博士論文

IoTを指向したシステム開発環境の研究

Research on a development environment for Internet of Things

拓殖大学大学院　工学研究科

電子情報工学専攻

氏　　名：プラウィーン　アモーンタマウット

Praween Amontamavut

学籍番号：4D301

指導教員：早川　栄一　教授

拓殖大学大学院　工学研究科　電子情報工学専攻

2016年度　博士論文

拓殖大学大学院　工学研究科　電子情報工学専攻
博士(工学)
2016 年度　博士論文　要旨

TAKUSHOKU UNIVERSITY
Engineering
Graduate School of Engineering, Electronics and Information Sciences
Doctor of Philosophy (Engineering)
Ph.D. Thesis (2016.)

IoT を指向したシステム開発環境の研究

Research on a development environment for Internet of Things

4D301 Praween Amontamavut

本論文では，“Blue-Sky”というモノのインターネット化（IoT）を指向したシステム開発環境の研究と IoT 学習支援への適用について述べる．

本研究は IoT における産業改革 4.0 の時代に応いて，サービスを融合する今後の IoT の概念と懸命を，高等学校機関や大学の IoT 教育をはじめ，企業の導入教室を想定した IoT プログラミングの学習支援への適用の IoT 開発環境を開発する目的である．

本 IoT 学習支援教室は，40～50 人程度の小規模の教室を想定している．IoT プログラミングの学習は，学習者が Rest API を用いる IoT のセンシング・アクチュエーションの実行方法をはじめ，コマンドラインベーススクリプトや JavaScript で IoT の基本的なアプリケーションが簡潔的に短いコード行数で短時間に開発が可能になりつづ，この開発の有効性の検証や，そして，統一画面上にデバイスの監視機能に応じるデバイス名やデバイス IP などというデバイスの情報を取得することで，IoT のプロトタイピングが容易になる．さらに，通信遅延と OS レベルからアプリケーションまでの連動動作との懸命を意識させ，実際の IoT アプリケーションを開発する際にサービスを融合する IoT の概念と懸命を習得する可能な IoT 学習支援開発環境として提供する．そして，小規模 IoT 学習支援教室を立ち上げ，拓殖大学工学部 4 年生と実験をして，その

有効性を検証した.

このような小規模の IoT 支援教室においては，組込み機器が多種多様であるので，ノードを管理することが難しい．例えば，各ノードへのセンシング・アクチュエーションを実行するプログラミングインターフェースは統一ではない．これで，統合的なプログラミングインターフェースの提供が難しい問題があるので，各ノードへの接続性に圧迫するアクセスの不便性が引き起こすものである．そして，IoT はデータを活用することで，多量のデバイスから取得したサイズの大きいデータの管理が難しい問題がある.

そして，システムを開発する際に，OS レベルのプロセスをはじめ，ネットワークと連動するアプリケーションの動作の基に生じるオーバヘッドを検証する必要であるが，従来の組込みシステムは OS レベルのプロセスをネットワーク経由にシンプルな方法や可視化ツールはない問題がある．さらに，従来の IoT アプリケーションの開発方法においては，ネットワークに関するオーバヘッドを容易に検証する定常的な通信遅延のキャップチャーリングが可能なツールはないので，本研究の IoT の学習支援教室に適用することが難しい問題がある.

さらに，学生の使いやすさや，定常的に通信遅延というネットワークのオーバヘッドを容易に測定しながら短時間に IoT のプログラミングが可能とした効果を向上させるために，システムの開発しやすさやデバイスの接続性を向上させる利便性，そして，IoT の理解度を検証する必要なので，このような利便性の高い開発環境を備えることが難しい問題がある.

これに対して，本開発環境は，モノに搭載された組込みシステムの OS とそこに組み込まれた Linux のシステムインターフェースの実行が，ゲートウエイの Blue-Sky サーバに応じてまるで仮想なデバイスへの制御命令を補充して，Rest API を基づいたブラウザ上に複数の IoT のアプリケーションにおいて，そのモノに搭載するセンシング・アクチュエーションを実行させるモデルである．IoT は物理空間における実世界の環境計測や自動反応である多様なアプリケーションへサービスを提供することで，我々の生活を支えてくれる多種多様なデバイスを利用するものであることから，Linux カーネルの Ftrace をはじめ，OS のレベルのタスク，通信の遅延，アプリケーションの動作を考慮することである．そして，学習支援教室においては，デバイスの種類や数の増加によって指導者と学習者へ負荷を与える問題を引き起こすこととなり，使いやすさを阻害することを解決するように，IoT の学習支援教室へ有効性の検証を行う．そし

て，使いやすさと接続性を向上するため，異プロトコルをサッポートして，ROS の機能を本開発環境から使えるように拡張する．

まず，Linux カーネルの Ftrace をブラウザ上のアプリケーションが容易に実行ができるような低オーバヘッドプロセスロギング機構を構築する設備を開発した．本設備は，カーネルのコンテキストスイッチへの影響を低く抑えることと，IoT に関する通信負荷を圧迫させないように，低オーバヘッドの圧縮手法を設計して，圧縮機能を従来の Ftrace 機構に追加したことと，カーネル空間に動作するトレースドライバを実装した．本機構はユーザ空間を介せずにネットワーク経由で圧縮されたコンテキストスイッチのログをエクスポートする．これで，本機構は Linux のコンテキストスイッチログを取得する際に，カーネルレベルのタスクを影響せずに把握することが可能であることと IoT に圧迫されないようにしたので有効であると明らかになった．そして，本機構は Rest API に置いて機構を操作させるブラウザを用いたことで，本システムの IoT の使いやすさが向上された．

そして，IoT 向きウェブベース開発環境を開発した．本開発環境は，本研究において Linux ベースの組込み機器に置いて，本システムの Blue-Sky モジュールと連携操作する既に開発された Blue-Sky サーバのゲートウェイの Rest API を基づいて開発された開発環境である．これは，ブラウザ上に短時間に少ない行数のコードのコマンドラインベーススクリプトと JavaScript の IoT のプロトタイピングやプログラミングを行うことによって，定常的な通信遅延のキャップチャーリング機能やデバイスの接続状態の監視機能を統一的な画面内に操作するので，使いやすさと合わせて短いコードの IoT の基本的なアプリケーションを開発することが可能な開発環境であることで，ネットワークの異構成に応じる通信遅延と IoT アプリケーションとの連動動作とその懸命を把握することが可能な開発環境であった．

そして，本開発環境を基づいて，IoT プログラミング学習支援環境を立ち上げて実際の IoT の学習者と実験をした．本実験は，JavaScript の経験があるが，JavaScript で IoT の開発経験がない拓殖大学の工学部 4 年生を対象として，IoT の基本的なアプリケーションの開発課題を与えることと，アンケートにおいて被験者に対する使いやすさや IoT の理解度を取ることで，その有効性を測定した．

さらに，接続性を向上させるには，より多様なデバイスのアクセスが可能にするために異種プロトコルのサッポートにおいて，ROS が統一な環境で扱える

ように Blue-Sky を拡張した．これは，本開発環境のブラウザ上で統一された画面内にノードの接続状態や通信状態の監視と可視化が可能であり，ROS のノードと本システムのノードの両方に IoT プログラミングが可能となったため，使いやすさを合わせてより多種多様なデバイスの接続性を向上された．

本開発環境を開発したことで，著者は IoT の実本質が理解できた．IoT は単一なシステムではなく，組込みシステム，ネットワーク，アプリケーションやソフトウェアの開発，そしてプログラミング手法をはじめ，幅広い分野に応じてお互いに影響を与える．システムを融合するには，各システム自体のオーバヘッドの検討が不可欠であるが，異なるシステムを一つに融合する際にオーバヘッドのトレードオフは避けられない．例えば，ゲートウェイであるサーバを中継して，仮想化に見えるようなデバイスの管理機能があるとしても，異なるネットワーク構想において，往復遅延時間が生じたことで，センシング・アクチュエーションの実行 IoT のアプリケーションの動作が異なるオーバヘッドのトレードオフや，従来の IoT のアプリケーションの開発において，各デバイスに動作されているモジュールは，デバイスの仕様によって IoT のアプリケーションの動作が異なるオーバヘッドのトレードオフや，開発環境によってアプリケーションの開発時間やアプリケーションのコードのサイズが高い，言語によって連動するアプリケーションの動作が変動したりする遅延時間のオーバヘッドのトレードオフが挙げられる．

このようなオーバヘッドのトレードオフ問題を解決するには，可能な限りオーバヘッドのトレードオフの差分を抑えることである．融合されたシステムに応じて，各システム自体にオーバヘッドがあり，そのオーバヘッドを各システムと調整し，十分に抑える有効性を検証することによって，システム全体のオーバヘッドのトレードオフの差分を抑えることができると明らかになった．例えば，コードのサイズとネットワークの構想におけるトレードオフのオーバヘッドは，可能な限り短時間に IoT のアプリケーションを開発して，実際に動作させ，ネットワークの負荷が高い回線であれば，異なるネットワーク回線に移動させることで，ネットワークの負荷を十分に抑えられる．なので，IoT における産業改革 4.0 は，単一な IoT システムを改革する意味ではなく，IT の基本教育をはじめ，IT 業界の各分野の御協力に応じてサービスやシステムを融合する IoT は起動が可能なように改革することである．

目次

第１章　諸言

本章では，研究の背景，目的について述べる．

1.1　背景

モノのインターネット化(以下，IoT)やサイバフィジカルシステム(以下，CPS)の研究開発はチャレンジ課題が多様であり[1][2]，研究の歴史も長い研究分野である．

世界初計算機というENIAC[3][4]が西暦1946年に開発されて以来，CPSは西暦1973年に計算のタスクスケジュールで全てのジョブが各タスクのデッドラインに終わらせるリアルタイムコンピューテーションとして組込みシステムなどの研究が始めた[5][6][7]．西暦1990年には物理システムと計算とのインタラクションに注目するようになってきた．西暦2006年頃には研究者がリアルタイムシステム・ハイブリッジシステム・制御システムを“cyber-physical systems”に名づけた[5]．現在のCPSはサイバードメインと物理ドメイン同士にネットワークを介してタイトに統合するインタアクション可能な計算システムに注目するようになっている．

西暦1969年にはインターネットがARPANET[8]において開発され，世界中の計算機がネットワークで繋げるようになっている．Kevin Ashton氏 [9][10]はIoTに関してサプライチェーン管理のコンテキストを西暦1999年に述べ，ユビキタスコンピューティングの子係になり続いてきた．これがきっかけでセンサーネットワークの統合可能なアーキテクチャのトレンドになってきた[9]．

ユビキタスコンピューティング[11][9]は西暦2000年から大規模なインターネットユーザに対し仮想マシンやデータサイエンスなどによってサービスを作り出し続いているクラウドコンピューティングの一員である．

西暦2012年にはRoberto Di Lauro氏が“Sensing Instrument as a Service (SIaaS)”[12]を提案した．SIaaSは新サービスモデルであり，物理世界に属するモノをユビキタスサービスに付け加えられた．これによって，cloud-centric IoTが現れるものである．cloud-centric IoTは“Internet-centric”と“Things-centric”の視点がある[13]．さらに西暦2014年にドイツにおける「インダストリー4.0」[14]の影響で，IoTは産業の工場に応じるセンシング・アクチュエーションサービスに応じる研究開発が行われている[15][16]．

このような状況に対して，西暦2015年に世界中のIoTに関連する研究所に応じる全定義[17]により，CPS/IoTはアプリケーションレイヤーに応じてHTTPを

用いるインターネット経由システムが将来の IoT の規格になると期待されている．これで，日本は西暦 2015 年に IT 人材白書の調査を示す図 1[18]によると，IoT に重要である組込み技術者はウェブ技術・スマートデバイスに関する技術力が要求されてきた．本研究は IoT を支えるエンジニアの育成をはじめ，IoT を指向したシステム開発環境の研究を行うことである．

IT 人材白書 2015 [18]　P.4 図 1-1-2 より

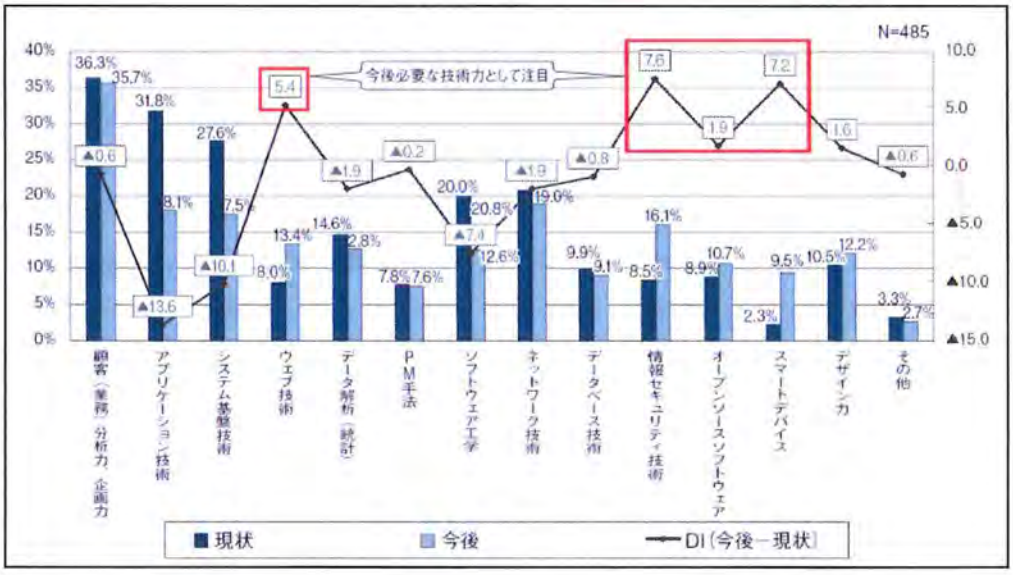


図 1　組込み技術者が 2015 年に必要と考える技術力

近年，モノのインターネット化(IoT) [19]の研究開発や教育が行われている．IoT はホームオートメーションやロボットや環境計測や農業をはじめ，ネットワークに繋ぐ組込み機器(以下，ED)で融合することにおいて，多種多量のデバイスにおけるセンシングデータを活用することで，アクチュエーションを行うサービスを，遠隔にインターネットで提供するシステムである．

実際に，多国間での共同開発や運用する IoT プロジェクト例は，米国のカリフォニアの EverySense 会社とインフォコム株式会社の共同開発を行っている IoT で漁業支援が挙げられる．そして，日本国内も多国間で共同教育プロジェクトである“Education Network for Practical Information Technologies”(EnPiT)[20]を実施されている．EnPiT は日本国内をはじめ，分野，地域を越えた実践的な情報教育プロジェクトであり，クラウドコンピューティング分野，セキュリティ分野，組込みシステム分野，ビジネスアプリケーション分野が実施されている．

しかし，IoT 分野に関する教育や開発は難しい．その理由は，多種多量の ED をネットワークで統一した環境が不足している点と，管理オーバヘッドを意識したシステム構成の調整を可能にするシステムや，それを扱える人材が不足している点にある．これらの問題に対して，小規模クラスでの教育利用をはじめ，低オーバヘッドで容易に統一可能な開発環境が必要となってきている．

IoT の教育においては次の問題がある．

(1) 多種多量の ED の統合的な管理において，各ノードの管理や各ノードへのプログラミングインターフェースと大量データの管理が難しい．

(2) OS レベルのプロセスをはじめ，ネットワークの構造の違いによるアプリケーションの動作の違いによって生じるオーバヘッドの監視や分析が難しい．

(3) クラスにおいて学生が容易に開発やプロトタイピングができる，使いやすさの利便性の高い開発環境が不足しているという問題がある．

1.2 目的

本研究は IoT の教育やプロトタイピングを対象として，多種類のデバイスを対応する組込み Linux をベースにして，低オーバヘッドで実行するシステムの開発や監視を容易にする環境に関する研究開発を行うことを目的である．

具体的には，次のとおりである．

(1) OS レベルのプロセスへの低オーバヘッドのロギング機構の開発

(2) IoT アプリケーションを対象とした通信の状況の監視，センシング・アクチュエーションの実行，テスト，デバッグを支援するプログラミング環境の開発

(3) 多量のデバイスから取得されたセンサデータやトレースデータに対応するスケラブルなクラウドストレージの構築

(4) クラスでの学生への教育を容易にする開発しやすいウェブベース開発環境の構築

(5) 多様なデバイスを扱うための異種通信プロトコルへの対応

1.3 本論文の構成

本論文は，本章を含めて 7 章から構成される．第 2 章では，IoT プラットフォームや IoT の定義を述べ，第 3 章では，圧縮を備えたトレース機構の開発を述べ，第 4 章は，ウェブブラウザベースのプログラミング環境を述べる．第 5 章に，IoT プログラミング支援環境とした BlueSky/Edu を述べて，第 6 章に，異種プロトコルサポートによる多様なデバイスアクセス環境の構築を述べた後に，第 7 章に成果と結論について述べる．

第２章 IoT プラットフォーム

本章では，現代の IoT のプラットフォームを述べ，世界中における IoT に関する定義と本研究のIoTの定義を述べる.

2.1 IoT 開発環境・プログラミング学習支援環境

2.1.1 APOLLO[21]

APOLLO プラットフォームは，ネットワーク，デバイスの管理，サービス，そして，アプリケーションの枠組に占める小説のように Internet of Things/Internet of Services（以下，IoT/IoS）アーキテクチャである．統一なプラットフォームに応じて，幅広いスマートデバイスの統合やセンシングとアクティングの両方にハンドルする可能なアーキテクチャである．これを繋ぐためには，既存な Telco cellular インフラと用いることで，完全に統合することが可能である．しかし，これは OS に近いアーキテクチャを容易に考慮が可能な設備が用意されていないことや，通信遅延のキャップチャリング機能を定常的に行う IoT プログラミング支援開発環境が備えていないことで，本来 IoT の本質の把握することが難しい.

2.1.2 Smart Cloud of Things: An Evolved IoT Platform for Telco Providers[22]

Smart Cloud of Things (SCoT)は，ETSI と合わせ，構築されたリッチなサービスの実行環境である．これを基づいて，デバイスとサービスをエンドユーザがアクセスしやすくする統合ポータルを提供する．SCoT は一般的なプラットフォームとして，IoT/IoS の統合シナリオを基づいた APOLLO の進化 IoT プラットフォームである．スマートな農業というスマートファームへの適用プラットフォームとして扱われ，ネットワークの負荷を圧迫しないようにセンシングデータの PayLoad のサイズの削減を主張している．しかし，このプラットフォームは，定常的な通信ネットワークのキャップチャ―リング機能が提供されていない．そして，OS に近いアーキテクチャを容易に考慮が可能な設備が用意されていない．さらに，プログラミング支援環境は提供されていない.

2.1.3 Eclipse IoT[23]

さらに，IoT の開発環境が多く登場している．例えば，IoT のオープンの規格の基に IoT のソリューションを大きな産業技術において実際にチャレンジする開発フレームワークである．実際に大規模な産業技術において，多種多数のデバイスを用いることで，多様な end-to-end ソリューションを用いるバックエンドシステムの基にしたセンシング・アクチュエーションを行う再利用可能な IoT アプリケーションの開発を支えるゲートウェイを提供するオープンソースプロジェクトである．これは，大手ベンダーである Amazon Web Services (AWS)，Microsoft Azure をはじめたクラウドインフラストラクチャ上に運用される仕組みである．しかし，Eclipse IoT はデータ通信に関するソリュションの指定が不明であり，通信遅延のキャップチャー機能が提供されていないものである．そして，OS に近いアーキテクチャを容易に考慮が可能な設備が用意されていない．

2.1.4 Microsoft Azure IoT Hub[24]

マイクロソフト株式会社は，既存の技術資産，デバイス，およびデータを活用する IoT からビジネス価値を創出することで，モバイルとクラウドに重点を置くエンタープライズを対象としてゼロから検討を行う必要がなく迅速に汎用的な IoT を用いることで，イノベーションの実現に迎える基盤なもので，各カストマーのビジネスに発展しやすくする Internet of Your Things を可能にする技術を提供する．IoT アプリケーションのコードのサイズが大きい．そして，OS に近いアーキテクチャを容易に考慮が可能な設備が用意されていない．

2.1.5 Scratch[25]

Scratch はプログラミング初心者を対象として，インタラクティブ的なプログラミング教育である．形のあるブロック的なものをプログラミングコンテキストの変わりに置き換えることで，プログラミングの学習がより分かりやすいものであり，ブラウザ上に稼動することで，初心者に向けるプログラミング学習支援に用いる効果が高いと思われている．しかし，Scratch は IoT プログラミング学習支援機能がない．そして，OS に近いアーキテクチャを容易に考慮が可能な設備が用意されていない．

2.2 IoTの定義

2.2.1 世界中におけるIoTに関する定義

2.2.1.1 ETSI

ETSI[17]は，欧国において，インターネット技術，ブロードキャスト，ラジオー，モバイルを含んだ“Information Communication Technology” (ICT)のための適用可能な規格を生産する．最初のETSIはIoTを言及さていなかったが，2010年から，類似な概念である“Machine-to-Machine” (M2M)の通信を定義した．直接に人間を介入する必要がない二つ以上のエンティティの間の通信であり，通信のプロセスと自動化の決定をするつもりである．

2.2.1.2 NIST

米国の“The National Institute of Standards and Technology” (NIST)[17]は2014年にIoTをCPS/IoT技術の基に説明した．スマートアメリカグロバル町のチャレンジするためのIoTの説明である．CPSは時にIoTを指すことが可能であり，デバイスをよりスマートに繋ぎ，健康管理や運送システムを新たな方法を挙げる．CPS/IoT技術を用いることで，より効果的な日常生活を向上させるスマートシティ/コミュニティを建てる．

2.2.1.3 IP for Things

IoTの今後に，毎日にIoTのオブジェクトを既に利用することになったら，RFIDなどの特別な通信プロトコルを利用する必要がなくなり，IPアドレスを既に割当てる一般的なインターネットのノードを利用することも避けられない．これで，IPv4がより多くノード数を対応する可能にするためにIPv6を作り出された．現在に，Wirelessネットワーク専用のローエネルギーを対象としたIPv6である6LoWPANが登場された[17]．

2.2.1.4 IoT-A Project

IoT-A Project[17]は，欧国のプロジェクトであり，はじめにIoTのためのアーキテクチャ参照モデルを開発した．2013年に“Enabling Things to Talk” [26]を基づいてIoTを説明した．プロトコルレベルのインタラクションと管理やセキュリティを払ってIoTのアーキテクチャ参照モデルを開発している．このモデルは，ドメインモデル，情報モデル，関数モデルがある．

2.2.1.5 CERPIIoT Project

CERPIIoT Project[17]は，2010 年に欧国のフレームワークプログラム 7 のプロジェクトである．このプロジェクトは次のように IoT を定義した．

「IoT は　今後のインターネットの部分には，物理と仮想の“Things”は，アイデンティティを持つ相互運用可能な通信プロトコル，標準のある自己設定能力，物理属性，個性的な仮想，そしてインテリジェントインターフェースの使用を含む動的なグロバルネットワークインフラにおいて，ネットワークの情報をシームレスに統合することを定義すべきである．そして，“Things”の IoT はビジネスのアクティブ要素として，人間を介入するか否かは関係なく，トリッガーアクションと“real/physical world”のイベントへの自動的反応に応じて，情報と環境計測に関する“sensed”の情報とした環境におけるそれ自体のデータの交換を行う通信が可能であり，これらのインタアクトが可能である社会プロセスとなるのが期待されている．インターネットをかけるこれらの“Smart Things”とのインタラクションサービス設備という形であるインターフェースは，個人情報侵害問題やこれらを含めた全てのキュウリアクセス情報についてセキュアに通信するものである．」

2.2.2 本研究の IoT の定義

本研究の IoT 定義は，図 2 に示す．本 IoT はウェブを介するインターネットが届ける場所におけるモノである組込みシステムを組合せ，サイバー空間に用いる OS レベルからアプリケーションまでのタスク，ネットワークの通信までの行動を考慮する可能にさせる，センシング・アクチュエーションを行うことで幅広い分野に対応させ，使いやすくするサービスを，人間の快適を支えるように教育の根本から業界へ展開するように提供する融合システムである．

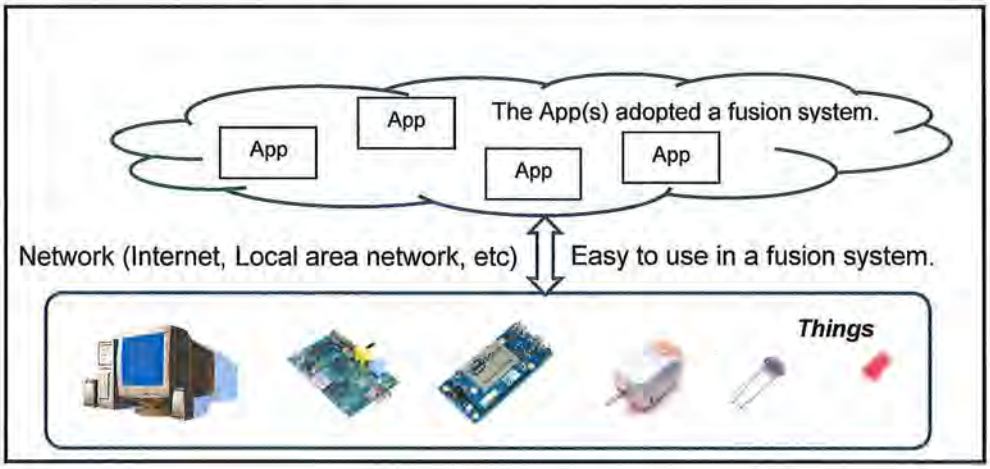


図 2 本研究による IoT の定義

2.3 本研究に応じる IoT の学習支援および教育の問題

(1) **IoT 支援教室への適用が難しい問題**：

IoT 支援教室は，組込み機器が多種多様であるので，ノードを管理することが難しい．例えば，各ノードへのセンシング・アクチュエーションを実行するプログラミングインターフェースは統一ではない．これで，統合的なプログラミングインターフェースの提供が難しい問題があるので，各ノードへの接続性に圧迫するアクセスの不便性が引き起こすものである．そして，IoT はデータを活用することで，多量のデバイスから取得したサイズの大きいデータの管理が難しい問題がある．

(2) **サービスを融合する IoT が難しい問題**：

システムを開発する際に，OS レベルのプロセスをはじめ，ネットワークと連動するアプリケーションの動作の基に生じるオーバヘッドを検証する必要であるが，従来の組込みシステムは OS レベルのプロセスをネットワーク経由にシンプルな可視化ツールや方法はない問題がある．さ

らに，従来の IoT アプリケーションの開発方法においては，ネットワークに関するオーバヘッドを容易に検証する定常的な通信遅延のキャップチャーリングが可能なツールはないので，本研究の IoT の学習支援教室に適用することが難しい問題がある．

(3) **学習者・学生等のエンドユーザの考慮が難しい問題**：
学生の使いやすさや，定常的に通信遅延というネットワークのオーバヘッドを容易に測定しながら短時間に IoT のプログラミングが可能とした効果を向上させるために，システムの開発しやすさやデバイスの接続性を向上させる利便性，そして，IoT の理解度を検証する必要であるので，このような利便性の高い開発環境を備えることが難しい問題がある．

これに対して，本開発環境は，モノに搭載された組込みシステムの OS とそこに組み込まれた Linux のシステムインターフェースの実行が，ゲートウエイの Blue-Sky サーバに応じてまるで仮想なデバイスへの制御命令を補充して，Rest API を基づいたブラウザ上に複数の IoT のアプリケーションにおいて，そのモノに搭載するセンシング・アクチュエーションを実行させるモデルである．IoT は物理空間における実世界の環境計測や自動反応である多様なアプリケーションへサービスを提供することで，我々の生活を支えてくれる多種多様なデバイスを利用するものであることから，Linux カーネルの Ftrace をはじめ，OS のレベルのタスク，通信の遅延，アプリケーションの動作を考慮することである．そして，学習支援教室においては，デバイスの種類や数の増加によって指導者と学習者へ負荷を与える問題を引き起こすこととなり，使いやすさを阻害することを解決するように，IoT の学習支援教室へ有効性の検証を行う．そして，使いやすさと接続性を向上するため，異プロトコルをサッポートして，ROS の機能を本開発環境から使えるように拡張する．

第３章　圧縮を備えたトレース機構の開発

本章では,圧縮を備えたトレース機構の開発について述べる.

3.1　まえがき

システムの高機能化やネットワークに伴い，ネットワークからデバイス操作やデータ転送を行うために，Linux をベースに組込みシステムを構築することが増加している．Linux が持つ移植性の高さやネットワークの安定性などから，Android[27]による携帯電話・スマートフォンや，TV，ホームサーバからロボットまで多くのデバイスで用いられつつある．また，Raspberry Pi[28]などのネットワーク接続可能で小型，安価な機器も数多く登場していて，これらを利用することで安価なシステム構築が容易になりつつある．また Internet of Things(IoT)[29][30]でも，オープンソースであることから Raspberry Pi などを試作に用いる例[31]が増加している．このような組込みシステムでは，デバイス制御や GUI 部分でプロセスやスレッドを用いて実装することが多い．システムの応答性の向上や，制御プログラムの動作を把握するには，プロセスやスレッドに関して実行時のプロセス情報を取得することが重要になる．

Linux では，複数のプロセスやスレッドのトレースは，デバッグファイルシステムという疑似ファイルを介してカーネル空間にあるトレースデータを読み書きすることで，プロセスのコンテキストスイッチやイベントのプローブ，関数単位での実行詳細を，トレースデータとして取得することができる．これらのデータは人間が読みやすいように，ASCII 文字列で取得することができる．ユーザ空間では，これらのトレースデータを取得し，格納することで，カーネルの起動から任意の時刻までのシステム動作の監視が可能になる．これまで，トレースデータを効率よく扱い，分かりやすく表示するためのツール群が開発されてきた[32][33][34][35][36].

しかし，上記のトレースメカニズムを，組込みシステムに適用することには問題がある．第１に，利用者の行動やイベントによって，保存すべきトレースデータのサイズが変化し予測が難しい点である．組込みシステムでは長期にわたって稼働することが多く，トレースデータのためのメモリを事前に多く確保しておく必要がある．しかし，一般的に組込みシステムに搭載されているメモリは少なく，大きくなりがちなトレースデータをシステム内に保持することが難しい.

第２に，コンテキストスイッチ情報を取得，保存するときに，その機構自体がアプリケーションプロセスの動作に影響を与えるという問題がある．例えば，起動するプロセスの数が多い場合，一般的に生成されるトレースデータのサイズは増加する．カーネル内部でトレースデータを保持するメモリサイズには制約があることから，データ消失を防ぐにはユーザ空間への頻繁なデータのコピーが必要になる．このコピー自体がコンテキストスイッチを生じさせて，測定対象となるアプリケーションの動作に影響を与えてしまう．コンテキストスイッチ数やトレースデータ保持のメモリサイズを低減させようとしてコピーの回数を減らした場合，カーネル内部でトレースデータが消失してしまう可能性がある．これは，Linux カーネルではコンテキストスイッチ情報の取得にはリングバッファを用いているので，リングバッファからコピーできなかった過去のトレースデータは上書きされてしまうからである．

これらの問題を解決するために，本論文では組込み Linux をターゲットとして，プロセスやスレッドのプロセス生成およびコンテキストスイッチ時のトレースデータのサイズ低減手法を提案する．組込みシステム向けの低オーバヘッドなコンテキストスイッチに関するトレースデータ圧縮機能を実装し，その有効性を評価した．さらに，プロセススケジューリングへの影響を抑えつつ，トレースデータ保存のメモリ使用量を減らすために，ネットワーク経由でトレースデータを取得可能な分離型 Linux プロセストレース機構を実現した．この二つの機構を用いることで，ユーザ空間を経由したリモートマシンへのトレースデータコピーと比較して，従来の Ftrace を用いてローカルディスクへ保存した場合とほぼ同等のオーバヘッドで，リモートマシンへトレースデータを転送することが可能になった．また，アプリケーションの動作への影響や内部のメモリへのスイッチへの影響を抑えたことから，長期にわたって稼働するネットワーク接続された組込みシステムへ適用することが可能になった．

3.2 従来のトレース機構

本節は，従来の Linux のトレース機構について述べる．

3.2.1 Ftrace:Linux のトレース機構

本システムは，Linux カーネル 2.6.29 に含まれる軽量なカーネルトレースである Ftrace[37][38]をベースとして開発した．Ftrace は Linux カーネル内部付属のトレーサであり，2.6.27 から追加された．ユーザ空間のトレースツールは，この機構を通してカーネルの様々な動作結果を容易に取得できる．Ftrace はカーネルバージョンによって提供される機能が異なるが，本システムで対象とする Ftrace の機能は次のとおりである．

- **function**: すべてのカーネル関数をトレースする．
- **function_graph**: すべてのプローブ可能なカーネルおよびユーザプロセス内に実行する関数をトレースし，呼出しグラフ化した文字列を出力する．
- **sched_switch**: カーネルのスケジューリングにより発生したプロセス状態の遷移をトレースする．

本システムでは，プロセス切替えに関する情報を出力する sched_switch に注目した．本章では，この部分の機構の概要について述べる．

3.2.2 トレース機構の動作

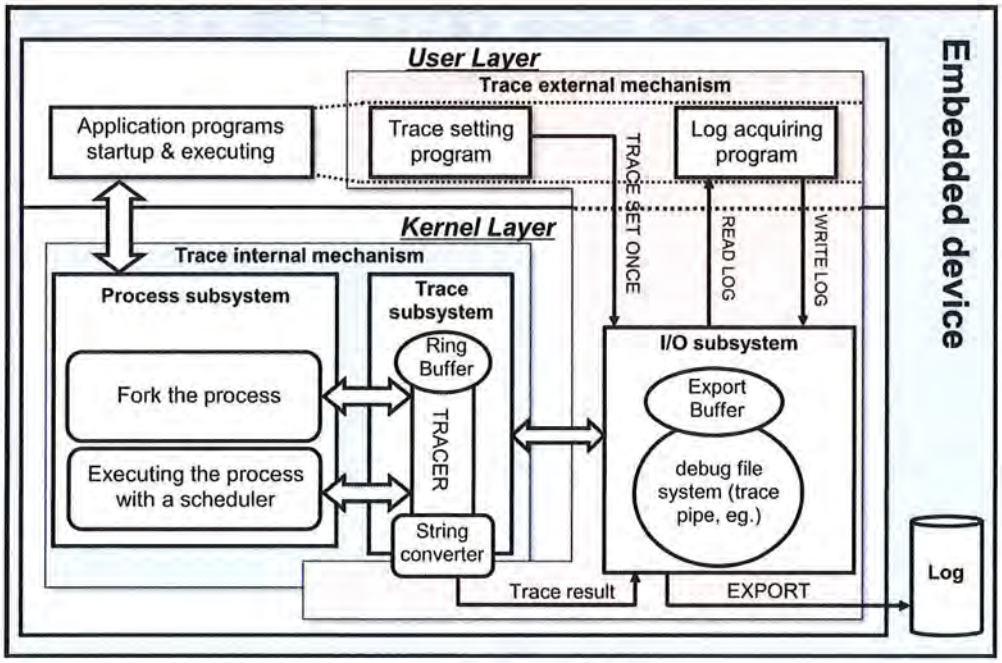


図 3　従来のトレース機構

従来のトレース機構を図 3 に示す．トレース機構は，トレース内部機構(Trace internal mechanism)とトレース外部機構(Trace external mechanism)の二つから構成されている．

トレース内部機構は，プロセススケジューリングを管理するプロセス管理サブシステム(Process subsystem)と関連して，コンテキストスイッチ時のトレース情報を生成する機能を提供する．これをトレースサブシステム(Trace subsystem)と呼ぶ．トレースサブシステムは，トレース外部機構のトレース設定プログラム(Trace setting program)で設定されたパラメータに従ってトレースデータを取得する．例えば，デバッグファイルシステムの trace_enable を 1 に設定すると，リングバッファは初期化されてトレースデータを収集するようになる．プロセスに関するトレースデータを取得する時にプロセスをブロックすることはなく，設定したリングバッファのサイズを超えた場合は過去のデータは上書きされる．

トレース外部機構は，カーネル空間内にあるトレース内部機構から，ユーザ空間上にあるユーザプロセス(Application programs)に，I/O サブシステム(I/O subsystem)を通じてトレースデータをエクスポートする．このとき，トレース内部機構は，トレースポイントフラグを付けたデータに対して，リングバッファからエクスポートバッファにトレースデータを収集する．これを文字列化機構(String Converter)によって ASCII 文字列に変換した後にデバッグファイルシステムの tracepipe を通じてユーザ空間にあるトレースデータ取得プログラム(Log acquiring program)にエクスポートする．

3.2.3 コンテキストスイッチのトレースデータ

図 4 に Ftrace が生成するコンテキストスイッチのトレースデータ（以下，Ftrace 形式）の実例を示す．このデータは，プロセス生成やプロセス状態の切換え時のプロセス情報を Ftrace を用いてトレースした時の情報である．

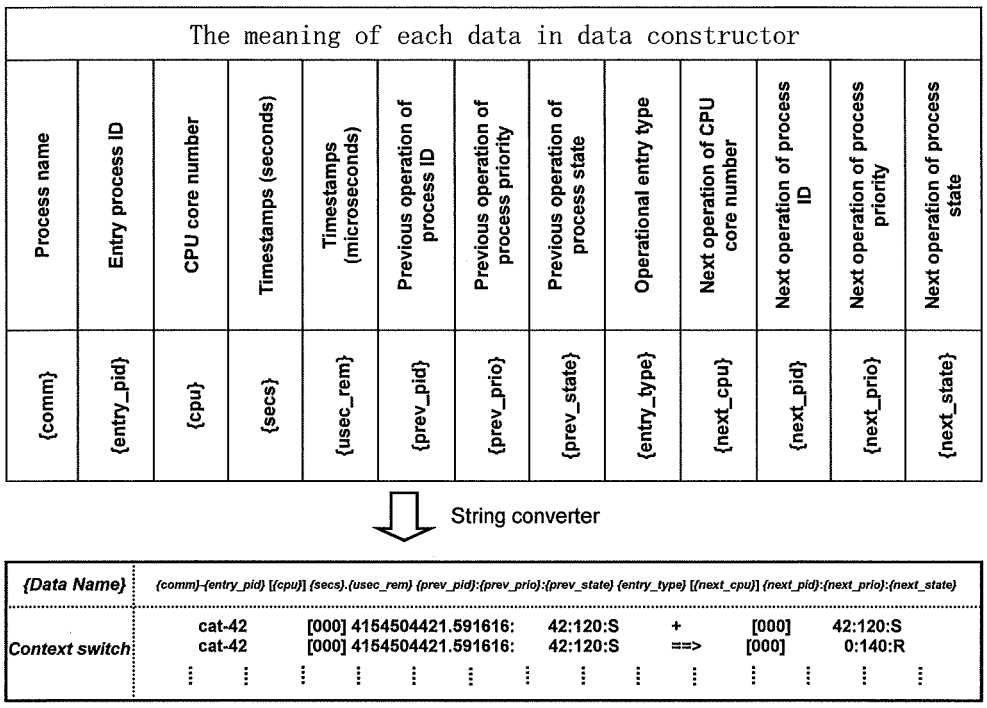


図 4 コンテキストスイッチトレースデータの例

Ftrace 形式は，リングバッファに格納されるバイナリデータを，プログラムで扱いやすくするために，図 4 のように ASCII 文字列化しエクスポートバッファに収集した後，ユーザ空間で動作するプロセスにエクスポートする形になっている．

3.2.4 従来のトレース機構の問題点

従来の Ftrace 形式を組込みシステムで用いる場合の問題点は次の 3 点である.

(1) リングバッファがトレースデータ生成とユーザ空間へのコピーとの速度差を吸収できず, データが消失する可能性がある. Linux カーネルでは, エクスポートバッファのサイズは 1 ページに限定されている. コンテキストスイッチのトレースデータは ASCII 文字列に変換し, ユーザ空間へエクスポートされるので, データのエクスポートが間に合わない可能性がある. この場合, リングバッファ上のデータは上書きされ, データが消失してしまう.

(2) トレースデータのユーザ空間へのコピー時に発生したコンテキストスイッチが, プロセススケジューリングに影響を与える可能性がある. ユーザプロセスは, トレースデータをデバッグファイルシステム経由でユーザ空間へコピーする. データコピー自体はスケジューリングの対象になることから, トレースデータの取得や保存がスケジューリングそのものに影響を与えてしまう. スケジューリング自体への影響を避けるには, コピーするプロセスの優先度を下げるなどの方法があるが, この場合データがカーネル内部に保持されることから, (1)で述べたデータ消失の危険が生じる.

(3) トレースデータの保存容量に限界がある. 多くの組込みシステムは機器内のフラッシュ ROM の容量が小さいことから, 機器内に大量のデータを保存することはできない. 例えば, カーネル 2.6.29 のデフォルト状態の Android Froyo でブラウザを起動させ, タッチパネルをタッチし続けると, 毎秒 135KBytes 程度のトレースデータを生成する. 1 日あたり約 11.7GB の大きさとなってしまうことから, 長時間稼働するシステムのトレースデータを組込み機器の内部に保持するのは難しい.

