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①

（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1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①

第1回

加速度を学ぶ

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。

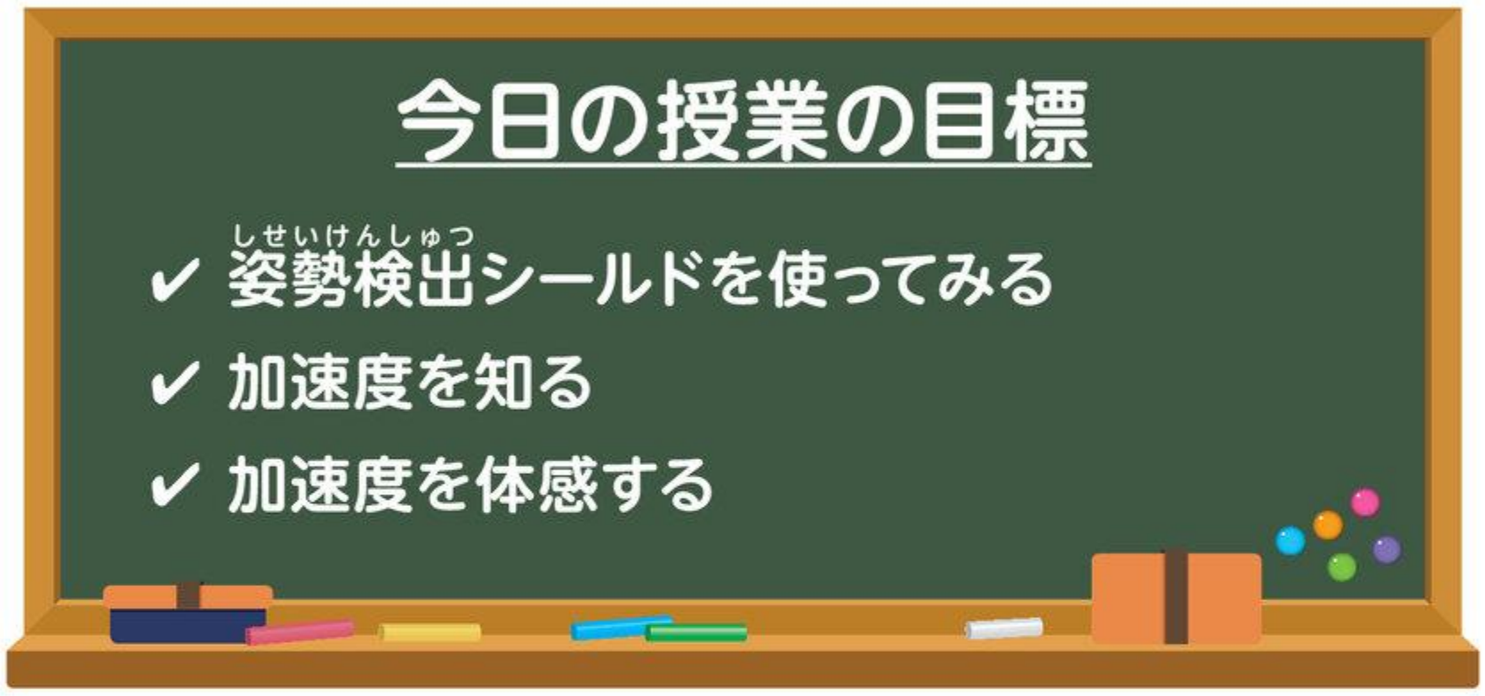
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（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. 加速度を学ぶ（目安 10 分）

0.0. 「加速度を学ぶ」でやること



突然ですが、「倒立振子ロボット」と聞いて、どんな動きをするロボットか思い浮かびますか？「倒立」とは上下逆さまに立っていること（「逆立ち」のことを「倒立」と呼んだりしますね）、「振子」とは「ふりこ」のことです。つまり「倒立振子」は、上下逆さまのふりこということになります。

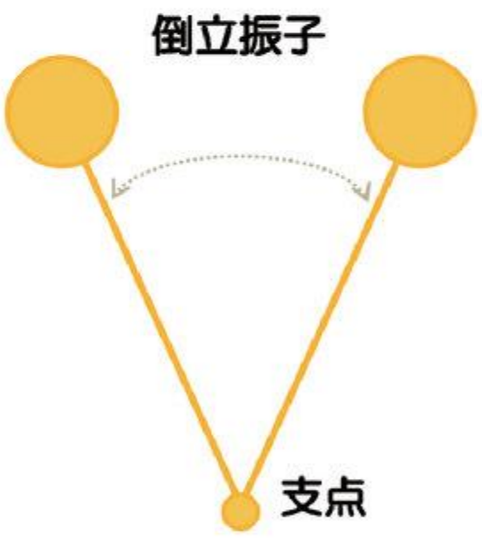


図 0-0　倒立振子の動きのイメージ

当然ですが、ふりこは倒立していない、つまりおもりが下に吊り下げられているものの方が安定しています。倒立振子は安定していない分、なんらかの方法でバランスを取る必要があるのですね。

バランスの取りかたにもいくつか種類がありますが、もっとも一般的なのは「地面と接する部分にタイヤなどを取りつけ、おもりの動きに合わせてタイヤを制御してバランスを取る」という「台車駆動型倒立振子」でしょう。

おそらく、皆さんにとっては「一輪車」と言われるとイメージしやすいのではないでしょうか。

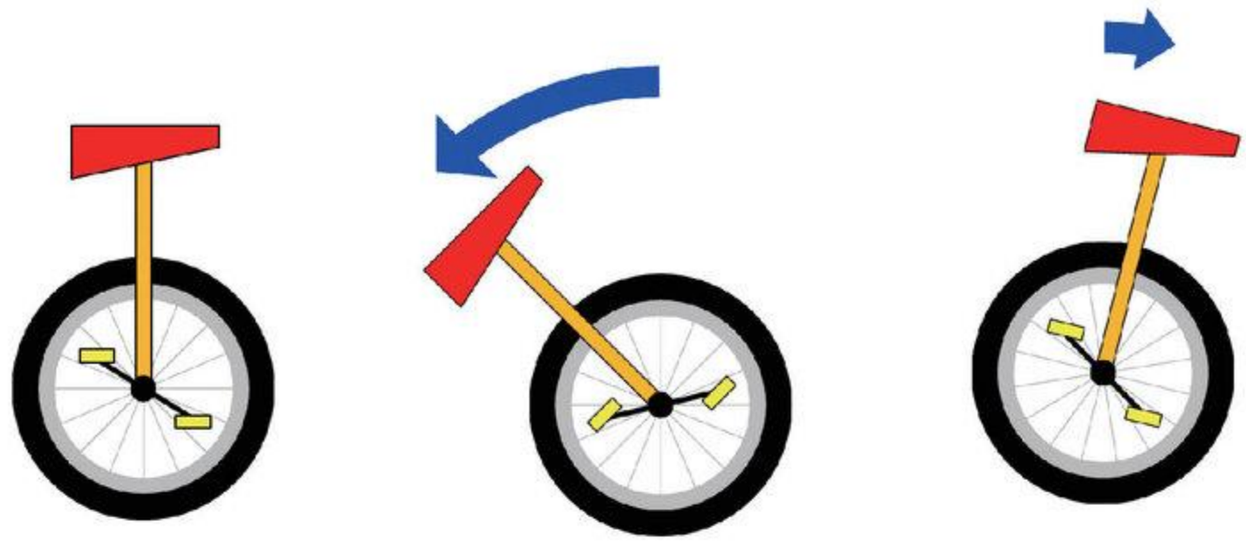


図 0-1　一輪車のバランス

乗り手が前後に傾(かたむ)いたら、それに合わせてタイヤを前後に回転させることで倒れるのを防ぎますね。
これから製作する倒立振子(とうりつしんし)ロボットも、基本的にはこれと同じです。傾(かたむ)き具合をセンサーで検知して、それに合わせてモーターを回転させればよいわけです。

では、ここで必要なのは「何」を検知するセンサーでしょうか。これまでに使ってきたタッチセンサーやカラーセンサーは、今回の制御には向いていないと思いませんか。
今回はその名も「姿勢検出(しせいけんしゅつ)シールド」という、新登場の各種センサーがセットになったシールドを使います。搭載(とうさい)されているセンサーは「加速度センサー」「角速度センサー（ジャイロセンサー）です。
「加速度」「角速度」どちらもまだ学校の理科では学んでいない！という人がほとんどだと思いますので、まずはこれらの考え方をしっかり学び、センサーの使い方をマスターしてから倒立振子(とうりつしんし)ロボットを製作していきましょう。

目に見えないものだって重要ダ！　物理量をとらえるゾ。

0.1. 必要なもの

ロボプロシールド、マトリクス LED シールドなどのピンヘッドが曲がっている場合は、ラジオペンチを使って元のように整えておきましょう。

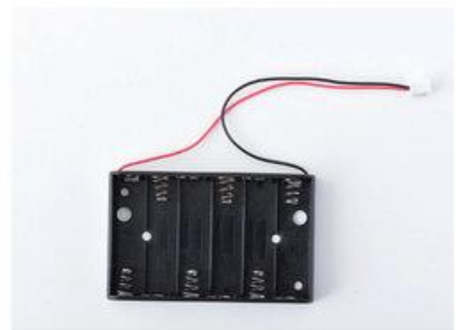
USB ケーブル 1	マイコンボード 1	ロボプロシールド 1	電池ボックス 1
			
