

# 修士論文

題目

IoT機器からの通知に基づいた機器監視サービスの開発

学籍番号・氏名

15006・宮坂 虹櫻

指導教員

横山 輝明

提出日

2017年2月13日

神戸情報大学院大学  
情報技術研究科 情報システム専攻

# 目 次

|  |    |
|--|----|
| 第 1 章 はじめに                               | 1  |
| 1.1 研究の背景                                | 1  |
| 1.2 IoT サービスの維持における課題                    | 1  |
| 1.3 研究の目的                                | 2  |
| 1.4 本論文の構成                               | 2  |
| 第 2 章 IoT サービスの提供における問題                  | 3  |
| 2.1 IoT サービスの構造と事例                       | 3  |
| 2.2 IoT サービス開発の事例                        | 4  |
| 2.2.1 株式会社ルナネクサスでの事例                     | 4  |
| 2.2.2 岡本商店街における事例                        | 5  |
| 2.3 IoT サービスの提供者の課題                      | 6  |
| 第 3 章 IoT 機器からの通知に基づく機器監視サービスの提案         | 7  |
| 3.1 既存手法の問題点                             | 7  |
| 3.2 IoT 機器監視・管理サービスの提案                   | 8  |
| 第 4 章 機器監視サービスの実装                        | 9  |
| 4.1 要件と機能の抽出                             | 9  |
| 4.2 提案サービスの構成                            | 9  |
| 4.3 エージェントプログラムの設計と実装                    | 10 |
| 4.4 エージェントプログラム用インターフェースの設計              | 10 |
| 4.5 エージェントプログラムとエージェントプログラム用インターフェース間の通信 | 14 |
| 4.6 機器状態データベースの設計                        | 16 |
| 4.7 機器情報データベースの設計                        | 16 |
| 4.8 可視化アプリケーションの設計                       | 17 |
| 4.9 サービス利用のイメージ                          | 17 |
| 4.9.1 機器の追加                              | 17 |

|                             |           |
|-----------------------------|-----------|
| 4.9.2 機器の削除                 | 18        |
| 4.10 実装                     | 18        |
| <b>第 5 章 機器監視サービスの動作テスト</b> | <b>19</b> |
| 5.1 検証目的                    | 19        |
| 5.2 検証方法                    | 19        |
| 5.3 検証結果                    | 19        |
| 5.4 考察                      | 20        |
| <b>第 6 章 考察</b>             | <b>22</b> |
| <b>第 7 章 おわりに</b>           | <b>24</b> |
| <b>参考文献</b>                 | <b>26</b> |
| .1 岡本商店街での事例                | 26        |

# 図 目 次

|  |    |
|--|----|
| 2.1 IoT サービスの構造図                                   | 3  |
| 2.2 株式会社ルナネクサス サービスイメージ図                           | 4  |
| 2.3 岡本商店街人流観測 構成図                                  | 5  |
| 3.1 サービス構成図  | 8  |
| 4.1 サービス構成図  | 10 |
| 4.2 エージェントプログラムの動作                                 | 11 |
| 4.3 エージェントプログラム用インターフェースの動作 (機器から登録用トークンが送られてきた場合) | 12 |
| 4.4 エージェントプログラム用インターフェースの動作 (機器からの状態の通知があった際)      | 13 |
| 4.5 機器 ID の取得/発行・登録時の通信                            | 15 |
| 4.6 機器状態の通知/監視時の通信                                 | 15 |
| 4.7 ユーザテーブルと機器情報テーブルの関連                            | 17 |
| 5.1 IoT サービスの構成図                                   | 20 |
| 1 岡本商店街人流観測 構成図                                    | 28 |
| 2 岡本商店街人流観測 使用した機器                                 | 28 |
| 3 岡本商店街人流観測可視化アプリケーションスクリーンショット                    | 29 |

# 表 目 次

|                        |    |
|------------------------|----|
| 4.1 機器状態管理テーブル         | 16 |
| 4.2 ユーザテーブル (Users)    | 17 |
| 4.3 機器情報テーブル (Devices) | 17 |

## 内容梗概

近年，IoT が注目を集めている。IoT とは，コンピュータをさまざまなモノに取り付けることで，利便性の向上を図る概念である。近年の半導体技術の進歩により，コンピュータが安価・小型になったこと，インターネットへの通信が様々な場所で安価に行えるようになったことにより，安価に実現可能となった。

それらのモノが連携して提供するサービスは IoT サービスと呼ばれ，より生活に身近なサービスの登場が期待されている。IoT サービスは，IoT 機器とサーバーがインターネットを介して通信し合うことで，成り立っている。IoT 機器は，モノにコンピュータが取り付けられた機器で，周囲の状況を検知，または，周囲へ働きかける機能を持つ。サーバーは，IoT 機器からの情報を蓄積・分析し，IoT 機器へ指示を送るか，ユーザーへ分析結果を表示する機能を持つ。これら IoT 機器とサーバーが連携することで，IoT サービスは利便性をユーザーへ提供している。

IoT サービスを円滑に提供するには，IoT 機器とサーバーの連携を正常に維持しなければならない。そのため，IoT 機器の動作状態や通信状態の監視が重要となる。IoT 機器の監視は，接続されるネットワークが多様であることから，通知型の監視を用いる必要がある。そのため，多量に存在する IoT 機器へ個別に設定をする必要がある。また，多量の IoT 機器を監視サーバへ登録する必要もある。これら設定は，IoT 機器の交換や追加の際にも発生するため，IoT サービスの開発・運用の負担となっている。

本研究では，これら負担が，IoT 機器と監視サーバに分散して設定が存在することが原因だと考えた。そこで，分散して存在する設定をサーバにて一元的に管理し，IoT 機器への個別の設定，監視サービスへの登録作業を省力化することを提案する。サーバから IoT 機器へ個別の ID を付与し，機器への設定を簡略化し，ID を付与した時点で機器を登録することで，登録を自動化する。機器は，個別の ID に紐付いた URL へ https にて通知することで，機器の監視を可能にした。

# 第1章 はじめに

## 1.1 研究の背景

近年，IoT が注目を集めている。IoT とは，様々なモノがインターネットにつながり，相互に情報をやり取りすることで，利便性の向上を図る概念である。近年の半導体技術の進歩により，コンピュータが安価・小型になったこと，インターネットへの通信が様々な場所で安価に行えるようになったことにより，注目が集まっている。IoT の具体的な例としては，建設重機の盗難防止，プリンタのトナー発注自動化，体温や脈拍等の収集・可視化等が挙げられる。

IoT サービスとは，IoT による利便性を顧客に提供するサービスの事で，より生活に身近なサービスの登場が期待されている。上記の例では，プリンタのトナー発注自動化が IoT サービスにあたる。IoT サービスは，IoT 機器とサーバがインターネットを介して通信し合うことで成り立っている。IoT 機器は，モノにコンピュータが取り付けられた物で，周囲の状況を検知，または，周囲へ働きかける機能を持つ。サーバーは，IoT 機器からの情報を蓄積・分析し，IoT 機器へ指示を送るか，ユーザーへ分析結果を表示する機能を持つ。これら IoT 機器とサーバーが連携することで，IoT サービスは利便性をユーザーへ提供している。

このような IoT サービスを円滑に提供するためには，IoT 機器・インターネットへの接続・サーバの状態を監視し，必要に応じて，機器の交換等を行う必要がある。

## 1.2 IoT サービスの維持における課題

しかし，IoT 機器の監視は，技術的課題から既存手法の適応が難しい。既存手法は，機器の監視のためには，監視サーバと機器の両方に設定が必要となる。IoT 機器が多量に使用されること，IoT 機器の追加・撤去・交換は頻繁にあることから，監視サーバへの登録や IoT 機器への個別の設定が，大きな負担となっている。

IoT サービスの提供者として，太陽光発電発電に係る機器の監視サービスを提供している株式会社ルナネクサスが挙げられる。株式会社ルナネクサスとは，大阪にある組み込み機器メーカーである。株式会社ルナネクサスでは，太陽光発電事業を展開している事業主に対し，発電に係る機器の状態や，発電量等を可視化できるサービスを展開している。しかし，IoT 機器の数に増減があることや，故障などによる交換により次のような事が問題となっている。

- 各 IoT 機器の状態を監視するために、多数の IoT 機器へ個別の設定をしなければならないこと
- 増減や交換の度に、機器監視システムへの登録をしなければならないこと
- 交換の為に現地に行くも、類似の機器が多数存在し、外観から交換対象機器がわかりづらいこと

### 1.3 研究の目的

そこで、本研究では、IoT 機器へ個別の ID を付与しなければならない負担、監視サービスに登録を行わなければいけない負担を軽減するために、監視サーバから IoT 機器に対して ID を配布し、監視サーバにて ID を一元的に管理する事を提案し、システムを作成する。サービス管理者は、機器の追加の際に、監視サービスから登録用 ID と URL を取得し、全ての機器に対し同一の ID と URL を設定し、機器を起動させる。機器側では、登録用 ID と URL から、監視サービスにアクセスし、個別の ID の発行を受ける。監視サービス側では、アクセスのあった順に重複の無い ID を発行し、登録を行う。管理者は、機器を起動した順に ID の印刷、貼り付けを行う。

前項に挙げた IoT 機器の監視の為の管理が負担である問題は、機器監視サーバと各 IoT 機器に設定が分散している事が原因であると分析し、機器監視サーバでの設定の一元管理と、各 IoT 機器への個別の設定の簡略化により、負担を軽減することを目的とする。

