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 虹槻

指導教員

横山 輝明

提出日

2017**年** 1**月** 28**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

内容梗概

近年, 半導体技術の進歩により, コンピューターの小型化・低価格化が進んでいる。また, インターネット回線網の普及もあり、Internet of Things という概念が注目され、それによって収益を得る IoT サービスが登場してきた。Internet of Things(IoT) とは、様々な物がインターネットにつながり、相互に情報を交換し合うことで、様々な自動化を実現する概念である。

しかし、IoT サービスを開発・運用するには、開発コストの問題・セキュリティーの問題・稼働率の問題など様々な問題がある。

そこで、本研究では、IoT デバイスの死活監視問題に焦点を当て、IoT サービスとは独立した IoT デバイスの監視サービスを開発することにより、デバイスの故障検知に係る問題の解決を図ることにした。システムの構築に先立って、どのような機能が必要となるのか、実験し、デバイスの電源の状態 (電源が入っているのか・入っていないのか)・ネットワークの状態 (インターネットへ接続されているのかいないのか) が時系列に沿って整理されている事で、対処が決まる事が分かった。そこで、上記必要な機能を実装したシステムを提供し、協力者の理解を得て検証し評価を得た。

第1章　序論

本章では、論文の構成のと、研究の背景及び現状の課題について記述し, 本研究の目的について述べる.

1.1　研究の背景

1.1.1　IoT の登場

近年、IoT が注目を集めている。IoT とは Internet of Things の略で、全ての物がインターネットに接続し、相互に情報を交換しあうことで様々な自動化を実現しようとする概念である。「モノのインターネット」とも呼ばれる。

以前より、家電など様々な物をインターネットに繋ぎ自動化しようといった考えはあったが、近年の半導体技術の進歩によるコンピューターの小型化・低価格化と、一般へのインターネットの普及により、実現可能な考えとして注目されている。

次は IoT という言葉がまだ一般に言われていなかった時代の IoT サービスの事例である。

- Trojan Room coffee pot(1993 年)
  ケンブリッジ大学の学生が、廊下に設置されていたコーヒーメーカーを確認しに行く手間を省いたもので、IoT の源流と言える。Web カメラがコーヒーポットを撮影し、HTTP を使用して配信するものであった。(https://en.wikipedia.org/wiki/Trojan_Room_coffee_pot)

- HTCPCP(1998 年)
  ジョークとして発案された RFC である。コーヒーポットをインターネット越しに自動化するためのプロトコルを定めたもの。実際に実装してみたという例が複数ある他、お茶を入れるための拡張プロトコルも定められている。(http://www.geocities.jp/toyprog/rfc/rfc7168.txt) (https://tools.ietf.org/html/rfc2324)

- I-Pot(2001 年)
  象印マホービンが販売をしていたインターネットにつながる給湯器である。「みまもりホットライン」というサービスと連携している。ポットを使用すると、サーバーに通知が送られ、サーバー側で 1 週間の使用回数が閲覧できる他、指定したメールアドレスに対しメールを送る機能もあった。(https://seniorguide.jp/article/1001078.html)

このように以前から IoT があったことが分かる。

次は日本におけるインターネット世帯利用率の推移である。2000 年末には、約 34 %程度であったが、2001 年末には、およそ 2 倍となる約 60 %の世帯で利用されるようになり、2003 年末には約 88 %の世帯で利用されるようになっていることが分かる。(http://www2.ttcn.ne.jp/honkawa/6200.html) 次はパーソナルコンピュータの平均価格の推移である。(https://middle-edge.jp/articles/I0001147) 2000 年台では、17 万円前後していたコンピューターが 2010 年には 8.8 万円と約半分の値段になっている。また、ノート PC の出荷台数の比率から、価格以外に大きさも小型化

