分散組込みシステム向き Webベース開発環境

プラウィーン アモーンタマウット[1,a)]　早川 栄一[2,b)]

概要: CPS や IoT などの分散組込みシステムは, センサ・アクチュエータを備えた多様な組込みシステムとサーバ群とをネットワークを介して接続するシステムである. 幅広い分野への応用を可能にするには, 多様な組込みシステムのプログラミングやテストを容易に行える開発環境が必要になる. 本研究は, CPS/IoT の要求を考慮して, Web ベースで容易にコーディングおよびセンサやアクチュエータのモニタリングが可能な開発環境を構築した. 本環境の特徴は次のとおりである: (1) コマンドベースおよびスクリプト言語でのデバイスのコーディングが可能, (2) タイムスタンプを含めた通信データのトレースおよび閲覧が可能, (3) 複数の組込み機器のアクセス状況の閲覧が可能.

キーワード: CPS, IoT, Web, プログラミング環境, 分散組込みシステム

Web-based Programming Environment for Networked Embedded System

Praween AMONTAMAVUT[1,a)]　Eiichi HAYAKAWA[2,b)]

Abstract: Web-based distributed embedded system is a system that included CPS and IoT consists of heterogenous embedded system, sensors, and actuators connected to cloud servers. Developing the system requires simple prototyping and programming environment in order to support practical use in various fields. We developed the Web-based coding and monitoring environment for the system. The features of the system are following: (1) presenting the simple coding interface with command line and Javascript via the Web server, (2) browsing the accessible embedded system nodes on the browsers, and (3) monitoring and browsing the communication behavior of the sensors and actuators on the browser.

Keywords: CPS, IoT, Web, Networked Embedded System, Web-based Programming Environment

1. はじめに

サイバーフィジカルシステム (CPS) [1][2][3] は, センサやアクチュエータを備えたフィジカルシステムである組込みシステムと，ネットワーク上のサーバによる計算・情報の処理というサイバーシステムを結合してインタラクションをすることにより，実世界に対してより高度なサービスを提供するシステムであり， [1] と [2] などで示されているように, 現在様々な研究開発が行われている. また, インターネットを注目して, モノにおいて人間を支えるインタラクションが可能なサービスやモノへのアクチュエーションなどを可能にさせる情報技術であるモノのインターネット (IoT) [4] の概念も注目を集め, これについても多くの研究開発が行われている. CPS や IoT は類似の概念であり [5], どちらもサイバー世界において実世界という物理システムと情報を結びつける分散組込みシステムとして考えることができる.

このような分散組込みシステムは, センサ・アクチュエータを備えた多様な組込みシステムを結び付けるサーバ群とネットワークを介して接続するシステムによって構成することができる. 農業や医療, 交通, ホームオートメション, システム教育など幅広い分野への応用を可能にする IT・ICT の統合システムの構築が行われている [5].

このような分散型の組込みシステムのプログラミングは容易ではない. その理由としては, 複数，多種のデバイスを扱う必要があり , そのためのプログラミング環境を個別

[1] 拓殖大学大学院工学研究科電子情報工学専攻
Takushoku University, Graduate School of Engineering, Electronics and Information Sciences
[2] 拓殖大学工学部情報工学科
Takushoku University, Faculty of Engineering, Department of Computer Science
[a)] praween@hykwlab.org
[b)] hayakawa@cs.takushoku-u.ac.jp

に用意する必要があることである．環境そのものや API が異なっていることが多く, それらの違いを吸収するためのライブラリ群を用意する必要があるが, そのためのコストは種類や台数に比例して高くなっていく．また, 離れた場所に設置されたデバイスを操作することから，デバイスの状況やデータのやりとりを確認することが難しくなっている．これらのデバイス群のモニタリングは, マシンの種類によって個別のツールを用意しなければならないことが多く, 手間がかかる．さらに, 時間制約を持つデバイスの場合には, ネットワーク構成の違いやプロトコルによる通信遅延を意識する必要があるが，そのための監視システムが必要となる．

