Tutorial 2A

システム創造プロジェクト TA: 田所祐一

2018 年 10 月 9 日（火）

1 はじめに

Tutorial 2A では，システム創造プロジェクトで使用するマイコンの開発環境に慣れることを主な目的とする．ここでは，マイコンボードは Arduino UNO を使用することを前提としている．システム創造プロジェクトの授業内では使用するマイコンの指定はされていないが，上記以外のマイコン及びマイコンボードを使用する場合は TA 側でサポートすることが困難であるため，他のマイコンを使用する際は自己責任で使用するようにしてもらいたい．

今回の Tutorial では，開発環境の導入と簡単な入出力のテスト，DC モータと RC サーボモータの駆動を行う．実際に取り組む内容としては以下のようになっている．

- 開発環境の導入
- `setup` と `loop` の理解，シリアル通信
- ピン入出力 (LED の点滅・スイッチの検出)
- PWM 出力 (DC モータ・RC サーボモータの駆動)

まず，開発環境のインストールを行い，サンプルプログラムを実行して動作を確認する．つぎに，ピンの入出力に使用する関数について説明し，デジタル値の入出力と，擬似的なアナログ値 (PWM) の出力を行う．最後に，Tutorial 1 で製作した機体を使って，DC モータと RC サーボモータの駆動を行う．

注意	メンバー全員が内容を理解できるように確認し合って進めること

注意	Tutorial のプログラムや資料は必要に応じてシステム創造プロジェクトのウェブページ (`http://www.cyb.sc.e.titech.ac.jp/cspweb2018/FrontPage.html`) からダウンロードすること

2 開発環境

本年度のシステム創造プロジェクトでは，開発環境として **Arduino IDE** を使用する．Arduino IDE は無償でダウンロードできる．Windows 10 では，「ストアアプリ」からインストールすることができる．その他の OS やバージョンを使用している場合は，各自方法を調べてもらいたい．

2.1 Arduino IDE の設定

今回使用するマイコンボード **Arduino UNO** を使用できるように，IDE の設定を行う．

1. Arduino IDE を起動し，USB ケーブルで Arduino をコンピュータに接続する
2. 「ツール」→「シリアルポート」から，Arduino が接続されている COM ポートを選択する
3. 「ツール」→「ボード」から，「Arduino/Genuino UNO」を選択する

注意	COM ポートの番号は使用するコンピュータによって異なる．また，別の USB ポートに接続した際に変わる場合もあるので，設定をその都度確認すること．

2.2 Arduino の入出力ピン

ピンの配置と各種機能との対応をまとめた図を Fig. 1 に示す．Arduino UNO には，図中の水色で示された 20 本の入出力ピンが用意されている．このうち，ピン 0 と 1 はコンピュータと Arduino とのシリアル通信に使用されているため，入出力ピンとしては使用しない．残りのピン 2–19 は，デジタル入出力を行うピンとして使用できる．また，ピン 14–19 (A0–A5) はアナログ電圧の入力ピンとしても使用できる．

