マルチコプタ公開資料

2016/4/14

東京大学　岩滝宗一郎

東京大学　唐澤宏之

東京都立産業技術高等専門学校　粂田瞭

横浜国立大学　虎谷大地

はじめに

この資料は，飛行ロボットコンテストマルチコプタ部門に参加する機体を製作するための技術資料です．この資料を参考にすると，マルチコプタを自作することができるように書かれていますが，この資料を理解するためには電子工作や材料力学等の基本的な知識を必要とするので，適宜調べながら読み進めて下さい．

飛行ロボットコンテストのマルチコプタ部門は第 11 回大会から開催され，資料作成時はまだ 1 回しか開催されていません．マルチコプタの製作には構造・電子回路・制御等の幅広い工学的知識が必要ですが，自作のための資料はあまり多くありません．そのため入門者が自作しようとするとそのハードルは高く，実際，大会会場で「参加したかったがつくり方が分からず参加できなかった」という声を多く聞きました．そこで本資料は，第 11 回飛行ロボットコンテストマルチコプタ部門参加者有志によって作成されました．資料の内容は実際にコンテスト参加機体を製作したメンバーによって書かれたので，マルチコプタを自作するためにきっと役に立つと思います．

最近では完成度の高い市販のマルチコプタが多く売られており，自作しなくてもマルチコプタを気軽に楽しむことができるようになりました．しかしながら，前述したようにマルチコプタの製作には様々な知識が必要となるため，マルチコプタを自作すると幅広い工学的知識が身につきます．マルチコプタを自作することは大変ですが，実際にものをつくって得た知識は生きた知識として，今後他のものをつくる際にも役に立つので是非挑戦してみて下さい．

マルチコプタ公開資料製作グループ

目次

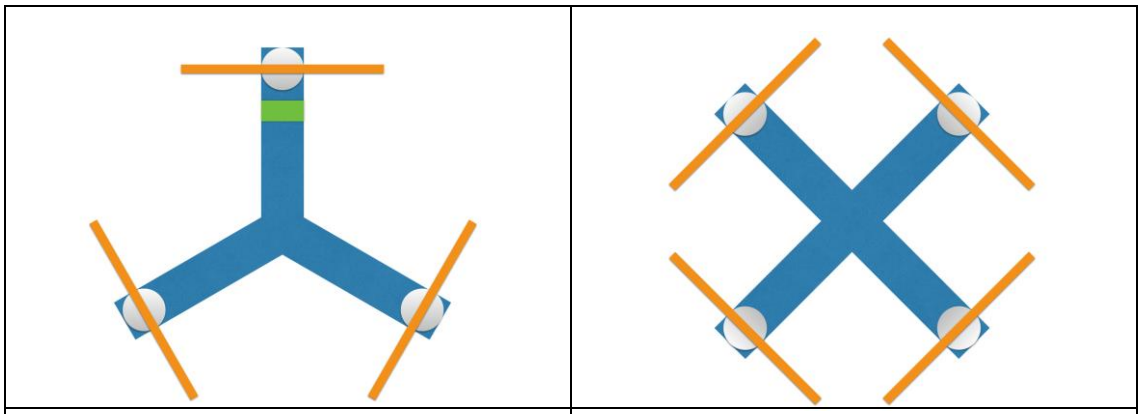
第 1 章　マルチコプタ

マルチコプタ（マルチロータとも呼ばれる）とは，図 1.1 に示すような複数のロータを持つ回転翼機のことである．最近ではドローンと呼ばれることが多く，定義に関しては様々な意見があるが，元々ドローンとは無人の航空機全般を指す用語である．マルチコプタがドローンと呼ばれるのはここ数年，マルチコプタ型無人航空機が一気に広まったからであるが，固定翼機型でも自律飛行を行うことができればドローンであると言える．逆にマルチコプタ型でも自律飛行を行う機能が無ければ単なるラジコンとなるが，最近ではこのタイプもドローンと呼ばれることが多い．本稿は飛行ロボットコンテストのマルチコプタ部門に参加する機体を製作するための技術資料なので，このタイプの機体をマルチコプタと呼ぶこととする．

図 1.1　マルチコプタ

1.1. マルチコプタの種類

マルチコプタはそのロータの数によって分類される．現在一般的なマルチコプタは図 1.2 に示す 4 種類である．



トライコプタ（ロータ 3 つ）	クアッドコプタ（ロータ 4 つ）
ヘキサコプタ（ロータ 6 つ）	オクトコプタ（ロータ 8 つ）

図 1.2　マルチコプタの種類

クアッドコプタ・ヘキサコプタ・オクトコプタでは図 1.3 に示すように，隣り合うロータの回転方向を逆にすることで，ロータが回転する際の反トルクを打ち消し機体がヨーまわりに回転しないようにする．一方，トライコプタでは時計まわり（ClockWise: CW）と反時計まわり（Counter-ClockWise: CCW）のロータを同じ数にできないため，1 つのロータの推力方向を可変にすること等で機体の回転を防ぐ．

なお，トライコプタはロータの数が少なく軽量化しやすいメリットがあり，ヘキサコプタ・オクトコプタではロータが何らかの原因で 1 つ停止しても即墜落しないため，安全性・安定性面で優れているというメリットがある．しかしながら多くのマルチコプタで，ロータ角度の可変機構を必要とせず，ロータの数を少なくできる（効率，保守の観点からロータ数が少ない方が利点が多い）という理由から，クアッドコプタが採用されている．

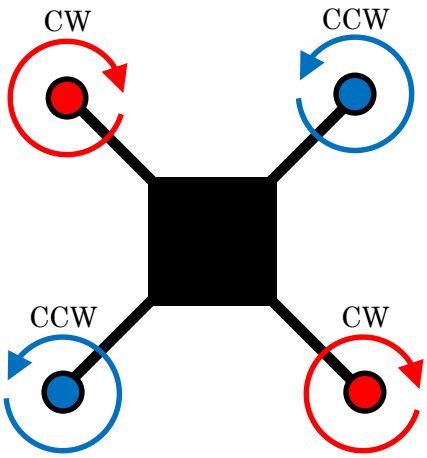


図 1.3　ロータの回転方向

1.2.　市販のマルチコプタ

ここ数年で数多くのマルチコプタが市販されるようになり，その性能は年々向上している．マルチコプタの特徴や運用方法を理解するため，可能であればマルチコプタを製作する前にこれらの市販品を購入して飛行させることを勧める．

DJI 社
PHANTOM シリーズ
http://www.dji.com/jp

Parrot 社
BEBOP シリーズ
http://www.dji.com/jp

3DR 社
SOLO シリーズ
https://store.3dr.com/t/solo-and-accessories

図 1.4　市販されているマルチコプタ

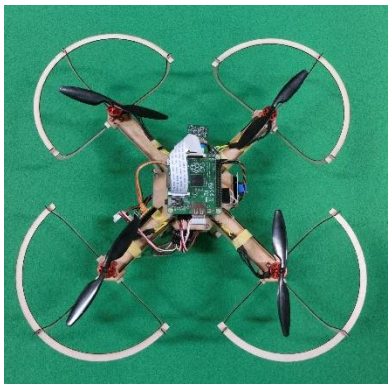
ただし 2015 年 9 月より航空法が改正され，状況によってはマルチコプタを飛行させるために国交省航空局に申請が必要な場合がある．適切な申請を行わないと航空法違反となる可能性があるので，マルチコプタを飛ばす際には安全と法律に十分注意し，自己責任のもと飛行させなければならない [1-1].

1.3. 本資料の構成

マルチコプタを作成するには，センサ等の電子工作，制御，マルチコプタ本体の構造等の工学に関する幅広い知識が必要となる．本稿ではこれらの知識を各要素に分けて解説する．第 2 章ではマルチコプタ設計手順のおおまかな流れについて述べる．第 3 章，第 4 章ではマルチコプタのハードウェア製作に関して述べる．マルチコプタは固定翼のラジコンと比べて，電装系の重要度が高い．そこで本資料では，ハードウェアを第 3 章の電装系と第 4 章の非電装系に分けて解説する．第 5 章ではマルチコプタの制御アルゴリズムとそのプログラミングについて述べる．第 6 章では飛行ロボットコンテストにおけるミッションについて，特に空撮と Rocking Wings について述べる．第 7 章では，マルチコプタを製作するうえで注意すべきこと，起こりやすい問題やその解決策について解説する．

また本資料では，実際のコンテスト参加機体がどのような構成になっていたかのヒントを示すため，東大チーム（Uppsala）と横浜国大チーム(YAL BEST A DRONE）の機体が採用していたパーツやアルゴリズムを，以下のように［東大］,［横国］と適宜示す．

［東大］Uppsala



［横国］YAL BEST A DRONE

参考文献

[1-1] 国土交通省　無人航空機（ドローン・ラジコン機等）の飛行ルール，
http://www.mlit.go.jp/koku/koku_tk10_000003.html，2016/04/10 アクセス．

第 2 章　設計

飛行ロボットコンテストの競技機体を製作するためには，機体レギュレーションに沿った機体を設計する必要がある．またミッションに挑戦するチームは，そのミッションを行うために必要な機構やシステムを設計段階で考慮する必要がある．以下にレギュレーションやミッション機器の例を示す．

- 重量制限
- プロペラガードの装着
- 物資投下機構

2.1.　マルチコプタの設計手順

競技機体を設計するためには様々な要素を複合的に考えながら設計する必要がある．設計の手順は様々だが，ここでは大まかなマルチコプタ設計手順の一例を紹介する．本稿では飛行ロボットコンテスト参加機体に関して述べているため，機体の性能要求は機体レギュレーションを満たしつつ，ミッションを達成できる機体となるが，同じ手順でも性能要求を新たに設定すれば，同様の手順で異なるミッションを行う機体を設計することができる．

a)　マルチコプタの種類の決定

マルチコプタを設計するためには，第 1 章で紹介したトライコプタ・クアッドコプタ・ヘキサコプタ・オクトコプタ等のマルチコプタの種類を決定する必要がある．この種類を決めることで，大まかな形はもちろん，必要なモータや ESC（Electric Speed Controller の略，スピードコントローラ，アンプとも呼ばれる）の数，フライトコントラーラの性能（制御できるモータの数）などが決まる．

b)　候補となる部品の選定

マルチコプタの種類が決定したら，次に機体に使用する候補となる部品の選定を行う．（各部品選定の詳細に関しては後述）必要な部品や材料の例を以下に示す．

- フレームの材料（金属，樹脂，木材等）
- フライトコントローラの材料（基板，マイコン，センサ，その他電子部品等）
- モータ
- ESC
- プロペラ
- バッテリ
- 送信機と受信機

また機体の設計には関係ないが、ラジオペンチ，ニッパ，はんだごてなど組み立てに必要な工具も一緒に準備しておくとよい．

c)　重量の検討

機体をレギュレーションの重量制限内に収めるため，また必要な推力を計算するためには機体の重量を設計段階から考慮する必要がある．まずは b)で選定した部品を元に重量計算を行う．ここでは例として，東大チームの機体の内訳を示す．

表 2.1　重量の計算［東大］

部品名	重量［g]	部品名	重量［g]
フライトコントローラ基盤	16	受信機+ケーブル	20
木製フレーム+接着剤	30	シリアル変換基板	3
構造用スペーサ+ねじ	11	電源基板	10
(モータ+ESC+プラグ)x4	57	投下装置	10
プロペラ x4	5	カメラ（Raspberry pi)	33
バッテリ	52	プロペラガード	20
モータ固定用ボルトナット	5	カウンターウェイト	20
		合計	292