本研究は, 上記のような CPS/IoT の要求を考慮し満たした分散組込みシステムの開発環境を容易に開発が可能にするのが目的である．

2. 要求分析

本システムの要求分析は次のとおりである．

(1) **複数のデバイスに対応した環境をインストールせずに利用できること**

ネットワーク型の組込みシステムでは, 複数, 多種のデバイスを用いる．近年では, Raspberry Pi や Intel Edison などのデバイスを安価に入手することが可能であるが, それらは同じ Linux が動作している環境でも, 入出力の構造や API が異なるために，単一のプログラムでアクセスすることができない．これらのボード類やセンサが混在した環境を用いる場合に，それぞれに適したプログラミング環境を用意するのは非常に手間がかかる．Scratch[6] や Python のようにボード内にプログラミング環境を用意できる場合もあるが，多くの場合, 開発環境としては非力なことが多い．Eclipse[7] のような本格的な IDE はインストールに時間がかかることもあり, 利用者がすぐに利用可能にするには, 開発環境をインストールせずに利用できることが望ましい．

(2) **プロトタイピングが容易なこと**

IoT のシステムでは, Web を利用していることから, 動的なデバイスやサービスの追加が容易である必要がある．それにはデバイスやシステム構成まで含めて, プログラミングを行う前に容易にプロトタイピングを行い, 構成についてテストが容易になっている必要がある．また, 記述性がよく Web と親和性の高いプログラミング言語が利用できる必要がある．

(3) **システムや通信の状態をプログラミング環境と連動させてモニタリングできること**

CPS などで時間制約のあるシステム構築を行う場合, ネットワークでの通信遅延や構成を見直す必要がでてくる．これには, 通信状態のモニタリングが必要となる．通常, このようなモニタリングは, 別のツールを用いることが多いが, これはシステムインストールを伴うために, 利用コストが高くなる．これらのモニタリングおよび, その結果の可視化ツールが開発システムと同一の環境内で利用できる必要がある．

3. システムの特徴

前章の要求分析を受け，次のような特徴を持つ分散組込みシステム向き Web ベース開発環境を構築した．

(1) **Web ベースのプログラミング環境の提供**

開発環境をインストールせずに CPS/IoT のプログラミングが可能なように, Web ベースの開発環境を提供する．これによって Web ブラウザだけ用意できれば, ソフトウェアをインストールせずにすぐにプログラミングが可能になる．特に IoT の環境では通信や操作の基盤として Web を用いることが多く, 親和性が高いという利点もある．

(2) **コマンドラインとプログラミング言語の両方のサポート**

システムの構成や動作を確認するために, 簡易なコマンドラインインタフェースを提供する．これによって, 対話的にシステム状態を変更することが可能になり, システム構成の変更に追従することが容易になる．また，プログラミングについては Web 環境と親和性の高い Javascript の API およびコーディング環境を用意した．これによって, プロトタイピングや実験から，実際のシステムまでをシームレスに実装していくことが可能になる．

(3) **Web ベースのモニタリングおよび可視化システムの提供**

組込みシステム間の通信やノードの状態をモニタリングして，可視化する環境を提供する．これによって, ネットワーク構造や状態によるシステムの変化や問題点を発見しやすくする．この環境は, Web ベースで提供することで, 上記のプログラミング環境と統合して利用することが可能である．

4. 設計

本節は分散組込みシステム向き Web ベース開発環境の設計と実装を述べる．

4.1 全体構成

本システムの階層を図 1 に示す．本システムは三つのレイヤから構成する．

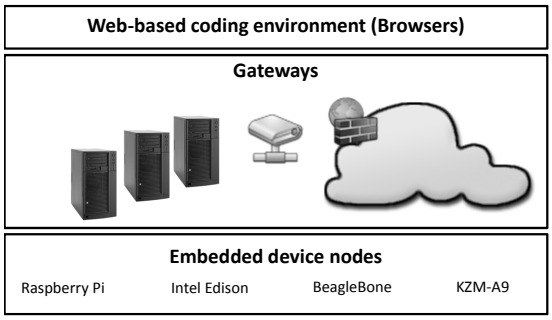

図 1　システムの階層
Fig. 1　Layer of the system.

(1) **組込み機器ノード** (Embedded device nodes)
センサやアクチュエータを接続して, ネットワークを介してサービスを受けるための組込み機器ノードを用意する. Raspberry Pi や Intel Edison, BeagleBone, KZM-A9 など Linux が動作するシステムを想定している. プログラムのデプロイやシステムのモニタリングを行って, サーバ群と協調して動作するように，組込み機器ノードの制御プログラム(以下，Blue-Sky モジュールと呼ぶ）を実行する. 制御プログラムによってシステムの差異を吸収することが可能になる.

