ウェアラブルシステムにおける動的に変化する複数のデバイスの管理機能の研究

YAMADA, Masayuki / 山田, 真之

(出版者 / Publisher)
法政大学大学院情報科学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 情報科学研究科編 / 法政大学大学院紀要. 情報科学研究科編
(巻 / Volume)
11
(開始ページ / Start Page)
1
(終了ページ / End Page)
6
(発行年 / Year)
2016-03-24
(URL)
https://doi.org/10.15002/00012916

ウェアラブルシステムにおける動的に変化する複数のデバイスの管理機能の研究

A Study on Management Functions of Multiple and Dynamic Devices in a Wearable System

山田　真之
法政大学大学院情報科学研究科情報科学専攻
E-mail: masayuki.yamada.ub@stu.hosei.ac.jp

***Abstract*—Wearable devices, which can be worn by humans and animals, have drawn great attentions in recent years. In the current systems, wearables and applications are closely bounded with each other, it is thus difficult to make the systems extensible and flexible. Therefore, we propose a novel wearable system, called Wear-I, which is consisted of multiple wearables and able to carry out various wearable services. However, because these multiple wearables are often heterogeneous and dynamic change during their usages, it is crucial to effectively manage these wearables. This research is to study the necessary device management functions that are categorized into three parts: device information management (DIM), device usage management (DUM), and device operation management (DOM). DIM is for managing device registrations, working states and using logs. DUM is to support retrievals of device information, invocations of necessary devices in an application, and knowledge bases about the device features and usages. DOM is to provide the management of device information database, an interface for a user to operate devices and database, and a set of APIs directly useable by various wearable applications. A series of experiments have been conducted to present and test the proposed device management functions.**

Keywords—wearable; Wear-I; device; sensor; management; registration; profile; state; energy

I. 序論

従来の物よりもさらに小型化され，人や動物が身に着けることのできるコンピュータである「ウェアラブルデバイス」が，昨今注目を集めている．ウェアラブルデバイスを扱う研究やサービスは二つのアプローチに分けられる．一つは，UP 2[1]のような特定のデバイスを中心としたものであり，様々なアプリケーションがデバイスを中心として存在する．もう一つは，複数のデバイスを使用して礼拝動作の検知を行った Muaremi らの研究[2]のような特定のアプリケーションを中心としたものであり，アプリケーションを実行するために，様々なデバイスを扱うものである．これら二つのアプローチによる研究やサービスは盛んに行われているが，それぞれ問題が存在する．

デバイスを中心としたものは，アプリケーションが特定のデバイスに依存してしまうというデメリットが存在する．例えば，利用者の持つデバイスの電力がなくなる，デバイスを装着することを忘れる，デバイス自体が壊れる等，そのデバイスを使用できない状況ではサービスを続けることができない．これはアプリケーションに対して新たなデバイスを適用することができないためである．また，アプリケーションを中心としたものは，各々のデバイスがアプリケーションに依存しているため，デバイスを新たなアプリケーションに対して同時に使用する事や，再利用する事が難しい．

現状では，デバイスとアプリケーションが互いに依存しあっており，十分にデバイスの持つ機能を生かすことができていない． そこで，本研究室ではこの問題を解決するために，Wear-I[3]というシステムの研究を行っている．このシステムは，ユーザの手元にあるスマートフォンやウェアラブルデバイス内に実装されたローカルコントローラと，サーバやクラウド上に置かれたリモートコントローラにより構成される．これらのコントローラがユーザの身に着けるウェアラブルデバイスや，建物等の環境内に設置されたデバイス，またネットワーク上のソーシャルサービスとアプリケーションとの相互接続を行い，デバイスや取得データの管理や制御を可能とすることを目標としているが，システムの具体的な設計や実践的な成果はまだ得られていない．

そこで，本研究では Wear-I の内，ローカルコントローラにおいて実装される，デバイス管理機能の設計と，一部機能の実装を行う．管理機能は動的に変化するデバイスの数や状態を把握し，明示的にデバイスを指定することなく，センサデータの取得が可能である．これにより，アプリケーションは使用したいセンサを持つデバイスを容易に扱うことが可能となり，デバイスとアプリケーション間の依存関係が緩和されたことを目指す．

第Ⅱ章では関連研究について述べる．第Ⅲ章では Wear-I の概要を説明する．第Ⅳ章では，本研究で設計したデバイス管理機能の概要について記述する．第Ⅴ章では Device Information Management (DIM) について，第Ⅵ章では Device Usage Management (DUM)について，第Ⅶ章では Device Operation Management (DOM)について説明する．第Ⅷ章では部分的に実装したデバイス管理機能実装を実験により評価し，第Ⅸ章では，結論と今後の課題について論じる．

