

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 IoT 機器監視・管理サービスの提案	7
3.2 提案サービスの構成	7
3.3 エージェントプログラムの設計と実装	9
3.4 エージェントプログラム用インターフェースの設計と実装	9
3.5 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	12
3.6 機器状態データベースの設計と実装	15
3.7 本サービス利用のイメージ	15
3.8 機器監視サービスの構成	15
3.8.1 エージェントプログラム	16
3.8.2 エージェントプログラム用インターフェース	16
3.8.3 機器状態データベース	16
3.8.4 機器情報データベース	17
3.8.5 Web アプリケーションサーバ	17
3.8.6 監視アプリケーション	18

3.9 機器監視サービスの実装	20
3.9.1 エージェントプログラムの実装	20
3.9.2 エージェントプログラム用インターフェースの実装	22
3.9.3 機器情報データベース	22
3.9.4 機器状態データベース	23
3.9.5 監視アプリケーション	24
3.9.6 Web アプリケーション	24
3.9.7 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の実装	24
3.9.8 Web アプリケーションサーバと Web アプリケーション間の通信の実装	26
3.10 ソースコード	26
3.11 サービスによる監視のイメージ	26
 第 4 章 機器監視サービスの動作テスト	 27
4.1 技術検証	27
4.1.1 検証シナリオ	27
4.1.2 検証結果	28
4.2 サービスの特徴	28
4.3 考察	28
 第 5 章 おわりに	 30
 第 6 章 謝辞	 31
 参考文献	 32
.1 岡本商店街での事例	32

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
3.2 エージェントプログラムの動作	10
3.3 エージェントプログラム用インターフェースの動作	11
3.4 機器 ID の取得/発行・登録時の通信	13
3.5 機器状態の通知/監視時の通信	14
3.6 システムのブロック図	15
3.7 ログイン画面	19
3.8 機器状態一覧画面	19
3.9 機器状態一覧画面（小さく表示）	20
3.10 機器状態詳細表示	20
3.11 機器追加ダイアログ	21
3.12 機器情報編集ダイアログ	21
3.13 過去の状態表示ページ	21
3.14 機器情報テーブルとユーザテーブルの関連	23
3.15 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の間のメッセージシーケンス図	24
3.16 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の間のメッセージシーケンス図	25
4.1 IoT サービスの構成図	27
1 岡本商店街人流観測 構成図	34
2 岡本商店街人流観測 使用した機器	34
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	35

表 目 次

3.1 機器情報テーブル (物理名 : devices)	22
3.2 ユーザテーブル (物理名 : User)	23
3.3	23

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，注目が集まっている。

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。数多く，さまざまなネットワークを介して接続される IoT 機器の監視は困難な問題である。IoT 機器が設置される様々なネットワークの構成を把握することは，IoT 機器が多量であることを考えると現実的ではない。また，従来の監視手法はパーソナルコンピュータを対象としたもので，利用しにくい。現状としては IoT サービス開発者が，IoT サービス毎に実装しているため負担が大きいといった問題がある。そのため，設置されるネットワークに関係なく状態が監視できることが求められる。また，IoT 機器の状態を一覧して確認できることや，IoT 機器の過去の動作状態や通信状態を確認できることが必要である。IoT サービスの開発者の負担を減らすためにも，IoT 機器の監視サービスが必要である。

そこで，我々は，IoT 機器からの通知に基づいた機器監視サービスを提案する。IoT 機器が自身の過去の動作状態や通信状態を記録することで，設置されるネットワークに関係なく状態監視をすることを可能にする。また，サービスを機器監視に特化させ独立させることで，IoT サービスに変更を与えること無く，機器を監視することを可能にする。この仕組みを用いることで，IoT 機器が設置されるネットワークに関係無く状態を監視することや，IoT 機器の過去の状態や通信状態を確認することを容易にし，IoT サービス開発者はサービスの開発に専念できる。本研究では，IoT 機器からの通知による機器の設置環境によらない機器の監視を行うことにより，IoT サービスの維持を容易にするシステムの開発に取り組む。

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。また、IoT 機器の監視のためには、監視サーバと IoT 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。聞き側では、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。

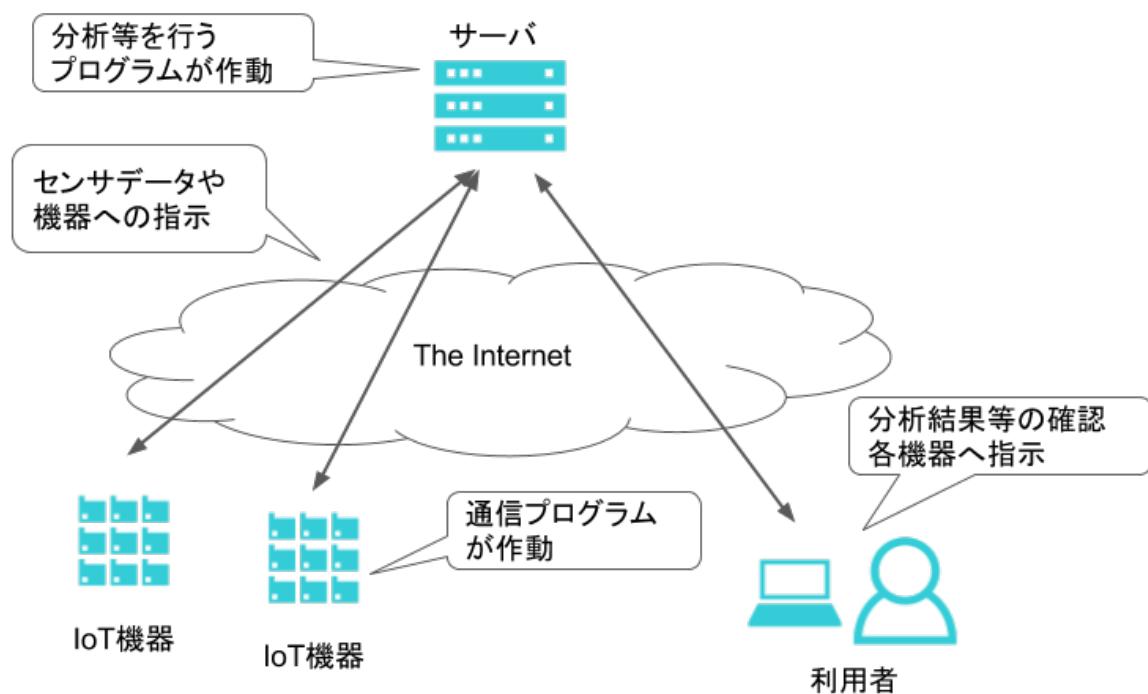


図 2.1: IoT サービスの構造図

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開発している。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

