マイコンを用いたハードウェアの制御 (2)

フィールドロボティクス分野

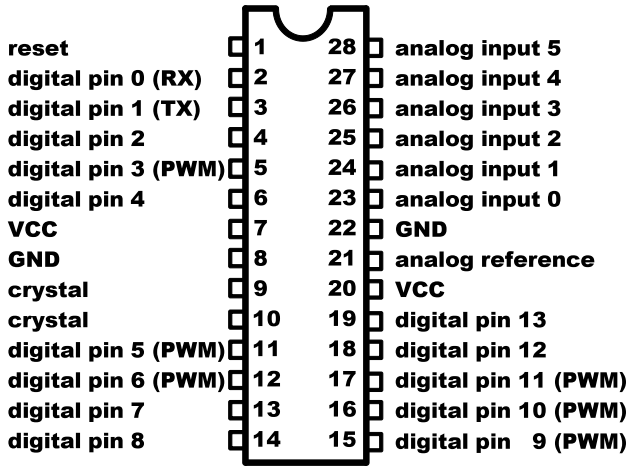

図 2(再掲)　ATmega328 の Arduino ピン配置

課題

課題 6　7 セグメント LED 表示器

７セグメント LED とは、7 つの LED を用いてアラビア数字を表現する表示器で、今回は４ケタのものを使用する。発光ダイオードを使用しているので、アノード側をまとめ、カソード側を個別に ON-OFF するかにするか､カソード側をまとめてアノード側を個別に ON-OFF するかによって 2 種類存在するが、今回は図 2 にあるような、アノード側をまとめたアノードコモン型を使用する。

課題 6-1　一桁点灯

図 4 のピン配置図を参考に、表示器左側にある電流制限抵抗(680Ω)にジャンパーケーブルで接続する。

ピンの番号は通常、左下から右下、右上から左上に順に振られるので、左下から 1,2,3,4,5,6,右上から 7,8,9,10,11,12 である。

まず右端(DIG4)のみ接続する。右端のアノードコモン(6 番)を+5V に接続する。

「a」(11 番)を左端の抵抗の片側に接続し、別のジャンパー線を 0V に接続、これを先ほどの抵抗の反対側に接触させ 7 セグの「a」が点灯することを確認する。

同様に「b」(7番)を左端から 2 番目の抵抗の下側に接続し、先ほどの 0V に接続したジャンパー線をこの抵抗の別端に接触させ「b」のセグメントが点灯することを確認、以下 c,d,e,f,g を点灯の確認しながら接続していく。

抵抗の、先ほど確認のために 0V を接触させていた側をマイコンに接続していく。「a」を ditgital pin 2、「b」を ditgital pin 3、「c」を ditgital pin 4、「d」を ditgital pin 5、「e」を ditgital pin 6、「f」を ditgital pin 7、「g」を ditgital pin 8 へと接続する。(digital pin 0,dititalpin1 を用いてもよいのだが、パーソナルコンピュータとの通信に使用されているため、今回は空けておく。)

File->New で新しいウインドウを開きサンプルプログラム Blink を参考に 0～9 までの文字を 1 秒間隔で表示するプログラムを作成する。ただし、アノードがまとまって+5V に接続されているので、カソードの接続されているピン

を 0V(LOW)にすることで LED が点灯することに注意すること。 また、以降の課題のために、一桁の数字を渡すとその数字を表示するサブルーチンを作成すること。