Arduino Uno R3 Pinout

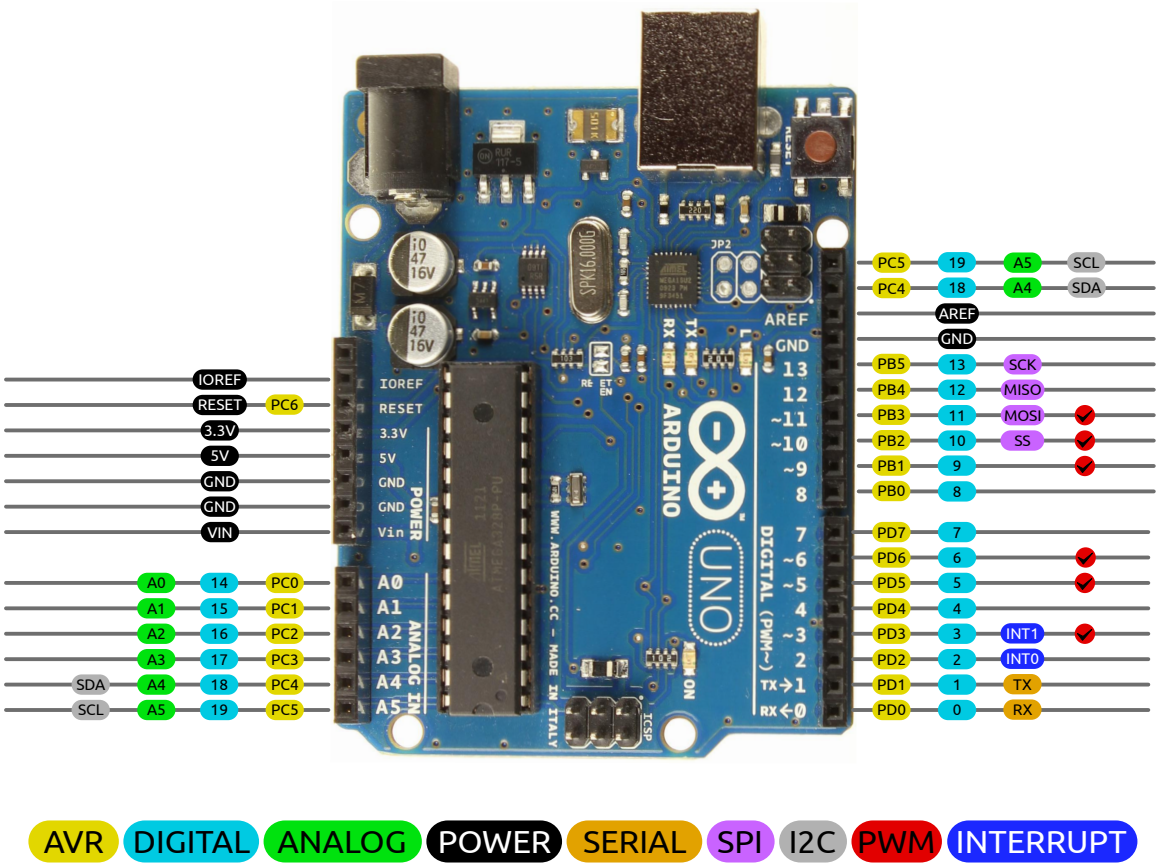

Fig. 1: Arduino UNO のピン配置 (出典: `https://github.com/Bouni/Arduino-Pinout`)

⚠ 注意	Arduino の詳しい機能や使用できる関数については，Arduino の公式マニュアル (`https://www.arduino.cc/reference/en/language/functions/communication/serial/`) を参照すること.

3 HelloWorld の表示

本節では，Arduino のプログラム (スケッチと呼ぶ) の構造を理解し，シリアル通信を使用して動作の確認を行う．マイコンの動作中に，内部状態やセンサの計測値などを確認することは，通常難しい．シリアル通信を使用すれば，コンピュータに情報を送信して確認できるため，不具合箇所や状況の特定が容易になることが期待できる．まずは基本的な使用方法を確認することからはじめる．

3.1 Arduino スケッチの基本構造

Arduino IDE を起動すると，以下のようなプログラムが書かれたウィンドウが表示される．

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

上記のスケッチは，そのまま実行しても何もしないプログラムになっている．Arduino では，上記の setup と loop という 2 つの関数を骨組みにして，プログラムを作成する．

- setup()
  電源投入・リセット時に一度だけ実行される関数．変数やピンの初期設定を行う処理を記述する．
- loop()
  繰り返し実行される関数．入出力の制御を行う処理を記述する．

スケッチの基本的な文法は C 言語と同じである．ただし，Arduino スケッチには，通常の C 言語のプログラムのように main 関数は存在しない[*1]．また，通常のマイコンを使用するときのように，ビット演算を使用してレジスタを操作する必要もない．これらは，Arduino IDE が基本的な動作をシンプルに記述できるようにするライブラリを備えているためであり，通常のマイコンやマイコンボードとは異なる点である．

