マイコンを用いたハードウェアの制御 (1)

フィールドロボティクス分野

はじめに

コンピュータの基本的な動作および外部機器の制御方法を理解するために、二週にわたってマイクロコンピュータを用いた LED(発光ダイオード)の点灯やサーボモータの制御などをおこなう。
マイクロコンピュータには AVR 社の ATmega328 を用い、開発環境には Arduino を使用し、回路の制作にはソルダーレスブレッドボードを使用する。

使用するマイクロコンピュータについて

ATmega328 は命令 16 ビット長、データ 8 ビット長の 1 チップマイクロコンピューターで 1 チップ上に CPU,EEPROM,RAM,パラレル入出力インターフェース、シリアルインターフェース、AD 変換器、PWM コントローラ、タイマ、割り込みコントローラ等を搭載している。

開発環境について

今回は開発環境として Arduino を使用する。Arduino はパーソナルコンピュータ上のプログラム作成機能と、マイコン上のブートローダから成り、パーソナルコンピュータ上のプログラム作成機能は各種の便利なライブラリが使用可能で、比較的簡単に機器制御ができるようになっている。
開発に用いられるのは Arduino 独自の言語だが、ほぼ C、C++言語である。
作成されたプログラムはパーソナルコンピュータ上で機械語にコンパイルされ、シリアル通信を通してマイコン上のブートプログラムに送信される。
ブートプログラムは受け取ったプログラムを自身の ROM に書き込み、実行するという仕組みになっている。

ソルダーレスブレッドボードについて

今回電子回路を作成するにあたってソルダーレスブレッドボードを使用する。ソルダーレスブレッドボードは、はんだ付けをせずに試験的に回路を作成する場合に使用される。
今回使用するブレッドボードは横長の状態で見た場合左右にひかれた青線、赤線のひかれた部分の穴は電源用で横向きにすべて電気的につながっている。またその他の穴は縦方向に 5 穴 1 セットとして電気的につながっている。また電気的につながっていない穴同士をつなぐにはジャンパーケーブルを使用する。

課題

課題 1 デジタル出力による LED の点滅

デジタル出力

デジタル出力とはマイクロコンピュータのピンの電圧を HIGH(TTL の場合 5V)か LOW(0V)かを設定する仕組みである。

<u>課題 1-1 単に、LED を点灯させる</u>

LED とは Light Emitting Diode の略で日本語では発光ダイオードと訳される。アノードからカソードに向けて電流を流すと特定の波長の光を発生する。(今回は 565nm、636nm のものを使用)　電球と異なり、一定以上の電圧を与えないと発光せず(今回は 1.9V のものを使用する)、それ以上の電圧を与えると電流が過剰に流れるため、今回のように 5V の電圧を用いて発光させようとする場合には、抵抗などを用いて電流を制限する必要がある。

　どちらがアノード(+)でどちらがカソード(−)かはピン(足)の長さで判断するのだが(長いほうが＋)今回実験で使用する発光ダイオードは足がすでに切りそろえられている可能性があるので、図 1 を参考に透明な樹脂内部の金属部分の形状で判断する。

ブレッドボードに 5V の電源アダプタを接続する。

ブレッドボード上の赤 LED のアノードと＋5V、カソードにつながった 680Ωの抵抗の LED とは別の側と 0V を接続すると LED が点灯することを確認する。また、LED を逆に刺しても点灯しないことを確認する

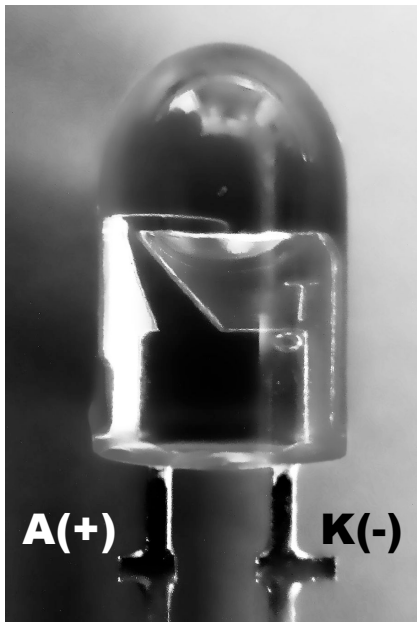

図 1　LED（発光ダイオード）

<u>課題 1-2 マイコンにより LED を点滅させる。</u>

赤 LED のアノードを+5V から抜き、digital pin 13 に接続する。(ピンは位置は図 2 参照)

