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

アームロボット②

第4回

アームロボットで文字や記号をかく

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。

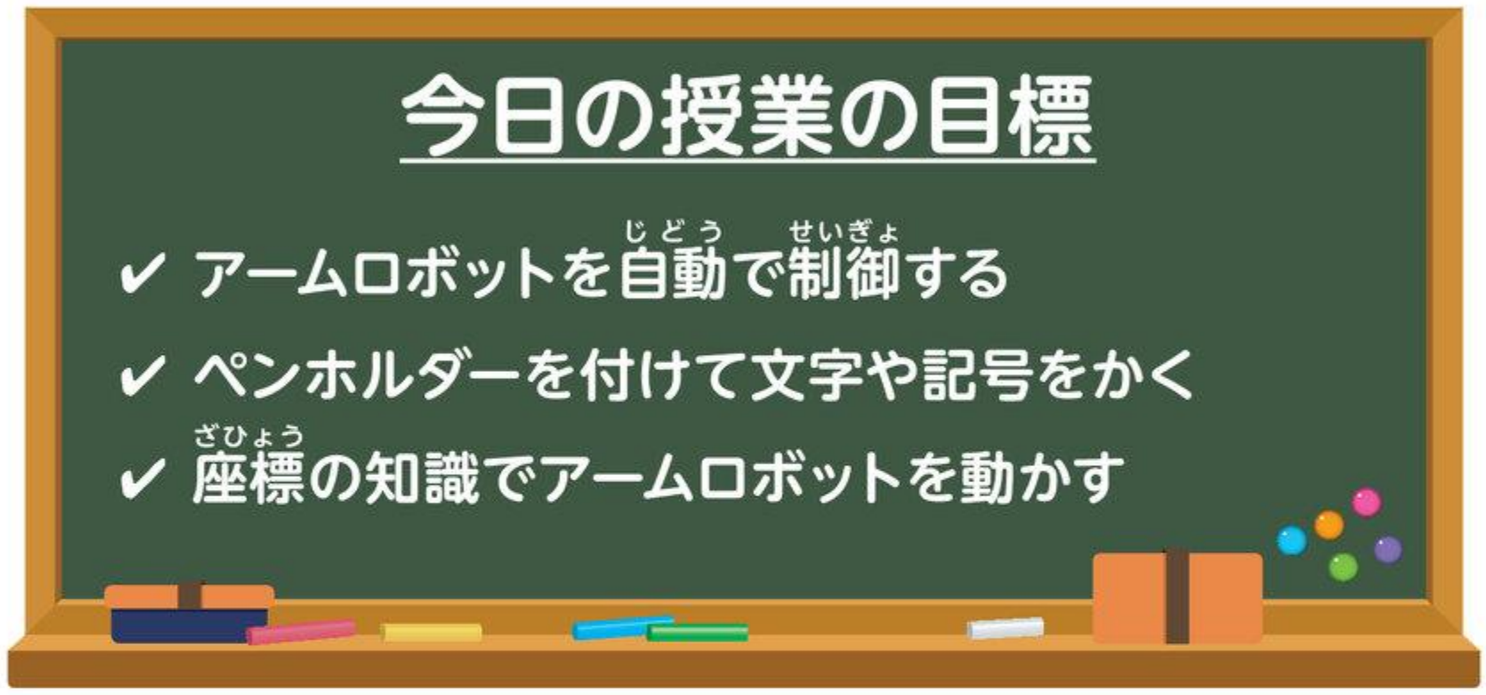
○　今回の目標を黒板に予め書いておきます。

（授業の目標を明確化することは大変従業なことですので、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 分程度を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．アームロボットで文字や記号をかく（目安５分）

0.0. 「アームロボットで文字や記号をかく」でやること



今回の授業では、アームロボットの自動制御方法を勉強します。
アームロボットのハンドをペンにかえて、文字や記号をかかせてみましょう。最初はペンを持てるようにロボットを改良し、手動でやってみます。なかなか思い通りにいかないかと思いますので、次にはプログラムを使いましょう。
自由に文字や記号をかくためには、数学の知識が必要です。
以前勉強した「座標」というものを覚えているでしょうか？　今回はその「座標」を活用します。数学の知識を活用して、もっと自由にロボットを使いこなせるようになりましょう！

ペンロボットに、勝手に学校のテストをやらせることもできるかなぁ？　よ～し、スゴイロボをつくるために勉強ダー！

0.1. 必要なもの

前回までに使っていたアームロボット本体の他に、以下のパーツを用意しましょう。

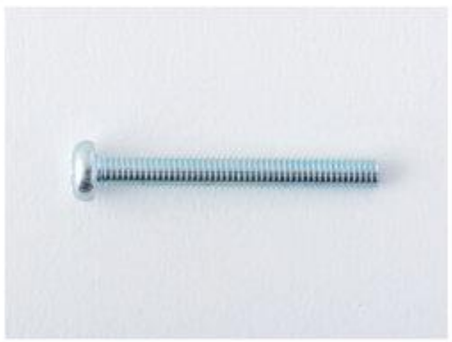
ドライバー 1	USB ケーブル 1	コントローラー 1	C-9 (アームロボットパーツ) 1
			
C-10 (アームロボットパーツ) 1	サインペン 1	AC アダプター 1	タイヤ 1
			
M3L25 ネジ 1			
			

図 0-0　必要なもの

1．アームロボットの自動制御（目安35分）

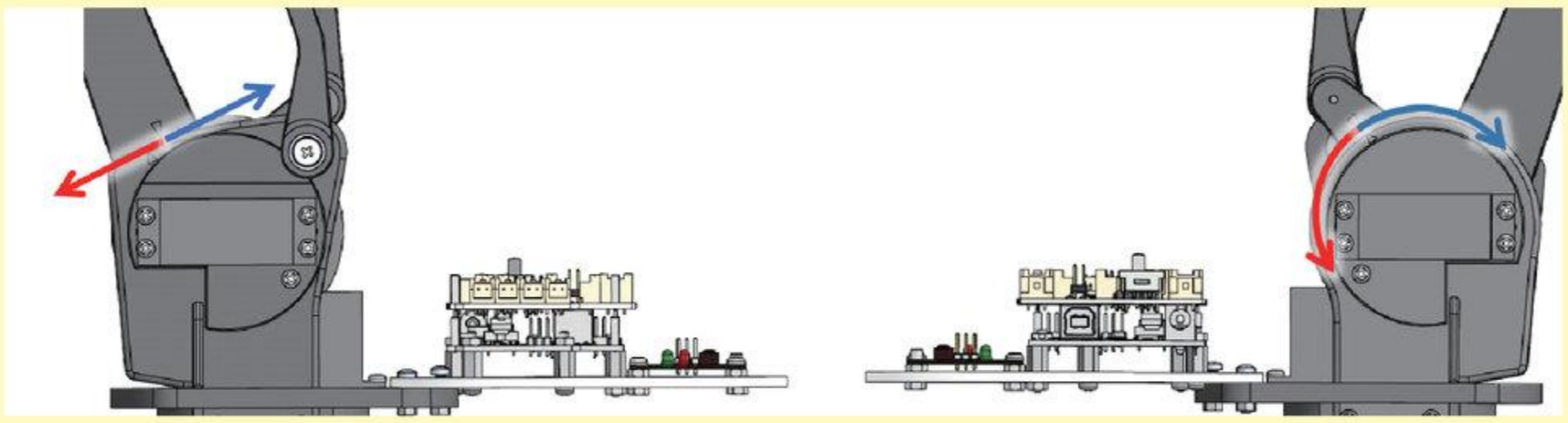
1.0. セミオートマチック（半自動）操作

1）動作確認

自動制御に挑戦する前に、まずは、前回と同じプログラム「semiAutoArm」で動作確認をします。そして、プログラムの原理の理解も深めましょう。
プログラム実行の前に、アームロボットの初期位置の調整が必要な人は前回確認した調整値を見直して、プログラムの中に値を書き込みましょう。

POINT

アームロボットの初期位置の調整は念入りに行いましょう。 servoL の調整は＋の角度を入力すると青い矢印方向、－の角度を入力すると赤い矢印方向に調整されます。 servoR の調整は、＋の角度を入力すると青い矢印方向、－の角度を入力すると赤い矢印方向に調整されます。
調整値は以降のプログラムでも必要なので、しっかり記録しておきましょう。



servoL　　servoR

プログラムの書き込み

RoboticsProfessorCourse2 > ArmRobot3 > semiAutoArm

プログラム「semiAutoArm」より抜粋

```
void setup(){
    pinMode(D3, INPUT_PULLUP);
                                // D3に接続されたタッチセンサーを読み取る
    ps2x.config_gamepad(13, 11, 10, 12, true, true);
                                // コントローラーを使うときのオマジナイ
    while(!digitalRead(D3));
    armBot.setup(0.0, 0.0, 0.0, 0.0);
                                // 調整値（servoR,servoL,servoROT,servoGRIPの順）
    delay(3000);
```

調整値を入力してください

armBot.setup(0.0,0.0,0.0,0.0);

