[APNOTE10]

リモート温度・赤外線センサーデータを XBee デバイス経由で収集、集計グラフ表示や在宅監視を行う

ABS-9000 DeviceServer
APNOTE10 Rev A.1.0
2010/03/31

オールブルーシステム (All Blue System)
ウェブページ: www.allbluesystem.com
コンタクト:contact@allbluesystem.com

1　　イントロダクション

屋外や屋内に設置した複数のセンサーデバイス(温度、赤外線(人体)センサーを内蔵)の計測データを定期的に無線でサーバーに送信することで集計グラフの作成や、センサーからのイベントを検出することでリアルタイムの在宅監視を行うシステムを構築する例を説明します。

各センサーデバイス内では、マイクロコントローラ(Atmega644P) によって定期的なサンプリングと、サーバーからのリクエストパケットによってリモートコマンド実行が行われます。サーバーとセンサーデバイス間の通信には XBee[1] デバイスを使用します。サンプリングデータやセンサーデバイスのリモートコマンドなどはすべて XBee モジュールのデータパケット中に格納されて送受信されます。マイクロコントローラ上で動作するプログラムは、オールブルーシステム が提供している “TDCP (Tiny Device Control Program)” を使用しています。TDCP は DeviceServer のライセンスと共に使用する場合にはフリーで製品に搭載できます。(*1 参照)

XBee デバイスは、Digi International Inc. 社製の IEEE 802.15.4 RF モジュールを使用します。サーバー側とセンサーデバイスの両方で使用しています。

センサーデバイスで計測された温度・赤外線センサーのサンプリングデータは、無線(XBee経由)で サーバーPC (DeviceServer) に送られて内部のデータベースに格納されます。
その後 クライアントPC や Webページ(Flash)アプリケーション)、電子メールなどからのリクエストでデータベース中のセンサーデータを集計してレポートやグラフを作成します。クライアントからは、任意のタイミングでリモートのセンサーデバイスに対して、マニュアルサンプリングや動作モードの変更、ポート値の操作を行うこともできます。

センサーデバイス側で赤外線センサーデータ入力カウント値が、設定値を超えたときにイベントパケットをサーバーに送信して処理を行うこともできます。

(*1) TDCP プログラムについて

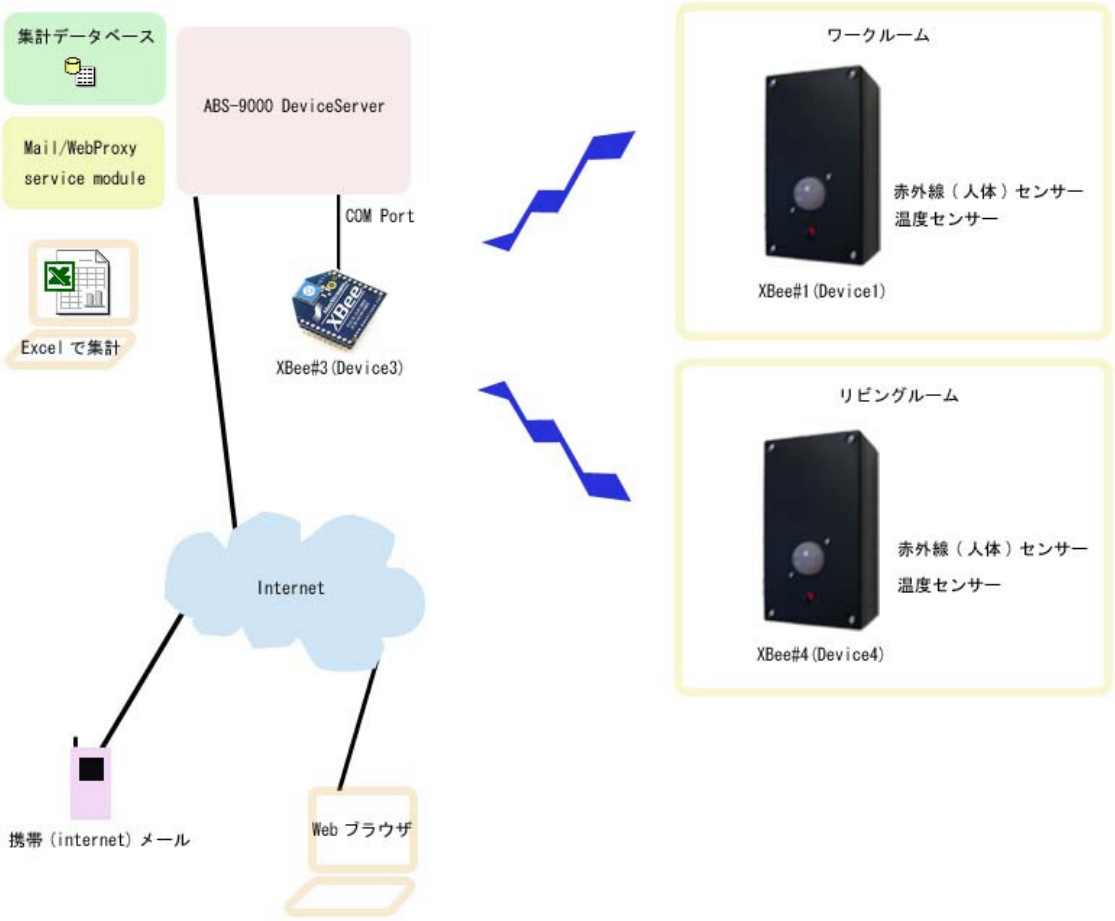
センサーデバイス中のマイクロコントローラ上で動作させるプログラムはオールブルーシステムが提供している “TDCP (Tiny Device Control Program)” を搭載しています。DeviceServer のライセンスと共に使用する場合にはフリーで製品に搭載できます。TDCP 詳細とセンサーデバイス中に使用したベースボードのマニュアルがフリーで公開しています。下記のWebページを参照して下さい。

TDCP 紹介とファームウエアダウンロード　http://www.allbluesystem.com/

TDCP マニュアル　http://www.allbluesystem.com/TDCP/TDCP_Users.pdf

1.1　　システム全体構成図

[1] XBee XBee® and XBee□PRO® are registered trademarks of Digi, Inc.



XBee デバイス名 (Device1,Device3,Device4)について

テスト環境で使用した上記の XBee デバイスのノード名は Device1,Device3,Device4 になっています。Device2 は使用していません。このマニュアル中の説明はすべてこれらのノード名を使用していますのでご注意ください

システムは複数のセンサーデバイスと、サーバーから構成されています。
センサーデバイスは監視対象の各部屋に設置されています、また赤外線センサーと温度センサーを内蔵して、定期的にサーバーにサンプリングデータをサーバーに送信しています。

サーバーでは、各センサーデバイスから送信されたサンプリングデータを内部のデータベースに蓄積してクライアントからのリクエストに応じて集計を行います。サーバーに接続した PC から集計結果を利用して、エクセルのグラフを作成したり、遠隔のWeb ブラウザから任意の日付の集計グラフを表示できます。携帯電話のメール(Internetメール)から集計条件を指定して、センサーデータの集計結果をメールで受信することもできます。

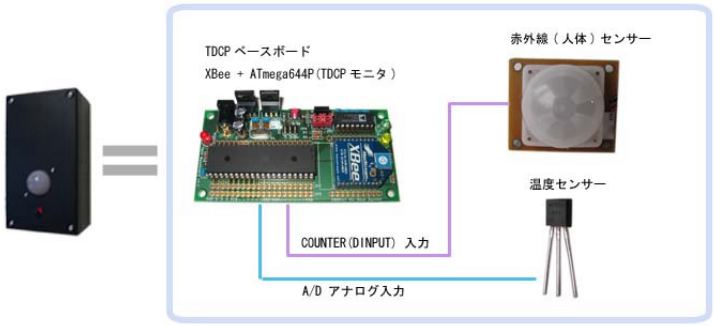
赤外線センサーの検出値(人体の動きを検出)が予め決められた設定値を超えた場合に、携帯電話のメール(Internetメール) に警報を送信できます。また反対に、一定期間赤外線センサーに一切 検出がなかった場合に、警報メールを送信できます。

センサーデバイスとサーバーにはそれぞれ XBee 無線モジュールが接続されていて、センサーデータのサンプリング

値の送信やサーバーからのセンサーデバイス操作などはすべて無線でリモートコントロールされます。サーバーとセンサーデバイス間をネットワークケーブルやシリアルケーブルで接続する必要がないので、設置場所が自由に決めることができ設置場所の変更も簡単になります。

センサーデバイスとサーバー間の無線通信がエラーになった場合でも、TDCP が持つパケット通信のリトライ機能によって通信安定性を向上させています。またシステムを構成するセンサーデバイスの一部がダウンまたは一時的な通信不能になった場合でも、システム全体の動作には影響せず該当センサーのサンプリング集計値のみに影響が限定されるようになっています。サーバー自身が一時的にダウンした場合でも、特別な設定操作を行わずにサーバー復帰後は自動的にサンプリングや監視状態を継続できます。

1.2 センサーデバイス構成図

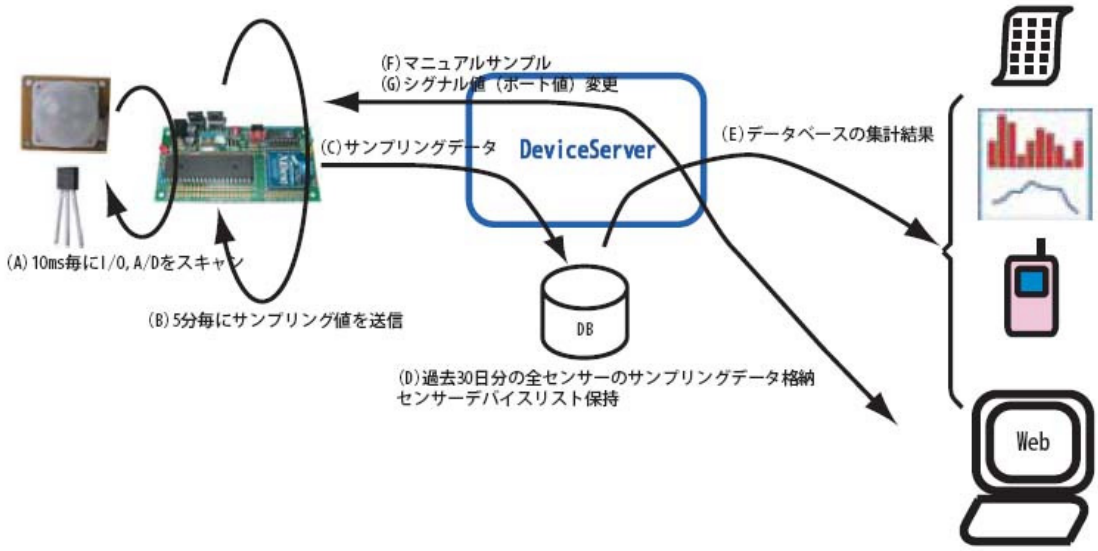


センサーデバイスはマイクロコントローラとXBee デバイスから成る TDCP ベースボードに、赤外線センサーと温度センサーが接続された構成になっています。

TDCP ベースボードの A/D 変換入力ポートに温度センサーが接続され、デジタル入力ポートに赤外線センサーが接続されています。赤外線センサーは人体の動作を検出したときに断続的に ON-OFF を繰り返す仕様で、これをデジタル入力ポートからカウンタ入力値として取り込みます。

これらのセンサーデータを取り込むために、TDCP ベースボードで動作する TDCP プログラムは app_mode= 8 で動作しています。TDCP のモードについては、"TDCP ユーザーマニュアル"
(http://www.allbluesystem.com/TDCP/TDCP_Users.pdf) を参照して下さい。

1.3 センサーデータの処理フロー



システム内で、センサーデータやイベントがどうのように処理されていくかの動作フローを説明します。

センサーデバイスでは、10ms 間隔でデジタル入力ポート(赤外線センサー)と A/D 変換入力を監視しています。デジタル入力ポートではチャタリングを除去した後、有効なデジタル入力の変化数をカウンタ値してインクリメントしています。(A)

サンプリング間隔の 5 分ごとに、現在のカウンタ値(赤外線センサーの変化数)と A/D 変換入力値(現在の温度センサー値)をがサンプリングデータとして取得します。(B)

サンプリングデータは センサーデバイス内の XBee モジュールからデータパケットとして送信され、サーバー(DeviceServer)側に接続されたXBee デバイスで受信されます。(C)

サーバー側では XBee データパケット受信時にイベントハンドラスクリプト(XBEE_TDCP_DATA)が実行されます。イベントハンドラスクリプト中で赤外線センサーのカウンタ値と温度を計算した後、サーバー(DeviceServer) のデータベースに登録します。(D)

クライアントからは、任意のタイミングで サーバーのデータベースにアクセスしてセンサーデータの集計を行って、グラフや集計レポートを作成できます。(E)

クライアントから各センサーデバイスに対して、現在のセンサー値を取得するためにマニュアルサンプリングを指示できます。この場合には、クライアントからサンプリングリクエストがネットワーク経由でサーバーに送信されます。サーバー では対応するリクエストコマンド(TDCPコマンド) を、各センサーデバイスの XBee モジュールに対してデータパケットに入れて送信します。受信したセンサーデバイスでは、データパケット中のリクエストコマンドに従ってマニュアルサンプリングが実行され、現在のセンサーデータを含めたリプライパケットが サーバー に XBee モジュール経由で送信されます。

その後 サーバー はリクエストを発行したクライアント側に、センサーデバイスから受信したマニュアルサンプリングデータをネットワーク経由で渡します。

同様の仕組みで、センサーデバイスの設定値変更や I/O ポートの操作リクエストをクライアントから何時でも行えます。(F) (G)

2 アプリケーション例

システム構成図で示した複数のセンサーデバイスとサーバーで動作する、いくつかのアプリケーションを作成します。この章では各アプリケーションの機能と動作概要について説明してあります。アプリケーションごとの詳しい設定方法と詳しいスクリプトファイルの内容については ”セットアップ” の章と “スクリプト作成”の章を参照してください。

