

平成27年度 東京工業大学ロボット技術研究会 回路講習会④

日時:平成27年5月28日(木) 18:00~19:30

場所:S221

講義担当者:けり @Ryokeri14

第4回 全内容

1. [宿題\(加算器\)の解説](#)
2. [AD\(ADC\)変換とは](#)
3. [AD変換を使う](#)
4. [AD変換の性能](#)
5. [角度検出](#)
6. [可変抵抗](#)
7. [可変抵抗の種類](#)
8. [可変抵抗で回転角度検出](#)
9. [回転数検出](#)
10. [ロータリーエンコーダ](#)
11. [ジャイロ, 加速度, 地磁気センサ](#)
12. [障害物を見つける, 距離を測る](#)
13. [フォトランジスタ](#)
14. [赤外線センサ](#)
15. [フォトリフレクタ](#)
16. [PSDセンサ\(距離センサ\)](#)
17. [超音波センサ](#)
18. [マイクロスイッチ](#)
19. [ノイズを減らすフィルタ回路](#)
20. [フィルタ回路の種類](#)
21. [ローパスフィルタ](#)
22. [ハイパスフィルタ](#)
23. [オペアンプによる增幅回路](#)
24. [オペアンプ](#)
25. [オペアンプの機能](#)
26. [理想的なオペアンプ](#)
27. [反転増幅回路](#)
28. [非反転増幅回路](#)
29. [ボルテージフォロア](#)
30. [入力インピーダンス](#)
31. [出力インピーダンス](#)
32. [オペアンプについて](#)
33. [おすすめのオペアンプ](#)
34. [オペアンプとトランジスタ](#)
35. [電流を検出する](#)
36. [ライントレース](#)
37. [ライントレース\(直進\)](#)
38. [制御](#)
39. [ライントレース\(旋回\)](#)
40. [ロボコンで使う回路](#)
41. [メッセージ](#)

第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

← 目次に戻る

*クリックするとそのページにジャンプします。

第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

← 目次に戻る

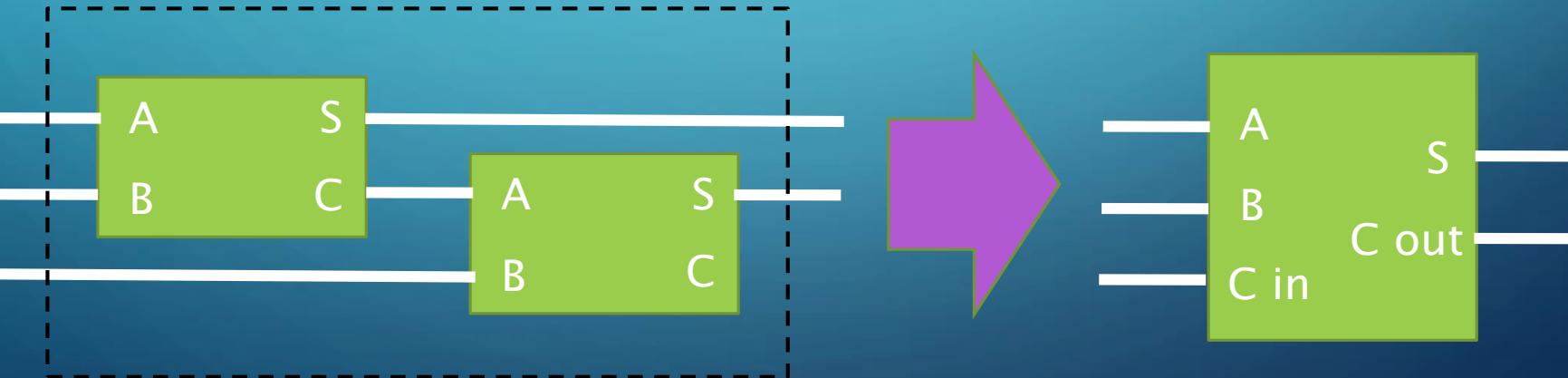
*クリックするとそのページにジャンプします。

1. 半加算器と全加算器
2. N bit加算器

宿題の解説

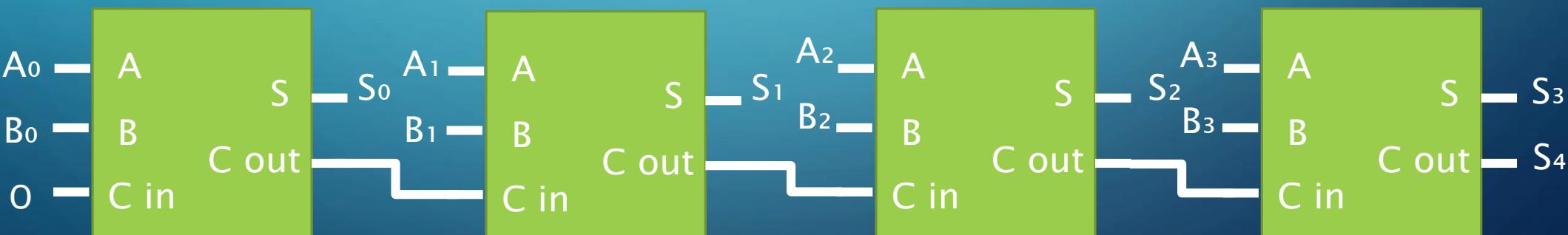
- ・半加算器は、2入力の加算器。
- ・全加算器は、3入力の加算器。

半加算器2つで、全加算器を作れる



全加算器

- 全加算器は、n個用意すると、n bit加算器ができる。
- 例：4bit加算器
- 2つの2進数「 $A_3A_2A_1A_0$ 」と「 $B_3B_2B_1B_0$ 」を入れると「 $S_4S_3S_2S_1S_0$ 」が出てくる。
- 例えば「1010」と「0011」を入れると、「01101」が出てくる。
 - 出力は1桁多くなる



第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

1. ADCとは
2. AD変換
3. AD変化を使う
4. AD変換の性能

← 目次に戻る

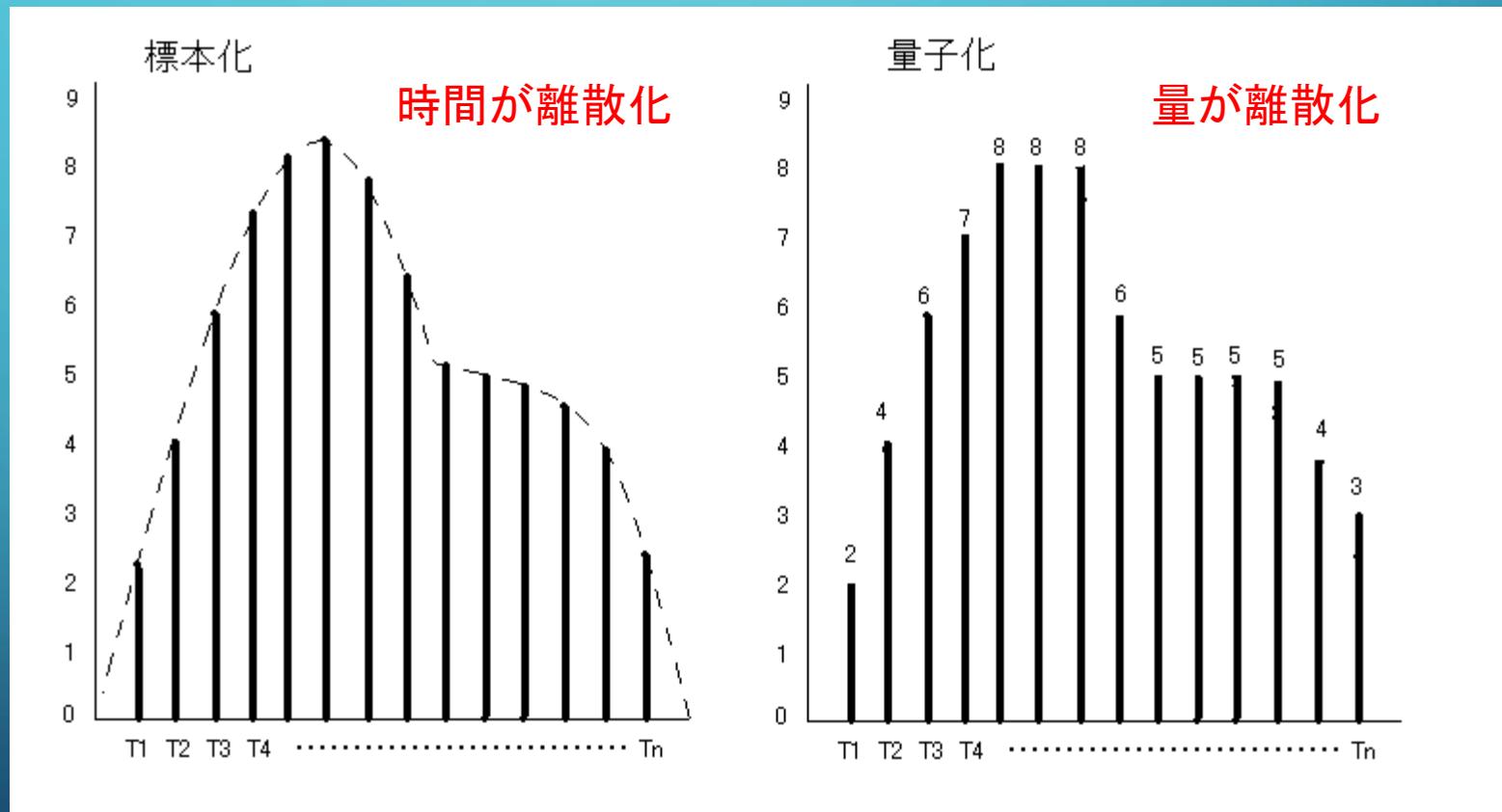
*クリックするとそのページにジャンプします。

AD変換(ADC)とは

- アナログデータをデジタルデータに変換する.
 - マイコンはデジタルデータしか扱えないから.
- 具体的には、センサなどが出力する電圧を測って、数値に変えること.
- AD変換は人がするのではなく、機械にやらせる。

AD変換

- AD変換すると時間と量が離散化する。



AD変換

- 標本化(サンプリングともいう)→**時間が離散化**
 - 標本化とは一定間隔でデータを採取すること. 休みなしにデータをとり続けることはできない. サンプリングは早い方が正確だが情報量が増える.
- 量子化→**量が離散化**
 - 量子化とは読み取ったデータに数値を当てはめること. ぴったり合う数値がなければ近い数字に丸め込むのでズレが生じる.
 - 量子化の単位としては, bit数を目安にする.
例: 10bit AD変換

AD変換を使う

- AD変換ICを使う
 - 精度の高いものを選ぶことができる.
- マイコン内蔵のAD変換器を使う
 - 使うのが容易

AD変換を使う

- AD変換には基準電圧が必要
 - これを, V_{ref} という \leftarrow Voltage Reference
- 8bit AD変換器の場合
 - V_{ref} を8bit等分, つまり256等分する.
 - 測った電圧が256分のいくつかを, 出力する.

AD変換の性能

- 8bitで0V～5VをAD変換するとき....
 - 0V～5Vが256段階に離散化される.

$$\frac{5}{256} \cdot 255[V] \rightarrow 11111111$$

$$\frac{5}{256} \cdot 254[V] \rightarrow 11111110$$

.....

$$\frac{5}{256} \cdot 1[V] \rightarrow 00000001$$

$$0[V] \rightarrow 00000000$$

実際には

$$0[V] \sim \frac{5}{256} \cdot 1[V]$$

の範囲が入力された時に
00000000が得られる.

必ず誤差が出る

第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

目次に戻る

*クリックするとそのページにジャンプします。

1. 角度検出
2. 可変抵抗
3. 可変抵抗の種類
4. 可変抵抗で回転角度検出

角度検出

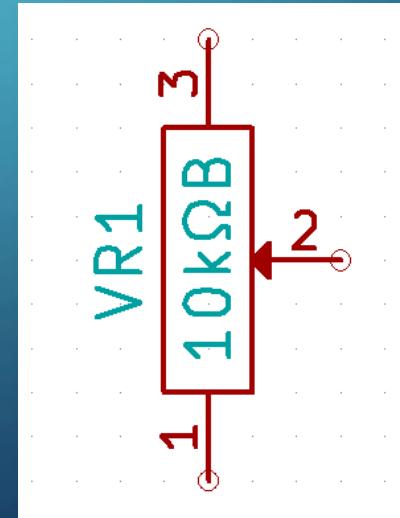
- 腕の角度や、舵の方向を知りたいときに使う.
- 可変抵抗(ポテンショメータ)とAD変換器を使う.

