

目指せ究極のオリジナルサーボ！

～ 俺サーボは漢の浪漫 ～

2004/06/05

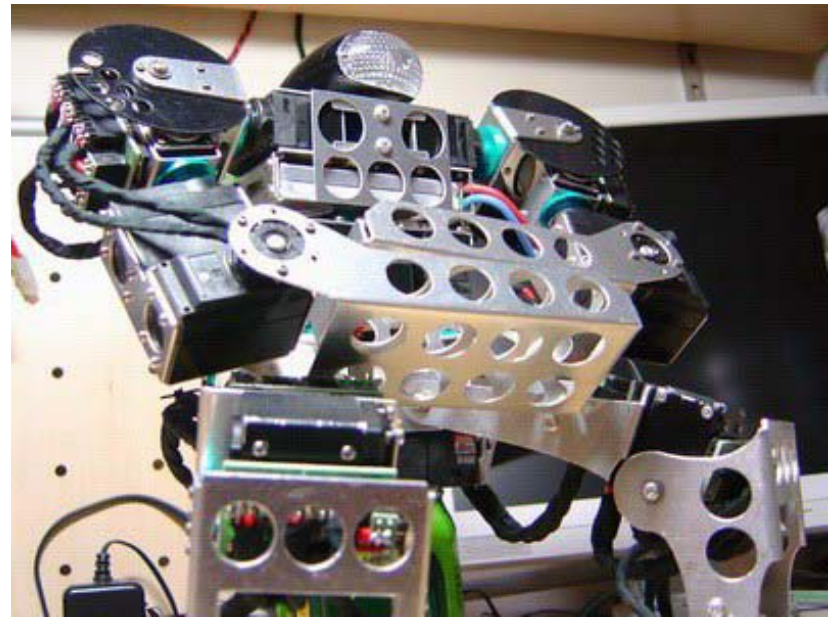
バーニング宮田

<http://www.geocities.jp/mimiin/>

Let's Burning !

サーボの役割

- ◆ ロボットの間接を駆動するユニット
- ◆ マイコンから角度を指示して間接を動かす



サーボの重要性

重量	ロボットの半分以上がサーボ重量 (56%)
消費電力	9割以上がサーボの消費電力 (94%)
スピード 保持トルク 追従性	実現可能なモーション、ロボットの大きさ/重さ 限界性能を決める
再現性	加熱、経年変化でモーションが再現しなくなる
振動	無駄な振動対策、ネジの緩み、格好悪い
音	不快感、作業中の疲労、近所迷惑

歩行時3.1A(94%) 直立時300mA(60%)

漢だったら

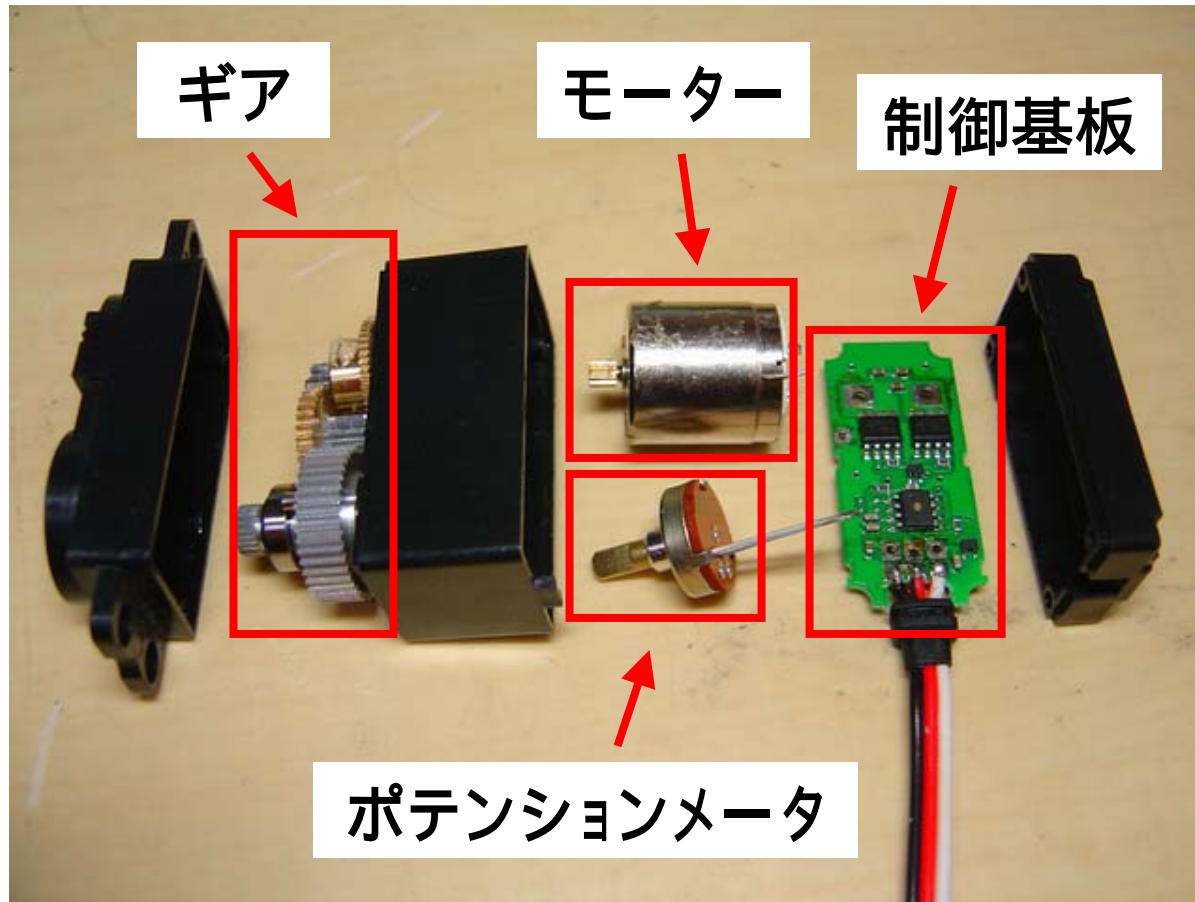
サーボの性能がロボットの性能に大きく影響
市販品に頼っていては他人と変わらない

オリジナルサーボで**究極**を目指せ！

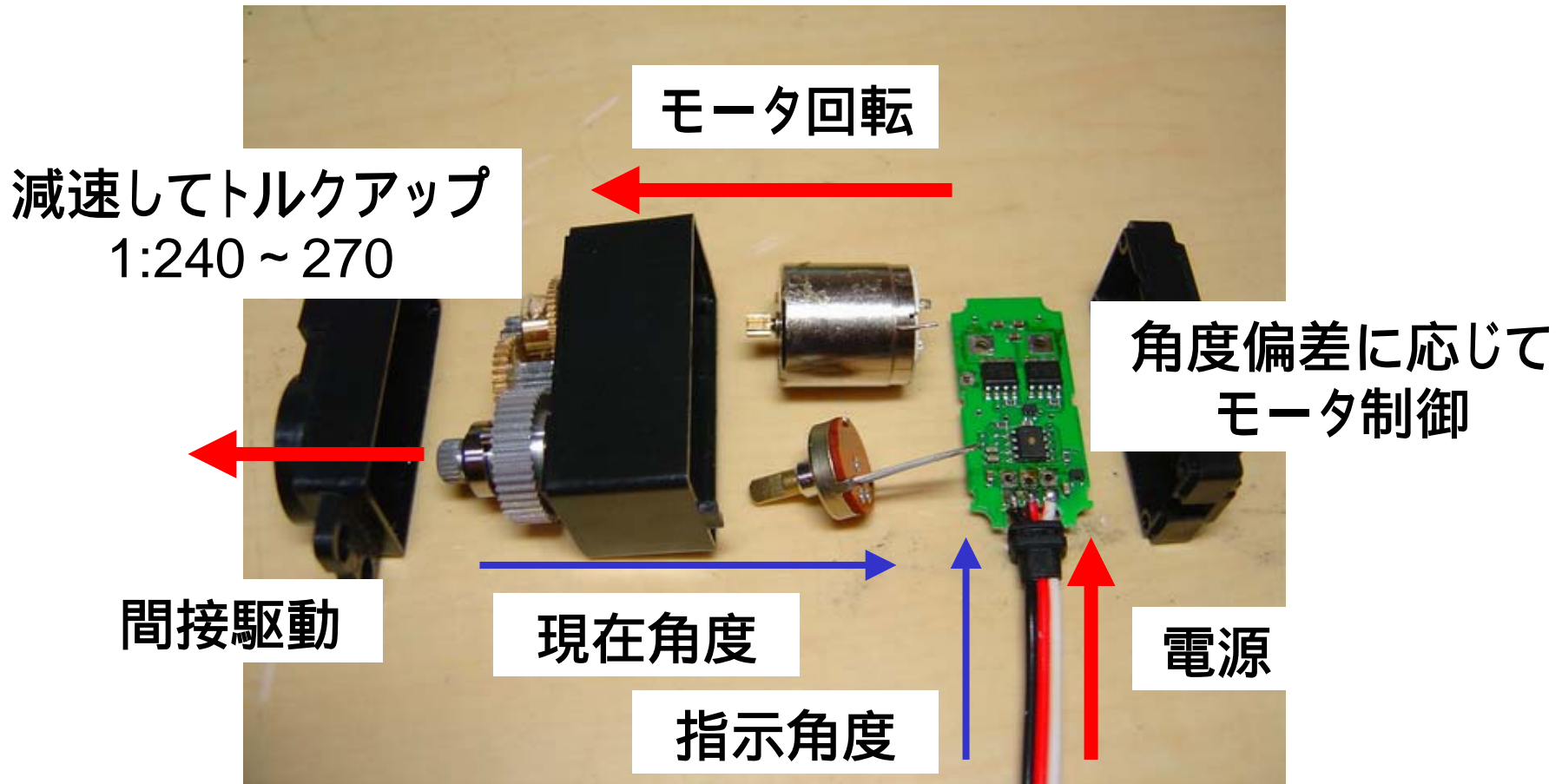
ただし・・・

- ◆市販のデジタルサーボは結構高性能
- ◆市販品を越えるためには気合と時間と**絶対**に**勝つ**！という信念が必要！
- ◆一度市販のデジタルサーボを使ってからチャレンジするのがお勧め！

