山口大学工学部
YAMAGUCHI UNIVERSITY
FACULTY OF ENGINEERING

平成 25 年度情報技術研究会演習

mbed を利用した IEEE1888 センサーノード作成

（於：平成 26 年 3 月 18 日　九州工業大学情報工学部）

田内　康

参考資料は、　http://mbed.org/users/yueee_yt/notebook/giken9　にあります。

内容

1 mbed LPC1768 の概要 1

1.1 ☆Board Orange 2

2 mbed を使うための C++言語の知識 3

2.1 コンパイラのドキュメントの場所 3

2.2 C Data Type 3

2.3 使用するための準備（ログイン ID 発行） 4

2.4 mbed.org のホームページ構成 5

3 Hello World！ 5

3.1 ログイン（まだログインしていなければ） 5

3.2 コンパイルと実行形式のファイルの保存 5

4 温度計の作成（LCD の温度表示） 9

4.1 プログラムの作成 9

4.1.1 テンプレートプログラムの読み込み 9

4.1.2 RTOS ライブラリの読み込み 10

4.1.3 プログラムの作成 12

4.1.4 ハードの接続 13

4.1.5 備考 13

5 Web サーバ１ 14

5.1 mbed のプログラム 14

5.2 Web プログラム(test.htm) 15

5.3 使い方 15

6 Web サーバ２温度計測 16

6.1 mbed のプログラム 16

6.2 Web プログラム(temp.htm) 17

7 IEEE1888WriteClient 18

7.1 概要 18

7.2 mbed のプログラム 18

1　mbed　LPC1768 の概要

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。

ハードウエ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－を接続する。

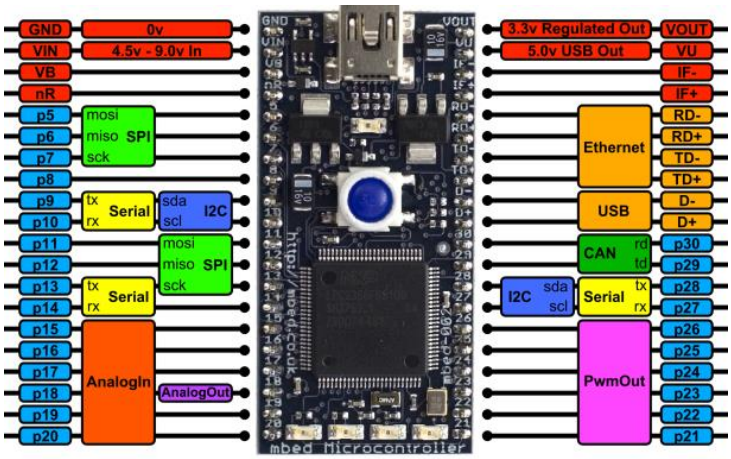


図　mbed のピン配置

1.1　☆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。本書では、☆Board Orange を使用するので、ピン配置はそれに従う。詳しくは下記ホームページをご覧の事。USB を利用する場合は電源が不足するので 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/

表　☆Board Orange で使用されている Pin

(オレンジ色は☆Board Orange で使用されている)

ピン	備考
GND	GND
VIN	アダプタ
VB	バッテリ
nR	リセット
5	SD
6	SD
7	SD
8	SD
9	DIO, I2C sda
10	DIO, I2C scl
11	DIO, SPI mosi
12	DIO, SPI miso
13	DIO, Serial TX/SPI sck
14	DIO, Serial RX
15	DIO, AnalogIn
16	DIO, AnalogIn
17	DIO, AnalogIn
18	DIO, AnalogIn/Out
19	DIO, AnalogIn
20	DIO, AnalogIn

ピン	備考
VOUT	3.3V
VU	USB の電源
IF-	利用不可
IF+	利用不可
RD-	LAN
RD+	LAN
TD-	LAN
TD+	LAN
D-	USB
D+	USB
30	LCD
29	LCD
28	LCD
27	LCD
26	LCD
25	LCD
24	LCD
23	DIO, PWM
22	DIO, PWM
21	DIO, PWM

