



# mbed の利用方法

田内 康  
2012/01/04

## 目次

1	mbed を利用するための電子部品等の基礎知識 .....	1
1.1	mbed 概要 .....	1
1.2	簡単な電気回路(ハードウェア) .....	2
1.2.1	発光ダイオード (LED) .....	2
1.2.2	ステッピングモータ .....	3
1.2.3	ラジコン (RC) サーボモータ .....	5
1.2.4	プッシュスイッチ .....	6
1.2.5	ロータリエンコーダ .....	7
1.2.6	可変抵抗器(ポテンショメータ).....	8
1.2.7	スピーカ .....	8
1.2.8	圧電ブザー(圧電スピーカ).....	9
1.2.9	照度センサー .....	9
1.2.10	温度センサー.....	10
1.2.11	人感センサー.....	10
1.2.12	ソリッドステートリレー (SSR) .....	11
1.2.13	コイル系素子とマイコンの接続.....	11
1.2.14	信号レベルの異なる他の回路の信号の接続 .....	12
1.3	mbed 周りの周辺機器 .....	13
1.3.1	USB .....	13
1.3.2	Ethernet .....	13
1.3.3	シリアルインターフェイス(RS-232C) .....	14
1.4	☆Board Orange .....	15
1.5	m3pi Robot.....	16
2	mbed を使うための C++言語の知識.....	18
2.1	コンパイラのドキュメントの場所 .....	18
2.2	C Data Type .....	18
3	Hello World ! .....	19
3.1	ログイン .....	19
3.2	コンパイルと実行形式のファイルの保存 .....	19
3.3	実行 .....	21
3.4	問題 .....	21
4	ライブラリの作り方 .....	22
4.1	元のプログラムをサブルーチンに変更 .....	22

4.2	クラスに変更する.....	23
4.3	ファイルの分割.....	25
4.4	ライブラリの登録.....	26
4.5	ライブラリが判るようにドキュメントを記す.....	26
4.6	コード(*.h)に説明を書く.....	27
4.7	ドキュメントの確認.....	28
4.8	ライブラリを使う.....	29
4.9	問題.....	29
5	液晶キャラクタディスプレイ (LCD).....	30
5.1	TextLCD の命令(メソッド).....	30
5.2	Hello の表示(サンプルプログラムの実行).....	30
5.2.1	サンプルプログラムの読み込み.....	30
5.2.2	プログラムの修正.....	31
5.2.3	実行結果.....	31
5.3	数値を出力.....	32
6	タイマーや割り込み.....	33
6.1	Timer.....	33
6.2	wait.....	33
6.3	Ticker.....	33
6.4	Timeout.....	33
6.5	RTC.....	33
6.5.1	プログラム.....	33
6.5.2	実行結果.....	34
7	mbed についている USB を利用.....	35
7.1	USB を使用したシリアル通信.....	35
7.1.1	Windows のドライバーインストール.....	35
7.1.2	プログラムの作成.....	36
7.1.3	TeraTerm の立ち上げと設定.....	36
7.1.4	実行.....	36
7.2	mbed 内蔵の USB Flash メモリに保存(local file system).....	37
7.2.1	プログラム.....	37
8	SD カードの利用方法.....	37
8.1	ファイルの作成.....	37
8.2	ディレクトリの作成.....	38
8.3	ファイル一覧の作成.....	38
9	デジタル出力 (LED).....	39

9.1	変数を使う方法 .....	39
9.2	配列を利用する方法 .....	40
9.3	BusOut とシフトレジスタを使う方法 .....	41
10	PWM .....	42
10.1	PWM (LED) .....	42
10.1.1	プログラム 1 .....	42
10.1.2	プログラム 2 .....	43
10.1.3	応用 1 理解しましょう。 .....	43
10.2	PWM(モータを回す) .....	44
10.2.1	応用 1 .....	44
10.2.2	応用 2 .....	44
11	デジタル入力 (スイッチの動作回路) .....	45
11.1	回路 .....	45
11.2	プログラム .....	45
11.3	応用 1 (割り込み) .....	46
11.4	応用 2(チャタリングのチェック) .....	47
12	AD コンバータ .....	48
12.1	可変抵抗器 .....	48
12.2	温度センサ .....	49
13	USB スレーブ機能 .....	50
13.1	USB マウスをエミュレーションする .....	50
13.1.1	ハンドブックのサンプル実行 (USB Mouse Hello World) .....	50
13.1.2	アナログマウス (JoyStick の接続) .....	52
13.2	USB オーディオ .....	52
14	USB マスター機能 .....	53
14.1	USB フラッシュメモリの利用 .....	53
14.1.1	用意 .....	53
14.1.2	ファイルの書き込み .....	54
14.1.3	ディレクトリの作成 .....	54
14.1.4	ファイルの一覧表示 .....	55
15	m3pi で遊ぼう .....	56
15.1	Hello World .....	56
15.2	LED の使用 .....	57
15.2.1	前提知識 .....	57
15.2.2	プログラム .....	57
15.2.3	実行例 .....	57

15.3	ライントレースロボット .....	58
16	Ethernet の利用方法.....	59
16.1	はじめに .....	59
16.2	Web サーバ.....	59
16.2.1	はじめに(参考プログラムの読み込みと実行).....	59
16.2.2	普通の Web サーバ.....	61
16.2.3	制御用 Web サーバ (その 1) .....	64
16.2.4	制御用 Web サーバ (その 2) .....	66
16.3	NTP Client.....	70
17	IEEE1888 を利用する .....	78
17.1	IEEE1888SDK.....	78
17.2	FIAP Hello World.....	79
17.2.1	インポート .....	79
17.2.2	プログラムの修正 .....	79
17.2.3	コンパイルと実行 .....	79
17.2.4	問題.....	79
17.3	温度センサー端末を作成する。 .....	80
17.3.1	必要なプログラム・ライブラリをインストール .....	80
17.3.2	プログラムの変更 .....	80
17.3.3	実行.....	82
18	音の出力 .....	83
18.1	必要な知識.....	83
18.2	DigitalOut による音だし .....	84
18.2.1	接続.....	84
18.2.2	プログラム .....	84
18.2.3	応用 .....	84
18.3	PWM による音だし.....	85
18.3.1	接続.....	85
18.3.2	プログラム .....	85
18.3.3	疑似 Sin 波の出力 .....	86
18.4	DAC による音だし .....	87
18.4.1	接続.....	87
18.4.2	プログラム .....	87
18.5	応用 和音による音だし .....	88
18.6	応用.....	89
18.6.1	周波数を式で表す (Tedd OKANO さんより) .....	89

18.6.2	USB Audio.....	89
19	ステッピングモータの使用 .....	90
19.1	ステッピングモータの HelloWorld.....	90
19.1.1	プログラム .....	91
19.2	角度を指定して回す .....	92
19.2.1	プログラム .....	92
19.2.2	応用 .....	92
20	参考 .....	93

本書は mbed NXP LPC1678 と☆Board Orange を利用することを前提で書いています。  
mbed と書かれている場合は mbed NXP LPC1678 を指します。

資料は

mbed ホームページ <http://mbed.org/> 及び各ユーザ  
電子工作の実験室(後閑さんの HP) <http://www.picfun.com/>  
WikiPedia ホームページ <http://jp.wikipedia.org/>

他色々なホームページ

を引用・参考させて頂きました。

# 1 mbed を利用するための電子部品等の基礎知識

## 1.1 mbed 概要

mbed NXP LPC1768 は、ARM Cortex M3 を中心にデザインされた組み込みの開発セットである。プログラム開発の統合環境（コンパイルやファイル管理）などはクラウド(Web)で行い、チップへのプログラム書き込みは USB Flash メモリの感覚で扱える。従って、WEB が使える環境と USB フラッシュメモリが扱える環境があれば、OS や場所を問わずして使用できる。

ハードウェア特徴として以下の点があげられる。

- CPU : 32-bit ARM Cortex-M3 (96MHz)
- Program 領域 : 512KB FLASH
- Memory 領域 : 32KB RAM
- 外部入出力 : Ethernet, USB Host and Device, CAN, Serial, SPI, I2C, ADC, DAC, PWM, other I/O interfaces.
- 機能 : RTC

PC に繋いで使用する場合は USB から電源が供給され、ボード上の三端子レギュレータで 3.3V に変換して使用される。単独で使用する場合は 4.5V-9V のアダプタまたは電池を+(プラス)を VIN に-(マイナス)を GND に接続して使用する。

mbed は 3.3V で動作するので、デジタル出力の High は 3.3V になる。ポート毎に最大 40mA の電流出力が出来る(ポートの合計出力は最大で 400mA まで)。ボードすべてで 500mA の電力を超えない様に使用する。通常、ボードで使用される電力は 200mA なので、周辺機器で使用できる電力は 300mA になる。(Ethernet を使用しないときのボードで使用する電力は 100mA)。USB からの電源供給の場合は 460mA のヒューズが入っているが、他の場合は無いので注意して使用する事。三端子レギュレータは最大 0.8A の仕様である。

あわせて、mbed の取り扱いには、高密度基板(6層基板)のため反りとかの注意が必要です。

時計 (RealTimeClock) の時刻を保存するには 1.8V-3.3V の電池を VB に+ GND に-を接続します。

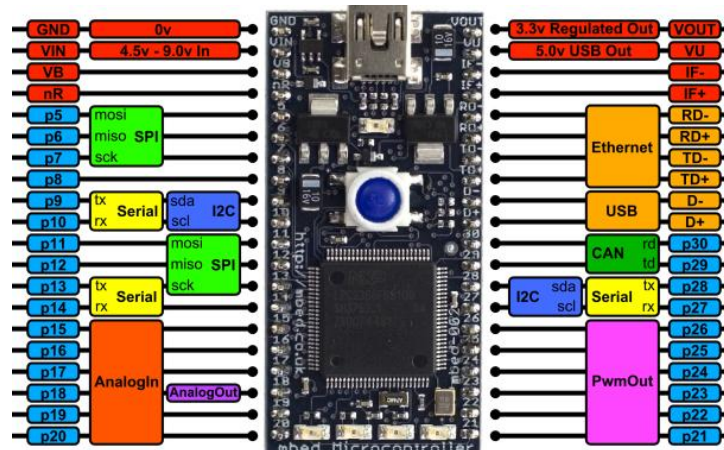


図 mbed のピン配置

## 1.2 簡単な電気回路(ハードウェア)

mbed は、各ピンより 40mA の出力や入力を超えない様に使用する。それを踏まえて周辺回路を構成する。マイコンには流れ込む電流(シンク電流)と吐き出す電流(ソース電流)の値が異なっている物もあり、よくあるマイコンの解説書にはシンク接続するものが多い。参考までに、mbed ボード上のチップ LPC1768 はシンク 50mA、ソース 45mA でとなっており、mbed の 40mA の制限はボード設計などの仕様で決定されていると思われる。

### 1.2.1 発光ダイオード(LED)

発光ダイオードは、半導体の発光素子で、これまでの電球などに比べ消費電力が少ないという特徴がある。しかしながら、マイコンの回路においては電力を多く消費されている素子の一種であり、消費電力を重視する場合は、様々な工夫がいる。

使用時には極性があるので注意する。足の長い方が陽極(アノード Anode)で短い方が陰極(カソード Cathode)である。一般的な砲弾型 LED の最大電流は 20mA、高輝度な物は 30mA 以上の物もある。電子回路を製作するときは、これらの電流値を超えない様に設計する。大よその電流  $i$  は(電源  $V$  - 順方向電圧  $V_f$ ) ÷ 電流制限抵抗  $R$  で計算できる。通常動作確認用の LED として点灯させる場合は、3mA も流せば十分であり、PWM など、明るさ制御が必要な場合は、定格電流程度まで流せる様に設計しておく。LED テスターがあれば、どの程度の光があるか判り便利である。抵抗の代わりに定電流ダイオード(CRD)を利用したり、LED ドライバを使用する事もある。

表、図から、mbed に直接青や白色の LED を接続して点灯させることは  $V_f$  が 3V 以上となるため困難である( $V_f$  の低いものを選択すれば可能)。

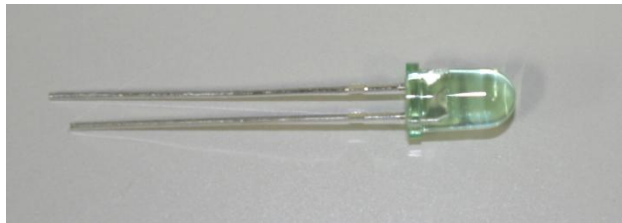


図 1.2 LED の例

表 LED の順方向電圧の参考値(※秋月電子 HP の商品から)

色	順方向電圧 $V_f$	備考
白	3V~4.2V	
青	3.2V~3.4V	
赤	1.8V~2.6V	3.3V なら 50-1k $\Omega$
黄	1.8V~2.4V	3.3V なら 50-1k $\Omega$
緑	1.8V~2.6V	3.3V なら 50-1k $\Omega$

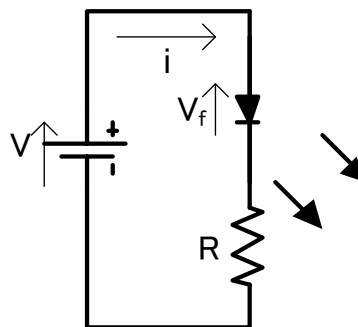


図 発光ダイオードの回路

$V$ :電源電圧、  $V_f$ : ダイオードの順方向電圧、  $R$ : 電流制限抵抗、 電流 :  $i$



## 1.2.2 ステッピングモータ

ステッピングモータは角度を正確に回す事が出来るモータです。これを利用して移動距離を指定するステージなどが作成できます。ただし、必要のトルクが足りないと脱調してしまいます。エンコーダ付きサーボモータなどを利用すると、正確に動作させることが出来る。

本書では、ステッピングモータとしては P-4241（秋月電子通商 1度/Pulse）、ドライバ IC としては I-2030（秋月電子通商）を使用する。



図 ステッピングモータ（秋月電子通商 P-4241）

ステッピングモータを動かすには何通りかパルスの与え方がある。例えば、下記表の Step1 → 2 → 3 → 4 → 1 → 2……と繰り返すことでも回すことが出来る。

表 ステッピングモータの動作

Step	1	2	3	4	1	2	3	4
A 端子	1	1	0	0	1	1	0	0
NotA 端子	0	0	1	1	0	0	1	1
B 端子	0	1	1	0	0	1	1	0
NotB 端子	1	0	0	1	1	0	0	1

(A Common および B Common は電源に Pullup する)

これらの駆動回路を FET を用いて作ることが出来るが、ドライバー IC を使うと手軽に出来る。

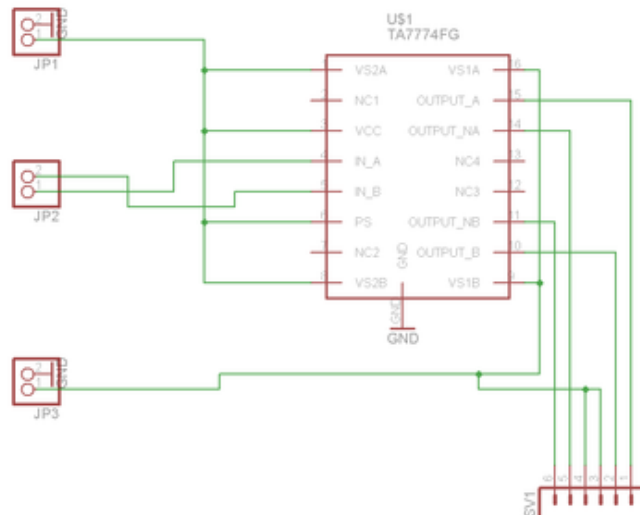
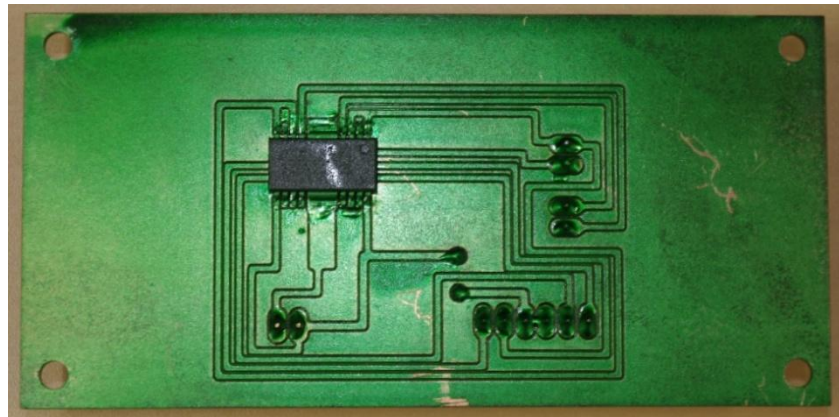


図 ドライバ IC 回路

- JP1 は操作用電源(mbed の VOUT (3.3V 端子)と GND)
- JP2 は操作用信号線(mbed の p19 と p20)
- JP3 はステッピングモータ用の電源(今回は 9V アダプタ)
- SV1 はステッピングモータを接続する。



(表)



(裏)

図 ステッピングモータドライバ回路(自作)

電圧を高くすることで速く回せますが、その分電力が必要になりますので、かなり熱くなる。制止時の省エネ設定とかドライバーの機能を使ったりする。

### 1.2.3 ラジコン (RC) サーボモータ

ラジコンで、エンジンの出力調整やハンドルの制御に昔から利用されていたが、最近ではロボットの関節などによく利用される。サーボモータとは別物である。これを利用すると簡単に角度を制御することが出来る。

RC サーボモータは、電源(モータ駆動と制御用)、信号線、GND の 3 本の線を接続する必要がある。制御信号はほとんど流れない (最大でも 1mA 程度) ので、直接 mbed に接続できる。ただし、電源には駆動トルクに応じて電流が流れるので、別途電源を用意した方が良い。無負荷時は 50mA 程度ですが、負荷時(ロボットなどに使用)の時は数 100mA の電流が必要です。

(参考 : [http://www2.plala.or.jp/k\\_y\\_yoshino/6legs/servo.html](http://www2.plala.or.jp/k_y_yoshino/6legs/servo.html))

なお、角度の制御は方形波のパルス幅で行う。一般的に方形波の周期  $T=10\text{-}20\text{msec}$  でパルス幅  $t$  をニュートラルの時間に設定すると中心、それを可変することで  $\pm 90$  度の範囲で動く。それ以上も動くが、内部のギヤなどが故障する可能性がある。

これは、特別なライブラリがない場合は、一般的に PWM の機能を使用して制御する。

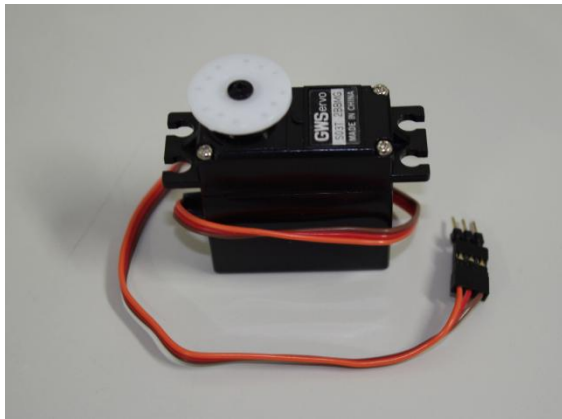


図 1.4 RC サーボモータ

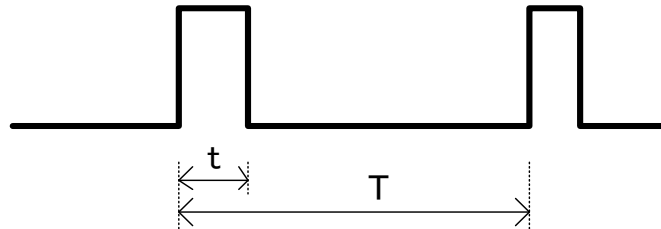


図 RC サーボモータの制御

表 サーボモータの制御信号

メーカー	ニュートラル [ $\mu$ sec]	可変範囲 [ $\mu$ sec]	制御線	電源	GND	備考
双葉電子工業 (株) (Futaba)	1520	$\pm 500$	白 (1ピン)	赤 (2ピン)	黒 (3ピン)	旧製品は $1310 \pm 500$ [ $\mu$ sec]
三和電子機器 (株) (SANWA)	1500	$\pm 600$	青 (1ピン)	赤 (3ピン)	黒 (2ピン)	
日本遠隔制御 (株) (JR)	1500	$\pm 500$	橙 (1ピン)	赤 (2ピン)	茶 (3ピン)	
近藤科学 (株) (KO PROPO)	1520	$\pm 500$	白 (1ピン)	赤 (2ピン)	黒 (3ピン)	

※機種によって異なる場合があるので、使用時は確認する事。

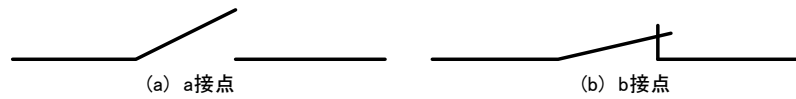
## 1.2.4 プッシュスイッチ

プッシュスイッチは押したときに導通になる a 接点(NO)と押したときに非導通になる b 接点(NC)の 2 種類がある。ここでは、一般的に使われる a 接点で考える。

Mbed のデジタル入力には、Normal モードと Pull Up モード、Pull Down モードの 3 種類があり、図 1.10 の様に接続する。Normal モードの抵抗値は  $10k\Omega$  程度、PullUp モードの抵抗はなくても良いが設定間違い時にショートしない様に、 $1k\Omega$  程度の抵抗を取り付ける。



図 タクトスイッチ



図スイッチ記号

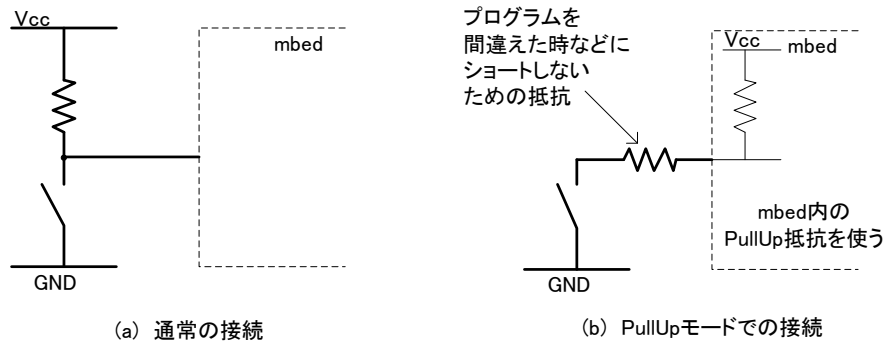


図 スイッチの接続

入力の OFF→ON と ON→OFF の閾値の電圧が一緒の場合は、閾値付近の電圧でハンチング等の悪影響を及ぼす。そのため、OFF→ON と ON→OFF の閾値の電圧が異なるシュミットトリガがよく使われる。mbed は、全 Pin とも閾値 1= $0.7V_{dd}$ (2.3V)、閾値 2= $0.3V_{dd}$ (1V)のシュミットトリガ入力。

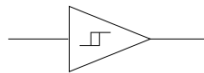


図 シュミットトリガ記号

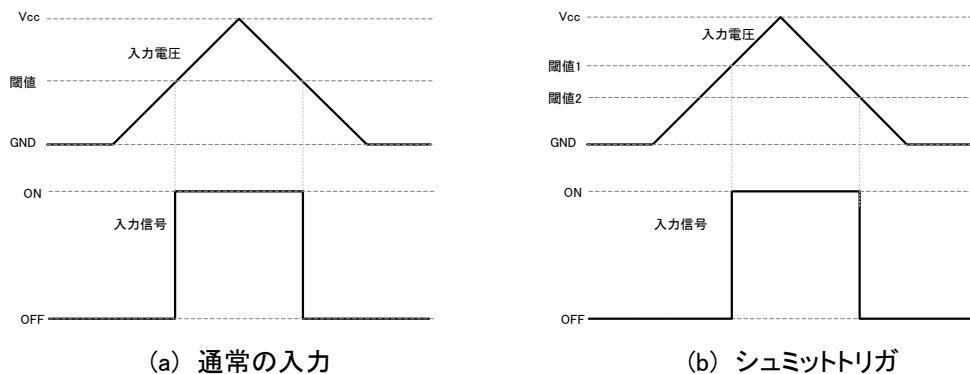


図 シュミットトリガと通常入力の違い

なお、カウンタ等で利用する場合は、シュミットトリガでもチャタリングの影響を受けるので、さらに積分回路を追加したり、ソフトウェアで対処したりする必要がある。

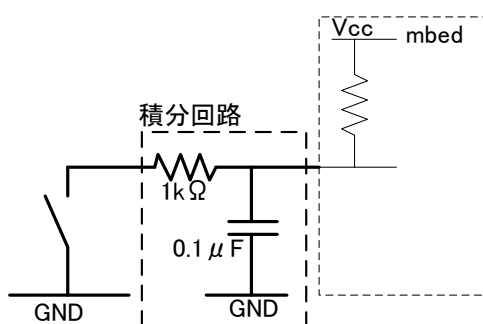


図 プルアップモード+積分回路の例（時定数 0.1mS）

### 1.2.5 ロータリエンコーダ

ロータリエンコーダは回転角度や速度を取得するために使われる。エンコーダには光学式とスイッチ式の二種類がある。光学式は高価であるがチャタリングの影響を受けなく、回路構成が楽である。一方スイッチ式は安価であるが、チャタリングの影響を受けるため、回路やプログラムに工夫が必要である。

