修士論文

公民館向け建物 OS における設備制御と資源最適化に関する研究

2210119　CHEN　Xiang

主指導教員　丹　康雄
審査委員主査　丹　康雄
審査委員　長谷川　忍
ベウラン ラズバン
リム 勇仁

北陸先端科学技術大学院大学
先端科学技術研究科
(情報科学)

令和 6 年 2 月

Abstract

In modern building management, the role of Building Operating Systems (OS) is becoming increasingly crucial. A Building OS is a system that integrally controls the facilities and services within a building, aiming to enhance operational efficiency and safety. In Japan, Shimizu Corporation's DX-Core has been a pioneer in this field, establishing new standards in building operations and driving automation and efficiency in building management.

However, applying Building OS in community facilities like public halls presents unique challenges, differing from commercial and office buildings. Public halls, in particular, serve as regular community activity centers while also needing to function as emergency shelters during disasters. This dual role demands a Building OS capable of flexible and rapid response in both normal and emergency situations.

This research proposes a new Building OS, 'Multimode Kominkan Operating System (MKOS),' designed to meet these specific needs. MKOS encompasses various modes covering everyday operational management of public halls to functionalities needed for emergency shelters during disasters. This system achieves optimal facility control and resource optimization, addressing the varying environmental conditions and diverse user needs that traditional Building OS could not.

The core of MKOS is the 'Data Sharing Interpretation Module,' efficiently processing commands from multiple services and avoiding device control conflicts. This module allows MKOS to translate abstract user commands into specific facility operations, optimally managing the building's environment. Moreover, MKOS provides developers and programmers with clear, user-friendly APIs, ensuring the system's scalability and flexibility.

With these features, MKOS has the potential to become the new standard Building OS for community facilities like public halls. This research examines the design, implementation, and through simulated experiments, explores the effectiveness and potential of MKOS.

概　要

現代の建物管理において、建物 OS（Operating System）の役割はますます重要になっている。建物 OS は、建物内の設備やサービスを統合的に制御し、効率的な運用と安全性の向上を図るシステムである。日本において、この分野の先駆者である清水建設の DX-Core は、建物の運用における新しいスタンダードを築き、建物管理の自動化と効率化を推進してきた。

しかし、公民館のような地域コミュニティ施設における建物 OS の適用は、これまでの商業ビルやオフィスビルとは異なる課題を抱えている。特に、公民館は日常的なコミュニティ活動の場であると同時に、災害時には緊急避難所としての機能を果たす必要がある。この二重の役割は、建物 OS に対して通常時と非常時の双方において柔軟かつ迅速に対応できる能力を求める。

本研究では、このような特殊なニーズに応えるために、新たな建物 OS として「Multimode Kominkan Operating System（MKOS）」を提案する。MKOS は、公民館の日常的な運用管理から、災害時における緊急避難所としての機能までをカバーする多様なモードを備えている。本システムは、従来の建物 OS では対応が難しかった、変化する環境条件と多様な利用者ニーズに対して、最適な設備制御と資源最適化を実現する。

MKOS の核となるのは、「データ共有解釈モジュール」であり、複数のサービスからの命令を効率的に処理し、デバイス制御の衝突を回避する。このモジュールにより、MKOS は利用者からの抽象的な命令を具体的な設備操作に変換し、建物内の環境を最適に管理する。さらに、MKOS は開発者やプログラマに対して、明確で使いやすい API を提供し、システムの拡張性と柔軟性を保証する。

以上の特徴により、MKOS は公民館のような地域コミュニティ施設における新たな建物 OS の標準となり得る。本研究では、MKOS の設計、実装、そして模擬実験を通じて、その効果と可能性を詳細に検証する。

図 目 次

表 目 次

目 次

第1章　序論

1.1　研究の背景と目的

1.1.1　研究背景

現代の建物運用では、効率的なエネルギー使用、快適な居住環境、安全性の向上が重要視されている。これらの目的を達成するため、建物管理システムの進化が求められており、特に建物 OS の役割が注目されている。図 1.1[1] は建物 OS の概念を示す。建物 OS は、建物内の様々な設備やシステムを統合し、総合的な制御を可能にする。しかし、既存の建物 OS は主に商業施設やオフィスビル向けに開発されており、公共施設や地域コミュニティ施設の特有のニーズには十分対応していない。特に、公民館のような施設では、日常的なコミュニティ活動の支援と災害時の避難所としての機能の両方を果たす必要がある。[2]

図 1.1: 建物 OS の概念図

1.1.2 研究目的

この研究の主な目的は、公民館向けの新しい建物 OS、「Multimode Kominkan Operating System (MKOS)」の設計と実装にある。MKOS は、公民館の日常的な運用管理と災害時の非常モードの両方をサポートすることで、これまでの建物 OS の限界を克服することを目指している。さらに、MKOS は複数のサービスからの命令を効率的に処理し、デバイス制御の衝突を回避するための「データ共有解釈モジュール」を備えている。このモジュールは、抽象度の異なる命令を解釈し、具体的な設備操作に変換することが可能である。本研究は、MKOS の設計原理、主要コンポーネント、そして実装方法を詳細に検討し、システムの有効性を模擬実験を通じて評価する。また、開発者やプログラマがシステムを容易に利用できるよう、明確で使いやすい API の提供にも焦点を当てる。これにより、公民館のような地域コミュニティ施設における建物管理の新たなスタンダードを築くことを目指す。

