

第 28 回全国産業教育フェア山口大会  
第 26 回全国高等学校ロボット競技大会山口大会  
自立型ロボット製作情報交換会

## 1. 競技規則の確認

### 実施規則【暫定版 1】から自立型ロボットに関する箇所の抜粋

---

#### 1 競技内容

##### (2) 競技概要

競技時間は 3 分間。

自立型ロボットは、錦帯橋ゾーンの橋を渡って移動し、壇ノ浦ゾーンの槍を相手側に押し込むことで、該当する地域エリアの得点を 2 倍にする。

#### 2 ロボットの規格及び製作規定

##### (1) 製作するロボット

自立型ロボット 1 台とする。リモコン型ロボットだけでも出場できるが、その場合、勝敗判定時のロボットの総重量は自立型ロボットの重量を 5kg として算出する。

##### (2) サイズ及び重量

###### ② 自立型ロボット

外寸：幅 300mm×奥行 300mm×高さ 300mm 以内

※外寸は展開時の最大寸法とする（競技中を含む）。

重量：5kg 以内

※重量とは、ロボットの本体、動力源等のロボット構成部品の総重量を示す。

##### (4) 制御方法・機構

⑤ 自立型ロボットの制御方式は、競技者の起動操作により起動し、全ての動作を完全に自立して行うこととする。無線等による外部からの制御は一切禁止する。

⑥ リモコン型ロボット及び自立型ロボットにおいて、各ロボットの分離と子機の使用は認めない。

#### 3 橋の製作規定及び設置方法

##### (1) 製作する橋

競技開始前に錦帯橋ゾーンに架けておく「橋 1」を 2 つ、競技中に橋置場からリモコン型ロボットが搬送して錦帯橋ゾーンに架ける「橋 2」を自作する。「橋 2」の個数は問わない。これらの橋を錦帯橋ゾーンに架けることで、橋の上を自立型ロボットが移動して壇ノ浦ゾーンの槍を押し込むことができるようになる。

##### (2) 製作規定

###### ① 橋 1

各チームで自作したものを 2 つ用意しておく。

###### ② 橋 2

各チームで自作したものを用意しておく。競技開始前に、橋置場内に完全に（空中を含む）収まり、橋置場

の床面から488mm の高さ（コートAとコートBとの仕切り板の高さ）を越えないように配置できる寸法とする。競技開始後の展開は自由とし、材質、重量、色等は問わない。

#### 4 競技コートの仕様

##### (2) 各ゾーン・エリア（図1参照）

###### ②移動エリアⅠ

フラットな床面で、リモコン型ロボットが自由に移動できる。自立型ロボットは移動エリアⅠの床面に触れてはならない。

###### ④スタートエリアⅡ

自立型ロボットのスタートエリアである（705.5mm×544mm、床面の高さ300mm）。自立型ロボットは、このエリアにおいてメンテナンス、センサやプログラムの調整をすることができる。ただし、熱源（はんだごて）や動力源を要する工具の使用を禁止する。パソコン等を用いたプログラムの書き換えも禁止する。

###### ⑥錦帯橋ゾーン

競技者が自作した橋を架けるゾーンで、幅300mm の橋脚が511mm 間隔で6脚（移動エリアⅡ側から「橋脚①」、「橋脚②」、「橋脚③」、…、「橋脚⑥」と呼ぶ）設置してある。このゾーンに橋を架けることで、この橋の上を自立型ロボットが移動できるようになる。

競技開始前、橋脚①と橋脚②の間及び橋脚③と橋脚④の間に、それぞれ自作した「橋1」を架けておく。ただし、隣接する橋脚間の長さ（89mm+511mm+89mm）を越えたり、壇ノ浦ゾーンや移動エリアⅡに侵入（上空を含む）したり、移動エリアⅠの床面に触れて架けることはできない。

なお、競技開始以降、リモコン型ロボットが、橋置場に配置した「橋2」を搬送してこのゾーンに架ければ、自立型ロボットの移動範囲を広げることができる。

###### ⑦壇ノ浦ゾーン

自立型ロボットが錦帯橋ゾーンの橋を渡り、槍を押し込むために侵入するゾーンである。各地域エリアに対応した槍を相手側に押し込むことで、その地域エリアの得点が2倍になる。競技開始前に「橋1」を設置する際、このゾーンに侵入（空中を含む）することはできない。また、自立型ロボット以外のものが、このゾーンの槍に触れてはならない。

#### 6 競技方法

##### (1) 競技形式

予選は、壇ノ浦ゾーンに設置される槍の長さを相手コートに影響がない328mm として、全チームが競技を行い、上位48 チームが決勝トーナメント戦に進出する。決勝トーナメント戦では、壇ノ浦ゾーンに配置される槍の長さを940mm として、対戦型で競技を行う。なお、勝敗は「8勝敗の判定基準」に基づく。

##### (2) 競技内容

###### ⑤競技開始

競技開始の合図（ブザー）以後、競技者は以下のことができる。

イ) 自立型ロボットをスタートさせることができる。

⑦自立型ロボットはスタートエリアⅡ、移動エリアⅡ、錦帯橋ゾーン及び壇ノ浦ゾーンを移動することがで

きる。自立型ロボットが錦帯橋ゾーンに架かった橋の上に一度でも完全に乗り（車輪や無限軌道等がすべて橋の上にある状態）、かつ、競技終了時点で自コート内（操縦エリアより内側）に自立型ロボットがあれば、「20 点」を加算する。また、壇ノ浦ゾーンにある各地域エリアに対応した（各地域エリアの正面に 1 個ずつ設置された）「槍」を押すことができる。競技終了時点で自立型ロボットがスタートエリアⅡに（上空を含めて）完全に戻っており、槍を相手側へ押し込んだ状態であれば、その槍の正面にある地域エリアの得点が 2 倍になる。なお、自立型ロボットは 1 回のスタートで複数の槍を押してよいこととする。

⑧自立型ロボット以外のもの（リモコン型ロボット、橋、各アイテム等）が壇ノ浦ゾーンの槍に触れてはならない。

#### ⑩競技終了

競技開始から 3 分経過すると競技終了となる。

自立型ロボットがスタートエリアⅡに（上空を含めて）完全に戻っており、槍を相手側へ押し込んだ状態であれば、その槍の正面にある地域エリアの得点が 2 倍になる。ただし、自立型ロボット以外のものが壇ノ浦ゾーンの槍に触れていると審判が判断した場合、その槍は、触れているチーム側に押し込まれたものとみなす。

自立型ロボットが競技中、錦帯橋ゾーンに架かった橋の上に一度でも完全に乗っており、この時点で自コート内（操縦エリアより内側）に自立型ロボットがあれば、「20 点」を加算する。

#### ⑪「Vゴール」（予選時のみ）

錦帯橋ゾーンの橋脚②と橋脚③の間及び橋脚④と橋脚⑤の間、橋脚⑤と橋脚⑥の間すべてに橋 2 が正しく架かっており、橋脚①と橋脚②の間及び橋脚③と橋脚④の間に橋 1 が架かっている状態で、自立型ロボットが壇ノ浦ゾーンの 5 本の槍を全て押し込み、リモコン型ロボットが全ての地域エリアに全てのアイテムを指定されたとおりに置いて、2 台のロボットが各スタートエリアに戻ったところで、競技者はコントロールボックスを床に置いて手を上げて「競技終了宣言」をすることができる。一度競技終了宣言を行うと、再スタートすることはできない。競技終了後、審判が満点を確認した場合に「Vゴール」達成とし、競技終了宣言をした時間を「Vゴール達成時間」とする。満点となっていない場合、通常の得点集計を行う。