Ach と Bch が異なる位相で信号が出てくるので、その時の情報から回転向き、カウント数から距離が判る。割り込みを使えると簡単に処理することが可能です。



図 スイッチ式ロータリエンコーダ

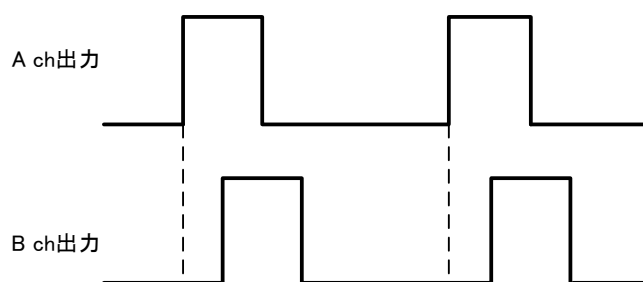


図 ロータリエンコーダの出力

### 1.2.6 可変抵抗器(ポテンシオメータ)

可変抵抗器でよく利用されるのはボリュームである。これには A タイプと B タイプがある。通常の音量調整で使用するものは A タイプである。これは、小さな音の変化が調整できる様にするため、非線形的な変化をする。B タイプのボリュームは直線的に変化するのでマイコンなどの操作で使用するのに都合が良い。

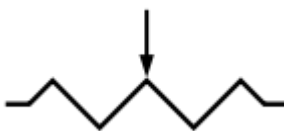


図 ポテンシオメータの図記号



図 ボリューム

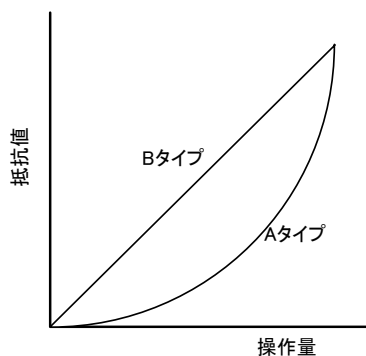


図 操作量と抵抗値の関係

### 1.2.7 スピーカ

スピーカは電磁石により振動板を動かす。後述の圧電ブザーよりも音は良いが、大きな電流が必要なので、通常使用するには、アンプが必要である。



図 スピーカ(写真 秋月電子通商)

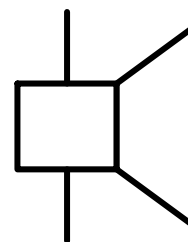


図 スピーカ図記号

### 1.2.8 圧電ブザー(圧電スピーカ)

圧電ブザーは、セラミックの圧電効果により振動子を動かし音を出す。ほとんど電流は流れない特徴がありマイコンと相性が良い。方形波を与えると大きな音が発生するが、サイン波では小さな音しか出ない。

可逆素子なので、振動を与えると高電圧を発生するので、マイコンとの間に保護抵抗(1kΩ程度)を接続する。



図 圧電ブザー

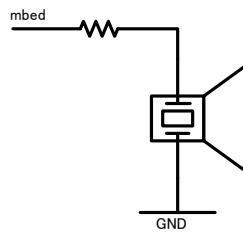


図 圧電ブザーの接続方法

### 1.2.9 照度センサー

照度センサーは明るさを感知する。フォトトランジスタや太陽電池が使われるが、人の目と同じような波長を合わせているものもある。



図 照度センサの例 (Panasonic NaPiCa)

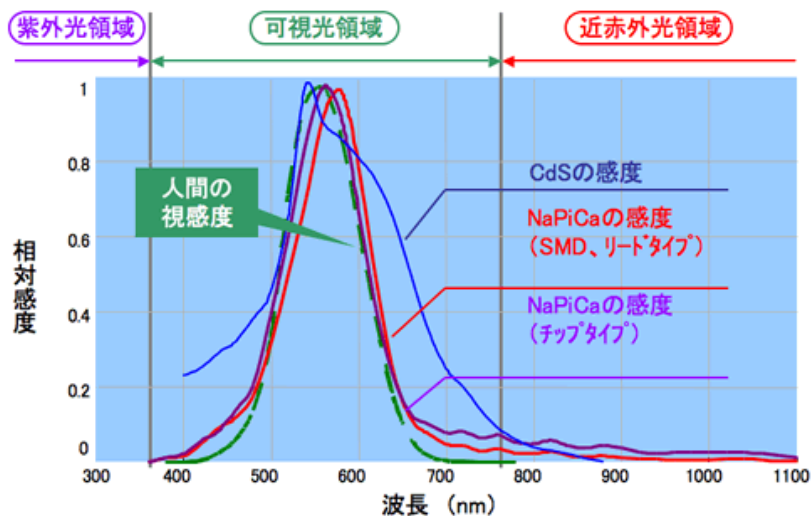


図 視感度図(Panasonic 電工 ホームページより引用)

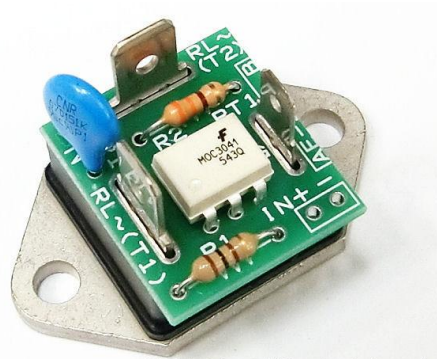
※NaPiCa は短い脚の方が+なので注意が必要





### 1.2.12 ソリッドステートリレー (SSR)

ソリッドステートリレーを利用すると、通常のコンセント電圧 (AC100V) をマイコン(5V)でコントロールできる。ゼロクロス機能を持ったものは0Vで電源を ON-OFF するので、雑音が低減するが反応が少し遅くなる。



※写真はキットを組み立てたものです。

図 35A ソリッドステートリレー  
(写真：秋月電子通商ホームページから引用)  
※35A で使用する場合は放熱板が必要



図 Panasonic 製 SSR  
[1A、2A の基板上で使用するもの]  
(写真：Panasonic ホームページから引用)

他にフォトMOSリレーなども制御することが出来る。もちろん通常の接点式リレーも使用できるが、その場合はコイル系負荷の接続を参考にし、FET スイッチやフリーホイールダイオードを使用の事

### 1.2.13 コイル系素子とマイコンの接続

モータやリレーなど電磁石を利用するものは、エネルギーを蓄積している、スイッチを切った途端、大きな逆電圧が発生する。対処した回路でない場合は素子を壊す。一般的には、フリーホイールダイオードを利用し、対処する。フリーホイールダイオードはショットキーダイオードなど高速動作が可能なダイオードの利用が必要。ON-OFF の周波数でダイオードを選択する。

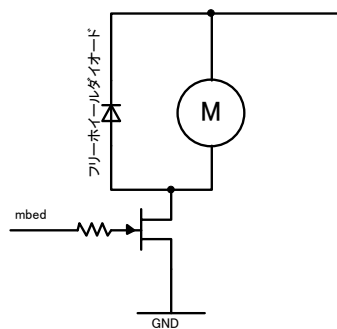


図 モータの接続の例

### 1.2.14 信号レベルの異なる他の回路の信号の接続

mbed は電源が 3.3V なので、同じ 3.3V で動作する IC などとの接続は直接可能である。安全の為に絶縁する場合や、基準の電圧が異なる場合はフォトカプラなどを使用する。mbed の外部出力の簡略図を図 1.6 に示す。

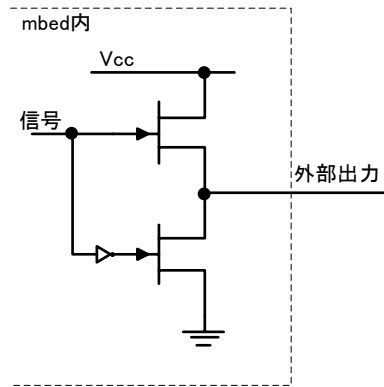


図 1.6 通常出力

TTL(5V)との出力の接続はオープンドレイン接続やバッファなどを利用する。また、簡易的に抵抗分圧で行う場合もある。

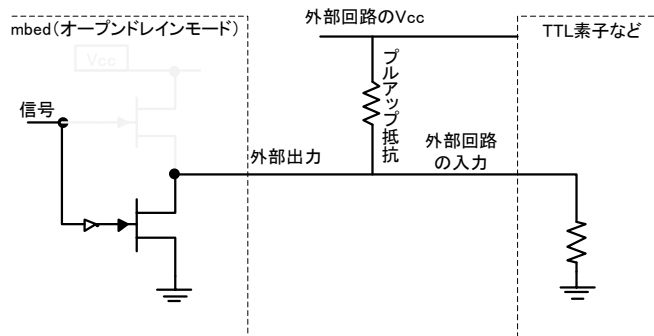


図 1.7 オープンドレインモード

無線モジュールの Digi 社の XBee は 3.3V なので、そのまま利用可能



図 XBee (Digi 社 写真は XBee-ZB)

### 1.3 mbed 周りの周辺機器

#### 1.3.1 USB

mbed には、2 種類の USB が備わっている。一つは、ボードに付いているミニ USB 端子で、もう一つは D+(Pin31)、D-(Pin32)である。

ミニ USB 端子は、プログラムのファイルを置いたり、シリアル通信に使ったりできる。

D+ (Pin31) ,D-(Pin32)は、LPC1768 の機能としてプログラムで使用する。キーボード・マウスをエミュレーション、USB フラッシュメモリを接続して書き込み、USB Audio、USB Midi などで使用できる。☆Board Orange の USB 端子に接続されている。

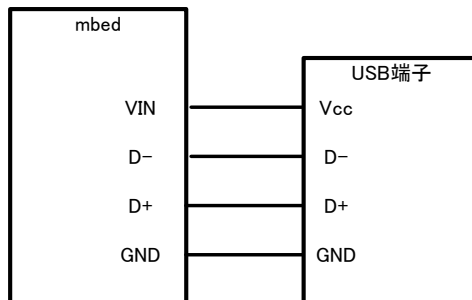


図 USB の接続図

#### 1.3.2 Ethernet

TD+, TD-, RD+, RD-は Ethernet の RJ45 端子に接続する。RJ45 端子はトランス式のものを採用する事が推奨されている。☆Board Orange の RJ45 端子に接続されている。

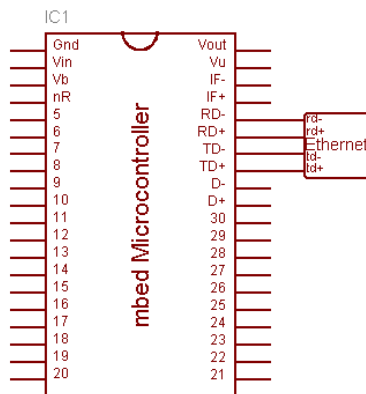


図 Ethernet 接続図

### 1.3.3 シリアルインターフェイス(RS-232C)

mbed には 3 つの Serial 出力端子がある。RS-232C で使用する場合は MAX232 などのドライバを利用し、レベル調整などをする。なおドライバ IC の時点で信号が反転される。

XBee と接続する場合は同じレベルの電圧であるのでシリアルの送信、受信線と GND の 3 本(もしくは電源 VOUT も含めた 4 本)を直接接続する。

表 mbed で使用可能なシリアル端子と注意事項

端子	備考
P9,10	☆Board Orange では I2C を使用するときには、PIN が重なるので使用できない。
P13,14	☆Board Orange では SPI を使用するときには、PIN が重なるので使用できない。
P27,28	☆Board Orange では LCD に接続されているので使用できない。

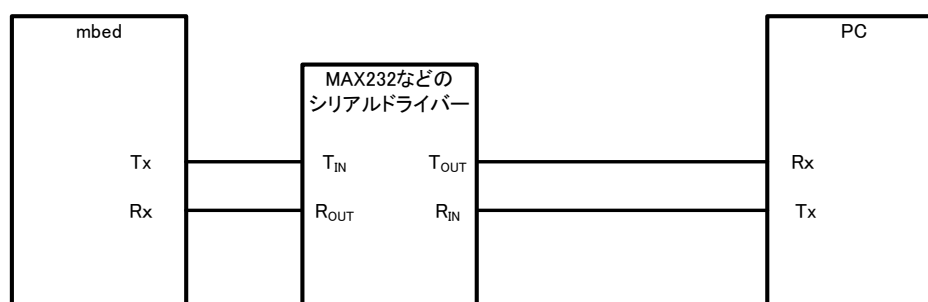


図 PC との接続図

※MAX232 などのシリアルドライバ IC はスイッチング昇圧したりします。バスコンが上手くついていないと動作に影響が出ます。

## 1.4 ☆Board Orange

☆Board Orange は mbed 用のベースボードで Ethernet、Text LCD、USB (A)、MicroSD が備わっており、簡単に使用できるようになっている。テスト用ボードとして最適で使いやすい。本書でのピン配置はこれに従う。詳しくは下記ホームページをご覧の事。USB を利用する場合は AC アダプタを利用する。

AC アダプタを利用する場合は 5V でセンター出力が+ (プラス)のものを使用する。

[http://mbed.org/users/logic\\_star/notebook/star\\_board\\_orange/](http://mbed.org/users/logic_star/notebook/star_board_orange/)

表 ☆Board Orange で使用されている Pin

ピン	備考	ピン	備考
GND	GND	VOUT	3.3V
VIN	アダプタ	VU	USB の電源
VB	バッテリー	IF-	利用不可
nR	リセット	IF+	利用不可
5	SD	RD-	LAN
6	SD	RD+	LAN
7	SD	TD-	LAN
8	SD	TD+	LAN
9	DIO, I2C sda	D-	USB
10	DIO, I2C scl	D+	USB
11	DIO, SPI mosi	30	LCD
12	DIO, SPI miso	29	LCD
13	DIO, Serial TX/SPI sck	28	LCD
14	DIO, Serial RX	27	LCD
15	DIO, AnalogIn	26	LCD
16	DIO, AnalogIn	25	LCD
17	DIO, AnalogIn	24	LCD
18	DIO, AnalogIn/Out	23	DIO, PWM
19	DIO, AnalogIn	22	DIO, PWM
20	DIO, AnalogIn	21	DIO, PWM



図 1.17 ☆Board Orange

## 1.5 m3pi Robot

m3piはPololuの3piをmbed用に拡張したものである。cookbookにLibraryも載っているのでmbedで簡単に使用する事が出来る。3piは動作が速くなるように回路に工夫がしてあるが、その部分は割愛する。

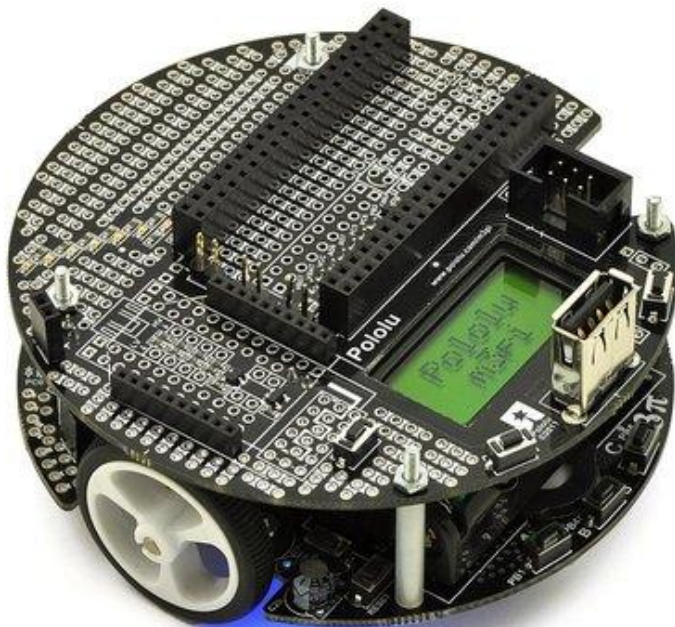


図 1.18 m3pi Robot(写真：Pololu 社ホームページより引用)

表のの配置を図 に回路図を図 に示す。LED は外側が LED1 で内側が LED8。

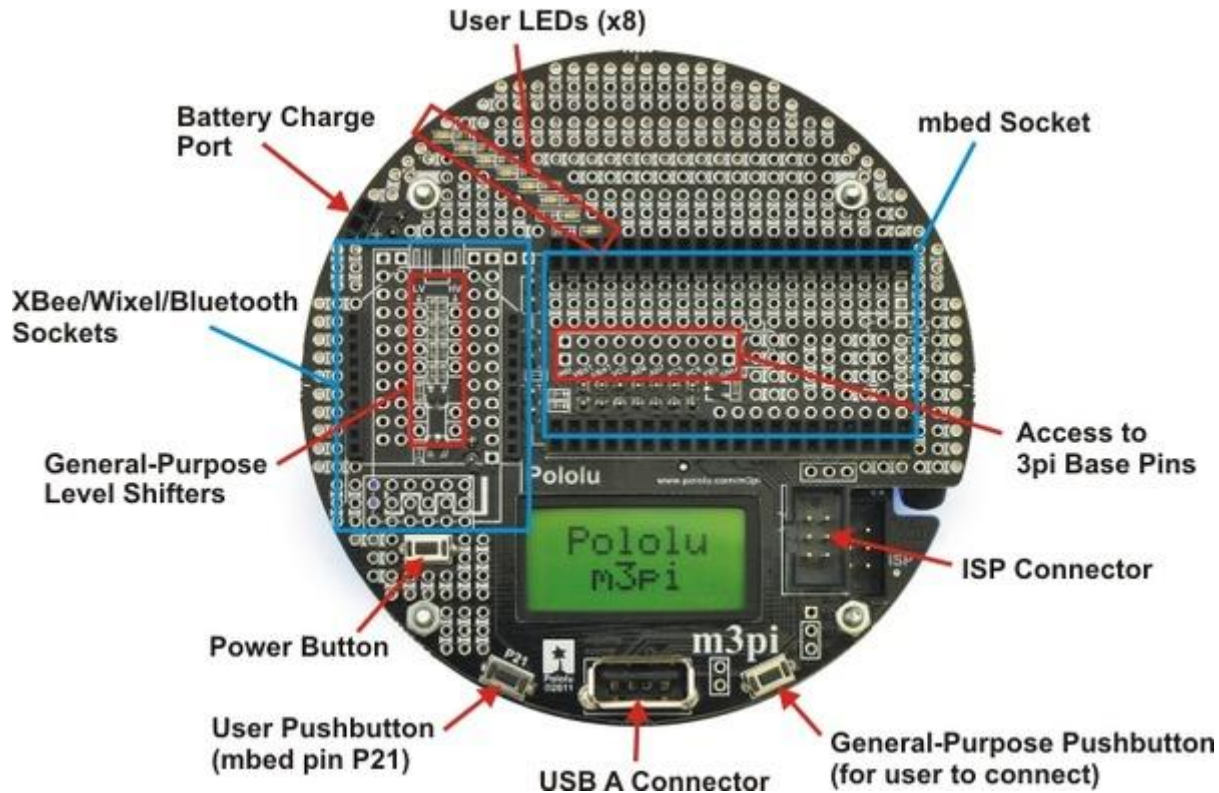


図 m3pi のボタン等の配置図

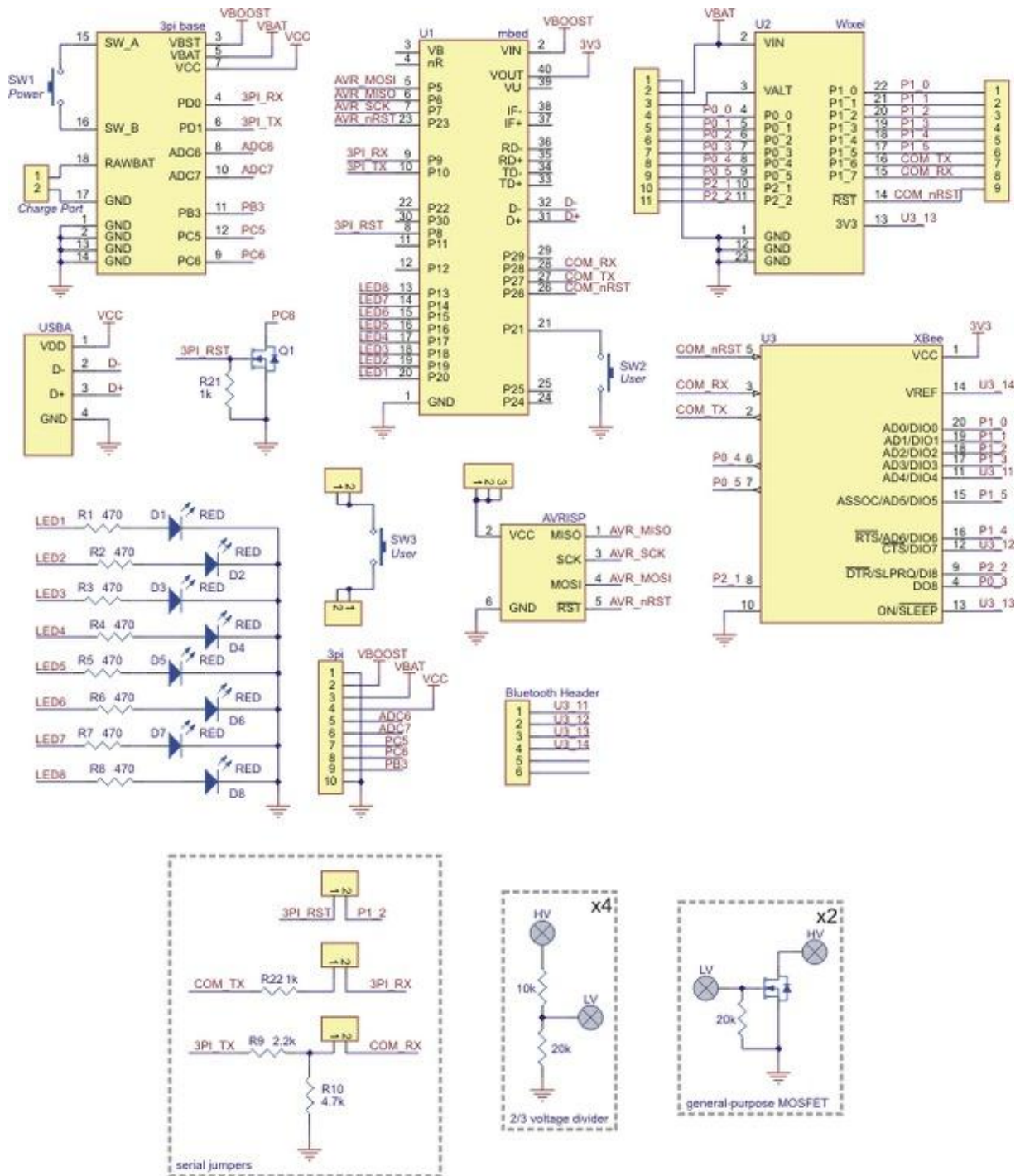


図 m3pi の回路図

mbed へのプログラムの書き込みは m3pi に差し込んだままの状態で行える。書き込みが終了したのち USB ケーブルを外して m3pi の Power Button を押す。

## 2 mbed を使うための C++ 言語の知識

### 2.1 コンパイラのドキュメントの場所

コンパイラのドキュメントは以下のところにある。

<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0348bj/index.html>

### 2.2 C Data Type

<http://mbed.org/handbook/C-Data-Types> より引用

mbed では以下のデータタイプが使用できる。

通常以下の用途に使用する

- int 型はカウンタ変数として (for loop counts, variables, events)
- char 型はキャラクターや文字列として
- float 型は、一般的な計測値として (seconds, distance, temperature)
- uint32\_t 型は 32 ビットレジスタアクセス用として
- The appropriate stdint.h types for storing and working with data explicitly at the bit level

整数型のデータタイプ

C type	stdint.h type	Bits	Sign	Range
<b>char</b>	uint8_t	8	Unsigned	0 .. 255
<b>signed char</b>	int8_t	8	Signed	-128 .. 127
<b>unsigned short</b>	uint16_t	16	Unsigned	0 .. 65,535
<b>short</b>	int16_t	16	Signed	-32,768 .. 32,767
<b>unsigned int</b>	uint32_t	32	Unsigned	0 .. 4,294,967,295
<b>int</b>	int32_t	32	Signed	-2,147,483,648 .. 2,147,483,647
<b>unsigned long</b>	long uint64_t	64	Unsigned	0 .. 18,446,744,073,709,551,615
<b>long long</b>	int64_t	64	Signed	-9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807

浮動小数点型のデータタイプ

C type	IEE754 Name	Bits	Range
<b>float</b>	Single Precision	32	-3.4E38 .. 3.4E38
<b>double</b>	Double Precision	64	-1.7E308 .. 1.7E308

#### Pointers

The ARMv7-M architecture used in mbed microcontrollers is a 32-bit architecture, so standard C pointers are 32-bits.

#### 注意

short, int, long long は符号型、char は符号なしが標準

Because the natural data-size for an ARM processor is 32-bits, it is much more preferable to use int as a variable than short; the processor may actually have to use more instructions to do a calculation on a short than an int!

