特別研究報告書

# IoT 環境における状況依存型サービス連携の実現

指導教員　　石田 亨 教授

京都大学工学部情報学科

渡辺 隆弘

平成 28 年 2 月 6 日

# IoT 環境における状況依存型サービス連携の実現

渡辺 隆弘

内容梗概

近年,「いつでも，どこでも，何でも，誰でも」ネットワークに繋がる「ユビキタスネットワーク社会」が構想されてきた．様々なデバイスがネットワークに接続されるようになると，それらのデバイス間での情報交換やデータの収集，それに基づく自動化が行われ，新たな付加価値を生むようになる．しかし，現状ではセンサーはただ値を取得するだけにとどまっており，ユーザのニーズにあったデータの取得というものは行うことができない．そこで，センサーデータを用いてより複雑な処理を行うために，複合サービスとの連携を行うことを考える．センサーデータを用いて複合サービスの実行を行うことで，センサーから取得できる周囲の状況に基づいて複合サービス内のサービスを選択し，複雑な処理を実行することができる．しかし，現状では，センサーに統一された枠組みが存在せず，センサーを利用するためにはそれぞれのセンサーの仕様に基づいて別々のシステムを構築する必要がある．また，IoT 環境では，実環境のデータを利用するため，ユーザはリアルタイム性を求めるが，現在はデータをデータベースに格納し，抽出するため，リアルタイム性は実現できていない．本研究の目的は IoT 環境において状況に基づいたサービス実行を行う手法の提案と実装である．そのために，以下の 2 点の解決すべき課題が存在する．

1. センサーのサービス化

   IoT 環境では周囲のセンサーから取得されるデータや，Web から取得されるデータなど様々なフォーマットを持つデータが利用されることが考えられるため，それらの仕様の違いを気にすることなくシステムを実装できるような，統一化された枠組みを構築する必要がある．

2. リアルタイム性を保証したサービス選択

   現在は Web サービスの選択にユーザの知識や経験が必要になる．また，ユーザはリアルタイム性を要求するため，実行するサービス選択をユーザの手を介さず，またリアルタイムに行えるようにする必要がある．

上記の課題を解決するために，本研究では，センサーのサービス化手法を提案し，センサーデータに統一化された枠組みを与える．これにより，センサーデータを利用するシステムを実装する際，センサーの種類を考えることなく実装を

行うことができる．
次に，センサーデータによるリアルタイムなサービス選択手法を提案する．イベント処理システムを応用し，センサーからデータを取得した際に，事前に用意したルールに従って処理を行い，利用するサービスの選択と，サービスへの入力を複合サービスに与える．この手法により，ユーザが複合サービスを利用する際に原子サービスの選択をする必要がなくなり，ユーザの知識や経験を問わず，最適なサービス選択を行うことができる．また，イベント処理システムを利用することで，センサーデータを受け取ったと同時にサービスを実行することができ，サービスのリアルタイム性も保証できる．
最後に，これらの提案に基づくシステムを実装した．温度，湿度センサーの存在する環境下で，センサーデータによる複合サービスのサービス選択，実行を実現した．
本研究の貢献は以下の 2 点である．

1. センサーのサービス化手法の提案と実装
   センサーデータについて画一化されたデータ定義を与えることでセンサーをサービス化する手法を提案した．これにより，ユーザはセンサーの仕様を気にすることなくデータを利用することができるようになる．また，実際にセンサーサービスインターフェースを実装することで，画一された枠組みとして機能することの実例を示した．
2. リアルタイム性を保証した複合サービスのサービス選択手法の提案と実装
   センサーの値をイベントとして CEP エンジンに挿入し，リアルタイムで処理，アクションとして複合サービスへの入力生成とサービス実行を行うことによって，ユーザがサービスのリクエストを送信することなく，リアルタイムかつ自動的なサービス実行が可能となった．実際にこの手法を用いたシステムを実装することにより実例を示した．

# Realization of situated service composition in IoT environment

Takahiro Watanabe

**Abstract**

In recent years, a "ubiquitous network society" connected to a network "anytime, anywhere, whatever, anyone" has been conceived. When various devices are connected to the network, information exchange between these devices, collection of data and automation based thereon are performed, and new added value is generated. However, at present, the sensor only acquires the value, and it is impossible to acquire the data that meets the needs of the user. Therefore, in order to perform more complicated processing using sensor data, consider cooperating with compound service. By executing the composite service using the sensor data, it is possible to select a service in the composite service based on surrounding situations obtainable from the sensor, and to execute complicated processing. However, at present, there is no unified framework in the sensor, and in order to use the sensor, it is necessary to construct a separate system based on the specification of each sensor. In the IoT environment, since the data of the real environment is used, the user desires real time property, but since the data is currently stored in the database and extracted, the real time property can not be realized.

The purpose of this research is the proposal and implementation of a method to execute service based on situation in IoT environment. Therefore, the following two problems exist.

1. Construct of sensor service

   In the IoT environment, it is conceivable that data having various formats such as data acquired from surrounding sensors and data acquired from the Web are used, so that It is necessary to construct a unified framework being able to implemented the system without worrying about the difference between those specifications that can be done.

2. Service selection guaranteeing real time property

   Currently it requires user's knowledge and experience to select web service. In addition, since the user demands real-time nature, it is necessary to

select a service to be executed without intervention of the user and in real time.

In order to solve the above problem, this research proposes a service method of sensor and gives a unified framework to sensor data. As a result, when implementing a system that uses sensor data, it is possible to implement without considering the type of sensor.

Next, we propose a real-time service selection method based on sensor data. By applying the event processing system, when acquiring data from the sensor, process according to the prepared rules, select the service to use and give input to the service to the composite service. This method eliminates the need for the user to select an atomic service when using a compound service and can select an optimum service regardless of user's knowledge or experience. In addition, by using the event processing system, it is possible to execute the service at the same time that the sensor data is received, and it is also possible to guarantee real-time service of the service.

Finally, we implemented a system based on these proposals. In the environment where temperature and humidity sensor exist, we realized service selection and execution of composite service by sensor data.

The contribution of this research is the following two points.

1. Propose and implement of method of constructing sensor service

   We proposed a method to construct sensor service by providing unified data definition about sensor data. This allows the user to use the data without worrying about the specification of the sensor. Also, I showed an example of functioning as a standardized framework by actually installing a sensor service interface.

2. Propose and implement of service selection method of composite service with guaranteed real time property

   By inserting the value of the sensor as an event into the CEP engine, processing in real time, input generation to the composite service and execution of the service are performed as actions, so that the user can perform automatic service execution in real time It became possible. An actual example was shown by actually implementing a system using this method.

# IoT 環境における状況依存型サービス連携の実現

## 目次


**第 1 章　はじめに** **1**

**第 2 章　関連研究** **3**

2.1　Internet of Things . . . . . 3

2.2　Complex Event Processing . . . . . 4

2.3　サービス連携 . . . . . 4

2.3.1　言語グリッド . . . . . 5

**第 3 章　提案手法** **5**

3.1　課題点 . . . . . 5

3.2　センサーのサービス化手法 . . . . . 7

3.3　状況依存型サービス選択手法 . . . . . 10

**第 4 章　提案アーキテクチャ** **12**

4.1　アーキテクチャ図 . . . . . 12

4.2　センサーサービスインターフェース . . . . . 13

4.3　パブリッシャー . . . . . 13

4.4　プロセッササービスインターフェース . . . . . 13

4.5　CEP プロセッサ . . . . . 13

**第 5 章　実装** **14**

5.1　シチュエーション . . . . . 14

5.2　仕様 . . . . . 14

5.2.1　センサーデバイス部 . . . . . 14

5.2.2　ユーザデバイス部 . . . . . 14

5.2.3　メイン部 . . . . . 16

5.2.4　複合サービス部 . . . . . 18

5.3　考察 . . . . . 20

**第 6 章　終わりに** **21**

**謝辞** **22**

**参考文献** **22**

**付録 A-1**
A.1 デバイスでセンサーデータを取得し、サーバーへ送信するモジュールのソースコード . . . . . A-1
A.1.1 MyviewController.java . . . . . A-1
A.1.2 WaikikiSensor.java . . . . . A-4
A.2 受信したデータをルールエンジンに挿入し、状況に応じた出力を得るモジュールのソースコード . . . . . A-5
A.2.1 ObservationReceiverImpl.java . . . . . A-5
A.2.2 Translator.java . . . . . A-8
A.2.3 Binding.java . . . . . A-10
A.2.4 DroolsManager.java . . . . . A-11
A.2.5 DroolsUtil.java . . . . . A-12
A.2.6 TargetLanguage . . . . . A-14
A.2.7 VoiceText.java . . . . . A-15
A.2.8 TransTextToSpeech.java . . . . . A-16
A.2.9 badminton.drl . . . . . A-17
A.3 オムロンのセンサー定義 . . . . . A-29
A.3.1 EnvSensor.java . . . . . A-29
A.3.2 EnvSensorListener.java . . . . . A-33
A.3.3 EnvSensorScanner.java . . . . . A-33
A.4 OpenIoT のデータ定義 . . . . . A-36
A.4.1 Observation.java . . . . . A-36
A.4.2 ObserbationProperty.java . . . . . A-39

# 第1章　はじめに

近年,「いつでも，どこでも，何でも，誰でも」ネットワークに繋がる「ユビキタスネットワーク社会」が構想されてきた．接続機器として代表的なものとして，従来はパソコンやスマートフォンが挙げられるが，センサーデバイスの普及に伴い，車や家電といった物理機器，建物もネットワークに接続されるようになった．このように様々なデバイスがネットワークに接続されるようになると，それらのデバイス間での情報交換やデータの収集，それに基づく自動化が行われ，新たな付加価値を生むようになる．例えば，以下のような例が挙げられる．

- 離れた場所の環境を知る温度，湿度，気圧．照度といった環境をセンサーによって知ることができる．
- 物体の動きを知る物体の動き (衝撃，振動，移動など) を知ることができる．
- 物体の位置を知る物体の位置 (存在，通過など) を知ることができる．
- 機器の制御を行う空調の制御，照明の制御などを離れた場所から操作することができる．

このような仕組みは Internet of Things(IoT) と呼ばれる仕組みであり，急速に発展している．[1][2]
また，Internet of Service(IoS) と呼ばれる，Web アプリケーションやサービスを組み合わせ，新たなサービスを構成する仕組みが存在する．[3]

本研究では，IoT 環境での状況依存型サービス連携を実現することを目的とする．この目的を実現するために以下の問題点が存在する．

1. センサーの仕様の不統一性
   現状は同じ種類のセンサー (温度センサーや湿度センサーなど) でも，通信手段やデータフォーマットなどに差異がある．IoT 環境において，センサーデータを利用して Web サービスを実行することを考える．周囲の環境に存在するセンサーから値を取得しサービスを実行するが，そのセンサーの仕様が統一されていないために，それぞれのセンサーの仕様ごとにシステムの実装を行う必要がある．
2. リアルタイム性を保証したサービス選択
   リアルタイム性を保証したサービス選択として，
   - サービス選択

- リアルタイム性

の 2 点について問題がある．

これまで，複合サービス内の原子サービスの選択は，ユーザによって指定する方向で行われてきた．例えば，言語グリッドの翻訳サービスのうち，辞書翻訳を利用することを考える．言語グリッドの辞書翻訳には様々なサービスが登録されており，ユーザがどの辞書を用いるか指定する．

つまり，原子サービスの選択にユーザの知識や経験が要求されるため，問題点としては以下の二つが例として考えられる．

- ユーザが初めて複合サービスを利用する際にどのような原子サービスを利用すれば適当かが分からない
- ユーザのサービスに対しての知識が不足しているために，ユーザのサービス選択がユーザの要求に関わらず固定化されてしまい，ユーザの要求を満たすよりよい原子サービスの組み合わせがあるにもかかわらず，より質の低いサービス選択を行ってしまう

また，リアルタイム性については，IoT 環境では，周囲の環境のデータを利用するという特性上，リアルタイムに変化する周囲の状況をその都度サービス実行に反映させることができるようにする必要があるために生じる問題である．複合サービスは，複数の Web サービスを組み合わせたものであるため，実行の仕様は Web サービスに基づく．Web サービスはリクエストに応じてレスポンスを返す形式であるため，Web サービスを利用するためには，ユーザは Web サービスにリクエストを送信する必要がある．また，現在の IoT 環境では，収集したデータをデータベースに格納し，そこからデータを抽出するという形式であるために，リアルタイム性は確保されていない．

以上の問題点があげられる．

本研究では，センサーのサービス化手法を提案し，センサーデータに統一化された枠組みを与える．これにより，センサーデータを利用するシステムを実装する際，センサーの種類を考えることなく実装を行うことができる．

次に，リアルタイム性を保証したサービス実行問題を解決するために，センサーデータによるサービス選択手法を提案する．イベント処理システムを応用し，センサーからデータを取得した際に，事前に用意したルールに従って処理を行い，利用するサービスの選択と，サービスへの入力を複合サービスに与える．この

手法により，ユーザが複合サービスを利用する際に原子サービスの選択をする必要がなくなり，ユーザの知識や経験を問わず，最適なサービス選択を行うことができる．また，イベント処理システムを利用することで，センサーデータを受け取ったと同時にサービスを実行することができ，ユーザの手を介さないサービスのリアルタイム実行も実現できる．
最後に，これらの提案に基づくシステムを実装した．温度，湿度センサーの存在する環境下で，センサーデータによる複合サービスのサービス選択，実行を実現したものである．
本稿の構成は以下である．2 章では，IoT と IoS の連携に関する先行研究について記述する．3 章では，解決すべき課題点について述べた後，提案手法としてセンサーのサービス化手法と状況依存型サービス選択手法について述べる．4 章では，3 章で提案した手法を一般的なシステムに応用する際のアーキテクチャの概要について述べる．5 章では，4 章で提案したアーキテクチャを用いて実装したシステムの概要，仕様について述べたのち，動作確認とシステムについての考察を行う．