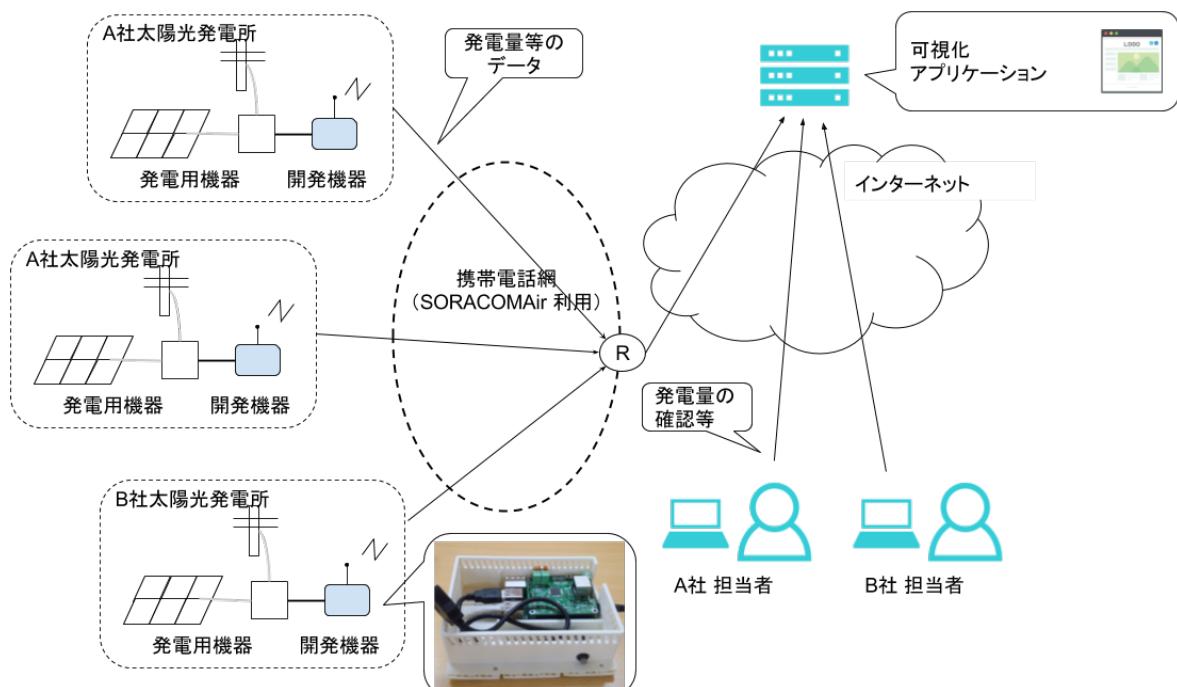


図 2.2: 株式会社ルナネクサス サービスイメージ図

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 サーバをた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 摂津本山駅の間にあ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。

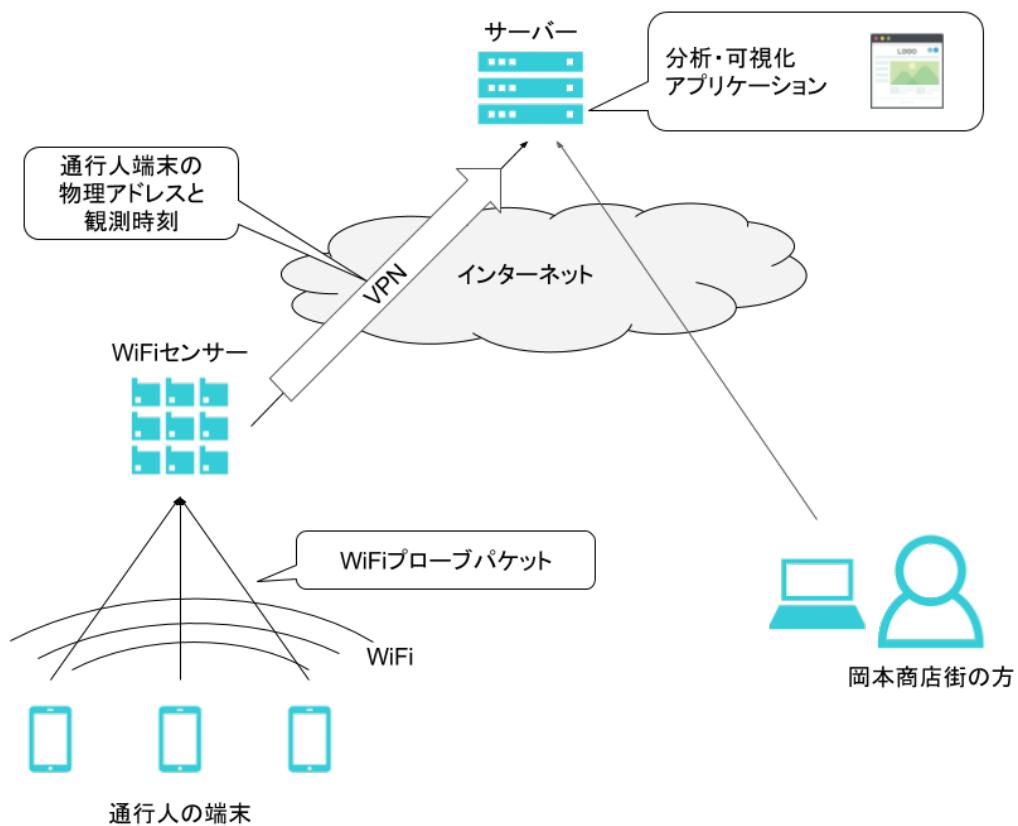


図 2.3: 岡本商店街人流観測 構成図

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 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、IoT 機器を監視する必要がある。IoT 機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。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。

