2.4　割り込み

マイコンには任意のタスク処理を実行しているときに，何らかの要因により強制的に別の処理を実行する「割り込み」という仕組みがある。割り込みを使用すると，マイコンがタスク処理を実行中に，より重要または緊急性の高い処理が要求されたとき，実行している処理を中断して優先的に割り込み処理を行える。

実験では，割り込みとして，マイコン内蔵のタイマ機能により一定の時間間隔で生成される「タイマ割り込み」，およびマイコンの外部に接続されたデバイスの状態変化によって割り込みが発生する「外部割り込み」を扱う。いずれの割り込みもマイコンの通常処理とは無関係（非同期）に発生する。マイコンのプログラムをより効率的に実行するためには，割り込み機能の原理の理解と実装方法を知ることは重要である。

実験目標：

・割り込みの動作原理を知る。
・ポーリングと割り込みの処理の違いを理解する。
・タイマ割り込みの動作原理を理解して使いこなせる。
・外部割り込みの動作原理を理解して使いこなせる。

2.4.1　割り込みとは

割り込みの例として，スマートフォンを例に挙げよう。スマートフォンは，電話以外にも様々な機能が実装された携帯できる高機能な組み込み機器である。スマートフォンの心臓部には低電圧で動作するマイコンが使用されている。スマートフォンのバッテリィを長持ちさせるためには，割り込み処理の実装が必須である。

割り込み処理の一例として，スマートフォンでメールを受信することを考える。スマートフォンで“メールを受信する”には，メールが基地局に届いたかどうか確認する必要がある。通常，あなたに届いたメールは携帯の基地局にあり，一定の時間間隔で同期が取られ，スマートフォンにメールが取り込まれる。メールの受信確認の頻度（5 分，10 分，15 分，30 分など）は，通常メニューよりソフトウェアで設定変更できる。このメールの受信確認の流れを，これまでに実験を行ったスイッチの状態を確認するプログラムと同様の処理で行うと図 2.55 のようになる。縦軸はイベントの発生状態，横軸は経過時間（分）である。メールの確認というイベント（タスク）が数分間の間に何度となく実行されていることが確認できる。このような処理は「**ポーリング処理**」と呼ばれる。このときメールの受信確認のために，何らかの状態判定や演算，基地局への無

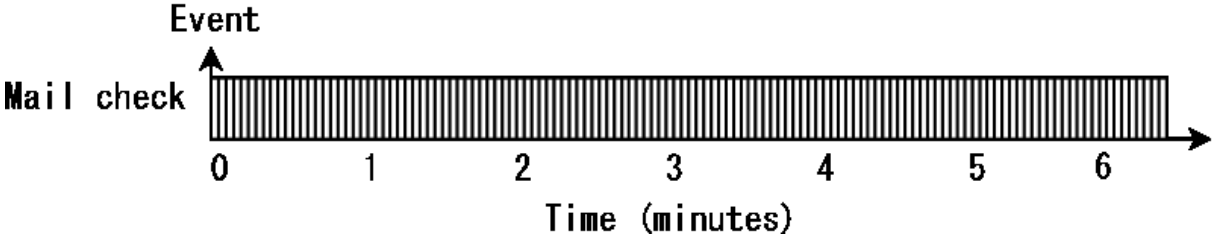

図 2.55　スマートフォンのメールの確認（ポーリング処理）

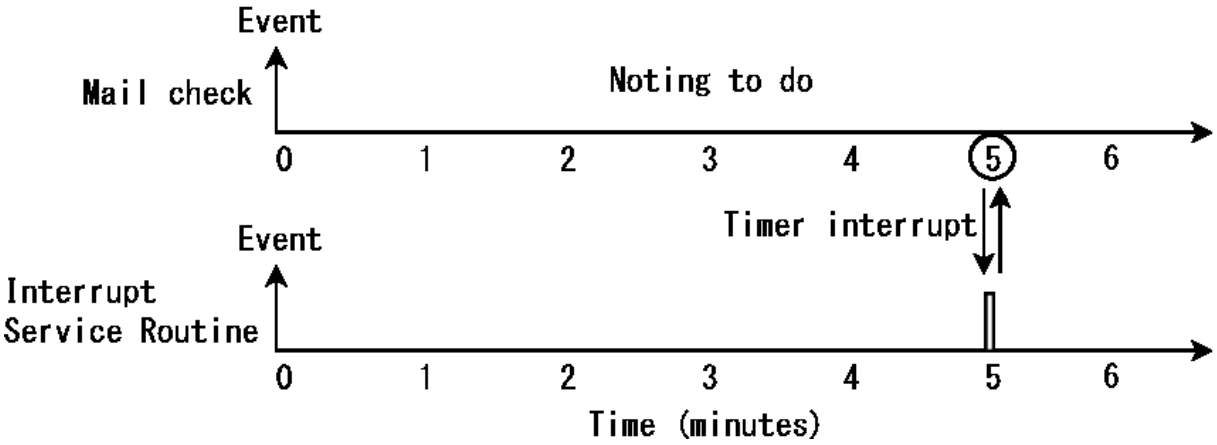

図 2.56　スマートフォンのメールの受信確認（タイマ割り込み）

線通信などを行う必要があり相当量のエネルギーが消費される。1 つのタスクをメールの受信確認に要するエネルギーとすると，タスクの総実行回数が消費エネルギーとみなすことができる。

スマートフォンはバッテリィで動作しているので，なるべく余分なところでエネルギーを使いたくない。そこで，タスクの実行回数を減らすために，タイマ割り込み(Timer Interrupt)を用いる。タイマ割り込みは，マイコンのプログラムを実行しているシステムクロックをもとに生成され，一定の時間間隔で割り込みを発生させることができる。タイマ割り込みを 5 分間に設定，つまりメールの受信確認を 5 分ごとに行うように設定する。すると，マイコンの通常のプログラム（ポーリング処理）とは別にタイマにより 5 分間が計測されて 5 分たったときにタイマ割り込み（Timer Interrupt）が発生する。このとき，先のメールの受信確認の処理は図 2.56 のように表せる。ポーリング処理では頻繁にメールの受信確認を行っていたのが，タイマ割り込みを用いると，ほとんどメール受信確認のタスク処理を行う必要がなくなる。ただ，タイマ割り込みが発生したときにメールの受信確認というタスクを実行するのみでよくなり，無駄なメールの受信確認処理を大幅に減らせる。一般に，割り込みが発生したときに実行されるタスクは，**割り込みサービスルーチン（ISR: Interrupt Service Routine）**と呼ばれる関数である。割り込みサービスルーチンは，あらかじめ設定した割り込み要因，ここではタイマ割り込みの 5 分間，に条件が合致したときにのみ実行が許される関数であり，ユーザが自由に呼び出すことができない関数である。また，図 2.56 のように割り込みサービスルーチンの実行が終了すると，割り込み前の処理状態に戻り，通常のタスク処理が再び実行される。タイマ割り込みはメールの受信確認以外にも使用されている。スマートフォンの操作をせず一定時間経過したとき画面が暗くなる（バックライトの白色 LED が消灯），無線電波の受信レベル（アンテナマーク）の表示更新（受信信号強度（RSSI 値）の計測）などに使用されており，スマートフォンの消費電力の低減に貢献している。