対戦型競技として行う決勝トーナメント戦では、Vゴールは適用しない。

#### (3) 「中断」

次の条件になると、審判は「中断」を宣言する。

- ①競技中にスタートエリア以外で、競技者が審判の許可なくロボットやアイテムに触れた場合。
- ②競技者が競技コート内に、審判の許可なく侵入した場合。
- ④自立型ロボットがフライングスタートをした場合。
- ⑤自立型ロボットがスタートエリアⅡ・移動エリアⅡ・壇ノ浦ゾーン・錦帯橋ゾーン以外に出たと審判が判断した場合。
- ⑥自立型ロボットが故障、暴走等で競技に影響を及ぼすと審判が判断した場合。
- ⑦自立型ロボット以外のものが槍に触れたと審判が判断した場合。

(2) 槍を押し込んでいるか、押し込んでいないかの判断は次の写真による。

※コンパネから完全に白いラインが出ている状態を「押し込んでいる」と判断する。

①予選（槍の長さ328mm）



競技前の槍の位置



押し込んでいる場合



押し込んでいない場合（例）

②決勝トーナメント戦（槍の長さ940mm）



競技前の槍の位置



押し込んでいない場合



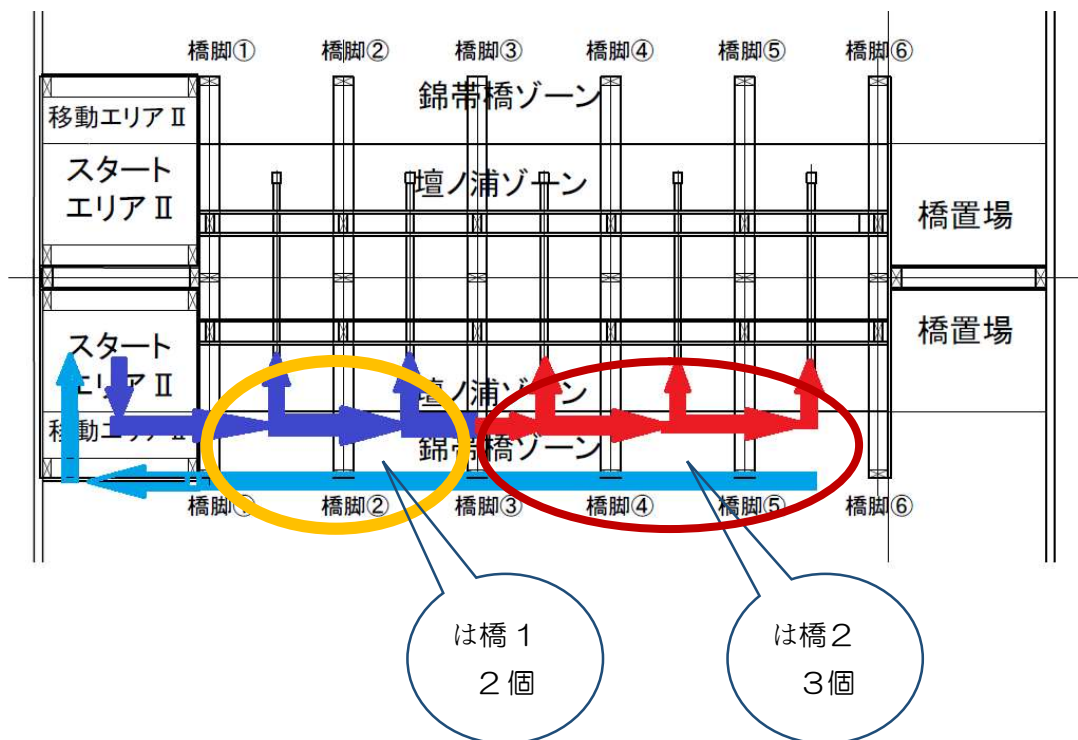
（例押し込んでいる場合）

## 2.どんなロボットを作れば良いか

左右モーター2を搭載し、前進、後進、右90度旋回、左90度旋回、180度旋回の各動作がしっかりできるロボットが良いのではないかと考えられる。センサーはライン読み取り用に4～8個（デジタル）、橋があるかないかの判断に1個（デジタル）が良いと思われる。ロボットの形状は相撲ロボットの様な箱型で、体当たりで槍を押し出す機構でよいと思われる。

ルール上気になるのは、「自立型ロボットは移動エリアⅠの床面に触れてはならない」の部分で、ロボットは床に落下してはならないこと判断できる。予選競技ではないが、決勝で槍の押し合いになった場合相手に押され（押し返され）床面に落ちてしまうことあるのではということである。コースを見て判断する必要がある。吸引装置搭載が必要となることも想定される。

## 走行軌跡と動作



## 動作手順

競技開始前、橋1を2個橋脚①～③に競技者がかける。

競技開始後、橋2をリモコンロボットにより橋脚③～⑥まで3個かける。

- (1) 自立型ロボットは競技開始後、槍2本を押し、橋脚③まで進む。
- (2) 橋2がかかったのを確認するまで待つ。
- (3) 橋2がかかったら、残りの槍を押しするために移動再開する。
- (4) すべての槍を押し終わったらスタートエリアに戻る。

### 3.Arduinoによる制御

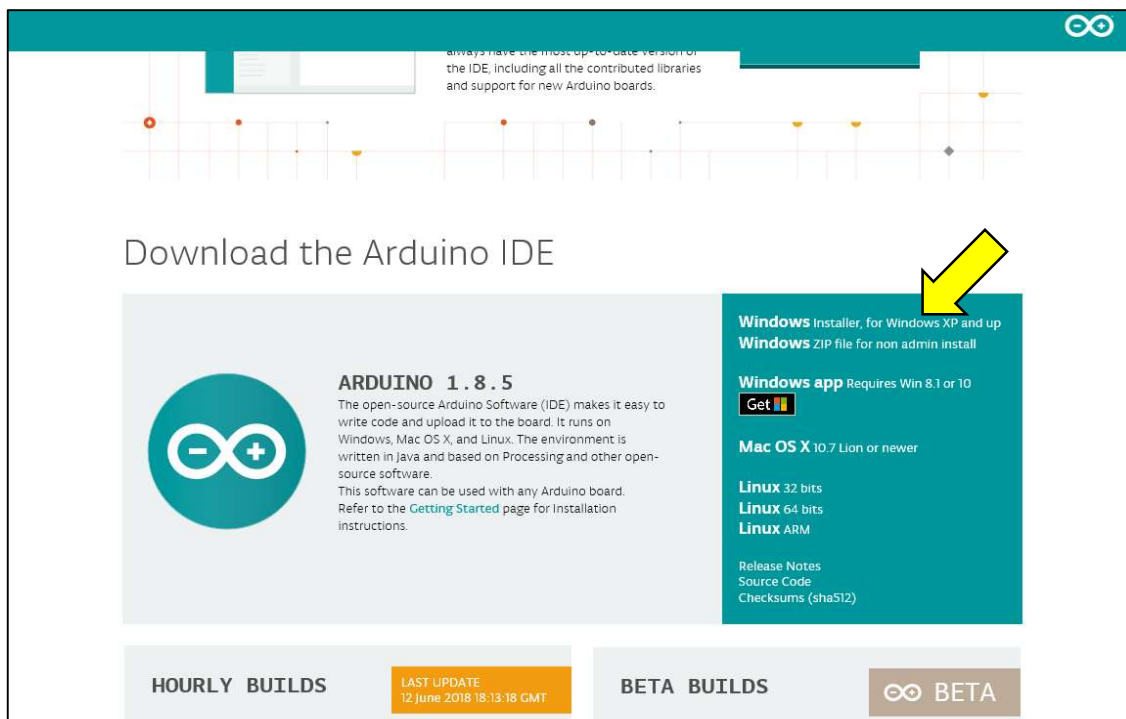
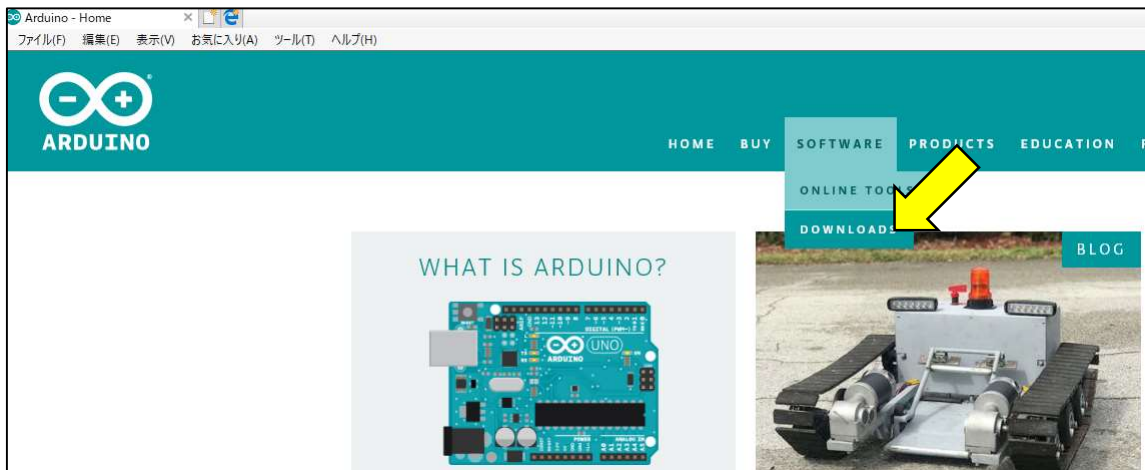
#### (1) 準備