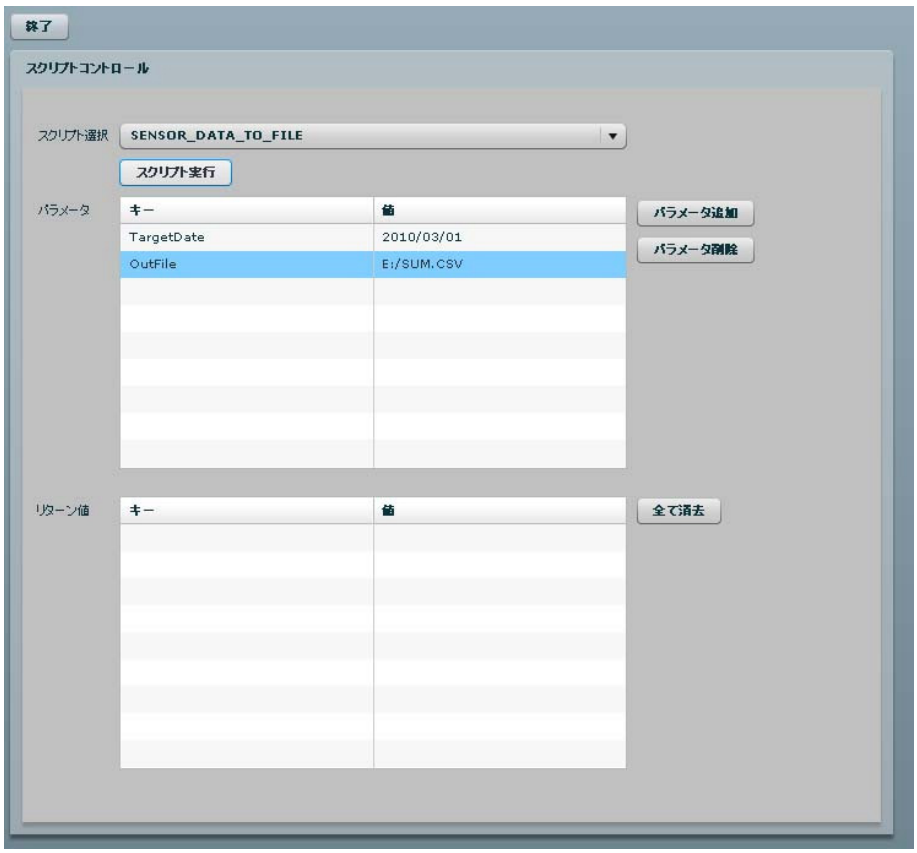
ここで説明したアプリケーション以外にも、センサーデバイスからのイベントハンドラスクリプト(TDCPイベント)を作成することで、リアルタイムにセンサーに応答するようなシステムも構築できます。

2.1 計測データの集計グラフを EXCEL で表示

サーバーのデータベースに格納されたセンサーデータを集計して、クライアントPC のエクセルでグラフ表示させるアプリケーションについて説明します。

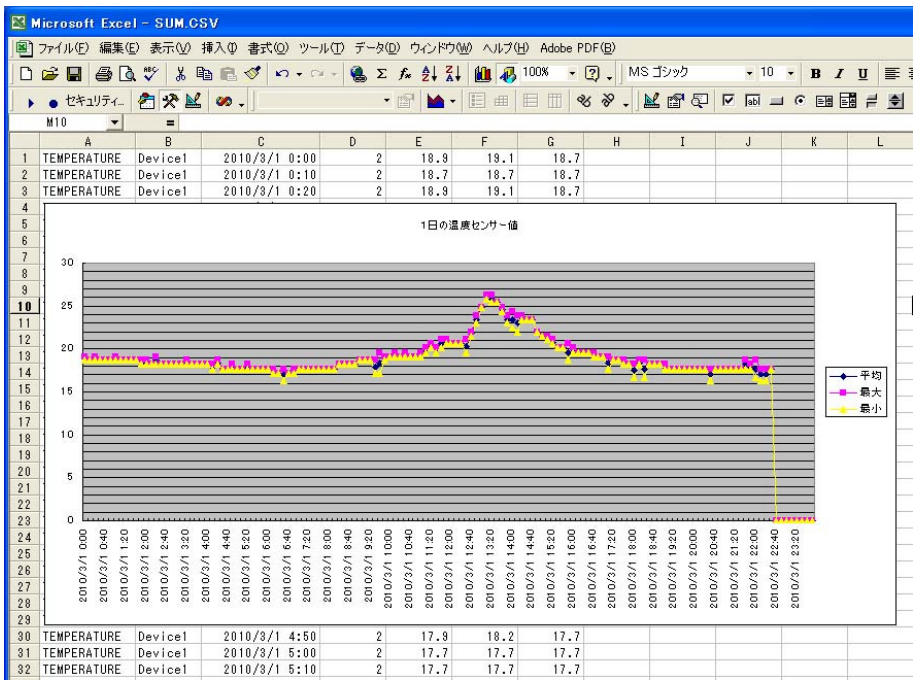
クライアントPCから、集計を行うための SENSOR_DATA_TO_FILE スクリプトを実行します。スクリプトには集計対象期間やセンサー名のキーが指定されて、指定期間内のセンサーデータ（温度、赤外センサーカウント値）を 10分ごとに集計します。集計結果は、合計値・平均値・最大値・最小値を各集計時間間隔ごとに計算して、CSV ファイルに出力します。クライアント PC からCSV ファイルを読み込んで エクセル上でグラフを作成します。

スクリプト実行は、DeviceServer インストール時にセットアップされた Webブラウザで動作する Flash アプリケーションの “スクリプトコントロール” または、DeviceServerのクライアントプログラムから”ScriptTest” プログラムを実行します。下記は、Flash アプリケーションの “スクリプトコントロール”から実行した画面例です。



スクリプトパラメータで集計対象日と集計結果を出力するCSV ファイル名を指定しています。スクリプト実行ボタンを押すと集計が開始され、指定されたファイル名で集計結果がカンマ区切り形式で出力されます。

エクセルから集計結果の CSV ファイルを開いて、グラフ表示した例が下記になります。

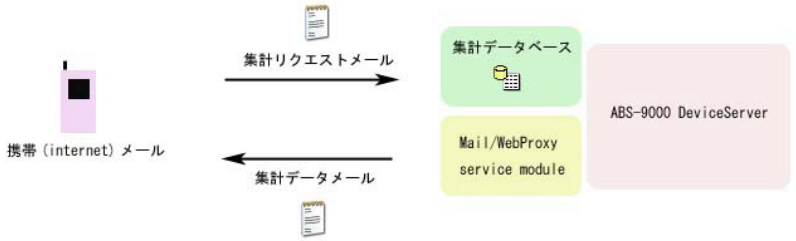


複数のセンサーデバイスの計測データまとめて同一グラフに表示したり、集計間隔（現在のスクリプト中では10分単位）や集計対象期間の変更もスクリプトを変更することで簡単にできます。

このアプリケーションの詳しいセットアップ方法とスクリプトの説明は、”スクリプト作成” の章を参照して下さい。

主に関連するスクリプトファイル(SENSOR_DATA_TO_FILE)

2.2 携帯電話(電子メール)で計測データを集計



携帯電話の電子メールやインターネットメールを使用して、センサーデータの集計操作ができます。

この機能は、DeviceServer のメールコマンド機能を使用しています。リクエストメール中に指定した条件でスクリプトを起動して集計を行います。集計結果はリクエスト送信元にメールで送信されます。

最初に、DeviceServer のメール設定を行ったメールアカウントに対してログインメールを送信します。

ログインメール送信	
メール宛先	your_mail_address@your_mail_domain
メール件名	$LOGIN$
メール本文	user_name user_password

メール宛先は、DeviceServer のメール設定で指定したPOPサーバーのメールアドレスを指定します。

user_name, user_password には、DeviceServerのユーザー情報に登録されたものを使用します。DeviceServerのユーザー設定で、メールコマンド応答先が設定されている場合には、送信を行った携帯やリモートPC のメールアドレスがメールコマンド応答先と一致していないときは、DeviceServer からのリプライメールを受け取ることができませんので注意してください。

ログイン認証成功すると、DeviceServer から下記の様なメールが送信されてきます。

ログイン結果メール受信	
メール件名	$REPLY$ -LOGIN SUCCESS-
メール本文	ログイン成功 ST02296B192B391F

２行目の文字列がセッショントークンです。ログインに失敗したときは、エラー内容が記載されたメールが送られて

きます。ログインに成功したときは、続けてセンサーデータ集計スクリプトを起動するメールを送信します。

スクリプト実行メール送信	
メール宛先	your_mail_address@your_mail_domain
メール件名	$SCRIPT$
メール本文	ST02296B192B391F SENSOR_DATA_TO_MAIL TargetDate 2010/03/01 NO_REPLY

メール宛先は、DeviceServer で設定したPOPサーバーのメールアドレスを指定します。
メール本文の１行目にはログイン成功時に送信されてきたメール中のセッショントークンを記入します。２行目に集計スクリプト名(SENSOR_DATA_TO_MAIL) を指定します。３行目以降はスクリプトパラメータを指定します。ここでは、"TargetDate" を指定して集計対象日付を 2010/03/01 に指定しています。NO_REPLY は、スクリプト実行が成功した場合に、そのステータスをメールで知らせる機能を無効にする指定です。NO_REPLY を指定しなくても動作は変わりませんが、携帯電話を使用している場合など受信データ量を少なくしたい場合に、受け取るメール数を１つ減らせます。スクリプト実行に失敗したときは NO_REPLY の指定に関わらずエラー内容が記載されたメールが送られてきます。

DeviceServer でスクリプトが実行されてセンサーデータの集計が終了すると、集計結果メールが送信されてきます。

集計結果メール受信	
メール件名	センサーデータ集計
メール本文	---------- デバイス Device1　日付: 2010/03/01 サンプル　温度 IR平均/ IR最大 ---------- 00:00 [2] 18.9　 0.0/　 0.0 00:10 [2] 18.7　 0.0/　 0.0 ... 途中省略 ... 06:10 [2] 17.5　 0.0/　 0.0 06:20 [2] 17.2　405.0/ 502.0 06:30 [2] 17.0　425.5/ 783.0 06:40 [2] 17.2　119.5/ 197.0 06:50 [2] 17.5　121.0/ 242.0 07:00 [2] 17.7　 0.0/　 0.0 ... 途中省略 ... ---------- デバイス Device4　日付: 2010/03/01

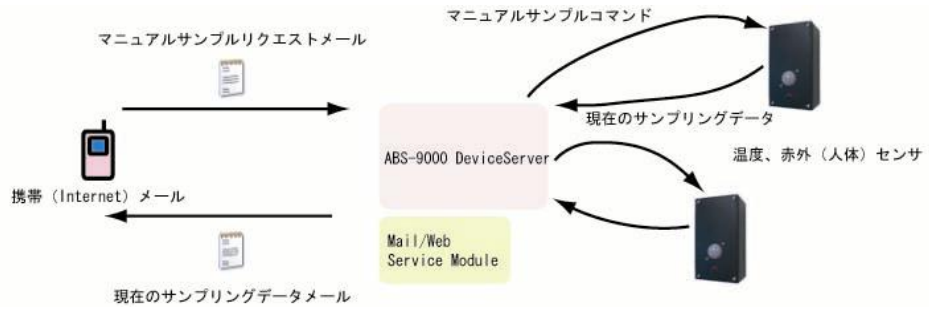
```
サンプル   温度 IR平均/ IR最大
-----------------------------------
00:00 [2] 16.1     0.0/    0.0
00:10 [2] 16.1     0.0/    0.0
... 途中省略 ...
06:00 [2] 14.9     9.0/   18.0
06:10 [2] 14.7    36.0/   72.0
06:20 [2] 14.7     7.0/   10.0
06:30 [2] 14.7    46.0/   92.0
06:40 [2] 14.7   103.0/ 206.0
06:50 [2] 14.7   237.0/ 296.0
... 途中省略 ...
22:30 [2] 15.2    26.0/   44.0
22:40 -
22:50 -
23:00 -
23:10 -
23:20 -
23:30 -
23:40 -
23:50 -
```

集計方法や集計結果のメールフォーマットを変えたいときには、スクリプトを変更することで簡単にできます。

このアプリケーションの詳しいセットアップ方法とスクリプトの説明は、”スクリプト作成” の章を参照して下さい。

主に関連するスクリプトファイル(SENSOR_DATA_TO_MAIL)

2.3 携帯電話(電子メール)で現在のセンサー値を取得



携帯電話の電子メールやインターネットメールを経由して、現在のセンサーデータのサンプリング値を取得できます。

この機能は、DeviceServer のメールコマンド機能を使用してスクリプトを起動して、そのスクリプト中からすべてのリモートセンサーデバイスに対してマニュアルサンプリングを実行します。センサーデバイスのサンプリング結果をまとめて、スクリプトからメールで送信します。

マニュアルサンプリングを実行した時に、定期的にセンサーデバイスで実行されている自動サンプリングと、データベースにセンサーデータを登録する動作に影響を及ぼさないようにします。

最初に、DeviceServer のメール設定を行ったメールアカウントに対してログインメールを送信します。

ログインメール送信	
メール宛先	your_mail_address@your_mail_domain
メール件名	$LOGIN$
メール本文	user_name user_password

メール宛先は、DeviceServer のメール設定で指定したPOPサーバーのメールアドレスを指定します。
user_name, user_password には、DeviceServerのユーザー情報に登録されたものを使用します。DeviceServerのユーザー設定で、メールコマンド応答先が設定されている場合には、送信を行った携帯やリモートPC のメールアドレスがメールコマンド応答先と一致していないときは、DeviceServer からのリプライメールを受け取ることができませんので注意してください。

ログイン認証成功すると、DeviceServer から下記の様なメールが送信されてきます。

ログイン結果メール受信	
メール件名	$REPLY$ -LOGIN SUCCESS-
メール本文	ログイン成功 ST02296B192B391F

２行目の文字列がセッショントークンです。ログインに失敗したときには、エラー内容が記載されたメールが送られてきます。ログインに成功したときには、続けてマニュアルサンプリング用スクリプトを起動するためのメールを送信します。

スクリプト実行メール送信	
メール宛先	your_mail_address@your_mail_domain
メール件名	$SCRIPT$
メール本文	ST02296B192B391F SENSOR_FORCESAMPLE_MAIL NO_REPLY

