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

倒立振子ロボット②

（第3回/第4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3回授業日　2024年　月　日

第4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2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

倒立振子ロボット②

第3回

計測データの活用方法を考える

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。

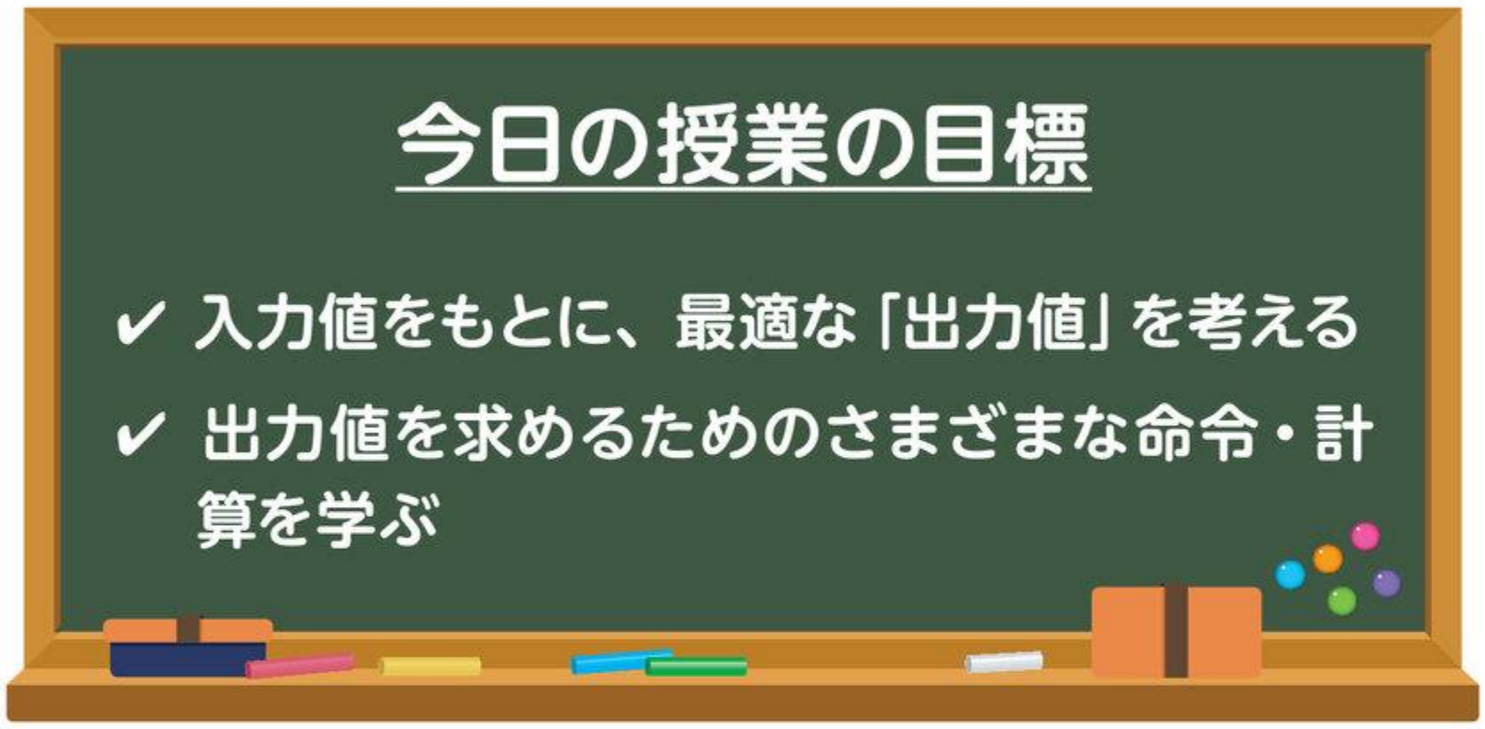
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（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. ロボットの「入力」と「出力」（目安 30 分）

0.0.「ロボットの「入力」と「出力」でやること



これまでの 2 回の授業で、「加速度」と「角速度」という 2 つの数値について学び、姿勢検出シールドでそれぞれを計測していましたね。今回も、この 2 つのセンサーを活用します。

さて、皆さんはこれまで「センサーで測った結果を使って、ロボットの動きを決める」という経験を何度もしてきました。
たとえば「〇〇センサーの値が××以上なら△△する」とか、「〇〇センサーの△△の値を、××パーツの命令に利用する」とかですね。

今回も「センサーの値を、他のパーツの命令に使う」のは一緒ですが、実際にプログラムをつくっていく中で「センサーの値をそのままでは使えない」という状況に直面します。
「こんなとき、どんな処理をつけたせばいいのか？どういった計算を取り入れればよいのか？」をじっくり学び、自分で考えるという経験を積んでみましょう。

0.1. 必要なもの

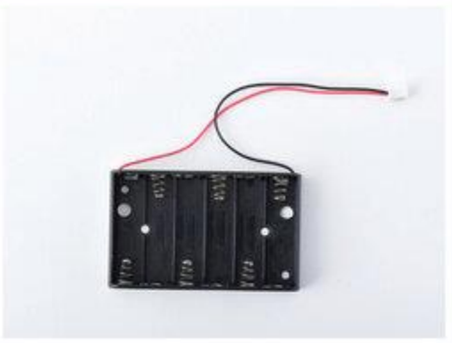
USB ケーブル	1	マイコンボード	1	ロボプロシールド	1	電池ボックス	3
							
スピーカー	1	姿勢検出シールド	3				
							

図 0-0　必要なもの

前回まで使っていたユニットから、「マトリクス LED シールド」と「7 セグメント LED」を取り外したものを使います。

0.2. 動作確認

今回もまず、姿勢検出シールドが正常に動作しているか確認しておきましょう。

プログラムの書き込み

RoboticsProfessorCourse2 > Inverted3 > MPUTest_6axis

プログラムの書き込みが終わったらシリアルモニターを開き、通信速度を「115200baud」に設定します。

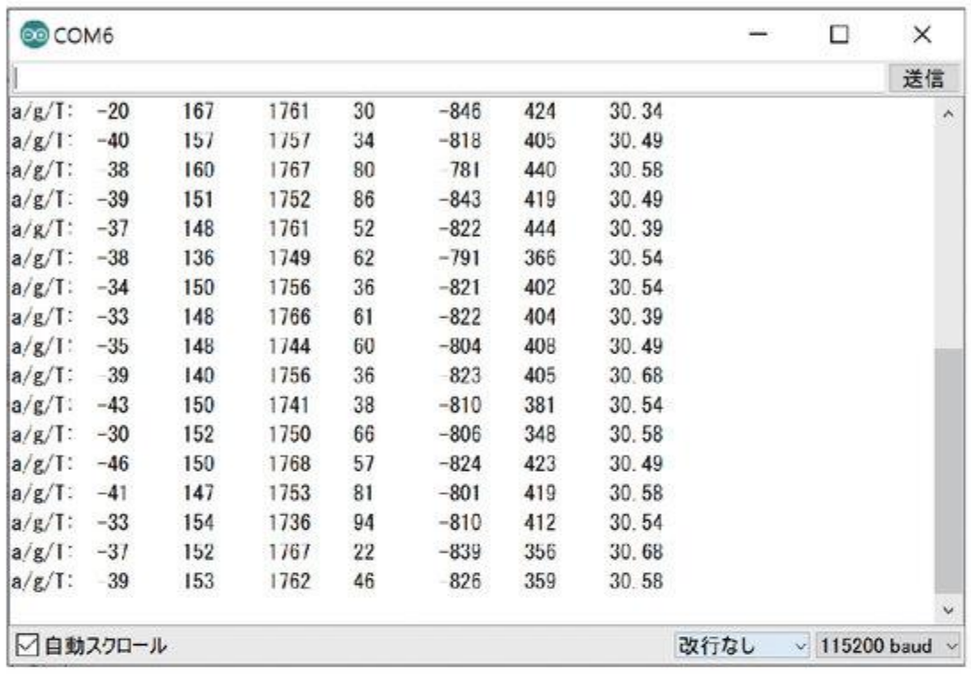


図 0-1　シリアルモニターの表示

数字が絶え間なく更新されていますね。
姿勢検出シールドをゆっくり動かしたり回したりして、1～6 列目の数字が変化することを確かめます。
もし以下のような表示になったら、一旦マイコンボードを PC から取り外し、姿勢検出シールドの接続部分を確認します。