# 第 2 章　関連研究

本研究は IoT 環境において周囲の状況をリアルタイムに取得し，適切なサービス連携を行うことを目的とする．本章では IoT についての導入と，状況依存型処理の重要性，サービス連携に関しての説明を行う．

## 2.1　Internet of Things

Internet of Things(IoT) とは，様々な物理機器，建物，乗り物などにセンサーやソフトウェアを組み込むことで，情報交換やデータの収集を行えるネットワークを構築する仕組みである．[3] では，アイデンティティ，物理的属性，および仮想パーソナリティ，知的インターフェースを使用し、情報ネットワークにシームレスに統合されている物理的，もしくは仮想的な ”モノ ” に存在する標準および相互運用可能な通信プロトコルに基づく，自己構成能力を備えた動的なグローバルネットワークインフラストラクチャとして定義されている．
IoT は、物理的な世界と仮想的な世界を橋渡しすることで，スマートな都市，スマートな工場，資源管理、交通機関、健康、福利厚生など、多くのアプリケー

ション分野に影響を与える．しかし、ソフトウェアアプリケーションの中でIoTを活用することは、ネットワーキングからアプリケーション層まで，特に超大規模，極端な異質性，IoTの動的性などの大きな課題を抱えていることが指摘されている．[4] また，世界中において配備されているセンサーの数は急速に増加しており，これらのセンサーは膨大な量のデータを生成しつづけるが，それらのセンサーデータに価値を生み出すためにそれらを理解する必要がある．[5] では，IoTパラダイムにおける，状況認識の重要性を明らかにしており，流動的にデータが生産しつづけられる状況下において，状況に応じた処理を行うことの必要性が示されている．

## 2.2 Complex Event Processing

Complex Event Processing(CEP，または複合イベント処理) とは，刻々と生成されるデータをリアルタイムに処理するための方式である．事前に定義したルールに，リアルタイムにデータを挿入し，そのルールに応じて即座に処理を行う．これまでのビッグデータ分析の方法は，データをデータベースに蓄積し，任意のタイミングで参照し，分析するという手法であったために，情報の処理に時間がかかるという問題点があった．CEPは対象のデータを直近の範囲に絞り，メモリ上に読みこんで処理を行うため処理を高速化でき，”直近の数秒以内に”などの条件に沿ってデータを処理することが可能となる．本研究では，このCEPをストリーム形式であるセンサーデータに対し応用することを考える．

## 2.3 サービス連携

サービス連携とは，Internet of Service(IoS) 基盤に集積された各原子サービスを組み合わせ，ユーザの要求を満たす高い品質 (QoS，またはQuality of Service) の複合サービス (Composite Service) を構成する技術である．Webサービスは、異なるQoSにもかかわらず，重複または同一の機能を提供するため．特定の複合サービスに参加するサービスを決定する選択肢が必要となる．[6] サービス連携の手法としては，サービス連携をQoSの最適化問題とみなして扱う研究 [7][8]，ユーザを中心にQoSを計算する研究 [9][10] などがある．ユーザに対してどのようにサービスを組み合わせるかということが課題となっている．

### 2.3.1 言語グリッド

言語グリッド [1)][11] は機械翻訳などの言語資源を共有可能にする多言語サービス基盤である．多言語コミュニケーションに対する需要の高まりに比例して言語資源は急速に増加しているが，知的保護の問題や機能の違いにより利用可能性に優れない．その問題に対して，言語グリッドは，言語資源を共通の Web サービスの形式にサービス化する多言語基盤を実現することで解決している．[12] これにより，利用者は言語グリッドにアクセスすることで，大学や研究機関，企業が提供する言語サービスを利用し，さらにそれらのサービスを組み合わせて用いることができる．また，利用者がその目的に合わせて，新たな言語サービスを作成し登録することも可能である．本研究で実装したシステムのうち，複合サービスの部分において言語グリッドを利用しており，その中でも，形態素解析，辞書連携翻訳，音声合成サービスを利用している．

# 第 3 章　提案手法

本章では，現状の課題を説明した後に，センサーのサービス化を行うための手法と，センサーから取得したデータによって，複合サービスのサービス選択，サービス実行を自動で行うための手法を提案する．

## 3.1　課題点

状況に応じたサービス選択を行うために，センサーから取得した情報によって複合サービスへの入力を変更することを考える．その際に以下の課題点が生じる．

1. センサーの仕様の不統一性
   現状は同じ種類のセンサー (温度センサーや湿度センサーなど) でも，通信手段やデータフォーマットなどに差異がある．Web サービスを利用する際には，その場所に存在するセンサーから値を取得しサービスを実行するが，そのセンサーの仕様が統一されていなければ，それぞれのセンサーの仕様ごとにシステムの実装を行う必要が生まれる．
2. 状況に応じたサービスの選択
   これまで，複合サービス内の原子サービスの選択は，ユーザによって指定

[1)] http://langrid.org/jp/

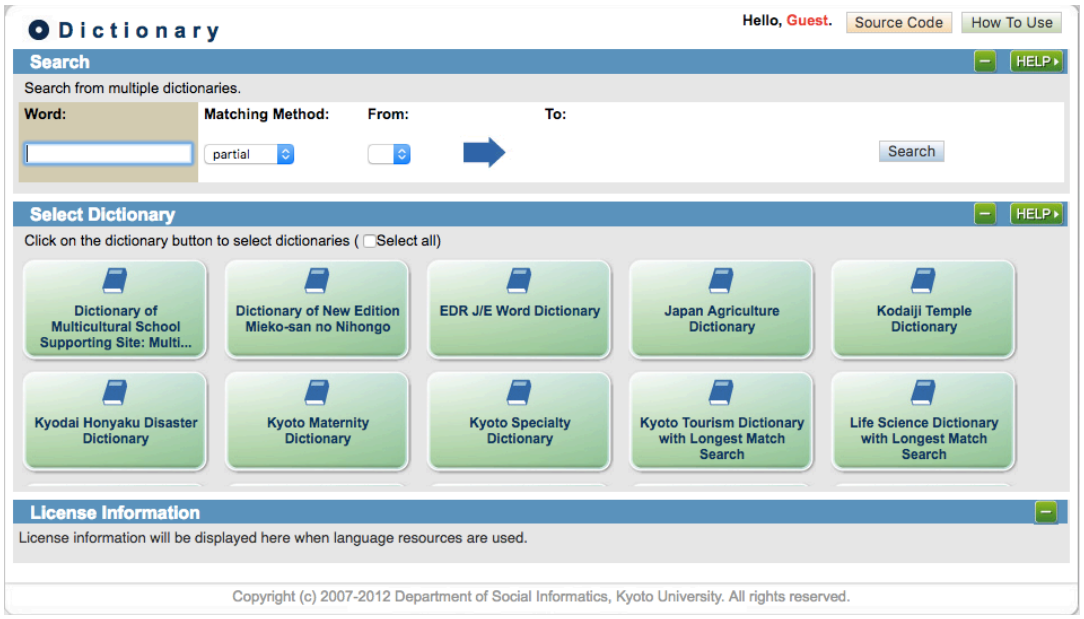


図 1: 言語グリッド playground

する方向で行われてきた．例えば，言語グリッドの翻訳サービスのうち，辞書翻訳 [1] を利用することを考える．言語グリッドの辞書翻訳サービスは図 2 のようなインターフェースになっている．様々な種類の辞書が登録されており，ユーザはどの辞書を選択するかという情報と，翻訳したい文章を入力としてサービスに与える．

つまり，原子サービスの選択にユーザの知識や経験が要求されるため，以下のような問題点が生じる．

- ユーザが初めて複合サービスを利用する際にどのような原子サービスを利用すれば適当かが分からない
- ユーザのサービスに対しての知識が不足しているために，ユーザのサービス選択がユーザの要求に関わらず固定化されてしまい，ユーザの要求を満たすよりよい原子サービスの組み合わせがあるにもかかわらず，より質の低いサービス選択を行ってしまう

3. 複合サービスのリアルタイム実行

   複合サービスは，複数の Web サービスを組み合わせたものであるため，実行の仕様は Web サービスに基づく．Web サービスはリクエストに応じてレ

[1] http://langrid.org/playground/dictionary.html

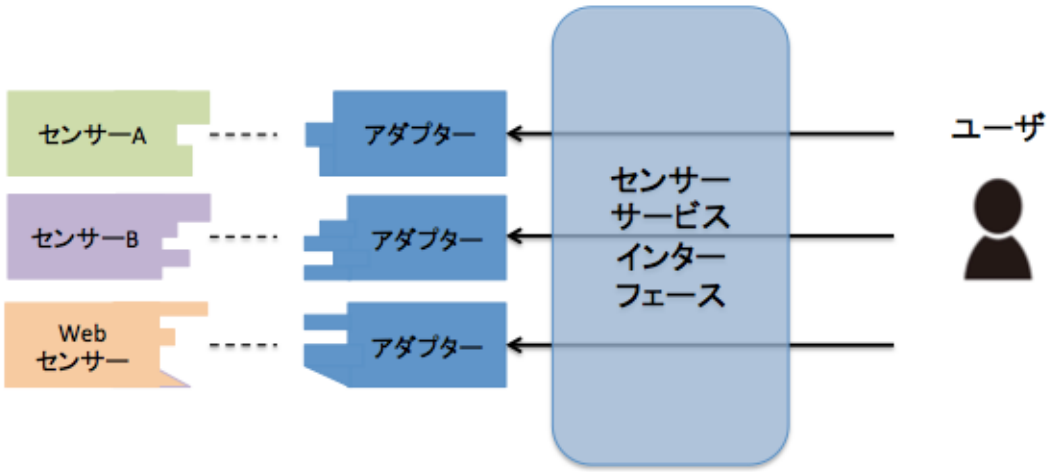


図 2: センサーのサービス化概要

スポンスを返す形式であるため，Web サービスを利用するためには，ユーザは Web サービスにリクエストを送信する必要がある．

3.2 節，3.3 節でこれらの課題点を解決するための手法を提案する．

## 3.2 センサーのサービス化手法

本節では，センサーのサービス化手法を提案する．現状は，前述した通りセンサーの仕様が画一化されていないために，センサーを利用するシステムを実装する際，センサーの種類によって異なる実装が必要であるという問題点が存在する．この問題点を本提案は解決する．ここで，2.3.1 章に述べた言語グリッドにおける言語資源のサービス化手法を応用する．言語グリッドにおいては，複雑な契約や知的財産，データ構造やインターフェースの多様性を持つ言語資源をサービス化し共有する多言語基盤を実現している．サービスの集合知を形成する枠組みをサービスグリッドと呼び，そのためのミドルウェアが開発されている．[13]

本研究では，言語グリッドにおける言語資源をセンサー資源に置き換える．センサー資源も言語資源と同様に，複雑なデータフォーマットやインターフェースの多様性を有している．共通のセンサーサービスインターフェースを開発することで，これらの多様性を持つセンサー資源をサービス化し，共有することが可能になる．

概要を図 3.2 にしめす．現状，センサー資源はそれぞれ異なった仕様を持って

いる．たとえば，同じ温度を取得するものであっても，Bluetooth による通信によってデータを送信するものや，Web センサーとして HTTP の形式でデータを与えるものがある．このためユーザは，それらのデータを利用しようとする際にそれぞれ異なる要求を行わなければならず，煩雑である．本提案では，センサーサービスインターフェースとしてユーザが画一化して利用できるインターフェースを設ける．センサー資源それぞれに対しアダプターという，センサー資源を画一化したインターフェースへ変換する機構を実装することでセンサーのサービス化手法として実現される．具体的にはデータ定義として OpenIoT[1)] のセンサー定義を用い，種々のセンサー資源から取得したデータをこのデータ形式に再形成する．OpenIoT はオープンソースで実装されている IoT プラットフォームである．OpenIoT ではセンサーから取得したデータをミドルウェアを通じてデータベースに格納している．OpenIoT のセンサー定義の例は以下のソースコード A.4.1 であり，Observation オブジェクトとして実装される．データの値，取得時間や，温度，湿度，照度といったデータタイプを示す propertyType などが存在する．

```
1  //Observation
2
3    private String id;
4    private Date times;
5    private String sensorId;
6    private String featureOfInterest="";
7    private ArrayList<ObservedProperty> readings;
8    private String metaGraph;
9    private String dataGraph;
10
11
12 //ObservedProperty
13
14   private static final long serialVersionUID = 1L;
15   private Object value;
16   private Date times;
```

[1)] http://www.openiot.eu/

```
17    private String propertyType;
18    private String unit;
19    private String observationId;
```

ソースコード 1: センサー定義例

センサーの開発者は，センサーから値を取得した際に，Observation を作成し，各変数に取得した値を格納するようにサービスを構成する．システム開発者はこのサービスの仕様に従ってシステムを実装することで，ユーザからは種々のセンサー間の違いは隠蔽され，画一化されたセンサーサービスとしてデータを利用することができる．例えば，センサーから温度 20 ℃，湿度 50%のデータを取得した際には以下のソースコード 2 のように Observation を生成する．

```
1  Observation o = new Observation();      //
       Observationオブジェクトの作成
2  ArrayList<ObservedProperty> readings = new ArrayList<
       ObservedProperty>();       //ObservedPropertyのリストの作成
3  ObservedProperty tempProperty = new ObservedProperty();      //
       ObservedPropertyオブジェクトの作成
4  ObservedProperty humdProperty = new ObservedProperty();
5  tempProperty.setPropertyType("http://openiot.eu/ontology/ns/
       AirTemperature");        //propertyTypeの設定
6  humdProperty.setPropertyType("http://openiot.eu/ontology/ns/
       AtmosphereHumidity");
7  tempProperty.setValue(20);      //valueに値を格納
8  humdProperty.setValue(50);
9  readings.add(tempProperty);       //
       ObservedPropertyのリストに追加
10 readings.add(humdProperty);
11 o.setReadings(readings);;      //
       Observationに作成したリストを格納
```