II. 関連研究

不特定多数のアプリケーションのために，デバイスやそれらが持つセンサ，および web サービスの情報に対して管理やデー

Supervisor: Prof. Jianhua Ma

タの取得を行う研究は，ユーザの身に着ける複数のウェアラブル加速度センサに対して電源管理や，データの管理を行うシステムの開発を行った村尾らの研究[4]や，論理エージェントと加速度センサやマイク等のセンサ情報取得部の分離を行い，それらの通信仕様の策定やセンサの管理を行うエージェントプラットフォームの開発を行った顧らの研究[5]，また特定の web サービスの機能を組み合わせることで，利用者の好みに合わせた新しいサービスを作り出すことのできる IFTTT[6]等多くの物が存在する．しかしながら，これらの研究やサービスは，アプリケーションとデバイス等の情報源の分離を行うことは可能ではあるが，使用できるセンサの種類や応用できるアプリケーションの範囲に問題が存在する．加速度センサや web サービス等の特定の情報源にしか対応していない，もしくはアプリケーションをエージェント化しなければならず，簡単にはデバイスを扱えない等，システムの汎用性が低い．そのためユーザが扱いたいアプリケーションやデバイスが，システムに対応していない可能性が高く，既存のデバイスやアプリケーションを応用することが難しい．

さらに，不特定多数のウェアラブルデバイスとアプリケーションを繋ぐという点で本研究と類似したサービスである Device Web API Manager[7]について述べる．このサービスはスマートフォンの端末内に仮想サーバを立てることで，サーバがウェアラブルデバイスとアプリケーションを繋ぎ，データの取得を行うことや，バイブレーション等のデバイスの機能を使用する事を可能とする．これは，Open Mobile Alliance[8]によって策定された OMA Generic Open Terminal API Framework V1.0 (GotAPI)[9]の一部を実装したサービスである．このサービスは，アプリケーションが様々なデバイスを使用する事を可能とするが，デバイスの管理を行うことはできない．例えば，アプリケーションが求める機能を持つデバイスを検索することはできないため，アプリケーション側から明示的に使用するデバイスを指定しなければならないし，またデバイスの状態の変化を仮想サーバが読み取り，アプリケーションに対して通知することもできない．このサービスは，あくまでデバイスとアプリケーションを容易に接続させることができるだけであり，スマートフォンに接続されたデバイスの管理はしていない．

本研究では，ウェアラブルデバイスとアプリケーションを繋ぐだけではなく，デバイスに対する管理や制御を行うことで，デバイスとアプリケーション間の依存関係を緩和させることを目指す．

III. WEAR-I システムについて

現状のウェアラブルデバイスを使用した研究やサービスでは，デバイスとアプリケーションが互いに依存関係にあり，柔軟性や拡張性に問題が存在する．これらの問題を解決するために，本研究室では Wear-I というシステムの研究を行っている．このシステムは，近い未来に到来するであろう，多くの人々が様々な種類のウェアラブルデバイスを身体の異なる部位に装着する社会を見据え，それらのデバイスを人間の手足や目鼻等の運動器官や感覚器官のように，自由に扱うことを可能とする．

Wear-I はユーザが扱う様々な目的を持つアプリケーションに対して，複数のデバイスを扱う上で総合的かつ適合的なサービスを行う．Wear-I は様々なウェアラブルデバイスをユーザが身に着けている状態を前提としている．それらのデバイスは既存の研究やサービスとは異なり，特定のアプリケーションに対して固定されておらず，様々なアプリケーションに対して，動的かつ適応的に一貫した動作を行うことが可能である．人間の運動器官や感覚器官のように，Wear-I はウェアラブルデバイスに対して，接続，管理，統合することで，それらを組織化して制御を行うことを可能とする．Fig. 1 に，Wear-I の概要を示す．

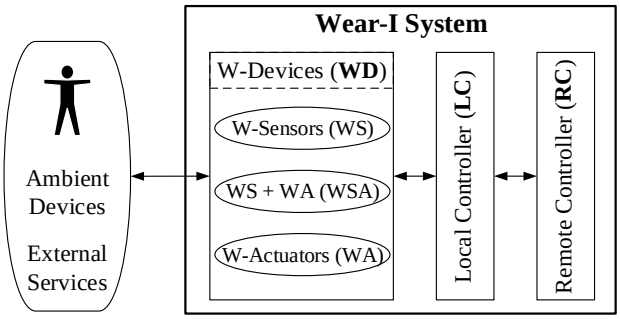

Fig. 1. Wear-I System の概要.

