博士論文

IoT センサデータ地産地消基盤に関する研究

中村 優吾

2020 年 3 月 17 日

奈良先端科学技術大学院大学
情報科学研究科 情報科学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士（工学）授与の要件として提出した博士論文である．

中村 優吾

審査委員：

安本 慶一 教授	（主指導教員）
中島 康彦 教授	（副指導教員）
荒川 豊 客員教授	（副指導教員）
諏訪 博彦 特任准教授	（副指導教員）
水本 旭洋 大阪大学 特任助教	（副指導教員）

IoT センサデータ地産地消基盤に関する研究*

中村 優吾

内容梗概

Internet of Things 技術の急速な発展に伴い，ユビキタスコンピューティング環境の実現が現実的なものとなりつつある．しかしながら，生活空間や身の回りのあらゆるモノが IoT デバイス化され相互に連携しながら，あらゆる場所で即時かつ安全に人々の生活を支援し，地域社会を豊かにする真のユビキタスコンピューティング環境の実現には至っていない．真のユビキタスの環境を実現するには，(1) 実環境で利用できる IoT デバイスの選択肢が少なく，センシング対象が限られてしまう，(2) IoT センサデータのタイムリーな処理・分析・応用ができていない，という 2 つの問題点を解決する必要がある．本博士論文研究では，地域における IoT センサデータの生産を促進するとともに，収集された IoT センサデータを地域に存在する計算資源を活用して即時に処理・分析・応用する「地産地消」を基本コンセプトとする，IoT センサデータの地産地消基盤の実現を目指した研究を行った．具体的には，前述の 2 つの問題点を解決するために，A: 実用可能な IoT デバイス開発とセンサデータ収集の簡略化，B: エッジ IoT デバイス群による弾力性のあるデータ処理の実現という 2 つの研究課題に取り組んだ．研究課題 A に関して，8 つのセンサ（加速度，ジャイロ，磁気，光，UV，温度，湿度，圧力），BLE 通信モジュール，フラッシュメモリ，バッテリー，充電回路を搭載した小型マルチセンサボード，データ収集用のソフトウェア，自由に拡張可能な 3D プリンタ用のケースデータによって，モノの IoT デバイス化とデータ収集のプロセスの簡略化を実現する SenStick プラットフォームを設計・開発した．そして，ベルトや箸の IoT デバイス化といったケーススタディを通じて SenStick プラットフォームを評価し，SenStick を活用することで開発プロセスが簡略化され，試作された IoT デバイスを用いて，長時間の行動データ収集および高精度（F 値 0.95）の日常行動認識を実現できること

*奈良先端科学技術大学院大学 情報科学研究科 情報科学専攻 博士論文, NAIST-IS-MTDD1761011, 2020 年 3 月 17 日.

を確認した．研究課題 B に関して，地域に存在する IoT デバイス群をセンサデータプロバイダ，計算資源プロバイダ，サービスコンシューマとして抽象化し，それらのリソースをセンサデータ処理サービスの需要に応じて調整・分配しながら，一つのサービス系として弾力性のあるデータ処理を実現する IFoT プラットフォームを設計し，プロトタイプを開発した．そして，実機とシミュレーション（エリア：2km × 2km，ノード数：4000 台）の評価実験によって，250ms の遅延要求を満たすタイムリーな処理を実現すると共に，サービスの需要が増加した場合でも，近隣のエッジ IoT デバイスを活用したスケールアウトによって QoS を維持できること確認した．

キーワード

Internet of Things，IoT センサデータ収集，IoT 試作プラットフォーム，エッジコンピューティング，ストリーム処理，分散処理

A Study on In-situ IoT Sensor Data Sensing and Processing Platform*

Yugo Nakamura

Abstract

Due to the rapid growth of the Internet of Things technology, the realization of the ubiquitous computing society has been expected. However, the attractive vision of ubiquitous computing society that all the smart object around us supports people's lives and community timely and safely has not been completely achieved. In order to achieve the attractive vision of ubiquitous computing society, solutions of two problems are necessary: (1) types of available IoT devices are limited, (2) Real-time processing, analysis, and utilization of IoT sensor data are difficult. In this doctoral dissertation research, we aim to realize the In-situ IoT sensor data sensing and processing platform which promotes the collecting of IoT sensor data and processes the collected IoT sensor data quickly using the computation resources in the local area based on the "local production for local consumption" concept. In this dissertation, we focus on two research challenges to solve the two problems mentioned above: (A) How to simplify IoT device development and sensor data collection, (B) How to elastically process sensor data by using edge IoT devices. Regarding research challenge A, we designed and developed a comprehensive sensing platform called SenStick, which is composed of hardware (ultra-tiny all-in-one sensor board including 8 sensors: acceleration, gyro, magnetism, light, UV, temperature, humidity, and pressure, BLE communication module, flash memory, and battery), software (iOS, Android, and PC), and 3D case data. The platform aims to simplify the

*Doctoral Dissertation, Graduate School of Information Science,
Nara Institute of Science and Technology, NAIST-IS-DD1761011, March 17, 2020.

prototyping IoT device and data collection process. The SenStick platform was evaluated through case studies of IoT chopsticks and belt. We confirm that prototyping and data collecting processes are simplified by using SenStick, and long-term data collection and daily activity recognition with high-accuracy (F value 0.95) are achieved. Regarding research challenge B, we designed an elastic sensor data processing platform called IFoT(Information Flow of Things), which coordinates sensor data processing service according to service demand by using resources of local IoT devices abstracted as sensor data providers, computing resource providers, and service consumers. Then, we developed a prototype of IFoT platform and evaluated IFoT platform by real device experiments and simulation (area: 2km x 2km, number of nodes: 4000). We have confirmed that IFoT can achieve timely processing that satisfies a 250ms delay requirement, and maintain QoS by using adaptive scaling out that dynamically offloading to neighboring edge IoT devices even when service demand increases.

Keywords:

Internet of Things, IoT sensor data collection, IoT rapid prototyping platform, edge computing, stream processing, distributed processing

目次

図目次

表目次

第 1 章　序論

1.1　IoT システムの現状

1991 年，マーク・ワイザ氏によって将来のコンピューティング環境のあるべき姿として，ユビキタスコンピューティングの概念が提唱された．ユビキタスコンピューティング [1,2] のパラダイムでは，人々は計算機の存在を意識することがなくなり，我々の生活空間や身の回りのあらゆるモノに搭載された計算機は，相互に連携しながら，その場のコンテキストに応じて我々の日常的な活動を支援することが想定されている．それから約 30 年の時を経て，IoT（Internet of Things）[3] 技術の発展とともに，ユビキタスコンピューティングの実現が現実的なものとなりつつある．近年の IoT システムは，図 1.1 に示すように，センシング，ネットワーキング，コンピューティング，アプリケーションという 4 つの要素から構成されており，それぞれセンサデータの収集，センサデータの転送，センサデータの処理，センサデータの応用という役割を担っている．IoT システムの開発においては，「IoT システムをどの分野に導入するのか」，「センサデータをどのように処理・応用するのか」など，目的とするアプリケーション像を定めた上で，センシング，ネットワーキング，コンピューティングという各要素において，「どんなデバイスから何のセンサデータを集めるのか」，「データを転送するための通信方式はどうするのか」，「セ

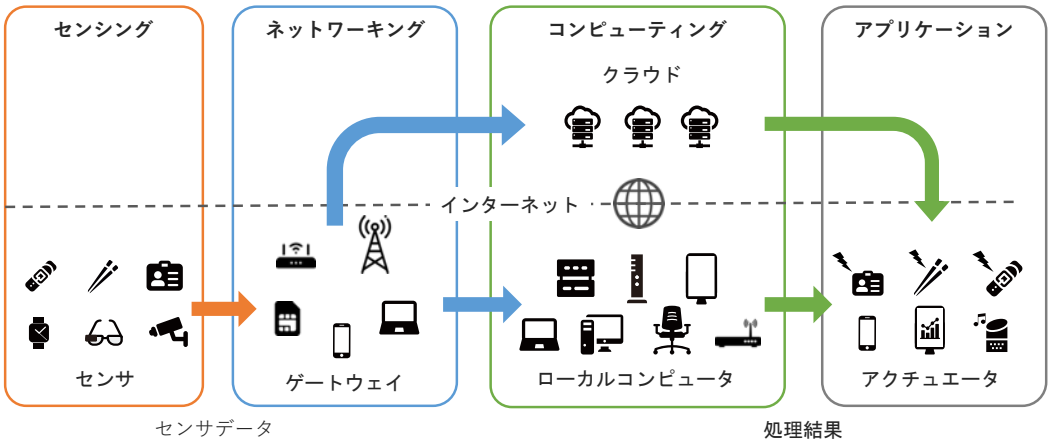


図 1.1: IoT システムの基本構成

ンサデータをどこで，どのように処理するのか」など必要に応じてデザインすることが非常に重要である．

例えば，リストバンド型のウェアラブルセンサを用いてユーザの加速度データを収集し，加速度データから機械学習ベースの手法を用いてユーザの日常行動（座る，歩く，走る，etc.）を認識し，その結果をユーザのスマートフォンアプリケーションを通じて可視化することで，ライフロギングアプリケーションが実現できる．また，ユーザが長時間座っていることを認識して，近くのデジタルサイネージが歩行を促すように話かけてくるなど，IoT 技術を活用することで，ユーザに対してコンテキストアウェアなサービスを提供することが可能である．しかしながら，座りながらどんな行動をとっているのかなど，より細かな行動コンテキストを認識するためには，リストバンド型のウェアラブルセンサでは不十分である．例えば，箸を IoT デバイス化することができれば，食べ方や食べる量など，食事行動を認識することが可能になり，食べ過ぎの人に注意を促すことが可能になる．また，このようなコンテキスアウェアサービスを実現するためには，センサデータからコンテキストを推論するためのデータ処理を素早く実行することが求められる．

IoT システムの構成要素をまとめたタクソノミーを図 1.2 に示す．また，以下に，各カテゴリの説明を記述する．

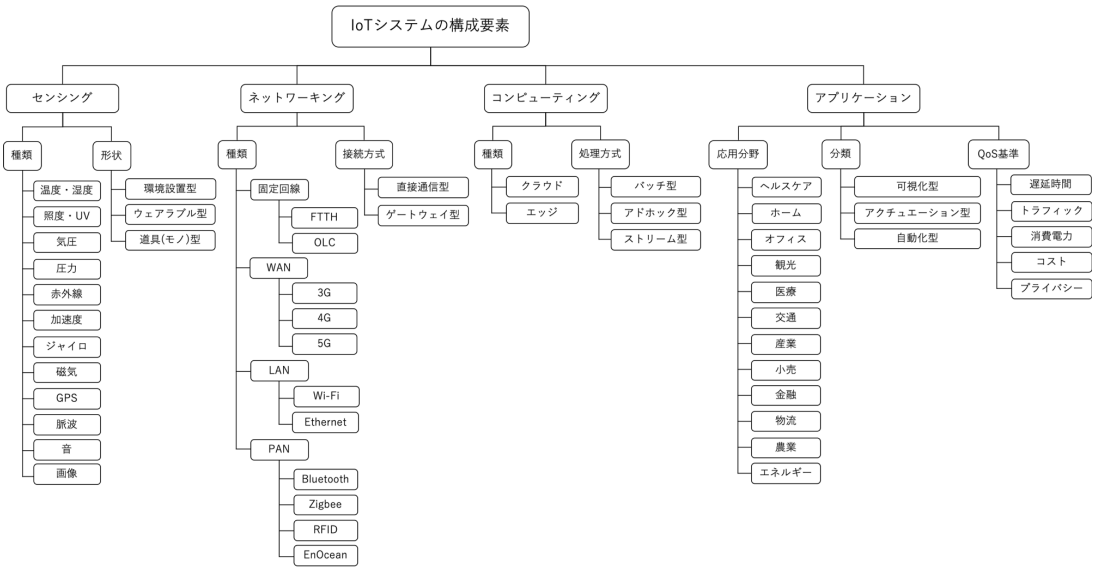


図 1.2: IoT システムの構成要素を分類したタクソノミー

- **センシング**：センシングパートは，センサの種類とセンサの形状・用途という 2 つの項目で分類できる．センサの種類は，環境の状況を把握するための温度，湿度，照度，気圧センサや，対象物の動きを把握するための加速度，ジャイロ，磁気センサが代表的であり，近年，普及しているほとんどの IoT デバイスにこれらのセンサが搭載されている．また，音を取得するマイクや，画像を取得するカメラもセンサの一種として捉えることができる．センサの形状は，環境の状況を確認する目的で屋内や屋外の定点に設置する環境設置型，スマートウォッチやスマートグラスなど身体に装着するウェアラブル型，箸やペンなど常に身につけている訳ではないが食事中など特定のシチュエーションで利用する道具型という 3 つの項目で分類することができる．現状，道具型の IoT デバイスはあまり普及しておらず，IoT の普及に伴って今後の増加するカテゴリだと考えられる．
- **ネットワーキング**：ネットワーキングパートは，ネットワークの種類とインターネットへの接続方式という項目で 2 つの分類できる．ネットワークの種類は，固定回線，WAN（Wide Area Network），LAN（Wide Area Network），PAN（Personal Area Network）という 4 つの項目で分類できる．インターネットへの接続方式は，IoT デバイスが固定回線や WAN を通じてダイレクトにインターネットに接続する直接通信型と，インターネットに接続しているゲートウェイデバイスに LAN や PAN 経由で接続し，ゲートウェイデバイスを介して間接的にインターネットに接続するゲートウェイ型に分類できる．例えば，SIM を搭載したスマートフォンは直接接続型であり，Bluetooth や Zigbee を搭載した小型センサ端末はゲートウェイ型に分類できる．
- **コンピューティング**：コンピューティングパートは，コンピューティングパラダイムの種類とデータ処理方式という 2 つの項目で分類できる．IoT データのコンピューティングパラダイムとして，クラウドコンピューティングとエッジコンピューティングが存在する．現状の IoT システムの多くは，クラウドコンピューティングを活用している．しかし，近年は，即時性を求めるアプリケーションに対応するために，コアネットワーク上のクラウドではなく，ネットワークの端（エッジ）に位置する通信基地局側に隣接するサーバ

でデータを処理するエッジコンピューティングパラダイムが注目を集めている．データ処理方式は，ストレージに保存されたデータを一括処理するバッチ型，ストレージに保存されたデータをリアルタイムに近い速度で処理するアドホック型，生成されたデータをストレージに保存することなくリアルタイムに処理するストリーム型に分類できる．

- **アプリケーション**：アプリケーションパートは，対象とする応用分野，アプリケーションの種類，QoS（Quality of Service）基準という 3 つの項目で分類できる．IoT アプリケーションは，ヘルスケアからエネルギーまで多岐の分野での応用が期待されている．アプリケーションタイプの分類として，収集されたセンサデータを分析し，人が見てわかる形に変換して表示する可視化型，センサデータから得られた分析結果に基づいて，身の回りに存在するデバイスを駆動させるアクチュエーション型，自動車の自動運転やドローンの自動制御などセンサデータに基づいてデバイス自体を自動で操作・制御する自動化型に分類できる．QoS 基準は，アプリケーションの性質に依存し，一般的には，遅延時間，ネットワークトラフィック，消費電力，コスト，プライバシーなどが考えられる．

1.2 IoT システムの問題点

身の回りのあらゆるモノが IoT デバイス化され，相互に連携しながら，あらゆる場所で即時かつ安全に人々の生活を支援し，地域社会を豊かにする真のユビキタスコンピューティング環境の実現に向けては，図 1.3 に示すように，理想的な IoT システムと現状の IoT システムの間にいくつかのギャップが存在する．1 つ目に，センシングパートにおいて，実環境で利用できる IoT デバイスの選択肢が少なく，センシング対象が限られてしまう点である．IoT 技術の発展により，多くの企業から様々な IoT デバイスが製造されているが，箸やペンなど身の回りのあらゆるモノが IoT デバイス化された真のユビキタスコンピューティング社会が実現されているとは言い難い．今後は，企業だけでなく一般の個人が身の回りの問題を解決するために，自由にオリジナルの IoT デバイスを試作しながら，様々な状況をセンシングし，サービスとして活用することが理想的である．2 つ目に，コンピューティング

パートにおいて，IoT センサデータのタイムリーな処理・分析・応用ができていない点である．現状，IoT システムの多くが生のセンサデータをすべてクラウドサーバにアップロードしてから処理するクラウド集中型の処理方式を採用している．しかし，クラウドへのデータ転送にかかる高い通信コストやデータ発生源とクラウドを往復する際に発生する遅延時間などの点で効率的ではない．IoT 技術の普及に伴って，今後ますます身の回りに存在する計算資源が質的にも量的にも増加することから，状況に応じて，これらの計算資源を活用しながらデータを素早く処理することが理想的である．これらを踏まえて，本研究では以下に示す 2 つの問題点の解決を目指す．

問題点 1：実環境で利用できる IoT デバイスの選択肢が少なく，センシング対象が限られてしまう

ウェアラブルコンピューティング，IoT 技術の発展に伴い，ここ数年で，様々な IoT デバイスが普及したものの，現在，普及しているデバイスの多くは，環境設置型もしくは，リストバンド型やメガネ型などに限られており，ユビキタスコンピューティングのビジョンで想定されているような，箸やペンなど身の回りのあらゆるモノが IoT デバイス化された社会が実現されているとは言い難い．また，普及している全ての IoT デバイスから生のセンサデータを取得できるとは限らず，販売元の企業から特別なライセンスアカウントを発行しなければセンサデータを取得できない

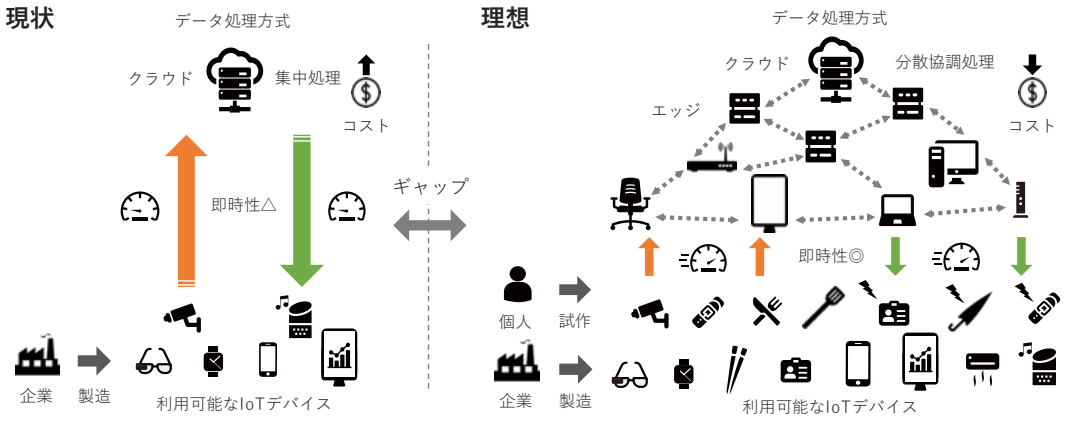


図 1.3: IoT システムの現状と理想のギャップ

場合が存在する．そのため，企業だけでなく，一般の個人が自由な発想で IoT デバイスを試作し，データを収集して，自身の問題解決に応用できる環境の実現が，IoT 技術のさらなる発展に必要不可欠であると考える．近年，メイカーズムーブメント [4,5] やデジタルファブリケーション技術 [6] の発展に伴い，Arduino, mbed, Raspberry Pi などのさまざまなラピッドプロトタイピング用のマイコンが普及し，一般ユーザでも様々なセンサ部品を繋いで，オリジナルの IoT デバイスを開発したり，3D プリンタを活用して気軽に外装ケースをデザインできるようになりつつある．しかしながら，普及しているマイコンボードのサイズは比較的大きく，スタンドアローンで動作するためには，充電回路やストレージなどを追加する必要があり，最終的なデバイスサイズが大きくなってしまう．結果として，一般ユーザがペンや箸などの小さなモノに組み込み可能で，実用に耐えうる小型のデバイスを開発するのは非常に困難である．また，デバイスを一から開発する場合，デバイスだけでなくデバイスからデータを収集するためのソフトウェアも開発する必要が出てくる．それぞれのデバイス毎に，データ収集用のソフトウェアを開発するのは，時間的なコストが発生し，容易ではない．そのため，誰もが簡単に，あらゆる形状の IoT デバイスを試作し，様々なフィールドで素早くセンサデータを収集することを可能にするための解決策が必要である．

問題点 2：IoT センサデータのタイムリーな処理・分析・応用ができていない

現状の IoT システムでは，IoT デバイスで収集した生のセンサデータがすべてクラウドサーバにアップロードされ，クラウド上で集中的にすべてのデータ処理が実行されている．このようなクラウド集中型の処理方式では，クラウドへのデータ転送にかかる高い通信コストやデータ発生源とクラウドを往復する際に発生する遅延時間，プライバシ性の高いデータをクラウドに蓄積するリスクが発生するなど，いくつかの問題が残されている．今後は，IoT デバイスがさらに増加することが予想され，地域で生成されるデータ量が増大することから，クラウド依存のアプローチの問題が深刻化する恐れがある．近年，通信基地局に専用の処理サーバ（エッジ/フォグサーバ）を導入し，データ発生源の近くに存在するサーバ上で即時性が求められる処理タスクを実行するエッジコンピューティング [7–9]・フォグコンピューティング [10–12] と呼ばれる新しいコンピューティングパラダイムが注目を集めて

いる．しかしながら，これらのパラダイムの普及は初期段階であり，全ての基地局にエッジサーバが導入されているわけではないため，ユーザがいる場所の近くに必ずしもエッジサーバが存在しないことが考えられる．そのため，エッジサーバが存在しない環境でも，データ発生源の近くに存在する IoT デバイスなどローカルの計算資源を有効活用しながら，弾力性のあるデータ処理を提供するための仕組みが必要となる．

1.3　研究目的・課題とその解決策

本研究では，図 1.4 に示すように，地域における農作物の生産を促進し，収穫された農作物を積極的に地域で消費することによって地域社会を活性化させることを狙った地産地消に着目し，IoT 技術を用いて地域に顕在化する様々な課題を解決すべく，「地域における IoT センサデータの生産を促進するとともに，収集された IoT センサデータを地域に存在する計算資源を活用して即時に処理・分析・応用する」IoT センサデータの地産地消の実現を目的とする．地域社会において，IoT センサデータの地産地消を実現することによって，以下に示すいくつかの利点が期待できる．

まず，誰もが自由な発想で IoT デバイスの試作やセンサデータの収集をできるようにする地産の促進によって，これまでにない斬新な IoT デバイスのユースケースやキラーアプリケーションの創出が期待できる．近年，内部のアイデア，データ，ノウハウを積極的に公開し，外部のリソースとの統合を図ることで新しい価値を生み出すオープンイノベーション [13] の重要性が高まっている．そのため，IoT 技術の応用においても，新しいモノを IoT デバイス化するアイデアやノウハウ，収集したセンサデータや，そのデータの処理方法などをコミュニティ内で積極的に公開することが有効であり，IT 教育環境の充実といった観点からも社会的な波及効果が

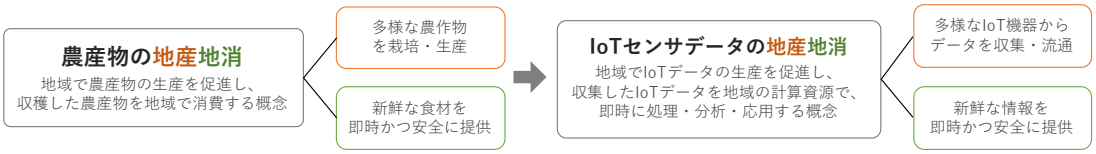


図 1.4: IoT センサデータの地産地消

期待できる．

次に，既に導入されている IoT デバイスの余剰な計算資源を有効活用する地処の促進によって，システム全体としてのコストの節約が期待できる．近年，計算機科学の技術発展によって，デバイス 1 台あたり計算能力はますます向上している．また，IoT の普及に伴い，今後は，ますます地域社会に導入される IoT デバイスの数は増加することが予測されており，結果として，1 つの地域コミュニティが抱える計算資源の総量は増加していくことが予想できる．そのため，ローカルに存在する余剰の計算資源を有効活用する地産地消アプローチは急激な技術革新を伴った時代の流れに順応している．また，クラウドサーバにデータをアップロードするために使用していた通信帯域やデータ処理に使用していたクラウドサーバの計算資源など，これまで消費していた高価な使用コストを削減できる．

最後に，センサデータや処理プログラム，計算資源のシェアリングによる地域における新しい経済圏の確立が期待できる．近年，集めた IoT センサデータを他者に売買することが可能な IoT データ取引プラットフォーム EverySense [14] をはじめとして，国内外で IoT センサデータの取引市場など IoT 技術を活用した新しい経済圏の構築を目指す取り組みが増え始めている．今後は，この流れがさらに加速し，IoT センサデータだけでなく，IoT センサデータを価値化するための処理プログラムや，IoT センサデータを処理するための計算資源すらも，需要に応じて相互に取引される時代が訪れる可能性があり，IoT 技術の浸透によって生まれるこのよ

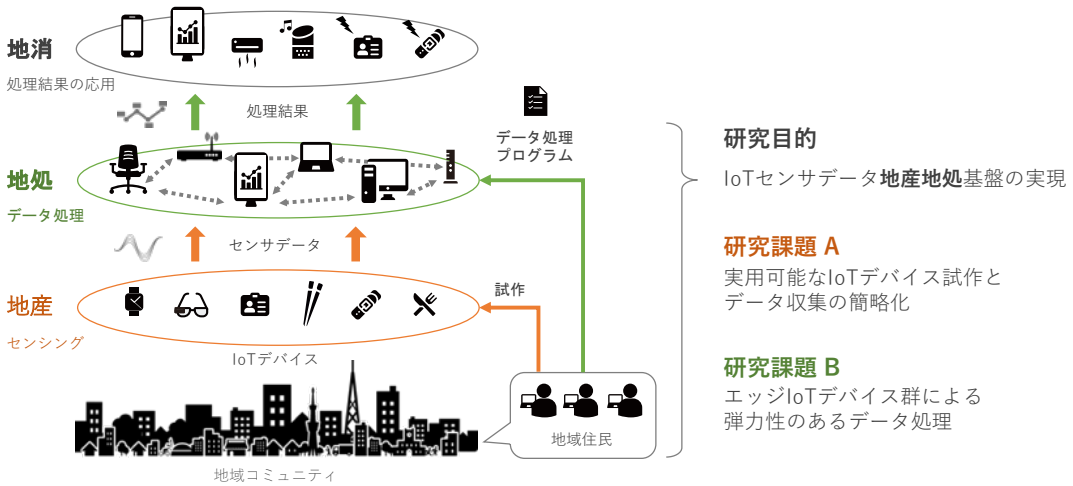


図 1.5: IoT センサデータ地産地消基盤の概要

うな新しい経済圏の発展が，地域活性化に貢献することが期待できる．

これらの内容を踏まえて，本研究では，地域に存在する IoT デバイスを最大限に有効活用しながら，IoT 技術の力で地域に顕在化する様々な課題を解決することを可能にする「IoT センサデータ地産地消基盤」（図 1.5）の実現を目指す．そして，前節で述べた 2 つの問題点「1: 実環境で利用できる IoT デバイスの選択肢が少なく，センシング対象が限られてしまう」，「2: IoT センサデータのタイムリーな処理・分析・応用ができていない 」を解決するために，それぞれ 2 つの研究課題「A: 実用可能な IoT デバイス開発とセンサデータ収集の簡略化」，「B: エッジ IoT デバイス群による弾力性のあるデータ処理」を設定する．そして，本論文では，これら 2 つの研究課題に取り組んだ結果をまとめている．

研究課題 A：実用可能な IoT デバイス試作とセンサデータ収集の簡略化

研究課題 A の概要を図 1.6 に示す．研究課題 A では，「問題点 1： 実環境で利用できる IoT デバイスの選択肢が少なく，センシング対象が限られてしまう」を解決することを目指す．この問題の原因として，これまでセンサが導入されていない新しいフィールドでのデータ収集を可能にする IoT デバイスのアイデアを思いついたとしても，実用可能なハードウェアを試作するのが困難であることが考えられる．例えば，日常の食卓環境でユーザの食事行動を分析するために，実用可能な箸

図 1.6: 研究課題 A の概要

型の IoT デバイスを試作しようとした場合，デバイスサイズを非常に小型にする必要があり．Arduino などの市販のマイコンを用いて実現することは困難である．加えて，試作したデバイスを用いたセンサデータ 収集を行うためには，データ収集用の専用ソフトウェアを一から開発する必要があるため，データを収集するためのシステムを構築するのに時間がかかりすぎてしまう．本研究では，モノの IoT デバイス化におけるハードウェアとソフトウェアの開発プロセスを簡素化することによって，これらの問題の解決を試みる．そして，誰でも簡単に対象のフィールドに適した IoT デバイスを試作でき，素早くセンサデータを収集できるような環境の実現を目指す．この目的の実現に向けて，どこにでも組み込み可能な超小型のマルチセンサボード，マルチプラットフォームに対応したデータ受信用のソフトウェア，自由にデバイス形状をデザイン可能な 3D プリンタ用のケースデータという全く新しい組み合わせによって，包括的なモノの IoT デバイス化とデータ収集をサポートする SenStick プラットフォームを提案する．SenStick プラットフォームの詳細は，2 章に記述されている．本論文では，ベルトや箸の IoT デバイス化など，いくつかのケーススタディを通じて，SenStick プラットフォームの有効性を検証した．

研究課題 B：エッジ IoT デバイス群による弾力性のあるデータ処理

研究課題 B の概要を図 1.7 に示す．研究課題 B では，「問題点 2 : IoT センサデータのタイムリーな処理・分析・応用ができていない」を解決することを目指す．この問題の原因として，現状の IoT システムの多くがクラウド集中型の処理方式を採用していることが考えられる．これによって，データ発生源とクラウド間でデータを往復する際に発生する高い通信コストや遅延時間の無駄が発生している．近年，データ発生源の近くに存在するエッジサーバ上で即時性が求められる処理タスクを実行するエッジコンピューティングが注目を集めているが，普及の初期段階であり，全ての基地局にデータ処理専用のサーバが導入されているわけではない．本研究では，エッジサーバが存在しない環境でも，データ発生源の近くに存在する IoT デバイスなどローカルの計算資源を有効活用しながらデータ処理を行うことで，これらの問題の解決を試みる．そして，データ発生源に存在するエッジ IoT デバイス群による弾力性のあるデータ処理の実現を目指す．この目的の実現に向けて，地域

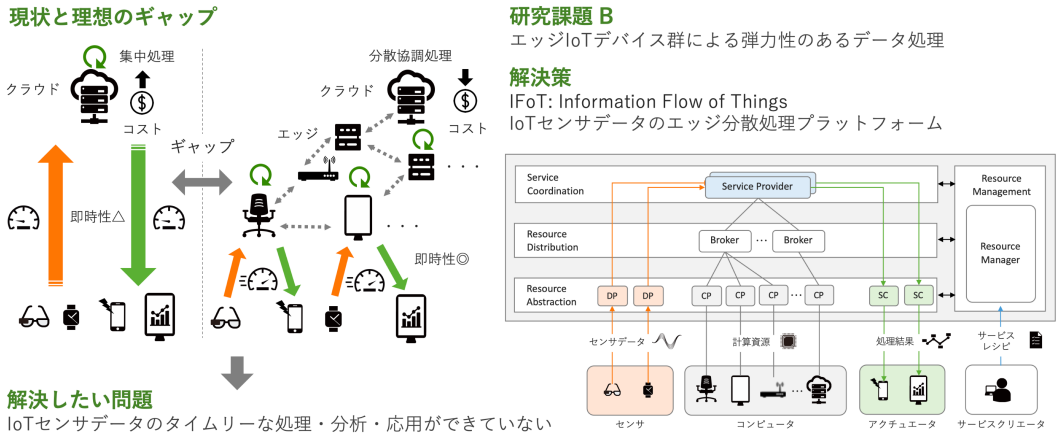


図 1.7: 研究課題 B の概要

に存在する IoT デバイス群をセンサデータプロバイダ，計算資源プロバイダ，サービスコンシューマとして抽象化し，それらのリソースをセンサデータ処理サービスの需要に応じて調整・分配しながら，一つのサービス系として弾力性のあるデータ処理を実現する IFoT プラットフォームを提案する．IFoT プラットフォームの詳細は，3 章に記述されている．本論文では，IFoT プラットフォームのプロトタイプを開発し，実機とシミュレーションの評価実験によって，その有効性を検証した．

1.4 論文構成

研究課題と論文構成の対応関係を図 1.8 に示す．2 章では，課題 1：実用可能な IoT デバイス開発とデータ収集の簡略化を達成すべく，誰もが簡単に，あらゆる形状の IoT デバイスを試作し，様々なフィールドで素早くセンサデータを収集することを可能にする SenStick プラットフォームについて記述する．3 章では，課題 2：エッジ IoT デバイス群による弾力性のあるデータ処理を達成すべく，既存のクラウドやエッジコンピューティングのパラダイムで活用することを想定されていなかったローカルに存在する無数のエッジ IoT デバイスが有している余剰の計算資源を用いて，QoS を考慮した効率的なセンサデータ処理を実現する IFoT プラットフォームについて述べる．最後に，4 章で，本研究で得られた結果をまとめ，今後の方向性について記述する．

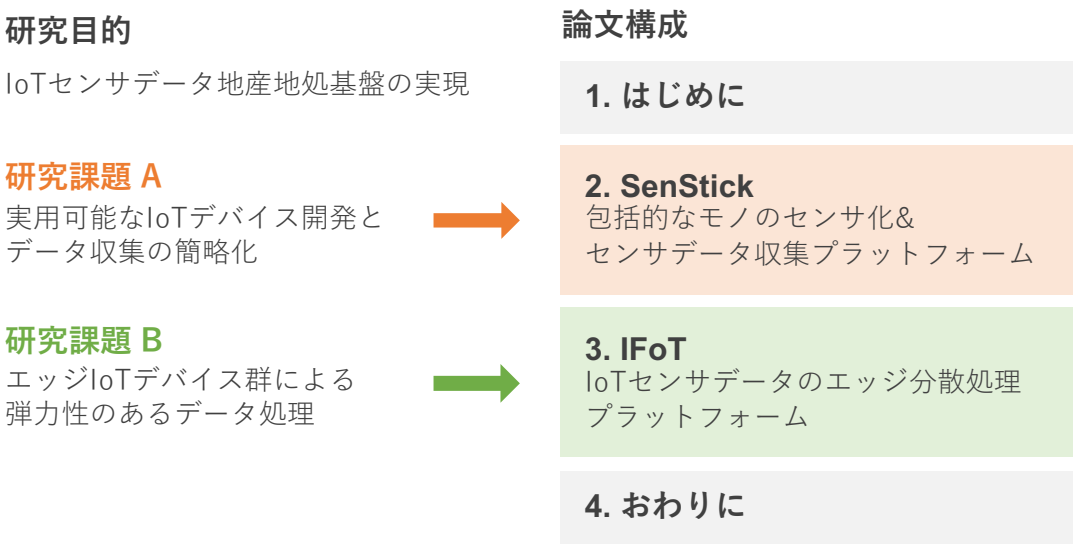


図 1.8: 研究課題と論文構成の対応関係

第 2 章　SenStick: 包括的な IoT データ生成/収集基盤

本章では，「 実環境で利用できる IoT デバイスの選択肢が少なく，センシング対象が限られてしまう」という問題点を解決すべく，小型のマルチセンサボード，データ受信用のソフトウェア，自由に拡張可能な 3D プリンタ用のケースデータという新しい組み合わせによって，モノの IoT デバイス化におけるハードウェアとソフトウェアの開発プロセスを簡素化し，誰もが簡単に，あらゆる形状の IoT デバイスを試作し，様々なフィールドで素早くセンサデータを収集することを可能にする SenStick プラットフォームについて記述する．

2.1　はじめに

Internet of Things の急速な進歩によって，現実世界から多くの種類の情報を感知することが可能になっている．人間行動認識の研究分野では，様々なセンサを備えた IoT デバイスが，人々の行動を感知するためのツールとして広く使用されている．そして，センサがどこに配置されるかという観点から，3 つのアプローチに分類できる．

1 つ目のアプローチは，スマートホームやスマートオフィスのような環境内にセンサを配備する方法である．このアプローチでは，RFID，カメラ [15]，マイク [16]，消費電力計 [17] などの様々なセンサが，人々の活動を監視するために使用されてきた．