3.2 提案サービスの構成

提案サービスは図のように、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トークンの整合

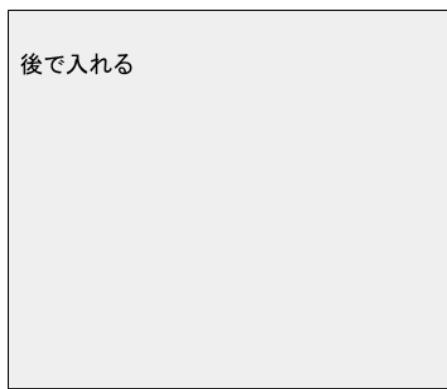


図 3.1: サービス構成図

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、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。

3.3 エージェントプログラムの設計と実装

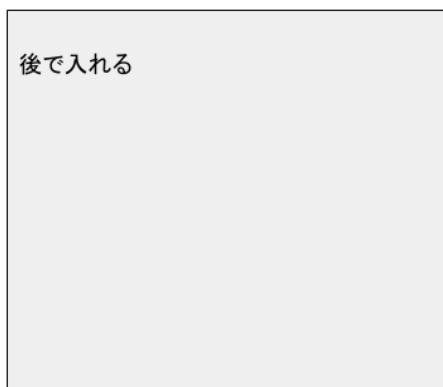
エージェントプログラムは、IoT 機器上で動作し、ID の取得機能と、定期的な状態の通知機能を持つ。ID の取得機能とは、ユーザから設定されたトークンを監視サーバへ送信し、機器固有の ID を取得・設定を行う機能である。定期的な状態の通知機能とは、1 分毎にネットワークの状態、機器の状態を送信する機能である。ネットワークの状態とは、IoT 機器から監視サーバへの到達性であり、「監視サーバへ接続できなかった回数」の事とする。また、機器の動作状態とは、エージェントプログラムがサーバへネットワークの状態を通知できているかどうかであり、通知できている場合は「異常なし」と判断する。過去のいつの時点で通信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

3.4 エージェントプログラム用インターフェースの設計と実装

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



後で入れる

図 3.2: エージェントプログラムの動作

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む機能である。

動作の流れとしては、図のようになる。IoT 機器から、トークンが送られてきた場合、インターフェースは

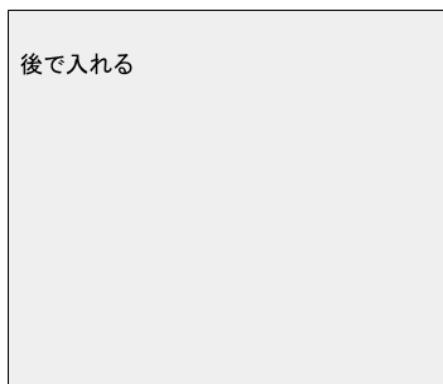


図 3.3: エージェントプログラム用インターフェースの動作

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

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とした。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

