講師用

ロボット博士養成講座

ロボティクスプロフェッサーコース

不思議アイテムⅢ－２①

（第1回/第2回テキスト）

必ず、生徒に授業日と自分の名前を記入させるようご指導をお願いいたします。

第1回授業日　2024年　月　日

第2回授業日　2024年　月　日

名前

Human ヒューマンアカデミー ジュニア STE"AMスクール

ロボット博士養成講座 ロボティクスプロフェッサーコース

2024年10月授業分

ロボット博士養成講座

ロボティクスプロフェッサーコース

不思議アイテムⅢ-2①

第1回

液晶ディスプレイに図形を表示させる

講師用

目　　次

○　授業開始にあたって

授業のはじめは、着席させ、大きな声であいさつしてから始めます。

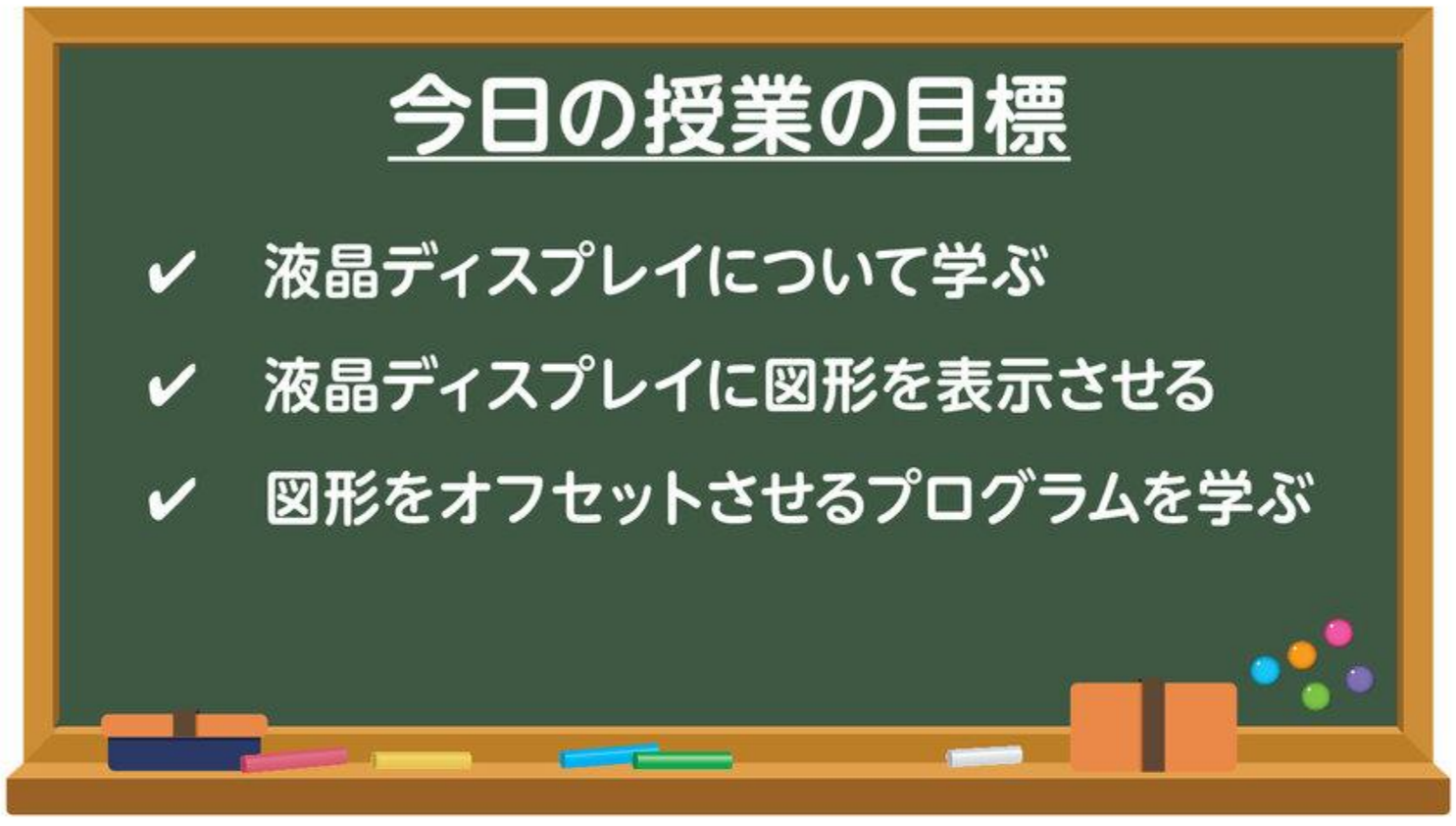
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（授業の目標を明確化することは大変重要なことですので、生徒によく理解させます）

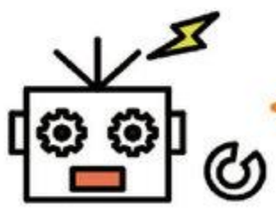
目安時間は授業時間 120 分のうち、休憩 10 分程度取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0.　液晶ディスプレイに図形を表示させる（目安 10 分）

0.0.　「液晶ディスプレイに図形を表示させる」でやること



このタームでは「液晶ディスプレイ」を使います。
「液晶」ということばは聞いたことがある人も多いかもしれませんね。携帯電話やスマートホン、携帯ゲーム機、テレビ、パソコンなど、一般的に「画面」とよばれるものの多くは液晶を利用してつくられています。ロボプロで扱う液晶ディスプレイも、実際に少し前の携帯電話で使われていたのと同じものです。
これまで使用してきたマトリクスLED、7セグメントLEDと比べて扱いが難しそうに見えるかもしれません。実際、マトリクスLEDは8×8、全部で64個のLEDを点（ドット）として使っていましたが、液晶ディスプレイは160×128、全部で20,480個ものドットを持っています。
ただ、裏(うら)を返せば液晶ディスプレイはマトリクスLEDと比べて「ドットが増えただけ」であり、基本的な制御(せいぎょ)のしかたはほとんど同じなのです。むしろ、今までは表現できなかった「色」を取り入れることで、よりたくさんの事ができるようになりますよ！



液晶ディスプレイを使えばゲームもできる？！

0.1. 必要なもの

今回は、以下のパーツを使用します。
液晶ディスプレイシールドの画面には、被膜（フィルム）がついていますので、剥がしてから使ってください。

USB ケーブル 1	マイコンボード 1	姿勢検出シールド 1	液晶ディスプレイシールド 1
			

図0-0　必要なもの

講

今回、各「やってみよう」「チャレンジ課題」の解答例のプログラムを生徒にも確認できるように巻末に掲載しております。
理由としては、問題数が多く講師の方々の解説の時間が足りないかもしれないからです。講師の方は適宜生徒に巻末の解答例を確認させながら授業を進めてください。

0.2. 液晶ディスプレイとは？

1）一般的な液晶ディスプレイ

「液晶ディスプレイ」はスマホやタブレットだけでなく、テレビ等にも広く使われている表示装置です。
そもそも「液晶」とは、液体と固体の中間状態の物質で、ディスプレイ画面の他にもプロジェクターやFRP（繊維強化プラスチック）などにも使われています。そして、電圧をかけることで配列が変わるという性質を持っています。
液晶ディスプレイでは、そのような液晶の性質と、偏光フィルターの組み合わせを利用して、画像を表示しています。液晶と偏光フィルターで光の通り道を開けたり遮ったりするシャッターの役割をしている、とイメージするとわかりやすいかもしれません。

2）今回使用する液晶ディスプレイ

今回使う液晶ディスプレイは、目には見えにくいですが、**図0-1**のように、ドットがたくさん（160x128 ＝ 20480個）並んでいます。さらに、マトリクスLEDとは異なり「フルカラーLED」が使われています。ただし、使い方は、冒頭でも説明した通り、マトリクスLEDと基本的には同じです。