3.2 HelloWorld の作成と実行

Arduino のシリアル通信機能 (Serial) を使用して，コンピュータ上にメッセージを表示するスケッチを作成する．プログラムを以下のように変更する．

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(1000000);
  Serial.println("Hello, World!");
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

[*1] 実際には，setup を実行して loop を繰り返し実行するような main 関数がすでに用意されていて，我々は上記の 2 つの関数だけを実装すれば良いようになっている．

- `Serial.begin(baud)`
  シリアル通信機能を初期化する関数．`baud` にボーレート (通信速度) を bps (ビット/秒) 単位で指定する．今回は 1Mbps(1,000,000bps) に設定している．
- `Serial.println(str)`
  文字列 + 改行を送信する関数．`str` に文字列を指定する．

画面左上の「→」マークのボタンをクリックすると，スケッチがコンパイルされ，Arduino に書き込まれる．エラーが発生した場合は，2.1 節の設定ができているかどうかや，プログラムが正しく書かれているかどうかを再度確認せよ．

スケッチを書き込むと，Arduino がリセットされ，自動的にスケッチが実行される．今回作成したスケッチは，`setup` 関数の中にメッセージを表示するプログラムを記述したため，リセット時に一度だけメッセージが表示されることになる．

「ツール」→「シリアルモニタ」から Arduino IDE のシリアルモニタを開いて，実際にメッセージが表示されることを確認する．通信速度を合わせるために，シリアルモニタの下部に「***** bps」と書かれている項目を，「1000000 bps」に変更する．Arduino 本体についているリセットボタンを押すと，プログラムが再度実行され，”Hello, World!”と一度だけ表示されることが確認できる．

3.3 数値を表示する

Arduino には，通常の C 言語の `printf` 関数に相当する機能がない．したがって，数値を表示する際には，前項の `Serial.println` 関数や，つぎに述べる `Serial.print` 関数を組み合わせて表示する必要がある．

- `Serial.print(str)`
  文字列を送信する関数．`Serial.println` とは異なり，改行をしない．

これを使用して，経過したおよその秒数を表示する以下のようなプログラムを作成する．スケッチを実行すると，およそ 1 秒ごとに数値がカウントアップして表示されることが確認できる．

```
unsigned int count = 0;

void setup() {
  Serial.begin(1000000);
}

void loop() {
  Serial.print("Count: ");
  Serial.print(count);
  Serial.println("");
  count++;
  delay(1000);
}
```

- `unsigned int`
  符号なし整数型．Arduino UNO では 16bit であり，0〜65535 までの数値を表現できる．
- `delay(ms)`
  ディレイ (時間待ち) 関数．`ms` には，待つ時間をミリ秒 (1/1000 秒) 単位で指定する．

`print` と `println` に分ける代わりに，`Serial.println(count);` としてもよい．`Serial.println("")` で改行を行うのは便利なテクニックなので，覚えておくとよい．

`delay` 関数は待つだけの処理であるため，`loop` 関数の処理にかかる厳密な時間は，`Serial.print[ln]` と `count++` の処理時間 + 約 1 秒となる．`loop` 関数の処理時間をより正確にするためのプログラムは，次回の Tutorial で扱う．

4 LED を点滅させる

本節では LED を点滅させることを通して，Arduino でのデジタル出力の方法を確認する．Arduino にはサンプルスケッチがたくさん用意されているため，それを利用しながらプログラムを書き進めると効率が良い．ファイル→スケッチ例→ 01.Basics → Blink を開くと，つぎのようなプログラムが表示される (コメント略).

