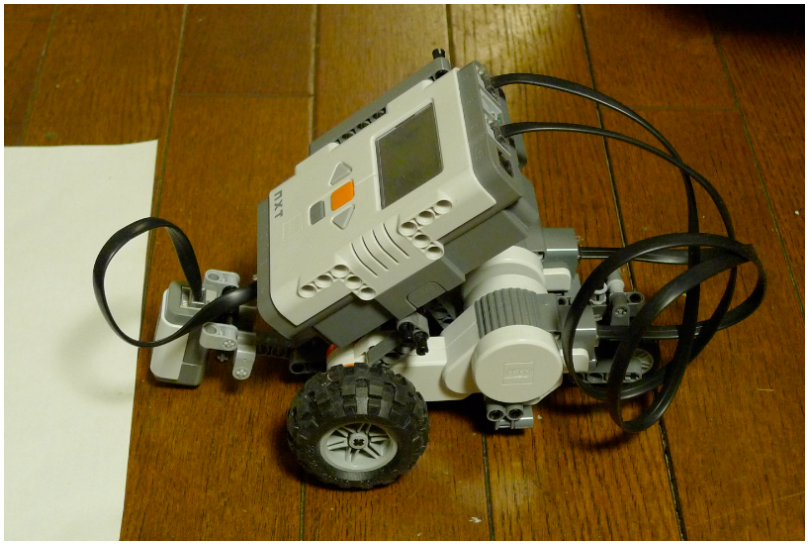


コンピュータ制御

2015/09/15

愛知大学

有澤　健治

Lesson 1 (準備)

運営ルール

2人1組 (座席指定、ノートパソコン指定、ロボットセット指定)

注意

・他のクラスと教材が共用になる
・毎回課題が出る
・課題報告の提出で出席扱い (課題をこなして)
・遅刻の扱い (10分の遅刻は欠席扱い)
・休みの扱い (4回休むとアウト)

休んだ場合

・次の週までに補習
・補習を怠ったら 10 点減点
・場所:、方法
・課題報告を出す (課題をこなして、 次回講義までに、僕に直接渡す事)

毎回の終了時の確認

・ロボットの電源 (ON になっていないか?)
・ロボットの部品 (床に落ちていないか?)

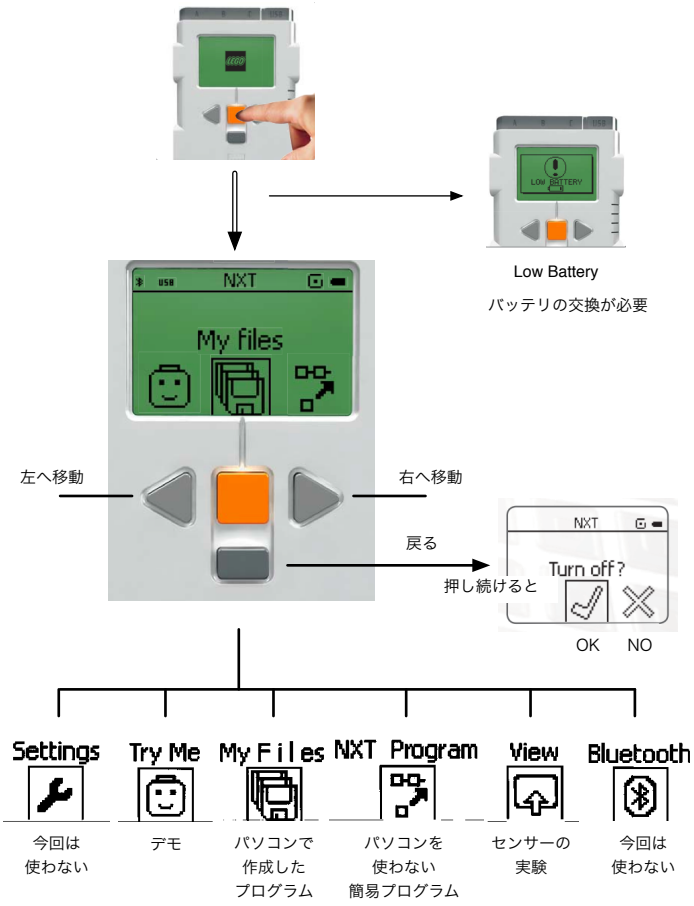
教材

・LEGO Kit
・Users Guide (LEGO Kit 附属のもの)
・ノートPC

今日の作業

教材(LEGO)の動作確認　　# パソコンは使わない
バッテリーの確認　　# 赤いボタンでメニューが出るか?
メニューの見方
終了と電源 OFF

メニューチャート



用語

LEGO NXT

Lesson 2 (Try Me)

今日の作業
パソコンは使わない
Lego Kit のみ

動作確認
Try Me
1. Try Touch
2. Try Light
3. Try Ultrasonic
4. Try Sound
5. Try Motor
注: テキストとは図が異なる

注意事項
課題の内容に応じて適切なセンサーを取り付け、正しく配線を行う事。
バッテリがへたっている場合には Try Motor でのモーターの動きが悪くなる。(⇒ 交換)

