■Gotoronic

arduinoというMCU+開発環境を用い、市販で入手できる電子部品ほか材料を用いて
個人で手軽に電動変速器を自作するためのソリューションを提供する。

構成：MCUボード、変速機を駆動するサーボ、サーボ固定するステー、駆動用のロッド、
変速トリガを入れるスイッチ、バッテリ、ハーネス、etc…

■arduinoとは

[入門]Arduinoのはじめかた

いわく：一言でいうと「初心者でも簡単に扱えるマイコンボード」です。
また、オープンソース・ハードウェアと呼ばれるもので
回路からソフトウェアの全てが公開されています。
このため、これまで電子工作にあまりなじみのなかった人や
マイコンのプログラミングに詳しくない人でも、
すぐに始める事が出来るように考えられています、
このため電子工作の入門にお勧めの一つとなっています。

用語：IDE＝開発環境。各自PCにインストールする。
スケッチ＝arduino本体（MCUボード）に書き込むプログラム。

参考サイト

Arduino 日本語リファレンス

オリジナルの英文を訳した文章が主。わりとわかりやすい。

workshop/ServoBasis/main

サーボライブラリ詳説

■サーボの制御について

Gotoronicでは"writeMicroseconds(uS)"コマンドを主に用いている。
サーボに対しマイクロ秒単位で角度を指定します。標準的なサーボでは、
1000で反時計回りにいっぱいまで振れます。2000で時計回りいっぱいです。
製品によっては、この範囲に収まらず700～2300といった値を取るものもあります。
※1usecごとの移動θ目安　180deg/(2000usec-1000usec)＝0.18deg

Gotoronicでは1stギヤ位置をpos0として定義する。
この時サーボとしては動作最大角度に近い側になっている。
（※使うサーボ製品ごとに状況変わります）
段数が上がるにつれ、任意のステップ量だけ数値を減らす方向。