3.3 組込みLinux向け分離型トレース機構

本節は，本研究の組込みLinux向け分離型トレース機構について述べる．

3.3.1 設計方針

前節で述べた問題を解決するために，ネットワーク接続された組込みLinuxを対象とした，分離型トレース機構を設計した．従来のFtraceに対して，プロセスの状態遷移に関するトレースデータの取得オーバヘッドを低減する改良を行った．また，組込みデバイス側への保存サイズを減らすために，トレースデータを保存する部分を“Log Acquiring Server”として組込みシステムの外部に分離した．本機構の設計方針は次のとおりである．

(1) **カーネル内部のトレースデータを圧縮する：** カーネル内部で多くのデータを保持可能にするために，カーネル内部でのトレースデータに対してデータ構造に依存した圧縮を行い，より多くのデータを保持できるようにする．

(2) **トレースデータを生成する計算機とデータを保存する計算機を分離する：** 組込みシステムの主記憶および二次記憶のサイズは制限されていることから，トレースデータをすべて組込み機器内部で保持するのは難しい．また，ネットワーク接続された組込みシステム環境では，複数の組込みシステムからトレースデータを収集する機構が必要になる．そこで，トレース対象の計算機にデータを保存せず，ネットワークを介して外部の計算機へのコピーを行う．

(3) **ユーザ空間へのデータコピーを極力さける：** スケジューリングへの影響を避けるために，ユーザ空間へのトレースデータのコピーは行わず，カーネル内部だけで処理する．

3.3.2 トレース機構の設計

本システムの全体構成を図 5 に示す．本システムはロギング環境（Logging Environment，以下 LE）ログの監視環境(Log Monitoring Environment，以下 LME)から構成される．

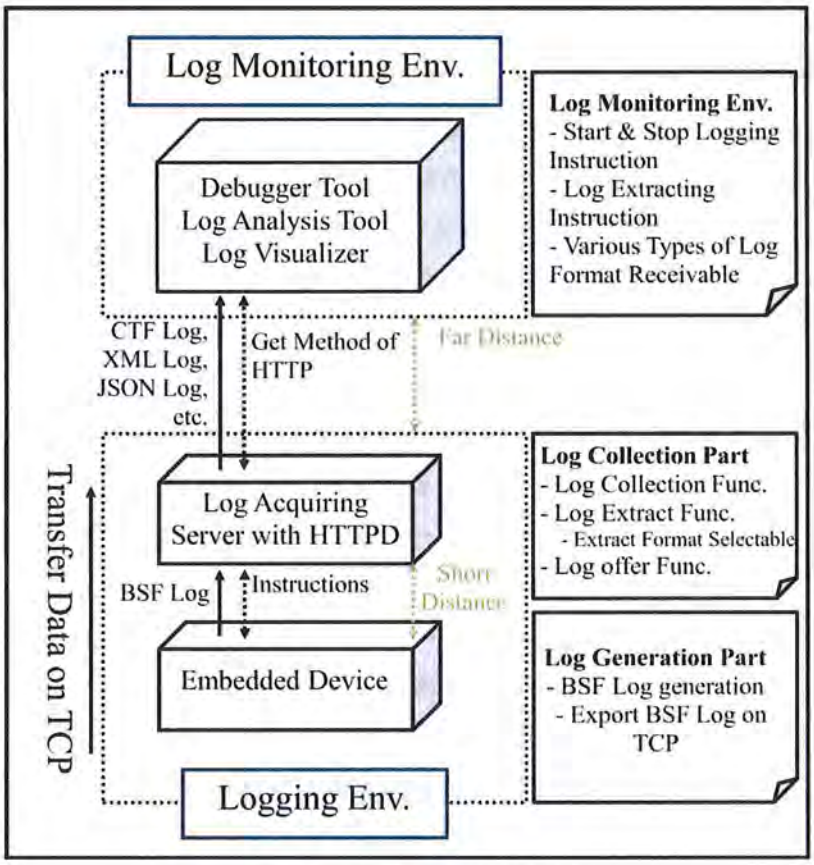


図 5　システムの全体設計

3.3.2.1　LE の設計

LE では，トレースデータ生成側(Log Generation Part)の組込み機器(以下，生成側)とトレースデータ取得側(Log Collection Part)のログ収集計算機（以下，取得側）の二つから構成する．これによって，トレースデータの保存は生成側の組込み機器から分離されるので，連続したサイズの大きいトレースデータの保存にかかるオーバヘッドを減らすことができる．

本システムでは従来のカーネルスケジューラとそれに関連したコード，およびトレースデータを生成しエクスポートする機構は変更せず，そのまま利用している．これは，トレース内部機構がプロセス管理部との関連性が高く，この

部分の変更はカーネル全体に影響を与えるからである．カーネルスケジューラとトレース内部機構の構造体を変更することは，カーネルコードの変更に対する追従性を悪化させることから，本システムでは採用していない．

本トレース機構の構成を図 6 に示す．図 6 の生成側に，従来の機構に圧縮機能(Compressor)を持ったトレースドライバ(Trace Driver)を追加する．トレースドライバはカーネル空間(Kernel Layer)で動作するサブシステムであり，次の二つの役割を備えている．

- **トレース制御部(Trace Controller)** は，トレースサブシステム(Trace Subsystem)の確認機構とデータ圧縮機能を備える．前者はシステムの起動時にトレースの有無を確認する．後者はトレーサが有効の場合にだけ起動する．圧縮機能の設計は次の節で述べる．
- **デバイス制御部(Device Controller)** は，トレースデータ取得サーバプログラム(Log acquiring server program)にデータを転送する．トレースデータをユーザ空間へコピーせず，直接トレースデータ取得サーバに TCP 通信によってエクスポートする．TCP 通信を用いることで，トレースデータのエクスポートに対して通信メディアを自由に選択することができる．また，帯域の狭い通信メディアの場合においても，組込みシステム内部のリングバッファに保持するデータおよびネットワークで転送するデータを圧縮することで，従来の方法でのデータ転送と比較してシステムのメモリ消費を抑えつつ，データを送ることが可能になる．

  エクスポートの処理は従来のトレースと同様に 100 m 秒ごとのタイマ割込みで行われる．そして，“Logging Instructions”は組込み機器に対しトレースデータの取得の開始停止の命令と定義する．本命令は，1) トレースの開始，2) トレースの停止の二種類の命令がある．これらの命令は，トレースデータ取得サーバからトレース制御部を送ることで，外部からトレースの開始，停止の指定が可能になる．

トレースデータ取得サーバは，トレースの開始命令・トレースの停止命令を組込システム側に渡すことによって，組込みシステム側からのすべてのトレースデータを取得して保存し，データ解凍機能により圧縮されたトレースデータを解凍する．取得側への転送は図 3 図 3 のようにユーザ空間へのコピーをせず，上記のトレースドライバによってカーネル内部から直接行う．

そして，LME に対して，HTTP の GET メゾッドをベースにした ftrace を実行する Rest API を提供する．API は次のとおりである．

- **トレースの開始 API**：LE のトレースの開始命令の実行 API である．
- **トレースの停止 API**：LE のトレースの停止命令の実行 API である．
- **トレースデータの解凍 API**：圧縮されたトレースデータを JSON, XML 形式に解凍 API である．
  - トレースする日日毎に分解する機能の指定が可能である．
  - トレースデータの行数毎に分解する機能の指定が可能である．
- **トレースデータの取得 API**：解凍された各種類のトレースデータの取得 API である．
  - トレースする日日毎に取得する機能の指定が可能である．
  - トレースデータの行数毎に取得する機能の指定が可能である．
- **トレースデータの一覧を表示する API**：トレースデータ取得サーバに保存されたトレースデータを一覧に表示する．

3.3.2.2 LME の設計

LME はブラウザにおけるトレースデータの監視ツールとトレースデータの可視化ツールを設計する．トレースデータ取得サーバが提供した API を用いることで設計する．

- **トレースデータの監視ツール**：
  トレースデータ取得サーバの全ての API を実行するツールである．本ツールを用いることによって，LE の組込み機器側を触れずにブラウザのボタンにクリックするだけでカーネルレベルの ftrace の開始・停止，圧縮されたトレースデータの解凍，解凍されたトレースデータの取得が可能になる．通常にトレースデータを監視する際には複数回に取得する場合があるので，組込み機器側でデータアクセスのコンテキストが生じるオーバヘッドがあるが，本ツールを利用することで，組込み機器の直接にトレースデータをアクセスすることなくサーバに保存されたトレースデータを取得するなの

で，トレース停止後に起動中の組込み機器へのデータアクセスによる影響を抑える．

- **トレースデータの可視化ツール：**
  トレースデータの可視化ツールは 2011 年から本研究室の修士の学生と共同開発となってきた Android OS のプロセスを可視化する環境[39]をベースに，トレースデータ取得サーバの API の，1）トレースの開始 API，2）トレースの停止 API，3）トレースデータを JSON 形式に解凍する API，4）JSON 形式のトレースデータを取得する API の四つの API を利用する．これで，プロセスを可視化する際に，組込み機器の直接にトレースデータを取得することなく，データアクセスによる影響を抑えるとともに，サイズの小さい JSON 形式を利用することで，データの通信容量を低減される．

3.3.3 LE における圧縮機能の設計

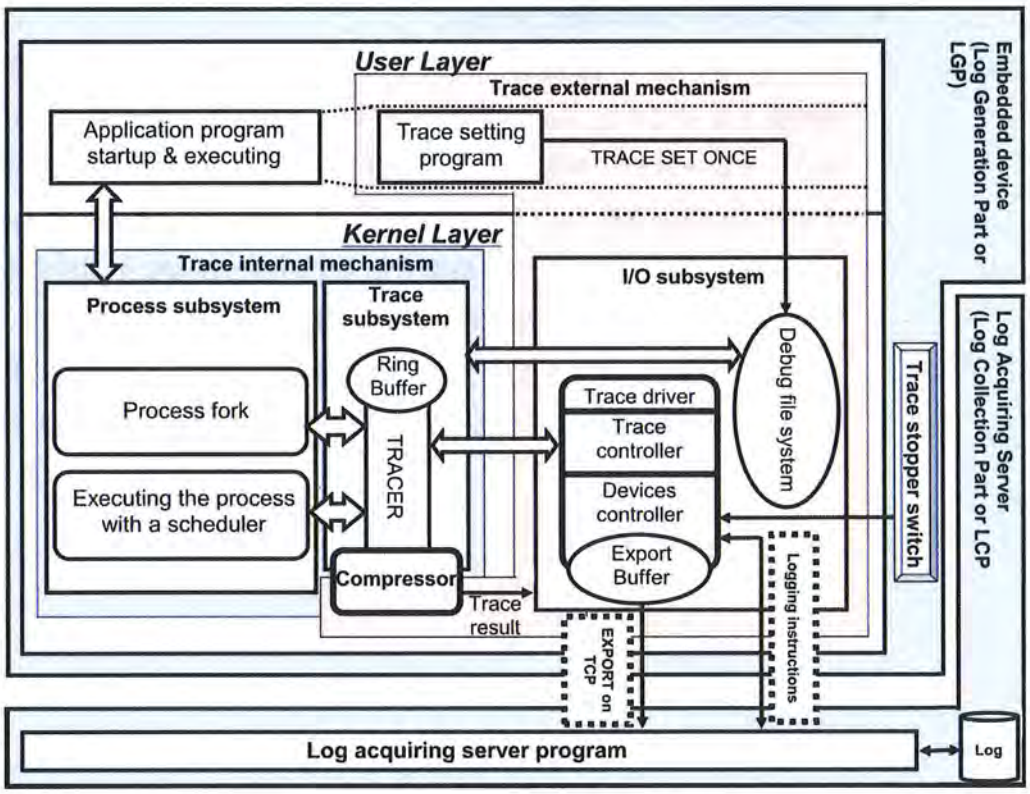


図 6　本提案のトレース機構の全体構成

圧縮機能(Compressor)は，トレースドライバのトレース制御の機能として，従来の文字列化機能に追加することで実現する．プロセスへの影響を減らし，より長い時間トレースデータ保存するには，従来の文字列化機能より圧縮率がよく，低オーバヘッドな方法を検討する必要がある．従来の文字列化機能でのデータ構造を表 1 に示す．これに対して，本圧縮機能でのデータ構造を表 2 に示す．従来の Ftrace 形式では，リングバッファから出力されたデータを元データよりサイズが大きい文字列に変換して，サイズが小さいエクスポートバッファに格納している．これに対して，本圧縮機能では文字列化機能と同様にデータをリングバッファからエクスポートバッファに格納するが，データは特徴に応じて文字列化せず，よりデータサイズの小さいデータ表現を用いてエクスポートバッファに格納するようにする．

表 1　文字列化機能でバッファに格納する領域の比較

Data name	Data size in ring buffer (Bytes)	Data size in export buffer (Bytes)
comm	16	16
entry_pid	4	7
cpu	4	5
secs	8	2～11
usec_rem	8	7
prev_pid	4	7
prev_prio	1	4
prev_state	1	3
entry_type	1	3
next_cpu	4	5
next_pid	4	7
next_prio	1	4
next_state	1	3
合計	**57 Bytes**	**Min74～Max83 Bytes**

このデータ構造の特徴を活かした形で圧縮するために，従来の Ftrace 形式の特徴を分析したところ，特徴は次の 3 点であることが明らかになった．

(A)　データの表現がメモリの確保領域より広く，同じデータの繰返し頻度が高いデータ

(B)　データの表現がメモリの確保領域より狭いデータ

(C)　データの表現がメモリの確保領域において十分なデータ

(A)に属するデータは，プロセス名や時間などを表現する“comm, cpu, secs, usec_rem, next_cpu”である．これらは可変長文字列なので，文字列表現である文字列部(String section)とこれにおけるデータ長を表すバイナリ部(Binary section)から構成する．特に，バイナリ部は文字列部の最大文字数と相当するビットのデータ表現の領域を確保するようにする．そして，事前に出力したデータと一致したデータならば文字列は出力せず，文字数を表すバイナリ部は 0 のデータを書き込む．comm を例として，データが“er.ServerThread”で 15 文字の文字列長の場合を考える．

表 2　データの特徴による分類と圧縮

Data name	Data charecteristic	Data size in ring buffer (Bytes)	Data size in export buffer	
			Binary section (bit)	String section (Bytes)
comm	(A)	16	5	0～16
entry_pid	(C)	4	32	null
cpu	(A)	4	2	0～3
secs	(A)	8	5	0～10
usec_rem	(A)	8	5	0～6
prev_pid	(C)	4	32	null
prev_prio	(C)	1	8	null
prev_state	(B)	1	3	null
entry_type	(B)	1	2	null
next_cpu	(A)	4	2	0～3
next_pid	(C)	4	32	null
next_prio	(C)	1	1	null
next_state	(B)	1	3	null
Total		57 Bytes	17Bytes	min0～max38Bytes
			Min17～Max56 Bytes	

ここで，表 1 のエクスポートバッファを参照すると，comm はリングバッファとエクスポートバッファの格納領域が 16 バイトである．これで，表 2 のように文字列部の確保領域の最小値・最大値を 0～16 バイトにするように定義することができる． バイナリ部は 16 バイトを表す 5 ビットのメモリ領域を用意にすればよい．このとき，文字列部に文字列表現の“er. ServerThread”を書き込み，バイナリ部に 15 を書き込む．これは一つ前の出力データと見なして，次の出力データを考える．出力データが一つ前の出力データと同じ“er. ServerThread”となる場合は，comm の文字列部に何も書き込まないようにする．これで，comm の文字列部は 0 バイトになるため，バイナリ部には 0 を書き込む．

(A)のデータは繰返し頻度が高いので，これによって圧縮率を向上させることができる． しかし，文字コードに変換や文字列の長さを計算するオーバ

ヘッドが存在する．このオーバヘッドを吸収するために(B)と(C)に属するデータを次に考える．

(B)に属するデータは，プロセスの状態を表す“prev_state, entry_type, next_state”である．これらのデータの特徴は，いずれも少ない固定のビット数で表すことができる点である．そこで，ビットフィールドによりビット表現を縮め，バイナリデータをバイナリ部にだけに書き込むようにする．例えば，prev_state のデータを考える場合，これは前のプロセス状態を表すデータであり，8 つの状態だけで表すことができる．表 1 のリングバッファの格納領域を参照すると，文字化した 1 バイトデータを固定 3 バイトの文字列のデータにする必要はないので，表 2 のように 3 ビットのバイナリデータにすれば十分である．(B)のデータは文字列で表現する必要はないことから，データのサイズは小さくなり， 転送オーバヘッドが(A)より低くなる．

(C)に属するデータは，プロセス ID や優先度などを表す“entry_pid, prev_pid, prev_prio, next_pid, next_prio”である．これらのデータはシステムの起動中に生成する動的なデータなので，データの確保領域は予測できない．さらに，表 1 のリングバッファ格納領域を参照すると，リングバッファに格納する領域はエクスポートバッファよりも少ないので，格納領域を縮める必要がないデータである．このことから表 2 では，そのままバイナリコードでバイナリ部だけにデータを書き込むようにする．データ自体を加工せずに書き込むことで，オーバヘッドを低くすることができ，(A)のオーバヘッドを吸収することもできる．

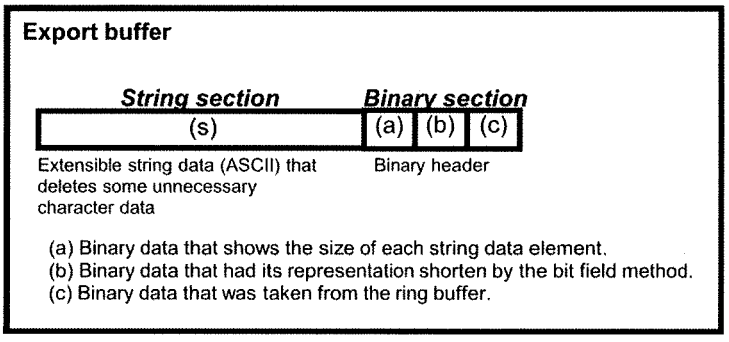


図 7　圧縮機能によりエクスポートバッファに格納する 1 行のトレースデータ

これらの特徴に基づいて，データの特徴で分類し，図 7 のようにエクスポートバッファに書き込む．可変長の文字列としては“comm，cpu，secs，usec_rem，next_cpu”の文字列部(s)をこの順番に書き込む．

バイナリ部は，バイナリデータの情報自体と文字列部の各要素の文字列長のデータを持つバイナリヘッダも格納する．バイナリヘッダは次の三つのデータ(a)～(c)を書き込む．(a)は“comm，cpu，secs，usec_rem，next_cpu”のデータ，(b)は“entry_type，prev_state，next_state”のデータ，(c)は“entry_pid，prev_pid，next_pid，prev_prio，next_prio”のデータを，それぞれこの順序で書き込む．

また，バイナリヘッダは後方に配置する．この理由は，(a)の可変長文字列の各要素の文字列長をまとめるのに発生するオーバヘッドを減らすためである．

このフォーマットを the mixed Binary and String Format（以下，BSF）と定義する．これは文字コードとバイナリーコードを連続した形で構成しているからである．このフォーマットを使用するトレースデータを BSF Log と呼ぶ．

3.3.4 データの解凍方法

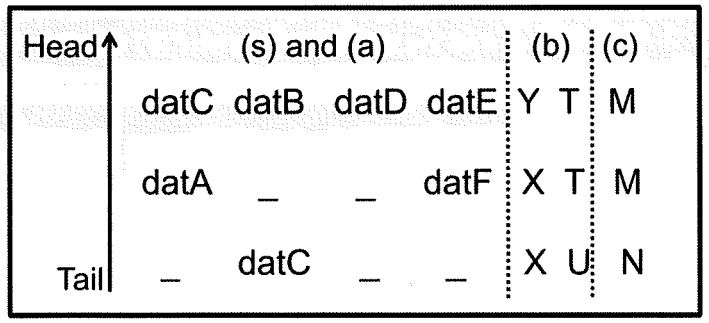


図 8　中間ログのデータ構造

末尾から固定サイズのヘッダバイナリを読み込み，図 7 における(a)～(c)の各データを図 8 のような中間ログとして解析する．(a)のデータ部分は各要素の文字列の長さをバイナリ部から読み込み，各要素の文字列を分離する．文字列の長さが 0 の場合は，文字列部にデータがエクスポートされていない．これによって要素の分離が困難になることを回避するため，マーカを“_”の 1 バイトに格納する．そうではない場合は，その要素の長さだけを各要素の文字列に格納する．(b)と(c)のデータ部分はそのまま文字列化し，格納する．

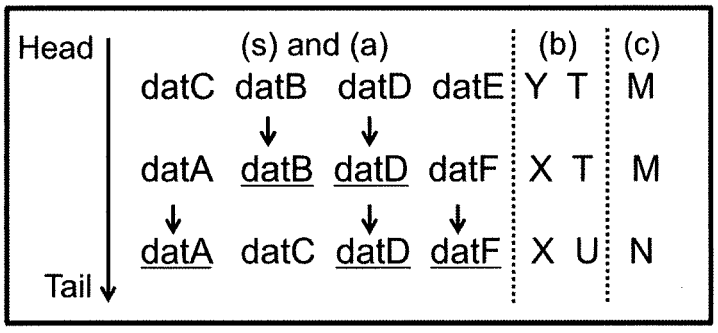


図 9　解凍した最終ログのイメージ

解凍自体のオーバヘッドによる組込みシステムの影響を避けるために，解凍機能は，“Log Acquiring Server”で提供する．上記に述べたデータの解凍方法で BSF ログを解析した後，図 9 のように先頭から読み込み，解凍することで従来の Ftrace 形式，JSON 形式および XML 形式に変換する．

3.4 LE の実装

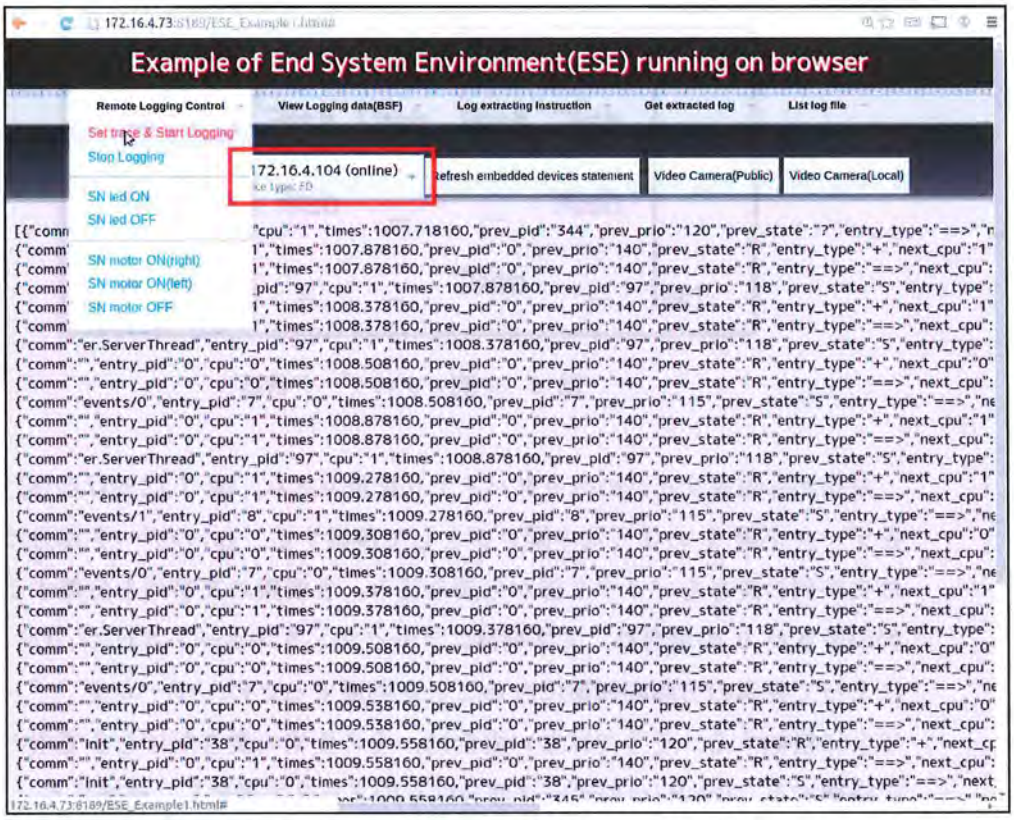


図 10 トレースデータの監視ツールの実行例

図 10 は LE のトレースデータ取得サーバの Rest API を用いたログの監視ツールの実行例を表す．これは，対象とするデバイスを選択して，トレースの開始 API を実行した後に，トレースの停止 API を実行し，JSON 形式に解凍する API を実行した後に，JSON 形式のトレースデータを取得する API を実行するものである．

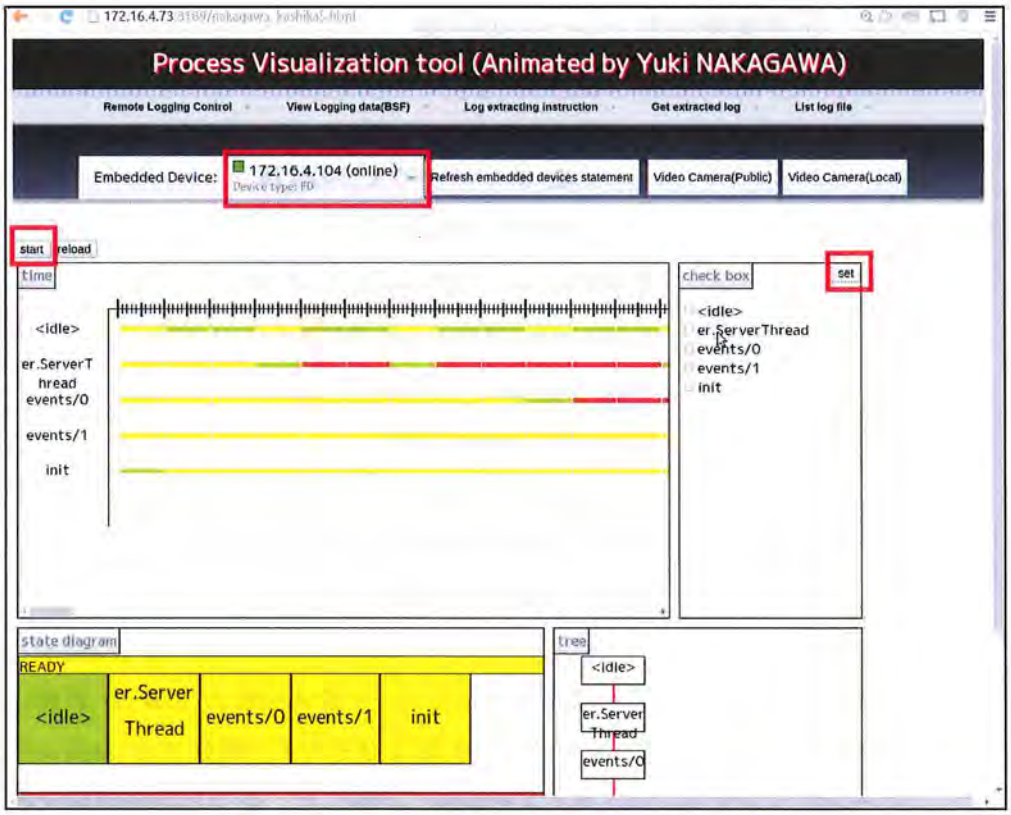


図 11　トレースデータの可視化ツールの実行例

図 11 は 2011 年から本研究室の修士の学生と共同開発となってきた Android OS のプロセスを可視化する環境[39]で LE のトレースデータ取得サーバの Rest API を用いて JSON 形式のトレースデータをベースにしたプロセスの可視化ツールとした LME のツールの実行例を表す.

これは，対象とするデバイスを選択して，トレースの開始 API を実行した後に，トレースの停止 API を実行し，JSON 形式のトレースデータに解凍する API を実行した．この時トレースデータがトレースデータ取得サーバに保存されたので，“set”のボタンをクリックしたら，トレースデータ取得サーバから JSON 形式のトレースデータを取得する API を実行して，[39]の可視化プログラムが取得されたトレースデータを初期化される．そして，“start”ボタンをクリックしたら，[39]の可視化ツールがアニメーションに対象とする組込み機器のプロセスを可視化させる.

3.5 圧縮機能付きトレースの評価

3.5.1 評価システムの環境

表 3 ハードウェアの仕様

KZM-A9 評価ボード	
CPU	ARM Cortex-A9MPCoreTM Dual CPU
	Operating Frequency 533MHz
Memory	Main Memory 512MB(DDR2-533)
	Internal SRAM 128KB
Touch Panel	Controller: EMMA-EV2 on-chip display
	LCD: WVGA(800*480 Panel, RGB(8:8:8))
	HDMI: Sil9024A(HDMI1.3)

本提案手法を，京都マイクロコンピュータ株式会社の KZM-A9 評価ボード上に動作する Linux カーネル 2.6.29 およびその上で動作する Android 2.2 をベースにシステムプロトタイプを実装し評価した．表 3 は評価環境を示す．

表 4 カーネルコードの変更

Kind	File Name	Patch (Lines)	Add (Lines)
Kernel internal code	trace.c	16	0
	trace.h	1	0
Trace driver code	trace_driver.c	0	2523
	trace_getmajorid.c	0	85
	compresslog.h	0	33

本システムを実装するために変更した部分を，表 4 に示す．該当カーネルのコードに 17 行のパッチが必要である．また，カーネル内からソケット通信を行うために ksocket モジュール ksocket code[40]を用いた．これは，カーネル内でソケット API を呼出すためのライブラリである．

3.5.2 評価実験と結果

本評価実験では，ユーザ空間でトレースデータ取得プログラムを起動させ，Ftrace を用いてネットワークを経由せずに SD カードに保存するトレース機構（以下，“Existing Trace”）とトレースデータをネットワーク経由で転送した場合（以下，“Existing Trace via Ethernet”）を基準にして，提案トレース機構（以下，“Proposed Trace”）を評価することを目的とする．

“Existing Trace” と “Existing Trace via Ethernet” は，Ftrace の sched_switch で実験を行った．なお，本評価実験ではネットワークに 100Mbps のイーサネットを用い，cpufreq の機能を用いず CPU 周波数は 533MHz 固定とした．

タイマ割込みについては NO_HZ 指定による変動周期タイマを用いた．変動周期タイマでは，タイマ割り込みは必要な時だけに発生する．また，高速なハードウェアタイマについては使用していない．

本実験では，ユーザがシステムに対して一定の行動を行うようにして，実験環境の再現性を確保した．具体的には，Android のブラウザ[1]を起動させて，重さ 500g のテキストブックをタッチパネル上に載せた．評価ターゲットのマシンには抵抗膜式のタッチパネルが搭載されているので，これによってパネルから同じデータを取得することが可能になる．実験はこの時点からトレースの実行時間(T)を 10 秒ごとに増やし，150 秒までの 15 回のデータを１セットとして，３セット試行を行い，その平均値を求めた．

---

[1] User-Agent: Mozilla/5.0 (Linux; U; Android2.2; ld-us; kmc_kzm9d Build/FRF91) AppleWebKit533.1(KHTML, likeGecko) Version/4.0 Mobile Safari/533.1

評価実験は次の評価項目に着目した.

1) **エクスポートされたトレースデータのサイズ**(Exported Log Data Size)

各トレース機構において，エクスポートされたトレースデータのサイズを表す.

- **Existing Trace** では，トレースデータを圧縮せずに，SD カードにおける入出力を用いた場合のサイズを表す．本トレースは従来の Ftrace 形式でエクスポートする.
- **Existing Trace via Ethernet** では，データを圧縮せずに，ユーザ空間でイーサネットを経由したデータ転送を行った場合のサイズを表す．本トレースは従来の Ftrace 形式でエクスポートされる.
- **Proposed Trace** では，データを圧縮し，カーネル空間でイーサネットを経由したデータ転送を行った場合のサイズを表す．本トレースは BSF ログ形式でエクスポートする.

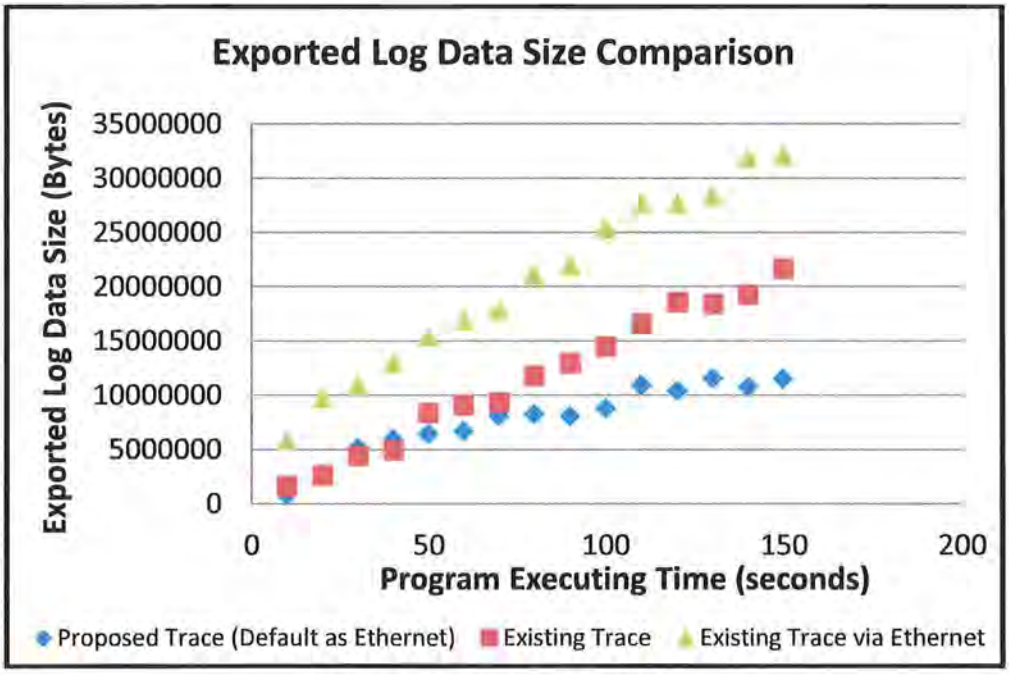


図 12　データのサイズの比較

実験結果を図 12 に示す．すべての方式において時間とともにトレースデータサイズは増加しているが，“Proposed Trace”は“Existing Trace via Ethernet”と比較してデータサイズの増大がゆるやかであり，SD カードを用いた既存のトレース方式とほとんど同じデータサイズで転送することが可能である.

2) **コンテキストスイッチ数** (The Number of Context switch) はプロセスのディスパッチ時にコンテキストを切り替えたプロセスの回数を表す.

- **Existing Trace** では，トレースデータを圧縮せずに SD カードへ入出力を行った場合の回数を表す.
- **Existing Trace via Ethernet** では，トレースデータを圧縮せずに，ユーザ空間へエクスポートし，イーサネットを経由したデータ転送を行った場合の回数を表す.
- **Proposed Trace** では，トレースデータを圧縮し，カーネル空間でイーサネットを経由したデータ転送を行った場合の回数を表す.

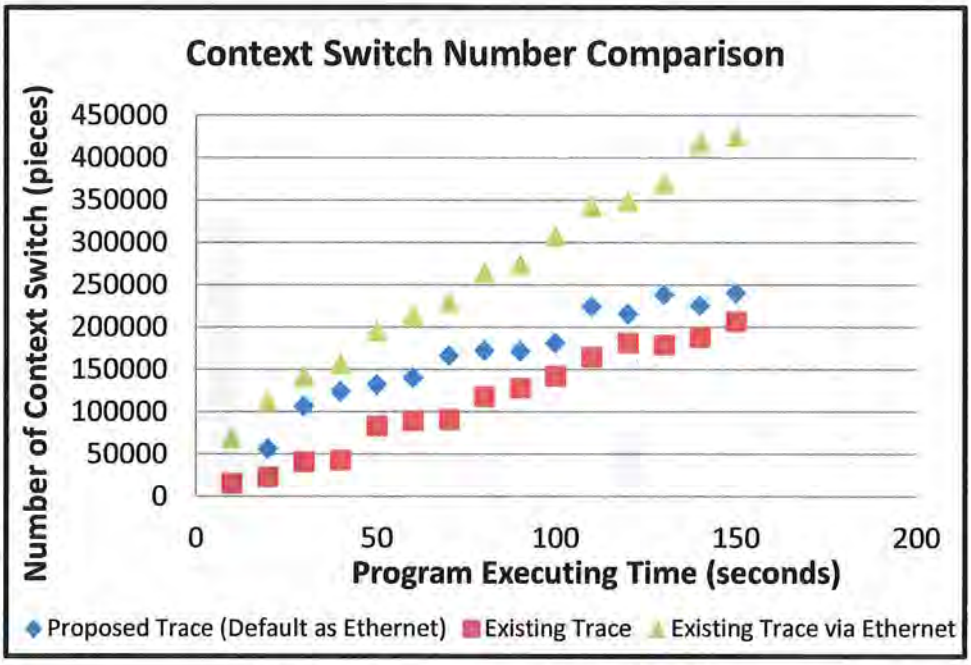


図 13　コンテキストスイッチ数の比較

実験結果を，図 13 に示す．いずれの方式も時間の経過とともにコンテキストスイッチ数は増加している．ユーザ空間からイーサネット経由でデータを転送する“Existing Trace via Ethernet”が一番多く，SD カードを用いた“Existing Trace”がもっとも少ない．“Proposed Trace”は一部で“Existing Trace”よりもコンテキストスイッチ数が増えるものと，“Existing Trace via Ethernet”よりも少ない数となっている.

3) **割込みの頻度** (Interrupt Frequency) は，上記のトレースを行う最中に発生した割込み頻度を表す．

- **system_timer** は，CPU0 へのタイマ割込みの頻度である．
- **system_timer1** は，CPU1 へのタイマ割込みの頻度である．
- **emxx_sdc** は，SD カードにおける入出力の割込みの頻度である．
- **eth0** は，イーサネットでのネットワークのデータ転送による割込みの頻度である．
- **kzm9d_touch** は，タッチパネルに対する入出力の割込みの頻度である．

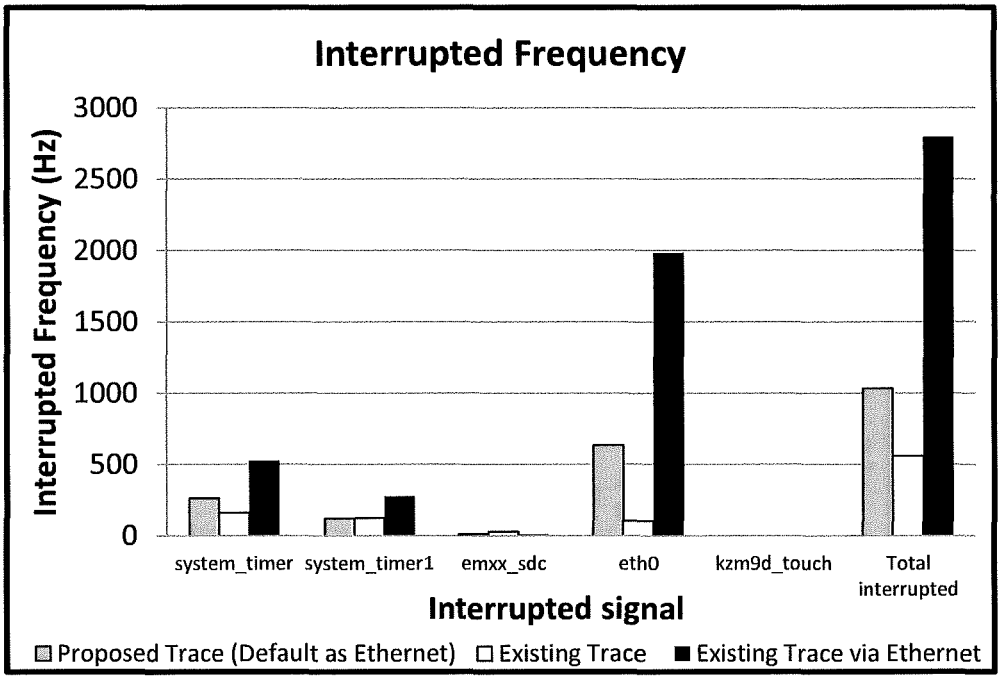


図 14　割込み頻度の比較

実験結果は図 14 に示す．提案手法は，“Existing Trace”よりも割込み頻度は高いが，“Exsiting Trace via Ethernet”と比較して，割込み頻度を低減していることが分かる．

4) **ログの圧縮率** (Percentage of Compressed Log)

本提案トレース機構の圧縮機能によるデータの圧縮率を表す．圧縮率は，本システムの“Log Acquiring Server”において BSF ログ形式を Ftrace 形式のトレースデータに変換した後，BSF ログのサイズと Ftrace 形式のサイズとを比較したものである．

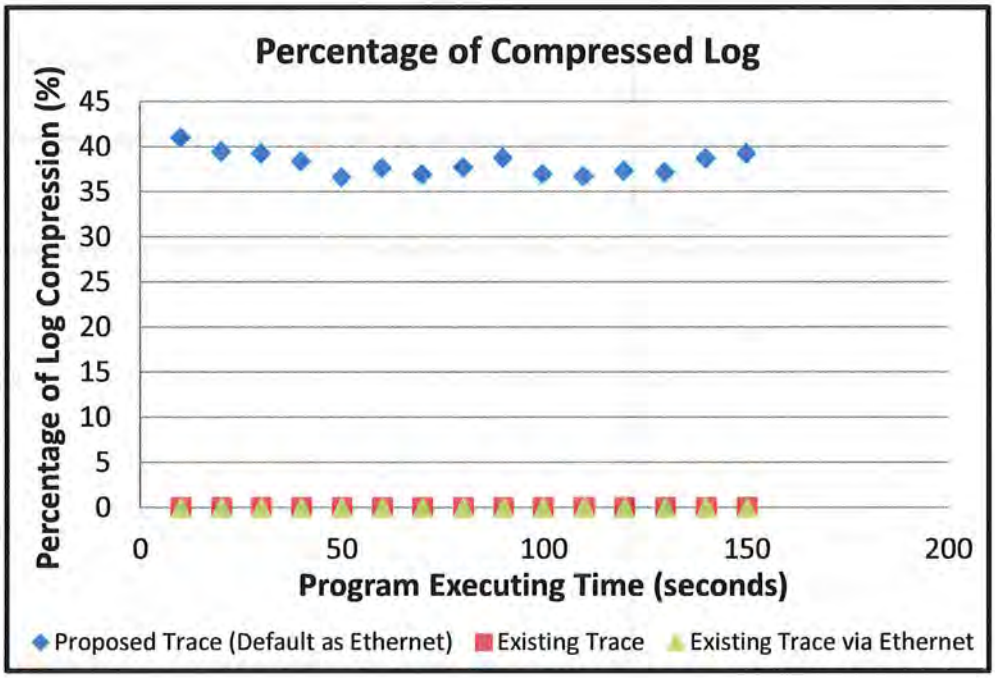


図 15　ログデータの圧縮率

実験結果は図 15 に示す．提案した手法は多少の変動はあるものの，ほぼ40%の圧縮率でトレースデータを圧縮できていることが分かる．

5) **システムのオーバヘッド (Overhead of System)**

提案手法を実現するために機能の追加，分離を行った際に，それぞれの機能の実行時間がどの程度変わるかをオーバヘッドとして計測した．具体的には，ネットワークのセットアップ，圧縮，文字列化，データのエクスポートのオーバヘッドに関して130秒間実行を3回行いその平均値を取得した．

表 5 130 秒に実行した場合のシステムのオーバヘッド

	Existing Trace	Existing Trace via Ethernet	Proposed Trace
Data Size (Bytes)	18902754	17261556	10380256
Compressor (sec)			0.948(0.73%)
String Converter(sec)	1.15(0.88%)	0.926(0.71%)	
Writing Data to Export Buffer(sec)	0.298(0.23%)	0.279(0.21%)	0.26(0.20%)
Network Communication (sec)		4.16(3.20%)	1.18(0.91%)
Total	1.45(1.11%)	5.37(4.13%)	2.39(1.84%)

- **Network Communication** は，イーサネットによるデータ転送をセットアップするのにかかるオーバヘッドである．
- **Compressor** は，本提案方式のデバイス制御におけるデータを圧縮するのにかかるオーバヘッドである．
- **String Converter** は，従来方式での文字列化にかかるオーバヘッドである．
- **Writing Data to Export Buffer** は，エクスポートバッファにデータを書き込むのにかかるオーバヘッドである．

実験結果を表 5 に示す．提案した手法は，“Existing Trace”よりもオーバヘッドが大きいが，“Existing Trace via Ethernet”と比較して，ほぼ半分のオーバヘッドとなっている．特に，ネットワーク転送を行うためのセット

アップのオーバヘッドが大幅に低減している．また，文字列化にかかるオーバヘッドが圧縮とほぼ同等の時間である．

3.5.3 考察

本節では，前章の実験結果について考察する．

3.5.3.1 エクスポートされたトレースデータのサイズとコンテキストスイッチの数

図 12 より，提案された手法は，データの増加率が既存の方法をイーサネットで送信することに比べて低く抑えられていることが分かる．これによって，システムが保持しなければならないトレースデータのサイズを押さえるとともに，外部に転送するときのネットワーク帯域を低く押さえることが可能となっている．

“Existing Trace via Ethernet”においてデータサイズが上方にシフトしているが，これは Trace pipe を用いることによるオーバヘッドと．ネットワークを介したデータ転送を必要とするためであると考えられる．

また，起動後 90 秒のあたりでデータサイズが小さくなる現象が起きている．これは図 15 からデータサイズ

の圧縮率が高くなっていることが分かる．これによってデータサイズが小さくなったことが原因であると考えられる．また，それ以降“Existing Trace”と比較してデータサイズが小さくなっているが，本圧縮方式では前のデータに影響を受けることから，一度圧縮率が上がったことでそれ以降のデータサイズが影響を受けているものである．

図 13 から，コンテキストスイッチ数についても“Existing Trace via Ethernet”より小さな増加率となっていて，提案手法がコンテキストスイッチ数を低減させていることが分かる．このグラフでも，“Exisiting Trace via Ethernet”が上方にシフトしているが，この理由としてはユーザ空間へのデータのコピーとネットワーク転送ためのシステム呼出しによってコンテキストスイッチが増加しているためだと考えられる．

また，30 秒～80 秒，100 秒，130 秒以降でコンテキストスイッチの増加が見られるが，図 15 と比較すると圧縮率の変動に関連していることが分かる．これは，圧縮率が落ちている部分では，エクスポートする回数が増えることでネットワークの転送が必要となるので，その分コンテキストスイッチが増加しているものと考えられる．

3.5.3.2 “Trace Driver”の影響と圧縮機能の効果

表 5 を基に，データの特徴(A)，(B)，(C)のオーバヘッドを減らす効果について考察する．システムのオーバヘッドについては，“Proposed Trace”での圧縮機能のオーバヘッドが，“Existing Trace”の文字列化とほぼ同等であると明らかになった．しかし，コンテキストにアプリケーション名の異なるデータへの切替えが頻発する場合，データの特徴(A)の圧縮率が低下してしまう．しかし，特徴(B)と(C)は上記のデータに依存しないことから，一定の圧縮をすることは可能である．また，今回は未実装だが，(A)の部分で登場するプロセス名などの文字列は頻繁に変わらないことから，変換テーブルを用意し，ID を割り当てることに圧縮率を改善することも可能である．

3.5.3.3 割込み頻度とトレースデータとの関係

図 13 と図 14 から，割込みの頻度が多いほどコンテキストスイッチの数が多くなるので，トレースデータサイズが増えログ消失の可能性が高くなることが分かる．“Existing Trace”は割込みの頻度の合計が最小なので，コンテキストスイッチ数が最も少なく，1 秒当たりのトレースデータの生成のサイズが最小になる．その結果として，リングバッファが新しいデータで上書きされる可能性は最小になり，トレースデータ消失の可能性も最小になると考えられる．そして，“Existing Trace via Ethernet”のデータから，そのトレースデータをイーサネット経由でマシンの外部に転送する場合，同様に割込みの頻度の合計が最大になるのでトレースデータ消失の可能性も最大になると考えられる．

これに対して，本提案のトレース機構では，ネットワーク通信の割込み回数を“Existing Trace via Ethernet”の約 30%に削減し，トレースデータサイズを従来の 40%に圧縮して転送することが可能である．この理由としては，イーサネット経由で圧縮されたデータを外部に転送するときに，トレースデータのサイズを 40%に小さくした転送が可能であることと，カーネル内で転送処理を行うことによる割込み回数の低減が考えられる．ログデータを低減させることで，ネットワークに占めるログデータ転送に必要となる帯域を減らすことができる．これは，多くの機器がネットワークに接続される場合により効果的である．

タッチパネルの割込み(kzm9d_touch)については，実験結果によるとタッチパネルの割込み発生していない．これは，抵抗膜式のタッチパネルなので固定の抵抗力にしか割込みが発生しないものである．本を載せた瞬間にタッチパネルの割込みが発生するが，実験中にはパネル上の本を移動させていないため

である.