2 つ目のアプローチは，人間の身体にセンサを取り付ける方法である．この方法では，従来，スマートフォンが多数のセンサ（加速度，ジャイロ，地磁気，照度，気圧，etc.）に加えて，通信機能，処理プロセッサ，大容量ストレージを備えているという理由から汎用のセンシングツールとして広く使用されてきた [18, 19]．今日では，ウェアラブルコンピューティングの普及に伴って，スマートフォンの代わりに，スマートウォッチやスマートアイウェアのようなウェアラブル機器に使用されている [20]．JINS MEME [21] は，メガネと鼻パッドのブリッジに眼電図（EOG）電極を備えたスマートアイウェアである．眠気や疲労などの内部コンテキストを認識するために，目の動きを測定する．また，我々の研究グループでも，腹囲を測定

できるスマートベルト [22] の研究を進めている．

3 番目のアプローチは，私たちの生活を取り巻くさまざまなモノにセンサーを埋め込む新しいパラダイムである．村尾ら [23] は，トイレットペーパーホルダーに加速度センサーを取り付けて，人を認識する研究に取り組んできる．HAPIfork [24] は，食事行動を監視するための加速度センサーをフォークに埋め込みました．また，居住者のさまざまな活動を検知するために複数のセンサをリモコンを追加する研究も存在する．我々は，近い将来，IoT の成長により多くのものがセンサや通信機能を備えた IoT デバイスになると考えている．そして，身の回りのモノから生成された新しいセンサデータに焦点を当てた研究が加速すると，これまでよりも細かい行動の認識が可能になり，新しいコンテキストアウェアサービスを実現できる可能性がある．例えば，箸を IoT デバイス化することができれば，食べ方や食べる量など，食事行動を認識することが可能になり，食べ過ぎの人に注意を促すことが可能になる．本論文では，センサや通信機能を非電子的な身の回りのモノに追加する手順について，「モノの IoT デバイス化」という単語を使用する．

このように，箸やペンなどをはじめとして身の回りの小さなモノの IoT デバイス化が期待される一方で，現状では小さなものの IoT デバイス化するためには企業とのコラボレーションが必須であり，一般の個人や研究者にとっては難易度は高い．近年，メイカーズムーブメントやデジタルファブリケーション技術の発展に伴い，Arduino, mbed, Raspberry Pi などのさまざまなラピッドプロトタイピング用のマイコンが普及し，一般ユーザでも様々なセンサ部品を繋いで，オリジナルの IoT デバイスを開発したり，3D プリンタを活用して気軽に外装ケースをデザインできるようになりつつある．しかしながら，普及しているマイコンボードのサイズは比較的大きく，スタンドアローンで動作するためには，充電回路やストレージなどを追加する必要があり，最終的なデバイスサイズが大きくなってしまう．結果として，一般ユーザがペンや箸などの小さなモノに組み込み可能で，実用に耐えうる小型のデバイスを開発するのは非常に困難である．また，デバイスを一から開発する場合，デバイスだけでなくデバイスからデータを収集するためのソフトウェアも開発する必要が出てくる．それぞれのデバイス毎に，データ収集用のソフトウェアを開発するのは，時間的なコストが発生し，容易ではない．IoT の本質はデータ分析であるため，身の回りのあらゆるモノの IoT 化とそれらのデバイスからのデータ

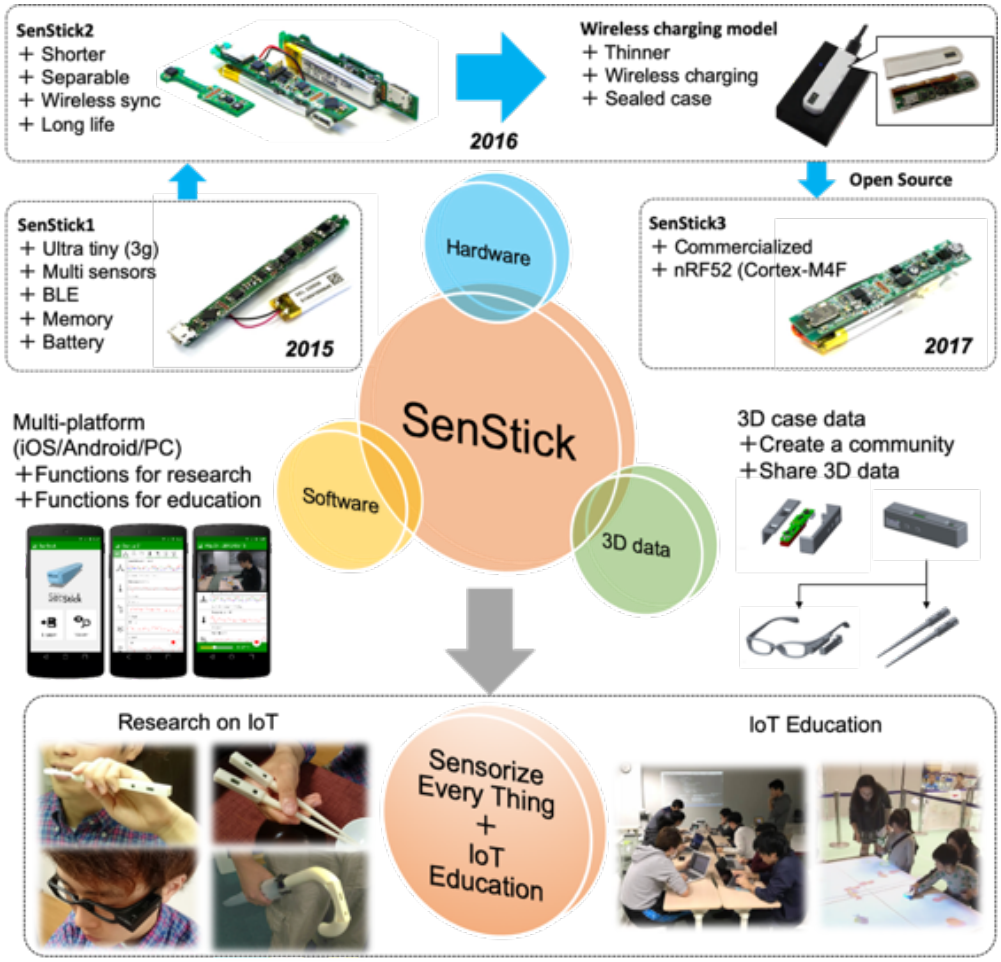


図 2.1: SenStick プラットフォームの概要

収集プロセスを簡略化するための解決策が必要である．そして，一般の個人が自由な発想で IoT デバイスを試作し，データを収集して，自身の問題解決に応用できる環境の実現が，IoT 技術のさらなる発展に必要不可欠であると考える．

本研究では，「いかにして，これまで IoT が導入されていないフィールドに，素早くセンサを導入し，長時間のセンサデータを収集するのか」という研究課題を解決すべく，どこにでも組み込み可能な超小型かつ拡張可能なハードウェア，マルチプラットフォームに対応したデータ収集ソフトウェア，目的に応じて適したデバイス形状にすることができる 3 D ケースデータという，これまでにない全く新しい組み合わせによって，対象のフィールドに適した IoT デバイスの試作コストを簡

素化し，素早くデータ素早くセンサデータを収集できる環境を提供する SenStick プラットフォームを提案する．SenStick プラットフォームの概要を図 2.1 に示す．SenStick ハードウェアは，8 つのセンサ（加速度，ジャイロ，磁気，照度，紫外線強度，温度，湿度，圧力），フラッシュメモリー，BLE モジュール，バッテリー，充電回路が高密度に組み込まれた超小型のマルチセンサボードである．サイズは，55mm（W）× 10mm（H）× 5mm（D）であり，その重量はバッテリーを含めて約 3（g）である．バッテリ寿命はを搭載することで 15 時間 40mAh になる．また，BLE 通信を介した無線データ同期機能と DFU（Device Firmware Update）も実装されている．I2C 対応の外部拡張端子を備えており要望に応じてデバイスを拡張可能である．例えば，振動モータを追加することで，SenStick を触覚アクチュエータとしても活用可能である．SenStick ソフトウェアとして，モバイルデバイス（iOS，Android）およびデスクトップ PC（Linux，MacOS，および Windows）用のサポートアプリケーションが提供されている．各アプリケーションは，センサデータのリアルタイムに監視および記録することができ，各センサのパラメータも設定可能である．PC バージョンでは，複数の SenStick と同時に接続が可能である．また，行動認識のための機械学習モデルを構築する際に使用できるセンサデータから特徴量を抽出するためのライブラリも提供されている．SenStick 3D データとして，3D プリンタで出力可能な SenStick 用の 3D ケースデータが提供されている．CAD ソフトを使用して，基本形の 3D ケースデータを拡張することが可能なため，様々な形状ので 3D ケースを簡単に設計することができる．

本章の残りの部分は次のように構成されている． 最初にさまざまな既存製品を紹介し，SenStick と比較する．次に，SenStick プラットフォームの要件を整理し，SenStick プラットフォームの詳細について説明する．その後， SenStick を使用したいくつかのケーススタディを紹介し，SenStick プラットフォームの有効性について考察する．最後に，本章のまとめを記述する．

2.2 関連研究

行動認識の研究で活用されている，競合のセンサーを表 2.1 にまとめる．

Texas Instruments の SensorTag [25] は，SenStick と同様に最も競争力のある

センサーボードである．SensorTag に埋め込まれたセンサーは，UV センサーを除き，SenStick とほぼ同じである．ただし，記録機能とフラッシュメモリは搭載されていない．SensorTag を使用する場合は，BLE 通信によって，スマートフォンで常に接続することを前提としているため，スタンドアローンでのデータ収集はできない．SensorTag の主なターゲットは環境センシングであるため，加速度をはじめとしたモーションセンサのセンシング周波数は最大 1Hz に制限されており，電源には非充電式コイン電池が採用されている．そのため，ボードの形状，サイズ，重量は，SenStick よりも大きく，小さなモノの IoT デバイス化には適していない．

IoT Smart Module [26] は，SensorTag のような複数のセンサーを搭載した小さなセンサーボードである．サポートアプリケーションを通じて，センサデータの記録やセンサの設定が変更できるが，サポートアプリケーションが Android OS しか対応していないという欠点がある．SensorTag と同様に，非充電式コイン電池を採用しており，SenStick よりもサイズが大きいため，身体への装着などウェアラブル用途に限られている．SensorTag よりも高いサンプリング周波数を設定できるが，ユーザーは定期的にコイン電池を交換する必要がある．また，BLE の伝送容量を

表 2.1: 競合センサとの仕様比較

		SenStick	SensorTag (CC2650)	IoT Smart Module	TSND121	AX6
						
Size (mm)		50(W)x10(H)x10(D)	42(W)x32(H)x10(D)	44(W)x27(H)x11(D)	37(W)x46(H)x12(D)	23(W)x32.5(H)x8.9(D)
Weight (g)		3.5	28	-	22	11
Stand-alone recording		○	×	×	×	○
Smartphone app		○	○	○	×	×
Desktop app		○	△	×	○	○
Battery life		15 (hour)	1 (year)	1 (year)	6 (hour)	6 (hour)
Battery type		Rechargable / Removable	Coin Battery	Coin Battery	Rechargeable	Rechargeable
Network		BLE	BLE	BLE	Bluetooth 2.0	BLE
Waterproof		△	×	×	×	○
Data processing		○	×	×	×	×
I2C Extention		○	×	×	×	×
Sensors	Acceleration	○	○	○	○	○
	Gyro	○	○	×	○	○
	Magnetic	○	○	○	○	×
	Air pressure	○	○	○	○	×
	Temperature	○	○	○	×	×
	Humidity	○	○	○	×	×
	Light	○	○	○	×	×
	UV	○	×	○	×	×
	Mic	×	○	×	×	×

考慮すると，センシングにおけるサンプリング周波数が高い場合，センサデータが欠損する場合がある．したがって，正確なセンサデータを必要とする研究には適していない．

ATR-Promotions の TSND121 [27] は，ロボティクスの研究分野で使用されている，もう 1 つの競争力のあるセンサである． 9 軸のモーションセンサーと気圧センサーを備えており，データの送信に Classic Bluetooth 2.0 を採用している．主に，人間やロボットに装着させる用途で使われているため，SenStick よりもサイズが大きく，重い仕様となっている．そのため，箸やペンなど小さなモノへの装着には適していない．制御 PC から最大 7 つのセンサと同時に接続できるが，センサデータを記録するために常に PC と接続している必要があるという制約が存在する．

Axivity 社の AX6 [28] は，耐水性を備えた長時間データ収集用の 6 軸慣性計測ユニット（IMU）である．AX6 は，加速度計，ジャイロ，温度，照度センサに加えて，1024MB のメモリを備えており，加速度を 100Hz のサンプリングレートで最大 7 日間連続で計測可能である．ただし，地磁気，湿度，気圧，UV センサーなどのいくつかのセンサを搭載していない．まだ，データの記録専用のデバイスとなっており，データ収集後に専用のアプリケーションから USB 経由でデータを取り出す必要がある．そのため，通信機能を備えておらず，リアルタイムでのデータ受信などの用途には活用できない．

このように，既存のプラットフォームでは，「いかにして，これまで IoT が導入されていないフィールドに，素早くセンサを導入し，長時間のセンサデータを収集するのか」という研究課題を解決することができていない．一方，SenStick プラットフォームでは，この問題を解決するために，どこにでも組み込み可能な超小型かつ拡張可能なハードウェア，マルチプラットフォームに対応したデータ収集ソフトウェア，目的に応じて適したデバイス形状にすることができる 3 D ケースデータという，これまでにない全く新しい組み合わせを採用している．これによって，誰でも簡単に対象のフィールドに適した IoT デバイスを試作でき，素早くセンサデータを収集できる環境を提供しているという点で高い有用性と新規性を兼ね備えている．加えて，SenStick ハードウェアが，I2C 対応の外部拡張端子を備えているため，対象のフィールドや用途に応じて，新しいセンサやアクチュエータを追加することができるという拡張性の面でも高い優位性がある．

2.3　システム要件

本節では，SenStick プラットフォームのユースケースを示すとともに，SenStick プラットフォームの要件をまとめる．

2.3.1　ユースケースシナリオ

SenStick プラットフォームでは，利用できる IoT デバイスの種類が少ないという問題を解決するために，一般の個人が自由な発想で簡単に IoT デバイスを試作し，データを収集・分析できる環境の実現することを目標としている．SenStick のユースケースを図 2.2 に示す．このように SenStick では，モノの IoT デバイス化のアイデア着想から，データ収集・分析までのプロセスを簡素化し，素早く実施できるようにすることを目指している．そのために，IoT デバイス化のためのハードウェア，データ収集・分析のためのソフトウェア，様々な形状をデザイン可能な 3D データという 3 要素で，モノの IoT デバイス化を支援する．

図 2.2: SenStick のユースケースシナリオ

2.3.2　システム要件

ハードウェア要件

SenStick プラットフォームは，既存のスマートフォンやウェアラブルデバイスでは実現できなかった身の回りの小さなモノを IoT デバイス化することを目的としている．従って，センサや通信モジュール，フラッシュメモリなどの重要な機能を小さなセンサボードに組み込む必要があり，各センサーは高いサンプリングレートでデータを収集できる必要がある．BLE 通信の帯域は信頼性の高いデータ通信を保証するものではないため，スマートフォンなどの記録デバイスがない状態でも，スタンドアロンでデータを収集・記録できる必要がある．また，コイン電池を頻繁に交換するのは不便であるため，SenStick には充電式の電源システムが求められる．さらに，目的の用途によっては，SenStick ハードウェアに標準搭載されている素子で不十分な場合が考えられるため，I2C などの規格に従って自由に拡張可能であることが望ましい．

ソフトウェア要件

SenStick プラットフォームは，IoT デバイス化された成果物からのデータ収集と分析を簡素化することを目的としている．従って，サポートソフトウェアはマルチプラットフォームである必要があり，さまざまなオペレーティングシステムとの互換性が必要である．サポートソフトウェアには，データの記録などの基本的な機能だけでなく，複数の SenSticks からのセンシングデータの同時記録などの追加機能，データ分析用の特徴量抽出プログラムや，データ処理用のシンプルなプログラミングインターフェイスも求められる．

3D データ要件

SenStick プラットフォームは，身の回りのあらゆるモノの IoT デバイス化を促進することを目的としている．しかしながら，箸をはじめとした小さなモノは，単に SenStick を装備するだけでは不恰好になってしまう場合が考えられる．そのため，近年の 3D プリンターの普及を考慮して，3D プリンターで出力可能かつ CAD ソフトを使用して自由に拡張可能である SenStick の 3D ケースを提供することが望ましい．加えて，誰かが作成した 3D ケースデータは，SenStick のコミュニティサイトを通じて他のユーザーと共有できることが望ましい．

2.4　SenStick プラットフォーム

SenStick プラットフォームは，ハードウェア，ソフトウェア，3D データという 3 つの要素で構成されており，これらを通じて，あらゆるモノの IoT デバイス化とデータ収集・分析をサポートすることを目指している． そのため，SenStick プラットフォームは，IoT デバイスのラピッドプロトタイピングや，IoT の教育の用途で有用である． 本節では，これらの各要素の詳細について説明する．

2.4.1　SenStick ハードウェア

セクション 1 で説明したように，SenStick ハードウェアは 2015 年からを 3 回更新されている．ここでは 2017 年に開発された SenStick3 の仕様に基づいて記述する．センサーボードのサイズは 55mm（W）× 10mm（H） × 5mm（D）．重量は

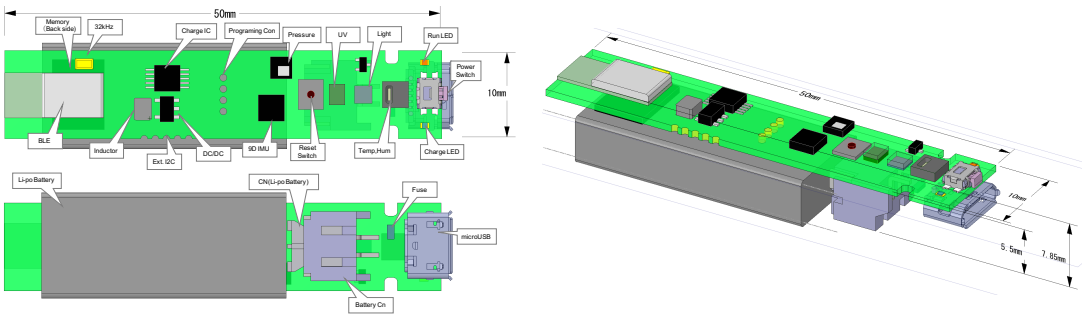


図 2.3: SenStick ハードウェアの基盤レイアウト

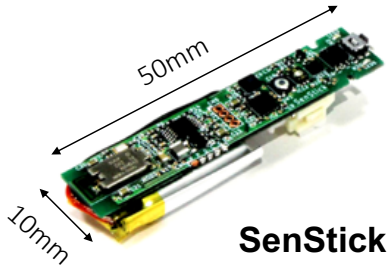


図 2.4: 実際の SenStick ハードウェア

表 2.2: SenStick ハードウェアに搭載されているセンサ素子

センサの種類	モデル番号	消費電力
加速度・ジャイロ・地磁気	MPU-9250	280 μA $\sim$ 3.7 mA
気圧	LPS25HBTR	4 μA $\sim$ 25 μA
温度/湿度	SHT20	270 μA $\sim$ 330 μA
照度	BH1780GLI-E2	120 μA $\sim$ 200 μA
紫外線 (UV)	VEML6070	100 μA $\sim$ 250 μA

バッテリーを含めて約 3（g）である．この小さなボードには，8 つのセンサ（加速度，ジャイロ，磁気，光，UV，温度，湿度，圧力），フラッシュメモリ（32M バイト），BLE，バッテリー，充電回路が搭載されている．SenStick ハードウェアの基板レイアウトを図 2.4 に示す．センサ素子の詳細なモデル番号を表 2.2 に示す．全てのセンサは世界中で使用されている人気のある素子であり，それらの性能は各メーカーによって保証されている．

SenStick ハードウェアの新規性は，マルチセンサ，BLE 通信機能とフラッシュメモリの組み合わせにある．ほとんどすべての BLE ベースのデバイスはスマートフォンに接続されていることを前提としているため，センサデータを記録するための大容量のメモリを搭載していない．BLE の通信帯域は，高いサンプリング周波数で複数のセンサから取得されたすべてのセンサデータを送信するには不十分である．そのため，大容量のフラッシュメモリを搭載した新しい BLE ベースのセンシングボードを設計した．これにより，ユーザーはすべてのセンシングデータを正確に記録できる．その結果，スマートフォンとの接続がなくてもスタンドアローンで動作することが可能である．

SenStick ハードウェア内のファームウェアは，前述の要件を考慮して設計および開発されている．ファームウェアによって提供される SenStick の汎用属性（GATT）プロファイルを，図 2.5 に示す．SenStick の GATT サーバーは，3 つのサービス（SenStick コントロールサービス，メタデータ読み取りサービス，センササービス）を提供する．SenStick コントロールサービスは，センシングの開始と停止などの操作指示を提供する．メタデータ読み取りサービスは，ログデータのメタ

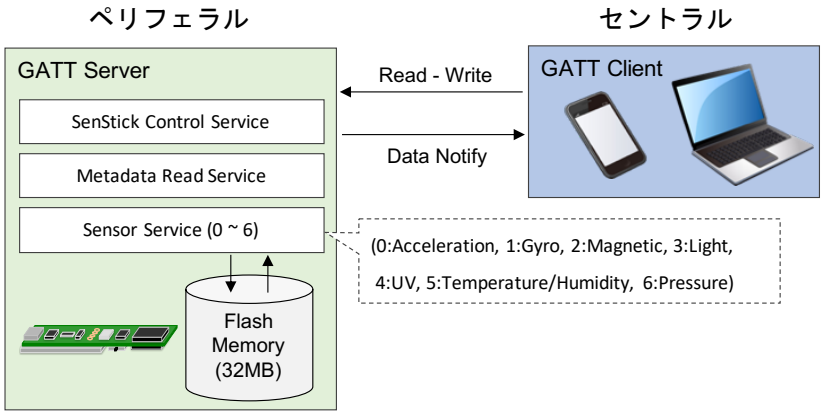


図 2.5: SenStick の GATT (Generic attribute profile) サービス

表 2.3: センサごとの割り当てメモリ容量および記録可能な最大サンプル数

	割り当てメモリ容量	サンプル数
加速度	10.2 M バイト	1.7 M サンプル
ジャイロ	10.2 M バイト	1.7 M サンプル
地磁気	10.2 M バイト	1.7 M サンプル
照度	340 K バイト	170 K サンプル
紫外線 (UV)	340 K バイト	170 K サンプル
温度・湿度	170 K バイト	170 K サンプル
気圧	170 K バイト	170 K サンプル

情報を読み取る機能を提供する．センササービスは，各センサ（0 : 加速度，1 : ジャイロ，2 : 磁気，3 : 光，4 : UV，5 : 温度/湿度，6 : 圧力）のセンシングに関わるパラメータを設定する機能を提供する．また，リアルタイムセンサデータ読み取り機能，およびログデータ読み取り機能という 2 つのデータ読み取り機能を提供する．

基本的に，すべてのセンサデータは，BLE 通信を通じてリアルタイムで受信側のアプリケーションに送信される．センサデータの量が BLE の伝送容量を超えると，センサデータの一部がドロップされる．したがって，リアルタイムセンサデータ読み取り機能は，ステータスの監視や，正確なデータを必要としないインタラクティブアプリケーションに適している．例えば，歯磨きの動作を検出して，磨いた回数

をスマートフォンアプリケーション上でリアルタイムに可視化するなどの用途で活用できる．正確なデータ取得のためには，内部のフラッシュメモリに保存されたセンサデータを取り出すことができるログデータ読み取り機能が適している．

表 2.3 に示すように，フラッシュメモリの固定容量が各センサに割り当てられている．サンプル数は，割り当てられた容量で記録できるサンプルの総数を示している．

SenStick は，1 つのセンサのサンプル数が割り当てられたストレージ容量を超えると，ロギングを停止する．加速度，ジャイロ，および磁気センサーの最大記録時間は，サンプリング間隔 10 ミリ秒で 4 時間 40 分（サンプリング周波数：100Hz），サンプリング期間 30 ミリ秒で 14 時間 10 分（サンプリング周波数：33Hz）である．これより長時間のデータロギングを実現したい場合は，受信用ソフトウェア側で定期的にログデータを取り出し，逐次メモリをリセットするなどの工夫が必要となる．他のセンサの最大記録時間は，サンプリング周期 200 ミリ秒で 9 時間 26 分である（サンプリング周波数：5Hz）．

2.4.2 SenStick ソフトウェア

SenStick ソフトウェアは，iOS および Android 用のモバイルアプリケーションと，Windows, Mac, Linux の PC でのデータ受信用に，node.js で記述されたライブラリを提供している．基本的に，それらのソフトウェアは，SenStick ハードウェアの操作機能，パラメータ設定機能，リアルタイムデータ受信機能が備わっている．iOS アプリケーションの追加機能は，ファームウェア更新機能である．node.js のライブラリは，ブラウザベースのエディタを通じてパズル感覚で簡単にプログラミングが可能な Node-Red との互換性を備えており，初心者でもセンサデータの処理が簡単に行えるという利点がある．これらのソフトウェアはすべて Github 上で共有されているため，すべてのユーザは自分の目的に基づいてアプリケーションを作成できる．

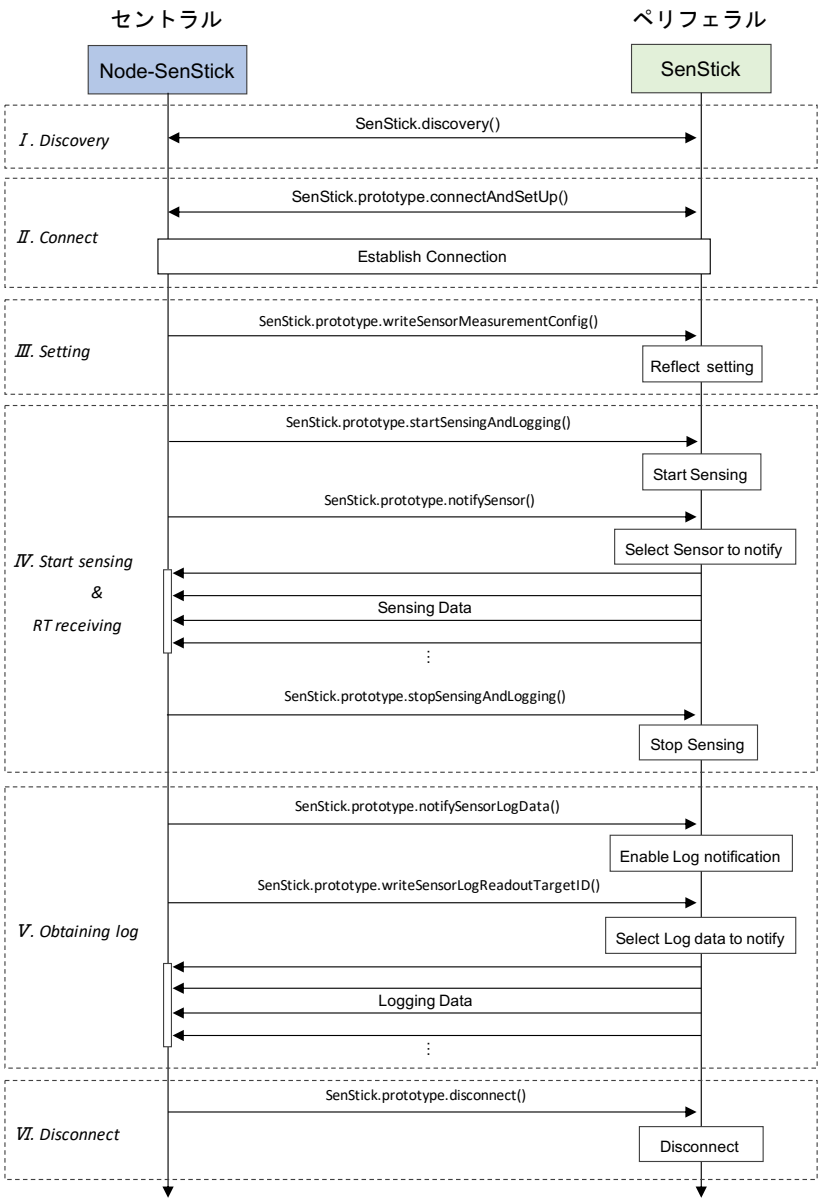


図 2.6: SenStick とデータ収集ソフトウェア（node-senstick）間の通信シーケンス

1. データ収集ソフトウェア

■通信プロトコル　ここでは，SenStick とソフトウェア間の通信プロトコルについて説明する．node.js 用に開発されたライブラリである「node-senstick」の仕様に基づいていて記述する．図 2.6 は，SenStick（ペリフェラル）と PC 端末（セント

ラル）間のプロトコルシーケンスを示している．

1. **Discovery:**
   まず，アプリケーションはライブラリの*SenStick.discovery()* メソッドを使用して周囲の *SenStick* を見つける必要がある． *SenStick.discovery()* メソッドが *SenStick* を見つけると，*SenStick* の *UUID* と *LocalName* を引数として含む *SenStick* オブジェクトのインスタンスを持つコールバック関数が呼び出される．
2. **Connect:**検出プロセスでターゲット *SenStick* が見つかった場合，アプリケーションは*SenStick.prototype.connectAndSetUp()* メソッドを呼び出してターゲット *SenStick* との接続を確立する．接続を確立した後，このメソッドはターゲット *SenStick* から操作に必要な情報を取得する．
3. **Setting:**
   接続の確立後，アプリケーションは*SenStick.prototype.writeSensorMeasurementConfig()* メソッドを使用して，各センサーの *3* つのパラメーター（操作モード，サンプリング間隔，測定範囲）を設定する． 動作モードは，各センサーの検知とロギングの検証/無効化を設定するパラメーターである．サンプリング間隔と測定範囲は，センサーの目的と用途によって変更される各センサーのパラメータである．
4. **Start sensing & Real-time data receiving:**
   SenStick.prototype.startSensingAndLogging() メソッドが呼び出されると，*SenStick* はセンシングとロギングを開始する．有効なセンサのデータのみが *BLE* を介して送信され，フラッシュメモリにも記録される．各ログの *ID* は自動的に割り当てられる．センシング中，*SenStick* はリアルタイムでセンサデータを通知する．この通知を受信するために，アプリケーションは*SenStick.prototype.notifySensor()* メソッドを呼び出し（*Sensor* は各センサー名を表す），センサーの通知をアクティブにする．センサデータの通知を受信すると，*sensorChange* イベントが発行される．*SenStick.prototype.stopSensingAndLogging()* メソッドが呼び出されると，*SenStick* はセンシングとロギングを停止する．

5. **Log data reading:**
   フラッシュメモリに記録されたログデータをアプリケーションに読み込むには，アプリケーションは*SenStick.prototype.notifySensorLogData()* メソッドを呼び出し（センサーは各センサー名を表す），ログデータ通知を有効にする．アプリケーションが*SenStick.prototype.writeSensorLogReadoutTargetID()* メソッドを呼び出して有効なログ *ID* を指定すると，*SenStick* はログに記録されたセンサーデータの通知を順次開始する． ログデータの通知を受信するには，*sensorLogDataReceived* イベントが発行される．この通知をアプリケーションが受信することにより，メモリに記録されたログデータを取得することができる．
6. **Disconnect:**
   データの取得が完了すると，*SenStick.prototype.disconnect()* メソッドが呼び出され，*SenStick* の接続が解放される． そして，*SenStick* は再びアドバタイズを開始する．

■**ユーザインタフェース**　iOS バージョンに基づいたアプリケーションのユーザインタフェースについて説明する．図 2.7 は，SenStick の ios 版モバイルアプリケーションを示している． 図 2.7 に示すように，ユーザはデバイスリストからターゲット SenStick を選択する．選択動作に伴って，アプリケーションはメインビューを表示する．メインビューでは，ユーザはセンシング対象とするセンサの ON/OFF を選択できる． データを即座に監視することを目的として，BLE 通信経由で受信したすべてのセンサデータがリアルタイムでグラフビューに表示される． アプリケーションの GUI を通じて，サンプリング周波数，検出範囲など，各センサのパラメータ設定を自由に変更できる．また，ファームウェアの更新機能も備えている．ハードウェアの拡張の伴って，ファームウェアを更新する場合には，この機能が活用できる．SenStick のフラッシュメモリに保存されたセンサデータは，ログリストとして表示され，CSV 形式でスマートフォンにダウンロードできる．

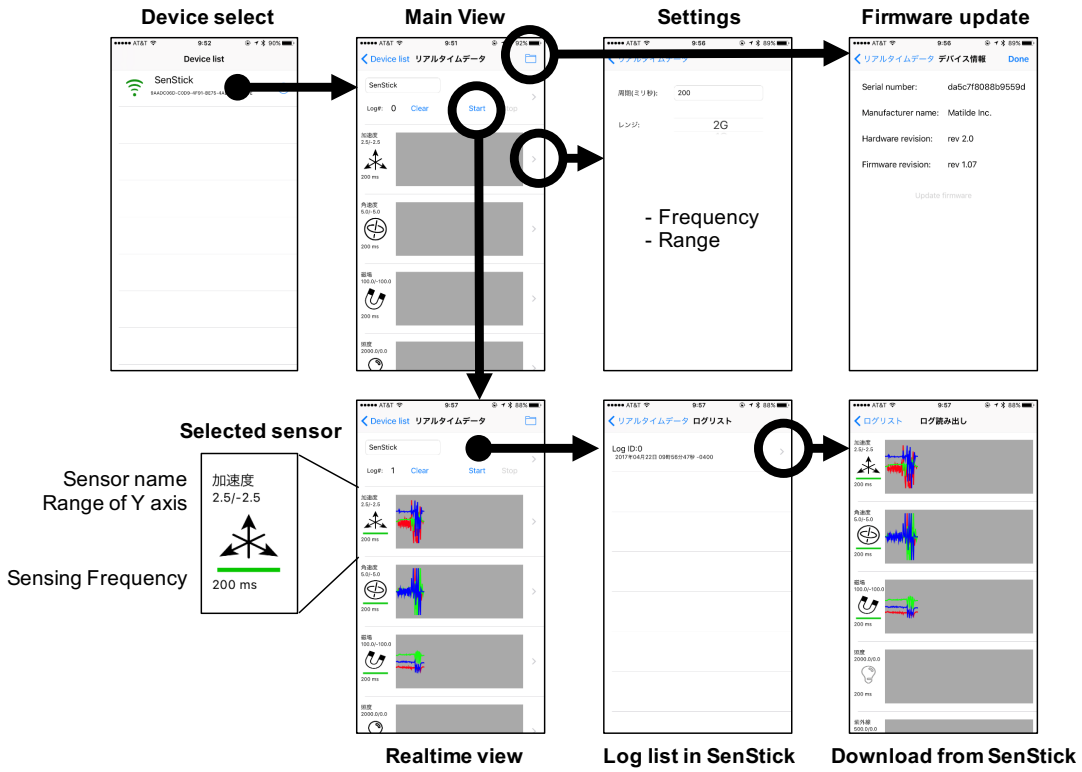


図 2.7: SenStick iOS アプリケーションの UI フロー

2. 特徴量抽出ソフトウェア

センサデータ収集の次のステップは，データ分析である．一般的に，IoT センサデータの分析には機械学習手法が用いられる．近年，機械学習の注目によって，様々な有用な機械学習ライブラリ [29–32] が提供されている．しかしながら，加速度などのセンサデータをベースとした行動認識の場合．画像とは異なり，生のセンサデータをそのまま機械学習にかけても認識制度が向上しないという欠点がある．そのため，加速度をはじめとした時系列センサデータの分析において，機械学習モデルの認識精度を向上させるためには，特徴量のエンジニアリングが非常に重要である．しかしながら，特徴量エンジニアリングでは，データ分析者の長年の経験や勘といった暗黙的な知見も重要になってくるため，初心者が有用な特徴量を見つけ出すのが困難であるという問題がある．そこで SenStick ソフトウェアでは，この背景に基づいて，行動認識の分野の既存研究 [33–35] の調査に基づいて，有効性が

確認されている複数の特徴量抽出関数をリストアップし，ライブラリとして提供している．SenStick ソフトウェアでサポートしている特徴量抽出関数を 2.4 に示す．これによって，初心者のユーザであっても，素早くデータ分析を行うことができ，トライアンドエラーを繰り返すことで有効な機械学習モデルが構築可能となる．つまり，モノを素早く IoT デバイス化し，データを収集し，素早くデータを分析をして，活用することが可能になる．

表 2.4: SenStick プラットフォームで提供する特徴量抽出関数の一覧

Function	**Description**	**Formula**	**Type**
mean(s)	Arithmetic mean	$\bar{s} = \frac{1}{N}\sum_{i=1}^{N} s_i$	T, F
std(s)	Standard deviation	$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i - \bar{s})^2}$	T, F
mad(s)	Median absolute deviation	$median_i(\mid s_i - median_j(s_j) \mid)$	T, F
max(s)	Largest values in array	$max_i(s_i)$	T, F
min(s)	Smallest value in array	$min_i(s_i)$	T, F
energy(s)	Average sum of the square	$\frac{1}{N}\sum_{i=1}^{N} s_i^2$	T, F
sma(s_1, s_2, s_3)	Signal magnitude area	$\frac{1}{3}\sum_{i=1}^{3}\sum_{j=1}^{N} \lvert s_{i,j} \rvert$	T, F
entropy(s)	Signal entropy	$\sum_{i=1}^{N}(c_i \log(c_i)), c_i = s_i / \sum_{j=1}^{N} s_j$	T, F
iqr(s)	Interquartile range	$Q3(s) - Q1(s)$	T, F
autoregresion(s)	4th order Burg autoregression coefficients	$a = arburg(s, 4), a \in \mathbb{R}^4$	T
correlation(s_1, s_2)	Pearson Correlation coefficient	$C_{1,2}/\sqrt{C_{1,1}C_{2,2}}, C = cov(s_1, s_2)$	T
angle(s_1, s_2, s_3, v)	Angle between signal mean and vector	$\tan^{-1}(\parallel [\bar{s}_1, \bar{s}_2, \bar{s}_3] \times v \parallel, [\bar{s}_1, \bar{s}_2, \bar{s}_3] \cdot v)$	T
range(s)	Range of smallest value and largest value	$max_i(s_i) - mix_i(s_i)$	T
rms(s)	Root square means	$\sqrt{\frac{1}{N}(s_1^2 + s_2^2 + \cdots + s_N^2)}$	T
skewness(s)	Frequency signal skewness	$E[(\frac{s-\bar{s}}{\sigma})^3]$	F
kurtosis(s)	Frequency signal kurtosis	$E[(s-\bar{s})^4]/E[(s-\bar{s})^2]^2$	F
maxFreqInd(s)	Largest frequency component	$argmax_i(s_i)$	F
meanFreq(s)	Frequency signal weighted average	$\sum_{i=1}^{N}(is_i)/\sum_{j=1}^{N} s_j$	F
energyBand(s, a, b)	Spectral energy of a frequency band [a, b]	$\frac{1}{a-b+1}\sum_{i=a}^{b} s_i^2$	F

N: Signal vector length, Q: Quartile, T: Time domain features, F: Frequency domain features, s: Sensor data divided for each time window.

