


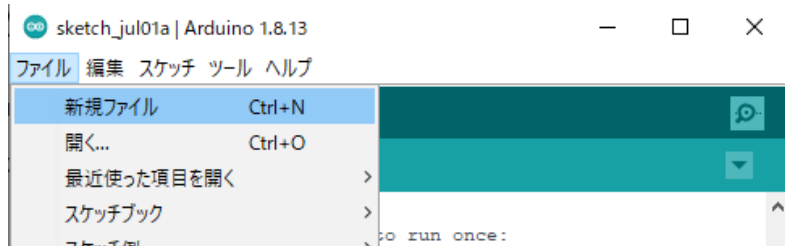
iCar - Arduino入門編

株式会社 カーネル・ソフト・エンジニアリング

Arduinoでプログラムを作成しよう

Arduinoの導入が出来たら、プログラムを作成しましょう。
ここからは、ArduinoでLEDが1秒間隔で点灯するプログラムを作成します。
※このプログラムは、「1_led_1s_on」というファイル名で、サンプルプログラムに含まれています。


- (1) デスクトップまたはスタートメニューより  のアイコンのArduinoを起動します。
- (2) ファイル->新規ファイルをクリックし、下記のプログラムを入力します。

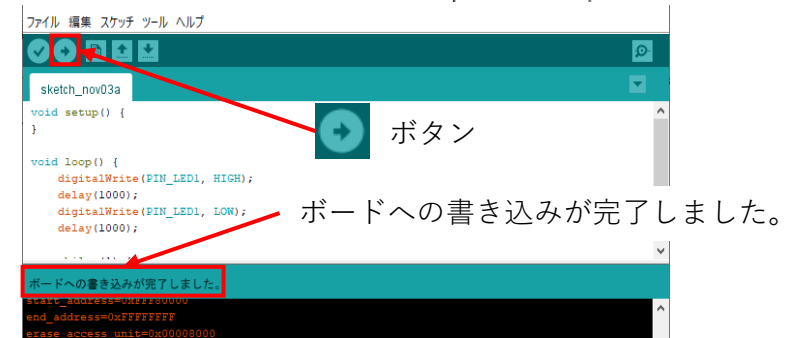
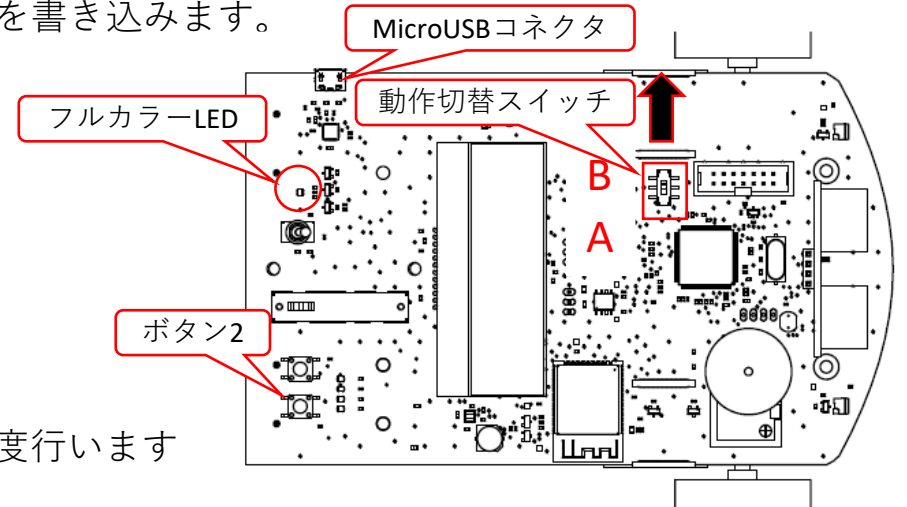


```
void setup() {  
}  
  
void loop() {  
  digitalWrite(PIN_LED1, HIGH);  
  delay(1000);  
  digitalWrite(PIN_LED1, LOW);  
  delay(1000);  
  
  while (1) {}  
}
```