(2) **ゲートウェイ** (Gateways)
多種, 複数の組込み機器を統合して, プログラミング環境を提供し, 外部のクラウドサーバへのサービス中継を行う部分である. コーディングおよび可視化の環境を提供する Web サーバ機能と，組込み機器ノードと通信するための API から構成されている. API を介した通信はログとして保存することで, やりとりを可視化することが可能になる. また, バックエンドのクラウドサービスとしては, 分散ファイルシステムである GlusterFS や Key-Value Store である redis を用いたデータストアを用意して，センサデータの保存を可能にしている. ゲートウェイ上では組込み機器ノードは仮想的なデバイスとして管理しているので，ノードで不足している機能はゲートウェイでソフトウェアにより補完することも可能である.

(3) **Web ベースコーディング環境** (Web-based coding environment)
開発者はブラウザにおける Web 上で提供しているプログラミング環境を用いて, センサやアクチュエータのプログラミングやテストを行うことができる. 本環境が対象とする言語はコマンドスクリプトと Javascript である. コマンドスクリプトは本コーディング環境のプロトタイピングで用いるスクリプトである. javascript は開発言語として利用できる. また, データ通信のシーケンスやデバイス情報の可視化環境も併せて提供する.

この階層構成に基づき, 図 2 の全体構成を採用した.

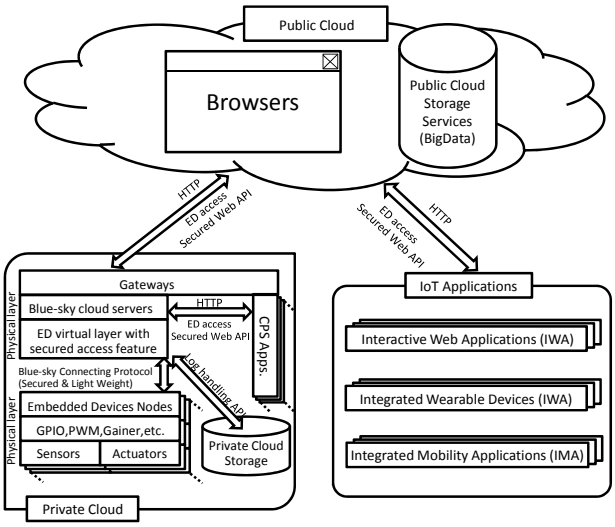

図 2　システムの全体構成
Fig. 2　Overall structure of the system.

Blue-sky [8] は, プライベートクラウド群とそれらの間でデータの共有を行うパブリッククラウドからなるハイブリッドクラウド構成となっている. このような構造にすることで, セキュリティが必ずしも強固ではないノード群を保護し, デバイスの管理を容易にしている. CPS/IoT では, 複数のデバイスのデータを共有して利用できるところが大きな利点となっている. これらを安全に行うために, ゲートウェイを介してパブリッククラウドへ接続する. このようなハイブリッドクラウド上に Sensing Instrument as a Service (SIaaS)[9] を構築することで, 安全かつ管理の容易なシステムの構築が可能になる.

4.2　Blue-Sky モジュール:組込み機器ノード制御プログラムの設計

組込み機器のノードは，セキュアに実行するためにゲートウェイ内部のプライベートネットワークで動作する.

各ノード上では Blue-Sky モジュールと呼ぶ制御プログラムが動作している. 制御プログラムの機能としては，ゲートウェイから来たデバイスアクセスの GPIO への中継，ゲートウェイへの接続，ノード状態やログのゲートウェイへの送信機能を持つ. ノードの発見とゲートウェイへの接続は，プライベートネットワーク内のブロードキャストによって行う.

4.3　ゲートウェイの設計

ゲートウェイは，Web サーバとしてプログラミング機能を提供するとともに, パブリックネットワークからの組込み機器へのアクセス, 外部のクラウドサービスへの中継, および開発したプログラムのデプロイのサポートを行う. ゲートウェイに接続されたノードは, ゲートウェイ内部で仮想的な組込みデバイスとして状態を保持して管理するこ

とができる. これによって, デバイスの変化の検知やデバイスで不足する機能のソフトウェア的な実装などを可能にする. また, ノードデバイスの外部クラウドサービスへのアクセスをプロキシとして中継することで, アクセス制御を行いセキュリティを向上させる.

4.4 コマンドおよび API の設計

表 1 コマンドスクリプトにおける命令リスト
Table 1 command list of the command script for sensing/actuating.

コマンド名	機能
lsn	Local で接続する各デバイスが提供するセンシング・アクチュエーティング機能を Local API として呼び出すコマンドであるので, Local コマンドと呼ぶ.
gsn	Public で接続する各デバイスが提供するセンシング・アクチュエーティング機能を Public API として呼び出すので, Public コマンドと呼ぶ. これは本環境が二つ以上に設置されたゲートウェイの Public Domain がある場合に扱われる.
loop	コマンドを繰り返す.
sleep	ミリ秒の単位でスリープさせる.