ここで説明したタイマ割り込み以外にも**外部割り込み**がある。外部割り込み(External Interrupt)は，スイッチなどマイコンに接続されたデバイスの状態変化によって発生する割り込みである。マイコンの外に接続されたデバイス（外部）からの割り込みのため，外部割り込みと呼ばれている。例えば，スマートフォンのタッチパネルでキーが押される状況を考える。あなたがいつ押すかわからないキーの状態をつねに調べ続けるのは，先のメールの受信確認と同様に無

駄なタスク処理である。そこで外部割り込みを用いると，“キーが押された”ということを割り込み要因として，割り込みコントローラがその状態を検出して割り込みが発生する。これにより，マイコンはキーの状態を気にせずに別の処理に専念できるようになる。つまり，キーが押されたときにのみ，押されたキーに応じた処理（割り込みサービスルーチン）を実行すれば良いのでキーを入力したときのレスポンスも良くなる。タイマ割り込みが一定の時間間隔で割り込みが発生するのに対して，外部割り込みではマイコン外部に接続されたデバイスの都合によって割り込みが発生するため不定期に割り込みが発生することになる。

以上，ポーリング処理，タイマ割り込み，外部割り込みの処理をまとめると図 2.57 のように表せる。ポーリング処理は，逐次与えられたタスク処理を繰り返し実行する。タイマ割り込みは，あらかじめ設定した時間間隔（T）で割り込みが発生して割り込みサービスルーチン（ISR）が実行される。外部割り込みは，マイコンに接続されたデバイスの状態変化を要因として割り込みが不定期に発生し割り込みサービスルーチンが実行される。

プログラムでは，ポーリング処理，タイマ割り込み，外部割り込みを組み合わせて使用できる。ただし，割り込み処理の特徴として，たとえポーリングでタスク処理の最中であっても，割り込みが発生すれば，ただちに割り込みサービスルーチンを実行する必要があり，タスク処理は途中で中断される。このとき，割り込みサービスルーチンの実行に時間がかかると，他のタスク処理に影響を与えることがあるので注意が必要である。タイマ割り込みでは，割り込み周期 T 以内に処理を終わらせる必要がある。タイマ割り込みの周期 T 以内に終わらない場合には，割り込みは持ち越されて，割り込み終了後に実行されることになり正しい周期で割り込みが実行されなくなる。外部割り込みでは，状態の検出に対するフラグを立てて，ポーリングでフラグを処理するなど割り込みサービスルーチン内での処理が最短で終わるように工夫が必要である。

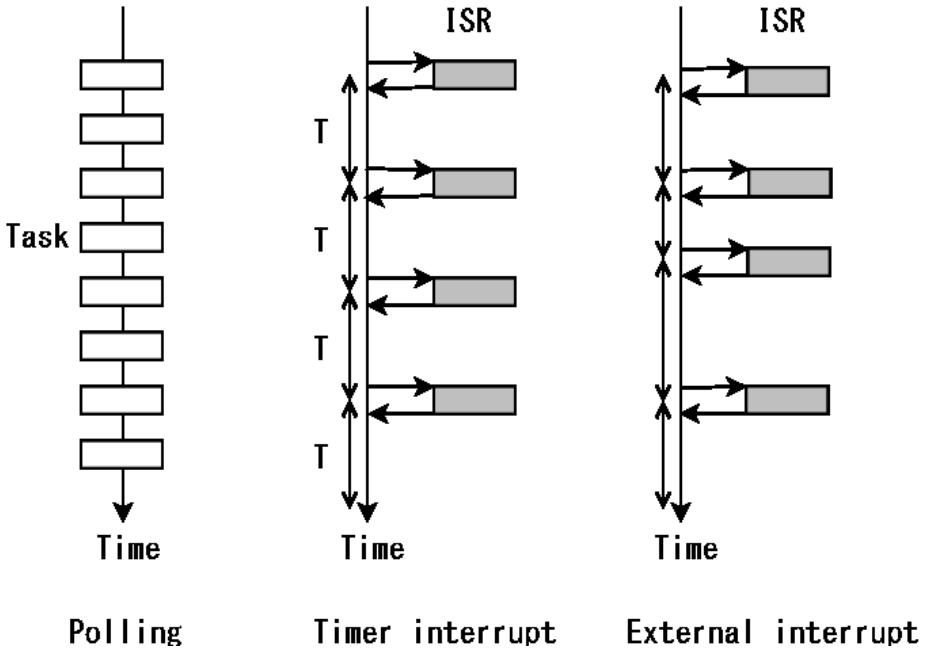

図 2.57　ポーリング処理とタイマ割り込み，外部割り込み処理

2.4.2 ポーリングの問題点

ポーリングは，LED の点滅やスイッチの状態確認を始めとして，無限ループ内で逐次タスク処理を実行する方式である。図 2.58 のように，無限ループ内で定期的に要求（リクエストやイベント）の有無をチェックして，要求があるときにその処理を実行するとする。ポーリングは，プログラムの作成は比較的簡単だが，

①要求の有無を確認する時間によっては，要求を取りこぼす場合がある。

②要求の有無を確認する時間間隔が長いと，要求に素早く応答できないことがある。

③要求の有無を確認する時間が異なる場合，応答までの時間も異なり安定した応答時間が保証されない，などの問題がある。したがって，リアルタイム性が要求される場合には，ポーリングより割り込みを用いる方が良いことがある。

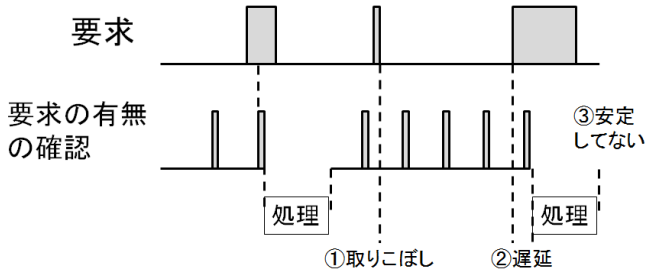

図 2.58　ポーリング処理の問題点

2.4.3　タイマ割り込み

タイマとは，正確な時間を計るときに使用するマイコンの内蔵機能の一つである。図 2.59 に示すように，タイマには時間を計測するタイマカウンタがあり，システムクロック（Arduino UNO は 16MHz）に同期してカウントアップ（またはカウントダウン）するハードウェアカウンタである。タイマカウンタは，ポーリングによるタスク処理に関係なく，システムクロックに合わせて一定の時間間隔でカウントアップ（またはダウン）される。8 ビットタイマの場合，0 から 255 までカウントできる。タイマ割り込みが発生するタイミングは，図 2.59 のようにタイマカウンタがある値 N に一致したときや，タイマカウンタがオーバーフローしたときに発生する。これが繰り