1.2 研究の範囲と重要性

1.2.1 研究範囲

本研究は、公民館という特定のコンテキストに焦点を当て、その運用に適した建物 OS の設計と実装を目指している。研究の範囲は以下のように定義される。

1. **システム設計：**Multimode Kominkan Operating System（MKOS）の基本アーキテクチャとコンポーネント設計
2. **モジュール開発：**特に「データ共有解釈モジュール」の開発に重点を置き、多様な命令を効率的に処理する機能を含む
3. **モード切り替え機能：**平常時と非常時の運用モードを切り替える機能の設計
4. **API 開発：**開発者やプログラマがシステムを容易に利用できるよう、使いやすい API の提供
5. **模擬実験：**MKOS の機能性と効果を評価するための実験

1.2.2 研究の重要性

公民館は、日常のコミュニティ活動の場であり、災害時には避難所としての役割も果たす。この二重の機能は、公民館向けの建物 OS に特有の挑戦をもたらす。

MKOS の開発は、以下の点において重要である。

1. **災害対応：**非常時における迅速かつ効果的な運用をサポートすることで、地域コミュニティの安全とサポートを強化する
2. **エネルギー効率と環境快適性：**効率的な設備制御を通じてエネルギー消費を最適化し、利用者の快適性を向上させる
3. **汎用性と拡張性：**明確な API を提供することで、システムの汎用性と将来的な拡張性を確保する
4. **技術革新：**公共施設の建物管理における新しいパラダイムを提案し、他の類似施設における適用の可能性を開く

この研究は、公民館だけでなく、類似の多目的公共施設における建物管理システムの発展に貢献する可能性を秘めている。また、公民館のような地域コミュニティ施設に特化した建物 OS の開発は、地域社会のニーズに対応するための重要なステップである。

第2章　公民館とその役割

2.1　公民館の概要

公民館は、地域コミュニティの中核としての役割を担っている。図 2.1 [3] は公民館の役割を示す。これは、地域住民が集まり、様々な社会的、教育的、文化的活動を行う場所である。公民館の主な目的は、地域コミュニティの絆を強化し、住民間の交流と協力を促進することにある。

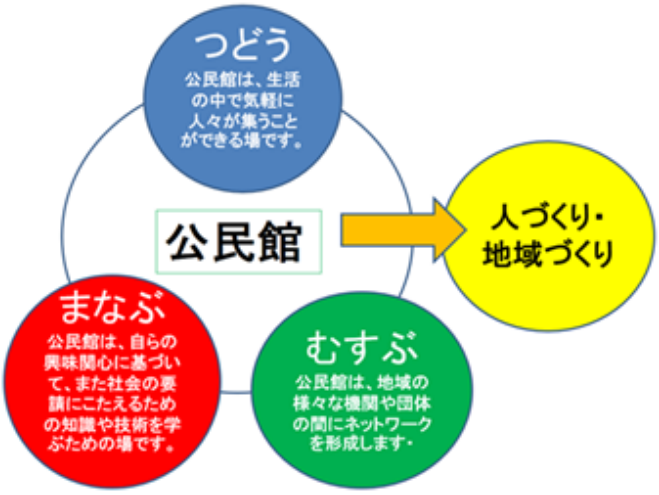

図 2.1: 公民館の役割

多くの公民館では、様々なクラスやワークショップ、イベントが定期的に開催される。これらは地域住民に学びの機会を提供し、異年齢層や異なる背景を持つ人々の間の相互理解を促進する。また、公民館は地域の情報発信の場としても機能し、住民にとって重要な情報源となっている。

公民館の設計は、多目的利用が可能であることが求められる。一般的に、広いホール、会議室、教室、図書室、時には体育施設やキッチンなど、さまざまな機能を備えた施設が含まれる。これにより、地域によって異なる需要に柔軟に対応することができる。

また、公民館は災害時の避難所としての役割も果たす。地震や洪水などの自然

図 2.2: 災害時の避難所としての公民館 [2]

災害が発生した際、公民館は地域住民の安全な避難場所となり、緊急時のサポートを提供する。このため、公民館の建物は安全性が高く、緊急時には迅速に機能を切り替えることができるように設計されている必要がある [4]。公民館の運営は通常、地方自治体や地域のボランティア団体によって行われる。これにより、地域のニーズに密接に対応し、住民の参加と協力を促進することが可能となる。

2.2　公民館の社会的、文化的役割

公民館は日本の地方自治体によって運営され、住民の教育、文化活動、および地域コミュニティの発展を支援する目的で設立される。これら施設は、住民の教養向上、健康増進、情操教育に貢献し、生活文化の振興や社会福祉の向上に寄与する。[5]

公民館が提供するサービスと活動は多岐にわたる。範囲は定期講座、討論会、講習会、講演会、展示会の開催に及ぶ。これら活動は住民に知識や技能を提供し、生涯学習の機会を広げる役割を果たす。また、図書や資料の提供を通じて、地域の文化的資源としての機能も担う。[6]

