

実習 2 mbed マイコンによる計測システムの構築

平成 24 年度 中国・四国地区国立大学法人等技術職員研修

8月30日 鳥取大学

1	はじめに.....	1
1.1	mbed マイコン.....	1
1.2	☆Board Orange.....	1
1.3	mbed の端子.....	2
2	mbed の使い方.....	3
2.1	PC の準備.....	3
2.2	（省略）ユーザの登録.....	3
2.3	ホームページのログイン.....	3
2.4	開発環境のページに行く.....	4
2.5	プログラムの登録.....	4
2.6	実行.....	2
2.7	デバッグの方法.....	3
2.7.1	mbed ドライバのインストール.....	3
2.7.2	TeraTerm の起動.....	5
2.7.3	プログラムの起動.....	5
3	簡単な例題.....	6
3.1	スイッチの動作.....	6
3.2	外部デジタル出力.....	6
3.3	カウンタの作成.....	7
3.4	圧電ブザー.....	8
4	計測システムの作成.....	9
4.1	概要.....	9
4.2	ハードウェア.....	9
4.3	準備.....	10
4.4	LCD の表示.....	11
4.5	Ethernet ライブラリの導入.....	12
4.6	NTP のプログラムの作成.....	14
4.7	温度計測し、USB フラッシュメモリに保存する部分を作成.....	17
4.8	Web サーバの構築.....	22
5	さらに大規模なデータ収集システムの構築.....	27
5.1	IEEE1888 でのデータ収集プログラム（Write 手順）.....	28
5.2	FIAP Storage のデータ表示（FETCH 手順）.....	30
5.2.1	課題.....	30

6	さらに.....	31
6.1	m3pi.....	31
6.1.1	m3pi の Hello World	32
6.1.2	LED の表示	33
6.1.3	ライントレースロボット	34
6.1.4	LineMaze	34
6.2	USB.....	41
6.2.1	USB マウスエミュレート.....	41
6.2.2	USB キーボードエミュレート	43
6.2.3	USB オーディオ	43
7	参考	43

1 はじめに

1.1 mbed マイコン

mbed は nxp セミコンダクター社の製品で、現在は 2 つのハードウェアが販売されている。この実習では mbed LPC1768 を利用する。

- 高性能 mbed LPC1768 ・ ・ Cortex-M3 98MHz

LPC1768FBD100 搭載 (100MHz, Flash : 512KB, RAM : 64KB)

Flash : 512KB, RAM : 32KB

Ethernet, USB(Host/Device), CAN など

消費電流 : 60-120mA

- 低消費電力 mbed LPC11U24 ・ ・ Cortex-M0 48MHz

LPC11U24FBD64 搭載 (50MHz, Flash : 32KB, RAM : 10KB)

Flash : 32KB, RAM : 8KB

USB Device, など

消費電流 : 1-16mA

開発はホームページ(<http://mbed.org/>)上で行うので、Windows, MAC, Unix など種類を選びません。また、ホームページ上には、HandBook(オフィシャルな内容)とCookBook(Userの開発した便利な機能など)が公開されているので、それらを活用し開発することが可能である。また、情報の共有を行いながら開発したり、また、世界中に公開したりすることも可能である。

スイッチサイエンス、秋月電子通商などで購入できます。

1.2 ☆Board Orange

☆Board Orange は日本の有志で開発された mbed を簡単に試することができるベースボードです。mbed ホームページ上には国内外の他のベースボードも紹介されていますが、国内では一番購入しやすいです。きばん本舗(キット販売もあり)、スイッチサイエンスなどで購入できます。

利用方法

電力をあまり必要としないときは PC (USB) から mbed 経由で供給します。USB で電流を必要とする機器を接続するときは、DC 5V のアダプタを利用する必要があります。

1.3 mbed の端子

☆Board Orange で利用するピンと mbed LPC1768 のピンをまとめると表 1.1 になる。

表 1.1 m b e d を ☆Board Orange で使用した際のピン

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(RTC 保存用)	IF-	利用不可
nR	リセット	IF+	利用不可
5	SD(SD_DI)	RD-	LAN
6	SD(SD_DO)	RD+	LAN
7	SD(SD_CK)	TD-	LAN
8	SD(SD_CS)	TD+	LAN
9	DIO, I2C sda	D-	USB
10	DIO, I2C scl	D+	USB
11	DIO, SPI mosi	30	LCD(DB7)
12	DIO, SPI miso	29	LCD(DB6)
13	DIO, Serial TX/SPI sck	28	LCD(DB5)
14	DIO, Serial RX	27	LCD(DB4)
15	DIO, AnalogIn	26	LCD(E)
16	DIO, AnalogIn	25	LCD(RW)
17	DIO, AnalogIn	24	LCD(RS)
18	DIO, AnalogIn/Out	23	DIO, PWM
19	DIO, AnalogIn	22	DIO, PWM
20	DIO, AnalogIn	21	DIO, PWM

の部分 は ☆Board Orange で使用している。

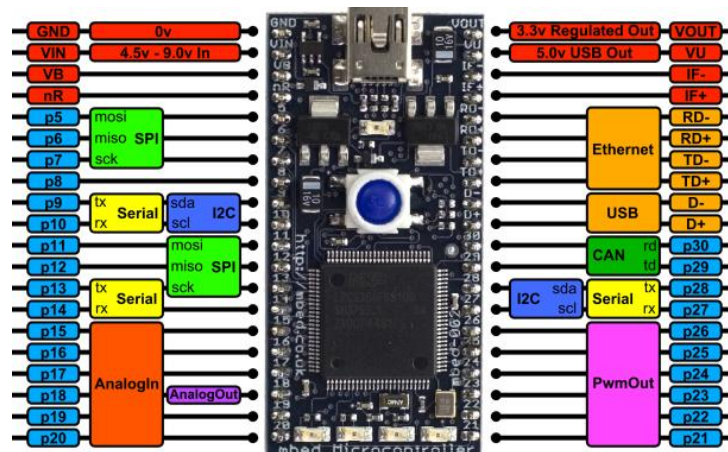


図 1.1 mbed のピン配置

2 mbed の使い方

2.1 PC の準備

- (1) ネットワーク接続できる環境にする。
- (2) mbed を PC に USB ケーブルで接続する。PC からは USB フラッシュメモリに見える。

※USB3.0 のポートの場合は上手くいかない事例の報告がある。その時は短い USB ケーブルを利用すると良い。

2.2 (省略) ユーザの登録

研修用の mbed はすでに登録済みなのでこのステップは省略する。

mbed を購入し、最初に使用する場合は mbed 内の MBED.HTM をダブルクリックすることで登録用のホームページにとぶ。mbed のフラッシュメモリを Format しても、USB を再接続することで、MBED.HTM は自動生成される。

