2.2　Arduino 開発環境とディジタル I/O

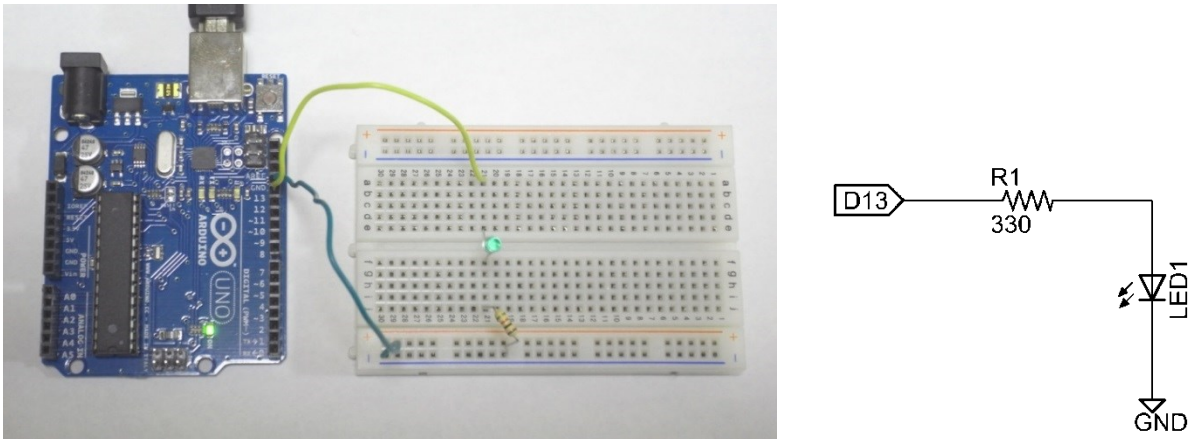
Arduino 開発環境とマイコンのディジタル I/O を使用した実験を行うことで，Arduino の基本的な使用方法やマイコンの開発方法を習得する。

実験目標：

・Arduino 開発環境に慣れる。
・Arduino UNO マイコンボードの仕組みを理解する。
・Arduino とブレッドボードの接続を行う。
・ディジタル IO ポートの原理を理解する。
・ディジタル IO ポートのプログラムを作成する。

2.2.1　Arduino 開発環境

Arduino 開発環境は，マイコンを搭載した Arduino ボード（ハードウェア）と Arduino ボードにアップロードするスケッチ（小さなコンピュータプログラム）を作るための Arduino IDE（Integrated Development Environment：統合開発環境）という開発プラットフォーム（ソフトウェア）より構成される。実験に使用する Arduino UNO に搭載されたマイコンは，Atmel 社（米国）の AVR ATmega 328P という 8 ビットマイコンである。



(a)　　(b)

図 2.13　Arduino UNO ボードのディジタル I/O と回路図

2.2.2　マイコンについて

マイコン（マイクロコントローラ，マイクロコンピュータ）とは，中央演算処理装置(CPU)，メモリ，入出力インタフェース(周辺回路) をワンチップのパッケージに納めた LSI の総称である。マイコンは，電気信号を入出力した機器制御を主な目的とし，家電製品などの制御用として製品内部に組み込まれている。電源 ON と同時に自己に内蔵したプログラムにしたがって動作する。マイコンの開発は，汎用のパソコンとは異なり，キーボードやディスプレイがないためクロス開発という手法で開発を行う。

・**CPU**　メモリに書き込まれているプログラムから一動作毎の命令を順々に読みだし，それを解釈し，一連の命令にしたがって数値計算，論理演算等を実行し，処理する電子回路である。また，その実行過程で必要となる内部一時記憶回路(レジスタ）やカウンタ等の補助回路も内蔵している。

・**メモリ**　プログラムやデータをディジタル信号として書き込み，保存しておく場所である。CPU から直接読み出し/書き込みのできる主メモリと，その他の補助メモリに分類できる。主メモリは CPU からの読み出し/書き込みの両方ができる RAM (Random Access Memory) とメモリ内容を読み出すだけで変更のできない読み出し専用メモリ ROM (Read Only Memory) がある。ROM には，通常使用時とは異なる条件を与えると消去/書き換えが可能な P-ROM，フラッシュ ROM などがある。補助メモリには，ハードディスク(HDD)，SD カード，USB メモリ，CD-ROM，DVD などがある。一般に読み出し/書き込みに要する時間は主メモリよりも遅い。

・**入出力インタフェース**　周辺装置の信号を CPU に直接入出力できるように，電気的特性やタイミングを整合させる中継用の回路である。マイコンの入出力インタフェースとして，2 値（0, 1）のディジタル信号の入出力にはディジタル IO ポート，電圧や電流などの連続量を扱うには AD/DA 変換器より構成されるアナログ入出力ポート，外部の信号変化を捉える割り込みポートなどが標準で実装されることが多い。また，有線通信機能（シリアル，I2C，SPI など）や無線通信機能（Wi-Fi，Bluetooth，ZigBee など）等のインタフェースを内蔵するマイコンもある。

・**組み込みシステム**　マイコンは，身の回りの家電製品にほとんどといっていいほど使用されている。たとえば，エアコン，掃除機，洗濯機，電話機，ゲーム機，プリンタ，リモコン，テレビ，DVD，ビデオなど，その数を挙げればきりがないほどである。パソコンが普及したとはいえ，快適な生活を送るにはマイコンの利用は必須といえるだろう。またマイコンは，家電製品のみならず工業用製品や医療用製品など実に幅広い分野で利用されている。このようなマイコンが介在するシステム(汎用のパソコンを除く）は，一般に“組み込みシステム(エンベデッドシステム)” と呼ばれている。実験で使用する Arduino UNO マイコンボードを用いて製作するロボットも組み込みシステムの一つである．マイコンのプログラム開発を通じて，組み込みシステム開発を経験できれば様々な分野での応用が期待できる。

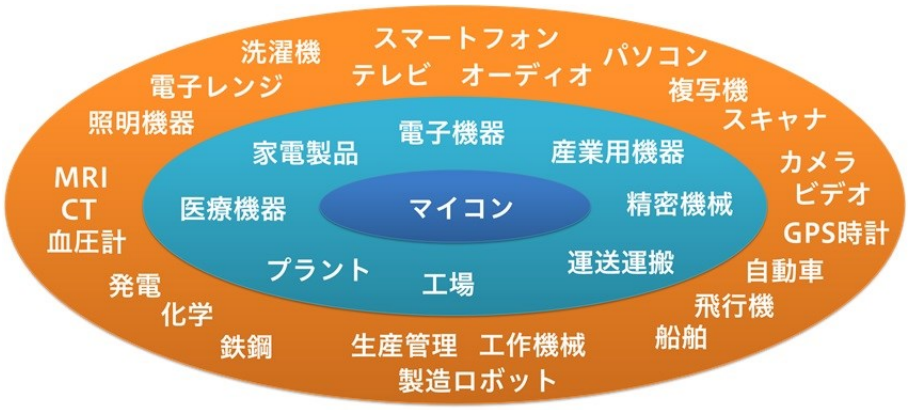


図 2.14　組み込みシステムの利用例

・組み込みシステムの例

組み込みシステムの例として“電子レンジ” について調べてみよう．ほとんどの人が電子レンジを利用したことがあると思う。スーパやコンビニなどで買った冷凍食品は，電子レンジを使って調理することでおいしく頂ける。その一連の流れは次のようになる。