RCサーボの構成

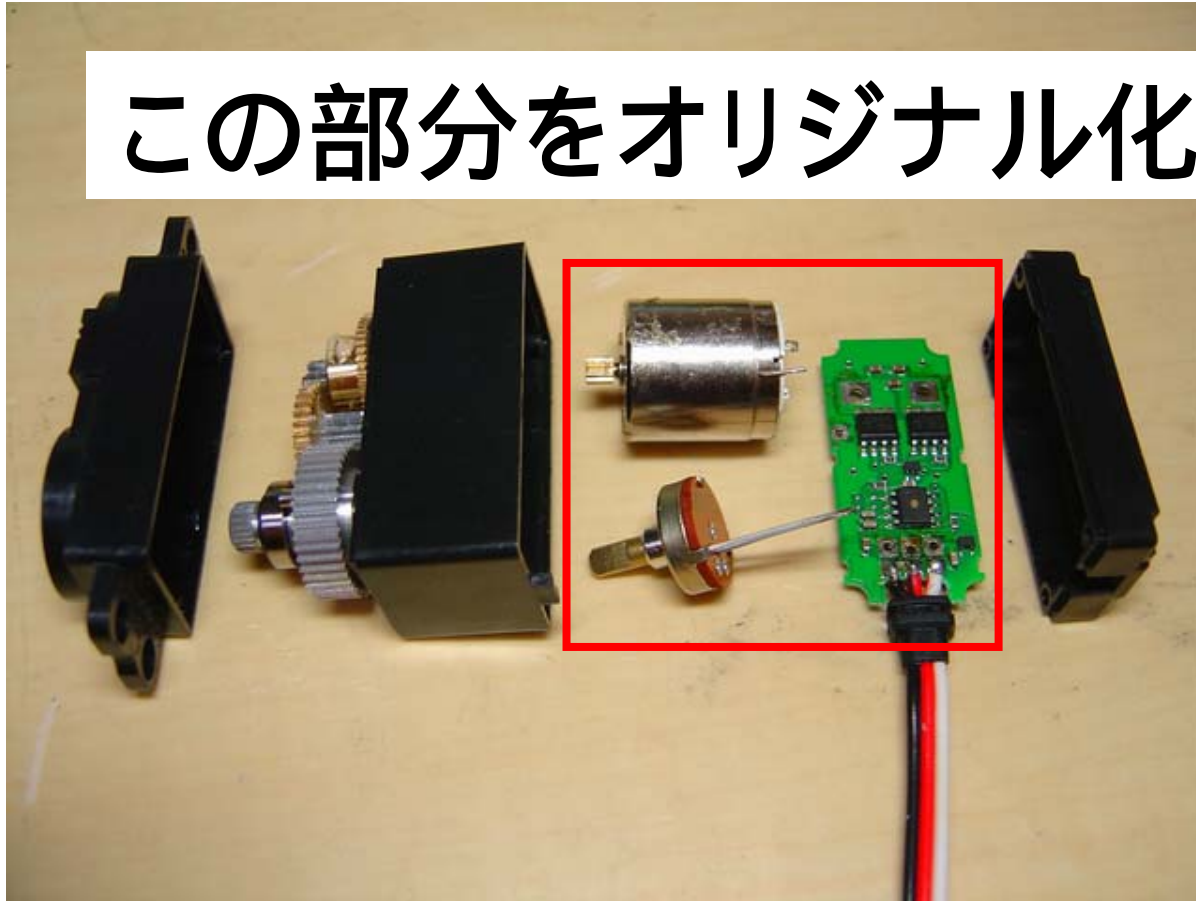


RCサーボの構成



俺サーボ

この部分をオリジナル化！



Let's Burning !

俺サーボ

- ◆ **制御基板**をオリジナル化し、**モータ**、**ポテンション**を改良または高性能品に交換することで重量以外の**全ての要素を改善可能**(消費電力、保持トルク、再現性、振動、音etc)
- ◆ ギアやケースは素性の良いサーボが豊富にある。オリジナル化で得られるメリットは少ない

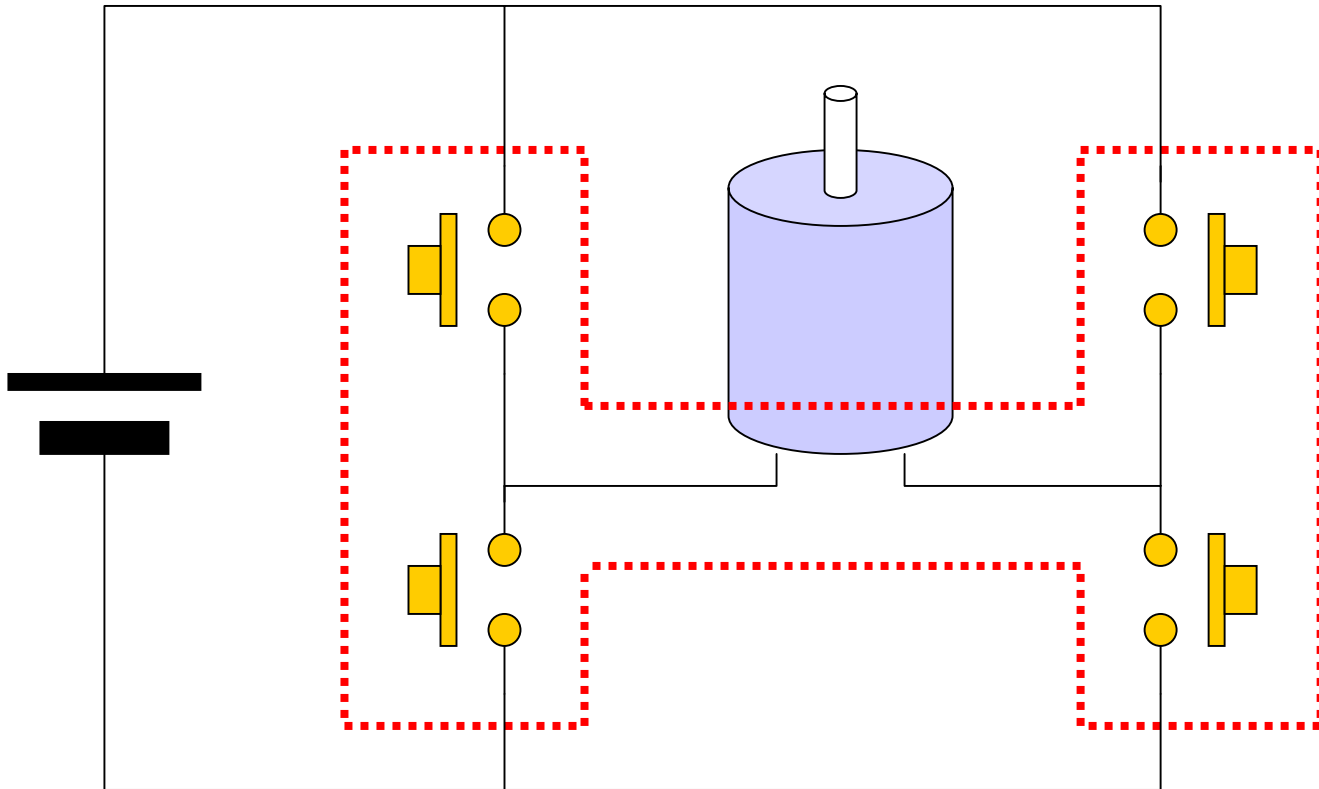
サーボ制御基板の役割

- ◆ 指示角度を受け取る
- ◆ 現在角度を取得する
- ◆ 偏差から制御量を計算する
- ◆ モータを制御する
- ◆ 状態(角度、トルク、電圧etc)を返す