2.3 ホームページのログイン

<http://mbed.org/>に接続し、Login or singup をクリックする。

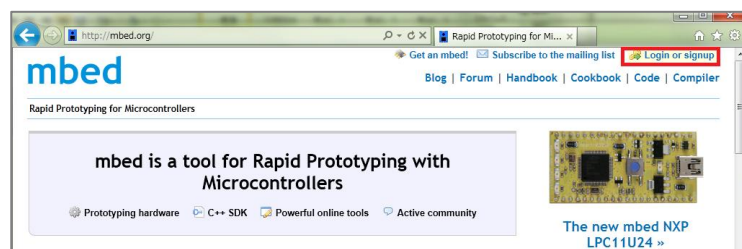


図 2.1 mbed のホームページ

ユーザ名とパスワードを入力し Login ボタンを押す。



図 2.2 ログイン画面

2.4 開発環境のページに行く

Compiler をクリックすると開発環境の画面になる。

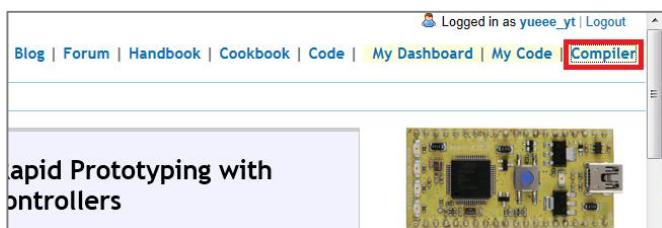


図 2.3 コンパイラへのリンク

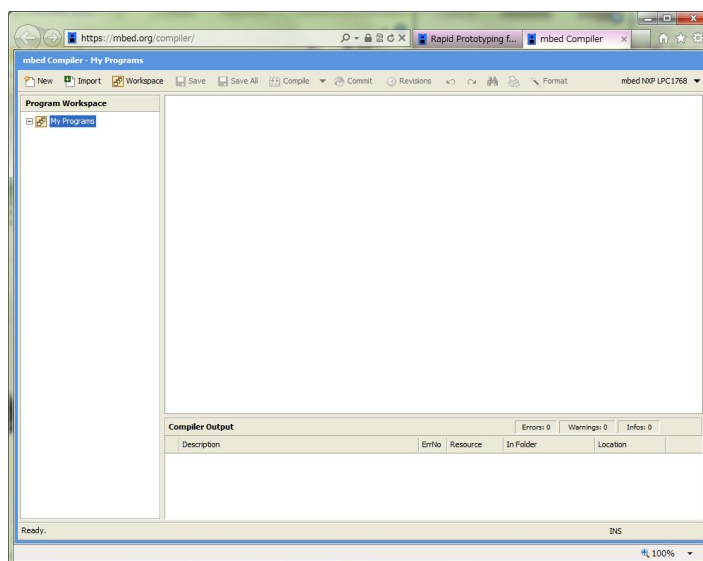


図 2.4 開発環境画面

2.5 プログラムの登録

New を押してプログラム名を登録する。（ここでは h24_HelloWorld）とする。

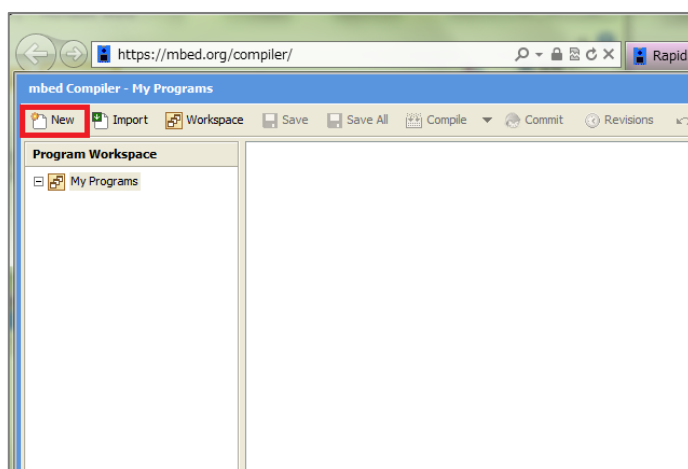


図 2.5 コンパイラ画面

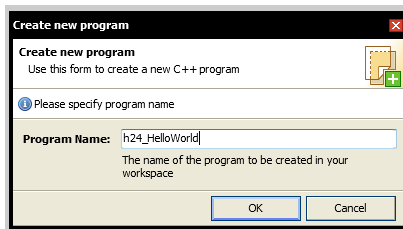


図 2.6 プログラム作成時のダイアログ画面

左の Workspace にプログラムが表示される。増えると Workspace ボタンを押すことで整理（表示・非表示を選択）することが可能である

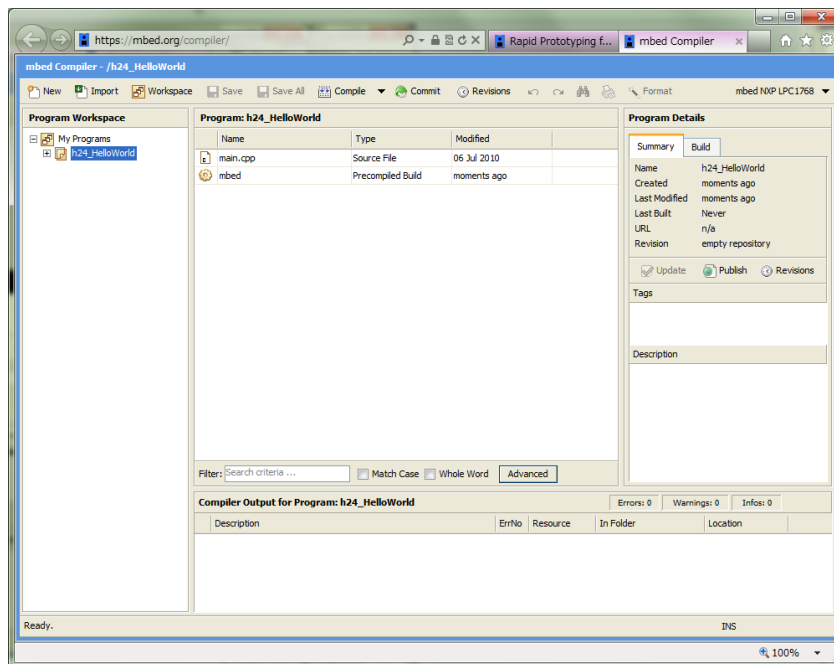


図 2.7 mbed 開発画面

main.cpp 内にサンプルの LED チカチカプログラムがかかれた状態で、初期化される。

```
#include "mbed.h"

DigitalOut myled(LED1);

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

1. mbed オフィシャルライブラリの Include
3. myled のクラス定義
6. 無限ループ
7. myled=1 は LED1 が点灯
8. 0.2 秒待ち

Compile ボタンをクリックするとコンパイルが始まり、エラーが無く実行形式が出来たら、ファイルの保存を聞いてくる。ファイルは mbed に保存する。

2.6 実行

mbed 本体のリセットボタンを押すと、内部に記録している一番新しいタイプスタンプのプログラムが実行される。

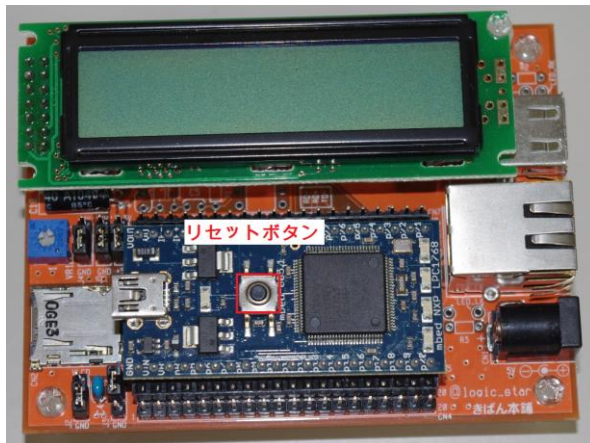


図 2.8 リセットボタン

※もし、プログラムの異常で書き込みが出来なくなった時はリセットボタンを押しながらファイルの転送処理をする。

2.7 デバッグの方法

mbed にはリアルタイム Debugger は備わっていない。しかしながら USB ケーブルを使用したシリアル通信を利用することで、実行途中の内容を PC に表示させることが可能である。

2.7.1 mbed ドライバのインストール

mbed のホームページからシリアルドライバをダウンロードし実行する。
今まで、フラッシュメモリとして認識していた mbed が複合デバイスとして認識されるようになり、USB を使用したシリアル通信を行う事が出来るようになる。

ハンドブックの Debugging から、

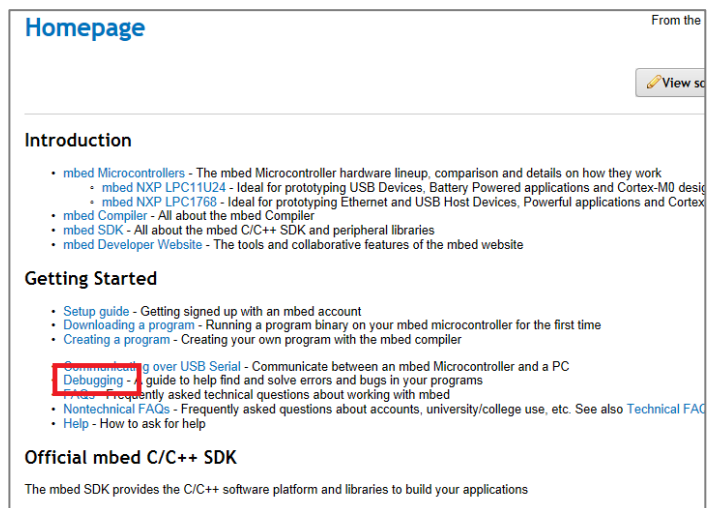


図 2.9 Debug の説明へのリンク

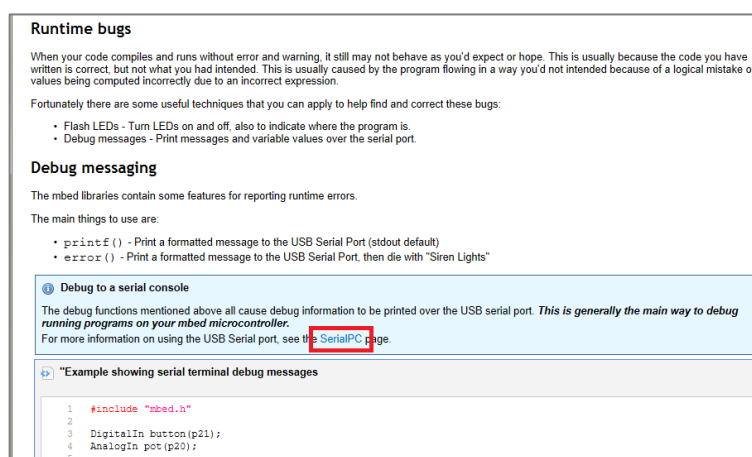


図 2.10 SerialPC へのリンク

Serial Communication with a PC

The mbed Microcontroller can communicate with a host PC through a "USB Virtual Serial Port" over the same USB cable that is used for programming.

This enables you to:

- Print out messages to a host PC terminal (useful for debugging!)
- Read input from the host PC keyboard
- Communicate with applications and programming languages running on the host PC that can communicate with a serial port, e.g. perl, python, java and so on.

Hello World!

```

1 #include "mbed.h"
2
3 Serial pc(USBTX, USBRX); // tx, rx
4
5 int main() {
6     pc.printf("Hello World!\n");
7 }

```

Host interface and terminal applications

Your mbed Microcontroller can appear on your computer as a serial port. On Mac and Linux, this will happen by default. For Windows, you need to install a driver:

Windows

See [Windows-serial-configuration](#) for full details about setting up Windows for *serial communication* with your mbed Microcontroller

It is common to use a *terminal application* on the host PC to communicate with the mbed Microcontroller. This allows the mbed Microcontroller to print to your PC screen, and for you to send characters back.

図 2.11 mbed のドライバ画面へのリンク

※ドライバは、mbed を変更した場合などで、再インストールする必要がある。

下記のようにプログラムを変更し、コンパイルして mbed に保存する。

```

#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1;
        printf( "On %n" );
        wait(0.2);
        myled = 0;
        printf( "Off %n" );
        wait(0.2);
    }
}

```

2.7.2 TeraTerm の起動

準備

TeraTerm のフォルダ内の ttermpro.exe をダブルクリックして起動する。

起動したら、新しい接続のダイアログでシリアルポートを選択し mbed に接続する。



図 2.12 TeraTerm 立ち上げ画面

設定

通信速度

通信速度は mbed のプログラムで指定が無い場合は、9600bps になっている。変更している場合は TeraTerm の設定→シリアルポートで変更する

受信改行処理

通常は、LF なので TeraTerm の設定→端末で変更する。



図 2.13 端末の設定ダイアログ

2.7.3 プログラムの起動

mbed をリセットし、プログラムを実行する。

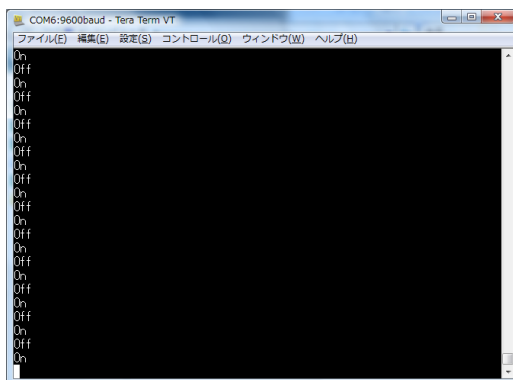


図 2.14 実行画面

3 簡単な例題

3.1 スイッチの動作

スイッチとの接続は Pinmode を PullUp にすることで、2本の結線で簡単に行える。

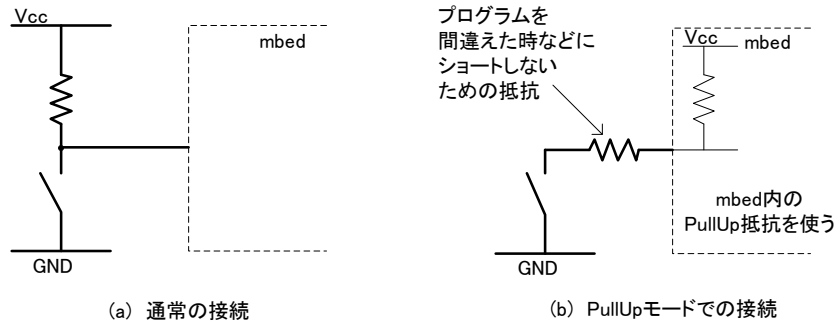


図 3.1 スイッチの接続

抵抗 1kΩを介して mbed の Pin18 にスイッチを接続し、ボタンが押している間光るプログラムを検証する。

```
#include "mbed.h"

DigitalOut myled(LED1);
DigitalIn sw(p18);

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

3.2 外部デジタル出力

上記プログラムの

DigitalOut myled(LED1); → DigitalOut myled(p17);

にして実行する。P18 には発光ダイオードを接続する。(電流制限抵抗は 1kΩとする)

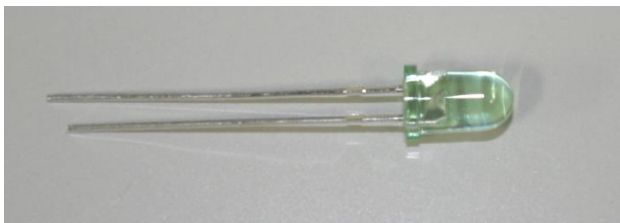


図 3.2 LED(足の長い方が+)

