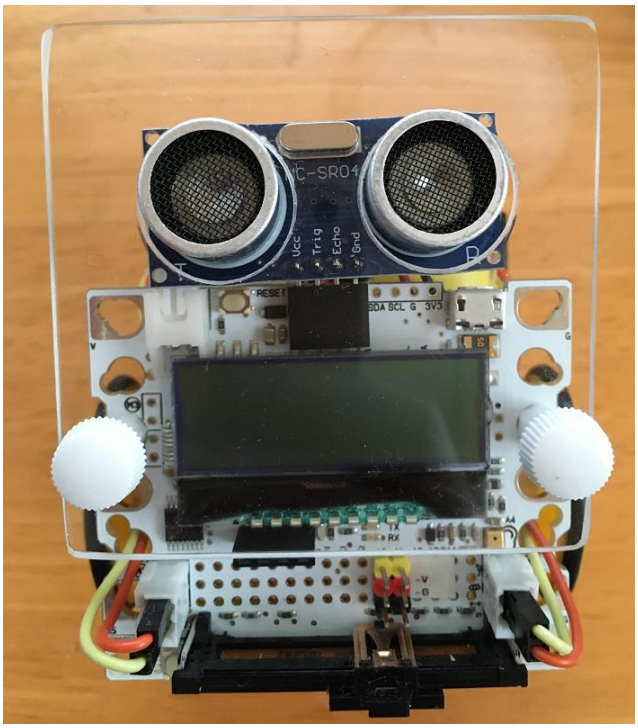
SmallBot 回路マニュアル



2022 年 08 月
PastelMagic

目次

1. SmallBot の入出力 - 6 -
2. モーター制御 - 10 -
2.1. モータ制御のハードウェア - 10 -
2.1.1. モーター駆動回路 - 10 -
2.1.2. モータードライバの入力と基本動作 - 13 -
2.1.3. PWM 入力による回転数制御 - 15 -
2.2. モータ駆動のソフトウェア - 16 -
2.2.1. Arduino の設定とモーターの回転数 - 16 -
2.3. プログラム中のモーター設定とロボットの進行方向 - 17 -
2.3.1. サンプルプログラム中のピン名称設 - 17 -
2.3.2. ピンの初期化 - 18 -
2.3.3. プログラムとロボットの動き - 18 -
3. スイッチ入力 - 23 -
3.1. スイッチ入力のハードウェア - 23 -
3.1.1. スイッチ入力回路 - 23 -
3.1.2. HIGH/LOW 判定 - 24 -
3.1.3. スイッチ入力回路の動作 - 25 -
3.2. スイッチ入力のソフトウェア - 26 -
4. LED - 28 -
4.1. LED 出力のハードウェア - 28 -
4.1.1. LED 点灯回路 - 28 -
4.1.2. LED 点灯回路の動作 - 29 -
4.2. LED 点灯ソフトウェア - 30 -
5. スライドボリューム（可変抵抗器） - 31 -
5.1. スライドボリューム入力のハードウェア - 32 -
5.1.1. スライドボリューム入力回路 - 32 -
5.2. スライドボリューム入力回路の動作 - 32 -
5.2.1. スライドボリューム読み込みソフトウェア - 34 -
5.2.2. map()関数 - 34 -
6. フォトセンサ - 35 -
6.1. SmallBot のフォトセンサのハードウェア - 36 -
6.1.1. フォトセンサ回路 - 36 -

6.1.2. フォトトランジスタの動作 - 37 -
6.2. フォトセンサのソフトウェア - 37 -
7. マイク入力 - 41 -
7.1. マイク入力のハードウェア - 42 -
7.1.1. マイク入力回路 - 42 -
7.1.2. HPF - 42 -
7.1.3. HPF のカットオフ周波数 - 44 -
7.1.4. 増幅回路 - 45 -
7.1.5. オペアンプの基本動作 - 45 -
7.1.6. 増幅回路の動作 - 46 -
7.1.7. 非反転増幅回路の入出力関係 - 47 -
7.1.8. ピーク検出回路 - 49 -
7.1.9. ピーク検出回路の基本動作 - 50 -
7.1.10. ピーク検出回路のシミュレーション - 52 -
7.1.11. マイク入力回路のシミュレーション - 53 -
7.2. マイク入力のソフトウェア - 55 -
8. 超音波距離センサ - 56 -
8.1. 超音波距離センサのハードウェア - 56 -
8.1.1. 超音波距離センサの外観 - 56 -
8.1.2. 超音波距離計測の仕組み - 57 -
8.1.3. 超音波距離計測動作 - 58 -
8.1.4. 超音波距離センサ回路図 - 59 -
8.2. 超音波距離センサソフトウェア - 60 -
9. キャラクタ LCD - 62 -
9.1. キャラクタ LCD のハードウェア - 62 -
9.2. キャラクタ LCD のソフトウェア - 63 -

図 1：SmallBot の入出力デバイス配置.....- 6 -
図 2：SmallBot の入出力接続.....- 7 -
図 3：左右のモーター配置と回転方向.....- 10 -
図 4：モータードライバの周辺回路の接続とプログラム中のピン定義例.....- 11 -
図 5：モータードライバの入力信号と基本動作イメージ（1／2）.....- 12 -
図 6：モータードライバの入力信号と基本動作イメージ（2／2）.....- 13 -
図 7：PWM 入力によるモーターの回転数制御（1／2）.....- 14 -
図 8：PWM 入力によるモーターの回転数制御（2／2）.....- 15 -
図 9：analogWrite()関数の設定値と出力波形、モーターの回転数.....- 16 -
図 10：サンプルプログラム中のピン名称定義.....- 17 -
図 11：モーター関連ピンの初期化.....- 18 -
図 12：モータードライバの設定とロボットの移動方向(1/3).....- 20 -
図 13：モータードライバの設定とロボットの移動方向(2/3).....- 21 -
図 14：モータードライバの設定とロボットの移動方向(3/3).....- 22 -
図 15：SmallBot の押しボタンスイッチ入力回路.....- 23 -
図 16 入力電圧と HIGH/LOW 判定.....- 24 -
図 17：スイッチ入力の動作.....- 25 -
図 18：サンプルプログラムのスイッチ読み込み.....- 26 -
図 19:LED 点灯回路.....- 28 -
図 20：LED 点灯回路の動作.....- 29 -
図 21：サンプルプログラム中の LED 点滅.....- 30 -
図 22：スライドボリュームの基本構造.....- 31 -
図 23：スライドボリューム入力回路.....- 32 -
図 24：スライドボリュームの動作.....- 33 -
図 25：スライドボリュームの Ra の値と Vin の関係.....- 33 -
図 26：サンプルプログラムのスライダ値読み込み部分.....- 34 -
図 27：map()関数による値の変換.....- 35 -
図 28：フォトセンサ入力回路.....- 36 -
図 29：フォトトランジスタの動作.....- 37 -
図 30：サンプルプログラムのフォトセンサ入力.....- 37 -
図 31：ライントレース動作.....- 38 -
図 32：線の左側にロボットがある場合の動作例.....- 39 -
図 33：SmallBot のマイクの位置.....- 41 -

図 34：マイク入力回路 ..... - 41 -
図 35：周波数と HPF 出力レベルの変化 ..... - 43 -
図 36：マイク入力の HPF の特性 ..... - 44 -
図 37：マイク増幅回路 ..... - 45 -
図 38：オペアンプの基本動作 ..... - 46 -
図 39：非反転増幅回路の動作 ..... - 47 -
図 40：非反転増幅回路の増幅動作 ..... - 48 -
図 41：非反転増幅回路の出力波形 ..... - 48 -
図 42：ピーク検出回路 ..... - 49 -
図 43：ピーク検出回路の考え方 ..... - 50 -
図 44：ピーク検出回路の動作 ..... - 51 -
図 45：ピーク検出回路シミュレーション回路 ..... - 52 -
図 46：ピーク検出回路シミュレーション結果 ..... - 53 -
図 47：マイク入力回路シミュレーション回路 ..... - 54 -
図 48：マイク入力回路シミュレーション結果 ..... - 54 -
図 49：サンプルプログラムの音声入力処理 ..... - 55 -
図 50：超音波距離センサの位置 ..... - 56 -
図 51：超音波距離センサの外観 ..... - 56 -
図 52：超音波距離センサモジュールの動作 ..... - 57 -
図 53：超音波距離センサ回路図 ..... - 59 -
図 54：サンプルプログラムの音波距離計測部分 ..... - 60 -
図 55：キャラクタ LCD の位置 ..... - 62 -
図 56：キャラクタ LCD の接続 ..... - 62 -
図 57：サンプルプログラムのキャラクタ LCD 制御（抜粋） ..... - 63 -
図 58：setCursor()の引数と表示開始位置 ..... - 65 -

1. SmallBot の入出力

図 1 は SmallBot の入出録デバイスの配置を示したものです

また、これらの入出力デバイスとマイコンの接続関係をまとめたのが図 2 です。

SmallBot に使われているマイコンボートは RDC-104 Type II で、使用されているマイコンは、Atmel 社の ATMega16U4 というものです。

接続図のピン番号はソフトウェア開発環境である、Arduino-IDE 上で使われるピン番号です。実際のマイコンのピン番号とは異なります

SmallBot には次のような入出力デバイスが接続されています。
それぞれのデバイスについての詳細は後ほど説明します。

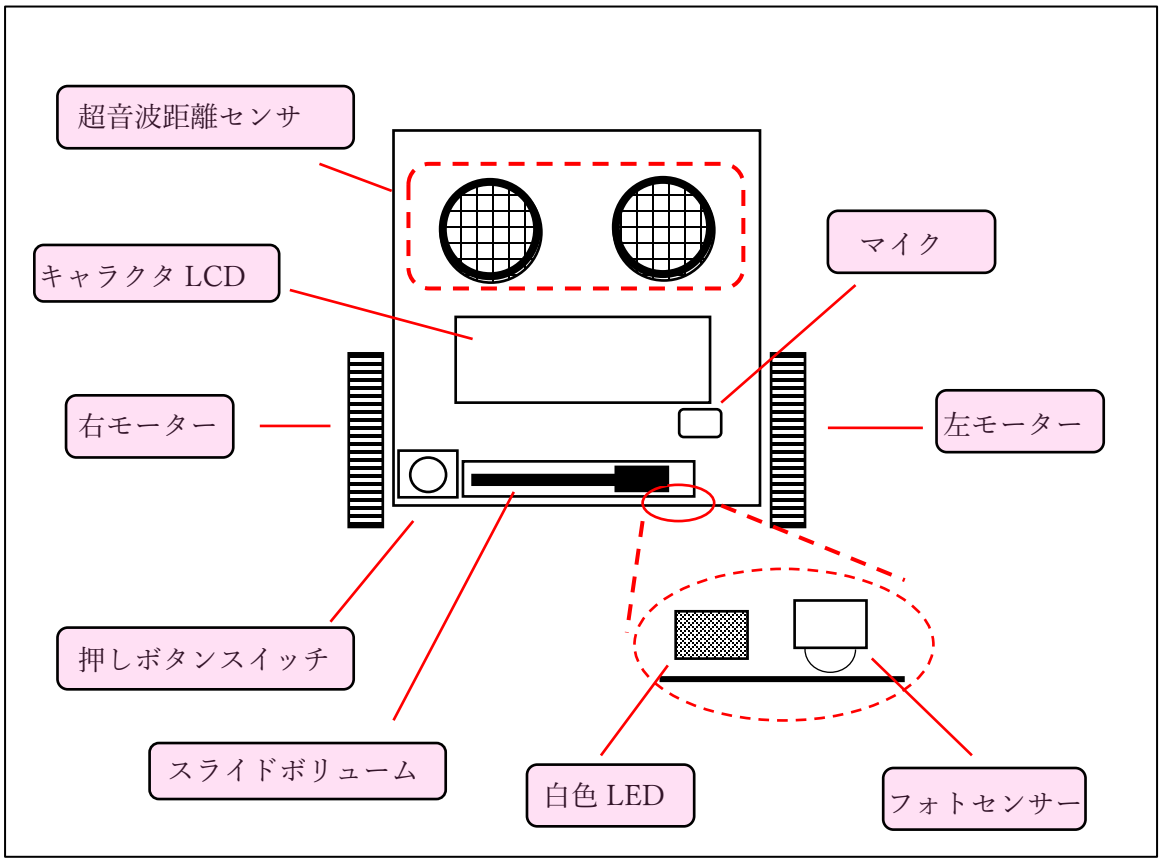


図 1：SmallBot の入出力デバイス配置

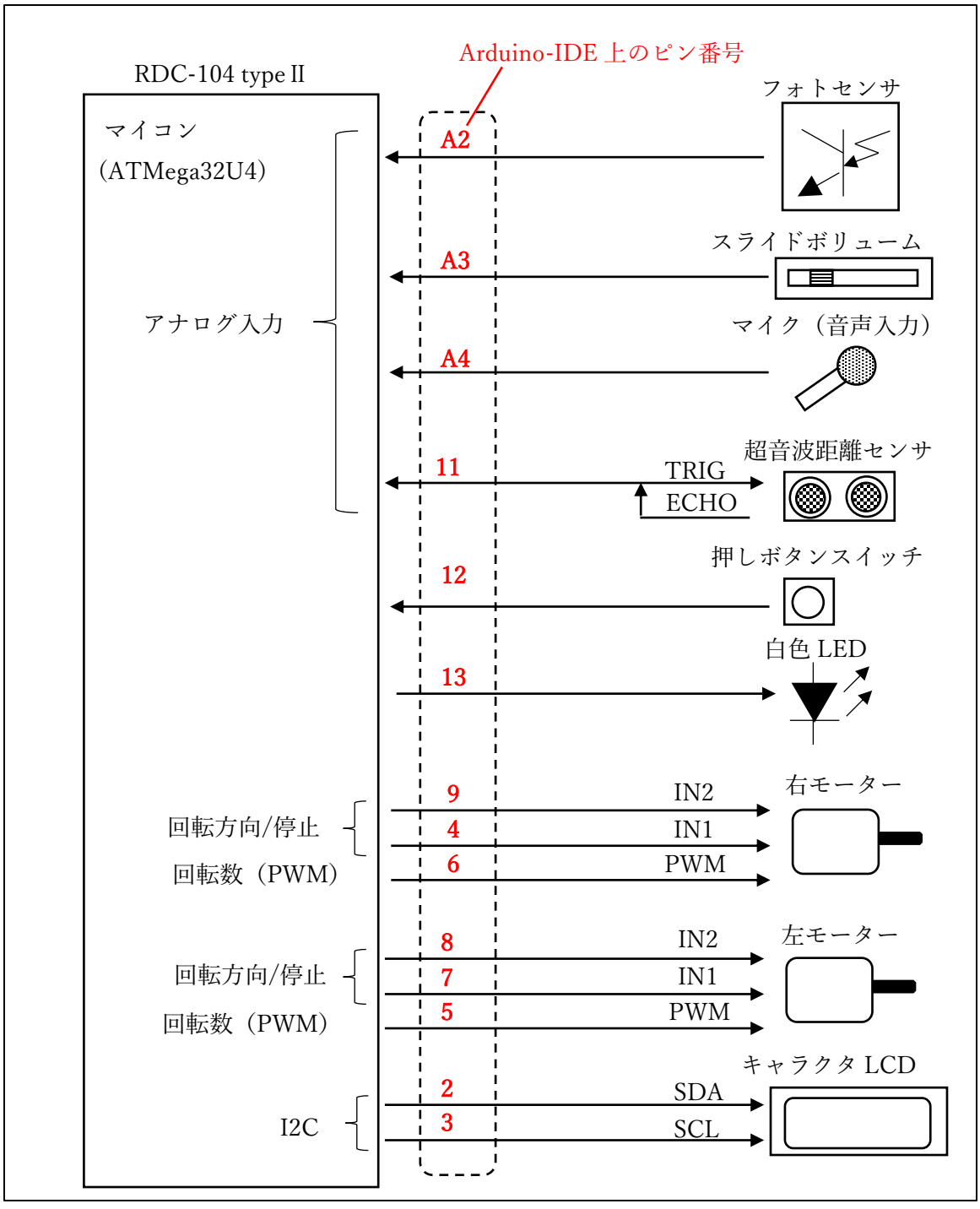


図 2：SmallBot の入出力接続

〇左右モーター

ロボットの前進、後退、旋回などを行うのがロボットの左右に付いているモーターです。

左右のモーターそれぞれを駆動する信号は 3 本あり、3 本のうちの 2 本がモーターの回転方向やブレーキ動作などを指示する信号（IN1/IN2)、1 本がモーターの回転数を制御する信号（PWM）信号です。

〇スイッチ入力

押すと ON、離すと OFF になる押しボタンスイッチが 1 つ用意されています。用途は自由です。サンプルプログラムでは動作モードを切り替えるのに利用しています。

〇LED 出力

ユーザが自由に点滅させることができる白色 LED が 1 つ用意されています。プログラムの動作状態を表示するのにも利用できます。
サンプルプログラムでは動作モードに応じて、消灯、点滅、連続点灯と切り替わるようにしています。

〇スライドボリューム

つまみの位置を左右に動かすことで抵抗値が変化するものです。

SmallBot では本体正面に取り付けられていて、抵抗値の変化によって変化する電圧をマイコンのアナログ入力で読み込むことができるように設計されています。

これによって、マイコンでつまみの位置を知ることができます。

サンプルプログラムでは障害物回避動作に入る距離の調整などに使っています。

〇フォトセンサ

フォトセンサは明るさを検出するものです。SmallBot ではフォトトランジスタと呼ばれる半導体式の光センサが使われています。

フォトセンサを利用して明るさに応じた電圧を生成し、マイコンのアナログ入力で読み取れるようにしています。周囲の明るさをマイコンでしることができるわけです。

SmallBot のフォトセンサは床の明るさを検出するように下向きにとりつけられています。サンプルプログラムではライントレース（床に書かれた線を辿って動く）モードでフォトセンサを利用しています。

〇マイク入力

SmallBot の基板上にはマイクが取り付けられていて、周囲の騒音レベ

ルを計測できるようになっています。音声の取り込みではなく、騒音計のように、大まかな音声の大きさを計測するものです。

計測されたデータの用途は自由です。サンプルプログラムでは手を叩くなどすると方向転換をする動作モード（ダンスモード）でマイク入力を利用しています。

〇超音波距離センサ

超音波距離センサモジュールは、超音波を使って対象物までの距離計測を行うモジュールです。

超音波を発信し、対象物に反射して戻ってくるまでの時間を計測することで距離を計測するという仕組みになっています。

超音波の発信や受信の処理は超音波距離センサモジュール内部で処理されますので、マイコン側では細かい処理は不要です。

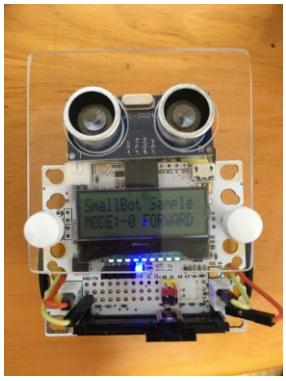
超音波距離センサモジュールには超音波発信を指示する TRIG 入力信号と計測結果を示す ECHO 出力信号の二つの信号線があります。それぞれの信号を別々のピンに接続するのが一般的ですが、に SmallBot ではマイコンのピンの入出力モードを切り替えて利用することで、1 本のピンで超音波距離センサモジュールを動かしています。

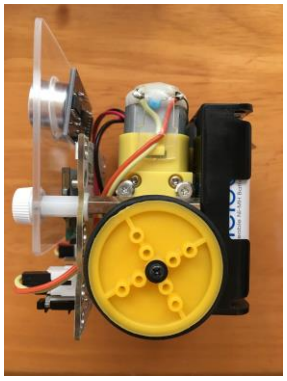
〇キャラクタ LCD

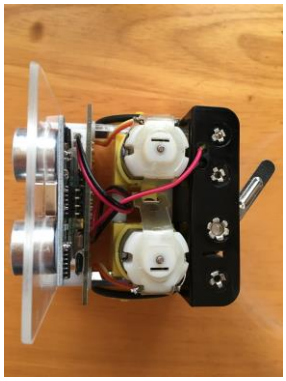
SmallBot には文字や数字を表示できる、キャラクタ LCD モジュールが搭載されています。

表示可能な文字はアルファベットや数字で、表示できる行数は 2 行で、1 行に表示できる文字数は 16 文字です。

サンプルプログラムでは動作モードやロボットの進行方向を表示するのに利用しています。







2. モーター制御

2.1. モータ制御のハードウェア

2.1.1. モーター駆動回路

図 3 は SmallBot の左右モーターの配置、図 4 は SmallBot のマイコンとモーターの接続図です。

ロボットを上から見て進行方向に向かって右側のモーターが右モーター、左側のモーターが左モーターです。

マイコン（ATMega32U4）とモーターの間にはモータードライバと呼ばれる、モーター駆動用 IC があり、マイコンから与えられた制御信号によってモーターが駆動されます。

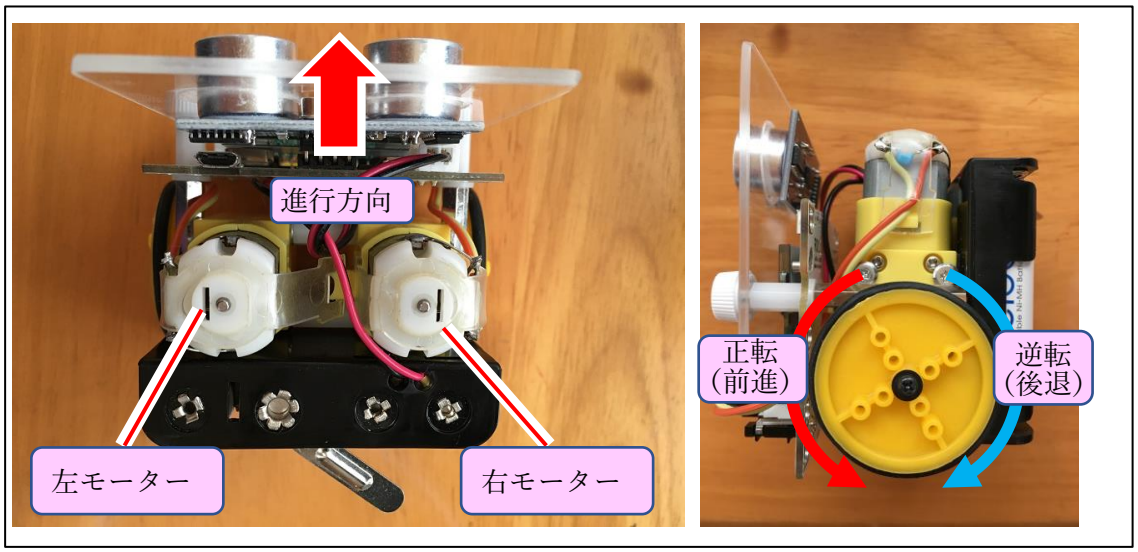


図 3：左右のモーター配置と回転方向

SmallBot で使用されているモータードライバ IC は二つのチャンネル（A チャンネルと B チャンネル）があってモーターを二つ制御できるようになっています。SmallBot では A チャンネルを右側の車輪用のモーター、B チャンネルを左側の車輪用のモーター用として利用しています。