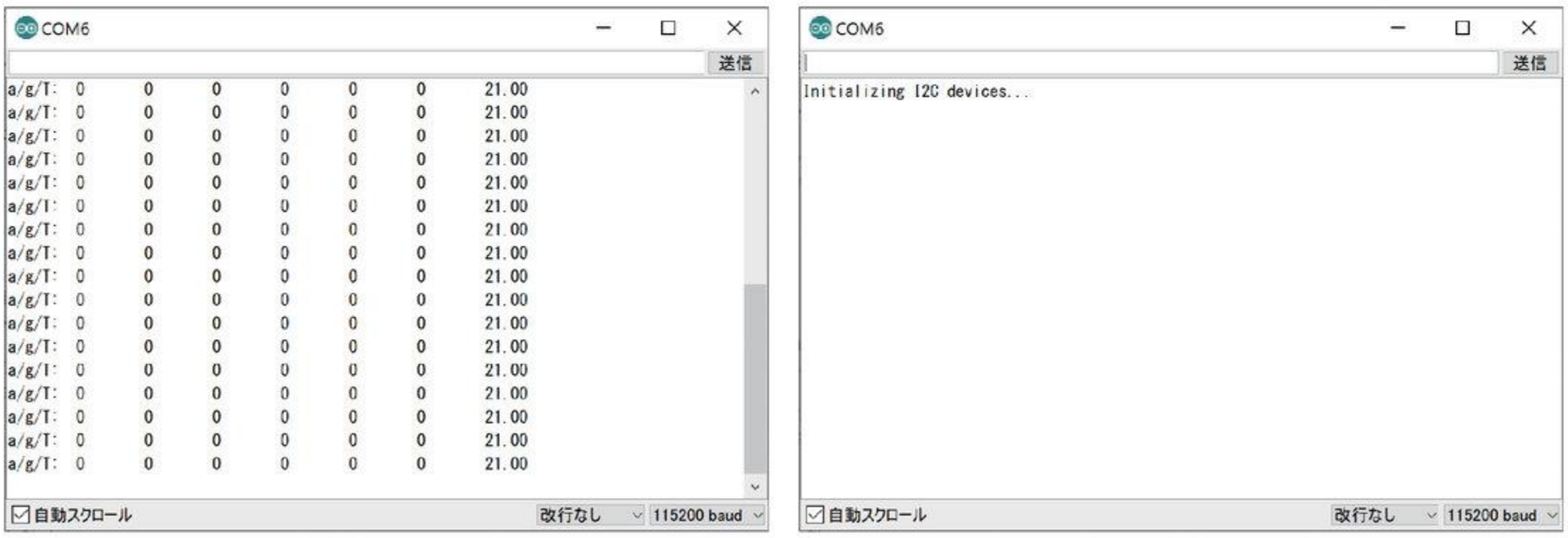


図 0-2　通信に失敗している例

⚠注意！

姿勢検出（しせいけんしゅつ）シールドは非常に細かい信号をやりとりするため、ちょっとした接続のぐあいで通信が途切れたり復活したりします。

姿勢検出（しせいけんしゅつ）シールドのピンをきちんと奥まで差し込（こ）んだり、反対にほんの少しだけ浮かしてみたりして、うまく動作するように調整しましょう。

講

今回は計測値のうち、左の３つの値（ax ～ az）を使用します。シールドが急加速・急減速したときに値が大きく変化する状態が正常ですが、値がうまく変化しない、そもそも計測がはじまらないなどの場合は「注意」内の記載通り接続の調整を行なってください。姿勢検出シールドを奥までしっかり差し切るより、ほんの少し（～ 0.5 ミリ程度）浮かせた方が接続がよくなる場合などもあります。

0.3. 入力値の範囲（はんい）をつかむ

シリアルモニターに表示される値のうち、注目すべきは「加速度」「角速度」の値です。

図 0-3　各列の値が表すもの

プログラムを見てみると、この 6 つの計測値はそれぞれ ax ay az gx gy gz という変数に入れられていることがわかります。
6 つの変数に計測値を記録したあと、シリアルモニターに変数の値を表示するという命令になっていますね。

プログラム「MPUTest_6axis」より抜粋（ばっすい）

```
Serial.print(ax); Serial.print("\t");
Serial.print(ay); Serial.print("\t");
Serial.print(az); Serial.print("\t");
Serial.print(gx); Serial.print("\t");
Serial.print(gy); Serial.print("\t");
Serial.print(gz); Serial.print("\t");
```

プログラミングに入る前に、まずこの６つのセンサーの値がいくつくらいの範囲になるのか、メモしておきましょう。

やってみよう！

プログラム「MPUTest_6axis」を実行しながら姿勢検出シールドを水平なところに置いて、ax ～ gz の６つの数がいくつくらいになるかを確認しよう。
終わったら、今度は姿勢検出シールドを揺らしたり回したりして「いくつからいくつまでの間に収まるか」も確認しよう。
結果は下の記入欄にメモしておいてね。

※「大まかな値」がわかればよいので、500 刻み、または 1,000 刻みくらいのキリのいい数でＯＫだよ！

	静止時の値	値の範囲
ax	くらい	最小　～最大　の間
ay	くらい	最小　～最大　の間
az	くらい	最小　～最大　の間
gx	くらい	最小　～最大　の間
gy	くらい	最小　～最大　の間
gz	くらい	最小　～最大　の間

0.4. 「入力」と「出力」

メモはできましたか？

今記録した値は、加速度センサーや角速度センサーが計測した値、つまり「ロボットに取り込まれた数値」ですね。
このように、ロボットがセンサーなどを使って情報を取り入れることを「入力」といいます（取り入れた情報や数値そのものを『入力』とよぶこともあります）。

反対に、ロボットがモーターを回転させたり、スピーカーから音を出したり、マトリクス LED を光らせたりするとき、コンピューター（マイコン）から信号を出してそれぞれのパーツを動作させています。
このように、コンピューター（マイコン）が他のパーツに情報を送ることを「出力」といいます。

これまでも、さまざまな入力・出力を行うロボットをつくってきましたね。

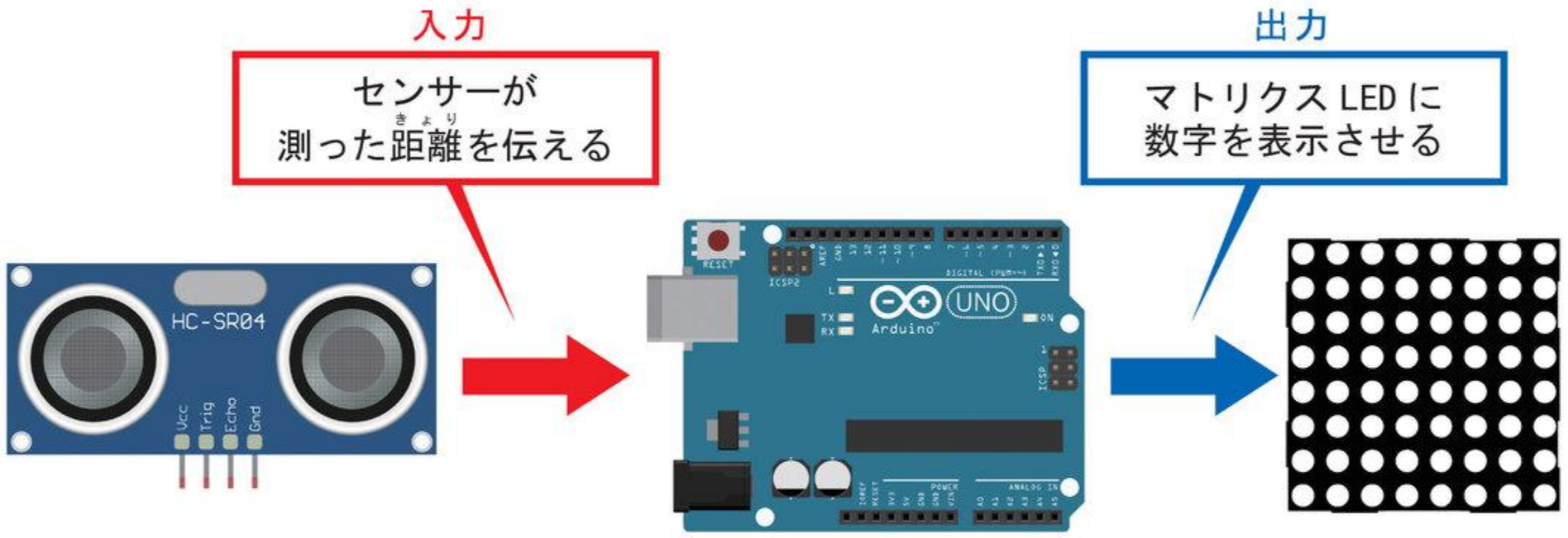


図 0-4　入力と出力の例

豆知識

入力と出力をあわせて「入出力」や、英単語（input と output）の頭文字を使って「I/O」などとよぶこともあります。

今回は、「入力→出力」の流れを皆さん自身で考え、つくっていくことにチャレンジしてみましょう。
入力値の情報は集まったので、これを少しずつ出力用の情報に直していくことになりますね。