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

このスケッチをそのまま実行すると，Arduino のメインボードに搭載された橙色の LED が点滅する．

- `pinMode(pin, mode)`
  ピンのモードを設定する関数．`pin` にはピン番号を，`mode` には `INPUT`(入力), `OUTPUT`(出力), `INPUT_PULLUP`(プルアップつきの入力) のいずれかを指定する．
- `digitalWrite(pin, output)`
  ピンのデジタル出力を行う関数．`pin` にはピン番号を，`output` には `HIGH` または `LOW` を指定する．
- `LED_BUILTIN`
  Arduino の基板上の LED に対応するピン番号の定数 (Arduino UNO では 13).

デジタル入出力ピンは，使用する前に `pinMode` 関数でモードを設定しなければならないことに注意する．ピンの電圧の状態を変えるには，`digitalWrite` 関数を使用する．`HIGH` を指定するとおよそ 5V，`LOW` を指定するとおよそ 0V の電圧にセットされる．

5 スイッチ入力を処理する

5.1 サンプルスケッチの利用

シールド基板に搭載したスイッチを使用して，物理的な入力を受け付けるようにする．ファイル→スケッチ例→02.Digital → Button を開くと，つぎのようなプログラムが表示される (コメント略).

```
const int buttonPin = 2;
const int ledPin =  13;

int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

- `digitalRead(pin)`
  ピンのデジタル入力を行う関数．`pin` にピン番号を指定する．ピンの電圧が高いときには HIGH を，低いときには LOW を返す.

前回の Tutorial では，シールド基板のピン 12 にスイッチを接続しているため，`buttonPin` の値を `2` から `12` に変更してから，スケッチを実行する．また，シールド基板にはプルアップ/ダウン抵抗を取り付けていないため，`pinMode(buttonPin, INPUT_PULLUP)` としてマイコン内蔵のプルアップ抵抗を使用する．

今回の基板では，スイッチを押していない状態でピンの電圧は高く，スイッチを押すとピンが GND と導通して電圧が低くなる．したがって，実行結果としては，スイッチを押すと Arduino 本体の LED が消灯し，スイッチを離すと LED が点灯する．

POINT	試技会のプログラムでは，電源ケーブルをスタートスイッチ代わりにするよりも，今回のプログラムを参考に，スイッチを使用して動作を開始させることを推奨する．

5.2 チャタリングへの対処