### 1.4 本論文の構成

第 2 章では、IoT サービスの維持に関する背景と、IoT 機器の監視・管理に関する問題を述べる。第 3 章では、第 2 章で述べた問題を分析し、IoT 機器の監視・管理はどうあるべきか提案を述べる。第 4 章では、IoT 機器監視・管理サービスの実装の詳細について述べる。第 5 章では、実験により IoT 機器監視・管理サービスがもたらす効果を検証し、考察を述べる。第 6 章では、本研究に関する評価について述べる。第 7 章では、本研究を通して得られた知見や今後の課題について述べる。

# 第2章 IoT サービスの提供における問題

## 2.1 IoT サービスの構造と事例

IoT とは、「モノのインターネット」とも呼ばれる、様々なモノがインターネットを介して通信し合うことで自動化を図る概念である。IoT サービスとは、IoT による利便性を顧客へ提供するものである。IoT サービスは、図のように IoT 機器、ネットワーク（インターネット）、サーバから構成されている。

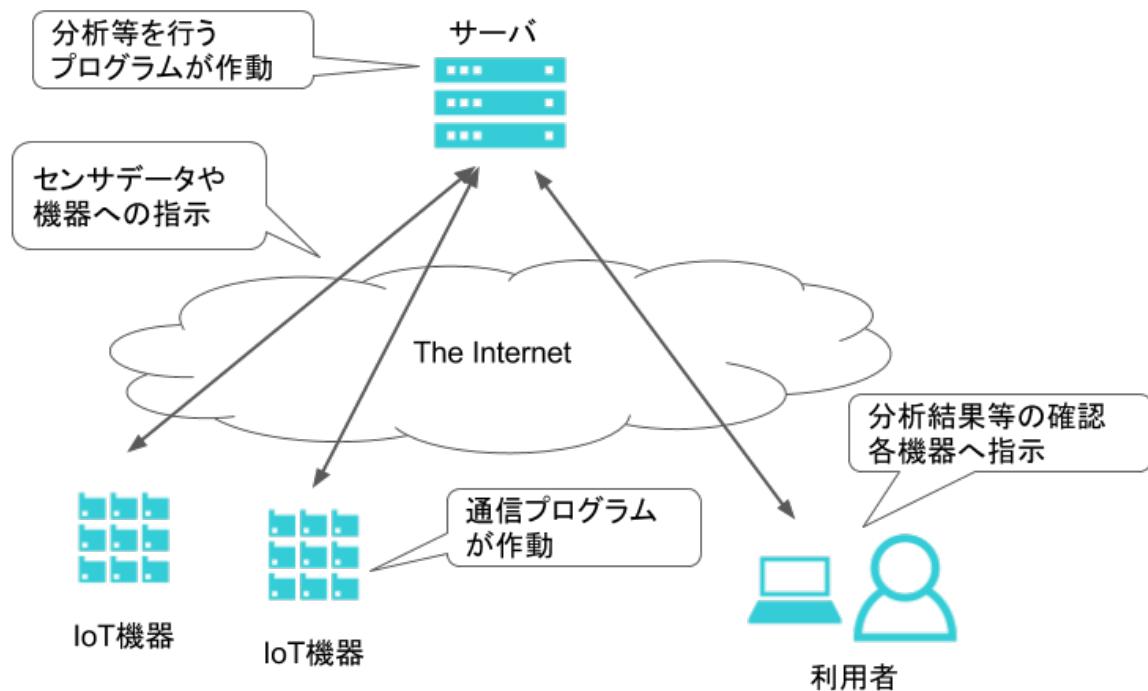


図 2.1: IoT サービスの構造図

IoT 機器は、RaspberryPi や Edison 等の小型コンピュータや組み込み機器が使用され、周辺環境を検知する他、周辺環境へ働きかける事を行う。IoT 機器では、通信プログラムが作動しており、インターネットを介して、検知した周辺環境に関する情報や、IoT 機器への指示をやり取りする。IoT 機器は、Wifi や有線接続等を用いて、インターネットへ接続された携帯電話網や社内 LAN 等のネットワークに接続される。

サーバは、インターネット上に設置され、蓄積・分析・可視化等を行うプログラムが作動している。このプログラムは、IoT 機器から送られてきた情報を蓄積・分析し、必要に応じて IoT 機器へ指示を送る他、分析結果を利用者へ可視化する。利用者は、サーバへインターネットを介して接続し、分析結果の閲覧・機器への指

示を行う。

## 2.2 IoT サービス開発の事例

### 2.2.1 株式会社ルナネクサスでの事例

株式会社ルナネクサスでは、太陽光発電事業を展開している事業主に対し、発電に係る機器の状態や、発電量等を可視化できるサービスを展開している。太陽光発電は、太陽光パネルと発電した電力を送電する為の「パワーコンディショナ」と呼ばれる機器で成り立っている。

株式会社ルナネクサスでは、そのパワーコンディショナに独自に開発した IoT 機器を取り付け、発電量や発電機器の異常などを収集する。収集したデータは、SORACOM Air という携帯電話網を利用したインターネット接続サービスを使用してインターネット上にあるサーバへ送信される。サーバ上では、各 IoT 機器から送られてきた情報を蓄積し、可視化する。これによって、発電事業を展開している事業主に、各発電所まで行かなくても発電量や発電機器の異常等を確認することができる、という利便性を提供している。図 2.2 は株式会社ルナネクサスの IoT サービスイメージである。太陽光発電所は僻地にあることが多いので、利用できるネット

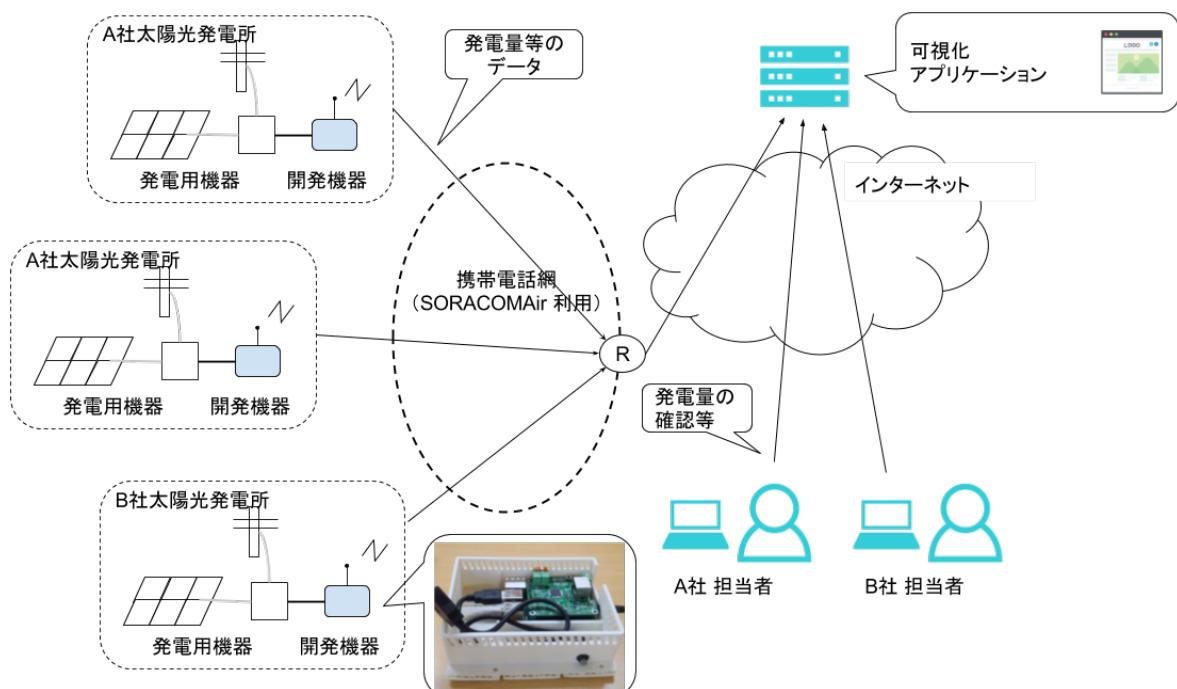


図 2.2: 株式会社ルナネクサス サービスイメージ図

ワークが無いことが多い。このことから、インターネットへの接続に SORACOM Air を選択した。

しかし、SORACOM Air 回線は、プライベートアドレスを使用しており、IoT 機器からの通信は通過するものの、インターネット側から Ping などによる確認を行うことが出来ない。そのため、各 IoT 機器の監視とメンテナンスの為に、VPN を利用し、定期的に VPN 越しにログインすることで監視を行っている。VPN 越し

にログインするためには、各 IoT 機器に VPN クライアントを導入し、新たに VPN サーバを立ち上げる必要があった。また、手動で各機器へログインすることは、手間である。

### 2.2.2 岡本商店街における事例

岡本商店街とは、神戸市東灘区にある阪急岡本駅と JR 摂津本山駅の間に位置する商店街のことである。商店街の方に人流を可視化する IoT サービスを提供し商店街の活性化に役立てるといった趣旨で、2016 年 2 月 7 日から 2016 年 3 月 14 日まで観測を行った。人流観測とは、各地点から各地点迄をある時に移動した人数を観測するものである。今回は、ある地点を通過した人物が以前どの地点を通過していたのかを観測したかった為、携帯電話についている WiFi 機能が送出する電波を利用して観測を行うこととした。

観測・分析・可視化を行うシステムの構成としては図 1 の様になる。各店舗に設置させてもらった観測機器は、店舗に敷設されたネットワーク回線を通して、インターネットへと繋がっている。各観測機器は、観測したデータをインターネット上のサーバへと送信し、そのサーバの上で蓄積・分析・可視化を行う形となっている。