ソースコード 2: Observation 生成例

これにより，ユーザのリクエストは以下図 3.2 のように簡潔化される．

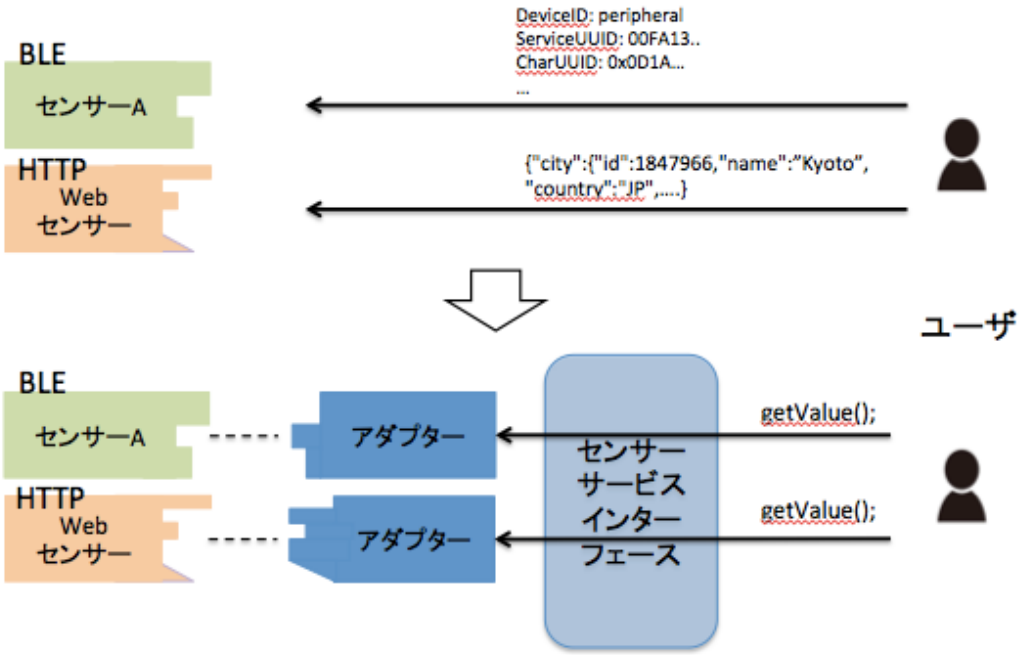


図 3: ユーザリクエスト図

### 3.3 状況依存型サービス選択手法

本節では，センサーの値によって複合サービス中の原子サービスを選択する手法を提案する．センサーから取得した値を CEP エンジンによって処理することによってこの手法は実現される．まず状況によって変化させるべき部分は以下の例が存在する．

- 状況に応じてサービスへの入力を切り替える例温度に応じてユーザに翻訳サービスを利用したメッセージを与えるとする．利用するサービスは翻訳サービスで変化はしないが，温度に応じてサービスへの入力を切り替える必要がある．
- 利用するサービスを切り替える例
- 複合サービス内の原子サービスを切り替える例

これらの状況依存型選択を行うために，ECA ルールを応用することを考える．ECA ルールとは，アクティブデータベースにおいて自動的に実行する処理を定義するためのルール定義であり，

**E** ：Event

**C** ：Condition

**A** ：Action

の 3 つからなる．イベントが発生した際，その状況に応じてアクションを実行する，というルールの実行を行う．

本研究では，ECA ルールを CEP エンジンで実現する．つまり，ECA ルールを以下のように適用する．

**E** ：センサーからのデータの取得

**C** ：センサーから取得した値

**A** ：選択するサービスとサービスへの入力の生成，サービスの実行

以上から，センサーからデータを取得した際，サーバーから CEP エンジンにデータを挿入し，事前に定義されたルールに基づいて，選択するサービスとサービスへの入力の生成とサービスの実行を行うという一連の処理が実行される．また，本研究では CEP エンジンにおいて適用するルールは事前に定義されているものとし，状況に応じてどのような処理を実行すべきかというルールの構成の点についての議論は行わない．

この手法により以下の 2 点の問題点が解決される．

1. 複合サービス内の原子サービスの選択

   ユーザがサービス選択を行わなければならないという問題点が存在した．一方，本提案では，専門家が一度ルールを作成すれば，センサーの値によって分岐するルールに従って原子サービスの選択を行うことができ，サービス連携においてユーザのサービスに対しての知識や経験に関わらず一定の質の高いサービス合成が可能となる．

2. 複合サービスのリアルタイム実行

   サービス実行のためにユーザは Web サービスにリクエストを送信する必要があった．一方，本提案では，センサーの値をイベントとして CEP エンジンに挿入し，リアルタイムで処理，アクションとして複合サービスへの入力生成とサービス実行を行うことによって，ユーザがサービスのリクエストを送信することなく，リアルタイムかつ自動的なサービス実行が可能となる．

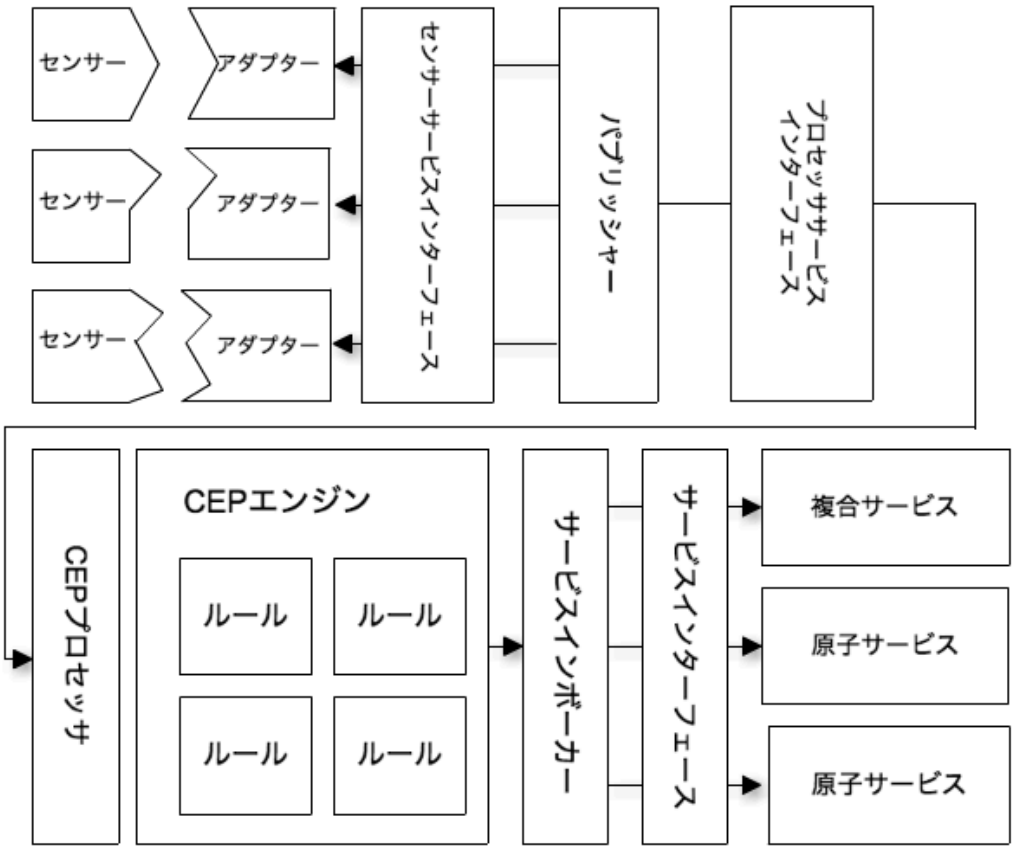


図 4: アーキテクチャ図

# 第 4 章　提案アーキテクチャ

本章では，前章に説明した提案手法に基づいて，IoT 環境下で複合サービスの選択，実行を行うシステムのアーキテクチャの提案を行う．アーキテクチャは大きく分けてセンサーデバイス，ユーザデバイス，複合サービスの 3 つの領域に分割される．ユーザデバイスの内部にセンサーからのデータを受け取りサーバーへ送信するレシーバー，データを受け取り，イベントを構成して CEP エンジンに挿入するサーバー，システム開発者が規定したルールに基づいて挿入されたイベントの処理を行い，複合サービスへ入力を与える CEP エンジンが存在する．

## 4.1　アーキテクチャ図

提案するアーキテクチャのアーキテクチャ図を以下 4 に示す．

## 4.2　センサーサービスインターフェース

センサーサービスインターフェースは 3.2 節で説明したセンサーのサービス化手法を用いて，種々のセンサー間の差異を覆い隠すラッパーとしての役割を果たす．ユーザデバイスがセンサーから値を取得する際の統一された枠組みとして存在し，レシーバーに対し規定された形でデータを送信する．

## 4.3　パブリッシャー

パブリッシャーはセンサーサービスインターフェースを介してそれぞれのセンサーからデータを取得するレシーバーとしての役割と，データを処理部である CEP プロセッサに送信する役割を持つ．パブリッシャーがデータを取得するためのリクエストは，センサーサービスインターフェースの仕様に基づき一元化されている．

## 4.4　プロセッササービスインターフェース

サーバーに挿入されたイベントを，規定されたルールに基づいて処理する．ルールはシステムの設計者が自由に定めることができ,「あるイベントが起きた際に何らかの処理が行われる」という形の ECA ルールで規定される．ルールエンジンはイベントの処理の結果，利用する複合サービスの仕様に基づいて入力を生成し，複合サービスへ与える．例として複合サービス内の選択サービスとサービスへの入力が挙げられる．例として，言語グリッドの場合は，

**選択サービス**　辞書翻訳，使用する辞書

**入力**　翻訳元言語，翻訳先言語．翻訳したい文章

を言語グリッドに与えると，辞書翻訳を利用できる．イベント挿入後，イベントの処理を行うタイミングは任意であり，挿入と同時に処理を行うことでリアルタイムな複合サービスの実行が実現できる．

## 4.5　CEP プロセッサ

複合サービスは構成要素として複数の原子サービスまたは複合サービスがあり，それぞれのサービスを組み合わせて新たなサービスを実行することができる．CEP エンジンから入力が与えられたら，サービスを実行し，ユーザに出力として与える．

# 第5章　実装

本章では，前章に提案したアーキテクチャの実装について説明し，動作確認と評価について述べる．最後に実装の結果に対して考察を行う．

## 5.1　シチュエーション

体育館を利用するユーザに，温度，湿度などの情報から運動への助言を音声で与えるシステムを実装することを考える．ユーザは様々な言語圏のユーザが想定されるため，それぞれのユーザが利用する言語に基づいてアナウンスを行う必要がある．体育館には温度センサー，湿度センサーが存在しており，システムはそれらのセンサーから取得したデータと，ユーザが自身の使用する言語を自身の持つデバイスによりシステムに与えた入力に基づいてサービスを実行する．音声はユーザデバイスではなくシステムを含むコンピュータから出力される．

## 5.2　仕様

Java を用いて実装した．以下に各モジュールの詳細を述べる．実装図は以下図 5 である．大きく分けてセンサーデバイス部，ユーザデバイス部，メイン部，複合サービス部に分けられる．

### 5.2.1　センサーデバイス部

体育館に設置することを想定するセンサーデバイスは，(株) オムロンの環境センサー[1)] とする．このセンサーによって取得できるデータタイプの中から，今回は温度データと湿度データを利用する．センサーは Bluetooth Low Energy(BLE) を用いて通信を行う．

### 5.2.2　ユーザデバイス部

ユーザデバイス部では，

- アダプター
- パブリッシャー

の 2 つを備えたアプリケーションを実装した．全体として，センサーから取得したデータと，ユーザから入力された使用言語の情報を統一されたインターフェースとして形成し，メイン部に送信する役割を果たす．構成要素は以下．

[1)] http://www.omron.co.jp/ecb/products/sensor/special/environmentsensor/

- アダプター
  取得したデータをセンサーサービスインターフェースとして OpenIoT のセンサー定義を用いてサービス化する．取得される情報は，
  - 温度データ
  - 湿度データ
  - 利用者の言語情報

  の三つである．温度データ，湿度データについては，センサーデータより値とデータタイプを取得し，Observation[A.4.1] 中の ObservationProperty[A.4.2] のうち，value と propertyType に格納する．手順は以下．
  1. Observation オブジェクトを生成する．
  2. ObservedProperty として tempProperty，humdProperty を作成する．それぞれ，温度のデータ，湿度のデータを格納するオブジェクトである