モータードライバには IN1,IN2、PWM という 3 つの入力信号があります。IN1、IN2 入力がモーターの回転方向を決める信号、PWM 入力が回転速度を決める（モーターに与える電力

を制御する）信号です。

サンプルプログラムでは右モーター（A チャンネル）を M1、左モーター（B チャンネル）を M2 を信号名の頭につけることにしています。たとえば、右モーターの信号は M1_1、M1_2、M1_PWM

左モーターの信号は M2_1、M2_2、M2_PWM という具合です。

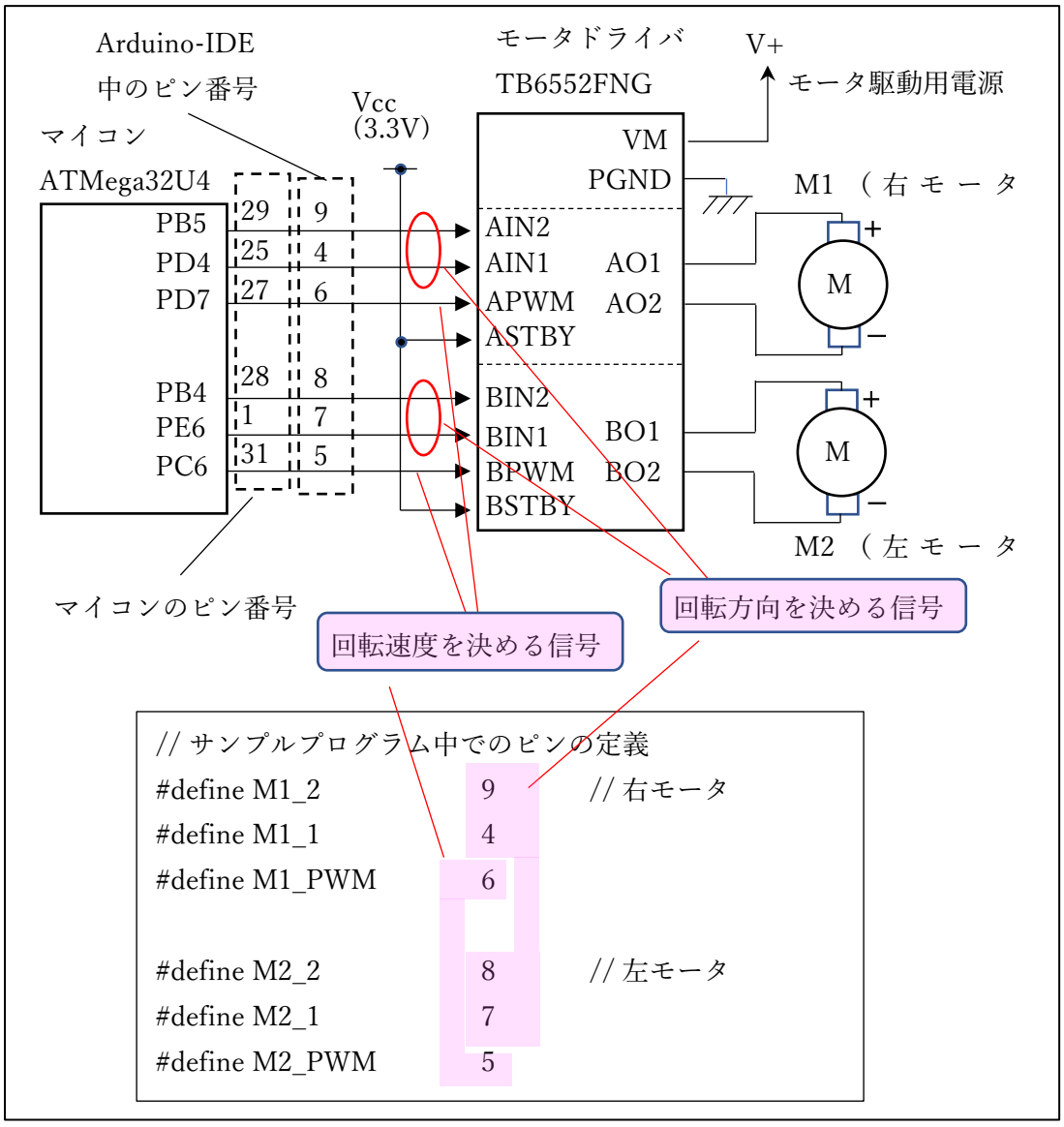


図 4：モータードライバの周辺回路の接続とプログラム中のピン定義例

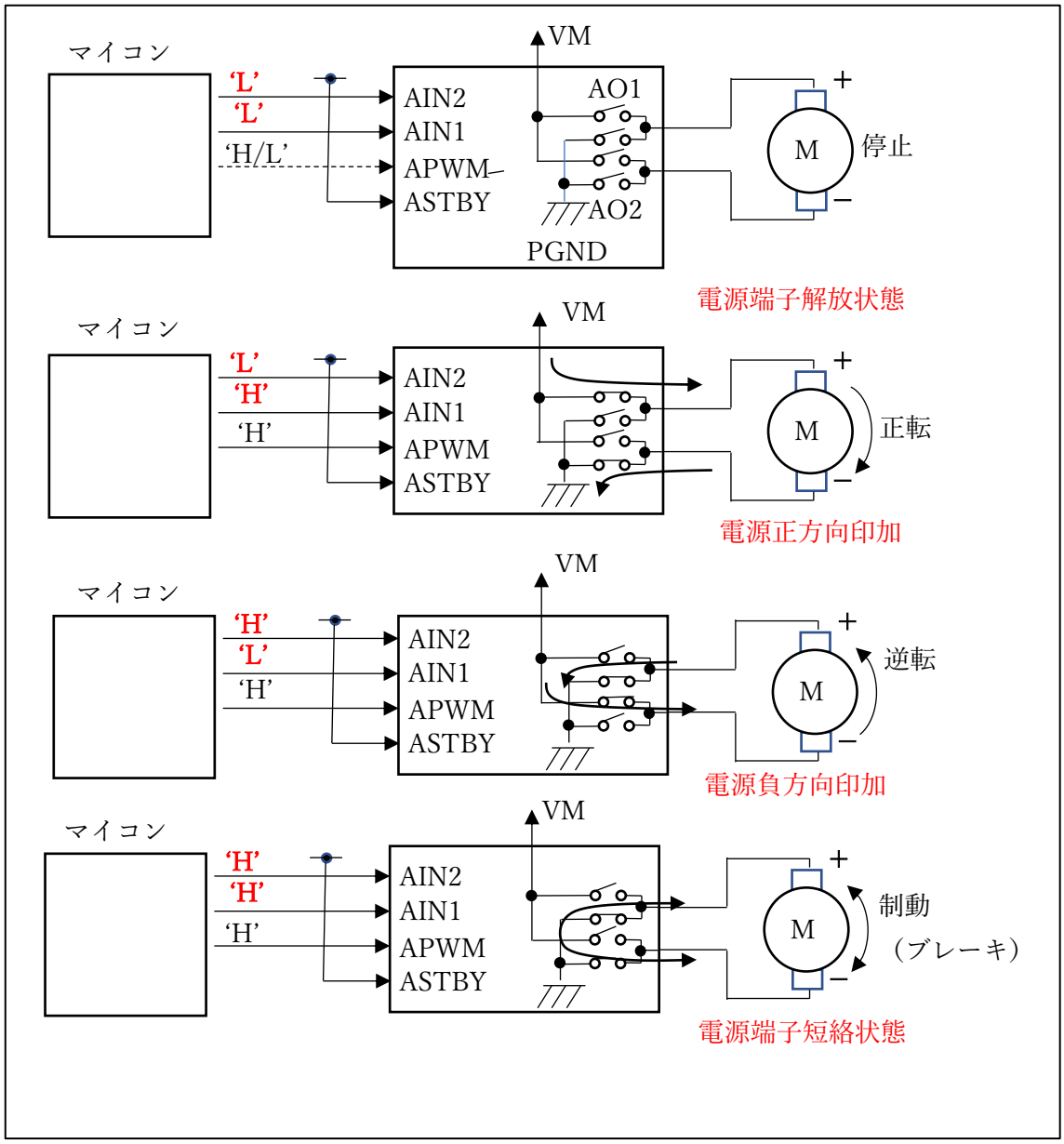


図 5：モータードライバの入力信号と基本動作イメージ（1／2）

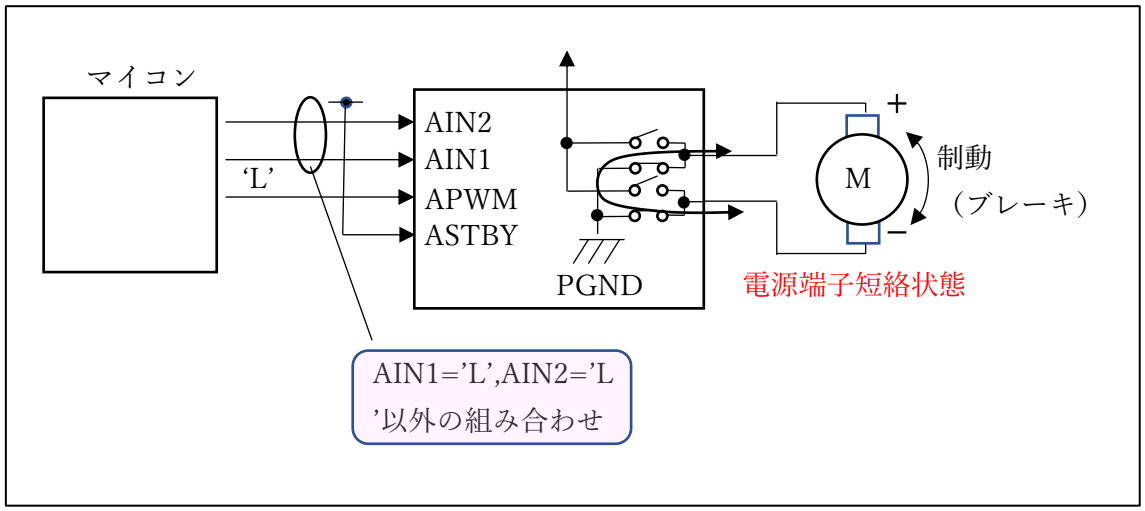


図 6：モータードライバの入力信号と基本動作イメージ（2／2）

2.1.2. モータードライバの入力と基本動作

図 5 と図 6 はモータードライバへの入力信号とモータの駆動の関係を示したものです。

モータードライバの STBY（スタンバイ）入力は'L'レベル（Low:電圧が低い状態）にするとスタンバイ状態（非動作状態）なる入力信号です。SmallBot ではスタンバイモードは使用しませんので'H'レベル（High：電圧が高い状態）にして常時動作状態にしています。

PWM や IN1/IN2 入力はモーターが回転する方向などを決める入力端子で、次のような動作になります。

・OFF

IN2='L'、IN1='L'、PWM='H/L'

PWM 入力の状態に関係なく、モータードライバは OFF 状態になります。モーターの端子は解放状態と同じになり、モーターの軸を手で回すと軽く回ります。

・正転

IN2='L'、IN2='H'、PWM='H'

モーターに電圧がかかります。SmallBot ではこの時にロボットが前進する方向に車輪が回転するように設計されていますので、図では「正転」と書いています。

・逆転

IN2＝'H'、IN1='L'、PWM=H'

IN2='L'、IN1='H'の時とは逆向きにモーターに電圧がかかります。

SmallBot ではこの時にロボットが後退する方向に車輪が回転するように設計されていますので、図では「逆転」と書いています。

・制動

IN2＝’H’、IN1=’H’、PWM=’H/L’

IN2=’H’、IN1=’L’,PWM=’L’

IN2=’L’、IN1=’H’,PWM=’L’

　モーターの両方の端子が PGND に接続（接地）され、モーターの両方の端子が短絡されたのと同じ状態になります。

　モーターの端子を短絡すると自分自身が発生した電力で逆方向に回そうとする力が生まれますので、電気的にブレーキをかけたような状態になります(※)。回転しているときにこのモードにすると OFF 状態にしたときよりも速く停止しますし、停止状態から回転させる時も OFF 状態から回す時よりも大きな力が必要です。

※：電車やハイブリッド車などの回生ブレーキと同じ仕組みです。

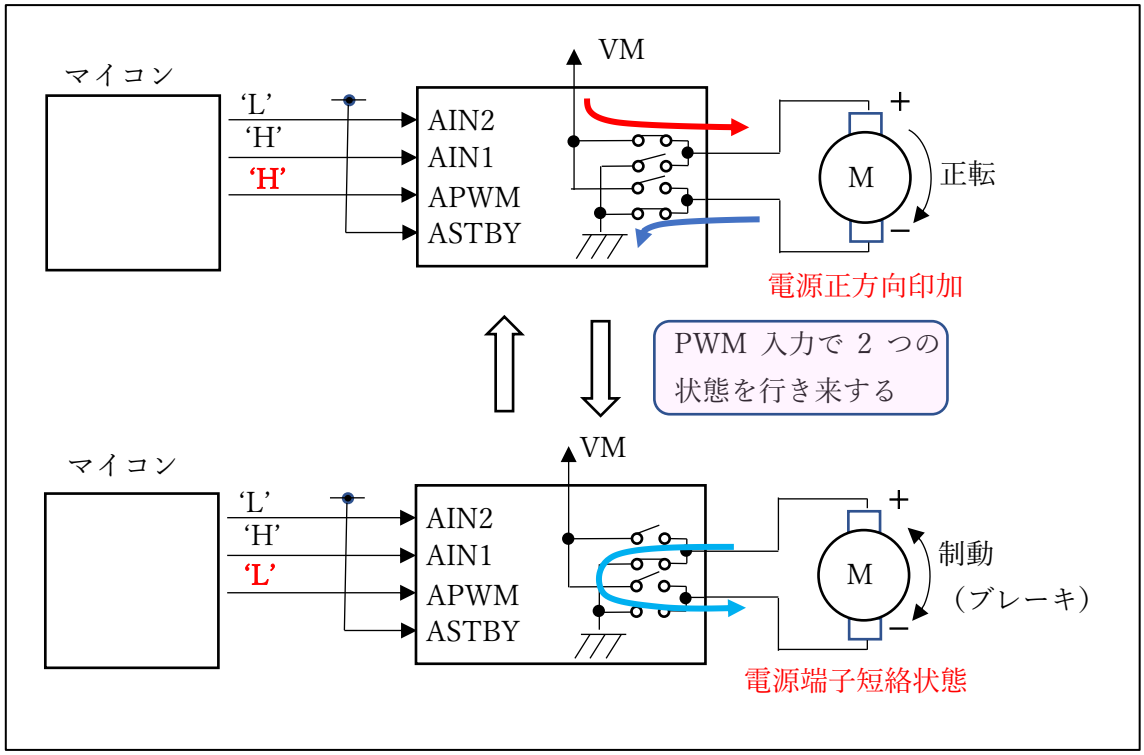


図 7：PWM 入力によるモーターの回転数制御（1／2）

2.1.3. PWM 入力による回転数制御

PWM 入力を利用するとモーターの電源を ON/OFF することができます。モーターに与えられる電力が変化しますので回転数も変化します。

図 7、図 8 は PWM 入力によるモーターの回転数制御を表しています。

IN1 と IN2 でモーターの回転方向を決め、PWM 入力を'H'にすると、図の上側のように指定された方向にモーターが回転します。PWM 入力を'L'にすると、図の下側のようにモーターは制動状態になり、停止します。

PWM 入力の ON/OFF の周期を速くすると、モーターが細かく加減速され、ON 期間の比率に応じた回転数で回転するようになります。ちょうどモーターに与える電圧を変化させたのと同じような効果になります。

SmallBot でもこれを利用してモーターの回転数を調整しています。

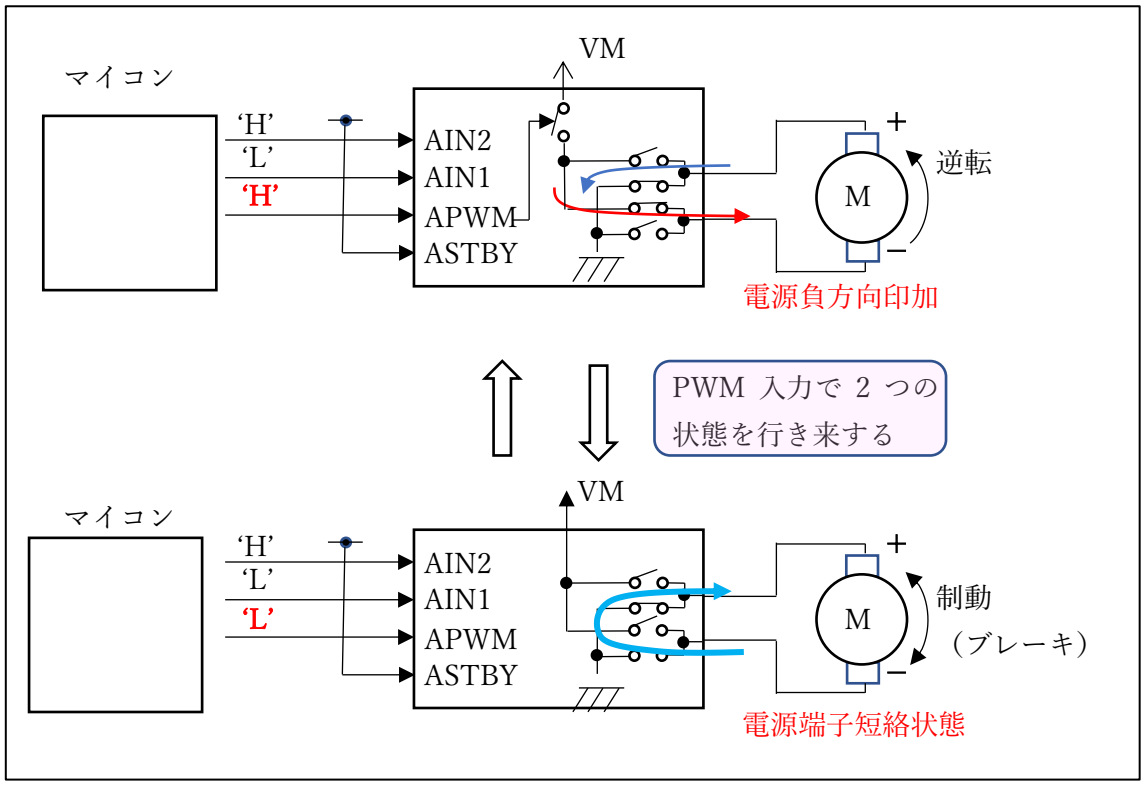


図 8：PWM 入力によるモーターの回転数制御（2／2）

2.2.　モータ駆動のソフトウェア

2.2.1.　Arduino の設定とモーターの回転数

PWM 入力の一周期の中の ON の期間が長くなるほどモーターの駆動力が大きくなり、回転数が上昇します。図 9 はこれを図にしたものです。

SmallBot ではソフトウェア開発環境として Arduino-IDE を利用していますが、このライブラリ関数の一つ、analogWrite()を利用することで、自動的に一定周期で ON/OFF を繰り返す波形が出力されるようになります。

analogWrite()関数の引数の値は 0 から 255 の範囲で、値が大きいほど ON の期間が長くなり、モーターの回転が速くなります。(※)

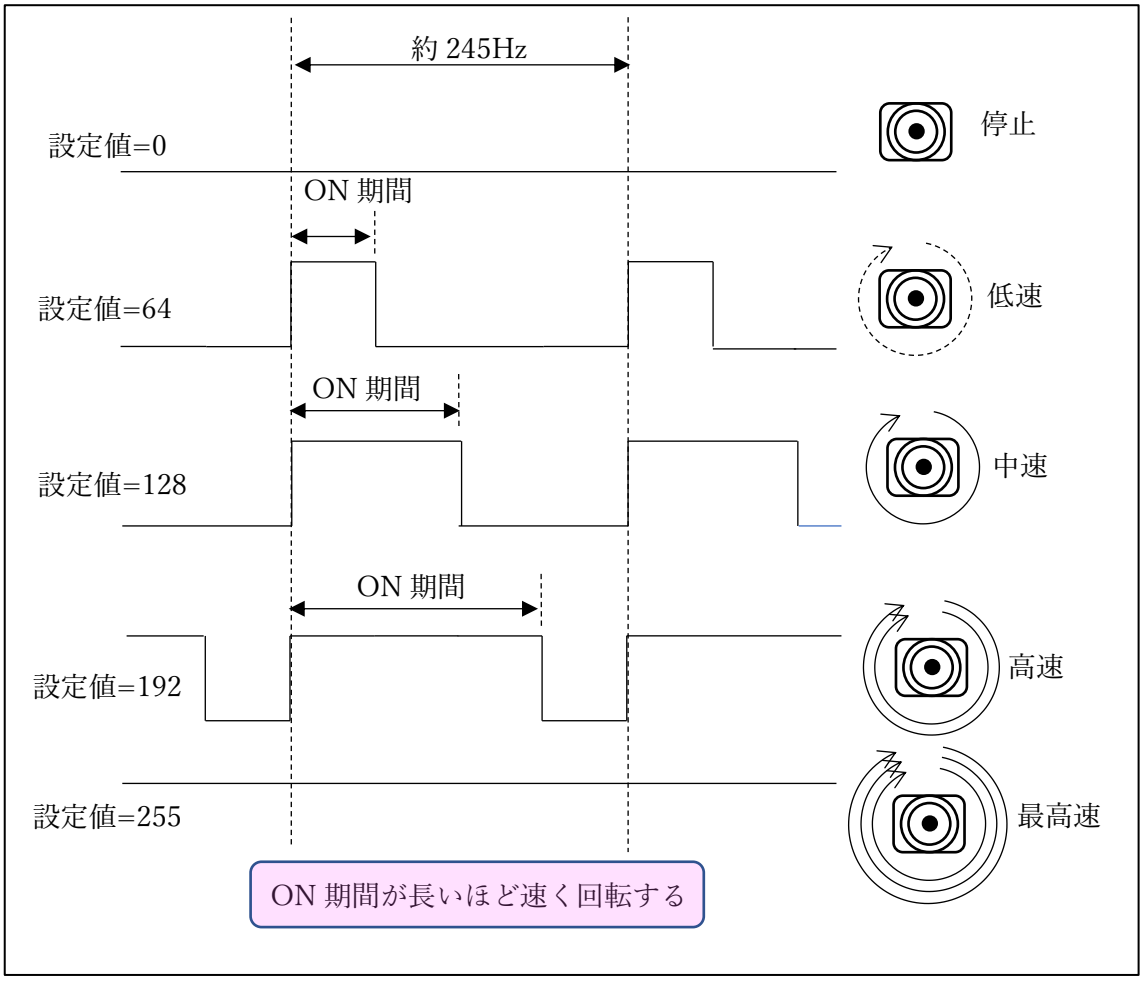


図 9：analogWrite()関数の設定値と出力波形、モーターの回転数

※ON 期間の比率が 0%ならば停止、100%ならば最高速度になりますが、ON 期間比率とモーターの回転数は比例するわけではありません。一定値以下だとモーターが回転し始めることができず、停止したままになります。