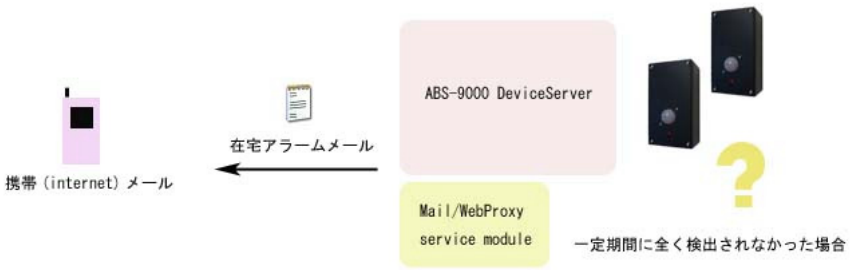
メール宛先は、DeviceServer で設定したPOPサーバーのメールアドレスを指定します。
メール本文の１行目にはログイン成功時に送信されてきたメール中のセッショントークンを記入します。２行目に集計スクリプト名(SENSOR_FORCESAMPLE_MAIL) を指定します。NO_REPLY は、スクリプト実行が成功した場合に、そのステータスをメールで知らせる機能を無効にする指定です。NO_REPLY を指定しなくても動作は変わりませんが、携帯電話を使用している場合など受信データ量を少なくしたい場合に、受け取るメール数を１つ減らせます。スクリプト実行に失敗したときは NO_REPLY の指定に関わらずエラー内容が記載されたメールが送られてきます。

DeviceServer でスクリプトが実行されてマニュアルサンプリングが終了すると、サンプリング結果のメールが送信されてきます。

サンプリング結果メール受信	
メール件名	現在のセンサーデータ
メール本文	デバイス名 Device1　赤外センサカウント 0　温度 16.7 デバイス名 Device4　赤外センサカウント 24　温度 14.7

サンプリング結果のメールフォーマットを変えたいときには、スクリプトを変更して簡単にできます。
このアプリケーションの詳しいセットアップ方法とスクリプトの説明は、”スクリプト作成” の章を参照して下さい。
主に関連するスクリプトファイル(SENSOR_FORCESAMPLE_MAIL)

2.4　赤外線(人体)センサーに一定期間反応が無い場合にアラームメール送信



一定期間、センサーデバイスの赤外線(人体)センサーからの検出が無かった場合に、アラームメールを送信できます。

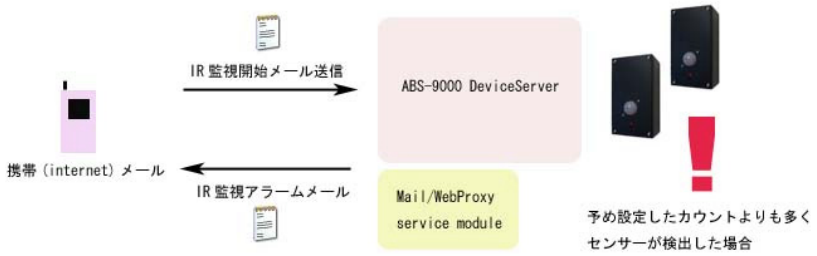
定期的にサンプリングされるセンサーデバイス中の赤外線センサーのカウンタ値が、２時間を越えて一度も検出されなかった場合に、ACTIVITIES_CHECK スクリプトが実行されてスクリプト中で指定したメールアドレスにアラームメールを送信します。

アラームメール受信	
メール件名	不在警報
メール本文	一定期間IRセンサーからの検出がありませんでした

アラームメールの送信条件は AM7:00 から PM 10:00 の間に限定して就寝時間中など活動していない時間帯を除外するようにスクリプトを作成しています。赤外線センサーデータの有無は ２時間ごとに ACTIVITIES_CHECK スクリプトでチェックして、その間に一回もセンサーの検出がなかった場合にアラームメールが送信されます。チェック間隔やアラームメールの内容はスクリプトを変更することで自由に設定できます。

このアプリケーションの詳しいセットアップ方法とスクリプトの説明は、”スクリプト作成” の章を参照して下さい。
主に関連するスクリプトファイル(PERIODIC_TIMER, ACTIVITIES_CHECK,REGISTER_ACTIVITY)

2.5 赤外線(人体)センサーで一定以上検出した場合にアラームメール送信



センサーデバイスの赤外線(人体)センサーで、あらかじめ決められた値よりも多くカウンタ値の検出があった場合に、アラームメールを送信できます。

検出はセンサーデバイスごとにリアルタイムで処理され、センサーデバイスで実行中の定期的なサンプリングや DeviceServer 側でのポーリング操作によるスクリプト実行のタイミングとは違って、直ぐに IR_SENSOR_ALERT_MAIL スクリプトが実行されます。

センサーデバイスの赤外線検出のカウンタ値があらかじめ決められた値(初期値は 10)を越えると、TDCP プログラム中でイベント(CHANGE_COUNT_EXCEED)が発生してサーバーに接続したXBee デバイスにイベントパケットが送信されます。サーバー(DeviceServer)のイベントハンドラスクリプト XBEE_TDCP_DATA から、アラームメール送信用の IR_SENSOR_ALERT_MAIL スクリプトが実行されます。
アラームメールは以下の内容で送られてきます。

IRセンサー警報受信	
メール件名	IRセンサー警報
メール本文	IRセンサーデバイスのカウンタ値が上限値を超えました デバイス名:Device1 カウンタ値:10

カウンタ値が超えた場合のイベント検出を開始するために、各センサーデバイスのカウンタイベント有効フラグを操

作する IR_WATCH_START スクリプトを実行します。このスクリプトは、センサーデバイスの TDCP プログラムにカウンタ値のクリアと、カウンタイベントの検出フラグを有効にするためのコマンドを実行します。

カウンタイベントを検出すると、イベントパケットが送信され同時にカウンタイベント検出フラグはクリアされます。イベントを検出する前に、各センサーデバイスでの検出を止めたい場合には、IR_WATCH_STOP スクリプトを実行します。

IR_WATCH_START,IR_WATCH_STOP スクリプトは、Web の “ScriptControlプログラム”や、電子メール経由で実行できます。以下に、電子メール経由でセンサーデバイスの監視を開始する例を説明します。
最初に、DeviceServer にログインするためのメールを送信します。

ログインメール送信	
メール宛先	your_mail_address@your_mail_domain
メール件名	$LOGIN$
メール本文	user_name user_password

メール宛先は、DeviceServer のメール設定で指定したPOPサーバーのメールアドレスを指定します。
user_name, user_password には、DeviceServerのユーザー情報に登録されたものを使用します。DeviceServerのユーザー設定で、メールコマンド応答先が設定されている場合には、送信を行った携帯やリモートPC のメールアドレスがメールコマンド応答先と一致していないときは、DeviceServer からのリプライメールを受け取ることができませんので注意してください。

ログイン認証成功すると、DeviceServer から下記の様なメールが送信されてきます。

ログイン結果メール受信	
メール件名	$REPLY$ -LOGIN SUCCESS-
メール本文	ログイン成功 ST02296B192B391F

２行目の文字列がセッショントークンです。ログインに失敗したときは、エラー内容が記載されたメールが送られてきます。ログインに成功したときは、続けてマニュアルサンプリング用スクリプトを起動するためのメールを送信します。

スクリプト実行メール送信	
メール宛先	your_mail_address@your_mail_domain
メール件名	$SCRIPT$
メール本文	ST02296B192B391F IR_WATCH_START

	CounterReset 1

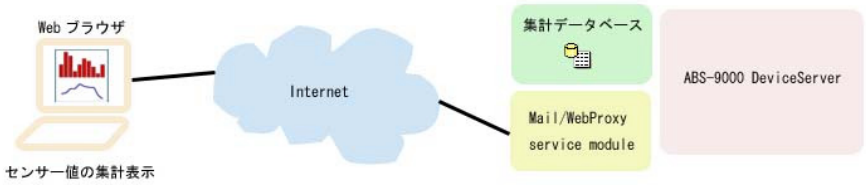
メール宛先は、DeviceServer で設定したPOPサーバーのメールアドレスを指定します。
メール本文の１行目にはログイン成功時に送信されてきたメール中のセッショントークンを記入します。２行目に集計スクリプト名(SENSOR_FORCESAMPLE_MAIL) を指定します。３行目以降はスクリプトパラメータを指定します。ここでは、"CounterReset" に "1"を指定して現在のカウンタ値をリセットしてからイベント検出のフラグを有効にしています。スクリプト実行に成功すると下記のメールが送信されてきます。

スクリプト実行結果メール受信	
メール件名	$REPLY$ -SCRIPT SUCCESS-
メール本文	スクリプト実行成功

この時点でカウンタ値の検出が有効になっています。

赤外線センサーのカウンタイベントの検出対象となるカウンタ値はデフォルトで 10 になっています。この値を変更するときは、 IR_WATCH_START スクリプト中に TDCP コマンド "change_count_high,20" の様なコマンドを追加して実現できます。このアプリケーションの詳しいセットアップ方法とスクリプトの説明は、"スクリプト作成" の章を参照して下さい。主に関連するスクリプトファイル(IR_SENSOR_ALERT_MAIL, IR_WATCH_START, IR_WATCH_STOP)

2.6 Web ページ(Flash アプリケーション)から集計グラフ表示

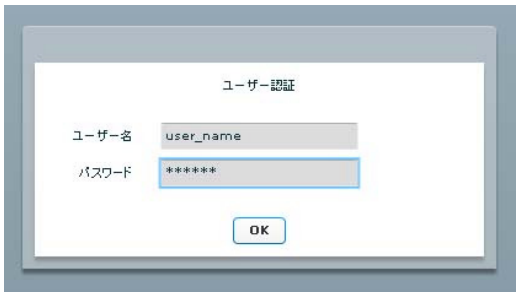


インターネットまたはLAN 上の Webブラウザから、現在のセンサーデータのサンプリング値と指定した日付のセンサーデータグラフを参照できます。

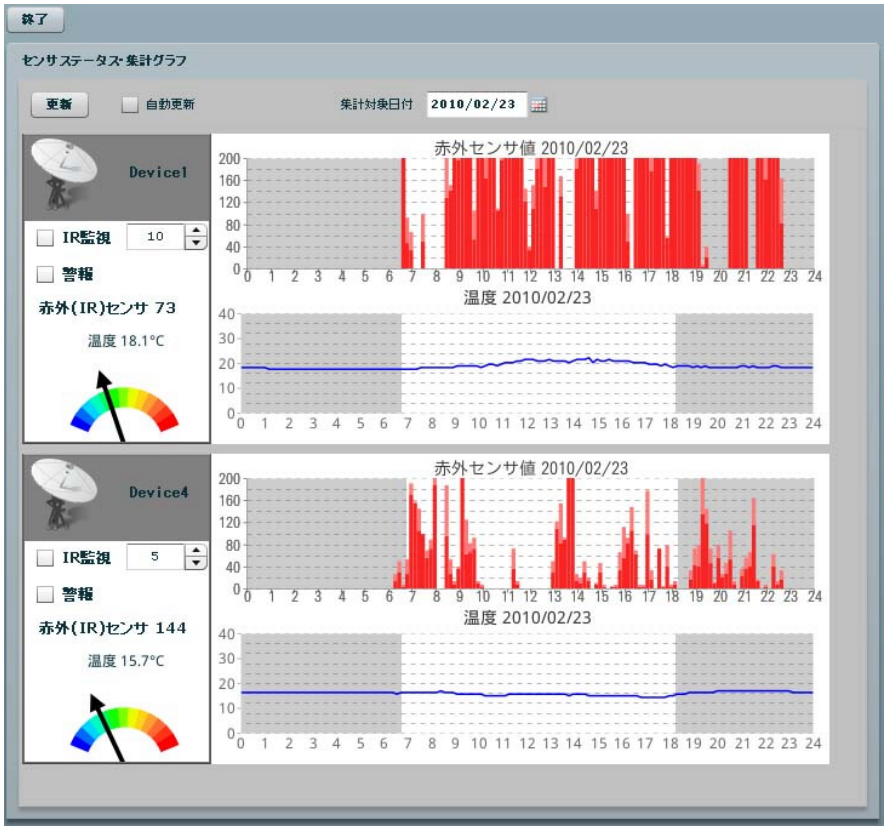
この機能は、DeviceServer の WebProxy 機能を使用して Flash アプリケーション(IRSensorMgr) を Web ブラウザ中で実行します。Flash アプリケーションとDeviceServer がネットワーク通信を行って、データベースの集計操作やリモートセンサーデバイスに対してのマニュアルサンプリング、コンフィギュレーション設定・取得コマンドの実行などを行います。

WebProxy(DeviceServer)を設定したPC の IRSensorMgr アプリケーションにWebブラウザでアクセスします。
(例：http://localhost/remote/IRSensorMgr.html)

ログイン画面が表示されますので、DeviceServer に登録されているユーザーアカウントでログインします。



ログインに成功すると下記のセンサー画面が表示されます。セットアップ済みのすべてのセンサーデバイスがリスト表示されます。３つ以上のデバイスがある場合にはスクロールバーが表示されますので、該当デバイスをマウスで選択できます。



各デバイスの左パネルには現在の赤外線センサーと温度センサーの値が表示されます。”更新”ボタンを押すと最新の情報に更新されます。これらの値は、サーバーから随時リモートセンサーデバイスに対してマニュアルサンプリングを実行した結果が表示されます。

右パネルには”集計対象日付”で指定した日付の集計グラフが表示されます。日付を変更するごとにサーバーで集計動作が実行されますので、右パネルの各グラフ画面が表示されるまでに少し時間がかかる場合があります。

自動更新にチェックを入れると、約１分ごとに更新ボタンを押したのと同様に画面が最新の情報に更新されます。

“IR” 監視チェックボックスは、前述の “IR監視アラートメール” の監視状態を示します。ここでチェックボックスを操作して、監視状態の開始と停止を指定できます。チェックボックスの右側の値は、IRカウンタのスレシュホールド値の表示・変更用です。チェックボックスを操作すると、サーバー経由でリモートセンサーデバイスをリアルタイムに操作して監視ステータスを変更します。

“警報” チェックボックスは、センサーデバイス中 TDCP ベースボードの PORTA bit#7 にブザーを接続していた場合に、アラーム音(ピピピ... 連続音) 出力を変更できます。チェックボックスを操作すると、サーバー経由でリモートセンサーデバイスをリアルタイムに操作してブザー出力を変更します。

