UECS-Pi SDK デベロッパーガイド





2016/05/11

【改定履歴】

版	改定内容	改定日
1.0	・初版作成	2014/11/03
1.1	・Raspberry Piの I2C ポートの有効化方法を追記	2014/11/07
	・その他誤記修正	
1.2	・ライブラリバージョンアップに伴う、サンプルコードの微修正	2016/05/11
	・その他誤記修正	

目次

1. UECS-Pi SDK とは	5
1.1. 概要	$\dots 5$
1.2. UECS-Pi SDK の機能	6
2. 開発環境セットアップ	9
2.1. Eclipse セットアップ	9
2.2. Eclipse への UECS-Pi SDK セットアップ	12
2.3. Eclipse 上での Tomcat セットアップ	14
2.4. Eclipse 上での UecsPi_SDK 実行	19
3. 基本ライブラリ、フレームワーク概説	20
3.1. 全体ライブラリ構成	20
3.2. API ドキュメント(JavaDoc)、ソースコード	$\dots 22$
3.3. ノード、CCM 送受信機能	23
3.3.1. ノード基本機能	23
3.3.2. CCM と CCM サービス	24
3.4. デバイス、コンポーネント機能	24
3.4.1. ノード、デバイス、コンポーネントの概念	24
3.4.2. ノード、デバイス、コンポーネントの設定ファイル	26
3.4.3. デバイス通信・制御のライブラリ	26
3.4.4. コンポーネントと CCM	27
3.4.5. ノード、デバイス、コンポーネント、CCM の動作	27
3.4.6. Raspberry Pi へのデバイス接続方法	30
3.5. WebUI 機能	33
3.5.1. 基本ページレイアウトとレイアウトの継承	33
3.5.2. WebUI 出力のためのファイルセット	35
3.5.3. 基本 WebUI 一式	37
4. チュートリアル	42
4.1. プロジェクト構成	42
4.2. ノード、WebUIアプリケーションクラス	43
4.2.1. ノードクラス	43
4.2.1. WebUI アプリケーションクラス	45
4.3. Sample 1: ダミーセンサ	46
4.3.1. デバイス、コンポーネントクラス	47
4.3.1. WebUI 設定画面クラス	49
4.4. Sample 2: タイマー動作アクチュエータ	53
4.4.1. デバイス、コンポーネントクラス	55
4.4.2. WebUI 設定画面クラス	57
4.5. Sample 3:シリアル通信	58

	4.5.1.	デバイス、コンポーネントクラス	59
	4.5.2.	WebUI 設定画面クラス	61
4	.6. San	nple 4:I2C 通信	62
	4.6.1.	デバイス、コンポーネントクラス	63
	4.6.2.	WebUI 設定画面クラス	65
5.	UECS-F	アi アプリケーションのインストール	66
5	.1. Ras	pberry Pi セットアップ	66
	5.1.1.	SD カードの準備	67
	5.1.2.	Raspberry Pi の初期設定	69
	5.1.3.	Raspberry Pi のユーザ設定	72
	5.1.4.	IP アドレス設定	73
	5.1.5.	DNS 設定	74
	5.1.6.	シリアル通信(UART)の有効化	75
	5.1.7.	I2C ポートの有効化	76
	5.1.8.	Tomcat の設定	.77
	5.1.9.	各種スクリプトの転送と登録	82
5	.2. アフ	プリケーションインストール	84
	5.2.1.	WAR ファイルファイルの作成	84
	5.2.2.	実機インストール	86
	5.2.3.	web.xml について	. 88
6.	利用ライ	、 センス	. 89
7.	免責事項	Į	. 89
8.	お問い合	わせ	.89

1. UECS-Pi SDK とは

1.1. 概要

UECS-Pi(ウエックスパイ)は、英国ラズベリーパイ財団によって開発された、ARM プロセッサを搭載 したシングルボードコンピュータ「Raspberry Pi(ラズベリーパイ) Model B/B+/2」上で動作する、ユビ キタス環境制御システム(UECS: Ubiquitous Environment Control System)実用通信規約 Ver1.00-E10 (以下、E10 規格) 準拠のソフトウェアです。

UECS-Pi SDK は、UECS-Pi のオープンソース版であり、E10 規格に準拠したソフトウェアを作成し、 UECS 機器を自作するためのソフトウェア開発キットです。商用製品にも利用可能なオープンソースライセ ンスのライブラリを用いていますので、企業・個人に関わらず開発者は無償で利用できます。開発者は、 UECS-Pi SDK で提供されるベースコードに、接続したいデバイスの制御コードと Web UI コードを追加で 作成するだけで、短期間で E10 規格準拠の UECS 機器の作成が可能です。



🗵 1 : Raspberry Pi Model B



🗵 2 : Raspberry Pi Model B+

1.2. UECS-Pi SDK の機能

UECS-Pi SDK は開発者に以下の機能を提供します。

No.	説明
1	E10 規格に則って動作するノードと CCM の基本ライブラリ
2	ノード設定、CCM 情報確認、ログ確認等が行える基本 Web UI 機能一式
3	Raspberry Pi に接続したデバイスを制御するための基本ライブラリ、フレームワーク
4	接続したデバイス用の Web UI を作成するための基本ライブラリ、フレームワーク

表 1: UECS-Pi SDK が提供する主機能

開発者はこれらの機能を用いて UECS 機器を開発します。例えば様々なセンサや、アクチュエータを接続して動作する UECS 機器を作成できます。UECS-Pi SDK では下図のように、Raspberry Pi 基板上の GPIO 端子や USB シリアル変換アダプタで接続されたセンサデバイスを UECS 仕様のセンサとして利用す ることが可能です。また GPIO 端子からアクチュエータに接続して、各種制御機器を動作させる事も可能で す。



図 3:センサ接続イメージ



図 4:アクチュエータ接続イメージ

UECS-Pi SDK には、上図にあるようなデバイスを制御するための基本ライブラリ、フレームワークが用意されています。

Raspberry Pi に接続したセンサやアクチュエータの設定は、WebUI から行う事が可能です。WebUI も 基本ライブラリ、フレームワークが用意されています。



図 5: Web UI イメージ

UECS-Pi SDK を使って開発できる UECS 機器の全体動作イメージを示します。



図 6: UECS-Pi SDK で開発する UECS 機器の動作イメージ

2. <u>開発環境セットアップ</u>

2.1. Eclipse セットアップ

本章では、Windows 環境での開発を例に説明します。Java が動作する Linux、Mac OS 等の環境でも開発 可能です。

No.	名称	説明	ダウンロード URL
1	Pleiades	オープンソースの統合開発環境Eclipseに各種プラ	http://mergedoc.sourceforge.jp/
		グインを統合した日本語版パッケージ	
		表 2: Eclipse セットアップに必要なソフトウ	ェア一覧

Pleiades の配布ページに移動し、Pleiades ALL in One ダウンロードから Eclipse 4.3 Kepler をクリックします。

MerceDoc Project	en 23/ma 5,646,4	67 downloads SCIENTICROP
Pleiades プラグイン日本語化プラグイン	Pleiades - Eclipse プラグイン日本語化プラグイン	Dynamic ADP Translation Container for Eclipse
JStyle 改行タブ表示プラグイン	Pleiades All in One ダウンロード	
MergeDoc / Javadoc 日本語化	Pleiades All in One は Windows 向けに開発対象プログラミング言語別でパッケージング グインのセットです。ダウンロードした zip ファイルを解束 (<u>zio 解発時の注音)</u> し、eclipse Eclipse を利用できます。Linux や Mac などで日本語化する場合は、Pleiades プラグイン本	した Eclipse 本体と Pleiades および使利なブラ .exe を起動すれば、すぐに日本語化された 体をダウンロードしてください。
フォーラム	Eclipse 4.4 Luna Pleiades All in One Pleiades All in One	Cone Eclipse 3.7 Indigo
チケット	Felloro 2.6 Holion	Ediaco 2.2 Europa
プロジェクト Wiki	Pleiades All in One Pleiades All in One Pleiades All in One	One Pleiades All in One
Qiita	ダウンロード数比較(1環酸集計)	
プログ	バージョン x バッケージ別ダウンロード数 bit	(数別ダウンロード数
	Pattorn Umade Dor Pipe Pipe Pipe Pipe	9 520A 6407
	Pleiades ブラグイン・ダウンロード	
	図 7: Pleiades 配布ページ	

② Java の Full Edition をダウンロードします。32bit か 64bit かは開発用 PC に応じて選択して下さい。



図 8: Pleiades ダウンロードページ

 ③ exe ファイルがダウンロードされるので実行します。下のポップアップが出た場合は「実行」ボタンを 押して下さい。

開い	ているファイル - セキュリティの警告
я	羌行元を確認できませんでした。このソフトウェアを実行しますか?
	名前:yashi¥Downloads¥pleiades-java-4.3.2-sfx (2).exe 発行元: 不明な発行元 種類: アプリケーション 発信元: C;¥Users¥kobavashi¥Downloads¥pleiades-java-4
	実行(R) キャンセル
	┚このファイル閉く前に常に警告する(₩)
	20ファイルには、発行元を検証できる有効なデジタル署名がありません。信頼で きる発行元のソフトウェアのみ実行してください。 <u>実行することのできるソフトウェアの</u> 詳細を表示します。
	図 9 : exe ファイル実行時ポップアップ

④ Pleiades 関連ファイルの解凍先を尋ねられるので「参照」ボタンから指定します。指定を終えた場合か、初期の「C:¥pleiades」で良い場合は「解凍」ボタンを押すと指定フォルダにファイルが展開されます。

● 日本語 Eclipse Pleia	des All in One Java 自己經東書庫
	 解凍ボタンを押すと解凍を開始します。 参照ボタンで解凍先フォルダを選択できます。また、直接入力することもできます。 解凍先フォルダが存在しない場合は、解凍前に自動的にに作成します。
PLEIADES ALL IN ONE Eclipse	解演先(0) C¥dev¥pleiades] 参照(₩) 解演状況
	解凍 キャンセル
	<u>図 10:解凍画面</u>

⑤ 解凍後 Pleiades の解凍先フォルダに移動し「eclipse」フォルダの「eclipse.exe」をダブルクリックします。

	2m *	第45日 時	1846	#77		
🚖 お気に入り	6.61	30411193	GEAR	542		
ConeDrive	🍌 configuration	2014/10/14 10:32	ファイル フォル			
タウンロード	🍶 dropins	2014/03/21 21:49	ファイル フォル			
■ デスクトップ	🍌 features	2014/10/14 10:32	ファイル フォル			
「日日の市市」の場所	🍶 jre	2014/03/21 21:52	ファイル フォル			
111 111 11 11 11 11 11 11 11 11 11 11 1	🎍 p2	2014/02/24 6:02	ファイル フォル			
	🌲 plugins	2014/03/21 21:49	ファイル フォル			
21229	🍶 readme	2014/03/21 21:49	ファイル フォル…			
□ ドキュメント	.eclipseproduct	2014/02/12 8:03	ECLIPSEPRODU	1 KB		
1 ビクチャ	artifacts.xml	2014/10/14 10:32	XML ドキュメント	237 KB		
EF7	eclipse.exe	2013/10/09 14:09	アプリケーション	312 KB		
1 ミュージック	Seclipse.exe -clean.cmd	2013/09/07 10:55	Windows コマン…	1 KB		
	eclipse.ini	2014/10/14 10:32	構成設定	1 KB		
■ コンピューター	eclipsec.exe	2013/10/09 14:09	アプリケーション	24 KB		
BOOTCAMP (C:)						
WinData (D:)						
MacData (Y:)						
- Mar(05 (7:)						
() = 10000 (L1)						
■ ネットワーク						

図 11:解凍先フォルダ画面

⑥ Eclipse 起動時に、ワークスペースを作成するフォルダを指定する画面が開きます。ワークスペース とは、Eclipse で作成したプロジェクトのリソースを格納するフォルダです。これは後で変更可能で す。指定が終わったら「OK」ボタンを押すと Eclipse が起動します。

🗑 ワークスペース・ラン	チャー		×
ワークスペースの選択			
Eclipse は、ワークス・ このセッションに使用	ペースと呼ばれるフォルダーにプロジェクトを保管します。 するワークスペース・フォルダーを選択してください。		
ワークスペース(W):	/workspace	▼ 参照(B)	
🔲 この選択をデフォノ	レトとして使用し、今後この質問を表示しない(U)		
		ок <i>‡</i> т>tл	,

図 12: ワークスペースフォルダ指定画面

⑦ 起動が完了すると以下の画面になります。これで Eclipse のセットアップは完了です。



図 13: Eclipse 初期画面

2.2. <u>Eclipse への UECS-Pi SDK セットアップ</u>

No.	名称	説明	ダウンロード URL
1	UECS-Pi SDK	UECS機器開発のためのオープンソースSDK	http://www.wa-bit.com/business_info-2/
			smart-agri/uecs-pi
	表 3:Ecli	pse への UECS-Pi SDK セットアップに必	要なソフトウェア一覧

- ① UECS-Pi SDK の配布ページに移動し、zip ファイルをダウンロードして適当な場所に解凍します。
- ② Eclipse を開き、画面左上の「ファイル」メニューの中の「インポート」をクリックします。

Ċ, I	ava - Ech se		
ファ	イル(F) 編集(E) ソース(S)	リファクタリング(T)	ナビ
	<u>が帽(*)</u> ファイルを開く(.)	Alt+Shift+N ►	C)
	閉じる(C) すべて閉じる(L)	Ctrl+W Ctrl+Shift+W	48
R.	保管(S) 別名保管(A)	Ctrl+S	
R	すべて保管(E) 前回保管した状態に戻す(T)	Ctrl+Shift+S	
ď	移動(V) 名前変更(M)	F2	
8	リフレッシュ(F)	F5	
200 200	増択リソースのカウント タブ <-> スペースの変換		
	行区切り文字の変換(V)	•	
-	印刷(P)	Ctrl+P	
	ワークスペースの切り替え(W) 再開	+	
è	インボート(I)		
-	プロパティー(R)	Alt+Enter	
	終了(X)		
_			-

③ 「一般」の中の「既存プロジェクトをワークスペースへ」をクリックし、「次へ」ボタンをクリックし ます。

インボート					
選択 アーカイブ・ファイルま	たはディレクトリーから	新規プロジェクトで	を作成します。		Ľ
インポート・ソースの選択	R(s):				
フィルター入力					
 ● 一般 ● アーカイブ・ ● ファイル・シック ● ファイル・シック ● ファインク・ ● 登録プロジェ ■ 協士 > ● CV5 > ● 目8 > ● GRt > ● Maven 	ファイル ステム セット クトをワークスペースへ				L L
1		< 戻る(8)	次へ(N) >	完了(F)	キャンセル

