2020 操舵引き継ぎ資料

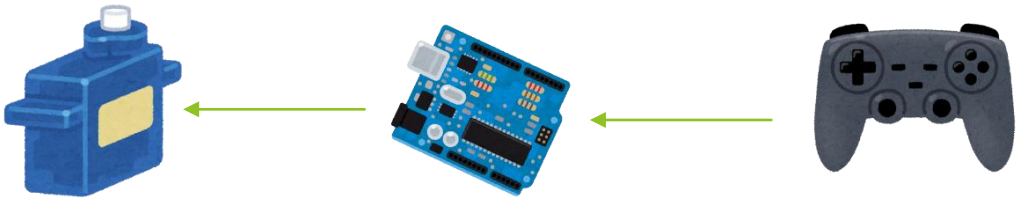
初めに

　まず初めに、今回の操舵はワイヤーリンケージではなく、フライ・バイ・ワイヤを用いて作成しています。それを念頭においた上で読んでください。フライ・バイ・ワイヤとはつまりマイコンでサーボモータを制御して、サーボモータで尾翼を駆動する方式のことです。

　また、このサーボモータと尾翼の接合方法についてはここでは説明を行えません。理由は筆者がその部分を担当していなかったためです。
ではどうするかですが、その部分は設計に任せてください。サーボモータの仕様を伝えて強度計算が行われたしっかりしたものを作ってもらうと安心です。

基本設計について

　操舵はアナログスティックを倒すことで、尾翼のサーボモータを駆動させる回路及びプログラムの作成が最終目的となります。



　そこで、必要になるのは「マイコン」「アナログスティック」の３つです。その他に、これらを繋ぎ合わせる基板の設計。マイコンへのプログラミングなどとやらなければいけないことは沢山あります。

マイコンについて

使用する製品について

使用するマイコンは試用に「Arduino Uno」を。実際に基板に組み込むために「Arduino Nano」を使用します。Nano は Uno の互換品で機能はそのまま、大きさだけ小さくなったものです。ただ、Nano は小さく、テストに使用するには不便なため、試し段階であれば Uno の使用が推奨されます。

Arduino Nano 販売ページ(秋月電子)
Arduino Uno 販売ページ(秋月電子)

メーカーは「Arduino Srl」で、上記の製品は秋月電子や千石電商、Amazon などでも販売されています。また、Amazon では廉価版、一般にはコピー商品と呼ばれるものが販売されているのでそちらを購入するのも良いでしょう。無論、廉価版でも機能に違いはありません。ただ、保護回路が省略されていたりするので、その点には注意が必要です。

製品の特徴について

- プログラム作成が楽で、プログラム作成のための資料が多くある。
  - Arduino 以上に初心者に優しいマイコンは無いため、これを使用している。
  - 入門用の学習書や、学習サイトが豊富なため、自分自身で学べる。

補足

以前は RENESAS のマイコンを使用していた。そちらはより軽量でより単価が安く、より高性能となっています。しかしながら使用するには一定以上の高度な技術と知識が必要となるので推奨しません。RENESAS 以外にも pic マイコンという安価なマイコンがあるため、使用するにはそちらを選択してください。なぜなら、RENESAS に比べて pic は多少資料が多いため、いざとなったらインターネットで検索をすれば使用方法などの知識を得やすいからです。

もし、Arduino から別のマイコンへと乗り換えるのであれば引継ぎはしっかりと行ってください。

アナログスティックについて

　アナログスティックに関しては、本資料を読むよりも先に該当のデータシートを見ることを強く推奨します。また、本資料にはデータシートには書いていない注意事項を記載しているので、その点については本資料で確認してほしいです。

使用する製品について

　使用しているアナログスティックは「SainSmart」の「JoyStick Module」で、秋月電子で販売されています。

JoyStick Module 販売ページ(秋月電子)

製品の特徴について

- 中央部と外縁部のデッドゾーンが大きく、繊細な動きをしづらい。

補足

　特徴の項目からも分かるように、この使用しているアナログスティックは性能が低いです。そのため、実際に機体に搭載する際はより高機能なアナログスティックを購入、使用する必要があります。また、アナログスティックを変更した場合は、それに応じてマイコン側のプログラムも適宜変更してください。

