

愛知工業大学情報科学部情報科学科
コンピュータシステム専攻

令和4年度 卒業論文

独立したコミュニティにおける
滞在ウォッチの安定運用のための
システム拡張に関する研究

2023年2月

研究者 K19036 亀田優作
K19074 外山瑠起

指導教員 梶 克彦 准教授

目 次

第 1 章 はじめに	3
1.1 背景	3
1.2 滞在ウォッチの安定運用のためのシステム拡張	4
1.3 論文構成	4
第 2 章 関連研究	5
2.1 屋内位置推定に関する研究	5
2.2 在室者検出及び在室管理システムに関する研究	6
2.3 コミュニケーション促進に関する研究	6
第 3 章 在室管理プラットフォーム	
「滞在ウォッチ」	7
3.1 「滞在ウォッチ」のシステム構成	7
3.2 在室者情報を用いた応用システムについて	12
第 4 章 独立したコミュニティにおける滞在ウォッチの安定運用のためのシステム拡張	15
4.1 安定運用に向けた既存システムの再構築	15
4.1.1 サーバ側システムの再設計と再構築	15
4.1.2 クライアントサイドの実装	17
4.2 利用者の管理とアクセス制御システムの整備	18
4.3 スマートフォンアプリによるビーコンと実デバイス	22
4.3.1 BLE ビーコンのアプリケーション化	22
4.3.2 スマートフォンビーコンと実デバイスによるビーコンのハイブリッド活用	23
第 5 章 おわりに	26
5.1 まとめ	26
5.2 今後の課題	26
謝辞	28

第1章 はじめに

本章では、研究背景や我々の先行研究について述べ、本研究の目的と概要を説明する。

1.1 背景

近年社会的に重要な課題の1つとして、在室者管理が挙げられる。在室者管理は環境保護やエネルギー効率化、セキュリティ強化などの目的から、家庭やオフィスなどの建物内での生活やビジネスに関する様々なアプリケーションに応用される。例えばエネルギー管理は居住者がいる場合といない場合でエネルギー消費量が大きく異なるため、在室者管理によって建物内でのエネルギー消費を最適化できる。またセキュリティや照明などのシステムの自動化にも応用され、これにより安全管理やエネルギー効率化を図れる。在室者管理は災害時や緊急事態においても重要な役割を持つ。災害時には避難した居住者の確認が重要であり、在室者管理を使用すれば確認作業をスムーズに行える。

また在室者管理はコミュニティにおいても様々なメリットがある。その1つにコミュニティ内でのコミュニケーション促進が挙げられる。在室者管理を行い在室者情報を可視化すれば、コミュニティ内で誰がいるのかを確認できる。これによりコミュニティ内での交流や活動がスムーズになり、同じコミュニティに属する人たちが同じ時間にいる場合には共同での過ごし方を提案できる。また在室者が多い時間帯、少ない時間帯の傾向が把握できればコミュニティ内でのイベントやミーティングのタイミングを調整しやすくなる。これにより参加者が参加しやすい時間帯にイベントやミーティングを提案できるため参加者が集まりやすくなり、コミュニケーションの促進を行える。

さらに近年在室者管理は新型コロナウイルスの影響により、感染拡大を防ぐ上で有効な手段と考えられる。在室者管理を使用すれば、感染者が訪れた場所や接触した人々を特定し消毒や清掃を行える。また濃厚接触者を特定し隔離や検査を行えば早期発見や早期対応が可能になる。そのためコミュニティなどで在室システムを導入すれば感染リスクを低減するために活用できると考えられる。

在室者管理の研究は、学術界や産業界においても注目を集めており、20世紀後半から様々な方法が提案してきた。在室者管理を行う方法としてICカードを用いて在室者情報を能動的に記録する方法がある。例として学生が教室に入るときにICカードをタッチし、出るときにも同様にタッチを行うと学生が教室に在室した時間を記録できるシステムが挙げられる。ICカードの使用により、手動での在室情報の管理から自動化が可能になる。この方法では在室者情報を確実に記録できるがICカードを用いて能動的に記録する必要がある。利用者への負担がICカードの記録忘れを引き起こし在室情報の不正確さを招く可能性がある。この問題の解決策として、様々なセンサを使用した方法が提案されている。例えば、照度センサ、温度センサ、音声センサ、カメラなどが挙げられる。これらセンサを使用して得られたデータを処理し在室者がいるかどうかを判定する。ただしこれらのセンサを使用した在室者管理は、環境条件や居住者の生活スタイルなどにより精度が異なるため、正確な在室者管理ができない場合がある。このように在室者管理は重要な課題であり様々なアプリケーションに応用されるが、環境によって正確な検出が難しいという問題もある。また深層学習を使用した在室者管理システムがある。深層学習は大量の画像や音声などのデータを特徴量として抽出し、学習を行うため高精度の人物の在室検出が可能である。しかし深層学習を使用した在室者管理には、大量のデータが必要でありデータ収集や学習には多くのリソースが必要なのが問題点である。またプライバシ保護の観点から、カメラの使用は避けられる場合が多い。

我々の先行研究としてBLE(Bluetooth Low Energy)ビーコン(以下、ビーコン)を用いた在室者管理プラットフォーム「滞在ウォッチ」を提案している。ビーコンの価格は低コストであり普及しやすいと考えられる。

えられる。これにより建物全体に導入が容易であり在室者検出のカバー範囲を広げられる。またビーコンは低消費電力であり長期間にわたって使用できる。これにより、在室者検出を24時間行えるため建物内の生活やビジネスに対して常時モニタリングを行える。さらにビーコンはビーコン単体だけでは個人を特定できず、例え紛失したとしても個人を特定できないためプライバシ保護にも適している。滞在ウォッチはビーコンを持ち歩き在室情報を記録するメンバ、現在の状況や履歴を閲覧したりAPIを通して在室情報を利用する利用者（メンバや管理者も利用者になりうる）、メンバ管理、メンバへのビーコン配布、利用者の登録を行う管理者、システムを開発する開発者が存在する。メンバの負担軽減のため、在室者情報をビーコンで受動的に記録する方法を採用している。ビーコンを持ったメンバが部屋に訪れると受信機が検知し、サーバに在室者情報を送信し、データベースに記録する。データベースに保存された情報は、独自に作成したWeb APIによって外部からの利用が可能である。先行研究として、Web APIを使用した退勤管理システムや在室情報可視化システム、部屋利用者の来訪促進システム、コミュニケーションプラットフォームなどがある。

1.2 滞在ウォッチの安定運用のためのシステム拡張

本研究の目的は滞在ウォッチによる複数コミュニティ間連携の実現である。複数コミュニティ間とは空間的な距離が近いコミュニティ間と定義する。例として大学の研究室間やビルのオフィス間などが挙げられる。滞在ウォッチにおける複数コミュニティ間連携によって得られるメリットとして、まずコミュニティ間でのコミュニケーションの増加が挙げられる。普段コミュニケーションを取らない別コミュニティの人々とのコミュニケーションが促進される。他のコミュニティからの新しい知見やアイデアの共有による知識量や創造性の向上、スマートなコミュニケーションやリソースの共有による生産性の向上が期待できる。またコミュニティ間の連携が容易になるため、新しいプロジェクトやイベントが生まれる可能性がある。

しかし滞在ウォッチを複数コミュニティ間で連携するには大きく分けて2つの問題点が存在する。まずコミュニティで独立した運用ができていない点が挙げられる。「独立した運用」とは、各コミュニティで自らのシステムや装置を運用できる状態を指す。複数コミュニティ間で連携するためには、各コミュニティで滞在ウォッチを運用する必要がある。これを行うには各コミュニティが設備やシステムを持つ必要がある。それぞれのコミュニティが独立した設備やシステムを持っていれば、各コミュニティが独自に在室情報を収集し管理できるためそれらを連携させられる。しかし既存の滞在ウォッチは單一コミュニティを前提として設計されているため各コミュニティが独立した運用を行えない。次に在室情報を長期に渡り継続的に記録できない点が挙げられる。在室情報の長期に渡る継続的な記録は、コミュニティ間でのコミュニケーションを促進するための基礎となる情報であるため重要である。しかし、既存の滞在ウォッチはBLEビーコンのみを使用して運用しているためバッテリがなくなると在室情報を記録できない。バッテリの交換を促しているが交換がされず放置される場合がある。