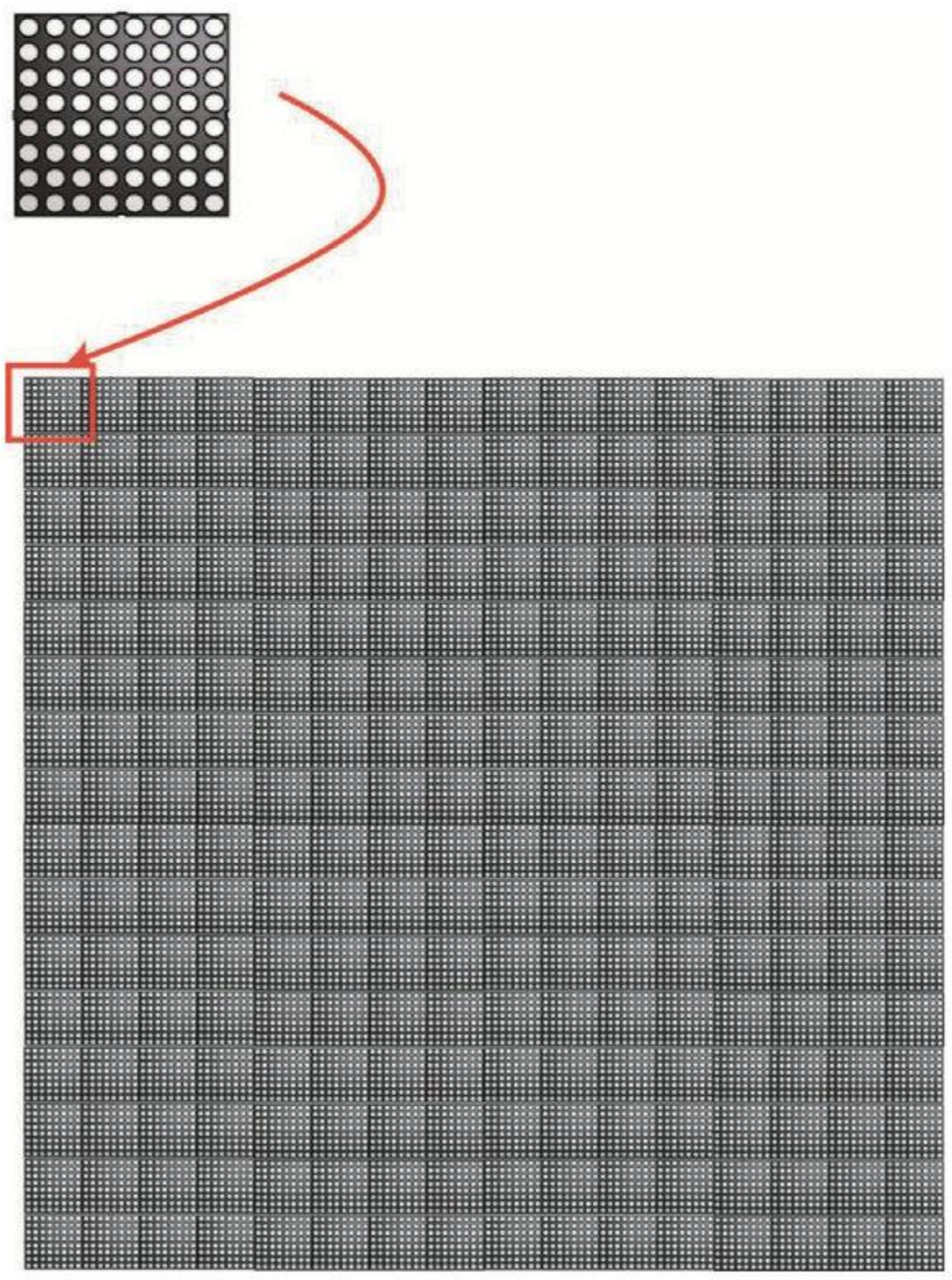


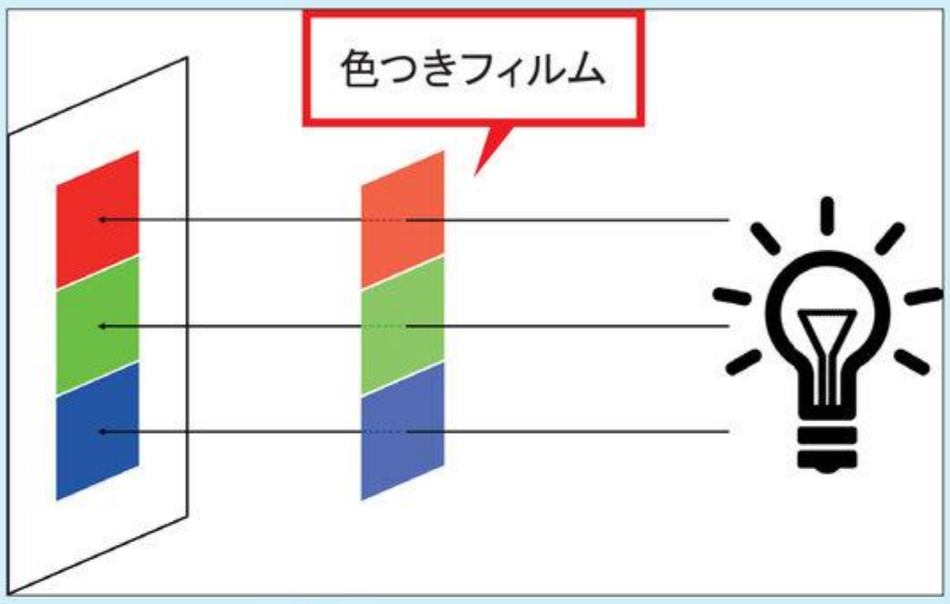
図0-1　マトリクスLEDと液晶ディスプレイシールドの比較イメージ

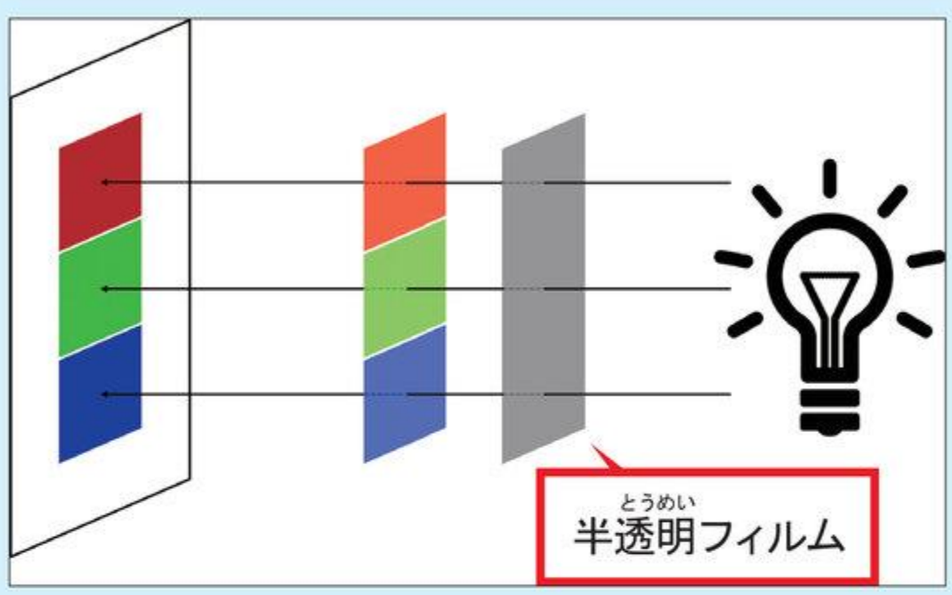
豆知識

液晶ディスプレイは、液体（Liquid）・結晶（Crystal）・ディスプレイ（Display）の頭文字を取って『LCD』と呼ばれることもあります。

液晶ディスプレイの構造

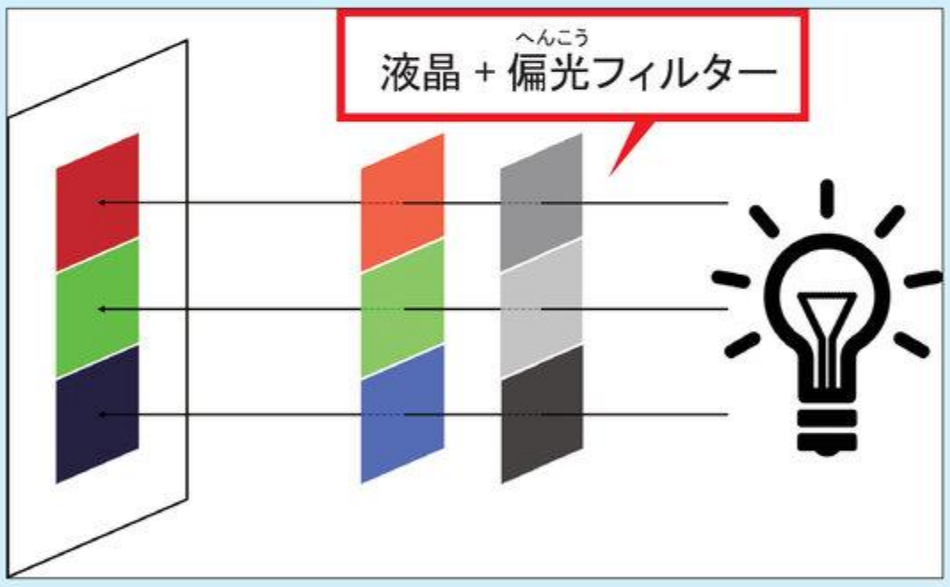
色つきの透明フィルムなどを通した光は、スクリーンなどに映すとフィルムと同じ色になりますね。
ここにさらに、光を少しだけ遮る半透明フィルムを追加すると、スクリーンに映る色も変化します。





フィルムの半透明ぐあいを変化させれば、映る色も変化していきますね。

前のページの説明にもある通り、「液晶」は特殊なフィルター（偏光フィルター）と組み合わせると、かける電圧によって光の遮りぐあいが変化する、つまり、「電気によって半透明ぐあいを操作できる半透明フィルム」のような性質をもちます。
この性質をうまく利用すれば思い通りの位置に、思い通りの色を発生させることができます。とても大まかですが、これが液晶ディスプレイのしくみです。



ところで、いまや「液晶」は「画面」「ディスプレイ」の代名詞です。スマートホンの画面にひびが入ることを「液晶が割れてしまった」と言う人も多いのではないでしょうか（ちなみに、ここで割れているのは内部の液晶ではなく、表側のガラス面なので、厳密には正しくない言いかたですね）。

ただ、世の中のディスプレイがすべて液晶でできているわけではありません。液晶ディスプレイは液晶自体が光るわけではないので別に光源が必要ですが、最近では「有機ELディスプレイ」という、光源を必要としないディスプレイも登場しはじめています。
いつか、皆さんが「液晶が～」と言ったら「『液晶』って言いかた、古いですね～」と若者に笑われる世の中が来るかもしれません。そんな時代には、どれほど薄く、きれいに映るディスプレイが登場しているのでしょうか。今から楽しみですね！

1．液晶ディスプレイを使ってみよう（目安 90 分）

1.0. 組み立てと表示テスト

1）組み立て

では、実際に液晶ディスプレイを使ってみましょう。まずは組み立てです。**図1-0**を参考にマイコンボードと各シールドを組み立てましょう。

下から、マイコンボード・姿勢検出シールド・液晶ディスプレイシールド、という順になるように組み立てます。姿勢検出シールドに液晶ディスプレイシールドを接続するときには、ソケットを間違えないようにしましょう。