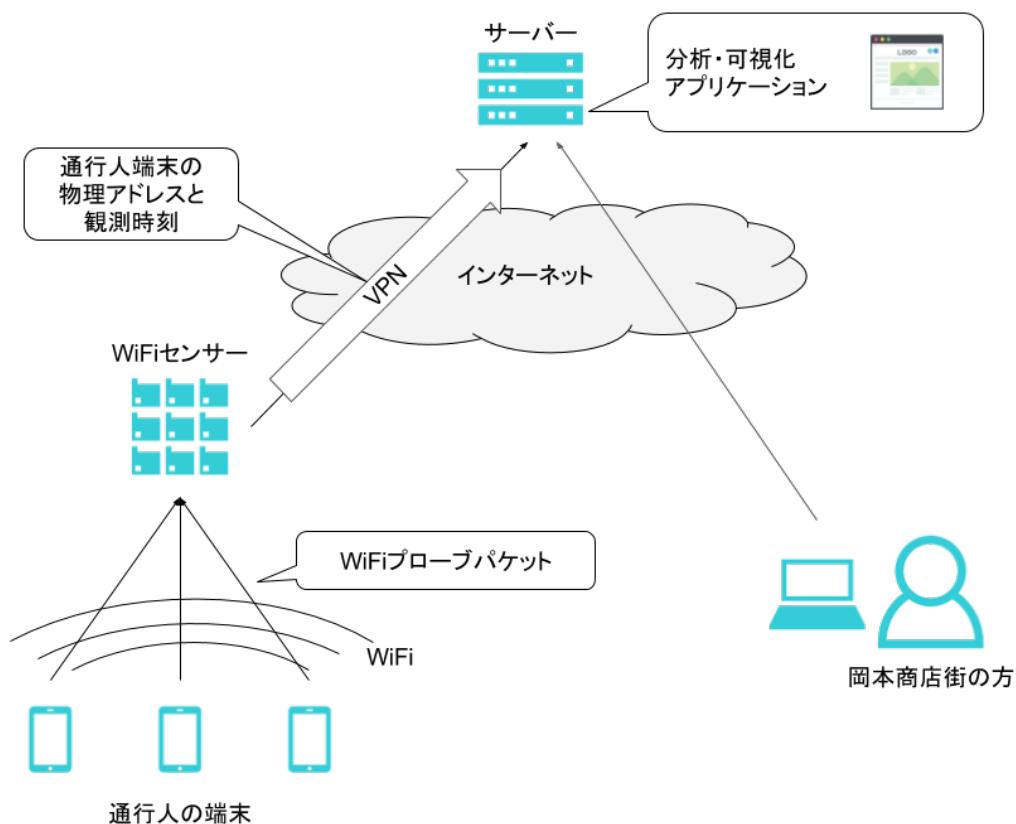


図 2.3: 岡本商店街人流観測 構成図

実験の際に、次の様な問題や負担があった。

- 電源が抜けている・ネットワークが切断されているといった要因で、観測できていないことがあった。

- 上記要因にて観測できていない事に気づくことが遅れた .

これらから , IoT 機器の監視が必要と考え , VPN 越しにログインすることで , 確認を行うこととした . しかし , 各観測機器に VPN を介してログインすることは手間であり , 負担であった . また , 実験では観測機器の数が 5 台のみであったが , 大規模に観測を行うことを考えた場合 , 手動での確認はとても大きな負担となる . そのため , 効率化を図る必要があったと感じる .

## 2.3 IoT サービスの提供者の課題

これら事例と実験から IoT サービスの円滑な提供において , 開発は容易だが , IoT サービスの構造を維持するためには , IoT 機器の監視・管理が必要となっている事が分かった . しかし , IoT 機器の監視・管理には , 次のような課題がある .

- IoT 機器が多量である

そのため , 各機器へ設定することは大きな負担となる . また , 監視サーバへの登録も負担となる .

- IoT 機器が接続されるネットワークが多様である

そのため , 既存の問い合わせによる監視手法は適応できない . しかし , 通知による監視を行うには , 各機器へ設定する必要がある .

- IoT 機器の追加や交換が頻繁にある

そのため , 監視サーバへの登録作業を頻繁に行う必要がある .

- 外観が似通っている

そのため , 交換のため現地に行くも , 交換すべき機器がどれなのか判別がつかない .

私は , その中でも , 監視サーバへの登録の負担と IoT 機器への設定の負担について取り上げ分析することにした . IoT 機器の監視のためには , IoT 機器が接続されるネットワークの多様性から , IoT 機器からの通知による監視手法を取らざる終えない . しかし , そのためには IoT 機器と監視サーバに対し , 識別子 (ID) を設定する必要がある . IoT 機器に対し , 固有の識別子を与え , 監視サーバに識別子を伝えることで , 監視サーバはどの機器からの通信なのか見分けることができる . また , 監視サーバに対し , 予め識別子を設定しておくことで , 監視対象とする機器以外からの不正なアクセスを遮断することができる . よって , IoT 機器へ識別子を設定し , 監視サーバに対して登録を行う必要がある . しかし , IoT サービスでは , IoT 機器が多量に使用されることや , 頻繁な交換・追加・撤去がありうるため , これら設定や , 設定の変更は大きな負担となっている .

# 第3章 IoT 機器からの通知に基づく機器監視サービスの提案

## 3.1 既存手法の問題点

既存の監視手法として次のような手法がある .

- 監視サーバからの問い合わせによる監視  
Ping,SNMP による問い合わせ等がこの手法にあたる .
- 機器からの通知に基づく監視  
SNMPTrap による通知がこの手法である .

監視サーバからの問い合わせによる監視を IoT サービスに適応した場合 , 次のような問題がある .

- IoT 機器が接続するネットワークが , プライベートアドレスを使用している場合がある  
IoT 機器が接続するネットワークがプライベートアドレスである場合 , 監視サーバから問い合わせを行うことができず , 監視することができない .
- IoT 機器が接続するネットワークにセキュリティの設定が施されている場合がある  
特に , インターネットからの特定ポートに対するアクセスをブロックしている場合がある . この場合も , 監視サーバからの問い合わせを行うことが出来ないため , 監視することができない .
- IoT 機器が移動する場合がある  
IoT 機器は移動する物に取り付けられる場合がある . この場合 , IoT 機器が接続するネットワークが頻繁にかわるため , 監視サーバが問い合わせを行う宛先を頻繁に更新しなければならない . また , 複数の機器が移動を繰り返す中で , 監視サーバとのやり取りに混乱が生じる .

このため , 監視サーバからの問い合わせによる監視は用いることができない .

一方 , 機器からの通知では , これら問題を解決する事ができる . しかし , 各 IoT 機器に対し個別の ID を設定する必要があり , IoT 機器が大量であることから大きな負担となる . また , どちらの手法を取るにしても , 機器監視サーバへの登録作業は行う必要があり , 負担となる .

### 3.2 IoT 機器監視・管理サービスの提案

前章で述べたように、IoT サービスの円滑な提供の為には、IoT サービスの構造を維持する必要が有り、そのためには、IoT 機器を監視する必要がある。IoT 機器が接続するネットワークが、プライベートアドレスを利用している等、多様であることから、機器からの通知による監視手法を取る必要がある。しかし、IoT サービスにおいて、IoT 機器は多量に使用されることや、頻繁な交換・追加がありうることから、IoT 機器へ個別の設定をすること、監視サーバへ IoT 機器を登録することが大きな負担となっている。

そこで私は、これら負担は IoT 機器と監視サーバに設定が分散していることが原因と考え、設定を監視サーバにて一元的に管理し、IoT 機器への設定を省力化する事を提案し、サービスを開発する。このサービスの利用者は、提案するサービスから、機器追加用のトークンを受け取り、各 IoT 機器に対して同一のトークンを設定し、提供するプログラムを作動させる。提供するプログラムは、監視サーバに対して、このトークンを送信し、返答として個別の ID を受け取り、自身に設定する。サーバでは、トークンの有効性を検証し、機器の追加を自動的に行う。図 3.1 は、提案するサービスの構成図である。

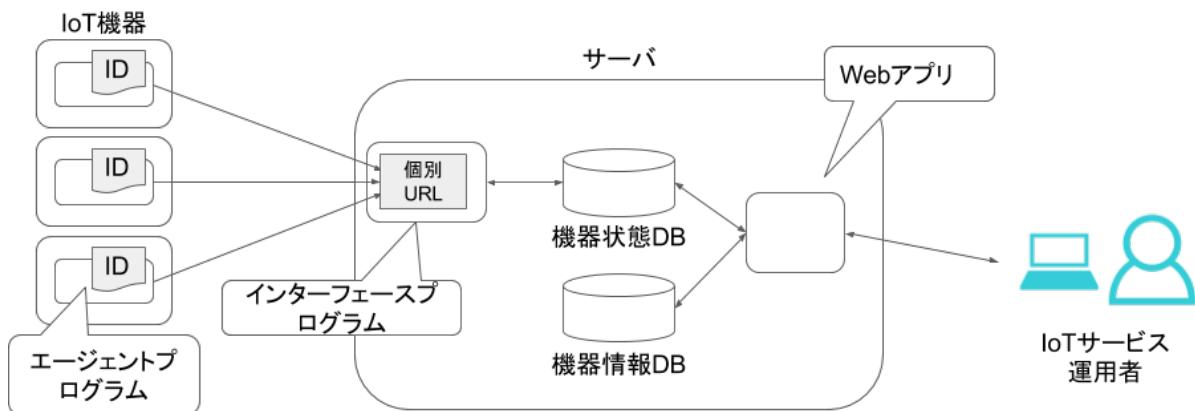


図 3.1: サービス構成図

プログラムを作動させた順に、登録された事を確認し、機器に対して ID の貼り付け等を行う。また、IoT 機器上で動作する提供プログラムは、定期的に監視サーバに対しネットワークの状態や稼働状況等を報告することで、監視を行う。