( ) 内は、(S0 ｻｰﾎﾞ ﾓｰﾀｰ [R] , S1 ｻｰﾎﾞ ﾓｰﾀｰ [L], S2 ｻｰﾎﾞ ﾓｰﾀｰ [ROT] , S3 ｻｰﾎﾞ ﾓｰﾀｰ [GRIP]) の並びです。

コントローラーで操作し、各サーボモーターが思った方向に動くか確認しましょう。動かない部位があった場合は、ケーブルが正しく接続されているか、プログラムが正しく読み込まれているか、パーツが正しく組み立てられているか、などを確認しましょう。

2）「semiAutoArm」のポイント

「semiAutoArm」は、「ArmControl」と同じくコントローラーでアームロボットを制御（せいぎょ）するプログラムです。
ただ、「ArmControl」が「角度」を指定するプログラムだったのに対し、こちらは少しちがった形で命令を出しています。
プログラムの中身を見てみましょう。

プログラム「semiAutoArm」より抜粋（ばっすい）

```
armBot.setPosition(xx, zz, rr);

if(ps2x.Button(PSB_PAD_UP)){      // もし十字ボタンの上が押されたら
    armBot.gripperCatch();        // ハンドを閉じる
}

if(ps2x.Button(PSB_PAD_DOWN)){    // もし十字ボタンの下が押されたら
    armBot.gripperRelease();      // ハンドを開く
}
```

命　　令　「setPosition」
実行結果：指定の位置にハンドの先端が来るよう、モーターを回転させる
使 い 方：armBot.setPosition(80, 20, 50);
// 前方 80mm、高さ 20mm の位置にハンドを移動させ、ベース部を 50 度回転させる

「縦・横」の 2 方向だけ指示すればよかったマトリクス LED とちがい、アームロボットには「高さ」の指定も必要です。そのため、「前後方向・高さ・左右回転」の 3 つの値を書いていますね。

ちなみに、十字キーの上・下を押したときの命令も確認しておきます。

命　　令「gripperCatch」
動作結果：ハンドを閉じる

命　　令「gripperRelease」
動作結果：ハンドを開く

この 2 つの命令は、値を指定する必要はありません。

1.1. オートマチック（自動）操作

いよいよ自動制御に挑戦です。プログラム「AutoArmCatch」を実行します。
さきほどのプログラムと同じように調整値を入力してから実行しましょう。

armBot.setup(0.0,0.0,0.0,0.0);

( ) 内は、(S0 ｻｰﾎﾞ ﾓｰﾀｰ [R] , S1 ｻｰﾎﾞ ﾓｰﾀｰ [L], S2 ｻｰﾎﾞ ﾓｰﾀｰ [ROT] , S3 ｻｰﾎﾞ ﾓｰﾀｰ [GRIP]) の並びです。

プログラムの書き込み

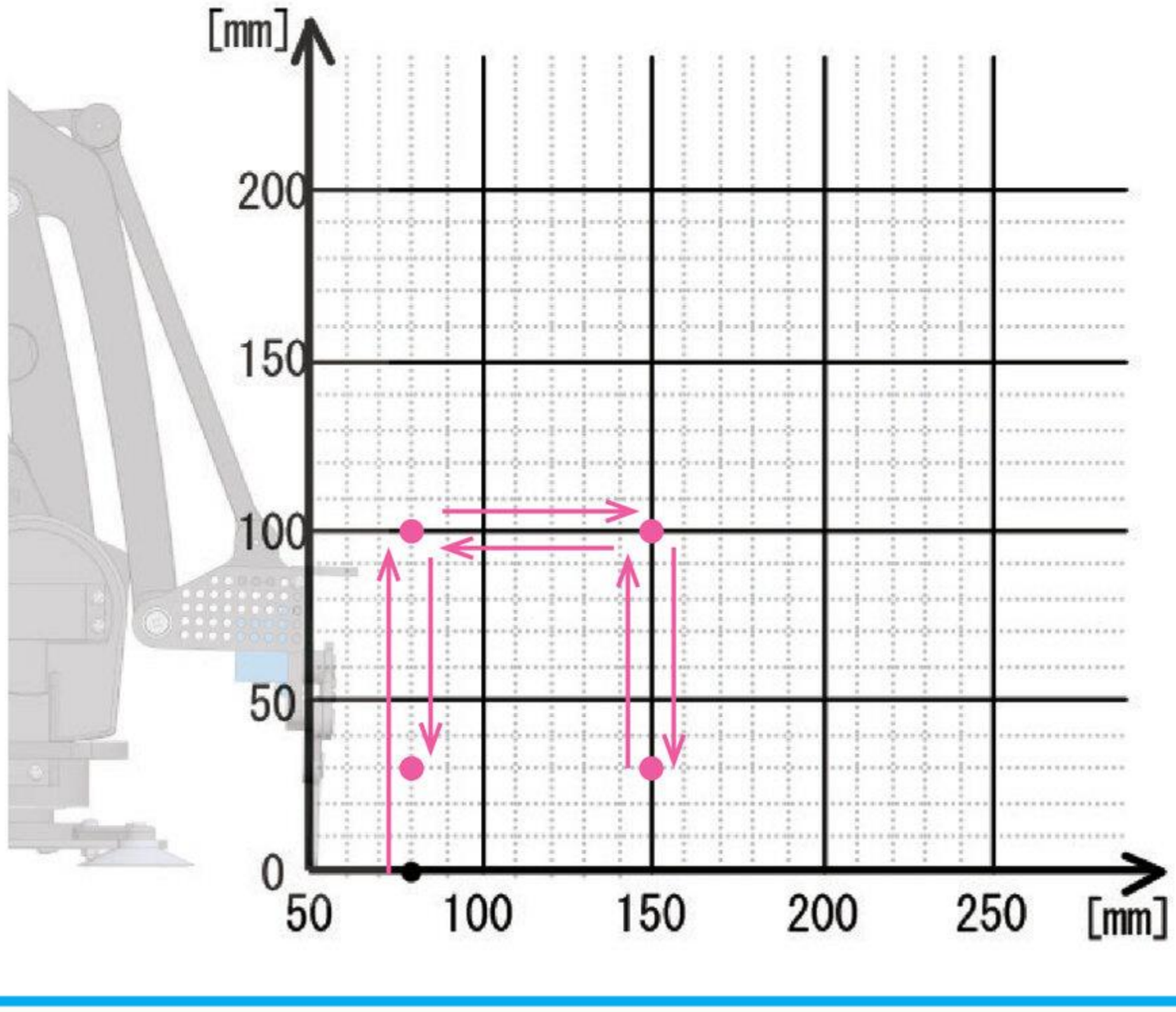
RoboticsProfessorCourse2 > ArmRobot3 > AutoArmCatch

実行結果：D3 に接続したタッチセンサーを押す回数によって、それぞれ以下のような反応を示す。

1 度押す → アームロボットの調整値を反映する
2 度押す → 動作開始位置 setPosition にアームが動く
3 度押す → 動作開始

やってみよう！

プログラム「AutoArmCatch」からアームの先がどのような座標を通るか読み取り、以下の図にプロットしてみよう。あらかじめプロットしてある黒い点は void setup() 内の setPosition(80, 0, 0); の部分だよ。

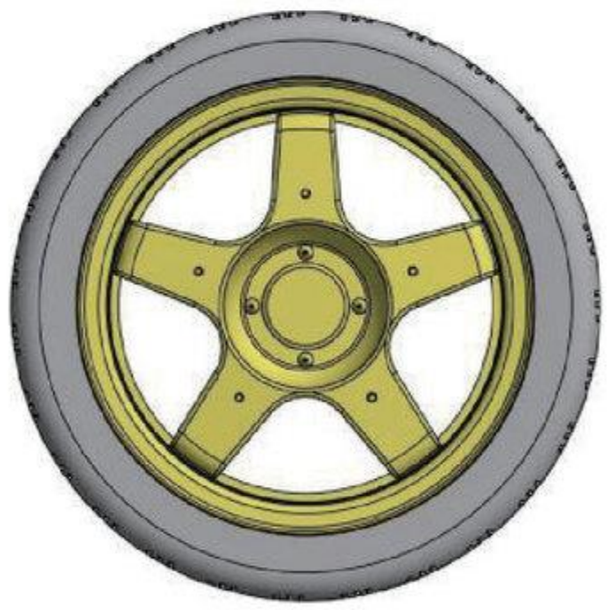


では今度は、実際にものを置いて、アームロボットに自動搬送をさせてみましょう。

やってみよう！

プログラム「AutoArmCatch」でタイヤを上手にキャッチしてみよう。
タイヤは、ハンドがちょうどキャッチできる位置に置こう。接地面から吸盤が浮いた状態だと、上下方向の制御に影響がでるので、しっかりと取り付けてね。

キャッチするには、ハンドに引っ掛かりやすいタイヤが最適だよ。持参してきていない場合は他の物を代用しよう。



講　プログラム「AutoArmCatch」を実行後、タッチセンサーを3回押すと動作開始になりますが、2周目からはタッチセンサーを1回押せば動作が開始されます。

2. ペンホルダーを付けて文字や記号をかく
（目安 65 分）

2.0. ペンホルダーの取り付け

アームロボットの手先を付けかえて、サインペンを固定します。
まず、ハンドを取り外し、それから図のようにペンホルダーを取り付けていきます。
ネジ類はハンドがついていた時と同じものを使用します。

