

IEEE1888 通信ボード (学習用キット)

ユーザズ・マニュアル

2013 年 03 月版

1. はじめに

このマニュアルは、IEEE1888 通信ボード(学習用キット)の使い方、プログラミング方法、ハードウェアの構成方法を記載しています。このマニュアルに従って作業を進めていくことにより、この通信ボードに関する一通りの知識を身に着けることができます。また、組込みマイコンやデジタル回路についての知識があれば、さらに多様な応用に結び付けていくことができるようになります。なお、IEEE1888 通信ボードは **Arduino** をベースとしています。

■2011 年 11 月版 (Arduino-0023 プラットフォーム向け)からの更新内容

- **Arduino-1.0** プラットフォームに完全対応
- Arduino Uno ブートローダを搭載
- 初期ファームウェアで コマンドライン接続設定可能に
- 正規の MAC アドレス付与 (OUI コード:B0-12-66)

2. キットの内容

IEEE1888 通信ボード、学習用シールド¹、台座、AC アダプタ、USB ケーブル、LAN ケーブル、DVD-R、初期設定の情報が同梱されていることを確認してください。

DVD-R には、

- Arduino IDE (フォルダ名: arduino-1.0)
- IEEE1888 通信ライブラリ (フォルダ名: FIAPUploadAgent)
- Time ライブラリ (フォルダ名: Time)
- arduino-0023 対応ライブラリ (FIAPUploadAgent, EthernetDHCP, EthernetDNS)

が、同梱されています。

(*) Arduino-1.0 開発環境をお使いください(Arduino-1.0.x 系は注意が必要です)。

¹ Arduino(本 IEEE1888 通信ボードの原型となった組込みハードウェア)の世界では、ボード本体に接続する基板のことを、“シールド”と呼びます。

3. 組み立てる

まず、IEEE1888 通信ボードに台座を取り付けます(図 1)。その後、学習用シールドをコネクタに接続してください(図 2)。これで組み立ては完了です。

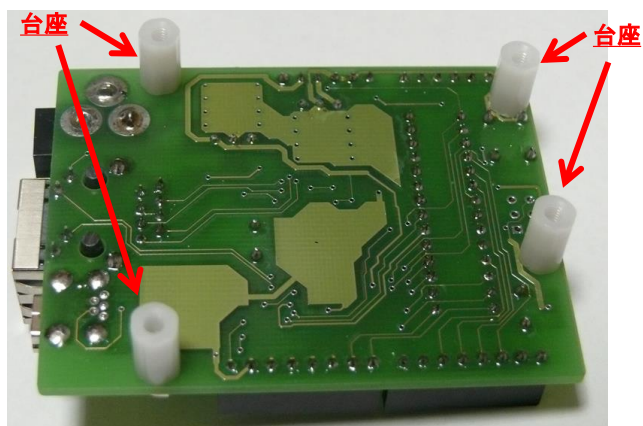


図 1: 台座の取り付け

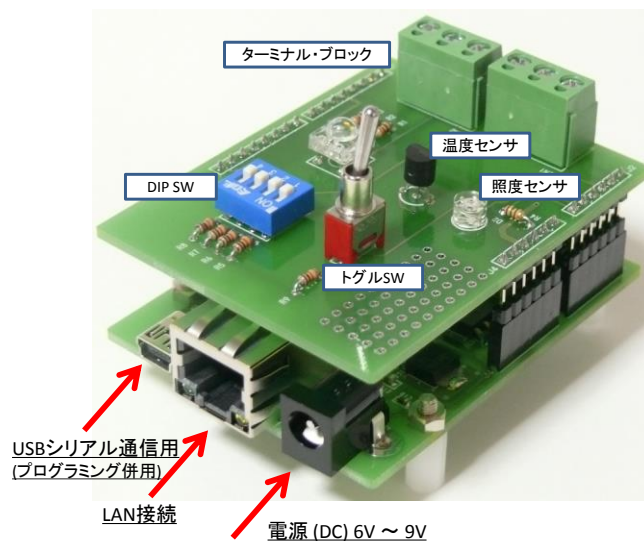


図 2: 学習用シールドを搭載した様子(完成図)

4. 初期設定で動作を確認する

DHCP(動的ホスト設定プロトコル)が有効なネットワークに接続し、電源を入れれば、初期設定情報に基づいて立ち上がります。LEDインジケータが、図3のように変化すれば成功です(LEDインジケータの意味は表1の通りです)。

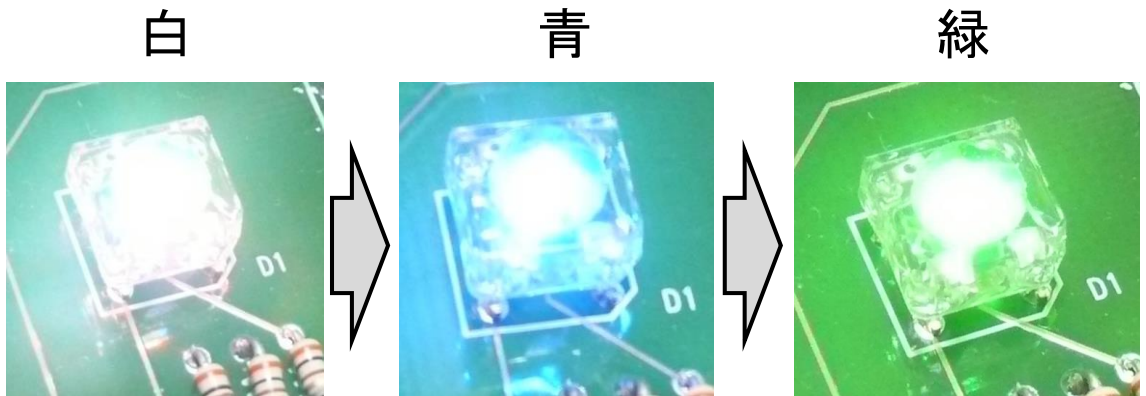


図 3: 電源の立ち上げから成功までの LED 表示 (白 → 青 → 緑)

表 1: LED インジケータによるステータス表示の内容

LED の色	意味
白色	NTP による時刻情報の取得中
青色	サーバへのアクセスおよびアップロード処理中
緑色	アップロード成功
消灯	次の測定までの待機中
紫色	サーバへの接続失敗 (TCP 接続失敗)
黄色	HTTP 通信エラー
水色	IEEE1888 通信エラー
赤色	その他のエラー

成功すると、fiap-sandbox サーバに計測データがアップロードされます。アップロードされたデータは、次の URL,

<http://fiap-sandbox.gutp.ic.i.u-tokyo.ac.jp/>

から閲覧できます(図 4)。同封されている「初期設定の情報」を参照し、該当するポイント ID を探してください。ポイント ID をクリックすると、アップロードされたデータのトレンドをグラフ(あるいは一覧)表示します(図 5)。

http://gutp.jp/Arduino/Sample05/DIPSW	2011-11-15T14:03:41.000+09:00	15
http://gutp.jp/Arduino/Sample05/Illuminance	2011-11-15T14:03:41.000+09:00	306
http://gutp.jp/Arduino/Sample05/TGLSW	2011-11-15T14:03:41.000+09:00	ON
http://gutp.jp/Arduino/Sample05/Temperature	2011-11-15T14:03:41.000+09:00	23.4