サーボモータについて

サーボモータはアナログスティックと異なり、データシートが複雑で読むのが辛いと思うので、ここではできるだけ噛み砕いて書いていきたいと思います。

ただ、噛み砕くということは細部について触れないということなので、そこについてはデータシートや、公式が配布している資料を読んでください。

使用する製品について

使用するサーボモータは「近藤科学」の「KRS-4031HV ICS」で、通販などで販売されています。(昔はツクモロボット王国という店で店頭販売されてたけど、最近はどこで買えるのかわからないです。申し訳ないです。)

近藤科学 公式販売ページ

製品の特徴について

- 「ICS3.5」という近藤科学独自のシリアル通信規格での制御が可能なため、信頼性が高い。無論、pwm 制御も対応しており、そちらを利用して制御することもできる。
- 駆動電圧が 9-12V で、リポバッテリーを直付けで駆動できる。
- サーボホーンは付属していないため、この製品に対応したものを購入する必要がある。
  例：近藤科学 サーボホーン公式販売ページ

補足

この他にもサーボモータは様々な企業が販売しているため、一番使いやすいサーボモータを探してみるのも良いでしょう。ただ、その際はトルクや駆動電圧には細心の注意を払う必要があります。また、近藤科学のサーボモータにも色々種類があるので、尾翼の駆動のために更にトルクが必要になった場合などは、トルクの大きいものに買い替えてください。

また、サーボモータを駆動するための ICS3.5 という通信規格ですが、当然ではありますが、Arduino でポンと出力することはできません。そのため、適切な回路設計とソフトウェア設計が必要です。

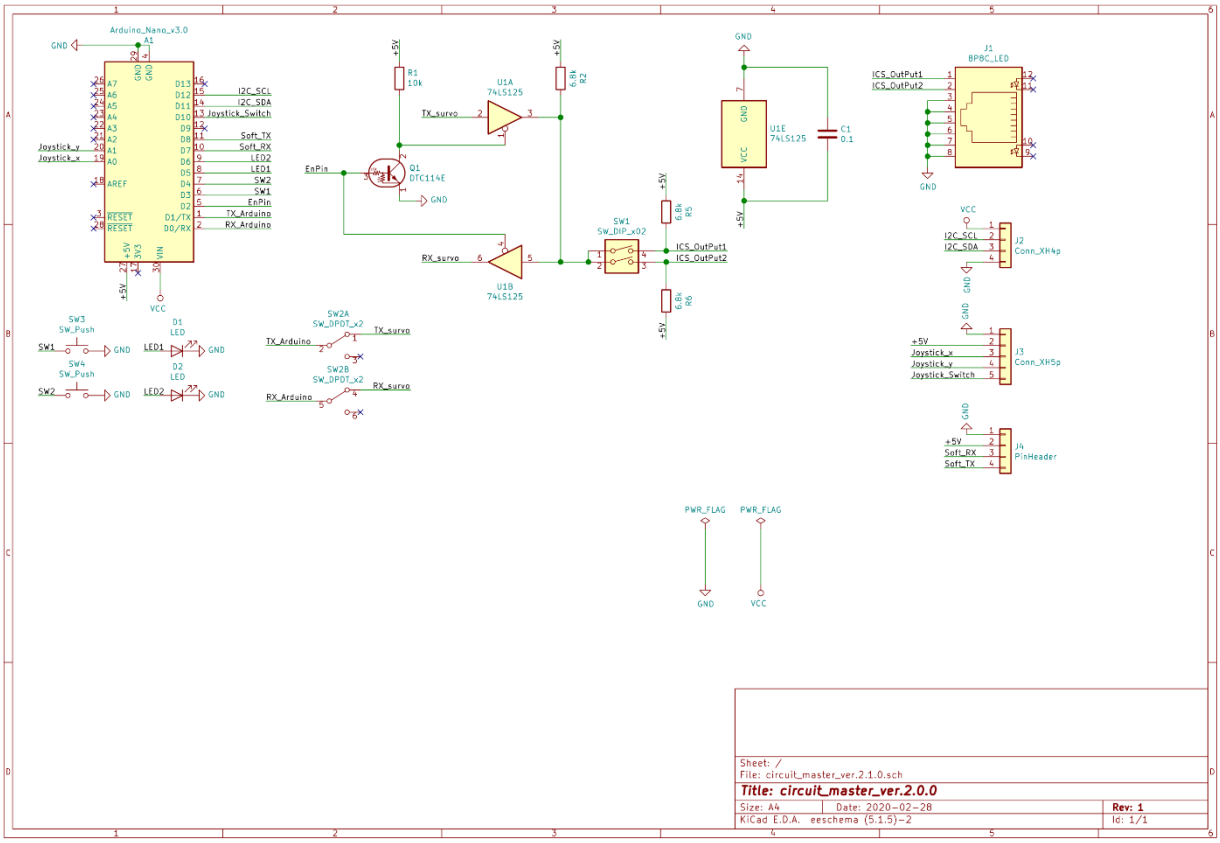
一応、公式側から Arduino 用のライブラリが公開されているのでそれを利用してもいいでしょう。(筆者は参考サイト様(リンク)を見ながら自分で組んだので、どのようにライブラリを利用するのか分からないので解説はできません。)