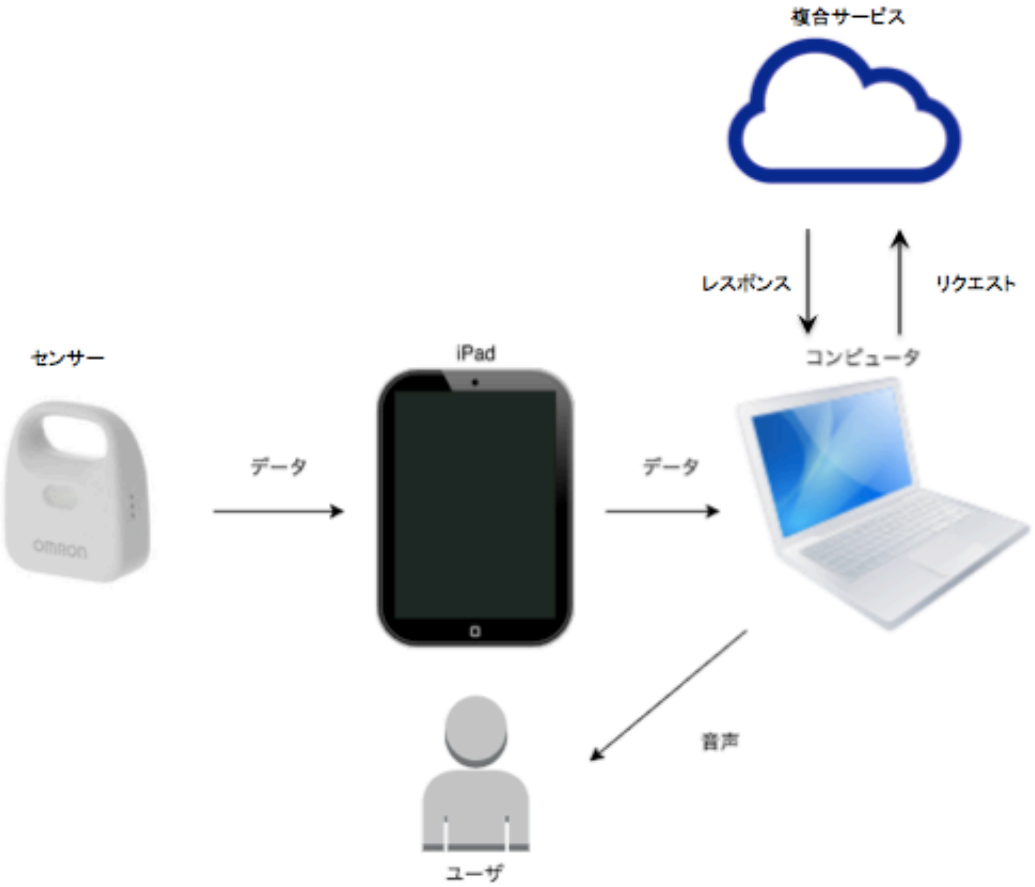


図 5: 実装概要

3. ObservedProperty それぞれに，データタイプを示す PropertyType とデータの値を格納する．
4. tempProperty と humdProperty を Observation オブジェクトに格納する．

- パブリッシャー
  パブリッシャーは生成した Observation オブジェクトをサーバに送信する．

### 5.2.3 メイン部

メイン部では

- プロセッササービスインターフェース
- CEP プロセッサ
- CEP エンジン
- ルール
- スピーカー

の 5 つを実装した．全体として，ユーザデバイスから取得したデータをルールエンジンで処理し，複合サービスへの入力を生成し，送信する役割を果たす．また，音声合成サービスにより生成された音声ファイルを再生する．

- CEP エンジン
  CEP エンジンとして，Java で実装されたイベントエンジンである Drools[1)] を利用する．Drools は Java で実装されたルールエンジンである．中心となるのは推論エンジンであり，事前に構成したルールに基づき，事実に対して条件が真のものを実行する．ルール言語として Drools Rule Language(DRL) を使用する．DRL ファイルは複数のルール，クエリ，型，関数，およびリソース宣言を指定するテキストファイルである．ルールはソースコード 3 のように構成される．ルールは 3.3.2 節で説明した ECA ルールに基づいており，ある条件を満たすセンサーデータが取得された際に，それに応じた処理として選択するサービスとサービスへの入力を生成し，サービスに与えるという処理が行われる．

```
1 rule "名前"
2 when
3     <処理を実行する際のセンサーデータの値>*
4 then
```

[1)] https://www.drools.org/

```
5       <選択するサービスとサービスへの入力をサービスに与える>*
6   end
```

ソースコード 3: drl ファイル仕様

- ルール

  ルールとして”badminton.drl”を実装した．ルールの概要を表 1 に挙げる．本実装は Wet Bulb Globe Temperature(湿球黒球温度,WBGT) を，ユーザへの注意喚起の基準として用い，今回は

$$WBGT = \begin{cases} T + (H - 80)/5 & (80 \leq H) \\ T - (80 - H)/5 & (H < 80) \end{cases} \tag{1}$$

  T:temperature(℃),H:humidity(%)

  と温度と湿度の値から近似して求める．ルールファイルは大きく分けて 5 つのルールから構成されている．

  - WBGCClac ルール

    温度データと湿度データ，利用言語の情報が挿入された際に実行されるルールであり，ユーザデバイスからイベントが挿入された際に最初に実行されるルールである．湿度 80%を基準としてルールは分岐し，それぞれ式 (1) に基づいて WBGT の値を計算する．その後，求めた WBGT を value とし,propertyType として WBGT を持つ ObservedProperty と，入力された言語を value，propertyType として TargetLanguage を持つ ObservedProperty を追加して Observation を生成し，ルールファイルにイベントとして再挿入する．

  - Phase ルール

    Phase ルールはユーザに周囲の温度，湿度によって運動への注意喚起を行うためのルールである．WBGCClac ルールによって再挿入されたイベントにより実行されるルールであり，WBGT の値によって 5 つに細分化される．言語グリッドの辞書翻訳サービスと音声合成サービスにそれぞれ入力を与え，その結果をスピーカーによって出力する．

  - Shuttle ルール

    Shuttle ルールは，バドミントンをするユーザが使用するべき適切な

シャトルコックの種類を温度に応じて提示するルールである．温度の値によって6つに細分化される．言語グリッドの辞書翻訳サービスと音声合成サービスにそれぞれ入力を与え，その結果をスピーカーによって出力する．

- Strings ルール

  Strings ルールはバドミントンをするユーザのラケットのストリングスの張る強さについて助言を与えるためのルールである．温度の値によって3つに細分化される．言語グリッドの辞書翻訳サービスと音声合成サービスにそれぞれ入力を与え，その結果をスピーカーによって出力する．

- floor ルール

  floor ルールは体育館を利用するユーザに対し，湿度による体育館の床への影響について注意喚起を行うルールである．湿度が90%以上の際に実行され，言語グリッドの辞書翻訳サービスと音声合成サービスにそれぞれ入力を与え，その結果をスピーカーによって出力する．

#### 5.2.4 複合サービス部

複合サービスとして，言語グリッドを利用する．言語グリッドは，登録された言語サービスを自由に組み合わせて新しい言語サービスを生み出すサービス連携基盤である．本研究では，言語グリッドに登録されているサービスの中から，最長一致辞書連携翻訳サービス，音声合成サービスを利用する．

- 翻訳部

  翻訳部は，言語グリッドから提供されているサービスの中から，最長一致辞書連携翻訳サービスを利用する．このサービスは，形態素解析サービス，最長一致二ヶ国語辞書サービス，翻訳サービスからなる．これらの複合サービス内から提供されているサービスを自由に組み合わせることができる．本実装では，

  **形態素解析サービス**　MeCab

  **最長一致二ヶ国語辞書サービス**　スポーツ辞書翻訳サービス

  **翻訳サービス**　KyotoUJServer

  を利用する．それぞれ利用するサービスを選択し，翻訳元言語，翻訳先言語，翻訳したい文を入力として与えると，結果として，それらのサービスを利用した翻訳文を返す．

表 1: badminton.drl

| ルール名 | 条件 | 出力 |
|---|---|---|
| WBGCcalc1 | H $\geq$ 80 | WBGT の計算 WBGT = T + (H - 80)/5<br>Propertytype の作成:WBGT,Temperature<br>Humidity,targetlanguage について<br>再構成した Observation の再挿入 |
| WBGCcalc2 | H $<$ 80 | WBGT の計算 WBGT = T - (80 - H)/5<br>Propertytype の作成:WBGT,Temperature<br>Humidity,targetlanguage について<br>再構成した Observation の再挿入 |
| イベントの再挿入後，テキストと利用サービスを<br>Web サービスへの入力としたルールが実行される | | |
| Phase<br>ルール | 辞書翻訳 (スポーツ辞書) サービスと音声合成サービスを<br>利用し，運動中のユーザへの注意喚起を行う | |
| Phase5 | WBGT $\geq$ 31 | 入力: ”運動を中止しましょう.” |
| Phase4 | 28 $\leq$ WBGT $<$ 31 | 入力: ”激しい運動は避け、積極的に休息と水分補給を行いましょう.” |
| Phase3 | 25 $\leq$ WBGT $<$ 28 | 入力: ”激しい運動を行う際は,30 分おきくらいに休息をとりましょう.” |
| Phase2 | 21 $\leq$ WBGT $<$ 25 | 入力: ”水分補給には十分気をつけましょう.” |
| Phase1 | WBGT $<$ 21 | 入力: ”熱中症の危険は少ないですが, 適宜水分補給をしましょう.” |
| Shuttle<br>ルール | 辞書翻訳 (バドミントン辞書) サービスと音声合成サービスを<br>利用し，シャトルコックの選択に関する情報を与える | |
| shuttle1 | T $\geq$ 33 | 入力:”1 番のシャトルを使いましょう.” |
| shuttle2 | 27 $\leq$ T $<$ 33 | 入力:”2 番のシャトルを使いましょう.” |
| shuttle3 | 22 $\leq$ T $<$ 27 | 入力:”3 番のシャトルを使いましょう.” |
| shuttle4 | 17 $\leq$ T $<$ 22 | 入力:”4 番のシャトルを使いましょう.” |
| shuttle5 | 12 $\leq$ T $<$ 17 | 入力:”5 番のシャトルを使いましょう.” |
| shuttle6 | T $<$ 12 | 入力:”6 番のシャトルを使いましょう.” |

| ルール名 | 条件 | 出力 |
|---|---|---|
| Strings ルール | 辞書翻訳 (バドミントン辞書) サービスと音声合成サービスを利用し，ラケットのストリングスに関する情報を与える | |
| strings1 | T > 25 | ”適正温度のときより+1 ポンドのガットが適切です.” |
| strings2 | 15 ≤ T ≤ 25 | ”適正” |
| strings3 | T < 15 | ”適正温度のときより-1 ポンドのガットが適切です.” |
| floor ルール | 辞書翻訳 (スポーツ辞書) サービスと音声合成サービスを利用し，体育館の床面の結露による転倒を警告する | |
| floor | H ≥ 90 | ”湿度が高く床が滑りやすくなっています. 気をつけましょう.” |

- 音声合成部
  音声合成部は，言語グリッドにおいて提供されている音声合成サービス (VoiceText サービス) を利用する．入力として音声を流す言語，音声化するテキスト，オーディオファイルの形式を与えると，入力に応じたオーディオファイルを生成する．作成されたオーディオファイルを流すためのスピーカーモジュールを新たに実装することで，音声合成，出力部として実装した．
  以上より，前 5.2.3 節に基づいて生成されたサービスへの入力に基づいてテキストの翻訳を行い，音声出力を行うという一連の機能が実装される．

## 5.3 考察

本節では，5.2 節で実装したシステムについて考察を与える．
まず，本実装では，センサー資源として (株) オムロンの環境センサーと，無料天気予報 API である OpenWeatherMap[1)] を用いた．ユーザは環境センサーは BLE によって，OpenWeatherMap は JSON によってデータを取得する．両者の仕様は大きく異なるため，データを利用する際に別々のデータ取得モジュールを実装する必要があった．本実装においては，共通のセンサーサービスイン

[1)] http://openweathermap.org/

ターフェースと，それと環境センサー，OpenWeatherMap をつなぐアダプターを実装することで，データ取得部を一通りの実装で実現することができた．また，ユーザの利用言語もこのインターフェースの形に形成し，利用出来ることを示した．このように，新たなセンサー資源が増えた際も，センサーの仕様に合わせてアダプターを実装すれば，それまでの既存のシステムが適用できる．
次に，サービス連携について，本実装では，CEP エンジンを用い，状況に合わせて利用するサービスやサービスに与える入力を自動で生成するという方法を取った．結果として，周囲の温度，湿度に応じて言語グリッドの利用サービスと入力が変化させることができることを確認した．しかし，問題点もいくつか浮上した．一つ目はサービス連携の QoS に関する課題である．サービス連携において，QoS は重要視される点であるが，本実装では経験のあるユーザ（専門家）が事前に決めたルールに基づいてサービス合成を行っているため，システムを利用するユーザそれぞれに対し QoS が最高であるサービスの組み合わせが行なわれているとは限らない．もう一つはセンサーデータの利用法である．本実装では，データを取得する度にルールを実行し，サービス実行を行っている．しかし，実際は，温度や湿度といったデータは短時間で急激に変化することは考えにくく，また，ごく近い距離同士に設置されたセンサーで取得したデータ間に大きな違いがあるとは考えにくい．センサーデータは膨大な量であるために，これらの時間的，あるいは距離的特徴を捉え，よりコストのかからないデータ取得法が必要であると考えられる．

# 第 6 章　終わりに

本研究では，センサーのサービス化手法を提案し，センサーデータに統一化された枠組みを与えた，センサーデータを利用するシステムを実装する際，センサーの種類を考えることなく実装を行うことを可能とした．
次に，複合サービス内の原子サービスの選択問題の解決と，サービスのリアルタイム実行の 2 点を解決するために，センサーデータによるサービス選択手法を提案した．イベント処理システムを応用し，センサーからデータを取得した際に，事前に用意したルールに従って処理を行い，利用するサービスの選択と，サービスへの入力を複合サービスに与える．この手法により，ユーザが複合サービスを利用する際に原子サービスの選択をする必要がなくなり，ユーザの知識

や経験を問わず，最適なサービス選択を行うことができた．また，イベント処理システムを利用することで，センサーデータを受け取ったと同時にサービスを実行することができ，ユーザの手を介さないサービスのリアルタイム実行も実現した．

最後に，これらの提案に基づくシステムを実装した．温度，湿度センサーの存在する環境下で，センサーデータによる複合サービスのサービス選択，実行を実現したものである．

本研究の貢献は以下の 2 点である．

1. センサーのサービス化手法の提案と実装
   センサーデータについて画一化されたデータ定義を与えることでセンサーをサービス化する手法を提案した．これにより，ユーザはセンサーの仕様を気にすることなくデータを利用することができるようになる．また，実際にセンサーサービスインターフェースを実装することで，画一された枠組みとして機能することの実例を示した．
2. リアルタイム性を保証した複合サービスのサービス選択手法の提案と実装
   センサーの値をイベントとして CEP エンジンに挿入し，リアルタイムで処理，アクションとして複合サービスへの入力生成とサービス実行を行うことによって，ユーザがサービスのリクエストを送信することなく，リアルタイムかつ自動的なサービス実行が可能となった．実際にこの手法を用いたシステムを実装することにより実例を示した．

# 謝辞

本研究を行うにあたり, 貴重な資料をご提供いただきました株式会社オムロン様に深く感謝申し上げます. そして本研究を行うにあたり, 熱心なご指導, ご助言を賜りました石田亨教授に厚く御礼申し上げます. また, 日頃より数々のご助言をいただきました中口孝雄特定研究員, 林冬惠助教をはじめ, 石田・松原研究室の皆様方に心より感謝いたします．

# 参考文献

[1] Atzori, L., Iera, A. and Morabito, G.: The Internet of Things: A survey, *Computer Networks*, Vol. 54, No. 15, pp. 2787–2805 (2010).

