画像処理講座

この講座では画像処理を行ってもらいます。ヘルスケアロボットによる介助支援では，ロボットの人・物体認識が大切です。画像認識についてプログラミングすることで、ロボットの認識を学べます。

今回は OpenCV という画像処理ライブラリを使います。ライブラリというのはプログラムに引用してあげるだけで、特定の処理を行ってくれる、便利なものです。OpenCV には様々な機能があるので、今回はその一部を体験してもらいます。最終的には画像処理の結果を元にロボットを動かしてもらうので楽しみにしていてください。

OpenCV について書かれたサイトはたくさんあるので、この講座が終わった後もぜひご自分で探求してみてください。

目次

① ファイルの実行

・実行したいプログラムを開く

② 画像の読み込み

・カメラからの画像情報を格納する
・フォルダ内にある画像を読み込む
・読み込んだ画像を表示する

③ 色の画像処理

・HSV の H・S・V の範囲を指定して表示する
・表示したい色の範囲を見つけて表示する

④ 画像を操作

・読み込んだ画像の大きさ・表示位置の指定をする

⑤ ロボットを動かす

・完成したプログラムでロボットアームを動かす

① ファイルの実行

プログラムを動かすには機械語に翻訳しないといけません。ターミナル（ctrl + Alt +t で起動）で実行ファイルのあるディレクトリに行きそこで make と打ち込んでください。

```
cd ~/opt/tutorial/matsuda/hebiArm/hebiImage
make
```

カメラを使うために次の 3 つのプログラムを別ウィンドウで事前に起動してください。「yarpserver」はノード間の通信ができます。「astraGraber」はカメラからのデータを PC で受け取ります。「rangeFilter」はカメラデータをこの講座で作るプログラムのノードに送ります。

```
yarpserver
astraGraber
rangeFilter          //astra Stereo を使う場合「rangeFilter --typestereo」を入力
```

これらは常時起動しておいて大丈夫です。

次のコマンドで今回使うプログラムを起動します。

```
./hebiImage
```

② 画像の読み込み

画像処理をするなら元となる画像があったほうがいいと思います（無くても画像処理はできます)。この章では画像をプログラムに読み込ませる方法を学びます。

OpenCV では画像情報を Mat 型に格納します。

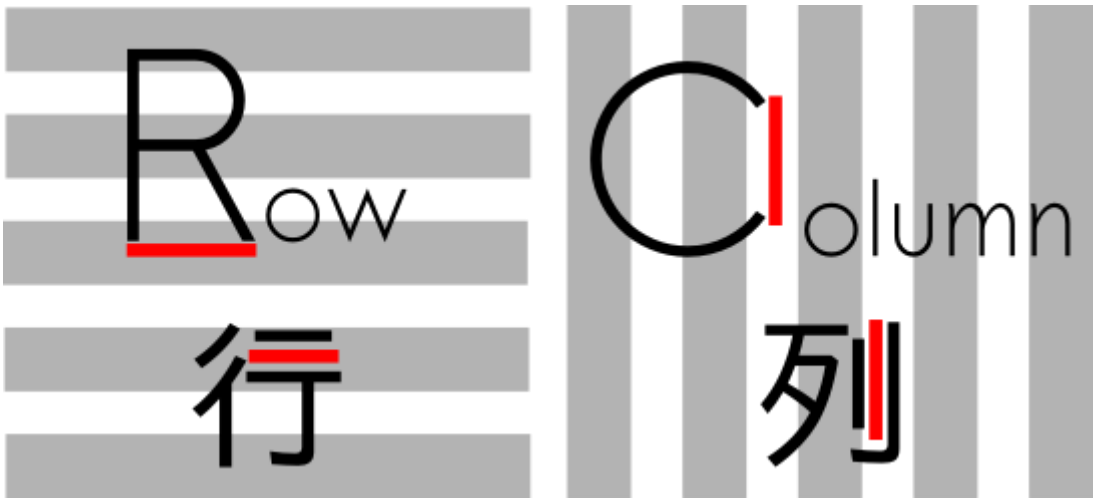
```
Mat　変数；
```

デスクトップにある robot をクリックします。

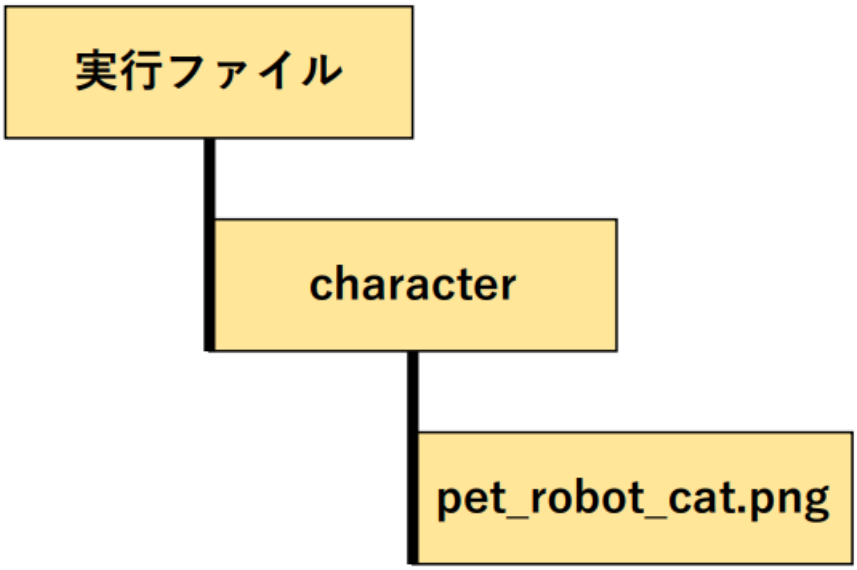
このプログラムでは次のように格納しています。

```
Mat rgb=rp->getRgb();              //カメラからの画像を格納（型の宣言と同時に格納）
Mat hsv ,hsv_mask ,hsv_extract;    //後で使う変数を宣言（宣言のみ）
```

```
// playground を行 rgb.rows、列 rgb.cols、Mat タイプ CV_8UC3 で宣言しています。
//rgb.rows は 480、rgb.cols は 640 の数値が入っています。
// CV_8UC3 は 1 画素に R,G,B が入っています。
Mat playground = cv::Mat::zeros(rgb.rows, rgb.cols, CV_8UC3);
```




```
//（””）内は読み込む画像の場所です。
//「.」（ピリオド）は現在の実行ファイルがある場所を指します。
Mat character = imread("./character/pet_robot_cat.png");//画像読み込み
```



