コンピュータシステムを
作っちゃおう！

目的・ねらい

パソコンとは違う小さなシステムを自分で組み上げる体験を通して、ハードウェアに対する理解を深めるとともに、身の回りのシステムに興味をもって欲しいと考えています。プログラムをすべて理解させるのではなく、特に初心者に対しては数値を変えたら動きが変わった、という体験をさせることに重点を置いてください。上級者に対しては、自分でどんどん変えさせて構いません。初心者には成功体験が、上級者にはトライ＆エラーが大事です。

実施内容の概要

Arduino（アルデュイーノ）という小さなコンピュータを使って、音を鳴らす・音の高さを変える・サーボモーターを回す・距離センサーを使う、といったことを体験しながらプログラムの動作を理解し、パソコンとは違う小さなコンピュータの世界に触れます。

講師用の実施手順の詳細

準備することがら、物品など

- 作業しやすいようにバナナジャック化された Arduino ボードと周辺機器を使います。
- 使うコンピュータでは Arduino.exe とメモ帳をタスクバーに登録しておきます。Windows+1, Windows+2 などで起動させると無駄な時間をとりません(Windows10 の場合)。
- Arduino は Java Accessibility Switch が ON になっていないと音声出力しないので、java¥bin ディレクトリにて「jabswitch -enable」を実行しておきます。(ただし違う Java を読み込むような PATH の設定だとうまくいかないことがあります。)スクリーンリーダーには NVDA を用います。
- メモ帳やエディタでの編集には、使い慣れたスクリーンリーダーを使わせる方がよいかもしれません。その際は、Arduino の音声出力をあきらめてショートカットだけの操作をする、もしくはそこの操作は補助者がやってしまう、という選択肢もあります。いずれにせよプログラムを音声で確認する場合に、中カッコなどの記号をきちんと読むように調整しておくことが大事です。
- ツールメニュー以下のシリアルポートやボードの種類を設定しておきます。
- サンプルプログラムをドキュメントフォルダ＞Arduino フォルダ内においておきま

す。Alt メニューのファイル➡スケッチから開かせます。動作確認もしておきます。最初に接続した際に音が鳴らないよう、空のスケッチを最後に入れておきます。

実施手順の詳細

導入部分

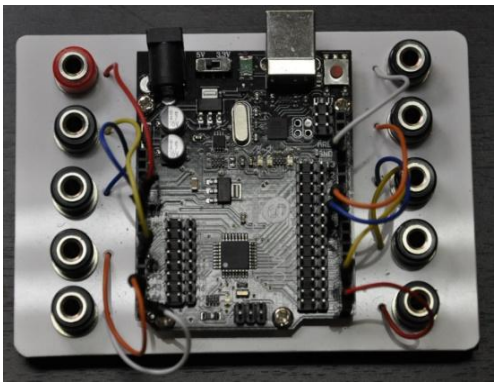
概要を説明し、以下の用語説明をします。

- 「Arduino」とは今日使う小さなコンピュータである。
- 「スケッチ」とは Arduino を動かすプログラムである。
- スケッチ（プログラム）を作るためのプログラム＝ソフトウェアがある。
- そのソフトウェアの名前も Arduino(.exe)である。
- アナログは連続する（＝つながっている）量、温度や角度など実世界に存在する量
- デジタルはバラバラの量、コンピュータに入るとデジタル化される。