図 15:インポート画面 1

 ④ ルート・ディレクトリの選択で先ほど zip ファイルを解凍したフォルダ(UecsPi_SDK フォルダ)を選 択し「完了」ボタンを押します。

プロジェクトのインポート 既存の Eclipse プロジェクトを検索するディレクトリーを選択します。	
●ルート・ディレクトリーの選択(T): C:¥Users¥kobayashi¥Desktop¥UecsPi_SDK ◆	参照(R)
 ○ アーカイブ・ファイルの選択(A): プロジェクト(P): 	参照(R)
UecsPi_SDK (C:¥Users¥kobayashi¥Desktop¥UecsPi_SDK)	すべて選択(S)
	選択をすべて解除(D) リフレッシュ(E)
□ プロジェクトをワークスペースにコピー(C)	
ワーキング・セット 「ワーキング・セットにプロジェクトを追加(T)	
ワーキング・セット(0):	選択(E)
(N) > 売了(F)	キャンセル

図 16:インポート画面 2

⑤ パッケージ・エクスプローラーの中に UECS-Pi SDK プロジェクトが追加されて表示されます。これで Eclipse への UECS-Pi SDK セットアップは完了です。この時点では Tomcat の定義がされていないの でエラーが出ています。



図 17: UECS-Pi SDK プロジェクト追加完了画面

2.3. Eclipse上でのTomcatセットアップ

① Eclipse を開き「ウィンドウ」メニューの「設定」をクリックします。



図 18:ウィンドウメニュー画面

② 設定ウィンドウが表示されるので「サーバー」の「ランタイム環境」を選択し「追加」ボタンをクリックします。

📄 設定								
フィルター入力	サーバー・ランタイム環境	Ê	⇔ • ⇔ • •					
▷ 一般	サーバー・ランタイム環境を追加、除去、または編集します。							
AmaterasUML	サーバー・ランタイム環境(N):							
⊳ Ant	名前	型	追加(A)					
Checkstyle ▷ DBViewer プラグイン	🖥 J2EE プレビュー	J2EE プレビュー	編集(E)					
⊳ Java			除去(R)					
▷ Java EE ▷ Java 永続化								
JavaScript			検索(S)					
 LimyEclipse Mayen 			 列(C)					
⊳ Mylyn								
NTail OuickPEx								
Tomcat								
> Web			_					
▷ Web サービス ▷ WindowBuilder								
> XML								
▷ インストール/更新			-					
⊿ サ −/(−								
	L							
ランタイム環境								
起動								

図 19: ランタイム環境設定画面 1

③ 「Apache」から「Apache Tomcat v7.0」を選択して、「新規ローカル・サーバーの作成」にチェック を入れたら「次へ」ボタンをクリックします。

新規サーバー・ランタイム環境	
新規サーバー・ランタイム環境を定義します	
追加サーバー・アダプ ランタイム環境のタイプを選択(R):	<u> プターのダウンロード</u>
フィルター入力	
Apache Apache Apache Apache Tomcat v3.2 Apache Tomcat v4.0 Apache Tomcat v4.1 Apache Tomcat v5.0 Apache Tomcat v5.5 Apache Tomcat v5.6 Apache Tomcat v7.0 Apache Tom	× E
☑ 新規□−カル・サーバーの作成(C)	
? < 戻る(B) 次へ(N) > 完了(F) #	マンセル

図 20: ランタイム環境設定画面 2

④ Tomcat インストールディレクトリを、Pleiades フォルダの「tomcat」フォルダ内にある「7」フォル ダに設定し「完了」ボタンをクリックします。

⑧ 新規サーバー・ランタイム環境
インストール・ティレクトリーを指定してくたさい
名前(M):
Apache Tomcat v7.0
Dutata 2 2 (S=1) - 1 (A 22 (S=1)) - C:¥dev¥pleiades¥tomcat¥7 参照(R)
apache-tomcat-7.0.47 ダウンロードしてインストール…
JRE(J):
▼ 12×1~10/800 JRc(1)
< 戻る(B)

図 21: ランタイム環境設定画面 3

⑤ サーバが追加されるので選択して、「編集」ボタンをクリックします。

🕑 設定			
フィルター入力	サーバー・ランタイム環境		⇔ • ⇒ • •
▷ 一般	サーバー・ランタイム環境を	追加、除去、または編集します。	
AmaterasERD		、 、	
AmaterasUML	サーハー・ランタイム壊現(N):	
⊳ Ant	<u>4</u> #	#	追加(A)
Checkstyle	🕆 Apache Tomcat v7.0	Apache Tomcat v	(
▶ DBViewer フラグイン		200 70 22	福葉(E)
⊳ Java			除去(R)
> Java EE			
▷ Jāvā 水税化			検索(S)
JavaScript LimyEdipso			
LimyEclipse			列(C)
> Maven			
⊳ NTail			
QuickREx			
▶ Tomcat			
> Web			
▷ Web サービス			
WindowBuilder			
> XML			
▷ インストール/更新			
> コード・リコメンダー			
⊿ サーバー			
Overlays			
プロファイラー			
ランタイム環境			
音声			
起動			
ターミナル			

図 22: ランタイム環境設定画面 4

⑥ 「JRE」を Java7 に変更し「完了」ボタンをクリックします。

🕑 サーバー・ランタイム環境の編集	0	Perce-	an Acces		
Tomcat サーバー インストール・ディレクトリーを指定して	ください				
名前(M):					
Apache Tomcat v7.0					
Tomcat インストール・ディレクトリー(D):					
C:¥dev¥pleiades¥tomcat¥7					参照(R)
				apache-tomcat-7.0.47	ダウンロードしてインストール
JRE(J):					
java7				•	インストール済みの JRE(I)
ワークベンチ・デフォルト JRE					
javao java7					
3	< 戻る(B)	次/	\(N) >	完了(F)	キャンセル

図 23: ランタイム環境設定画面 5

これでサーバの設定は完了です。「OK」ボタンをクリックします。

図 24: ランタイム環境設定画面 6

⑧ UecsPi_SDK プロジェクトを右クリックし「プロパティー」をクリックします。

パッケーシ		クスプローラー 10	8410 700	3
Uecs	ni er	(規(W) (規(W)		表示
		新規ウィンドウで騙く(N) 型階層を聞く(N) 表示(W)	Alt+S	F4 hift+W •
> Go re > Go st > Go W		コピー(C) 体飾名のコピー(Y) 貼り付け(P) 削除(D)		Ctrl+C Ctrl+V Delete
	b.	コンデキストから除去 ビルド・パス(B) ソース(S) リファクタリング(T)	Ctrl+Alt+Shift Alt+ Alt+	+Down Shift+S • Shift+T •
	2 2	インボート(I) エクスポート(0)		
	\$	バクを検索 リフレッシュ(F) プロジェクトを開じる(S) 無関係なプロジェクトを閉じる(U) ワーキング・セットの割り当て(A).		, F2
		Windows エクスプローラーで歸く コマンド・プロンプトで歸く プロファイル(P) 実行(R) 検証(V) 比較(A) ローカル・ヒストリーからの儀元(*)	r)	· · · ·
		Quick JUnit(Q) Checkstyle Java EE ツール		:
Г		チーム(E) 日本 プロパティー(R)	Alt	+Enter

⑨ プロパティーから「Java コンパイラー」をクリックし、JDK 準拠を 1.7 に変更します。



図 26: Java コンパイラー設定画面

① 「サーバー」をクリックし、常に使用するサーバに作成したサーバを設定して「OK」ボタンをクリックします。これで Eclipse 上での Tomcat セットアップは完了です。

🕃 UecsPi_SDK のプロパティー		
フィルター入力	ーパーゼ	⇔ • ⇔ • •
 > リソース AnyEdit ツール Checkstyle DBViewer プラグイン FindBugs > Java エディター > Java エディター > Java コンパイラー 	プロジェクト: UecsPi_SDK (J2EE Web モジュール) このプロジェクトを実行するときは常に以下のサーバーを使用(F): <なし> 目 ローカル・ポスト の J2EE プレビュー 倍 ローカル・ポスト の Tomcat v7.0 サーバー	
Java のビルド・パス Javadoc ロケーション JavaScript Jsp ブラグメント Limy Tormat Web Content の設定値 Web プロジェクトの設定		
Web ページ・エディター > XDoclet サービス・ポリシー ターゲット・ランタイム タスク・タグ > タスク・リポジトリー デプロイメント・アセンブ		
ビルダー プロジェクト・ファセット プロジェクト参照 プロパティー・エディター リファクタリング・ヒスト 検証 実行/デパッグ設定	デフォルトの寝元(D)	達用(A)
?	ОК ‡	アンセル

図 27:サーバー設定画面

2.4. <u>Eclipse 上での UecsPi_SDK 実行</u>

① UecsPi_SDK プロジェクトを Eclipse 上で実行したい場合は、プロジェクトを選択して「デバック」もしくは「実行」ボタンをクリックします。(初回起動時に実行方法選択ダイアログが表示されますので、「サーバーで実行」を選択してください)



図 28: UecsPi SDK の実行

② 実行するとログイン画面が Eclipse 上に表示されます。「admin」と入力すればログイン出来ます。

Second Se		
	vabit.uecs.pi.webui.signin.SignInPage	▶ 🔮
	MyAppNode	
Login	管理バスワード	
Powerd by U	UECS ®. (http://www.uecs.jp/)	

図 29: Eclipse 上でのログイン画面

Eclipse 上での UECS-Pi SDK 実行は、作成した Web UI コードなどの動作テストに使えます。開発 PC に接続できない機器との通信テストは Eclipse 上では行えませんので、Raspberry Pi 実機にインストールし てテストする必要があります。実機を使った実行方法は、「<u>5. UECS-Pi アプリケーションインストール</u>」以降で説明します。

- 3. 基本ライブラリ、フレームワーク概説
 - 3.1. 全体ライブラリ構成

UECS・Pi SDK の全体は、E10 規格をベースにした UECS 基本ライブラリをコアとし、そこに Raspberry Pi 上で UECS・Pi SDK が基本的な動作を行うためのライブラリ(以下、UECS・Pi 基本ライブラリ)をラッ プした構成になっています。UECS・Pi SDK を使用する開発者に提供されているのが、この2層構成ライブ ラリから成っているフレームワークです。開発者はそのフレームワークをベースにコードを作成する事とな ります。

4.Raspbe	erry Pi 用	13.接続するデバイス& コンポーネント制御機能			14.接続するデ ^{ィッ} れ 用 WebUI 機能	
ノード基本機能		5.Raspberry Pi 用デバイス& コンポーネント基本機能			7.WebUI 基本機能	
1.ノード 2.CCM		3.7° ^ 17&	0 D: 4 I	6.DB 機能		11.Wicket
基本機能	機能	基本機能	□2/# -秋/ 8.Pi4J 基本機能		QLite	12.Tomcat

図 30: UECS-Pi SDK の全体ライブラリ構成

上図の 1~3 が UECS 基本ライブラリ、4~12 が UECS-Pi 基本ライブラリ、13~14 が開発者追加コード です。UECS-Pi 基本ライブラリのノード、デバイス、コンポーネント関連の機能は UECS 基本ライブラリ を拡張して作成されています。また開発者が追加コードを作成する時は、基本的には UECS-Pi 基本ライブ ラリのクラスを拡張してコードを作成します。

開発者追加コードは大きく分けると「接続するデバイスを制御するコード」と「接続デバイス用の WebUI コード」の2種類あります。そのため UECS-Pi SDK では、デバイス接続用と WebUI 作成用の2つのライ ブラリを提供しています。

No.	分類	名称	主な機能
1	UECS基本ライブラリ	ノード基本機能	ネットワーク通信、CCMサービス、デバイス管理機能
2	(uecs-core.jar)	CCM機能	CCMサービス、各種CCMの定義
3		デバイス&コンポー	デバイス基本動作、コンポーネント基本動作
		ネント基本機能	

以下表に、UECS-Pi SDK を構成するライブラリの機能概要を示します。

4	UECS-Pi基本ライブラリ	Raspberry Pi用ノー	DB連携、DB登録済みデバイスの起動
	(uecs-pi-core.jar)	ド基本機能	
5		Raspberry Pi用デバ	I2C、シリアル通信(UART、USB・シリアル)、GPIO接
		イス&コンポーネン	続を使った、デバイスとコンポーネントの基本動作
		ト基本機能	
6		DB機能	ノード、デバイス、コンポーネント等のデータ処理
7		WebUI基本機能	UECS-Pi SDKの基本WebUI一式
8	UECS-Pi基本ライブラリ	Pi4J	Raspberry Piに接続したデバイスをJavaから制御する
	(サードパーティ製)		参照URL : <u>http://pi4j.com/</u>
9	UECS-Pi基本ライブラリ	OrmLite	ORMマッパー
	(サードパーティ製)		参照URL : <u>http://ormlite.com/</u>
10	UECS-Pi基本ライブラリ	SQLite	軽量RDBMS
	(サードパーティ製)		参照URL: <u>https://bitbucket.org/xerial/sqlite-jdbc</u>
11	UECS-Pi基本ライブラリ	Wicket	WebUI用のWebフレームワーク
	(サードパーティ製作)		参照URL : <u>https://wicket.apache.org/</u>
12	UECS-Pi基本ライブラリ	Tomcat	WebUIやWicketが動作するHTTPアプリケーションサー
	(サードパーティ製)		バ
			参照URL : <u>http://tomcat.apache.org/</u>
13	開発者作成コード	接続するデバイス&	開発者が接続するデバイス用の制御コード
		コンポーネント制御	
		機能	
14	開発者作成コード	接続するデバイス用	開発者が接続するデバイス用のWebUIコード
		WebUI機能	

表 4: UECS-Pi SDK を構成するライブラリの機能概要

3.2. API ドキュメント(JavaDoc)、ソースコード

開発時に利用する主要なライブラリ群の API ドキュメントとソースコードは、UecsPi_SDK プロジェク ト内の「src_3rd」フォルダ内に含まれています。必要に応じて、Eclipse の JavaDoc・ソースコードのバイ ンド機能を用いて参照することが可能です。また、ライブラリ自体を自由にカスタマイズすることも可能で す。



図 31: JavaDoc/ソースコード jar

3.3. <u>ノード、CCM</u>送受信機能

3.3.1. ノード基本機能

UECS では、制御したい環境内にコンピュータとセンサやアクチュエータを設置し、イーサネットで接続します。この機器は「ノード」と呼ばれます。ノード同士は、E10 規格に則った XML データの相互通信を行い、計測や制御を自律的に行います。この XML データは「UECS 共用通信子(UECS-CCM)」と呼ばれます。以下に UECS の模式図を示します。



図 32 : UECS ノードが配置されたガラスハウスの模式図

※UECS についての詳細はユビキタス環境制御システム研究会 HP(<u>http://www.uecs.jp/</u>)を参照して下さい。
 ※CCM についての詳細は UECS 通信実用規約 1.00-E10(<u>http://www.uecs.jp/uecs/kiyaku/UECSStandard100_E10.pdf</u>)を参照して下さい。

例えば上図では、気温センサから CCM が流れ、暖房機のノードがそれを受け取って点火したり、換気 窓が閉じたり、状況確認・設定端末に気温が表示されるといった事が可能です。

UECS 基本ライブラリの中心になっているのが、この機能を実装した CCM 機能です。CCM 機能は、 内部で CCM 受信と CCM 周期送信を行います。この機能により UECS-Pi SDK は他 UECS 機器と連携 を行います。

また UECS-Pi SDK のノードは、ノード自身のセットアップ(ホストの IP 設定、ノード状態通知 CCM(end.kNN)登録、ブロードキャストアドレスの設定)や、DB に登録されているデバイス起動を行い ます。

