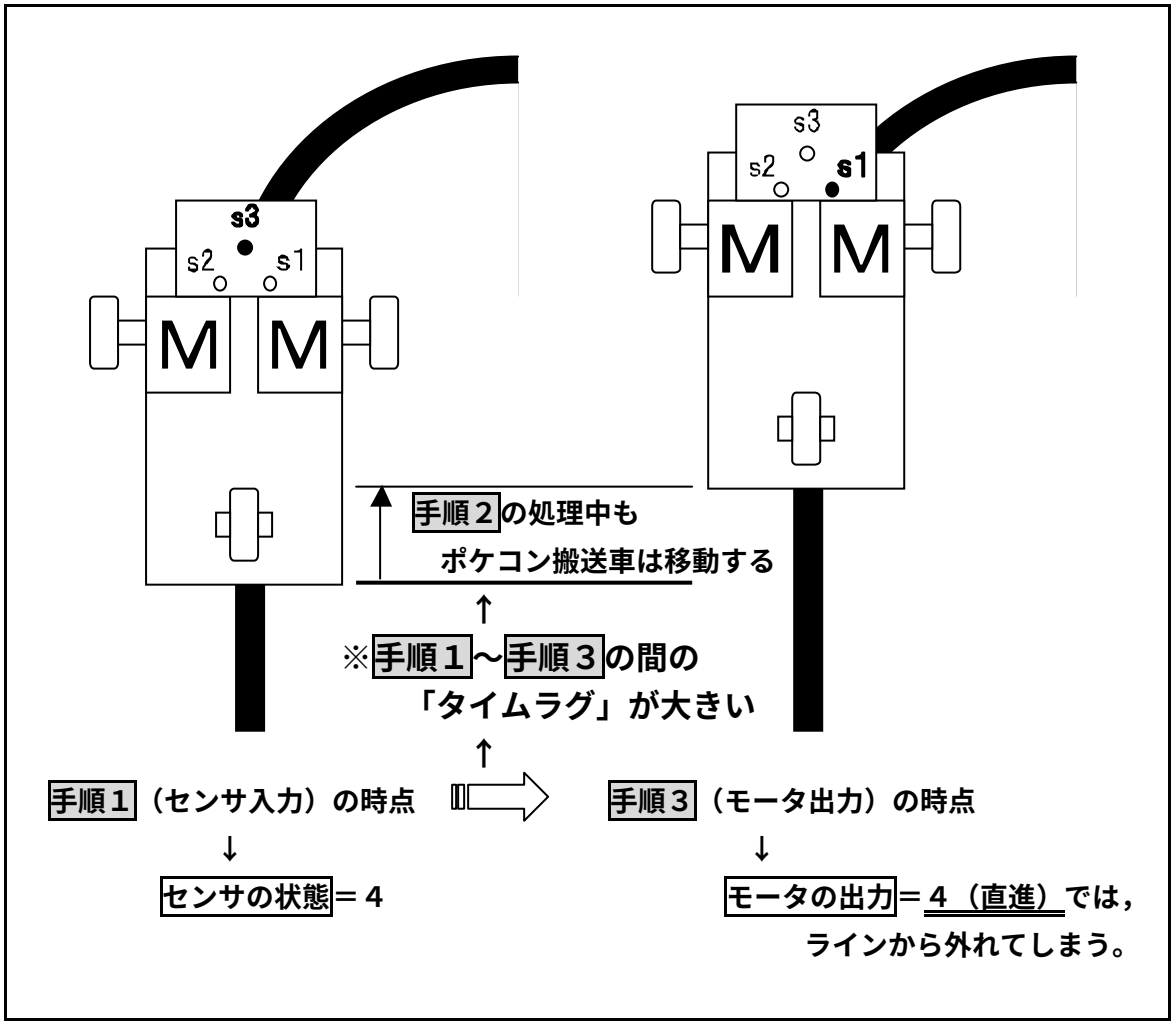
※自分の「ポケコン」と「搬送車」で試してみよう。

気付いたことをまとめよう。

［考察］前のページのプログラムで「**ライントレース**」制御が，

うまくできないのは，なぜ？

＜答え＞　・・・・　それは，「**タイムラグ**」が大きいから（下図に注目）



※「**タイムラグ**」とは，

・・・・　センサの状態を知ってから，モータ制御データを出力するまでの時間的なズレ（遅れ）のこと

＜「**タイムラグ**」が大きくなる原因とは，何か？＞

手順１（センサ入力）と手順３（モータ出力）の間に，
手順２（制御データの作成）があるため

［考察］「タイムラグ」を小さくするには，どうすればいい？

＜答え＞　・・・・　**魔法の引き出し**（＝「**配列**」）を使う

センサ入力とモータ出力の関係を，**魔法の引き出し**（＝「**配列**」）の名前と中身の関係に当てはめると，以下で示すように，手順２が省略できるのだ。

＜センサ入力とモータ出力の関係＞

センサ入力	0	1	2	3	4	5	6	7
モータ出力	4	5	6	7	4	5	6	7
	⇩	⇩	⇩	⇩	⇩	⇩	⇩	⇩
引き出しの名前	D(0)	D(1)	D(2)	D(3)	D(4)	D(5)	D(6)	D(7)
引き出しの中身	4	5	6	7	4	5	6	7

※引き出しの名前（「配列」の各要素）は，先頭からのズレ（オフセット）で指定する。

＜プログラム化＞

手順１　ラインの検出（センサからの入力）　⇨　「ＩＮＰ命令」

手順２　・・・・　制御データの作成は，不要

手順３　配列データの出力（モータへの出力）　⇨　「ＯＵＴ命令」

手順４　手順１へ戻る

※配列データの出力とは，

D(0) を指定すると，　4　が出力される。

↑　引き出しの名前＝「センサ入力」

↑　引き出しの中身＝「モータ出力」

＜「配列」（＝魔法の引き出し）を用いると，＞

手順２（制御データの作成）がなくなるので，手順１（センサ入力）と 手順３（モータ出力）の間の「タイムラグ」が小さくなる。

○「配列」を使用する時は，下記のような前処理が必要。

```
前処理①      10  ＤＩＭ    D( ７ )
前処理②      20  D( ０ )  ＝  ４
