卒業プロジェクト　2014年度（平成26年度）

リアルタイム情報の効率的閲覧システムの研究

慶應義塾大学 環境情報学部

中山　拓哉

増井俊之研究会

2015年1月

卒業プロジェクト　2014 年度（平成 26 年度）

リアルタイム情報の効率的閲覧システムの研究

論文要旨

天気予報や電車の運行情報など様々な情報をを Android Widget を用いて表示することは広く用いられている手法だが、ユーザが望むあらゆる情報に対して Android Widget が存在するとは限らない。しかも、既存の Android Widget を使用する場合、画面領域の制約があるため表示できる情報の数が限られてしまう。

そこで本研究ではユーザが必要とする複数の情報をコンパクトに表示することが出来る Android Widget を作成し、ユーザが効率良く価値の高い情報を得られることを目指した。

キーワード

Linda、Android、Android Widget

慶應義塾大学 環境情報学部

中山　拓哉

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

近年、スマートフォンなどのモバイルインターネット端末が爆発的に普及し、あらゆる年代の利用者に用いられるようになった。もはや生活の一部とも言える存在となったスマートフォンだが、私が日常的に使用する中で不便だと感じる点がある。それは Android Widget を用いた情報の獲得についてである。Android Widget を画面に配置しておくと、ユーザが能動的に操作をしなくとも、最新の情報が常に画面に表示されるので情報を効率的に入手することが出来る。

しかし、Android Widget はいくつかの問題を抱えている。まず Android 端末の一つの画面に置くことの出来る Android Widget の数は限られているため、獲得できる情報の種類も限られてしまうという問題である。(図 1.1) 例えば天気予報を見たければ天気予報の Android Widget を画面に設置し、別の情報が必要であればまた別の Android Widget を設置する必要がある。Android Widget の性質上、取得したい情報に付随して不要な情報までもが表示され、それによって画面スペースが埋められてしまうというケースも存在する。また、自分の取得したい情報に対応した Android Widget が存在しない場合、自ら Android Widget を作成する以外の対応策がない。現状では Android Widget の作成は多数のスマートフォンユーザにとって技術的なハードルが高く、問題の解決法としては現実的ではない。

本研究では単体で複数の情報を表示することの出来る Android Widget を作成することでこれらの問題の解決を図った。

1.2　目的

本研究の目的は、ユーザが必要とする様々な情報をコンパクトに表示するシステム及びインタフェースの実現である。それによって情報の獲得が容易になり、さらに取得する情報からノイズが減ることになり、ユーザは効率良く価値の高い情報を得ることが出来るようになる。

1.3　本論文の構成

第 2 章では研究内容を説明する。第 3 章ではプロトタイプの実装方法を解説する。第 4 章では考察を書く。最後に第 5 章にて結論を書き本論文をしめることとする。添付として参考文献を追記する。

図 1.1: 現状の Android Widget を使用した画面

第2章　研究内容

本章では、本研究の内容を説明する。

2.1　システム概要

本システムの目的はユーザが必要とする複数の情報を提供することで、ユーザが効率良く価値の高い情報を得られるようにすることである。

以下にシステムの構成を示す。

表 2.1: システム構成

システム	概要
Linda[2]	取得したデータの保存先。
JSON 生成サーバ	Android クライアントからのリクエストがに応じ Linda からデータを取得し Android クライアントに送信する。外部から取得したデータを Linda に書き込む。
WEB クライアント	Android クライアントに送信する情報を選択する為のビューを提供する。
Android クライアント	サーバにデータを要求し、ユーザに情報を表示する。
データベース	サーバのデータを保存している。

取得したデータの保存先として、データをクラウド上で共有し並列処理で同時に多くのクライアントを処理できるフレームワークである Linda を利用した。

サーバは各種 API などを用いてユーザが必要とする情報を取得し Linda へと書き込み、Android クライアントからのリクエストに応じて Linda からデータを取得し JSON 形式でレスポンスを返す。Android クライアントはユーザの操作に応じてサーバにリクエストを送り、サーバから返された JSON をパースして表示する。

2.1.1　データの取得と記録

本システムでは三種類の方法でデータを取得している。

1. Web API による情報取得

   まず一つ目として既存の API を用いて取得する方法である。天気予報や株価の情報など比較的ポピュラーな情報であれば既に API が存在するので、それらを用いて定期的にサーバがデータを取得し Linda へ書き込む。(図 2.1) の例では Yahoo!の提供する API である YQL[1] を用いて取得した天気予報を Android Widget として表示している。