していることが分かる。他に 2007 年に iPhone が発売されていることから、2000 年台前半に急速に小型化した事が伺える。

これらから、近年のコンピュータの小型化・低価格化とインターネットの普及により、IoT が再び注目されるようになった事が分かる。

IoT については、2011 年のガートナーの調査によれば、５年～１０年後には、主流の考えとして採用されるであろうと予測されている。(https://www.gartner.co.jp/press/html/pr20110907-01.html) また、2015 年の情報通信白書では、「IoT による産業へのインパクト」というタイトルで IoT について述べられていることから、今後も注目を集めていく事が推測される。(http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/html/nc254100.html)

現在、IoT は注目を集めており、インターネットにつながる小型で安価なコンピュータが数多く登場してきている他、様々な IoT サービスが開発されている。次は、IoT で頻繁に使用される小型コンピュータの一例である。

- RaspberryPi
  Linux OS が動く 3000 円前後のコンピューターとして 2012 年に発売される。教育用の小型コンピュータとして開発された経緯があるが、他のハードウェアとの接続を考えた GPIO や、ネットワークへの接続するための Ethernet インターフェースがあることから、IoT 開発にて頻繁に使われている。また、より小型化を図った RaspberryPiZero や、組み込み機器への搭載を考えた ComputeModule 等、バリエーションが豊富である。(https://www.raspberrypi.org/)

- Intel Edison
  当初より、IoT を意識して作られたコンピュータ。2014 年に発売された。RaspberryPi とは違い、IoT 向けに作られているので、ディスプレイポートや、音声の出力ポートは存在しない。また、SD カード大のサイズと、とても小型である。Wifi,Bluetooth のインターフェースの他、USB インターフェース、GPIO が付いている。(http://edison-lab.jp/edison/)

- WioNode
  SeedStudio が開発した IoT デバイス。RaspberryPi につなげて使う IoT デバイス「GrovePi」の各モジュールが、使用できる。また、Wifi 経由でスマートフォンをつなぎ設定できるのも特徴の一つである。Wifi 付きのマイコンボードとしても利用できる。(https://www.seeedstudio.com/Wio-Node-p-2637.html)

次は IoT サービスの一例である。

- 萌香 (コーヒーポットの自動化) (http://alpha.mixi.co.jp/entry/2009/10723/)
- 太陽光発電の発電量の監視ある株式会社 R 社様では、太陽光発電所の発電量を監視し、配信するために IoT を使用している。太陽光発電所では、太陽光パネルで発電した電力をパワーコンディショナに繋ぎ、そこから送電線へ送電している。R 社様では、そのパワーコンディショナに R 社製機器を繋ぎ、インターネットを経由して情報を送るサービスを展開している。
- 農業の自動化
- パスの運行情報の掲示

このように、今後も IoT サービスの開発が活発になる事が予想される。

1.1.2 IoT サービスの構造

IoT サービスの基本構造は次のとおりである。IoT 機器・ネットワーク・サーバーの 3 層構造となっている。

ネットワークについては、末端のネットワークとインターネットに別れる。末端のネットワークとは、IoT 機器がつながるネットワークのことで、IoT 機器が直接インターネットへ接続されている場合は、存在しない。末端のネットワークには非 IP 網と IP 網の 2 つが有り、末端のネットワークが非 IP 網である場合、インターネットは IP 網であるため、プロトコル変換を行うゲートウェイを置く必要がある。また IP 網である場合も、インターネットと分離させるといった意味で、間に NAPT を置く。NAPT とは、ネットワークを分離する仕組みのことで、内側で発生した通信は外へ通すが、外で発生した通信は内側へは通さないという特徴がある。

IoT 機器自体に目を向けると、IoT 機器には、センサ型機器とアクチュエータ型機器、またそれらを統合したハイブリッド型機器とが有る。センサ型機器は、周囲の状態を察知し、サーバーへ通知する。アクチュエータ型機器は、サーバーから通知を受け、周囲に対し何らかの動作を行うものである。ハイブリッド型は、それら 2 つの機器をひとまとめにした物である。