3.3.2. CCM と CCM サービス

CCM 機能の周期送信処理では CCM が周期的に送信されます。複数の CCM を送信する事も可能です。 ノードは「CCM サービス」を使い、CCM を送信します。CCM サービスとは、CCM の状態を管理する サービスクラスです。ノード、CCM サービス、CCM の関係は下図のようになっています。



図 33: ノード、CCM サービス、CCM の関係

Ccm クラスの派生クラスが、com.wabit.uecs.ccm、com.wabit.uecs.protocol パッケージに含まれてい ますが、AbstractUecsNode 内部で自動的に利用されますので、開発者が通常のアプリケーション開発で 直接これらの Ccm 派生クラスを意識する必要はありません。

- 3.4. デバイス、コンポーネント機能
 - 3.4.1. ノード、デバイス、コンポーネントの概念

UECS-Pi SDK にデバイスを接続する際は、デバイスだけでなく、ノードとコンポーネントを考えま す。ノード、デバイス、コンポーネントは概念です。1つのノードには複数のデバイスを所属させられ、 1つのデバイスには複数のコンポーネントを所属させられます。



図 34:ノード、デバイス、コンポーネントの関係

例えば電流・電圧・電力計測が出来る INA226 モジュールを Raspberry Pi に接続して UECS-Pi SDK として動作させるとすると、見た目の上では Raspberry Pi がノード、INA226 がデバイス、INA226 に付いている電流・電圧・電力を測るそれぞれのセンサがコンポーネントとなります。



図 35 : INA226 接続イメージ

配布直後の UECS-Pi SDK は何も接続されていないノードとなっています。開発者はそこに様々なデ バイスを接続し、デバイスとデバイスに関連付くコンポーネントのプログラムを作成する事となります。



3.4.2. ノード、デバイス、コンポーネントの設定ファイル

図 36:ノード、デバイス、コンポーネントと各設定ファイルの関係

ノード、デバイス、コンポーネントは、各々設定ファイルを持っています。設定ファイルでは定数の設 定を行い、ノード、デバイス、コンポーネントの各ファイルでそれを利用してプログラムを記述していま す。UECS・Pi SDK では設定を key / value の形で扱っています。プログラムを記述する側のファイルで は、設定ファイルに記述された key を呼び出すことで、その key に関連付く value を利用します。

3.4.3. デバイス通信・制御のライブラリ

UECS-Pi SDK のフレームワークに用意されているデバイス用の各種通信方式は I2C、シリアル通信 (UART、USB-シリアル)です。また、GPIO 接続によるアクチュータ制御用のライブラリも用意され ています。開発者は各方式の基本ライブラリクラスを拡張して、接続するデバイス制御用のオリジナルク ラスを作成します。そこに接続するデバイスに関連付くコンポーネント用のクラスを作成する事で、 UECS-Pi SDK に接続するデバイスとコンポーネントを動作させる準備が出来ます。これらのライブラリ クラスは、UECS 基本ライブラリの派生クラスとして、com.wabit.uecs.pi パッケージ以下に含まれてい ます。また、開発した UECS 機器の上でデバイスやコンポーネントの設定値をユーザーに変更させる必 要がある場合は、WebUI の作成を行う事になります。(チュートリアルの章で詳細を説明します)

3.4.4. コンポーネントと CCM

コンポーネントをノード内で動作させると、必ず対応する CCM も動作します。コンポーネントが起動 する時に、自らに関連付く CCM サービスをノードに登録するためです。1つのコンポーネントに所属す る CCM の数は、コンポーネントの種類によって変わります。

コンポーネントの種類は、センサ、アクチュエータ、コントローラの3種類です。以下にそれぞれの機 能を示します。

クラス分類	機能	登録される CCM の種類
SensorComponent	センサー機器と直接通信してCCM送信、あるいは他のUECS	DataCcm(受信/送信)
の派生クラス	センサーノードからのCCM受信でセンサー値を受信する。	
ActuatorComponent	リレー制御やモーター制御により、接続された制御機器を動	OprCcm(送信側)
の派生クラス	作させる。ノード自身の自律動作以外に、コントローラから	RcACcm (受信側)
	の遠隔制御CCMを受信して動作する。	RcMCcm(受信側)
ControllerComponent	アクチュエータとネットワーク経由でCCMのやり取りを行	OprCcm (受信側)
の派生クラス	って遠隔コントロールする。	RcACcm(送信側)
	※例えばツマミがひねられたらコントローラがCCMを送信するようにし	RcMCcm(送信側)
	ておき、アクチュエータ側にそのCCMが来たら機器を作動させるプログ	
	ラムを入れておけば、間接的な機器制御が行える。	

表 5: 各コンポーネントの機能

これらの CCM がコンポーネントによってノードに登録される事により、ノードの周期送信機能がそれ らの CCM を送信する事や、WebUI の CCM 一覧確認画面からそれらの CCM を確認する事が可能になり ます。

3.4.5. <u>ノード、デバイス、コンポーネント、CCMの動作</u>

これまでに紹介したノード、デバイス、コンポーネント、CCMの関係を図示します。



また、ノード、デバイス、コンポーネント、CCMの動作は一連の流れになっています。UECS-Pi SDK の起動プロセス~定常動作にそれが現れています。以下に図示します。



図 38: UECS-Pi SDK 起動プロセスシーケンス

順序	動作名称	動作内容
1	電源投入	電源投入され、OS→Tomcatが起動。Tomcatはweb.xmlで定義されている
		WebUI初期処理実行クラスを起動
2	WebUI初期処理実行	WebUI初期処理実行クラスはノードとWebUIトップ画面を起動
	クラス起動	
3	ノード起動1	ホストIP設定、ノード用CCM登録、ブロードキャストアドレス設定を行う
4	ノード起動2	ノードが所属するデバイスを起動
5	デバイス起動	各デバイスが所属コンポーネントを起動
6	コンポーネント起動	各コンポーネントがCCMを作成し、CCMサービスの形でノードに登録する
7.1	ノード定常動作開始	ノードが定常動作開始。CCMサービスに登録済のCCMの周期送信と、CCM受

		信を行う
7.2	コンポーネント定常	コンポーネントが定常動作開始。何か変化があった時にデータ(センサの取得
	動作開始	値や、アクチュエータの動作値)を更新し、同時に所属するCCMの値も更新す
		る

表 6: UECS-Pi SDK 起動プロセス解説

またノード再起動の時は、それまで登録されていたデバイスや CCM サービスはリセットされ、改めて DB からデバイスやコンポーネントが呼び出されて動作を開始します。

3.4.6. Raspberry Pi へのデバイス接続方法

センサデバイスを Raspberry Pi に接続する時は以下のイメージのように接続します。



図 39:センサ接続イメージ

(※) GPIO接続のセンサーユニットは、5Vまたは3.3V電源、3.3V TTLレベルI/O対応のものであれば、直接接続可能です。

次にアクチュエータの接続方法を示します。アクチュエータで制御可能な機器は、リレーを1つ利用した[ON/OFF]によるスイッチ制御アクチュエータが最大8系統、リレーを2つ利用した[ON/OFF/正/逆]による[0%~100%]のポジション制御アクチュエータが最大4系統まで接続できます。



図 40:アクチュエータ接続イメージ

GPIO端子は以下のようにアサインされているので、利用目的に沿った形でセンサやリレーユニットと 接続して下さい。

25	23	21	19	17	15	13	11	9	7	5	3	1
26	24	22	20	18	16	14	12	10	8	6	4	2

図 41: GPIO ピン番号

ピン番号	名称	機能説明	備考
1	3.3V DC Power	3.3V 電源	
2	5.0V DC power	5.0V 電源	
3	SDA0(I2C)	I2C シリアルデータ (SDA)	
4	5.0V DC Power	5.0V 電源	
5	SCL0(I2C)	I2C シリアルロック (SCL)	
6	GND	グランド	
7	GPIO 7	デジタル OUT [L /H]	
8	TxD	UART TX	
9	GND	グランド	
10	RxD	UART RX	
11	GPIO 0	デジタル OUT [L /H]	
12	GPIO 1	デジタル OUT [L /H]	
13	GPIO 2	デジタル OUT [L /H]	
14	GND	グランド	
15	GPIO 3	デジタル OUT [L /H]	
16	GPIO 4	デジタル OUT [L /H]	
17	3.3V DC Power	3.3V 電源	
18	GPIO 5	デジタル OUT [L /H]	

19	SPI MOSI	SPI 用ピン
20	GND	グランド
21	SPI MISO	SPI 用ピン
22	GPIO 6	デジタル OUT [L /H]
23	SPI SCLK	SPI 用ピン
24	SPI CE0 N	SPI 用ピン
25	GND	グランド
26	SPI CE1 N	SPI 用ピン

<u>表 7: GPIO 端子アサイン表</u>

3.5. WebUI 機能

UECS-Pi SDK の WebUI は、フレームワークに Apache Wicket (以降 Wicket) を利用しています。Wicket はオブジェクト指向型の Web アプリケーションフレームワークで、Web ページのデザインや、Web ページ の部品をオブジェクトとして扱います。これらは独自拡張可能で、Wicket はこのようなオブジェクトを組 み合わせて様々な Web ページを作成します。また UECS-Pi SDK では基本的な機能を持つページをあらか じめ準備しています。

3.5.1. 基本ページレイアウトとレイアウトの継承

WebUI 作成の際は、親ページクラスのレイアウトを継承して、子ページクラスを作成します。例えば UECS-Pi SDK のトップページ、CCM 一覧ページ等の全ての画面ページは、LayoutPage というクラス を継承する事で、そのレイアウトを継承しています。LayoutPage は、HeaderPanel、MenuPanel、 CostomFeedbackPanel、FooterPanel といった部分表示クラスを内包しています。



図 42: LayoutPage クラスと他の全ての WebUI クラスの関係

LayoutPage は、ベースのレイアウトクラスとして機能するために、LayoutPage.html という対になる HTML ファイルを持っています。これが継承されるレイアウトの大枠になります。同様に、内包する部 分表示クラスにも対になる HTML ファイルが存在します。これらが合成された結果が最終的に HTML コードとして出力されます。



図 43: LayoutPage.java が形成しているレイアウト

上図の右側にあるのが、LayoutPage.html に集約されているページ部品です。これらの HTML ファイ ルに出力されるコンテンツや、ユーザ操作時の挙動は、対になる Java クラスに記載されています。

すなわち Wicket では、Java クラスはデータや動作を記述するファイルとして、HTML ファイルは Java クラスに記述されたデータや動作を出力するテンプレートファイルとして機能します。



図 44: Wicket における Java ファイルと HTML ファイルの模式図

これら LayoutPage、ヘッダ、メニューパネル、コンテンツ、フッタの Java ファイルと HTML ファイルは、ページ部品のクラスライブラリとして機能します。

基本ページー式の全てのページは、この LayoutPage.html のレイアウトを継承した上で、コンテンツ 部分に各々のページ独自の内容を出力しています。これが Wicket の機能です。

開発者が接続デバイス用 WebUI を作成する際も、基本ページー式と同じように LayoutPage.java を継承したクラスファイルを作り、このレイアウトを継承して作成していきます。継承したレイアウトをデザインのベースとして使う事で、開発者はコンテンツ部分に出力する接続デバイス用コンテンツ作成に注力出来ます。

またレイアウトを構成する Java クラスや HTML ファイルを独自に作成して差し替える事により、基本レイアウトを変更した WebUI を作成する事も可能です。(詳細は<u>チュートリアル</u>を参照)

3.5.2. WebUI 出力のためのファイルセット

Wicket では、基本的に Java ファイル (xxx.java) と HTML ファイル (xxx.html) が、1 つの Web ペ ージを出力するためのファイルセットになります。また多言語表示対応させるためのリソースプロパティ ファイル (xxx.properties) をそこに加える事も出来ます。これらが連携して Web リクエストが来た時に 表示される WebUI を作成します。

⊿	ر ا	myapp.webui.sample1	
	\triangleright	DummyDevicePage.java	
		DummyDevicePage_en.properties	
		DummyDevicePage_ja.properties	
		📄 DummyDevicePage.html	

図 45:Webページ出力のためのファイルセット例

各ファイルの機能概要は以下の通りです。

順序	動作名称	動作内容
1	Javaファイル	HTMLファイルに表示するデータや、HTTPリクエスト処理の挙動が記述されて
		いる
2	HTMLファイル	Javaファイルに記載した内容を反映するテンプレートになる。またプロパティフ
		ァイルのテキストを表示するテンプレートにもなっている

3	プロパティファイル	各言語に対応したテキストー式が記述されている。日本語で表示する時には日本
		語用 (_ja)の、英語で表示する時には英語用 (_en)のプロパティファイルのテ
		キストが、HTMLファイルに表示される。これはブラウザのロケールによって切
		り替わる。多言語対応させる必要がない場合は、HTMLファイルに直接テキスト
		を書いても良い。

表 8:WebUIを構成する各ファイルの機能概要
3.5.3. 基本 WebUI 一式

ここでは UECS-Pi SDK に始めから用意されている基本 WebUI 一式のコンテンツと使い方を解説しま す。基本 WebUI は以下の 5 機能です。また UECS-Pi SDK はこれらの WebUI が持つ機能に加え、ログ アウト機能を持っています。

No.	名称	機能
1	認計ページ	認証を得ていないユーザが UECS-Pi にアクセスした時に表示されるページ
2	トップページ	ログイン直後、あるいは上部メニューの「トップ」をクリックすると表示され
		るページ
3	CCM 一覧ページ	CCMの状態-覧を確認できるページ
4	状態ログページ	ノードやデバイスの状態を確認できるページ
5	セットアップページ	上部メニューのセットアップにマウスを重ねた際に、プルダウンで出てくるメ
		ニューをクリックすると表示されるページ。初期はノード用のみ。開発者はこ
		こに接続デバイス用の WebUI を作成する

表 9:基本ページ機能概要

3.5.3.1. トップページ

ログイン直後、あるいは上部メニューの「トップ」をクリックすると表示されるページです。 UECS-Pi SDK の動作状態や、接続されているセンサとアクチュエータの情報が一覧化されて表示さ れます。「更新」ボタンをクリックすると、表示されている値が更新されます。

	MyA	ppNode	
ップ CCM一覧	状態ログ	セットアップ	ログアウト
["4七時間			
F 1八版 IPアドレス 192.168.1.101 現在	時刻 2014-11-03 16:09:21	状態 正常	
ノサー	5 A-	38-1-14	Think
1 Dummy Sensor	名称	現在他	更新日時 2014-11-03 14:39:45
2 Serial Port Sensor		%	2014-11-03 14:39:45
3 I2C Bus Sensor		ppm	2014-11-03 14:39:45
フチュエーター	動作モード スタンドアロン	現在値 OFF	更新日時 2014-11-03 14:39:45
	Powerd by UECS	D. (http://www.uecs.jp/)	

図 46:トップページ

3.5.3.2. <u>認証ページ</u>

MyAppNode
-Login- 管理パスワード
Powerd by UECS @. (http://www.uecs.jp/)