Digital pin をマイコンのプログラムによって一定時間ごとに 0V と+5V に変化させることで LED を点滅させることができるはずである。

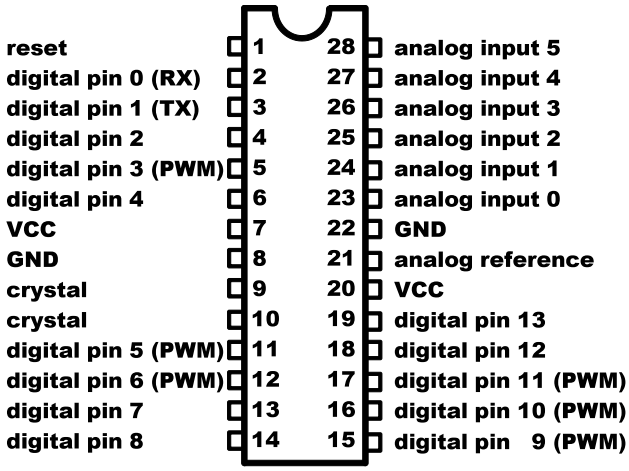

図 2　ATmega328 の Arduino ピン配置

ブレッドボードと PC を USB ケーブルで接続する。
PC 側で Arduino.exe を起動する。
Tools -> Board で Arduino Duemilanove or Nano w/ ATmega328 が選択されていることを確認する。
File -> Examples -> Digital -> Blink でサンプルプログラム Blink を開く
表示されたプログラムを確認する。C 言語同様/* */で囲まれた部分と//から行末まで(C++と互換)は、コメントである。
C 言語では main()関数が実行されるが、Arduino では main()関数は使用せず setup()関数が起動時に 1 回だけ実行されたのち loop()関数が無限ループとして実行されるので、setup()内で各種の設定を行い、loop()内で実際の処理をおこなう。
setup()関数、loop()関数は返値なしの void 型で宣言し、もちろん値も渡さないので()内も void である。
使用されている関数:
pinMode(ピン番号,OUTPUT か INPUT) によって指定番号のピンを出力に用いるか、入力に用いるかを決定する。
digitalWrite(ピン番号,HIGH か LOW) で出力に指定したピンを+5V(HIGH)か 0V(LOW)にする。
delay(ミリ秒数) 指定時間何もしない。

 ボタンを押すか、メニューから、Sketch -> Verify/Comple あるいは Control+R キーによってプログラムがコンパイルされる。エラーがある場合には画面下部に赤文字で表示されるが、サンプルプログラムなのでエラーは発生せず、Done Compiling.と表示される。

 ボタンを押すか、メニューから、File -> Upload to I/O Board あるいは Control+U キーによってコンパイルされたプログラムが、ブレッドボード上のマイコンに送信される。
コンパイルが済んでいなければ、自動的にコンパイルされてから送信される。
ボード上のマイコンはプログラムを受信すると、自身のプログラム ROM 上に受信したプログラムを書き込み、その後そのプログラムを実行する。
LED が点滅することを確認したら、delay 関数に渡すミリ秒数を変更して、点滅速度が変化することを確認してみる。

課題 1-3 LED の交互点滅
緑色の LED のアノードを digital pin 12 に接続する。ピンの位置は図 2 を参照。LED のカソードに接続された抵抗の LED とは別の側を 0V に接続する。
File -> New または Conrtol+N キーを押し新しいウインドウを開く。
Blink のウインドウ内のテキストをすべて新しいウインドウにコピーする。
Blink のウインドウを Close する。
新しいウインドウのプログラムを改造して、赤と緑の LED が交互に発光するようなプログラムを作成する。
これをプログラム側ではなく回路側で行うには、緑 LED のアノードを +5V に接続し、カソードにつながった抵抗の反対側を digital pin 13 (赤 LED がつながっているのと同じ pin)に接続すれば実現できるはずである。このように、回路によって pin が +5V で点灯か 0V で点灯かを決めることができる

課題 2 タイマ割り込み

割り込みとは、通常の作業(今回の場合は loop()関数内のプログラム)を実行中に、特定のきっかけで別作業を行い、元の作業に戻る仕組みである。

今回使用するのはタイマ割り込みで、loop()内での作業とは別に一定時間ごとに別作業を行うというものである。

通常マイコンのプログラミングでタイマ割り込みを使用する場合は煩雑な設定が必要だが、Arduino には MsTimer2 というライブラリが用意されており細かな設定はできないが、簡単に使用することができる。