モーターの回転数は負荷の大きさによっても変化しますので、指定した回転数で回転させるためには、回転数を検出するセンサーを利用して、設定値に近づけるように制御する必要があります。

SmallBot には回転数のセンサは搭載していませんので、サンプルプログラムでは実機での動きを見ながら設定値を決めました。

2.3. プログラム中のモーター設定とロボットの進行方向

2.3.1. サンプルプログラム中のピン名称設

図 10 はサンプルプログラム中でピン名称の設定を行っている部分です。

ピンの番号は右側にある数値で示されています。これはプログラムの読みやすさを上げるためのものです。

たとえば、右側のモータードライバの PWM の設定値を 64 にする（'H'の期間が$\frac{64}{255}$になる）ならば

analogWrite(6,64);

のように書いても良いのですが、単に"6"という数字では何に対する設定なのかわかりにくく、間違えやすくなります。

図のように#define 文を使って定義しておくと、

analogWrite(M1_PWM,64);

という具合にわかりやすく記述することができ、間違えにくくなります。

```
// マイコンのピン（端子）番号の定義
#define M1_2      9     // 右モーターの IN2 ピン
#define M1_1      4     // 右モーターの IN1 ピン
#define M1_PWM    6     // 右モーターの PWM ピン

#define M2_2      8     // 左モータ用の IN2 ピン
#define M2_1      7     //
#define M2_PWM    5
```

ピンの名称　　ピンの番号

図 10：サンプルプログラム中のピン名称定義

2.3.2. ピンの初期化

マイコンのピンはリセット後、すべて入力になります。出力として使用したいピンは出力モードに初期化する必要があります。

ソフトウェアで'H'/'L'を都度設定するものは OUTPUT、PWM 出力するものは PWM_OUTPUT に設定します。

図 11 はサンプルプログラム中のモーター関連のピン初期化部分を抜き出したものです。

M1_1 や M1_2 等は
pinMode(ピン番号,OUTPUT);

M1_PWM や M2_PWM は
pinMode(ピン番号,PWM_OUTPUT);
と初期化します。

```
void setup() {
・・・・
        pinMode(M1_1, OUTPUT);
        pinMode(M1_2, OUTPUT);
        pinMode(M1_PWM, OUTPUT);
        pinMode(M2_1, OUTPUT);
        pinMode(M2_2, OUTPUT);
        pinMode(M2_PWM, OUTPUT);
・・・・
}
```


図 11：モーター関連ピンの初期化

2.3.3. プログラムとロボットの動き

サンプルプログラム（Arduino-IDE）中の設定とロボットの動きの関係を示したのが図 12、図 13、図 14 です。右モーターを M1、左モーターを M2 とします。

(1) 停止

M1、M2 とも停止状態（モータードライバの IN1、IN2 を LOW にする）にします。

ピンの HIGH、LOW の状態を設定するには digitalWrite()関数を使用して、

digitalWrite(ピン番号、状態);

のように記述します。

たとえば、M1_1（4 番ピン）を

LOW にしたいなら、
digitalWrite(M1_1、0);
というように記述します。
Arduino-IDE では 0 が LOW、1 が HIGH と名称定義されていますので、
digitalWrite（M1_1, LOW）;
と記述することもできます。どちらでも同じ動作になりますが、HIGH や LOW を使った方が間違えにくくて良いでしょう。

(2) 前進

左右のモーターをともに前進方向（正方向）に同じ速度で回転させるとロボットは前進します。
モーターの回転方向が前進方向なので、
M1_1=M2_1=HIGH
M1_2=M2_2=LOW
と設定します。
回転速度は analogWrite()によって PWM 出力の ON 期間の比率を設定します。サンプルプログラムでは

analogWrite(M1_PWM,120);
analogWrite(M2_PWM,120);
という具合にして、両方の速度を 120 に設定しています。

(3) 左回転（超信地旋回）

SmallBot の車輪の回転方向や速度は左右独立して設定できます。
左右の車輪の同じ速度で逆向き（片方を前進方向、他方を後退方向）に回転させると、ロボットは車軸の中央部分を軸にして旋回します。これを超信地旋回とも呼びます。
左（反時計回り）に超信地旋回させたいときは、右の車輪を前進方向、左の車輪を後退方向に回転させます。
プログラムでは図のように、
M1_1= HIGH、M1_2=LOW
M2_1=LOW、M2_2=HIGH
と設定しています。回転速度は左右とも同じ値にします。値を大きくするほど回転する速度も速くなりますが、サンプルプログラムでは直進時と同じ値（120）に設定しています。
サンプルプログラムではダンゴムシ（超音波距離センサによる障害物回避）やダンス（音声入力による方向転換）モードの時の方向転換に超信地旋回を使用しています。

なお、プログラムは上から順に実行されますので、厳密に言えば、片方のピンの設定をしてからもう一方のピンの設定をするまでの間にわずかな時間差がありますが、マイコンの動作はモーターの応答時間よりも十分に高速なので同時に設定されると考えても問題ありません。

(4) 右回転（超信地旋回）

車輪の回転方向を左回転の時と逆、つまり右の車輪を後退方向、左の車輪を前進方向に回転させるとロボットはその場で右回転（時計回り）します。

(5) 左旋回

左右の車輪を前進方向に回転させると、ロボットは前進しますが、このとき回転数に差があると、直線的には進まず、回転数の差に応じた半径の円を描くように前進します。

差が少なければ半径は大きくなり、差が大きければ半径は小さくなります。

右の車輪が左の車輪よりも速く回転すれば、ロボットは左回り（反時計回り）の円を描くように移動していきます。

サンプルプログラムではフォトセンサを使って床に描かれた線を辿るライントレースモードのときに旋回動作を利用しています。

PWM の設定値は

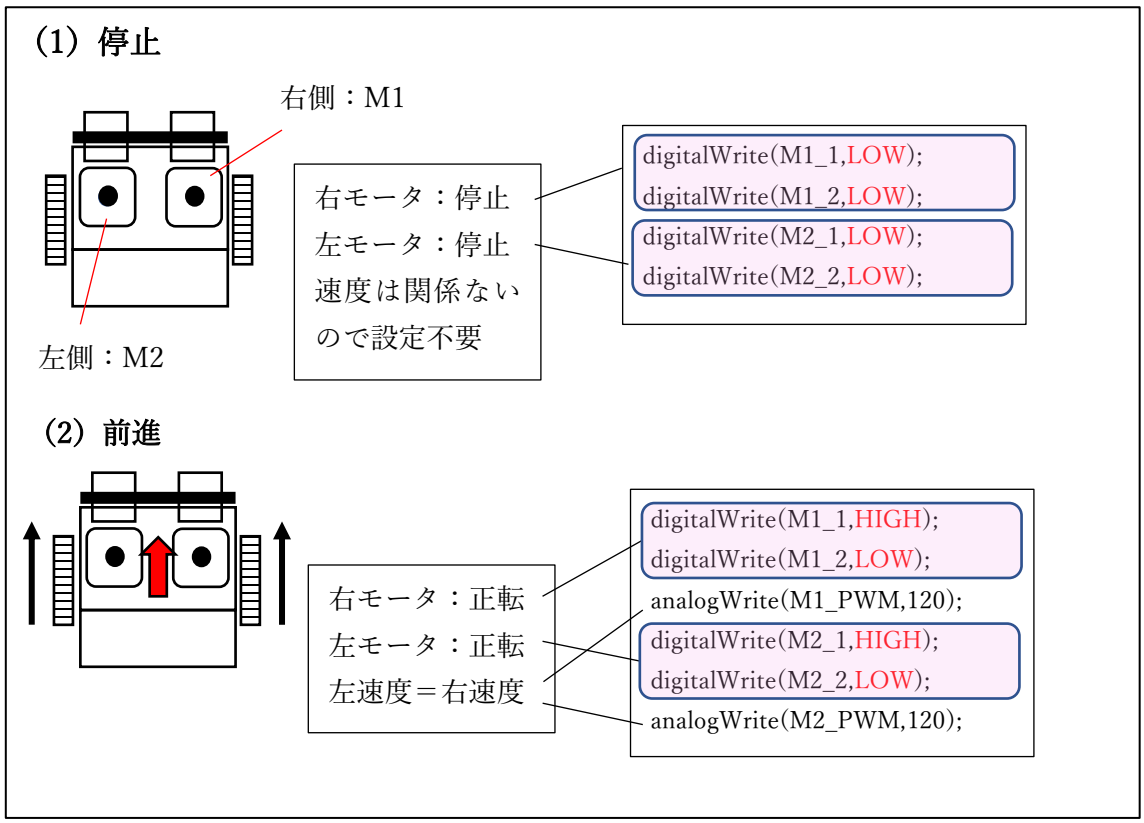


図 12：モータードライバの設定とロボットの移動方向(1/3)

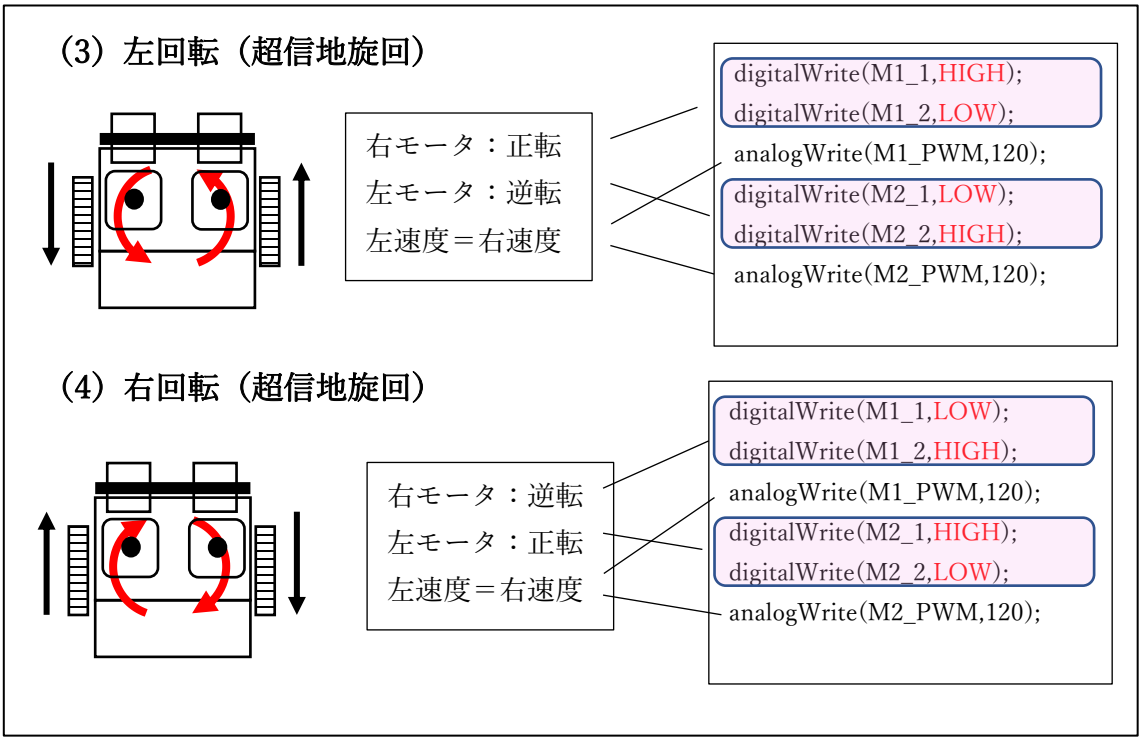


図 13：モータードライバの設定とロボットの移動方向(2/3)

・右車輪：120
（直進時と同じ）
・左車輪：80
にしています

・右車輪：80
（直進時と同じ）
・左車輪：120
に設定しています

(6) 右旋回

左旋回の時と逆に、左の車輪の速度を右の車輪の速度よりも大きくするとロボットは右回り（時計回り）の円を描くように移動します。

サンプルプログラムでは

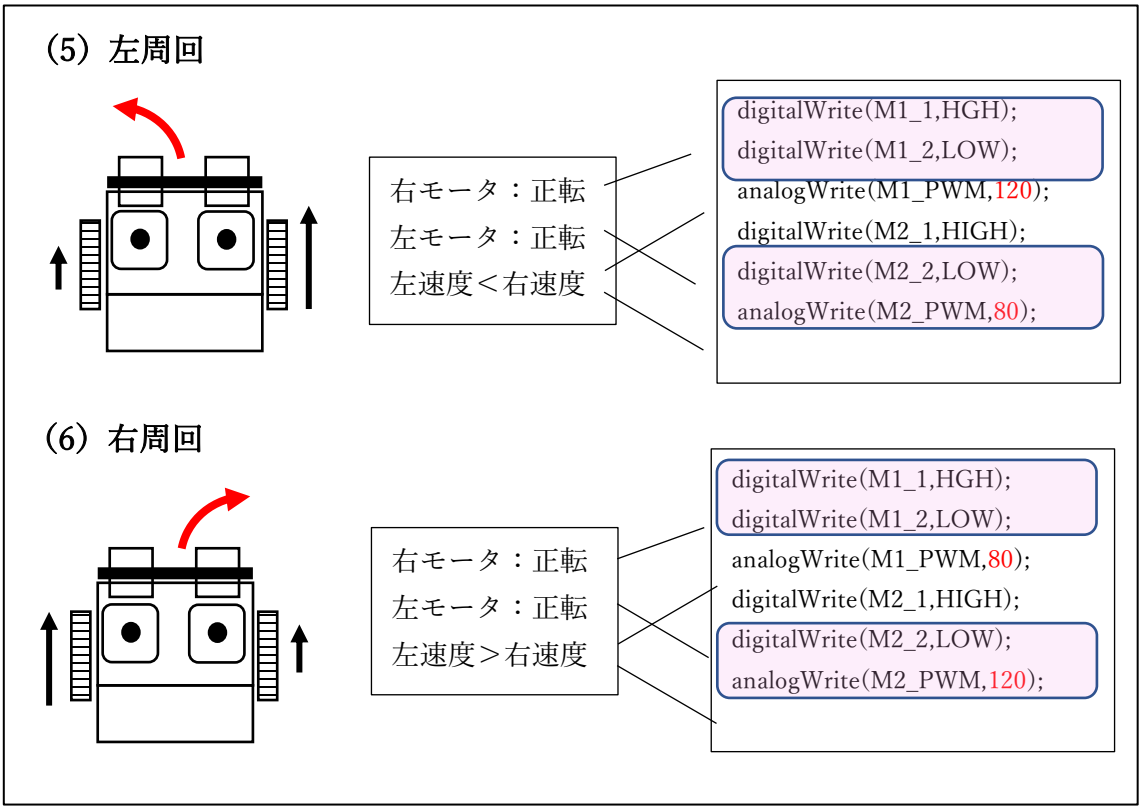


図 14：モータードライバの設定とロボットの移動方向(3/3)

3. スイッチ入力

3.1. スイッチ入力のハードウェア

3.1.1. スイッチ入力回路

図 15 は SmallBot の押しボタンスイッチ（ロボットの正面パネルの左下隅のところにあります）の配置と入力回路を示したものです。

スイッチは図の右側のように正面左下にあり、回路上では Arduino-IDE 上のプログラム中の 12 番ピン（マイコンの 26 番ピン）に接続されています。スイッチは通常 OFF 状態で、ボタンを押すと ON になります。

マイコン内部にはプルアップ抵抗があり、ピンのモード設定で Vcc と接続／切断することができるようになっており、スイッチ入力として使うときはこれを ON にします。

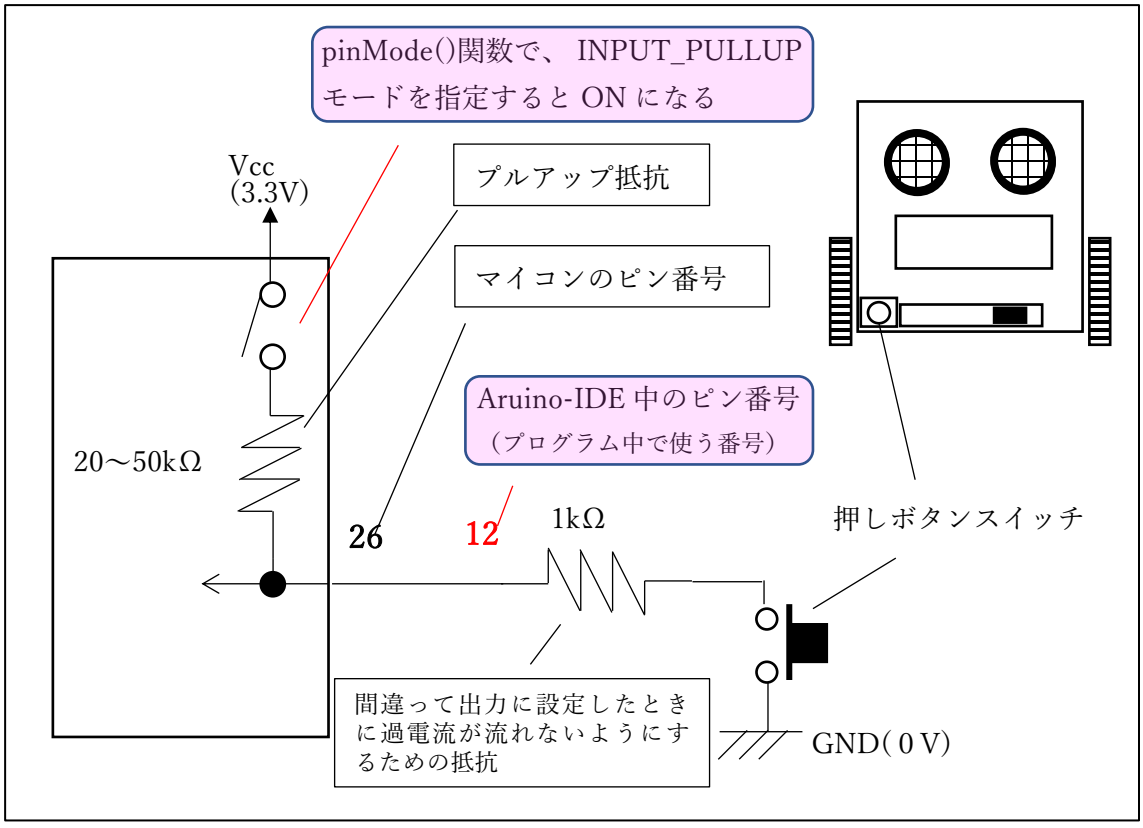


図 15：SmallBot の押しボタンスイッチ入力回路

3.1.2. HIGH/LOW 判定

マイコンのピンをデジタル入力モードで利用した場合、ピンに与えられた電圧が高い（HIGH）状態ならば'1'、低い（LOW）状態ならば'0'として扱われます。

HIGH 電圧や LOW レベル電圧として扱われることが保証される電圧は製造時のばらつきや動作条件によって変化しますので、どちらとして扱われるか定まらない領域があります。これを示したのが図 16 です。（※）

SmallBot に使われているマイコン（ATMEGA32U4）の場合、マイコンの電源電圧を Vcc とすると、

・0.2×Vcc+0.1V 以下は LOW
・0.2×Vcc+0.9V 以上が HIGH

となっています。SmallBot の場合、Vcc は 3.3V ですので、

・LOW：0.2×3.3+0.1=0.76[V]以下
・HIGH：0.2×3.3＋0.9=1.56[V]以上

となります（※）

これは、0.76V から 1.56V の間に'1'と判断するか'0'と判断するかの境界となる電圧（スレッショルド電圧と呼びます）があるということを意味しています。

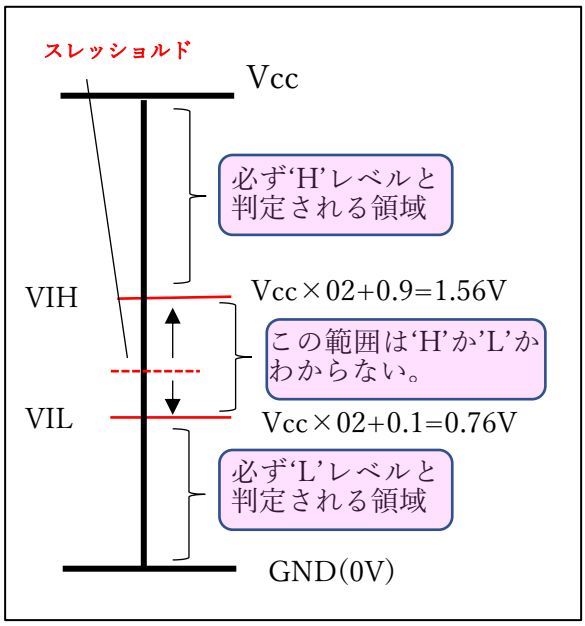


図 16 入力電圧と HIGH/LOW 判定

※：データシートなどでは、LOW レベルと判断される電圧の上限を VIL、HIGH レベルの下限を VIH と表記しています。

※：入力電圧が VIL と VIH の間にある場合、動作条件などにより HIGH と LOW のどちらとして判断されるかはわかりませんので、ハードウェア設計時は入力電圧がこの範囲にならないようにしたり、この範囲に入っても動作に大きな問題がでないように気をつける必要があります。

3.1.3. スイッチ入力回路の動作

図 17 はスイッチを ON/OFF したときの動作を示したものです。

先ほど触れたとおり、マイコンの入力ピンはデジタル入力モードに設定すると、入力電圧が高い（HIGH）ときには‘1’、低い（LOW）時には’0’として読みだされます。

スイッチが OFF の場合、プルアップ抵抗によってピンの電圧はほぼ Vcc（3.3V）になります。VIH の下限の 1.56V よりも高いため、HIGH と判定されます。