図 2-0　ペンホルダーへの改造

図 2-1　ペンホルダーの取り付け位置

＜組み立て手順①＞

C-9 パーツを M3L10 タッピングネジ B（× 2）で取り付けます。C-9 パーツの上下に注意しましょう。

ネジ位置は赤い点が目印となります。

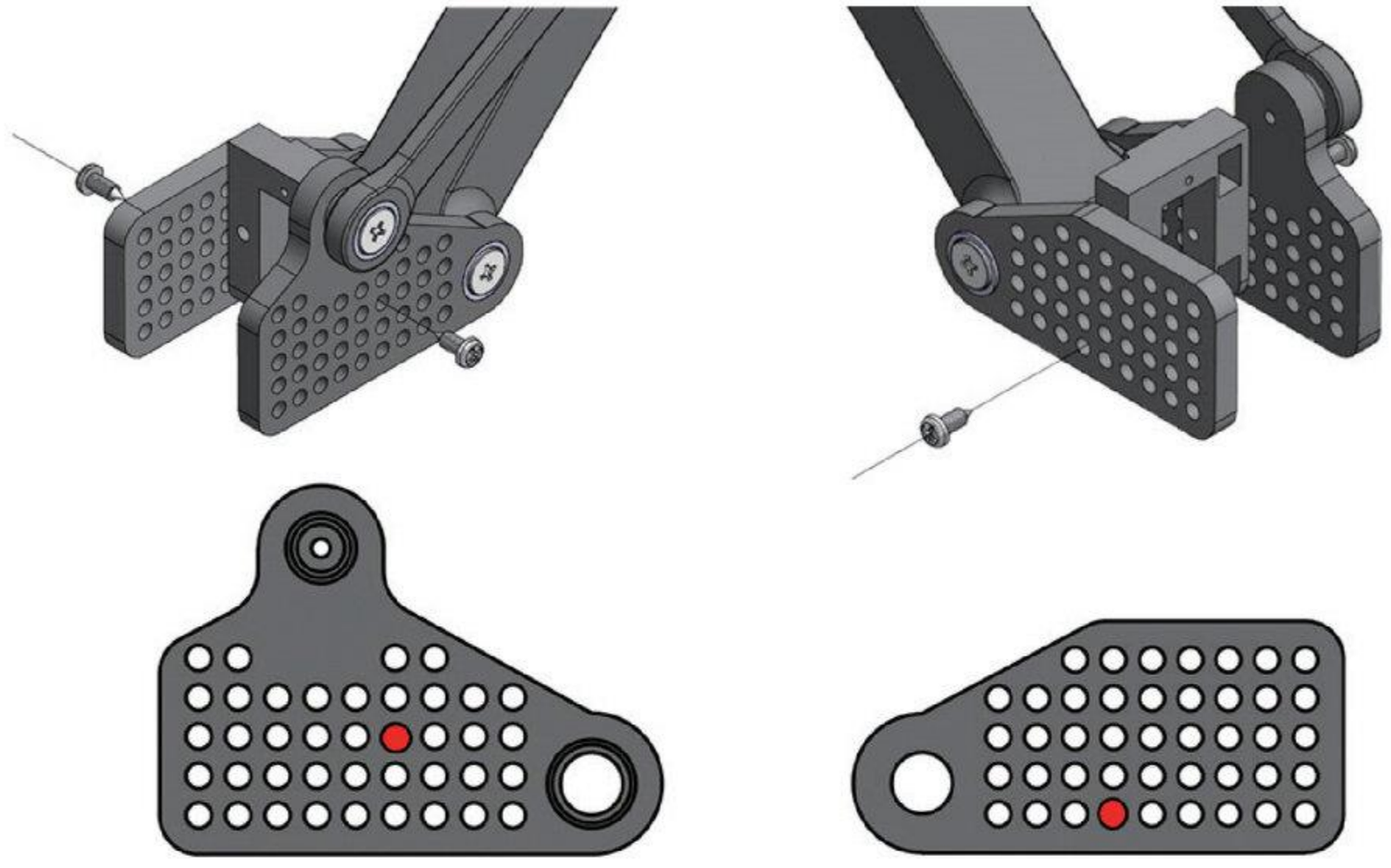


図 2-2　C-9 パーツの取り付け

＜組み立て手順②＞

サインペンをセットして、その上から C-10 パーツで挟(はさ)みます。 最後に、M3L25 ネジを C-10 パーツのネジ穴から C-9 パーツのネジ穴に通してサインペンを固定します。このときに、サインペンが手で上下に動かせるくらい緩(ゆる)めに加減しましょう。

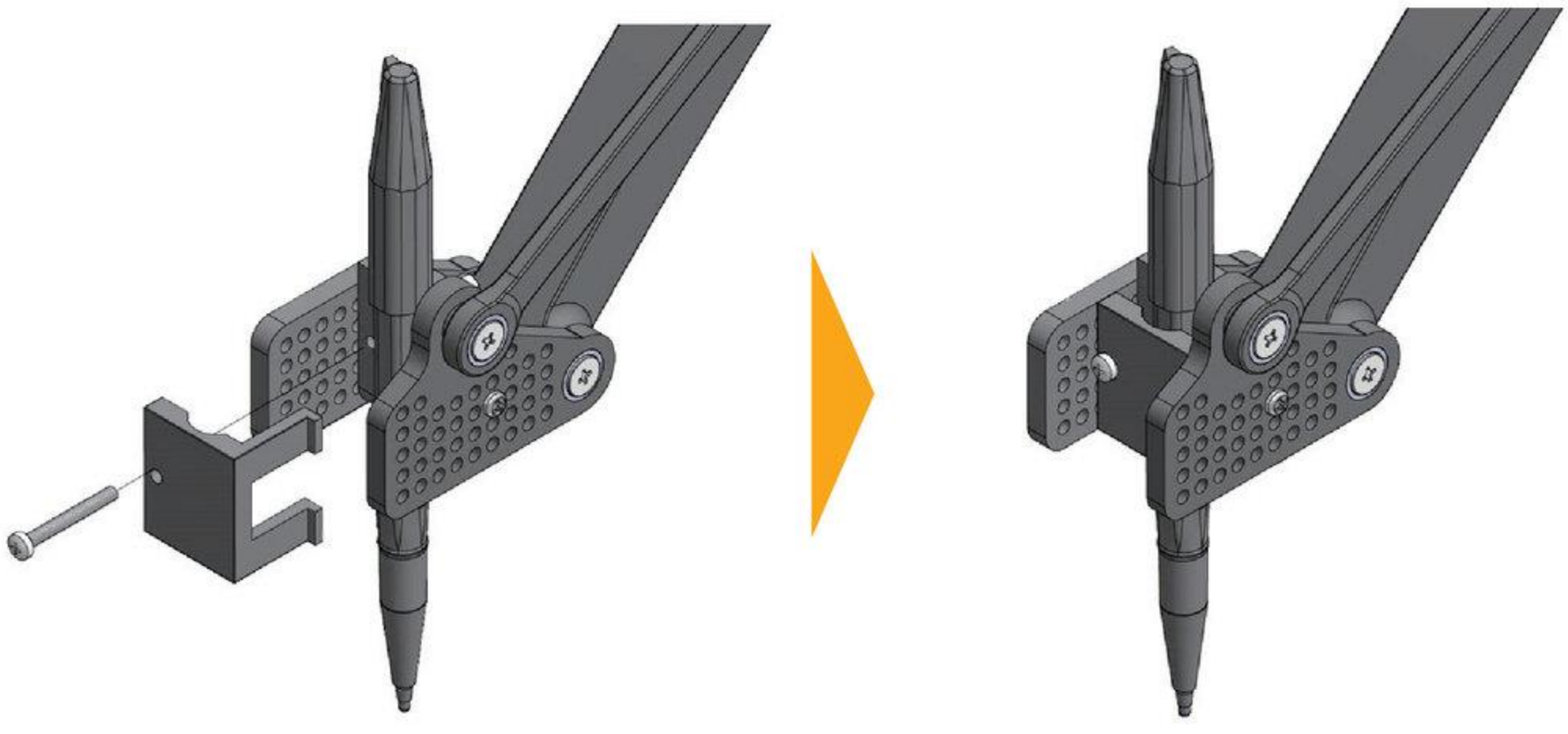


図 2-3　C-10 パーツとサインペンの取り付け

2.1. ホルダーの調整と手動操作

プログラムを実行する前にサインペンの調整をしましょう。
さきほど取り付けた M3L25 ネジを調整して、サインペンを手で抜いてみたとき力を入れずに抜ける程度にします。後ろに付けたペンキャップに引っかかっている程度にします。
しっかりと固定するとペン先が潰れてしまうことがありますので注意しましょう。
調整が終わったら、「semiAutoArm」のプログラムを実行して、コントローラーの操作で紙に好きな文字や記号をかいてみましょう。