Wear-I はウェアラブルデバイスを三つに分類して管理を行う．具体的には，情報の取得を行うウェアラブルセンサ (WS)，情報の出力を行うウェアラブルアクチュエータ (WA)，WS と WA の機能を併せ持つウェアラブルセンサ＆アクチュエータ (WSA)に分類する．各デバイスはユーザに対して情報の入出力を行い，時には環境内に設置されたユーザの周囲に存在するデバイスと通信しながら動作する．各デバイスの制御はローカルコントローラ (LC)とリモートコントローラ (RC)で行われる．LC はデバイスの直接制御を行う機能を持ち，基本的にユーザの持つスマートフォン上に実装されることを想定している．RC は，サーバやクラウド上に存在し，LC の管理やデバイスより取得したデータの解析等を行う．

Fig. 2 に Wear-I のレイヤモデルを示す．Wear-I は他のウェアラブルデバイスを扱うシステムとは異なり，様々なアプリケーションに対して，動的にデバイスの数や状態の変化に対応することが可能である．しかし，そのために他のシステムと比べて多くのモジュールを持ち，システムの開発には系統的なアプローチが必要であると考えられる．そのため Wear-I の機能は，デバイスの管理や通信処理，システムを扱うためのプラットフォーム，データ管理，データ処理，タスク管理，アプリケーション管理を行う基本的な七つのレイヤと，システム全体の制御，セキュリティ，電源管理，ユーザビリティ管理を行う複数の層にわたって存在する四つの機能に分割される．

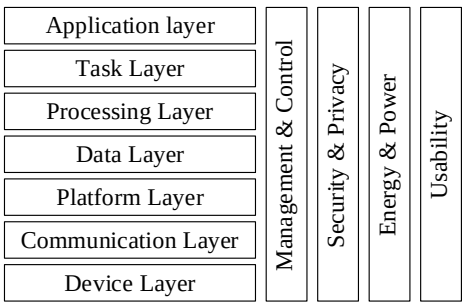

Fig. 2. Wear-I System レイヤモデル.

IV. デバイス管理機能概要

本研究では Wear-I の内，LC におけるデバイス管理機能の研究を行う．これは，Wear-I レイヤモデルの Device Layer に当たる部分である．デバイス管理機能は主に三つの機能を持ち，一つ目は，LC に接続されたウェアラブルデバイスのデバイス名，

MAC アドレス，デバイスの持つセンサ等のデバイス情報や，デバイスの現在の動作状況，デバイスの動作ログを取得しデータベースに対して保管する機能である．これらの保管を行うことで，アプリケーションが体中に着けられた複数のデバイスを組み合わせて，センサデータの取得や使用しているデバイスの異常を見つけ出すことが可能となる．二つ目は，保管されたデバイス情報より，アプリケーションが要求する条件に適合するセンサを選択し，センサデータの取得を行う機能である．これにより，アプリケーションは使用したいセンサを明示的に指定しなくとも，LC に対して，体のどの部位の，どんなセンサデータが欲しいのか通知を行いさえすれば，自動的に適切なセンサデータの取得を行うことが可能となる．三つ目は，データの取得を行っているデバイスが電源やセンサの不調等により継続して動作を行うことができなくなってしまった場合，同一機能を持つ他の正常なセンサを稼働させることにより，アプリケーションの動作を継続する機能である．これらの機能 によりデバイスの数や状態の変化に対して動的に対応することが可能となる．

Fig. 3 に，管理機能の構成を示す．デバイス管理機能は，大きく分けて Device Information Management (DIM)，Device Usage Management (DUM)，Device Operation Management (DOM)の三つのモジュールにより構成される．さらにそれぞれのモジュールはいくつかの子モジュールを持つ．DIM は，デバイス情報を取得する D-Profile (DP)，デバイスの現在の動作状況を読み取る D-State (DS)，デバイスの動作ログデータを取得する D-Log (DL)を持つ． DUM は，データベースに保存されたデバイス情報を検索する D-Retrieval (DR)，センサデータの取得を行う D-Invocation (DI)，アプリケーションの指定した条件を基に実際に使用するセンサを決定し，使用しているセンサに問題が起きたときに対処する D-Knowledge (DK)を持つ．DOM はデバイス管理機能におけるデータベースである DM Database (DMDB)，ユーザインタフェースを提供する DM User Interface (DMUI)，アプリケーションに対して，デバイス管理機能を扱うための API を提供する DM Application Interface (DAPI)を持つ．

