

卒業研究

論文名 安定飛行ができるドローンの開発

指導教員名 伊藤 恒平教授

提出年月日 平成30年2月2日

提出者 機械工学科

氏名 天王地 亮太

金沢工業高等専門学校

目次

第 1 章 序 論	2
1.1 研究の背景	2
1.2 研究の目的	3
1.3 論文の構成	3
第 2 章 フレームの設計製作	4
2.1 ドローンの飛行原理	4
2.2 設計	6
2.3 製作	11
第 3 章 回路設計製作	12
3.1 ドローンの制御システム	12
3.2 使用する電子部品	12
3.3 使用するマイコンの機能	18
3.4 ジャイロセンサ	19
第 4 章 ジャイロセンサとマイコン間の通信	20
4.1 通信方法	20
第 5 章 角速度及び角度の取得について	31
5.1 MTU 機能を利用したマルチコプタの方向転換方法	31
5.2 角度推定方法	31
5.3 フィードバック制御の流れ	33

第 6 章 結 論	35
6.1 本 研 究 の ま と め	35
6.2 今 後 の 課 題	35
参 考 文 献	36
謝 辞	37
付 錄 A プ ロ グ ラ ム	38

第1章

序論

1.1 研究の背景

現在、図 1.1 のようなドローンは物資輸送、それぞれが自作した物でスポーツ・レースを行ったり、空から農薬を撒くなどの農業、果ては火山の火口や災害などで人が立ち入れない所の調査など、幅広く利用されている。

本年は、自作したドローンで飛行ロボットコンテストという室内飛行機の大会に出場しようと試みた。飛行ロボットコンテストとは、2006 年から開催された日本航空宇宙学会が主催の室内飛行用航空機型ロボットによる競技。自作ドローンが参加可能な、マルチコプタ部門は 2015 年の第 11 回から登場した。



図 1.1 ドローン

1.2 研究の目的

ドローンの製作に必要なプログラムと制御に関する知識がなく、結果大会のエントリーの締切までに作り上げることができず、参加すること事態できなかった。そこで製作に関わる制御の知識をドローンを製作しながら学んでいき、ドローンが外部からの突然加わる力によって傾き墜落しないような安定飛行ができる機体の製作を目標とする。

1.3 論文の構成

この論文では以下のように構成されている。1章では、我々の研究目的やドローンの原理について説明したものである。

2章では、ドローンのフレームを設計製作に関するものである。

3章では、使用する電子部品、センサー、マイコンの機能を説明したものである。

4章では、センサーとマイコン間の通信方法を記す。

5章では、ドローンの全体の制御の流れや、センサーからのデータを処理し本体の制御に反映する方法を記す。

6章では、前章の方法で得た結果を記す。

7章では、本研究において大きな力となった関係者各員に謝辞を記す。

第2章

フレームの設計製作

2.1 ドローンの飛行原理

ドローンは4～8枚ほどのプロペラが同時に回転して飛びますが、図2.1のように右上のプロペラは右回転、左上のプロペラは左回転と、隣り合うプロペラとは逆回転をしている。これにより、本体が回転しようとするチカラを相殺して、飛行が可能になっている。

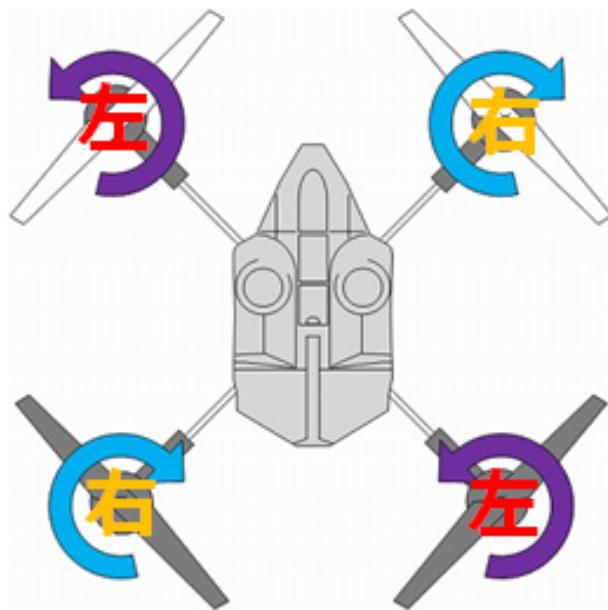


図2.1 ドローンのプロペラ回転方向

前進をする場合は、図2.2のように進みたい方向のプロペラの回転数を遅くし、その反対のプロペラの回転数を速くすることで、前進を行う。又、後退ならばその反対のことをすることで後退する。

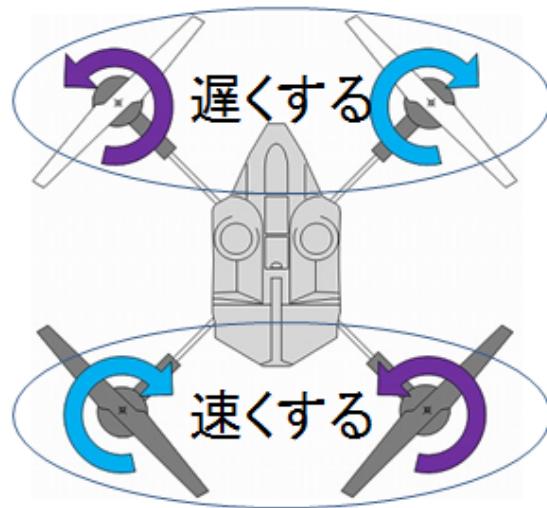


図 2.2 ドローンを前に進ませるとき

右に移動する場合は、図 2.3 のように右側のプロペラの回転数が遅く、左側のプロペラの回転数が速くします。又、左に移動する場合もこの方法と反対のことを行なえばよい。

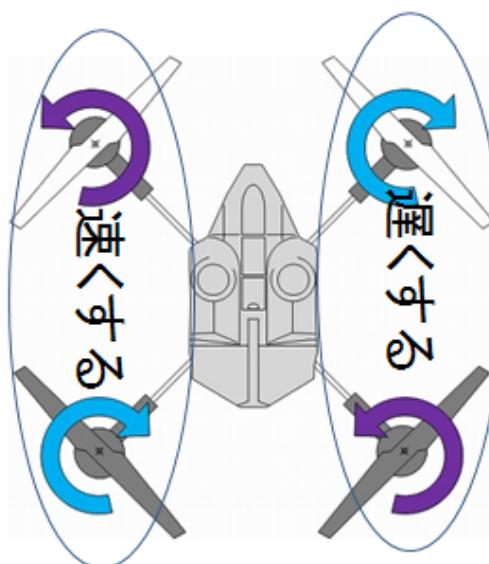


図 2.3 ドローンを右へ移動するとき

左回転をする場合は、図 2.4 のように右前のプロペラと左後ろのプロペラの回転数が速く、左前のプロペラと右後ろのプロペラの回転数が遅くする。又、右に移動する場合もこの方法と反対のことを行なえばよい。

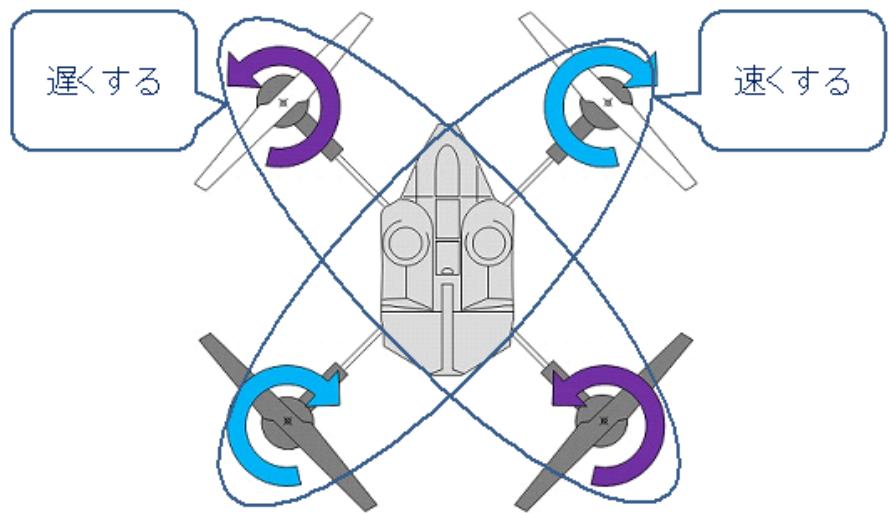


図 2.4 ドローンを左回転させるとき

2.2 設計

始めにインターネット等で図 2.5 のような市販のドローンを調べた。



図 2.5 市販品例

それをもとにして形だけを作り出したものが図 2.6 のフレームとなり、

ここから軽量化をコンセプトとし電子部品を含め総重量300g以下を目指し改良を続け製作を行ってきた。なお、フレームの材料は厚さ2mmのベニヤ板を使用している。

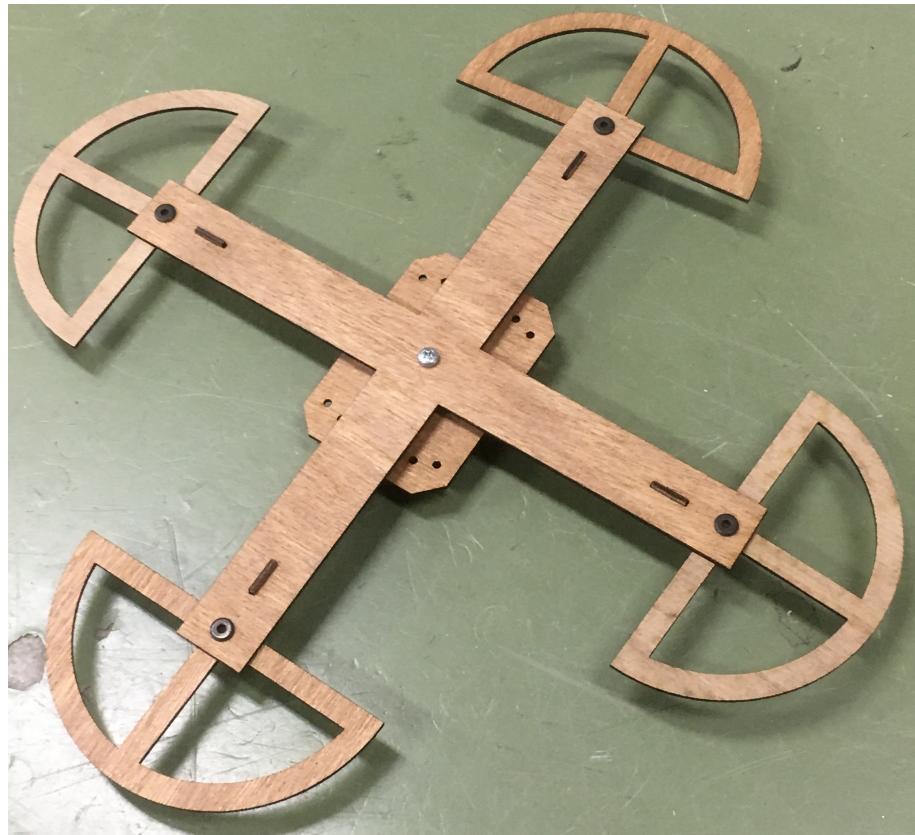


図2.6 フレーム初期案

この初期案はとにかく不格好で、重量もフレームだけで80g以上もあった。ここでの改良は、腕や土台、プロペラガードを一つのパーツとし、各パーツの接合をねじ等にするとその分重量が増量してしまうので各パーツの接合部をはめ合いに、接着剤で接着可能なものとした。さらにフレームの所々にトラス構造のような穴をあけ、軽量化とともに剛性も上げた。