また，SD カード(emxx_sdc)の割込みについては，実験結果によると“Proposed Trace”と“Existing Trace via Ethernet”は割込みが発生している．これは，本実験環境の KZM-A9 評価ボードは SD カードからカーネルをおよび Android，ブラウザを起動することから，このブラウザプログラムによるページングによって，SD カードへの入出力が発生して，割込みが発生していると考えらえれる.

3.5.3.4 転送処理自体をカーネル空間内で処理したことの影響

通常 Ftrace はリングバッファに収集されたトレースデータを 100m 秒ごとのタイマ割込みによってエクスポートするが，本実験環境では NO_HZ の指定による変動周期のため，タイマ割込みはシステムが busy か idle にかかわらず，必要な合だけに発生する．本実験の結果によるタイマ割込みは，図 14 “system_timer”と“system_timer1”である．この結果から見ると，“system_timer”と“system_timer1”の差分は Ftrace の処理によるタイマ割込みであると考えられる．その場合，システムタイマの割込みの頻度を従来の 50%に削減したことが分かる．原因としては，Existing Trace via Ethernet においてユーザ空間に起動する図 3 の“Log acquiring program”自体のディスパッチの回数の削減が挙げられる．そして，NO_HZ の指定のため， Ftrace の Trigger は二つの CPU コアのどちらかに偏って発生している現象が発生していることが考えられる．評価実験結果によると，Ftrace のデータエクスポートは CPU0 の方で主に処理されていることから，“system_timer”と“system_timer1”とが異なった結果となっている.

一方，カーネルに転送処理を移行したことにより，カーネル自体の割込みがマスクされる可能性がある．実験ではタイマ割り込みが NO_HZ の指定による変動周期なので，システムが busy でも idle でもタイマ割込みは必要なタイミングでだけに発生する．タイマ割込みのマスクの頻度は，測定該当トレースの“system_timer”，“system_timer1”の合計と Existing Trace の“system_timer”，“system_timer1”の合計との差分で表すことができる．本提案手法の割込みのマスクは 98 回であり，Existing Trace via Ethernet の割込みのマスクは 515 回である．これは圧縮によるトレースデータサイズの低減，およびユーザ空間へのデータコピー回数の低減による効果である．この結果，本提案手法は，ユーザ空間でイーサネットを用いた場合と比較してマスク

の回数を約 20%に削減したことが分かる．これにより，本方式がシステムの応答性の面でも改善されていることが分かる．

3.5.3.5 Log Acquiring Server のオーバヘッド

“Log Acquiring Server”におけるオーバヘッドは，従来の Ftrace 形式と JSON 形式とがほぼ同等である．従来のトレース機構と同等のオーバヘッドで Log Monitoring Environment においてブラウザに適用するツールやアプリケーションを実装するのに有効であると考える．

これらの考察から，本提案手法は，カーネル内に実装した低オーバヘッドの圧縮機能により，不要な割込みの頻発や割込みのマスクなどのシステムのスケジューリングに与える悪影響の要素を小さくしたことと Log Monitoring Environment におけるブラウザに適用するアプリケーションを実装するのにも有効であることで，ネットワーク接続された組込みシステムのトレース機構を利用することで，可視化アプリケーションを作ったりするのに有効であることが明らかになった．

3.6 関連研究

3.6.1 LTTng/LTTv

従来のトレース機構として，Linux Trace Toolkit Next Generation (LTTng)と Linux Trace Toolkit Viewer (LTTv)がある[34]．LTTng は標準のカーネルトレースのようにシステムコールを用いずに，ユーザ空間でロギングを制御する低オーバヘッドのトレース機構であり，Common Trace Format(CTF)に変換したトレースデータをユーザ空間にエクスポートする．これは機器自体にトレースデータ保存する形である．そして，SSL によってデータを送信し，ホスト側で起動する LTTv で解析して可視化するツールキットである．

CTF を利用することで複数のツールでトレースデータを利用することができる．また，トレース機構のオーバヘッドは低く，ユーザ空間にエクスポートし，カーネルを通して二次記憶上に保存する．しかし，LTTng では各ツールはユーザ空間で実行されることから，コンテキストスイッチへの影響を避けることはできない．また，ネットワークを介して複数のマシン間でトレースデータをやり取りする場合，ユーザ空間のツールを利用するために実行コストがかかるという問題がある．

3.6.2 TLV

Trace Log Visualizer(TLV)[35]は汎用性と拡張性を実現するのが目的とする可視化ツールである．汎用性を実現するためにトレースデータの標準形式を定め，これを変換ルールによってユーザが外部から指定した仕組みで提供する．汎用的な可視化表示機構を提供し，ユーザが可視化表示項目に合わせて拡張することができる．

これは，マルチコアをサポートした RTOS である TOPPERS を対象とした可視化ツールであり，組込み Linux カーネルに対しては現在サポートが行われていない．

3.6.3 SystemTap

SystemTap[33]はユーザ空間およびカーネル空間のイベントを取得する Linux のプローブツールである．これは，拡張性，使いやすさ，性能，透明性，簡潔性，柔軟性，安全性の中で，必要なものを選択して利用できる．カーネルデバッグユーザは特定のトラップスクリプト言語を書き込んでコンパイルすることによって特定のイベントを取得することができる．しかし，SystemTap 自

体はユーザ空間で動作することから，スケジューリングへの影響を生じる可能性があるので，コンテキストスイッチのディスパッチに影響を与えないトレース機構が必要である．

3.7 結論

本章では，組込み Linux システムを対象として低オーバヘッドなプロセスのトレース機構を提案し，実装評価を行った．システムはトレースデータ生成側の組込み機器とトレースデータ取得側の計算機とに分離し，カーネル内部でデータを転送することで，生成側への影響を減らしつつトレースデータを取得することが可能になった．トレース時にデータフォーマットの特徴を活かして，実用上問題がない低オーバヘッドの圧縮機構を組み込むことで，より多くのトレースデータをメモリ内に保持できるようになり，システムのリアルタイム性への影響を減らすことを可能にした．これを Linux システム上で実装評価を行い，本方式を，従来の Ftrace をメモリ上に記録した場合およびトレースデータをユーザ空間からネットワークで転送した場合と比較を行った．その結果，本提案方式は，平均で 40%のトレースデータの圧縮が可能であることが明らかになった．

結果として，ユーザ空間へのプロセス切替えによる割込みの頻度や割込みマスクの回数を減らしつつ，生成されるトレースデータのサイズを削減させたことで，システムのスケジューリングやネットワークの帯域などの影響を低減させつつトレースすることを可能にしたことから，システムにおけるトレース機構として有効であることが明らかになった．

第４章　ウェブブラウザベースのプログラミング環境

本章では，ウェブブラウザベースのプログラミング環境について述べる．

4.1　まえがき

モノのインターネット化（Internet of Things，以下 IoT）[41]は，アクチュエーションやセンシングによるインタラクションを行う組込みシステムを内蔵した複数のモノを，インターネットによって接続することによって，サービスを提供するシステムである．IoT での対象としては，ホームオートメーションや遠隔地間でのデータのセンシングやアクチュエーションがある．インターネットによる広域での接続性によって，位置に依存しない形でデバイスをサポートすることが可能になることから，より広い分野での応用が可能になる．

しかし，IoT 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備える必要があるが，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．従来の組込みシステムと比較して，よりオープンな形でデバイスを扱うことから，マシンローカルなプログラミングモデルを中心としたシステムでは，サポートが不十分である．

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通信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備えなければならないことが多く，開発者にとって手間がかかる．さらに，IoT システムを構築する際には，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ので，そのための監視システムが必要となる．

このような問題に対して，本研究は HTTP をベースとした IoT システムの構築を行った．HTTP をベースとすることでデバイスとサーバ間の接続性が向上し，より簡単にシステムを構築することができる．また，システムの開発を容易にするために，ブラウザベースのシステム開発環境を提供した．この開発環境では，ブラウザ上でのコーディングや，各デバイスの操作，さらに通信のモニタリングと表示を可能とした．また，デバイスにアクセスする API を提供することで，デバイスの種類に依存しない形でのプログラミングを可能にした．モニタリングについては，HTTP ヘッダ内に通信情報を内包することで，定常的なモニタリングを可能にした．

4.2 要求分析

本環境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**

IoT で用いる組込みシステムは, 従来の PC とは異なり, 複数, 多種なデバイスを, いろいろな場所で用いることが多い.

また, 最近では Raspberry Pi や Intel Edison などの Linux が稼働するデバイスを安価に入手することが可能であるが, バージョンなどにおいて入出力ハードウェアが異なる場合があり, マシンに応じたプログラムを動作させる必要がある.

Scratch[42][25]や Python の IDLE のようにボード内で実行可能なプログラミング環境も存在するが, 利用するにはマシンごとの接続を切り替えたり, 遠隔ログインなどの手間がかかる. Eclipse[43]などの本格的な統合開発環境はインストールに時間がかかることや要求する計算機資源量が大きい.

IoT 環境では, 実行するマシンが備えている計算機資源は複数存在する. 拡張が難しいことや, インストールコストが高いことから, このような環境を管理する機構自体は多様なマシンで動作し, 管理オーバヘッドが動作に影響を与えにくく, インストールコストが低いことが求められる.

(2) **プロトタイピングが容易なこと**

IoT においては, デバイスの追加が用意であり, それに合わせて動的なサービスの追加が必要になる. デバイスの接続は, Linux に搭載された GPIO や SPI や PWM などのシステムインターフェースを制御することで, センサ・アクチュエータを制御する Linux のモジュールを用いることが多いが, デバイスの意味付けはユーザのプログラムによって与えることになる. そこで, 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, 記述しやすく, 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**

多くのモノが繋がる IoT では，ネットワークでの通信時間に応じて，プログラムやシステム構成を見直す必要がでてくる．見直しを行う際に，通信状態のキャプチャや解析などのモニタリング[26]が必要となる．IoT 環境では，センサ／アクチュエータを持つ複数のデバイスをネットワーク経由で扱うことから，ネットワークの性能がプログラムやシステム構成に影響を与える．ネットワーク構成は，制御などで大きく性能が変化することから，システムのプロトタイピング，および性能測定，さらにプログラム実行先の容易な変更が必要になる．

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．

通信状態のキャプチャで生じるオーバヘッドについては，通信への影響が少ないことが望ましい．実行環境のオーバヘッドが通信に大きく影響が出る場合，キャプチャしたデータの妥当性の検証が難しくなるためである．

4.3 特徴

前節の要求分析を受け，次のような特徴を持つ IoT 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 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 ブラウザだけを用意できれば，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．

また，本システムでは，接続のしやすさを考え，HTTP をベースとした通信プロトコルを採用した．IoT システムの通信プロトコルとしては，HTTP 以外に Websocket[44]や，MQTT[45]などを用いる例がある．Websocket は，リモートセンシングを行う場合，サーバにプッシュすることが可能であり，コネクション生成やデータプッシュに伴うオーバ

ヘッドを低減させることができる，多くのブラウザでサポートされている．また、MQTT は pub/sub アーキテクチャに基づいた軽量なプロトコルで，IoT 環境において多くの利用実績がある．

その一方で，Websocket はプロキシを介したネットワーク接続では利用できないことがあり，3G などの低速回線や，遠隔地間の接続ではメッセージの遅延が生じるなどの問題がある．また，ブラウザによっては未サポートとなっているものがあり，接続性の点で問題が生じる．MQTT は，プロキシや NAT を含む環境での利用に工夫が必要であり，ブラウザベースの環境では利用しにくい．

HTTP は接続やデータサイズの点でオーバヘッドが生じるというデメリットがあるものの，接続可能性が高く，環境を整備するコストが低いことから，本システムでは基本的なプロトコルとして採用した．

(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センサの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) **複数のデバイスを統一的に管理できるフレームワークの提供**：

複数，多種のデバイスを統一的に管理するために，センサを接続するノードで実行する管理機構を提供する．各ノードは実行するプログラムからは，ネットワークで接続されたリモートデバイスとして見えるように仮想化を行った．これによって，デバイスの細かな仕様の違いを意識することなく，プログラミングを行うことが可能になった．

この管理機構は，Linux 上で Java 言語を用いて記述し，ファイルシステムを経由してデバイスにアクセスを行う．これによって，Linux が稼働するデバイスであれば，本システムで利用することが可能になる．

4.4 設計

本節は IoT を対象とした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．

4.4.1 全体構成

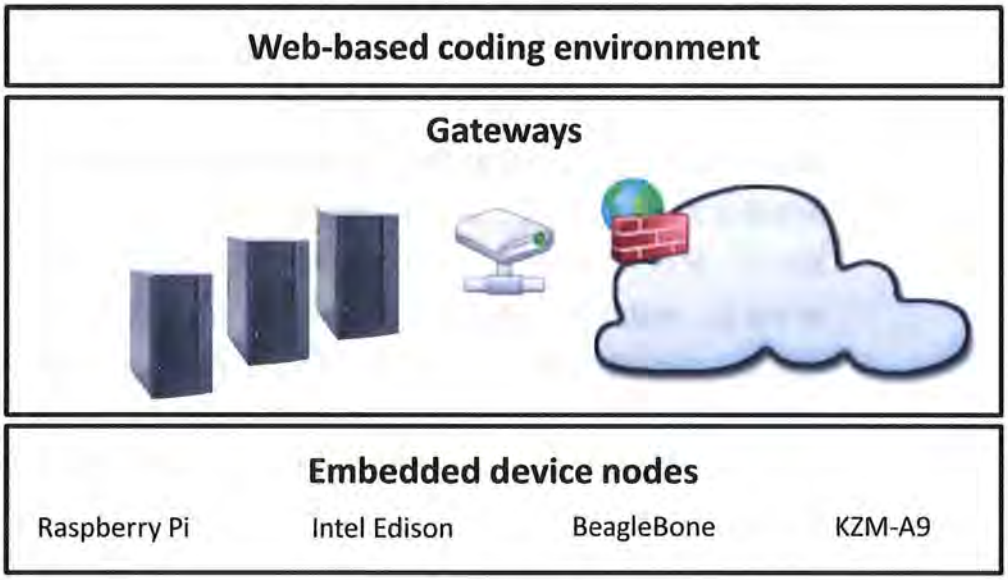


図 16　システムの階層

本システムの階層を図 16 に示す．本開発環境は三つのレイヤから構成する．

(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い，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)を実行する．ノードを制御プログラムの管理下に置くことで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．ブラウザベースのプログラミングエディタ，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 を介した通信シーケンスはログとして保存することで，ノードとサーバ間の通信を可視化することが可能になる．また，分散ファイルシステムである GlusterFS [46]や Key-Value Store である redis [47]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行い，後で分析することを可能にしている．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上で実行する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 17 の全体構成を採用した.

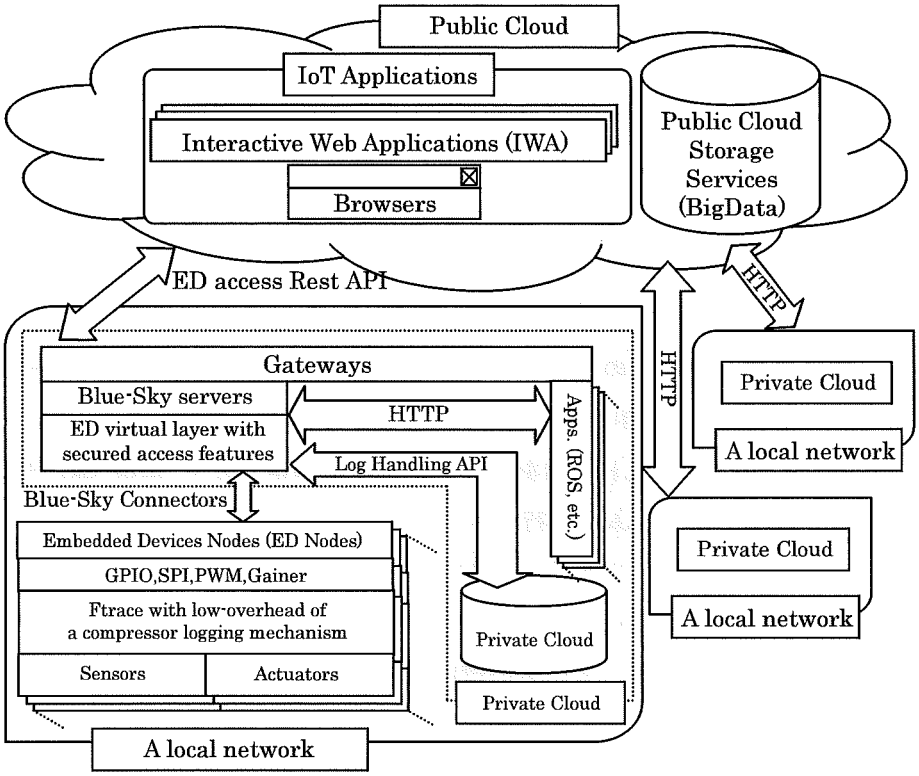


図 17 システムの全体構成

図 17 はシステムの全体構成を示す. 図 17 の Blue-Sky サーバ[48]はゲートウェイレイヤに内装されたサーバである. 本サーバは，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. IoT では，複数のデバイスノードから取得したデータを共有して利用可能でなければならない. 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に接続可能な形態とする. このようなハイブリッドクラウド上に Sensing Instrument as a Service (SIaaS) [12]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.

4.4.2 Blue-Sky モジュール：組込み機器ノードの制御機能の設計

組込み機器ノードの制御機能は，各ノードの制御プログラムをセキュアに実行することと，組込み機器ノードやゲートウェイにおける処理の負荷と使用するネットワークの通信量を減らすことにある．

ゲートウェイと同一にプライベートネットワークで動作することで，セキュアに組込み機器のノードを実行する．ゲートウェイは外部ネットワークから各組込み機器ノードへのアクセスを制限している．このような構造にすることで，センサデータの不正な取得や，侵入によるアクチュエータの不正な駆動を防ぎ，安全なプライベートクラウド環境を提供することができる．

組込み機器ノードのレイヤにおける各ノード上では，ゲートウェイへの接続やノードの発見，GPIO などのハードウェアインターフェースの中継，ノード状態の取得を行うノード状態をゲートウェイに送信したり Blue-Sky モジュールと呼ぶ制御プログラムを実行する．

ノードの発見は，プライベートネットワーク内にブロードキャストパケットを送信することによって行う．ノードを発見した後に，サーバ内にノードインスタンスが生成され，ノードの情報はサーバ内で管理する．このような形態にすることで，ノードへの直接的なアクセスによる通信量の増加および，ノードの負荷を減らし，ノード情報の利用を容易にした．

ゲートウェイは，ノード上のハードウェアへの操作命令を中継する．操作命令としては，GPIO および SPI といった低レベルなハードウェアインタフェースを提供している．これは，ノード上での仮想化によってチューニングの余地がなくなることを防ぐためである．

4.4.3 ゲートウェイの設計

プログラミング機能を持つウェブコンテンツを提供するとともに，パブリック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ノードへ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．これによりノードの負荷と通信量を減らすことができる．

パブリッククラウドサービスからノードへのアクセスを可能にするために，ゲートウェイ内部のノードインスタンスを経由して，組込み機器ノードレイヤへのアクセスするための REST API を外部に提供する．ノード情報の管理や API の処理はゲートウェイ上で動作するために，計算機資源の貧弱な組込みノード上での処理を抑えられる．

ゲートウェイでは，パブリッククラウド上のストレージと連携して，ログデータを保存することができる．ゲートウェイはセンサ関連の API を中継する時点でそのデータをログとして取得することができる．ログは，パブリッククラウドが提供するストレージ上へ格納して，センサデータの解析に用いる．

また，Web 開発環境で開発したコードを，ゲートウェイやノード上にデプロイして，実行する機能も提供する．これは，通信やマシンの負荷に合わせたシステム構成の変更を可能にするためである．負荷や通信量の測定はゲートウェイが提供する API 節 4.4.4 を用いて行い，その結果に応じた形でノードもしくは，ゲートウェイなどでプログラムを実行することができる．これによって，通信オーバヘッドやシステム負荷を考慮してプログラムの実行先を，遠隔地もしくはデバイスの近くに移送して実験することが容易になる．

4.4.4 スクリプトおよびAPIの設計

本開発環境のAPIは，Webサービスにおける REST API として実現されている．これによって Web プログラミングと親和性のある形で機能を提供する．API はゲートウェイが提供し，Public API と Local API の二つに分類される．Public API は，外部ネットワークからでも呼び出すことができる API である．一方，Local API は，アクセス制限を設定可能な API であり，モータやロボットアームといった物理世界に影響を与えるアクチュエーションを行う場合を想定している．このように二つの API に分けることによって，アクチュエーションの安全性を確保している．

REST API を提供することにより，メモリ性能が高くない組込み機器ノード上でセッションの状態を管理する必要がなくなり，通信データのサイズが大きい HTTP を用いた場合でも，接続性を向上させて，アプリケーションを開発することが可能である．

API の URL は，次の形式となっている．

『http://[ゲートウェイ]/ETLog?instruction=[命令]&opt1=[ノードの IP かノード名]&opt2=[各ノードのインターフェース名・サブシステム名]&opt3=[サブシステムのパラメタ 1]&opt4=[サブシステムのパラメタ 2]...』

API に用いる命令は，1) ゲートウェイに接続するノード状態のデータを取得する ls 命令，2) 各ノードのサブシステムやインターフェースをアクセスする sensornetwork 命令と actuatorcontrol 命令がある．API は，各命令に関する実行結果を body コンテンツに格納することで，各デバイスへの実行エラーのハンドリングは簡潔に記述することが可能になる．

インタフェース名・サブシステム名としては，GPIO や SPI が挙げられる．例えば，組込み機器ノード IP: x.x.x.x における GPIO の 21 ピン番号に 1 の信号を出力する場合は，URL のエンコードは，

『http://[ゲートウェイ名]/ETLog?instruction=sensornetwork&opt1=x.x.x.x&opt2=gpio&opt3=set&opt4=21&opt5=1』

である．

本開発環境はこの REST API を実行するコマンドや JavaScript の関数の API を設計する．例えば，コマンドの場合，

『lsn x.x.x.x gpio set ピン番号 データ(0 または 1)』

においては，ローカルネットワーク内のノードに sensornetwork 命令を実行

するコマンドである.

プログラムをノードにデプロイするには，デプロイ先の情報が必要である.情報を取得するためには,

『http://[ゲートウェイ名]/ETLog?instruction=wherecodereceiver&opt1=noneFix&opt2=json』

として指定する．これによって得られた情報を元に，節 4.4.3 で述べたプログラムをどこにデプロイするかを決定することができる.

4.5 実現

本説は，実現について述べる．

4.5.1 IoT 向きブラウザベース開発環境

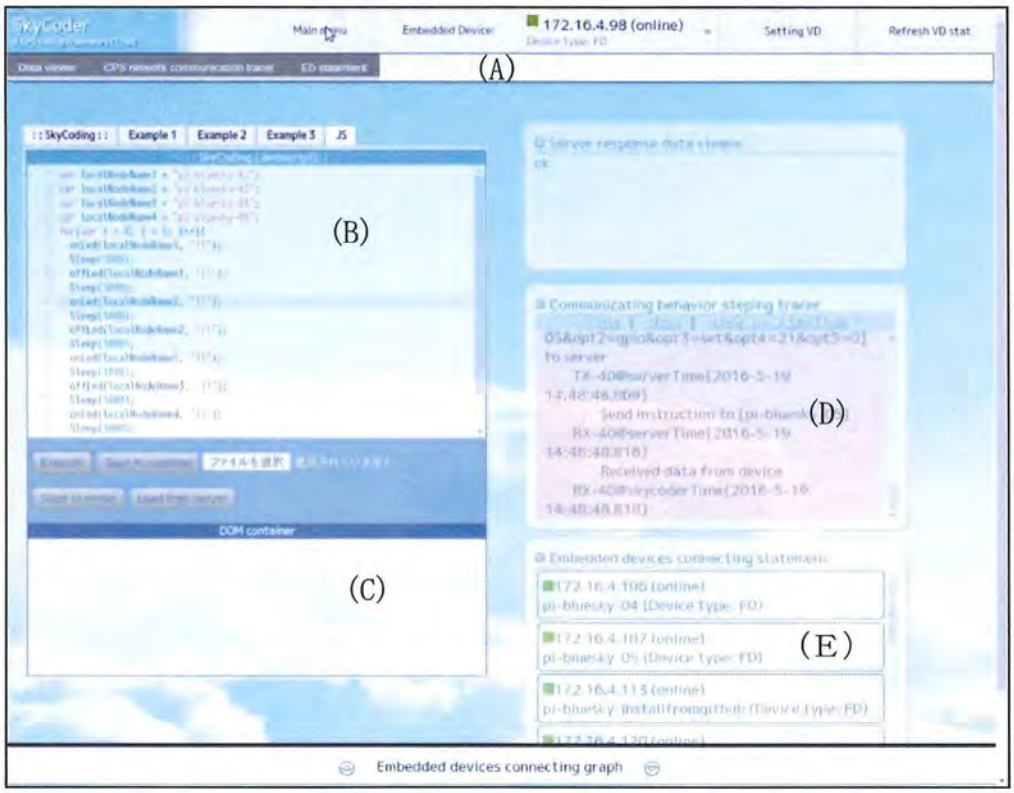


図 18 IoT 向きブラウザベースのシステム開発環境の実行例

図 18 に Web ベース開発環境の実行例を表す．本開発環境（以下，“skycoder”）は，ゲートウェイ上の Web サーバで提供し，ローカルの Web ブラウザ上で実行する．本環境は，ゲートウェイが提供する API と連携させて，JavaScript と HTML/CSS における DOM[49]や XMLHttpRequest[50]や Ajax によって実現した．本環境のメインウィンドウは，次の 5 つのコンポーネントから構成している．

(A) ツールバー

(B) コーティングエディタ

(C) 描画コンテナー

(D) 通信シーケンスの監視

(E) 組込み機器ノード接続状態の監視

また，組込み機器のノード状態の監視，データの通信シーケンスの監視，タイムラインで通信シーケンスの可視化の機能を提供する．この理由としては，IoT では多くのノードにセンサ，アクチュエータを接続し，それらを管理する必要があること，ノードとサーバ，ブラウザとの間で通信が発生し，それがシステムの性能やプログラムに影響を与えることから，これらを開発者に分かりやすく提示する必要があるためである．

次にメインウィンドウのコンポーネントについて説明する．

(A) ツールバー

ブラウザの狭い画面を有効に活用するために，各コンポーネントの表示，非表示を指定することができる．また，この画面では表示していない全デバイスの接続状態の可視化ウィンドウ，および通信シーケンスのタイムライン可視化ウィンドウなどの可視化ウィンドウを呼び出すことができる．

(B) コーディングエディタ

コーディングエディタは，コード入力を行うコンポーネントである．コマンドスクリプト，および JavaScript の二つの言語への対応，および複数台数のマシンでのコーディングを容易にするタブを備えている．通常コードはブラウザ上で動作するが，本ツールの下部に備えた実行ボタンや保存ボタンを用いて，プログラムをゲートウェイ，もしくはノード上で実行することが可能である．

図 19 は，IP アドレス 172.16.4.68 のノードの GPIO ピン 21 に LED を接続して，LED を 3 回に点滅した後に，IP アドレス:172.16.4.65 のノードに接続したセンサデータを取得するコマンドラインスクリプトをコーディングする例である．本ツールでスクリプトを実行すると，1 行ずつステップ実行によって，デバイスへのアクセスを確認することができる．実行結果は，コンソールの下部にある表示ウィンドウに表示される．コマンドラインスクリプトを用いることで，対話的にデバイスの動作を確認することが可能になる．

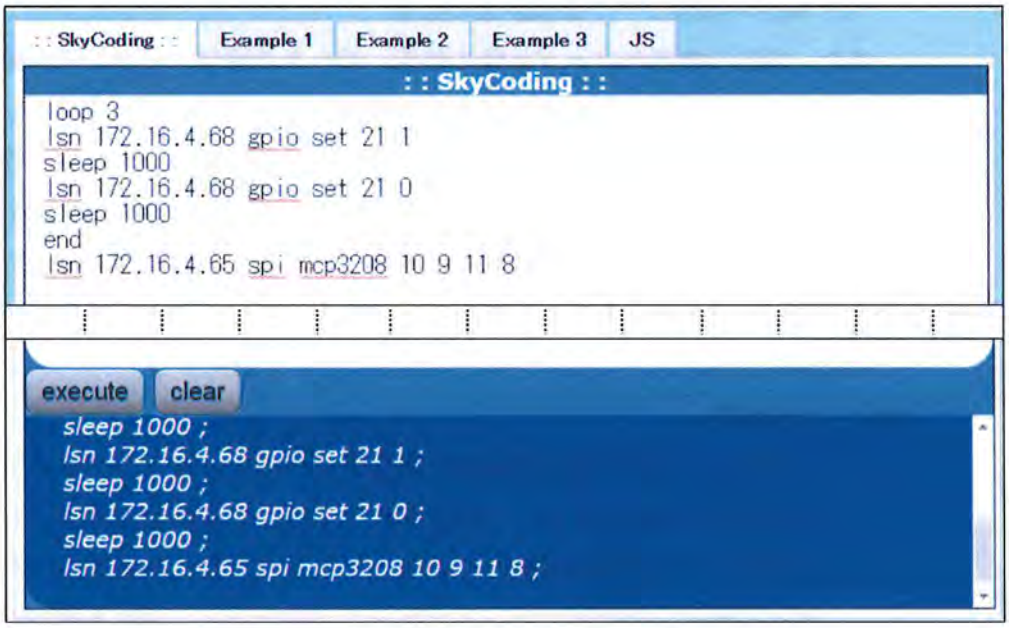


図 19　コーディングツールのコマンドラインタブにコーディング事例

図 20 はコーディングエディタを用いて，複数のセンサノードからセンサデータを取得する JavaScript プログラムの開発例である．Blue-Sky が提供する sensornetwork API および，それを用いた上位 API である onLed/offLed を用いて LED へアクセスするとともに，SPI で接続されたセンサからの値を取得している．コーディングエディタでは，JavaScript の構文エディタとして利用することができる．また，実行結果については(C)の描画コンテナに表示することが可能である．さらに，ノードとの通信については，通信シーケンス表示コンポーネントで見ることが可能である．

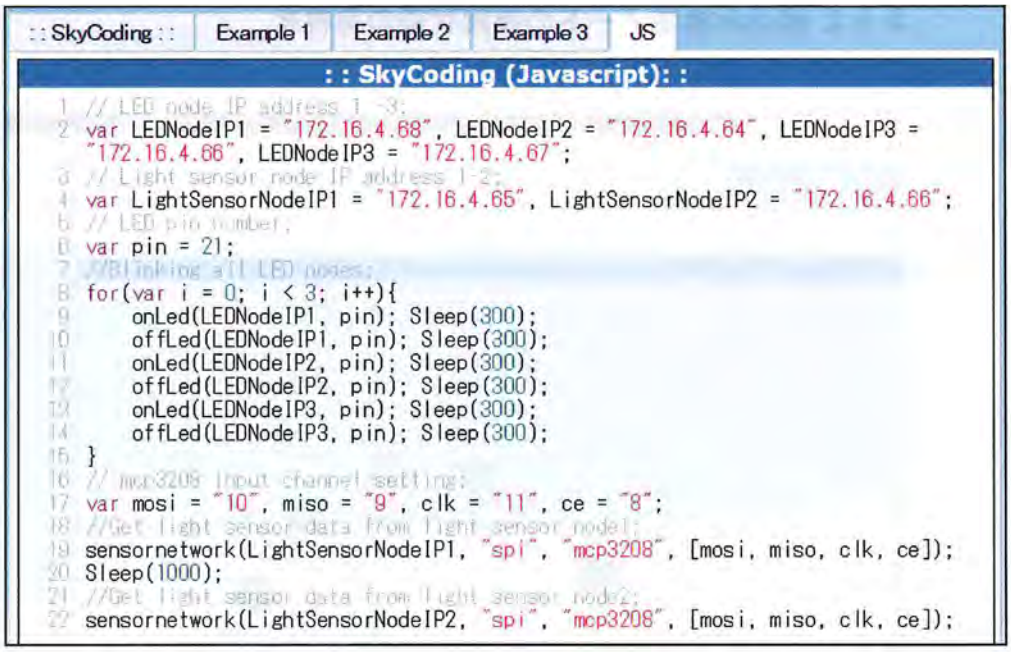


図 20　コーディングツールの JavaScript タブにアプリケーションの開発事例

(C)　描画コンテナ

本コンポーネントは，(B)でコーディングした JavaScript プログラム内からアクセス可能な描画コンテナである．このコンテナには結果を自由に表示することが可能である．本コンテナは，図 20 から“getDomContainer();”で呼び出す．

(D)　通信シーケンスの監視

ノードとブラウザ間のデータのやりとりを監視するコンポーネントである．この詳細については，4.5.3.2 で述べる．

(E)　組込み機器ノード接続状態の監視

接続している組込み機器ノードの全体の接続状態を確認ができるように開発環境をノードとして表示する．組込み機器ノードの接続状態を表形式で表示することで，デバイス状態を確認できるようにする．

4.5.2 組込み機器ノードの接続状態の可視化

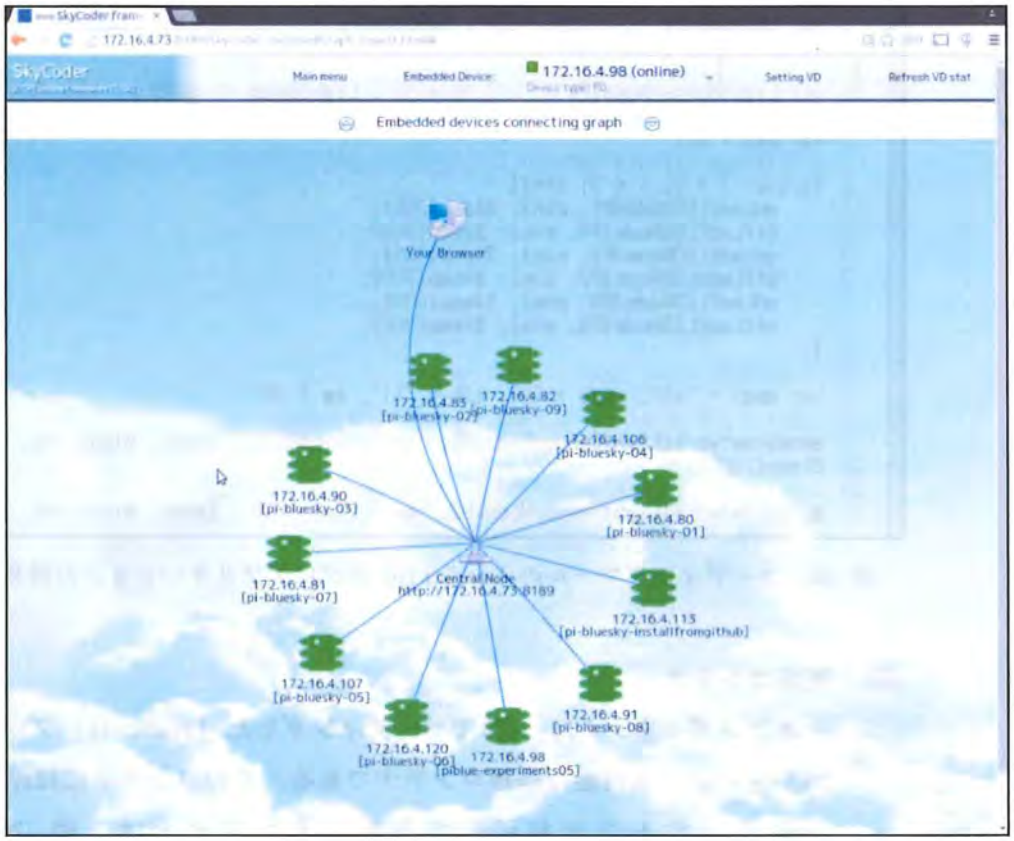


図 21　組込み機器ノードの接続状態の可視化

図 21 はゲートウェイに接続している組込み機器のノードの状態の可視化の例である．ノード状態は開発環境内に表形式で表示されているが，ノード数が増加すると一覧性が低下する．そこで，ノード状態を動的に取得し，グラフィカルに表示することで，ノードの接続状態の変化や，IP アドレスや接続デバイスなどのノードの情報を把握しやすくしている．この表示には，vis.js[51]の可視化データセットを用いた．

4.5.3 タイムスタンプ付きデータによる通信シーケンスの監視

4.5.3.1 HTTP のレスポンスヘッダ内へのタイムスタンプの格納

```
Accept-Ranges:bytes
Access-Control-Allow-Origin:*
Access-Control-Expose-Headers:Et-Server-Timestamp,
                              Server-Et-Timestamp
Connection:close
Content-Length:2
Content-Type:text/html
Et-Server-Timestamp:1463624813713
Server:Blue-sky-Server/1.1
Server-Et-Timestamp:1463624813708
```

図 22　HTTP レスポンスヘッダ内のタイムスタンプ

プログラムを修正せずにネットワークの通信状態を把握するために，HTTP のレスポンスヘッダ内にタイムスタンプを格納して，常に RTT の計測ができるようにした．

図 22 は Blue-Sky のサーバからの HTTP レスポンスヘッダを示す．ヘッダには，時間に関する情報を格納するフィールドを拡張している．Server-ET-Timestamp フィールドは，Blue-Sky サーバの上に組込み機器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 フィールドは，組込み機器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．この二つの時刻を用いて，通信シーケンスの表示および，その可視化であるタイムライン表示を行う．

4.5.3.2　データの通信シーケンスの監視

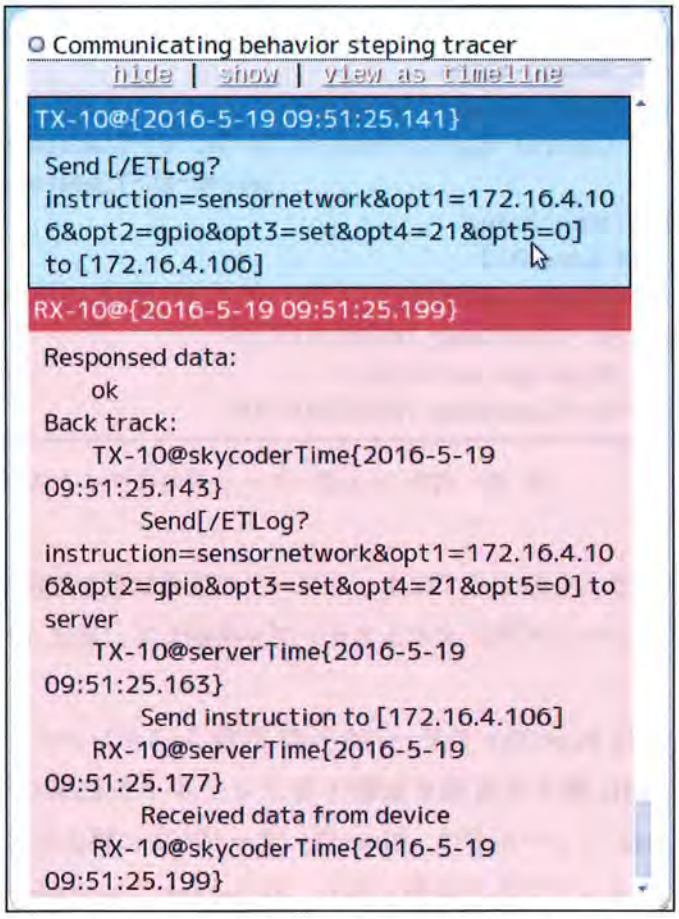


図 23　タイムスタンプ付属データの通信シーケンスの監視

図 23 は通信シーケンスの監視の実現を示す．本通信シーケンスは，ブラウザ，サーバ，ノードそれぞれ間のデータの通信状態を監視するツールである．TX はデータの送信，RX はデータの受信，その後ろの数字はデータの送受信の回数を表す．“@”の後はグリニッジ標準時(以下，“GMT”)のタイムスタンプである．タイムスタンプを GMT によって統一することで，時間帯をまたいだ地点間での通信をモニタリングすることが可能になる．プログラムを実行している skycoder からノードへ通信を行う場合，送信時のタイムスタンプである skycoderTime を記録する．次にこの通信を受け取った Blue-Sky サーバが組込み機器ノードに送信するタイムスタンプである serverTime を記録する．ノードから結果が返信されるときにも，同じようにサーバおよび skycoder でタイムスタンプを記録する．このように通過したシステムの時間をすべて保存し，

送信側へ返すことにより，ネットワークの通信時間に関する情報を把握することが可能になる．

4.5.3.3 タイムラインで通信シーケンスの可視化

図 24 はデータの通信シーケンスをタイムラインとして可視化するツールである．データの通信シーケンスの監視ツールにスタンプされたタイムスタンプを vis.js のデータセット[52]にマッピングするとともに RTT を計算することで，図 24 の形式で可視化することができる．図 20 のアプリケーションを実行した後に，図 23 の“view as timeline”にクリックすると，すべての TX と RX の情報に付加されたタイムスタンプ情報から RTT 計算して表示を行う．

本ツールは vis.js の可視化ライブラリの一つであるタイムライン[52]を用いて実装した．

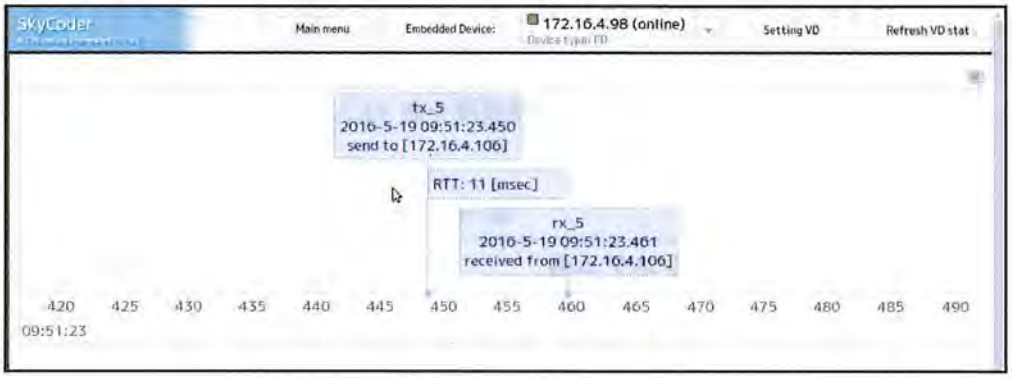


図 24　タイムラインで通信シーケンスの可視化

4.6 システムの評価

本節は，評価実験，実験結果と考察を述べる．

4.6.1 評価概要

本評価実験では，次の項目について行った．

(1) フレームワーク自体のオーバヘッド計測

(2) IoTアプリケーションのコードサイズおよび開発時間に関する評価

(3) RTTの計測および可視化ツールを用いた性能向上の事例

(1)は，要求分析におけるモニタリングおよび特徴においてHTTP通信を用いたことによる，システムオーバヘッドの影響について評価を行った．(2)は，実際に２つのIoTアプリケーションを開発し，そのコードサイズを既存の手法と比較検討して，本システムの有効性を検証した．(3)は，タイムライン可視化ツールの有効性を検証するために，インターネット接続された組込み機器ノードおよびゲートウェイに対して，ノードのLEDにアクセスするプロトタイプを作成，計測し，システムを改良することで，性能向上が可能になった事例について検討する．本評価の実験環境を表 6に示す．

表 6　実験の環境

システム名		仕様
skycoder	ブラウザ	Chrome 46.0.2490.86
	OS	Ubuntu 12.04.5 LTS
サーバ	CPU	AMD Athlon(tm) IIX2 260 Processor
組込機器	Raspberry Pi	Model B+ v.1.2
	Blue-Sky モジュール	v.pre-release
	node.js	v.0.6.19
	java	v.1.8.0
	OS	Raspbian (v.2015-05-05)

4.6.2 評価実験

4.6.2.1　フレームワーク自体のオーバヘッド計測

フレームワーク自体のオーバヘッドを計測した．本システムでは，組込み機器ノードへのアクセスに対して，必ずゲートウェイ上にある Blue-Sky サーバを経由する．そこで，ノード上の LED へのアクセスを行う場合の，組込み機器

ノードおよびBlue-Skyサーバの通信オーバヘッドについて計測した.