マトリクス LED シールド 1	**マトリクス LED 1**	**スピーカー 1**	**7 セグメント LED 1**
			
姿勢検出シールド 1			
			

図 0-2　必要なもの

講

今回の授業で使用する、「姿勢検出シールド」は、三軸（xyz 軸）加速度センサー、三軸角速度（ジャイロ）センサーを搭載しています。
この姿勢検出シールドを活用して、加速度、角速度の学習と、ロボットへの活用について学習します。第 1 回は、加速度の学習となります。

１．加速度（目安１０分）

1.0.　加速度とは

１）身のまわりの加速度

加速度とは「勢い」という言葉が一番近いかもしれません。
「速度」と同じ漢字を使っていますが別物です。まぎらわしいと思いますので、しっかりと説明します。

たとえば、皆さんが電車や車などの乗り物に乗っているとき、乗り物が加速すれば後ろに、減速すれば前に倒れそうになりますよね。その「身体を倒そうとする力」の大きさは何によって変わるのか考えてみてください。
乗り物の速度でしょうか？　しかし、時速 300km ほどで走る新幹線に乗っているからと言って、中の人々がものすごい勢いで飛ばされたりすることはありません。反対に、時速 30km ほどでゆっくり走っている電車でも、急ブレーキがかかると思い切り倒れそうになりますね。これは、「身体を倒そうとする力」が「速度」ではなく「加速度」によって変わるからです。

「加速度」とは簡単に説明すれば「どれだけ急激（きゅうげき）に速度が変化するか」の値をさします。停止していた自動車でも、思い切りアクセルを踏（ふ）んで一気に加速したら、もとの速度は 0 であっても加速度は大きくなります。そのため、急加速する車内では中の人にも大きな力がかかります。反対に、時速 300km ちょうどを維持しながら走っている新幹線は、速度が変化しているわけではないので加速度は 0 ということになります。よって、車内の人には力がかからず、踏（ふ）ん張（ば）ったりする必要がないというわけです。

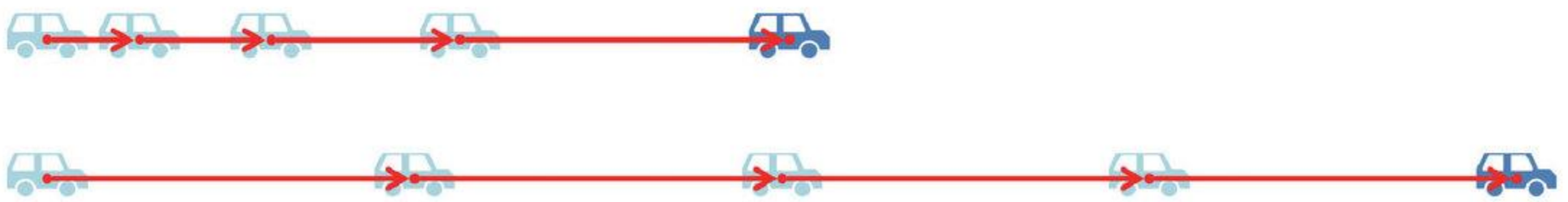


図 1-0　下図は「速度」は大きいが「加速度」は 0

もう少しくわしく説明しましょう。
ものには「力がくわわらない限り、直前の動きをずっと続けようとする」という性質（慣性（かんせい）といいます）があります。静止している物体はずっと静止したまま、動いている物体はずっと同じ速さで動き続けようとするのです。もし静止している物体を動かしたり、動いている物体を加速・減速させたいときには力をくわえなければいけない、という事でもありますね。

たとえば速度 0 の電車が走り始める場合を考えてみましょう。この場合、加速のためのモーターなどの力はあくまで「電車」にかかります。乗っている人たちにはこの力はかからないので、電車が加速し始めても「静止したまま」を保とうとしますね。
すると乗っている人たちにとっては「身体は元の場所に居続けようとしているのに、床が前に進んでいく」という状況になりますね。このため、身体を後ろに傾(かたむ)けるような力を感じるというわけです。

図 1-1　静止状態の電車が加速する場合

走っていた電車がブレーキをかけたときはどうでしょうか。速度が小さくなっていくので、加速度はマイナスになるはずですね。

この場合、電車が走っていたわけですから、乗っている人たちも「電車と同じ速度でずっと動き続けようとしている」ということになります。ここで電車がブレーキをかけるとどうなるかわかりますか？
答えは「身体は進もうとしているのに床だけ減速した」ですね。こうなると、前に倒そうとする力を感じることになるでしょう。

図 1-2　運動状態の電車が減速する場合

ちなみに同じ速度のまま走り続けている場合、つまり加速度がゼロの場合は、「身体が進もうとしていて、床も同じ速度で進んでいる」という状態ですから、乗っている人は力を感じません。

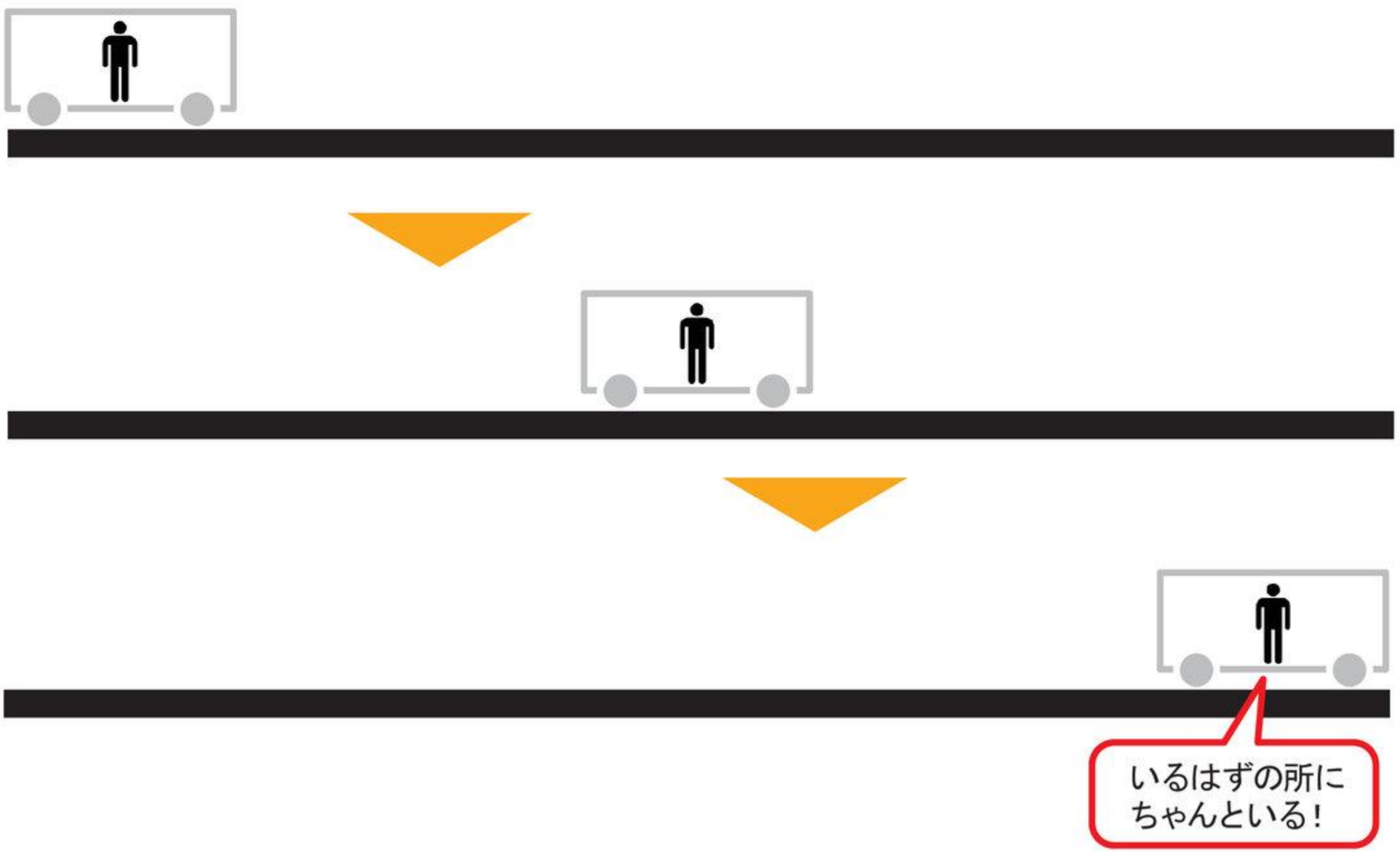


図 1-3　一定速度で動き続ける場合

まとめると、「ものに力をくわえると、加速度が生じる。生じた加速度が大きいほど、くわわった力も大きいといえる」ということになります。

倒立振子（とうりつしんし）ロボットも加速度をはかることで、「前と後ろどちらの向きに倒れそうになっているか」「また、その倒れる力はどれくらいか」を知ることができます。これに合わせて、足元のモーターの回転を制御すればいいわけですね。