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

3.5 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・登録時は図のような動作をし、機器状態の通知/監視の際は、図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「xxxxxxxxx」、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": "xxxxxxxxx" }
```
2. { "deviceid": "yyyyyyyy" }
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。

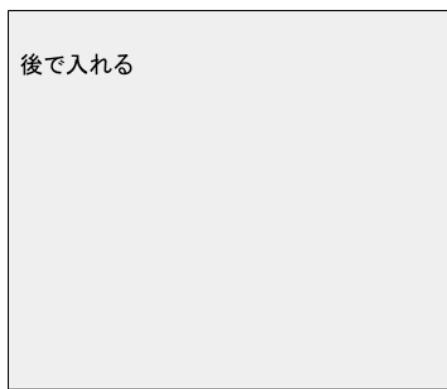
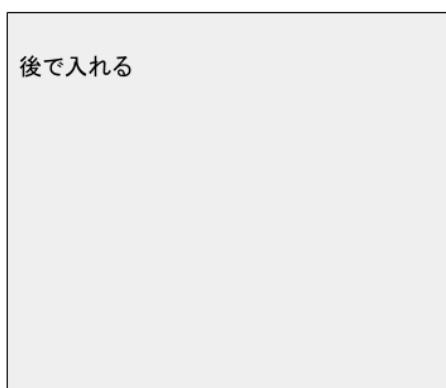


図 3.4: 機器 ID の取得/発行・登録時の通信



後で入れる

図 3.5: 機器状態の通知/監視時の通信

3.6 機器状態データベースの設計と実装

機器状態データベースは、機器ごとに機器 ID の名前をついた。

3.7 本サービス利用のイメージ

ユーザーは、本サービスの利用に際し、次のような動きをする。

1. 予め固有の ID とエージェントプログラムがインストールされた機器を買ってくる
2. ユーザは、その機器に対しサービスで利用するアプリケーションをインストールし、設置する
3. 機器は、設置後ネットワークに接続され次第、サービスへ、機器の状態の通知を行う

3.8 機器監視サービスの構成

前章に述べたアイディアに基づきシステムを構築した。システムは、エージェントプログラム、機器状態データベース、機器情報データベース、エージェントプログラム用インターフェースプログラム、Web アプリケーションサーバ、Web アプリケーションから成り立っている。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、Web アプリケーションサーバと Web アプリケーションは、インターネットを介して通信しあう。図 3.16 は、システムのブロック図である。

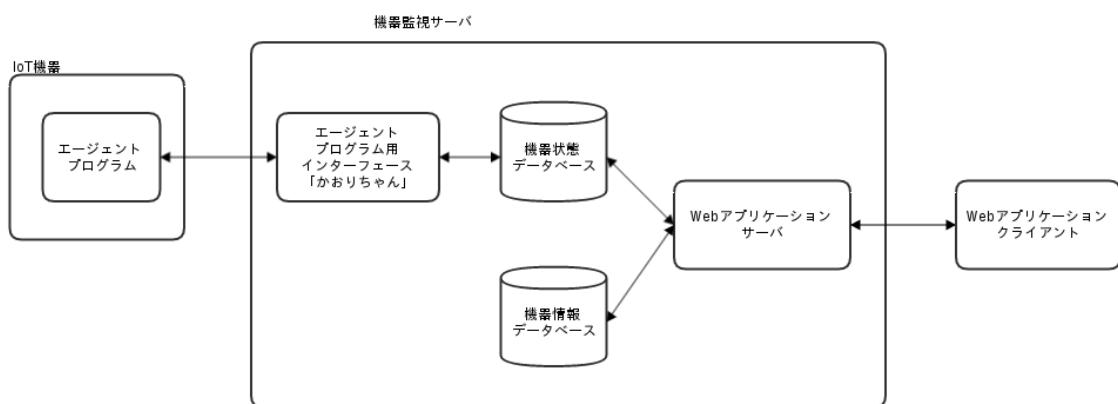


図 3.6: システムのブロック図

エージェントプログラムは、IoT 機器上で動作するプログラムである。定期的に自身の状態を状態蓄積システムへ送信する役割を果たす。定期的かつ自発的に状態を送信することで、ネットワーク環境によらない機器の監視を可能にした。IoT 機器として頻繁に使用される RaspberryPi を想定し作成した。エージェントプログラム用インターフェースは、各 IoT 機器上で動くエージェントプログラムから送られてきた状態を、時刻と

と共に機器状態データベースへ書き込む役割を果たすプログラムで、機器監視サーバー上で動作する。機器状態データベースとは、機器の状態を時系列に沿って蓄積するデータベースである。機器監視サーバー上で動作する。機器情報データベースとは、機器 ID や、機器名、ユーザー名を記録するデータベースである。機器監視サーバー上で動作する。

Web アプリケーションサーバーは、Web ページや Web アプリケーション自体を配信する。ユーザーが利用するブラウザからの要求に答え、現在の機器の状態や、機器名等を返答する。

Web アプリケーションとは、ブラウザ上で動作するプログラムである。ユーザーからの入力を受け付け、ユーザーへ表示する他、必要な機能を問い合わせる。

これら各要素が連携することで、機器の監視を実現している。

3.8.1 エージェントプログラム

エージェントプログラムとは、IoT 機器上にインストールされるプログラムである。エージェントプログラムの役割は、定期的に送信失敗回数をエージェントプログラム用インターフェースへ報告することである。送信失敗回数とは、ネットワークの不具合等により、機器監視サーバーへ送信されなかつた報告の数である。自発的に状態を報告するため、IoT 機器にプライベートアドレスが付与されていても、状態を検知することができる。また、HTTP を用いるため、間のネットワークにてブロックされることがない。

3.8.2 エージェントプログラム用インターフェース