他にも公式から、ICS3.5 用のモジュールが購入できるみたいです。自分で回路を設計できない場合はそれを利用するのが一番安全です。

あと、ソフトウェア側の解説でも記述するのですが、Arduino のソフトウェアシリアル機能は 115200bps に対応していないので、Arduino で ICS3.5 を使用するならハードウェアシリアルを使用してください。

操舵基板(回路)について

　操舵基板というのは、マイコンとその他の電子部品を詰め込んだ基板のことを指します。下の画像が現在使用している操舵基板の回路図です。



操舵基板の役割について

　操舵基板は操舵の電気的部分に関するほぼ全てを担います。そのためこの基板と「メイン基板」「尾翼側にあるサーボモータ」「コックピットのアナログスティック」を接続する必要があります。

　また、情報処理もこの基板内で行いたいので、Arduino 本体もこの基板に搭載することとなります。

　それ以外にも、Arduino の動作状況を確認するための LED や、Arduino のデバッグを行うためのシリアル通信用のポート、Arduino を操作するためのスイッチも搭載しています。

設計について

回路図の設計は KiCad を用いて行いました。なので、回路図や基板を見る場合はそれを利用してください。自身で新規に基板や回路を作成する場合は、KiCad を用いても良いですし、Eagle を用いても良いでしょう。

なお、回路の設計には「KiCad Basics」と「自作基板への道」という参考資料を活用しました。この 2 つは引き継ぎの資料に同梱してあるので活用してください。

回路について

回路図とパーツについては、渡したデータを参照すれば、どれがどれに対応しているか分かるかと思います。

設計時の注意点

- サーボモータは Arduino のハードウェアシリアルで制御する必要があります。そのため、UART を ICS3.5 へ変換する回路はハードウェアシリアルの出力のピンへと接続しましょう。
- ICS3.5 へ変換する回路を接続したままだと、Arduino への書き込みができません。そのため、ここの接続を切るためのスイッチが必要です。上記回路図上だと SW2 に該当します。
- ハードウェアシリアルはサーボモータとの通信に専用されます。そのためデバッグ用にソフトウェアシリアルを用いて Arduino と通信するためのピンを用意しましょう。
  恐らくですが、Arduino Nano はソフトウェアシリアルに使用できるピンに制限はないと思われます。D2~D12 かに割り当てれば問題ないはずです。
- インジケータ用の LED と、スイッチは用意しておきましょう。忘れるソフトウェア設計面でも支障がでます。

備考(LAN コネクタについて)

今回の基板では LAN コネクタを搭載することにしています。これは操舵基板と尾翼のサーボモータを接続するためのものです。つまり、操舵基板とサーボモータは LAN ケーブルで接続するということです。