2）G とは

先ほどの説明にもあった通り、ものには力がかかるとその速度が変化するという性質があります。そして地球上のものには「重力」がはたらいていますね。下へ向かって落下している物体は、重力によって落下速度が変化していくことになります。このときの加速度を「重力加速度」といいます。地球上だと、重力以外なんの力もくわわっていなければ、落下速度は 1 秒たつごとにおよそ時速 35km ずつ大きくなっていきます（場所や高度によって少しずつ差は出ます）。10 秒間落下すれば時速 350km にもなる、という計算です。

ところで「ロケットの打ち上げでは、乗組員の身体に 3G もの力がかかる」などといった話を聞いたことがないでしょうか。
重力加速度とおなじ、つまり 1 秒につきおよそ時速 35km ずつ速くなっていくような加速度を「1G」と表します。ですから、上にかいたロケットのたとえであれば、乗組員は 1 秒につきおよそ時速 105km ずつ速くなっていくような加速度を受けることになります。当然、その加速度に応じて身体が引っ張られるような力を感じますが、「G」は正確には力の単位ではなく、加速度の単位であることは覚えておきましょう。

ちなみに、F1 レースではコーナーを曲がる際、レーサーの身体に 6G もの加速度が発生しているそうです。F1 レーサーはスポーツ選手と同様、本格的に体をきたえますが、その理由にも納得ですね。

コラム　雨粒が痛くないのはなぜか

雨雲は季節や地域などにもよりますが、高いところで数千 m も上空に発生します。ここから落ちてきた雨粒はとても長い時間をかけて落下してきますから、地表に届くころにはとんでもない速度になっているのではないでしょうか。もし 3000m 上空の雨雲から雨粒が落ちてきたら、理論上は 25 秒ほど落下し、最終的に時速 900km くらいになっているはずです。ピストルの弾より少し遅いくらいの速度ですから、いかに小さく軽い雨粒とはいえかなりの威力（いりょく）になりそうですよね。しかし、雨粒が当たったせいで骨折したとか、雨粒でアスファルトが穴だらけになったといった話はなかなか聞きません。

さて、雨粒が上空から落ちてくるとき、くわわる力は「重力」だけでしょうか？
もう１つ、重要な力があります。それは空気抵抗です。
空気中をものが進むとき「抵抗を受ける面積」と「ものの速度」が大きくなるほど受ける空気抵抗も大きくなります。
ですから、雨粒が加速していくとそれに合わせて空気抵抗も大きくなっていくのです。重力の大きさは加速しても一定のままなので、やがて重力の大きさと空気抵抗の大きさが等しくなってしまい、雨粒にはたらく力が差し引きゼロになり、加速度もゼロになる、つまりそれ以上加速をしなくなる、というわけです。
実際の雨粒は、粒の大きさにもよりますがだいたい時速 10~30km くらいで地表に降ってくるものが多いようです。

2. 姿勢検出シールドの組み立てと加速度の応用（目安 85 分）

2.0. 姿勢検出シールドとは

冒頭でも説明しましたが、姿勢検出シールドには、「加速度センサー」、「角速度センサー（ジャイロセンサー)」などが入っています。いずれも高精度なロボットの動作に必要になる場合があります。

「加速度センサー」は、加速度を検出します。身近なところではスマートフォンにも搭載されていて、機器を持った人が歩くと、「動いた / 止まった」ということを知ることができます。動き始めるときには進行方向に加速がかかりますし、また、止まるときには逆に進行方向に対して減速がかかります。さらに、このセンサーは地面に対して、どのくらい傾いているかを検出することができます。

2 つ目の「角速度センサー（ジャイロセンサー)」は、回転角速度の測定ができる「慣性センサー」と呼ばれるものの一種です。角速度は、「ある物体の角度が一定時間あたりにどれだけ変化しているか」を示す値です。つまり物体が回転している速度を示します。これもスマートフォンやゲーム機には一般的に搭載されています。

加速度、角速度センサーともに、一度に検出できる軸方向の数によって「1 軸」「2 軸」「3 軸」と示し、この軸方向は、**図 2-0** の、「x 軸」、「y 軸」、「z 軸」の方向となります。

姿勢検出シールドは、このような目には見えない力を見えるようにすること（可視化）に役立ちます。ロボットの状態を可視化して、そのときの状態に合わせて制御をする目的で使います。

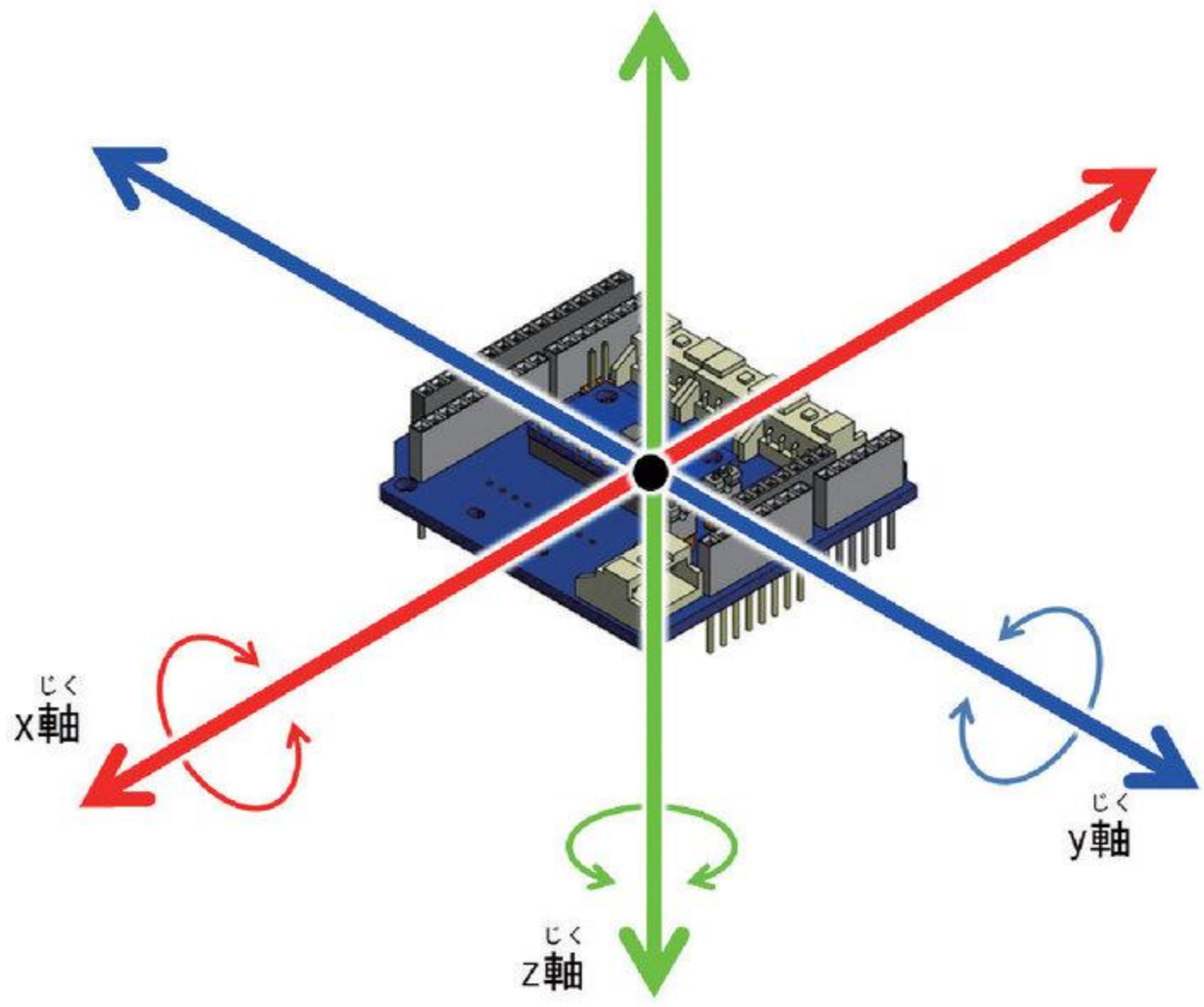


図 2-0　姿勢検出シールドの 3 軸

2.1. 姿勢検出シールドの組み立て

それでは準備をしていきます。
まず、姿勢検出シールドをマイコンボードに組み込みましょう。

図 2-1 の順番で組み立ててください。下から、マイコンボード、姿勢検出シールド、ロボプロシールド、マトリクス LED シールド、7 セグメント LED です。
それぞれのシールドのピンはピンソケットにきちんとはめ込んでください。