エージェントプログラム用インターフェースとは、サーバー上で動くプログラムである。名前を「かおりちゃん」とした。エージェントプログラム用インターフェースの役割は、エージェントプログラムから送信されたメッセージを受け取り、現在の時刻と正常である旨を機器状態データベースへ書き込む。また、エージェントプログラムから送られた送信失敗回数から、IoT 機器がインターネットから切断された時刻を逆算し、機器状態データベースへ書き込む事も行う。

3.8.3 機器状態データベース

機器状態データベースとは、サーバ上で動作するデータベースである。各 IoT 機器の状態を時刻とともに記録する。機器状態監視システムの中心にあるデータベースである。

3.8.4 機器情報データベース

機器情報データベースとは、サーバー上で動作するデータベースである。各 IoT 機器の機器 ID、機器名、機器詳細情報、ユーザーのメールアドレスとパスワードを記録する。

3.8.5 Web アプリケーションサーバ

Web アプリケーションサーバとは、サーバー上で動作するプログラムである。Web アプリケーションや Web ページの配信、Web アプリケーションからの要求の処理などを行う。必要に応じて、機器状態データベースと機器情報データベースへアクセスを行う。次に、Web アプリケーションとのインターフェースを挙げ、それについて説明する。

ログイン機能

Web アプリケーションから、メールアドレスとパスワードを受け取り、機器情報データベースのユーザー一覧と照合する。照合した結果、ユーザー名とパスワードが合致したユーザーが存在すれば、HTTP クッキーにセッションキーをセットし、機器状態一覧ページへのリダイレクトを返す。合致したユーザーが存在しなかった場合、エラーメッセージを返す。

ログアウト機能

Web アプリケーションに、HTTP クッキーから該当のセッションキーを削除するよう要求する。

機器情報・機器状態取得機能

ログインチェックを行った後、機器情報データベースと機器状態データベースより、全 IoT 機器の機器情報と機器状態を返す。

機器 ID 生成機能

ログインチェックを行った後、機器情報データベースに存在しない、ランダムな機器 ID を返す。

機器 ID 重複チェック機能

ログインチェックを行った後、Web アプリケーションから機器 ID を受け取る。機器情報データベースに該当の機器 ID が存在するかしないかを返信する。

機器作成機能

ログインチェックを行った後，Web アプリケーションから，機器 ID，機器名，機器の詳細と，機器の作成なのか編集なのかの指示を受け取る．機器の作成であった場合，機器 ID に重複が無いことを確認したうえで，機器情報データベースに該当のエントリを作成・機器状態データベースにメジャーメントを作成する．作成されたか，されなかったかを返信する．機器の編集であった場合，機器情報データベースから，受け取った機器 ID を持つものを探しだし，編集する．編集できたか否かを返信する．

機器削除機能

ログインチェックを行った後，Web アプリケーションから，機器 ID を受け取る．機器情報データベースから，受け取った機器 ID を持つものを削除する．その後，機器状態データベースから，メジャーメントを削除する．削除されたか否かを返信する．

過去の機器状態取得機能

ログインチェックを行った後，機器状態データベースから，全ての IoT 機器の過去の状態をまとめ，返信する．

3.8.6 監視アプリケーション

Web アプリケーションとは，ユーザーのブラウザ上で動作するアプリケーションである．ユーザーへグラフィカルインターフェースを提供する．必要に応じて，Web アプリケーションサーバへ必要な情報を要求する．Web アプリケーションは，3 つの Web ページから成り立っている．以下に各ページとそれぞれの役割を述べる．

- ログインページ

正当なユーザーであることを確認するために，メールアドレスとパスワードの入力を求める．メールアドレスとパスワードは，Web アプリケーションサーバーへ送信される．図 3.7 は，ログインページのスクリーンショットである．

- 機器状態一覧ページ

機器状態を一覧して表示する他，機器の作成や，機器情報の編集，機器の削除等を行う事ができる．現在の機器の状態を取得するため，定期的に Web アプリケーションサーバと通信する．また，機器の作成や機器情報の編集，削除の為，Web アプリケーションサーバと通信する．

図 3.8・図 3.9・図 3.10 はこのページのスクリーンショットである．それぞれ，一覧表示・縮小一覧表示・詳細な情報の表示をしている．図 3.11・図 3.12 は，それぞれ，機器追加ダイアログ・機器情報編集ダイアログのスクリーンショットである．