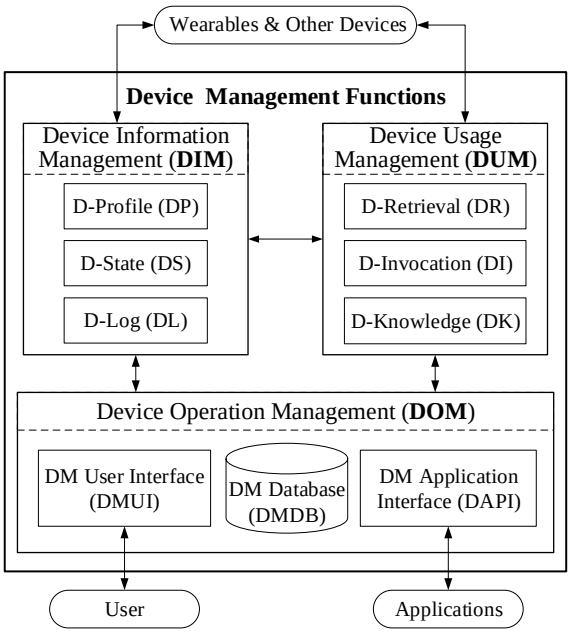

Fig. 3. デバイス管理機能の構成.

V. Device Information Management (DIM)

DIM は，コントローラと接続されたデバイスの MAC アドレスや使用部位，実装されているセンサ等の，デバイス情報の取得を行うことや，デバイスの動作状態のモニタリング，デバイス使用ログの記録を行う．DIM は D-Profile (DP)， D-State (DS), D-Log (DL)の三つの子モジュールにより構成される Fig. 4 に，DIM の構成を示す．

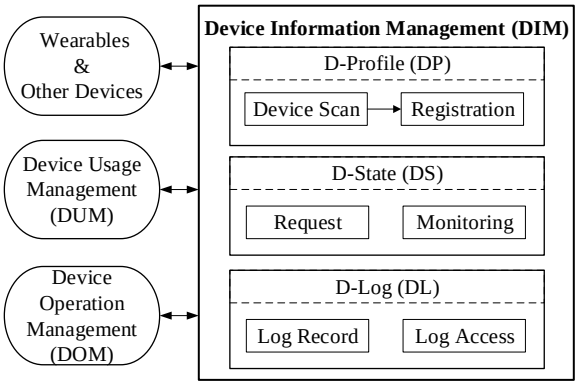

Fig. 4. DIM の構成.

DP はデバイスの情報を取得し，それらをデータベースに保管するモジュールである．DP は二つの子モジュールを持ち，まず Device Scan で，接続されたデバイスのデバイス名，MAC アドレス，および Bluetooth Low Energy の GATT プロファイルで使用可能なセンサ情報の取得が，自動的に行われる．また，初回起動時には LC に内蔵されているセンサ情報の取得も行う．具体的に取得できるセンサ情報は，ウェアラブルデバイスは，心拍，血圧，体温，屋内位置（緯度，経度）を，LC は，加速度，ジャイロ，地磁気，傾き，気圧，照度，近接，重力，直線加速度，回転ベクトル，温度，湿度である．次に，Registration で，Device Scan により自動的に取得された情報以外に必要なものを，ユーザが手動で入力する．入力する情報は，デバイスの種別，センサの位置情報である．デバイスの種別は，WS, WA, WSA, LC, AD （環境内に設置されたデバイス）である．センサの位置情報は，人体解剖学[10]を基に定義した 24 項目である．これらの情報はプルダウンメニューにより入力可能であるため，ユーザが文字入力をする必要はない．最後に，取得されたデバイスの情報をデータベースに保管する．

DS は LC に接続され，センサデータの取得を行っているデバイスに対し，そのデバイスが正常な動作を行っているか判定し，異常が発覚した場合，DUM の DK に対して適切な通知を出すモジュールである．DS は Request と Monitoring の二つの子モジュールを持つ．Request は DUM のモジュールである DK の要求に従い，特定のデバイスに対して正常に動作をしているのか検査をし，その結果を返すと同時にデータベースに結果を保存する．検査項目はデバイスとの接続状態，デバイスの電池残量，センサの稼働状態である．それぞれの取る値は，デバイスとの接続状態は電波強度を，デバイスの電池残量は残量の数値を，センサの稼働状態はそれぞれのセンサデータの値を取得する Monitoring は，現在センサデータの取得を行っているデバイスに対して，正常な動作を行っているか定期的に検査を行う．検査項目は上記の Request と同一である．データベースへの保存も同様に行う．