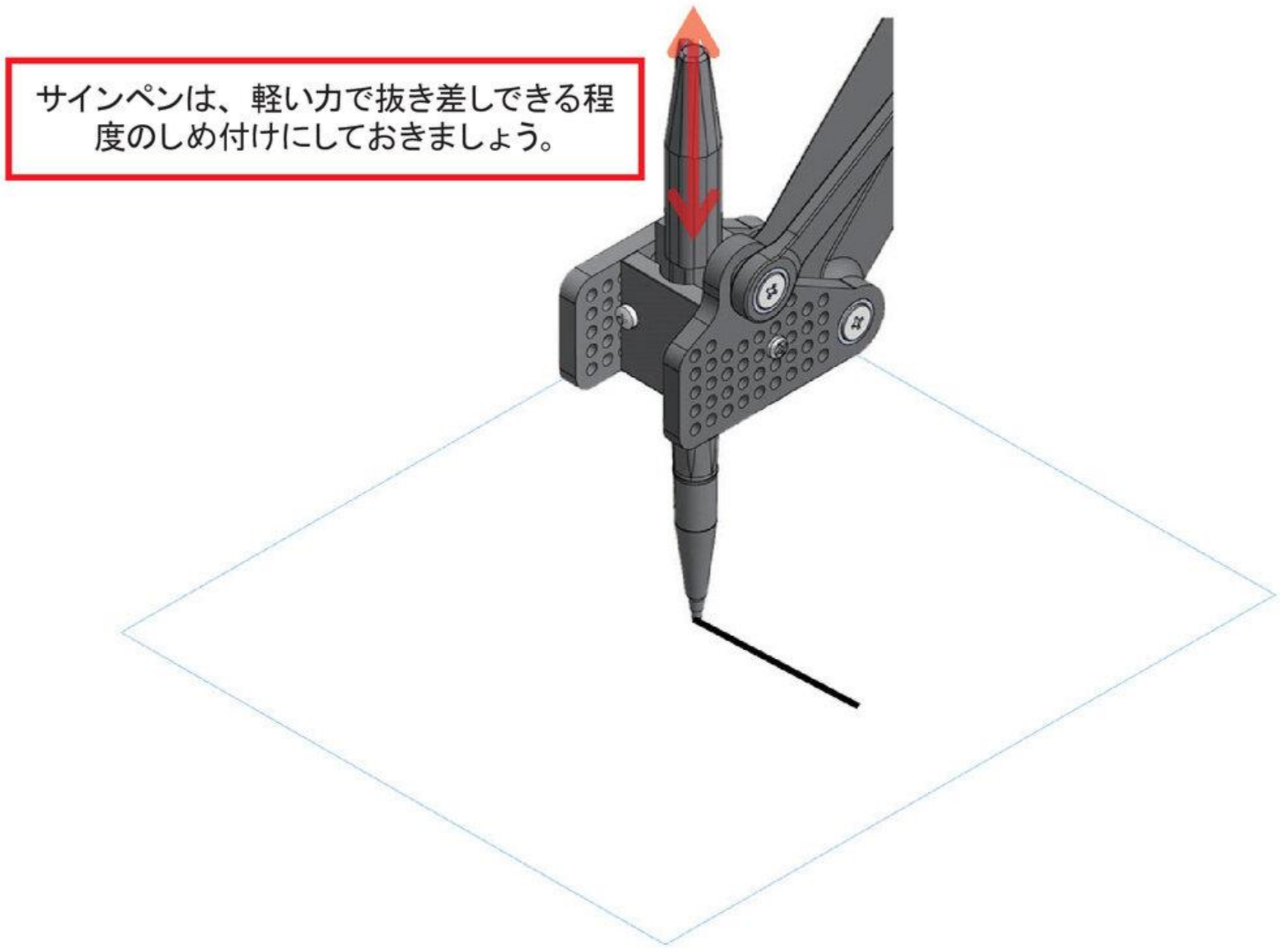


図 2-4 ペンの調整

どうですか？　手動でかくのはなかなか難しかったかと思います。
続いては、プログラムの力で自動でかくことにチャレンジをしてみましょう！

講

ペンホルダーにおさまるサイズであれば、付属のサインペン以外を使用することもできます。
ただし、アームロボットはその制御方法上、あまり硬いペン先のもの（ボールペンなど）や、筆圧の細かい制御が必要なもの（芯の折れやすいシャープペンなど）は向きません。
反対に、筆ペンなどある程度上下しても問題ないペンでの筆記を得意としています。

2.2. 文字や図形をかいてみよう

いよいよ自動で文字や記号をかくプログラムを実行します。
ロボットがかいているときに紙がズレたり折れ曲がったりしないように、手をそえるか、テープで四すみを貼り付けるなどして固定しましょう。
また、初期位置の調整が必要な場合は、プログラムへの入力を忘れないようにしましょう。

armBot.setup(0.0,0.0,0.0,0.0);

( ) 内は、(S0 ｻｰﾎﾞ ﾓｰﾀｰ [R] , S1 ｻｰﾎﾞ ﾓｰﾀｰ [L], S2 ｻｰﾎﾞ ﾓｰﾀｰ [ROT] , S3 ｻｰﾎﾞ ﾓｰﾀｰ [GRIP]) の並びです。

では、以下のプログラムを実行してください。 D3 に接続されたタッチセンサーが動作開始の合図です。

プログラムの書き込み
RoboticsProfessorCourse2 > ArmRobot4 > drawT

実行結果：“T”をかく。

次は図形に挑戦してみます。線があまりに歪(ゆが)むようだったら、サインペンのしめ付け具合を微調整しながらすすめましょう。では、必要に応じて調整値を入力して、以下のプログラムを実行してタッチセンサーで動作をさせてみましょう。

プログラムの書き込み
RoboticsProfessorCourse2 > ArmRobot4 > drawSquare

実行結果：四角の軌跡(きせき)をかく。

講

●ペンホルダーの調整
ペンホルダーの調整は、サインペンが紙に届く高さを確認してからしめ付け具合を調整させてください。ネジをきつくしめ過ぎると、アームロボットの力でペン先がつぶれたり、かいている最中に紙が破けたりします。最初はサインペンがペンホルダーに引っかかっている程度にしてかいて、具合を見ながら微調整させてください。

●文字や図形の精度
アームロボットのもともとの性能はもちろんのこと、個体差や環境差などが起因することによってあまりきれいな線を引くことができないこともあります。しかし、本題からそれず、「座標を使うことによりアームの制御を行っている」ということを学習のポイントにしてください（工業用ロボットも多軸の座標で制御されています）。

2.3.　文字や図形をかくしくみ

ここで、文字や図形をかくしくみを少し勉強しましょう。冒頭でも紹介しましたが、「座標」を利用しています。
以前勉強しましたが、座標とは住所のようなものです。何丁目何番地というように、場所がわかるように数字をつけることです。今回使うのは「直交座標」とよばれます。碁盤の目のように、平面を仕切って、x,y 軸を立てて数字を振り分けます。
下図のように数字を場所に割り振ります。するとロボットがどこに行けば良いか、とてもわかりやすくなるのです。