１．加速度センサーの「入力」を使う（目安 50 分）

1.0.　スピーカーの動作確認

今回は、入力に姿勢検出（しせいけんしゅつ）シールド上のセンサーを使いますが、出力には「スピーカー」を使います。
動きや向きを変えると、鳴る音が変わるようにするということですね。

まずは、スピーカーが鳴るプログラムを書き込（こ）んでスピーカーの動作確認とともに、スピーカーに出力する命令を見ておきましょう。

プログラムの書き込み
RoboticsProfessorCourse2 > Inverted3 > Tone_base

実行結果：スピーカーから同じ音がくり返し鳴り続ける

聞いただけで何の音かわかった人は凄いですね！
これは「ラ」の音を鳴らし続けるプログラムです。

講
スピーカー以外に想定されていないパーツ（モーターやマトリクス LED など）が接続されている場合、音がうまくならない場合があります。
必ずテキストの記載通りのパーツ構成の状態でプログラムを実行してください。

プログラムの中身は、このようなつくりになっています。

プログラム「MPUTest_6axis」より抜粋（ばっすい）

```
#include <Tone.h>
#include <RPlib.h>

Tone spk;
```

① 前置きの設定

```
void setup(){
  spk.begin(D2);
}
```

② setup の命令

```
void loop(){
  spk.play(440,500);
  delay(50);
  spk.stop();
  delay(500);
}
```

③ loop の命令

① 前置きの設定

#include	各パーツに出せる命令がまとめられた設定ファイル（ライブラリ）を読み込むための構文です。「Tone.h」「RPlib.h」は、スピーカー用の命令が含まれているライブラリです。
Tone spk;	スピーカーの「名前」を設定するための命令です。以降、spk.〇〇 とかかれた命令は「スピーカーに出している命令」ということになります。

② setup の命令

spk.begin(D2);	ロボプロシールドの D2 に接続したパーツを、スピーカーとして設定するという命令です。

③ loop の命令

spk.play(440,500);	スピーカーから音を出すための命令です。 カッコ内の１つ目の数字が周波数（Hz）です。ここで音の高さを決められます。440Hz は「ラ」の音を表します。
spk.stop();	スピーカーの音を止める命令です。 ギアドモーターなどと同じく、一旦この命令を入れておかないと音が途切れることなく鳴り続けます。

1.1.　加速度と周波数を連動させる

１）音を鳴らす機能の追加

このプログラムを、「姿勢検出シールドの動きに合わせて音の高さが変わる」ように改造してみましょう。

たとえば、音の高さが x 方向の加速度に合わせて変化するようにしてみましょう。
姿勢検出シールドを図の方向に勢いよく動かせば、一瞬だけ音が高くなるようなプログラムです。

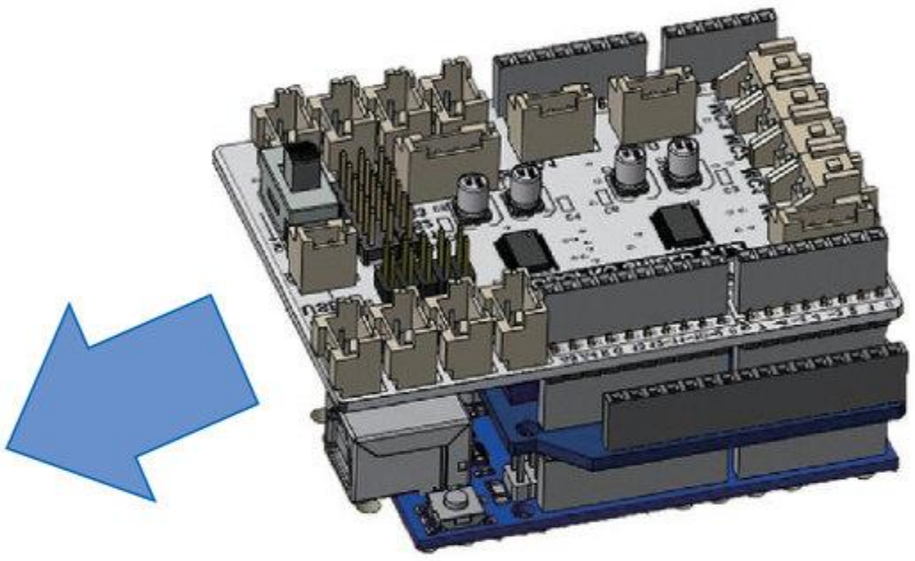


図 1-0　x 方向に加速させる

先ほど使った「MPUTest_6axis」では、x 方向の加速度が ax という変数に記録されていましたね。これを周波数に使えばよさそうです。
まずは、「MPUTest_6axis」でも「Tone_base」のようにスピーカーを使えるようにしましょう。２つのプログラムを合体させていきます。

やってみよう！

プログラム「MPUTest_6axis」に「Tone_base」の命令を移植して、スピーカーから音を鳴らす機能を追加しよう！

ヒント
「MPUTest_6axis」も、よく見れば① 前置き　② setup　③ loop　の３つに分かれているね。
それぞれの部分に、「Tone_base」の命令をうつしていこう。

講
解答例は以下のプログラムです。
RoboticsProfessorCourse2 > Inverted3 > MPU_Tone_base
「TONE_base」から移植した行には、その旨コメントがついています。

2）入力と出力の連動

「姿勢検出シールドで加速度をはかりつつ、スピーカーから音も出る」というプログラムになりましたね。
あとは、加速度を音の周波数に連動させるだけです。

```
spk.play(440,500);
delay(50);
```

▼

```
spk.play(ax,500);
delay(50);
```

こんな感じでよさそうですね。
実際にプログラムをかき換え、実行してみましょう。

…どうでしょうか。ねらい通りに音が鳴りましたか？
「動かす前からかなり高い音が鳴ってしまう」
「そもそも、まったく音が鳴らない」
「動かしても音の高さが変わらない」
「ちょっと動かすだけで一気に音が高くなってしまう」
など人によって差があると思いますが、想像していたようには音が出なかったのではないでしょうか。
周波数を加速度の値におきかえただけなのに、なぜ動作がおかしくなってしまったのでしょう。

ここで、下の表をチェックしてみましょう

表 1-0　音階と周波数

音階	1 オクターブ下のラ	…	ド	レ	ミ	ファ	ソ	ラ	シ	ド	…	1 オクターブ上のラ
周波数	220	…	262	294	330	349	392	440	494	523	…	880

※ play 命令では小数が使えないため、表の中の値は小数点以下を四捨五入したものです。

やってみよう！

はじめにやっておいた「値の範囲のチェック」や上の表を参考に、play 命令に ax をそのまま使うとうまくいかなかったのはなぜか考え、下の欄に文章でまとめよう。

ax の値はマイナスになってしまうことがあるから

加速度の値が、周波数に使うには大きすぎるから

3）入力した値を調整するための準備

まとめられましたか？

ax をそのまま周波数に使うと、このような問題がありますね。

問題① ax の値はマイナスになることがある
問題② ax の値は数千まで上がるため、周波数にするには範囲が大きすぎる

つまり「ax の値がマイナスにならないようにする工夫」「ax の値を周波数としてちょうどいい範囲に収める工夫」が必要ということになります。

この２つは、どちらもプログラムを改良することで実現できます。
まずは、周波数の部分に ax を直接入れなくても済むように、「周波数を入れておくための変数」をつくっておきましょう。下の赤字の部分を追加・変更します。

```
int freq;

void loop(){

    (中略)

    spk.play(freq,500);
    delay(50);
    spk.stop();
    delay(500);
}
```

freq という変数をつくり、この変数を周波数として使っています。
とはいえ、まだ freq には何の値も入っていないので、このままではねらい通りの音は鳴りません。
音を鳴らす命令より前に「freq の値を決める作業」が必要ですね。