図 4: 該当するポイント ID を探し、タイムスタンプが最新であるかどうかを確認する。

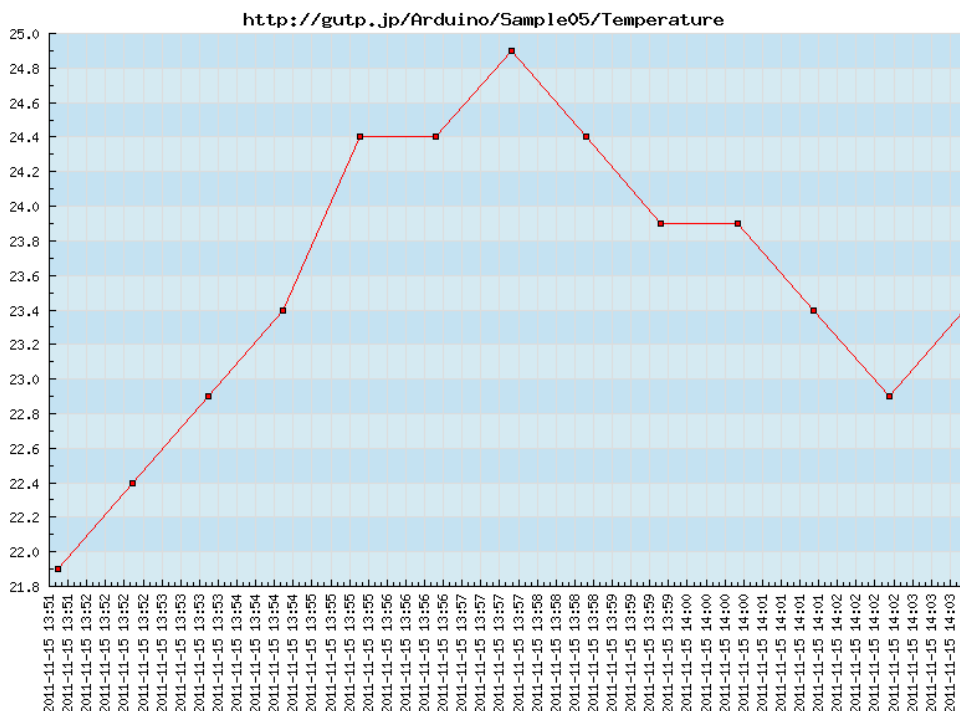


図 5: ポイント ID をクリックすると、直近のトレンドが、一覧表示される。

※ 注意事項

- ・ DHCP が有効であっても、NTP が無いネットワークの場合は、時計を合わせることができないため、正常に動作しません。
- ・ DHCP が有効であっても、外部ネットワークとの接続に HTTP PROXY サーバを介す必要がある場合は正常に動作しません。
- ・ 本キットは学習用のため、観測される温度センサの値や、照度センサの値は、正確ではありません。特に、温度センサは高めの値を出力します。これはボード自身の発熱の影響を受けるためです。

5. ファームウェアを変えずに使う：カスタマイズ設定編

本章は、学習キットのファームウェアを変えずに、パソコンを使ってネットワーク設定等を変更する方法を解説します。設定には、シリアル通信ターミナルソフトが必要になります。本マニュアルでは、**Putty** を使う場合を想定します。

5. 1. **Putty** のダウンロード

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html> より、**putty.exe** (図 6) をダウンロードします。このソフトウェアは、5.3 節以降で使います。



図 6: **putty** のアイコン

5. 2. 通信ボードとの **USB** 通信を確立する

通信ボードに電源を投入し、**USB** ケーブルでパソコンと接続してください(図 7)。



図 7: 通信ボード(学習キット)をパソコンと **USB** 接続する (**AC** アダプタも接続する)

USB を接続すると、パソコン側でデバイスドライバのインストールが行われます。そして、しばらくすると「デバイスを使用する準備ができました」というポップアップメッセージが出現します(図 8)²。

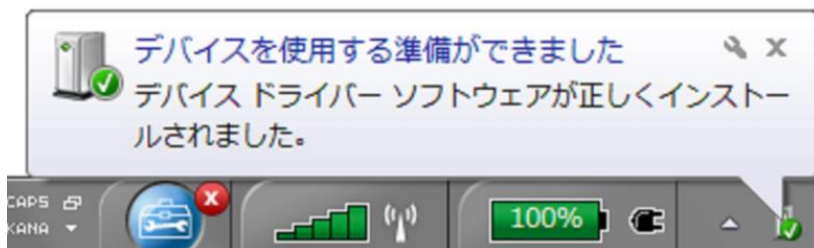
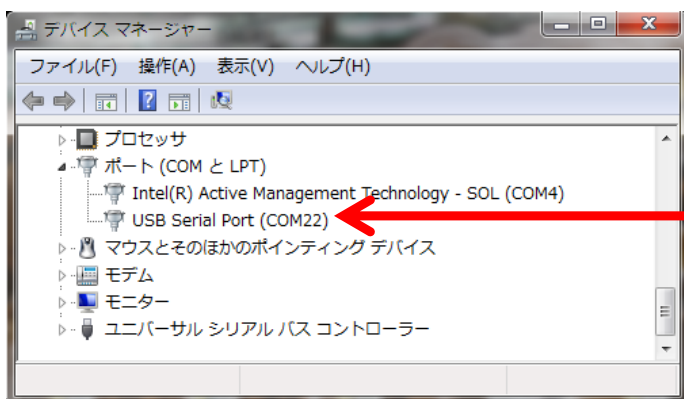


図 8: デバイス ドライバー ソフトウェアが正しくインストールされることを確認する

その後、この通信ボードに対して、COM 番号が割り当てられます。(COM5 あるいは COM14 などと表現されます)。この COM 番号が実際の設定時に必要になります。

COM 番号を調べるには、まず、デバイスマネージャー（「コントロールパネル」→「システムとセキュリティ」→「システム」→「デバイスマネージャー」と進みます）を開きます。そして、「ポート(COM と LPT)」のアイコンを開いて確認します。図 9 の例では、COM22 であることがわかります。



IEEE1888通信ボード接続
COMポート番号
がCOM22であることがわかる

図 9: USB 接続し、デバイスマネージャーで COM ポート番号を確認する
(この例では COM22)

² もしここで失敗した場合には、<http://www.ftdichip.com/FTDrivers.htm> から、適切なバージョンのソフトウェアをダウンロードしてインストールしてください。

5. 3. Putty で接続する

Putty を起動し、シリアル通信(Serial)を選択し、先ほど調べた COM 番号を入力します(図 10-a). その後、シリアル通信の詳細設定画面を開き、図 10-b のように、9600bps, 8 ビット, ストップビット長 1, パリティ無, フローコントロール無に設定してください³. その後、Open をクリックします.



(a) Serial 通信モードの選択

(b) シリアル通信の設定と接続開始

図 10: Putty のシリアル通信設定

接続に成功すると、図 11 のような画面が表示されます(何も表示されない場合は Enter キーを押してみてください). ネットワーク接続が無い場合は、DHCP や NTP リクエストに対し failure という結果が表示されます.

5. 4. コマンドとパラメータ設定

学習キットのデフォルトのファームウェアには、コマンド機能とパラメータ設定機能があります. まず、コマンドとしては、

conf: conf (設定) モードに移行します (データ送信は中断されます)

exit: conf (設定) モードから離脱します(データ送信が行われるようになります)

show: 現在の設定内容の閲覧

save: 現在の設定内容の保存

help または **?:** コマンド一覧, パラメータ一覧の表示

³ putty には、この設定を保存しておく機能もあります (本マニュアルでは割愛します).