2.4.3　SenStick 3D ケースデータ

モノの IoT デバイス化を行うためには，SenStick を取り付けるためのアタッチメントケースが必要である．我々は，3D プリンターによってプリントアウト可能な SenStick の基本的な 3D ケースを設計した．図 2.8 に基本的な 3D ケースに入った SenStick を様々なモノに取り付けた例を示す．このように，SenStick は，ケースを含めても非常に小型であるため，多種多様なモノに取り付けることが可能である．また，犬や猫にセンサを装着する場合，体重の 10 %以下である必要があるなどの倫理規範が存在するが，SenStick は重さが 3g と非常に軽量であり，スタンドアローンでデータロギングが可能であることから，動物の行動センシングといった用途にも活用可能である．しかし，箸やペンなど小さなモノの場合，センサを貼り付けるだけでは，少し動作の邪魔になり，使いにくくなってしまうといった場合も考えられる．そのため，SenStick の 3D ケースデータは，CAD ソフトウェアによって編集することで自由に拡張することが可能であり，ユーザーは箸，歯ブラシなどの形状をした派生ケースを設計することができる． 実際，我々の研究室では，図 2.9 に示すように，いくつかの派生ケースの設計に成功している．これらのケースデータはすべて，Github を通じて共有されている． 基本型の 3D ケースのおかげで，各派生ケースは数分で設計することができる．たとえば，メガネの SenStick ケースは，SenStick の基本ケースと i.Design Studio によって配布されたメガネの

図 2.8: 様々なモノに対する基本ケースの装着例

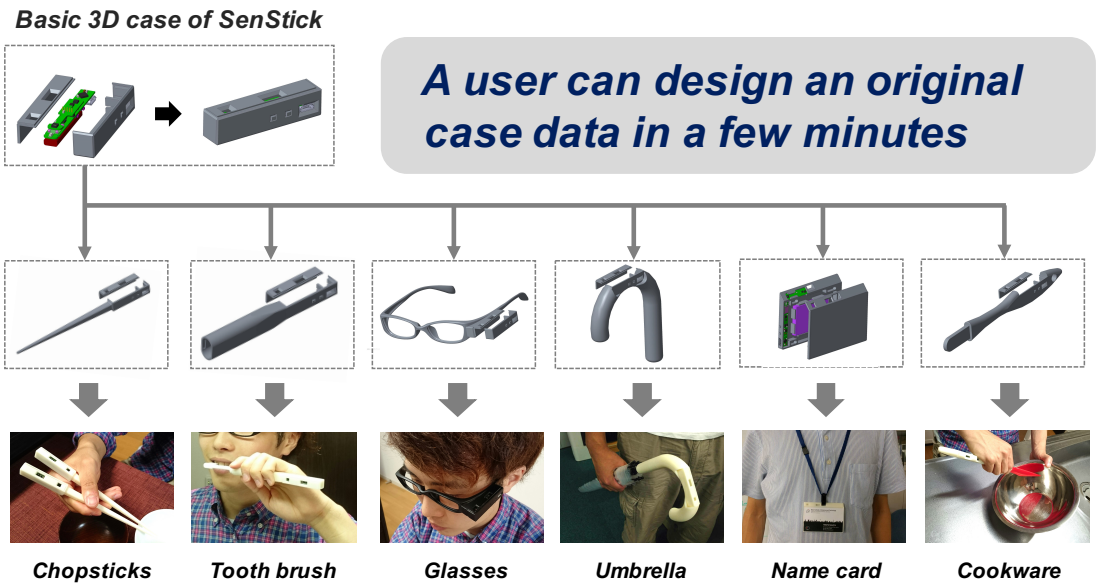


図 2.9: 多様な SenStick3D ケースの例

オープン 3D データを組み合わせて作成されている．近年の 3D プリンターの普及によって，3D モデリング用のツールとリファレンスが強化されており，すべてのユーザーは自分の希望する SenStick ケースを自分で設計・印刷し，好みのオブジェクトに SenStick を取り付けることが可能である．

2.5 SenStick のケーススタディ

超小型のマルチセンサボードである SenStick は，日常生活のさまざまな小さなモノに埋め込むことが可能である．このセクションでは，SenStick を利用したモノの IoT デバイス化のケーススタディを紹介する．はじめに，箸の IoT デバイス化の事例を用いて，前節で説明した SenStick プラットフォームのハードウェア，ソフトウェア，3D データという 3 要素がデバイス開発，データ収集という過程でどのように機能するのか提示する．次に，ベルト型のウェアラブルデバイスである Waiston Belt の事例を示し，SenStick が IoT 技術を活用した日常行動認識サービスの実現に対して十分に寄与することを示す．最後に，授業によって得られた SenStick の応用事例を示し，SenStick が IoT 教育の用途でも有用であることを示す．

2.5.1 箸の IoT デバイス化

近年，ウェアラブル技術の発展により，日常生活における運動や睡眠の質を分析し，それらの質をユーザに対してフィードバックすることが可能になりつつある．しかしながら，食事に関しては，食習慣や食事行動の質を計測する技術が実現されていない．食事も，運動や睡眠と共に，生活習慣における重要な 3 大要素として位置付けられているため，食事に関わる行動習慣を計測することは，生活習慣病の予防やユーザの健康を促進する上でも非常に重要であると考える．そこで我々は，多くの場合，日本人が箸で食事をすることに着目し，箸から得られたセンサデータを分析することで，「どのように食べたのか」「何を食べたのか」など，ユーザの食習慣や食事行動を監視する目指す．この目的を達成するために，箸型の SenStick の開発を試みた．まず，SenStick プラットフォームを活用した箸の IoT デバイス化プロセスを図 2.10 示す．このように，SenStick の基本型 3D ケースを CAD ソフト上で拡張することによって，簡単に箸型のケースを作成可能である．SenStick を活用した成果物と既存のプロトタイピングで開発した箸型 IoT デバイスの比較を図 2.11 に示す．このように，SenStick プラットフォームを活用した場合と比較して，不恰好なものとなっている．また，デバイスからセンサデータを収集するために，ソフトウェアを開発する必要があるが，SenStick プラットフォームを活用することで，その手間を省いて，ユーザは素早くデータ収集を行える．図 2.12 に，実際の食

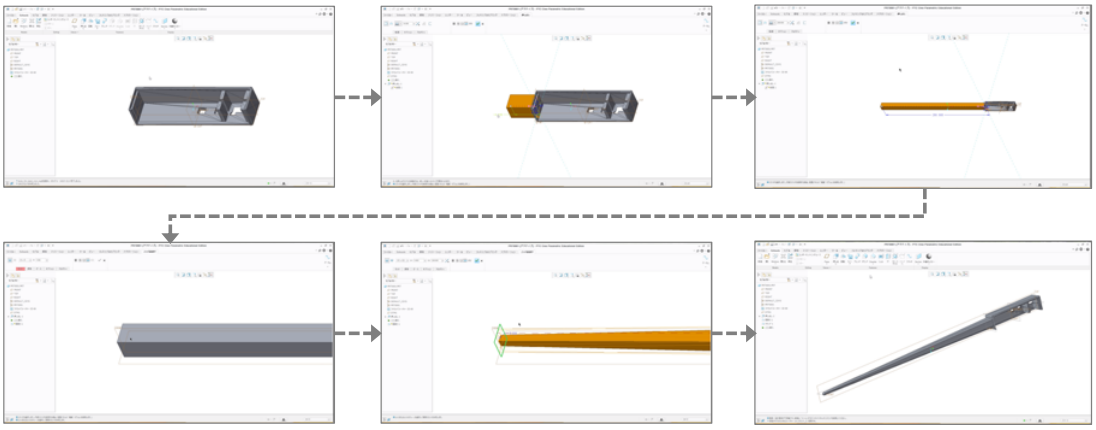


図 2.10: 箸型の SenStick 3D ケース作成過程

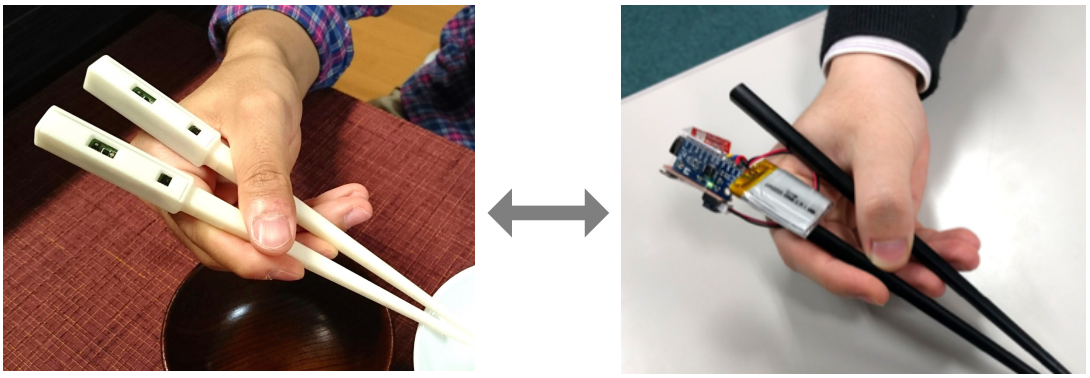


図 2.11: 従来のプロトタイピングによってら得られた成果物との比較

事行動における 2 つの箸から得られたセンサデータ（加速度・ジャイロ）の波形を示している．このように，開く，閉じる，持ち上げる，戻すといった食事行動における箸の基本動作に関して，センサデータの特徴的な波形パターンが出ていることが確認できる．そのため，DTW などのパターン認識手法を活用することで，食事中に食べ物を口に運んだ回数のカウントすることが可能であり，口に運んぶ頻度から，早食いを検知して，ゆっくり食べるようユーザに対して促すなどのアプリケーションが実現可能である．

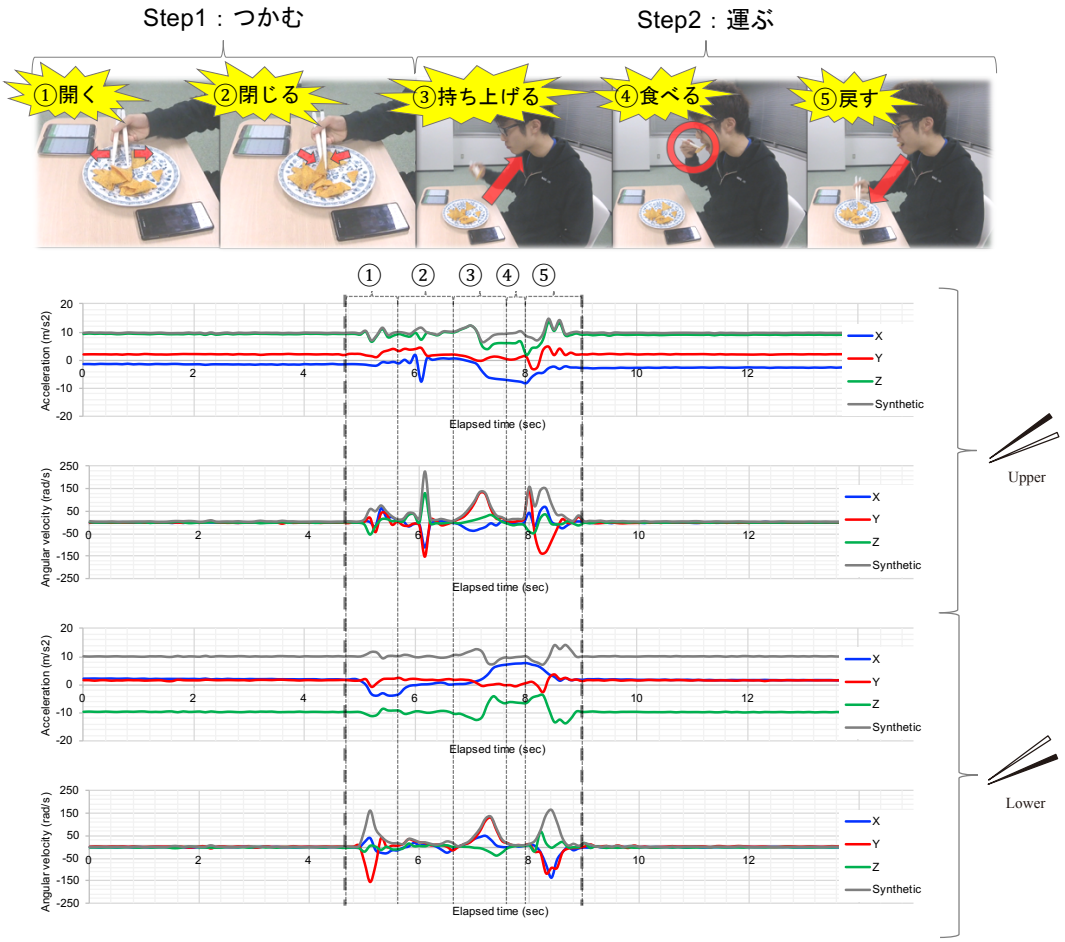


図 2.12: 食事動作におけるセンサデータ波形

2.5.2 ベルトの IoT デバイス化：Waiston Belt

糖尿病，心臓病，動脈硬化，脳卒中をはじめとした生活習慣病の進行中の流行は，世界中で緊急の問題と見なされている [36]．これらの生活習慣病の主な要因は，運動不足，悪い姿勢，過食，スナックなどの不健康な生活習慣である [37]．McKinsey & Company のレポートによると，人々の生活習慣が改善しない場合，成人人口のほぼ半数が 2030 年までに過体重または肥満になることが予測されている [38]．生活習慣病は，主に人々の生活習慣によって引き起こされるため，予防可能である．我々は，ウェアラブルコンピューティングテクノロジーを使用して，悪い生活習慣

から健康的な生活習慣への変化を促進することにより，この問題を解決できるのではないかと考えている．そのため，我々の研究グループでは，ユーザの腹囲測定や日常行動の計測を行うベルト型ウェアラブルデバイスである Waiston Belt の開発を進めている．Waiston Belt のユースケースでは，オフィスワーカーが毎日，装着するベルトを IoT デバイス化することを考えているため，最終的なデバイスは小型であることが望まれる．しかしながら，以前のバージョン Waiston Belt2 は，マイコンとして Arduino を使用しているためデバイスサイズが大きく，常用するためには，ベルトデバイスに加えてメインの制御ボードを持ち運ぶ必要があるなど制約があった．そこで，本研究では，SenStick を活用して，Waiston Belt のバージョンアップに取り組んだ．

悪い生活習慣から健康的な生活習慣への行動変容を促進するためには，センシング機能だけでなくアクチュエーション機能が必要となる．そのため，WaistonBelt X では，(1) 腹囲の測定，(2) 日常生活行動および座り姿勢の認識といセンシング機能だけでなく， (3) 悪習慣矯正のためのコンテキストアウェア振動介入というアクチュエーション機能も提供している．しかし，SenStick のメインボードはセンシング機能しか有していない．そこで，我々は SenStick の I2C 対応の外部拡張端

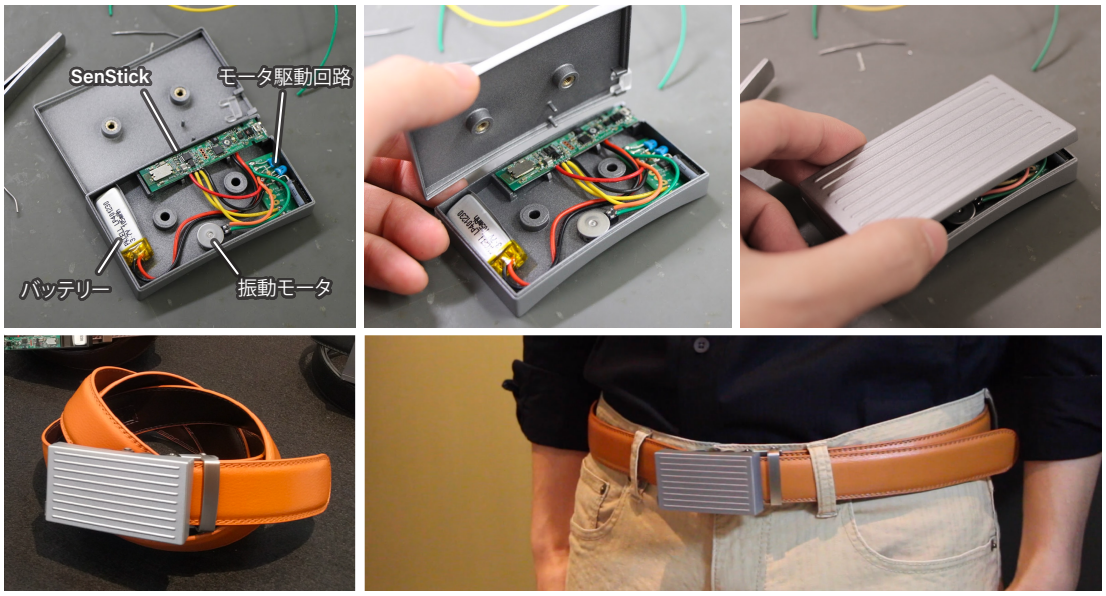


図 2.13: SenStick を用いた WaistonBelt X の外観

子を活用し，メインボードに振動モータを追加することによって，(3) のアクチュエーション機能を実現した．図 2.13 に *WaistonBelt X* の外観を示す．このように，*WaistonBelt X* は，一般的なベルトバックルに取り付けられるように設計されている．SenStick を活用することで，センサ，アクチュエータ，通信機能，メモリ，バッテリー (110 mAh) など必要なすべての機能がバックルサイズのケースに収まっている．SenStick は，10 mAh のバッテリーにつき約 1 時間動作できるため，110 mAh のバッテリーによって，約 11 時間連続で動作することが可能である．動作時間はバッテリーの容量に依存するため，より大きな容量のバッテリーを取り付けることで動作時間を増やすことができる．

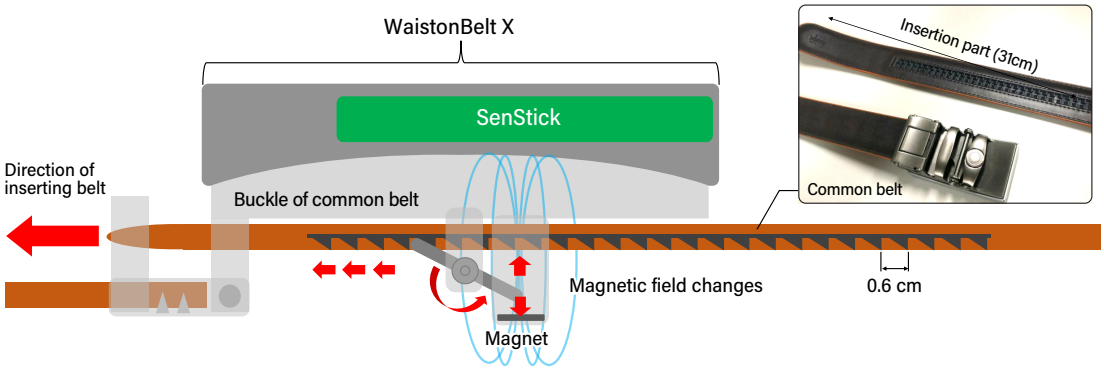
以下では，SenStick をベースに開発された Waiston Belt X で提供している 3 つの主要機能について示す．

ベルトによる腹囲測定機構

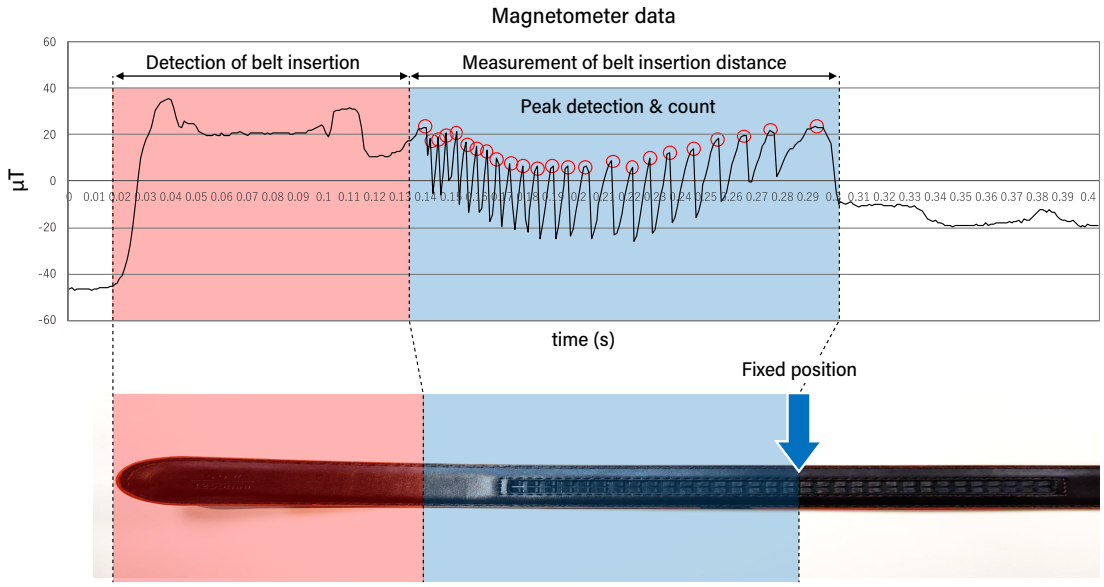
■提案手法　WAISTON Belt では，ベルトの全長が既知であることを前提として，ベルトの挿入距離を測定することにより，お腹周り（ベルトで形成される環の内周）の測定を算出している．これまでのプロトタイプ〜 [39, 40] では，ベルトの挿入距離を測定する手法として，バックルの内部にロータリーエンコーダを設置し，ベルト挿入時の回転数を測定する手法を採用していた．しかしながら，バックルの内部にロータリーエンコーダを設置する場合，デバイスのサイズが大きくなってしまう問題や，ベルトを締めるたびに磨耗が生じてしまうといった問題が存在した．そこで，新たに高精度な測定を可能としつつも，より小型化可能な腹囲測定手法の検討を行った．具体的には，図 2.14（a）に示すような，既存の無段階式ベルトに取り付けられているベルト固定機構の仕組みに着目した．無段階式ベルトには，0.6cm 間隔でベルトを固定するための凸加工が施されており，ベルトの挿入と同期してピボット式金属ブレードが上下に繰り返し移動するという特徴がある．そして，ベルトを挿入する際のブレードの上下運動をバックル内部に設置されている SenStick の磁気センサによって計測し，図 2.14（b）に示すように，磁気センサから得られた時系列データのピークをカウントすることでベルトの挿入距離を計算する．

■評価実験

実験設定　提案する手法は，地磁気センサを用いベルト挿入時の磁場の変化をもとに挿入距離を推定し，ベルトの総延長との差を算出することに基づいて腹囲を測定するものである．本実験では，ベルト挿入時の挿入距離の推定精度に関して検証する．実験では，WAISTON Belt を用いて挿入距離の推定値を取得し，距離測定器



(a) ベルト挿入時の磁場変化のメカニズム



(b) ベルトを挿入することにより生成される磁場測定データ

図 2.14: 腹囲測定のメカニズム

表 2.5: ベルト挿入距離測定制度実験結果

絶対誤差 [cm]		相対誤差 [%]	
平均	標準偏差	平均	標準誤差
0.93	1.01	3.00	3.24

（巻き尺）を用いて挿入距離の正解値を測定し，その 2 値の平均誤差を算出し評価する．なお，評価にあたっては，試行回数を 100 回とし，各試行の挿入距離は最もベルトを挿入した場合を想定し，31 cm （図 2.14（a）を参照）とした．

実験結果　表 2.5 に磁気センサを用いたベルト挿入距離測定実験の結果を示す．31cm 挿入時の絶対誤差の平均および標準誤差は，それぞれ 0.93cm，1.01cm となっており，ベルト状の凹凸のピークが 0.6cm（これは本手法の分解能に相当）であることを考慮すると，誤差およびばらつき共に，良好な結果といえる．また，相対誤差が 3 %以下となっていることからも，地磁気センサを用いたベルト挿入距離測定機構は，腹囲測定において十分な精度が得られるといえる．

日常生活行動および座位姿勢の認識

■提案手法　*WaistonBelt X* は，3 軸の加速度計とジャイロスコープを使用して，ユーザの基本的な日常生活行動を認識する．データ前処理プロセスでは，SenStick から取得される加速度（Acc.XYZ），ジャイロ（Gyro.XYZ）信号に対し，メディアンフィルタと 20Hz の 3 次バタワース・ローパスフィルタをかけてスパイクノイズなどのノイズ除去を行う．身体動作の 99 %が 15Hz 以下に含まれており，人間の行動を捕捉するのに十分であることから，20Hz のフィルタを選定している [41]．ノイズ除去された加速度信号（Acc.XYZ）には重力および体動成分が混在しているため，0.3Hz のバタワース・ローパスフィルタをかけ，低周波数帯の重力成分（GravityAcc.XYZ）とそれ以外の体動成分（BodyAcc.XYZ）に分離させる．この分離によって得られた重力成分は身体の姿勢判定などに有効的であり，体動成分は身体の動作や揺れなどを表すものとして考えられる．さらに，時間微分を計算することによって取得されるジャーク信号（BodyAccJerk.XYZ）と 3 軸信

号からユークリッド距離を算出することで取得される合成信号（GravityAccMag，BodyAccMag，BodyAccJerkMag）を生成する．ジャイロ信号も同様に，体動成分の角速度（BodyAngularSpeed-XYZ）に加えて，角加速度（BodyAngularAcc-XYZ）および合成信号（BodyAngularSpeedMag, BodyAngularAccMag）を生成する．次に，上記の波形の中から GravityAcc.XYZ，GravityAccMag，BodyAngularSpeed-XYZ を除く，7 つの波形に対して，高速フーリエ変換（FFT）を適用し，周波数領域にマッピングした信号を生成する．結果として，このプロセスを通じて，表 2.6 に示す， 時間ドメイン 10 個，周波数ドメイン 7 個の合計 17 種類の信号が抽出される．

特徴量抽出プロセスでは，前処理プロセスで得られた 17 個の信号を 1.56 秒のタイムウィンドウ幅（128 サンプル），50% のオーバーラップで分割する．運動認識に関する先行研究 [42] によって述べられているように，1 秒以上のタイムウィンドウによって取り出されたデータには各運動の特徴を表すサンプルが十分に含まれていること，高い認識精度が得られていることからタイムウィンドウは 1.56 秒とする．そして，各ウィンドウから特徴量を抽出する．具体的には，1.56 秒のタイムウィンドウで区切られた信号に対して，表 2.4 に示す特徴量抽出関数を適用し，時

表 2.6: 加速度・ジャイロセンサから取得した時間および周波数ドメインの信号

名前	種類 (T:Time, F:Freq.)
Body Acc	T, F
Gravity Acc	T
Body Acc Jerk	T, F
Body Angular Speed	T, F
Body Angular Acc	T
Body Acc Magnitude	T, F
Gravity Acc Mag	T
Body Acc Jerk Mag	T, F
Body Angular Speed Mag	T, F
Body Angular Acc Mag	T, F

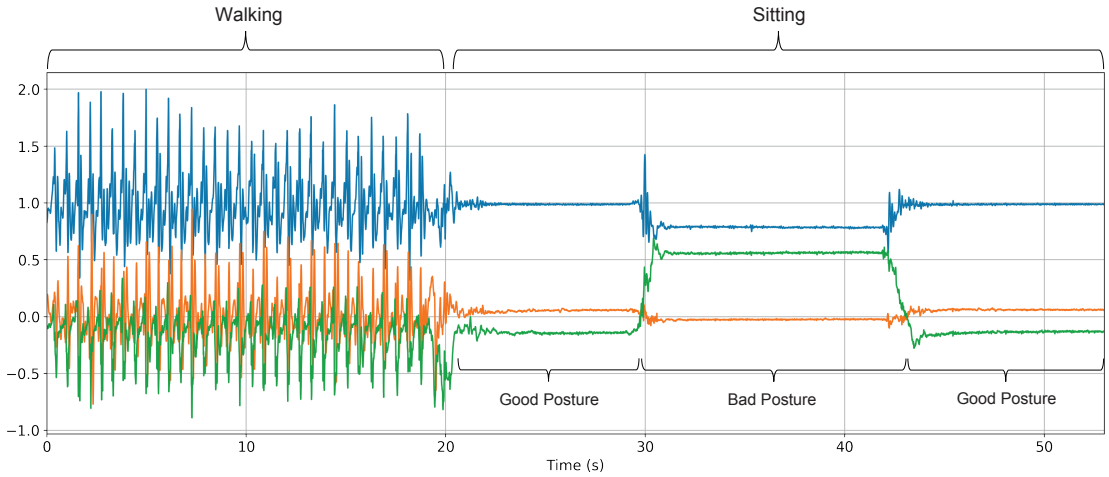


図 2.15: 歩行時および着座時の加速度データ波形

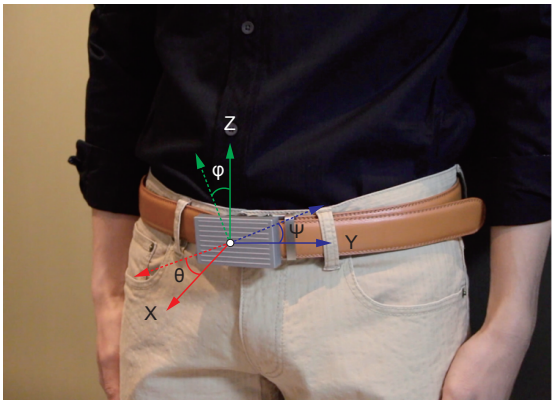


図 2.16: WaistonBelt X の姿勢角

間ドメイン特徴量（Time domain features）と周波数ドメイン特徴量（Frequency domain features）を算出する．結果として，1.56 秒の各ウインドウデータから，561 個の特徴量ベクトルを抽出する．抽出された特徴量をそれぞれ標準化し，特徴量ベクトルを機械学習アルゴリズムに学習させて，日常行動認識モデルを生成する．機械学習アルゴリズムは，既存研究 [35] によって有効性が確認されているランダムフォレスト手法を採用する．*WaistonBelt X* は，上記のプロセスで生成された日常行動認識モデルを使用して，ユーザの基本的な日常生活行動を監視する．

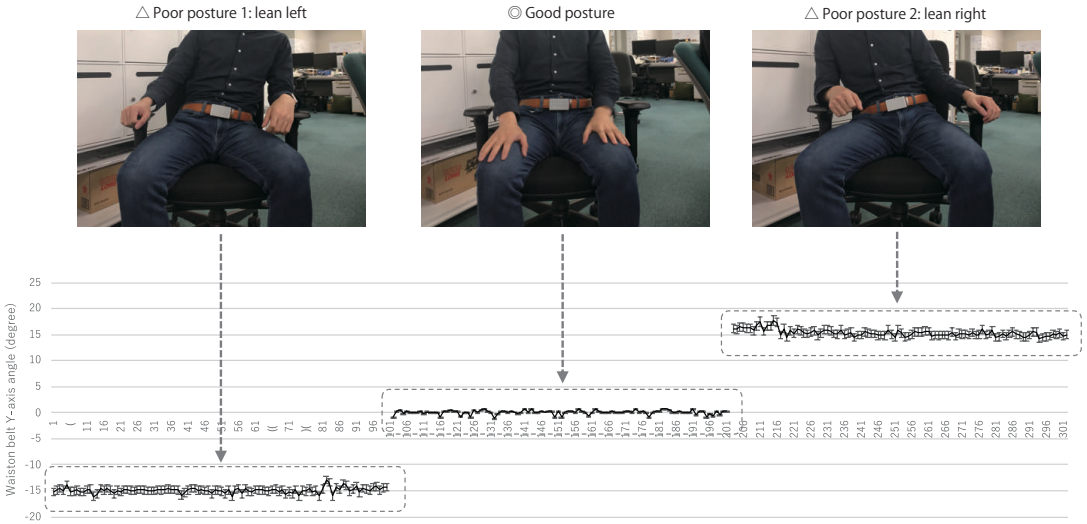


図 2.17: Y 軸角度と座位姿勢の関係

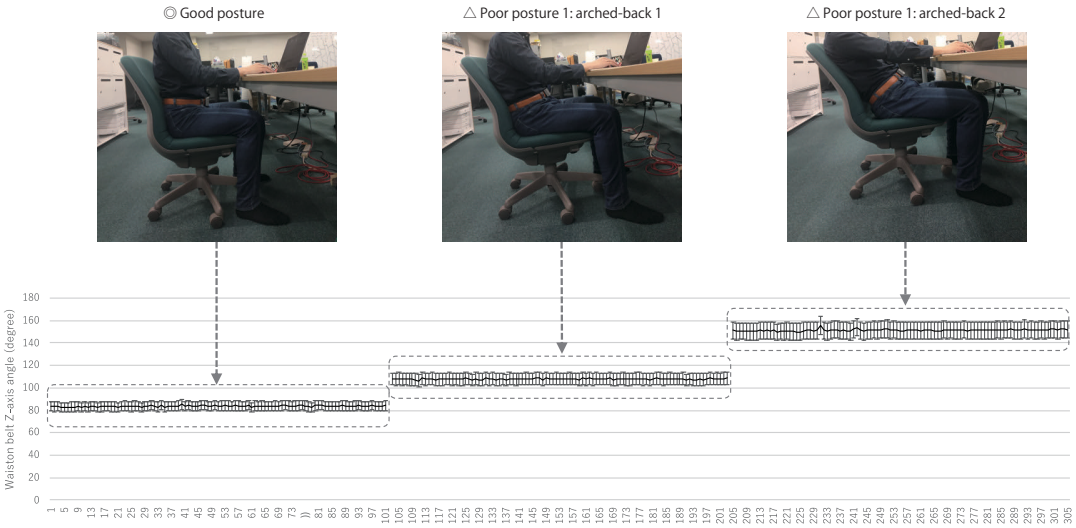


図 2.18: Z 軸角度と座位姿勢の関係

$$\theta = \tan^{-1}\left(\frac{Acc_X}{\sqrt{Acc_Y^2 + Acc_Z^2}}\right), \psi = \tan^{-1}\left(\frac{Acc_Y}{\sqrt{Acc_X^2 + Acc_Z^2}}\right), \phi = \tan^{-1}\left(\frac{\sqrt{Acc_X^2 + Acc_Y^2}}{Acc_Z^2}\right) \tag{2.51}$$