豆知識

freq という変数名は、「周波数」という意味の英単語「frequency」をもとにしています。

1.2. 加速度の値を「変換」する

では、freq の値を決める作業をしていきましょう。

まずは、「 ax の値がマイナスにならないようにする工夫」からです。

下の赤字の部分を追加しましょう。

```
int freq;

void loop(){

    (中略)

    freq = abs(ax);
    spk.play(freq,500);
    delay(50);
    spk.stop();
    delay(500);
}
```

命　　令「abs」
実行結果：かっこの中の数を絶対値に変換する
使 い 方：a = abs(x); // 変数 x の値の絶対値を、変数 a に入れる

「絶対値」とは、「その数の 0 からの距離（きょり）」を表す数です。
たとえば「0 より 3 大きい数」は +3 ですし、「0 より 3 小さい数」は -3 です。でも、+3 も -3 も「0 から 3 だけ離れた数」なのは同じですよね。よって「+3 の絶対値」は 3、「-3 の絶対値」も 3 になります。
このように「絶対値」を使うことで、もし ax がマイナスの値だった場合も、むりやりマイナスのない数に変換することができます。

これで、freq にはプラスの数（または 0）しか入らなくなりました。一歩前進です。

講 「絶対値」は中 1 数学の学習内容です。生徒がまだ学んでおらずピンと来ていない場合などは「とにかく abs を使うとマイナスを消せる」という部分だけでも理解させてあげてください。

次に、ax の値の範囲を、周波数にちょうどいい範囲に直しましょう。
たとえば、下の図のような場合を考えます。

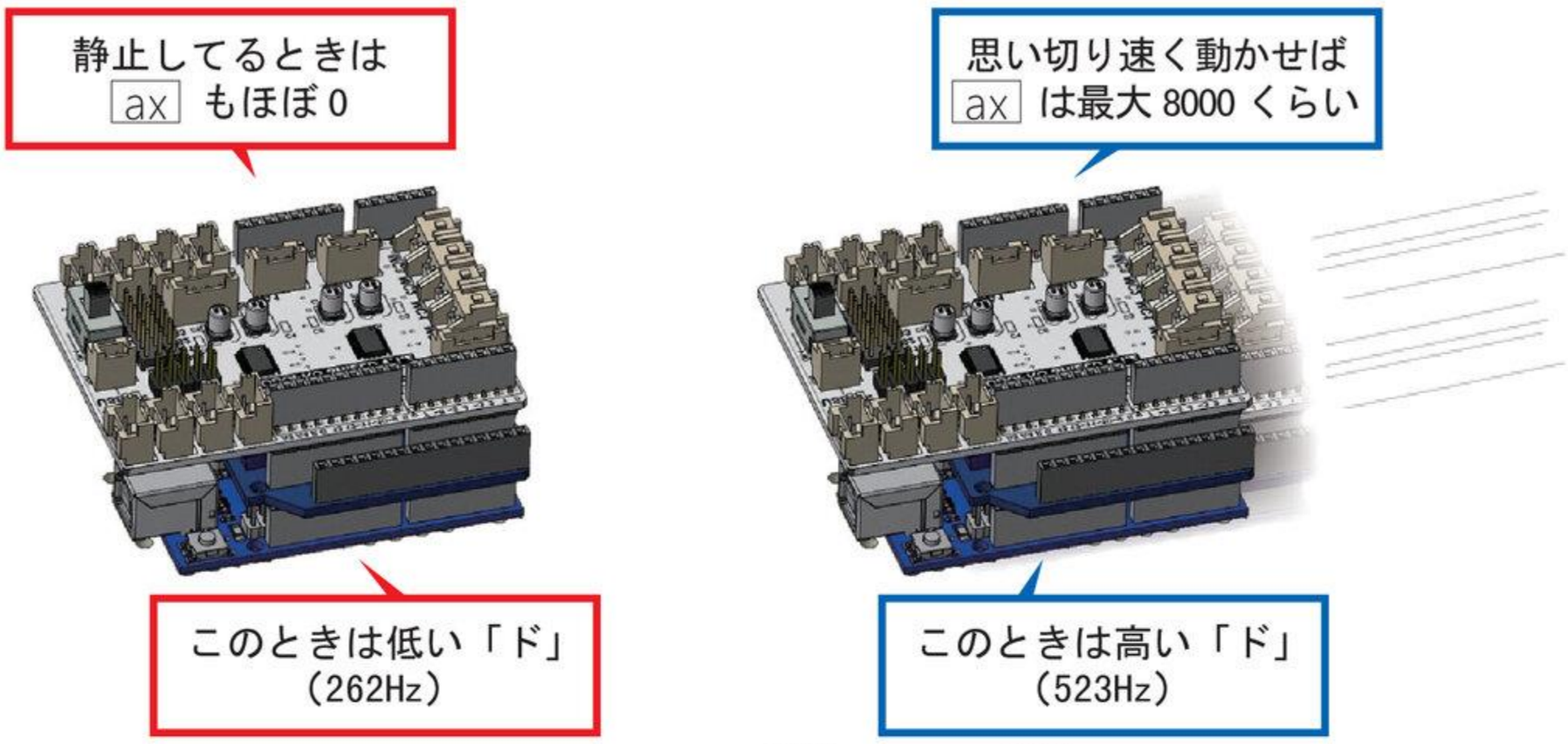


図 1-1　加速度の範囲と周波数の範囲

この場合は、下のようなプログラムになります。

```
freq = abs(ax);
freq = map(freq, 0, 8000, 262, 523);
spk.play(freq,500);
delay(50);
spk.stop();
delay(500);
```

命　　令「map」
実行結果：変数の値の範囲を、指定した範囲に変換する
使 い 方：x = map(a, 0, 200, 0, 50);
//0 ～ 200 の範囲の値をとる変数 a を、0 ～ 50 の範囲に収めて変数 x に入れる

このように値の範囲が変換されます。周波数として丁度いい数になりました。

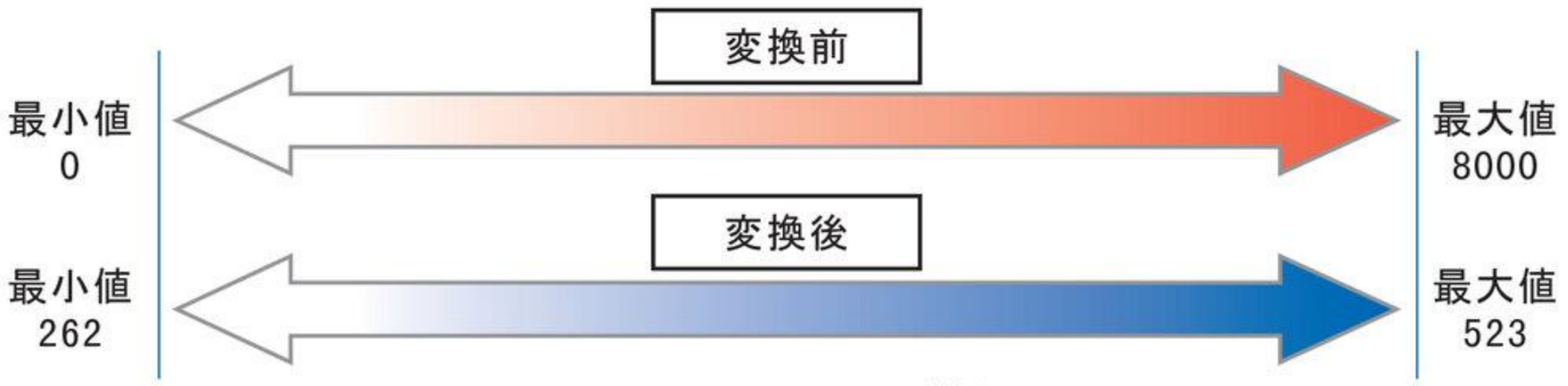


図 1-2　変換前後の値の範囲

ただこのプログラムの場合、「全力で加速させたときの ax の値」が最高値になるので、音を高くしていくのはなかなか大変です。
「そこまで本気でなくても出せる加速度」を最大値にできれば、ラクに高い音を出すことができますね。