スイッチが ON の時は内部プルアップ抵抗と 1kΩの抵抗で分圧された電圧になります。

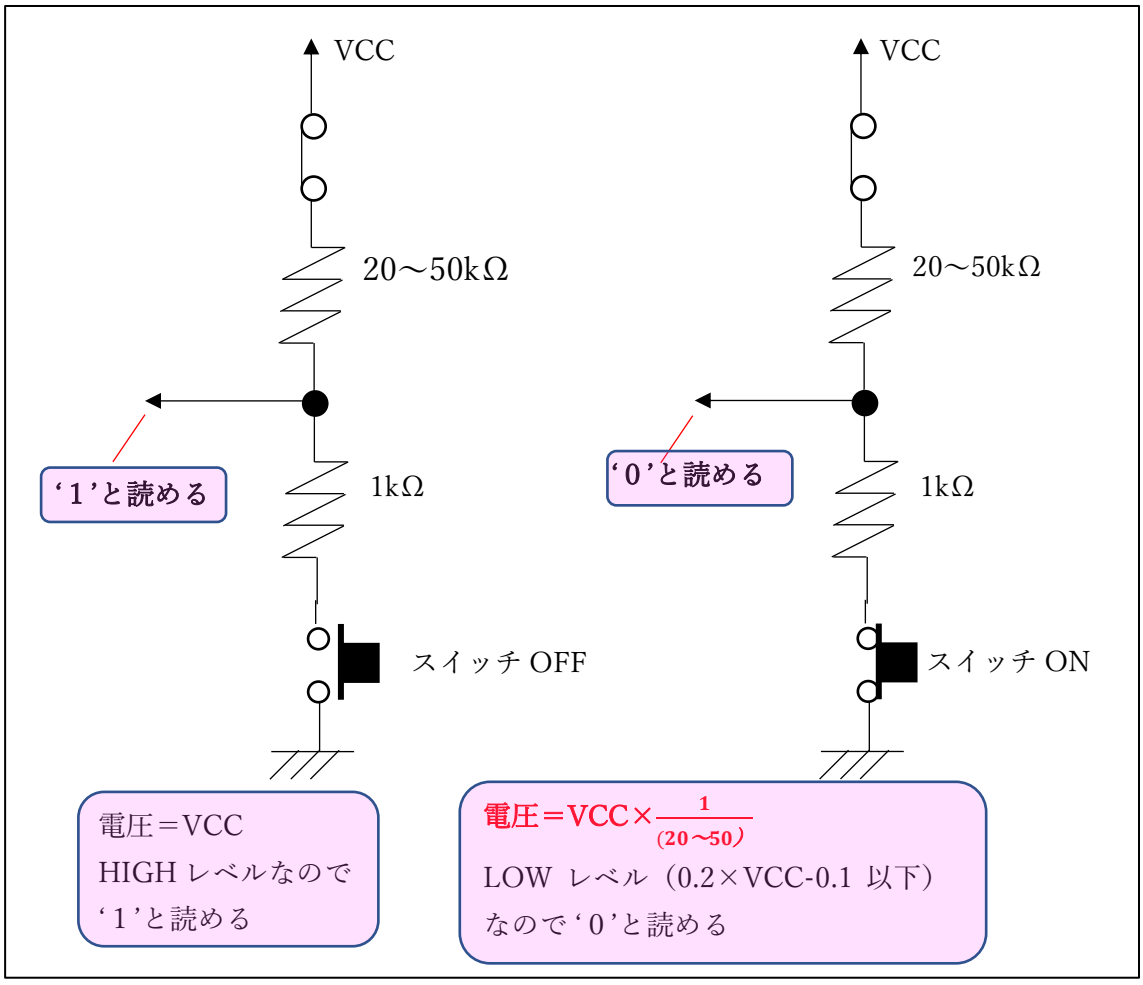


図 17：スイッチ入力の動作

内部プルアップ抵抗の値は 20kΩ～50kΩとなっています。仮に ON 時の電圧が最も高くなる 20kΩとすると、スイッチが ON の時の入力電圧 Vin は次のように算出されます。

$$Vin=\frac{1}{(20+1)}\times Vcc$$

$$=\frac{1}{(20+1)}\times 3.3$$

≒0.16[V]

これは VIL の 0.76V よりも低いので LOW と判定されます。

つまり、スイッチが

・OFF ならば HIGH（‘1’）

・ON ならば LOW（’0’）

になるわけです。

3.2. スイッチ入力のソフトウェア

図 18 はサンプルプログラムからスイッチの読み込みに関係する部分を抜き出してみたものです。

スイッチの読み込みに使うポートはプルアップ抵抗付を ON にしますので、

PinMode（）の第二引数のモード設定で INPUT_PULLUP にします。

第一引数はピン番号の指定です。スイッチ読み込み用ポートは 12 番です。サンプルでは#define 文を使って BUTTON_PIN という名称でアクセ

```
・・・
#define BUTTON_PIN     12      // 押しボタンスイッチは 12 番ピン
・・・
void setup() {                  押しボタンスイッチはプルアップ付きの入力とする
・・・
    pinMode(BUTTON_PIN, INPUT_PULLUP);
・・・
}

#define SWON    0
#define SWOFF   1               スイッチが ON ならば’0’、
void loop() {                   スイッチが OFF ならば’1’
    int swval;
    swval = digitalRead(BUTTON_PIN);    // スイッチの状態読み込み
    ・・・
        if (swval == SWON) {        // スイッチが ON
  ・・・
```

図 18：サンプルプログラムのスイッチ読み込み

スできるようにしています。

なお、プルアップ抵抗を使用しない（プルアップを OFF にする）場合は第二引数を“INPUT”にします。(※)

ピンの状態の読み込みは pinRead() 関数を使用します。第一引数でピン番号を指定すると、戻り値でピンの状態（スイッチの状態）が返されます。戻入力が HIGH なら 1 、LOW ならば 0 が戻り値になります。

SmallBot のスイッチ入力回路ではスイッチが OFF でピンが HIGH、ON になると LOW になりますので、

・スイッチが OFF ならば 1
・スイッチが ON ならば 0
となります。

たとえばスイッチが ON のときに何らかの処理をするような場合であれば、

```
#define SWON 0
swdat = pinRead(BUTTON_PIN);
if (swdat == SWON) {
  スイッチが ON の時の処理
}
```

という具合にすればよいわけです。

※：リセット後は自動的に INPUT になりますので、INPUT のまま使うならば設定は省略可能です。

4. LED

LED(Light Emission Diode：発光ダイオード)は、回路の信号の状態を目視確認したり、ソフトウェアの動作状態を表示するなど、システムの動作状態を示す表示器として広く利用されています。

SmallBot に使われている LED の 1 つがユーザが自由に ON/OFF(点灯／消灯)できるものになっています。発光色は白色です。

4.1. LED 出力のハードウェア

4.1.1. LED 点灯回路

図 19 は SmallBot 上でユーザが自由に利用可能な LED（以下、ユーザ LED と略します）の位置と点灯回路を示した物です。

ユーザ LED はロボットの右下のところにあります。ライントレースするときに床を照らすためにも利用しています。

回路上では ArduinoIDE 上の 13 番ピン（マイコンの PC7:32 番ピン）に接続されています。

ユーザ LED に使われている白色 LED はマイコンの I/O ポートで直接駆動することが難しいため、FET を追加しており、ポートが HIGH の時に点灯するようになっています。

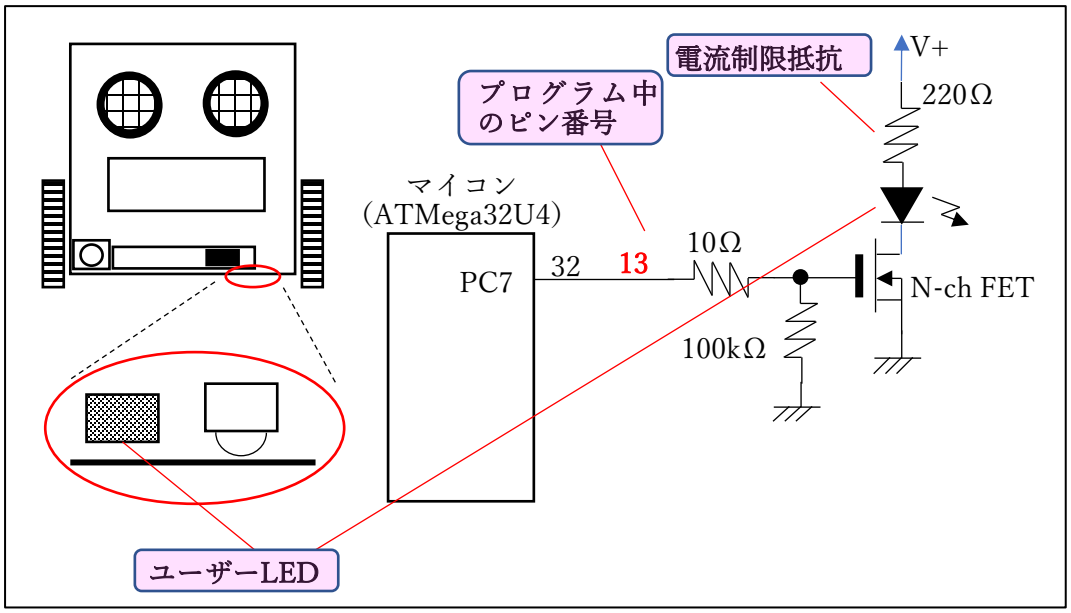


図 19:LED 点灯回路

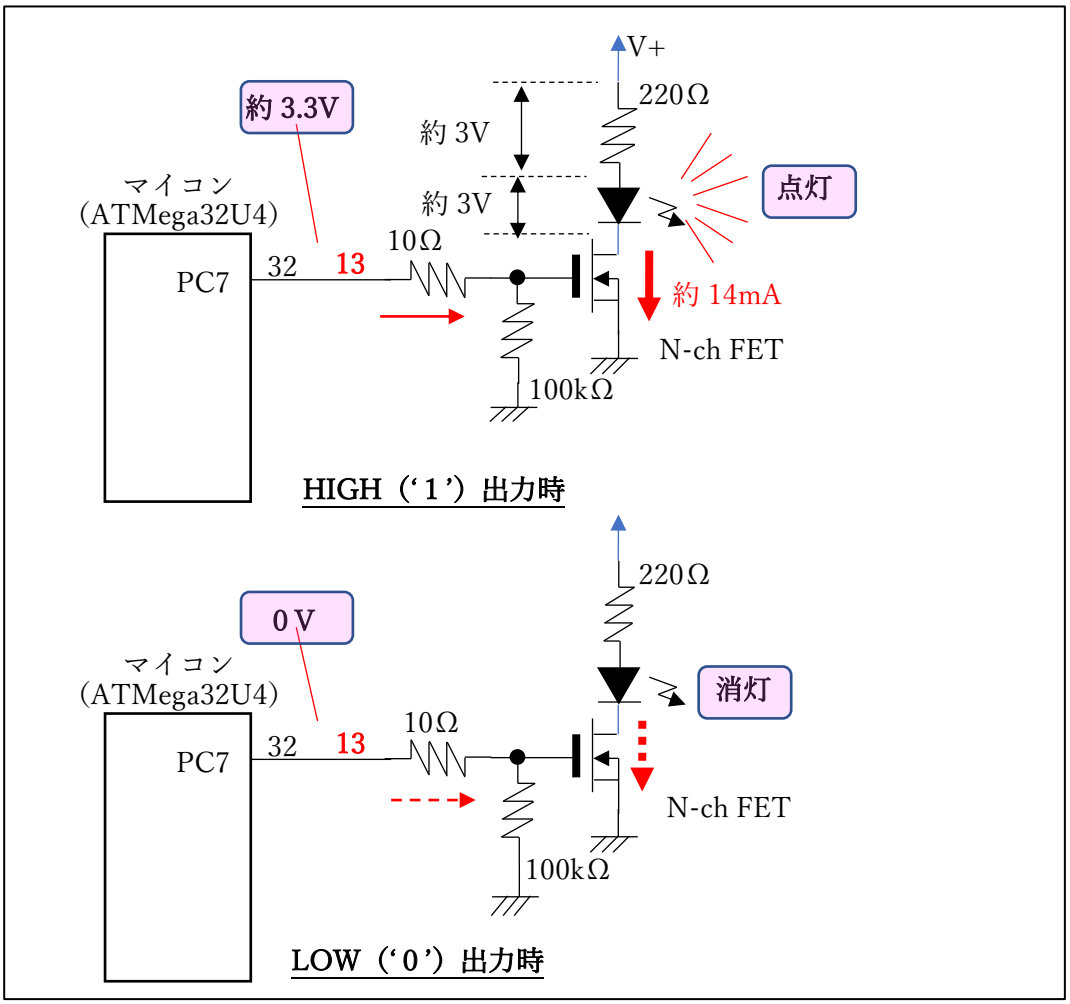


図 20：LED 点灯回路の動作

4.1.2. LED 点灯回路の動作

LED 点灯回路の動作を示したのが図 20 です。N-ch の FET は電圧で動作するスイッチのようなものと考えると良いでしょう。

マイコンの出力が HIGH（ソフトウェアで’1’を書き込む）になると、FET が ON になり、LED に電流が流れます。LED につながっている 220Ωの抵抗は LED に流れる電流を制限するものです。白色 LED が点灯させたときの電流は数 mA から 20mA 程度です。このときの LED の端子間電圧（順方向降下電圧）は製品や動作条件によって異なりますが、おおむね 3V 程度あります。仮に 3V として、FET の部分の降下電圧をほぼ

0V、電源電圧を 6V とすると、図のように 220Ωの抵抗の端子間電圧は 3V となります。

ここから LED に流れる電流（抵抗を流れる電流と同じ）は

$\frac{3}{220} \fallingdotseq 0.014[\mathrm{A}] = 14[\mathrm{mA}]$

となります。

マイコンの出力が LOW（ソフトウェアで'0'を書き込む）のときは FET が OFF 状態になりますので、LED に電流は流れず、消灯します。

4.2. LED 点灯ソフトウェア

図 21 はサンプルプログラムから LED 点滅部分に関係する部分を切り出したものです。

LED は Arduino-IDE 上の 13 番ピンに接続されていますので pinMode()関数で 13 ピンのモードを OUTPUT に設定しています。

digitalWrite()がピンへの出力を行う関数で、第一引数でピン番号（今回は 13)、第二引数は、LED を点灯するときは 1 を、消灯するときは 0 を与えます。

```
#define LED_PIN  13
void setup() {
    ……
    pinMode(LED_PIN, OUTPUT);
    ……
}

Void loop() {
    ……
    digitalWrite(LED_PIN, 1);  // LED 点灯
    ……
    digitalWrite(LED_PIN, 0);  // LED 消灯
    ……
}
```

LED は 13 ピンを使用している

ピンをデジタル出力モードにする

1 を書き込むと LED が点灯する

0 を書き込むと LED が消灯する

図 21：サンプルプログラム中の LED 点滅

5.　スライドボリューム（可変抵抗器）

スライドボリュームはツマミの位置を左右に動かすことで抵抗値を変化させることができる部品で、「可変抵抗器」と呼ばれるものの一つです。SmallBot ではツマミの位置で電圧が変化するようにして、これをマイコンのアナログ入力で読み取れるようにしています。読み取った値によって、ツマミの位置を判断できます。

サンプルプログラムでは、ダンゴムシモード時の方向転換する角度の調整や、ダンスモードの時の音量レベル設定、ライントレース時に線の色の白黒の区別をするレベルを設定するのに使用しています。

スライドボリュームの基本的な構造は図 22 のようになっています。

電気抵抗の大きなもの（抵抗体）が塗布された棒状の物の両端に端子（固定電極）が取り付けられています。この棒の上を移動できる摺動子と呼ばれる電極が移動します。

摺動子の位置を変化させると、固定電極との間の抵抗値が変化します。

スライドボリュームの等価回路は図の右側のようになります。、抵抗体の両端の抵抗 Rc、図の端子 A と摺動子の間の抵抗を Ra、端子 B と摺動子の間の抵抗を Rb とすると、

Rc＝Ra＋Rb（=一定）

0≦Ra≦Rc

0≦Rb≦Rc

という関係が成り立ちます。

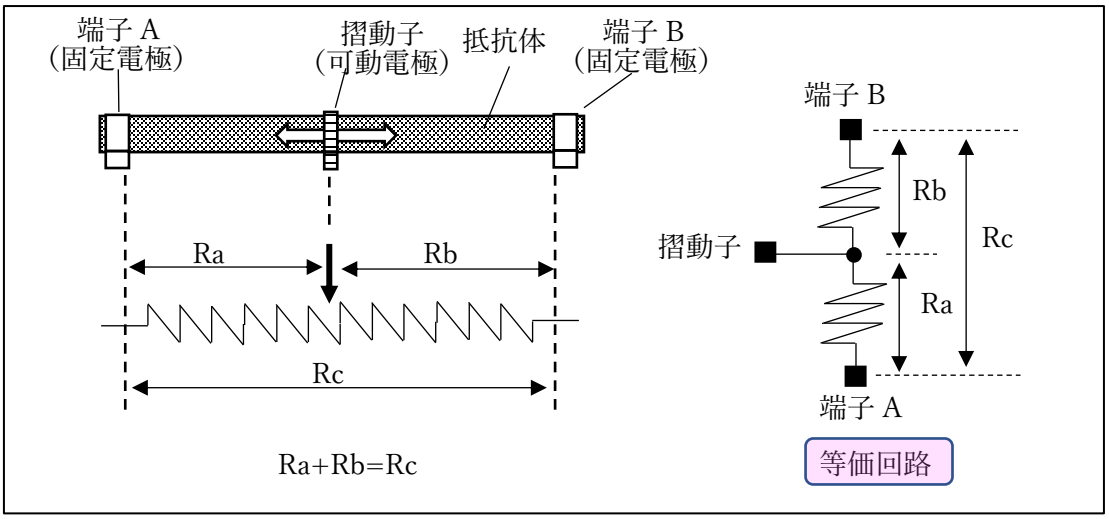


図 22：スライドボリュームの基本構造

5.1. スライドボリューム入力のハードウェア

5.1.1. スライドボリューム入力回路

s

SmallBot のスライドボリューム入力部分の回路は図 23 のようになっています。スライドボリュームの両端の端子に Vcc と GND を接続して、摺動子端子を Arduino-IDE の A3 ピン（マイコンの 39 番ピン：PF4）に接続しています。

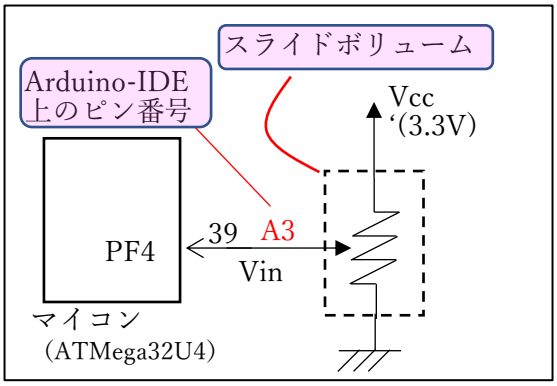


図 23：スライドボリューム入力回路

5.2. スライドボリューム入力回路の動作

スライドボリュームの摺動子を動かしたときの動作を示したのが図 24 です。スライドボリュームは図の右側のように二つの抵抗で表すことができます。摺動子を動かすと Ra と Rb の比率が変化します。

摺動子の電圧 Vin を Vcc、Rc、Ra を使って表すと

$Vin=\frac{Ra}{Rc}\times Vcc$

$=\frac{Vcc}{Rc}\times Ra$（0≦Ra≦Rc）

となります。

この式の中の Vcc は電源電圧ですから一定値（SmallBot では 3.3V）ですし、Rc はスライドボリュームの製品によって決まっています（製品の仕様には Rc の値が記載されています)。

従って、$\frac{Vcc}{Rc}$の部分は定数になりますので、これを k と表せば

$Vin = k \times Ra \left(k\text{は比例定数}\right)$

という式と同じです。つまり、Ra と Vin は比例するわけです。

たとえば、Vcc が 3.3V で、Rc が 5kΩ、摺動子が GND 側から測って 2kΩの位置にあれば、

$Vin = \frac{3.3}{5\times10^3}\times 2\times 10^3$

$= 1.32[V]$

となります。

Ra と Vin の関係をグラフで示したのが図 25 です。　Ra が 0[Ω]から Rc[Ω]まで変化すれば、Vin は 0V（GND）から Vcc まで変化することになります。

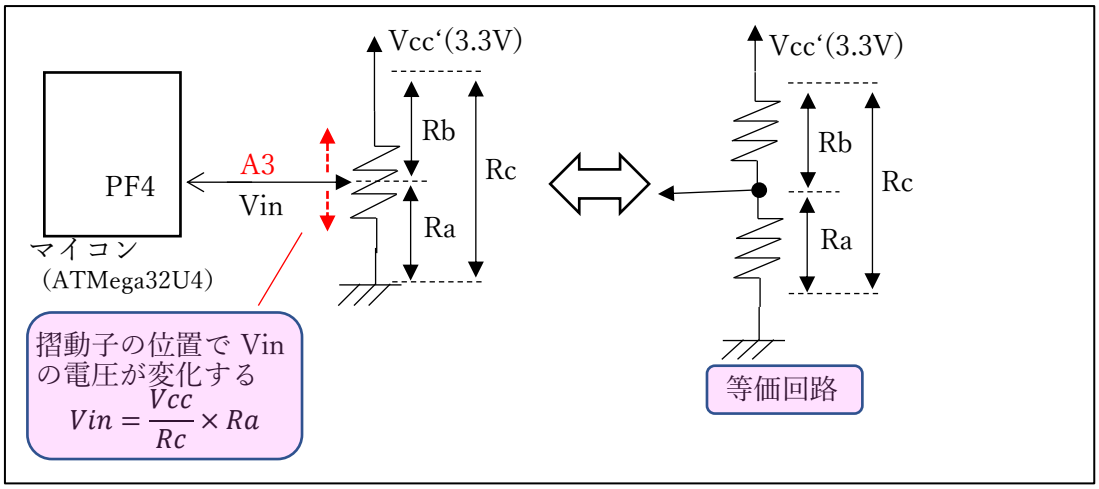


図 24：スライドボリュームの動作

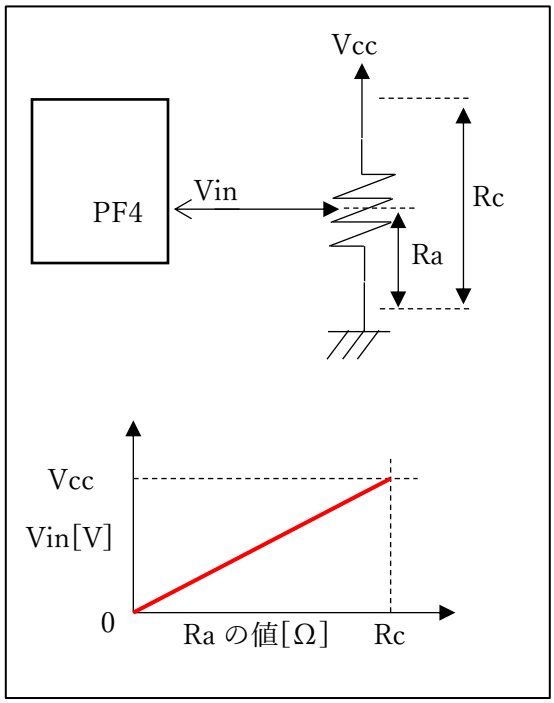


図 25：スライドボリュームの Ra の値と Vin の関係

5.2.1. スライドボリューム読み込みソフトウェア

図 26 はサンプルプログラム中のスライダの値読み込み部分を切り出してきたものです。

デフォルトで I/O ピンは入力モードになりますので、LED 点灯の時のように pinMode()でピンの動作モードを設定する必要はありません。

入力値はスイッチ入力のようなデジタル値（1 か 0 か）ではなく、アナログ値として読み込みます。これには analgoRead()関数を使用します。

analogRead()関数の引数にはピン番号（スライダの場合は“A3”）を指定します。

analogRead()の戻り値が入力電圧を表す値です。SmallBot の場合、アナログ値は 0～1023 の範囲の値になり、入力が 0V なら 0、Vcc の時は 1023 が返されます。