体育やレクリエーション活動は、住民の健康維持や社会的交流の場を提供する。これら活動は、年齢や背景の異なる住民が集まり、相互理解と協力の精神を育む機会を創出する。さらに、地域団体や機関との連携を通じて、公民館は地域社会のネットワークを強化し、地域の問題解決に貢献する。

公民館は施設を公共的に利用可能にし、地域のイベントや活動の中心地となる

ことが多い。このようにして公民館は、地域住民の生活の質の向上と共に、地域コミュニティの結束を強化する重要な役割を担う。

公民館は、単なる教育やレクリエーションの場にとどまらず、地域の伝統や文化の保存、発展にも寄与する。地域の祭りや行事、伝統芸能の展示などを通じて、地域固有の文化的アイデンティティを育み、次世代への伝承をサポートする。

これら活動を通じて、公民館は地域住民の学びの場としてだけでなく、地域コミュニティを結束させ、強化する中心的な役割を果たす。公民館の存在は、地域社会の持続可能な発展に不可欠であり、その社会的および文化的役割は計り知れない価値を持つ。

2.3 非常時における公民館の重要性

2.3.1 非常時について

内閣府防災担当が定める警戒レベルシステムは、非常時の状況認識と対応策の決定において、指標としての役割を果たす。このシステムは、災害の進行状況や緊急性を明確にし、関連する各機関への迅速かつ適切な情報伝達を保証する [4]。警戒レベルは 1 から 5 のスケールで表され、レベルが上がるにつれて、公民館に求められる役割の緊急性と範囲が拡大する。図 2.3 [7] は警戒レベルによる防災気象情報と対応する行動である。

レベル 1（災害発生の可能性あり）では、公民館は地域住民へのリスク情報の普及と意識啓発を行う。レベル 2（災害注意報）で、公民館は避難準備や避難勧告の受信・発信基地として機能し始める。レベル 3（災害警報）以上では、公民館は避難所としての機能を全面に展開し、レベルが高まるにつれて、医療支援、物資供給、情報提供所としての活動を強化する。

レベル 4（避難指示（緊急））とレベル 5（災害緊急事態）では、公民館は避難生活の維持管理と心理的サポートの場を提供し、地域の安全と住民の生命保護を最優先の任務とする。このように、公民館は災害対応の各段階において中核的な役割を果たし、防災・減災の戦略的な拠点としての役割を担う。

この警戒レベルに基づいた対応フレームワークは、公民館が災害時における情報伝達と資源配分において、効率的かつ有効に機能するための基盤を提供する。また、公民館が持つ地域に密着した特性を活かし、地域住民が直面するリスクに対して、より具体的で適切な対応を促す。

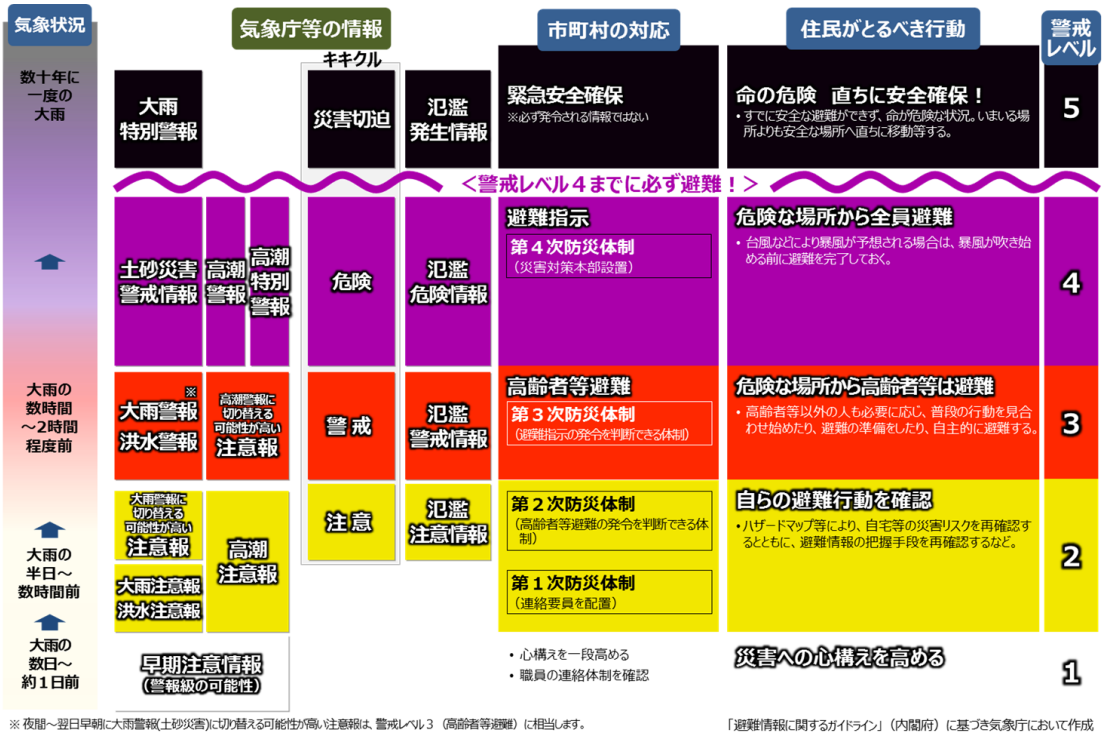

図 2.3: 段階的に発表される防災気象情報と対応する行動

2.3.2　公民館の避難所利用

非常時、特に自然災害や緊急事態が発生した際、公民館は地域コミュニティにとって極めて重要な役割を果たす。これら施設は、避難所としての機能を持ち、地域住民に安全と避難の場を提供する [4]。