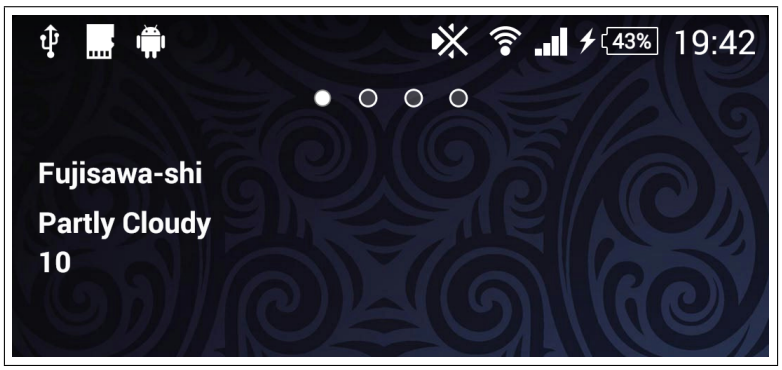

図 2.1: 天気予報を Andoid Widget として表示

2. スクレイピングによる情報取得

   二つ目は API が存在しない Web ページ上の情報である。例として自分の利用する公共交通機関の運行情報や特定の商品の在庫情報などである。これらのデータは Kimono[2] というスクレイピングツールを用いて取得したい部分に対応する API を作成しデータを取得する。(図 2.2) の例では小田急電鉄の Web ページ[3](図 2.3) にて公開されている列車の運行情報を Kimono を用いて API 化して取得し、Android Widget として表示している。

図 2.2: 小田急電鉄の運行情報を表示

[1] https://developer.yahoo.com/yql/

[2] https://www.kimonolabs.com/
指定した Web ページをスクレイピングし、API 化するサービス。

[3] http://www.odakyu.jp/cgi-bin/user/emg/emergency_bbs.pl

図 2.3: 小田急電鉄の運行情報を公開している Web ページ

3. センサーによる情報取得

   三つ目はセンシングデータである。自室などに設置した各種センサーからのデータに関してはセンシングする機器から直接 Linda に書き込むか、もしくはセンシングする機器からサーバ経由で Linda に書き込む。(図 2.4) は Arduino と明るさセンサー、温度センサーを用いて Linda にそれぞれの値を書き込む為の機構である。(図 2.5) ではこの機構を用いて取得したデータを表示している。

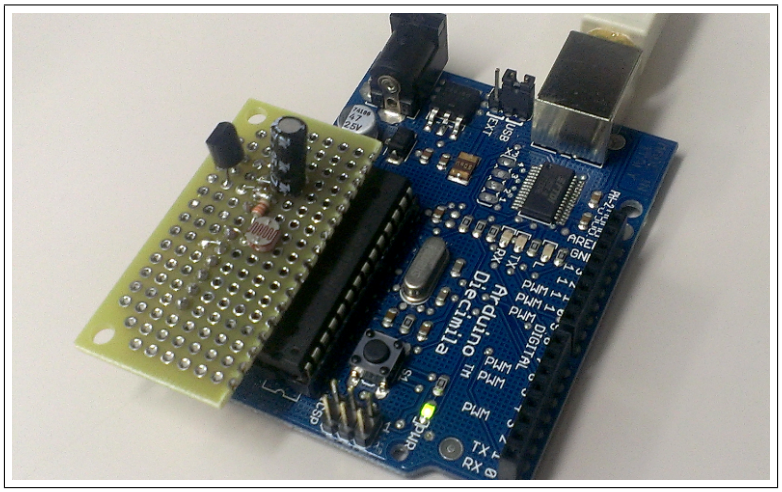

図 2.4: 明るさと温度を取得するセンサーと Arduino

2.1.2　ユーザが受け取るデータの選択

ユーザは受け取りたい情報を Web ページ上から指定する。Linda サーバに存在するデータの中から受け取りたい情報に対応した TupleType と TupleName のペアを指定する。(図 2.6) このペアはデータベースに保存され、JSON を生成する際に参照される。