- ◆ **私の経験**に基づいて説明する

PWMでモータ制御

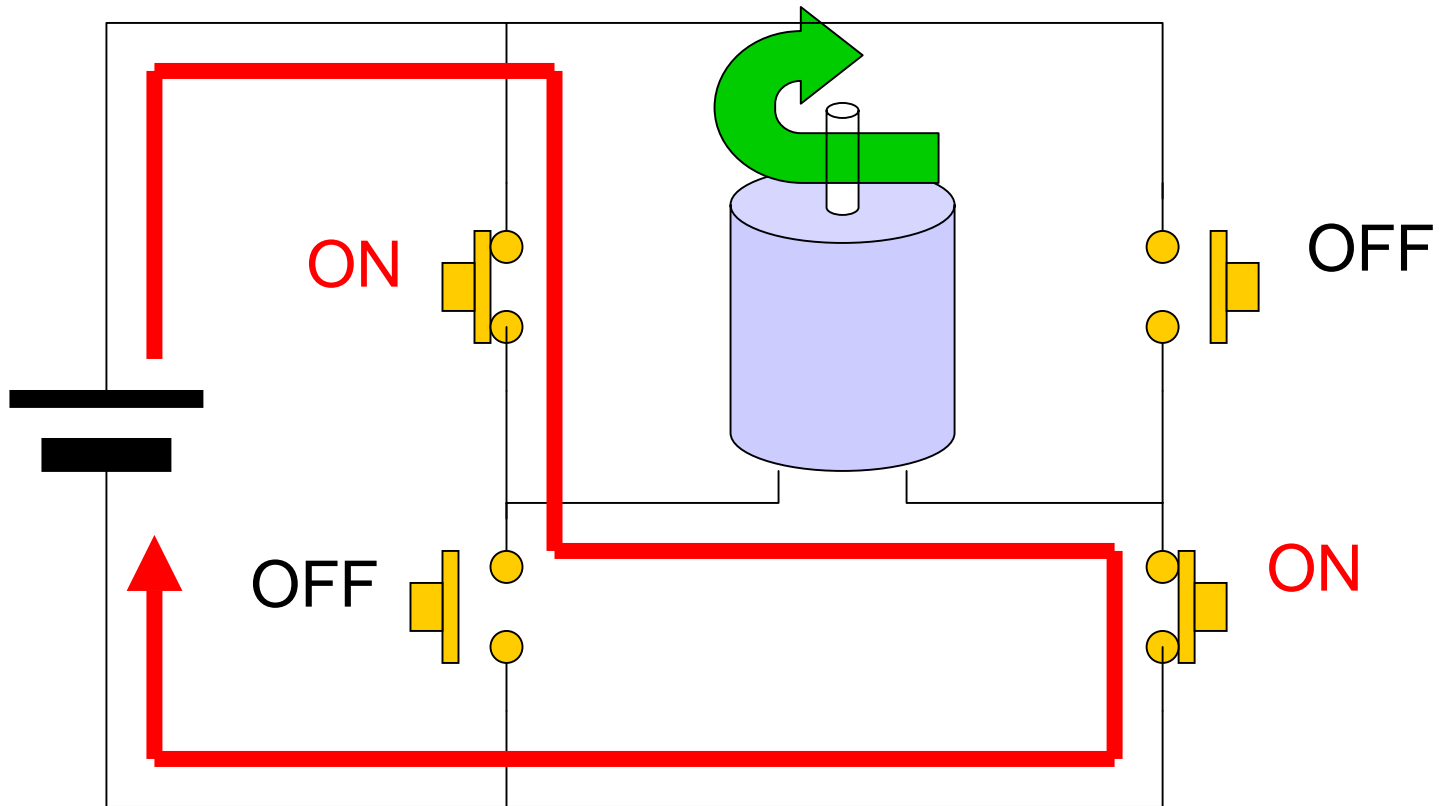
モータを制御する



これがH-ブリッジ！要はスイッチが4個

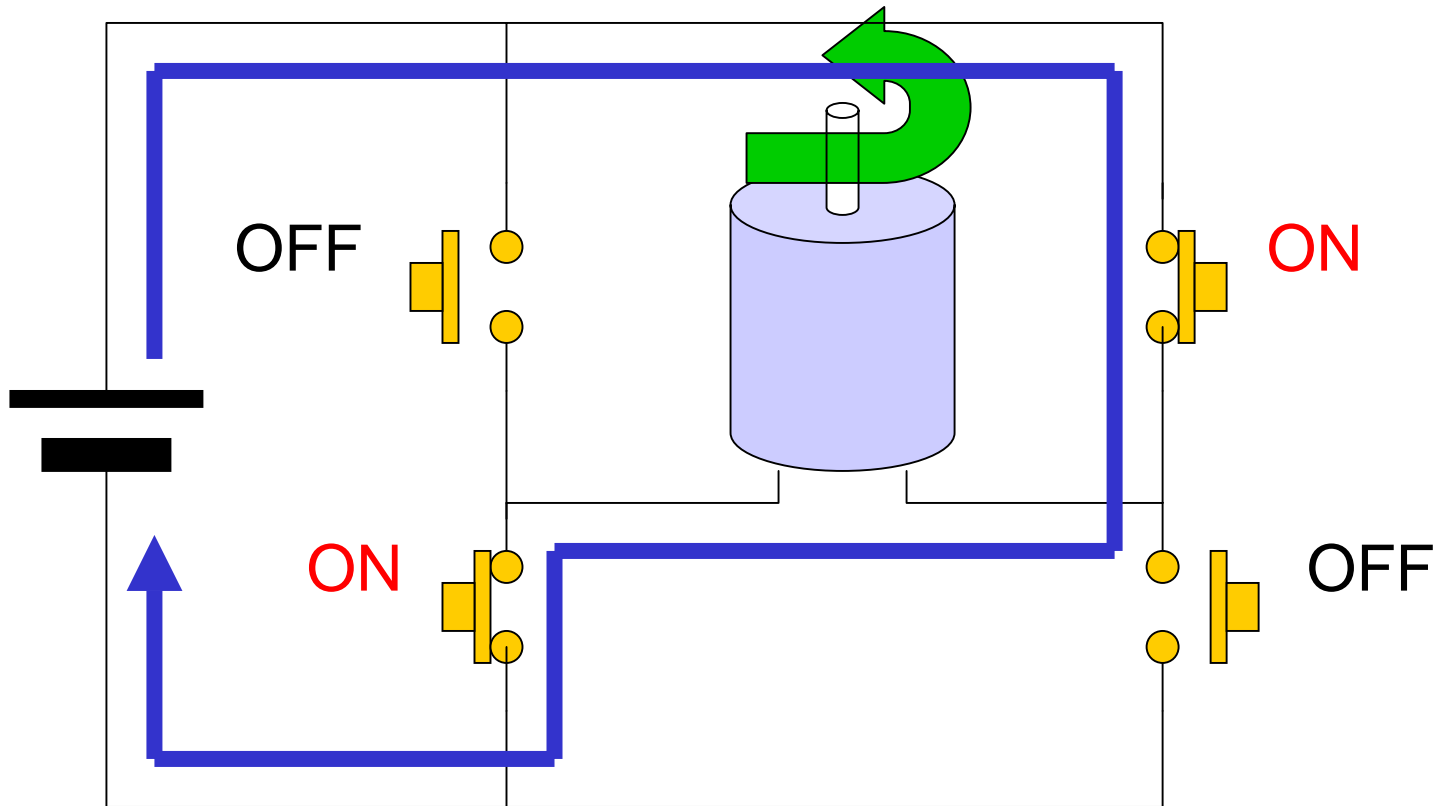
Let's Burning !

モータを制御する



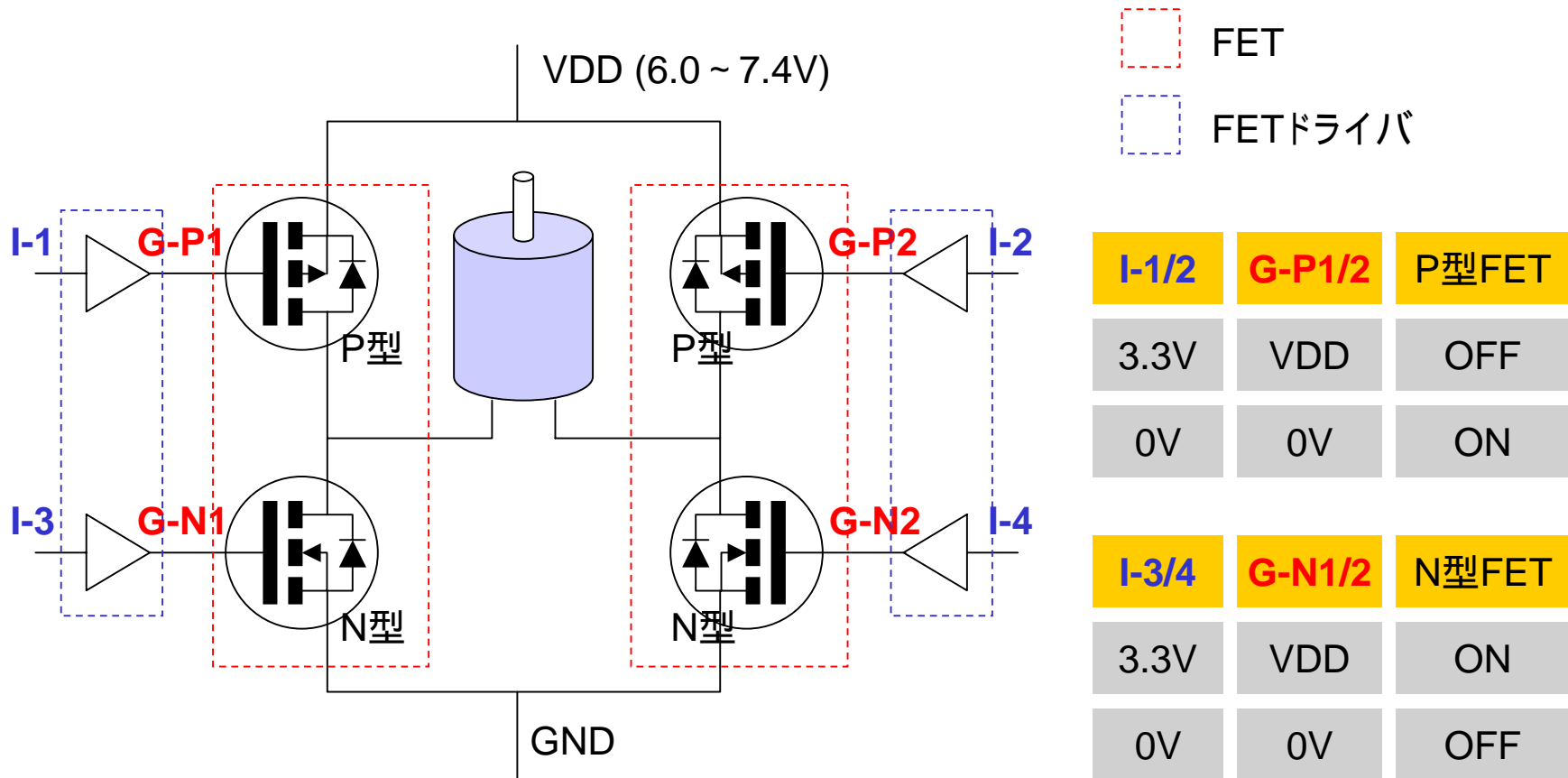
Let's Burning !

モータを制御する



Let's Burning !

FET + FETドライバでスイッチング



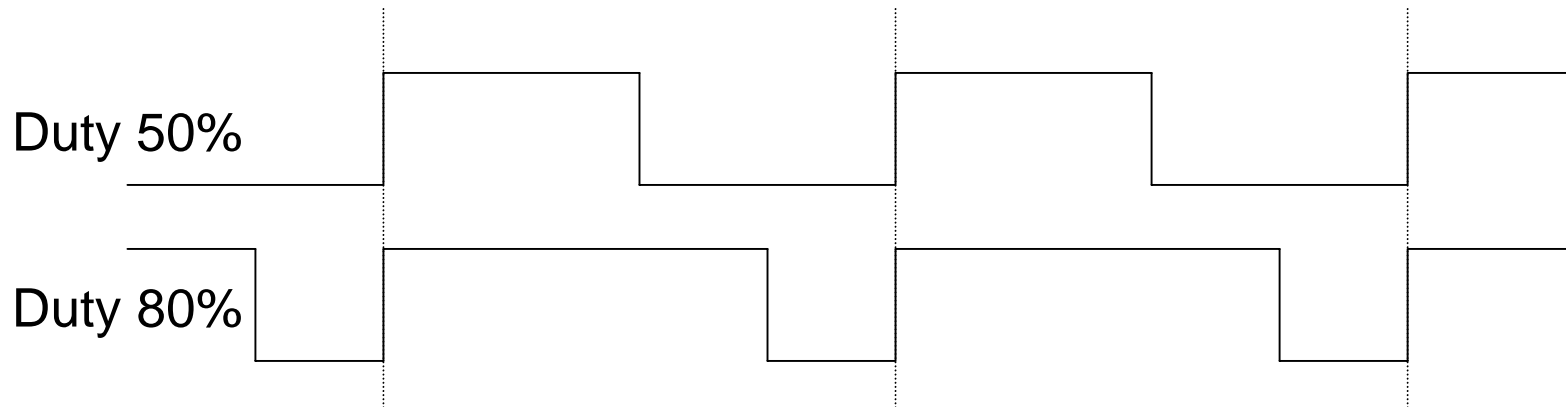
Let's Burning !

FET + FETドライバでスイッチング

- ◆ FET (スイッチの役割)
 - ゲートの電圧を操作することでON/OFF切り替え
 - N型 5V以上でON、0VでOFF
 - P型 VDD-5V以下でON、VDDでOFF
 - FETのゲートの静電容量は大きい(数百pF ~ 数千pF)
- ◆ FETドライバ (FETのゲートを高速駆動)
 - マイコンの信号をFETを駆動できるレベルに変換
 - 入力0Vで出力0V
 - 入力3.3Vで出力VDD
 - 出力電流は1.5Aから9A以上流せるものもある
(マイコンの出力電流は良くても $\pm 20\text{mA}$)