これらの問題を解決した滞在ウォッチの運用を安定運用と定義し本研究では滞在ウォッチを複数コミュニティ間で連携するために、独立したコミュニティにおける滞在ウォッチの安定運用のためのシステム拡張について提案する。安定運用に向けたアプローチとして既存システムの再構築、独立した運用に向けて利用者の管理とアクセス制御システムの整備、長期に渡る継続的な在室情報の記録に向けてBLEビーコンのアプリケーション化とアプリケーション化を行ったスマホビーコンと実デバイスビーコンによるハイブリッドシステムの構築を行う。

1.3 論文構成

第2章では、本研究と関連した研究との違いを比較する。第3章では、在室管理プラットフォーム「滞在ウォッチ」について述べる。第4章では、独立したコミュニティにおける滞在ウォッチの安定運用のためのシステム拡張について述べる。第5章では本研究に対するまとめと今後の課題について述べる。

第2章 関連研究

本研究と本研究に関連する研究との違いを比較する 2.1 節では屋内位置推定に関する関連研究について述べる。2.2 節では、部屋におけるスマートフォンやビーコン、IC カードを用いた在室者の検出方法及び在室管理システムに関する関連研究について述べる。2.3 節ではコミュニケーション促進の関連研究について述べる。

2.1 屋内位置推定に関する研究

Wi-Fi を用いた屋内位置推定に関する研究がある [?][?]. Wi-Fi 測位は Wi-Fi アクセスポイントの信号強度を利用して位置推定を行う手法である。この手法では無線 LAN のアクセスポイントから発信される電波強度を測定、アクセスポイントとの距離を推定し測位を行う。Wi-Fi 測位のメリットにコストが低い点がある。Wi-Fi 技術は多くの所で使用されており、多くの建物には Wi-Fi アクセスポイントが設置されている。そのため追加の設備費用が不要である。さらに Wi-Fi 測位の主な測位対象であるスマートフォンやタブレットなどの携帯端末には、基本的に Wi-Fi チップが搭載されているため導入が容易である。欠点はアクセスポイントの設置数によって測位精度が大きく左右される点である。

カメラを使用した屋内位置推定に関する研究がある [?][?]. カメラを使用し環境中に存在する特徴的なランドマークや形状を識別し、それらを用いて位置を推定する。具体的には画像認識や 3D 情報を使った手法がある。画像認識を使用した手法はカメラが撮影した画像から特徴的な目印となるものを認識し、それらを元に位置を推定する。特徴的な目印となるものには QR コード、バーコード、AR マーカーなどが挙げられる。3D 情報を使用した手法は立体的な情報を取得するステレオカメラや RGB-D カメラを用いて、3D 空間内の距離や障害物などから位置を推定する。この手法では自己の位置を推定する SLAM アルゴリズムや、点群データから環境の詳細な 3D 情報を抽出する点群処理技術が使用されている。これらの手法は GPS や Wi-Fi 測位などの手法と比べて、屋内でも高精度の測位が可能である。ただしカメラを使用するため人物や物体などのプライバシーに敏感な情報が含まれる可能性があり、それらに考慮する必要がある。加えて光の強弱や反射などの影響も受けやすいため、それらが測位精度に影響を与える可能性がある。

磁気指紋法を使用した屋内位置推定に関する研究がある [?][?]. これは磁気センサーを搭載したスマートフォンなどのデバイスを使用して、磁気指紋を測定しそれをもとに屋内の位置を推定する手法である。磁気指紋とは地球の南極から北極に向かって均一に流れている磁場が観測点の周囲の構造に依存して変化するパターンである。各地点の磁気の大きさをあらかじめ測定しておき、その情報をもとに磁気指紋データベースを作成する。磁気センサで取得した磁気の大きさの情報を磁気指紋データベースに照会してその位置を推定する。ただし地球の磁場は時間とともに変化しており、磁気指紋のデータベースが古くなっている場合、現在の磁場と異なるため正確な位置推定ができない可能性がある。さらに磁気指紋は建物や鉄骨などの金属構造物の影響を受けるため、それらが存在する場所では正確な位置推定ができない可能性がある。

本研究では狭い範囲でより正確な屋内位置推定が可能である、ビーコンと部屋に設置した受信機を使用した手法を採用する。この手法ならば受信機の設置やビーコンの設定などのコストがかかるが Wi-Fi 測位や磁気指紋方を使用した手法と比べて、環境に大きく左右されず正確な屋内位置推定が可能である。またカメラを用いた方法と比較してビーコンを用いた屋内位置推定は、ビーコン単体で個人を特定できないためプライバシーに対するリスクが低い。

2.2 在室者検出及び在室管理システムに関する研究

在室者を検出する手法として IC カードを用いた検出方法 [?] [?] [?] [?] [?] [?] や、ビーコンの受信電波強度を利用した検出方法 [?][?] がある。IC カードを用いた在室者検出方法はメンバに IC カードを携帯してもらい、専用の機器などを用いて在室者を検出する手法である。必要となる機器を導入した後はメンバは IC カードを用いるだけで在室者検出ができる。しかし IC カードを用いた方法では、メンバが

ビーコンを用いた在室者検出方法には二つある。1つ目はメンバにビーコンを携帯してもらい、在室者を検出したい部屋に受信機を置く方法がある。部屋を利用するメンバの在室者検出はビーコンの受信電波を受信機が取得するだけで行えるので、自動で行える。また、ビーコンはサイズが小さいものが多く、メンバが携帯する負担もからない。在室者検出する部屋に受信機を置き、メンバはビーコンを携帯するだけなのでコストも抑えられる。2つ目にメンバにはスマホを携帯してもらい、在室者を検出したい部屋にビーコンを置く方法がある。スマホの普及が進んでいる[?]ので、導入コストはビーコンのみであるため、1つの方法よりもコストを抑えられる。しかし、全員がスマホを所持しているわけではないので、スマホを所持していない利用者には別の検出方法を導入する必要がある。

在室者を検出し，在室者情報を管理するシステムに関する研究がある[?][?][?]. 大学や会社では在室者管理システムを用いて，講義の出欠[?][?][?][?][?] や勤怠管理[?] が行われている. スマホを用いて在室者を検出し，在室者情報を管理するシステムは，スマホを所持している人が多いため，新たに必要となる機器の数が少なく，コストを抑えられる[?]. またビーコンを用いて在室者を検出し，在室者情報を管理するシステムはメンバにビーコンを携帯してもらい，在室者を検出したい部屋に受信機を設置すれば在室者を検出できるので，コストを抑えられる[?][?]. 本研究では先ほど述べたようにスマートフォンを所持していないメンバも想定し，メンバ一人一人にビーコンを携帯してもらい，在室者を検出したい部屋に受信機を設置するシステムとした.

2.3 コミュニケーション促進に関する研究

第三者間のコミュニケーションや知っている人同士のコミュニケーションを支援、促進する研究がある。まず見知らぬ他人や顔だけは知っているがコミュニケーションを取らない人とのコミュニケーションを支援促進するものがある[?][?][?][?]. これらは気軽に他者とのコミュニケーションができると示唆している。しかしそうした人同士のコミュニケーションを促進するものではなく、あくまで見知らぬ他者やコミュニケーションを取らない人とのコミュニケーションを支援、促進するものである。また公共空間でのコミュニケーションを支援、促進する研究がある[?][?][?][?]. 見知らぬ人や顔だけは知っている関係ではなく、同じ空間を共有している人同士のコミュニケーションを支援、促進するものである。コミュニケーションのきっかけや共有できる情報を提示するものとして、本研究でも同じ空間を共有している人同士のコミュニケーションを促進する方法として参考にする。

第3章 在室管理プラットフォーム 「滞在ウォッチ」

本章では、ビーコンを用いた在室者検出および在室管理プラットフォーム「滞在ウォッチ」について述べる。3.1節では、「滞在ウォッチ」のシステム概要、利用した機器やサーバ、ビーコンと個人の紐付け、在室者推定、在室者管理サーバを述べる。3.2節では、滞在ウォッチによって記録された在室者情報を用いた応用システムについて述べる。

3.1 「滞在ウォッチ」のシステム構成

滞在ウォッチは、ビーコンを持ち歩き在室情報を記録するメンバ、メンバ管理、メンバへのビーコン配布、利用者の登録を行う管理者、現在状況や履歴を閲覧したり API を通して在室情報を利用する利用者（メンバや管理者も利用者になりうる）、システムを開発する開発者がおり、メンバがビーコンを携帯し、部屋ごとに設置された受信機によりビーコンを検出する手法で、在室者管理を自動に行う。メンバには一人1つビーコンを携帯してもらう。サーバには部屋メンバの名前とビーコンのIDを登録する。ビーコンは周囲に数秒に1回電波を発信する。受信機が検出したビーコンのIDと電波強度は日時や在室した部屋名とともにサーバに送信され記録される。記録した在室者情報は Web API を通して利用可能であり、勤怠管理システムや来訪促進システムといった様々な応用を想定している。