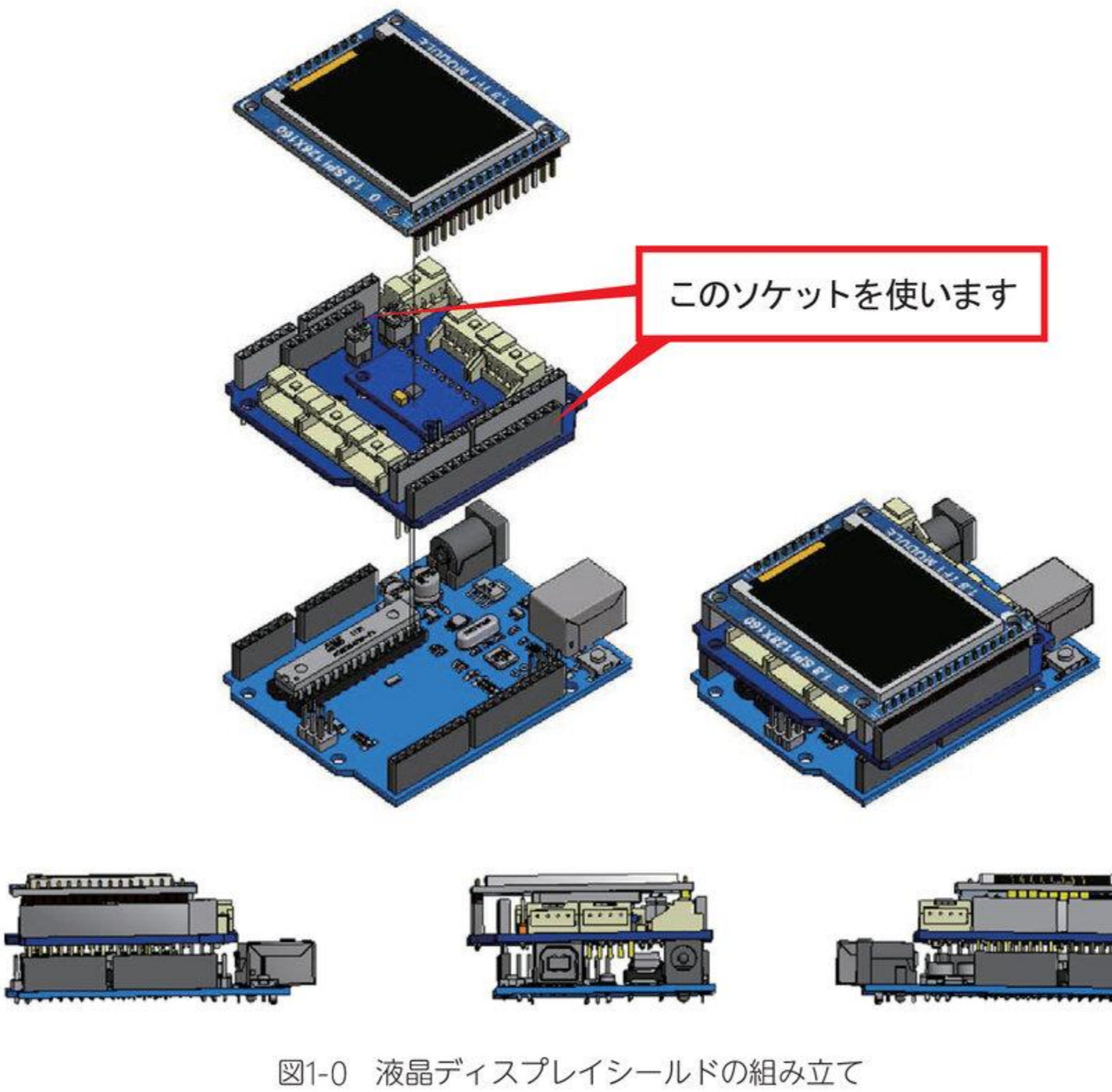


図1-0　液晶ディスプレイシールドの組み立て

2）表示テスト

組み立てが終わったら次のプログラムを実行してみましょう。ディスプレイシールド動作テストプログラムです。ディスプレイに結果が出ます。どんな表示がされるか確かめてみましょう。

プログラムの書き込み

RoboticsProfessorCourse3　>　MagicItemLCD1　>　graphicsDEMO

1.1.　線を描いてみる

1）液晶ディスプレイの座標

ディスプレイに図形やキャラクターを表示させるには「座標」を使いプログラムをします。座標の取り方に関しては注意が必要です。液晶ディスプレイは**図1-1**のような座標をとります。一般的な数学とはちょっと違う座標の取り方をしているので注意しましょう。この液晶ディスプレイの場合は、**x軸は160までy軸は128**までです。

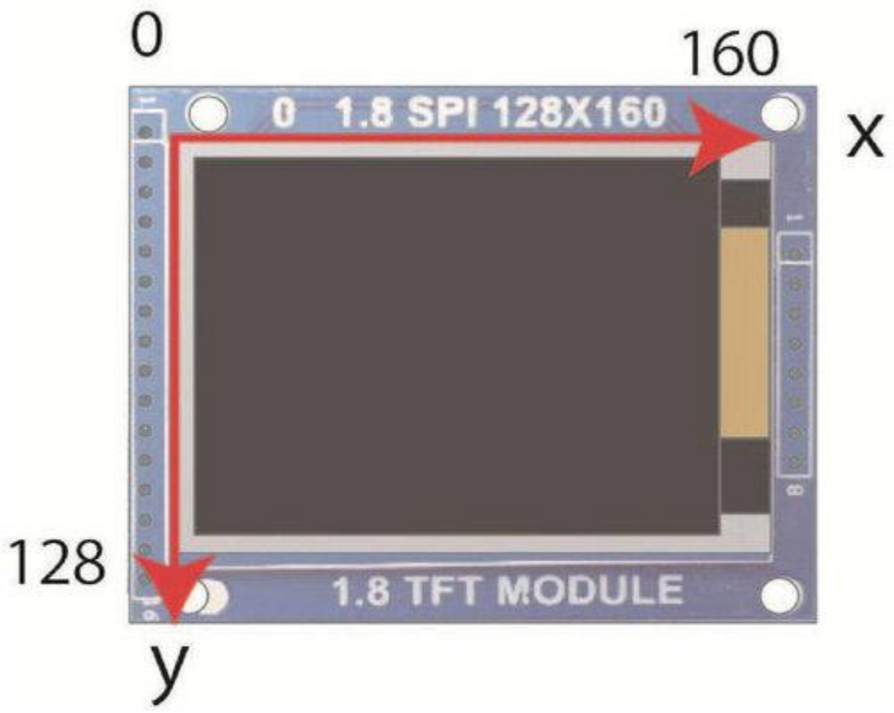


図1-1　液晶ディスプレイ上のx座標とy座標

2）線をかくプログラム

まず、はじめにディスプレイに線を描いてみます。線をかくにはx軸、y軸が交わる点を2つ指定します。そうすると2点をつなぐ線が引かれます。

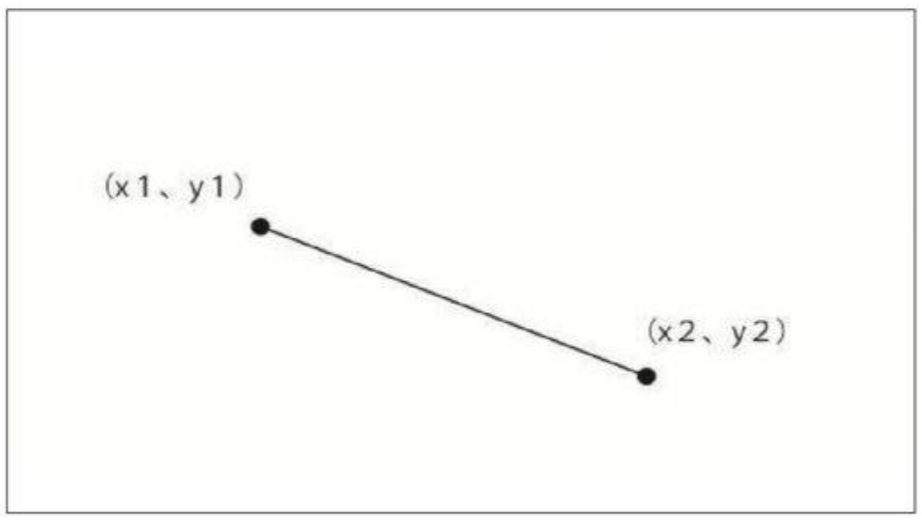


図 1-2　線の引き方

では、次のプログラムで実際に線が描かれるところを確認しましょう。

プログラムの書き込み

RoboticsProfessorCourse3　>　MagicItemLCD1　>　drawLine

プログラムを実行すると、液晶ディスプレイに**図1-3**のように白線が斜(なな)めに1本表示されます。線を表示するためには、①「背景色」、②「線の色」、③「線の形（長さや向き）」の3つをプログラムする必要があります。

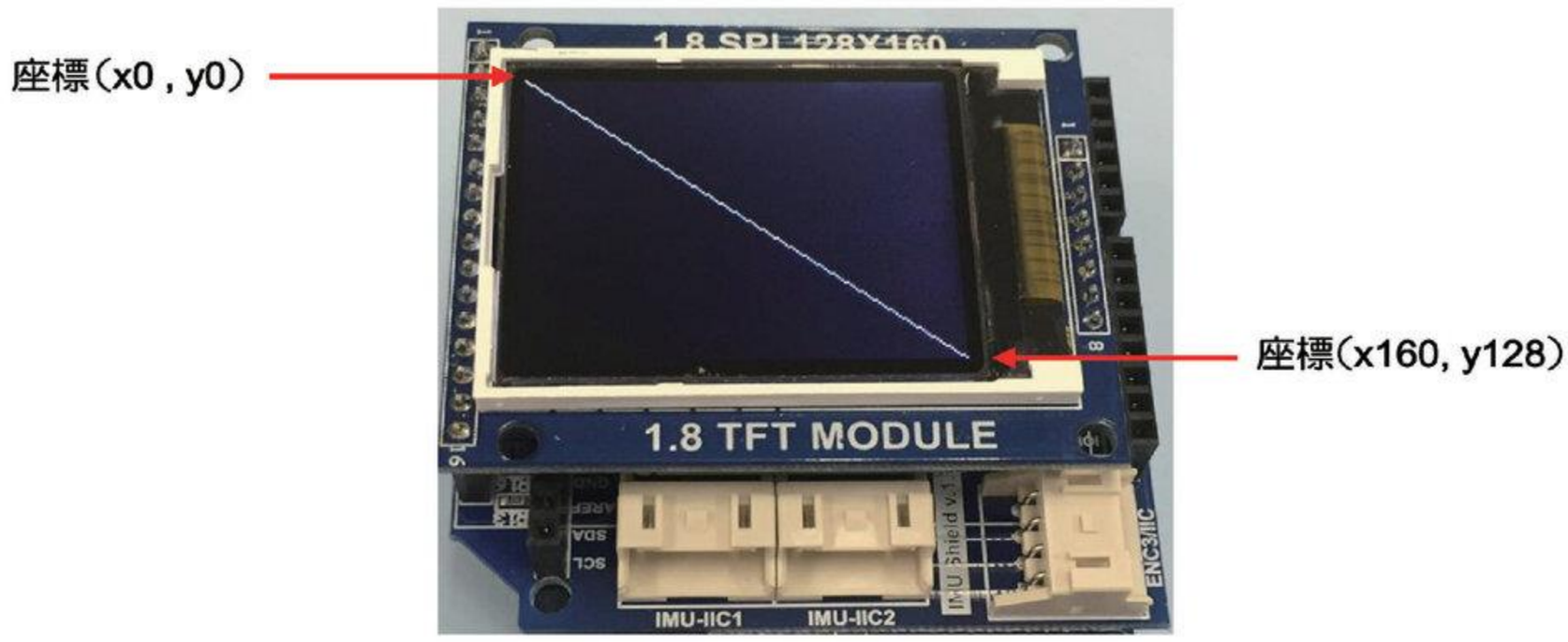


図1-3　drawLine

実際にプログラム「drawLine」を見てみましょう。黄色部分の①～③の関数で表示の内容を指定しています。この関数は、第2 回以降も使われますので、覚えておきましょう。