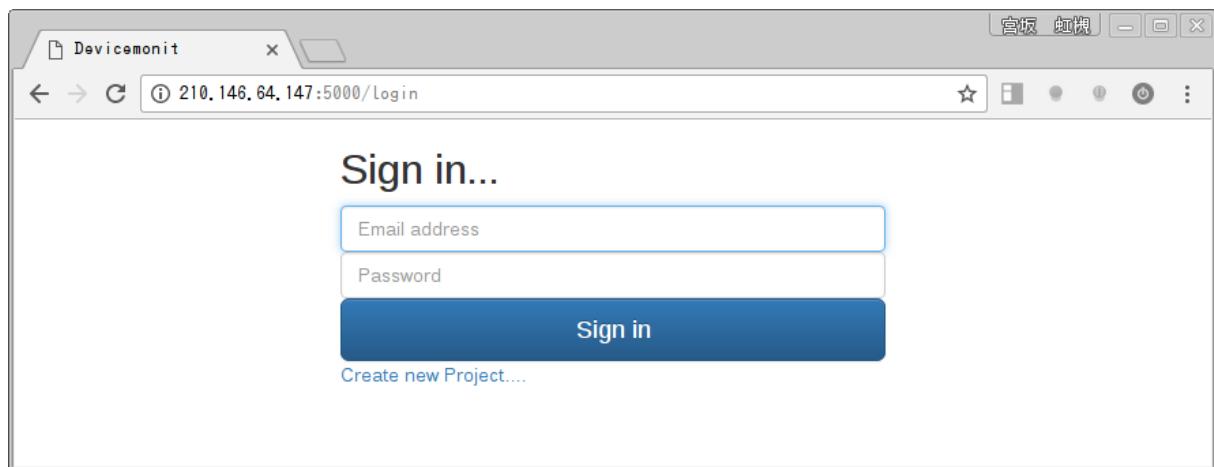


図 3.7: ログイン画面

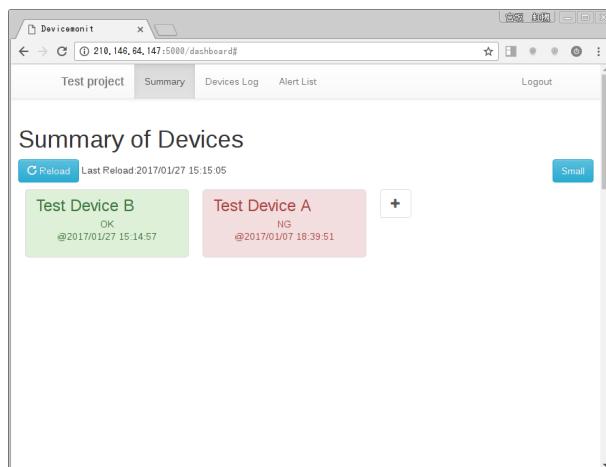


図 3.8: 機器状態一覧画面

機器が多くなっても一覧して分かるよう、小さく表示する表示の切替機能も実装した。

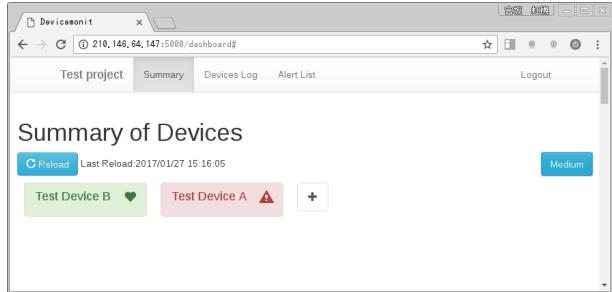


図 3.9: 機器状態一覧画面（小さく表示）

図 3.10 は、各機器の詳細を表示している様子である。各機器を模した図形をクリックすることで、詳細表示の切替が行える。

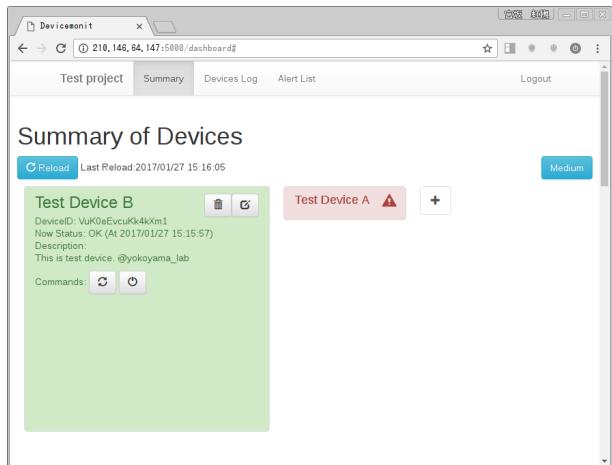


図 3.10: 機器状態詳細表示

図 3.11 は、新たに機器を追加する際に表示されるダイアログである。

図 3.12 は、機器の詳細情報を編集する際の画面である。

- 過去の機器状態一覧ページ

過去の機器状態を時刻と共に整理し、一覧表示するページである。現在の機器の状態を取得するため、定期的に Web アプリケーションサーバと通信をする。図 3.13 は、スクリーンショットである。

3.9 機器監視サービスの実装

3.9.1 エージェントプログラムの実装

エージェントプログラムの役割は、送信失敗回数を定期的に IoT 機器監視サーバーに送信することにある。約 1 分おきに、現在の送信失敗回数と過去の送信失敗回数を、エージェントプログラム用インターフェースへ送信する。どのような Linux 環境でも動作することを考え、Shell スクリプトにて実装した。

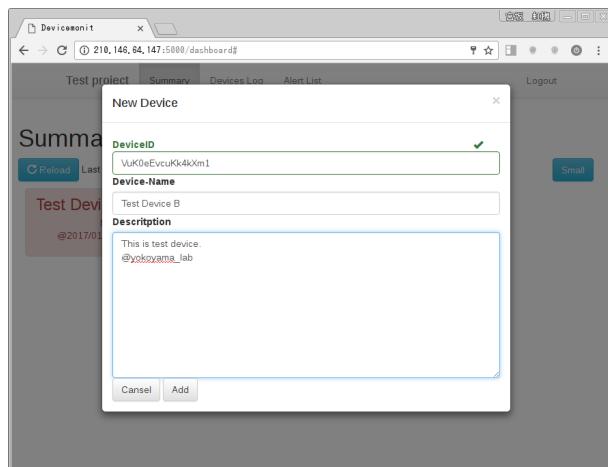


図 3.11: 機器追加ダイアログ

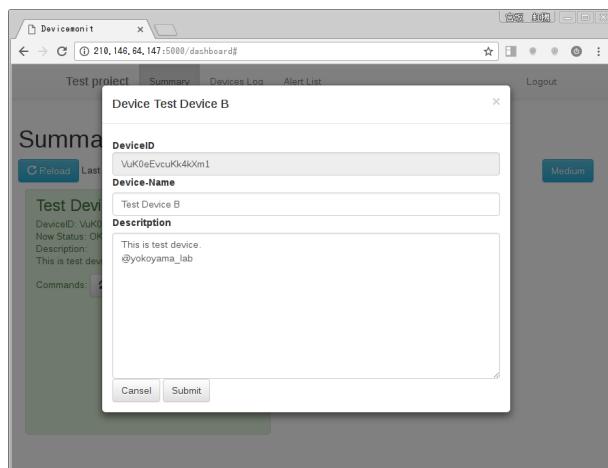


図 3.12: 機器情報編集ダイアログ

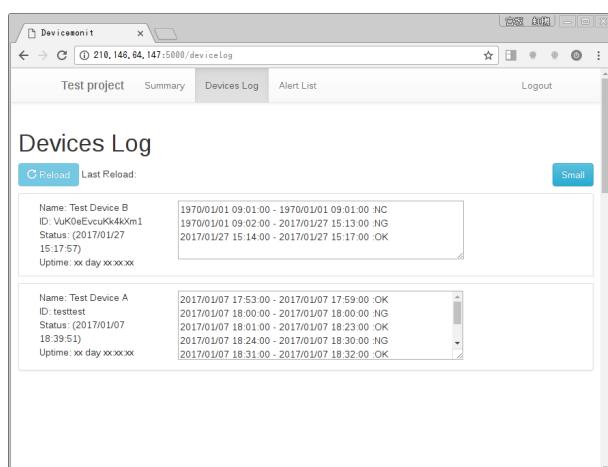


図 3.13: 過去の状態表示ページ

エージェントプログラムの動作は次のようになる .

1. エージェントプログラムが起動されると , まず過去に記録された送信失敗回数を読み出す .
2. エージェントプログラム用インターフェースに対し , 過去に記録された送信失敗回数と , 現在の送信失敗回数を送信する .
3. サーバーから応答があった場合 , 過去に記録された送信失敗回数と , 現在の送信失敗回数をクリアしする .
サーバーから応答がない場合 , 現在の送信失敗回数をインクリメントする .
4. 1分間スリープし , 再び 2 から繰り返す .

3.9.2 エージェントプログラム用インターフェースの実装

エージェントプログラム用インターフェースの役割は , エージェントプログラムから送信失敗回数を受け取り , 時刻と正常である旨を機器状態データベースへ書き込むこと , また , エージェントプログラムから送られた送信失敗回数から , インターネットより切断された時刻を逆算し , 機器状態データベースへ書き込む事である . Falcon と呼ばれる API の作成に特化したフレームワークを使用し , Python にて実装した . また , 機器状態データベースへの書き込みには , InfluxDBClient というライブラリを使用した .

3.9.3 機器情報データベース

機器情報データベースの役割は , 機器の名前 , 機器の詳細説明 , 機器 ID , ユーザー ID , ログイン用メールアドレスとパスワードを記録し保持する事である . ユーザ ID とは , システムにてユーザーを識別するための識別子である . 機器 ID は , システムにて IoT 機器識別するための識別子である . 機器情報データベースには , SQLite3 を用いた . 機器情報データベースには , 次のようなテーブルが用意されている .

機器情報テーブル

機器 ID , ユーザー ID をキーとして , 機器の名前 , 機器の詳細説明を記録し保持するテーブルである .

表 3.1: 機器情報テーブル (物理名 : devices)

論理フィールド名	物理フィールド名	型	制約
機器 ID	did	Text	Primary key
ユーザ ID	uid	Integer	Primary key
機器名	name	Text	
詳細説明	description	Text	

ユーザー テーブル

ユーザー ID をキーとして、ユーザー名とパスワードを記録し保持するテーブルである。