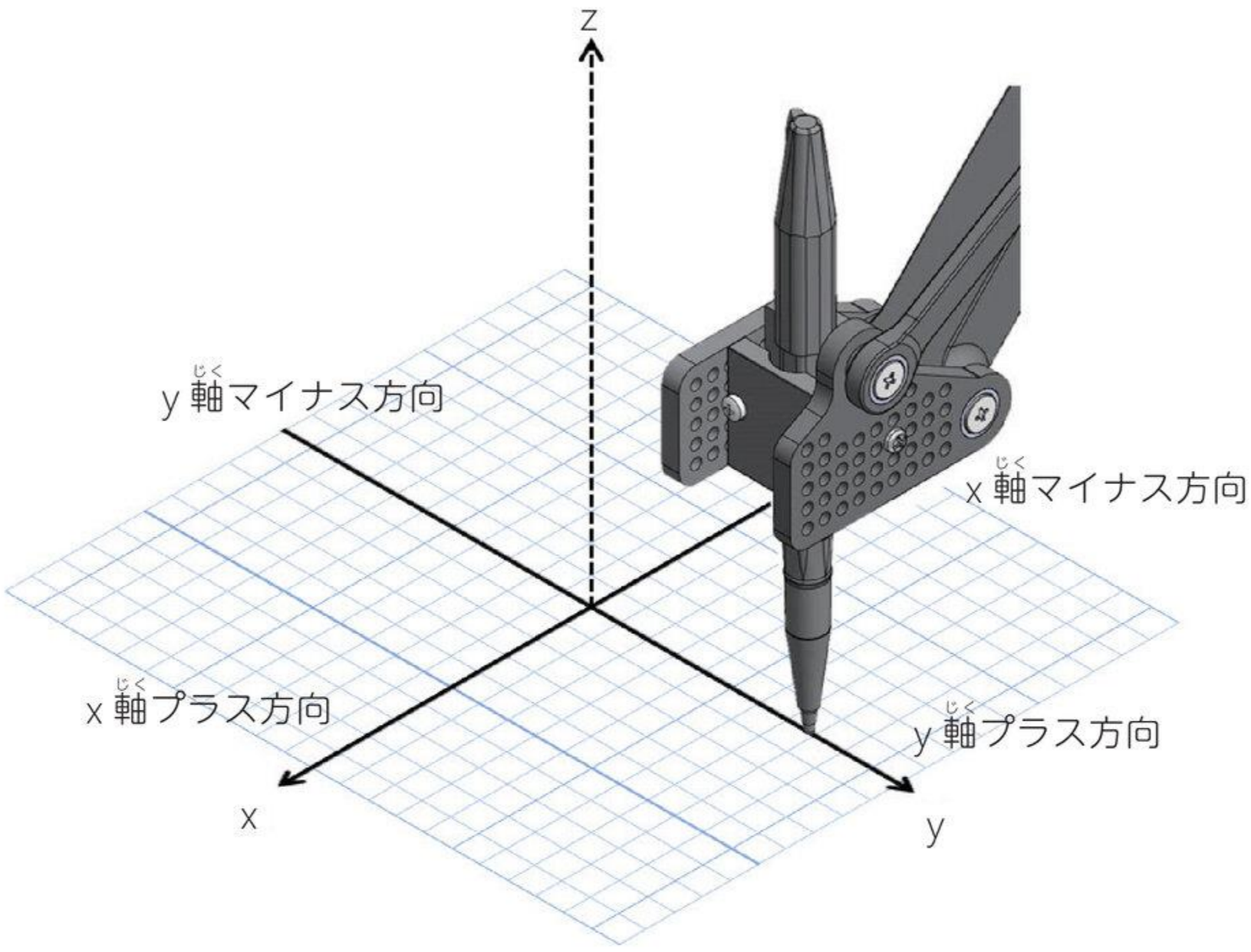


図 2-5　アームロボットの座標軸

プログラム内では平面方向を xy、上下方向を z の数値にて指定しています。ただし、今回は高さの指定はプログラム内で決めているので、x と y だけを見ていきましょう。
アームロボットから向かって、x 軸が前後方向の動き、y 軸が横方向の動きになります。
すでに習っている人はピンときたかもしれませんが、通常、１次関数や２次関数で使用するグラフは y が縦軸、x が横軸です。
アームロボットの座標軸は y が横軸、x が縦軸になるので、注意しましょう。

2.4. 座標のプログラム

それでは、さきほどのプログラム「drawT」の中身を確認してみましょう。
一つひとつ詳しく見ていきます。

プログラム「drawT」より抜粋

```
#define P_NUM 4  // 座標の数（変更するところ）
double start_point[1][2] = {
    0, 0
};
```

POINT

#define は Arduino 言語を含む、C 言語の便利な機能です。プログラム中の定数に対して名前を付けることができます。#define で定義された定数は値へと置きかえられます。

使い方：#define 定数名 値

プログラム「drawT」では、P_NUM という定数名をつけて値に「4」を入れています。
これは、サインペンが移動する座標の数が以下の図のように 4 点あることを示しています。
また、座標のポイントはかくものが複雑になるほど多くなります。

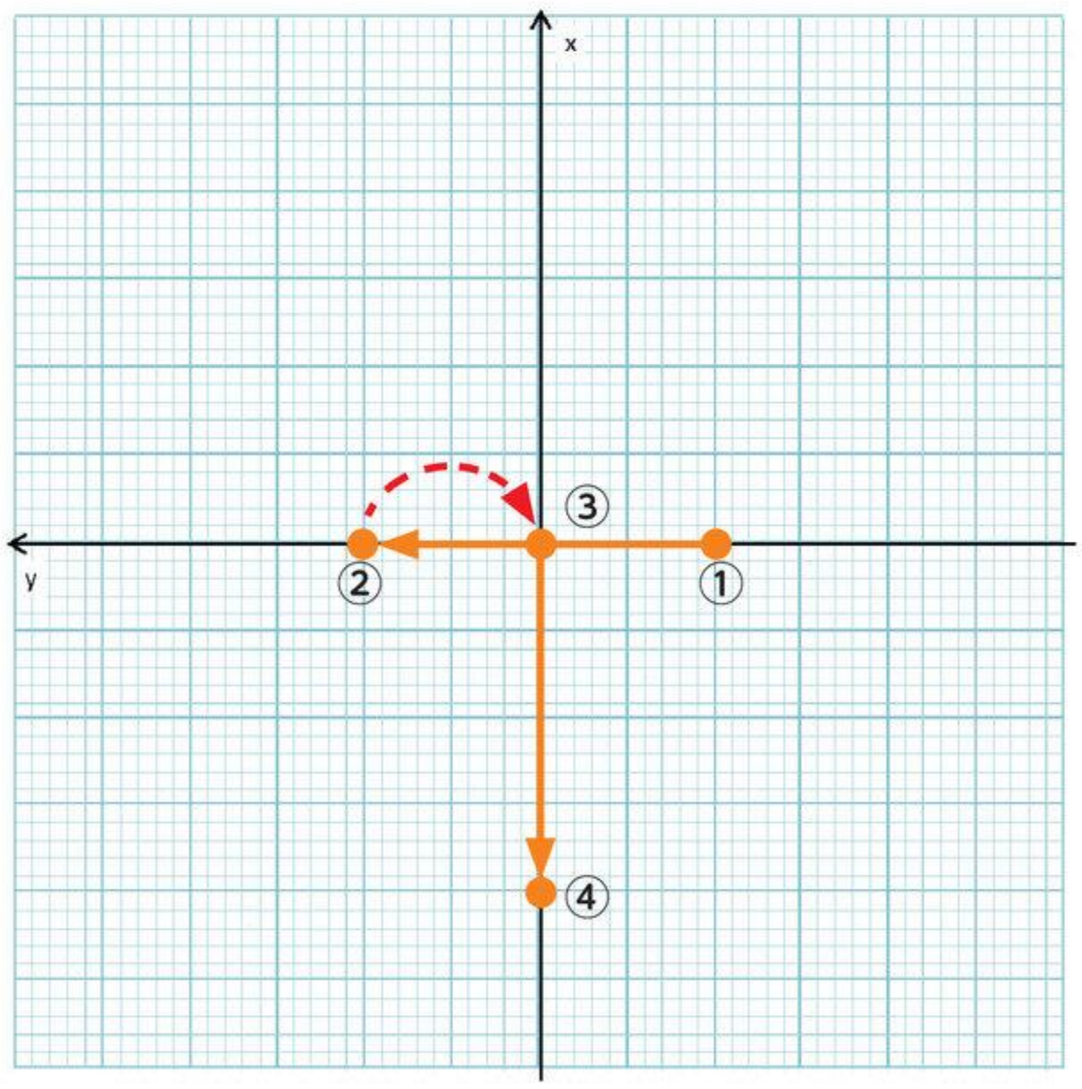


図 2-6　アルファベット「T」の場合の座標（ポイント）

アルファベットの「T」を手書きするのと同じ要領ですね。
①にペンを置き、②まで線を引き、一旦ペンを紙から離します。 そして、③にペンを置き、④まで線を引き、ペンを紙から離します。

POINT

double start_point[1][2] は、動作開始位置の座標を示しています。 {} 内に座標が入ります。
動作開始位置は x 軸、y 軸ともに 0（原点）です。

次に、void loop() 以降を見てみましょう。

プログラム「drawT」より抜粋

```
void loop(){
  while(!digitalRead(D3));    // D3に接続されたタッチセンサーを読み取る

  double point[P_NUM][2] = { // 移動座標（変更するところ）
      {
          0, -10
      }                          // Point1
      , {
          0, 10
      }                          // Point2
      , {
          0, 0
      }                          // Point3
      , {
          -20, 0
      }                          // Point4
};
```

POINT

ここでは、**図 2-6** の座標のポイントを示しています。 {} 内は x 軸、y 軸の座標が入ります。移動座標 x 軸と y 軸のプラス、マイナス方向の値を入れています。 さきほど説明した 4 つのポイントの座標を指定しています。

何かをかくときには点と線が必要ですが、ここまでで座標上の「点」を決めるプログラムはわかってきたでしょうか？ 今度は「線」を引くプログラムの 2 つの要素を見ていきましょう。

まず 1 つ目の要素はペンを紙に置く動作と、ペンを紙から離す動作を制御するプログラムです。

プログラム「drawT」より抜粋

```
int pen_status[P_NUM - 1] = {  //ペンの状態（変更するところ）
    DOWN,                      //Point1-2
    UP,                        //Point2-3
    DOWN                       //Point3-4
};
```

上記は、ペンの高さ（z 軸）を UP、DOWN で指定します。 2 点の座標の間の命令になります。

POINT

2 つの点を結ぶ線は、点の数より 1 つ少なくなりますよね。そのため、さきほど説明した `#define` で定義された、`P_NUM 4` から 1 を差し引き、`int pen_status[P_NUM-1]` と記述します。
線を引くときはペンを下ろし、引き終わったらペンを離すので、2 点間は `DOWN` か `UP` の命令をします。 Point が増えるときは、`DOWN` または `UP` を追記します。