プログラム「draw Line」より抜粋(ばっすい)

```
void setup(void){
    TFTscreen.begin();
    ①TFTscreen.background(0,0,0);          //背景色は黒
    ②TFTscreen.stroke(255,255,255);        //線の色は白
    ③TFTscreen.line(0,0,160,128);          //(0,0)から(160,128)へ線を描く
}
```

３）線を表示させる関数

①背景色：３原色のR（赤)、G（緑)、B（青）の値を入力します。

命　　令「TFTscreen.background」
実行結果：指定した色の背景を表示する
使い方：TFTscreen.background([B 値], [G 値] , [R 値]);

[B 値]　青の値
[G 値]　緑の値
[R 値]　赤の値
※RGBの値は0 から255 まで指定できます（ただし今回のディスプレイでは16ビットカラー 65,536 色までしか表現はできません)。

②線の色：３原色のR(赤)、G(緑)、B（青）の値を入力します。

命　　令「TFTscreen.stroke」
実行結果：指定した色の線を表示する
使い方：TFTscreen.stroke ([B 値] , [G 値] , [R 値]);

[B 値]　青の値
[G 値]　緑の値
[R 値]　赤の値
※RGBの値は0から255 まで指定できます（ただし、今回のディスプレイでは16ビットカラー 65,536 色までしか表現はできません)。

例：TFTscreen.stroke(255,255,255); //線の色は白

③線の形（長さや向き)：始点と終点の２点の座標を定めます。

命　　令「TFTscreen.line」
実行結果：線を描く
使い方：TFTscreen.line([座標x1],[座標y1],[座標x2], [座標y2]);

例：TFTscreen.line(0,0,160,128);　//(0,0) から(160,128) へ線を描く

講 RGBとは、色の表現方式の一つで、赤（Red)、緑（Green)、青（Blue）の配合比率を変化させて、すべての色を表現する方式です。コンピューターで図形や画像、動画などを扱う際の標準的な色表現の一つで、ディスプレイ装置などで利用されます。

4）色の表示

以前のテーマでも解説したとおり、コンピューターは基本的に「0」と「1」の2つだけを用いた数え方「2進法」を使います。では、コンピューターはどのように画像のデータを作っているのでしょうか。ためしに、下の図のような白黒のグラデーションの画像を考えてみましょう。

図1-4　白黒グラデーション

はじめに説明したとおり、液晶ディスプレイは細かいマスが集まってできています。マトリクス LED を使ったとき、0 と 1 をひたすら並べてかくことで図形をかいたのを覚えていますか。同じ原理で、0 と書かれたマスは黒、1 と書かれたマスは白を表示することにして、16 マスの画面でグラデーション画像をつくってみましょう。

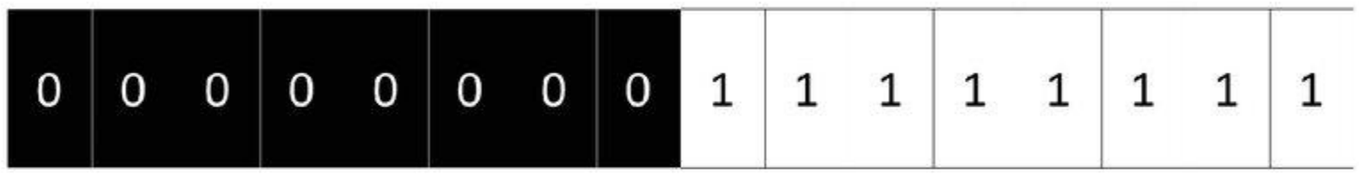


図1-5　1ビットカラー

グラデーションの再現とまでは言えませんが、白黒に塗(ぬ)りわけた画像になりました。もっと元の画像に近づけたければどうすればよいでしょうか。
ためしに、1 マスに 2 けたの値を使うことにしてみましょう。すると 00、01、10、11 の 4 パターンの数がつくれるため、白と黒以外に 2 色の中間色、つまりグレーが使えます。

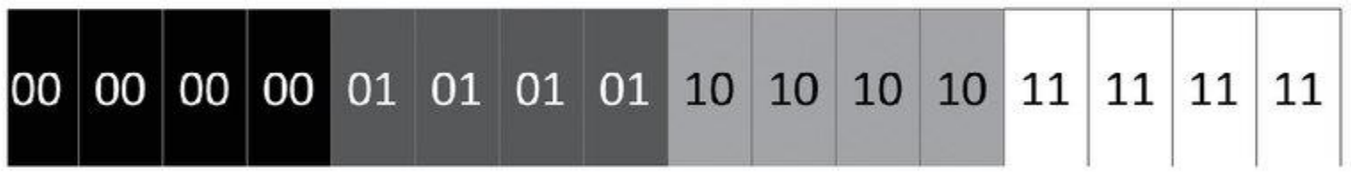


図1-6　2ビットカラー

少しだけ元のグラデーションに近づきました。1 マスに割り当てるけたの数を「ビット」とよびます。1 ビットの画像は白黒の 2 色、2 ビットの画像は 4 色でした。3 ビットでは 8 色、4 ビットでは 16 色ですね。

図1-7　3ビットカラーと4ビットカラー

ビット数が大きくなるほど、より細かな色分けが可能になるのです。そのかわり、1 マスあたりのデータ量が多くなるので、画像そのもののデータ量は大きくなってしまいます。

カラー画像も同じ原理で、赤、緑、青それぞれを何色ずつかに分解して、3色を混ぜて色をつくっているのです。
今回扱ったプログラムでは、それぞれ256色ずつに分解していましたね。2進法では8けたで0～255までを表現できるので、3色あわせて24けた、つまり24ビットの画像という事になります。
16進法だと0～255をちょうど2けたで表しきれるため、16進数を3色分、計6けた並べて色を表現することもあります。「カラーコード」といい、ホームページをつくるときなどに使われています。

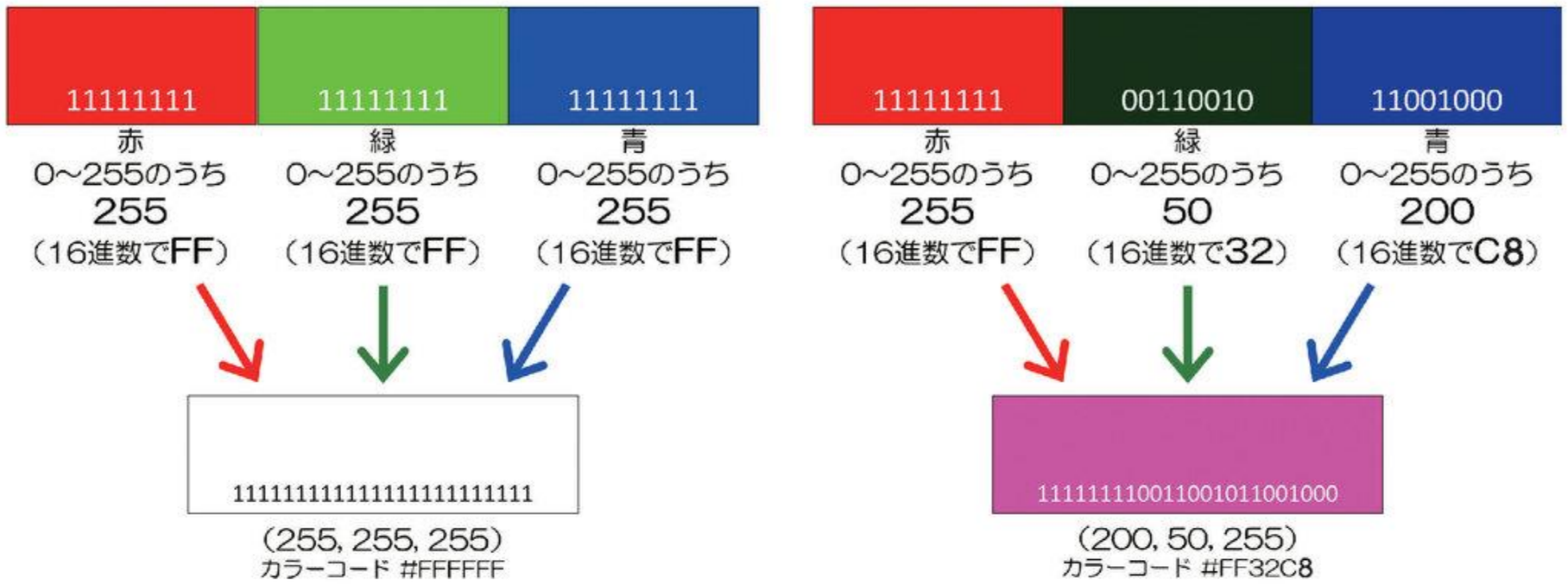


図1-8　24ビットフルカラー

24ビットカラー画像は、赤緑青それぞれ256通りの色が選べるので、全部で256×256×256＝約1,677万色もの色を作れます。ただ、今回は扱っている液晶ディスプレイの性能に限界があり、実際は赤5ビット（32色）、緑6ビット（64色）、青5ビット（32色）の16ビットカラーで画像を表示しています。全部で32×64×32＝65,536色ですね。緑だけ細かく分けられているのは、人間の目は赤や青と比べ、緑色を細かく見分けるのが得意だからだそうです。

一般的によく使われる色の一覧をまとめますが、他にも細かな色がたくさんあるので、少しずつ数値を変更しながら好きな色を探してみてもよいでしょう。

表1-0　RGBカラー

Red	Green	Blue	色
255	255	255	
0	0	0	
255	0	0	
0	0	255	
0	255	0	
255	255	0	
0	255	255	
255	0	255	
110	60	170	
50	50	50	

※RGBは、プログラム上ではBGRの並びになっているので注意しましょう。

やってみよう！

プログラム「drawLine」をアレンジして、線の色をかえてみよう。
`TFTscreen.stroke(255,255,255);`の値を書きかえて、線を赤にしてみよう。
赤ができたら、ほかの色でも描いてみよう。

ヒント
表1-0によると、「赤」はR255、G0、B0だよ。

線ができたら、背景色や線の長さや向きも自由にかえてみましょう。

講　プログラムに慣れるために、時間を決めて変更を繰り返させてみましょう。時間に余裕がある際は、線の座標を変更させてみてください。
なお、次の頁の「ステップアップ」では、座標を扱う問題にチャレンジします。

5）さまざまな線を描いてみる

ステップアップ

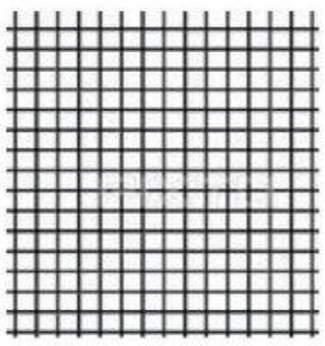
プログラム「drawLine」を変更して、①②の内容を表示させよう。
160×128のディスプレイの枠内に入れるのがポイントだよ。