が用意されています。例えば、`show` と入力して、`Enter` キーを押すと、図 12 のように、現在の設定内容が表示されます(実際に表示される設定内容は図 12 と同じとは限りません)。

Starting IEEE1888 Learning Kit ... が表示されれば成功！

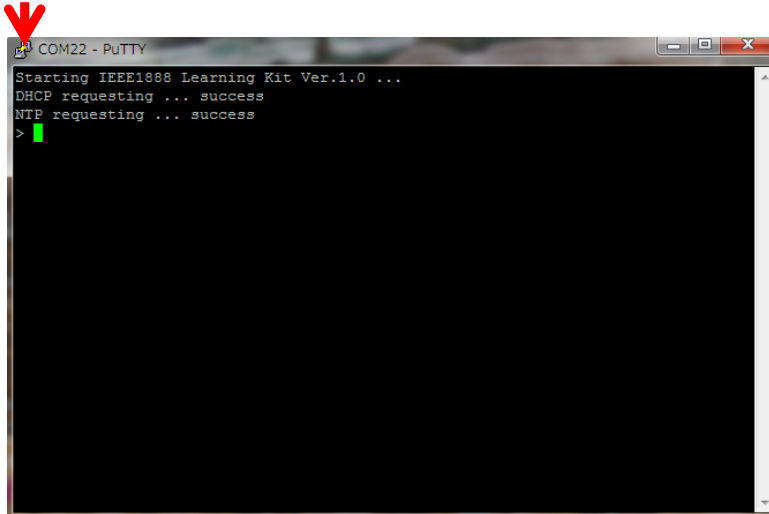


図 11: IEEE1888 通信ボードへの接続に成功した様子

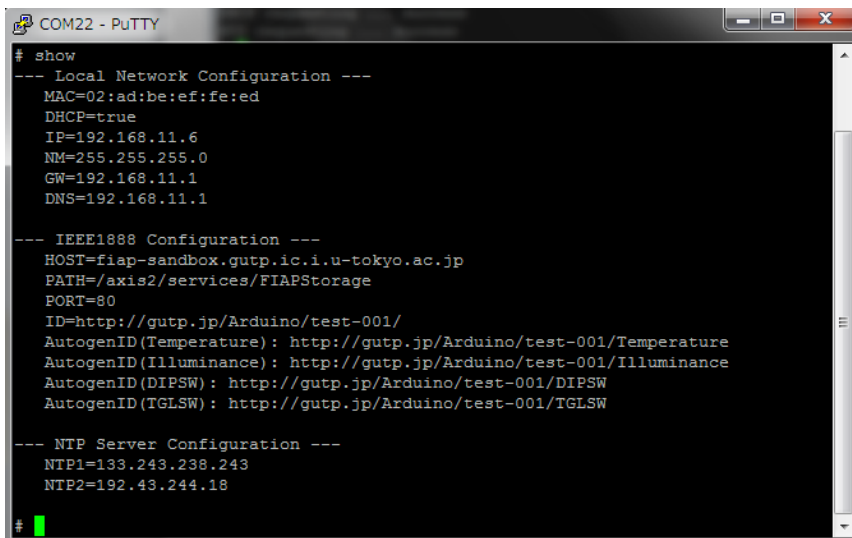


図 12: `show` コマンドによる設定内容の表示

一方、パラメータ設定は、`conf` コマンドを入力して設定モード(`#`による受付モード)に移行した上で、

パラメータ名=設定値 <<Enter>>

を入力することで行います。例えば、パラメータとしては、

- ・ **IP**: この機器の IP アドレス
- ・ **NM**: この機器が接続されているネットワークセグメントのネットマスク
- ・ **GW**: このネットワークのゲートウェイアドレス

などが用意されており、IP=192.168.11.55 などと入力することによって、その設定内容を更新することができます(MAC は読み取り専用パラメータなので設定変更できません)。

なお、パラメータ値を変更したら、**save** コマンドを実行し、通信ボード内部に設定内容を保存するようにしてください(図 13)。



図 13: パラメータ値の変更 と 設定内容の保存

6. 新規ファームウェアを開発する：プログラミング環境の整備

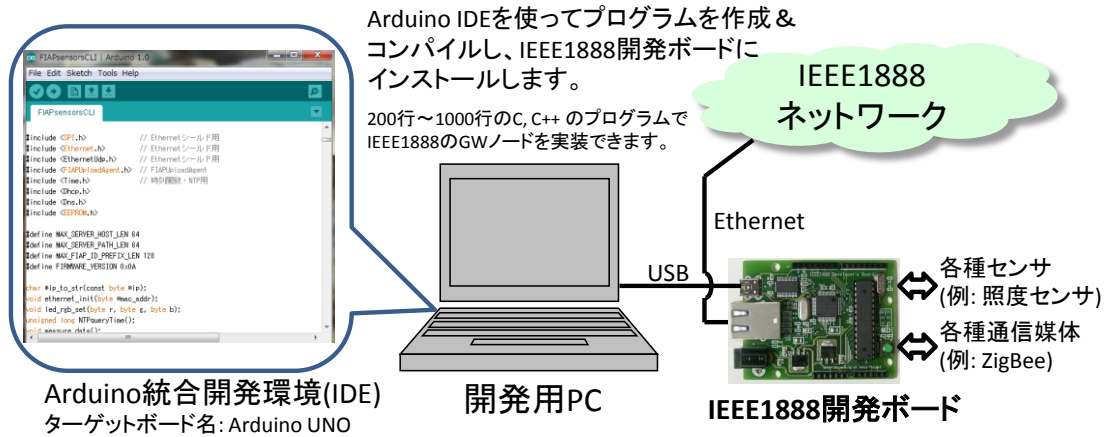


図 14: プログラミングのための環境設定(全体像)

プログラミング環境の設定は、次の4つのステップで構成されます。

- Arduino 統合開発環境(IDE)のインストール
- ライブラリの登録
- FTDI USB-Serial インタフェース・ドライバのインストール (オプション)
- ターゲットボード、シリアルインタフェースの設定

6. 1. Arduino 統合開発環境(IDE)のインストール

DVD-R に含まれている Arduino IDE (arduino-1.0)を、適当なディレクトリ(例えば C:¥Program Files¥arduino-1.0)にコピーしてください。Arduino IDE は、Arduino の公式ページ(<http://arduino.cc/en/Main/Software>)からも、ダウンロードすることができます。2012年06月現在、arduino-1.0.zip というファイルがダウンロードできるので、上記のように適当なディレクトリに展開してください。

※ arduino.exe へのショートカットを、デスクトップに作成しておくと、便利です。

(*) 特別な理由が無い限り、Arduino-1.0 開発環境をお使いください(Arduino-1.0.x 系は注意が必要です)。

6. 2. 必要なライブラリの登録

このボードの IEEE1888 通信対応化は、ソフトウェアによって実現されています。そのソフトウェアは FIAPUploadAgent という名前にライブラリ化されており、その設定を事前に施しておく必要があります。

ライブラリの設定は、Windows の場合は、ドキュメントフォルダの Arduino フォルダに対して行います。ここでは、FIAPUploadAgent ライブラリの他に Time ライブラリの設定方法についても記載します。IEEE1888 通信は、FIAPUploadAgent ライブラリがあれば実現できますが、時刻情報の管理に Time ライブラリがあると便利です。またネットワーク通信に DHCP や DNS 機能があると、利便性がさらに向上します。

(注) Arduino-1.0 プラットフォームでは、EthernetDHCP や EthernetDNS ライブラリの追加は不要になりました。

▲FIAPUploadAgent ライブラリ (arduino-1.0 対応)

Arduino 向けのオープンソースの IEEE1888 通信ライブラリ (WRITE クライアント専用) です。同梱の DVD-R に FIAPUploadAgent というフォルダ名で含まれています。

▲Time ライブラリ

Arduino 上で、時計を Unixtime で管理するためのライブラリです。このライブラリは、同梱の DVD-R にも Time というフォルダ名で含まれていますが、オリジナルは <http://www.arduino.cc/playground/Code/Time> から入手できます。

※ ※ ※

ライブラリは、フォルダとなって提供されているので、入手後、それらをユーザ・ドキュメントの Arduino フォルダの libraries フォルダ(なければ作成のこと)に配置してください(図 15)。その後、Arduino を再起動すると、これらのライブラリは自動的に読み込まれます。図 16 のように、メニューで、Sketch→ Import Library ... と進み、ライブラリが登録されていることを確認して、登録完了です。