In code ported from other platforms, especially 8-bit or 16-bit platforms, the data types may have had different sizes. For example, int may have been represented as 16-bits. If this size has been relied on, some of the code may need updating to make it more portable. In addition, it is quite common that programmers will have defined their own types (UINT8, s8, BYTE, WORD, ..); it is probably better to convert this to the stdint.h types, which will be naturally portable across platforms.



### 3 Hello World !

“Hello World !”とは、プログラムを取り組むための最初のプログラムを一般的に指す。これは、プログラムに特に意味があるわけではないが、一連の動作を確認するために使う。組み込み系では、LEDの点滅のプログラムを実行することが多い。

#### 3.1 ログイン

(1) <http://mbed.org/> に接続して Login or signup をクリックする。

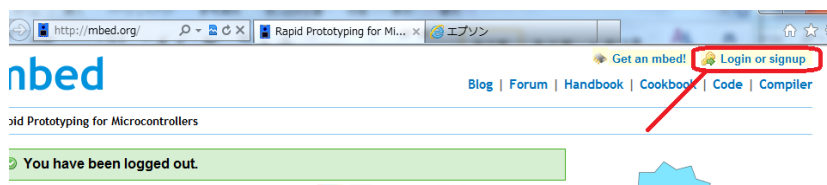


図 2.1 mbed ホームページ

(2) ユーザ名とパスワードを入力後、Login ボタンを押す。



図 2.2 ログイン画面

#### 3.2 コンパイルと実行形式のファイルの保存

(1) コンパイラ画面へ移るために Compiler をクリックする。



図 2.3 コンパイラ画面へ移る

(2) 新しいプログラムを作成するので、New をクリックする。

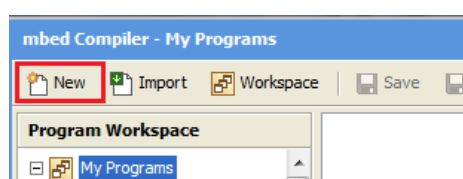


図 2.4 コンパイラ画面

(3) プログラム名を入力して OK ボタンを押す。

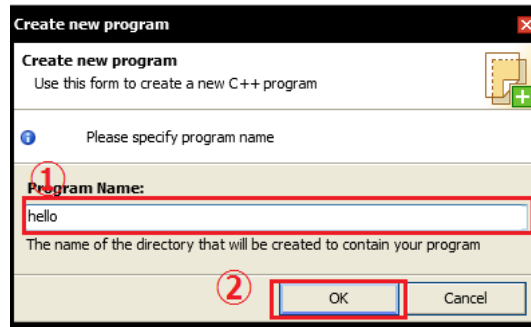


図 2.5 プログラム名入力画面

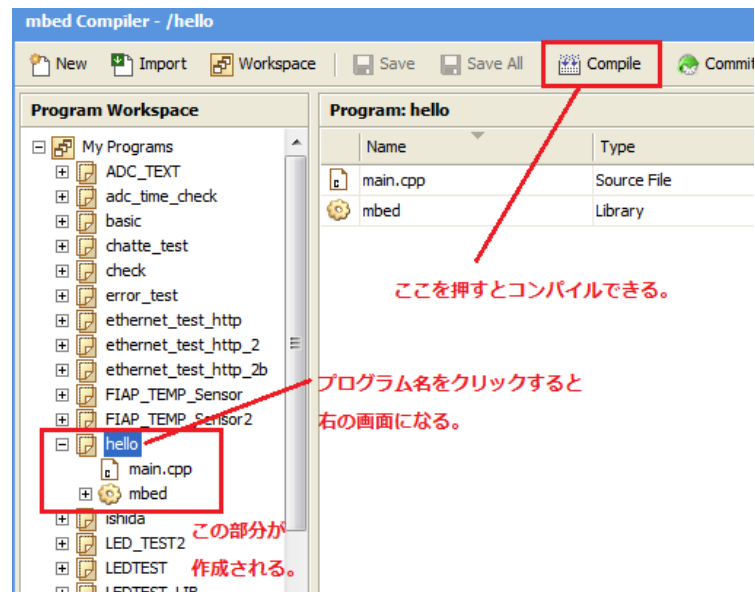


図 2.6 コンパイル画面 1

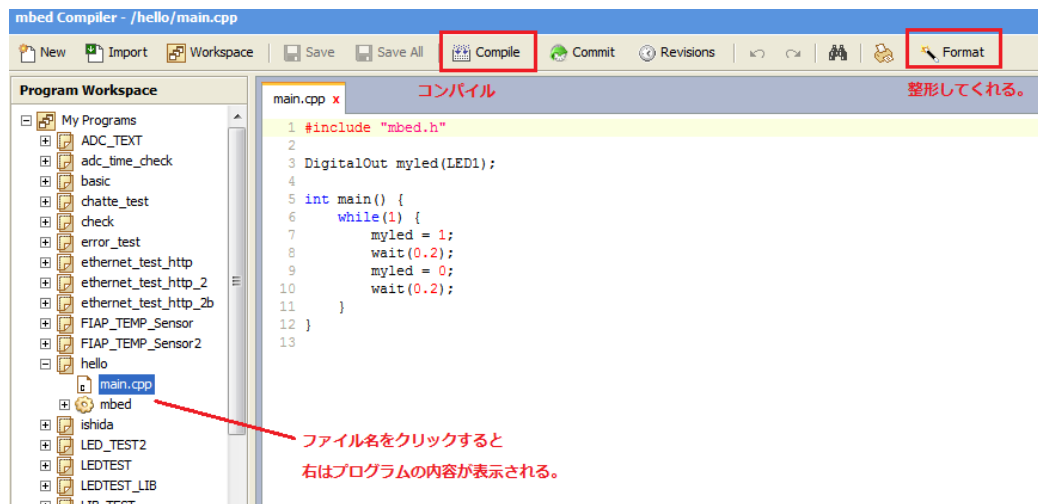


図 2.7 コンパイル画面 2

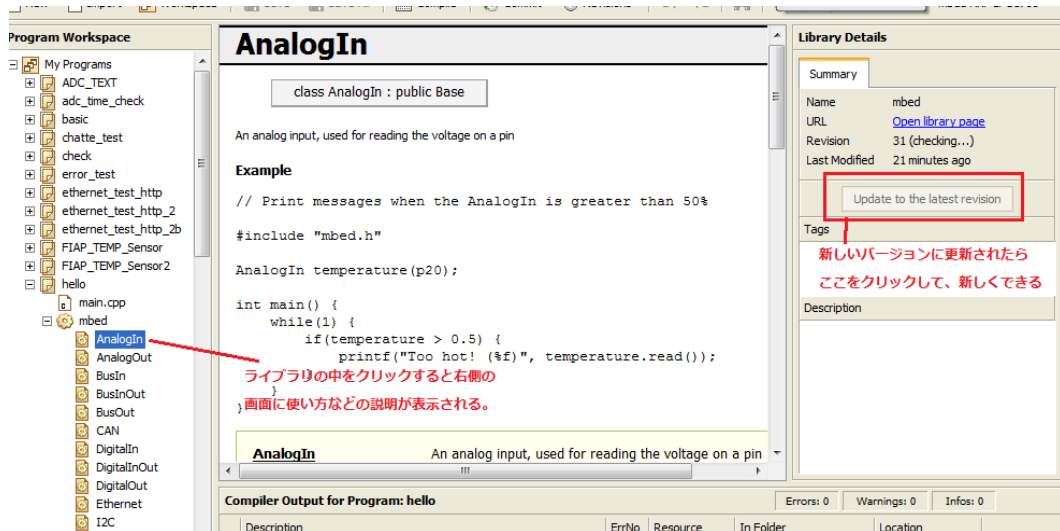


図 2.8 コンパイル画面 3

(4) mbed を USB で PC と接続し、コンパイルボタンを押し、ファイルを保存する。

### 3.3 実行

(1) mbed 上のリセットボタンを押す。

押すと、FLASH メモリ上の最新のプログラムが CPU にロードされる仕組みになっている。

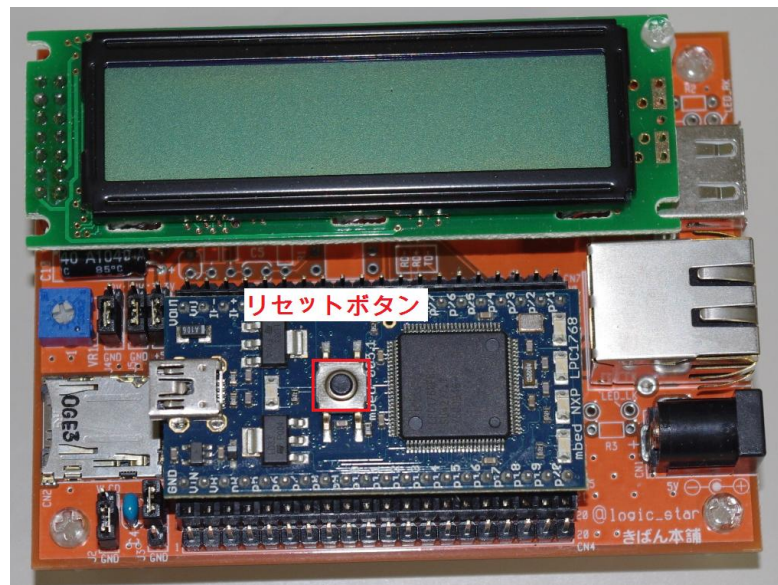


図 2.9 mbed のリセットボタン

### 3.4 問題

プログラムを理解しましょう。

wait 関数は何をしますか？

Program Workspace の mbed ライブラリを展開して wait\_api をクリック

## 4 ライブラリの作り方

ライブラリは、プログラムなどで再利用する時に便利な形である。今回は Hello World のプログラムを改良してライブラリを作成する。

LED の ON、OFF、SWAP の 3 つのメソッド（命令）を作る。

main.cpp（これが元）

```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}
```

### 4.1 元のプログラムをサブルーチンに変更

main.cpp（赤字が変更したところ）

```
#include "mbed.h"

DigitalOut myled(LED1);

void ON() {
    myled=1;
}
void OFF() {
    myled=0;
}
void SWAP() {
    myled=!myled;
}

int main() {
    while (1) {
        ON();
        wait(0.2);
        OFF();
        wait(0.2);
    }
}
```

## 4.2 クラスに変更する

main.cpp 赤が変更したところ

```
#include "mbed.h"

class LED {
public:
    LED(PinName pin):_pin(pin) {
        _pin=0;
    }
    void ON() {
        _pin=1;
    }
    void OFF() {
        _pin=0;
    }
    void SWAP() {
        _pin=!_pin;
    }
private:
    DigitalOut _pin;
};

LED led(LED1);

int main() {
    while (1) {
        led.ON();
        wait(0.2);
        led.OFF();
        wait(0.2);
    }
}
```

~~DigitalOut myled(LED1);~~は要らなくなる。  
かわりにクラスの初期化”LED led(LED1);”が入る。

```
LED(PinName pin):_pin(pin) {
    _pin=0;
}
```

は  
コントラクタ指定で LED が初期化した時に pin を \_pin に移す。  
または初期化時に \_pin=0 をして、LED を消灯する



### 4.3 ファイルの分割

ライブラリにするためにファイルを3つに分割する。

ヘッダーファイル(led.h) ここにクラスの定義を書く

```
#ifndef MBED_LED_H
#define MBED_LED_H

#include "mbed.h"

class LED {
public:
    LED(PinName pin);
    void ON();
    void OFF();
    void SWAP();
private:
    DigitalOut _pin;
};
#endif
```

大きなプログラムになった時に二重定義にならない様に#ifndef, #define, #endif を加える。

ライブラリ本体(led.cpp) ここにクラスの内容を書く

```
#include "led.h"
#include "mbed.h"

LED::LED(PinName pin):_pin(pin) {
    _pin=0;
}

void LED::ON() {
    _pin=1;
}

void LED::OFF() {
    _pin=0;
}

void LED::SWAP() {
    _pin=!_pin;
}
```

メインプログラム(main.cpp)

```
#include "mbed.h"
#include "led.h"

LED led(LED1);

int main() {
    while (1) {
        led.ON();
        wait(0.2);
        led.OFF();
        wait(0.2);
    }
}
```

※きちんとライブラリが出来ているとメインプログラムが判りやすくなる。

## 4.4 ライブラリの登録

他のプログラムでも利用できるように、ライブラリを登録しておきます。なお、色々な人に使ってもらえるように、公開することも可能です。

### (1) ライブラリの作成 (変更)

プログラム名でマウスを右クリックします。メニューで **New Library...** を選択する。  
ライブラリ名を **led** で登録する。

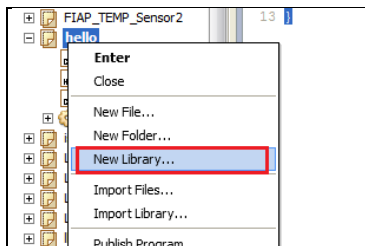


図 ライブラリの作成

### (2) ファイルの移動

led ライブラリの中に **led.cpp** と **led.h** を移動する。(ドラッグ&ドロップで出来る)



図 フォルダ画面

### (3) 登録

led ライブラリの上に右クリックして **Publish Library** を選択

万人に公開しない場合は **Add to my public mbed profile** のチェックを外して **OK** ボタンを押す。

**Description** には何に使うライブラリ化など、**Revision** は変更点などを記入。これは、後でも編集できる。

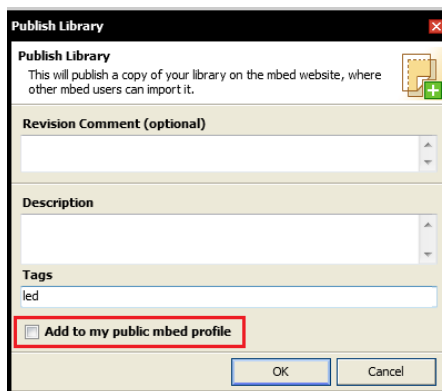


図 ライブラリの Publish 画面

### (4) 完了

led のライブラリがまとまる。

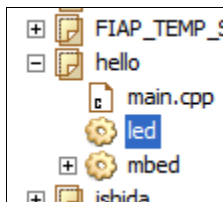


図 フォルダ画面(led ライブラリがライブラリに)

## 4.5 ライブラリが判るようにドキュメントを記す

今は下図のような状態で何もドキュメントが有りません。これでは、そのあと何をしようとしていたのか判りません。mbed のクラウドではヘッダーファイルにコメントを書くことで、自動的にドキュメントを生成します。





図 ライブラリー一覧の画面

#### 4.6 コード(\*.h)に説明を書く

前もってライブラリを編集モードにする。(ライブラリの上で Edit Library をクリック)  
 下記のようにコメントを書き加える。(赤字の部分)

```

/* LED Library
 * V1.0
 */
#ifndef MBED_LED_H
#define MBED_LED_H

#include "mbed.h"
/** LED Class
 *
 * Example:
 * @code
 * #include "mbed.h"
 * #include "led.h"
 *
 * LED led(LED1);
 *
 * int main() {
 *     while (1) {
 *         led.ON();
 *         wait(0.2);
 *         led.OFF();
 *         wait(0.2);
 *     }
 * }
 * @endcode
 */
class LED {
public:
    /** Create led object
     *
     * @param pin LED Pin
     */
    LED(PinName pin);
    /** LED ON
     * @return void
     */
    void ON();
    /** LED OFF */
    void OFF();
    /** LED SWAP */
    void SWAP();
private:
    DigitalOut _pin;
};
#endif

```

前回と同様 Publish する。

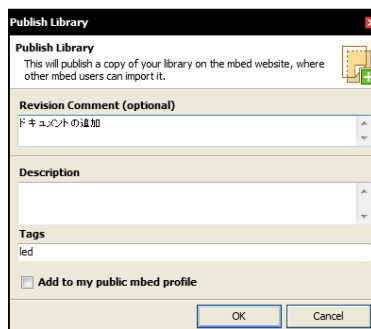


図 Publish 画面

#### 4.7 ドキュメントの確認

同様にライブラリの画面を確認すると API という欄が出来る。

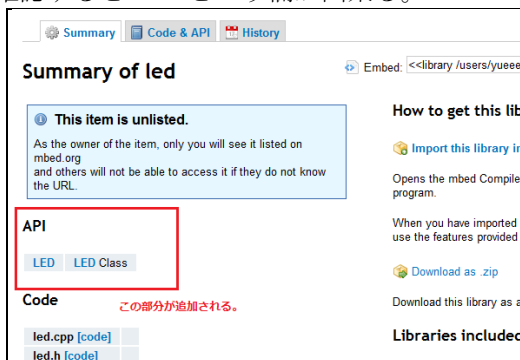


図 ドキュメントが加わったライブラリ

クリックすると自動でドキュメントができてい事が確認できる。

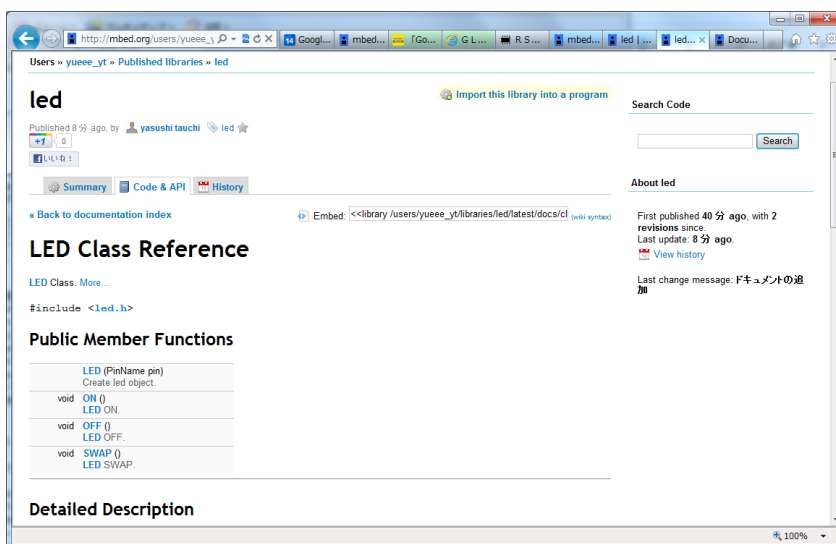


図 API の画面

## 4.8 ライブラリを使う

### (1) 新たなプログラムの作成

New をクリックしてプログラムを作成する。プログラム名は何でも結構です。

### (2) ライブラリの引用

Import ボタンを押してライブラリをインポートします。TargetPath がプログラム名になっているのを確認する。

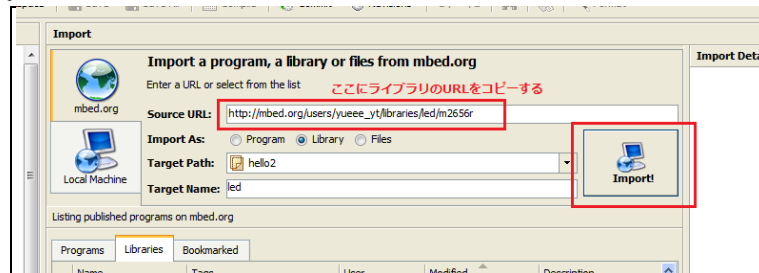


図 インポート画面

### (3) プログラムの作成

ライブラリを使用したプログラムを作成する。

```
main.cpp x led.cpp x
1 #include "mbed.h"
2 #include "led.h"
3
4 LED led(LED2);
5
6 int main() {
7     while(1) {
8         led.SWAP();
9         wait(0.2);
10    }
11 }
12
```

図 ライブラリを使用した簡単なプログラム

### (4) 確認

コンパイル&実行で確認する。

## 4.9 問題

p 10 に LED を接続して点灯するプログラムを作成する。

---

---

---

---

---

---

---

---

## 5 液晶キャラクタディスプレイ (LCD)

ここでは、☆Board Orange に備わっている LCD を利用する。

### 5.1 TextLCD の命令(メソッド)

mbed クラウド内の Cookbook で公開してされている TextLCD Class のメソッドは printf, cls, locate, putc の 4 つしかなく単純である。

表 TextLCD の命令

<b>cls()</b>	画面のクリア
<b>locate (int column, int row)</b>	0,0 が左上
<b>int putc (int c)</b>	1 文字出力
<b>printf (const char *format,...)</b>	printf スタイルの出力

### 5.2 Hello の表示(サンプルプログラムの実行)

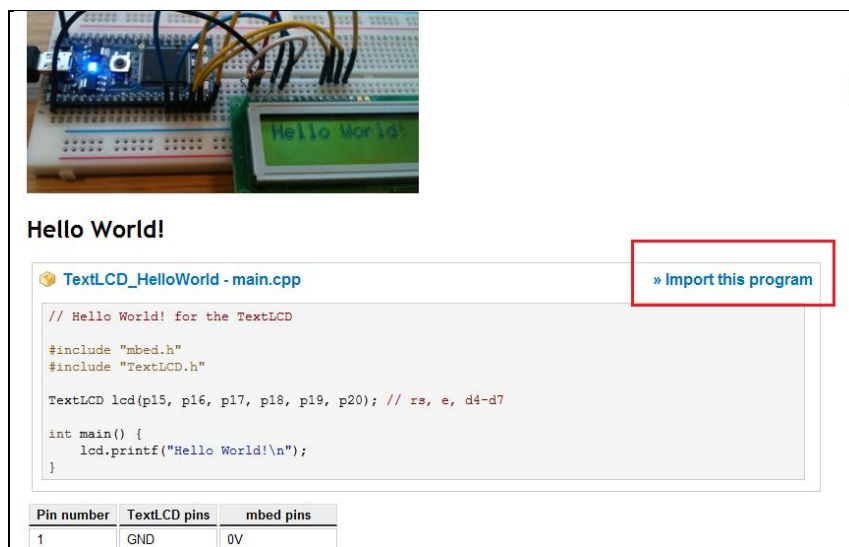
#### 5.2.1 サンプルプログラムの読み込み

(1)TextLCD のページに移動

mbed ホームページの Cookbook の LCDs and Displays の Text LCD をクリック

(2)サンプルプログラムの読み込み

ホームページ内の Import this program をクリック



Pin number	TextLCD pins	mbed pins
1	GND	0V

図 Cookbook の TextLCD のページ

OK をクリック

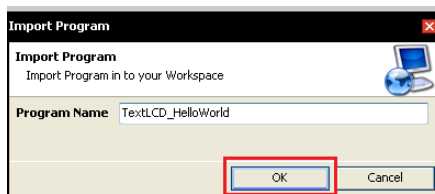


図 インポートのダイアログ画面

取り込めたら TextLCD\_HelloWorld のプログラムフォルダが出来、下図の様な画面が表示

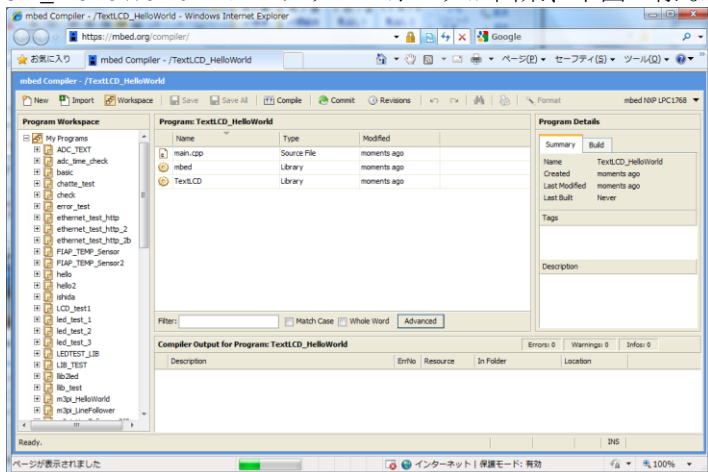


図 インポート完了画面

### 5.2.2 プログラムの修正

サンプルプログラムと☆Board Orange では LCD の接続している mbed の Pin の場所が異なるので ☆Board Orange の Pin に合わせてプログラムを変更したのち、コンパイルと実行。

main.cpp の

TextLCD lcd(p15, p16, p17, p18, p19, p20); // rs, e, d4-d7 を  
TextLCD lcd(p24, p26, p27, p28, p29, p30); // rs, e, d4-d7 に変更

```
// Hello World! for the TextLCD

#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30); // rs, e, d4-d7

int main() {
    lcd.printf("Hello World!\n");
}
```

### 5.2.3 実行結果



図 LCD の表示

### 5.3 数値を出力

TextLCD\_HELLO のプログラムを変更して実行する。

```
// Hello World! for the TextLCD

#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30); // rs, e, d4-d7

int main() {
    int i;
    lcd.cls();
    lcd.printf("Hello World!¥n");
    for(i=10;i>0;i--){
        lcd.locate(0,1);
        lcd.printf("Countdown %2d ", i);
        wait(0.1);
    }
}
```

lcd.printf 内の書式指定は、数値の桁数が判る場合は指定する。判らない場合は、桁が減った時でも対処できる様に空白などを、加えておく。

## 6 タイマーや割り込み

マイコンを使う上で欠かせないのがタイマーや割り込みである。

### 6.1 Timer

Timer は時間の計測に使用する。ただし 32bit の  $\mu$  秒カウンタなので  $2^{31}-1$  (30 分程度) しか計測できない。それ以上は RTC(リアルタイムクロック)を使用の事