① line命令とfor文を使って横縞を描いてみよう。縞の間隔の幅を10にしてみよう。

ヒント

どの線も始点・終点のｘ座標は変わらないから、ｙ座標だけ 10 ずつ増加するようにすればいいよね！

② line命令とfor文を使って格子模様を描いてみよう。



講

解答プログラムはそれぞれ以下となります。
RoboticsProfessorCourse3 ＞ MagicItemLCD1 ＞ drawLineAns1
RoboticsProfessorCourse3 ＞ MagicItemLCD1 ＞ drawLineAns2
解答例は巻末に記載します。
なお、for 文の使い方については、豆知識を参照させてください。

豆知識

“for文”と“比較演算子”

★forのくり返し条件式の使い方

for(int y = 0; y <= 128; y += 10) → for(初期値 ; 条件式 ; 増減幅)

TFTscreen.line(0, y, 160, y); → 上で条件を定義したyを代入すると、0～128まで10ピクセルごとにx軸0～160までの線を引く命令になります。

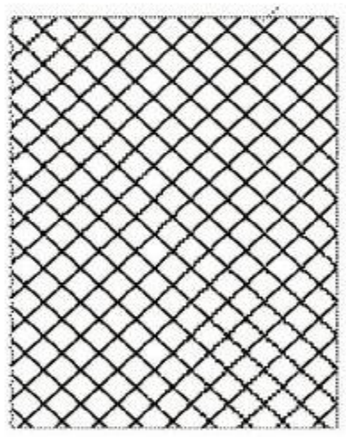
★比較演算子の使い方

==	2つの値が同じである場合に成立したとみなします
!=	2つの値が異なる場合に成立したとみなします
<	前の値より後の値が大きい場合に成立したとみなします
>	前の値より後の値が小さい場合に成立したとみなします
<=	前の値より後の値が大きいか同じである場合に成立したとみなします
>=	前の値より後の値が小さいか同じである場合に成立したとみなします

講 ここでは、比較演算子の使い方を確認してください。以降の課題にもfor文が使われていますので、わからなくなったら表を確認させてください。
また、方眼紙などを使って手で書きながら進めると理解しやすくなります。

チャレンジ課題

line命令とfor文を使って、斜め45度の格子模様を描いてみよう。



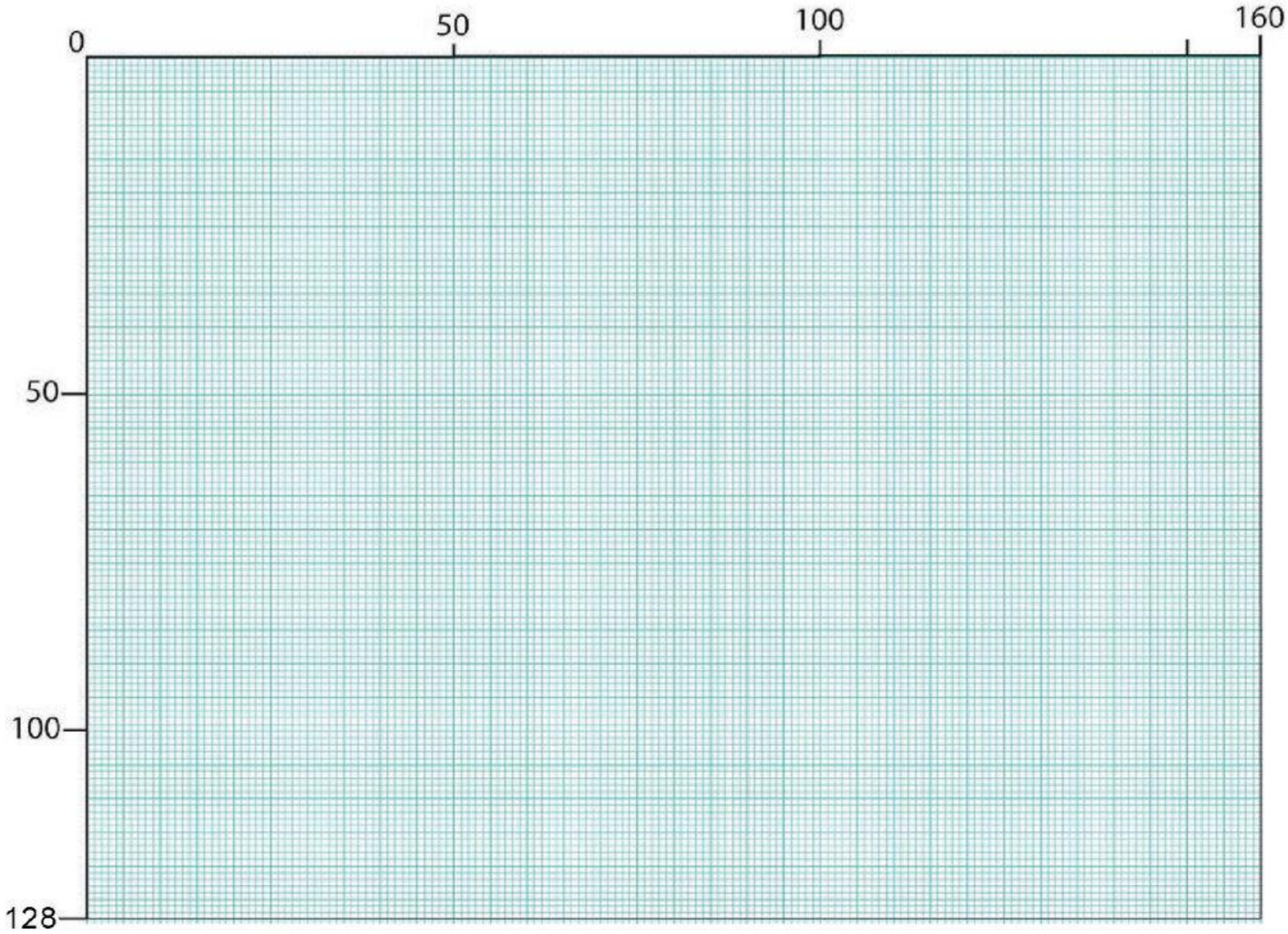
ヒント

液晶ディスプレイの座標は (0, 0) から (160, 128) しかないけど、その範囲に無い座標も指定できるよ！

講 解答プログラムは以下となります。
RoboticsProfessorCourse3 > MagicItemLCD1 > drawLineAns3
解答例は巻末に記載します。

チャレンジ課題

方眼紙に手書きで線を書いて座標をとってみよう。
できあがったら、座標のポイントを表に控えて、プログラムに書き込んでみよう。



ポイント	X	Y
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

1.2. 四角形を描いてみる

続いて、四角形を描いてみましょう。線をかく命令を組み合わせて四角形を描いてもよいのですが、四角形をかくための専用の命令もあります。四角形は**図1-9**のように四角形の頂点の位置を決めて、縦横(たて)の幅を設定します。まずは、プログラムを実行してみましょう。

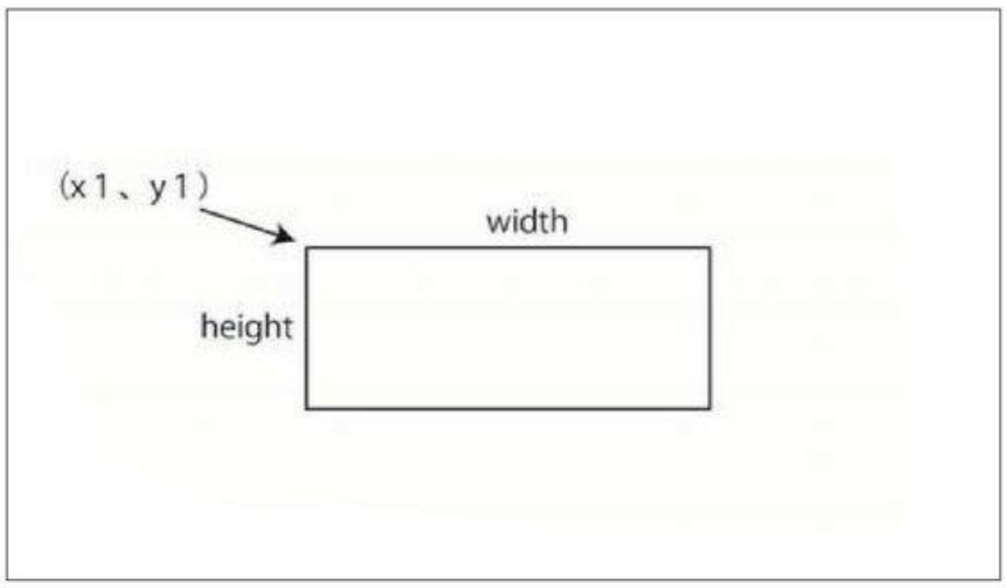


図 1-9　プログラム「drawRectangle」の解説

次のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse3　＞　MagicItemLCD1　＞　drawRectangle

実行結果：画面に四角形が表示される。
ちなみに「rectangle」は長方形を意味します。では、プログラムの中を見てみましょう。

プログラム「drawRectangle」より抜粋(ばっすい)

```
void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0, 0, 0);      //背景色は黒
    TFTscreen.stroke(255, 255, 255);    //線の色は白
    //  TFTscreen.fill(0,255,255);      //黄色の塗りつぶしを設定
    TFTscreen.rect(20, 30, 40, 50);     //(20,30)に(幅40,高さ50)の四角形を描く
}
```

新たな命令が出てきましたね。内容を確認しておきましょう。

命　　令「TFTscreen.rect」
実行結果：四角形を描く
使い方：TFTscreen.rect ([座標x1],[座標y1],[幅],[高さ]);
例：TFTscreen.rect (20, 30, 40, 50);
　　//(20,30) に（幅40、高さ50）の四角形を描く

やってみよう！

プログラム「drawRectangle」の中にある次の黄色いラインのコメントアウト（//）を削除して、プログラムを実行してみよう。どんな表示になるかな？

```
void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0, 0, 0);    //背景色は黒
    TFTscreen.stroke(255, 255, 255); //線の色は白
    //   TFTscreen.fill(0,255,255);     //黄色の塗りつぶしを設定
    TFTscreen.rect(20, 30, 40, 50);   //(20,30)に(幅40,高さ50)の四角形を描く
}
```

