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

二足歩行ロボット③

（第５回/第６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５回授業日 2024年　　月　　日

第６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3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

二足歩行ロボット③

第5回

二足歩行ロボットのプログラミング②

講師用

目　　次

0.　二足歩行ロボットのプログラミング②

0.0.　「二足歩行ロボットのプログラミング②」でやること

0.1.　必要なもの

1.　サーボモーターの機能を組み込む

1.0.　サーボモーターの機能を組み合わせる（1）

1.1.　マイコンについて詳しく知る

1.2.　サーボモーターの機能を組み合わせる（2）

1.3.　プログラムの記述内容を整理する

1.4.　二足歩行ロボットの機能統合の制御

2.　まとめ

○　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。

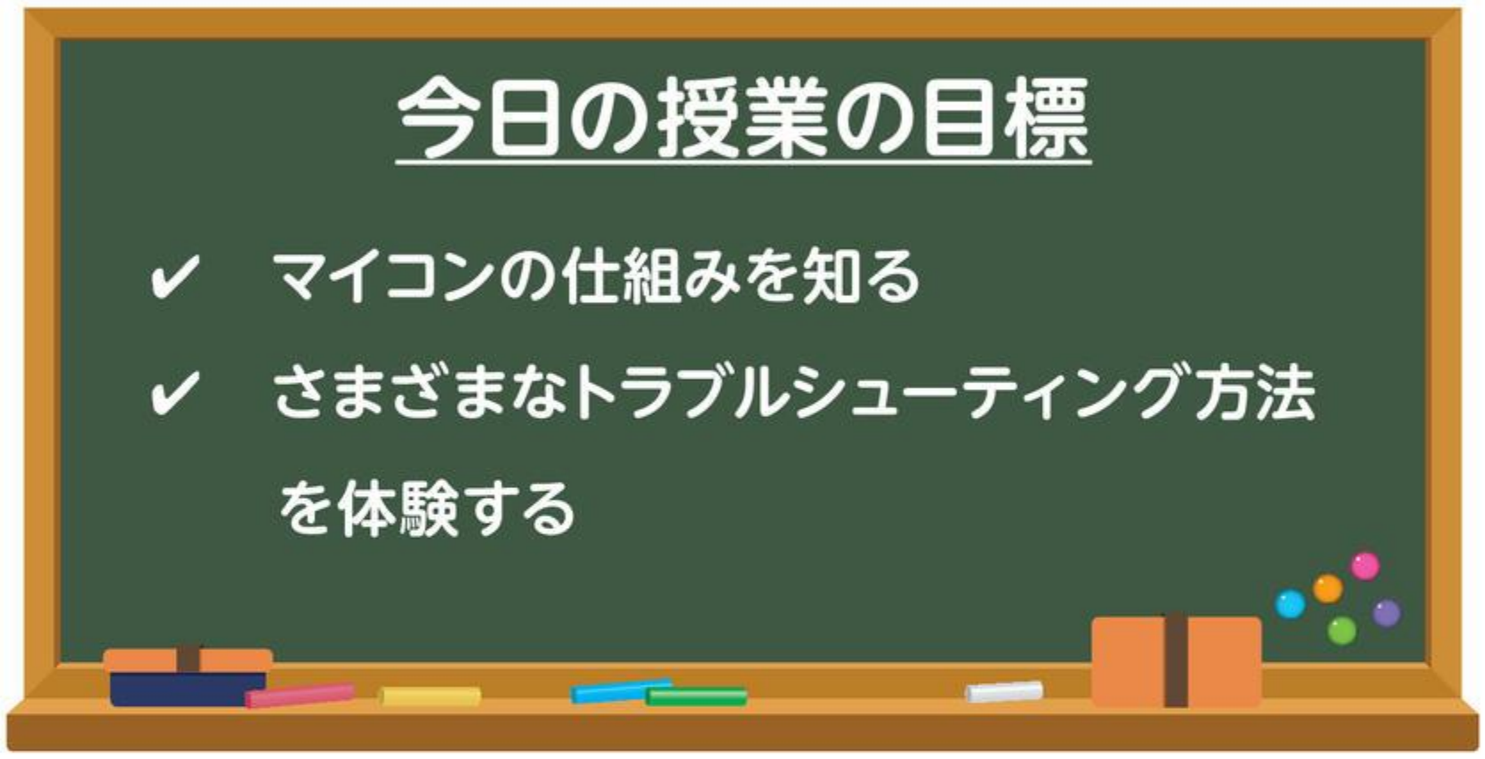
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（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）

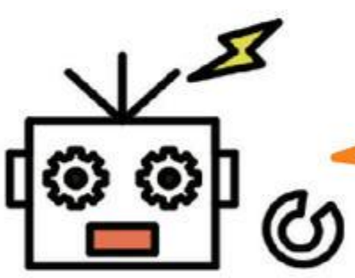
目安時間は授業時間 120 分のうち、休憩 10 分程度取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0.　二足歩行ロボットのプログラミング②（目安５分）

0.0.　「二足歩行ロボットのプログラミング②」でやること



今回の授業では、二足歩行ロボットに、いろいろな機能を順番に搭載させていき、ロボットをつくり上げる過程を学びます。前回まではいろいろな機能をプログラムに統合していきましたが、今回は、ロボットの関節を動かすサーボライブラリを統合します。
第4回は機能（ライブラリ）を追加する度に出てくるさまざまな問題を解決しましたが、今回はどうなるでしょうか?!
さらにロボプロの授業で使っているマイコンボードの構造や機能についても紹介します。
今まで何気なく使っていましたが、どのような仕組みになっているのでしょうか。