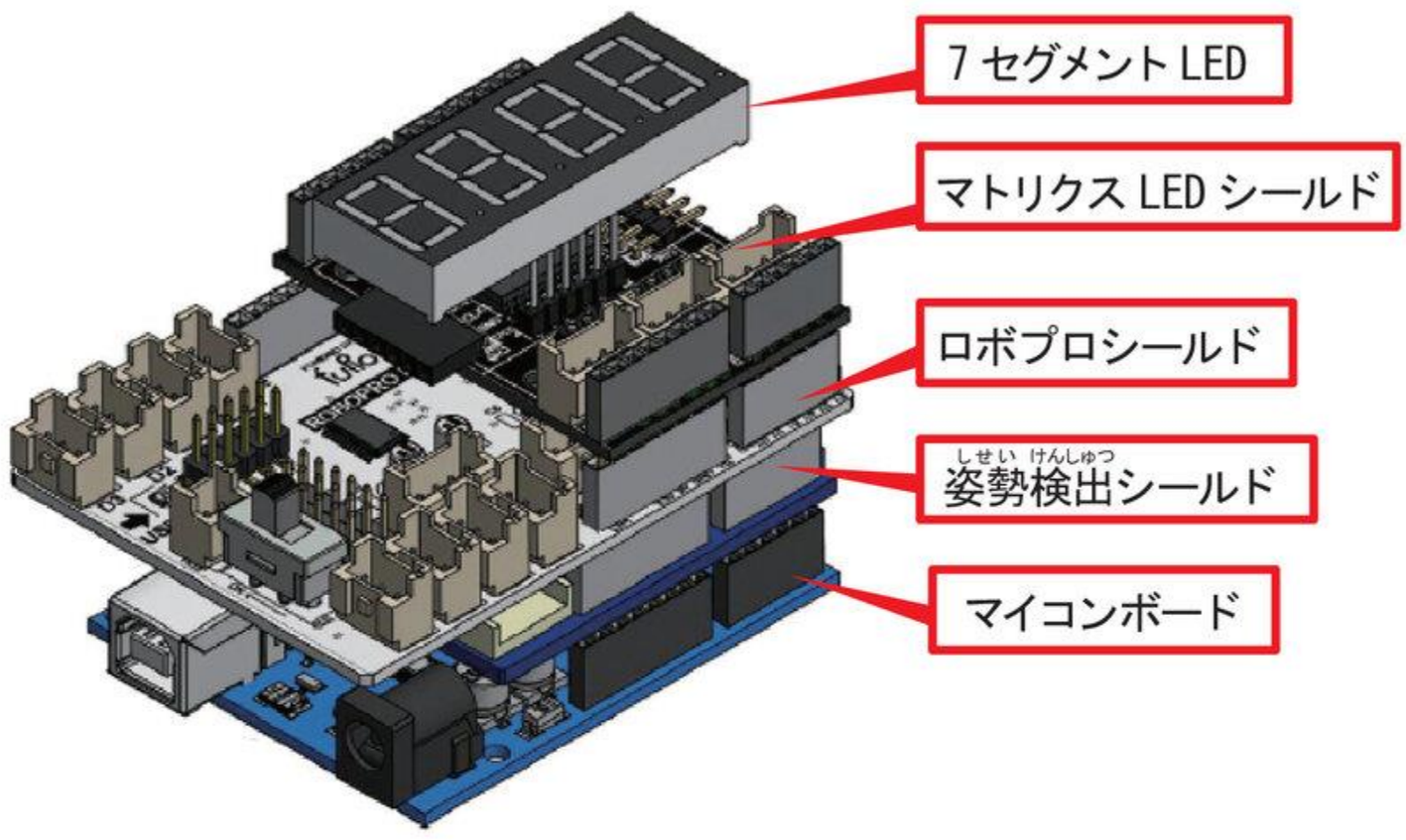


図 2-1　姿勢検出シールドの接続

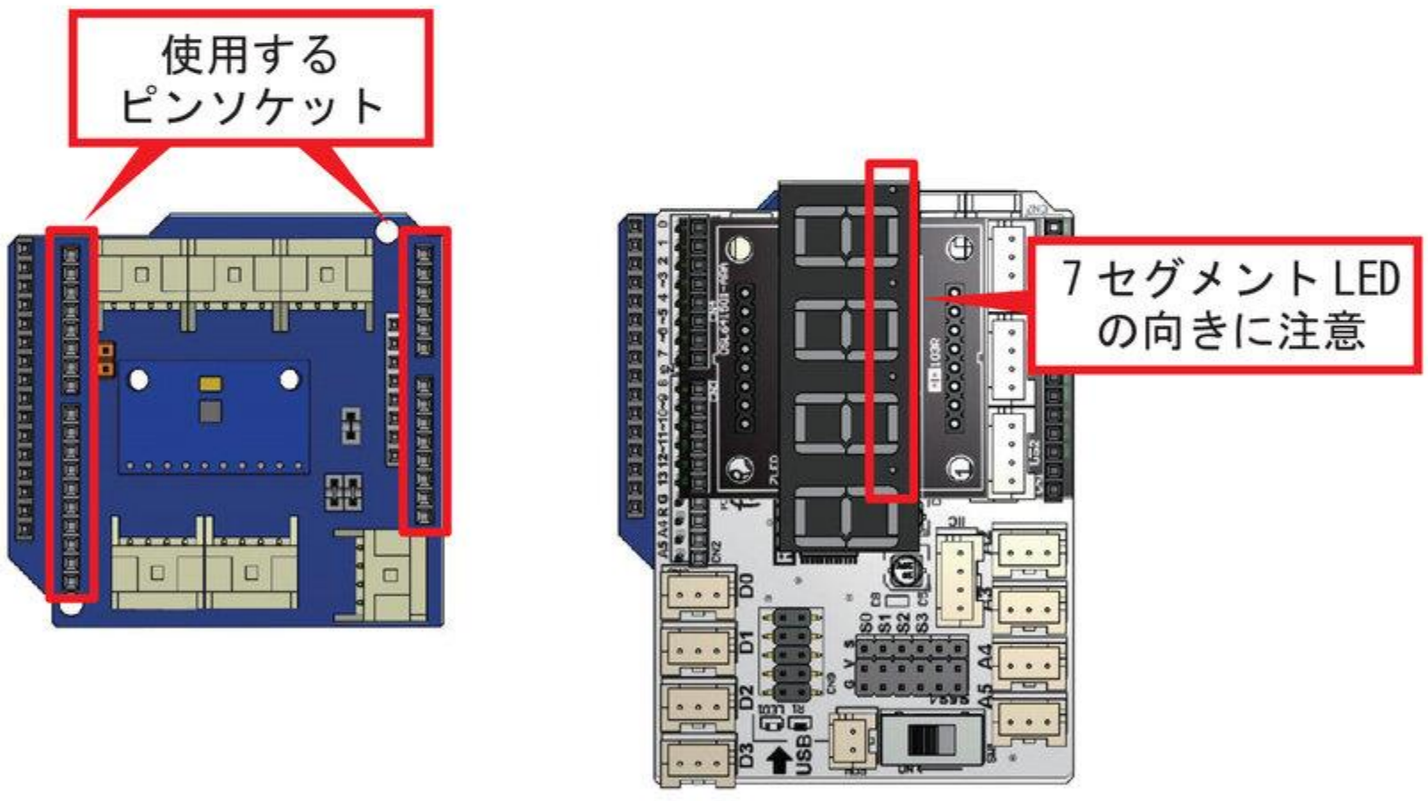


図 2-2　姿勢検出シールドの組み立て時の注意

講　ピンヘッドの折れ曲がりはラジオペンチで丁寧に直して接続させてください。折れ曲がった状態で接続すると、取り外しのときに、ピンソケットが基板から剥離する恐れがあります。

2.2. 動作確認

ここでは加速度の検出をします。さきほど組み立てた姿勢検出ユニットを USB ケーブルでパソコンと接続し、電池ボックスも接続してください。
接続できたら姿勢検出ユニットをテーブルにいったん置いて、以下のプログラムを書き込みましょう。