①arduino IDEのインストール

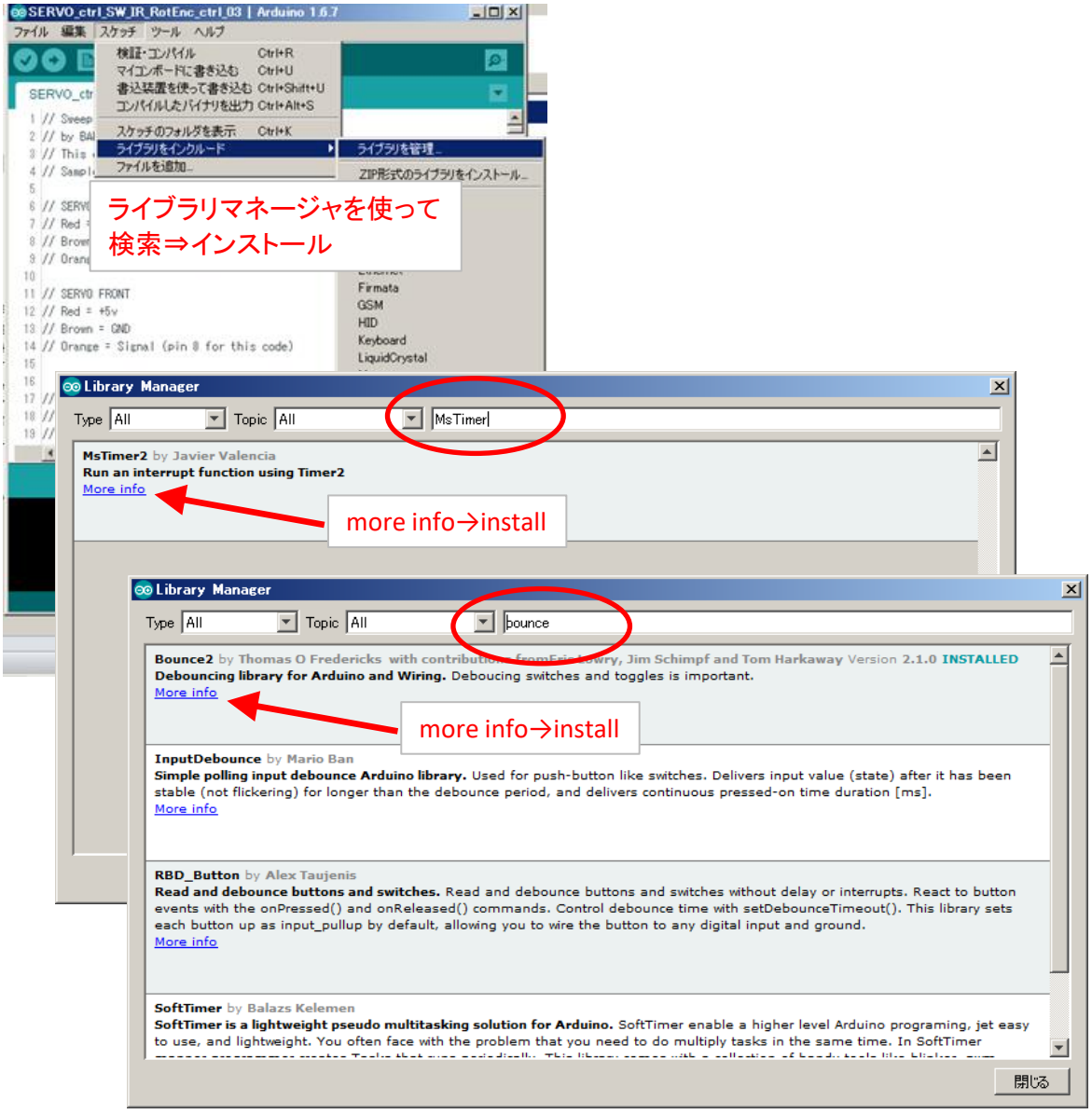
https://www.arduino.cc/
https://www.arduino.cc/en/Main/Software
からIDE（開発環境）をDL、インストールする。

②ライブラリの追加

標準装備外のライブラリを追加する必要があります。
progrmfiles＞Arduino＞libraliesの下に各ライブラリ名のフォルダを作って
ライブラリのzip内容を展開しておく。

・MsTimer2：タイマーライブラリ
http://www.musashinodenpa.com/arduino/ref/index.php?f=1&pos=2021

・Bounce2：チャタリング対策ライブラリ
http://playground.arduino.cc/Code/Bounce



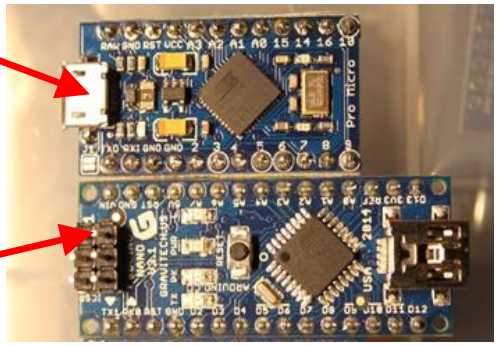
注）最近のIDEの機能です。
versionが古い場合は更新するか、
Arduino→libralies　のフォルダに
必要なファイルを格納してIDEを再起動する。