その結果を，表 7に示す.

表 7　システムオーバヘッド

サーバ（ms）	組込み機器ノード（ms）
2	8

4.6.2.2　IoTアプリケーションのコードサイズおよび開発時間に関する評価

開発環境を用いてアプリケーション開発を行う場合に，コードサイズが短いことは，開発の容易さに繋がる．そこで，本環境の API を用いた場合と，JavaScript を用いてすべて記述した従来の場合とでコードサイズを比較して，開発のしやすさについて評価を行う.

開発対象のアプリケーションは次の二つである.

- **開発 1**：組込み機器ノード上のアクチュエータと，ブラウザ上に表示されたボタンとで通信を行う．具体的には，LED を 2 回点滅した後に，ボタンにクリックすることによって，Raspberry Pi に付けた LED を点灯，消灯を行う．本環境による開発はブラウザ上の skycoder だけを用いた．一方，従来の開発では，組込み機器ノード上で node.js を用いた Web サーバを記述し，ブラウザ上の JavaScript プログラムからこのサーバへアクセスすることで，同じ機能を作成した.
- **開発 2**：組込み機器ノード上のセンサからデータを取得し，それをブラウザへ表示する．本環境による開発では，ブラウザ上に起動する skycoder だけで，Raspberry Pi の SPI に取り付けた光センサの値を取得する．一方，従来の開発では，Raspberry Pi 上に node.js 上で Web サーバを実現し，adc-pi-spi[53]を用いて光センサの値を取得することで，同じ機能を作成した.

本実験の結果を表 8 に示す.

表 8　開発コスト

			アプリ	サーバ	合計
開発 1	従来の開発	行数(行)	155	90	245
	本環境での開発	行数(行)	3	0	3
開発 2	従来の開発	行数(行)	144	117	261
	本環境での開発	行数(行)	2	0	2

本開発環境を用いた場合の開発 1 のコードを図 25，開発 2 のコードを図 26 に示す.

```
var domcontainer = getDomContainer();
domcontainer.innerHTML = "<input type='button' name='onled' id='onled' value='On LED'
onClick='sensornetwork(□172.16.4.105□, □gpio□, □set□, □21□, □1□);'>";
domcontainer.innerHTML += "<input type='button' name='offled' id='offled' value='Off LED'
onClick='sensornetwork(□172.16.4.105□, □gpio□, □set□, □21□, □0□);'>";
```

図 25　本開発環境で開発 1 の開発コード

```
var nodeIP = "172.16.4.66", mosi = "10", miso = "9", clk = "11", ce = "28";
sensornetwork(nodeIP,"spi", "mcp3208", [mosi,miso,clk,ce]);
```

図 26　本開発環境で開発 2 の開発コード

また，本開発環境を用いた場合の開発時間は，開発 1 が 4 分，開発 2 が 1 分であった.

4.6.2.3 RTT の計測および可視化ツールを用いた性能向上の事例

本開発環境の可視化ツールの有効性を示すために，LED 点灯を行うアプリケーションの開発を行った．研究室内から，インターネットで接続された遠隔地にあるゲートウェイおよび，ゲートウェイに接続された組込み機器ノード上の LED を点滅させるアプリケーションを，ローカルのブラウザ上で開発し，そのブラウザ上で実行するプロトタイプを作成した．そして，プログラム実行時の RTT をタイムライン可視化ツールで確認し，通信オーバヘッドを減らすために，ブラウザ上で実行していたプログラムを，ゲートウェイ上の Blue-Sky サーバ内で実行するようにプログラムを移動して，その効果を可視化ツールで確認するようにした．これらはすべて Blue-Sky の開発環境内から行った．

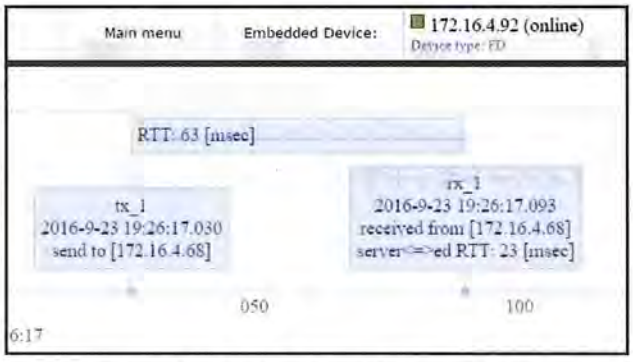


図 27　開発したブラウザ上で実行した場合の RTT 表示

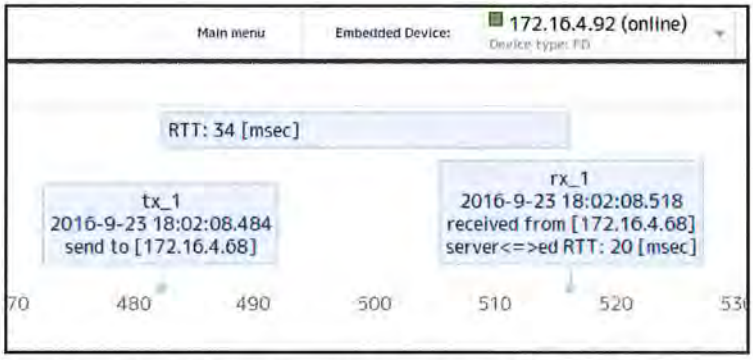


図 28　ゲートウェイ内の Blue-Sky サーバ内で実行した場合の RTT 表示

ローカルのブラウザ上で実行したときのタイムライン表示を図 27 に，Blue-Sky サーバ上で実行したときの表示を図 28 に示す．また，継続して動作させた場合の RTT の変化を図 29 に示す．ローカルのブラウザで実行した場合は RTT が 63ms であったものが，プログラムを組込み機器ノードに近いゲートウェイ

上で実行することで，34ms に低減している．

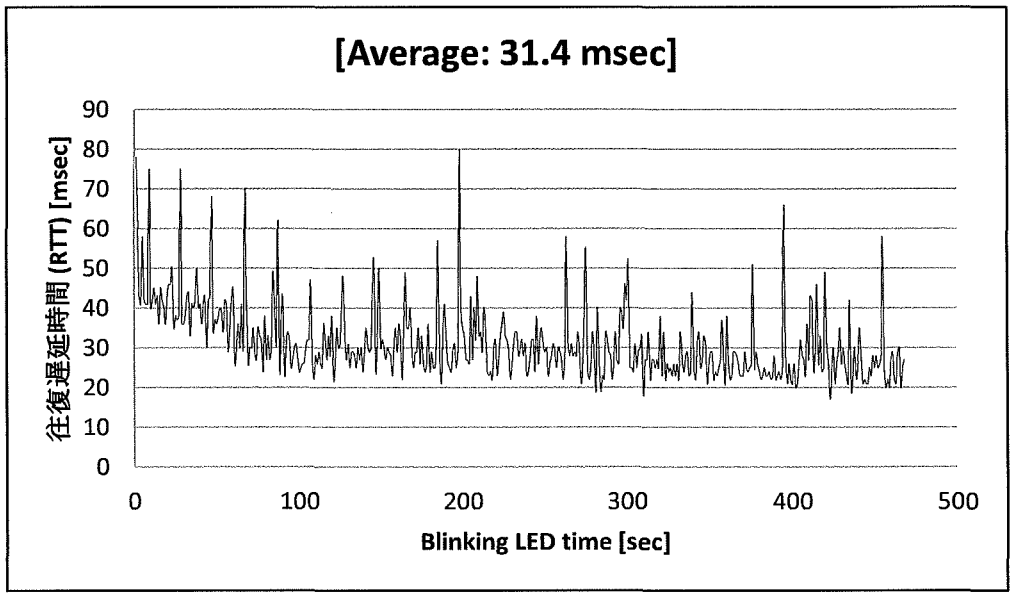


図 29　プライベートネットワーク内で LED を点滅させるアプリケーションの RTT

4.6.3 考察

本節では，前節の評価実験についての考察を行う．

4.6.2.1 では，フレームワーク自体のオーバヘッドの計測を行った．その結果，サーバは 2ms，組込み機器ノードでは 8ms であった．IoT ではインターネット接続によって各組込み機器ノードを利用することになるので，ネットワーク通信は有線環境で数 10ms，無線環境では 100ms 単位となる．本システムでは，通信オーバヘッドの高い HTTP を用いたにもかかわらず，サーバおよびノードでは小さいオーバヘッドとなり，システムのモニタリングやアプリケーション開発に対する影響は小さいことが明らかになった．要求分析において，フレームワークがモニタリングに影響を与えないという要件を満たしている．

4.6.2.2 では，開発したアプリケーションのコードサイズを，本環境を用いない場合と比較した．その結果，アプリケーション開発者は少ないコードで機能を実装することが可能になった．また，本環境では，ゲートウェイ上にサーバを用意していて，組込み機器ノード上でデバイスプログラミングを行う必要がないことから，サーバのコストがかからない点もコードサイズの削減に寄与している．また，本システムの API はハードウェアに近い形式で利用可能であることから，利用しやすいこともメリットとなっていることが明らかになった．開発時間についてはいずれも短時間であった．

4.6.2.3 では，プロトタイピングおよびシステムのタイムライン可視化によって，ネットワークの影響を低減させるための開発方法の事例について検討した．プログラムを変更せずに RTT を取得して表示することが可能であり，この環境を用いてまずブラウザ内でプロトタイプシステムをコーディング後，稼働させて計測を行い，可視化を行うことが可能である．プログラムのゲートウェイ上への移動や，再計測も同じブラウザ内で可能であり，4.6.2.2 で示したように短いコードでの開発と合わせて，プロトタイピングおよび改良が容易であることが分かる．IoT システムでは，ネットワークシステムの構成を意識したプログラム構造や動作環境が重要であり，そのための容易な計測と表示，改良が可能である本開発環境が IoT システムの開発にとって有効であることが明らかになった．また，RTT を継続的に監視した図 29 からもシステム自体も安定して利用することが可能であることが分かった．

4.7 関連研究

Aneka[54]は 2011 年に Christian Vecchiola 氏などによって，開発されたクラウドシステムである．このクラウドシステムは，ワークロードの監視， フレキシブルシケジューリングの要約，基本の VM オペレーションの管理，動的かつオープンなシステムである．システムの設計は Software as a Services (SaaS)として実装されたものであり，センサやアクチュエータに対するプログラミングを可能にする．しかしながら，多種，複数のデバイスへの対応や，センサや通信までを統合した開発環境は提供されていない点が異なる．

Xively[55]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．本開発環境では， 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[56]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．本開発環境では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．

IoT 開発フレームワークとしては，Eclipse IoT[57]がある．Eclipse IoT は，メッセージングプロトコルに MQTT を利用し，REST ベースの簡易的な HTTP プロトコルである CoAP[58]，およびノード管理のための LwM2M[59]を備えている．本システムとの共通的としては，REST ベースのプロトコルを備えている点，および Java サポートが挙げられる．一方で，本システムではブラウザベースの開発環境および可視化環境を備え，容易にプロトタイプ作成や性能測定が可能である点が異なる．さらに，通信プロトコル内部に通信状況を計測する機構を備えることで，プログラムの改変や別のツールを用いることなく，通信状態のキャプチャリングが可能になる．また，HTTP ベースの通信をベースに構築することで，ネットワーク構築を柔軟に行うことが可能である．

4.8 結論

本章では，IoT を対象とした分散組込みシステム向き Web ベース開発環境の開発について述べた．本環境は，HTTP をベースとして，センサ／アクチュエータを備える組込み機器ノード上で管理機構を稼働させ，ブラウザベースの開発環境からプログラミングやデバイスの操作を可能にした．また，組込み機器ノードの管理や，セキュリティの向上，通信状態データの保存のためにゲートウェイおよび，管理サーバを開発した．通信情報については，HTTP ヘッダ内部に保存することで，プログラムを変更せずに定常的なモニタリングが可能になった．ブラウザベースの開発環境では，各ノードの状態監視，コーディング，性能測定の機能を備え，インストールすることなく IoT プログラムの開発が可能になった．このシステムについて，フレームワーク自体のオーバヘッド，および２つの IoT アプリケーションについてのコードサイズの測定，さらに RTT の可視化ツールを用いた開発事例について評価を行い，本システムの有効性を確認した．

今後の課題としては，より多くの台数，異なる場所での動作検証，および制御を含むアプリケーションでの有効性の検証，である．現時点では，20 台の組込み機器ノードを接続した場合の動作を確認しているが，これをより多くの台数にした場合のシステムの負荷や開発の利便性について検証を行う．また，現在は同一市内にノードおよび開発環境を備えた場合の動作を検証したが，これを国外や無線環境を用いた場合の性能測定および有効性の検証を行う予定である．具体的には，タイと日本との間での接続性検証を行う予定である．今回の評価実験では，センサ／アクチュエータをそれぞれ単体で利用したプログラムであるが，実際の応用としてはフィードバックを含む制御で用いられる場合がある．このような場合における API や可視化システムの有効性について検証していく．

第５章　IoT プログラミング学習支援環境 BlueSky/Edu の開発

本章では，IoT プログラミング学習支援環境 BlueSky/Edu の開発について述べる．

5.1　まえがき

モノのインターネット化（Internet of Things，以下 IoT）は，従来の組込みシステムをネットワーク接続で接続しサーバのリソースを用いることで，「モノ」単体では提供することが難しい機能を提供できるようになる．センサからのデータを収集し，そのデータを加工，分析し，その結果をアクチュエータを通して反映していく．アメリカにおける NSF による CPS 研究支援や，ドイツにおける「インダストリー4.0」を始めとして多くの産業に関わる分野となりつつある[60]．

このような IoT の隆盛に伴い，これまでの組込みシステム教育も大きく影響を受ける．特に，従来の制御中心の教材から，ネットワーク接続された環境に対応できるようなプログラミング教育が必要となってきている．これを後押しするように，多くの学習環境や教材が利用可能になりつつある．例えば，遠隔地間のセンサについては JOSE[61]のような広域に設置したセンサとローカルのセンサやアクチュエータを組み合わせて利用できる環境が整いつつある．また，LEGO のような従来の組込みシステム教材をクラウドサービスと連携させた教育環境[62] が提供され始めている．Xively[63] や Milkcocoa[64] などの IoT 向きのクラウドサービスが安価に利用できることも，このような IoT プログラミング教育を始めやすくなってきている要因となっている．拓殖大学工学部情報工学科でも，Raspberry Pi とクラウドサービスを組み合わせた IoT の演習を３年間にわたって実施してきた．

その一方で，このような IoT プログラミング教育は，教授者と学習者の両方にとって負荷が増えることになる．従来の組込みシステムと比較して，扱うデバイスは複雑になる．また，台数が多いことから，その管理に関するトラブルは増加するが，限られた時間内に解決できなければならない．また，学習者から見ると，組込みシステムとは異なるプログラミングモデルを持つ，ネットワークプログラミングの両方を限られた時間で習熟する必要がある．このようなコストが高いことから，IoT プログラミングを中等，高等教育機関において教育するには難しい問題となっている．

5.2 目的

目的は，中等教育や高等教育機関において利用可能な，IoT プログラミングを修得させるための環境を提供することである．

このようなプログラミング環境を構築するために，我々が開発した IoT 実行支援フレームワークである Blue-Sky[65]を用いて，セットアップや保守などのマシン管理を効率化するための機構，ブラウザベースのプログラミングおよび実行環境による簡易なプログラミング環境の提供，さらにセンサ情報およびノード状態の可視化機構による容易な状態把握を可能にした．この環境を用いて実際に評価実験を行い，本システムの有効性を確認した．

5.3 問題分析

5.3.1 IoT プログラミング

IoT プログラミングにおいては，センサノードにおけるデータの取得，そのデータをネットワークを介してサーバに収集，分析，さらに，その結果を元にしたアクチュエータの操作という三つの側面がある．ここでは，センサデータへのアクセスおよびアクチュエータの駆動では，組込みシステムとして，センサおよびアクチュエータのプログラミングが必要となる，また，データ分析では，サーバサイドでのプログラミングが必要となる．さらに，これらのコンポーネントはネットワークで接続されているので，データ通信のためにネットワークプログラミングが必要となる．

センサネットワークなどと比較して，IoT の大きな特徴の一つにインターネットを介したノードの接続が挙げられる．この場合，位置的，ネットワークの状況によって，センサやアクチュエータの操作にどのような影響を与えるのかを理解し，対応できるスキルを身につける必要が出てくる．

5.3.2 IoT プログラミング教育の問題点

前節で示した IoT プログラミング教育を行う上での問題点は次の 3 点である.

(1) クラスルームにおけるセットアップコストの増大

(2) 動作の不可視性

(3) 手軽に利用可能な開発環境の不足

第１に，演習室や教室などのクラスルームにおいてマシンやネットワークをセットアップするコストの増大がある．実際に，我々がパブリッククラウドを利用した組込みシステム演習を行った経験では，ネットワーク接続し，センサやアクチュエータなどのデバイスを接続できるマシンを数十台セットアップするにはかなりの時間と手間がかかる．これは，従来の組込みシステムに加えて，ネットワーク接続することからマシンのケーブリングが煩雑になる．また，機器の導入や追加が順次行われることから，教室全体でマシンの種類が揃わず，ポートや仕様の違いがトラブルを引き起こすこともあった．センサなどのデバイスを直接扱うことから，物理的な故障や破壊を含むハードウェアのトラブルも多く，マシンの交換においてプログラムの修正に手間がかかることになる．さらに，開発用のマシンとデバイス動作用のマシンを分けて扱う場合，このセットアップコストはさらに増えることになる．

また，ネットワーク構成を柔軟に設定できなければならない．クラスルームにおいて数十台のマシンが常に外部のサービスを利用できるネットワーク環境を備えているわけではない．この場合でも，クラス内で演習が可能な環境を構築できる必要がある．また，インターネット接続可能な環境の場合は，外部からの侵入や攻撃を防ぎ，教室内のマシンをセキュアにしておく必要がある．さらに，クラスルーム内においてもアドレス間違いなどによるセンサやアクチュエータの意図しない動作を防ぐ仕組みが必要である．

第２に，動作の不可視性に対する対応が必要となる点である．IoT プログラミングではネットワークを扱うことは不可欠である．従来の組込みシステムの教育であれば，デバイス自体の動作によってプログラムの結果を理解することはできた．しかし，ネットワークを介して接続することによって，その部分が作成したシステムにどのように影響するかを理解することは難しい．特に，遠隔地間でのセンサやアクチュエータの利用では，通信や処理プログラムの遅延がシステム全体に影響を与えることから，その動作がよりわかりにくくなって

いる．また，教授者側にとっては，数十台のマシン状態の把握や，プログラミングの動作結果のトレースが必要となる．通常の演習と比較して，動作の再現が難しく，トレーサビリティが低いことから，デバッグのサポートがより難しくなっている．

第３に，手軽に利用可能な開発環境の不足が挙げられる．組込みシステムへの要求の増大によって，開発環境の複雑さは増加し，短時間での習熟が難しくなってきている．また，開発環境のインストール時間や要求するリソースも増えていることから，用意すべきマシンへの要求が高くなっているのが現状である．さらに，PC で開発する場合は，ターゲットマシンへの遠隔ログインやコピーなどの操作が必要となり，ログ取得やデバッグなどのコストが高くなってきている．

5.4 設計方針

前節に上げた問題に対応するために，我々はこれまで開発した IoT プログラム実行支援フレームワークである BlueSky を利用して，IoT プログラミング教育をサポートするシステムを開発した．これを BlueSky/Edu と呼ぶ．BlueSky/Edu の設計方針は次のとおりである．

5.4.1 ブラウザベースプログラミング環境

BlueSky/Edu では，ブラウザベースのプログラミング環境を提供して，セットアップせずにプログラミングや実行を可能にする．今日，多くのシステムでは Webkit をベースとしたブラウザが動作可能である．そこで，ユーザが触れる開発環境をブラウザ上で動作させることにより，マシンへのセットアップをせずに，プログラミング環境を利用可能にする．これによって，セットアップコストを低減させることに加え，マシンのトラブルによる交換時に，ユーザデータのコピーや再設定のコストを減らす事が可能となる．

また，プログラミングについては，従来のプログラミング言語によるプログラミングに加えて，コマンドラインベースによるステップバイステップの操作が可能な環境を用意した．これは，IoT プログラミングでは従来の組込みシステムと比較して構成要素が増えていることから，問題発生時に原因の特定が難しくなっている．この問題に対して，コマンドラインベースでセンサやアクチュエータを駆動することで，問題の切り分けを容易にする．

サポートするプログラミング言語は，ブラウザ上で動作可能な JavaScript

をベースにして，インストールのコストを低減させている．システム呼出しのAPI を切り分けることにより，言語ごとのブリッジを用意することで教材に沿った言語をサポートすることも可能である．

5.4.2 ブラウザベース可視化環境の提供

IoT におけるプログラミングにおいて重要となる，センサおよびネットワークの二つに対して，ブラウザベースの可視化環境を用意する．センサについては，センサのデータ遷移をタイムラインベースで可視化可能にして，センサの値の変化の把握を容易にする．ネットワークについては，通信状態を可視化できるようにして，通信コストがプログラムの実行に与える影響を見えるようにタイムラインベースの可視化環境を用意する．可視化に必要となるデータ取得は，プログラムを改変せずに可能なように，プロトコルベースでサポートすることにより，実行後に実行データの分析や教授者への相談，対応が容易になる．また，マシンの各状態はグラフィカルにを把握できるようにすることで，故障時の切り分けや対応が容易になる．

5.4.3 ノードの管理および実行機構の提供

プログラム実行のモデルとして，ブラウザおよびノード上でプログラムを実行し，センサやアクチュエータを搭載したノードをプログラムから操作するモデルとする．このモデルでは，プログラムの自由度は制限されるが，プログラムが理解しやすいこと，ノードの交換に対応しやすいこと，さらに，遠隔地に設置したセンサ／アクチュエータの操作と同じプログラミングモデルで利用可能な点から，このモデルと採用する．プログラムの実行場所を変化できるように，ブラウザ上での実行だけではなく，ノード上へのプログラムのデプロイと実行も可能とする．

各センサノードはプログラムからは仮想的なノードとして扱う．これによって，デバイスの交換や，ハードウェアの差異の吸収を容易にする．

システムは，プライベートネットワーク内だけでも動作可能とすることで，ネットワーク接続に問題のある環境でも実行を可能とする．さらに，センサやアクチュエータへのアクセスに対しては，システムレベルで認証を行うことにより，クラスルーム内での誤操作やいたずらなどによるトラブルを低減させている．

5.5 設計

本節は，システムの設計について述べる．

5.5.1 全体構成

全体構成を図 30 に示す．BlueSky/Edu は，センサやアクチュエータが搭載されたデバイスノード群と，それらを接続するクラスネットワーク，さらに，クラスネットワークと外部接続およびクラス内ネットワークサービスを提供するゲートウェイから構成する．

デバイスノードでは，センサおよびアクチュエータを接続し，IoT の「モノ」の部分を提供している．デバイスノード上では，BlueSky/Edu のノード管理プログラムである NodeManager が動作し，ノードの状況の管理，およびセンサ／アクチュエータへの中継，アクセス制御を行う．

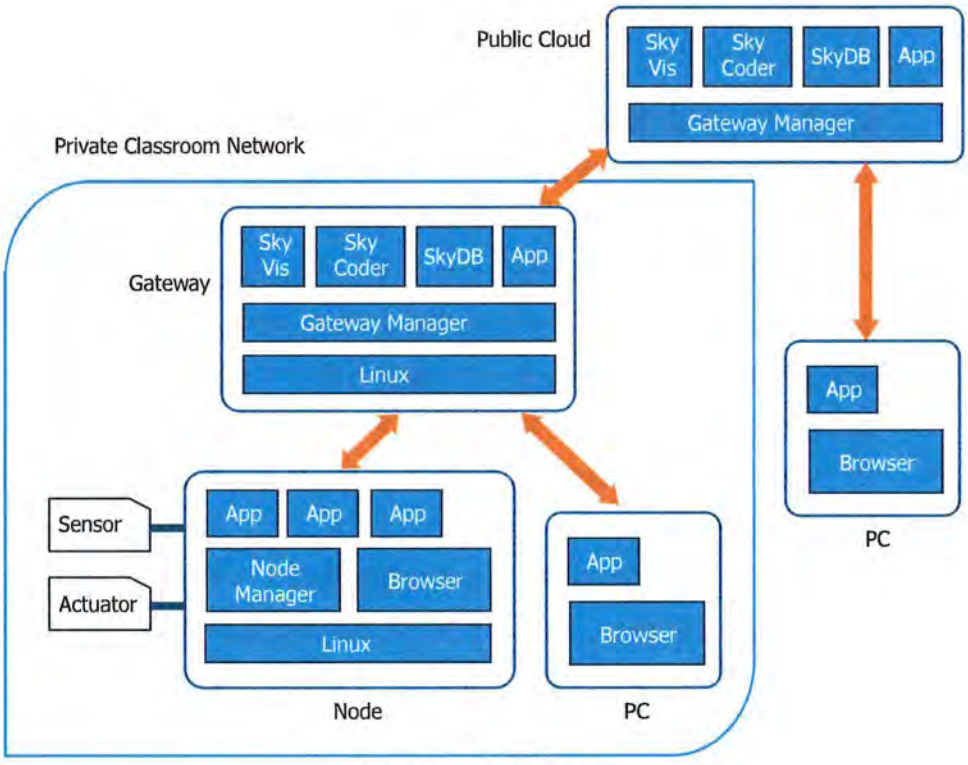


図 30　全体構成図

ゲートウェイでは，インターネット接続のためのルータ，およびセキュアにするためのファイアウォール，デバイスノードからの情報収集機構を Gateway Manager によって提供する．さらに，BlueSky/Edu の各種サービスを提供するためのサーバとしても動作可能である．これによって，インターネット接続が難しい，もしくは帯域幅の狭い環境でもプログラミング環境を提供することができる．

プログラミングおよびプログラム実行は，PC もしくはノード上で起動するブラウザで行う．ノード上のブラウザでプログラミングすることで，クラス内で必要となるマシン数を減らすことが可能であり，教授者の管理の手間を減らすことができる．より快適なプログラミングを行えるように，PC 上のブラウザを用いることも可能である．

BlueSky/Edu では，ノードおよびゲートウェイ上で動作するプログラム，およびクラウドサーバ上で動作するサービス群から構成されている．主なサービスとしては次のものがある．

- SkyVis: 可視化機構
- SkyCoder: プログラミング環境
- SkyDB: センサデータベース

これらのサービスは，ゲートウェイ上とクラウドサーバのどちらでも動作させることが可能である．ネットワークリソースが豊富で，現場でのサーバ管理コストを減らしたい環境では，これらのサービスをパブリッククラウドサーバ上で提供することを選択できる．一方，ネットワーク帯域幅が狭く，クラス全体からの外部サービスへのアクセスに問題が生じる場合は，ゲートウェイ上ですべてのサービスを提供して，外部への通信量を減らす選択をすることができる．

ノードおよびゲートウェイ間の通信には，接続性の高い HTTP をベースとして用いている．プロキシ環境下にあることの多い教育機関でも，外部のセンサノードとの接続性を保証することができる．また，通信時のオーバヘッド測定を自動で行うために，HTTP ヘッダにトレース情報を保持した形で通信を行う．トレースデータは，BlueSky が提供するデータベース上に保存し，可視化を行う際に利用することができる．さらに，RESTful な API を用意して，プログラムから簡単に操作することが可能である．

システムは Java で記述することで，Linux および Java が動作する環境であ

れば，同じように利用できる．

5.5.2 NodeManager：ノード管理機構

NodeManager は各ノード上で動作するノード管理機構である．機能としては，次のものを提供する．

- ポートマッピング機構
- ボードのコンフィグレーション管理
- ボードのシステム状態，ネットワーク状況の送信
- ノードへのアクセス制御

クラスルームでは，ノードの故障による交換や種類の異なるボードの利用が避けられない．Raspberry Pi でもバージョンによってポートの位置が異なることがあり，交換によって，配線しなおしやプログラムの作り直しが生じることがある．これに対して，ポートを仮想化し，従来の接続先をできる限り維持したまま，ボードの交換を可能にする．ノードの種類に応じたコンフィグレーションを保持することで，対応するポート番号の変更を容易に可能にしている．

ネットワーク接続やボード状況のシステムから収集し，定期的にゲートウェイに送信している．ゲートウェイはこれを収集して，後述する可視化環境によってノードの状況を表示する．

また，サーバサイドの JavaScript 実行環境である node.js を用いて，NodeManage に JavaScript のプログラムをデプロイして，起動することが可能である．学習者は，遠隔でプログラムを動作させることが可能となり，プログラムの動作場所やネットワーク帯域幅による，実行結果の違いを容易に実験することができる．

各ノードは，ローカルなネットワーク内部ではアクセスが可能である．IP アドレスの指定間違いやいたずらによって学習が妨害されてしまう可能性がある．そこで，ノードのセンサへのやりとりにはクッキーベースの軽量なアクセス制御機構を提供している．クッキーを持つマシンからのアクセスに限定することで，学習者が利用しているセンサ／アクチュエータの専有や共有を容易に制御することが可能である．

5.5.3 SkyCoder：ブラウザベースプログラミング環境

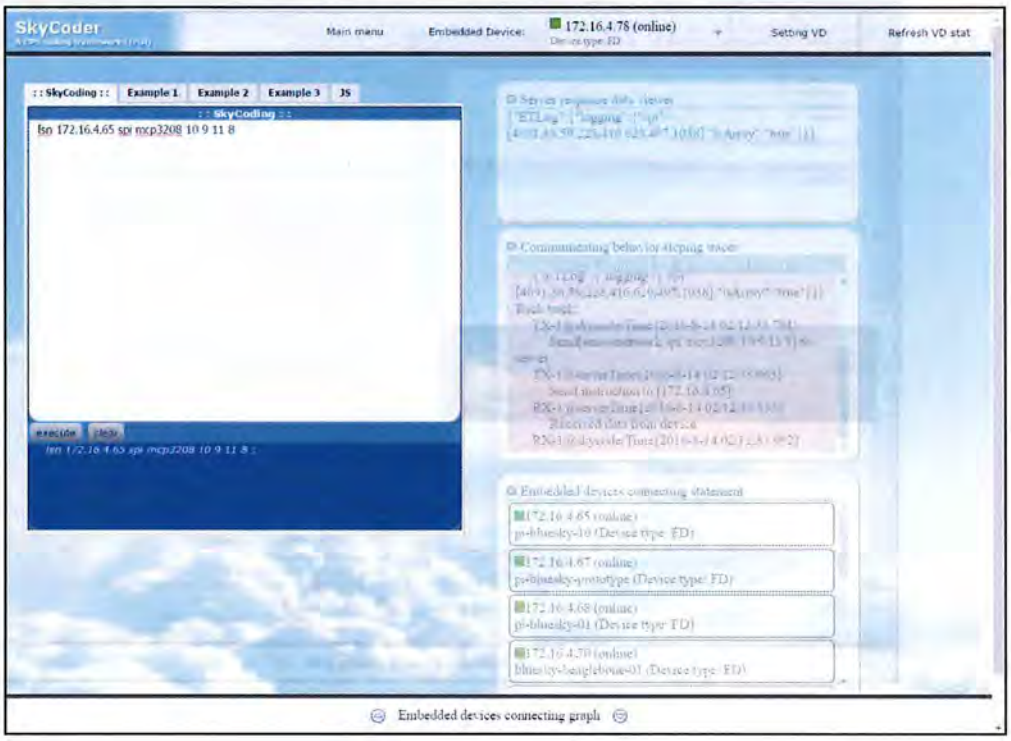


図 31　SkyCoder の実行例

SkyCoder はブラウザだけで利用可能なプログラミング環境である．ブラウザだけでプログラミングを行うことができる．ブラウザ上でシンプルなエディタを用いて，ノードへの動作をプログラミングすることができる．図 31 に SkyCoder の実行例を示す．ブラウザ内の左側にエディタ，右側にデバイスの状態を表示している．エディタではコマンドラインベースで，コマンドおよび IP アドレス，デバイス種類，ポート，データを指定することで，ステップバイステップでセンサやアクチュエータに対する操作を行う．IoT 環境では，学習者の手元にデバイスだけではなく，遠隔先のセンサやアクチュエータを操作することが必要になるが，コマンドラインベースで，動作を確認しながら操作を進めていくことができる．提供するコマンドとしては，ローカルネットワーク内のセンサに対する操作を行う lsn コマンドと，ゲートウェイの先のパブリックネットワーク上に設置されたセンサ／アクチュエータへのアクセスを行う gsn コマンドを提供する．また，サーバとノードとの通信内容のデータビューアーやトレーサを常に表示している．これによって，プログラム内で表示を追

加することなく，通信内容をモニタリングすることができる．

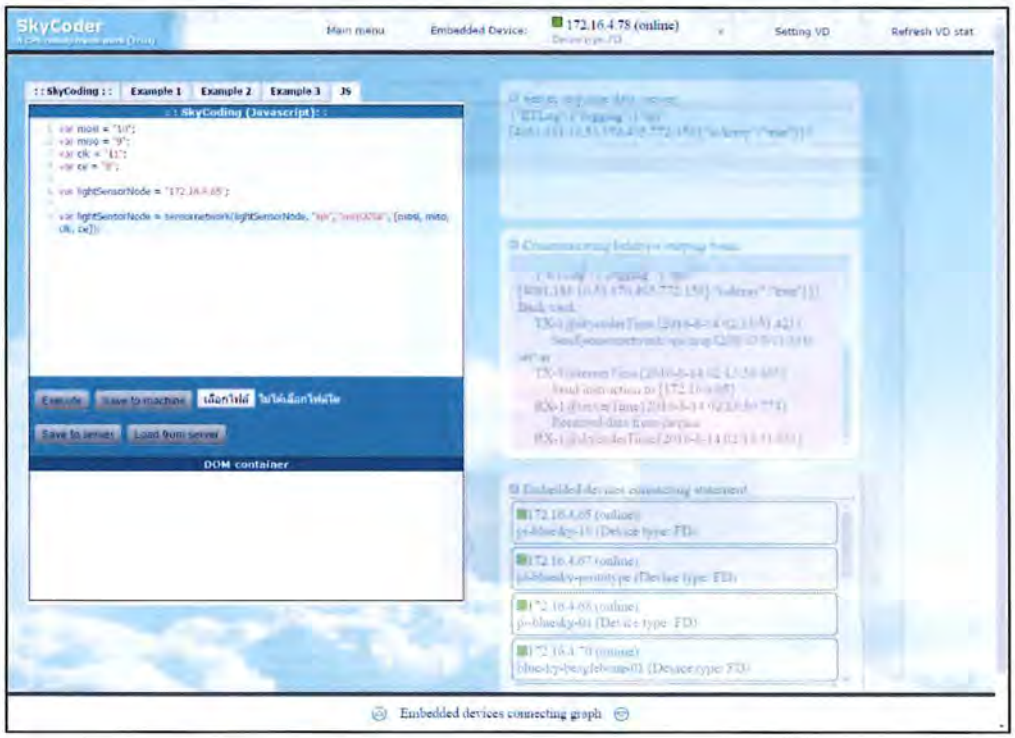


図 32　JavaScript のプログラミング例

図 32 に SkyCoder での JavaScript のプログラミング例を示す．同じウィンドウの別のタブで JavaScript のプログラムを作成して，実行することができる．センサ／アクチュエータ操作を行う BlueSky API を通して，プログラミングを行う．センサからのデータを収集して，JavaScript プログラム内で分析し，その結果を基にアクチュエータを操作するという，IoT の基本的な構成を容易に試すことが可能になる．

この JavaScript のプログラムは，エディタ内から実行場所を指定することが可能である．ブラウザ内で実行もしくは，サーバ，ノードなど複数の環境で動作させることができる．これによって，遠隔地の設置されたセンサなどを用いる場合，実行場所を変更することで動作の影響を確認することが可能になる．この結果については，後述の SkyVis で可視化して確認することが容易にできる．

```
var deviceIP ="172.16.4.68";
var pin =21;
for ( var i = 0; i < 100; i ++){
        sensornetwork ( deviceIP , " gpio ", " set ", pin , 1);
        Sleep (1000);
        sensornetwork ( deviceIP , " gpio ", " set ", pin , 0);
        Sleep (1000);
}
```

図 33　LED の点滅プログラム例

JavaScript で LED を点滅させるプログラム例を図 33 に示す．遠隔ノード上のデバイス呼出しはすべて sensornetwork()関数を用いる．ターゲットのホスト名（IP アドレス），デバイスの種類，入出力方向，ピン番号，出力値（もしくはなし）を指定することで，デバイスへのアクセスを可能にする．

5.5.4 SkyVis:可視化機構

SkyVis は，学習者に現在動作しているシステム状態を分かりやすく表示する可視化機構である．SkyVis では，ノード管理および学習支援のために次の可視化機構を提供している．

- ノード状態の可視化
- タイムラインベースの通信トレーサ
- センサデータグラフ

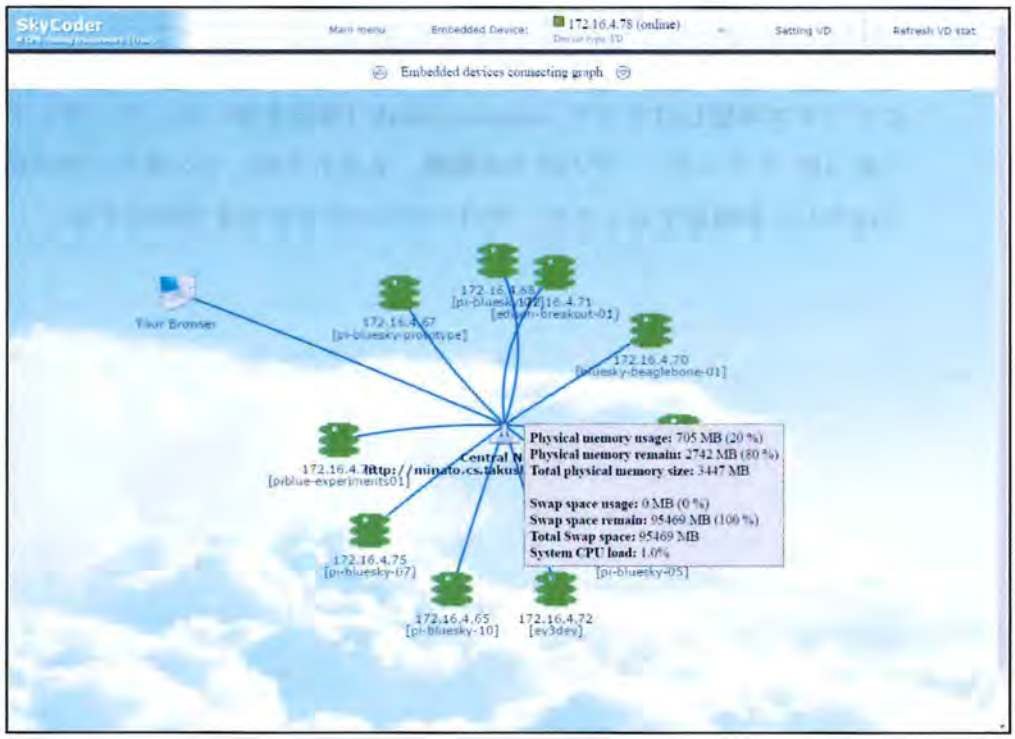


図 34　NodeViewer の実行例

図 34 にノード状態の可視化ツールを示す．ノード可視化は，現在接続されているノードをシステムの状態とともに表示する．ノードの種類が異なるものについてはアイコンを変更し，システム状態をポップアップでシステム状態を表示することで，複数のノードの状態を一度に確認することが可能である．ノードのレイアウトは自動で決定されるので，ノード数が増えた場合でも表示レイアウトを調整する必要がない．これによって，数十台のマシンを管理する必要がある教授者にとって，ノードの状態を容易に確認することができる．また，学習者は他のノードのセンサの利用をする場合に，センサの状態や種類を

確認して，プログラミングすることができる．

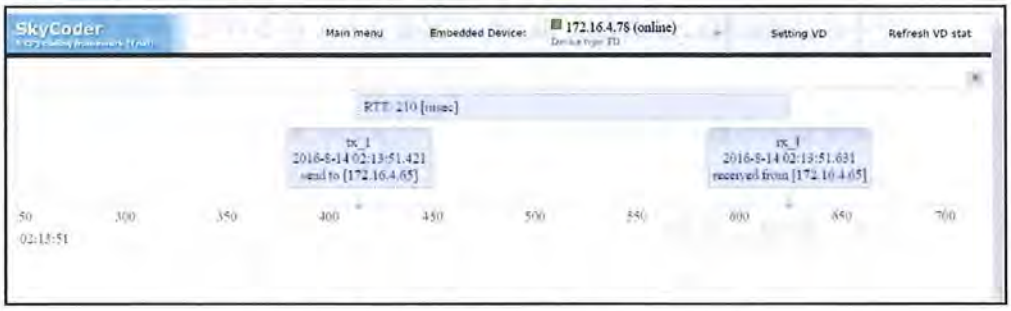


図 35　タイムラインベースの通信トレーサ

図 35 にタイムラインベースの通信トレーサを示す．横軸に経過時間を取り，ブラウザからリクエストが送られて，ノードで処理され，戻ってくるまでの時間をグラフ化する．図 35 では，タイのバンコクから東京にあるセンサノードへの通信例である．tx 1 が 1 回目のリクエスト，rx 1 がそれに対応するレスポンスを表し，その間の時間 Round Trip Time(以下，RTT)である 210ms を上部に表示している．このようにタイムラインベースでグラフ化することで，帯域幅や設置されたセンサの場所などによって，どのように通信状態が変わるかを容易に確認して，プログラムに反映させることが可能になる．

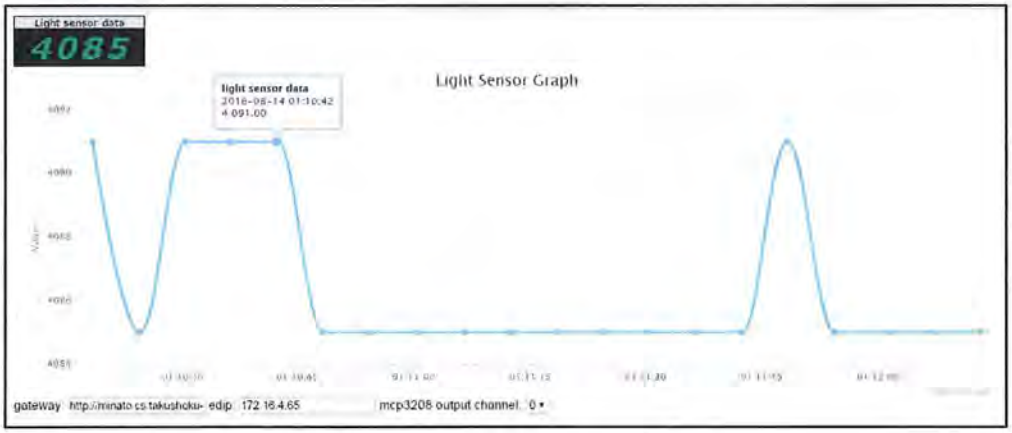


図 36　センサグラフの実行例

図 36 にセンサのデータグラフを示す．これも通信と同様に，横軸に経過時間，縦軸にセンサ値を取り，それによってセンサのデータがどのように変わるかを示している．センサ値の時系列の変化を見ることができるとともに，詳細な情報をポップアップで表示することができる．また，ノードやデバイスの種

類を指定することで，複数のセンサに対応させることが可能になる．センサデータは通信時に自動的に SkyDB に保存されているので，学習者はログ取得を明示することなく，センサ値の可視化が可能になる．教授者にとっても，実行結果を後から追う事が可能になり，デバッグの支援に役に立てることができる．

5.6 実現と評価

本節では，実現と評価実験について述べる．

5.6.1 実現環境

本システムは Linux 上で実現している．すでに対応しているノードは，Raspberry Pi 2/3, KZM-A9, Beagle-BoneBlack, Intel Edison である．また，対応する入出力の種類としては，GPIO, SPI, PWM が利用可能であり，多くのデバイスを利用することができる．ゲートウェイについては，Ubuntu Server を用いている．システム全体は Java で記述することにより，JVM が動作するマシンであれば容易に稼働させることが可能である．現在 20 台の RaspberryPi のノードを接続，実行できることを確認した．

5.6.2 評価

本節では，BlueSky/Edu の評価について述べる．実際の BlueSky/Edu を利用した実験を行い，システムの有効性を検証する．

5.6.2.1 評価実験

本学情報工学科の 4 年生 5 名に対して，次の二つの課題を出した．

(1) 遠隔ノード上の GPIO に接続された 1 つの LED を 1 秒ごとに点滅する JavaScript プログラムを作成する．

(2) PWM API を用いて，LED を消灯状態から点灯状態までをゆっくりと点灯させる JavaScript プログラムを作成する．

(3) IoT の基本的なアプリケーションの開発課題を作成する．本課題は 16 戸の部屋の内に，消された一つのランプを教えてくれる IoT アプリケーションである．

この課題について，作成した行数と作成時間を記録した．対象となる被験者群は，基本的な組込みシステムに関する知識はあるが，JavaScript で組込みシステムを制御した経験はない．

また，システムの使いやすさおよび分かりやすさの 2 点について，5 段階の

スコアをつけてもらった.