(1) 冷凍食品を電子レンジの中に入れる
(2) あたため温度および時間を入力する
(3) あたためスタートボタンを押す
(4) あたため時間のあいだ待つ
(5) あたため完了を知らせる音がなる
(6) 温かくなった食品を取り出しおいしく頂く

さて，電子レンジの中でマイコンはどのような役割を担っているだろうか。手順(1) から(5) の流れに注目しよう。

それぞれの手順は，

(1) 電子レンジのドアを開けて，食品をターンテーブルにのせる。ドアの開閉が検出されて，あたため設定ができるようになる。

(2) 冷凍食品を温めるのに必要な温度を，“強”，“中”，“弱” あるいは具体的な“ 温度(500W や 600W)” のボタンを押して入力する。あたため時間は，“分” ならびに“秒” ボタンを押して入力する。このとき，表示パネルには設定した時間や温度が表示される。

(3) あたためスタートボタンを押すと，手順(2) で設定した温度と時間であたためが開始される。もし，途中で温めを止めたくなったり温め時間や温度を変更したいときには，とりけしボタンを押すことで直ちに温めをやめ手順(2) の状態に戻せる。

(4) 冷凍食品は，設定した時間の間，一定の温度を保ちながら温められる。このとき表示パネルには経過時間が表示され，あと何分何秒であたためが完了するのかをカウントダウンして知らせてくれる。

(5) 表示パネルの時間表示が 0 になり，あたためが完了したことを音を鳴らして知らせてくれる。以上の手順により，冷凍製品が適切な温度に温められる。マイコンとその周辺回路によって，これら一連の処理が行われる。下線部がマイコンの介在している箇所であり，ボタンの入力や温度，時間の設定，温め温度を一定に保つ制御，表示パネルへの経過表示を始めとして多くの要素があるが，利用する側にとっては使いやすいインタフェースになっている。このように，使い方がわからず分厚いマニュアルを開く必要のない操作の明瞭さが組み込みシステムの特徴の一つである。

表 2.11 基礎実験の項目に対応するマイコンの機能と用途

基礎実験の項目	マイコンの機能	電子レンジでの用途
ディジタル IO	I/O ポート	押しボタン入力や各種エラーなどの検出用
AD 変換，PWM	AD 変換器	温度や重さ，形状などセンサ入力用
AD 変換，PWM	PWM	スピーカで音や音声を鳴らす
割り込み	タイマ割り込み	動作の異常や処理の中断用
割り込み	外部割り込み	扉の開閉などの ON，OFF センサ入力用
AD 変換，PWM	PWM	ヒータやモータの駆動系制御用
シリアル通信	シリアル通信	―

・クロス開発　マイコンを用いた組み込みシステムの開発には「クロス開発」という方法で開発を行う。クロス開発では，開発に使用するホストコンピュータと開発したソフトウェアを実行するターゲットコンピュータで構成される。ホストコンピュータでは，ターゲットコンピュータ用にオブジェクトコードが生成できるクロスコンパイラ，クロスアセンブラ，クロスデバッガ，クロスリンカなどのクロス開発環境が整っている。ターゲットコンピュータは，ホストコンピュータとの通信機能，簡易デバッグ，プログラム実行機能を持った ROM モニタプログラムが書き込まれている。

2.2.3　Arduino UNO マイコンボード

Arduino UNO マイコンボードは，簡単にマイコンの機能を試したり，試作を行ったりするためのマイコンの評価ボードである。Arduino UNO には，図 2.15 のように，マイコンのすべてのピンと接続されたソケット，電源供給を行うための USB コネクタ，リセットスイッチ，動作確認用の LED（ON），テスト用の LED（L）が実装されている。

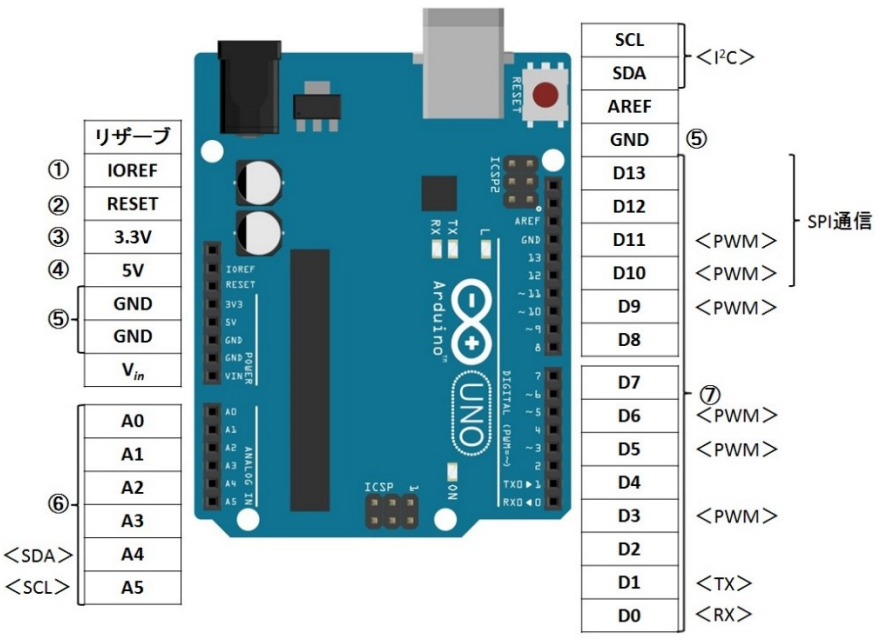


図 2.15　Arduino UNO マイコンボードのピン配置

図 2.15 の Arduino UNO の 4 本のピンソケットの内側には，マイコンの機能が白文字で印刷（シルク印刷）されている。

・電源

① **IOREF**：シールドがボードの Vcc 電圧を検出するための IOREF であり，Arduino Uno は 5V の Vcc に接続されている。
② **RESET**：リセットボタンを追加できる。
③ **3.3V**：3.3V の電源出力（最大 50mA の電源供給が可能）
④ **5V**：5V の電源出力（最大 500mA の電源供給が可能）
⑤ **GND**：グランド