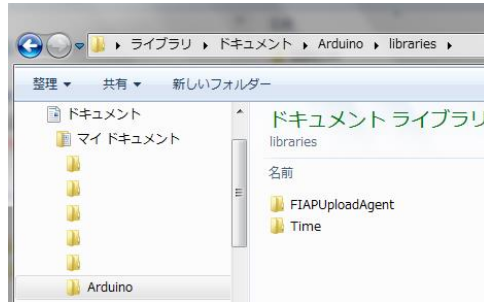


図 15: Arduino の libraries フォルダへライブラリを設定

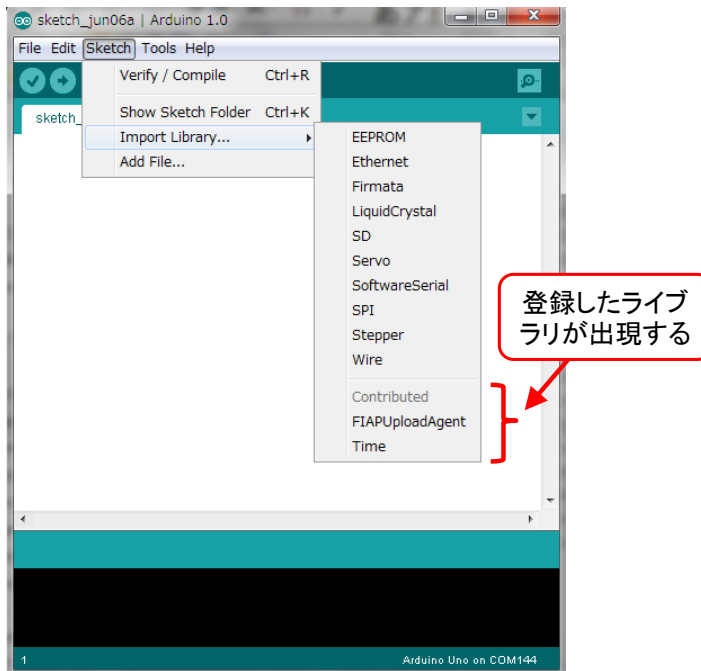


図 16 : ライブラリ登録の確認

6. 3. FTDI USB-Serial インタフェース・ドライバのインストール (オプション)

IEEE1888 通信ボードを開発用 PC に USB で接続すると、ドライバのインストールが行われます。成功すると、図 17 のようにポップアップが出現しますが(初回のみ)、もし、そうならない場合は、下記 URL にアクセスし、FTDI ドライバのダウンロードおよびインストール

を行ってください(OS によって、インストールすべき内容は異なりますので具体的な指示は、この Web サイトに従ってください).

<http://www.ftdichip.com/FTDrivers.htm>

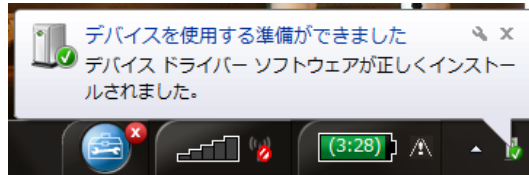


図 17 : ドライバのインストールに成功すること

6. 4. ターゲットボードの選択とシリアルポートの選択

6. 4. 1. ターゲットボードの選択

Arduino IDE のメニューで、Tools → Board と進むと、図 18 のように、様々なターゲットボードの種類が出てきます。ここでは、ターゲットボード名に **Arduino Uno** を指定してください(2011 年 11 月版では、**Arduino Duemillanove** でしたが、2012 年 6 月版より変更されました).

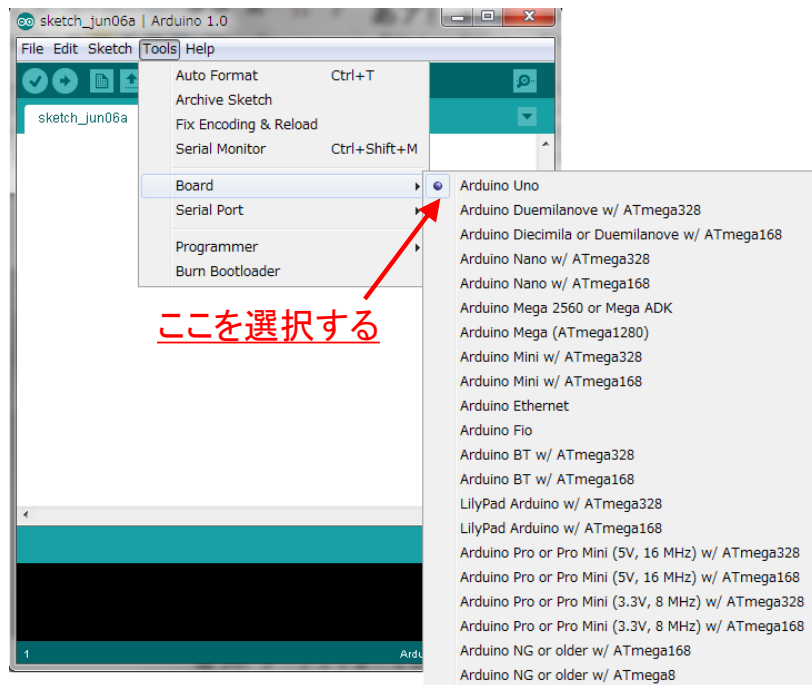
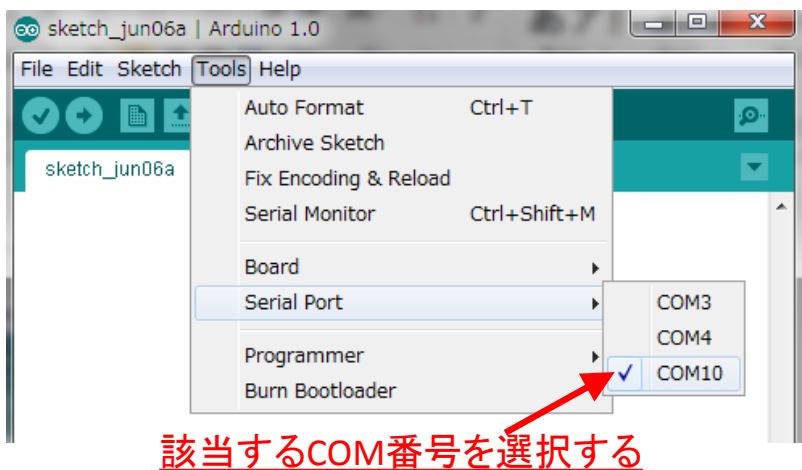


図 18: ターゲットボードの選択

6. 4. 2. シリアルポートの選択

ボードがパソコンに正しく接続されると USB シリアル通信インタフェースに対し、COM ポートが割り当てられます。Arduino IDE のメニューで、Tools → Serial Port と進む(図 19)と、COM が出現するので、その COM を選択しておいてください。図 19 の場合は、COM 10 になっています。



該当するCOM番号を選択する

図 19: 該当する COM を選択する

※ ※ ※

以上で、プログラミング環境の整備は完了です。

7. ファームウェアを作成する: サンプルプログラムの書込み

7. 1. 付属のサンプルプログラムについて

サンプルプログラムは、ライブラリを設定したときに、Arduino IDE に同時に読み込まれています。IEEE1888 通信関係のサンプルプログラムは、図 20 のように、メニューから File --> Examples と進むと、FIAPUploadAgent という項目が現れ、その FIAPsensorsCLI になります。

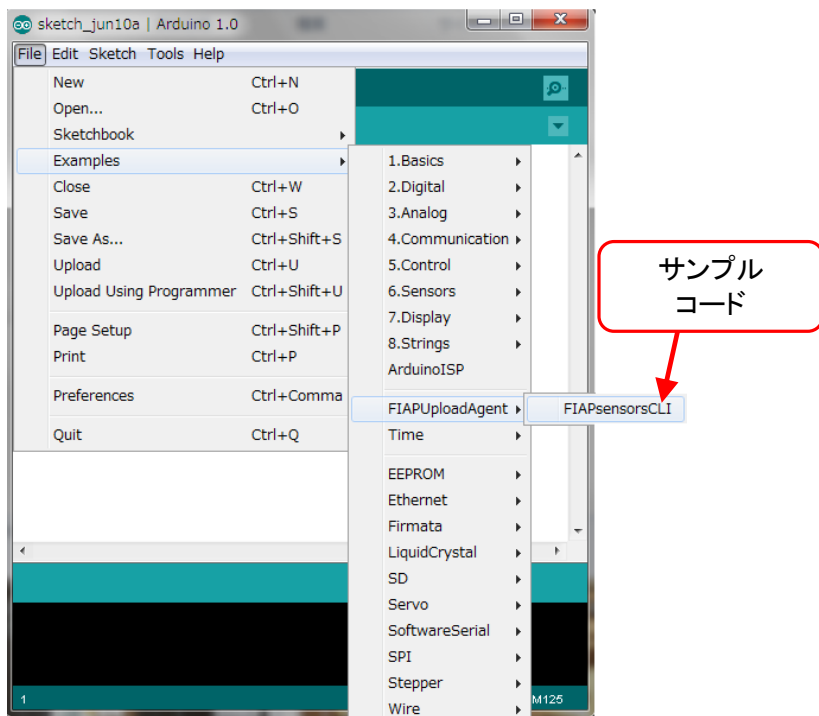


図 20: サンプルコードの読み込み方法

FIAPsensorsCLI は FIAPUploadAgent を利用するプログラムで、学習キット用のシールドを使う場合を前提に組まれています。コマンドラインインタフェース(CLI)を備え、プログラミング環境が無い人でも、各種通信設定(IP アドレスやポイント ID の設定など)を行えるようになっています。なお、DHCP および DNS も有効にできます。