2.15 は，ユーザの歩行時および着座時における *WaistonBelt X* から取得した生の加速度データ信号波形を示す．まず，*WaistonBelt X* は，上記の基本的な日常生活認識プロセスによって，着用者が歩いているか座っているかなどのコンテキストを認識する．その後，*WaistonBelt X* は，着用者が座っていることを認識し，座位姿勢が良いか悪いかを判断する．*WaistonBelt X* では，座位姿勢が腰周りの角度と密接に関係していることに着目し，ベルトの姿勢角からユーザの姿勢の良し悪しを判定する．具体的には，加速度センサの値から 2.16 に示す 3 つの姿勢角を計算し，以下の 2 つのルールに基づいて判断する．(a) 左右の傾きが水平となっている（Y 軸の回転 $\psi \simeq 0$ 度：2.17 を参照），(b) 前後の傾きはほぼ垂直となっている（Z 軸の回転 $phi \simeq 80 \sim 90$ 度：2.18 を参照）．

■評価実験

実験設定　*WaistonBelt X* の有効性を評価するために，腰だけでなく，手首，胸，足首など，複数の体位にセンサを装着した場合の認識性能を比較する．基本的な日常行動認識で対象とする行動ラベルは，(1) 寝る，(2) 座る，(3) 立つ，(4) 歩く，(5) 階段を降りる，(6) 階段を上がる，および (7) 走るを設定する．そして，17 人の被験者（性別：男性 13 人，女性 4 人，年齢：23.4 ± 1.0，身長：169.6 ± 6.3 cm，体重：61.2 ± 10.3 kg）によって，評価のためのセンサデータを収集した．被験者の体重と身長の関係は，2.19 に示される．各実験協力者はウエスト周りに *WaistonBelt X* を，右手首，左手首，胸，右足首，左足首に SenStick を着用する．そして，各実験協力者は大学のキャンパスで対象となる行動を実行した．結果として，合計約 250 時間の日常生活行動データセットを収集した．この実験では，収集したセンサデータセットに対して，以下の 2 つ種類の交差検証を検討した．10 フォールド交差検証では，データセットをランダムにシャッフルし，各グループが同じクラスの比率をカバーする 10 個のグループに分割される．そして，各フォールドで，9 つのグループに属するセンサーデータがトレーニングデータとして使用され，残りのグループのセンサデータをテストデータに使用する．Leave-one-person-out 交差検証では，各フォールドで 16 人がトレーニングデータとして使用され，残りの 1 人がテストデータに使用される．

実験結果　2.20（a）は，10 フォールド交差検証の日常生活行動認識結果の混同行列を示している．結果は，基本的な日常生活行動の分類アルゴリズムが非常によく

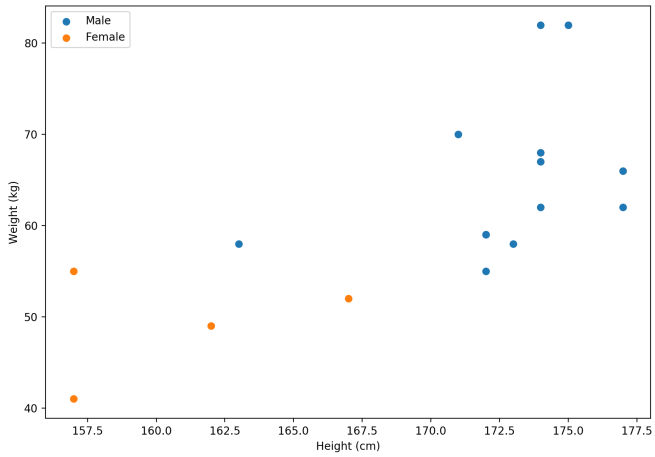
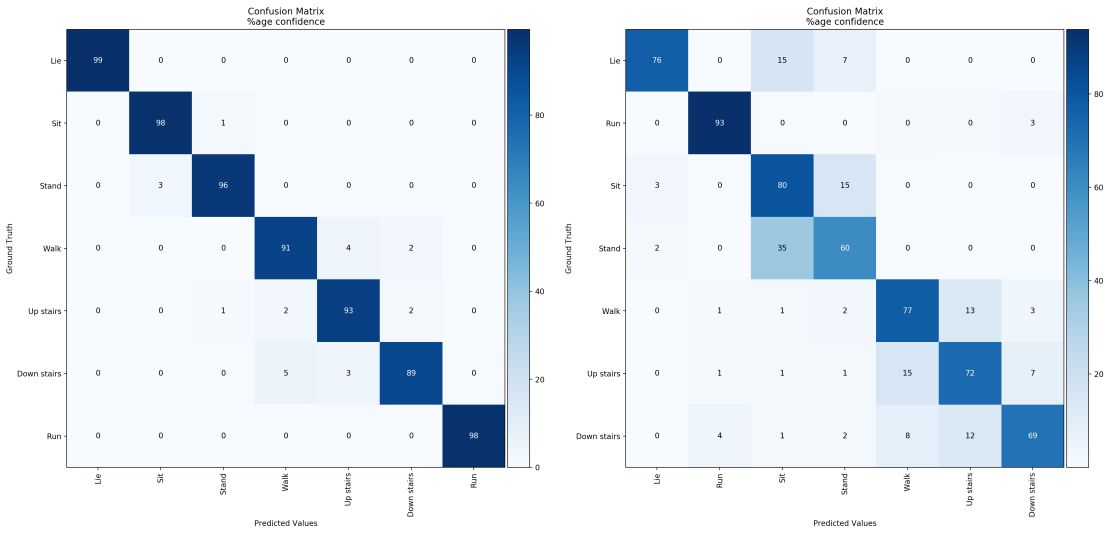


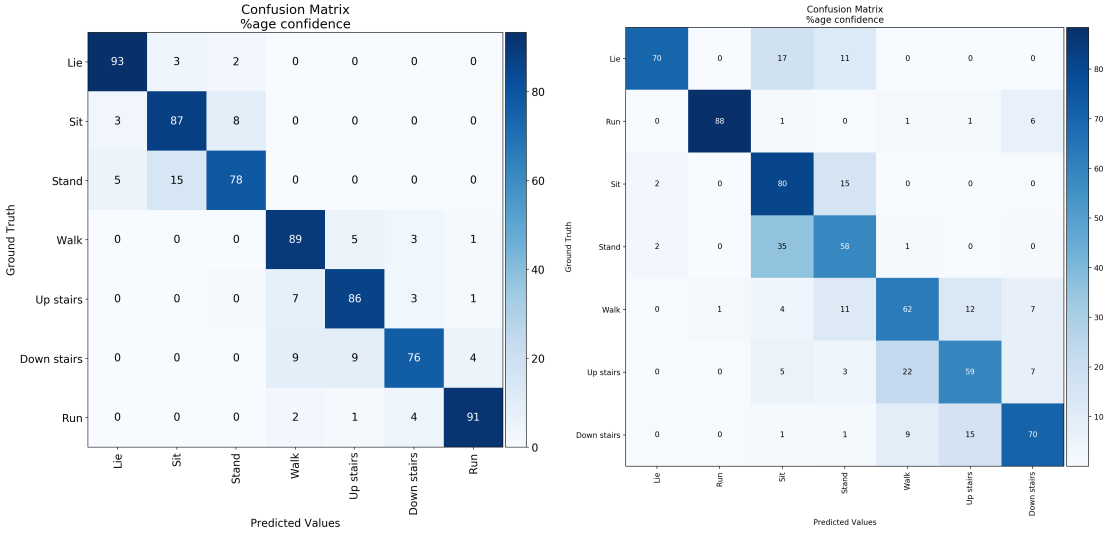
図 2.19: 実験協力者の体重と身長の関係



(a) 10-fold 交差検証結果　　(b) Leave-one-person-out 交差検証結果

図 2.20: 日常行動認識おける交差検証の結果

機能したことを示している（F1 スコアは 0.95）．2.20（b）は，1 パーソンアウト交差検証における日常生活行動認識結果の混同行列を示している．結果は，提案された方法が 0.82 の F1 スコアを達成し，座っているときと立っているとき，歩いているとき，階段を上下するときに誤分類が発生したことを示している．これは，実



(a) 男性実験協力者の結果　　(b) 女性実験協力者の結果

図 2.21: 性別の違いと認識結果の関係 (Leave-one-person-out 交差検証)

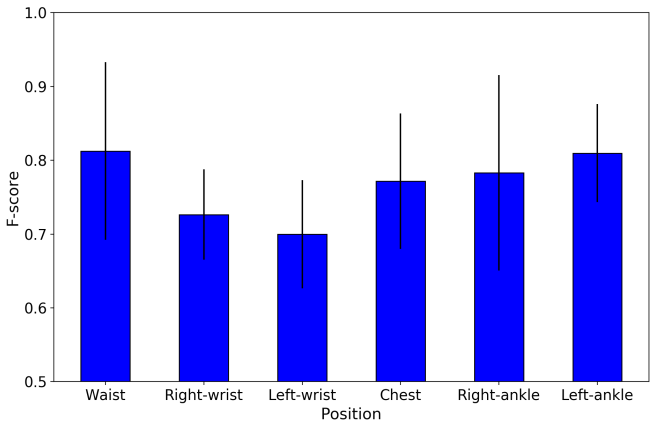


図 2.22: 各身体位置毎の認識精度の比較

験協力者間の体格や性別などの個人差が原因となって，精度が減少したと考えられる．男性と女性の体格の違いの影響を調査するために，性別ごとに集計した 1 パーソンアウト交差検証の認識結果を 2.21 に示す．男性実験協力者の結果：2.21（a）は F1 スコア 0.86 を達成し，女性実験協力者の結果：2.21（b）は 0.67 の F1 スコアを達成したことを確認した．2.19 に示されているように，女性の実験協力者の点

が他の点から遠いことが確認できる．この結果は，女性の被験者は男性の被験者よりも体格の差が大きいことを示している．そのため，体格の差が女性の結果が男性の結果よりも劣っている理由であると考えられる．この結果から，多くの被験者のデータを取得できる場合には，近しい体格の被験者を選択したグループの認識モデルを学習することが，より正確な認識結果を達成するのに有効であることが考えられる．2.22 は，各身体位置のパフォーマンス比較を示す．この結果は，ベルト装着位置である腰の位置が他の身体位置と比較して最高の精度で活動認識を達成することを示している．特に，ウェアラブルデバイスが装着される最も標準的な位置である手首よりも明らかに正確である．ベルトはオフィスワーカーが日常的に着用するアイテムであるため，*WaistonBelt X* のようなベルトを IoT デバイス化するアプローチがウェアラブルデバイスのさらなる発展に寄与することが期待される．また，*WaistonBelt X* とリストバンドタイプのウェアラブルデバイスの組み合わせにより，着用者が座りながらパソコン作業をしているのか，食事をしているなど，より高いレベルのコンテキストを認識できる可能性がある．

コンテキストアウェア振動介入

■提案手法　近年，モバイルアプリケーションの発展に伴い，様々なアプリケーションからの多くのプッシュ通知が発行され，スマートフォンやスマートウォッチの画面上に表示される．このような視覚的な通知は，ユーザにとってのある種の割り込みとなり，作業中のユーザの注意力を奪う可能性がある．そのため，少ない情報量かつ適切なタイミングで介入することがユーザの行動変容を促すために重要な鍵となる．実際，振動触覚ナビゲーションシステムが，認知的，視覚的作業負荷が高い条件下ではユーザの行動を促すために従来の視覚的な介入よりも有用であることを示した研究成果も存在する [43]．そこで，*WaistonBelt X* は，ベルト上の振動モータによるいくつかの振動パターンを使用し，ユーザの行動コンテキストに応じて触覚的な介入を行うコンテキストアウェア振動介入を実現する．このコンテキストアウェア振動介入では，上記で説明した日常行動認識の結果を介入のトリガーとして使用する．たとえば，*WaistonBelt X* がユーザが座っていることを検出すると，システムは自動的に姿勢監視モードに移行し，猫背が続いている場合など，着用者の姿勢に応じて介入する．

■評価結果

実験設定　この実験では，コンテキストアウェア振動介入の有効性を評価するために， 図 2.23 に示すように，ユーザーの姿勢改善のための行動変容シナリオに焦点を当てた．このシナリオでは，ユーザの姿勢が悪い場合にユーザに対する嫌子として振動介入を提供し，ユーザーの姿勢が良好な場合に介入を停止することによって，ユーザの姿勢改善対する影響を調査した．座位姿勢は，*WaistonBelt X* の姿勢角（図 2.16 を参照）から判定された．具体的には，姿勢角が水平（Y 軸回転 $\psi \simeq 0$ 度）および垂直（Z 軸回転 $\phi \simeq 80 \sim 90$ 度）に保たれた状態を良い座位姿勢として定義した．実際には，11 人の一般ユーザ（20 代の男性）を対象とした実験を実施した．実験中，ユーザーは *WaistonBelt X* を着用し，合計 40 分間の座りながら任意のパソコン作業を行った．そして，以下に示すように，10 分間隔で介入の有無を切り替えることにより，介入の有効性を検証した．

- セット 1: 振動介入なし（10 分）
- セット 2: 振動介入あり（10 分）
- セット 3: 振動介入なし（10 分）
- セット 4: 振動介入あり（10 分）

実験結果　図 2.24 は，姿勢を改善するための介入実験の結果を示している．横軸は各ユーザーを示し，縦軸は各ユーザーの 4 つの実験からの悪い姿勢率（10 分間に悪い姿勢としてカウントされる時間の割合）を示している．結果は，介入ありの期間は，悪い姿勢の割合が減少することを示している．また，ベルトによるコンテキストアウェア振動介入によって，セット 1 と 2 の間の全体的な姿勢改善率（悪い姿勢の減少具合）は，平均で 84.8 %であった．ただし，セット 2 と 3 および 3 と 4 の間で姿勢改善率が増加または減少するユーザがいるため，ユーザの属性に応じて振動介入の有効性に違いがあることが確認できる．

図 2.25（a）は，介入のわかりやすさに関するアンケートの結果，特にベルトからの振動触覚介入が他の通知デバイス（スマートフォンやスマートウォッチなど）と比較して気付きやすいかどうかを示している．図 2.25（b）は，*WaistonBelt X* を装着したときのユーザの不快感に関するアンケートの結果を示している．結果は，SenStick を活用して，*WaistonBelt X* をバックルサイズの小型デバイスに収めた

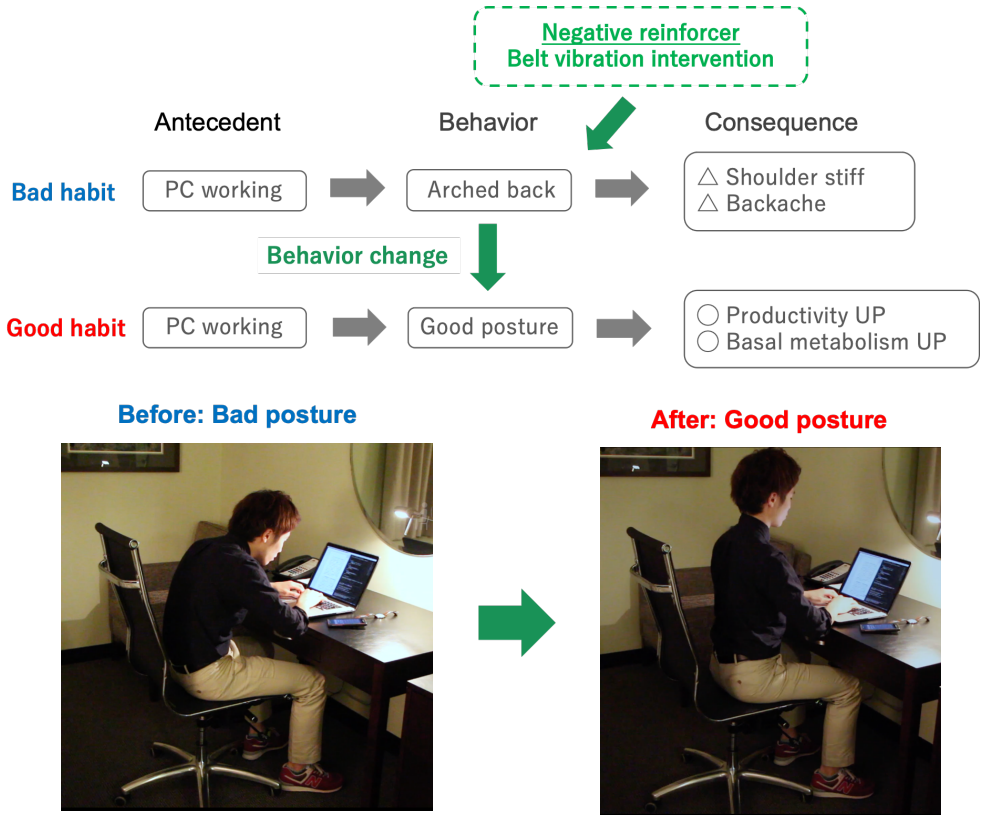


図 2.23: 想定する行動変容シナリオ（悪い姿勢の矯正）

ことによって，ユーザーが感じる不快感を軽減できることを示している．

2.5.3 その他の事例

SenStick は，IoT プロトタイピングの講義でも使用された．生徒は，SenSticks を使用して，独自の IoT デバイスを簡単に実現した．ここでは，開発された IoT デバイスの一部を紹介し，学生が回答した SenStick の有用性に関する調査結果を紹介する．

図 2.26 に示すようなメガネ型の SenStick を示している．学生は，メガネ型 SenStick の照度センサを使用して部屋の中にいるユーザの位置推定に取り組んだ [44]．具体的には，照度の強さから複数の照明装置とメガネの間の距離を推定し，三辺測量法を使用してユーザの位置推定を行った．SenStick を使用することに

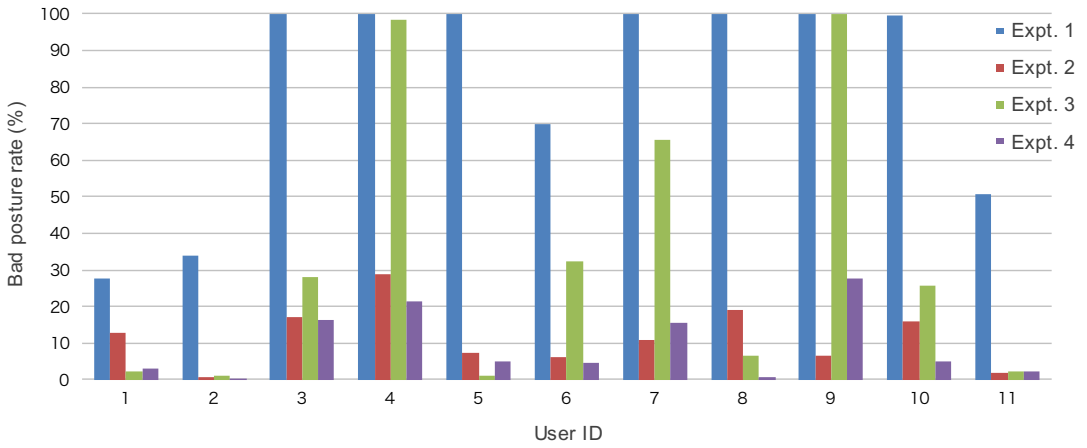


図 2.24: 振動介入によるユーザごとの悪姿勢率の変化

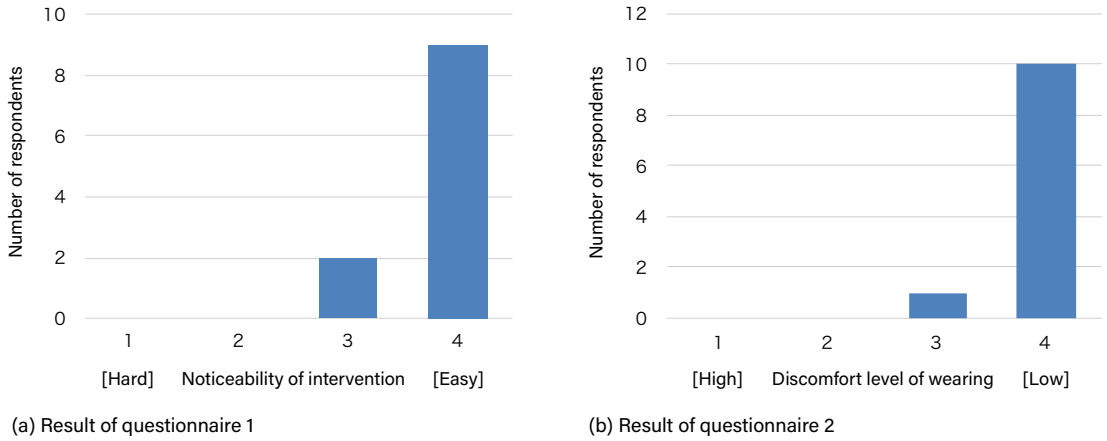


図 2.25: 介入実験に関するアンケート結果

よって，照度センサを備えたメガネを簡単に開発することに成功している．

図 2.27 は，SenSticks を備えた歯ブラシを示している．この歯ブラシ型 SenStick は，磨きた動きをカウントし，リアルタイムでスマートフォンアプリケーションにカウント数が表示するシステムとなっている．SenStick を活用することで，成果物のサイズは非常にコンパクトであり，普段の歯ブラシと大差ない大きさである．そのため，ユーザは，特に違和感なく，いつもと変わらない歯磨き動作を行うことができる．しかしながら，このような水回りで使用するデバイスには，防水性能が求

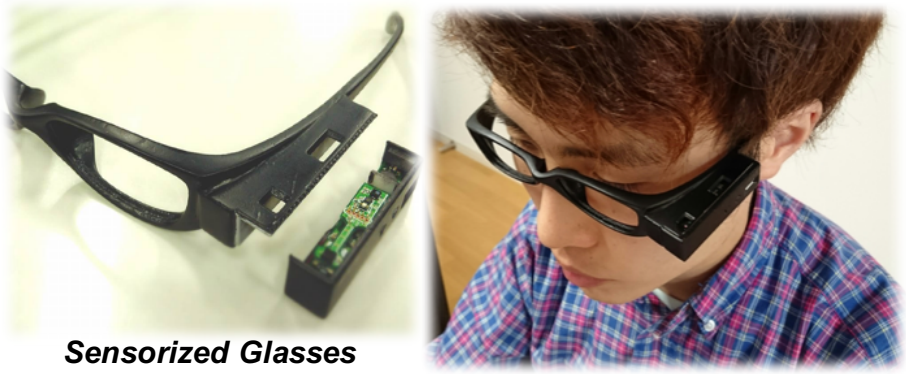


図 2.26: メガネ型 SenStick

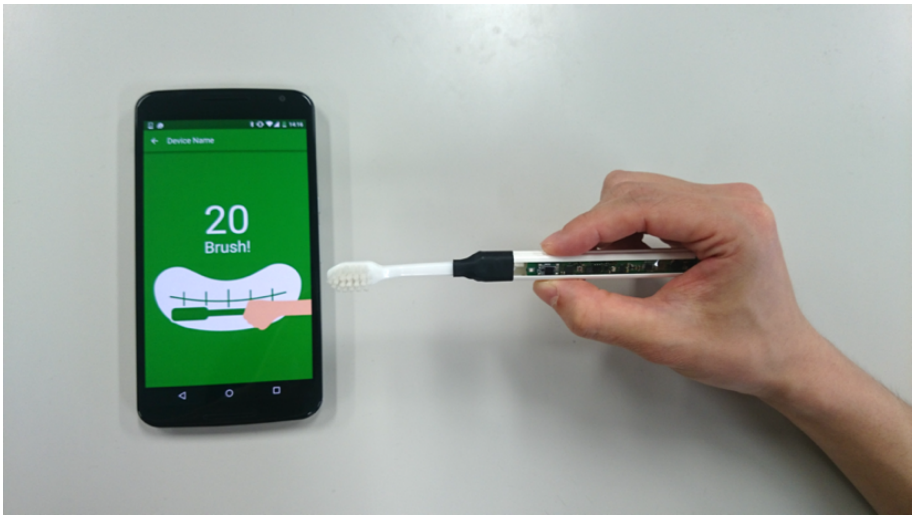


図 2.27: 歯ブラシ型 SenStick

められる．そのため，SenStick の防水化は今後の課題である．

鉛筆で書かれた文字を認識するために，図 2.30 に示す鉛筆型の SenStick が開発された．このデバイスは，鉛筆の正確な動きを検出するために，上下に 2 つの SenStick を搭載しているが鉛筆のサイズは妥当である．

図 2.31 は，名札型 SenStick と SenStick から得られたデータを可視化するデジタルサイネージを示している．バッテリーを 60mAh から 900mAh に変更しているため，名札型 SenStick は 1 週間以上動作することが可能である．従業員や学生

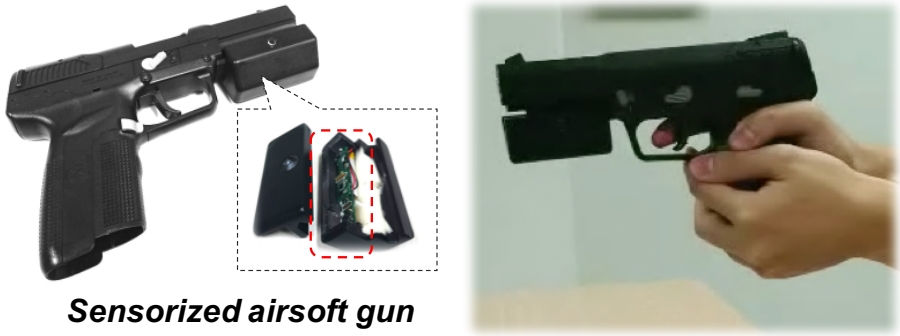


図 2.28: エアソフトガン型 SenStick

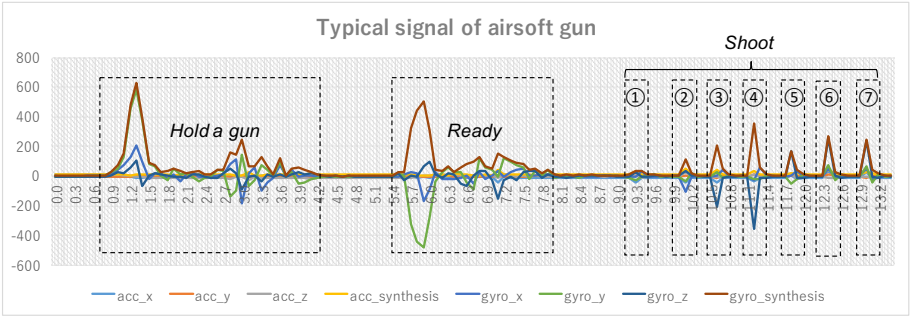


図 2.29: エアソフトガンから得られるセンサデータの例

の日常活動を記録して，健康をサポートすることを目的としている．デジタルサイネージは，ユーザーが目の前に立ったときに，BLE 通信を経由してユーザのアクティビティデータを可視化する．

図 2.28 は，サバイバルゲーム用のエアソフトガン型 SenStick である．サバイバルゲームは，エアソフトガンと呼ばれるレプリカの武器を使用して，参加者が互いに衝突することで対戦相手を排除するスポーツである．エアソフトガン型 SenStick から得られた典型的なセンサデータ のグラフを図 2.29 に示す．ユーザは，SenStick をエアソフトガンに取り付けて，ボールのショット数をカウントした．

図 2.32 は，子供向けのデジタル金魚すくいゲームで使用されるポイ型 SenStick を示している．このシステムでは，子どもたちは，ポイ型 SenStick を使用して，投影マッピングによって表示されるデジタル魚を捕まえることができる．このシステムは，大学のオープンキャンパスイベントで実際に使用され，約 100 人の子供がこ

図 2.30: ペン型 SenStick

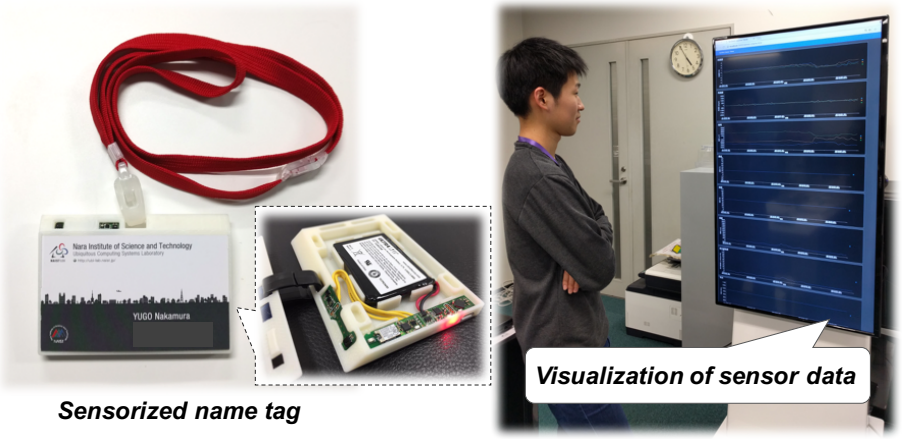


図 2.31: ネームプレート型 SenStick

のゲームを楽しんで，IoT テクノロジーに関心を示した．

上記のプロトタイプを開発した 10 人の大学院生（9 人の男性と 1 人の女性）から得られたアンケート結果を表 2.7 に示す．アンケートは 5 段階の形式である（5：強く同意する，4：少し同意する，3：同意しない，同意しない，2：少し同意しない，1：強く同意しない）．質問 1 と 2 の結果から，多くの参加者は，SenStick プラットフォームが IoT 研究と IoT 教育に役立つと考えていることが確認できる．また，質問 3 の結果は，多くの学生が SenStick を再び使用したいことを示している．質問 4 と 5 の結果から，SenStick の使いやすさと仕様に関する満足度が比較的高いこと

図 2.32: 金魚すくいのポイ型 SenStick

表 2.7: SenStick に関するアンケート結果

Question contents	Score
Q1. SenStick は IoT の研究開発に役立つと思いますか？	4.9
Q2. SenStick は IoT の教育に役立つと思いますか？	4.7
Q3. SenStick をまた使ってみたいと思いますか？	4.9
Q4. SenStick は使いやすかったですか？	4.1
Q5. SenStick のスペックは適切でしたか？	4.1

が確認できる．また，SenStick の良かった点として，「IoT プロトタイピングのアイデアをすばやく試すことができた」，「センサデータを収集するのが簡単だった」，「SenStick は小さく，ほとんどのモノに簡単に取り付けられた」などの意見が記述されていた．一方，SenStick の改善点として，「SenStick デバイス内でデータ処理をしたい」，「ファームウェアを簡単に変更したい」などの意見があった．これらの改善点を踏まえて，九州工業大学 田中准教授のグループで，SenStick のファームウェアを mruby/c という Ruby ライクの軽量言語で自由に記述可能な拡張版の開発が進められており，2019 年現在，SenStick3+mruby/c 教育キット*として，一般販売が開始されている．

*SenStick3+mruby/c 教育キット: http://senstick.ruby-b.com/

2.6　まとめ

本章では，「 実環境で利用できる IoT デバイスの選択肢が少なく，センシング対象が限られてしまう」という問題点を解決すべく，8 つのセンサ（加速度，ジャイロ，磁気，光，UV，温度，湿度，圧力），BLE 通信モジュール，フラッシュメモリ，バッテリー，充電回路を搭載し，どこにでも組み込み可能な超小型のマルチセンサボード，スマートフォンや PC などマルチプラットフォームに対応したデータ収集用のソフトウェア，自由にデバイスの形状をデザイン可能な 3D プリンタ用のケースデータという，新しい組み合わせによって，誰でも簡単に対象のフィールドに適した IoT デバイスを試作でき，素早くセンサデータを収集できる環境を提供する SenStick プラットフォームを紹介した．そして，ベルトや箸の IoT デバイス化といったケーススタディを通じて SenStick プラットフォームの有用性を評価し，SenStick を活用することで開発プロセスが簡略化されることを示した．また，SenStick を用いて様々なモノを IoT デバイス化することができ，試作された IoT デバイスを用いて，長時間の行動データ収集および高精度（F 値 0.95）の日常行動認識を実現できることを示した．これらのことから，SenStick は，IoT 関する研究開発と教育という 2 つの側面で有用であり，より多くの SenStick が普及することで IoT 技術の更なる発展に寄与することが期待できる．

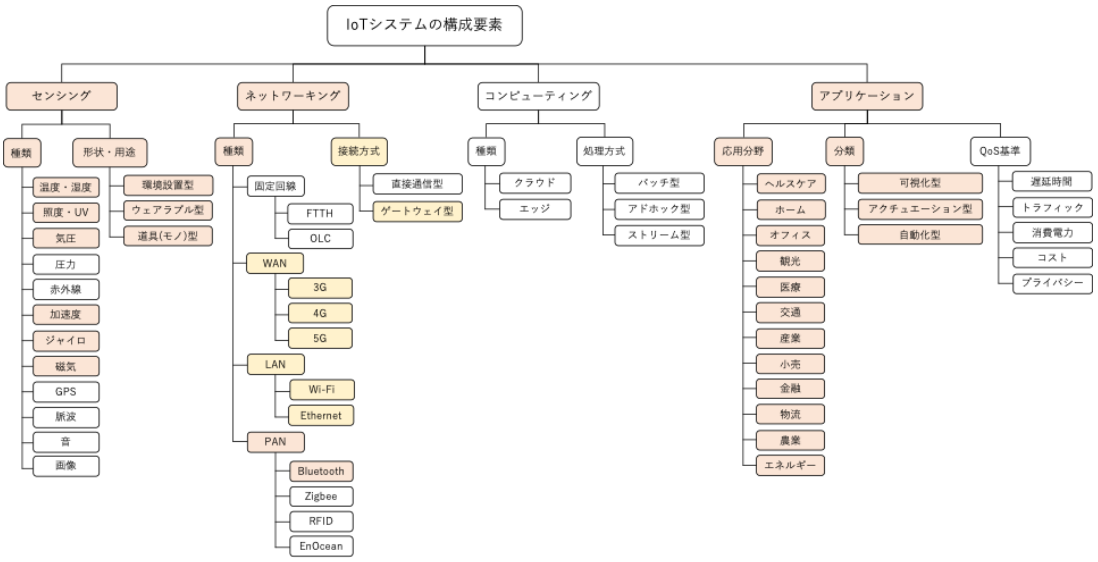


図 2.33: 図 1.2 のタクソノミーに対する SenStick プラットフォームのカバー範囲

図 2.33 に，1 章の図 1.2 で示した IoT システムのタクソノミーに対する SenStick プラットフォームのカバー範囲を示す．まず，SenStick ハードウェアは，温度，湿度，照度，UV，気圧，加速度，ジャイロ，磁気という主要な 8 種類のセンサを備えており，デバイスが非常に小型かつ様々な形状の 3D ケースを必要に応じて作成できることから，あらゆる形状・用途のセンサデバイスを簡単に試作することが可能となっている．また，SenStick ハードウェアは，BLE 通信に対応しており，ゲートウェイとなるスマートフォンや，PC に対してデータを転送することが可能である．転送モードとして，内部メモリに記録したセンサデータログを送信するモードとリアルタイムにセンサデータ をストリーミングするモードを備えていることから，使い方次第であらゆるアプリケーションに応用可能である (対応箇所：ピンク色)．インターネットへの接続方法に関して，スマートフォンやノート PC などのゲートウェイ端末で実行するデータ受信ソフトウェアを拡張することによって，SenStick からゲートウェイ端末を介して WAN や LAN 経由でのデータ転送も可能である (対応箇所：黄色)．