このアプリケーションの詳しいセットアップ方法とスクリプトの説明は、”Web クライアントプログラム設定” の章を参照して下さい。 TDCP のブザー出力についての詳しい内容は、”TDCP ユーザーマニュアル” の app_mode =8 の説明を参照して下さい。 (http://www.allbluesystem.com/TDCP/TDCP_Users.pdf)

3 必要な機材・リソース

3.1 サーバーPC

必要なシステムやデバイス	説明
ABS-9000 DeviceServerの動作しているPC	DeviceServer の動作する PCが１台必要です。
XBee デバイス	サーバー PC 用に Digi International Inc. 社製 XBee IEEE 802.15.4 デバイスが 1台必要です。DeviceServer の COM ポートに接続して各センサーデバイス間との通信を行います。 DeviceServer は、XBee デバイスのファームウエアバージョンの “10CD” にのみ対応しています。(必要に応じて XBee ファームウエアの更新を行ってください)

3.2 センサーデバイス

必要なシステムやデバイス	説明
TDCP プログラムが動作している CPU ボード	“TDCP ユーザーマニュアル” のTDCP リモートコントローラボード作成例を参照してください。 (http://www.allbluesystem.com/TDCP/TDCP_Users.pdf)
XBee デバイス	サーバー PC 用に Digi International Inc. 社製 XBee IEEE 802.15.4 デバイスが 1台必要です。DeviceServer の COM ポートに接続して各センサーデバイス間との通信を行います。 DeviceServer は、XBee デバイスのファームウエアバージョンの

	“10CD” にのみ対応しています。(必要に応じて XBee ファームウエアの更新を行ってください)
温度センサー	LM35DZ アナログ出力可能であればどの様なデバイスでも構いませんが、スクリプト中の温度計算と A/D変換のVref部分の変更が必要になります。
赤外線（人体）センサー	焦電型赤外線センサモジュール（デジタル出力タイプ、検出時のデータホールド機能無し） 今回使用したのは、秋月電子通商 “SE-10” です。これ以外でもデジタル出力可能なタイプであれば使用できます。

4 セットアップ

4.1 XBee デバイス初期設定(サーバー側・センサーデバイス共通)

XBee デバイスを DeviceServer とセンサーデバイスの TDCPで使用するために初期設定が必要です。
使用するすべての XBee デバイスで下記の設定を行ってください。

最初に DeviceServerとの接続に必要な最低限の設定を COM ポート経由で行います。Sparkfun Electoronics 社製の XBee Explorer USB などを使用して、仮想 USB ポート経由で接続して設定してください。(これ以外の方法で COM ポート接続する場合も手順は同じです)

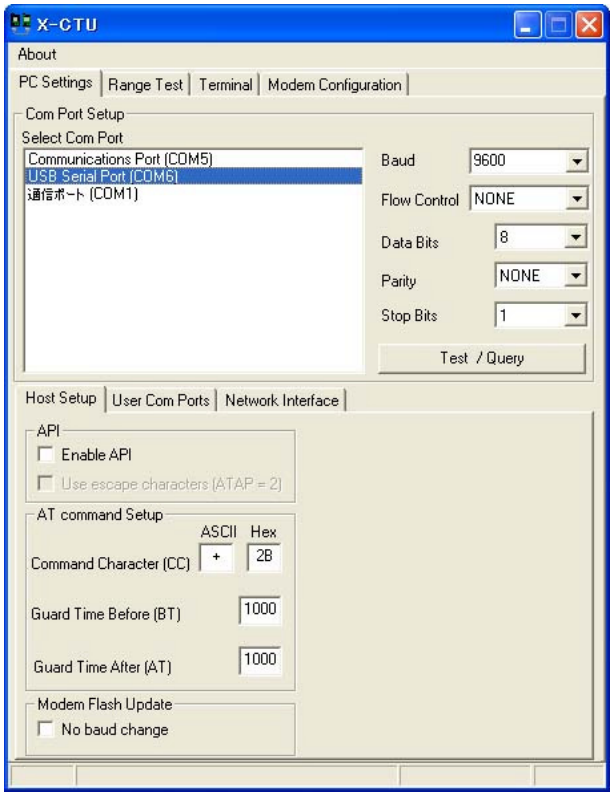
センサーデバイスの XBee デバイスについては TDCP ベースボードのコンソールポートのジャンパーを設定して、コンソールポートが RS-232C ラインバッファを経由して直接 XBee デバイスのシリアルポートと接続された状態で操作を行います。TDCP ベースボードのコンソールジャンパー設定の詳しい手順については、”TDCP ユーザーマニュアル”を参照して下さい。http://www.allbluesystem.com/TDCP/TDCP_Users.pdf
DeviceServer に接続する XBee デバイスの設定に使用した XBee Explorer USBのソケット上で、センサーデバイスで使用する XBee デバイスに入れ替えて設定しても構いません。

XBee デバイス デフォルト値から変更が必要な設定値	
API モード	1 (default は 0)
PAN(Personal Area Network) ID	任意の値 (default は0x3332) デフォルトの値のままだと、予期しないデバイスからのフレームを受信したり、間違ってデバイスを操作する恐れがありますので、適当な任意の値を設定するようにしてください。このマニュアルではコントローラ側 PANに 0xAB90、センサー側 PAN に0xAB91 を使用しています。

	コントローラ側 PAN と センサー側 PAN で別々の値を設定できますが、同一 PAN 内にある DeviceServer の COM ポートに接続する XBee デバイスとリモート XBee デバイスは必ず同じ値を使用してください。
16bit Source Address	同一PAN ID 内でユニークな値 (default は 0x0000) この値は、ここで設定しなくても後から XBee 管理プログラムで設定可能ですが、デバイス一覧から選択したデバイスがどのデバイスであるかを見分けることが容易になるように便宜的にここで設定します。 すべてのデバイス間で違った値を設定してください。 (0x0000, 0xFFFF, 0xFFFE を除く) 例えば、0x0001, 0x0002, 0x0003 など

上記３つの 初期設定のコマンドをXBee に送信するために、XBee デバイスを PC のCOM ポートに接続して Digi international Inc. 社製の X-CTU プログラム、または汎用のターミナルエミュレータプログラムなどを使用します。XBee とCOM ポートのボーレートは初期設定の 9600 bps にして下さい。（ターミナルエミュレータを使用するときは、ローカルエコー ON, 受信時の改行 CR + LF にするとコマンド実行の結果が見やすくなります）

X-CTU プログラムを起動して、COM ポートを選択します。ここでは、USB Serial Port(COM6) を選択しています。

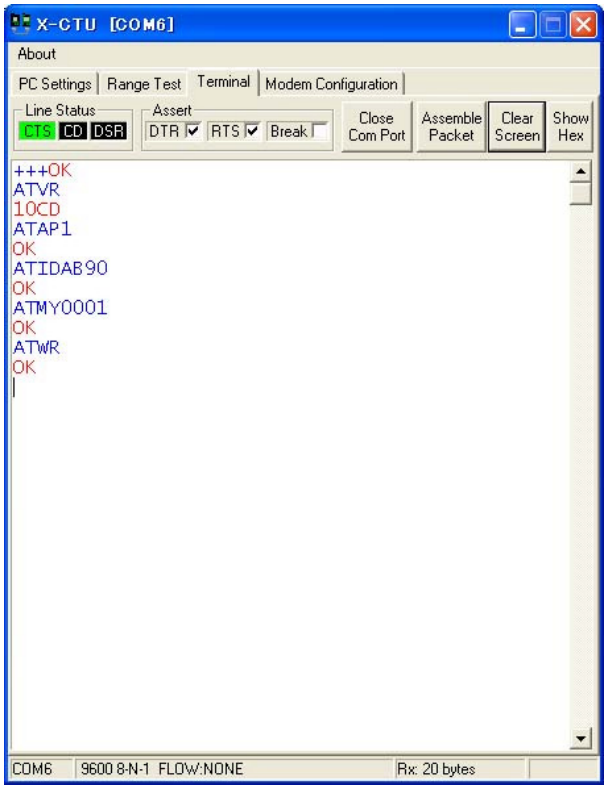


Terminal タブを選択してターミナル画面を表示します。キーボードから、"+++" を入力して、コマンドモードに入

ります。コマンドモードに入ると “OK” が表示されますので、続けて以下のコマンド文字列を入力してください。
コマンド入力の時間がかかりすぎると、自動的にコマンドモードから抜けてしまいますので、そのときには、”+++” を入力して最初からコマンドを入力し直して下さい。

```
ATVR
ATAP1
ATIDAB90
ATMY0001
ATWR
```

最初に ATVR でファームウエアバージョンを表示しています。”10CD” 以降になっていることを確認してください。
ATAP1 は、API モードを “1” に設定しています。ATIDAB90 は PAN_ID を 0xAB90 に設定しています。もし別の PAN_ID を使用するときには適宜変更してください。次に、ATMY0001 で、デバイスの16 bit Source Address を “0x0001” に設定しています。この部分は、デバイスごとにユニークな値になるように変更して下さい。
最後に、ATWR で、設定値を不揮発メモリに書き込みます。入力時の画面表示は以下のようになります。



X-CTU プログラムを終了します。 その後、XBee Explorer USB に接続する XBee デバイスを切り替えて、 使用するすべての XBee デバイスについて同様に初期設定を行って下さい。
このときに、設定した16bit Source Address の値をデバイス機器にマーキングしておくと、後で XBee デバイス管理プログラムでデバイスを選択するときに、識別し易くなります。

システム上の XBee デバイスの設定値例は、以下のようになります。

デバイス番号(用途)	API モード(ATコマンド)	PAN_ID(ATコマンド)	16bit Address(ATコマンド)
XBee#1(センサーデバイス)	1(ATAP1)	0xAB90(ATIDAB90)	0x0A01(ATMY0A01)
XBee#2(DeviceServer接続)	1(ATAP1)	0xAB90(ATIDAB90)	0x0B02(ATMY0B02)
XBee#4(センサーデバイス)	1(ATAP1)	0xAB90(ATIDAB90)	0x0D04(ATMY0D04)

4.2 XBee デバイスを DeviceServer に接続

XBee デバイスの初期設定後に、XBee#2(サーバー) を PC の COM ポートに接続します。

デバイスが PC に接続されたら、DeviceServer から XBee デバイスを使用するために COM ポートの設定を行います。

サーバー設定プログラム(ServerInit.exe)を起動して、XBEE タブを選択して COM ポート番号を設定して “XBEE 機能を有効にする” にチェックをつけてください。

また、I/O データ受信時に XBEE_IO_DATA イベントハンドラを実行するために、”イベントパケット受信時にスクリプト実行” にもチェックを付けてください。

サーバー設定プログラムの “次へ” を押して”完了”ボタンが表示されるまで進めて設定を完了して下さい。

4.3 センサー側デバイス接続

センサーデバイスに内蔵する XBee デバイスを、一時的にサーバー PC の XBee Explorer USB などに接続して初期設定した場合には、取り出してセンサーデバイスの TDCP ボードにセットします。

温度センサーと赤外線センサーの接続を行った後にセンサーデバイスの電源を接続して、サーバー側の XBee と通信できる状態にしておきます。

これ以降の センサーデバイス上の XBee デバイスの設定変更や、センサーデバイスの設定(TDCP プログラムのコンフィギュレーション)はすべて サーバー PC からリモートコマンドで操作できます。

4.4 マスター登録と XBee 詳細設定

センサーデバイスの XBee デバイスと、サーバー PC に接続した XBee デバイスを DeviceServer のマスターファイ

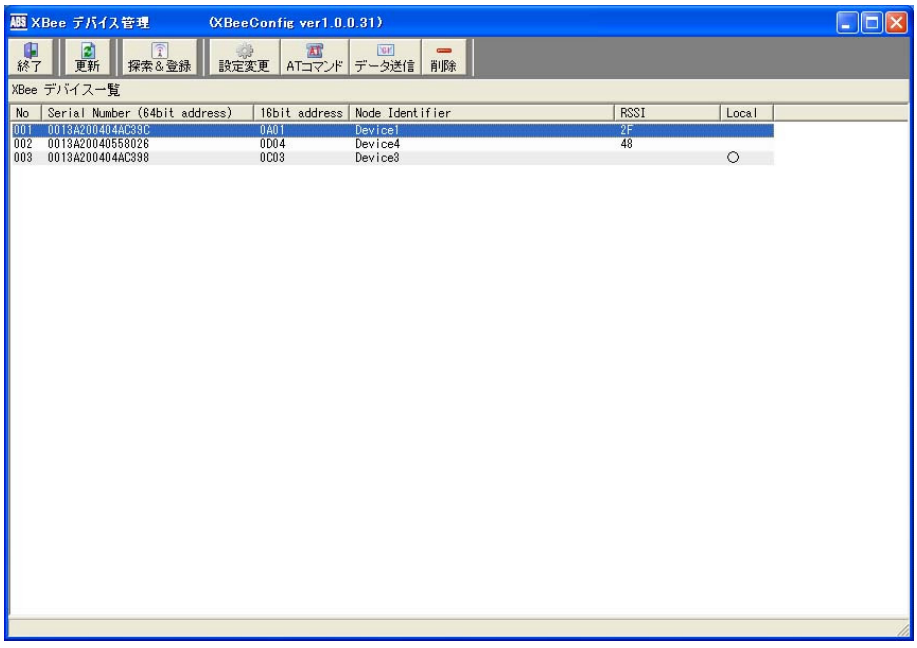
ルに登録します。XBee デバイスの初期設定で設定しなかった NodeIdentifierなどの詳細設定もここで行います。

XBee デバイス管理プログラムを使用して、同一 PAN ID をもつ XBee デバイスを DeviceServer に登録したり、デバイス自身の設定内容を変更します。プログラムメニューから “ALL BLUE SYSTEM” -> “クライアント起動”を選択・実行します。ログインするときは、管理者特権をもったユーザー（例えば DeviceServer セットアップ時に管理者アカウントとして登録したユーザーなど）でログインしてください。デスクトッププログラムが起動したら、“XBee” ツールボタンを選択してXBee デバイス管理プログラムを起動します。

DeviceServer では 登録済みの XBee デバイスをマスターファイルに記録しています。
XBee デバイス登録は、”探索＆登録” ボタンを押すことで、同一 PAN ID のリモートデバイスを見つけて、自動的にマスターファイルに登録します。既に、登録済みのXBee デバイスの項目は最新の情報でマスターファイルの内容が更新されます。DeviceServer に COMポートで直接接続された XBee デバイスについても同様に自動登録されます。

