心理学実験におけるArduinoの活用：フリッカー刺激呈示装置の開発

高　橋　　優（基礎教育センター工学部会）

今日，心理学実験を行う上でパーソナルコンピュータ（以下コンピュータ）は欠かすことのできない道具となっている．刺激の作成はもちろん，刺激の呈示や実験参加者の反応の記録にもコンピュータは使用され，試行のランダム化や呈示時間の制御，反応の記録の自動化などを可能としている．コンピュータを用いた心理学実験の入門書も出版されており（たとえば北村・坂本，2004），学部における実験演習などを想定したテキストでもコンピュータの使用を前提にしたものが多い（西本，2012；酒井・森下・松本，2007；水野，2004）．

しかしながら，コンピュータ単体では実施が困難な実験もある．そのひとつとして，フリッカーテストを挙げることができる．点滅する光刺激の交代速度が速くなると，ちらつき（flicker）が消失し，光が点灯しているように知覚される．このときの交代の頻度を臨界融合頻度（critical fusion frequencyまたはcritical flicker frequency：臨界ちらつき頻度，以下CFF）と呼ぶ．CFFは光刺激の輝度や大きさ，波長により変化するが，高輝度であれば50～60Hzに達する（佐藤，2007；和気・長谷川，1994）．CFFは疲労によっても変化することから，疲労の程度を測る指標としても用いられている．

CFFの測定では，閾値を探すために光源の点滅周期を小さな変化量で操作することが求められる．しかし，液晶ディスプレイやCRTなどのコンピュータの標準的な表示装置では，表示の更新は画面を書き換える頻度である垂直走査周波数に制約されるため，こうした操作は困難である．たとえば，ディスプレイの垂直走査周波数が60Hzならば1/60秒単位でしか点滅を制御できないため，ディスプレイ上で光点の点滅する周波数を1Hzずつ変化させるような制御はできない．このため，光刺激を呈示するためには，点滅の同期を細かく制御できる外部装置をディスプレイとは別に用意する必要がある．しかし，こうした装置は高価であることが多い．

こうしたときにArduinoが便利である．Arduinoはオープンソースのプ

ロトタイピング用プラットフォームとしてイタリアで開発されたマイコンで，１台約２～3,000円で入手することができる．Arduinoにはいくつかのモデルがあるが，現時点の標準的なボードであるArduino Unoの場合，コンピュータと接続するためのUSBポートの他，14のデジタル入出力（うち６はパルス幅変調（pulse width modulation; PWM）出力が可能），６のアナログ入力ピンを持つ（Arduino, 2010）．Unoの他にも，入出力のピン数の増設されたモデルや追加機能の盛り込まれたモデルなど，さまざまなバリエーションがある．また，オープンソース・ハードウェアであるため，多くの業者から互換機も販売されている．

制御にはC++に似た独自の言語（Arduino言語）が用いられる．プログラム（Arduinoでは「スケッチ」と呼ぶ）は，コンピュータ上の統合環境で作成・コンパイルされ，USBを介してArduinoへと転送される．

刺激呈示やセンサなどの外部装置の制御をArduinoに任せることができるため，ハードウェアの制御に慣れていない者でも，それほど苦労せずに装置を開発することができる．時間制御について心理学実験に十分な程度の精度を持つ一方で，金額的にも手軽であることから，心理学における実験プラットフォームとしても注目されており（D'Ausilio, 2012），認知心理学（Gianelli, Ranzini, Marzocchi, Micheli, & Borghi, 2012）や視覚研究（Teikari, Najjar, Malkki, Knoblauch, Dumortier, Gronfier, & Cooper, 2012），社会心理学（Huynh, Hardy, Pezzo, & Wilder, 2012）において使用され始めている．