以下では、FIAPsensorsCLI を例に上げ、プログラムを書込む方法を記載します。まず、図 20 を参考に、FIAPsensorsCLI のサンプルコードを読み込んでおいてください。

7. 2. ビルドとインストール

パソコンと IEEE1888 通信ボードを接続し、図 21 のように Arduino IDE の アップロード ボタンをクリックします。ソフトウェアのビルドが実行され、成功すると、生成されたバイナリ イメージが Arduino 本体に書き込まれます。図 22 のように、Done uploading と表示されたら、ビルドおよびインストールが完了です。

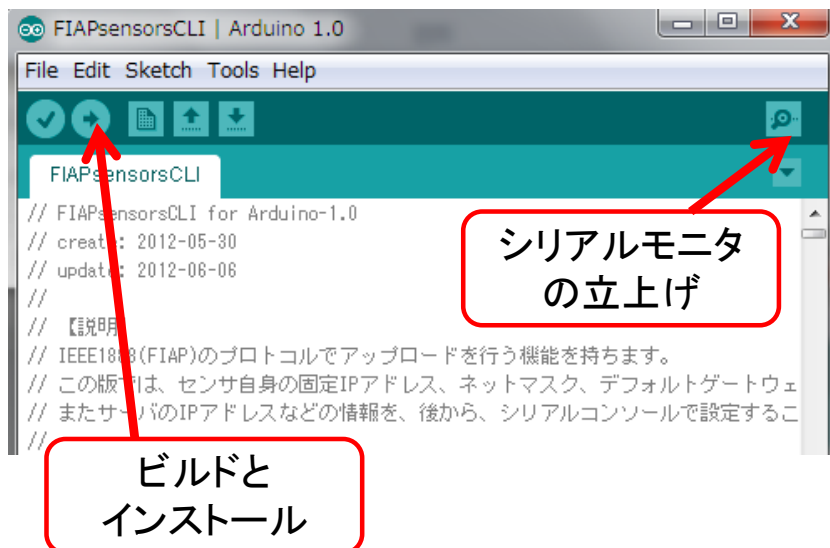


図 21: ソフトウェアの書込みと動作確認方法

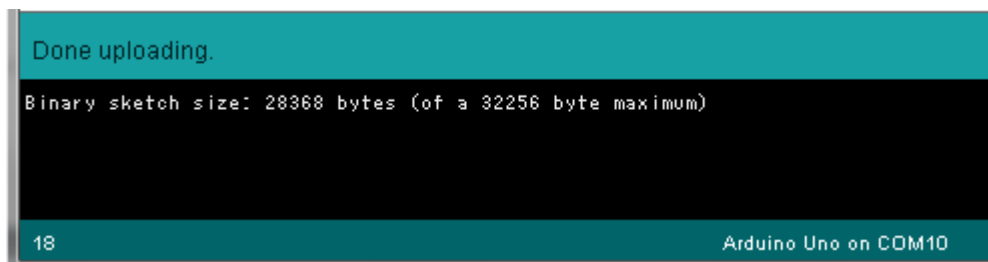


図 22: 「Done uploading」 -- 書込みが成功したことを示すサイン

ファームウェア書込みが完了すると IEEE1888 通信ボードは自動的に再起動します。この際に想定されたネットワークに接続されていれば、センサからの観測データを、サーバにアップロードします。

図 21 に示す方法で Arduino IDE のシリアルモニタを起動すれば、5.4 節で述べた設定インタフェースに接続することができます (図 23)。

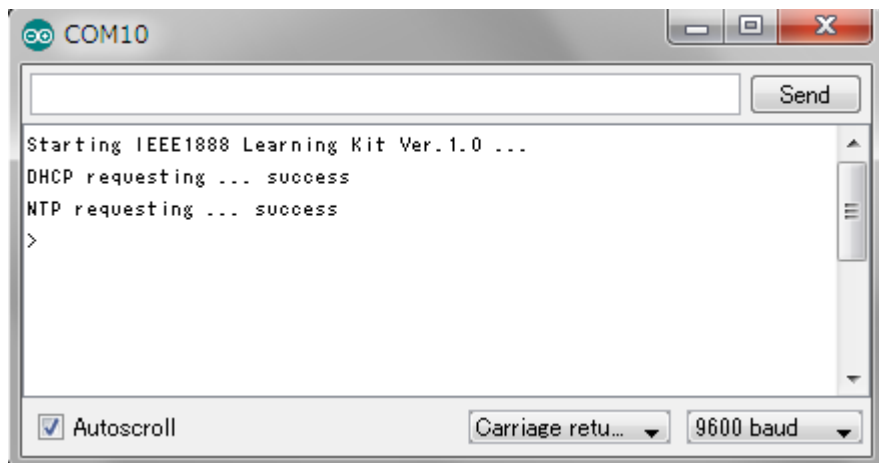


図 23: シリアルモニタの画面

8. IEEE1888 通信機能のプログラミング方法

ここでは IEEE1888 通信プログラミングの方法を述べます。Arduino IDE をオープンし、次のように記述し、必要なファイルをインクルードしてください。

```
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <FIAPUploadAgent.h>
#include <Time.h>
#include <Dhcp.h>
#include <Dns.h>
#include <EEPROM.h>
```

IEEE1888 通信に限らず、ネットワーク接続を行うためには、SPI.h と Ethernet.h のファイルが必要になります。FIAPUploadAgent.h が IEEE1888(WRITE クライアント)の通信ライブラリを参照します。Time.h をインクルードすることによって、通信ボード上で、Unix timestamp での時刻管理が可能になります。EthernetUdp.h は、このボードを、ネットワーク上の NTP(時刻)サーバと UDP 通信させ、時刻同期を行うため使います。Dhcp.h を読み込んでおくと、DHCP サーバから IP アドレスを取得できるようになります。また、Dns.h を読み込んでおくことで、DNS 解決を行うことができるようになります。EEPROM.h を使えば、ボード上の変数を、不揮発性メモリに記憶しておくことが可能になります。

IEEE1888 の通信を行うには、まず、FIAPUploadAgent のインスタンスを作成する必要があります。その作成方法は、

```
FIAPUploadAgent  FIAP;
```

です。これは、どの関数にも属さないグローバル領域に書いておきます。

次に、setup 関数内で、

```
FIAP.begin( ... );
```

を実行しライブラリを初期化します。

loop 関数内で、

```
int ret=FIAP.post( ... );
```

を実行させることでサーバにデータを送信します。

ここで、setup 関数や loop 関数は、Arduino が標準で装備している関数で、setup 関数は起動時に一度だけ呼ばれ、loop 関数はその後、何度も呼ばれます。

