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、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

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, 家庭へのインターネットの普及により, 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合い様々な自動化を実現する IoT が注目されている.

それを意識して、IoT デバイスとして簡単に使用することができるコンピューターが登場している。

- 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。

- 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。

より IoT に特化したものとして WioNode という物がある。
以下のようにそれらを使った IoT サービスの開発も盛んに行われている。

- 太陽光発電の発電量の監視
- 農業の自動化
- パスの運行情報の掲示

ここで、IoT サービスとは、IoT による自動化を提供する物のうち、デバイスから得た情報を蓄積・分析し、結果を元に、表示等の動作を行うものと定義する。IoT サービスの構成としては、複数のデバイスから１つのコンピューターへ情報を送り、その上で上で蓄積・解析し、結果を表示する等の動作を行っているものが多い。

1.2　問題

このように、IoT デバイスの価格が下がることで、IoT サービスの開発にかかるコストが低減され、開発への垣根が下がる一方で、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。

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してくれます。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す。Grafana とは可視化ツールの一つで、グラフの描画等が行えます。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 が動く必要が有ります。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。

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 デバイスの稼働状態がわかる
  IoT デバイスの稼動状態は、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。

- IoT デバイス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 デバイスの停止・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 デバイス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 デバイス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

- 機器の現在の状態を一覧して確認することができる
  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。
- 機器の過去の状態を時刻と共に確認・取得することができる
  各機器の状態について、いつからいつまで稼働していたのか、容易に確認することができる必要がある。
- 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　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

4.2 構成部品のそれぞれの動き

4.2.1 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。

4.2.2　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

4.3　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。

4.4　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

よって、このアプローチは有効であるということが分かった。
今後の課題としては、次のような物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。

第7章　謝辞

第8章　参考文献