では、他の文字や図形をかくサンプルプログラムも実行してみましょう。

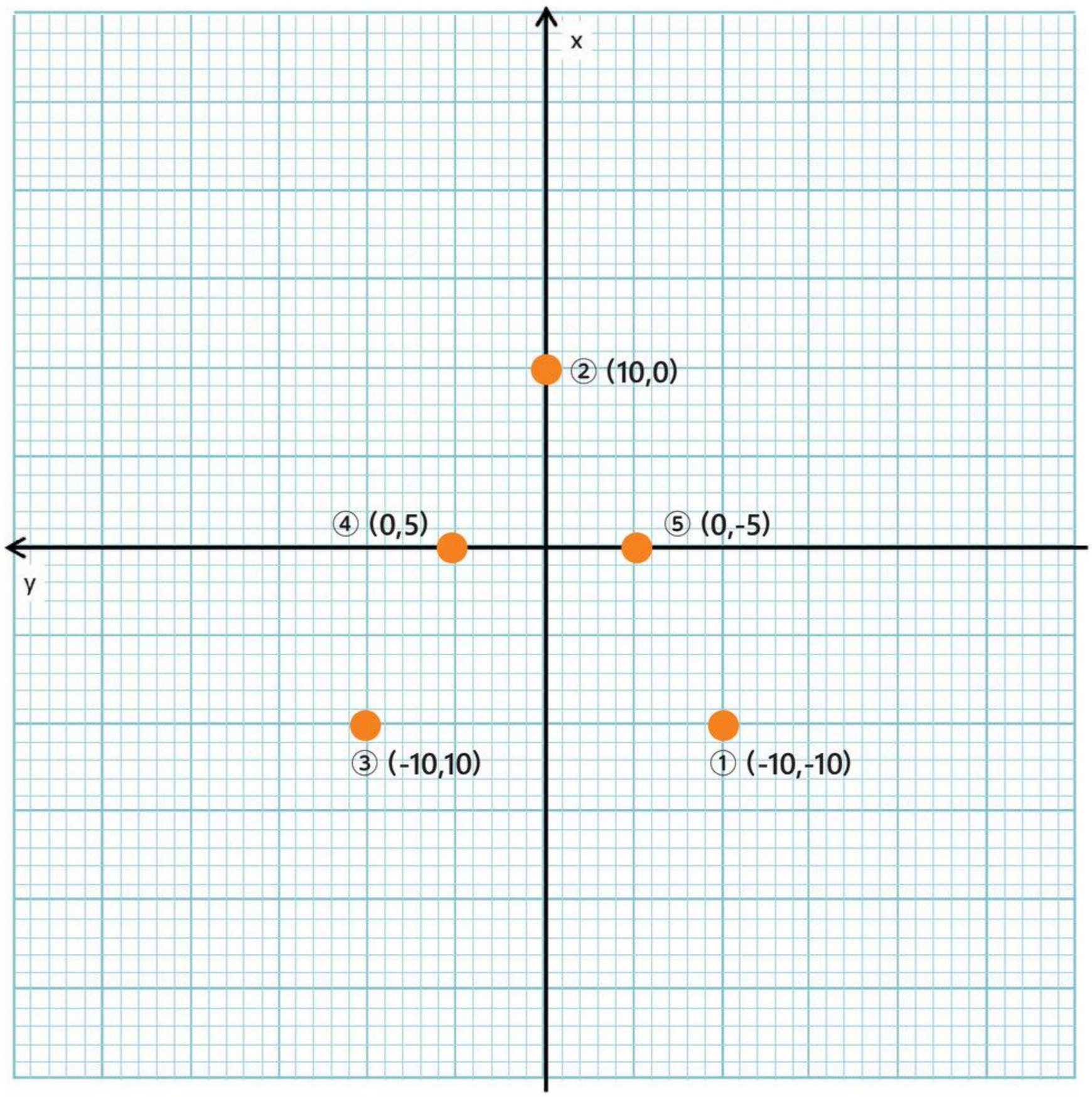
やってみよう！

記号の“☆（五芒星）”をかくプログラムを実行しよう。
実行後、プログラムの中身もチェックしよう。

プログラムの書き込み
RoboticsProfessorCourse2 ＞ ArmRobot4 ＞ drawStar

ステップアップ

プログラム「drawT」を変更して、アルファベットの「A」をかくようにしてみよう。
下の図は「A」をかくための点の座標の一例だよ。



講　解答例プログラムは以下です。
RoboticsProfessorCourse2 > ArmRobot4 > drawA

チャレンジ課題

大きい四角の中に小さい四角が入るような二重の四角をかいてみよう。
巻末の方眼紙を下書きに使ってね。

講　解答例プログラムは以下です。
RoboticsProfessorCourse2 > ArmRobot4 > drawDsquare

3. まとめ（目安５分）

今回はアームロボットのセミオートマチック（半自動）とオートマチック（自動）のそれぞれの制御について勉強しました。プログラムにより、コントローラーの操作をスムーズにすることや、座標を使ってペンアームを巧みに制御する方法が少しわかってきましたか？　授業で使うアームロボットは、関節が動いてしまったり、モーターの動作の精度に限界があり、文字や記号が歪んでしまったかもしれません。しかし、アームロボットがどのような制御をしているかが理解できたと思います。次回はセンサーを使ってロボットらしく自動化して動かしてみましょう。
ではまた！

≪次回必要なもの≫

次回はアームロボットと、今回取り外したハンド部の他、以下のパーツを持ってきましょう。

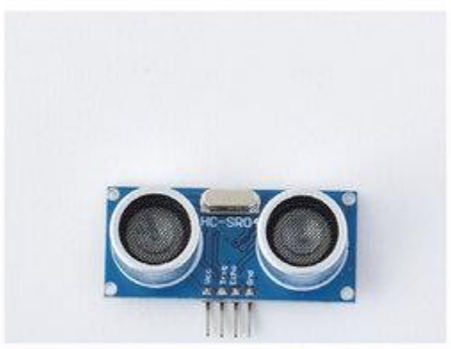
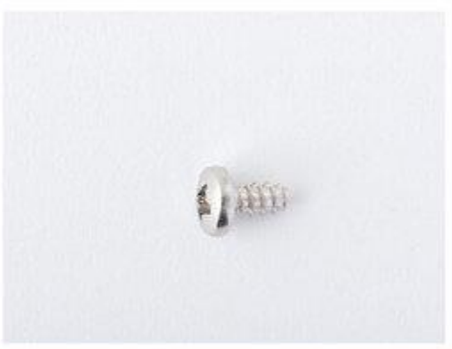
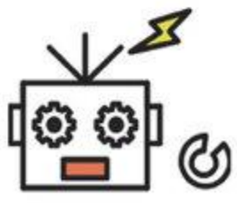
ラジオペンチ 1	ドライバー 1	USB ケーブル 1	センサー L 字ステイ 1
			