そして、Arduino はパソコンがなくても「コンピュータとして」単体で動くことも説明しておきます。

ARDUINO の概要説明

Arduino ボードの右上の触覚手掛かりシールなどを使い、同じ方向を向かせてから触らせます。触らせながら、以下を確認させ、説明します。



視覚障害者向けにバナナジャックを装備した Arduino ボード

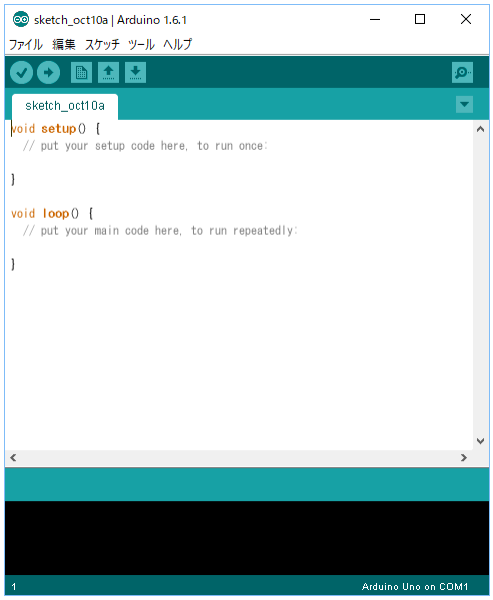
- USB コネクターの位置
- USB ケーブルを通してスケッチが Arduino に書き込まれること。
- バナナジャックの位置と数、そして実際にはそれらがケーブルを通して Arduino のコネクターに接続されていること。
- 静電気破壊の恐れがなければ CPU のチップや水晶発振器などを説明する。
- アナログ入力をする端子やデジタルの入出力をする端子があること。

これらの触察の際はケガをしないように、またハードウェアを壊してしまわないようにゆっくり触るよう指導します。触らせる方向を統一しておかないと説明に苦労するので、向きを変えないように伝えておきます。

アナログ入力については「電圧の高さを検出する」という程度の説明をします。デジタルの方は５Ｖか 0V かを判断したり出力したりするという説明をします。（高学年の場合には、５Ｖといっても「だいたい 5V くらい」という話を盛り込むのも良いかもしれません。）

ARDUINO.EXE の起動と使い方の説明

コンピュータを起動し、Windows+1 で Arduino.exe を、必要であれば Windows+2 でメモ帳やエディタを起動します。メモ帳やエディタを使う場合は Alt+Tab でのアプリケーション切り替えができることを確認してください。プログラムの編集は Arduino.exe がうまく読む場合には、そのまま使うことも可能ですが、音声化がうまくいかなかったり、編集に困難があったりするような場合はメモ帳などを使わせてください。Ctrl+A, Ctrl+C, Alt+Tab, Ctrl+A, Ctrl+V、の一連のシーケンスでコピー＆ペーストすることなどを説明し、操作について確認します。作業に慣れている生徒なら問題ないのですが、作業の速度差が出てしまう場合があります。適宜補助して、プログラムの内容を読ませ理解させることに時間を使うように流れをコントロールしてください。



Arduno.exe を起動した様子

スケッチを Arduino へアップロードするショートカットキーは Ctrl+U であることをここで説明します。適宜自由にメニューを読ませてショートカットキーを確認させても良いです。アップロードの際には確実に Arduino がケーブルで接続されていることを確認するよう指導してください。画面が見える生徒に対しては、ボタンアイコンの説明もしてください。コンパイルは Ctrl+R ですが、かえって混乱させてしまうようなら、あえてワークショップでは説明しなくて

も良いかと思います。デバッグペインへは、F6 で移動できます。F8 での移動をするバージョンもありましたが、現状では F8 は境目のバーへのフォーカスの移動のようです。

もしプログラムに詳しい生徒の場合には、setup()と loop()の 2 つのセクションがあり、setup()は最初の一回、loop()は無限繰り返しであることなども説明すると良いでしょう。