### 6.2 wait

プログラム中に待ち時間を入れる時に使用する。NOP タイプなので資源は利用しません。

### 6.3 Ticker

通常のタイマー割り込みです。定期的な時間毎に処理をする時に使用する。

### 6.4 Timeout

一定時間経ったら割り込みをする。

### 6.5 RTC

RTC (リアルタイムクロック) は簡単に言えば時計です。設定は 1970 年 1 月 1 日からの秒をセットして使用する。しかし、秒の計算をするのは大変なので関数が用意されている。

#### 6.5.1 プログラム

下記のプログラム例は 2011/5/5 10:00:00 をセットする。月は 0 から 11 で与える。1 引く必要があることに注意。なお、ボタン電池で時刻を保存することも可能。

```
#include "mbed.h"

int main() {
    // setup time structure for Thu, 5 May 2011 10:00:00
    struct tm t;
    t.tm_sec = 00;    // 0-59
    t.tm_min = 00;    // 0-59
    t.tm_hour = 10;   // 0-23
    t.tm_mday = 5;    // 1-31
    t.tm_mon = 4;     // 0-11
    t.tm_year = 111;  // year since 1900

    time_t seconds = mktime(&t);
    printf("Time as seconds since January 1, 1970 = %d\n", seconds);

    set_time(seconds);
    //Read RTC Data
    for(int i=0; i<10; i++){
        seconds = time(NULL);
        printf("Time as a string = %s", ctime(&seconds));
        wait(1);
    }
}
```

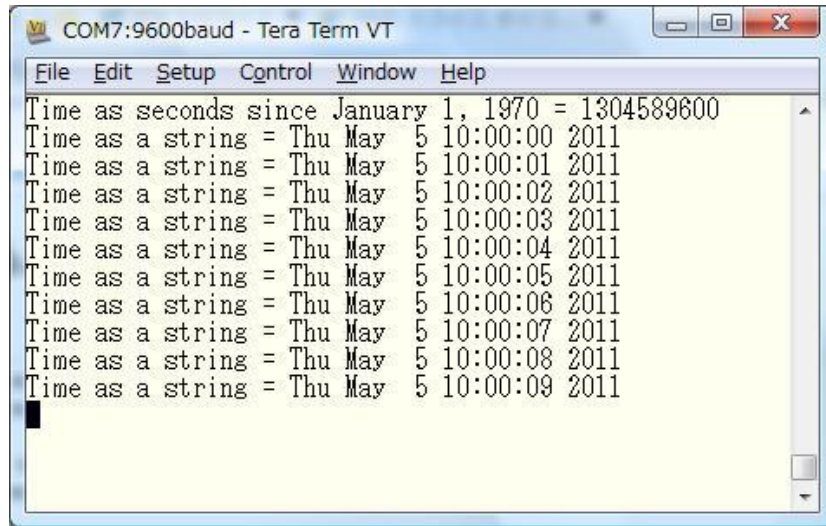
time 構造体を作成し、mktime で 1970 年からの秒に変換する。

set\_time で RTC に入力し、rtc の設定完了。

あとはデータの呼び出し

wait(1)は 1 秒待ち

## 6.5.2 実行結果



```
COM7:9600baud - Tera Term VT
File Edit Setup Control Window Help
Time as seconds since January 1, 1970 = 1304589600
Time as a string = Thu May 5 10:00:00 2011
Time as a string = Thu May 5 10:00:01 2011
Time as a string = Thu May 5 10:00:02 2011
Time as a string = Thu May 5 10:00:03 2011
Time as a string = Thu May 5 10:00:04 2011
Time as a string = Thu May 5 10:00:05 2011
Time as a string = Thu May 5 10:00:06 2011
Time as a string = Thu May 5 10:00:07 2011
Time as a string = Thu May 5 10:00:08 2011
Time as a string = Thu May 5 10:00:09 2011
```

図 RTC のデータ読出し画面

より、正確な時計を利用した場合は、外付けの RTC を別途使う。



## 7 mbed についている USB を利用

### 7.1 USB を使用したシリアル通信

#### 7.1.1 Windows のドライバーインストール

まず、PC (Windows) にドライバーを入れる必要がある。ドライバーと合わせて通信ソフト(例えば TeraTerm)なども確認のため必要。

ドライバーの Install は [handbook](#) の [Windows serial configuration](#) に記されている。

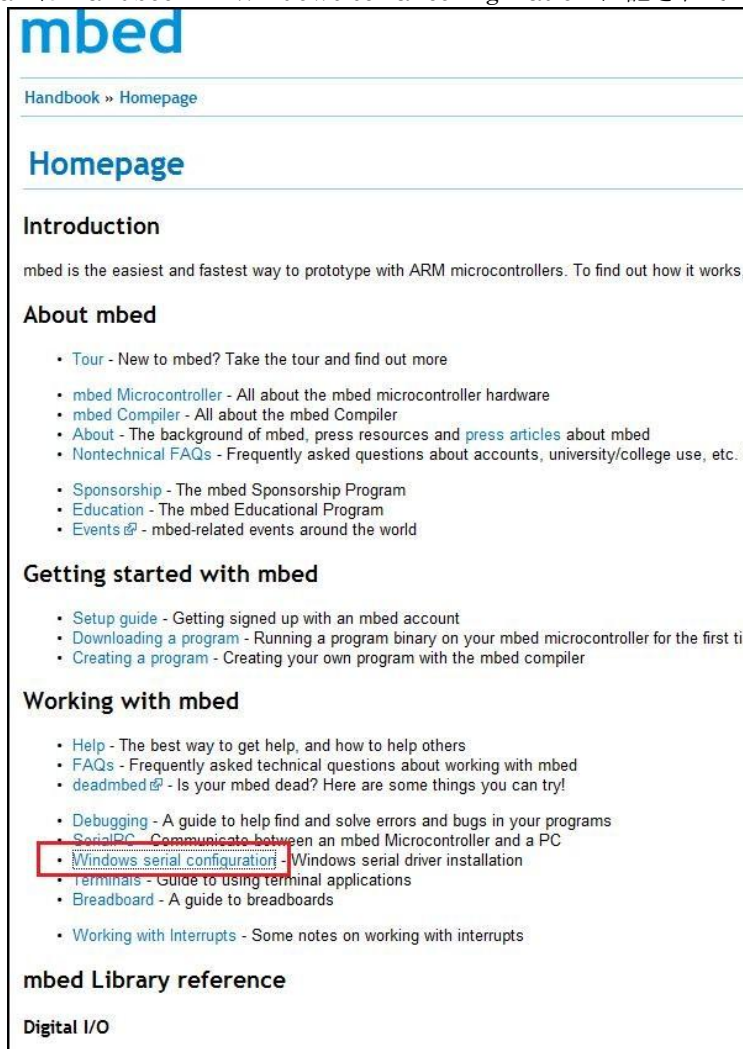


図 handbook 画面



図 ダウンロードする画面

もし、再度必要になった場合は、¥program files¥mbed の中にドライバーが保存されているのでそれも利用出来る。

### 7.1.2 プログラムの作成

```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    int i;
    for (i=0;i<11;i++)printf("Test Count: %d ¥n", i);
}
```

### 7.1.3 TeraTerm の立ち上げと設定

TeraTerm をシリアルで接続する。

TeraTerm の Control>Terminal で Recive を LF に設定する。

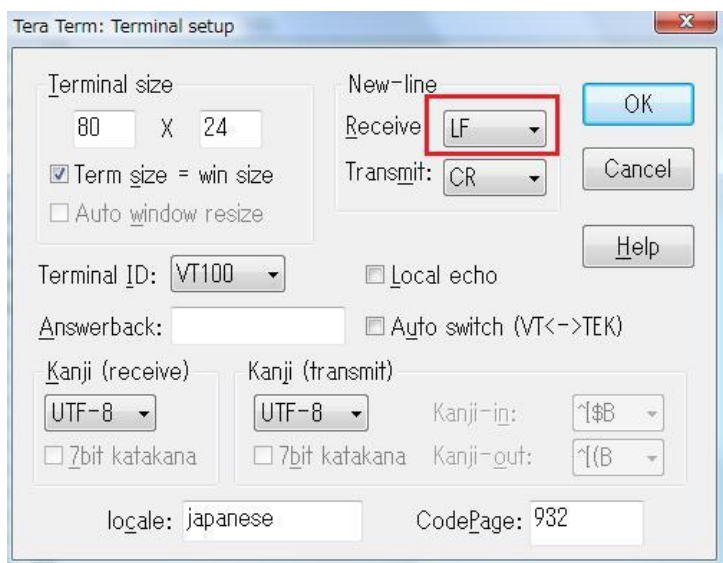


図 TeraTerm 設定画面

### 7.1.4 実行

リセットボタンを押すと mbed が実行し、TeraTerm にメッセージが表示される。

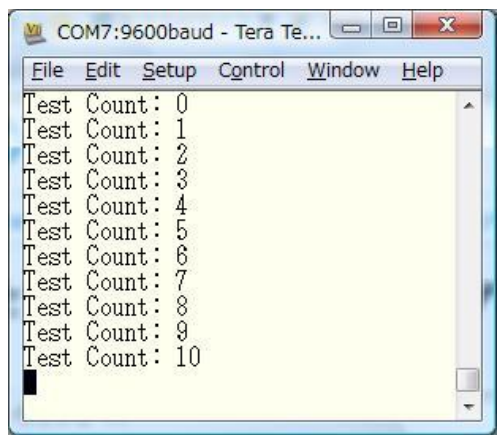


図 実行画面

## 7.2 mbed 内蔵の USB Flash メモリに保存(local file system)

mbed を PC に接続すると USB フラッシュメモリみたいに見えますが、プログラム側(mbed 側)からも利用可能である。(handbook の local file system を参照。)

mbed 側から書き込んでいる時は、PC 側が切り離されるので、短時間で書き換えを行うものは USB フラッシュメモリや SD カードを利用した方が良い。

### 内蔵の USB フラッシュメモリの利用条件

- 8.3 ファイルネームのみ
- サブディレクトリは使用不可
- fseek の利用不可(“w”)フラグ時

### 7.2.1 プログラム

out.txt というファイルを作成して、その中にテキストを書く例

```
#include "mbed.h"

LocalFileSystem local("local");

int main() {
    FILE *fp = fopen("/local/out.txt", "w");
    for(int i=0; i<11; i++)
        fprintf(fp, "Output Count:%d ¥r¥n", i);
    fclose(fp);
}
```

## 8 SD カードの利用方法

### 8.1 ファイルの作成

SD カードを利用するには、Cookbook の SD Card File System を参考にする。サンプルと☆ BoardOrange は同じピンを使用しているので、SDFileSystem の定義はそのまま利用できる。サンプルは、書き込みの終了が判らないので、TextLCD のライブラリも加えて試す。

### Storage, Smart Cards, and Magnetic Cards

- [SD Card File System](#)
- [USBMSDHost - USB MSD \(FLASH Disk\) Host](#)
- [Wav\\_SD\\_Card\\_Read-for-RS-EDP](#) - Reading the header of a .wav file stored on an SD card.
- [SD\\_Card\\_Write-for-RS-EDP](#) - Writing a character string to a file on an SD card.
- [Smart Card](#)
- [Magnetic Card Reader](#) - How to connect and use the Apollo Magnetic Card Reader from Sparkfun

### Digital Signal Processing

☒ クックブックのページ

```
#include "mbed.h"
#include "TextLCD.h"
#include "SDFileSystem.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30);
SDFileSystem sd(p5, p6, p7, p8, "sd");

int main(void) {
    FILE *fp = fopen("/sd/test.txt", "w");
    if (NULL != fp) {
        fprintf(fp, "StarBoard Orange. mbed NXP LPC1768.");
        fclose(fp);
    } else {
        error("Open failed.¥n");
        return 1;
    }
    lcd.cls();
    lcd.locate(0,0);
    lcd.printf("Finish");
    return 0;
}
```

## 8.2 ディレクトリの作成

内蔵 USB の localfilesystem とは異なり、サブディレクトリも作成できるか確認する。

```
#include "mbed.h"
#include "TextLCD.h"
#include "SDFileSystem.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30);
SDFileSystem sd(p5, p6, p7, p8, "sd");

int main(void) {
    DIR *d;
    struct dirent *p;
    lcd.cls();
    d = opendir("/sd");
    if ( d != NULL )
    {
        while ( (p = readdir(d)) != NULL )
        {
            lcd.printf("%s    %n", p->d_name);
            wait(0.5);
        }
    }
    lcd.locate(0,1);
    lcd.printf("** Finish ** ");
    return 0;
}
```

## 8.3 ファイル一覧の作成

SD カード内のファイル名の一覧を LCD に書き出す。もちろん日本語のファイル名は無理です。

```
#include "mbed.h"
#include "TextLCD.h"
#include "SDFileSystem.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30);
SDFileSystem sd(p5, p6, p7, p8, "sd");

int main(void) {
    DIR *d;
    struct dirent *p;
    lcd.cls();
    d = opendir("/sd");
    if ( d != NULL )
    {
        while ( (p = readdir(d)) != NULL )
        {
            lcd.printf("%s    %n", p->d_name);
            wait(0.5);
        }
    }
    lcd.locate(0,1);
    lcd.printf("** Finish ** ");
    return 0;
}
```

SD カードでロガーなどが作れることがわかったと思う。

## 9 デジタル出力 (LED)

ここでは順次点滅する LED のプログラムについて考える。

### 9.1 変数を使う方法

変数を 1 ずつ増やしていき、Switch 文で場合分けすると簡単で、見易くなる。

led\_test\_1.cpp

```
#include "mbed.h"

DigitalOut led1(LED1);
DigitalOut led2(LED2);
DigitalOut led3(LED3);
DigitalOut led4(LED4);

int main() {
    unsigned int i=0;
    while (1) {
        if (i==4) i=0; // i=4
        led1=0;
        led2=0;
        led3=0;
        led4=0;
        switch (i) {
            case 0:
                led1=1;
                break;
            case 1:
                led2=1;
                break;
            case 2:
                led3=1;
                break;
            case 3:
                led4=1;
        }
        i++;
        wait(0.1);
    }
}
```

## 9.2 配列を利用する方法

Pin を配列として登録し利用できる。プログラムは単純になる。

(led\_test\_2.cpp)

```
#include "mbed.h"

DigitalOut led[]={LED1, LED2, LED3, LED4};

int main() {
    unsigned int i=0;
    while (1) {
        led[0]=0;
        led[1]=0;
        led[2]=0;
        led[3]=0;

        led[i]=1;
        wait(0.1);

        i++;
        if(i==4) i=0;
    }
}
```

バリエーションとして左右に振るもの(ナイトライダー)を考える。配列内に同じ Pin があってももちろん構わない。

(led\_test\_2\_1.cpp)

```
#include "mbed.h"

DigitalOut led[]={LED1, LED2, LED3, LED4, LED3, LED2};

int main() {
    unsigned int i=0;
    while (1) {
        led[0]=0;
        led[1]=0;
        led[2]=0;
        led[3]=0;

        led[i]=1;
        wait(0.1);

        i++;
        if(i==6) i=0;
    }
}
```

### 9.3 BusOut とシフトレジスタを使う方法

デジタル出力をバスとして定義する方法もある。これを利用するとさらに楽になる。

(led\_test\_3.cpp)

```
#include "mbed.h"

BusOut leds(LED1, LED2, LED3, LED4);

main() {
    int i=0;
    while (1) {
        leds = 1 << i;
        wait(0.1);

        i++;
        if (i==4) i=0;
    }
}
```

同じくアレンジ版

(led\_test\_3\_2.cpp)

```
#include "mbed.h"

BusOut leds(LED1, LED2, LED3, LED4);

main() {
    while (1) {
        leds = 1; //0b0001
        wait(0.1);
        leds = leds << 1; //0b0010
        wait(0.1);
        leds = leds << 1; //0b0100
        wait(0.1);
        leds = leds << 1; //0b1000
        wait(0.1);
        leds = leds >> 1; //0b0100
        wait(0.1);
        leds = leds >> 1; //0b0010
        wait(0.1);
        leds = leds >> 1; //0b0001
        wait(0.1);
    }
}
```

## 10PWM

PWM はパルス変調(pulse Width modulation)の略で、一周期の内の On 時間の占める割合 (Duty 比) を変化させることで、ON-OFF しかないデジタル出力で、LED の明るさを変化させたり、モータの速度を変化させたりする。

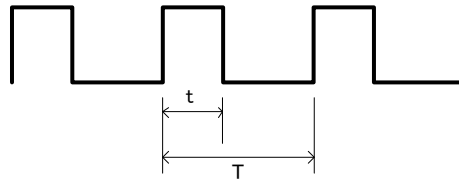


図 PWM の概念図  
 $t/T$  が Duty 比

mbed のハードウェア PWM で利用できるのは LED1-4, p 21-p 26 までである。☆Board Orange は LED1-4, p21-p23 が利用できる。ただし LED4 と p23 の同時使用が出来ません。mbed は、すべてのポートで同じ周期になる制約があります。

### 10.1 PWM (LED)

#### 10.1.1 プログラム 1

mbed 上の LED1-4 も PWM で接続されているので PWM の動作を簡単に確認することが出来る。  
(pwm\_led.cpp)

```
#include "mbed.h"

PwmOut led(LED1);

int main() {
    while(1) {
        for(float p = 0.0f; p < 1.0f; p += 0.01f) {
            led = p;
            wait(0.01);
        }
        for(float p = 1.0f; p > 0.0f; p -= 0.01f) {
            led = p;
            wait(0.01);
        }
    }
}
```

main 関数内の while 文で無限ループ

始めの for 文で led の出力を 0→1 にする。(0.01 秒ずつ 1%ずつ増加)

次の for 文で led の出力を 1→0 にする。(0.01 秒ずつ 1%ずつ減少)



## 10.1.2 プログラム 2

バリエーション(pwm\_led\_2)

```
#include "mbed.h"

PwmOut led1(LED1);
PwmOut led2(LED2);
PwmOut led3(LED3);
PwmOut led4(LED4);

int main() {
    unsigned int i=0;
    while (1) {
        if (i==6)i=0;// i=4
        switch (i) {
            case 0:
                led1=1;
                led2=0.5;
                led3=0.1;
                led4=0.01;
                break;
            case 1:
                led1=0.5;
                led2=1;
                led3=0.5;
                led4=0.1;
                break;
            case 2:
                led1=0.1;
                led2=0.5;
                led3=1;
                led4=0.5;
                break;
            case 3:
                led1=0.01;
                led2=0.1;
                led3=0.5;
                led4=1;
                break;
            case 4:
                led1=0.1;
                led2=0.5;
                led3=1;
                led4=0.2;
                break;
            case 5:
                led1=0.5;
                led2=1;
                led3=0.5;
                led4=0.1;
                break;
        }

        i++;
        wait(0.1);
    }
}
```

### 10.1.3 応用 1 理解しましょう。

## 10.2 PWM(モータを回す)

プログラムは LED の場合と変わらない。  
周期は 500Hz 以上で利用する。

フリーホイールダイオードの周波数特性と PWM 独特のノイズの音を考慮しながら周波数は決定する必要がある。(列車などは、聞いて不快でない周波数を選択している場合もある)

また、周波数を上げるためには、mbed と FET の間の回路の検討を要する。FET のゲートにたまった電荷を早く逃がす必要があり。フォトカプラ TLP250 などを間に利用する。

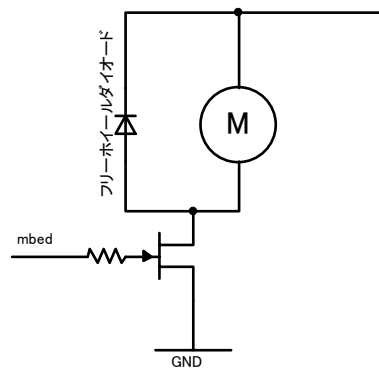


図 mbed とモータの接続図

### 10.2.1 応用 1

出力を 5 秒ごとに 10% ずつ増やして、100% 後に止めて LED を点灯するプログラムを作成する。

### 10.2.2 応用 2

周期(440Hz, 1kHz, 2kHz)を変えて、唸りがあるか確認する。  
負荷がある方がわかり易い。

## 11 デジタル入力（スイッチの動作回路）

### 11.1 回路

デジタル回路でスイッチを接続するときは下図左のように接続するのが一般的であるが、mbed はプルアップ抵抗を持っているので、プログラム内で Mode を PullUp モードに指定するれば、下図右のように接続しできる。（つまり信号線と GND の 2 本で良くなる）抵抗はモード指定を間違えた時に mbed が壊れないようにするための抵抗です（1kΩ程度で良い）。

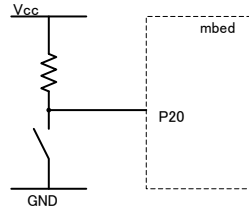


図 Normal モード時の接続（PullUp 回路）

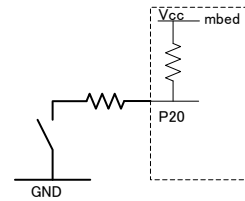


図 PullUp モード時の接続

実際にブレッドボードに回路を作ると下図のようになる

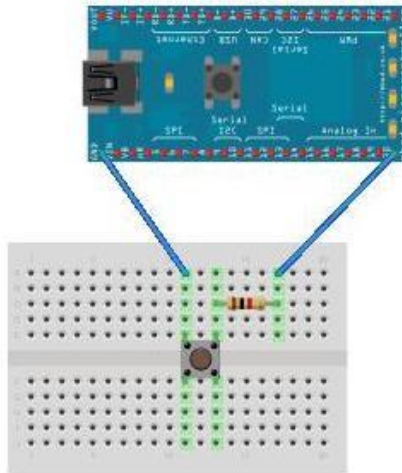


図 mbed とブレッドボードの接続

### 11.2 プログラム

スイッチを押している時だけ LED1 が点灯するようなプログラムを作る。重要なのはスイッチを押したときに DigitalIn で Off になることである。

```
#include "mbed.h"

DigitalOut led(LED1);
DigitalIn sw(p20);

int main() {
    sw.mode(PullUp);
    while(1) {
        if(!sw) led = 1;
        else led=0;
    }
}
```

main 関数内の while は無限ループ

if 文で sw=0(つまり押されている場合)なら led を点灯、それ以外は消灯。

### 11.3 応用 1 (割り込み)

割り込みは、プログラムを動作している時に、何かの原因で他の動作を入れることである。ハードによるものと、タイマーなどソフトによるものがある。ここではスイッチを押すごとに led を On-Off するプログラムを考える。

Pin19 と Pin20 は割り込みの使用出来ないので Pin10 に変更して実行する。

```
#include "mbed.h"

DigitalOut led1(LED1);
DigitalOut led4(LED4);
InterruptIn sw(p10);

void event() {
    led4=!led4;
}

int main() {
    led4=0;
    sw.mode(PullUp);
    sw.rise(&event);
    while (1) {
        led1 = !led1;
        wait(0.2);
    }
}
```

初期化で LED 1 を led1、LED4 を led4 に割り込み入力として p10 を sw として登録する。

event 関数は led4 の状態を反転するだけである。

まず、main 関数が呼ばれると

- led4 を消灯
- 割り込み入力を PullUp モード
- また sw を Rise 時に event 関数を呼ぶ  
(Rise 時 つまり 1→0(sw を押したときに対応)に変化した時)

設定する。

チャタリングの影響で上手く動作しない。下記の様に変更して確認する。

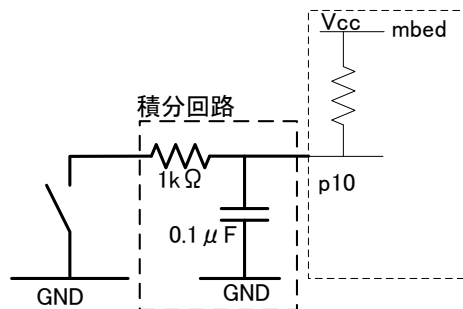


図 積分回路を加えた sw 回路

## 11.4 応用 2(チャタリングのチェック)

LED の点灯回路であれば前のプログラムで問題はない。しかし数を数えるカウンタなどは `sw` のチャタリングの影響を受ける為上手く動作しない。ここでは LCD を用いて何回ぐらい ON-OFF を繰り返しているかチェックする。

コードを入力する前に `TextLCD` のライブラリのインポートが必要

```
#include "mbed.h"
#include "TextLCD.h"

DigitalOut led(LED1);
DigitalIn sw(p20);
TextLCD lcd(p24, p26, p27, p28, p29, p30);
Ticker timer;
int i=0;

void tick(void)
{
    lcd.cls();
    lcd.printf(" %d ", i);
    if(sw.read()==0) led=1; else led=0;
}

int main() {
    int ssw=1;
    sw.mode(PullUp);
    timer.attach(&tick, 0.1);

    while(1) {
        if(sw.read()==1) {
            if(ssw==0) {
                ssw=1;
            }
        } else {
            if(ssw==1) {
                i++;
                ssw=0;
            }
        }
    }
}
```