本 API は, Rest API の概念を基づく, 設計されたハイブリッド API である. ハイブリッド API は, Public API と Local API の二つに分類される. Public API は, Public Cloud サービスからでも本 API を呼び出すことができる. 一方, Local API は, アクセス制限を設定可能な API であり, 特に物理世界に影響を与えるアクチュエーションに関する API を想定している.

表 1 は, 本 API に応じるコマンドスクリプトのリストに示す. これらのコマンドは, センシング・アクチューションをテストする時に用いる. 例えば, GPIO pin 21 に繋がる LED ノード IP: x.x.x.x に点灯するテストコマンドを実行する場合は, `lsn x.x.x.x gpio set 21 1` というコマンドを指定する.

表 2 javascript におけるセンシング・アクチュエーションの API リスト
Table 2 list of the Javascript sensing/actuating API functions.

関数名	引数	機能
onLed	第 1: LED のノード IP 第 2: ノードに指した LED のピン番号	Local の LED を点灯させる.
offLed	第 1: LED のノード IP 第 2: ノードに指した LED のピン番号	Local の LED を消灯させる.
Sleep	スリープ時間	ミリ秒の単位にスリープさせる.

表 2 は, 本 API に応じる javascript ベース API のリストに示す. この API は, 上記で述べたハイブリッド API から作成した API であり, ユーザがこれを拡張拡張したり異なる API 関数を定義したりすることが可能である. 例えば, 表 2 の onLed の場合は, lsn を実装する関数を拡張した関数であり, `onLed('x.x.x.x', '21');` で実行するときには, GPIO pin 21 に繋がる LED ノード IP:x.x.x.x が点灯する.

5. 実現

本章では，これらの設計に基づくシステムの実現について述べる.

5.1 Web ベース開発環境の実行例

図 3 は分散組込みシステム向き Web ベース開発環境のスクリーンショットを表す. 本環境の画面と機能の実装に関する詳細を述べる.

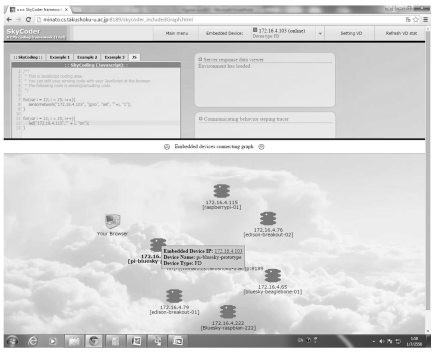

図 3 分散組込みシステム向き Web ベース開発環境
Fig. 3 A web-based environment for networked embedded system.

本開発環境は, 画面上部から次の 6 つのコンポーネントから構成した.

(A) **汎用ツールバーのボタン**
汎用ツールバーは, ボックス (C), (D), (E) などの表示/非表示を指定するボタンが配置される. ブラウザの狭い画面を有効に利用するために用いられる.

(B) **コーディングボックス**
コーディングボックスはコード入力を実際に行うボックスである. 二つの言語を対応するために, タブとして配置する. コーディングボックスの下部には実行ボタンや保存ボタンを配置して，プログラムのデプロイを容易にする.

(C) **サーバのレスポンスデータの監視ボックス**
サーバからのレスポンスを監視するボックスである. レスポンスデータは大きくないので, ボックスの高さは固定になっている.

(D) **データの通信を監視するボックス**

データの通信の監視結果を表示する．通信によって大きさが異なるので, ボックスの高さは調整可能に設計される．

(E) 全デバイスの接続状態を監視するボックス
接続する組込み機器全体の接続状態を表示する．これは数が多いため, スクロール可能である．

(F) 全デバイスの接続状態の可視化
接続する多様な組み込み機器が 50 個以上の場合には, (E) では, 見づらくなるため, その接続状態をグラフで可視化する．

プログラミングした後実行ボタンを押すと, オンラインで対象とするセンサ・アクチュエータのノードと連携して，物理エンティティの変化を監視することが可能になる．

実行中には, ボックス (C) で各命令や API を提供するサーバのレスポンスデータをオンラインに監視することができる．実行終了後に, ボックス (D) によって, データ通信の時系列データを確認することができる．これは，ブラウザ，ゲートウェイと組込みシステムそれぞれで 通信プロトコルに組み込まれたタイムスタンプを用いて表示する．