③ardinoMCUボードの導入

③−1　promicroの場合

https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide/installing-windows

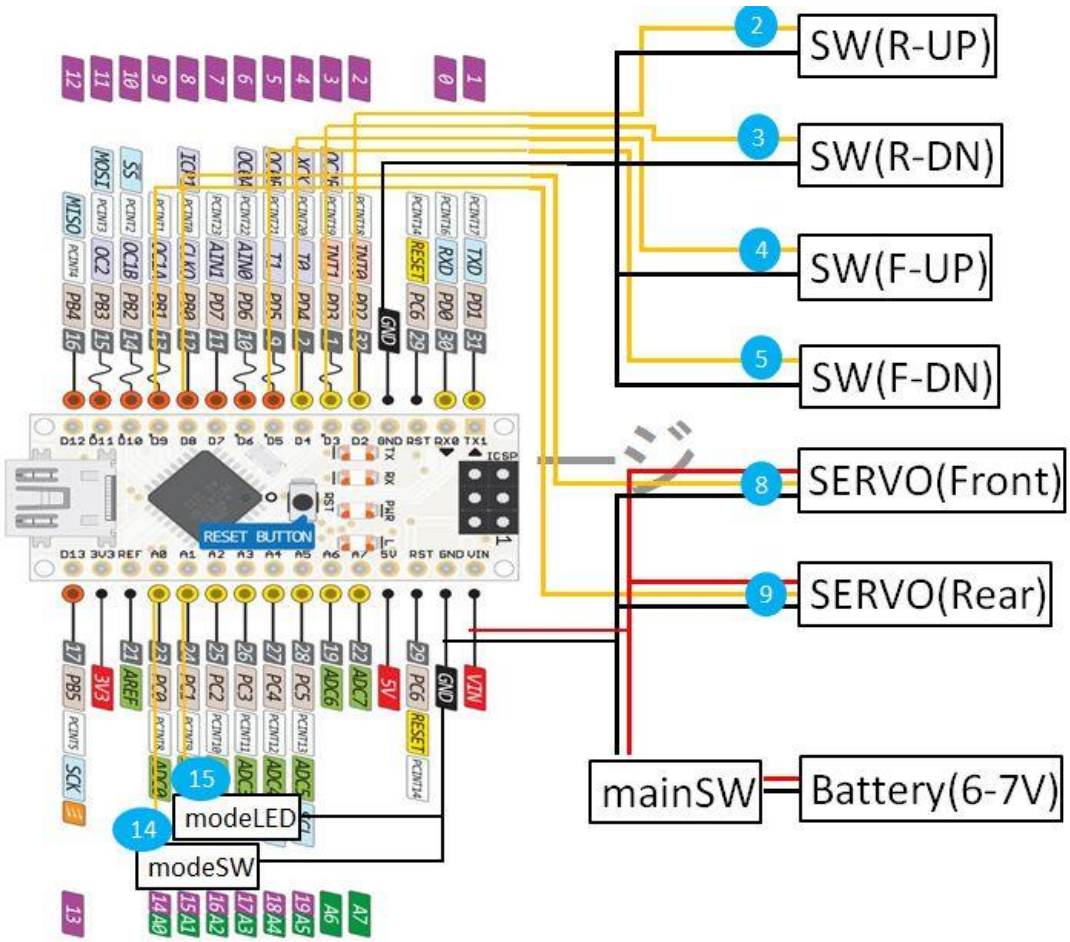
ドライバをDLしデスクトップ等に展開しておく。
MCUボードをPCにUSB接続する。
上記ドライバでセッティングする。
″ボードマネージャ″でMCU種を指定し、
設定する。上記URL中段からの説明参照。
注：MCU設定は5V16MHzを選ぶこと。