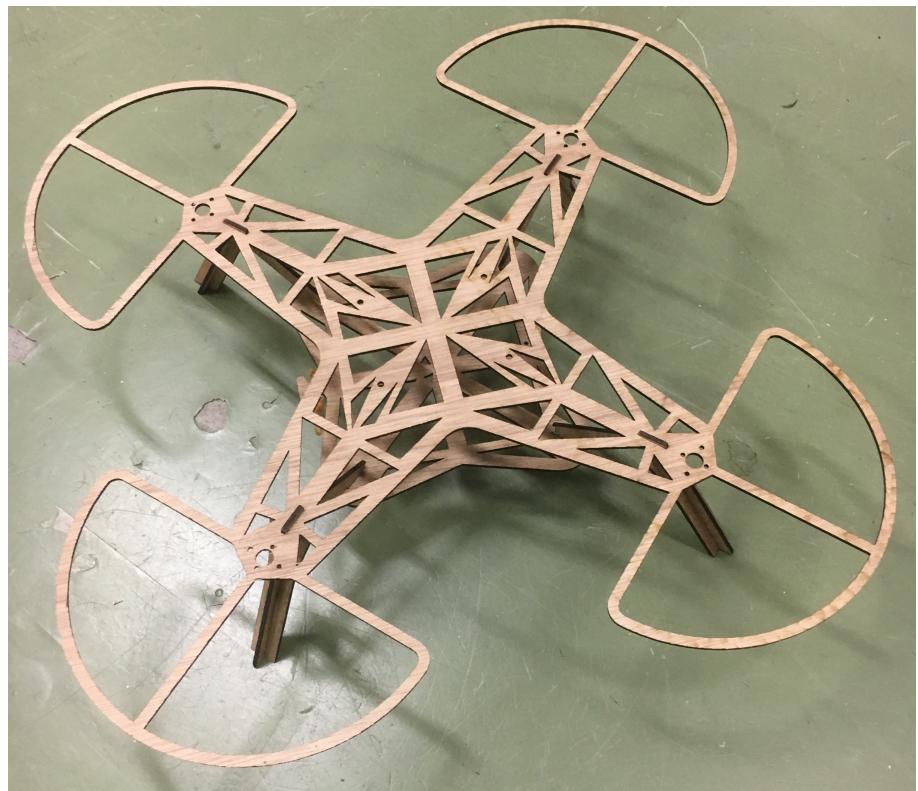


図 2.7 改良その 1

ここからトラス構造の剛性を失わずにどれだけ開けた穴を広げられるか、他の開け方ではどれだけ軽量化を図れるかを実際に製作し重量を測定した。なお、この時点の重量は 70g 前後。



図 2.8 改良その 2

ここでは主に形だけでは無く、使用するベニヤ板の密度が小さいものを使用した。その結果約 40g ほどまでに軽量化した。

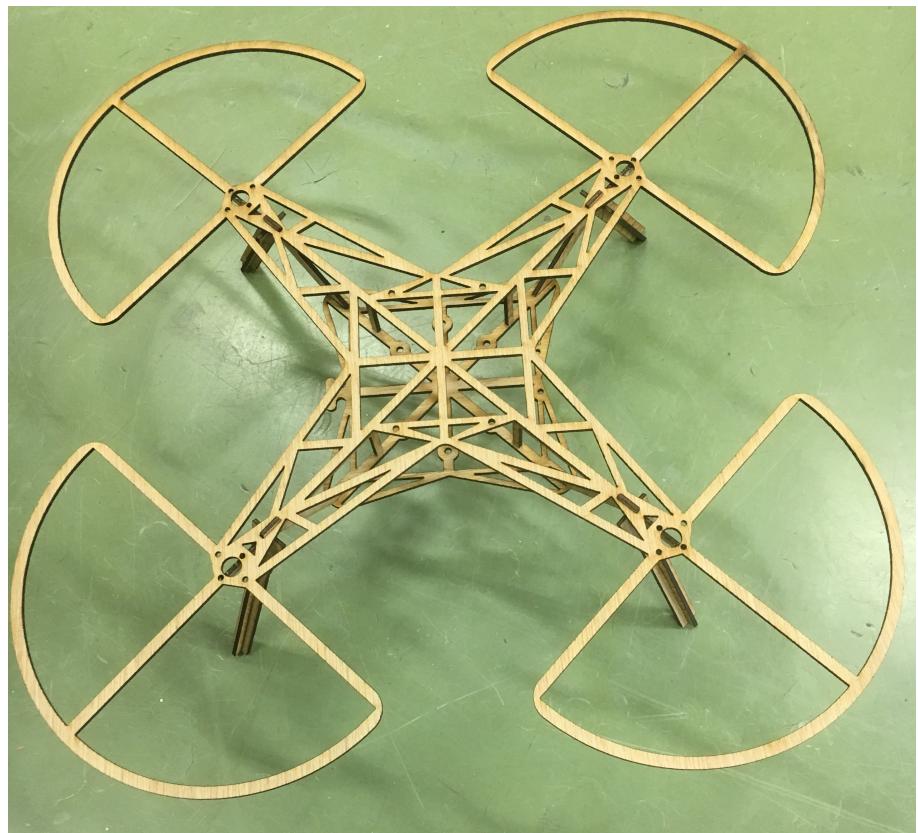


図 2.9 改良その 3

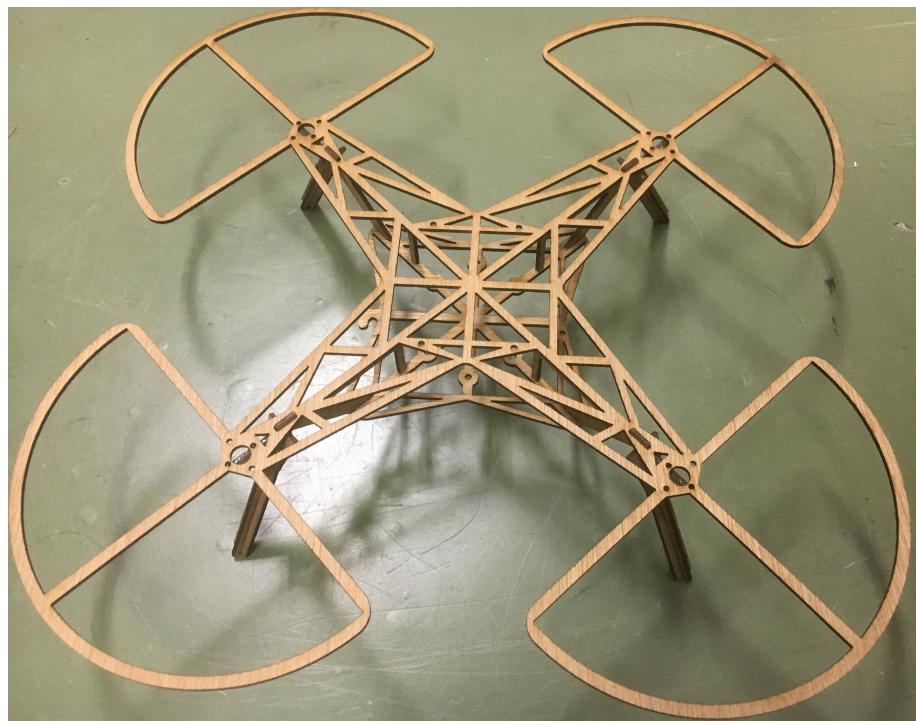


図 2.10 完成したベニヤフレーム

2.3 製作

このフレーム形を手作業で製作するにはかなりの根気や技術、時間を必要とする。この製作を図2.11のレーザー加工機を使用することによってこの形を綺麗に作るための技術や時間を短縮することに成功した。



図 2.11 レーザー加工機

第3章

回路設計製作

3.1 ドローンの制御システム

今回自作する制御システムのブロック線図を図3.1に記す。

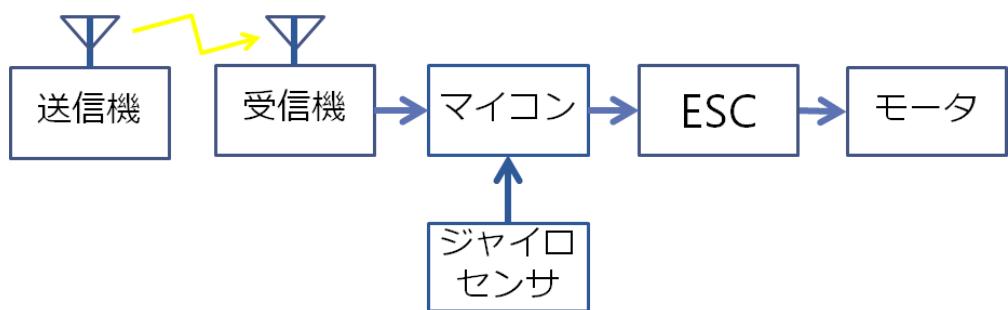


図3.1 制御システムブロック線図

この流れは送信機からの指示を受信機で受け取りマイコン内でセンサーからのデータとともに処理することで出力するデータを出す。その後,ESCに送りその信号でモーターを回転させる。このESCとはマイコンからの信号によりモーターの回転速度を制御するモータードライバーの一機能である。

3.2 使用する電子部品

今回の製作に必要な電子部品は以下のようとなる。

- 1) マイコン
- 2) ジャイロ

- 3) ESC
- 4) モータ
- 5) 受信機
- 6) 送信機
- 7) バッテリー
- 8) 電源関係

このドローンの頭脳ともいえるセンサーや図3.3の受信機からのデータを処理しESCにモーターの回転数を指示する信号を送ったりするマイコンは図にある秋月電子のRX621で、センサーは図3.2の秋月電子のRX621を使用する。

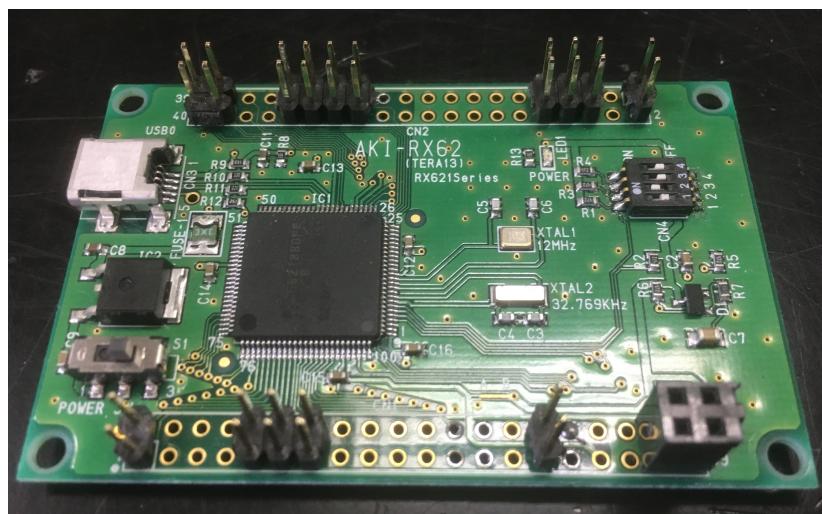


図3.2 マイコン



図 3.3 受信機

今回 の 研究 の 要 と な る センサー は 図 3.4 の GY-80 と い う ジャイロ、 加 速 度、 地 磁 気 と い っ た 複 数 の センサー が 一 つ と な っ た も の で、 そ の 中 の ジャイロ センサ だ け を 使用 す る。