LAN ケーブルで接続するのは何点かメリットがあります。

1. 接続、切断がアタッチメントを用いることで容易に可能であること。
2. 挿し間違いが極めて発生しづらい。
3. ケーブルの代替品が簡単に用意できる。

もちろん、通常のケーブルと比べて重量の増加というデメリットも発生するので、安定重視なら LAN ケーブルを。効率重視なら専用ケーブルを選択するのが良いでしょう。

備考(各要素との接続端子について)

アナログスティックと操舵基板間

XH 規格の圧着端子をつけた銅線を用いて接続します。その際には「アナログスティック_x 軸」「アナログスティック_y 軸」「アナログスティック_スイッチ」「5V」「GND」の 5 ピン必要となるため、対応したものを選びましょう。

ケーブルは一般的な安価な銅線で問題ないでしょう。

サーボモータと操舵基板間

前述の通り、LAN ケーブルを用いて接続します。とは言え、LAN ケーブルとサーボモータを直接繋ぐことはできないので、LAN ケーブルをピンヘッダへ変換する基板を尾翼周辺に配置して、変換をした後にサーボモータへ接続することとなります。また、その変換基板にサーボモータ用電源も接続し、サーボモータはその変換基板とのみ接続できるようにしました。

操舵基板からは「GND」「サーボモータ 1 の通信線」「サーボモータ 2 の通信線」の 3 種類。LAN ピンヘッダ変換基板では、サーボモータそれぞれに「GND」「通信線」「VCC(12V)」の 3 つ。操舵基板との「GND」「サーボモータの通信線」×2 (ただし LAN を用いる)。サーボモータ電源(リポバッテリー)との「GND」「VCC(12V)」の 2 つ(ピンヘッダ)を用意します。

操舵基板(基板)について

前項では主に回路について記述しました。本項では基板の設計について書いていきます。基板の設計の画像はレイヤーでごちゃごちゃしているので載せません。KiCad を使って自身で実際に確認してみてください。

設計について

基板の設計ですが、回路の「設計について」の項目と内容は同じのため省略します。

発注について

発注は FusionPCB が恐らく一番安価だと思います。品質も問題ないので活用しましょう。

一応、学内のものつくりセンターに基板作成機がありますが、使用方法が面倒な上、品質もお世辞にも良いとは言えないので、おすすめしません。もしどうしても使用したいならば、事前にものつくりセンターに登録を行い、使用方法を確認しておきましょう。

使用方法は wiki に記述されているはずです。(なかったら申し訳ないです。)

設計及び発注時の注意点

- 一つ一つの電子素子(トランジスタとか)のデータシートをみて、ピン配置について確認しましょう。稀に予定していたものとズレていることがあります。また、回路図とも照らし合わせて考え通りに割り当てがなされているかを確認しましょう。
- 発注する際は必ず発注先のサイトでガーバーデータの生成方法を確かめましょう。そこに一通り生成方法が記述されているので、生成で手間取ることもなく進められるはずです。
- 誤発注を避けるため、ガーバーデータを提出したら、確認ページを見ましょう。そうすればきっと誤発注が防げる…はず。

Arduino に書き込むプログラムについて

この項では Arduino に書き込むプログラムについて解説します。恐らく長くなるので心して読んでください。なお、基本的な Arduino のプログラミングはこなしたものとして説明を行います。そのため、Arduino ってなんぞやって人は、事前に「Arduino 入門」とかで検索をして、少なくとも LED をボタンで光らせる程度はできるようになっておいてください。

ソースコードのダウンロードについて

Arduino へ書き込むソースコードは筆者の github 上→(リンク)にあります。また、ソースコードだけでなく、本資料や回路図などなどもあるので確認してみてください。

