「やってみよう プログラミング」 補助資料 －ドリトル編－

プログラミング言語「ドリトル」は兼宗 進
（大阪電気通信大学）によって開発されました。
http://dolittle.eplang.jp/

● p20 ② オブジェクトに命令する（反復）

```
かめた＝タートル！作る。
「
    かめた！１００　歩く。
    かめた！１４４　右回り。
」！５　繰り返す。
```

● p21 ④ 作った部品（図形）を使う

```
かめた＝タートル！作る。
「
    かめた！１００　歩く　１４４　右回り。
」！５　繰り返す。
星＝かめた！（黄色）図形を作る。
星！－１００　１００　移動する。
星！２０　右回り。
```

● p21 ① タートルを動かす

```
かめた＝タートル！作る。
時計＝タイマー！作る。
時計！「
    かめた！１００　歩く。
    かめた！１４４　右回り。
」実行。
```

● p21 ② 0.5 秒毎に 10 秒間命令を実行

```
かめた＝タートル！作る。
時計＝タイマー！作る。
時計！０．５秒　間隔。
時計！１０秒　時間。
時計！「
    かめた！１００　歩く。
    かめた！１４４　右回り。
」実行。
```

● p21　② １秒毎に５回命令を実行

```
かめた＝タートル！作る。
時計＝タイマー！作る。
時計！１秒　間隔。
時計！５回　回数。
時計！「
    かめた！１００　歩く。
    かめた！１４４　右回り。
」実行。
```

● p22　③ 図形を動かす

```
かめた＝タートル！作る。
「
    かめた！１００　歩く　１４４　右回り。
」！５　繰り返す。
星＝かめた！（黄色）図形を作る。
時計＝タイマー！作る。
時計！「
    星！１０　－１０　移動する。
    星！２０　右回り。
」実行。
```

● p22　Challenge　「違う形の図形を描いてみよう」　プログラム例（四角形の例）

```
かめた＝タートル！作る。
「
    かめた！１００　歩く　９０　右回り。
」！４　繰り返す。
```

● p22　Challenge　「図形を違う方向へ移動させてみよう」　プログラム例（左上へ移動）

```
かめた＝タートル！作る。
「
    かめた！１００　歩く　１４４　右回り。
」！５　繰り返す。
星＝かめた！（黄色）図形を作る。
時計＝タイマー！作る。
時計！「
    星！－１０　１０　移動する。
    星！２０　左回り。
」実行。
```

● p22　Challenge　「図形をゆっくり動くようにしてみよう」　プログラム例

```
かめた＝タートル！作る。
「
    かめた！１００　歩く　１４４　右回り。
」！５　繰り返す。
星＝かめた！（黄色）図形を作る。
時計＝タイマー！作る。
時計！「
    星！２　－２　移動する。
    星！５　右回り。
」実行。
```

● p22　ボタンを押すと図形が赤くなる

```
かめた＝タートル！作る。
「
    かめた！１００　歩く　１４４　右回り。
」！５　繰り返す。
星＝かめた！（黄色）図形を作る。

赤色ボタン＝ボタン！”赤色”作る。
赤色ボタン：動作＝「
    星！（赤）塗る。
」。
```

● p24　送信プログラム（完成）

```
サーバー！”localhost”　接続。
かめた＝タートル！作る。

「かめた！４０　歩く　１２０　右回り。」！３　繰り返す。
三角＝かめた！（赤）図形を作る。
サーバー！”sankaku”　（三角）書く。

「かめた！４０　歩く　９０　右回り。」！４　繰り返す。
四角＝かめた！（青）図形を作る。
サーバー！”shikaku”　（四角）書く。
```

● p25　受信プログラム（完成）

```
サーバー！”localhost”　接続。

名前入力エリア＝フィールド！作る。
呼び出しボタン＝ボタン！”呼び出し”　作る。

名前＝”未設定”。
呼び出しボタン：動作＝「
    「名前！＝”未設定”」！なら「
        受信内容！消える。
    」実行。

    名前＝名前入力エリア！読む。
    受信内容＝サーバー！（名前）読む。
」。
```