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

[3] Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., Jubert, I. S., Mazura, M., Harrison, M., Eisenhauer, M., Doody, P., Peter, F., Patrick, G., Sergio, G., Harald, Sundmaeker Alessandro, B., Ignacio Soler, J., Margaretha, M., Mark, H., Markus, E. and Pat, D.: Internet of Things Strategic Research Roadmap, *Internet of Things Strategic Research Roadmap*, pp. 9–52 (2009).

[4] Bouloukakis, G., Georgantas, N., Billet, B., Bouloukakis, G., Georgantas, N. and Revisiting, B. B.: Revisiting Service-oriented Architecture for the IoT : A Middleware Perspective To cite this version : Revisiting Service-oriented Architecture for the IoT : A Middleware Perspective (2016).

[5] Perera, C., Zaslavsky, A., Christen, P. and Georgakopoulos, D.: Context Aware Computing for The Internet of Things, *IEEE Communications Surveys & Tutorials*, Vol. 16, No. 1, pp. 414–454 (2014).

[6] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J. and Chang, H.: QoS-aware middleware for Web services composition, *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, pp. 311–327 (2004).

[7] Alrifai, M. and Risse, T.: Combining global optimization with local selection for efficient QoS-aware service composition, *Proceedings of the 18th international conference on World wide web - WWW '09*, p. 881 (2009).

[8] Alrifai, M., Skoutas, D. and Risse, T.: Selecting skyline services for QoS-based web service composition, *Proceedings of the 19th international conference on World wide web*, Vol. 2588, No. 5, pp. 11–20 (2010).

[9] Lin, D., Shi, C. and Ishida, T.: Dynamic Service Selection Based on Context-Aware QoS, *Proceedings of the 2012 IEEE Ninth International Conference on Services Computing*, pp. 641–648 (2012).

[10] Shi, C., Lin, D. and Ishida, T.: User-centered QoS computation for web service selection, *Proceedings - 2012 IEEE 19th International Conference on Web Services, ICWS 2012*, pp. 456–463 (2012).

[11] Ishida, T.: The Language Grid: Service-Oriented Collective Intelligence

for Language Resource Interoperability. Springer, 2011. ISBN 978-3-642-21177-5. (2011).

[12] Ishida, T., Murakami, Y., Inaba, R. and Lin, D.: The Language Grid : Service-Oriented Multi-Language Infrastructure, No. 1, pp. 2–10 (2012).

[13] Yohei Murakami, Donghui Lin, Toru Ishida, M. T.: Federation Architecture for Multilingual Service Infrastructure, No. 6, pp. 1094–1101 (2014).

[14] Ishida, T., Murakami, Y., Inaba, R. and Lin, D.: The Language Grid : Service-Oriented Multi-Language Infrastructure, No. 1, pp. 2–10 (2012).

[15] Ye, J., Dobson, S. and McKeever, S.: Situation identification techniques in pervasive computing: A review, *Pervasive and Mobile Computing*, Vol. 8, No. 1, pp. 36–66 (2012).

# 付録

実装のソースコードを添付する.

## A.1 デバイスでセンサーデータを取得し、サーバーへ送信するモジュールのソースコード

### A.1.1 MyviewController.java

```
1  package org.langrid.waikiki.sensor;
2
3  import java.net.MalformedURLException;
4  import java.net.URL;
5  import java.util.ArrayList;
6
7  import org.langrid.waikiki.sensor.omron.EnvSensor;
8  import org.langrid.waikiki.sensor.omron.EnvSensorScanner;
9  import org.langrid.waikikiws.service.ObservationReceiverImpl;
10 import org.langrid.waikikiws.service.api.ObservationReceiver;
11 import org.openiot.lsm.beans.Observation;
12 import org.openiot.lsm.beans.ObservedProperty;
13 import org.robovm.apple.coregraphics.CGRect;
14 import org.robovm.apple.foundation.NSBundle;
15 import org.robovm.apple.foundation.NSURL;
16 import org.robovm.apple.uikit.UIColor;
17 import org.robovm.apple.uikit.UIView;
18 import org.robovm.apple.uikit.UIViewController;
19 import org.robovm.apple.webkit.WKScriptMessage;
20 import org.robovm.apple.webkit.WKScriptMessageHandlerAdapter;
21 import org.robovm.apple.webkit.WKUserContentController;
22 import org.robovm.apple.webkit.WKWebView;
23 import org.robovm.apple.webkit.WKWebViewConfiguration;
24
25
26 import jp.go.nict.langrid.client.jsonrpc.JsonRpcClientFactory
       ;
27 import net.arnx.jsonic.JSON;
28
```

```
29 public class MyViewController extends UIViewController {
30   public MyViewController() {
31     // Get the view of this view controller.
32     UIView view = getView();
33
34     // Setup background.
35     view.setBackgroundColor(UIColor.white());
36
37     WKUserContentController controller = new
           WKUserContentController();
38     controller.addScriptMessageHandler(new
           WKScriptMessageHandlerAdapter() {
39       @Override
40       public void didReceiveScriptMessage(
             WKUserContentController c, WKScriptMessage message)
             {
41         System.out.println("message: " + message.getBody());
42         if(message.getName().equals("handler")){
43           if(message.getBody().toString().equals("startScan"
                 ))
44             startScan();
45           if(message.getBody().toString().equals("stopScan"))
46             stopScan();
47         }
48       }
49     }, "handler");
50     WKWebViewConfiguration config = new
           WKWebViewConfiguration();
51     config.setUserContentController(controller);
52     CGRect frame = view.getFrame();
53     wv = new WKWebView(
54         new CGRect(frame.getMinX(), frame.getMinY() + 16,
55             frame.getWidth(), frame.getHeight() - 16),
56         config);
57     view.addSubview(wv);
58     NSURL bu = NSBundle.getMainBundle().getBundleURL();
```

```
59      wv.loadFileURL(new NSURL(bu.toString() + "index.html"),
            bu);
60
61      try {
62        client = new JsonRpcClientFactory().create(
63            ObservationReceiver.class,
64            new URL("http://10.229.248.86:8080/waikikiws/
                  services/ObservationReceiver")
65            );
66      } catch (MalformedURLException e) {
67        e.printStackTrace();
68      }
69    }
70
71    private void startScan(){
72      scanner.startScan(s -> {
73        System.out.println(JSON.encode(s).toString());
74        wv.evaluateJavaScript("found(" + JSON.encode(s) + ");",
              null);
75        // 送信
76        client.notify(createObservation(s)); //s = {"brightness
              ":-112,....}
77      });
78    }
79
80    private void stopScan(){
81      scanner.stopScan();
82    }
83  //s.get~で要素の値を取り出してObservationを生成
84    private Observation createObservation(EnvSensor s){
85      String TEMPERATURE = "http://openiot.eu/ontology/ns/
            AirTemperature";
86      String HUMIDITY = "http://openiot.eu/ontology/ns/
            AtmosphereHumidity";
87
88      Observation o = new Observation();
```

```
89      ArrayList<ObservedProperty> readings = new ArrayList<
            ObservedProperty>();
90      ObservedProperty tempProperty = new ObservedProperty();
91      ObservedProperty humdProperty = new ObservedProperty();
92
93      double temperature = s.getTemperature()/100;
94      double humidity = s.getHumidity()/100;
95
96      tempProperty.setPropertyType(TEMPERATURE);
97      humdProperty.setPropertyType(HUMIDITY);
98      tempProperty.setValue(temperature);
99      humdProperty.setValue(humidity);
100     readings.add(tempProperty);
101     readings.add(humdProperty);
102     o.setReadings(readings);
103     return o;
104   }
105
106   private ObservationReceiver client;
107   private EnvSensorScanner scanner = new EnvSensorScanner();
108   private final WKWebView wv;
109 }
```

### A.1.2 WaikikiSensor.java

```
1  package org.langrid.waikiki.sensor;
2
3  import org.robovm.apple.foundation.NSAutoreleasePool;
4  import org.robovm.apple.uikit.UIApplication;
5  import org.robovm.apple.uikit.UIApplicationDelegateAdapter;
6  import org.robovm.apple.uikit.UIApplicationLaunchOptions;
7  import org.robovm.apple.uikit.UIScreen;
8  import org.robovm.apple.uikit.UIWindow;
9
10 public class WaikikiSensor extends
       UIApplicationDelegateAdapter {
11     private UIWindow window;
12     private MyViewController rootViewController;
```

```
13
14      @Override
15      public boolean didFinishLaunching(UIApplication
            application, UIApplicationLaunchOptions launchOptions)
            {
16          // Set up the view controller.
17          rootViewController = new MyViewController();
18
19          // Create a new window at screen size.
20          window = new UIWindow(UIScreen.getMainScreen().
                getBounds());
21          // Set the view controller as the root controller for
                the window.
22          window.setRootViewController(rootViewController);
23          // Make the window visible.
24          window.makeKeyAndVisible();
25
26          return true;
27      }
28
29      public static void main(String[] args) {
30          try (NSAutoreleasePool pool = new NSAutoreleasePool
                ()) {
31              UIApplication.main(args, null, WaikikiSensor.
                    class);
32          }
33      }
34  }
```

## A.2 受信したデータをルールエンジンに挿入し、状況に応じた出力を得るモジュールのソースコード

### A.2.1 ObservationReceiverImpl.java

```
1  package org.langrid.waikikiws.service;
2
3  import java.io.IOException;
```

```
4  import java.util.ArrayList;
5  import java.util.List;
6  import java.util.Map;
7
8  import org.langrid.waikikiws.DroolsManager;
9  import org.langrid.waikikiws.service.api.ObservationReceiver;
10 import org.langrid.waikikiws.service.api.
       ObservationReceiverDebug;
11 import org.openiot.lsm.beans.Observation;
12 import org.openiot.lsm.beans.ObservedProperty;
13 import org.langrid.waikikiws.service.TargetLanguage;
14
15 public class ObservationReceiverImpl
16 implements ObservationReceiver, ObservationReceiverDebug{
17   @Override
18   public void notify(Observation o){
19     //Observationをルールエンジンへ挿入する
20     DroolsManager.getSession().insert(o);
21     //言語の指定
22     DroolsManager.getSession().insert(new TargetLanguage("en"
           ));
23   }
24
25   public static String TEMPERATURE = "http://openiot.eu/
         ontology/ns/AirTemperature";
26   public static String HUMIDITY = "http://openiot.eu/ontology
         /ns/AtmosphereHumidity";
27
28   //デモ用の関数
29   @Override
30   public void dummyNotify(double temperature, double humidity
         ,String tlanguage) throws IOException {
31
32     Observation o = new Observation();
33     ArrayList<ObservedProperty> readings = new ArrayList<
           ObservedProperty>();
34     ObservedProperty tempProperty = new ObservedProperty();
```

```
35      ObservedProperty humdProperty = new ObservedProperty();
36      tempProperty.setPropertyType(TEMPERATURE);
37      humdProperty.setPropertyType(HUMIDITY);
38      if (tlanguage.equals("api")) {
39        Map<String, Object> m = new WeatherAPI().APIDebug();
40        List<Map<String, Object>> list = (List<Map<String,
              Object>>)m.get("list");
41        Map<String, Object> data = (Map<String, Object>)list.
              get(0).get("main");
42  //      System.out.println(JSON.encode(m, true));
43        double temp = Double.parseDouble(data.get("temp").
              toString()) - 273.15;
44        double humid = Double.parseDouble(data.get("humidity").
              toString());
45        tempProperty.setValue(temp);
46        humdProperty.setValue(humid);
47        readings.add(tempProperty);
48        readings.add(humdProperty);
49        o.setReadings(readings);;
50        DroolsManager.getSession().insert(o);
51        DroolsManager.getSession().insert(new TargetLanguage("
              en"));
52        System.out.println(temp);
53        System.out.println(humid);
54      }else {
55        tempProperty.setValue(temperature);
56        humdProperty.setValue(humidity);
57        readings.add(tempProperty);
58        readings.add(humdProperty);
59        o.setReadings(readings);;
60        DroolsManager.getSession().insert(o);
61        DroolsManager.getSession().insert(new TargetLanguage(
              tlanguage));
62      }
63    }
64  }
```