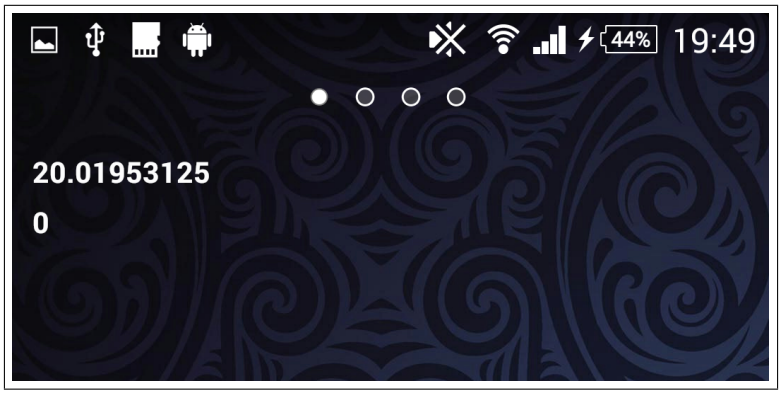

図 2.5: センサーデータを Widget として表示

図 2.6: Android クライアントに送信するデータの選択画面

2.1.3 ユーザへの情報の提供

ユーザが選択した情報はサーバによって取得されJSON形式でAndroid端末へと送られる。(図2.7)Androidは受け取ったJSONをパースしAndroid Widgetとして表示する。(図2.8)

サーバへのリクエストは15分間隔のAndroid Widgetの自動更新の際と、ユーザがAndroid Widgetをタップした際に実行される。

```
{
    "info": [
        {
            "0": "2015年01月16日14時43分現在",
            "1": "小田急線は平常どおり運転しております。",
            "2": "We are out of inventory. Please check back soon.",
            "3": "Fujisawa-shi",
            "4": "Mostly Cloudy"
        }
    ]
}
```

図 2.7: サーバから送られるJSONの例

図 2.8: Android Widgetとして表示する例

第3章　実装

この章では本研究でのプロトタイプを実装し、その流れを解説する。

3.1　サーバサイド

実装に際して Linda とのデータのやり取りが多いため、サーバサイドは Node.js[1]を用いて実装した。またデータベースには mongoDB[2]を採用している。ユーザに提供するデータの格納先として Linda を利用し、データの取得及び書き込みを行なっている。言語は Javascript を使い、Node.js の socket.io[3]で実装した。mongoDB にはユーザが指定した TupleType と TupleName のペアを保存している。Linda サーバに対して socket.io を用いて常にコネクションを貼り、Android クライアントからのリクエストに応じて、mongoDB に保存されている TupleType と TupleName のペアを元にデータを取得している。

各種 API へのリクエストとそれによって取得したデータの Linda への書き込みを 5 秒ごとに実行し Linda 上の情報を最新の状態に保っている。

3.2　Android クライアントサイド

3.2.1　データの取得

言語は Java を使い、サーバへのリクエストには JSON 形式のデータの取得を容易に行うために Volley[4]を使用した。

また Android Widget の自動更新は最短でも 15 分間隔のため、取得したいデータの種類によってはもっと新しい情報が欲しいという状況が想定される。そのため表示内容の手動更新機能をつけた。表示内容を手動更新する操作には Android Widget のタップを採用することで、情報を表示するスペースを圧迫することを回避した。

[1] https://nodejs.org/
[2] http://www.mongodb.org/
[3] http://socket.io/
[4] Android の HTTP 通信用ライブラリ。ネットワーク通信処理やキャッシュ処理を簡単に行うことが出来る。

3.2.2　データの表示

サーバから受け取った JSON は一要素ずつ Android Widget 上の textview に挿入されテキストとして表示される。表示する要素の行数に応じて Android Widget のサイズを変更可能にすることで、必要以上に画面スペースを占有してしまう事を防いでいる。

第4章　考察

4.1　本研究についての考察

今回作成したシステムを用いることで現状の Android Widget の問題をいくつか解決することができた。多くの Android Widget を置くことで画面スペースを占有してしまう問題に対しては、情報をテキストとして並べることで省スペースを実現した。必要な情報に付随して不要な情報も表示されるという問題があったが、ユーザ自身が表示される情報を指定するため、不要な情報は表示されなくなった。自分の取得したい情報に対応する Android Widget が存在しない場合、自ら Android Widget を作成するしか対応策が無かったが、Web ページ上のテキストデータやセンシングデータなどを取得可能になった。これらの面からユーザに効率良く価値の高い情報を提供するという目的は達成できた。

しかし、今回作成したシステムを実際に利用した結果として以下のような問題点が発見された。まず、テキスト形式のデータしか扱う事ができないが、取得したい情報がテキスト形式のデータとは限らないという点である。(図 4.1) は葉山の風速情報であるが、テキスト形式のデータではなく風速計をライブカメラで撮影した映像によって配信している。今回作成したシステムではこういった画像や映像で配信される情報を取得することができない。