サーバからの通知の受け取り方には、2 種類あり、サーバーへ定期的にアクセスし、自分宛のメッセージが無いか確認するポーリング型、サーバーから直接通知を受け取るサーバー通知型がある。それぞれ以下のような利点・欠点がある

- ポーリング型
  利点としては、インターネットに直接繋がなくても良いため、多くのネットワーク環境にて利用できる事にある。欠点としては、不要なアクセスを頻繁にする必要があるため、ネットワークに負荷をかけてしまう。

- サーバー通知型
  利点としては、不要なアクセスがおこらないことにある。欠点としては、常に待ち受けていなくてはならないので、電池持ちが悪い。また、インターネットに直接つながるため、セキュリティ面でも不安を抱える事になる。さらに、直接インターネットにつながるネットワークでないと利用出来ないことが致命的である。サーバーから通知が来るので、NAPT 環境下では採用できない。

上記理由により、ポーリング型が多く取り入れられている。

サーバーは、IoT 機器からの通知を蓄積・分析し、IoT 機器へメッセージを送付、若しくは、ユーザーへ表示したり、ユーザーからの入力を受け付けたりする。

1.2 問題

このように、IoT デバイスの開発が盛んに行われる一方で、IoT サービスの運用において、次のような問題がある。

- 数が多くて管理しきれない問題
  - 設置前の設定において、どのデバイスをどこに設置すれば良いのかわからなくなる -¿ ラベリングにて解決
  - 設置後、どのデバイスがどこに設置されたのかわからなくなる -¿ 帳簿をつけることで解決
  - 設定の際に、個別の設定をしなければならないのが面倒
    具体的には、デバイスに振る ID 等。
    ラベリングと整合性が取れていなければならない。

- 稼働状況の監視が面倒な問題

- 設置したものが正常に稼働し始めたかどうか確認するのが面倒
  設置者が、デバイスの操作を知っている必要が有る。
  また、ディスプレイ等をつけないことが多いので、別途確認する手段（ディスプレイとキーボードを持参等）を用意する必要がある。
- 設置後、正常に稼働しているのか確認するのが面倒
  NAPT の内側に設置されている事が多いので、Ping や snmp では確認できない。
  また、ネットワークの断絶等があった場合、稼働状況を確認できない。
- いつ稼働していていつ稼働していなかったのか管理するのが大変
  いつ稼働していていつ稼働していなかったのかがわからないと、データを正確に分析する事が出来ない。

稼働状況の監視については、IoT サービスで行うことがある程度可能だが、サービス自体に手を加える必要があるため、開発のコストが高くなる。その中で、私は、IoT デバイスの状態監視に着目した。

1.3 研究の目的

そこで、IoT サービスとは独立した IoT デバイスの監視サービスを開発することにより、これらの問題を解決できるのではないかと考えた。本研究では, IoT デバイスの監視サービスを開発することで、IoT デバイスの状態監視を簡単化することを目的とする。

1.4 本論文の構成

本論文では、第一章にて研究の背景及び現状の課題、本研究の目的について述べる。第二章では、既存の解決策とその課題を分析し、第三章にて、サービスの開発を提案し、そのための要求の分析を行い、第四章では、サービスの構成と機能、ユーザーの動きについて述べる。第五章では、検証として行った実験と考察、第六章では、成果と今後の課題について述べる。

第2章　既存の解決策とその課題

序論で述べたとおり、本研究で解決する問題は以下の３つである。

- 設置したものが正常に稼働し始めたかどうか確認するのが面倒
  設置者が、デバイスの操作を知っている必要が有る。
  また、ディスプレイ等をつけないことが多いので、別途確認する手段（ディスプレイとキーボードを持参等）を用意する必要がある。
- 設置後、正常に稼働しているのか確認するのが面倒
  NAPT の内側に設置されている事が多いので、Ping や snmp では確認できない。
  また、ネットワークの断絶等があった場合、稼働状況を確認できない。
- いつ稼働していていつ稼働していなかったのか管理するのが大変
  いつ稼働していていつ稼働していなかったのかがわからないと、データを正確に分析する事が出来ない。