課題
以下の Try Me を実行し、観察し気付いた事を書きなさい。

1. Try Touch
タッチした時に起こる事

2. Try Light
白い紙と黒い紙のセンサーの反応の違い

3. Try Ultrasonic
センサーの前に手を置く、
手との距離を変えてみる
音の変化を確認する

4. Try Sound　　# モーターをBとCに
サウンドセンサーに音を拾わせる(手を叩く、口笛を吹く)

5. Try Motor　　# モーターのケーブルの接続先(メニューに注意)
車を手で動かして音の変化を確認する
モーターの動きを捉えていることが分かる

Lesson 3 (View)

今週の課題は?
View にはセンサーの働きが数値で分かる実験が揃っている。後に、センサーの反応に応じてモーター等を動かすプログラムを作ることになるが、今回の実験はその時に役に立つはずである。
View のメニューに関する詳しいチャートが PDF 版の “Users Guide” p.22 に載っている*。

Sound
音の強度を測る
Sound (dB)　　# 物理的な音の強さ
Sound (dBA)　　# 人が感じる音の強さ、今回は使わない

Light
光の強さを測る (色は識別できない)
Reflected light (反射光)　# センサーに付いている LED が光り、この光の反射を見ている
Ambient light (環境からの光)　　# LED は光らない
Light Sensor

Temperature
温度を測る
省略(温度センサーが無い)
Temperature (°C)　# Celsius
Temperature (°F)　# Fahrenheit's、今回は使わない (今でも英米圏で使用されている単位)

Rotation
モーターの回転量を測る
Rotation*
Motor (rotations)　# 1回転 = 1
Motor (degrees)　# 1回転 = 360 (degree)

Touch
接触した事を感知する
Touch

Ultrasonic
超音波を使って距離を測る
Ultrasonic (inch)　# inch 単位、今回は使わない。 1 inch = 2.54 cm
Ultrasonic (cm)　　# cm 単位

注 *: http://ar.aichi-u.ac.jp/lecture/lego/ に
Users Guide、NXC Guide、NXC tutorial、および昨年度のテキスト
が置かれている。「コンピュータ制御」用に準備したノートパソコンはネットに接続されていないので、机の中にあるコンピュータを使ってアクセスし、ダウンロードしたらよい。

Lesson 4 (NXT Program)

NXT Program

NXT Program では極く簡単なプログラムを初等的な方法で作成でき、そして、それを実行できる。プログラムの作成にはパソコンは必要ではない。アイコンを並べてプログラムができあがる。

今回の課題

ページ番号は Users Guide (冊子版)
課題1. (p.23)
課題2. (p.27)
課題3. (p.31)
課題4. (p.35)
課題5. (p.39) # light が必要。携帯電話の画面の光で代用してみる。
課題6. (p.45)

NXT program の作り方: Users Guide (PDF版) p.15, p.16
NXT Program の読み方: Users Guide (PDF版) p.21

次の図(アイコン)で示される NXT プログラムは

1. 前進して時計回り(右回り)にターン
2. 音を感知すると
3. 後進して時計回り(右回り)にターン
4. 音を感知すると
5. 繰り返す(1から4を)

この例に習って課題1から課題6の NXT プログラムの意味を日本語で書きなさい。
(動作を確認しながら意味を理解したらよい。意味が分かるように書けばよい)

例題(課題)の NXT プログラムはどれも5個のアイコンから構成されているのだが.... 5個でなくてはならないのか?
空白アイコン(Empty) の説明が無いのだが.... 推理してみる。
数字が書かれているアイコンがあるが、この数字に関する説明がない。数字の意味は何だろう? 推理してみる。

Lesson 5 (Linux)

今後はロボットのプログラムをパソコン(ノートPC)で行う。ロボット上で直接プログラムするのに比べて、(a) 複雑なプログラムが可能である事、(b) プログラムを残す事ができる事、などの利点がある。ロボットのプログラムは Windows、Mac、Linux の上で可能だが、ここでは Linux を使う。Windows, Mac には子どもでも楽しめるように、初等的な方法でプログラムできるソフトが存在する(Users Guide PDF 版 p.46 ~ p.57)が我々はそのようなソフトには頼らない。

Linux について

Unix の一種(Unix には多数の変種が存在する)
Unix とは
　　1970年頃、Bell 研究所
Unix からの派生 OS　　# Mac OSX を除き、無料、ソース公開
・Linux # Xubuntu(Ubuntu からの派生OS) など
・Free BSD　# Mac OSX は FreeBSD を Apple 流にアレンジ

なお、実習用にインストールされている Linux は Xubuntu であるが、最小構成になっており、実習に必要の無いものは捨てられている。

Linux 標準教科書

近頃無料で質の良い Linux のテキストが現れた。PDF 版である。
http://www.lpi.or.jp/linuxtext/text.shtml

ユーザ名/パスワード

	クラス A	クラス B
ユーザ名	mei	sat
パスワード	totoro	totoro

注意: 打ち込んだパスワードは見えない

ダウンロード「Xubuntu の使い方」

http://ar.aichi-u.ac.jp/lecture/lego/ に 「xubuntu の使い方」

用語

　　ディレクトリ=フォルダ

マウス操作

右クリック、左クリックが基本。中央クリック(ホイールを強く押す)も使えるが、特殊。
新たにファイルを作るとき
　　マウスカーソルを、(作りたいフォルダの)空きスペースに置く。
端末(Terminal)を開くとき
　　マウスカーソルを、(プログラムが置かれているフォルダの)空きスペースに置く。