読み込んだ画像が本当に格納されているか気になる人はこれを書いてください。
imshow("rgb",rgb);　　//imshow（”ウィンドウ名”,表示したい画像）
Ctrl と S を同時に押すことでプログラムを保存することが出来ます。

画像の向きが嫌だなと感じた人はこれを書いてください
// 0 :上下反転
// 1 :左右反転
// -1:上下左右反転
flip(rgb, rgb, 0);　//(入力,出力,処理の種類)

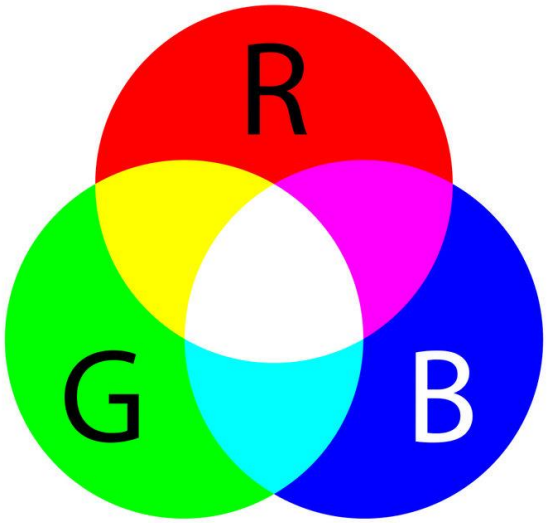
プログラムは基本的に上から順に処理されます。処理がうまくされないなと感じたら、順番を確かめてください。

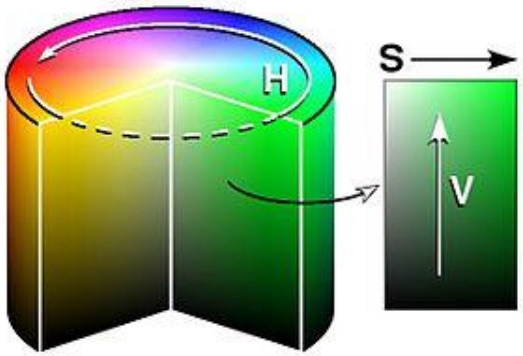
③ 色の画像処理

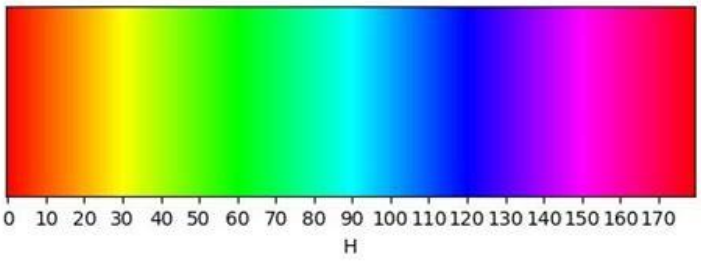
次は色抽出をしてもらいます。画像の中の特定の色を抽出します。

まず抽出しやすいように画像の色を RGB 画像から HSV 画像に変換します。
cvtColor(rgb, hsv, CV_BGR2HSV);

最初 rgb は B（青）G（緑）R（赤）の順で画素の数値が格納されていました。それを H（色相）S（彩度）V（明度）の数値に変換しました。







RGB だと色を指定する際にそれぞれの数値を足した色を想像してかかないといけないので、
色指定がちょっと面倒です。
HSV だと色が指定しやすいので使ってみてください。

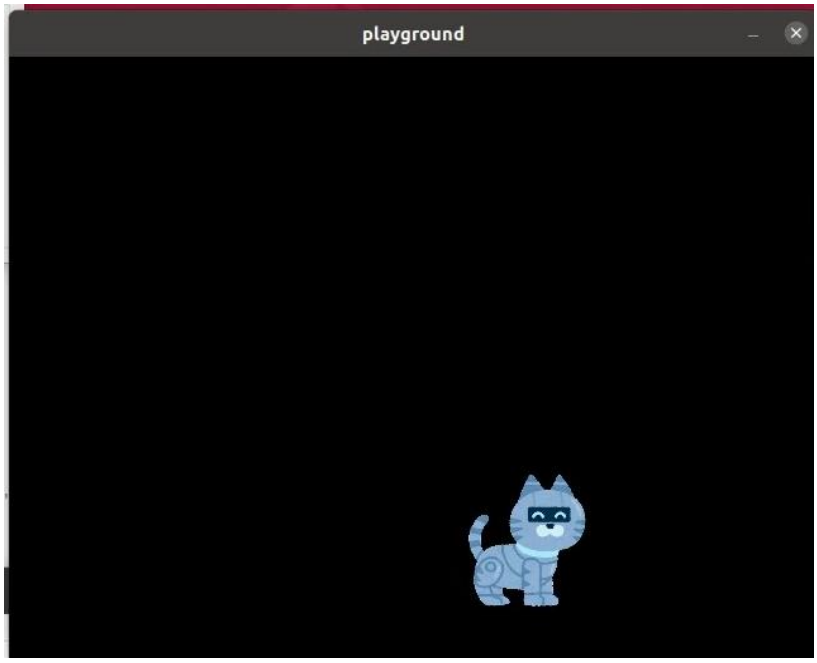
抽出する色を指定してみましょう。Scalar は数値のみを格納してくれます。
//H:0-180, S:0-255, V:0-255
Scalar s_min = Scalar(□, □, □);//数値を入れてみよう
Scalar s_max = Scalar(□, □, □);// 数値を入れてみよう
inRange(hsv, s_min, s_max, hsv_mask);//入力， 最小値， 最大値， 出力

先ほど出力した hsv_mask は閾値に含まれている画素を白で出力したものになります（気になる場合は imshow で見れます）。これをオリジナルの画像と重ね合わせると、抽出した場所だけが見えます。

rgb.copyTo(hsv_extract, hsv_mask);　　//出力，入力

④ 画像を操作

画像処理は慣れてきたでしょうか。この章では画像の中の画像を動かしてもらいます。playground という画像の中にこのように別の画像を入れます。



int main()の上にこのようなコードがあります。int main()内で PinP_tr を書くと、この中の処理をしてくれます。

```
void PinP_tr(const cv::Mat &playground, const cv::Mat &character, const int tx, const int ty){
    //前景画像の変形行列
    cv::Mat mat = (cv::Mat_<double>(2,3)<<0.3, 0.0, tx -(character.cols*0.3*0.5), 0.0, 0.3, ty -(character.rows*0.3*0.5));
    //アフィン変換の実行
    cv::warpAffine(character, playground, mat, playground.size(), CV_INTER_LINEAR, cv::BORDER_TRANSPARENT);
}
```

x と y に好きな数字を入れてみてください。chracter の位置が数値の場所になります。少し戻って character に格納する png ファイルを変えれば別の画像が描画されます。