 〃           30  D( １ )  ＝  ５
 〃           40  D( ２ )  ＝  ６
 〃           50  D( ３ )  ＝  ７
 〃           60  D( ４ )  ＝  ４
 〃           70  D( ５ )  ＝  ５
 〃           80  D( ６ )  ＝  ６
 〃           90  D( ７ )  ＝  ７  ・・・・ 100 行目に続く
```

※10 行目の説明：

ＤＩＭ　Ｄ（７） の配列宣言で，８個のメモリ（記憶場所）が確保される。

⇧ 最後の要素名で，配列の大きさを指定

⇩ センサの状態（情報量）は，$2^3=8$

問題 前処理の続きを作ってみよう。　　ヒント：poke6 を実行してみよう

（手順２は省略）

```
手順１      100  ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿
手順３      110  ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿
手順４      120  ＧＯＴＯ    100
```

※自分の「ポケコン」と「搬送車」で試してみよう。

◎しかし，「配列」を用いても，「タイムラグ」を完全に無くすことはできない。
確実に「ライントレース」制御するためには，モータのスピードを遅くする必要がある。（その方法は，次の章で詳しく説明するよ！）

→ とりあえず，「ブレーキ」を入れて実験：どこに入れる？＿＿＿＿＿行目

（それでも，「ライントレース」制御しないときは，何か重い物を乗せてみよう。）

気付いたことをまとめよう。

7　モータの速度制御　～確実に「ライントレース」制御するために～

○確実に「ライントレース制御」するためには，モータのスピードを遅くする必要がある。以下に，その方法を示す。

（1）ギアボックスのギア比を変える

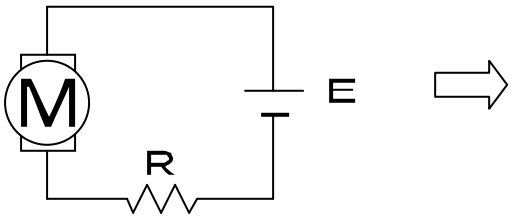
ギア比	64.8：1	41.7：1
最高効率時の 1 分間の出力軸の回転数［rpm］	約１６０	約２５０
最高効率時の出力軸の回転トルク［g/cm］	約１０４０	約６７０