全体の重量が重量制限内におさまるよう，フレームの目標重量の決定，各部品の見直しなどを行う．

d)　機体設計

全ての部品の選定を終えたら，機体の設計に進む．最初に決定したマルチコプタの種類を元に大まかな形状を考え，そのあとモータや ESC・フライトコントローラ・バッテリ等の取り付けや配線の取り回しを考える．ミッションに挑戦するチームは，そのミッションに必要な機構等も追加する．

実際の設計ではこの手順 1 回で機体が要求を満たすことはあまり無いため，この手順を繰り返し検討することで，機体レギュレーションを満たしつつ，ミッションを達成することができる機体を製作する必要がある．

第 3 章　ハードウェア（電装系）

この章では実際にマルチコプタを製作するうえで必要となる電装系ハードウェアについての説明を行う．マルチコプタは固定翼機と比べ，電装系の占める割合が大きい．そこで本稿ではハードウェアを大きく電装系（電子回路等）と非電装系（機体フレーム等）に分けて解説を行う．

3.1.　電装系の構成

マルチコプタを作成するうえで必要になる電装系について説明する．マルチコプタの飛行に必要な電装系の構成を図 3.1 に示す．パイロットが送信機（プロポ）から送信した指令は機体の受信機で受信され，指令に従って制御ボードがモータの出力を計算し，機体の姿勢などをコントロールする．制御ボードからの出力は直接モータを回転させることはできないので，ESC が制御ボードからの信号に従ってモータを駆動する．また各コンポーネントに必要な電力を供給するバッテリ，電源回路も必要である．

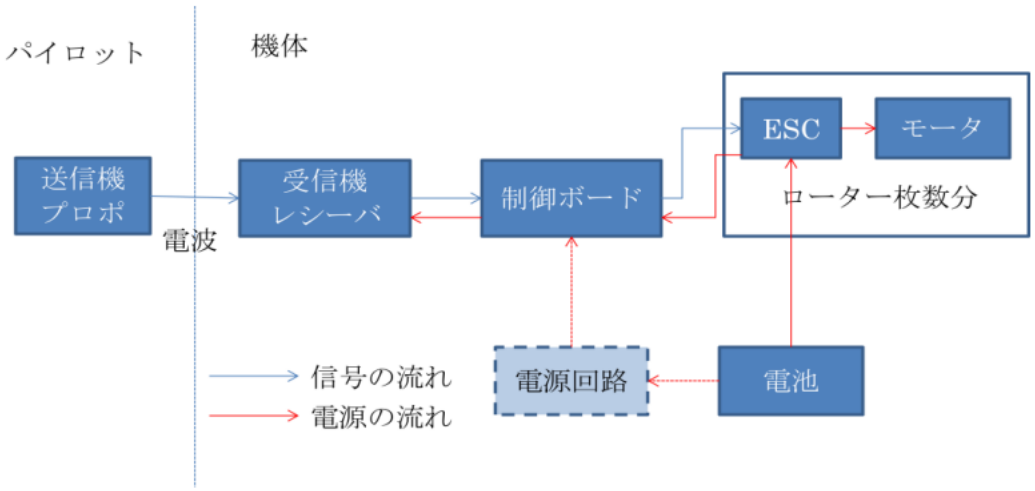


図 3.1　電装系の構成

3.1.1.　ESC とモータ

マルチコプタの飛行に必要な推力はすべてモータにより生み出されるため，モータの選定は重要である．飛行ロボットコンテストではバッテリを動力源とするので，直流電源で駆動できるモータが用いられることが多い．そのようなモータには，「ブラシ付き DC モータ」と「ブラシレスモータ」がある．ブラシ付き DC モータとブラシレスモータはそれぞれ長所と短所があり必要に応じて使い分けられているが，コンテストにおいては入手性の良さからブラシレスを使うチームが多いため，以下ではブラシレスの場合について説明する．

ブラシレスモータはブラシ付き DC モータとは異なり，3 本の線を有し（ブラシ付き DC

モータは＋と－の 2 本), モータを回転させるためには ESC という回路基板が必要となる. 3 本の線をそれぞれ ESC に接続する必要があるが，これら 3 本の線には＋や－といった区別は無く，3 本中の 2 本の線を入れ替えることによって回転方向を逆転させることが出来る．ESC を選定するときは以下のことに気を付ける.

- モータのタイプ：ブラシレスモータにはブラシレス用の ESC が必要
- モータの消費電流：モータの最大電流以上の定格が必要
- 電源電圧：使用するバッテリに適合したものが必要
- BEC・OPTO : BEC は ESC 基板上に電源電圧から 5V の電圧を発生させる回路が搭載されている．その電源を使ってサーボモータや制御ボードを動作させることができる．OPTO には外付けの電源回路が必要となる.

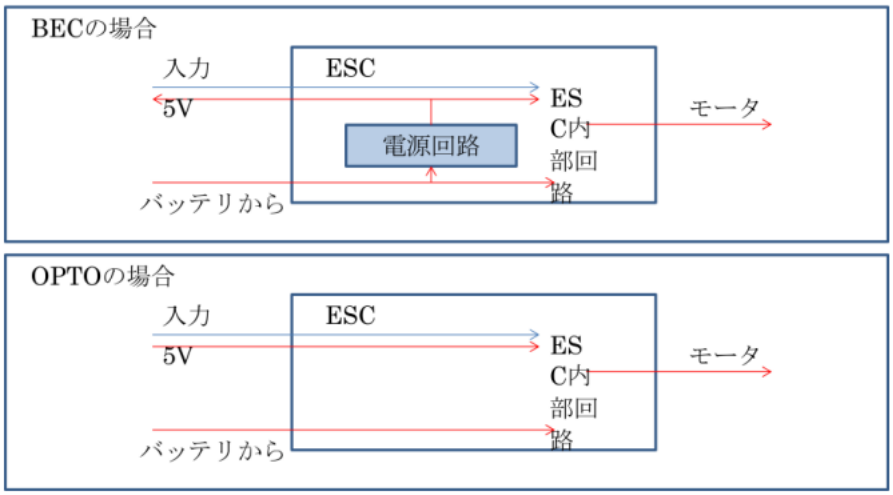


図 3.2　BEC と OPTO の違い

［東大］モータ：Turnigy 1811 brushless Outrunner 2900kv，ESC：5.8 A
［横国］モータ：コスモテック CT1811-2000 2000kv，ESC：コスモテック CT-7A

3.1.2.　送信機と受信機

操縦者のコマンドを機体へ伝達するために送信機（プロポ）と受信機が必要となる．飛行コマンドには基本的に 4 チャンネル（スロットル，ロール，ピッチ，ヨー）必要で，その他にミッション用に何チャンネルか必要となる．ルールによって必要なチャンネル数は変わってくるので，十分なチャンネル数のものを選んでほしい.

［東大］8ch のものを使い, 2 つのスイッチを物資投下とロッキングウイングに割り当てた.
［横国］Futaba R6106HFC (6ch)，チャンネル構成は東大と同じ.

3.1.3. 制御ボード

マルチコプタを飛行させるには自動制御による機体姿勢の安定化が必須である．センサの情報と，操縦者からのコマンドに基づきモータへの出力を行う部品が制御ボードである．制御ボードには数多くの製品が販売されているが，自作することも可能である．大会で課せられるミッションを遂行するためには，市販基板を用いる場合でもソフトウェアの変更が必要になる場合がある．

3.1.3.1. 市販の制御ボードを改造する場合

市販の制御ボードには Multiwii や PixHalk というオープンソースハードウェアのものがある．これらのボードは Arduino*ベースのシステムで，プログラムの変更は PC 上の Arduino IDE で可能となっている．（PixHalk は以前のバージョンまでが Arduino ベースのボードであったが現在は異なる．ただしプログラミングは Arduino IDE で可能.）また専用 GUI を用いて飛行制御のパラメータを書き換えることもできる．制御ボードを自作する場合については後述する．

* Arduino : マイコンボードの一種．初心者でも使いやすいように様々な工夫がされており，様々なセンサをつなぐためのライブラリも充実している．

3.1.4. バッテリ

バッテリには軽量かつ大容量のリチウムポリマバッテリ（LiPo）が用いられることが多い．バッテリの電気的な性能は次の 3 つの指標で表される．

- 公称電圧［V］: バッテリの電圧を表す．LiPo の場合，この値はセル数［S］で決まることが多い．通常の LiPo は 1 S あたり 3.7 V なので，2 S で 7.4 V，3 S で 11.1 V となる．
- 容量［mAh］: バッテリが，ある電流をどのくらいの時間流せるかを表す.
- 最大放電電流［C］: そのバッテリから瞬間的に取り出すことのできる電流を表す.

例えば（2 S / 7.4 V, 850 mAh, 25 C）というバッテリがあったとすると，このバッテリは 2 S で公称電圧 7.4 V，850 mA の電流を 1 時間流せる容量を持ち，最大 25 C つまり 850×25 = 21250 mA の電流を瞬間的に流すことができるということを表している．公称電圧の上限は大会レギュレーションによって決まっているので，十分な飛行時間が確保できるように容量を選び，また飛行時に流れる電流がバッテリの最大放電電流を下回るように選定する．一般に大容量のバッテリほど重く寸法も大きいので，ミッション用の機材や他のコンポーネントとのバランスを考え設計することが重要となる．

［東大］HYPERION リチウムポリマーバッテリ, 25 C, 2 S, 850 mAh
［横国］HYPERION リチウムポリマーバッテリ, 25 C, 2 S, 850 mAh

3.1.5. 配線

電源の配線を図 3.3 に示す．BEC 搭載の ESC を使用する場合，受信機や制御回路へは一つの ESC の電源出力を使用し，ほかの ESC と制御回路との接続にはグラウンドと信号線のみを配線する．これは各 ESC がバッテリから作り出す 5V 電源電圧にわずかな誤差があった場合，電圧の低い ESC へ電流が逆流し故障する恐れがあるためである．BEC の供給できる電流には限りがあるので，RaspberryPI など大電流を必要とする回路には別個に電源回路を用意するのが望ましい．また ESC が OPTO の場合には電源回路が別途必要である．配線は確実に接続し，熱収縮チューブなどで確実に絶縁する．