- **避難所としての機能：**自然災害時、公民館は避難所として機能し、住民に安全な避難場所を提供する。これには、食料、水、医療用品、暖房などの基本的な生活支援が含まれる。避難所としての公民館は、コミュニティのメンバーが集まり、情報を共有し、相互支援を行う中心地となる。
- **情報提供とコミュニケーションの中心：**災害発生時、公民館は重要な情報の配布点となり、住民に災害関連のアップデート、安全な避難路の指示、救援活動に関する情報を提供する。地域住民間のコミュニケーションを促進し、緊急事態における協力と支援の精神を強化する。
- **地域社会の安定化に貢献：**災害後の混乱やパニックを防ぎ、地域社会の安定化に寄与する。公民館の運営スタッフやボランティアは、住民の不安を和らげ、必要なサポートを提供する。長期間の避難が必要な場合、公民館は一時的な居住地として機能し、地域コミュニティの持続的な支援を提供する。

- **復旧・復興活動の基盤：**災害後の復旧・復興活動において、公民館はボランティアや支援団体の拠点となることが多い。地域のニーズに応じた復旧活動の計画や実施において、公民館は重要な役割を担う。

災害後の復旧・復興活動において、公民館はボランティアや支援団体の拠点となることが多い。地域のニーズに応じた復旧活動の計画や実施において、公民館は重要な役割を担う。このように、非常時における公民館の役割は、単に物理的な避難所を提供することに留まらず、地域コミュニティの安全、情報共有、精神的支援、そして復旧・復興の中心として機能する。そのため、公民館向けの建物 OS は、非常時に迅速かつ効果的に機能を切り替える能力を備えている必要がある。

第3章　既存の建物OSとその課題

3.1　建物OSの概要

清水建設によって開発された建物デジタルプラットフォーム「DX-Core(建物 OS)」は、建物内の建築設備、IoT デバイス、各種アプリケーションの相互連携を容易にする基本ソフトウェアである。インターネットとクラウドの普及、IoT・AI 技術の進展に伴い、建築設備システムの見直しが必要となった背景から開発された。このシステムは、従来の建築設備の枠を超えた新サービス・ソリューションの実現を目指し、施設・街区でのデジタルトランスフォーメーション (DX) 化を促進することを目的としている [8]。技術的特徴として、以下の点が挙げられる。まず、API

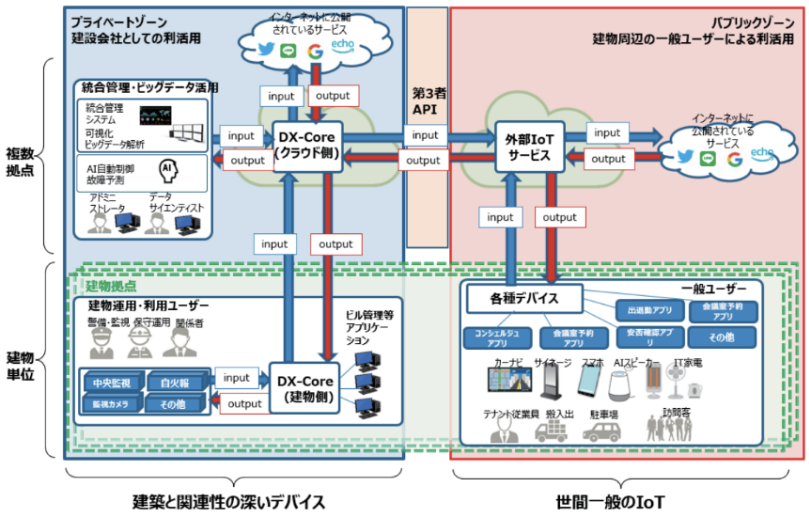

図 3.1: DX-Core の全体構成

接続とルール設定により、建築設備システムや IoT システム、インターネット上のソフトウェアを統一的に扱うことが可能である。次に、データプラットフォーム機能により、サブシステム間のデータ連携を容易にし、複数のサブシステムにまたがるデータの収集・連携・活用を実現する。図 3.2 [8] は DX-Core とサブシステムの関係を示す。共通マップ基盤によって、建築に関する情報を平面図・系統