戻された値を Val、電源電圧を Vcc とすると、入力電圧 Vin は

$$Vin=\frac{Vcc}{1023}\times Val\ [V]$$

となります。Vcc が 3.3V ならば、

$$\frac{3.3}{1023}\fallingdotseq 0.0032$$

ですので、約 3.2mV 単位で入力電圧を算出できます。

```
void loop() {
・・・
  sliderValue = analogRead(A3);          // スライダーの読み取り
・・・
  Timer = map(sliderValue, 0, 1023, 60, 1200);  // スライダ値によって回転量を決定
・・・
}
```

スライダが接続されている A3 ポートの電圧をアナログ値として読み込み（0～1023）

0～1023 の値を 60～1200 の値に変換

図 26：サンプルプログラムのスライダ値読み込み部分

5.2.2. map()関数

map()関数はアナログ入力とは直接関係ありませんが、サンプルプログラム中でアナログ値の変換用として利用されていますので、ここで説明しておきましょう。

図 27 は map()関数による値の変換動作を図示したものです。

map()関数は第二、第三引数で指定された範囲の値（この例では 0～1023）が第四、第五引数で指定した範囲（この例では 60～1200）になるような規則に従って第一引数の値

（この例では SliderValue）を変換します。変換後の値は関数の戻り値として返されます。図のように
map(sliderValue, 0, 1023, 60, 1200);
ならば変換規則は

$$\frac{1200-60}{1023-0} \times SliderValue$$
$$= \frac{1140}{1023} \times SliderValue + 60$$

となります。

なお、サンプルプログラム中の 60 や 1200 という値は実際にロボットを動かしながら試行錯誤して決めた値です。

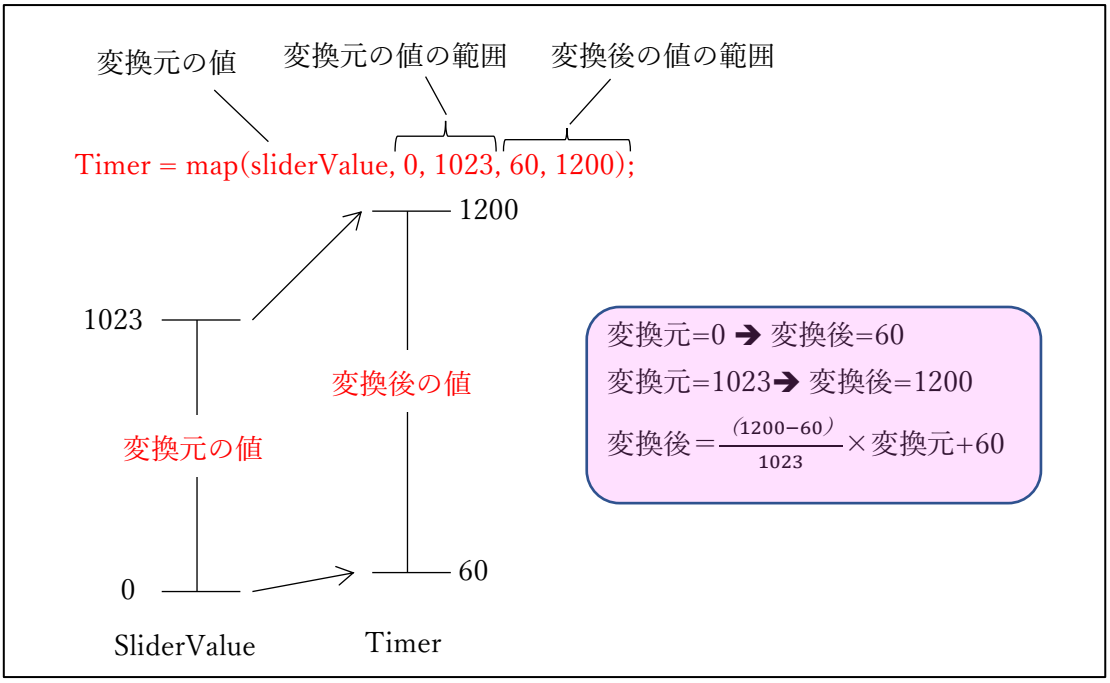


図 27：map()関数による値の変換

6. フォトセンサ

フォトセンサは周囲の明るさの計測を行ったり、LED などの光源と組み合わせて障害物の有無を判定したり、光を使ったデータ通信など様々な用途で使われています。暗くなったら点灯するライトや、赤外線を使った自動ドア、赤外線リモコンなどもフォトセンサの応用です。

6.1. SmallBot のフォトセンサのハードウェア

6.1.1. フォトセンサ回路

SmallBot のフォト（光）センサの位置と、入力回路は図 28 のようになっています。

フォトセンサはロボットを正面から見て右下のところに下向きに付いており、ロボットの走行中に床の明るさを判定できるようになっています。

サンプルプログラムではこれを利用してライントレース動作を実現しています。

SmallBot に使われているフォトセンサはフォトトランジスタと呼ばれるもので（※）、電圧をかけた状態で光を受けるとそれに応じて流れる電流が変化します。

SmallBot の場合には回路図のように、フォトトランジスタに抵抗をつけて電流変化を電圧変化にして、これをマイコン（Arduino—IDE 上の A2 ピン）で読み取るようにしています。

※：以前は CdS セルと呼ばれる、硫化カドミウムを利用したものが広く利用されていましたが、現在は小型で高感度であり、応答速度も速いフォトトランジスタが広く利用されています。

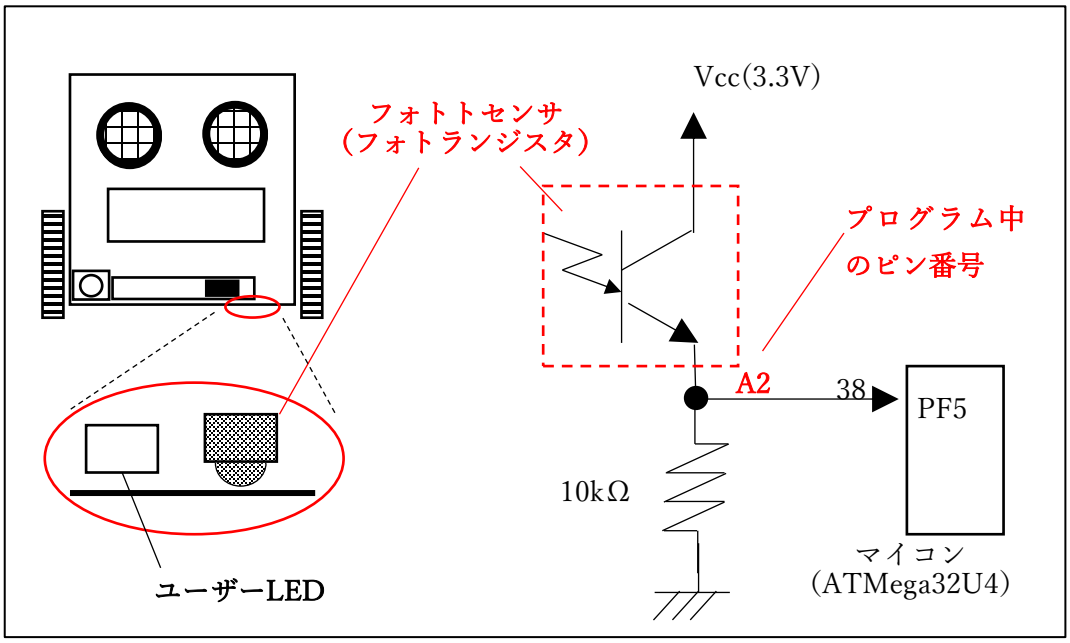


図 28：フォトセンサ入力回路

6.1.2. フォトトランジスタの動作

フォトセンサとして現在よく利用されているのが、フォトトランジスタと呼ばれるデバイスです。

フォトトランジスタの基本動作を示したのが図 29 です。フォトトランジスタに光が当たると、光の強さに応じた電流が流れます。光が強いほど電流が大きくなります（限度はありますが）。

図のような回路にすると、フォトトランジスタに流れる電流が多いほど高い電圧が得られるようになりますので、明るさの変化に応じた電圧が得られるわけです。

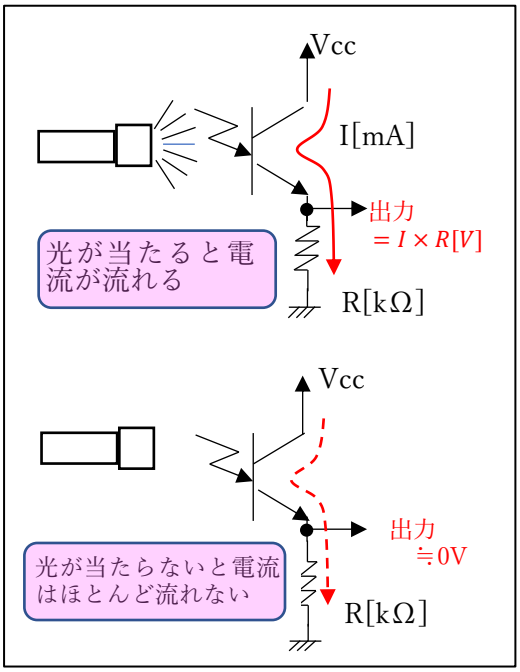


図 29：フォトトランジスタの動作

6.2. フォトセンサのソフトウェア

図 30 はサンプルプログラムのフォトセンサの入力部分を切り出してきたものです。フォトセンサの信号はアナログ入力として扱います。

アナログ電圧として読み出す時は analogRead()関数を使用します。第一引数でピンを指定します。

SmallBot ではアナログ入力が可能な A2 ピンに接続していますので、第一引数は”A2”にします。

```
void Mode_LINE() {
    sliderValue = analogRead(A3); // スライダーの読み取り
    if (analogRead(A2) > map(sliderValue, 0, 1023, 0, 300)) { // センサ値読み取り
    // 右旋回
        Set_Motor(C_RIGHT);
    } else {
    // 左旋回
        Set_Motor(C_LEFT);
    }
}
```

フォトセンサの出力電圧の読み込み
（0～1023 の値が返ってくる）

図 30：サンプルプログラムのフォトセンサ入力

analogRead()によってピンの電圧レベル（0～Vcc）が 0～1023 の値に変換されます。

サンプルプログラムではフォトセンサの値とスライドボリュームの値を比較して、フォトセンサの値の方が大きければ（明るければ）右に旋回、小さければ（暗ければ）左に旋回するようにしていますので、スライドボリュームのツマミを動かすことで、左旋回と右旋回の境界となる明るさを変更することができます。

図 31 はライントレースの動作を示したものです。

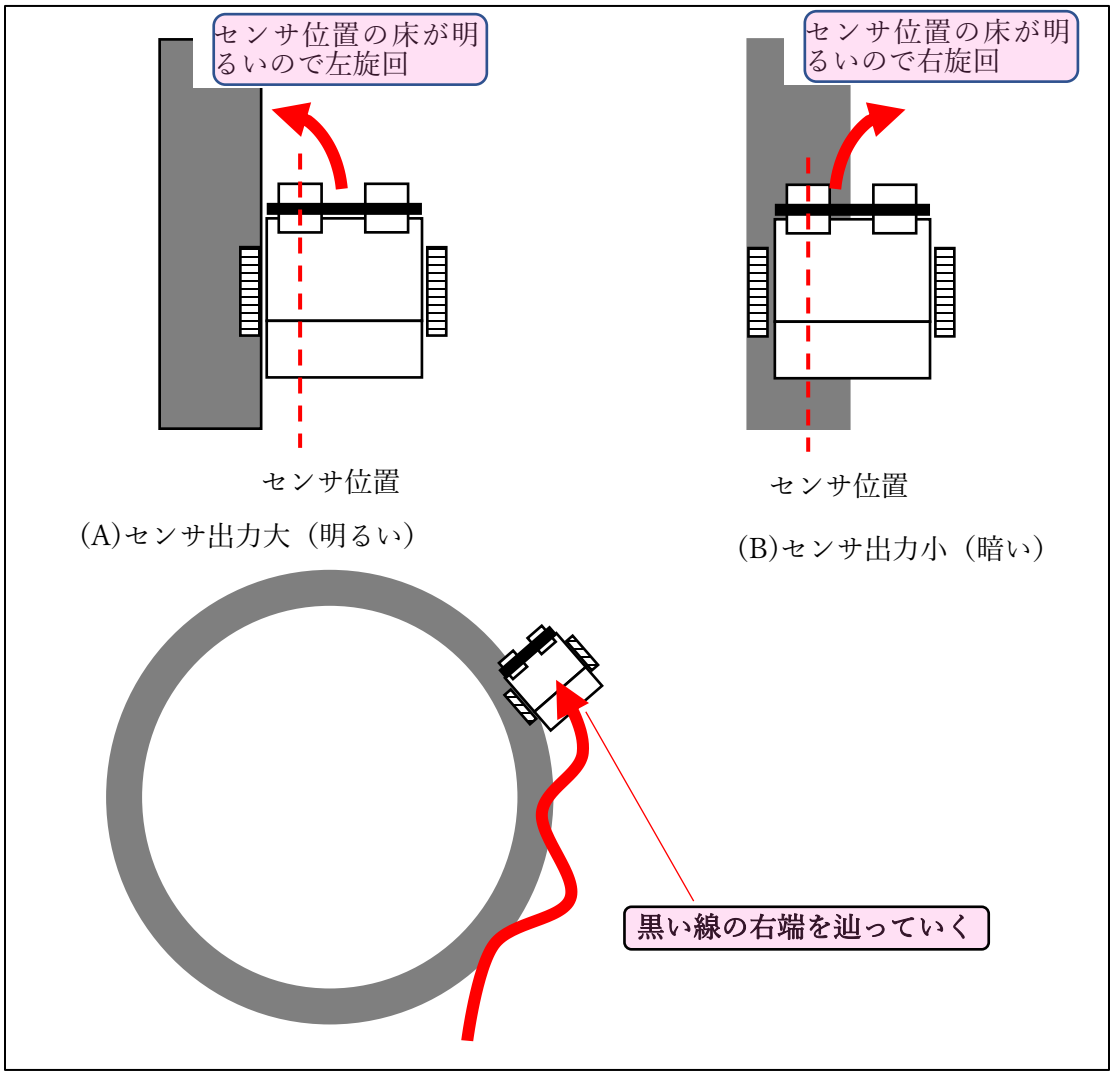


図 31：ライントレース動作

図のように、白地の上に黒い領域があり、ロボットを線の右端に置きます。

図の（A）のようにフォトセンサが白い領域にあると左に旋回して黒い領域側に向かいます。

旋回して(B)のようにセンサが黒い領域にかかると、今度は右に旋回して白い領域に向かいます。

この結果、図の下側のように、左右に振れながら黒い線を辿るように進みます。

黒い線の左側の白い領域に置いてスタートした時の動作例を示したのが図 32 の(A)です。図のように黒い線の左側の白の領域にロボットのフォトセンサがあると、ロボットは左旋回します。旋回中に黒い線を捉えると今度は右旋回して、先ほどの例とは逆向きに線を辿っていきます。旋回しているところに黒い線が無ければ白い領域で旋回し続けることになります。

（B）のように黒い線の上にあれば、右に旋回していきますので、線の右側に出ていき、先ほどと同じように線の右側を辿るような動きになります。

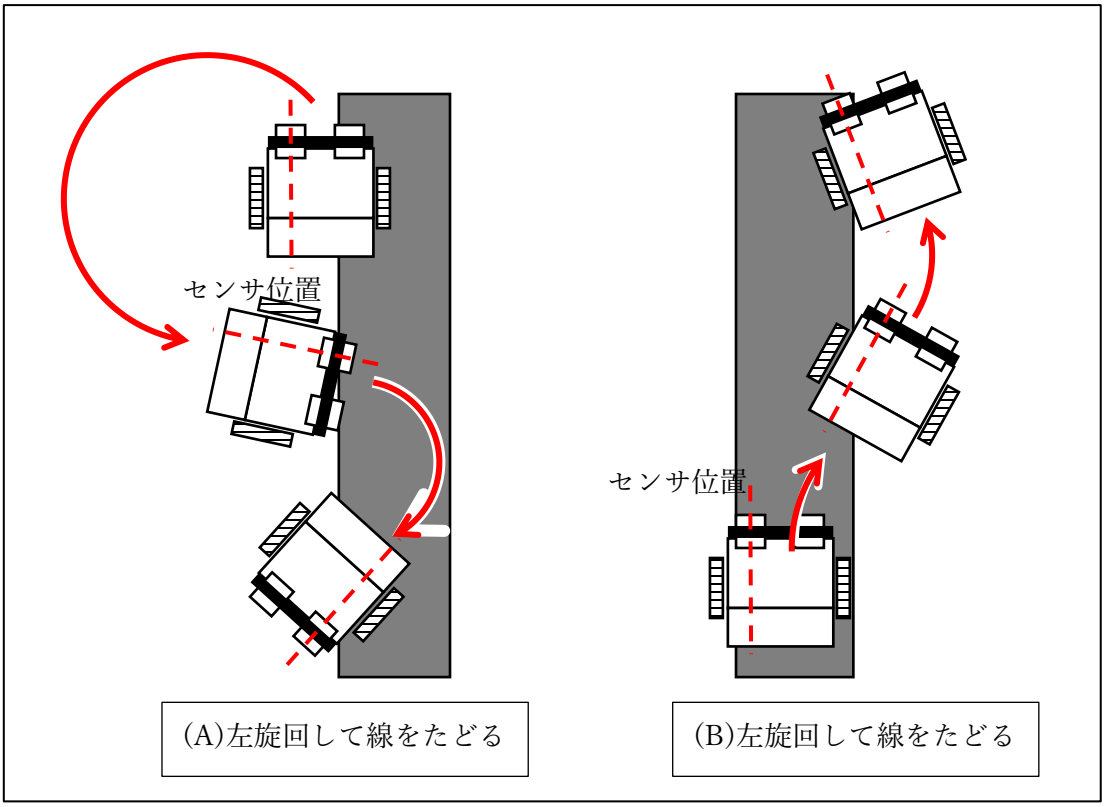


図 32：線の左側にロボットがある場合の動作例

線の太さ、ロボットの初期位置や進行させたい方向に対する角度、ロボットの旋回時の半径などの条件によっては線の右側から左側に飛び出して行ってしまうなど、いろいろなケースが考えられます。

7. マイク入力

SmallBot のマイコンボード上にマイクがついていて、周囲の音声の大きさ（騒音レベル）を知ることができるようになっています。

マイクは図 33 に示すように、正面パネルの右下のところにあります。

金属ケースに小さい穴の開いているデバイスがマイクです。

サンプルプログラムでは、音に反応して進行方向を変える動作をするダンスモードでマイク入力を利用しています。

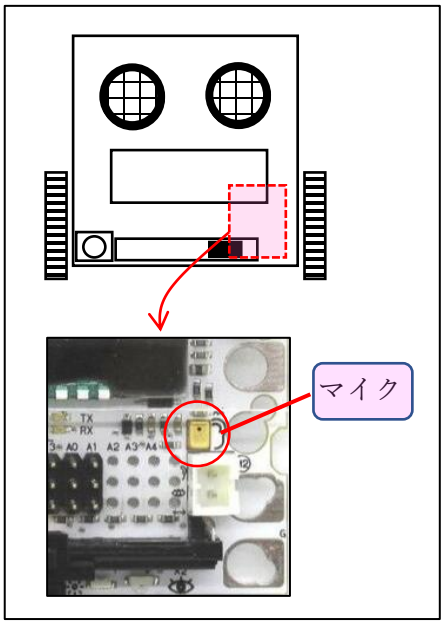


図 33：SmallBot のマイクの位置

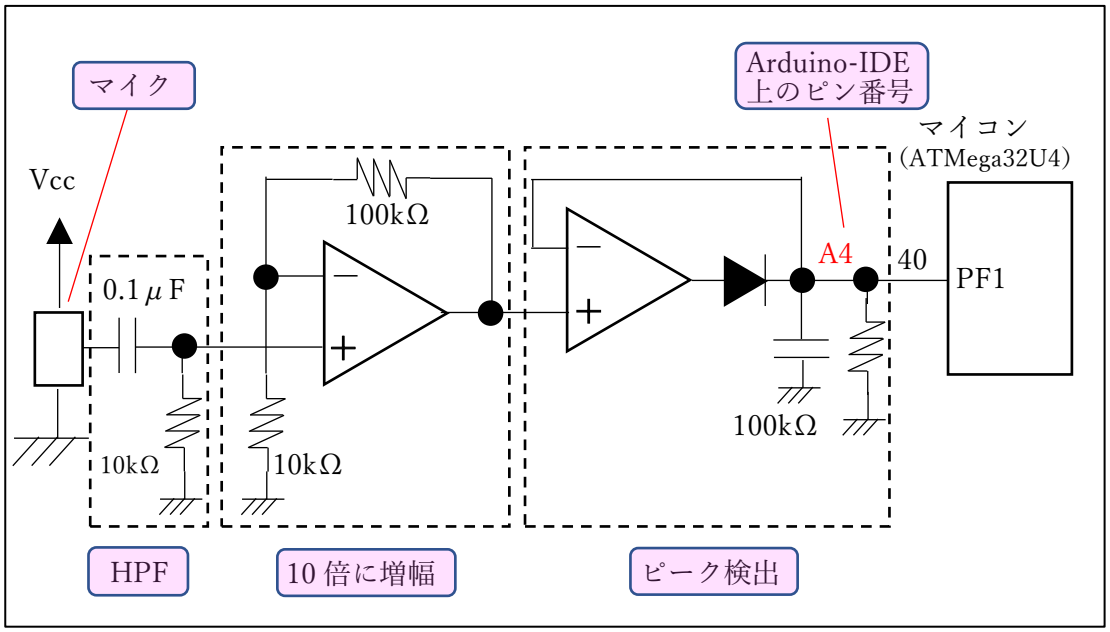


図 34：マイク入力回路

7.1. マイク入力のハードウェア

7.1.1. マイク入力回路

図 34 は SmallBot のマイク入力回路です。左端にあるのがマイクです。

マイクから出力された音声信号は、HPF（ハイパスフィルタ）によって直流成分を含めた低い周波数をカットしてから増幅し、さらにピーク検出回路によって音声波形のピーク値を抽出します。大きい音が入ってくると電圧が高くなり、小さい音の時は電圧が低くなるという具合に、音声の大きさに応じた電圧が得られます。

これをマイコンでアナログ値として取り込むことで、周囲の騒音レベルを知ることができます。

7.1.2. HPF

マイクの音声信号出力は、コンデンサと抵抗でできた HPF（ハイパスフィルタ）をとおります。

1V の正弦波の周波数を変化させながら、出力レベルがどのように変わるのかをシミュレーションした結果が図 35 です。

HPF 部分の回路を書き直すと図の上にあるような、コンデンサと抵抗の直列回路になっています。

入力する信号の周波数を f(Hz)、コンデンサの容量を C（F)、抵抗の値を R（Ω）として、入力電圧（VIN）と出力電圧（VOUT）の関係を考えてみます。コンデンサのインピーダンス（交流に対する抵抗のようなものです）Z[Ω]は