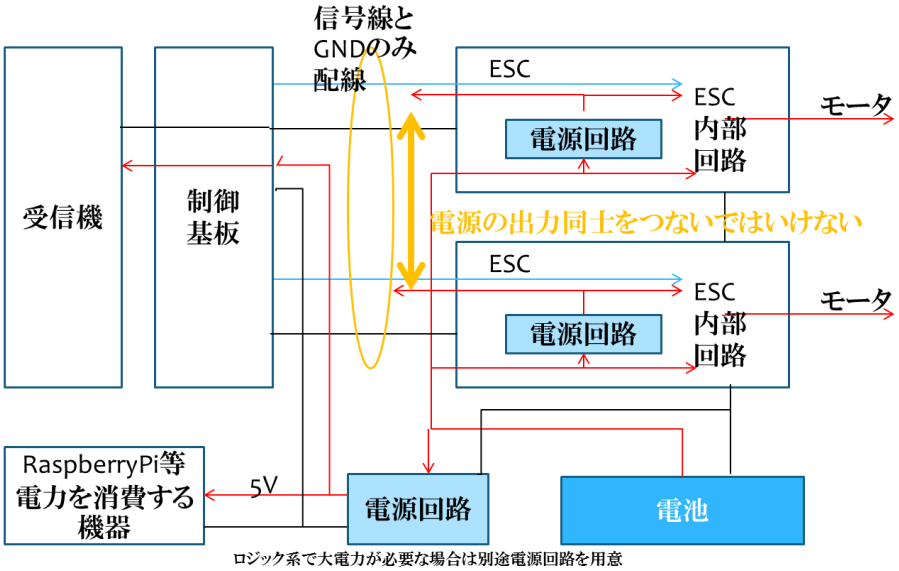


図 3.3　電源の配線

3.2. 制御ボードの自作

制御ボードを自作する場合には以下に示すような構成要素が必要となる．

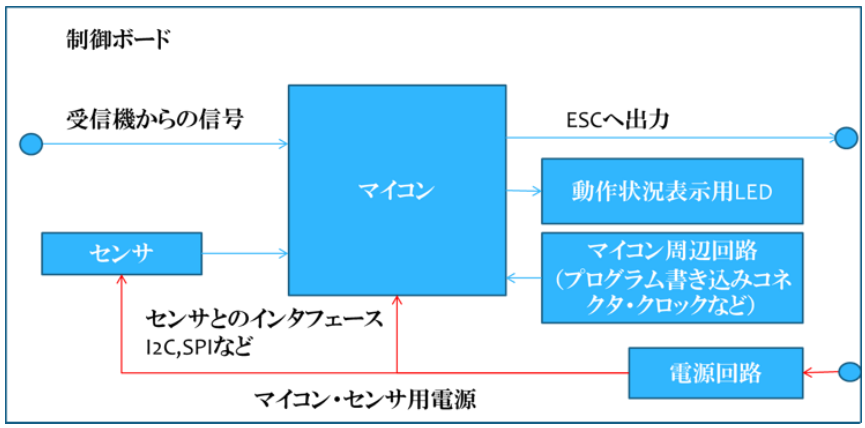


図 3.4　制御ボードの構成

- マイコン：受信機・センサから値を読み取り制御を行う.
- 電源：マイコン・センサに電力を供給する.
- センサ：機体の状態を計測する.

オプションとして以下のような要素もある.

- LED：動作状況の表示.
- デバッグ用インターフェース：マイコンへのプログラムの書き込み・開発中のデバッグ（動作確認）用.

3.2.1. マイコン

機体を安定化させたり，自動制御したりする制御アルゴリズムを実行する．Arduino や mbed**などのマイコンモジュールを使うと周辺回路がすでに完成しており，便利なライブラリが充実しているので使いやすいが，制御ボード全体の重量は重くなってしまう．一方，マイコン IC 単体を使って基板を一から設計することも可能ではあるが，その場合はマイコンを動作させるための周辺回路も自作する必要がある．その代わりに小型で軽量な回路を作ることができる.

** mbed：マイコンボードの一種．Arduino 同様，初心者でも使いやすいように様々な工夫がされている．開発環境がクラウド上にあるという特徴がある.

［東大］主制御基板：STM32F401，受信機デコード基盤：LPC810
［横国］mbed LPC1768

3.2.2. センサ

機体を安定化させるために，機体の角速度や加速度を計測する．数年前までは機体の角速度を計測するためにジャイロセンサを 3 つ（3 軸），加速度を計測するために加速度センサ 3 つ…と多くのセンサを必要とした．しかし近頃は 3 軸の加速度と角速度，地磁気がワンチップにまとまった，非常に省電力・省スペースなセンサ IC が出回っている．制御ボードを自作する場合は是非検討してほしい.

［東大］LSM9DS0　［横国］MPU-9150

3.2.3. マイコンのプログラミング

制御プログラムを構成するとき，プログラムの構造の見通しを良くするために制御アルゴリズムと，実際のハードウェアへの入出力を行う部分を分離するとよい．つまり，制御アルゴリズムはセンサ値やプロポの指令値が分かっているものとして動作し，「どのようにセンサから値を取り込むか」とか，「どのように受信機の値を読むか」という部分とは分けて考える．こうすれば，例えば，あとでセンサの機種を変更した場合にも制御アルゴリズムは

大きく変更しなくてよくなる．本章では受信機やセンサ，ESC とのインターフェース方法について説明する．

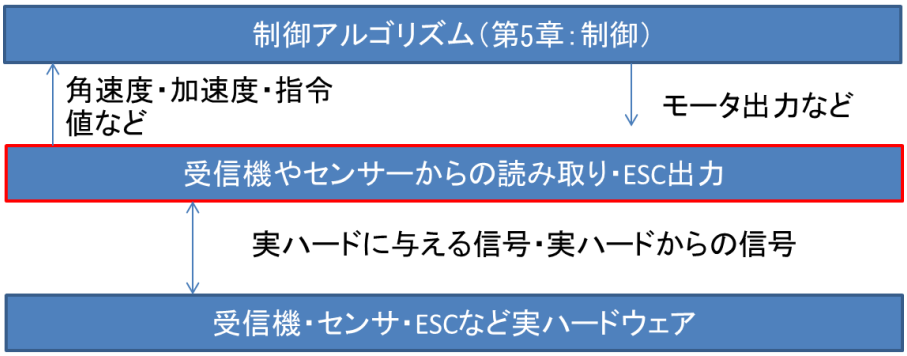


図 3.5　この章で扱う範囲（赤枠部分）

3.2.3.1.　タイマ割り込み

機体の制御を行うためには，図 3.6 に示すように制御周期ごとにセンサ値の読み取り，計算，ESC への出力を実行する必要がある．そのような一定の時間間隔で処理を行う際に用いるのがタイマ割り込み***である．

*** 割り込み（Interrupt）：通常，プログラムは書いてある順に実行されるが，ある条件が発生したときに現在実行しているプログラムを中断して，別の処理を行い，その処理が終了したら元のプログラムに復帰するという動作を行わせることを割り込みという．割り込みを発生させる条件はタイマのほかにも，入力の変化，通信の完了などを設定できる．どんな割り込みが使えるかは使用するマイコンによるので，データシートや解説書で調べてほしい．

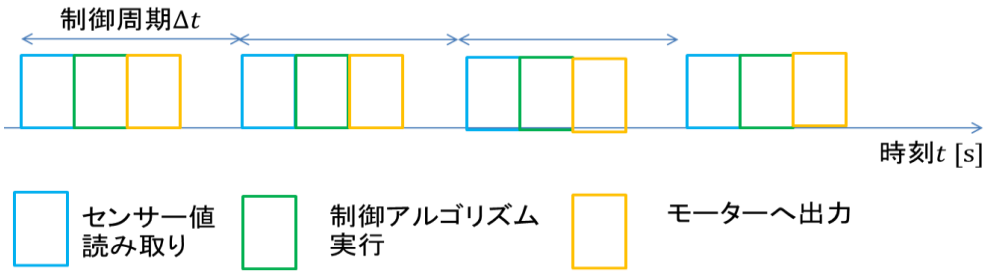


図 3.6　制御周期の考え方

Arduino や mbed ならタイマ割り込みのライブラリを探してきてそれを使用するのがよい．標準の loop 内で delay()や wait()といった関数を呼ぶやり方でも同様の動作をするプログラムを書くことができるが，その方法では delay や wait 以外の処理にかかる時間が正確に見積もれず，あまり正確な制御周期が設定できないため勧めない．ARM 系のマイコンならば

Systick 割り込みを用いて一定時間ごとに制御を実行する方法がよい.

3.2.3.2.　ESC への出力

ESC への指令は，図 3.7 に示すように一定周期のパルスの High の長さを変化させることで行われる．このような信号はマイコンのタイマ機能に「PWM 出力」機能があればそれを用いて実装することが可能である．Arduino なら Servo クラスを，mbed なら PwmOut クラスを使えば容易に実装可能である．ただし，これらのクラスで使用するタイマと制御周期生成に用いるタイマが重複しないように注意する．

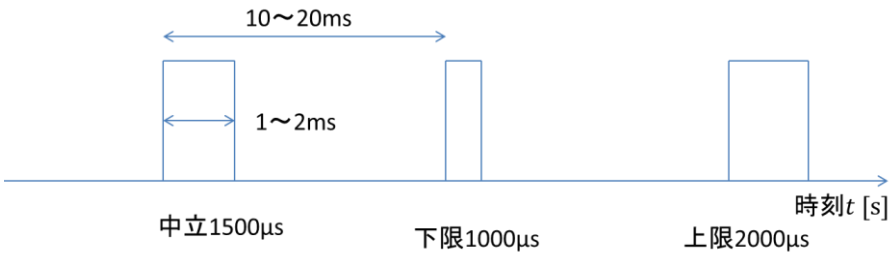


図 3.7　ESC へのパルス信号

もし PWM 出力の数が不足する場合は, パルス周期がパルス幅に比べて長いことを利用し, 1 つのタイマで複数の RC パルスを発生させることもできる．図 3.8 にその方法を示す.

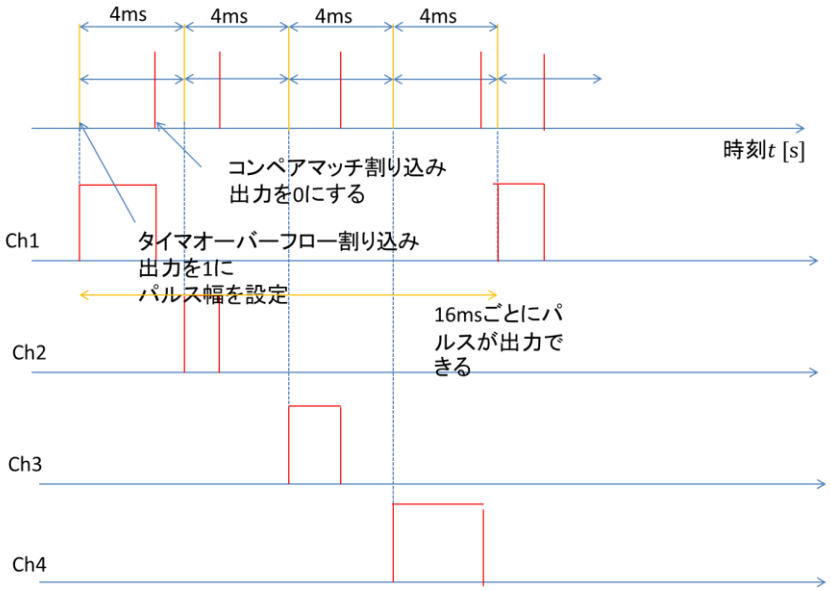


図 3.8　1 つのタイマを用いて複数のパルスを発生させる方法

図 3.8 はコンペアマッチ割り込みとオーバーフロー割り込みを用いて一つのタイマで複数のパルスを発生させる例である．まず，タイマは 4 ms でオーバーフローするように設定しておき，オーバーフロー割り込みで Ch1 を High にする．同時に Ch1 で出力すべきパルス幅だけ時間が経過したときにコンペアマッチ割り込みが発生するようにする．コンペアマッチ割り込みでは Ch1 を Low に戻す．次のオーバーフロー割り込みでは Ch2 に対し同様の操作を行う．この例では 16 ms 周期でパルスが出力されている．