5.6.2.2 評価結果

課題(1)の行数と作成時間に関する結果を図 37 に，課題(2)の結果を図 38 に，課題(3)の結果を図 39 示す.

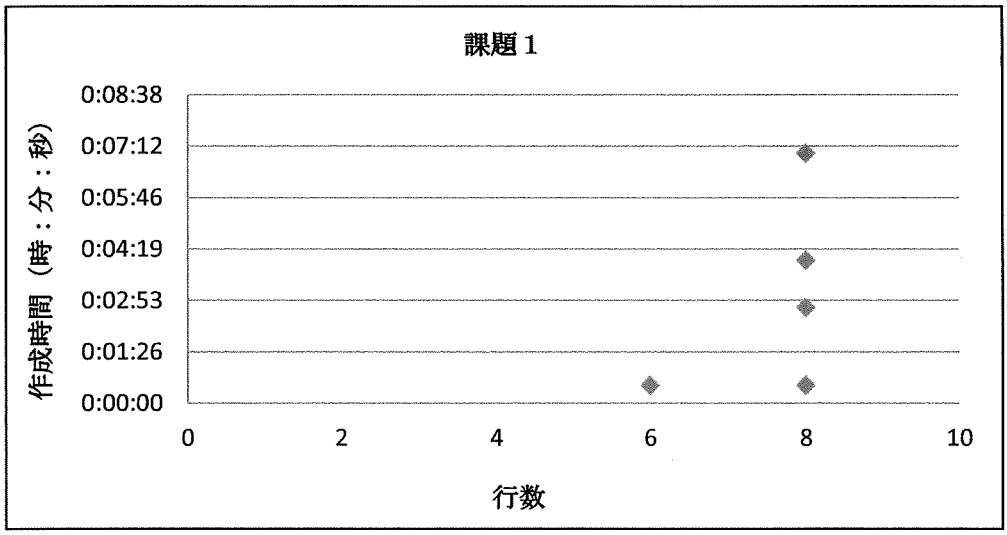


図 37 課題 (1) の実験結果

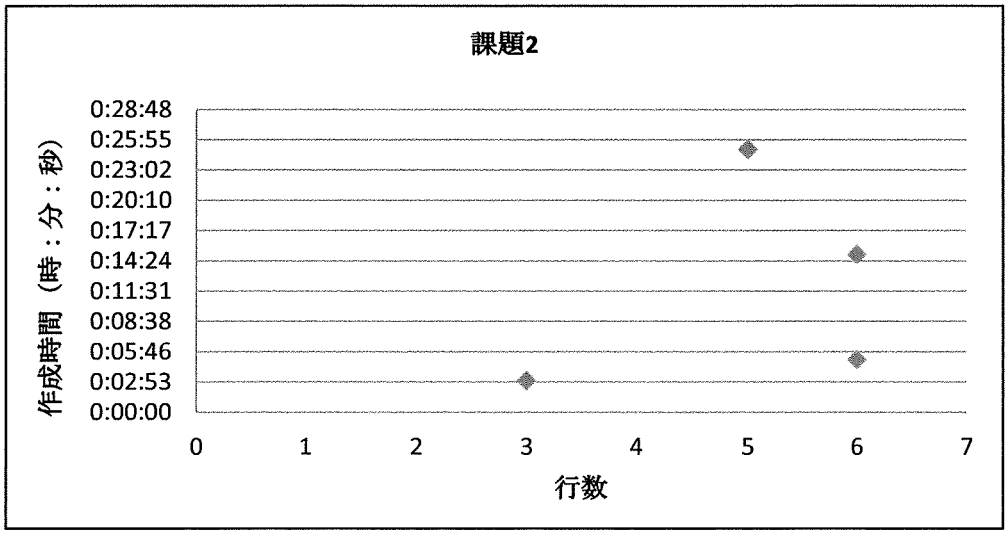


図 38 課題 (2) の実験結果

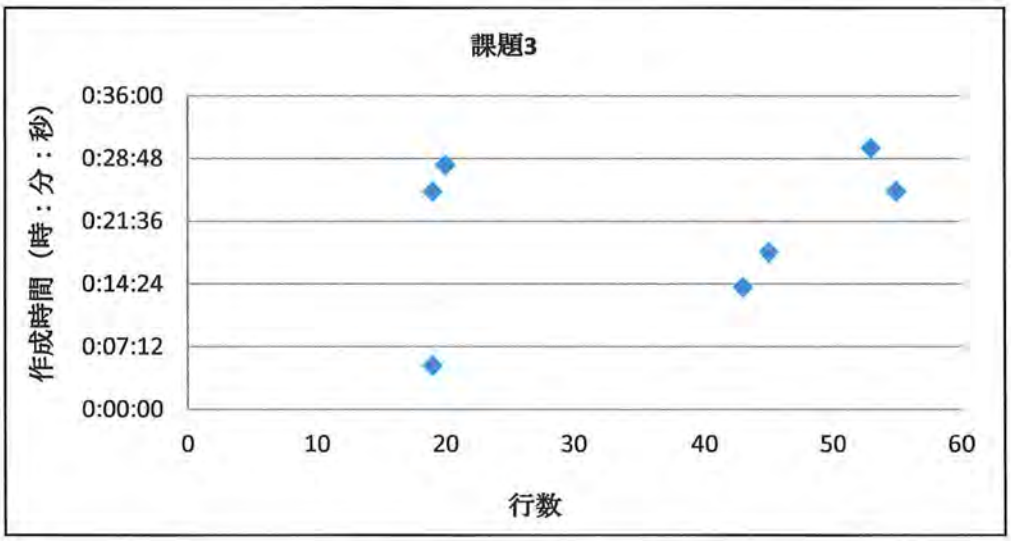
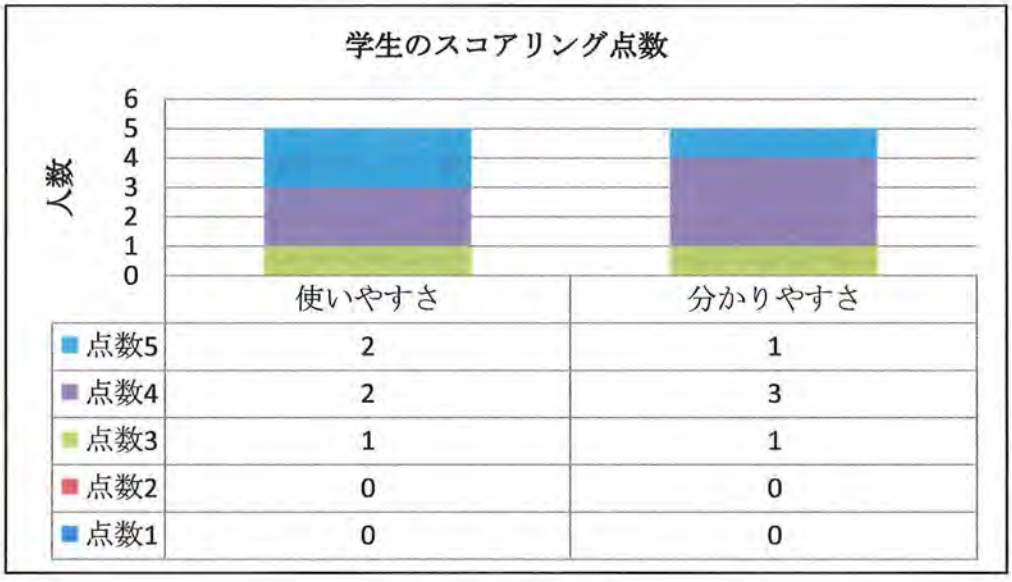


図 39　課題（3）の実験結果



	使いやすさ	分かりやすさ
点数5	2	1
点数4	2	3
点数3	1	1
点数2	0	0
点数1	0	0

図 40　アンケート結果

評価は横軸に作成した行数，縦軸に作成時間をを取ってプロットした．課題(2)で被験者数が 1 名少ないのは，作成時間の取得ができなかったからである．

課題(1)ではほとんどの被験者が 8 行で遠隔の LED 点灯のプログラムを作成することができた．作成時間に関しては早い学生は 30 秒程度で，遅い学生でも 8 分以内にプログラムを作成することができた．

課題(2)については多くの被験者が6行程度で作成している．作成時間の分散は大きく，最大で26分程度，最小で3分弱で開発している．

課題(3)については被験者が最大で55行，最小で19行のIoTの基本的なアプリケーションを，30分以内の短時間に開発することができた．行数の乱れは被験者のコードを見ると，配列の宣言方法や配列の扱い方が異なるわけである．

アンケート結果を図40に示す．すべてのスコアが3以上であり，4，5が大半であった．

5.6.2.3 考察

これらの実験結果から，本システムの有効性について考察する．課題(1)では，ほとんどの被験者が短い時間で完了している．LEDを点灯させる課題は容易で理解しやすいが，IoTプログラミングのようにネットワークを用いた場合，通信部分を作成する必要があり，コード行数が増加しがちである．それに対して，本システムでは遠隔でデバイスを操作するAPIを提供して，通信部分を記述せずにデバイスを操作することが可能である．これによって，短い時間と短い行数でコードを完成させることが可能になった．APIは通信部分を隠蔽している一方で，ノードのデバイス部分については過度の抽象化を行わず，ハードウェアと対応しやすくしている．このことから，APIの使い方に習熟しなくても利用可能であったと思われる．これは，短い授業時間内で動作させる必要ある，クラスルームでは有効であると思われる．

課題(2)については，PWMの概念の理解度に差があり，完成時間には大きな差が生じたと思われるが，その一方，行数はほぼ同等であり，到達度の差は生じていない．

課題3については被験者が短時間に基本的なIoTのアプリケーションを開発することが可能なので，本開発環境がIoTプログラミング学習支援に適用であると明らかになった．

アンケート結果については，使いやすさおよび分かりやすさについて，いずれも高いスコアとなっている．これは，被験者群が日常使い慣れているWebブラウザ上で利用可能な形態にしたことで，プログラミング環境の使い方で戸惑うことがなかったことを表しているものと思われる．

5.7 関連研究

IoT が産業として重要になりつつあることから，その教育環境についても研究が行われている．[66]は，IoT デバイスおよび IoT ゲートウェイ上でアプリケーションを実行でき，ゲートウェイを介して外部のクラウドサーバに接続可能なシステムのプロトタイプについて提案している．システム構成は BlueSky/Edu と同じであり，ゲートウェイおよびノードでのアプリケーション実行についても同等の機能を提供している．その一方で，教育現場におけるマシン管理やセットアップコスト，さらにプログラミング環境セットアップなどの手間についてはサポートされていない．BlueSky/Edu ではこの部分をブラウザベースの環境によってサポートすることで，教授者および学習者のコストを下げることを目的としている．

[67]は，ネットワークに関する M2M と IoT アプリケーションを対象として，IoT のシステムソフトウェアモデルのそのソリューションに関して述べている．サービス統一のために，HTTP および JSON を用いて，リアクティブプログラミングモデルを利用している．リアクティブプログラミングは大規模プログラミングでは有効な手法であるが，初学者が用いるにはコード記述やモデルの点で難しいものと思われる．BlueSky/Edu では基本的な JavaScript の構文および，単純な API を提供することで，モデル学習にかかる時間を短縮し，動くシステムに触れることができる点が有益である．

[68]は，ETSI M2M の上に、バスのトラッキングシステムを開発する IoT/M2M アプリケーションのプロトタイプシステムである．センサノードで HTTP サーバを起動し，直接 HTTP POST と GET メソッドを送信して，センサデータを扱うことができる構造となっている．RESTful なシステム構成を採用している点は，本システムと同等であるが，プログラミング環境を提供している点が大きく異なっている．このシステムでは Android アプリケーションを対象としていることから，教育現場で開発環境とは別に Android が動作するマシンを用意しなければならないという問題も生じる．

[69] は，ZigBee を用いたワイヤレスセンサネットワーク上で IoT 技術を体験するものである．オープンソースとなっていて，多様なデバイスに対応している点が利点である．一方，教育現場では，無線環境は必ずしもうまく動作しないことがある．また，Arduino をベースとして環境を構築しているが，IoT 環境ではネットワークおよびソフトウェアのウェイトが高いことから，より高速，高性能な実行環境を提供すべきである．Arduino の開発環境はシンプルで

利用しやすいが，セットアップコストは以前として存在する．

5.8 結論

本章では，IoT プログラミング学習を支援する環境として開発した BlueSky/Edu の設計および評価について述べた．セットアップや保守などのマシン管理を効率化するためのノード管理機構，ブラウザベースのプログラミングおよび実行環境を用いた，理解しやすいプログラミング環境の提供，さらにセンサ情報，通信情報およびノード状態の可視化機構による容易な状態把握を可能な環境を提供した．この環境を大学生 5 名に対してプログラミングおよびアンケートによる評価を行い，短時間で IoT デバイスのプログラミングが可能であることを明らかにした．また，アンケートのスコアもの用いて実際に評価実験を行い，使いやすさが 5 段階で平均 4.2，分かりやすさが平均 4 と高評価となった．この結果，本システムが教育環境として有効であることが明らかになった．

第６章　異種プロトコルサポートによる多様なデバイスアクセス環境の構築

本章では，異種プロトコルサポートによる多様なデバイスアクセス環境の構築について述べる．

6.1 まえがき

近年，CPS[2]や IoT[70]といった，ネットワークに接続された複数の組込みデバイスを用いてセンサデータの取得や加工，物理世界の操作などを行うシステムが数多く登場している．これに伴い，組込みシステムおよびネットワークシステムの両方に精通したエンジニアの育成[14] [1]が必要とされている．特に，従来のシンプルなセンサやアクチュエータだけではなく，複数のセンサが搭載されたセンサボードや，ロボットといったデバイスを遠隔から操作することも行われている．

このような状況に対して，拓殖大学工学部早川研究室では，分散組込みシステムの教育や開発を対象としたフレームワークである Blue-Sky[48]の開発を行っている．Blue-Sky は，IoT を主な対象として，Linux が動作する組込み機器のマシン管理および監視システムと，通信状態やマシンの監視を行うゲートウェイ，および Web ブラウザベースの開発環境を提供することで，インストールのコストを下げつつ，多くのマシンの操作および管理を可能にするフレームワークである．これを用いて，演習室などでの IoT の教育環境[48]およびシステム開発を行うことを想定している．

しかし，Blue-Sky では，組込み機器に搭載するデバイスは，GPIO や SPI といった低レベルなインタフェースへアクセスする API だけを提供している．これは，当初のターゲットデバイスが Raspberry Pi などの CPU ボードであり，シンプルなデバイスだけを対象としていたためである．しかし，最近の IoT システムでは，センサボードやロボットなど高機能かつ複雑なデバイスプログラミングを必要とするデバイスを扱うことも増えてきている．このことから，Blue-Sky においても，このようなデバイスに対応する必要が出てきた．

このような問題に対して，我々はロボットアプリケーション開発のフレームワークである ROS[71][72]に着目した．ROS は，ハードウェア抽象化機構，デバイスドライバ，ライブラリ群，可視化ツール，pub/sub の基づいたメッセージ通信，およびパッケージ管理を提供し，複数コンポーネントにおけるアプリ

ケーション開発をサポートしているオープンソースプロジェクトである．現在，多くのデバイスベンダが ROS 対応のライブラリを公開していることから，デバイスに関する複雑なプログラミングを行うことなく，センサやアクチュエータ，さらにはロボットなどを利用することができる．

そこで，本研究の目的は，IoT の学習者を対象として，Blue-Sky から ROS 対応の各種デバイスを扱えるように，システムを拡張することである．ROS と Blue-Sky では通信のモデルが異なることから，これらの違いを吸収する機構を設計実装した．また，ROS および Blue-Sky のノードが混在する環境において，Blue-Sky が提供する監視および可視化システムを利用できるようにすることで，Blue-Sky の開発環境内でシステム全体での動作を理解しやすくすることが可能になった．

6.2 ROS

ROS は，ロボットソフトウェアの開発を目的としたフレームワークである．ROS は，コンピュテーションを行うプロセスである各ノードを XML-RPC[73]ベースにおける publish/subscribe (pub/sub)通信することによって動作する．

本研究で対象とするROSのコンポーネントは次の三つである．

(1) **プロジェクトの作成ツール群**

作成ツール群はCMakeマクロのビルドパッケージであるcatkin[74]のコマンド群によってワークスペースや複数階層で作成可能なパッケージとして構成される．ROS パッケージはこのマクロ群で宣言されたパッケージライブラリを指定することによって catkin によるビルドが可能である．これらのツール群やライブラリはシステム開発を行う OS 上にインストールする必要がある．

(2) **ノードの可視化ツール**

可視化ツールとしては rqt_graph[75]を提供している．ROS のノード[72]は実行されているプロセスのことを定義している．さらに，ROS はモジュール化されていて，ノードはソフトウェアのモジュールや実行自体そのものとすることも可能である．これで，rqt_graph は各 ROS のコンピューテションというノードの状態や通信間の繋がりを可視するプラグインである．

(3) **pub/sub の通信を処理する publisher**

ROS は，roscore[76]と呼ぶ基本的なプログラムやノードの集合である．ノード間で通信する場合には，roscore は ROS のパッケージで作成されたプログラムやノードのデータを publish/subscribe(以下，“pub/sub”)のメッセージング方式で収集し，“/rosout”[77]のトピックでログを publish する機能を持つ．ちなみに，pub/sub 通信モデルは，非同期モデルでありメッセージの送信者である publisher は受信者である subscriber の情報を持たずにメッセージを送る．subscriber は，publisher の情報を持たずに，受信対象となるクラスだけを指定し，そのクラスのメッセージだけを受け取る．このため，送信者と受信者とは疎結合であり，スケーラブルなシステムを作ることができる．一つの ROS の基に作成されたすべてのプログラム群やノードの状態は，roscore から取得することが可能である．

6.3 問題分析

我々の開発したフレームワークである Blue-Sky を ROS に適用する場合の問題点は次のとおりである.

(1) **Blue-Sky と ROS とでの提供する機能の違い:**

ROS は, 各 OS に属する言語のライブラリやそのビルドシステムまで統合されたシステムである. 一方, Blue-Sky は, サーバが提供する Blue-Sky API を用いてデバイスの機能を呼び出すことで, アプリケーションを構築し, 実行する. ROS のロボットアプリケーションから Blue-Sky の機能をアクセス可能にするには, 低オーバヘッドでこの二つの仕組みを繋ぐ必要がある.

(2) **Blue-Sky と ROS との通信:**

ROS のノードの状態を取得する場合, roscore との通信が必要となる. roscore との通信は, pub/sub 通信モデルによるが, Blue-Sky は HTTP による通信を基本としていることから, この二つの通信モデルを変換する必要がある. Blue-Sky の開発環境は, ブラウザ上で JSON ベースで通信を行うために, ROS がベースにしている XML-RPC のデータ形式をそのまま扱うことができない. また, Blue-Sky の開発環境[48]は, JavaScript を利用したブラウザアプリケーションなので, roscore の XML データ形式を利用することが難しい.

(3) **Blue-Sky 上での ROS の可視化のための拡張:**

ROS は rqt_graph と呼ぶ可視化ツールを備えているが, 単一のツールであり, システムのインストールや設定が必要であり, 複数台数を管理しなければいけない教室環境においては管理コストが上がる. また, 複数のツールをまたいで利用することは, 利用者の利便性を阻害することになる.

6.4 設計方針

前節に述べた問題分析に対して，我々が開発した Blue-Sky の環境を ROS に適用可能に拡張することで対応する．拡張する部分は次の２点である．

(1) **Blue-Sky の開発ライブラリの ROS 拡張:**

Blue-Sky で組んだ IoT アプリケーションから ROS のノードが提供するデバイスに低オーバヘッドでアクセスできるようにする．これにより，Blue-Sky から ROS が提供する高水準な API を備えたデバイスを利用することが可能になる．

(2) **Blue-Sky の可視ツールの ROS 可視ツール拡張:**

Blue-Sky の Web ベース開発環境でブラウザ上に Blue-Sky のノードと ROS のノードの両方を可視化可能にする．これにより，開発者や学習者は各ノードの状態や通信を監視することができる．IoT では，多くの機器の状態の監視や，通信オーバヘッドによるプログラムでの対応などが必要になることから，そのような課題を容易に行えるようにする．

6.5 設計

6.5.1 全体構成

図 41 にシステムの全体構成を示す．本報告では，利用者は Blue-Sky の開発環境およびノードを利用し，さらにそこから ROS が動作するノードを利用することが可能になる．Blue-Sky のサーバは，ROS と通信をするために，Multi-purpose Handler と呼ぶプロトコル変換の機構を実装した．この機構を用いて，roscore のトピックをハンドルすることで，ROS ノードのデータを取得することが可能になる．

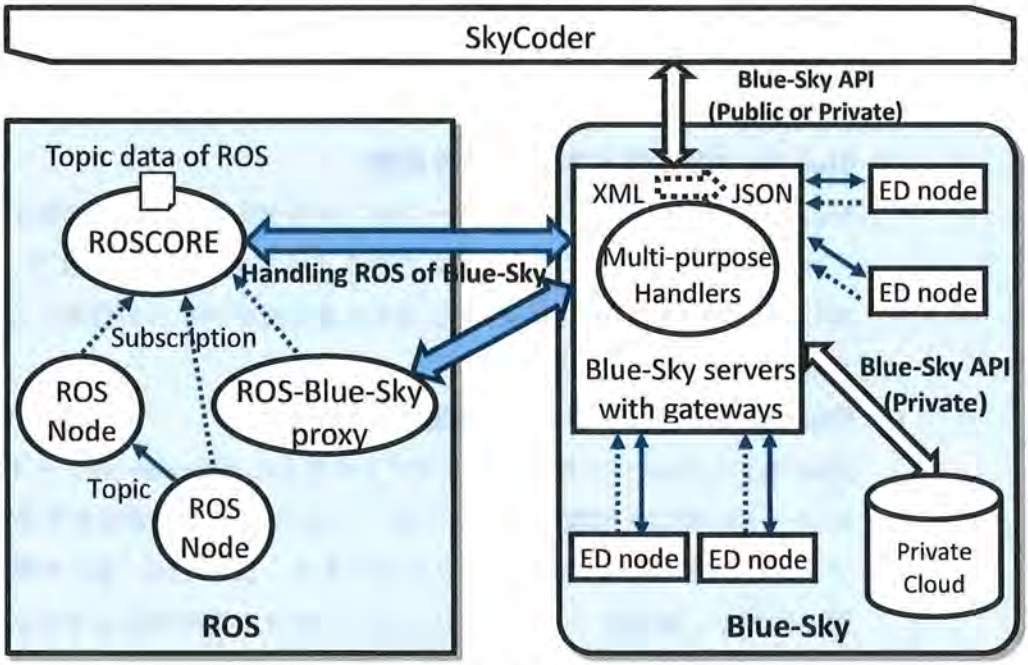


図 41　Blue-SkyのROS拡張モデル

そして，ROS ノードの状態は本システムの skycoder[48]で可視化する．可視化データは JSON 形式にするため，Blue-Sky サーバの Multi-purpose Handlers で，XML-RPC を JSON-RPC[78]に変換する機能を追加することで行う．

6.5.2 ROS-Blue-Sky プロキシ

Blue-Sky から，ROS の各ノードを呼び出すために，roscore が動作するシステム上で Blue-Sky との通信を行うプロキシを起動する．プロキシは次の役割を持つ．

- Blue-Sky からのトピックアクセス要求を ROS の各ノードへの通信に置き換える
- ROS から生成されたデータを Blue-Sky gateway に中継する
- roscore に対して，各ノードの操作を依頼する
- ノード名の管理
- 通信オーバヘッドの測定

プロキシは，ROS のノードの一つとして扱われている．このような設計にすることで，他のノードやライブラリを変更することなく，その機能を利用することができる．また，プロキシは，pub/sub 通信の時間情報を取得し，それを Blue-Sky へ送信することでノードの通信オーバヘッドを測定する．

```
# Flashing LED and getting the light
# sensor data for 10 times.
loop 10
lsn 172.16.4.103 gpio set 22 1    # ED nodeの利用
sleep 300
lsn 172.16.4.222 get a 3 0    # ED nodeの利用
sleep 600
lsn ros://node0/onLed   # ROSノードの利用
sleep 600
lsn ros://node0/offLed   # ROSノードの利用
sleep 600
end
```

図 42　コマンドラインによる ROS ノードへのアクセス例

```
/**
 * Flashing LED and getting the light sensor data
 * for 10 times with API oriented function.
 */
for(var i = 0; i < 10; i++) {
  l_sensornetwork("172.16.4.103", "gpio", "set", "22", "1");
  Sleep(300);
  l_sensornetwork("172.16.4.222", "get", "a", "3", "0");
  Sleep(600);
  l_sensornetwork("ros://node0/onLed");
  Sleep(600);
  l_sensornetwork("ros://node0/offLed");
  Sleep(600);
}
```

図 43　JavaScript による ROS ノードへのアクセス例

図 42 は Blue-Sky から ROS ノードを利用する例である．skycoder のコマンドラインから利用する場合，ros://ノード名/トピック名で，ros のノードとトピックを指定する．例では，node0 において onLed と offLed というトピックがすでに定義されている場合は，Blue-Sky の ED ノードと同じアクセスメソッドによってアクセスすることができる．skycoder で JavaScript を用いたプログラムの例を図 43 に示す．これも，Blue-Sky のノードと同じメソッドによってアクセスすることが可能である．

6.5.3 roscore からの情報取得機能

roscore から通信に関する情報取得を行うために，Blue-Sky サーバ上に Multi-purpose Handlers と呼ぶハンドラを提供する．本機能は roscore の

XML-RPC のメソッドを Blue-Sky サーバから呼び出すために用いる.

```
<?xml version='1.0'?><methodCall>
<methodName>getSystemState</methodName>
<params><param>
<value><string>/rosnode</string></value>
</param></params></methodCall>
```

図 44　Multi-purpose Handlers に追加した roscore に対する
ハンドリング RPC の呼び出しメソッド

```
XML-RPC response data:
<?xml version='1.0'?>
<methodResponse><params><param><value>
<array><data><value><int>1</int></value>
<value><string>current system state</string>
</value>...</param></params></methodResponse>

converted to JSON-RPC
{ "ETLog" :{ "ros" :{ "jsonrpc" :{ "jsonrpc" :
 "2.0" , "result" :{{ "params" :{ "param" :
{ "value" :{ "array" :{ "data" :{ "value" :[{ "int" :1},
{ "string" : "current system state" }...}}}}}
```

図 45　応答データを JSON-RPC に変換する

図 44 に ROS で用いる XML-RPC の呼出しメソッド例を示す. このデータを本サーバのハンドリングで roscore に渡し, ROS のノード XML データを取得する. そして, 図 45 のように JSON のログに変換し, Blue-Sky 上のストレージへ格納したり, skycoder に提供する.

6.6 ROS ノード可視ツールの設計

ROS ノードについても, Blue-Sky の ED ノードと同じように可視化可能なように, skycoder 上の可視化ツールを拡張した. ROS のノードをブラウザ上で可視化するには, 本システムの Multi-purpose Handlers によって変換された JSON 形式の ROS ノードデータを取得し, 本システムで扱っている可視化ライブラリである visjs[51]の dataset[79]にマッピングした後に Network[80]可視化モジュールで表示する. マッピングオーバヘッドを減らすために, ROS ノードが起動されないか roscore が起動していない場合は可視化の処理をしないように設計する.

これによって， ROS ノードに対しても，従来の可視化ツールのように開発環境上にインストールする必要なく利用することが可能になった．

6.7 評価

本節は実現および評価について述べる．

表 9　ROS 実験環境

System Name	Specification	Unit
ROS	・ indigo	1
GUEST OS	・ Ubuntu 14.04.3 LTS	1
HYPERVISOR	・ VMware Player v. “7.1.2”	1
NATIVE OS	・ Ubuntu 12.04.5 LTS	1
	・ Intel(R) Core(TM) i7-2600 CPU@ 3.40GHz.	

表 9 に本システムの実装実験における ROS 実験環境を示す．ROS は indigo を仮想マシンにインストールし，次の二つのワークスペースを作成する．

- ワークスペース１： nodename のパッケージを作成し nodename というノード名を初期化する std_msg の標準出力にメッセージを送信するアプリケーション
- ワークスペース２： rosjava のパッケージを作成し，デフォルトに提供されている Talker と Listener のアプリケーション

roslunch や roscore を起動させて，各ワークスペースのアプリケーションを起動する．ここで，図 46 のように rqt_graph で ROS ノードを表示する．

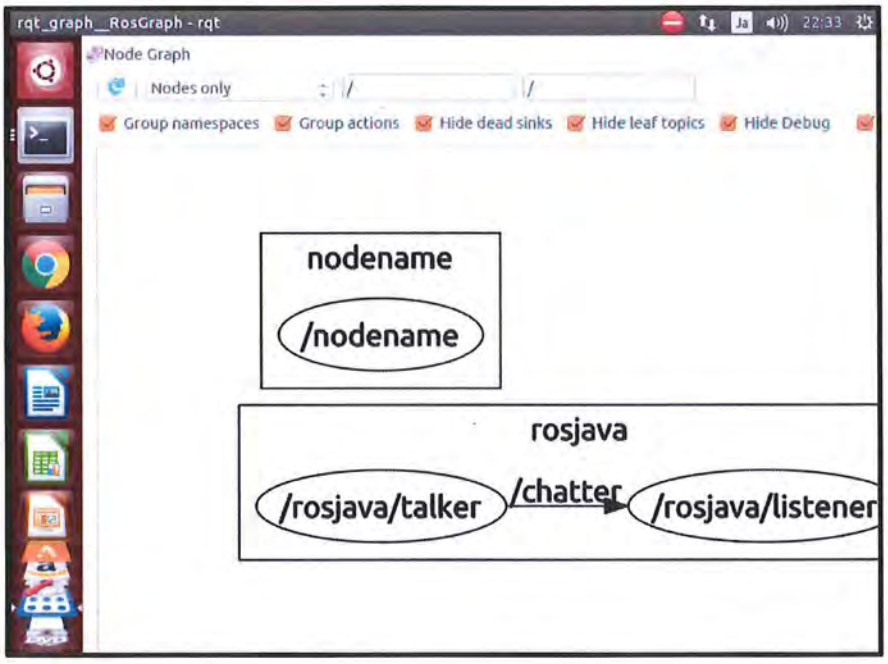


図 46　ROS の rqt_graph

表 10 Blue-Sky 実験環境

System Name		Specification	Unit
Skycoder	Browser	・Google Chrome v. “48.0.2564.103 m”	1
	OS	・Windows 7	
		・Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz	
Blue-Sky servers	JAVA	・java version “1.7.0_55” OpenJDK Runtime Environment(IcedTea 2.4.7)	2
	OS	・Ubuntu 13.10	
		・AMD Athlon(tm) II X2 260 Processor	
Raspberry Pi	JAVA	・java version “1.8.0”	4
		・Java(TM) SE Runtime Environment	
	OS	・Debian GNU/Linux 7.8 (wheezy)	
		・Arm (BCM2708)	
Intel Edison	JAVA	・java version “1.8.0_25”	1
		・Java(TM) SE Runtime	
	OS	・Linux 3.10.17-poky-edison+	
		・Genuine Intel(R) CPU 4000 @ 500MHz	
KZM-A9	Platform	・Android v. “2.2”	1
		・Linux Kernel v. “2.6.29”	

そして，表 10 に示したように Blue-Sky のシステムを起動し，ワークスペース 2 のアプリケーションの初期化を行った後，本拡張ライブラリを用いてアプリケーションを作成する．

(1) Blue-Sky の ROS 拡張ライブラリを ROS のパッケージにおける“lib”に入れる.

(2) 表 10 表に示したように 100Base-TX で接続した Raspberry Pi 上で動作する Blue-Sky の組込み機器ノード上の LED を 2 秒ずつに点滅するアプリケーションを作成する.

これによって, 図 48 のように物理的な組込み機器の LED が 2 秒ずつに点滅することで動作を確認した. さらに, 本システムの skycoder 上に図 47 はワークスペース 1 とワークスペース 2 で作成した ROS のノードを表示することが可能であった.

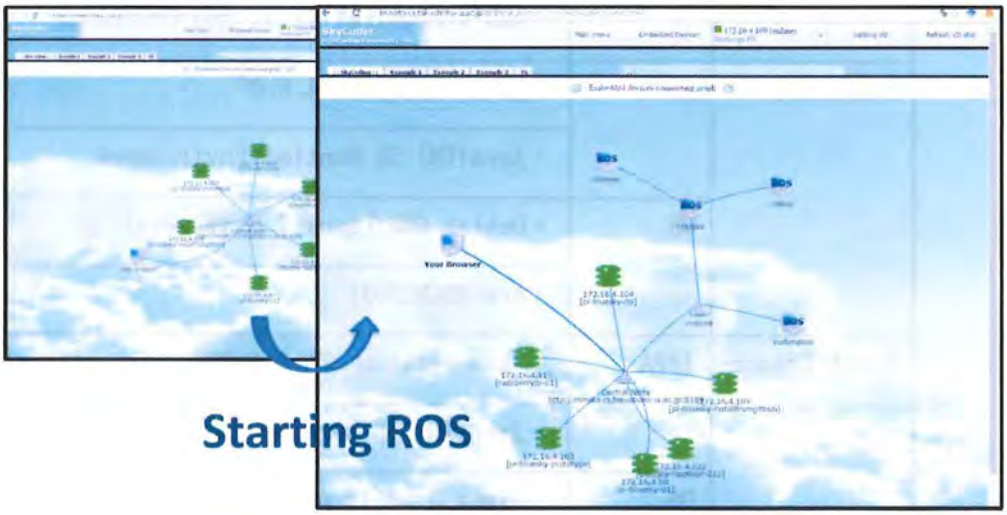


図 47　Blue-Sky の skycoder で ROS ノードの可視

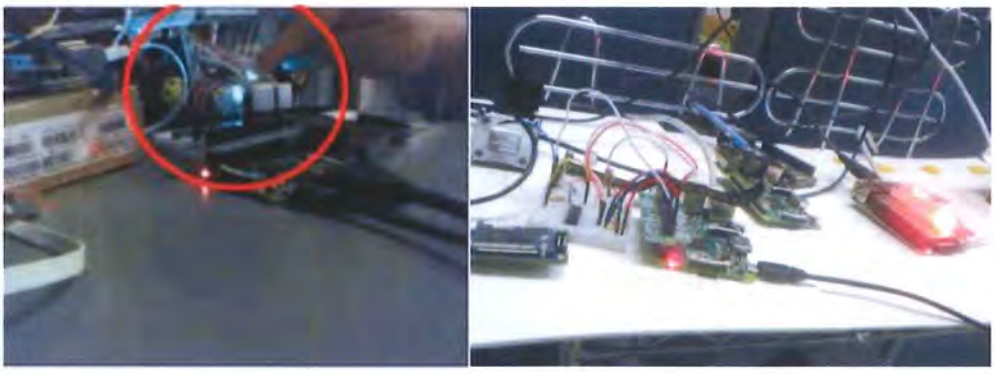


図 48　物理的な組込み機器のノード

8 分間実行した場合の RTT は，図 49 のようになった．本拡張ライブラリの RTT は平均 19.897 ミリ秒であり，下限値は 12～53 ミリ秒と小さい値である．このことから，拡張ライブラリとしては十分な性能を持っているものと思われる．拡張ライブラリとして有効であると考える．

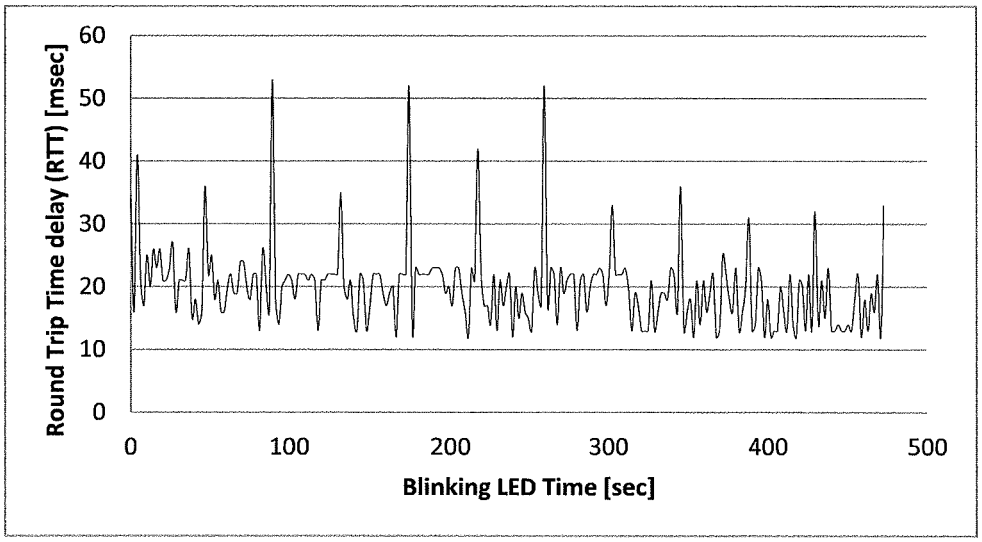


図 49　Blue-Sky の ROS 拡張ライブラリアクセスの往復遅延時間

さらに，ローカルなマシンで rqt_graph を利用不可能な ROS の開発環境においても，本可視ツールを利用することによって，インストールせずに ROS ノードを起動するだけで，ROS ノードと本システムのノードの同時に容易にトレースや監視を行うことが可能になった．これは遠隔地に設置されたデバイスの監視や，教室のように多くのデバイスを扱う環境では有効であると考えられる．その一方，各ノードの通信状態の表示は不十分であり，この部分には改善が必要であることが明らかになった．

6.8 関連研究

ROS[71][72]の rqt_graph[75]は OS 上の GUI で ROS ノードを可視することで，そこで開発した ROS アプリケーションにおける roscore[76]のトピックを解析し，ノードとして表示するものである．

遠隔地に設置されたデバイスなどのようにディスプレイを持たない環境では，rqt_graph で可視することが難しい．遠隔地のロボットやセンサ類を操作することで，IoT の効果を学ばせたい場合にはこのような機能では不十分である．リモートデスクトップ[81]を用いることも可能であるが，台数が増えた場合には管理が難しいことや，データ転送に関する負荷や遅延が高い．

さらに，rqt_graph はネットワークに接続するノードを想定していないため，ネットワーク間に実施されるノード間の通信に関する往復遅延時間の測定を含めて，トレースや監視が不可能である．これに対して，本システムでは，通信時間を含めた測定や，ノードの監視が可能であり，組込みシステムとネットワークの両面を理解する必要がある IoT の学習環境としては有効であると考えられる．

6.9 結論

本章は分散組込みシステム向き Web ベース開発環境 Blue-Sky の ROS 拡張の方式と実装について述べた．Blue-Sky に対して，ROS 上のデバイスを扱うための拡張を行い，ROS の状態の可視化機能を追加した．これによって，Blue-Sky が備える Web ブラウザベースの開発／実行環境上で，ROS と Blue-Sky のノードの操作や，状態監視，通信のトレースを透過的に行うことが可能になった．

第7章 結言

本章では，成果，結論を述べた後，今後の課題を述べる．

7.1 成果

本研究は，“Blue-Sky”という IoT を指向したシステム開発環境を開発し評価した．Blue-Sky は組み込み機器に対して Linux ベース IoT である．本環境は次のような設備を用意した．

(1) 組込み機器におけるカーネル空間に圧縮を備えた ftrace 機構を用意した．これは，組込み機器のカーネルにコンテキストスイッチへの影響を抑えづつ，トレースの開始やトレースの停止は実行 API による実行機能を用意したことで，ブラウザから ftrace の開始や ftrace の停止を実行することが可能になった．従来の ftrace と本提案の ftrace 機構と比較する評価を行った．結果として，ユーザ空間へのプロセス切替えによる割込みの頻度や割込みマスクの回数を減らしつつ，生成されるトレースデータのサイズを削減させたことで，システムのスケジューリングやネットワークの帯域などの影響を低減させつつトレースすることを可能にしたことから，システムにおけるトレース機構として有効であることが明らかになった．

(2) IoT 向き Web ベース開発環境の開発は，各組込み機器において，組込み機器側に Linux のインターフェーズである SPI や GPIO などを制御する Blue-Sky モジュールを備え，これらへの実行 API を備えたことで，ウェブブラウザからセンシング・アクチュエーションを実行することが可能になった．さらに各実行 API に応じる HTTP の応答ヘッダにタイムスタンプを押したことで，IoT に重要なシステムオーバヘッドである往復遅延時間の測定が容易になった．そして，ブラウザ上ではコマンドラインベーススクリプトと JavaScript のコーディングエディタや組込み機器の接続状態の監視や通信シーケンスの監視およびこのもののタイムラインで可視化を備えた．これで，センシング・アクチュエーションのコーディングは，各組込み機器のノードに対して直接に触れずに簡潔に組んで，実行，そして往復遅延時間の測定によって，IoT の性能を把握することが可能になった．そして，フレームワーク自体のオーバヘッド，および実際に 2 つの IoT アプリケーションの開発についてのコードサイズ

の測定，さらに RTT の可視化ツールを用いた開発事例について評価を行い，本システムの有効性を確認した．

(3) BlueSky/Edu: IoT プログラミング学習支援クラスは IoT 向き Web ベース開発環境を適用し，IoT プログラミング学習支援クラスを立ち上げて，拓殖大学工学部の 4 年生の協力を依頼して IoT プログラミング学習支援に関する実験を行いました．IoT の実用的な課題を 3 つに設定して JavaScript における IoT のプログラミングを組んで，分かりやすさや使いやすさをアンケートで取って，IoT に関する理解度とシステムの利便性を評価した．評価結果は本システムの利便性が高い点数となり，IoT に関する理解度も高い点数なので，本システムは IoT プログラミング学習支援クラスを適用する可能であると明らかになった．

(4) 異種プロトコルサポートによる多様なデバイスアクセス環境の構築は，多くのベンダが利用する ROS　工場や高レベルのロボットに搭載する複数のセンサ・アクチュエータを制御するようになりつづ IoT を対応するために，Blue-Sky の ROS 拡張の方式を設計し実装した．これで，ウェブベース開発環境からでも簡潔に ROS のノードの接続状態の可視化，ノードの操作，通信のトレースを透過的に行うことが可能になった．

7.2 結論

サービスを融合する IoT を指向したシステム開発環境を開発した．本開発環境の開発を通ったことで，IoT の本質が把握できるようになった．サービスを融合する IoT の本質は，組込みシステムをはじめ，幅広い分野に応じる本質に大きく影響を与えることが明らかになった．特に，ネットワークに繋がらない従来の組込みシステム分野にも，組込みシステムの教育の奥から産業業界まで大きく影響を与えている．

本開発環境の IoT は，組込みシステムに応じる ftrace を基づいて，ネットワークに繋ぐ新たなロギング機構を通って，OS のタスクやプロセスやコンテキストスイッチやネットワーク通信にかかるオーバヘッドを考慮し，IoT の本質を圧迫しないような機構である．そして，OS のユーザ空間に Blue-Sky モジュールにおいて OS のシステムインターフェースを制御する．そして，ゲートウェイについては Blue-Sky サーバ群によるデバイスの仮想化での管理や接続性，ネットワークに応じる通信の構成，通信遅延によるシステム動作全体の考慮や，対象としたアプリケーションの本質に応じるプログラミング手法の革

新が挙げられる．

これによって，著者は IoT の実本質が理解できた．IoT は単一なシステムではなく，組込みシステム，ネットワーク，アプリケーションやソフトウェアの開発，そしてプログラミング手法をはじめ，幅広い分野に応じてお互いに影響を与える．システムを融合するには，各システム自体のオーバヘッドの検討が不可欠であるが，異なるシステムを一つに融合する際にオーバヘッドのトレードオフは避けられない．例えば，ゲートウェイであるサーバを中継して，仮想化に見えるようなデバイスの管理機能があるとしても，異なるネットワーク構想において，往復遅延時間が生じたことで，センシング・アクチュエーションの実行 IoT のアプリケーションの動作が異なるオーバヘッドのトレードオフや，従来の IoT のアプリケーションの開発において，各デバイスに動作されているモジュールは，デバイスの仕様によって IoT のアプリケーションの動作が異なるオーバヘッドのトレードオフや，開発環境によってアプリケーションの開発時間やアプリケーションのコードのサイズが高い，言語によって連動するアプリケーションの動作が変動したりする遅延時間のオーバヘッドのトレードオフが挙げられる．

このようなオーバヘッドのトレードオフ問題を解決するには，可能な限りオーバヘッドのトレードオフの差分を抑えることである．融合されたシステムに応じて，各システム自体にオーバヘッドがあり，そのオーバヘッドを各システムと調整し，十分に抑える有効性を検証することによって，システム全体のオーバヘッドのトレードオフの差分を抑えることができると明らかになった．例えば，コードのサイズとネットワークの構想におけるトレードオフのオーバヘッドは，可能な限り短時間に IoT のアプリケーションを開発して，実際に動作させ，ネットワークの負荷が高い回線であれば，異なるネットワーク回線に移動させることで，ネットワークの負荷を十分に抑えられる．なので，IoT における産業改革 4.0 は，単一な IoT システムを改革する意味ではなく，IT の基本教育をはじめ，IT 業界の各分野の御協力に応じてサービスやシステムを融合する IoT は起動が可能なように改革することである．

7.3 今後の課題

今後の課題は，ウェブブラウザベースのプログラミング環境に関してより多くの台数，異なる場所での動作検証，および制御を含むアプリケーションでの有効性の検証，である．現時点では，20 台の組込み機器ノードを接続した場合の動作を確認しているが，これをより多くの台数にした場合のシステムの負荷や開発の利便性について検証を行う．また，現在は同一市内にノードおよび開発環境を備えた場合の動作を検証したが，これを国外や無線環境を用いた場合の性能測定および有効性の検証を行う予定である．具体的には，タイと日本との間での接続性検証を行う予定である．今回の評価実験では，センサ／アクチュエータをそれぞれ単体で利用したプログラムであるが，実際の応用としてはフィードバックを含む制御で用いられる場合がある．このような場合における API や可視化システムの有効性について検証していく．

謝　辞

拓殖大学大学院電子情報工学専攻の博士課程の研究を進めたことに亘って，以下の通りの方々からの応援と助言と御指導をいただいた．心から感謝の意を表す．

早川栄一教授には，多くの助言や御指導をいただいたことで，大変お世話になった．ここに心から感謝の意を表す．

拓殖大学の教授に助言・コメントをいただいたことで，大変お世話になった．ここに心から感謝の意を表す．

日本の文学部省の基づいた産業技術大学院大学(AIIT)で実施した“Education Network for Practical Information Technologies ”(EnPiT)のコースを心から感謝の意を表す．そこで，コラボレートに社会人学生との経験の交換が可能であった場なので，立派なチャンスであった．

そして，以下の通りの研究会や学会の方々には，御コメントや御意見をいただいた．ここに深く感謝の意を表す．

1. Embedded System Symposium (ESS) は大学の方々だけではなく多くの企業の研究所の研究員の参加によって，そこで学生達とのアイディアの交換や多くコメントをいただいき，研究を進めたことで，ここに心から感謝の意を表す．
2. 組込みシステム研究発表会 (ETNET) は参加された方々のコメントと質疑に応じて研究を進めたことで，心から感謝の意を表す．
3. IEEE Information Communication Technologies International Student Project Conference (ICT-ISPC)は多くのコメントや意見をいただき，研究を進めたことで，心から感謝の意を表す．

早川研究室の学生の方々には，ミーティングおよび研究室の話しかけることにおいて研究を進めたことで，多く有益な助言をいただき，研究の参考になった．そして，IoT 学習支援教室の有効性を検証するための実験を御協力していただいた学生の方々にもここに深く感謝の意を表す．

最後に，拓殖大学工学事務所の方々には，多くの応援や一言についていただいたので，心から感謝の意を表す．

参 考 文 献

[1] R. R. Rajkumar, I. Lee, L. Sha and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Proceedings of the 47th Design Automation Conference*, 2010.