本家のリファレンスは以下になります。

https://www.arduino.cc/en/Guide/HomePage

圧電スピーカーでメロディ作成

最初に圧電スピーカーを使って音楽を鳴らすサンプルを試します。好みのメロディを構築させて興味をもってもらうのが目的です。具体的には以下の手順で進めます。

- まずファイル➡スケッチブックから最初のサンプルスケッチを開かせます。そのまま読むのが困難な場合はメモ帳にプログラムをコピーさせます（Ctrl+a, Ctrl+c, Alt+Tab, Ctrl+v など）。生徒が分かっている場合には、直接 INO ファイルを編集させるのも良いかもしれません。最初なので適宜手助けして、生徒間のペースを整えます。
- ファイルを開いた後に NVDA でも読ませつつプログラムの内容を順に説明します。
  - スラッシュが 2 つ続いた後はコメントでプログラムには影響しない説明文であることを伝えます。この説明文をコメントと呼ぶことや「コメントアウト」という言葉などを教えるのも良いでしょう。
  - 定数名が数値を意味することを理解しているか確認しつつ進めます。プログラムの基本としての「定数」「変数」の話がしたくなるところですが、難しいことは言わずに「文字が数字の値を意味する」のような説明にとどめておきます。変数は次のサンプルで出てきます。
  - tone()は音を鳴らす命令で、その後に delay()という命令が必要なことを教えます。適宜英語の意味なども説明に入れると良いでしょう。生徒の様子を見て難しく感じてしまいそうならば、これらの命令で 1 つの音を鳴らす、といった程度の説明でも構いません。これらの編集を後ほどしてもらうことも伝えます。
- Arduino のボードに圧電スピーカーを接続させます。赤をデジタル 10 番に、黒を GND に接続させます。ここは「自分で挿す」ことに重点を置いてください。ただし、乱暴に扱って壊してしまわないよう、ゆっくり触るよう、分からないことがあったら聞くよう指導してください。「高価なものである」と最初に伝えておくことも重要です。
- 接続が終わったら、Ctrl＋U を押させ、サンプルスケッチをコンパイル＆アップロードします。

- モバイルバッテリーなどがあれば、コンピュータから外して単体で実行してみます。単純なオルゴール的なものですが、プログラムが転送されて中で動いていることを実感してもらいます。
- サンプルプログラムが動いたら、音階を自由に組み合わせて自由にメロディを作ってもらいます。ここで比較的多くの時間を割いて、成功体験をしてもらいます。この作業は最後に持ってくる構成も良いと思います。
- 音楽の才能のある生徒さんには、何か課題曲を与えても良いでしょう。
- 進度の早い生徒さんや音楽の素養のある生徒さんには音の長さも変えさせます。delay の値を調整して「間」を作ることも試してもらいます。

スピーカーの接続の際には、「USB コネクターを上に向けて置いて、右側の一番上のバナナジャックがデジタルの 10 番ピンです」「その下が GND です」といった声掛けで説明します。

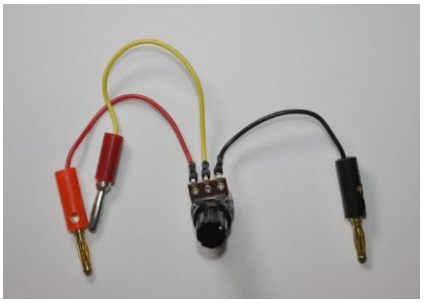
プログラムの説明時、tone()メソッドは 3 つの引数を持つことを生徒に合った言葉で説明します。「カッコの中に 3 種類の値をカンマで区切って書く」「1 番目の値はピン番号、これはすべて同じなので 2 行目で 10(番)になるようにしてある」「2 番目の値が周波数、数字が大きいほど高い音」「3 番目の値は音の長さ、これもサンプルでは同じ値にしてある」など適宜かみ砕きながら説明します。

```
// 音の長さを 300 ミリ秒に設定します。
#define DURATION 300
// スピーカーのピンをデジタル 10 番とします。
#define SPEAKER_PIN 10
// プログラムには関数という命令のまとまりがあります。中カッコで区切られています。
// setup()という関数は最初に実行されます。しかし今回は何も書きません。
void setup() {
// 中身はカラです。
}
//loop()という関数は無限に繰り返されます。
void loop() {
  // ドの周波数を 300 ミリ秒鳴らします。  delay という命令で次の命令まで待ちます。
  tone(SPEAKER_PIN,262,DURATION);  delay(DURATION);
  // レ
  tone(SPEAKER_PIN,294,DURATION);  delay(DURATION);
  // ミ
  tone(SPEAKER_PIN,330,DURATION);  delay(DURATION);
  // ファ
  tone(SPEAKER_PIN,349,DURATION);  delay(DURATION);
```

```
  // ソ
  tone(SPEAKER_PIN, 392,DURATION);  delay(DURATION);
  // ラ
  tone(SPEAKER_PIN,440,DURATION);  delay(DURATION);
  // シ
  tone(SPEAKER_PIN,494,DURATION);  delay(DURATION);
  // ド
  tone(SPEAKER_PIN,523,DURATION);
  // 3秒待つ
  delay(3000);
}
```