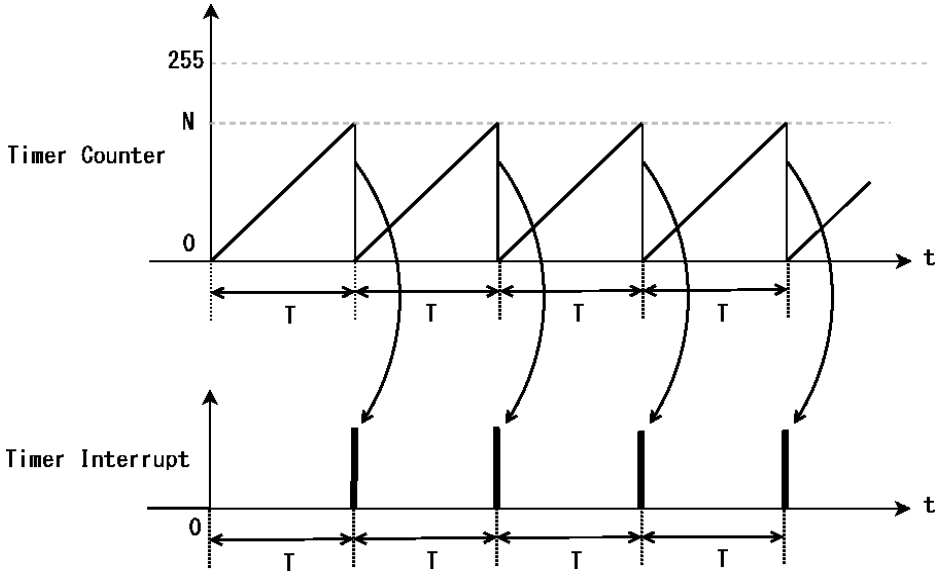

図 2.59　タイマカウンタとタイマ割り込みの発生

返されることにより，一定の時間間隔 (T) でタイマ割り込みが起こる。タイマ割り込みの条件は，プログラムで変更できる。タイマ割り込みを使用すると，他のタスク処理が実行されていても，正確な時間間隔で要求の有無の確認が可能になる。つまり，ポーリング処理の③の問題点を回避できる。タイマ割り込み処理は，Arduino のプログラムの void loop()関数内に処理はなく，タイマ割り込み要因に応じて呼び出される独立した関数（割り込みサービスルーチン）である。Arduino IDE では Timer1 および MsTimer2 のライブラリなどが準備されており，簡単な設定で割り込みを使用できる。

2.4.4　Arduino のタイマ割り込み

Arduino には，MsTimer2 というタイマ割り込みを処理するためのライブラリが用意されている。このライブラリを使うと，タイマ割り込みの発生時間間隔を設定して，割り込みサービスルーチンを記述するのみでタイマ割り込みを使用できる。次にライブラリの詳細について説明する。

MsTimer2：ミリ秒単位でカウントし，割り込み処理を実行するためのライブラリである。

MsTimer2::set 関数：タイマ割り込みが発生する時間の設定（ミリ秒単位），およびタイマ割り込みが発生するたびに呼び出される関数（割り込みサービスルーチン）を指定する。割り込みサービスルーチンは引数を取らない void 型で宣言する。

MsTimer2 は Timer2 を使用しており，Timer2 は D3, D11 の PWM にも使用されているので，analgWrite を併用する場合に注意が必要となる。**（MsTimer2 を使用して割り込み処理を実装している場合は，D3，D11 をディジタル PWM 出力として使用しないこと）**

書式：MsTimer2::set(ms, void(*f)())
引数 1：ミリ秒単位で指定する
引数 2：割り込み時に実行する関数，引数・戻り値なしの void 型で宣言する
戻り値：なし

MsTimer2::start 関数：MsTimer2 のタイマ割り込みを開始する関数である。
書式：MsTimer2::start();
引数：なし
戻り値：なし

MsTimer2::stop 関数：MsTimer2 のタイマ割り込みを終了する関数である。
書式：MsTimer2::stop();
引数：なし
戻り値：なし

＜演習 2.4.1＞　タイマ割り込みライブラリ MsTimer2 の登録

MsTimer2 ライブラリの登録を行う。

(1) MsTimer2.zip をダウンロードする(manaba にアップロードしている。解凍不要)。

(2) arduino を起動する。

$ arduino&

(3) ライブラリを追加する。

・ファイルメニューから，「スケッチ」→「ライブラリをインクルード」を選択

・「Zip 形式のライブラリをインストール」をクリックして，MsTimer2.zip を選択

(4) MsTimer2 が追加されたか確認する。

・ファイルメニューから，ファイル→スケッチの例→MsTimer2 があるか確認（リストの一番下）

＜演習 2.4.2＞　タイマによる一定時間間隔の割り込みによる LED の点滅

MsTimer2 のサンプルスケッチを用いて，タイマ割り込みによる LED の点滅を行う。サンプルスケッチでは，タイマ割り込みが 500ms 間隔で発生するように設定している。また，タイマ割り込みが発生したときに実行される割り込みサービスルーチンとして flash 関数も設定されている。割り込みサービスルーチン flash では，500ms 毎に LED の点灯，消灯を繰り返すように変数 output が HIGH から LOW，LOW から HIGH に更新される。

マイコンのポート	ポートの入出力	接続先
D13	ディジタル出力	LED1（赤）

(1) PC から USB ケーブルを抜く。

(2) 図 2.60 の回路を組む。(Arduino と PC を接続してはならない)