[2] L. A. Edward, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008.

[3] J. P. Eckert and J. Mauchly, "Outline of Plans for Development of Electronic Computers," [Online]. Available: http://archive.computerhistory.org/resources/access/text/2010/08/102660910-05-01-acc.pdf. [Accessed 01 06 2015].

[4] H. H. Goldstine and A. Goldstine, "The Electronic Numerical Integrator and Computer (ENIAC)," *IEEE Annals of the History of Computing,* vol. 18, no. 1, pp. 10-16, 1996.

[5] K. D. Kim and P. R. Kumar, "Cyber-Physical Systems: A Perspective at the Centennial," *Proceedings of the IEEE,* vol. 100, no. Special Centennial Issue, pp. 1287-1308, 2012.

[6] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM,* vol. 20, no. 1, pp. 46-61, 1973.

[7] L. Sha, K.-E. Årzén, T. Abdelzaher, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky and A. K. Mok, "Real Time Scheduling Theory: A Historical Perspective," *Real-Time Syst.,* vol. 28, pp. 101-155.

[8] B. Bolt and i. Newman, "A history of the ARPANET: The first decade.," Bolt Beranek and Newman, 1981. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=

html&identifier=ADA115440. [Accessed 01 07 2015].

[9] S. Leimeister, M. Böhm, C. Riedl and H. Krcmar, "The Business Perspective of Cloud Computing: Actors, Roles and Value Networks," in *European Conference on Information Systems (ECIS)*, 2010.

[10] K. Ashton, "That 'internet of things' thing," *RFiD Journal,* vol. 22, no. 7, pp. 91-114, 2009.

[11] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao and C. Fu, "Cloud computing: a perspective study," *New Generation Computing,* vol. 28, no. 2, pp. 137-146, 2010.

[12] R. F. L. a. R. M. Di Lauro, "SIaaS-sensing instrument as a service using cloud computing to turn physical instrument into ubiquitous service," in *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, 2012.

[13] T. Lopez, D. Ranasinghe, M. Harrison and D. McFarlane, "Adding sense to the Internet of Things an architecture framework for smart objective systems," in *Pervasive Ubiquitous Computing 16*, 2012.

[14] 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)," *情報管理,* 第 巻57, 第 11, pp. 826-834, 2015.

[15] IBM, "IBM Watson Internet of Things(モノのインターネット化)," IBM, [Online]. Available: http://www.ibm.com/internet-of-things/jp-ja/. [Accessed 29 08 2016].

[16] M. Windows, "Azure IoT Suite," Microsoft Windows, [Online]. Available: https://www.microsoft.com/ja-jp/server-cloud/products-Microsoft-Azure-IoT-Service.aspx?WT.srch=1&WT.mc_ID=SEM_jHjUAPFU. [Accessed 29 08 2016].

[17] R. Minerva, A. Biru and D. Rotondi, Towards a definition

of the Internet of Things(IoT), IEEE Internet Initiative, 2015, pp. 1-86.

[18] 独立行政法人情報処理推進機構, *IT人材白書2015,* 文京区本駒込2-28-8, 東京都: 独立行政法人情報処理推進機構, p. 4.

[19] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *"Future Gener. Comput. Syst.",* p. 1645-1660, 2013.

[20] "Education Network Practic Information and Technologies," Ministry of Education, Culture, Sports, Science and Technology of Japan, [Online]. Available: http://www.enpit.jp/. [Accessed 01 09 2016].

[21] M. Antunes, J. P. Barraca, D. Gomes, P. Oliveira and R. L. Aguiar, *Unified platform for M2M Telo Providers,* Springer International Publishing, 2014, pp. 436-443.

[22] M. Antunes, J. P. Barraca, D. Gomes, P. Oliveira and R. L. Aguiar, "Smart Cloud of Things: An Evolved IoT Platform for Telco Providers," *Journal of Ambientcom,* vol. 1, pp. 1-24, 2015.

[23] Eclipse, "IoT Services & Frameworks," eclipse, [Online]. Available: https://iot.eclipse.org/frameworks. [Accessed 29 08 2016].

[24] Microsoft, "Azure IoT Suite," Microsoft Windows, [Online]. Available: https://www.microsoft.com/ja-jp/server-cloud/products-Microsoft-Azure-IoT-Service.aspx?WT.srch=1&WT.mc_ID=SEM_jHjUAPFU. [Accessed 2016 08 29].

[25] MIT, "About Scratch," The Lifelong Kindergarten Group at the MIT Media Lab, [Online]. Available: https://scratch.mit.edu/about/. [Accessed 01 01 2016].

[26] A. B. Kramp, M. Bauer, M. Fiedler, Thorsten, R. v.

Kranenburg, S. Lange and S. Meissner, "Enabling Things to Talk," Springer Heidelberg New York Dordrecht London, 2013, p. 208.

[27] C. Maia, L. M. Nogueira and L. M. Pinho, "Evaluating android os for embedded real-time systems," in *6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications*, 2010.

[28] "RaspberryPi," [Online]. Available: http://www.raspberrypi.org/. [Accessed 01 01 2015].

[29] S. Li, L. Da Xu and S. Zhao, "The internet of things: a survey.," *Information Systems Frontiers,* vol. 17, no. 2, pp. 243-259, 2015.

[30] L. Atzori, A. lera and G. Morabito, "The internet of things: A survey," *Computer networks,* vol. 54, no. 15, pp. 2787-2805, otc 2010.

[31] B. Eric, "Embedded Linux Keeps Growing Amid IoT Disruption, Says Study," [Online]. Available: https://www.linux.com/news/embedded-linux-keeps-growing-amid-iot-disruption-says-study. [Accessed 01 07 2016].

[32] 中川裕貴, 早川栄一 , 西野洋介, "AndroidにおけるOSプロセス可視化環境の開発," IPSJ, 2011.

[33] E. Frank Ch, P. Vara, C. Will, N. Hien, H. Martin, K. Jim and C. Brad, "Architecture of systemtap: a Linux trace/probe tool," [Online]. Available: https://sourceware.org/systemtap/archpaper.pdf. [Accessed 15 08 2015].

[34] D. Mathieu and D. Michel R., "The LTTng tracer: A low impact performance and behavior monitor for GNU/Linux," *Linux Symposium,* vol. 1, pp. 209-223, 2006.

[35] 後藤隼弐, 本田晋也, 長尾卓哉 , 高田広章, "トレースログ可視化ツールの開発(ドライバ,ツール,組込技術とネットワークに関するワークショップETNET2009),"

電子情報通信学会技術研究報告. CPSY, コンピュータシステム, 第 巻108, 第 463, pp. 73-78, 2009.

[36] M. Sugaya, H. Takamura, Y. Ishiwata, S. Kagami and K. Kuramitsu, "Online Kernel Log Analysis for Robotics Application," *Journal of Information Processing,* vol. 21, no. 1, pp. 53-66, 2013.

[37] "Ftrace," [Online]. Available: http://elinux.org/Ftrace. [Accessed 01 01 2015].

[38] "ftrace," [Online]. Available: https://access.redhat.com/site/documentation/ja-JP/Red_Hat_Enterprise_Linux/6/html/Developer_Guide/ftrace.html. [Accessed 01 01 2016].

[39] Y. Nakagawa, P. Amontamavut, Y. Nishino and E. Hayakawa, "Android OSにおけるプロセス可視化環境," in *SWEST13,* 2011.

[40] s. xian-guang, "ksocket," [Online]. Available: http://ksocket.sourceforge.net. [Accessed 01 07 2016].

[41] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol. 29, no. 7, pp. 1645-1660, 2013.

[42] T. Imai, "|ScratchとRaspberry Piの遊び方|," [オンライン]. Available: http://pushl.net/teach/. [アクセス日: 01 07 2015].

[43] "What is Eclipse and the Eclipse Foundation?," The Eclipse Foundation, [Online]. Available: https://eclipse.org/org/. [Accessed 20 06 2015].

[44] "The WebSocket Protocol," Internet Engineering Task Force (IETF), [Online]. Available: https://tools.ietf.org/html/rfc6455. [Accessed 30 09 2016].

[45] U. Hunkeler, H. L. Truong and A. Stanford-Clark, "MQTT-S-A

publish/subscribe protocol for Wireless Sensor Networks," 2008.

[46] "GlusterFS: Storage for your Cloud," [Online]. Available: https://www.gluster.org/. [Accessed 03 08 2016].

[47] "redis," [Online]. Available: http://redis.io/. [Accessed 03 08 2016].

[48] アモーンタマウットプラウィーン , 早川栄一, "分散組込みシステム向きWebベース開発環境," 著: *組込みシステムシンポジウム2015論文集*, 2015.

[49] J. Marini, Document Object Model, 1 ed., New York: McGraw-Hill, Inc., 2002.

[50] A. v. Kesteren, J. Aubourg, J. Song and H. R. M. Steen, "W3C Working Draft 30 January 2014," W3C, 2014. [Online]. Available: https://www.w3.org/TR/01XMLHttpRequest/. [Accessed 01 01 2015].

[51] A. B.V., "http://vis.org," Almende B.V., [Online]. Available: http://visjs.org. [Accessed 11 01 2015].

[52] B. Almende, "Timeline," Almende, B.V., [Online]. Available: http://visjs.org/docs/timeline/. [Accessed 11 01 2015].

[53] C. Fehmer, "adc-pi-spi," [Online]. Available: https://www.npmjs.com/package/adc-pi-spi. [Accessed 2015 01 03].

[54] C. VECCHIOLA, X. CHU, M. MATTESS and R. BUYYA, "ANEKA-INTEGRATION OF PRIVATE AND PUBLIC CLOUDS," in *Cloud Computing: Principles and Paradigms*, 2011.

[55] x. Logmeln, "Solutions by Department," xively, [Online]. Available: https://xively.com/solutions-department/. [Accessed 06 07 2015].

[56] "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]. Available:

http://www.ibm.com/developerworks/jp/cloud/library/cl-orionsummary/. [アクセス日: 15 09 2015].

[57] "IoT Services & Frameworks," Eclipse, [Online]. Available: http://iot.eclipse.org/frameworks. [Accessed 29 08 2016].

[58] C. Bormann, "CoAP," TZI, [Online]. Available: http://coap.technology/. [Accessed 30 09 2015].

[59] oma, *Lightweight Machine to Machine Architecture,* 2012 Open Mobile Alliance Ltd., 2012.

[60] 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),"
情報管理, 第 巻57, 第 11, pp. 826-834, 2015.

[61] 情報通信研究機構,
"大規模IoTサービスの実証ができるセンサー・クラウド基盤JOSE," 情報通信研究機構, [オンライン]. Available: https://www.nict.go.jp/nrh/nwgn/jose.html. [アクセス日: 10 08 2016].

[62] Afrel, "IoT カリキュラム学習セット," 株式会社Afrel, [オンライン]. Available: http://afrel-shop.com/shopdetail/000000000370/. [アクセス日: 10 08 2016].

[63] LogMeIn, "Xively," LogMeIn, [Online]. Available: https://www.xively.com/. [Accessed 10 08 2016].

[64] uhuru, "Milkcocoa," uhuru, [オンライン]. Available: https://mlkcca.com/. [アクセス日: 10 08 2016].

[65] P. Amontamavut and E. Hayakawa, "Blue-sky Logger:A Scalable Remote Monitoring Platform for Embedded System on Linux," in *Procs. of The 1st IEEE International Conference on Cyber-Physical Systems, Networks, and Applications*, 2013.

[66] S. OKAZAKI, M. INOUE, T. NAKAJIMA, T. SHIOTSUKI, H. KAMBE and H. KOIZUMI, "Education system of sensor technologies

in IoT," 2016.

[67] D. Namiot and M. Sneps-Sneppe, "On IoT Programming," *International Journal of Open Information Technologies,* vol. 2, no. 10, pp. 25-28, 2014.

[68] F. J. Lin, Y. Ren and E. Cerritos, "A Feasiblity Study on Developing IoT/M2M Applications over ETSI M2M Architecture," in *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, 2013.

[69] D. Dobrilovic, Z. Stojanov and B. Odadzic, "Teaching application development for RFID/ZigBee networks using open source hardware," in *Telecommunications (BIHTEL), 2014 X International Symposium on*, 2014.

[70] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645-1660, 2013.

[71] A. Blasdel, A. Leeper, A. R. Tsouroukdissian, A. Hornung, A. Hendrix, C. Rockey, D. Stonier, D. Coleman, D. Hershberger, D. Gossow, D. Lu, D. Thomas, DorianScholz, E. Fernandez, E. Perko and F. Lier, "About ROS," ROS, [Online]. Available: http://www.ros.org/about-ros. [Accessed 01 02 2016].

[72] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[73] P. Merrick, S. Allen and J. Lapp, *XML remote procedure call (XML-RPC),* Google Patents, 2006.

[74] T. Straszheim, M. Kjaergaard, B. Gerkey and D. Thomas, "catkin," ROS, [Online]. Available: http://docs.ros.org/api/catkin/html/. [Accessed 08 12 2015].

[75] D. Thomas, "rqt_graph," ROS, [Online]. Available: http://wiki.ros.org/rqt_graph. [Accessed 17 12 2015].

[76] D. Thomas, "roscore," ROS, [Online]. Available: http://wiki.ros.org/roscore. [Accessed 17 12 2015].

[77] I. Saito, "rosout," ROS, [Online]. Available: http://wiki.ros.org/rosout. [Accessed 1 March 2016].

[78] M. Morley, "JSON-RPC 2.0 Specification," JSON-RPC google group, [Online]. Available: http://www.jsonrpc.org/specification. [Accessed 11 02 2016].

[79] B. Almende, "DataSet," Almende, B.V., [Online]. Available: http://visjs.org/docs/data/dataset.html. [Accessed 11 01 2015].

[80] B. Almende, "network," Almende, B.V., [Online]. Available: http://visjs.org/docs/data/dataset.html. [Accessed 1 March 2016].

[81] K. Bhogal, R. Peterson and L. Seacat, *Bandwidth usage and latency reduction of remote desktop software based on preferred rendering of a user selected area,* Google Patents, 2011.

付　録

付録 A. Blue-Sky サーバのマニュアル

付録A.1　まえがき

Blue-Sky サーバは，著者が拓殖大学早川研究室において，開発されたソフトウェアである．本ソフトウェアは，CPS/IoT を構築可能な Tiny HTTPD を含めた IoT の管理サーバ群である．本ソフトウェアは Java と Node.js[2]から作成された．Java のバイトコードはバージョン 1.6 なので，1.6 以上の Java JDK[3]を用いることである．そして，Node.js は v0.10.37 以上を用いることである．Blue-Sky のサーバ次のとおりである．

- **Blue-Sky の Core サーバ**：
  本サーバは通常の HTTPD サーバの機能として扱うことも可能であり，IoT を構築することも可能である．IoT を構築する場合は Redis と共に動作する．本サーバはゲートウェイ階層の中央サーバであることで，本システムをアクセスする入口である．本サーバはセンシング・アクチュエーションを行う API を提供する．そして，圧縮を備えた ftrace の制御 API およびトレースデータのアクセス API を提供する．
- **Blue-Sky の skycoder のコードを保存するサーバ**：
  skycoder で組んだ JavaScript コードをデプロイしたり保存したりするサーバである．本サーバは Redis と共に動作する．
- **Blue-Sky の skycoder の通信情報のデータを保存するサーバ**：
  skycoder の通信シーケンスや HTTP ヘッダのタイムスタンプを保存するサーバである．本サーバは redis と共に動作する．
- **Blue-Sky の skycoder のユーザの管理サーバ**：
  skycoder のユーザ名やパスワードやトークン（1.1.1 の enduser.db）を管理する．skycoder のバックエンドとして構築される．本サーバは sqlite3 と共に動作する．

---

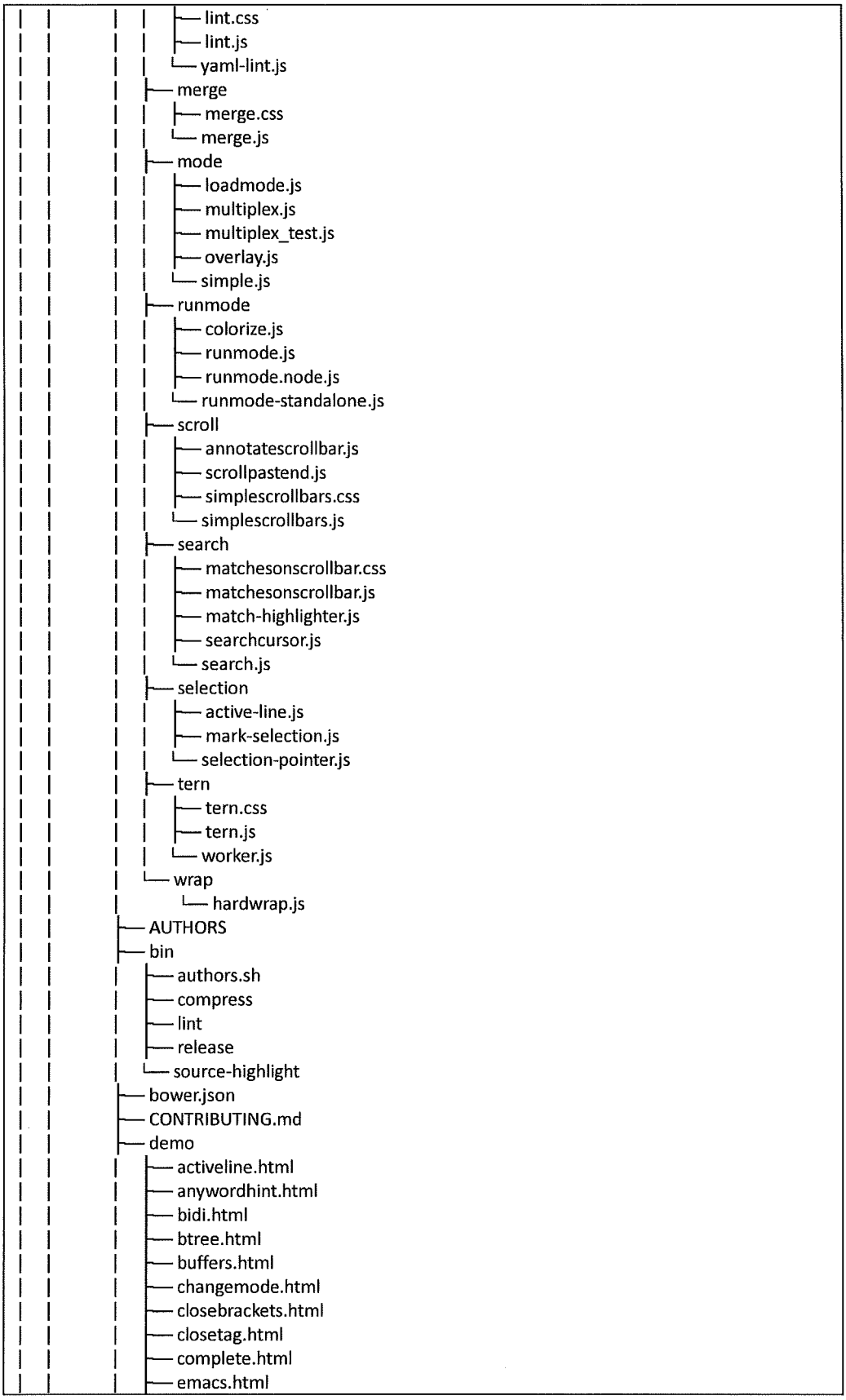
[2] https://nodejs.org

[3] https://www.java.com

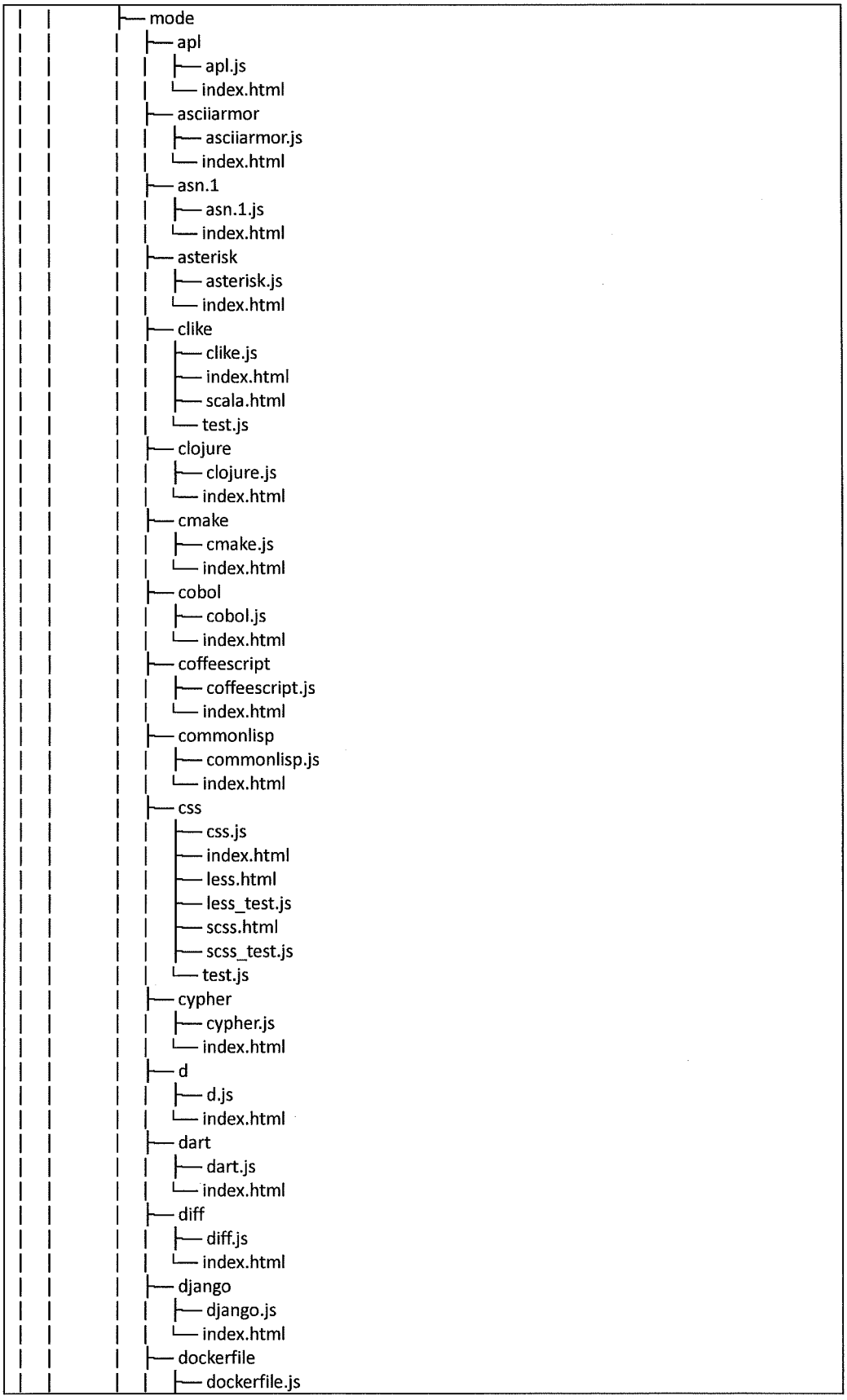
付録A.2　ファイルリスト

1.1.1 Blue-Sky の Core サーバのリスト

```
./Blue-Sky_Core
├── EDLCP.sh
├── enduser.db
├── HttpdAsDeamon.sh
├── Httpd.cfg
├── Httpd.jar
├── Httpd.sh
├── Inetpub
│   ├── Base64EncodeDecode.js
│   ├── codemirror
│   │   └── codemirror-5.3
│   │       ├── addon
│   │       │   ├── comment
│   │       │   │   ├── comment.js
│   │       │   │   └── continuecomment.js
│   │       │   ├── dialog
│   │       │   │   ├── dialog.css
│   │       │   │   └── dialog.js
│   │       │   ├── display
│   │       │   │   ├── fullscreen.css
│   │       │   │   ├── fullscreen.js
│   │       │   │   ├── panel.js
│   │       │   │   ├── placeholder.js
│   │       │   │   └── rulers.js
│   │       │   ├── edit
│   │       │   │   ├── closebrackets.js
│   │       │   │   ├── closetag.js
│   │       │   │   ├── continuelist.js
│   │       │   │   ├── matchbrackets.js
│   │       │   │   ├── matchtags.js
│   │       │   │   └── trailingspace.js
│   │       │   ├── fold
│   │       │   │   ├── brace-fold.js
│   │       │   │   ├── comment-fold.js
│   │       │   │   ├── foldcode.js
│   │       │   │   ├── foldgutter.css
│   │       │   │   ├── foldgutter.js
│   │       │   │   ├── indent-fold.js
│   │       │   │   ├── markdown-fold.js
│   │       │   │   └── xml-fold.js
│   │       │   ├── hint
│   │       │   │   ├── anyword-hint.js
│   │       │   │   ├── css-hint.js
│   │       │   │   ├── html-hint.js
│   │       │   │   ├── javascript-hint.js
│   │       │   │   ├── show-hint.css
│   │       │   │   ├── show-hint.js
│   │       │   │   ├── sql-hint.js
│   │       │   │   └── xml-hint.js
│   │       │   ├── lint
│   │       │   │   ├── coffeescript-lint.js
│   │       │   │   ├── css-lint.js
│   │       │   │   ├── javascript-lint.js
│   │       │   │   └── json-lint.js
```


lint.css
lint.js
yaml-lint.js
merge
merge.css
merge.js
mode
loadmode.js
multiplex.js
multiplex_test.js
overlay.js
simple.js
runmode
colorize.js
runmode.js
runmode.node.js
runmode-standalone.js
scroll
annotatescrollbar.js
scrollpastend.js
simplescrollbars.css
simplescrollbars.js
search
matchesonscrollbar.css
matchesonscrollbar.js
match-highlighter.js
searchcursor.js
search.js
selection
active-line.js
mark-selection.js
selection-pointer.js
tern
tern.css
tern.js
worker.js
wrap
hardwrap.js
AUTHORS
bin
authors.sh
compress
lint
release
source-highlight
bower.json
CONTRIBUTING.md
demo
activeline.html
anywordhint.html
bidi.html
btree.html
buffers.html
changemode.html
closebrackets.html
closetag.html
complete.html
emacs.html

```
│   │   │   ├── folding.html
│   │   │   ├── fullscreen.html
│   │   │   ├── hardwrap.html
│   │   │   ├── html5complete.html
│   │   │   ├── indentwrap.html
│   │   │   ├── lint.html
│   │   │   ├── loadmode.html
│   │   │   ├── marker.html
│   │   │   ├── markselection.html
│   │   │   ├── matchhighlighter.html
│   │   │   ├── matchtags.html
│   │   │   ├── merge.html
│   │   │   ├── multiplex.html
│   │   │   ├── mustache.html
│   │   │   ├── panel.html
│   │   │   ├── placeholder.html
│   │   │   ├── preview.html
│   │   │   ├── requirejs.html
│   │   │   ├── resize.html
│   │   │   ├── rulers.html
│   │   │   ├── runmode.html
│   │   │   ├── search.html
│   │   │   ├── simplemode.html
│   │   │   ├── simplescrollbars.html
│   │   │   ├── spanaffectswrapping_shim.html
│   │   │   ├── sublime.html
│   │   │   ├── tern.html
│   │   │   ├── theme.html
│   │   │   ├── trailingspace.html
│   │   │   ├── variableheight.html
│   │   │   ├── vim.html
│   │   │   ├── visibletabs.html
│   │   │   ├── widget.html
│   │   │   └── xmlcomplete.html
│   │   ├── doc
│   │   │   ├── activebookmark.js
│   │   │   ├── compress.html
│   │   │   ├── docs.css
│   │   │   ├── internals.html
│   │   │   ├── logo.png
│   │   │   ├── logo.svg
│   │   │   ├── manual.html
│   │   │   ├── realworld.html
│   │   │   ├── releases.html
│   │   │   ├── reporting.html
│   │   │   ├── upgrade_v2.2.html
│   │   │   ├── upgrade_v3.html
│   │   │   ├── upgrade_v4.html
│   │   │   └── yinyang.png
│   │   ├── index.html
│   │   ├── keymap
│   │   │   ├── emacs.js
│   │   │   ├── sublime.js
│   │   │   └── vim.js
│   │   ├── lib
│   │   │   ├── codemirror.css
│   │   │   └── codemirror.js
│   │   ├── LICENSE
```

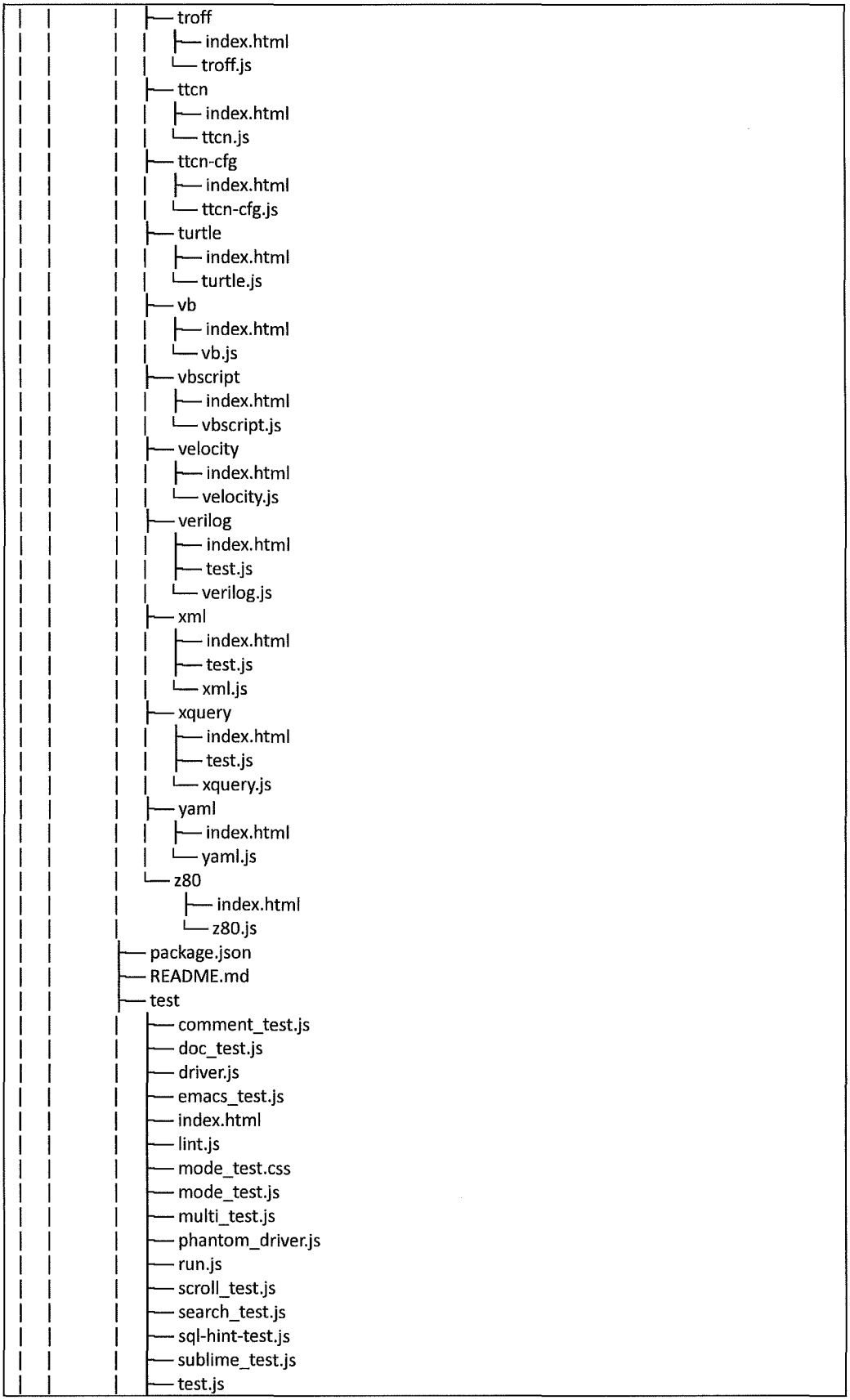

mode
apl
apl.js
index.html
asciiarmor
asciiarmor.js
index.html
asn.1
asn.1.js
index.html
asterisk
asterisk.js
index.html
clike
clike.js
index.html
scala.html
test.js
clojure
clojure.js
index.html
cmake
cmake.js
index.html
cobol
cobol.js
index.html
coffeescript
coffeescript.js
index.html
commonlisp
commonlisp.js
index.html
css
css.js
index.html
less.html
less_test.js
scss.html
scss_test.js
test.js
cypher
cypher.js
index.html
d
d.js
index.html
dart
dart.js
index.html
diff
diff.js
index.html
django
django.js
index.html
dockerfile
dockerfile.js

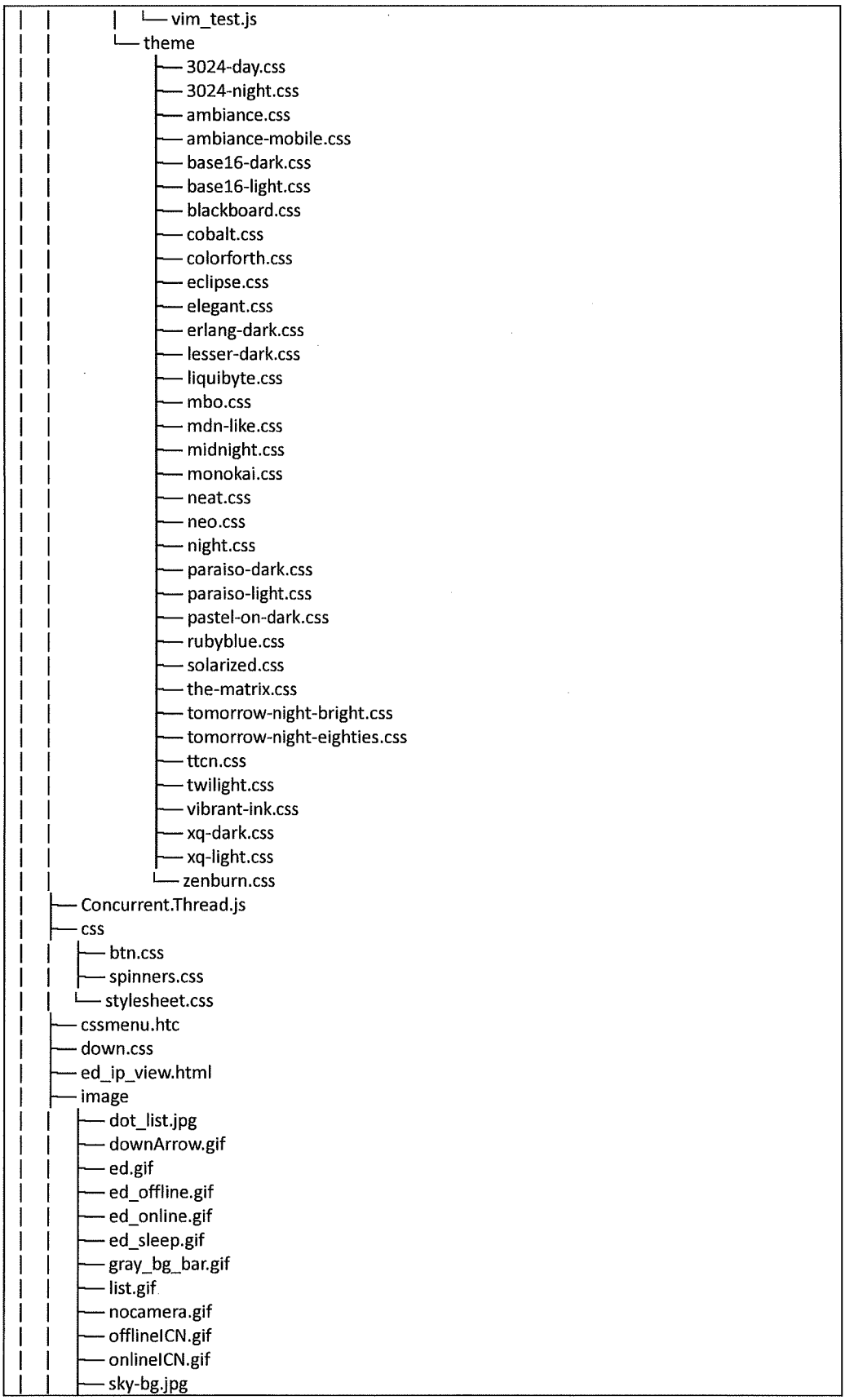
```
│   │   │   │   └── index.html
│   │   │   ├── dtd
│   │   │   │   ├── dtd.js
│   │   │   │   └── index.html
│   │   │   ├── dylan
│   │   │   │   ├── dylan.js
│   │   │   │   └── index.html
│   │   │   ├── ebnf
│   │   │   │   ├── ebnf.js
│   │   │   │   └── index.html
│   │   │   ├── ecl
│   │   │   │   ├── ecl.js
│   │   │   │   └── index.html
│   │   │   ├── eiffel
│   │   │   │   ├── eiffel.js
│   │   │   │   └── index.html
│   │   │   ├── erlang
│   │   │   │   ├── erlang.js
│   │   │   │   └── index.html
│   │   │   ├── forth
│   │   │   │   ├── forth.js
│   │   │   │   └── index.html
│   │   │   ├── fortran
│   │   │   │   ├── fortran.js
│   │   │   │   └── index.html
│   │   │   ├── gas
│   │   │   │   ├── gas.js
│   │   │   │   └── index.html
│   │   │   ├── gfm
│   │   │   │   ├── gfm.js
│   │   │   │   ├── index.html
│   │   │   │   └── test.js
│   │   │   ├── gherkin
│   │   │   │   ├── gherkin.js
│   │   │   │   └── index.html
│   │   │   ├── go
│   │   │   │   ├── go.js
│   │   │   │   └── index.html
│   │   │   ├── groovy
│   │   │   │   ├── groovy.js
│   │   │   │   └── index.html
│   │   │   ├── haml
│   │   │   │   ├── haml.js
│   │   │   │   ├── index.html
│   │   │   │   └── test.js
│   │   │   ├── handlebars
│   │   │   │   ├── handlebars.js
│   │   │   │   └── index.html
│   │   │   ├── haskell
│   │   │   │   ├── haskell.js
│   │   │   │   └── index.html
│   │   │   ├── haxe
│   │   │   │   ├── haxe.js
│   │   │   │   └── index.html
│   │   │   ├── htmlembedded
│   │   │   │   ├── htmlembedded.js
│   │   │   │   └── index.html
│   │   │   ├── htmlmixed
```