PinP_tr(playground, character, □, □);　　//ベースの画像，入れたい画像，　x 座標, y 座標
imshow("playground",playground);

張り付けた画像のサイズや座標の貼り付け位置が気になりませんか？それでしたら void PinP_tr の内容を書き換えましょう。ここではアフィン変換をしていて画像を変形できます。2 行 3 列の行列で a が x 方向 d が y 方向の倍率です。tx,ty が画像を置く場所になります。他にも値を変えることで回転させたり平行四辺形にしたりできるので、興味がある人は調べてみてください。

a	b	tx
c	d	ty

```
cv::Mat_<double>(2,3)<<a, b, tx, c, d, ty
```

この章の目的は画像の操作なので貼り付けた画像を動かしてみましょう。せっかく前章で色抽出をしたので、これを使います。hsv_mask の重心を位置を画像の貼り付け位置にします。

```
//重心計算
Moments mu = moments( hsv_mask, false );
Point2f mc = Point2f( mu.m10/mu.m00 , mu.m01/mu.m00 );
```

mc には重心の x,y 座標が入っていますがそのままではわからないので、見えるようにします。hsv_extract の imshow を後ろに作っておいてください（じゃないと見えません）。

```
//circle(画像，中心座標，半径，色(BGR)，線幅，連結)
circle( hsv_extract, mc, 4, Scalar(255, 200, 0), 2, 4);
```

数値を文字で確認するならこれを書いてください。

```
int x = mc.x;
int y = mc.y;
cout << "x="<< x << " y="<< y << endl;
```

x,y 座標が出たので PinP_tr に反映させましょう。すると、色抽出にから画像を動かせます。

⑤ ロボットを動かす

ここまで来たら、このコメントアウトを外してください。これは画面を縦と横をそれぞれ半分に分割して、それぞれの領域ごとに character が移動するプログラムです。ロボットアー

ムは座標情報を元に 4 つのコマンドで動くよう設定されているので、これはそのままロボットアームのシミュレーションになります。

```
if(abs(x) <= rgb.cols && abs(y) <= rgb.rows){
    vector<Point2f> tar(4);
    int w = rgb.cols;
    int h = rgb.rows;
    tar[0].x=w*3/4;
    tar[0].y=h*3/4;
    tar[1].x=w*3/4;
    tar[1].y=h*1/4;
    tar[2].x=w*1/4;
    tar[2].y=h*1/4;
    tar[3].x=w*1/4;
    tar[3].y=h*3/4;
    vector<double> dst(4);
    for(int i=0; i<4; i++){
        dst[i]=distance(tar[i],x,y);
    }
    int p=-1;
    if(!argmin(dst,p)){
        continue;
    }
    PinP_tr(playground, character, tar[p].x, tar[p].y);
}
```

これを書くと領域が見やすくなるのでお勧めです。
// # line(画像, 始点座標(x,y), 終点座標, 色, 線幅, 連結)
line(hsv_extract,Point(rgb.cols/2, 0), Point(rgb.cols/2, rgb.rows), cv::Scalar(255, 0, 0), 1, 4);
line(hsv_extract,Point(0, rgb.rows/2), Point(rgb.cols ,rgb.rows/2), cv::Scalar(255, 0, 0), 1, 4);



できたら動いている様子を見せてください。合格したらロボットアームを動かします。
チェックリスト

- 色抽出ができているか
- 重心座標が正しい値を出せているか
- character が動いているか

画像処理講座

＜講座概要＞

① 最初に画像認識に関する内容を講座します。主に、人の動作を認識する方法についてです。

② 次に、（時間が許す限り）文章生成に関する内容を講座します。ここでは、比較的新しい機械学習の方法を使用して、文章生成を体験してもらいます。

③ 講座終了 30 分前からは、実際に私が開発している音声対話システムの体験も兼ねて実験に協力していただきます。動作認識・文章生成がどのようにして音声対話システムに利用されているかが分かります。

今回、「python」と呼ばれるプログラミング言語を使用します。最近話題のＡＩ・機械学習を知る上で重要な言語であり、かつコード内容が比較的分かり易いため、プログラミング言語を学ぶ上で取り組みやすい言語だと思います。今回教える用途以外にも、機械学習の用途は多くありますので、調べてみるといいかもしれません。

目次

① 使用するフォルダ

② 顔を「点」で見よう

③ 自分の動作を見てみよう

④ 文章を生成してみよう

⑤ 実際にシステムと会話してみよう

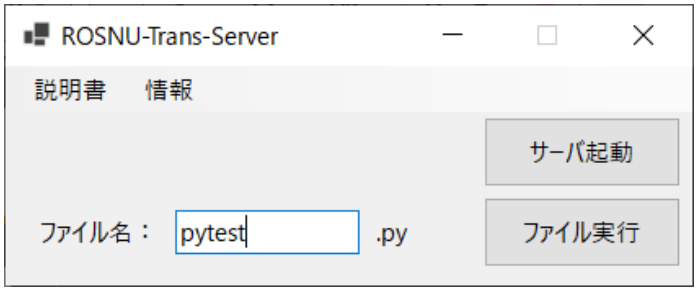
① 使用するフォルダ

今回使用するフォルダは、「programming」というフォルダです。フォルダ内には、主に以下のファイル・フォルダが入っています。

programming
　|
　|- datas：③で使用するフォルダ
　|- python37：プログラミング言語が入っているフォルダ
　|- result：後で使用します
　|
　|- generate.py
　|- mediapipe.py
　|- pytest.py
　|- spot.py
　|- svr.py
　|- text2text.py
　|
　|- python_start.exe：py ファイルを簡単に実行するためのアプリ