2.1　IoT サービスの構造の分析

ここで、対象となる IoT サービスの構造について説明する。IoT サービスは、IoT デバイス（機器)、ネットワーク、サーバーの 3 層から成り立っている。IoT デバイスは、センシングしたデータを送り出す、または、送られてきたデータを元に何か動作を行うものである。ネットワークは、末端のネットワークと、インターネットとに別れる。末端のネットワークとインターネットとの出口には、NAPT が置かれていることが多い。サーバーは、IoT デバイスから送られてきたデータを蓄積・分析し、表示または、IoT デバイスへ通知する。

以下に状態監視のシステムを導入？しない場合の解決策を上げる

2.2　デバイス設置箇所に行って、直接確認する

直接設置箇所に行き、確認を行うという方法である。この場合、ディスプレイとキーボードを持参するなどする必要がある。そして、デバイスを良く知る者を行かせる必要もある。設置台数が多いことや、設置個所が離れていることから、あまり現実的ではない。

2.3　ICMP Ping を活用する

ICMP とは、InternetControlMessageProtocol の略であり、IP パケットの送り元から送り先への間で起きた問題を通知する役割を持つ。ICMP には、ICMP echo request と、ICMP echo repry が定義されており、ICMP echo request を受け取った機器は、ICMP echo repry を返送しなければならない。Ping は ICMP echo を送信する為のプログラムで、IP 網のトラブルの発見の他、特定の IP アドレスを持つ機器が稼働しているかどうか確認するためにも使われている。Ping を使用して、IoT

デバイスの稼働確認を行うというのがこの解決策である。しかし、ICMP パケットは、セキュリティの都合上、ネットワーク機器で転送しないよう設定されている場合が多い。また、一般的なネットワークでは、インターネットとの接続点にてネットワークを分離している事がある。その場合、IoT デバイスの IP アドレスは、IoT デバイスの所属するネットワークでのみ通用するアドレスとなるため、外部から ICMP パケットを送ることは出来ない。

2.4 SNMP を利用する

SNMP とは、SimpleNetworkManagementProtocol の略で、ネットワークに接続された機器を監視・制御するために作られている。CPU の状態や起動してからの時間、メモリ使用量等を取得できるが、設定がめんどくさい。また、制御もできるため、ネットワーク機器にてブロックされてしまう恐れがある。

しかし、これらの手法では、解決に至っていない。そこで、通常（？）は、次のような方法で解決を図っている。

2.5 Zabbix を使用する

2.6 Fluentd Elasticksearch Kibana を利用する

Fluientd とは、ログの転送を目的としたアプリケーション。ログを転送する他、場合によってはバッファリング等も行う。Elasticksearch とは、データベースの一つ。Kibana とは、可視化ツールの一つで、グラフの描画などが行える。しかし、それぞれが「そのための」プログラムなので、それらを使って可視化までこぎつけるには、それぞれに対して設定が必要と鳴る。また、Fluentd が動くデバイスである必要がある。更に、Kibana は可視化の為のシステムなので、一つのデバイスの状態を可視化するだけでも、ダッシュボードの作成からパネルの作成までする必要が有り、多数のデバイスの状態を見るためには、それを複数回繰り返す必要がある。

2.7 Telegraf Influxdb Grafana を利用する

Influxdb とは時系列データベースの一つで、データを時系列に沿って整理し格納する。Telegraf とは Influxdb へメモリ使用量等を格納するためのエージェントプログラムである。Grafana とは可視化ツールの一つで、グラフの描画等を行う事ができる。これも、FluentdElastickSearchKibana の時のように、それぞれに対して設定が必要で、デバイスの上で Telegraf が動く必要がある。また、Grafana についても、Kibana と同様で、多数のデバイスを監視しようとするととても手間がかかる。