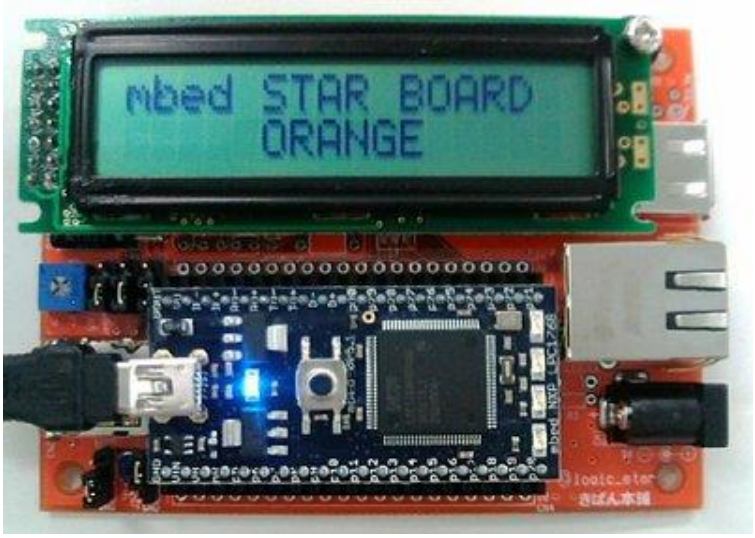


図 1.17　☆Board Orange

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

mbed では以下のデータタイプが使用できる。（http://mbed.org/handbook/C-Data-Types より引用）

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
char	uint8_t	8	Unsigned	0 .. 255
signed char	int8_t	8	Signed	-128 .. 127
unsigned short	uint16_t	16	Unsigned	0 .. 65,535
short	int16_t	16	Signed	-32,768 .. 32,767
unsigned int	uint32_t	32	Unsigned	0 .. 4,294,967,295
int	int32_t	32	Signed	-2,147,483,648 .. 2,147,483,647
unsigned long	long uint64_t	64	Unsigned	0 .. 18,446,744,073,709,551,615
long long	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
float	Single Precision	32	-3.4E38 .. 3.4E38
double	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!

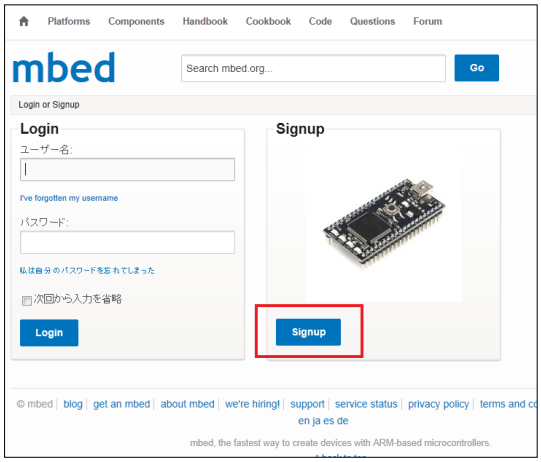
In code ported from other platforms, especially 8-bit or 16-bit platforms, the data types may have had different sizes. For example, int may have been represented as 16-bits. If this size has been relied on, some of the code may need updating to make it more portable. In addition, it is quite common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

2.3　使用するための準備（ログイン ID 発行）

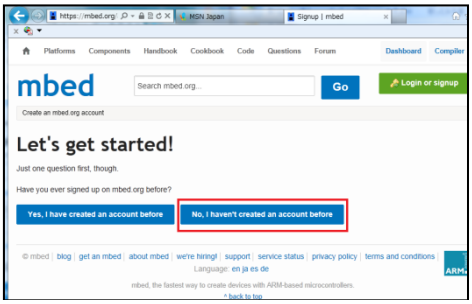
PC の USB 端子に mbed を接続すると通常の USB フラッシュメモリとして PC に認識される。その中の MBED.HTM をダブルクリックし、mbed のホームページに接続すると以下の画面が出てくる。

※MBED.HTM は FORMAT しても、リセットすると自動で作成される。

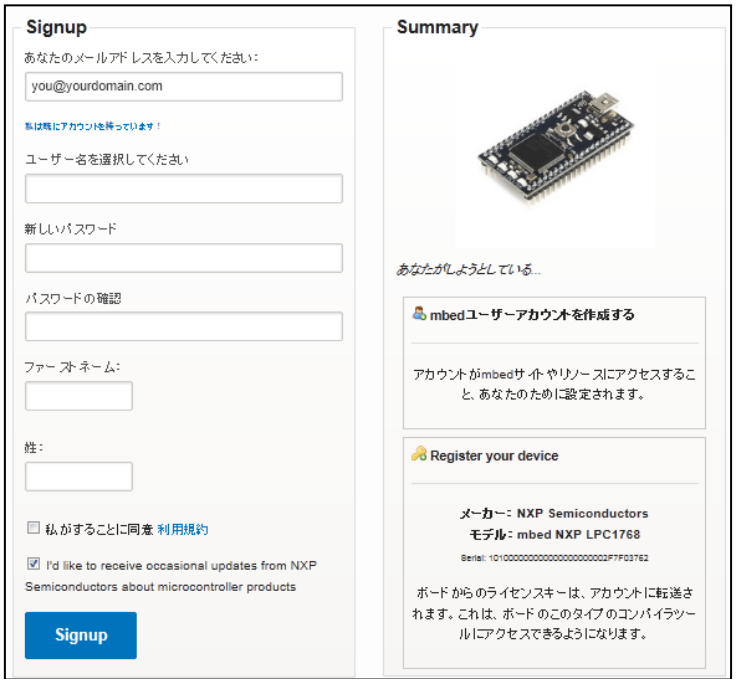
ユーザ名を持っていない方は Signup ボタンを押します。（他の種類の mbed しか利用したことのない人も）



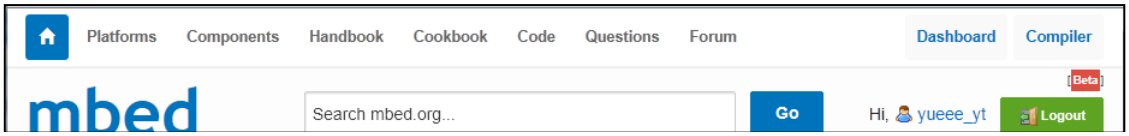
No, I haven't created an account before ボタンを押す。