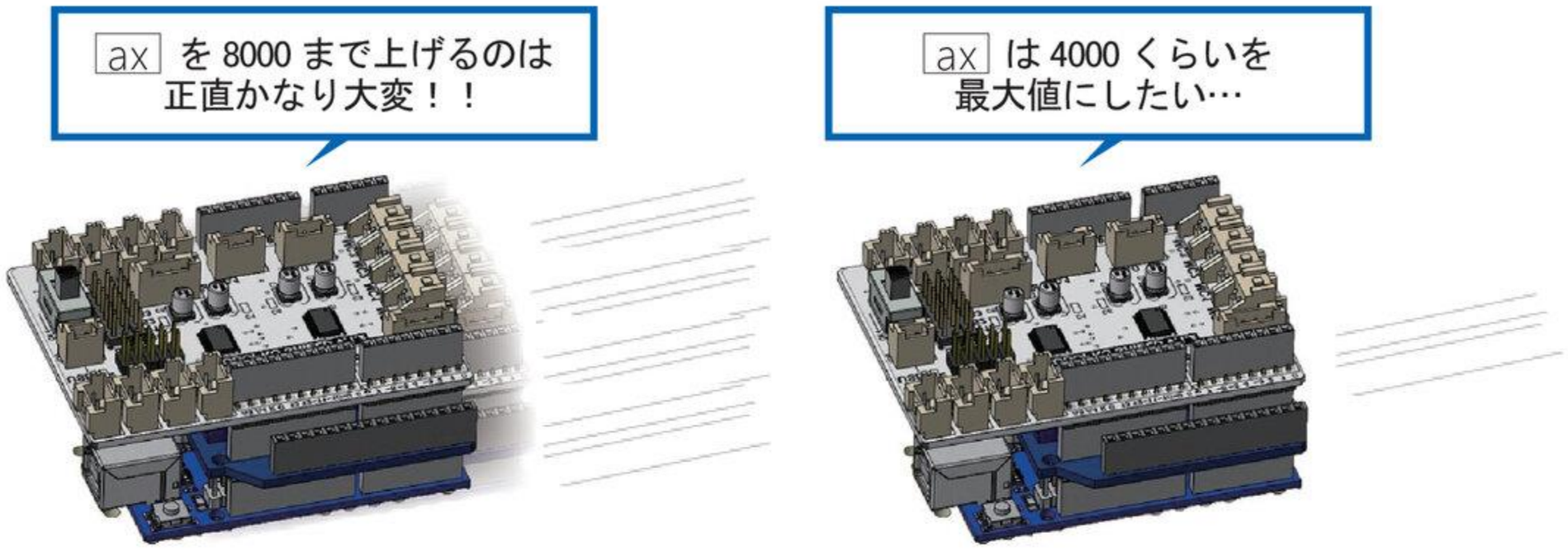


図 1-3　加速度に「上限」を設定したい

そんなときは、この命令が役立ちます。追加してみましょう。

```
freq = abs(ax);
freq = constrain(freq, 0, 4000);
freq = map(freq, 0, 8000, 262, 523);
spk.play(freq,500);
delay(50);
spk.stop();
delay(500);
```

命　　令「constrain」
実行結果：変数の最大値と最小値を設定する
使 い 方：a = constrain(a, 0, 100);
// 変数 a の値が 0 未満になったら 0 に、100 より大きくなったら 100 に直す

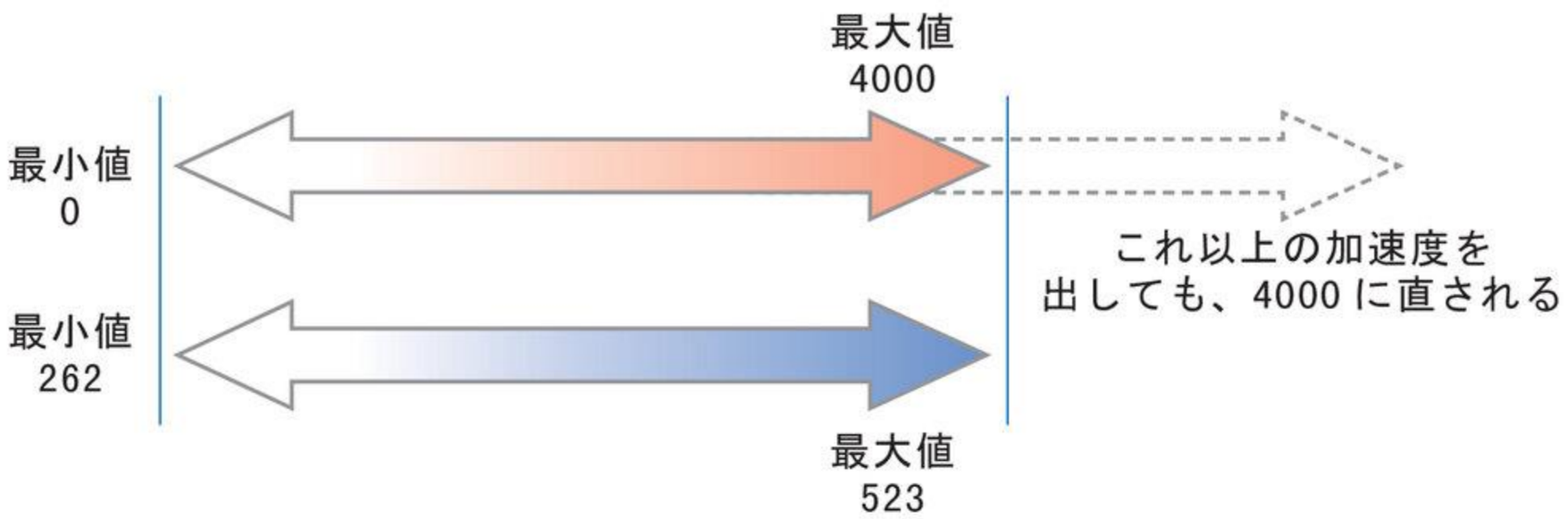


図 1-4　変換前後の値の範囲

やってみよう！

図 1-4 のように値の範囲を設定したい場合、プログラムの中にはまだ修正すべき所が１か所だけ残っているよ！どこか探して、修正してみよう！

講

解答例は以下のプログラムです
RoboticsProfessorCourse2>Inverted3>MPU_Tone_freq
constrain 命令で上限を 4000 に引き下げたため、map 命令で指定する値の範囲も 0 ～ 4000 に修正します。

2. 三次元の加速度を計算で求める（目安 25 分）

2.0. 感知している加速度の向きを確認する

スピーカーの音が加速度に連動するようにはなりましたね。でも、周波数を決めるのに使ったのはあくまで ax だけです。つまり、姿勢検出（しせいけんしゅつ）シールドをどの向きに動かしても、音の高さを決めるときは x 方向の加速度しか感知してくれないことになります。