$Z = \frac{1}{j \cdot 2\pi fC}$[Ω]（j は虚数単位）となりますので、

$VOUT = \frac{R}{\frac{1}{\frac{j}{2\pi fc}} + R} \times VIN = \frac{1}{\frac{1}{j \cdot 2\pi fCR} + 1} \times VIN$

となります。周波数 f が大きくなると、$\frac{1}{j \cdot 2\pi fCR}$は 0 に近付きますので、

$VOUT = \left(\frac{1}{0+1}\right) \times VIN = VIN$

となり、VOUT は VIN に近付きます。

一方、f が小さくなると$\frac{1}{j \cdot 2\pi fCR}$の値が大きくなりますので、$\frac{1}{\frac{1}{j \cdot 2\pi fCR} + 1}$は 0 に近付き、$VOUT = 0 \times VIN = 0$となります(※)

図 35 を見ると周波数が低い時は出力レベルが小さくなり、周波数が高くなると入力と同じレベル（1V）に近付くことが確認できます。

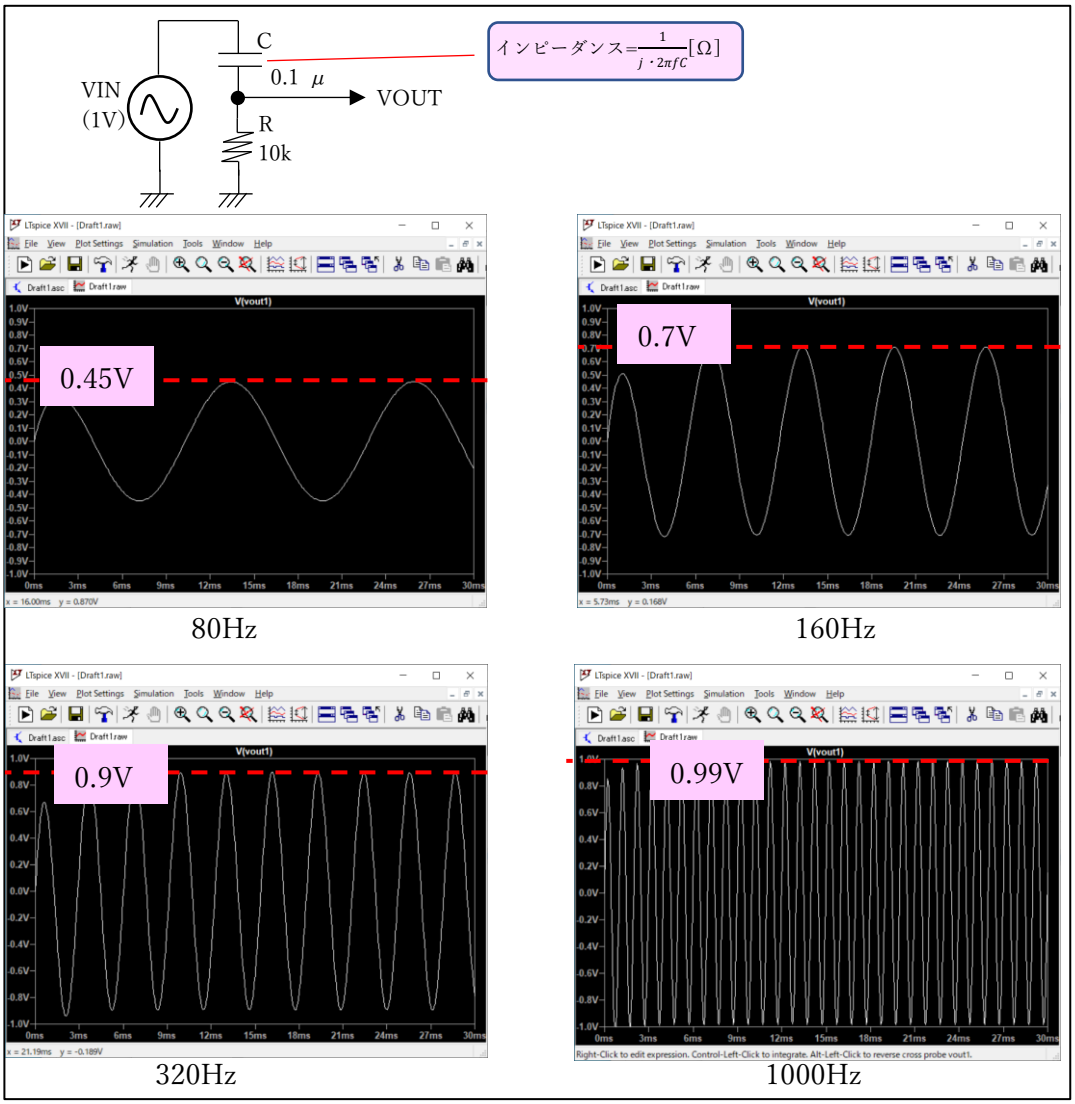


図 35：周波数と HPF 出力レベルの変化

※HPF の周波数の低い領域の特性

f(周波数)が十分小さければ、“+1”の影響が小さくなり、無視できるくらいになりますので、

$VOUT \fallingdotseq \frac{1}{\frac{1}{j\cdot 2\pi fCR}} \times VIN = j\cdot 2\pi fCR \times VIN$

とみなせます。f 以外を固定値にすると、周波数と VOUT が比例します。つまり、周波数が 2 倍になれば、VOUT が 2 倍、$\frac{1}{2}$になれば、VOUT も$\frac{1}{2}$になります。

7.1.3. HPF のカットオフ周波数

$\frac{1}{2\pi fCR}=1$ のとき、すなわち $f=\frac{1}{2\pi CR}$ のときにどうなるか考えると

$VOUT=\frac{1}{-j+1}\times VIN$

$=\frac{1+j}{(1-j)(1+j)}\times VIN=\frac{1+j}{(1-j)(1+j)}\times VIN$

$=\frac{1}{2}(1+j)\times VIN$

ここで、$VIN=V\sin\theta$

という

$VOUT=\frac{1}{2}(1+j)\times V\sin\theta$

ここで $\cos\theta=j\sin\theta$ より

$VOUT=\frac{1}{2}V(sin\theta+cos\theta)$

$=\frac{1}{2}V(\sqrt{2}\sin(\theta+\frac{\pi}{4})$

$=\frac{1}{\sqrt{2}}V(\sin(\theta+\frac{\pi}{4})$

となって位相が$\frac{\pi}{4}$(45 度)進み、出力電圧は入力の$\frac{1}{\sqrt{2}}\fallingdotseq 0.7$ 倍になります。

ここで C=0.1 μF、R＝10kΩを代入すれば、

$f=\frac{1}{2\pi CR}\fallingdotseq 159(\mathrm{Hz})$

のときに電圧が 0.7 倍程度になり、これよりも低い周波数では急激に電圧が低下します。

この、電圧が $\frac{1}{2}$ になる周波数を「カットオフ周波数」と呼んでいます。

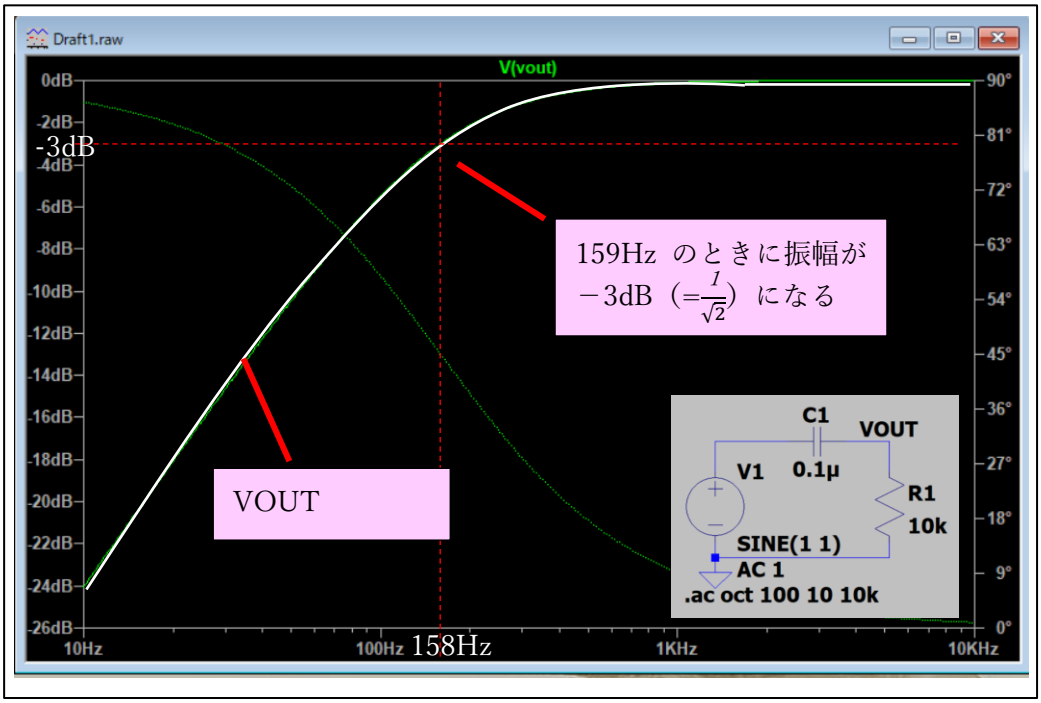


図 36：マイク入力の HPF の特性

7.1.4. 増幅回路

マイクの出力電圧は HPF を通ったあとの信号はレベルが小さく、そのままではマイコンで扱うのに不便なため、増幅回路で信号レベルを大きくします。図 37 は SmallBot のマイク入力信号の増幅回路です。

中央にある三角形のものがオペアンプと呼ばれるものです。図のように抵抗をつないだ回路は非反転増幅回路と呼ばれるもので、増幅率は

$\frac{R1+R2}{R1} \times VIN = VOUT$

となります。図では R1=10 k Ω、R2=100kΩですので、増幅率は 11 倍になります。

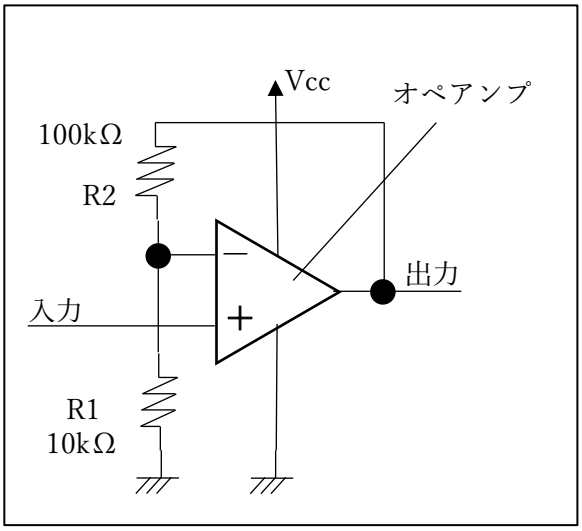


図 37：マイク増幅回路

7.1.5. オペアンプの基本動作

オペアンプは増幅回路や次で説明するピーク検出回路でも使われていますので、ここで基本的な動作を説明しておきましょう。

図 38 はオペアンプのシンボルと基本的な動作を示した物です。オペアンプは非常に大きな増幅率を持った増幅回路です。＋入力と－入力の二つの入力があり、この差分を増幅して出力電圧にします。

式で表すと、オペアンプの増幅率（オープンループゲインと呼びます）をβとすると出力電圧は

出力電圧＝((+入力電圧)－(－入力電圧))×β

となります。

ただし、入出力とも電源電圧の範囲内（この場合、0～Vcc）に制約されます。

オペアンプのオープンループゲインは非常に大きな値ですので、＋入力と－入力の差が極めて小さい時を除いて

・[+入力]>[-入力]ならば出力は Vcc
・[+入力]<[-入力]ならば 0V

となります。（実際には Vcc より少し低く、0V より少し高い電圧が限界になります）

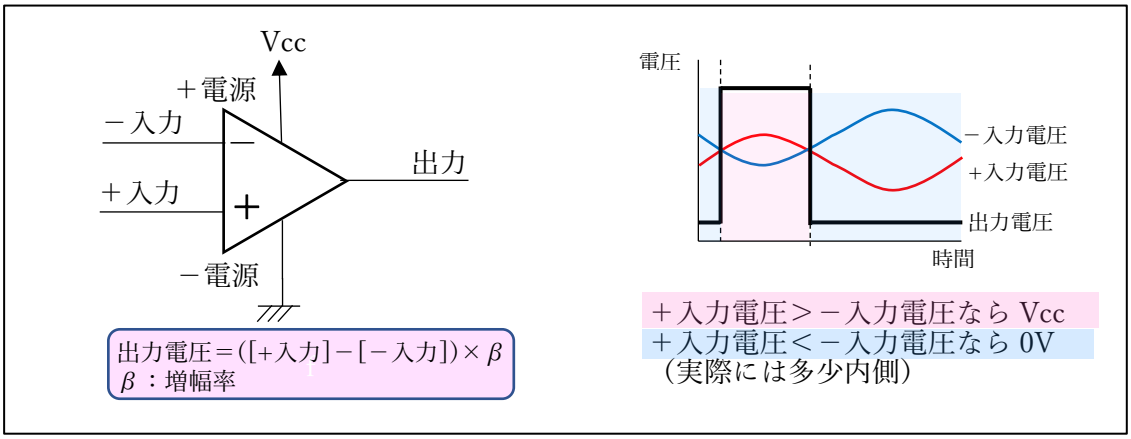


図 38：オペアンプの基本動作

7.1.6. 増幅回路の動作

図 39 はオペアンプを利用した増幅回路（非反転増幅回路）の動作を示したものです。
オペアンプの＋入力電圧を X、－入力電圧を Vm、出力電圧を Y、オープンループゲインをβ、とすると、

$$Vm = Y \cdot \frac{R1}{R1+R2}$$

となります。またオペアンプ自身の増幅動作から、

$$Y = \beta \cdot (X - Vm)$$

となります（※）。

この二つの式を連立して Vm を消去し、Y と X の関係の式にしてみましょう。、

$$Y = \beta\left(X - Y \cdot \frac{R1}{R1+R2}\right)$$

$$Y \cdot \left(1 + \beta \cdot \frac{R1}{R1+R2}\right) = \beta \cdot X$$

$$Y \cdot \left(\frac{R1+R2+\beta \cdot R1}{R1+R2}\right) = \beta \cdot X$$

ここから

$$Y = \left(\frac{R1+R2}{R1+R2+\beta \cdot R1}\right) \cdot \beta \cdot X$$

$$= \left(\frac{R1+R2}{\frac{R1+R2}{\beta}+R1}\right) \cdot X$$

βが十分に大きければ $\frac{R1+R2}{\beta} = 0$ と見なせますので、

$$\boldsymbol{Y = \left(\frac{R1+R2}{R1}\right) \cdot X = \left(1+\frac{R2}{R1}\right) \cdot X}$$

となります。
図のように R1=10kΩ、R2=100kΩならば
Y=11・X
となって、増幅率 11 倍の増幅器として動作することになります

※：式をお変形すると $(X - Vm) = \frac{Y}{\beta}$ となりますので、βが極めて大きい値ならば、
$X - Vm = 0$、つまり $X = Vm$ と見なせます。

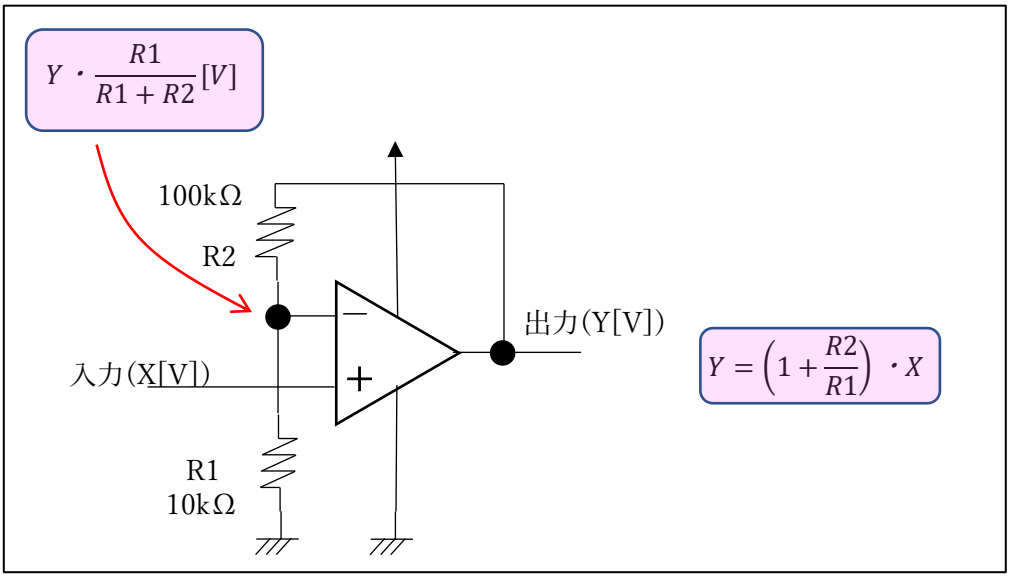


図 39：非反転増幅回路の動作

7.1.7. 非反転増幅回路の入出力関係

非反転増幅回路の入出力の関係をグラフとして表したのが図 40 です。

オペアンプの出力は電源電圧の範囲内（この回路であれば 0～Vcc）までしか振れません。また、実際にはオペアンプの最高出力電圧は Vcc よりも少し低くなります。オペアンプの種類や出力電流にもよりますが、1～2V 以上になる場合もあります。SmallBot で使われているオペアンプは出力電流が 50μA のときに 1.35V 程度となっています。

また、最低出力電圧も 0V までは落ちず、1～2V 程度あることもあります。SmallBot に使われているオペアンプではこの値は 0.1V 程度となっています。

非反転増幅回路の出力電圧の計算値がこの範囲外になった場合、図のようにリミットがかかったようになります。

図 41 は入力信号に対してどのような出力波形が得られるのかを図示したものです。グラフの下側のような入力信号があった場合、図の右のように正負両方に振れた場合、＋側だけが残されて、－側に振れた分が切り取られたような波形になります。

図には描いていませんが、＋側に大きく振れたときは出力電圧が Vcc 近くでリミットがかかりますので、頭が潰れたような波形になります。

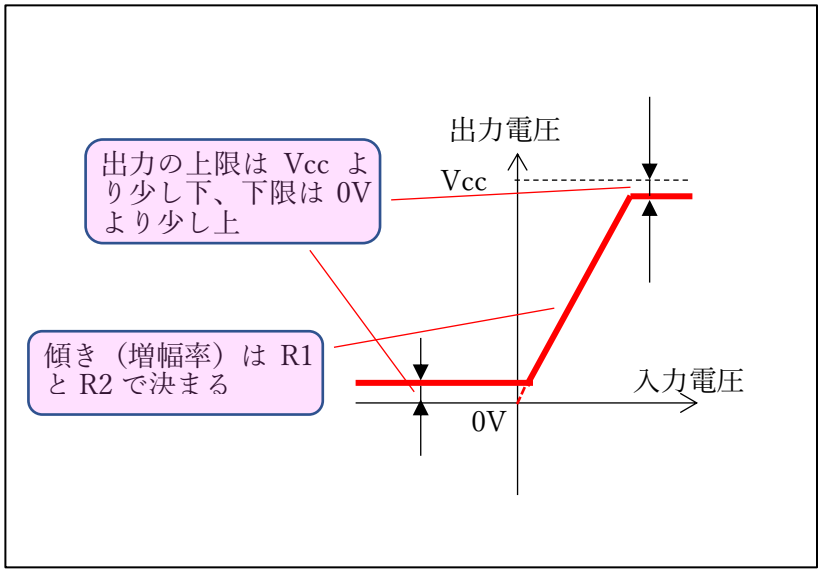


図 40：非反転増幅回路の増幅動作

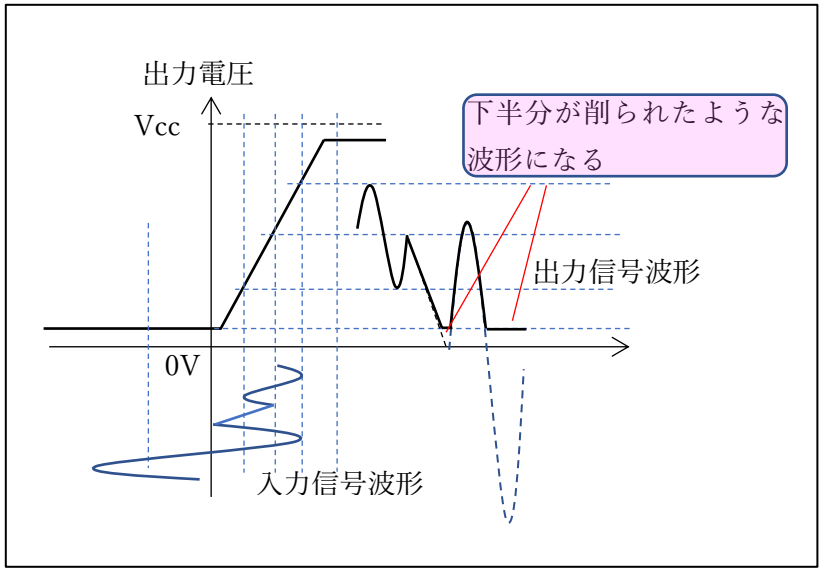


図 41：非反転増幅回路の出力波形

7.1.8. ピーク検出回路

図 42 はピーク検出回路です。増幅された音声信号は音声の振動に合わせて激しく上下する波形ですので、マイコンなどで音声の大きさを知るのには不便です。

オーディオアンプなどに付いているレベルメーターのように、音声の波形に合わせた細かい上下動をせず、音声の大きさに応じて動くような信号を作るのがピーク検出回路です。

ピーク検出回路のオペアンプの出力にはダイオードが付いていて、ダイオードの先が出力（VOUT）になっています。ダイオードは一方方向にだけ電流が流れるというもので、図であれば、左から右にだけ電流が流れ、逆向きには流れません。

ダイオードの先にはコンデンサがついていています。これがピーク電圧を保持するためのものです。ダイオードが付いているため、VOUT 電圧よりも OpOUT の電圧くなるとき、すなわち VIN>VOUT の時だけ、コンデンサに充電されることになります。

コンデンサと並列につながっている抵抗はコンデンサの放電用です。

オーディオのレベルメーターは、上昇方向は素早く応答しますが、下降方向はゆっくり下がっていきます。これと同じように抵抗を通して少しずつ放電することで、コンデンサの電圧が少しずつ下がるようにしているわけです。