これによって, システムを開発する際に, 各アプリケーションに応じる各命令の実行の待ち時間を確認することが可能になる．これによって, アプリケーション開発時にチューニングや最適化を行うことができる．例えば, 待ち時間が長い場合に, サーバの負荷やネットワークのトラフィック量に応じて, センシングの頻度を減らしたり，他のノードに切り替えることも可能になる．

5.2 タイムスタンプ付属データの通信シーケンスの監視

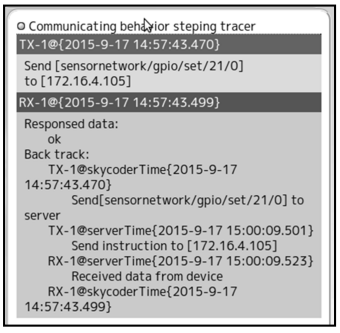

図 4 タイムスタンプ付属データの通信シーケンスの監視
Fig. 4 Monitoring the Sequence of Data Communication with Timestamps.

図 4 は (D) の実現に示す．タイムスタンプは, Bluesky-server[8] が提供する Server-ET-Timestamp と ET-Server-Timestamp である HTTP のレスポンスヘッダにスタンプされたデータを, コールバックで DOM のエレメントに書き加える方法で実現する．Bluesky-server のタイムスタンプのデータは, システム時刻の生データであるので, ブラウザで GTM に変換する必要である．Server-ET-Timestamp は, 組込みノードに命令を渡す前にスタンプされた時刻であり, ET-Server-Timestamp は組込みノードがその命令を受け取って反応したときにスタンプされた時刻である．ブラウザのタイムスタンプは, ブラウザの時刻で Bluesky-server にデータを送受する前後にスタンプされたデータで実現した．

6. 評価実験

本章は, 評価実験, 実験結果と考察を述べる．

6.1 実験

本実験は, 開発コストとシステムの性能に関して, 本環境のと従来の開発のを比較する開発実験の評価である．開発実験は, 開発コストが分かるように簡潔なアプリケーション (以下, アプリ) を開発する．アプリは, ブラウザ上のボタンにクリックしたことによって, Raspberry Pi に付け加えた LED ノードを, 点灯したり, 消灯したりするものである．本アプリでは, データの通信のタイムスタンプが確認できるようにする．アプリの開発は, 本環境におけるアプリ開発と従来の開発方法の両方で開発する．本環境による開発は図 3 で示したブラウザだけで開発する．従来の開発は, RaspberryPi 上に起動する node.js における Web サーバと LED の点灯・消灯の機能を開発した後にブラウザ上起動するアプリを開発するものである．そして, 開発コストとシステムの性能をそれぞれに把握する．開発コストはコーディングと開発時間である．システムの性能は, データ通信シーケンスにおいて LED を点灯・消灯するのにどの程度遅延するかについて, ラウンドトリップディレイタイム (RTT) を測定する．

6.2 結果と考察

表 3 は, 開発コストの比較を示す．本開発環境によるアプリの開発は, サーバを開発しなくてよいため, サーバの開発にかかるコストが 0 になり, 6 行の少ないコードで 5 分だけで本アプリを簡易に開発することが可能である．

表 3 開発コスト
Table 3 Cost of Development.

		アプリ	サーバ	合計
従来の開発	コーディング行数 (行)	155	90	245
	開発時間 (分)	100	90	180
本環境での開発	コーディング行数 (行)	6	0	6
	開発時間 (分)	5	0	5

表 4 は, Local Network のデータの通信経路において LED を一回に点灯するのにかかる RTT を計測してシステムのオーバヘッドを示す．本環境では, 従来の開発方法より 2m 秒実行時間が少ないことが明らかになった．

表 4　システムの性能
Table 4　Performance of System.

	RTT (msec)
従来の開発方法で開発したアプリ	31
本環境で開発したアプリ	29

7. 関連研究

電力設備専用である遠隔制御・監視可能なネットワーク組み込みシステムの統合 [10] が存在する．このシステムは Telnet ベースと Web server ベースのシステムとして開発された．両者とも組込みシステム上で起動するプログラムである．メモリの資源の制限やプロセッサの処理能力に制限がある組込みシステムに対しては, Telnet インタフェースを提供する．Web server ベースは小さいサーバとして動作することから, 計算機資源がある程度豊富なシステムを対象としている．しかし, Telnet ベースのものはデバイスに直接の外部コマンドでアクセスできることから, セキュリティ面で問題がある．特に, ターゲットとなる電力設備は, 外部から自由にアクセスかつ制御が可能となっていると問題を引き起こす可能性がある．