可変抵抗(ポテンショメータ)

- 三本足の部品で、つまみを回すことで抵抗値を変えられる。
- 1回転のものから何十回転するものまであるが、機能は同じ
- 回路図の2番ピンが上下に動くイメージ。
 - 1~3間の抵抗は常に一定。
 - 1~2の抵抗値 + 2~3の抵抗値 = 1~3の抵抗値

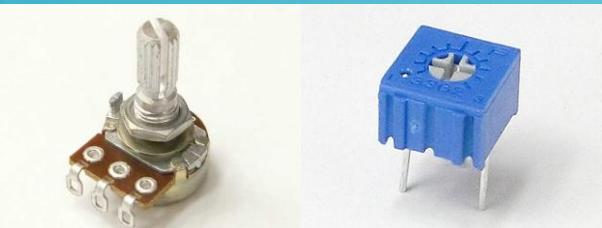


回路図

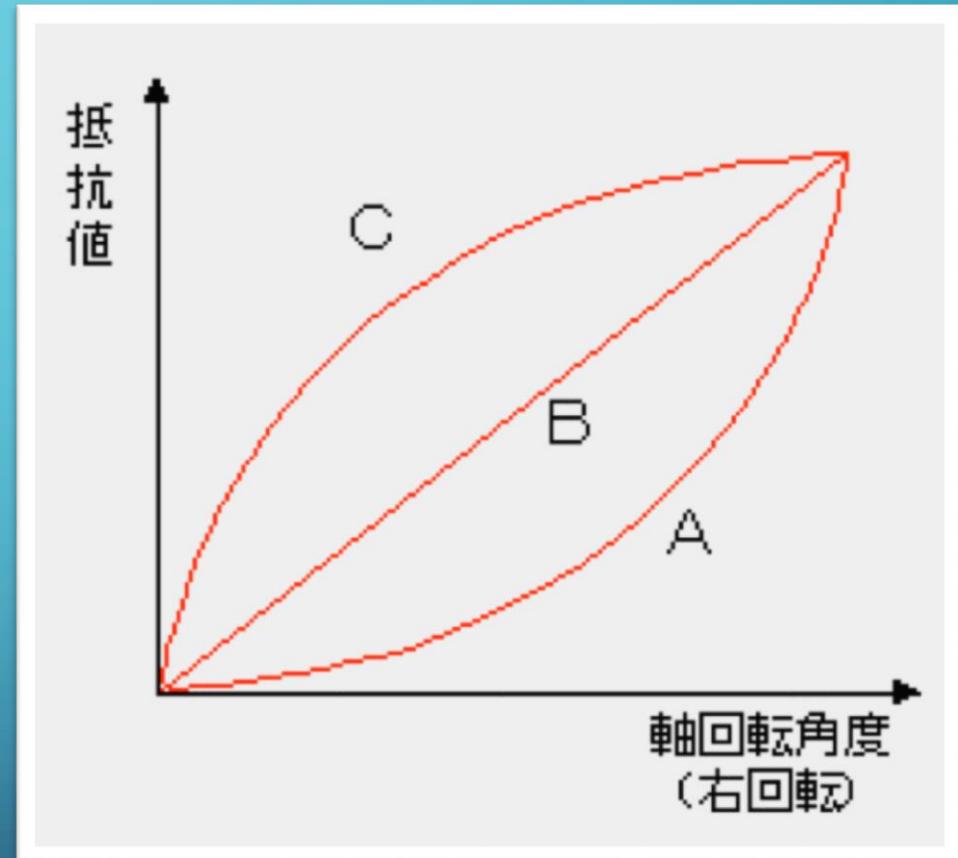


可変抵抗

- 手で回すものをボリューム、ドライバで回すものを半固定抵抗という。



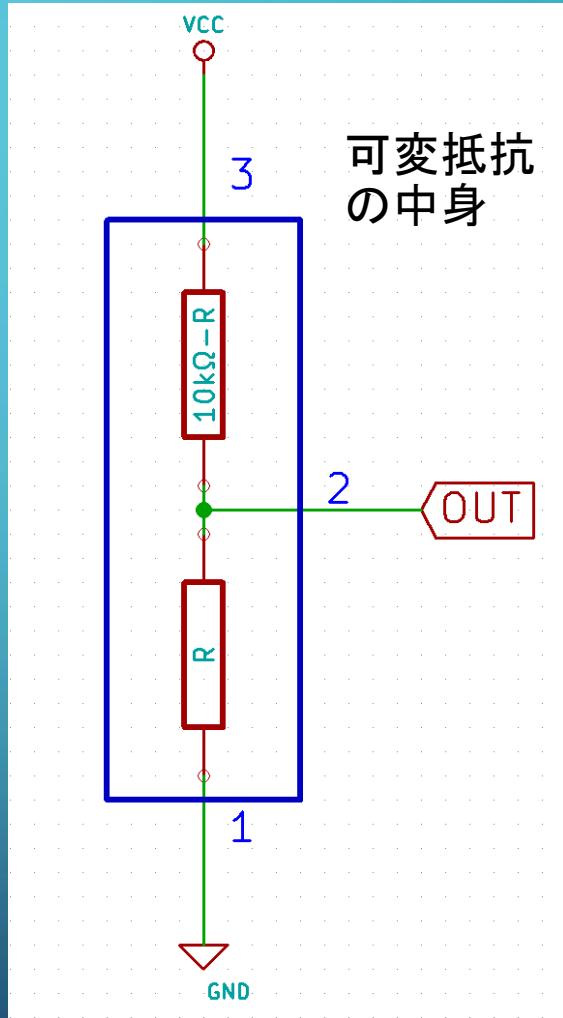
- 抵抗が曲線変化するものもある
 - Aカーブ: 音響関係で使う
 - Bカーブ: ポテンショメータとして使う
 - Cカーブ: ほとんど使わない



可変抵抗で回転角度検出

- 1番ピンをGND, 3番ピンをVCCにつないで, 2番ピンをAD変換器につなぐ.
- 抵抗値を測るのではなく, 電圧を測って角度を検出する.

$$OUT \text{ [V]} = VCC \times \frac{\text{回転角}}{\text{全回転角}} \text{ [V]}$$



第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

目次に戻る

*クリックするとそのページにジャンプします。

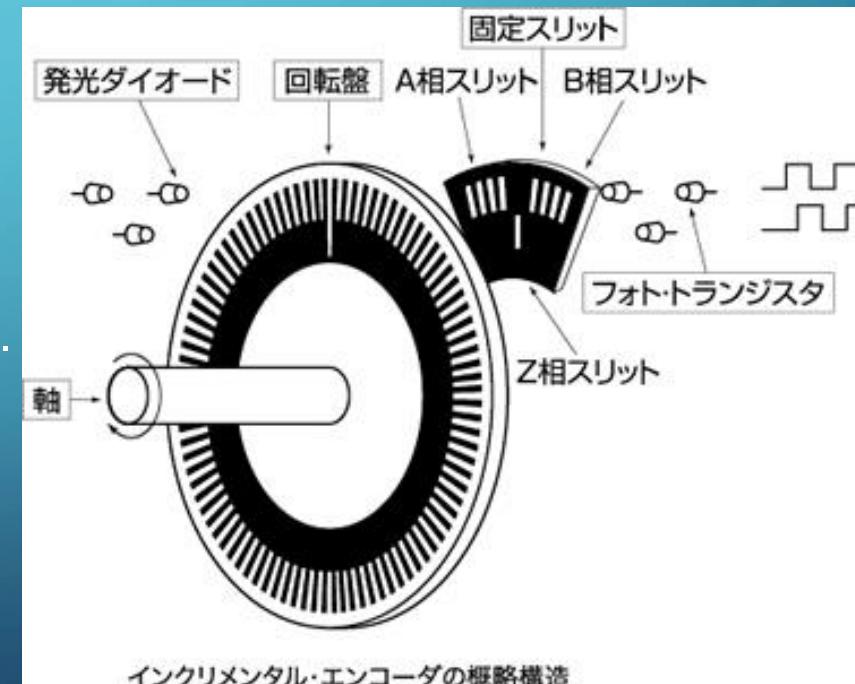
1. 回転数検出
2. ロータリーエンコーダ
3. ジャイロ, 加速度, 地磁気センサ

回転数検出

- タイヤの回転数を知りたい
- 可変抵抗だと回転数が固定されている
- 無限回転のロータリーエンコーダを使う

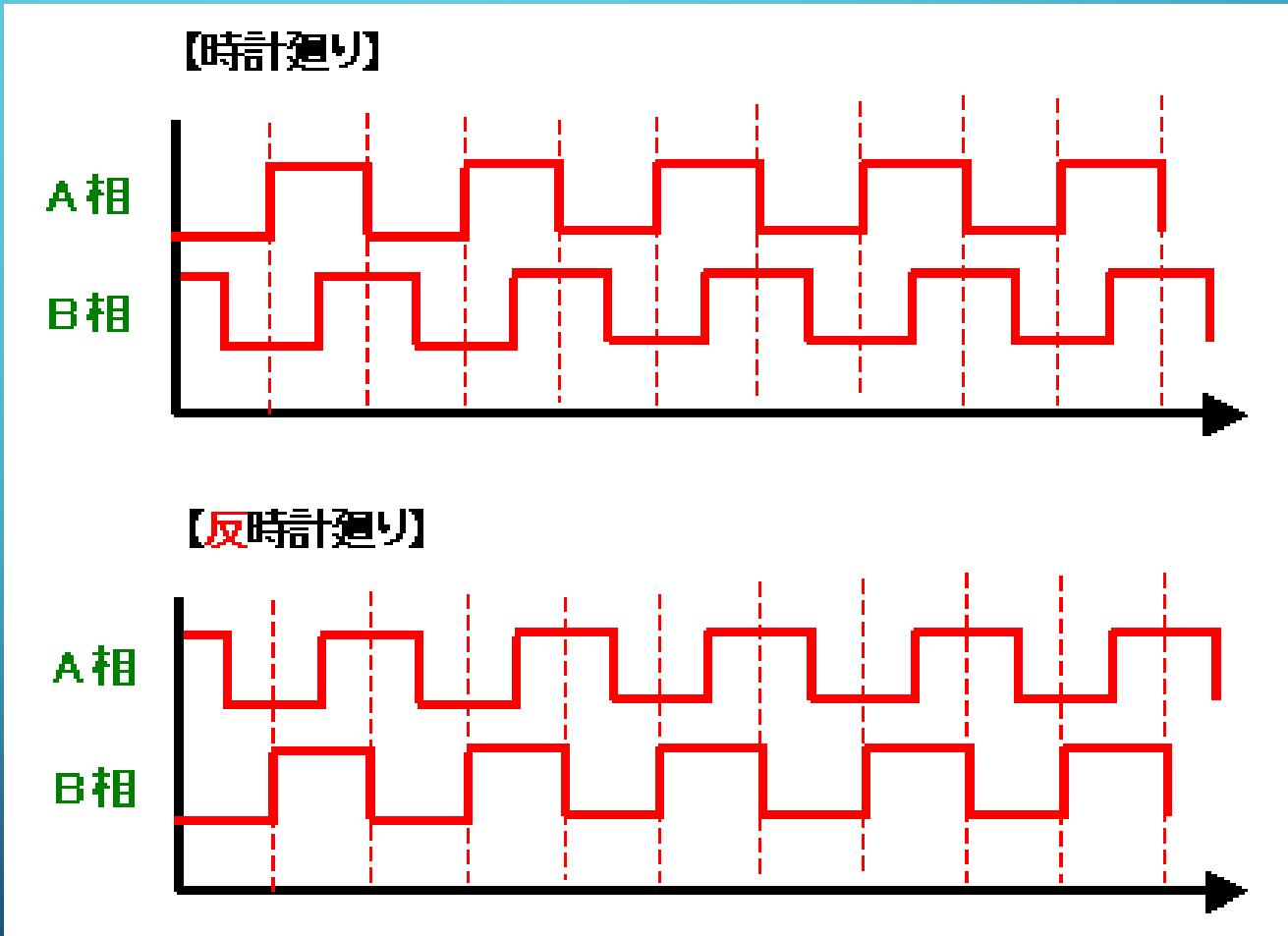
ロータリーエンコーダ

- 光学式, 磁気式, 機械式がある.
- 出力はA相, B相の2端子.
 - それぞれからパルス波が出てくる.
 - 2つの位相のずれにより回転方向がわかる.
 - 光学式の場合, 回転円盤が光を遮ることでパルス波が出てくる.