0.1. 必要なもの

前回使った「二足歩行ロボット」とコントローラーを使用します。
また、電源はACアダプターを使うので準備しましょう。

1．サーボモーターの機能を組み込む（目安100分）

1.0. サーボモーターの機能を組み合わせる (1)

前回は、超音波距離センサー、マトリクスLED、スピーカー、コントローラーを同時に使えるようなプログラムをつくりました。
二足歩行ロボットを完成させるには、ここにサーボモーターの機能を統合し、コントローラーの各ボタンに関連付けをする必要がありますね。他にも、少し便利な機能を追加したりしてみましょう。

プログラムの書き込み

RoboticsProfessorCourse3 > BiRobot5 > BiRobotTest0_lib

プログラムをマイコンボードに書き込もうとするとエラーが出ます。このエラーは後ほど解消するとして、まずはプログラムのつくりを見ていきましょう。
このプログラムは、前回の最後のプログラム「USSDispToneRemoteTest0_lib」と腕のサーボモーターの調整用に使ったプログラム「ArmOrigin0」とを組み合わせたようなつくりになっています。
追加されたプログラムは次のようになります。

プログラム「BiRobotTest0_lib」より抜粋

```
#include <Servo.h>  // サーボモーターを使うためのライブラリ（設定ファイル）

#include  "SpriteData.h"

NewPingKAI mySonar(US2, US2, 200);              // 超音波距離センサーの初期化
Matrix  myMatrix = Matrix(11, 13, 1);           // マトリクスLEDの初期化
Tone    myTone;
PS2X    myPS2X;
Servo   servoLF, servoRF, servoBD, servoLA, servoRA;  // サーボモーターを管理する機能
```

まず、サーボモーターの機能を追加するため、サーボモーター用のライブラリ「Servo.h」を取り込みます。
LF、RF、BD、LA、RAはそれぞれLeft Foot（左足）、Right Foot（右足）、Body（胴体）、Left Arm（左腕）、Right Arm（右腕）の略で、S0～S4の各サーボモーターに対応しています。

次に、それぞれのサーボモーターの初期位置の設定を行います。このとき、連続して一度に初期位置を設定してしまうとすべてのサーボモーターが同時に大きく動き出すため、電源電圧が低下して、マイコンの働きが不安定になることがあります。
対策として、サーボモーターが動き出すタイミングを少しずつずらします。
プログラム「BiRobotTest0_lib」では、S0 ～ S4 まで５つのサーボモーターを0.1秒間隔でずらしています。

プログラム「BiRobotTest0_lib」より抜粋

```
    servoLF.attach(S0);   //servoLF と名づけたモーターを S0 に接続
    servoLF.write(90);    //servoLF の目標位置を 90 度  に設定
    delay(100);

    servoRF.attach(S1);   //servoRF と名づけたモーターを S1 に接続
    servoRF.write(90);    //servoRF の目標位置を 90 度  に設定
    delay(100);

    servoBD.attach(S2);   //servoBD と名づけたモーターを S2 に接続
    servoBD.write(90);    //servoBD の目標位置を 90 度  に設定
    delay(100);

    servoLA.attach(S3);   //servoLA と名づけたモーターを S3 に接続
    servoLA.write(90);    //servoLA の目標位置を 90 度  に設定
    delay(100);

    servoRA.attach(S4);   //servoRA と名づけたモーターを S4 に接続
    servoRA.write(90);    //servoRA の目標位置を 90 度  に設定
    delay(100);
}
```

あとは、コントローラーが操作されたとき、対応するサーボモーターを回転させるような命令を追加してあげるだけですね。

このプログラムはひとまず「アナログスティックを上下に倒すと、腕のモーターが回転する」という命令を追加しています。

プログラム「BiRobotTest0_lib」より抜粋

```
int posLA, posRA;                    // 左右の腕のポジション
int ry, ly;                          // アナログ右スティック、左スティック

myPS2X.read_gamepad();               // コントローラーから値を読み取る

ry = myPS2X.Analog(PSS_RY);          // アナログ右スティックのたて位置 (-128 ～ +128)
ly = myPS2X.Analog(PSS_LY);          // アナログ左スティックのたて位置 (-128 ～ +128)

if (( -3 < ry ) && ( ry < 3 )){
    ry = 0;                          // アナログ右スティックのたて位置が -2 ～ 2 の間の
                                        場合は 0 に差しかえる
}
if (( -3 < ly ) && ( ly < 3 )){
   ly = 0;                           // アナログ左スティックのたて位置が -2 ～ 2 の間の
                                        場合は 0 に差しかえる
}
posRA = 90 + ry;                     // アナログ右スティックの上下方向真中の位置を
                                        90 にした角度司令値に変換する
                                     // -38 ～ 218 の値になる
posLA = 90 + ly;                     // アナログ左スティックの上下方向真中の位置を
                                        90 にした角度司令値に変換する
                                     // -38 ～ 218 の値になる

servoLA.write(posLA);                // 角度司令値を servoLA に反映させる
servoRA.write(posRA);                // 角度司令値を servoRA に反映させる
```

たとえば ry が-60になるまでスティックを倒せば posRA は90-60＝30となるため、右腕モーターが30度になるまで回転します。

ところで、サーボモーターの回転角度は0度～180度の範囲しか指定できなかったはずです。しかし、posLA や posRA は、ry や ly の値を考えると-38 ～ +218の範囲となってしまうので、スティックを倒しきるまえに限界になってしまうことがわかりますね。この問題も、後ほど工夫して解消しましょう。