FIAPUploadAgent が提供するメソッドの種類と、その引数や返り値の意味定義は、表 2 および表 3 を、ご参照ください。

表 2: FIAPUploadAgent の API

メソッド名	引数	型	意味
begin	server_host	const char*	サーバのホスト名を指定(HTTP ヘッダの Host パラメータに利用) “fiap-sandbox.gutp.ic.i.u-tokyo.ac.jp ” などへのポインタ
	server_path	const char*	IEEE1888 サービスを提供している HTTP サーバのパス “/axis2/services/FIAPStorage” などへのポインタ
	server_port	uint16_t	サーバの TCP ポート番号 通常は 80 (HTTP) を指定する
	fiap_id_prefix	const char*	ポイント ID の共通部分をプレフィックスとして指定する “http://gutp.jp/Arduino/demo/” などへのポインタ
	戻り値	void	戻り値は無し
post	v	struct fiap_element*	サーバに送信するセンサデータ(fiap_element 構造体)の配列へのポインタ (表 2 参照)
	esize	uint8_t	fiap_element 構造体の個数
	戻り値	int	サーバとの通信状態を応答 成功: FIAP_UPLOAD_OK TCP 接続失敗: FIAP_UPLOAD_CONNFAIL HTTP サーバエラー: FIAP_UPLOAD_HTTPERR IEEE1888 エラー応答: FIAP_UPLOAD_FIAPERR

表 3: fiap_element 構造体

変数	型	意味
cid	const char*	ポイント ID のポストフィックス (センサごとに個別に指定される) “Temperature” や “Illuminane” 等へのポインタ ポイント ID は、表 1 の fiap_id_prefix と組み合わせて生成される
value	char*	このポイントに結び付けられて送信される値(文字列)へのポインタ
year	uint16_t	送信する値の時刻 (年の部分)
month	uint8_t	送信する値の時刻 (月の部分) 1 - 12
day	uint8_t	送信する値の時刻 (日の部分) 1 - 31
hour	uint8_t	送信する値の時刻 (時の部分) 0 - 23
minute	uint8_t	送信する値の時刻 (分の部分) 0 - 59
second	uint8_t	送信する値の時刻 (秒の部分) 0 - 59
timezone	char*	送信する値の時刻 (タイムゾーン表記) “+00:00” などへのポインタ

9. 周辺装置とのインタフェース(ハードとソフト)

9. 1. ハードウェア

IEEE1888 通信ボードのピンコネクタ配置は、図 24 に示す通り、Arduino Duemilanove や Arduino Uno と互換になっています(最近追加された IOREF, SDA, SCL 端子はありません)。ただし、pin10~13 は、TCP/IP コントローラで利用されているため、インターネット接続を行う場合は、これらのピンの利用には注意が必要です。また、pin0~1 は、USB シリアルインタフェースで利用しているため、USB シリアルポートを利用する場合は、これらのピンも注意して利用する必要があります。

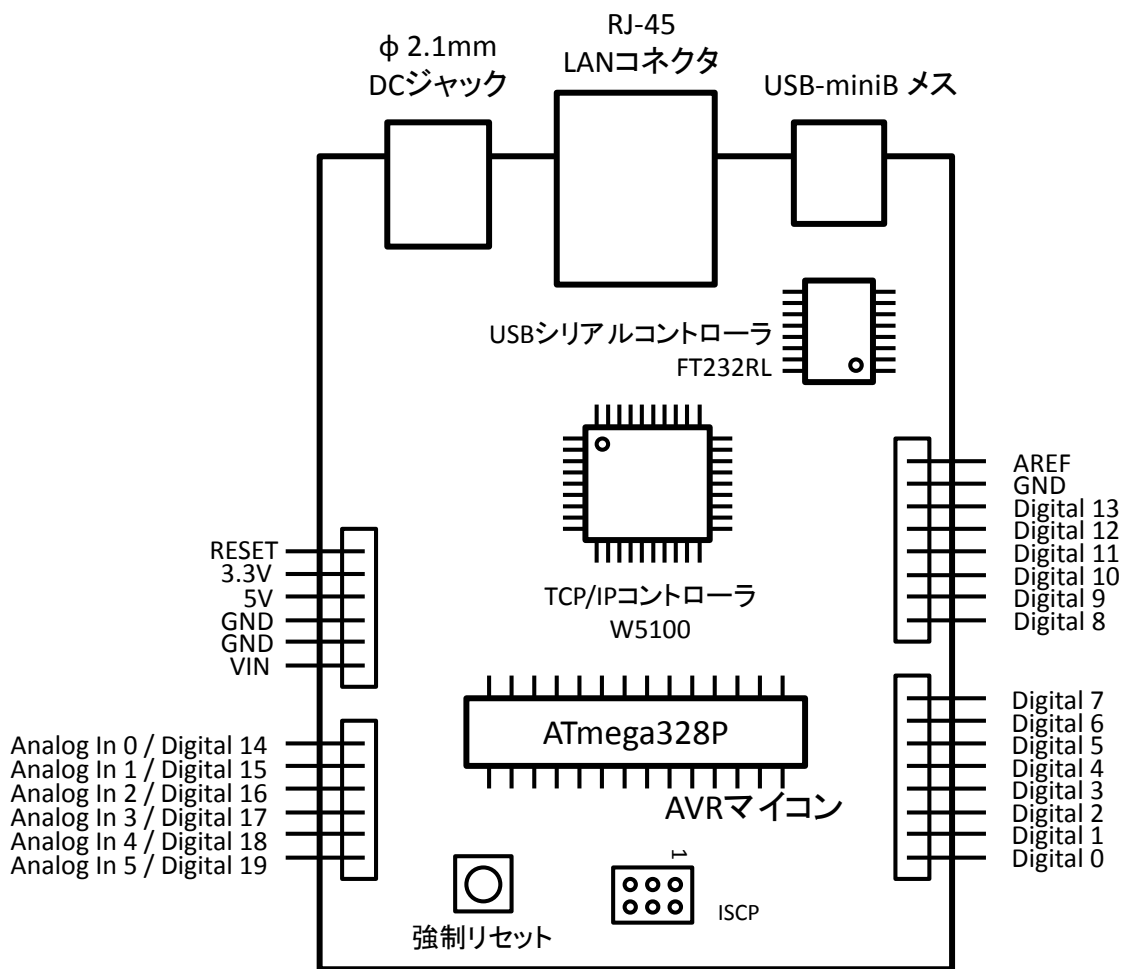


図 24: IEEE1888 通信ボードのピン配置図

9. 2. ソフトウェア

図 24 の各ポートへのソフトウェアでのアクセス方法(API: Application Programming Interface)は、各種用意されています(<http://arduino.cc/en/Reference/HomePage> を参照). 代表的なものとして、

```
pinMode(...); // デジタルポートの入出力モードを指定
digitalRead(...); // 指定したデジタルポートの HIGH/LOW 状態を取得
digitalWrite(...); // 指定したデジタルポートに状態(HIGH or LOW)を設定する
analogRead(...); // 指定したアナログポートの電圧を AD 変換する(10 ビット)
analogWrite(...); // 指定したデジタルポートから PWM 波形を出力する (8 ビット PWM)
Serial.XXX(...); // USB シリアルとの通信 (ATmega328P のシリアル通信モジュール利用)
```

SoftwareSerial.XXX(...); // 汎用デジタルポートをシリアル通信ポートとして利用する
などがあります。以下、これらの利用方法を記載します。

9. 2. 1. デジタル値の読み込み

Digital 3 をデジタル入力ポートとして使用する場合、

```
pinMode(3, INPUT);
```

と、setup 関数内などで宣言しておけば、

```
digitalRead(3); // HIGH or LOW を返す
```

と記述することで、そのポートの値(HIGH or LOW)を読むことができます。

9. 2. 2. デジタル値の設定

Digital 4 をデジタル出力ポートとして使用する場合は、

```
pinMode(4, OUTPUT);
```

と、まず宣言しておきます(setup 関数内などで).