ホームディレクトリ (家のアイコン)

実習で作ったプログラムなどの置き場所

ファイルの名付け規則

英字(アルファベット)、数字、下線記号(“_”)、ハイフン(“-“)、ピリオド(“.”)
これ以外も使えるが推奨しない。(注意が必要になる)
英字は大文字と小文字が区別される(Windows の場合には区別されない)

コマンド

Unix 系OSのコマンド環境は非常に良くできている。
とりあえず記憶して欲しいコマンド二つ。

```
pwd             # print working directory (作業場所を知る)
ls -l           # 作業場所にあるファイルの一覧を表示する
```

終了の方法

終了のメニュー(logout)に従って終了(shut down)する事。
最後に電源を OFF にする。

Lesson 6 (NXC Program 1)

例題

```
task main()
{
  OnFwd(OUT_A, 75);
  OnFwd(OUT_C, 75);
  Wait(4000);
  OnRev(OUT_AC, 75);
  Wait(4000);
  Off(OUT_AC);
}
```

`OnFwd(OUT_A, 75);`
　ラベル A に接続されたモーターを 75% の速さで前向きに回転させる

`Wait(4000);`
　4秒間プログラムの実行を止める。その間モーターは回転し続ける。

`OnRev(OUT_AC, 75);`
　ラベル A と C に接続されたモーターを同時に 75% の速さで逆向きに回転させる。

`Off(OUT_AC);`
　ラベル A と C に接続されたモーターを止める。

プログラム作成からロボットでの実行までの流れ

1. プログラムの編集 (ファイル拡張子は “nxc” にすること)
2. コンパイル(compile) & ロード(load) (USB ケーブルをロボットと接続して)

```
nbct a.nxc        # a.nxc から a.rxe を生成し、ロボットにロード
```

注意: “overwrite existing file “◯◯.rxe” (y/n)?”
のメッセージが出るかも知れない。
対応: y を打つ。このメッセージは a.rxe が既にロードされている時に出る。

3. ロードされた事の確認

```
t2n -ls           # a.rxe が見える → プログラム名は a
```

4. ラン(run)　　# ロボット側で実行 (USB ケーブル外して)
My Files → Software Files → プログラム名 a を選ぶ

注: 編集するファイルの名前が XYZ.nxc であれば、プログラム名 XYZ を選ぶ。

トラブル

usb_strerror = No error
Error:
usbnxt: upload can'y initiate upload (reply ……)

アップロードできない。原因として考えられるのは、
(a) LEGO側のプログラム格納に許される容量をオーバした。
(b) アップロードするファイルと同名のプログラムが存在し、実行中である。

問題1

モータの回転速度を 30% にし、待ち時間を2秒間にしてみよう。

問題2

図1のプログラムで

```
OnFwd(OUT_A, 75);
OnFwd(OUT_C, 75);
```

を1つの命令に纏めてみよう。

問題3

このプログラムは車を1往復させるが、2往復させるプログラムを作ってみよう。
コメント: 戻り方がバックなのは気に入らないが、この問題の解決は後回しにしましょう。

コメント

(1) NXC の書き方はプログラミング言語Cと似ているが完全に同じではない。(NXC = “Not eXactly C”)
なおプログラミング言語Cは現代のコンピュータの主力言語である。コンピュータのあらゆる種類のプログラムはCの上に成立していると言っても過言ではない。

(2) nbct コマンドは有澤がこの講義のために作ったものである。このファイルは

```
/usr/local/bin
```

に置かれている。nbct の内容は

```
cat /usr/local/bin/nbct
```

で見える。(Unix は自由にコマンドが作れるようになっている)
処理の本質的な部分を例示すると

```
nbct a.nxc
```

の場合には

```
nbc -O=a.rxe a.nxc
t2n -put a.rxe
rm a.rxe
```

が実行されるように作られている。

Lesson 7 (NXC Program 2)

例題

```
#define MOVE_TIME    500
#define TURN_TIME    500

task main()
{
    int n;
    n = 0;
    while(n < 4){
        OnFwd(OUT_AC, 75);
        Wait(MOVE_TIME);
        OnFwd(OUT_A, 50);
        OnRev(OUT_C, 50);
        Wait(TURN_TIME);
        n++;
    }
    Off(OUT_AC);
}
```

図1. b.nxc

問題1

Lesson 6 に習って図1の b.nxc を実行しなさい。
さらに数字を調整して、90度のターンを4回繰り返して、出発点に戻ってくるようにしなさい。
障害物の周りを1周させてみる。
(調整はトライ&エラーしかないと思う)

いろいろな条件

```
==  equal to (等しい)
<   smaller than (小さい)
<=  smaller than or equal to (小さいか等しい)
>   larger than (大きい)
>=  larger than or equal to (大きいか等しい)
!=  not equal to (等しくない)
```