DL は使用しているデバイスの持つセンサの使用開始，終了情報や，エラー情報をログデータとして記録するモジュールで

ある．ログデータは DUM の DK でセンサを決定するときに使用され，またユーザに対して可視化を行うことも可能である．DL は Log Record，Log Access の二つの子モジュールより成り立つ．Log Record は使用しているデバイスの動作ログを取得し，データベースに保存する．ログデータはセンサの使用開始なら 1，正常にデバイスを終了したなら 2，接続状態の異常による終了なら 3 のようにイベント id をデータベースに保存する．Log Access は他のモジュールの要求に応じ，データベースに保存されているログデータを読み込み，データを送信する．

VI. Device Usage Management (DUM)

DUM は DIM で取得したデバイス情報を使用し，アプリケーションが必要とする機能を持つデバイスを検索，デバイスの取得したセンサデータをアプリケーションに通知，また，デバイス動作ログを読み取り，電源切れ等によりデバイスが使用できなくなった場合に他のデバイスを探し，アプリケーションの使用デバイスを代替する．DUM は，D-Retrieval (DR)，D-Invocation (DI)，D-Knowledge (DK)の，三つの子モジュールから成る． Fig. 5 に DUM の構成を示す.

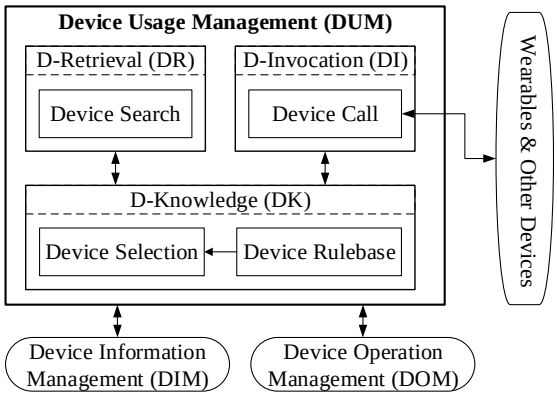

Fig. 5. DUM の構成.

DR はデータベースに登録されたデバイスの検索を行い，条件に適合するセンサを持つデバイスを見つけ出すモジュールである． DR は，Device Search という子モジュールにより成り立つ．Device Search は DK から送られてきたデバイスの条件により，データベースに保存された情報の検索を行い，条件に適合したデバイスのリストを作成する．条件として使用できる要素は，デバイスの名前，MAC アドレス，種別，通信方法，およびセンサ種別，センサ位置である．また，センサ位置については，オプションにより，使用したいセンサ位置に近い位置の物を，第二候補として検索を行うことが可能である．そして，検索されたデバイスのリストを DK に送信し，実際にアプリケーションが使用するデバイスは DK により決定される．

DI はデバイスからセンサデータの取得をし，またデータ取得時の取得頻度の制御を行うモジュールである．DI は，Device Call という子モジュールより成り立つ．Device Call は，DK から使用したいデバイスのデバイス種別と MAC アドレス，センサ種別が送られてきた際，デバイスよりセンサデータを取得してアプリケーションに返す．また，センサデータの取得頻度の制御を行う．

DK は DS による状態検査や DL が取得したログデータを基に，アプリケーションがどのデバイスを使用すれば一番良いのか，また使用しているデバイスに問題が起きたときにどの様な対処を行うのか判断を行うモジュールである．DK は Device Selection と Device Rulebase の二つの子モジュールにより成り立つ．Device Selection は DR の Device Search によって作成されたデバイスのリストを基に，アプリケーションが実際に使用するデバイスの決定を行う．決定には，DS の Request 機能により検査されたデバイスの状態と，DL の Log Access により取得されたリスト内のデバイスの過去の利用履歴を総合的に解析して判断を行う．Device Rulebase は，使用しているデバイスに問題が起きたときに問題が起きたデバイスのセンサデータの取得を終了し，Device Selection により，使用デバイスの再決定を行う．

VII. Device Operation Management (DOM)

DOM はユーザとのコミュニケーション用 GUI の設定や，アプリケーションとの通信，システムの持つデータベースの運用を行う．DOM は，DM Database (DMDB)，DM User Interface (DMUI)，DM Application Interface (DAPI)の三つの子モジュールより成り立つ． Fig. 6 に DUM の構成を示す.