今後は，SenStick ハードウェアのエナジーハーベスト化や，SenStick ハードウェア内のプロセッサを活用したデータ処理やデバイス間のマルチホップ通信などの検討を進める．また，多様な IoT デバイスを用いてさまざまな行動コンテキストを認識するだけでなく，どんなタイミングで，どのようなアクチュエーションを行うと人々の行動変容が促せるのかといった観点から，SenStick をベースとした新しい IoT システムをデザインし，様々な実証実験を実施する予定である．

第 3 章　IFoT: IoT データのエッジ分散処理基盤

前章では，モノの IoT デバイス化とセンサデータ収集を簡素化する，SenStick プラットフォームについて記述した．本章では，「IoT センサデータのタイムリーな処理・分析・応用ができていない」という問題点を解決すべく，データ発生源の近くに存在する IoT デバイス群の計算資源を有効活用しながら効率的にデータ処理を行う IFoT プラットフォームについて記述する．IFoT プラットフォームでは，ローカルに存在する様々な IoT デバイスをセンサデータプロバイダ，計算資源プロバイダ，サービスコンシューマとして抽象化し，一つの系として弾力性のあるデータストリーム処理サービスを提供することを目指している．

3.1　はじめに

近年，Internet of Things（IoT）技術は社会を大きく変える可能性が高いことから大きな注目を集めている．IoT デバイスの数は指数関数的に増加し，2020 年には 500 億以上に達すると予想されている [45]．これに伴い，実世界に展開された無数の IoT デバイスを有効活用し，人々の生活や地域社会の質を向上させる様々な地域密着型の IoT サービス（地域 IoT サービス）を実現することが期待されている．地域 IoT サービスの典型的な例としては，地域の人々が身につけているウェアラブルデバイスから得られるセンサデータを解析し，認識した行動を時間軸にマッピングするライフロギングサービスや，地域の主要スポット（公園/店舗）やモビリティ（バス/車）に設置されている IoT デバイスから得られる騒音や混雑度などの環境情報を地図にマッピングする街の状況可視化サービスなどを想定している．これらのサービスは，地震/洪水のような自然災害時には，避難行動の支援や生存確認などに活用できることから，平常時だけでなく非常時においても，継続的に提供されることが望ましい．そのため，地域 IoT サービスを効率的かつ低コストで提供/維持するためのデータ処理プラットフォームの存在が重要である．

現状，クラウドコンピューティングが，IoT サービスを実現する上での標準的なプラットフォームとしてみなされている [46]．しかしながら，クラウドベースのアプローチは，運用コストが高く，ネットワーク障害に対する回復力が低いことか

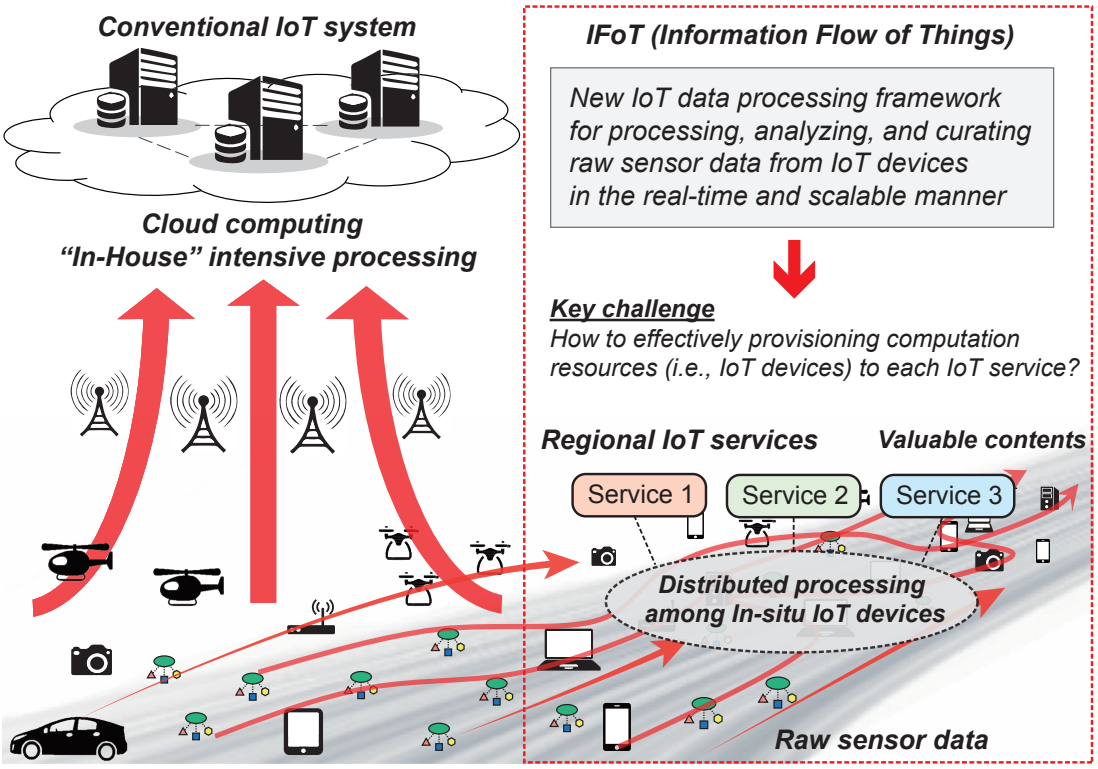


図 3.1: IFoT (Information Flow of Things) のコンセプト

ら，地域 IoT サービスを実現するという点では適していない可能性がある．また，遅延の制約がある IoT サービスを実現する上で，データの発生源から遠いサーバ上で処理タスクを実行するアプローチは通信コストや通信遅延の点で効率的ではない．近年は，これらの背景から，クラウドベースのアプローチの限界に対処するべく，フォグ・エッジコンピューティングと呼ばれる新しいコンピューティングのパラダイムが注目を集めている [47] [48]．これらのパラダイムでは，遅延の影響を受けやすいタスクをサービスユーザに近いコンピューティングリソースに割り当てる手法が採用されている．しかし，エッジコンピューティングのパラダイムは普及の初期段階であり，全ての基地局にデータ処理専用のサーバが導入されているわけではない．

これらの背景を踏まえて，我々の研究グループでは，エッジコンピューティングのアプローチをさらに拡張した次世代型 IoT データ流処理基盤として IFoT

（Information Flow of Things）[49] の実現を目指している．IFoT では，図 3.1 に示すように「地域で生成された IoT データを，地域で処理・分析・応用して有効活用する：IoT データの地産地消」をコンセプトに，エッジサーバが存在しない環境でも，データ発生源に近い IoT デバイスの計算資源を有効活用して効率的に IoT データを処理し，低コストかつ質の高い IoT サービスを提供することを目指している.. 地域 IoT サービスの QoS（Quality of Service）を保証するためには，各 IoT サービスに対してユーザが要求する遅延制約を満足するように，必要となる計算資源をいかに効率的に確保するかが重要な課題である．しかしながら，IFoT で想定する多種多様な IoT デバイスで構成された分散システムは，デバイスの計算能力，デバイス間のネットワーク性能，デバイスの配置密度などに関して異質性を有するため，ユーザの計算需要を満たすように計算資源を確保し，QoS の高いサービスをプロビジョニングすることは困難な制約付き最適化問題となりえる．

本章では，上記の問題を，(a) 対象の地域エリアに存在する IoT デバイス群で構成されるリソースグラフ，(b) 地域エリアで展開される IoT サービスの集合，(c) 一定時間にユーザから各 IoT サービスに対して送信されるセンサデータパケットの集合，(d) センサデータパケットに対応する処理タスクグラフの集合，を入力として，QoS の最大化に向けてセンサデータストリームを素早く処理するための (e) タスク実行スケジュール（対象地域に存在する IoT デバイスに対する各タスクの割り当てプラン）を求める遅延制約付き地域 IoT サービスプロビジョニング問題として定式化する．そして，地域に存在する IoT デバイス群を効率的に管理およびネットワーキングしながら，計算需要に応じた弾力性のあるサービスプロビジョニングを提供する IFoT プラットフォームの設計を示す．また，遅延制約付き地域 IoT サービスプロビジョニング問題を解決するために，IFoT プラットフォーム上で実行されるスケジューリング手法についても説明する．この手法は，(1) IoT サービスの計算需要に応じて動的に計算資源の選択エリアを拡張する適応型スケールアウトと (2) タブー探索に基づく地産地消タスクスケジューリングから構成されている．IFoT プラットフォームの有効性を検証するために，前章で紹介したベルト型 IoT デバイスを用いた行動認識サービスを題材とした実機の IFoT プロトタイプシステムによる評価実験と 4000 台の IoT デバイスが展開された地域エリア（エリア：2km × 2km）で IFoT プラットフォームを展開することを想定したシミュレーションに

よる評価実験を行った．その結果，プロトタイプシステムを用いた評価実験では，ローカルの IoT デバイス群を有効活用することで 250ms の遅延要求を満たすタイムリーな行動認識サービスを実現可能であることを確認し，シミュレーションによる評価実験では，サービスの需要が増加した場合でも，近隣のエッジ IoT デバイスにオフローディングするアダプティブスケールアウトを使用することで QoS を維持できること確認した．

3.2 関連研究

本節では，本研究の関連する既存研究として，分散コンピューティングシステム，クラウドコンピューティング，エッジコンピューティングの分野で検討されてきたタスク割り当て手法について概説し，本研究の位置付けを明確にする．

3.2.1 分散コンピューティングシステム

分散処理システムにおけるプロセッサへのタスクの割当てを最適化する問題は長年関心が集まっており，ヒューリスティックな手法を活用した様々な手法が検討されてきた [50–52]．Shem ら [53] は，グラフマッチングアプローチに基づいて分散コンピューティングシステムのタスク割り当てモデルを提案している．彼らは，タスクの割り当てをグラフマッチング問題として定式化した．これは，最小のタスクターンアラウンドタイムでタスクグラフからプロセッサグラフへの準同型を見つける最適化問題である．彼らは，状態空間探索アルゴリズムを適用することによって，単一タスクグラフに対するタスク割り当てを解決した．したがって, このアプローチは，我々が対象とする複数タスクグラフの割り当てを考慮していない．また，Salman らは，均質の分散コンピューティングシステムに対する複数タスクの割り当て問題を解決するために，パーティクル群最適化に基づくタスク割り当てアルゴリズムを提案した [54]．このアプローチは，我々が対象とする異種の処理資源を有する分散 IoT システム環境には適用が難しい．また，基本的に，クラスタコンピューティングやグリッドコンピューティングの研究分野で検討されてきた既存手法の多くは，バッチ型の大規模な計算タスクを細かく分割し，インターネットを介

して世界中に存在する余剰の計算資源に分配した後に，分散で計算タスクを実行することを想定しているため，IoT サービスで重要となる即時性や地域性という要素が考慮されていない．

3.2.2　クラウドコンピューティングシステム

クラウドコンピューティングシステムは，分散コンピューティングシステムの一種であり，クラウドコンピューティングシステムにおけるタスク割り当ては，分散コンピューティングシステムと同様に重要である．クラウドコンピューティングシステムにおけるタスク割り当ての研究 [55] [56] [57] における課題は，計算資源を適応的にプロビジョニングし，クラウド上で提供するサービス内のすべてのジョブの期限を守りながら, ランニングコスト（金銭的）を最小化することである．これらの研究では，処理資源として仮想マシンをほぼ無限にプロビジョニングすることを仮定しているため，使用可能な処理資源の制限は考慮されていない．さらに，仮想マシンのランニングコストを最小限に抑えることに重点が置かれているため，処理遅延を考慮しても，データ通信遅延は考慮されていない．したがって，これらのアプローチは，利用可能な物理 IoT デバイスおよび処理リソースの制限，ならびにタスク間のデータ通信遅延を考慮する必要がある分散 IoT システム環境には適していない．

3.2.3　エッジコンピューティングシステム

最近の研究では，エッジおよびフォグコンピューティングシステムにおけるリソースプロビジョニング手法にも注目が集まっている．Xu ら [58] は，エッジコンピューティングシステムで位置ベースの拡張現実感ゲームなどの位置情報に基づく遅延に敏感なアプリケーション実行するプラットフォームを提案している．このプラットフォームでは，地域エッジのマイクロデータセンターまたは大規模データセンターの処理資源をクラウドにプロビジョニングする．ただし，大規模なサービスエリアをカバーしながら遅延を減らすためには，多数のエッジサーバーを展開する必要があるため，プラットフォームの実現に高い資本コストと保守コストが必要で

ある． Ardagna ら [59] [60] は，クラウドコンピューティングとフォグコンピューティングを組み合わせたリソースプロビジョニングプラットフォームを提案している．このプラットフォームでは，フォグコロニーと呼ばれる（フォグサーバ集合）の処理資源を活用し，追加のリソースが必要な場合にのみクラウドのリソースを使用するアプローチを採用している．しかし，このアプローチは，計算資源の需要に応じてフォグコロニーをスケールアウトすることは検討されていない．したがって，利用可能な計算資源が近くに存在する場合にも，クラウド上の追加計算資源を活用することから，近隣のリソースを活用する場合と比較して，遅延時間が長くなる場合がある．

3.2.4 本研究の位置付け

本研究では，即時性と地域性に着目し，既存のクラウドやエッジコンピューティングのパラダイムで活用することを想定されていなかったローカルに存在する無数のエッジ IoT デバイスが有している余剰の計算資源を有効活用しながら，即時性の求められる IoT アプリケーションを高い QoS で提供するという，これまでにない新しい問題を対象としており，我々が知る限りでは，地域の存在する IoT デバイス群を最大限に有効活用して，低コストかつ質の高い地域 IoT サービスを実現するための最初のアプローチである．また，提案する IFoT プラットフォームは，既存の分散システムモデルを拡張し，地域で利用可能なエッジ IoT デバイス群の効率的な管理およびネットワーキングを可能にするとともに，サービスの計算需要に応じて，QoS を満たすように現場の計算資源を確保しながら，弾力性のあるデータ処理サービスを提供するという点で新規性がある．

3.3 問題設定とプラットフォーム要件

3.3.1 地域 IoT サービスプロビジョニング問題

本研究で対象とする問題は，(a) 対象の地域エリアに存在する IoT デバイス群で構成されるリソースグラフ $R = (P, L)$，(b) 地域エリアで展開される IoT サービスの集合 $S = \{s_1, s_2, ..., s_n\}$，(c) 一定時間にデータプロバイダから各 IoT

サービス $s \in S$ に対して送信されたセンサデータパケットの集合 $SD_s \subset SD$, (d) それぞれのセンサデータパケット sd に対応する処理タスクグラフの集合 $\cup_{sd \in Q_s} G_{sd} (= (T_{sd}, E_{sd}))$ を入力として，QoS の最大化に向けて処理結果を素早く生成するための (e) タスク割り当てプランを求める問題である．以下では，想定環境や問題の定義について詳しく説明する．

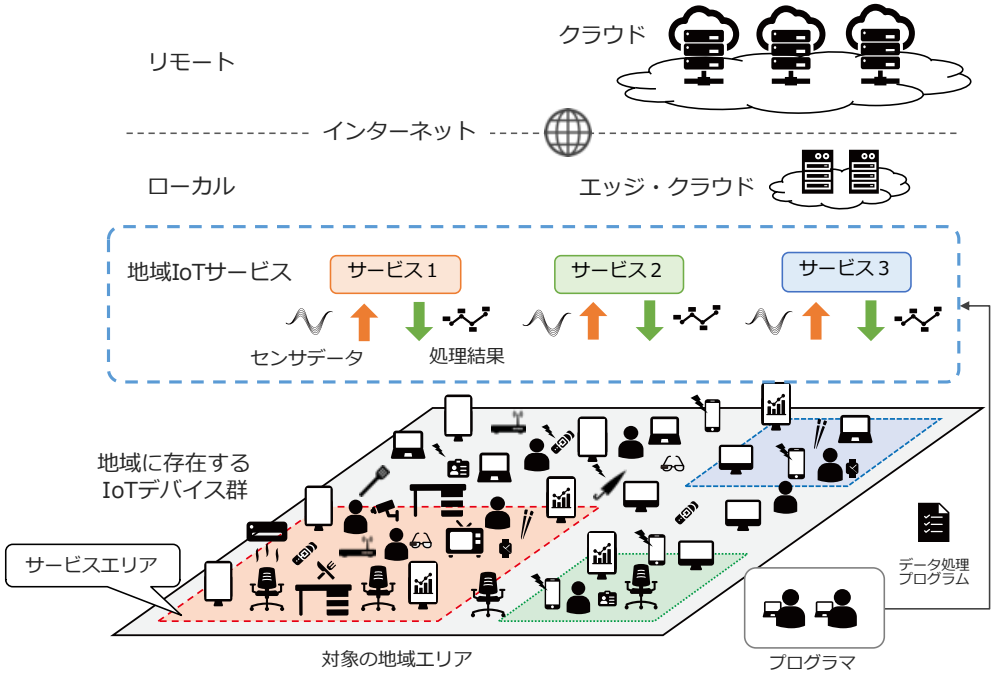


図 3.2: IFoT の想定環境

地域エリア

図 3.2 に，本問題で想定する地域エリアを示す．想定する地域エリアには多数の IoT デバイスが展開されている．これらの IoT デバイスは，地域ネットワークに接続しており，相互にコミュニケーションが可能なものとする．また，地域専用のエッジクラウドが展開されており，インターネットを介して，クラウドとも連携可能である．そして，地域エリアには，複数の IoT デバイスを活用した IoT サービスが展開されている．各 IoT サービスにはサービスエリアが設定されており，ユーザは，これらのエリア内で，サービスの恩恵を受けることができる．IoT サービス

は，センサから取得したセンサデータを処理・解析し，ユーザや環境のコンテキストを推定した上で，アクチュエータと連動しながら，ユーザに情報の提示や介入を行うコンテキストアウェアサービスを想定している．これらのサービスを実現する上で必要となるデータ処理プログラムは，サービス実施者であるプログラマによって提供されるものとする．IFoT プラットフォームでは，地域に存在する IoT デバイス群やエッジクラウド，リモートに存在するクラウドをうまく活用しながら，これらのサービスを高い品質でプロビジョニングすることが求められる．

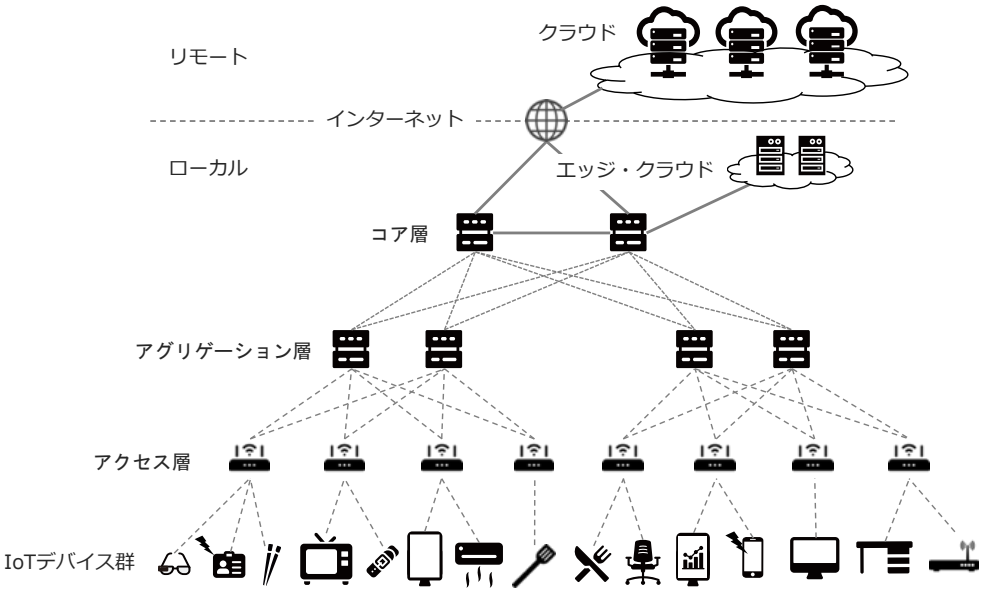


図 3.3: 想定するネットワーク構成

地域ネットワーク

地域ネットワークの構成を図 3.3 に示す．このように，地域ネットワークは，コア層，アグリゲーション層，アクセス層からなる階層構造となっている．このようなネットワークアークテクチャは，既存研究 [61, 62] においても想定されており，非常に一般的な構成となっている．各 IoT デバイスはこの地域ネットワークを介して相互に接続可能である．また，コア層を介して，エッジ・クラウドと接続できる．また，インターネットを介して，クラウドと接続することが可能である．このようなネットワーク構成となっていることから，近くに存在するデバイスほど，短

い通信遅延で相互にコミュニケーションが可能である．地域に展開された計算資源（IoT デバイス）とそれらを結ぶ地域ネットワークで構成されるグラフ構造をリソースグラフと呼び，$R = (P, L)$ と表記する．ここで，P はプロセッサ（IoT デバイス）の集合であり，L は 2 つの頂点間を結ぶ無向リンクの集合を表す．

地域 IoT サービス

地域 IoT サービスの概要を図 3.4 に示す．地域エリアで提供される地域 IoT サービスの集合を $S = \{s_1, s_2, ..., s_n\}$ と表す．このように，地域 IoT サービスは，センサからのセンサデータストリームを入力として，何らかの計算処理を実行し，出力として処理結果のストリームをアクチュエータに送信するものとする．センサデータストリームは，1 度に 1 つづつ到着するデータ要素のシーケンス $sd_1, sd_2, ...$ である．各データ要素 sd は，タイムスタンプとセンサデータから構成されている．処理結果ストリームも基本的にはセンサデータストリームと同様の構成を想定する．IoT サービス s を提供する上で，必要となるデータ処理プログラムは，サービスクリエータから提供されるサービスレシピに記述されている．ここで，センサデータをどのように処理するかを示した順序付きのタスクの集合をタスクグラフと呼び G_s と表す．基本的には，(1) Start, (2) データ収集タスク, (3) データ処理タスク, (4) データ集約タスク, (5) end という 5 つのタスクから構成されるタスクグラフを想定する．センサがセンサデータを送信してから，アクチュエータが処理結果の受信するまでの時間をサービス遅延と定義する．このサービス遅延に関して，あらかじめサービスクリエータから，サービスレベルアグリーメントとして満足できる目標遅延時間（PD: Preferable Delay）と許容可能な限界遅延時間（MD: Marginal Delay）が定義されているものとする．

各サービスに対する動的リソースサブグラフ

各 IoT サービス $s \in S$ は, それぞれ自身のサービスを提供するための計算資源プールとして $R = (P, L)$ のサブグラフである R_s を確保する．リソースサブグラフ R_s は以下のように定義される．ここで，$pos(p)$ はプロセッサ p の位置を示し，$area(s)$ はサービス s のサービスエリアを示す．

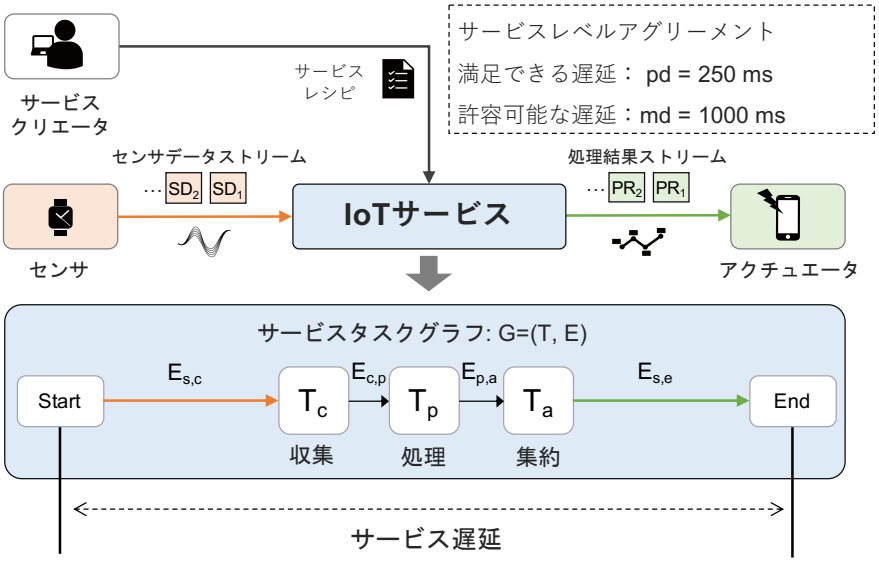


図 3.4: IoT サービスの概要

$$R_s = (P_s, L_s) \tag{3.31}$$

where

$$P_s = \{p | p \in P_A \wedge pos(p) \in area(s)\}$$
$$L_s = \{l | pos(l.st) \in area(s) \wedge pos(l.ed) \in area(s)\}$$

上記に示すように，初期状態では，各サービス s のリソースサブグラフ R_s の計算資源選択エリアはサービスエリアに $area(s)$ 絞られているが，各サービスはこのリソースグラフを動的に拡張することでスケールアウトすることが許可されている．拡張可能な動的リソースグラフ R_s^{i+} を以下のように定義する．$R_s^{(i-1)+} = (P_s^{(i-1)+}, L_s^{(i-1)+})$ から拡張されたサービス s のリソースサブグラフを $R_s^{i+} = (P_s^{i+}, L_s^{i+})$ と表す．ここで $R_s^{0+} = R_s$ であり，$|P_s^{i+}| = |P_s^{(i-1)+}| + 1$ が成り立つ．つまり，計算資源選択エリアを拡張することによって，最低 1 つの計算資源プロバイダが R_s^{i+} に追加される*．

*各サービスの資源選択エリアは，対象の地域エリア全体まで拡張され，他のサービスのリソース領域と重複しても良い．

タスク割り当てとサービス遅延時間

ここでは，説明を簡略化するために，対象の地域エリアに存在するユーザから発行されるすべてのセンサデータパケット集合 SD に付随する全てのタスクグラフの集合 G を以下のように定義する．

$$G = (T, E) \text{ where } T = \bigcup_{sd \in SD} T_{sd}, E = \bigcup_{sd \in SD} E_{sd} \tag{3.32}$$

そして，対象の地域エリアに存在する全てのプロセッサ $p \in P$ とタスク $t \in T$ のすべての組み合わせを表す行列の要素として，変数 $x_{t,p}$ を定義する．t が p に割り当てられている場合は 1 になり，それ以外の場合は 0 になる．本問題では，T のすべてのタスクが 1 つのプロセッサ $p \in P$ に割り当てられるものとする．この制約条件を以下の式で表す．

$$\forall t \in T, \sum_{p \in P} x_{t,p} = 1 \tag{3.33}$$

各タスク t は $comp(t)$ で示される必要な計算量を持ち，各プロセッサ p は $cap(p)$ で示される所定のタスク受け入れ上限計算キャパシティを持つ．p は $cap(p)$ の範囲内で，できるだけ多くのタスクを受け入れることが可能である．この制約条件を以下の式で表す．

$$\forall p \in P_A, \sum_{t \in \{t | x_{t,p} = 1\}} comp(t) \leq cap(p) \tag{3.34}$$

図 3.5 に示すように，サービスタスクの実行時には，利用可能な計算資源の状況に応じて，各センサデータパケット sd に対して，処理タスクの並列数を変更したランタイム用のタスクグラフ G_{sd} が生成される．タスクグラフ G_{sd} を実行する際の総遅延時間は，Start から End までのすべての実行パス間（各パスには，収集タスク，処理タスク，および集約タスクが順に 1 つずつ含まれている）の最大処理/通信遅延である．

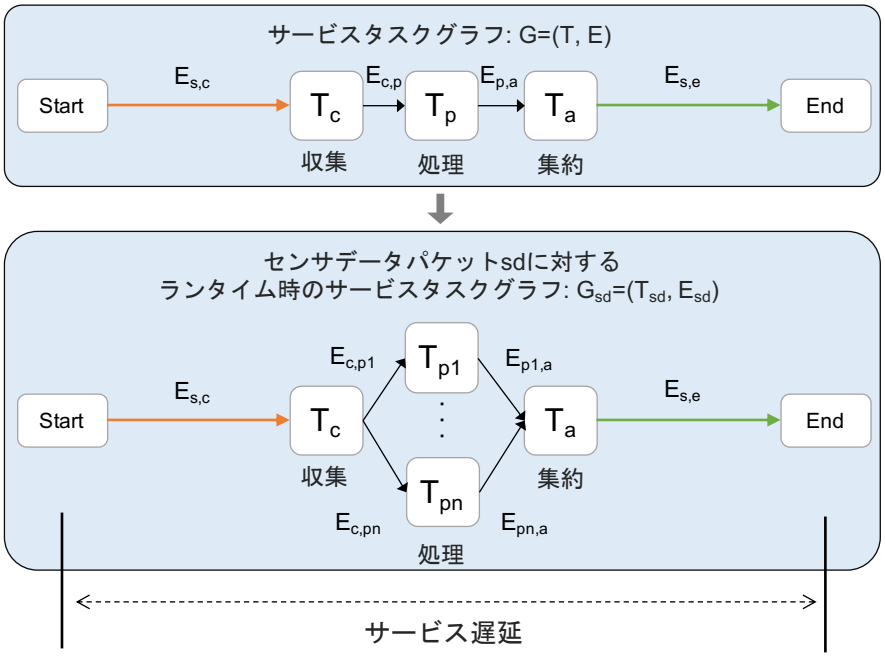


図 3.5: センサデータパケットに対するサービスタスクグラフ

タスクグラフ $G_{sd} = (T_{sd}, E_{sd})$ の経路数を n_{sd} とする．$path_{sd,i} = \langle Start, tc_{sd,i}, tp_{sd,i}, ta_{sd,i}, End \rangle$ は，G_{sd} の i 番目の実行経路を表す．ここで，$0 \leq i \leq n_{sd}$ であり，$tc_{sd,i}$, $tp_{sd,i}$, $ta_{sd,i}$ は，それぞれタ収集タスク，処理タスク，集約タスクを表す．ここで，$T_{sd} = \bigcup_{0 \leq i \leq n_{sd}} \{tc_{sd,i}, tp_{sd,i}, ta_{sd,i}\}$ である．

タスク t が割り当てられるプロセッサを $p(t)$ とし，t が出力するデータのサイズを $dsize(t)$ とする．ここでは，プロセッサ p 上でタスク t が実行されるときの処理遅延を見積もる関数 $pt(t,p)$ およびサイズ $dsize$ のデータがリンク $l \in L$ を介して転送されるときの通信遅延を見積もる関数 $dt(l, dsize)$ が利用可能であると仮定する．

各実行パスについて，$delay(path(sd,i)) = delay(\langle Start, tc_{sd,i}, tp_{sd,i}, ta_{sd,i}, End \rangle$ で表される処理遅延と通信遅延の合計は，次の式で計算される．

$$delay(path(sd,i)) = \sum_{p \in P_A} [pt(tc_{sd,i}, p) \cdot x_{tc_{sd,i},p} + pt(tp_{sd,i}, p) \cdot x_{tp_{sd,i},p}$$

$$
\begin{aligned}
&+pt(ta_{sd,i}, p) \cdot x_{ta_{sd,i},p}] \\
&+dt(link(Start, tp_{sd,i}), dsize(tc_{sd,i})) \\
&+dt(link(tc_{sd,i}, tp_{sd,i}), dsize(tc_{sd,i})) \\
&+dt(link(tp_{sd,i}, ta_{sd,i}), dsize(tp_{sd,i})) \\
&+dt(link(ta_{sd,i}, End), dsize(ta_{sd,i}))
\end{aligned} \tag{3.35}
$$

ここで

$$
link(t, t') = (p(t), p(t')), \forall t, t' \in T_{sd}
$$

そして，G_{sd} を実行するための総遅延時間 $D(sd)$ は, 以下の式で表すことができる．

$$
D(sd) = \max_{0 \leq i \leq n_{sd}} delay(path(sd, i)) \tag{3.36}
$$

QoS の効用関数

本問題では，サービスクリエータによって指定されるサービスレベルアグリーメントに基づき，以下のような QoS（Quality of Service）を表す効用関数を定義する．

$$
U(sd) \stackrel{\text{def}}{=} \begin{cases}
1 & (D(sd) < pd) \\
1 - \dfrac{1}{1 + e^{5(ad - D(sd))/(ad - pd)}} & (pd \leq D(sd) \leq ad) \\
\dfrac{1}{1 + e^{5(D(sd) - ad)/(md - ad)}} & (ad < D(sd) \leq md) \\
0 & (md < D(sd))
\end{cases} \tag{3.37}
$$

$$
ad = \frac{pd + md}{2} \tag{3.38}
$$

この効用関数は，先行研究 [63] における効用関数のデザインに基づいている．図 3.6 に，pd と md の異なる値に対する効用関数の曲線を示す．このように，サービス遅延が満足できる遅延時間 pd を満たす場合には最大値，許容可能な限界遅延時間 md を上回ると効用が 0 となる関数となっている．

問題定義

本研究で対象とする問題は，(a) 対象の地域エリアに存在する IoT デバイス群で構成されるリソースグラフ $R = (P, L)$，(b) 地域エリアで展開される IoT サービスの集合 $S = \{s_1, s_2, ..., s_n\}$，(c) 一定時間にユーザから各 IoT サービス $s \in S$ に対して送信されたセンサデータの集合 $SD_s \subset SD$，(d) それぞれのセンサデータパケット sd に対応する処理タスクグラフの集合 $\cup_{sd \in SD_s} G_{sd} (= (T_{sd}, E_{sd}))$，を入力として，全てのセンサデータパケットに対するサービス効用関数を最大化するような (e) タスク割り当てプラン $x_{t,p}$ を決定することが目的である．したがって，本問題の目的関数を以下のように定義する．

$$Maximize \sum_{s \in S} \sum_{sd \in SD_s} U(sd) \tag{3.39}$$
$$subject\ to\ (3.33)\ (3.34)$$

この問題は，NP-hard であると知られている RCPSP（Resource-Constrained Project Scheduling Problem）[64] のインスタンスであるため，NP-hard 問題である．

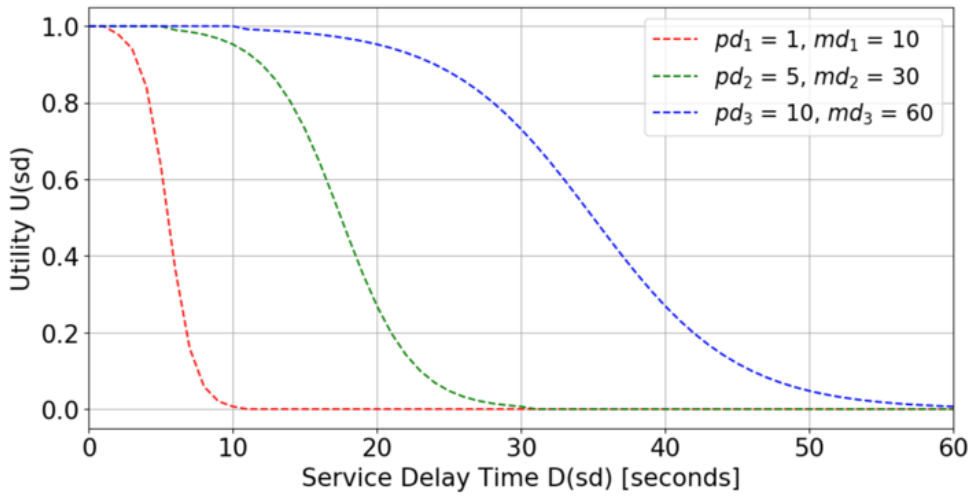


図 3.6: QoS 効用関数グラフ

3.3.2 IFoT プラットフォームの要件

本研究では，上記に示した地域 IoT サービスプロビジョニング問題を効率的に解くとともに，以下に示す 2 つの要件を満たしたプラットフォームの設計開発を行う．

要件 1: IoT デバイス群の効率的な管理およびネットワーキング

IFoT プラットフォームでは，地域に存在する IoT デバイス群のリソースを有効活用しながら，品質の高い IoT サービスをユーザに提供することを目指している．想定する地域エリアには，無数の IoT デバイスが展開されているため，これらのリソースをいかに効率的に管理するかが重要である．また，センサ，コンピュータ，アクチュエータなどそれぞれサービスにおける役割が異なるため，役割に応じてそれらを効率的にネットワーキングする必要がある．