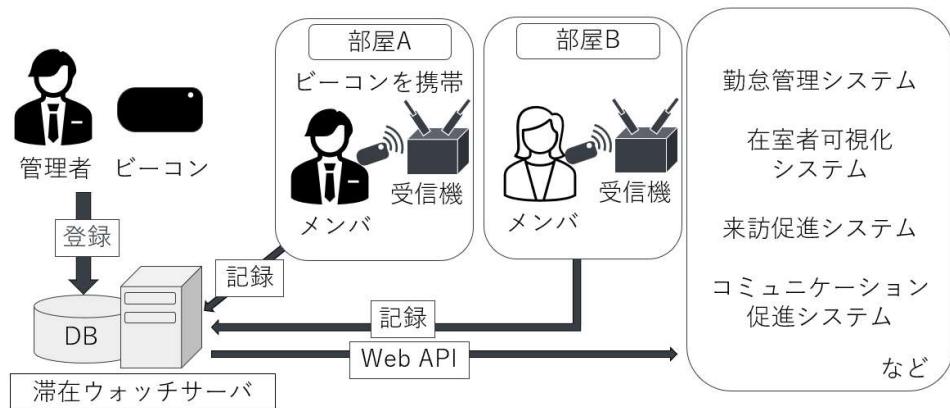


図 3.1: 「滞在ウォッチ」の概要図

本手法に関連するビーコン、受信機、サーバを示す。メンバが持つビーコンには、長期的な運用を考慮しバッテリ交換が可能かつ小型なFCS1301[?]を利用している。ビーコンは図3.2のように様々な大きさや形のものがある。複数のビーコンの比較を表3.1に示す。ボタンは定期的な電波発信に加えて押した時にも電波発信できるため、点検時に利用できる。FCS1301の用途は、財布やパスケースなどの貴重品に付ける紛失防止や子供の荷物などに付ける見守り支援がある。財布やパスケースに取り付けたり入れられるサイズである。そのため、メンバが持つのに適している。またバッテリ交換の際の様子を図3.3に示す。特殊な器具などを使う必要がなく、簡単にバッテリ交換ができる。FCS1301ではボタンを押すとペアリング、長押しするとスリープモードへ移行し、保管の間省電力モードになる。ビーコンの電波

送信の間隔は FCS1301 の規格で最大の 10 秒ごとに設定している。1 部屋ごとに台設置する受信機には図 3.4 の低価格な Raspberry Pi[?] を利用している。1 つの部屋に 1 つずつ受信機を設置する。



図 3.2: ビーコンの種類

表 3.1: ビーコンの比較

ビーコン名	サイズ	バッテリ交換	ボタン
FCS1301	縦 46.0 mm × 横 24.5 mm × 厚さ 3.5 mm	○	○
MAMORIO	縦 35.5 mm × 横 19.0 mm × 厚さ 3.4 mm	×	×
WICED	縦 60.0 mm × 横 37.0 mm × 厚さ 10.0 mm	○	○
estimote	縦 55.0 mm × 横 38.0 mm × 厚さ 15.0 mm	×	×



図 3.3: ビーコンのバッテリ交換



図 3.4: 実際に使用しているビーコンを検知する受信機

メンバがビーコンを携帯し部屋に訪れると、部屋ごとに設置された受信機でビーコンを検出する。受信機が送信するデータを JSON 形式で図 3.5 に示す。Beacons の配列には UUID と RSSI のキーとプロパティを持つオブジェクトが格納されている。UUID はビーコンに割り当てられた固有の ID であり、RSSI はビーコンからの受信信号強度である。rssi の値は DBM(デシベル・ミリワット)で表される。信号強度が高

いほど RSSI の値は大きくなり、信号強度が低いほど RSSI の値は小さくなる。通常 -30dBm から -90dBm の範囲で表され -30dBm から -50dBm の範囲は強い信号とされており、-70dBm から -90dBm の範囲は弱い信号とされている。このデータでは UUID と RSSI のデータを持つオブジェクトが 4 つ存在しているためメンバのビーコンが 4 つ検出されている。さらに BEACONS と ROOMID をラップしたオブジェクトを作成している。ROOMID は各部屋の受信機ごとに割り当てられた固有の ID であり .ENV ファイルに記述されている環境変数である。受信機側のプログラムは Python で実装されており、Python で受信したデータと .ENV ファイルから読み込んだ ROOMID を利用して辞書型で 3.5 のような形式のデータを作り JSON 形式に変換して POST メソッドでサーバに送信する。

```
{
  "Beacons": [
    {
      "uuid": "e7d61ea3f8dd49c88f2ff2484c07ac00",
      "rssi": -60
    },
    {
      "uuid": "e7d61ea3f8dd49c88f2ff2484c07ac01",
      "rssi": -55
    },
    {
      "uuid": "e7d61ea3f8dd49c88f2ff2484c07ac02",
      "rssi": -65
    },
    {
      "uuid": "e7d61ea3f8dd49c88f2ff2484c07ac03",
      "rssi": -75
    }
  ],
  "roomId": 1
}
```

図 3.5: 受信機が 4 人分のビーコン情報をサーバに送信する際のデータ

受信機から送ってきたデータをサーバ側で処理する。データベースには現在在室している人の情報を保存する Stayers テーブルが存在する。サーバ側のプログラムは Stayer テーブルと受信機側から送信されたデータを比較して、Stayers テーブルの情報を更新する。Stayer テーブルのカラムが存在しない時に 3.5 の形式のデータを受信した場合受信した ROOMID と UUID をもとに 4.1 で後述する表 4.1 と Users テーブルと表 3.3 に示す Rooms テーブルを参照して Stayers テーブルに新しいレコードを 4 つのレコードを追加する。この状態で図 3.6 に示すデータを受信した場合、その部屋には誰も存在しておらず在室者は退出したと判断し Stayers テーブルからレコードを削除する。部屋ごとに受信機を設置する際に、設置する

表 3.2: Stayers テーブル

プロパティ	型	説明
id	bigint	主キー
user_id	bigint	副キー（ユーザーテーブルの主キー）
room_id	bigint	副キー（ルームテーブルの主キー）
rssi	bigint	電波強度
create_at	datetime	作成日時
update_at	datetime	更新日時
delete_at	datetime	削除日時

表 3.3: Rooms テーブル

プロパティ	型	説明
id	bigint	主キー
name	longtext	部屋名
create_at	datetime	作成日時
update_at	datetime	更新日時
delete_at	datetime	削除日時

部屋が隣接する場合にビーコンは周囲数メートルから数十メートルに電波を発信するため、隣接した複数の部屋で同様の在室判定を行う可能性がある。そこで、ビーコンが発信する電波強度に着目する。ビーコンの電波強度は表 3.4 に示すように壁や扉のコンクリートや鉄板の障害物によって減衰する [?]. そのため、隣接した複数の部屋では電波強度に明らかな差があると考えられる。受信機から送られてきた電波強度を比較すれば、正確な在室判定が可能である。したがって、在室判定にビーコンの受信電波強度を利用して隣接した部屋での誤検出の防止が可能だと考える。その具体的な例として 3.5 の隣接した ROOMID が別の部屋の ID である場合が挙げるこのパターンの場合 Stayers テーブルに存在している RSSI の値より RSSI の信号強度が高い場合、部屋を移動したと判断し Stayers テーブルの ROOMID と RSSI の値を更新する。

```
{
  "Beacons": [
    ...
  ],
  "roomId": 1
}
```

図 3.6: 受信機が受信するビーコンが存在しない際にサーバに送信するデータ

表 3.4: 高周波 (RF) 電波を反射／吸収する物質

障壁の種類	木材	合成物質	ガラス	水	煉瓦	大理石	土壁	コンクリート	金属
干渉の可能性	低	低	低	中	中	中	高	高	非常に高い

記録した在室者情報は Web API を通して利用可能であり、他のプログラムからも利用できる。具体例として滞在ウォッチの Web サイトが挙げられる。Web サイトを図 3.7 に示す。カラムの一番左には現在の在室者の名前が表示される。これは一意な利用者の名前である。真ん中のカラムには利用者の属性が表示される。例では研究室の名前、研究室における所属するグループの名前、現在の学年が表示されている。右のカラムには部屋の研究室の名前と部屋名が表示される。部屋名のみでは研究室によって同じ名前を使用している可能性があるため、研究室名も表示している。

The screenshot shows a web-based application titled "Stay Watch". At the top, there is a navigation bar with tabs: "New! Stay Watch" (highlighted in blue), "滞在者" (Occupant), "滞在履歴" (Stay History), "利用者情報" (User Information), and "滞在者マップ" (Occupant Map). On the right side of the header is a user profile icon.