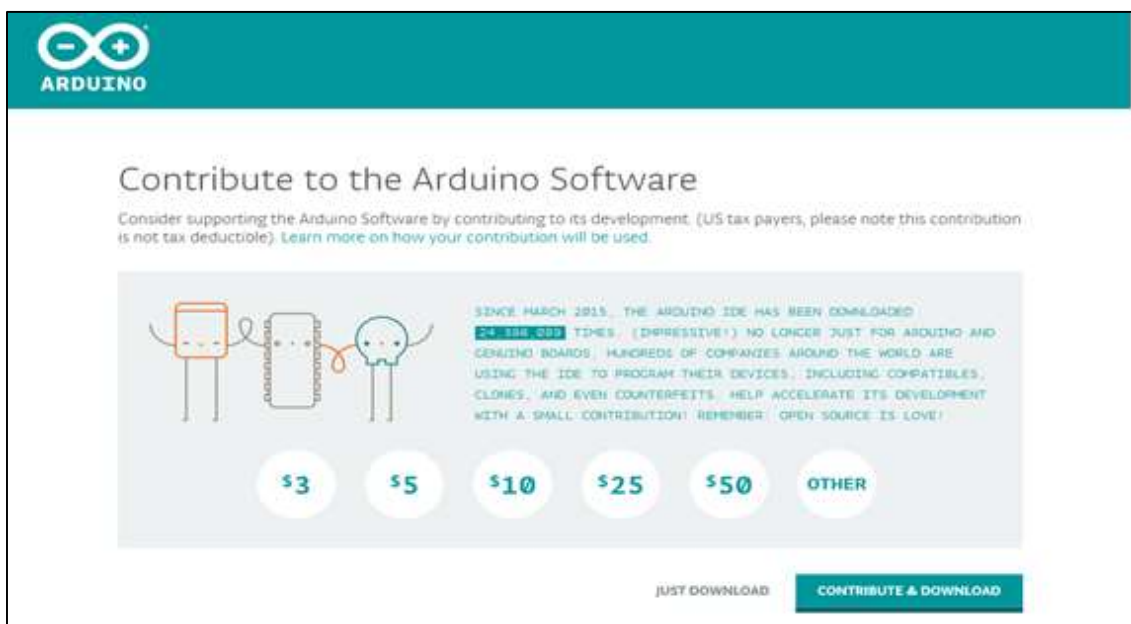
今回使用はArduino UNO(Genuino Uno)

購入先は秋月電子通商

#### (2) 開発環境

<http://www.arduino.cc/>からダウンロード





【JUST DOWNLOAD】をクリックし、ダウンロードする。  
その後ダウンロードされたファイルをクリックし展開する。



### (3) ボードの接続

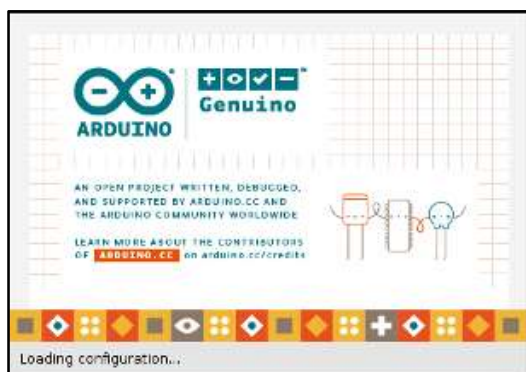
ArduinoをUSBケーブルでPCと接続する。ドライバーは自動識別される。  
Arduinoの電源はUSBより供給される。

(注 IEDのバージョンによっては認識されない場合があるようです。)

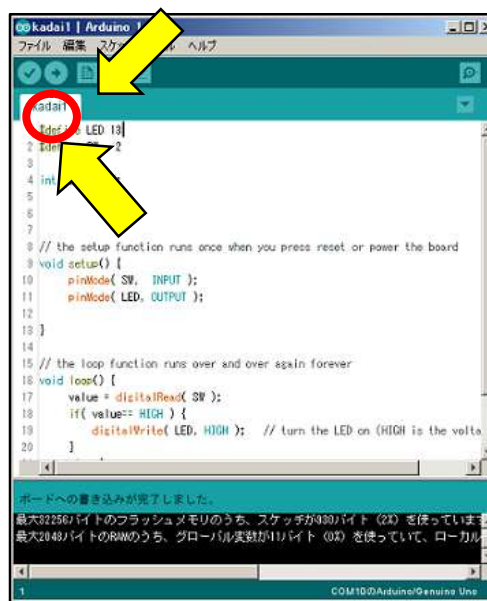


### (4) プログラム開発

デスクトップにできたアイコンをクリックするとシステムが立ち上がり、  
プログラム作成することができる。



システム起動時画面



プログラム開発後、コンパイル&書き込み

(5) 今回のロボットを作るうえで最小限覚えておきたいこと

基本的なロボットを作るのであれば基本入出力(「デジタル入力」「デジタル出力」)だけでも大丈夫です。まずは基本を理解して下さい。

課題1 スイッチ入力 LED出力 (デジタル入力 デジタル出力)

SWのON/OFFを入力し、ArduinoオンボードLEDを点灯/消灯させる。

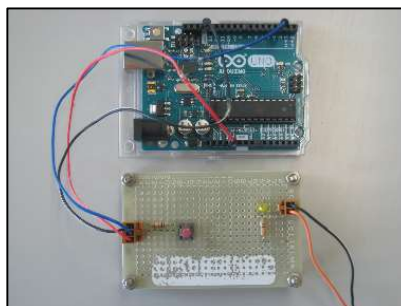
SWは2番ピンに接続

LEDはCPUに搭載されているもの(13番ピン)を使用

```
#define LED 13 // on board LED
#define SW 2 // SW
int indata =0;

void setup(){
    pinMode( LED, OUTPUT ); //LEDの接続されているピンは出力
    pinMode( SW, INPUT); //SWの接続されているピンは入力
}

void loop(){
    indata = digitalRead( SW ); //SWの状態入力
    if( indata==HIGH) { //もしSWがONであれば
        digitalWrite( LED, HIGH); // LED点灯
    } //
    else { //さもなければ(SWがOFFであれば
        digitalWrite( LED, LOW); // LED消灯
    }
}
```



上記課題だけでは他のロボットには流用できません。他に「PWM波出力(アナログ出力)」「シリアル通信(モニターへの表示)」「アナログ入力」「タイマー割り込み処理」などを確認します。本講習会用にいくつか実験してみました。掲載しますのでご参考下さい。ここまでくれば他のロボットCPUとして採用することができると思います。



課題2 PWM波出力（アナログ出力）

LEDの明るさを変更する。

Arduinoは3,5,6,9,10,11番の合計6ピンがPWM端子として使用できます。  
このプログラムはLEDを消灯から点灯まで、また点灯から消灯まで徐々に明るさを変化させていきます。