(3) Arduino UNO ボードとパソコンを USB ケーブルで接続する。

(4) arduino を起動する。

(5) Arduino UNO ボードの選択，シリアルポートの設定をする。

Arduino UNO ボードの選択：

ファイルメニュー：ツール，マイコンボード，Arduino UNO

シリアルポートの選択：

ファイルメニュー：ツール，シリアルポート，/dev/ttyACM0

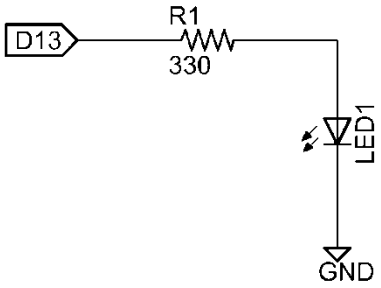

図 2.60　LED の発光回路

(6) サンプルスケッチを開く。

ファイル→スケッチの例→MsTimer2→FlashLed を開く。

(7) ファイルメニュー　書き込み

(8) 図 2.61 のように，LED が 500ms ごとに点灯，消灯を繰り返すことを確認する（周期 T=1.0 秒で点滅する：周波数 f=1.0 Hz)。

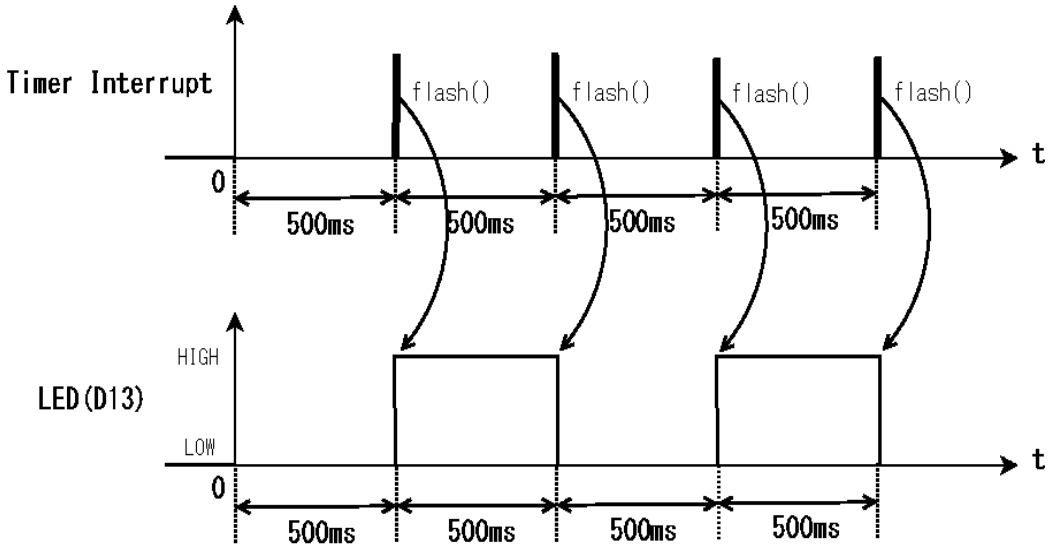

図 2.61　MsTimer2 サンプルプログラム（FlashLed）の実行波形

```
#include <MsTimer2.h>

void flash() {    // 割り込みサービスルーチン（LED の点滅）
  static boolean output = HIGH;
  digitalWrite(13, output); // LED を D13 に出力設定
  output = !output;   // HIGH->LOW, LOW->HIGH 反転させる
}
void setup() {
  pinMode(13, OUTPUT); // 13 番ピンを出力ポートに設定
  MsTimer2::set(500, flash); //  タイマ割り込み間隔の設定(T=500ms)
  MsTimer2::start(); //  タイマ割り込み開始
}
void loop() {   // ポーリングによるタスクなし
}
```

＜演習 2.4.3＞　タイマ割り込みの時間間隔の変更

ポーリングとタイマ割り込みを組み合わせてプログラムすることで，ポーリングの合間にタイマ割り込みが発生していることを確認する。ポーリングとして LED1 が周期 1 秒で点滅し，タイマ割り込みで LED2 を一定周期 2 秒で点滅する様子を見る。

マイコンのポート	ポートの入出力	接続先
D13	ディジタル出力	LED1（赤）
D9	ディジタル出力	LED2（黄）

(1) 図 2.62 の回路をブレッドボード上に組む。（Arduino と PC を接続してはならない）

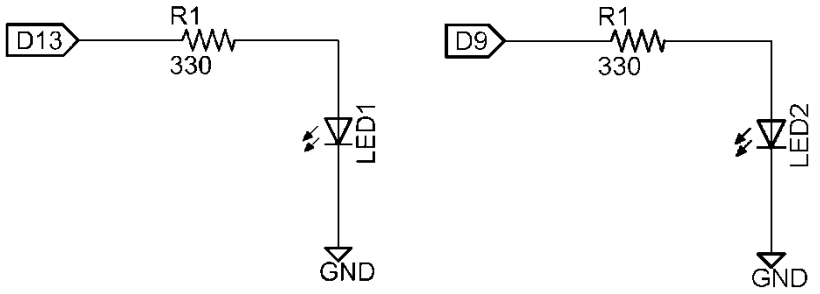

図 2.62　LED の発光回路

(2) サンプルスケッチを開く。

ファイル→スケッチの例→MsTimer2→FlashLed を開く。

(3) D13 ポート，D9 ポートをディジタル出力に設定する。

(4) タイマ割り込みの時間間隔を 1000msec (1 秒)に設定する。

(5) ポーリング（loop()内）に LED1 を 500ms 毎に点灯，消灯を繰り返すタスクを追加する。

(6) スケッチに名前を付けて保存する（Tr243_20180502 など）。

(7) ファイルメニュー　書き込み

(8) 図 2.63 のように，LED1 がポーリングにより 500ms 毎に点灯，消灯を続け，LED2 がタイマ割り込みにより 1000ms 毎に点灯，消灯を繰り返すことを確認する。

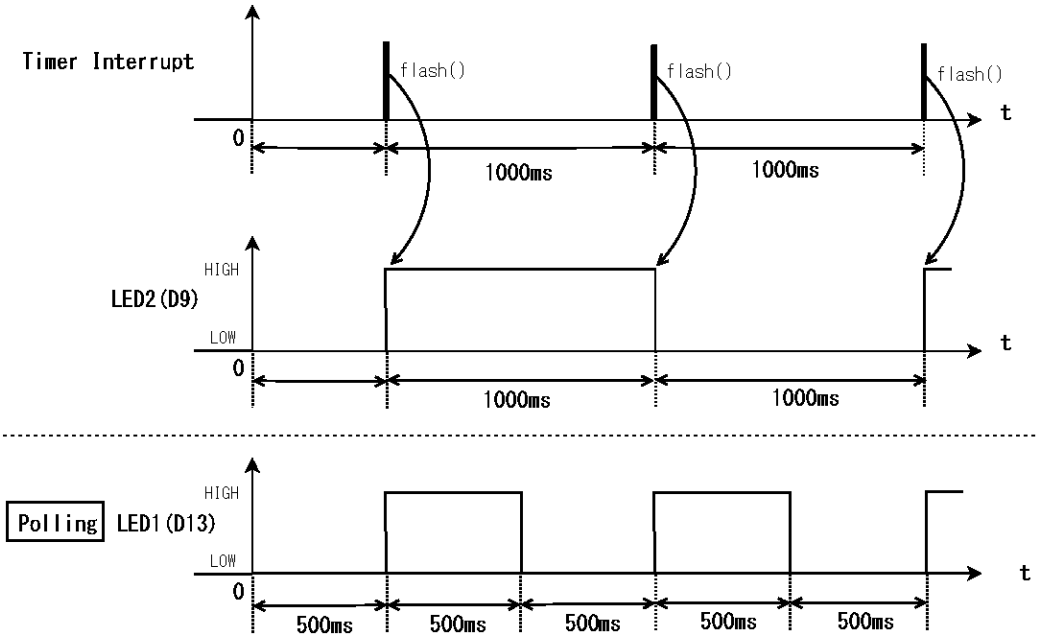

図 2.63　ポーリングと割り込みによる周期の異なる LED の同時点滅

```
#include <MsTimer2.h>
const int LED1_PIN = 13; // 13 番ピンに LED を接続
const int LED2_PIN = 9; // 9 番ピンに LED を接続

void flash() {    // 割り込みサービスルーチン（LED の点滅）
  static boolean output = HIGH;   // output の初期状態を HIGH にする
  digitalWrite(LED2_PIN, output); // 9 番ピンに出力（HIGH:5V, LOW:0V）
  output = !output;   // HIGH->LOW, LOW->HIGH 反転させる
}
void setup() { // 初期設定
  pinMode(LED1_PIN, OUTPUT); // 13 番ピンを出力ポートに設定
  pinMode(LED2_PIN, OUTPUT); // 9 番ピンを出力ポートに設定
  MsTimer2::set(1000, flash); // タイマ割り込み間隔の設定(T=1000ms)
  MsTimer2::start(); // タイマ割り込み開始
}

void loop() { // ポーリング
    digitalWrite(LED1_PIN, HIGH); // 13 ピンに 5V 出力
    delay(500); // 500ms 待機
    digitalWrite(LED1_PIN, LOW); // 13 ピンに 0V 出力
    delay(500); // 500ms 待機
}
```