The main content area is titled "滞在者一覧" (List of Occupants). It displays a table with five rows, each representing an occupant:

Name	Attribute	Room
kameda	梶研 口けーション B4	梶研-学生部屋
toyama	梶研 B4	梶研-学生部屋
rui	梶研 センシング B4	梶研-学生部屋
ohashi	梶研 口けーション B4	梶研-学生部屋
shamo	梶研 口けーション B4	梶研-学生部屋

図 3.7: 在室者の一覧画面

3.2 在室者情報を用いた応用システムについて

部屋利用するメンバの在室者情報を用いれば、部屋の管理者や利用者にとって便利な応用システムが考えられる。1つ目の応用システムとして、勤怠管理システムが考えられる。勤怠管理システムの代表例としてタイムカードでの打刻や、IC カードや QR コード、スマートフォンによる記録が挙げられる。これらのシステムでは利用者が能動的に記録する必要がある。本研究でのビーコンを用いた手法では能動的な記録を必要とせず、自動で記録されるため、利用者の負担や記録忘れが減ると考えられる。

2つ目の応用システムとして、コミュニケーション機会の損失を減らすシステムが考えられる。研究室では利用者同士の活発なコミュニケーションが求められる。しかし研究室内で目的とする人物と直接会うのを希望していたとしても、その人物が来るかどうかはその人にとって不明である。会うのを希望する人に対して SNS などのコミュニケーションツールを使っていつ頃くるかと言った連絡ができるが、仲が良い人でないとメッセージのやり取りに心理的なハードルが存在する。そこでこれを改善する既存研究として図 3.9 に示す「今日の滞在」というシステムがある。このシステムでは普段の在室者の入退室情報を用いて、来そう、帰りそうという状態を予測し、その予測結果と実際の部屋の在室者と退室者の状態を提示するシステムである。この情報を Web サイトと研究室内のディスプレイに表示し、来そう、帰りそうという表示をもとにコミュニケーション機会の損失を軽減するのが目的である。

3つ目の応用システムとして、来訪促進システムが考えられる。在室者情報を用いたゲーミフィケーションに基づくシステムが考えられる。具体例として在室履歴を用いたペット育成ゲームが考えられる。

在室履歴を用いたペット育成ゲームの概要を図 3.9 に示す。部屋利用者の在室履歴からペットに餌を与えるようなシステムである。部屋利用者はペットに餌を与えるためには部屋に在室する必要がある。そのためペットに愛着があるなら部屋利用者は来訪する必要がありこれによって来訪促進が期待できる。



図 3.8: コミュニケーション機会の損失を減らすシステム

ある滞在条件を満たすとペットの満腹度効率が向上

来訪頻度	同室人数	組み合わせ	満腹度
多い 	多い 	珍しいグループ A D	 増えやすい
少ない 	少ない 	いつものグループ A B C	 増えにくい
曜日によって来訪する人の偏りをなくす		来訪する人の固定化を防ぐ	

図 3.9: ペット育成ゲームのアプリのイメージ

第4章 独立したコミュニティにおける滞在ウォッチの安定運用のためのシステム拡張

滞在ウォッチを複数コミュニティ間で運用するには、複数の問題が存在する。独立したコミュニティで運用できない問題や長期に渡り在室情報を継続的に記録できない問題である。これらの問題を解決した安定運用に向けての取り組みについて述べる。4.1章で既存の滞在ウォッチに独立運用するためのシステムの再構築について述べる。4.2章で利用者の管理とアクセス制御システムの整備について述べる。4.3章でスマートフォンアプリによるビーコンと実デバイスの併用について述べる。

4.1 安定運用に向けた既存システムの再構築

既存の滞在ウォッチは長期に渡り継続的に保守・開発していくのが困難だったため安定運用を目指す上でシステムの再構築を行った。4.1.1章では、サーバ側のシステムの再設計を行い、4.1.2章ではクライアント側のシステムの再設計を行った。

4.1.1 サーバ側システムの再設計と再構築

サーバ側には独立したバックエンドシステムとの連携が容易かつWebアプリケーションのより高い可用性とスケーラビリティが実現できるREST APIを採用した。REST APIは、多種多様なクライアントからアクセスができるため、様々なデバイスやプラットフォームからアクセスできる。これにより、より広いユーザ層からアクセスが期待できる。

滞在ウォッチのサーバ側システムはPythonで構築されていたが、高負荷環境でパフォーマンスに影響を与える可能性があったのと動的型付け言語の開発、保守のしづらさを解消するため、Go言語を使用してシステムの再構築を行った。Web APIは大量の処理をする必要があり、Pythonの実行速度の遅さが、高負荷環境でパフォーマンスに影響を与える可能性がある。また動的型付け言語なため、型の不明確さによってバグが見つけづらく保守性が低い。そこで静的型付けであり、高速な処理能力と小さなメモリフットプリントを持つため、Web APIの開発に適しているGo言語を採用した。Go言語は高いパフォーマンスを持つ。C言語のような低レベルな言語と同等のパフォーマンスを持ちながら、高水準の言語のような簡単な構文を持っているのに加え、並列処理を容易に実現できるため、高負荷な環境でのWeb APIの開発にも適している。そのため、WebAPIの要求に適合する十分なパフォーマンスを提供できる。また構造体とインターフェース型を備えており、WebAPIの開発に必要な柔軟性を持っている。これにより、開発者は、WebAPIを実装するために適した方法を選択できる。さらに、標準パッケージによるHTTPサーバのサポートを持つため、Web APIの開発に必要な機能を簡単に実装できる。

サーバで使用するデータベースは無料で利用でき、高いスケーラビリティを持つため、WebAPIの要求に対応可能なMySQLを採用した。MySQLは多くのプラットフォームでサポートされており、多くの言語に対応しており、開発者にとって選択肢が広がる。またMySQLは優れたセキュリティ機能を持つため、WebAPIのセキュリティの確保ができる。

Go言語でMySQLの操作を行う際Go言語のORMライブラリであるGormを採用した。ORMはデータベースを操作を行うための手法の一つである。これによりデータベース操作を行う際に、SQL文を直接記述する必要がなくなりコードがすっきりし可読性が高くなる。またSQL文を直接記述しないためSQL

表 4.1: Users テーブル

プロパティ	型	説明
id	bigint	主キー
name	longtext	名前
uuid	longtext	使用者のビーコンの UUID
email	longtext	email
role_id	bigint	外部キー (roles テーブルの主キー)
create_at	datetime	作成日時
update_at	datetime	更新日時
delete_at	datetime	削除日時

表 4.2: Roles テーブル

プロパティ	型	説明
id	longtext	主キー
name	longtext	役割の名前
create_at	datetime	ユーザ情報の作成日時
update_at	datetime	ユーザ情報の更新日時
delete_at	datetime	ユーザ情報の削除日時

インジェクション対策になる。SQL インジェクションとは不正な SQL 文の挿入を行い、データベースに対して攻撃を行う手法である。ORM はプログラマが入力したデータを自動的にエスコープし、サイバー攻撃につながるような文字を無効化するため SQL インジェクション攻撃を防止できる。Gorm は Go 言語向け ORM ライブリの一つである。Gorm はデータベースの挿入、更新、削除、検索などの基本的な機能操作をサポートしている。また自動的なマイグレーション機能を持っている。これにより開発者はデータベーススキーマを手動で管理せずに、スキーマの変更を自動的に反映できる。Go の ORM ライブリとして他に Xorm などの選択肢もあったが複雑なクエリを簡単に作成するための豊富なクエリビルダー API を提供しているため Gorm を選択した。

後述 4.2 項で述べる利用者の管理とアクセス制御システムを実装するために、データベースに User テーブルと Roles テーブルを新しく作成した。表 4.1 と 4.2 にそれらのスキーマを示す。既存のシステムでは存在していなかった email プロパティと role_id プロパティを追加した。email プロパティは管理者が利用者の email を登録すると保存される。4.3 で解説するがこの email 存在するかで利用者の認証を行う。role_id は roles テーブルの主キーである。roles テーブルには管理者と一般利用者の 2 つのロールが存在する。役割の名前の変更や役割名が追加された場合にも対応できるように roles テーブルを作成している。