・アナログ

⑥ **A0～A5**：アナログ入力ポート
A4(SDA)，A5(SCL)は I²C 通信ポートと兼用

・ディジタル

⑦ **D0～D13**：ディジタル I/O ポート（最大 40mA の電源供給が可能）
RX←D0，TX→D1 はシリアル通信ポート
D3，D5，D6，D9～D11 は PWM のアナログ出力
D10～D12 は SPI 通信ポート
とそれぞれ兼用している。

・AVR ATmega328P マイコン

Arduino UNO マイコンボードに搭載されているマイコンは，AVR 社の ATMega328P である。次に，ATMega328P マイコンの仕様を示す。

動作電圧（5V）：マイコンの電源としてパソコンの USB ケーブルを介して 5V が供給される。

動作クロック周波数(16MHz)：マイコンの動作クロックは 16MHz である。マイコン横に実装された水晶発振子によりクロックが生成される。クロックに同期してマイコンのメモリに記憶された命令が実行される。

Flash メモリ（32 kB）：Arduino IDE で作られたスケッチは，マイコンが解読できる機械語に変換（コンパイル）されて不揮発性の Flash メモリに書き込まれる。電源が切れても内容は保存される。**Flash メモリの容量を超えるスケッチは書き込むことが出来ないので注意すること。**

SRAM（1 kB）：スケッチでデータの保存用に使うメモリである。電源が切れると内容は不定になる。

EEPROM（2 kB）：スケッチで電源が切れても記憶したいデータの保存に使用される。データの書き込みにはマイコンに内蔵されている書き込み回路を使用するため，専用の EEPROM ライブラリを使用する。

2.2.4　Arduino IDE（開発環境）

Arduino IDE を起動すると，図 2.16 の Arduino IDE のウィンドウが表示される。Arduino IDE には，プログラムを作成するためのエディタ，Arduino マイコンを設定するためのツール，プログラムの検証（コンパイル），書き込みを行うボタン，シリアル通信モニタ，デバッグ表示などの機能がある。

Arduino IDE による開発の手順は，次の①から③の手順で行う。

① Arduino IDE のエディタで Arduino のプログラムのスケッチを作成

② スケッチをコンパイル

③ Arduino マイコンボードにプログラムの書き込み

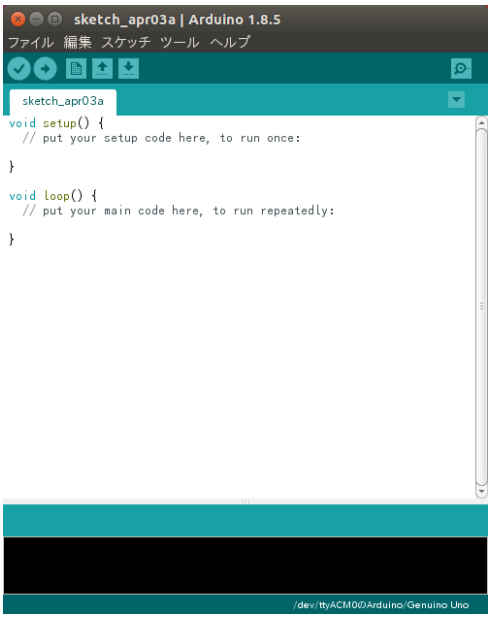


図 2.16　Arduino IDE ウィンドウ

2.2.5　Arduino の初期設定

Arduino IDE は，いろいろな種類の Arduino マイコンボードに対応している。そのため，Arduino 開発環境を使用する前に，実験で使用するマイコンボードの設定を行う必要がある。

(1) 使用するマイコンボードの設定

メニューバーの「ツール」を選択して，使用するマイコンボード（Arduino/Genuino UNO）を選択する。選択されたマイコンボードには「●」が記される（図 2.17）。

(2) シリアルポートの設定

① Arduino UNO マイコンボードとパソコンを USB ケーブルで接続する。

② Arduino IDE の「ツール」→シリアルポートを選択し，Arduino UNO が接続されているポート(/dev/ttyACM0)を選択する。選択されたポートには「✔」が記される（図 2.18）。

注意：USB ポートの差し替え，PC の再起動によって上記の設定が無効になることがある。マイコンに正しくプログラムが書き込めない，通信エラーが出るなどの場合には再確認が必要である。