3.2.3.3.　受信機からの入力

受信機が出力する信号は図 3.7 の信号と同様なパルス信号である．使用するマイコンに「PWM 入力」機能があればそれを用いて実装できる．PWM 入力機能がない場合は図 3.9 に示すような「インプットキャプチャ」機能を用いる．これは入力ピンの状態が変化するとそのときのタイマカウンタの値を保存する機能である．要するに立ち上がり，立ち下りそれぞれの時点でのタイマカウント値を得ることにより，パルスの幅を求める．この場合入力 1 つにつき，2 つのキャプチャ機能を用いる．

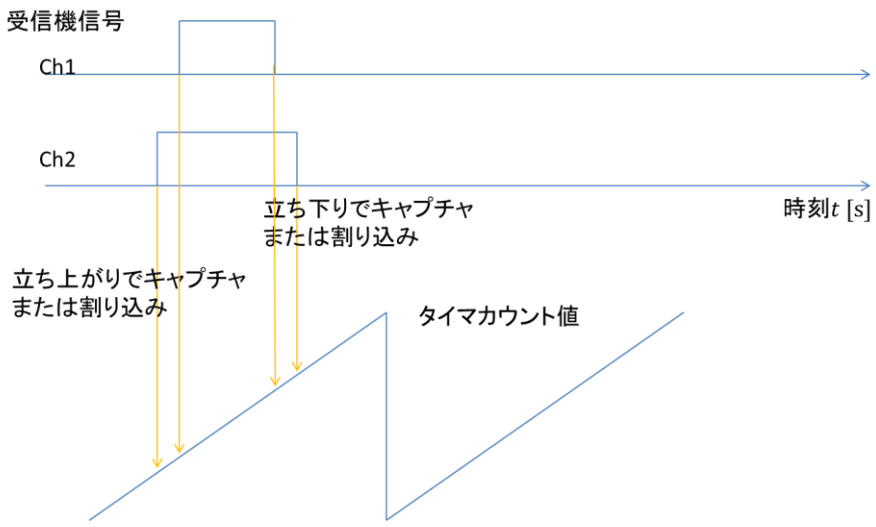


図 3.9　受信機信号の読み取り

3.2.3.4.　センサ値の読み取り

数年前までセンサはアナログセンサが主流であったため，センサ値の読み取りには AD 変換を行っていた．しかしながら，前述したような近年主流のセンサ IC はデジタルセンサと呼ばれるものであり，CPU との接続は主にシリアル通信バスの一種である I2C または SPI で行われる．I2C は別名 TWI (Two Wire Interface) とも呼ばれ，図 3.10 に示すように 2 本の配線とプルアップ抵抗というシンプルな回路で使用できるというメリットがある．通信速度は 100~400kHz 程度と低速である．SPI では通信速度は 10MHz 以上と高速だが，

MISO, MOSI, SCK の 3 本が必要で，さらに通信相手を指定する SS 信号線がセンサの数だけ必要になり，配線が多いというデメリットがある．

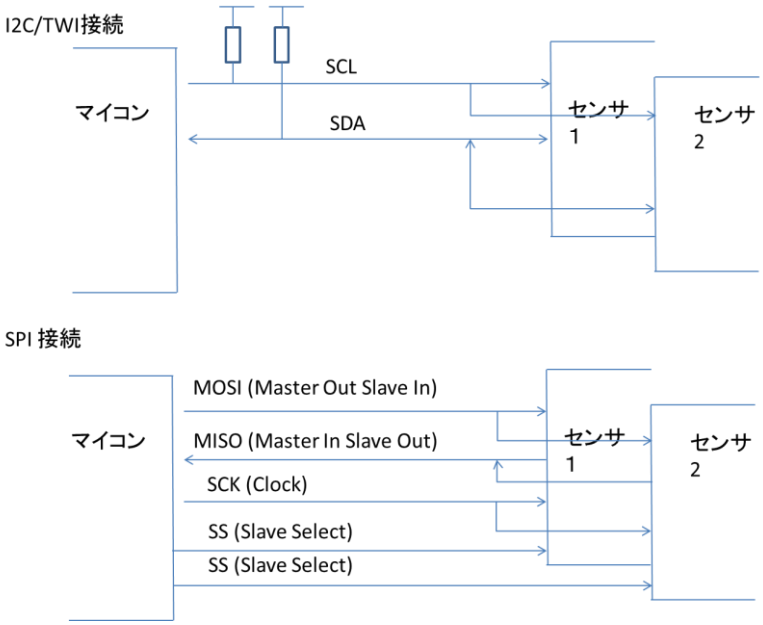


図 3.10　マイコンとセンサの接続

使用するセンサや必要な通信速度に応じて通信方式を選ぶとよい．I2C，SPI で通信を行う際，基本的にはマイコンに内蔵されている通信モジュールを用いる．特に Arduino や mbed を用いる場合，センサの型番とマイコン名で検索すると，インターネット上にサンプルプログラムが公開されている場合があるので，それらを参考にするとよい．

参考文献

- ARM マイコンの使い方関連

[3-1] ARM 特集記事，トランジスタ技術，2011 年 3 月号，2012 年 12 月号，2014 年 2 月号．

[3-2] 川内康雄，“STM32 マイコン徹底入門（TECH I Processor)”，CQ 出版，2010.
ARM の CPU を搭載した STM32 というマイコンの説明をしてある本，解説サイト(http://miqn.net/)も充実．

[3-3] ねむいさんのぶろぐ，http://nemuisan.blog.bai.ne.jp/，2016/04/11 アクセス．
ARM マイコンを使う上で有益な情報がある．

- 電子回路設計関連

[3-4] 大類重範，“アナログ電子回路”，日本理工出版会，1999.
アナログ回路（増幅回路など）の説明，飛行ロボコンではあまり出番がないかも．

[3-5] 白土義男, “図解　デジタル IC のすべて”, 東京電機大学出版局, 1984.
若干古いがデジタル回路の基礎について知ることができる．特に IC を使う際の回路的な注意点などが参考になる．

第 4 章 ハードウェア（非電装系）

ここでは，マルチコプタを構成するハードウェアの中で電装系以外のものついて解説する．非電装系には構造系であるフレームや，駆動系であるプロペラやモータがある．また機体によっては，プロペラガードや推力方向制御機構といった要素を持つ場合もある．

4.1. フレーム

フレームは機体の最も基礎となる部品であり，マルチコプタの各部品を保持する役割がある．またフレームは各部品を保持する剛性を持つ必要があり，そのまま機体の構造となる．図 4.1 に示すように，フレームは大きく分けて制御基板やバッテリなど電装系が配置される中央部と，モータとプロペラを保持するアーム，地面との接点になる脚により構成される．

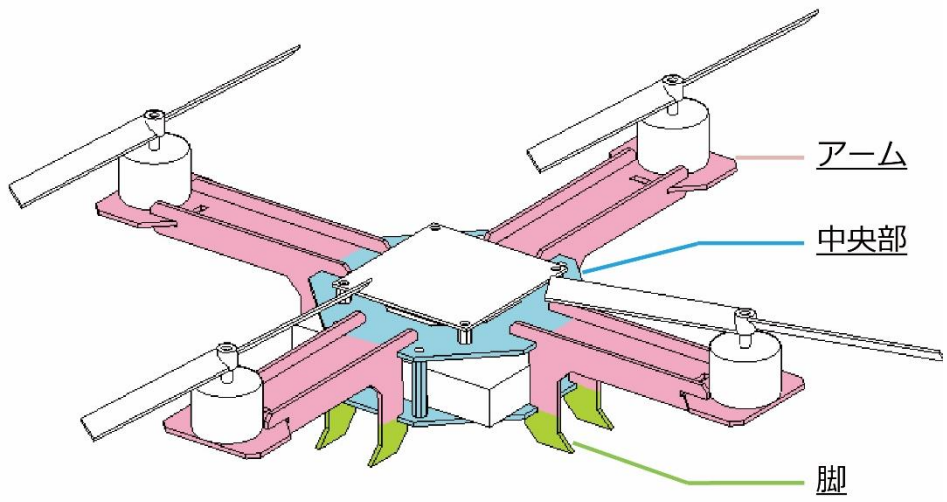


図 4.1　フレーム構成要素の機能による分類

4.1.1.　フレームの構成要素

a) 中央部

中央部は幾何的に機体の中央にあり，電装部品（制御基板や受信機，バッテリ）を保持する部分である．電装部品を保持するための面積が最低限必要となる一方で，あまりにも大きな面積を取り過ぎると重量が増えてしまうことに加えて，プロペラの回転面と中央部が重なってしまい，プロペラが発生する気体の流れを阻害してしまう可能性があるため，あまり大きすぎてはいけない．

b) アーム

アームは中央部から突き出している構造で，基本的にはアーム 1 本がモータ 1 個とプロペラ 1 個を保持している．アームの剛性が小さいと，変形や振動が大きくなりそれだけ制

御に悪影響を与える可能性がある．また中央部とアームが一体化して区別がなく，中央部が直接モータを保持する機体も考えられる．

c) 脚

脚は地面に接地し，上部にある中央部・アームを保持する要素であり，着陸時にかかる衝撃に耐えうる強度を持つ必要がある．脚は (i) 各アームの下につける，(ii) 機体の中央部につける，の 2 方式があり，図 4.1 の例では(ii)の方式を採用している．(i) の方式では脚を比較的外側につけられるため，着陸時の安定性を増すことができる．一方，アームは中央部に比べて歪みやすいため，(i) の方式では着地時の衝撃でアームが歪んでしまう可能性があるので，その場合には (ii) の方がよい．また脚の長さに関しては，脚が短いほど構造の強度が大きく重量が小さいというメリットがあるが，地面効果を受けやすくなり離着陸時の制御が難しくなるというデメリットもある．

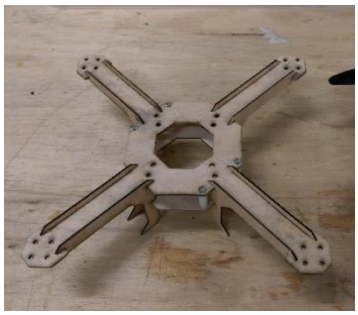
これらの要素に加え，後述するプロペラガードも機体の剛性を担保することがあり，その場合は構造の一部といえる．

4.1.2. 材料

フレームに用いられる材料には，マルチコプタの空を飛ぶという要求機能から比強度（材料の引張強さを質量で除したもの）が高く，また学生が扱えるものという要求機能から入手性が良く加工がしやすいものが用いられる．これらの条件を満たす材料としては代表的なものでプラスチック（ABS，アクリル），木材（航空ベニア），強化プラスチック（CFRP），金属（アルミニウム）がある．材料の機械的特性や加工方法に関しては飛行ロボットコンテスト Webpage の「機体製作のアドバイス」を参考にされたい[4-1].