サーバサイドプログラムの開発及びデプロイメントする際にはプログラムを実行するために必要な全ての環境をコンテナ内にパッケージを行い、開発環境と本番環境での環境差を最小限に抑えられる Docker を採用した。Docker を使用すれば、アプリケーションを実行するために必要な全ての依存関係を内蔵したイメージを作成できる。これにより開発者はアプリケーションを実行するために必要な環境を 1 つのイメージにまとめられ、任意の環境に簡単に展開でき開発環境の構築が容易になる。さらに Docker は他のアプリケーションの影響を受けずに実行可能である。これによりアプリケーションが依存しているライ

プラリやフレームワークのバージョンなどの環境の違いによって、アプリケーションが動作しないといった問題を回避できる。またコンテナを水平にスケールアウトできるため、アクセスが集中した場合にもスムーズな処理を行える。デプロイ先は研究室内のサーバを利用した。研究室のサーバの利用はクラウドサービスに比べて長期的にコストを削減できる。

4.1.2 クライアントサイドの実装

クライアントサイドでは、WebAPIを利用して、Web アプリケーションを実装した。Web アプリケーションのフレームワークとして、Next.js を採用した。Next.js は React.js をベースにしており、React.js の開発を効率化に加え、サーバーサイドによるレンダリングのサポートによって SEO 対策やパフォーマンスの向上を実現できる。React.js は JavaScript のライブラリであり、コンポーネントベースのアーキテクチャを採用している。コンポーネントを独立した単位として開発すると、開発者はそれぞれのコンポーネントに集中できるため開発効率が向上する。コンポーネントは再利用可能な単位であり複数のページで同じコンポーネントが利用でき、これにより開発コストを削減し保守性を向上させる。また React は仮想 DOM を採用しており実際の DOM (Document Object Model) を操作する代わりに、JavaScript のオブジェクトとして扱う仕組みである。仮想 DOM は実際の DOM と比較して、更新が必要な部分だけを計算を行うためパフォーマンスの向上が図れる。

基本的に既存の滞在ウォッチで同じように滞在者、滞在履歴、利用者情報のページが存在する。変更点として在室時間をガントチャートで可視化している画面が挙げられる。図 4.1 にその画面を示す。この画面の緑の部分にカーソルを合わせると、入室時間と退室時間を確認できる。日付と部屋を指定するタブがあり、そのタブに応じたそれぞれのデータを表示される。

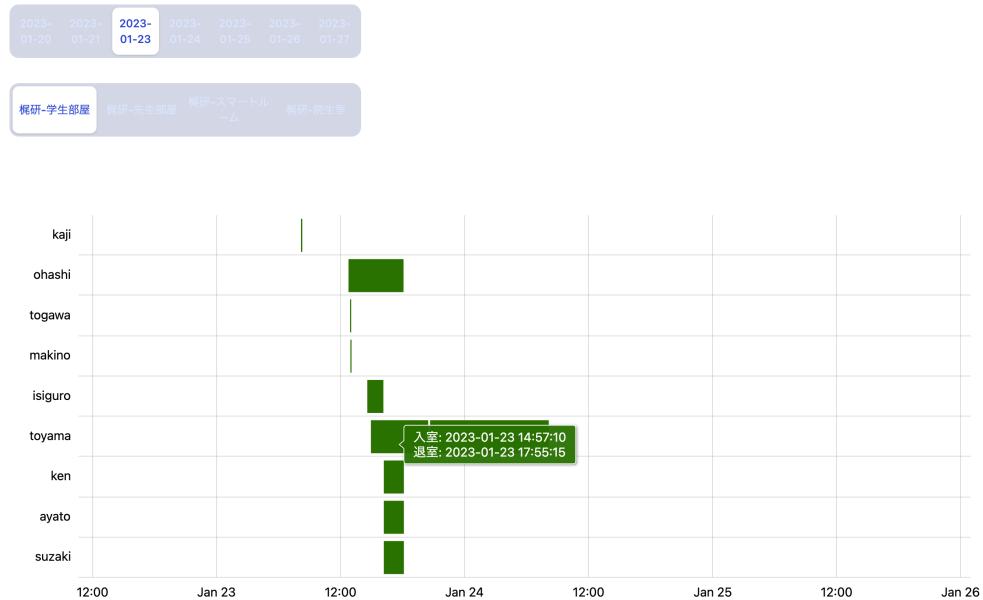


図 4.1: ガントチャートを用いた在室時間の可視化

また滞在情報をフロアマップ上に可視化を行った。図 4.2 にその画面を示す。数字が表示されているところにカーソルを合わせると示した図のように、その部屋に在室している利用者の名前が書かれたツールチップが表示される。既存の滞在ウォッチでは在室者をリスト形式で表示するのみであったが、フロアマップ上に在室者を可視化によって、在室者の位置を把握しやすくなりコミュニケーション促進につながる可能性が高いと考えられる。

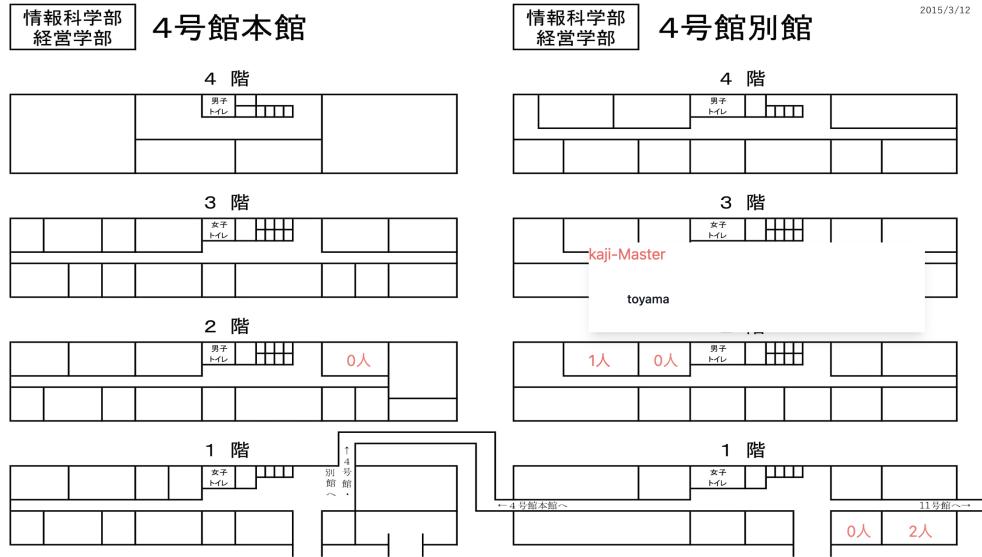


図 4.2: フロアマップを用いた在室者を可視化

4.2 利用者の管理とアクセス制御システムの整備

独立した運用を行うためには利用者の管理とアクセス制御システムの整備が必要である。既存の滞在ウォッチは単独コミュニティでのみの運用を前提としており開発者と管理者が同一であった。Web 上からユーザの登録をするシステムが存在せず開発者がデータベースに対して直接変更を行う SQL を発行していた。これは單一コミュニティでの運用であったため成り立っていたが、複数コミュニティ間で運用を行う場合、コミュニティの数が増えるに連れてシステム開発者の利用者管理の負担が大きくなり運用するのは難しくなる。また在室者情報はプライバシに関わるため、部外者が Web ページにアクセスしても在室者情報を閲覧できないようにする必要があるにも関わらず、誰でも閲覧可能な状態であった。これらを解決するには利用者のアクセス制御システムの整備とコミュニティごとの管理者を作り、各コミュニティで独立した運用を行う必要がある。各コミュニティごとに管理者が存在すればコミュニティの利用者の管理を開発者が全て行う必要がないため負担が軽減される。

そこで Firebase Authentication の Google プロバイダーによる OAuth 認証を実装した。OAuth (Open Authorization) は、ユーザーがサービスプロバイダー（例えば Google, Facebook など）に対して、別のサービスにアクセスするための権限を与えるためのオープンスタンダードの認証プロトコルである。OAuth 以外の認証方式として Basic 認証などがある。Basic 認証はユーザ名とパスワードをベーシック認証ヘッダーに埋め込んで送信する方式である。この方式はセキュリティ上の問題が存在する。パスワードが暗号化されていないため、パスワードが盗まれる危険がある。また、Basic 認証はバックエンド側でユーザ名とパスワードを保存する必要があり、管理を行うにはパスワードのハッシュ化など適切な処理を施す必要がある。それと比較して OAuth 認証では、ユーザのアカウント情報を第三者にアプリケーションに渡さずにアクセス権を付与できるため、セキュリティ上のメリットがある。OAuth 認証では、アクセス権を持つアプリケーションにのみ有効で、期限切れになると、使用できなくなる。これにより、アクセス権を付与したアプリケーションが不正にアクセスするのを防止できる。これらの理由により、OAuth 認証を採用した。OAuth プロバイダーには Apple, Facebook など複数の選択肢がある中で Google を採用した。Google アカウントは世界中で広く利用されており、多くの人がアカウントを持っており、多数の利用者に対応可能である。