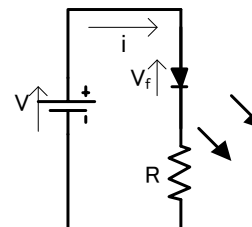


図 3.3 LED の点灯回路

3.3 カウンタの作成

タクトスイッチは ON-OFF 時にチャタリングが発生する。ソフト的にこれを解決する方法の一つとしてウェイト時間を設ける方法がある。下記の様なプログラムでカウンタの動作を確認することができる。(Wait をコメントしたり、時間を調節したりして、確認する)

```
#include "mbed.h"

BusOut leds (LED4, LED3, LED2, LED1);
InterruptIn sw(p18);
void countUp(void)
{
    static unsigned char i=0;
    i++;
    if(i==16) i=0;
    leds=i;
    wait(0.1); //Here!
}

int main()
{
    sw.mode(PullUp);
    sw.rise(&countUp);
    while(1);
}
```

表 3.1 ウェイト時間の検討

Wait	使い物になる？
なし	
0.001(1ms)	
0.01(10ms)	
0.1(100ms)	

3.4 圧電ブザー

圧電ブザーは電力を必要としないので、組み込み系マイコンに適した部品である。反面、音を出すためには方形波を必要とする。これは PWM を利用すると簡単に使える。ここでは Sw(p18)を押して 1 秒間音が出るようにする。圧電ブザーは、抵抗 1kΩを介して p21 に接続する。ここでは、440Hz で考える。圧電ブザーは Duty 比を増やしてもブザー音は大きくならないことを確認する。

```
#include "mbed.h"

PwmOut bz(p21);
DigitalIn sw(p18);

int main()
{
    sw.mode(PullUp);
    bz.period(1.0/440.0);
    while(1) {
        if(!sw) {
            bz=0.5; //Here
            wait(1);
            bz=0;
        }
    }
}
```

(参考)通常のデジタル出力を 2 つ使い交互で OnOff することで音を大きくすることができる。圧電ブザーの一方を p20 に、もう一方を p21 に接続する。

```
include "mbed.h"

DigitalOut bz1(p20);
DigitalOut bz2(p21);
DigitalIn sw(p18);

int main()
{
    int i;
    sw.mode(PullUp);
    while(1) {
        if(!sw) {
            for(i=0; i<880; i++) {
                bz2=bz1;
                bz1=!bz1;
                wait(1.0/880.0);
            }
            bz1=bz2=0;
        }
    }
}
```

半周期のループ数なので 2 倍の 880 回ループになる。

4 計測システムの作成

4.1 概要

計測システムにおいては時間とともにデータを記録する必要がある。時間は、mbed のマイコンに内蔵している Real Time Clock (RTC) を使用することで、簡単に取り扱いできる。RTC は、初期の日時を設定すると、あとは自動で時間を取得できる時計である。初期の時刻設定は手動や、GPS、Ethernet で NTP サーバなどを利用して行う。

ここでは時間を NTP サーバから取得し 5 秒ごとにデータを収集するシステムを考える。データは USB フラッシュメモリ保存し、Web 公開出来るものを作成する。

表 4.1 システムの構成

IP アドレス	DHCP 取得
時刻収集先 (NTP サーバ)	ntp.nict.jp
温度センサ	ナショナルセミコンダクタ LM35
データ更新頻度	5 秒

4.2 ハードウェア

下図のように接続する。

温度センサー(LM35)は 4~30V で動作する。温度は 10mv/°Cで出力される。mbed の出力(VOUT)は 3.3V なので動作が不安定になる可能性があるので、mbed の入力電源(Vin)もしくは、USB 入力電源(VU)を利用するとよい。

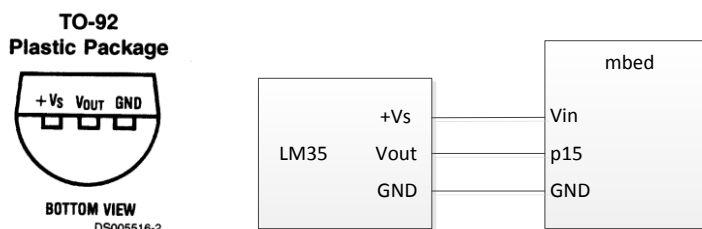


図 4.1 回路図

4.3 準備

mbed ホームページにログインする。

新しいプログラムを作成する準備をする。（“temp_web”として説明を続ける。）

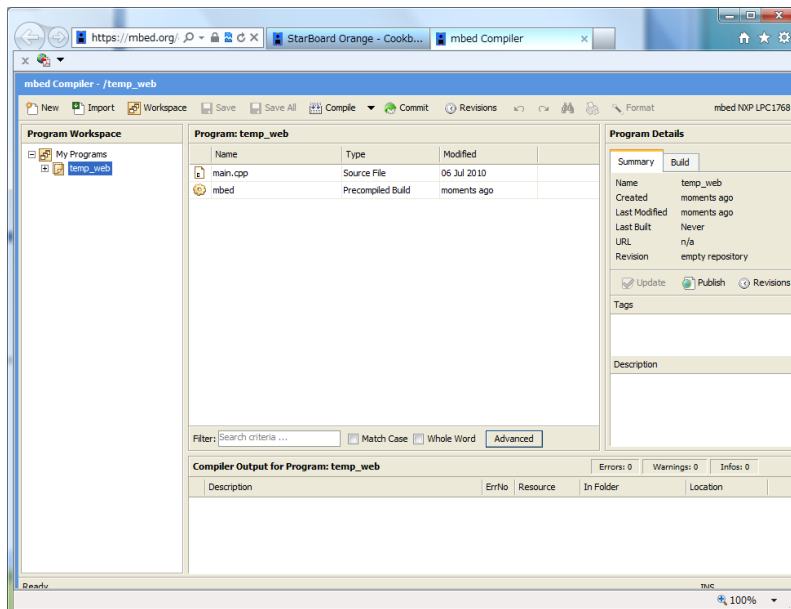


図 4.2 コンパイラ画面

一旦コンパイルして、動作するか確認する。

4.4 LCD の表示

クックブックの Text LCD ページに行く

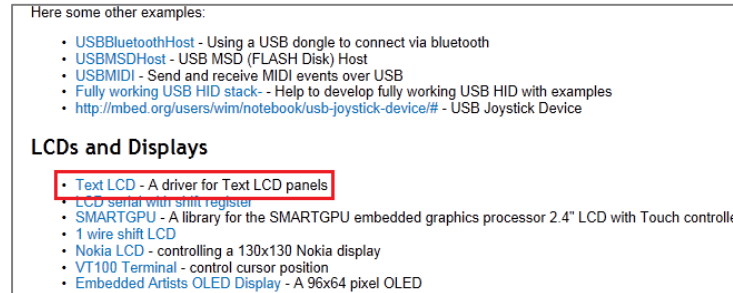


図 4.3 TextLCD ライブラリへのリンク画面

プログラムにライブラリを挿入するので **Import this library into a program** をクリック



図 4.4 テキストライブラリのインポートへのリンク画面

Path を今からインストールするプログラムに変更する。(ここでは temp_web)