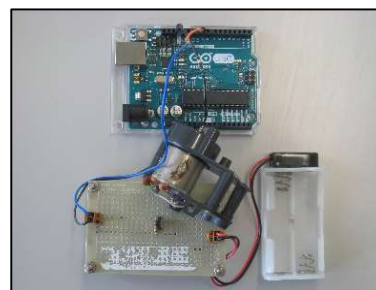
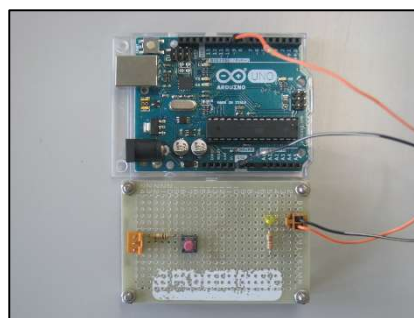
LEDをモーターに交換すると、モーターは低速から高速まで回転します。モーター速度制御も全くおなじプログラムです。

```
#define LED 9 // LED
int i = 0;

void setup(){
  pinMode( LED, OUTPUT );
  //LEDの接続されているピンは出力
}

void loop(){
  analogWrite( LED, 0 ); //LED消灯
  delay(500); //500ms待つ

  for( i=0;i<255;i++ ) { //消灯から点灯
    analogWrite( LED, i ); //LED点灯
    delay(50); //50ms待つ
  }
  for( i=255;i>0;i-- ) { //点灯から消灯
    analogWrite( LED, i ); //LED点灯
    delay(50); //50ms待つ
  }
}
```



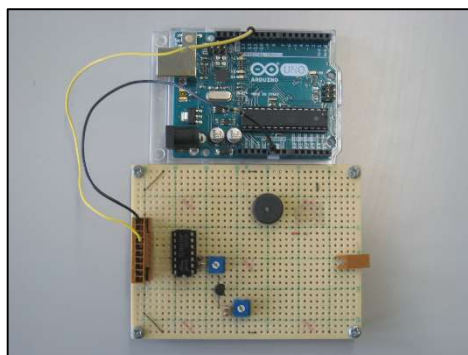
課題2-2 PWM波出力（アナログ出力）

圧電ブザーを鳴らす。

PWM出力ができると圧電ブザーから簡単に音を鳴らすことができます。

```
#define BEAT 300 // 音の長さを指定  
#define PINNO 12 // 圧電スピーカを接続したピン番号
```

```
void setup() {  
}  
void loop() {  
    tone(PINNO,262,BEAT); // ド  
    delay(BEAT);  
    tone(PINNO,294,BEAT); // レ  
    delay(BEAT);  
    tone(PINNO,330,BEAT); // ミ  
    delay(BEAT);  
    tone(PINNO,349,BEAT); // ファ  
    delay(BEAT);  
    tone(PINNO,392,BEAT); // ソ  
    delay(BEAT);  
    tone(PINNO,440,BEAT); // ラ  
    delay(BEAT);  
    tone(PINNO,494,BEAT); // シ  
    delay(BEAT);  
    tone(PINNO,523,BEAT); // ド  
    delay(3000); // 3秒後に繰り返す  
}
```



課題3 アナログ入力&SIO

アナログセンサを入力し電圧測定し値をPC画面に出力する。

初めて使うセンサーは、必ずこのプログラムを使いどのような値が出力されるのか確認します。その後どのように利用するか検討します。最初は可変抵抗を使いプログラムの動作とPCの表示を確認します。

// 電圧測定

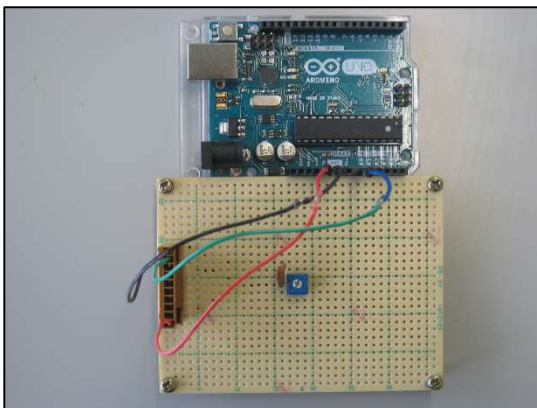
```
static int sensor = 0;
static int indata = 0;

double v1 = 0;
double L = 0;

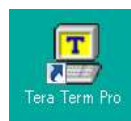
void setup() {
  Serial.begin(9600);
}

void loop() {
  indata = analogRead( sensor );
  v1 = 4.9 * indata / 1024.0;

  Serial.print( indata ); Serial.print( " ");
  Serial.print( v1 ); Serial.print( "[V]" );
  Serial.println();
  delay(1000);
}
```

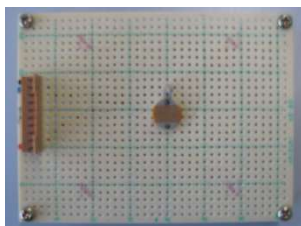


通信用ソフトを使って

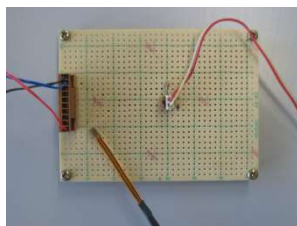


Tera Term - V		
File	Edit	Setup
1023	4.90 [V]	
957	4.58 [V]	
902	4.32 [V]	
872	4.17 [V]	
840	4.02 [V]	
807	3.88 [V]	
781	3.74 [V]	
744	3.56 [V]	
721	3.45 [V]	
693	3.32 [V]	
658	3.15 [V]	
618	2.96 [V]	
566	2.71 [V]	
529	2.53 [V]	
494	2.36 [V]	
429	2.05 [V]	
374	1.78 [V]	
313	1.50 [V]	
240	1.15 [V]	
12	0.06 [V]	
0	0.00 [V]	
n	n	n

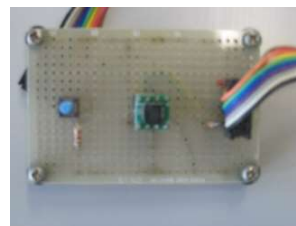
よく使う他のセンサー



Csd



感圧



加速度

距離センサーでの例

例えば自立型ロボットで良く使われている SHARP 距離センサーを採用する場合、まずセンサーがどのような値が出力されるのか課題 3 のプログラムを使い確認します。その値を見、利用方法を検討します。次の数値はセンサー前に箱を置き計測した値です。