表 3.2: ユーザ テーブル (物理名: User)

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, auto increment
メールアドレス	email	Text	
パスワード	pass	Text	

各 テーブル の 関連

図 3.14 は、各 テーブル の 関連を 表 して いる。ユーザ テーブル と 機器情報 テーブル は、ユーザ ID にて 関連付け かれ て いる。ユーザ は 0 以上 の 機器情報 と 関連付け られ、機器情報 は 一 つ の ユーザ に 関連付け られ て い る。

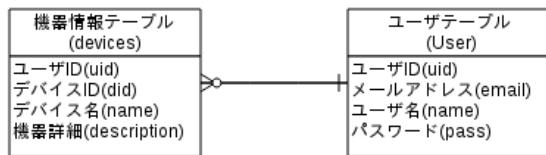


図 3.14: 機器情報 テーブル と ユーザ テーブル の 関連

3.9.4 機器状態データベース

機器状態データベースの役割は、機器の状態を時刻と共に記録・保持することである。機器状態データベースには、Influxdb を用いた。機器 ID をメトリクス名(テーブル名)とし、時刻をキーとして、機器の状態を記録している。

表 3.3:

フィールド名	型	制約
時刻	Timestamp	
状態	Text	

3.9.5 監視アプリケーション

監視アプリケーションの役割は、与えられた HTTP リクエストを元に、Web ページや、各種情報を返却することにある。Flask と呼ばれる Web アプリケーションフレームワークを用いた。Python を使用している。Web サーバーアプリケーションの設計と Web アプリケーションの設計から、下記の様に実装した。やり取り先頭に付いている GET や POST は HTTP メソッド、/login 等は URL を示している。図 3.15 は、サーバー上で動く監視アプリケーションとユーザのブラウザ上で動く Web アプリケーションとのやり取りを表している。

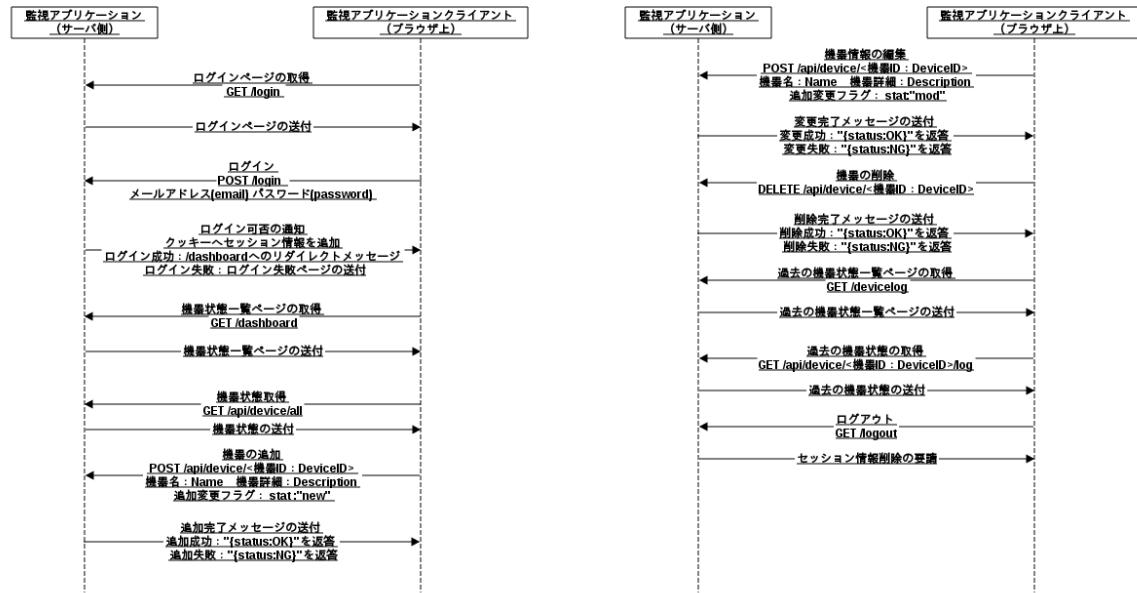


図 3.15: エージェントプログラムとエージェントプログラム用インターフェースの間のメッセージシーケンス図

3.9.6 Web アプリケーション

Web アプリケーションの役割は、ユーザーインターフェースを提供することである。Bootstrap, JQuery というライブラリを用いて作成した。HTML,CSS,Javascript で書かれている。定期的に Web アプリケーションサーバから状態を取得し、HTML,CSS を用いて表示する。

3.9.7 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の実装

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は、インターネットを介して、HTTP というプロトコルを用いて通信する。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は次の様な形になる。

1. エージェントプログラムがエージェントプログラム用インターフェースに対し、送信失敗回数を報告する。

2. エージェントプログラム用インターフェースは、時刻と正常である旨を機器状態データベースへ送信する。
3. 機器状態データベースより、正常に書き込んだというメッセージが返ってくる。
4. エージェントプログラム用インターフェースは、必要に応じて送信失敗回数からインターネットから切断された時刻を逆算し、その時刻と切断されていた旨を機器状態データベースへ送信する。
5. 機器状態データベースより、正常に書き込んだというメッセージが帰ってくる。
6. 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に対し、正常に受け付けたというメッセージを返す。

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の間の通信は、約1分おきに繰り返される。

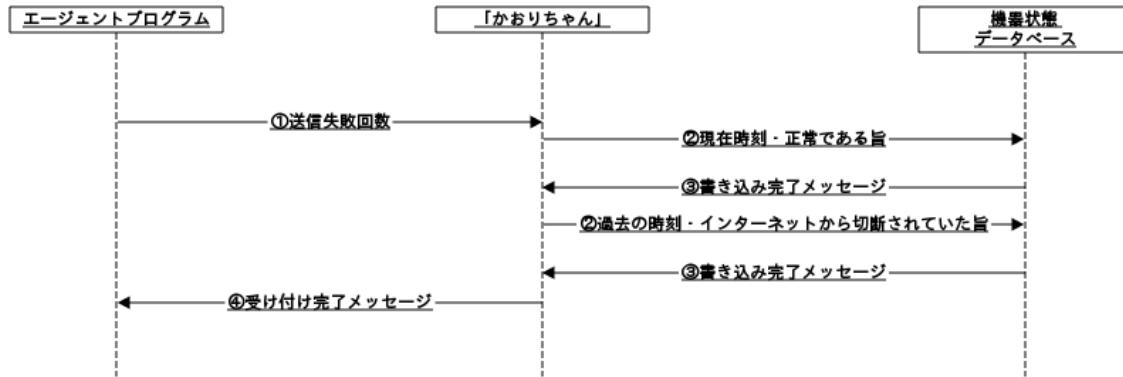


図 3.16: エージェントプログラムとエージェントプログラム用インターフェースの間のメッセージシーケンス図

HTTP 上でやり取りするデータの形式としては、Javascript Object Notation(JSON) という形式を用いる。エージェントプログラムからは、次のような形式でメッセージが送られる。

ソースコード 3.1: エージェントプログラムからエージェントプログラム用インターフェースに送られるメッセージの書式