マトリクス LED シールド 1	超音波距離センサー 1	センサーカバー 1	M3L6 タッピングネジ (B) 2
			
M3 ナット 2	600mm センサーケーブル 1	AC アダプター 1	M3L12 ネジ 2
			

図 3-0　次回必要なもの

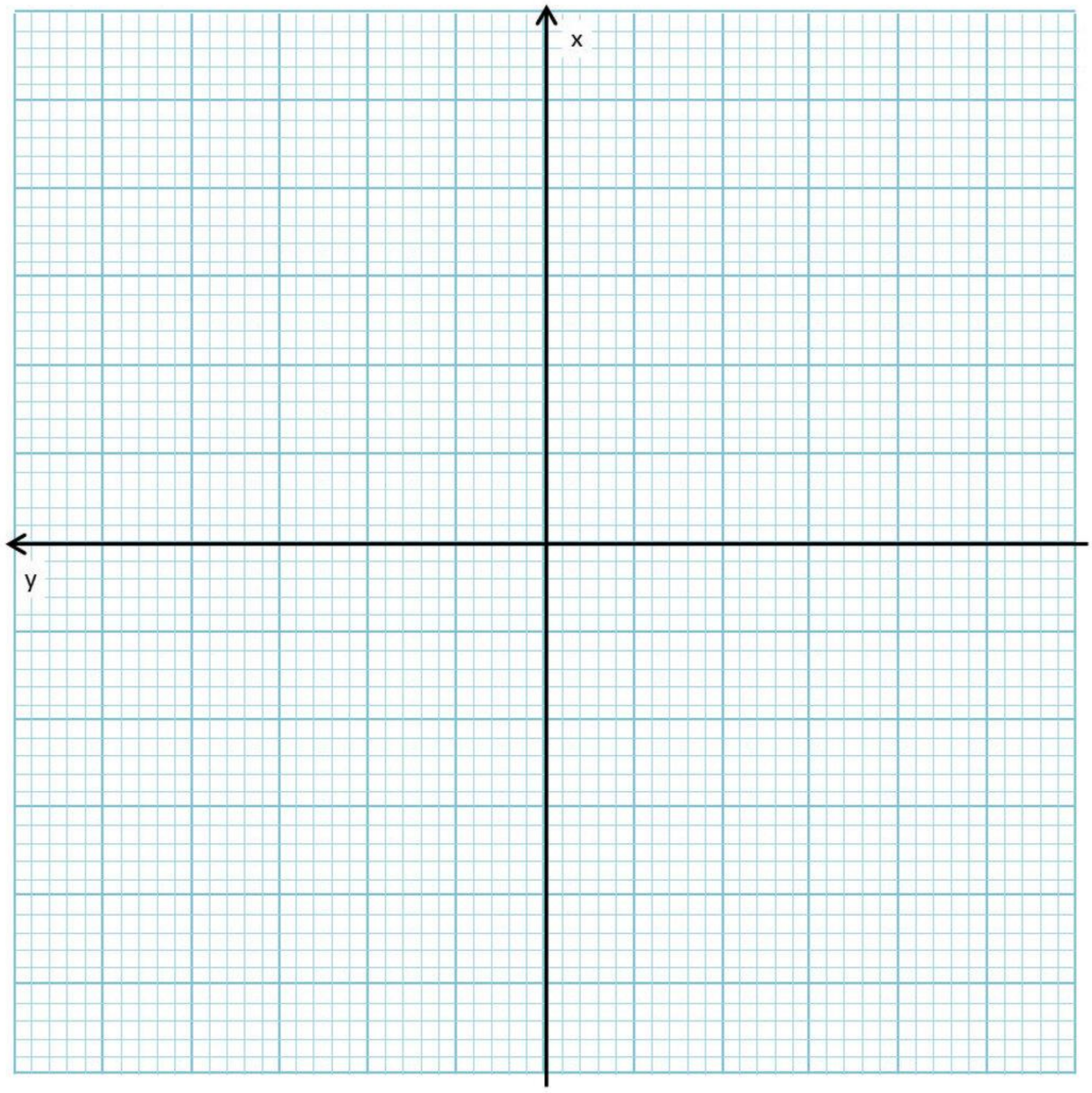


座標を使ったテクニックは、何度もトライすることでマスターすることができるよ！　がんばロー、オー！

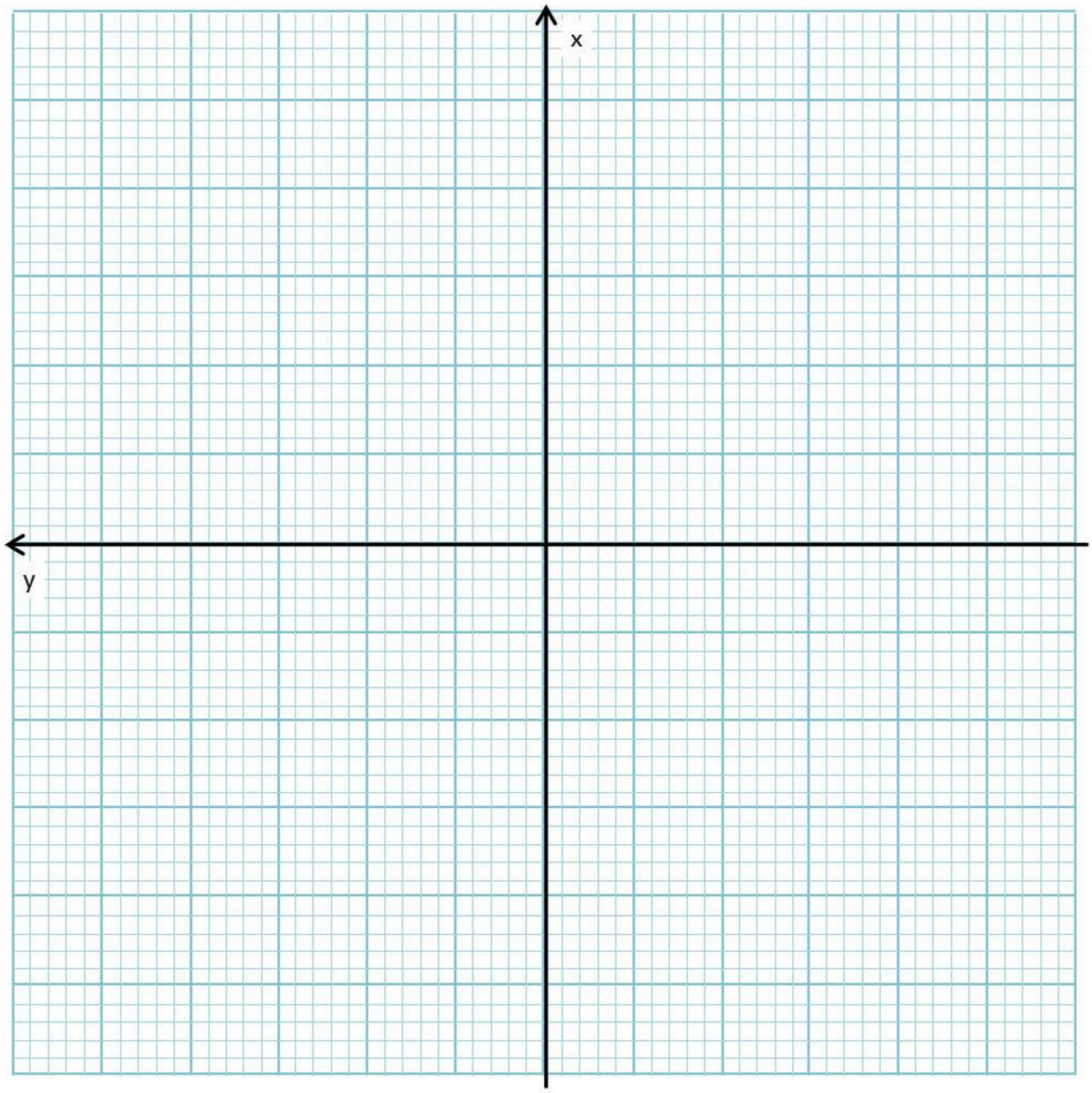
講

○　以下の理解度を確認します。
　・アームロボットを自動で制御する
　・ペンホルダーを付けて文字や記号をかく
　・座標の知識でアームロボットを動かす
○　次回のテーマは「センサーを使ったカシコイロボット」です。
　パーツを忘れずに持参するようにご指導ください。