目には見えないが，スイッチは操作すると機械的に振動するため，電気的には HIGH と LOW の状態が頻繁に切り替わる状態になっている (これを**チャタリング**と呼ぶ)．そのため，`digitalRead` の結果も頻繁に入れ替わり，ソフトウェア的には何度もスイッチを押したり離したりしたことになってしまう (スケッチ例→ 01.Basics → DigitalReadSerial を実行すると，この様子が確認できる).

チャタリングへの対処方法は数多くあるが，最も簡単なソフトウェア的な解決方法は，`digitalRead` を振動よりも遅い時間間隔 (約 50〜100ms 程度) で取得することである (スケッチ例→ 02.Digital → Debounce に実装例がある)．また，抵抗とコンデンサを使用してローパスフィルタを構成することで，回路的にチャタリングを抑制することもできる．

6 モータを駆動する

6.1 analogWrite を使用した DC モータの駆動

Table 1: 入力信号と回転状態の対応

IN1 (左: ピン 5, 右: ピン 9)	**IN2** (左: ピン 6, 右: ピン 10)	状態
HIGH	LOW	正転
LOW	HIGH	逆転
LOW	LOW	フリー
HIGH	HIGH	ブレーキ

一般にマイコンのピンは大きな電流を流せないため，モータを駆動するためにはモータドライバと呼ばれる電子回路を使用する．この授業では，Texas Instruments 社製の DRV8835 という IC を使用する．DRV8835 には H ブリッジ回路が内蔵されており，2 本の PWM 信号を入力として，モータの正転・逆転・フリー・ブレーキの 4 つの状態を制御できる．信号の電圧と回転の状態の対応表を Table 1 に示す．

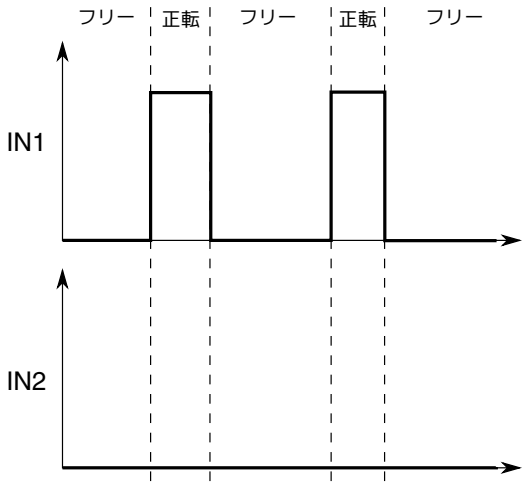

Fig. 2: PWM 使用例 1

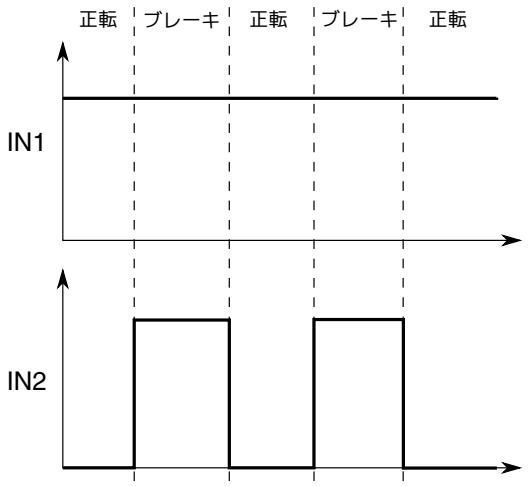

Fig. 3: PWM 使用例 2

Fig. 2 では，IN2 を LOW(PWM のパルス幅 0) にした状態で IN1 に PWM 信号を入力し，フリーと正転の状態を高速に切り替えることによって，モータの正転速度を調節している．IN1 と IN2 の信号を入れ替えることで，逆転速度も同様に調節することができる．また，Fig. 3 のように片方の信号を HIGH(PWM のパルス幅最大) にして，もう一方に PWM 信号を入力することで，ブレーキをかけながら正転・逆転させることができる．基本的には Fig. 2 の駆動方式がシンプルでわかりやすいが，急激な減速を行う必要がある場合は Fig. 3 の方式が便利な場合がある．

Arduino で PWM 信号を出力するには，`analogWrite` 関数を利用する．つぎのようなスケッチを実行する．

スケッチを実行する前に，配線が正しいか再確認すること．