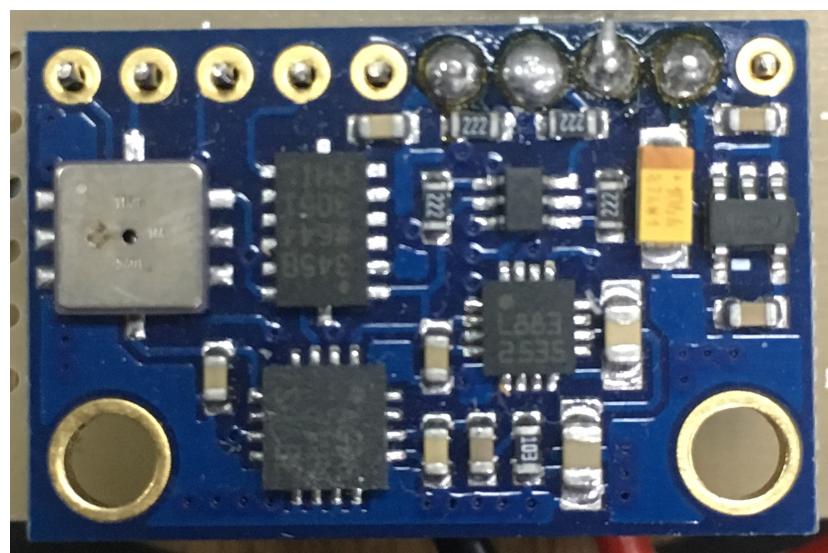


図 3.4 ジャイロセンサー

ド ロ ー ン を 飛 ば す た め の モ ー タ ー は 図 3.5、 マ イ コ ン か ら の 指 示 を モ ー タ ー へ と 繋 ぐ た め の ESC は 図 3.6 を 使用 す る。



図 3.5 モーター

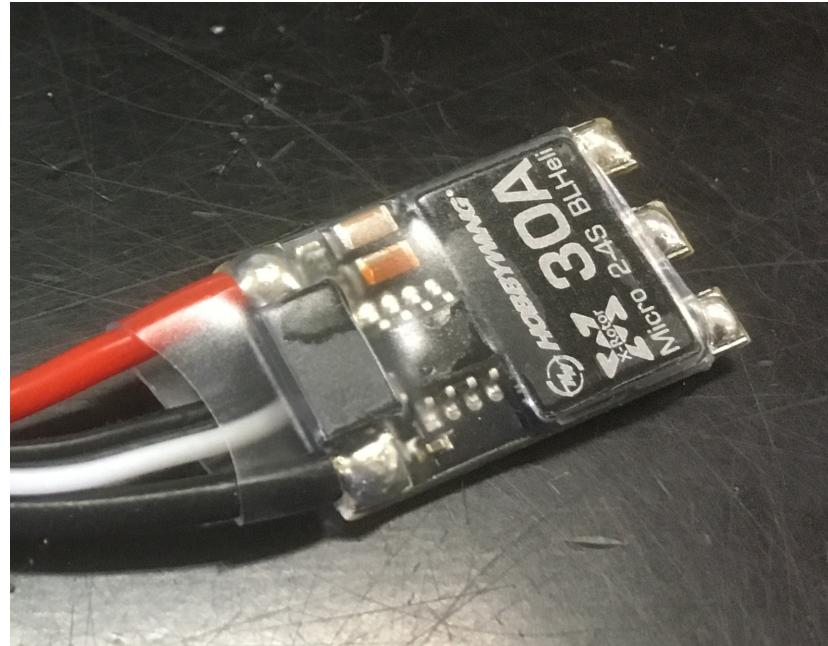


図 3.6 ESC

図 3.7 のバッテリーはドローンの電源で 7.4 V の電圧を持っている。しかしマイコンとセンサーは 3.3 V、受信機は 5 V の電圧を必要とし

て い る た め 図 3.8 の レ ギ ュ レ ー タ で 3 . 3 V, 図 3.9 の DCDC コンバ ー タ で 5 V に 電 壓 を 変 換 し て 使用 す る.



図 3.7 バッテリー

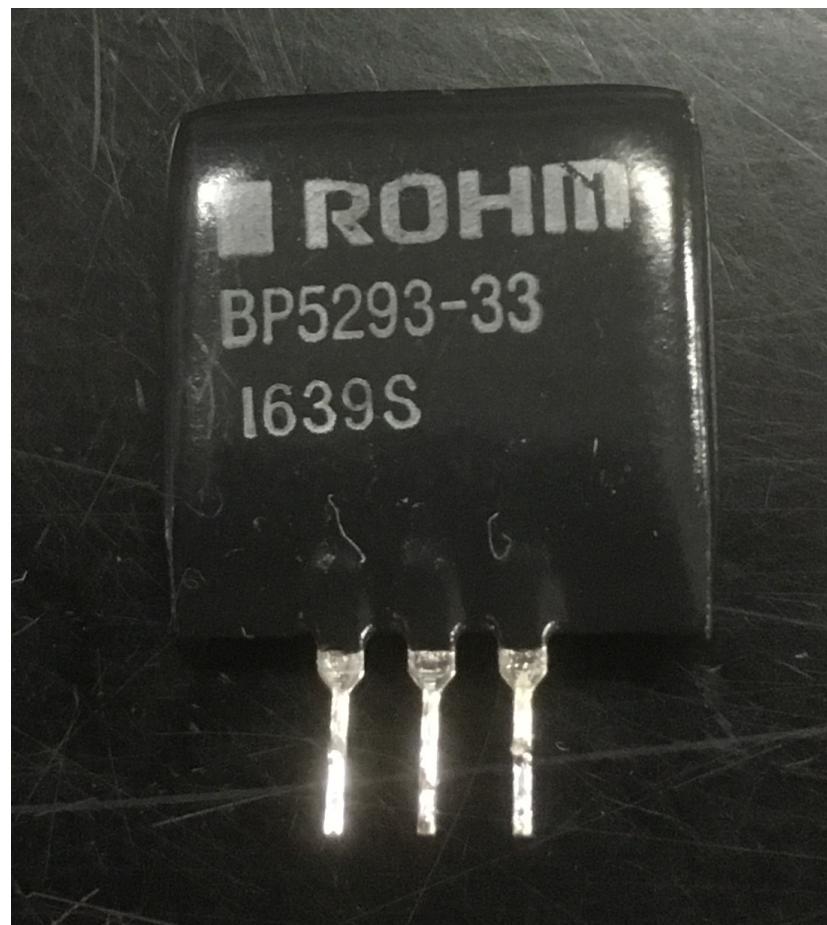


図 3.8 レギュレータ



図 3.9 DCDC コンバータ

以上 の 部 品 を 使用 し 製 作 し た 基 板 が 図 3.10 と な る。

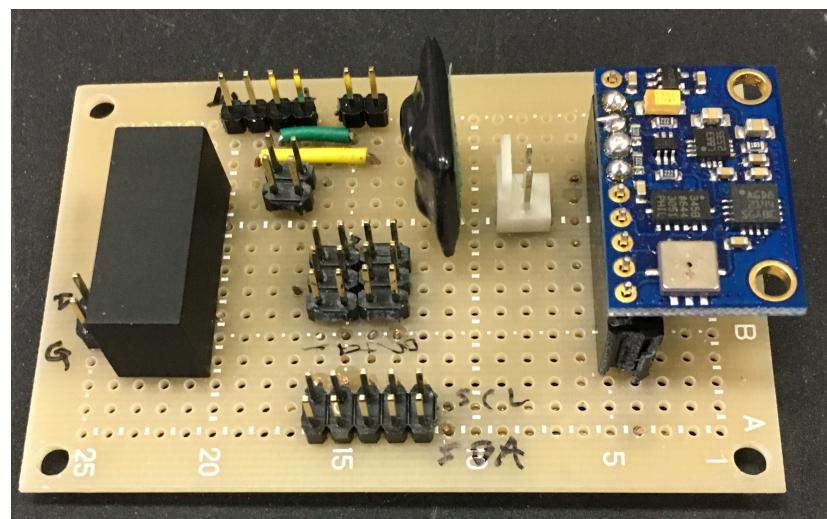


図 3.10 電子回路

フレームに使用電子部品を配置した自作ドローンの完成図が以下の図 3.11 と な る。



図 3.11 自作ドローン完成図

3.3 使用するマイコンの機能

今回使用したマイコンに含まれている機能は以下のようになっている。

3.3.1 マルチファンクションタイマパルスユニット 2 (MTU2)

受信機から送られる信号をこの機能で処理することによって ESCへと信号を送り、モーターを制御する。

この機能を使用するためには、使用する送信機のパルス幅をオシロスコープ等で測り、プログラムで書き込む必要があり、更に割り込みというある時間や条件がそろうと今読み込んでいるプログラムの読み込みを中断し、別のプログラムを読み込むといった動作がありそのプログラムとのかみ合いをそろえたりするのに苦労した。だが、設定はややこしいもののプログラムの設定さえ間違っていなければ問題なくこなせるものもある。

3.3.2 I2C バスインタフェース (RIIC)

この機能はマイコンがI2C通信をするための機能で、マイコン側のI2C通信の設定、ジャイロセンサとデータのやり取りをするために使用される。

3.4 ジャイロセンサ

主に市販で売られているドローンや自作で製作するときに使われているマイコンはライトコントローラーと呼ばれるジャイロセンサ、加速度センサ、気圧センサ、GPSといった機体の姿勢を制御し安定させるセンサーを含んだコンピューターを使用しドローンを制御するのだが、今回はマイコンとセンサーを別々にし、シリアル通信を利用してセンサーから来るデータのやり取りを行った。

第4章

ジャイロセンサとマイコン間の通信

4.1 通信方法

ジャイロセンサとマイコンとのデータのやり取りはI2C通信を使用しデータ通信を行う。

4.1.1 I2C通信とは

I2C(Inter-Integrated Circuit)とは、フィリップス社が開発した複数のデバイスとシリアル通信を行う通信方法の1つである。この通信方法の大きな特徴はデータ通信のすべての権限を持つデバイス(マスタ)とそれ以外のデバイス(スレーブ)間で2本の信号線SDA(シリアルデータ)とSCL(シリアルクロック)でマスタとスレーブ間のデータ通信を行う。ここでのマスタ側のデバイスはマイコンで、スレーブ側をジャイロセンサとする。この関係図を図4.1に示す。