白枠で枠内が黄色い長方形ができましたか？

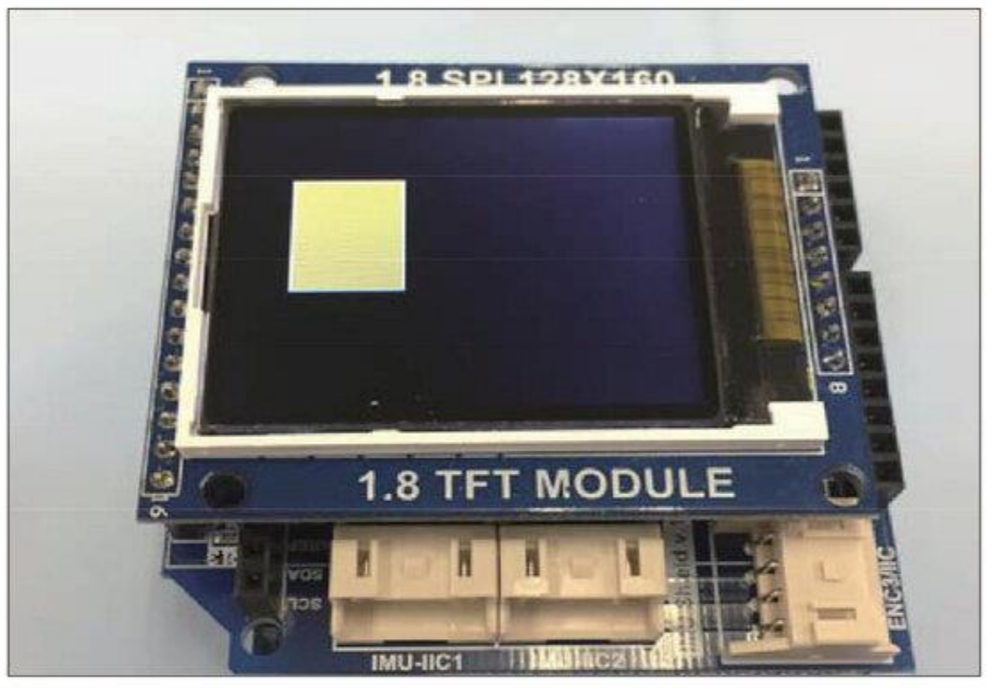


図1-10　プログラム「drawRectangle」の修正後の表示

塗りつぶしは、TFTscreen.fill(0, 255, 255);で設定します。ここでもRGBの並びに注意しましょう。

命　令　「TFTscreen.fill」

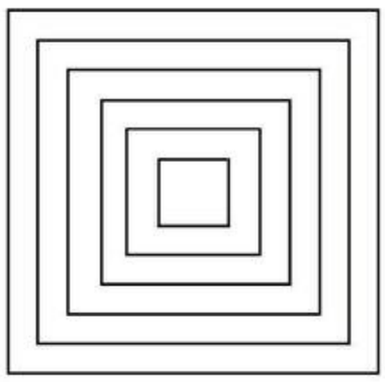
実行結果：描かれる図形を指定した色で塗りつぶす

使い方：TFTscreen.fill ([B 値], [G 値] , [R 値]);

例：TFTscreen.fill(0, 255, 255);　//黄色の塗りつぶしを設定

ステップアップ

プログラム「drawRectangle」を変更し、図のように四角形が重なるようにしてみよう。
一番小さい正方形の一辺の長さを20として、上下左右にそれぞれ10ずつ広げた正方形を描いていこう。

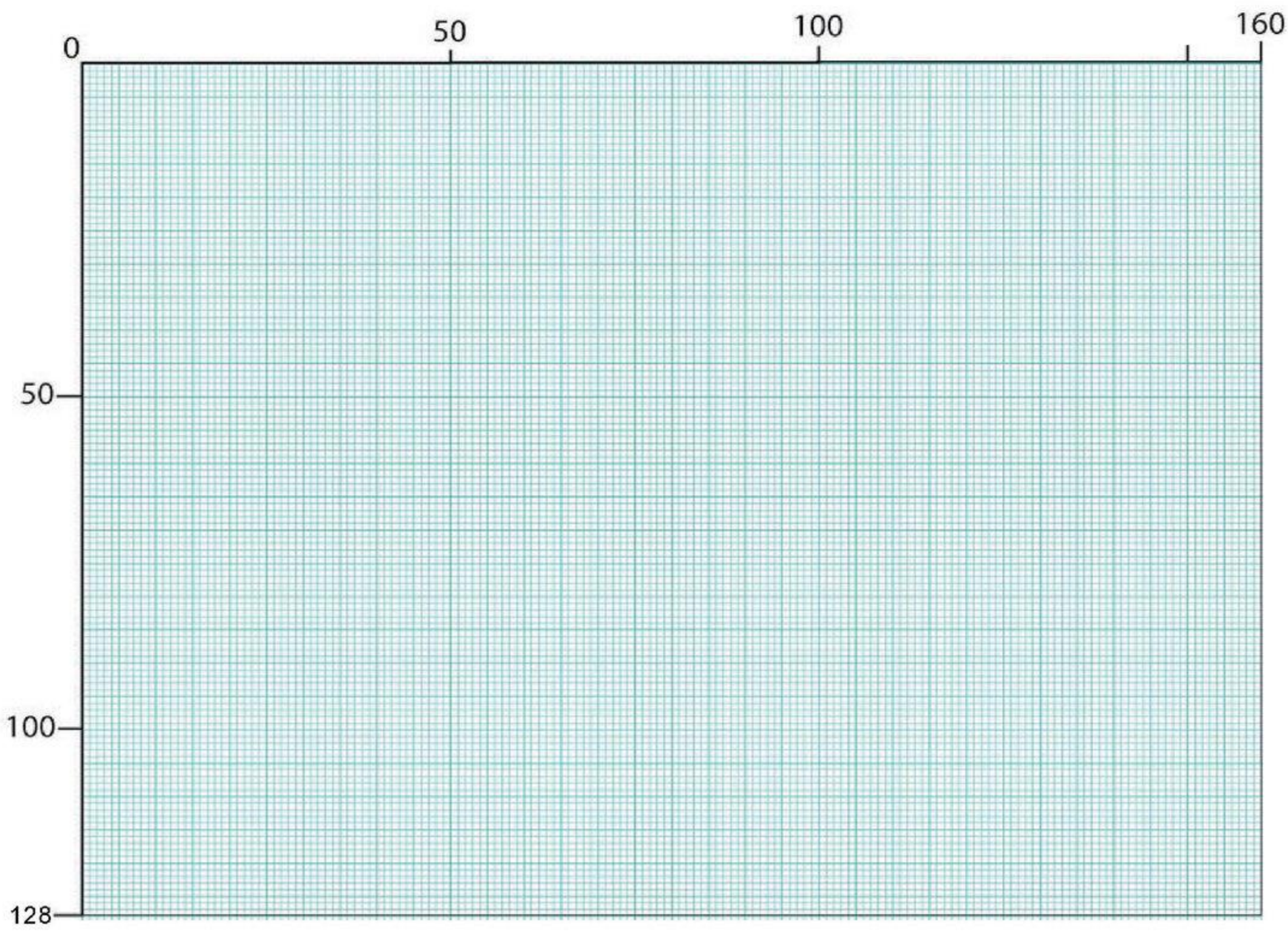


ヒント

「正方形の頂点の位置」「正方形の幅および高さ」をどんどん変更していく必要があるね！
for 文を使ってみよう！

講

解答プログラムは以下となります。
RoboticsProfessorCourse3 ＞ MagicItemLCD1 ＞ drawRectangleAns1
解答例は巻末に記載します。



1.3.　円を描いてみる

次は、円を描きます。円は中心点の位置と半径を設定します。

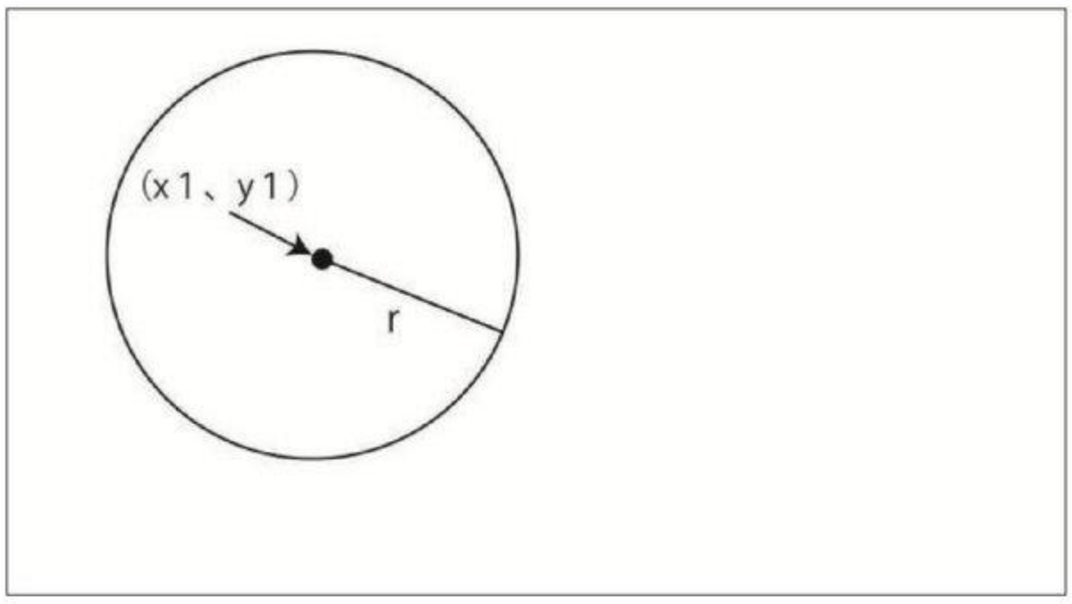


図 1-11　プログラム「drawCircle」の解説

次のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse3　＞　MagicItemLCD1　＞　drawCircle

プログラム「drawCircle」より抜粋（ばっすい）

```
void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0, 0, 0);     //背景色は黒
    TFTscreen.stroke(255, 255, 255);  //線の色は白
    //  TFTscreen.fill(255,255,0);      //水色の塗りつぶしを設定
    TFTscreen.circle(20, 30, 10);      //(20,30)に半径10の円を描く
}
```

命令　「TFTscreen.circle」

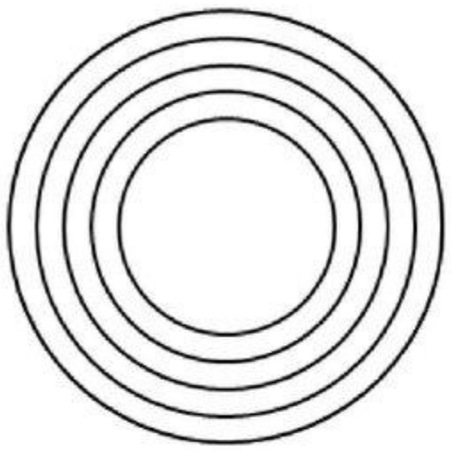
実行結果：円を描く

使い方：TFTscreen.circle ([座標x1],[座標y1],[半径r]);