```
#define MOTORL_IN1 5
#define MOTORL_IN2 6
#define MOTORR_IN1 9
#define MOTORR_IN2 10

void setMotorPulse(int left, int right) {
  if(left > 0) {
    analogWrite(MOTORL_IN1, min(left, 255)); analogWrite(MOTORL_IN2, 0);
  } else {
    analogWrite(MOTORL_IN1, 0); analogWrite(MOTORL_IN2, min(-left, 255));
  }
  if(right > 0) {
    analogWrite(MOTORR_IN1, min(right, 255)); analogWrite(MOTORR_IN2, 0);
  } else {
    analogWrite(MOTORR_IN1, 0); analogWrite(MOTORR_IN2, min(-right, 255));
  }
}

void setup() {
  pinMode(MOTORL_IN1, OUTPUT); pinMode(MOTORL_IN2, OUTPUT);
  pinMode(MOTORR_IN1, OUTPUT); pinMode(MOTORR_IN2, OUTPUT);
}

int pulse = 0;
int step = 1;

void loop() {
  if(pulse > 200) { step = -1; }
  else if(pulse < -200) { step = 1; }

  pulse += step;
  setMotorPulse(pulse, pulse);
  delay(20);
}
```

- `analogWrite(pin, output)`
  ピンの PWM 出力を行う関数．`pin` にはピン番号を，`output` にはパルス幅に対応する 0–255 の値を指定する．

Arduino スケッチでは，通常の C 言語と同様に定数や関数の定義が可能である．1–4 行目では，左右のモータを駆動する IN1, IN2 信号に対応するピン番号を定義している．また，`setMotorPulse` 関数を定義して，左右のモータの駆動に関する処理をまとめている．`analogWrite` に指定する指令値の最大値は 255 であるため，`setMotorPulse` 関数内で値を制限している．

このスケッチを実行すると，左右のモータが正転→逆転→正転→…を繰り返す．実際に地面に置いて走行できることを確認せよ．機体は前後に移動する．

 注意	機体が前後に移動せずに，回転したりモータが回らなかったりする場合は，前回の Tutorial 資料を参照して配線を確認すること．

機体がまっすぐ走行しないことが観察できるだろう．次回の Tutorial では，線に沿って機体をまっすぐ走行させるために，地面に引かれた線をフォトリフレクタで読み取り，簡単な制御を加える．

6.2 ServoTimer2 ライブラリを使用した RC サーボモータの駆動

システム創造プロジェクトでは，マニピュレーション用にラジコン用サーボモータ（RC サーボモータ）が用意されている．RC サーボモータは，小型 DC モータとポテンショメータ（可変抵抗器），ギヤ，制御回路を小さなユニットにまとめたもので，入力信号のパルス幅に対応した角度を保つように内部でフィードバック制御を行なっている．

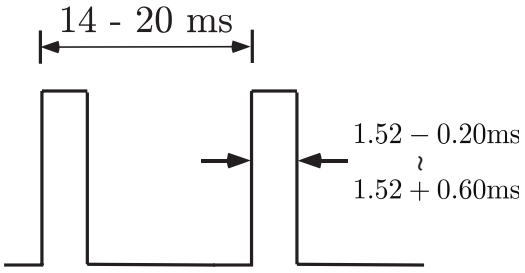