課題 2-1 サンプルプログラムの実行

File->Example->MsTime2->FlashLed で FlashLed を開き、プログラムを読む。

#include <MsTimer2.h>　割り込みを利用するライブラリ MsTimer2 の関数を利用ためにインクルードする。

MsTimer2::set(ミリ秒数,関数名)指定したミリ秒に一度、関数名(ポインタ)の関数を実行する。ここでは 500 ミリ秒に一度 flash()を実行する。

MsTimer2::start() MsTimer2::set で設定した動作を開始する。

MsTimer2::stop()例題には出てこないが、Timer 動作を停止する。

プログラムの内容を理解したら、コンパイルして転送し、LED が点滅することを確認する。

割り込みプログラム作成時の注意事項：

割り込み処理と通常処理の間で変数のやり取りをする場合は、使用する変数の宣言時に volatile 修飾子を付ける必要がある。たとえば int 型の変数 foo を割り込み処理と通常処理の両方から用いたい場合は、

volatile int foo;

のように宣言する。

課題 2-2 交互点滅

File->New で新しいウインドウを開き、FlashLed の内容をすべてコピー、ペーストし、FlashLed のウインドウはクローズする。

Blink の時と同様に、交互に点滅するように書き加えてみる。

課題 2-3 割り込みと delay の比較

FlashLed では loop()内は空なので、loop()内に課題 1 の Blink のプログラムを書き加え実行させてみる。ただし光らせる LED は Digital Pin 12 に接続した緑色 LED とし、delay のミリ秒は 500 として FlashLed の割り込み時間と同じにする。

コンパイル、転送し、赤と緑の点滅時間を比較する。

課題 3　デジタル入力によるタクトスイッチの読み取り

デジタル入力はデジタル出力とは逆に、入力用ピンにかかっている電圧が LOW であるか HIGH であるかを読み取る仕組みである。

課題 3-1 Button

発光ダイオードの近くにあるタクトスイッチを用いてデジタル入力を行う。タクトスイッチの一方の接点は＋５V に接続されており、もう一方は 10KΩの抵抗を介して 0V に接続されている。このため、抵抗がつながっているタクトスイッチの脚の部分は、スイッチが押されない状態では 0V、スイッチが押されると 5V となる。

File->Example->Digital->Button で Button を開きプログラムを読んで理解する。

digitalReal(pin 番号) 関数が Pin の状態を読み取って+5V であれば 1 を返し、0V であれば 0 を返す。

コメントの内容に従って、Digital Pin 2 にタクトスイッチの抵抗側を接続する。また、Digital Pin 13 に LED が接続されていることを確認する。

Button プログラムをコンパイル、マイコン側に転送してボタンを押して LED が点灯することを確認する。

課題 3-3　トグル

button のプログラムの内容をコピーし、新たに Toggle というプログラムを作成してペーストする。

これを改造して、一度ボタンを押すと LED が点灯し、もう一度押すと消灯するプログラムを作成せよ。

課題 4 外部割込み

タイマ割り込みは一定時間が経過するごとに割り込みを発生させて、それをきっかけに特定の動作を行う仕組みであったが、この”きっかけ”をタイマなどのコンピュータ内部に組み込まれたデバイスではなく、外部のデバイスから行う仕組みが外部割込みで、通常ピンの電圧状態やその変化によって割り込みが発生するようになっている

課題 4-1 外部割込みを用いた入力の読み取り

Button や debounce のプログラムではメインルーチンの中で一度だけボタンの状態を読み込むわけだが、ほかの仕事が忙しくて、ボタンを押されたときにたまたまボタンの状態を読み込むタイミングが回ってこなかった、などということが起こりうる。そこで常にピンの状態を見張り、条件が合えば割り込みを発生させて所定の仕事を行わせるという、外部割込みの仕組みを使用する。

課題 2 ではマイコン内のタイマが時刻を告げた時に割り込みが発生し、指定したルーチンが実行されたわけだが、マイコン外部からの信号の入力によっても割り込みが発生する。

新しくウインドウを開きそこに以下のプログラムを入力する。

```
int pin = 13;
volatile int state = LOW;

void setup() {
  pinMode(pin, OUTPUT);
  attachInterrupt(0, blink, RISING);
```

```
}

void loop() {
}

void blink() {
  state = !state;
  digitalWrite(pin, state);
}
```

プログラムを読んで理解する。

attachInterrupt(割り込み番号,呼び出す関数,トリガの種類)

割り込み番号は digital pin 2 が 0, digital pin3 が 1 である(INT0,INT1)