可変抵抗器によるアナログ入力

2番目のスケッチはアナログ入力を学びます。入力によって音階が変化することを理解し、手元の可変抵抗器がプログラムに影響していることを感じてもらいます。「可変抵抗器」という名称は親しみがないかもしれませんので、「ボリュームのつまみ」など適宜言い方を変えてみてください。（つまみ、も死語かもしれませんが…）



可変抵抗器

- ファイル➡スケッチブックからスケッチを開かせます。
  - 3つの定数の宣言文を解説します。ここまではスピーカーのサンプルと同じです。
  - 変数宣言が出てきました。ここで定数はその名の通り値が変わらないもの、変数はプログラム中で変化する入れ物、ということを説明します。また、intについては整数（integer）であることも説明しておきます。
  - setupは今回も何もないので飛ばします。もとよりカットしておくのも一案です。
  - analogRead()というメソッド（命令）で指定されたポートの電圧を測ることを教えます。
  - 圧電スピーカーの周波数として値が与えられ、出力されることを説明します。
- Arduinoからスピーカーの片方のプラグを抜き、音が鳴らないようにしてからCtrl+Uでスケッチのコンパイルとアップロードを実施します。
- USBケーブルを抜いた後にArduinoのボードに可変抵抗器を接続させます。赤いケーブルを5Vに、黒いケーブルをGNDに、黄色いケーブルをアナログの0番に接続しま

す。スピーカーも再度接続します。

- 可変抵抗器の黄色いケーブルの電圧は、つまみを捻ると 0V から 5V に変わること、アナログ入力端子では、0V から 5V までの間の電圧値が 0 から 1023 になることを説明しておきます。可変抵抗器の各端子の意味と仕組み、電圧が変わる原理なども時間に応じて説明すると良いでしょう。
- USB ケーブルを電源もしくは PC につなぎます。
- サンプルが動いたら、プログラムを自分で変えさせます。例えば sensorValue の値を sensorValue*2 にしたり 1023-sensorValue にしたりさせてその変化を予想させ、実験します。変数に代入するタイミングで変えても tone を呼び出すタイミングで変えても同じということを示すことで、「プログラムの正解はひとつではない」ことを理解してもらうと良いでしょう。
- analogRead()の解説は以下のページにあります。

http://www.musashinodenpa.com/arduino/ref/index.php?f=0&pos=2113

```
//音の長さを定義します。
#define DURATION 50
// スピーカーのピンをデジタル 10 番とします。
#define SPEAKER_PIN 10
// 可変抵抗から取得するピンをアナログ 0 番とします。
#define RESISTOR_PIN A0
// 可変抵抗器からの値を入れる変数 sensorValue を宣言して初期化します。
int sensorValue = 0;

void setup() {
  // 今回も何も書きません。
}

void loop() {
  //analogRead 命令でアナログ 0 番から値を読込んで、sensorValue に代入します。
  sensorValue = analogRead(RESISTOR_PIN);
  //sensorValue の周波数の音を 50 ミリ秒鳴らします。
  tone(SPEAKER_PIN, sensorValue, DURATION) ; delay(DURATION);
}
```

サーボモーターでループを学ぶ

3番目のスケッチはサーボモーターを使います。サンプルスケッチの確認、スケッチのアップロード、サーボモーターの接続、実行、の順に進めます。（サーボモーターが接続されているとUSBの電流が吸われてうまく書き込めない場合があります）