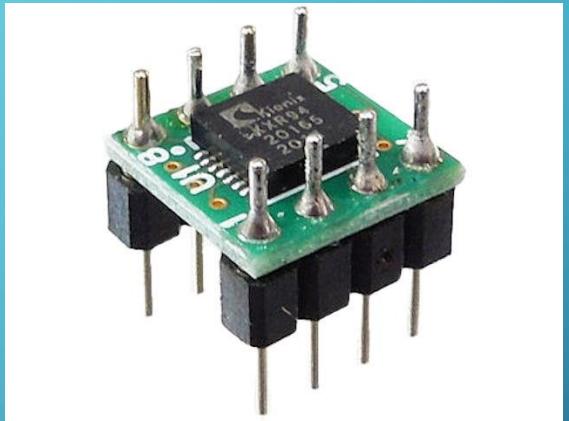
ロータリーエンコーダ

- A相とB相の波形
- ソフトウェアでこれを解釈して、回転方向と速度を得る。



ジャイロ, 加速度, 地磁気センサ

- ジャイロセンサ(角速度センサ)
 - 回転を検出
 - 加速度センサ
 - 移動を検出
 - 地磁気センサ
 - 方位を検出
-
- 3軸センサ: ジャイロ(x,y,z)
 - 6軸センサ: ジャイロ(x,y,z), 加速度(x,y,z)
 - 9軸センサ: ジャイロ(x,y,z), 加速度(x,y,z), 地磁気(x,y,z)



加速度センサモジュール

第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

目次に戻る

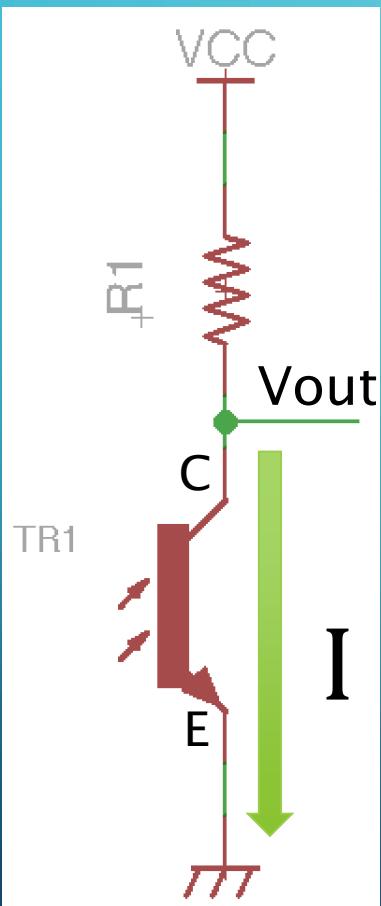
*クリックするとそのページにジャンプします。

1. 障害物を見つける, 距離を測る
2. フォトトランジスタ
3. フォトリフレクタ
4. PSDセンサ
5. 超音波センサ
6. マイクロスイッチ

障害物を見つける, 距離を測る

- 赤外線センサ(フォトリフレクタ)を使う
 - ライントレース
 - 測距
- 超音波センサを使う
 - 測距
- マイクロスイッチ
 - ぶつかると反応

フォトトランジスタ(光センサ)



フォトトランジスタは、
光が入るとC→Eに電流が流れる。

$$V_{out} [V] = V_{CC} - R_1 \cdot I [V]$$

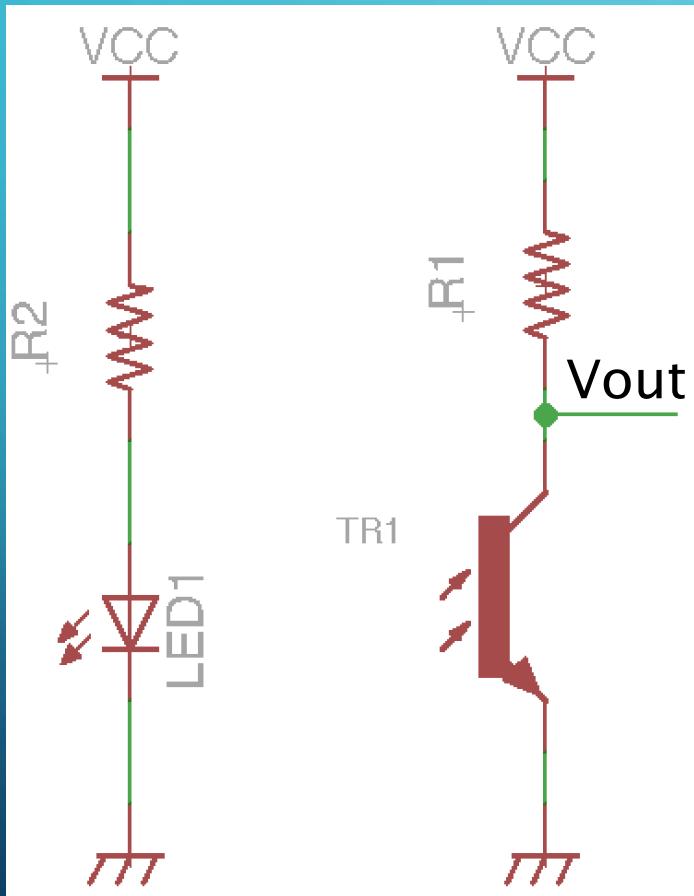


赤外線センサ(フォトリフレクタ)

- 赤外線を当てて、光が跳ね返ってくるかを調べる。
- 黒ならば、光が吸収されてしまう返って来ない。
- 白ならば、光が返ってくる。
- 赤外線センサで読み取る。



フォトリフレクタ



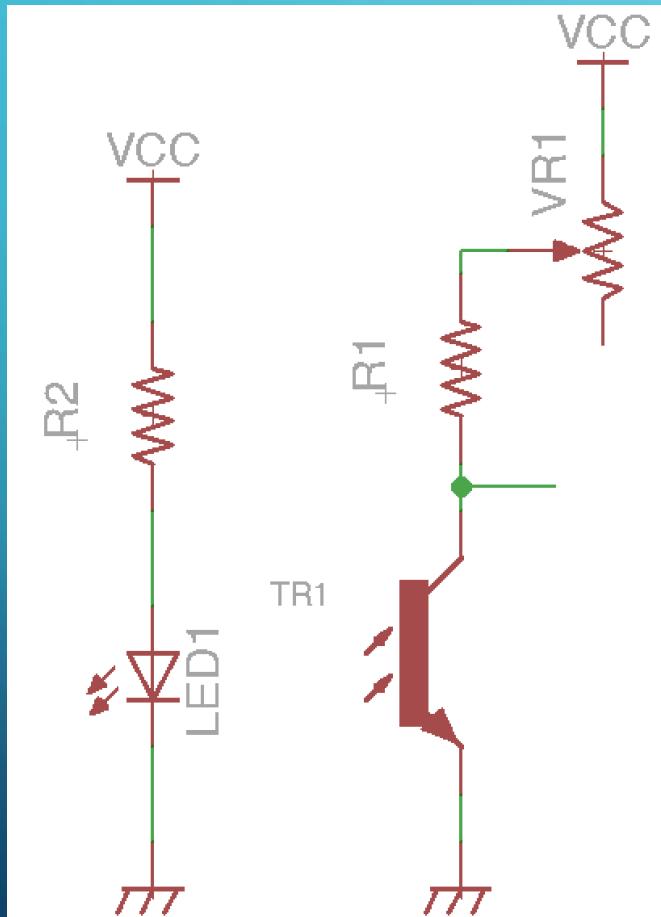
白っぽい物があるとき.....

- 1 光が多く反射する
- 2 フォトトランジスタに
多く電流が流れる
- 3 R1での電圧降下が大きくなる
- 4 Voutが小さくなる

黒っぽい物があるとき.....

- 1 光があまり反射しない
- 2 フォトトランジスタに
あまり電流が流れない
- 3 R1での電圧降下が小さくなる
- 4 Voutが大きくなる

フォトリフレクタ



- 実際には固定抵抗と可変抵抗を入れる.
- VR1とR1はいろいろ実験していい値を見つける.
- フォトトランジスタには $10\mu\text{A} \sim 100\mu\text{A}$ くらいのオーダーの電流が流れる.
- $(R1 + VR1)$ は $10\text{k}\Omega$ のオーダーだと出力電圧がはっきりと変化する. (経験上)

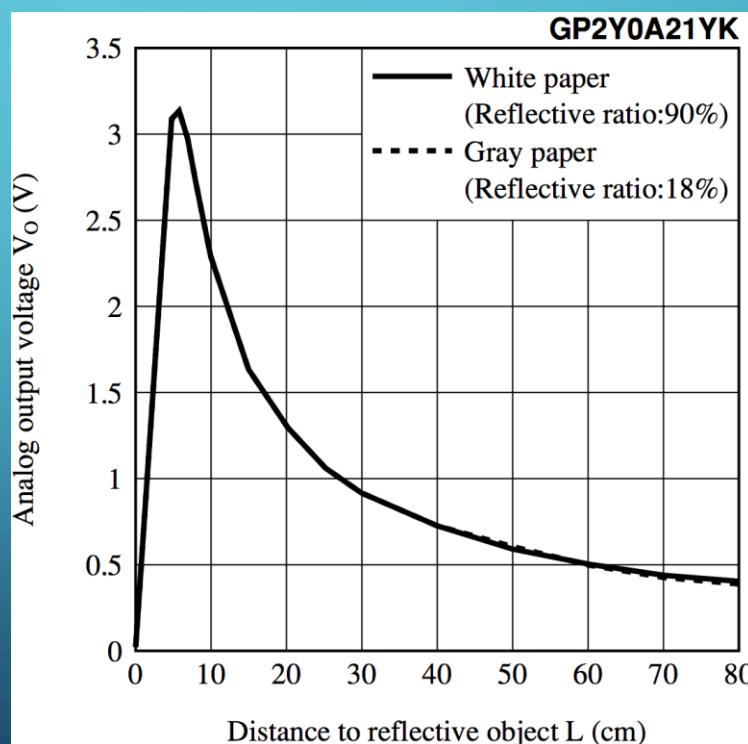
例えば $R1 = 10\text{k}$, $VR1 = 50\text{k}$ とか

PSDセンサ(測距センサ)

- Position Sensitive Detector
- 10cm～1mくらいが測れる
- 距離に応じて出力電圧が
アナログ的に変化
→ADCを使う



例:GP2Y0A21YK
(秋月に売ってる)



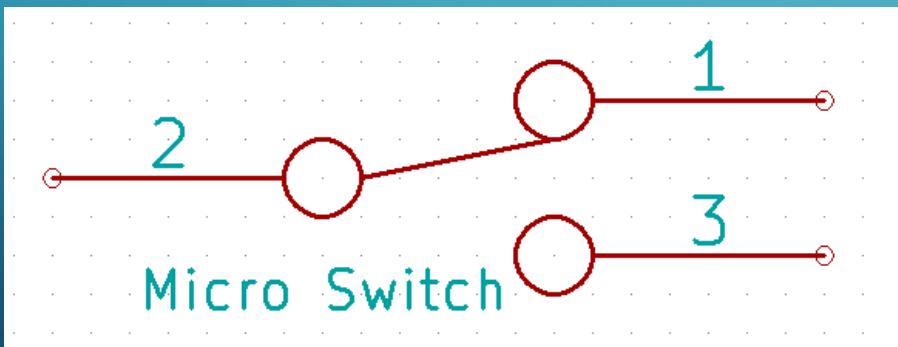
超音波センサ

- 赤外線センサより精度が高い.
- 波を放ってから返ってくるまでの時間の差を測り, 距離を求められる.



マイクロスイッチ

- スイッチで衝突などを検出できる
 - マイクロスイッチがよく使われる
- 3本足なので、押すとオン、押すとオフの両方が使える。



第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

目次に戻る

*クリックするとそのページにジャンプします。

1. ノイズを減らすフィルタ回路
2. フィルタ回路の種類
3. ローパスフィルタ
4. ハイパスフィルタ

ノイズを減らすフィルタ回路

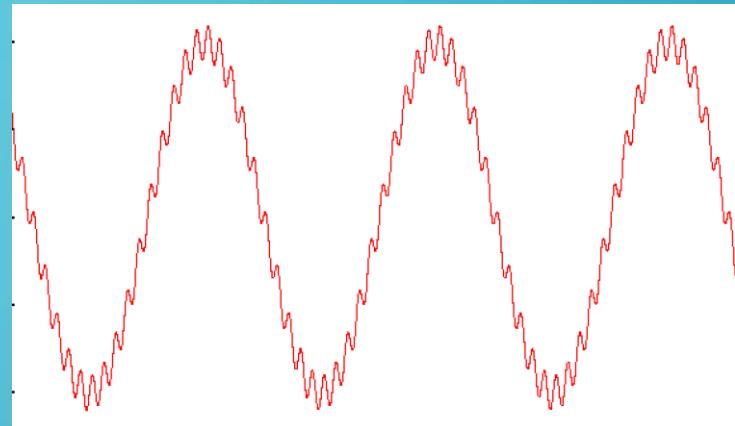
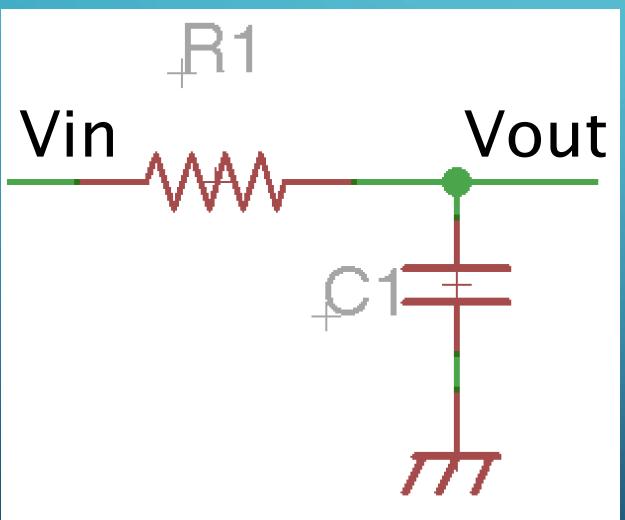
- センサなどから採取したデータは、ノイズが多い
- 1つの波に、さまざまな周波数の波が混ざっている。
- 周波数の違いを利用してノイズを減らす
→ フィルタ回路

フィルタ回路の種類

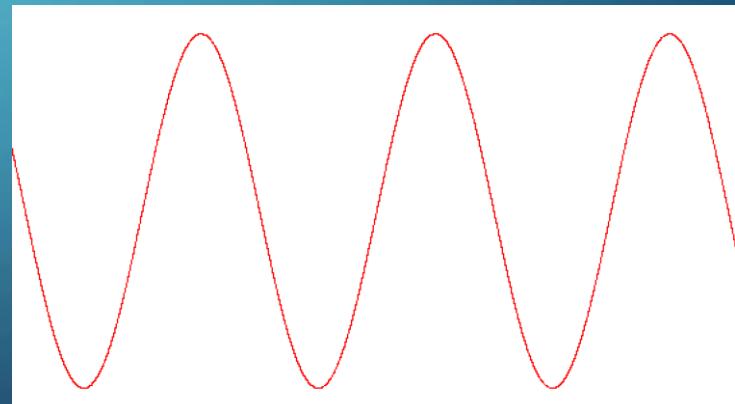
- ローパスフィルタ
 - 低周波をパス, 高周波をカット
- ハイパスフィルタ
 - 低周波をカット, 高周波をパス
- バンドパスフィルタ
 - 高周波と低周波をカットして, 欲しい周波数だけGET

ローパスフィルタ(LPF)

- 低い(LOWな)周波数成分は通過(PASS)させる.
- 高い(HIGHな)周波数成分は、減衰させる.