また、このプログラムを書き込もうとすると、以下のような文がメッセージエリアに表示され実行することができません。

```
Servo¥Servo.cpp.o: In function `__vector_11':
C:¥Program Files (x86)¥Arduino¥libraries¥Servo/Servo.cpp:103:
multiple definition of `__vector_11'
Tone¥Tone.cpp.o:C:¥Program Files (x86)¥Arduino¥libraries¥Tone/Tone.
cpp:421: first defined here
```

前回も、似た文面のエラーメッセージを見たはずです。覚えているでしょうか？
「Tone」ライブラリと「NewPing」ライブラリが、ともに「タイマー 2」の機能を使おうとした結果、__vector_7 というエラーが出ていましたよね。__vector_11 もタイマー機能の取り合いによるものです。

__vector_7 のときは「NewPing」ライブラリそのものを書きかえ、タイマー 2 を使用しないようにして取り合いを解消しました。今回も同じようにすればよいでしょうか。
しかし、タイマー機能を使わないということは、いずれかの機能が使えなくなる事を意味します。せっかく取りつけたパーツなのですから、すべて活用したいですよね！

そこで、まずは各パーツのタイマー機能の使い方を調べ、取り合いが発生しなくなるようにうまく「仲直り」させられる方法がないか、探っていきましょう！

講

マイコン（Arduino）でなんらかの時間的に正確な処理をしたい時にArduinoのタイマーライブラリを利用します。Arduinoのタイマーライブラリは Timer1とMsTimer2があり、これらのライブラリを利用すると、一定時間ごとに関数を「割り込み」で呼び出すことができます。

1.1. マイコンについて詳しく知る

いつもはロボプロシールドの陰(かげ)に隠(かく)れてほとんど見ることはありませんが、**図1-0**の矢印で示したところがマイコンです。
このマイコンは、足が28本生えた、ATMEL社の「ATMEGA328P」という名前のチップで、「AVRマイコン」というシリーズです。