変数

データの入れ物。名前を付けて使う。データの種類を指定する。int は整数(integer)を表す。

名前規則

変数には名前を付ける。また define は名前を他のデータに置き換える。名前は英字または数字または下線記号('_')から構成され、数字で始まってはならない。予約語(既に意味付けられている語)(task, main, define, int, while など)は名前の規則に合っているが、変数名や define での名前には使えない。例えば n, n2, xyz などは名前に使えるが、2n は使えない。

繰り返し文

C では、いくつかの繰り返し文が仕えるが while は基本的

Lesson 8 (Light Sensor 1)

例題

```
#define THRESHOLD 60
task main()
{
      SetSensorLight(IN_3);
      OnFwd(OUT_AC, 75);
      while(Sensor(IN_3) < THRESHOLD);
      Off(OUT_AC);
}
```

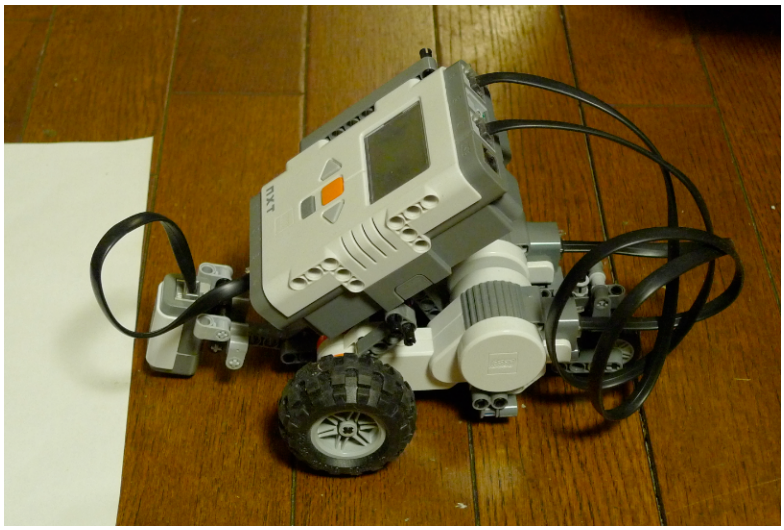
```
# define THRESHOLD 60
    THESHOLD を 60 として定義する

SetSensorLight(IN_3)
   IN_3 にライトセンサーを設定する

while(Sensor(IN_3) < THRESHOLD);
   以下の「説明」を見よ

Off(OUT_AC)
   OUT_AC のモーターを停止する
```

図3. c.nxc

組み立て

光センサーは IN_3
モーターは A と C

光センサーの組み立ては
テキストp.32,33
を見よ。



プログラムの目標

ロボットを白紙に向けて動かす。センサーが白紙を感知したら停止する。

説明

Lesson 7 の例題では

```
while(条件) {  ....  }
```

この場合には条件が満たされている間 { } で囲まれた部分を繰り返す。一般的に言えば { } の中には複数の命令が書かれるが、命令が1個の場合には

```
while(条件) 命令;
```

のように { } を省略できる。命令が存在しない場合には

```
while(条件);
```

である。この場合には条件が破られるまでプログラムはここで停止する。(ロボットが停止する訳ではない。モーターの回転は

```
OnFwd(OUT_AC, 75);
```

でコントロールされており、回転を変化させる命令は実行されてないので、ロボットは前進し続ける。)

今回の例では条件の部分は

```
Sensor(IN_3) < THRESHOLD
```

である。これは「IN_3 のセンサーの値が THRESHOLD (=60) より小さい」の意味である。

注意 ここで使用している言語は本物のC言語ではないので、小数点の付く数は使えない!

問題1

THRESHOLD(しきい値) の値は調整が必要かも知れない。反射光の強さの情報は Lesson 4 (View) の “Reflected light” を使えば分かるので、白い紙と床とで各々の反射光の強さを測定し、その中間を THRESHOLD に採用すれば白い紙と床を区別できるはずである。反射光の強さを実際に測定しなさい。そして、この測定によって最適なしきい値を割り出しなさい。

問題2

ロボットを白い紙の中央部に置いて、ロボットを動かし、センサーが紙の外に出たらロボットを停止させなさい。

問題3

例題のプログラムでは、センサーが白い紙を感知すると直ちに停止する。この時にはロボットの本体は紙の外にいることになる。そこでもう少し難しい問題を考える。直ちに停止しないで、さらに紙の上を走り、センサーが紙の外に出たら停止するようにしなさい。(ヒント: 例題と問題2を組み合わせる。)

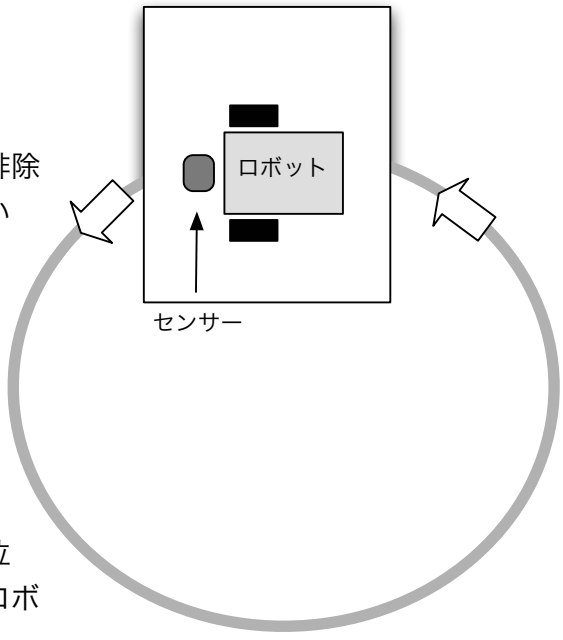
注意:

```
while(Sensor(IN_3) < THRESHOLD);
```