飛行ロボットコンテストにおいて，マルチコプタは固定翼機と異なり，通常仕様においても衝撃荷重が加わる．衝撃荷重を受ける部分では強度の出にくい接着ではなく，ネジによる締結や 3D プリンタなどによる一体成型構造を使うのが好ましい．

［東大］航空ベニア（t1.5 と t1.0）



［横国］中央部・脚：アルミ発泡材ハニカム板
アーム：アルミニウム

4.1.3.　フレーム設計の際のポイント

a)　軽量化に向けて

構造の重量を減らしつつも強度を出すためには，断面二次モーメントの考え方が役に立つ．構造は材料の分布を曲げの中心から遠ざければ遠ざけるほど曲がりにくくなるという性質があり，その曲がりにくさを数値化したのが断面二次モーメントである．例えば図 4.2 に示す断面積が同じ二本の棒に同じ力をかけることを考えると，中が中空で材料の分布が中心から遠い (ii) は，中がびっしり詰まっている (i) の変形を 1 とすると 0.35 しか変形しない．また，棒にかかる最大応力（材料の破壊に関係）も (ii) は (i) を 1 とすると 0.52 と小さくなる．この考え方は図 4.3 のように建物の構造材の設計にも用いられている．

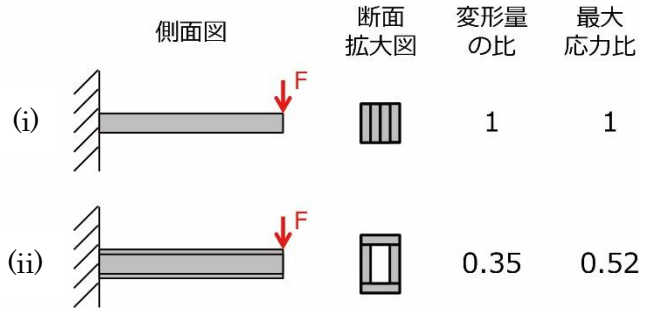


図 4.2　材料の分布と変形・最大応力との関係

図 4.3　軽量化の例（構造用 H 型鋼）

b)　パーツ配置

設計の際は部品の配置や配線の取り回しなど，図上では考えにくい事柄もあるため，フレームに関しては Trial & Error を繰り返すのが良い．また，配置を考える際以下のようなポイントに注意されたい．

- 制御モデルに沿うように加速度・ジャイロセンサは機体の重心・回転中心に近づける．
- 試験段階において各部品が取り出しやすいようにする．（受信機をマジックテープで取り付ける等）
- ドリフトの発生を抑えるために各プロペラの中心と機体の重心を近づける．
- 接地時に機体が安定するように重心は低くする．

4.2. プロペラ

プロペラは翼型の断面を持つブレードが回転することにより空気の流れを作り出し，その反力として回転軸方向の推力を得るものである．回転の動力は後述するモータにより与えられる．

プロペラの特性を決めるパラメータの中で選定時に考慮する必要があるものは直径，ピッチ，ブレードの枚数である．このうち直径とピッチは [inch] の単位で表記される．直径はそのままプロペラの直径を表し，ピッチとはプロペラが一回転する間に進む距離（空気を押し出す距離）を表す．直径・ピッチと静止推力 T・必要動力 P との関係を示したのが式(4.1)，(4.2)である[4-2].

$$T = (d/10)^3 \times (p/10) \times (N/1000)^2 \times 22 \tag{4.1}$$

$$P = (d/10)^4 \times (p/10) \times (N/1000)^3 \times 0.45 \tag{4.2}$$

T: 静止推力 [g]　　d: 直径 [inch]　　p: ピッチ [inch]

N: 回転数 [rpm]　　P: 必要動力 [W]

ブレードの枚数，翼型，コード長などにより右辺末尾の係数が上下するため，TとPの実測値は変わってくるが，TとPの大まかな評価には有用な式である．例えば静止推力を一定としたときに直径dを 2 倍にすると，回転数Nは約 0.35 倍，必要動力Pは 0.7 倍となる．このようにプロペラ直径を大きくすると必要動力が小さくなり効率が高くなる．

また，マルチコプタは先述のように正回転，逆回転のプロペラを組み合わせて用いるため，CW（時計まわり），CCW（反時計まわり）のように回転方向を明記して販売されている．

［東大］直径：5 inch，ピッチ：3 inch　　［横国］直径：6 inch，ピッチ：3 inch

4.3. モータ

モータは電力を軸の回転運動に変換するものである．マルチコプタに使用できるモータはいくつかあるが，ここでは第 3 章で述べたようにブラシレスモータについて述べる．

モータの性能を決める数値として KV 値と定格電圧がある．KV 値は無負荷時の回転数をその際の電圧で除したものであり，[rpm/V] という単位で表現される．定格電圧はモータに与えることができる電圧である．通常のマルチコプタはバッテリに LiPo を用いるので，この値によって使用できる LiPo のセル数が決まる．

モータに与える電力は，ESC に与える信号を変化させることで変えることができ，それに従ってモータの回転数，プロペラの推力が変わる．また，電源に使用する LiPo のセル数によってもこれらの性能値は変わってくる．

このほかのモータのスペックとしてはモータのサイズ，軸の径，極の数，固定用のアタッチメントの有無がある．販売店（Hobbyking など）によってはこれらも詳細に示している

ので設計の際の参考にされたい.

4.4. プロペラガード

プロペラガードは文字通りプロペラの周りを覆うガードのことであり，主に 2 つの役割を持つ．まず一つは機体が墜落・衝突した時，プロペラや制御基板を衝撃から守る役割であり，もう一つは人間・機材を損傷させないという役割である．図 4.4 に市販のマルチコプタにおけるプロペラガードを示す．どのプロペラガードも軽量でかつプロペラの空力特性に大きな影響を与えないように設計されている．機体の調整や操縦練習においては，機体の墜落が頻発する．そのためプロペラガードは，墜落が起きても壊れないようにするか，壊れても素早く再生産が可能なようにすることで円滑な調整や練習ができるようになる．

また，プロペラガードを取り付けることにより機体の慣性モーメントと重量は大きく増加する．それにより機体の制御特性も変化するため，制御ゲインの再調整，モータのグレードアップなどの対応が必要となることがある．

a) プラスチックに穴をあけたもの

(b) ロッドを丸めて柵にしたもの

(c) プロペラを覆わない平面状のも

(d) 全体を覆う籠状のもの

図 4.4　市販のマルチコプタにおけるプロペラガードの例

4.5. 推力方向制御機構

推力方向制御機構は図 4.5 に示すような，サーボモータなどにより推力の方向を変える機構である．この機構はマルチコプタの中でプロペラの数が奇数のもの（トライコプタやペンタコプタ等）やプロペラの配置が正多角形ではないものに搭載され，機体に働く反トルクを 0 にするために用いられる．

図 4.5　トライコプタにおける推力方向制御機構の例

参考文献

[4-1]　全日本学生室内飛行ロボットコンテスト「製作ノウハウ」,
http://indoor-flight.com/experts，2015/11/15 アクセス.

[4-2]　Hikari,“マルチコプターを作ろう”，Amazon Services International, Inc.，2014.

[4-3]　電動 RC 専科！「トライコプター8 号の改 9/21」,
http://blogs.yahoo.co.jp/nagoya11758/39746515.html, 2015 年 11 月 15 日アクセス.

第 5 章　制御

この章ではマルチコプタを飛行させるための制御アルゴリズムについて解説する．つまり図 5.1 に示すように，実ハードとのインターフェースが完成していて，センサからの加速度や角速度といった情報，プロポからの指令値が分かっているときに，それらの情報からモータ出力を決定する制御アルゴリズムについて説明する．

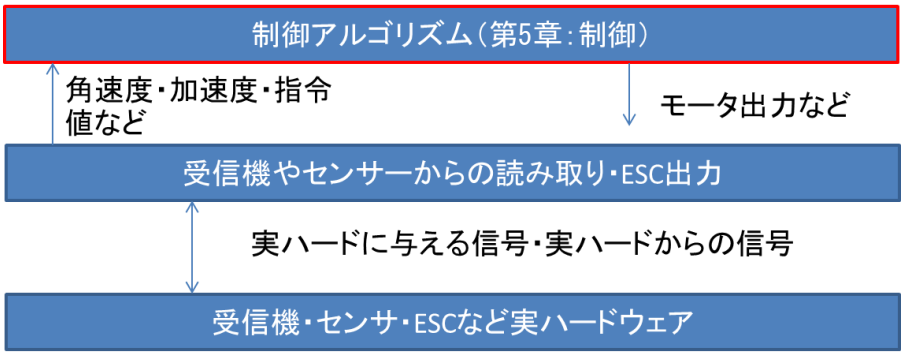


図 5.1　この章で扱う部分（赤枠部分）

5.1.　制御系の表記法

図 5.2 に制御アルゴリズムへの入力と出力をブロック線図表記で書いた図を示す．処理を長方形で書き，入出力を矢印で書く．ブロック線図表記は制御モデルを検討するときにも用いられる．また制御アルゴリズムを考える際，「状態」という考え方を使う場合もある．たとえば，マルチコプタの電源を投入した際，初めはモータが回転しない状態になっており，プロポに所定の操作を行うことで，モータが回転する状態に遷移するといったような処理を考える場合である．このような場合は状態遷移図というものを用いて検討する．図 5.3 にその例を示す．状態と入力，出力の組をステートマシンという．

本章でははじめに，プロポの指令に追従して機体を安定に飛行させるための制御系について説明する．

図 5.2　制御アルゴリズムへの入出力をブロック線図で書く場合

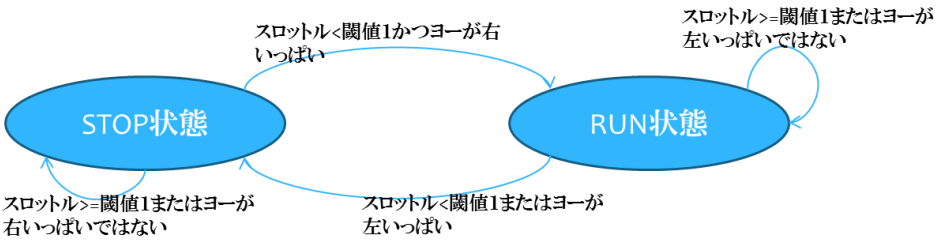


図 5.3　制御アルゴリズムのふるまいを状態遷移図で書く例

5.2.　機体の安定化・指令値への追従

機体を指令値（目標姿勢）に追従させるためには，センサの値から機体の姿勢を求め（姿勢推定），機体の姿勢から必要なモータ出力を求める（制御器）という大きく分けて 2 つの機能が必要となる．マルチコプタの飛行では，ロール・ピッチ軸回りの傾斜に応じて，ロータ推力の水平成分が生じて，水平方向の加速が生じるので，ロール・ピッチの傾斜を制御する必要がある．また機体の方向を制御するために，ヨー軸回りの回転角速度を制御する．

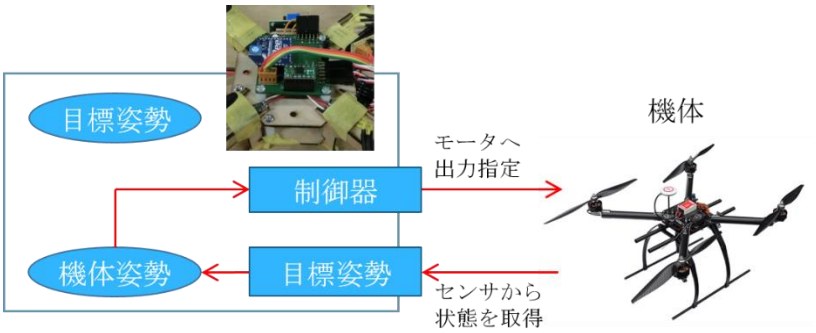


図 5.4　姿勢制御系の概要

5.2.1.　機体の姿勢の表現

機体の姿勢の表現方法には複数の方法がある．

- ロール・ピッチ・ヨー角で表現するやり方（オイラー角表現）
- 回転行列で表現するやり方
- クォータニオンを用いて表現するやり方

オイラー角表記は直感的に分かりやすいが，特異姿勢が生じる場合があるほか，微小な角度変化を積算していく場合に計算が複雑になりがちという欠点がある．回転行列は特異姿勢が生じないが，本来 3 つの数値で表せる回転を 3x3 行列で表記するために，回転行列の正規直交性を満たさなくなる場合がある．クォータニオンは回転を表現するやりかたで，特異姿勢がなく，行列よりメモリ使用量が少なく，微小な変化を積算する方法がわかりやすいという利点がある [5-1].