プログラムを書き込みしよう

プログラムが完成したら、下記の手順でiCarにプログラムを書き込みます。

1. iCarの電源をOFFします
2. 動作切替スイッチをB側に切り替えます
3. iCarとPCをUSBケーブルで接続します
4. iCarのボタン2を押しながら電源をONしてプログラム書き込みモードにします
モードに入ると、フルカラーLEDが緑色に点灯します
(LEDが緑色に点灯したらボタン2を離します)
※LEDが緑とならない場合は、電源OFFして4.をもう一度行います
5. Arduino IDEの  ボタンをクリックして、プログラムを書き込みます
6. Arduino IDEの画面に「ボードへの書き込みが完了しました。」と表示されたら、書き込み完了です
7. iCarのボタン2を押す(※1)か、電源OFF→ONする(※2)と書き込んだプログラムが開始します。



※1. iCarのボタン2を押して書き込んだプログラムを開始した場合は、ボタン2を5秒以上押すことで、再度プログラム書き込みモードにすることができます。(フルカラーLEDが緑色に点灯します)

※2. 電源OFF→ONで書き込んだプログラムを開始した場合は、ボタン2を長押ししても、プログラム書き込みモードに入りません。ボタン2の長押しで書き込みモードに入れたくない場合は、プログラム書き込み後に電源OFF→ONしてプログラムを開始します。

LED1秒点灯プログラムを作成しよう(解説1)

前のページまでに作成したLED1秒点灯プログラムを解説します。

<pre>void setup() {</pre>	<p>setup関数 プログラム起動時に1度だけ実行される関数 初期化处理などを記述する (今回は処理なし)</p>
<pre>void loop() {</pre>	<p>loop関数・・・処理を記述する。この関数は繰り返し呼び出される</p>
<pre> digitalWrite(PIN_LED1, HIGH);</pre>	<p>digitalWrite関数を呼び出して、LED1を点灯させる。 PIN_LED1・・・LED1を示す、HIGH・・・点灯を示す</p>
<pre> delay(1000);</pre>	<p>delay関数を呼び出して、1000ミリ秒=1秒待つ</p>
<pre> digitalWrite(PIN_LED1, LOW);</pre>	<p>digitalWrite関数を呼び出してLED1を消灯させる。 PIN_LED1・・・LED1を示す、HIGH・・・点灯を示す</p>
<pre> delay(1000);</pre>	<p>delay関数を呼び出して1000ミリ秒=1秒待つ</p>
<pre> while (1) {}</pre>	<p>無限ループ whileの()の中身が0以外の場合繰り返す ずっと1なのでずっと繰り返し。 (これ以降プログラムを実行しないようにする)</p>
<pre>}</pre>	<p>loop関数終わり</p>

プログラムを変更しよう その1(while(反復)を使う)

前のページで作成した、LED1秒点灯プログラムを、反復処理を使って、LEDがずっと点滅するプログラムに変更してみましょう。
※このプログラムは、「2_led_blink」というファイル名で、サンプルプログラムに含まれています。

プログラムを下記のように変更します

```
void setup() {  
}  
  
void loop() {  
  while (1) {  
    digitalWrite(PIN_LED1, HIGH);  
    delay(1000);  
    digitalWrite(PIN_LED1, LOW);  
    delay(1000);  
  }  
}
```

追加

追加

追加

while (1) {}削除

行頭に半角スペース4つ追加
(もしくはTAB追加)

プログラムが完成したら、実行してみましょう。LED1が1秒点灯、1秒消灯を繰り返すはずですが。

プログラムを変更しよう その1(while(反復)を使う) (解説)