プログラムの書き込み

RoboticsProfessorCourse2 ＞ Inverted1 ＞ MPUtest

プログラムが書き込まれたら、シリアルモニターを立ち上げて、通信速度を「115200baud」とします。

図 2-3　シリアルモニターの表示のやり方

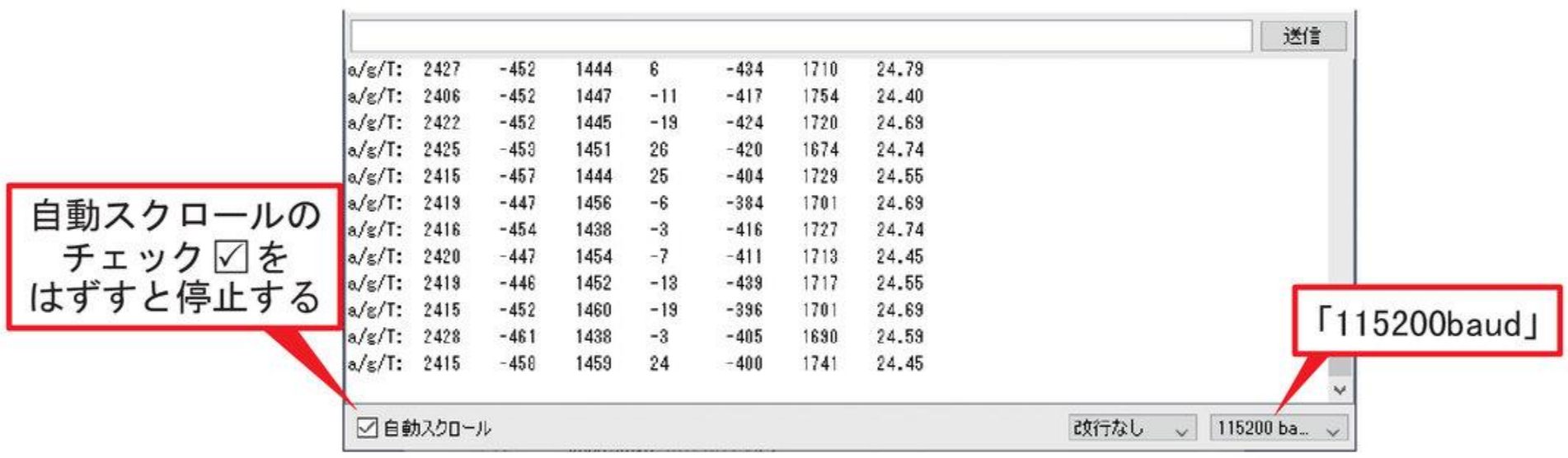


図 2-4　シリアルモニターの操作

図 2-4 のように 10 列の数字が表示されて、絶えず更新されている状態が正常です。姿勢検出ユニットを動かすと、数字が変化しますか？ 変化しない数字があれば、シールド同士がしっかりと接続されているか、ピンが曲がってしまってソケットに入っていないものがないか、確認してください。ただし、一番右側の数字は温度センサーのため、あまり変化がありません。

講 電池ボックスは、この場面では使用しないので邪魔になるようであれば接続しなくても大丈夫です。ただし、この後姿勢検出ユニットを左右や上下に振るなどする場面では使用した方が便利ですので、必要になったら再び接続し、使用してください。

姿勢検出ユニットを机の上に置いて静止した状態でも、シリアルモニターに表示される値はわずかな振動にも反応して動いています。では、姿勢検出ユニットを手で持ち上げて、図の矢印の方向にゆっくりと動かし、そのときの変化をシリアルモニターで確認してみましょう。すごいスピードで値が変化するので、ときどき画面左下の自動スクロールを止めて確認してください。表示されるそれぞれの列の値は**表 2-0** のように設定されています。

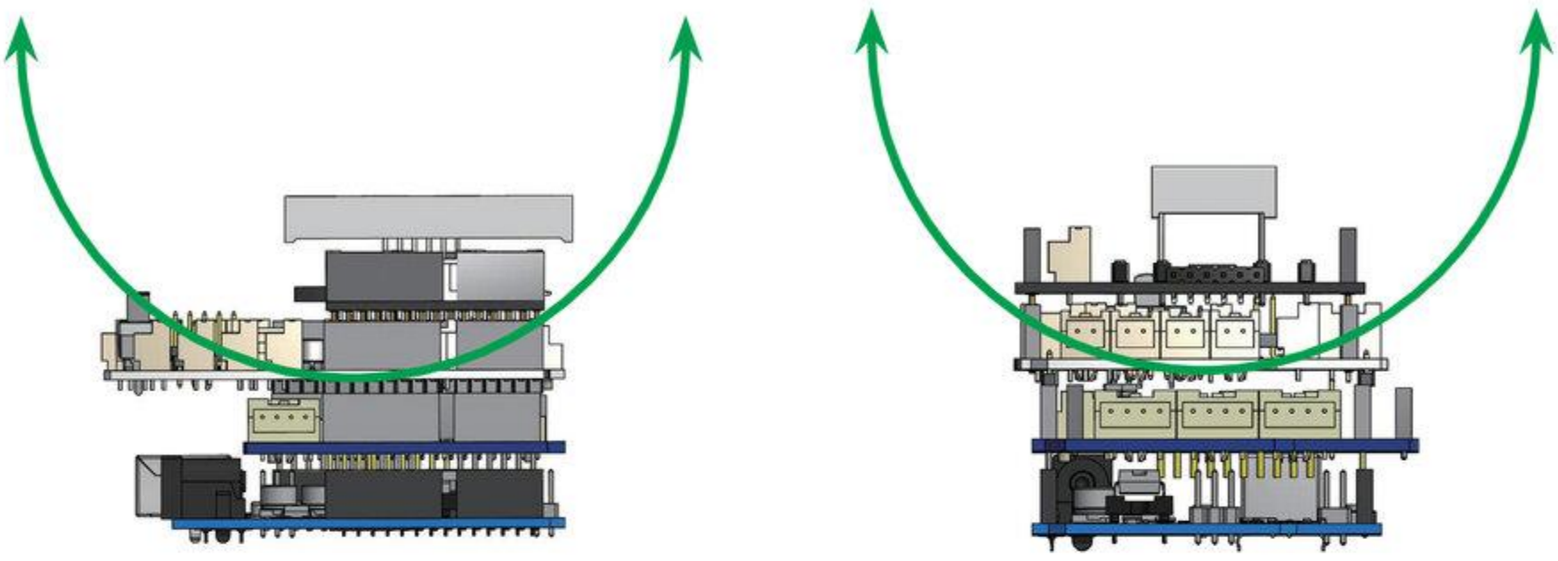


図 2-5　姿勢検出ユニットの動作確認

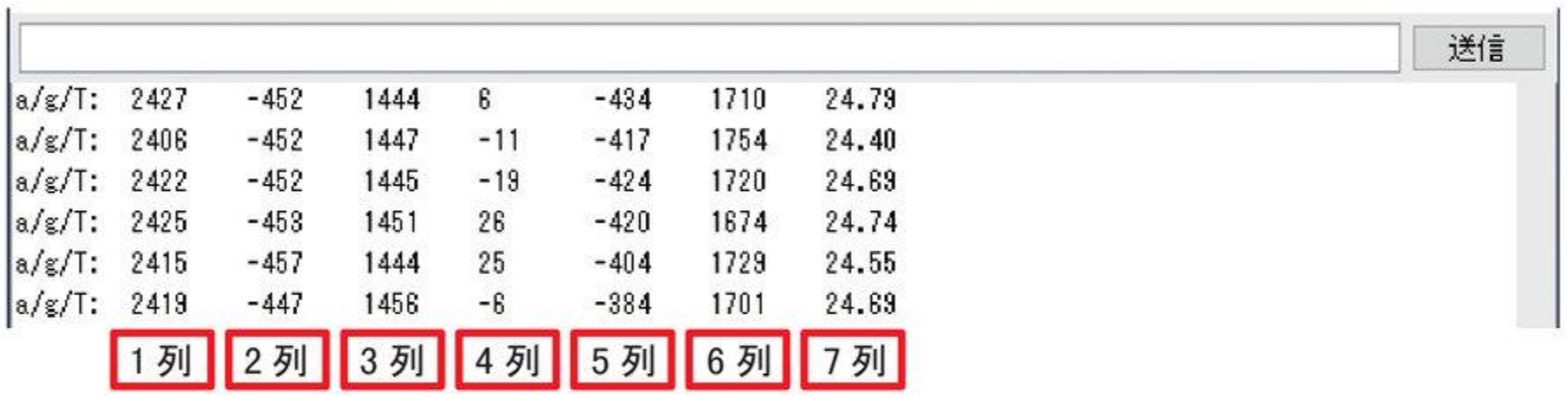


図 2-6　シリアルモニターの列

表 2-0　シリアルモニターの列表示の設定

1 列	x 軸方向の加速度（軸方向への加速度）
2 列	y 軸方向の加速度（軸方向への加速度）
3 列	z 軸方向の加速度（軸方向への加速度）
4 列	x 軸の回転角速度（軸まわりの単位時間あたりの回転角）
5 列	y 軸の回転角速度（軸まわりの単位時間あたりの回転角）
6 列	z 軸の回転角速度（軸まわりの単位時間あたりの回転角）
7 列	温度（℃）センサーの温度

それでは、センサーの状態を見ていきましょう。**図 2-7** を参考にしながら動かし、値の変化をシリアルモニターで観察しましょう。
三次元空間を表すには、グラフのときにやった x、y 軸だけでなく z 軸を用いて表現します。グラフに奥行き方向をつけたのが z 軸と思ってください。なお、**図 2-7** の軸方向は加速度と角速度センサーのものとなります。