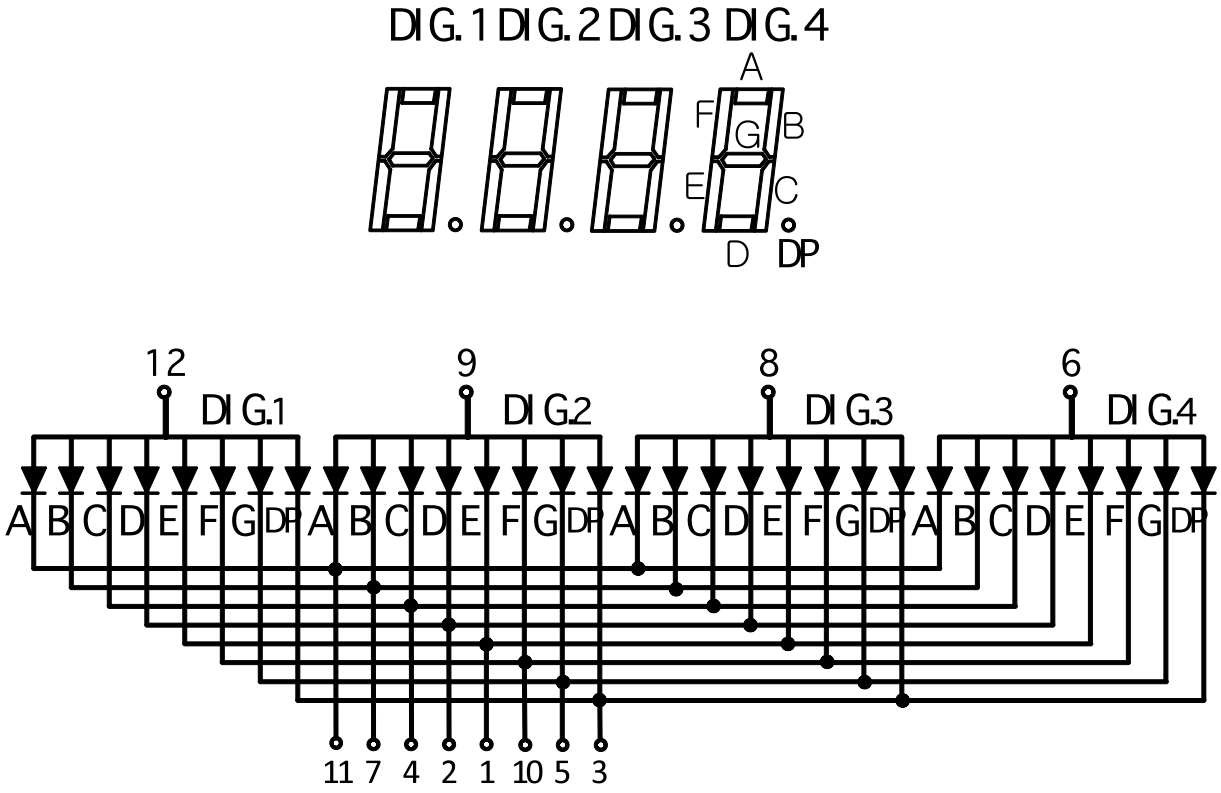

図 4　アノードコモン 4 桁 7 セグ LED のピン配置

課題 6-2 四桁点灯

残りの 7 セグメントを点灯させる場合、マイコンのピンをもう 7×3=21 ピン使用すればよいのだが、ピンの数には限りがあるので(今回の場合は analog input 用のピンを digital out に変更して用いたとしても 19 ピン)、多桁表示にはダイナミック表示といわれる方法を用いる。

7 セグメントのダイナミック表示では、一桁ずつ LED を点灯させ、これを高速で切り換えることによって同時に光っているように人間の目に錯覚させる方法である。人間だから錯覚するわけであってカメラなどで高速シャッター撮影すると、一桁しか写らない。

DIG4 のアノードコモンを+5V から抜き、DIG3 のアノードコモンを+5V に繋いで、課題 3-1 のプログラムを実行させ、右側が正しく点灯することを確認する。

このアノードコモンをマイコンから ON(+5V)して、点灯する桁を制御するわけだが、7 セグメント LED のアノードコモンをマイコンに直接つなぐと、最大 LED7 個分の電流を流さなければならないことからマイコンにとって少々負担が大きい。そこで今回はトランジスタでスイッチングすることにする。