```
1 {"seq":<NOW>, "stat":"OK", "log":{"seq":<PAST>}}
```

< NOW > には、現在の送信失敗回数が入る。< PAST > には、過去の送信失敗回数が入る。

また、エージェントプログラム用インターフェースからは、次のような形式で応答がある。

ソースコード 3.2: エージェントプログラム用インターフェースからエージェントプログラムへの応答の形式

```
1 {"stat":"OK", "time":<NOWTIME>}
```

ここで、< NOWTIME > にはサーバー側の現在時刻が入る。

3.9.8 Web アプリケーションサーバと Web アプリケーション間の通信の実装

Web アプリケーションサーバと Web アプリケーションは、インターネットを介し、HTTP というプロトコルを用いて通信する。HTTP 上でやり取りするデータの形式としては、Javascript Object Notation(JSON) という形式を用いる。

3.10 ソースコード

下記 URL にて、ソースコードを公開している。<https://github.com/miyasakakoki/devicemonit/tree/feature>

3.11 サービスによる監視のイメージ

従来は、VPN を使用して、サーバと各 IoT 機器を繋ぎ、ssh 等でログインすることで、監視を行っていた。しかし、本サービスを利用することで、サービスにログインすることで監視を行うことができるようになる。

第4章 機器監視サービスの動作テスト

4.1 技術検証

作成したシステムの動作検証を行った。

4.1.1 検証シナリオ

技術検証は、1台の IoT 機器で成り立っている IoT サービスを想定し行った。IoT 機器には、RaspberryPi を使用する。図 4.1 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。



図 4.1: IoT サービスの構成図

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 時間を行い、途中何度も IoT 機器 (RaspberryPi) の電源を抜き、正常に検知することを確認する。その後、電源が抜けていた期間について、IoT 機器の記録を確認する。

4.1.2 検証結果

IoT 機器の監視については問題なく動作した事を確認した。機器の電源を抜くと、1 分程度で表示状態が変わり、数分後、機器の電源を再び入れると、2 分程度で表示状態が戻ることを確認した。

1. 予め固有の ID とエージェントプログラムがインストールされた機器を買ってくる
2. ユーザは、その機器に対しサービスで利用するアプリケーションをインストールし、設置する
3. 機器は、設置後ネットワークに接続され次第、サービスへ、機器の状態の通知を行う

IoT サービス提供者が手軽に利用できることを確認した。

4.2 サービスの特徴

他の関し手法と比較した本サービスの特徴として、通知型で設定不要、規模に対応できることが挙げられる。
独立したサービスとしてまとまっていることが挙げられる。

4.3 考察

ユーザーテストから本システムは、ある程度有効であることが分かったが、以下の点について課題があることが分かった。

- 機器への設定について

設定ファイルの編集等の手間は削減されたが、エージェントプログラムを IoT 機器にインストールするのに手間がかかっていることが分かった。従来手法を用いて機器の監視をする際は、自動起動の設定を行う必要はなかったが、本プログラムでは必要としている。自動起動の為の設定ファイルを自動生成するか、あるいはエージェントプログラム配布の際、簡単なインストールスクリプト等と一緒に配布することで、大きな手間の削減になると感じている。

- ユーザインターフェースについて

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。

- エージェントプログラムについて

本サービスで提供しているエージェントプログラムは、その他のパッケージに依存しない単純な構成であるため、移植性が高い。

また、本来ならば、株式会社ルナネクサスにて、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5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。

第6章 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。

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 (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。

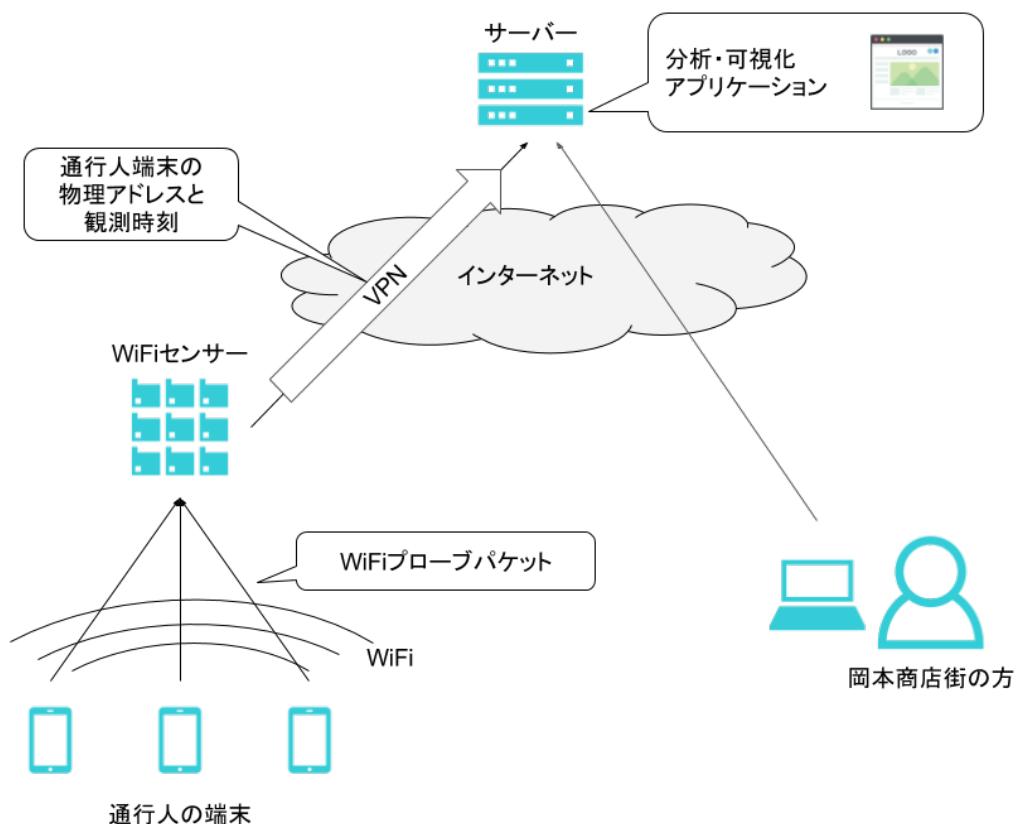


図 1: 岡本商店街人流観測 構成図



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