その後、LOW 出力をする場合は、

```
digitalWrite(4, LOW);
```

とし、HIGH 出力する場合は、

```
digitalWrite(4, HIGH);
```

のように、記述します。

9. 2. 3. アナログ値の読み込み

Analog In 2 をアナログ入力ポートとして使用する場合は、

```
analogRead(2); // 0~1023 を返す
```

のように記述します。AREF(=通常 5V)の電圧に対する電圧比を、0~1023 で返します。1024 が AREF の電圧に対応します。

9. 2. 4. アナログ値(PWM)の出力

アナログ出力は、正式には PWM 比(0~255)でのデジタル出力となります。PWM 出力ができるのは、Digital 3, 5, 6, 9, 10, 11 です。

```
analogWrite(3, 128);
```

とすると、Duty 比 50%の PWM 信号を出力します。

9. 2. 5. USB シリアルとの通信

IEEE1888 通信ボードの Digital 0, Digital 1 は、それぞれ TX, RX として機能し、これらは 1k Ω の抵抗を介して、FT232RL(USB シリアル通信チップ)に接続されています。従って、Arduino に標準装備されている Serial インスタンスを使って、USB シリアルとの通信ができます。

まず、通信速度の設定のため、

```
Serial.begin(9600);
```

を、setup 関数内などで宣言します。

※ USB シリアルへの書出し方法

USB シリアルに書き出す(つまり、接続されているコンピュータに送り出す)には、

```
Serial.println(“Hello World!!” );
```

と記述します。

(* println メソッドは最終文字の後、自動的に改行文字を出力します。print メソッドを使えば、そのような改行文字は出力されません。

※ USB シリアルからの読み込み方法

USB シリアルから送られてきた文字を読み出す(つまり、接続されているコンピュータから送られてきた文字を読み取る)には、

```

while(Serial.available()){
    char c = Serial.read();
    ... 処理 ...
}

```

とします。

9. 2. 6. ソフトウェア・シリアル通信の設定

汎用の入出力ポートを使ってシリアル通信を行うために、**SoftwareSerial** ライブラリが標準で装備されています。Arduino-1.0 プラットフォームでの **SoftwareSerial** は、従来版と比べて機能および性能が向上し、**Serial** とほぼ同じように使用することができます。以下が、その使い方です。

```

// XBeeReader 1.0 for Arduino 1.0 development platform
#include <SoftwareSerial.h> // XBEE からの読取り用

// ピン番号の設定 (ハードウェア依存)
// XBEE シールド (インタフェース拡張ボード Xbee 実装タイプ)
#define XBEE_TX 14
#define XBEE_RX 15

// Xbee モジュールとの通信シリアル
SoftwareSerial xbeeSerial=SoftwareSerial(XBEE_RX, XBEE_TX);

// 初期設定ルーチン
void setup()
{
    // I/O 初期化
    Serial.begin(9600); // USB シリアルとの通信速度を 9600bps に設定
    pinMode(XBEE_RX, INPUT); // RX を入力モードに設定
    pinMode(XBEE_TX, OUTPUT); // TX を出力モードに設定
    xbeeSerial.begin(9600); // XBEE との通信を 9600bps で行う
}

// メインループ
void loop()
{
    while(1){
        char c;
        while(xbeeSerial.available()){
            c=xbeeSerial.read();
            Serial.print(c);
        }
    }
}

// end of code

```


10. 2. ZigBee(XBee)モジュールの接続例

IEEE1888 通信ボードに ZigBee モジュールを接続することで、IEEE1888 と ZigBee の変換 GW を実現することができます。その時に用いられる回路構成の例を図 26 に示します。

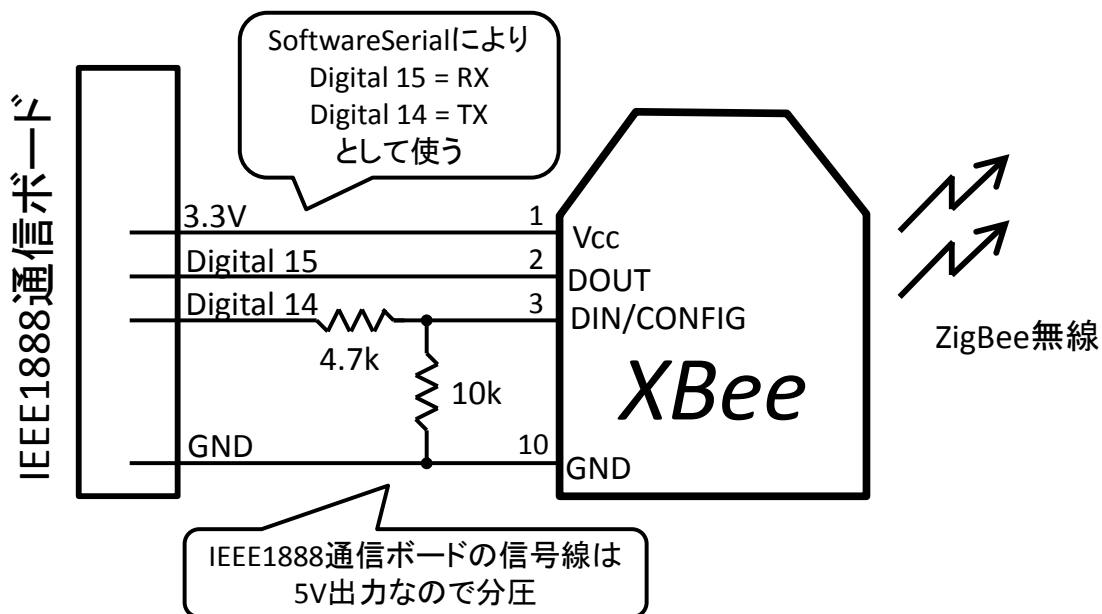


図 26: ZigBee モジュール(XBee)の接続例

10.3. RS232C ドライバの接続例

IEEE1888 通信ボードに RS232C ドライバを接続することで、IEEE1888 と RS232C の変換 GW を実現することができます。その時に用いられる回路構成の例を図 27 に示します。

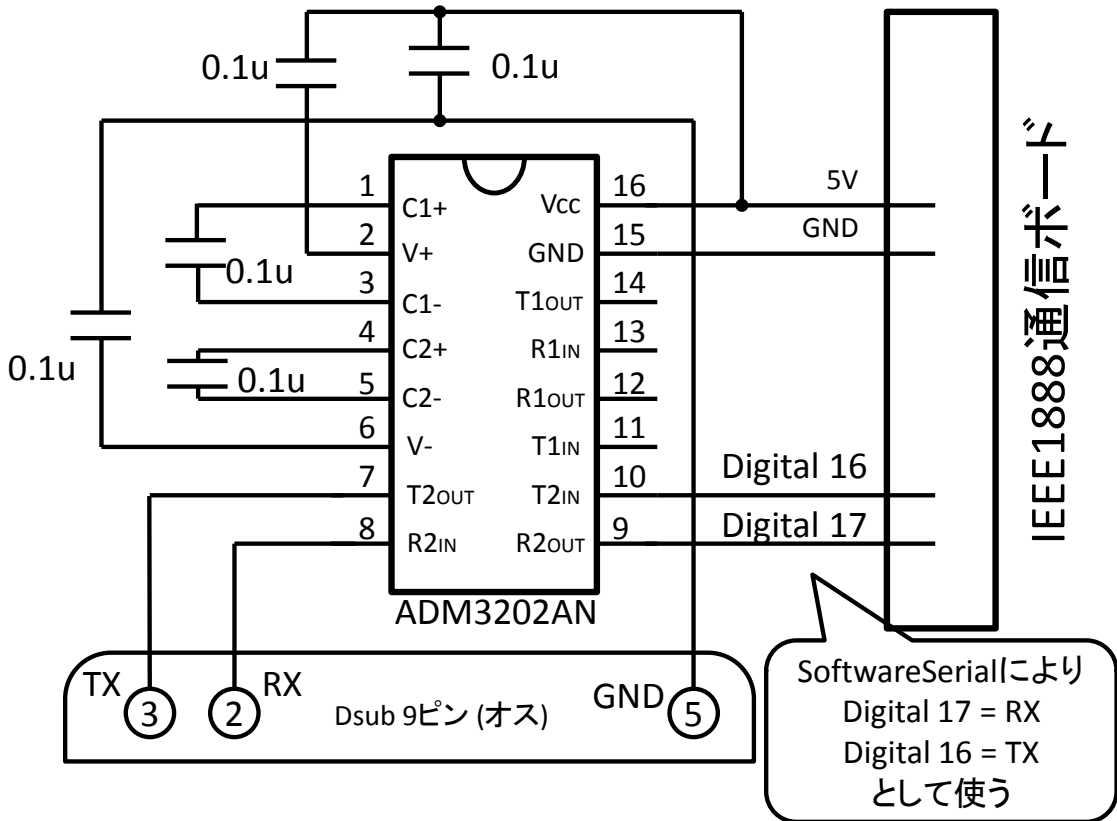


図 27: RS232C ドライバの接続例

10.4. パルス出力端子の接続例

IEEE1888 通信ボードに、電力メータなどからのパルス出力信号を取り込むための、回路構成例を図 28 に示します。この回路例は、オープンコレクタ出力(フォトカプラなど)でパルス出力される機器へのインタフェース回路となっています(パルス出力の形態に応じて、この回路は異なってきますので、利用においては注意が必要です)。