＜課題 2.4.1＞ タイマ割り込みによる LED 点灯の繰り返しフェイドアウト

図 2.62 の回路を組み，タイマ割り込みの時間間隔を 20ms に設定する。LED1 をディジタル出力に設定，LED2 をディジタル PWM 出力にして

・ポーリング：LED1 を 500ms 毎に点灯，消灯の繰り返し，

・タイマ割り込み： 5 秒(5000ms)の間に発光状態から消灯状態へフェイドアウトを繰り返すプログラムを作成せよ。さらに，図 2.64 の割り込み周期，**LED2, LED1 の信号変化の様子をグラフに記入せよ。**

－LED1 を D13 ポートに接続，LED2 を D9 ポートに接続

－D13 をディジタル出力に設定，D9 ポートをディジタル PWM 出力に設定

－ポーリングで，演習 2.4.3 同様に 500ms 間隔で LED1(D13)を点灯，消灯する。

－タイマ割り込みの割り込み時間間隔を 20ms に設定して，5 秒ごとに発光状態から消灯状態へフェイドアウトを繰り返す割り込みサービスルーチンを作成する。

（2-3 AD 変換の演習 2.3.4 および発展課題 2.3.3 とは異なり，タイマ割り込みを使用して Duty 比を変更する）

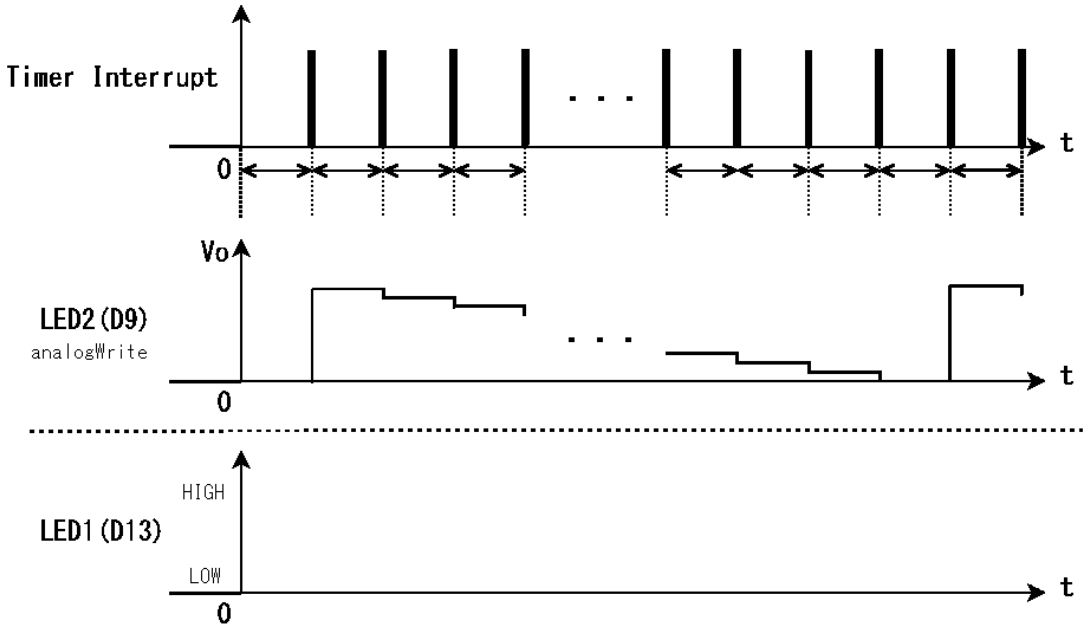

図 2.64 タイマ割り込みによる LED の点滅

2.4.5 millis 関数を使用した時間計測処理

タイマ割り込みの使用が困難な場合がある。例えば, MsTimer2 のライブラリを使用する場合, D3 および D11 ポートを使用することができないが，どうしても D3 または D11 ポートを使用しなければならない場合がある。このような場合，MsTimer2 のライブラリを使用することは容易ではない。対策として，別のタイマ割り込み用のライブラリを使用するか，もしくは，時間を計測する関数を使用する，などで対応する方法が挙げられる。本節では，時間を計測する関数について紹介する。Arduino IDE には，時間を計測する関数として millis 関数，micros 関数などが準備されている。ここでは millis 関数を取り上げる。

millis 関数：ミリ秒間隔で時間を計測する関数である。
書式：millis()
引数：なし
機能：スケッチが実行されてからの経過時間を計測
戻り値：unsigned long 型の値でスケッチがスタートしてからの経過時間をミリ秒単位で出力
（unsigned long は 4 バイトの符号なし整数：0～4,294,967,295）

＜演習 2.4.4＞　millis 関数による LED の点滅

millis 関数を使用して LED を点滅させる。具体的には，millis 関数を使用して 750ms を計測して，750ms 毎に LED の点灯，消灯を繰り返すための回路およびスケッチを構築する（LED は 1.5 秒間隔で点滅する）。

マイコンのポート	ポートの入出力	接続先
D13	ディジタル出力	LED1（赤）

(1) PC から USB ケーブルを抜く。
(2) 図 2.60 の回路を組む（演習 2.4.2 参照）。（Arduino と PC を接続してはならない）
(3) Arduino UNO ボードとパソコンを USB ケーブルで接続する。
(4) arduino を起動する。
(5) Arduino UNO ボードの選択，シリアルポートの設定をする。
Arduino UNO ボードの選択：
ファイルメニュー：ツール，マイコンボード，Arduino UNO
シリアルポートの選択：
ファイルメニュー：ツール，シリアルポート，/dev/ttyACM0
(6) 新規スケッチを開く。
下記のサンプルスケッチを参考に，millis 関数を用いて時間を計測するスケッチを作成する。
(7) スケッチに名前を付けて保存する（Tr244_20180502 など）。
(8) ファイルメニュー　書き込み
(9) LED が 750ms ごとに点灯，消灯を繰り返すことを確認する（周期 T=1.0 秒で点滅する：周波数 f=1.0 Hz）。

```
unsigned long timePrev = 0; //  基準となる時間を格納

void flash() {   // LED の点滅処理
  static boolean output = HIGH;
  digitalWrite(13, output); // LED を D13 に出力設定
  output = !output;   // HIGH->LOW, LOW->HIGH 反転させる
}
void setup() {
  pinMode(13, OUTPUT); // 13 番ピンを出力ポートに設定
}
void loop() { // millis 関数による時間計測
  unsigned long timeNow = millis(); // millis 関数を用いて現在の時間情報を取得
  if (timeNow - timePrev >= 750) { // 750ms 以上経過
    flash(); // flash 関数の処理を実行
    timePrev = timeNow; //  時間情報の更新
  }else{}
}
```

＜課題 2.4.2＞　millis 関数による照度センサのデータ取得

millis 関数を用いて一定時間間隔で照度センサのデータを取得する。millis 関数によって決められたある程度正確な時間間隔でデータを取得できることを確認する。