が終了した時に

```
Sensor(IN_3) > THRESHOLD
```

となっている保証はない。つまり等しくなる可能性を排除できない。この現象はセンサーが紙と床の境界を見ている時に発生し得る。



問題4

右図のようにロボットを白い紙の中央部に置いて、ロボットを動かし、センサーが紙の外に出たら(ロボットを停止させなさいで)、ロボットの車の回転数を右と左で変える。するとロボットは円を描いて元の紙の位置に戻って来るはずである。紙の位置に戻って来たらロボットを停止させる。

Lesson 8 (補足)

Lesson 9 の問題3

紙の方向に向けてロボットを放し、紙を抜けたら止める
のがテーマ
とした場合、成功する場合もあるが、失敗する場合もある。何故か?

ヒント

```
Sensor(IN_3) < THRESHOLD
```

の条件が破れた時には

```
Sensor(IN_3) >= THRESHOLD
```

となっている

問題1

失敗する場合は、ロボットのセンサーが、床から紙に移る時にどのような条件が成り立つ場合か?
またロボットが確実に床から紙に移行できるようにするためには、上のプログラムをどのように修正したらよいか?

Lesson 9 (Light Sensor 2)

例題

```
#define THRESHOLD 60
task main()
{
    SetSensorLight(IN_3);
    OnFwd(OUT_AC, 75);
    while(true){
        if(Sensor(IN_3) > THRESHOLD){
            OnRev(OUT_AC, 75);
            Wait(1000);
            OnFwd(OUT_AC, 75);
        }
    }
}
```

光センサーは IN_3 へ
モーターは A と C

光センサーの組み立ては
テキストp.32,33
を見よ。

図1. d1.nxc （出典: “NXC tutorial”)

動作

光センサーが白い紙を捉えるとロボットは 1秒間 バックし、また前進する。また白い紙を捉えて....
但し THRESHOLD の値を適切に設定しなくてはならないだろう。
(Lesson 8 課題1 を見よ)

解説

true (真)

```
while(true){ .... }
```

繰り返しの条件が常に true (真) なので、無制限に { } の中を繰り返す。

```
if(条件){ .... }
```

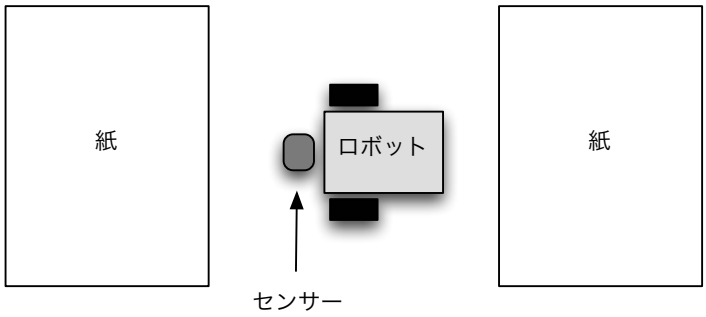
```
if(Sensor(IN_3) > THRESHOLD){ .... }
```

条件が成立する時に { } の中を実行する。ここでは条件として

```
Sensor(IN_3) > THRESHOLD
```

が指定されている。

問題1 2枚の紙を床の上に40cm程離して置き、その間にロボットを置く。ロボットの動作は、2枚の紙の間の往復運動の繰り返しとする。



次に、この動作を可能にするプログラムの例を示す。
??? の部分は適当に埋める事

```
#define THRESHOLD ???
task main()
{
      SetSensorLight(IN_3);
      while(true){
            OnFwd(OUT_AC, 75);
            Wait(???);
            while(Sensor(IN_3) < THRESHOLD);
            OnRev(OUT_AC, 75);
            Wait(???);
            while(Sensor(IN_3) < THRESHOLD);
      }
}
```

魔の境界問題: このプログラムでは

```
Wait(500);
```

がないとうまく動作しない。(500 に関してはもっと適切な値があるかも知れないが...)
これを外して実行してみよ。そして何故うまく動作しないか考えてみよ。
ヒント: while から抜け出した直後には

```
Sensor(IN_3) >= THRESHOLD
```

となっているはずである。

問題2 問題1において、ロボットが白い紙に出会ったら、向きを変えて戻ってくるようにしなさい。ロボットの向きを変えるには左右の車の回転を逆にすればよいであろう。(正確に U ターンさせるのは難しく、そのためにロボットはしだいに紙から離れて行くと思われる。2週回ぐらいできれば可とする。)

問題1では左の紙と右の紙でのロボットの動き方が異なるが、問題2ではUターンなので、ロボットから見た時には左の紙での動作と右の紙での動作には違いはないはずである。そのように考えると問題1のプログラムよりもシンプルにプログラムできるはずである。