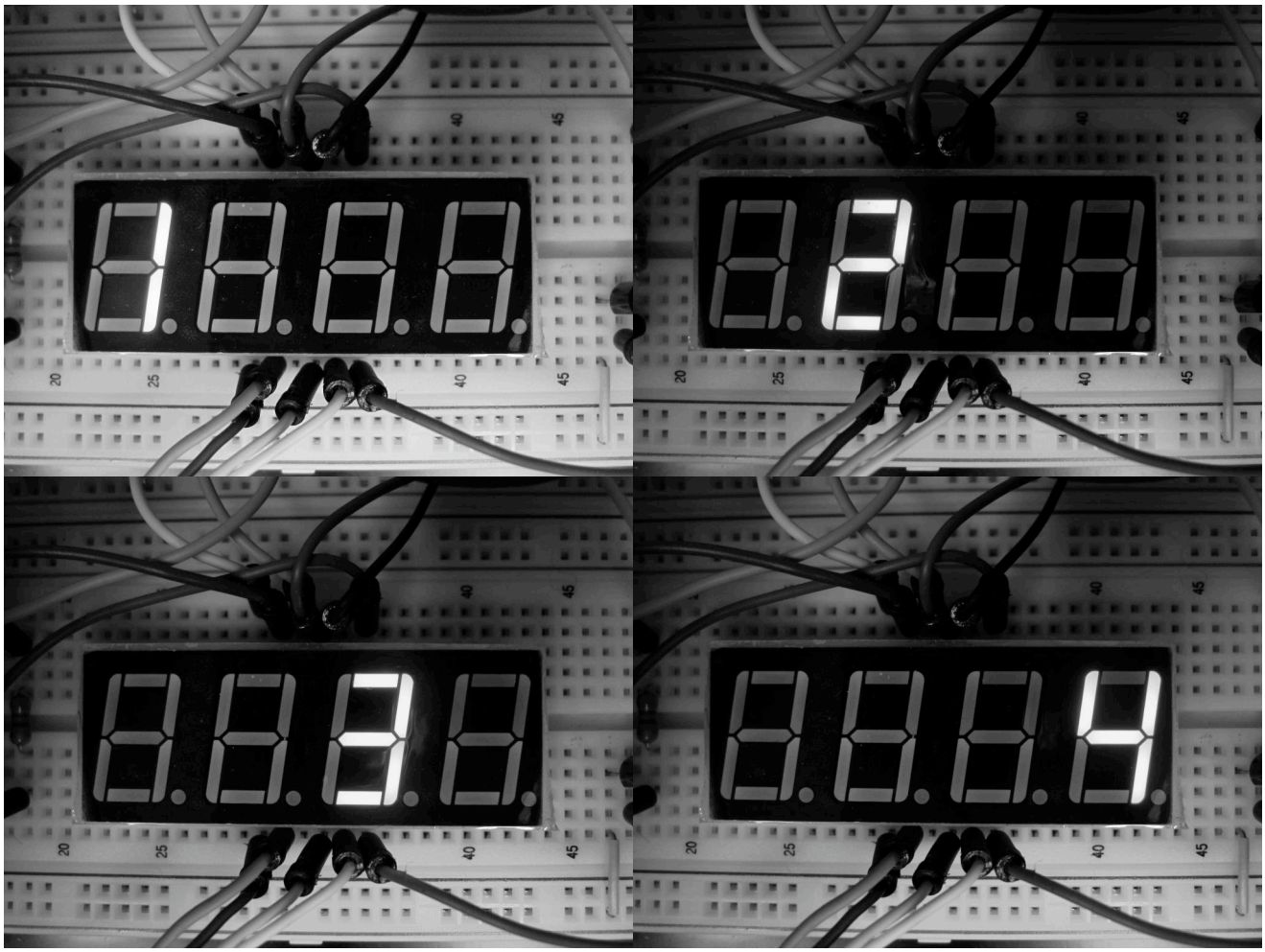

図 5　ダイナミック点灯

1234 と表示する場合、順に点灯してゆき、高速に切り替えることで 4 ケタが同時に表示されているように見せる。

7 セグ表示器右側のトランジスタ 2SA1015 は正面(文字が書いてある面)からみて左側からエミッタ(E)、コレクタ(C)、ベース(B)となっている。**それぞれのトランジスタのエミッタが+5V、ベースが 5.1KΩ 抵抗につながっていることを確認する。**

右端トランジスタのコレクタに DIG4 のアノードコモンを、順に DIG3,DIG2,DIG1 アノードコモンをそれぞれのトランジスタのコレクタに接続。

右端のトランジスタのベースにつながっている 5.1KΩ のトランジスタとは別側を digital pin 13,残りのトランジスタも順に digital pin 12, digital pin 11, digital pin 10 へと接続してゆく。 CPU のピン配置は図 2 参照

課題 2-2 割り込みによる交互点滅のプログラムを改造し、digital pin 2～8 には数字「5」を出力、digital pin 10 digital pin 11 digital pin 12 digital pin 13 を 1 つずつ順に LOW(0V)にすることで、7 セグ表示器のそれぞれの桁に「5」が表示されることを確認し、交互点滅切り替え時間を短くすることで、同時に光っているように見えることを確認する。

4 ケタで異なる数字を表示するようにプログラムを改造する。（HIGH,ではなく LOW で点灯するのは、PNP 型トラ

ンジスタによりOV と5V が反転しているため。）

ヒント:

まずは固定した数字(1234 など)を表示するプログラムを作成してみる。 時間割込が発生するたびに、

(1) Dig.1 だけを ON (pin13->LOW, pin12->HIGH, pin11->HIGH, pin10->HIGH), pin2,3,4,5,6,7,8 から 1 を表示する.