### A.2.2 Translator.java

```
1  package org.langrid.waikikiws.service;
2
3  import java.io.IOException;
4  import java.io.InputStream;
5  import java.io.InputStreamReader;
6  import java.io.Reader;
7  import java.net.MalformedURLException;
8  import java.net.URL;
9  import java.util.Arrays;
10 import java.util.List;
11 import java.util.Properties;
12
13 import org.langrid.waikikiws.Bindings;
14 import org.langrid.waikikiws.service.api.TranslatorService;
15
16 import jp.go.nict.langrid.client.RequestAttributes;
17 import jp.go.nict.langrid.client.soap.SoapClientFactory;
18 import jp.go.nict.langrid.commons.cs.binding.BindingNode;
19 import jp.go.nict.langrid.service_1_2.
       AccessLimitExceededException;
20 import jp.go.nict.langrid.service_1_2.
       InvalidParameterException;
21 import jp.go.nict.langrid.service_1_2.
       NoAccessPermissionException;
22 import jp.go.nict.langrid.service_1_2.
       NoValidEndpointsException;
23 import jp.go.nict.langrid.service_1_2.ProcessFailedException;
24 import jp.go.nict.langrid.service_1_2.ServerBusyException;
25 import jp.go.nict.langrid.service_1_2.
       ServiceNotActiveException;
26 import jp.go.nict.langrid.service_1_2.
       ServiceNotFoundException;
27 import jp.go.nict.langrid.service_1_2.translation.
       TranslationService;
28
29 public class Translator implements TranslatorService{
```

```
30    public Translator() throws IOException {
31      Properties p = new Properties();
32      try(InputStream is = getClass().getResourceAsStream("/
            langrid.properties");
33          Reader r = new InputStreamReader(is, "UTF-8")){
34        p.load(r);
35      }
36      this.url = p.getProperty("url");
37      this.userId = p.getProperty("userId");
38      this.password = p.getProperty("password");
39    }
40    @Override
41    public String translate(String sourceLang, String
          targetLang, String source) {
42      List<BindingNode> bindings = Arrays.asList(
43          new BindingNode("MorphologicalAnalysisPL", "Mecab"),
44          new BindingNode("TranslationPL", "KyotoUJServer")
45          );
46  //    List<BindingNode> bindings = Bindings.getBindings();
47      try {
48        TranslationService trans = new SoapClientFactory().
              create(
49            TranslationService.class,
50            new URL(url + "
                  TranslationCombinedWithBilingualDictionaryWithLongestMatchSearch
                  "),
51            userId, password
52            );
53        for(BindingNode n : bindings){
54          ((RequestAttributes)trans).getTreeBindings().add(n);
55        }
56        return trans.translate(sourceLang, targetLang, source);
57      } catch (MalformedURLException |
            AccessLimitExceededException |
            InvalidParameterException |
            NoAccessPermissionException | ProcessFailedException |
            NoValidEndpointsException | ServerBusyException |
```

```
        ServiceNotActiveException | ServiceNotFoundException e
        ) {
58        throw new RuntimeException(e);
59      }
60    }
61
62    private String url;
63    private String userId;
64    private String password;
65  }
```

### A.2.3 Binding.java

```
1  package org.langrid.waikikiws;
2
3  import java.util.Arrays;
4  import java.util.List;
5
6  import jp.go.nict.langrid.commons.cs.binding.BindingNode;
7
8  public class Bindings {
9    public static List<BindingNode> getBindings() {
10     return bindings;
11   }
12   public static void binding1(){
13     bindings = Arrays.asList(
14         new BindingNode("MorphologicalAnalysisPL", "Mecab"),
15         new BindingNode("TranslationPL", "KyotoUJServer")
16         );
17     //System.out.println("binding1");
18   }
19   public static void binding2(){
20     bindings = Arrays.asList(
21         new BindingNode("MorphologicalAnalysisPL", "Mecab"),
22         new BindingNode("
               BilingualDictionaryWithLongestMatchSearchPL", "
               KyotoTourismDictionaryDb"),
23         new BindingNode("TranslationPL", "KyotoUJServer")
```

```
24          );
25      //System.out.println("binding2");
26    }
27    public static void setBindings(List<BindingNode> bindings)
          {
28      Bindings.bindings = bindings;
29    }
30    private static List<BindingNode> bindings = Arrays.asList(
31        new BindingNode("MorphologicalAnalysisPL", "Mecab"),
32        new BindingNode("TranslationPL", "KyotoUJServer")
33        );
34  }
```

### A.2.4 DroolsManager.java

```
1   package org.langrid.waikikiws;
2
3   import java.io.IOException;
4
5   import org.kie.api.runtime.KieSession;
6
7   public class DroolsManager {
8     public static synchronized KieSession getSession(){
9       if(session == null){
10        try {
11          session = DroolsUtil.createStreamSessionFromResource(
                "/badminton.drl");
12        } catch (IOException e) {
13          throw new RuntimeException(e);
14        }
15        Thread t = new Thread(() -> {
16          session.fireUntilHalt();
17        });
18        t.setDaemon(true);
19        t.start();
20        org.kie.api.runtime.rule.FactHandle.State.class.getName
              ();
21      }
```

```
22      return session;
23    }
24
25    private static KieSession session;
26  }
```

### A.2.5 DroolsUtil.java

```
1  package org.langrid.waikikiws;
2
3  import java.io.IOException;
4  import java.io.InputStream;
5
6  import org.kie.api.KieBase;
7  import org.kie.api.KieBaseConfiguration;
8  import org.kie.api.KieServices;
9  import org.kie.api.builder.KieBuilder;
10 import org.kie.api.builder.KieFileSystem;
11 import org.kie.api.builder.Message;
12 import org.kie.api.builder.Results;
13 import org.kie.api.conf.EventProcessingOption;
14 import org.kie.api.runtime.KieContainer;
15 import org.kie.api.runtime.KieSession;
16
17 public class DroolsUtil {
18   public static KieSession createSessionFromResource(Package
       pkg, String rulePath) throws IOException{
19     return createSessionFromResource(
20         "/" + pkg.getName().replaceAll("\\.", "/") + "/" +
             rulePath);
21   }
22
23   public static KieSession createSessionFromResource(String
       rulePath)
24   throws IOException{
25     KieServices kieServices = KieServices.Factory.get();
26     KieFileSystem kfs = kieServices.newKieFileSystem();
```

```
27      try(InputStream is = DroolsUtil.class.getResourceAsStream
            (rulePath)){
28        // for each DRL file, referenced by a plain old path
              name:
29        kfs.write("src/main/resources" + rulePath,
30            kieServices.getResources().newInputStreamResource(
                  is));
31        KieBuilder kieBuilder = kieServices.newKieBuilder( kfs
              ).buildAll();
32        Results results = kieBuilder.getResults();
33        if( results.hasMessages( Message.Level.ERROR ) ){
34          System.out.println( results.getMessages() );
35          throw new RuntimeException("### errors ###" );
36        }
37        KieContainer kieContainer = kieServices.newKieContainer
              (
38            kieServices.getRepository().getDefaultReleaseId()
                  );
39        KieBase kieBase = kieContainer.getKieBase();
40        return kieBase.newKieSession();
41      }
42    }
43
44    public static KieSession createStreamSessionFromResource(
          Package pkg, String rulePath) throws IOException{
45      return createStreamSessionFromResource(
46          "/" + pkg.getName().replaceAll("\\.", "/") + "/" +
                rulePath);
47    }
48    public static KieSession createStreamSessionFromResource(
          String rulePath)
49    throws IOException{
50      KieServices kieServices = KieServices.Factory.get();
51      KieFileSystem kfs = kieServices.newKieFileSystem();
52      try(InputStream is = DroolsUtil.class.getResourceAsStream
            (rulePath)){
```

```
53          // for each DRL file, referenced by a plain old path
                name:
54          kfs.write("src/main/resources" + rulePath,
55                  kieServices.getResources().newInputStreamResource(
                        is));
56          KieBuilder kieBuilder = kieServices.newKieBuilder( kfs
                ).buildAll();
57          Results results = kieBuilder.getResults();
58          if( results.hasMessages( Message.Level.ERROR ) ){
59            System.out.println( results.getMessages() );
60            throw new RuntimeException("### errors ###" );
61          }
62          KieContainer kieContainer = kieServices.newKieContainer
                (
63                  kieServices.getRepository().getDefaultReleaseId()
                        );
64          KieBaseConfiguration config = KieServices.Factory.get
                ().newKieBaseConfiguration();
65          config.setOption( EventProcessingOption.STREAM );
66          KieBase kieBase = kieContainer.newKieBase(config);
67          return kieBase.newKieSession();
68        }
69      }
70  }
```

### A.2.6 TargetLanguage

```
1   package org.langrid.waikikiws.service;
2
3
4   public class TargetLanguage {
5     public TargetLanguage(){
6     }
7
8     public TargetLanguage(String targetlang) {
9       super();
10      this.targetlanguage = targetlang;
11    }
```

```
12
13    public String getTargetlang() {
14      return targetlanguage;
15    }
16
17    public void setTargetlang(String targetlang){
18      this.targetlanguage = targetlang;
19    }
20
21    private String targetlanguage;
22
23  }
```

### A.2.7 VoiceText.java

```
1  package org.langrid.waikikiws;
2
3  import java.net.URL;
4  import jp.go.nict.langrid.client.soap.SoapClientFactory;
5  import jp.go.nict.langrid.service_1_2.speech.Speech;
6  import jp.go.nict.langrid.service_1_2.speech.
       TextToSpeechService;
7  import javax.sound.sampled.*;
8
9
10 import java.io.*;
11
12 public class VoiceText {
13     public void voicetext(String text,String lang) throws
           Exception {
14
15     // TODO 自動生成されたメソッド・スタブ
16       TextToSpeechService c =
17           new SoapClientFactory().create(
18                TextToSpeechService.class,
19                new URL("http://langrid.org/service_manager/
                      invoker/kyoto1.langrid:VoiceText"),
20                "ishida.kyoto-u", "tWJaakYm");
```

```
21          Speech s = c.speak(lang, text, "woman","audio/x-wav"
                );
22
23          byte[] buf = s.getAudio();
24          ByteArrayInputStream stream = new
                ByteArrayInputStream(buf);
25        AudioInputStream ais = AudioSystem.getAudioInputStream(
              stream);
26        byte[] data = new byte [ais.available()];
27        ais.read(data);
28        ais.close();
29          AudioFormat af = ais.getFormat();
30          DataLine.Info info = new DataLine.Info(SourceDataLine
                .class, af);
31          SourceDataLine line = (SourceDataLine)AudioSystem.
                getLine(info);
32          line.open();
33          line.start();
34          line.write(buf,0,buf.length);
35          line.drain();
36          line.close();
37      }
38  }
```

### A.2.8 TransTextToSpeech.java

```
1  package org.langrid.waikikiws;
2
3  import org.langrid.waikikiws.VoiceText;
4  import org.langrid.waikikiws.service.Translator;
5
6  public class TransTextToSpeech {
7
8    public void transtexttospeech(String text,int i) throws
         Exception{
9      Translator trans = new Translator();
10     VoiceText tts = new VoiceText();
11     String lang;
```

```
12      if (i == 0) {
13        lang = "en";
14      }
15      else{
16        lang = "zh-CN";
17      }
18      String transtext = trans.translate("ja",lang,text);
19      tts.voicetext(transtext,lang);
20    }
21  }
```