本番用のプログラムは「master」というディレクトリに入っているので、そこから最新のバージョンをダウンロードしてください。

概要

このプログラムは 6 つのファイルがあります。それぞれ以下のような役割を持ちます。

1. steering_master_ver.1.0.ino：setup 関数や loop 関数の存在するメイン部。
2. servo_ver.1.0.ino：サーボモータの駆動に関する関数群
3. analog_stick_ver.1.0.ino：アナログスティックに関する関数群
4. command_line_ver.1.0.ino：コマンドラインに関する関数群
5. memory_ver.1.0.ino：メモリの読み書きに関する関数群
6. subs_ver.1.0.h：構造体や変数、関数の宣言をまとめたもの。

これらそれぞれについて順に解説をしていきます。

subs_ver.1.0.h について

このファイルでは、他のファイルが使う変数、定数、関数、構造体の宣言を行っています。

Steering_master_ver.1.0.ino について

まずは各種パラメータの初期値を設定する。ここの値は自由に設定して構わないです。

次に setup 関数だが、ここにある関数 init_stick, init_cmd, init_servo, init_memory は subs_ver.1.0.h にて宣言されている。定義は別ファイルにあります。

Arduino が起動されてループすることになる loop 関数だが、ここではサーボの駆動、デバッグモードへの移行判定、スティックの押し込み判定を行っています。また、デバッグモードへ移行した場合は、ロードスイッチが押されるかの判定を行います。

スティックの押し込み、デバッグモード、パラメータロードスイッチの詳細については、ソースコードと同階層にある「read_me.txt」を読んでください。

servo_ver.1.0.ino について

基本的には参考サイトを基にコードを作成しています。特に、サーボモータへ送るコマンドの部分に関しては、完全にコピーしているだけです。（参考サイト様：https://qiita.com/kedtn/items/7174f5112d99f12ce0e7）

私が作成した部分について解説します。

関数 move_servo ではスティックの情報を取得。計算をし、そのデータを送信するだけとなっています。

関数 calc_angle ではスティックのアナログ値などから、サーボモータへ送る値を計算します。細かい計算式については、ソースコードと同階層にある「calc_angle について.txt」を読んでください。

calc_angle には計算式以外に気をつけなければいけないことがあります。計算する際の値が大きくなり、計算順序によっては値がオーバーフローします。そのため、calc_angle の計算式を変更する場合はそこに注意してください。また、挙動が変化した場合は、デバック用とコメントを残したところを使用して、送信値が正常かどうか判断してください。異常の場合は、()をつけて計算順を変化させるか、型を変えてオーバーフローしない型の使用をしてください。

関数 ics_set_id はサーボモータへ ID の書き込みをするものです。

注意

サーボモータへの ID 書き込み時は注意が必要です。接続しているサーボモータ全てへ ID の書き込みがされるからです。そのため、この関数を使用する際は、サーボモータへの接続を切断してください。物理的に切断してもいいですが、基板上の Servo selector スイッチを使用して切断しても OK です。

現状ある関数は少ないですが、必要に応じてコマンドを増やしてください。コマンドは ICS3.5 のソフトウェアマニュアルと、参考サイトを使用すれば作成できるはずです。

analog_stick_ver.1.0.ino について

ここでは、スティックのスイッチがつながっているピンの初期化用の init_stick。そして、スティックの各種状態値を取得する stick_state のみ記述しています。

command_line_ver.1.0.ino について

コマンドラインは、デバッグモード中に Arduino と対話形式でやり取りするためのものです。

システム自体は単純です。cmdline 関数は、まずシリアル通信の受信をおこない、コマンドの取得をします。その後に、そのコマンドに対応する関数が存在するかを検索します。存在すればその関数を実行します。

コマンドを追加したい場合は、上部の「COMMAND_TOTAL_NUM」を適宜増やし、コマンドテーブルにコマンドを追加。コマンド関数を作成し、コマンドが実行された際の処理内容を関数内に記述してください。

memory_ver.1.0.ino について

Arduino 内部には不揮発性メモリがあり、EEPROM の各種関数を使用して書き込みと読み込みができるようになっています。不揮発性なので電源を落としても値を保持できるので、いちいち Arduino へプログラムを書き込むことなくパラメータの変更をさせることができます。

関数 save_parameters では各種パラメータを保存することが、関数 load_parameters では各種パラメータを ROM から読み取ることができます。