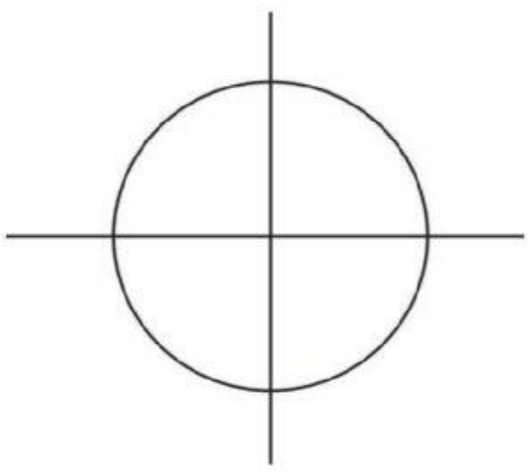
例：TFTcreen.circle(20, 30, 10);　//(20,30) に半径10の円を描く

ステップアップ

①大きさの異なるいくつかの円を同じ中心をもつように描いてみよう。たとえば、5重の同心円をfor文を使って描いてみよう。



②最初に十字を描いて、交点が中心になるように円を描いてみよう。



講

①と②の解答プログラムはそれぞれ以下となります。
RoboticsProfessorCourse3 > MagicItemLCD1 > drawCircleAns1
RoboticsProfessorCourse3 > MagicItemLCD1 > drawCircleAns2
①と②の解答例は巻末にも記載します。

1.4.　三角形を描いてみる

今度は、三角形をディスプレイに描いてみましょう。三角形をかくには3つの頂点の座標を定める必要があります。

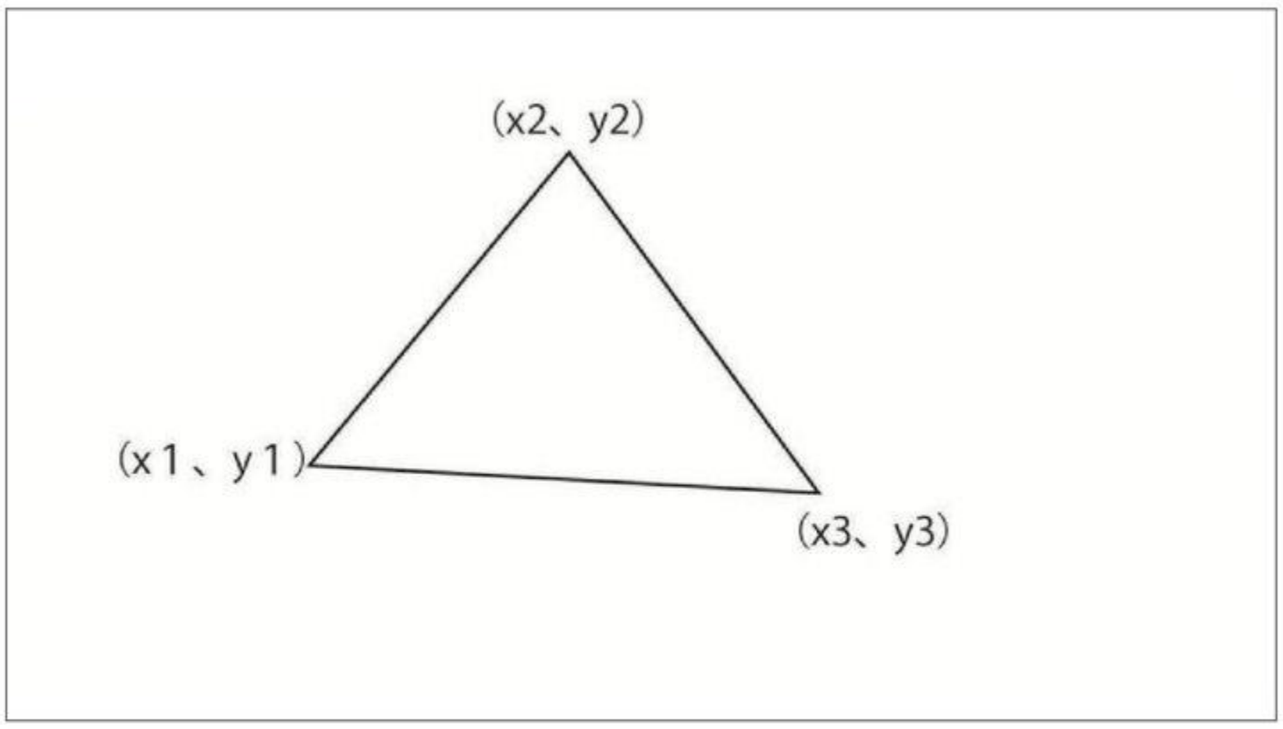


図 1-12　プログラム「drawTriangle」の解説

次のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse3　＞　MagicItemLCD1　＞　drawTriangle

実行結果：画面に三角形が表示される。

では、プログラムの中を見てみましょう。

プログラム「drawTriangle」より抜粋（ぱっすい）

```
void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0, 0, 0);                    //背景色は黒
    TFTscreen.stroke(255, 255, 255);                  //線の色は白
    TFTscreen.triangle(50, 0, 0, 100, 100, 100);
    //(50,0) (0,100) (100,100)を頂点とする三角形を描く
}
```

以下が新しい命令です。

命　令　「TFTscreen.triangle」

実行結果：三角形を描く

使い方：TFTscreen.triangle([座標x1], [座標y1], [座標x2], [座標y2],
　　[座標x3], [座標y3]);

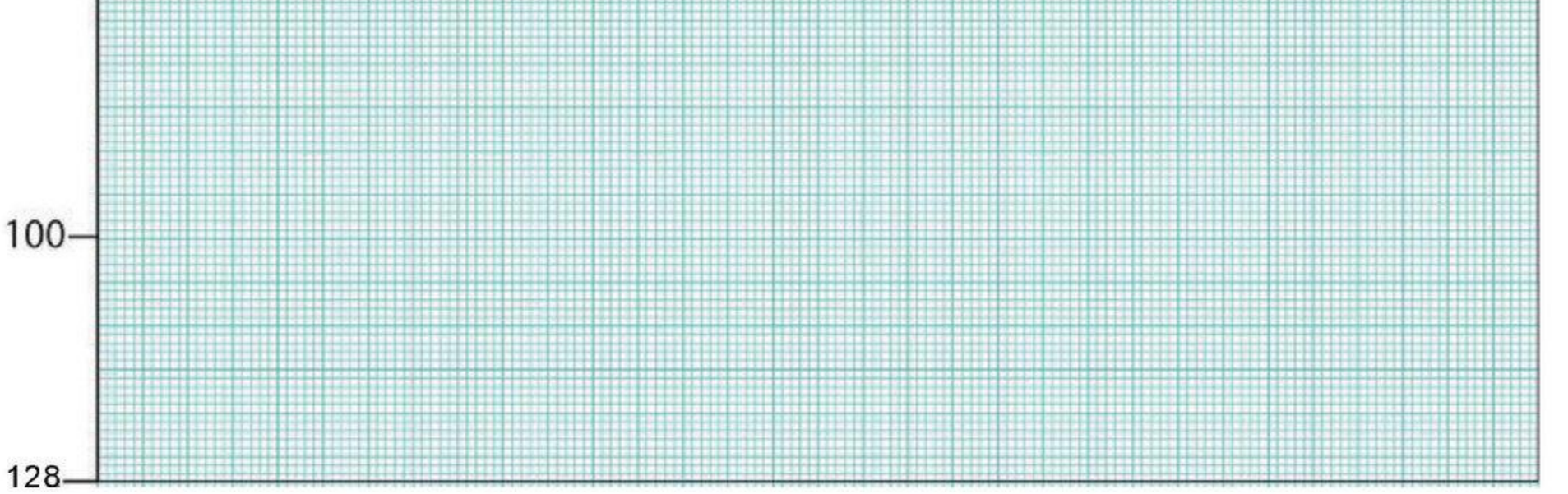
例：TFTscreen.triangle(50, 0, 0, 100, 100, 100);
//(50,0) (0,100) (100,100)を頂点とする三角形を描く