Fig. 4:　RC サーボモータの入力信号

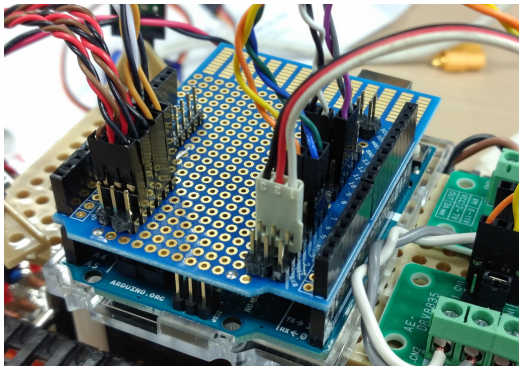

Fig. 5:　RC サーボモータの接続例

Fig. 4 のように制御パルスの周期は 14–20 [ms] で，パルス幅が 1.52 [ms] のときにニュートラル (中立) 位置になり，1.52 + 0.60 [ms] のときにおよそ +60° となり，1.52 – 0.20 [ms] のときにおよそ –20° となる．なお RC サーボモータのハードウェア的な中立位置とパルス幅を 1.52 [ms] 与えたときの位置は異なるので注意せよ．

サーボモータの内部では駆動軸にポテンショメータが取り付けられており，ポテンショメータの角度に対応した幅のパルスが発生するようになっている．入力されたパルス幅とサーボモータ内部のパルス幅が異なっている場合，RC サーボモータ内部のモータが動作し，二つのパルス幅が等しくなるまで駆動軸が回転する．この様な方式にすることで，サーボモータの駆動軸角度が外力によって変化した場合でもサーボモータの駆動力の範囲内で元の角度に戻ろうとする力を発生させることができる．

今回配布する RC サーボモータは双葉電子工業製の S3003 であり，仕様は Table 2 のとおりである．

Table 2:　S3003 の仕様

寸法	40.4 × 19.8 × 36 [mm]
重量	37.2 [g]
動作電源	4.8-6.0 [V]
動作スピード	0.23 [sec/60°]（4.8 [V] 時），0.19 [sec/60°]（6.0 [V] 時）
出力トルク	3.2 [kg·cm]（4.8 [V] 時），4.1 [kg·cm]（6.0 [V] 時）
動作角度	−90 – +90 [deg] 程度

また，配線の割り当ては，1 ピン (白) : PWM(制御信号)，2 ピン (赤) : VCC，3 ピン (黒) : GND となっている．前回の Tutorial で製作したシールド基板のピンヘッダに直接挿すことができる．Arduino のピン 2 に RC サーボモータを接続したときの写真を Fig. 5 に示す．

Arduino の PWM 周波数はおよそ 490Hz と 980Hz に設定されているが，RC サーボモータの駆動信号には約 50Hz の PWM を使用する必要がある．ServoTimer2 ライブラリ[*2]を使用することで，RC サーボモータに対応した PWM 信号を簡単に出力できる[*3]．

配布資料の `Software` フォルダに，ServoTimer2 ライブラリを同梱している．Arduino IDE を開き，メニューから「スケッチ」→「ライブラリをインクルード」→「ZIP 形式のライブラリをインストール」をクリックし，`Software` フォルダの中の `ServoTimer2.zip` を選択する．

[*2] `https://github.com/ld21/ServoTimer2`

[*3] Arduino に搭載されている AVR マイコンのタイマレジスタを操作することでも実現できるが，この授業では取り扱わない．

注意	Arduino UNO で analogWrite を使用できるピンは 3, 5, 6, 9, 10, 11 ピンに限られているが，ServoTimer2 ライブラリではその他のピンも指定できる．

注意	ServoTimer2 ライブラリを使用するとピン 3, 11 で analogWrite が使用できなくなるため，注意すること．

以下に，RC サーボモータを指定した角度に回転させるサンプルスケッチを示す．