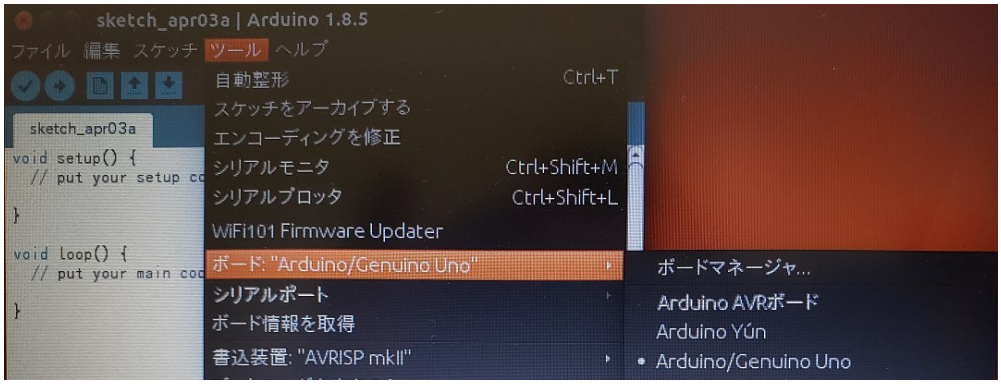


図 2.17　マイコンボードの設定（Arduino UNO を選択）

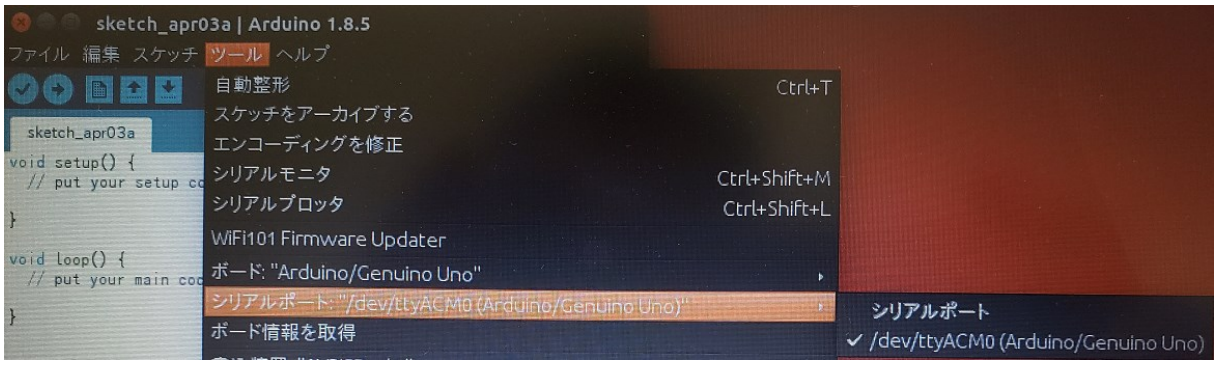


図 2.18　シリアルポートの設定（/dev/ttyACM0 を選択）

2.2.6　プログラムのコンパイルと書き込み

プログラムが完成した時点で，プログラムのコンパイルを行う。コンパイルを行いエラーが無いことを確認した後，マイコンにプログラムの書き込みを行う。

(1) コンパイル

コンパイルはツールバーの一番左のアイコンをクリックするか，もしくは，メニューバーの「スケッチ」を選択し，「検証・コンパイル」を選択する。

(2) 書き込み

書き込みツールバーの左から 2 番目のアイコンをクリックするか，もしくは，メニューバーの「ファイル」を選択し，「マイコンボードに書き込む」を選択する。

図 2.19　コンパイルと書き込み

2.2.7　Arduino のスケッチ

Arduino で作成されるプログラムは，スケッチと呼ばれる。スケッチは，C 言語のプログラムと同じ扱いのため，プログラムは必ず半角英数文字（記号含む）で入力する。コメント以外に，1 文字でも半角英数文字以外の全角文字（日本語）があると，Arduino IDE ではコンパイルされずエラーになるので注意すること。

また，スケッチは関数という命令をひとまとめにしたもので構成される。図 2.20 に関数のイメージを示す。関数の内容はカッコ｛　｝で囲む。関数内に記述された処理には，代入，演算，条件分岐などがある。さらに，関数は他の関数から呼ばれることで実行され，呼ばれた関数は処理を実行すると元の関数に戻る。また，呼ぶ関数と呼ばれる関数との間で，データが受け渡しされる場合がある。呼ぶ関数へ渡すデータを「引数」といい，呼んだ関数に渡すデータを「戻り値」という。引数や戻り値が無い関数では「空」を意味する「void」を代わりに記述する。

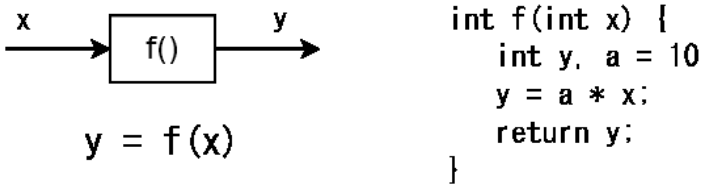


図 2.20　関数のイメージ（x の値を 10 倍して返す関数）

マイコンはハードウェアの設定と無限ループの繰り返し処理を行うため，図 2.21 に示すようにスケッチには初期設定用に setup 関数，無限ループ用に loop 関数が用意されている。この初期設定，繰り返し処理の関数内の処理はユーザが必ず記述する。

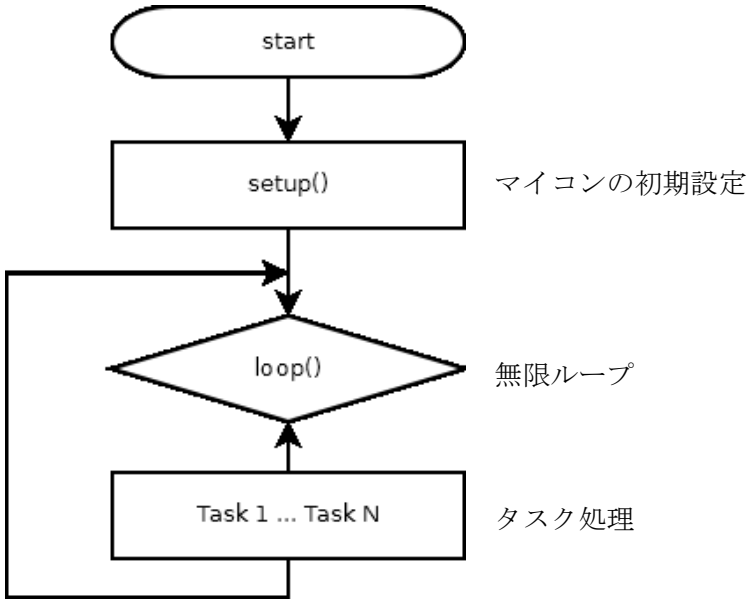


図 2.21　マイコンの初期設定と繰り返し処理（無限ループ）

初期設定の setup 関数は，以下のように記述する。

```
#define LED1 13 //  「LED1 は 13 と置き換えられる」と定義
void setup() {
    /*   ポートの初期化などの初期設定が記述される。*/
    /*   例：  pinMode(LED1, OUTPUT); */
}
```

setup 関数は ATmega328P マイコンの初期設定を行う関数であり，マイコンが起動したときに，最初に 1 回だけ実行される。スケッチの一番はじめに記述する（ただし，変数を定義する場合はその限りではない）。また，setup 関数は省略できないことに注意する。

繰り返し処理を行う loop 関数は，C 言語では無限ループに相当し，以下のように記述する。

```
void loop() {
    /*　主な処理内容が記述される。ディジタル入出力，アナログ入出力など　*/
}
```

この loop 関数は主処理を記述する関数であり，その処理は電源を切るまで繰り返される。また，setup 関数同様に loop 関数も省略できないことに注意する。

・スケッチの基本構造

スケッチの基本的流れは図 2.21 に示すとおり，初期設定（ポートの設定など）の後に，繰り返し処理が実行される。図 2.22 は Blink というサンプルスケッチであり，13 番ポートに接続された LED（Arduino Uno 上のオレンジの LED）を 1 秒間隔で点滅を繰り返す処理が記述されている。