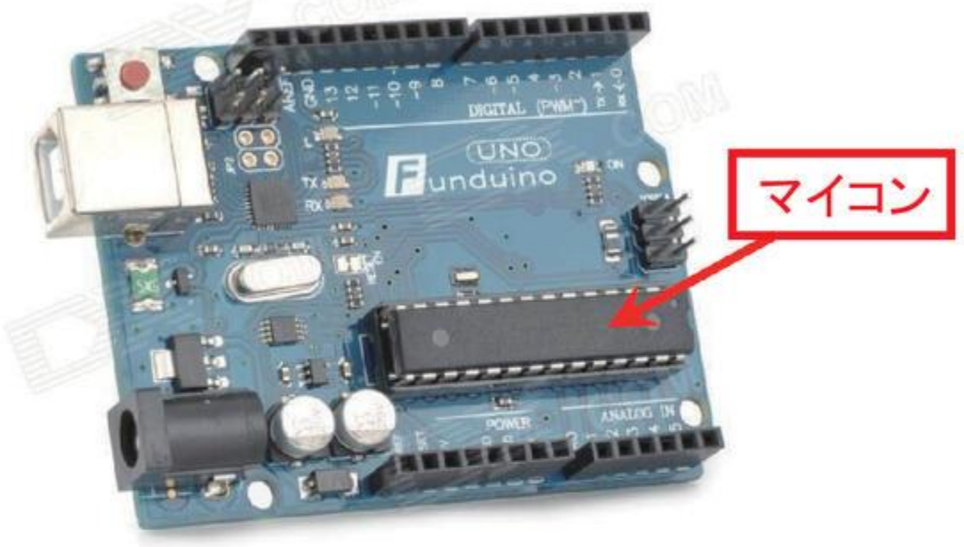


図1-0　マイコンボード

図1-1がマイコンの中身です。「ATMEGA328P」を含む、「AVRマイコン」のブロック図になります。

講　ブロック図は参考程度に確認してください。ブロック図のいくつかの機能を次ページ以降で説明します。

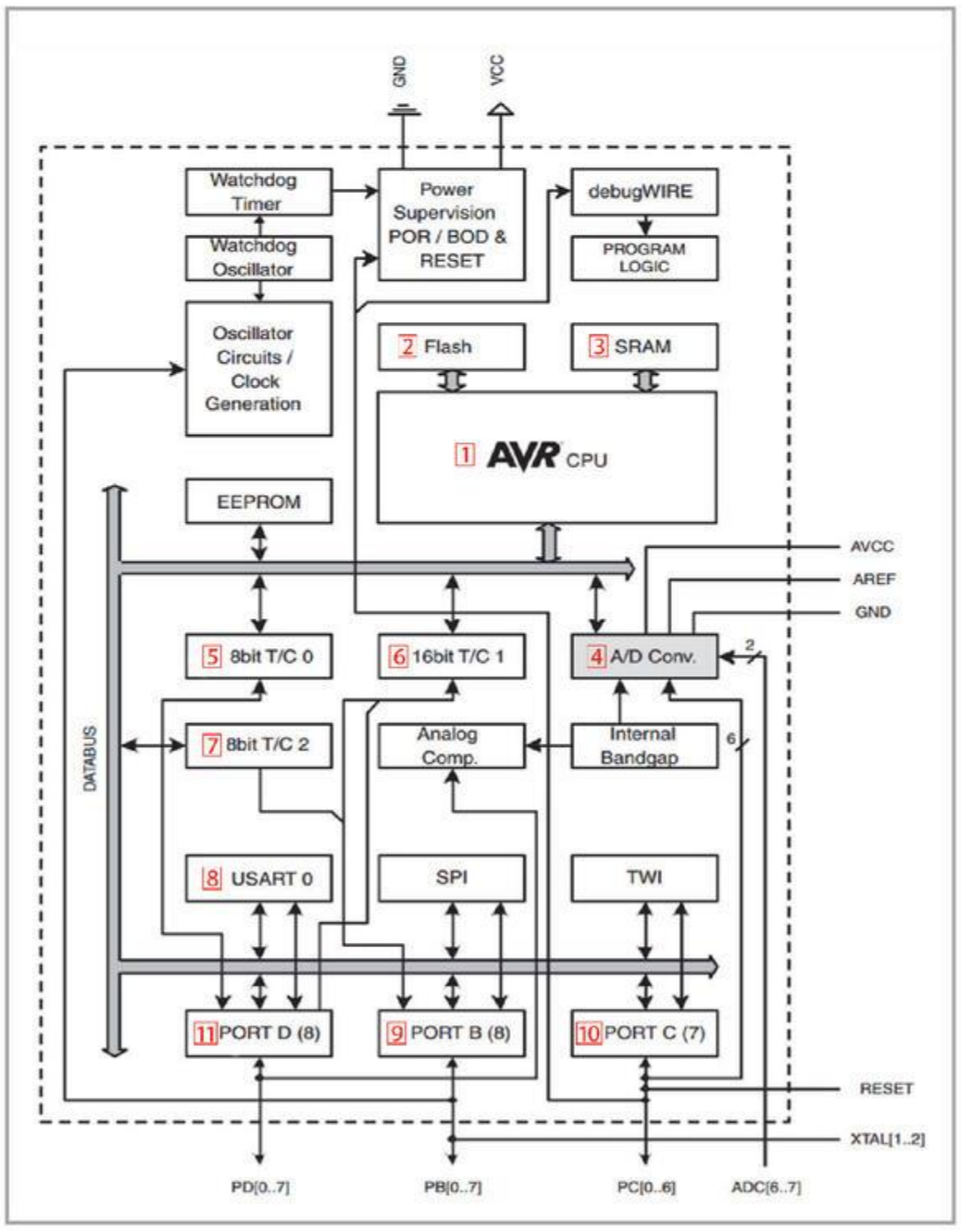


図1-1　AVRマイコンのブロック図

講　巻末にマイコンのブロック図の拡大版を用意してあります。
必要であればご利用ください。

図 1-1 のなかで、5 ～ 7 の「○○ bit T/C」という名前のブロックがタイマー 0 ～ 2 にあたります。
タイマーは文字通り「時計」に近いもので、「モーターの回転を 2000 ミリ秒継続する！」「ロボットの動作中、100 ミリ秒に 1 回のペースでこの処理を行う！」といった、時間に関する命令を処理する機能です。

かんたんに説明すると、一定のペースで回路から発せられる電気信号（クロック信号）を数えて、処理開始からどれくらいの時間が経ったかをカウントしてくれるようなものです。