Google アカウントを用いた OAuth 認証は、Google アカウントを使って、サービスプロバイダー (Google) に対して、利用者が滞在ウォッチシステムにアクセスするための権限を与える。利用者は Google アカウントにログインし、Google が提供する OAuth 認証プロセスを通じて、滞在ウォッチシステムにアクセス

できる。OAuth認証により、利用者は自分のGoogleアカウントの情報（パスワードなど）を滞在ウォッチシステムに入力をせずに、安全にアクセスできる。しかしGoogleアカウントによるOAuth認証だけでは意図しない外部のユーザのアクセスを防げない。なぜならGoogleアカウントを持っているユーザなら誰でも認証を行えるためである。そこでGoogleアカウントによるOAuth認証に加えて滞在ウォッチサーバ側で独自のユーザ認証を行った。具体的には滞在ウォッチ管理者が登録したユーザのみを認証するというものである。滞在ウォッチサーバのデータベースにはGoogleのアカウントのユーザ名であるメールアドレスが保存してある。この保存されたメールアドレスは滞在ウォッチの管理者が登録したものであるため、メールアドレスが存在する場合は管理者の許可のある正規のユーザと判別できる。

管理者のユーザ登録フロー図を図4.3に示し具体的に説明する。まず初めに利用者は管理者に対してGoogleアカウントを報告する。報告後管理者がWebの管理者ページのユーザ登録画面からGoogleアカウントを登録する。ユーザには2種類のパターンが想定される。

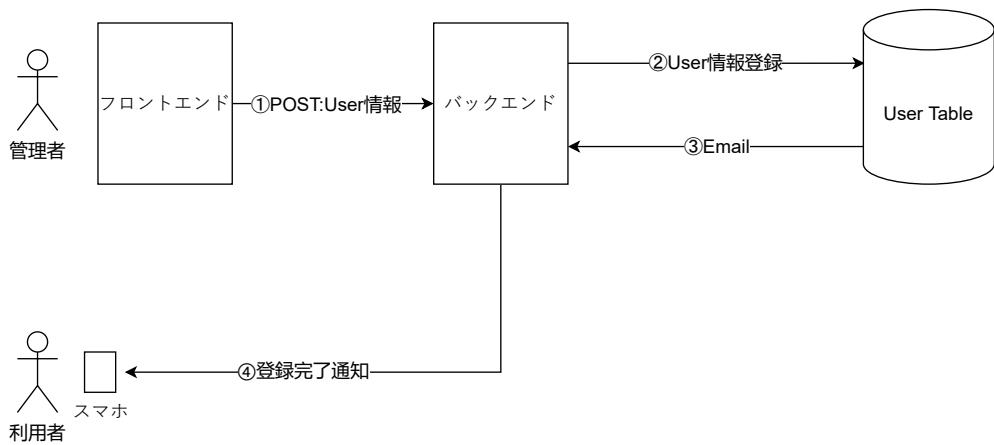


図4.3: 認証システム：管理者側のユーザ登録フロー図

まず1つ目はBLEビーコンが既に登録済みのユーザ場合の登録である。図4.4にその登録画面を示す。この登録画面は以前から滞在ウォッチユーザとして登録がされておりビーコンに関する情報とユーザ情報が既に存在するが、Googleアカウントの情報とユーザロールがないユーザを想定している。ユーザネームはデータベースに存在しているためバックエンドからデータを取得できる。この登録画面は、以前から「滞在ウォッチユーザ」として登録されているユーザ向けで、ビーコンに関する情報やユーザネームの情報は既に存在しており、Googleアカウントの情報やユーザロールのみがないユーザを対象としている。取得した利用者のユーザネームデータを管理者が選択できるようにセレクトボックスを配置した。ユーザロールの項目でユーザの権限レベルの指定ができる。一般ユーザと管理者ユーザの2つがあり、一般ユーザは滞在ウォッチの在室者情報の閲覧などを行え、管理者ユーザは一般ユーザの権限に加えてユーザの登録を行える。管理者が登録ボタンを押すと、POSTリクエストがサーバ側に送信される。登録されると管理者が選択したデータベースに存在するユーザにメールアドレスが紐づき、サーバ側は対象のメールアドレスに対して登録完了のメールを送信する。メールアドレスに対して通知を行うのはユーザがいつ登録が完了したのか把握できるようにするためである。2つ目に想定されるのがビーコンが登録されていない新規ユーザである。登録画面を図4.5に示す。4.4との違いとして4.5の緑枠で囲まれたところが異なっている。これは4.3章で説明するスマホビーコンを使用する新規ユーザを想定しており新規ユーザにはユーザネームがないため管理者はユーザネームを入力する必要がある。その他は先に述べた登録画面と同じであり、登録ボタンを押しサーバに対してリクエストを送信する。先に述べたものとの違いはサーバ側は新規のUUIDの生成を行うところである。スマホビーコンの初期セットアップ時にUUIDの設定を行う、その際ユーザのGoogleアカウントに対応したUUIDをセットアップできる。

メンバーの招待

The screenshot shows a registration form titled 'メンバーリスト登録' (Member List Registration). At the top, there are two tabs: 'BLEビーコン登録済み' (BLE Beacon Registered) and 'BLEビーコン未登録' (BLE Beacon Not Registered), with the former being active. The main form contains three fields: 'ユーザ選択 *' (User Selection *), 'Gmailアドレス *' (Gmail Address *), and 'ユーザロール *' (User Role *). The 'ユーザ選択' field has a dropdown menu with 'ユーザを選択' (Select User). The 'Gmailアドレス' field contains 'your@gmail.com'. The 'ユーザロール' field has a dropdown menu with '一般ユーザ' (General User). A large blue button at the bottom right is labeled '登録する' (Register).

図 4.4: BLE ビーコン登録済みの登録画面

次にユーザ側の認証のフローを図 4.6 に示しその説明する。Firebase Auth で認証が成功すると JWT(JSON Web Token) トークンが発行される。Firebase Auth の JWT トークンは Firebase Auth が認証済みのユーザーを確認するために使用するトークンである。JWT トークンは、JSON 形式の文字列では発行者、トークンの有効期限、トークンの使用目的、サブジェクトが含まれておりユーザを一意に識別可能である。JWT トークンは、フロントエンドから滞在ウォッチのバックエンドに送信される。滞在ウォッチのバックエンドは Firebase Auth の API を使用して、JWT トークンの検証を行い、トークンが有効であるか確認する。有効な JWT トークンであった場合、アプリケーションのバックエンドは、そのトークンに含まれる情報を使用して、Firebase Auth が認証済みかを確認する。認証済みであった場合は次にメールアドレス情報を確認する。データベースにメールアドレスが存在する場合フロントエンドに対してアクセスの許可を行う。このプロセスによって不正なユーザの在室者の閲覧を防いでいる。よって適切な範囲での在室者情報の共有が可能である。

メンバーの招待

The screenshot shows a registration form for users who have not registered via BLE beacons. The form includes fields for 'User Name *' (containing 'your name'), 'Gmail Address *' (containing 'your@gmail.com'), and 'User Role *' (set to 'General User'). A large blue 'Register' button is at the bottom.