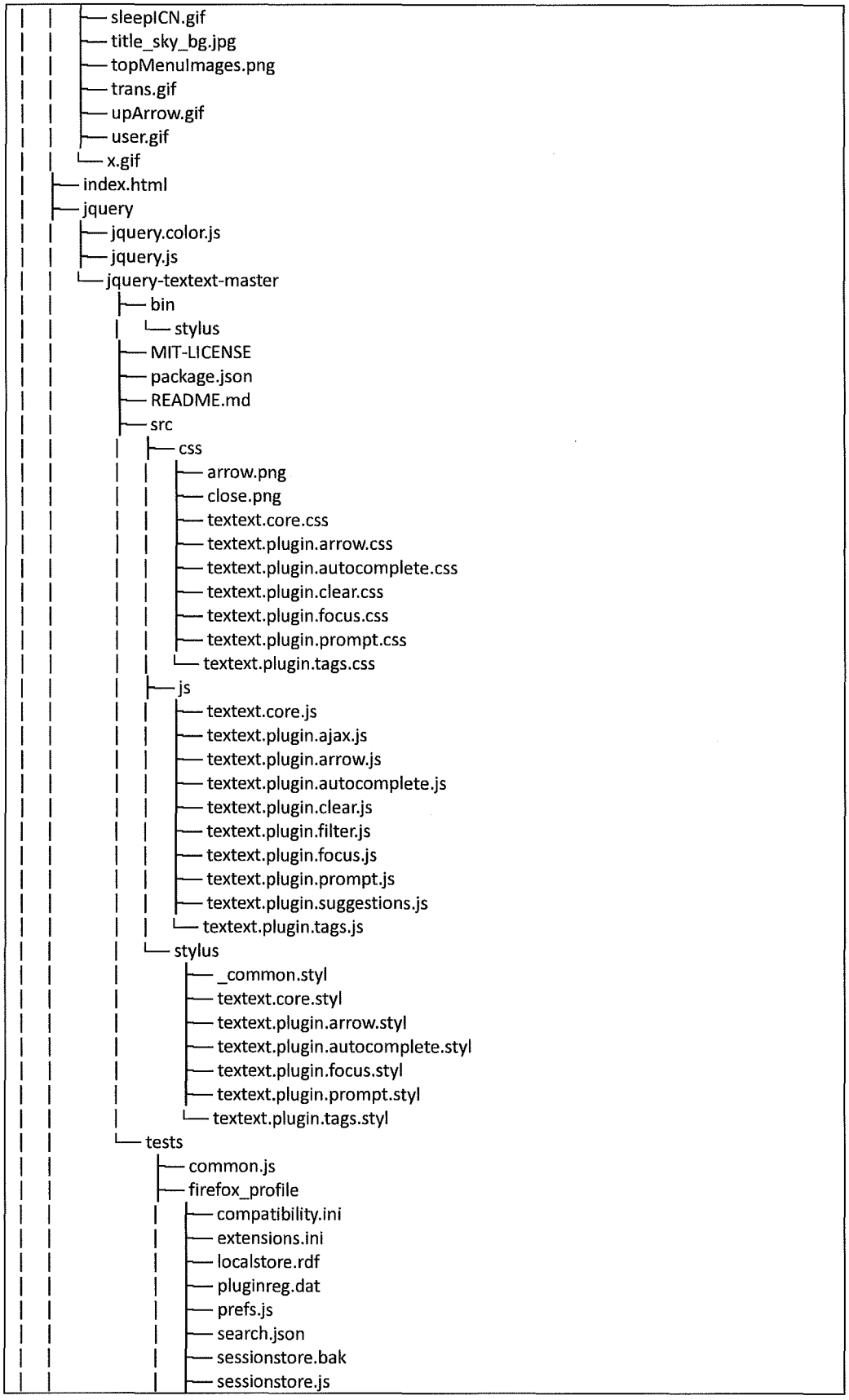
```
|   |   |   |   ├── htmlmixed.js
|   |   |   |   └── index.html
|   |   |   ├── http
|   |   |   |   ├── http.js
|   |   |   |   └── index.html
|   |   |   ├── idl
|   |   |   |   ├── idl.js
|   |   |   |   └── index.html
|   |   |   ├── index.html
|   |   |   ├── jade
|   |   |   |   ├── index.html
|   |   |   |   └── jade.js
|   |   |   ├── javascript
|   |   |   |   ├── index.html
|   |   |   |   ├── javascript.js
|   |   |   |   ├── json-ld.html
|   |   |   |   ├── test.js
|   |   |   |   └── typescript.html
|   |   |   ├── jinja2
|   |   |   |   ├── index.html
|   |   |   |   └── jinja2.js
|   |   |   ├── julia
|   |   |   |   ├── index.html
|   |   |   |   └── julia.js
|   |   |   ├── kotlin
|   |   |   |   ├── index.html
|   |   |   |   └── kotlin.js
|   |   |   ├── livescript
|   |   |   |   ├── index.html
|   |   |   |   └── livescript.js
|   |   |   ├── lua
|   |   |   |   ├── index.html
|   |   |   |   └── lua.js
|   |   |   ├── markdown
|   |   |   |   ├── index.html
|   |   |   |   ├── markdown.js
|   |   |   |   └── test.js
|   |   |   ├── mathematica
|   |   |   |   ├── index.html
|   |   |   |   └── mathematica.js
|   |   |   ├── meta.js
|   |   |   ├── mirc
|   |   |   |   ├── index.html
|   |   |   |   └── mirc.js
|   |   |   ├── mllike
|   |   |   |   ├── index.html
|   |   |   |   └── mllike.js
|   |   |   ├── modelica
|   |   |   |   ├── index.html
|   |   |   |   └── modelica.js
|   |   |   ├── mumps
|   |   |   |   ├── index.html
|   |   |   |   └── mumps.js
|   |   |   ├── nginx
|   |   |   |   ├── index.html
|   |   |   |   └── nginx.js
|   |   |   ├── ntriples
|   |   |   |   ├── index.html
```

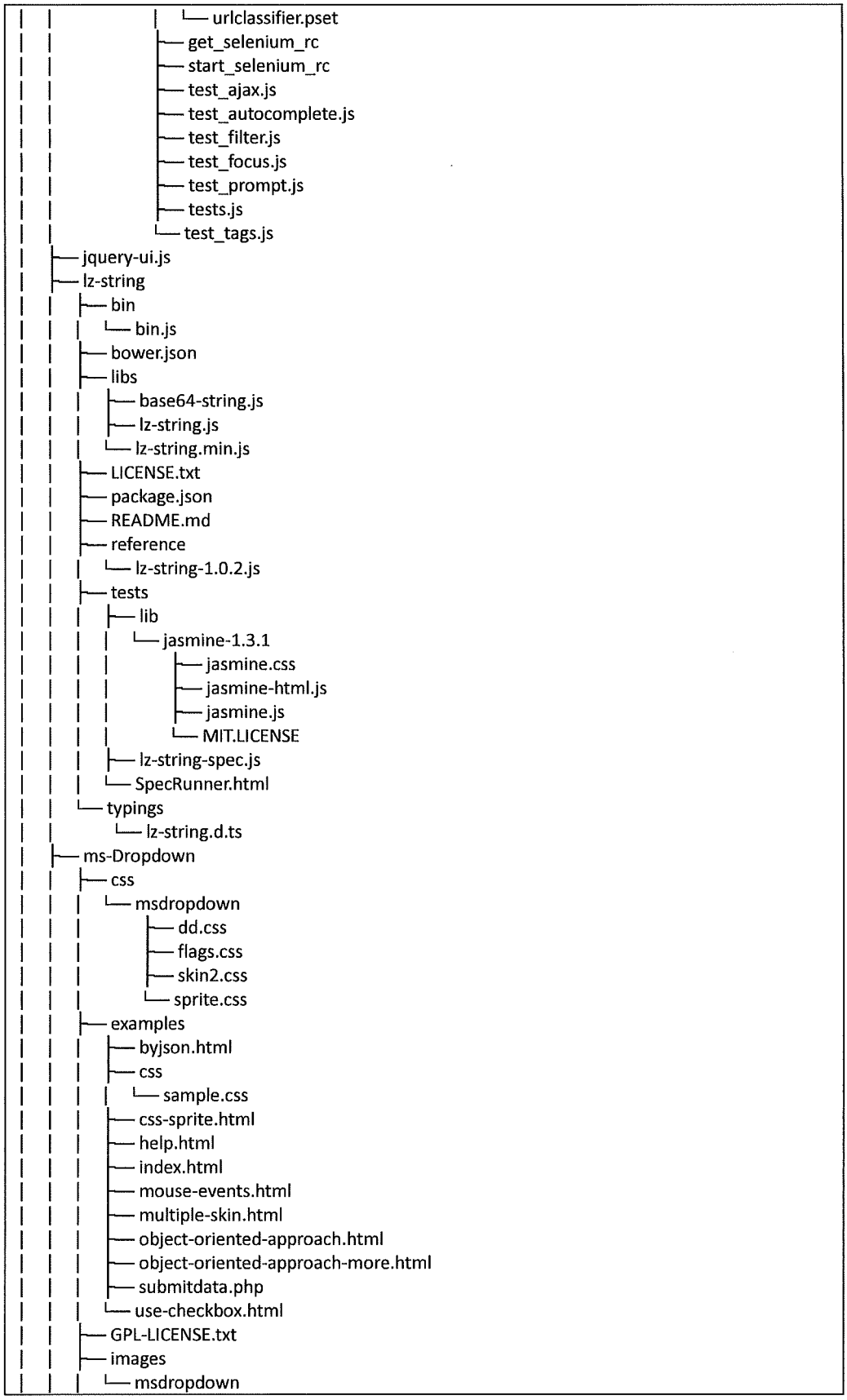
```
|   |   |   |   └── ntriples.js
|   |   |   ├── octave
|   |   |   |   ├── index.html
|   |   |   |   └── octave.js
|   |   |   ├── pascal
|   |   |   |   ├── index.html
|   |   |   |   └── pascal.js
|   |   |   ├── pegjs
|   |   |   |   ├── index.html
|   |   |   |   └── pegjs.js
|   |   |   ├── perl
|   |   |   |   ├── index.html
|   |   |   |   └── perl.js
|   |   |   ├── php
|   |   |   |   ├── index.html
|   |   |   |   ├── php.js
|   |   |   |   └── test.js
|   |   |   ├── pig
|   |   |   |   ├── index.html
|   |   |   |   └── pig.js
|   |   |   ├── properties
|   |   |   |   ├── index.html
|   |   |   |   └── properties.js
|   |   |   ├── puppet
|   |   |   |   ├── index.html
|   |   |   |   └── puppet.js
|   |   |   ├── python
|   |   |   |   ├── index.html
|   |   |   |   └── python.js
|   |   |   ├── q
|   |   |   |   ├── index.html
|   |   |   |   └── q.js
|   |   |   ├── r
|   |   |   |   ├── index.html
|   |   |   |   └── r.js
|   |   |   ├── rpm
|   |   |   |   ├── changes
|   |   |   |   |   └── index.html
|   |   |   |   ├── index.html
|   |   |   |   └── rpm.js
|   |   |   ├── rst
|   |   |   |   ├── index.html
|   |   |   |   └── rst.js
|   |   |   ├── ruby
|   |   |   |   ├── index.html
|   |   |   |   ├── ruby.js
|   |   |   |   └── test.js
|   |   |   ├── rust
|   |   |   |   ├── index.html
|   |   |   |   └── rust.js
|   |   |   ├── sass
|   |   |   |   ├── index.html
|   |   |   |   └── sass.js
|   |   |   ├── scheme
|   |   |   |   ├── index.html
|   |   |   |   └── scheme.js
|   |   |   ├── shell
|   |   |   |   ├── index.html
```

```
|   |   |   |   ├── shell.js
|   |   |   |   └── test.js
|   |   |   ├── sieve
|   |   |   |   ├── index.html
|   |   |   |   └── sieve.js
|   |   |   ├── slim
|   |   |   |   ├── index.html
|   |   |   |   ├── slim.js
|   |   |   |   └── test.js
|   |   |   ├── smalltalk
|   |   |   |   ├── index.html
|   |   |   |   └── smalltalk.js
|   |   |   ├── smarty
|   |   |   |   ├── index.html
|   |   |   |   └── smarty.js
|   |   |   ├── solr
|   |   |   |   ├── index.html
|   |   |   |   └── solr.js
|   |   |   ├── soy
|   |   |   |   ├── index.html
|   |   |   |   └── soy.js
|   |   |   ├── sparql
|   |   |   |   ├── index.html
|   |   |   |   └── sparql.js
|   |   |   ├── spreadsheet
|   |   |   |   ├── index.html
|   |   |   |   └── spreadsheet.js
|   |   |   ├── sql
|   |   |   |   ├── index.html
|   |   |   |   └── sql.js
|   |   |   ├── stex
|   |   |   |   ├── index.html
|   |   |   |   ├── stex.js
|   |   |   |   └── test.js
|   |   |   ├── stylus
|   |   |   |   ├── index.html
|   |   |   |   └── stylus.js
|   |   |   ├── tcl
|   |   |   |   ├── index.html
|   |   |   |   └── tcl.js
|   |   |   ├── textile
|   |   |   |   ├── index.html
|   |   |   |   ├── test.js
|   |   |   |   └── textile.js
|   |   |   ├── tiddlywiki
|   |   |   |   ├── index.html
|   |   |   |   ├── tiddlywiki.css
|   |   |   |   └── tiddlywiki.js
|   |   |   ├── tiki
|   |   |   |   ├── index.html
|   |   |   |   ├── tiki.css
|   |   |   |   └── tiki.js
|   |   |   ├── toml
|   |   |   |   ├── index.html
|   |   |   |   └── toml.js
|   |   |   ├── tornado
|   |   |   |   ├── index.html
|   |   |   |   └── tornado.js
```


troff
index.html
troff.js
ttcn
index.html
ttcn.js
ttcn-cfg
index.html
ttcn-cfg.js
turtle
index.html
turtle.js
vb
index.html
vb.js
vbscript
index.html
vbscript.js
velocity
index.html
velocity.js
verilog
index.html
test.js
verilog.js
xml
index.html
test.js
xml.js
xquery
index.html
test.js
xquery.js
yaml
index.html
yaml.js
z80
index.html
z80.js
package.json
README.md
test
comment_test.js
doc_test.js
driver.js
emacs_test.js
index.html
lint.js
mode_test.css
mode_test.js
multi_test.js
phantom_driver.js
run.js
scroll_test.js
search_test.js
sql-hint-test.js
sublime_test.js
test.js


vim_test.js
theme
3024-day.css
3024-night.css
ambiance.css
ambiance-mobile.css
base16-dark.css
base16-light.css
blackboard.css
cobalt.css
colorforth.css
eclipse.css
elegant.css
erlang-dark.css
lesser-dark.css
liquibyte.css
mbo.css
mdn-like.css
midnight.css
monokai.css
neat.css
neo.css
night.css
paraiso-dark.css
paraiso-light.css
pastel-on-dark.css
rubyblue.css
solarized.css
the-matrix.css
tomorrow-night-bright.css
tomorrow-night-eighties.css
ttcn.css
twilight.css
vibrant-ink.css
xq-dark.css
xq-light.css
zenburn.css
Concurrent.Thread.js
css
btn.css
spinners.css
stylesheet.css
cssmenu.htc
down.css
ed_ip_view.html
image
dot_list.jpg
downArrow.gif
ed.gif
ed_offline.gif
ed_online.gif
ed_sleep.gif
gray_bg_bar.gif
list.gif
nocamera.gif
offlineICN.gif
onlineICN.gif
sky-bg.jpg


sleepICN.gif
title_sky_bg.jpg
topMenuImages.png
trans.gif
upArrow.gif
user.gif
x.gif
index.html
jquery
jquery.color.js
jquery.js
jquery-textext-master
bin
stylus
MIT-LICENSE
package.json
README.md
src
css
arrow.png
close.png
textext.core.css
textext.plugin.arrow.css
textext.plugin.autocomplete.css
textext.plugin.clear.css
textext.plugin.focus.css
textext.plugin.prompt.css
textext.plugin.tags.css
js
textext.core.js
textext.plugin.ajax.js
textext.plugin.arrow.js
textext.plugin.autocomplete.js
textext.plugin.clear.js
textext.plugin.filter.js
textext.plugin.focus.js
textext.plugin.prompt.js
textext.plugin.suggestions.js
textext.plugin.tags.js
stylus
_common.styl
textext.core.styl
textext.plugin.arrow.styl
textext.plugin.autocomplete.styl
textext.plugin.focus.styl
textext.plugin.prompt.styl
textext.plugin.tags.styl
tests
common.js
firefox_profile
compatibility.ini
extensions.ini
localstore.rdf
pluginreg.dat
prefs.js
search.json
sessionstore.bak
sessionstore.js


urlclassifier.pset
get_selenium_rc
start_selenium_rc
test_ajax.js
test_autocomplete.js
test_filter.js
test_focus.js
test_prompt.js
tests.js
test_tags.js
jquery-ui.js
lz-string
bin
bin.js
bower.json
libs
base64-string.js
lz-string.js
lz-string.min.js
LICENSE.txt
package.json
README.md
reference
lz-string-1.0.2.js
tests
lib
jasmine-1.3.1
jasmine.css
jasmine-html.js
jasmine.js
MIT.LICENSE
lz-string-spec.js
SpecRunner.html
typings
lz-string.d.ts
ms-Dropdown
css
msdropdown
dd.css
flags.css
skin2.css
sprite.css
examples
byjson.html
css
sample.css
css-sprite.html
help.html
index.html
mouse-events.html
multiple-skin.html
object-oriented-approach.html
object-oriented-approach-more.html
submitdata.php
use-checkbox.html
GPL-LICENSE.txt
images
msdropdown

```
|   |   |       ├── icons
|   |   |       |   ├── 1273404841_tick_16.png
|   |   |       |   ├── Amex-56.png
|   |   |       |   ├── blank.gif
|   |   |       |   ├── browser-chrome.png
|   |   |       |   ├── browser-firefox.png
|   |   |       |   ├── browser-msie.png
|   |   |       |   ├── browser-opera.png
|   |   |       |   ├── browser-safari.png
|   |   |       |   ├── button-large.gif
|   |   |       |   ├── Cash-56.png
|   |   |       |   ├── demoaction.jpg
|   |   |       |   ├── Discover-56.png
|   |   |       |   ├── enabled.gif
|   |   |       |   ├── flagssprite_small.png
|   |   |       |   ├── giftlelo_dot_com.jpg
|   |   |       |   ├── icon_calendar.gif
|   |   |       |   ├── icon_cart.gif
|   |   |       |   ├── icon_cd.gif
|   |   |       |   ├── icon_email.gif
|   |   |       |   ├── icon_faq.gif
|   |   |       |   ├── icon_games.gif
|   |   |       |   ├── icon_lockopen.gif
|   |   |       |   ├── icon_music.gif
|   |   |       |   ├── icon_phone.gif
|   |   |       |   ├── icon_sales.gif
|   |   |       |   ├── icon-search.gif
|   |   |       |   ├── icon_secure.gif
|   |   |       |   ├── icon_video.gif
|   |   |       |   ├── jquery-msdropdown.gif
|   |   |       |   ├── Mastercard-56.png
|   |   |       |   ├── msDropDown.gif
|   |   |       |   ├── Paypal-56.png
|   |   |       |   ├── preview.gif
|   |   |       |   ├── sprite.gif
|   |   |       |   ├── sprite.jpg
|   |   |       |   ├── tick.gif
|   |   |       |   └── Visa-56.png
|   |   |       ├── skin1
|   |   |       |   ├── dd_arrow.gif
|   |   |       |   └── title-bg.gif
|   |   |       └── skin2
|   |   |           ├── dd_arrow.gif
|   |   |           ├── icon-ok.gif
|   |   |           └── title-bg.gif
|   |   ├── index.html
|   |   ├── js
|   |   |   ├── jquery
|   |   |   |   ├── jquery-1.9.0.min.js
|   |   |   |   └── tester.js
|   |   |   └── msdropdown
|   |   |       ├── jquery.dd.js
|   |   |       └── jquery.dd.min.js
|   |   ├── MIT-LICENSE.txt
|   |   ├── Readme.md
|   |   └── Readme.txt
|   ├── rosExt.js
|   ├── sky-bg.jpg
```

```
|   |-- manager.html
|   |-- skycoder.html
|   |-- skycoder_includedGraph.html
|   |-- skycoder_includedGraph_rosext_ev3dev.html
|   |-- skycoder_includedGraph_rosext.html
|   |-- SpryAssets
|   |   |-- SpryCollapsiblePanel.css
|   |   |-- SpryCollapsiblePanel.js
|   |   |-- SpryTabbedPanels.css
|   |   |-- SpryTabbedPanels.js
|   |   |-- SpryValidationSelect.css
|   |   |-- SpryValidationSelect.js
|   |   |-- SpryValidationTextarea.css
|   |   `-- SpryValidationTextarea.js
|   |-- title_sky_bg.jpg
|   |-- vis
|   |   |-- CONTRIBUTING.md
|   |   |-- dist
|   |   |   |-- img
|   |   |   |   |-- network
|   |   |   |   |   |-- acceptDeleteIcon.png
|   |   |   |   |   |-- addNodeIcon.png
|   |   |   |   |   |-- backIcon.png
|   |   |   |   |   |-- connectIcon.png
|   |   |   |   |   |-- cross2.png
|   |   |   |   |   |-- cross.png
|   |   |   |   |   |-- deleteIcon.png
|   |   |   |   |   |-- downArrow.png
|   |   |   |   |   |-- editIcon.png
|   |   |   |   |   |-- leftArrow.png
|   |   |   |   |   |-- minus.png
|   |   |   |   |   |-- plus.png
|   |   |   |   |   |-- rightArrow.png
|   |   |   |   |   |-- upArrow.png
|   |   |   |   |   `-- zoomExtends.png
|   |   |   |   `-- timeline
|   |   |   |       `-- delete.png
|   |   |   |-- vis.css
|   |   |   |-- vis.js
|   |   |   |-- vis.map
|   |   |   |-- vis.min.css
|   |   |   `-- vis.min.js
|   |   |-- docs
|   |   |   |-- css
|   |   |   |   |-- prettify.css
|   |   |   |   `-- style.css
|   |   |   |-- dataset.html
|   |   |   |-- dataview.html
|   |   |   |-- graph2d.html
|   |   |   |-- graph3d.html
|   |   |   |-- img
|   |   |   |   |-- graph
|   |   |   |   |   |-- graph120x60.png
|   |   |   |   |   `-- graph.png
|   |   |   |   |-- graph3d120x60.png
|   |   |   |   |-- graph3d.png
|   |   |   |   |-- timeline
|   |   |   |   |   |-- timeline120x60.png
```

```
│   │   │   │   └── timeline.png
│   │   │   ├── vis_overview.odg
│   │   │   └── vis_overview.png
│   │   ├── index.html
│   │   ├── lib
│   │   │   └── prettify
│   │   │       ├── lang-apollo.js
│   │   │       ├── lang-css.js
│   │   │       ├── lang-hs.js
│   │   │       ├── lang-lisp.js
│   │   │       ├── lang-lua.js
│   │   │       ├── lang-ml.js
│   │   │       ├── lang-proto.js
│   │   │       ├── lang-scala.js
│   │   │       ├── lang-sql.js
│   │   │       ├── lang-vb.js
│   │   │       ├── lang-vhdl.js
│   │   │       ├── lang-wiki.js
│   │   │       ├── lang-yaml.js
│   │   │       ├── prettify.css
│   │   │       └── prettify.js
│   │   ├── network.html
│   │   └── timeline.html
│   ├── examples
│   │   ├── graph2d
│   │   │   ├── 01_basic.html
│   │   │   ├── 02_bars.html
│   │   │   ├── 03_groups.html
│   │   │   ├── 04_rightAxis.html
│   │   │   ├── 05_bothAxis.html
│   │   │   ├── 06_interpolation.html
│   │   │   ├── 07_scrollingAndSorting.html
│   │   │   ├── 08_performance.html
│   │   │   ├── 09_external_legend.html
│   │   │   ├── 10_barsSideBySide.html
│   │   │   ├── 11_barsSideBySideGroups.html
│   │   │   ├── 12_customRange.html
│   │   │   ├── 13_localization.html
│   │   │   ├── 14_toggleGroups.html
│   │   │   ├── 15_streaming_data.html
│   │   │   ├── 16_bothAxis_titles.html
│   │   │   ├── 17_dynamicStyling.html
│   │   │   ├── 18_scatterplot.html
│   │   │   ├── 19_labels.html
│   │   │   ├── default.css
│   │   │   └── index.html
│   │   ├── graph3d
│   │   │   ├── default.css
│   │   │   ├── example01_basis.html
│   │   │   ├── example02_camera.html
│   │   │   ├── example03_filter.html
│   │   │   ├── example04_animate.html
│   │   │   ├── example05_line.html
│   │   │   ├── example06_moving_dots.html
│   │   │   ├── example07_dot_cloud_colors.html
│   │   │   ├── example08_dot_cloud_size.html
│   │   │   ├── example09_mobile.html
│   │   │   ├── example10_styles.html
```

```
|   |   |   |   ├── example11_tooltips.html
|   |   |   |   ├── example12_ticks.html
|   |   |   |   ├── index.html
|   |   |   |   └── playground
|   |   |   |       ├── csv2array.js
|   |   |   |       ├── csv2datatable.html
|   |   |   |       ├── datasource.html
|   |   |   |       ├── datasource.php
|   |   |   |       ├── index.html
|   |   |   |       ├── playground.css
|   |   |   |       ├── playground.js
|   |   |   |       └── prettify
|   |   |   |           ├── lang-apollo.js
|   |   |   |           ├── lang-css.js
|   |   |   |           ├── lang-hs.js
|   |   |   |           ├── lang-lisp.js
|   |   |   |           ├── lang-lua.js
|   |   |   |           ├── lang-ml.js
|   |   |   |           ├── lang-proto.js
|   |   |   |           ├── lang-scala.js
|   |   |   |           ├── lang-sql.js
|   |   |   |           ├── lang-vb.js
|   |   |   |           ├── lang-vhdl.js
|   |   |   |           ├── lang-wiki.js
|   |   |   |           ├── lang-yaml.js
|   |   |   |           ├── prettify.css
|   |   |   |           └── prettify.js
|   |   |   ├── index.html
|   |   |   ├── network
|   |   |   |   ├── 01_basic_usage.html
|   |   |   |   ├── 02_random_nodes.html
|   |   |   |   ├── 03_images.html
|   |   |   |   ├── 04_shapes.html
|   |   |   |   ├── 05_social_network.html
|   |   |   |   ├── 06_groups.html
|   |   |   |   ├── 07_selections.html
|   |   |   |   ├── 08_mobile_friendly.html
|   |   |   |   ├── 09_sizing.html
|   |   |   |   ├── 10_multiline_text.html
|   |   |   |   ├── 11_custom_style.html
|   |   |   |   ├── 12_scalable_images.html
|   |   |   |   ├── 13_dashed_lines.html
|   |   |   |   ├── 14_dot_language.html
|   |   |   |   ├── 15_dot_language_playground.html
|   |   |   |   ├── 16_dynamic_data.html
|   |   |   |   ├── 17_network_info.html
|   |   |   |   ├── 18_fully_random_nodes_clustering.html
|   |   |   |   ├── 19_scale_free_graph_clustering.html
|   |   |   |   ├── 20_navigation.html
|   |   |   |   ├── 21_data_manipulation.html
|   |   |   |   ├── 22_les_miserables.html
|   |   |   |   ├── 23_hierarchical_layout.html
|   |   |   |   ├── 24_hierarchical_layout_userdefined.html
|   |   |   |   ├── 25_physics_configuration.html
|   |   |   |   ├── 26_staticSmoothCurves.html
|   |   |   |   ├── 27_world_cup_network.html
|   |   |   |   ├── 28_world_cup_network_performance.html
|   |   |   |   ├── 29_neighbourhood_highlight.html
```

```
├── 30_importing_from_gephi.html
├── 31_localization.html
├── 32_hierarchicaLayoutMethods.html
├── 33_animation.html
├── 34_circular_images.html
├── 35_label_stroke.html
├── 36_HTML_in_Nodes.html
├── 37_label_alignment.html
├── 38_node_as_icon.html
├── data
│   └── WorldCup2014.json
├── graphviz
│   ├── data
│   │   ├── fsm.gv.txt
│   │   ├── hello.gv.txt
│   │   ├── process.gv.txt
│   │   ├── siblings.gv.txt
│   │   ├── softmaint.gv.txt
│   │   ├── traffic_lights.gv.txt
│   │   ├── transparency.gv.txt
│   │   ├── twopi2.gv.txt
│   │   ├── unix.gv.txt
│   │   └── world.gv.txt
│   ├── graphviz_gallery.html
│   └── screenshots
│       ├── fsm.png
│       ├── hello.png
│       ├── softmaint.png
│       └── traffic_lights.png
├── img
│   ├── indonesia
│   │   ├── 10.png
│   │   ├── 11.png
│   │   ├── 12.png
│   │   ├── 13.png
│   │   ├── 14.png
│   │   ├── 1.png
│   │   ├── 2.png
│   │   ├── 3.png
│   │   ├── 4.png
│   │   ├── 5.png
│   │   ├── 6.png
│   │   ├── 7.png
│   │   ├── 8.png
│   │   └── 9.png
│   ├── refresh-cl
│   │   ├── Hardware-Fax-icon.png
│   │   ├── Hardware-Laptop-1-icon.png
│   │   ├── Hardware-Mobile-Phone-icon.png
│   │   ├── Hardware-My-Computer-3-icon.png
│   │   ├── Hardware-My-PDA-02-icon.png
│   │   ├── Hardware-My-PDA-04-icon.png
│   │   ├── Hardware-My-PDA-05-icon.png
│   │   ├── Hardware-My-Phone-Picture-icon.png
│   │   ├── Hardware-Printer-Blue-icon.png
│   │   ├── license.txt
│   │   ├── Misc-Scanner-default-icon.png
│   │   └── Network-Drive-icon.png
```

```
│   │   │   │   │   ├── Network-Internet-Connection-icon.png
│   │   │   │   │   ├── Network-Pipe-icon.png
│   │   │   │   │   ├── ros.png
│   │   │   │   │   ├── System-Firewall-2-icon.png
│   │   │   │   │   └── System-Globe-icon.png
│   │   │   │   └── soft-scraps-icons
│   │   │   │       ├── Document-icon24.png
│   │   │   │       ├── Document-icon32.png
│   │   │   │       ├── Document-icon48.png
│   │   │   │       ├── Email-icon24.png
│   │   │   │       ├── Email-icon32.png
│   │   │   │       ├── Email-icon48.png
│   │   │   │       ├── Folder-icon24.png
│   │   │   │       ├── Folder-icon32.png
│   │   │   │       ├── Folder-icon48.png
│   │   │   │       ├── Folder-icon64.png
│   │   │   │       ├── license.txt
│   │   │   │       ├── Smiley-Angry-icon.png
│   │   │   │       ├── Smiley-Grin-icon.png
│   │   │   │       ├── User-Administrator-Blue-icon.png
│   │   │   │       ├── User-Administrator-Green-icon.png
│   │   │   │       ├── User-Coat-Blue-icon.png
│   │   │   │       ├── User-Coat-Green-icon.png
│   │   │   │       ├── User-Coat-Red-icon.png
│   │   │   │       ├── User-Executive-Green-icon.png
│   │   │   │       ├── User-Preppy-Blue-icon.png
│   │   │   │       └── User-Preppy-Red-icon.png
│   │   │   └── index.html
│   │   └── timeline
│   │       ├── 01_basic.html
│   │       ├── 02_interactive.html
│   │       ├── 03_performance.html
│   │       ├── 04_html_data.html
│   │       ├── 05_groups.html
│   │       ├── 06_event_listeners.html
│   │       ├── 07_custom_time_bar.html
│   │       ├── 08_edit_items.html
│   │       ├── 09_order_groups.html
│   │       ├── 10_limit_move_and_zoom.html
│   │       ├── 11_points.html
│   │       ├── 12_custom_styling.html
│   │       ├── 13_past_and_future.html
│   │       ├── 14_group_performance.html
│   │       ├── 15_item_class_names.html
│   │       ├── 16_navigation_menu.html
│   │       ├── 17_data_serialization.html
│   │       ├── 18_range_overflow.html
│   │       ├── 19_localization.html
│   │       ├── 20_click_to_use.html
│   │       ├── 21_set_selection.html
│   │       ├── 22_window_adjustment.html
│   │       ├── 23_data_attributes.html
│   │       ├── 24_all_data_attributes.html
│   │       ├── 25_background_areas.html
│   │       ├── 26_external_data.html
│   │       ├── 27_templates.html
│   │       ├── 29_hiding_times.html
│   │       └── 30_subgroups.html
```

```
|   |   |       ├── 31_background_areas_with_groups.html
|   |   |       ├── 32_grid_styling.html
|   |   |       ├── 33_custom_snapping.html
|   |   |       ├── 34_add_custom_timebar.html
|   |   |       ├── 35_item_ordering.html
|   |   |       ├── data
|   |   |       |   ├── basic.json
|   |   |       |   └── wk2014.json
|   |   |       ├── img
|   |   |       |   ├── attachment-icon.png
|   |   |       |   ├── blog-post-edit-icon.png
|   |   |       |   ├── comments-icon.png
|   |   |       |   ├── community-users-icon.png
|   |   |       |   ├── Hardware-Mobile-Phone-icon.png
|   |   |       |   ├── license_aesthetica-2.txt
|   |   |       |   ├── license_refresh-cl.txt
|   |   |       |   ├── license.txt
|   |   |       |   ├── mail-icon.png
|   |   |       |   ├── notes-edit-icon.png
|   |   |       |   ├── product-icon.png
|   |   |       |   └── truck-icon.png
|   |   |       ├── index.html
|   |   |       └── requirejs
|   |   |           ├── requirejs_example.html
|   |   |           └── scripts
|   |   |               ├── main.js
|   |   |               └── require.js
|   |   ├── HISTORY.md
|   |   ├── LICENSE-APACHE-2.0
|   |   ├── LICENSE-MIT
|   |   ├── NOTICE
|   |   └── README.md
|   ├── zeroclipboard
|   |   ├── jquery.zclip.js
|   |   ├── jquery.zclip.min.js
|   |   ├── ZeroClipboard.js
|   |   └── ZeroClipboard.swf
|   └── ZeroClipboard.swf
├── lib
|   ├── jackson
|   |   ├── jackson-annotations-2.6.4.jar
|   |   ├── jackson-core-2.6.4.jar
|   |   ├── jackson-databind-2.6.4.jar
|   |   ├── jackson-dataformat-avro-2.6.4.jar
|   |   ├── jackson-dataformat-cbor-2.6.4.jar
|   |   ├── jackson-dataformat-csv-2.6.4.jar
|   |   ├── jackson-dataformat-protobuf-2.6.4.jar
|   |   ├── jackson-dataformat-smile-2.6.4.jar
|   |   ├── jackson-dataformat-xml-2.6.4.jar
|   |   ├── jackson-dataformat-yaml-2.6.4.jar
|   |   ├── jackson-datatype-guava-2.6.4.jar
|   |   ├── jackson-datatype-jdk7-2.6.4.jar
|   |   ├── jackson-jr-all-2.6.4.jar
|   |   ├── jackson-module-jaxb-annotations-2.6.4.jar
|   |   ├── LICENSE.md
|   |   ├── README.md
|   |   └── stax2-api-3.1.2.jar
|   ├── java_websocket.jar
```

```
|   ├── org.json
|   |   └── json-20151123.jar
|   └── sqlite-jdbc-3.7.2.jar
├── redis-2.6.9.tar.gz
├── SkyBlue
|   ├── EndUserManager
|   |   ├── DoLogin.class
|   |   ├── DoLogout.class
|   |   ├── Session$BSTSession.class
|   |   ├── Session$BSTSession$Node.class
|   |   ├── Session.class
|   |   ├── Session$SessionInfo.class
|   |   └── TimeoutAutoCounter.class
|   ├── LCP
|   |   ├── EDLoggingControl
|   |   |   ├── ED_conn.class
|   |   |   ├── ED_ConnectionManagement.class
|   |   |   ├── ED_ConnectionManagement$ClearDeadedConnED.class
|   |   |   ├── ED_database$BST.class
|   |   |   ├── ED_database$BST$Node.class
|   |   |   ├── ED_database.class
|   |   |   ├── ED_database$ED_data.class
|   |   |   ├── EDLCP.class
|   |   |   ├── LogDecompressor
|   |   |   |   ├── decompressJNI.class
|   |   |   |   ├── libdecompressBinlog3.so
|   |   |   |   ├── logReaderAndDecompressJNI.o
|   |   |   |   └── StoreDataToKVS.class
|   |   |   ├── ObjectLoadBalancing.class
|   |   |   ├── redisControl.class
|   |   |   ├── TwoWaysCommunicationThread.class
|   |   |   ├── TwoWaysCommunicationThread$TwoWaysCommunication.class
|   |   |   ├── UserCtrl_thread.class
|   |   |   └── VideoStreaming.class
|   |   ├── EndUserManager
|   |   |   ├── DoLogin.class
|   |   |   ├── DoLogout.class
|   |   |   ├── DoRegister.class
|   |   |   ├── Session$BSTSession.class
|   |   |   ├── Session$BSTSession$Node.class
|   |   |   ├── Session.class
|   |   |   ├── Session$SessionInfo.class
|   |   |   └── TimeoutAutoCounter.class
|   |   ├── Httpd
|   |   |   ├── core
|   |   |   |   ├── BackupLog.class
|   |   |   |   ├── configDat.class
|   |   |   |   ├── configReader.class
|   |   |   |   ├── GC.class
|   |   |   |   ├── HttpAccessLogger$1.class
|   |   |   |   ├── HttpAccessLogger.class
|   |   |   |   ├── Httpd.class
|   |   |   |   ├── HTTPReceiveHandlerGarbager.class
|   |   |   |   ├── HTTPRecieveHandlerInfo.class
|   |   |   |   ├── HttpRecieveRequestHandler.class
|   |   |   |   ├── InputStreamExtend.class
|   |   |   |   ├── RosConnGC.class
|   |   |   |   ├── ServerSetting.class
```

```
│   │   │   │   └── ThreadedServerUserHandler.class
│   │   │   ├── Methods
│   │   │   │   ├── AuthDatabase$AuthInfo.class
│   │   │   │   ├── AuthDatabase$BSTAuth.class
│   │   │   │   ├── AuthDatabase$BSTAuth$Node.class
│   │   │   │   ├── AuthDatabase.class
│   │   │   │   ├── Authentication.class
│   │   │   │   ├── AuthTableList$AuthTable.class
│   │   │   │   ├── AuthTableList.class
│   │   │   │   ├── EmbeddedLog.class
│   │   │   │   ├── EmbeddedLog_opt.class
│   │   │   │   ├── ETLog.class
│   │   │   │   ├── FileFilter.class
│   │   │   │   ├── Get.class
│   │   │   │   ├── HeaderReader.class
│   │   │   │   ├── HeaderReader$Header.class
│   │   │   │   ├── HeaderResponse.class
│   │   │   │   ├── HeaderResponse$Header.class
│   │   │   │   ├── HeaderVal.class
│   │   │   │   ├── HttpMethodsUtil.class
│   │   │   │   ├── methodCall.class
│   │   │   │   ├── methodResponse.class
│   │   │   │   ├── Methods.class
│   │   │   │   ├── MIME.class
│   │   │   │   ├── MIMEType.class
│   │   │   │   ├── Options.class
│   │   │   │   ├── ParamParser.class
│   │   │   │   ├── ParamParser$paramElement.class
│   │   │   │   ├── Post.class
│   │   │   │   ├── ProtocolUpgrader.class
│   │   │   │   ├── QueryElement.class
│   │   │   │   ├── QueryStringAnalysis.class
│   │   │   │   └── SendData.class
│   │   │   └── utils
│   │   │       ├── FileDat.class
│   │   │       ├── memCached$Buckets.class
│   │   │       ├── memCached.class
│   │   │       ├── memCached$HashElements.class
│   │   │       ├── memCachedHashTBL.class
│   │   │       └── RosHandler.class
│   │   ├── Security
│   │   │   └── SkyBlueCipher.class
│   │   ├── Tunnel
│   │   │   ├── StreamConnector.class
│   │   │   ├── TunnelChannel.class
│   │   │   ├── TunnelChannelSessionInfo.class
│   │   │   └── Tunnel.class
│   │   └── Websocketd
│   │       ├── Encryption.class
│   │       ├── Websocket.class
│   │       └── WsStreamer.class
│   └── lib
│       └── sqlite-jdbc-3.7.2.jar
├── startRedis.sh
└── stopAllServer.sh

210 directories, 999 files
```

1.1.2 Blue-Sky の skycoder のコードを保存するサーバリスト

./Blue-Sky_codeReceiver

```
-rw-rw-r--  1 praween praween   761  9 月  26  2015 LICENSE
-rw-rw-r--  1 praween praween    15  9 月  26  2015 README.md
-rw-rw-r--  1 praween praween  11K 11 月  10  2015 codeReceiver.js
-rw-rw-r--  1 praween praween   339  9 月  26  2015 exec_codeReceiver.js
drwxrwxr-x 49 praween praween 4.0K 10 月  12  2015 node_modules
-rw-rw-r--  1 praween praween   402 10 月  12  2015 package.json
```

1.1.3 Blue-Sky の skycoder の通信情報のデータを保存するサーバリスト

./Blue-Sky_commReceiver

```
-rw-rw-r--  1 praween praween   761  2 月   7  2016 LICENSE
-rw-rw-r--  1 praween praween    15  2 月   7  2016 README.md
-rw-rw-r--  1 praween praween   11K  2 月   8  2016 commReceiver.js
-rw-rw-r--  1 praween praween   438 11 月   2 21:52 exec_commReceiver.js
drwxrwxr-x 90 praween praween 4.0K  2 月   7  2016 node_modules
-rw-rw-r--  1 praween praween   408  2 月   7  2016 package.json
```

1.1.4 Blue-Sky の skycoder のユーザの管理サーバリスト

./Blue-Sky_manager

```
-rw-rw-r--   1 praween praween 5.0K  7 月  15 16:15 enduser.db
-rw-rw-r--   1 praween praween  21K  7 月  15 16:29 enduser_managements.js
drwxrwxr-x 118 praween praween 4.0K  7 月   9 03:59 node_modules
-rw-rw-r--   1 praween praween  554  7 月  15 17:33 package.json
-rw-r--r--   1 praween praween 2.0K  7 月   8 16:05 test.db
-rw-rw-r--   1 praween praween  189  7 月  15 17:32 test_manager.js
-rw-rw-r--   1 praween praween  189  7 月  15 18:38 run_manager.js
```

付録A.3　サーバの起動停止方法

1.1.1，1.1.2，1.1.3，1.1.4 の各サーバの起動手順や停止方法を述べる．

付録A.3.1　Blue-Sky の Core サーバの起動停止方法

1.1.1 の Blue-Sky の Core サーバの起動停止は次の手順である．

1. **Httpd.cfg**: Core サーバにおける機能の設定ファイルである．初期状態は，wwwroot_path のマクロを設定する必要である．直接にエディターで修正する．付録 A.4 の 25 行目の“WWWROOT_PATH ”マクロを“/home/{yourUserName}/{yourBlue-SkyDir}/Inetpub”に設定する．
2. **startRedis.sh**：Redis[4]を起動させるファイルである．初期状態は，“redis-2.6.9.tar.gz”を解凍する必要である．解凍コマンドは“tar -xvfz redis-2.6.9.tar.gz”である．本ファイルを実行して redis を起動する．
3. **Httpd.sh**：サーバを通常に起動させるファイルである．フォントグラウドに起動するので，サーバの起動中の応答メッセージの確認が可能であるが，サーバの負荷を高くすることがあるのでお勧めしないが，サーバが通常に起動するかを確認するためには本ファイルを実行するのにお勧めする．
4. **HttpdAsDeamon.sh**：サーバをデーモンに起動させるファイルである．これを実行して本サーバが起動する．
5. **stopAllServer.sh**：起動中の Blue-Sky サーバを停止させるファイルである．本ファイルを実行すると全ての Blue-Sky サーバが完全に停止される．

[4] http://redis.io/download

付録A.3.2 Blue-Sky の skycoder のコードを保存するサーバの起動方法

1. redis が付録 A. 3. 1 と同じローカルマシンで起動する場合
   a. “redis-2. 6. 9/redis. conf” の “port” 番号を付録 A. 3. 1 と異なる番号 (x) に修正する. この x は他のプログラムがバインドされていないことを確認した上に修正することである.
   b. “redis-2. 6. 9/redis. conf” の “port” 番号を付録 A. 3. 1 と異なる番号 (x) に修正する. この x は他のプログラムがバインドされていないことを確認した上に修正することである.
   c. 1. 1. 2 の “exec_codeReceiver. js” を下記の通りに修正する.

```
var exec = new codeReceiver({redis:{host:"0.0.0.0",port:x}});
```

   d. 1. 1. 2 の “exec_codeReceiver. js” を “nodejs” で実行する.
2. redis が付録 A. 3. 1 と異なるマシンで起動する場合
   a. redis を通常に起動する.
   b. 1. 1. 2 の “exec_codeReceiver. js” を下記の通りに修正する. なお, x. x. x. x は redis の起動マシンの IP アドレスである.

```
var exec = new codeReceiver({redis:{host:"x.x.x.x",port:6379}});
```

   c. 1. 1. 2 の “exec_codeReceiver. js” を “nodejs” で起動する.