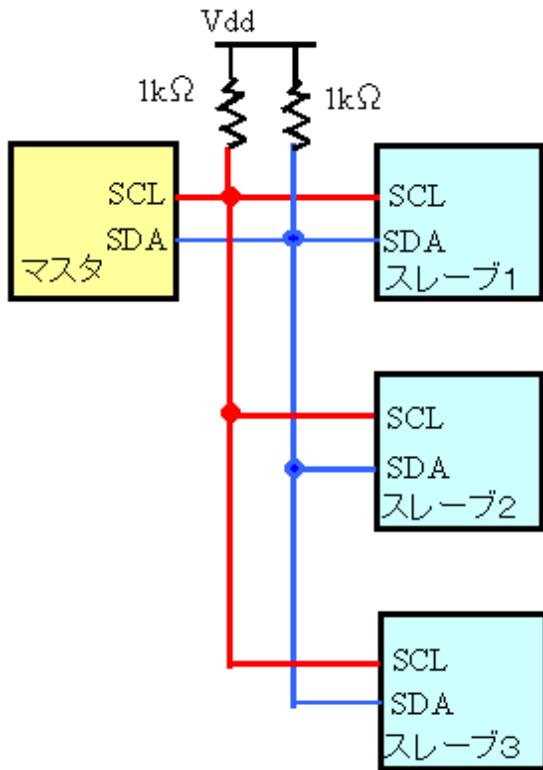


図 4.1 I2C 通信

SCL と SDA

図 4.1 にある SCL はスレーブとの同期をとるための信号線で、通常はマスターからスレーブへの一方通行の線である。一方、SDA は SCL に同期し、データの転送に用いる信号線であり、マスターとスレーブ、どちらからも送信される。

書き込みの流れ

マスター側からスタートコンディションを発行し通信を開始したら、マスター側から”通信するスレーブ”と”マスター側が書き込み”アドレスを8bit 分送信しスレーブ側から正常に送信されたことを意味するACKという1bit 分のデータが返される。この次からスレーブに書き込みたいデータを約1byte分送りACKがスレーブから送られる。そしてすべてのデータの送信が終了したら、ストップコンディションを発行し通信を終える。

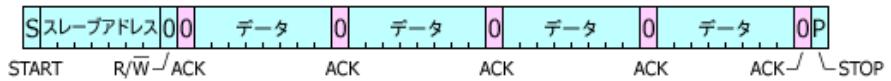


図 4.2 I2C 書込み

読み込みの流れ

マスタ側からスタートコンディションを発行し通信を開始したら、マスタ側から”通信するスレーブ”と”マスタ側が読込む”アドレスを8bit分送信しスレーブ側からACKが返される。そしてスレーブから送られるデータを約1byte分もらい,ACKをスレーブに送る。そしてすべてのデータの送信が終了したら、ストップコンディションを発行し通信を終える。



図 4.3 I2C 読込み

4.1.2 通信方法

通信プログラムの流れはジャイロセンサーのマニュアルの方法で図4.4のようなフローチャートとなる。マニュアルより、図4.7の流れは図4.4の所の’ステータスレジスタのアドレスを送る’の所から’データをもらう’までの部分となる。

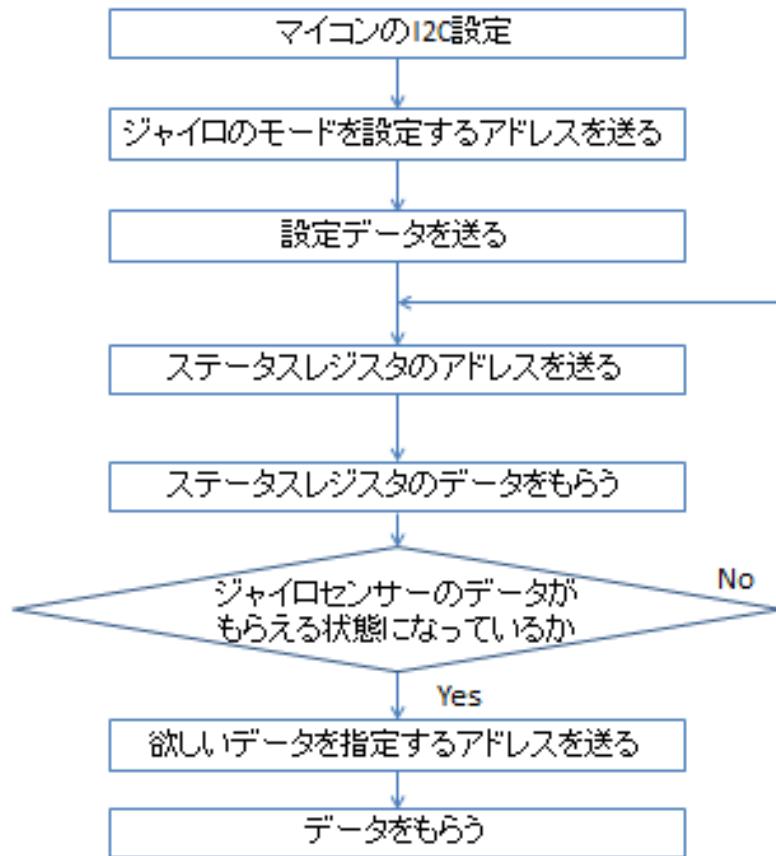


図 4.4 通信フローチャート

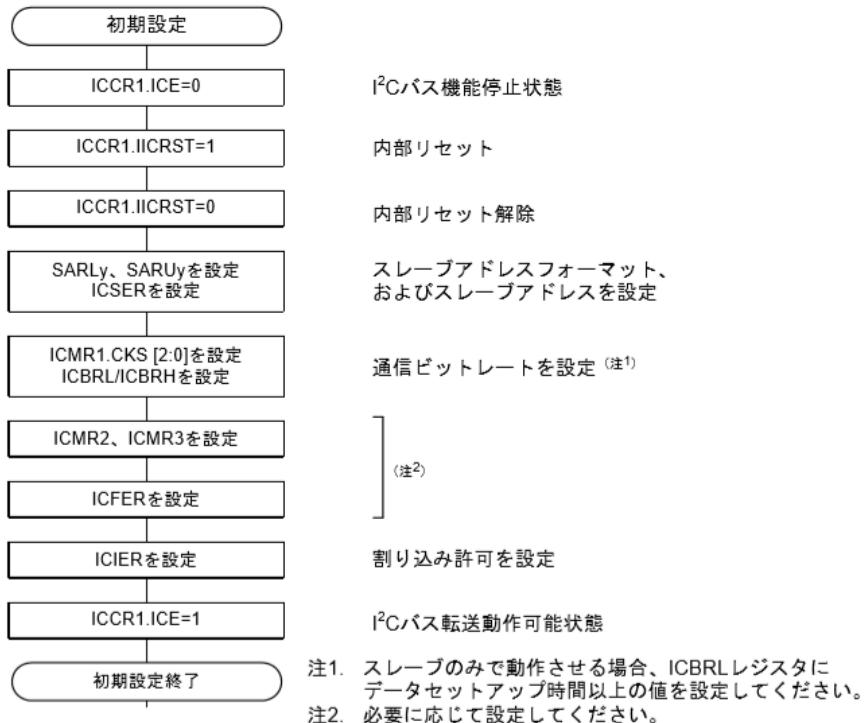


図 4.5 マイコンの I2C 通信設定手順

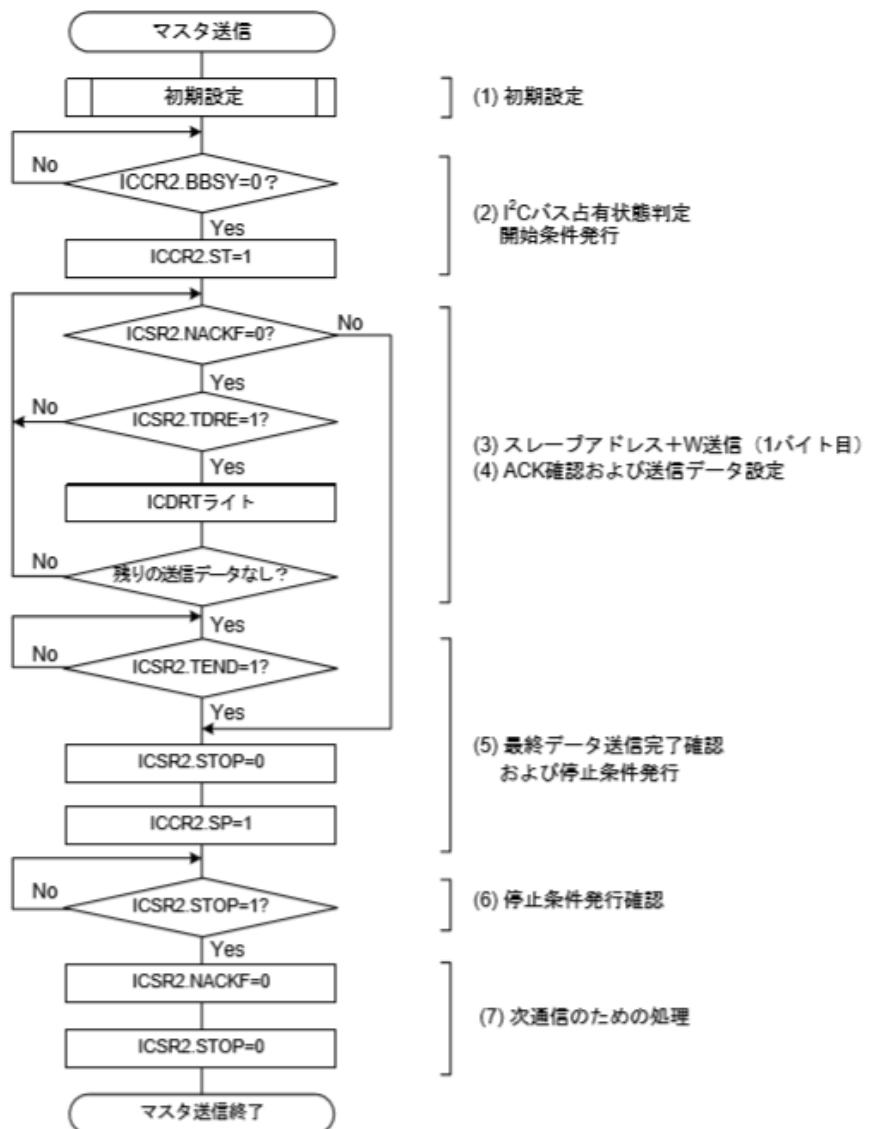


図 4.6 I2C 通信, 送信時プログラミング手順

3.1 Reading angular rate data

3.1.1 Using the status register

The device is provided with a STATUS_REG which should be polled to check when a new set of data is available. The reading procedure should be the following:

1. Read STATUS_REG
2. If STATUS_REG(3) = 0 then go to 1
3. If STATUS_REG(7) = 1 then some data have been overwritten
4. Read OUT_X_L
5. Read OUT_X_H
6. Read OUT_Y_L
7. Read OUT_Y_H
8. Read OUT_Z_L
9. Read OUT_Z_H
10. Data processing
11. Go to 1

図 4.7 ジャイロセンサーのデータ受信手順

始めにマイコン側で通信速度や使用するレジスタの設定を行いジャイロセンサーを起動させるために、書き込みでジャイロの設定するレジスタのアドレスを送り起動するようにと設定を通信で書き換える。次にジャイロが起動しているかを調べるために読み込みでステータスレジスタのデータをもらうことによって起動しているかどうかを調べる。ここでジャイロが起動していなければ、もう一度調べるところから戻る。無事起動できていれば、ジャイロの欲しいデータがあるレジスタを書き込みでアドレスを送り、指定したのち、読み込みでデータをもらう。なお、ジャイロからもらうデータはX軸Y軸Z軸の上位と下位それぞれ8bitずつに分けられた6つのデータをもらう。