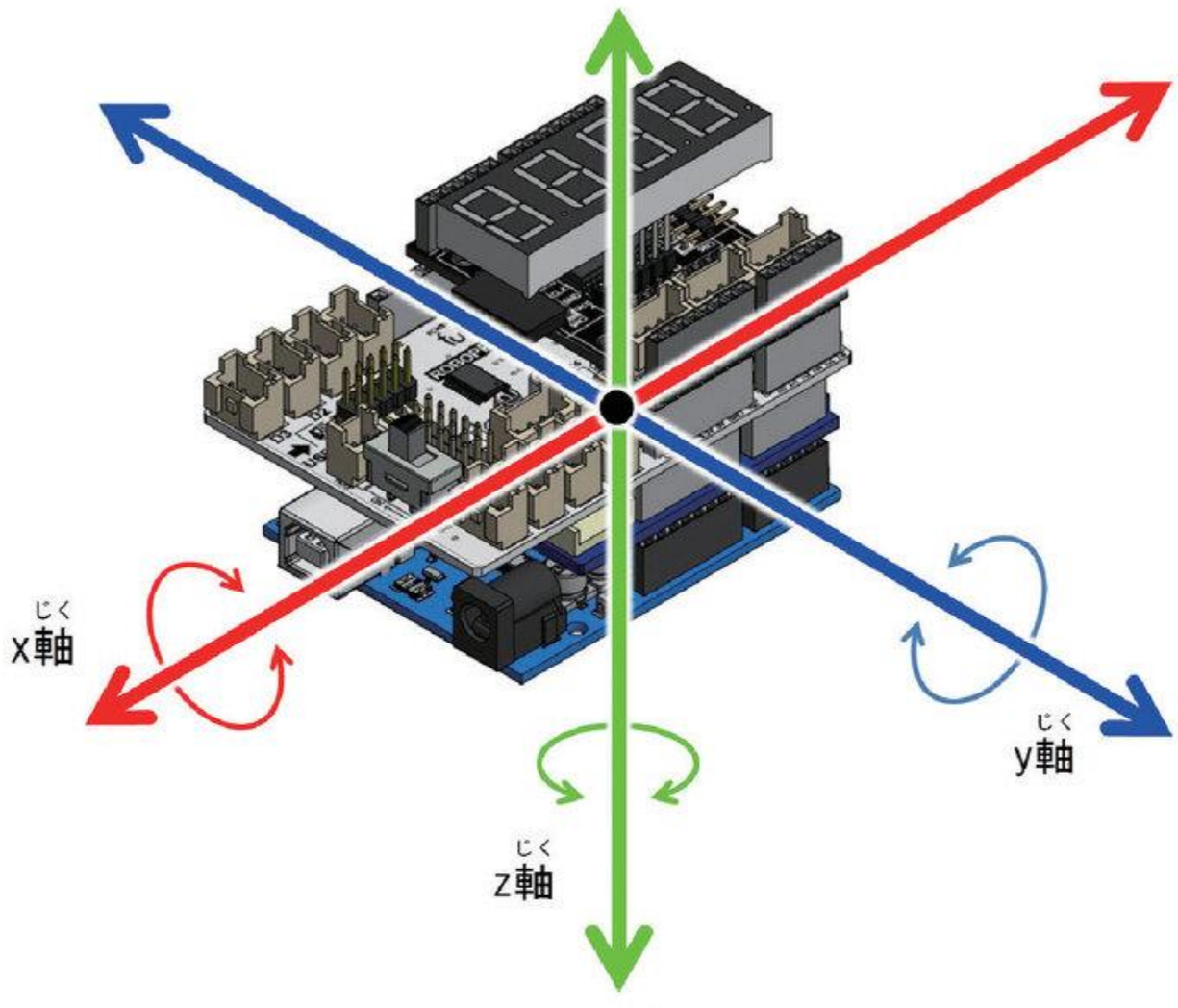


図 2-7　軸方向

2.3. 重力加速度をはかる

重力加速度とは、すでに説明したように、地球の重力が地上にある物体に影響をおよぼす加速度です。手放したものは地面に向かって落下していきますが、速度は一定ではなく、だんだんと加速しますよね。これが重力加速度です。
それでは、重力加速度を体験してみましょう。以下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
RoboticsProfessorCourse2 > Inverted1 > GravityLED

実行結果：加速度の値が 7 セグメント LED に表示される。傾けるなどすると様々に値が変化する。

机に LED を上向きにして置いていれば、100 くらいの数字が表示されます。z 軸方向の加速度が検出されているのです。これで、重力加速度を検出できました。表示が 100 のときに、ちょうど 1G ということになります。持ち上げたり、傾けたりもしてみましょう。真横にすれば重力加速度がないため、どの方向に転がしても 0 となり、ひっくり返せば－ 100 になります。

講　安全対策として、姿勢検出ユニットを動かす際には、USB ケーブルをパソコンから外し、電池ボックスからの電力で動かすように指導してください。電池ボックスのリード線がちぎれないように手でおさえて実験するようにしてください。

やってみよう！

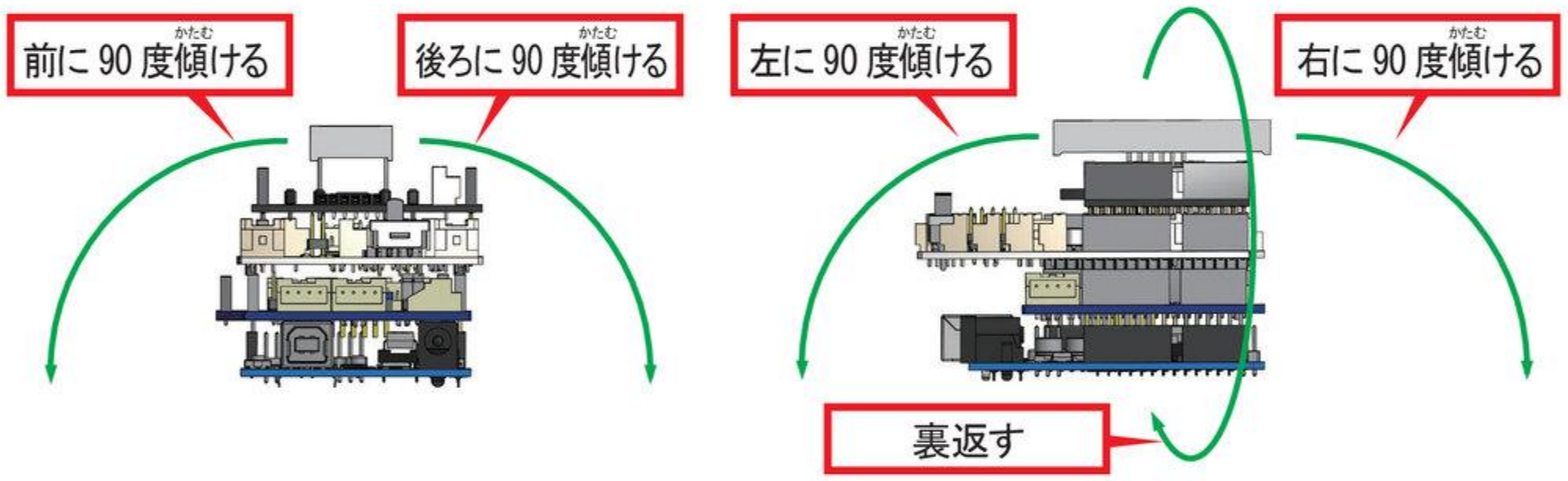
プログラム「GravityLED」を変更して x 軸、y 軸の値も 7 セグメント LED に表示させよう。
変数 ax が x 軸、ay が y 軸の加速度の変数になるよ。以下の黄色の部分の変数を入れかえたら、x、y 軸に対する加速度を表示できるよ！ 実験してみよう。

```
void loop(){
    accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my, &mz);
    // 姿勢センサーの検出
    temp = accelgyro.getTemperature() / 333.87 + 21; // 温度検出

    lc.setDec(0, az / 20);                    // z軸の加速度を7セグメントLEDに表示

    delay(500);
}
```

変数 az のとき、机に置いた状態（初期値）では、100 が表示されたね。
そして下の図のように前後左右に傾けていくと、100 程度から 0 程度に値が変化したね。
さらに裏返すと、－ 100 程度に値が変化したよね。
では、変数を ax と、ay に変更したときに同じように動かして、値を確認しよう。



講

・ax に変更した場合
　初期値：0　裏返す：0　前後：0
　左：値が約 0 ～ 100 まで変化　右：値が約 0 ～－ 100 まで変化

・ay に変更した場合
　初期値：0　裏返す：0　左右：0
　前：値が約 0 ～－ 100 まで変化　後ろ：値が約 0 ～ 100 まで変化

やってみよう！

最大の G を記録保持するようにプログラム「GravityLED」をかきかえよう。以下の黄色の部分を追加や変更（へんこう）してみよう。

```
int a_max = 0;                                          // a_maxという名前の
                                                           変数を用意

void loop(){
    accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my, &mz);
    // 姿勢センサーの検出
    temp  = accelgyro.getTemperature() / 333.87 + 21; // 温度検出

    a_max = max(a_max, az);                             // max関数を使い、大
                                                           きい方の値を返す


    lc.setDec(0, a_max / 20);                           // z軸の加速度を7セグ
                                                           メントLEDに表示

    delay(10);
}
```