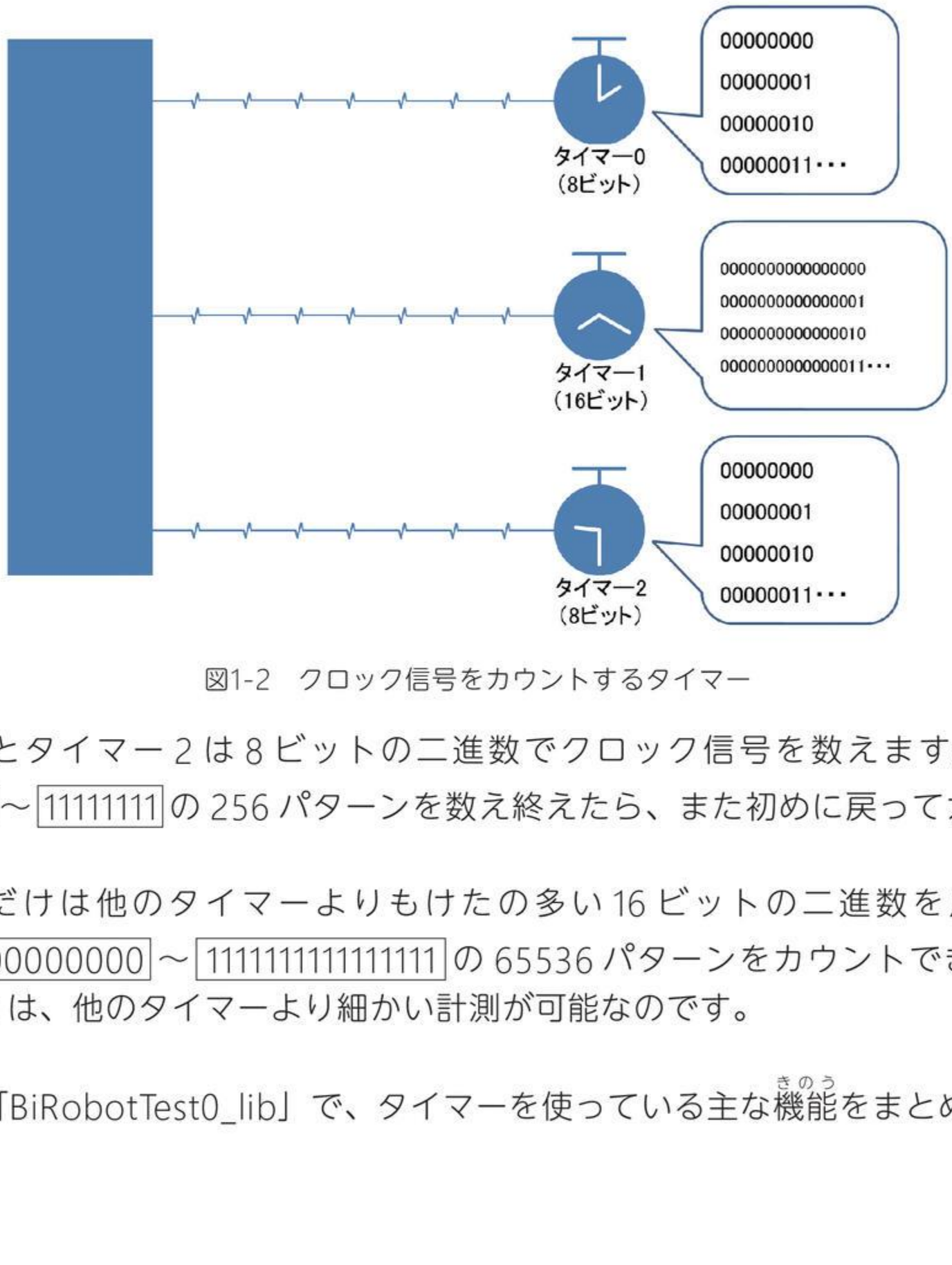


図1-2　クロック信号をカウントするタイマー

タイマー 0 とタイマー 2 は 8 ビットの二進数でクロック信号を数えます。そのため、00000000 ～ 11111111 の 256 パターンを数え終えたら、また初めに戻ってカウントを再開します。
タイマー 1 だけは他のタイマーよりもけたの多い 16 ビットの二進数を用いるので、0000000000000000 ～ 1111111111111111 の 65536 パターンをカウントできます。つまりタイマー 1 は、他のタイマーより細かい計測が可能なのです。

プログラム「BiRobotTest0_lib」で、タイマーを使っている主な機能をまとめます。

POINT

① Arduinoシステム
ロボットに搭載されているパーツや機能に関係なく使われる、大本の時間管理です。
`delay();` などは、この機能を利用している代表的な命令です。
タイマー0は基本的にArduinoシステムが独占しているため、もし他の機能でタイマー0を使用してしまうと、`delay();` 命令などが無効になってしまいます。

②「NewPing」ライブラリ（超音波距離センサーのライブラリ）
本来はタイマー2を使用するライブラリですが、前回の授業で「Tone」ライブラリとタイマーの取り合いになってしまったので、タイマー2を使わないで済む「NewPingKAI」ライブラリをつくりましたね。
この場合、「NewPingKAI」ライブラリはタイマー0のArduinoシステムによる時間管理の一部を、タイマー2の代わりとして使います。

③「Tone」ライブラリ（スピーカーのライブラリ）
「NewPing」ライブラリとタイマー2の取り合いをしていましたが、そちらを「NewPingKAI」ライブラリにつくり直したことでタイマー2を独占できるようになりました。
ただし、Toneライブラリはもともと、スピーカーを最大3つ接続し、3つバラバラに演奏ができるよう、念のためすべてのタイマーをタイマー2、タイマー1、タイマー0の順で使用できるようなつくりになっています。

④「Servo」ライブラリ（サーボモーターのライブラリ）
1つの制御をおよそ10～20msという長い時間かけて行います（マイコンにとっては長時間なのです！）。制御が終わる前にカウントがリセットされてしまうと困るので、カウントできる数が多いタイマー1を使用します。

やってみよう！

各種機能の説明を読んで、どのライブラリが原因でタイマーの取り合いが発生しているか調べてみよう！
また、ライブラリの内容をどのように変更したら取り合いが解消できるか考えてみよう！

問題のライブラリ：Tone ライブラリ

変更内容：タイマー 2 以外を使用しないようにする

チャレンジ課題

自信のある人は直接「Tone」ライブラリを見て、どこを書きかえたらよいか考えてみよう！
実際に書きかえて試してみてもいいね！　書きかえるべきファイルは「Tone.cpp」だよ！

講

プログラム「BiRobotTest1_lib」内の「ToneKAI.cpp」が書きかえ後のライブラリになります。90行目前後と420 ～ 440行目あたりに書きかえた部分がありますので、もし書きかえに挑戦している生徒がいたら確認してあげてください。
なお、ライブラリファイルは基本的にCドライブ内「Program Files」または「Program Files (x86)」→「Arduino」→「libraries」と辿っていくと見つかります。

1.2. サーボモーターの機能を組み合わせる（2）

以下のプログラムの中身を確認していきます。まずは、プログラムを実行してみましょう。

プログラムの書き込み

RoboticsProfessorCourse3 > BiRobot5 > BiRobotTest1_lib

実行結果：二足歩行ロボットの胸のマトリクスLEDでハートが脈動し、それに合わせてスピーカーから音が流れる。さらに、超音波距離センサーに手をかざすと、バーグラフがマトリクスLEDに表示される。そして、コントローラーの左右のアナログスティックを上下に動かすと、左右の腕が上下に動く。

これで、やっと二足歩行ロボットに搭載したすべての機能が一通り動かせるようになりましたね。しかし、このプログラムでロボットを操作してみると、少し反応が悪いように感じませんか？