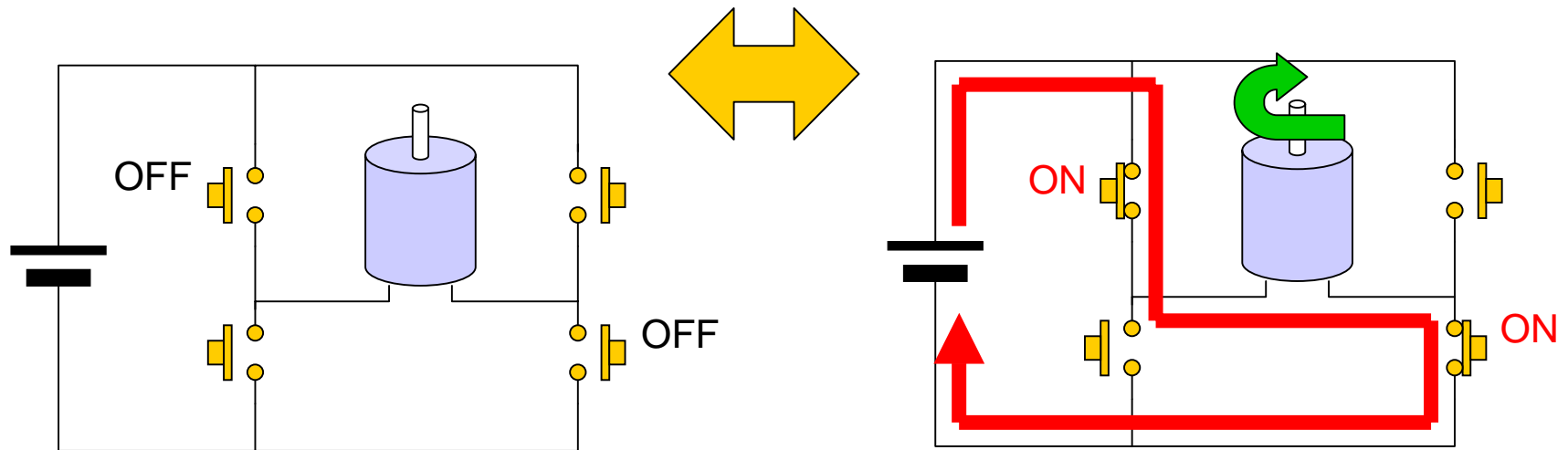
PWMで出力制御

- ◆ 正転/逆転はできるが、常にフルパワー
- ◆ PWMを使って出力制御を行う
- ◆ PWM信号
 - ◆ 周期が一定でパルスの幅が可変



PWMで出力制御

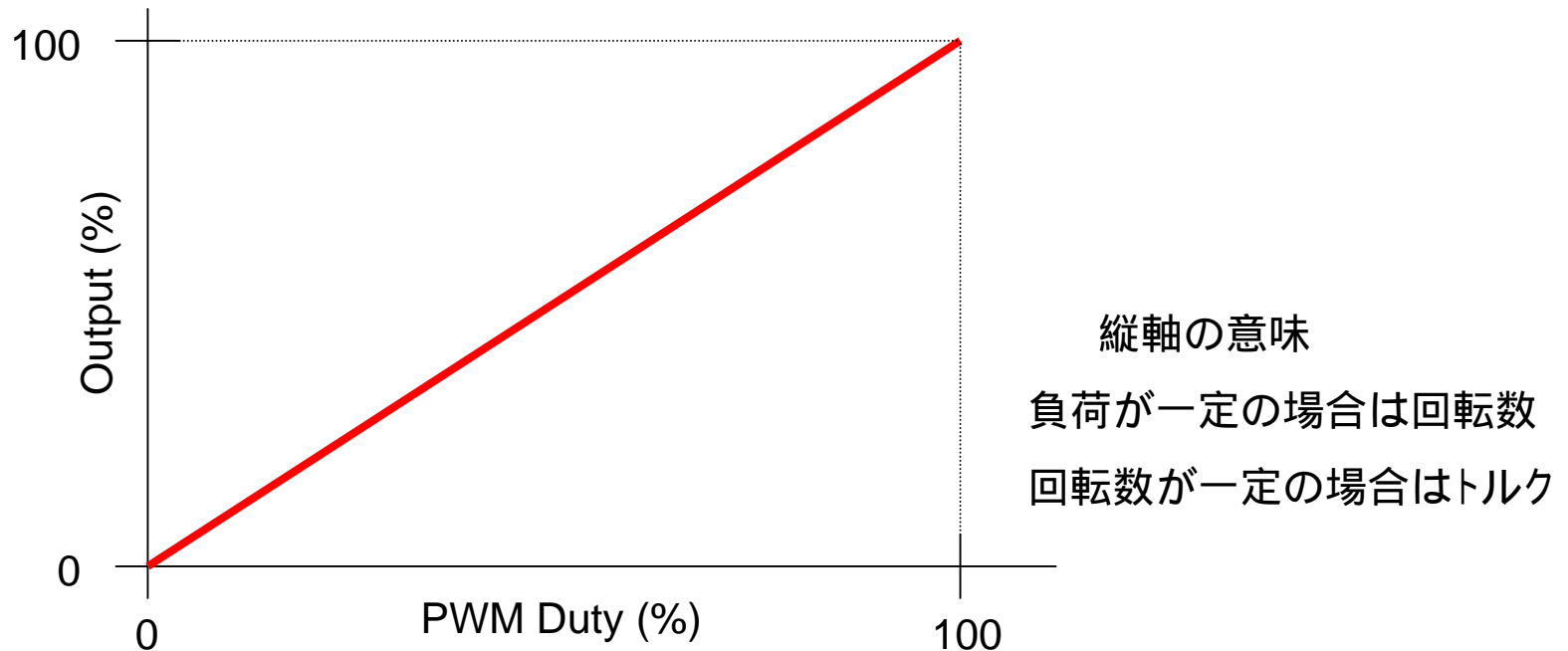
- ◆ PWM信号がHighの時にH-ブリッジON
- ◆ PWM信号がLowの時にH-ブリッジOFF



Let's Burning !

PWMで出力制御

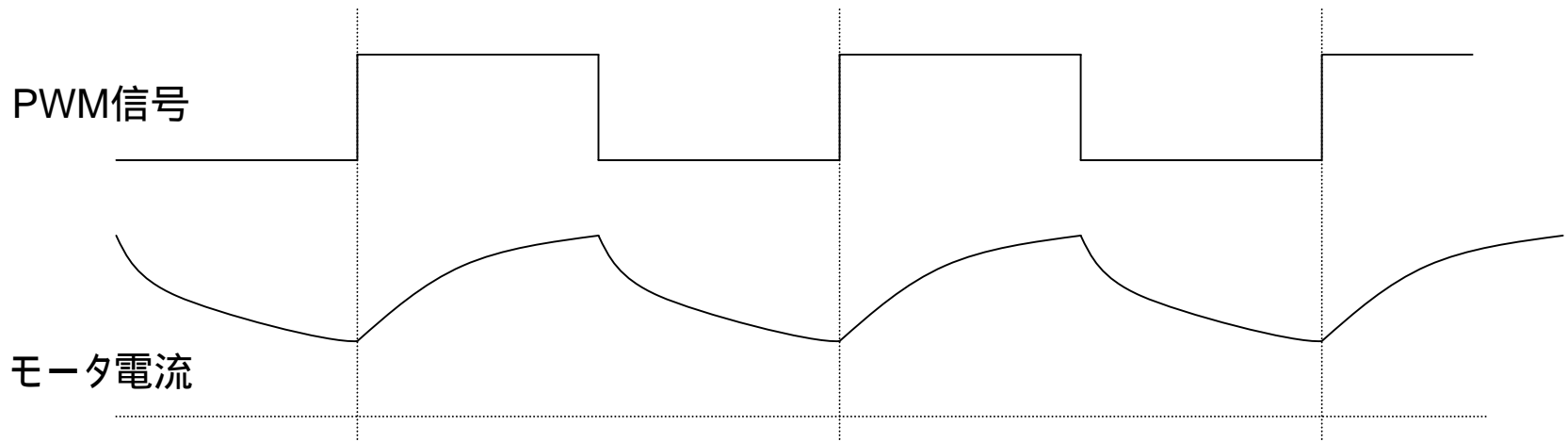
◆PWMでリニアに出力制御できるのが**理想**



Let's Burning !

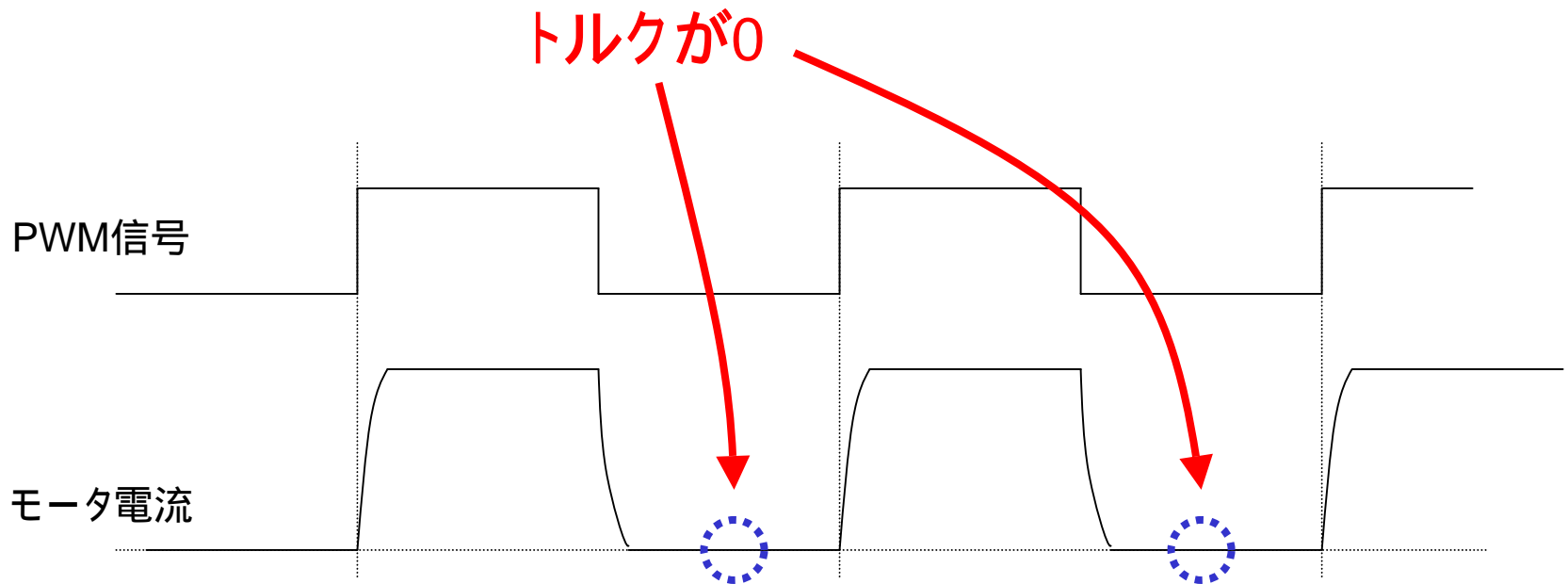
リニア制御

- ◆ モーターはコイル、トルクは電流に比例
- ◆ コイル時定数(数十KHz)の数倍のPWM周期
- ◆ 逆起電力による電流が流れる帰還ルートが必要



PWM周期が遅いと・・・

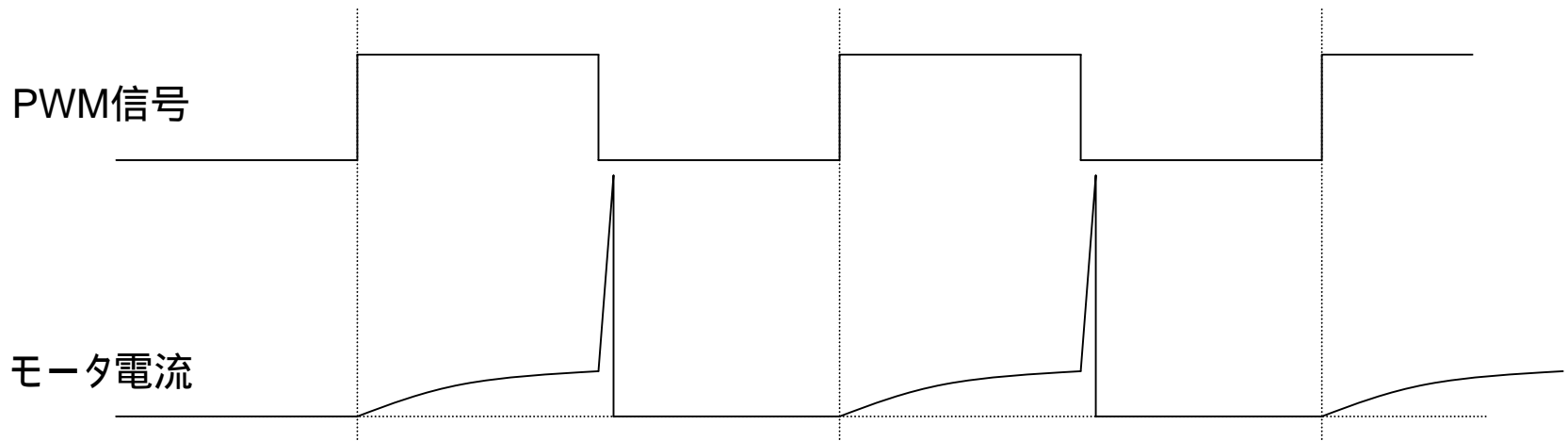
- ◆タダのON-OFF制御
- ◆トルクが0の期間が存在する



Let's Burning !