あと、詳細情報を入力する。（ユーザ名、新しいパスワードは次回ログインに必要になるので記録する。）



2.4　mbed.org のホームページ構成



mbed のホームページは Platforms、Component、Handbook、Cookbook、Code、Questions、Forum、Dashboard、Compiler から成る。Handbook、Cookbook、Forum、Compiler の項目は以前からある項目で、他は最近追加された。内容が被っている部分（たとえば Component と Cookbook など）は、その名残と思われる。

Platforms ---------- mbed の種類（もともと mbed 出ないものを mbed 化するための情報など）
Component ---------- 目的別、使うセンサーなどの情報
Handbook ---------- mbed の Official な情報
Cookbook ---------- ユーザが作成したものを集めた情報
Code ---------- 公開されているプログラム
Questtions ---------- 質問
Forum ---------- フォーラム・掲示板機能
Dashboard ---------- ダッシュボード
Compiler ---------- コンパイラ画面に移行

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（まだログインしていなければ）

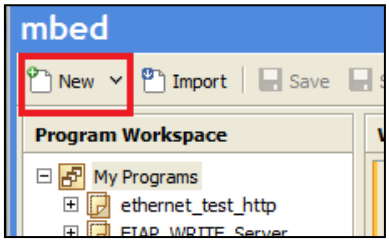
（1）http://mbed.org/　に接続して Login or signup をクリックする。

（2）ユーザ名とパスワードを入力後、Login ボタンを押す。

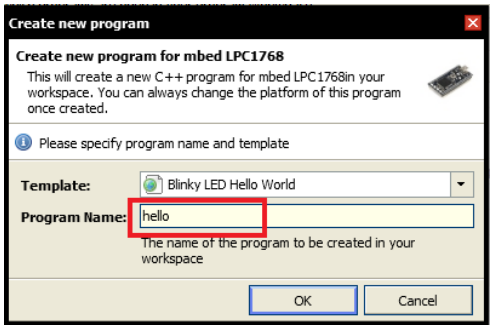
3.2　コンパイルと実行形式のファイルの保存

（1）コンパイラ画面へ移るために Compiler をクリックする。

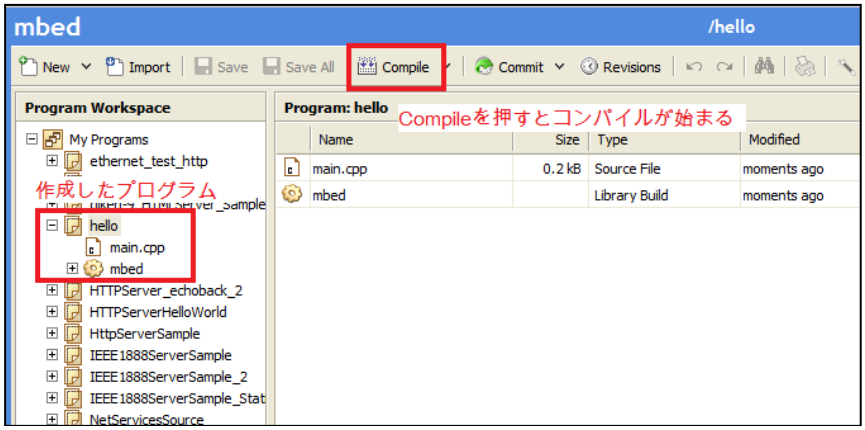
（2）新しいプログラムを作成するので、New をクリックする。



（3）プログラム名を入力して OK ボタンを押す。（template に Blinky LED hello World を選択）



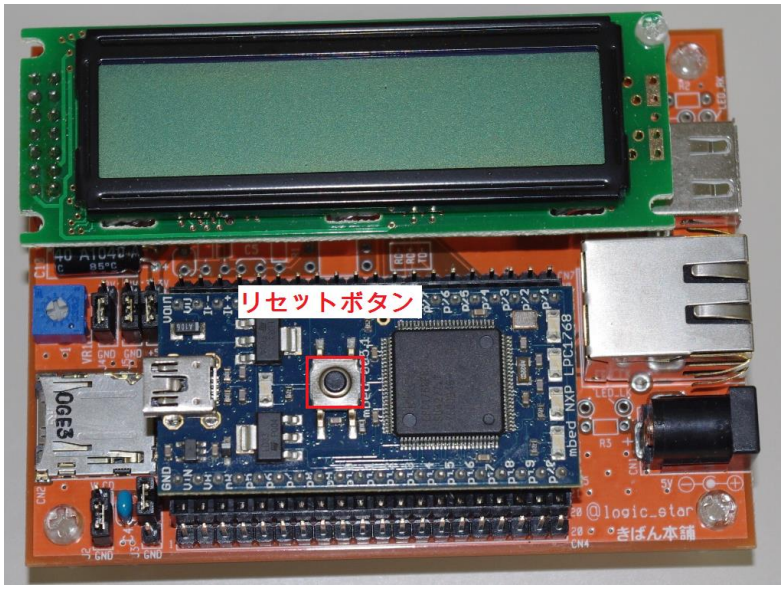
（4）Compile ボタンをおしてコンパイルすると、コンパイルが始まり、終わるとファイルの保存を聞いてくるので、mbed のローカルファイルシステムに保存する。