10cm		11cm		12cm		13cm		14cm	
458	2.19 [V]	437	2.09 [V]	396	1.89 [V]	366	1.75 [V]	348	1.67 [V]
458	2.19 [V]	422	2.02 [V]	393	1.88 [V]	366	1.75 [V]	348	1.67 [V]
470	2.25 [V]	423	2.02 [V]	398	1.90 [V]	367	1.76 [V]	348	1.67 [V]
457	2.19 [V]	423	2.02 [V]	394	1.89 [V]	367	1.76 [V]	348	1.67 [V]
452	2.16 [V]	422	2.02 [V]	394	1.89 [V]	366	1.75 [V]	348	1.67 [V]
455	2.18 [V]	422	2.02 [V]	397	1.90 [V]	366	1.75 [V]	348	1.67 [V]
455	2.18 [V]	422	2.02 [V]	393	1.88 [V]	366	1.75 [V]	348	1.67 [V]
472	2.26 [V]	424	2.03 [V]	393	1.88 [V]	367	1.76 [V]	348	1.67 [V]
457	2.19 [V]	436	2.09 [V]	393	1.88 [V]	367	1.76 [V]	349	1.67 [V]
455	2.18 [V]	426	2.04 [V]	391	1.87 [V]	366	1.75 [V]	349	1.67 [V]
455	2.18 [V]	424	2.03 [V]	401	1.92 [V]	367	1.76 [V]	349	1.67 [V]
455	2.18 [V]	422	2.02 [V]	394	1.89 [V]	366	1.75 [V]	348	1.67 [V]
454	2.17 [V]	422	2.02 [V]	398	1.90 [V]	367	1.76 [V]	349	1.67 [V]
461	2.21 [V]	422	2.02 [V]	397	1.90 [V]	366	1.75 [V]	350	1.67 [V]
459	2.20 [V]	422	2.02 [V]	396	1.89 [V]	366	1.75 [V]	350	1.67 [V]
458	2.19 [V]	422	2.02 [V]	395	1.89 [V]	366	1.75 [V]	351	1.68 [V]
458	2.19 [V]	428	2.05 [V]	393	1.88 [V]	366	1.75 [V]	348	1.67 [V]
454	2.17 [V]	422	2.02 [V]	398	1.90 [V]	370	1.77 [V]	352	1.68 [V]
461	2.21 [V]	425	2.03 [V]	397	1.90 [V]	366	1.75 [V]	354	1.69 [V]
459	2.20 [V]	423	2.02 [V]	393	1.88 [V]	366	1.75 [V]	362	1.73 [V]
458	2.19 [V]	423	2.02 [V]	394	1.89 [V]	367	1.76 [V]	357	1.71 [V]
455	2.18 [V]	423	2.02 [V]	394	1.89 [V]	367	1.76 [V]	348	1.67 [V]
458	2.19 [V]	422	2.02 [V]	397	1.90 [V]	367	1.76 [V]	344	1.65 [V]

15cm		20cm		25cm		30cm		なし	
343	1.64 [V]	273	1.31 [V]	221	1.06 [V]	216	1.03 [V]	52	0.25 [V]
330	1.58 [V]	262	1.25 [V]	222	1.06 [V]	217	1.04 [V]	55	0.26 [V]
328	1.57 [V]	263	1.26 [V]	225	1.08 [V]	217	1.04 [V]	47	0.22 [V]
329	1.57 [V]	263	1.26 [V]	221	1.06 [V]	230	1.10 [V]	48	0.23 [V]
330	1.58 [V]	264	1.26 [V]	220	1.05 [V]	217	1.04 [V]	40	0.19 [V]
330	1.58 [V]	274	1.31 [V]	221	1.06 [V]	217	1.04 [V]	47	0.22 [V]
330	1.58 [V]	262	1.25 [V]	223	1.07 [V]	219	1.05 [V]	51	0.24 [V]
331	1.58 [V]	262	1.25 [V]	231	1.11 [V]	217	1.04 [V]	40	0.19 [V]
332	1.59 [V]	263	1.26 [V]	221	1.06 [V]	217	1.04 [V]	47	0.22 [V]
336	1.61 [V]	264	1.26 [V]	221	1.06 [V]	218	1.04 [V]	56	0.27 [V]
336	1.61 [V]	276	1.32 [V]	221	1.06 [V]	231	1.11 [V]	52	0.25 [V]
330	1.58 [V]	262	1.25 [V]	225	1.08 [V]	217	1.04 [V]	51	0.24 [V]
329	1.57 [V]	262	1.25 [V]	221	1.06 [V]	217	1.04 [V]	52	0.25 [V]
330	1.58 [V]	263	1.26 [V]	221	1.06 [V]	219	1.05 [V]	42	0.20 [V]
330	1.58 [V]	264	1.26 [V]	221	1.06 [V]	218	1.04 [V]	51	0.24 [V]
331	1.58 [V]	268	1.28 [V]	223	1.07 [V]	217	1.04 [V]	47	0.22 [V]
330	1.58 [V]	260	1.24 [V]	224	1.07 [V]	217	1.04 [V]	47	0.22 [V]
332	1.59 [V]	262	1.25 [V]	221	1.06 [V]	220	1.05 [V]	51	0.24 [V]
334	1.60 [V]	262	1.25 [V]	221	1.06 [V]	217	1.04 [V]	52	0.25 [V]
339	1.62 [V]	264	1.26 [V]	221	1.06 [V]	217	1.04 [V]	58	0.28 [V]
328	1.57 [V]	267	1.28 [V]	222	1.06 [V]	217	1.04 [V]	55	0.26 [V]
330	1.58 [V]	258	1.23 [V]	234	1.12 [V]	219	1.05 [V]	60	0.29 [V]
330	1.58 [V]	262	1.25 [V]	221	1.06 [V]	217	1.04 [V]	62	0.30 [V]
330	1.58 [V]	262	1.25 [V]	221	1.06 [V]	217	1.04 [V]		

### 課題3-2 PSDセンサーの例

センサーに付属している特性図を見ると、「距離と発生する電圧が直線になっておらず、正確な距離測定には向かないかな」とも認識されますが、測定結果見ると、「ロボット前方の障害物の有無」や「壁との距離を一定に保ち走行する」などに利用できるのではと判断できます。増幅回路の追加やプログラムの工夫を検討してみてください。

```
static int psd = 0;
static int indata = 0;
int distance = 0;
double v1 = 0;
double L = 0;

void setup() {
    Serial.begin(9600);
}

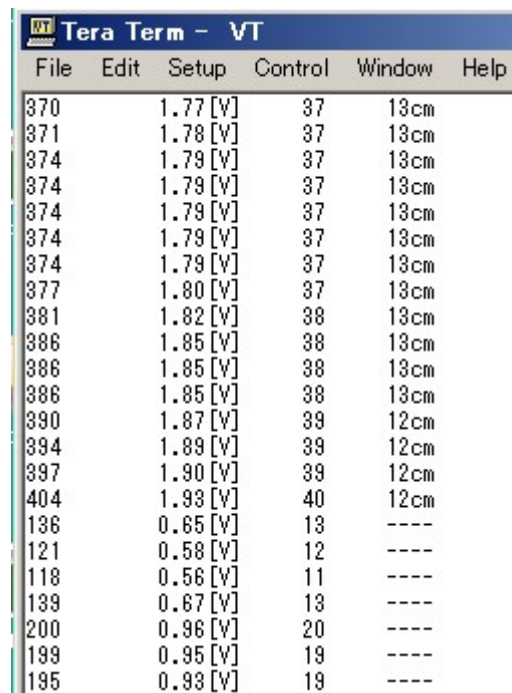
void loop() {
    indata = analogRead( psd );
    v1 = 4.9 * indata / 1024.0;
    distance = indata / 10.;

    Serial.print( indata ); Serial.print( " ");
    Serial.print( v1 ); Serial.print( "[V] ");
    Serial.print( distance ); Serial.print( " ");

    if ( distance > 44 ) {
        Serial.print( "10cm" );
    } else if( distance > 41 ) {
        Serial.print( "11cm" );
    } else if( distance > 38 ) {
        Serial.print( "12cm" );
    } else if( distance > 35 ) {
        Serial.print( "13cm" );
    } else if( distance > 33 ) {
        Serial.print( "14cm" );
    } else if( distance > 31 ) {
```



```
        Serial.print( "15cm" );  
    } else if( distance > 25 ) {  
        Serial.print( "20cm" );  
    } else if( distance > 21 ) {  
        Serial.print( "25cm" );  
    } else if( distance > 20 ) {  
        Serial.print( "30cm" );  
    } else {  
        Serial.print( "----" );  
    }  
    Serial.println();  
    delay(500);  
}
```



The screenshot shows a terminal window titled "Tera Term - VT" with a menu bar (File, Edit, Setup, Control, Window, Help). The terminal displays a table of data with four columns: a numerical ID, a voltage reading in Volts [V], a number, and a distance in centimeters (cm). The data points are as follows:

ID	Voltage [V]	Number	Distance (cm)
370	1.77	37	13cm
371	1.78	37	13cm
374	1.79	37	13cm
374	1.79	37	13cm
374	1.79	37	13cm
374	1.79	37	13cm
374	1.79	37	13cm
377	1.80	37	13cm
381	1.82	38	13cm
386	1.85	38	13cm
386	1.85	38	13cm
386	1.85	38	13cm
390	1.87	39	12cm
394	1.89	39	12cm
397	1.90	39	12cm
404	1.93	40	12cm
136	0.65	13	----
121	0.58	12	----
118	0.56	11	----
139	0.67	13	----
200	0.96	20	----
199	0.95	19	----
195	0.93	19	----