ここで出てくる命令「max」は以下のようなものです。

命　　令「max」
実 行 結 果：数値 1 と数値 2 を比べて、大きい方の数値を返す
使　い　方：max([数値 1],[数値 2])

この関数を使えば簡単ですね。
ところで、なぜ max を 20 でわっているのかというと、7 セグメント LED に表示する 1G の値をちょうど 100 にしたかったからです。実は、1G を検知したとき、そのままだと表示される値は 2000 くらいです。そこで、20 でわって、1G がちょうど 100 として表示されるようにして、直感的に分かりやすくしているのです。

講 解答プログラムは以下となります。
RoboticsProfessorCourse2 ＞ Inverted1 ＞ GravityLEDmax

やってみよう！

センサーをふり回してみて、最大何Ｇまで表示を出せるか試してみよう。
また、どのようなときに数字が大きくなるか、観察して以下の欄に考えを記入してみてね。
ただし、たたきつけたり、ぶつけたりしないように気を付けてね！

センサーの下をたたいたり、衝撃を与えたりするなど瞬間のインパクトが大きいとき。

講　衝撃を与えたときの作用を観察させてください。最大で 16G（表示でいえば 1600）まで検出できます。力加減に気をつけさせましょう。

2.4.　角度をはかる

重力加速度を利用して傾斜計（地盤の傾斜を測定する装置）を作ってみましょう。まずは、以下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

RoboticsProfessorCourse2 > Inverted1 > inclinometer

実行結果：z 軸に対する傾きの角度が 7 セグメント LED に表示される。
姿勢検出ユニットが z 軸方向から糸で吊り下げられているとイメージしてください。その状態から傾けたときの角度を表示しているわけです。

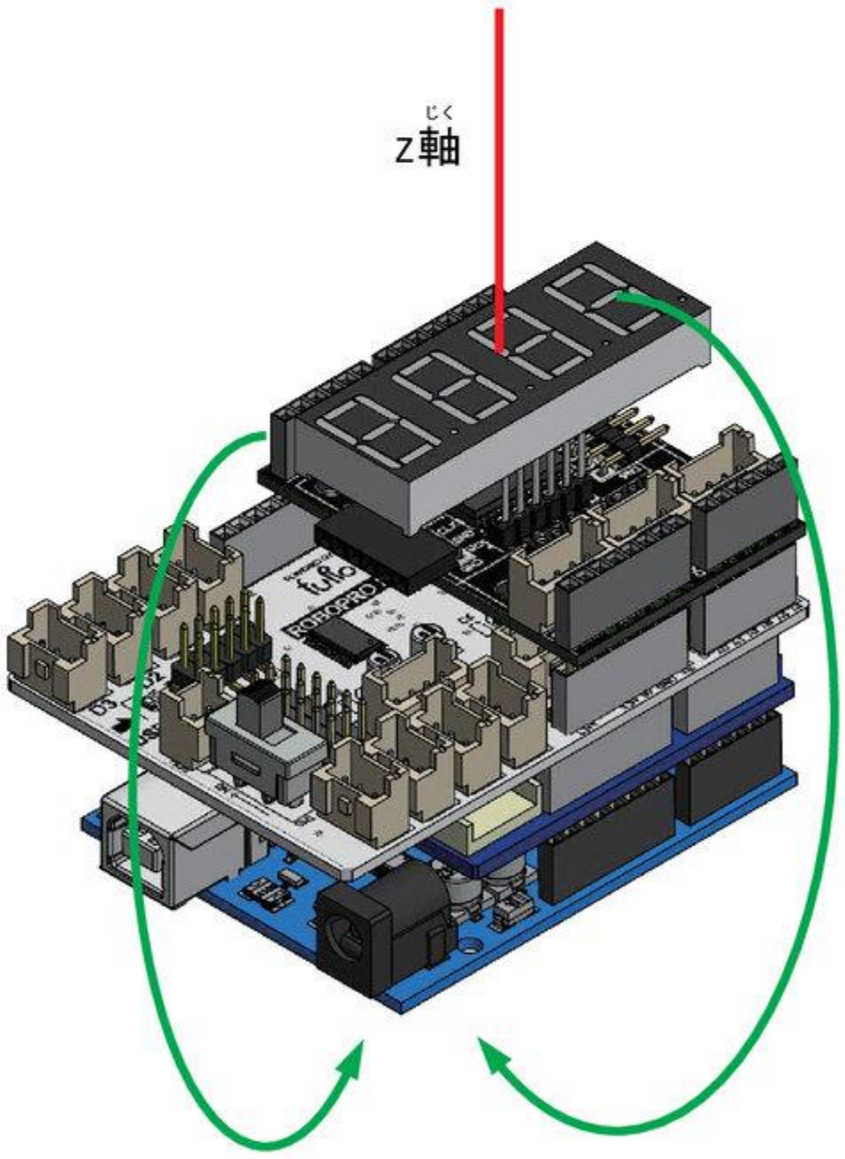
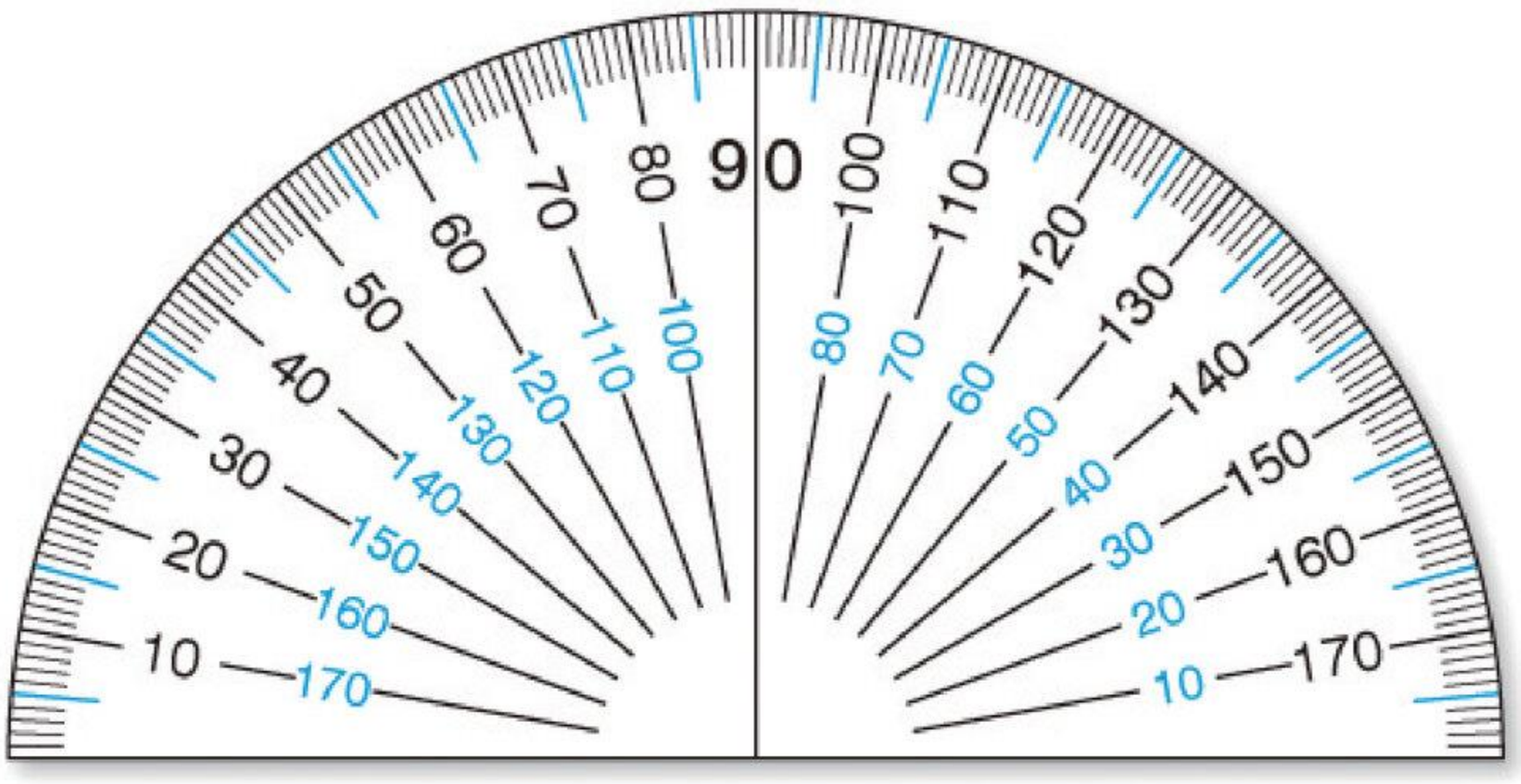