サーボモーター

- ファイル➡スケッチブックからスケッチを開かせます。
- ヘッダファイルのインクルードについては「おまじない」という言い方は避け、「サーボへの命令を記述するために必要なコード」と伝えます。
- forループについての書式を学びます。「for()の中にはセミコロンで区切られて3つの領域があり、最初の領域で変化する変数の初期値（最初の値）を、2番目の領域で終わる条件を、3 番目の領域で変化の仕方を指定する」「中カッコで囲まれた部分が指定回数だけ回る、そして変数の中身も変化する」ということを説明します。ここではposが0から180まで変化することを理解してもらいます。難しい様子でしたら、すべて理解させなくて構いません。どこを変えたら角度が変わるか、を教えて自由課題ではそこを編集させてください。
- Ctrl+Uでコンパイル、アップロードを実施します。
- サーボの電源とPWM端子を接続します。
- 自分たちで角度の指定を変えさせたり、ステップ（＝速度）を変えさせたりして動きの変化を楽しんでもらいます。

```
// サーボを使うためのヘッダファイル（宣言ファイル）を読み込みます。
#include <Servo.h>
// サーボオブジェクトを作ります。
Servo myservo;
// サーボモーターの角度を記憶しておく変数 pos を宣言してゼロで初期化
int pos = 0;
void setup() {
  // 0 番ピンをサーボモーターと関連づけます。
  myservo.attach(0);
}
void loop() {
  // 指定回数まわす for ループです。pos の中身は 0 から 180 まで 1 ずつ増えます。
  for(pos = 0; pos <= 180; pos += 1){
```

```
    // サーボを pos の角度に変更して 10 ミリ秒待ちます。
    myservo.write(pos);   delay(10);
  }
  // 1 秒待ちます。
  delay(1000);
  // 次にまた for ループで pos の値を 180 から 0 まで減らします。
  for(pos = 180; pos >= 0; pos -= 1){
    myservo.write(pos);   delay(10);
  }
  // 1 秒待ちます。
  delay(1000);
}
```

可変抵抗器のアナログ入力とサーボモーターを組み合わせてみる

可変抵抗器からの入力の仕方とサーボモーターへの出力の仕方が分かったので、それらを組み合わせて「可変抵抗器をねじるとサーボが動く」というサンプルを動かしてみます。

- ファイル➡スケッチブックからスケッチを開かせます。
- 先ほど学んだ 2 つのプログラムが組み合わさっていることを理解させます。どのコードがどちらのものであったかなどを答えさせても良いでしょう。
- サンプルが動いたら、可変抵抗器の向きと逆向きに動かすには、角度を 90 度までにするには、といった課題を出してみます。
- 時間を持て余している生徒さん、すでにプログラミング経験が豊富な生徒さんには、個別に条件分岐書式などを教えて一定値を超えたらサーボモーターが振り切れるものを作れ、などの課題を与えても良いでしょう。

```
// サーボを使うためのヘッダファイル（宣言ファイル）を読み込みます。
#include <Servo.h>
// 可変抵抗から取得するピンをアナログ 0 番とします。
#define RESISTOR_PIN A0
// サーボオブジェクトを作ります。
Servo myservo;
// 可変抵抗器からの値を入れる変数 sensorValue を宣言して初期化します。
int sensorValue = 0;
void setup(){
  // 0 番ピンをサーボモーターと関連づけます。
  myservo.attach(0);
}
```

```
void loop() {
  // analogRead 命令でアナログ 0 番から値を読込んで、sensorValue に代入します。
  sensorValue = analogRead(RESISTOR_PIN);
  // 0-1023 を 0-180 に変換します。
  sensorValue = map(sensorValue, 0, 1023, 0, 180);
  // サーボを pos の角度まで動かして 10 ミリ秒待ちます。
  myservo.write(sensorValue);  delay(10);
}
```

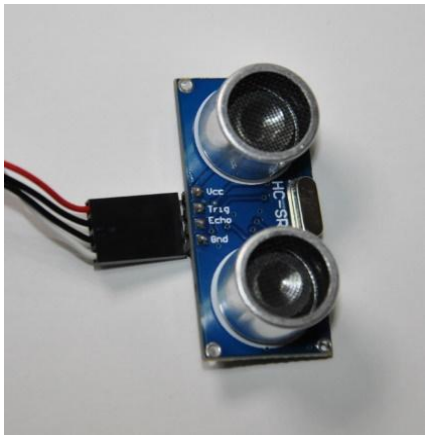
分岐の例

```
void loop() {
  sensorValue = analogRead(RESISTOR_PIN);
  if(sensorValue > 500){
    myservo.write(100);
    delay(10);
  }else{
    myservo.write(0);
    delay(10);
  }
}
```

センサーを使う

最後に超音波距離センサーを使ってみます。まずセンサー自体の説明をしてからサンプルを動かします。

- センサーには電源のプラスとマイナス（5 V と GND)、TRIG と ECHO という 4 つの端子があります。TRIG（トリガー）は、Arduino からセンサーへの信号線です。この端子にタイミングの指示があったら、超音波を発信します。ECHO（エコー）はセンサーから Arduino への信号線です。超音波発信から受信までどれくらいの時間が経ったかを、パルスの長さで戻してくれます。
- プログラムにある pulseIn というメソッドは ECHO の信号をマイクロ秒で答えてくれます。