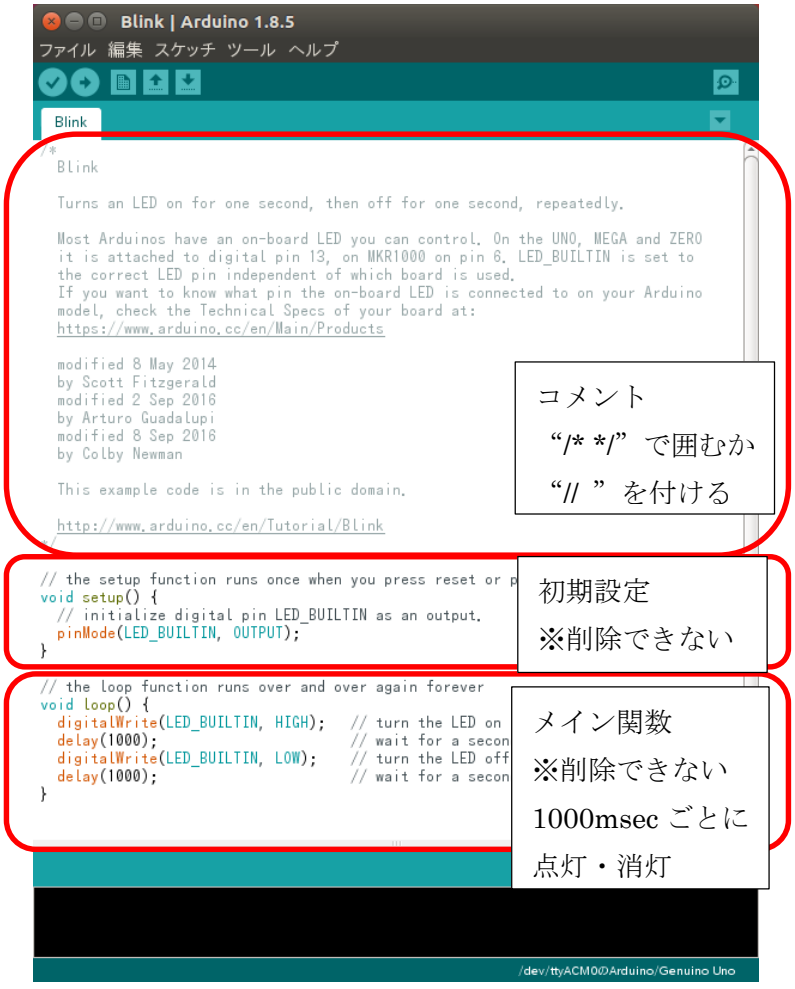


図 2.22 サンプルスケッチ Blink

2.2.8　ディジタル IO ポート

ディジタル信号を扱うポートであり，マイコンの設定により入力ポートまたは出力ポートとして機能する。マイコン内部で 1 本の端子につき 1 ビット（1 か 0 を表現するディジタル回路のデータの単位）のデータを記憶する領域とつながっている。

・出力ポートに設定した場合

ポートに書き込まれた 1 または 0 の値は，それぞれ端子より電圧として 5V または 0V として出力される。出力ポートは一度書き込まれると，次に書き換えられるか電源を切らない限り，同じ状態が維持される。

・入力ポートに設定した場合

端子にかかる電圧レベルの HIGH または LOW を検出して，1 または 0 をポートに反映する。ATmega328P では，0.6～電源電圧 Vcc+0.5V まで（HIGH）を 1 とし，－0.5～0.3V まで（LOW）を 0 とみなす。

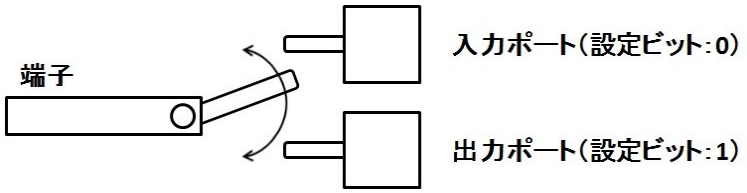


図 2.23 ディジタル入出力ポート

＜演習 2.2.1＞　マイコンによる LED の点滅

Arduino IDE の操作方法およびデジィタル出力の動作を理解するために，サンプルスケッチを使用して，Arduino UNO ボード上の LED（L）を点滅させる。

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

(2) Arduino を起動する。

端末より

$ arduino&

と入力する。

(3) 図 2.24 のように，Arduino 初回起動時にポップアップが表示されるので，新規フォルダ作成をクリックして，”Arduino”と入力してフォルダを作成する。この Arduino フォルダをスケッチの保存場所として指定する。(Arduino 起動後，環境設定によりスケッチの保存場所を確認しておく。)