付録A.3.3 Blue-Sky の skycoder の通信情報のデータを保存するサーバの起動方法

1. redis が付録 A. 3. 1 と同じローカルマシンで起動する場合
   a. “redis-2. 6. 9/redis. conf” の “port” 番号を付録 A. 3. 1 と異なる番号（x）に修正する．この x は他のプログラムがバインドされていない番号を確認した上に修正することである．
   b. “redis-2. 6. 9/redis. conf” の “port” 番号を付録 A. 3. 1 と異なる番号（x）に修正する．この x は他のプログラムがバインドされていないことを確認した上に修正することである．
   c. 1. 1. 3 の “exec_commReceiver. js” を下記の通りに修正する．

```
var exec = new commReceiver({redis:{host:"0.0.0.0",port:x}});
```

   d. 1. 1. 3 の “exec_commReceiver. js” を “nodejs” で起動する．
2. redis が付録 A. 3. 1 と異なるマシンで起動する場合
   a. redis を通常に起動する．
   b. 1. 1. 3 の “exec_commReceiver. js” を下記の通りに修正する．なお，x. x. x. x は redis の起動マシンの IP アドレスである．

```
var exec = new commReceiver({redis:{host:"x.x.x.x",port:6379}});
```

   c. 1. 1. 3 の “exec_commReceiver. js” を “nodejs” で起動する．

付録A.3.4 Blue-Sky の skycoder のユーザの管理サーバの起動方法

1. sqlite3 の開発ライブラリを nodejs が扱えるように設定する.

```
sudo apt-get install libsqlite3-dev
```

2. 1.1.4 の “run_manager.js” の dbPath を修正する. dbPath は 1.1.1 の “enduser.db” の絶対パーツに修正することである.

```
var setting = {dbPath/enduser.db};
```

3. 付録 A.3 で Blue-Sky の Core サーバ(以下, bGateway)を起動した後に, nodejs で “run_manager.js” を起動する.
4. “http://bGateway:port/manager.html” をブラウザで開くものである.

付録A.4 Blue-Sky の core サーバのコンフィグレーション

Httpd.cfg

```
#-------------------------------------------------------------------------------
# Blue-Sky configure
#-------------------------------------------------------------------------------
# This is configuration file for setting the Httpd server up to be initialized the
# SkyBlue package.
#
#
# Created date: 10 Aug 2011
# Last modified date: 1 July 2016
#-------------------------------------------------------------------------------
# Email: not001praween001@gmail.com
#-------------------------------------------------------------------------------


# The server package configure can be set by rewrite the config macro following.

#-------------------------------------------------------------------------------
# Config macro: WWWROOT_PATH
#-------------------------------------------------------------------------------
# This is the root directory of Blue-sky server. Please, set it default as your
# Blue-sky server directory in ${BLUESKY_ABSOLUTE_PATH}
# $> pwd
# Using the path result from above command line as the ${ BLUESKY_ABSOLUTE_PATH}
# concat with "/Inetpub".
WWWROOT_PATH = /home/praween/Inetpub

#-------------------------------------------------------------------------------
# Config macro: INDEX_FILE
#-------------------------------------------------------------------------------
# Set the default index file.
# You can set the file name for locating and directing to the first page or
# homepage. The file should not be included '/' or '¥' or special escape character
# except '.' and ','
INDEX_FILE = index.htm, index.html

#-------------------------------------------------------------------------------
# Config macro: HTTPD_PORT
#-------------------------------------------------------------------------------
# This is the port for listening the server.
HTTPD_PORT = 8189

#-------------------------------------------------------------------------------
# Config macro: CLOSE_PASS
#-------------------------------------------------------------------------------
# The password is used when you do remote shutdown the server with remote command
# line instruction of the server.
CLOSE_PASS = 01014321dcba0101
```

```
54
55  #-------------------------------------------------------------------------------
56  # Config macro: HTTP_ACCESS_LOG_ENABLE
57  #-------------------------------------------------------------------------------
58  # The macro is to enable HTTP accessing log feature.
59  # For example:
60  HTTP_ACCESS_LOG_ENABLE = false
61
62
63
64
65  #-------------------------------------------------------------------------------
66  # Config macro: HTTP_ACCESS_LOG_PATH
67  #-------------------------------------------------------------------------------
68  # Define the path store HTTP access file. The macro will be working when set the
69  # "HTTP_ACCESS_LOG_ENABLE = true".
70  # You need to change it to your directory when you enable the macro
71  # "HTTP_ACCESS_LOG_ENABLE = true".
72  HTTP_ACCESS_LOG_PATH = /home/praween
73
74  #-------------------------------------------------------------------------------
75  # Config macro: BSFLOG_PATH
76  #-------------------------------------------------------------------------------
77  # Note: About embedded logging system of log file. (Compressed Log ftrace)
78  # BSFLOG_PATH is the path that store BSF log from specific embedded device.
79  # You need to change here to your appreciated directory.
80  BSFLOG_PATH = /home/praween/ftrace/praween/BSFLog
81
82  #-------------------------------------------------------------------------------
83  # Config macro: LOGEXTWORKING_PATH
84  #-------------------------------------------------------------------------------
85  # Log extraction working path.
86  # You need to change here to your appreciated directory.
87  LOGEXTWORKING_PATH = /home/praween/embeddedLoggingSystemWK
88
89  #-------------------------------------------------------------------------------
90  # Config macro: CACHED_PATH
91  #-------------------------------------------------------------------------------
92  # Contents cached path.
93  # You need to change here to your appreciated directory.
94  CACHED_PATH = /home/praween/Inetpub/skyblueCached
95
96  #-------------------------------------------------------------------------------
97  # Config macro: INMEMORYFILECACHED
98  #-------------------------------------------------------------------------------
99  # The in-memory cached file is the the virtual file that is using the memory
100 # mapping optimized feature. We can enable or disble this feature by set the macro
101 # to be "true" or "false".
102 # INMEMORYFILECACHED = true
103 INMEMORYFILECACHED = false
104
105
106 #-------------------------------------------------------------------------------
107 # Config macro: SHOW_MES
108 #-------------------------------------------------------------------------------
109 # This macro is for developer only. If you not the developer, set it to be "false".
110 SHOW_MES = true
111 #SHOW_MES = false
112
113 #-------------------------------------------------------------------------------
114 # Config macro: AUTH
115 #-------------------------------------------------------------------------------
116 # This is a "basic authentication" macro configure. The path of the web-root
117 # directory can be set by this macro following the rule.
118 # @RULE:
119 #   AUTH = path1:username1:password1, path2:username2:password2, ...
120 # @RULE DESCRIPTION:
121 #   The "path1" is the relative directory on web-root directory that defines at
122 #   the macro WWWROOT_PATH authentication path. The "username1" and "password1" are
123 #   the username and password of the "path1". The rule allows you to be chaining
124 #   define the authentication path by join it with ",".
125 #
126 #AUTH = needAuth:test:test, needAuth2:test2:test2, /:test:test, auth.html:test:test
127 AUTH = needAuth:test:test, needAuth2:test2:test2
128
129 #-------------------------------------------------------------------------------
130 # Embedded Device logger configure
131 #-------------------------------------------------------------------------------
132 # Enable KVS mode. It use redis for storing data.
133 # Notice: You must start KVS server before start this server.
134 KVS_MODE = true
135 KVS_DBTYPE = redis
136 KVS_SERVERIP = 127.0.0.1
137 KVS_SERVERPORT = 6379
138
139
140 # Store bsf log to redis.
141 SAVEBSFTOKVS = false
142
143
144
```

```
# Here is websocket protocol forwarding feature setting macro section.
# Server forwards the connection to the defined destination when client
# request to upgrade protocol to websocket protocol.
WS_SERVERIP = 127.0.0.1
WS_SERVERPORT = 8080

# FTRACE access Filtering feature
ED_FTRACE_AUTH_ENABLE = true

#-------------------------------------------------------------------------------
# Skycoder relational configure
#-------------------------------------------------------------------------------
# NOTE: The deploying server have to run before SkyBlue Httpd server.
#       The deploying server located at "./deploymentModules/node/exec_deployer.js"
DEPLOYMENT = true
DEPLOYERIP = 172.16.4.73
DEPLOYERPORT = 8393
CODERECEIVERIP = 127.0.0.1
CODERECEIVERPORT = 8595
# Communication receiver
COMMURECEIVER = true
COMMURECEIVERIP = 172.16.4.73
COMMURECEIVERPORT = 8596
# Extension of Enduser Database Manager
# MEMO: the enduser database server was created as an extension feature support
# nodejs
#      locate at
# [../virture_device_managements/enduser_managements/enduser_managements.js]
#          run it with: [nodejs test_manager.js] or [forever test_manager.js]
#          config the enduser_managements.js with specify options in
[test_manager.js]
ENDUSERDBEXT = true
ENDUSERDBEXTIP = 172.16.4.73
ENDUSERDBEXTPORT = 11111

#-------------------------------------------------------------------------------
# Extension configuration for ROSCORE
#-------------------------------------------------------------------------------
ROSEXT = true
ROSCOREIP = 127.0.0.1
ROSCOREPORT = 11311
BLUESKYROSPROXYIP = 127.0.0.1
BLUESKYROSPROXYPORT = 8993


```

付録 B. Blue-Sky モジュールのマニュアル

付録B.1 まえがき

Blue-Sky モジュールは本システムの組込み機器において起動する組込むソフトウェイである．本ソフトウェイを組込み機器に組み込んだことで，本システムの IoT を構築するものになる．付録 A に従ってサーバを起動して，本モジュールを組込んだ組込み機器を起動することである．本モジュールは次のとおりに構成される．

- **組込み機器に関する I/O の制御モジュール**：
  本モジュールは著者が拓殖大学の早川研究室の学生所属の際に開発した組み込むソフトウェイである．本モジュールは **Blue-Sky** サーバにおける仮想インスタンスという ED ノードを生成するのに必要な情報を提供する．さらに ED ノードへのセンシング・アクチュエションを行う際に，Linux の GPIO，PWM などのシステムインターフェースを **Blue-Sky** サーバの命令に従って制御するものである．本モジュールを初期設定して以来，組込み機器に電源を入れただけで，システムが起動するものである．
- **組込み機器に付け加えたウェブカメラモジュール**：
  本モジュールは IP ベースネットワークを用いる JPEG フレームのウェブ IP カメラである mjpeg_streamer[5] を用いた．**Skycoder** でウェブカメラを再生するようにするには組込み機器で本モジュールを起動する必要である．

[5] https://github.com/jacksonliam/mjpg-streamer

付録B.2 Blue-Sky モジュールのファイルリスト

付録 B. 1 で述べた Blue-Sky モジュールの実行可能なファイルを一覧する.

付録B.2.1 組込み機器に関する I/O の制御モジュールのファイルリスト

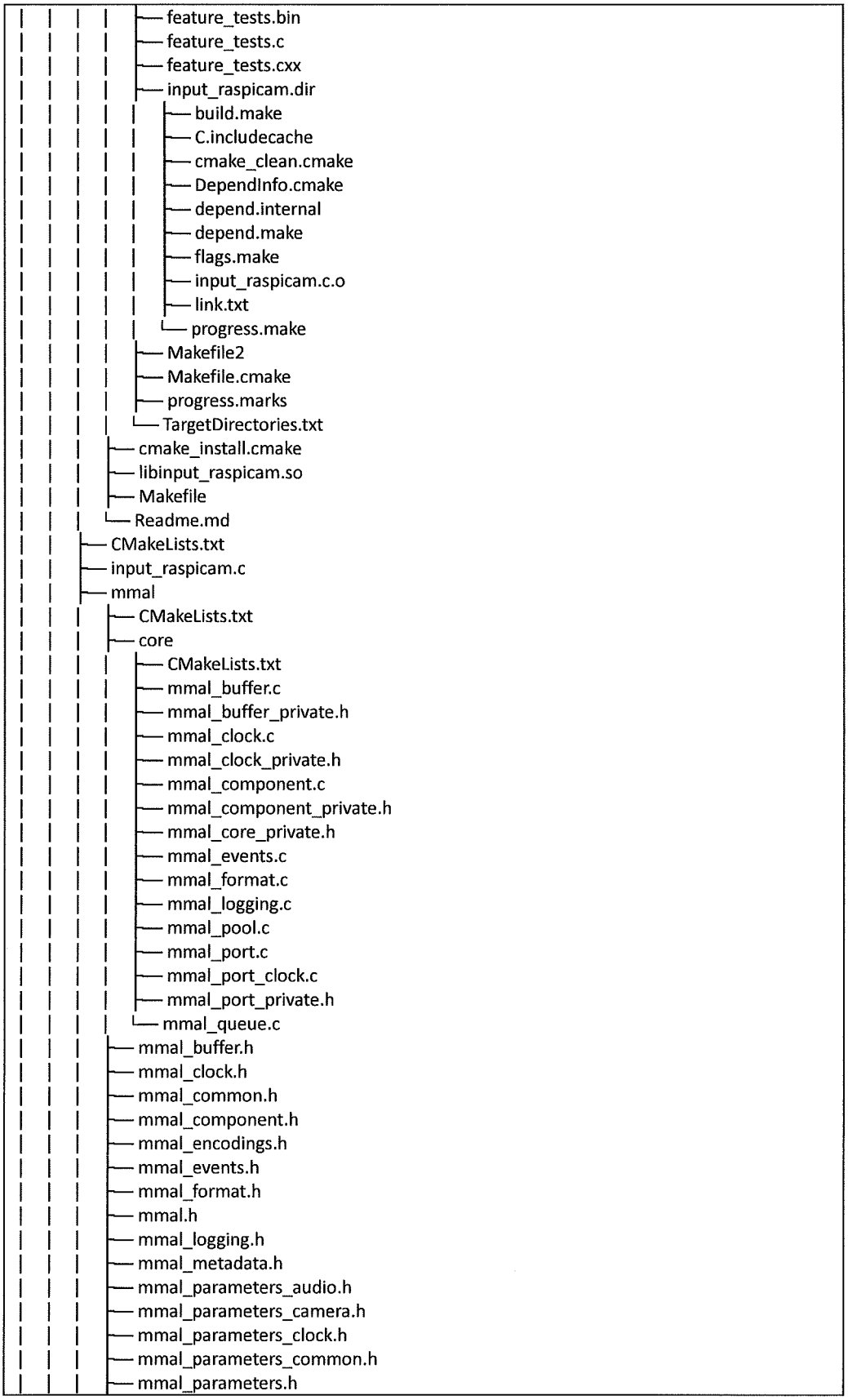
```
/skyblue
├── Build.txt
├── GPIO-GainerSerial-modules
│   ├── bin
│   │   ├── configDat.class
│   │   ├── configReader.class
│   │   ├── GainerControl.class
│   │   ├── GainerController.class
│   │   ├── GainerTestJava.class
│   │   ├── gmJNI.class
│   │   ├── GPIO.class
│   │   ├── GPIOTYPES.class
│   │   ├── GPS.class
│   │   ├── GPS_GT_723F.class
│   │   ├── ModuleRegister.class
│   │   ├── PushingData.class
│   │   ├── PWM.class
│   │   ├── PWMTYPES.class
│   │   ├── SensorPoint.class
│   │   ├── SensorPointHandling.class
│   │   ├── SkyblueModuleDat.class
│   │   ├── skymod.cfg
│   │   └── SPI.class
│   ├── run.sh
│   └── shared
│       ├── libgmAin.so
│       ├── libgmAout.so
│       ├── libgmClose.so
│       ├── libgmDH.so
│       ├── libgmDL.so
│       ├── libgmDout.so
│       └── libgmOpen.so
├── init.d
│   ├── skyblue
│   └── skyblue-module
├── install
│   ├── install-pi-blaster.sh
│   └── install-skyblue.sh
├── install.sh
└── LCP-connector
    ├── bin
    │   ├── ConnectToLCP.class
    │   ├── EDAccessToken.class
    │   ├── EDUtilityInfoHandlingCenter.class
    │   ├── EDUtilityInfoHandlingCenter$EDUtilityInfoHandler.class
    │   ├── SkyBlueCipher.class
    │   ├── SkyBlueConnectToLCP.class
    │   └── UserCommandHadler.class
    ├── javaPath
    └── run.sh

7 directories, 42 files
```

付録B.2.2　組込み機器に付け加えたウェブカメラモジュールのファイルリスト

/mjpg_streamer

```
├── CHANGELOG
├── input_raspicam.so
├── input_uvc.so
├── LICENSE
├── Makefile
├── mjpg_streamer
├── mjpg_streamer.c
├── mjpg_streamer.h
├── mjpg_streamer.o
├── output_file.so
├── output_http.so
├── plugins
│   ├── input_control
│   │   ├── dynctrl.c
│   │   ├── dynctrl.h
│   │   ├── input_uvc.c
│   │   ├── Makefile
│   │   ├── uvc_compat.h
│   │   └── uvcvideo.h
│   ├── input_file
│   │   ├── input_file.c
│   │   └── Makefile
│   ├── input.h
│   ├── input_http
│   │   ├── input_http.c
│   │   ├── Makefile
│   │   ├── misc.c
│   │   ├── misc.h
│   │   ├── mjpg-proxy.c
│   │   ├── mjpg-proxy.h
│   │   └── version.h
│   ├── input_ptp2
│   │   ├── input_ptp2.c
│   │   ├── input_ptp2.h
│   │   └── Makefile
│   ├── input_raspicam
│   │   ├── build
│   │   │   ├── CMakeCache.txt
│   │   │   ├── CMakeFiles
│   │   │   │   ├── 3.2.0-rc2
│   │   │   │   │   ├── CMakeCCompiler.cmake
│   │   │   │   │   ├── CMakeCXXCompiler.cmake
│   │   │   │   │   ├── CMakeDetermineCompilerABI_C.bin
│   │   │   │   │   ├── CMakeDetermineCompilerABI_CXX.bin
│   │   │   │   │   ├── CMakeSystem.cmake
│   │   │   │   │   ├── CompilerIdC
│   │   │   │   │   │   ├── a.out
│   │   │   │   │   │   └── CMakeCCompilerId.c
│   │   │   │   │   └── CompilerIdCXX
│   │   │   │   │       ├── a.out
│   │   │   │   │       └── CMakeCXXCompilerId.cpp
│   │   │   │   ├── cmake.check_cache
│   │   │   │   ├── CMakeDirectoryInformation.cmake
│   │   │   │   ├── CMakeOutput.log
│   │   │   │   ├── CMakeTmp
```


feature_tests.bin
feature_tests.c
feature_tests.cxx
input_raspicam.dir
build.make
C.includecache
cmake_clean.cmake
DependInfo.cmake
depend.internal
depend.make
flags.make
input_raspicam.c.o
link.txt
progress.make
Makefile2
Makefile.cmake
progress.marks
TargetDirectories.txt
cmake_install.cmake
libinput_raspicam.so
Makefile
Readme.md
CMakeLists.txt
input_raspicam.c
mmal
CMakeLists.txt
core
CMakeLists.txt
mmal_buffer.c
mmal_buffer_private.h
mmal_clock.c
mmal_clock_private.h
mmal_component.c
mmal_component_private.h
mmal_core_private.h
mmal_events.c
mmal_format.c
mmal_logging.c
mmal_pool.c
mmal_port.c
mmal_port_clock.c
mmal_port_private.h
mmal_queue.c
mmal_buffer.h
mmal_clock.h
mmal_common.h
mmal_component.h
mmal_encodings.h
mmal_events.h
mmal_format.h
mmal.h
mmal_logging.h
mmal_metadata.h
mmal_parameters_audio.h
mmal_parameters_camera.h
mmal_parameters_clock.h
mmal_parameters_common.h
mmal_parameters.h

```
│   │   │   ├── mmal_parameters_video.h
│   │   │   ├── mmal_pool.h
│   │   │   ├── mmal_port.h
│   │   │   ├── mmal_queue.h
│   │   │   ├── mmal_types.h
│   │   │   ├── util
│   │   │   │   ├── CMakeLists.txt
│   │   │   │   ├── mmal_component_wrapper.c
│   │   │   │   ├── mmal_component_wrapper.h
│   │   │   │   ├── mmal_connection.c
│   │   │   │   ├── mmal_connection.h
│   │   │   │   ├── mmal_default_components.h
│   │   │   │   ├── mmal_graph.c
│   │   │   │   ├── mmal_graph.h
│   │   │   │   ├── mmal_il.c
│   │   │   │   ├── mmal_il.h
│   │   │   │   ├── mmal_list.c
│   │   │   │   ├── mmal_list.h
│   │   │   │   ├── mmal_param_convert.c
│   │   │   │   ├── mmal_param_convert.h
│   │   │   │   ├── mmal_util.c
│   │   │   │   ├── mmal_util.h
│   │   │   │   ├── mmal_util_params.c
│   │   │   │   ├── mmal_util_params.h
│   │   │   │   ├── mmal_util_rational.c
│   │   │   │   └── mmal_util_rational.h
│   │   │   └── vc
│   │   │       ├── CMakeLists.txt
│   │   │       ├── mmal_vc_api.c
│   │   │       ├── mmal_vc_api_drm.c
│   │   │       ├── mmal_vc_api_drm.h
│   │   │       ├── mmal_vc_api.h
│   │   │       ├── mmal_vc_client.c
│   │   │       ├── mmal_vc_client_priv.h
│   │   │       ├── mmal_vc_msgnames.c
│   │   │       ├── mmal_vc_msgnames.h
│   │   │       ├── mmal_vc_msgs.h
│   │   │       ├── mmal_vc_opaque_alloc.c
│   │   │       ├── mmal_vc_opaque_alloc.h
│   │   │       ├── mmal_vc_shm.c
│   │   │       └── mmal_vc_shm.h
│   │   ├── RaspiCamControl.c
│   │   └── RaspiCamControl.h
│   ├── input_testpicture
│   │   ├── input_testpicture.c
│   │   ├── Makefile
│   │   ├── pictures
│   │   │   ├── 160x120_1.jpg
│   │   │   ├── 160x120_2.jpg
│   │   │   ├── 320x240_1.jpg
│   │   │   ├── 320x240_2.jpg
│   │   │   ├── 640x480_1.jpg
│   │   │   ├── 640x480_2.jpg
│   │   │   ├── 960x720_1.jpg
│   │   │   └── 960x720_2.jpg
│   │   └── testpictures.h
│   ├── input_uvc
│   │   ├── dynctrl.c
```

```
|   |   |-- dynctrl.h
|   |   |-- dynctrl.lo
|   |   |-- huffman.h
|   |   |-- input_uvc.c
|   |   |-- input_uvc.so
|   |   |-- jpeg_utils.c
|   |   |-- jpeg_utils.h
|   |   |-- jpeg_utils.lo
|   |   |-- Makefile
|   |   |-- uvc_compat.h
|   |   |-- uvcvideo.h
|   |   |-- v4l2uvc.c
|   |   |-- v4l2uvc.h
|   |   `-- v4l2uvc.lo
|   |-- output_autofocus
|   |   |-- Makefile
|   |   |-- output_autofocus.c
|   |   |-- processJPEG_onlyCenter.c
|   |   `-- processJPEG_onlyCenter.h
|   |-- output_file
|   |   |-- examples
|   |   |   |-- change_filename.sh
|   |   |   |-- ftp_upload.sh
|   |   |   `-- show_filename.sh
|   |   |-- Makefile
|   |   |-- output_file.c
|   |   |-- output_file.h
|   |   `-- output_file.so
|   |-- output.h
|   |-- output_http
|   |   |-- httpd.c
|   |   |-- httpd.h
|   |   |-- httpd.lo
|   |   |-- Makefile
|   |   |-- output_http.c
|   |   `-- output_http.so
|   |-- output_rtsp
|   |   |-- Makefile
|   |   |-- output_rtsp.c
|   |   `-- rtsp.c
|   |-- output_udp
|   |   |-- Makefile
|   |   `-- output_udp.c
|   `-- output_viewer
|       |-- Makefile
|       `-- output_viewer.c
|-- README
|-- README-UNSTABLE
|-- scripts
|   `-- make_deb.sh
|-- start.sh
|-- stream.sh
|-- TODO
|-- utils.c
|-- utils.h
|-- utils.o
`-- www
    |-- bodybg.gif
```

```
    ├── cambozola.jar
    ├── control.htm
    ├── example.jpg
    ├── favicon.ico
    ├── favicon.png
    ├── fix.css
    ├── functions.js
    ├── index.html
    ├── java_control.html
    ├── java.html
    ├── javascript.html
    ├── javascript_motiondetection.html
    ├── javascript_simple.html
    ├── java_simple.html
    ├── jquery.js
    ├── jquery.rotate.js
    ├── JQuerySpinBtn.css
    ├── JQuerySpinBtn.js
    ├── jquery.ui.core.min.js
    ├── jquery.ui.custom.css
    ├── jquery.ui.tabs.min.js
    ├── jquery.ui.widget.min.js
    ├── LICENSE.txt
    ├── rotateicons.png
    ├── sidebarbg.gif
    ├── spinbtn_updn.gif
    ├── static.html
    ├── static_simple.html
    ├── stream.html
    ├── stream_simple.html
    ├── style.css
    └── videolan.html

29 directories, 232 files
```

付録B.3 Blue-Sky モジュールの初期設定方法

付録 B. 2. 1，付録 B. 2. 2 の初期設定方法について述べる．

付録B.3.1　組込み機器に関する I/O の制御モジュールの初期設定

1. 付録 B.2.1 の“install.sh”を実行する．
2. 付録 B.2.1 の“skymod.cfg”を初期設定する．本設定は付録 B.4 の“MODULE_GPIO_ALLOWPINS”マクロは，GPIO のアクセス権限を指定するものである．組込み機器にアクセス不可能な GPIO 番号はここに書き込まないようにする．付録 B.4 の“MODULE_GPIO_ALLOWPINS”に示す例は，RaspberryPi のアクセス可能な gpio ピン番号である．
3. ユーザ名を組込み機器に付け加える設定を行う．
    a. ユーザ名を組込み機器に付け加える設定をしない場合は skycoder でアクセスする時に username:“guest”，password:“guest”でアクセスすることになる．
    b. ユーザ名を組込み機器に付け加える設定をする場合は付録 B.2.1 の“LCP-connector/bin”に行って，“EDAccessToken.class”を java で実行すると，トークンメッセージが表示する．これをコピーしておく．

```
cd LCP-connector/bin
# The token keyword is unnecessary for remembrance in your brain,
# not like password
java EDAccessToken [input your token keyword here]
```

    c. 付録 A.3.4 を起動した上に，“http://bGateway:port/manager.html”をブラウザで開き，ユーザ名とパスワードを指定した後に，コピーしたトークンメッセージを edAccessToken の空欄を入力して，保存する．
    d. 組込み機器を再起動する．
    e. 付録 A.3.1 に従って Core サーバを起動して，skycoder を別の PC のブラウザで開くことである．Skycoder のアクセス URL は次のとおりである．
        - “http://bGateway:port/skycoder.html”
          ノードの可視化やタイムラインで可視化のない初期バージョンである．

- “http://bGateway:port/skycoder_includedGraph.html”
  ノードの可視化やタイムラインで可視化のあるバージョンである.
- “http://bGateway:port/skycoder_includedGraph_rosext.html”
  ROS の拡張を行ったバージョンである.

付録B.3.2　組込み機器に付け加えたウェブカメラモジュールの初期設定

1. 付録 B.2.2 のルーツディレクトリに行って，ウェブカメラモジュールをコンパイルする.

```
cd 付録 B.2.2 のルーツディレクトリ
make
sudo make install
```

2. 付録 B.2.2 の“stream.sh”というカメラ起動ファイルを初期設定する．付録 B.5 は初期設定の例である.
3. カメラを起動する.

```
./stream.sh
```

4. Skycoder に応じるツールバーで，カメラの初期設定をした ED ノードを選択することで，カメラを開くことである.

付録B.4 組込み機器に関する I/O の制御モジュールコンフィグレーション

skymod.cfg

```
# This is skyblue modules register configuration file.
# Configuration class schemes (Case sensitive):
#       MODULE_{User module name}=enable || disable
#       MODULE_{User module name}_CLASS={class name}
#               **{User module name}: You can establish your module
#                  name in skyblue.
# Author: Praween AMONTAMAVUT
################################################################################
# Do not change the initial feature's configuration below.
################################################################################
MODULE_GAINER=enable
MODULE_GAINER_CLASS=GainerController

MODULE_GPIO=enable
MODULE_GPIO_CLASS=SensorPoint
MODULE_GPIO_CLASS_RELATIVEOBJ=MODULE_GAINER
MODULE_GPIO_ALLOWPINS=2,3,4,17,27,22,10,9,11,5,6,13,19,26,14,15,18,23,24,25,8,7,12,16,21
################################################################################
# You can add your configuration below.
################################################################################

```

付録B.5 ウェブカメラモジュールを起動するシェルファイルの設定例

stream.sh

```
#!/bin/sh
# Author: Praween Amontamavut
# Email: praween@hykwlab.org
# ========================================
# Caution: Do not change the listening PORT here.
PORT="8292"
# Setting accessing username and password here. The system asks for doing the basic
# authentication on the browser when you accessing the camera via skycoder.
ID="idforyourcamera"
PW="passwordforyourcamera"
# Check your web camera hardware specification to the support JPEG streaming size
and the frame rate setting here.
SIZE="320x240"
FRAMERATE="10"

export LD_LIBRARY_PATH=/usr/local/lib
mjpg_streamer -b ¥
        -i "input_uvc.so -f $FRAMERATE -r $SIZE -d /dev/video0 -y" ¥
        -o "output_http.so -w /usr/local/www -p $PORT -c $ID:$PW"
```

付録 C. Blue-Sky の Core サーバにおける Rest API のリスト

本節は，Blue-Sky の Core サーバが提供する API を一覧する．Skycoder に関する Rest API のリストと圧縮を備えたトレース機構に対する Rest API のリストの二種類がある．

付録C.1 Skycoder に関する Rest API のリスト

Rest API のリスト		
1.	URL	http://bGateway:port/ETLog?instruction=ls&opt1=noneFix&opt2=edconnected
	説明	ED ノードの接続状態のデータを取得する．本 API はログイン認証が不要である．
2.	URL	http://bGateway:port/ETLog?instruction=videostreaming&opt1=[edNodeIP]&opt2=mjpgstreamer&opt3=8292&opt4=Lz9hY3Rpb249c25hcHNob3Q=
	説明	付録 B.3.2 に応じる初期された ED ノードに搭載されたウェブカメラをストリミング開始させる API である．
3.	URL	http://bGateway:port/ETLog?instruction=wherecodereceiver&opt1=noneFix&opt2=json
	説明	Skycoder におけるコードエディターに組んだ JavaScript ベースコードをデプロイする先を問い合わせる API である．応答データ形式は JSON 形式である．
4.	URL	http://bGateway:port/ETLog?instruction=wherecodereceiver&opt1=noneFix&opt2=xml
	説明	Skycoder におけるコードエディターに組んだ JavaScript ベースコードをデプロイする先を問い合わせる API である．応答データ形式は xml 形式である．
5.	URL	http://bGateway:port/ETLog?instruction=wherecommureceiver&opt1=noneFix&opt2=json
	説明	Skycoder におけるタイムスタンプを押したデータの通信シーケンスの保存先を問い合わせる API である．応答データ形式は json 形式である．
6.	URL	http://bGateway:port/ETLog?instruction=wherecommureceiver&opt1=noneFix&opt2=xml
	説明	Skycoder におけるタイムスタンプを押したデータの通信シーケンスの

		保存先を問い合わせる API である．応答データ形式は xml 形式である．
7.	URL	http://bGateway:port/ETLog?instruction=systemload&opt1=noneFix&opt2=json
	説明	サーバの負荷を問い合わせる API である．応答データ形式は json 形式である．
8.	URL	http://bGateway:port/ETLog?instruction=systemload&opt1=noneFix&opt2=xml
	説明	サーバの負荷を問い合わせる API である．応答データ形式は xml 形式である．
9.	URL	http://bGateway:port/ETLog?instruction=sensornetwork&opt1=[edNodeIP]&opt2=gpio&opt3=set&opt4=[GPIO pin number]&opt5=[0 or 1]
	説明	指定される ED ノードの GPIO ピンに対して，0 または 1 に信号を出力する．
10.	URL	http://bGateway:port/ETLog?instruction=sensornetwork&opt1=[edNodeIP]&opt2=gpio&opt3=get&opt4=[GPIO pin number]&opt5=[0 or 1]
	説明	指定される ED ノードの GPIO ピンの信号値を取得する．
11.	URL	http://bGateway:port/ETLog?instruction=sensornetwork&opt1=[edNodeIP]&opt2=pwm&opt3=set&opt4=[GPIO pin number]&opt5=[0 ~ 1.0 is a ratio of PWM in floating point number]
	説明	指定される ED ノードの GPIO ピンに対して，指定された 0~1.0 の PWM の周波数の割合で信号を出力する．
13.	URL	http://bGateway:port/ETLog?instruction=actuatorcontrol&opt1=[edNodeIP]&opt2=pwm&opt3=set&opt4=[GPIO pin number]&opt5=[0 ~ 1.0 is a ratio of PWM in floating point number]
	説明	アクチュエーションを行うために，指定される ED ノードの GPIO ピンに対して，指定された 0~1.0 の PWM の周波数の割合で信号を出力する．
14.	URL	http://bGateway:port/ETLog?instruction=sensornetwork&opt1=[edNodeIP]&

		opt2=spi&opt3=mcp3208&opt4=10&opt5=9&opt6=11&opt7=8
	説明	指定される ED ノードにおいて，mosi をピン 10 に，miso をピン 9 に，clk をピン 11 に，ce ピンを 8 に配線された mcp3208 の A/D 変換の SPI を利用したセンサのデータを取得する.

付録C.2　圧縮を備えたトレース機構に対する Rest API のリスト

Rest API のリスト		
1.	URL	http://bGateway:port/ETLog?instruction=startlogging&opt1=[edNodeIP]
	説明	指定される ED ノードに対して，本研究のトレース機構の ftrace を開始させる.
2.	URL	http://bGateway:port/ETLog?instruction=stoplogging&opt1=[edNodeIP]
	説明	指定される ED ノードに対して，本研究のトレース機構の ftrace を停止させる.
3.	URL	http://bGateway:port/ETLog?instruction=getLog&opt1=[edNodeIP]&opt2=strbin
	説明	指定される ED ノードに対して，BSF のトレースデータを取得する.
4.	URL	http://bGateway:port/ETLog?instruction=extractLog&opt1=[edNodeIP]&opt2=json
	説明	指定される ED ノードに応じる BSF のトレースデータを json 形式のトレースデータに解凍させる.
5.	URL	http://bGateway:port/ETLog?instruction=extractLog&opt1=[edNodeIP]&opt2=xml
	説明	指定される ED ノードに応じる BSF のトレースデータを xml 形式のトレースデータに解凍させる.
6.	URL	http://bGateway:port/ETLog?instruction=getLog&opt1=[edNodeIP]&opt2=json
	説明	指定される ED ノードに応じて，解凍された json 形式のトレースデータを取得する.
7.	URL	http://bGateway:port/ETLog?instruction=getLog&opt1=[edNodeIP]&opt2=xml
	説明	指定される ED ノードに応じて，解凍された xml 形式のトレースデータを取得する.
8.	URL	http://bGateway:port/ETLog?instruction=downloadlog&opt1=[edNodeIP]&opt2=json
	説明	指定される ED ノードに応じて，解凍された json 形式のトレースデータをダウンロードする.
9.	URL	http://bGateway:port/ETLog?instruction=downloadlog&opt1=[edNodeIP]&opt2=xml

	説明	指定される ED ノードに応じて，解凍された xml 形式のトレースデータをダウンロードする.
10.	URL	http://bGateway:port/ETLog?instruction=getLog&opt1=[edNodeIP]&opt2=json&opt3=20130222&opt4=3~5
	説明	指定される ED ノードに応じて，2013 年 2 月 22 日にトレースをした BSF のトレースデータを json 形式に解凍させたトレースデータを 3 行目から 5 行目の部分に取得する.
11.	URL	http://bGateway:port/ETLog?instruction=getLog&opt1=[edNodeIP]&opt2=xml&opt3=20130222&opt4=3~5
	説明	指定される ED ノードに応じて，2013 年 2 月 22 日にトレースをした BSF のトレースデータを xml 形式に解凍させたトレースデータを 3 行目から 5 行目の部分に取得する.
12.	URL	http://bGateway:port/ETLog?instruction=getLog&opt1=[edNodeIP]&opt2=json&opt3=20130222&opt4=contentpieces
	説明	指定される ED ノードに応じて，2013 年 2 月 22 日にトレースをした BSF のトレースデータを json 形式に解凍させたトレースデータの全ての行数を問い合わせる.
13.	URL	http://bGateway:port/ETLog?instruction=getLog&opt1=[edNodeIP]&opt2=xml&opt3=20130222&opt4=contentpieces
	説明	指定される ED ノードに応じて，2013 年 2 月 22 日にトレースをした BSF のトレースデータを xml 形式に解凍させたトレースデータの全ての行数を問い合わせる.
14.	URL	http://bGateway:port/ETLog?instruction=ls&opt1=noneFix&cts
	説明	BSF 形式のトレースデータを一覧する.
15.	URL	http://bGateway:port/ETLog?instruction=ls&opt1=noneFix&json
	説明	JSON 形式のトレースデータを一覧する.
16.	URL	http://bGateway:port/ETLog?instruction=ls&opt1=noneFix&xml
	説明	XML 形式のトレースデータを一覧する.

付録 D. Skycoder の JavaScript の関数の API リスト

コマンド名	パラメター		機能
lsn	第1： デバイスIPの指定	デバイスのローカルIP	Localで接続する各デバイスがセンシング・アクチュエーティング機能をPrivate APIとして呼び出すLocalコマンドである.
	第2： サブシステムの指定	gpio	
		pwm	
		spi	
	第3： 各サブシステムのパラメター	GPIO: setまたはget	
		PWM: set	
		SPI: AD変換の型番（例：mcp3028)	
	第4： 各サブシステムのパラメター	GPIO: ピン番号の指定	
		PWM: ピン番号の指定	
		SPI: MOSI ピン番号の指定	
	第5： 各サブシステムのパラメター	GPIO: 0と1の値の指定	
		PWM: 0から1の範囲の浮動小数の指定	
		SPI: MISOピン番号の指定	
	第6： 各サブシステムのパラメター	SPI: CLKピン番号の指定	
	第7： 各サブシステム	SPI: CEピン番号の指定	

	のパラメター		
	第8： 各サブシステム のパラメター	SPI: AD変換のチャンネル(CH)番号の指定（有無可)	
gsn	第1： Publicドメイン名・IP宛の指定	BlueSkyのクラウドサーバが運用するゲートウェイのドメインである.	Localで接続する各デバイスがセンシング・アクチュエーティング機能をPublic APIとして呼び出すGlobalコマンドである. アクセスするゲートウェイのPublic Domainが必要である.
	第2： デバイスIPの指定	デバイスのローカルIP	
	第3： サブシステムの指定	gpio	
		pwm	
		spi	
	第4： 各サブシステム のパラメター	GPIO: setまたは get	
		PWM: set	
		SPI: AD変換の型番（例：mcp3028)	
	第5： 各サブシステム のパラメター	GPIO: ピン番号の指定	
		PWM: ピン番号の指定	
		SPI: MOSI ピン番号の指定	
	第6： 各サブシステム のパラメター	GPIO: 0と1の値の指定	
		PWM: 0から1の範囲の浮動小数の指定	
		SPI:	

		MISOピン番号の指定	
	第7： 各サブシステム のパラメター	SPI: CLKピン番号の指定	
	第8： 各サブシステム のパラメター	SPI: CEピン番号の指定	
	第9： 各サブシステム のパラメター	SPI: AD変換のチャンネル(CH)番号の指定（有無可)	
loop	繰り返す回数		コマンドを繰り返す.
sleep	スリープタイム		ミリ秒単位でスリープさせる.

付録 E. JavaScript ベースの関数

関数名	引数		機能
l_sensornet work	第1： デバイスIPの指定	デバイスのローカルIP	Localで接続する各デバイスがセンシング・アクチュエーティング機能をPrivate APIとして呼び出すLocal関数である.
	第2： サブシステムの指定	gpio	
		pwm	
		spi	
	第3： 各サブシステムの引数	GPIO: setまたはget	
		PWM: set	
		SPI: AD変換の型番（例：mcp3028）	
	第4： 各サブシステムの引数	GPIO: ピン番号の指定	
		PWM: ピン番号の指定	
		SPI: 通信ピンの配列で指定 配列の要素 [MOSI, MISO, CLK, CE]のピン番号	
	第5： 各サブシステムの引数	GPIO: 0と1の値の指定	
		PWM: 0から1の範囲の浮動小数の指定	
		SPI: AD変換のチャンネル(CH)番号の指定（あってもなくてもいい）	
g_sensornet work	第1： Publicドメイン名・IP宛の指定	BlueSkyのクラウドサーバが運用するゲートウェイのドメインである.	Localで接続する各デバイスがセンシング・アクチュエーティング機能をPublic
	第2： デバイスIPの指定	デバイスのローカルIP	

	第3： サブシステムの指定	gpio	APIとして呼び出すGlobal関数である． アクセスするゲートウェイのPublic Domainが必要である．
		pwm	
		spi	
	第4： 各サブシステムの引数	GPIO: setまたはget	
		PWM: set	
		SPI: AD変換の型番（例：mcp3028）	
	第5： 各サブシステムの引数	GPIO: ピン番号の指定	
		PWM: ピン番号の指定	
		SPI: 通信ピンの配列で指定する 配列の要素 [MOSI, MISO, CLK, CE]のピン番号	
	第6： 各サブシステムの引数	GPIO: 0と1の値の指定	
		PWM: 0から1の範囲の浮動小数の指定	
		SPI: AD変換のチャンネル(CH)番号の指定（あってもなくてもいい）	
l_actuatornetwork	第1： デバイスIPの指定	デバイスのローカルIP	Localで接続する各デバイスにアクチュエーション機能をPrivate APIとして呼び出すLocal関数である．
	第2： アクチュエターの出力信号値の指定	信号は0～255の範囲である．	
g_actuatornetwork	第1： Publicドメイン名・IP宛の指	BlueSkyのクラウドサーバが運用するゲートウェイのドメインである	Localで接続する各デバイスにアク

	定	.	チュエーション機能をPublic APIとして呼び出すPublic関数である.
	第2： デバイスIPの指定	デバイスのローカルIP	
	第3： アクチュエターの出力信号値の指定	信号は0～255の範囲である.	
lightSensorDat	デバイスのローカルIPの指定		Localの光センサのデータを取得する.
lightSensorDatAt	第1： Publicドメィン名・IP宛の指定	BlueSkyのクラウドサーバが運用するゲートウェイのドメインである.	Globalの光センサのデータを取得する.
	第2： デバイスIP宛の指定	デバイスのローカルIPの指定	
onLed	デバイスのローカルIP指定		LocalのLEDを点灯させる.
onLedAt	第1： Publicドメィン名・IP宛の指定		GlobalのLEDを点灯させる.
	第2： デバイスIP宛の指定		
offLed	デバイスのローカルIP指定		LocalのLEDを消灯させる.
offLedAt	第1： Publicドメィン名・IP宛の指定		GlobalのLEDを消灯させる.
	第2： デバイスIP宛の指定		
adConverterSensor	第1： デバイスIP宛の指定		アナログ・ディジタル変換を用いたSPI 通信によるセンシングデータを取得する.
	第2： AD変換の型番の指定（例：mcp3028）		
	第3： MOSI に繋ぐ GPIO のピン番号の指定		

	第4： MISO に繋ぐ GPIO のピン番号の指定	
	第5： CLK に繋ぐ GPIO のピン番号の指定	
	第6： CE に繋ぐ GPIO のピン番号の指定	
	第7： 非同期通信の指定（true or false）．通常は false に指定する．	
sensorGraph	無し	センシングデータの統合的なモニタリング機能の表示
listEDNodes	無し	ゲートウェイサーバに接続中の ED ノードの情報を一覧する．戻る値は JSON オブジェクトである．
Sleep	スリープタイム	ミリ秒単位でスリープさせる．