要件 2: 計算需要に応じた弾力性のあるサービスプロビジョニング

IFoT プラットフォームでは，地域に存在する IoT デバイス群のリソースを有効活用しながら，複数の IoT サービスを展開することを目指している．クラウドコンピューティングとは異なり，ローカルの計算資源は有限であるため，各サービスに対してどれくらい計算資源を割り当てるのかが重要である．IoT サービスの計算需要は，サービスに参加するユーザの数によって変動するため，各サービスの計算需要に応じて適切な量のリソースを割り当てながら，高い品質のサービスを提供することが求められる．

3.4 IFoT プラットフォーム

本節では，提案する IFoT プラットフォーム全体の設計や地域 IoT サービスプロビジョニング問題を解決するリソース確保およびスケジューリング手法について記述する．図 3.7 に，IFoT プラットフォームの基本コンセプトを示す．このように，IFoT プラットフォームでは，センサからのセンサデータストリームの入力として，地域に存在するローカルの計算資源を最大限有効活用しながら，弾力性のあるストリーム処理サービスを提供し，素早く処理結果ストリームをアクチュエータに流通することを基本コンセプトとしている．以下では，IFoT プラットフォームのシス

テムアーキテクチャおよびワークフローを示すとともに，地域 IoT サービスプロビジョニング問題を解決すべく IFoT プラットフォーム内で実行されるスケジューリングアルゴリズムについて記述する．

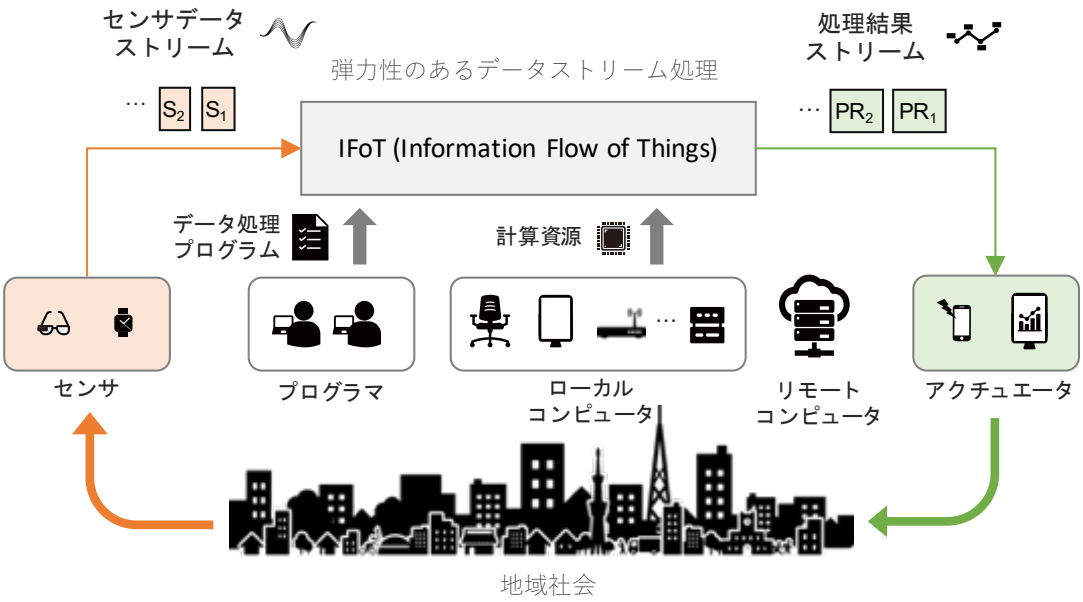


図 3.7: IFoT プラットフォームの基本コンセプト

3.4.1 システムアーキテクチャ

IFoT プラットフォームのアーキテクチャを図 3.8 に示す．このように IFoT プラットフォームは，リソースマネジメント機構，リソースアブストラクション機構，リソースディストリビューション機構，サービスコーディネーション機構という 4 つの機構から構成されている．以下で，IFoT プラットフォームにおける各機構の役割を説明する．

リソースマネジメント機構

リソースマネジメント機構は，IFoT プラットフォームに参加しているリソースの管理を担当する．また，対象エリア内で展開されている全てのサービスを管理

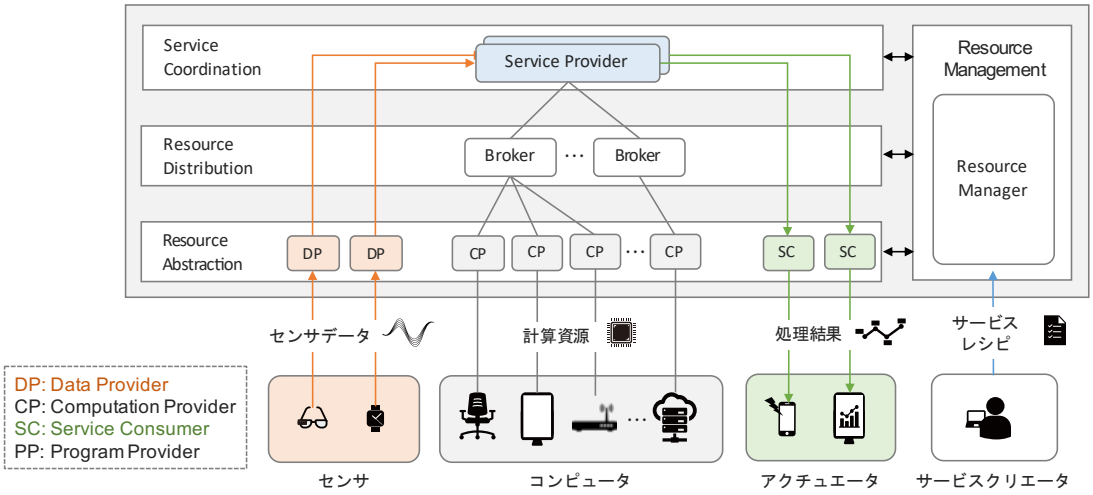


図 3.8: IFoT プラットフォームのアーキテクチャ

する役割も担う．リソースマネージャは，IFoT プラットフォームに参加する全てのノードがアクセス可能な well-known サーバである．基本的に，リソースマネージャは，地域内のエッジクラウドなどエリア内でもっとも堅牢なサーバに割り当てられる．また，障害に対応するためクラウドにもバックアップが準備されている．対象エリアが広大な場合は，小エリアごとに複数のリソースマネージャを設置することも可能である．デバイスの登録やサービスの起動プロセスは，リソースマネージャを介して行われる．各デバイスがどの位置に存在するかといった初期情報は，各デバイスが IFoT プラットフォームに登録される際に，デバイスオーナーによって付与される．サービスにおけるデータ処理プログラムやサービスレベルアグリーメントなどの仕様情報は，サービスのオーガナイザーが作成したサービスレシピに記述されている．リソースマネージャはサービスレシピを元にサービスの起動処理を行う．

サービスコーティネーション機構

サービスコーディネーション機構は，サービスのオーガナイザーから提供されたサービスレシピに基づいて，データ処理サービスを調整する役割を担う．サービスコーディネーション機構は，複数のサービスプロバイダで構成されており，１つのサービスにつき１つ以上のサービスプロバイダが割り当てられる．サービスプロ

バイダは，サービスのデータ処理をする上で必要となるリソース調整やタスクのスケジューリングを行うコンポーネントである．サービスエリアが広範囲な場合や，サービス負荷が高い場合には，複数のサービスプロバイダが起動する．この時，サービスプロバイダは P2P のオーバーレイネットワークで接続する．

リソースディストリビューション機構

リソースディストリビューション機構は，サービスのデータ処理を実行する際のサービスプロバイダと計算プロバイダ間のメッセージングを中継する役割を担う．複数のブローカーで構成されており，サービスプロバイダが配下の計算資源プロバイダの状況を確認する際に発生する通信の負荷を軽減する目的がある．各ブローカーは，配下にいる複数の計算資源プロバイダからハートビートメッセージを単位時間（1 秒）毎に受信し，各計算資源プロバイダの状況を確認する．そして，ブローカーは，現在利用可能な計算資源プロバイダを把握し，集約した情報をサービスプロバイダに共有する．

リソースアブストラクション機構

リソースアブストラクション機構は，各デバイスを役割に応じたコンポーネントとして抽象化する役割を担う．具体的には，センサ，コンピュータ，アクチュエータをデータプロバイダ，計算資源プロバイダ，サービスコンシューマとして抽象化する．データプロバイダは，センサデータを提供するコンポーネントである．計算資源プロバイダは，サービスの指令に応じて，センサデータを処理するコンポーネントである．サービスコンシューマは，処理結果を受信し，何らかの形でユーザにアクチュエーションするコンポーネントである．

3.4.2 システムワークフロー

IFoT プラットフォームの一連のワークフローを図 3.9 に示す．ワークフローは，リソースの登録，サービスの起動，サービスの発見，サービスの実行という 4 ステップとなっている．以下に，それぞれのステップについて説明する．

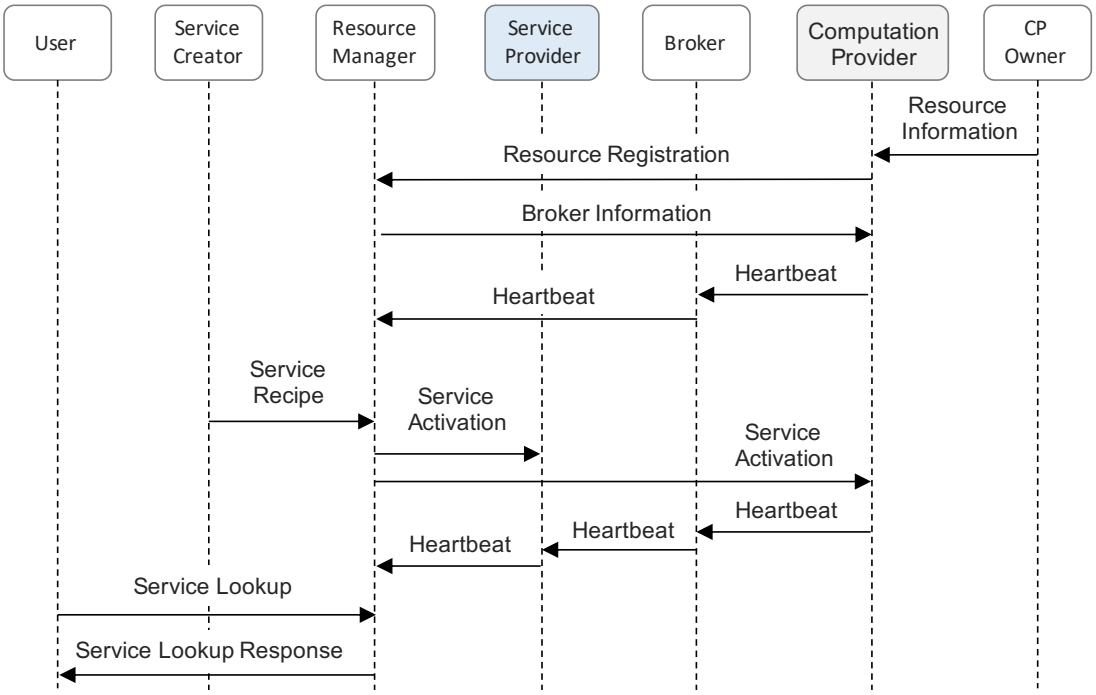


図 3.9: リソースの登録からサービス起動＆発見までのワークフロー

リソースの登録

リソースの登録ステップでは，計算資源プロバイダのオーナーが自身のデバイスに対して，IFoT ミドルウェアをインストールし，名前，位置情報や共有可能な計算資源など初期のリソース情報の設定を行う．その後，デバイスでは，初期セットアップファイルが起動し，地域のリソースマネージャに，入力された設定情報を反映したリソース登録メッセージが送信される．リソースマネージャは，計算資源プロバイダのスペックや位置に応じて，直属のブローカーを決定し，接続するべきブローカーの情報を計算資源プロバイダに返信する．その後は，計算資源プロバイダは，ブローカーに定期的に処理負荷など自身の状態を報告するハートビートメッセージを送信する．ブローカーは，配下の計算資源プロバイダから得られるハートビートメッセージを集約し，リソースマネージャに送信する．これによって，リソースマネージャは，対象エリア内で利用可能な計算資源の状況を把握する．

サービスの起動

サービスの起動ステップでは，はじめに，サービスクリエータはデータ処理プログラムおよびサービスエリア，サービスアグリーメント情報が示されたサービスレシピを作成し，そのレシピファイルをリソースマネージャに送信する．サービスレシピを受け取ったリソースマネージャは，サービスエリアの中から，いくつかのブローカーを選出し，サービスアクティベーションメッセージを送信し，サービスプロバイダコンポーネントを起動する．また，サービスエリア内の計算資源プロバイダに対しても，データ処理プログラムの関数インスタンスを起動する目的で，サービスアクティベーションメッセージを送信する．これによって，サービスの実行に必要となるプログラムやライブラリ，学習モデルが計算資源プロバイダにインストールされる．その後，計算資源プロバイダからブローカー，ブローカーからサービスプロバイダー，サービスプロバイダーからリソースマネージャへと段階的に集約されながら，ハートビートメッセージが共有される．

サービスの発見

サービスの発見ステップでは，ユーザが現在利用可能なサービスを探す目的で，リソースブローカーに対してサービスルックアップメッセージを送信する．リソースマネージャは，ユーザからのサービスルックアップメッセージを受信すると，定期的なハートビートメッセージの監視結果に基づいて，サービスのマッチングを行い，現在利用可能なサービスの情報をユーザに返却する．サービスプロバイダは，サービスアクセス手段として，データプロバイダおよびサービスコンシューマ用の API を有しており，サービスルックアップレスポンスには，このようなサービスアクセス手段に関する情報も記載されている．ユーザは，サービスアクセス手段の情報に基づいて，自身のセンサ（データプロバイダ）やアクチュエータ（サービスコンシューマ）を IFoT プラットフォームと連携させることが可能である．

サービスの実行

サービスの実行ステップにおける，データプロバイダからサービスコンシューマまでのワークフローを図 3.10 に示す．サービスプロバイダでは，データプロバイダおよびサービスコンシューマに対するサービスアクセスプロトコルとして多対多の

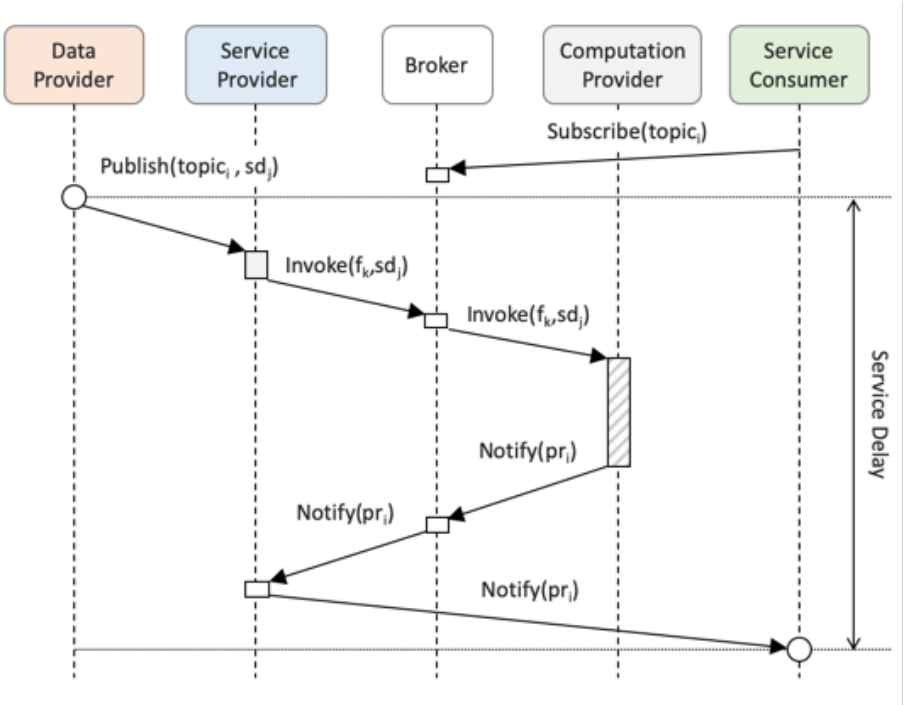


図 3.10: サービスにおけるデータ処理ワークフロー

メッセージングを効率化可能な Pub/Sub モデルを採用している．サービスプロバイダは，一つのデータプロバイダ毎に一意のストリームトピックを管理しており，データプロバイダは，対象のサービスプロバイダに対してセンサデータをパブリッシュすることでセンサデータの送信を行う．送信されたセンサデータは，サービスプロバイダで受信されるとともに，センサデータを処理するためのサービスタスクをどの計算資源プロバイダで実行するのかスケジューリングされる．そして，スケジューリング結果に基づいて，センサデータはブローカーを介して計算資源プロバイダに送信され，計算資源プロバイダ上でデータ処理が実行される．実行結果は，ブローカーを介してサービスプロバイダに送信され，最終的に，処理結果のデータがサービスコンシューマに到達する．このワークフローを繰り返すことで，IFoT プラットフォームでは，センサデータのストリーム処理を実現している．ここで，データプロバイダがセンサデータをパブリッシュしてから，サービスコンシューマで処理結果を受信するまでの時間がサービス遅延である．計算資源プロバイダは，計算量の大小が多様な IoT デバイスで構成されることから，サービス遅延は，サー

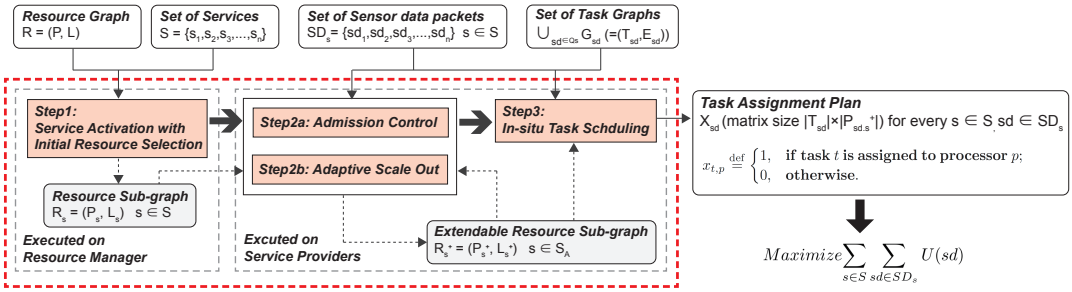


図 3.11: 提案手法の処理フロー

ビスに対してどれくらいの計算資源プロバイダを割り当てるのか？また，サービスタスクをどのようにスケジューリングするのか？によって大きく結果が変わってくる．以下の節では，IFoT プラットフォーム上でサービスプロビジョニングする際のスケジューリング問題を定式化するとともに，その問題を解決するヒューリスティックアルゴリズムを提案する．

3.4.3 サービスプロビジョニング手法

ここでは，前節で述べた地域 IoT サービスプロビジョニング問題を解決すべく，IFoT プラットフォーム内で実行されるヒューリスティックなサービスプロビジョニング手法について記述する．

基本方針

我々は遅延制約付き地域 IoT サービスプロビジョニング問題を解決するために次の 2 つの基本方針を採用している．（*1*）**地産地消タスク割り当て**：データプロバイダからサービスプロバイダに対して送信されるセンサデータパケット sd 毎に生成されたタスクグラフ G_{sd} は，サービスエリア $area(sd.s)$ および拡張された計算資源選択エリア内の計算資源プロバイダに対して優先的に割り当てられ，処理される．（*2*）*FCFS* **タスク割り当て**：センサデータパケットの集合 SD に対して，SD 内の各パケット sd に対応するタスクグラフ G_{sd} のすべてのタスクは，センサデータパケットの生成順序に従い，到着順（FCFS, First-Come-First-Served）で利用可能

な計算資源プロバイダに割り当てられる．

提案手法の処理ステップ

図 3.11 に提案手法の処理ステップを示す．提案手法は，(1) 各サービスの起動時の初期リソースの確保，(2) 各センサデータパケットの受付制御とリソース選択領域の適応型スケールアウト，(3) 受理したセンサデータパケットの地産地消タスクスケジューリングの 3 ステップで構成される．

Step 1: サービス起動時の初期リソース確保

ステップ 1 は，リソースマネージャ上で実行される．このステップでは，リソースマネージャは各サービスを起動すると共に，それぞれのサービスを提供する際の初期リソースとなるリソースサブグラフ $R_s = (P_s, L_s)$ を選択する．図 3.12 に示すように P_s からサービスプロバイダとして最も強力で最も安定した計算資源プロバイダが選択され，その後，サービスプロバイダが起動するとともに，サービスプロバイダ上でステップ 2 とステップ 3 が実行される．

Step 2: センサデータパケット受付制御と適応型スケールアウト

このステップは，ステップ 2a : センサデータパケットの受付制御とステップ 2b : 適応型スケールアウトという 2 つのサブステップで構成されている．ステップ 2a において，サービスプロバイダがセンサデータパケット sd を受信すると，タスクグラフ G_{sd} が生成される．具体的には，以下の手順でタスクグラフ G_{sd} が生成される．

(1) まず，サービスプロバイダは単位センサデータパケット sd とサービスのタスクグラフ G_s における各タスク (T_c, T_p, T_a) の処理負荷から 3 ステップ (Collecting, Processing, Aggregating) それぞれのタスク負荷を算出する．(2) 次に，サービスプロバイダはヒューリスティックな方法で，タスクグラフ G_s を現在のリソースサブグラフ R_s（または R_s^{i+}）を使用して限界遅延 $md_{sd.s}$ 以内で処理できるかどうかを判断する．可能な場合は，現状の G_s を G_{sd} とする．(3) 限界遅延 $md_{q.s}$ 以内で処理できない場合は，タスクグラフ G_s におけるデータ処理タスク T_p を，複数のサブタスク $st_1, ..., st_{k_t}$ に分割する．このとき $comp(st_i) \leq cap(p)(1 \leq i \leq k_t)$ が

成り立つようにする．(4) そして，データ処理タスク T_p をサブタスク $st_1, ..., st_{k_t}$ の並列実行で置き換えることによって，センサデータパケット sd に対応したタスクグラフ G_{sd} が生成される．そして，再度，現在のリソースサブグラフ R_s（または R_s^{i+}）を使用して限界遅延 $md_{sd.s}$ 以内で処理できるかどうかを判断する．並列化しても限界遅延を満たせない場合は，ローカルの計算資源で実行できないとみなし，クラウドにオフロードする．

ステップ 2b において，サービスプロバイダはサービス s のリソース使用量を定期的にチェックし，サービス s の計算リソースが不十分であれば計算資源選択エリアを拡張してリソースグラフ R_s^{i+} を生成する**適応型スケールアウト**を実行し，センサデータパケットが処理される．

Step3: 地産地消タスクスケジューリング

最後に，ステップ 3 において，サービスプロバイダは，受け入れられたセンサデータパケット sd に対するタスクグラフ $G_{sd.s}$ のスケジューリングを行う．具体的には，タブーサーチを元にした探索手法を使用して，行列サイズ $|T_{sd}| \times |P_{sd.s}^{i_s+}|$ をもつ準最適なタスク割り当て X_q を生成する．

ステップ 2 およびステップ 3 の詳細なアルゴリズムを以下に説明する．

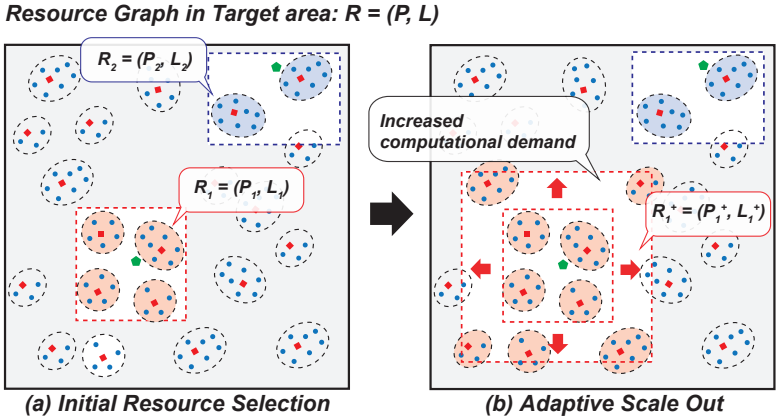


図 3.12: 初期リソース確保と適応型スケールアウトの様子

センサデータパケット受付制御と適応型スケールアウト

サービスプロバイダはセンサデータパケット sd を受け取ると，サービス $sd.s$ のサービスレベルアグリーメント（遅延制約）を踏まえて sd が受付可能かどうかを判断する（予測遅延時間が $md_{sd.s}$ 以下であるか否かを判断）．アルゴリズム 1 は，センサデータパケット sd に対する受付制御のための処理関数 $isAcceptable$ を示す．入力パラメータは，センサデータパケット sd，リソースサブグラフ $R_{sd.s}$（または拡張リソースグラフ $R^{i+}_{sd.s}$），タスクグラフ G_{sd} および限界遅延時間 $md_{sd.s}$ である．出力は，センサデータパケット sd が受け入れ可能かどうかを示すブール値である．センサデータパケット sd が受け入れ可能な場合，最初のリソース割り当て X_0 が，グリーディアルゴリズムに従って生成される．センサデータパケット sd が受け入れ出来ない場合は，ローカルでの実行が困難だと判断し，クラウドにオフロードされる．

アルゴリズム 1 では，変数 $P, X, CPD, falsecount$ が初期化される（2 行目〜5 行目）．ここで，P は割り当てられていないプロセッサの集合，X は要素が $x_{t,p}$，CPD は，タスクを実行できるプロセッサが見つからない場合の計算資源の不足分の合計値，$falsecount$ は誤割り当ての数を示している．6 行目では，md_c, md_p，md_a と表記されているタスクの収集，処理，および集計の限界遅延時間 $md_{sd.s}$ を分割することによって決定される．ここで，それぞれの分割率は経験的に決定された値（事前に与えられている値）を使用する．7 行目から 16 行目では，タスク T_{sd} とプロセッサ P の間のタスク割り当てが決定される．具体的には，遅延制約を満たす t から p の代入が見つかった場合，$x_{t,p}$ は 1 に設定され，p は P から削除される（9 行目〜12 行目）．それ以外の場合，t の実行に必要な計算リソースは CPD に加算され,$falusecount$ が 1 増加する（13〜15 行目）．最後に，17 行目〜21 行目で，$falsecount$ の値に応じて true（最終タスク割り当て X_0 も記録される.）または false（CPD も同様）が返される．

X_0 は，$isAcceptable$ の生成された行列であり，アルゴリズム 2 に使用される．CPD は適応的スケールアウトに使用される（図 3.11 のステップ 2b）．$CPD > 0$ の場合，S ブローカーは拡張リソースグラフ R^{i+}_{s} に対して i を増加させる. これによって図 3.12 (b). 示すように，CPD で表される計算資源の不足を補うことが可能になる．

Algorithm 1 アドミッション制御アルゴリズム

Require: $G_{sd} = (T_{sd}, E_{sd})$, $R_{sd.s} = (P_{sd.s}, L_{sd.s})$, $md_{sd.s}$
Ensure: Bool Value that indicates if a sensor data packet is acceptable, X_0 (if acceptable), CPD (if not acceptable)
procedure ISACCEPTABLE($G_{sd}, R_{sd.s}, md_{sd.s}$)
$P \leftarrow P_{sd.s}$
$X \leftarrow \mathbf{O}$
$CPD \leftarrow 0$
$falsecount \leftarrow 0$
determine values of md_c, md_p, md_a so that $md_c + md_p + md_a = md_{sd.s}$
for all $t \in T_{sd}$ **do**
$md \leftarrow md_c/md_p/md_a$ depending on type of t
find $p \in P$ s.t. $proctime(t, p) \leq md$
if such p is found **then**
$x_{t,p} \leftarrow 1$
$P \leftarrow P \backslash \{p\}$
else
$CPD \leftarrow CPD + comp(t)$
$falsecount \leftarrow falsecount + 1$
end if
end for
if $falsecount = 0$ **then**
$X_0 \leftarrow X$
return *true*
else
return *false*
end if
end procedure

地産地消タスクスケジューリング

センサデータパケット sd が $isAcceptable$ によって受け入れられると，タスクグラフ G_{sd} がスケジューリングされる．このステップでは，サービスプロバイダは，ユーティリティ関数 $\sum_{sd \in SD} U(sd)$ の合計を可能な限り増加させるために，$isAceptable$ によって導出された解よりも，優れたタスク割り当てプランを見つけようとする．そこで，本研究では，メタヒューリスティック検索を用いて初期割り当てプラン X_0 を改善するアルゴリズムを提案する．具体的には，タブー検索アルゴリズム [50] を使用してタスク割り当てプランを調整する．

アルゴリズム 2 は，タブー探索に基づいて提案された現場タスクスケジューリング手法を示す．入力パラメータは，センサデータパケット sd 対するタスクグラフ G_{sd}，リソースサブグラフ $R_{sd.s}$，および $isAcceptable$ によって導出された初期タスク割り当て X_0 である．出力は，タブー探索によって調整されたタスク割り当てプラン X_{sd} である．

アルゴリズム 2 では，最初に変数 $X, X^*, Tabu, cnt$ が初期化される（1〜4 行

Algorithm 2 タブー探索に基づく地産地消タスクスケジューリング

Require: $G_{sd} = (T_{sd}, E_{sd})$, $R^{i+}_{sd.s} = (P^{i+}_{sd.s}, E^{i+}_{sd.s})$, $X_{sd.init}$
Ensure: X_{sd}: Task Assignments
```
/*Initialization*/
X, X* ← X_0
Tabu ← O
cnt ← 0
while cnt ≤ MAXCNT do
    reduce_best ← ∞
    /* obtain the best neighbor plan*/
    X', X̄ ← X
    t' ← randselect(T_sd), p' ← p(t')
    for all t ∈ T_sd do
        for all p ∈ P^{i+}_{sd.s} | p ≠ p(t) do
            if Tabu_{t,p} = 0 & comp(t) ≤ remcap(p) then
                x̄_{t,p(t)} ← 0, x̄_{t,p} ← 1
                gain ← Utility(X̄) − Utility(X)
                if gain > gain_best then
                    gain_best ← gain
                    t' ← t, p' ← p
                end if
            end if
        end for
    end for
    x'_{t',p(t')} ← 0, x'_{t',p'} ← 1
    /* update the best plan*/
    if gain_best ≥ 0 then
        if Utility(X') > Utility(X*) then
            X* ← X'
        end if
    else
        Tabu_{t',p(t')} ← 1
    end if
    X* ← X'
    cnt ← cnt + 1
end while
X_sd ← X*
return X_sd
```

目）．ここで X, X^* は現在のタスク割り当てプランと最適な割り当てプランである．$Tabu$ はタブー行列で cnt は反復回数を制御するカウンタを示す． $p(t)$ はタスク t が割り当てられているプロセッサを示す．メインプロセスは 5 行目〜28 行目にあり，5 行目の $MAXCNT$ は反復回数を指定する定数である．8 行目では，最良近傍解 X' と近傍解 $\bar{X}$ が X で初期化される． 9 行目では，変数 t' と p' が初期化されている．ここで，t' と p' は，それぞれ T_{sd} と X で割り当てられたプロセッサからランダムに選択されたタスクである.10 行目〜19 行目のループは，現在の解 X で 1 つのタスク割り当てを変更することによって生成されたすべての近傍解 $\bar{X}$ の中から選択された最良の近傍解 X' を見つける．具体的には，p に t を実行する

ための残りの計算能力（$emcap()$）があれば，すべてのタスク割り当て $x_{t,p(t)} = 1$ は，新しい割り当て $\bar{x}_{t,p} = 1$ に変更される（12〜13 行目）．次に，新しいタスク割り当てプランによってタスクグラフを実行するための全体的なユーティリティが増加すると，X' が更新される（14-17 行目と 20 行目）．14 行目では，$Utility(X)$ は，現在のタスク割り当てプラン X に基づいてタスクグラフを実行するときの QoE を推定する関数である（タスクグラフを処理するための遅延が計算され，その後，遅延に基づいてユーティリティが計算される）．22 行目〜27 行目では，最良解 X^* は，現在の最良解 X^* よりも優れている場合，最良近傍解 X' によって更新される．反復が停止した後，現在の最良解 X^* が最終解として X_{sd} に代入される（29〜30 行目）．

3.5 評価実験

本節では，IFoT プラットフォームの有効性を検証すべく，実機ベースのプロトタイプシステムを用いた性能評価実験と，大規模エリアでの展開を想定したシミュレーションによる評価実験の結果を示す．以下では，それぞれの評価実験について記述する．

3.5.1 実機プロトタイプシステムによる評価実験

ここでは，IFoT プラットフォームの PoC（Proof of Concept）プロトタイプの実装について説明するとともに，2.5.2 章で説明したベルト型 IoT デバイスを用いた行動認識サービスを題材とした，プロトタイプシステムの性能評価実験の結果を示す．

IFoT プラットフォームのプロトタイプ

IFoT プラットフォームの PoC（Proof of Concept）プロトタイプシステムの概要を図 3.13 に示す．このように，サービスプロバイダ，ブローカー，計算資源プロバイダ，データプロバイダ，サービスコンシューマという 5 つのコンポーネントからなるシステムとなっている．本プロトタイプは，図 3.10 に示したデータ処理ワー

クフローの動作確認を行う目的で実装されているため，リソースマネージャの統合は今後の課題である．各コンポーネント間の通信プロトコルを図 3.14 に示す．機械学習などのデータ分析ライブラリとの親和性を考慮し，プログラミング言語は，Python を採用した．また，サービスにおける Pub/Sub メッセージングやハートビートメッセージなど各コンポーネント間の通信プロトコルは，軽量のメッセージング通信用フレームワークである ZeroMQ [65] を活用して，設計・開発を行っている．

行動認識サービスを題材とした性能評価実験

2.5.2 章で説明したベルト型 IoT デバイスを用いた行動認識サービスを題材として，上記で説明した IFoT プラットフォームの PoC プロトタイプの性能評価実験を行った．評価実験時のシステムの基本ワークフローは，以下の通りである．まず，サービスコンシューマは，行動認識サービスで得られるコンテキストデータをあらかじめサブスクライブする．次に，データプロバイダは，ベルト型 IoT デバイスから取得できるセンサデータをサービスプロバイダに対して，逐次パブリッシュする．サービスプロバイダは，受信したセンサデータパケットをブローカを介しながら，利用可能な計算資源プロバイダにスケジューリングする．各計算資源プロバイダは，サービスプロバイダからの Invoke メッセージに応じて行動認識タスクを