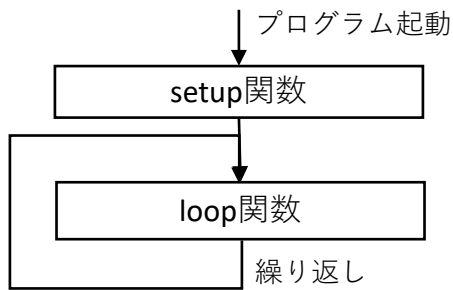
前のページで作成したLEDがずっと点滅するプログラムを解説します。

<pre>void setup() { }</pre>	<p>setup関数 プログラム起動時に1度だけ実行される関数 初期化処理などを記述する (今回は処理なし)</p>
<pre>void loop() {</pre>	<p>loop関数・・・処理を記述する。この関数は繰り返し呼び出される</p>
<pre> while (1) { digitalWrite(PIN_LED1, HIGH); delay(1000); digitalWrite(PIN_LED1, LOW); delay(1000);</pre>	<p>while (1){・・・無限ループ開始 {から}までの間がループ処理の対象</p>
<pre> }</pre>	<p>LED1を点灯→1秒待つ→LED1を消灯→1秒待つの処理 行頭にスペースを入れる変更をしたのは、 インデント(字下げ)して、プログラムを見やすくするためです。 プログラムが見やすくなり、{}の閉じ忘れを発見するのに有用です。 【ルール】 {が出たら、次の行から行頭に記述するスペースの数を増やします。 }が出たら、その行以降行頭に記述するスペースの数を減らします。 ※iCarではスペース4つでインデントを行っています スペース2つやTABであらわしているプロジェクトもあります。</p>
<pre>}</pre>	<p>無限ループ終わり</p>
	<p>loop関数終わり</p>

プログラムを変更しよう その1(while(反復)を使う) (別解)

実は前のページまでに作成した、LEDがずっと点滅するプログラムは、**while**を使わずに単に作成することも出来ます。
 ※このプログラムは、「2_led_blink_another」というファイル名でサンプルプログラムに含まれています。

iCarの起動の流れ



iCarのプログラムは、プログラム起動時に**setup**関数が1度呼び出された後、**loop**関数が呼び出しされます。
loop関数の処理が終わっても、すぐに再び呼び出しされるため、**while**による無限ループ処理を書かなくても、LEDの点滅処理だけ書くことで、ずっとLED点滅を実現することが出来ます。
 逆に、**loop**関数最後まで実行したら、それ以降プログラムを実行してほしくない場合は**loop**関数最後に無限ループを記述します。(使用例：LED1秒点灯プログラム)

※本動作は、上位のモジュールで、**while (1) {loop();}**のような処理があることで実現しています。

```
void setup() {
}
```

setup関数 プログラム起動時に1度だけ実行される関数
 初期化処理などを記述する (今回は処理なし)

```
void loop() {
```

```
  digitalWrite(PIN_LED1, HIGH);
  delay(1000);
  digitalWrite(PIN_LED1, LOW);
  delay(1000);
```

loop関数・・・処理を記述する。この関数は繰り返し呼び出される

LED1を点灯→1秒待つ→LED1を消灯→1秒待つの処理

```
}
```

loop関数終わり

プログラムを変更しよう その2(if(分岐)を使う)

前のページで作成したLEDがずっと点滅するプログラムを、分岐処理を使って、ボタン1を押している時だけ、LEDが点滅するプログラムに変更してみましょう。

※このプログラムは、「3_until_button_led_blink」というファイル名でサンプルプログラムに含まれています。

```
void setup() {  
}  
  
void loop() {  
  while (1) {  
    if (digitalRead(PIN_BTN1) == HIGH) {  
      digitalWrite(PIN_LED1, HIGH);  
      delay(1000);  
      digitalWrite(PIN_LED1, LOW);  
      delay(1000);  
    }  
  }  
}
```