XBee デバイス管理プログラムの “探索＆登録” ツールボタンを押します。

登録確認ダイアログが表示されますので、”OK” を押します。
DeviceServer の COM ポートに直接接続された XBee デバイスで “Node discover” が実行され、付近にある同一 PAN ID の XBee デバイス情報を取得して、自動的にマスターファイルに登録が行われます。XBee デバイス管理プログラムのデバイス一覧には、登録済みのXBee デバイスが表示されます。



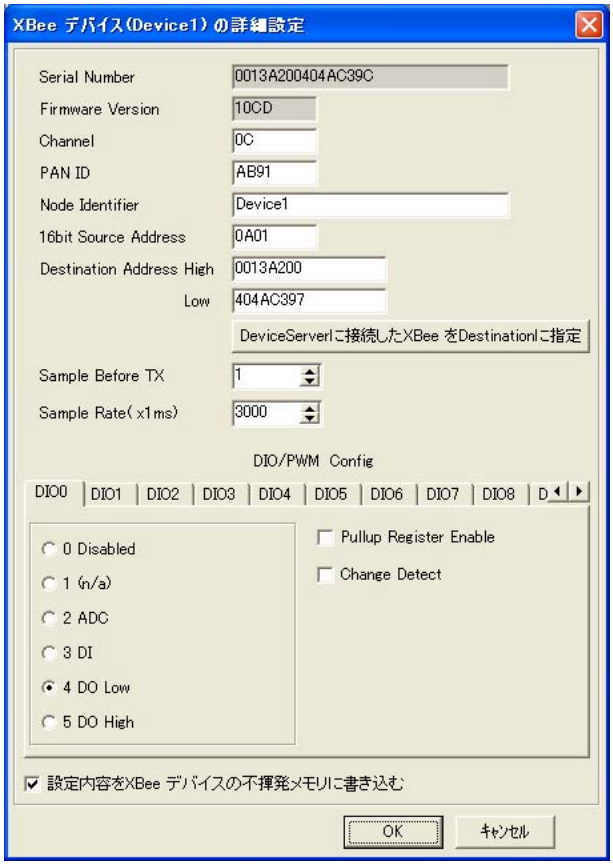
(Node Identifier 項目は、詳細設定修正後に 再度 “探索＆登録” ボタンを押すことで上記のように表示されます)

XBee デバイスの詳細設定を変更するために、XBee デバイス管理プログラムのデバイス一覧から対象デバイスを選択して、”設定変更” ツールボタンを押します。初期設定時に 16 bit Source Address を設定したときは、その値がデバイス一覧に表示されていますので、変更対象の XBee デバイスを確認できます。

選択したXBee デバイスと通信を行って、現在のデバイス情報を取り込みます。
もしエラーが発生したときは、選択したXBee デバイスとの間で通信ができない状態になっていますので、通信経路や電源を確認してください。

XBee デバイスの現在の設定値を取り込んだ後、詳細設定変更ダイアログが表示されます。
ここでは各 XBee デバイスの Node Identifier の設定を行って下さい。Node Identifier に指定できる文字は ASCII で 20文字までです。また、ダイアログに表示されている 16 bit Source Address が、対象のデバイスであるかどうかの確認も行ってください。ここで 16 bit Source Address を任意の値に変更することもできます。

デバイス番号 (16 bit Source Address)	デバイス名 (Node Identifier)
XBee#1 (0x0A01) センサーデバイス	Device1
XBee#3 (0x0C03) DeviceServer 接続	Device3
XBee#4 (0x0D04) センサーデバイス	Device4



項目を修正した後、 ”設定内容を XBee デバイスの不揮発メモリに書き込む” にチェックを付けて “OK” を押してく

ださい。

今回のシステムでは XBee デバイス上の I/O や ADC、サンプリング機能は使用しませんので、"Node Identifier" 以外の項目を設定する必要はありません。

XBeeデバイスの Node Identifier を変更したときは、XBee デバイス管理プログラムのデバイス一覧に表示されている、マスターファイルも更新しておく必要があります。XBee デバイス管理プログラムの "探索＆登録" ツールボタンを押します。

デバイスのNode Identifier または 16bit Source Address以外の詳細設定を変更する場合には、マスターファイルの更新は必要ありません。

4.5 センサー側デバイス TDCP 設定

センサーデバイスの TDCP ボードの設定を行います。温度センサー用の A/D 変換入力と赤外線センサーのカウンタ入力をサーバーに定期的に送信するために、下記のTDCP コンフィギュレーション値を設定します。

TDCP プログラム設定項目

項目名称	設定内容	設定用 TDCPコマンド
TDCP 動作モード	app_mode=8 に設定する A/D、デジタルカウンタ入力を行うモード	"app_mode,8"
サンプリングデータとイベントデータ送信先 XBee アドレス	サーバーPC に接続した XBee デバイスのシリアル番号(64bitアドレス)	"server_addr,<64bitAddr(*1)>"
デジタル入力ポートのプルアップ	すべてのビットのプルアップを有効にする	"pullup,FF"
デジタル入力変化イベントの検出	すべてのビットのイベント検出を有効にする	"change_detect,FF"
自動サンプリング間隔	5 分ごとにサンプリングを実行して、サーバーにサンプリングイベントを送信する。	"sampling_rate,300"
デジタル入力（カウンタ入力）のイベントが発生する制限値	デジタル入力ポート(カウンタ入力ビット #4..#7)のいずれかが 10 回以上変化した場合にイベントを送信する(*3)	"change_count_high,10"
コンフィギュレーション保存	TDCP 設定内容を CPU 内蔵の EEPROM に保存	"config_save"
CPU リセット	TDCP プログラムのリセット。 コンフィギュレーションで保存された新しい app_mode とプルアップ設定で再起動する。	"reset"(*2)

(*1) サーバーPC に接続した XBee のシリアル番号は XBee デバイス管理プログラムからデバイス一覧で確認するか、スクリプト中から xbee_my_serial_number() 関数を使用して取得できます。

(*2) リセットコマンド実行時は、リモートデバイスからのリプライパケットは常に返らないので、スクリプトから xbee_tdcp_command() 関数を使用してリセットコマンドを送信する場合には、no_result パラメータに true を指定して下さい。
(*3) “赤外線(人体)センサーで一定以上検出した場合にアラームメール送信” アプリケーションで、赤外線センサーが反応したら直ぐにアラームを発生したい場合にはこの値を 1 に設定してください。

設定用 TDCP コマンドは、スクリプト中にリモートコマンドを記述して実行する方法と、１コマンドごとに XBee デバイス管理プログラムから送信する方法のどちらか方法で実行できます。

下記に、それぞれの方法で設定を行う場合について記述しますが、どちらかの方法で設定してください。
複数のセンサーデバイスがある場合には対象デバイスを切り替えてすべてのセンサーデバイス中の TDCP プログラムの設定を行ってください。

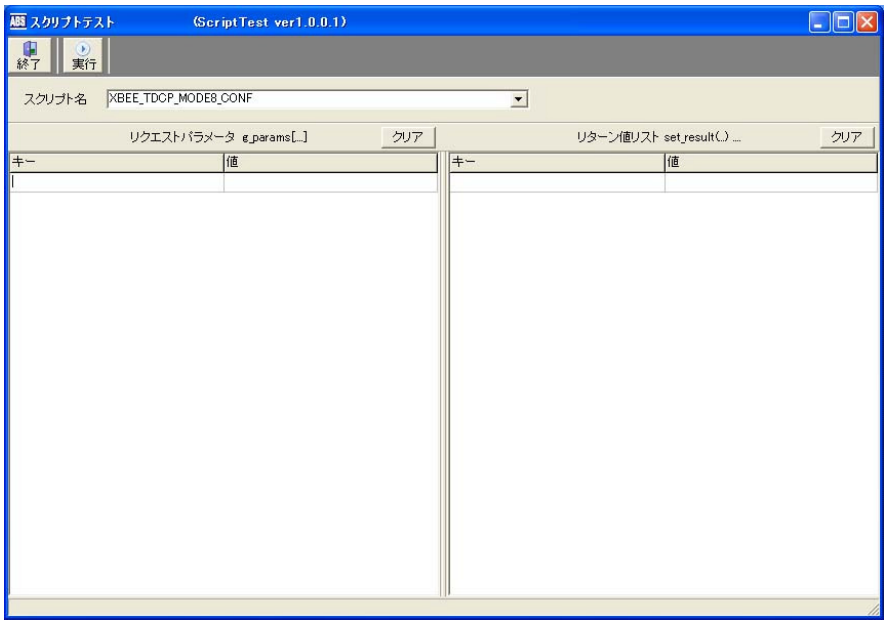
4.5.1 XBEE_TDCP_MODE8_CONF スクリプト作成

下記のスクリプトを作成して、センサーデバイスの TDCP プログラムの設定を行います。

```
file_id = "XBEE_TDCP_MODE8_CONF"
--[[
          TDCP デバイス初期設定用スクリプト
          スクリプト中の device = "xxxxx" 部分を 使用するXBee デバイスの
          NodeIndentifier に変更してからスクリプトを実行してください。
          複数の TDCP デバイスを使用する場合には、全ての XBee デバイスの
          NodeIndentifier についてこのスクリプトを修正・実行してください。
          スクリプトを実行すると、TDCP デバイスに下記の設定が行われた後
          EEPROM に保存された後、デバイスがリセットされます。
]]
local device = "Device1"

log_msg("start..",file_id)
local svr_addr,key,val,stat,result
stat,svr_addr = xbee_my_serial_number()
if (not stat) or (svr_addr == "") then error() end
log_msg("DeviceServer's XBee address is " .. svr_addr,file_id)

stat,result = xbee_tdcp_command(device,"app_mode,8")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"server_addr," .. svr_addr)
if (not stat) or (result[2] ~= "1") then error() end
```

```
stat,result = xbee_tdcp_command(device,"pullup,FF")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"change_detect,FF")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"sampling_rate,300")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"change_count_high,10")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"config_save")
if (not stat) or (result[2] ~= "1") then error() end

stat,result = xbee_tdcp_command(device,"reset",true)
if (not stat) then error() end
```

ファイル名(XBEE_TDCP_MODE8_CONF.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。スクリプト実行は、Web の “ScriptControl プログラム” または DeviceServerのクライアントプログラムから実行します。下記は、DeviceServer のクライアントプログラムから実行した画面例です。



“実行”ボタンを押すとスクリプトがサーバーで実行され、リモートセンサーデバイスの TDCP プログラムの設定が行われます。ログには、下記の様なコマンド実行ごとのリモートデバイスからの応答パケット受信が記録されま

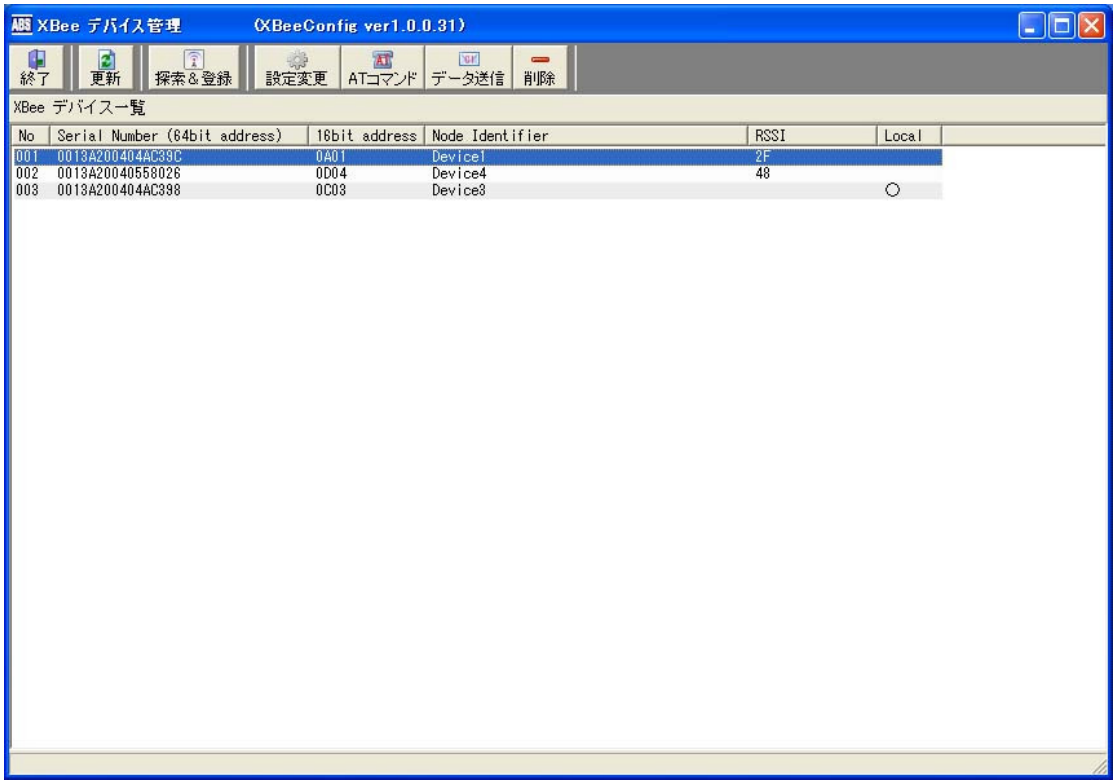
す。”$$$<文字列>” の後のカンマに続けて “1” が返っていればコマンド実行が成功したことを示しています。

```
2010/03/01 09:30:08 falcon      XBEE_TDCP_MODE8_CONF 0 start..
2010/03/01 09:30:08 falcon      XBEE_TDCP_MODE8_CONF 0 DeviceServer's XBee address is 0013A200404AC398
2010/03/01 09:30:08 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$61052,1
2010/03/01 09:30:09 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$26988,1
2010/03/01 09:30:09 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$16124,1
2010/03/01 09:30:09 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$85710,1
2010/03/01 09:30:09 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$86327,1
2010/03/01 09:30:09 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$25659,1
2010/03/01 09:30:10 falcon      XBEE_TDCP_DATA       0 Device1[0A01,0013A200404AC39C] RFData = $$$68306,1
```