帰還ルートが無いと・・・

- ◆コイルに溜まったエネルギーはノイズになる
- ◆回路にダメージを与える
- ◆PWMのパルス幅を増やしても電流は増えない

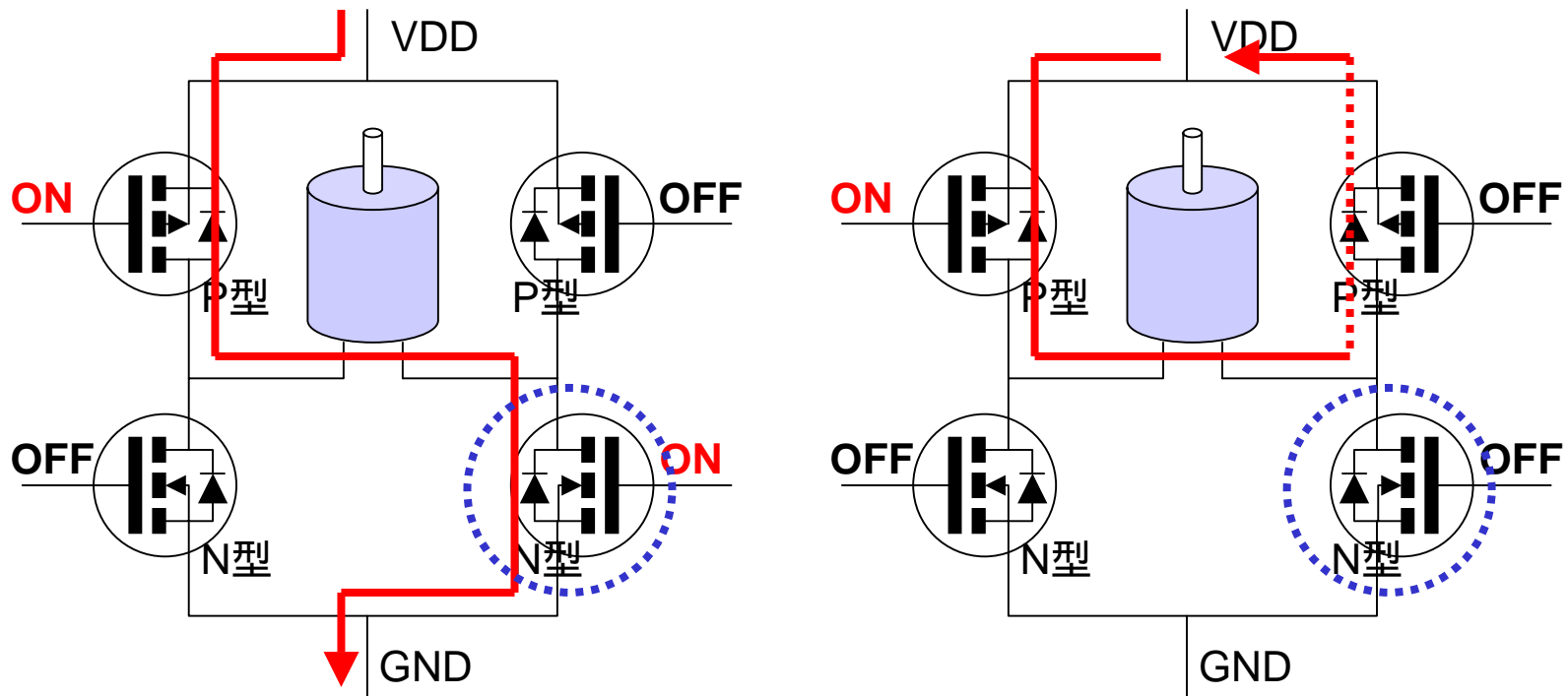


Low-Side PWM駆動

- ◆H-ブリッジでモータをPWM駆動する**真に最適**な方法の説明はあまり文献がない
- ◆私が考える**最適**な駆動方法を説明する

Low-Side PWM駆動

◆ Low-Side(N型)だけPWM駆動する



Let's Burning !

Low-Side PWM駆動

◆メリット

リニアに制御できる(電流が連続に流れる)
高速スイッチング (N型の限界まで)
貫通電流が流れない

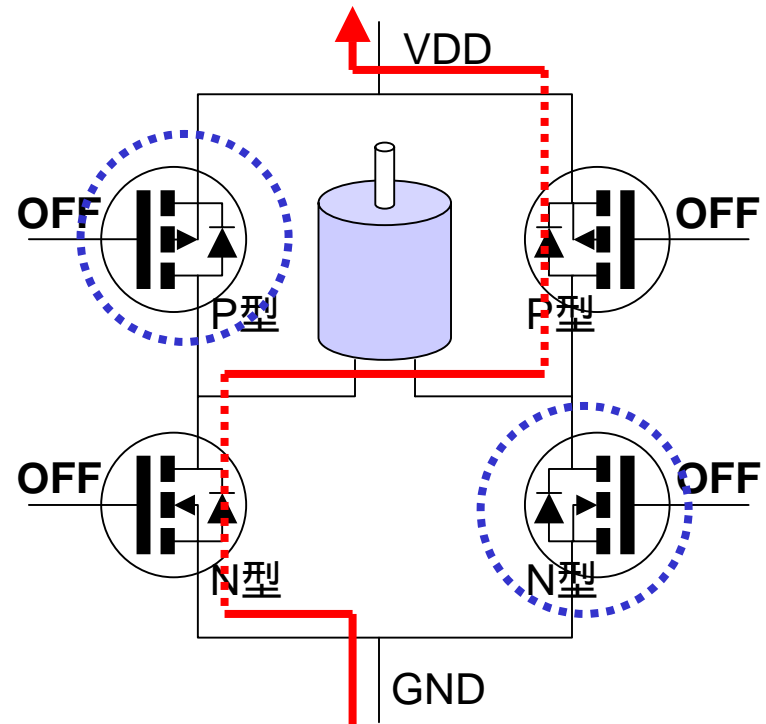
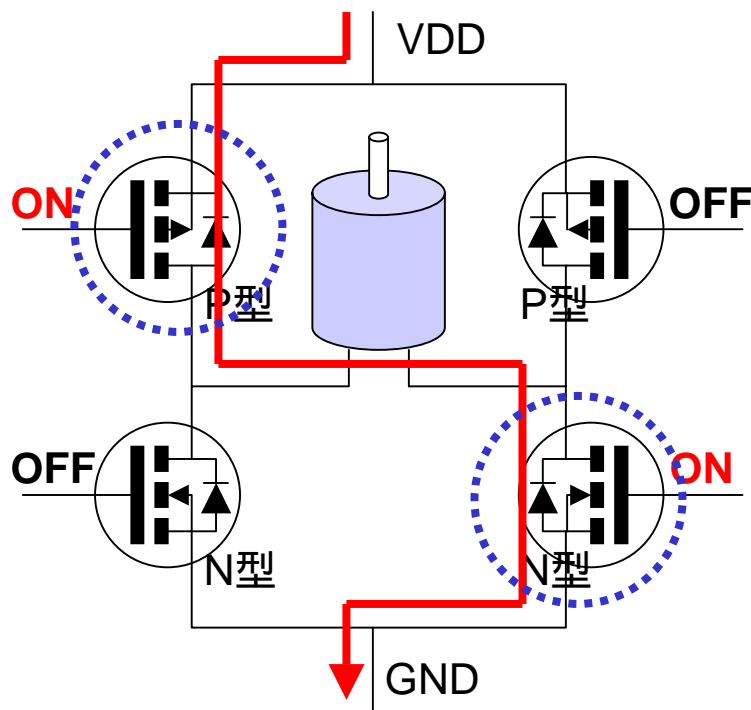
◆デメリット

ゲート4つを個別に制御する必要あり

その他のPWM駆動 (ON-Free)

◆ ON-Free

電源電圧を超えないと
逆起電力による電流が流れない！



Let's Burning !

その他のPWM駆動 (ON-Free)

◆メリット

ゲート4つを2線で制御可能
貫通電流が流れない

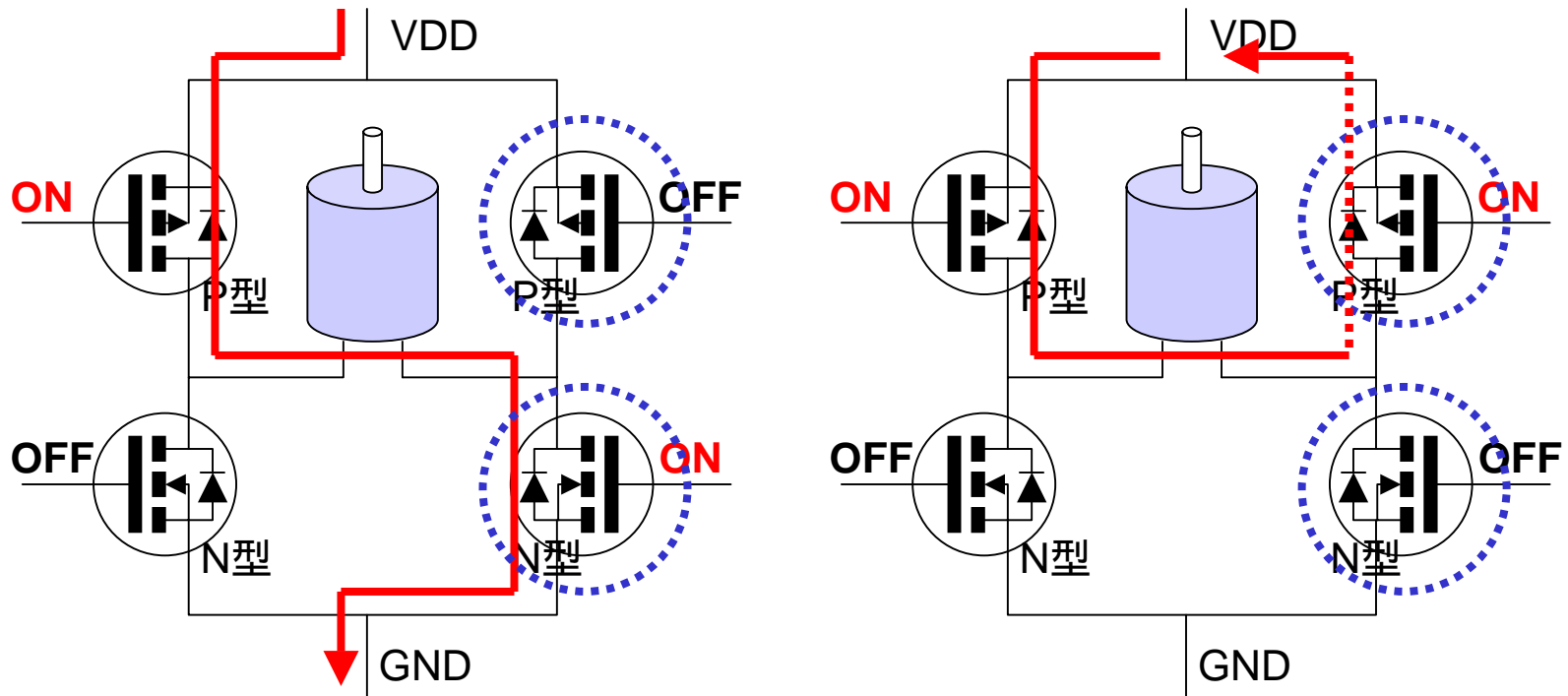
◆デメリット

リニアに制御できない(電流が連続に流れない)
高速スイッチングできない (P型の限界まで)

その他のPWM駆動 (ON-Break)

◆ ON-Break

縦のP/N FETのON/OFFを同時に切り替える瞬間に貫通電流が流れる！



Let's Burning !