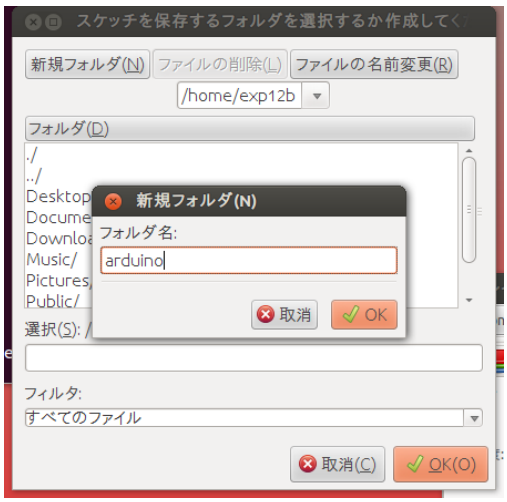
(4) Arduino UNO ボードの選択，シリアルポートの設定をする。

Arduino UNO ボードの選択：

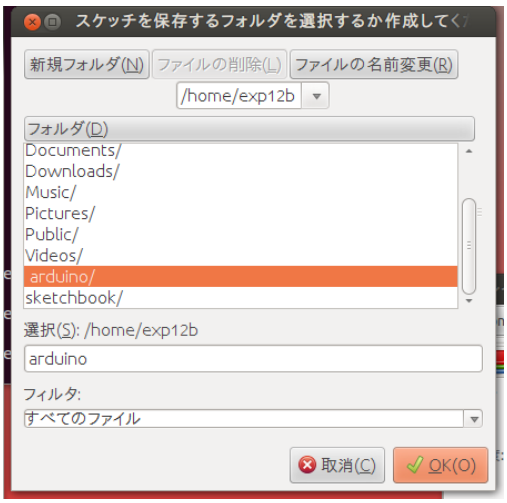
メニューバー：ツール，マイコンボード，Arduino/Genuino UNO

シリアルポートの選択：

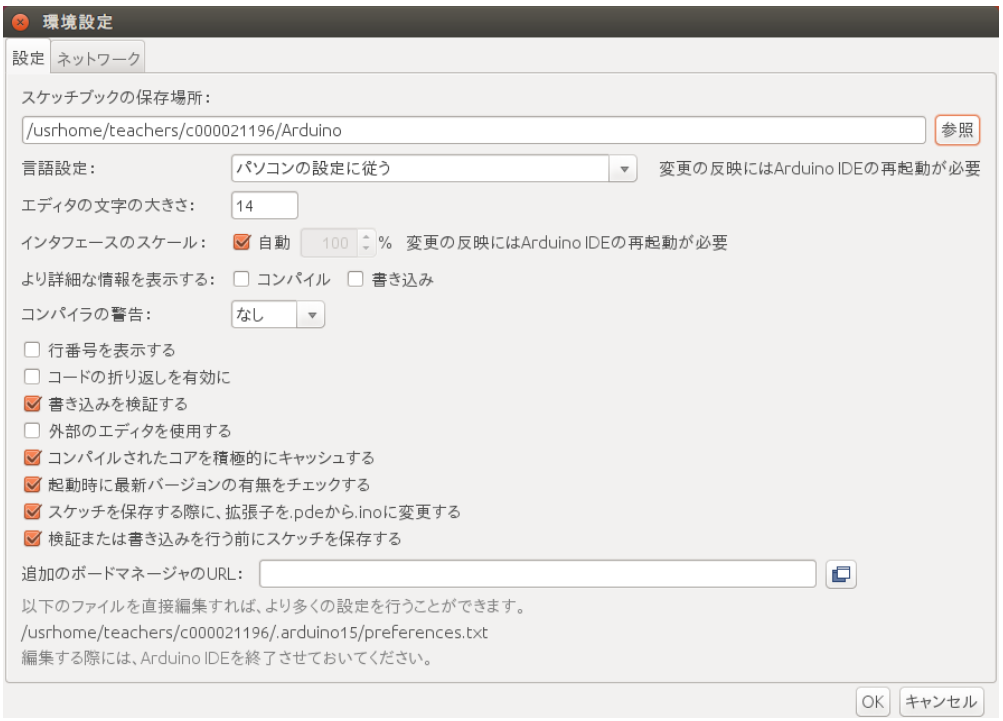
メニューバー：ツール，シリアルポート，/dev/ttyACM0



(a) Arduino フォルダの作成



(b) スケッチ保存先の選択



(c) Arduino 起動後，ファイル→環境設定メニューでスケッチの保存場所を確認

図 2.24 スケッチの保存場所

(5) メニューバー：「ファイル」の「環境設定」を選択し，スケッチの保存場所が Arduino というフォルダであることを確認する（図 2.24(c)を参照）。Arduino フォルダに設定されていない場合，「参照」ボタンをクリックし，Arduino フォルダを必要に応じて作成し，Arduino フォルダを保存場所に設定する。

(6) エディタの文字の大きさを“14”に設定する（図 2.24(c)を参照）。

(7) サンプルスケッチを開く。

ファイル→スケッチの例→01:Basics→Blink を開く。

(8) ファイルメニューの書き込みボタンをクリックする。
(9) Arduino UNO ボードの LED (L) が点滅することを確認する。

2.2.9 Arduino の API

以下に Arduino IDE に用意されている関数のうち，本節で使用する関数について説明する。なお，これらの関数はコンパイル時に実体（ソースファイル）がリンクされ，その都度呼び出され，処理が終了したら元の関数に戻る。

pinMode 関数：入出力ポートの方向を指定する関数である。使用する端子の番号と入出力の方向を引数としてもち，戻り値のない（void）関数である。入力ポートにしたい場合は「INPUT」，出力ポートにしたい場合は「OUTPUT」を半角大文字で記述する。（Arduino IDE は大文字・小文字を区別するので注意すること。）

例：pinMode(13, OUTPUT);　//13 番ポートを出力ポートに設定

digitalWrite 関数：出力ポートに 1 か 0 のディジタル値を書き込む関数である。使用する端子の番号と，1 のときは「HIGH」，0 のときは「LOW」を記述する。戻り値は無い（void）関数である。(digitalWrite を使用するには, pinMode 関数により出力ポートに設定しておく必要がある。)

例：degitalWrite(13, HIGH); //13 番の出力ポートに 1 の値を書き込む

digitalRead 関数：入力ポートの状態（HIGH:5V，LOW：0V）を読み込む関数である。引数にポート番号を取り，入力ポートのレジスタの状態を読み込む関数である。ATmega328P マイコンの初期状態では，出力ポートは 0 が設定されている。

例：int val = digitalRead(13); //13 番の出力ポートの状態を int 型の変数 val に入力する

delay 関数：一定時間の遅延を発生させる関数である。delay 関数の引数は指定した時間が記述され，その時間だけ処理が停止する戻り値の無い（void）関数である。引数で渡す数値の単位は ms（ミリ秒=0.001 秒）である。指定できる数字は 1～4,096,000,000（約 1,137.777 時間）である。

例：delay(1000); //1,000 を引数として記述した場合，1 秒間処理が停止する

以下に使用頻度の高い構文を示す。

for 文：カウンタを使って決まった回数を繰り返すのに便利な構文である。カウンタの初期値，繰り返すための条件，カウンタのカウントの処理を“;”で区切って記述する。

```
********************************
for( カウンタの初期設定; 繰り返すための条件; カウント) {
    /*　繰り返される処理内容　*/
}
********************************
```

if 文: 任意の条件の場合にのみ処理をするのに便利な構文である。条件式を記述する。通常は else と対で記述される。

```
********************************
if ( 条件式 ) {
    /*  処理内容  */
}
else {
}
********************************
```

while 文：任意の条件式が成立する間，処理を繰り返すのに便利な構文である。条件式を記述する。

```
********************************
while( 条件式 ) {
    /*  処理内容  */
}
********************************
```

switch 文：switch で判定した式の値より，該当する値の case の箇所に記述されている処理を実行する。switch の式の値が定数 1 と等しければ処理 1 を実行する。処理 1 の実行後に break 文を事項すると switch 処理は終了する。break 文が無い場合は, そのまま次の処理を実行する。default は，どの case の値とも等しくなかった場合に実行される。

```
********************************
switch( 式 ) {
    case 定数 1:
      処理 1;
      break;
    case 定数 2:
      break;
    default:
      処理 3;
      break;
}
********************************
```

以下に本節で使用する修飾子を示す。

const：変数は参照可能であるが，書き換え不可にするための修飾子である。値が決まっている周波数やポート番号など，定数に使用される。“const”を使用する場合は必ず初期値を設定する必要がある。

例：const int LED_PIN = 13; // “LED_PIN”という文字は数字の 13 を意味すると宣言

＜演習 2.2.2＞　LED の点滅速度を変更

スケッチの書き方および使用する API を理解し，LED の点滅速度を変更する。

(1) サンプルスケッチ（Blink.ino）を開く。

・Blink.ino

```
void setup() { /*  初期設定  */
  pinMode(13, OUTPUT); // 13 番ピンを出力ポートに設定
}
void loop() { /*  無限ループ関数  */
  digitalWrite(13, HIGH);    // 電圧レベルを HIGH に
  delay(1000);               // 1000 ms 間待つ
  digitalWrite(13, LOW);     // 電圧レベルを LOW に
  delay(1000);               // 1000 ms 間待つ
}
```