```
#include <ServoTimer2.h>

ServoTimer2 servo;

void setup() {
  servo.attach(2, 544, 2400);
}

void loop() {
  servo.write(2400);
  delay(1000);
  servo.write(544);
  delay(1000);
}
```

- ServoTimer2 servo;
  ServoTimer2 のインスタンス servo を生成する．
- ServoTimer2::attach(pin)
  ServoTimer2 で使用するピンを設定する．pin にピン番号を指定する．
- ServoTimer2::write(pulseWidth)
  PWM 信号のパルス幅を pulseWidth に設定する．

POINT	RC サーボモータを複数接続する場合は，以下のように ServoTimer2 のインスタンスを複数作成し，それぞれに対して attach 関数を用いてピン番号を指定する．

```
#include <ServoTimer2.h>

ServoTimer2 servo1;
ServoTimer2 servo2;

void setup() {
  servo1.attach(2, 544, 2400); // ピン 2 に 1 つめのサーボを
  servo2.attach(3, 544, 2400); // ピン 3 に 2 つめのサーボを attach
}

...
```

6.3　RC サーボモータの駆動電源に関する注意事項

今回の Tutorial では，RC サーボモータの電源を Arduino に搭載されている 3 端子レギュレータから供給しているが，このような構成は推奨しない．なぜなら，RC サーボモータは駆動開始時や負荷がかかるときに大きな電流が流れるため，Arduino に供給されている電源電圧が降下することが考えられるからである．このとき，つぎのような問題が発生する可能性が高い．

- センサに正常な電圧が供給できなくなるため，AD 変換値が信頼できない数値になる
- (最悪の場合) マイコンに十分な電圧が供給されなくなり，強制的にリセットされる

対策として，RC サーボモータのために 5V 電源を別途用意することを推奨する．Figs. 6, 7 のように，シールド基板に 3 端子レギュレータや DC/DC コンバータを増設すると良い．

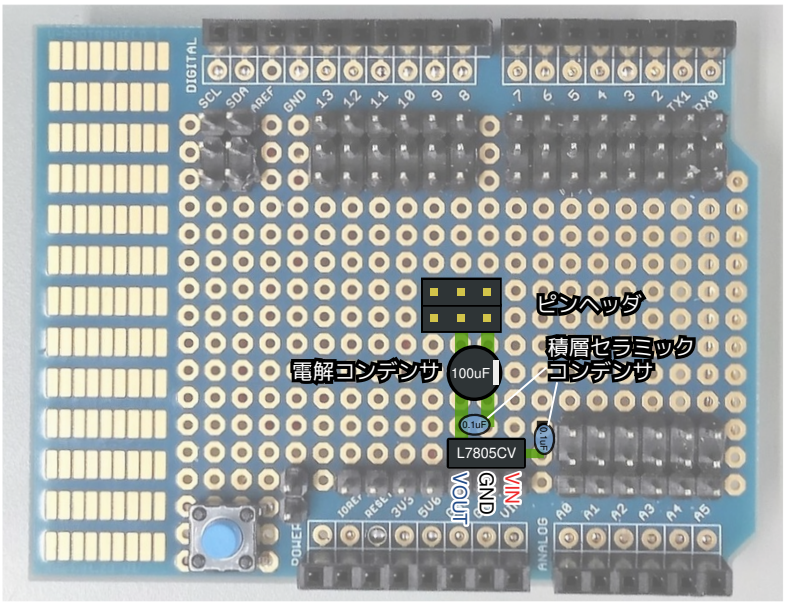

Fig. 6:　3 端子レギュレータの増設 (表面)

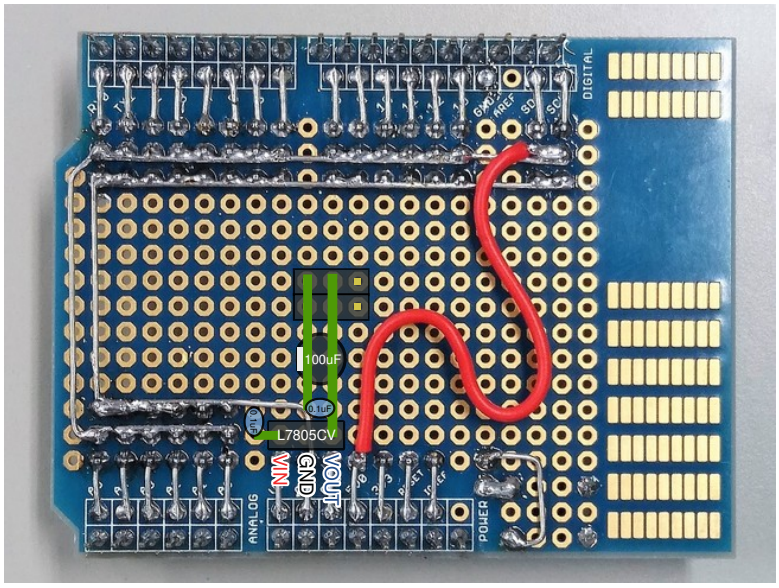

Fig. 7:　3 端子レギュレータの増設 (裏面)

部品リスト

- L7805CV 三端子レギュレータ (5V 1.5A 出力) : 1 個
- 積層セラミックコンデンサ (50V 0.1uF) : 2 個
- 電解コンデンサ (25V 100uF) : 1 個
- RC サーボモータ接続用ピンヘッダ 1×3 : 2 個

 注意	ここで紹介した部品は配布部品には含まれていない．必要に応じ，各自購入して増設してほしい．

 警告	大きな電流を連続的に流す必要がある場合は，3 端子レギュレータが発熱して高温になってしまう．その場合は，低発熱・高校率の DC/DC コンバータを代わりに使用すると良い[*4]．

 POINT	アクチュエータ駆動用の電源はロジック回路 (マイコン，通信等) やアナログ回路 (センサ等) の電源と別に用意することが望ましい．アクチュエータは駆動時に電源の電圧降下を引き起こしたり，大きなノイズを出力したりするためである．

[*4] 例: http://akizukidenshi.com/catalog/g/gK-09981/