液晶表示は、計測動作に影響するので 100mS 毎に更新する。

変数 `ssw` は前回の `sw` の状態を持っており、これが変化した時に変数 `i` をカウントアップする。

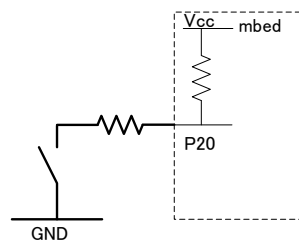


図 通常の入力回路

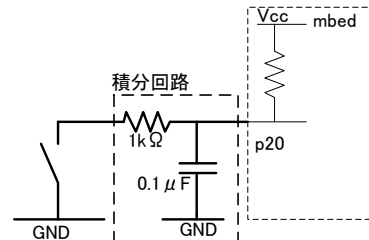


図 積分回路を加えた入力回路

積分回路中の抵抗  $R$  は短絡防止用に設けている  $1k\Omega$  をそのまま使用する。誤接続用の抵抗と  $0.1\mu F$  のコンデンサで積分回路(時定数  $0.1ms$ )にする。

実際に回路を変えてどれくらいチャタリングが発生するか確認する。

## 12AD コンバータ

AD コンバータはアナログ値をデジタルに変換することである。mbed の場合は p15 から p20 端子の 6 本で利用できる。AnalogIn.read 関数で 0 から Vcc(通常 3.3V)を 0 から 1 の形で読み取ることができる。

AnalogIn.read\_u16 関数では 0x0-0xFFFF の unsigned short で戻り値が戻ってくるが、そもそも mbed 上の NXP LPC1768 は 12 ビットの AD コンバータがないので 0xffff でマスクする例も多い。

また、NXP LPC1768 は高速な 10 ビットのモードとかも搭載しているが、mbed クラウド上のメーカ提供・推奨のライブラリには無い。しかし、mbed クラウド上で公開しているユーザもいる。

### 12.1 可変抵抗器

ここでは、アナログ JoyStick で確認をする。

※TextLCD のライブラリが必要。

```
#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30); // rs, e, d4-d7
AnalogIn x(p19);
AnalogIn y(p20);

int main() {
    lcd.cls();
    lcd.printf("ADC TEST\r\n");
    while (1) {
        lcd.locate(0, 1);
        lcd.printf("x=%4.2f y=%4.2f", x.read(), y.read());
        wait(0.1f);
    }
}
```

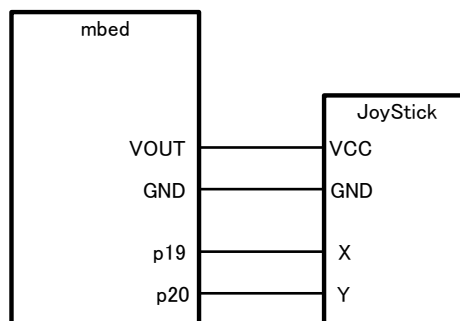


図 ジョイスティックとの接続図

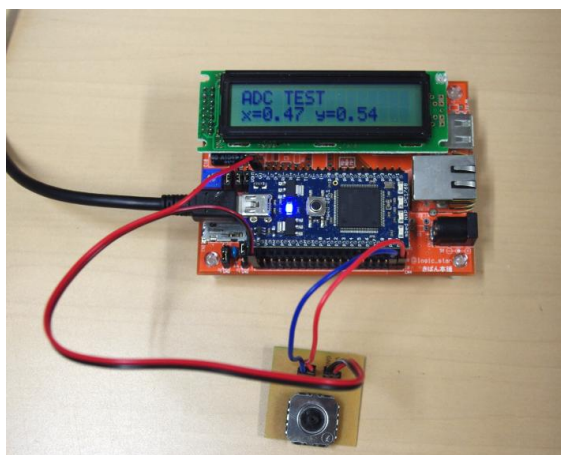


図 ジョイスティックとの接続写真

## 12.2 温度センサ

LM35D を利用した温度計を作成する。

LM35D は動作電圧 4V からであるので VIN もしくは VU (USB から供給される電源) を利用する。  
[室温程度なら 3.3V でも動作はする。]

```
#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30); // rs, e, d4-d7
AnalogIn temp(p20);

int main() {
    lcd.cls();
    lcd.printf("Temperature %n");
    while (1) {
        lcd.locate(0,1);
        lcd.printf("%4.2f Deg", temp.read()*330.0);
        wait(0.1f);
    }
}
```

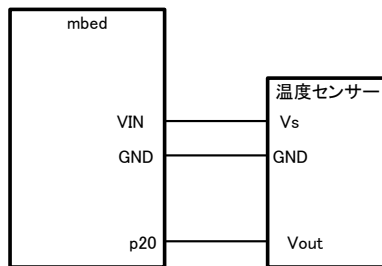


図 温度センサーとの接続

TO-92  
Plastic Package



BOTTOM VIEW

Order Number LM35CZ,  
LM35CAZ or LM35DZ  
See NS Package Number Z03A

図 LM35D のピン配置

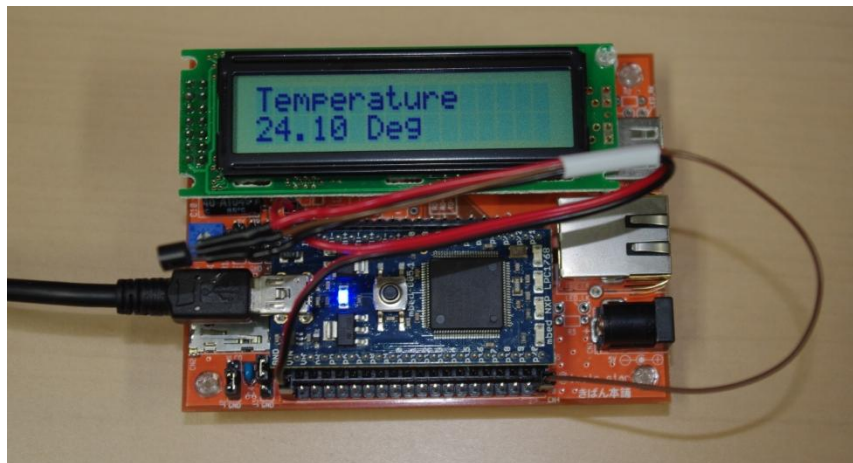


図 温度センサーの動作

## 13 USB スレーブ機能

PC をマスター、mbed をスレーブとした USB 接続をここでは説明する。mbed のハンドブックで説明がある通り、マウス、キーボード、複合デバイス（キーボード+マウス）、HID デバイス、USB シリアル、USB MIDI、USB Audio、USBMSD などが利用出来る。

### 13.1 USB マウスをエミュレーションする

#### 13.1.1 ハンドブックのサンプル実行（USB Mouse Hello World）

ハンドブックの中の Networking & Comms に USB マウスの項目がある。

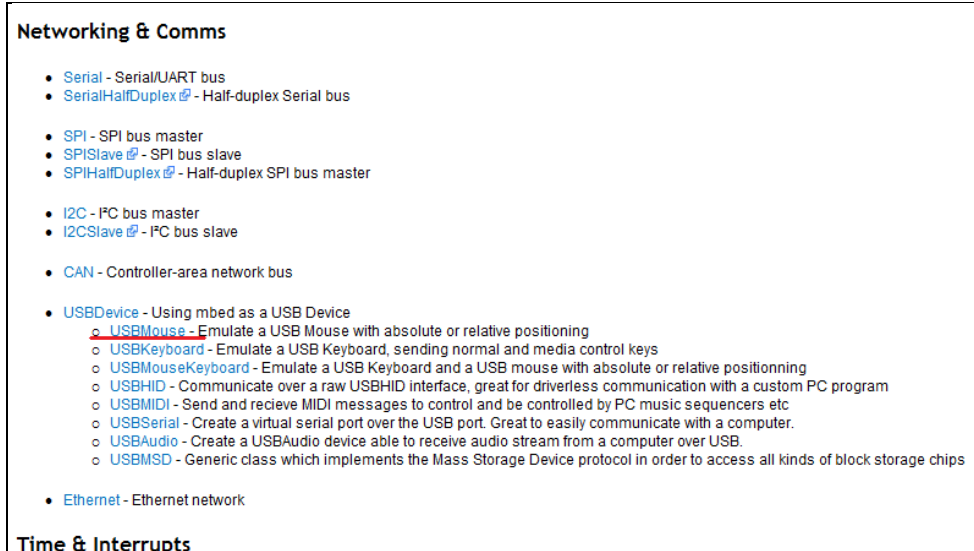


図 ハンドブックの USBMouse

(1) プログラムをインポート

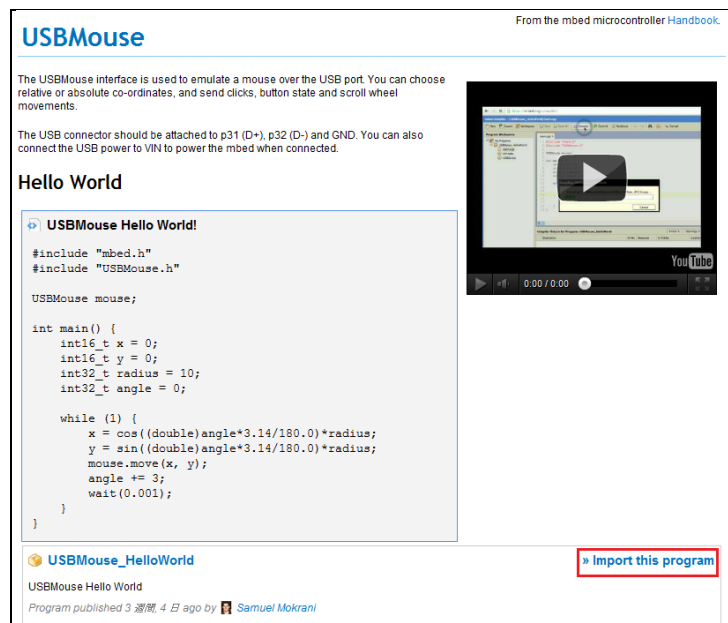


図 USBMouse のページ

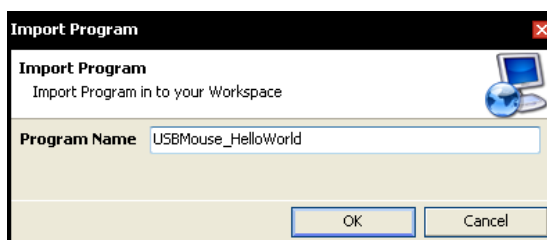


図 インポート先のダイアログ



(2) コンパイルして **mbed** に実行ファイルを入れる。

(3) **mbed** のマイクロ USB は外す。

(4) 図を参考に接続をする。

※**mbed** のマイクロ USB は **mbedIO** というものに繋がっているので、ここでつかう USB ではありません。注意！

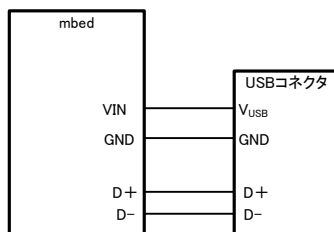


図 USB の接続図

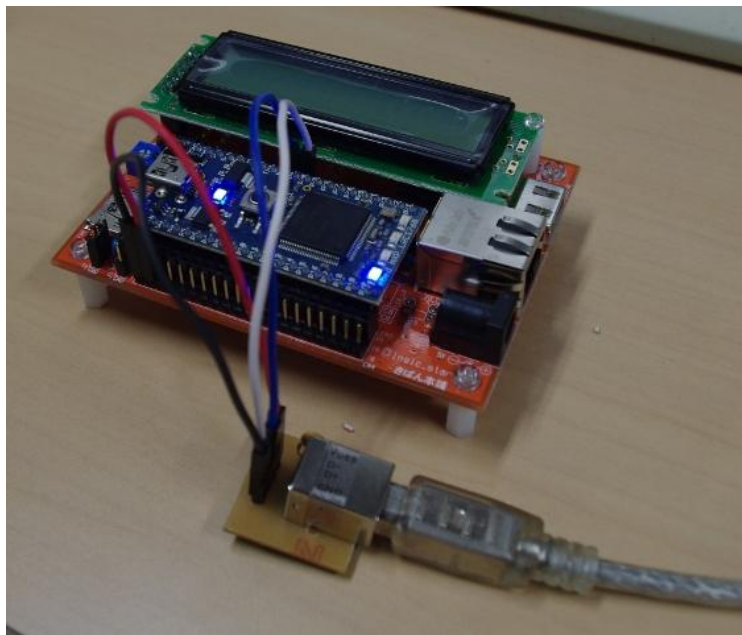


図 接続写真

(5) 正常であれば、PC に USB を接続するとデバイスを認識したのちカーソルが動く。

### 13.1.2 アナログマウス (JoyStick の接続)

アナログ入力で使用した JoyStick を接続して、アナログマウスを作る。

```
#include "mbed.h"
#include "USBMouse.h"

DigitalOut led(LED1);
USBMouse mouse;
AnalogIn x(p19);
AnalogIn y(p20);

int main() {
    float xx,yy;
    int mx,my;
    while(1) {
        xx=x.read();
        yy=y.read();
        if((xx<0.45) || (xx>0.55))mx=(xx-0.5)*10.0;
        else mx=0;
        if((yy<0.45) || (yy>0.55))my=(yy-0.5)*10.0;
        else my=0;
        mouse.move(mx,-my);
        wait(0.01f);
    }
}
```

少し中点の値に遊びがあった方が良いので 0.45 から 0.55 までを不感帯とする。

## 13.2 USB オーディオ

後述の音の部分に書いているので参照

## 14 USB マスター機能

### 14.1 USB フラッシュメモリの利用


#### 14.1.1 用意

##### (1) サンプルプログラムの取り込み

USB フラッシュメモリの書き込みには USB MSC (Mass Storage Class) Host ライブラリが必要です。CookBook の USB の USBMSDHost を参考にプログラムを取り込みます。

Inspired by [this thread](#) I've been working on making mbed read USB sticks. After making an initial port of NXP's [USBHostLite example](#), I got stuck since my mbed was an older, LPC2368 model, and did not have USB host. However, kind people helped me to get a 1768 unit much faster than Mouser could get it in stock, and with the [help of Ilya](#) I finally have a working example that plugs into the standard FATFileSystem.

### Usage

You can try my code by importing this project:  [MSCUsbHost](#)

To use it in your program, you will need to copy everything from the USBHostLite folder, and also MSCFileSystem.h/cpp files.

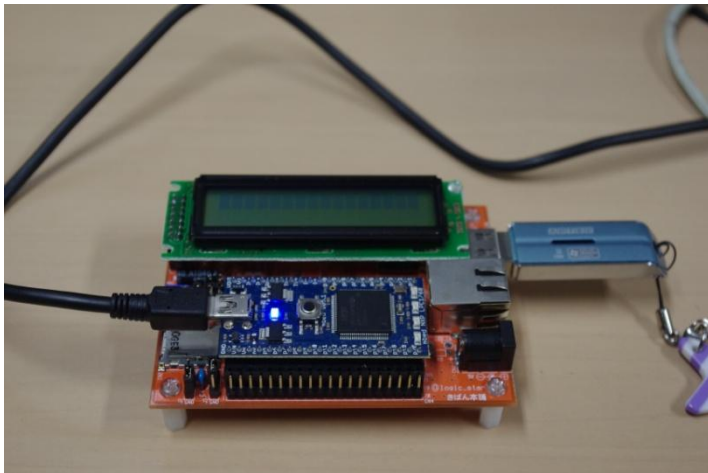
Usage is similar to LocalFileSystem or SDFileSystem classes:

```
#include "mbed.h"
#include "MSCFileSystem.h"

MSCFileSystem msc("msc"); // Mount flash drive under the name "msc"

int main()
{
    printf("\nTesting file write:\n");
    FILE *fp = fopen( "/msc/msctest.txt", "w");
    if ( fp == NULL )
```

##### (2) USB メモリを接続する。



##### (3) TeraTermなどで mbed の Serial on USB を接続する。

##### (4) Reset ボタンを押す。

### 14.1.2 ファイルの書き込み

```
#include "mbed.h"
#include "TextLCD.h"
#include "MSCFileSystem.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30);
MSCFileSystem usb("usb");

int main(void) {
    FILE *fp = fopen("/usb/test.txt", "w");
    if (NULL != fp) {
        fprintf(fp, "StarBoard Orange. mbed NXP LPC1768.");
        fclose(fp);
    } else {
        error("Open failed. %n");
        return 1;
    }
    lcd.cls();
    lcd.locate(0,0);
    lcd.printf("Finish");
    return 0;
}
```

### 14.1.3 ディレクトリの作成

```
#include "mbed.h"
#include "TextLCD.h"
#include "MSCFileSystem.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30);
MSCFileSystem usb("usb");

int main(void) {
    mkdir("/usb/mydir", 0777);
    FILE *fp = fopen("/usb/mydir/sdtest.txt", "w");
    if (fp == NULL) {
        error("Could not open file for write %n");
    }
    fprintf(fp, "Hello fun SD Card World!");
    fclose(fp);
    lcd.cls();
    lcd.locate(0,0);
    lcd.printf("Finish");
    return 0;
}
```

#### 14.1.4 ファイルの一覧表示

```
#include "mbed.h"
#include "TextLCD.h"
#include "MSCFileSystem.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30);
MSCFileSystem usb("usb");

int main(void) {
    DIR *d;
    struct dirent *p;
    lcd.cls();
    d = opendir("/usb");
    if ( d != NULL )
    {
        while ( (p = readdir(d)) != NULL )
        {
            lcd.printf("%s    %n", p->d_name);
            wait(0.5);
        }
    }
    lcd.locate(0,1);
    lcd.printf("** Finish ** ");
    return 0;
}
```

## 15m3pi で遊ぼう

Pololu 3pi を mbed で利用できるようにした m3pi は [cookbook](#) で公開されている。(Tutorials and Examples の Robotics) また、その中で、ライブラリも公開され、mbed から簡単に使用できる。

### Tutorials and Examples

- [ADC Performance](#) - How to get the best ADC performance from your mbed

### Internet of Things

- [RFID Tweeter](#) - Simple "Internet of Things" example using RFID and Twitter
- [Internet of Things](#) - mbed demo of 'the internet of things', using wifi to send sensor data
- [Websocket and Mbed](#) - Make your own Internet of Things project!

### Software Development

- [Writing a Library](#) - How to write your own library
- [Documenting a Library](#) - How to get API documentation automatically generated for your own library
- [Calling Library API Functions](#) - How to use the official mbed libraries
- [Using mbed with GCC and Eclipse](#) - Getting started using the mbed chip offline with Linux, Eclipse, and GCC
- [Using mbed with GCC and a text editor](#) - using the mbed chip offline (Windows, modifiable for Linux)
- [Using mbed libraries with GCC](#) - Offline compilation with mbed libraries
- [Bit-Banding - \(Non-Interruptible\) Atomic bit modification](#)
- [Object Oriented Programming Review](#) - A fairly simple review of OOP focusing on class inheritance and polymorphism. This is a simple game that you can use to build a much more complicated project.

### Hardware Tutorials

- [Pushbuttons and switches](#) - demo code and videos using internal pull-ups, switch debouncing, interrupts, and callbacks
- [Drivers, Relays, and Solid State Relays](#) - How to control high current or high voltage DC and AC devices using digital outputs on mbed
- [Serial Interrupts](#) - How to get started using serial interrupts with buffering and demo code
- [Power Management](#) - How to get started using power management features to reduce power and demo code
- [WatchDog Timer](#) - How to use the watchdog timer, brown-out detection, and a short code example
- [I2C Debug Tool](#) - Hit GUI buttons in RealTerm to experiment with a new I2C device without writing code.

### Robotics

- [Pololu m3pi](#) - The Pololu 3pi robot using mbed as the controller
- [mbed Rover](#) - Combining motors, QEI, PID control and an IMU
- [iRobot Create Robot or a Roomba](#) - How to get started using mbed for control
- [Sparkfun's Magician Robot base kit](#) - How to get started using mbed for control

☒ [mbed の Cookbook の m3pi のページ](#)

### 15.1 Hello World


最初に慣れる為に、m3pi\_HelloWorld を実行する。

### Hello World!

This simple program will drive the m3pi forward, spin it left, reverse it and spin it right.

[m3pi\\_HelloWorld](#) [» Import this program](#)

Hello world program that just gets the m3pi moving  
Program published 13 5月 2011 by [Chris Styles](#)



[You Tube Clip](#)

### Hardware

There is very little connectivity required between the mbed and the 3pi to make it all work.

☒ [m3pi の Cookbook のページ](#)

## 15.2 LED の使用

### 15.2.1 前提知識

Cookbook によると m3pi はハーフスピードで 720deg/sec であるのでフルスピードで 1 回転するには 0.25 秒である。フルスピードで回転するには左のモータを 1、右のモータを -1 に設定する。

### 15.2.2 プログラム

m3pi\_HelloWorld を書き換える。

```
lude "m3pi.h"

m3pi m3pi;

DigitalOut led1(p19);
DigitalOut led2(p18);
DigitalOut led3(p17);
DigitalOut led4(p16);
DigitalOut led5(p15);
DigitalOut led6(p14);
DigitalOut led7(p13);

int s1[]={0,1,0,0,0,1,0,1,1,1,1,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,1,0,0};
int s2[]={0,1,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0};
int s3[]={0,1,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0};
int s4[]={0,1,1,1,1,1,0,1,1,1,1,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0};
int s5[]={0,1,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0};
int s6[]={0,1,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0};
int s7[]={0,1,0,0,0,1,0,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,1,0,0,1,1,1,0,0};

int main() {
    int i,j;
    m3pi.locate(0,1);
    m3pi.printf("LO World");
    wait(2.0);
    m3pi.left(1); // Turn left at half speed
    m3pi.right(-1); // Turn left at half speed
    wait(0.5); // wait half a second
    for (j=0; j<10; j++) {
        for (i=0; i<31; i++) {
            led1=s1[i];
            led2=s2[i];
            led3=s3[i];
            led4=s4[i];
            led5=s5[i];
            led6=s6[i];
            led7=s7[i];
            wait(0.005);
        }
        wait(0.25-0.005*31);
    }
    wait(0.5); // wait half a second
    m3pi.stop();
}
```

### 15.2.3 実行例



図 m3pi の LED を光らせた例

## 15.3 ライントレースロボット

ラインレースロボットは m3pi の cookbook 内の Resources Line following activities にプログラムが用意されている。

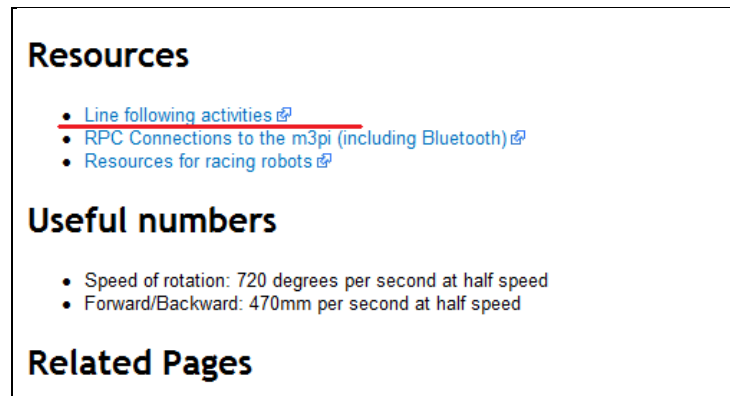


図 m3pi のクックブックのページ

PID を利用したプログラムは、パラメータさえあえばスムーズに動くのでお試しあれ！  
プログラム中の

```
#define MAX 1.0      で最大のスピードを  
#define P_TERM 1    で比例定数を  
#define I_TERM 0    で積分定数を  
#define D_TERM 20   で微分定数を
```

変更できる。

複雑なコースの場合は P\_TERM を 0.6 位でチャレンジしてみる。



## 16 Ethernet の利用方法

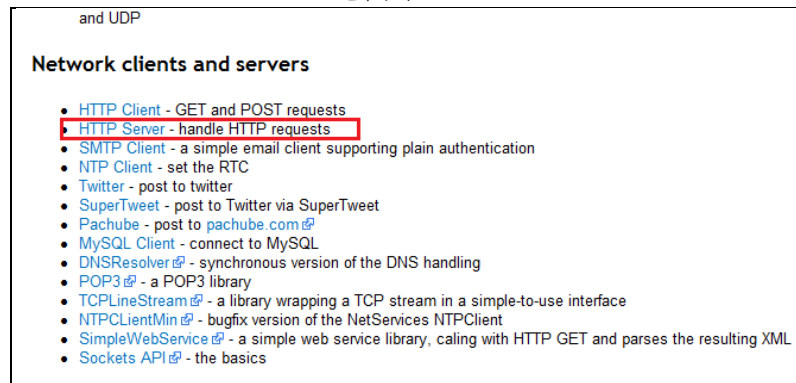
### 16.1 はじめに

mbed には、最初から Ethernet の物理層の IC が備わっている。したがって、RJ45 端子(トランス付き)を接続すると簡単に使用できるようになっている。mbed 用の Ethernet の関連のプログラム・ライブラリは数多くあるが、ここでは代表的に Cookbook に載っているものを使用する。