図 4.5: BLE ビーコン未登録の登録画面

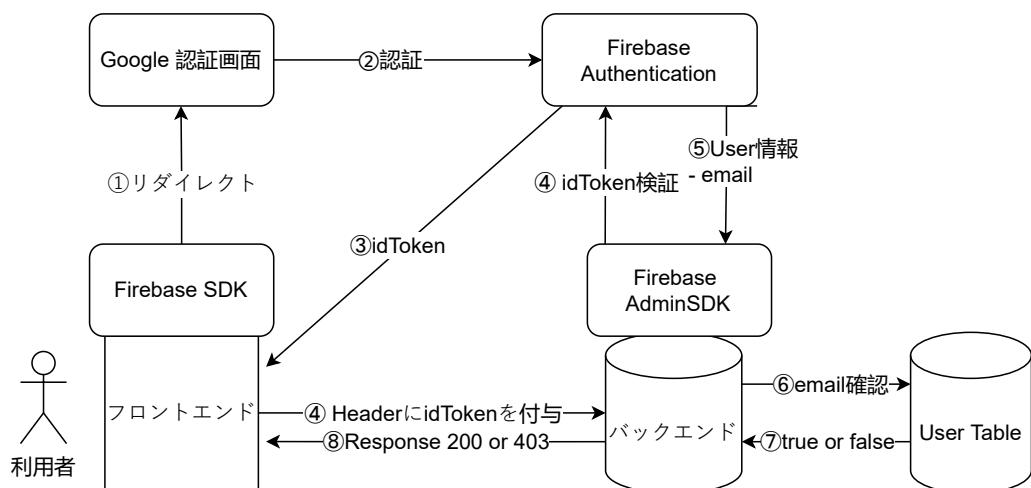


図 4.6: 認証システム: ユーザ側のフロー図

4.3 スマートフォンアプリによるビーコンと実デバイス

本節ではBLEビーコンのアプリケーション化及びそれらのハイブリッド活用について述べる。初めに滞在ウォッチで使用しているBLEビーコンの仕様について説明する。BLEビーコンとはBLE規格のペリフェラル動作を用いて、周囲にアドバタイズを行うデバイスとセントラル動作を用いて周囲をスキャンするデバイスによって構成される。ペリフェラル動作とは通信を受け付ける子機としての動作である。セントラル動作によってスキャンしたデバイスに反応しGATT(Generic ATTribute profile)内に保持したサービスやキャラクタリストリックのデータの送受信を行う。キャラクタリストリックとは、そのデバイスが保持するサービスやデータであり、本研究では個人識別符号として利用するUUIDが該当する。UUIDとは、オブジェクトを一意に識別するために重複がない前提で用いる128bitの数値である。キャラクタリストリックのデータとしてUUIDをビーコンは保持している。

4.3.1 BLEビーコンのアプリケーション化

実デバイスによるビーコンは低コストで携帯性も高いが、滞在ウォッチの運用において利便性が低い点があり、可用性の低さに繋がる問題がある。可用性とは、メンバの在室情報が長期にわたって継続的に記録される能力と定義する。利便性において問題となるのは、バッテリ切れによる問題や、初期設定が複雑な点である。我々が使用しているビーコンFCS1301は、CR2016というバッテリを使用している。このCR2016は一般的なコイン型リチウム電池であり、バッテリ残量を外見上から判断できない。バッテリ残量を確認するためには、ビーコンメーカーが指定したアプリケーションを使用し、実デバイスによるビーコン一台ごとに接続する必要がある。その作業をメンバは手間と考える恐れがある。手間を理由に放置される可能性があり、放置された場合可用性が低下する。また初期設定時もビーコンメーカー指定のアプリケーションを使用する必要があり、管理者がUUIDを入力する必要がある。UUIDは英数混合の32文字で構成されている上に、間違えてしまった場合は滞在ウォッチに記録されない。この作業を新規で利用するメンバの人数と同じ回数行う必要があり、管理者に多大な負担がかかる。また、使用しているビーコンはバッテリ切れによるデータ初期化が発生しない点を重視して購入を行ったが、システム運用中にバッテリ切れが発生していない状態でも、UUIDが設定したものと異なる数値に変更されるケースが見られた。システム上メンバに割り当てられたUUIDの検知によって在室状態を検知するため、UUIDが勝手に書き換わる状態では可用性が維持できない。これらの状況から、実デバイスによるビーコンのみの運用では、メンバにとっての利便性が低下してしまい、それが原因としてシステムの可用性が低下してしまうと考えられる。

上記の問題のアプローチとしてメンバの利便性を向上させるため、スマートフォンアプリによるビーコン動作を行った。スマートフォンアプリを使用した場合、BLEビーコンと違いスマートフォンは通常バッテリが内蔵されているため、ビーコンとしての動作に必要なバッテリをスマートフォンのバッテリで代用できる。その結果、実デバイスによるビーコンで発生していたバッテリ交換をする手間が削減され、利便性が向上すると考えられる。またスマートフォンユーザによってスマートフォンはコミュニケーションツールとしての用途から、バッテリを維持する傾向があるため可能性の向上も期待できる。実デバイスによるビーコンは、バッテリ残量やバッテリ切れを通知するユーザインターフェースを持たない。しかし、スマートフォンは液晶画面を持っており、スマートフォン自体のバッテリ残量やスマートフォンアプリによるビーコンの動作状況などの可視化が可能である。そのためビーコン動作状況の表示によってビーコン動作の停止に気が付きやすく、メンバによるアプリケーションの再起動が行われた場合、可用性の向上が期待できる。これらのメリットからスマートフォンアプリによるビーコン動作がユーザの利便性を向上させ、システムの可用性へつながると考えた。

スマートフォンビーコンは基本的にバックグラウンドに常駐させる利用法を想定し実装した。既存研究ではメンバに実デバイスによるビーコンを携帯させ、能動的な記録動作の必要がない。スマートフォンのバックグラウンドに常駐させる方法は実デバイスによるビーコンと同様に能動的な記録動作の必要がないため実デバイスによるビーコンと同等の利便性がある。そのためバックグラウンドへの常駐がアプリケー

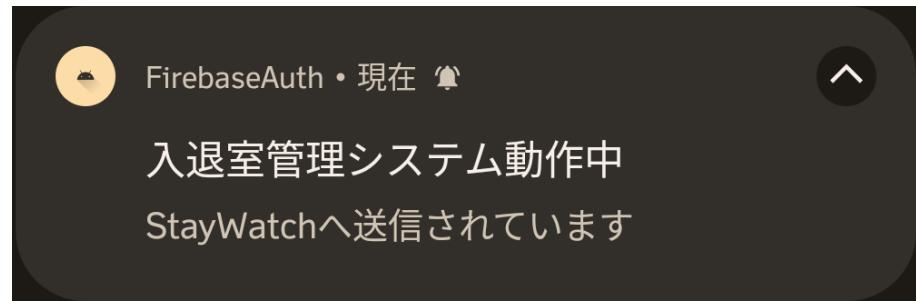


図 4.7: 通知領域におけるビーコンの動作状況の表示

ション化の前提とした。その点を踏まえて動作プラットフォームを選定した。選定理由は技術調査をした際に、現状の iOS ではフォアグラウンド動作はするもののバックグラウンド動作に制限がかかっており実デバイスによるビーコンと同等の利便性の担保が困難であると判明したため、対応プラットフォームを Android のみとした。

実装に伴い、アプリケーションのプログラムは Kotlin を用い、Bluetooth 関連の実装においては Alt-Beacon というライブラリを利用した。選定した理由としては、Android 標準のライブラリについての情報が少なく、一部公式ドキュメントが削除されている中で AltBeacon に関する情報は多く提供されているからである。ビーコン動作をする上で、Android のプライバシ規則によって、バックグラウンド上でのユーザへの通知無しでの動作は禁止されている。その点と、先述のスマートフォンが表示機能を持つ点を利用して図 4.7 に示す通り通知領域上でビーコン動作中は常時表示する仕様とした。この仕様によって Android のプライバシ規則を遵守している。さらにユーザが入退室のたびにアプリの起動を行う必要がない実デバイスによるビーコンと同様の利便性を担保した。

ビーコンのアプリケーション化に伴い、管理者が実デバイスによるビーコンで行っていた登録作業をより簡略化した。従来では管理者が新しいメンバが増えるたびにシステムに Google アカウント、名前、UUIDなどを登録し、割り当てられた UUID を実デバイスによるビーコンに設定する作業を行っていた。しかし先述の Firebase によるログインを利用しメンバに紐づいている Google アカウントを用いて、スマートフォンビーコン上で利用する UUID を自動で設定できるようにした。図 4.8 に示すサインインで管理者が登録した Google アカウントと一致する場合にログインを行う。その後 API からメンバの情報を受け取り UUID をアプリ内部に設定する。以降はログイン時に受け取った UUID を用いてビーコン動作を行う。

4.3.2 スマートフォンビーコンと実デバイスによるビーコンのハイブリッド活用

スマートフォンビーコンのみを利用する場合、様々な状況下でメンバの継続した利用が困難であるため、実デバイスによるビーコンと併用できるシステムとした。長期にわたり継続的に利用するメンバにとって、実デバイスによるビーコンは先述の通りバッテリ交換の手間がある。メンバはその手間からバッテリ交換をしないで放置する場合がある。その場合は在室情報が記録されず可用性が低下する。スマートフォンビーコンはバッテリ交換を手間に考えて放置するメンバにとって、バッテリ交換の手間がないため有用である。しかしそマートフォンを携帯していないメンバ、スマートフォンビーコンの利用に伴うバッテリ消費が気になるメンバなども想定される。