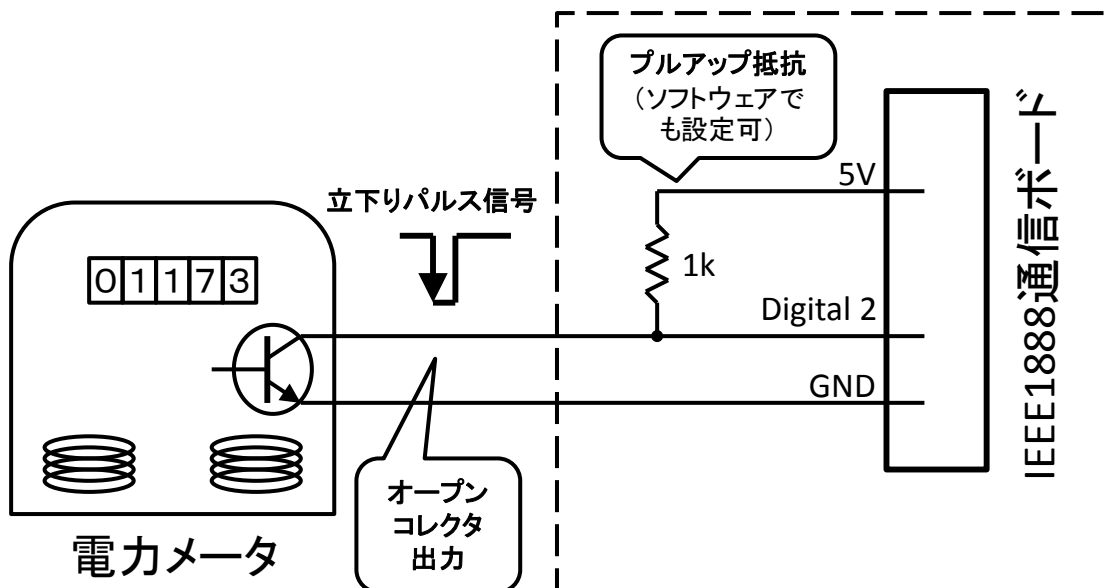


図 28: パルス出力信号とのインタフェース回路例

ここで接続している Digital 2 ポートは、割り込み生成が可能なポートとなっています。次のプログラムによって立下りパルスをソフトウェアで扱えるようになります(もちろん、ピンの状態をポーリングすることで立下りを検出する方法もあります)。

```
void setup(){
  Serial.begin(9600);
  // 0 = digital2; 1 = digital3; パルス検出→ pulse_detect 関数を呼ぶ
  attachInterrupt(0, pulse_detect, FALLING);
}
void pulse_detect(){
  Serial.println( "Pulse Detected!!" );
}
```

10. 5. シールドの実装方法について

Arduino シールドの設計は自由自在です。Arduino のピン配置の関係で、通常のユニバーサル基板は搭載しにくいですが、Arduino 向けのユニバーサル基板(例えば、サンハヤト製の UB-ARD01)を購入すれば、様々なシールドを実装することができます。「しっかりした基板にしたい」というご要望があれば、本マニュアル末尾の連絡先までメールをお送りください。IEEE1888 の普及にとって重要な案件と判断されれば、積極的に基板化を進めてまいります(その際にプロトタイプが既にあるとスムーズです)。

■お問合せ先

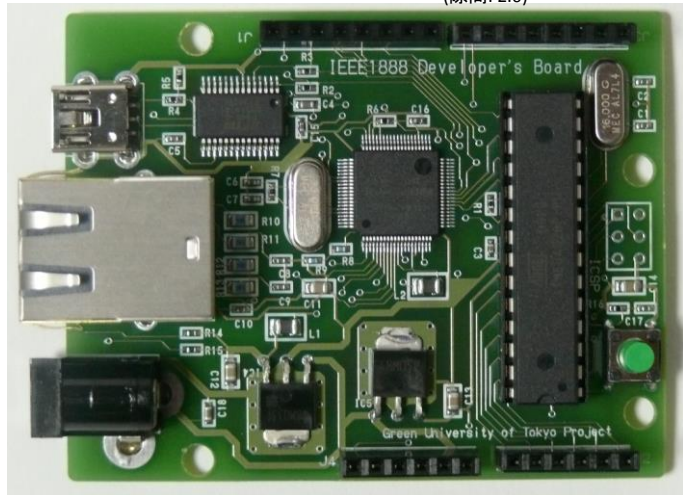
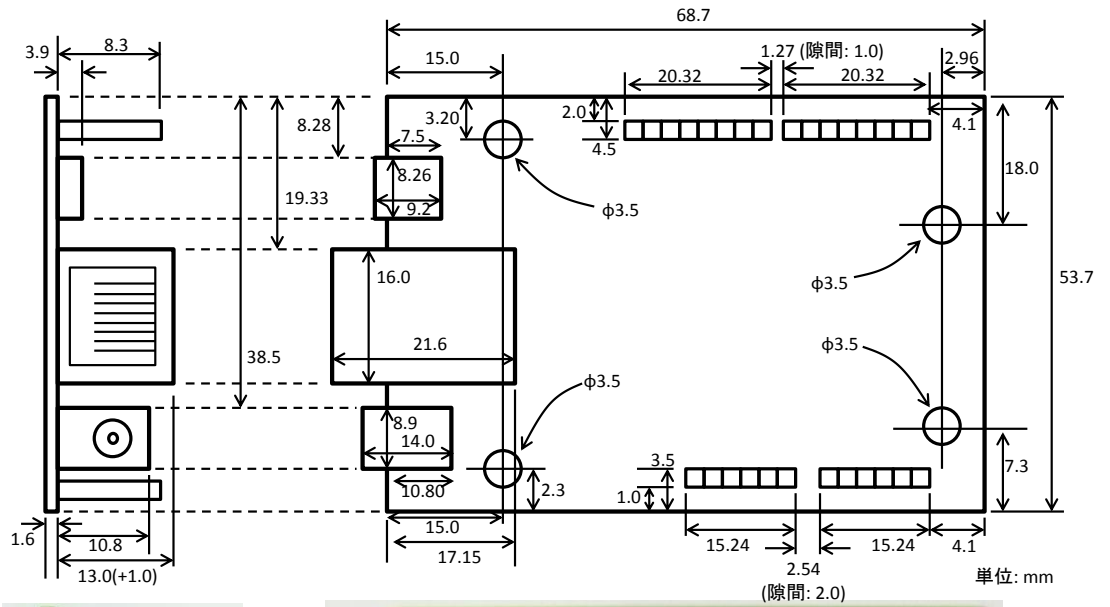
※ 技術的な内容

東京大学 落合秀也 (ochiai@vdec.u-tokyo.ac.jp)

※ 製造販売に関する内容

フタバ企画 (info@futaba-kikaku.jp)

付録 A: IEEE1888 開発ボード 外形図



IEEE1888開発ボード 外形図
2013年02月19日