4.1.3 プログラム製作

製作した主な関数の大まかな流れが以下のようになり、それぞれ使用する関数の説明を次に説明する。

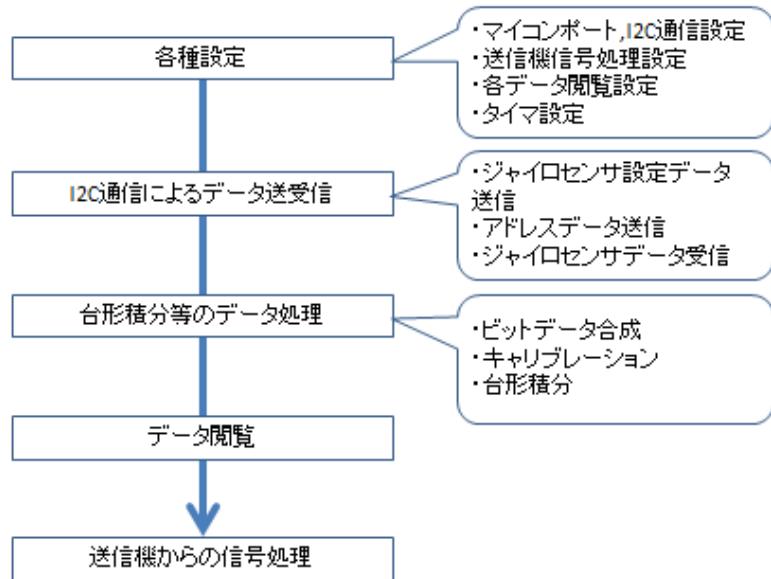


図 4.8 使用関数フローチャート

メイン

- 1) 関 数 名:main()
- 2) 機 能: こ こ で 各 関 数 の 处 理 を 行 う.
- 3) 引 数: な し
- 4) 戻 り 値: な し

マイコンポート設定

- 1) 関 数 名:i2c_portset()
- 2) 機 能: マ イ コン で 使用 す る ポ ー ト を 起 動 す る.
- 3) 引 数: な し
- 4) 戻 り 値: な し

マイコン I2C 通信設定

- 1) 関 数 名:i2c_set()
- 2) 機 能: I2C 通 信 の 通 信 速 度 等 の 設 定. マ イ コン マ ニ ュ ア ル の RIIC0 の 説

明をもとに通信するデバイスに合わせた設定をする。

3) 引数:なし

4) 戻り値:なし

送信機信号処理設定

1) 関数名:receiver_main()

2) 機能:送信機の信号処理に必要なマイコン内のMTU機能,PWM機能の設定。

3) 引数:なし

4) 戻り値:なし

各データ閲覧設定

1) 関数名:init_serial()

2) 機能:シリアル通信を利用し,パソコンでジャイロセンサのデータを見れるようにするための設定。

3) 引数:なし

4) 戻り値:なし

タイマ設定

1) 関数名:time_CMT0()

2) 機能:一定時間ごとにジャイロセンサのデータを貰えるようになる.CKSとCMCORのビットでデータを貰う速度を変えられる。

3) 引数:なし

4) 戻り値:なし

ジャイロセンサ設定データ送信

1) 関数名:gyro_set()

2) 機能:ジャイロセンサが測定する軸等の設定をI2C通信のデータ送

信を利用して設定する。今回は特に複数のデータを送るわけではないため引数は使わず直接データ等を打ち込んでいる。

3) 引数:なし

4) 戻り値:なし

アドレスデータ送信

1) 関数名:write_addr()

2) 機能:欲しいデータがあるデバイスにアドレスで指定する。引数にアドレスを書き込むことで、アドレスデータを送れる。

3) 引数:通信したいスレーブデバイスのアドレス

4) 戻り値:なし

ジャイロセンサデータ受信

1) 関数名:read_data()

2) 機能:指定したデバイスからデータを貰う。貰ったデータはreturnで引数に格納されることにより,X = read_data()でデータを別の変数に書き込むことが可能。

3) 引数: returnで返されたジャイロセンサのデータ。

4) 戻り値:ジャイロセンサから貰ったデータ。

ビットデータ合成

1) 関数名: deta_H_ave(),deta_HL()

2) 機能:上位8bitに2 6 5をかけ、下位8bitのデータと足し合わせることで1つの16bitのデータとなる。

3) 引数:X,Y,Zのhighとlowデータ

4) 戻り値:highとlowが1つとなったデータ

キャリブレーション

- 1) 関数名:cya()
- 2) 機能:ジャイロセンサのデータを平均化させその値と比べることにより、角速度に近い値を出す。
- 3) 引数:なし
- 4) 戻り値:キャリブレーションデータ

台形積分

- 1) 関数名:integral()
- 2) 機能:ビットデータ合成で得た角速度データをここで計算して、角度データにする。
- 3) 引数:ビットデータ合成で得た角速度データ。
- 4) 戻り値:積分結果。

送信機信号処理

- 1) 関数名:receiver_main_while()
- 2) 機能:送信機からの信号を各ESCに送る信号へ処理する。
- 3) 引数:なし
- 4) 戻り値:なし

各データ閲覧

- 1) 関数名:deta_look(),deta_cross(),serial()
- 2) 機能:ジャイロセンサから受け取ったデータや、マイコン内で処理しているデータを引数として関数に組み込むとデータが見えるようになる。deta_look()でジャイロセンサからそのまま受け取ったデータ,deta_cross()でビットデータ合成で得たデータ,serial()で台形積分した計算結果が閲覧できる。
- 3) 引数:それぞれのジャイロセンサのデータ。

4) 戻り値：なし

第5章

角速度及び角度の取得について

ここではドローンを飛ばすための制御プログラムについて説明する。

5.1 MTU機能を利用したマルチコプタの方向転換方法

受信機からマイコンへ送られる信号を図5.1 インプットキャプチャで拾いTGRAとTGRBに反映し、これらの値を引くことによりパルス幅を作り出しESCへ、そのパルスを送る仕組みとなっている。

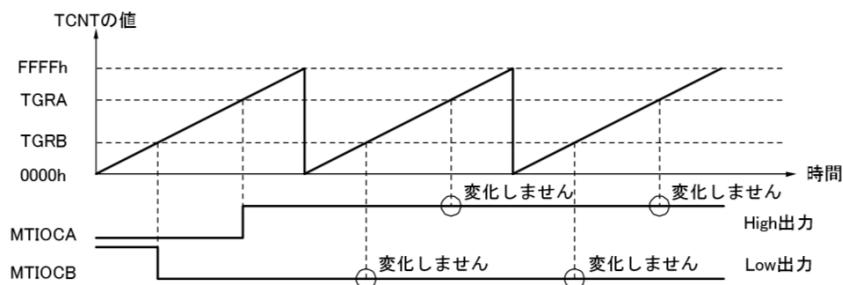


図5.1 インプットキャプチャ

5.2 角度推定方法

ジャイロから送られるデータは角度ではなく角速度である。よって、ジャイロから送られた角速度のデータを角度に変換する動作をプログラムにする必要がある。ここではその方法を説明する。

5.2.1 データキャリブレーション

ただジャイロセンサーから取った値ではまだ使用できるものではなく、最初に図 5.2 のように大体 10000 近くのデータを取得しそのデータを平均した値を制御に反映させるデータから引く動作が必要となる。これをキャリブレーションという。

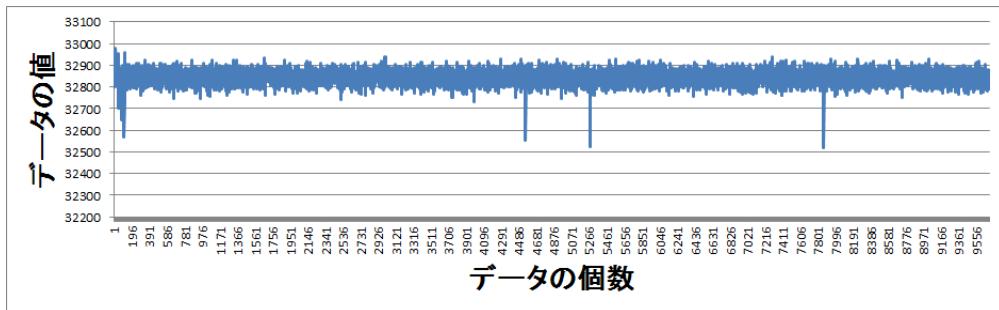


図 5.2 キャリブレーション用データ

5.2.2 台形法での積分方法

ジャイロセンサーからもらえるデータはあくまで角速度のデータなので積分をして角度に直す必要がある。今回は台形法という定積分を近似計算する方法を使用し、常時値を足し合わせることによって角度を求める。使用する式は以下の式(5.1)となる。

$$d = \sum \frac{(a+b)t}{2} \quad (5.1)$$

表 5.1 文字意味

d	角度
a	ひとつ前のデータ
b	新しいデータ
t	時間

5.2.3 データを取った結果

キャリブレーションで得たデータが以下の図 5.3 となりそこから式 (5.1) で計算し、得た値を図 5.4 に示す。



図 5.3 キャリブレーションしたデータ

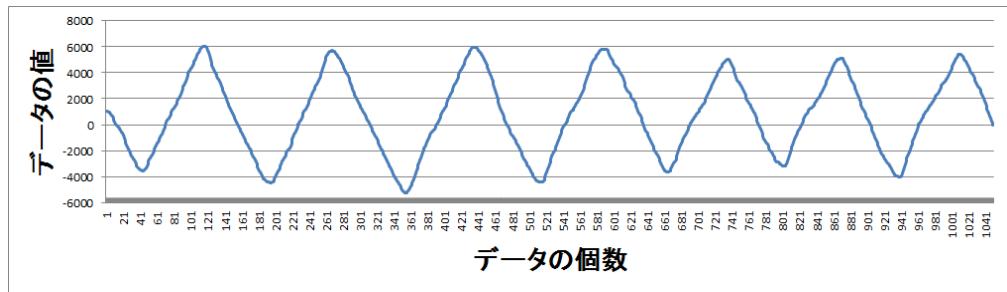


図 5.4 台形積分したデータ

しかし、これらの取得したデータは生データであって物理量ではないためこの角度のデータ量はどれほどの大きさなのかは不明である。しかし、図 5.4 のデータの個数 3400 以降でセンサを 1 秒ごとに左右に 30 度ほど傾けた時のデータ値の揺れ幅の大きさが静止した時との差でどれ程の値の大きさなのかが分かる。

5.3 フィードバック制御の流れ

ジャイロセンサからもらったデータを処理しドローンの制御をするためにフィードバック制御のサイクルを使用しの図 5.5 となっている。

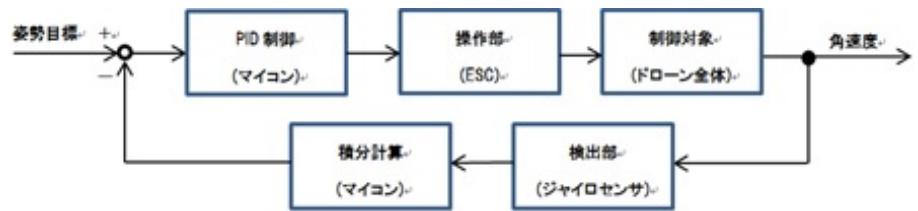


図 5.5 フィードバック制御

第6章

結論

6.1 本研究のまとめ

データの取得及びまで至れたが、今だドローンを飛ばすまでには到達できていない。しかし、後は取得したデータをもとに角度を算出し、各ドローンの制御プログラムを合わせ、ジャイロセンサの値でモーターを制御するというところまで到達しているため、残りの時間の間この製作をしていく予定である。