（5）実行

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。



（６）プログラムの確認と修正

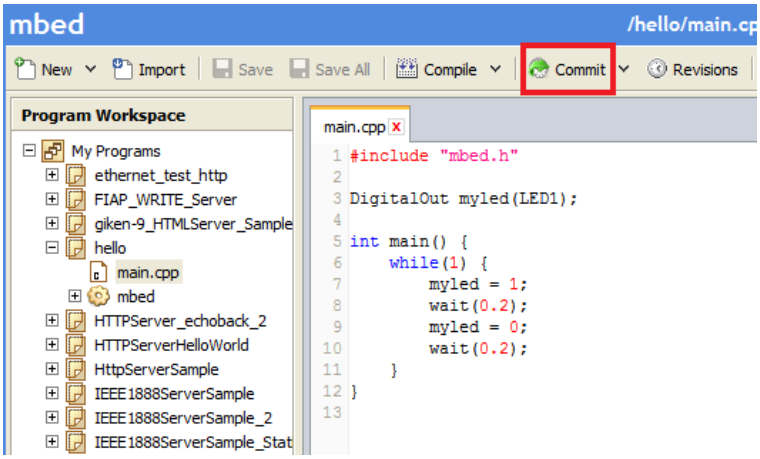
もう少し、ゆっくり点滅させる。

今回作成されたプログラム main.cpp をみると、以下のようになっている。

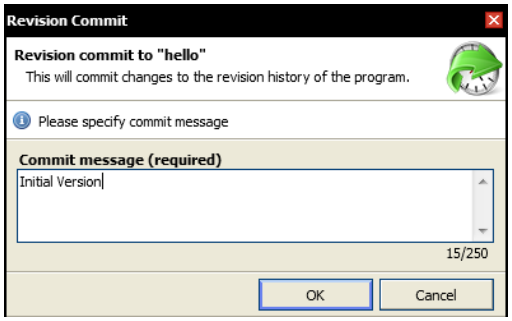
```
#include "mbed.h"

DigitalOut myled(LED1);

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

mbed コンパイラにはリビジョン管理機能もあり、利用すると簡単に前の状態にもどせる。

commit ボタンを押す。



Commit　Message を書き、OK を押します。

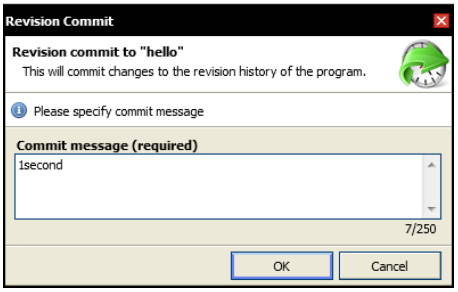


wait 文を 0.2 から 1.0 に修正します。

```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1;
        wait(1.0);
        myled = 0;
        wait(1.0);
    }
}
```

Compile します。mbed に転送し、実行します。

上手く動けば、再度 Commit します。



前の状態に戻すには Revision ボタンを押し、前の状態に Change する。

4　温度計の作成（LCD の温度表示）

ここでは、温度センサーを接続して温度計を作成する。

mbed なので RTOS（Real Time OS）が利用できるので使用して作ってみる。

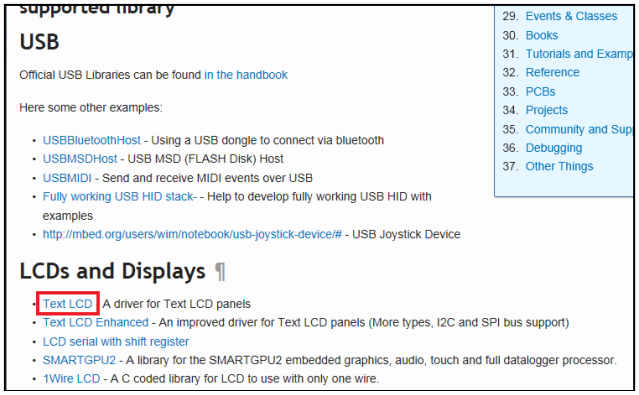
温度の計測は 2 秒間隔、表示は 3 秒間隔に行う。

mbed のプログラム作成方法は、今から作成したい事の参考になるようなプログラムから、発展し、完成させるスタイルである。ここでは TextLCD のプログラムを利用させて頂き、そこから、発展させてプログラムを作成する。

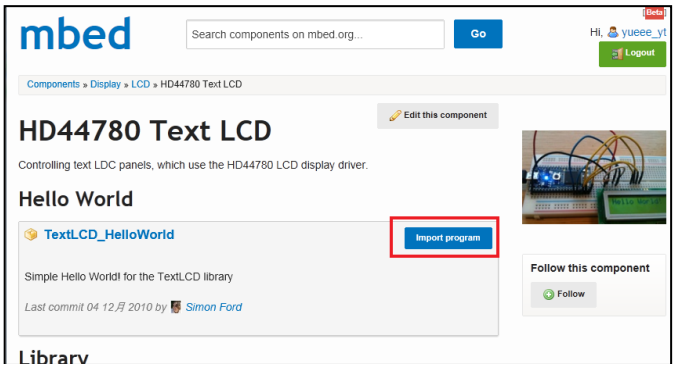
4.1　プログラムの作成

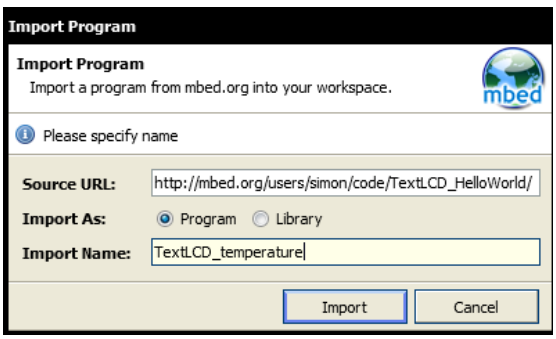
4.1.1　テンプレートプログラムの読み込み

CookBook の LCDs and Displays にある TextLCD をクリックする。



Import Program ボタンを押して、プログラムをインポートする。ここでは、インポート先のプログラム名を“TextLCD_temperature”とする。





main.cpp のプログラムで LCD のピンを指定している部分を Star Board Orange に合うように変更する。(赤字が修正箇所)

Star Board Orange の情報も Cook Book にある。

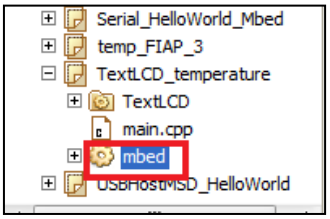
```
// Hello World! for the TextLCD