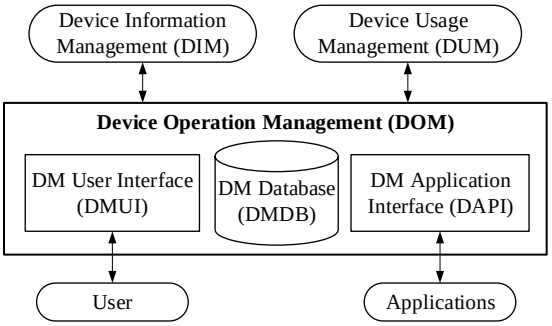

Fig. 6. DOM の構成.

DMDB はデバイスの持つ機能や，現在の動作状況，過去の動作ログの保存，読み出しを行うデータベースである．DMDB の構成を Fig. 7 に示す．なお，図中のテーブル間のリレーションにおける 1 と*は，それらのテーブルが一対多の関係にあることを示している．

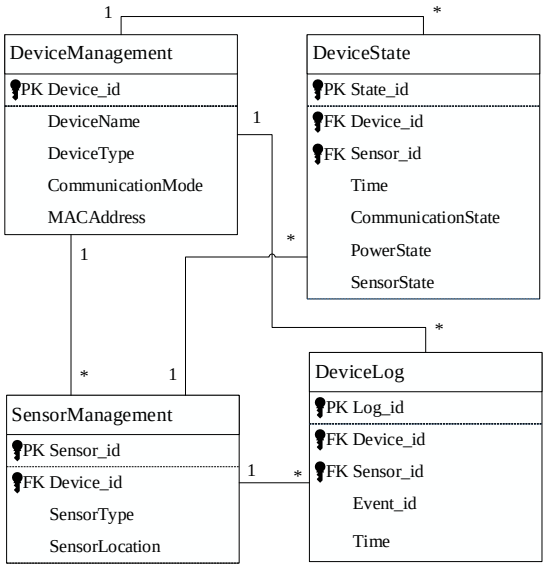

Fig. 7. DMDB の構成.

DMDB はデバイス自体の基本的な情報を保存する Device Management，デバイスの持つセンサ情報を保存する Sensor Management，デバイスの動作状況を保存する Device State，デバイスの過去の動作ログを保存する Device Log の四つのテーブルより成り立つ．

DMUI はシステムとユーザ間のインターフェースに対応しており，デバイス管理機能内のモジュールがユーザからの入力を求めたり，ユーザがデータベースに保管されているデータの出力を求めた際に，それらの視覚化を行うモジュールである．視覚化を行う物は， DP では，スキャンしたデバイスリストとユーザ入力フォームの二つの視覚化を， DMDB においては，Device Management テーブル，Sensor Management テーブル，Device State テーブル，Device Log テーブルの各テーブルに保存された情報の視覚化を行う．

DAPI はアプリケーションに対してデバイス管理機能を扱うための API を提供するモジュールである．これは，Android のプロセス間通信機能である AIDL により実装される．AIDL はクライアント側プロセスがホストと通信することで，ホスト側プロセスのオブジェクトにあるメソッドを利用するための機能である．これらの通信は非同期で行われるため，クライアントはホストに対するリクエストのコールバックを受け取る必要がある．また，クライアントは，ホストにより生成された通信用定義である aidl ファイルを取り込まなければならない．

VIII. 実装と実験

第Ⅳから Ⅶ 章で示したデバイス管理機能の内，特に重要なモジュールを Android スマートフォン上に実装し，アプリケーションが特定の条件を持つセンサを必要とする状況を想定し実験を行った．実装したモジュールは DIM の D-Profile，D-State，DUM の D-Retrieval，D-Invocation，DOM の DM User Interface，DM Database である．実験項目は，接続したデバイス情報を取得できるか，指定した条件を満たすセンサを持つデバイスを検索できるか，デバイスを使用しセンサデータの取得ができるかの三つである．この実験により，実装したモジュールが正しく動作を行うか確認した上で，MAC アドレス等を使用して明示的に使用するデバイスのセンサを指定しなくとも，アプリケーションが必要とするセンサを持つデバイスからセンサデータを取得することが可能か検証を行う．これにより，デバイスとアプリケーション間の依存関係が緩和され，アプリケーションが容易に必要なセンサを持つデバイスを扱えることを示す．使用するデバイスを Fig. 8 で示す．

Fig. 8.　使用するデバイス

実験に使用するデバイスは，LC として Android スマートフォンの SH-04F，センサデータ取得用に心拍計の機能を持つ MIO LINK および Polar H6 である． MIO LINK と Polar H6 については GATT プロファイルを使用する事で容易にセンサデータを取得できる．また，SH-04F に内蔵されたセンサを読み取り，一部のセンサデータを取得する．内蔵されたセンサは加速度，ジャイロ，地磁気，傾き，照度，近接，重力，直線加速度，回転ベクトルの 9 種類であり，この中で加速度とジャイロデータの取得を行う． また UP24 と Ring についても，デバイス情報を取得する．

次に実験結果を示す．Fig. 9 は周囲のデバイスの検出からデバイス情報をデータベースに保存するまでの結果を示している．

Fig. 9.　デバイス検出からデバイス情報登録までの結果.