2.8 SORACOM Air を利用する

IoT 向けネットワークとして SORACOMAir というサービスがある。SORACOMAir は携帯電話回線を使用し、IoT 向けのネットワークを提供している。携帯電話回線を使用するため、SIM カードを用いており、SIM カードの ID により、個体の識別と通信状況の確認が行える。しかし、IoT デバイスに SIM カードリーダ（？）等を接続する必要が有る。また、SORACOMAir では、インターネットとの接続に NAPT を使用しているため、外から状態を確認することは困難である。

2.9 VPN を利用する

VPN とは、VirtualPrivateNetwork の略で、インターネット越しに仮想的で閉鎖的なネットワークを構築するものである。しかし、VPN を利用するには、VPN クライアントが使用できる必要がある。また、仮想的なネットワークの IP アドレス帯と IoT 機器がつながっているネットワークの IP アドレス帯が被らないように調節する必要があるため、IoT 機器がつながるネットワークの構成を全て把握している必要がある。(IP アドレス帯がかぶると、どちらの IoT 機器がどちらのインターフェースにパケットを送り出すべきなのかわからなくなるため、調節が必要) IoT 機器の数を考えると、ネットワークの構成を把握することは困難である。

第3章　提案する解決策

そこで、新たにIoT機器の稼動状態を監視することに特化したシステムを開発・提供することで、問題が解決できるのではないかと考えた。まず、システムの要件を抽出するために、以下のような実験を行った。

3.1　岡本商店街での事例

岡本商店街とは、兵庫県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事である。目的は、岡本商店街内を往来する人の流れを観測し、商店街の活性化につなげる事であったが、有用な知見（？）が得られたので、事例として上げることにする。

3.1.1　実験概要

スマートフォンのWifi接続機能をONにした時、スマートフォンから定期的にプローブパケットというものが、送信される。そのプローブパケットを観測するためのソフトウェアとして、ampsenceがある。本実験では、そのampsenceとRaspberryPiを用いてプローブパケットを観測し、Webアプリケーションによって可視化することを行った。RaspberryPiは、岡本商店街内にある商店に設置させてもらい、インターネットとの接続は、商店に敷設済みのWifiネットワークを使用させてもらう他、1台のみSORACOM Airを使用した。SORACOM Airとは、IoT向けデータ通信を提供するサービスで、携帯電話網を用いている。そのため、RaspberryPiに、SIMモジュールとSIMカードを接続しなければならなかった。

3.1.2　課題と考察

その実験の際、次のような問題があった。

- 機器が稼働していないことがあった
  コンセントが抜け、機器が稼働していないことがあった。
  そのため、分析する段階において、機器からログを回収し、全ての機器が稼働している時間帯を選び、分析する必要があった。
  また、そのログのタイムスタンプが、UnixTimeであったため、可読性が低く時間がかかった。
  機器の稼働を監視していなかったので、機器が復旧するまでに必要以上に時間がかかり、分析の際、必要なデータが集まらないため、分析を諦めたものもあった。

- 迅速に稼働状態の確認が行えない
  RaspberryPiを使用したセンサーであったため、ディスプレイやキーボードはついておらず、現地に行って確認するためにはディスプレイやキーボードを持っていく必要があった。
  また、現地に行かなくてはならないのでとても手間であった。
  そのため、遠隔からアクセスすることにした。

- 遠隔からのアクセスができない問題
  しかし、遠隔からのプログラムの修正、稼働の監視を行う際、商店のネットワークと SORACOM Air のネットワークにはインターネットとの間に NAPT が入っており、外からアクセス不能だった。
  VPN を利用しアクセスするが、稼働の確認毎に VPN サーバーにアクセスしてからの確認となるため、VPN に関する知識が必要だった。
  また、簡単に稼働を確認できなかった。

- 機器の回収に手間がかかった
  機器を回収する際、保存していたデータを破損しないために、正常にシャットダウンさせる必要があった。
  そのため、回収作業の際もディスプレイとキーボードを持参し正常にシャットダウンさせてからコンセントを抜く等、手間がかかった。