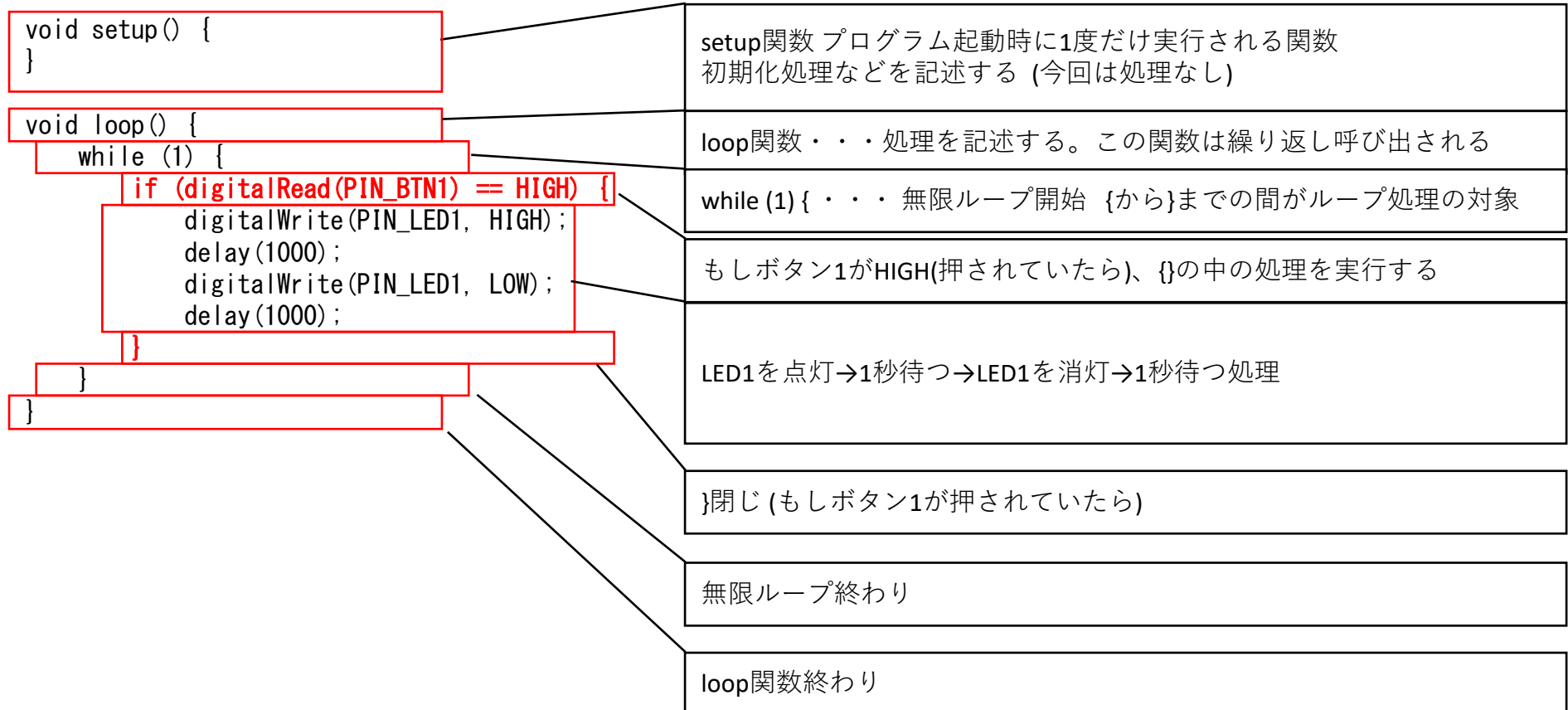
The diagram shows the code from the previous block with three callout boxes pointing to specific lines:

- A box labeled "追加" (Add) points to the line `if (digitalRead(PIN_BTN1) == HIGH) {`.
- A box labeled "行頭に半角スペース4つ追加 (もしくはTAB追加)" (Add 4 half-angle spaces to the line head (or add TAB)) points to the first line of the if block: `digitalWrite(PIN_LED1, HIGH);`.
- A box labeled "追加" (Add) points to the closing brace of the if block: `}`.

プログラムが完成したら、実行してみましょう。ボタン1を押している間だけ、LED1が1秒点灯、1秒消灯を繰り返す動作となるはずです。

プログラムを変更しよう その2(if(分岐)を使う)

前のページで作成したボタン1を押している時だけ、LEDが点滅するプログラムを解説します。



iCar関数一覧 (1/2)

iCarで使用できる関数の一覧 (使用頻度が高いものを抜粋して説明しています)

使用可能な全関数については、iCar製品ホームページで公開している、関数リファレンスを参照してください。
(関数リファレンスを解凍し、indexファイルを開くと、リファレンスをブラウザで開くことができます)

関数	機能	引数
<code>digitalWrite(PIN_LEDn, HIGH/LOW);</code>	LEDを点灯・消灯させます	1: Pin (操作するLED番号) PIN_LEDn n=LED番号 2: HIGH/LOW . . . (HIGH=点灯, LOW=消灯)
<code>analogWrite(PIN_BUZZER, freq);</code>	ブザーを鳴らします	1: Pin (PIN_BUZZER固定) 2: freq . . . 周波数 (8~16000) ※範囲外は停止
<code>analogWrite(PIN_MOTOR_n, spd);</code>	モータL, Rを指定出力で動かします	1: Pin (動かすモータ) PIN_MOTOR_n n=L=左, n=R=右 2: spd . . . 速度(%) (-100=後退~0=停止~100=前進) ※範囲外指定でブレーキをかける
<code>analogWrite(PIN_FLED_n, duty);</code>	フルカラーLEDの明るさを設定します	1: Pin (操作するLED) PIN_FLED_n n=R=赤, n=G=緑, n=B=青 2: duty . . . 明るさ(0(消灯)~100(%))
<code>LcdDrv_update();</code>	LCDを更新します ※本関数を呼び出して初めて更新が反映される	なし
<code>LcdDrv_clear();</code>	LCDをクリアします	なし
<code>LcdDrv_print("text");</code>	LCDに文字を表示します	1: 文字列 (char *型)
<code>LcdDrv_setNum(num, digit);</code>	LCDに正の10進数の値を表示します ※表示桁を超えた桁は切り捨てされる	1: num . . . 表示値 2: digit . . . 表示桁
<code>LcdDrv_setHex(num, digit);</code>	LCDに16進数の値を表示します ※表示桁を超えた桁は切り捨てされる	1: num . . . 表示値 2: digit . . . 表示桁
<code>LcdDrv_setCursor(y, x);</code>	LCDに書き込む文字の位置を指定します	1: y=縦位置 2: x=横位置
<code>delay(n);</code>	nミリ秒間待機します	1: n . . . 待機時間 (ミリ秒)