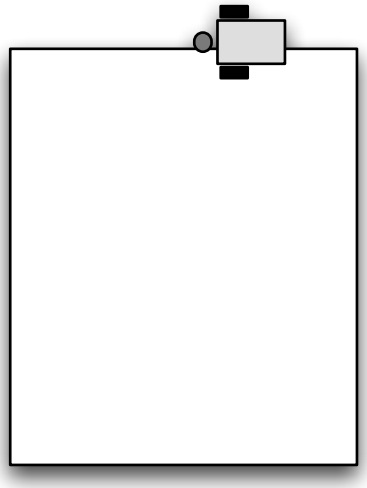
Lesson 10 (Light Sensor 3)

例題

大きな白紙を左回り(反時計回り)に周回させる。白紙は灰色の床の上に置かれている。

プログラム例

```
#define THRESHOLD ??
task main()
{
      SetSensorLight(IN_3);
      while(true){
            OnFwd(OUT_A, 75);
            while(Sensor(IN_3) < THRESHOLD);
            Off(OUT_A);
            OnFwd(OUT_C, 75);
            while(Sensor(IN_3) >= THRESHOLD);
            Off(OUT_C);
      }
}
```



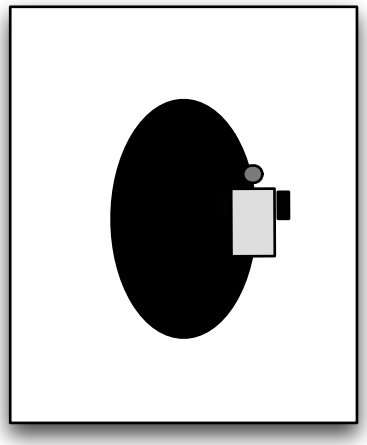
光センサーは IN_3 へ
モーターは A(右モーター) と C(左モーター)

解説

紙の周囲を左回り(反時計回り)で周回させる。そのために、
センサーが一紙の中に入れば、右のモーターを止め、左のモーターを動かす。
センサーが一紙の外に出れば、左のモーターを止め、右のモーターを動かす。
以上を繰り返す。

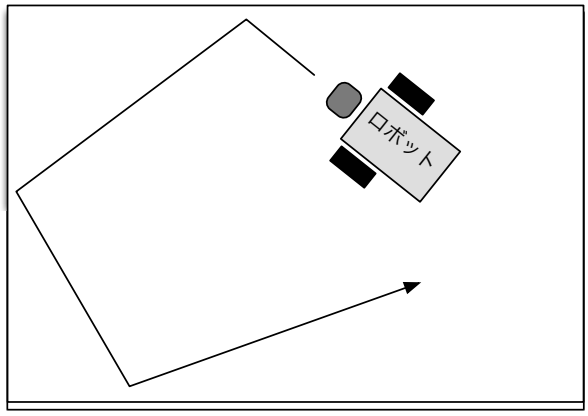
問題1

右図のように白紙の中央に描かれた黒い楕円の周囲をロボットを左回りに周回させる。
注釈: この問題は Lesson 5 (NXT Program) 課題4 で行ったが成功しなかった。(NXT ではロボットの旋回能力に問題があったのである。) しかし今回はきめの細かい制御ができるので成功する可能性がある。



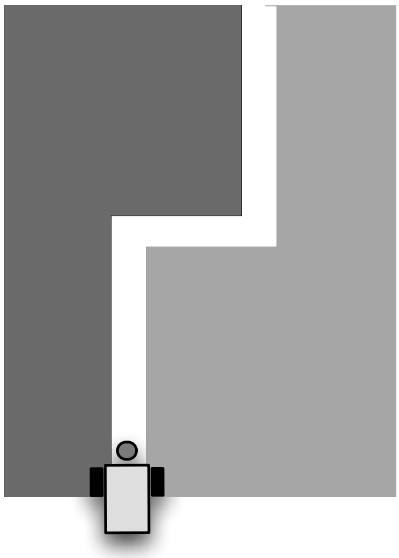
問題2

白い紙の中でロボットをスタートさせる。
紙の中でのロボットの動きは直進とする。
紙からセンサーがはみ出したら、センサーが紙の中に入るまでロボットの左の車輪を止める。右の車輪はその間、前進方向に回転したままとする。
(従ってロボットは左に向きを変える)
センサーが紙の中に入ったら再び直進させる。
以上を繰り返す。(ロボットは紙の中を左回りにぐるぐる回るはずである)



Lesson 11 (Light Sensor 4)

問題1



紙から白い場所に置いて、白い部分からできるだけはみ出さないで紙から脱出させるプログラムを考えてみよう。
ロボットのセンサーは、紙の白色の場所、紙の黒色の場所、灰色の場所の各々を区別しなくてはならない。そこで次のように考えれば良いと思われる。

注意: 実際には灰色ではなく緑色である

動き方の規則 (ルールをシンプルに)
1. 白色の部分では直進
2. 灰色では左向きに旋回
3. 黒色では右向きに旋回

まず、この問題に現れる3つの色(白色、灰色、黒色)を区別するための適切なしきい値を決定する。各色の反射率は Lesson 4 (View) の “Reflected light” を使えば分かる。