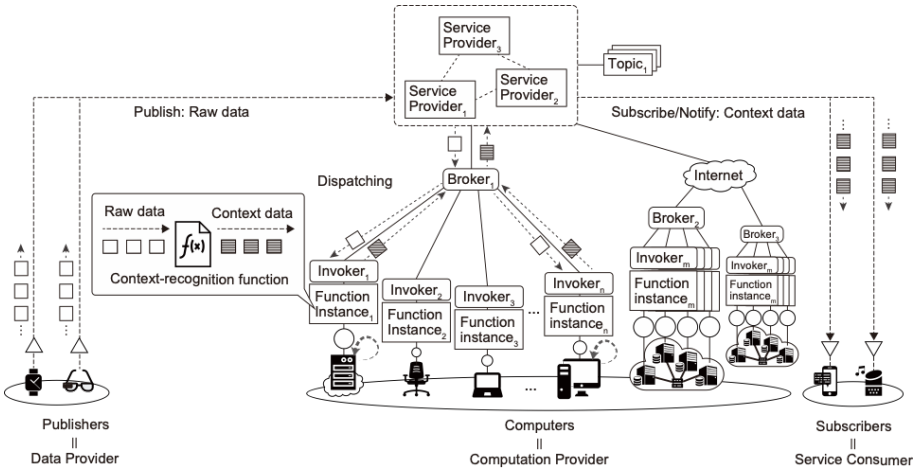


図 3.13: IFoT プラットフォームの PoC プロトタイプ

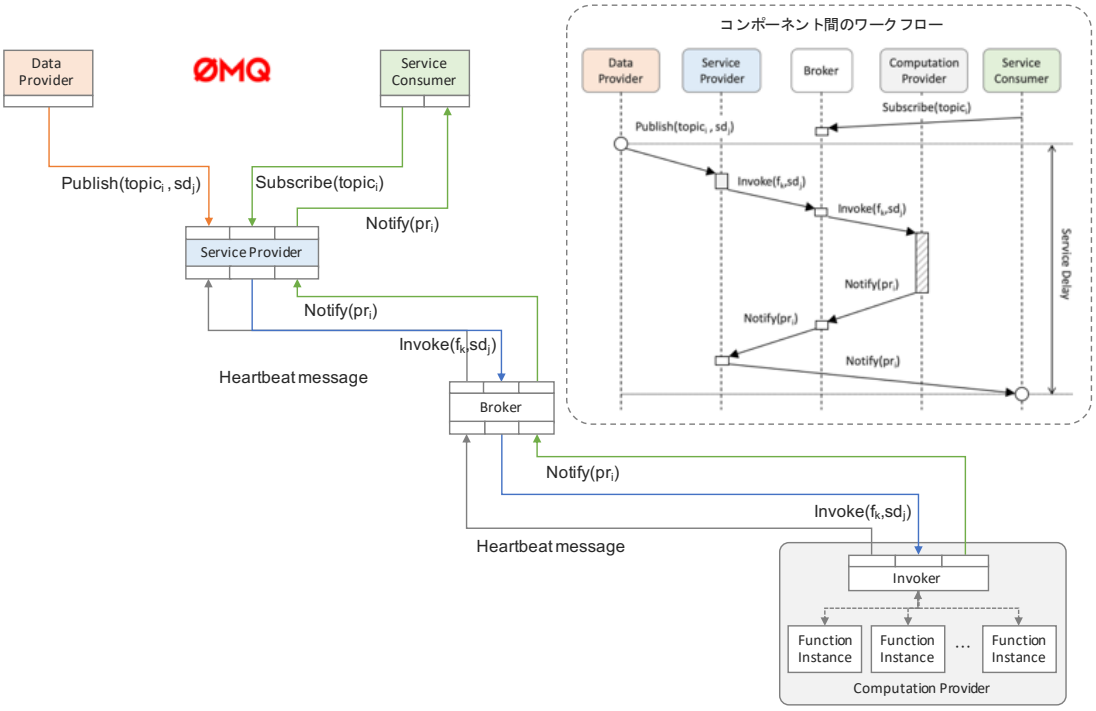


図 3.14: PoC プロトタイプのプロトコル実装

実行し，処理結果としてユーザのコンテキスト推論結果をサービスプロバイダに返却する．評価指標は，データプロバイダがセンサデータパケットのパブリッシュしてから，そのセンサデータパケットに対する行動認識結果であるコンテキストデータがサービスコンシューマに届くまでのサービス遅延時間である．図 3.15 に，複数の計算資源毎に 2 (req/s) の頻度でパブリッシュされた 100 個のセンサデータパケットを処理した時のサービス遅延時間の比較を示す．横軸は，デバイスの種類，縦軸はサービス遅延時間（s）となっている．青色のプロットが処理時間，赤色のプロットが通信遅延時間を示している．1 から 5 番がローカル（奈良先端大のネットワーク内）のデバイス，6,7 番がクラウド（Amazon EC2 インスタンス in US を使用）のサーバとなっている．クラウドのリソースを使った処理は，遅延時間の大半を通信遅延となっている一方で，ローカルでのデータ処理は通信遅延が少ない結果となっている．このことからローカルの計算資源を活用したデータ処理が通信遅延

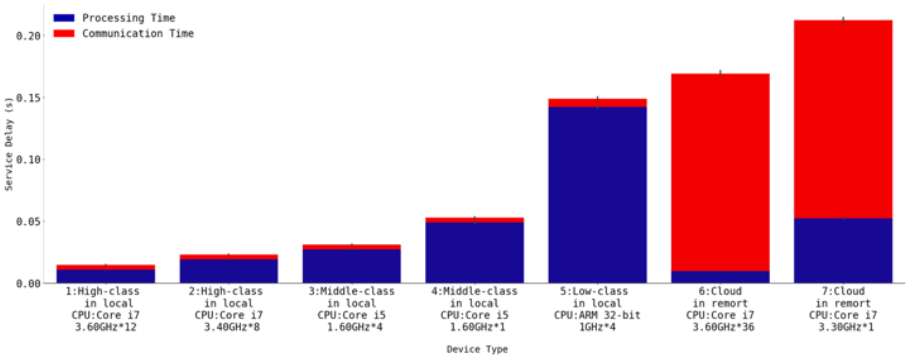


図 3.15: ローカルとクラウドにおけるサービス遅延時間の比較

時間を大幅に削減できることを示している．5 番は手のひらサイズのシングルコンピュータであるラズベリーパイによる結果であるが，200ms 以内で行動認識処理を実行できることが確認できる．また，5 番の結果は，6 番のリモートに存在する高性能なクラウドサーバで実行した結果よりも優れていることから，ローカルの IoT デバイス群を活用したデータ処理パラダイムの有効性を示している．

IFoT のプロトタイプシステムのスケーラビリティおよびオーバーヘッドを確認する目的で，上記のベンチマーク結果に基づいて，100 台規模の計算資源プールをエミュレーションし，IFoT プラットフォームのサービスプロバイダに対する負荷試験を行った．まず，Low クラスデバイス 60 台，Middle クラスデバイス 30 台，High クラスデバイス 10 台の計算資源プロバイダ，3 台のブローカー，3 台のサービスプロバイダからなる IFoT システムを起動する．そして，図 3.16 に示すように，サービスプロバイダにパブリッシュするセンサデータパケットを徐々に増加させた状態で，IFoT システムが提供する行動認識サービスの性能（センサデータパケットを行動認識結果に変換し，サービスコンシューマに届けるまでの遅延時間）を計測した．その結果を図 3.17 に示す．結果から，サービスの負荷が向上した場合でも，サービス遅延時間をおおむね 250ms 以下に抑えながら，データ処理できていることが確認できる．また，実験結果から (a) 区間においては，18 台の計算資源プロバイダを用いてデータ処理を実行しているのに対して，計算需要が増加した (b) 区間においては，100 台の計算資源プロバイダを用いてデータ処理を実行していることが分かった．このことから，サービスプロバイダは，計算需要に応じて，

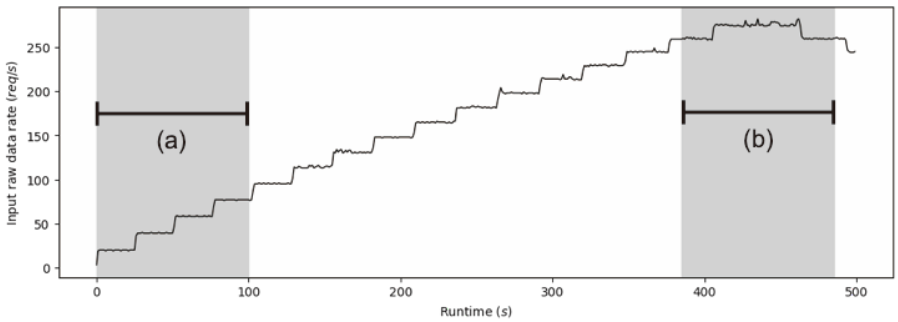


図 3.16: サービスプロバイダが送信されるセンサデータ

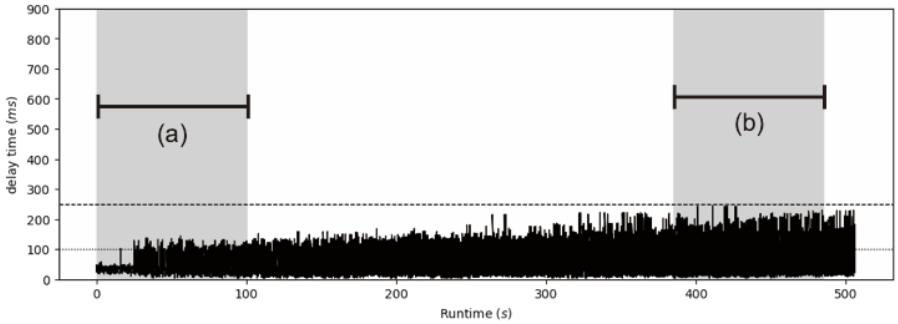


図 3.17: 図 3.16 のセンサデータを行動認識処理した際の処理遅延時間

利用する計算資源の量を調整しつつ，サービスレベルアグリーメントを満たすように意思決定していることが確認できた．

3.5.2 シミュレーションによる評価実験

ここでは，大規模エリアで IFoT プラットフォームを展開することを想定し，シミュレーションによる評価実験の結果を示す．これらのシミュレーションでは，スケジューリング手法のパフォーマンスを評価するためのメトリックとして，遅延時間，ローカルで受け入れられるセンサデータパケットの数，および QoS 効用関数の値を使用する．

表 3.1: シミュレーションのリソースパラメータ

パラメータ	値 (s)
Number of workers (nodes)	4000
Number of High-class-nodes	1000
Number of Low-class-nodes	3000
Number of nodes per section	0 to 24
Processing power of High-class-nodes	1000 to 3000
Processing power of Low-class-nodes	500 to 1000
Network speed of High-class-nodes to R-broker	5 to 50 (Mbps)
Network speed of Low-class-nodes to High-class-nodes	50 to 500(Mbps)

表 3.2: シミュレーションのサービスパラメータ

パラメータ	値 (s)
Input sensor data size range of Home service	1 to 20 (MB)
Collecting task cost of Home service	100
Processing task cost of Home service	1000
Aggregating task cost of Home service	100
Input sensor data size range of Office service	5 to 50 (MB)
Collecting task cost of Office service	500
Processing task cost of Office service	2000
Aggregating task cost of Office service	300
Input sensor data size range of Tourism service	10 to 200 (MB)
Collecting task cost of Tourism service	1000
Processing task cost of Tourism service	5000
Aggregating task cost of Tourism service	500

シミュレーション設計とパラメータ設定

提案アルゴリズムの有効性を明らかにするために，図 3.18 のような仮想的な IoT 環境を作成し，シミュレーションによる比較実験を行った．比較実験に用いたシミュレータは SimPy と NetworkX という Python フレームワークを使って実装さ

表 3.3: シミュレーションケース I (平日の昼間)

パラメータ	値 (s)
Simulation duration	1 (h)
Simulation runs	10 Runs
Sensor data packet arrival rate per section of Home service λ_h	1 (sd/min)
Sensor data packet arrival rate per section of Office service λ_o	6 (sd/min)
Sensor data packet arrival rate per section of Tourism service λ_t	2 (sd/min)

表 3.4: シミュレーションケース II (休日の昼間)

Parameter	**Value(s)**
Simulation duration	1 (h)
Simulation runs	10 Runs
Sensor data packet arrival rate per section of Home service λ_h	2 (sd/min)
Sensor data packet arrival rate per section of Office service λ_o	1 (sd/min)
Sensor data packet arrival rate per section of Tourism service λ_t	6 (sd/min)

表 3.5: シミュレーションケース III (夜間)

Parameter	**Value(s)**
Simulation duration	1 (h)
Simulation runs	10 Runs
Sensor data packet arrival rate per section of Home service λ_h	6 (sd/min)
Sensor data packet arrival rate per section of Office service λ_o	1 (sd/min)
Sensor data packet arrival rate per section of Tourism service λ_t	2 (sd/min)

れている．シミュレーションは，R ブローカ，IoT デバイス（*High-class-nodes*, *Low-class-nodes*），地域 IoT サービス，およびセンサデータパケットで構成されている．High クラスノードおよび Low クラスノードは，それぞれ高性能のラップトップなどのハイパワーな IoT デバイス，Raspberry Pi などのローパワーな IoT デバイスを想定してモデル化されている．シミュレーションでは，対象の地域エリアのリソースグラフ $R = (P, L)$，サービスの集合 S，各サービスのサービスエリア

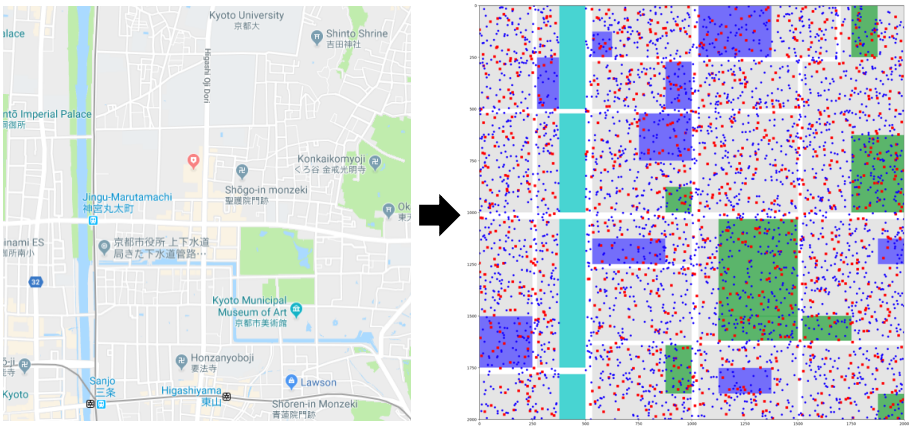


図 3.18: シミュレーション上の地域 IoT エリア

$area(s)$，各サービスに対するセンサデータパケットの集合 SD_s，各センサデータパケットに対応するタスクグラフの集合 $\cup_{sd \in SD_s} G_{sd} (= (T_{sd}, E_{sd}))$ が与えられる．各センサデータパケット sd とそれに対応するタスクグラフ G_{sd} は，ポアソン到着プロセスに基づいて生成される．

シミュレーションでは，まず，サービスエリアごとにサービスプロバイダが選択され，サービスのリソースグラフが生成される．次に，各サービスエリアのサービスプロバイダがセンサデータパケットを受信し，各パケット毎のタスクグラフを生成し，アドミッションコントロール（アルゴリズム 1）によって評価する．受け入れられた場合，タスクグラフの各タスクをプロセッサに割り当てることにより，以下で説明する各アルゴリズムを使用してタスクグラフがスケジュールされる．†

シミュレーションのパラメータを表 3.1，表 3.2 に示す．R-ブローカー，High クラスノード，Low クラスノード，IoT サービスの数は，それぞれ 1, 600, 1400, および 10 である．各デバイスの処理能力およびデバイス間の通信速度は，これまでの研究での実測値 [66,67] に基づいて決定されている．2 つの従来のアルゴリズム：ランダムタスクスケジューリング，グリーディタスクスケジューリングが，2 つの

†複数のタスクがプロセッサに割り当てられている場合，タスクの負荷に応じて処理能力が低下すると想定している．

提案されたアルゴリズム：タブサーチ，アダプティブスケールアウト付きタブーサーチに対するベースラインとして使用される．

シミュレーションにおける 3 つのケースシナリオ

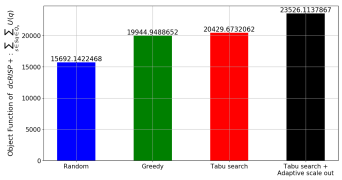
シミュレーションのターゲット地域として，図 3.18 の左に示すように，京都の東山地区にある 2Km $times$ 2Km の観光エリアを選択した．実際のターゲットエリアは，図 3.18 の右に示すように，住宅（ホーム）エリア，オフィスエリア（商店街を含む），および観光エリア（寺院と神社）という 3 つの異なるエリアで構成されている．3 つの異なる地域 IoT サービス:（1）ホームサービス，（2）オフィスサービス，（3）観光サービスがこれらの領域でそれぞれ提供されると仮定する．これらの地域の人口は日時によって異なるため，表 3.3 に示すように，平日の昼間，休日の昼間，夜間という 3 つのケースについて，各サービスの 3 つのセンサデータパケット到着率を設定する．センサデータパケット到着率は各ブロック（125m $times$ 125 m）で定義されている．ターゲット領域全体が 8 つの $times$ 8 ブロックに分割される．1 時間のシミュレーション時間で 10 回シミュレーションを実行する．

テーブル 3.2 に示されているパラメーターを使用して，ポアソン到着プロセスに基づいて各サービスにセンサデータパケットが送信される．この表では，収集，処理，集計の各タスクのデータサイズと計算コストなどのパラメータが設定されている．各サービスの計算コストは，モーションセンサーを使用した不審者の監視，ウェアラブルデバイスを活用したオフィスワーカーのモニタリング，および観光エリアに存在する観光客がアップロードする写真/ビデオデータに対するプライバシー除去のためのモザイク処理サービスをそれぞれ想定して決定されている．

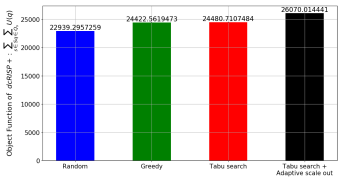
評価手法と評価指標

シミュレーションによって，評価する手法として，(1) ランダムタスクスケジューリングと (2) グリーディタスクスケジューリングという 2 つのベースライン手法と２パターンの提案手法 (3)(4) を設定した．

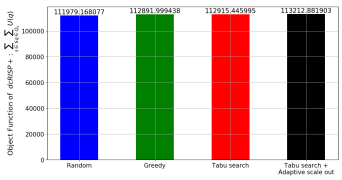
1. ランダムタスクスケジューリング: 各サービスのリソースサブグラフからランダムにプロセッサを選出し，タスクを割り当る．
2. グリーディタスクスケジューリング：各サービスのリソースサブグラフから



ケース I (平日の昼間)

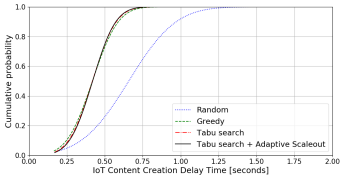


ケース II (休日の昼間)

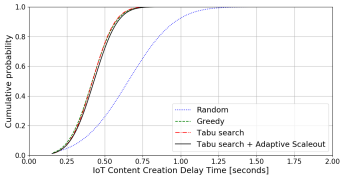


ケース III (夜間)

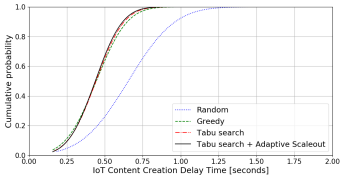
図 3.19: 各ケースにおけるすべてのセンサデータパケット SD のユーティリティ関数 $U(sd)$ の合計



ケース I (平日の昼間)



ケース II (休日の昼間)



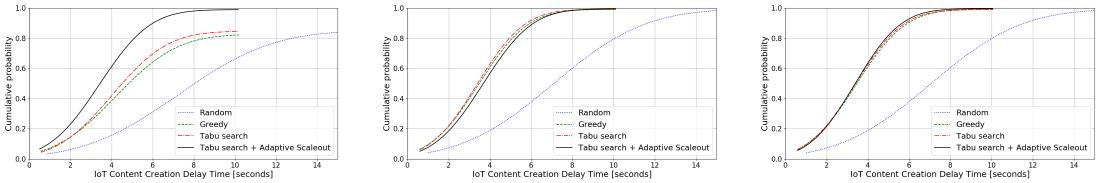
ケース III (夜間)

図 3.20: スマートホームサービスの遅延時間の累積分布関数（CDF）曲線

グリーディ手法に基づいてプロセッサを選択し，タスクを割り当る．

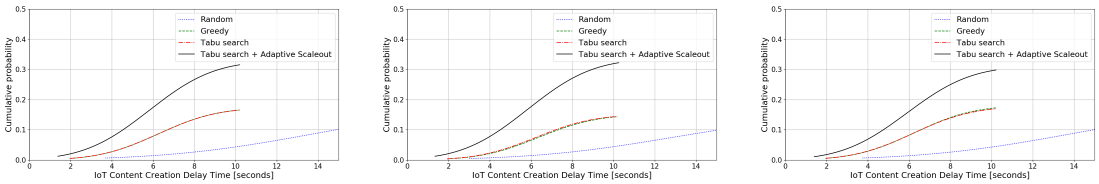
3. タブー探索に基づくタスクスケジューリング: 各サービスのリソースサブグラフから 3.4.3 節で説明したタブー探索アルゴリズムに基づいてプロセッサを選択し，タスクを割り当る．
4. タブー探索に基づくタスクスケジューリング + 適応型現場スケールアウト．（３）の手法に加えて，サービスの負荷状況に応じて適応型スケールアウトを実施し，リソースサブグラフの拡張を行う．

各手法のパフォーマンスを評価するための評価指標は，遅延時間，受け入れられたセンサデータパケットの数，および効用関数の値である．効用関数の値は，pd=10(s) および md=30(s) の効用関数曲線に基づいて計算される．



ケース I (平日の昼間)　　ケース II (休日の昼間)　　ケース III (夜間)

図 3.21: スマートオフィスサービスの遅延時間の累積分布関数（CDF）曲線



ケース I (平日の昼間)　　ケース II (休日の昼間)　　ケース III (夜間)

図 3.22: スマート観光サービスの遅延時間の累積分布関数（CDF）曲線

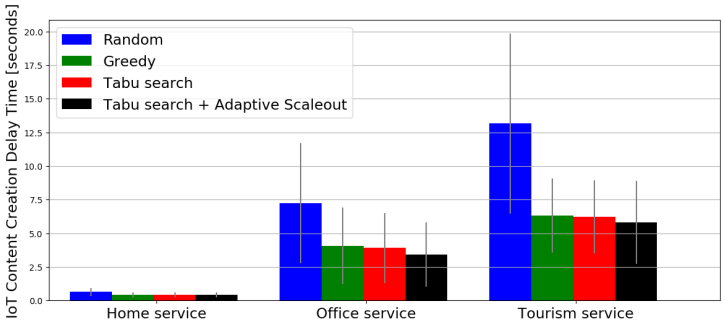


図 3.23: 95 %信頼区間の各サービスの平均遅延時間 l

評価結果と考察

■QoS 効用関数の値　図 3.19 は，それぞれ 4 つのメソッドのすべてのセンサデータパケットのユーティリティ関数値の合計を示している．結果は，提案された方法 1 および 2 がすべての場合においてベースライン方法よりも優れた QoS を持って

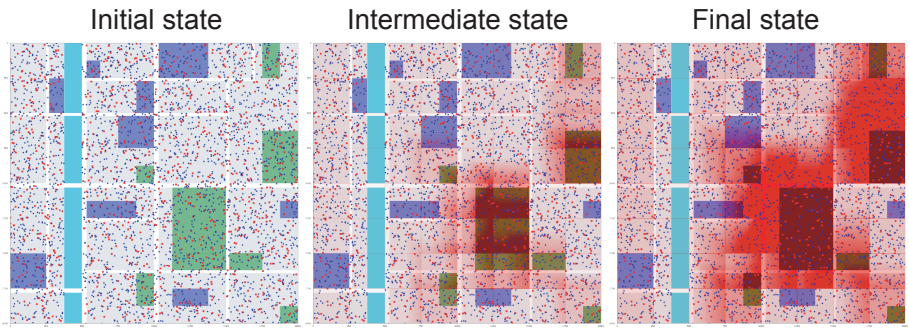


図 3.24: ケース 2 におけるアダプティブスケールアウトのスナップショット

いることを示唆している．また，適応スケールアウトを行う提案手法 2 は，提案手法 1 よりもすべてのケースで QoS を達成した．

■遅延時間とセンサデータパケットの受託率　各手法の平均遅延時間は，ランダム：17.83 秒，グリーディ：14.95 秒，提案手法 1 ：9.79 秒，提案手法 2 ：10.07 秒である．この結果は，提案手法 1 ，2 は，ランダムやグリーディよりも遅延時間が短いことを示している．また，遅延時間の標準偏差は，ランダム：11.33 秒，グリーディ：5.19 秒，提案手法 1 ：4.38 秒，提案手法 2 ：4.30 秒である．この結果は，提案手法 1 ，2 は，ランダムやグリーディよりも各センサデータパケットに対する遅延時間の差が小さいこと示している．

図 3.20, 3.21, 3.22 は，各ケースの各メソッドのすべてのセンサデータパケットの遅延時間の累積分布関数（CDF）曲線を示している．

各グラフの各曲線の最高点は，センサデータパケットの受託率を示している．すべての図で，提案された方法 2（適応スケールアウトによるタブ探索）の許容率（最高点の値）は，他の方法よりも高くなっている．提案された方法 2 は，Office サービスのケース 1（平日昼間）で最も効果的に機能したことが確認できる．

観光サービスでは，提案された方法 2 が他の方法よりもはるかにうまく機能している．ただし，ターゲットエリアのデバイスの処理デバイスが十分ではなかったため，すべてのメソッドの受け入れ率は低くなる（0〜30 %）．適応スケールアウトに含めるクラウドリソースを検討する場合，受け入れ率を上げることができるが，これは今後の作業の一部となる．これは，提案された方法 2 の適応スケールアウト

が，高い計算要求を必要とするこのような場合に効果的であることを意味する．

ほとんどの図で，CDF 曲線の遅延時間は，提案された方法 1 および 2（適応型スケールアウトを使用したタブ検索およびタブ検索）の方が低いことが確認できる．ケース 2（ホームおよびオフィスサービス）では，提案された方法 2 のパフォーマンスは，貪欲およびタブー検索（適応スケールアウトなし）よりも低くなっている．これらの結果は，家庭およびオフィスサービスに属する計算リソースが，計算需要が高い観光サービスにも使用されたという適応スケールアウトの影響を受けている．ただし，図 3.19 に示されているように，提案された方法 2 は QoE 全体よりも優れているため，これは懸念事項ではない．

図 3.23 は，3 つのシミュレーションケースの結果を含む各サービスの各メソッドの平均遅延時間を示している．エラーバーは 95 %信頼区間（CI）を示す．この結果は，提案された方法 1 および 2 が，ランダムおよび貪欲な方法よりも遅延時間が短いことを示している．また，提案手法 2 の遅延時間は，提案手法 1 よりも短いことが確認できる．

ケース 2（休日の昼間）のシミュレーション中の適応スケールアウトのスナップショットを図 3.24 に示す．レッドゾーンは，各サービスのリソース選択領域である．図 3.24 に示すように，送信されたセンサデータパケットに応じて，一部のサービスはサービスエリアの適応スケールアウトを実行する（リソース選択エリアはサービスエリアより突出している）．ケース 2 では，観光サービスがより多くのセンサデータパケットを受信し，計算需要が他の 2 つのサービス（自宅，オフィス）よりも高いため，観光サービスが適応スケールアウトをより頻繁に実行することが想定される．初期状態では，図 3.24 の左側に示すように，適応スケールアウトはまだどのサービスエリアにも適用されていない．図 3.24 の中間状態では，観光サービスはサービスエリアをスケールアウトし，リソースの選択はホームサービスエリアと重複していることが確認できる．これは，観光サービスがホームサービスに属するリソースを使用することを意味する．図 3.24 の最終状態では，観光サービスのリソース選択エリアがオフィスサービスエリアに到達した．図 3.24 は，センサデータパケットを処理するのに十分なリソースのないサービスが，適応スケールアウトを使用してリソース選択領域を適応的に拡張していることを示している．ケース 2 では，リソース選択領域の重複が発生するため（図 3.24 の中間状態と最終状

態)，図 3.20，3.21，3.22 に示すように，家庭およびオフィスサービスでの適応スケールアウトを使用したタブー検索のパフォーマンスは，欲張りおよびタブー検索よりもわずかに劣ることが確認できる．

評価結果のまとめ

評価結果を以下のように要約する．

- タブー探索を用いた提案手法 1 ， 2 は，各センサデータパケットに対する遅延時間を短くすることを確認した（図 3.20, 3.21, 3.22）．遅延時間の短縮は，サービスの質の向上を意味する．このことから，タブー探索に基づく地産地消タスクスケジューリングは，各センサデータパケットに対するサービスの質の向上に寄与している．
- 適応型スケールアウトを用いた提案手法 2 は，センサデータパケットの受託数を増加することを確認した（図 3.20, 3.21, 3.22）．センサデータパケット受託数の増加は，サービス量の向上を意味する．このことから，適応型スケールアウトはサービス量の向上に寄与している．
- 提案手法 2 は，タブー探索によって各センサデータパケットに対するサービスの質を，適応型スケールアウトによってサービスの量を向上させている．その結果，最も高い QoS を得ることができた．

3.6 まとめ

本章では, 地域に存在する IoT デバイス群をセンサデータプロバイダ，計算資源プロバイダ，サービスコンシューマとして抽象化し，それらのリソースをセンサデータ処理サービスの需要に応じて調整・分配しながら，一つのサービス系として弾力性のあるデータ処理を実現する IFoT プラットフォームの設計を示した．また，(a) 対象の地域エリアに存在する IoT デバイス群で構成されるリソースグラフ，(b) 地域エリアで展開される IoT サービスの集合，(c) 一定時間にユーザから各 IoT サービスに対して送信されるセンサデータパケットの集合，(d) センサデータパケットに対応する処理タスクグラフの集合，を入力として，QoS の最大化に向

けてセンサデータストリームを素早く処理するための (e) タスク実行スケジュールを求める遅延制約付き地域 IoT サービスプロビジョニング問題を定式化した．そして，この問題を解決すべく，(1) IoT サービスの計算需要に応じて動的に計算資源の選択エリアを拡張する適応型スケールアウトと (2) タブー探索に基づく地産地消タスクスケジューリングからなるサービスプロビジョニング手法を提案した．IFoT プラットフォームの有効性を示すために，前章で紹介したベルト型 IoT デバイスを用いた行動認識サービスを題材とした実機の IFoT プロトタイプシステムによる評価実験と 4000 台の IoT デバイスが展開された地域エリア（エリア：2km × 2km）で IFoT プラットフォームを展開することを想定したシミュレーションによる評価実験の結果を示した．プロトタイプシステムを用いた評価実験では，ローカルの IoT デバイス群を有効活用することで 250ms の遅延要求を満たすタイムリーな行動認識サービスを実現可能であることを確認し，シミュレーションによる評価実験では，サービスの需要が増加した場合でも，近隣のエッジ IoT デバイスにオフローディングするアダプティブスケールアウトを使用することで QoS を維持できること確認した．

図 3.25 に，序論の図 1.2 で示した IoT システムのタクソノミーに対する IFoT プラットフォームのカバー範囲を示す．このように，現状の IFoT プラットフォー

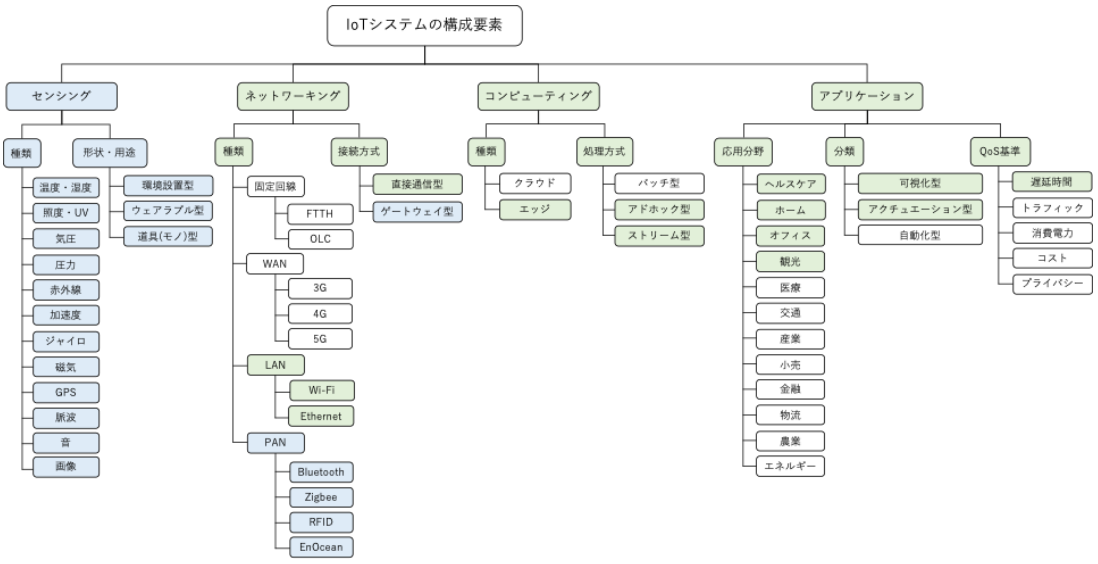


図 3.25: 図 1.2 のタクソノミーに対する IFoT プラットフォームのカバー範囲

ムは，主にヘルスケア，スマートホーム，オフィス，観光分野において，ユーザや空間の状況を反映したデータを可視化したり，状況に応じて何らかのアクチュエータを作動させるコンテキストアウェアアプリケーションを基本ターゲットとしている．そして，センサデータが提供されてから，処理結果を提示するまでの遅延時間を QoS 基準として採用した．センサデータ の処理方式は，ローカルのエッジ IoT デバイス群の計算資源を活用した，即時性の高いストリーム処理や，必要に応じてデータ処理を実行するアドホック処理をサポートしている．センサデータ処理を行うデバイスは，地域エリアの LAN にダイレクトで接続している直接通信型のデバイスを想定している (対応箇所：緑色) ．IFoT プラットフォームに対してセンサデータを送るためには，PAN 通信経由で受信したセンサデータをゲートウェイデバイス上で IFoT プラットフォーム用の Pub/Sub プロトコル（IP ネットワーク）に変換する必要がある．現状，センサデバイスから何らかの PAN を通じてゲートウェイデバイスまでデータを転送するプログラムに関しては，IFoT プラットフォームのサポート対象外であり，センサデバイスに応じて個別のプラグインを開発する必要がある．２章で提案した SenStick プラットフォームは，Node-SenStick というゲートウェイ用のライブラリがすでに開発されているため，このプログラムを拡張することによって，IFoT プラットフォームと連携させることが可能である (対応箇所：青色).