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

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

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

ここまでで、きちんと表示されるかコンパイル＆実行してみる。

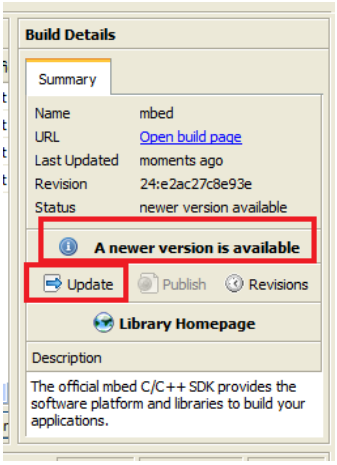
4.1.2 RTOS ライブラリの読み込み

RTOS を読み込む前に mbed のライブラリのバージョンを確認します。古いサンプルプログラムをインポートすると古い時点のライブラリを一緒に読み込みます。RTOS は、比較的新しく、適宜更新されているので、mbed ライブラリも最新版の方が良い。※ただし、逆に新しいライブラリだと動かないものもありますので注意が必要。

Program Workspace 内のインポート先のプログラムの mbed をクリックします。

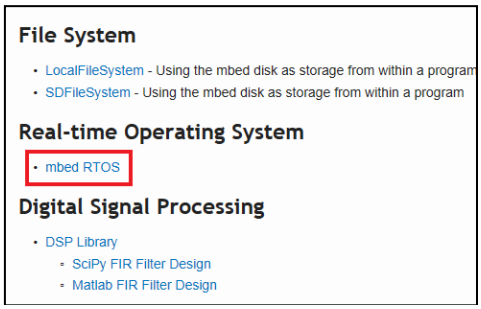


ブラウザの右側に情報が出ます。新しいバージョンの物があれば Update します。

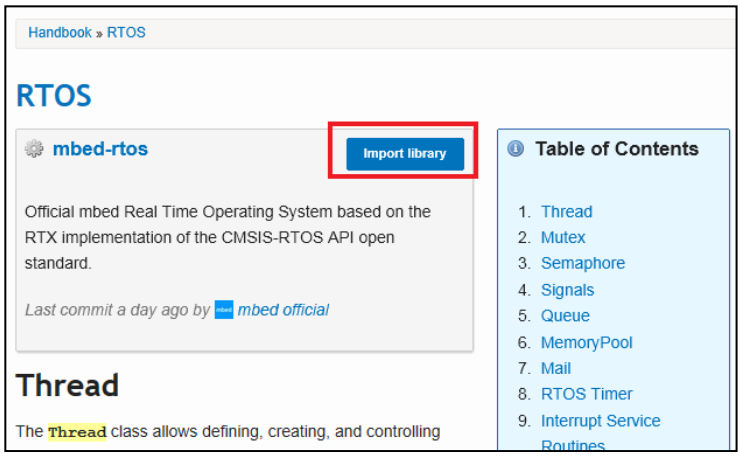


RTOS のライブラリをインポートします。コンパイラ画面からも Import 出来ますが、ここでは、Handbook から Import します。

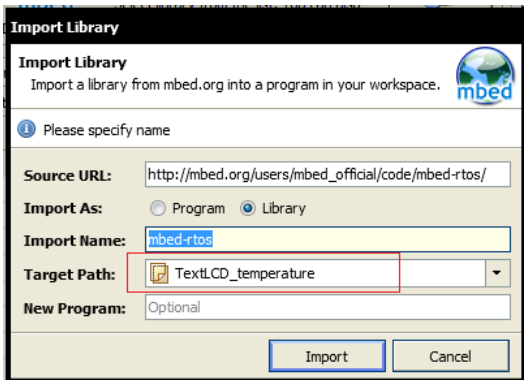
Handbook 内の mbed RTOS をクリックします。



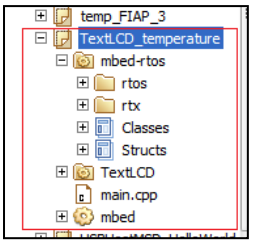
ライブラリをインポートしたいので　Import library をクリックします。