※それ以外のファイル・フォルダは、今回使用しません。
試しに pytest.py を起動してみましょう。

＜起動方法＞
① 「python_start.exe」を二回クリックし、開く。
② ファイル名隣の入力欄に「pytest」と入力。



③ 「ファイル実行」ボタンを一回クリックする。

課題①：「pytest.py」を起動する

ねらい：本講座におけるファイルの起動方法を学ぶ

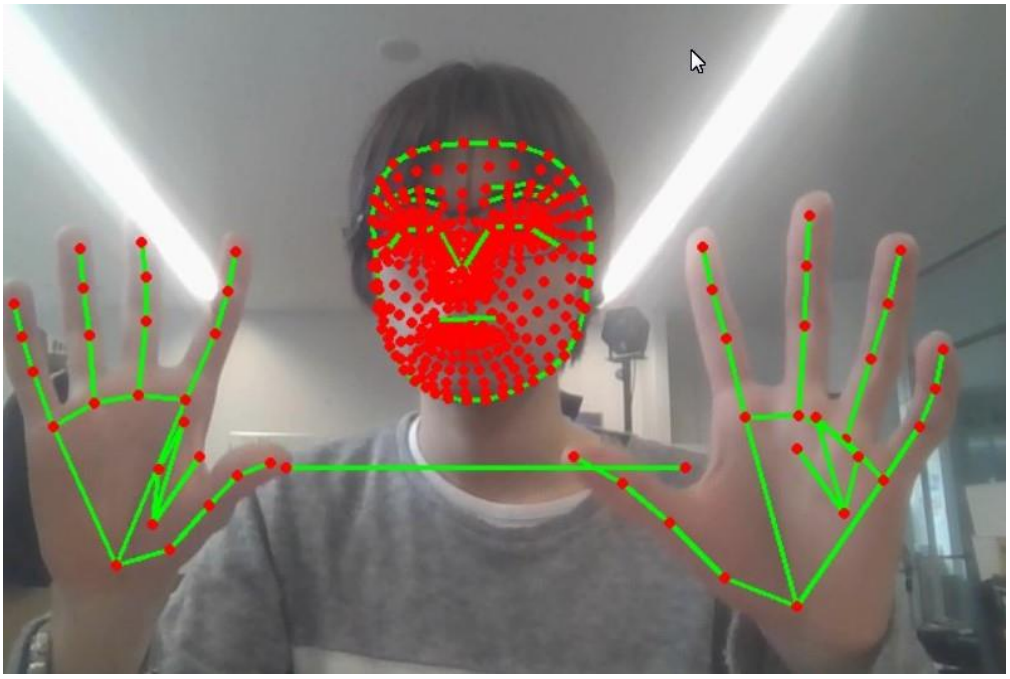
② 顔を「点」で見てみよう

まずは、自分の顔、手、腕などを点や線として見てみましょう。

課題②：「mediapipe.py」を起動する

ねらい：顔を「点」で見るとはどういうことかを体験を通じて学習する。

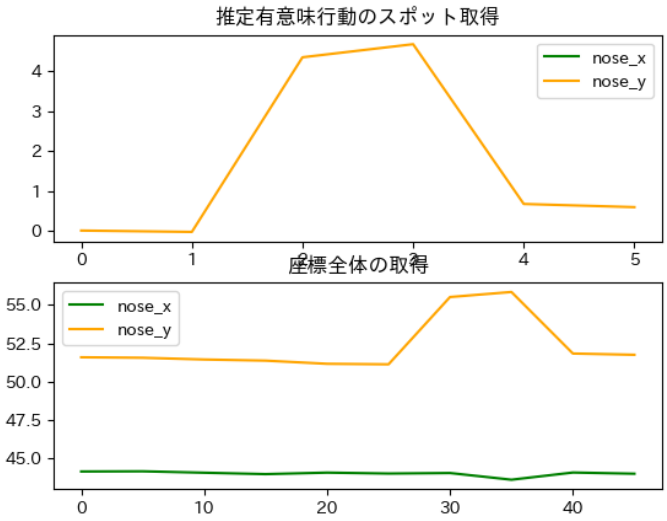
起動すると、顔が点と線で覆われ、腕や手にも点や線が見えると思います。
また、黒い画面には数字が流れているのが分かるかと思います。これは、鼻の頂点付近の座標データを取得し、出力しているためです。このように、点からは座標情報（x, y, z 軸）を得ることができます。



③ 自分の動作を見てみよう

いよいよ、自分の動作を「線」で見てもらいます。

使用する py ファイルは「spot.py」です。ファイルの実行により、以下のようなグラフが作成されます。



下のグラフは一定区間の全体の座標情報、上のグラフはその中から特に動作が大きかった区間の座標情報です。このように、動作は座標情報の変化により、線として現れます。
上のグラフ情報で得られた座標情報は、別のファイルに保存し次のステップで使用します。
では、「spot.py」を起動し、カメラの移る範囲内で右手を適当に動かしてみましょう。

課題③：「spot.py」を起動する

ねらい：座標をどのように記録し、使用するのかを確認する

※今回は座標情報を保存しません。なので、「ログデータの保存」の場面では「n」と入力後、エンターキーを押してください。

次に、自分の動作を記録していきます。今回は「バイバイ」のしぐさを記録します。
先ほどと同じく、起動した後に右手で「バイバイ」のしぐさをしてみましょう。

課題④：「バイバイ」のしぐさを 10 回以上記録する

ねらい：機械学習におけるデータセット収集の過程を学ぶ

「ログデータの保存」の場面では「y」を入力した後にエンターキーを押してください。

保存すると、「base.csv」と「naka.csv」が作成されます。
最後に、記録したデータを使って、線形回帰によって１つの線を書いていきます。

課題⑤：「svr.py」を起動する

ねらい：機械学習の１つの方法である「線形回帰」がどういうものか、体験を通じて学ぶ

起動すると、「base.csv」に記録されている座標情報（点）を機械学習し、新たに黒い線（SVR）が作成されます。
この線には一回で記録できる以上の座標情報が入っているため、認識対象の座標情報の数によらず、その動作が「バイバイ」かどうか判定できます。

さて、今の py ファイルでは「base.csv」しか情報を読み取れません。なので、これを「naka.csv」を認識できるように変更しましょう。

課題⑥：「naka.csv」を使用できるようにした上で「svr.py」を起動する

ねらい：プログラミング言語における文字列の扱い方を知る

＜py ファイルの編集について＞
「svr.py」ファイルを二回クリックすることでファイルの編集を行えます。

＜python のコードについて＞
プログラミングでは、「変数」を定義しながら処理を進めていきます。
例） a＝“あいうえお” → a 変数に文字「あいうえお」を定義