本稿ではArduinoの利用例として，フリッカーテストの演示用刺激呈示装置の開発例を紹介する．

１．装置の作製

１．１ ハードウェア

今回はArduinoを用いてLEDの点滅を制御する装置を作製した．点滅の周期の範囲は31～99Hzとし，ボタン操作により１Hz単位で上下することとした[1]．装置は，点滅する光刺激を呈示する刺激呈示用LED，周期を上下させる変更スイッチ，現在の周波数の表示する数値表示部の３点で構成

1　本来であれば，20Hzくらいから表示するべきだが，今回使用した環境（Arduino 1.0.3）においてArduino Duemilanoveもしくは Unoを用いた場合，tone()関数により30Hz以下の周波数を指定しても正常に動作しなかったため，正常に動作する31Hzからとした．
　なお，Arduino Leonardoを用いると，２Hzから表示可能だった．

される．

呈示周波数に関しては，今回のように数値表示部を用意する方法の他に，USB経由で接続したコンピュータのシリアルコンソールに呈示周波数を表示させる方法もある．今回は実験演習における使用を想定したため，装置単体でも実験ができるよう[2]，独立した表示部を持たせた．周波数を変更するスイッチも同様である．

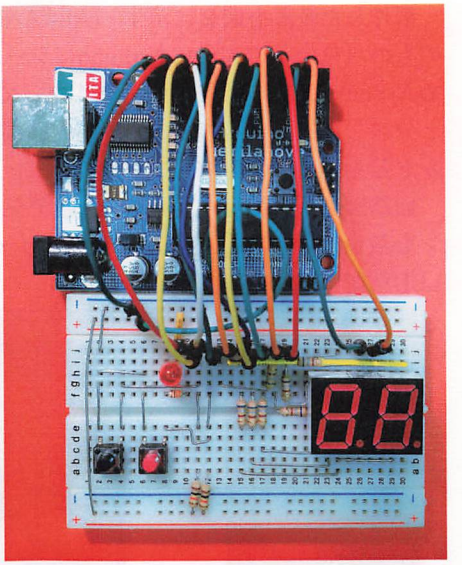

図1．ブレッドボード上でのフリッカーテストの演示用刺激呈示装置の試作．上がArduino，下がブレッドボード．

装置はブレッドボードでの試作（図1）の後，シールド上に実装した．シールドとはArduinoの入出力ピン上に直接重ねて接続することのできるアドオン基板で，シールド上の回路・部品によってネットワークやストレージ，センサなどの機能を追加することができる．あらかじめ部品の取り付けられたシールドも多く市販されているが，今回はなにも回路が用意されていないユニバーサル基板タイプのシールドを用い，各部品を配線した．

2　別途電源を用意することで，プログラム転送後にArduinoをコンピュータと分離して単体で動作させることができる．

製作した装置を図２に示す．(A)はArduinoにシールドを取り付けた状態のもの（亀の子のようにArduinoにシールドを重ね合わせて取り付けるためArduinoは下に隠れて見えなくなっている），(B)はシールドをArduinoと分離し裏返したものである．左右両端のピンを差し込むによって，シールドとArduinoは電気的に接続される．

回路図を図３に示す．Arduinoのデジタル入出力ピンのうち，02～10ピンを数値表示部に，11・12ピンをスイッチに，13ピンを刺激光呈示用のLED制御に用いた．

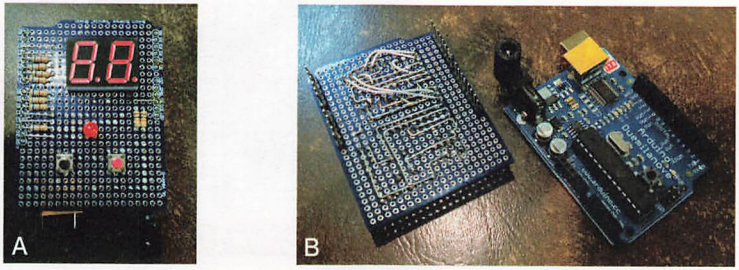

図２．フリッカーテストの演示用刺激呈示装置．(A) Arduinoにシールドを取り付けた状態．上部の7セグLEDが周波数を表示する数値表示部，中央が刺激呈示用LED，下部の2つのボタンが呈示周波数変更用のスイッチ．(B) シールド（左側）を分離し裏返した状態．右側はArduino．ピンの位置を合わせて挿入することでシールドを取り付けることができる．