３つの頂点座標を与えると、その頂点を線で結んでくれます。

やってみよう！

頂点座標
（10,10）（40,25）（25,40） の三角形を描いてみよう。

方眼紙も使ってイメージを膨(ふく)らませましょう。

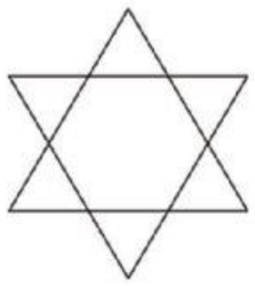


講

解答プログラムは以下となります。
RoboticsProfessorCourse3 ＞ MagicItemLCD1 ＞ drawTriangleAns1

チャレンジ課題

六芒星（ろくぼうせい）を描いてみよう。

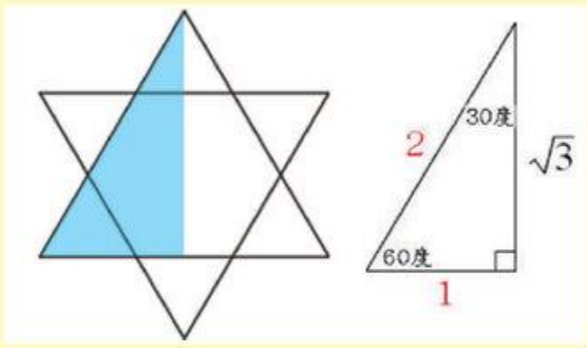


講

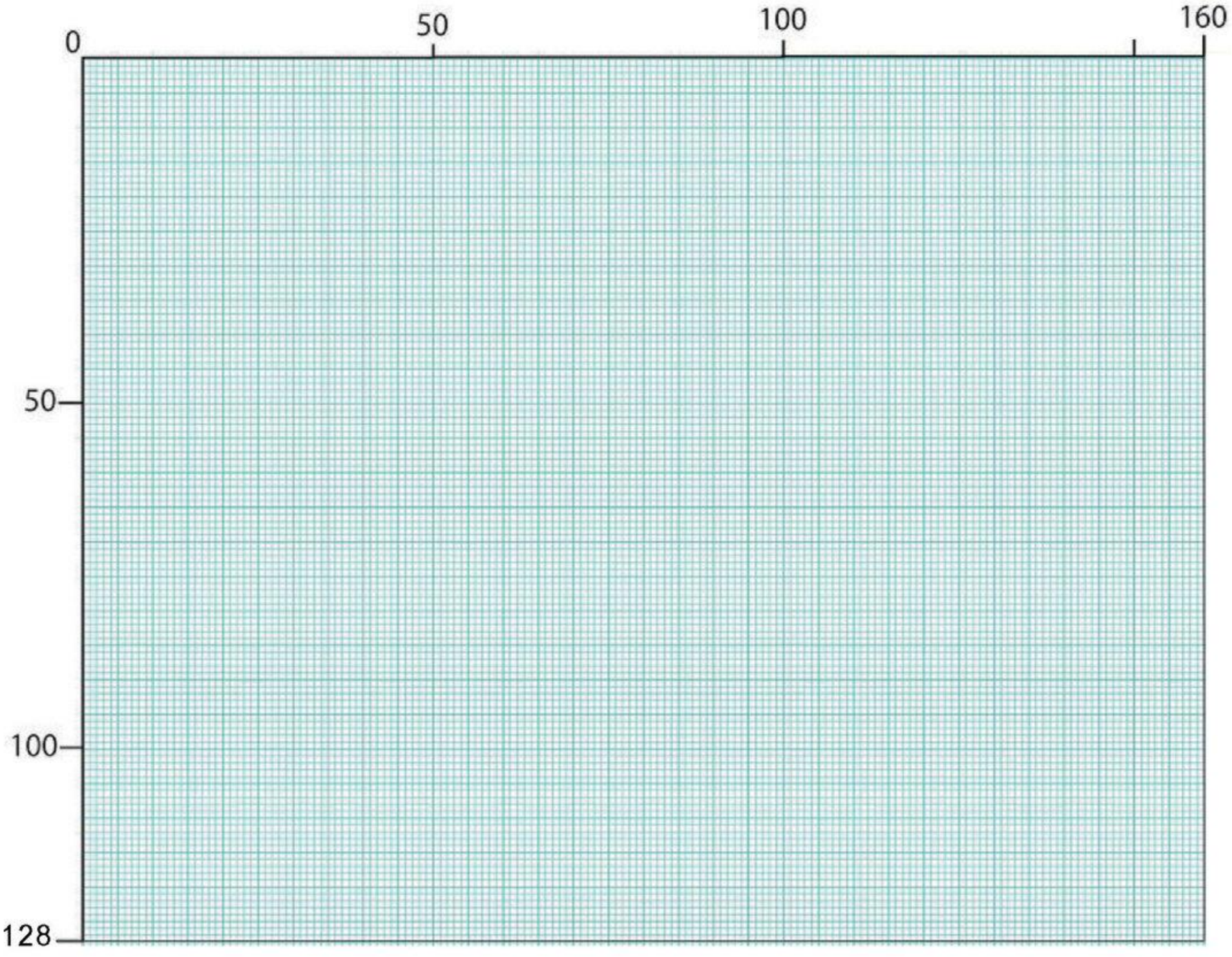
解答プログラムは以下となります。
RoboticsProfessorCourse3 ＞ MagicItemLCD1 ＞ drawTriangleAns2
解答例は巻末に記載します。
なお、方眼紙で座標を確認して、三角形を出来るだけ正確に重ねてプログラムしても構いません。
生徒の学年や理解度に応じて様々な方法がありますので、トライアルアンドエラーでチャレンジさせてください。

ヒント

合同な２つの正三角形が組み合わさっているよ！
正三角形を、下図の青く塗った部分だけ残すように半分に切ると直角三角形になるね。この三角形の3辺の長さは、必ず1：2：$\sqrt{3}$（約1.73）になるようにできているよ！正三角形の高さを求めるときに使ってみよう！



計算が難しい場合は、まずは方眼紙を使って六芒星（ろくぼうせい）を描いてみよう。



2．まとめ（目安５分）

今回は、液晶ディスプレイに図形の表示をさせる方法の基礎を学びました。マトリクスLEDの復習にもなったかと思います。今回の方法を応用すれば、基本の図形を組み合わせて、自由に絵をかくこともできます。

次回は、この表示方法を使って、センサーの値などをもっとわかりやすく表示していきます。ロボットの状況がさらに詳しくわかってくると、プログラムをもっとかしこくできますよ。

《次回必要なもの》

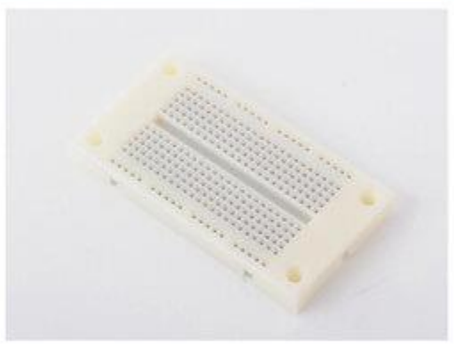
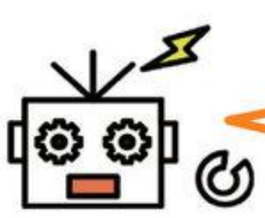
USB ケーブル 1	マイコンボード 1	ロボプロシールド 1	タッチセンサー 2
			
超音波距離センサー 1	301 ブレッドボード 1	ジャンパー線 65	可変抵抗ボリューム 2
			
タクトスイッチ 10	姿勢検出シールド 1	液晶ディスプレイシールド 1	
			

図 2-0　次回必要なもの



次回は、センサーの反応を表示させる体験をするよ！

講

○以下の理解度を確認します。
　液晶ディスプレイについて学ぶ
　液晶ディスプレイに図形を表示させる
　図形をオフセットさせるプログラムを学ぶ

○次回のテーマは、「液晶ディスプレイにセンサー情報を表示させる」であることを告知します。

P.12　ステップアップ①　解答例「drawLineAns1」

```
#include <TFT.h>
#include <SPI.h>

TFT TFTscreen = TFT(A2, 1, 9);          //液晶ディスプレイを使うときのオマジナイ

void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0, 0, 0);      //背景色は黒
    TFTscreen.stroke(255, 255, 255);    //線の色は白
    for (int y = 0; y <= 128; y += 10) //線のy座標を10ずつ増加させる
        TFTscreen.line(0, y, 160, y);  //(0,y)から(160,y)へ線を描く
}

void loop(){
}
```

P.12　ステップアップ②　解答例「drawLineAns2」

```
#include <TFT.h>
#include <SPI.h>

TFT TFTscreen = TFT(A2, 1, 9);          //液晶ディスプレイを使うときのオマジナイ

void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0,0,0);        //背景色は黒
    TFTscreen.stroke(255,255,255);      //線の色は白
    for(int y = 0; y <= 128; y += 10)  //横線のy座標を10ずつ増加させる
        TFTscreen.line(0,y,160,y);      //(0,y)から(160,y)へ線を描く
    for(int x = 0; x <= 160; x += 10)  //縦線のx座標を10ずつ増加させる
        TFTscreen.line(x,0,x,128);      //(x,0)から(x,128)へ線を描く
}

void loop(){
}
```

P.13　チャレンジ課題　解答例「drawLineAns3」

```
#include <TFT.h>
#include <SPI.h>

TFT TFTscreen = TFT(A2, 1, 9);              //液晶ディスプレイを使うときのオマジナイ

void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0,0,0);            //背景色は黒
    TFTscreen.stroke(255,255,255);          //線の色は白
    for(int y = 0; y <= 300; y += 10)
        TFTscreen.line(y,0,0,y);            //(y,0)から(0,y)へ線を描く
    for(int x = -150; x <= 150; x += 10)
        TFTscreen.line(x,0,160,160 - x); //(x,0)から(160,160-x)へ線を描く
}

void loop(){
}
```

P17　ステップアップ　解答例「drawRectangleAns1」

```
#include <TFT.h>
#include <SPI.h>

TFT TFTscreen = TFT(A2, 1, 9);              //液晶ディスプレイを使うときのオマジナイ


void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0, 0, 0);          //背景色は黒
    TFTscreen.stroke(255, 255, 255);        //線の色は白
    // TFTscreen.fill(0,255,255);           //黄色の塗りつぶしを設定
    for (int i = 0; i <= 100; i += 20){  //四角形の辺を20ずつ大きくする
        TFTscreen.rect(50 - i / 2, 50 - i / 2, 20 + i, 20 + i);
//(50-i/2,50-i/2)に(幅20+i,高さ20+i)の四角形を描く
    }
}

void loop(){
}
```

P.19　ステップアップ①　解答例「drawCircleAns1」

```
#include <TFT.h>
#include <SPI.h>

TFT TFTscreen = TFT(A2, 1, 9);              //液晶ディスプレイを使うときのオマジナイ


void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0, 0, 0);          //背景色は黒
    TFTscreen.stroke(255, 255, 255);        //線の色は白
    //  TFTscreen.fill(255,255,0);          //水色の塗りつぶしを設定
    for (int i = 0; i < 50; i += 10)        //半径を10ずつ大きくする
        TFTscreen.circle(50, 50, 10 + i); //(50,50)に半径10+iの円を描く
}

void loop(){
}
```

P.19　ステップアップ②　解答例「drawCircleAns2」

```
#include <TFT.h>
#include <SPI.h>

TFT TFTscreen = TFT(A2, 1, 9);          //液晶ディスプレイを使うときのオマジナイ

void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0, 0, 0);      //背景色は黒
    TFTscreen.stroke(255, 255, 255);    //線の色は白
    TFTscreen.line(0, 70, 160, 70);
    TFTscreen.line(70, 0, 70, 128);
    TFTscreen.circle(70, 70, 50);       //(70,70)に半径50の円を描く
}

void loop(){
}
```

P.22　チャレンジ課題　解答例　「drawTriangleAns2」

```
#include <TFT.h>
#include <SPI.h>
#include <math.h>

#define Ox 50
#define Oy 50
#define L  20                          //正三角形の一辺の長さ
#define RT3 1.73                       //√3

TFT TFTscreen = TFT(A2, 1, 9);         //液晶ディスプレイを使うときのオマジナイ

void setup(void){
    TFTscreen.begin();
    TFTscreen.background(0, 0, 0);     //背景色は黒
    TFTscreen.stroke(255, 255, 255); //線の色は白

    TFTscreen.triangle(
        Ox + 0, Oy -  L * RT3,
        Ox + L * 1.5, Oy + L * RT3 / 2,
        Ox - L * 1.5, Oy + L * RT3 / 2);

    TFTscreen.triangle(
        Ox + 0, Oy + L * RT3,
        Ox + L * 1.5, Oy - L * RT3 / 2,
        Ox - L * 1.5, Oy - L * RT3 / 2);
}

void loop(){
}
```