● p26　送信プログラム

```
サーバー！”localhost”　接続。
かめた＝タートル！作る。

かめた：絵を描く＝「
帆＝自分！
      ９０　左回り　１００　歩く
    １５０　右回り　　９０　歩く　閉じる
    （黄色）図形を作る。

船＝自分！
      ６０　左回り　　３０　歩く　　９０　左回り　１０　歩く
      ９０　右回り　　５０　歩く　１２０　右回り　４０　歩く
      ６０　右回り　１００　歩く　　６０　右回り　４０　歩く
    １２０　右回り　　５０　歩く　　９０　右回り　１０　歩く
      ９０　左回り　閉じる
    （赤）図形を作る。
」。

かめた！絵を描く。

サーバー！”kameta”　（かめた）書く。
```

● p26　受信プログラム

```
サーバー！”localhost”　接続。

名前入力エリア＝フィールド！作る。
呼び出しボタン＝ボタン！”呼び出し”　作る。

名前＝”未設定”。
呼び出しボタン：動作＝「
    「名前！＝”未設定”」！なら「
        受信内容！消える。
    」実行。

    名前＝名前入力エリア！読む。
    受信内容＝サーバー！（名前）読む。
    受信内容！絵を描く。
」。
```

● p28　LED の点灯プログラム

```
システム！”studuino”　使う。
最初に実行＝「
    ＳＴ！”Ａ４”　デジタルＬＥＤ。
」。
繰り返し実行＝「
    ＳＴ！”Ａ４”　１　書く。
    ＳＴ！１０００　待つ。
    ＳＴ！”Ａ４”　０　書く。
    ＳＴ！１０００　待つ。
」。
ＳＴ！転送。
```

● p29　赤外線センサーに手をかざすと LED が点灯するプログラム

```
システム！”studuino”　使う。
最初に実行＝「
    ＳＴ！”Ａ４”　デジタルＬＥＤ。
    ＳＴ！”Ａ０”　赤外線センサー。
」。
繰り返し実行＝「
    「（ＳＴ！”Ａ０”　読む）＞１００」！　なら「
        ＳＴ！”Ａ４”　１　書く。
    」そうでなければ「
        ＳＴ！”Ａ４”　０　書く。
    」実行。
」。
```

● p29 Challenge 「一度点灯したら，しばらく点灯し続けるようにしてみよう」 プログラム例

```
システム！”studuino” 使う。
最初に実行＝「
    ＳＴ！”Ａ４” デジタルＬＥＤ。
    ＳＴ！”Ａ０” 赤外線センサー。
」。
繰り返し実行＝「
    「(ＳＴ！”Ａ０” 読む）＞１００」！ なら「
        ＳＴ！”Ａ４” １ 書く。
        ＳＴ！５０００ 待つ。
    」そうでなければ「
        ＳＴ！”Ａ４” ０ 書く。
    」実行。
」。
```