マイコンのポート	ポートの入出力	接続先
A0	アナログ入力	照度センサ

－ 図 2.65 の回路を組む。抵抗 R は 10kΩ を使用

－ millis 関数を使用して，アナログポートに接続された照度センサの状態を 500ms 間隔で取得するスケッチを作成

－シリアルモニタを使用して，500ms 間隔の照度センサの状態を確認

・手で光を遮る

・光を当てる（手で光を遮らない）

・上記 2 つの状態を素早く繰り返す

ヒント：analogRead を使用する

シリアルモニタを使用して確認する

※照度センサ以外の回路は絶対に接続しない。

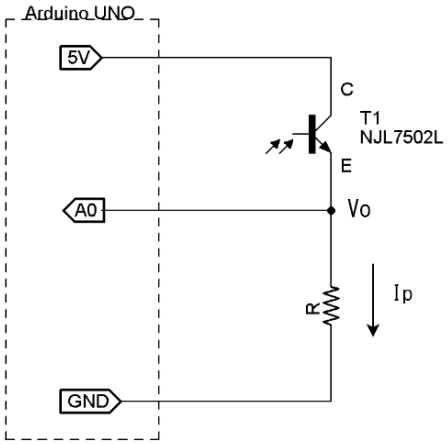

図 2.65　照度センサの回路

2.4.6　外部割り込み

外部割り込みは，マイコンに接続されたデバイスの状態変化によって割り込み要求が発生する。

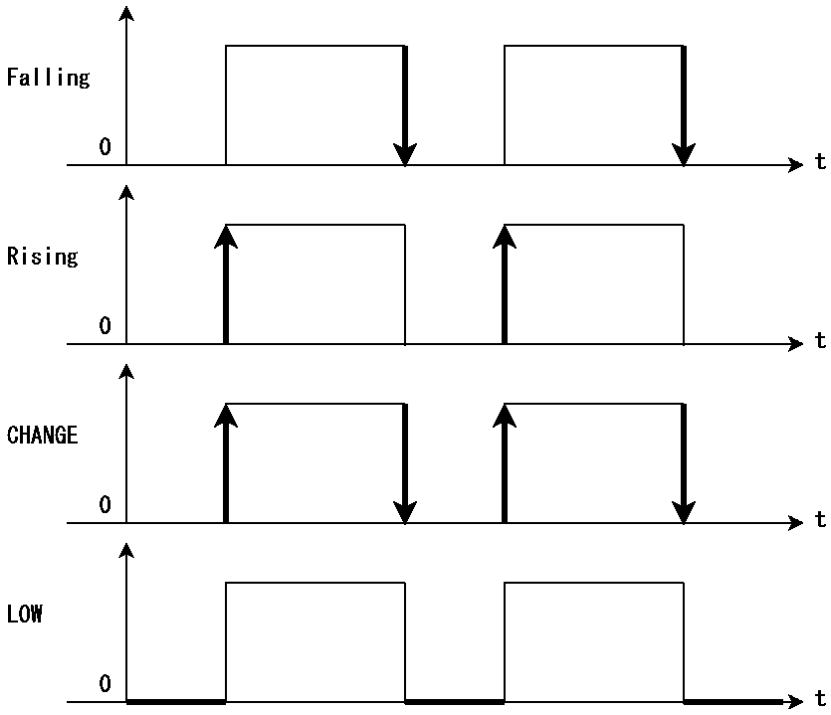

図 2.66　外部割り込みの発生

外部割り込みの発生要因として，図 2.66 に挙げる状態変化がある。デバイスの状態が HIGH から LOW に変化したとき，立ち下がりエッジ（FALLING）で割り込みがかかるという。逆に，LOW から HIGH に変化したときには立ち上がりエッジ(RISING)，また，HIGH から LOW，LOW から HIGH に変化したときには両エッジ(CHANGE)で割り込みが発生する。通常は HIGH レベル状態にあり，何らかの要因で LOW レベル状態が続いたときにも割り込みが発生する。外部割り込みは，割り込みが発生すると同時に割り込みサービスルーチンが呼ばれるため，外部デバイスの状態変化に対してただちに処理を実行できる。これにより，2.4.2 節に挙げたポーリング処理時の①②③の問題を回避できることがある。

2.4.7　Arduino の外部割り込みの API

Arduino の外部割り込みを扱うための API を次に示す。

attachInterrupt 関数：D2，D3 ポートを使用して外部からの入力信号により割り込み処理を実行するための関数である。

書式：attachInterrput(interrupt, ISR, mode)

引数：interrupt では，“0”または“1”を指定，**0 の場合 D2 ポート，1 の場合 D3 ポート**が外部割り込みポートに設定される

引数：ISR では，割り込み時に実行する関数を指定，関数は引数なしの void 型で宣言する

引数：mode では，指定されたポートの状態に応じて割り込みを発生させるかを指定する。**LOW, CHANGE，FALLING, RISING** の 4 つのモードがある。LOW は LOW レベルになったときに割り込みが発生する。CHANGE は信号レベルが変化したときに割り込み発生し，FALLING は信号が HIGH から LOW に変化したとき，RISING は信号が LOW から HIGH に変化したときに発生する。

戻り値：なし

detachInterrput 関数：指定した外部割り込みを停止する関数である。

書式：detachInterrupt(interrupt)

引数：interrupt では，“0”または“1”を指定，0 の場合 D2，1 の場合 D3 ポートが設定

戻り値：なし

＜演習 2.4.5＞　スイッチ動作の外部割り込みによる LED の点灯・消灯

外部割り込みの割り込み発生要因としてスイッチを用いる。スイッチの状態変化を検出して割り込みを発生させ，外部割り込みの設定の違いによる割り込み動作を LED1 により確認する。