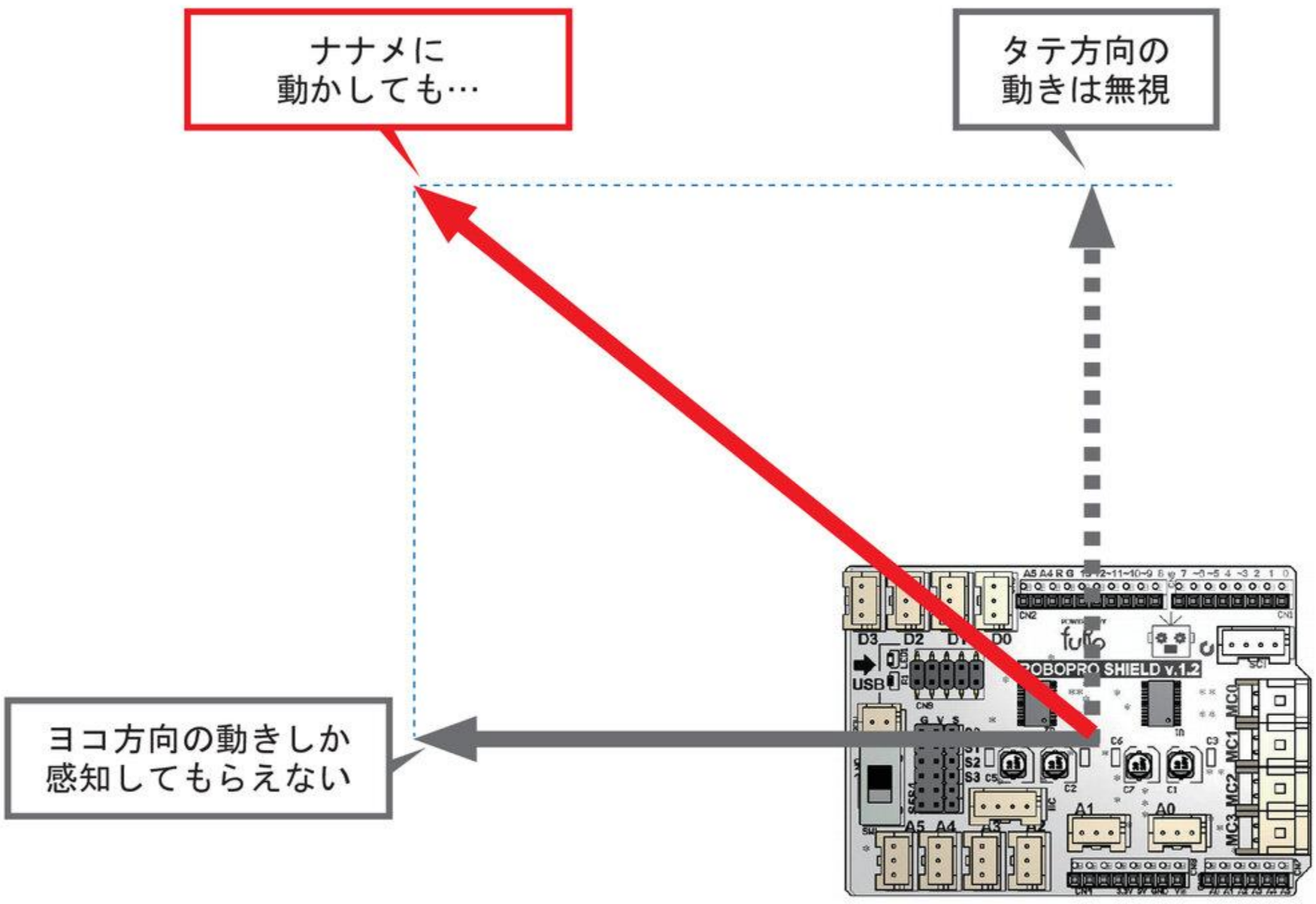


図 2-0　1つの方向の加速度しか検出されていない

姿勢検出（しせいけんしゅつ）シールドをキッチリ x 方向にだけ動かすのは難しいですし、ナナメに動かしたらナナメ向きの加速度を使って音を変えてほしいですよね。

そこで「y、z 方向の加速度も使ってナナメ方向の加速度を計算する」という命令を追加し、プログラムを改良していきましょう。

まず、姿勢検出シールドを使って「はかれるもの」と「はかれないもの」を整理しておきましょう。

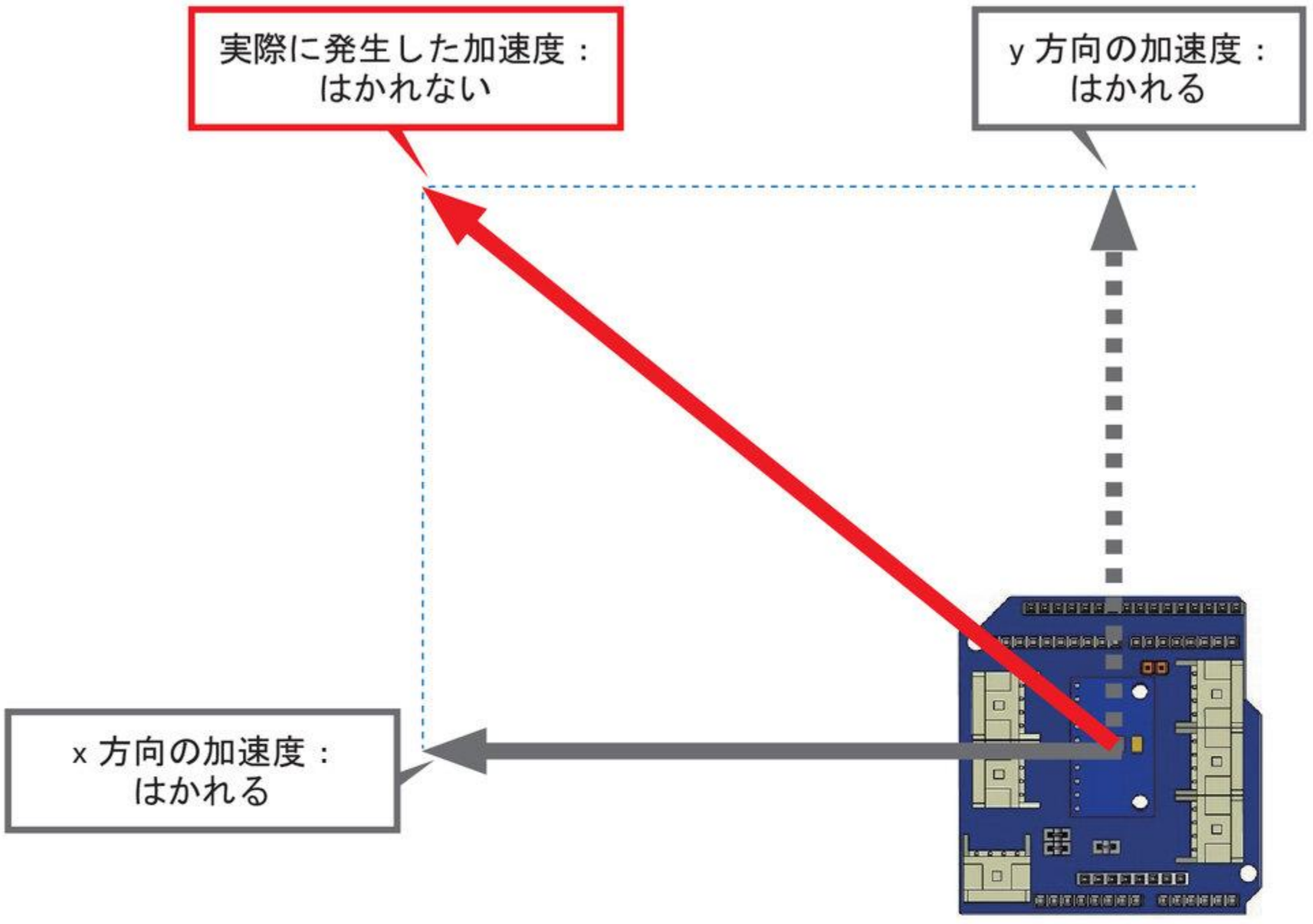


図 2-1　はかれる向きの加速度と、はかれない向きの加速度

このように、x 方向や y 方向（実際には、さらに z 方向も）の加速度であれば、姿勢検出シールド上の加速度センサーではかる事ができますね。
一方、シールドをナナメに動かした場合、その向きの加速度を直接はかる事はできません。
でも、ナナメに動かしたのであれば、ナナメ向きの加速度こそが「実際に生じた加速度」に最も近いはずですよね。どうにかして求められないでしょうか。

ここで、数学の知識を活用しましょう。

2.1. 三平方の定理

突然ですが、下のような直角三角形を見てみましょう。

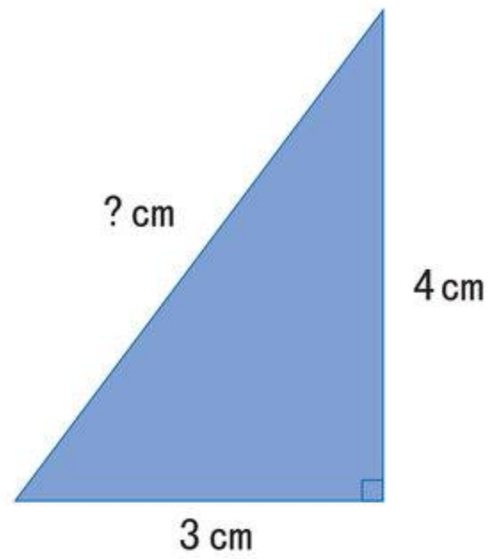
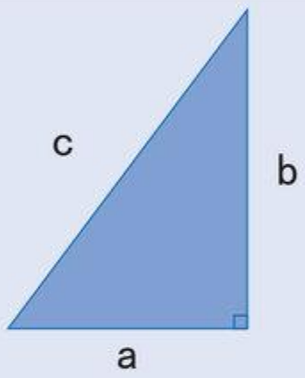