“Device1” の設定が終了したら、スクリプトの内容を編集して設定対象デバイスを “Device4” に変更して同様にスクリプトを実行します。

4.5.2 XBee デバイス管理プログラムから TDCP コマンド実行

DeviceServerのクライアントプログラムを起動して、XBee デバイス管理プログラムを開きます。



“Device1” を選択して、”データ送信” ボタンを押します。下記のダイアログが表示されて、リモートのXBee デバイスに XBee API データパケットを送信できます。

前述の 設定用 TDCP コマンドを入力して “送信ボタン”を押します。

この時、TDCP コマンドプリフィックス文字列の “$$$” とカンマ “,” を 設定用 TDCP コマンドとパラメータの前に付けてください。”$$$” の代わりに “$$$abc” の様な文字列にするとリモートデバイスでコマンドが実行された結果のリプライデータパケットを受け取ることができます。リプライパケット受信時にはログに下記のような内容が出力されます。

```
2010/03/01 22:33:50 falcon      XBEE_TDCP_DATA      0 Device1[0A01,0013A200404AC39C] RFData = $$$abc,1
```

リクエスト時に指定した TDCP コマンドプリフィックスの後のカンマに続けて “1” が返った場合にはコマンド実行が成功したことを示します。コマンド実行が失敗した場合には “0” が返ります。ただし、“reset” コマンド実行時にはリプライパケットは返りませんので注意してください。TDCP コマンドプリフィックスについては ”TDCP ユーザーマニュアル”を参照して下さい。

すべての 設定用 TDCP コマンドを同様に実行して下さい。“Device1” の設定が終了したら”閉じる” ボタンでデバイスリスト画面に戻ります。同様の手順で “Device4” の設定を行ってください。

5 スクリプト作成

システム全体の動作をコントロールするスクリプトを記述します。

⚠ 注意

スクリプト中に日本語を記述するときは、スクリプトファイルを UTF-8N 形式で保存してください。Shift_JISや UTF-8 BOM付き形式などで保存すると、DeviceServer でエラーが発生します。Windows付属のワードパッドやメモ帳ではこの形式で保存できませんので、別途 UTF-8N 形式で保存可能なエディタソフト（*1）を使用してください。

（*1）TeraPad などのソフトウエアがよく使用されています。

5.1 XBEE_TDCP_DATA スクリプト作成

サーバーPC でセンサーデバイスからサンプリングデータやイベントデータ、リクエスト応答パケットなどを受信し

たときに実行されるスクリプトを作成します。

DeviceServer をインストールしたときに、初期ファイルとして XBee データパケット内容をログに出力する機能のみが記述されていますので、これに追加記述します。

サンプリングイベントを受信した場合には、データベースに登録する REGISTER_ACTIVITY スクリプトを実行します。他のイベント処理をスクリプトに追加した時でも、REGISTER_ACTIVITY スクリプトの処理時間に影響されないように、script_fork_exec() を使用して別スレッドで REGISTER_ACTIVITY スクリプトを実行するようにします。

デジタル入力ポートのカウンタ値が、設定した値よりも大きくなった場合には CHANGE_COUNT_EXCEED イベントが発生して、XBEE_TDCP_DATA スクリプトが実行されます。この時には、アラームメールを送信するための IR_SENSOR_ALERT_MAIL スクリプトを実行します。

サンプリングイベント (SAMPLING) とカウンタイベント(CHANGE_COUNT_EXCEED)で、センサーデバイスから XBee データパケットで渡されるデータフォマットについては、"TDCP ユーザーマニュアル"の "TDCP動作モード (app_mode=8)"の章と "TDCPイベントリファレンス"の章を参照して下さい。

```
file_id = "XBEE_TDCP_DATA"
--[[
XBEE_TDCP_DATA スクリプト起動時に渡される追加パラメータ
--------------------------------------------------------------------------------
キー                値                                          値の例
--------------------------------------------------------------------------------
APIType             フレームデータ中のAPI Type(16進数2桁)       81
SourceAddress       フレームデータ中のSourceAddress
                    16bit アドレスの場合(16進数4桁)             0A01
                    64bit アドレスの場合(16進数16桁)            0013A200404AC397
SerialNumber        XBee デバイスの SerialNumber
                    DeviceServer に保持されたマスターファイルを使用して、
                    SourceAddress から変換した値が設定される。  0013A200404AC397
NodeIdentifier      XBee デバイスの NodeIdentifier。
                    DeviceServer に保持されたマスターファイルを使用して、
                    SourceAddress から変換した値が設定される。  Device1
RSSI                フレームデータ中のRSSI(16進数2桁)           45
Options             フレームデータ中Options                     00
TDCP_COUNT          TDCP データカラム数                         2
TDCP_<Column#>      TDCP データ値(ASCII 文字列)
                    TDCP_1 は常にコマンドプリフィックス文字列を表す  "$$$1234"
                    "$$$" で始まり、0文字以上の任意の文字列が後に続く。
```

```
                    TDCP_2 はコマンド実行ステータスを表す                    "1"
                    "1" はコマンド実行成功、"0" は失敗を示す
                    イベントデータの場合にはイベント名が入る
                    TDCP_3以降のデータはTDCPコマンド毎に決められた、オプション文字列が入る
                    <Column#> には 最大、TDCP_COUNT まで 1から順番に
                    インクリメントされた値が入る。
]]

local rf_data = "";
local tkey,i;
local max_idx = tonumber(g_params["TDCP_COUNT"]);
for i = 1,max_idx,1 do
          tkey = "TDCP_" .. tostring(i);
          if (i == 1) then
                    rf_data = g_params[tkey]
          else
                    rf_data = rf_data .. "," .. g_params[tkey]
          end;
end;
log_msg(g_params["NodeIdentifier"] .. "[" .. g_params["SourceAddress"] .. "," .. g_params["SerialNumber"] ..
"] RFData = " .. rf_data,file_id)
--------------------------------------------------------------------------
-- サンプリングイベントを受信した場合は統計用データベースに登録するための
-- スクリプトを実行する。
--------------------------------------------------------------------------
local stat;
if g_params["TDCP_2"] == "SAMPLING" and g_params["TDCP_3"] == "8" then
          local key_list =
list_to_csv("SerialNumber","RemoteDeviceName","ActivityLevel","TemperatureLevel")
          local val_list =
list_to_csv(g_params["SerialNumber"],g_params["NodeIdentifier"],g_params["TDCP_5"],g_params["TDCP_9"])

          stat = script_fork_exec("REGISTER_ACTIVITY",key_list,val_list)
          if not stat then error() end
end

--------------------------------------------------------------------------
-- CHANGE_COUNT_EXCEED イベントを受信した場合は、アラートメール送信のための
-- スクリプトを実行する
```

```
--------------------------------------------------------------------------
local stat;
if g_params["TDCP_2"] == "CHANGE_COUNT_EXCEED" and g_params["TDCP_3"] == "8" then
          local key_list = list_to_csv("SerialNumber","RemoteDeviceName","ActivityLevel")
          local val_list =
list_to_csv(g_params["SerialNumber"],g_params["NodeIdentifier"],g_params["TDCP_4"])

          stat = script_fork_exec("IR_SENSOR_ALERT_MAIL",key_list,val_list)
          if not stat then error() end
end
```

ファイル名(XBEE_TDCP_DATA.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.2 REGISTER_ACTIVITY スクリプト作成

センサーデバイスからサンプリングデータを受信したときに XBEE_TDCP_DATA スクリプトから呼び出されて、データベースに登録するためのスクリプトを作成します。

スクリプトパラメータには、センサーデバイス名とサンプリング値が渡されてきますので、そのデータを元に実際の温度を計算してデータベースに格納します。同様に赤外線センサーのカウンタ値もデータベースに格納します。スクリプトに渡されるパラメータ詳細はスクリプトのコメントを参照してください。

温度センサーの計測には A/D 変換機能を使用していますので、各センサーデバイスの実際の温度を計算するためには A/D リファレンス電圧を得ておく必要があります。センサーデバイスではリファレンス電圧用のハードウエアを使用しない単純な構成なので、各センサーデバイスごとに AVRef 電圧を事前にデジタル電圧計で計測しておいて、それを温度計算に使用します。事前に計測した AVRef 電圧は各センサーデバイスごとに違った値になりますので、その値を DeviceServer のデータベースに格納しておきます。このために、システム起動時に一回だけ SENSOR_AVREF_SETUP スクリプトを実行してAVRef 電圧をデータベースに格納します。

温度計算は、各センサーデバイスごとの AVRef 電圧をデータベースから取得して、その値と現在の A/D 変換値を元に摂氏温度を得ます。データベースに格納する時には、後で集計時に検索しやすいようにデバイス名をキーにして登録します。赤外線センサーのカウンタ値も同様にデータベースに格納します。

一定期間センサーデバイスの赤外線(人体)センサーからの検出が無かった場合を判断するために、赤外線センサーカウンタ値が 0 よりも多い場合には 共有データ(RESIDENTS_ACTIVITIES_FLAG) にフラグをセットします。このフラグの値は２時間ごとに ACTIVITIES_CHECK スクリプトでチェックされています。

クライアントプログラムや他のスクリプトから、現在サンプリングデータを送信している有効なセンサーデバイス一覧を取得できるように、データベース(キー名 “TDCPDeviceList”) にデバイス名とシリアル番号も登録しておきます。

```
file_id = "REGISTER_ACTIVITY"
--[[
REGISTER_ACTIVITY スクリプト起動時に渡されるパラメータ
-------------------------------------------------------------------------------
キー値              値                                          値の例
-------------------------------------------------------------------------------
RemoteDeviceName    監視用リモートデバイス名                    LivingRoom
SerialNumber        監視用リモートデバイスのシリアル番号        0013A200404AC397
ActivityLevel       赤外センサーで関知した動作レベル(カウンタ値)        120
TemperatureLevel    温度センサーの値を A/D 変換した値           35
-------------------------------------------------------------------------------
カウンタ値      統計用データベースキー名            "SENSOR_IR_" + <SerialNumber>
温度センサーAD値 統計用データベースキー名           "SENSOR_TP_" + <SerialNumber>
統計用データベースに登録する値                      ActivityLevel を数値に変換した値
]]
local temperature;
local key,val,stat;
local adc_vref = 5.00;
-------------------------------------------------------------
-- DeviceServer 起動後に一回だけ SENSOR_AVREF_SETUP スクリプトを実行する
-------------------------------------------------------------
stat,val = get_shared_data("DO_SENSOR_AVREF_SETUP")
if not stat then error() end
if val == "" then
          if not inc_shared_data("DO_SENSOR_AVREF_SETUP") then error() end
          if not script_exec("SENSOR_AVREF_SETUP","","") then error() end
end
-------------------------------------------------------------
-- ADC Vref 値をデータベースから取得する
-------------------------------------------------------------
stat,val = get_permanent_data("AVref_" .. g_params["RemoteDeviceName"])
if not stat then error() end
adc_vref = tonumber(val);


-------------------------------------------------------------
-- 温度計算
-------------------------------------------------------------
if g_params["SerialNumber"] and g_params["RemoteDeviceName"] and g_params["ActivityLevel"] and
```

```
g_params["TemperatureLevel"] then
          temperature = (100 * adc_vref * tonumber(g_params["TemperatureLevel"])) / 1024;
          log_msg(string.format("%s IR-count = %s temperature = %2.3g adc_vref = %2.4g",
            g_params["RemoteDeviceName"],g_params["ActivityLevel"],temperature,adc_vref),file_id)
else
          log_msg("*ERROR* parameter error",file_id);
          error();
end;
------------------------------------------------------------
-- デバイス名とシリアル番号をデータベースに登録
------------------------------------------------------------
stat = add_permanent_strlist("TDCPDeviceList",g_params["SerialNumber"] .. "," ..
g_params["RemoteDeviceName"],true)
if not stat then error() end
------------------------------------------------------------
-- サンプリング値をデータベースに登録
------------------------------------------------------------
key = "SENSOR_IR_" .. g_params["SerialNumber"]
val = tonumber(g_params["ActivityLevel"])

stat = add_stat_data(key,val)
if not stat then error() end

key = "SENSOR_TP_" .. g_params["SerialNumber"]
stat = add_stat_data(key,tostring(temperature))
if not stat then error() end
------------------------------------------------------------
-- ACTIVITIES_CHECK で、IR センサーが
-- 0 以上のカウンタ値を取得したかどうかをチェックするための
-- フラグを設定する
------------------------------------------------------------
if val > 0 then
          if not set_shared_data("RESIDENTS_ACTIVITIES_FLAG","1") then error() end
end
```

ファイル名(REGISTER_ACTIVITY.lua) で DeviceServer のスクリプトフォルダ

“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.3 SENSOR_AVREF_SETUP スクリプト作成

前述の REGISTER_ACTIVITY スクリプトから1 回だけコールされて、センサーデバイスごとのリファレンス電圧値を DeviceServer のデータベースに格納するためのスクリプトを作成します。

センサーデバイスの TDCP ベースボードの AVRef 値をデジタルテスタで測定して、その値をスクリプト中に記入してデータベースに登録します。データベースに登録した AVRef 値は A/D 変換値から温度を計算するときに、他のスクリプトやクライアントプログラムから使用されます。

```
File_id = "SENSOR_AVREF_SETUP"
--[[
TDCP デバイスの A/D 変換に使用される A/D Vref 値をデータベースに保存する。クライアントプログラム(Flash)
からこの値を取得して、A/D ポートに接続された各種センサーの検出値を計算するときに利用する。A/D Vref 値が
変化した場合には必ずこのスクリプトを修正した後、手動で実行してください。DeviceServer の再起動時に、
REGISTER_ACTIVITYスクリプト中から一回だけこのスクリプトが自動で実行されます。

データベースに保存するパラメータ
--------------------------------------------------------------------------------
キー値      値                                                          値の例
--------------------------------------------------------------------------------
AVref_<DeviceName>
          <DeviceName> で指定された XBee デバイス(Node Identifier)
          が接続されたTDCP デバイスの A/D Vref 値を指定する
          デバイスが複数ある場合には全ての A/D Vref 値を同様に指定する
                                                                      "5.00"
--------------------------------------------------------------------------------
]]
log_msg("start..",file_id)

-- A/D Vref 値をクライアント側で利用するためにデータベースに保存する
if not set_permanent_data("AVref_Device1","4.90") then error() end
if not set_permanent_data("AVref_Device4","5.01") then error() end
```