これにより、監視サーバへ各 IoT 機器を登録する負担と、IoT 機器へ個別の設定を行うことを省力化する。さらに、通知型で https を用いた通信により、ネットワークの多様性に対応する。また、サービスとして提供することで、従来では不可欠であった監視サーバの構築作業の負担も解決することができる。

# 第4章 機器監視サービスの実装

## 4.1 要件と機能の抽出

以下に要件を整理する。

- IoT 機器への設定が簡略化されること
- IoT 機器が接続されるネットワークの違いによる影響を受けないこと
- 監視サーバ上の論理的な機器と監視対象である物理的な機器との対応がわかりやすいこと

上記要件から、次のような機能を抽出した。

- 機器からの通知による監視情報の収集
- 監視サーバにて機器 ID と URL の対応を管理することで、監視対象機器を特定
- 機器の ID を機器本体に掲示することで、外部からわかりやすくすること

## 4.2 提案サービスの構成

提案サービスは図 4.1 のように、IoT 機器上で動作するエージェントプログラム、監視サーバ上で動作するエージェントプログラム用インターフェース、機器状態データベース、機器情報データベース、可視化アプリケーションから構成されている。エージェントプログラムは、設定により与えられた機器追加用トークンを監視サーバに送信し、重複無い機器 ID を取得する他、定期的に監視サーバに対して、ネットワークの状態や稼動状態を通知する。エージェントプログラム用インターフェースは、エージェントプログラムから送信された機器追加用トークンの整合性を確認し、機器 ID の発行、データベースへの登録を行う。また、エージェントプログラムから送信されたネットワークの状態や稼動状態をデータベースへ書き込む。機器状態データベースは、機器の状態を蓄積するために使用し、機器情報データベースは、ユーザと機器 ID を結びつけ、機器に関する情報を記憶するために用いる。可視化アプリケーションは、各データベースと連携し、機器追加用トークンの発行や、ユーザ管理、機器状態の可視化を行う。ユーザは可視化アプリケーションへブラウザからアクセスすることで、本サービスを利用する。

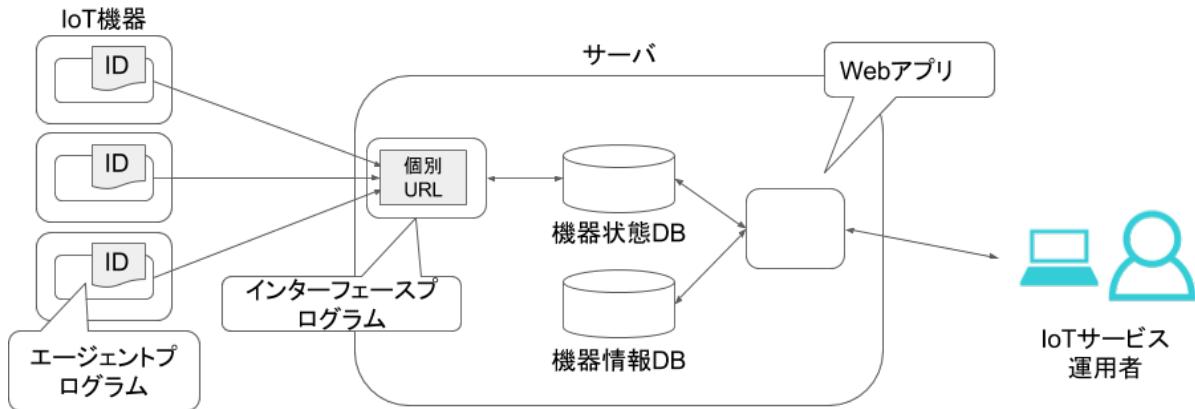


図 4.1: サービス構成図

### 4.3 エージェントプログラムの設計と実装

エージェントプログラムは、IoT機器上で動作し、IDの取得機能と、定期的な状態の通知機能を持つ。IDの取得機能とは、ユーザから設定されたトークンを監視サーバへ送信し、機器固有のIDを取得・設定を行う機能である。定期的な状態の通知機能とは、1分毎にネットワークの状態、機器の状態を送信する機能である。ネットワークの状態とは、IoT機器から監視サーバへの到達性であり、「監視サーバへ接続できなかった回数」の事とする。また、機器の動作状態とは、エージェントプログラムがサーバへネットワークの状態を通知できているかどうかであり、通知できている場合は「異常なし」と判断する。過去のいつの時点で通信が途切れ、いつの時点で通知不能になったのか、判断するために「過去にサーバへ接続できなかった回数」も合わせて送ることとした。

動作の流れとしては、図のようになる。まず、IoT機器はユーザより設定されたトークンを監視サーバへ送信し、個別のIDを取得する。このIDを自身へ設定しなおし、1分おきに監視サーバのIDと紐付いたURLに対して「過去にサーバへ接続できなかった回数」「現在サーバへ接続出来なかった回数」を送信する。「過去にサーバへ接続できなかった回数」及び「現在サーバへ接続できなかった回数は、サーバへの接続ができた時点で初期状態(0回)に戻る。

実装としては、対象となるIoT機器をRaspberryPiと仮定し開発した。エージェントプログラムは、汎用性を高めるため、シェルスクリプトで作成した。監視サーバへの通信には、セキュリティの設定の有無(ネットワークの多様性)を考慮し、httpsを用いる。

### 4.4 エージェントプログラム用インターフェースの設計

エージェントプログラム用インターフェースは、監視サーバ上で動作するプログラムで、トークンの検証機能、IDの発行機能、登録機能、機器状態の受け付け機能を持つ。トークンの検証機能とは、IoT機器から送信

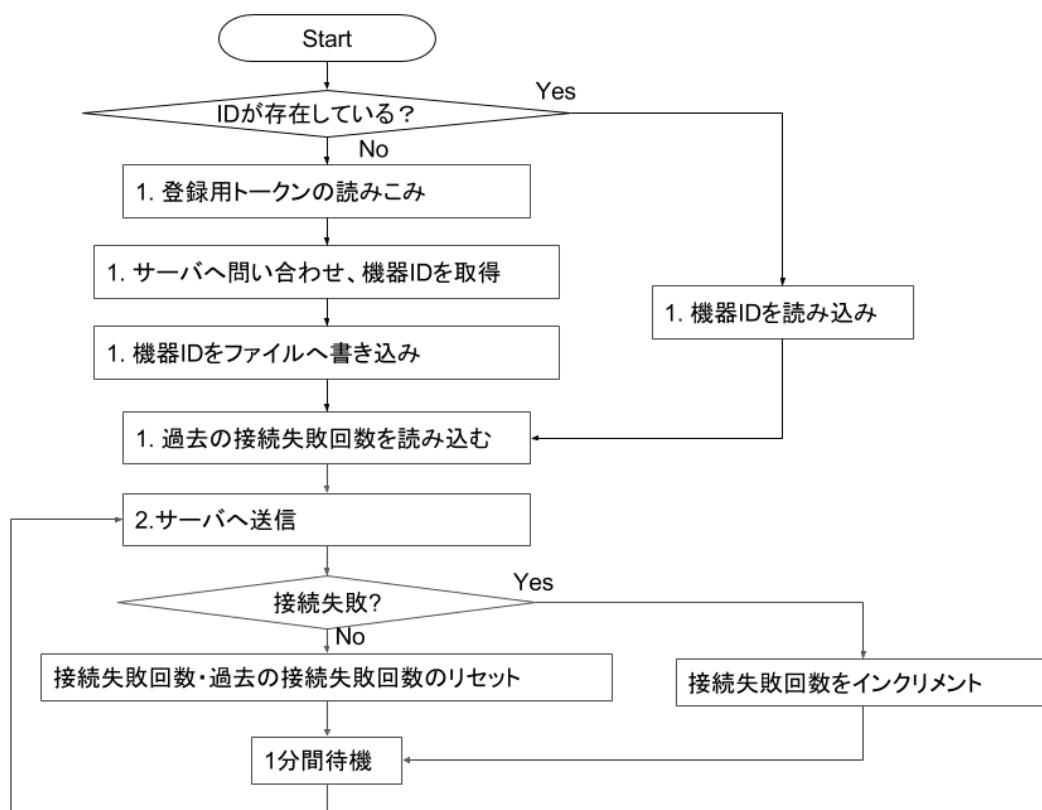


図 4.2: エージェントプログラムの動作

されたトークンの整合性を検証する機能である。トークンの整合性とは、トークンに紐付いたユーザが現在機器の追加を許可しているかである。ID の発行機能とは、機器に対し、重複のない ID を発行する機能である。登録機能とは、可視化アプリケーションへ、機器とユーザの関係を登録する機能である。機器状態の受け付け機能とは、IoT 機器から送信された機器状態等から、過去のどの時点でネットワークから切断されたのか等を逆算し、機器状態データベースへ書き込みむ機能である。