ターゲットが今回 Import するプログラム（ここでは"TextLCD_temperature"）になっているか確認し、Import ボタンをクリックする。



この様なプログラム構成になっているか確認する。



4.1.3　プログラムの作成

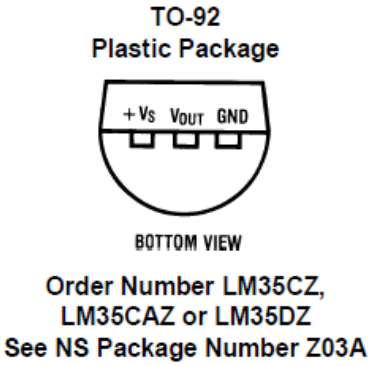
main.cpp のプログラムを修正します。今回使用する温度センサ LM35D は 10mV/℃なので電圧を 100 倍すると温度になる。

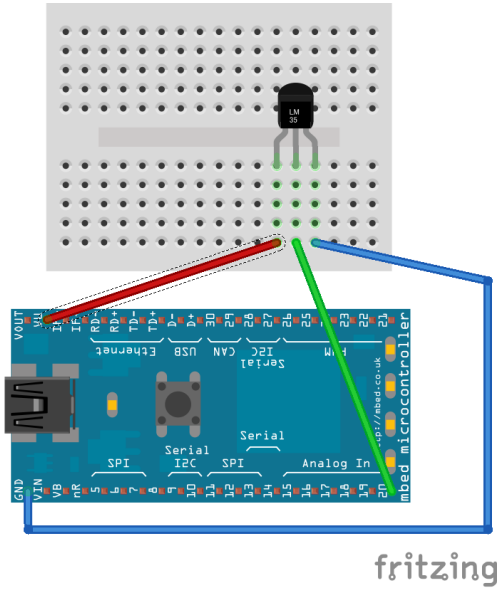
```
#include "mbed.h"
#include "TextLCD.h"
#include "rtos.h"

TextLCD lcd(p24, p26, p27, p28, p29, p30); // rs, e, d4-d7
AnalogIn ain(p20);
DigitalOut led(LED1);

float Temp;

void display(void const *n)
{
    static int num=0;
    lcd.locate(0,0);
    lcd.printf("Count %3d",num);
    lcd.locate(0,1);
    lcd.printf("Temp=%4.1f",Temp);
    num++;
    if(num>999)num=0;
}

void measure(void const *n)
{
    Temp=ain.read()*3.3*100;   読んだ値(0-1)に 3.3 を掛け電圧に変換、さらに 100 を掛け温度に
    led=!led;
}

int main()
{
    led=0;
    lcd.cls();
    RtosTimer measure_timer(measure);
    RtosTimer display_timer(display);
    measure_timer.start(2000);   計測は 2 秒毎に
    display_timer.start(3000);   表示は 3 秒毎に
    Thread::wait(osWaitForever);
}
```

4.1.4　ハードの接続

ここでは LM35D を使用する。極性を間違えないように接続してください。

LM35D の電源は 4～30V なので、mbed の電源電圧(3.3V)では不足する。ここでは、USB の出力電圧（VU）を使用する。





4.1.5　備考

ロガーを作りたいとき（保存をするとき）は Local ファイルシステムでなく SD 等使用することをお勧めします。mbed が Local ファイルシステムにアクセスするときは一旦 PC から切り離されるので、それに伴う弊害が起こるからである。

5　Web サーバ 1

今回使用する mbed LPC1768 は Ethernet 機能を有し、簡単に Web サーバを構成することも可能である。ここでは、mbed には RPC やローカルファイルシステムの仕組みを利用して Web サーバを構築する例を提示する。ローカルファイルシステムは 7.3 ファイル形式しか使えないので、○○.htm という表記になる。

RPC を使用することで、mbed のプログラムと HTML が分離した状態に出来、わかりやすくなる。

5.1　mbed のプログラム

(※このプログラム(ライブラリ)はまだ作成途中の例で、バグがあります。)

```
#include "mbed.h"
#include "rtos.h"
#include "EthernetInterface.h"
#include "HTTPServer.h"
#include "mbed_rpc.h"
#include "TextLCD.h"

EthernetInterface eth;
LocalFileSystem local("local");
DigitalOut led4(LED4);
//DigitalOut led1(LED1);
TextLCD lcd(p24, p26, p27, p28, p29, p30);

void LcdWrite(Arguments* arg, Reply* r);void LcdWrite(Arguments* arg, Reply* r);
void aliveState(void const *args)
{
    while (true) {
        led4 = !led4;
        Thread::wait(1000);
    }
}
int main()
{
    printf("********** PROGRAM START ***********¥r¥n");
    Thread thread(aliveState);
    lcd.cls();
    lcd.locate(0,0);

    printf("********** RPC Initialize ***********¥r¥n");
    RPC::add_rpc_class<RpcDigitalOut>();
    RPC::construct<RpcDigitalOut, PinName, const char*>(LED1, "led1");
    RPC::construct<RpcDigitalOut, PinName, const char*>(LED2, "led2");
    RPC::construct<RpcDigitalOut, PinName, const char*>(LED3, "led3");
    RPCFunction rpcFunc(LcdWrite, "LcdWrite");

    printf("EthernetInterface Setting up...¥r¥n");
    if(eth.init()!=0) {                                    //for DHCP Server
      // if(eth.init("133.11.168.23","255.255.255.0","133.11.168.1")!=0) { //for Static IP Address
        printf("EthernetInterface Initialize Error ¥r¥n");
        return -1;
    }
    if(eth.connect()!=0) {
        printf("EthernetInterface Connect Error ¥r¥n");
        return -1;
    }
     printf("IP Address is %s¥r¥n", eth.getIPAddress());
     printf("NetMask is %s¥r¥n", eth.getNetworkMask());
     printf("Gateway Address is %s¥r¥n", eth.getGateway());
     printf("Ethernet Setup OK¥r¥n");
```

```
    FSHandler::mount("/local","/");

    lcd.locate(0,0);
    lcd.printf("%s",eth.getIPAddress());
    HTTPServerStart(80);
}

void LcdWrite(Arguments* arg, Reply* r)   //RPC Call
{
    lcd.locate(0,1);
    lcd.printf("%s",arg->argv[0]);
}
```