ファイル名(SENSOR_AVREF_SETUP.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.4 PERIODIC_TIMER スクリプト作成

PERIODIC_TIMER スクリプトは、DeviceServer で１分ごとに起動されます。
このスクリプト中に、データベース中の古いセンサーデータを消去する SENSOR_DATA_PURGE スクリプトと、一定期

間、センサーデバイスの赤外線(人体)センサーからの検出が無かった場合を判断するための ACTIVITIES_CHECK スクリプトを２時間ごと起動するような記述を追加します。

共有変数 “TIME_2H” は、2時間(120分)をカウントするために使用しています。（別の名前を使用しても構いません）

```
file_id = "PERIODIC_TIMER"
local stat,val
------------------------------------------------------------
-- 2時間に一回 SENSOR_DATA_PURGE, ACTIVITIES_CHECK
-- スクリプトを実行する
------------------------------------------------------------
stat,val = inc_shared_data("TIME_2H")
if not stat then error() end
if (tonumber(val) == 1) then
          if not script_fork_exec("SENSOR_DATA_PURGE","","") then error() end
          if not script_fork_exec("ACTIVITIES_CHECK","","") then error() end
end
if (tonumber(val) > 120) then
          stat = set_shared_data("TIME_2H","")
          if not stat then error() end
end
```

ファイル名(PERIODIC_TIMER.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.5 SENSOR_DATA_PURGE スクリプト作成

前述の PERIODIC_TIMER スクリプトから２時間ごとにコールされて、過去30 日よりも古いサンプリングデータを削除します。データベースの拡張を制限する目的でスクリプトを設定しますが、ここで削除されたサンプリングデータよりも以前のデータは集計できなくなります。スクリプトを変更して保存対象となる日数を自由に変更することもできます。

```
file_id = "SENSOR_DATA_PURGE"
-- 30 日以前のデータを削除する
log_msg("start..",file_id)
local now = os.date "*t"
local stat,yyyy,mm,dd = inc_day(-30,now["year"],now["month"],now["day"])
local timestamp = string.format("%4.4d/%2.2d/%2.2d 23:59:59",yyyy,mm,dd)
```

```
if not clear_stat_data("SENSOR_",timestamp,"") then error() end
```

ファイル名(SENSOR_DATA_PURGE.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.6 ACTIVITIES_CHECK スクリプト作成

前述の PERIODIC_TIMER スクリプトから２時間ごとにコールされて、一定期間、センサーデバイスの赤外線(人体)センサーからの検出が無かった場合にアラームメールを送信します。

スクリプト中のメール送信アドレス設定部分を修正してメールの宛先を正しく設定してください。

```
mail_addr = "不在警報メール宛先 <your-mail-addr@mail-domain.com>"
```

赤外線データを検出したかどうかのフラグ(共有データ) は “RESIDENTS_ACTIVITIES_FLAG” をチェックしています。
このフラグは REGISTER_ACTIVITY スクリプトからセットされます。

検出対象となる時間帯を変更するときには、スクリプト中の現在時間を時間判断している部分を変更して下さい。

```
file_id = "ACTIVITIES_CHECK"
log_msg("start..",file_id)
local mail_addr = "不在警報メール宛先 <your-mail-addr@mail-domain.com>"
local flag_name = "RESIDENTS_ACTIVITIES_FLAG"
-----------------------------------------------------------------------
-- 前回このスクリプトを実行してから、今回このスクリプトを実行する
-- までの間に、IR カウンタの値を全く検出していない場合には、警告メールを
-- 送信する。このスクリプトの実行間隔は PERIODIC_TIMERスクリプト中で
-- 設定されている。
-----------------------------------------------------------------------
local now = os.date "*t"
local stat,flag = get_shared_data(flag_name)
if not stat then error() end
-----------------------------------------------------------------------
-- 検出対象となる時間帯を 07:00 - 22:00 の日中に限定している
-----------------------------------------------------------------------
if (flag == "") and (now["hour"] >= 7) and (now["hour"] <= 22) then
        local body = {}
        table.insert(body,"一定期間IRセンサーからの検出がありませんでした")
        if not mail_send(mail_addr,"","不在警報",unpack(body)) then error() end
        log_msg("residents activities warning",file_id)
```

```
end
if not set_shared_data(flag_name,"") then error() end
```

ファイル名(ACTIVITIES_CHECK.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.7 IR_SENSOR_ALERT_MAIL スクリプト作成

前述の XBEE_TDCP_DATA スクリプトから “CHANGE_COUNT_EXCEED” イベントパケットを受信したときにコールされて、センサーデバイスの赤外線(人体)センサーで、あらかじめ決められた値よりも多くカウンタ値の検出があった場合に、アラームメールを送信します。

カウンタ値が超えた場合のイベント検出を開始するための IR_WATCH_START スクリプトが DeviceServer のメールコマンド機能で実行されていたときは、その時のメール送信元に対してアラームメールを送信するようにします。もし、IR_WATCH_START スクリプトがメールコマンド以外の方法で実行されていたときには、IR_WATCH_START スクリプト中に記述したメールアドレスに送信します。このときに使用するメール送信アドレス設定部分を修正して、メールの宛先を正しく設定してください。

```
mail_addr = "センサー警報メール宛先 <your-mail-addr@mail-domain.com>";
```

```
file_id = "IR_SENSOR_ALERT_MAIL";
--[[
IR_SENSOR_ALERT_MAIL スクリプト起動時に渡されるパラメータ
-------------------------------------------------------------------------------
キー値              値                                    値の例
-------------------------------------------------------------------------------
RemoteDeviceName    監視用リモートデバイス名              LivingRoom
SerialNumber        監視用リモートデバイスのシリアル番号  0013A200404AC397
ActivityLevel       赤外センサーで関知した動作レベル(カウンタ値)  120
-------------------------------------------------------------------------------
]]
log_msg("start..",file_id);
-----------------------------------------------------------------------
-- IR_WATCH_START スクリプトをメールコマンドから実行していた場合には、
-- メールの宛先をその時のメールコマンドリプライ先に設定する
-----------------------------------------------------------------------
local stat,mail_addr = get_shared_data("IR_SENSOR_ALERT_MAIL_ADDR")
if not stat then error() end
if mail_addr == "" then
```

```
        mail_addr = "センサー警報メール宛先 <your-mail-addr@mail-domain.com>";
end
-----------------------------------------------------------------------
-- 警報メール送信
-----------------------------------------------------------------------
if g_params["RemoteDeviceName"] and g_params["SerialNumber"] and g_params["ActivityLevel"] then
        local body = {};
        table.insert(body,"IRセンサーデバイスのカウンタ値が上限値を超えました");
        table.insert(body,"デバイス名:" .. g_params["RemoteDeviceName"]);
        table.insert(body,"カウンタ値:" .. g_params["ActivityLevel"]);
        stat = mail_send(mail_addr,"","IRセンサー警報",unpack(body));
        if not stat then error() end;
else
        log_msg("パラメータが指定されていません",file_id);
end;
```

ファイル名(IR_SENSOR_ALERT_MAIL.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.8 IR_WATCH_START スクリプト作成

デジタル入力カウンタ値(赤外線センサー)が、設定値を超えたときに発生するイベントを有効にするための IR_WATCH_START スクリプトを作成します。

サンプリングデータを定期的に送信しているすべてのセンサーデバイスに対して、TDCP コマンド “change_count_check,1” を送信してイベント監視状態にします。デバイス一覧はデータベース(キー名 “TDCPDeviceList”) を検索してデバイス名を取得します。このデバイス名は REGISTER_ACTIVITY スクリプト中からサンプリングデータをデータベースに登録するときに同時に登録されています。

スクリプトパラメータに “CounterReset” が指定されているとイベント監視状態にする前に現在のカウンタ値を 0 にクリアします。これは、既に赤外線センサーで検出済みだった場合にこのスクリプトを実行すると直ぐにアラームメールが送信されてしまうのを回避するために指定します。ただし、カウンタ値をクリアすると定期的なサンプリングで集計している赤外線センサーがクリアされることになりますので、集計結果に影響があります。

メールコマンドからスクリプトが起動された場合には、そのメール送信元アドレスを共有変数に保存します。これは後で、カウンタ値が超えた場合のイベント検出で送信されるアラームメールの宛先に利用されます。

```
file_id = "IR_WATCH_START"
--[[
```

```
IR_WATCH_START スクリプト起動時に渡されるパラメータ
--------------------------------------------------------------------------------
キー値              値                                          値の例
--------------------------------------------------------------------------------
CounterReset        値は空文字以外の任意の文字列                "1"
                    パラメータ指定時には各センサーのカウンタ監視を有効に
                    する前に現在のカウンタ値を 0 にする
                    このパラメータを指定すると、定期的に取得するセンサー集計データ
                    のカウンタ値もクリアされる。
--------------------------------------------------------------------------------
]]
log_msg("start..",file_id)
------------------------------------------------------------------------
-- メールコマンドでIR_WATCH_STARTが実行された場合は、警報メールの宛先
-- をメールコマンドのリプライアドレスに変更するために、共有データに
-- リプライアドレスを保存しておく
------------------------------------------------------------------------
if g_params["$REPLYTO$"] then
        if not set_shared_data("IR_SENSOR_ALERT_MAIL_ADDR",g_params["$REPLYTO$"]) then error() end
else
        if not set_shared_data("IR_SENSOR_ALERT_MAIL_ADDR","") then error() end
end


------------------------------------------------------------------------
-- 統計用データベースに保管された全てのTDCP デバイスを対象とする
------------------------------------------------------------------------
local stat,strlist = get_permanent_strlist("TDCPDeviceList")
for key_dev,val_dev in ipairs(strlist) do
        device = csv_to_tbl(val_dev)
        ------------------------------------------------
        -- IR センサーのカウンタ値を強制的にクリアする
        ------------------------------------------------
        if g_params["CounterReset"] then
                if not xbee_tdcp_safe_retry(device[2],"change_count_reset") then error() end
        end
        ------------------------------------------------
        -- IR センサカウンタ値チェックを有効にする
        ------------------------------------------------
        if not xbee_tdcp_safe_retry(device[2],"change_count_check,1") then error() end
```

```
end
```

ファイル名(IR_WATCH_START.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.9 IR_WATCH_STOP スクリプト作成

カウンタ値が超えた場合のイベント検出を中止するための IR_WATCH_STOP スクリプトを作成します。

カウンタ値が超えたときのイベントが発生した場合には、自動的にセンサーデバイスのイベント監視状態は解除されますのでこのスクリプトを実行する必要はありません。

サンプリングデータを定期的に送信しいるすべてのセンサーデバイスに対して、TDCP コマンド
“change_count_check,0” を送信してイベント監視を解除します。デバイス一覧はデータベース(キー名
“TDCPDeviceList”) を検索してデバイス名を取得します。このデバイス名は REGISTER_ACTIVITY スクリプト中からサンプリングデータをデータベースに登録するときに同時に登録されています。

```
file_id = "IR_WATCH_STOP"
log_msg("start..",file_id)
--------------------------------------------------------------------
-- 統計用データベースに保管された全てのTDCP デバイスを集計対象とする
--------------------------------------------------------------------
local stat,strlist = get_permanent_strlist("TDCPDeviceList")
for key_dev,val_dev in ipairs(strlist) do
        device = csv_to_tbl(val_dev)
        ------------------------------------------------
        -- IR センサカウンタ値チェックを無効にする
        ------------------------------------------------
        if not xbee_tdcp_safe_retry(device[2],"change_count_check,0") then error() end
end
```

ファイル名(IR_WATCH_STOP.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.10 SENSOR_DATA_TO_FILE スクリプト作成

クライアントPC のエクセルで表やグラフ表示をさせるための CSV ファイル形式の集計データを作成します。この CSV ファイルを作成するために、サーバーのデータベースに保存されているセンサーデータを集計するスクリプト(SENSOR_DATA_TO_FILE) を作成します。

集計対象日付は、スクリプトパラメータ “TargetDate” で指定されたものを使用しますが、省略時にはスクリプトを実行した日が対象になります。集計結果の出力ファイル名は “OutFile” パラメータで指定します。

データベースから集計するときには センサー名のキーを指定して指定期間内のセンサーデータ（温度、赤外線センサーカウント値）を 10分ごとに集計します。集計は summary_stat_data() 関数を使用して、合計値・平均値・最大値・最小値を、各集計時間間隔で出力します。

サンプリングデータを定期的に送信しているすべてのセンサーデバイスに対して、集計計算を行って CSV ファイルに出力します。

```
file_id = "SENSOR_DATA_TO_FILE"
--[[
SENSOR_DATA_TO_FILE スクリプト起動時に渡されるパラメータ
-------------------------------------------------------------------------------
キー値              値                                      値の例
-------------------------------------------------------------------------------
TargetDate          集計対象日付(YYYY/MM/DD)                "2010/01/31"
                    パラメータ省略時はスクリプトが起動された日付が
                    集計対象日になる
OutFile             集計結果(CSV)を出力するファイル名       "e:/summary.csv"
                    パラメータ省略時は "c:/summary.csv"になる
-------------------------------------------------------------------------------
]]
log_msg("start..",file_id)
-----------------------------------------------------------------
-- TargetDate パラメータが指定されていない場合にはスクリプトが起動
-- された日を集計対象にする
-----------------------------------------------------------------
local timestamp
if g_params["TargetDate"] then
    timestamp = g_params["TargetDate"] .. " 0:0:0"
else
          local now = os.date "*t"
          timestamp = string.format("%4.4d/%2.2d/%2.2d 0:0:0",now["year"],now["month"],now["day"])
end

local outfile
if g_params["OutFile"] then
```

```
        outfile = g_params["OutFile"]
else
        outfile = "c:/summary.csv"
end
-------------------------------------------------------------------
-- 出力ファイルオープン
-------------------------------------------------------------------
local file;
file = io.open(outfile,"w+");
if (file == nil) then error() end;
-------------------------------------------------------------------
-- 統計用データベースに保管された全てのTDCP デバイスを集計対象とする
-------------------------------------------------------------------
local key,val,key_dev,val_dev
local device
local stat,strlist = get_permanent_strlist("TDCPDeviceList")
for key_dev,val_dev in ipairs(strlist) do
        device = csv_to_tbl(val_dev)
        log_msg("SerialNumber=" .. device[1] .. " NodeIdentifier=" .. device[2] ,file_id)
        ---------------------------------
        -- 温度データの集計を行う
        ---------------------------------
        stat,datetime,sample,total,mean,max,min = summary_stat_data("SENSOR_TP_" .. device[1],
                                                    timestamp,600,144)
        if not stat then error() end
        for key,val in ipairs(datetime) do
                file:write(string.format("TEMPERATURE,%s,%s,%d,%2.3g,%2.3g,%2.3g¥n",
      device[2],val,sample[key],mean[key],max[key],min[key]))
        end

        ---------------------------------
        -- 赤外センサカウンタの集計を行う
        ---------------------------------
        stat,datetime,sample,total,mean,max,min = summary_stat_data("SENSOR_IR_" .. device[1],
                                                    timestamp,600,144)
        if not stat then error() end
        for key,val in ipairs(datetime) do
                file:write(string.format("IR_COUNT,%s,%s,%d,%4.5g,%4.5g,%4.5g¥n",
      device[2],val,sample[key],mean[key],max[key],min[key]))
```

```
        end


end
-------------------------------------------------------------------
-- 出力ファイルクローズ
-------------------------------------------------------------------
file:close();
```