↓ LPF



ローパスフィルタ

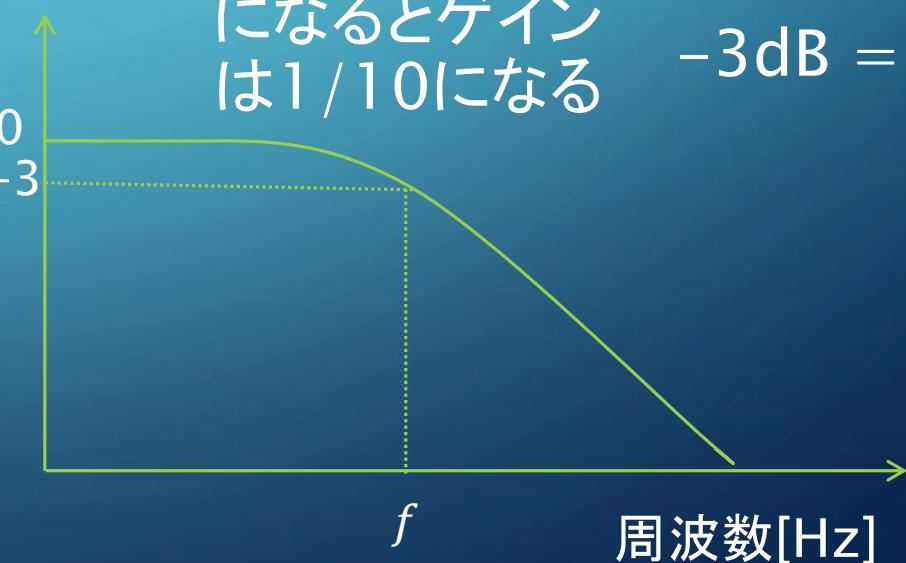
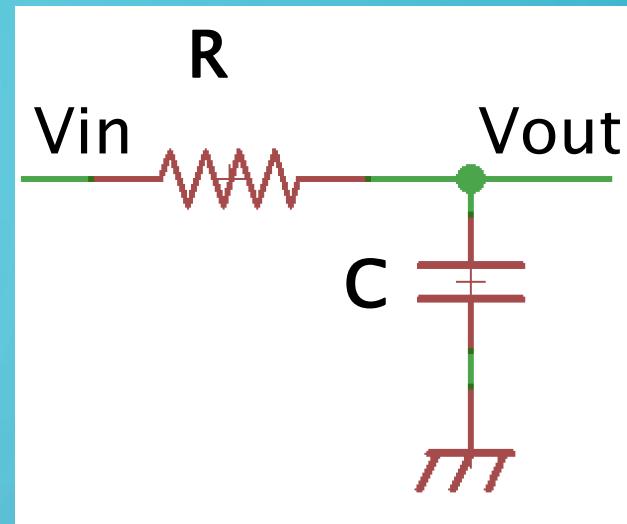
- R と C の決め方
 - 減衰させる周波数で決定する.
- $f = \frac{1}{2\pi RC}$ [Hz]

縦軸・横軸ともに
対数目盛り

ゲイン[dB]

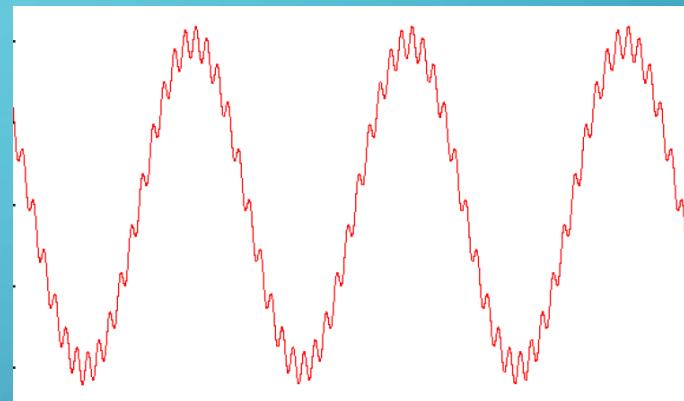
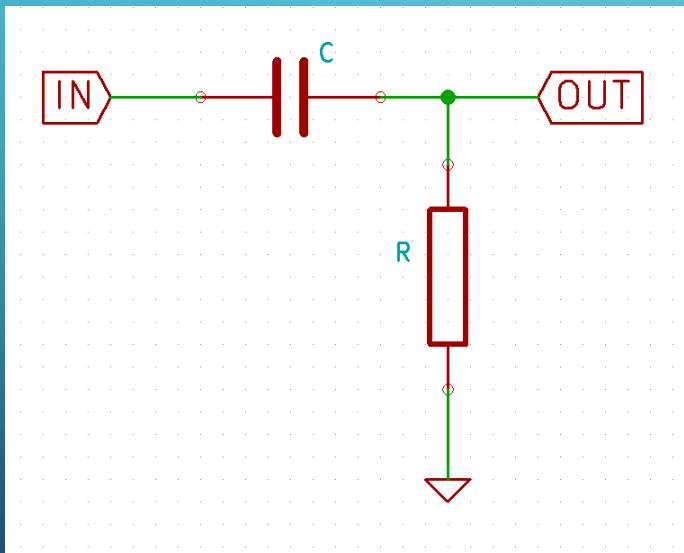
周波数が10倍
になるとゲイン
は1/10になる

$$\begin{aligned}0\text{dB} &= 1\text{倍} \\ -3\text{dB} &= \frac{1}{\sqrt{2}}\end{aligned}$$

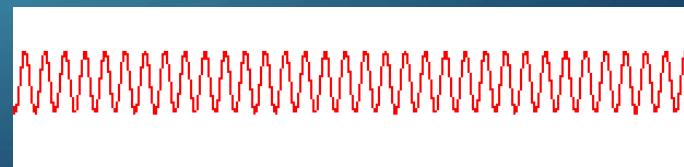


ハイパスフィルタ(↔ローパスフィルタ)

$$f = \frac{1}{2\pi RC} [\text{Hz}]$$



HPF



第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

目次に戻る

*クリックするとそのページにジャンプします。

1. オペアンプによる増幅回路
2. オペアンプ
3. オペアンプの種類
4. 理想的なオペアンプ
5. 反転増幅回路
6. 非反転増幅回路
7. ボルテージフォロア
8. 入力インピーダンス
9. 出力インピーダンス
10. オペアンプについて
11. おすすめのオペアンプ
12. オペアンプとトランジスタ

オペアンプによる増幅回路

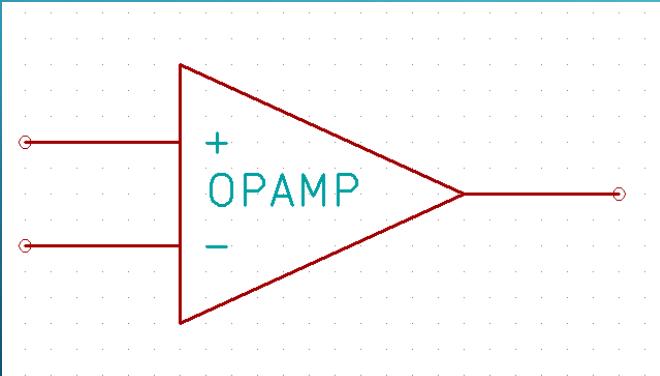
- ・センサの讀取値などは微小な電圧である.
- ・マイコン内蔵のAD変換器では読み取りにくい.
→オペアンプで電圧を増幅する

- ・抵抗分圧では電流を確保できない.
- ・電圧は変えずに電流を流せるようにしたい.
→ボルテージフォロア

オペアンプ

- 基本回路は三本足. 左の2つが入力で, 右の1つが出力.
- 入力端子には非反転入力(+)と反転入力(−)がある.
- 電源電圧範囲外の電圧は出力できない←注意

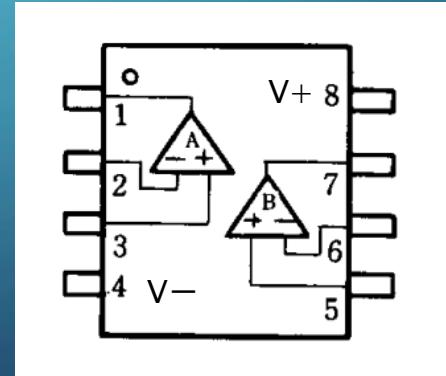
回路図



パッケージ



2回路入っている

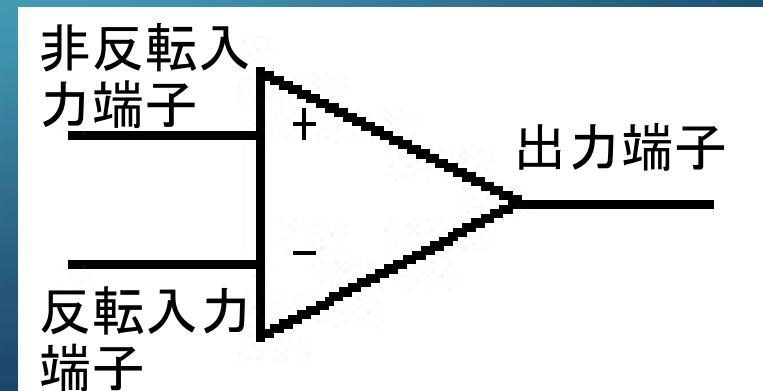


オペアンプの機能

- 非反転増幅回路
- 反転増幅回路
- 加算回路
- 減算回路
- 微分回路
- 積分回路
- コンパレータ

理想的なオペアンプ

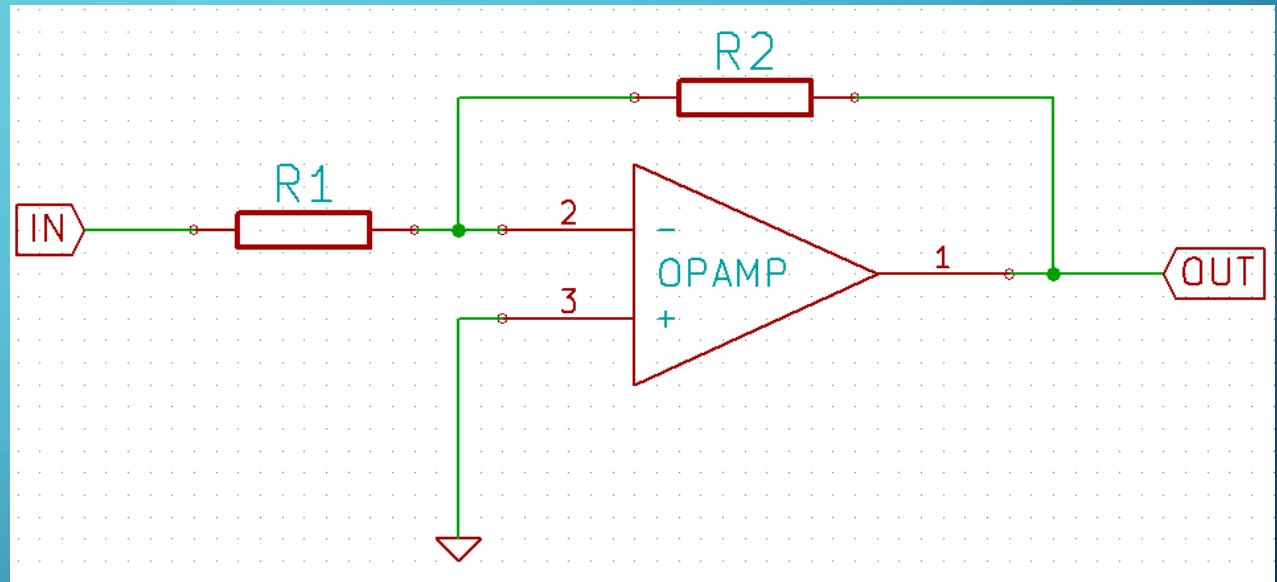
- 非反転入力端子と反転入力端子は同電位になる.
- 非反転入力端子と反転入力端子のインピーダンス(抵抗)は無限大で、電流が流れ込まない.
- 非反転入力と反転入力の差が増幅される.
- 何もつながないと倍率は無限大.
→フィードバックして倍率を一定にする