白色、灰色、黒色で各々の反射光の強さを測定し、白色と灰色の中間値を `THRESHOLD1`、灰色と黒色の中間値を `THRESHOLD2` とする。すなわち

白色 `>= THRESHOLD1 >` 灰色 `>= THRESHOLD2 >` 黒色

すると

白色では　`Sensor(IN_3) >= THRESHOLD1`
灰色では　`Sensor(IN_3) < THRESHOLD1` かつ `Sensor(IN_3) >= THRESHOLD2`
黒色では　`Sensor(IN_3) < THRESHOLD2`

の条件が成立する。

論理の「かつ」の C 言語での表現は && である。
従って灰色の条件は

```
Sensor(IN_3) < THRESHOLD1 && Sensor(IN_3) >= THRESHOLD2
```

と書ける・

プログラムを考える場合には

`Sensor(IN_3) < THRESHOLD2` なら黒色
そうではなく　`Sensor(IN_3) < THRESHOLD1`　なら灰色
その他は白色

と考えてもよい。「そうではなく」は `else` で表す。

次のプログラムで ?? に数字を書きなさい。

```
#define THRESHOLD1 ??
#define THRESHOLD2 ??
task main()
{
      SetSensorLight(IN_3);
      while(true){
            if(Sensor(IN_3) < THRESHOLD2){
                  /* black, turn right */
                  Off(OUT_A);
                  OnFwd(OUT_C, 75);
                  while(Sensor(IN_3) < THRESHOLD2);
            }
            else if(Sensor(IN_3) < THRESHOLD1){
                  /* gray, turn left */
                  Off(OUT_C);
                  OnFwd(OUT_A, 75);
                  while(Sensor(IN_3) >= THRESHOLD2
                        && Sensor(IN_3) < THRESHOLD1);
            }
            else{
                  /* white, go straight ahead */
                  OnFwd(OUT_AC, 75);
                  while(Sensor(IN_3) >= THRESHOLD1);
            }
      }
}
```

C言語のコメント

`/*` から `*/` まではコメントと言い、実行に影響は無い。メモとして使う。

`if(条件){ .... }else{ .... }`

`else` の後の命令は `if` の条件が成立しないときに実行される。

多分、ロボットは、黒と白との境界を走ろうとするだろう。観察してみる。
(ゆっくりと動かした方が観察しやすい)

また、そのように考えられる理由は?
(これは難しい。ヒント: センサーの解像度が低いので境界がボケて見える)

Lesson 12 (超音波センサー)

例題

超音波センサーを IN_4 に接続する。
目標: ロボットを壁に向けて動かし、壁の手前で(衝突する前に)停止させる。

```
#define NEAR 15 //cm
task main()
{
    SetSensorLowspeed(IN_4);
    OnFwd(OUT_AC,50);
    while(SensorUS(IN_4) > NEAR);
    Off(OUT_AC);
}
```

SetSensorLowspeed(IN_4)

超音波センサーとの通信をセットする。センサーポートに IN_4 を指定。

SensorUS(IN_4)

この値は、超音波センサーから送られてくる、距離情報である。(単位は cm)

注意1: 15cm と書いたが、いい加減である。Lesson 3 (View) の Ultrasonic (cm) を再度試し、プログラムの NEAR の値を決定しなさい。(View には inch 単位のものもあるので注意)

注意2: Ultrasonic センサーは、結構敏感である。ロボットを動かす時の手の位置に注意。

実際に試してみる。15cm で止まるか? (実際に止まった距離は?)

問題1

ロボットを壁に向けて動かし、壁の手前で(衝突する前に)向きを変えて戻ってくる。
戻りは、1秒程動かし、停止させればよい。

問題2

壁の50cm手前に紙を置いて、そこからロボットを壁に向けて動かし、ロボットは壁の手前で(衝突する前に)向きを変えて紙の位置まで戻ってくる。紙を感知したら停止させる。
(その時、停止した時のロボットの位置は、紙の外になる)

問題3

壁の50cm手前に紙を置いて、そこからロボットを壁に向けて動かし、ロボットは壁の手前で(衝突する前に)向きを変えて紙の位置まで戻ってくる。紙の上で停止させる。

Lesson 13 (Rotation Sensor)

例題1 ロボットを前に2秒間進め、後に向きを変えて戻ってくるようにする。

```
void rotate(int angle, int pwr)
{
      int motor_angle;
      motor_angle = 500*angle/100; /* replace 500 by a suitable value */
      ResetRotationCount(OUT_AC);
      Wait(100); /* wait until the counter is really reset */
      if(motor_angle > 0){
            OnFwd(OUT_A,pwr);
            OnRev(OUT_C,pwr);
            while(MotorRotationCount(OUT_A)
            - MotorRotationCount(OUT_C) < motor_angle);
      }
      else{
            OnRev(OUT_A,pwr);
            OnFwd(OUT_C,pwr);
            while(MotorRotationCount(OUT_C)
            - MotorRotationCount(OUT_A) < -motor_angle);
      }
      Off(OUT_AC);
}

task main()
{
      OnFwd(OUT_AC,75);
      Wait(2000);
      rotate(180,30); /* turn 180 degrees with 30% power */
      OnFwd(OUT_AC,75);
      Wait(2000);
      Off(OUT_AC);
}
```