図 2-2　直角三角形

3 本ある辺のうち、タテ向きの辺とヨコ向きの辺は長さがわかっています。
では、ナナメ向きの辺は何 cm でしょうか？
…小学校の算数であれば、「実際のサイズで直角三角形を描いて定規ではかる」という求め方をしていたかもしれません。ただ、実は直角三角形であれば計算で求めることもできます。そのためには、直角三角形の以下のような性質を使います。



直角三角形の 3 辺の長さが左の図のように a,b,c なら、以下の式が成り立つ。

$$a^2 + b^2 = c^2$$

（右上の小さな数字は「同じ数どうしをその回数だけかけ算する」という記号。「a^2」は「a × a」と同じ）

上の三角形で考えてみると、$a^2 + b^2$ の部分は 3 × 3 ＋ 4 × 4 ですね。計算すると 25 になります。これが c^2 、つまり？×？と同じ数になるようです。同じ数を 2 回かけて、答えが 25 になるのは「?」がいくつの時でしょうか。

1 × 1 = 1 , 2 × 2 = 4 , 3 × 3 = 9 … 5 × 5 = 25

そう、5 のときですね。ということで、ナナメ向きの辺の長さは 5cm であることがわかります。このように、直角三角形は「3 本中 2 本の辺の長さがわかれば、残り 1 本は計算で長さを求められる」という性質があります。

豆知識

a^2 のように同じ数を 2 回かける事を「2 乗」や「平方」といいます。上の式では 3 つの辺の「平方」を使いましたよね。
この直角三角形の性質のことを「三平方の定理」といいます。

講　平方根が中 3 数学の内容で、生徒が学んでいない可能性が高いためなるべく数学用語を使わずに説明しています。

なぜいきなり「三平方の定理」の話をしたかというと、前のページで確認したこの図が関係しています。

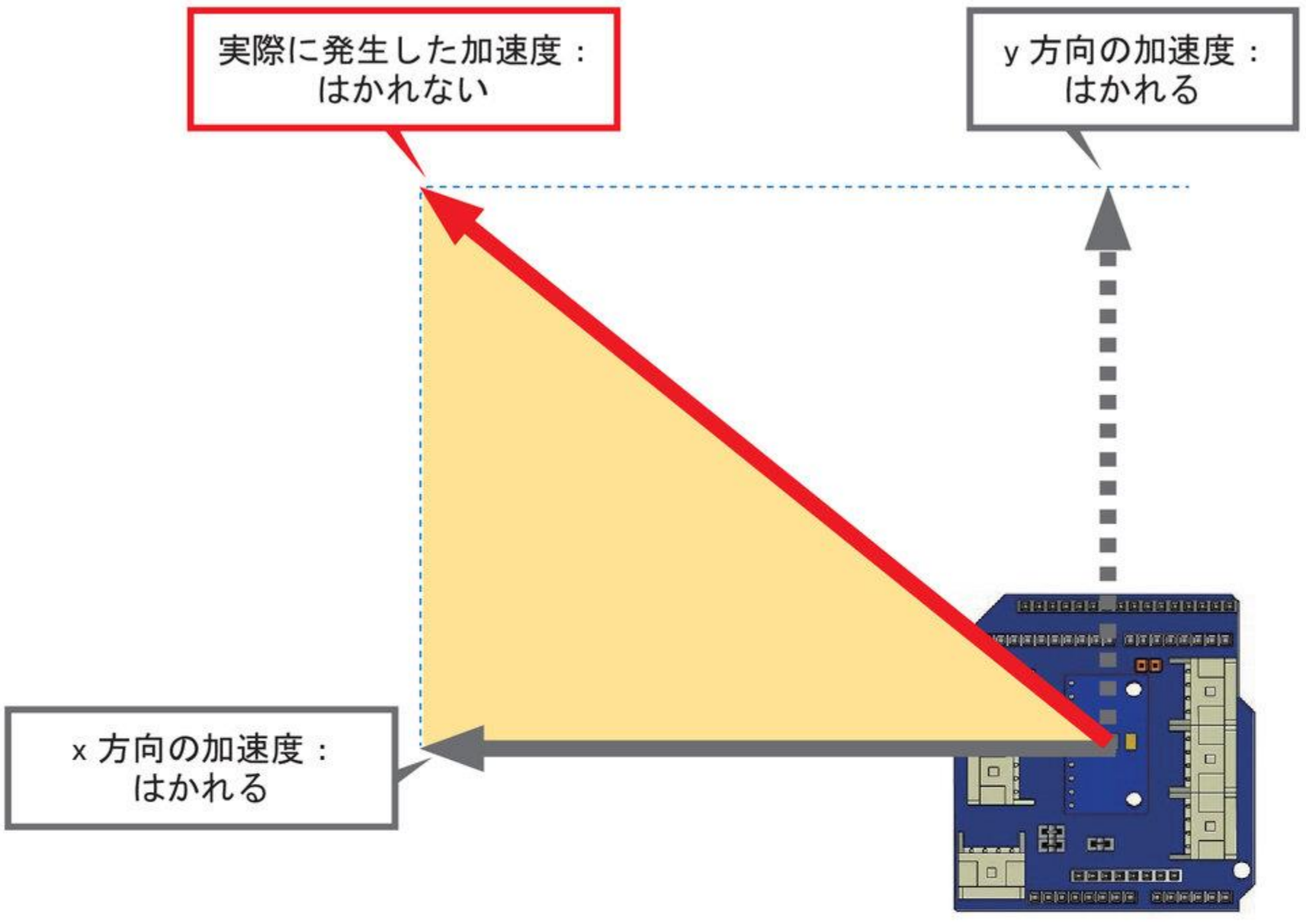
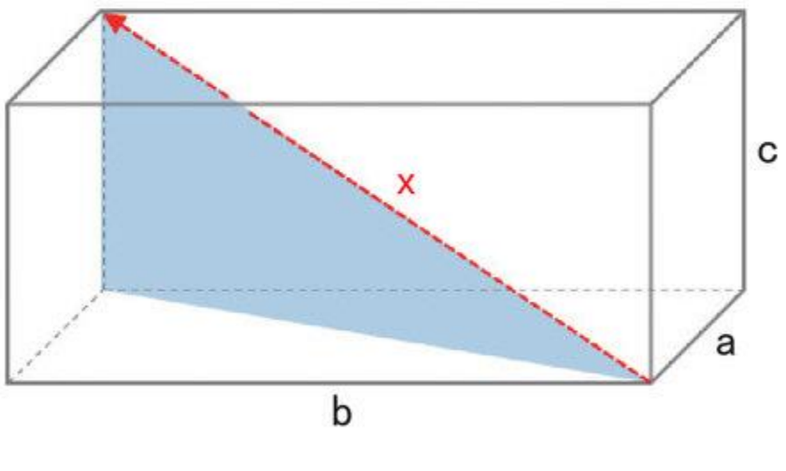


図 2-1　はかれる向きの加速度と、はかれない向きの加速度（再掲）

x 方向の矢印と、y 方向の矢印は直角に交わっています。
ということは、「実際に発生した加速度」の赤い矢印と合わせて、黄色く塗られた部分は直角三角形になっているはずです。つまり、ここで三平方の定理を使えるわけですね。「x 方向の加速度」「y 方向の加速度」は計測していますから、つまり 3 つの辺のうち 2 つの長さ（加速度の値）はわかっているという事です。ここから残り 1 つの辺、つまりナナメ向きの「実際に発生した加速度」を求めることもできるわけですね！



ちなみに、ここに高さ c が加わって直方体になっても、対角線の長さ x は

$$a^2 + b^2 + c^2 = x^2$$

で求めることができます。
これで z 方向の加速度 az まで含めて計算することができますね。

講　ax と ay と az はすべて垂直になっているので、三平方の定理を使う事で実際にシールドを動かしている向き(ナナメ)の加速度を算出できます。そのための説明ページです。