トリガの種類は、そのピンの状態がどうであれば割り込みを発生させるかで、

LOW: ピンが LOW の時

CHANGE; 状態が変化したとき

RISING: LOW から HIGH に変わったとき

FALLING: HIGH から LOW に変わったとき

の 4 種類がある。

attachInterrupt で設定された割り込みは **detachInterrupt**(割り込み番号)で停止させることができる。

またすべての割り込みを禁止する場合は **noInterrupts**()関数を用い、再び割り込みを許可する場合は **interrupts**()関数を用いる

プログラムをコンパイル転送し、toggle のような動作をするかどうか確認せよ。

また、うまく toggle しない場合があることも確認する。

うまく toggle しないのは Debounce のところで述べたように、チャタリングが起きて割り込みが何度も発生しているからである。

上のプログラムを改造して Debounce のようにチャタリング対策を付け加えよ

(書き方はいろいろあるが、先の MsTimer2 を使うのがスマート。)

課題 5　PWM

ここまででではコンピュータからの出力は 0V か 5V かのいずれかであったが、コンピュータからアナログのデバイスを制御したい時などアナログの出力が必要となる場合がある。このような場合デジタルアナログ変換器(DA 変換器)が用いられる。 ATMega328 には DA 変換器は内蔵されていないが、PWM 出力機能が内蔵されており、Arduino から簡単に使用することができる。 PWM とは Pulse-width Modulation(パルス幅変調)の略で連続的に発生させた矩形波の HIGH と LOW の時間割合を図 3 のように変化させるしくみで、制御される対象にとって十分に高い周波数の矩形波を用いることで結果的に DA 変換機と同等の働きをさせることができる。

課題 5-1 LED の明滅

今回はこの仕組みを用いて LED を徐々に明るくまた、暗く光らせてみる。

File->Examples->Analog->Fading でサンプルプログラム Fading を読み込む。

Fading のコメント欄にしたがって Digital Pin 9 を LED のアノードへ繋ぎ、カソードから抵抗を通して GND(OV)へ結線する。ピン配置については図 1 を参照すること。

プログラムを読んで内容を理解する。

analogWrite(pin 番号、数値)で指定 Pin から PWM が出力される。PWM 出力可能は Pin は図1で PWM と記されている Pin のみである。出力される PWM 信号は HIGH,LOW の比が「(指定値) 対 (255-指定値)」となり、周波数は Arduino では Pin 3,9,10,11 で 490Hz、Pin 5,6 で 980Hzである(変更は可能)。

プログラムをコンパイル転送し、動作を確認する。

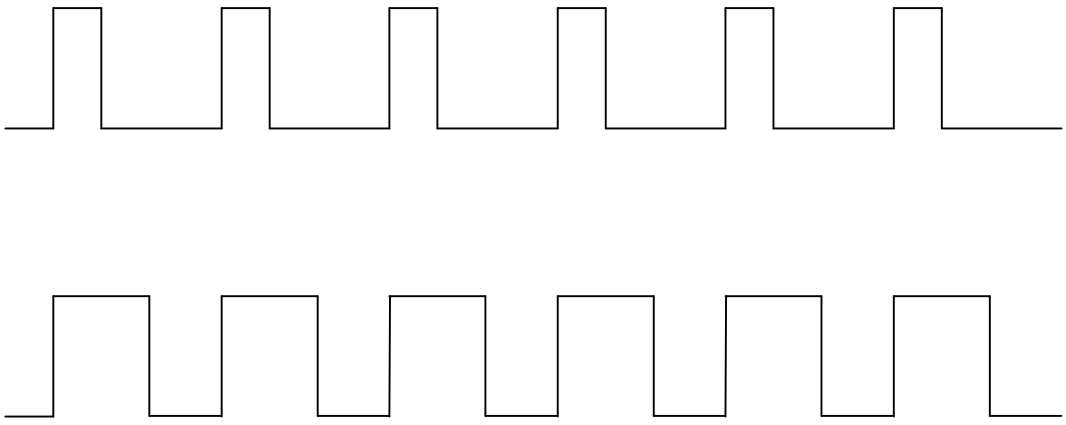

図 3 PWM の例

課題 5-2 LED 明滅時間の変更

新しいファイルを作り、Fading をコピーして、LED を 2 つ、同時に異なる時間間隔で明滅させるプログラムに変更せよ。

前に使用した MsTimer2 を使用するとよい。ただし、Timer2 を使用する場合は PIN 3 と PIN 11 を PWM として使用できないので別の PIN を使用すること。

メモ