6.2 今後の課題

センサを90度に傾けた時のデータとジャイロから得た生データを比べ、角度を算出する。このデータをもとにして安定化制御を行う。

参考文献

- [1] 新海栄治, 石黒裕紀, 伊藤彩子, 仲めぐみ, 藤澤幸穂: RXマイコンのすべて, 株式会社 電波新聞社, 2012
- [2] I2C通信の使い方-電子工作の実験室,<http://www.picfun.com/c15.html>,

謝辞

本論文作成にあたりテーマの決定, 研究の考え方, 方法のまとめ方など全てにおいて長期にわたって厳しくも熱意のあるご指導, ご鞭撻していただいた, 伊藤恒平教授に厚く御礼申し上げます。

特に分析においても論文の書き方においても論文を何度も読んでいただき, 指導していただいた伊藤恒平教授に大変ご苦労をかけてしましましたことにも心よりお詫び申し上げたいです。

同級生のメンバーには論文の作成, 修正にご協力いただき心より感謝しております。その他、助けていただいた多くの皆様に心から感謝しております。ありがとうございました。

付録 A

プログラム

saucecode A.1 メイン関数 (RX621)

```
*****  
/*  
 */  
/* FILE : I2C.c  
 */  
/* DATE : Tue, Sep 05, 2017  
 */  
/* DESCRIPTION : Main Program  
 */  
/* CPU TYPE : RX62N  
 */  
/*  
 */  
/* This file is generated by Renesas Project Generator (Ver.4.52).  
 */  
/* NOTE: THIS IS A TYPICAL EXAMPLE.  
 */  
/*  
 */  
*****
```

```
#include "iodefine.h"  
#include "CMT.h"
```

```

#include "vect.h"
#include "typedefine.h"

//#include "typedefine.h"

#ifndef __cplusplus
//#include <iostream> // Remove the comment when you use
//_SINT ios_base::Init::init_cnt; // Remove the comment when you use
#endif

void main(void);
#ifndef __cplusplus
extern "C" {
void abort(void);
}
#endif
#include "rxserial.h"

void main(void)
{
    init_serial(); // データ閲覧設定
    i2c_portset(); // マイコンポート設定
    i2c_set(); // I2C通信設定

    gyro_set(); // ジャイロ設定データ送信

    // キャリブレーション
    cya();

    // 送信機信号処理設定
    receiver_main();
}

```

```
    time_CMT0(); // タイマー設定タイマー開始,  
    // 割込みによるジャイロセン  
サのデータ取得
```

```
    while(1) receiver_main_while(); // 送信機信号処理  
}
```

```
#ifdef __cplusplus  
void abort(void)  
{
```

```
}
```

```
#endif
```

soucecode A.2 マイコンポート設定 (RX621)

```
#include "CMT.h"  
#include "vect.h"  
#include "typedefine.h"  
#include "iodefine.h"  
  
void i2c_portset(void)  
{  
    PORTD.DDR.BYTE=0xff;  
    PORTD.DR.BYTE=0x00;
```

```
    /* port set */  
    PORT1.DDR.BIT.B3=0;  
    PORT1.DDR.BIT.B2=0;  
    PORT1.ICR.BIT.B3=1;  
    PORT1.ICR.BIT.B2=1;
```

```
}
```

```
void time_CMT0(void)
```

```
{
```

```
    MSTP(CMT0)=0;
```

```

CMT0.CMCR.BIT.CKS=1;
CMT0.CMCOR=37500;           //0.05 s?
CMT0.CMCR.BIT.CMIE=1;

IEN(CMT0,CMI0)=1;
IPR(CMT0,CMI0)=1;

CMT.CMSTR0.BIT STR0=1;

}

```

saucecode A.3 ジャイロセンサ送受信プログラム (RX621)

```

#include "iodefine.h"

// ジャイロモード設定アドレス送信
void gyro_set()
{
    // I2C バスビジー、エンプティ判定
    while (RIIC0.ICCR2.BIT.BBSY==1);

    // スタートコンディション発行
    RIIC0.ICCR2.BIT.ST=1;

    if (RIIC0.ICSR2.BIT.NACKF==0)
    {
        // エンプティフラグ
        while (RIIC0.ICSR2.BIT.TDRE==0);

        // 送信モードアドレス送信
        RIIC0.ICDRT=0xd2;

        while (RIIC0.ICSR2.BIT.TDRE==0);

        // データアドレス指定
    }
}

```

```

RIIC0.ICDRT=0x20;

while(RIIC0.ICSR2.BIT.TDRE==0)PORTD.DR.BYTE=0x02;

// データ送信
RIIC0.ICDRT=0x8f;

while(RIIC0.ICSR2.BIT.TEND==0)PORTD.DR.BYTE=0x01;

// ストップ動作
RIIC0.ICSR2.BIT.STOP=0;
RIIC0.ICCR2.BIT.SP      =1;

}

else if(RIIC0.ICSR2.BIT.NACKF==1)
{

RIIC0.ICSR2.BIT.STOP=0;
RIIC0.ICCR2.BIT.SP      =1;

}

while(RIIC0.ICSR2.BIT.STOP==0);
PORTD.DR.BYTE=0x00;

// 次の通信への処理
RIIC0.ICSR2.BIT.NACKF=0;
RIIC0.ICSR2.BIT.STOP=0;
}

// レジスタアドレス送信
void write_addr(unsigned char addr)
{
    // i2c バスビジー、エンプティ判定
    while(RIIC0.ICCR2.BIT.BBSY==1);
}

```

```

// スタートコンディション発行
RIIC0.ICCR2.BIT.ST=1;

// エンプティフラグ
while(RIIC0.ICSR2.BIT.TDRE==0);

// 送信モードアドレス送信
RIIC0.ICDRT=0xd2;

while(RIIC0.ICSR2.BIT.TDRE==0);

if(RIIC0.ICSR2.BIT.NACKF==0){

    // データアドレス指定
    RIIC0.ICDRT=addr;
    while(RIIC0.ICSR2.BIT.TDRE==0);

    while(RIIC0.ICSR2.BIT.TEND==0);

    // ストップ動作
    RIIC0.ICSR2.BIT.STOP=0;
    RIIC0.ICCR2.BIT.SP      =1;
}

else if(RIIC0.ICSR2.BIT.NACKF==1)
{

    RIIC0.ICSR2.BIT.STOP=0;
    RIIC0.ICCR2.BIT.SP      =1;
}

while(RIIC0.ICSR2.BIT.STOP==0);

// 次の通信への処理

```

```

RIIC0 . ICSR2 . BIT . NACKF=0;
RIIC0 . ICSR2 . BIT . STOP=0;
}

// データをもらう
char read_data(char data)
{
    // マスター受信開始
    while(RIIC0 . ICCR2 . BIT . BBSY==1);
    RIIC0 . ICCR2 . BIT . ST=1;

    while(RIIC0 . ICSR2 . BIT . TDRE==0);

    // 受信モードアドレス送信
    RIIC0 . ICDRT=0xd3;
    while(RIIC0 . ICSR2 . BIT . RDRF!=1);

    // データ受信動作
    if(RIIC0 . ICSR2 . BIT . NACKF==0)
    {
        RIIC0 . ICMR3 . BIT . WAIT=1;
        RIIC0 . ICMR3 . BIT . RDRFS=1;
        RIIC0 . ICMR3 . BIT . ACKBT=1;

        // ダミー受信
        data=RIIC0 . ICDRR;

        while(RIIC0 . ICSR2 . BIT . RDRF!=1);

        // ストップ動作
        RIIC0 . ICSR2 . BIT . STOP=0;
        RIIC0 . ICCR2 . BIT . SP      =1;

        // データ受信
    }
}

```

```

deta=RIIC0.ICDRR;

RIIC0.ICMR3.BIT.ACKBT=1;
RIIC0.ICMR3.BIT.WAIT=0;

}

else if(RIIC0.ICSR2.BIT.NACKF==1)
{
    RIIC0.ICSR2.BIT.STOP=0;
    RIIC0.ICCR2.BIT.SP      =1;

    // ダミー受信
    deta=RIIC0.ICDRR;
}

while(RIIC0.ICSR2.BIT.STOP==0);

// 次の通信への処理
RIIC0.ICSR2.BIT.NACKF=0;
RIIC0.ICSR2.BIT.STOP=0;

return(deta);
}

```

soucecode A.4 ジャイロセンサデータ処理プログラム (RX621)

```

#include "iodefine.h"
#include "Deta.h"
#include "CALC.h"

unsigned char status;
long x_deta_ave, y_deta_ave, z_deta_ave;
int c;

void read_xyz()
{

```

```

// x1 データ受信
write_addr(0x28);
Xl=read_data();

// xh データ受信
write_addr(0x29);
Xh=read_data();

// y1 データ受信
write_addr(0x2a);
Yl=read_data();

// yh データ受信
write_addr(0x2b);
Yh=read_data();

// z1 データ受信
write_addr(0x2c);
Zl=read_data();

// zh データ受信
write_addr(0x2d);
Zh=read_data();

//PORTD.DR.BYTE=0x03;

}

// キャリブレーション用データ合成
void deta_HL_ave(void)
{
    // データを合わせる
    x_deta=Xh*256+Xl;
    y_deta=Yh*256+Yl;
    z_deta=Zh*256+Zl;
}

```

```

x_deta=x_deta -65536/2;
y_deta=y_deta -65536/2;
z_deta=z_deta -65536/2;

}

// データを合わせた時の値を見る
void deta_cross(long x,long y,long z)
{
    char deta[10];

    itoa(x,deta);
    sendString(deta);
    sendString(",");
    itoa(y,deta);
    sendString(deta);
    sendString(",");
    itoa(z,deta);
    sendString(deta);
    sendString("\r");
}

// 積分結果を見る
void serial(long x,long y,long z)
{
    char detas[10];

    if(x<0){
        sendString("-");
    }
    itoa(x,detas);
    sendString(detas);
    sendString(",");
    if(y<0){
        sendString("-");
    }

```

```

        }
        itoa(y, detas);
        sendString(detas);
        sendString(",");
        if(z<0){
            sendString("-");
        }
        itoa(z, detas);
        sendString(detas);
        sendString("\r");
    }

}

```

```

// 積分する
void integral(long x, long y, long z)
{
    long x_deta_before, y_deta_before, z_deta_before;
    long Sum_x, Sum_y, Sum_z;
    long dx, dy, dz;
    long i;

    // 台形面積求める
    dx=((x_deta_before+x)/2)/30;
    dy=((y_deta_before+y)/2)/30;
    dz=((z_deta_before+z)/2)/30;

    // 前のデータ保存
    x_deta_before=x;
    y_deta_before=y;
    z_deta_before=z;

    // 角度計算
    x_de=dx+x_de;

```

```

y_de=dy+y_de ;
z_de=dz+z_de ;

PORTD.DR.BYTE= 2;

// 角度データを見る
serial(x_de,y_de,z_de);

}

// 安定化制御用のデータを合わせる
void deta_HL(void)
{
    long x_d,y_d,z_d;

    // データを合わせる
    x_deta=Xh*256+Xl;
    y_deta=Yh*256+Yl;
    z_deta=Zh*256+Zl;

    x_d=x_deta-x_deta_ave;
    y_d=y_deta-y_deta_ave;
    z_d=z_deta-z_deta_ave;

    // 台形積分する
    integral(x_d,y_d,z_d);

    // データを見る
    // deta_cross(x_d,y_d,z_d);
}

// 受信データを見る関数
void deta_look(void)