マイコンのポート	ポートの入出力	接続先
D2	ディジタル入力（外部割り込み 0）	スイッチ
D13	ディジタル出力	LED1（赤）

(1) 図 2.67 の回路を組む。（Arduino と PC を接続してはならない）

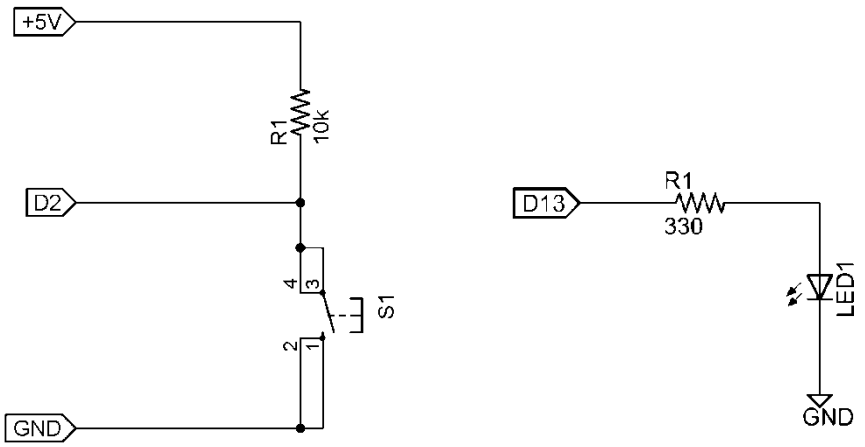

図 2.67　スイッチによる外部割り込みの動作確認用回路

(2) 新規スケッチを開く。

```
const int LED_PIN = 13; // LED を D13 に接続
volatile int output = LOW; //  ディジタル出力の値を格納する変数（初期：“LOW”）
/*  初期設定  */
void setup() {
    pinMode(LED_PIN, OUTPUT); // D13 をディジタル出力に設定
  /*
  外部割り込みの設定
  割り込みに使用するディジタルポートを D2 に設定（1 番目の引数=“0”）
  割り込みサービスルーチン blink 関数を実行（2 番目の引数=関数名）
  （例）割り込み要因としてスイッチの状態が“LOW”のときに発生（3 番目の引数）
  */
  attachInterrupt(0, blink, LOW);
}
/*  メイン関数  */
void loop(){
  digitalWrite(LED_PIN, output); //  スイッチを押すたびに点灯・消灯を繰り返す
}
/*  割り込みサービスルーチン  */
void blink(){ // 割り込みの条件にあわせ LED が点灯・消灯を繰り返す
  output = !output; //  ディジタル出力を反転（HIGH→LOW，LOW→HIGH）
}
```

(3) ファイルメニュー　名前を付けて保存，書き込み（Tr245_20190510 など）

(4) attachInterrupt を使用して，スイッチの外部割り込みにより LED を点灯・消灯するプログラムを作成する。ただし，外部割り込みの条件を以下の 4 種類に変更して，それぞれの動作を確認する（図 2.67 参照）。

① 外部割り込み要因 FALLING に変更して，外部割り込み動作を LED の状態で確認する。

② 外部割り込み要因 RISING に変更して，外部割り込み動作を LED の状態で確認する。

③ 外部割り込み要因 CHANGE に変更して，外部割り込み動作を LED の状態で確認する。

※スイッチを ON・OFF しているにも関わらず，外部割込みの処理が正確に実行されない場合があることを確認する。これはチャタリングという現象である。チャタリングについて調査し，レポートにて報告せよ。

・volatile 修飾子

プログラム中の volatile 修飾子は，コンパイラの最適化によって定義した変数が削除されたり，コードが書き換わるのを防止するためのものである。マイコンのプログラムでの使い方は，グローバル変数として定義した変数で，かつ割り込みサービスルーチン内で使用する変数には必ず volatile 修飾子を付けるようにする。

割り込みサービスルーチンは割り込み要因に対して呼ばれる関数であり，メイン関数から呼ばれる関数ではない。そのため，割り込みサービスルーチン内で使用している変数がメイン関数内で参照されない場合には，コンパイラの最適化によって変数が無くなってしまうことがある。そのため，割り込みを用いるプログラムでは，変数が何処で参照（使用）されているかをよく見て volatile 修飾子をつける必要がある。

＜課題 2.4.3＞ スイッチ動作の外部割り込みによる LED 発光のフェイドイン・フェイドアウト

図 2.68 の回路を組む。 ポーリングで LED の発光がフェイドインを繰り返しておき，外部割り込みスイッチが押されると，ただちに LED の発光がフェイドアウトに切り替わるプログラムを作成せよ（図 2.68 参照）。

マイコンのポート	ポートの入出力	接続先
D2	ディジタル入力（外部割り込み 0）	スイッチ
D9	ディジタル PWM 出力	LED1（赤）

－外部割り込み要因は FALLING に設定

－フェイドアウト：原則として 3 秒(3000ms)の間に発光状態から消灯状態へ

－フェイドイン：原則として 3 秒(3000ms)の間に消灯状態から発光状態へ

－スイッチの切り替えは図 2.69 を参考すること

※注意；タイマ割り込み，もしくは millis 関数を用いて時間計測すること。

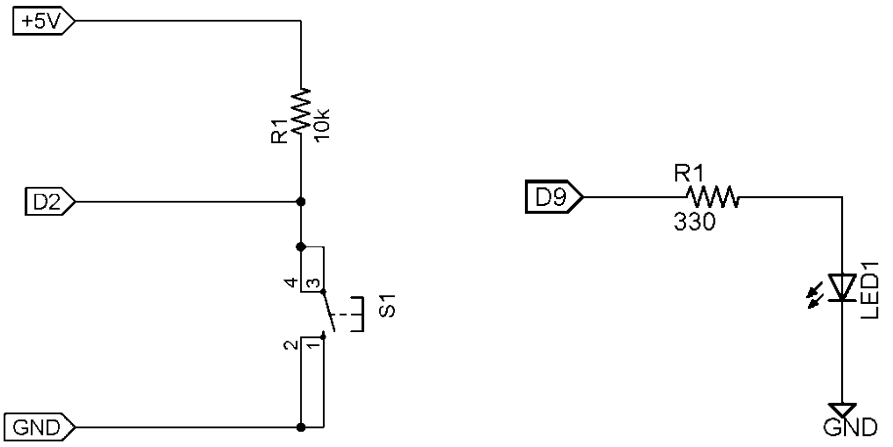

図 2.68　スイッチによる外部割り込みと PWM 出力回路

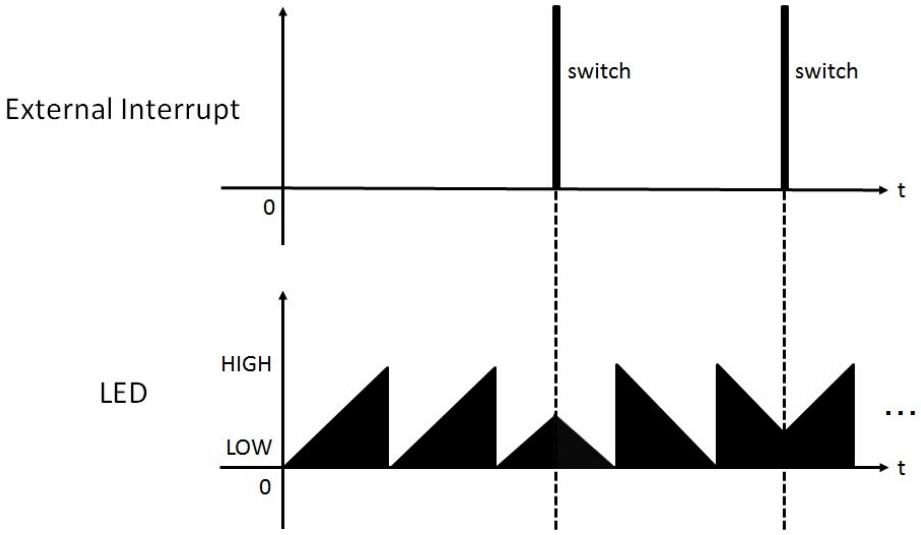

図 2.69　スイッチの外部割込みによる LED の発光

＜発展課題 2.4.1＞　ポーリングによる動作状況の表示とタイマ割り込みによる照度の計測終了表示

図 2.70 の回路を組み，アナログ入力ポート（A0）に接続された照度センサの状態を 1 秒間隔で取得せよ。計測開始から 10 秒経過したときに，LED2 を 1 秒点灯させた後消灯させて計測の終了を知らせよ。このとき，1 秒毎に照度をシリアルモニタに表示させよ。また，照度計測中は，ポーリングで 500ms 毎に LED1 を点滅させて動作中であることを示せ。ただし，10 秒間の計測は繰り返し行うこと。ただし，ポーリング処理においても delay を使用してはならない。