認証を得ていないユーザが UECS-Pi にアクセスした時に表示されるページです。

図 47:認証ページ

3.5.3.3. <u>CCM 一覧ページ</u>

上部メニューの「CCM 一覧」をクリックすると表示されるページです。UECS-Pi SDK に登録さ れている CCM(共用通信子)の状態が一覧化されて表示されます。「更新」ボタンをクリックすると、 表示されている値が更新されます。

Lyd CCM-B HMB10 Lyd-y-J LJD701 CM-B No CAMEA No No<	MyAppNode						
No. 24% CCMEX SR Utable E16930 WRV 1 MyAppNode IndaCMC (1-11) [A-15-0] S 0 0 2014-11.03 14:39.45 1 2 Dumny Senor Inda/Turne [1-1) [A-105-0] S 0.0 C 2014-11.03 14:39.45 1 3 Timer Switch [opt] Switchorpt.1 (1-1) [A-105-0] R 0 2014-11.03 14:39.45 1 4 Timer Switch [rcM] SwitchorA.1 (1-1) [S-10.40] R	トップ	CCM一覧	状態ログ	セットア	ップ	ログアウト	
No. 2:4F CCMGE/8 S/R 9/2:46/1 JEMA362 9/JER/9 * 1 MyAppNole end.cMC(1:1)[A:15:0] S 0 0 2014:11:03 14:39:45 * 2 Dummy Sensor InAirTemp(1:1:1)[A:105:0] S 0.0 C 2014:11:03 14:39:45 * 3 Timer Switch [opt] Switchrch.1 (1:1) [S:10:0] R 0 2014:11:03 14:39:45 * 4 Timer Switch [rcA] SwitchrcA.1 (1:1) [S:15:0] R - - - 5 Striat Port Sensor InAirTemu(1:1:1) [A:108:0] R - - - 7 I2C Bus Sensor InAirCO2 (1:1:1) [A:108:0] S ppm 2014:11:03 14:39:45 -	CCM一覧————						
1 MyAppNode cnd.cMC (1-1) (A-15.0) S 0 2014-11.03 14/39.45 2 Dummy Sensor InAirTemp (1-1) (A-105.0) S 0.0 C 2014-11.03 14/39.45 3 Timer Switch (prd) SwitchrcA1 (1-1) (S-1M.0) R 0 2014-11.03 14/39.45 4 Timer Switch (prd) SwitchrcA1 (1-1) (S-1M.0) R 5 Timer Switch (rcA) SwitchrcA1 (1-1) (S-1M.0) R 6 Sarial Port Sensor InAirTempi (1-1) (A-105.0) S % 2014-11.03 14/39.45 7 I2C Bus Sensor InAirCO2 (1-1) (A-105.0) S % 2014-11.03 14/39.45	No. 名称		CCM定義	S/R	現在値	更新時刻	期限切 🔺
2 Dummy Sensor InAirTemp (1-1) [A-105-0] S 0.0 C 2014:11-03 14:39:45 3 Timer Switch [opt] Switchopr.1 (1-11) [A-104-1] S 0 2014:11-03 14:39:45 4 Timer Switch [rcA] Switchopr.1 (1-11) [A-104-0] R 5 Timer Switch [rcA] SwitchoreA1 (1-1) [S-15-0] R 6 Serial Port Sensor InAirHumid (1-1) [A-105-0] S % 2014:11-03 14:39:45 7 I2C Bus Sensor InAirCO2 (1-1:) [A-105-0] S ppm 2014:11-03 14:39:45	1 MyAppNode	cnd.cl	IC (1-1-1) [A-1S-0]	S	0	2014-11-03 14:39:45	
3 Timer Switch (pr] Switchopr.1 (1-1.1) [A-1M-I] S 0 2014-11-03 14/39-45 4 Timer Switch (prA) SwitchorA.1 (1-1.1) [S-1M-0] R	2 Dummy Sensor	InAirT	'emp (1-1-1) [A-10S-0]	S	0.0 C	2014-11-03 14:39:45	
4 Timer Switch [rcA] SwitchrcA.1 (1-1.1) [S-1K-0] R 5 Timer Switch [rcA] SwitchrcA.1 (1-1.1) [S-1S-0] R 6 Serial Pere Searce InA/FRUGO (1-1.1) [A-108-0] S % 7 I2C Bus Sensor InA/FRUGO (1-1.1) [A-108-0] S ppm	3 Timer Switch [opr]	Switch	iopr.1 (1-1-1) [A-1M-1]	S	0	2014-11-03 14:39:45	
5 Timer Switch (rcM) SwitchrcM1 (1-1-1) [S-15-0] R 6 Serial Port Sensor InAirHumid (1-1-1) [A-105-0] S % 7 I2C Bus Sensor InAirCO2 (1-1-1) [A-105-0] S ppm	4 Timer Switch [rcA]	Switch	rcA.1 (1-1-1) [S-1M-0]	R			
6 Senial Port Sensor InAirFumid (1-1:) [A-105-0] S % 2014-11-03 14:39-45 7 I2C Bus Sensor InAirCO2 (1-1:1) [A-108-0] S ppm 2014-11-03 14:39-45	5 Timer Switch [rcM]	Switch	rcM.1 (1-1-1) [S-1S-0]	R		-	
7 [I2C Bus Sensor InAirCO2 (1-1-1) [A-105-0] S ppm 2014-11-03 14:39-45	6 Serial Port Sensor	InAirF	Iumid (1-1-1) [A-10S-0]	S	%	2014-11-03 14:39:45	
· 更新	7 I2C Bus Sensor	InAirO	CO2 (1-1-1) [A-10S-0]	S	ppm	2014-11-03 14:39:45	
							更新
			Toward by ODOS	5. (imp.//ww	nuccos(p/)		
condo oy obco o. (mps/mmacosys/	図 48 : CCM 一覧ページ						

3.5.3.4. <u>状態ログページ</u>

上部メニューの「状態ログ」をクリックすると表示される画面です。各「カテゴリ」に分類された ログ情報が最新順に一覧化されて表示されます。「カテゴリ」には、ノード、デバイス、その他、全 体があります。

		UEOS)-Pi		
トップ	CCM一覧	状態ログ	セットアップ	ログアウト	
- 状態ログ					
カテニシ	ノード・				
時刻	カデニシ	· 10494941-44-44-4	メッセージ		^
2014-10-24 09:29:53	/ - r	ノードか起動されました。			
					-
				クリア	「新
		Powerd by	/ UECS.		

図 49:状態ログページ

ログカテゴリを変更した時に表示される内容を下表に示します。

No.	項目	説明
1	ノード	UECS-Pi 全体のノード機能に関するログ情報が出力されます。主に、起動・
		停止ログ、ネットワーク接続エラー情報などが含まれます。
2	デバイス	設定登録されたセンサやアクチュエータに関するエラー情報などが出力され
		ます。
3	その他	その他付帯機能に関するログが出力されます。
4	全体	全てのカテゴリのログが一覧表示されます。
		<u>表 10:ログカテゴリ一覧</u>

3.5.3.5. セットアップページ

上部メニューのセットアップにマウスを重ねた際に、プルダウンで出てくるメニューをクリックす ると表示される画面です。ノードや各種デバイスの設定を行うページですが、はじめに用意されてい るのはノード設定ページのみなので、それについて説明します。ノード設定ページを表示するには、 上部メニューから、「セットアップ」→「ノード設定」をクリックします。

	UECS-Pi					
トップ CCM-	一覧 状態ログ	セットアップ	ログアウト			
ノード名	UECS-Pi					
ノード種別名(kNN)	cMC					
room-region-order (priority)	1 - 1 - 1 (1)					
現在時刻	変更 2014-10-24 09:34:27 時間管理CCM(Date,Time)で補正を行う					
管理バスワード						
アアドレス	● DHCP取得 ○ 固定					
ボードモデル	Model B 🔻					
		保存				
	Powerd by UEC	S.				

図 50:ノード設定ページ

ノード設定ページ使用時は、以下表を参考に接続動作させたい UECS/LAN ネットワークに合わ せた設定値に変更し「保存」ボタンをクリックして設定を保存してください。

No.	項目	説明				
1	ノード名	UECS 通信のノードスキャン応答 CCM の <name>項目に使用されます。設</name>				
		定名は画面上部のタイトルにも使用されます。設定文字は ASCII コード(ISO				
		646-1991)[半角の英数字記号]のみ使用が許可されています。				
2	ノード種別(kNN)	UECS 通信で使用される CCM 識別子のノード種別に使用されます。ノード				
		種別名の一覧は UECS 実用通信規約の仕様書に記載されています。				
3	room-region-order	UECS 通信で使用される、各種区分番号です。LAN に接続される他の UECS				
	(priority)	ノードの設定値と重複しないように設定してください。				
		・room : 部屋番号 [0~127]				
		・region :系統番号 [0~127]				
		・order : 通し番号 [0~30000]				
		・priority:優先順位 [0~30]				
4	現在時刻	「変更」をチェックして時刻を修正入力すると、Raspberry Pi 内部の時刻が				
		変更されます。				
		「時間管理 CCM(Date, Time)で補正を行う」をチェックすると、UECS 通信				

		規約で定義されている時間管理サーバが発信する CCM(Date, Time)を受信し		
		て、内部時刻を自動的に補正します。		
		(注:インターネット通信可能な LAN に接続されている場合は、自動的に NTP サーバ		
		から現在値が取得されます。NTP サーバに接続できず、Date/Time CCM も受信できな		
		い場合は、電源が OFF になると、現在時刻もリセットされます。)		
5	管理パスワード	設定画面にログインするためのパスワードを変更したい場合に、新しいパスワ		
		ードを入力します。		
6	IPアドレス	Raspberry Pi 本体の IP アドレスを設定します。		
		・「DHCP」 LAN 内の DHCP サーバから自動的に IP アドレス情報を取得し		
		ます。		
		・「固定 IP」設定値項目が表示されますので、接続する LAN 環境に合わせた		
		設定値を入力してください。		
		(注:UECS-Pi SDK のデフォルトでは「DHCP」になっていますが、Eclipse 上で開発		
		している状態では IP アドレス変更は機能しません。Raspberry Pi 実機にインストール		
		した際に、必要に応じて設定変更してください。)		

表 11:ノード設定画面項目

4. チュートリアル

UECS-Pi SDK では様々なセンサを使った環境測定や、アクチュエータを使った機器制御が行えます。ここ では先の章で Eclipse にインポートした「UecsPi_SDK」プロジェクトファイル内に含まれているサンプルコ ードの解説をします。センサとアクチュエータを使う簡単なチュートリアルを紹介し、その基礎を理解します。 作成したソフトウェアのインストール方法は「<u>UECS-Pi アプリケーションのインストール</u>」の章を参照して 下さい。

4.1. <u>プロジェクト構成</u>

Eclipse にインポートした「UecsPi_SDK」プロジェクトに含まれるフォルダやファイルについて概説します。



図 51: プロジェクト内フォルダ構成

No.	フォルダ1ファイル名	説明
1	/src/以下フォルダ	アプリケーションのソースコードがパッケージ構成ツリー
		の形でフォルダ分けされて格納されます。
		「myapp」はサンプル用の仮パッケージ名ですので、任意
		のパッケージ名に変更可能です。
2	/src/logback.xml	プログラム内部出力ログの設定ファイルです。
		内部デバッグ出力の切り替えなど、アプリケーション開発時
		と本番稼働時で設定を変更することで出力内容を切り替え
		ることが可能です。設定ファイルの書式は、オープンソース
		ログ出力ライブラリの Logback の書式に従います。
		(http://logback.qos.ch/)
3	/WebContent/css/styles.css	WebUI 画面の配色や文字サイズなどを定義する CSS ファ
		イルです。開発者は独自の CSS に差し替えることで画面デ
		ザインを変更可能です。
4	/WebContent/WEB-INF/lib	アプリケーションが動作時に参照するライブラリ jar ファ
		イルが格納されています。開発者が独自の機能を開発するた
		めに必要なライブラリ jar を追加することが可能です。
5	/WebContent/WEB-INF/web.xml	Tomcat が読み込む Web アプリケーション設定ファイルで
		す。Wicket 関連の設定が記述されています。

表 12:フォルダ、ファイル説明

4.2. <u>ノード、WebUI アプリケーションクラス</u>

まず始めに、UECS-Pi SDK で独自のアプリケーションを作成する際に基本となる UECS ノードクラス と WebUI アプリケーションクラスの解説をします。

4.2.1. <u>ノードクラス</u>

ノード部分の基本的な構成は「ノードクラス+ノード設定クラス」が1セットとなります。



それでは、ノードクラスとノード設定クラスのコードを見ていきます。MyPiNode クラスは、UECS-Pi 基本ライブラリに用意されている親クラス(UecsPiNode)を継承することで UECS ノードとしての基本機 能を持たせています。追加のコードとして、installProcess()メソッドをオーバーライドすることで、初回 起動時にデバイスやコンポーネントを内部DBに初期登録させています。本サンプルでは、チュートリア ルで解説する4つのデバイスクラスを初期登録しています。動作させるデバイスが固定している場合は、 初期登録しておくことが簡単ですが、初期起動時にはデバイスを登録せず、WebUI 画面でユーザが設定 変更する操作と連動して、動的にデバイスやコンポーネントを登録、削除させることも可能です。

MyPiNode.java

public class My	PiNode extends UecsPiNode {
public MyPil super(n }	Node() { ew MyPiNodeConfig());
protected v	oid installProcess() throws Exception {
Databaso pub	eManager. callInTransaction(new Callable <void>() { lic Void call() throws Exception {</void>
	<pre>// Sample 1 DummyDevice dummyDevice = new DummyDevice(DatabaseUtils.nextDeviceId()); DummySensor dummySensor = new DummySensor(DatabaseUtils.nextComponentId()); dummyDevice.addComponent(dummySensor); DatabaseUtils.saveDevice(dummyDevice);</pre>
TimerSwitchActua	<pre>// Sample 2 SimpleGpioDevice gpioDevice = new SimpleGpioDevice(DatabaseUtils.nextDeviceId()); TimerSwitchActuator timerSwitch = new ator(DatabaseUtils.nextComponentId()); gpioDevice.addComponent(timerSwitch); DatabaseUtils.saveDevice(gpioDevice);</pre>
	<pre>// Sample 3 SimpleSerialDevice serialDevice = new SimpleSerialDevice(DatabaseUtils.nextDeviceId()); RegexSensor regexSensor = new RegexSensor(DatabaseUtils.nextComponentId()); serialDevice.addComponent(regexSensor); DatabaseUtils.saveDevice(serialDevice);</pre>
	<pre>// Sample 4 SimpleI2cDevice i2cDevice = new SimpleI2cDevice(DatabaseUtils.nextDeviceId()); I2cCo2Sensor co2Sensor = new I2cCo2Sensor(DatabaseUtils.nextComponentId()); i2cDevice.addComponent(co2Sensor); DatabaseUtils.saveDevice(i2cDevice);</pre>
}	return null;
});	
}	
}	