図??は、エージェントプログラムからトークンが送られてきた場合の動作である。IoT 機器から、トークン

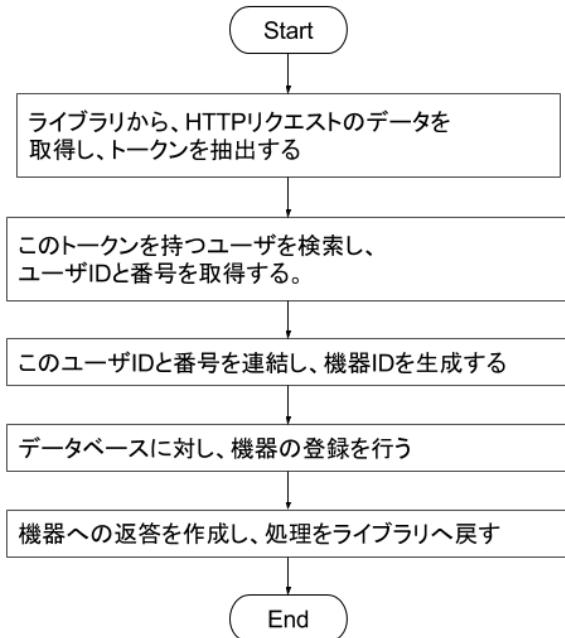


図 4.3: エージェントプログラム用インターフェースの動作 (機器から登録用トークンが送られてきた場合)

が送られてきた場合、インターフェースは次のような動作を行う。

1. トークンの整合性の確認
2. ユーザ情報の取得
3. 機器 ID の生成
4. 機器情報データベースへ関係を格納 (登録)
5. 機器に対し、機器 ID を返信

図??は、エージェントプログラムから機器の状態が通知してきた場合の動作である。

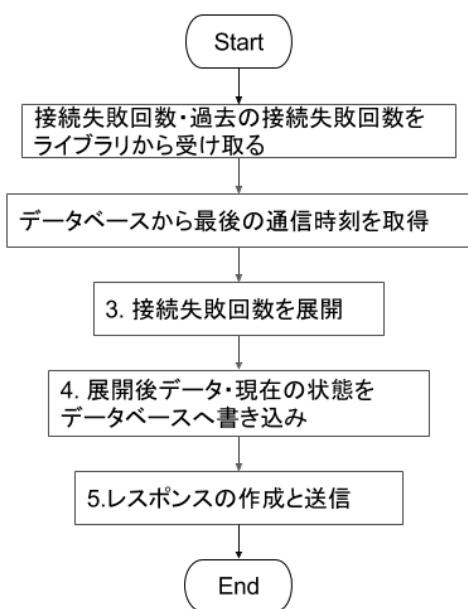


図 4.4: エージェントプログラム用インターフェースの動作（機器からの状態の通知があった際）

また、IoT 機器から機器の状態が送られてきた場合は、次のような動作を行う。

1. 機器 ID と URL の合致を確認
2. 過去にサーバへ接続できなかった回数と、データベース上の最後の通信の記録から、通知不能になった時刻の推測
3. 現在サーバへ接続出来なかった回数と、現在時刻から、通知可能になった時刻の推測
4. 機器状態データベースに対し、通知不能になった時刻と、通知可能になった時刻、現在時刻の 3 時点に対し、状態の変化を書き込む
5. 現在時刻と受理した旨を返信

実装としては、Python3 を用い、Falcon と呼ばれる WebAPI の作成に特化したライブラリを使用した。機器 ID は、トークンと乱数を、sha256 と呼ばれるハッシュアルゴリズムを使用してハッシュ化した物を使用する。可視化アプリケーションへの登録は、機器情報データベースへの書き込みをもって、登録とすることとした。このプログラムを用いて、エージェントプログラムと通信を行うために、次のような URL を定義した。

- GET /deviceid
- POST /deviceid<sub>i</sub>

先頭の GET・POST は、HTTP リクエストを表す。また、deviceid<sub>i</sub> は、各機器の機器 ID を表す。

## 4.5 エージェントプログラムとエージェントプログラム用インターフェース間の通信

エージェントプログラムとエージェントプログラム用インターフェース間の通信には、HTTPS を用いる。機器 ID の取得/発行・登録時は図 4.5 のような動作をし、機器状態の通知/監視の際は、図 4.6 のような動作をする。

各通信の書式は JavaScript Object Notation(JSON) という形式を用いる。トークンを「xxxxxxxx」、機器 ID を「yyyyyyyy」、過去にサーバへ接続できなかった回数と現在サーバへ接続できなかった回数をそれぞれ「pppp」と「qqqq」とすると、図中の各通信は次のようなメッセージとなる。

1. GET /deviceid
  - { "token": "xxxxxxxx" }
2. { "deviceid": "yyyyyyyy" }

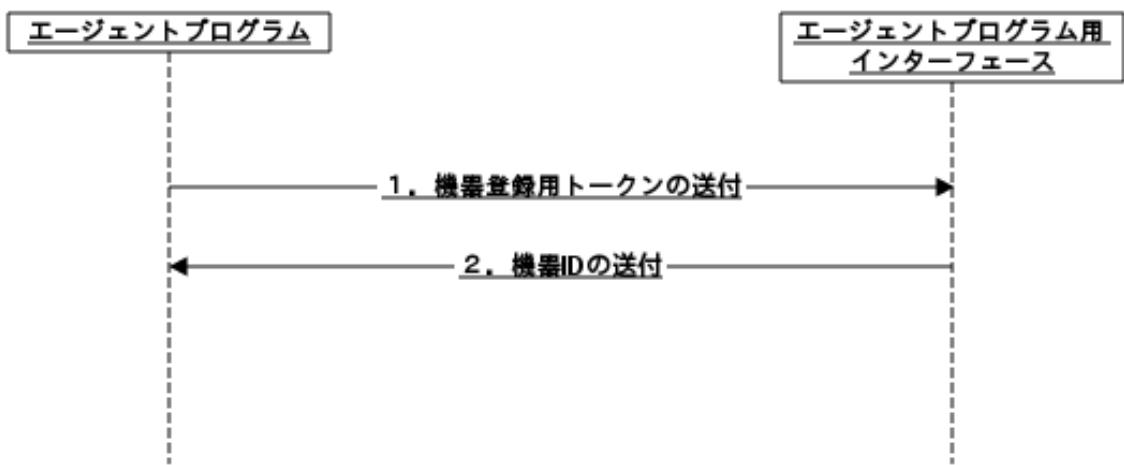


図 4.5: 機器 ID の取得/発行・登録時の通信

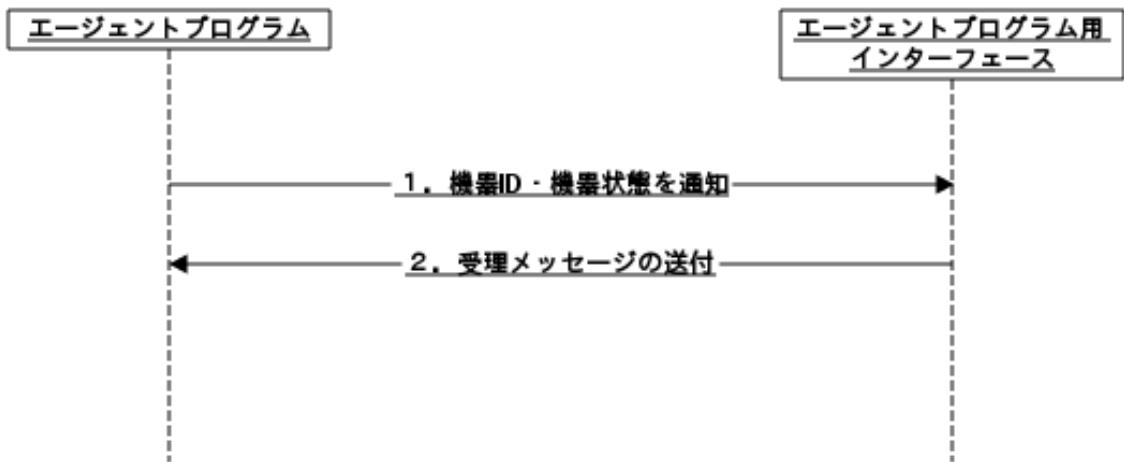


図 4.6: 機器状態の通知/監視時の通信

3. POST /yyyyyyyyy
 

```
{ "past": "pppp", "now": "qqqq" }
```
4. { "stat": "OK" }

また、トークンが不正であった場合や、デバイス ID が存在しない場合等は、サーバはデバイスに対して、HTTP Not Found(404) を送信する。

## 4.6 機器状態データベースの設計

機器状態データベースは、機器ごとに機器 ID と同一の名前のテーブルを作成し、機器状態を管理する。この機器 ID と同一の名前を持つテーブルを機器状態管理テーブルとする。この機器状態管理テーブルは、表 4.1 のような構造を持つ。

| 表 4.1: 機器状態管理テーブル |          |        |
|-------------------|----------|--------|
| 論理フィールド名          | 物理フィールド名 | 型      |
| 時刻                | time     | time   |
| 状態                | stat     | string |

データベースは Influxdb を利用した。InfluxDB では、時刻を主キー (index) として扱うので、このような構造になっている。状態には、OK・NC が入り、OK は機器に異常が無かったこと、NC は、機器に異常は無かつたがサーバへ到達できなかった事を表す。また、各テーブルはエージェントプログラム用インターフェースが自動で作成し、可視化アプリケーションから、必要に応じて参照、削除を行う。