### 16.2 Web サーバ

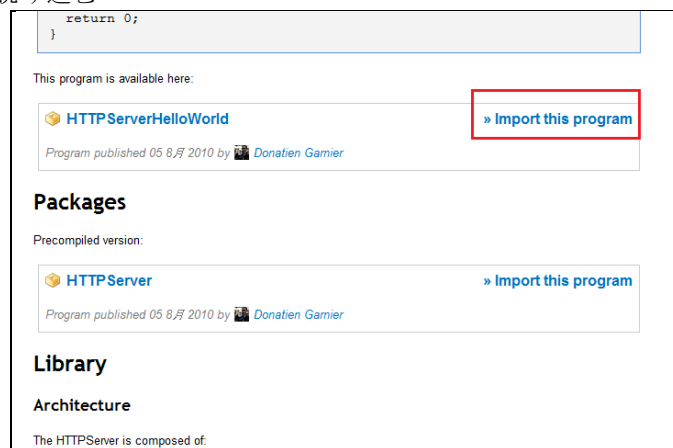
#### 16.2.1 はじめに(参考プログラムの読み込みと実行)

(1) Cookbook の HTTP Server のページを開く

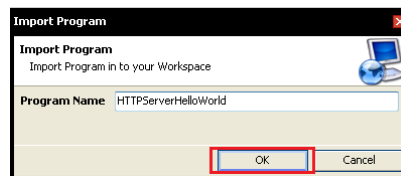


☒

(2) プログラムを読み込む



☒ サンプルプログラムの場所



☒ インポートのダイアログ

(3) コンパイルしてプログラムを mbed に保存する。

(4) TeraTerm など mbed の Serial on USB を接続する。

(5) Reset ボタンを押す。

- (6) PC の Web ブラウザを接続する。  
TeraTerm 上に DHCP で取得した IP アドレスが表示される。

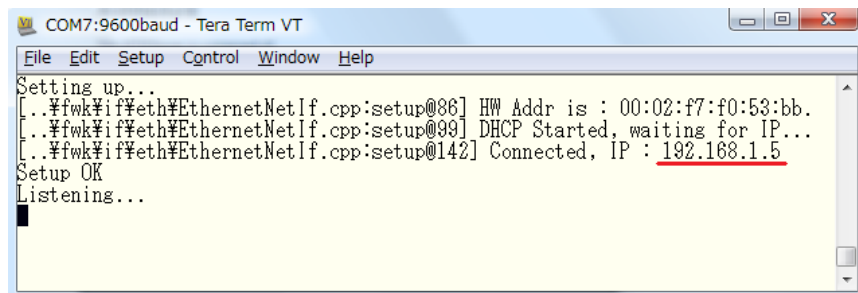


図 TeraTerm の画面

表示したアドレスを Web ブラウザで接続する。

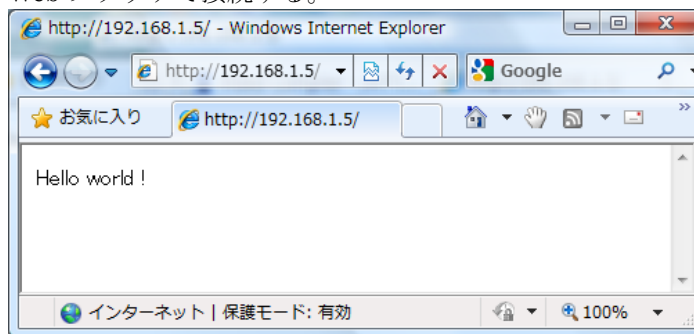


図 ブラウザ画面

“Hello world!” と表示されたら成功。

#### プログラムの説明

```
#include "mbed.h"
#include "EthernetNetIf.h"
#include "HTTPServer.h"

EthernetNetIf eth;
HTTPServer svr;

DigitalOut led1(LED1);

int main() {
    printf("Setting up... \n");
    EthernetErr ethErr = eth.setup(); //DHCP より IP アドレスを得る。
    if(ethErr)
    {
        printf("Error %d in setup. \n", ethErr);
        return -1;
    }
    printf("Setup OK \n");

    svr.addHandler<SimpleHandler>("/"); //Default handler
    svr.bind(80); //Port80 に bind する。

    printf("Listening... \n");

    Timer tm; //Timer
    tm.start(); //タイマーをスタートさせる。
    //Listen indefinitely
    while(true)
    {
        Net::poll(); //仕事がないかポーリングする。(ここの返答時間は処理量によって異なる)
```

```

る)
    if(tm.read() > .5) //0.5秒以上経ったらLEDの反転処理に入る
    {
        led1=!led1; //Show that we are alive
        tm.start(); //Timerを初期化する。
    }
}

return 0;
}

```

つまり LED が点滅している時は、処理がない(処理が終了している)事を示している。  
 svr.addHandler<SimpleHandler>("/"); は/(つまりルート)は SimpleHandler に接続している。  
 SimpleHandler は”hello world!”を返すだけのルーチン

## 16.2.2 普通の Web サーバ

普通の Web サーバはファイルを置くだけであり、そのファイルの位置と Web のルートを関連付けるだけで出来上がる。mbed の内蔵のフラッシュメモリは制約があるのでお勧めしません。

ここでは、USB フラッシュメモリ (USB MSD) を利用する場合を考えます。

(1) プログラムを作成します。

ここでは Web\_Server という名前にします。

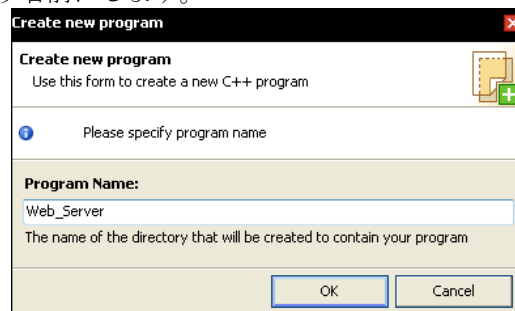


図 プログラム名の入力

(2) 必要なファイル(ライブラリ)を集める。

先に USB フラッシュメモリのプログラム (MSBUsbHost) から USBHostLite(フォルダ)、  
 FATFileSystem(ライブラリ)、MSCFileSystem.cpp、MSCFileSystem.h をコピーする。  
 HTTPServerHelloWorld から EthernetNetIF(ライブラリ)、HTTPServer(ライブラリ)をコピーする。

Web\_Server の上にドラッグ&ドロップでコピーできます。

(3) プログラムを作成する。(前のプログラムから赤字の部分を変更する)

(main.cpp)

```
#include "mbed.h"
#include "EthernetNetIf.h"
#include "HTTPServer.h"
#include "MSCFileSystem.h"

EthernetNetIf eth;
HTTPServer svr;

DigitalOut led1(LED1);
MSCFileSystem usb("usb");

int main() {
    printf("Setting up...%n");
    EthernetErr ethErr = eth.setup();
    if (ethErr) {
        printf("Error %d in setup.%n", ethErr);
        return -1;
    }
    printf("Setup OK%n");
    //svr.addHandler<SimpleHandler>("/"); //Default handler
    FSHandler::mount("/usb", "/"); //Mount /usb path on web root path
    svr.addHandler<FSHandler>("/"); //Default handler

    svr.bind(80);

    printf("Listening...%n");

    Timer tm;
    tm.start();
    //Listen indefinitely
    while (true) {
        Net::poll();
        if (tm.read() > .5) {
            led1=!led1; //Show that we are alive
            tm.start();
        }
    }

    return 0;
}
```

(4) USB メモリの用意

USB に html ファイルを置いて、☆Board orange に接続する。

ファイル名は 8.3 形式にする。

(index.htm)

```
<html>
<head>
<title>mbed web server</title>
</head>
<body>
<p>test mbed</p>
</body>
</html>
```

(5) 実行

リセットして、PC の Web ブラウザから接続できるか確認する。

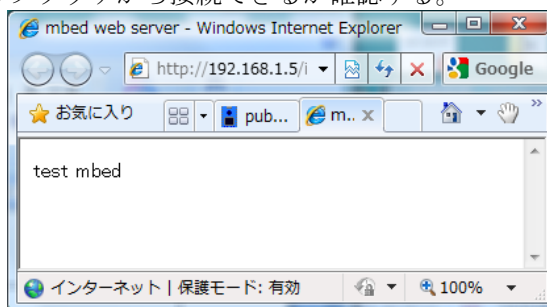


図 Web ページ

※このライブラリではファイル名は 8.3 形式で、マルチコネクトはサポートされていないようです。ファイル容量の大きな画像とかは開けません。

### 16.2.3 制御用 Web サーバ (その 1)

mbed を Http サーバとして mbed の LED のランプを制御するプログラムを作る。これは、普通の Web サーバにライブラリを加えることで可能なので、前章の続きでおこなう。

#### (6) プログラムを変更する

```
#include "mbed.h"
#include "EthernetNetIf.h"
#include "HTTPServer.h"
#include "MSCFileSystem.h"

EthernetNetIf eth;
HTTPServer svr;

DigitalOut led1(LED1, "led1");
DigitalOut led2(LED2, "led2");
DigitalOut led3(LED3, "led3");
DigitalOut led4(LED4, "led4");
MSCFileSystem usb("usb");

int main() {
    printf("Setting up... %n");
    EthernetErr ethErr = eth.setup();
    if (ethErr) {
        printf("Error %d in setup. %n", ethErr);
        return -1;
    }
    printf("Setup OK %n");

    Base::add_rpc_class<DigitalOut>();

    //svr.addHandler<SimpleHandler>("/"): //Default handler
    FSHandler::mount("/usb", "/"); //Mount /usb path on web root path
    svr.addHandler<FSHandler>("/"); //Default handler
    svr.addHandler<RPCHandler>("/rpc");

    svr.bind(80);

    printf("Listening... %n");

    //Timer tm;
    //tm.start();
    //Listen indefinitely
    while (true) {
        Net::poll();
        // if (tm.read() > .5) {
        //     led1=!led1; //Show that we are alive
        //     tm.start();
        // }
    }

    return 0;
}
```

#### (7) 実行と確認

コンパイルしてリセットを押す。

web ブラウザから

<http://mbedIP/rpc/led4/write 1>

とすると LED4 が点灯することを確認する。

<http://mbedIP/rpc/led4/write 0>

とすると LED4 が消灯する。

(8) ホームページ上から制御

ホームページ上からLEDを制御したい場合は、mbedのプログラムを改変する必要がなくなり、後はHTMLファイルやJavaScriptのファイルを用意するだけである。

(a) JavaScriptのファイルの用意

/cookbook/Interfacing-with-JavaScriptのmbedRPC.jsをUSBフラッシュメモリに保存する。

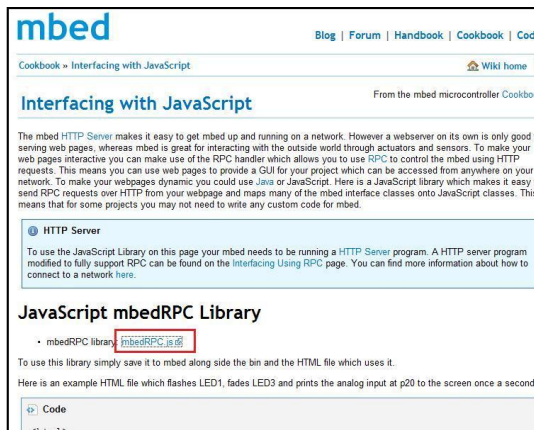


図 Cookbook から mbedRPC.js をダウンロード

(b)html ファイルを用意

(test.htm)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-2022-jp">
<title>mbed test web</title>
<script type="text/javascript" src="mbedRPC.js" language="javascript"></script>
<script type="text/javascript">
    mbed = new HTTPRPC();
    led1 = new DigitalOut(mbed, LED1);
    led2 = new DigitalOut(mbed, LED2);
    led3 = new DigitalOut(mbed, LED3);
    led4 = new DigitalOut(mbed, LED4);
</script>
</head>
<body>
<form action="test.htm" method="get">
<button type="submit" ID="btn_LED1a" onclick="led1.write(1)">LED1 点灯</button> <br >
<button type="submit" ID="btn_LED2a" onclick="led2.write(1)">LED2 点灯</button> <br >
<button type="submit" ID="btn_LED3a" onclick="led3.write(1)">LED3 点灯</button> <br >
<button type="submit" ID="btn_LED4a" onclick="led4.write(1)">LED4 点灯</button> <br >
<br>
<button type="submit" ID="btn_LED1b" onclick="led1.write(0)">LED1 消灯</button> <br >
<button type="submit" ID="btn_LED2b" onclick="led2.write(0)">LED2 消灯</button> <br >
<button type="submit" ID="btn_LED3b" onclick="led3.write(0)">LED3 消灯</button> <br >
<button type="submit" ID="btn_LED4b" onclick="led4.write(0)">LED4 消灯</button> <br >
<br>
</form>
<script type="text/javascript">
var led_1=led1.read();
var led_2=led2.read();
var led_3=led3.read();
var led_4=led4.read();

document.write("LED1 は");
if(led_1==0) document.write("消灯");else document.write("点灯");
document.write("しています<br>");

document.write("LED2 は");
if(led_2==0) document.write("消灯");else document.write("点灯");
```

```
document.write("しています<br>");

document.write("LED3 は");
if(led_3==0) document.write("消灯");else document.write("点灯");
document.write("しています<br>");

document.write("LED4 は");
if(led_4==0) document.write("消灯");else document.write("点灯");
document.write("しています<br>");
</script>
</body>
</html>
```

### 16.2.4 制御用 Web サーバ (その2)

☆BoardOrange 上の LCD に文字を書くサンプル

(9) ライブラリ TextLCD の追加

Cookbook の TextLCD ライブラリのページを開いて、Import this library into a program をクリックする。

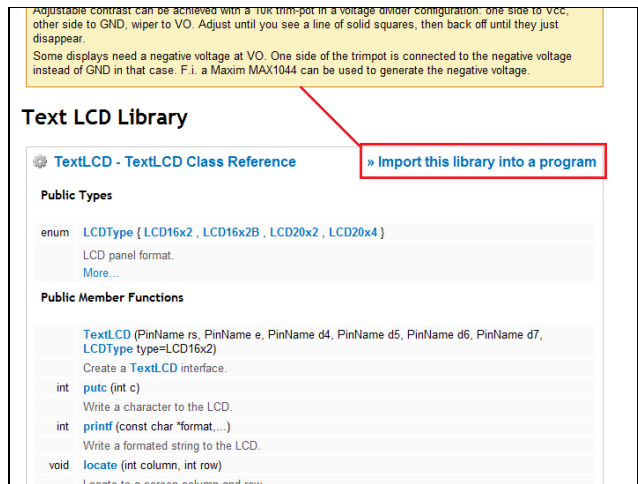


図 TextLCD ライブラリのページ

クリックするとどこに入れるか聞いてくるので適当に合わせます。

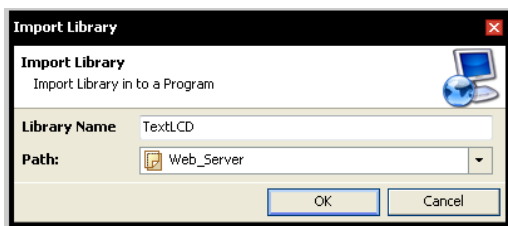
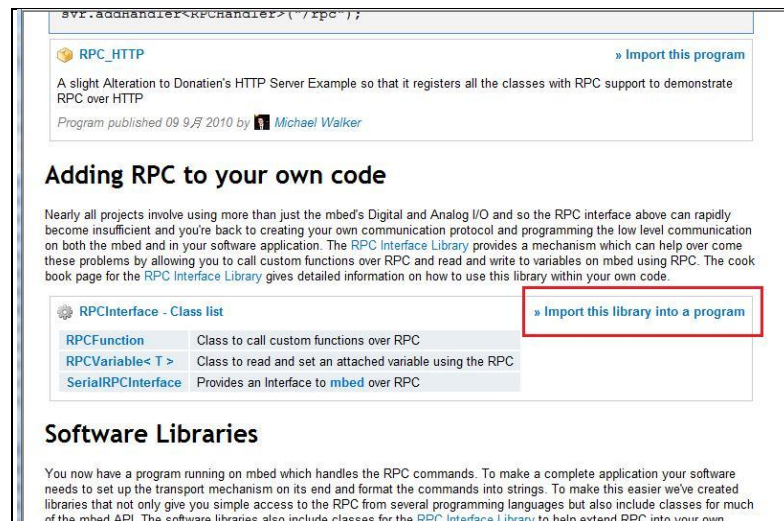


図 ライブラリのインポートダイアログ



## (10) ライブラリ RPC の追加

前のコードに RPC を追加します。Cookbook の Interfacing Using RPC のライブラリを TextLCD ライブラリと同様に Import します。



The screenshot shows the mbed IDE interface. At the top, there is a code editor with the line `svr.addHandler<RPCHandler>("/rpc");`. Below the editor, there is a section for the **RPC\_HTTP** library, which includes a description and a link to "Import this program". Below that, there is a section for the **RPCInterface - Class list**, which includes a table of classes and a link to "Import this library into a program". The link is highlighted with a red box. Below the class list, there is a section for **Software Libraries** with a paragraph of text.

図 RPC Interface ライブラリのインポート

## (11) プログラムの変更

赤字が変更した部分

```
#include "mbed.h"
#include "EthernetNetIf.h"
#include "HTTPServer.h"
#include "MSCFileSystem.h"
#include "TextLCD.h"
#include "RPCFunction.h"

EthernetNetIf eth;
HTTPServer svr;

DigitalOut led1(LED1, "led1");
DigitalOut led2(LED2, "led2");
DigitalOut led3(LED3, "led3");
DigitalOut led4(LED4, "led4");
MSCFileSystem usb("usb");

TextLCD lcd(p24, p26, p27, p28, p29, p30);
void LcdWrite(char *input, char *output);
RPCFunction rpcFunc(&LcdWrite, "LcdWrite");

void LcdWrite(char *input, char *output)
{
    lcd.locate(0, 1);
    lcd.printf("%s", input);
}

int main() {
    printf("Setting up... \n");
    EthernetErr ethErr = eth.setup();
    if (ethErr) {
        printf("Error %d in setup. \n", ethErr);
        return -1;
    }
    printf("Setup OK \n");
}
```

```
Base::add_rpc_class<DigitalOut>();

//svr.addHandler<SimpleHandler>("/"); //Default handler
FSHandler::mount("/usb", "/"); //Mount /usb path on web root path
svr.addHandler<FSHandler>("/"); //Default handler
svr.addHandler<RPCHandler>("/rpc");

svr.bind(80);

printf("Listening...%n");

//Timer tm;
//tm.start();
//Listen indefinitely
while (true) {
    Net::poll();
    // if (tm.read() > .5) {
    //     led1=!led1; //Show that we are alive
    //     tm.start();
    // }
}

return 0;
}
```

## (12) HTML ファイルの用意

前回の test.htm に加える。赤字が変更した部分。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-2022-jp">
<title>mbed test web</title>
<script type="text/javascript" src="mbedRPC.js" language="javascript"></script>
<script type="text/javascript">
    mbed = new HTTPRPC();
    led1 = new DigitalOut(mbed, LED1);
    led2 = new DigitalOut(mbed, LED2);
    led3 = new DigitalOut(mbed, LED3);
    led4 = new DigitalOut(mbed, LED4);
    led = new RPCFunction(mbed, "LcdWrite");
</script>
</head>
<body>
<form action="test.htm" method="get">
<button type="submit" ID="btn_LED1a" onclick="led1.write(1)">LED1 点灯</button> <br >
<button type="submit" ID="btn_LED2a" onclick="led2.write(1)">LED2 点灯</button> <br >
<button type="submit" ID="btn_LED3a" onclick="led3.write(1)">LED3 点灯</button> <br >
<button type="submit" ID="btn_LED4a" onclick="led4.write(1)">LED4 点灯</button> <br >
<br>
<button type="submit" ID="btn_LED1b" onclick="led1.write(0)">LED1 消灯</button> <br >
<button type="submit" ID="btn_LED2b" onclick="led2.write(0)">LED2 消灯</button> <br >
<button type="submit" ID="btn_LED3b" onclick="led3.write(0)">LED3 消灯</button> <br >
<button type="submit" ID="btn_LED4b" onclick="led4.write(0)">LED4 消灯</button> <br >
<br>
</form>
<script type="text/javascript">
var led_1=led1.read();
var led_2=led2.read();
var led_3=led3.read();
var led_4=led4.read();
```

```
document.write("LED1 は");
if(led_1==0) document.write("消灯");else document.write("点灯");
document.write("しています<br>");

document.write("LED2 は");
if(led_2==0) document.write("消灯");else document.write("点灯");
document.write("しています<br>");

document.write("LED3 は");
if(led_3==0) document.write("消灯");else document.write("点灯");
document.write("しています<br>");

document.write("LED4 は");
if(led_4==0) document.write("消灯");else document.write("点灯");
document.write("しています<br>");
</script>
<input type="text" id="textbox" ></input><button onclick="lcd.run(textbox.value);">送信</button>
<br>
</body>
</html>
```

(13) 実行  
リセットして実行

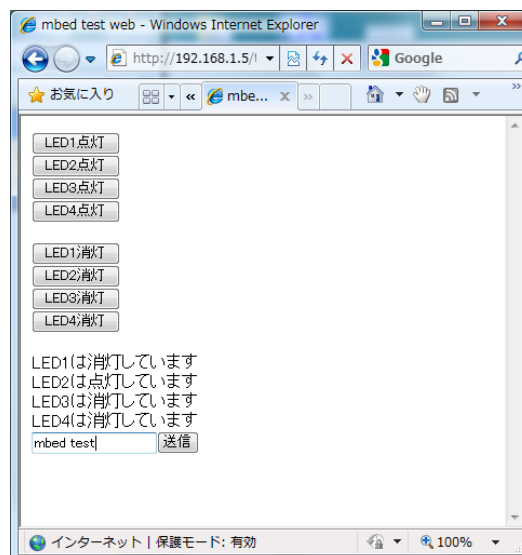


図 ブラウザ画面

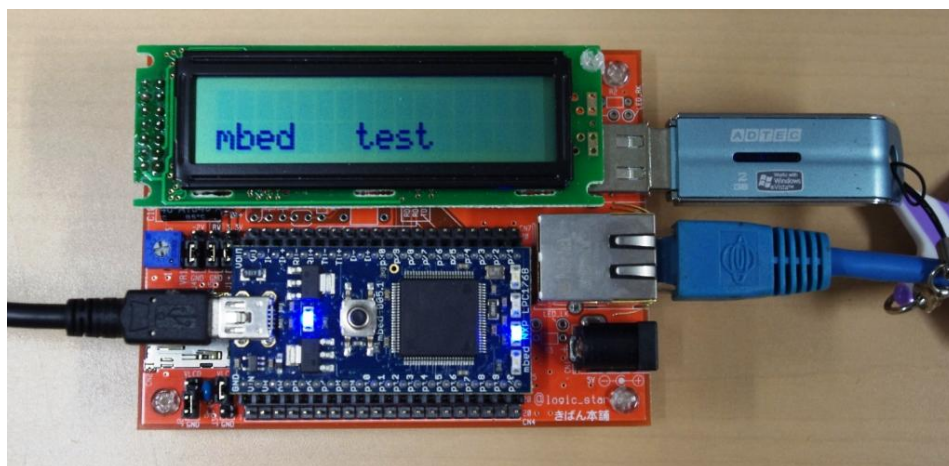


図 実行結果

## 16.3 NTP Client

NTP (Network Time Protocol) サーバはネットワーク上で正確な時間を得る為のものです。これはロガーなど作る時に便利である。ここではインターネット時計を作成する。ここでは、別なライブラリの使い方もしてみる。

### (1) プログラムの作成

New ボタンを押し新しいプログラムを作成する。(サンプルは”Clock”という名前で登録する)