マイコンのポート	ポートの入出力	接続先
A0	アナログ入力	照度センサ
D13	ディジタル出力	LED1（赤）
D9	ディジタル出力	LED2（黄）

ヒント；

－割り込みサービスルーチン内で照度センサの状態を 1000ms 間隔で取得してシリアルモニタに表示する。

－10 秒が経過したとき，LED2 を 1 秒点灯させた後消灯させて計測終了を知らせる。（LED2 の 1 秒の点灯は，割り込みサービスルーチン内で delay を使わずに点灯させること）

－照度の計測中には，ポーリングで LED1 を 500ms 毎に点灯，消灯を繰り返す。

－照度の計測後は，LED1 は消灯するように設定する。

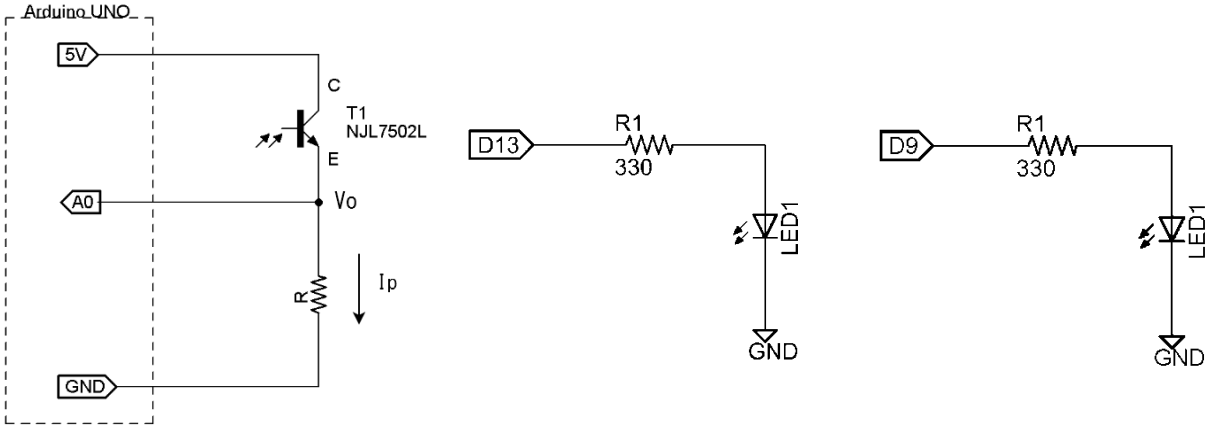

図 2.70　照度センサの回路と LED 点灯回路

＜発展課題 2.4.2＞ ポーリングによる動作状況の表示と疑似タイマ割り込みによる照度の計測終了表示

発展課題 2.4.1 の問題に対して，millis 関数を用いて実装せよ（**MsTimer2 は使用しない**）。

ヒント；

－時間計測用の変数を複数準備する。

※※注意：発展課題 2.4.1 と同様に，ポーリング処理においても delay は使用してはならない

＜発展課題 2.4.3＞ チャタリング問題の解決

演習 2.4.5 において生じるチャタリングの問題を解決する回路またはスケッチを作成せよ。

ヒント；

－演習 2.2.3 を参考にする（スイッチの状態確認）。

－演習 2.4.4 を参考にする（タイマ割り込みを使用しない時間計測）。

【レポート 2.4　（2019 年 5 月 17 日(金) 12：50 締切)】

※スケッチには詳細なコメントを記述すること。コメントが無いものは採点対象外とする。

※自身で作成・変更したスケッチ部分を示すこと。サンプル全てをそのまま貼付けしない。

レポート 2.4.1　基礎実験第 4 回の概要

基礎実験第 4 回目の実験の目的，実施した実験の概要，および理解した事柄を 100～200 字程度で説明せよ。

レポート 2.4.2　回路実装

レポート概要：基礎実験 4 で使用した回路のブレッドボード配線図を報告せよ。

レポート 2.4.3　タイマ割り込みによる LED の点滅

レポート概要：課題 2.4.1 において作成したスケッチをおよび完成させた図 2.63 を報告せよ。また，MsTimer2 を使用せずにポーリング処理のみを使用した場合，どのような問題点があるのか，調査し報告せよ。

レポート 2.4.4　タイマ割り込みによる照度センサのデータ取得

レポート概要：課題 2.4.2 において作成したスケッチを報告せよ。また，「手で光を遮る」「光を遮らない」状態を素早く繰り返した時に生じた取りこぼしについて生じた原因を考察せよ。さらに，取りこぼしを回避する方法案記せ。

レポート 2.4.5　スイッチ動作の外部割り込みによる LED の点灯・消灯

レポート概要：演習 2.4.5 において生じるチャタリングについて調査し報告せよ。また，attachInterrupt を使用せずにポーリング処理のみを使用した場合，どのような問題が生じるのか，調査し報告せよ。

レポート 2.4.6　スイッチ動作の外部割り込みによる LED 発光のフェイドイン・フェイドアウト

レポート概要：課題 2.4.3 で作成したスケッチを報告せよ。また，作成したスケッチで工夫した点を記せ。

レポート 2.4.7　発展課題 2.4.1

レポート概要：発展課題 2.4.1 において作成したスケッチと工夫点を報告せよ。照度センサの値の変化をグラフ化し，その結果を報告せよ。

レポート 2.4.8　発展課題 2.4.2

レポート概要：発展課題 2.4.2 において作成したスケッチと工夫点を報告せよ。照度センサの値の変化をグラフ化し，その結果を報告せよ。

レポート 2.4.9　発展課題 2.4.3

レポート概要：発展課題 2.4.3 において作成したスケッチと工夫点を報告せよ。

・使用部品一覧

使用部品一覧

部品	個数	備考
Arduino Uno	1	
USBケーブル　Aオス-Bオス　1.5m A-B	1	
ブレッドボード　EIC-801	1	
ブレッドボード・ジャンパーワイヤ　EIC-J-L	1	
3mm 赤色LED OSDR3133A	1	
3mm 黄色LED OSYL3133A	1	
1/4W 330Ω	2	LED
1/4W 10kΩ	2	スイッチ，照度
タクトスイッチ	1	
照度センサ	1	

参考図書

[1] Massimo Banzi，船田 巧：Arduino をはじめよう －第 2 版－，株式会社オライリー・ジャパン (2014).

[2] 河連 庸子，山崎 文徳，神原 健：Arduino　スーパーナビゲーション，株式会社リックテレコム (2012).

[3] 神崎 康宏：Arduino で計る，測る，量る，CQ 出版株式会社 (2013).

[4] 田中 博，芹井 滋喜：PIC16 トレーナによるマイコンプログラミング実習，学校法人 東京電機大学(2013).