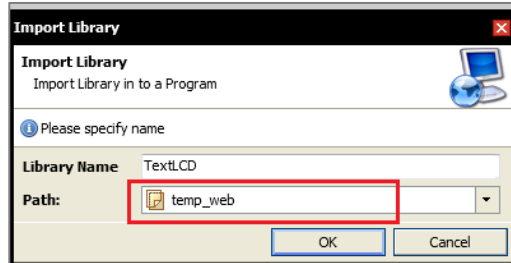


図 4.5 ImportLibrary のダイアログ画面

main.cpp のプログラムを変更する。

初期にできるサンプルのプログラムを 2 行目以降、削除して次のプログラムに変更する。

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

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

int main() {
    lcd.cls();
    lcd.locate(0, 0);
    lcd.printf("Program Start");
}
```

4.5 Ethernet ライブラリの導入

最近オフィシャルの Ethernet ライブラリが登場したため以下のライブラリは非推奨となっています。しかし、まだ対応する HTTPServer モジュールが無いので、ここでは古いユーザライブラリを使用する。

7月に mbed のオンラインコンパイラの仕様変更”Compile C as C and C++ as C++”が告知され、User ライブラリでは未対応になったままのものがある。Ethernet ライブラリも未対応の部分があるため、他のライブラリと一緒に再コンパイルするとエラーが発生する。

今回は配布したプログラムを利用する。配布プログラムは、ソースファイルの拡張子を.c から.cpp に変更しただけでホームページと同じである。

インポートの方法

1.Import を押す。

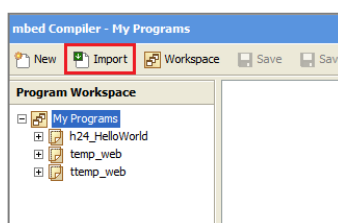


図 4.6 インポートボタン

2.Import 画面

- ①Local Machine を選択。
- ②参照ボタンを押し、配布プログラムの“4_6_NTP.zip”を選択。登録されると下段にリストアップされる。
- ③TargetName を入力。同名のプログラムは作れないのでここでは“temp_web2”と指定。
- ④Import ボタンを押す。

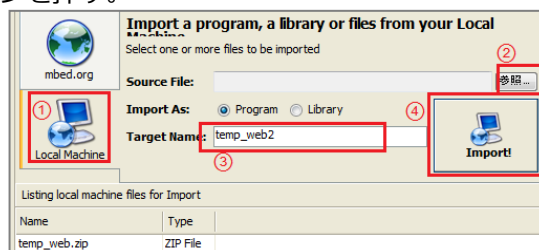


図 4.7 Import 画面

下のダイアログは OK を押す

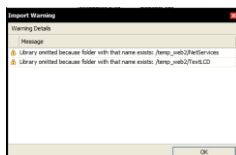


図 4.8 ワーニング画面

3. 4.7 実行へ

Ethernet ライブラリの導入方法

CookBook の NetServices を選択

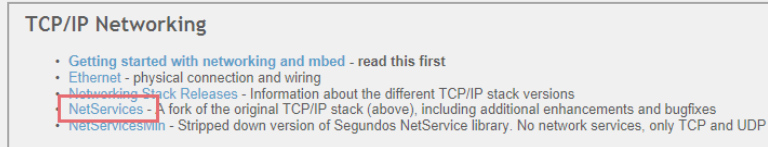


図 4.9 TCPIP のクックブックのネットワーキング

Import this library into a program でライブラリを読み込む

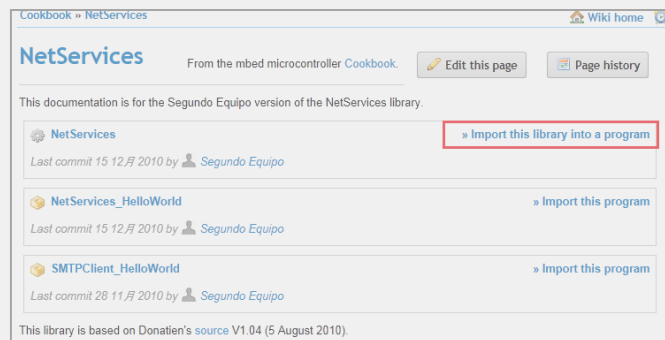


図 4.10 NetServices 画面

Path を目的のプログラムに合わせて読み込む。

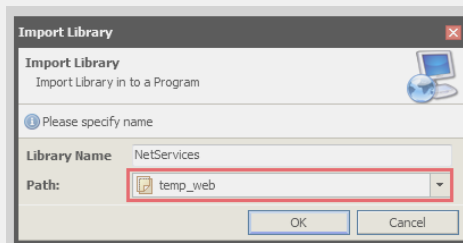


図 4.11 Import のダイアログ

正常に読み込めると Program Workspace が以下ようになる。

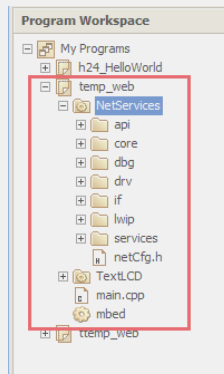


図 4.12 プログラムワークスペース

4.6 NTP のプログラムの作成

main.cpp に NTP の時間を読み取るプログラムを追加する。この時点で、時間を取得し、PC に送信する。

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

#define NTPServer "ntp.nict.jp"           NICT の NTP サーバ

#if 1
// Use DHCP
EthernetNetIf ethif;                     if 1(true)なのでこちらが実行される DHCP
#else
// Use "static IP address" (Parameters:IP, Subnet mask, Gateway, DNS)
EthernetNetIf ethif(IpAddr(xx,xx,xx,xx), IpAddr(xx,xx,xx,xx), IpAddr(xx,xx,xx,xx), IpAddr(xx,xx,xx,xx));
#endif

TextLCD lcd(p24, p26, p27, p28, p29, p30);
MSCFileSystem msc("usb");
LocalFileSystem local("local");
DigitalOut led1(LED1, "led1");
DigitalOut led2(LED2, "led2");
DigitalOut led3(LED3, "led3");
DigitalOut led4(LED4, "led4");
AnalogIn ain(p15, "ain");
float temp;
time_t ctTime;

void lastdate(char *input, char *output);
NTPClient ntp;                            NTP のクラス定義
HTTPServer svr;

RPCVariable<float> rpcv(&temp, "Temp");
RPCFunction rpcf(&lastdate, "LastDate");

void lastdate(char *input, char *output) {
    if (input[0]=='0') led4=0;
    if (input[0]=='1') led4=1;
    char ldate[32];
    strftime(ldate, 32, "%x %X", localtime(&ctTime));
    strcpy(output, ldate);
}

void mesure_temp() {
    char buffer[9];
    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);

    char filename[23];
```

```

    strftime(filename, 23, "/usb/data/%Y%m%d.dat", localtime(&ctTime));
    FILE *fp= fopen(filename, "a");
    if ( fp == NULL ) {
        printf("Could not open file for write\n");
    } else {
        fprintf(fp, "%s . %4.1f %r\n", buffer, temp);
    }
    fclose(fp);
}

int main() {
    time_t ctTime;
    char buffer[32];

    lcd.cls();
    lcd.locate(0, 0);
    lcd.printf("Program Start");

    if (ethif.setup()) {
        error("Ethernet setup failed.");
        return 1;
    }

    IpAddr ethIp=ethif.getIp();
    lcd.locate(0, 0);
    lcd.printf("%3d.%3d.%3d.%3d", ethIp[0], ethIp[1], ethIp[2], ethIp[3]);
    Host server(IpAddr(), 123, NTPServer);
    ntp.setTime(server);
    //UTC→JST +9Hour (32400Sec)          NTPの時刻はUTCなので9時間足してJSTにする
    ctTime = time(NULL);                いったん時刻を呼び出す
    ctTime+=32400;                       9時間足す
    set_time(ctTime);                    再度代入
    ctTime = time(NULL);                現在時を得る
    strftime(buffer, 32, "%x %X", localtime(&ctTime)); 時間を書式化してbufferに代入
    printf("%s \n", buffer);            PCにbufferの内容を送信
    printf("finish \n");

    Base::add_rpc_class<DigitalOut>();
    Base::add_rpc_class<AnalogIn>();
    FSHandler::mount("/local", "/");
    svr.addHandler<RPCHandler>("/rpc");
    svr.addHandler<FSHandler>("/");
    svr.addHandler<SimpleHandler>("/hello");
    svr.bind(80);

    Timer tm;
    Timer tm2;
    tm.start();
    tm2.start();
    while (true) {
        Net::poll();
        if (tm.read()>.5) {
            led1=!led1; //Show that we are alive
            tm.start();
        }
        if (tm2.read()>5) {
            led2=!led2;
            tm2.start();
            mesure_temp();
        }
    }
}

```

実行

実行すると、LCD 上に DHCP から取得した IP アドレスを、TeraTerm 上には NTP サーバから取得した時間が表示される。

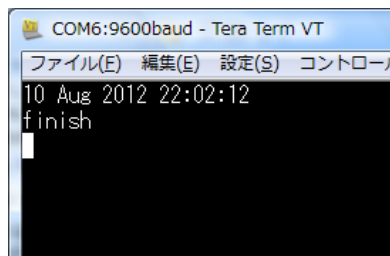


図 4.13 テラタームの実行画面



図 4.14 実行画面

※一部の DHCP サービスとは相性が悪いものもあるみたいです。その時は静的アドレスを割り当ててください。

4.7 温度計測し、USB フラッシュメモリに保存する部分を作成

温度センサの出力は 10.0mV/°C である。20°C の時は 200mV、40°C の時は 400mV の出力がある。

mbed のアナログ入力は p 15- p 20 までなので p 15 を使用する。

データ更新頻度を 5 秒間隔にする。時間の割り込みは Timer を使用する。

mbed のアナログ入力は 0 ~ 1 で出力される。動作電圧は 3.3V なので、3.3 を掛けることで入力電圧になる。

ライブラリのインポート

USB フラッシュメモリ (USB Mass Storage Class) のライブラリは Cookbook の Storage から、行く

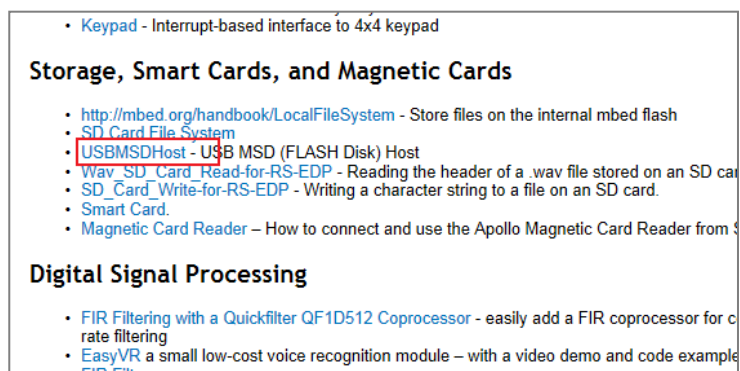


図 4.15 USBMassStorageDevice の選択画面



図 4.16 USBMSDHost のホームページ

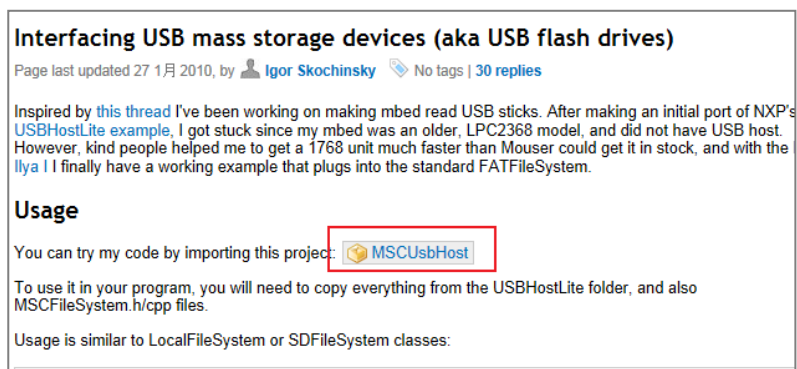


図 4.17 ライブラリへのリンク

ページのアドレスをコピーして、ライブラリとして Import する。

① アドレスをコピーする。

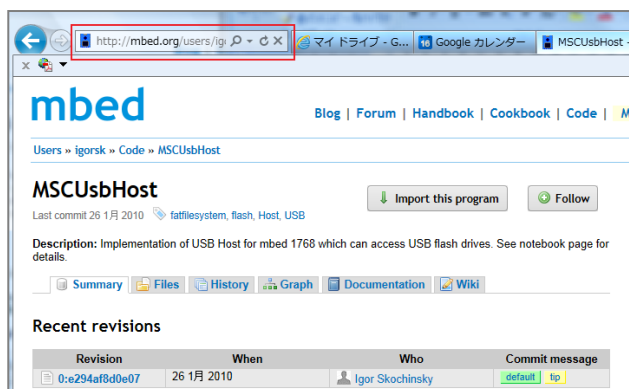


図 4.18 ライブラリのページ

② コンパイラの開いているブラウザ画面で Import ボタンを押す。

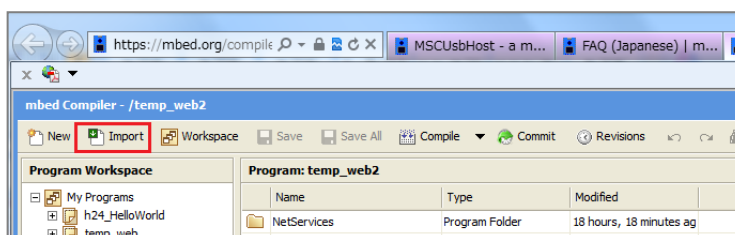


図 4.19 インポートボタン

③ インポートする

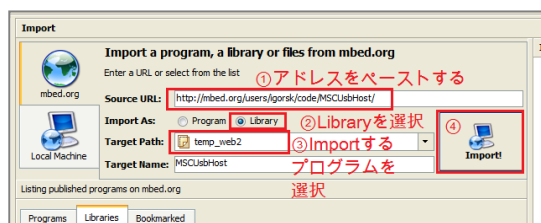


図 4.20 インポート画面

- ④ Import したファイルのうち不要なファイル（main.cpp と mbed ライブラリ）を削除

※Import したフォルダ内のファイルです。

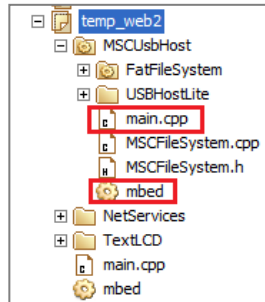


図 4.21 プログラムワークスペース

- ⑤ プログラムが今まで通りコンパイルできるか確認する。

プログラム

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

#define NTPServer "ntp.nict.jp"

#if 1
// Use DHCP
EthernetNetIf ethif;
#else
// Use "static IP address" (Parameters:IP, Subnet mask, Gateway, DNS)
EthernetNetIf ethif(IpAddr(xx, xx, xx, xx), IpAddr(xx, xx, xx, xx), IpAddr(xx, xx, xx, xx), IpAddr(xx, xx, xx, xx));
#endif