上記より、機器の稼動状態を確認する事はとても重要であることが分かった。

3.2 学内での wifi を用いた人流観測

学内にて、wifi を用いた人流観測を行った。構成としては、先の事例での構成とほぼ同じものを学内でも行った。先の事例から得た問題点を踏まえて行った。

3.2.1 課題と考察

その際にも次のような問題があった。

- 分析の為に、機器の状態を常に監視している必要があった
  岡本商店街での実験を踏まえ、機器の状態を定期的に監視し、機器トラブルに備えた。

- 機器がネットワークに接続できていない事があった
  機器のトラブルにより、機器がネットワークに接続できていない事があった。
  機器の様子を見に行った所、電源が入っているのに繋がっていないというトラブルがあった。

3.3 要件の抽出

上記の事例・実験から、以下のような機能が必要となることが分かった。

- IoT 機器の稼働状態がわかる
  IoT 機器の稼動状態は、稼働している・稼働していない・ネットワークに接続されていないの３つ必要である。また、一覧で閲覧することが可能である事が望ましい。

- IoT 機器の稼働状態の記録を閲覧することができる
  データの分析を行う際に、それら稼働状況の記録が必要になる事が分かった。
  また、それらの記録が時刻と共に、整理されている必要があることも分かった。

- IoT 機器の停止・再起動を行うことができる
  保存していたデータを破損しないために、IoT 機器を正常にシャットダウンさせる必要があり、その手順のために IoT 機器を回収することが手間であることが分かった。

そこで、IoT サービスとは独立した、IoT デバイスの稼動状態を監視・管理することを簡単にするサービスを開発し、提供すれば良いのではないかと考えた。

第4章　設計と実装

実験から抽出した要件に基づいて、システムを作成した。要件は以下のとおりである

- IoT 機器の稼動状態が確認できる
  機器の状態については、機器が稼働していない・機器が稼働している・機器が稼働しているがネットワークに繋がっていないの３つ。
- IoT 機器の稼動状態の記録を閲覧することができる
  IoT 機器の稼動状態の記録が時系列に整理され、確認できることが必要である。
- IoT 機器の停止、再起動を行うことができる
  回収に備えて、機器の停止、再起動を行う事ができると良い。

4.1　想定するユーザーの定義

ユーザーは、IoT サービスを構築しようとしている SI さんと定義する。

4.2　課題と機能の関係

本システムは各課題に対し、以下のような機能を用いて解決する。

- IoT 機器の稼動状態が簡単に確認できない
  直近の IoT 機器から通知をすることにより、直近の IoT 機器の状態を取得し、Web 画面にて一覧表示することにより解決する。
- IoT 機器の稼動状態の記録を整理するのに困る
  IoT 機器から定期的に通知をもらい、それをサーバー側で整理してから格納し、ユーザーが見やすいよう Web 画面で表示することで解決する。
- IoT 機器の停止、再起動がしにくい
  IoT 機器からの通知の返答として、シャットダウンや再起動と言ったメッセージを渡すことで、機器の停止、再起動を可能にする。また、Web 画面からこれらを行えるよう工夫した。

4.3　システムの構成

システムの構成は以下のとおりである。本システムは、エージェント、サーバーの２つの要素で成り立っている。エージェントは IoT デバイスにインストールされ、サーバーに対し定期的に通知を送る。また、サーバーは、エージェントからの通知を記録する。サーバーはユーザーへのインターフェースも提供しており、ユーザーはブラウザを使用して本システムを利用する。

- エージェントについて

- インストールされる機器
  RaspberryPi(Raspbian jessie)・Intel Edison(yocto linux) にインストールされる。
- 機能
  システムに対し、現在の状態を定期的に通知する。
- 構成要素
  シェルスクリプト一つ（agent.sh)

- サーバーについて
  - 環境
    Ubuntu16.04 (xenial)
  - インストールしたもの
    Python3 Influxdb
  - 構成要素とその説明
    * API サーバ
      · 使用したライブラリ
        Falcon
      · 機能
        エージェントから送られてきたデータを展開しログ用データベースに格納する。
      · ソースコード