5.2.2. 状態推定

一般的なマルチコプタにおいて，センサから得られる機体姿勢に関する情報は加速度と角速度である．理想的には角速度を積算すれば姿勢が計算できるはずだが，実際は角速度センサ出力にはノイズやドリフトがあるので，積算し続けると時間の経過とともに誤差が蓄積してしまう．そこで複数のセンサ情報を組み合わせてより確からしい値を求める必要がある．複数のセンサを組み合わせるというアプローチは「センサフュージョン」と呼ばれており，これまで数多くのセンサフュージョンアルゴリズムが提案されている．下記にいくつかのアルゴリズムを紹介するので各自調べてみてほしい．

a) 相補フィルタ［横国］

角速度情報はドリフトなどの影響で長期的に見ると精度があてにならないが，短期的には精度が（それなりに）良い．一方加速度センサは，短期的に見ると機体の運動や振動などの影響を強く受けるが，長期的に見るとその観測値は鉛直方向を向いているはずである．そこで角速度センサの値にハイパスフィルタをかけ，加速度センサの値にローパスフィルタをかけて加算することでより信用のおける値を得るアルゴリズムが相補フィルタである．実際の計算では，例えば式(5.1)のように角速度から求めた姿勢角と加速度から求めた姿勢角，を一定の割合で加算することで姿勢角を推定する．

$$\theta_{est} = 0.8\,\theta_{gyro} + 0.2\,\theta_{acc} \tag{5.1}$$

θ_{est}: 推定姿勢角　　θ_{gyro}: 角速度を積分して求めた姿勢角

θ_{acc}: 加速度から求めた姿勢角

どのような割合で加算すると最も推定精度が高くなるかは，センサの種類や機体の振動等によって変わってくるため，試行錯誤が必要となってくる．相補フィルタは実装が簡単な反面，後述の Kalman フィルタ等と比べると推定精度がやや落ちる．

横国チームは相補フィルタを採用し，推定精度を保つために図 5.5 に示すようにセンサと本体の間に防振フォームを挟むことでフレームの振動によるノイズの軽減を図った．

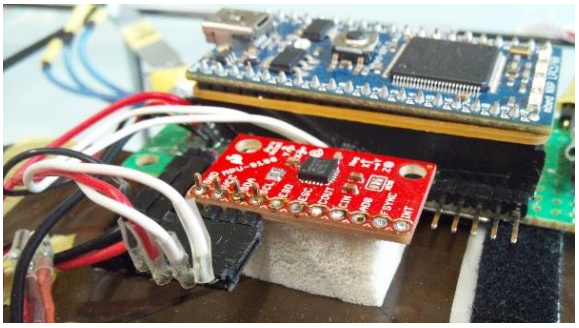


図 5.5　センサと防振フォーム（赤い基盤がセンサ，白いのがフォーム）

b) Kalman フィルタ

機体の姿勢を不確かさをもった状態量と考え，そのときに観測される加速度と実際に観測された加速度を比較することで姿勢を補正するアルゴリズム．姿勢の推定に広く用いられている．

c) Madgwick フィルタ［東大］

2010 年に Madgwick によって提案された姿勢推定アルゴリズム．まず角速度を積算して姿勢を求め，そのとき観測されるはずの加速度と実際に観測された加速度を比較して姿勢を求めるという点で Kalman フィルタと似ているが，Kalman フィルタより高速に計算することができる．Madgwick によれば Kalman フィルタと同等の精度が出ると主張されている [5-2].

東大チームは性能と速度が出るとされている Madgwick フィルタを用いた．

5.2.3. PID 制御

機体の姿勢が得られたら，機体の姿勢や角速度をもとにモータの出力を計算する必要がある，その出力を計算する方法（制御則）には様々な種類がある．特にマルチコプタは多入力多出力のシステムなので，各軸の干渉を厳密に考慮するためには，現代制御論に基づくモデルベースな考え方が必要となる．ただし機体のバランスがよく，ロール・ピッチ・ヨーの運動がほぼ独立しているとみすことができれば，ロール・ピッチ・ヨーを分離した単純な PID 制御でも飛行させることはできる．PID 制御とは何かというと，制御対象の量をx，目標値をx_{ref}，出力をyとすると以下の式で書かれる制御則である．

$$y = K_p(x_{ref} - x) + K_i\int (x_{ref} - x)dt + K_d\frac{d}{dt}(x_{ref} - x) \tag{5.1}$$

簡単な例として，図 5.6 に示すようなある質量mの台車を位置x_{ref}に追従させるために，モータで台車に力fを与える場合を考える．

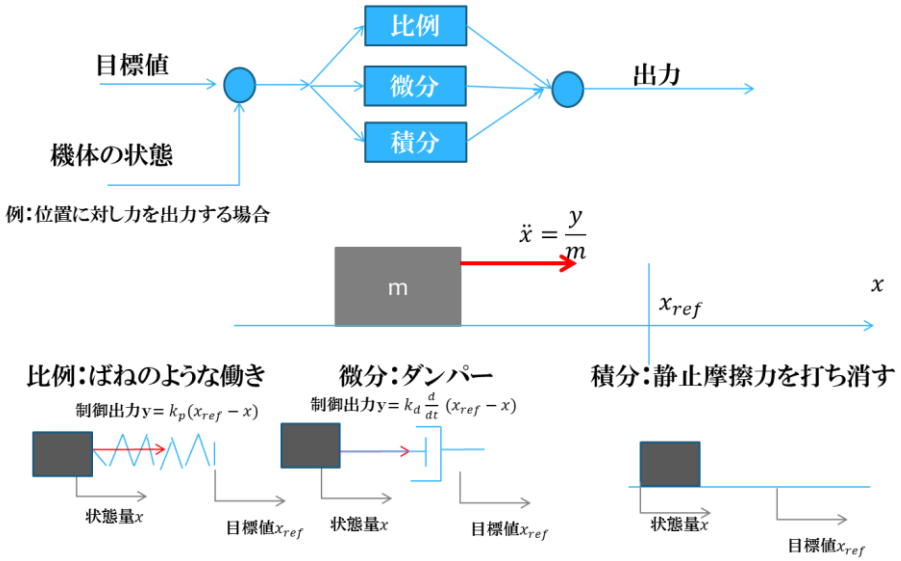


図 5.6　PID 制御の例

この例では比例制御はばねのような作用をし，微分制御はダンパ，積分制御は静止摩擦を消すような働きをする．制御対象によってどのような振る舞いをするかは変わってくるが，大まかな振る舞いはこの例のようになる．

マルチコプタでは応答の速い角速度を制御し，安定化された角速度に対し姿勢を制御するため，図 5.7 のような制御系が考えられる．

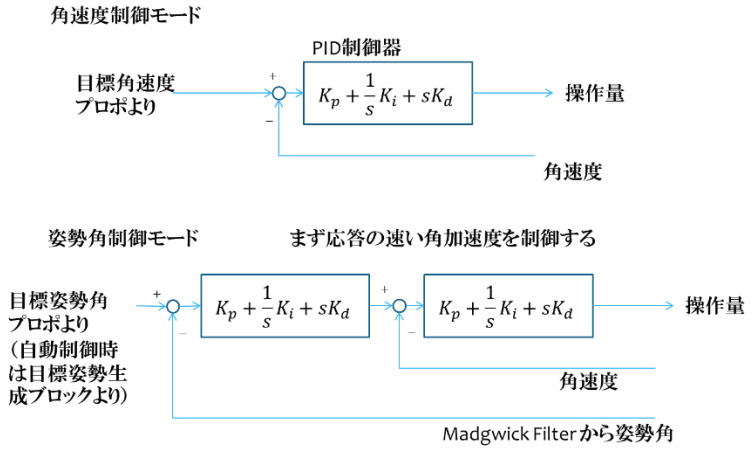


図 5.7　マルチコプタ制御の例

5.3.　PID 制御器の調整

制御対象の物理モデルが正確にわかっていれば計算によって制御パラメータを計算できるが，実際にはすべてのパラメータを知ることは難しい．そこで，試行錯誤的な方法でパラメータを決める方法について説明する．パラメータを決める際には，制御ループの内側の制

御器から調整する．

初めに P・I・D 各ゲインを 0 にする．この場合機体は不安定になる．次に P ゲインだけを少しずつ上げていくと，機体は安定するようになる．さらにゲインを上げると激しく振動し始めるようになる．そこで P ゲインを少し減らし，振動が起きる手前に P ゲインを設定する．次に I ゲインをくわえる．機体を傾けたとき復元力が発生する．増やしすぎると応答が遅くなるので，不安定にならない程度に加える．最後に D ゲインを応答が早くなりかつ不安定にならないようにくわえる．

外側の制御ループは PD 制御でよく，先ほどの手順に準ずる．

5.4. 制御ボードへの実装

上記のような制御則に実際に仕事をさせるために，プログラムで実装する方法を説明する．制御アルゴリズムは一定の制御周期で実行されるとする．つまり制御アルゴリズムの関数（ここでは仮に ControlAlgorithmExecute()とする）がΔt秒ごとに実行されるとする．

5.4.1. 入力→処理→出力型（PID 制御，姿勢推定）

微分を差分で，積分を積算で表す．

$$\frac{dx}{dt} \sim \frac{x_t - x_{t-1}}{\Delta t}, \int x dt \sim \Sigma x \Delta t \tag{5.2}$$

プログラムではそれぞれ下記のリストのように書ける．

```
static double prevx=0.0;  //微分のために$x_{t-1}$を保存しておく
double result_of_differencial=(x - prevx)/DELTA_T;
prevx=x;
static double sum=0.0;  //積分のために$\Sigma x \Delta t$を保存しておく
sum+=x*DELTA_T;
double result_of_integral=sum;
```

したがって PID 制御は下記のようになる．

```
static double prev_err=0.0;
static double sum_err=0.0;
double err=x_ref – x;
sum+=err*DELTA_T;
double output=KP*err + KI*sum + KD*(err – perv_err);
```

```
prev_err=err;
```

各パラメータを構造体で持っていてもよいかもしれない．ヘッダファイルで

```
typedef struct{
        double KP;
        double KI;
        double KD;
        double sum;
        double prev_err;
} PID_Info_t;
```

としておいて，ソースコードに

```
void PID_Init(PID_Info_t*pidparam,double kp,double ki,double kd){
        pidparam->KP=kp;
        pidparam->KI=ki;
        pidparam->KD=kd;
        pidparam->prev_err=0;
        pidparam->sum=0;
}

double PID_Exec(PID_Info_t*pidparam, double x,double x_ref){
        double err=x_ref – x;
        pidparam->sum+=err*DELTA_T;
        double result=pidparam->KP*err + pidparam->KI*pidparam-
>sum+
        pidparam->KD*(err-pidparam->prev_err);
        pidparam->prev_err=err;
        return result;
}
```