(2) Dig2 だけを ON, pin2-8 で 2 を表示する

(3) Dig3 だけを ON, pin2-8 で 4 を表示する

(4) Dig4 だけを ON, pin2-8 で 4 を表示する

以上を繰り返すようにすればよい。

つづいて0. 2秒ごとに（delay 関数を使うとよい）9999から 1 ずつカウントダウンして表示するプログラムを作成せよ。

ヒント:

ある 4 ケタの数字から 1 桁ずつ数字をとりだすには、x/1000%10, x/100%10, x/10%10, x%10 とすれば、それぞれの桁の数字になる。

余裕のある人は 0～F の 16 進数を表示するように改造し 00～FFFF までの表示をおこなうプログラムを作成してみる。

補足: さらに pin を減らすにはシフトレジスタを使う。

課題 7　AD 変換

AD 変換とは電圧を計測しその値を(アナログ値)を数値(デジタル値)に変換することで、今回使用している ATMEGA328 では、Analog0～Analog5 までの 6 つのアナログ入力を 10Bit の分解能でデジタル値に変換できる。入力された電圧は、0V を 0、参照電圧(通常 5V)を 1024(10BIT),とする値に変換されてコンピュータに読み取られる。たとえば読み取られた値が 300 であれば、その時の電圧は 5V/1024×300＝1.46V である。

課題 7-1 可変抵抗器の電圧の読み取り

ブレッドボード上のスライド式の可変抵抗器には左に３、右に３の電極がある、このうち左の右端の電極が 0V、右の真ん中の電極が 5V に接続されていることを確認する。

つぎに右側の左端の電極から、ジャンパーケーブルで Analog0 へ接続する。

現在 digital pin13 には DIG.4 のアノードコモンがつながっているが、気にせずそのまま LED 赤のアノードに接続、LED 赤のカソードは 680Ω の抵抗を通して 0V に接続する。

これで http://arduino.cc/en/Tutorial/AnalogInput の Schematic にあるのと同じ接続になる。

Arduino 開発環境の File->Examples->Analog->AnalogInput を選択しサンプルプログラムを読み込む。

プログラムを読んで理解する。

analogRead(A0)　は AnalogPin0 の電圧を AD 変換して読み取り、その値を返す。

課題 7-2 AD 変換値の 7 セグ LED への出力

AnalogInput を改造し、AD 変換した値を 7 セグ LED に出力するプログラムを作成する。

課題 7-3 距離センサによる距離の計測

可変抵抗器からのジャンパーケーブルを取り除く。

距離計（シャープ GP2D12）のケーブル 3 本のうち、赤を５V、黒を０V に接続、もう一本を analog input 0 に接続する。

距離計の距離-出力電圧の関係は Web を検索してメーカーのデーターシートをみつけること。グラフの双曲線風の部分のみ(3cm-40cm)をもちいて読み取った電圧値から距離に換算し、その値を 0.1mm 単位で 7 セグ LED に出力するプログラムを作成する。処理は一度ではなく無限に連続して行うようにする。

課題 8　PC とのシリアル通信

Arduino はシリアル通信を通して PC で作成したプログラムを転送して実行するわけだが、プログラム実行中はシリアル通信は空いているので、これを用いて PC と通信を行うことができる。

課題 8-1　可変抵抗器の電圧出力をパーソナルコンピュータに送信する

AD 変換の結果を、マイコンのプログラム転送に用いていたシリアル通信を通してパーソナルコンピュータに送信する。

パーソナルコンピュータの Arduino の File->Example->Analog->AnalogInSerial をロードする。

プログラムを読んで理解する。

Serial.begin(baudrate) はシリアル通信を開始する関数で、baudrate で通信速度を指定する。今回は 9600baud で使用する。(baud は人名だが、なぜか小文字)