今後，WAN が 4G から 5G に移行することで，フロント側の通信基地局の性能が向上し，大容量，超多数端末接続，超低遅延のネットワークを提供することが可能になり，IoT だけでなく，自動運転，AR，VR，遠隔制御など即時性の求められるアプリケーションがこれまで以上に増加すると予想される．また，現在，ゲートウェイ型のインターネット接続を採用している端末の多くが，5G の SIM を搭載して直接インターネットに接続する直接通信型に移行することが考えられる．そのため，現状の IFoT プラットフォームは，地域 LAN に接続されたエッジ IoT デバイスへのオフローディングを対象としているが，今後は，5G を始めとした WAN に接続したエッジ IoT デバイスへの対応も進める予定である．また，スマートフォンや車などの移動体端末が持つ計算資源を状況に応じて活用するオフローディング手法の検討を進める．IFoT プラットフォームにとって，5G の普及は，より多数のローカルデバイスをネットワークに接続することが可能になり，デバイス間の通信

も高速化されることから，大きなメリットがあると考える．複数の IoT デバイス群を用いたデータ処理アプローチの懸念事項として，クラウドなど専用の処理サーバと比較して電力効率が悪いため，データ処理における消費電力が増加してしまうという点が考えられる．しかしながら，5G が普及した時代では，クラウドへのデータ送信電力の増加やコアネットワークにおけるトラフィックの増大も解決すべき重要な課題であるため，これらの問題を解決するためにも，センサデータの発生源付近で，可能な限りデータ処理を実行し，処理結果だけをクラウドに集約するというアプローチが必要不可欠である．提案する IFoT プラットフォームは，既存のクラウドコンピューティングや通信基地局に設置されたエッジクラウドを置き換えるものではなく，あくまで共存する立場であるため，アプリケーションの性質に応じて，エッジ IoT，エッジクラウド，コアクラウドを柔軟に使い分けることできるデータ処理フレームワークの検討を進める予定である．そのためにも，サービスの遅延時間だけでなく，計算資源や通信資源を利用する際に発生する金銭的コスト，消費電力，サービスの提供者やサービスの利用者の嗜好をモデル化し，これらを多目的最適化問題として捉えながら，それぞれのトレードオフを分析していく予定である．また，実機の大規模テストベットを活用した長期間の実験を実施し，サステイナビリティの観点からも IFoT プラットフォームの改良を進める．さらに，データ処理電力効率の観点から，CPU だけでなく，GPU や FPGA を搭載した IoT デバイスの統合も考慮し，より異種性のある IoT デバイス群から構成されるリソースプールをセンサデータ処理に有効活用する方法を検討する．また，プライバシーやセキュリティーを考慮したデータ処理手法や通信プロトコルの統合方法についても検討を進める予定である．

第 4 章　結論

本研究では，地域における IoT センサデータの生産を促進するとともに，収集された IoT センサデータを地域に存在する計算資源を活用して即時に処理・分析・応用する「地産地消」を基本コンセプトとして，地域に存在する IoT デバイスを最大限に有効活用しながら，IoT 技術の力で地域に顕在化する様々な課題を解決することを可能にする「IoT センサデータ地産地消基盤の実現」を目指した研究に取り組んだ．IoT センサデータ地産地消基盤を実現するためには，あらゆるフィールドでセンサデータの収集を可能にすると共に，クラウドやエッジクラウドの利用が難しい地域郊外やルーラルエリアにおいても利用可能なローカルの計算資源を用いてデータ処理サービスを提供できる環境を整えることが重要である．そのために，本論文では，2 つの問題点（1）実環境で利用できる IoT デバイスの選択肢が少なく，センシング対象が限られてしまう，（2）IoT センサデータのタイムリーな処理・分析・応用ができていないに着目し，それぞれを解決する 2 つの研究課題 A: 実用可能な IoT デバイス開発とセンサデータ収集の簡略化，B: エッジ IoT デバイス群による弾力性のあるデータ処理の実現に取り組んだ．以下，各研究課題に対して，得られた成果と今後の展望について述べる．

研究課題 A に関して，8 つのセンサ（加速度，ジャイロ，磁気，光，UV，温度，湿度，圧力），BLE 通信モジュール，フラッシュメモリ，バッテリー，充電回路を搭載し，どこにでも組み込み可能な超小型のマルチセンサボード，スマートフォンや PC などマルチプラットフォームに対応したデータ収集用のソフトウェア，自由にデバイスの形状をデザイン可能な 3D プリンタ用のケースデータという，新しい組み合わせによって，誰でも簡単に目的とするフィールドに適した IoT デバイスを試作でき，素早くセンサデータを収集できる環境を提供する SenStick プラットフォームを設計・開発した．そして，ベルトや箸の IoT デバイス化といったケーススタディを通じて SenStick プラットフォームの有効性を評価し，SenStick を活用することで開発プロセスが簡略化されることを示した．また，SenStick を用いて様々なモノを IoT デバイス化することができ，試作された IoT デバイスを用いて，長時間の行動データ収集および高精度（F 値 0.95）の日常行動認識を実現できることを示した．SenStick プラットフォームを利用したユーザのアンケートを通じて，

IoT 関する研究開発と教育という 2 つの側面で有用性があることを確認した．今後は，SenStick ハードウェアのエナジーハーベスト化や，SenStick ハードウェア内のプロセッサを活用したデータ処理やデバイス間のマルチホップ通信などの検討を進める．また，多様な IoT デバイスを用いてさまざまな行動コンテキストを認識するだけでなく，どんなタイミングで，どのようなアクチュエーションを行うと人々の行動変容が促せるのかといった観点から，SenStick をベースとした新しい IoT システムをデザインし，様々な実証実験を実施する予定である．

研究課題 B に関して，地域に存在する IoT デバイス群をセンサデータプロバイダ，計算資源プロバイダ，サービスコンシューマとして抽象化し，それらのリソースをセンサデータ処理サービスの需要に応じて調整・分配しながら，一つのサービス系として弾力性のあるデータ処理を実現する IFoT プラットフォームを設計し，プロトタイプを開発した．そして，実機とシミュレーション（エリア：2km × 2km，ノード数：4000 台）の評価実験によって，250ms の遅延要求を満たすタイムリーな処理を実現すると共に，サービスの需要が増加した場合でも，近隣のエッジ IoT デバイスを活用したスケールアウトによって QoS を維持できること確認した．現状の IFoT プラットフォームは，地域 LAN に接続されたエッジ IoT デバイスへのオフローディングを対象としているが，今後は，5G を始めとした WAN に接続したエッジ IoT デバイスへの対応も進める予定である．また，スマートフォンや車などの移動体端末が持つ計算資源を状況に応じて活用するオフローディング手法の検討を進める．また，アプリケーションの性質に応じて，エッジ IoT，エッジクラウド，コアクラウドを柔軟に使い分けることできるデータ処理フレームワークの検討を進める予定である．そのためにも，サービスの遅延時間だけでなく，計算資源や通信資源を利用する際に発生する金銭的コスト，消費電力，サービスの提供者やサービスの利用者の嗜好をモデル化し，これらを多目的最適化問題として捉えながら，それぞれのトレードオフの分析を進める．また，実機の大規模テストベットを活用した長期間の実験を実施し，専門家と連携しながらセキュリティーやプライバシー，サステイナビリティの観点からも IFoT プラットフォームの改良を進める．また，ローカルの計算資源のシェアリングを加速するためのゲーミフィケーションメカニズムのデザインや，データ処理電力効率の観点から，CPU だけでなく GPU や FPGA など計算資源もモデル化し，より異種性のある IoT デバイス群から構成

されるリソースプールを有効活用したセンサデータ処理手法の検討を進める．加えて，ローカルの IoT デバイス群が自らの置かれた状況や近隣のリソースの負荷状況を自律分散的に把握しながら，状況に応じて実行するタスクの種類やタスクの計算負荷を調整する自己適応メカニズムについても検討を進める．

そして，本論文の大目標である「IoT センサデータ地産地消基盤」を実現するために，センシングの多様性を向上させる SenStick と，コンピューティングの柔軟性を向上させる IFoT をシームレスに統合するための GUI など，操作性を向上させるためのインターフェースを検討し，誰もが気軽に思いついた IoT システムを構築し，実環境で展開できるようなプラットフォームの構築を進める予定である．

謝辞

本研究を進めるにあたり，奈良先端科学技術大学院大学 安本 慶一 教授には，御多忙の中にも関わらず幾度も議論の時間を設けて頂き，研究全般に関し多大なるご指導・ご助言を賜りました．研究テーマの決定から論文の執筆にいたるまで，手厚いご指導を頂きましたことに，感謝の意を表すとともに，心より厚く御礼申し上げます．

奈良先端科学技術大学院大学 中島 康彦 教授には，ご多忙の中にも関わらず，論文審査員を引き受けていただくとと共に，貴重な御意見を賜りました．心より感謝申し上げます．

九州大学 荒川 豊 教授には，本研究を進めるにあたり，分野横断的な知見や様々な考え方をご指導・ご助言を賜りました．特に，SenStick や WaistonBelt の研究に関して，多大なるお力添えをいただきました．感謝の意を表すとともに，心より厚く御礼申し上げます．

大阪大学 山口弘純 准教授には，本研究を進めるにあたり，的確なご指導および鋭い御助言を頂きました．心より感謝申し上げます．

奈良先端科学技術大学院大学 諏訪 博彦 特任准教授には，社会学的な立場からのご指導・ご助言を頂くとともに，他分野の新たな知見を得るチャンスを頂きました．心より感謝申し上げます．

奈良先端科学技術大学院大学 藤本 まなと 助教には，学生に非常に近い立場から，研究者の先輩として進路や申請書の執筆に関して相談に乗っていただきました．心より感謝申し上げます．

大阪大学 水本 旭洋 特任助教には，学生に非常に近い立場から，研究の方針や進め方に関して相談に乗っていただきました．心より感謝申し上げます．

奈良先端科学技術大学院大学 松田 裕貴 助教には，同期入学ということもあり，互いに切磋琢磨しながら様々なプロジェクトを進めると共に，友人という立場から，公私にわたって相談に乗っていただきました．心より感謝申し上げます．

金岡恵事務補佐員，山内奈緒事務補佐員，尾川恵理事務補佐員には，学会や出張に関する事務処理を始めとし，様々な面で研究生活を支えていただきましたこと，謹んで感謝申し上げます．

また，私生活においても大変御世話になった河中 祥吾 氏をはじめとする本学ユビキタスコンピューティングシステム研究室の皆様に，感謝の意を表します．最後に，今日までの学生生活を様々な面から支えていただいた父母妹，そして，常に暖かく見守り支えていただいた， 津嶋路香氏に心から感謝の意を表します．

参考文献

[1] Mark Weiser. Ubiquitous computing. *Computer*, No. 10, pp. 71–72, IEEE, 1993.

[2] Mark Weiser. The computer for the 21st century. *IEEE pervasive computing*, Vol. 1, No. 1, pp. 19–25, IEEE, 2002.

[3] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, Vol. 29, No. 7, pp. 1645–1660, Elsevier, 2013.

[4] Dale Dougherty. The maker movement. *Innovations: Technology, Governance, Globalization*, Vol. 7, No. 3, pp. 11–14, MIT Press, 2012.

[5] Erica Rosenfeld Halverson and Kimberly Sheridan. The maker movement in education. *Harvard educational review*, Vol. 84, No. 4, pp. 495–504, Harvard Education Publishing Group, 2014.

[6] Paulo Blikstein. Digital fabrication and 'making' in education: The democratization of invention. *FabLabs: Of machines, makers and inventors*, Vol. 4, pp. 1–21, 2013.

[7] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, Vol. 3, No. 5, pp. 637–646, IEEE, 2016.

[8] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, Vol. 11, No. 11, pp. 1–16, 2015.

[9] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, Vol. 45, No. 5, pp. 37–42, ACM, 2015.

[10] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog

computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16. ACM, 2012.

[11] Ivan Stojmenovic and Sheng Wen. The fog computing paradigm: Scenarios and security issues. In *2014 Federated Conference on Computer Science and Information Systems*, pp. 1–8. IEEE, 2014.

[12] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog computing: A platform for internet of things and analytics. In *Big data and internet of things: A roadmap for smart environments*, pp. 169–186. Springer, 2014.

[13] Henry Chesbrough, Wim Vanhaverbeke, and Joel West. *Open innovation: Researching a new paradigm.* Oxford University Press on Demand, 2006.

[14] Everysense. `https://every-sense.com/`. Accessed: 2019-12.

[15] Oliver Brdiczka, Matthieu Langet, Jérôme Maisonnasse, and James L Crowley. Detecting human behavior models from multimodal observation in a smart home. *Automation Science and Engineering, IEEE Transactions on*, Vol. 6, No. 4, pp. 588–597, IEEE, 2009.

[16] Jianfeng Chen, Alvin Harvey Kam, Jianmin Zhang, Ning Liu, and Louis Shue. Bathroom activity monitoring based on sound. In *Pervasive Computing*, pp. 47–61. Springer, 2005.

[17] Gerald Bauer, Karl Stockinger, and Paul Lukowicz. Recognizing the use-mode of kitchen appliances from their current consumption. In *EuroSSC*, pp. 163–176. Springer, 2009.

[18] Kazushige Ouchi and Miwako Doi. Indoor-outdoor activity recognition by a smartphone. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 600–601. ACM, 2012.

[19] Ashiq Anjum and Muhammad Usman Ilyas. Activity recognition using smartphone sensors. In *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pp. 914–919. IEEE, 2013.

[20] Oscar D Lara and Miguel A Labrador. A survey on human activity recogni-

tion using wearable sensors. *Communications Surveys & Tutorials, IEEE*, Vol. 15, No. 3, pp. 1192–1209, IEEE, 2013.

[21] Jins meme. `https://jins-meme.com/`. Accessed: 2019-12.

[22] Yuki Matsuda and et al. Waistonbelt: A belt for monitoring your real abdominal circumference forever. In *UbiComp/ISWC'15, Demo*.

[23] Kurahashi Masaya, Muaro Kazuya, Terada Tsutomu, and Tsukamoto Masahiko. User identification based on rotation of toilet paper. In *IPSJ DICOMO 2015 (In Japanese)*, Vol. 2015, pp. 1217–1225. IPSJ, 2015.

[24] Hapifork. `https://www.hapi.com/`. Accessed: 2019-12.

[25] Sensortag cc2541 (texas instruments). `http://www.ti.com/tool/cc2541dk-sensor`. Accessed: 2019-12.

[26] Iot smart module. `http://www.alps.com/j/iotsmart/`. Accessed: 2019-12.

[27] Tsnd121 (atr-promotions). `http://www.atr-p.com/products/TSND121.html`. Accessed: 2019-12.

[28] Ax6 (axivity). `http://axivity.com/product/AX6`. Accessed: 2019-12.

[29] Ian H Witten, Eibe Frank, Leonard E Trigg, Mark A Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations, 1999.

[30] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, Vol. 12, No. Oct, pp. 2825–2830, 2011.

[31] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

[32] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram

Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, Vol. 17, No. 1, pp. 1235–1241, JMLR. org, 2016.

[33] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, 2013.

[34] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, Vol. 171, pp. 754–767, Elsevier, 2016. doi: 10.1016/j.neucom.2015.07.085.

[35] Timo Sztyler, Heiner Stuckenschmidt, and Wolfgang Petrich. Position-aware activity recognition with wearable devices. *Pervasive and mobile computing*, Vol. 38, pp. 281–295, Elsevier, 2017. doi: 10.1016/j.pmcj.2017.01.008.

[36] Karl-Heinz Wagner and Helmut Brath. A global view on the development of non communicable diseases. *Preventive medicine*, Vol. 54, pp. S38–S41, Elsevier, 2012.

[37] Mukesh Sharma and PK Majumdar. Occupational lifestyle diseases: An emerging issue. *Indian journal of occupational and environmental medicine*, Vol. 13, No. 3, p. 109, Wolters Kluwer–Medknow Publications, 2009.

[38] Richard Dobbs and James Manyika. The obesity crisis. http://www.mckinsey.com/mgi/overview/in-the-news/the-obesity-crisis, 2015.

[39] Yuki Matsuda, Takashi Hasegawa, Keiichiro Iwanami, Naoya Saito, Takuya Ishioka, Ismail Arai, Yutaka Arakawa, and Keiichi Yasumoto. Waistonbelt: A belt for monitoring your real abdominal circumference forever. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15 Adjunct)*, pp. 329–332, 2015. doi: 10.1145/2800835.2800869.

[40] Yuki Matsuda, Takashi Hasegawa, Ismail Arai, Yutaka Arakawa, and Keiichi Yasumoto. Waistonbelt 2: a belt-type wearable device for monitoring abdominal circumference, posture and activity. In *The 9th International Conference on Mobile Computing and Ubiquitous Networking (ICMU '16)*, pp. 1–6, 2016. doi: 10.1109/ICMU.2016.7742089.

[41] Dean M Karantonis, Michael R Narayanan, Merryn Mathie, Nigel H Lovell, and Branko G Celler. Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE transactions on information technology in biomedicine*, Vol. 10, No. 1, pp. 156–167, IEEE, 2006.

[42] Uwe Maurer, Asim Smailagic, Daniel P Siewiorek, and Michael Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pp. 4–pp. IEEE, 2006.

[43] Linda R Elliott, Jan van Erp, Elizabeth S Redden, and Maaike Duistermaat. Field-based validation of a tactile navigation device. *IEEE transactions on haptics*, Vol. 3, No. 2, pp. 78–87, IEEE, 2010.

[44] Moriya Kazuki, Fujimoto Manato, Arakawa Yutaka, and Yasumoto Keiichi. Indoor localization based on distance-illuminance model and active control of lighting devices. In *Indoor Positioning and Indoor Navigation (IPIN), 2016 IEEE*, pp. 1–6. IEEE, 2016.

[45] D Evans. The internet of things: How the next evolution of the internet is changing everything. http://blogs.cisco.com/news/the-internet-of-things-infographic/. Accessed: 2017-06.

[46] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, Vol. 29, No. 7, pp. 1645–1660, Elsevier, 2013.

[47] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the*

First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12, pp. 13–16, ACM, 2012. doi: 10.1145/2342509.2342513.

[48] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, Vol. 45, No. 5, pp. 37–42, ACM, 2015.

[49] Keiichi Yasumoto, Hirozumi Yamaguchi, and Hiroshi Shigeno. Survey of real-time processing technologies of iot data streams. *Journal of Information Processing*, Vol. 24, No. 2, pp. 195–202, 2016. doi: 10.2197/ipsjjip.24.195.

[50] Stella CS Porto and Celso C Ribeiro. A tabu search approach to task scheduling on heterogeneous processors under precedence constraints. *International Journal of high speed computing*, Vol. 7, No. 01, pp. 45–71, World Scientific, 1995.

[51] Tracy D Braun, Howard Jay Siegel, Noah Beck, Ladislau L Bölöni, Muthucumaru Maheswaran, Albert I Reuther, James P Robertson, Mitchell D Theys, Bin Yao, Debra Hensgen, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed computing*, Vol. 61, No. 6, pp. 810–837, Elsevier, 2001.

[52] Andrew J Page and Thomas J Naughton. Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pp. 8–pp. IEEE, 2005.

[53] Chien-Chung Shen and Wen-Hsiang Tsai. A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion. *IEEE Transactions on Computers*, Vol. 100, No. 3, pp. 197–203, IEEE, 1985.

[54] Ayed Salman, Imtiaz Ahmad, and Sabah Al-Madani. Particle swarm opti-

mization for task assignment problem. *Microprocessors and Microsystems*, Vol. 26, No. 8, pp. 363–371, Elsevier, 2002.

[55] Qian Zhu and Gagan Agrawal. Resource provisioning with budget constraints for adaptive applications in cloud environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pp. 304–307. ACM, 2010.

[56] Ming Mao and Marty Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 49. ACM, 2011.

[57] Danilo Ardagna, Michele Ciavotta, and Mauro Passacantando. Generalized nash equilibria for the service provisioning problem in multi-cloud systems. *IEEE Transactions on Services Computing*, Vol. 10, No. 3, pp. 381–395, IEEE, 2017.

[58] Jinlai Xu, Balaji Palanisamy, Heiko Ludwig, and Qingyang Wang. Zenith: Utility-aware resource allocation for edge computing. In *Edge Computing (EDGE), 2017 IEEE International Conference on*, pp. 47–54. IEEE, 2017.

[59] Olena Skarlat, Stefan Schulte, Michael Borkowski, and Philipp Leitner. Resource provisioning for iot services in the fog. In *Service-Oriented Computing and Applications (SOCA), 2016 IEEE 9th International Conference on*, pp. 32–39. IEEE, 2016.

[60] Olena Skarlat, Matteo Nardelli, Stefan Schulte, and Schahram Dustdar. Towards qos-aware fog service placement. In *Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on*, pp. 89–96. IEEE, 2017.

[61] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, Vol. 38, pp. 63–74. ACM, 2008.

[62] Claudio Cicconetti, Marco Conti, and Andrea Passarella. Low-latency distributed computation offloading for pervasive environments. In *Perva-*

sive Computing and Communications (PerCom), 2019 IEEE International Conference on. IEEE, 2019.

[63] Imad Abdeljaouad and Ahmed Karmouch. Monitoring iptv quality of experience in overlay networks using utility functions. *Journal of Network and Computer Applications*, Vol. 54, pp. 1–10, Elsevier, 2015.

[64] Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research*, Vol. 112, No. 1, pp. 3–41, Elsevier, 1999.

[65] Pieter Hintjens. *ZeroMQ: messaging for many applications.* " O'Reilly Media, Inc.", 2013.

[66] Yugo Nakamura, Hirohiko Suwa, Yutaka Arakawa, Hirozumi Yamaguchi, and Keiichi Yasumoto Yasumoto. Design and implementation of middleware for iot devices toward real-time flow processing. In *Distributed Computing Systems Workshops (ICDCSW), 2016 IEEE 36th International Conference on*, pp. 162–167. IEEE, 2016.

[67] Yugo Nakamura, Hirohiko Suwa, Yutaka Arakawa, Hirozumi Yamaguchi, and Keiichi Yasumoto Yasumoto. Middleware for proximity distributed real-time processing of iot data flows. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pp. 771–772. IEEE, 2016.

業績リスト

論文誌

主著

1. Yugo Nakamura, Yuki Matsuda, Yutaka Arakawa, and Keiichi Yasumoto, "WaistonBelt X: A Belt-Type Wearable Device with Sensing and Intervention Toward Health Behavior Change," Sensors 2019, Vol.19, No.20, Article ID 4600, 21 pages, doi:10.3390/s19204600. (IF = 3.031) (2 章に対応)
2. Yugo Nakamura, Yutaka Arakawa, Takuya Kanehira, Masashi Fujiwara, and Keiichi Yasumoto : SenStick: Comprehensive Sensing Platform with an Ultra Tiny All-In-One Sensor Board for IoT Research, Journal of Sensors, vol. 2017, Article ID 6308302, 16 pages, 2017. doi:10.1155/2017/6308302. (IF = 2.057) (2 章に対応)

共著

1. 梅木寿人，中村優吾，藤本まなと，水本旭洋，諏訪博彦，荒川豊，安本慶一：混雑度の偏りを考慮した避難所決定手法，情報処理学会論文誌，vol. 60，no. 2，pp. 608-616，2019 年 2 月.
2. Yohei Torigoe, Yugo Nakamura, Manato Fujimoto, Yutaka Arakawa, and Keiichi Yasumoto : Strike Activity Detection and Recognition Using Inertial Measurement Unit Towards Kendo Skill Improvement Support System, Sensors and Materials, vol. 32, no. 2, pp. 651-673, Feb. 2020. (IF = 0.519)

国際会議

主著

1. Yugo Nakamura, Teruhiro Mizumoto, Hirohiko Suwa, Yutaka Arakawa, Hirozumi Yamaguchi, Keiichi Yasumoto: In-Situ Resource Provisioning with Adaptive Scale-out for Regional IoT Services, The Third ACM/IEEE Symposium on Edge Computing (SEC 2018), pp. 203-213, Bellevue, USA, October 25-27, 2018. (Acceptance rate = 38
2. Yugo Nakamura, Teruhiro Mizumoto, Hirohiko Suwa, Yutaka Arakawa, Hirozumi Yamaguchi, and Keiichi Yasumoto: Design and Evaluation of In-situ Resource Provisioning Method for Regional IoT Services, IEEE/ACM International Symposium on Quality of Service 2018 (IwQoS 2018), Poster, Banff, Alberta, Canada 2018. (3 章に対応)
3. Yugo Nakamura, Yoshinori Umetsu, Jose Paolo Talusan, Keiichi Yasumoto, Wataru Sasaki, Masashi Takata, Yutaka Arakawa, Multi-stage activity inference for locomotion and transportation analytics of mobile users, Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, pp. 1579-1588, Singapore, Singapore, 2018.
4. Yugo Nakamura, Takuya Kanehira, Yutaka Arakawa, and Keiichi Yasumoto: SenStick 2: ultra tiny all-in-one sensor with wireless charging(DEMO), The ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2016), 2016. (2 章に対応)
5. Yugo Nakamura, Yutaka Arakawa, and Keiichi Yasumoto: Smart Experiment Platform for Wearable Computing, First Workshop on Eye Wear Computing collocated with ISWC/UbiComp (EYEWEAR 2016), 2016. (2 章に対応)
6. Yugo Nakamura, Yutaka Arakawa, and Keiichi Yasumoto: SenStick: A Rapid Prototyping Platform for Sensorizing Things, 2016 Ninth International Conference on Mobile Computing and Ubiquitous Networking

(ICMU 2016), 2016. (2 章に対応)

7. Y. Nakamura, H. Suwa, Y. Arakawa, H. Yamaguchi, K. Yasumoto: Middleware for Proximity Distributed Real-time Processing of IoT Data Flows, The 36th IEEE International Conference on Distributed Computing Systems (ICDCS 2016), Demo, 2016. (3 章に対応)
8. Y. Nakamura, H. Suwa, Y. Arakawa, H. Yamaguchi, K. Yasumoto: Design and Implementation of Middleware for IoT Devices toward Real-Time Flow Processing, The 1st Workshop on Edge Computing (WEC 2016), 2016. (3 章に対応)

共著

1. Hirohiko Suwa, Atsushi Otsubo, Yugo Nakamura, Masahito Noguchi: Data Collection and Index Creation for the Vacant Rental Property using IoT Sensing, 2019 IEEE 8th Global Conference on Consumer Electronics(GCCE 2019) OS-RET, Osaka, Japan, October 15-18, 2019.
2. Jose Talusan, Francis Tiausas, Sopicha Stirapongsasuti, Yugo Nakamura, Teruhiro Mizumoto, and Keiichi Yasumoto: Evaluating Performance of In-Situ Distributed Processing on IoT Devices by Developing a Workspace Context Recognition Service, Information Quality and Quality of Service for Pervasive Computing (IQ2S2019), p.633-638, Kyoto, Japan, March 2019.
3. Masashi Takata, Yugo Nakamura, Yohei Torigoe, Manato Fujimoto, Yutaka Arakawa, and Keiichi Yasumoto: Strikes-Thrusts Activity Recognition Using Wrist Sensor Towards Pervasive Kendo Support System, Workshop on Sensing Systems and Applications Using Wrist Smart Devices (WristSense2019), pp. 243-248, Kyoto, Japan, March 2019.
4. Yoshinori Umetsu, Yugo Nakamura, Yutaka Arakawa, Manato Fujimoto, and Hirohiko Suwa: EHAAS: Energy Harvesters As A Sensor for Place Recognition on Wearables, IEEE International Conference on Pervasive

Computing and Communications (PerCom2019), pp. 222-231, Kyoto, Japan, March 2019. (Acceptance rate = 19.8

5. Masashi Takata, Yugo Nakamura, Manato Fujimoto, Yutaka Arakawa, Keiichi Yasumoto: Investigating the Capitalize Effect of Sensor Position for Training Type Recognition in a Body Weight Training Support System, Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, pp. 1404-1408 Oct. 2019.
6. Yutaka Arakawa, Yugo Nakamura, Hirohiko Suwa, Yoshinori Umetsu, Manato Fujimoto, Keiichi Yasumoto: Feasibility Study Toward a Battery-free Place Recognition System Based on Solar Cells, Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, Demo, Oct. 2019.
7. Jose Paolo Talusan, Yugo Nakamura, Teruhiro Mizumoto, Keiichi Yasumoto: Near Cloud: Low-Cost Low-Power Cloud Implementation for Rural Area Connectivity and Data Processing, 1st IEEE International Workshop on Flow Oriented Approaches in Internet of Things and Cyber-Physical Systems (InfoFlow 2018), Tokyo, Japan, July, 2018.
8. Kazuhito Umeki, Yugo Nakamura, Manato Fujimoto, Yutaka Arakawa, Keiichi Yasumoto: Real-Time Congestion Estimation in Sightseeing Spots with BLE Devices, IEEE International Conference on Pervasive Computing and Communications (PerCom 2018), Demo, Mar 2018.
9. Masato Hidaka, Yuki Matsuda, Shogo Kawanaka, Yugo Nakamura, Manato Fujimoto, Yutaka Arakawa, and Keiichi Yasumoto: A System for Collecting and Curating Sightseeing Information toward Satisfactory Tour Plan Creation, The Second International Workshop on Smart Sensing Systems (IWSSS' 17), Aug. 2017.

国内会議

主著

1. 中村 優吾, 荒川 豊, 安本 慶一：ウェアラブルセンサ装着位置/向きの違いにロバストな行動認識システムの実現に向けたデータ変換手法の検討, マルチメディア、分散、協調とモバイル (DICOMO2019) シンポジウム，福島県，2019 年 7 月.
2. 中村 優吾, 水本 旭洋, 諏訪 博彦, 荒川 豊, 山口 弘純, 安本 慶一：地域 IoT サービスに対する計算需要に応じた適応型地産地処リソース配分手法の提案, マルチメディア、分散、協調とモバイル (DICOMO2018) シンポジウム，福井県，2018 年 7 月．(3 章に対応)
3. 中村優吾，水本旭洋，諏訪博彦，荒川豊，山口弘純 (大阪大)，安本慶一 : IoT データ流の実時間処理を実現する IoT デバイス向け分散処理ミドルウェアの設計と評価, 第 82 回モバイルコンピューティングとパーペイシブシステム研究会，東京大学 本郷キャンパス，東京都，2017 年 3 月．(3 章に対応)
4. 中村優吾，諏訪博彦，荒川豊（奈良先端大)，山口弘純（大阪大），安本慶一（奈良先端大) : 観光案内向け CGM キュレーションのためのローカル IoT プラットフォームの提案, マルチメディア，分散，協調とモバイル（DICOMO 2016）シンポジウム，三重県，2016 年 7 月．(3 章に対応)
5. 中村優吾（奈良先端大)，Tony Shi（Unitec)，諏訪博彦，荒川豊（奈良先端大)，山口弘純（大阪大)，安本慶一（奈良先端大)：ローカル環境での効果的な動画像解析を実現する分散処理システムの提案, 情報処理学会 第 79 回モバイルコンピューティングとパーベイシブシステム（MBL）研究会，沖縄産業支援センター，沖縄県，2016 年 5 月.
6. 中村優吾，森下滋也，前中省吾，安本慶一（奈良先端大)，福倉寿信，佐藤啓太，清原博文（デンソー)：価値共創キュレーションシステムの実現向けて－ ICT による新しい価値創造に向けた取り組み－, サービス学会 第 4 回 国内大会学会，2016 年 3 月 28-29 日.
7. 中村優吾，前中省吾，森下滋也，安本慶一（奈良先端大)，福倉寿信，佐藤啓太，清原博文（デンソー)：価値共創キュレーションシステムの構想 ～会員

制タクシーの会話支援を実例とした概念設計〜, 情報処理学会 第 78 回全国大会，2016 年 3 月 10-12 日.

8. 中村 優吾, 諏訪 博彦, 荒川 豊, 山口 弘純, 安本 慶一：多様な IoT データストリームをクラウドレスで分散処理するミドルウェアの設計, 第 77 回モバイルコンピューティングとパーベイシブシステム研究会, 8 pages, pp.1–8, 愛知県豊田市, 2015 年 12 月 4 日. (3 章に対応)
9. 中村 優吾, 松田 裕貴, 荒川 周造, 金平 卓也, 安本 慶一：GAIFoT：情報流を活用した地域分散型アンビエントインターフェース, ポスター発表, 第 23 回マルチメディア通信と分散処理ワークショップ（DPSWS 2015）論文集, 6 pages, pp.264–269, 長崎県雲仙市, 2015 年 10 月 14 日.
10. 中村 優吾, 松田 裕貴, 荒川 周造, 金平 卓也, 安本 慶一：地域情報流の高次利用を促す地域分散アンビエントインタフェース (GAIFoT) の提案, ポスター発表, 電子情報通信学会技術研究報告, コンピュータシステム研究会 (CPSY), 千葉県幕張市, 2015 年 10 月 8 日.

共著

1. 諏訪 博彦, 大坪 淳, 中村 優吾, 野口 真史：新たな賃貸物件探索指標構築のためのセンシングシステム, マルチメディア、分散、協調とモバイル (DICOMO2019) シンポジウム，福島県，2019 年 7 月.
2. 諏訪博彦，大坪敦，中村優吾，野口真史：IoT センシングによる賃貸物件快適度推定のためのデータ収集，人工知能学会全国大会，朱鷺メッセ 新潟コンベンションセンター，新潟県，2019 年 6 月.
3. 鈴木優，藤本まなと，吉野幸一郎，日高真人，Nguyen Quynh Mai，永野一馬，中村優吾，大坪敦，田中翔平，安本慶一，中村哲：京都インバウンド観光における IoT2H 観光情報アプリケーションの構築，第 11 回データ工学と情報マネジメントに関するフォーラム（DEIM2019），DEIM Forum 2019 I3-4，pp. 1-9，ホテオークラ JR ハウステンボス，長崎県，2019 年 3 月.
4. 鳥越庸平, 高田将志, 中村優吾, 藤本まなと, 荒川豊, 安本慶一：剣道上達支援のための IMU を用いた打突動作認識，研究報告ユビキタスコンピューティ

ングシステム (UBI), 2019-UBI-61, pp. 1-7, 東京大学駒場 II キャンパス, 東京都, 2019 年 3 月.

5. Stirapongsasuti Sopicha, Nakamura Yugo, Suwa Hirohiko, Yutaka Arakawa, Keiichi Yasumoto: Feasibility Study on Distributed Sensor Processing in BLE Mesh Network，第 89 回モバイルコンピューティングとパーベイシブシステム研究会 (MBL2018)，宮崎県，2018 年 11 月.
6. 諏訪博彦，大坪敦，中村優吾，野口真史: 新たな賃貸物件探索指標のための IoT センシングデバイスの検討，グループウェアとネットワークサービスワークショップ 2018，長野県，2018 年 11 月.
7. 梅津 吉雅，中村 優吾，荒川 豊， 藤本 まなと， 安本 慶一 : EHAAS : 環境発電素子の発電量に基づくウェアラブル場所推定システム, マルチメディア、分散、協調とモバイル (DICOMO2018) シンポジウム，福井県，2018 年 7 月.
8. 高田将志, 中村優吾, 藤本 まなと, 荒川 豊, 安本慶一 : メニュー推薦に向けたセンサ取り付け位置に依存しない自重トレーニング種目認識手法の提案, マルチメディア、分散、協調とモバイル (DICOMO2018) シンポジウム，福井県，2018 年 7 月.
9. 諏訪 博彦, 中村 優吾, 野口 真史 : IoT センシングによる新たな賃貸物件探索指標の検討, マルチメディア、分散、協調とモバイル (DICOMO2018) シンポジウム，福井県，2018 年 7 月.
10. 梅津 吉雅， 中村 優吾， 荒川 豊， 藤本 まなと， 諏訪 博彦， 安本 慶一 : 環境発電素子の発電量に基づく行動認識手法の提案, 第 87 回モバイルコンピューティングとパーベイシブシステム（MBL)，2018-MBL-87，イーフ情報プラザ，沖縄県，2018 年 5 月.
11. 梅津 吉雅, 藤原 聖司, 中村 優吾, 藤本 まなと, 荒川 豊, 安本 慶一: 環境発電素子の発電量に基づく屋内行動認識システムの検討, 2018 年電子情報通信学会 (IEICE) 総合大会 ISS 特別企画「学生ポスターセッション, 東京, 2018 年 3 月.
12. 高田将志, 中村 優吾, 藤本 まなと, 荒川 豊, 安本 慶一: ウェアラブルデバイスを用いた体幹トレーニング種目認識手法, 2018 年電子情報通信学会 (IEICE)

総合大会 ISS 特別企画「学生ポスターセッション, 東京, 2018 年 3 月.

13. 高田将志, 中村優吾, 藤本まなと, 荒川豊, 安本慶一: 体幹トレーニング支援に向けたウェアラブルデバイスによる種目認識手法の提案 第 177 回ヒューマンコンピュータインタラクション研究, 東京, 2018 年 3 月.
14. 梅木寿人, 中村優吾，藤本まなと，水本旭洋，諏訪博彦，荒川豊，安本慶一: 災害時の混雑情報を考 慮した避難所決定手法の提案, 第 86 回モバイルコンピューティングとパーベイシブシステム研究会 (MBL2018), 東京, 2018 年 2 月.
15. 梅木寿人，中村優吾，水本旭洋，藤本まなと，諏訪博彦，荒川豊，安本慶一：混雑情報を考慮した災害発生時の避難場所決定手法に関する一検討, 第 25 回 マルチメディア通信と分散処理ワークショップ (DPSWS2017)，北海道，2017 年 10 月.
16. 金谷勇輝, 中村優吾, 諏訪博彦, 荒川豊, 安本慶一：観光動画キュレーションの実現に向けたハイライト抽出手法の検討, 2017 年度 情報処理学会関西支部 支部大会，大阪大学中之島センター，大阪府，2017 年 9 月.
17. 日高真人，松田裕貴，河中祥吾，中村優吾，藤本まなと，荒川豊，安本慶一：実時間観光コンテンツ提供に向けた観光情報収集・キュレーションシステムの提案，第 68 回高度交通システムとスマートコミュニティ研究発表会，公立はこだて未来大学，北海道，2017 年 3 月.
18. 松田裕貴，Akpa Akpro Elder Hippocrate，Konan N’djabli Cedric Ange，中村優吾，前田直樹，千住琴音，荒川豊：位置情報サービスにおける個人特化型ゲーミフィケーション ～スタンプラリーイベントを通じた「慣れ」「飽き」の調査～, 社会システムと情報技術研究ウィーク in ルスツリゾート，ルスツリゾートホテル，北海道，2017 年 3 月.
19. 日高真人，中村優吾，荒川豊，安本慶一：実時間観光プランニングシステムのための CGM のキュレーションの課題検討, 2016 年度 情報処理学会関西支部 支部大会，大阪大学中之島センター，大阪府，2016 年 9 月.
20. 梅木寿人，中村優吾，水本旭洋，藤本まなと，荒川豊，安本慶一：実時間雨量情報に基づく動的ハザードマップの検討, 2016 年度 情報処理学会関西支部 支部大会，大阪大学中之島センター，大阪府，2016 年 9 月.

21. 宵憲治， 荒川周造， 中村優吾， 前田直樹， 松本誠義， 平部裕子， 安本慶一, リモコン操作ログと照明演出装置を用いた室内快適化補助システム, 研究報告コンピュータセキュリティ (CSEC), pp. 1-8 , 2016 年 2 月.