実デバイスによるビーコンとスマートフォンビーコン双方にメリットデメリットが存在する。それらはメンバの環境によって重要視する点が異なる。スマートフォンにアプリケーションの導入を拒否する理由として、実デバイスによるビーコンで十分だと考えている、スマートフォンのリソースを BLE ビーコンに割り当てたくない、アプリケーションの導入によるセキュリティリスクを懸念している、新しい技術に対して不信感を抱いている、等が挙げられる。アプリがスマートフォンに対応しておらず利用できない状態として、対応していない OS 及びバージョンを利用している、スマートフォンの性能が不十分、スト



図 4.8: アプリケーションのログイン画面

レージに十分な空き容量がない、アプリケーションの導入が制限されている端末、等が挙げられる。また実デバイスによるビーコンの携帯を拒否する動機として、スマートフォンビーコンで十分だと考えている、実デバイスによるビーコンの紛失を懸念している、実デバイスによるビーコンの携帯を手間に感じている、ミニマリズムを信奉しており、身の回りに新しいものを増やしたくない、等が挙げられる。

メンバの思想や所有している端末の都合など様々な要因が存在しており、それらをすべて満たした仕様の策定は困難である。よってスマートフォンビーコンと実デバイスによるビーコンのハイブリッド化によってメンバに応じた選択を可能にした。様々な環境の中でも選択肢を提供し、ユーザの利便性向上によってメンバのより長期に渡る継続的な利用が促進できデータの可用性向上につながると考えた。

ハイブリッド化にあたってスマートフォンビーコンで利用する UUID を実デバイスによるビーコンで利用する UUID と同じ値に設定しメンバの在室情報を記録している。スマートフォンビーコンと実デバイスによるビーコン双方をメンバが受け入れて同時に利用した場合、同じ UUID を利用しているため片方が停止した場合でも、もう片方のビーコンが検知され在室判定ができる。この方法は、メンバの利便性を向上できるのみならず、継続的にデータを記録する観点から見ても有用である。実デバイスによるビーコン、スマートフォンビーコンの単独対応とハイブリッド化を表 4.3 で比較する。

滞在ウォッチが実デバイスによるビーコンのみに対応していた場合は長期にわたり継続的に利用するメンバにとってはビーコンの電池交換が手間である。短期的に利用するメンバにとってはビーコンを携帯するだけで済み手軽である。しかしビーコンを不注意などで携帯していなかった場合はシステムに記録されない。スマートフォンビーコンのみに対応していた場合は長期に渡り継続的に利用するメンバにとってはアプリは手間が少ない。短期的に利用するメンバにとってはアプリケーションの登録は手間である。しかし、スマートフォンが利用できない状態ではシステムに記録されない。ハイブリッド化をした場合は、それぞれのユースケースに対応できる。

表 4.3: 各ビーコンのみ対応時とハイブリッド対応時の可用性の比較

メンバ システム	長期 メンバ	短期メンバ	スマホ不携帯	ビーコン不携帯
実デバイス ビーコンのみ	△ バッテリ交換の手間	○	○	×
スマホ ビーコンのみ	○	△ インストールの手間	×	○
ハイブリッド 対応	○	○	○	○

第5章 おわりに

5.1 まとめ

今回 BLE ビーコンを用いた在室者検出システム「滞在ウォッチ」を複数間コミュニティで連携するためのシステム拡張を行なった。滞在ウォッチは BLE ビーコンをメンバに持たせて、自動的に動作するシステムである。滞在ウォッチを複数間コミュニティで連携するには、独立したコミュニティで運用できない、長期に渡り在室情報を継続的に記録できない問題を解決した安定運用を行う必要がある。独立したコミュニティの運用へのアプローチとして利用者の管理とアクセス制御システムの整備を行なった。その結果、スケーラビリティの高い利用者の管理と適切な範囲での在室情報の共有が可能となった。長期に渡り在室情報を継続的に記録できない問題へのアプローチとしてスマホアプリによるビーコン化と実デバイスの併用を行なった。その結果、実デバイスのみで在室管理していた時と比べて、長期に渡り在室情報を記録できるようになった。

5.2 今後の課題

今後の課題としてまず BLE ビーコンによる在室判定の誤判定が発生する問題がある。例えば受信機の設置してある部屋の上の階の部屋にビーコンを持ったメンバがいる場合に受信機が設置してある部屋に入室したと判定される場合が挙げられる。これは BLE の電波が壁を通過するために発生する問題である。

この問題への対応策として、信号強度を利用する方法がある。信号強度を利用する方法は、受信機が受け取ったビーコンの信号強度を測定し、その部屋にいるかどうかを判定する方法である。信号強度は、送信機から受信機までの距離に比例して減衰する。そのため、受信機が受け取ったビーコンの信号強度が弱い場合は、その部屋にいる可能性が低いと判断できる。具体的には、受信機が受け取ったビーコンの信号強度を測定し、それが一定のレベル以上であれば、その部屋にいると判断し、そうでなければ、その部屋にいないと判断する。しかしこの方法は、受信機が受け取ったビーコンの信号強度を基にして、その部屋にいるかどうかを判断するため、部屋の形状や構造によっては、誤った判断をする可能性がある。全ての部屋で正しく判定するには、部屋ごとに応じた閾値を設定する必要があり、コストがかかる。ただし滞在ウォッチの普及度が上がり多くの部屋で受信機を設置できれば部屋ごとの電波強度を比べればいいため、部屋ごとに閾値を設定する必要はなくなる。

他の方法としては、BLE の以外の通信規格を使用する方法が考えられる。Ultra-Wideband(UWB) デバイスや Zigbee デバイスを使用した方法がある。UWB デバイスは、壁を貫通しにくく、高精度な位置測定を行えるが、新しい技術であるため、普及が送れており、対応するデバイスやアプリケーションが少ないのが欠点がある。Zigbee デバイスは、ワイヤレス通信技術の一種で、壁を貫通しにくい低効率の電波を使用し範囲が狭いため、隣接する部屋に影響を与えていくが、伝達速度が遅いため、大量のデータを伝送する場合には向いていない欠点がある。UWB デバイスや Zigbee デバイスを使用した方法は上記で述べた欠点もあるが部屋を貫通しにくく、誤判定が減少すると予想されるため、導入を検討する価値があると考える。

次に、BLE ビーコンの UUID の設定に手間がかかる問題がある。現在ビーコンの UUID をセットアップする際には、専用のアプリを使って UUID の書き込みを行う必要がある。この問題への解決策として、UUID を書き込むアプリを独自に作成する方法が考えられる。独自のアプリを作成すればそのアプリが滞在ウォッチサーバ側に対して HTTP リクエストを送りそのレスポンスとして UUID 取得してそれを BLE

ビーコンに書き込むようなシステムを構築できると考えられる。BLE ビーコンの提供先である株式会社フォーカスシステムズに対してビーコンへの書き込みを行う API を提供してもらえないか確認を行ったが、結果としてそのような API は提供していないとの回答であった。しかし専用のアプリで BLE ビーコンへの UUID の書き込みを行なっているため不可能ではないと考えられる。そのため BLE ビーコンのセットアップの設定の簡略化を行うために、4.3 章で話したスマホビーコンの機能として UUID を BLE ビーコンに書き込む機能の検討を行っていきたい。

プロジェクトの課題として、滞在ウォッチの複数コミュニティ間連携を実際に行い運用を行う必要がある。本研究では、滞在ウォッチが複数コミュニティ間で連携できるように独立したコミュニティにおいて滞在ウォッチの運用ができるように検討を行ったが、実際に複数間コミュニティ間で運用後に発生する問題も考えられる。そのため、運用後はメンバ、管理者、利用者からの意見や得られたデータを元に評価、システムの改善を行う予定である。また現状のシステムでは複数コミュニティ間でのコミュニケーションを促進する機能の実装には至っていない。そのため複数コミュニティに向けたイベント開催機能などを実装し運用・評価する必要がある。

謝辞

本研究を進めるにあたり、多くの御指導、御鞭撻を賜わりましたまず研究の主要な主導者である梶 克彦准教授に深く感謝致します。教授は私の意見を真剣に受け止め、熱心な指導をしてくださいました。また素晴らしい研究室の環境を提供し、私の成長に貢献してくださいました。次に、日頃から熱心に討論、助言してくださいました梶研究室のみなさんに深く感謝致します。さらに家族、友人、同僚にも感謝致します。彼らは私が論文を書くために専念できる環境を整えくれ私を励ましてくれました。最後にこの卒業論文を書けたことに感謝します。研究を通じて多くの学びを得て、自分の人生に大きな影響を与えた人間としてより成長できました。