実はこのプログラムでは、すべての命令の処理をメインループの `loop()` で実行していますが、このメインループはマトリクスLEDの表示サイクルを基準につくられているため、最後に250ms（0.25秒）の `delay();` が入っています。つまり、サーボモーターも1秒に4回しか動かないのです。

`delay();` の数字をもっと小さくすると反応は良くなりますが、ハートの点滅がどんどん速くなって、なんだか落ち着かない様子になってしまいます。

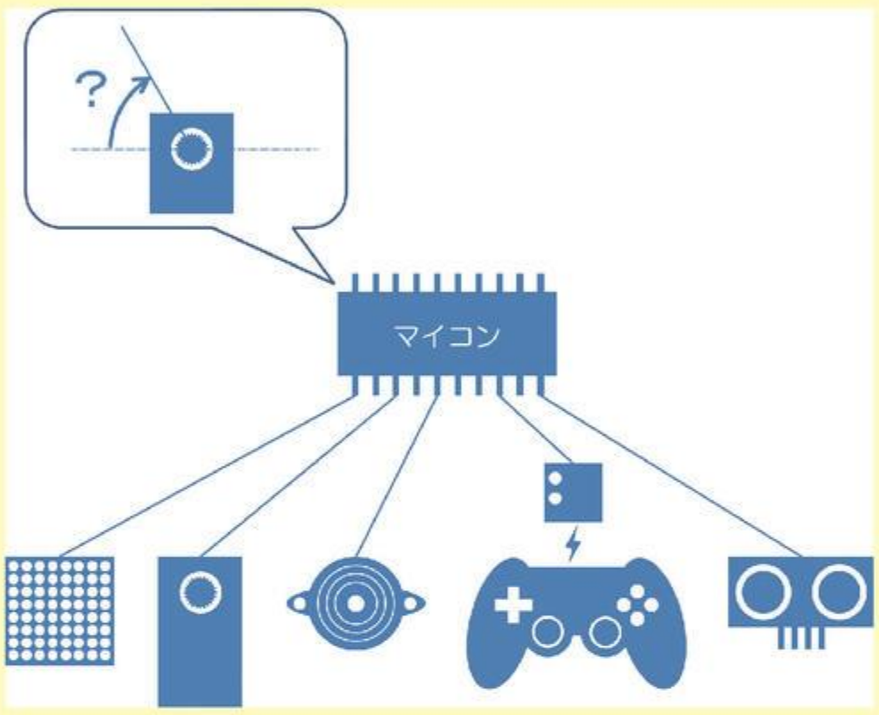
次は、この点を改良していきましょう。

1.3.　プログラムの記述内容を整理する

プログラム「BiRobotTest1_lib」のメインループ内には、大まかに分けて次のような処理が書かれています。

POINT

① コントローラーのボタン状況を読み取る。
② サーボモーターの回転軸を何度に回転させるかを決め、変数に入れる。
③ サーボモーターの目標位置を、②で求めた変数と同じ値にする。
④ 超音波距離センサーで障害物までの距離をはかる。
⑤ マトリクスLEDをいったん消した後、ハートやバーを表示する。
⑥ 表示するハートの大きさを切り替え、スピーカーから効果音を出す。



実際の動作がややぎくしゃくしているのは、①～⑥の動作を全て250ミリ秒おきに行っているせいでしたね。

プログラム「BiRobotTest1_lib」より抜粋

```
void loop(){

(中略)

    delay(250);
}
```

図1-3　プログラム「BiRobotTest1_lib」の各処理のタイミング

しかし、それぞれの動作にかけるべき時間はまちまちです。ですから、短時間でできる動作と時間がかかる動作とでループの間隔を分けることができれば、より反応のいいロボットにできますね。

1.4. 二足歩行ロボットの機能統合の制御

今回は「BiRobotTest1_lib」の各動作を、ループ間隔が小さい（20msで1ループ）もの、中くらい（100msで1ループ）のもの、大きい（300msで1ループ）ものの3種類に分けてみましょう。

POINT

・20msごとにループするもの
　③ サーボモーターの回転軸を、②で求めた変数へ向け回転させていく。

・100msごとにループするもの
　① コントローラーのボタン状況を読み取る。
　② サーボモーターの回転軸を何度に回転させるかを決め、変数に入れる。
　④ 超音波距離センサーで障害物までの距離をはかる。
　⑤ マトリクスLEDをいったん消した後、ハートやバーを表示する。

・300msごとにループするもの
　⑥ 表示するハートの大きさを切り替え、スピーカーから効果音を出す。

③については、モーターの動作はこまめに行わないと動きがぎこちなくなってしまいます。そのため、動作間隔を短くしてしまいましょう。
反対に、⑥はあまり高速でハートがアニメーションしても人間の目が追えませんよね。そのため、すこし間隔を長めにしておきます。
大まかな処理内容はつかめたでしょうか。
では、プログラムを書きかえてみましょう。

ステップアップ

プログラム「BiRobotTest1_lib」を書きかえ、それぞれの動作が適切な間隔（かんかく）でループするようにしてみよう！

ヒント

`void loop()` を１周するのに、何 ms かかるようにすべきかをはじめに考えておこう！

これまでのタームで `loop()` 以外のくり返し命令も学んだし、ループ調整用の変数をつくってみるという手もあるね。やり方は人それぞれなので、まずは自分の得意なやり方でやってみよう！
できたら他にもっと効率のいい書き方がないか探してみたり、それぞれの動作を関数でまとめてみたりして、より洗練（せんれん）されたプログラムを目指してみよう！