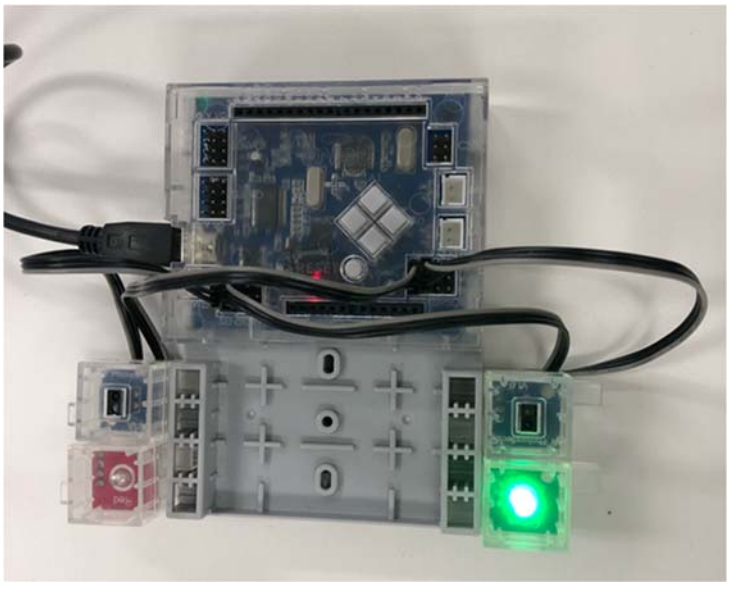
ポイント

LED の点灯命令のあとに「待つ」を入れることで，LED が点灯した状態で 5000ms 待機する。

● p29　Challenge　「LED と赤外線センサを追加して，手がかざされていない時だけ追加した LED が光るようにしてみよう」　プログラム例

```
システム!“studuino” 使う。
最初に実行＝「
    ＳＴ！“Ａ４”　デジタルＬＥＤ。
    ＳＴ！“Ａ５”　デジタルＬＥＤ。
    ＳＴ！“Ａ０”　赤外線センサー。
    ＳＴ！“Ａ１”　赤外線センサー。
」。
繰り返し実行＝「
    「(ＳＴ！“Ａ０”　読む）＞１００」！　なら「
        ＳＴ！“Ａ４”　１　書く。
    」そうでなければ「
        ＳＴ！“Ａ４”　０　書く。
    」実行。
    「(ＳＴ！“Ａ１”　読む）＞１００」！　なら「
        ＳＴ！“Ａ５”　０　書く。
    」そうでなければ「
        ＳＴ！“Ａ５”　１　書く。
    」実行。
」。
ＳＴ！転送。
```

接続例



ポイント

追加したセンサーの条件「(ＳＴ！“Ａ１”　読む）＞１００」が成り立った時に消灯するようになっている。

● p30　3.1 自動停止ロボット

```
システム！”studuino”　使う。
最初に実行＝「
    ＳＴ！ＤＣモーター。
    ＳＴ！”Ａ０”　赤外線センサー。
」。
繰り返し実行 ＝「
    「(ＳＴ！”Ａ０”　読む）＞５０」！　なら「
        ＳＴ！停止。
    」そうでなければ「
        ＳＴ！前進。
    」実行。
」。
ＳＴ！転送。
```

● p30　3.2 ライントレーサ

```
システム！”studuino”　使う。
最初に実行＝「
    ＳＴ！ＤＣモーター。
    ＳＴ！”Ａ１”　赤外線センサー。
」。
繰り返し実行＝「
    「(ＳＴ！”Ａ１”　読む）＞１００」！　なら「
        ＳＴ！左折。
    」そうでなければ「
        ＳＴ！右折。
    」実行。
」。
ＳＴ！転送。
```

● p31　3.3 荷物運びロボット

```
システム！”studuino”　使う。
最初に実行＝「
    ＳＴ！ＤＣモーター。
    ＳＴ！”Ａ０”　赤外線センサー。
    ＳＴ！”Ａ１”　赤外線センサー。
    ＳＴ！”Ｄ９”　サーボモーター。
    ＳＴ！”Ｄ９”　９０　書く。
」。
繰り返し実行＝「
    「(ＳＴ！”Ａ０”　読む）＞５０」！　なら「
        ＳＴ！停止。
        ＳＴ！”Ｄ９”　０　書く。
    」そうでなければ「
        ＳＴ！”Ｄ９”　９０　書く。
        「(ＳＴ！”Ａ１”　読む）＞１００」！　なら「
            ＳＴ！左折。
        」そうでなければ「
            ＳＴ！右折。
        」実行。
    」実行。
」。
ＳＴ！転送。
```

● p31 Challenge 「荷物をおろしたあと，振り返ってコースを戻るようにしてみよう」 プログラム例

```
システム！“studuino” 使う。
最初に実行＝「
    ＳＴ！ＤＣモーター。
    ＳＴ！“Ａ０” 赤外線センサー。
    ＳＴ！“Ａ１” 赤外線センサー。
    ＳＴ！“Ｄ９” サーボモーター。
    ＳＴ！“Ｄ９” ９０ 書く。
」。
繰り返し実行＝「
    「(ＳＴ！“Ａ０” 読む）＞５０」！ なら「
        ＳＴ！停止。
        ＳＴ！“Ｄ９” ０ 書く。
        ＳＴ！１０００ 待つ。
        ＳＴ！“Ｄ９” ９０ 書く。
        ＳＴ！後進。
        ＳＴ！１０００ 待つ。
        ＳＴ！左折。
        ＳＴ！３０００ 待つ。
    」そうでなければ「
        ＳＴ！“Ｄ９” ９０ 書く。
        「(ＳＴ！“Ａ１” 読む）＞１００」！ なら「
            ＳＴ！左折。
        」そうでなければ「
            ＳＴ！右折。
        」実行。
    」実行。
」。
ＳＴ！転送。
```