TextLCD lcd(p24, p26, p27, p28, p29, p30);
MSCFileSystem msc("usb"); 外付け USB フラッシュメモリを/usb として定義
LocalFileSystem local("local");
DigitalOut led1(LED1, "led1"); LED 1 をデジタル出力で led1 として定義 (RPC で使用するときには名前を登録)
DigitalOut led2(LED2, "led2"); LED2 をデジタル出力で led2 として定義
DigitalOut led3(LED3, "led3");
DigitalOut led4(LED4, "led4");
AnalogIn ain(p15, "ain"); p15 をアナログ入力で ain として定義
float temp; 温度用の変数 (どのルーチンからも見られるように大域で定義)
time_t ctTime; 最後の計測時刻保存用の変数 (どのルーチンからも見られるように大域で定義)

void lastdate(char *input, char *output);
NTPClient ntp;
HTTPServer svr;
```

```

RPCVariable<float> rpcv(&temp, "Temp");
RPCFunction rpcf(&lastdate, "LastDate");

void lastdate(char *input, char *output) {
    if (input[0]=='0') led4=0;
    if (input[0]=='1') led4=1;
    char ldate[32];
    strftime(ldate, 32, "%x %X", localtime(&ctTime));
    strcpy(output, ldate);
}

void mesure_temp() {
    char buffer[9];
    temp=ain*3.3*100.0;      入力値に 3.3 を掛けて電圧に、100 掛けて温度に変換
    ctTime = time(NULL);    データ収集時刻を ctTime に代入
    strftime(buffer, 9, "%X", localtime(&ctTime));    時刻を読みやすい表示で buffer に代入
    lcd.locate(0, 1);
    lcd.printf("%s %4.1fDeg", buffer, temp);

    char filename[23];
    strftime(filename, 23, "/usb/data/%Y%m%d.dat", localtime(&ctTime));    ファイル名を付ける
    FILE *fp= fopen(filename, "a");    ファイルを追記モードでオープンする
    if ( fp == NULL ) {
        printf("Could not open file for write\n");
    } else {
        fprintf(fp, "%s , %4.1f ¥r¥n", buffer, temp);    日付と温度をファイルに記録する
    }
    fclose(fp);
}

int main() {
    time_t ctTime;
    char buffer[32];

    lcd.cls();
    lcd.locate(0, 0);
    lcd.printf("Program Start");

    if (ethif.setup()) {
        error("Ethernet setup failed.");
        return 1;
    }

    IpAddr ethIp=ethif.getIp();
    lcd.locate(0, 0);
    lcd.printf("%3d.%3d.%3d.%3d", ethIp[0], ethIp[1], ethIp[2], ethIp[3]);
    Host server (IpAddr(), 123, NTPServer);
    ntp.setTime(server);
    //UTC-->JST +9Hour (32400Sec)
    ctTime = time(NULL);
    ctTime+=32400;
    set_time(ctTime);
    ctTime = time(NULL);
    strftime(buffer, 32, "%x %X", localtime(&ctTime));
    printf("%s ¥n", buffer);
    printf("finish ¥n");
}

```

```
Base::add_rpc_class<DigitalOut>();
Base::add_rpc_class<AnalogIn>();
FSHandler::mount("/local", "/");
svr.addHandler<RPCHandler>("/rpc");
svr.addHandler<FSHandler>("/");
svr.addHandler<SimpleHandler>("/hello");
svr.bind(80);

Timer tm;      LED1 (ハートビート) 用のタイマー定義
Timer tm2;     LED2 (計測間隔) 用のタイマー定義
tm.start();
tm2.start();
while (true) {
    Net::poll();
    if (tm.read() > 5) { tm がスタートしてから 0.5 秒以上経っていたら
        led1=!led1; //Show that we are alive LED1 を反転させる
        tm.start(); tm を再スタート
    }
    if (tm2.read() > 5) {
        led2=!led2;
        tm2.start();
        mesure_temp(); 計測のルーチンを CALL
    }
}
}
```

実行

USB フラッシュメモリに”data”フォルダを作成し☆Board Orange に接続。
LAN を接続してリセットボタンを押す。

解説

mbed には、定期的な割り込みを発生させる Ticker クラスが備わっているが、ここでは利用しない。理由は、USB や Ethernet など割り込みを必要とされそうな機能を多く使用するので、割り込みの衝突を避けるためである。シングルスレッドの組み込み系マイコンはこのあたりが弱い。今後は最近登場したスレッド管理ができる mbed RTOS を利用することで、可能になっていくと思われる。

4.8 Web サーバの構築

index.htm と jquery.js、mbedrpc.js は、mbed 本体の USB フラッシュメモリに保存する。
本体の USB フラッシュメモリは、下記の注意点がある。

※ファイル名は 8.3 形式

※サブディレクトリは扱えない。

ライブラリの Import

web から変数を見たり、サブルーチンを動かすためのライブラリを Import する。Web サーバの部分やピンの状態を制御したりする部分は、NTPClient で Import した Netservices ライブラリに含まれている。

クックブックの Interfacing with other Languages の Interface Using RPC を選択する。
なお、今回コピーした mbedrpc.js は Intefacing with JavaScript 内に置いてある。

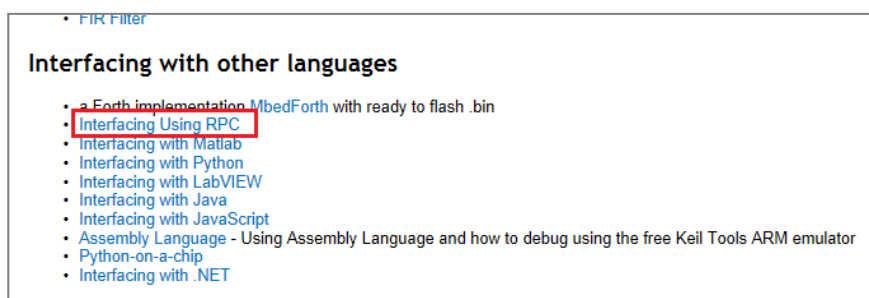


図 4.22 RPC ライブラリへのリンク

ページ内の Adding RPC to your own code のライブラリをインポートする。

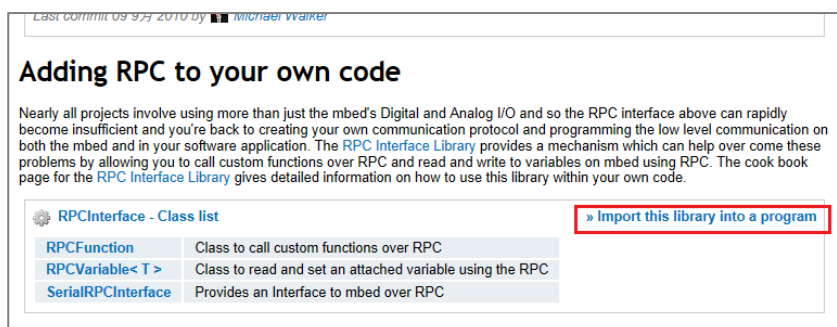


図 4.23 ライブラリのインポートへのリンク

プログラム

mbed のプログラム

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

#define NTPServer "ntp.nict.jp"

#if 1
// Use DHCP
EthernetNetIf ethif;
#else
// Use "static IP address" (Parameters: IP, Subnet mask, Gateway, DNS)
EthernetNetIf ethif(IpAddr(xx, xx, xx, xx), IpAddr(xx, xx, xx, xx), IpAddr(xx, xx, xx, xx), IpAddr(xx, xx, xx, xx));
#endif

TextLCD lcd(p24, p26, p27, p28, p29, p30);
MSCFileSystem msc("usb");
LocalFileSystem local("local"); // mbed 本体のフラッシュメモリを local と定義
DigitalOut led1(LED1, "led1");
DigitalOut led2(LED2, "led2");
DigitalOut led3(LED3, "led3");
DigitalOut led4(LED4, "led4");
AnalogIn ain(p15, "ain");
float temp;
time_t ctTime;

void lastdate(char *input, char *output);
NTPClient ntp;
HTTPServer svr;

RPCVariable<float> rpcv(&temp, "Temp");
RPCFunction rpcf(&lastdate, "LastDate");

void lastdate(char *input, char *output) { // Web から呼ばれる関数 時間を返す。
    if (input[0]=='0') led4=0; // 入力文字列が '0' なら LED 4 を消灯
    if (input[0]=='1') led4=1; // 入力文字列が '1' なら LED 4 を点灯
    char ldate[32];
    strftime(ldate, 32, "%x %X", localtime(&ctTime)); // 変数 ldate に書式化したデータ収集時間を代入
    strcpy(output, ldate); // 変数 output に ldate をコピー
}
```

```

void mesure_temp() {
    char buffer[9];
    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);

    char filename[23];
    strftime(filename, 23, "/usb/data/%Y%m%d.dat", localtime(&ctTime));
    FILE *fp= fopen(filename, "a");
    if ( fp == NULL ) {
        printf("Could not open file for write\n");
    } else {
        fprintf(fp, "%s , %4.1f %r\n", buffer, temp);
    }
    fclose(fp);
}

int main() {
    time_t ctTime;
    lcd.cls();
    char buffer[32];

    lcd.locate(0, 0);
    lcd.printf("Program Start");

    if (ethif.setup()) {
        error("Ethernet setup failed.");
        return 1;
    }

    IpAddr ethIp=ethif.getIp();
    lcd.locate(0, 0);
    lcd.printf("%3d.%3d.%3d.%3d", ethIp[0], ethIp[1], ethIp[2], ethIp[3]);
    Host server(IpAddr(), 123, NTPServer);
    ntp.setTime(server);
    //UTC-->JST +9Hour (32400Sec)
    ctTime = time(NULL);
    ctTime+=32400;
    set_time(ctTime);
    ctTime = time(NULL);
    strftime(buffer, 32, "%x %X", localtime(&ctTime));
    printf("%s %n", buffer);
    printf("finish %n");

    Base::add_rpc_class<DigitalOut>();      Digital ポートが Web から直接操作できるようにする
    Base::add_rpc_class<AnalogIn>();       Analog ポートが Web から直接操作できるようにする
    FSHandler::mount("/local", "/");     /local (mbed 本体のフラッシュメモリ) を web のルート "/" にする
    svr.addHandler<RPCHandler>("/rpc");
    svr.addHandler<FSHandler>("/");
    svr.addHandler<SimpleHandler>("/hello");
    svr.bind(80);                        ポート 80 に接続
}

```

```

Timer tm;
Timer tm2;
tm.start();
tm2.start();
while (true) {
  Net::poll(); Web から request が来ていないか問い合わせする
  if (tm.read() > 5) {
    led1=!led1; //Show that we are alive
    tm.start();
  }
  if (tm2.read() > 5) {
    led2=!led2;
    tm2.start();
    mesure_temp();
  }
}
}

```

HTML ファイル (index.htm として保存する。)

```

<html>
<head>
<title>平成24年度中国四国地区技術職員研修</title>
<script type="text/javascript" src="jquery.js" ></script>
<script type="text/jscript" src="mbedrpc.js"></script>
<script type="text/javascript">
  mbed = new HTTPRPC();
  led3 = new DigitalOut(mbed, LED3);
  ain = new AnalogIn(mbed, p15);
  vtemp = new RPCVariable(mbed, "Temp");
  ld = new RPCFunction(mbed, "LastDate");
  vled4 = "0";
  function loaded() { HTMLを読み込み後に実行
    $("#temp").text("データを待っています");
    tick();
    $("#btn_3").click(function () { IDがbtn_3をクリックした時の実行する内容を定義する
      vled = !led3.read();
      if (vled == true) led3.write(1);
      if (vled == false) led3.write(0);
    });
    setInterval("tick()", 2000); 2秒毎に tick を実行する
  }
  function tick() {led();temp();} tickはledとtemp関数を実行する
  function temp() {
    vvtemp = new Number(vtemp.read()); 温度を読み取る
    vain = ain.read()*3.3; p15のアナログ値を読み取る
    $("#temp").text(vvtemp.toFixed(2)); IDがtempのところのテキストを書き換える
    $("#date").text(ld.run(vled4)); mbedのLastDateを変数vled4の値を代入し実行し、返り値を得る。
    $("#ain").text(vain.toFixed(4)); 変数vainの値を少数第2位までテキストを書き換える
    if (vled4 == "0") vled4 = "1"; vled4の値を'0'と'1'で実行のたびに入れ替える
    else vled4 = "0";
  }
}