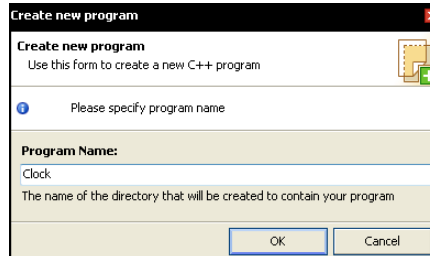


図 新しいプログラムのダイアログ

### (2) ライブラリのインポート

必要なライブラリは Ethernet、NTPClient、TextLCD である。

#### (a) Ethernet ライブラリ

Cookbook 中の TCP/IP Networking の Ethernet 選択する。

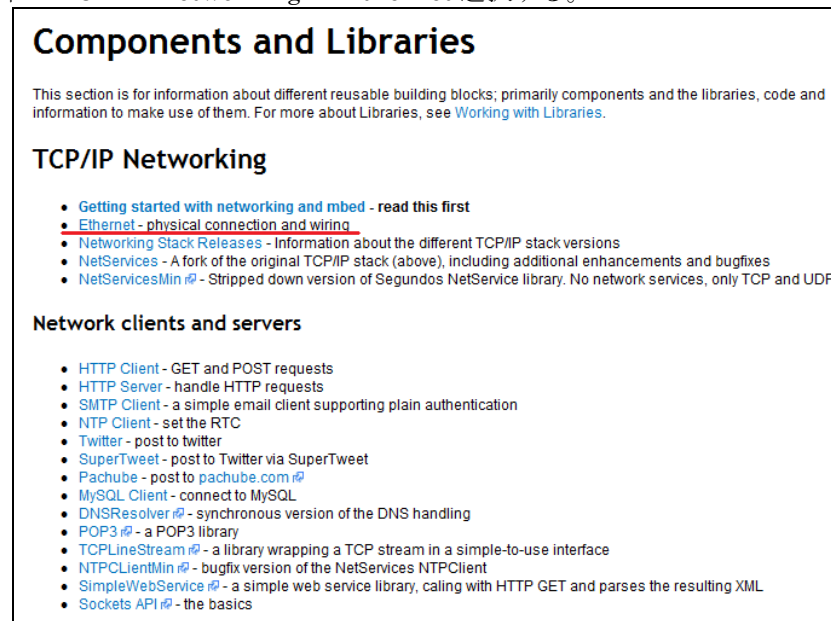


図 Cookbook の画面

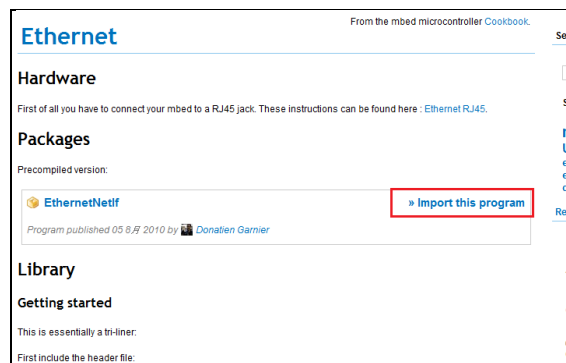


図 Ethernet ライブラリの画面

保存先を聞かれるのでそのまま OK を押す。

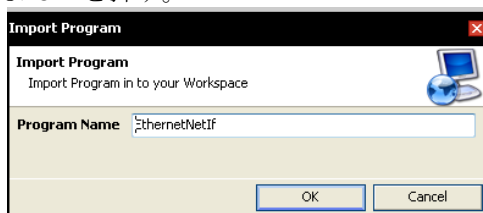


図 プログラムのインポートのダイアログ

EthernetNetIf プログラム(フォルダ)が出来る

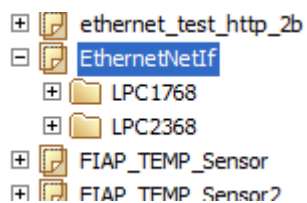


図 インポートされたプログラム

今回使用する mbed は NXP LPC1768 なので、LPC1768 を EthernetNetIf に Rename する。

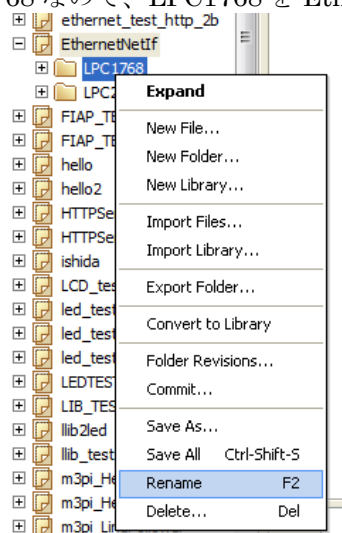


図 フォルダ上での右クリックメニュー

さらに Library に変換する

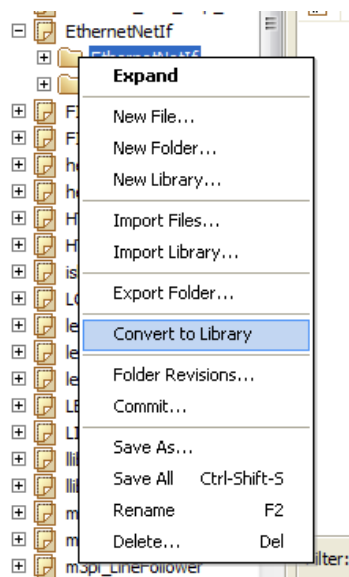


図 フォルダ上での右クリックメニュー

Library を Publish して、どこでも使えるようにする。

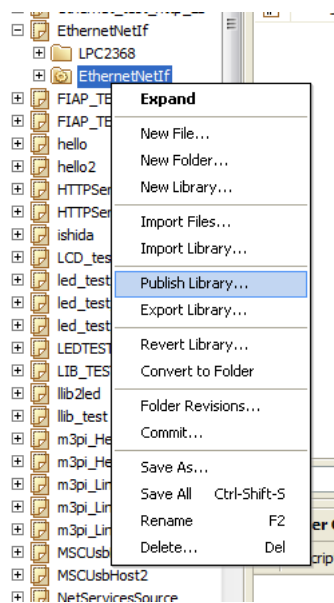


図 ライブラリの右クリックメニュー

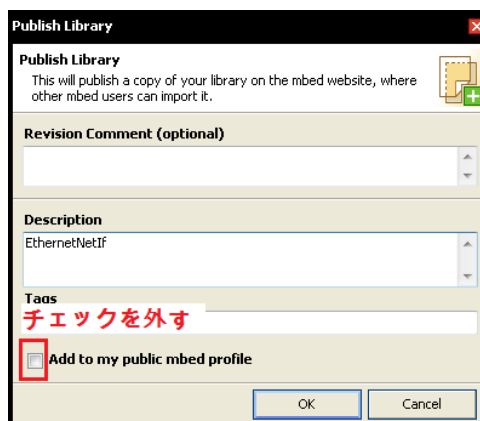


図 ライブラリのパブリッシュダイアログ

ライブラリに登録される。

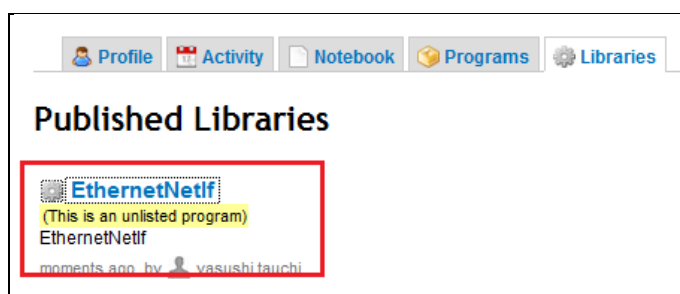
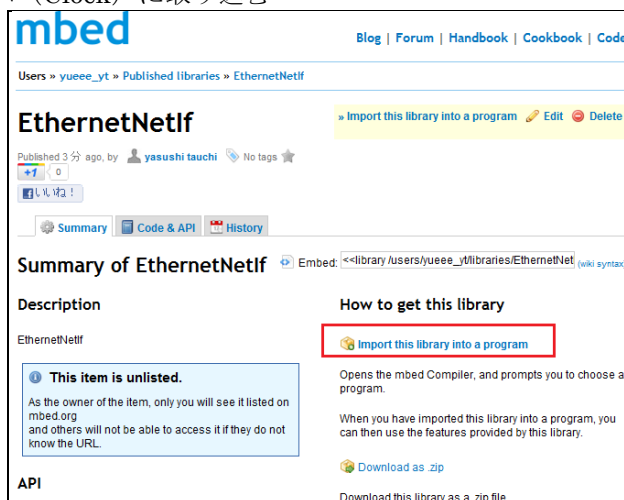
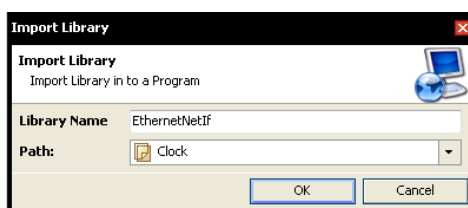


図 個人のライブラリ画面

ライブラリをプログラム (Clock) に取り込む



☒ Publish したライブラリのページ



☒ Import のダイアログ

(b)NTP Client ライブラリ

Cookbook 中の Network clients and servers 内の NTP クライアントを選択する。



☒ mbed ホームページの Cookbook

保存先を聞かれるのでそのまま OK を押す。



☒ NTPClient のページ

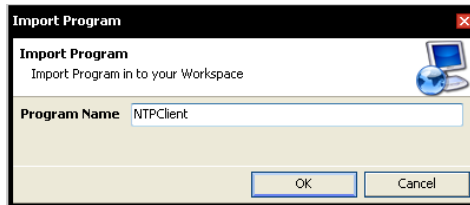


図 保存先を問うダイアログ

インポートできるとプログラムが加わる。

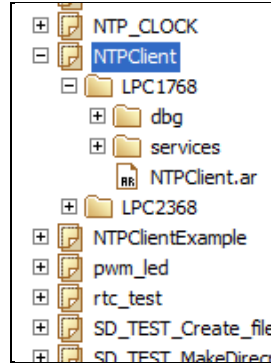


図 加わったプログラム

同様に今回使用する mbed は LPC1768 なので、LPC1768 を NTPClient に Rename する。

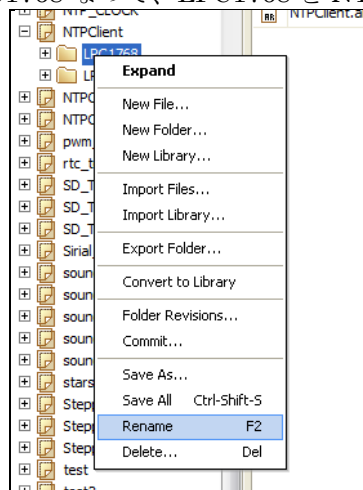


図 フォルダ名の変更

さらに Library に変換する

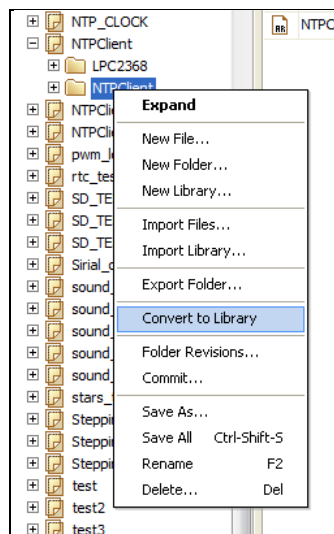
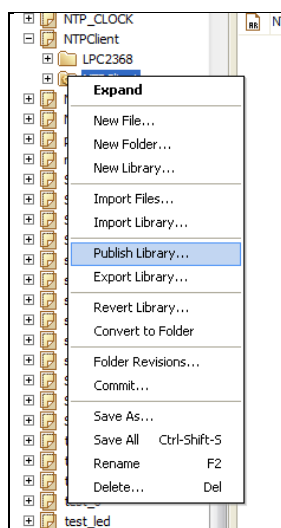


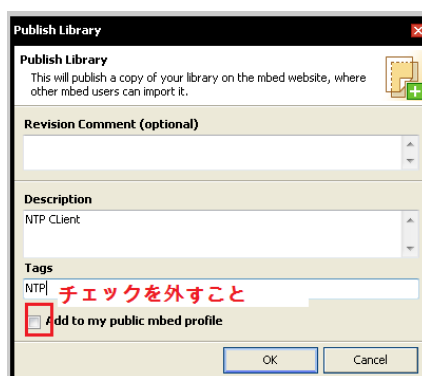
図 ライブラリに変更



Library を Publish して、どこでも使えるようにする。

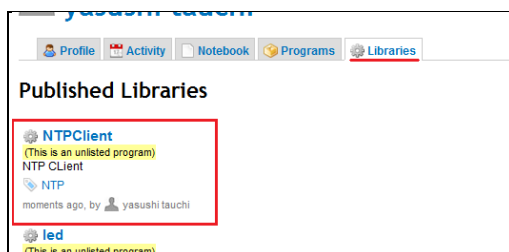


☒ ライブラリの Publish



☒ Publish のダイアログ

ライブラリに登録される。



☒ 個人のライブラリのページ

ライブラリをプログラムに取り込む

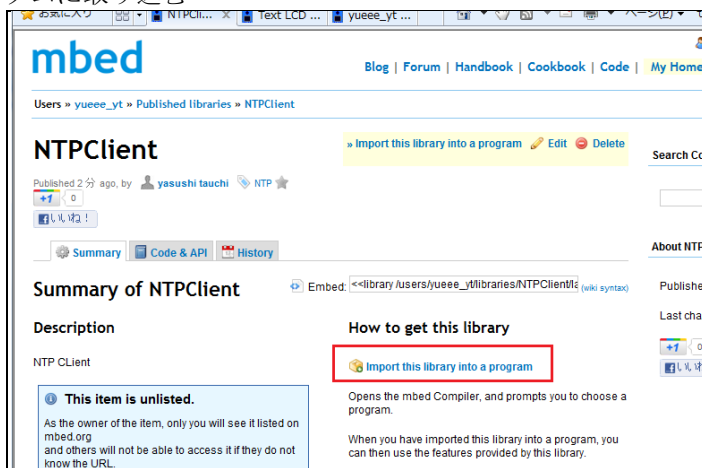


図 ライブラリのページ  
(Import this library into a program をクリック)

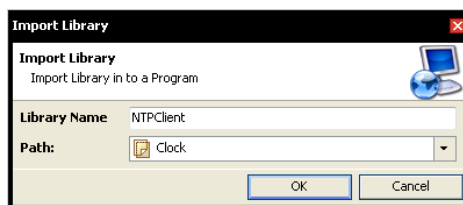


図 インポート先を問うダイアログ

### (c)TextLCD ライブラリ

TextLCD のライブラリはこれまで同様の方法でインポートする。

Cookbook のテキストライブラリのページを開く。

Import this library into a program をクリックする。

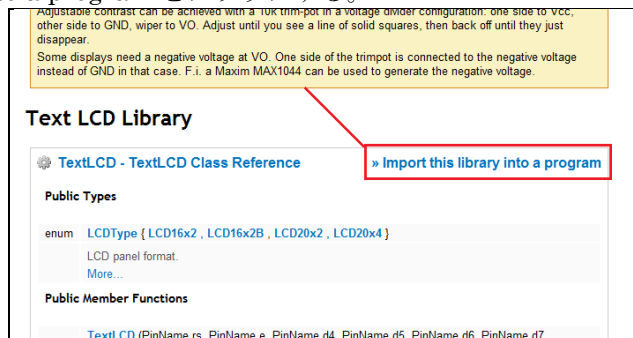


図 TextLCD ライブラリのページ

クリックするとどこに入れるか聞いてくるので同じプログラム名を選択する。

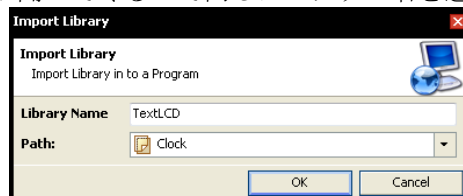


図 ライブラリをどのプログラムにインポートするか問うダイアログ

Clock のプログラム内は以下のようなライブラリ構成になる

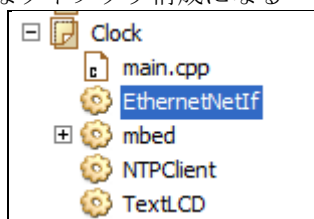


図 Clock のフォルダ内容

## (2) プログラムの作成

main.cpp 内のプログラムを書き換える。

```
#include "mbed.h"
#include "EthernetNetIf.h"
#include "NTPClient.h"
#include "TextLCD.h"

EthernetNetIf ethif; //for DHCP
//EthernetNetIf ethif(IpAddr (xxx, xxx, xxx, xxx), IpAddr (xxx, xxx, xxx, xxx),
IpAddr (xxx, xxx, xxx, xxx), IpAddr (xxx, xxx, xxx, xxx)); //for Static IP Address

NTPClient ntp;
TextLCD lcd(p24, p26, p27, p28, p29, p30);

int main() {
    char buffer[17];
    lcd.locate(0,1);
    lcd.printf("Start\r\n");
    lcd.locate(0,1);
    lcd.printf("Setting up... \r\n");
    EthernetErr ethErr = ethif.setup();
    if (ethErr) {
        lcd.locate(0,1);
        lcd.printf("Error %d in setup.\r\n", ethErr);
        return -1;
    }
    lcd.locate(0,1);
    lcd.printf("Setup OK\r\n");
    IpAddr ethIp=ethif.getIp();
    lcd.locate(0,1);
    lcd.printf("%d.%d.%d.%d", ethIp[0], ethIp[1], ethIp[2], ethIp[3]);
    wait(1.0f);

    time_t ctTime;

    Host server(IpAddr(), 123, "ntp.nict.jp"); //near ntp server
    ntp.setTime(server);
    //UTC-->JST +9Hour (32400Sec)
    ctTime = time(NULL);
    ctTime+=32400;
    set_time(ctTime);

    lcd.cls();
    while (1) {
        lcd.locate(0,0);
        ctTime = time(NULL);
        lcd.locate(0,0);
        strftime(buffer, 17, "%Y/%m/%d(%a)", localtime(&ctTime));
        lcd.printf("%s", buffer);
        lcd.locate(0,1);
        strftime(buffer, 17, "%X", localtime(&ctTime));
        lcd.printf("%s", buffer);
        wait(1.0f);
    }
    return 0;
}
```

※NTP Server は近くのアドレスに変更する。

## 17IEEE1888 を利用する

IEEE1888(Ubiquitous Green Community Control Network Protocol)はスマートグリッドなどのエネルギーを取り扱うプロトコルである。詳しくは東大グリーン ICT プロジェクトのサイト (<http://www.gutp.jp/>) やトランジスタ技術 2012 年 1,2 月号を参照の事

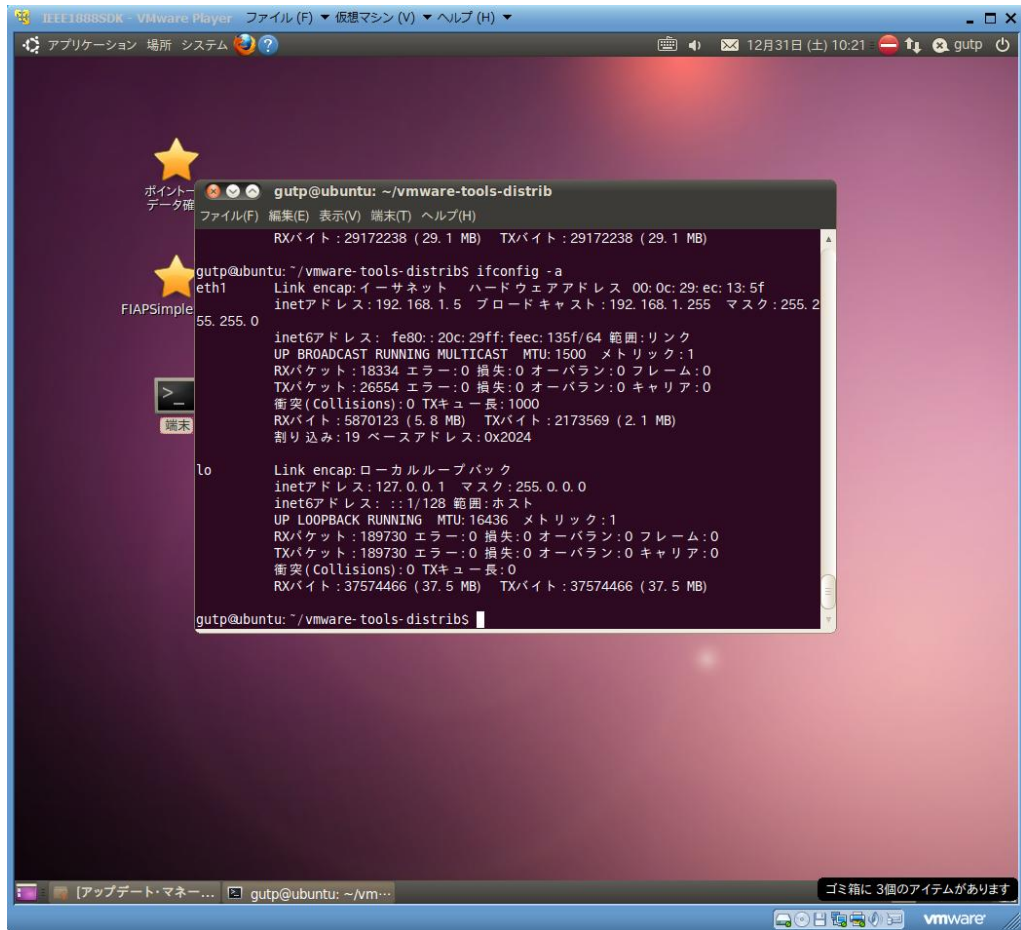
### 17.1 IEEE1888SDK

IEEE1888SDK は GUTP のホームページにある。 (<http://www.gutp.jp/>)

インストールドキュメントは SDK 中にある。

(要 VMplayer ダウンロード)

動作後は IP アドレスを確認する。(端末で `ifconfig -a`)



```
IEEE1888SDK - VMware Player  ファイル(F)  仮想マシン(V)  ヘルプ(H)
アプリケーション  場所  システム
12月31日(土) 10:21  gutp

gntp@ubuntu: ~/vmware-tools-distrib
RXバイト : 29172238 (29.1 MB)  TXバイト : 29172238 (29.1 MB)

gntp@ubuntu: ~/vmware-tools-distrib$ ifconfig -a
eth1
Link encap:イーサネット  ハードウェアアドレス 00:0c:29:ec:13:5f
inetアドレス:192.168.1.5  ブロードキャスト:192.168.1.255  マスク:255.255.255.0
inet6アドレス: fe80::20c:29ff:feec:135f/64 範囲:リンク
UP BROADCAST RUNNING MULTICAST  MTU:1500  メトリック:1
RXパケット:18334 エラー:0 損失:0 オーバラン:0 フレーム:0
TXパケット:26554 エラー:0 損失:0 オーバラン:0 キャリア:0
衝突(Collisions):0 TXキュー長:1000
RXバイト:5870123 (5.8 MB)  TXバイト:2173569 (2.1 MB)
割り込み:19 ベースアドレス:0x2024

lo
Link encap:ローカルループバック
inetアドレス:127.0.0.1  マスク:255.0.0.0
inet6アドレス: ::1/128 範囲:ホスト
UP LOOPBACK RUNNING  MTU:16436  メトリック:1
RXパケット:189730 エラー:0 損失:0 オーバラン:0 フレーム:0
TXパケット:189730 エラー:0 損失:0 オーバラン:0 キャリア:0
衝突(Collisions):0 TXキュー長:0
RXバイト:37574466 (37.5 MB)  TXバイト:37574466 (37.5 MB)

gntp@ubuntu: ~/vmware-tools-distrib$
```

図 IEEE1888 SDK 実行画面

## 17.2 FIAP Hello World

### 17.2.1 インポート

FIAP Hello World を Import する。

([http://mbed.org/users/yueee\\_yt/notebook/fiap/](http://mbed.org/users/yueee_yt/notebook/fiap/) 参照)

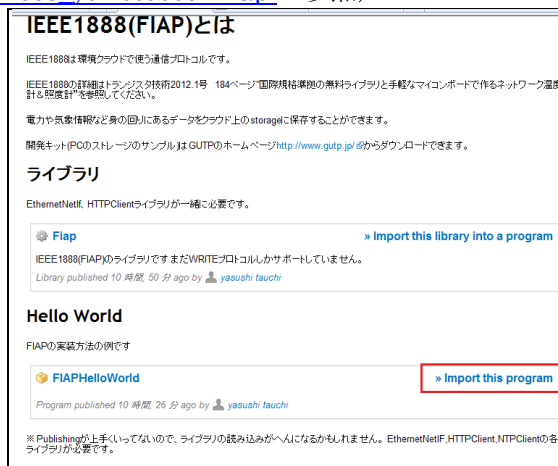


図 FIAP のページ

### 17.2.2 プログラムの修正

- (1) 6 行目 `#define ntp_server "ntp server address"`  
ntp server address の部分を最寄りの NTP サーバに変更する。
- (2) 15 行目の `FIAP fiap(http://192.168.1.5/axis2/services/FIAPStorage...`  
192.168.1.5 の部分を起動した SDK の IP アドレスに変更する。

### 17.2.3 コンパイルと実行

修正が終わったらコンパイルして BIN ファイルを mbed へインストールする。  
mbed(☆Board Orange)に LAN ケーブルを接続してリセットボタンを押し、実行(実行は約 30 秒程度かかります)

- LED1 が点灯したら DHCP 成功
- LED2 が点灯したら NTP 成功
- LED3 が点滅したら SDK へのデータ転送失敗
- LED4 が点滅したら SDK へのデータ転送成功

転送が成功したら SDK 内のポイント一覧データ確認を選択する(ダブルクリック)。  
[http://test.fiap.org/mbed\\_hello/P1](http://test.fiap.org/mbed_hello/P1) 及び [http://test.fiap.org/mbed\\_hello/P2](http://test.fiap.org/mbed_hello/P2) がある事を確認  
これが今回、mbed からアップロードされたデータである。