この時、”あいうえお”はダブルクォーテーションで囲む（””）必要があります。
例）a＝“あいうえお” → 文字（str 型）として認識。処理続行。
a＝ あいうえお → 数字（int 型）として認識。エラー。

④ 文章を生成してみよう

ここからは文章生成についてです。音声対話システムにおいて外すことのできない機能となっています。
まずは、「generate.py」を起動し、実際に文章を生成してみましょう。

課題⑦：「generate.py」を起動する

ねらい：機械によって文章が生成できることを、体験を通じて知る

黒い画面に「入力文：」と表示されたら、全角ローマ字入力で文字を入れてみてください。
入れた後にエンターキーを押すことで続きの文章も含めた 10 個の文章が出力されます。

文章が出力されましたが、少し読みにくいと思います。いらない文字が何かを考え、それを自動的に取り除いていきましょう。

課題⑧：「generate.py」ファイルを編集し、再度実行する

ねらい：文章生成処理におけるテキスト加工方法を学ぶ

＜文字の置換＞
python 言語における文字の置換方法は「replace」です。

例）a = “あいうえお”
a = a.replece(“い”, “”)　→「あいうえお」から「い」だけが消える
print(a)　→　「あうえお」

次に、使用する文章データを変えていきます。

課題⑨：読み込むモデル名を変更する

ねらい：モデルを変更することで処理に変化が出るかどうかを検証する

モデル名はフォルダ名と連動しています。現在、モデルは「rinna/japanese-gpt2-xsmall」を使用しているので、これを「datas」フォルダ内の「xsmall」に変更してください。

＜ファイルの指定＞
python の場合、ファイル指定は以下のように行います。
「./xxx/yyy」

課題⑧で使用していた文章データはオリジナル（＝ダウンロードそのまま）の物で、課題⑨は私がインターネット上で収集した文章データを用いて再度機械学習させたデータとなっ

ています。つまり、機械学習に使用するデータが違うと、出力される文章も違ってくるという事です。

次に、使用するフォルダを「datas」内の「small」に変更してみましょう。

課題⑩：読み込むモデル名を変更する

ねらい：モデルを変更することで処理に変化が出るかどうかを検証する

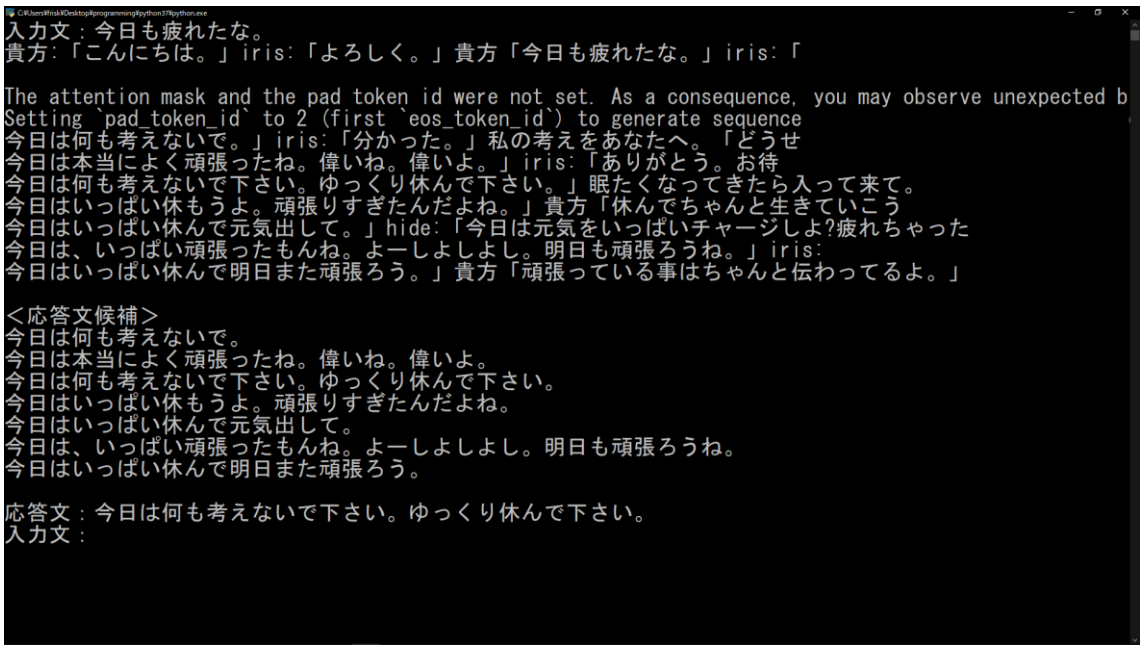
今度は処理時間が遅くなったかと思います。これは、元となるモデルの性能を示す「パラメータ数」が違う為です。パラメータ数は、多ければ多いほど性能が上がりますが、その分処理時間も長くなります。今回の場合、「xsmall」のモデルのパラメータ数が約 3000 万に対し、「small」は 1.1 億であることから、その性能差は分かるかと思います。

では、「text2text.py」を使って実際に音声対話システム同等の応答文を生成してみましょう。

課題⑪：システムの会話文を生成する。

ねらい：音声対話システムに実装している会話生成機能を作る

最終的な出力結果は、こうなります。



上のヒントを元に、課題を取り組んでみてください。

＜課題に取り組むにあたって＞
必ず if 文と for 文を使用します。
if 文：プログラミング上で「分岐」を行う処理

```
if a == “あ”:
    print(“a”)
elif a == “い”:
    print(“i”)
else:
    print(“n”)
```

この場合、a 変数に「あ」が入ってれば「a」、「い」だと「i」、それ以外は「n」と出力されます。

for 文：プログラミング上で「繰り返し」を行う処理

```
for i in [“a”, “i”, “u”, “e”, “o”]:
    print(i)
```

この場合、a~o が順に出力されていきます。

文字は、list という配列の型を使用することで、一文字ごとに分割する事ができます。
例）a = list(“あいうえお”) → [“あ”, ”い”, ”う”, ”え”, ”お”]
これを戻すには、「join」を使用します。
例）””.join([“あ”, ”い”, ”う”, ”え”, ”お”]) → “あいうえお”