MyPiNodeConfig.java も基本ライブラリに用意されている親クラス(UecsPiNodeConfig)を継承するこ とで、デフォルト設定値がプリセットされています。独自の設定値に変更したい場合に、コンストラクタ 内で任意の初期設定値に書き換えることができます。本サンプルでは、ノード名称を独自のものに変更し ていますが、そのほかにも UECS ID やベンダー名、ノード状態通知 CCM の room-region-order なども 初期値を変更できます。変更可能な設定値キーは、継承元クラスの定数で定義されていますので、JavaDoc を参照してください。

MyPiNodeConfig.java

```
public class MyPiNodeConfig extends UecsPiNodeConfig {
    public MyPiNodeConfig() {
        // ノード名称
        setString(KEY_NODE_NAME, "MyAppNode");
    }
}
```

4.2.1. WebUI アプリケーションクラス

WebUI アプリケーション部分では、Wicket フレームワークを利用した基底クラスを拡張し、画面パーツ クラス (xxxPanel.java, xxx)、HTML テンプレートファイル (xxxPanel.html)、プロパティファイル (xxx.properties) を作成します。



図 53: WebUI アプリケーション部分のファイル構成

本サンプルで作成している MyUIApplication クラスでは、先に作成した MyPiNode クラスをアプリケー ション内で呼び出し、カスタマイズされたメニュー(MyMenuPanel)とフッター(MyFooterPanel)に変更 するために、継承元クラスのメソッドをオーバーライドしています。

MyUIApplication.java

public class MyUIApplication extends WebUIApplication {

```
@Override
protected UecsPiNode createNodeInstance() {
    return new MyPiNode();
}
@Override
protected Class<? extends Panel> getMenuPanelClass() {
    return MyMenuPanel.class;
}
@Override
protected Class<? extends Panel> getFooterPanelClass() {
    return MyFooterPanel.class;
}
```

各クラスの実装は Wicklet の API に従っていますので、詳細は Wicket の HP(http://wicket.apache.org/) のユーザーガイドを参照してください。

4.3. <u>Sample 1 : ダミーセンサ</u>

このチュートリアルでは、ダミー値を送信する仮想センサのサンプルを解説します。実際のセンサ機器は 使いません。WebUIから設定値をセンサ値としてノードに認識させ、定義変更可能な任意のデータ CCM として送信を行います。

MyAppNode					
トップ	CCM一覧	状態ログ	セットアップ	ログアウト	
ーサンプル1設定[ダミー	-センサ]				
CCM表示名	Dummy Sen	sor			
CCM設定	項目名: InAir 送信レベル: A [room: 1	Temp ノー -10S-0 ♥単位: C] -[region: 1] -[ord	"種別: 精度: 1 er: 1] (priority: 1)	
ダミーセンサー	値 0				
					保存
		Powerd by UECS	®. (http://www.uecs.jp	<i>/</i>)	

図 54: ダミーセンササンプル WebUI (枠内がダミーセンサ入力値)

Γ		MyAppNode								
		ŀ	ップ	CCM一覧	状態ログ	セットア	ップ	ログアウト		
	Γ	CCM-	-覧		coure=	S/D	相大体	再和時初	#FRB.Fn	
		110.	-017		CCMLER	5/K	现往他	史和时刻	AND C 40	
		2	Dummy Sensor	InAir	Temp (1-1-1) [A-10S-0]	S	0.0 C	2014-11-03 14:39:45		
Ħ		3	Timer Switch [opr]	Swite	hopr.1 (1-1-1) [A-1M-1]	S	0	2014-11-03 14:39:45		
		4	Timer Switch [rcA]	Swite	hrcA.1 (1-1-1) [S-1M-0]	R				
		5	Timer Switch [rcM]	Swite	hrcM.1 (1-1-1) [S-1S-0]	R				1
		6	Serial Port Sensor	InAir	Humid (1-1-1) [A-10S-0]	S	%	2014-11-03 14:39:45		1
		7	I2C Bus Sensor	InAir	CO2 (1-1-1) [A-10S-0]	S	ppm	2014-11-03 14:39:45		

図 55: CCM 一覧画面(枠内がダミーセンサ入力値を反映した CCM)

4.3.1. デバイス、コンポーネントクラス

デバイス部分の基本的な構成は「デバイスクラス+デバイス設定クラス」が1セットと、「コンポーネ ントクラス+コンポーネント設定クラス」が任意セット数含まれる形です。本サンプルでは、コンポーネ ントはダミーセンサー1個となりますので、1セット用意されています。



それでは、デバイスクラスとデバイス設定クラスのコードを見ていきます。DammyDevice.java、 DammyDeviceConfig.javaは継承元のUECS-Pi基本ライブラリクラスのPiDeviceBase、PiDeviceConfig と同じ動作をするコードになっていますので、ここでは説明は割愛します。コンポーネントクラスとコン ポーネント設定クラスを見ていきます。

DammySensor.java

```
public class DummySensor extends PiSensorBase<DummySensorConfig> {
    public DummySensor(String id) {
        super(id, new DummySensorConfig());
    }
    @Override
    protected void onStart() throws Exception {
        // 起動時の処理を記述します
        Number val = getConfig().getFloat(DummySensorConfig.KEY_SENSE_VALUE, 0);
        super.setValue(val);
    }
}
```

```
@Override
protected void onStop() throws Exception {
  // 停止時の処理を記述します
}
```

DammySensor.java も、基本的には継承元のクラス PiSensorBase と同じ動作をするコードですが、起動 時に DammySensorConfig から KEY_SENSE_VALUE を取得して、setValue0に入力しています。この KEY_SENSE_VALUE が、WebUI から入力される、ダミーのセンサ取得値です。また setValue0は値が 入力されると、センサーコンポーネントに対応する CCM の値も自動的に更新されます。

DammySensorConfig.java

```
public class DummySensorConfig extends PiSensorConfig {
    /** 設定値項目キー:ダミーセンサー値 */
    public final static String KEY_SENSE_VALUE = "SenseValue";
    public DummySensorConfig() {
        // センサー値に対応するCCM情報を初期設定します。
        setString(KEY_COMPONENT_NAME, "Dummy Sensor");
        setString(KEY_COM_INFO_NAME, "InAirTemp");
        setString(KEY_COM_LEVEL, ComLevel. A_10S_0.toString());
        setInt(KEY_COM_REGION, 1);
        setInt(KEY_COM_PRIORITY, 1);
        setInt(KEY_COM_PRIORITY, 1);
        setString(KEY_COM_SIDE, "S");
        setString(KEY_COM_SIDE, "S");
        setInt(KEY_COM_CAST, 1);
        setInt(KEY_COM_CAST, 1);
        setInt(KEY_SENSE_VALUE, 0);
    }
}
```

DammySensorConfig.java では、コンポーネント設定値の定数定義と対応する CCM の初期設定を行っています。これらの初期値は、ダミーセンサ用の WebUI の初期値として使用されます。コンポーネントに設定できる値のキー定数は、継承元クラスに定義されていますので、JavaDoc を参照してください。

4.3.1. WebUI 設定画面クラス

次は、デバイス設定画面の WebUI コード部分を見ていきます。WebUI 部分の基本的な構成は「Java ファイル+HTML ファイル+プロパティファイル」です。

a 🔠 myapp.webui.sample1

- DummyDevicePage.java
 - DummyDevicePage_en.properties
 - DummyDevicePage_ja.properties
 - DummyDevicePage.html

図 57: WebUI 部分のファイル構成

DammyDevicePage.java

```
@AuthorizeInstantiation({Roles. ADMIN}) ----
                                           ---(1)
public class DummyDevicePage extends LayoutPage {
    private static final long serialVersionUID = 9205561784306869639L;
    private Form<PropertyConfig> deviceForm;
    private Log logger = LogFactory.getLog(getClass());
    private static String[] comLeve/s = new String[] {
            CcmLevel. A_1S_0. toString(),
            CcmLevel. A 1S 1. toString().
            CcmLevel. A_10S_0. toString(),
CcmLevel. A_10S_1. toString(),
            CcmLevel. A_1M_0. toString(),
            CcmLevel. A 1M 1. toString(),
            CcmLevel. B_0. toString(),
            CcmLevel. <u>B</u>1. toString(),
    };
    private String deviceId;
    private String componentId;
    public DummyDevicePage() {
    · 〈中略〉
    @Override
    protected void onInitialize() {
        super.onInitialize();
        // 設定対象となるデバイスとコンポーネントのIDを取得します。
        MyPiNode node = (MyPiNode)MyUIApplication. getNodeInstance();
        DummyDevice device = node. listDevices(DummyDevice.class).get(0);
DummySensor sensor = device.listComponents(DummySensor.class).get(0); -
                                                                               ---(4)
        deviceId = device.getId();
        componentId = sensor.getId();
        // コンポーネント設定値をWebUIフォームの初期値としてコピーします
```

```
formParams.putAll(sensor.getConfig());
        deviceForm = new Form<PropertyConfig>("deviceForm") {
            private static final long serialVersionUID = 1L;
            @Override
            public void onSubmit() { -----6
                try {
                    deviceForm.modelChanging();
                    // フォーム入力された設定値をセンサー設定に反映して
                    // DBに保存したのち、ノードを再起動します。
DatabaseManager.callInTransaction(new Callable<Void>() {
                        public Void call() throws Exception {
                            DummySensor sensor = (DummySensor)MyUIApplication
                                    . getNodeInstance().getDevice(deviceId)
                                    .getComponent(componentId);
                            sensor.getConfig().putAll(formParams);
                            DatabaseUtils. saveComponent(deviceId, sensor);
                            MyUIApplication.getNodeInstance().restart();
                            return null;
                        }
                    });
                    deviceForm.modelChanged();
                    success (MessageUtils. getMessage (MessageCode. SAVED));
                } catch (Exception e) {
                    error(MessageUtils.getMessage(MessageCode.ERROR));
                    logger.error("save error.", e);
                }
            }
        };
        // フォームに入力フィールドを登録します。
                                                          (7)
        deviceForm.setDefaultModel(new CompoundPropertyModel<PropertyConfig>(formParams));
        deviceForm
                .add(new RequiredTextField<String>(DummySensorConfig.KEY_COMPONENT_NAME, String.class)
                        . add (new StringValidator (1, 50));
        deviceForm. add (new RequiredTextField<String>(DummySensorConfig. KEY CCM INFO NAME,
String. class)
        · 〈中略〉
        add(deviceForm);
    }
```

DammyDevicePage.java では HTML ファイルに埋め込み表示するコンテンツや、HTML が操作され た時の処理を記述しています。コンテンツを作成しているコードでは、主に Wicket フレームワークが用 意している Form クラスに対して、入力フィールドのアイテムや、入力値バリデーションクラスの登録を 記述しています。

No.	コードの説明
1	Webページ表示の際に、認証チェックを行うためのアノテーション
2	LayoutPageクラスを継承する事で、UECS-Pi SDKのWebUIデザインを継承
3	Web画面の入力フォームフィールドの入力値を格納するためのデータオブジェクト

4	ノードに初期登録されているデバイスインスタンスを取得
5	デバイスに初期登録されているセンサーコンポーネントインスタンスを取得
6	Web画面の登録ボタンクリックのリクエスト処理を記述
7	WicketのFormオブジェクトに入力フィールドデータを関連付け、各項目に対応するフィールドオブジェク
	トや、入力バリデーションオブジェクトを生成して登録

表 1	3:	Dammy	Device	Page.i	ava 0	り処理
-----	----	-------	--------	--------	-------	-----

次は、HTML ファイルを見ていきます。「3.4.1.基本ページレイアウトとレイアウトの継承」で説明したように、WebUI の HTML ファイルは、Java ファイルに記述されている内容を表示するテンプレートになっています。

DammyDevicePage.html



DammyDevicePage.html では、LayoutPage を継承していますが、コンテンツは独自のものを表示します。 独自コンテンツが表示されるのは <wicket:extend> ~ </wicket:extend> タグ内です。 DammyDevicePage.javaのdeviceFormの内容は<form wicket:id="deviceForm">~</form">に表示されています。 また DammyDevicePage.java のデータ入力オブジェクトと HTML ファイル内のフォーム入力フィー ルド値を「wicket:id」属性値で互いに関連付けています。

No.	コードの説明
1	独自コンテンツ表示用タグ
2	wicket:messageタグ。プロパティファイルからテキストを取得している
3	DammyDevicePage.javaのdeviceFormの内容を反映するタグ
4	wicket:idでJavaクラス内のformParamsオブジェクトの値と関連付けている
5	deviceFormAreaにaddされた内容を呼び出している
6	「保存」ボタン。クリックするとJavaクラス内のFormオブジェクトのsubmitメソッド処理が行われる

表 14: DammyDevicePage.html のタグの意味

4.4. <u>Sample 2: タイマー動作アクチュエータ</u>

このチュートリアルでは、Raspberry Pi の GPIO 端子にリレーユニットを接続し、一定間隔で周期的に ON/OFF 動作をするサンプルを解説します。実際に機器同士を接続して動作を確認して下さい。



図 58:タイマー動作アクチュエータサンプル作成イメージ

WebUI 画面から、各種動作定義を行うことができます。

MyAppNode						
トップ C	CM一覧 状態ログ	セットアップ	ログアウト			
サンブル2設定[タイマー制御]						
GPIO 接続 とン	GPIO-0					
装置名称	Timer Switch					
CCM識別子	項目名: Switch 系統番号: 1	ノード種別:				
オディン	□ HL反転動作					
周期時間	秒ごとに	少間ONにする				
停止時間帯	. ~ .					
			_			
			保	存		
	Powerd by UEC	S ®. (http://www.uecs.jp/)			

図 59: タイマーアクチュエータサンプル WebUI 設定項目

No.	項目	説明
1	装置名称	接続した機器のラベル名称です。トップ画面などでアクチュエータ情報として表
		示される名称となります。
2	CCM識別子	UECS 通信で使用される CCM 識別子の情報です。設定可能な識別子のルールは、UECS
		実用通信規約、あるいは運用ガイドライン書類に記載されていますので、参考に設定してく
		ださい。
		[設定例]項目名=Relay, 系統番号=1, ノード識別子=cMC で設定された場合は、
		以下の3種類の CCM がノード内部に登録されます。
		・制御機器運転状態 CCM (送信)=Relayopr.1.cMC
		・遠隔制御指示 CCM (受信)=RelayrcA.1.cMC
		・遠隔操作指示 CCM (受信)=RelayrcM.1.cMC
3	オプション	[H/L 反転動作]
		デフォルト動作では GPIO ピン状態が、LOW=OFF、HIGH=ON として扱われます。
		接続しているリレーモジュールによっては動作が逆の場合がありますので、その場合はチェ
		ックを入れてください。
4	周期時間	周期条件を入力します。
		例:"[30]秒ごとに[10]秒間 ON にする"と入力された場合、
		[ON:10 秒]-[OFF:20 秒]-[ON-10 秒]-[OFF:20 秒]~(以降繰り返し)~
5	停止時間帯	一日のうち、定期動作を停止したい時間帯がある場合に指定できます。
		0 時をまたいだ設定も可能です。(例:23:00~2:00)
	+	

表 15: タイマーアクチュエータサンプル WebUI 設定項目解説

4.4.1. デバイス、コンポーネントクラス

デバイス部分のコードを解説します。

myapp.device.sample2
J SimpleGpioDevice.java
SimpleGpioDeviceConfig.java
I TimerSwitchActuator.java
I TimerSwitchActuatorConfig.java
図 60:デバイス部分のファイル構成