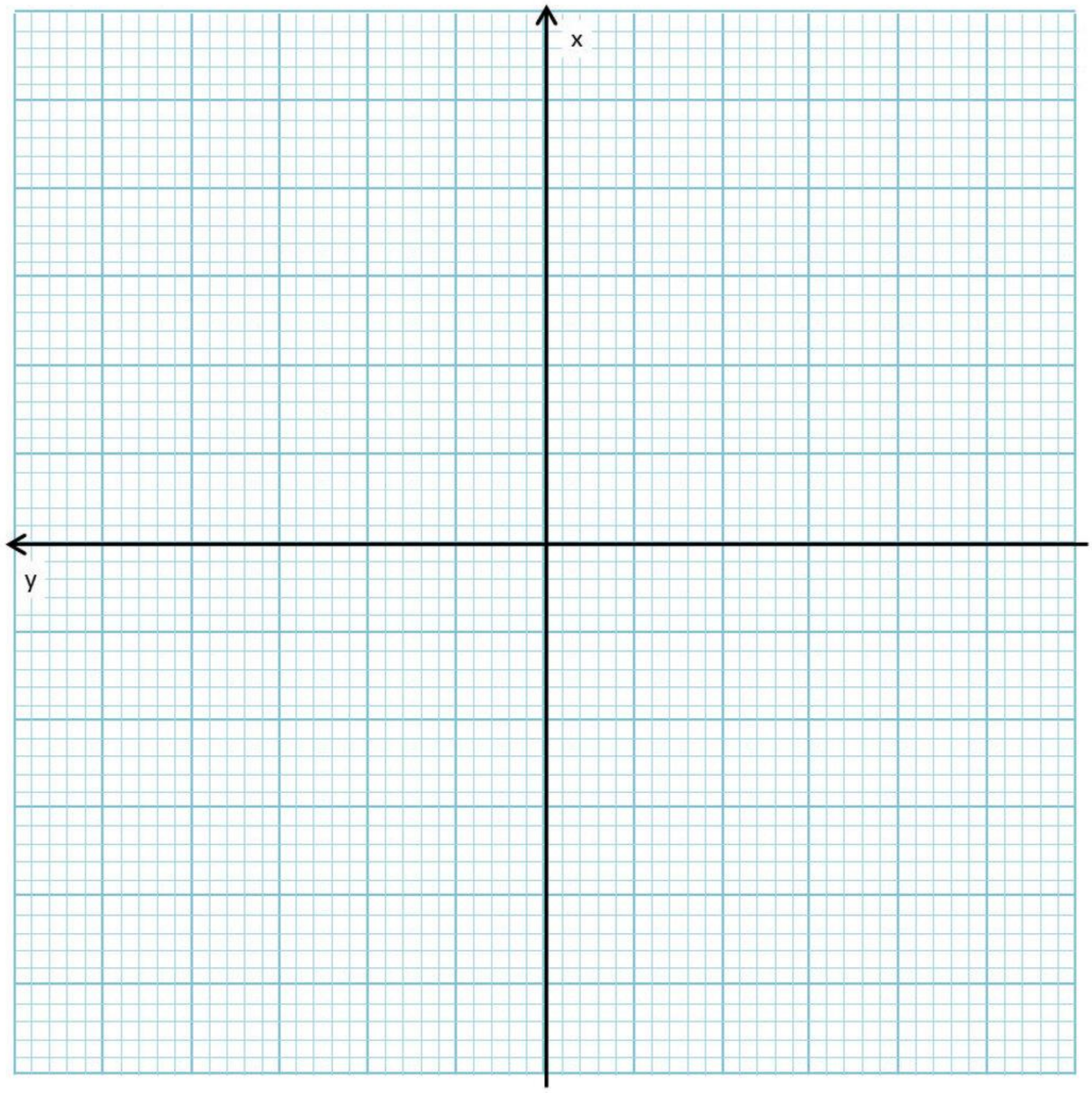
下書き用方眼紙①：スタート位置（ペンが下がる位置とxy軸の交点を合わせる）



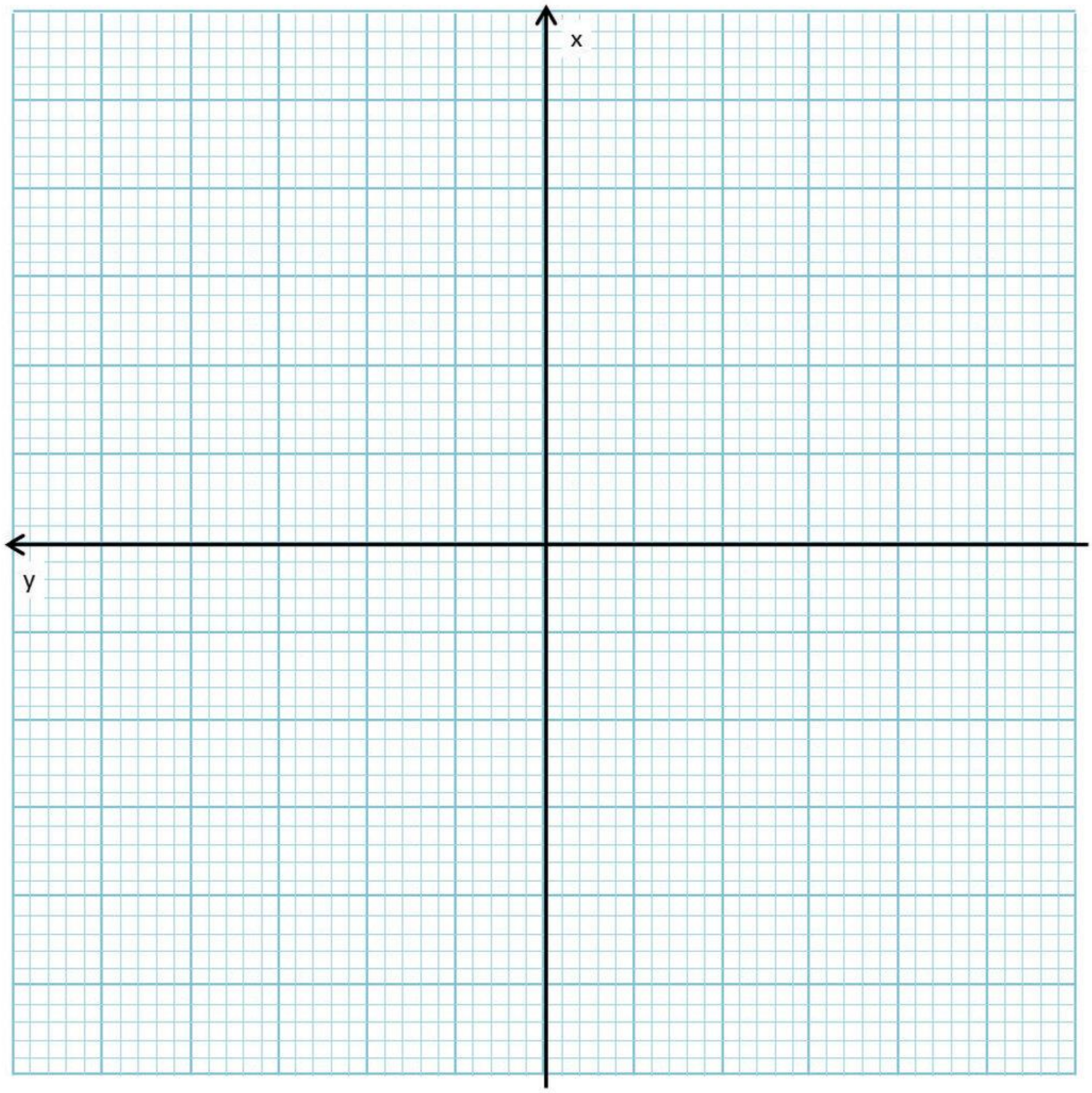
下書き用方眼紙②：スタート位置（ペンが下がる位置と xy 軸の交点を合わせる）



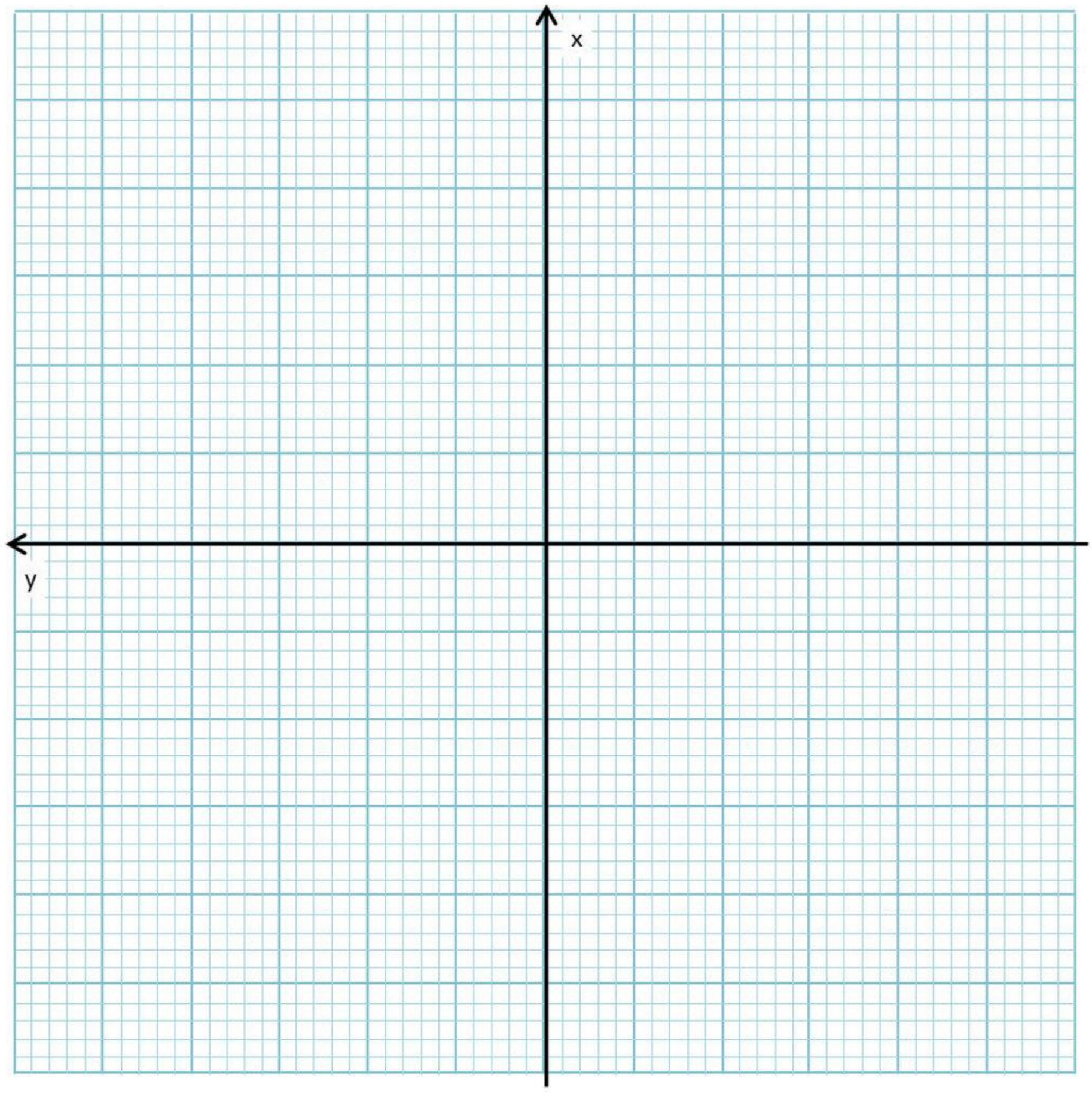
下書用き方眼紙③：スタート位置（ペンが下がる位置とxy軸の交点を合わせる）



下書き用方眼紙④：スタート位置（ペンが下がる位置と xy 軸の交点を合わせる）



下書き用方眼紙⑤：スタート位置（ペンが下がる位置とxy軸の交点を合わせる）



下書き用方眼紙⑥：スタート位置（ペンが下がる位置と xy 軸の交点を合わせる）