その他のPWM駆動 (ON-Break)

◆メリット

リニアに制御できる(電流が連続に流れる)
ゲート4つを2線で制御可能

◆デメリット

高速スイッチングできない (P型の限界まで)
貫通電流が流れる

PWM駆動方法のまとめ

- ◆リニア制御と貫通電流なしは**必須条件**
- ◆制御線の数
は工夫で何とかなる
- ◆モータのPWM駆動は**Low-Side PWMが最適**

	リニア制御	PWM速度	貫通電流	2線制御
Low-Side			なし	×
ON-Free	×	×	なし	
ON-Break		×	あり	

現在角度の取得

現在角度の取得

- ◆ 簡単に説明すると、ぶっちゃけこれだけ
 - ◆ 出力軸が回るとポテンションメーターも回る
 - ◆ ポテンションメーターの電圧が変化
 - ◆ 電圧をA/Dコンバータで読み取る
 - ◆ 電圧から出力軸の角度が分かる

しかし・・・

- ◆角度取り込み具合が悪いと振動の原因や再現性の悪化となる
- ◆振動が多いと消費電力の増加、ネジの緩み、再現性の悪化、安定性の低下、格好が悪いなど
悪影響大！
- ◆RCサーボのはコストの兼ね合いもあり、角度取り込みの精度はあまりよろしくないっていうか凄くイマイチ

徹底的に対策せよ！

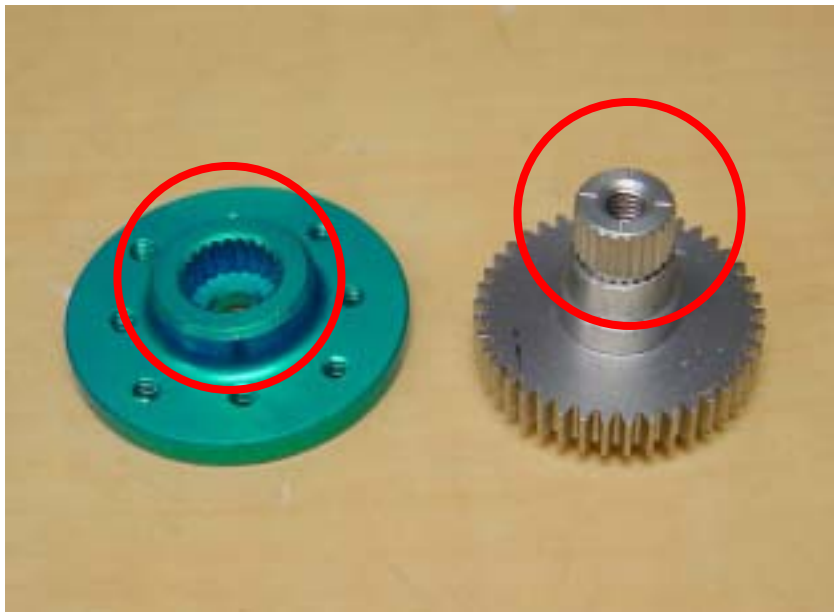
- ◆ 現在角度の取り込みはサーボの命！
- ◆ ここで性能が決まると言っても過言ではない！
- ◆ 「メカ」、「電気」、「ソフト」などあらゆる面から徹底的に対策を行うべし！

メカでの対策

◆振動対策：メカのガタを減らす

◆詰め物を噛ます

◆ネジロック剤で固定



メタルホーンでもガタはある

ガタは振動の原因になる

ネジロック剤は取れなくなる恐れあり

「ネジ」をネジロック剤で

固定するだけでもかなり違う！

メカでの対策

◆再現性の向上

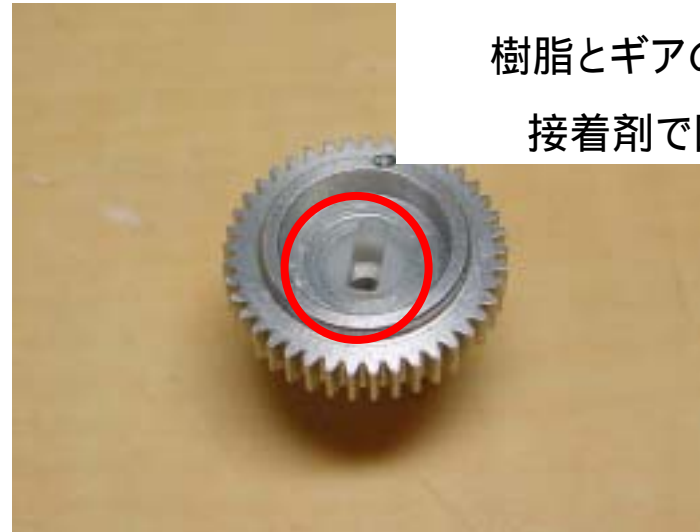
◆ポテンションメーターの軸を固定

◆サーボの出力軸を固定

どんなサーボでもズレる
原点が変わってしまう
半田で確実に固定



ズレにくいサーボも多い
樹脂とギアの間境
接着剤で固定



メカでの対策

◆耐久性の向上

- ◆ポテンションメータに接点グリスを塗る
- ◆ホームポジション付近は磨耗が激しい



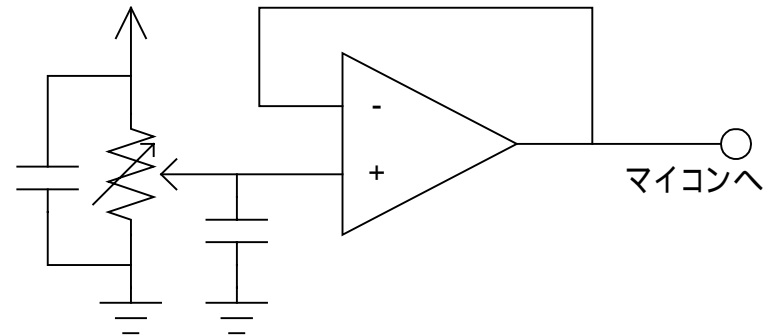
ポテンションは薄いマイナス
ドライバで分解できる
グリスはうっすらと塗る
ゴミや傷に注意

電気での対策

- ◆ 振動対策：ノイズを減らす
 - ◆ オペアンプでインピーダンス変換
 - ◆ Rail to Rail 入出力のオペアンプを使用する



端子の直近にボルテージフォロア

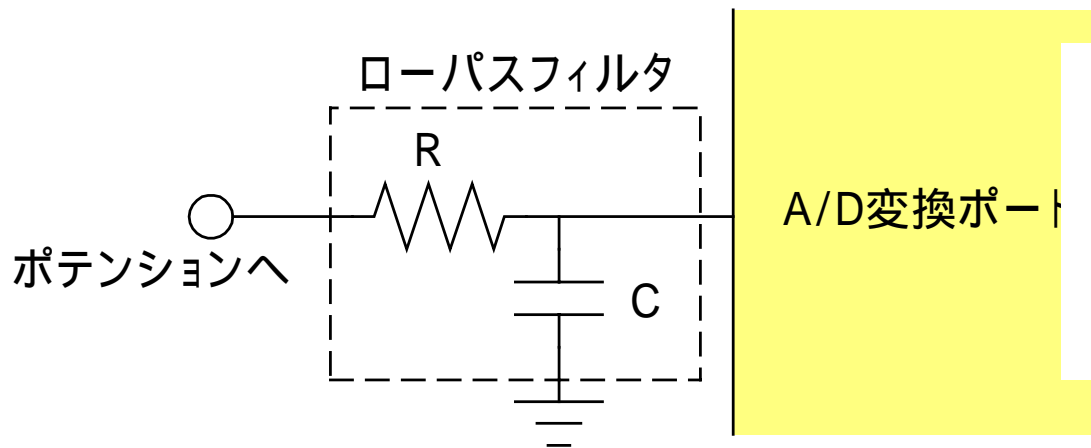


電気での対策

◆振動対策：ノイズを減らす

◆A/Dポートの直近にローパスフィルタ

◆カットオフ周波数を低く設定(数100Hz以下)



端子の直近にローパス

カットオフ周波数

$$f_c = 1/(2 RC)$$

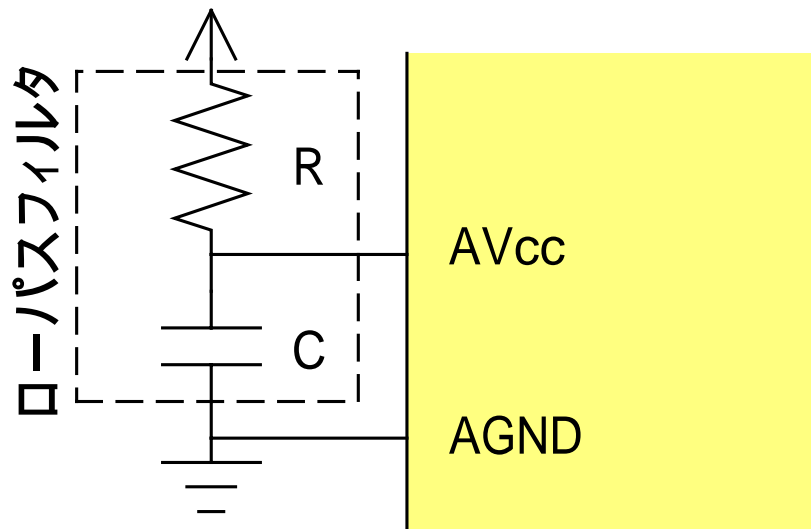
R:1K C:1uF で $f_c=160\text{Hz}$

電気での対策

◆振動対策：ノイズを減らす

◆アナログ電源の分離

◆LCが理想だがRCでローパスでも良い



Rは小さめ(1~5 程度)

端子の直近に配置する

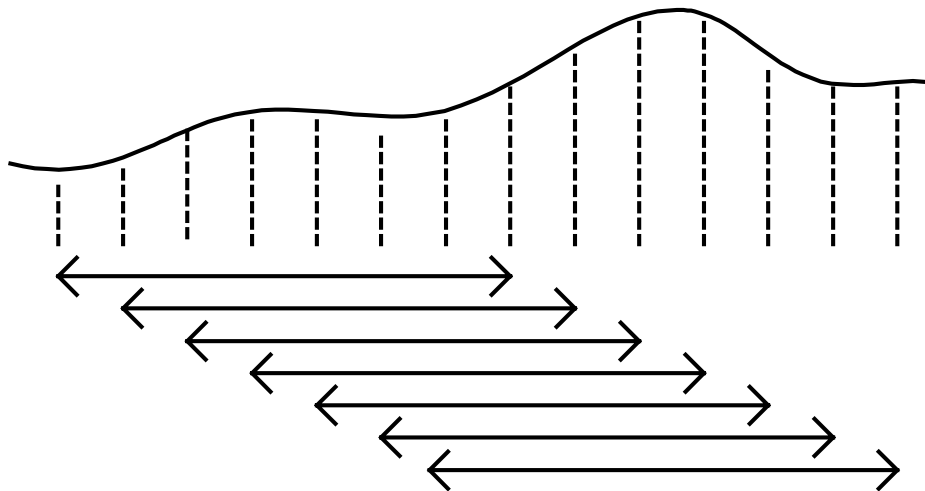
アナログ電源/GNDはデジタル系と
1点で接続

ソフトでの対策

◆ 振動対策: ノイズを減らす

- ◆ A/Dで取り込んだ値を平均化

- ◆ 区間平均をとる(制御周期の8~32倍)