反転増幅回路

- オペアンプの基本回路
- 増幅倍率kは

$$k = -\frac{R2}{R1} \text{ (負の数)}$$



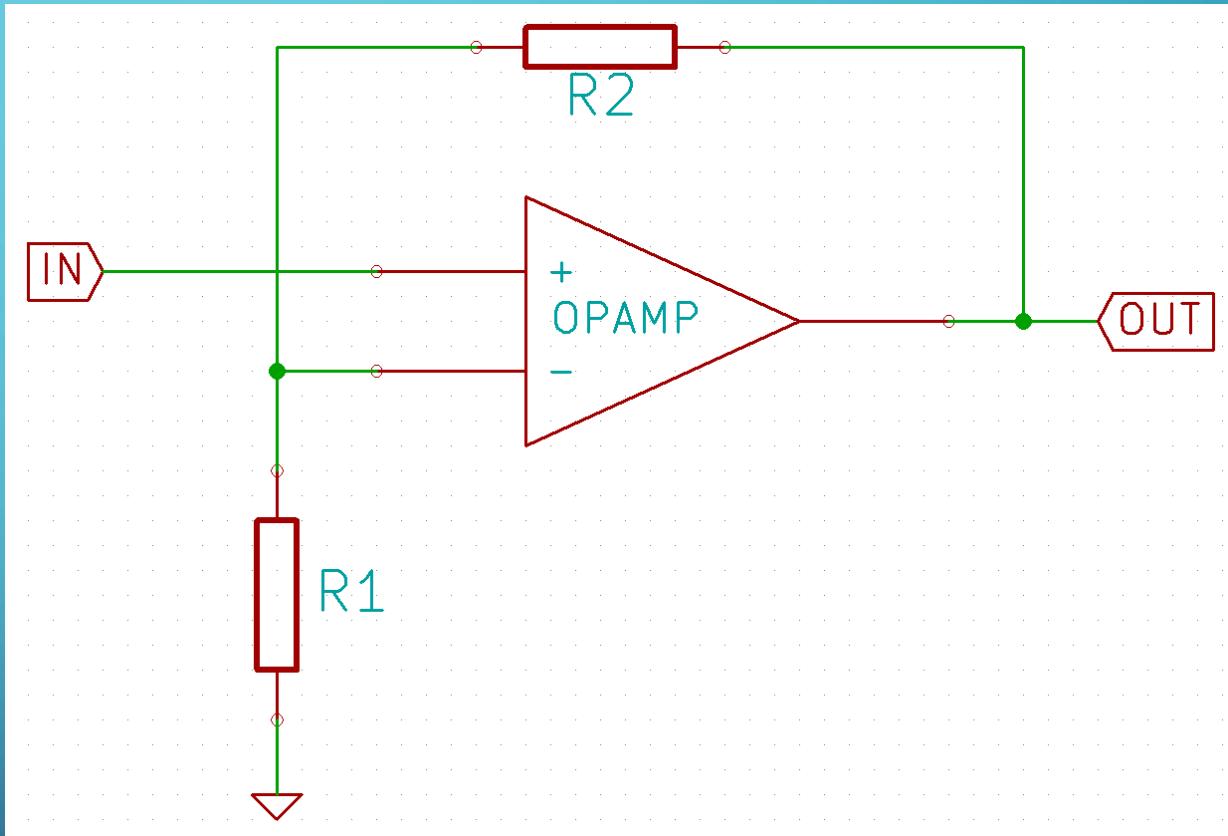
電源は+，−，GNDの両電源が必要→使いにくい

非反転増幅回路

- 一番よく使う増幅回路.
- 比例して電圧を増幅する.
- 増幅倍率kは

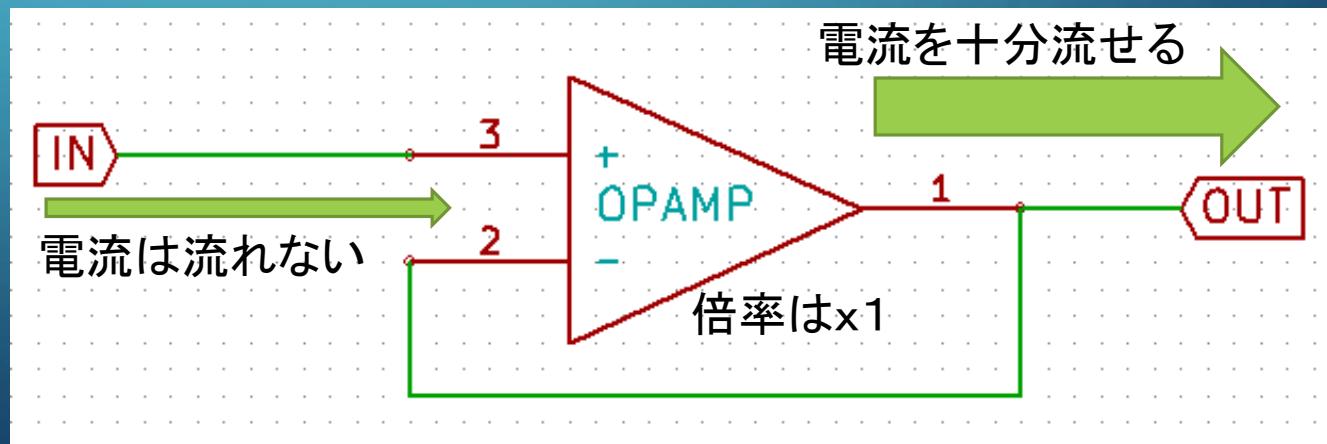
$$k = \frac{R_1 + R_2}{R_1} \text{(正の数)}$$

- 例えば, $R_1=2\text{k}\Omega$, $R_2=18\text{k}\Omega$
だと, 倍率は10.



ボルテージフォロア

- 増幅倍率が1の非反転増幅回路.
- 入力インピーダンスが高く、入力端子に電流は流さない.
 - 入力端子に電流を流すと電圧降下により電圧が変わってしまう
- 出力インピーダンスは低く、充分電流を供給できる.



入力インピーダンス

- デジタル・アナログ入力端子に直列に挿まれた抵抗のこと.
- 入力インピーダンスは、大きい方がよい。
→電流が流れにくくなるから.
- 信号入力端子は、電圧さえわかればいいので、電流はいらない.

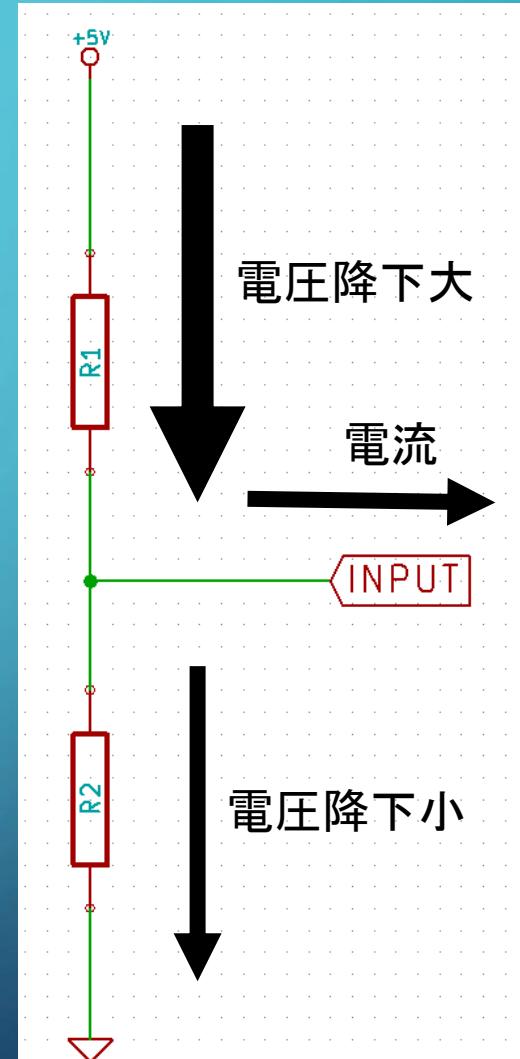
入力インピーダンス

- 抵抗分圧で考える.
- 2つの抵抗の電圧降下により電圧が決まる.

$$\text{INPUT} = \frac{R_2}{R_1+R_2} \times V_{CC} [\text{V}]$$

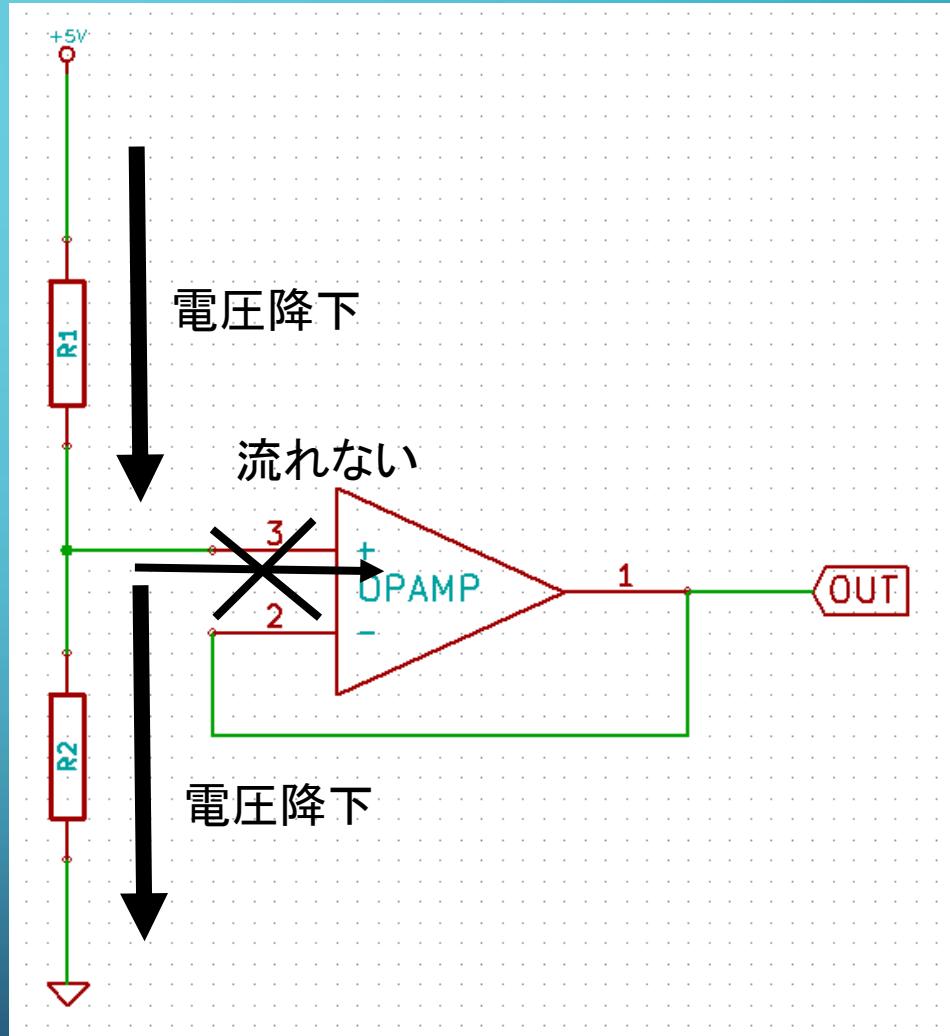
→ R1,R2を流れる電流は同じでないと困る.

- 入力側に電流が流れると、R1とR2に流れる電流が変わるので、分圧電圧が変わってしまう.