5.2　Web プログラム(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);
lcd = new RPCFunction(mbed,"LcdWrite");

</script>
</head>
<body>
<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 >
<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 >
<br>

<input type="text" id="textbox" ></input><button onclick="lcd.run(textbox.value);">送信</button> <br>

</body>
</html>
```

5.3　使い方

コンパイルして、LAN ケーブルを接続後、実行(リセット)することで、動き始めます。

http://IP アドレス/hello

http://IP アドレス/sample.htm

http://IP アドレス/test.htm　　下記の TextBOX に英数字を入れて送信すると TextLCD に表示できる。

などが実行可能です。また、RPC も有効にしているので

http://IP アドレス/rpc/led1/write 1　を実行して LED 1 を点灯したり、プログラムを実行したりできます。

6　Web サーバ２温度計測

温度表示の Web サイトも、rpc を利用すると簡単にできます。

6.1　mbed のプログラム

```
#include "mbed.h"
#include "rtos.h"
#include "EthernetInterface.h"
#include "HTTPServer.h"
#include "mbed_rpc.h"
#include "TextLCD.h"

EthernetInterface eth;
LocalFileSystem local("local");
DigitalOut led4(LED4);

DigitalOut led1(LED1);
AnalogIn ain(p20);

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

float Temp;

void LcdWrite(Arguments* arg, Reply* r);void LcdWrite(Arguments* arg, Reply* r);
void aliveState(void const *args)
{
    while (true) {
        led4 = !led4;
        Thread::wait(1000);
    }
}

void measure(void const *n)
{
    Temp=ain.read()*3.3*100;
    led1=!led1;
}

int main()
{
    printf("********** PROGRAM START **********¥r¥n");
    Thread thread(aliveState);

    led1=0;
    RtosTimer measure_timer(measure);
    measure_timer.start(2000);

    lcd.cls();
    lcd.locate(0,0);

    printf("********** RPC Initialize **********¥r¥n");
    RPC::add_rpc_class<RpcDigitalOut>();
    RPC::construct<RpcDigitalOut, PinName, const char*>(LED1, "led1");
    RPC::construct<RpcDigitalOut, PinName, const char*>(LED2, "led2");
    RPC::construct<RpcDigitalOut, PinName, const char*>(LED3, "led3");
    RPCFunction rpcFunc(LcdWrite, "LcdWrite");
    RPCVariable <float>rpcval(&Temp,"temp");
    printf("EthernetInterface Setting up...¥r¥n");
    if(eth.init()!=0) {                                    //for DHCP Server
      // if(eth.init("133.11.168.23","255.255.255.0","133.11.168.1")!=0) { //for Static IP Address
        printf("EthernetInterface Initialize Error ¥r¥n");
        return -1;
    }
```

```
    if(eth.connect()!=0) {
        printf("EthernetInterface Connect Error ¥r¥n");
        return -1;
    }
    printf("IP Address is %s¥r¥n", eth.getIPAddress());
    printf("NetMask is %s¥r¥n", eth.getNetworkMask());
    printf("Gateway Address is %s¥r¥n", eth.getGateway());
    printf("Ethernet Setup OK¥r¥n");

    FSHandler::mount("/local","/");

    lcd.locate(0,0);
    lcd.printf("%s",eth.getIPAddress());
    HTTPServerStart(80);
}

void LcdWrite(Arguments* arg, Reply* r)    //RPC Call
{
    lcd.locate(0,1);
    lcd.printf("%s",arg->argv[0]);
}
```

6.2　Web プログラム(temp.htm)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-2022-jp">
<title>mbed temp 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);
        lcd = new RPCFunction(mbed,"LcdWrite");
        vtemp = new RPCVariable(mbed, "temp");
</script>
</head>
<body>
<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 >
<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 >
<br>

<input type="text" id="textbox" ></input><button onclick="lcd.run(textbox.value);">送信</button> <br>
<br/>
今の温度は
<script type="text/javascript">
document.open();
        temp=new Number(vtemp.read());
        document.write(temp.toFixed(2));
        document.close();
</script>
度です。
</body>
</html>
```