と記述しておいて，制御ループ内で

```
PID_Info_t rollpid;
PID_Info_t pitchpid;
```

```
void setup(void){   //初期化
略
        PID_Init(&rollpid,1,2,3);
        PID_Init(&pitchpid,1,2,3);
略
}
void ControlAlgorithmExecute (void){   //制御周期ごとに呼び出される
        略・姿勢推定等
        double roll_output=PID_Exec(&rollpid,roll,roll_ref);
        double pitch_output=PID_Exec(&pitchpid,pitch,pitch_ref);
        略・出力等
```

というようにプログラムすると，複数の制御器をうまく扱える．

状態推定も同様に各状態やパラメータを構造体にまとめ，処理を独立した関数に書くとすっきりしたプログラムを書くことができる．

5.4.2. ステートマシン

ステートマシンの更新についても ControlAlgorithmExecute()関数内で行う．最も簡単な実装は if 文や switch 文で場合分けするものである．

```
void ControlAlgorithmExecute (void){   //制御周期ごとに呼び出される
        static int StateNumber=0;
        if(state==0){
                if(condtion1){
                        StateNumber=Val1;
                }else if(condition2){
                        StateNumber=Val2;
                }
        }else if(state==Val1){
後略
```

この場合記述するのは楽だが，状態数が増えてくると若干煩雑になりがちである．次のように書くと各ステートの処理を別々の関数に分離することができ，全体の見通しがよくなり，ステートの追加も簡単になる．

```
#define MAX_STATE_NUM N
```

```
typedef int (*StateProcess)(int prevstate);  //各ステートの処理
//int を引数にとり int を返す関数の関数ポインタ
typedef struct{
        StateProcess[MAX_STATE_NUM];  //ステートの個数分の関数ポインタ
        int next_state;  //次に実行されるステート
        int state;  //今のステート
}StateMachine_Info_t;
```

```
void State_Add(StateMachine_Info_t*sm, StateProcess proc, int num){
        sm->StateProcess[num]=proc;  //ステートを登録する
}
void State_Reset(StateMachine_Info_t*sm){
        sm->state=0;
        sm->next_state=0;
}
void State_Update(StateMachine_Info_t*sm){
        sm->next_state=sm->StateProcess[sm->state](sm->state);
//ステートマシンの状態 sm->state での処理を実行する.
        sm->state=sm->next_state;
}
```

```
StateMachine_Info_t sm;

int State0(int prev){
        if(Condition0){
                return Val0;
        }
        if(Condition1){
                return Val1;
        }
}
int State1(int prev){
略
}
```

```
void setup(void){
        State_Add(&sm,State0,0);
        State_Add(&sm,State1, Val1);
        State_Reset(&sm);
}
void ControlAlgorithmExecute (void){   //制御周期ごとに呼び出される
        State_Update(&sm);
        後略
```

このように意味のあるデータのまとまりごとに構造体にまとめ，それに対する処理を独立した関数にまとめることで，すべての処理を loop()関数内にべた書きするよりも見通しがよく，拡張性の高いプログラムを書くことができるようになる．

参考文献

[5-1] 矢田部学，“クォータニオン計算便利ノート”，MSS 技報，Vol.18，pp.29—34，www.mss.co.jp/technology/report/pdf/18-07.pdf，2016/4/10 アクセス．

[5-2] Sebastian O.H. Madgwick, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays”, 2010. www.x-io.co.uk/res/doc/madgwick_internal_report.pdf, 2016/4/10 アクセス．

[5-3] 小林伸明，“基礎制御工学”，共立出版，1988.
古典制御工学の入門書．

[5-4] 藤倉俊幸，“組込みシステム開発に役立つ理論と手法”，CQ 出版，2012.
組込みシステのプログラム設計，アルゴリズム設計の考え方についての本．

第 6 章　ミッション

ここでは，飛行ロボットコンテストにおける各ミッションの概要，その中で難関ミッションである空撮と Rocking Wings 攻略のヒントを示す．

6.1.　ミッション内訳

マルチコプタ部門が新設された第 11 回大会においては下表のようなミッションがあり，満点は 2900 点であった．

表 6.1　マルチコプタ部門のミッションと加点対象

ミッション	加点対象	点数
	プロペラガード点	100
	離陸点	100
空撮	空撮点	500
被災地着陸・ 救援物資輸送	物資輸送点	500
	被災地着陸点	500
	被災地特定点	200
Rocking Wings	Rocking Wings 点	500
	着陸点	200
	時間点	-300～+300
	設置減点	-50×回数

それぞれのミッションの概要については参考［6-1］の動画を参考にされたい．

6.2.　空撮

第 11 回大会では，機体に搭載したカメラによって地上の 5 つのフード内にある数字を読み，5 つの数字の中央値のフードの前に着陸して物資を投下するというものであった．ここでは重量が 100g 以下の小型ワイヤレスカメラのうち，代表的なものを紹介する．

6.2.1.　代表的なカメラ（情報は 2015 年 11 月時点のもの）

a)　Ai-Ball［横国］

重量：22 g　（電池込み）

寸法：W30xH32xL35 mm

価格：1 万円程度

解像度：640x480 px

特徴：解像度，圧縮方式，明るさが変更可能．大会使用実績有り．

b) Sony アクションカムミニ HDR-AZ1

重量：63 g
寸法：W24.2xH36.0xL74.0 mm
価格：3 万円程度
解像度：1920x1080 px
特徴：上位機種あり．大会使用実績有り．

c) Raspberry-Pi A+ with Camera module［東大］

重量：35 g（一式）
寸法：W65xH56xL10 mm (Raspberry-Pi A の寸法)
価格：1 万円程度(一式)
解像度：2592x1944 px
特徴：Ai-Ball より設定の自由度高．大会側で開発キットの案内が有り．大会使用実績有り．

d) GoPro Hero4 Session

重量：72 g
寸法：W38xH38xL38 mm
価格：4 万円程度
解像度：640x480 px
特徴：大会使用実績無し．

e) レッツコーポレーション L-MC4KBK

重量：36 g
寸法：W38.5xH50.3xL20.4 mm
価格：2.5 万円程度
解像度：1920x1080 px
特徴：大会使用実績無し．

f) NX-DRW10H

重量：38 g
寸法：W48xH48xL17 mm
価格：1.5 万円程度
解像度：1280x720 px
特徴：大会使用実績無し．

6.2.2. 空撮における注意事項

- 市販のワイヤレスカメラに改造を施したものは電波法に抵触することとなり，大会では使用ができなくなる．
- 一般的に受信機の信号とカメラの信号は共に 2.4 GHz 帯を使用している．そのため混線が発生し転送速度が低下することがある．
  → 映像の通信速度が遅い場合，(i) カメラ側で映像の品質を落とす，(ii) 映像受信側に感度の高い無線アダプタを用いる，(iii) 受信機とカメラを離す，等の改善策がある．

6.3. Rocking Wings

Rocking Wings とは機体を左右にロールさせる動作であり，戦闘機の曲技飛行などでも見ることができる．気圧計（や距離センサ）などの高度センサの有無によって戦略が異なる．加点対象となる Rocking Wings のロール回数や左右への移動量は第 11 回大会の動画を参考にされたい [6-2].

6.3.1. 高度センサがある場合

「高速に左右を往復する制御」+「高度を一定にする制御」を組み合わせることにより Rocking Wings が実現できる．高速に左右を往復する制御を行うためには，受信機からの指令（3.2.3.3.参照）で決定していた機体の目標姿勢角θ_{YawTar}（5.2.参照）を図 6.1 に示すように強制的に入力することで実現できる．このときのθ_{YawTar}の値は，Rocking Wings で傾かせたい量を設定する．

※ 距離センサを用いた場合，Rocking Wings 中に機体が斜めになった時には高度の補正が必要となる．

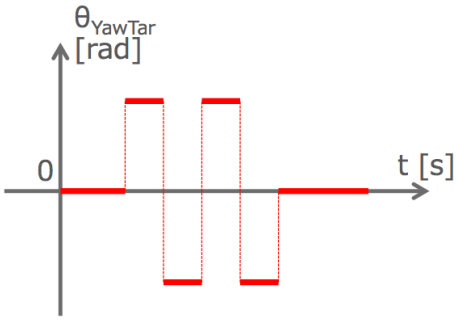


図 6.1　左右へ往復するための制御入力例

6.3.2. 高度センサがない場合

高度制御ができないので，あらかじめ高高度を保った状態から「高速に左右を往復する制御」をする必要がある．限られた条件下からしか Rocking Wings が開始できないため，条件を満たさなかった場合の墜落や，条件を満たすためのタイムロスが起こる可能性がある．

参考文献

[6-1] 第 11 回飛行ロボコン マルチコプタ部門ルール解説 by ROK,
https://www.youtube.com/watch?v=BSIICEiHfXI, 2016/04/10 アクセス.

[6-2] 第 11 回全日本学生室内飛行ロボットコンテスト 2 日目 決勝,
https://www.youtube.com/watch?v=cwuLUBZT0Es, 2016/04/10 アクセス.

第 7 章　その他

本章ではここまで述べてきた項目以外のマルチコプタを飛ばすためのコツや，気をつけるべきことについて述べる．

7.1. ゲインの決め方

マルチコプタの姿勢制御系には PID 制御が広く用いられている．姿勢制御はロール・ピッチ・ヨーの 3 軸まわりに対して行う必要があるが，ロール・ピッチまわりの制御とヨーまわりの制御は性質が異なる．

PID 制御のゲインは機体ごとに調整する必要があり，その値を決めるのは非常に難しい．理想的には機体のパラメータ（アーム長さ，ローター出力等）を全て計測し，計算やシミュレーションを行うことができれば根拠をもって決めることができるが，現実的にはパラメータの推定が難しい場合が多い．またそのような手順を踏まなくてもゲイン調整のみによって制御系を安定化させることができるのが，マルチコプタで広く用いられている PID 制御の利点でもあるので，ここでは PID 制御系を持つマルチコプタのゲイン調整法について述べる．

7.1.1. ロール・ピッチ制御系

ロール・ピッチ制御の目的は，姿勢角を一定値（ホバリング時は 0 deg）に保つことである．PID 制御のゲイン決定法は制御工学の分野で様々なものが開発されているが，本章では簡単にマルチコプタを飛行させることに重点をおいて，経験的な手法を紹介する．ロール・ピッチ制御の場合ある程度飛行するためには PD 制御を行う必要があるため，まずは PD ゲインのみを調整してから（I ゲインは 0 としておく），必要に応じて I ゲインを設定する．

7.1.1.1. 飛行前調整（オーダー合わせ）

最初から飛行可能なゲインを求めることは困難だが，ある程度オーダーを合わせておくことでおおまかなあたりをつけることができる．例えば全 ESC に PWM パルス 0.0015 s を入力したときにホバリングする機体の場合，(PWM のパルス幅は通常 0.001~0.002 [s])，想定している機体姿勢角が±30 deg であれば，P ゲインはだいたい (0.002−0.0015)/30 付近になると見積もることができる．同様に D ゲインも，センサを起動させて想定する周期で機体を持ち上げて動かしてみることでだいたいの角速度が分かるので，その値と PWM の範囲からおおよそのゲインを求めることができる．このような方法でゲインのオーダーを合わせておけば，多くの場合，安定化することはできなくてもある程度飛行することができる．

7.1.1.2. 飛行中の調整

7.1.1.1.の方法で求めたゲインで飛行させ，その挙動からゲインを調整することができる．以下に発生しやすい問題と，主な対処法を示す．

- 姿勢が保てない…D ゲインが低すぎる．
- ドリフトが発生する…P ゲインが低すぎる．または I 制御を入れる必要がある．
- 早い周期で振動する…D ゲインが高すぎる．