### A.2.9 badminton.drl

```
1  import org.openiot.lsm.beans.Observation;
2  import org.openiot.lsm.beans.ObservedProperty;
3  import org.langrid.waikikiws.TransTextToSpeech;
4  import org.langrid.waikikiws.service.TargetLanguage;
5  import java.util.ArrayList;
6
7  //テキストを音声出力する関数
8  function void TTTS(String text,int t){
9    TransTextToSpeech ttts = new TransTextToSpeech();
10   ttts.transtexttospeech(text,t);
11 }
12
13 //WBGTの計算
14 rule "WBGTcalc1"
15 when
16   $o: Observation()
17   $op1: ObservedProperty(
18     propertyType == "http://openiot.eu/ontology/ns/
         AirTemperature"
19     )
20     from $o.readings
21   $op2: ObservedProperty(
22     value >= 80 &&
23     propertyType == "http://openiot.eu/ontology/ns/
         AtmosphereHumidity"
```

```
24      )
25      from $o.readings
26    $t: TargetLanguage()
27  then
28    double tmp = Double.parseDouble($op1.getValue().toString
          ());
29    double hmd = Double.parseDouble($op2.getValue().toString
          ());
30    double WBGT = tmp + (hmd - 80) / 5;
31    System.out.println("WBGT : " + WBGT + "tmp: " + tmp + "hmd "
          + hmd);
32      Observation o1 = new Observation();
33      Observation o2 = new Observation();
34      Observation o3 = new Observation();
35    ArrayList<ObservedProperty> readings1 = new ArrayList<
          ObservedProperty>();
36    ArrayList<ObservedProperty> readings2 = new ArrayList<
          ObservedProperty>();
37    ArrayList<ObservedProperty> readings3 = new ArrayList<
          ObservedProperty>();
38    ObservedProperty wbgtProperty = new ObservedProperty();
39    ObservedProperty tlangProperty = new ObservedProperty();
40    ObservedProperty tmpProperty = new ObservedProperty();
41    ObservedProperty hmdProperty = new ObservedProperty();
42    wbgtProperty.setPropertyType("http://ishida.kyoto-u/
          watanabe/WetBulbGlobTemperature");
43    tmpProperty.setPropertyType("http://openiot.eu/ontology/ns/
          AirTemperature");
44    hmdProperty.setPropertyType("http://openiot.eu/ontology/ns/
          AtmosphereHumidity");
45    tlangProperty.setPropertyType("http://ishida.kyoto-u/
          watanabe/TargetTransLanguage");
46    wbgtProperty.setValue(WBGT);
47    tmpProperty.setValue(tmp);
48    hmdProperty.setValue(hmd);
49    String st =  $t.getTargetlang();
50    if(st.equals("en")){
```

```
51      tlangProperty.setValue(0);
52    }else if (st.equals("zh-CN")) {
53      tlangProperty.setValue(1);
54    }
55    readings1.add(wbgtProperty);
56    readings1.add(tlangProperty);
57    readings2.add(tmpProperty);
58    readings2.add(tlangProperty);
59    readings3.add(hmdProperty);
60    readings3.add(tlangProperty);
61    o1.setReadings(readings1);
62    o2.setReadings(readings2);
63    o3.setReadings(readings3);
64    insert(o1);
65    insert(o2);
66    insert(o3);
67  end
68
69  rule "WBGTcalc2"
70  when
71    $o: Observation()
72    $op1: ObservedProperty(
73      propertyType == "http://openiot.eu/ontology/ns/
          AirTemperature"
74      )
75      from $o.readings
76    $op2: ObservedProperty(
77      value < 80 &&
78      propertyType == "http://openiot.eu/ontology/ns/
          AtmosphereHumidity"
79      )
80      from $o.readings
81    $t: TargetLanguage()
82  then
83    double tmp = Double.parseDouble($op1.getValue().toString
        ());
```

```
84    double hmd = Double.parseDouble($op2.getValue().toString
          ());
85    double WBGT = tmp - (80 - hmd) / 5;
86    System.out.println("WBGT : " + WBGT + "tmp: " + tmp + "hmd "
          + hmd);
87      Observation o1 = new Observation();
88      Observation o2 = new Observation();
89      Observation o3 = new Observation();
90    ArrayList<ObservedProperty> readings1 = new ArrayList<
          ObservedProperty>();
91    ArrayList<ObservedProperty> readings2 = new ArrayList<
          ObservedProperty>();
92    ArrayList<ObservedProperty> readings3 = new ArrayList<
          ObservedProperty>();
93    ObservedProperty wbgtProperty = new ObservedProperty();
94    ObservedProperty tlangProperty = new ObservedProperty();
95    ObservedProperty tmpProperty = new ObservedProperty();
96    ObservedProperty hmdProperty = new ObservedProperty();
97    wbgtProperty.setPropertyType("http://ishida.kyoto-u/
          watanabe/WetBulbGlobTemperature");
98    tmpProperty.setPropertyType("http://openiot.eu/ontology/ns/
          AirTemperature");
99    hmdProperty.setPropertyType("http://openiot.eu/ontology/ns/
          AtmosphereHumidity");
100   tlangProperty.setPropertyType("http://ishida.kyoto-u/
          watanabe/TargetTransLanguage");
101   wbgtProperty.setValue(WBGT);
102   tmpProperty.setValue(tmp);
103   hmdProperty.setValue(hmd);
104   String st =  $t.getTargetlang();
105   if(st.equals("en")){
106     tlangProperty.setValue(0);
107   }else if (st.equals("zh-CN")) {
108     tlangProperty.setValue(1);
109   }
110   readings1.add(wbgtProperty);
111   readings1.add(tlangProperty);
```

```
112     readings2.add(tmpProperty);
113     readings2.add(tlangProperty);
114     readings3.add(hmdProperty);
115     readings3.add(tlangProperty);
116     o1.setReadings(readings1);
117     o2.setReadings(readings2);
118     o3.setReadings(readings3);
119     insert(o1);
120     insert(o2);
121     insert(o3);
122 end
123
124 //WBGTによって運動時の注意喚起を行うようにする
125 rule "phase5"   //WBGT>=31
126 when
127     $o: Observation()
128     $op1: ObservedProperty(
129       value >= 31 &&
130       propertyType == "http://ishida.kyoto-u/watanabe/
              WetBulbGlobTemperature"
131       )
132       from $o.readings
133     $op2: ObservedProperty(
134       propertyType == "http://ishida.kyoto-u/watanabe/
              TargetTransLanguage"
135       )
136       from $o.readings
137 then
138     System.out.println("運動を中止しましょう.");
139     TTTS("運動を中止しましょう
          .",Integer.parseInt($op2.getValue().toString()));
140 end
141
142 rule "phase4"   //28<=WBGT<31
143 when
144     $o: Observation()
145     $op1: ObservedProperty(
```

```
146       value < 31 && value >= 28 &&
147       propertyType == "http://ishida.kyoto-u/watanabe/
              WetBulbGlobTemperature"
148       )
149       from $o.readings
150     $op2: ObservedProperty(
151       propertyType == "http://ishida.kyoto-u/watanabe/
              TargetTransLanguage"
152       )
153       from $o.readings
154   then
155     System.out.println("激しい運動は避け、積極的に休息と水分補給を行いましょう.");
156     TTTS("激しい運動は避け、積極的に休息と水分補給を行いましょう.",Integer.parseInt($op2.getValue().toString()));
157   end
158
159   rule "phase3"   //25<=WBGT<28
160   when
161     $o: Observation()
162     $op1: ObservedProperty(
163       value < 28 && value >= 25 &&
164       propertyType == "http://ishida.kyoto-u/watanabe/
              WetBulbGlobTemperature"
165       )
166       from $o.readings
167     $op2: ObservedProperty(
168       propertyType == "http://ishida.kyoto-u/watanabe/
              TargetTransLanguage"
169       )
170       from $o.readings
171   then
172     System.out.println("激しい運動を行う際は,30分おきくらいに休息をとりましょう.");
173     TTTS("激しい運動を行う際は,30分おきくらいに休息をとりましょう.",Integer.parseInt($op2.getValue().toString()));
174   end
175
```

```
176 rule "phase2"   //21<=WBGT<25
177 when
178   $o: Observation()
179   $op1: ObservedProperty(
180     value < 25 && value >= 21 &&
181     propertyType == "http://ishida.kyoto-u/watanabe/
          WetBulbGlobTemperature"
182     )
183     from $o.readings
184   $op2: ObservedProperty(
185     propertyType == "http://ishida.kyoto-u/watanabe/
          TargetTransLanguage"
186     )
187     from $o.readings
188 then
189   System.out.println("水分補給には十分気をつけましょう.");
190   TTTS("水分補給には十分気をつけましょう
        .",Integer.parseInt($op2.getValue().toString()));
191 end
192
193 rule "phase1"    //WBGT<21
194 when
195   $o: Observation()
196   $op1: ObservedProperty(
197     value < 21 &&
198     propertyType == "http://ishida.kyoto-u/watanabe/
          WetBulbGlobTemperature"
199     )
200     from $o.readings
201   $op2: ObservedProperty(
202     propertyType == "http://ishida.kyoto-u/watanabe/
          TargetTransLanguage"
203     )
204     from $o.readings
205 then
206   System.out.println("熱中症の危険は少ないですが
        ,適宜水分補給をしましょう.");
```

```
207     TTTS("熱中症の危険は少ないですが,適宜水分補給をしましょう.",
            Integer.parseInt($op2.getValue().toString()));
208 end
209
210 //湿度が高いと床が滑りやすくなる注意
211 rule "floor"
212 when
213     $o: Observation()
214     $h: ObservedProperty(
215         value >= 90 &&
216         propertyType == "http://openiot.eu/ontology/ns/
                AtmosphereHumidity"
217         )
218         from $o.readings
219     $op2: ObservedProperty(
220         propertyType == "http://ishida.kyoto-u/watanabe/
                TargetTransLanguage"
221         )
222         from $o.readings
223 then
224     System.out.println("湿度が高く床が滑りやすくなっています
            .気をつけましょう.");
225     TTTS("湿度が高く床が滑りやすくなっています.気をつけましょう.",
            Integer.parseInt($op2.getValue().toString()));
226 end
227
228 //温度によって適切なシャトルの番号を提示する
229 rule "shuttle1"    //1番シャトル
230 when
231     $o: Observation()
232     $op1: ObservedProperty(
233         value >= 33 &&
234         propertyType == "http://openiot.eu/ontology/ns/
                AirTemperature"
235         )
236         from $o.readings
237     $op2: ObservedProperty(
```

```
238       propertyType == "http://ishida.kyoto-u/watanabe/
              TargetTransLanguage"
239       )
240       from $o.readings
241   then
242     System.out.println("1番のシャトルを使いましょう.");
243     TTTS("1番のシャトルを使いましょう.",Integer.parseInt($op2.
            getValue().toString()));
244   end
245
246   rule "shuttle2"    //2番シャトル
247   when
248     $o: Observation()
249     $op1: ObservedProperty(
250       value < 33 && value >= 27 &&
251       propertyType == "http://openiot.eu/ontology/ns/
              AirTemperature"
252       )
253       from $o.readings
254     $op2: ObservedProperty(
255       propertyType == "http://ishida.kyoto-u/watanabe/
              TargetTransLanguage"
256       )
257       from $o.readings
258   then
259     System.out.println("2番のシャトルを使いましょう.");
260     TTTS("2番のシャトルを使いましょう.",Integer.parseInt($op2.
            getValue().toString()));
261   end
262
263   rule "shuttle3"    //3番シャトル
264   when
265     $o: Observation()
266     $op1: ObservedProperty(
267       value < 27 && value >= 22 &&
268       propertyType == "http://openiot.eu/ontology/ns/
              AirTemperature"
```

```
269       )
270       from $o.readings
271     $op2: ObservedProperty(
272       propertyType == "http://ishida.kyoto-u/watanabe/
              TargetTransLanguage"
273       )
274       from $o.readings
275   then
276     System.out.println("3番のシャトルを使いましょう.");
277     TTTS("3番のシャトルを使いましょう.",Integer.parseInt($op2.
          getValue().toString()));
278   end
279
280   rule "shuttle4"    //4番シャトル
281   when
282     $o: Observation()
283     $op1: ObservedProperty(
284       value < 22 && value >= 17 &&
285       propertyType == "http://openiot.eu/ontology/ns/
              AirTemperature"
286       )
287       from $o.readings
288     $op2: ObservedProperty(
289       propertyType == "http://ishida.kyoto-u/watanabe/
              TargetTransLanguage"
290       )
291       from $o.readings
292   then
293     System.out.println("4番のシャトルを使いましょう.");
294     TTTS("4番のシャトルを使いましょう.",Integer.parseInt($op2.
          getValue().toString()));
295   end
296
297   rule "shuttle5"    //5番シャトル
298   when
299     $o: Observation()
300     $op1: ObservedProperty(
```

```
301       value < 17 && value >= 12 &&
302       propertyType == "http://openiot.eu/ontology/ns/
              AirTemperature"
303       )
304       from $o.readings
305     $op2: ObservedProperty(
306       propertyType == "http://ishida.kyoto-u/watanabe/
              TargetTransLanguage"
307       )
308       from $o.readings
309   then
310     System.out.println("5番のシャトルを使いましょう.");
311     TTTS("5番のシャトルを使いましょう.",Integer.parseInt($op2.
          getValue().toString()));
312   end
313
314   rule "shuttle6"    //6番シャトル
315   when
316     $o: Observation()
317     $op1: ObservedProperty(
318       value < 12 &&
319       propertyType == "http://openiot.eu/ontology/ns/
              AirTemperature"
320       )
321       from $o.readings
322     $op2: ObservedProperty(
323       propertyType == "http://ishida.kyoto-u/watanabe/
              TargetTransLanguage"
324       )
325       from $o.readings
326   then
327     System.out.println("6番のシャトルを使いましょう.");
328     TTTS("6番のシャトルを使いましょう.",Integer.parseInt($op2.
          getValue().toString()));
329   end
330
331   //温度によってラケットに張るストリングのテンションの助言をする
```

```
332 rule "strings1"    //温度が高いときのストリング
333 when
334   $o: Observation()
335   $op1: ObservedProperty(
336     value > 25 &&
337     propertyType == "http://openiot.eu/ontology/ns/
          AirTemperature"
338     )
339     from $o.readings
340   $op2: ObservedProperty(
341     propertyType == "http://ishida.kyoto-u/watanabe/
          TargetTransLanguage"
342     )
343     from $o.readings
344 then
345   System.out.println("適正温度のときより
        +1ポンドのガットが適切です.");
346   TTTS("適正温度のときより+1ポンドのガットが適切です.",Integer.
        parseInt($op2.getValue().toString()));
347 end
348
349 rule "strings2"    //適正温度のときのストリング
350 when
351   $o: Observation()
352   $op1: ObservedProperty(
353     value <= 25 && value >= 15 &&
354     propertyType == "http://openiot.eu/ontology/ns/
          AirTemperature"
355     )
356     from $o.readings
357   $op2: ObservedProperty(
358     propertyType == "http://ishida.kyoto-u/watanabe/
          TargetTransLanguage"
359     )
360     from $o.readings
361 then
362   System.out.println("適正");
```

```
363     TTTS("ガットの適正温度です
            .",Integer.parseInt($op2.getValue().toString()));
364 end
365
366 rule "strings3"     //温度が低いときのストリング
367 when
368     $o: Observation()
369     $op1: ObservedProperty(
370         value < 15 &&
371         propertyType == "http://openiot.eu/ontology/ns/
                AirTemperature"
372         )
373         from $o.readings
374     $op2: ObservedProperty(
375         propertyType == "http://ishida.kyoto-u/watanabe/
                TargetTransLanguage"
376         )
377         from $o.readings
378 then
379     System.out.println("適正温度のときより
            -1ポンドのガットが適切です.");
380     TTTS("適正温度のときより-1ポンドのガットが適切です.",Integer.
            parseInt($op2.getValue().toString()));
381 end
```