ステップアップ

ax ay az の 3 方向の加速度を元に三平方の定理を使い、ナナメ方向の加速度を算出して変数 freq とするプログラムにしてみよう！

ヒント

map 命令で周波数に直すより前に、三平方の定理を使う命令を入れればいいんだね！以下の 2 つの命令をうまくつかって、三平方の定理の式になるようにしよう！

命　　令「pow」

実行結果：ある数を、指定した回数だけかけ算する

使 い 方：x = pow(a, 2);

// 変数 a を 2 回かけ算して、変数 x に入れる

命　　令「sqrt」

実行結果：ある数の「2 乗する前の数」を求める

使 い 方：x = sqrt(a);

// 変数 a の「2 乗する前の数」を求め、変数 x に入れる

講

RoboticsProfessorCourse2 > Inverted3 > MPU_Tone_3axis

「数学の知識や力」というよりは「初耳の説明をいきなりされたとき、きちんと自分事として理解できる力」を問う事が目的です。

① ax ～ az をそれぞれ 2 乗する（pow 命令）→② 求めた 3 つの値を全部足す →③ 足した値の平方根を取る（sqrt 命令）→④ 求めた平方根を map 命令に入れて、周波数の値を出す

という流れになっているので、どのステップで止まっているのかをまず見てあげてください。

なお、元々 map 命令に入れていた「変換前の変数の範囲」が 0 ～ 4000 になっていますが、この部分も若干変更する必要があります（値を 1.8 ～ 2 倍程度にする）

3. 加速度センサーで「傾き具合」をはかる（目安10分）

3.0. 加速度センサーのその他の使いかた

「センサーが測った値をいったん計算式に入れて、“出力に適した値”に変換する」という流れを体験することができましたね。

今回の授業では「加速度」をそのまま計算に使っていたので、シールドを激しく動かすほど音が変わるユニットになりました。
でも、計算の方法を変えれば加速度センサーの他の使いかたもできるはずです。

たとえば、このプログラムを書き込んでみましょう。

プログラムの書き込み
RoboticsProfessorCourse2 > Inverted3 > toneypr

実行結果：シールドの傾き具合に応じて、スピーカーの音が変わる

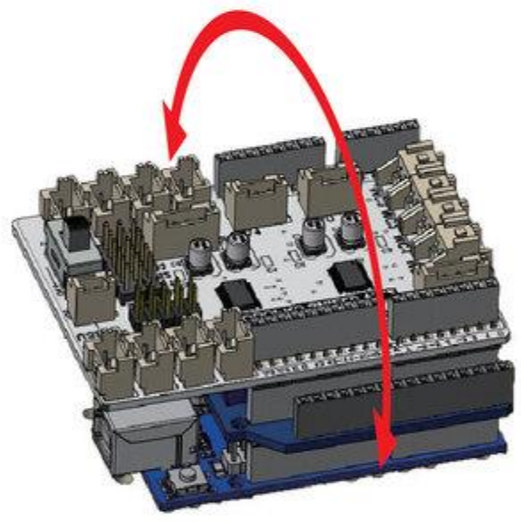


図 3-0　シールドを傾ける向き

シールドをただ静かに傾けているだけで、音が大きく変わりますね。プログラムの終わりの方を見てみましょう。

プログラム「toneypr」より抜粋（ばっすい）

```
        input = ypr[2] * 180 / M_PI + 180;
    (中略)
freq = constrain(input, 90, 270);
freq = map(freq, 90, 270, 262, 523);
spk.play(freq, 500);
```

最終的に変数 freq を周波数に使っているのはこれまでと同じですが、freq を算出するときに input というナゾの変数を使っていますね。
これがこのプログラムの一番肝心な部分です。
計算方法自体は非常に難しい（高校数学の知識をフル活用しています）ので省略しますが、とにかくこの input という変数は「今のシールドの傾き具合」を角度で表したものになっています。

シリアルモニタを表示しながらシールドを傾けてみると、2 ～ 3 桁の数字が次々出てきますね。
まっすぐ（シールドを上に向けて普通に机に置いた状態）のときが 180 度として、手前や奥にシールドを傾けると数字が変化するようになっています。

もしまっすぐおいたときに数字が 180 度から大きくずれているときは、プログラムの調整をしてみましょう。

チャレンジ課題

プログラム「toneypr」より抜粋（ばっすい）

```
input = ypr[2] * 180 / M_PI + 180;
```

① 角度の初期設定は、この赤字の部分の数字で調整できるよ。
ユニットを真っ直ぐ置いたときの input が 180 くらいになるように、必要に応じてここの数字を変えてみよう。
数字を変える必要があった場合は、いくつに変えたら丁度良くなったかメモしておこう。

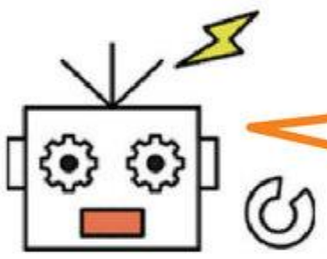
② 今はまっすぐから前後 90 度までの傾きしか感知しないことにしているよ。
プログラムを自由に変えて、もっと大きな傾きにも対応できるようにしたり、より高い・低い音まで鳴るようにしてみよう。

講

「toneypr」に登場する input という変数は、本ターム第 6 回で「倒立振子ロボ」を動作させるためのプログラムでも登場します。
よって、ここで角度の調整をする必要があった場合は、第 6 回でも同じ調整をすればよいことになります。生徒が値を変更していた場合は、いくつに変更したかメモなどして記録させておいてください。
②の課題は、constrain および map 命令内の数字（「90」や「270」）を自由に変更させてください。

4. まとめ（目安５分）

前回までは「センサーで何の値をはかっているか」を学んでいましたが、今回は「はかった値を変換して使う」ことを学びました。
「感じて・考えて・動く」という要素をロボットに持たせるためには、「使っているセンサーがどんな値を取りこんでくるのか」と「動かすためのパーツにはどんな値を使えばよいのか」の両方をよく知っておかなければなりません。そしてさらに「入力値を出力値に変換するとき、どんな計算式を使うとちょうどよくなるか」を考えるには、色んな数学の知識も必要になりますね。
学校の勉強はよく「〇〇なんて将来使わないのにどうして学ぶんだ！」なんてグチが出てきそうになりますが、少なくとも学校で学ぶ数学はロボット作りにはかなり直接的に活きてくることが感じ取れたと思います。



次回は、オムニホイールロボットにセンサーを搭載（とうさい）するよ！

講

○　以下の理解度を確認します。
・加速度センサーの入力値について整理する
・入力値を出力値に変換する
・三平方の定理を使い、より実践的な値を算出する方法を知る

○　次回のテーマは、「姿勢検出シールドを使ってロボットを動かす」であることを告知します。

≪次回必要なもの≫

次回は、今回使った姿勢検出（しせいけんしゅつ）ユニットの他に、以下のパーツを持ってきてください。

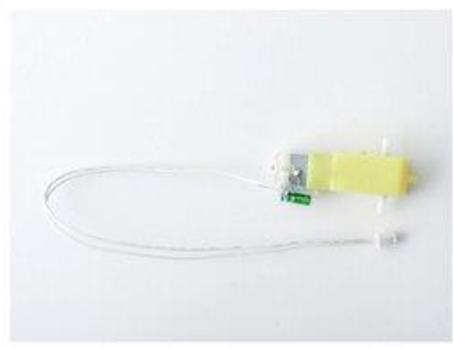
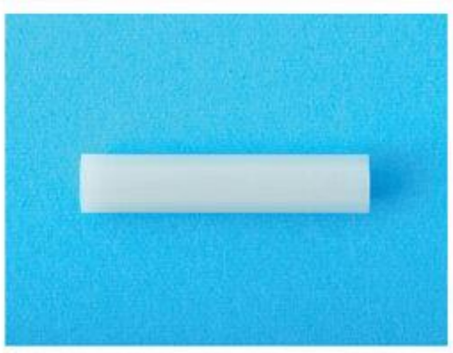
ラジオペンチ 1	ドライバー 1	USB ケーブル 1	ギアドモーター 3
			
タッチセンサー 1	モーター L 字ステイ 3	マトリクス LED 1	オムニホイール 3
			
M2.6L20 タッピングネジ (A) 3	赤円形ボード 1	白円形ボード 1	M3 ナット 19
			
M3L5 ネジ 7	M3L8 ネジ 11	M3L25 ネジ 6	8mm 角スペーサー 4
			
30mm 角スペーサー 3	7 セグメント LED 1		
			

図 4-0　次回必要なもの