```



```

function led() {
  vled = led3.read(); ||led3の値を読み取る
  if (vled == true) $("#btn_3").css("background-color", "Orange");
  もし、LEDが点灯していたら、IDがbtn_3の背景をオレンジ色にする
  else $("#btn_3").css("background-color", "White");
  それ以外(消灯)なら、背景を白色にする
}
</script>
</head>
<body onload="loaded()">
<table border="1">
<tr><td>現在の温度</td><td><span id="temp">000</span></td></tr>
<tr><td>更新日</td><td><span id="date"></span></td></tr>
<tr><td>アナログ入力値</td><td><span id="ain"></span></td></tr>
</table>
<br />
<input type="button" id="btn_3" value="led3" />
</body>
</html>

```

実行

index.htm と jquery.js、mbedrpc.js のファイルを mbed 本体のフラッシュメモリに置く。
 (jquery.js は jquery-1.8.0.min.js のファイル名を変更したもの。)

PC のブラウザから <http://xxx.xxx.xxx.xxx/index.htm> を開く。(約 30 秒程度読み込みに時間がかかる)

xxx.xxx.xxx.xxx は、☆Board Orange の LCD に表示された IP アドレス

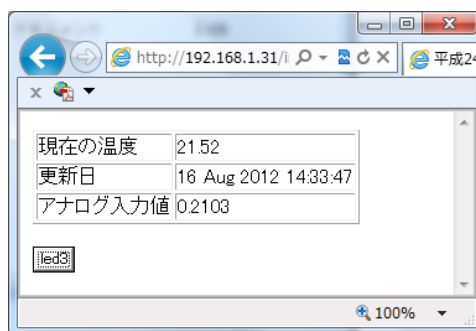


図 4.24 Web 実行画面

※オフィシャルライブラリが利用できるようになると、読み込み時間は改善されると思われる。

※スクリプトの内容を詳しく知りたい方は jquery の説明を読むと良い。

5 さらに大規模なデータ収集システムの構築

複数台のセンサー情報を集めてサーバに収集する方法の一つとして GUTP で考えられた IEEE1888(FIAP)を利用する方法がある。GUTP のホームページでダウンロードできる SDK には、PHP や Java、C# (.Net Framework) で利用する方法が説明されている。また、トランジスタ技術にはフィジカルコンピュータの Arudino での利用方法が掲載された。

SDK には、データを保存する FIAP Storage、1 分ごとにサンプルデータを作成する DataDummyLoader、簡単にデータを表示する FIAPSimpleSCADA から成る。

下図での GW は、mbed 等での作成が可能である。

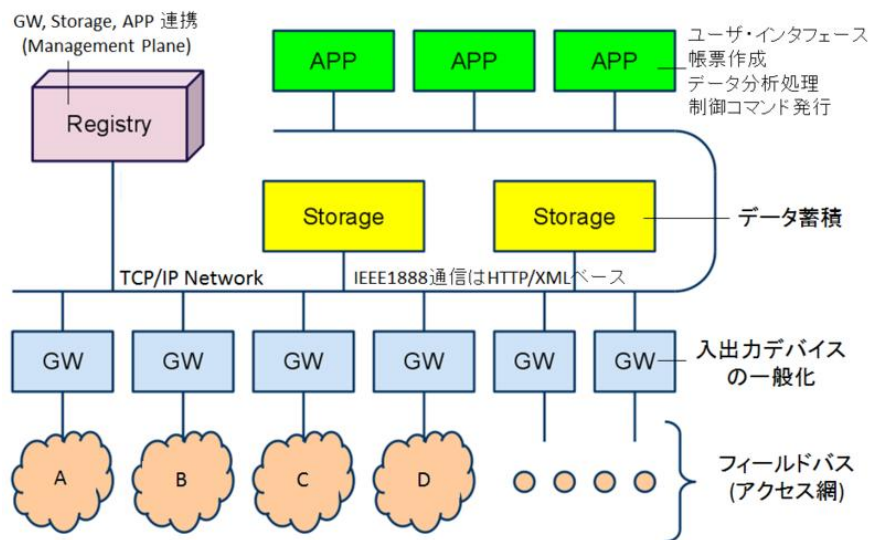


図 5.1 IEEE1888(FIAP)概念図

ここでは、下図の様のシステムを構成して検証する。

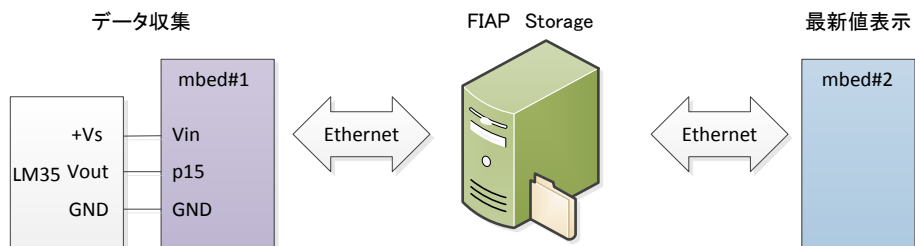


図 5.2 実験回路

5.1 IEEE1888 でのデータ収集プログラム (Write 手順)

IEEE 1888 (FIAP) ではデータを書き込むことを WRITE 手順と定義する。

このプログラムでは、新しい EthernetInterface ライブラリを使用するので NTP の設定の関数等が異なる。多数のデータを同時に保存する場合は element 内に並べて書けばよい。

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

#define NTPServer "ntp.nict.jp"

EthernetInterface eth;
NTPClient ntp;
time_t ctTime;

TextLCD lcd(p24, p26, p27, p28, p29, p30);
AnalogIn ain(p15);
DigitalOut led(LED1);
char timezone[] = "+09:00"; // JST
char atemp[6];
FIAP fiap("http://192.168.1.3/axis2/services/FIAPStorage");
struct fiap_element element[] = {
    {"http://csse-tech.jp/temp_tauchi", 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);
}
```

データの領域を確保するため
FIAPStorage のアドレス
データ受け渡しのための構造体
PointID、value(データ)、year、month、day、hour、minute、second、タイムゾーン
このルーチンに来たら LED を点滅させる
アナログ電圧より温度を求める
現在時刻を求める
表示用に時間を文字化する
LCD に表示
構造体の Value に値も入れる
year に年を入れる。RTC は 1900 年からの年を返す
month に月を入れる。RTC は 1 月が 0 を返す
データを 1 件保存する

```

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%rn");
if (ntp.setTime(NTPServer) == 0) {
printf("Set time successfully%r%rn");
time_t ctTime;
ctTime = time(NULL);
ctTime+=32400;
set_time(ctTime);
ctTime = time(NULL);
printf("Time is set to (JST): %s%r%rn", ctime(&ctTime));
printf("finish %r%rn");
} else {
lcd.locate(0,1);
lcd.printf("Error");
return -1;
}
//fiap.debug_mode=true;
//eth.disconnect();
while(true) {
tick();
wait(2);    2秒待つ
}
}

```

5.2 FIAP Storage のデータ表示 (FETCH 手順)

IEEE1888 ではデータを収集することを Fetch 手順と定義している。Fetch 手順では、ポイント ID のみ構造体に登録して fetch_last_data 関数を実行すると、データが入って戻ってくる。

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

EthernetInterface eth;

TextLCD lcd(p24, p26, p27, p28, p29, p30);
DigitalOut led(LED1);
char timezone[] = "+09:00"; // JST
FIAP fiap("http://192.168.1.3/axis2/services/FIAPStorage");
char atemp[10];
struct fiap_element element[] = {
    {"http://csse-tech.jp/temp_tauchi", atemp, NULL, NULL, NULL, NULL, NULL, NULL, timezone},
};

void tick(void)
{
    float temp;
    led=!led;
    fiap.fetch_last_data(element, 1);
    temp=atof(element[0].value);
    lcd.locate(0, 1);
    lcd.printf("%2d:%2d:%2d %4.1fDeg", element[0].hour, element[0].minute, element[0].second, temp);
}

int main()
{
    eth.init(); //Use DHCP
    eth.connect();
    lcd.cls();
    lcd.locate(0, 0);
    lcd.printf("%s", eth.getIPAddress());
    //fiap.debug_mode=true;
    while(true) {
        tick();
        wait(2);
    }
}
```

5.2.1 課題

- 2か所のデータを取得し、表示を2行にする
- 表示の値によってLEDを光らせる。26℃まで0個、26~28℃まで1個、28~30℃まで2個、30~32℃まで3個、32℃を超えたら4個表示するようなプログラムを作成
- 東京電力などの需要電力を表示する。

6 さらに

せっかくの機会なので、mbedにもっと触れてみましょう。

サンプルを用意しています。テキストでは、最低限の使い方やプログラムの説明を示します。

6.1 m3pi

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。

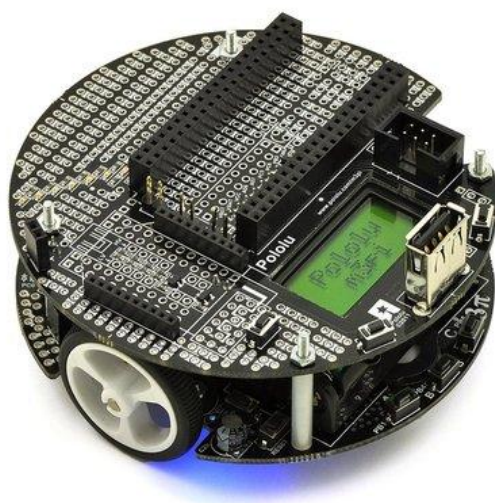


図 6.1 m3pi Robot(写真：Pololu 社ホームページより引用)