図 2-8　傾斜計

やってみよう！

姿勢検出（しせいけんしゅつ）シールドを分度器に合わせて傾（かたむ）けたときの値を確かめてみよう。



2.5.　加速度を体感する

次に、**図 2-9** のように、7 セグメント LED をマトリクス LED に付けかえてください。

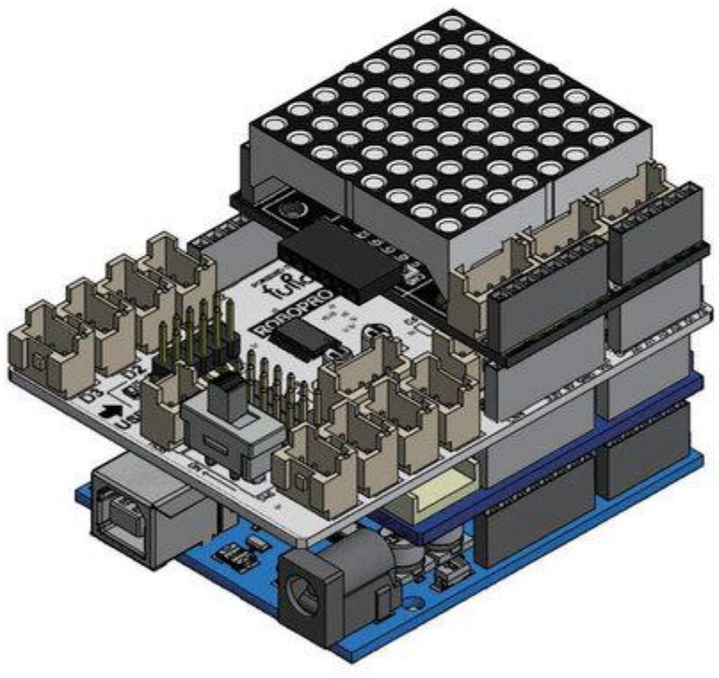


図 2-9　マトリクス LED シールドの付けかえ

以下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

RoboticsProfessorCourse2 > Inverted1 > AccelDrop

実行結果：姿勢検出（しせいけんしゅつ）ユニットを傾（かたむ）けたり、横に振ったりするとマトリクス LED のドットが動く。

講　マトリクス LED の取り付けの際、向きに注意させましょう。

「AccelDrop」はマトリクスLEDの特定の１ドットを点灯・消灯させる命令で出来ています。

プログラム「AccelDrop」より抜粋

```
myMatrix.write(x, y, HIGH);      // LEDを点灯
delay(50);
myMatrix.write(x, y, LOW);       // LEDを消す
```

変数 x 、 y はx軸方向、y軸方向の加速度によって決まります。しかし、計測値は先ほど説明した通り1Gあたり2000という非常に大きな値になってしまいます。当然マトリクスLEDはx座標、y座標とも0～7しかありませんから、加速度の値を0～7の範囲内に変換する処理が必要です。

プログラム「AccelDrop」より抜粋

```
x = map(ax, -16000, 16000, 0, 7);      // 0-7の間にする
y = map(ay, -16000, 16000, 7, 0);      // 0-7の間にする
```

命　　令　**「map(x, a1, b1, a2, b2);」**

実行結果：変数 x の値を最小a1、最大b1の範囲から最小a2、最大b2の範囲に変換する

使 い 方：map(ax, -16000, 16000, 0, 7);　　// 0-7の間にする

このように、map を使うことで計測した加速度の値をマトリクスLEDに反映させられるようになります。

続いて、以下のプログラムを見てみましょう。

プログラムの書き込み

RoboticsProfessorCourse2 > Inverted1 > LedAroundACC

実行結果：マトリクスLEDの外周のLEDが回転するように光る。姿勢検出ユニットを傾けると、回転速度が上がる。

1年目不思議アイテムⅠ -1で扱(あつか)ったプログラム「LedAround」のアレンジです。

1年目プログラム「LedAround」より抜粋(ばっすい)

```
const int dtime = 50;                    // LEDが移動する待ち時間

(中略)

void loop(){
    y = 0;
    for (x = 0; x < 8; x++){
        myMatrix.write(x, y, HIGH);  // LEDを点灯させる
        delay(dtime);
        myMatrix.write(x, y, LOW);   // LEDを消灯させる
    }
```

プログラム「LedAroundACC」より抜粋(ばっすい)

```
void loop(){
    accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my, &mz);
    // 姿勢センサーの検出
    dtime = map(abs(az),0,16000,1,100);
    y = 0;
    for (x = 0; x < 8; x++){
        myMatrix.write(x, y, HIGH);   // LEDを点灯
        delay(dtime);
        myMatrix.write(x, y, LOW);    // LEDを消灯
    }
```

for文の中身は、1年目のプログラムとまったく同じですね！　ただ、点灯時間を表す変数 dtime の決め方がちがっています。
はじめに「 dtime の値は50（ミリ秒）」と数字で宣言(せんげん)していただけの1年目プログラムとちがい、「LedAroundACC」では dtime をセンサーの検出(けんしゅつ)値に応じて決めています。ちなみに、abs(az) は変数 az の値の絶対値をさします。
ここでも map が使われていますね。今回は加速度を座標ではなく時間に変換するので、最大100ミリ秒程度になるようになっています。

では、1年目のプログラムをもう1つ開いて、同じく姿勢検出(しせいけんしゅつ)シールドで動作速度を制御できるようにアレンジしてみましょう！
アレンジ元のプログラムは以下の通りです。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemA4 > MatrixLineTurn

マトリクス LED 上に直線が表示され、直線がプロペラのようにぐるぐる回転しますね。

ステップアップ

プログラム「MatrixLineTurn」をかきかえ、回転速度を姿勢検出ユニットの加速度で変更できるようにしてみよう！

ヒント

delay(); の中身を変数 dtime にすることももちろん必要だけど、他にも必要なものがたくさんあるはずだよね。

「LedAround」と「LedAroundACC」のちがいをよく観察して、同じように要素をつけ足していこう！

講　解答例は以下のプログラムです。
RoboticsProfessorCourse2 > Inverted1 > MatrixLineTurnACC

プログラムの上から下まで、さまざまな部分を追加していったのではないでしょうか？
たとえば、はじめの部分を見てみましょう。

プログラム「MatrixLineTurn」より抜粋

```
#include <Sprite.h>
#include <Matrix.h>
```

プログラム「MatrixLineTurnACC」より抜粋

```
#include <Sprite.h>
#include <Matrix.h>
#include "Wire.h"
#include <math.h>
#include "I2Cdev.h"
#include "MPU6050.h"
```

マトリクス LED さえ使えればよかった「MatrixLineTurn」とちがい、「MatrixLineTurnACC」ではさらに姿勢検出シールドの各種機能を使わなければなりません。

プログラムのはじめにいつも #include という命令がかかれていましたね。これは「このプログラムでは○○の機能を使うから準備しておいてね」という宣言のようなものです。たとえばマトリクス LED を点灯させるとき、myMatrix.putch や myMatrix.putd2 といった命令を使っていましたよね。これらの命令の具体的な内容は Matrix.h というプログラムの中にかかれているので、まずはこのプログラムを呼び出し（引用）しておく必要があるということです。

このような、前準備として引用しておく設定用のプログラムを「ヘッダーファイル」などとよびます。詳しい使い方は他のテーマでまた何度か説明しますので、今はひとまず「使う機能を増やすときには #include を増やさないといけない」ということを理解しておきましょう。

2.6. 加速度を音で体感する

加速度は目に見えないので、もっとわかりやすくしてみましょう。
始める前に、スピーカーを D1 に接続してください。

チャレンジ課題

プログラム「LedAroundACC」をかきかえ、スピーカーからも加速度に応じて変化する音が出るようにしてみよう！

ヒント

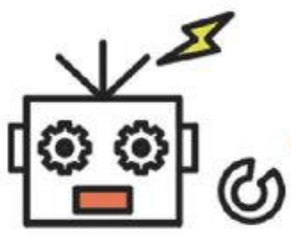
スピーカーから音を出すためのプログラムは覚えているかな？
`tone1.play([周波数], [時間]);` だったね。スピーカーという新機能を使うので、やはり `#include` を増やす必要があるよ。これまでのプログラムの中で、スピーカーを使っていたものを見直して、必要なヘッダーファイル名などを確認してみよう！

講

解答例は以下のプログラムです。
RoboticsProfessorCourse2 > Inverted1 > AccelTone

３. まとめ（目安５分）

今回は姿勢検出（しせいけんしゅつ）シールドの「加速度センサー」について勉強しました。加速度の理解は深まりましたか？ 勢いで発生する加速度、地球からずっと引っ張られている重力加速度、目には見えませんがいつもみなさんが体感している物理量です。
目に見えない力を数字を利用して使えるようになると、ロボットがさらに進化します。自分の傾（かたむ）きがわかったり、押された勢いがわかったりしたら、ロボットは転ばなくなりますよね。
この性質は、このあと作る倒立振子（とうりつしんし）ロボットに生かされます。
次回はさらに「角速度」に関して勉強していってみましょう。



加速度センサーを使った実験、どうだった？

次回は角速度を体感しよう！

講

○　以下の理解度を確認します。
- 姿勢検出シールドを使ってみる
- 加速度を知る
- 加速度を体感する

○　次回のテーマは「角速度を学ぶ」であることを告知して下さい。

≪次回必要なもの≫

・今回と同じパーツを使います。