7　IEEE1888WriteClient

7.1　概要

IEEE1888 は、正式名称が UGCCNet (Ubiquitous Green Community Control Network)で、東大グリーン ICT プロジェクトで作成され・中国が提案したプロトコルである。

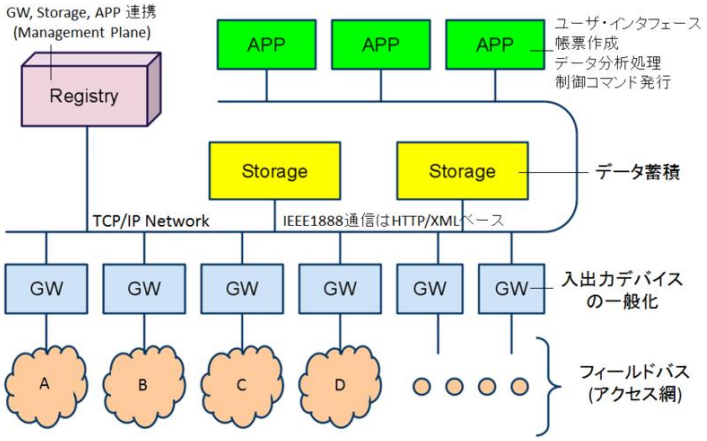
通信は XML を用いて行い、現在は TCP/IP 上の SOAP を利用しておこなう。

データは、場所を示す PointID、時刻、値で示す。PointID は URI を使用した名前空間で、実存する必要はないが、大きなシステムが構築される前提なので、

http://アドレス/建物/部屋番号/センサ名/センサ種別

の様な感じで、唯一の ID になるように指定する。

下記図は IEEE1888 のイメージ図である。GW、APP、Storage は機能によって呼び方が変わるだけで、インターフェイスは同じものを備える（不要なものはないかもしれない）。



FIAP　Storage やプログラム作成時の参照プログラムは、東大グリーン ICT プロジェクトのホームページ(http://gutp.jp/)からダウンロードできる。

7.2　mbed のプログラム

利用するためには、保存先(送信先)のアドレス、PointID などをあらかじめ決めておく必要がある。また、このプログラムでは NTP サーバから時刻を取得しているの、NTP サーバのアドレスが必要になる。

このサンプルプログラムでは、

NTP サーバアドレス　：　ntp.nict.jp

データの保存先　：　http://fiap-sandbox.gutp.ic.i.u-tokyo.ac.jp/axis2/services/FIAPStorage

PointID　：　http://giken9.jp/no_0/temp

としている。これらは必要に応じて変更の必要がある。

保存された内容は、http://fiap-sandbox.gutp.ic.i.u-tokyo.ac.jp/にアクセスすることで参照できる。

なお、このプログラムでは起動時のみ NTP サーバから時刻を取得しているので、長時間動かすプログラムを作成する場合は、このあたりの修正が必要となる。

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

#define NTPServer "ntp.nict.jp"

EthernetInterface eth;
NTPClient ntp;
Ticker timer1;
time_t ctTime;

TextLCD lcd(p24, p26, p27, p28, p29, p30);
AnalogIn ain(p20);
DigitalOut led(LED1);
char timezone[] = "+09:00";  // JST
char atemp[6];
FIAP fiap("http://fiap-sandbox.gutp.ic.i.u-tokyo.ac.jp/axis2/services/FIAPStorage");
struct fiap_element element[]= {
    {"http://giken9.jp/no_0/temp",atemp,NULL,NULL,NULL,NULL,NULL,NULL,timezone},
};


void tick(void )
{
    float temp;
    char buffer[9];
    led=!led;
    temp=ain*3.3*100.0;
    ctTime = time(NULL);
    strftime(buffer,9,"%X",localtime(&ctTime));
    lcd.locate(0,1);
    lcd.printf("%s %4.1fDeg",buffer,temp);
// Save to FIAPStorage
    sprintf(atemp,"%4.1f",temp);
    struct tm t = *localtime(&ctTime);
    element[0].value=atemp;
    element[0].year=t.tm_year+1900;
    element[0].month=t.tm_mon+1;
    element[0].day=t.tm_mday;
    element[0].hour=t.tm_hour;
    element[0].minute=t.tm_min;
    element[0].second=t.tm_sec;
    fiap.post(element,1);
}

int main()
{
//Ethernet Initialize
    eth.init(); //Use DHCP
    eth.connect();
    lcd.cls();
    lcd.locate(0,0);
    lcd.printf("%s", eth.getIPAddress());
    printf("Trying to update time...¥r¥n");
    if (ntp.setTime(NTPServer) == 0) {
        printf("Set time successfully¥r¥n");
```

```
        time_t ctTime;
        ctTime = time(NULL);
        ctTime+=32400;      UTC を JST に変換
        set_time(ctTime);
        ctTime = time(NULL);
        printf("Time is set to (JST): %s¥r¥n", ctime(&ctTime));
        printf("finish ¥n");
    } else {
        lcd.locate(0,1);
        lcd.printf("Error");
        return -1;
    }
    //fiap.debug_mode=true;
    //eth.disconnect();
    while(true) {
        tick();
        wait(60);       60 秒毎に実行
    }
}
```