課題4 割り込み処理（タイマー割り込み）

タイマー割り込み手順確認

割り込み処理で最もよく使うのはタイマー割り込みです。  
機械式割り込みも覚えたいところですが、まずはタイマー割り込みを覚えましょう。  
このプログラムはオンボードLEDを500ms毎に点灯，消灯を繰り返します。

```
// タイマー割り込み処理
#include <MsTimer2.h>

// 割り込み時に処理される関数
void flash() {
    static boolean output = HIGH; // プログラム起動前に1回だけHIGH(1)で初期化
    //digitalWrite( 13, output ); // 13番ピン(LED)を出力する(HIGH>ON LOW>OFF)
    digitalWrite( 2, output ); // 13番ピン(LED)を出力する(HIGH>ON LOW>OFF)
    output = !output; // outputの内容を反転させる
}

void setup() {
    pinMode( 2, OUTPUT); // 13番ピンを出力に設定(LED)
    //MsTimer2::set( 500, flash ); // 500ms毎の割り込み、
    //その時に処理する関数 flash()を呼び出し
    MsTimer2::set( 1000, flash ); //1000ms毎の割り込み、
    //その時に処理する関数 flash()を呼び出し
    MsTimer2::start(); // タイマー割り込み開始
}

// 繰り返し実行される処理
void loop() {
    //処理なし
}
```

課題4-2 割り込み処理例（タイマー割り込み例）

SWの押された回数を、割り込み処理で7セグLEDに表示する。

通常SWのカウントアップはLED8個のLEDボードで行うのですが、LEDボードの点灯/消灯が意外と複雑になります。（課題43参照）  
まずは7セグメントLEDで行います。  
このプログラムはSWの押された回数を7セグメントLED1桁に表示します。  
10回押された場合は“A”、15回押された場合は“F”と表示します。  
16回目で“0”に戻します。

```
// タイマー割り込み処理
```

```
#include <MsTimer2.h>
```

```
#define SW 12
```

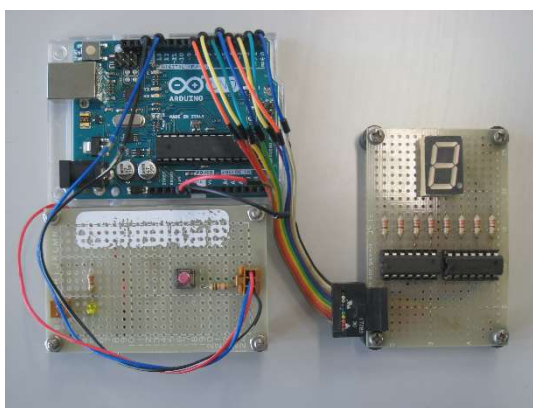
```
int count = 0;
```

```
int value = 0;
```

```
//7segLED レイアウトを定義
```

```
boolean Num_Array[16][8]={  
    {1,1,1,1,1,1,0,0}, // 0:0x3F のつもり  
    {0,1,1,0,0,0,0,0}, // 1:0x06 //  
    {1,1,0,1,1,0,1,0}, // 2:0x5b //  
    {1,1,1,1,0,0,1,0}, // 3:0x4F //  
    {0,1,1,0,0,1,1,0}, // 4:0x66 //  
    {1,0,1,1,0,1,1,0}, // 5:0x6d //  
    {1,0,1,1,1,1,1,0}, // 6:0x7d //  
    {1,1,1,0,0,0,0,0}, // 7:0x07 //  
    {1,1,1,1,1,1,1,0}, // 8:0x7f //  
    {1,1,1,1,0,1,1,0}, // 9:0x6f //  
    {1,1,1,0,1,1,1,0}, // A  
    {0,0,1,1,1,1,1,0}, // b  
    {0,0,0,1,1,0,1,0}, // C  
    {0,1,1,1,1,0,1,0}, // d  
    {1,0,0,1,1,1,1,0}, // E  
    {1,0,0,0,1,1,1,0} // F
```

```
};
```





```
//LED 表示関数
void NumPrint(int Number){
    for( int w=0; w<=7; w++){
        digitalWrite( w+1,Num_Array[Number][w] );
    }
}

// 割り込み時に処理される関数定義
void flash() {
    NumPrint( count );
}

void setup() {
    //1~8 番ピン デジタル出力へセット
    for ( int i=1; i<=8; i++){
        pinMode( i, OUTPUT );
    }
    pinMode( SW, INPUT_PULLUP );

    MsTimer2::set( 10, flash ); // 10ms 毎の割り込み、
                                //その時に処理する関数 flash()を呼び出し
    MsTimer2::start();         // タイマー割り込み開始
}

// 繰り返し実行される処理
void loop() {
    while( digitalRead( SW )==HIGH ) {} //SW 押されるまで待つ

    while( digitalRead( SW )==LOW ) {} // SW 放されるまで待つ

    count++;
    if( count==16) count=0;
}
}
```

課題4-3 割り込み処理例（タイマー割り込み例）

SWの押された回数を、割り込み処理でLEDボード(8個のLED)に表示する。

通常、課題4-2は8個のLEDボードで行います。

他のCPUでは「PADR=0x03;」などの記述ができ、非常に簡単なのですが  
Arduinoではこの記述ができません。意外と難しくなってしまうです。

このプログラムは課題4-2を8個のLEDボードに出力するものです。

```
// タイマー割り込み処理  
// LED ボード表示  
#include <MsTimer2.h>
```

```
#define BITS_DIGIT 8
```

```
#define SW          12
```

```
#define LED_BIT8    2
```

```
#define LED_BIT7    3
```

```
#define LED_BIT6    4
```

```
#define LED_BIT5    5
```

```
#define LED_BIT4    6
```

```
#define LED_BIT3    7
```

```
#define LED_BIT2    8
```

```
#define LED_BIT1    9
```

```
char BinaryString[ BITS_DIGIT + 1 ];
```

```
int cnt = 0;
```

```
int count = 0;
```