SimpleGpioDevice.java

```
public class SimpleGpioDevice extends GpioDeviceBase<SimpleGpioDeviceConfig> {
   public SimpleGpioDevice(String id) {
       super(id, new SimpleGpioDeviceConfig());
   }
    @Override
   protected void onInit() throws Exception {
       super.onInit();
       UecsPiNodeConfig nconf = (UecsPiNodeConfig) getNode().getConfig();
       for (IComponent<?> compo : listComponents()) {
            // アクチュエータCCMの属性値として、ノードのCCM属性値を引き継いで設定
           ComponentConfig conf = compo.getConfig();
           conf.setInt(ComponentConfig.KEY_CCM_ROOM, nconf.getInt(UecsPiNodeConfig.KEY_NODE_ROOM,
0));
            conf.setInt(ComponentConfig.KEY_CCM_REGION,
nconf.getInt(UecsPiNodeConfig.KEY_NODE_REGION, 0));
           conf.setInt(ComponentConfig.KEY_CCM_ORDER, nconf.getInt(UecsPiNodeConfig.KEY_NODE_ORDER,
0));
            conf.setInt(ComponentConfig.KEY_CCM_PRIORITY,
nconf.getInt(UecsPiNodeConfig.KEY_NODE_PRIORITY, 0));
       }
    }
```

SimpkeGpioDeviceConfig.java

```
public class SimpleGpioDeviceConfig extends GpioDeviceConfig {
    private static final long serialVersionUID = -2238752941334340868L;
    public SimpleGpioDeviceConfig() {
    }
}
```

デバイスクラスでは内包するコンポーネント(ここでは TimerSwitchActuator)の CCM の属性値(room / region / order / priority)をノードの設定値と同期させるために引継ぎコピーを行っています。デバイス 設定クラスは継承元のクラスと同じ動作を行っています。 次はアクチュエータクラス、アクチュエータ設定クラスを見ていきます。

```
TimerSwitchActuator.java
```

```
public class TimerSwitchActuator extends SwitchActuator<TimerSwitchActuatorConfig> {
   public TimerSwitchActuator(String id) {
        super(id, new TimerSwitchActuatorConfig());
   }
    @Override
   protected void onInit() throws Exception {
        super.onInit();
        TimerSwitchActuatorConfig config = getConfig();
        int periodicInterval = config.getInt(TimerSwitchActuatorConfig.KEY_PERIODIC_INTERVAL. 0);
        int onInterval = config.getInt(TimerSwitchActuatorConfig.KEY_PERIODIC_ON, 0);
        if (periodicInterval > 0 && onInterval > 0) {
            // 設定値が有効な場合は、定周期動作
           PeriodicSwitchAction periodAction = new PeriodicSwitchAction();
           periodAction.setPeriodicInterval (periodicInterval * 1000L);
           periodAction.setOnInterval(onInterval * 1000L);
 eriodAction.setInactiveStartTime(config.getInt(TimerSwitchActuatorConfig.KEY PERIODIC INACTIVE START H
OUR, −1),
                   config.getInt(TimerSwitchActuatorConfig.KEY_PERIODIC_INACTIVE_START_MIN, -1));
periodAction.setInactiveEndTime(config.getInt(TimerSwitchActuatorConfig.KEY_PERIODIC_INACTIVE_END_HOUR,
-1).
                   config.getInt(TimerSwitchActuatorConfig.KEY_PERIODIC_INACTIVE_END_MIN, -1));
            setAction(ActionMode. Autonomy, periodAction);
           setActionMode(ActionMode. Autonomy);
        } else {
            // 何も設定されない場合は、スタンドアローンモードで起動
           setActionMode(ActionMode.Standalone);
   }
```

アクチュエータクラスでは初期化を行う onInit()メソッドをオーバーライドして、周期動作の設定を行っています。周期 ON/OFF 動作を行うアクションクラスは、UECS 基本ライブラリで提供されています (PeriodicSwitchAction)ので、そのクラスを利用するだけで、アクチュエータの動作を定義する事が可能 です。ここでは、E10 規約で定義されている、アクチュエータの動作モードのうち、自律モード (ActionMode.Autonomy)に対応したアクションとして登録しています。定期動作以外にも、センサーCCM 連動や、時刻範囲指定で動作を行うようなアクションクラスも用意されていますので、詳細は JavaDoc を参照してください。

TimerSwitchActuatorConfig.java

public class TimerSwitchActuatorConfig extends SwitchActuatorConfig {

private static final long serialVersionUID = 1L;

/** 設定項目キー : 定周期動作間隔 */ public static final String KEY_PERIODIC_INTERVAL = "PeriodicInterval";

/** 設定項目キー : ON動作間隔 */ public static final String KEY_PERIODIC_ON = "PeriodicOn"; /** 設定項目キー : 動作停止開始時刻(時) * public static final String KEY_PERIODIC_INACTIVE_START_HOUR = "PeriodicInactiveStartHour"; /** 設定項目キー : 動作停止開始時刻(分) public static final String KEY_PERIODIC_INACTIVE_START_MIN = "PeriodicInactiveStartMinute"; /** 設定項目キー : 動作停止終了時刻(時) */ public static final String KEY_PERIODIC_INACTIVE_END_HOUR = "PeriodicInactiveEndHour"; /** 設定項目キー : 動作停止終了時刻(分) public static final String KEY_PERIODIC_INACTIVE_END_MIN = "PeriodicInactiveEndMinute"; public TimerSwitchActuatorConfig() { setInt (KEY_DEFAULT_VALUE, 0); setInt (KEY_FIXED_VALUE, 0); setBoolean (KEY_INVERSE, false); setBoolean (KEY IS REPRESENTATIVE, false); setInt(KEY_CCM_LINE, 1); setInt(KEY_GPI0_PIN, 0); setString(KEY_COMPONENT_NAME, "Timer Switch"); setString(KEY_CCM_INFO_NAME, "Switch"); }

TimerSwitchActuatorConfig.java では設定値キー定義と初期値設定を行っています。この定数は、 WebUI 設定画面の入力フィールドのキーとしても使用されます。

4.4.2. WebUI 設定画面クラス

WebUI 部分のファイルの構成は Sample 1 と全く同じ構造です。実装パターンも同様ですので、ここでは説明を割愛します。

🛛 赶 myapp.webui.sample2
D GpioActuatorPage.java
GpioActuatorPage_en.properties
GpioActuatorPage_ja.properties
📔 GpioActuatorPage.html

図 61: WebUI 部分のファイル構成

4.5. <u>Sample 3:シリアル通信</u>

個のチュートリアルでは、シリアル通信を使って外部接続されたデバイスから定期的に送信されるデータ 文字列を読み取ってセンサ値として CCM 送信するサンプルを解説します。データ文字列は、正規表現パタ ーンマッチングで数値部分を抽出します。



図 62:実際のシリアル通信デバイスとの接続イメージ

	MyAppNode								
	トップ	CCM一覧	状態ログ	セットアップ	ログアウト				
Г	-サンブル3設定 [シリア	ル通信]							
	シリアルポート								
	通信速度	9600	[bps]						
	値抽出パターン	\[HUMID:([0-	9.\-\+]+)\]						
						保存			
			Powerd by UECS	®. (http://www.uecs.jp/)				

No.	項目	説明			
1	シリアルポート	- シリアルポートの識別子を入力します。通常は以下で動作します。			
		GPIO ポートの UART(RX/TX)接続 :「/dev/ttyS0」			
		USB シリアル変換接続:「/dev/ttyUSB0」			
2	通信速度	シリアルポートの通信速度(bps)を入力します。接続する機器側の通信速度の仕様に			
		合わせてください。			
3	値抽出パターン	値を抽出する正規表現パターンを設定します。(書式は Java 正規表現ライブラリに			
		従います)			

表 16:シリアル通信サンプル WebUI 設定項目解説

4.5.1. デバイス、コンポーネントクラス

デバイス部分のコードを解説します。



図 63: デバイス部分のファイル構成

SimpleSerialDevice.java

```
public class SimpleSerialDevice extends SerialPortDeviceBase<SimpleSerialDeviceConfig> {
   public SimpleSerialDevice(String deviceId) {
        super(deviceId, new SimpleSerialDeviceConfig());
   }
    @Override
   protected void onInit() throws Exception {
        super.onInit();
       UecsPiNodeConfig nconf = (UecsPiNodeConfig) getNode().getConfig();
        for (IComponent<?> compo : listComponents()) {
              CCMの属性値として、ノードのCCM属性値を引き継いで設定
            ComponentConfig conf = compo.getConfig();
            conf.setInt(ComponentConfig.KEY_CCM_ROOM, nconf.getInt(UecsPiNodeConfig.KEY_NODE_ROOM,
0));
            conf.setInt(ComponentConfig.KEY_CCM_REGION,
nconf.getInt(UecsPiNodeConfig.KEY_NODE_REGION, 0));
            conf.setInt(ComponentConfig.KEY CCM ORDER, nconf.getInt(UecsPiNodeConfig.KEY NODE ORDER,
0));
            conf.setInt(ComponentConfig.KEY CCM PRIORITY.
nconf.getInt(UecsPiNodeConfig.KEY_NODE_PRIORITY__0));
       }
   }
    @Override
   public void dataReceived(byte[] data) {
        for (RegexSensor sensor : listComponents(RegexSensor.class)) {
            sensor.parseData(data);
    }
```

SimpleSerialDeviceConfig.java

```
public class SimpleSerialDeviceConfig extends SerialPortDeviceConfig {
    private static final long serialVersionUID = -5081030325821182630L;
    public SimpleSerialDeviceConfig() {
        setString(KEY_DEVICE_NAME, SimpleSerialDevice.class.getSimpleName());
        setString(KEY_SERIAL_PORT_ID, "");
        setInt(KEY_SERIAL_PORT_SPEED, 9600);
    }
}
```

l

UECS-Pi 基本ライブラリでは、シリアル通信を行う基底デバイスクラスが用意されています (SerialPortDeviceBase)ので、継承することでシリアル通信機能を持たせることができます。 SimpleSerialDeviceConfig で設定された通信ポート ID(*KEY_SERIAL_PORT_ID*) と通信スピード[BPS] (*KEY_SERIAL_PORT_SPEED*)で通信を行います。データが受信されると、非同期に dataReceived()メソッドが 起動されますので、dataReceived()メソッドをオーバーライドして受信データを処理するコード記述しま す。本サンプルでは実デバイスに対してデータ送信は行っていませんが、継承元クラスには、データ送信 を行う sendData()メソッドも用意されています。詳細は JavaDoc を参照してください。

```
RegexSensor.java
```

```
public class RegexSensor extends PiSensorBase<RegexSensorConfig> {
   private Pattern regexPattern;
   public RegexSensor(String id) {
       super(id, new RegexSensorConfig());
   }
   @Override
   protected void onInit() throws Exception {
       super.onInit();
       RegexSensorConfig conf = getConfig();
       String textPattern = conf.getString(RegexSensorConfig.KEY_REGEX_PATTERN);
        if (textPattern != null) {
            try {
                regexPattern = Pattern. compi/e(textPattern);
            } catch (PatternSyntaxException e) {
               notifyException(e);
       }
   }
   @Override
   protected void onStart() throws Exception {
    @Override
   protected void onStop() throws Exception {
   public void parseData(bvte[] data) {
       String text = null;
       try {
            text = new String(data, "us-ascii");
        } catch (UnsupportedEncodingException e) {
            return
       }
        if (regexPattern != null && text != null) {
            Matcher matcher = regexPattern.matcher(text);
            if (matcher.find()) {
                String sval = matcher.group(1);
                trv
                    double value = Double. valueOf(sval);
                    setValue(value);
                } catch (NumberFormatException e) {
                    notifyException(e);
                }
```

	}		
}			
}			

RegexSensorConfig.java

```
public class RegexSensorConfig extends PiSensorConfig {
    private static final long seria/VersionUID = -2298819350796006077L;
    public static final String KEY_REGEX_PATTERN = "RegexPattern";
    public RegexSensorConfig() {
        setString(KEY_COMPONENT_NAME, "Serial Port Sensor");
        setString(KEY_CCM_INFO_NAME, "InAirHumid");
        setString(KEY_CCM_LEVEL, ComLevel. A_10S_0.toString());
        setString(KEY_CCM_UNIT, "%");
        setString(KEY_REGEX_PATTERN, "¥¥[HUMID:([0-9.¥¥-¥¥+]+)¥¥]");
    }
}
```

RegexSensor.java は WebUI 画面から設定された正規表現パターンを元に、シリアル通信で得られたバ イト配列データを文字列に変換して値抽出を行っています。本サンプルでは、SerialDevice に対して1つ の RegexSensor インスタンスを持たせていますが、複数の RegexSensor を持たせるように拡張すれば、 異なる文字パターンを解析して複数のセンサー値を取得する機能も簡単に実現することができます。送受 信データは byte 配列で処理されますので、バイナリ値での通信にも対応可能です。

4.5.2. <u>WebUI 設定画面クラス</u>



図 64:WebUI 部分のファイル構成

SerialDevicvePage.java は、Sample1、Sample2 と同じ構造です。また対になる HTML ファイル等も 同様ですので、説明は割愛します。

4.6. <u>Sample 4 : I2C 通信</u>

最後に、I2C 通信方式で仮想の CO2 センサデバイスと通信するサンプルを作成します。実在の製品デバ イスとの通信ではありませんが、本サンプルコードを参考に、データ取得コマンドや受信データチェック処 理などをコーディングする事で、実在の I2C 通信デバイスと通信が可能になります。



図 65: I2C 通信デバイス接続イメージ

MyAppNode					
トップ	CCM一覧	状態ログ	セットアップ	ログアウト	
□ □ ^{サンブル4} 設定[I2C通信]]				
I2Cアドレス(HEX)) 0x 64				
計測間隔	10	秒			
保存					
Powerd by UECS [®] . (http://www.uecs.jp/)					

No.	項目	説明
1	I2Cアドレス	I2C バスに接続されたデバイスのアドレス(HEX 形式)を設定します
2	計測間隔	定期的にセンサー値取得コマンドを発行する時間間隔を設定します

<u>表 17: I2C</u> 通信サンプル WebUI 設定項目解説

4.6.1. デバイス、コンポーネントクラス

デバイス部分のコードを解説します。



SimpleI2cDevice.java

```
public class SimpleI2cDevice extends I2cDeviceBase<SimpleI2cDeviceConfig> {
   public SimpleI2cDevice(String deviceId) {
       super(deviceId, new SimpleI2cDeviceConfig());
   }
   @Override
   protected void onInit() throws Exception {
       super.onInit();
       UecsPiNodeConfig nconf = (UecsPiNodeConfig) getNode().getConfig();
       ComponentConfig conf = compo.getConfig();
           conf.setInt(ComponentConfig.KEY_CCM_ROOM, nconf.getInt(UecsPiNodeConfig.KEY_NODE_ROOM,
0));
           conf.setInt(ComponentConfig.KEY_CCM_REGION,
nconf.getInt(UecsPiNodeConfig.KEY_NODE_REGION, 0));
           conf.setInt(ComponentConfig, KEY CCM ORDER, nconf.getInt(UecsPiNodeConfig, KEY NODE ORDER,
0));
           conf.setInt(ComponentConfig.KEY CCM PRIORITY.
nconf.getInt(UecsPiNodeConfig.KEY_NODE_PRIORITY__0));
       }
   }
```