１．２ ソフトウェア

Arduino言語では，さまざまな処理をsetup()関数とloop()関数の中に記述する．setup()関数は起動時に実行されるもので，その後はloop()関数の記述内容が繰り返される．したがって，呈示のためのピン設定などは事前の設定はsetup()関数に，その後の処理はloop()関数内に記述すればよい．

今回作製したスケッチは付録として末尾に収録した．今回は点滅周期の発生に，Arduino言語が標準で備えているtone()関数を用いた．これは任意の周波数でデューティー比0.5の矩形波を発生させるものである．loop()関数の中で周波数上昇・下降ボタンが押されているかを調べ，押されている場合にはtone()関数で指定する周波数を１Hzずつ変更することとした．

周波数表示に用いた２桁の７セグLEDの表示制御に関しては，GarretLab (2012)をもとにした．使用するLEDに応じて表示桁数を２桁に削減し，アノードコモンタイプ用に修正した上で，使用ピン数を節約するためにDP（小数点）への配線を省略した．

2．演示実験の例

ここではひとつの例としてCFFを極限法により測定する方法を示す．実験では毎試行，事前に決めておいた周波数に実験者がセットした後，一定の距離・向きで装置のLEDを実験参加者に見せて，ちらついて見えるかどうかを評定させる．上昇系列ならばちらつきが消失するまで，下降系列ならばちらつくまで実験者がスイッチを操作して呈示する．各試行の初期呈示周波数は，上昇・下降系列それぞれであらかじめランダムな値を用意しておいて設定する．Microsoft Excelなどで生成しておくとよいだろう．[3] 実験参加者に装置を示す際に呈示周波数を見られてしまうことを避けるために，周波数表示部に覆いをしてから装置を示すようにする．

あるいは，調整法を用いて，実験参加者自身に周波数を変更させてもよい．その場合は，各試行の初期値に実験者がセットし表示部に覆いをした上で，実験参加者に装置を与える．

この装置は安価に作成することができる．今回の例の場合，シールド部

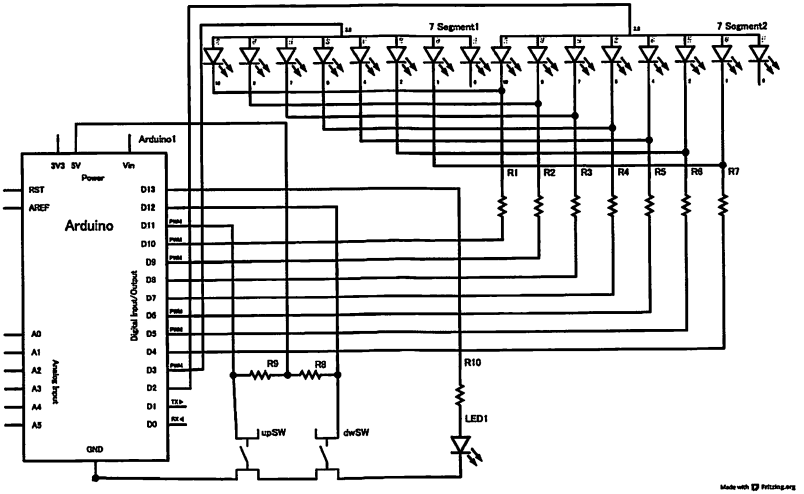

図3．フリッカーテストの演示用刺激呈示装置回路図．図中，R1～R7は680Ω，R8・R9は1KΩ，R10は330Ωである．7セグLEDにはHDSP-K211，LED1にはOSDR5113Aを用いた．