図 SDK のポイント一覧、データ確認画面

### 17.2.4 問題

プログラムを理解してみましょう

## 17.3 温度センサー端末を作成する。

### 17.3.1 必要なプログラム・ライブラリをインストール

(1) FIAPHelloWorld を FIAP\_TEMP としてインポートする。

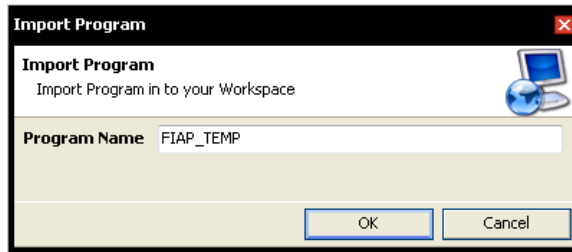


図 Import 画面

(2) TextLCD のライブラリをインポートする。

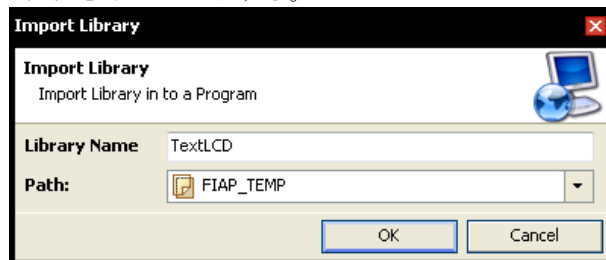


図 Import 画面

### 17.3.2 プログラムの変更

プログラムを変更する。

```
#include "mbed.h"
#include "EthernetNetIf.h"
#include "NTPClient.h"
#include "fiap.h"
#include "TextLCD.h"

#define ntp_server "NTP Server Address"

DigitalOut led1(LED1);
DigitalOut led2(LED2);
DigitalOut led3(LED3);
DigitalOut led4(LED4);

EthernetNetIf eth;
NTPClient ntp;
FIAP fiap("http://192.168.1.5/axis2/services/FIAPStorage", "http://test.fiap.org/mbed_temp");

TextLCD lcd(p24, p26, p27, p28, p29, p30);
AnalogIn temp(p20);
Ticker timer;

char timezone[] = "+09:00"; // JST
int ret;
time_t next_record;

struct fiap_element element[]={
    {"TEMP1", NULL, NULL, timezone},
    // {"P2", NULL, NULL, timezone},
};

void send_fiap_storage() {
    //data initialize
    char buffer[17];
```

```

float temp_data=temp.read()*330.0;
time_t seconds = time(NULL);
strftime(buffer, 17, " %X ", localtime(&seconds));
lcd.locate(0,0);
lcd.printf("%s", buffer);
lcd.locate(0,1);
lcd.printf("TEMP=%4.1f Deg", temp_data);
if (seconds>=next_record) {
    next_record+=60;
    char data[5];
    sprintf(data, "%4.1f", temp_data);
    element[0].value=data;
    element[0].t=localtime(&seconds);
    //do post
    ret=fiap.post(element, 1);
    return ;
}
}

int main() {
    led1=led2=led3=led4=0;
    lcd.printf("FIAP TEMP");
    fiap.debug_mode=true;
    //EthernetNetIf initialize
    EthernetErr ethErr = eth.setup();
    if (ethErr) {
        // lcd.locate(0,1);
        // lcd.printf("Error %d in setup.¥n", ethErr);
        return -1;
    }
    led1=1;
    //NTPClient initialize
    Host server(IpAddr(), 123, ntp_server);
    NTPResult Ntp=ntp.setTime(server);
    //UTC-->JST +9Hour(32400Sec)
    time_t ctTime;
    ctTime = time(NULL);
    ctTime+=32400;
    set_time(ctTime);
    led2=1;
    next_record = (time(NULL)/60+1)*60;
    timer.attach(&send_fiap_storage, 1);
    //do
    if (ret!=0) {
        led4=0;
        while (1) {
            led3=!led3;
            wait(0.5);
        }
    } else {
        led3=0;
        while (1) {
            led4=!led4;
            wait(0.5);
        }
    }
}
}

```

これは1秒ごとにデータを収集し、1分ごとの0秒の時点のデータをアップロードするプログラムです。

### 17.3.3 実行

温度センサーはアナログ入力と同様 p20 に接続する。

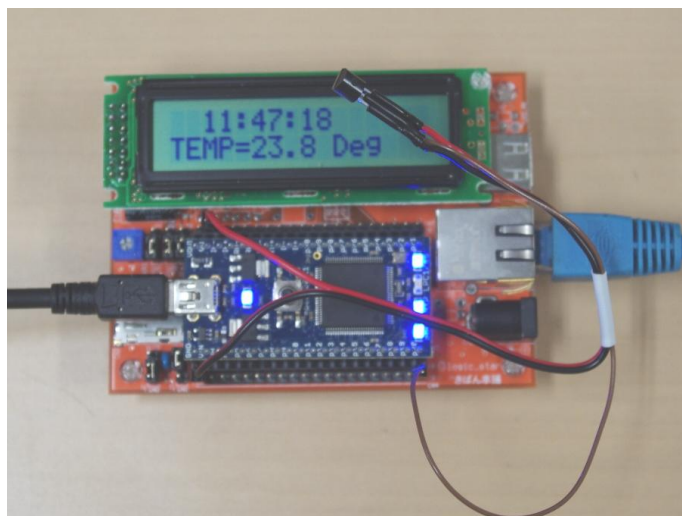


図 実行

実行し、数分後、SDK 内でデータの表示が出来る。ポイント一覧データ確認の PointID をクリックすると時系列のデータが表示出来る。

Point ID	Time	Value
<a href="http://test.fiap.org/mbed_hello/P1">http://test.fiap.org/mbed_hello/P1</a>	2012-01-02T10:53:31.000+09:00	009
<a href="http://test.fiap.org/mbed_hello/P2">http://test.fiap.org/mbed_hello/P2</a>	2012-01-02T10:53:31.000+09:00	090
<a href="http://test.fiap.org/mbed_temp/TEMP1">http://test.fiap.org/mbed_temp/TEMP1</a>	2012-01-02T15:10:00.000+09:00	23.4
<a href="http://test.fiap.org/mbed_yamanuchi-u.ac.jp">http://test.fiap.org/mbed_yamanuchi-u.ac.jp</a>		

図 ポイント一覧データ確認

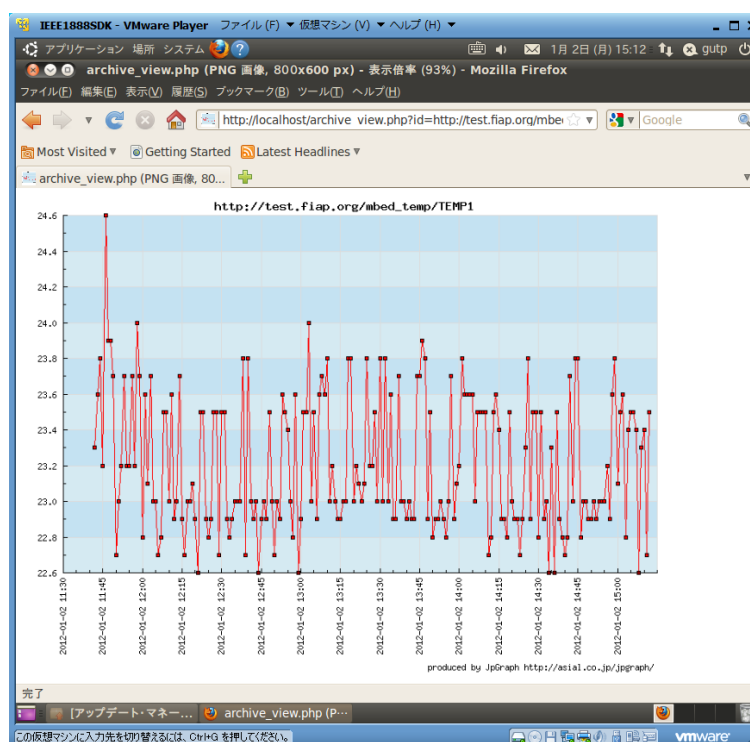


図 データの表示



## 18音の出力

mbed(マイコン)単体で、音楽を奏でる方法を考えると単純には下記のものがある。

- 1.DigitalOut によるもの
- 2.PWM によるもの
- 3.DAC によるもの

1.2.は矩形波が基本、3は様々な音を出すことが出来る。

### 18.1 必要な知識

音の周波数は等比数列で表すことができます。1 オクターブあがると 2 倍の周波数になる。  
1 オクターブあげるには周波数を 2 倍、2 オクターブあげるには 4 倍する。

表 音階と周波数 1)

	周波数
ド	261.626
レ	293.665
ミ	329.628
ファ	349.228
ソ	391.995
ラ	440.000
シ	493.883

圧電ブザーを鳴らすのには矩形波がよい。

参考：

- 1) FlawTips の音的ウェブ(ドレミの音階(平均律)に対応した周波数の表),  
<http://flawtips.ami.amigasa.jp/blog/060315.html>

## 18.2 DigitalOut による音だし

### 18.2.1 接続

圧電ブザーを p5 に接続する。

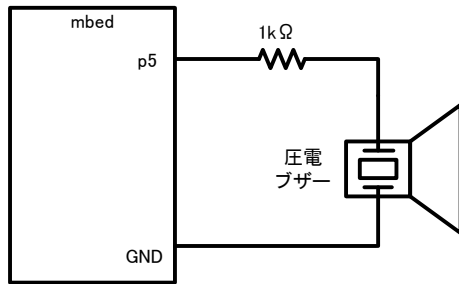


図 配線図

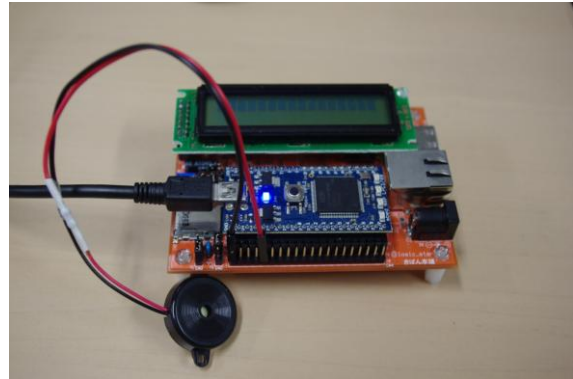


図 接続図

### 18.2.2 プログラム

Timer 割り込みの周期を変えて音を出しています。  
1 周期は ON-OFF の 2 回の割り込みが必要です。

```
#define mC 261.626
#define mD 293.665
#define mE 329.628
#define mF 349.228
#define mG 391.995
#define mA 440.000
#define mB 493.883

#include "mbed.h"

DigitalOut sp1(p5);
Ticker timer;

int oto=0;

void tick(void)
{
    sp1.write(oto);
    oto=!oto;
}

int main() {
    float mm[]={mC,mD,mE,mF,mG,mA,mB,mC*2};
    int i;
    for (i=0;i<sizeof(mm);i++) {
        timer.attach(&tick,1.0/mm[i]/2.0);
        wait(0.5f);
    }
    timer.detach();
    while(1);
}
```

### 18.2.3 応用

`float mm[]={mC*2,mD*2,mE*2,mF*2,mG*2,mA*2,mB*2,mC*4};`とすると、1 オクターブ上がる。  
高い音の方が聞こえ易いか確認する。

## 18.3 PWMによる音だし

### 18.3.1 接続

圧電ブザーを PWM 出力が使える p 25 に接続する。

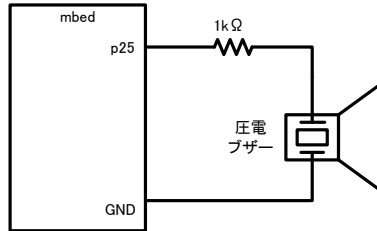


図 配線図

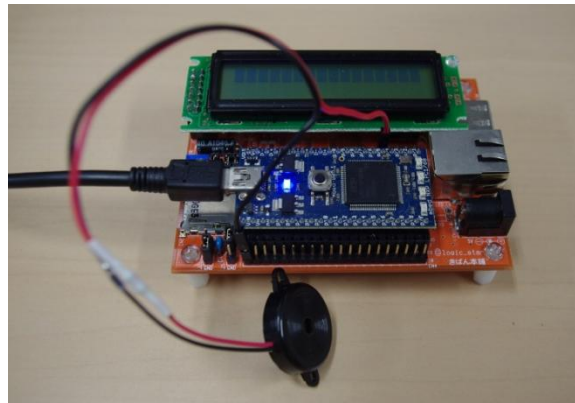


図 接続図

### 18.3.2 プログラム

これは単純に PWM の周期を変えるだけです。

```
#define mC 261.626
#define mD 293.665
#define mE 329.628
#define mF 349.228
#define mG 391.995
#define mA 440.000
#define mB 493.883

#include "mbed.h"

PwmOut sp1(p25);

int main() {
    float mm[]={mC,mD,mE,mF,mG,mA,mB,mC*2};
    int i;

    for (i=0;i<sizeof(mm);i++) {
        sp1.period(1.0/mm[i]);
        sp1.write(0.5f);
        wait(0.5f);
        sp1.write(0.0f);
    }
    while (1);
}
```

### 18.3.3 疑似 Sin 波の出力

これは圧電ブザーでは鳴りません。ヘッドフォンやスピーカを利用の事

```
#define mC 261.626
#define mD 293.665
#define mE 329.628
#define mF 349.228
#define mG 391.995
#define mA 440.000
#define mB 493.883

#include "mbed.h"

Ticker timer;
PwmOut sp1(p25);
float ms[180];
float m1;

void sound_out(void) {
    static float j=0;
    j=j+m1;
    if(j>180)j=j-180;
    sp1.write(ms[(int)j]);
}

int main() {
    float mm[]={mC,mD,mE,mF,mG,mA,mB,mC*2};
    int i;
    for (i=0;i<180;i++) {
        ms[i]=sin(2*3.1415*(float)i/180.0)/2.0+0.5;
    }

    sp1.period_us(10);
    timer.attach_us(&sound_out,100);

    for (i=0;i<sizeof(mm);i++) {
        m1=mm[i]*180/10000;
        wait(0.5f);
    }
    sp1.write(0.0f);
    while (1);
}
```

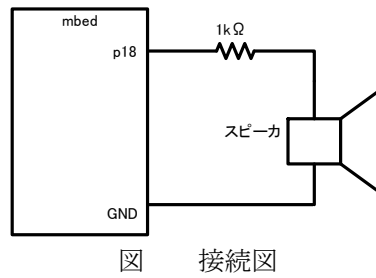
#### プログラムの説明

プログラムが始まると Sin 波のテーブル配列 **ms** に作ります。(今回は 2 度ずつのテーブル)  
そのあとは  $100 \mu\text{s}$  (10kHz) 毎に割り込みして、その時点の必要な出力を計算します。  
**mm[i]** には鳴らしている音の周期が入っており **m1** に  $100 \mu\text{S}$  (10kHz) で進む角度/2 を入れる。  
**sound\_out** ルーチンは、変数 **j** に角度/2 の値が入っており、呼ばれるごとに **m1** を加えて信号を出力する。

## 18.4 DACによる音だし

### 18.4.1 接続

DAC の出力は p18 のみです。



### 18.4.2 プログラム

```
#define mC 261.626
#define mD 293.665
#define mE 329.628
#define mF 349.228
#define mG 391.995
#define mA 440.000
#define mB 493.883

#include "mbed.h"

Ticker timer;
AnalogOut sp1(p18);
float ms[180];
float m1;

void sound_out(void) {
    static float j=0;
    j=j+m1;
    if(j>180)j=j-180;
    sp1.write(ms[(int)j]);
}

int main() {
    float mm[]={mC,mD,mE,mF,mG,mA,mB,mC*2};
    int i;
    //setting sincurv
    for (i=0;i<180;i++) {
        ms[i]=sin(2*3.1415*(float)i/180.0)/2.0+0.5;
    }

    timer.attach_us(&sound_out,100); //10kHz

    for (i=0;i<sizeof(mm);i++) {
        m1=mm[i]*180/10000;
        wait(0.5f);
    }
    sp1.write(0.0f);
    while (1);
}
```

PWM の出力を DAC に変更しただけです。

これもヘッドフォンもしくはスピーカーの方が聞きやすい

## 18.5 応用 和音による音だし

```
#define mC 261.626
#define mD 293.665
#define mE 329.628
#define mF 349.228
#define mG 391.995
#define mA 440.000
#define mB 493.883

#include "mbed.h"

Ticker timer;
AnalogOut sp1(p18);
float ms[180];
float m1,m2,m3;

void sound_out(void) {
    static float j1=0;
    static float j2=0;
    static float j3=0;
    j1=j1+m1;
    j2=j2+m2;
    j3=j3+m3;
    if (j1>180)j1=j1-180;
    if (j2>180)j2=j2-180;
    if (j3>180)j3=j3-180;
    sp1.write((ms[(int)j1]+ms[(int)j2]+ms[(int)j3])/3.0);
}

int main() {
    int i;
    //setting sincurv
    for (i=0;i<180;i++) {
        ms[i]=sin(2*3.1415*(float)i/180.0)/2.0+0.5;
    }
    timer.attach_us(&sound_out,100); //10kHz
    //ceg
    m1=mC*2*180/10000;
    m2=mE*2*180/10000;
    m3=mG*2*180/10000;
    wait(1.0f);
    //cfa
    m1=mC*2*180/10000;
    m2=mF*2*180/10000;
    m3=mA*2*180/10000;
    wait(1.0f);
    //ceg
    m1=mC*2*180/10000;
    m2=mE*2*180/10000;
    m3=mG*2*180/10000;
    wait(1.0f);
    //bdg
    m1=mB*180/10000;
    m2=mD*2*180/10000;
    m3=mG*2*180/10000;
    wait(1.0f);
    //ceg
    m1=mC*2*180/10000;
    m2=mE*2*180/10000;
    m3=mG*2*180/10000;
    wait(1.0f);
    timer.detach();
    sp1.write(0.0f);
    while (1);
}
```

## 18.6 応用

### 18.6.1 周波数を式で表す (Tedd OKANO さんより)

各音の周波数は等比数列で表される。したがって以下の様に初期化する事が出来る

```
#define    tone_A  440.0
float    tone_freq[ 12 ];

(以下プログラムの中で初期化)
for ( int i = 0; i < 12; i++)
    tone_freq[ i ] = pow( 2, ((float)(i + 3) / 12.0) - 1.0 ) * tone_A;
```

半音なしの場合は

```
#define    tone_A  440.0
float    doremi_freq[ 7 ]    = { 3, 5, 7, 8, 10, 12, 14 };

for ( int i = 0; i < 7; i++)
    doremi_freq[ i ] = pow( 2, (doremi_freq[ i ] / 12.0) - 1.0 ) * tone_A;
```

### 18.6.2 USB Audio

ハンドブック内の `USBAUDIO_speaker` も DAC から音を出力する。お試しあれ！  
プログラム中は“DAC で音だし”とあんまり変わりません。

<http://mbed.org/handbook/USBAudio>

template-typename T >  
void attach (T \*tptr, void(T::\*mptr)(void))  
Attach a nonstatic void/void member function to update the volume.

### More example

The following program is sending to a speaker all audio packets received. This means that you can play a music on your computer and listen it on your Mbed.

**USBAUDIO\_speaker** » Import this program

USBAudio speaker example  
Program published 4 週間 ago by Samuel Mokrani

### In details

#### Audio packet length

In this section, I will explain what kind of packets are received according to the frequency and the number of channels.  
An audio packet is received each millisecond. So let's say that a frequency of 48 kHz has been chosen with 2 channels (stereo).

図 USBAUDIO\_speaker のページ

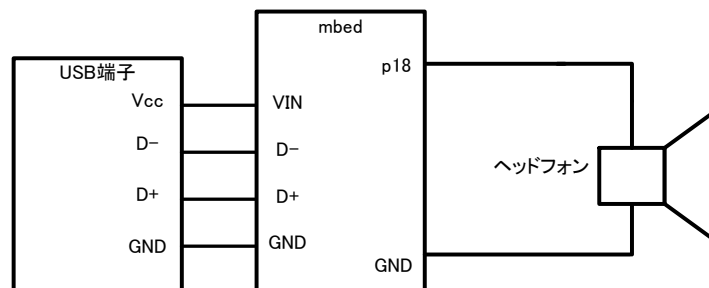
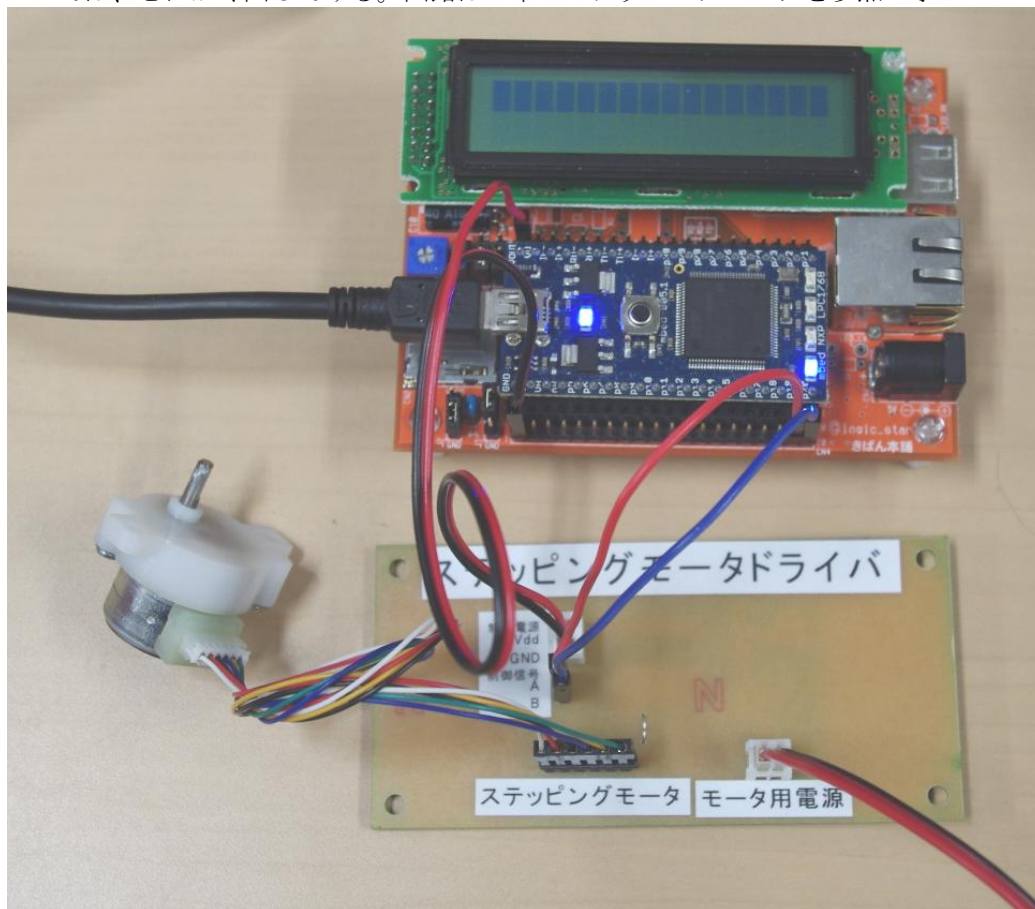


図 接続図

## 19 ステッピングモータの使用

### 19.1 ステッピングモータの HelloWorld

ここでは、とにかく回してみる。回路は1章のステッピングモータを参照の事





### 19.1.1 プログラム

```
#include "mbed.h"

DigitalOut myled(LED1);
DigitalOut step_a(p19);
DigitalOut step_b(p20);

Ticker timer1;

void add_step(void) {
    static int mode=0;
    switch (mode) {
        case 0 :
            step_b=0;
            step_a=1;
            mode=1;
            break;
        case 1 :
            step_a=1;
            step_b=1;
            mode=2;
            break;
        case 2:
            step_a=0;
            step_b=1;
            mode=3;
            break;
        case 3:
            step_a=0;
            step_b=0;
            mode=0;
            break;
    }
    myled=!myled;
}

int main() {
    timer1.attach(&add_step, 0.01);
    while (1);
}
```

タイマーを使って定期的に `add_step` を呼び出すことにより、モータ回す。

関数 `add_step` では変数 `mode` を `static` 割り当てし保存することで次のステップを動かす。

もちろん `main` 関数内の `timer1.attach(&add_step, 0.01);` の `0.01` を小さくすれば早く、大きくすれば遅くなる。ただし、あまり早くすると、モータが付いて行けず回らなくなる。これが脱調である。

## 19.2 角度を指定して回す

ここでは角度を指定して回す。使うステッピングモータはちょうど、1パルス1度なので、パルス数を回転角度が一致する。

### 19.2.1 プログラム

```
#include "mbed.h"

DigitalOut myled(LED1);
DigitalOut step_a(p19);
DigitalOut step_b(p20);

Ticker timer1;

int mode=0;

void step(int x) {
    int i;
    for (i=0;i<x;i++) {
        switch (mode) {
            case 0 :
                step_a=1;
                step_b=0;
                mode=1;
                break;
            case 1 :
                step_a=1;
                step_b=1;
                mode=2;
                break;
            case 2:
                step_a=0;
                step_b=1;
                mode=3;
                break;
            case 3:
                step_a=0;
                step_b=0;
                mode=0;
                break;
        }
        myled=!myled;
        wait(0.01f);
    }
}

int main() {
    int i;
    for (i=0;i<12;i++) {
        step(30);
        wait(0.5f);
    }
}
```

ポイントは、現在のステッピングモータのステップ No(このプログラムでは変数 `mode`)を記憶させておくこと。でなければ、少しずつずれる。ちなみにこのプログラムでは逆方向に回らない。

### 19.2.2 応用

正転逆転できるプログラムを作る

## 20 参考

参考になるホームページ

- NPX fan(in Japan) <http://mbed.org/users/nxpfan/>
- MBED を 256 倍使うための頁 <http://mbed.org/users/okini3939/notebook/mbed256/>

参考になる豆知識

- mbed の FLASH メモリは FORMAT で、すべて消しても”mbed.htm”はリセット後復旧します。