図等のマップ形式で表示することができる。また、第三者 API 機能により、既存の IoT プラットフォームとの連携を容易にする。活用シナリオとしては、３密場

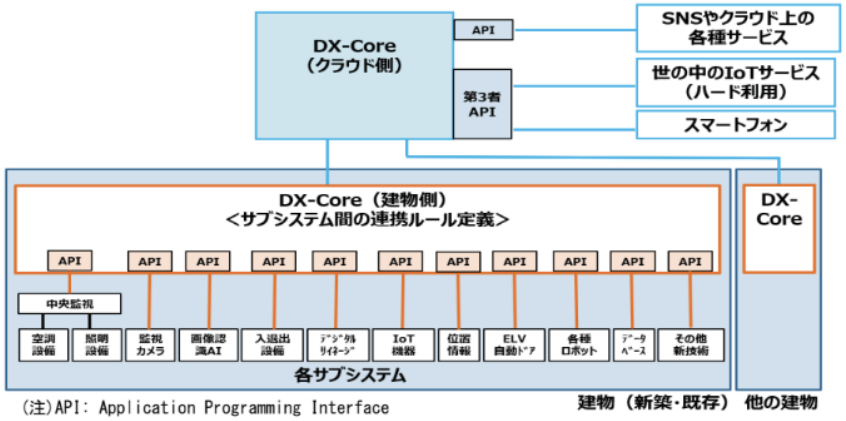

図 3.2: DX-Core とサブシステムの関係

所の換気強化システムや、館内利用者向けサービス（位置情報提供、空調・照明操作、館内情報・周辺施設情報表示など）、建物運用者向けサービス（ロボットによる館内配送・案内、人流計測機能、エリア侵入検知・エリア滞留検知機能など）が考えられる。

今後の展開としては、様々な施設・街区サービスへの応用や、スマートシティへの活用が重要な課題となる。デジタルツイン化を通じて、より良い建物環境の構築を目指している。DX-Core は、建物デジタルプラットフォームとしての重要性が高まっており、施設運用のデジタル化、スマートシティの実現に向けた取り組みが進められている。

3.2　現在の建物 OS の限界

現代の建物 OS は、建築設備の統合管理やデータ連携の効率化など多くの進歩をもたらしているが、依然としていくつかの重要な限界が存在する。これらの限界は、特に公共施設や多機能施設の運用において顕著になる。

1. **特定施設向けの設計：**
   多くの建物 OS は、商業ビルやオフィスビルのような特定の種類の施設向けに設計されており、公共施設や地域コミュニティ施設のような多目的・多機能施設のニーズを十分に満たしていない。

2. **柔軟性と拡張性の欠如：**
   システムの柔軟性や拡張性が不足している場合があり、新しい技術やサービスの導入が困難であったり、特定のベンダーの製品に依存することが多い。
3. **災害時の対応不足：**
   非常時、特に災害時の対応に特化した機能が不足していることが多く、緊急避難所として機能する公民館のような施設にとって、迅速な運用モードの切り替えや資源の最適化が困難である。
4. **ユーザーインターフェイスの問題：**
   一部の建物 OS は、直感的でないユーザーインターフェイスを持っており、非専門家が操作する際に難易度が高い。
5. **サステナビリティの観点からの不足：**
   エネルギー効率や環境負荷の観点での最適化が不十分な場合があり、持続可能な運用に対する要求に応えることが困難である。
6. **データセキュリティとプライバシーの課題：**
   多くの建物 OS では、データのセキュリティやプライバシーに対する対策が不十分であり、特に個人情報を含むデータの取り扱いにおいて懸念が存在する。

これらの限界は、特に災害時の緊急運用や多機能施設の日常運用において、効率的かつ効果的な建物管理を実現する上での障壁となっている。そのため、これらの限界を克服し、より柔軟で拡張性が高く、非常時にも対応可能な建物 OS の開発が求められている。

3.3 公民館向けの特有な要求

公民館向けの建物 OS には、特有の要求が存在する。これらの要求は、公民館の日常運用と非常時の対応、さまざまなデバイスへの対応、および高齢者や障害者への配慮を含む。以下に、これらの要求について詳述する。

平常時と非常時のモード切り替え:

公民館の建物 OS は、平常時と非常時の状況に応じて異なる動作モードを持つ必要がある。平常時には、日常的な運用と利便性を重視するが、非常時には人命を守ることが最優先となる。たとえば、地震や火災などの緊急事態が発生した際には、照明、案内表示、避難経路の案内などが自動的に最適化され、迅速な避難を支援する。

異なるデバイスへの対応:

公民館内には多種多様なデバイスが存在し、それらすべてを統合的に管理する

必要がある。これには、空調システム、照明、セキュリティシステム、情報表示システムなどが含まれる。これらのデバイスは異なるメーカーや技術基準に基づいている可能性があるため、建物 OS はこれらのデバイス間の互換性と統合を確保する柔軟性を持つ必要がある。

高齢者や障害者への対応:

公民館は、高齢者や障害者を含む幅広い利用者にサービスを提供するため、これらのグループに特化した機能が求められる。例えば、視覚や聴覚に障害のある人々のための特別な案内システムや、車椅子利用者のための自動ドア、エレベーターの操作支援などが考えられる。また、高齢者向けには、簡単な操作インターフェースや緊急時のアシスト機能が重要となる。