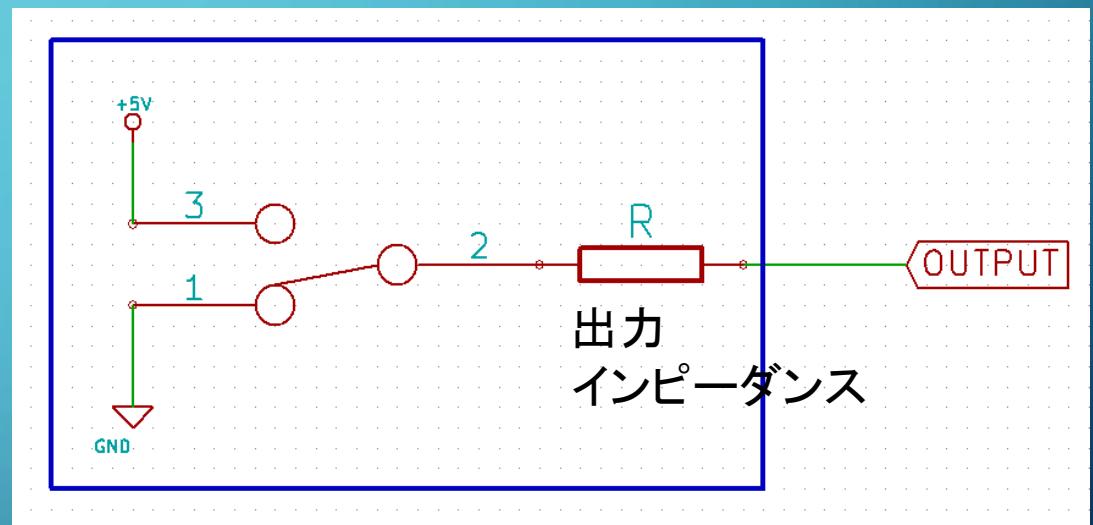
入力インピーダンスを大きく

- ボルテージフォロアを使う.
 - オペアンプの入力インピーダンスは高いので, 電流はほとんど流れない.
- 解決
- オペアンプの出力を入力端子につなげば, 入力インピーダンスを上げたことになる.



出力インピーダンス

- 出力端子に直列につながれた抵抗のこと.
- 出力電流が多いと、内部抵抗により出力電圧が下がってしまう.

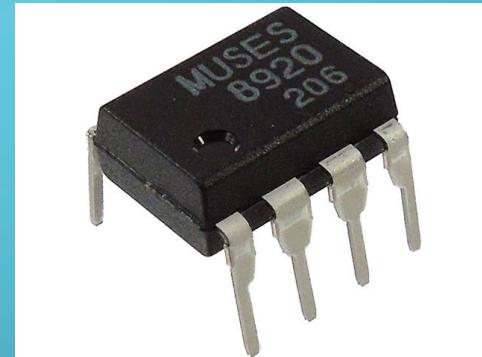
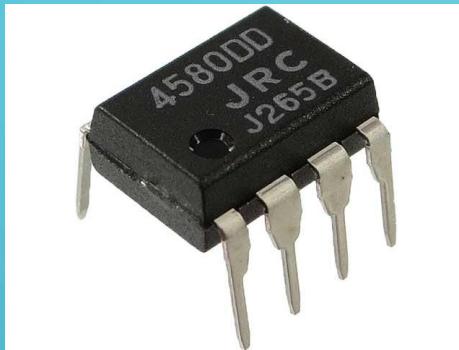


オペアンプについて

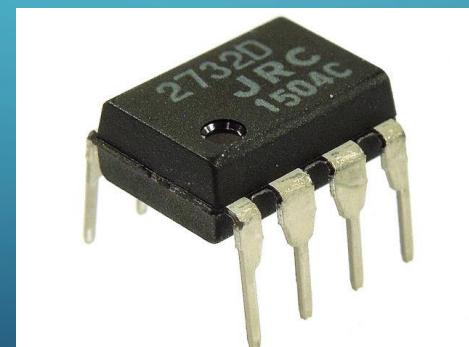
- オペアンプには、機能がたくさんある。
- 増幅倍率は、計算により導出できる。
- 種類がいろいろある。

おすすめのオペアンプ

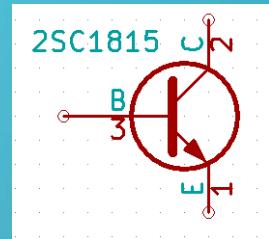
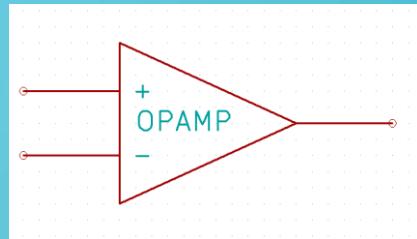
- オーディオ用
 - NJM4580DD
 - MUSES8920



- センサ用
 - NJM2732D



オペアンプ と トランジスタ



- オペアンプもトランジスタも「増幅作用」を持っている

1. トランジスタは、基準値よりも小さければ電流を流さず、基準値よりも大きければ電流を流そうとする。すなわち、全か無かの法則。

オペアンプはどんな電圧にも比例して一定倍率で増幅する。

2. トランジスタは、しぼりを開け閉めするイメージだが、オペアンプは電源端子を持っているので十分に電流を与えてくれる。

第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

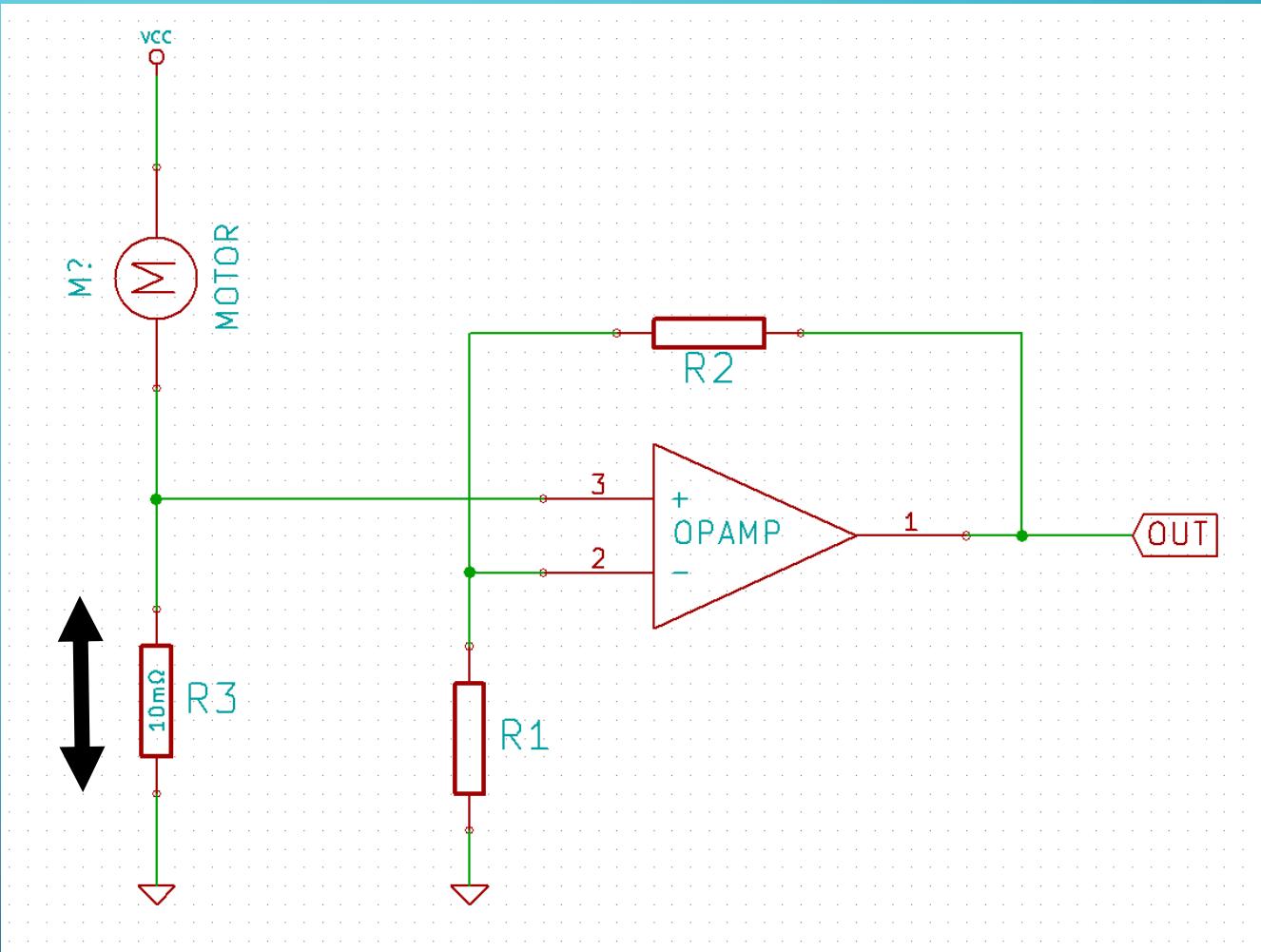
1. 電流を検出する

← 目次に戻る

*クリックするとそのページにジャンプします。

電流を検出する

- 測りたい回路に直列に小さな抵抗を挟む.
- その抵抗での電圧降下を測る.
- オペアンプで増幅する.



第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

[目次に戻る](#)

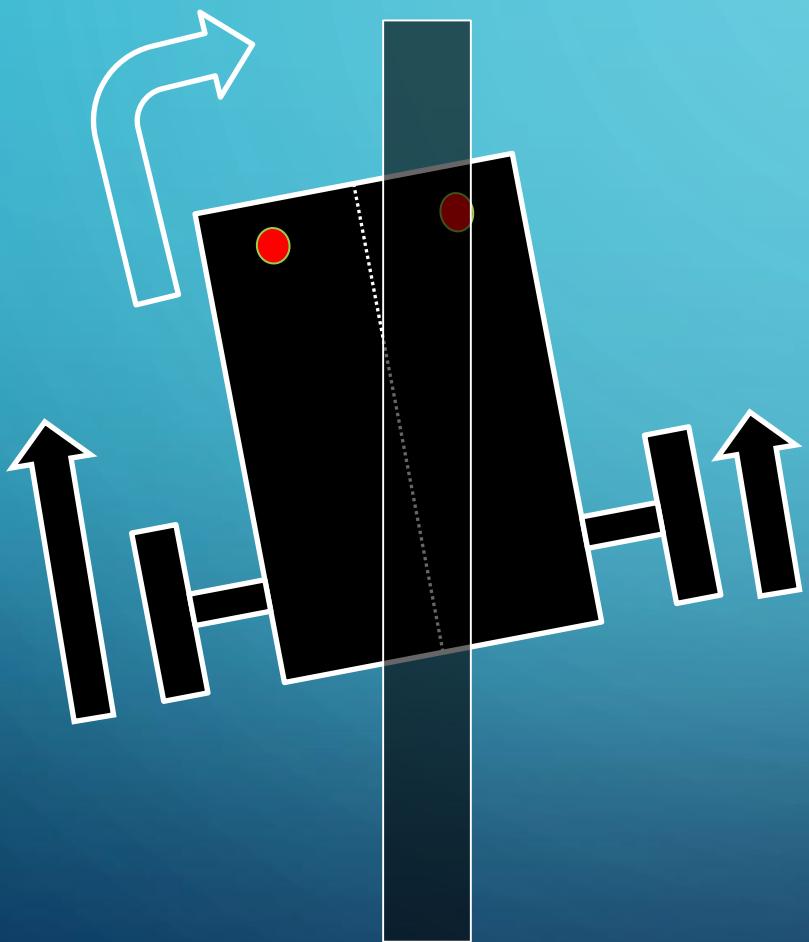
*クリックするとそのページにジャンプします。

1. ライントレース
2. ライントレース(直進)
3. 制御
4. ライントレース(旋回)

ライントレース

- ロボコンのフィールドには格子のラインが引いてある。そのラインをたどる（ライントレース）することでロボットはフィールド内を自由に動き回ることができる。
- より早く、ミスなくライントレースすることが大事である。

ライントレース(直進)