Serial.println(Value)は Value を数値のテキストデータとして送信し、改行信号も送信する。

プログラムをコンパイル、転送、実行し、Arduino の  ボタンを押しシリアルモニターを起動する。

シリアルモニターにマイクロコンピュータからの送信データが受信されていることを確認する。

課題 8-2 可変抵抗器の電圧出力をパーソナルコンピュータに送信し、その値に従いアニメーションを表示する。

シリアルモニターを終了する。

パーソナルコンピュータのデスクトップ上の Processing フォルダ内の processing.exe を実行する。

Arduino とそっくりな Window が開くので、Arduino と混同しないように注意すること。

File->Examples->HardSeigyo->_2_1 でプログラムをロードする。

Arduino ではプログラムはほぼ C,または C++であったが、Processing ではプログラムはほぼ JAVA なので少し様子がことなるが、JAVA 自身が、ほぼ C なので内容を理解できると思う。

import processing.sreial.*; でシリアル通信のライブラリをインポートする。

Arduino 同様 setup()が起動時に一度だけ実行される。size(xsize,yzize);で横 xsize,縦 ysize の描画用ウインドウを用意する。

comPort=new Serial(this,ポート名,baudrate);でシリアルポートクラスのインスタンスを得る。

このポート名を Aruduino の Tool->Serial Port で指定した COM ポート名に書き換えること。

draw()は Arduino における loop()同様、実行中繰り返し呼び出され実行される。(processing では draw()とは別に loop()を書くこともできる。)

background(color), バックグラウンドの色を設定する、8 ビットで指定した場合は白黒で 255 は白。

ellpse(x,y,xsize,ysize); 中心座標 x,y x 軸長 xsize,y 軸長 ysize の楕円を表示する。

serialEvent(Serial comPort) comPort に受信が発生すると前回の実験での割り込みのように起動される。

readStringUntil(value) 受信した文字列データを value まで読み込む。Arduino からの送信は改行で終了している

ので CR+LF 即ち 10 進数で 13,10 つまり 10 がくれば行の終了であるので 10 が来るまで文字列を読み込む。
trim(stringdata) stringdata についている文字列データ以外のもの(改行指定など)を取り除く。
int(data) data を int 型に変換する。

 ボタンを押し、プログラムを実行してみる。

課題 9　サーボモータの制御

課題 9-１　サーボモータのスイープ

ラジコン用のサーボモータを制御する。 ラジコン用のモータはPWM制御によって指定の角度になるようになっている。 通常 High 時間が 1.5ms で中央位置、1.0ms～2.0ms に変化させることで角度を指定するようになっており，サイクルは 50Hz（20ms）が標準とされている。

サーボモータのコネクタの茶色のケーブルと黒ジャンパーケーブルを接続し、0V に接続。
赤ケーブルを赤ジャンパーケーブルに接続し5V に接続、残りの一本(信号線)を Digital Pin 9 に接続する。
Arduino の File->Examples->Servo->Sweep を選択し、プログラムをロードする。
プログラムを読んで理解する。
myservo.attach(9); で、先に宣言した Servo myservo で使用する PIN を 9 番に指定する。
myservo.write(pos);でピン 9 に接続されたサーボを pos 角(0～180)に動かす。
実際にはサーボモータには個体差があり、PWM での High の時間と角度の関係は一つ一つ異なるため先の attach()で atach(pin 番号,0 度の時の PWM High 時間(μs),180 度の時の PWM Low 時間(μs)) で調整する必要がある。

課題 9-2　サーボモータと測距センサの組み合わせ

サーボモータで測距センサを 10 度ごとに動かし、距離を計測してシリアル通信でパーソナルコンピュータに送信する。 サーボモータの静定とセンサの出力安定に 1 秒(1000ms)を見込んでおく。
シリアル通信は、角度：AD 変換値<改行>　または、角度：距離<改行> とする。
Serial.print()が改行せずに送信、Serial.println()が改行して送信する関数なので
Serial.print(角度);
Serial.print(':');
Serial.println(距離);
とすればよい
先ほどの Sweep の内容を新しいウインドウにコピーし、プログラムを少々変更して上記を実現するプログラムを作る。 課題 1-4～1-6 あたりから、コードを引用すればよい。
シリアルモニタで正しく動作していることを確認したら、シリアルモニタを終了し、Processing の File->Examples->HardSeigyo->_2_2 をロードし、実行させて動作を確認する。 今回の測距モジュールは計測相手が液晶ディスプレイでは正しい値を返さないので注意すること。