ちなみに、オリジナル関数をつくりたいときは `loop()` のカタマリの外に `void 関数名(){ 動作内容 }` とかくんだったよね！

講

解答例は以下のプログラムです。
RoboticsProfessorCourse3 > BiRobot5 > BiRobotTest2_lib

これは `void loop()` 1周を20msとし、変数 `loopCounter` を1ずつ増加させていくことで、ループ5回（100ms）ごとに1回だけ動作を実行するようなつくりになっています。
あわせて、変数 `loopSubCounter` を併用することで、100msループ3回（300ms）ごとに1回だけ行う処理も作ってあります。

また、5ループ毎に超音波距離センサーで計測を行いますが、超音波を発してから戻ってくるまでの間プログラムが進行しません。
このときも1ループがちょうど20ミリ秒になるよう、超音波が往復した時間を20ミリ秒から差し引いています（変数 `usec` を1000で割っているのは、`usec` の単位がマイクロ秒であるためです）。

`void loop()` 1周を300msとして、for文などを活用して同様のループをつくるという手もあります。大まかにみればほぼ同じ動作となりますので、プログラムのつくりによらず予定通りの動き（アニメーションの速度を維持したまま、動作がスムーズになる）になっているかどうかで判定してあげてください。

コラム　マイコンに搭載された各種機能

さきほど「AVRマイコン」のブロック図を紹介しましたが、数ある機能のうち「タイマー」にしか触れていませんでした。
しかし実際には、皆さんの小指より細いようなこの小さなマイコンに、タイマー以外にもさまざまな機能が搭載されています。一通り解説します。

1 CPU(Central Processing Unit：中央演算装置)

マイコンの中枢になる部分です。名前の通り、演算をすることができますが、逆に演算以外は何もできません。そのため、他の機能を補う周辺機能が同じチップ内にたくさん搭載されています。
以前は、マイコンというとCPUのことで、その他の機能は別のチップに分かれていて、マイコンボードをつくるにはたくさんのチップを並べる必要がありました。しかし、半導体の集積技術の進歩により、今ではワンチップに周辺機能をまとめるのがあたり前になりました。

2 Flash（Flash Memory：フラッシュメモリー）

パソコン上で書かれたプログラムがコンパイルされてマイコンに格納できるデータとして、書き込み操作で格納される部分です。名前の通り、一括してフラッシュ（消去）でき、書きかえをすることができるメモリー（記憶領域）です。
一度書き込みをすると、電源を切ってもデータは消えませんが、書きかえができる回数には制限があります。最近は技術の進歩で、書きかえ可能回数が１万回以上になっているので、プログラミングで使用する場合は、ほぼ問題ありません。

3 SRAM（Static Random Access Memory：スタティックRAM）

CPUは、起動するとフラッシュメモリーに格納されたプログラムを読み出して、それに沿って計算を行います。しかし、CPUは計算しかできないので、計算した途中のデータや結果を覚えておく記憶領域が別に必要です。そのための部分がこのスタティックRAMになります。
スタティックとは、静的という意味で、通電している間は、一度書き込まれたデータは保存されます。ただし、電源を切ってしまうと、データは消えてしまいます。

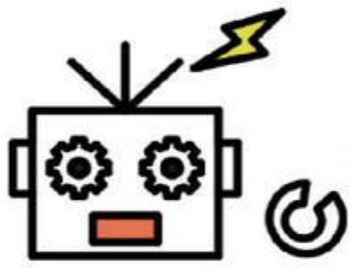
4 A/D Conv.(Analog/Digital Converter：A/D変換器)

コンピューターの世界では、デジタルで「0（LOW：0V)」と「1（HIGH:5V)」しかないわけですが、センサーなどを使うときには０Vでも５Vでもない中間のアナログの値が必要になることがあります。
そのような場合に、アナログの情報をマイコンで扱えるデジタルの情報に変換する装置がA/D変換器になります。明るさセンサーなどを使うときなどに使用した `analogRead（）` は、この周辺機能を使って実行されています。

5 8 bit T/C 0（8bit Timer Controller 0：8 bitタイマー０）
6 16 bit T/C 1（16bit Timer Controller 1：16 bitタイマー１）
7 8 bit T/C 2（8bit Timer Controller 2：8 bitタイマー２）
前回から何かと競合問題を起こしている、タイマーコントローラーです。AVR マイコンでは、8bit タイマー２つと、16bit タイマー１つの、合計３個のタイマーコントローラーを内蔵（ないぞう）しています。タイマーコントローラーは名前の通り、時計のような機能（きのう）で、マイコンから設定を送ると、CPU の処理（しょり）とは別に一定の間隔のパルスを出力したり、一定間隔で CPU に割り込（こ）みをかけたりすることができます。
8 USART 0（Universal Synchronous Asynchronous Receiver Transmitter0：汎用同期式（はんようどうきしき）/非同期式（ひどうきしき）シリアル送受信機（ユニバーサル　シンクロナス　アシンクロナス　レシーバー　トランスミッター））
シリアル通信で、プログラムの動作を電気信号入出力で確認するのを担当しています。
9 PORT　B（ポートB）
10 PORT　C（ポートC）
11 PORT　D（ポートD）
マイコンが外部の回路と信号のやり取りをする窓口が、このポートです。 デジタル信号「０（LOW：０V）」と「１（HIGH:5V）」の入出力を受け付けますが、A/D 変換器を使用する場合にはアナログの入力情報も受け付けることができます。