第4章　Multi-Mode Kominkan Operating Systemの提案

4.1　スマートビルシステムの概要

スマートビルは、デジタル技術とデータの活用により、従来の建物管理と運営を効率化し、快適性向上、省エネルギー化、持続可能性の実現に寄与する新概念の建物として位置づけられる。これらのビルでは、オペレーティングシステムを介してハードウェアとソフトウェアが分離され、設備と機能がデカップリングされる。ソフトウェア化された機能群の連携により、目標達成が図られる [9]。スマートビルの実現には、IT/IoT 技術、AI 技術、エネルギー管理技術など多岐にわたる技術の適用が必要である。これにより、新しい産業構造が形成され、多様な分野の人材や企業が交流し、ビル業界のアンバンドリング・リバンドリングが活性化し、新市場の創出が期待される。

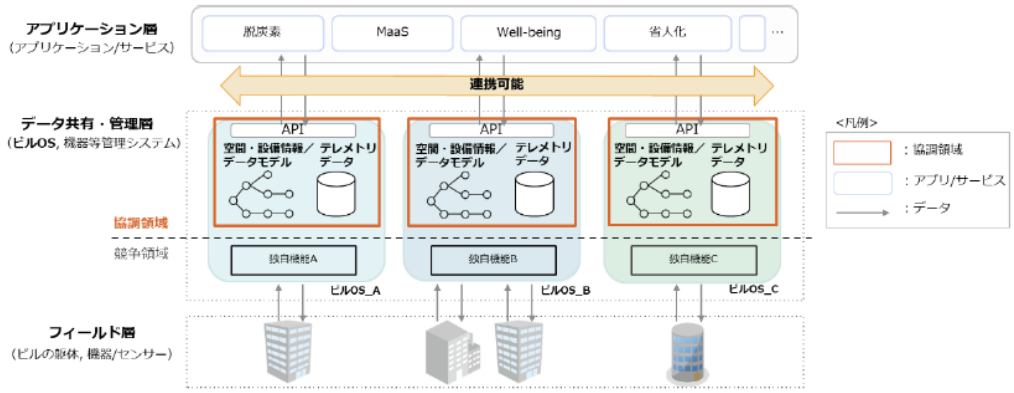

図 4.1: スマートビル 3 階層の関係性 1

スマートビルシステムアーキテクチャは、相互運用性と拡張性の確保を重視し、従来のビルとは異なるアプローチを採用している。このアーキテクチャは、建物とアプリケーションからなる二元的なシステムではなく、データ連携を保証する機能が介在する三層構造を基盤としている。図 4.1 [9] はスマートビル 3 階層の関

係性を示す。具体的には、アプリケーション層、データ共有・管理層、フィールド層の三つの階層から成り立っている。

アプリケーション層は、ビルと連携するアプリケーションやサービスが属する領域であり、特定のユーザーに対して価値を提供する役割を担う。この層では、主にデータ共有・管理層との間で建物関連データのやり取りが行われる。

データ共有・管理層は、フィールド層とアプリケーション層の間に位置し、データの流通を担う。この層の中核をなすビル OS は、アプリケーションやサービスによるデータの利活用を促進し、協調領域を設定している。この領域では、データモデルや通信インターフェースが標準化されており、複数のビル OS や建物をまたいだアプリケーションの展開が可能となる。

フィールド層は、ビルの躯体や照明、空調、計量等の設備サブシステム、中央監視システムなどが属する領域である。この層では、デバイスの制御指令に関する一部の仕様がデータ共有・管理層における協調領域として標準化されており、アプリケーションやサービスは共通の仕様でデバイスの制御を実行できる。[9]

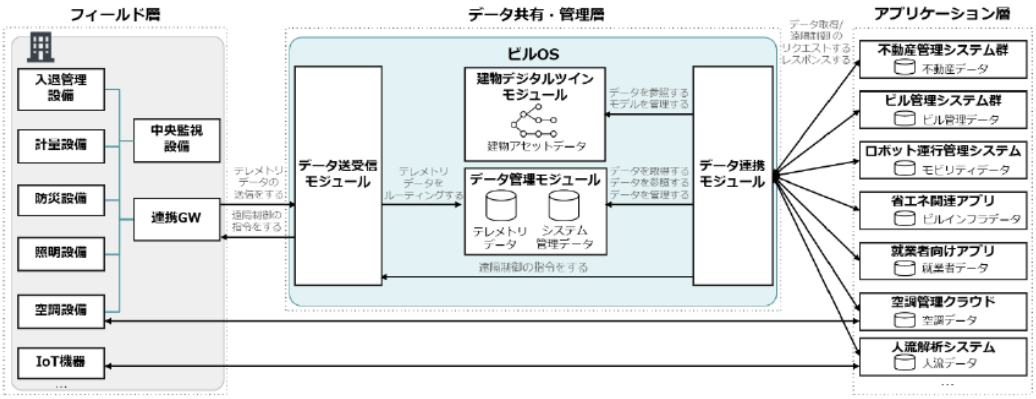

図 4.2: スマートビル 3 階層の関係性 2

このように、スマートビルシステムアーキテクチャは、各層間の疎結合な関係を保ちつつ、効率的かつ柔軟なデータ連携とシステム運用を実現するための設計がなされている。図 4.2 [9] はスマートビルのシステム構成を示す。データ共有・管理層が介在することで、アプリケーション層とフィールド層の間のデータやコマンドの流れがスムーズになり、システム全体の拡張性と適応性が向上している。また、ビル OS を中心とした協調領域の設定により、様々なビルやアプリケーション間でのデータの共有と利活用が容易になり、スマートビルの機能拡張や新たなサービスの開発が促進される。フィールド層では、ビルの物理的な設備やシステムがデータ共有・管理層を介して統合され、より効率的な運用と管理が可能となる。