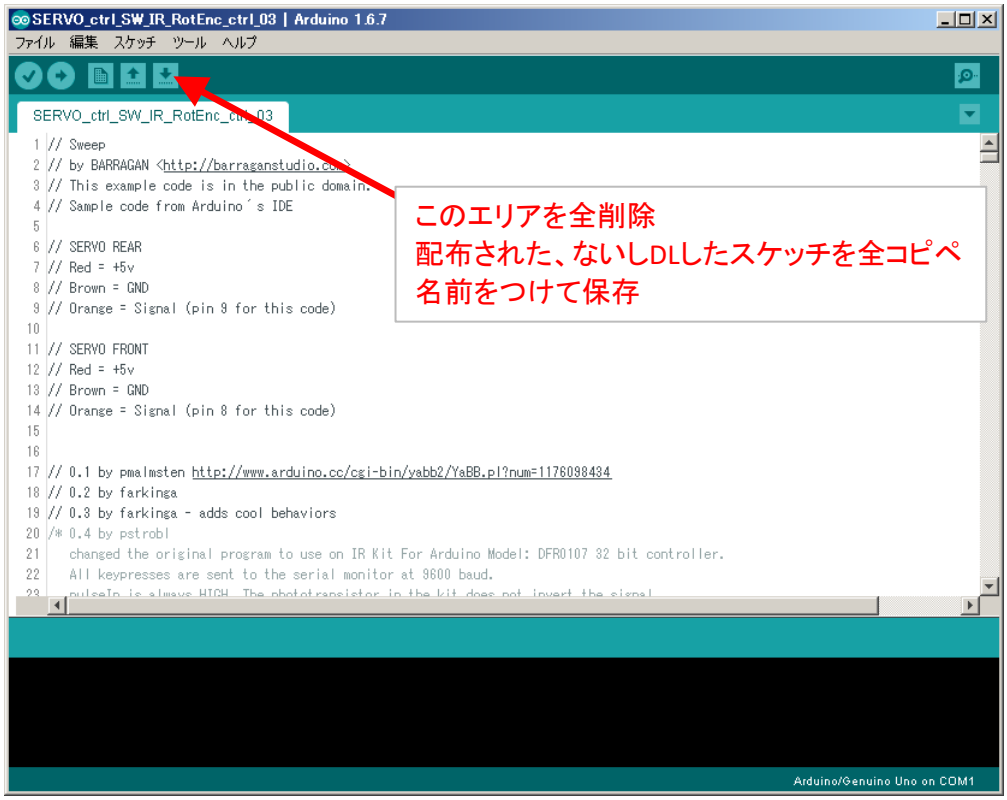
③−2　nano v3（モドキ）　の場合

https://www.arduino.cc/en/Main/ArduinoBoardNano
http://www.dorapro.co.jp/engineerblog/?p=720
CH340ドライバをDLしデスクトップに展開しておく。
MCUボードをPCにUSB接続する。
上記ドライバでセッティングする。
″ボードマネージャ″でMCU種を指定し、設定する
Select ″Arduino Diecimila, Duemilanove, or Nano w/ ATmega168″ or
″Arduino Duemilanove or Nano w/ ATmega328″
from the Tools > Board menu (according to the microcontroller on your board).
注：MCU設定は5V16MHzを選ぶこと。