超音波による距離センサー

- ファイル➡スケッチブックからスケッチを開かせます。
  - 宣言部分の定数を確認します。同時にボードのバナナジャックの対応位置なども説明すると良いでしょう。
  - double というのは整数ではなく小数を表す、という程度の説明をします。
  - デジタルポートの入力・出力の方向を setup で定めます。同時にシリアルポートの速度もここで指定していることに留意させてください。
  - パルスを送り、ECHO から戻ってくる値で距離を計算します。
  - センチメートル単位にして、それをもとに音階を出すことを説明します。
  - 同時にシリアルポートを通してパソコンと Arduino が通信することを説明します。ただしこのコードは弱視者・晴眼者の確認用なので、シリアルモニタはうまく読み上げません。
- 弱視の生徒の場合は動いている様子を、「シリアルモニタ」で確認することもできます。シリアルモニタは Ctrl+Shift+M で起動します。ボーレートに注意してください。9600 に設定します。
- 生徒の興味と体力があれば、0.017 という係数について少し詳しく説明します。(http://marupeke296.com/EL_Ard_No8_SuperSonic.html に詳しく書いてあります)時間と速さと距離の関係から、距離＝時間×速さです。ここで時間とは、pulseIn で得られるマイクロ秒単位の時間です。速さとは、音速の 340m/s です。cm 単位にすると 34,000cm/s です。そしてマイクロ秒を秒に直すには 1,000,000 で割ればよいので、プログラムの interval という変数を用いると、
  interval ÷ 1,000,000 × 34,000 = interval × 0.034 になります。これが往復の距離になるので、その半分、すなわち interval × 0.017 が片道の距離ということになります。

```
// 音の長さ、スピーカーピン、トリガーピン、エコーピンのナンバーを宣言します。
#define DURATION 100
#define BEEP_PIN 10
#define TRIG_PIN 2
#define ECHO_PIN 8
// interval と distance を宣言します。
int interval = 0;
double distance = 0;

void setup() {
  // 入出力の方向を決めてシリアルポートを開きます。
  // 2 番が出力、8 番が入力です。
  pinMode( 2, OUTPUT );
  pinMode( 8, INPUT );
  // Serial.begin( 9600 );//PC と通信する場合シリアルポートを開きます。
}
```

```
void loop() {
  // 10 マイクロ秒以上のパルスを送ります。HIGH, LOW を順番に指示します。
  digitalWrite( TRIG_PIN, HIGH );
  delayMicroseconds( 100 );
  digitalWrite( TRIG_PIN, LOW );
  // pulseIn()で ECHO ピンの HIGH である状態がどれくらい長いのか検出します。
  interval = pulseIn( ECHO_PIN, HIGH );
  // interval の値からセンチメートルに変換します。
  distance = interval * 0.017;
  // 音階で出力します。
  tone(BEEP_PIN, interval/10, DURATION);  delay(DURATION);
  // PC で状態を得る場合シリアル通信をします (Ctrl + Shift + M)
  //Serial.print( interval, DEC );
  //Serial.print( "¥t" );
  //Serial.print( distance, 4 );
  //Serial.print( "¥n" );
}
```

注意すべき点

- 直接ボードに触れて学習するので、冬の場合は特に生徒の静電気を逃がしておきます。服装なども注意して、場合によっては薄い手袋などを用意しておくと良いでしょう。部屋の気温が寒いと毛糸の上着などを脱いでもらうのが申し訳ないので、事前に部屋を暖めておくようにします。

到達目標

- コンピュータを使ったシステムの基本、入力→演算→出力という一連の流れを理解する。
- アナログ量を数値化してプログラムに利用していることを理解する。
- 世の中にある様々なセンサーについて興味を持つ。

生徒用資料

生徒用には、サンプルプログラムをあらかじめコンピュータに入れておきます。そのほか実際に触る Arduino は自分で組めるように大型化したものを準備します。