※速度の遅いほうが，「タイムラグ」が小さくなる。

（2）ポケコン搬送車を改造する

・・・・モータに流す電流を変えて，モータの速度を変える

☆「オームの法則」より
電流（Ｉ＝Ｅ／Ｒ）を変えるには，「電源電圧 Ｅ を変える」か，
「（可変）抵抗 Ｒ を［電源］と［モータ］の間に直列に入れる」。

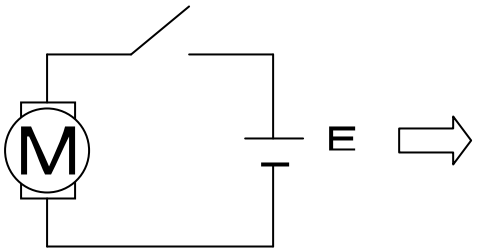


電源電圧 Ｅ を変える
↓
電池ボックスの接続を「直列（６ｖ）」から「並列（３ｖ）」に変える。

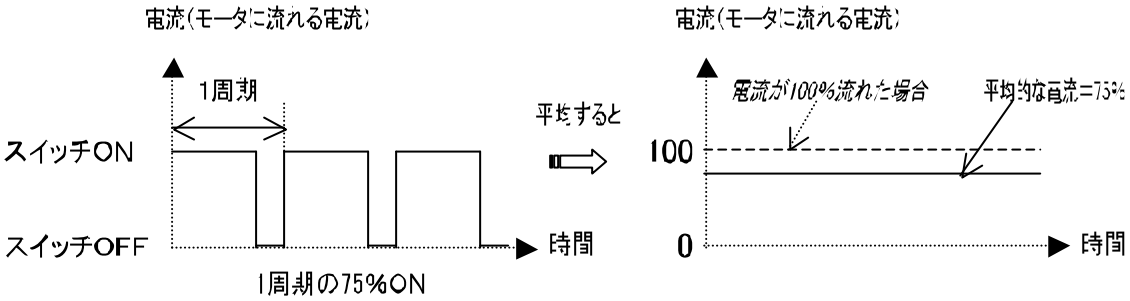
（可変）抵抗 Ｒ を入れる（リニア制御）
・・・・抵抗でエネルギーの一部が熱に変わる。

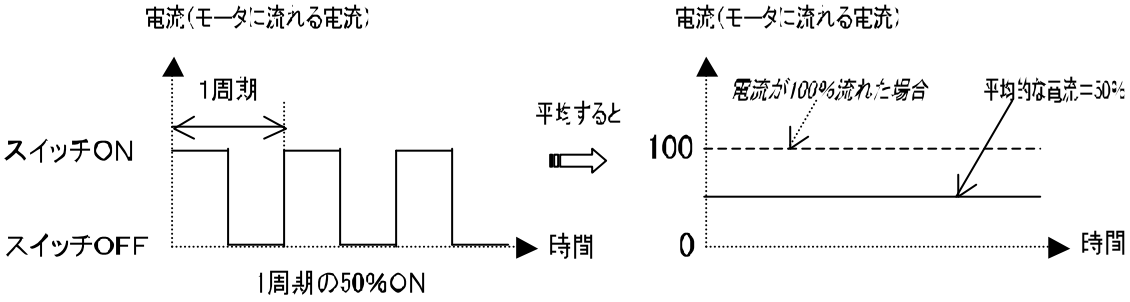
※（1）の方法も，（2）の方法も有効な手段ではあるが，モータの速度を「プログラム」で制御することはできない。

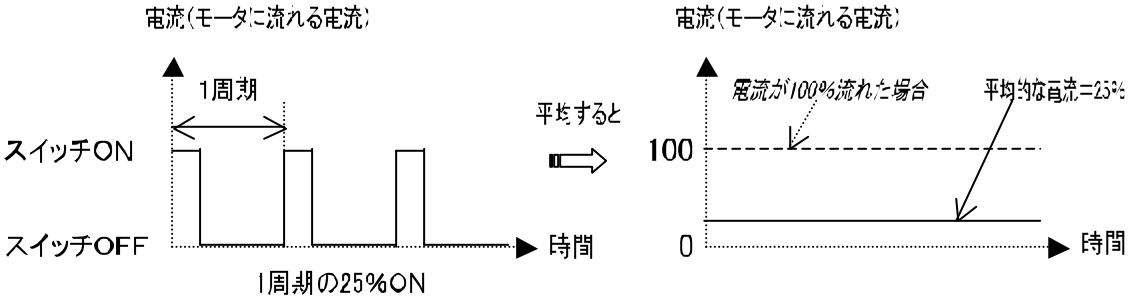
（３）パルス幅変調方式（ＰＷＭ制御）



（考え方）
スイッチを入れる時間を変えることで，モータに流す電流（平均電流）を変えて，モータの速度を変える。
↓
※「抵抗のようなエネルギーの無駄はない」







◎スイッチのＯＮ／ＯＦＦは，「プログラム」で制御することができる。
↓
ということは，「ＰＷＭ制御」行うプログラムを作成することができる。
↓
（この場合，搬送車の改造を行う必要はない）