3　たとえば上昇系列の場合，Microsoft Excelの分析ツールに含まれているRANDBETWEENを用い，「=RANDBETWEEN(31,35)」で31～35の範囲のランダムな整数を得ることができる．これを，試行数分だけコピーすればよい．

分の部品代は1,000円程度で，Arduino本体を加えても３～4,000円程度のため，製作の手間はかかるものの安価に多くの台数を用意できる．用意した台数分だけ，同時並行で実験を行えるので，時間的な制約のある実験演習の授業などでも実施しやすいだろう．また，コンピュータに接続せずに刺激を呈示できる点も，クラスや班単位で実験を行う場合には有利である．電源としてはACアダプタの他，電池ボックスとプラグを工作すれば乾電池（例えば９V角型電池の006P）でも動作可能である．

３．他の実験への応用

LEDの点灯，スイッチ・数値表示部の機能はスケッチ（プログラム）によって設定されるため，スケッチを書き換えることにより今回作製したシールドをフリッカーテストとは全く別の実験に用いることもできる．たとえば，数値表示部に数値を呈示し，記銘時に見たものかYES/NO判断を行う再認テストなども，スケッチを修正すれば可能である．記銘時にも装置を用いる場合，記銘時とテスト時で異なるスケッチを使用することになる．それぞれに別個の装置を用意してもよいが，スケッチの転送は１分程度であるため，その度ごとにコンピュータに接続してロードしてもよいだろう．

実験に応じて個別のシールドを作製するのではなく，販売されている既存のシールドを用いる方法もある．２行程度の文字LCDと１～数個のタクトスイッチを載せたシールドや，各種センサとLED・ジョイスティックなどを備えたシールドなど，さまざまなものが市販されているので，これらを利用すれば製作の手間をかけることなく実験を行うことができる．こうしたシールドをベースにして，さらに部品や基板を追加して実験に適したシールドを作製してもよいだろう．

４．実験に使用する場合の留意点

このように，コンピュータに加えて外部装置を必要とする様な実験を検討する場合，Arduinoの利用を検討する価値がある．以下に，Arduinoの使用を検討する際に留意しておいたほうが良い点を列挙する．

４．１ 装置製作のための技術

実際に装置を動作させるためには，プログラミングに加えて装置を作製

するための知識・技術が必要となる．LEDやスピーカなどの刺激呈示用の部品，スイッチなどの反応取得用の部品を回路として構成するだけでなく，この部品を制御するためのスケッチも作成できなければならない．

幸い，こうした回路構成はwebを検索すると大量に見つけることができる．スケッチについてもArduinoのコミュニティ内で豊富なサンプルが公開されており，Arduinoの統合環境にもその多くが収録されている．Arduinoに関する解説書も多数出ているので，参考になるだろう（e.g. CQ出版社，2010，2011；神崎，2012）．

シールド上に装置を作製する場合は，知識に加えてはんだ付けなどの製作技術や道具も必要となる．こうしたことに慣れていないのであれば，シールド上ではなくブレッドボード上でまず試作してみるとよい．ブレッドボードならば，ラジオペンチとニッパー程度を用意するだけでチャレンジできる．はんだ付け不要で，間違えたときに部品を取り付けなおすことも容易なので，製作はずっと容易になる．その後で，たとえば加藤（2007）などを参考にしながらシールド上での実装を図るとよいだろう．

4．2 ディスプレイの呈示能力

コンピュータ実験では，画像を容易に呈示することができるが，シールド上で同じことをするのは容易ではない．Arduinoにどんな表示装置を接続するかによるが，今回の数値表示部で用いた7セグのLEDでは，数値といくつかのアルファベットを呈示するのがせいぜいである．LCDを接続すればより多くの数字や文字を呈示できるようになるが，それでも表示可能な文字数や文字種の多様性などはディスプレイに遠く及ばない．