解説

ここでは指定された角度だけロボットの向きを変える命令を作って問題を処理する。つまり

```
      rotate(180,30)          # 30% のパワーで、正方向(反時計回り)に180度回転
```

のように使える新たな命令を作るのである。このような命令があれば他の問題に応用しやすい。ロボットの向きを変えるためには左右の車輪を逆方向に回転させればよい。指定された回転数だけ正確に車輪を回転させるために、ここでは回転センサー(rotation sensor)を使う。これを使うと、ロボットが前進する方向に車輪Aを90度回転させるには

```
      ResetRotationCount(OUT_A);
      OnFwd(OUT_A,30);
      while(MotorRotationCount(OUT_A) < 90);
```

となる。実験してみると、左右の車輪の回転角を同時に正確にコントロールする事は難しいことがわかる。つまり

```
      ResetRotationCount(OUT_AC);
      OnFwd(OUT_A,30);
```

```
OnRev(OUT_C,30);
while(MotorRotationCount(OUT_A) < 90);
```

としても実際には車輪Aが90度回転したときに車輪Bも90度(逆に)回転している保証はない。
そこで、ロボットの旋回角を決めるのは左右の車輪の回転角の差である事を利用して旋回角の精度を上げることにする。(車輪が逆方向に回転すると MotorRotationCount の値は負になる)

```
ResetRotationCount(OUT_AC);
OnFwd(OUT_A,30);
OnRev(OUT_C,30);
while(MotorRotationCount(OUT_A) - MotorRotationCount(OUT_C) < 180);
```

仮に車輪Aの回転角が 90 の時に車輪Bの回転角が -90 にならば、この条件は先の条件と一致することに注意しよう。(一致するように作ったのである)

ロボットを逆方向に旋回させる場合には A と C の立場を入れ替えることになる。

例題のプログラムでは ResetRotationCount の直後に Wait(100) を入れている。実験によるとカウンターをリセットしても直ちにはカウンターが 0 にはなってくれないからである。

車輪の回転角とロボットの旋回角は比例する。比例定数は実験的に定める方がやさしい。

さて例題のプログラムでは

```
rotate(180,30)
```

が実行される。すると

```
void rotate(int angle, int pwr)
```

の angle は 180 に、pwr は 30 に設定される。なお int は integer(整数)の意味である。

注 NXC Guide(p.57)によれば、ロボットを旋回させるために車輪を逆回転させる命令は LEGO NXC に初めから備わっています。

```
RotateMotorEx(OUT_AC,30,180,-100,true,true)
```

とすれば左右の車輪が180度だけ逆向きに回転するはずです。その結果ロボットは旋回するはず(180度旋回する保証はない)ですが、正の向き(反時計回り)に旋回するか、負の向きに旋回するかはマニュアルを見てもよく分かりません。実験するしかないでしょう。

問題1

例題のプログラムの

```
motor_angle = 500*angle/100;
```

の 500 を調整して、

```
rotate(180,30)
```

で 180 度旋回するようにしなさい。

問題2

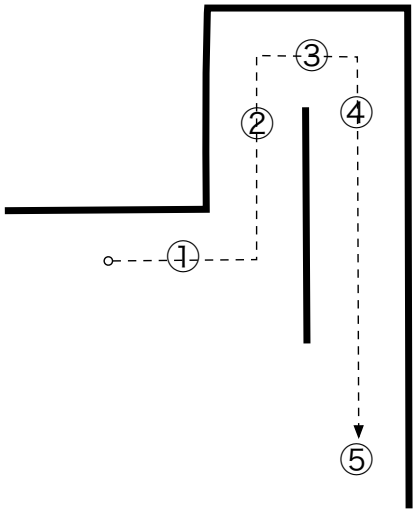
上の問題1の調整で rotate(180,75) でもちゃんと180度旋回するか吟味せよ。

問題3

1m程前に進み、反時計回りに180度旋回し、50cm程進み、時計回りに90度旋回して止まるプログラムを書きなさい。

Lesson 14 (回廊脱出作戦)

今日は最後の課題です。



壁に触れないで、右図のルートを通過させてください。
達成レベルに応じて、2,3,4,5 の点数を付けます。
5 は完全クリア
レベルは丸で囲った数字で示されています。

ヒント:
(1) 真っ直ぐに進むように調整する。
左右の車のパワーを調整する必要があるかも...
(2) 回転角の調整をしっかり行う
(3) 最後の難関(真っ直ぐルート)をどのようにクリアするか?

以上の(1),(2)が出来ていれば、レベル4までクリア可能だと思います。
レベル5は難しいかもしれません。あるいは調整をしっかりやれば可能かもしれません。調整だけでクリアできればラッキーです。

調整をしっかりしても、最後のコースに辿り着くまでに、ロボットの進行方向が壁に平行になっていない可能性が高いのです。その場合、壁に近づいた時に、ロボットの進行方向を右方向に調整するのか、左方向に調整するのかが大問題です。
1個の超音波センサーだけで判断することになるので、手段は限られます。
壁を感知した時に、ロボットを右(または左)に少し(例えば20度程回転させ、壁との距離が遠くになる方向に進めばよいと思われます。