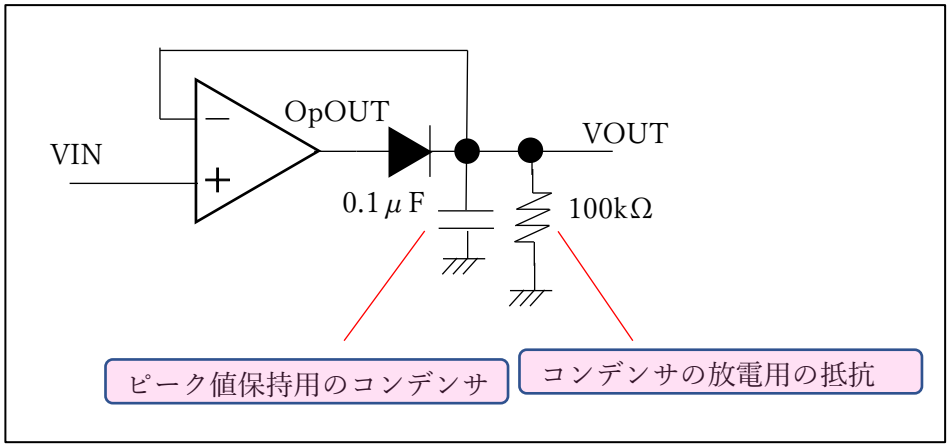


図 42：ピーク検出回路

7.1.9. ピーク検出回路の基本動作

図 43 はピーク検出回路の動作を表した物です。

VIN の電圧が VOUT より大きいとき、つまり現在保持しているピーク値よりも大きな入力があったときは、オペアンプの＋入力＞－入力となりますので、オペアンプの出力は Vcc 側に大きく振れようとして、ダイオードを通してコンデンサに充電します。このときの VOUT が－入力に入っていますので、＋入力＝－入力、つまり、VIN=VOUT となったところで安定します。

VIN<VOUT のときは図の下側のように、オペアンプの出力は 0V 側に振れますが、ダイオードがついているために VOUT には影響しません。

コンデンサが抵抗で放電されて少しずつ VOUT の電圧が下がり、VIN よりも低くなってくると、VIN>VOUT の時と同じ動作となり、VIN=VOUT になります。

この動作を波形で示したのが図 44 です。左端の VIN>VOUT の領域では、VOUT は VIN よりも電圧が高い

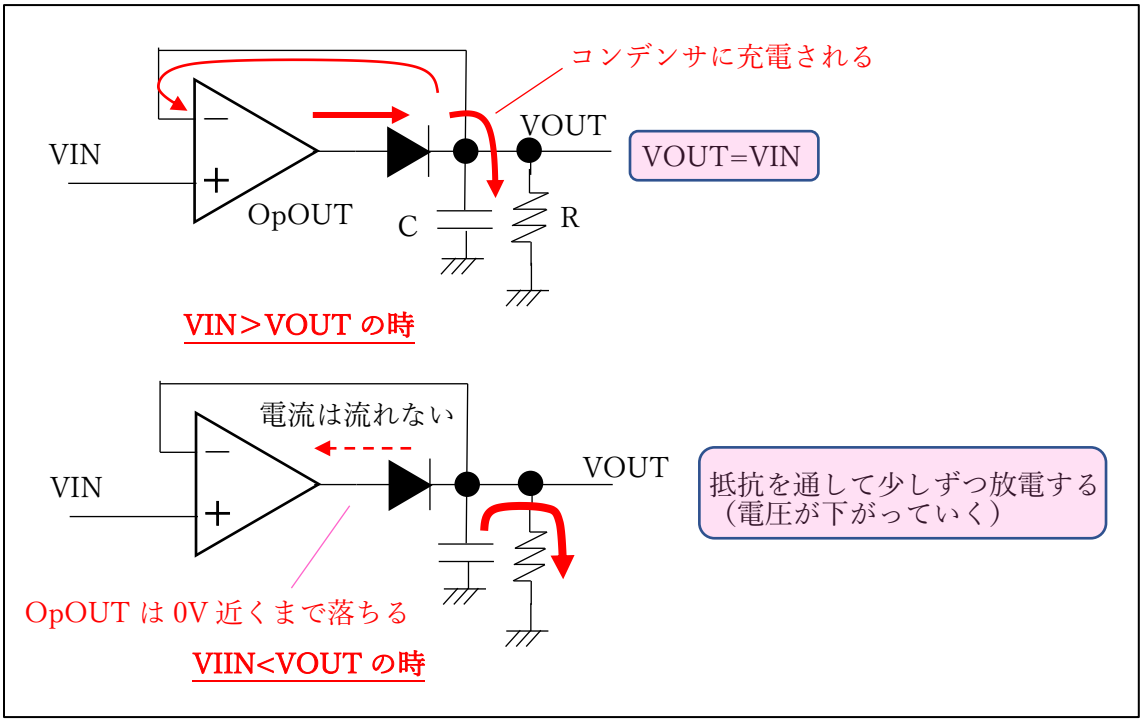


図 43：ピーク検出回路の考え方

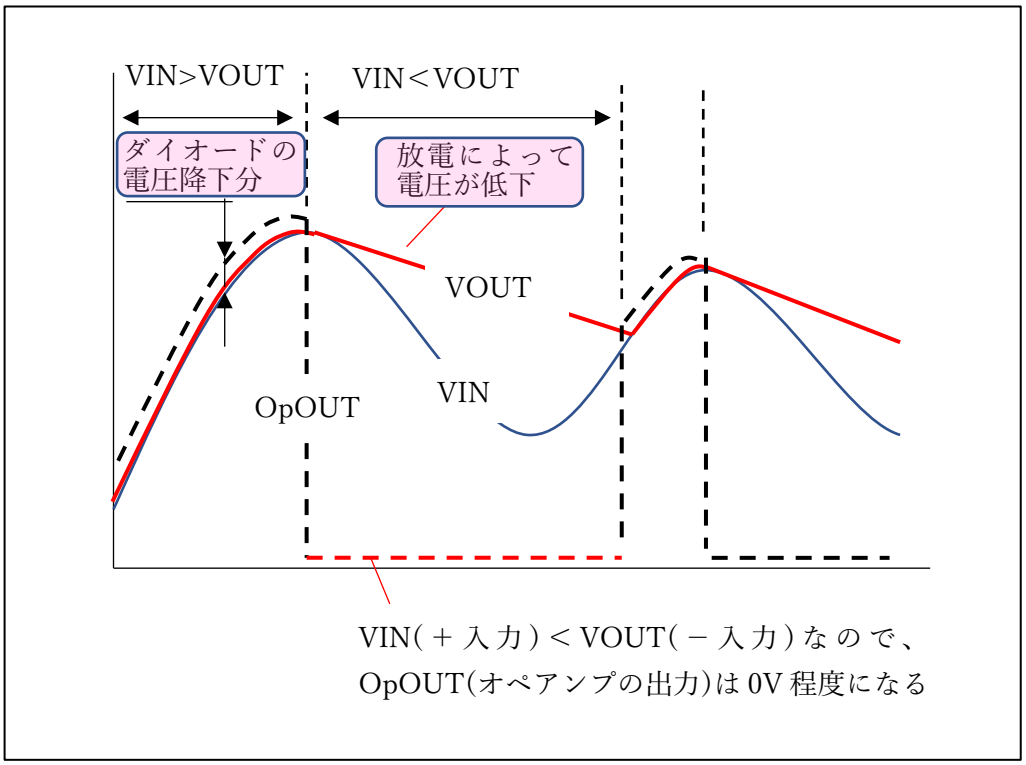


図 44：ピーク検出回路の動作

ため、VOUT が VIN と等しくなるように OpOUT の出力が追従していきます(※)。

山のピークを越えて、VIN が減少に転じると、今度は VIN<VOUT になりますので、OpOUT は 0V 近くまで低下します。ダイオードが無ければ VOUT 電圧も低下するのですがが、ダイオードがあるため VOUT の電圧は下がりません。

コンデンサに並列につながれた抵抗を通してコンデンサが少しずつ放電し、VOUT 電圧が下がっていきます。

VIN が再び VOUT 以上のレベルになれば、最初と同じように VIN を追いかけるように VOUT が上昇します。

※：VOUT=VIN の時、オペアンプの出力（OpOUT の電圧）はダイオードの降下電圧分だけ少し高めになります。

7.1.10. ピーク検出回路のシミュレーション

図 45 はピーク検出回路のシミュレーション回路です。

左端の電源はオペアンプ動作用の電源で、中央の二つの電源は入力信号の生成用です。正弦波を生成する二つの電源を直列にすることで二つの音が混ざった「和音」のようなものにしてみました。

この回路のシミュレーション結果が図 46 です。

オーディオのレベルメータの動作と同じように、上昇方向はすぐに追従し、下降方向はゆっくり下がるように VOUT が変化していることがわかります。

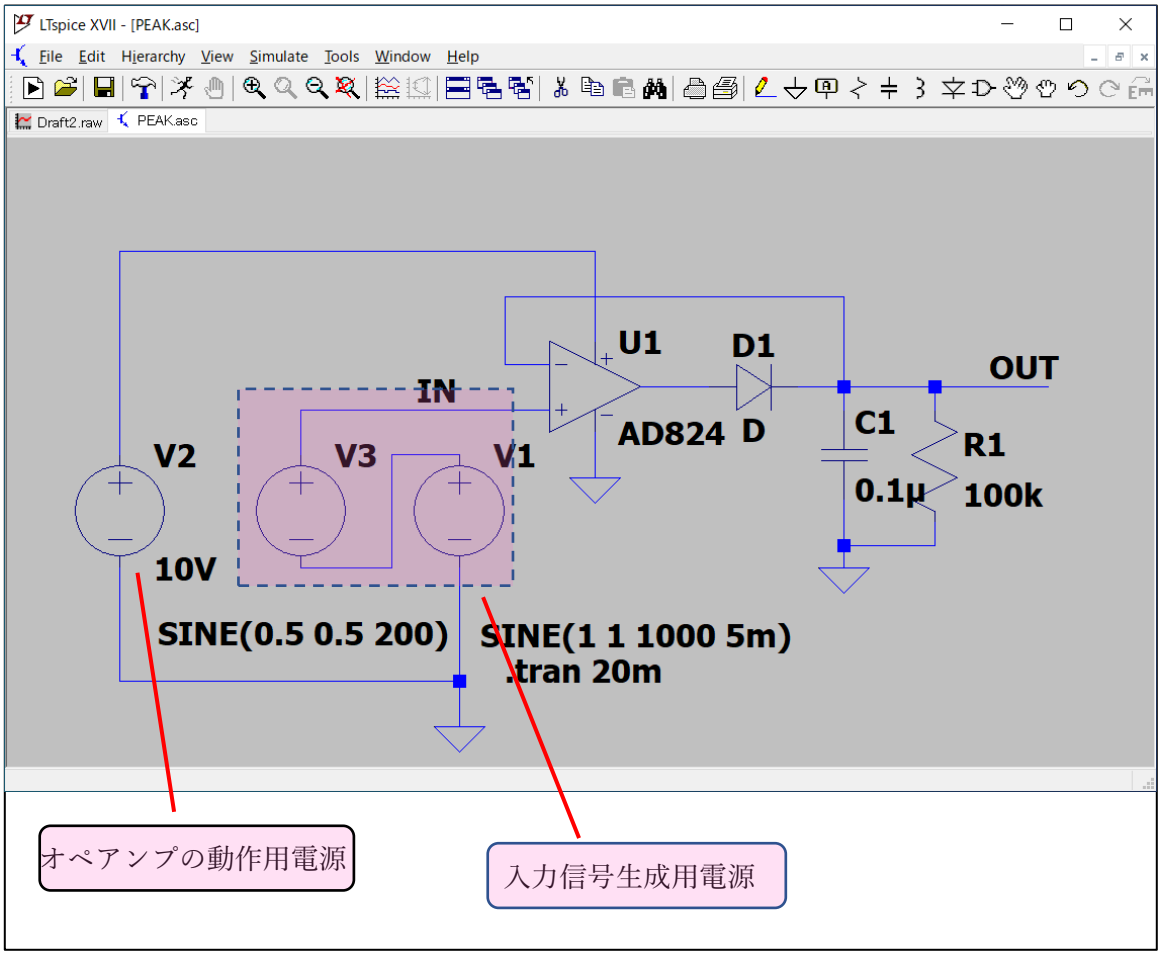


図 45：ピーク検出回路シミュレーション回路

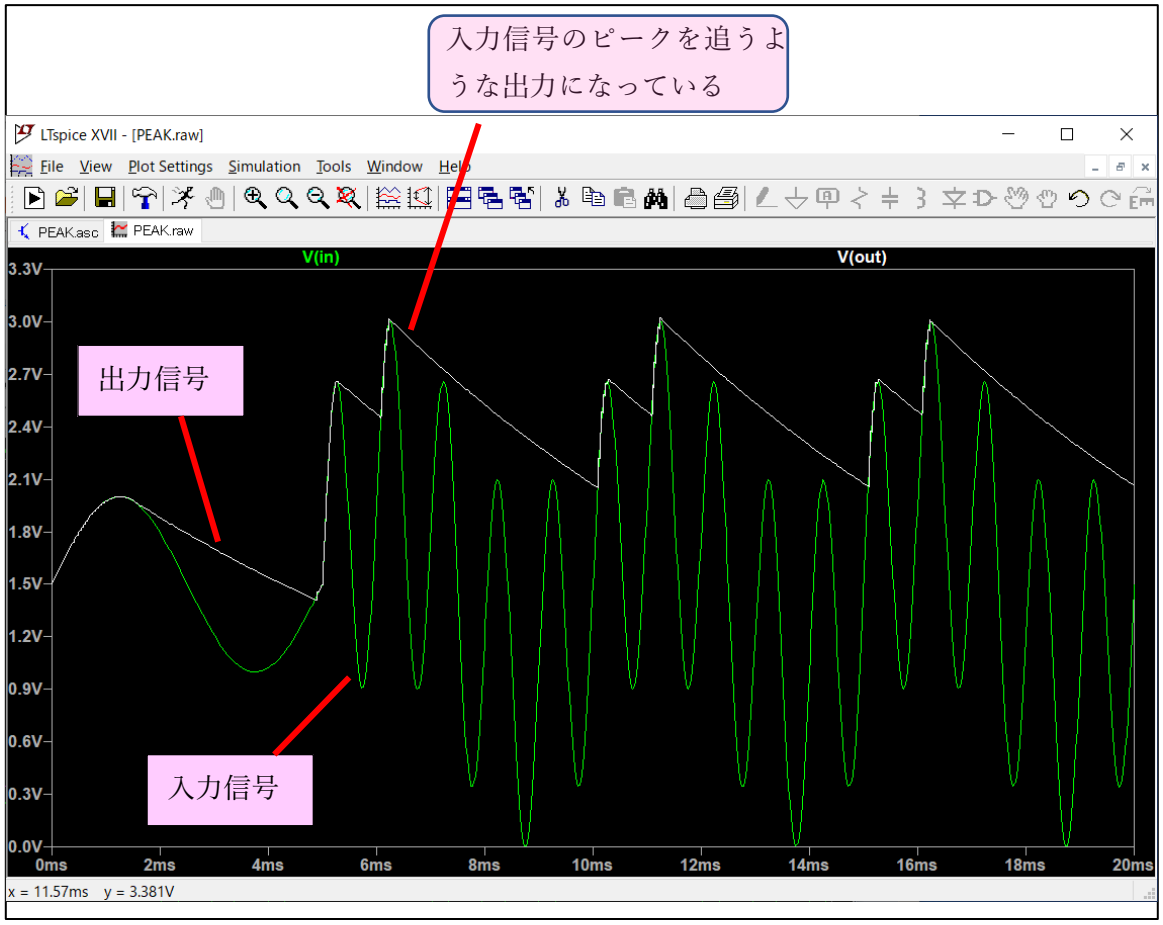


図 46：ピーク検出回路シミュレーション結果

7.1.11. マイク入力回路のシミュレーション

マイク入力回路全体のシミュレーションを行ってみました。シミュレーションに使用した回路図が図 45、シミュレーションした結果が図 46 です。

HPFOUT 出力は 0V を中心にして正負両方に振れる信号になります。

増幅回路で増幅できるのは入力信号のプラス側だけです。シミュレーション結果でも AMPOUT はプラス側（0V 以上）の部分だけが増幅された波形になっています。

この増幅回路の出力（AMPOUT）波形のピークが検出されて OUT 出力になっています。

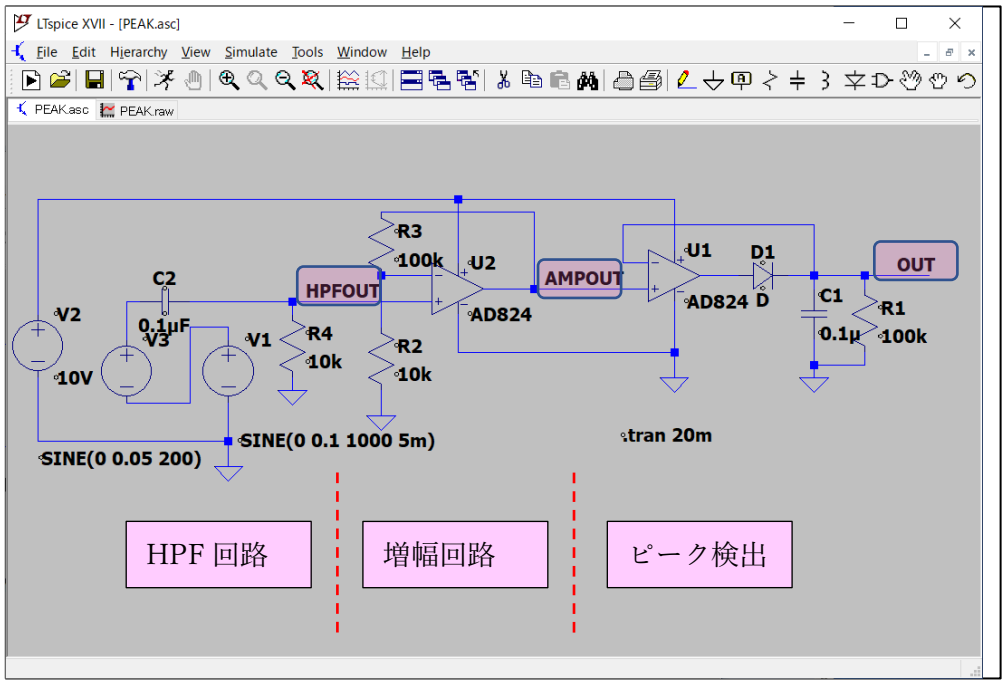


図 47：マイク入力回路シミュレーション回路

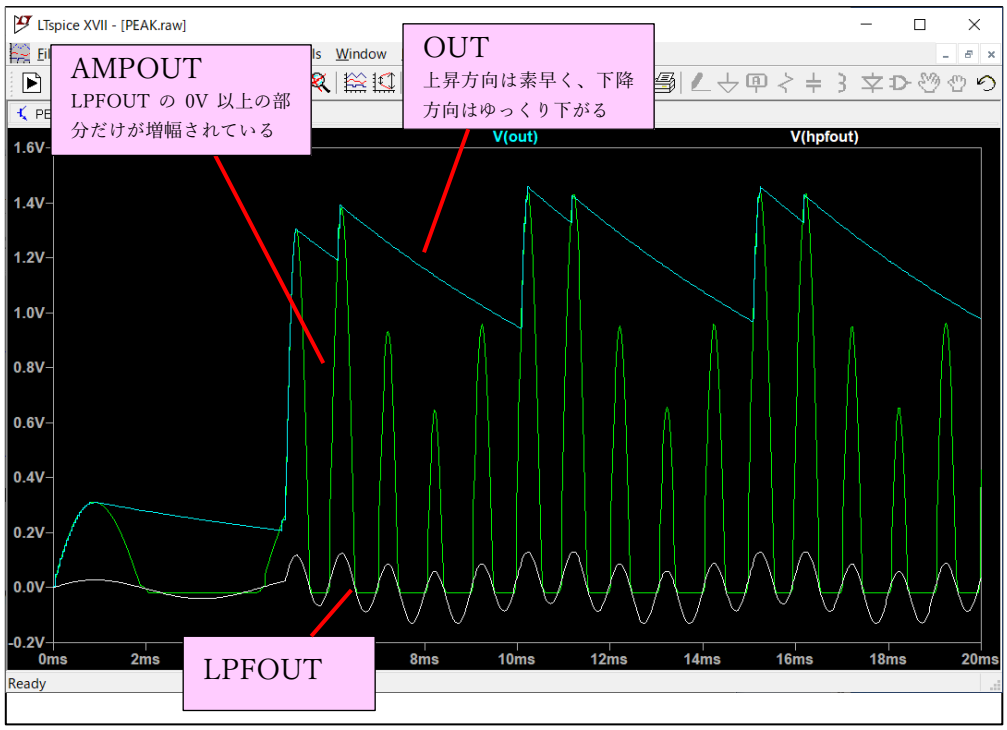


図 48：マイク入力回路シミュレーション結果

7.2. マイク入力のソフトウェア

図 49 はサンプルプログラムからマイク入力による動作（大きな音があると方向転換する）処理の一部を抜き出してきた物です。
マイク入力は HPF や増幅回路、ピーク検出回路を通った後、A4 ピン（Arduino-IDE 上でのピン番号）に入力されています。この電圧を analogRead()関数を使用してアナログ値として読み込みます。電圧値は 0～1023 の数値で返されます。

このようにして得られた音声レベル値がスライダで設定した値を超えると右回転するようにしています。

```
  unsigned int insound;
  unsigned int SoundLevel = 0;
  insound = analogRead(A4);          // 音声レベルを取得
  if (SoundLevel < insound)          // 音が大きかったら
    SoundLevel = insound;            // レベル更新

・・・
  sliderValue = analogRead(A3);                 // スライダーの読み取り
  if(map(sliderValue, 0, 1023, 10, 500) < SoundLevel) // 音声レベルの方が大きい
    RStateSOUND = RBT_SOUND_ROT;                // 右回転
  else    RStateSOUND = RBT_SOUND_FORWARD; // 小さかったら直進
    SoundLevel = 0;   // 音声レベルをクリア
・・・
```

図 49：サンプルプログラムの音声入力処理

8. 超音波距離センサ

SmallBot には前方の障害物までの距離を計測する、超音波距離センサモジュールが取り付けられています。超音波距離センサモジュールは図 50 のように、ロボット前面パネルの上部にあります。