## A.3 オムロンのセンサー定義

### A.3.1 EnvSensor.java

```
1  package org.langrid.waikiki.sensor.omron;
2
3  public class EnvSensor {
4    public EnvSensor() {
5    }
6    //EnvSensor 11個の変数
7    public EnvSensor(
8        String uuid,
```

```
9         int lineNo, int temperature, int humidity, int
              brightness,
10        int uvIndex, int pressure, int noise,
11        int discomfortIndex, int heatstrokeIndex, int
              cellVoltage) {
12      this.uuid = uuid;
13      this.lineNo = lineNo;
14      this.temperature = temperature;
15      this.humidity = humidity;
16      this.brightness = brightness;
17      this.uvIndex = uvIndex;
18      this.pressure = pressure;
19      this.noise = noise;
20      this.discomfortIndex = discomfortIndex;
21      this.heatstrokeIndex = heatstrokeIndex;
22      this.cellVoltage = cellVoltage;
23    }
24  //行番号が1,その他が2バイトずつの19バイト
25    public static EnvSensor create(String uuid, byte[] bytes){
26      if(bytes.length != 19) throw new RuntimeException("The␣
            length␣of␣bytes␣must␣be␣19");
27      return new EnvSensor(
28          uuid,
29          (int)bytes[0],
30          bytes[1] + (bytes[2] << 8),
31          bytes[3] + (bytes[4] << 8),
32          bytes[5] + (bytes[6] << 8),
33          bytes[7] + (bytes[8] << 8),
34          bytes[9] + (bytes[10] << 8),
35          bytes[11] + (bytes[12] << 8),
36          bytes[13] + (bytes[14] << 8),
37          bytes[15] + (bytes[16] << 8),
38          bytes[17] + (bytes[18] << 8));
39    }
40
41    public String getUuid() {
42      return uuid;
```

```
43    }
44    public void setUuid(String uuid) {
45      this.uuid = uuid;
46    }
47    public int getLineNo() {
48      return lineNo;
49    }
50    public void setLineNo(int lineNo) {
51      this.lineNo = lineNo;
52    }
53    public int getTemperature() {
54      return temperature;
55    }
56    public void setTemperature(int temperature) {
57      this.temperature = temperature;
58    }
59    public int getHumidity() {
60      return humidity;
61    }
62    public void setHumidity(int humidity) {
63      this.humidity = humidity;
64    }
65    public int getBrightness() {
66      return brightness;
67    }
68    public void setBrightness(int brightness) {
69      this.brightness = brightness;
70    }
71    public int getUvIndex() {
72      return uvIndex;
73    }
74    public void setUvIndex(int uvIndex) {
75      this.uvIndex = uvIndex;
76    }
77    public int getPressure() {
78      return pressure;
79    }
```

```
80    public void setPressure(int pressure) {
81      this.pressure = pressure;
82    }
83    public int getNoise() {
84      return noise;
85    }
86    public void setNoise(int noise) {
87      this.noise = noise;
88    }
89    public int getDiscomfortIndex() {
90      return discomfortIndex;
91    }
92    public void setDiscomfortIndex(int discomfortIndex) {
93      this.discomfortIndex = discomfortIndex;
94    }
95    public int getHeatstrokeIndex() {
96      return heatstrokeIndex;
97    }
98    public void setHeatstrokeIndex(int heatstrokeIndex) {
99      this.heatstrokeIndex = heatstrokeIndex;
100   }
101   public int getCellVoltage() {
102     return cellVoltage;
103   }
104   public void setCellVoltage(int cellVoltage) {
105     this.cellVoltage = cellVoltage;
106   }
107
108   private String uuid;
109   private int lineNo; // ("行番号: " + bytes[0]);
110   private int temperature; // ("温度: " + (bytes[1] + (bytes
        [2] << 8)));
111   private int humidity; // ("相対湿度: " + (bytes[3] + (bytes
        [4] << 8)));
112   private int brightness; // ("照度: " + (bytes[5] + (bytes
        [6] << 8)));
```

```
113     private int uvIndex; // ("UVI: " + (bytes[7] + (bytes[8] <<
            8)));
114     private int pressure; // ("気圧: " + (bytes[9] + (bytes[10]
            << 8)));
115     private int noise; // ("騒音: " + (bytes[11] + (bytes[12]
           << 8)));
116     private int discomfortIndex; // ("不快指数: " + (bytes[13]
            + (bytes[14] << 8)));
117     private int heatstrokeIndex; // ("熱中症危険度: " + (bytes
            [15] + (bytes[16] << 8)));
118     private int cellVoltage; // ("電池電圧: " + (bytes[17] + (
            bytes[18] << 8)));
119 }
```

### A.3.2 EnvSensorListener.java

```
1  package org.langrid.waikiki.sensor.omron;
2
3  public interface EnvSensorListener {
4    void onFound(EnvSensor sensor);
5  }
```

### A.3.3 EnvSensorScanner.java

```
1  package org.langrid.waikiki.sensor.omron;
2
3  import java.util.LinkedHashSet;
4  import java.util.Set;
5
6  import org.robovm.apple.corebluetooth.CBAdvertisementData;
7  import org.robovm.apple.corebluetooth.CBCentralManager;
8  import org.robovm.apple.corebluetooth.
       CBCentralManagerDelegateAdapter;
9  import org.robovm.apple.corebluetooth.CBCharacteristic;
10 import org.robovm.apple.corebluetooth.CBPeripheral;
11 import org.robovm.apple.corebluetooth.
       CBPeripheralDelegateAdapter;
12 import org.robovm.apple.corebluetooth.CBService;
13 import org.robovm.apple.foundation.NSData;
```

```
14 import org.robovm.apple.foundation.NSError;
15 import org.robovm.apple.foundation.NSNumber;
16
17 import jp.go.nict.langrid.client.jsonrpc.JsonRpcClientFactory
       ;
18 import jp.go.nict.langrid.repackaged.net.arnx.jsonic.JSON;
19
20 public class EnvSensorScanner {
21   private Set<CBPeripheral> peripherals;
22   private CBCentralManager central;
23   public void startScan(EnvSensorListener listener){
24     this.peripherals = new LinkedHashSet<>();
25     this.central = new CBCentralManager(
26         new CBCentralManagerDelegateAdapter(){
27           @Override
28           public void didUpdateState(CBCentralManager central
                ) {
29             switch(central.getState().toString()){
30               case "PoweredOn":
31                 central.scanForPeripherals(null, null);
32                 break;
33             }
34             super.didUpdateState(central);
35           }
36           @Override
37           public void didDiscoverPeripheral(CBCentralManager
                central, CBPeripheral peripheral,
                CBAdvertisementData advertisementData, NSNumber
                rssi) {
38             if(peripherals.contains(peripheral)) return;
39             peripherals.add(peripheral);
40             System.out.println(peripheral);
41             if("EnvSensor-BL01".equals(peripheral.getName
                   ())){
42               central.connectPeripheral(peripheral, null);
43             }
44           }
```

```
45          @Override
46          public void didConnectPeripheral(CBCentralManager
                central, CBPeripheral peripheral) {
47            peripheral.setDelegate(new
                 CBPeripheralDelegateAdapter(){
48              @Override
49              public void didDiscoverServices(CBPeripheral
                    peripheral, NSError error) {
50                for(CBService s : peripheral.getServices()){
51                  if(s.getUUID().toString().equals("0C4C3000
                        -7700-46F4-AA96-D5E974E32A54")){
52                    peripheral.discoverCharacteristics(null,
                          s);
53                  }
54                }
55              }
56              @Override
57              public void didDiscoverCharacteristics(
                    CBPeripheral peripheral, CBService service,
58                    NSError error) {
59                for(CBCharacteristic c : service.
                      getCharacteristics()){
60                  if(c.getUUID().toString().equals("0C4C3001
                        -7700-46F4-AA96-D5E974E32A54")){
61                    peripheral.readValue(c);
62                  }
63                }
64              }
65              @Override
66              public void didUpdateValue(CBPeripheral
                    peripheral, CBCharacteristic characteristic,
67                    NSError error) {
68                NSData data = characteristic.getValue();
69                listener.onFound(EnvSensor.create(peripheral.
                      getIdentifier().toString(), data.getBytes
                      ()));
70              }
```

```
71                 });
72                 peripheral.discoverServices(null);
73             }
74             @Override
75             public void didFailToConnectPeripheral(
                   CBCentralManager central, CBPeripheral
                   peripheral,
76                  NSError error) {
77                 System.out.println("failed to connected to" +
                       peripheral);
78             }
79         }, null
80         );
81     }
82     public void stopScan(){
83       central.stopScan();
84     }
85 }
```

## A.4 OpenIoTのデータ定義

### A.4.1 Observation.java

```
1  package org.openiot.lsm.beans;
2  /**
3  *     Copyright (c) 2011-2014, OpenIoT
4  *
5  *     This file is part of OpenIoT.
6  *
7  *     OpenIoT is free software: you can redistribute it and/or
       modify
8  *     it under the terms of the GNU Lesser General Public
       License as published by
9  *     the Free Software Foundation, version 3 of the License.
10 *
11 *     OpenIoT is distributed in the hope that it will be
       useful,
```

```
12 *     but WITHOUT ANY WARRANTY; without even the implied
       warranty of
13 *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
       See the
14 *     GNU Lesser General Public License for more details.
15 *
16 *     You should have received a copy of the GNU Lesser
       General Public License
17 *     along with OpenIoT.  If not, see <http://www.gnu.org/
       licenses/>.
18 *
19 *      Contact: OpenIoT mailto: info@openiot.eu
20 */
21 import java.util.ArrayList;
22 import java.util.Date;
23
24 /**
25  *
26  * @author Hoan Nguyen Mau Quoc
27  *
28  */
29 public class Observation implements java.io.Serializable {
30   private String id;
31   private Date times;
32   private String sensorId;
33   private String featureOfInterest="";
34   private ArrayList<ObservedProperty> readings;
35   private String metaGraph;
36   private String dataGraph;
37
38   public Observation(){
39     id = ""+System.nanoTime();
40     readings = new ArrayList<ObservedProperty>();
41   }
42
43   public String getId() {
44     return id;
```

```
45    }
46    public void setId(String id) {
47      this.id = id;
48    }
49    public Date getTimes() {
50      return times;
51    }
52    public void setTimes(Date times) {
53      this.times = times;
54    }
55    public String getSensor() {
56      return sensorId;
57    }
58    public void setSensor(String sensorId) {
59      this.sensorId = sensorId;
60    }
61    public String getFeatureOfInterest() {
62      return featureOfInterest;
63    }
64    public void setFeatureOfInterest(String featureOfInterest)
        {
65      this.featureOfInterest = featureOfInterest;
66    }
67    public ArrayList<ObservedProperty> getReadings() {
68      return readings;
69    }
70    public void setReadings(ArrayList<ObservedProperty>
        readings) {
71      this.readings = readings;
72    }
73
74    public void addReading(ObservedProperty reading){
75      readings.add(reading);
76    }
77
78    public void removeReading(ObservedProperty reading){
79      readings.remove(reading);
```

```
80    }
81
82    public String getMetaGraph() {
83      return metaGraph;
84    }
85
86    public void setMetaGraph(String metaGraph) {
87      this.metaGraph = metaGraph;
88    }
89
90    public String getDataGraph() {
91      return dataGraph;
92    }
93
94    public void setDataGraph(String dataGraph) {
95      this.dataGraph = dataGraph;
96    }
97
98  }
```

### A.4.2 ObserbationProperty.java

```
1  package org.openiot.lsm.beans;
2  /**
3  *     Copyright (c) 2011-2014, OpenIoT
4  *
5  *     This file is part of OpenIoT.
6  *
7  *     OpenIoT is free software: you can redistribute it and/or
       modify
8  *     it under the terms of the GNU Lesser General Public
       License as published by
9  *     the Free Software Foundation, version 3 of the License.
10 *
11 *     OpenIoT is distributed in the hope that it will be
       useful,
12 *     but WITHOUT ANY WARRANTY; without even the implied
       warranty of
```

```
13 *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
       See the
14 *     GNU Lesser General Public License for more details.
15 *
16 *     You should have received a copy of the GNU Lesser
       General Public License
17 *     along with OpenIoT.  If not, see <http://www.gnu.org/
       licenses/>.
18 *
19 *      Contact: OpenIoT mailto: info@openiot.eu
20 */
21 import java.util.Date;
22 /**
23  *
24  * @author Hoan Nguyen Mau Quoc
25  *
26  */
27 public class ObservedProperty implements java.io.Serializable
       {
28
29   private static final long serialVersionUID = 1L;
30   private Object value;
31   private Date times;
32   private String propertyType;
33   private String unit;
34   private String observationId;
35
36   public Object getValue() {
37     return value;
38   }
39   public void setValue(Object value) {
40     this.value = value;
41   }
42   public void setValue(Double value) {
43     this.value = Double.toString(value);
44   }
45   public void setValue(int value) {
```

```
46      this.value = Integer.toString(value);
47    }
48    public String getObservationId() {
49      return observationId;
50    }
51    public void setObservationId(String observationId) {
52      this.observationId = observationId;
53    }
54
55    public Date getTimes() {
56      return times;
57    }
58    public void setTimes(Date times) {
59      this.times = times;
60    }
61
62
63    public String getUnit() {
64      return unit;
65    }
66    public void setUnit(String unit) {
67      this.unit = unit;
68    }
69    public String getPropertyType() {
70      return propertyType;
71    }
72    public void setPropertyType(String obsClassURL) {
73      this.propertyType = obsClassURL;
74    }
75
76  }
```