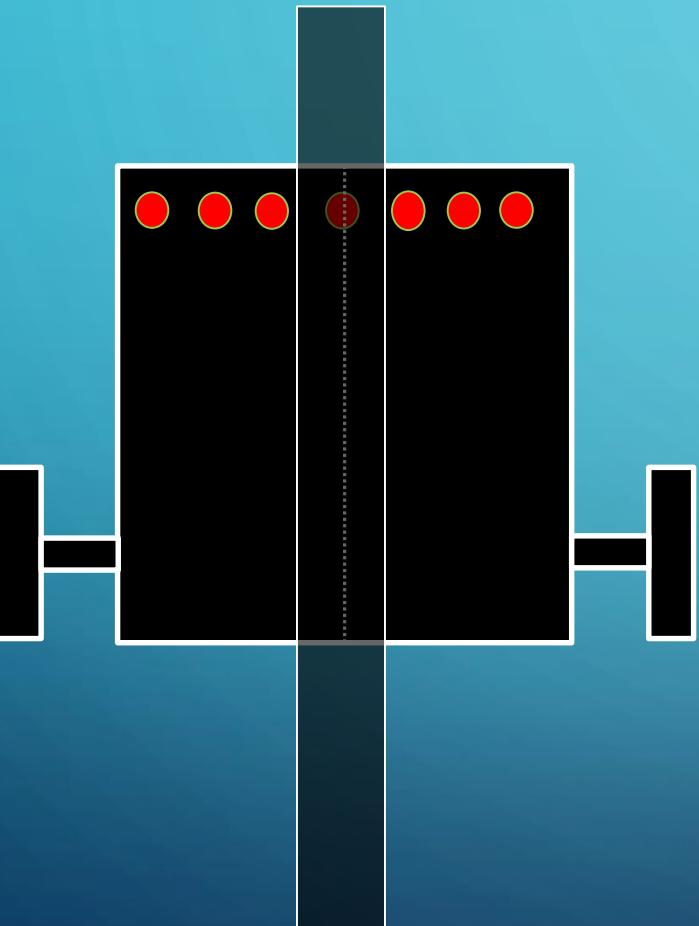


- 一番簡単な方法

- 右のセンサがラインに乗った→右へ旋回
- 左のセンサがラインに乗った→左へ旋回

2つのセンサ同士の幅	機体の動作
狭い	ラインから外れやすくなる
広い	ガタガタしてなかなか進まない

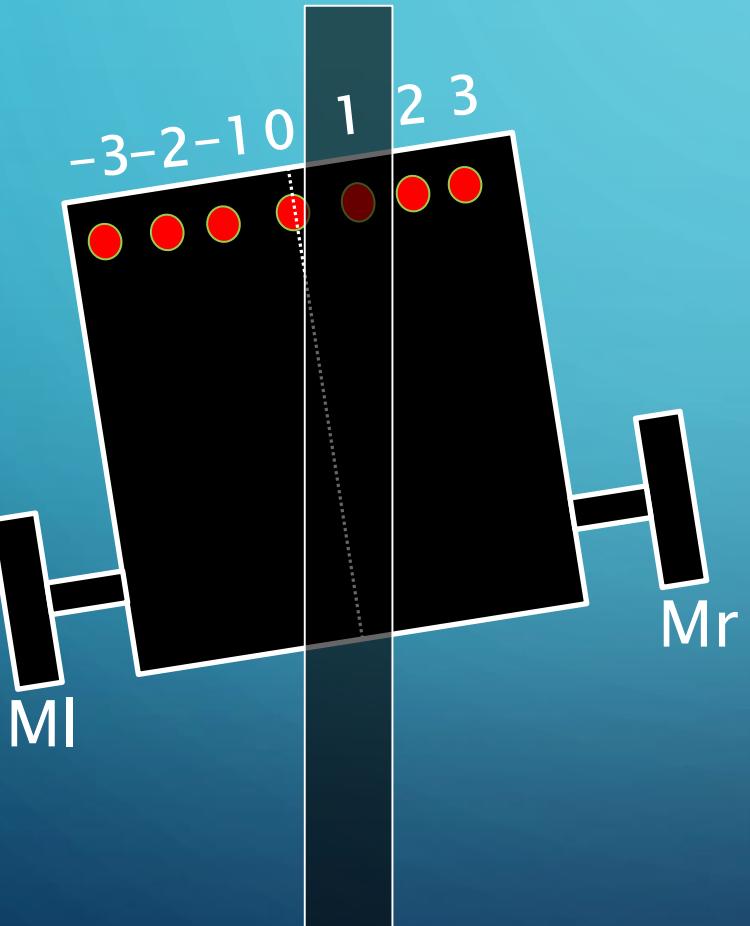
ライントレース(直進)



- センサを増やす
- 横に長いので、ラインを見失いにくい。
- センサの間隔が狭いので、細かく位置を把握できる
- 奇数個がいいと思う。

ライントレース(直進)

- 制御の例



センサに-3~3までの番号を付ける。

中心からのズレを e とおき,
例えばセンサ1がライン上にあるときには $e=1$ と表現する。

M_L, M_R : 右・左のモーターのduty比

M_0 : 直進時のモーターのduty比

K_p : 正の定数

$$M_L = M_0 + K_p \cdot e$$

$$M_R = M_0 - K_p \cdot e$$

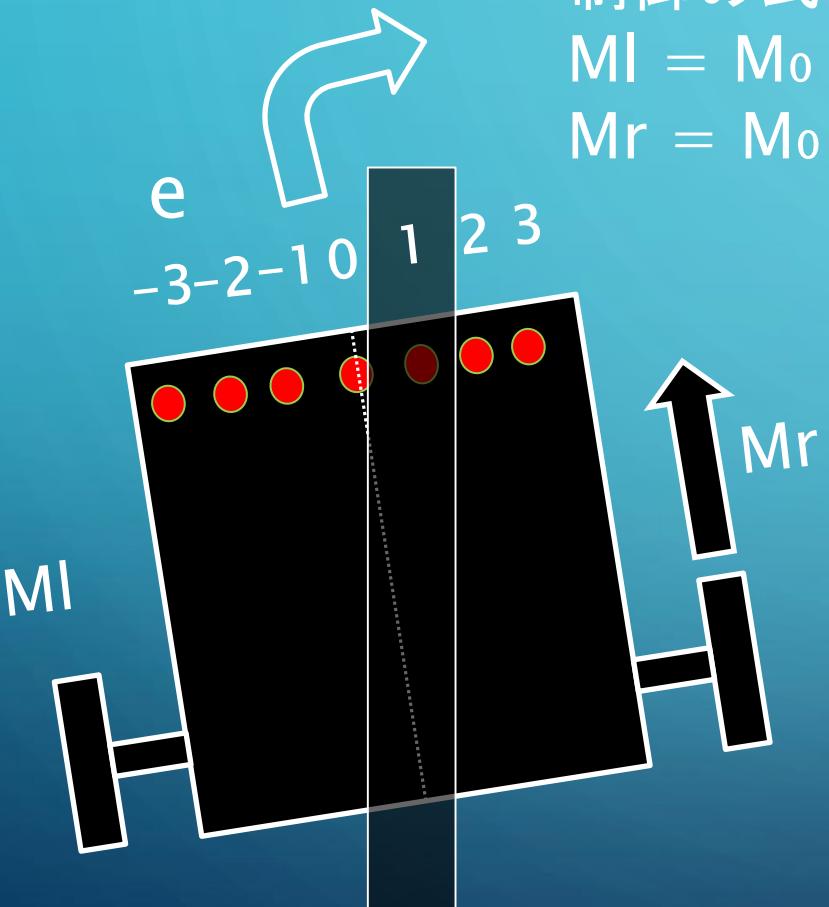
↑の式で左右のモーターの出力を計算する制御をする。

ライントレース(直進)

制御の式

$$MI = M_0 + K_p \cdot e$$

$$Mr = M_0 - K_p \cdot e$$



例えば $e > 0$ のとき,
 $MI > Mr$ になる



右に曲がる



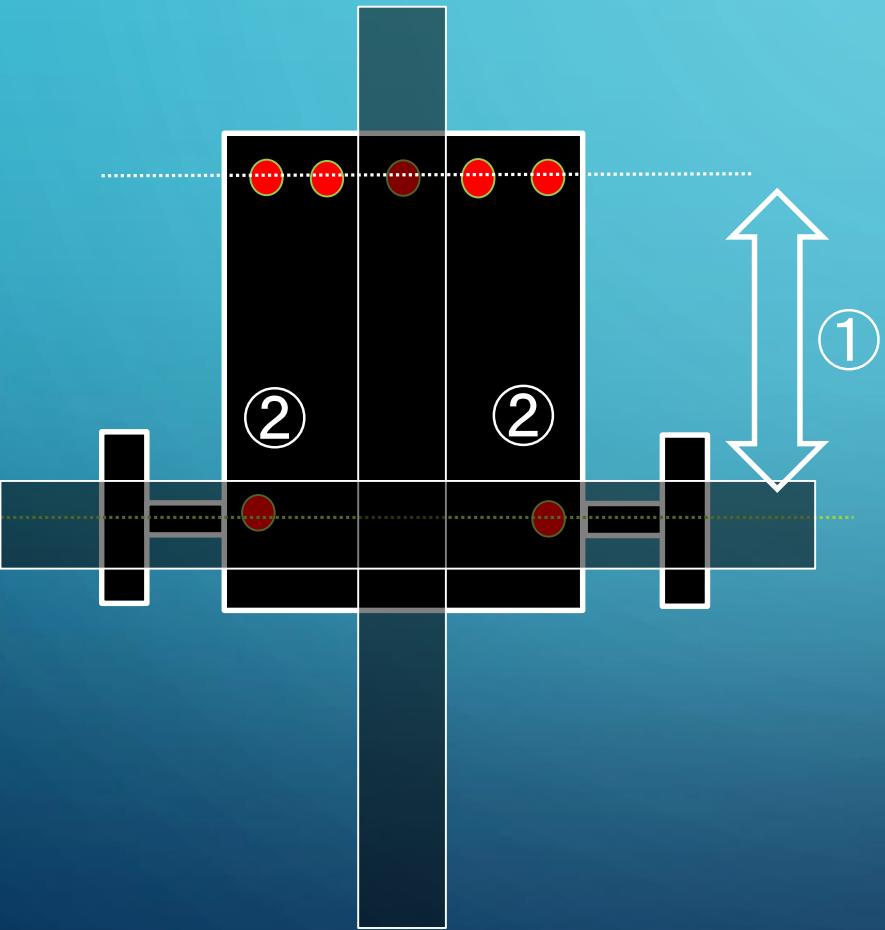
曲がる度合いは
 e (ズレ)に比例

制御

- 比例定数Kpはいろんな値で実験して求める.
- 偏差(ズレ)に比例させて制御量(モータ出力)を決める制御方法を
P制御といふ.
- PI制御, PD制御, PID制御とかもある.
 - P:比例, D:微分, I:積分

ライントレース(旋回)

後輪駆動の場合



- ①後輪の軸と前のセンサを離す。
→車体があまり回転しなくなるから
安定する。速く走れる。

- ②後輪の軸上にもセンサを付ける
→何マス進んだかを数える
→ラインの交点で停止できる
→簡単に旋回できる(次ページ)

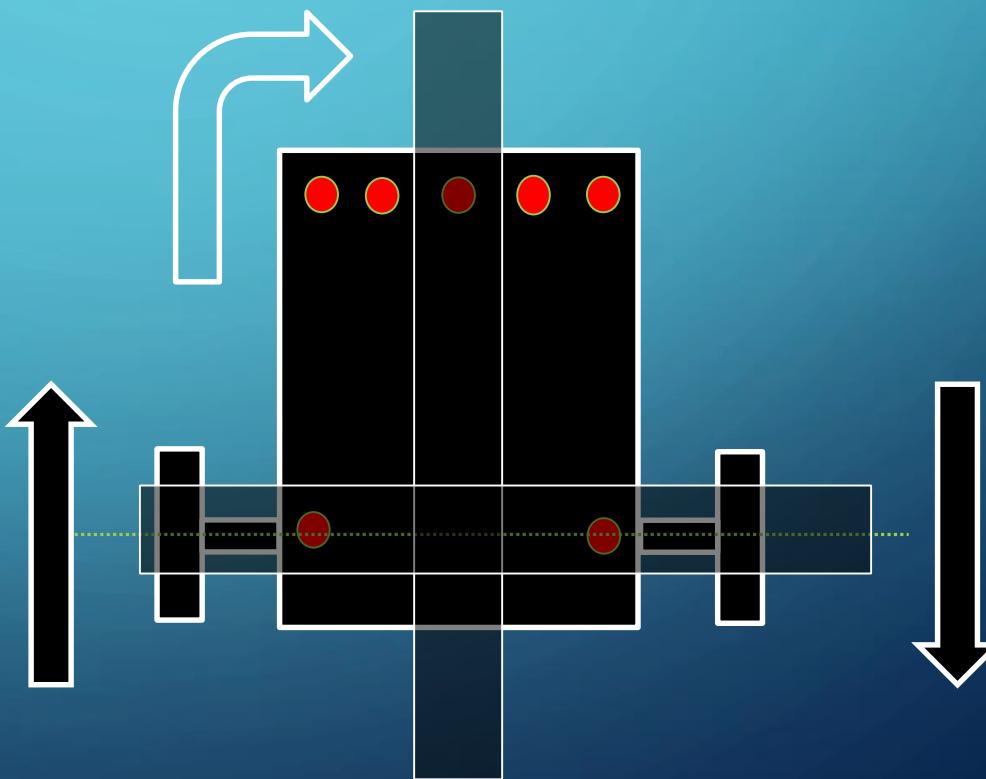
ライントレース(旋回)