まずデバイス管理機能の一部を実装したアプリケーションを起動すると，SH-04F が内蔵する各種センサを検索し，データベースに保存する．次に周囲に存在する Bluetooth デバイスの検出を行う．Fig. 9 の左下の図が検出されたデバイスのリストである．これを見ると今回実験で使用する MIO LINK および Polar H6 のスキャンに成功したことが分かる．次にデバイスリストより MIO LINK を選択し，デバイスの情報を取得した．そしてユーザ入力画面に遷移し，デバイスの種別とセンサ位置の入力を行った． Fig. 9 の右側の図は，MIO を選択した際のユーザ入力画面である．下線で示された値は，それぞれデバイス管理機能が自動的に読み取った値である．デバイス名と MAC アドレスは左下の図の丸で示されたものと同一の値であり，また MIO LINK の持つセンサは心拍計なので正しい値が取得されていることが分かる．点線で示された値はユーザの入力したものであり，デバイス種別に WS，センサ位置に forearm（前腕）を入力した．最後に DB INPUT ボタンを押すとデータベースに各種情報が保存される．この後，他のデバイスに対しても同様の操作を行った．Device Management テーブルに保存されたデバイス情報の値を TABLE Ⅰ に示す．

TABLE Ⅰ.　デバイス情報取得結果

Device id	*Device Name*	*Device Type*	*Communication Mode*	*MAC Address*
1	SH-04F	LC	Wi-Fi, BLE	24:26:42:7C:A7:A1
2	MIO	WS	BLE	DD:B7:C1:58:C9:24
3	Ring	WS	BLE	C3:D8:B5:BD:2C:93
4	UP24	WS	BLE	E2:87:FC:69:5E:C0
5	Polar H6 32F661F	WS	BLE	00:22:D0:32:FC:66

Device Management テーブルに SH-04F，MIO LINK，Ring，UP24，Polar H6 のデバイス情報が保存されていることが分かる．次に Sensor Management テーブルに保存された各デバイスの持つセンサ情報の値を TABLE Ⅱに示す.

TABLE II.　デバイスの持つセンサ情報取得結果

Sensor id	*Device id*	*Sensor Type*	*Sensor Location*
1	1	GYROSCOPE	LC
2	1	ACCELEROMETER	LC
3	1	MAGNETIC	LC
4	1	ORIENTATION	LC
5	1	ROTATION	LC
6	1	LINEAR	LC
7	1	GRAVITY	LC
8	1	PROXIMITY	LC
9	1	LIGHT	LC
10	2	Heart Rate	forearm
11	5	Heart Rate	breast

Device_id が 1 の SH-04F には内蔵された 9 個のセンサ情報が，Device_id が 2 の MIO LINK には Heart Rate センサ情報が，Device_id が 5 の Polar H6 にも同じく Heart Rate センサ情報が正しく保存されていることが分かる． これらの結果より，接続したデバイス情報を取得することが可能であることを示せた.

次にデバイスの現在の状態を取得し，問題がなければ使用したいセンサの条件を指定してデバイスの検索を行い，検索したデバイスの持つセンサのデータを取得する． Fig. 10 はデバイスの現在の状態と，検索条件を満たすセンサを持つデバイスのセンサデータが取得されたことを示す.

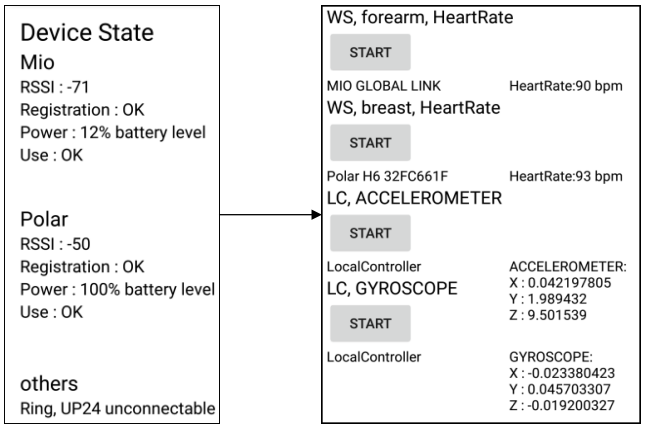

Fig. 10.　データベースに登録されたデバイスの現在の状態と，条件を満たすデバイスの検索およびセンサデータ取得結果.