    * ログ用データベース
      · 使用したもの
        Influxdb
      · 機能と説明
        ログを蓄積する。他にもデータベースとしての選択肢があったが、時系列に整理され、検索が早いらしいことから採用した。
    * デバイス情報用データベース
      · 使用したもの
        sqlite3
      · 機能と説明
        デバイスに関する情報を記録する。Python から使いやすかったので採用した。
    * Web アプリケーション
      · 使用したライブラリ
        Flask Bootstrap JQuery
      · 機能
        ユーザーからの操作を受け付け、データベースに反映する。また、必要な情報をデータベースから取得し表示する。
      · ソースコード

4.4　構成部品のそれぞれの動き

4.4.1　監視エージェントの動き

監視対象となる IoT 機器には、開発した監視エージェントが入っているものとし、起動時に自動でエージェントプログラムを起動するよう設定されているものとする。ログファイルの中には、過去のシーケンス番号が入っている。監視エージェントは、起動後、ログファイルがあるか確認し、あった場合、ログファイルから過去のシーケンス番号を読み出す。無かった場合、過去のシーケンス番号を 0 にする。その後、現在のシーケンス番号を 0 にする。監視エージェントは、監視サービスに対し過去のシーケンス番号と現在のシーケンス番号、自身が現在正常である旨のメッセージを送信する。監視サービスから返答があった場合、受理されたとみなし、現在のシーケンス番号を 0 にする。その後、過去のシーケンス番号とログファイルを削除する。また、返答に停止・再起動のコマンドが含まれていた場合、そのコマンドを実行し終了する。返答が無かった場合、ネットワークに障害があったとみなし、シーケンス番号をインクリメントし、ログファイルに保存する。この動きを約 1 分ごとに繰り返す。

4.4.2　サービスの動き

監視エージェントからメッセージを受け取った時の動き

監視エージェントからメッセージを受け取った場合、DB の末尾がコマンドを示すものであった場合、そのコマンドを変数に記録しておく。過去のシーケンス番号の数だけ、接続されていなかった旨を、最後に通信があった時刻から DB に格納し、現在のシーケンス番号の数だけ、接続されていなかった旨を、現在の時刻からさかのぼって DB に格納する。また、現在の時刻にて機器の状態が正常であった旨を DB に格納する。最後に、コマンドが格納されている変数の内容と、受理した旨のメッセージを監視エージェントに対し送信する。

ブラウザから入力を受けた場合の動き

まず、クッキーからセッションを読みだし、ログインしていないユーザーであった場合、ログインページへと誘導する。

- 現在の機器の状態の確認を求められた場合
  サービスは、DB から、機器の現在の状態とその他の機器情報を取得し、返却する。
- 機器情報の追加・変更・削除を求められた場合
  サービスは、DB へ変更を保存する。
- ログ一覧を求められた場合
  サービスは、DB から機器の状態を取得・整理し、返却する

4.5　ユーザーの動き？

ユーザーの動きは以下のとおりである。

- デバイスを追加するとき
  1. ブラウザからサービスにアクセスし、ログインする。

2. 画面から追加ボタン（＋ボタン）をクリックし、デバイス名、デバイスの説明を入力する。
   この際、デバイス ID がすでに決まっているのであれば、それも入力する。
3. デバイス ID をメモする。（既に決まっているのであれば、特段シなくても良い）
4. Add ボタンを押す
5. デバイスに対し、エージェントプログラムをインストールし、自動で起動するよう設定する。
   その際、エージェントへの引数として、サーバーの IP アドレス、デバイス ID を設定する。

- デバイスを削除するとき

1. ブラウザからサービスにアクセスし、ログインする。
2. 画面から該当のデバイスをクリックし、削除ボタンをクリックする。
3. 「ほんとに削除するの？」というダイアログが出るので、OK を押し、画面からデバイスが削除されたことを確認する。