(2) 0.5 秒毎に点滅させるよう delay 関数の引数を書き換える。

－delay(1000)を delay(500)に書き換える。

(3) 変更したスケッチを名前を変えて保存する（例：Blink_delay500_20190419 など）

(4) スケッチを書き込む。

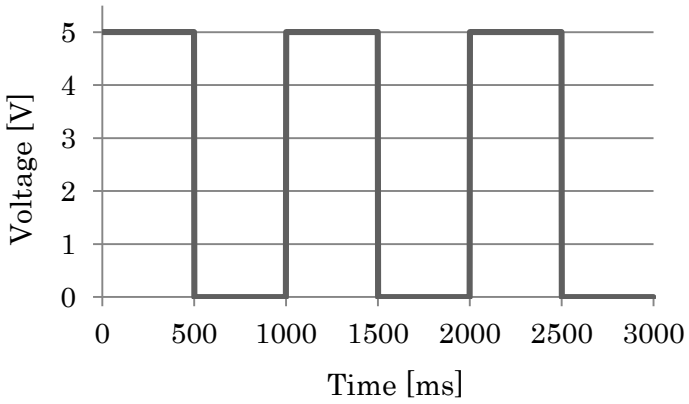


図 2.25 LED 点滅時のディジタル IO ポートから出力波形

＜課題 2.2.1＞　Arduino によるブレッドボード上の LED の点滅

図 2.26 の LED 回路をブレッドボード上に実装し，Arduino UNO の D13 に LED を接続し，ディジタル出力を操作し，1 秒間隔で LED を点滅させるプログラムを作成せよ。

※ボード上の LED（L）と同期して点滅することを確認する。

※LED と電源が繋がっていないことを確認する。

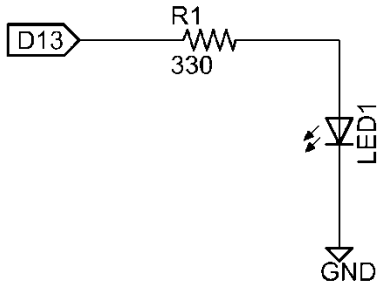


図 2.26　LED 回路

＜課題 2.2.2＞　LED の接続ポートを D3 に変更

課題 2.2.1 のディジタル IO ポートの接続先を D13 から D3 に変更せよ。このとき， ディジタル出力を操作し，LED が 1 秒間隔で点滅するようにプログラムを作成せよ。

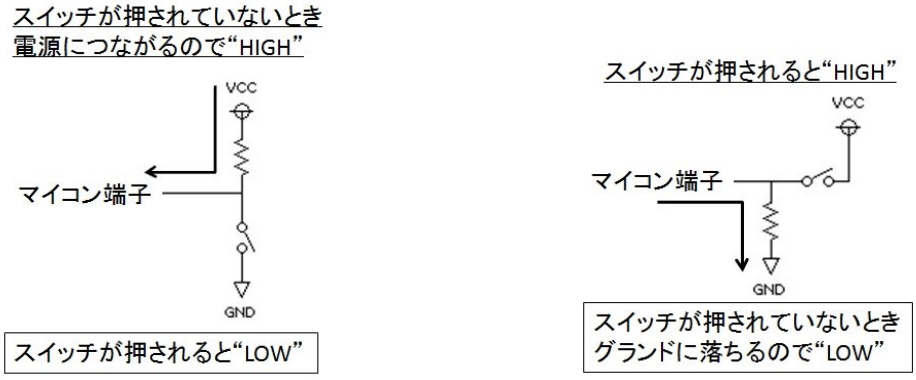
※LED を D3 ポートを接続し，ディジタル出力に設定する

2.2.10　プルアップ抵抗・プルダウン抵抗

ATmega328P では，電圧レベルを 0.6～電源電圧 Vcc+0.5V まで(HIGH)を 1 とし，－0.5～0.3V まで(LOW)を 0 とみなす。HIGH または LOW の電圧レベル以外の電圧を端子に加えた場合には，不安定な状態になりマイコンが誤動作を起こすことがある。そのため，プルアップ抵抗およびプルダウン抵抗を端子に取り付け，HIGH / LOW の状態を一定に保つようにする。

図 2.27(a)(b)はそれぞれプルアップ，プルダウン回路である。プルアップ回路では，スイッチがオフのとき，抵抗を介して電源と接続され，ディジタル入力は HIGH になる。スイッチがオンのとき，スイッチを介して GND とつながるため，ディジタル入力は LOW になる。

プルダウン回路では，プルアップ回路とは逆になり，スイッチがオンのときにディジタル入力は HIGH になり，オフのときに LOW になる。



(a)　　(b)

図 2.27 プルアップ・プルダウン抵抗

＜演習 2.2.3＞　スイッチによる LED の点灯・消灯

ディジタル入力の動作を理解するために，図 2.28 に示す回路をブレッドボード上に実装し，スイッチのオン・オフで Arduino Uno の LED（L）が点灯・消灯するプログラムを作成する（スイッチのオン・オフと連動）。また，作成したスケッチに名前を付けて保存せよ（例：Tr2-2-3_20190419）。

※LED を D13 ポートに設定する。

※ブレッドボード上にスイッチを配置して，D4 ポートに接続する。

※ブレッドボード上に 2 つの回路が別々に構築されていることを確認する。LED とスイッチの回路を繋いではいけない。

例）

```
/* 宣言 */
const int LED_PIN = 13; //　“LED_PIN”という文字は数字の 13 を意味すると宣言
const int SW_PIN = 4; //　“SW_PIN”という文字は数字の 4 を意味すると宣言
int sw1; // スイッチからの入力値を格納する変数
/* 初期設定 */
void setup() {
  pinMode(LED_PIN, OUTPUT); // LED を D13 ポートに設定し，ディジタル出力ポートとする
  pinMode(SW_PIN, INPUT); // スイッチを D4 ポートに設定し，デジィタル入力ポートとする
}
/* メイン関数 */
void loop() {
  sw1 = digitalRead(SW_PIN); // D4 ポートから入力される値(スイッチの状態)を sw1 に入力
  if (sw1 == LOW) { // スイッチ状態が LOW の場合に｛｝内の処理が実行される
    digitalWrite(LED_PIN, HIGH); // D13 ポートに HIGH を出力される(LED が点灯する)
  } else { // スイッチの状態が HIGH の場合に｛｝内の処理が実行される
    digitalWrite(LED_PIN, LOW); // D13 ポートに LOW が出力される(LED が消灯する)
  }
}
```

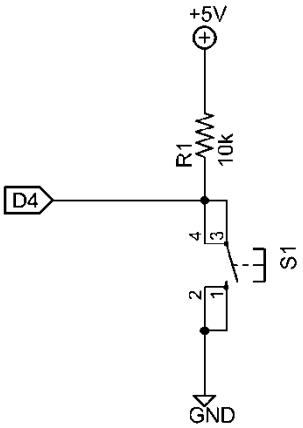


図 2.28　スイッチ動作の回路

＜課題 2.2.3＞　スイッチの状態を LED に表示

スイッチ（入力回路：図 2.28 を参考）を介して，赤色 LED（出力回路：図 2.26 を参考）を点灯・点滅可能な回路図を作成し，ブレッドボード上に実装せよ。また，**スイッチのオンのときに LED が点灯，オフのときに LED が 0.2 秒間隔で点滅するディジタル入出力を操作する**プログラムを作成せよ。