既知のバグ

- command_line_ver.1.0.ino の help 関数ですが、現在はコマンドリストを開く機能は搭載されていません。
- command_line_ver.1.0.ino には、現在 centerPS_x, centerPS_y, speed, stretch を変更する関数は搭載されていません。
- command_line_ver.1.0.ino には、現在 stretch, speed をサーボモータへ書き込む関数は搭載されていません。

使用方法について

本当は最初に書くべきでしたが、最後になってしまい申し訳ありません。基本的な使用方法は順に。副次的な機能は箇条書きで記述します。

1. 操舵基板とスティック。操舵基板とメイン基板。操舵基板と尾翼の変換基盤。尾翼の変換基盤とサーボモータ。この４つを接続します。(この他にも接続するものがあったら適宜接続を)
2. すると、メイン基板からの給電で Arduino が起動するので、操作ができるようになります。
    - パラメータを読み込む場合は、デバッグモード移行スイッチ(SW4)を押してデバッグモードに移行します。その後、パラメータ読み込みスイッチ(SW3)を押して読み込みを行います。
    - スティックの中央値を変更したい場合はスティックを押し込みます。押し込んだら、スティックを中央に戻すか、1 秒経過すると、押し込んだ場所を中央としてサーボモータを操作することができるようになります。
    - PC とシリアル通信する場合は、シリアル変換基板と PC を、シリアル変換基板と操舵基板を接続して、デバッグモードを ON してください。そうすることで、コマンドラインが有効化されてコマンドによる操作ができます。
      なお、PC と Arduino でシリアル通信する場合ですが、PC のシリアル通信のターミナルは Arduino の IDE のシリアルモニタを使用してください。
      COM ポートをシリアル変換基板のものに設定してシリアルモニタを起動すれば OK です。

備考(注意点など)

まず、注意して欲しいのが、サーボモータへ ID を書き込むときは、書き込みたいサーボモータのみ接続するようにしてください。両方接続したままだと、両方へ書き込みが行われてしまいます。

次は Servo↔PC スライドスイッチですが、コレが PC 側になっていると PC から Arduino へ書き込みができます。Servo 側の場合、サーボモータへ通信されます。

他には混線や、故障の原因になる可能性がある操作は無いですが、製品では無いので注意してしすぎることはないので、変な使用方法をしないようにしてください。

トラブルシューティング(既知のバグの対処法)

- Arduino へ書き込みができない。
  - Servo↔PC スライドスイッチが Servo 側になっていませんか？PC 側にしないと書き込みは行えません。
- サーボモータへ通信されない。
  - Servo↔PC スライドスイッチが PC 側になっていませんか？Servo 側にしないと通信はされません。
  - Servo selector が off になっていませんか？on にしないと通信はされません。
  - サーボモータの認識している ID と、送信している ID に相違はありませんか？分からなければ一度 ID の書き込みをしましょう。

読んでる人がやらなくてはならないこと

コレを読んでる人はいかに書いてあることをとりあえずやらなければいけません。(私がやり残したことですが…)

- 基板を新たに作成する。私が作ったのはデバッグのために圧着端子や、LAN ソケットが全てピンソケットになっています。これをちゃんと圧着のものと LAN ソケットになっているものにしてください。
- 尾翼の変換基板を設計・作成する。LAN からサーボモータが接続できるようなピンヘッダへ変換するものです。
- 操舵基板、スティックなど諸々を固定する方法の考案。そして機体に固定できるようにしてください。

以下はできればですが

- ソフトウェアを読んで、足りない機能の補完。
- この資料の足りない箇所の追記。

これらをやってくれれば、操舵係として失敗することは無いはずです。

最後に

上記のことと、実際のソースコード。回路図などを見れば大体は分かるはずです。(中には書き漏れがあるかもですが…)
もし、これらでも分からなくて「詰まった」と思ったら筆者の私に連絡してください。相当忙しくなけれ手助けができると思うので。

以下筆者について
役職など：20 代代表兼電装
mail1：y1810678@edu.cc.uec.ac.jp
mail2：forestry.for@gmail.com