Xively[11] は, プロダクトとユーザとの間をセキュアに接続するクラウドサービスであり, IoT のデータが備えるスケーラビリティへの対応や, データベース, 可視化機構を提供している．Xively では, 通信自体の時間情報は取得することができない．そのために, 分散型の組込みシステムでは, 通信オーバヘッドへの対応が難しい．本システムでは, センサデータそのものの時間情報に加えて, 通信自体の時間情報を取得し, 可視化することができる．これによって, 通信まで含めたチューニングを可能になる．

Orion[12] は, 2011 年に Eclipse Public License (EPL) の基に開始されたプロジェクトである．このプロジェクトは, ブラウザにおけるクラウド上で HTML, JavaScript, CSS などを含む Web ページが開発できる統合ツールを提供するのが目標である．本システムでは CPS/IoT の要求であるセンサデータや通信シーケンスを監視可能であり, それらのデータを統合環境内で同時に表示することができる点が異なる．

8. おわりに

本報告では, CPS や IoT などの分散組込みシステム向き Web ベース開発環境を開発について述べた．本開発環境はブラウザにおいて, プロトタイピング向きのコマンドラインインタフェースと，プログラミング向きの Javascript ベースの両方の環境を提供した．また，通信シーケンスについてタイムスタンプ付きのデータを取得し，ブラウザで表示させることにより，接続状態やデータ通信の確認を可能にした．実際に，簡単なアプリケーションを開発して，本システムを用いることで容易にプログラミング可能であることを示した．

今後の課題は，現在ブラウザで作成したプログラムの別マシンへのデプロイ機能を実装することである．また, 現在の Javascript API は低レベルのものが用意されているだけなので，これをクラス化, 階層化することで, マシンや環境の変更に対して, より適合しやすいものとする．さらに，このシステムを CPS/IoT の題材に対して適用し, 本システムの有効性について検証を行う予定である．

参考文献

[1] Lee, E. A.: Cyber-Physical Systems: Design Challenges, *11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, pp. 364–369 (2008).
[2] Rajkumar, R. R. et al.: Cyber-Physical Systems: The Next Computing Revolution, *Proceedings of the 47th Design Automation Conference*, pp. 731–736 (2010).
[3] steering Group, T. C.: Cyber-Physical Systems Executive Summary, The CPS steering Group (online), available from ⟨http://iccps2012.cse.wustl.edu/_doc/CPS-Executive-Summary.pdf⟩ (accessed 2015-07-01).
[4] Gubbi, J. et al.: Internet of Things (IoT) A vision architectural elements and future directions, *Future Generation Computer Systems 29*, pp. 1645–1660 (2013).
[5] 岩野和夫，高島洋典: サイバーフィジカルシステムと IoT(モノのインタネット)，情報処理，Vol. 57, No. 11, pp. 826–834 (2015).
[6] Imai, T.: —Scratch と Raspberry Pi の遊び方—,（オンライン），入手先 ⟨http://pushl.net/teach/⟩（参照 2015-07-01）.
[7] Foundation, E.: What is Eclipse and the Eclipse Foundation?, The Eclipse Foundation (online), available from ⟨https://eclipse.org/org/⟩ (accessed 2015-06-20).
[8] AMONTAMAVUT, P.: Bluesky-CPS, bluesky and Hayakawa Laboratory (online), available from ⟨https://github.com/Bluesky-CPS⟩ (accessed 2015-09-15).
[9] Di Lauro, R., Lucarelli, F. and Montella, R.: SIaaS - Sensing Instrument as a Service Using Cloud Computing to Turn Physical Instrument into Ubiquitous Service, *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pp. 861–862 (online), DOI: 10.1109/ISPA.2012.135 (2012).
[10] JenHao Teng, C.-Y. T. and Chen, Y.-H.: Integration of Networked Embedded Systems into Power Equipment Remote Control and Monitoring, *TENCON 2004*, pp. 566–569 (2004).
[11] xively Logmeln: Solutions by Department, xively (online), available from ⟨https://xively.com/solutions-department/⟩ (accessed 2015-07-06).
[12] developerWorks: Eclipse Orion の紹介: クラウドの中でクラウドに対応，developerWorks of IBM（オンライン），入手先 ⟨http://www.ibm.com/developerworks/jp/cloud/library/cl-orionsummary/⟩（参照 2015-09-15）.