この三層構造により、スマートビルは従来のビルディングマネジメントシステムを超えた、新たな価値創出とユーザーエクスペリエンスの向上を実現している。

公民館においても、スマートビルの概念を取り入れることで、地域コミュニティの中心としての機能を強化し、より効率的で快適な空間を提供することが可能となる。デジタル技術の活用により、公民館の運営やサービス提供が最適化され、地域住民のニーズに応じた多様な活動の支援が実現する。また、エネルギー効率の向上や持続可能な運営が可能となり、地域社会における環境負荷の軽減にも寄与する。公民館がスマートビルの概念を取り入れることは、地域コミュニティの活性化と持続可能な発展の両方を促進する重要なステップとなる。

4.1.1 Multi-Mode Kominkan Operating System の概要

MKOS（Multi-Mode Kominkan Operating System）は、スマートビルシステムアーキテクチャに基づいて設計された公民館専用の建物 OS である。このシステムは、平常時と非常時のモード切り替え機能を備え、公民館の運用効率と安全性を高めることを目的としている。[10]MKOS は以下の主要なモジュールから構成される。図 4.3 は設計したシステムの全体像を示す。

各モジュールの機能を以下にて記述する。

1. **データ共有解釈モジュール:** このモジュールは、システム内のデータを解釈し、他のモジュール間でのデータ共有を容易にする。データの意味や関連性を把握し、適切な処理や転送を行う。
2. **認証・認可モジュール:** システムへのアクセス制御を行い、セキュリティを保持する。ユーザーの認証と認可を管理し、不正アクセスを防ぐ。
3. **データ管理モジュール:** 公民館内のデータを管理し、整理、保存、更新する機能を提供する。
4. **平常時・非常時管理モジュール:** 平常時と非常時のモードを切り替え、各モードに応じた運用を実現する。非常時には迅速な対応と情報提供を行う。
5. **データ送受信モジュール:** システム内部のデータ共有と外部情報の共有管理を担う。内部ネットワークおよび外部ネットワークとのデータのやり取りを管理する。
6. **デバイス制御モジュール:** 公民館内の各種デバイスを制御する。デバイス情報データベースを通じて、各デバイスの状態や制御情報を管理し、必要に応じて適切な操作を実行する。