SimpleI2cDeviceConfig.java

```
public class SimpleI2cDeviceConfig extends I2cDeviceConfig {
    public SimpleI2cDeviceConfig() {
        setString(KEY_DEVICE_NAME, SimpleI2cDevice.class.getSimpleName());
        setInt(KEY_I2C_ADDRESS, 0x40);
    }
}
```

UECS-Pi 基本ライブラリでは、I2C 通信を行う基底デバイスクラスが用意されています(I2cDeviceBase) ので、継承することで I2C 通信機能を持たせることができます。SimpleI2cDeviceConfig で設定された I2C アドレス (*KEY_12C_ADDRESS*) に対して通信を行います。 本サンプルでは1つだけの I2C アドレスに対して通信を行っていますが、ノードに対して複数のデバ イスを持たせることで、複数のI2C アドレスデバイスと通信可能です。

```
I2cCo2Sensor.java
```

```
public class I2cCo2Sensor extends I2cSensorBase<I2cCo2SensorConfig> {
    private static final byte GET_CO2_DATA = (byte) 0x05;
    private byte[] buffer = new byte[2];
    public I2cCo2Sensor(String id) {
        super(id, new I2cCo2SensorConfig());
    }
    @Override
    public void onReadProcess(I2CDevice i2cDevice) throws Exception {
        I2cUtils.write(i2cDevice, 2, GET_CO2_DATA);
        int len = I2cUtils.read(i2cDevice, 100L, buffer, 0, 2);
        if (len >= 2) {
            int co2 = (buffer[0] & 0xff) * 256 + (buffer[1] & 0xff);
            setValue(co2);
        }
    }
}
```

I2cCo2SensorConfig.java

```
public class I2cCo2SensorConfig extends I2cSensorConfig {
    private static final long serialVersionUID = -321642178913071370L;
    public I2cCo2SensorConfig() {
        setString(KEY_COMPONENT_NAME, "I2C Bus Sensor");
        setString(KEY_CCM_INFO_NAME, "InAirCO2");
        setString(KEY_CCM_LEVEL, CcmLevel.A_10S_0.toString());
        setString(KEY_CCM_CAST, 0);
        setInt(KEY_SENSE_INTERVAL, 10);
    }
}
```

I2cCo2Sensorでは、周期動作をしてセンサ値を取得しています。ただし、その処理は継承元のクラス I2cSensorBase クラス内で自動的に行われるため、このクラスでは、周期動作タイミングで起動される、 onReadProcess()メソッドをオーバーライドして受信処理を記述しています。周期動作間隔は、 I2cSensorConfigで定義されている設定値(KEY_SENSE_INTERVAL)で変更可能です。

I2C センサクラスとの通信をする基本的な流れは、「センサクラス起動時に周期実行処理セット」→「処理タイミングが来たら onReadProcess 起動」→「I2cUtil.write0で、データ取得コマンド発行」→「I2cUtil.read0で、データを読み取り、値を setValue0で登録」→「CCM の値が更新される」という流れになります。

4.6.2. <u>WebUI 設定画面クラス</u>



図 67: WebUI 部分のファイル構成

I2cDevicvePage.java は、Sample1、Sample2、Sample3 と同じ構造です。また対になる HTML ファ イル等も同様ですので、説明は割愛します。

5. <u>UECS-Pi アプリケーションのインストール</u>

ここでは Raspberry Pi の初期設定、及び UECS-Pi SDK で開発したアプリケーションを Raspberry Pi にインストールする手順について説明します。

No.	名称	説明	関連情報
1	Raspberry Pi	UECS-Pi SDK動作製品本体	販売サイトURL
	Model B/B+		http://jp.rs-online.com/web/
2	ACアダプタ	Raspberry Pi の microUSB 電源端子に	
		5V-750mA 以上を給電可能なもの(周辺機器を接続	
		する場合 1.0~1.2A 程度を推奨)	
3	SDカード+SDカ	SDHC 4GB(Class4)以上	
	ードリーダ	※Type B+の場合は microSD	
4	HDMI接続ディ	Raspberry Pi の初期設定時に接続して使用	
	スプレイ+USBキ		
	ーボード		
5	LAN接続ハブ/ケ	Raspberry Pi と 開発用 PC をイーサネット経由	
	ーブル	で接続するための機器とケーブル	
6	Raspbian OS	Raspberry Pi 上で動作する Linux OS	ダウンロードURL
			http://www.raspberrypi.org/downl
			oads/
7	Win32 Disk	RAW ディスクイメージを SD カードに書き込むソ	ダウンロードURL
	Imager	フトウェア	http://sourceforge.jp/projects/sfnet
		※Mac OS/Linux を使われる方は、Win32 Disk	win32diskimager/releases/
		Imager ではなく「dd」コマンドで書き込みが可能	
		です。	

5.1. <u>Raspberry Pi セットアップ</u>

表 18: Raspberry Pi セットアップに必要な機器とソフトウェア

- 5.1.1. <u>SD カードの準備</u>
- ① Raspbian OS の配布ページに移動し、zip ファイルをダウンロードします。

ŝ	BLOG	DOWNLOADS	COMMUNITY	HELP	FORUMS	RESOURCES	Q
D	OWNLOADS						
Do	ownload our New O stem image.	ut Of the Box Softwa	are (NOOBS) or a spec	ific Operating			
N	OOBS						
Be <u>N</u> C th	ginners should sta 2 <u>088 SD card</u> in the e <u>NOOBS setup gui</u>	rt with NOOBS. You e swag store, or dow <u>de</u> in our help pages	can purchase a <u>pre-in</u> mload NOOBS below a	<u>stalled</u> ind follow			
		NOOBS Offline and netwo Version: 1. Release date: 20 More info + Download Torr Download ZIP	rk install 3.10 14-09-12		NOOBS LITE Network install on Version: 1.1. Release date: 20 More info + Download Torre Download ZIP	nly 3.10 114-09-12	
0	PERATING SYST	EM IMAGES					
Ad Do do	lvanced users may ownload an image b cumentation.	wish to install a spe velow and follow the	cific Operating System image installation gu	n image. ides in our			
	C	RASPBIAN Debian Wheezy Version: Se Release date: 20 More info + Ø Download Torr Ø Download ZIP	ptember 2 14 14-09-09		PIDORA Fedora Remix Version: 20 Release date: 20 More info + Download Torre Download ZIP) 114-07-24 ent	

図 68: Rapbian OS 配布ページ

- ② ダウンロードが完了したら、ファイルを適当な場所に解凍します。
- ③ Win32 Disk Imager 配布サイトに移動し、exe ファイルをダウンロードしてインストールします。
- ④ 使用する SD カードを SD カードスロットに差し込み、その後 Win32 Disk Imager を起動します。
- ⑤ フォルダのアイコンをクリックして、OS のイメージファイル(例: 2014-06-20-wheezy-raspbian.img)を選択します。「Write」ボタンをクリックして、SD カードへの書き込みを行います。

🍓 Win32 Disk Imager 📃 📼 🛛 🗶	
-Image File Device	-
1/Raspbian OS_2014-06-20/2014-06-20-wheezy-raspbian.img 📄 [E:¥]	-
Copy MD5 Hash: Progress	
Version: 0.9.5 Cancel Read Write Exit	
	н

図 69: Win32 Disk Imager 画面

⑥ 書き込み終了後、SD カードを SD カードスロットから取り外し、Win32 Disk Imager を終了してください。

5.1.2. <u>Raspberry Pi の初期設定</u>

- Raspberry Pi の SD カードスロットに OS イメージファイルを書き込んだ SD カードを差し込みます。次にキーボード、ディスプレイ、LAN ケーブル等の周辺機器を Raspberry Pi に接続します。最後に AC アダプタを接続すると Raspberry Pi が起動します。
- 2 起動後、下記のような Raspberry Pi Software Configuration Tool(raspi-config)画面が表示され ます。

Raspberry Pi Software Cor Setup Options	Forwards that all of the SD cord of
2 Change User Password	Change password for the default u
3 Enable Boot to Desktop	Choose whether to boot into a des
4 Internationalisation Options	Set up language and regional sett
5 Enable Camera	Enable this Pi to work with the R
6 Add to Rastrack	Add this Pi to the online Raspber
7 Overclock	Configure overclocking for your P
8 Advanced Options	Configure advanced settings
9 About raspi-config	Information about this configurat
<select></select>	<finish></finish>

図 70: Raspberry Pi Software Configuration Tool(raspi-config)画面

以下、順に設定を行ってください。

A) 1 Expand Filesystem

Enter を押すと次の画面が表示されます。「OK」を選択してください

B) 2 Change User Password

Enter を押すと、画面下部で新しいパスワードを尋ねられますので、2回、同じパスワード を入力してください。

※以降、Raspberry Pi にログインを行う際に必要になりますので忘れないようにしてください

C) 3 Enable Boot to Desktop/Scratch

設定する必要はありません。

D) 4 Internationalisation Options

Enter を押すと下記のような言語、タイムゾーン、キーボードレイアウトのメニュー画面に

遷移します。



図 71:言語、タイムゾーン、キーボードレイアウトのメニュー画面

以下、順に設定を行ってください。

No.	メニュー	説明
1	I1	en_US.UTF-8、ja_JP.UTF-8の二つを有効にします。有効にするにはスペー
	Change Locale	スキーを押して下さい。デフォルトロケールは en_US.UTF-8 に設定してくだ
		さい。
2	I2	Asia を選択した後、Tokyo を選択します。
	Change	
	Timezone	
3	13	下記の順に選択してください
	Change	①Generic 105-key (Intl) PC
	Keyboard	20ther
	Layout	③Japanese
		※Japanese (PC-98xx Series)ではないので注意してください
		(4) Japanese - Japanese (OADG 109A)
		⑤The default for the keyboard layout
		⁶ No Compose Key
		⑦Yes

表 19: Internationalisation Options の設定

E) 5 Enable Camera

設定する必要はありません。

F) 6 Add to Rastrack

設定する必要はありません。

G) 7 Overclock

設定する必要はありません。

H) 8 Advanced Options

Enter を押すと下記のような詳細メニュー画面に遷移します。

Advanced Options	'Pi Software	Configuration Tool (raspi-config)
Al Overscan A2 Hostname A3 Memory Split A4 SSH A5 SPI A6 Audio A7 Update		You may need to configure oversea Set the visible name for this Pi Change the amount of memory made Enable/Disable remote command lin Enable/Disable automatic loading Force audio out through HDMI or 3 Update this tool to the latest ve
	<select></select>	<back></back>

図 72:詳細メニュー画面

以下、順に設定を行ってください。

No.	メニュー	説明
1	A1 Overscan	設定する必要はありません。
2	A2 Hostname	設定する必要はありません。
3	A3 Memory Split	16(MB)と入力して OK を選択してください。
4	A4 SSH	Enable を選択してください。
5	A5 SPI	実行する必要はありません。
6	A6 Audio	実行する必要はありません。
7	A7 Update	実行する必要はありません。

表 20: Advanced Options の設定

上記の設定を完了後、Finishを選択してください。再起動の可否を尋ねられますので Yes を選択 して、Raspberry Pi の再起動を行ってください。

5.1.3. <u>Raspberry Pi のユーザ設定</u>

Raspberry Piの再起動後、下記のようにログインユーザを尋ねられます。
 raspberrypi login:_

② 下記のログインユーザとパスワードでログインしてください。
 User: pi
 Password: 『5.1.2. Raspberry Pi の初期設定』で設定したパスワード
 (変更していない場合、パスワードは"raspberry"になっています)

③ ログインが成功した場合、画面には下記のように"ユーザ@ raspberrypi"と表示されます。
 pi@raspberrypi~\$

④ 次に root ユーザのパスワードを変更します。下記の通りに入力してください。
 pi@raspberrypi~\$ sudo passwd root

⑤ パスワードを尋ねられますので、2回、同じパスワードを入力してください。
 ※このパスワードは root ユーザのパスワードになります。初期設定で設定したもの(pi ユーザ)とは違いますので注意してください。

⑥ パスワード変更に成功後、実行ユーザを root ユーザに切り替えます。
 pi@raspberrypi~\$ su

⑦ パスワードを尋ねられますので、先ほど設定した root ユーザのパスワードを入力してください。 ユーザが切り替わると下記のように画面表示が変更されます。

root@raspberrypi:/home/pi#

以降の手順は全て、この root ユーザで実行してください。
5.1.4. <u>IP アドレス設定</u>

① 下記の通りに入力し、ネットワークの設定ファイルを開いてください。

root@raspberrypi:~# vi /etc/network/interfaces

② 本設定ファイルではIPアドレス、サブネットマスク、ゲートウェイアドレスの設定を行います。下記の通りにファイルの内容を変更してください。

auto lo		
iface lo inet loopback #iface eth0 inet dhcp		
#allow-hotplug wlan0		
#iface wlan0 inet manual		
#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf		
auto eth0		
iface eth0 inet static		
address 192.168.0.xxx (←マシンの固定 IP アドレス)		
netmask <u>xxx.xxx.xxx (</u> ←サブネットマスク)		
gateway xxx.xxx.xxx (←ゲートウェイアドレス)		

5.1.5. <u>DNS 設定</u>

下記の通りに入力し、ネットワークの設定ファイルを開いてください。

root@raspberrypi:~# vi /etc/resolv.conf

② 本設定ファイルでは DNS の設定を行います。下記の通りにファイルの内容を変更してください。
 nameserver xxx.xxx.xxx (←DNS のアドレス)

③ 設定が完了しましたら、Raspberry Piの再起動を行ってください。

root@raspberrypi:~# reboot

5.1.6. <u>シリアル通信(UART)の有効化</u>

Raspberry Pi のシリアル通信用ポート(RX/TX ピン)を利用可能に設定します。
 "/boot/cmdline.txt "を編集します。

root@raspberrypi:~# vi /boot/cmdline.txt

② 下記の通りにファイルの内容を変更してください。(改行しているように見えますが、実際は一行としてください)

dwc_otg.lpm_enable=0 rpitestmode=1 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

③ 次に、"/etc/inittab "を編集します。

root@raspberrypi:~# vi /etc/inittab

④ 以下の行(最終行)をコメントアウトしてください。

#Spawn a getty on Raspberry Pi serial line #T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

⑤ 次に、"/etc/rc.local"を編集します。

root@raspberrypi:~# vi /etc/rc.local

⑥ 下記の通りにファイルの内容を変更してください。

Print the IP address
_IP=\$(hostname 'I) || true
if ["\$_IP"]; then
 printf "My IP address is %s¥n" "\$_IP"
fi
In 's /dev/ttyAMA0 /dev/ttyS0
exit 0

5.1.7. I2C ポートの有効化

初期設定では Raspberry Pi 基盤上の I2C/SPI ポートは無効になっているため有効化します。
 "/etc/modprobe.d/raspi-blacklist.conf "を編集します。

root@raspberrypi:~# vi /etc/modprobe.d/raspi-blacklist.conf

② 下記の通りにファイルの内容を変更して下さい。

blacklist spi and i2c by default (many users don't need them)

blacklist spi-bcm2708

#blacklist i2c-bcm2708

③ 認識させます。

root@raspberrypi:~# modprobe i2c-dev

④ 起動時に自動でロードさせるため、"/etc/modules "に i2c-dev を追加します。

root@raspberrypi:~# vi /etc/modules

etc/modules: kernel modules to load boot time.