平均を取る範囲をスライドさせていく
合計値から古い値を引いて新しい値
を足すだけ

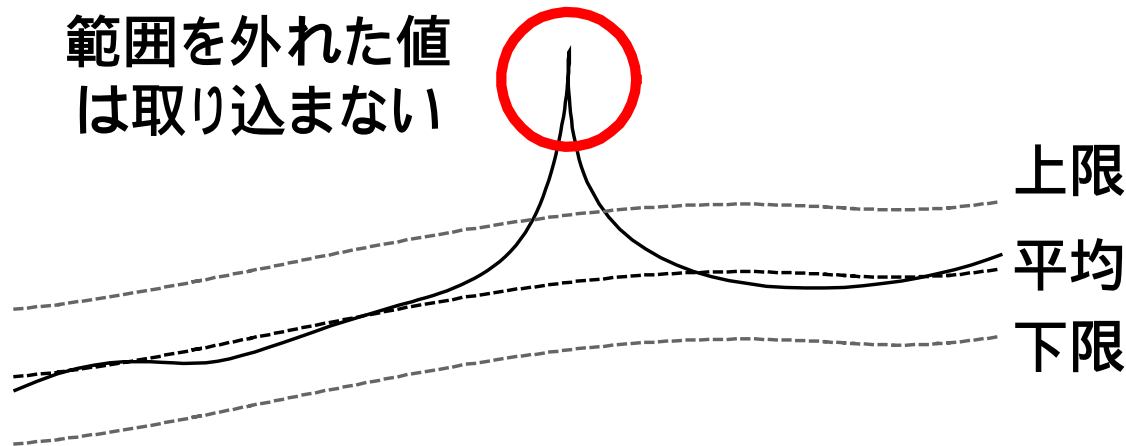
過去の値を区間分保存するためメモ
リを使うが高速

平均を取る範囲を広げすぎると応答
性が悪化し振動する

ソフトでの対策

- ◆ 振動対策：ノイズを減らす
 - ◆ ありえない値を捨てる
 - ◆ 平均値からのズレで判断

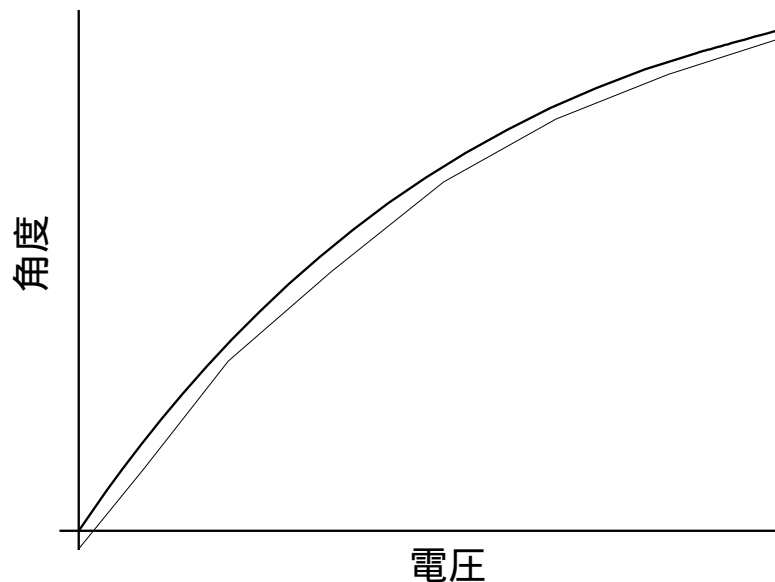
範囲を外れた値
は取り込まない



サーボの最大速度から取り込み範囲を設定
範囲を外れた値は明らかにノイズ

ソフトでの対策

- ◆ 直線性の向上: 角度を正確に取り込む
 - ◆ 電圧-角度変換係数を求める
 - ◆ 複数点で測定して直線性を高める



ポテンションの電圧-角度グラフは曲線
一本の直線で補完した場合は最大で3
度近く誤差が出る

折れ線で補完し誤差を減らす

20点以上測定し16本の直線に変換し
直している

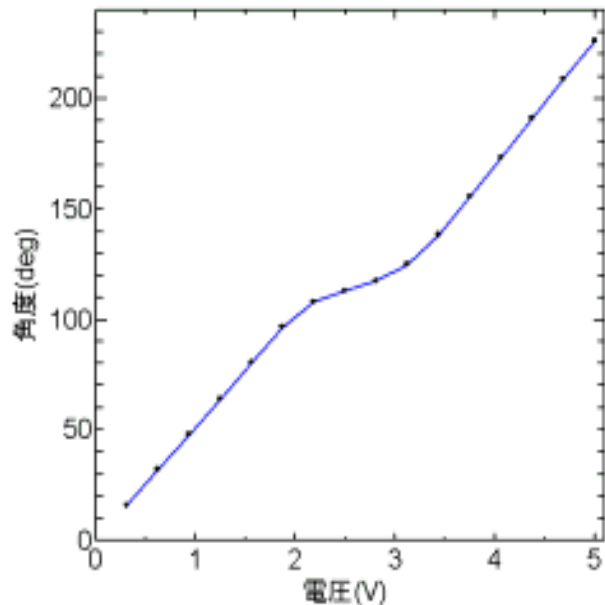
直立と屈伸の両方で足裏の位置がピ
タリと揃う

ソフトでの対策

◆ 標準のポテンションメーター

◆ 磨耗しやすく特性が変わる

◆ 高性能なポテンションを使うしかない



ホームポジション付近は磨耗しやすく
特性が大きく変わる

定期的に電圧-角度変換係数を測定し
直す必要がある

それでも直線では補完しきれない

現在角度の取得のまとめ

◆気付いた所からドンドン対策

◆角度取り込みに聖域なし！
とにかく徹底的に対策せよ！

次の俺サーボでは

- ◆ APLSのポテンションメータを使う
 - ◆ 耐久性、再現性、直線性、稼動範囲の向上
- ◆ 反対軸にポテンションメータを入れる
 - ◆ メカのガタをキャンセル
- ◆ 外付け12bit A/Dコンバータを使う
 - ◆ 精度が段違い！しかも高速！

PIDで位置制御

位置制御

- ◆ 現在角度と指示された角度の差(偏差)から制御量を決める
- ◆ 制御量に応じてモータをPWM制御する

PID制御とは？

◆P制御

◆比例制御：偏差に比例した制御量を出力

◆I制御

◆積分制御：偏差の積分に応じた制御量を出力

◆D制御

◆微分制御：偏差の微分に応じた制御量を出力

◆PID制御 = P + I + D制御

PID制御の例

◆ シンプルで使いやすい

◆ 最も単純な例(細かい工夫は色々ある)

```
e2 = e1; /* 過去の偏差の保存 */
e1 = target_pos - current_pos; /* 偏差の計算*/
e2 = e1 - e2; /* 偏差の微分の計算 */
P = Kp * e1; /* 比例制御 */
I = I + Ki * e1; /* 積分制御 */
D = Kd * e2; /* 微分制御 */
PID = P+I+D; /* 制御量の算出 */
PWM = PID >> PID2PWM; /* PWMのdutyに変換 */
```

PIDゲインの決定

◆制御パラメータ(K_p, K_i, K_d)の決定方法

◆ K_p (Pゲイン)

- ◆振動やオーバーシュートが出るまでとにかく上げる！

- ◆偏差2度前後でフルパワーが目安

◆ K_d (Dゲイン)

- ◆スムーズに動かすために必要

- ◆最大トルクの半分の重りを垂直方向に付け、ゆっくり上げ下げしたときにスムーズに動くことが目安

◆交互にPとDゲインを上げて追い込んでいく

PIDゲインの決定

◆Ki(Iゲイン)

- ◆偏差を減らすのに必要
- ◆0から最大トルクの半分の負荷を一気に掛け偏差1度以内で0.5秒以内に偏差0、最大トルクの半分の負荷を一気に抜いてオーバーシュート0.5度以内で0.1秒以内に偏差0が目安
- ◆ただし素早い動きには付いて行けない場合が多いので良し悪しがある
- ◆Pゲインが十分高いならI制御を行わないのもあり

PID制御の工夫

- ◆GAP(不感帯)は0が基本
 - ◆GAP制御を入れるとガタが多くなる
- ◆I制御(積分)にはリセットウィンドアップ
 - ◆制御量が最大値になったら積分を止める
- ◆D制御(微分)値を増やす
 - ◆区間合計をとるなどして値を増やす
- ◆A/D変換精度は最大限使う
 - ◆10bitA/Dなら10bit！1bitも捨ててはならない！

PID制御まとめ

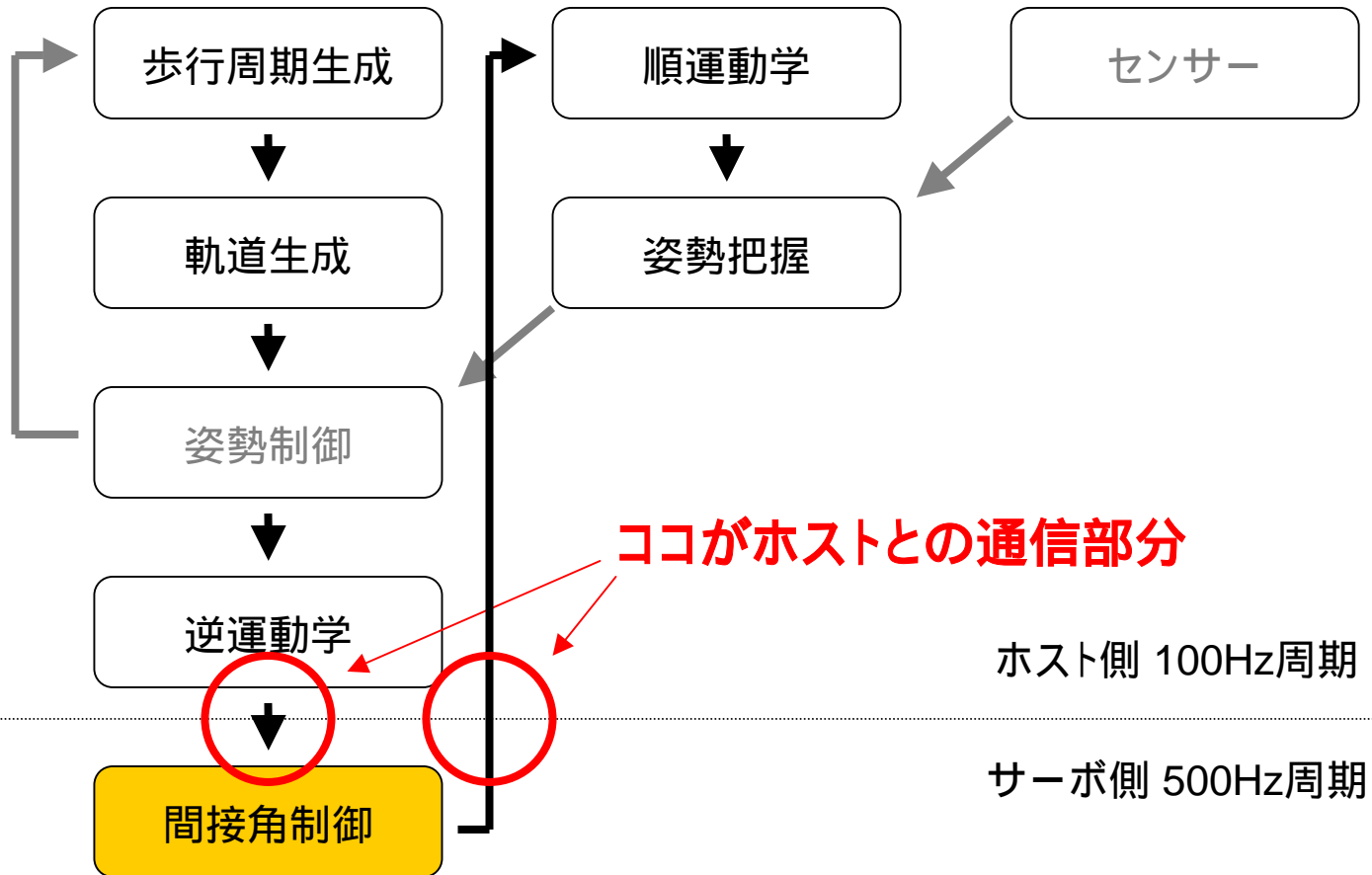
- ◆PID制御はシンプルで使いやすい
- ◆Pゲインを上げると保持トルクが向上
- ◆Dゲインを上げるとスムーズさが向上
- ◆I制御は良し悪しがある
- ◆原理を押さえて色々な工夫を盛り込む
- ◆ゲインが上手く上がらない場合は現在位置の取り込み方法を見直すべし