正直言って、めちゃくちゃ難しいので、最後まで終わらなくても結構です。
※講座終了 30 分前に切り上げます。

⑤ 実際にシステムと会話してみよう

では、実際に私が開発している音声対話システムを体験してみましょう。
使用後に感想などがあれば教えてください。

本講座は以上となります。お疲れ様でした。

ライントレースロボットを作ろう！

今回製作するのは、「ライントレースロボット」と呼ばれる、地面に引かれた線に沿って進む単純なロボットです。単純ですがロボットの基本的な要素が全て詰まっており、プログラミングの入門用教材としてよく使われるほか、初心者から上級者まで様々な競技会も開催されています。物流倉庫などでも利用されており、単純ゆえに奥の深いロボットです。

ロボットの核となるのは、「マイコン」と呼ばれる小さなコンピュータです。ロボットのプログラミングは、この非力なコンピュータの性能をいかに引き出すかにかかっています。

マイコンには通常のコンピュータと同じように様々な機能が備えられていますが、今回使うのは「シリアル通信」「ピンの読み取り」「ピンからの出力」の三つです。これらの機能を使って、ロボットのプログラムを構築していきます。

目次

① シリアル通信

② ピンの読み取り

③ ピンからの出力

④ プログラムについて

⑤ ロボットを走らせよう



Arduino Nano Every

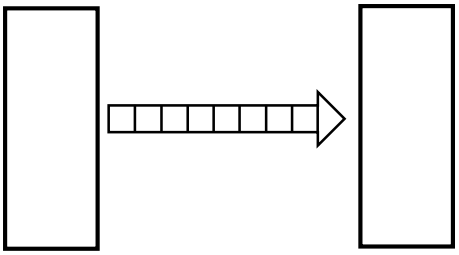
Arduino（アルディーノ）とは

2005 年にイタリアで開発されたワンボードマイコン（一つの基板に最小限の機能を納めたマイコン）。安価で使いやすく、ホビー用や教育用のマイコンとして世界中で利用されている。
様々な種類の Arduino ボードが存在するが、今回使うのは「Arduino Nano Every」という機種。

① シリアル通信

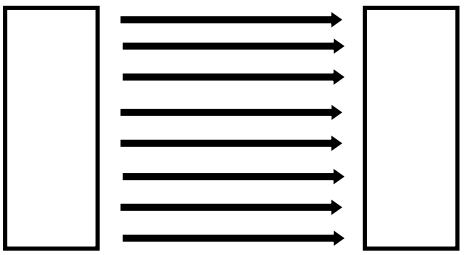
シリアル通信

情報を少ない通信線で 1 ビットずつ順番に送る通信方法。ケーブルが少なくて済むため取り扱いが簡単で、長距離の伝送にも向いている。古くはモールス信号や、最近の例では USB など昔から様々な場所で使われている。対義語はパラレル通信。



パラレル通信

情報を多くの通信線を使って複数ビットずつ同時に送る通信方法。シリアル通信よりも高速に通信できるが取り扱いが難しく、長距離の伝送には向かない。主にコンピュータ内部での短距離高速通信に使われる。対義語はシリアル通信。



シリアル（serial）：連続的な、ひとつづきの、直列
パラレル（parallel）：平行の、並列の

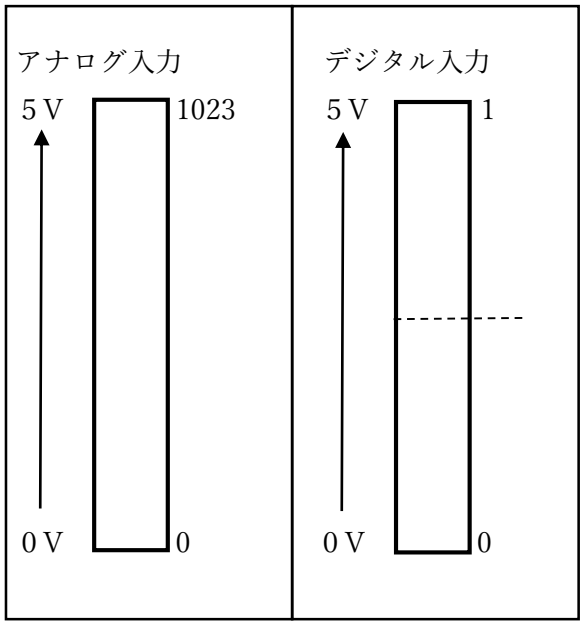
② ピンの読み取り

アナログ入力

マイコンのピンに入力された電圧を読み取る機能。
Arduino の場合は 0～5V の電圧を 1024 等分し、0～1023 の整数値で読み取る。
（読み取り値の例） 0 V→ 0 、　 5 V→1023、　 2.5V→511.5≒512、　 3 V→613.8≒614

デジタル入力

マイコンのピンに電圧が入力されているかどうかを 1 か 0 で読み取る機能。2.5V 以上で 1 になる。
デジタル入力を使う際は「プルアップ」というマイコンの機能を使うのが一般的。プルアップ機能を使うと読み取り値が反転するが、正確に読み取れる。
（読み取り値の例）
プルアップなし　 0 V→ 0 、　 5 V→1、　 2V→0、　 2.5V→1
プルアップあり　 0 V→ 1 、　 5 V→ 0 、　 2V→ 1 、　 2.5V→ 0