次に、取得したデータの表示方法に課題が残る。今回作成したようなプレーンなテキストだけを、ただ並べるだけのインタフェースが最善とは言えない。見やすさや分かりやすさを意識し、インタフェースを改善することが今後の課題として見つかった。

取得する情報を選択するインタフェースとして今回は TupleType と TupleName のペアをユーザの手入力によって指定する方法をとったが、この方法だと TupleType と TupleName のペアをユーザが覚えておく必要があり Linda サーバで管理しているデータの種類が多くなると管理しきれなくなってしまうことが想定される。また、取得したいデータが多くなれば、ペアを手作業で入力していく行為はユーザにとって負担になると考えられる。

4.2　今後の展望

考察で述べた三つの問題に対する解決法を提案する。

一つ目にテキスト形式のデータしか扱う事ができないという問題である。この問題に対しては Android クライアントが画像形式のデータを表示する仕様に変更する事で対応出来ると考えられる。サーバから送られる JSON に画像 URL が含まれていれば、それを展開して表示する仕様を実装することで解決できるだろう。


リビエラリゾート
貴方のマリンライフをトータルにサポートします。
リビエラリゾート
ホーム
ライブカメラ
風向風速計
３マリーナ 風向風速計
3marina
数字は瞬間風速（m/S)を表わし、単位（m/s）は秒速何メーターの風が吹いているいうことを示しています。
数字の周りには方位計があり、Nであれば北からの風が吹いているということです。
したがって、冬はN方向の風が（陸風）多く、波高もそれほど高くなりません。
一方、夏はS方向の風が（海風）多くなり、波高も高くなります。
リビエラ逗子マリーナ
シーボニアマリーナ
N
NW
NE
W
E
SW
SE
S
m/s
Seabornia

図 4.1: 葉山の風速情報

二つ目にテキストを並べて表示しているだけなので、情報の見やすさや分かりやすさが高いレベルで実現されていないという問題である。この問題に対しては、センサーから取得した数値データなどであれば閾値を設定し、閾値を超えれば通知領域に通知を出したり表示色を変更することなどで値の変化がわかりやすくなるだろう。数値ではなくテキストデータであっても、取得したデータの履歴を保存しておき、それと照らし合わせ平常時と違ったデータが検出された場合に通知をしたり、ユーザが任意にキーワードを設定し、そのキーワードが検出された場合に通知することなどで解決できるだろう。

最後に取得する情報を選択するインタフェースの問題を解決するためにはLindaサーバ上にある情報を予めリスト形式で表示しておくことが有効であるのではないか。情報のリストからユーザが選択する方法にすることで、ユーザはペアを記憶しておく必要がなくなり、更に入力の手間も省くことが出来る。

第5章　結論

本研究では、ユーザが必要とする複数の情報を表示するシステム及びインタフェースの実現を試みた。結果として必要な情報に付随して不必要な情報が表示される問題や、多くの Android Widget を置くことで画面スペースを占有してしまう問題、自分の取得したい情報に対応する Android Widget が存在しない場合、自ら Android Widget を作成するしか対応策が無い、といった問題を解決することができた。

そして、第 4 章で述べたようにいくつかの課題が見つかった。さらに効率のよい情報の獲得を目指し、これらの課題の解決に今後も取り組んでいきたい。

謝辞

本研究を進めるにあたり、数々のアドバイスを頂いた増井俊之教授に深く感謝申し上げます。
また、研究会のメンバーにも多くの支援を頂きました。ありがとうございます。

中山　拓哉

参考文献

[1] Android Developers. http://developer.android.com/.

[2] D. GALENTER. Generative communication in linda. *ACM Trans. Program. Lang. Syst.*, Vol. 7, No. 1, pp. 80–112, 1985.

[3] java. https://www.java.com/ja/.

[4] mongoDB. http://www.mongodb.org/.

[5] node.js. http://www.nodejs.org/.

[6] socket.io. http://socket.io/.

[7] 椎尾勇樹, 渡邉雄一, 松本真佑, 佐伯幸郎, 中村匡秀. ホームネットワークシステムにおけるプレゼンスセンシングのためのサービスフレームワークの提案. 電子情報通信学会技術研究報告. IE, 画像工学, No. 113(211), pp. 1–6, 2013.