※ブレッドボード上に LED1 個を配置して，D3 ポートに接続し，出力に設定する。

※ブレッドボード上にスイッチを配置して，D4 ポートに接続し，入力に設定する。

※ブレッドボード上に 2 つの回路が別々に構築されていることを確認する。LED とスイッチの回路を繋いではいけない。

＜課題 2.2.4＞　スイッチによる点灯・消灯する LED を変更

1 つの入力ポートに接続されたスイッチのオン・オフで，2 つの出力ポートにそれぞれ接続され 2 つの LED（赤色，黄色）を点灯・消灯させる回路図およびプログラムを作成せよ。

(1) 図 2.26 および図 2.28 を参考に，1 つのスイッチおよび 2 つの LED（赤色，黄色）をそれぞれ任意のディジタルポートに接続する回路図を作成する。このとき，ディジタル I/O ポートを 3 つ（1 つのディジタル入力，2 つのディジタル出力）使用すること。

(2) 回路をブレッドボード上に実装する。このとき，図 2.28 を参考に入力回路を，図 2.26 を参考に出力回路を組むこと。

(3) スイッチのオン・オフで交互に LED が点灯するプログラムを作成し実行する。

※ブレッドボード上に LED1（赤色）を配置して，D3 ポートに接続し，出力に設定する。

※ブレッドボード上に LED2（黄色）を配置して，D13 ポートに接続し，出力に設定する。

※ブレッドボード上にスイッチを配置して，D4 ポートに接続し，入力に設定する。

※ブレッドボード上に 3 つの回路が別々に構築されていることを確認する。

表 2.12：スイッチによる点灯・消灯する LED の状態

SW1	LED1（赤色）	LED2（黄色）
ON	点灯	消灯
OFF	消灯	点灯

＜発展課題 2.2.1＞ for ループを用いて点滅回数を制御

課題 2.2.2 で変更した回路を使用して，D3 ポートに接続した LED が 0.5 秒間隔で 5 回点滅を繰り返した後，2 秒間消灯するディジタル出力を操作するプログラムを for ループを使用して作成せよ（最後の消灯時間は 2.5 秒となる）。

・LED を D3 ポートを接続し，ディジタル出力に設定する

＜発展課題 2.2.2＞　スイッチ 2 個，LED1 個による論理回路

2 つの入力ポートにそれぞれ接続されたスイッチのオン・オフで，1 つの出力ポートに接続され LED（赤色）を点灯・消灯させる回路図およびプログラムを，表 2.13 および表 2.14 の条件を満たすように作成せよ。

(1) 図 2.26 および図 2.28 の回路を参考に，2 つのスイッチと 1 つの LED をそれぞれ任意のディジタルポートに接続する回路を組む。
(2) 表 2.13，表 2.14 の条件を満たすプログラムをそれぞれ作成する。
(3) 表 2.13，表 2.14 を完成させる。

※ブレッドボード上にスイッチ 1 を配置して，D3 ポートに接続し，入力に設定する。
※ブレッドボード上にスイッチ 2 を配置して，D4 ポートに接続し，入力に設定する。
※ブレッドボード上に LED（赤色）を配置して，D13 ポートに接続し，出力に設定する。
※ブレッドボード上に 3 つの回路（スイッチ 1，2 および LED）が別々に構築されていること

表 2.13：AND

SW1	SW2	LED1（点灯/消灯）
ON	ON	
ON	OFF	
OFF	ON	
OFF	OFF	

表 2.14：OR

SW1	SW2	LED1（点灯/消灯）
ON	ON	
ON	OFF	
OFF	ON	
OFF	OFF	

【レポート 2.2　（2019 年 4 月 19 日出題，2016 年 4 月 26 日 12：50 締切）】
※スケッチには詳細の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・変更したスケッチの部分を示すこと。サンプル全てをそのまま貼付しない。

レポート 2.2.1　基礎実験第 2 回の概要

基礎実験第 2 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 2.2.2　マイコンによる LED の点滅

演習 2.2.1 と演習 2.2.2 のスケッチを報告せよ。また，delay 関数の引数を点滅速度との関係について考察せよ。

レポート 2.2.3　Arduino によるブレッドボード上の LED の点滅

課題 2.2.1 において実装した回路図，ブレッドボード配線図およびスケッチを報告せよ。また，LED の動作原理を回路図と作成したプログラムより考察せよ。

レポート 2.2.4　Arduino によるブレッドボード上の LED の点滅

課題 2.2.2 において実装した回路図，ブレッドボード配線図およびプログラムを報告せよ。また，pinMode 関数の役割について考察せよ。

レポート 2.2.5　スイッチの状態を LED に表示

演習 2.2.3 で作成したスケッチを報告せよ。また，digitalRead 関数の役割について考察せよ。さらに，Arduino マイコンとスイッチを接続した場合の，スイッチの動作原理およびマイコンへの入力信号について，回路図とスケッチより説明せよ。

レポート 2.2.6　スイッチの状態を LED に表示

課題 2.2.3 で実装した回路図，ブレッドボード配線図およびスケッチを報告せよ。また，回路図およびスケッチよりスイッチの状態と LED の状態との関係を，ブレッドボード配線図およびスケッチより考察せよ。

レポート 2.2.7　スイッチによる LED 点灯・消灯する LED を変更

課題 2.2.4 で実装した回路図，ブレッドボード配線図および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 2.2.7　for ループを用いて点滅回数を制御（発展課題 2.2.1）

発展課題 2.2.1 のスケッチを報告せよ。また，スケッチ作成で工夫した点を記せ。

レポート 2.2.8　スイッチ 2 個，LED1 個による論理回路（発展課題 2.2.2）

発展課題 2.2.2 で実装した回路図，ブレッドボード配線図およびスケッチを報告せよ。また，各論理演算の式を記し，表 2.13，2.14 の結果を報告せよ。さらに，作成したスケッチで工夫し

た点を記せ。

レポート 2.2.9　回路実装およびプログラム作成状態の報告

うまく実装および作成できたかどうか，どの部分の実装・プログラム作成が失敗または難しかったか？

・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	
3mm 黄緑色LED OSNG3133A	1	
RGBフルカラーLED 5mm 4本足　OSTA3131A　カソードコモン	1	
1/4W 330Ω，510Ω	各3	LED
1/4W 10kΩ	2	スイッチ
1/4W 75Ω，1kΩ，50kΩ，100kΩ，1MΩ	各1	LEDの発光状態
タクトスイッチ	4	

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).