LEDは外側がLED1で内側がLED8。

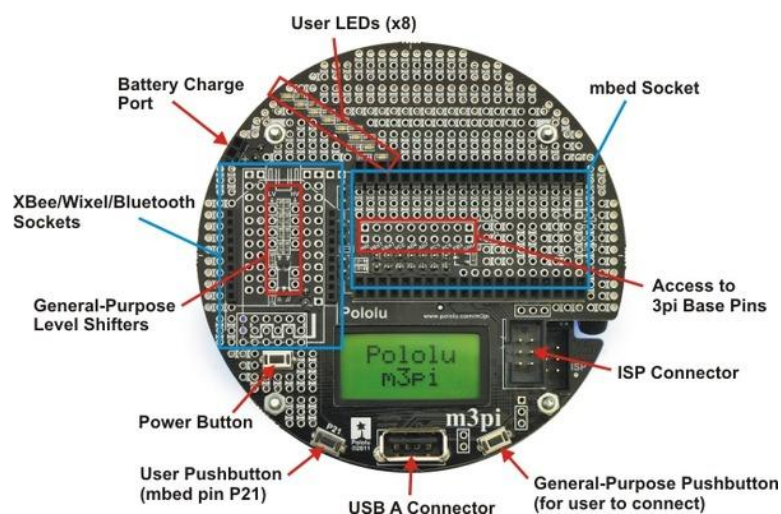


図 6.2 m3piのボタン等の配置図

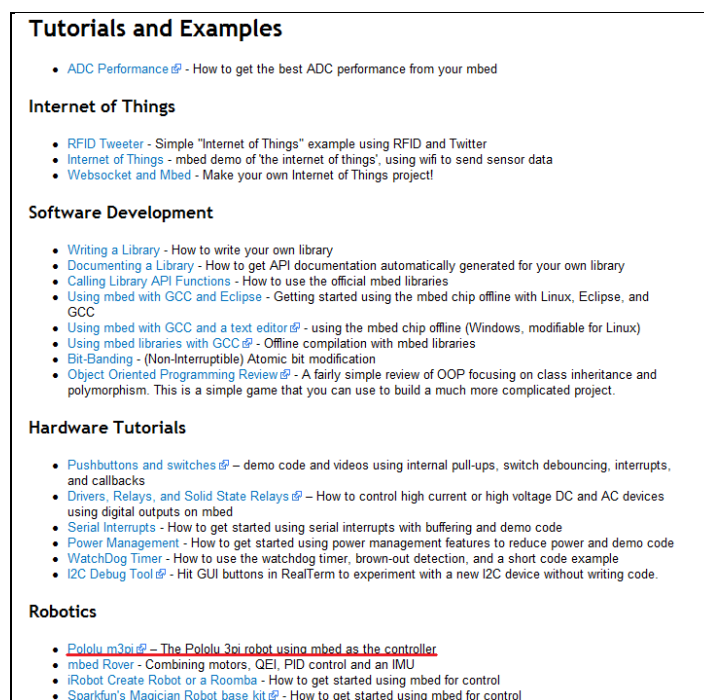
mbed へのプログラムの書き込みは m3pi に差し込んだままの状態で行える。書き込みが終了したのち USB ケーブルを外して m3pi の Power Button を押す。

※mbed に USB を接続したままで m3pi の電源を入れると mbed が切り離されるので注意する。

6.1.1 m3pi の Hello World

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.

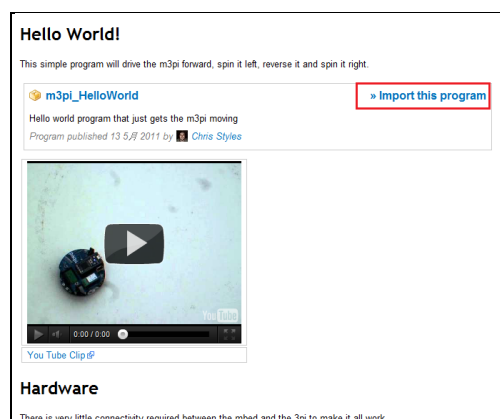
Hardware Tutorials

- [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

図 6.3 mbed の Cookbook の m3pi のページ



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](#)

[YouTube Clip](#)

Hardware

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

図 6.4 mbed Hello World のインポートのリンク

6.1.2 LED の表示

Cookbook に記載してある仕様によると m3pi はハーフスピードで 720deg/sec である。従って、フルスピードで 1 回転するには 0.25 秒である。

```
#include "mbed.h"
#include "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[]={0,1,0,0,0,1,0,1,1,1,1,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,0,0};
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[]={0,1,0,0,0,1,0,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,0,0};

int main() {
    int i, j;
    m3pi.locate(0,1);
    m3pi.printf("M3PI LED");
    wait(2.0);
    m3pi.left(1); // Turn left at full speed
    wait(0.1); // 回転が安定するまで待つ
    for (j=0; j<50; 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); // 空き時間の待ち
    }
    wait(0.5);
    m3pi.stop();
}
```

実行例

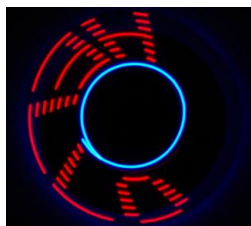


図 6.5 m3pi の LED を光らせた例

6.1.3 ライントレースロボット

ラインレースロボットは黒色の線(ビニールテープ)の上をなぞるように動作するものである。ここでは3種類見比べます。

☆PICなどで良くあるローレベルの制御(暗い方と逆のモータを制御)

```
#include "mbed.h"
#include "m3pi.h"

m3pi m3pi;
int main() {
    unsigned char x[10];
    int i;
    while(1) {
        m3pi.putc(0x86);
        for(i=0; i<10; i++) {
            x[i]=m3pi.getc();
        }
        m3pi.locate(0, 0);
        m3pi.printf("%2x %2x", (x[3]*0x100+x[2])/8, (x[7]*0x100+x[6])/8);
        m3pi.locate(0, 1);
        m3pi.printf("%2x %2x %2x", (x[1]*0x100+x[0])/8, (x[5]*0x100+x[4])/8, (x[9]*0x100+x[8])/8);
        if(x[1]*0x100+x[0]>x[9]*0x100+x[8]) {
            m3pi.left_motor(0.5);
            m3pi.right_motor(0);
        }else{
            m3pi.left_motor(0);
            m3pi.right_motor(0.5);
        }
    }
}
```

☆標準的なラインレース

クックブックのプログラム

m3piの高度なセンサー出力を利用する。

☆PIDを利用したラインレース

クックブックのプログラム

※パラメータ(比例係数p)を0.6に変えて実行してください。

6.1.4 LineMaze

ライン迷路は左手則を使って迷路を走ったのち、最短の距離で、ゴールに向かうものです。アルゴリズムは、3piのリソースにある。このアルゴリズムではループになるような迷路はサポートしない。

<http://www.pololu.com/file/0J195/line-maze-algorithm.pdf>

まっすぐ走れなければ、センサ情報が不確かになるので、基本は、PIDのラインレースになります。

```

#include "mbed.h"
#include "m3pi.h"
m3pi m3pi;
// Minimum and maximum motor speeds
#define MAX 0.3
#define MAX2 0.6 2度目のPIDは高速に
#define MIN 0
// PID terms
#define P_TERM 0.6
#define I_TERM 0
#define D_TERM 20
DigitalIn sw(p21);
char path[100] = "";
unsigned char path_length = 0;
void follow_segment(void);
void follow_segment2(void);
void m3pi_play( char x[]) { m3pi で音を出すために作った関数
    m3pi.putc(0xb3);
    int i;
    m3pi.putc(strlen(x));
    for (i=0; i<strlen(x); i++) {
        m3pi.putc(x[i]);
    }
}
void read_line(unsigned int sensors[5]) { m3pi のセンサー情報を読むために作った関数
    unsigned char x[10];
    int i;
    m3pi.putc(0x87);
    for (i=0; i<10; i++) {
        x[i]=m3pi.getc();
    }
    sensors[0]=(x[1]*0x100+x[0])*2;
    sensors[1]=(x[3]*0x100+x[2])*2;
    sensors[2]=(x[5]*0x100+x[4])*2;
    sensors[3]=(x[7]*0x100+x[6])*2;
    sensors[4]=(x[9]*0x100+x[8])*2;
}
char select_turn(unsigned char found_left, unsigned char found_straight, unsigned char found_right) {
// Make a decision about how to turn. The following code 左手則の解釈関数
// implements a left-hand-on-the-wall strategy, where we always
// turn as far to the left as possible.
    if (found_left)
        return 'L';
    else if (found_straight)
        return 'S';
    else if (found_right)
        return 'R';
    else
        return 'B';
}
void simplify_path() {
// only simplify the path if the second-to-last turn was a 'B'
    if (path_length < 3 || path[path_length-2] != 'B')
        return;
    int total_angle = 0;
    int i;
    for (i=1; i<=3; i++) {
        switch (path[path_length-i]) {

```

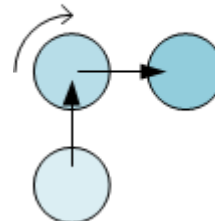
```

        case 'R':
            total_angle += 90;
            break;
        case 'L':
            total_angle += 270;
            break;
        case 'B':
            total_angle += 180;
            break;
    }
}
// Get the angle as a number between 0 and 360 degrees.
total_angle = total_angle % 360;
// Replace all of those turns with a single one.
switch (total_angle) {
    case 0:
        path[path_length - 3] = 'S';
        break;
    case 90:
        path[path_length - 3] = 'R';
        break;
    case 180:
        path[path_length - 3] = 'B';
        break;
    case 270:
        path[path_length - 3] = 'L';
        break;
}
// The path is now two steps shorter.
path_length -= 2;
}
void turn(char dir) {
    float a=0.0347;
    switch (dir) {
        case 'L':
            // Turn left.
            m3pi_play("T240ec");
            m3pi.left(0.2);
            wait(0.3125+a);
            break;
        case 'R':
            // Turn right.
            m3pi_play("T240ce");
            m3pi.right(0.2);
            wait(0.3125+a);
            break;
        case 'B':
            // Turn around.
            m3pi_play("T240cc");
            m3pi.left(0.2);
            wait(0.625+a);
            break;
        case 'S':
            // Don't do anything!
            break;
    }
}
void turn2(char dir) {
    float a=0.0347;
    switch (dir) {
        case 'L':

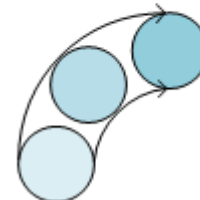
```

1回目の回転の関数
加速の時間を考慮した係数

1回目のターンは、
止まって回転



2回目のターンは、
止まらずに回転



2回目の回転プログラム 高速に曲がる様に

```

// Turn left.
                                m3pi_play("T240cc");
                                m3pi.right_motor(0.0);
                                m3pi.left_motor(0.6);
                                wait(0.223);
                                break;
                                case 'R':
// Turn right.
                                m3pi_play("T240cc");
                                m3pi.right_motor(0.6);
                                m3pi.left_motor(0);
                                wait(0.223);
                                break;
                                case 'B':
// Turn around.
                                m3pi_play("T240cc");
                                m3pi.left(0.2);
                                wait(0.625+a);
                                break;
                                case 'S':
// Don't do anything!
                                m3pi.forward(0.3);
                                wait(0.2+a);
                                break;
                                }
}
int main() {
    unsigned int sensors[5];
    sw.mode(PullUp);
    m3pi.locate(0,1);
    m3pi.printf("LineMaze");
    m3pi.leds(0);
    wait(2.0);
    unsigned char ll=0; LEDの表示の為に使用している変数
    m3pi.sensor_auto_calibrate();
    while (1) {
        follow_segment();
        m3pi.left_motor(0.2);
        m3pi.right_motor(0.2);
        wait(0.02);
//m3pi.left_motor(0);m3pi.right_motor(0);
        unsigned char found_left=0;
        unsigned char found_straight=0;
        unsigned char found_right=0;
        ll=0;
        read_line(sensors);
        if (sensors[0] > 1000) {
            found_left = 1;
            ll=ll|0x04;
        }
        if (sensors[4] > 1000) {
            found_right = 1;
            ll=ll|0x01;
        }
        wait(0.186);
        read_line(sensors);
        if (sensors[1] > 1000 || sensors[2] > 1000 || sensors[3] > 1000) {
            found_straight = 1;
            ll=ll|0x02;
        }
        if (sensors[1] > 1000 && sensors[2] > 1000 && sensors[3] > 1000) {
            ll=ll|0x08;
        }
    }
}