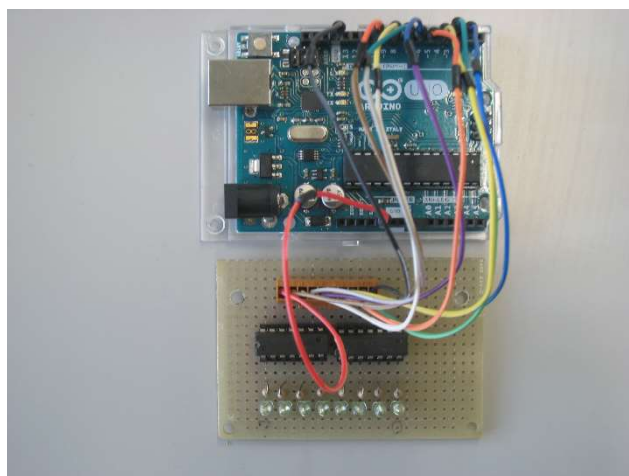
```
int before_count = 0;
```

```
/* 10進数-->2進数変換 */
```

```
void ChangeFunc(
```

```
    unsigned long int DecimalNumber,    /* 10進数[in] */
```

```
    char* BinaryString                    ) /* 2進数文字列化[out] */
```



```
{
    int i, k;
    for( i = 0, k = BITS_DIGIT - 1; k >= 0; i++, k-- )
    {
        if( (DecimalNumber >> i) & 1) BinaryString[k] = '1';
        else BinaryString[k] = '0';
    }
    BinaryString[i] = '\0';
}

void NumPrint( int Number )
{
    int i;
    ChangeFunc( count, BinaryString );

    for( i=0; i<BITS_DIGIT; i++ )
    {
        if( BinaryString[ i ] == '0')
        {
            digitalWrite( i+2, LOW );
        } else {
            digitalWrite( i+2, HIGH );
        }
    }
}

// 割り込み時に処理される関数
void flash()
{
    if( count==before_count ) {
        //なにもしない
    } else {
        NumPrint( count ); //LED ポート表示
    }
}
```

```
void setup()
{
    pinMode( LED_BIT8, OUTPUT);    // 2 番ピンを出力に設定(LED)
    pinMode( LED_BIT7, OUTPUT);    // 3 番ピンを出力に設定(LED)
    pinMode( LED_BIT6, OUTPUT);    // 4 番ピンを出力に設定(LED)
    pinMode( LED_BIT5, OUTPUT);    // 5 番ピンを出力に設定(LED)
    pinMode( LED_BIT4, OUTPUT);    // 6 番ピンを出力に設定(LED)
    pinMode( LED_BIT3, OUTPUT);    // 7 番ピンを出力に設定(LED)
    pinMode( LED_BIT2, OUTPUT);    // 8 番ピンを出力に設定(LED)
    pinMode( LED_BIT1, OUTPUT);    // 9 番ピンを出力に設定(LED)
    pinMode( SW, INPUT_PULLUP);    // 12 番ピン入出力に設定(SW)

    //MsTimer2::set( 500, flash ); // 500ms 毎の割り込み、
    MsTimer2::set( 1, flash );    // 1ms 毎の割り込み、
    //その時に処理する関数 flash()を呼び出し
    MsTimer2::start();           // タイマー割り込み開始
}

void loop()
{
    int i;
    // スタート時は count=0、従って LED 全消灯
    for( i=0; i<8; i++) {
        digitalWrite( i+2, LOW );
    }
    before_count = count;

    while(1) {
        while( digitalRead( SW )==HIGH ) {} //SW押されるのを待つ

        while( digitalRead( SW )==LOW ) {} //SW離されるのを待つ
        before_count = count;
        count++;
        if( count>255 ) count=0;
    }
}
```

課題5 サーボ制御  
ラジコンサーボ制御

ラジコンサーボはロボット制作の材料(部品)としてよく使われます。  
基本動作を確認します。

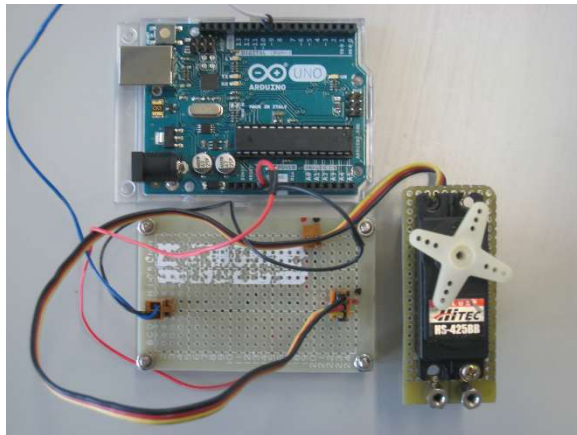
```
// rc_servo_sample1
#include <Servo.h>

Servo myservo;

int pos = 0;

void setup() {
  myservo.attach( 9 );
}

void loop() {
  for ( pos = 0; pos < 180; pos += 1 ) {
    myservo.write( pos );
    delay( 15 );
  }
  for ( pos = 180; pos > 0; pos -= 1 ) {
    myservo.write( pos );
    delay( 15 );
  }
}
```

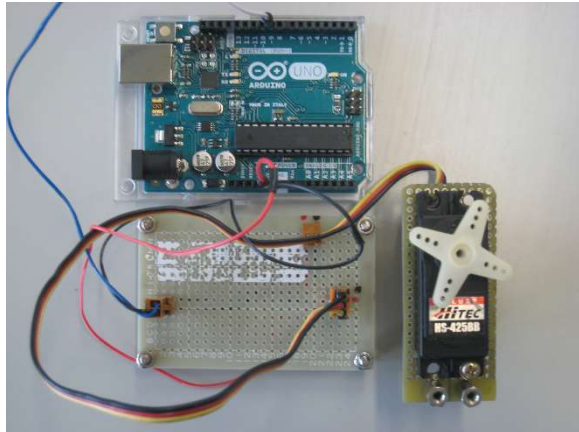


課題5-2 サーボ制御

ラジコンサーボ制御2

サーボニュートラを0とし、右0~90、左0~-90の値を引数とするhandle関数

```
// rc_servo_sample2
#include <Servo.h>
Servo myservo;
int pos = 0;
void handle( int angle ) {
    pos = angle + 90;
    myservo.write( pos );
}
void setup() {
    myservo.attach( 9 );
}
void loop() {
    int i;
    handle( 0 ); delay(500);
    for( i= 0; i<90; i++ ) {
        handle( i );
        delay( 15 );
    }
    for( i= 90; i>0; i-- ) {
        handle( i );
        delay( 15 );
    }
    for( i=0; pos>-90; i-- ) {
        handle( i );
        delay( 15 );
    }
    for( i=-90; i<0; i++ ) {
        handle( i );
        delay( 15 );
    }
}
}
```



課題6 簡単ライトレーサー  
ミニマイコンカー仕様ロボット

センサーボード (自作) センサー4個  
モータドライブボード (旧ミニマイコンカーモータドライブボード)  
SW1個, LED1個, モーター2個搭載

// ミニマイコンカーマイコンカー動作確認

```
#include <MsTimer2.h>
```

```
//モータドライブボード端子
```

```
// P7 動作確認 LED 茶
```

```
// P6 スタート SW 白
```

```
// P5 左 M 方向 橙
```

```
// P4 右 M 方向 黄
```

```
// P3 左 M PWM 緑
```

```
// P2 右 M PWM 青
```

```
// P1 未使用
```

```
// P0 未使用
```

```
#define ON_LED 13
```

```
#define SW 2 //白
```

```
#define LED 4 //茶
```

```
#define LMPWM 5 //緑
```

```
#define RMPWM 6 //青
```

```
#define LMDesision 7 //橙
```

```
#define RMDesision 8 //黄
```