ファイル名(SENSOR_DATA_TO_FILE.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.11 SENSOR_DATA_TO_MAIL スクリプト作成

携帯電話の電子メールやインターネットメールを経由して、センサーデータの集計を行うスクリプトを作成します。

機能は、前述のエクセル集計用の CSV ファイルを出力する SENSOR_DATA_TO_FILE スクリプトと同様ですが、メールで結果を送信するために適した集計フォーマットにしています。

集計対象日付は、スクリプトパラメータ “TargetDate” で指定されたものを使用しますが、省略時にはスクリプトを実行した日が対象になります。

データベースから集計するときには センサー名のキーを指定して指定期間内のセンサーデータ（温度、赤外センサーカウント値）を 10分ごとに集計します。集計対象となるのは、平均温度と赤外線センサーカウント平均・最大値です。サンプリングデータを定期的に送信しているすべてのセンサーデバイスに対して集計を行った後、スクリプト実行をリクエストしてきたメールアドレスに、集計結果が入ったメールを返信します。

```
file_id = "SENSOR_DATA_TO_MAIL"
--[[
SENSOR_DATA_TO_MAIL スクリプト起動時に渡されるパラメータ
-------------------------------------------------------------------------------
キー値              値                                  値の例
-------------------------------------------------------------------------------
TargetDate          集計対象日付(YYYY/MM/DD)            "2010/01/31"
                    パラメータ省略時はスクリプトが起動された日付が
                    集計対象日になる
このスクリプトはメールコマンドから実行されることを想定しています。
-------------------------------------------------------------------------------
]]
log_msg("start..",file_id)
```

```
-------------------------------------------------------------------------
-- TargetDate パラメータが指定されていない場合にはスクリプトが起動
-- された日を集計対象にする
-------------------------------------------------------------------------
local timestamp
if g_params["TargetDate"] then
    timestamp = g_params["TargetDate"] .. " 0:0:0"
else
          local now = os.date "*t"
          timestamp = string.format("%4.4d/%2.2d/%2.2d 0:0:0",now["year"],now["month"],now["day"])
end
-------------------------------------------------------------------------
-- 統計用データベースに保管された全てのTDCP デバイスを集計対象とする
-------------------------------------------------------------------------
local key,val,key_dev,val_dev
local device
local t_stat,t_datetime,t_sample,t_total,t_mean,t_max,t_min
local ir_stat,ir_datetime,ir_sample,ir_total,ir_mean,ir_max,ir_min
local body = {}

local stat,strlist = get_permanent_strlist("TDCPDeviceList")
for key_dev,val_dev in ipairs(strlist) do
          device = csv_to_tbl(val_dev)
          log_msg("SerialNumber=" .. device[1] .. " NodeIdentifier=" .. device[2] ,file_id)
          ---------------------------------------
          -- 温度データ集計
          ---------------------------------------
          t_stat,t_datetime,t_sample,t_total,t_mean,t_max,t_min =
                   summary_stat_data("SENSOR_TP_" .. device[1],timestamp,600,144)
          if not t_stat then error() end
          ---------------------------------------
          -- 赤外センサカウンタ集計
          ---------------------------------------
          ir_stat,ir_datetime,ir_sample,ir_total,ir_mean,ir_max,ir_min =
                  summary_stat_data("SENSOR_IR_" .. device[1],timestamp,600,144)
          if not ir_stat then error() end
          ---------------------------------------
          -- 集計データ出力
          ---------------------------------------
```

```
        table.insert(body,"---------------------------------")
        table.insert(body,"デバイス " .. device[2] .. "  日付: " .. string.sub(timestamp,1,10))
        table.insert(body,"サンプル  温度 IR平均/ IR最大")
        table.insert(body,"---------------------------------")
        for key,val in ipairs(t_datetime) do
                if t_sample[key] == 0 then
                        table.insert(body,string.format("%s -",string.sub(val,12,16)))
                else
                        table.insert(body,string.format("%s
[%d] %4.1f %6.1f/%6.1f",string.sub(val,12,16),t_sample[key],t_mean[key],ir_mean[key],ir_max[key]))
                end
        end
end
-------------------------------------------------------------------------
-- 集計結果をメールで送信する
-------------------------------------------------------------------------
if not mail_send(g_params["$REPLYTO$"],"","センサーデータ集計",unpack(body)) then error() end
```

ファイル名(SENSOR_DATA_TO_MAIL.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

5.12 SENSOR_FORCESAMPLE_MAIL スクリプト作成

携帯電話の電子メールやインターネットメールを経由して、現在のセンサーデータのサンプリング値を取得してメールで送信するためのスクリプトを作成します。

サンプリングデータを定期的に送信しているすべてのセンサーデバイスに対して、マニュアルサンプリングを実行します。すべてのセンサーデバイスのサンプリング結果をまとめて、スクリプト中からメールで送信します。

マニュアルサンプリングは TDCP コマンド “force_sample” を実行して現在の A/D 変換値とデジタル入力ポートのカウント値を取得します。その後、A/D 変換用 AVRef 電圧値をデータベースから取得して温度を計算しています。AVRef 電圧値は SENSOR_AVREF_SETUP スクリプトで登録されたものを利用しています。

```
file_id = "SENSOR_FORCESAMPLE_MAIL"
--[[
このスクリプトはメールコマンドから実行されることを想定しています。
]]
log_msg("start..",file_id)
local key_dev,val_dev
```

```
local device
local t_stat,t_result
local body = {}
local device_info
-----------------------------------------------------------------------
-- 統計用データベースに保管された全てのTDCP デバイスを集計対象とする
-----------------------------------------------------------------------
local stat,strlist = get_permanent_strlist("TDCPDeviceList")
for key_dev,val_dev in ipairs(strlist) do
        device = csv_to_tbl(val_dev)
        log_msg("SerialNumber=" .. device[1] .. " NodeIdentifier=" .. device[2] ,file_id)

        ---------------------------------------------------------------
        -- 現在の温度データ,赤外センサカウンタ を取得
        ---------------------------------------------------------------
        t_stat,t_result = xbee_tdcp_safe_retry(device[2],"force_sample")
        if not t_stat then error() end

        ---------------------------------------------------------------
        -- ADC Vref 値をデータベースから取得する
        ---------------------------------------------------------------
        local avref_stat,avref_val = get_permanent_data("AVref_" .. device[2])
        if not avref_stat then error() end
        local adc_vref = tonumber(avref_val);
        ---------------------------------------------------------------
        -- 温度計算
        ---------------------------------------------------------------
        local temperature = (100 * adc_vref * tonumber(t_result[9]   )) / 1024;
        ---------------------------------------------------------------
        -- メッセージ出力
        ---------------------------------------------------------------
        device_info = string.format("デバイス名 %s  赤外センサーカウント %s  温
度 %2.3g",device[2],t_result[5],temperature)
        table.insert(body,device_info)
        log_msg(device_info,file_id)
end
-----------------------------------------------------------------------
-- サンプリング値をメールで送信する
-----------------------------------------------------------------------
```

```
if not mail_send(g_params["$REPLYTO$"],"","現在のセンサーデータ",unpack(body)) then error() end
```

ファイル名(SENSOR_FORCESAMPLE_MAIL.lua) で DeviceServer のスクリプトフォルダ
“C:¥Program Files¥AllBlueSystem¥Scripts” に保管します。

6 Web クライアントプログラム設定

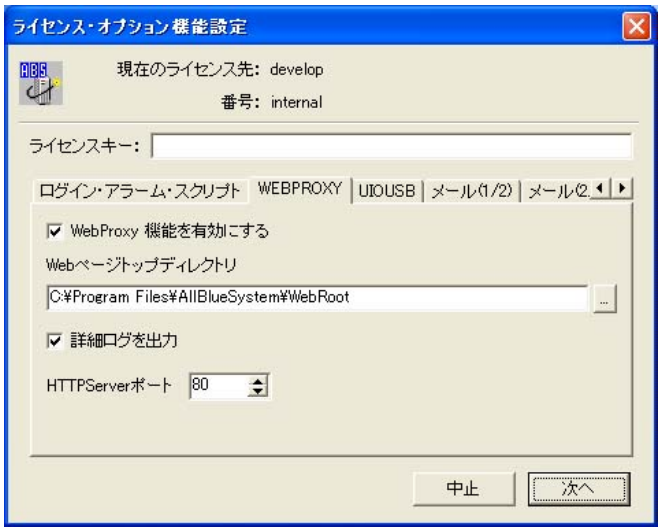
インターネットまたはLAN 上の Webブラウザから、現在のセンサーデータのサンプリング値と指定した日付のセンサーデータグラフを参照するための IRSensorMgr プログラムの設定方法について説明します。

6.1 DeviceServer の WebProxy セットアップ

DeviceServer には Webブラウザの Flash アプリケーションからコマンド操作をするためのインターフェイス WebProxy機能があります。DeviceServer インストール時に WebProxy のセットアップを行っていない場合は、下記の手順で設定を行います。

最初に、DeviceServer が動作している PCで既に HTTP サーバープログラムが動作していないことを確認してください。ポート番号80(http) でWebProxy が動作しますので、マイクロソフト社製 HTTPサーバー(IIS)や、Apache などのHTTPサーバープログラムを既に使用している場合には、同一 PC 上でDeviceServer の WebProxy 機能を動作させることはできません。これらの既存の HTTP サーバーとDeviceServer のWebProxy 機能を両方共使用したいときには、それぞれを別のPC に分離して設置します。設置の方法については、”DeviceServer ユーザーマニュアル” 中の “DeviceServerとHTTPサーバーを分離して設定する” の項を参照してください。

サーバー設定プログラム(ServerInit.exe) プログラムをメニューから選択して実行します。”WEBPROXY” 設定タブの内容を下記のようにして、”次へ” ボタンを続けて押していってサーバーの設定を完了してください。DeviceServer が再起動して WebProxy 機能が有効になります。



6.2 プログラムインストール

DeviceServer インストール時に、”スクリプト操作” など Webブラウザから実行する幾つかの Flash プログラムがインストールされています。ここでは、センサーデバイス操作を行うための Flash プログラムを DeviceServer に追加設定します。

添付ファイルの “WebRoot” フォルダ中の下記の２つのファイルを DeviceServer をインストールしたフォルダにコピーしてください。(デフォルトフォルダ “C:¥Program Files¥AllBlueSystem¥WebRoot¥remote”)

追加するファイル

- IRSensorMgr.swf
- IRSensorMgr.html

DeviceServer 自身の持つ http サーバー機能を使用しないで、別の http サーバープログラムを使用する場合や、http サーバーの動作する PC とDeviceServer の動作する PC が別ドメインにある場合には crossdomain.xml ファイルと servercfg.xml ファイルの設定が必要になります。詳しい設定方法は、”DeviceServer ユーザーマニュアル” 中の “DeviceServerとHTTPサーバーを分離して設定する” の項を参照してください。

6.3 ユーザーアカウントの設定

Webブラウザから実行する Flash アプリケーションは、ログイン認証を行ってから操作できるようになります。WebProxy経由でログイン可能にするために、クライアントプログラムを起動してユーザー管理プログラムを実行します。操作を行うユーザー情報のアプリケーション許可フラグ中の “WebLogin” にチェックを付けて下さい。

インストールする Flash プログラム IRSensorMgr.swf(センサー管理プログラム) は、プログラムの実行権限をユーザーが持っているかどうかのチェックを行っています。このため、ログインユーザー情報のアプリケーション許可フラグに“IRSensorMgr” フラグを追加指定する必要があります。ユーザー管理プログラムで、ログインユーザーを選択した後、”アプリケーション登録” ボタンを押して “IRSensorMgr” フラグを追加してください。

“OK” を押してユーザー情報を修正します。この状態で センサーアプリケーションが使用できる状態になっています。

WebProxy で設定を行った PC の Webページをアクセスしてログインしてください。

デフォルトの URL は http://localhost/remote/IRSensorMgr.html になります。

7 このドキュメントについて

7.1 著作権および登録商標

Windows 、Visual Basic および Excel は米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。ここではExcel® をエクセル、Visual Basic® for Applications をVBAと表記する場合があります。

XBee XBee® and XBee□PRO® are registered trademarks of Digi, Inc.

7.2 連絡先

オールブルーシステム (All Blue System)

ウェブページ http://www.allbluesystem.com

メール contact@allbluesystem.com

7.3 このドキュメントの使用について

このドキュメントは、ABS-9000 DeviceServer の一般的な使用方法と応用例について解説してあります。お客様の個別の問題について、このドキュメントに記載された内容を実際のシステムに利用するときには、ここに記載されている以外にも考慮する事柄がありますので、ご注意ください。特に安全性やセキュリティ、長期間にわたる運用を想定してシステムを構築する必要があります。

オールブルーシステムでは ABS-9000 DeviceServer の使用や、このドキュメントに記載された内容を使用することによって、お客様及び第三者に損害を与えないことを保証しません。 ABS-9000 DeviceServer を使用したシステムを構築するときは、お客様の責任の下で、システムの構築と運用が行われるものとします。

8 更新履歴

REV A.1.0 2010/3/31

初版作成