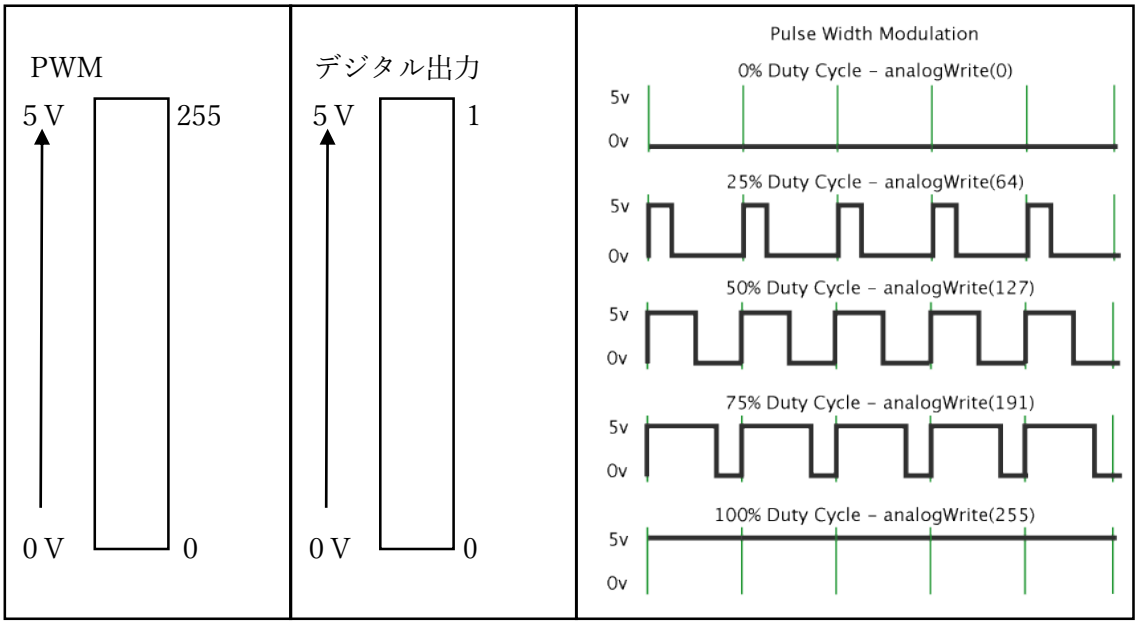
③ ピンからの出力

アナログ出力≒PWM（Pulse Width Modulation、パルス幅変調）

マイコンは 0 か 1 の電圧（0 V か 5 V）しか出力できない。しかし、それでモーターなどを制御しようとすると停止 or 全速力の二択になるのでとても扱いづらい。そこで、0 と 1 を超高速で切り替えることで疑似的に 0 と 1 の間の電圧を出力する。切り替えるタイミングによって 0〜1 の間で様々な電圧を出力できる。
Arduino の場合は、0〜1 を 256 等分した 0〜255 の値を指定することで、様々な電圧を出力する。

デジタル出力

PWM は便利だが、使える数に限りがある(Arduino Nano Every の場合は 3 番，5 番，6 番，9 番，10 番ピンの 5 つのピンでしか使えない)ので、PWMはモーターなど必要なところにだけ使い、他はデジタル出力で制御することになる。Arduino のデジタル出力は 0 か 1 の電圧（0 V か 5 V）しか出力できないが、ほぼすべてのピンで使うことができる。



http://www.musashinodenpa.com/arduino/ref/index.php?f=2&pos=1

④ プログラムについて

マイコンの機能を使いこなすために不可欠なのが、プログラムです。プログラムの言語には様々な構文が存在し、それらの文を組み合わせて実際に動作するプログラムを作ります。
今回使う構文は「if 文」「while 文」の二つです。