•••

•••

snd-bcm2835

i2c-dev

5.1.8. Tomcat の設定

No.	名称	説明	ダウンロード URL
1	Tomcat	オープンソースのHTTPサーバ	http://tomcat.apache.org/download-
			<u>70.cgi</u>
2	WinSCP	MS-Windows上で動く フリーのFTP、FTPS、	http://winscp.net/eng/download.ph
		SFTPクライアントプログラム。	p
2	PuTTY	フリーの Telnet/SSH クライアント。SSH は	http://hp.vector.co.jp/authors/VA02
		SSH1 と SSH2 プロトコルが実装されている。	4651/PuTTYkj.html

表 21: Tomcat の設定に必要なソフトウェア

アプリを動作させるために、まず Tomcat (Web サーバ)のインストールと設定を行います。ここでは Tomcat7 を対象に説明を行います。またここまでで Raspberry Pi のネットワーク設定が完了 しているので、ここからは PuTTY と WinSCP を使った操作を行います。

- ① Raspberry Pi からディスプレイ、キーボード等を取り外し、SD カード、ネットワークケーブ ル、AC アダプタのみが接続されている状態にします。
- ② WinSCP(または他の SCP クライアントソフト)をダウンロードして起動します。ここでは WinSCPを例に説明を行います。
- ③ WinSCP のログイン画面から「新しいサイト」をクリックし、下図にあるようにデータを入力 します。ユーザ名やホスト名は、これまで設定した内容を入力して下さい。ここではルートユ ーザでログインし、ルートフォルダで作業する事を想定します。

MinSCP ログイン		
「 新しいサイト	セッション 転送プロトコル(F) SFTP ホスト名(H) 192.168.0.70 ユーザ名(U) パス・ root ・・・・ 編集(F)	ポート番号(B) 22 ワード(P) 設定(D) ▼
管理(M) ▼	 ごうイン マ 	閉じる ヘルプ(H)

図 73: WinSCP ログイン画面

④ 「ログイン」ボタンをクリックし、パスワードを入力して Rapberry Pi にログイン出来れば成 功です。

ローカル(L) マーク(M)	ファイル(F)	コマンド	(C) セッション(S) オブション(O) リモート(B)	ヘルプ(H)	
		1.08			
H 2 5 10149(2)	1 (P) (P)	100	() +1-(Q) • #EXEMP - 73/01-	• 📂 •	
RPi C MILLI	セッション				
🕞 र • 🚰 💟 🔄 •	1	*	/ - > • 🚰 🗹 🗢 • 🔶 🛅 🛅 (กิ 🗃 强 วราวม	いの検索(F)
「油 アップロード(L) (油	» +	*	ダウンロード(L) 🙀 📝 総集(E) 🗙 🛃 🕞 プ	ロノ(ティ(P) 🔐 💦	
C:¥¥war8iar¥2014100)3	1			
名前 松陽子	サイズ	8 86	1 私張子 2	サイズ	更新日時
2		0 2.			2014/10
Duecs-pi-basic.iar	101.23	A	in		2014/06
uecs-pi-basic.war	9.828 K/B	w	toot		2014/10
uecs-pi-sdk.jar	41,836 B	A	lev		2014/10
uecs-pi-sdk.war	9,728 KiB	w.	tc		2014/10
	2000000		ome		2014/10
		1	b		2014/06
			ost+found		2014/06
			nedia		2014/06
		1	nnt		2014/05
			pt		2014/10
		1	iroc		1970/01
			pot		2014/10
		1	un		2014/10
		1	bin		2014/06
			elinux		2012/06
		1	rv		2014/06
		1	γs		1970/01
			mp		2014/10
			isr		2014/06
			ar		2014/06
•		1 4	m		

図 74: WinSCP ログイン後画面(画像の画面はルートフォルダ)

⑤ Tomcat をインストールするために、ブラウザ等で公式サイトのダウンロードページ (<u>http://tomcat.apache.org/download-70.cgi</u>)を開き、Core という項目の tar.gz リンクをクリ ックしてファイルをダウンロードします。バージョンは最新のものをダウンロードしてください。

	Apache Tomcat	Search the Site	ch	
Apache Tomcat	Tomcat 7 Downloads			
Home Taglibs Maven Plugin	Welcome to the Tomcat 7.x download page. This the latest version of Tomcat 7.0.x, as well as link	s page provides download links for obtain s to the archives of older releases.	ning	
Download	Quick Navigation			
Which version? Tomcat 8.0 Tomcat 7.0	KEYS 7.0.56 Browse Archives			
Tomcat 6.0	Release Integrity			
Tomcat Connectors Tomcat Native Archives	You must <u>verify</u> the integrity of the downloaded every release file. This signature should be mate	files. We provide OpenPGP signatures fo hed against the <u>KEYS</u> file which contains t	r the	
Documentation Tomcat 8.0	/e also provide an MD5 checksum for even uld calculate a checksum for your downlo	y bad,		
Tomcat 7.0 Tomcat 6.0	Mine are			
Tomcat Connectors Tomcat Native Wiki Migration Guide	You are currently using http://www.carfab.cor problem with this mirror, please select another <i>backup</i> mirrors (at the end of the mirrors list) th	n/apachesoftware/. If you encounter a mirror. If all mirrors are failing, there are at should be available.		
Problems? Security Reports	Other mirrors: http://www.carfab.com/apachesoftw	are/ Change		
Find help FAO	7.0.56			
Mailing Lists Bug Database IRC	Please see the <u>README</u> file for packaging inform contains.	ation. It explains what every distribution		
Get Involved	Binary Distributions			
Overview SVN Repositories	Core:			
Buildbot Reviewboard	 <u>zip (pgp, mds)</u> <u>tar.gz (pgp, md5)</u> 			
Tools	 32 bit Windows zip (ugp, md5) 64-bit Windows zip (ugp, md5) 			
Media	 64-bit Itanium Windows zip (pgp, 	<u>md5</u>)		

図 75: Tomcat ダウンロードページ

ここでは説明のため、Tomcat 7.0.56 を利用します。ダウンロードが完了したら、WinSCP を使って apache-tomcat-7.0.56.tar.gz ファイルを Raspberry pi にコピーします。

⑥ 次に PuTTY を起動します。

Reputity Configuration		×
Category:		
Session	Basic options for your PuTTYses	ssion
	Specify the destination you want to connec Host Name (or IP address)	ct to Port
Bell Features	192.168.0.70 Connection type: ⑦ Raw ⑦ Telnet ⑦ Rlogin @ SSH	22
Window Appearance Behaviour Translation Selection Colours Connection Data Proxy Telnet Rlogin Serial	Load, save or delete a stored session Saved Sessions RPi Default Settings RPi SheevaPlug SheevaPlug(SSH)	Load Save Delete
	Close window on exit:	
About	Open	Cancel

図 76: PuTTY コンフィグ画面

コンフィグ画面で、上図のように設定します。Host Name はこれまで設定した内容を入力して下 さい。

⑦ PuTTY で Raspberry Pi にアクセスします。以降のコマンドは PuTTY から実行します。



図 77: PuTTY コマンド入力画面

⑧ ルートフォルダに移動します。

root@raspberrypi:~# cd ../

⑨ Raspberry pi 側でコピーされたファイルを下記のように解凍します。

root@raspberrypi:/# tar xvzf apache-tomcat-7.0.56.tar.gz

 ① ファイルの解凍後、Tomcat のディレクトリが作成されますのでディレクトリごと下記のパス (/opt 以下)にコピーしてください。

root@raspberrypi:/# mv apache-tomcat-7.0.56/ /opt/apache-tomcat

① コピー後、下記の通りに入力し、環境設定ファイルを新規作成します。

root@raspberrypi:/# vi /opt/apache-tomcat/bin/setenv.sh

① 環境設定ファイルには下記の内容を記述してください。

#!/bin/sh

export CATALINA_OPTS="-Duser.language=ja -Duser.country=JP -Xms256M -Xmx384M -XX:MaxPermSize=128M"

③ ファイルの保存後、下記の通りに入力して環境設定ファイルの権限を変更します。
 root@raspberrypi:/# chmod 755 /opt/apache-tomcat/bin/seteny.sh

④ ブラウザからアクセスする用にポート番号をデフォルトの 8080 番から 80 番に変更します。
 まず、下記の通りに入力し、設定ファイルを開いてください。

root@raspberrypi:/# vi /opt/apache-tomcat/conf/server.xml

⑤ 下記のように port="8080"と設定されている部分を port="80"に変更します。

~~~~省略~~~~~ <Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" /> ~~~~~省略~~~~~

③ デフォルトのログ設定では長期運用を行うと、ログファイルが肥大化し、ディスク容量を圧迫 する恐れがあります。簡易的な対策として、既存のログ設定ファイルをリネームし、ログ出力 を停止させます。(ここでは改行が入っていますが、全て1行で入力してください)

root@raspberrypi:/# mv /opt/apache-tomcat/conf/logging.properties /opt/apache-tomcat/conf/logging.properties.org

### 5.1.9. 各種スクリプトの転送と登録

ここでは UECS-Pi アプリケーションの自動起動用スクリプトや IP アドレスをアプリケーション 上から変更可能とするためのシェルスクリプトを Raspberry Pi にインストールします。 scripts.tar.gz を WinSCP 等で Raspberry Pi にコピーします。scripts.tar.gz は、release フォルダに 入っています。



図 78: script.tar.gz の位置

① 下記の通りに入力し、スクリプトファイルを展開し、opt以下に保存します。

root@raspberrypi:/# tar zxvf scripts.tar.gz root@raspberrypi:/# mv scripts /opt/

② ファイルの保存後、下記の通りに入力してファイルの権限を変更します。
 root@raspberrypi:/# chmod 755 /opt/scripts/\*.sh

③ 次に、Tomcat 自動起動スクリプトが実行されるよう cron にスクリプトの登録を行います。下 記の通りに入力し、設定ファイルを開きます。

root@raspberrypi:/# crontab -e

④ 1分置きに起動スクリプト(process\_check.sh)が実行されるよう、下記の内容を登録してください。

\*/1 \* \* \* \* /opt/scripts/process\_check.sh

⑤ WinSCP から scripts.tar.gz ファイルを削除します。

# 起動と終了

起動スクリプトを cron に登録した場合、Tomcat は自動で起動するようになりますが、手動で Tomcat を起動、または終了させたい場合は下記のスクリプトを実行してください。

### 記動したい場合

root@raspberrypi:/# /opt/apache-tomcat/bin/startup.sh

② 終了したい場合

root@raspberrypi:/# /opt/apache-tomcat/bin/shutdown.sh

- 5.2. <u>アプリケーションインストール</u>
  - 5.2.1. WAR ファイルファイルの作成
  - ① Raspberry Piのセットアップが完了し、SD カード、AC アダプタ、ネットワークケーブルが接続されているのを確認します。
  - ② WinSCP を起動し、Raspberry Pi に接続します。
  - ③ Eclipse を起動します。
  - ④ UECS-Pi SDK に含まれている web.xml のコメントアウトを以下のように外します。



図 79:web.xmlの場所



図 80:コメントアウトを外す箇所

⑤ UECS-Pi SDK に含まれている build.xml を右クリックし「実行」→「2 And ビルド」をクリック します。



図 81: build.xml、uecs-pi-app.war の場所

⑥ uecs-pi-app.war が新しく作成されるので、WinSCP で/root/フォルダにコピーします。

5.2.2. 実機インストール

- ① PuTTY を起動し、Raspberry Pi に接続します。
- ② 下記の通りに入力し、uecs-pi-app.war ファイルを Tomcat の webapps フォルダに移動します。 root@raspberrypi:/# mv /root/uecs-pi-app.war /opt/apache-tomcat/webapps/

③ Tomcat を終了→起動し、war ファイルを読み込ませて立ち上げます。 root@raspberrypi:/# /opt/apache-tomcat/bin/shutdown.sh root@raspberrypi:/# /opt/script/process\_check.sh

- ④ ブラウザ (Internet Explorer 等)から Raspberry Pi に設定した IP アドレスの URL(http://[ノード IP アドレス]/uecs-pi-app/)に接続します。
- ⑤ ログイン画面が表示されるので、管理パスワードに admin と入力してログインして下さい。
   ※UECS-Pi SDK ログインパスワードは、ログイン後にノード設定画面で変更可能です。

| UECS-Fi<br>「ログイン」 |  |
|-------------------|--|
| Powerd by UECS.   |  |
| 図 82・ログインページ      |  |

⑥ ログイン後、トップページが表示されたらセットアップは成功です。



# 図 83:トップページ

5.2.3. web. xml について

UECS-Pi SDK に含まれる Web アプリケーション設定ファイル(web.xml)内には、アプリケーション 動作モードの定義が含まれており、通常はコメントアウトがされています。



図 84:web.xmlの場所



図 85:コメントアウト箇所

Eclipse で UECS-Pi SDK を動作させる際は、通常ここをコメントアウトしたまま実行して下さい。 Wicket の WebUI 処理のエラー情報が詳細に画面表示され、シリアル通信機能や、I2C 通信機能など、開 発用 PC ではテストできない機能が OFF となります。また実機インストールして、デバイス通信のテス トを行う際は、このコメントアウトを外して下さい。

#### 6. <u>利用ライセンス</u>

| No.                                  | 項目                  | 参照 URL                                   |
|--------------------------------------|---------------------|------------------------------------------|
| 1                                    | Raspbian OS (Linux) | http://www.raspbian.org/                 |
| 2                                    | Apache Tomcat       | http://tomcat.apache.org/                |
| 3                                    | Apache Wicket       | http://wicket.apache.org/                |
| 4                                    | Apache commons      | http://commons.apache.org/               |
| 5                                    | Pi4J                | http://pi4j.com/                         |
| 6                                    | SQLite JDBC Driver  | https://bitbucket.org/xerial/sqlite-jdbc |
| 7                                    | OrmLite             | http://ormlite.com/                      |
| 8                                    | SLF4J               | http://www.slf4j.org/                    |
| 9                                    | Logback             | http://logback.gos.ch/                   |
| 表 22: UECS-Pi SDK の使用オープンソースソフトウェア一覧 |                     |                                          |

### 7. <u>免責事項</u>

本製品を使用したことによる一切の損害(一次的、二次的に関わらず)に対し、当社では責任を負いません。

#### 8. お問い合わせ

本製品は無償製品ですので、基本的に当社はサポート保証責任を負いません。ただし、機能追加・品質改 善は随時行ってまいりますので、お気づきの点、ご質問、ご要望がございましたら、下記よりお問い合わせ ください。

(サポートメールアドレス): <u>support@wa-bit.com</u>