1. ラインの交点に後輪の軸が来るよう停止

2. その場で旋回を始める

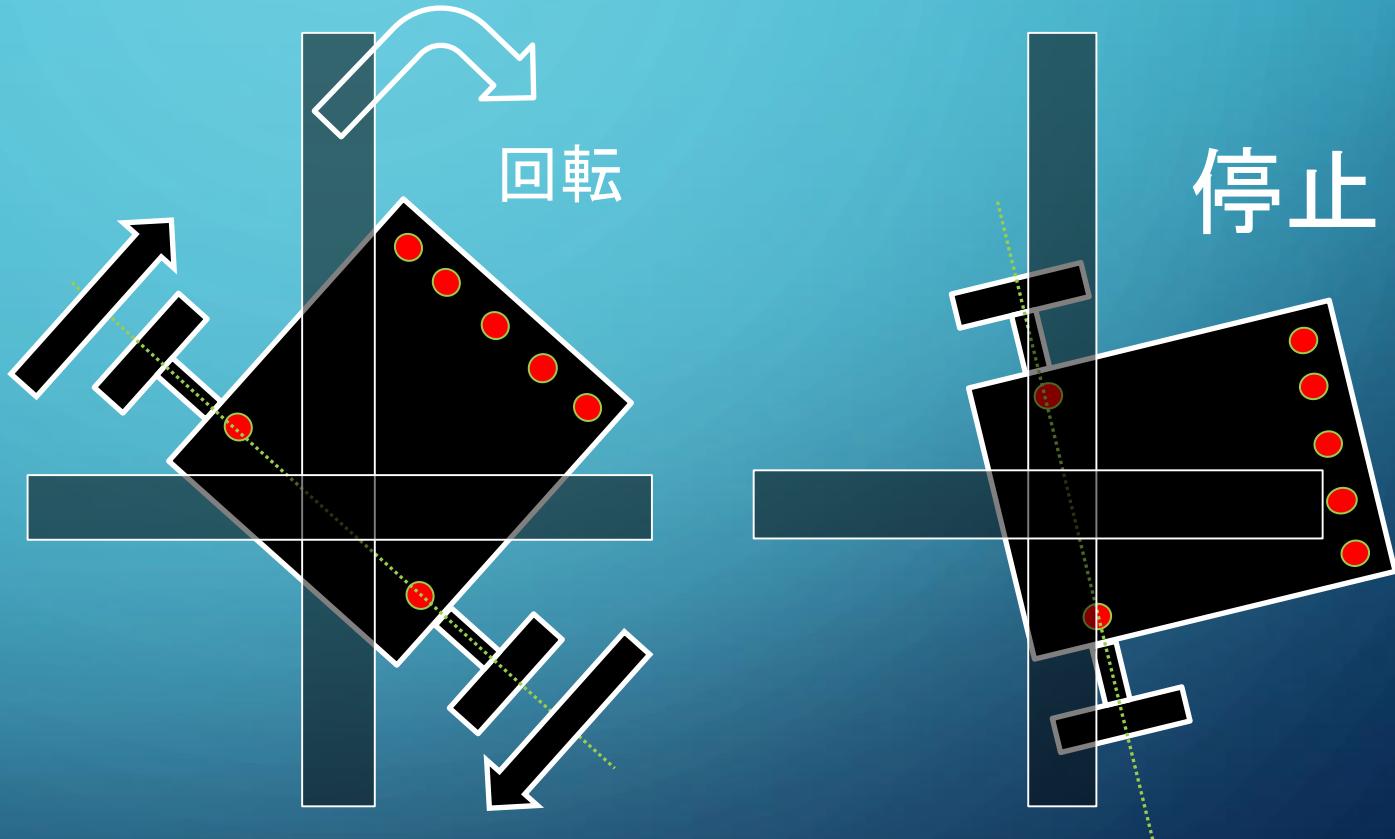
左のタイヤは、正転、

右のタイヤは、逆転。



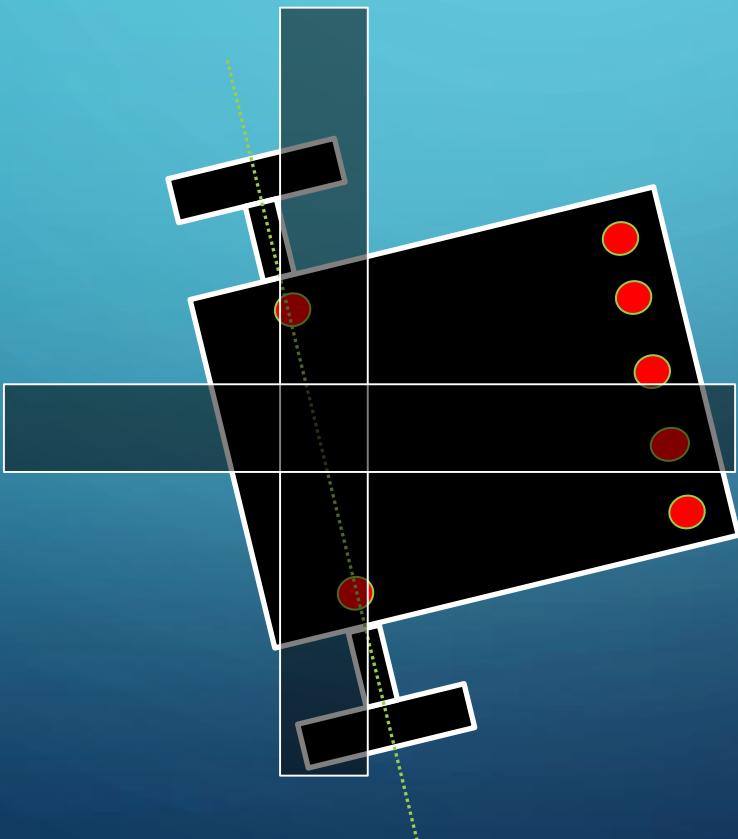
ライントレース(旋回)

3. 後ろのセンサが両方ともラインから外れる.
4. 後ろのセンサが両方ともラインにのる.
5. 停止する.



ライントレース(旋回)

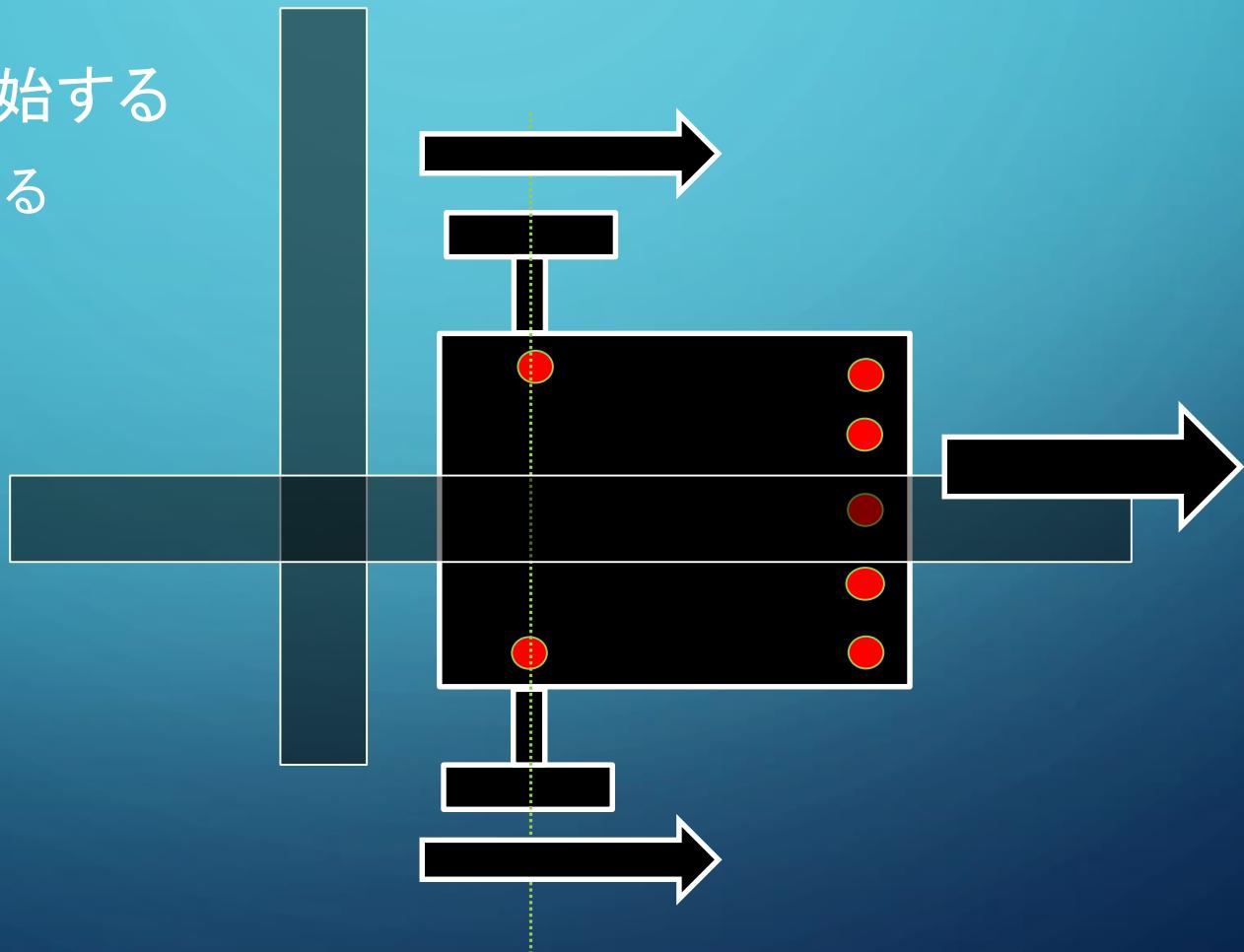
4. 後ろのセンサが両方ともラインにのる
5. 停止する



後ろのセンサが両方ともラインにのった瞬間,
機体は傾いている。

ライントレース(旋回)

6. 直進の制御を開始する
機体がまっすぐに戻る



第4回 目次

1. [宿題の解説](#)
2. [ADC](#)
3. [角度検出](#)
4. [回転数検出](#)
5. [障害物, ライン検知](#)
6. [フィルタ回路](#)
7. [オペアンプ](#)
8. [電流検出](#)
9. [ライントレース](#)
10. [終わりに](#)

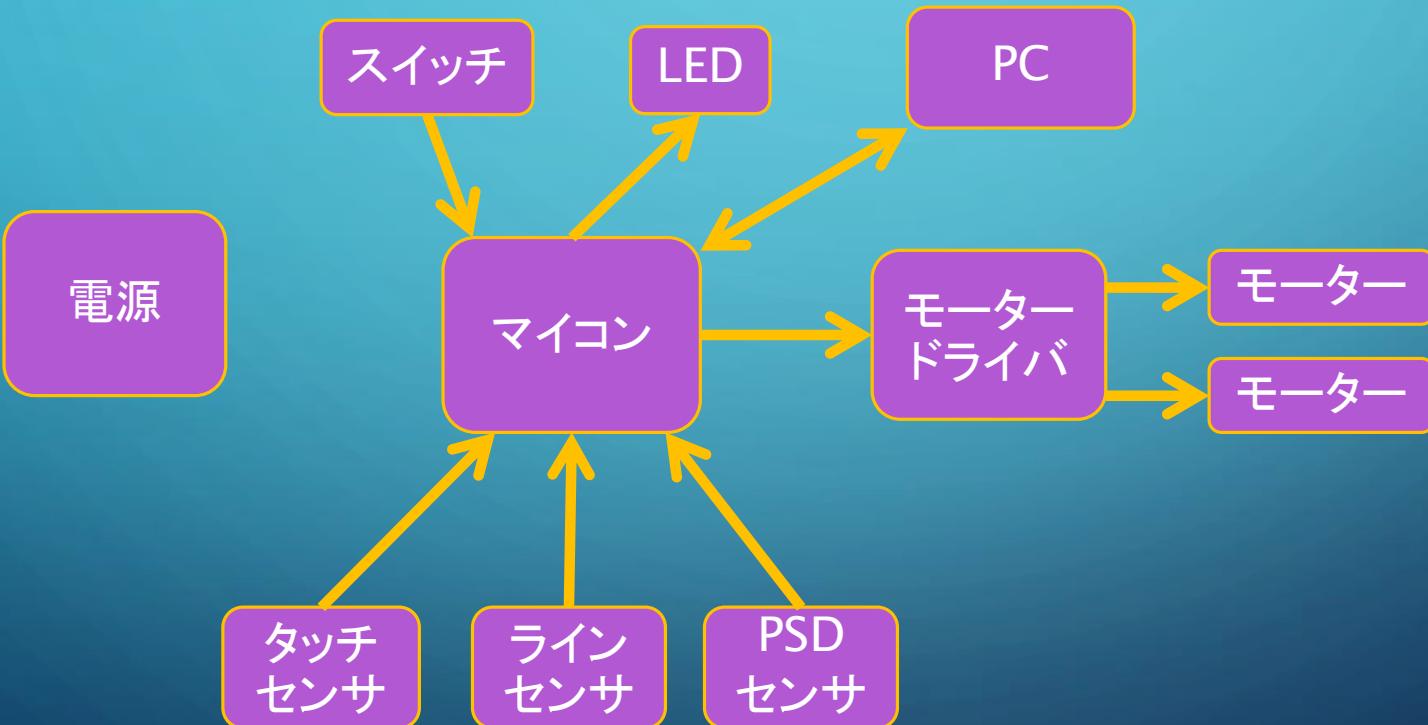
[目次に戻る](#)

*クリックするとそのページにジャンプします。

1. [ロボコンで使う回路](#)
2. [メッセージ](#)

ロボコンで使う回路

- 大きな回路は小さな機能ブロックの集合→ひとつひとつは単純
 - 例)ロボコンでよくある構成





メッセージ

- ・ものつくりを楽しもう
 - ・他人と共有しよう
 - ・イベントに出展、参加しよう
 - ・部室を大切に利用しよう。
- 

お疲れ様でした。
ご清聴ありがとうございます。