コンピュータ実験では教示などもディスプレイ上に表示して実験を行うことが多いが，Arduinoを使用する場合は無理にすべてをArduino上で行おうとせず，教示は口頭で行うなどしたほうがよいだろう．あるいは，コンピュータと接続して，教示はコンピュータのディスプレイ上で行うなど，それぞれで役割を分担して実験を進めるとよいだろう．

4．3 プログラム

既述のように，Arduinoのプログラミングは繰り返し実行されるloop()関数を中心としており，実験全体を単一のプログラムで記述するためには，ループを回るたびに状態を確認し，状態に応じた反応をするようスケッチ

を記述する必要があり，プログラム自体が読みにくいものとなってしまう．[4]

これを避けるためには実験のためのプログラムを，たとえば練習試行や本試行ごとといったような形で小分けして，それぞれの段階ごとに転送・実施すればよい．手間は掛かるが，入り組んだプログラムにしなくて済む．

このように，Arduinoを使う場合でも，知識・技能が必要となったり制約があったりする．それでも，コンピュータ単体ではできないような実験を安く手軽に可能にしてくれる点は，大変魅力的である．コンピュータを用いて実験制御を行っている者は，たいていプログラミングも慣れているので，プログラマブルで多様な入出力が可能なArduinoは，実施できる実験の幅を大きく広げるだろう．

References

Arduino (2010). Arduino Uno.
<http://arduino.cc/en/Main/Arduino BoardUno>（2013年１月23日）

CQ出版社 (2010). 電脳Arduinoでちょっと未来を作る（CQ ham radio増刊 マイコンと電子工作No.１）　CQ出版社

CQ出版社 (2011). Arduino工作アイデア集（CQ ham radio増刊 マイコンと電子工作No.６）　CQ出版社

D'Ausilio A., (2012). Arduino: a low-cost multipurpose lab equipment. *Behavior Research Methods*, 44, 305-13. doi: 10.3758/s13428-011-0163-z.

GarretLab (2012). ４桁７セグメントLEDで始めるArduino（Arduinoで遊ぶページ）
<http://garretlab.web.fc２.com/arduino/introduction/beginning_with_７segment_led/index.html>（2013年１月23日）

Gianelli, C., Ranzini, M., Marzocchi, M., Rettre Micheli, L., & Borghi, A. M. (2012). Influence of numerical magnitudes on the free choice of an object position. *Cognitive Processing*, 13, S185-S188. doi: 10.1007/

4　スイッチなどからの入力や，タイマをトリガーとした割り込みを用いる方法もあるが，ここでは述べない．

s10339-012-0483-7.

Huynh, T.-B., Hardy, L., Pezzo, M., & Wilder, O. (2012). The testing and design of an Arduino microcontroller board for the study of proxemics. *University of South Florida St. Petersburg Student Research Journal*, **2**, Issue 1.

神崎 康宏 (2012). Arduinoで計る，測る，量る　CQ出版社

加藤 ただし (2007). 図解 つくる電子回路 正しい工具の使い方，うまく作るコツ（ブルーバックスB-1553）　講談社

北村 英哉・坂本 正浩（編）(2004). パーソナル・コンピュータによる心理学実験入門：誰でもすぐにできるコンピュータ実験　ナカニシヤ出版

水野 りか (2004). Webを介してできる 基礎・認知心理学実験演習　ナカニシヤ出版

西本 武彦（編）(2012). 認知心理学ラボラトリー　弘文堂

酒井 浩二・森下 正修・松本 寛史 (2007). 今すぐ実験！パソコンで認知心理学実験　ナカニシヤ出版

佐藤 雅之 (2007). 臨界融合周波数（篠森 敬三（編）　視覚I－視覚系の構造と初期機能－（講座 感覚・知覚の科学I）　朝倉書店，p. 222.）

Teikari, P., Najjar, R. P., Malkki, H., Knoblauch, K., Dumortier, D., Gronfier C., & Cooper, H. M. (2012). An inexpensive Arduino-based LED stimulator system for vision research. *Journal of Neuroscience Methods*, **211**, 227-236. doi: 10.1016/j.jneumeth.2012.09.012.