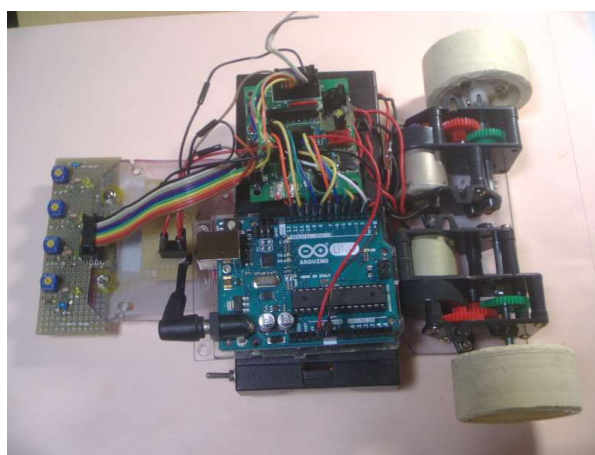
```
#define SENSOR_R 9
```

```
#define SENSOR_RC 10
```

```
#define SENSOR_LC 11
```

```
#define SENSOR_L 12
```

```
int cnt = 0;
```



```
int indata;
int sensor_value;

// 割り込み時に処理される関数
void flash() {
    static boolean output = HIGH; // プログラム起動前に 1 回だけ HIGH(1) で初期化
    cnt++; // カウントのみ。この例では使用せず。
}

void motor( int lspeed, int rspeed )
{
    int lpwm, rpwm;
    lpwm = lspeed * 2.54;
    rpwm = rspeed * 2.54;
    analogWrite( LMPWM, lpwm );
    analogWrite( RMPWM, rpwm );
}

void setup() {
    pinMode( SW, INPUT ); // 2 番ピンを出力に設定(SW)
    pinMode( LED, OUTPUT ); // 4 番ピンを出力に設定(LED)
    pinMode( LMPWM, OUTPUT ); // 5 番ピンを出力に設定(LED2)
    pinMode( RMPWM, OUTPUT ); // 6 番ピンを出力に設定(LED3)
    pinMode( LMDesision, OUTPUT ); // 7 番ピンを出力に設定(LED2)
    pinMode( RMDesision, OUTPUT ); // 8 番ピンを出力に設定(LED3)

    pinMode( ON_LED, OUTPUT );

    pinMode( SENSOR_R, INPUT ); // 9 番ピンを入力に設定(右 sensor)
    pinMode( SENSOR_RC, INPUT ); // 10 番ピンを入力に設定(右中 sensor)
    pinMode( SENSOR_LC, INPUT ); // 11 番ピンを入力に設定(左中 sensor)
    pinMode( SENSOR_L, INPUT ); // 12 番ピンを入力に設定(左 sensor)

    MsTimer2::set( 10, flash ); // 10ms 毎の割り込み、
    //MsTimer2::set( 1, flash ); // 1ms 毎の割り込み、
    //その時に処理する関数 flash() を呼び出し
```



```
MsTimer2::start();          // タイマー割り込み開始
}

// 割り込み処理は動いているか、この例では使用せず。
void loop() {
  int i;
  digitalWrite( LED, HIGH ); //LED 消灯
  analogWrite( LMPWM, 0 );
  analogWrite( RMPWM, 0 );
  digitalWrite( LMDesision, LOW );
  digitalWrite( RMDesision, LOW );

  for( i=0; i<5; i++) {
    digitalWrite( LED, LOW );   //LED 点灯
    delay(200);
    digitalWrite( LED, HIGH ); //LED 消灯
    delay(200);

    //digitalWrite( ON_LED, HIGH ); //ON_LED 点灯
    //delay(200);
    //digitalWrite( ON_LED, LOW ); //ON_LED 消灯
    //delay(200);
  }

  while(1) {
    sensor_value = 0;
    indata = !digitalRead( SENSOR_R );
    sensor_value = indata;

    indata = !digitalRead( SENSOR_RC );
    sensor_value = sensor_value + 2*indata;

    indata = !digitalRead( SENSOR_LC );
    sensor_value = sensor_value + 4*indata;

    indata = !digitalRead( SENSOR_L );
```

```
sensor_value = sensor_value + 8*indata;

switch( sensor_value ) {
  case 6: //○●●○
    digitalWrite( LED, LOW ); //LED 点灯
    digitalWrite( ON_LED, LOW ); //ON_LED 消灯
    motor( 100, 100 );
    break;

  case 2: //○○●○
    digitalWrite( LED, LOW ); //LED 点灯
    digitalWrite( ON_LED, LOW ); //ON_LED 消灯
    motor( 80, 60 );
    break;

  case 4: //○●○○
    digitalWrite( LED, LOW ); //LED 点灯
    digitalWrite( ON_LED, LOW ); //ON_LED 消灯
    motor( 60, 80 );
    break;

  case 3: //○○●●
    digitalWrite( LED, LOW ); //LED 点灯
    digitalWrite( ON_LED, LOW ); //ON_LED 消灯
    motor( 60, 40 );
    break;

  case 12: //●●○○
    digitalWrite( LED, LOW ); //LED 点灯
    digitalWrite( ON_LED, LOW ); //ON_LED 消灯
    motor( 40, 60 );
    break;

  case 1: //○○○●
    digitalWrite( LED, LOW ); //LED 点灯
```

```
digitalWrite( ON_LED, LOW ); //ON_LED 消灯
    motor( 30, 10 );
    break;

case 8: //●○○○
    digitalWrite( LED, LOW ); //LED 点灯
    digitalWrite( ON_LED, LOW ); //ON_LED 消灯
    motor( 10, 30 );
    break;

case 0: //○○○○
    digitalWrite( LED, HIGH ); //LED 消灯
    digitalWrite( ON_LED, HIGH ); //ON_LED 点灯
    motor( 0, 0 );
    break;

default:
    digitalWrite( LED, HIGH ); //LED 消灯
    digitalWrite( ON_LED, LOW ); //ON_LED 消灯
    motor( 0, 0 );
    break;
}
}
}
```

## コマンドの確認(リファレンスマニュアルより抜粋)

### 【デジタル入出力】

- `void pinMode (pin, mode);`
  - pin : ピン番号 2~13 (0~13), A0~A5
  - mode : 動作モード INPUT, OUTPUT, INPUT\_PULLUP
  - 返値 : なし

pin 番のデジタル入出力ピンを**入力モード** (INPUT) あるいは**出力モード** (OUTPUT) に設定します。INPUT\_PULLUP は INPUT と同様に入力モードになりますが、内蔵プルアップ抵抗が有効化される点が異なります。ピンの番号は、通常は 2~13 となりますが、PC と通信しないスタンドアロン動作時は 0~13 に広げることができます。また、アナログ入力ピン A0~A5 についても、この関数で動作モードを設定することで、デジタル入出力ピンとして利用できます。

- `void digitalWrite (pin, value);`
  - pin : ピン番号 2~13 (0~13), A0~A5
  - value (出力モードの場合) : デジタル出力値 LOW (0), HIGH (1)
  - 返値 : なし

pin 番のピンが出力モードの場合、オフ (LOW) あるいはオン (HIGH) を**出力**します。

- `int digitalRead (pin);`
  - pin : ピン番号 2~13 (0~13), A0~A5
  - 返値 : デジタル入力値 LOW (0), HIGH (1)

pin 番のピンの電圧が、オフ (約 0V) のときは LOW を返し、オンのときは HIGH を返します。入力モードだけでなく、出力モードでも使えます

## 【PWM 出力】

- void analogWrite (pin, value);
  - pin : ピン番号 3, 5, 6, 9, 10, 11
  - value : PWM デューティ比 (0~255)
  - 返値 : なし

pin 番のピンから PWM 信号 (約 490Hz) を出力します。そのデューティ比 (オンになっている時間の割合) は  $value / 255.0$  となります。つまり  $value = 0$  のときは常にオフ, 128 のときは 50%, 255 では常にオンとなります。analogWrite() を実行すると、ピンは自動的に出力モードになります。

## 【アナログ入力】

- int analogRead (pin);
  - pin : アナログピン番号 0~5 (a0~a5 に対応)
  - 返値 : アナログ入力値 (0~1023)

pin 番のアナログピン (a0~a5) の電圧をよみとり、マイナス電源の電圧 (0V) を 0 とし、プラス電源の電圧 (5V) を 1023 とする整数値 (10bit) にアナログ-デジタル変換します。たとえばポテンシオメータ (可変抵抗器) のツマミの角度を読み取ったり、あるいは何らかのセンサの出力信号を読み取るために使います。アナログ信号をデジタル値に変換するには、およそ 0.1ms かかります。

## 【サーボ制御出力】

まず、**Servo ライブラリを読み込みます**。Sketch > Import Library... > Servo を読み込んでください。スケッチの先頭に `#include <Servo.h>` が挿入されます。Servo クラスの変数（たとえば servo という名前のインスタンス）をつくると、以下のようなメソッドを利用できるようになります。

- `void servo.attach (pin);`
  - pin：ピン番号 2~13
  - 返値：なし

Servo クラスのインスタンス servo を、pin 番のピンに対応づけます。これでサーボ制御信号（回転位置は中央 90deg）が出力されるようになります。すでに他のインスタンスに対応づけたピンは指定しないほうがよいでしょう。（いずれかのピンが attach() されているとき、9番・10番ピンからの PWM 出力が出来なくなります。バグではなく仕様です。）

- `void servo.write (deg);`
  - deg：回転位置 (deg)
  - 返値：なし

Servo クラスのインスタンス servo を、回転位置 deg まで回転させます。単位は度 (deg) で、中央位置を 90deg とし、0~180deg の範囲で指定することができます。（あらかじめ servo をいずれかのピンに attach() しておく必要があります。）

- `void servo.detach ();`
  - 返値：なし

Servo クラスのインスタンス servo について、それまで attach() されていたピンとの対応づけを解除します。