```

```

{
    char  deta [10];

    itoa(Xl,deta );
    sendString( deta );
    sendString( " ,");
    itoa(Yl,deta );
    sendString( deta );
    sendString( " ,");
    itoa(Zl,deta );
    sendString( deta );
    sendString( " ,");
    itoa(Xh,deta );
    sendString( deta );
    sendString( " ,");
    itoa(Yh,deta );
    sendString( deta );
    sendString( " ,");
    itoa(Zh,deta );
    sendString( deta );
    sendString( "\r ");

}

// キヤリブレーション時のデータを見る
void deta_ave(unsigned short x,unsigned short y,unsigned short z)
{
    char  deta [10];

    itoa(x,deta );
    sendString( deta );
    sendString( " ,");
    itoa(y,deta );
    sendString( deta );
    sendString( " ,");
}

```

```

    itoa(z,deta);
    sendString(deta);
    sendString("\r");
}

// ジャイロセンサの動作確認
void CTRL_REG1(void)
{
    char b[100];
    itoa(status,b);
    sendString(b);
    sendString("\r");
}

// データキャリブレーション
void cya(void)
{
    /*unsigned short x_a=9999;
    unsigned short y_a=9999;
    unsigned short z_a=9999;*/
    long x,y,z;
    int count=1;
    long cou=0;
    x=0;
    y=0;
    z=0;
    x_deta_ave=0;
    y_deta_ave=0;
    z_deta_ave=0;

    while(count<=10000){
        do{
            //ステータスアドレス送信
            write_addr(0x20);

```

```

    // ステータスデータ受信
    status=read_data();

    // ステータスレジスタ見る
    //CTRL_REG1();

} while( status&0x08!=0x08);

// 各xyz データ受信
read_xyz();

// deta_look();

// H とL のデータを合わせる
deta_HL_ave();
//
if(count>=3001){
    // deta_cross();
    x=x+x_deta;
    y=y+y_deta;
    z=z+z_deta;
    cou++;
}

if(count==10000){
    x_deta_ave=x/cou;
    y_deta_ave=y/cou;
    z_deta_ave=z/cou;

    /* x_a=(unsigned short)x_deta_ave;
    y_a=(unsigned short)y_deta_ave;
    z_a=/*cou;( unsigned short)z_deta_ave; */

}

count++;

```

```

    }
    // deta_ave(x_a,y_a,z_a);
}

```

soucecode A.5 送信機信号処理プログラム (RX621)

```

#include "iodefine.h"
#include "INP.h"
#include "vect.h"
#include "typedefine.h"

// レシーバ関係のプログラム

unsigned short /*Pwidth*/ S13 ,Yw1,Pt2 ,R14 ,FR ,FL ,RR ,RL ,Org1 ,Org2 ,Org4 ,MTU2fla

/* インプットキャプチャ関数 */
// 1ch 設定
void init_MTU2(void)
{
    //MTU2 reset?
    MTU2.TGRA=0;
    //WakeupMTU2
    MSTP(MTU2)=0;
    //ICU for MTIO2A is enable
    PORT2.DDR.BIT.B6=0;
    PORT2.ICR.BIT.B6=1;
    //PLCK/64
    MTU2.TCR.BIT.TPSC=3;
    //Input capture start by falling edge
    MTU2.TIOR.BIT.IOA=10;
    //TCNT
    MTU2.TCR.BIT.CCLR=1;
    //MTU2.TCNTA performs count operation
    MTUA.TSTR.BIT.CST2=1;
    //TGIEA of MTU2 interrupt is enabled
    MTU2.TIER.BIT.TGIEA=1;
}

```

```

// TGIEA of MTU2 interrupt is enabled
IEN(MTU2, TGIA2)=1;
// TGIEA of MTU2 interrupt priority level is 1
IPR(MTU2, TGIA2)=1;
// Org1=MTU2.TGRA;
}

// 2 c h 設 定
void init_MTU4(void)
{
    //MTU4 reset?
    MTU4.TGRA=0;
    //WakeupMTU4
    MSTP(MTU4)=0;
    //ICU for MTIO4A is enable
    PORT2.DDR.BIT.B4=0;
    PORT2.ICR.BIT.B4=1;
    //PLCK/64
    MTU4.TCR.BIT.TPSC=3;
    //Input capture start by falling edge
    MTU4.TIORH.BIT.IOA=10;
    //TCNT
    MTU4.TCR.BIT.CCLR=1;
    //MTU4.TCNTA performs count operation
    MTUA.TSTR.BIT.CST4=1;
    //TGIEA of MTU4 interrupt is enabled
    MTU4.TIER.BIT.TGIEA=1;
    //TGIEA of MTU4 interrupt is enabled
    IEN(MTU4, TGIA4)=1;
    //TGIEA of MTU4 interrupt priority level is 1
    IPR(MTU4, TGIA4)=1;
    //Org2=MTU4.TGRA;
}

// 3 c h 設 定
void init_MTU8(void)
{

```

```

//MTU8 reset?
MTU8.TGRA=0;
//WakeupMTU8
MSTP(MTU8)=0;
//ICU for MTIO8A is enable
PORTA.DDR.BIT.B6=0;
PORTA.ICR.BIT.B6=1;
//PLCK/64
MTU8.TCR.BIT.TPSC=3;
//Input capture start by falling edge
MTU8.TIOR.BIT.IOA=10;
//TCNT
MTU8.TCR.BIT.CCLR=1;
//MTU8.TCNTA performs count operation
MTUB.TSTR.BIT.CST2=1;
//TGIEA of MTU8 interrupt is enabled
MTU8.TIER.BIT.TGIEA=1;
//TGIEA of MTU8 interrupt is enabled
IEN(MTU8,TGIA8)=1;
//TGIEA of MTU8 interrupt priority level is 5
IPR(MTU8,TGIA8)=1;
}

// 4 c h 設定
void init_MTU9(void)
{
    //MTU9 reset?
    MTU9.TGRA=0;
    //WakeupMTU9
    MSTP(MTU9)=0;
    //ICU for MTIO9A is enable
    PORTB.DDR.BIT.B0=0;
    PORTB.ICR.BIT.B0=1;
    //PLCK/64
    MTU9.TCR.BIT.TPSC=3;
    //Input capture start by falling edge
}

```

```

MTU9.TIORH.BIT.IOA=10;
//TCNT
MTU9.TCR.BIT.CCLR=1;
//MTU9.TCNTA performs count operation
MTUB.TSTR.BIT.CST3=1;
//TGIEA of MTU9 interrupt is enabled
MTU9.TIER.BIT.TGIEA=1;
//TGIEA of MTU9 interrupt is enabled
IEN(MTU9,TGIA9)=1;
//TGIEA of MTU9 interrupt priority level is 1
IPR(MTU9,TGIA9)=1;
//Org4=MTU9.TGRA;
}

/* PWM 関数設定*/
void init_MTU0(void)
{
    //WakeupMTU0
    MSTP(MTU0)=0;
    //Counts on PCLK/1
    MTU0.TCR.BIT.TPSC=3;
    //TCNT0 is cleared by TGRA compare match
    MTU0.TCR.BIT.CCLR=1;
    //PWM mode
    MTU0.TMDR.BIT.MD=2;
    //((initial:High)->(CompareMatch:Low))
    MTU0.TIORH.BIT.IOA=2;
    //((initial:Low)->(CompareMatch:High))
    MTU0.TIORH.BIT.IOB=1;
    //count is 13ms
    MTU0.TGRA=4875;
    //PWM duty is 1.2ms
    //S13=450;
    MTU0.TGRB=S13;
    //割り込み
}

```

```

MTU0.TIER.BIT.TGIEA=1;
IEN(MTU0,TGIA0)=1;
IPR(MTU0,TGIA0)=6;
}

void init_MTU1(void)
{
    //WakeupMTU1
    MSTP(MTU1)=0;
    //Counts on PCLK/1
    MTU1.TCR.BIT.TPSC=3;
    //TCNT1 is cleared by TGRA compare match
    MTU1.TCR.BIT.CCLR=1;
    //PWM mode
    MTU1.TMDR.BIT.MD=2;
    //((initial:High)->(CompareMatch:Low)
    MTU1.TIOR.BIT.IOA=2;
    //((initial:Low)->(CompareMatch:High)
    MTU1.TIOR.BIT.IOB=1;
    //count is 13ms
    MTU1.TGRA=4875;
    //PWM duty is 1.2ms?
    //Pwidth=450;
    MTU1.TGRB=S13;
    //割り込み
    MTU1.TIER.BIT.TGIEA=1;
    IEN(MTU1,TGIA1)=1;
    IPR(MTU1,TGIA1)=3;
}

void init_MTU6(void)
{
    //WakeupMTU6
    MSTP(MTU6)=0;
    //Counts on PCLK/1
}

```

```

MTU6.TCR.BIT.TPSC=3;
//TCNT6 is cleared by TGRA compare match
MTU6.TCR.BIT.CCLR=1;
//PWM mode
MTU6.TMDR.BIT.MD=2;
//(initial:High)->(CompareMatch:Low)
MTU6.TIORH.BIT.IOA=2;
//(initial:Low)->(CompareMatch:High)
MTU6.TIORH.BIT.IOB=1;
//count is 13ms
MTU6.TGRA=4875;
//PWM duty is 1.2ms
//Pwidth=450;
MTU6.TGRB=S13;
//割り込み
MTU6.TIER.BIT.TGIEA=1;
IEN(MTU6,TGIA6)=1;
IPR(MTU6,TGIA6)=5;
}

```

```

void init_MTU7(void)
{
    //WakeupMTU7
    MSTP(MTU7)=0;
    //Counts on PCLK/1
    MTU7.TCR.BIT.TPSC=3;
    //TCNT7 is cleared by TGRA compare match
    MTU7.TCR.BIT.CCLR=1;
    //PWM mode
    MTU7.TMDR.BIT.MD=2;
    //(initial:High)->(CompareMatch:Low)
    MTU7.TIOR.BIT.IOA=2;
    //(initial:Low)->(CompareMatch:High)
    MTU7.TIOR.BIT.IOB=1;
    //count is 13ms
}

```

```

MTU7.TGRA=4875;
//PWM duty is 1.2ms
//Pwidth=450;
MTU7.TGRB=S13 ;
// 割り込み
MTU7.TIER.BIT.TGIEA=1;
IEN(MTU7,TGIA7)=1;
IPR(MTU7,TGIA7)=2;
}

/* カウントスタート */
void start_MTU0(void)
{
    // Count start
    MTUA.TSTR.BIT.CST0=1;
}
void start_MTU1(void)
{
    // Count start
    MTUA.TSTR.BIT.CST1=1;
}
void start_MTU6(void)
{
    // Count start
    MTUB.TSTR.BIT.CST0=1;
}
void start_MTU7(void)
{
    // Count start
    MTUB.TSTR.BIT.CST1=1;
}

/* PWM 全関数出力 */
void sta_MTU0(void)
{

```

```

    //MTU0 initialization
    init_MTU0();
    //MTU0 count start
    start_MTU0();

}

void sta_MTU1( void )
{
    //MTU1 initialization
    init_MTU1();
    //MTU1 count start
    start_MTU1();

}

void sta_MTU6( void )
{
    //MTU0 initialization
    init_MTU6();
    //MTU0 count start
    start_MTU6();

}

void sta_MTU7( void )
{
    //MTU7 initialization
    init_MTU7();
    //MTU7 count start
    start_MTU7();

}

// 送信機信号処理設定
void receiver_main( void )
{
    long i;
    //MTU2 initialization