2. まとめ（目安5分）

今回は、二足歩行ロボットの各機能を動かすプログラムを全て統合することができました。
足とボディのサーボモーターの目標位置更新はまだプログラムされていませんが、原点位置での保持は正常に動作しています。
単独で使うときには問題がなくても、組み合わせるとうまく動かないということはよく起こります。
そんなときにはちゃんとした理由があります。問題を解決するには、知識と経験、そして何よりも、諦めないで問題を解決しようとする探究心が必要です。
次回は、二足歩行ロボットの足とボディも動かし、プログラムを完成させていきます。

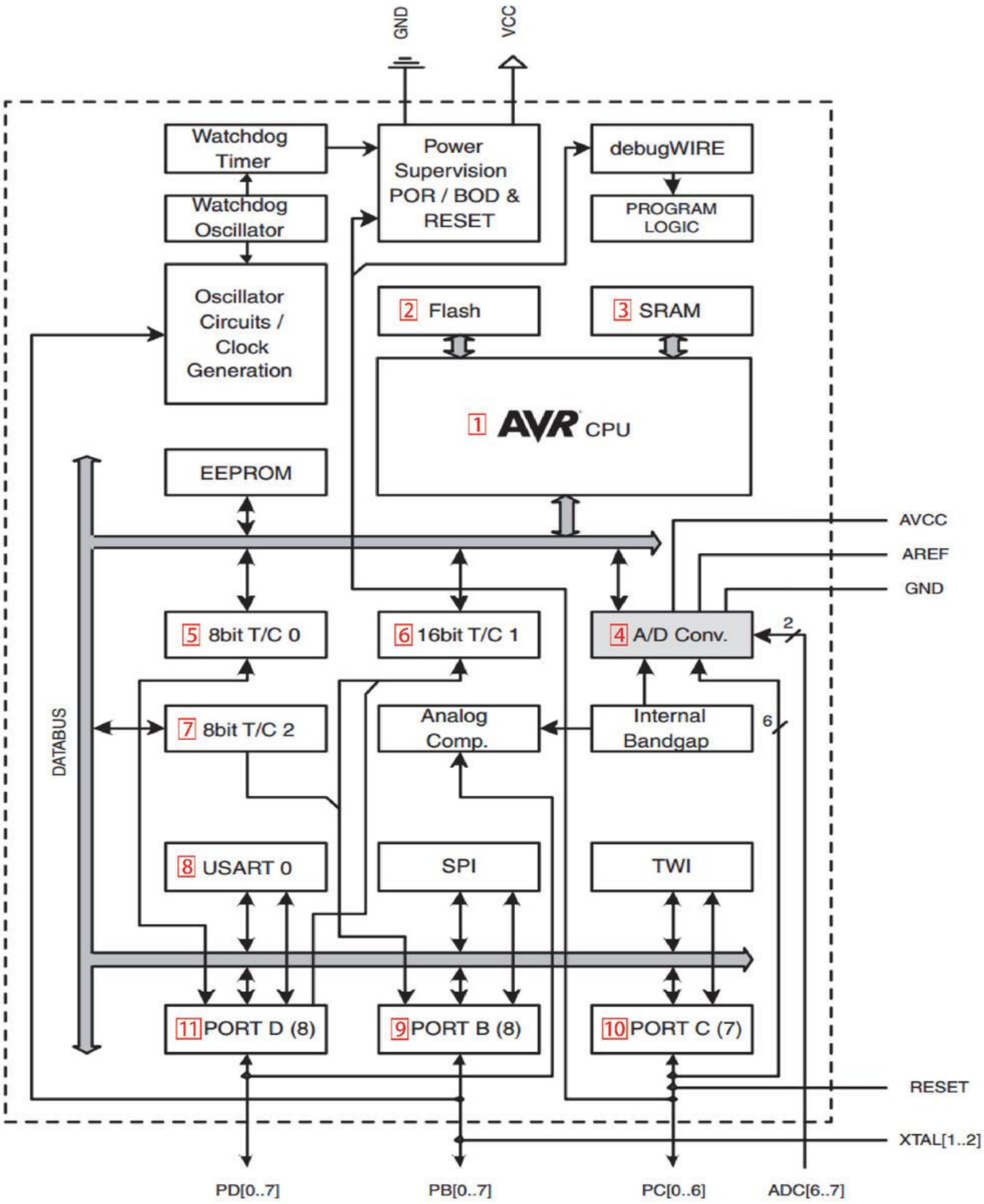


次回は最終回だぞ！ 二足歩行ロボットのすべての機能をフル活用してみよう。

講

〇以下の授業の目標を再確認します。
・マイコンの仕組みを知る
・さまざまなトラブルシューティング方法を体験する

〇次回のテーマは、「二足歩行ロボットのプログラミング③」です。
引き続き、完成した二足歩行ロボットとコントローラー・ACアダプターを持参させてください。
また、サーボモーターの原点位置が大きくずれている場合、持ち帰って補正させることを推奨します。


GND
VCC
Watchdog Timer
Watchdog Oscillator
Oscillator Circuits / Clock Generation
Power Supervision POR / BOD & RESET
debugWIRE
PROGRAM LOGIC
2 Flash
3 SRAM
1 AVR CPU
EEPROM
AVCC
AREF
GND
5 8bit T/C 0
6 16bit T/C 1
4 A/D Conv.
2
DATABUS
7 8bit T/C 2
Analog Comp.
Internal Bandgap
6
8 USART 0
SPI
TWI
11 PORT D (8)
9 PORT B (8)
10 PORT C (7)
RESET
XTAL[1..2]
PD[0..7]
PB[0..7]
PC[0..6]
ADC[6..7]