★ 「モータの速度制御」を行うプログラムを作ろう（ＢＡＳＩＣによる制御）

【プログラムによるPWM制御（考え方)】

○「スイッチＯＮの時間」と「スイッチＯＦＦの時間」の単位時間（＝長さ１）を同じにする。

①「スイッチＯＮの時間」（単位時間）の定義

ON-1	Ｘ ＝ ＩＮＰ	・・・・	ラインの検出
ON-2	ＯＵＴ Ｄ(Ｘ)	・・・・	制御データの出力
ON-3	ＰＲＩＮＴ Ｘ	・・・・	センサ入力の表示

※上の手順は,「ライントレース」制御するために，最低限必要なことである。

②「スイッチＯＦＦの時間」（単位時間）の定義

OFF-1	Ｘ ＝ ＩＮＰ	・・・・	ラインの検出
OFF-2	ＯＵＴ ７	・・・・	ブレーキの出力
OFF-3	ＰＲＩＮＴ Ｘ	・・・・	センサ入力の表示

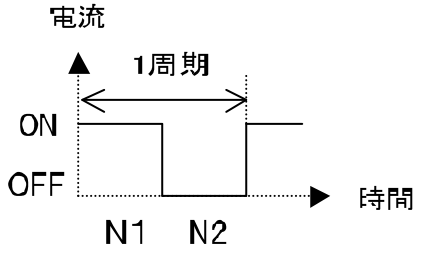
＜プログラム化＞　　～プログラムによるPWM制御～

手順１　下図のPWM制御のパラメータ（Ｎ１とＮ２）を決める

手順２　①「スイッチＯＮの時間」(ON-1 → ON-2 → ON-3)
を Ｎ１回 繰り返す

手順３　②「スイッチＯＦＦの時間」(OFF-1 → OFF-2 → OFF-3)
を Ｎ２回 繰り返す

手順４　手順２へ戻る



問題 次のプログラムを完成させよう。ヒント：poke7 を実行してみよう

```
                5 ‘ＳＡＶＥ “ＰＷＭ－Ｂ”←  この名前で保存しよう

前処理①        10  ＤＩＭ    Ｄ( ７ )
前処理②        20  Ｄ( ０ )  ＝  ４
  〃            30  Ｄ( １ )  ＝  ５
  〃            40  Ｄ( ２ )  ＝  ６
  〃            50  Ｄ( ３ )  ＝  ７
  〃            60  Ｄ( ４ )  ＝  ４
  〃            70  Ｄ( ５ )  ＝  ５
  〃            80  Ｄ( ６ )  ＝  ６
  〃            90  Ｄ( ７ )  ＝  ７

手順１          100  ＩＮＰＵＴ  “ｽｲｯﾁ ON=” ; Ｎ１
                110  ＩＮＰＵＴ  “ｽｲｯﾁ OFF=” ; Ｎ２

手順２          120  ＦＯＲ    Ｔ＝１    ＴＯ    Ｎ１

                130  ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

                140  ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

                150  ＮＥＸＴ    Ｔ

手順３          160  ＦＯＲ    Ｔ＝１    ＴＯ    Ｎ２

                170  ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

                180  ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

                190  ＮＥＸＴ    Ｔ

手順４          200  ＧＯＴＯ    120
```

実験１　「スイッチＯＦＦ」を１に固定して,「スイッチＯＮ」を１,２,３,･･･と大きくしていくと,「ポケコン搬送車」の動きはどう変わるか？

気付いたことをまとめよう。

実験２　「スイッチＯＮ／ＯＦＦ」ともに１，ともに２，ともに３,･･･と大きくしていくと,「ポケコン搬送車」の動きはどう変わるか？

気付いたことをまとめよう。