まず Fig. 10 の左がデバイスの現在の状態を示す．取得できる状態の値は通信強度，データベースに保存されているか，電池状態，デバイスの使用可否である．Fig. 10 を見ると MIO と Polar のそれぞれの値が正しく取得できたことと，Ring は使用できないことが分かる．次に Fig. 10 の右が，条件を満たすデバイスの検索およびセンサデータ取得結果である．画面上の START ボタンを押すと，データベースからボタンの上に書かれた条件を満たすセンサを持つデバイスの検索が行われる．条件を満たしたデバイスが見つかると，自動的にセンサデータの取得が開始され，画面上に現在のセンサデータの値が表示される．Fig. 10 を見るとそれぞれのセンサデータの取得が正常に行われていることが分かる．この結果より，アプリケーションが明示的に使用するデバイスのセンサを指定しなくとも，必要なセンサデータの条件を指定することで，条件を満たすデバイスからセンサデータを取得することが可能であることが示せた．したがって，デバイスとアプリケーション間の依存関係が緩和され，アプリケーションは容易に必要なセンサを持つデバイスを扱えるといえる.

IX. 結論

本研究は，ユーザの持つウェアラブルデバイスのデバイス情報，動作状態を読み取り，それらを基にアプリケーションの求める条件を満たしたセンサを持つデバイスの検索と，センサデータの取得を行うことが可能なデバイス管理機能について研究を行った．そして，デバイス管理機能全体の設計と一部の実装を行い，明示的にデバイスを指定することなく，センサデータの取得が可能であることを検証した．これにより，アプリケーションは使用したいセンサを持つデバイスを容易に扱うことが可能となり，デバイスとアプリケーション間の依存関係が緩和されたことを示した．今後は Wear-I システムのデバイス管理に対して応用することができると考えられる.

今後の課題としては，まず設計したデバイス管理機能の，すべてのモジュールの実装が第一に挙げられる．これにより設計に対する妥当性の検証を行わなければならない．次に，管理可能なデバイスの種類の拡大が挙げられる．現状の管理機能ではスマートフォンに実装されたセンサもしくは，Bluetooth Low Energy の GATT サービスを使用する事で取得を行うことが可能なセンサを持つデバイスの管理しか行うことができない．そのため，ウェアラブルデバイスの加速度計や，ジャイロセンサ等の有用なセンサの使用や，デバイスのバイブレーションや LED の点灯といったデバイスの動作に関する機能を使用する事ができない．よって，それらを使用可能とする，新たな機能の設計を行う必要がある．手法としては，開発メーカの SDK や API を取り込む，通信用のプロファイルを新たに設計し，管理機能とデバイスに対して実装する，他の研究やサービスを利用することの三つが挙げられる．最後に，今後はデバイス管理機能だけでなく，取得したセンサデータの管理や，データの加工機能，タスク機能，またサーバ側のデバイス管理機能等の Wear-I 全体の研究が必要となる.

文献

[1] UP2, https://jawbone.com/store/buy/up2

[2] A. Muaremi, G. Tröoster, J. Seiter, and A. Bexheti, “Monitor and Understand Pilgrims: Data Collection using Smartphones and Wearable Devices”, Proc UbiComp, International Workshop on Human Activity Sensing Corpus and Its Application (HASCA), 2013.

[3] J. Ma and R. Huang, ‘‘Wear-I: A New Computing Paradigm in Wearable Computing’’, Keynote, in Proc. IEEE Conf. Ubiquitous Computing & Commun., Liverpool, U.K., pp. 1063–1068, 2015

[4] 村尾和哉, 竹川佳成, 寺田努, 西尾章治郎, “ウェアラブルコンピューティングのためのセンサ管理デバイスの設計と実装”, 情報処理学会論文誌 49(9), pp. 3327-3339, 2008.

[5] 顧優輝, 真部雄介, 菅原研次, “ユビキタスデバイスとデジタルリソースを論理エージェントに接続するためのエージェントプラットフォームの開発”, 情報処理学会論文誌 56(1), pp. 284-294, 2015.

[6] IFTTT, https://ifttt.com/

[7] Device Web API Manager, http://www.gclue.io/dwa/index.html

[8] Open Mobile Alliance, http://openmobilealliance.org/

[9] OMA Generic Open Terminal API Framework V1.0 (GotAPI), http://www.gclue.io/dwa/index.html

[10] 人体解剖学, https://ja.wikipedia.org/wiki/人体解剖学