- 登録されているデバイス情報を変更するとき

1. ブラウザからサービスにアクセスし、ログインする。
2. 画面から該当のデバイスをクリックし、変更ボタンを押す。
3. デバイス作成時と同じようなダイアログが表示されるので、デバイス名やデバイス情報を編集する。
4. OK ボタンをクリックし、デバイスの情報が変わったことを確認する。

- デバイスの現在の状態を確認するとき

1. ブラウザからサービスにアクセスし、ログインする。
2. 該当のデバイスを確認する。
   緑が稼働している状態、赤が稼働していない若しくは、稼働しているかわからない状態である。

- デバイスの過去の状態を確認するとき

1. ブラウザからサービスにアクセスし、ログインする。
2. Log ページボタンを押す。
3. デバイスのログ一覧が出るので、該当デバイスを探しだし、確認する。

第5章　検証と考察

作成したシステムを検証するために以下のような実験を行った。

5.1　実験概要

学内にて Wifi プローブパケットを利用した人流観測の実験を再現する

5.1.1　実験目的

作成したシステムが IoT サービスの開発にどう影響するのか観測する。

5.1.2　実験方法

第３者に Wifi プローブパケットを利用した人流観測のサービスを開発してもらい、その様子を観測する。また、観測後にインタビューを行い、システムを利用した場合としない場合の違いについて聞く。

5.2　準備

RaspberryPi,Wifi ドングル, プローブパケット観測ソフトウェア, 監視用エージェントを被観測者に提供する。被観測者は、それらを用いて、学内の各階に誰が居るのか、また、人の移動の様子を可視化するサービスを構築してもらう。

5.3　経過

5.4　考察

実験により、次のような評価を得ることができた。
評価から、有効であると分かった。

第6章　結論

本研究の背景には、IoT サービスにおける機器監視の煩わしさが挙げられる。具体的には、

1. IoT サービスに組み込む形で機器監視を行った場合、IoT サービス毎に監視部分を開発するので、コストが高い
2. 設置箇所が遠く、分散して設置されるため、定期的に確認しに行くことは現実的ではない

が挙げられる。

そこで、機器監視をサービスから独立させ、IoT 機器監視サービスとして提供すれば良いのではないかと考え、開発を行った。しかし、既存手法を調査する中で、IoT 機器は NAPT 環境下に置かれる事があることがわかり、IoT 機器から定期的に通知を送ることで解決を図った。
プロトタイプの開発の結果、下記のような知見を得ることが出来た。

- NAPT 環境下にある機器の状態を監視するために、機器から状態を定期的に通知する手法は有効であることが分かった。
- また、既存の機器監視サービスの多くは、機器の IP アドレスをサーバー側で管理しなければならないが、本手法を用いることで、IP アドレスに関係なく監視できることが分かった。

また、今後の課題としては、次のような事があることが分かった。

- ユーザビリティーの向上
  現状では、デバイス ID やサーバー IP アドレスを IoT 機器に打ち込まなければならず、また、自動起動の設定も行わなければならない。
  シェリスクリプトを実行すれば自動起動の設定まで行われる等、いっそうの簡略化を行う必要がある。
- デバイスの識別問題の解決
  現状では、どのデバイスが何だったのかまでは管理できていない。
  QR コードを出力し、デバイスに貼り付けるなどによって、簡略化できると思われる。
- アラート機能の実装
  アラートメールの送信などもできれば、常に監視していなくて良くなるのではないかと思われる。

これらの課題を解決することで、IoT サービスの開発の手間を省けるのではないかと考える。

第7章　謝辞

第8章　参考文献

関連図書

[1] 平成 27 年版 情報通信白書 (総務省) 「IoT の実現に向けたアプローチと我が国 ICT 産業の方向性」より http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/html/nc254150.html

[2] 6 割が「IoT は流行語」–エスキュービズム調査 (ZDBet Japan) http://japan.zdnet.com/article/35093272/

[3] 先進テクノロジのハイプ・サイクル２０１１年 Gartner https://www.gartner.co.jp/press/html/pr20110907-01.html