## 4.7 機器情報データベースの設計

機器情報データベースは、ユーザと機器の関係を格納するために使用される。ユーザテーブル、機器情報テーブルの 2 つのテーブルが有り、ユーザに関する情報と、機器に関する情報を分けて記録する。図??は、ユーザテーブルと機器情報テーブルの関連を表す。ユーザ 1 つに対し機器は 0 以上あり、機器 1 つに対しユーザは 1 つである。ユーザテーブルと機器テーブルは、ユーザ ID で紐付いている。

表 4.3 はユーザテーブルの構造を表している。ユーザテーブルでは、サービスへのログイン時に使用するユーザ名とパスワードの他、内部で識別のため使用するユーザ ID、機器登録の際に使用されるトークン、機器 ID 生成の為に使用されるシーケンス番号が記録される。

表??は機器情報テーブルの構造を表している。

ユーザテーブル・機器テーブルは、エージェントプログラム用インターフェース・可視化アプリケーションから参照される。データベースには、sqlite3 を用いた。

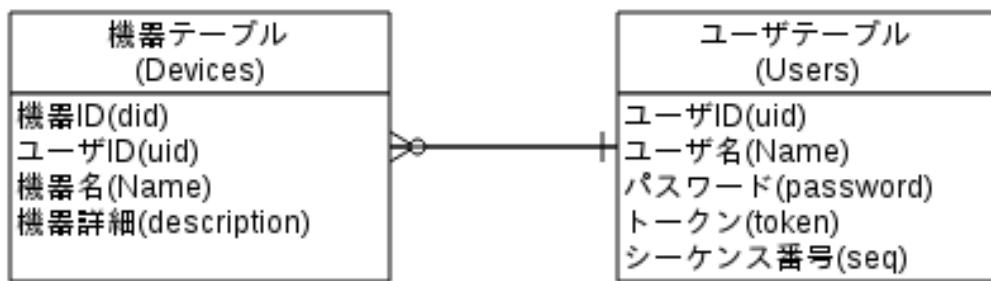


図 4.7: ユーザテーブルと機器情報テーブルの関連

表 4.2: ユーザテーブル (Users)

| 論理フィールド名 | 物理フィールド名 | 型       | 制約              |
|----------|----------|---------|-----------------|
| ユーザ ID   | uid      | integer | primary key     |
| ユーザ名     | Name     | string  | unique not null |
| パスワード    | password | string  | not null        |
| トークン     | token    | string  | unique          |
| シーケンス番号  | seq      | integer |                 |

## 4.8 可視化アプリケーションの設計

可視化アプリケーション次のような構造になっている。サーバーサイドプログラムは Python3 を使用し，Flask という Web アプリケーションフレームワークを用いて作成した。クライアントサイドプログラムは，HTML/CSS，Javascript を使用し，Bootstrap,Jquery というライブラリを用いた。

## 4.9 サービス利用のイメージ

ユーザが本サービスを利用して行うことのできる動作は、つぎのようになっている。

### 4.9.1 機器の追加

機器の追加は、次のような操作にて、行うことができる。

表 4.3: 機器情報テーブル (Devices)

| 論理フィールド名 | 物理フィールド名    | 型       | 制約                      |
|----------|-------------|---------|-------------------------|
| ユーザ ID   | uid         | integer | foreign key primary key |
| 機器 ID    | did         | string  | primary key             |
| 機器名      | name        | string  | not null                |
| 機器詳細     | description | string  |                         |

1. サービスにログインする
2. サービスの画面からトークンの発行を押し，トークンを保存する．
3. 各機器に対して，エージェントプログラムのインストールと，トークンの設定を行う．
4. 機器の電源を入れる
5. 機器が追加された事を確認し，機器 ID を機器に対してラベル等を貼り付ける．
6. 順次他の機器に対して同様の操作を行う．

#### 4.9.2 機器の削除

機器の削除の利用イメージは，次のようにになっている．

1. サービスにログインする
2. サービスの画面から，該当の機器を削除する
3. 物理的な機器の撤去を行う

### 4.10 実装

時間的制約から，機器からの通知による監視のみを実装した．付録としてソースコードを添付する．ソースコードは [github](#) にも上がっている．

# 第5章 機器監視サービスの動作テスト

## 5.1 検証目的

IoT 機器の監視・管理の際の問題が解決できたのか、検証を行う。

- 各 IoT 機器の状態を監視するために、多数の IoT 機器へ設定をしなければいけない負担が軽減された事
- 増減や交換の度に、機器監視システムへの登録をしなければならない負担がなくなったこと
- 機器からの通知に基づいた監視ができている事

実装の都合から、この 3 つの要件の中で機器からの通知に基づいた監視のみの検証を行う。機器から、機器 ID と接続出来なかった回数等を、デバイス ID に紐付いた URL に送信し、監視サービスにて、監視することができる事を確認する。そのため、各 IoT 機器に対する設定・サーバでの登録は、手動で行う。

## 5.2 検証方法

検証は次のような小規模な IoT サービスを想定し、行った。IoT 機器の数は、1台とし、RaspberryPi2 を使用することとした。OS は Raspbian jessie がインストールされていることとした。図 5.1 は使用した RaspberryPi2 と、IntelEdison である。今回は、IntelEdison は使用していないが、参考の為に上げた。

RaspberryPi2 は無線 LAN インターフェースを持たないので、バファロー製の無線 LAN インターフェースを使用した。期間は 2017 年 1 月 28 日正午から 1 時間行い、途中何度も IoT 機器の電源を抜き、正常に検知することを確認する。また、期間初めに、想定する IoT 機器の設定を行い、監視サービスにて発行した ID を設定する。

## 5.3 検証結果

まず、サービス側で機器 ID の登録を行った。ブラウザからサービスへログインし、登録ボタンをクリックして、登録用ダイアログを開いた。ここで、登録用ダイアログに表示されている ID をコピーしておく。登録用ダイアログにて、機器名と機器の詳細（機器名を「RaspberryPi A」、機器の詳細を「TestDeviceA」とした）を入力し、登録ボタンを押した。その後、サービスの画面上に「RaspberryPi A」という名前を持つデバイスが作成されたことを確認した。

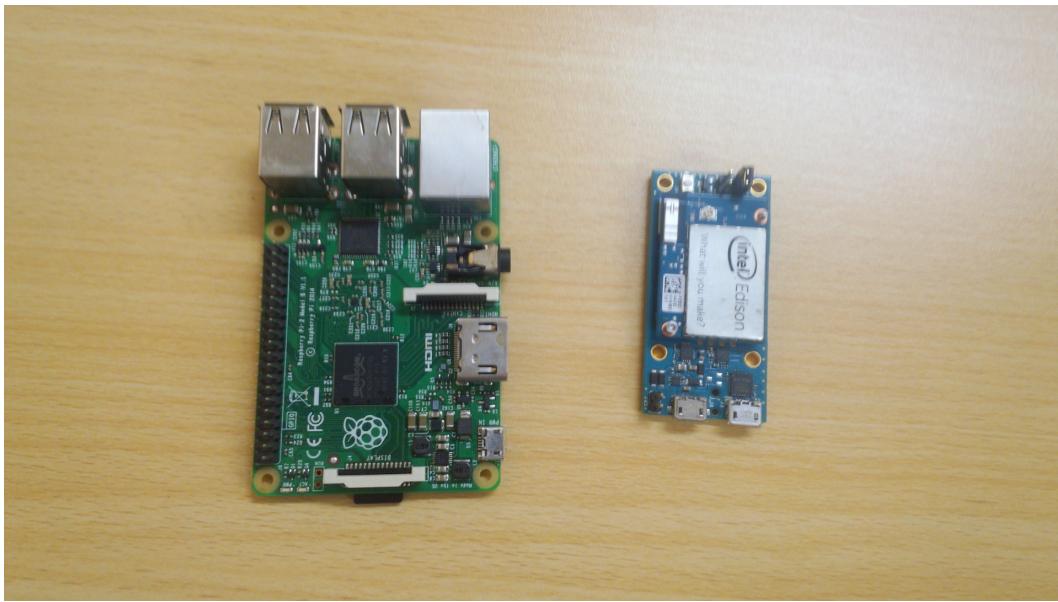


図 5.1: IoT サービスの構成図

次に，エージェントプログラムを機器の SD カードのホームディレクトリとなるディレクトリ（/home/pi/）へ，agent.sh という名前で書き込んだ。また，起動時にエージェントプログラムを実行するための設定ファイルを機器の SD カードの設定用ディレクトリ（/etc/systemd/system/）へ，devmon.service という名前で書き込んだ。この設定ファイルは，起動時に/home/pi/agent.sh のスクリプトを機器 ID を引数として実行するように指定している。

その後，機器の電源を入れ，機器上でエージェントプログラムが稼働していることを，systemctl status コマンドを用いて確認した。また，サービス側で指定した ID を持つ機器の状態が異常状態（稼働していないかネットワークから切断されている事を表す）から，正常状態（稼働していて，ネットワークが切断されていない事を表す）へ変化した事を確認した。機器から該当の ID に紐付いた URL に対し，通信が出来ている場合のみ，正常状態となるので，正常に ID と URL が紐付いている事が確認できたといえる。

また，電源ケーブルを抜き，サービス上で該当の機器の状態が正常状態から異常状態へ変化することを確認した。変化には 2 分程度かかり，これは，機器が 1 分おきに通知を行っていることと，ブラウザ側のプログラムにて 1 分おきにサーバへ該当機器の状態を問い合わせているためである。再度電源ケーブルを挿し直すと，1 分程度でサービス上の機器の表示が異常状態から正常状態へ変化した。