和気 典二・長谷川 敬 (1994). ちらつき　（大山 正・今井 省吾・和気 典二（編）新編 感覚・知覚ハンドブック　誠信書房，pp. 338-339.）

■付録

```
/*
 * フリッカー刺激呈示プログラム
 * 7セグLED表示部分についてはGarretLab (2012)をもとに,
 * 2桁・アノードコモンタイプ用に修正した：
 * http://garretlab.web.fc2.com/arduino/introduction/
 * beginning_with_7segment_led/index.html
 * LEDのA-FをArduinoの04-10ピンに,
 * アノードを02, 03ピンに割り当てた.
 * Masaru Takahashi 2013.
*/
const int flickPin = 13;
const int minFreq = 31;
const int maxFreq = 99;
const int upPin = 11;
const int dwPin = 12;
const int waitPeriod = 300;

int currFreq = minFreq;
int prevFreq = currFreq;

int upBtnState = 0;
int dwBtnState = 0;

const int cathode_pins[] = {10,9,8,7,6,5,4};
const int anode_pins[] = {2,3};
const int number_of_cathode_pins = sizeof(cathode_pins) / sizeof(cathode_pins[0]);
const int number_of_anode_pins = sizeof(anode_pins) / sizeof(anode_pins[0]);
int numbers_to_display = 0;

const int digits[] = {
  0b00111111, // 0
  0b00000110, // 1
  0b01011011, // 2
  0b01001111, // 3
```

```
  0b01100110, // 4
  0b01101101, // 5
  0b01111101, // 6
  0b00100111, // 7
  0b01111111, // 8
  0b01101111, // 9
};

// 1桁の数字(n)を表示する
void display_number (int n) {
  for (int i = 0; i < number_of_cathode_pins; i++) {
    digitalWrite(cathode_pins[i], digits[n] & (1 << i) ? LOW : HIGH);
  }
}

// カソードをすべてHIGHにする
void clear_segments() {
  for (int j = 0; j < number_of_cathode_pins; j++) {
    digitalWrite(cathode_pins[j], HIGH);
  }
}

// 数値を表示する
void display_numbers () {
  int n = numbers_to_display;
  for (int i = 0; i < number_of_anode_pins; i++) {
    digitalWrite(anode_pins[i], HIGH);
    display_number(n % 10);
    delayMicroseconds(100);
    clear_segments();
    digitalWrite(anode_pins[i], LOW);
    n = n / 10;
  }
}

void set_numbers(int n) {
```

```
  numbers_to_display = n;
}

void setup() {
  // setup for flicker LED and serial
  pinMode(flickPin, OUTPUT);

  pinMode(upPin, INPUT);
  pinMode (dwPin, INPUT);

  // setup for 7seg LED
  for (int i = 0; i < number_of_cathode_pins; i++) {
    // cathode_pinsを出力モードに設定する
    pinMode(cathode_pins[i], OUTPUT);
  }
  for (int i = 0; i < number_of_anode_pins; i++) {
    // anode_pinを出力モードに設定する
    pinMode(anode_pins[i], OUTPUT);
    digitalWrite(anode_pins[i], LOW);
  }
  set_numbers(currFreq);
  tone(flickPin, currFreq);
}

void loop() {
  upBtnState = digitalRead(upPin);
  dwBtnState = digitalRead(dwPin);

  if ((upBtnState == LOW) && (currFreq < maxFreq)) {
    noTone(flickPin);
    prevFreq = currFreq;
    currFreq++;
    set_numbers(currFreq);
    delay(waitPeriod);
    tone(flickPin, currFreq);
  }
```

```
  if ((dwBtnState == LOW) && (currFreq > minFreq)) {
    noTone(flickPin);
    prevFreq = currFreq;
    currFreq--;
    set_numbers(currFreq);
    delay(waitPeriod);
    tone(flickPin, currFreq);
  }
  display_numbers();
}
```