サンプルプログラムでは超音波距離センサを利用して、障害物を見つけると方向転換する動作（ダンゴムシモード）を行わせています。

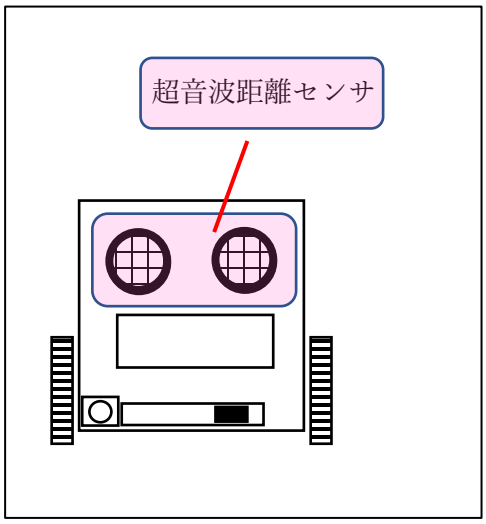


図 50：超音波距離センサの位置

8.1. 超音波距離センサのハードウェア

8.1.1. 超音波距離センサの外観

SmallBot に使われている超音波距離センサモジュールの外観は図 51 のようになっています。モジュールには金属製の円筒型の部品が二つ、左右に配置されています。両方とも同じ形をしていますが、向かって左側が超音波スピーカー（超音波送信機）で、右側が超音波マイク（超音波受信機）です。

モジュールの下には 4 本の端子があります。2 本が電源端子（Vcc と Gnd）で、残る 2 本（Trig と Echo）が距離計測動作用の信号です。

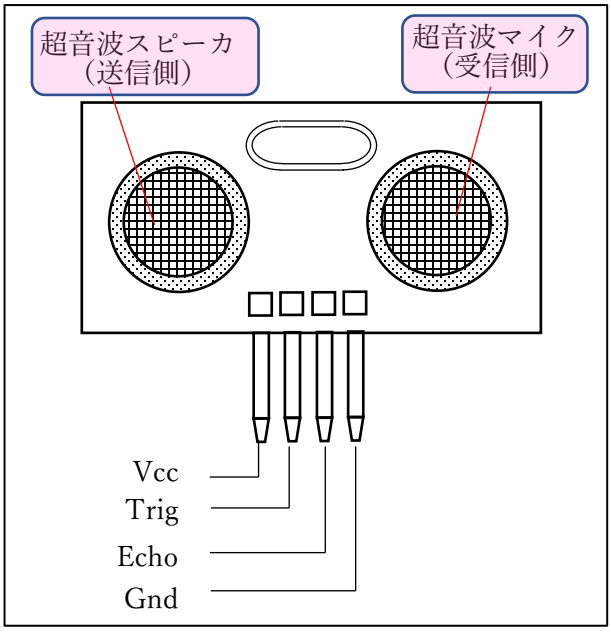


図 51：超音波距離センサの外観

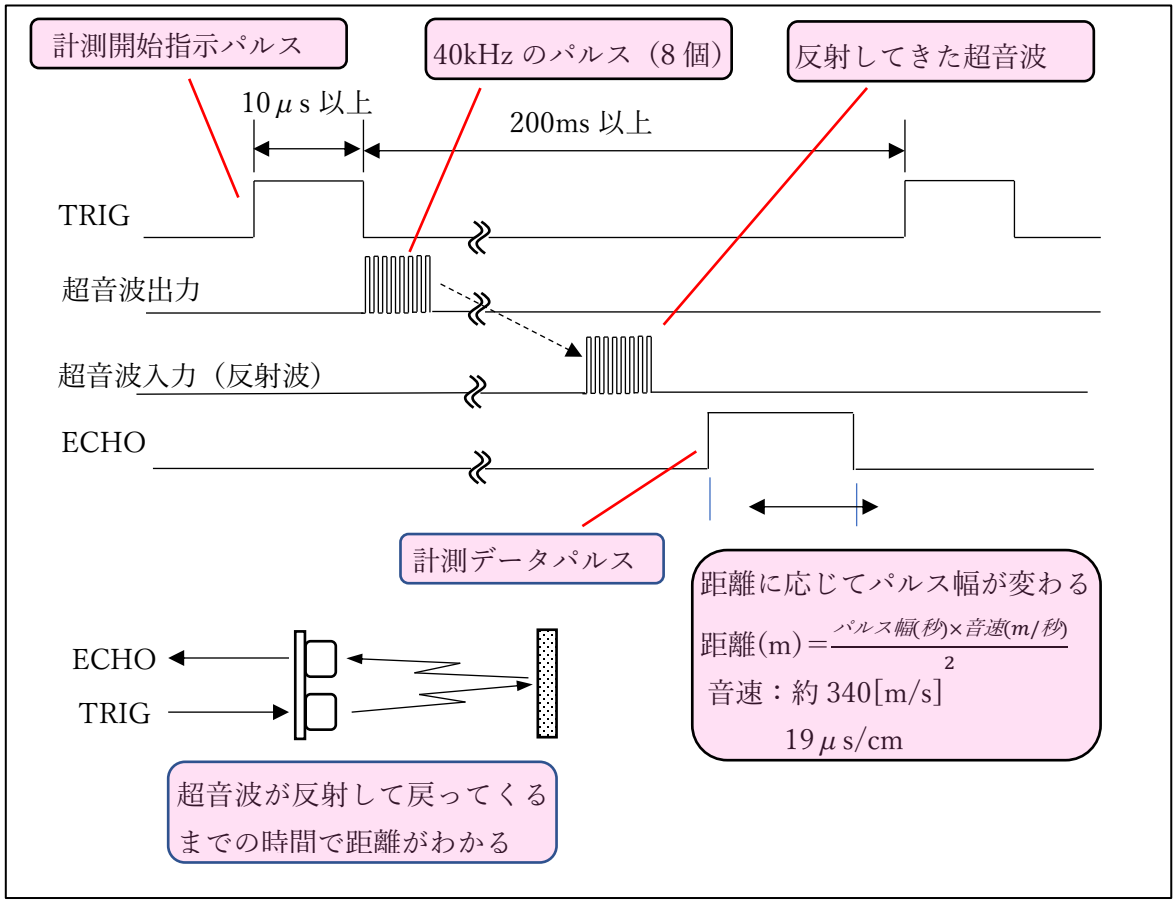


図 52：超音波距離センサモジュールの動作

8.1.2. 超音波距離計測の仕組み

図 52 は超音波距離センサモジュールの動作を示したものです。

超音波距離センサは、図の左下のようにスピーカーから超音波を発信し、物に当たって跳ね返ってきた反射音をマイクで検出します。

超音波を発信してから反射音が受信されるまでにかかった時間を音速で割ると往復の距離がわかります。

音波を反射した物体までの距離は片道分の距離ですから、このようにして算出された値の $\frac{1}{2}$ になります。

8.1.3. 超音波距離計測動作

図の上側は距離計測動作の信号波形を示したものです。

(1) センサの TRIG 入力に 10μs 以上の幅の HIGH レベルパルスを与えるます。モジュールが計測動作を開始します。

(2) 超音波マイクから 40KHz の超音波が超音波スピーカからごく短い時間（8 サイクル：200μs）出力されます。

(3) 超音波が物体に当たって反射した反射音を超音波マイクで受け取ります。

(4) 反射音が到達すると、超音波を発信してから反射音が受け取られるまでの時間分だけ、ECHO 信号が HIGH レベルになります。

反射音が到達するまでの時間は反射する物体までの距離によって変化します。

反射音は物体までの間を往復していますので、2 倍の距離を伝わってきます。反射音が到達するまでの時間を t (秒)、音速を s(m/秒)とすれば物体までの距離 L[m]は

$$L[m] = \frac{t \cdot s}{2}$$

として算出できます。音速は温度や気圧、大気の組成などによっても変わってきますが、通常は 340m/秒（≒29μs/cm）として計算すれば良いでしょう。

TRIG パルスの間隔は 200ms 以上必要となっていますので、1 秒間に 4，5 回程度の計測が可能です。

超音波計測モジュールで計測可能な距離は 2cm から 4.5m となっています。この範囲外になるようなごく近い距離や遠い距離は計測できません。

8.1.4. 超音波距離センサ回路図

超音波距離計モジュールとマイコンの接続を図 53 に示します。

超音波距離計モジュールは超音波発信を指示する TRIG 入力信号と、超音波の反射音が返ってくるまでの時間を示す ECHO 出力信号があります。

両者をそれぞれ別々のピンに接続するのが一般的ですが、SmallBot では TRIG と ECHO 信号出力が交互に行われることを利用して、両方をまとめて 1 本のピンで済ませています。

TRIG ピンを駆動して超音波を送信するときにピンを出力モードにしてパルス信号を出力します。パルス出力が終わったらピンを入力モードにして、超音波距離センサからの ECHO パルス出力が来るのを待ちます。(※)

※：TRIG ピンと ECHO ピンが共有されていますので、ECHO 出力パルスはそのまま TRIG 信号に入ります。
これで再び超音波発信されると無限ループのようになってしまいますが、実際には 200ms 以内（往復で 6.4m、対象物まで 3.4m に相当）の TRIG 信号は無視しますので、問題ありません。

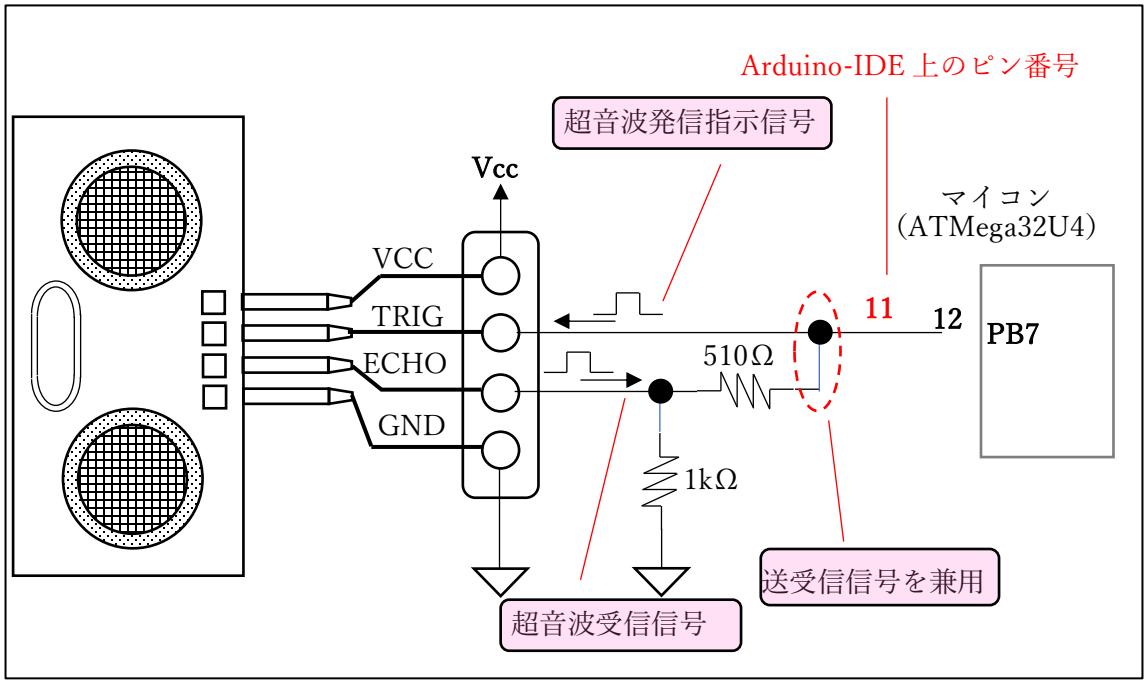


図 53：超音波距離センサ回路図

8.2. 超音波距離センサソフトウェア

図 54 はサンプルプログラムの超音波距離計測処理部分を抜き出してきたものです。

distance()関数を呼ぶと、戻り値で対象物までの距離を cm 単位で返します。この関数の中では、

(1) ピンを出力モードにする
(2) LOW 出力（0 書込み）
(3) HIGH 出力（1 書き込み）
(4) 5μs 待つ
(5) LOW を出力

として TRIG 信号を出力します。

```
#define PING_PIN          11 // 超音波測距センサは 11 番ピン
long distance(void)
{
  long duration, cm;

  pinMode(PING_PIN, OUTPUT);          // ピンを出力モードにする
  digitalWrite(PING_PIN, LOW);        // 出力をいったん LOW レベルにする
  delayMicroseconds(2);               // 少し待つ
  digitalWrite(PING_PIN, HIGH);       // 出力を HIGH レベルにする
  delayMicroseconds(5);               // 5μs 待つ（5μsHIGH になる）
  digitalWrite(PING_PIN, LOW);        // 出力を LOW レベルに戻す

  pinMode(PING_PIN, INPUT);           // ピンを入力モードい戻す
  duration = pulseIn(PING_PIN, HIGH); // HIGH レベルパルスの幅を計測
  return( microsecondsToCentimeters(duration)); // cm 単位に変換
}
// μs から cm に変換
// 音速を 340m/s とすると、1cm あたり約 29μs なので
// 29 で割ると往復距離になる。片道分（対象物までの距離）はこの半分
long microsecondsToCentimeters(long microseconds)
{
 return microseconds / 29 / 2;   // 往復距離なので、対象物までの距離は 1/2
}
```

TRIG 出力

ECHO 入力

図 54：サンプルプログラムの音波距離計測部分

TRIG 信号出力が終わったらピンを入力モードに戻して、Arduino のパルス入力ライブラリ関数である pulseIn()関数で HIGH レベルのパルスの幅を計測します。

この関数では入力が HIGH レベルのを待ち、LOW レベルに戻るまでの時間（パルス幅）を μs（10^{-6}*秒*）単位で返します。1 秒間待ってもパルスが来なければ 0 が返ります。

計測された値は μs 単位の値ですので、microsecondsToCentimeters()関数で距離に変換します。

音速を 340m/秒とすると(※)1cm あたりの時間は、

$$\frac{1}{340 \times 100} \times 10^6 \fallingdotseq 29[\mu s]$$

となりますので、プログラム中では計測された時間を 29 で割って cm 単位の距離に変換し、更に 2 で割ることで片道分の距離（対象物までの距離）を算出しています。

※：温度と音速の関係
大気中で 1 気圧のとき、気温を T℃とすると、気温と音速の関係は 331.5 + 0.61T[m/秒]という関係がありますので、気温 15℃の時にほぼ 340m/秒になります。

9. キャラクタ LCD

SmallBot の正面中央部にはキャラクタ LCD（液晶ディスプレイ）パネルが付いています。

キャラクタ LCD パネルは英数字やカタカナなどの表示を行うものです（グラフィック表示は行えません）。

表示可能な行数は 2 行で、1 行の文字数は 16 文字です。

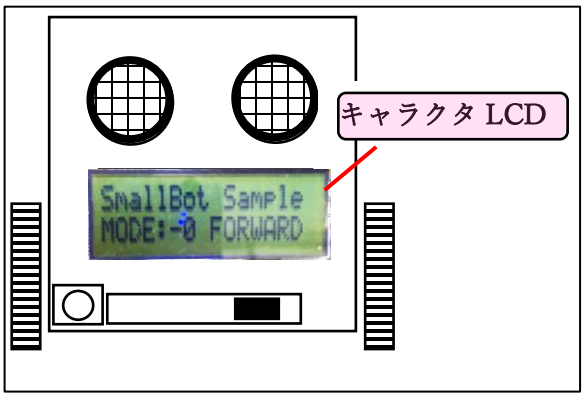


図 55：キャラクタ LCD の位置

9.1. キャラクタ LCD のハードウェア

SmallBot のキャラクタ LCD の接続は図 56 のようになっています。

SmallBot のキャラクタ LCD は I2C バスという 2 線式のシリアル通信インターフェースで接続されており、シリアル通信でコマンドを送って動かします。

SmallBot では Arduino-IDE のピン番号で 2 番ピンと 3 番ピンが I2C バス用になっています。

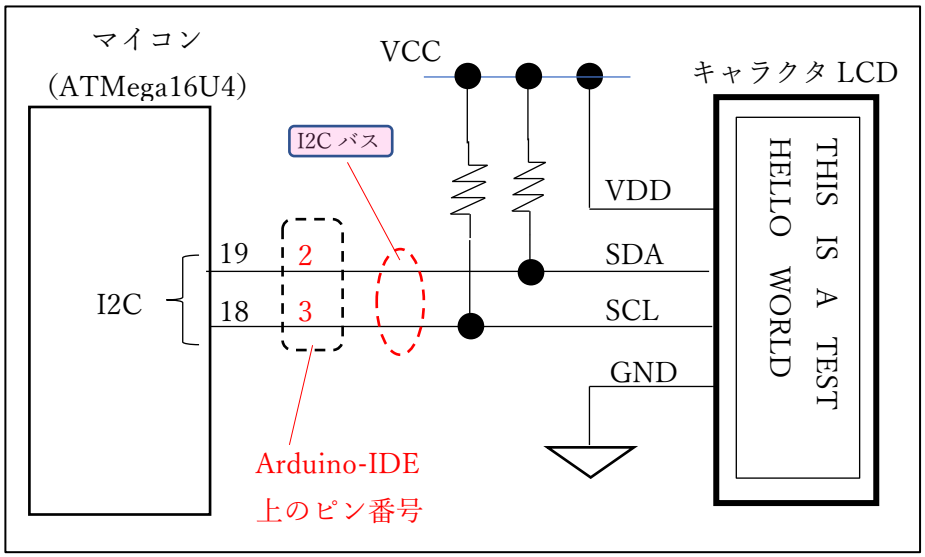


図 56：キャラクタ LCD の接続

9.2. キャラクタ LCD のソフトウェア

SmallBot のキャラクタ LCD は I2C バスで接続されています。Arduino-IDE には I2C バス接続の LCD を扱えるようにしたライブラリが用意されていますので、サンプルプログラムでもこれを利用しています。

図 57 はサンプルプログラム中でキャラクタ LCD の初期化や表示処理部分を抜粋したものです。

〇ヘッダファイルのインクルード

I2C 接続のキャラクタ LCD 用のライブラリの関数定義などはヘッダファイル、I2CliquieCrystal.h にまとめられています。プログラムの先頭部分で

#include <I2CLiquidCrystal.h>

としてヘッダファイルをインクルード（取り込み）します。

```
                    ヘッダファイル：I2CLCD ライブラリなどが定義されている
#include <I2CLiquidCrystal.h>      // LCD を使う。I2C_SDA：2 I2C_SCL:3
                                   LCD オブジェクトを作成
I2CLiquidCrystal lcd(40, false);   // StrawberryLinux 社扱い（I2C アドレス:0x3e)
                                   // コントラスト 40、3.3V 動作（5V ではない）
・・・・・
void DispMode(unsigned char Mode)
{
  lcd.setCursor(0,1);              // X,Y で開始位置指定（上から 2 行目の左端）
  lcd.print("MODE:-");             // 動作モード番号表示
  lcd.print(Mode);                 // Mode を数値表示
}

void setup() {
  lcd.begin(16,2);                 // 16x2 の LCD を利用
  lcd.setCursor(0,0);              // X,Y で位置指定(0,0)は左上隅
  lcd.print("SmallBot Sample");    // ”SmallBot Sample”と表示
  DispMode(0);                     // ”MODE:-0”と表示
・・・・・
}

void loop() {
・・・・・
        DispMode(RState);          // 現在の動作モードを表示
・・・・・

}
```

図 57：サンプルプログラムのキャラクタ LCD 制御（抜粋）

○LCD オブジェクトの作成

LCD を使用するため次のような記述で、LCD クラスから LCD オブジェクトを作成します。

```
I2CLiquidCrystal lcd(40, false);
```

I2CliquidCrystal というのがクラス名（変数の型のようなもの）で、次の lcd というのがオブジェクトの名称（変数の名前のようなもの）です。引数が二つありますが、第一引数（この例では 40）が LCD のコントラスト、第二引数（この例では false）が LCD が+5V 動作なのか否か（3.3V 動作なのか）を指定するものです。SmallBot では LCD を 3.3V で動作させていますので、第二引数は false にします。

○キャラクタ LCD の初期化

LCD オブジェクトを作成した段階では単にライブラリの基本的な初期化が行われただけで、ハードウェア（I2C バスや LCD パネル）の初期化は行われていません。

これらの初期化を行うライブラリ関数が begin()です。

```
lcd.begin(16,2);
```

のように、lcd オブジェクトの begin()関数（メソッド）を呼び出して I2C バスや LCD パネルの初期化を行います。

第一引数（この例では 16）が LCD パネルの 1 行の文字数、第二引数（この例では 2）が行数になっています。（※）

○カーソル位置の指定

LCD に文字を表示するときの表示開始位置（カーソル位置）を指定するのが、setCursor()関数です。

引数は二つで表示開始位置の X 座標（パネルの横方向のカラム位置)、Y 方向（パネルの縦方向の行位置）を指定します。

図 58 は setCursor()関数の引数と LCD パネル上の表示開始位置を示したものです。

X 方向は左端が 0、Y 方向は最上行が 0 です。たとえば

```
lcd.setCursor(0,1);
```

とすれば、X 方向は 0、Y 方向は 1 ですので、上から 2 行目（最上位が 0 行目なので、2 行目は 1 です）の左端を指定することになります。

※実は begin()は 3 つの引数を取り、三番目の引数で 1 文字のドット数を指定できるようになっています。引数を省略したときは 1 文字は 5×8 ドット（横 5 ドット、縦 8 ドット）の大きさという扱いになります。
SmallBot に使われているキャラクタ LCD も 5×8 ドットですので第三引数を省略しています。

○文字表示

キャラクタ LCD ライブラリで文字や数値の表示を行う関数が print()と println()関数です。println()関数は表示を行った後に改行する（表示した後にカーソル位置が一つ下の行の左端自動的に移動する）点が異なる他は print()と同じです。

print()関数は大きく分けて次の 4 つに分けられます。

・*print(整数, [DEC/HEX])*

引数の値を表示します。第二引数で表示を 10 進表示（DEC）にするのか、16 進表示（HEX）にするのかを指定します。省略すると 10 進になります。

例：print(32); ⇒ 32 と表示
　　print(32,16); ⇒ 20 と表示

・*print(浮動小数点値, 桁数)*

引数の値を少数で表示します。第二引数は小数点以下の桁数を指定します。省略すると 2 になります。

例：print(1.234,2);⇒1.23 と表示

・*print('文字')*

引数の 1 文字を表示します。

例：print('Z'); ⇒Z と表示

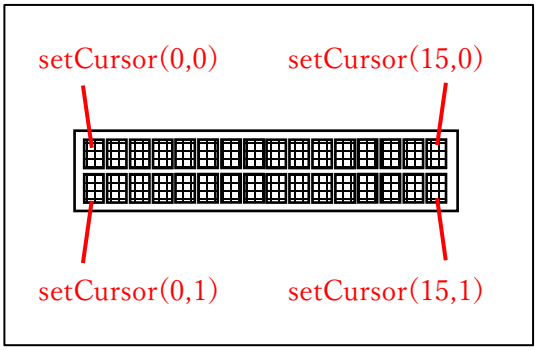


図 58：setCursor()の引数と表示開始位置

・*print("文字列")*

文字列を表示します

例：print("Hello"); ⇒Hello と表示

いずれの場合も print()ならば最後に表示した文字の位置、println()ならば一行下の左端にカーソルが移動します。(単に println()とすると、何も表示せず、カーソル位置が一つ下の行の左端に移動します)

カーソル位置を戻したり指定した位置から表示したいときは print()や println()を行う前に setCursor()でカーソル位置を指定します。

Version	日付	概要
1．0	2022/08/08	初版発行