```

```

init_MTU2 ();
init_MTU4 ();
init_MTU8 ();
init_MTU9 ();
sta_MTU0 ();
sta_MTU1 ();
sta_MTU6 ();
sta_MTU7 ();
if (MTU8.TGRA>730){
    while (MTU8.TGRA>730){
        FR=S13 ;
        FL=S13 ;
        RR=S13 ;
        RL=S13 ;
    }
    for ( i=0;i <=5000000;i++){
        FR=S13 ;
        FL=S13 ;
        RR=S13 ;
        RL=S13 ;
    }
}
for ( i=0;i <=100000;i++);
Org1=Yw1;
MTU2flag=1;
Org2=Pt2;
MTU4flag=1;
Org4=R14;
MTU9flag=1;

}

void serial_g (long x ,long y ,long z)
{
char deta [10];

```

```

    if (x<0){
        sendString (" - ");
    }
    itoa (x , detas );
    sendString (detas );
    sendString (",");
    if (y<0){
        sendString (" - ");
    }
    itoa (y , detas );
    sendString (detas );
    sendString (",");
    if (z<0){
        sendString (" - ");
    }
    itoa (z , detas );
    sendString (detas );
    sendString ("\r ");

}

// 送 信 機 信 号 处 理
void receiver_main_while (void)
{
    long x ,y ,z ;

    FR=S13+Pt2-R14+Yw1;
    FL=S13+Pt2+R14-Yw1;
    RR=S13-Pt2-R14-Yw1;
    RL=S13-Pt2+R14+Yw1;

    if (FR>710)FR=700;
    if (FR<460)FR=460;
}

```

```

if (FL>710)FL=700;
if (FL<460)FL=460;
if (RR>710)RR=700;
if (RR<460)RR=460;
if (RL>710)RL=700;
if (RL<460)RL=460;

x=X_gyro_esc ();
y=Y_gyro_esc ();
z=Z_gyro_esc ();

serial(x,y,z);

if (x<500){
    FR=FR+100;
    RR=RR+100;
    // FL=FL-100;
    // RL=RL-100;
}

if (x<-500){
    FL=FL+100;
    RL=RL+100;
    // FR=FR-100;
    // RR=RR-100;
}

if (y<700){
    FR=FR+100;
    FL=FL+100;
    // RR=RR-100;
    // RL=RL-100;
}

if (y<-700){
    RR=RR+100;
    RL=RL+100;
    // FR=FR-100;
}

```

```

        // FL=FL-100;
    }

}

void fr1( void )
{
    // FR1
    MTU7.TGRB=FR ;
}

void r12( void )
{
    // RL2
    MTU6.TGRB=RL ;
}

void f13( void )
{
    // FL3
    MTU1.TGRB=FL ;
}

void rr4( void )
{
    // RR4
    MTU0.TGRB=RR ;
}

void ch1( void )
{
    // 1 ch
    if (MTU2.TGRA<800){
        if (MTU2flag==0){
            Yw1=MTU2.TGRA ;

```

```

        }
        else {
            Yw1=MTU2.TGRA–Org1 ;
        }
    }

void ch2( void )
{
    //2 ch
    if (MTU4.TGRA<800){
        if (MTU4flag==0){
            Pt2=MTU4.TGRA;
        }
        else {
            Pt2=MTU4.TGRA–Org2;
        }
    }
}

void ch3( void )
{
    //3 ch
    if (MTU9.TGRA<800){
        if (MTU9flag==0){
            R14=MTU9.TGRA;
        }
        else {
            R14=MTU9.TGRA–Org4;
        }
    }
}

void ch4( void )
{

```

```

//4 ch
if (MTU8.TGRA<800){
    S13=MTU8.TGRA;
}
}

```

soucecode A.6 データ閲覧プログラム (RX621)

```

/*
 * rxserial.c
 *
 * Created on: 2017/08/16
 * Author: kouhei
 */

#include "iodefine.h"
#include "rxserial.h"

volatile char Buf[BUFSIZE];
volatile short Buf_start=0;
volatile short Buf_end=0;
volatile unsigned char Buffer_protect = 0;
volatile unsigned char SCI_stop = 1;

void init_serial(void)
{
    long i;

    Buf_start=0;
    Buf_end = 0;
    Buffer_protect = 0;
    SCI_stop = 1;
    /*
    MSTP(SCI1) = 0;                      // のスタンバイを解除SCI1
    //IOPORT
    PORT2.DDR.BIT.B6 = 1;      // PORT2-6 TxD1 出力に設定
    PORT3.DDR.BIT.B0 = 0;      // PORT3-0 RxD1 入力に設定
    PORT3.ICR.BIT.B0 = 1;      // 入力バッファ有効 [1]

```

```

//初期化スタートSCI1
SCI1.SCR.BYTE = 0; // SCR レジスタ初期化
SCI1.SCR.BIT.CKE = 0; // クロック選択 内臓ボーレートジェネレータ0: [2]
//SMR
SCI1.SMR.BIT.CKS = 0; // クロック選択 PCLK/1 n=0[3]
SCI1.SMR.BIT.MP = 0; // マルチプロセッサ通信機能 0: 禁止 1: 許可
SCI1.SMR.BIT.STOP = 0; // ストップビット長 0:1 ビット 1:2 ビット [3]
//SCI1.SMR.BIT.PM = 0; // パリティモードビット 0: 偶数パリティ 1: 奇数パリティ
SCI1.SMR.BIT.PE = 0; // パリティビットなし [3]
SCI1.SMR.BIT.CHR = 0; // データ長 0:8 ビット 1:7 ビット
SCI1.SMR.BIT.CM = 0; // コミニケーションモード 0: 調歩同期 1: クロック同期
SCI1.SCMR.BYTE = 0xF2; // スマートカードモードレジスタスマートカード使用しない(0xF2 初期値)
SCI1.BRR = 77; // ビットレート設定
9600bps PCLK=24MHz n=0 N=77
for(i=0;i<100000;i++); 謎のビット期間以上待つ秒か? // 1/(1/9600)
SCI1.TDR=0;
//SCI1.SCR.BIT.TIE = 1; // TXI 割込みを許可
IPR(SCI1,TXI1) = 0x7; // SCI1 の RXI1 の割り込み優先度を7に設定
IEN(SCI1,TXI1) = 0x1; // SCI1 の RXI1 の割り込みを有効化
IR(SCI1,TXI1) = 0x0; // SCI1 の RXI1 の割り込みフラグをクリア
IPR(SCI1,TEI1) = 0x7; // SCI1 の RXI1 の割り込み優先度を7に設定
IEN(SCI1,TEI1) = 0x1; // SCI1 の RXI1 の割り込みを有効化
IR(SCI1,TEI1) = 0x0; // SCI1 の RXI1 の割り込みフラグをクリア
RXI1
*/
//SCI1.SCR.BYTE = 0x30; // 受信許可ビットREと送信許可ビットTEを同時に1にして双方を許可する

```

```

MSTP(SCI3) = 0;           // SCI1 のスタンバイを解除

// IOPORT

PORT2.DDR.BIT.B3 = 1;    // PORT2-3 TxD3 出力に設定
PORT2.DDR.BIT.B5 = 0;    // PORT2-5 RxD3 入力に設定
PORT2.ICR.BIT.B5 = 1;    // 入力バッファ有効 [1]

// 初期化スタート SCI1

SCI3.SCR.BYTE = 0;        // SCR レジスタ初期化

SCI3.SCR.BIT.CKE = 0;    // クロック選択 0: 内臓ボーレート
                          // ジェネレータ [2]

// SMR

SCI3.SMR.BIT.CKS = 0;    // クロック選択 PCLK/1 n=0[3]
SCI3.SMR.BIT.MP = 0;     // マルチプロセッサ通信機能 0: 禁止
                          // 1: 許可

SCI3.SMR.BIT.STOP = 0;   // ストップビット長 0:1 ビット
                          // 1:2 ビット [3]

// SCI1.SMR.BIT.PM = 0;   // パリティモードビット 0: 偶数パリ
                          // ティ 1: 奇数パリティ

SCI3.SMR.BIT.PE = 0;     // パリティビットなし [3]

SCI3.SMR.BIT.CHR = 0;    // データ長 0:8 ビット 1:7 ビット

SCI3.SMR.BIT.CM = 0;    // コミニケーションモード 0: 調歩
                          // 同期 1: クロック同期

SCI3.SCMR.BYTE = 0xF2;   // スマートカードモードレジスタ
                          // スマートカード使用しない(0xF2 初期値)

SCI3.BRR = 77;           // ビットレート設定

9600bps PCLK=24MHZ n=0 N=77

for(i=0;i<100000;i++); // 謎のビット期間以上待つ(1/9600 秒
                          // か?)

SCI3.TDR=0;

// SCI1.SCR.BIT.TIE = 1; // TXI 割込みを許可

IPR(SCI3,TXI3) = 0x7;   // SCI3 のRXI3 の割り込み優先度を7に
                          // 設定

IEN(SCI3,TXI3) = 0x1;   // SCI3 のRXI3 の割り込みを有効化

IR(SCI3,TXI3) = 0x0;    // SCI3 のRXI3 の割り込みフラグをクリア

IPR(SCI3,TEI3) = 0x7;   // SCI3 のRXI3 の割り込み優先度を7に

```

設 定

```
IEN(SCI3,TEI3) = 0x1; // SCI3 のRXI3 の割り込みを有効化
IR(SCI3,TEI3) = 0x0;
}
```

```
int writeBuffer(char *s)
{
```

```
// short i;
int space;
```

```
// Buffer に書き込む余地があるか調べる
```

```
space=queryBufferSpace();
if (space<strlen(s)+10) {
```

```
    return 1;
}
```

```
    while(*s != '\0') {
```

```
        Buf[Buf_end]=*s;
```

```
        Buf_end++;

```

```
        Buf_end = Buf_end%BUFSIZE;
```

```
        s++;

```

```
}
```

```
    return 0;
}
```

```
int queryBufferSpace(void)
```

```
{
```

```
    if (Buf_start==Buf_end) return BUFSIZE-1;
```

```
    else return (BUFSIZE- Buf_end + Buf_start)%BUFSIZE-1;
```

```
}
```

```

int sendString(char *s)
{
    while( writeBuffer(s)==1);

    //PORTD.DR.BYTE= 1;
    SCI3.SCR.BYTE = 0xb0; //TIE , RIE , RE , TE を
    設定して割り込みを有効化
    if(SCI_stop){
        //PORTD.DR.BIT.B0=~PORTD.DR.BIT.B0;
        //PORTD.DR.BIT.B1=~PORTD.DR.BIT.B1;

        SCI3.SCR.BYTE = 0xb0; //TIE , RIE , RE ,
        TE を設定して割り込みを有効化
        SCI_stop = 0;
    }
    return 0;
}

int strlen(char *s)
{
    int cnt;
    cnt=0;
    while(*s != '\0'){
        s++;
        cnt++;
    }
    return cnt;
}

int itoa(long x, char *s)
{
    int i,j;
    if(x>0){

```

```

    for( i = 1; i*10<= x; i*=10);
    for(j = 0; 0 < i; i /=10 ,j++,s++) {
        *s = x / i + '0';
        x %= i;
    }
}

else {
    x=-x;

    for( i = 1; i*10<= x; i*=10);
    for(j = 0; 0 < i; i /=10 ,j++,s++) {
        *s = x / i + '0';
        x %= i;
    }
}
*s = '\0';
return 0;
}

```