```

```

        break;          1 回目の走行終了
        // return;
    }
    unsigned char dir = select_turn(found_left, found_straight, found_right);
    turn(dir);
    path[path_length] = dir;
    path_length++;
// You should check to make sure that the path_length does not
// exceed the bounds of the array. We'll ignore that in this
// example.
// Simplify the learned path.
    simplify_path();
    m3pi.leds(11);
}
m3pi.stop();
while (1) {          ボタンが押されるまで待つ
    if (sw==0)break;
}
wait(1);
//2nd loop
int i;
for (i=0; i<path_length; i++) {
// SECOND MAIN LOOP BODY
    follow_segment2();
// Drive straight while slowing down, as before.
// m3pi.left_motor(0.2);
// m3pi.right_motor(0.2);
// wait(0.02);
// read_line(sensors);
// wait(0.186);
// Make a turn according to the instruction stored in
// path[i].
        turn2(path[i]);
    }
// Follow the last segment up to the finish.
    follow_segment2();
    m3pi.stop();
}
void follow_segment() {          1 回目のPIDのプログラム
    unsigned int sensors[5];
    float right;
    float left;
    float current_pos_of_line = 0.0;
    float previous_pos_of_line = 0.0;
    float derivative,proportional,integral = 0;
    float power;
    float speed = MAX;
    int k=0;
    while (1) {
// Get the position of the line.
        current_pos_of_line = m3pi.line_position();
        proportional = current_pos_of_line;
// Compute the derivative
        derivative = current_pos_of_line - previous_pos_of_line;
// Compute the integral
        integral += proportional;
// Remember the last position.
        previous_pos_of_line = current_pos_of_line;
// Compute the power
        power = (proportional * (P_TERM) ) + (integral*(I_TERM)) + (derivative*(D_TERM)) ;
// Compute new speeds

```

```

        right = speed+power;
        left = speed-power;
// limit checks
        if (right < MIN)
            right = MIN;
        else if (right > MAX)
            right = MAX;
        if (left < MIN)
            left = MIN;
        else if (left > MAX)
            left = MAX;
// set speed
        m3pi.left_motor(left);
        m3pi.right_motor(right);
//Read Sensor
        if (k>200) {
            read_line(sensors);
            if (sensors[1] < 650 && sensors[2] < 650 && sensors[3] < 650) {
// There is no line visible ahead, and we didn't see any
                3つのセンサーが反応がなくなると先端?ループを出す
// intersection. Must be a dead end.
                m3pi.leds(0x07);
                return;
            } else if (sensors[0] > 1000 || sensors[4] > 1000) {
// Found an intersection.
                両端のセンサーが反応すると分岐点?ループを出す。
                m3pi.leds(0x0f);
                return;
            }
        }
        k++;
    }
}
void follow_segment2() {
    2回目のPIDプログラム
    unsigned int sensors[5];
    float right;
    float left;
    float current_pos_of_line = 0.0;
    float previous_pos_of_line = 0.0;
    float derivative,proportional,integral = 0;
    float power;
    float speed = MAX2;
    int k=0;
    while (1) {
// Get the position of the line.
        current_pos_of_line = m3pi.line_position();
        proportional = current_pos_of_line;
// Compute the derivative
        derivative = current_pos_of_line - previous_pos_of_line;
// Compute the integral
        integral += proportional;
// Remember the last position.
        previous_pos_of_line = current_pos_of_line;
// Compute the power
        power = (proportional * (P_TERM) ) + (integral*(I_TERM)) + (derivative*(D_TERM)) ;
// Compute new speeds
        right = speed+power;
        left = speed-power;
// limit checks
        if (right < MIN)
            right = MIN;
        else if (right > MAX2)

```

```

        right = MAX2;
        if (left < MIN)
            left = MIN;
        else if (left > MAX2)
            left = MAX2;
// set speed
        m3pi.left_motor(left);
        m3pi.right_motor(right);
//Read Sensor
        if (k>100) {
            read_line(sensors);
            if (sensors[1] < 650 && sensors[2] < 650 && sensors[3] < 650) {
// There is no line visible ahead, and we didn't see any
// intersection. Must be a dead end.
                m3pi.leds(0x07);
                return;
            } else if (sensors[0] > 1000 || sensors[4] > 1000) {
// Found an intersection.
                m3pi.leds(0x0f);
                return;
            }
        }
        k++;
    }
}

```

実行方法

- ① m3pi をコースのスタート地点に置き m3pi 上の電源をオンにする。
左手則に従いゴールまで行き止まる。
- ② m3pi をスタート地点まで移動させ m3pi 上の p 21 ボタンを押す。
最短コースかつ高速に移動する。

6.2 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。

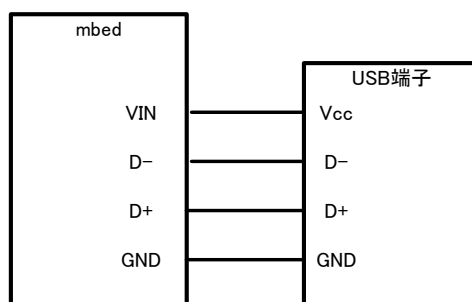


図 6.7 USB の接続図

6.2.1 USB マウスエミュレート

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

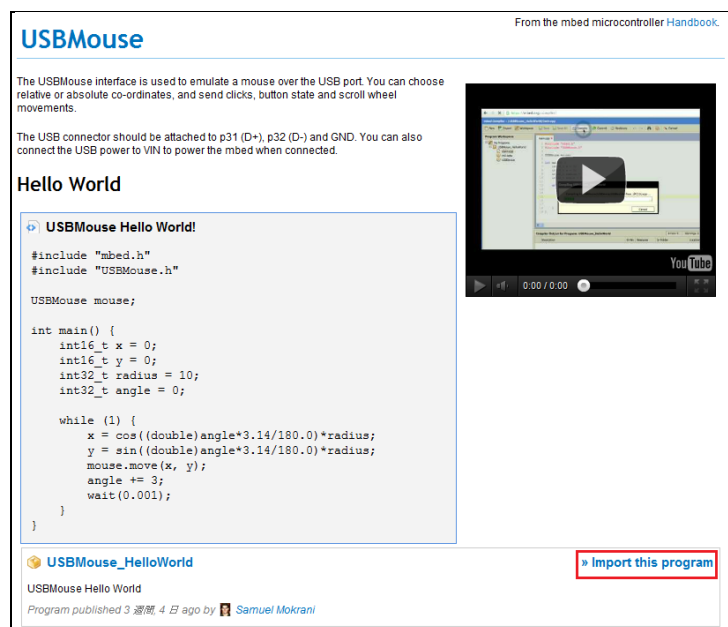
Networking & Comms

- [Serial](#) - Serial/UART bus
- [SerialHalfDuplex](#) - Half-duplex Serial bus
- [SPI](#) - SPI bus master
- [SPISlave](#) - SPI bus slave
- [SPiHalfDuplex](#) - Half-duplex SPI bus master
- [I2C](#) - I²C bus master
- [I2CSlave](#) - I²C bus slave
- [CAN](#) - Controller-area network bus
- [USBDevice](#) - Using mbed as a USB Device
 - [USBMouse](#) - Emulate a USB Mouse with absolute or relative positioning
 - [USBKeyboard](#) - Emulate a USB Keyboard, sending normal and media control keys
 - [USBMouseKeyboard](#) - Emulate a USB Keyboard and a USB mouse with absolute or relative positioning
 - [USBHID](#) - Communicate over a raw USBHID interface, great for driverless communication with a custom PC program
 - [USBMIDI](#) - Send and receive MIDI messages to control and be controlled by PC music sequencers etc
 - [USBSerial](#) - Create a virtual serial port over the USB port. Great to easily communicate with a computer.
 - [USBAudio](#) - Create a USBAudio device able to receive audio stream from a computer over USB.
 - [USBMSD](#) - Generic class which implements the Mass Storage Device protocol in order to access all kinds of block storage chips
- [Ethernet](#) - Ethernet network

Time & Interrupts

図 6.8 ハンドブックの USBMouse

(1) プログラムをインポート



The screenshot shows the mbed USBMouse page. It includes a title "USBMouse", a brief description of the interface, and a "Hello World" program. The code is as follows:

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

USBMouse mouse;

int main() {
    int16_t x = 0;
    int16_t y = 0;
    int32_t radius = 10;
    int32_t angle = 0;

    while (1) {
        x = cos((double)angle*3.14/180.0)*radius;
        y = sin((double)angle*3.14/180.0)*radius;
        mouse.move(x, y);
        angle += 3;
        wait(0.001);
    }
}
```

At the bottom, there is a red button labeled "Import this program".

図 6.9 USBMouse のページ

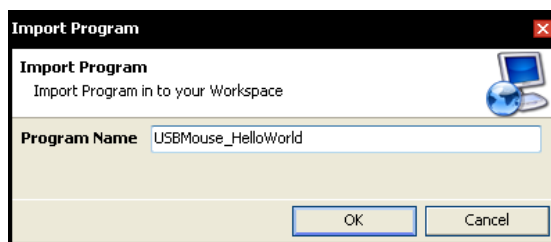


図 6.10 インポート先のダイアログ

- (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、ここであ
かう USB ではありません。注意！

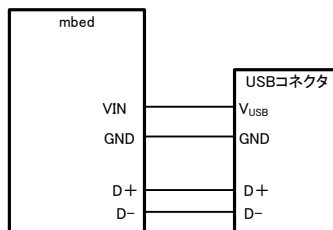


図 6.11 USB の接続図

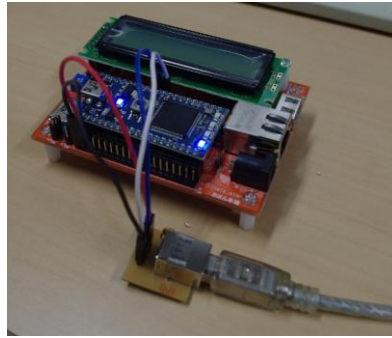


図 6.12 接続写真

(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。

6.2.2 USB キーボードエミュレート

マウスと同じように簡単に試せる。

6.2.3 USB オーディオ

最近の Arduino でもキーボードやマウスが簡単にエミュレートできるようになった。mbed

は、さらに高速なので USB オーディオ

ハンドブック内の USBAUDIO_speaker も DAC から音を出力できます。

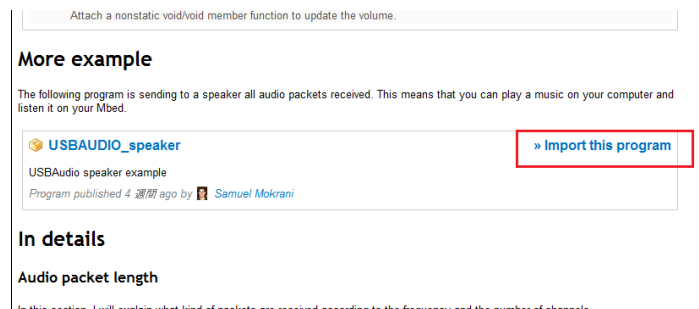


図 6.13 USBAUDIO_speaker のページ

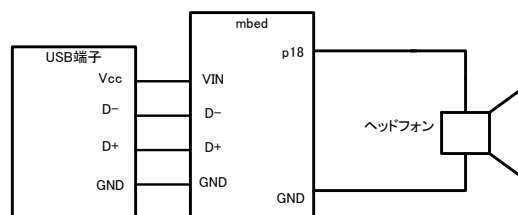


図 6.14 接続図

7 参考

http://www.nxp-lpc.com/lpc_micon/mbed/

<http://mbed.org/users/nxpfan/notebook>