これらから，監視対象機器と監視サーバが正常に連携し，機器の監視が行えることが分かった。

## 5.4 考察

提案する方法で IoT 機器からの監視が行えることは分かったが，下記問題については実装の都合上，確認が取れていない。

- 各 IoT 機器の状態を監視するために、多数の IoT 機器へ設定をしなければいけない負担
- 増減や交換の度に、機器監視システムへの登録をしなければならない負担

今後、実装を進め、検証を行うこととする。

また、検証に置いて次のような改善点が見つかった。

- 過去の記録の表示

現在、過去の機器状態の記録については、文字記録として表示しているが、グラフ表示等の方が見やすいと感じた。また、期間を指定して閲覧できる機能も必要であることが分かった。

- アラート機能

監視サービスの画面を見続けるわけには行かないので、メール等によるアラート機能等を追加する必要があると感じた

- グループ表示機能

グループ表示等が行えると尚見やすい事が分かった。

- 一覧表示

機器が少量である場合は問題ないが、数千台となった場合に、一覧での表示は、見難い事が分かった。全体やグループでの稼働率等を表示し、確認が必要な機器のみを一覧で表示する等の工夫が必要となることがわかった。

また、本来ならば、IoT サービスを提供している企業に使用して頂き、評価を得る必要があったが、双方のスケジュールの都合と開発の遅れから行うことが出来なかった。しかし、今後 IoT サービスの開発が盛んになることや、使用する IoT 機器の数が多くなることから、本サービスの必要性は高くなっていくと考えられる。

## 第6章 考察

検証では、機器 ID と URL の組をサーバ側で管理することで、機器からの通知に基づいて監視可能であることを確認した。検証では、サーバへの機器の登録、機器への ID の付与は手動で行ったが、機器からの通知による監視、サーバ上で URL と ID の組み合わせの管理について、確認を行った。これにより、機器の追加・移動の際に、機器への設定の手間や、機器が接続されるネットワークの調査等の手間を省くことができた。また、機器への ID 付与や機器の登録の作業が負担となつたので、実装を進め、省力化を図り検証を行う事を考えている。

- サーバから機器 ID を付与することで、機器への設定を簡略化し、負担を軽減する事
- IoT 機器の変更や追加の際に、サーバに対し登録を行わなければならない負担が軽減された事

今後の課題として、次があげられる。

- IoT 機器のセキュリティ

現状、IoT 機器から監視サーバまでの間は、SSL を用いることで暗号化している。しかし、IoT 機器自身を分解することで、機器 ID を取得し、機器のなりすましが行われる危険がある。そのため、なんらかの方法で機器へのなりすましを検知し通知することや、機器に耐タンパ性を持たせる必要があるように考える。

- 物理との紐付けの改良

現状では、機器に機器 ID を貼り付ける事で解決しているが、現地に行った際にどの機器 ID が交換の必要があるのか、見比べなくてはならない。数十桁の機器 ID を一瞬で判別することは困難であり、改良の余地があるように思う。具体的には、QR コードを用いて、スマートフォン等から監視サービスの特定の機器の状態閲覧画面にアクセスすることができるようになる等が必要である。その際には、セキュリティの観点から、一般の利用者へ対する説明等の表示と、機器の管理者に対する表示を分ける必要が出てくると考えている。

また、現状では、一旦サービスに登録されたことを確認してからラベリングとしているが、QR コードを使用した場合、機器に対して予めラベリングを施しておくことでより設定の手間が省けるように感じている。具体的な実装としては、監視サービス側にて QR コードをまとめて発行し、ユーザは、QR コードを電源を入れる順番に貼り付けていく。機器側は、アクセスのあった順に発行した QR コードに埋め

込まれた機器 ID を割り振っていく。これにより、物理的なヒモ付の手間や、交換の際の手間等がより省けると感じている。

- 通知間隔の問題

機器から監視サーバへの通知は、現状では 1 分おきに行っているが、各種 IoT 機器にてこの 1 分が妥当なものなのか検証する必要があるように感じている。何故ならば、省電力を目的として、機器自体のスリープや、ネットワークからの離脱等を行う場合があるためである。このような場合、監視サーバにて通知間隔の設定を行うことで、誤認等を防ぐ必要がある。また、該当の機器の状態が不安定である場合など、意図的に監視間隔を短く設定することもできるよう考慮する必要があると考える。

- 機器の操作

RaspberryPi 等、IoT 機器によっては、電源が急に抜けた場合、記録していたデータが失われてしまう物がある。そのため、撤去の際に電源を切る操作を行ってから、電源を切る必要がある。しかし、多くの IoT 機器は、画面やキーボード等はついておらず、現地に画面やキーボードを運び接続するか、遠隔から操作する他無い。また、現地にエンジニアを派遣する必要もある。この手間を解決するために、機器の監視サービスの機能として、電源の OFF・再起動ボタンや、遠隔からのログイン等を行えるようにする必要も感じる。

- IoT サービスとの連携

IoT サービスは、収集したデータの分析や可視化を行う。そのため、IoT 機器が現在稼働しているのか、過去に置いて稼働していたのか、等を知る必要がある。現状では、分析などの際に手動で入力することや、ある程度の誤差を容認することで解決しているが、機器監視サービスが、監視情報を IoT サービスに対して提供することでも解決できるのではないかと考えている。そのために、監視サービスが IoT サービスとのインターフェースを持つ必要があると考えている。

## 第7章 おわりに

IoT とは、様々なモノにコンピュータを取り付け、インターネットを介して相互に情報をやり取りすることで、様々な自動化を図ろうという概念である。IoT サービスとは、IoT による利便性をユーザーに提供するもので、IoT 機器とサーバーのプログラムがインターネットを介して通信し合うことで成り立っている。IoT サービスの円滑な提供のためには、この構造を維持しなければならない。そのため、IoT 機器の監視が不可欠である。

本論文では、その IoT 機器の監視がサービス提供者にとって負担となっている事を取り上げた。株式会社ルナネクサスが提供している太陽光発電の発電量の監視のための IoT サービスを主にとりあげ、聞き取りを行った。その中で IoT 機器の監視が負担である事を問題として捉え、分析した。また、岡本商店街にて行った実験から、監視における技術的課題を明確にした。

その中でも、技術的制約から IoT 機器へ個別の ID を設定することの負担、監視サーバに対して監視対象機器を指定する負担を取り上げ、これら負担は、IoT 機器と監視サーバに対して、個別に整合性の取れた設定をしなければならない事が原因と考えた。

そこで、本研究では、監視サーバにて、設定を一元的に管理することで、これら問題を解決した。各機器が接続されるネットワークが多様である問題を、IoT 機器から通知を送ることで解決し、各機器への設定と監視サーバへの登録が負担である問題を、監視サーバにて各 IoT 機器の設定を管理することで解決した。

この機器監視サービスを実現する為、IoT 機器にインストールされるエージェントプログラム、機器監視サーバ上で動作するエージェントプログラム用インターフェース、Web アプリケーションを作成した。

また、要件を満たしているか検証を行い、IoT 機器の動作状態について監視できていることを確認した。今後の課題として、IoT 機器への設定の簡略化や、ユーザインターフェースの向上、様々な IoT 機器への対応があることがわかった。本来ならば株式会社ルナネクサスにて使用してもらい評価を得るべきところではあるが、スケジュールの都合により、実現しなかった。しかし、今後 IoT サービスの普及がより進み、使用する IoT 機器の数が多くなることから、本サービスが求められることが推測される。

## 謝辞

本論文は、著者が神戸情報大学院大学情報技術研究科情報技術専攻在学中に、横山研究室にて行った研究をまとめたものです。本研究に関してご指導ご鞭撻を頂きました横山輝明講師に心より感謝申し上げます。また、本論文をご精読頂き、有用なコメントと励ましをくださった藤原明生准教授に深謝致します。

本研究を進めるにあたって、インタビューや訪問を受け入れてくださった、株式会社ルナネクサス 藤戸様に感謝申し上げます。そして、論文を書く際にアドバイスをくださった、嶋教授と、嶋研究室の渡邊香織さん、田頭潤さん、笠谷拓伸さん、また、研究についてアドバイスを頂いた横山研究室卒業生の鄒曉明さんと、良き議論相手である京都産業大学大学院修士1年目の石原真太郎さんに感謝致します。

# 参考文献

- [1] トレンド・イノベーション 稲田修一「ビッグデータ活用でビジネスはどう変わったか～コマツにおけるモノのインターネット事例から考える～」<https://www.salesforce.com/jp/blog/2013/12/vol3-bigdata.html> (2017年2月10日 閲覧)
- [2] Richo JAPAN Corp. 「出力機器のリモート管理サービス「@Remote」」<https://www.ricoh.co.jp/remote/> (2017年2月10日 閲覧)
- [3] 日経テクノロジーオンライン 高野 敦 「[ 生体センシング ] "着る "センサーで健康情報を計測」<http://itpro.nikkeibp.co.jp/article/COLUMN/20140526/559230/> (2017年2月10日 閲覧)
- [4] 株式会社 SORACOM 「SORACOM の概要」<https://soracom.jp/overview/> (2017年2月7日 閲覧)
- [5] TechCrunch Japan 「【詳報】ソラコムがベールを脱いだ、月額300円からのIoT向けMVNOサービスの狙いとは？」<http://jp.techcrunch.com/2015/09/30/soracom-launches-mvno-service-for-iot/> (2017年2月7日 閲覧)
- [6] 株式会社沖縄アイオー 金城辰一郎 「ソラコムによるIoTサービス内容とは？非エンジニアがその革新的な魅力と導入事例をわかりやすく徹底解説」[http://okinawa.io/blog/tech/soracom\\_iot](http://okinawa.io/blog/tech/soracom_iot) (2017年2月7日 閲覧)
- [7] IoTNews.jp 小泉耕二 「【前編】SORACOMが発表した新サービスは、なにがすごいのか？ SORACOM Connected」<https://iotnews.jp/archives/12037> (2017年2月7日 閲覧)
- [8] THE BRIDGE Takeshi Hirano 「SORACOMの凄さは第三者が「SIM」を自由に発行・運用できること――IoT向けモバイル通信PF、ソラコムが提供開始」<http://thebridge.jp/2015/09/soracom> (2017年2月7日 閲覧)