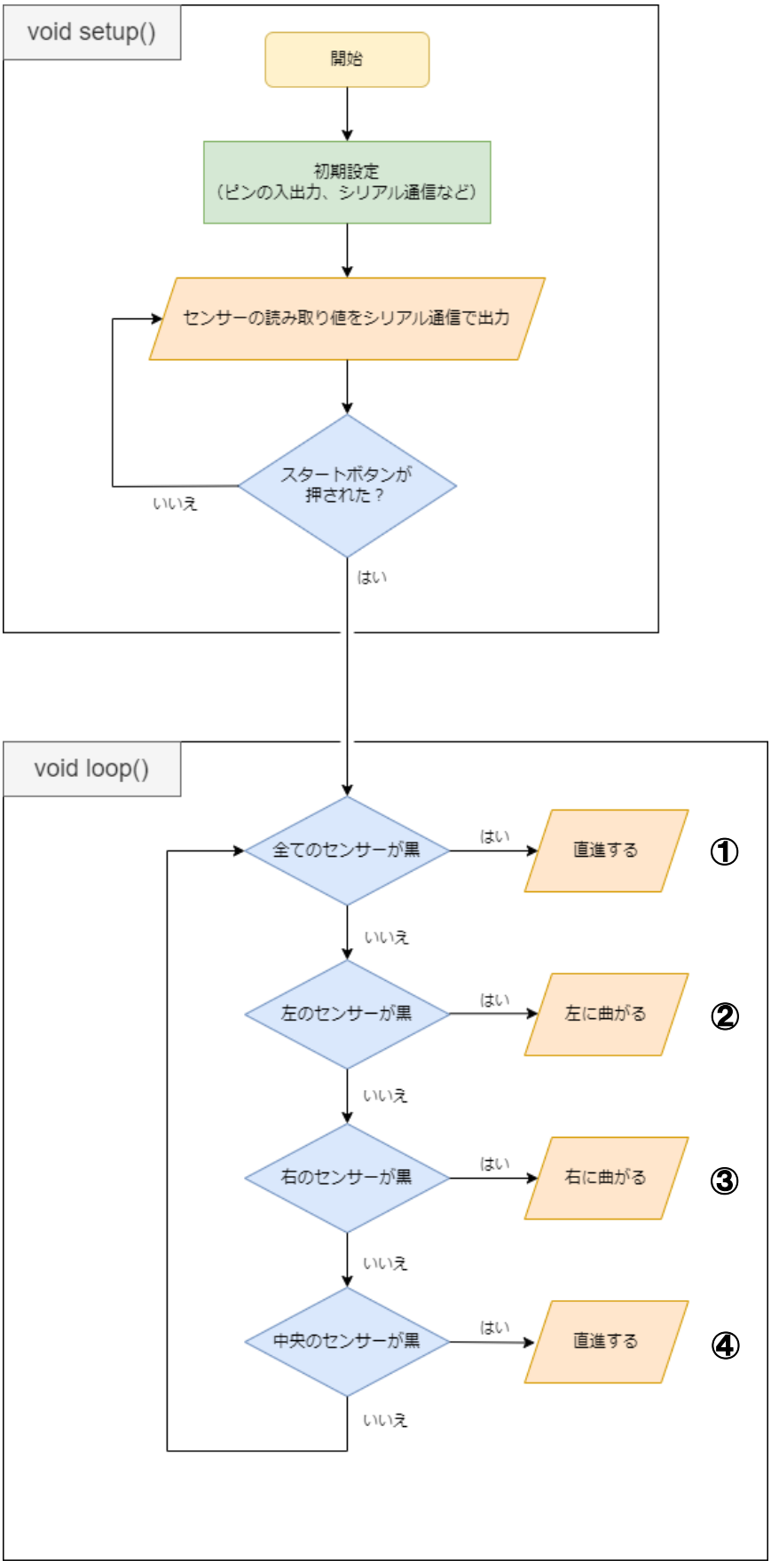
if 文

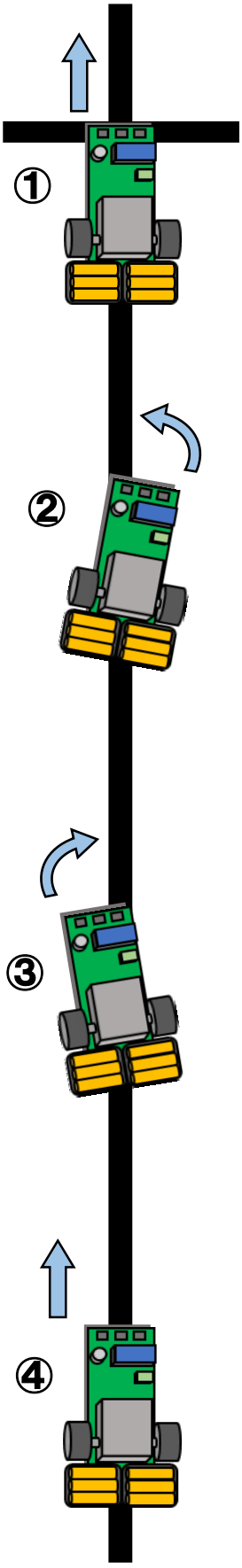
「もし〇〇だったらロボットに××させる」といった動きをさせる時に使う。
if( )←このカッコの中の条件が満たされているとき、そのあとの{ }で囲まれた部分の処理を実行する。
else と組み合わせて、「もし〇〇だったら××させる、そうでなければ△△させる」という使い方をすることも多い。

while 文

「〇〇している間、ロボットに××を繰り返させる」といった動きをさせる時に使う。
while( )←このカッコの中の条件が満たされている間、そのあとの{ }で囲まれた部分の処理をずっと繰り返す。

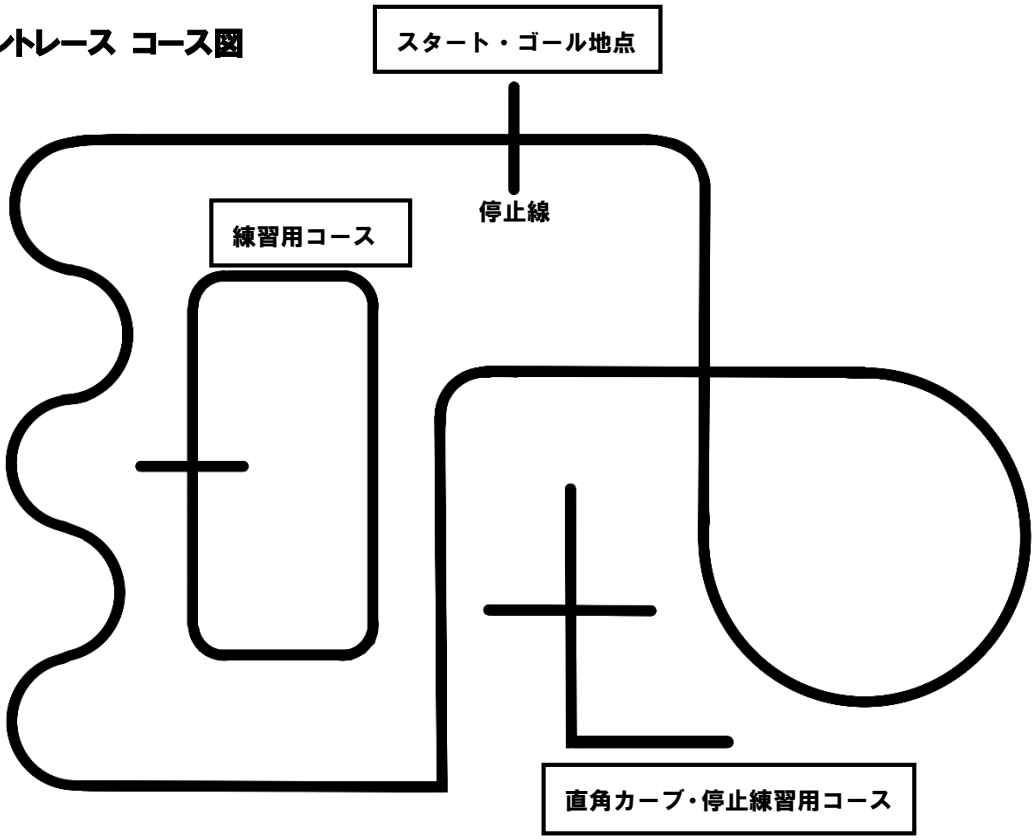
if…「もし～なら」
else…「～でなければ」
while…「～する間」


void setup()
開始
初期設定
（ピンの入出力、シリアル通信など）
センサーの読み取り値をシリアル通信で出力
スタートボタンが
押された？
いいえ
はい
void loop()
全てのセンサーが黒
はい
直進する
①
いいえ
左のセンサーが黒
はい
左に曲がる
②
いいえ
右のセンサーが黒
はい
右に曲がる
③
いいえ
中央のセンサーが黒
はい
直進する
④
いいえ


①
②
③
④

⑤ ロボットを走らせよう

ライントレース コース図



ロボットの走らせ方

① 機体をスタート地点に置き、スイッチを押してスタート
② コースを一周して
③ ゴールの停止線で止まる

停止線で止まるときのヒント

ロボットが停止線の上にいる時には、すべてのセンサーが黒を検知している。
→すべてのセンサーが黒になったら止まればいい？
しかし、停止線以外にもすべてのセンサーが黒を検知する場所があるので、工夫が必要。
全てのセンサーが黒になる場所は何個あって、停止線は何個目なのか……。

ライントレース達成率チェックシート

チェック項目	○/×	失敗した理由
スタートボタンが機能した		
LED が光った		
ラインに沿って直進できた		
ラインに沿ってカーブができた		
交差点を通過できた		
直角カーブを曲がれた		
蛇行ラインをトレースできた		
停止線で停止できた		
クリアタイム	：＿＿＿＿	

プログラミング体験教室
ライントレースロボットを作ろう！

神奈川工科大学
KANAGAWA INSTITUTE OF TECHNOLOGY