7.1.2. ヨー制御系

ヨー制御の目的は，姿勢角速度を一定に保つことである．（ロール・ピッチと異なり，一定の角度を目指すのではなく回らなければよい．）そのためヨー制御では基本的には D ゲインだけ設定すればよい．このような場合，基本的には小さなゲインから少しずつ大きなゲインに変えていくのが一般的なので，7.1.1.1.の方法と同様に，センサを起動させてある程度のあたりをつけてから小さめに設定，少しずつ大きくしていくという方法でゲインを調整するのがよい．

7.2. うまく飛ばないときは？

ここではうまく飛ばないときのよくある原因と，その対処法について解説する．各項の症状は，実際にコンテスト参加者達が機体製作時に陥ったものなので，同様の問題が発生した場合は参考にしてほしい．

7.2.1. プロペラが回らない

プロペラが回らないときは以下の問題が考えられる．

！これらのチェックを行うときは，安全のため必ずプロペラを一旦外す．

7.2.1.1. プロペラが回ろうとしているが，スロットルを上げても回らない場合

a) モータ軸がプロペラや構造に圧迫されている

モータからプロペラを外し，機体から外した状態で回転させてみる．

b) モータか ESC の故障

ESC の入力端子を直接受信機のスロットルチャンネルに接続し，チェックしたいモータ単体で回してみる．回らない場合は，モータ・ESC の組み合わせを変えてみて，モータと ESC のどちらが故障しているかをチェックする．

c) PWM の計算が間違っている

オシロスコープを持っている場合は，制御ボードからの PWM 信号をチェックしてみる．持っていない場合は，マイコンの PWM 関数に代入している値のログをとってチェックし

てみる．特に ESC は電源投入時の値がスロットルレバーの最低値付近（だいたいの受信機で 0.001 [s] 程度）でないと初期化されないことに注意．うまく初期化されているかどうかは，電源投入時の音で分かるようになっている．

7.2.1.2. プロペラが全く回らない場合

a) モータか ESC の故障

7.2.1.1.の b)と同じ．

b) 正しい PWM が生成できていない

7.2.1.1.の c)と同じ. ただしプロペラが全く動かない場合は, そもそも PWM 関数か, PWM 関数に代入する変数の更新が行われていない可能性がある．

7.2.2. スラストを上げると機体が転倒する

a) プロペラの上下の向きが間違っている

プロペラには推力を発生できる上下の向きがある．これを間違えると，そのプロペラだけ推力を発生できなくなり姿勢を保てない．横から見てやや膨らんでいる方を上側にする．

b) プロペラの回転方向が間違っている

マルチコプタの各モータは，時計まわり（CW）か反時計まわり（CCW）を適切に設定する必要がある．そのため各モータの回転方向に対応したプロペラを取り付けないと，うまく推力を発生することができない．

c) 姿勢制御則とモータ出力の対応が間違っている

姿勢が傾いたときに姿勢を戻すような制御，モータ配置になっているかを確認する必要がある．具体的にはログを取りながら機体を手で傾けてみて，センサの値，モータへの入力が意図した通りになっているかを確認する．

7.2.3. 機体がドリフトする

上手く離陸することができても，機体がその場にとどまらず，横方向に流れてしまう（ドリフトする）ことがある．

a) センサが傾いている

加速度による姿勢角推定を使っているとき，センサが地面に対して傾いていると推定される角度も傾いてしまう．これを解消するために，電源投入時にセンサキャリブレーションを行う必要がある．加速度センサの場合，電源投入時にまず加速度センサによる姿勢角を計算し，以降その値を 0 deg とする．具体的には，電源投入時に加速度センサが機体姿勢を 3

deg と計測したら，地面が傾いていない限りその傾きは機体やセンサの取り付けによる誤差なので，飛行中に加速度センサが計測する姿勢角を常に−3 deg することで誤差を解消する．このような処理をキャリブレーションと呼ぶが，ジャイロセンサにも同様の処理がよく用いられる．

b)　重心位置がずれている

機体重心位置の偏りはある程度であれば制御で吸収することができるが，偏りが制御性能を超えてしまうとドリフトが発生する．パーツ配置を変えるか，おもりを使って重心を調整する必要がある．

c)　ゲイン調整が上手くいっていない

7.1.1.2.参照.

d)　あて舵

機体の調整を十分に行っても多少のドリフトが発生することがある．その場合はプロポのトリム調整機能を使って舵のセンターをずらすことでドリフトを吸収する．プロポのトリム設定方法は商品によって異なるが，概ね図 7.1 の位置にあるトリムレバーで調整することができる．

図 7.1　トリムレバー

7.2.4. ヨーが回ってしまう

前述のドリフトは，機体がロール・ピッチまわりに傾いてしまうために発生する．同様に，ヨーまわりの姿勢制御が上手くいっていないと，離陸後に機体がヨーまわりに回ってしまうことがある．

a)　モータの回転方向

多くのマルチコプタは，モータの回転方向を互い違いにすることでプロペラの反トルク

を打ち消し，ヨーまわりを安定させている．そのため時計回り（CW）か反時計回り（CCW）の配置が間違っているとヨーが回ってしまうので，まずこれを確認する．（図 1.3）

b) ヨーゲインが小さすぎる

ヨーまわりの D ゲインが小さすぎる可能性がある．（ヨーまわりの角速度を十分に打ち消せていない．）ただし，ゲインをあげ過ぎると振動する可能性があるので注意する．

c) モータが傾いている

モータの軸が真上を向いていないと，プロペラの推力によりヨー方向にモーメントが発生してしまう．この推力によるモーメントはプロペラの反トルクに比べて大きいことがあるので，モータは真っすぐ取り付ける．

d) プロペラの劣化

墜落などにより先端が欠けたプロペラは発生する反トルクが減少する．そのため欠けたプロペラを使用していると，ヨー周りに回転してしまうことがある．

e) あて舵

ロール・ピッチのドリフト同様，ある程度はあて舵で対応することができる．

7.3. LiPo バッテリの取り扱いについて

リチウムポリマーバッテリ（通称 LiPo）は重量に対して蓄えることができるエネルギが高いため，近年ラジコン用のバッテリとして広く使われている．しかし，従来のバッテリに比べて発火・爆発の危険性が高く取り扱いには注意しなければならない．

7.3.1. やってはいけないこと

LiPo に関してやってはいけないことを以下に示す．もしやってしまった場合は，LiPo が発熱しないか，膨らみ続けていないか，変なにおいがしないか等をチェックする．そのような症状が見られた場合，その LiPo の使用を中止し，LiPo セーフに入れる等発火しても安全な状態にして様子を見る必要がある．

- 短絡：バッテリの+と−をショートさせる．
- 過充電：LiPo が満充電状態のときにさらに電圧をかける．
- 過放電：LiPo が放電状態のときにさらに電力を取り出そうとする．
- 強い衝撃：落下，鋭利なもので外皮を傷つける等，強い衝撃を与える．

7.3.2. 初使用時の慣らし

購入直後の LiPo は内部のリチウムが不活性なことがある．このような状態でいきなり使

用すると，リチウムが劣化しやすく LiPo の寿命を縮める恐れがある．それを回避するために購入直後の LiPo は慣らし運転を行うことが推奨されている．

具体的には以下の手順で充放電を行う．放電には充電器の放電モードを用いる．1/4C 放電→1/4C 充電→1/2C 放電→1/2C 充電→1C 放電→1C 充電．（基本的に購入時のバッテリはほぼ満充電状態なので放電からスタートしているが，必ず電圧をチェックして放電していれば充電から始める．）ここで C とは放電容量を意味しており，LiPo の放電性能を表している．1 C とはそのバッテリの容量と同じ値のアンペア数のことを指し，例えば 1000 mAh, 2 C といえば 2000 mA のことを指す．LiPo に記載されている値は放電容量を意味しているので，1000 mAh, 35 C と書いてあれば，その LiPo は最大で 35000 mA 流すことができるということを意味する．（その値を超えて使用すると発火の恐れがある．）基本的に LiPo バッテリ充電器は 1 C で充放電するように設定されるので，この値を調整することで 1/4 充放電等を行う．

7.3.3. 保管の方法

LiPo を使う場合, 使用時以外に保管にも注意する必要がある. 保管の仕方が悪いと, LiPo の劣化に伴う性能悪化や，最悪の場合発火・爆発する可能性もある．

7.3.3.1. LiPo セーフ

前述したように，LiPo は発火・爆発する危険性があるため，保管時には以下のようなセーフティバッグを使用することを勧める．このバッグは難燃性の素材でできており，かつ衝撃等にも強いため，仮に LiPo が爆発しても被害を最小限に抑えることができる．

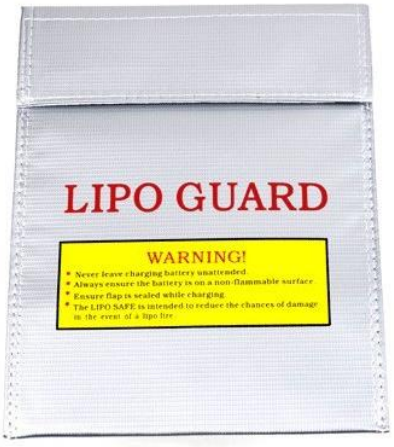


図 7.2　LiPo Guard リポバッテリーセーフティーバッグ

7.3.3.2. ストレージモード

LiPo を長期間保管する場合, 満充電や放電状態で保管すると LiPo の劣化の原因となる．

最近の充電器にはストレージモードという，LiPo を保管に適した電圧まで充放電してくれるモードがあり，長期保管する場合はこのモードで充放電するとよい．またストレージモードが無い場合でも，60%程度の充電で保管すると同様の効果が得られる．

7.4. 通販サイト

マルチコプタに関する部品を購入できる通販サイトをいくつか紹介する．最近はマルチコプタに特化したラジコンショップ等も増えてきているが，マルチコプタを自作するためにはモータやフレーム部材等を別々に購入することができる必要があるため，よりラジコンや模型に特化したサイトを利用するとよい場合がある．

- Hobbyking，http://www.hobbyking.com/hobbyking/store/index.asp
  海外サイトだが，ラジコン全般に関する商品が豊富．
- Cosmotech，http://www.ep-cosmotech.com/
  小型ラジコン向けの電装パーツが豊富．
- R/C ネットショップロビン，http://robin.jp/SHOP/16711/list.html
  ラジコン向け商品が豊富．特にカーボンロッド等，カーボン部材に強い．
- RC イーテック，http://www.rc-e-tech.co.jp/eccube/html/
  ここもラジコン向けカーボン部材が豊富．

7.5. 参考資料

マルチコプタを製作するために役立つ資料を紹介する．最近，マルチコプタや電子工作にに関する分かりやすく，手に入りやすい資料が続々発行されているので，適宜新しい資料を集めながら勉強することを勧める．

- 特集　飛行実験ずみ！最新モータ＆電池，トランジスタ技術，2015 年 12 月号．
  マルチコプタに関する特集が組まれている．
- 福田和宏，これ 1 冊でできる！Arduino ではじめる電子工作 超入門，ソーテック社，2014.
  第 3 章で取り上げた Arduino に関する入門書．電子工作の基礎を学ぶことができる．
- 勝純一，“超お手軽マイコン mbed 入門”，CQ 出版，2011.
  こちらも第 3 章で取り上げた mbed の入門書．サンプルプログラムが豊富．