## .1 岡本商店街での事例

### 実験概要

2015年12月8日から2016年2月26日まで、NPO法人コミュニティリンクへのインターンシップの一環として、岡本商店街にて人流観測を行った。岡本商店街とは、神戸市東灘区にある阪急岡本駅とJR摂津本山

駅の間にある商店街のことである。実験は、商店街の方に人流を可視化する IoT サービスを提供し、商店街の活性化に役立てるといった趣旨で行った。観測は、2016 年 2 月 7 日から 2016 年 3 月 14 日まで行った。

人流観測とは、各地点から各地点迄をある時に移動した人数を観測するものである。通常は、観察員がカウンタを用いて数えるが、それでは各地点間を移動した人数はわかるが、その人が以前どの地点に居たのかはわからない。そこで、携帯電話についている Wifi 機能を利用し、観測を行うこととした。携帯電話の Wifi 機能は、無線 LAN の接続に使われるが、接続毎に Wifi 機能を有効にすることが手間なため、常時 ON にしている人も少くない。携帯電話の Wifi 機能を有効にしている場合、携帯電話から接続可能な無線 LAN を探す為、プローブパケットというものが定期的に送出される。プローブパケットには、そのプローブパケットを送出した機器の物理アドレスが含まれており、個々の機器が識別可能である。そのプローブパケットを複数地点で観測し、含まれている物理アドレスと受信時刻を照合することで、携帯電話端末を持った人がどのように移動をしたのかが分かる。岡本商店街では、この原理を利用して、人流観測を行った。

#### 観測・分析・可視化システムの構成

岡本商店街の 5 店舗に開発した観測機器を設置し、サーバにて蓄積・分析・可視化を行った。構成としては、観測機器内で動作するプログラムが定期的にサーバーに観測データを送信する。サーバ上では、分析プログラムと可視化プログラムが動作しており、観測データは分析プログラムへ渡される。分析プログラムは、観測データを可視化プログラムの要求によって分析し、結果を可視化プログラムへ渡す。可視化プログラムは、ユーザからの操作によって分析プログラムを呼び出し、分析結果を Web インターフェースによって可視化する。このシステムのユーザーは、ブラウザから Web インターフェースへアクセスすることで、操作・分析結果の閲覧をすることができる。図 1 は、開発したシステムの構成を示している。

開発した観測機器は、RaspberryPi と Baffalo 製の Wifi ドングル、ampsence・rsync というソフトウェアを利用し作成した。図 2 は、開発した観測機器である。RaspberryPi とは、小型 PC の一つで安価入手が可能である。また、利用した RaspberryPi には Wifi インターフェースが存在しなかったので、Baffalo 製の USB 接続が可能な Wifi ドングルを接続した。ampsence とは、プローブパケットを受信しプローブパケットに含まれる物理アドレスと観測日時をファイルへ記録するソフトウェアである。rsync とは、ディレクトリの中を同期させるプログラムで、遠隔にあるコンピュータの指定したディレクトリの中と、手元のコンピュータの指定したディレクトリを同期することができる。開発した観測機器は、起動時に ampsence を動作させ、定期的に rsync を実行するよう設定した。これによって、各所に設置された観測機器の情報をサーバー上に集約・蓄積する。

作成した分析プログラムは、可視化プログラムから指定された期間内の観測データを読み込み、地点ごと観測時間ごとの物理アドレスの数の推移・地点ごと観測時間ごとの移動した人数の割合を集計し、可視化プログラムへ渡す。作成した可視化プログラムは、ユーザから指定された期間を分析プログラムへ渡し、分析結果を棒グラフや円グラフといった形で、ユーザーへ表示する。図 3 は作成した Web アプリケーションである。

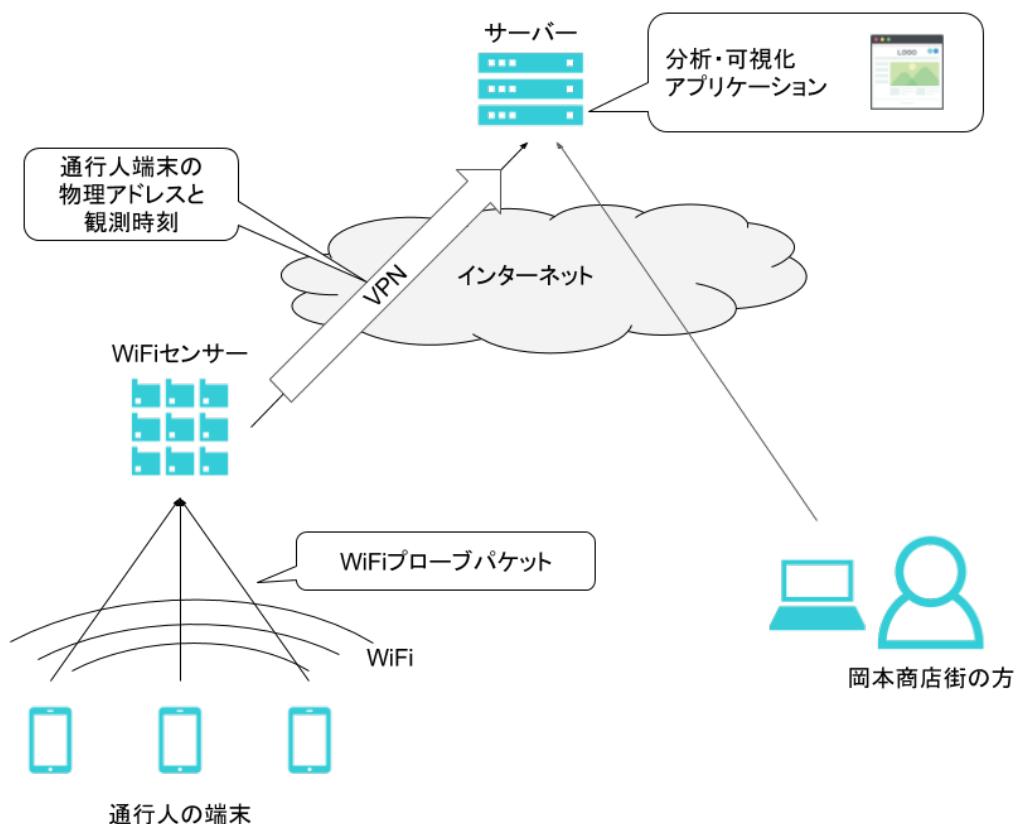


図 1: 岡本商店街人流観測 構成図



図 2: 岡本商店街人流観測 使用した機器

## 岡本商店街動態情報可視化システム

2016年 1月 1日 現在の動態情報



図 3: 岡本商店街人流観測可視化アプリケーションスクリーンショット

観測機器からサーバまでのネットワークは、観測した物理アドレスが流れるという点、ソフトウェアのアップデートの為にログイン可能にしたいといった要望から、VPNを使用した。VPNには、OpenVPNを使用した。そのため、観測機器とサーバにOpenVPNClientとOpenVPNServerをそれぞれインストールし、設定を行った。

また、観測機器、分析・可視化アプリケーションを開発するにあたって、次のような点を考慮した。

- ネットワーク機器の設定が不要であること

機器が設置されるネットワークは、店舗のネットワークである。そのため、ネットワーク機器の設定を変更することはできない。よって、観測機器と分析・可視化アプリケーションの連携が、ネットワーク機器の設定に左右されないよう設計する必要があった。

- トラブル対応のしやすさ

機器が設置される場所は、岡本商店街内の店舗である。そのため、設置やトラブル対応、回収については、予め店舗に連絡を入れなくてはならない。また、夜間や店舗の休日に対応することは困難である為、遠隔から観測機器にログインする必要があった。

- 設置箇所の問題

店舗によっては、店舗内に無線ネットワークが存在しなかったので、後述するSORACOM Airを用いることとした。

- データ損失に備える

サーバー側の不良によって観測データが損失する事に備え，機器自体にも情報を蓄積しておくことにした．

## 考察

実験では，次のようなトラブルがあった．

- 設置後，電源が抜けており，観測ができていないことがあった
- 設定ミスにより，観測できていなかったことがあった

その結果，次のような問題が起きた．

- 分析時に，観測できていない期間を推測する必要があった
- 観測データの不足によって，曜日ごとの来客数の動向等の分析を行うことが出来なかつた
- 何度も現地に行き，確認を行わなければならなかつた

その為，観測機器の監視と，観測機器ごとの設定の簡略化が必要であると考えた．しかし，観測機器の監視には次のような技術的困難や制約があり，行うことができなかつた．

- 機器が接続するネットワークの設定を変更することは出来ない
- 店舗ネットワーク及びSORACOMAir のネットワークでは，プライベートアドレスが利用されているため，インターネットから観測機器にアクセスすることが出来ない
- 提供しているサービスに監視機能を組み込む事が難しい
- 新規に監視サーバを立ち上げる事が負担となる