iCar関数一覧 (2/2)

iCarで使用できる関数の一覧 (使用頻度が高いものを抜粋して説明しています)

使用可能な全関数については、iCar製品ホームページで公開している、関数リファレンスを参照してください。
(関数リファレンスを解凍し、indexファイルを開くと、リファレンスをブラウザで開くことができます)

関数	機能	戻り値
analogRead(PIN_DISTANCE)	前方の物との距離を取得する	前方障害物との距離 [mm] (6027・・・未検出)
analogRead(PIN_BRIGHTNESS)	周囲の明るさを取得する	周囲の明るさ (0=明るい~4095=暗い) (※未確定時も4095)
analogRead(PIN_VOLUME)	スライドボリュームの位置を取得	スライドボリュームの位置 (0=後方~100=前方)
analogRead(PIN_LINE_L)	ラインセンサLの明るさを取得	ラインセンサLの直下の明るさ (0=明るい~4095=暗い)
analogRead(PIN_LINE_R)	ラインセンサRの明るさを取得	ラインセンサRの直下の明るさ (0=明るい~4095=暗い)
analogRead(PIN_TOGGLE)	スイッチの状態を取得	スイッチの位置 (0=後退方向, 1=中点, 2=前進方向)
digitalRead(PIN_BTN1)	ボタン1の状態を取得	1=押されている, 0=押されていない
digitalRead(PIN_BTN2)	ボタン2の状態を取得	1=押されている, 0=押されていない
millis()	プログラム起動からのミリ秒を返す	プログラム起動時からの時間 (ミリ秒)
tone(PIN_BUZZER, freq, period)	ブザーを一定時間鳴らします	1: Pin (PIN_BUZZER固定) 2: freq・・・周波数 (8~16000) ※範囲外は停止 3: period・・・鳴らす時間 ミリ秒 (0=ずっと)
noTone(PIN_BUZZER)	ブザーを止めます	1: Pin (PIN_BUZZER固定)

iCar Pin一覧

iCarでは前のページの関数一覧に記載されているように、各デバイスを種類に応じた制御関数(`digitalWrite`, `digitalRead`, `analogWrite`, `analogRead`)呼び出すことで制御します。関数呼び出し時に指定するPinについて下記のようになっています。

[AI]・・・`analogRead`関数で操作、 [DI]・・・`digitalRead`関数で操作
[AO]・・・`analogWrite`関数で操作、 [DO]・・・`digitalWrite`関数で操作

[AI] スライドボリューム	Pin0	Pin19	[AO] フルカラーLED B
[AI] 距離センサ	Pin1	Pin18	[AO] フルカラーLED G
[AI] 明るさセンサ	Pin2	Pin17	[AO] フルカラーLED R
[AI] ラインセンサ L	Pin3	Pin16	[AO] ブザー
[AI] ラインセンサ R	Pin4	Pin15	[AO] モーターR
[AI] トグルスイッチ(SW_1)	Pin5	Pin14	[AO] モーターL
[DI] ボタン1(BTN_1)	Pin6	Pin13	[DO] LED6
[DI] ボタン2(BTN_2)	Pin7	Pin12	[DO] LED5
[DO] LED1	Pin8	Pin11	[DO] LED4
[DO] LED2	Pin9	Pin10	[DO] LED3

Note: 実際のマイコンのPinと変換する処理をボードのライブラリ内で行っているため、マイコンのPin番号とは異なります。