ポイント

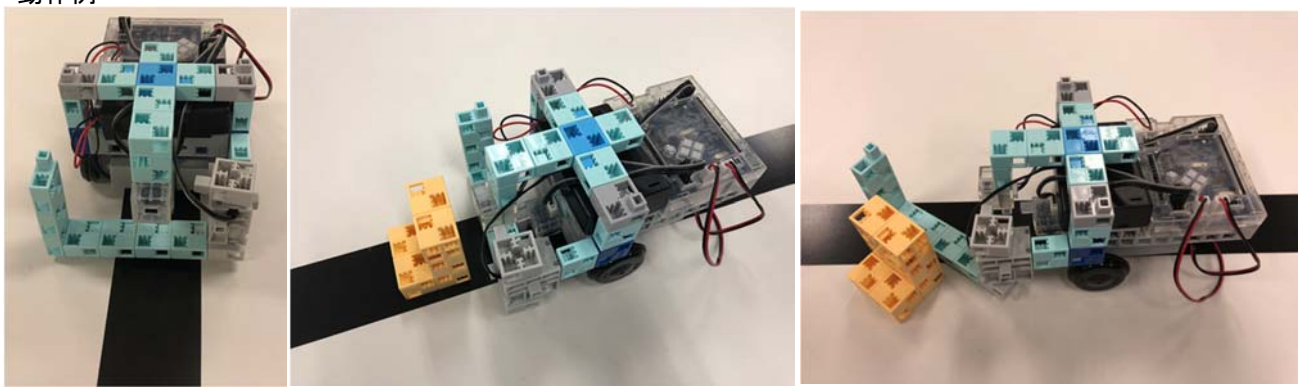
・モーターの制御時間はロボットにより異なる。
・荷物をおろしてから，逆走するときに壁に衝突しないように後進する
・コースの逆走のために，左折命令のあとに「ＳＴ！ ３０００ 待つ。」を入れることでロボットの向きを逆にすることができる。
・3000ms の時間を調整することで，180 度だけでなく 360 度回転することもできる。

● p31　Challenge　「サーボモーターを利用して，障害物をどけるロボットを作ってみよう」

プログラム例

```
システム!“studuino”　使う。
最初に実行＝「
    ＳＴ！ＤＣモーター。
    ＳＴ！“Ａ０”　赤外線センサー。
    ＳＴ！“Ａ１”　赤外線センサー。
    ＳＴ！“Ｄ９”　サーボモーター。
    ＳＴ！“Ｄ９”　１８０　書く。
」。
繰り返し実行＝「
    「(ＳＴ！“Ａ０”　読む）＞５０」！　なら「
        ＳＴ！停止。
        ＳＴ！“Ｄ９”　９０　書く。
        ＳＴ！５００ 待つ。
    」そうでなければ「
        ＳＴ！“Ｄ９”　１８０　書く。
        「(ＳＴ！“Ａ１”　読む）＞１００」！　なら「
            ＳＴ！左折。
        」そうでなければ「
            ＳＴ！右折。
        」実行。
    」実行。
」。
ＳＴ！転送。
```

動作例



ポイント

・モーターの制御時間はロボットにより異なる。
・壁を検知して荷物を下ろすロボットを改良し，サーボモーターをロボット前面につけている
・壁を認識したら，サーボモーターを 90 度に動かすことで，障害物をどけるように動く。
・90 度動く命令のあとに「ＳＴ！　５００　待つ。」がない状態だと，サーボモーターが 90 度に動く途中で，180 度に動く命令が走ってしまう。