ホストとの通信

ホストとの通信

- ◆サーボの仕事は位置(間接角)制御
- ◆ホストから指令位置を受信し、現在角度などの情報を返す
- ◆ノイズに強く、信号線数は少なく、安定して十分高い周期での通信を実現する

ロボット制御とサーボの関係



制御用ハードウェア



Host SH2 7047 50MHz

Serial 2ch(4pin)で24軸位置指示

Servo H8 3687 20MHz × 8

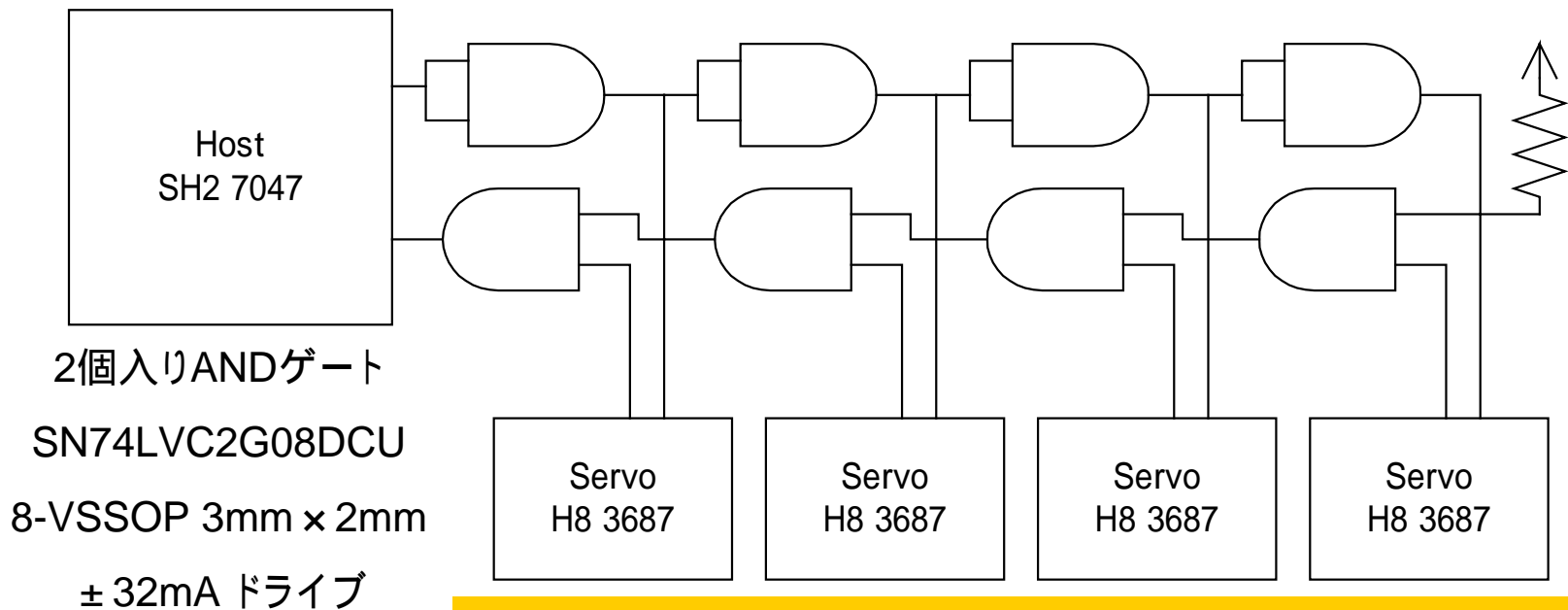
1 boardで 3軸制御



Let's Burning !

インターフェース

- ◆ どのマイコンで使える非同期シリアル(UART)を使う
- ◆ 周辺回路が増えるRS-XXX等のレベル変換は行わない
- ◆ 汎用ロジックICで複数のサーボを数珠繋ぎする



Let's Burning !

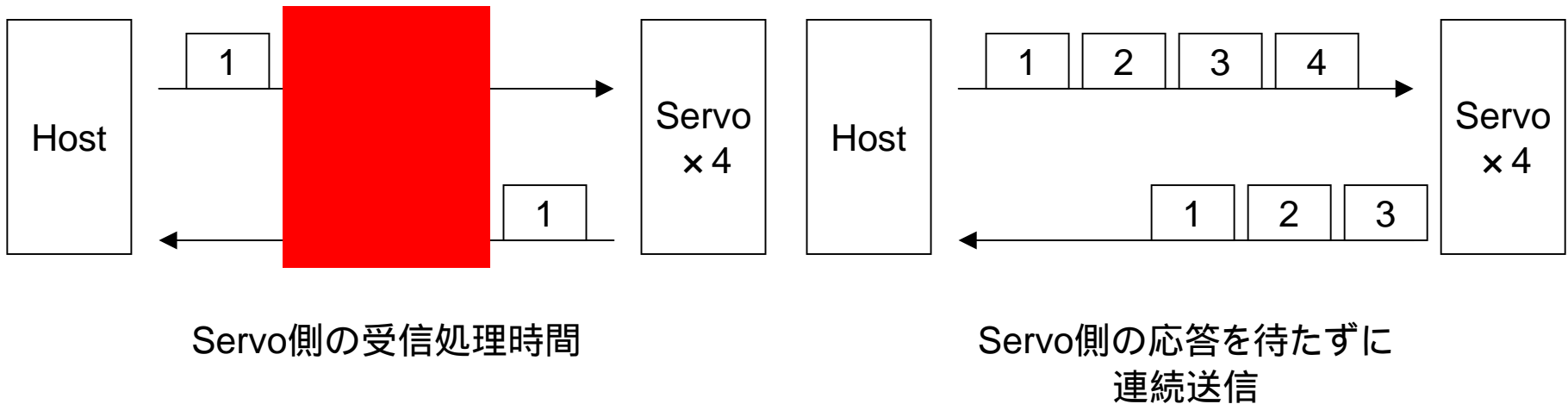
通信速度

- ◆ホストからの位置指示は100Hz以上が目安
- ◆ホストのシリアルチャンネルは多いほど良い
- ◆通信速度は遅いほど安定するし、通信速度も合わせやすい

- ◆156Kbpsのシリアル2chで、24サーボに250Hzで位置の送受信を行った

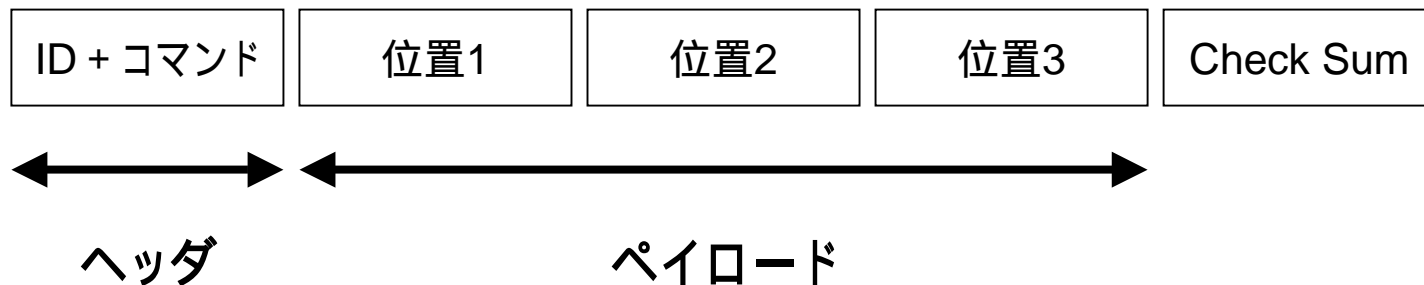
プロトコル

- ◆Req/Ack方式は効率が悪い
- ◆再送は必要ない
- ◆受信を待たずに連続して全2重通信



フォーマット

- ◆ヘッダでサーボIDとコマンドを指示
- ◆ペイロードで角度を指示する
- ◆複数サーボ分をまとめて送ると効率的
- ◆データが1bitでも化けるとひどい目に遭う。チェックサムは必須



ノイズ対策

- ◆ I2Cなどプルアップに頼るI/Fは最悪
- ◆ 差動(RS-485等)が理想だが半二重による通信効率の悪さと信号線数の二律背反
- ◆ 強力なロジックICを使い、全二重通信を実現しつつ信号線数も押さえる
- ◆ GNDと信号線をより線にすると効果的

ホストとの通信まとめ

- ◆少ない信号線で安定した通信を実現する
- ◆回路の工夫で信号線を少なくする
- ◆通信方式の工夫で通信効率を上げる
- ◆ノイズ対策はしっかりと！
- ◆どのマイコンでも使えるUARTを使い通信速度が遅くても実用に耐えられるように工夫する

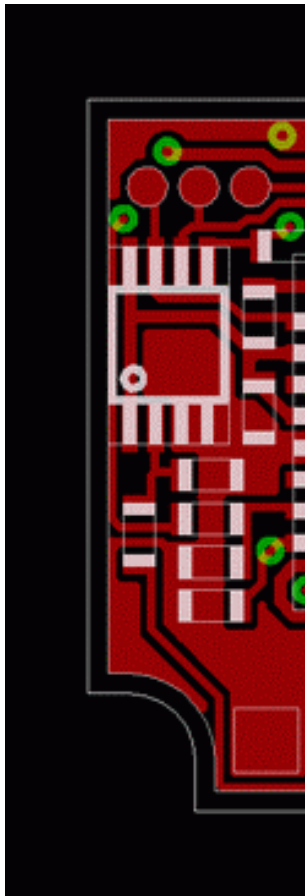
俺サーボの世界

- ◆初めは誰でも素人(私もそうだった)
- ◆やればやるほど奥の深さを実感できる
- ◆RCサーボでは到達できない高性能を実現可能
- ◆メーカーロボ打倒を目指すなら絶対必要！

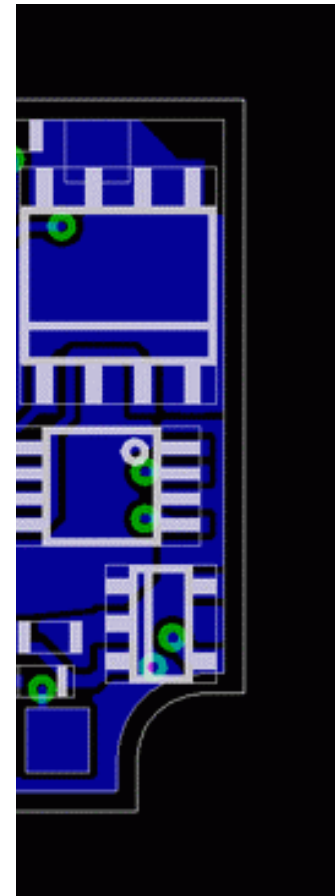
さぁ！あなたも深遠なる俺サーボの世界へ

Let's Burning !

Next ORE-Servo



超小型15.5 x 18.0mm基板
ATmega8L 8bit RISC CPU
高精度外付け12bit A/Dコンバータ
高精度電流検出
電源ノイズ対策強化
高速PWM駆動
4V ~ 16V / 5A 動作
シリアル1.5Mbps / RCサーボ
PID制御
真のコンプライアンス制御
角度、電流、電圧フィードバック
etc ...



Let's Burning !