④system



⑤スケッチの準備・編集



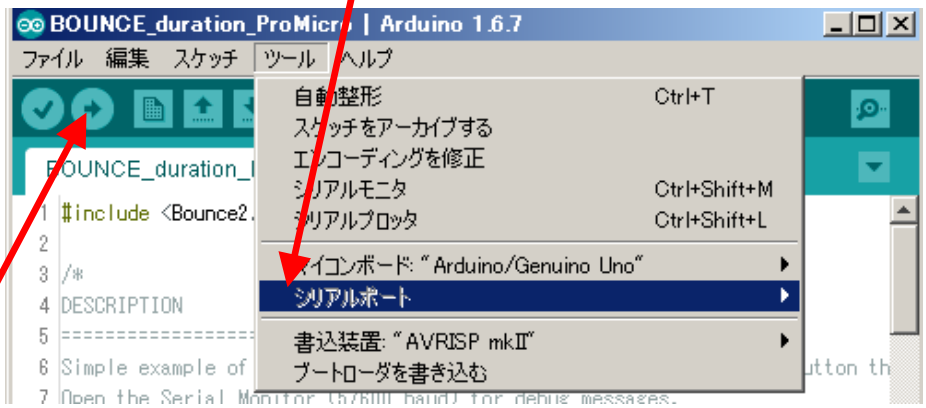
⑥スケッチのMCUへの書き込み

※スケッチの書き込みはMCUを単体にしてPCと接続した方が安定して動作します。

⑥-1　初期設定の書き込み（書き込み履歴のないMCUで最初の１回目だけ必要な作業。）

xxxx.inoを開く

comポートが開かれていることを確認する。



初期設定パラメータ確認後、スケッチの書き込みを実施する。

（EEPROMに初期設定が書き込まれます）

⑥-2　メインスケッチの書き込み

xxxx.inoを開く

上記同様、comポートが開かれていることを確認する。

設定パラメータ確認後、スケッチの書き込みを実施する。

※シフトセッティング等をチューニングするときは、都度パラメタを更新したスケッチを書き込んでください。

⑦メインスケッチパラメータ説明

スケッチ中に下記パラメタ群があります。セッティング後、MCUに書き込んで反映させる。

pin assignment

```
//////////////////////////////// MCU PIN ASSIGN START ////////////////////////////////////////
#define BUTTON_PIN 2  //REAR
#define BUTTON_PIN2 3 //REAR
#define BUTTON_PIN_f1 4  //FRONT
#define BUTTON_PIN_f2 5  //FRONT
#define SERVO_PIN_F 8  // SERVO F
#define SERVO_PIN_R 9  // SERVO R

//#define LED_PIN_MODSEL 16         // LED goes on when Fine Tun
//#define LED_PIN2 14         // LED goes on when pushing shift b
//#define BUTTON_PIN_MODESEL 10  //MODE SELECT
#define LED_PIN_MODSEL 15         // LED goes on when Fine Tune S
#define LED_PIN2 13         // LED goes on when pushing shift bot
#define BUTTON_PIN_MODESEL 14  //MODE SELECT (NANO)

////////////////////////////////  MCU PIN ASSIGN END ////////////////////////////////////////
```

MCUと周辺回路との
接続を変える場合は
ピンアサインのパラメタをいじる。

RearSetting(T.B.D)

```
//////////////////////////////// REAR SHIFT SETTING PARAMETER START
byte speed_step = 10;              //8=8速 9=10速 10=11速
int pos_zero = 1930;              // shift 0 position : about 2000~1800
byte pos_shiftstep = 77;
int pos_data_ORG[11] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}; // 11spe
byte fine_tune_pitch = 1;         //fine tune pitch data for setting mode

byte pos_shiftover_up = 35;
byte pos_shiftover_down = 30; // shi
int ovsft_delay = 500;          // over shift
byte sft_delay = 20;             // shift delay per

int wait_msec1 =  500;     //多段変則設定時
int wait_msec2 = 1000;     //多段変則設定時
int wait_msec3 = 1500;     //多段変則設定時
```

pos_zero = 1st位置の決定
pos_shiftstep = 1段ごとのピッチ
(数値はdegではないです)

pos_data= 各ギヤごとのfine tune data
上記パラメタで追い込みきれないときに使用

FrontSetting(T.B.D)

```
//////////////////////////////// FRONT SHIFT SETTING PARAMETER START  ////////////////////////////////////////

int pos_zero_f = 80;          // shift 0 position 180deg ->1st gear direction, 0deg ->10th gear direction,
int pos_inner = 70;       //80->70 @150809
int pos_outer = 150;      //120->120 @150809
int pos_shiftstep_f = 5;       // shift step degree
int pos_shiftover_up_f = 30;   // shift over degree 15->20 @150809
int pos_shiftover_down_f = 30; // shift over degree 15->40 @150809
int ovsft_delay_f = 2000;       // over shift delay [msec]
int sft_delay_f = 20;          // shift delay per gear pos [msec]

//////////////////////////////// FONT SHIFT SETTING PARAMETER END ////////////////////////////////////////
```

⑧動作確認

メカ操作中にUSBを接続し、シリアルポート開通を確認ののち、シリアルモニタを開くと、
メカ操作時ごとの動作パラメータが表示されます。デバッグ用にどうぞ。
（デバッグ時はシステムから切り離す必要はないが、書き込みは単体でやった方が安定動作しま