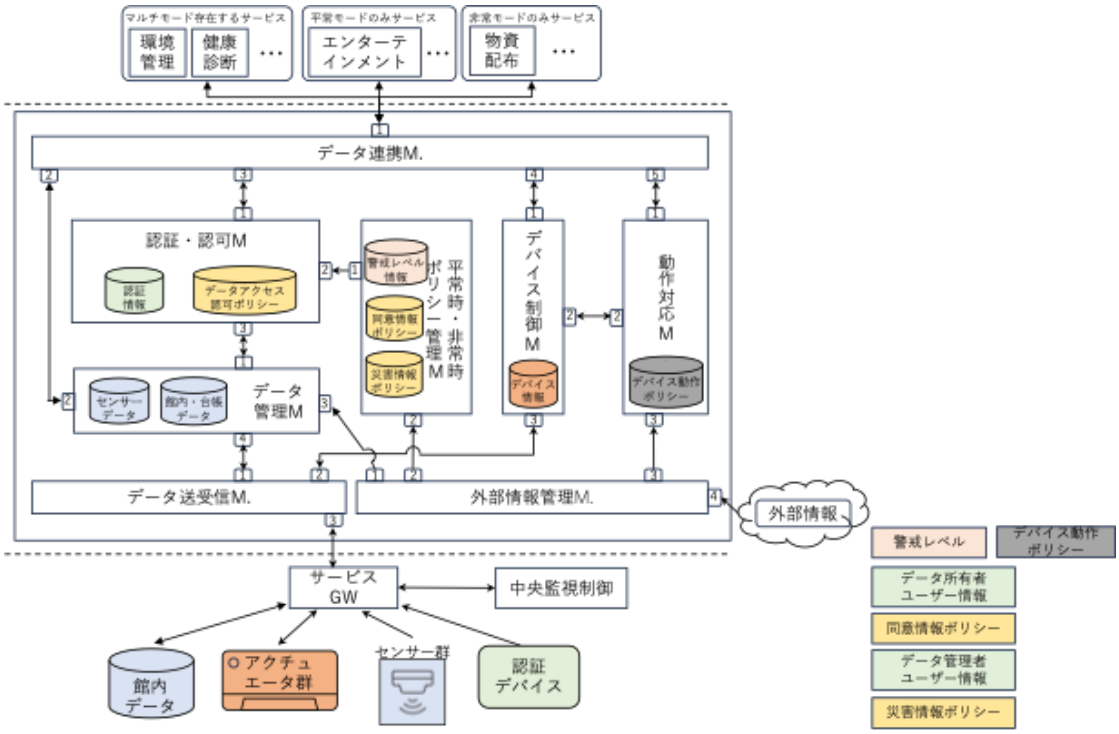

図 4.3: MKOS のシステム全体像

7. **動作対応モジュール:** デバイスの動作ポリシーを持つデータベースを含み、システム全体の動作を調整する。このモジュールは、特定の状況や要求に応じて、デバイスの動作を自動的に調整するポリシーを定義し、実行する。
8. **外部情報管理モジュール:** 外部からの情報（例えば気象情報や緊急通知）を受け取り、システム内での適切な反応を調整する。このモジュールは、外部環境の変化に対してシステムが適切に対応できるようにする。

MKOS の特徴は、平常時と非常時のモード切り替え機能にある。平常時には、公民館の日常的な運用に必要な機能を提供し、非常時には迅速な対応と情報提供を行うための優先度の高い命令を実行する。このモード切り替えは、公民館の安全性と効率性を大幅に向上させる。

図 4.4 の中で赤色点線示す部分は本文が設計したデータ共有解釈モジュール、デバイス制御モジュール、動作対応モジュールである。これらは、MKOS の中核を成す。これらのモジュールは、公民館内のデータとデバイスの効率的な管理と制御を可能にし、特に緊急時の迅速な対応を実現するために重要な役割を果たす。データ送受信モジュールと外部情報管理モジュールは、システム内外の情報の流れをスムーズにし、公民館の運用に必要な情報の収集と配布を効率的に行う。これにより、公民館の日常運用から緊急時の対応まで、幅広いシナリオに対応できる柔

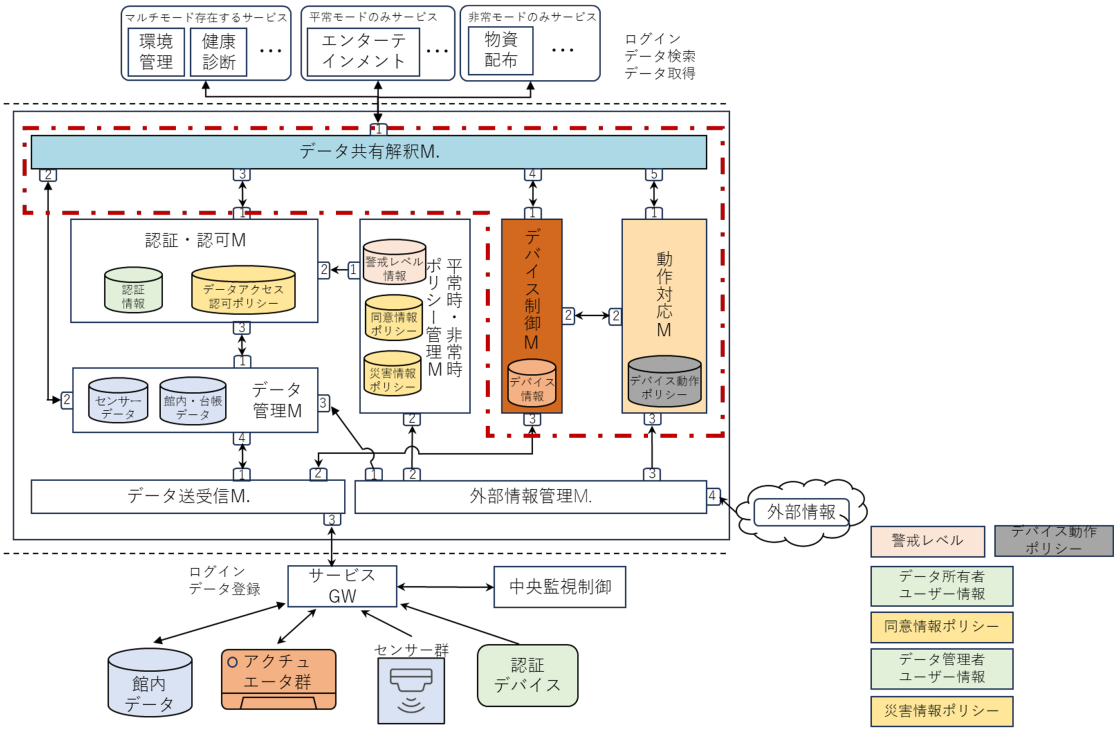

図 4.4: 本文で設計したモジュール

軟なシステム構成が実現されている。

MKOS は、これらのモジュールを統合することで、公民館の運用をスマートかつ安全に行うための包括的なソリューションを提供する。平常時では、公民館の日々の活動をサポートし、非常時には迅速かつ効果的な対応を可能にする。このシステムは、公民館の運用管理者にとって強力なツールとなり、利用者にとっては安全で快適な環境を提供する。また、MKOS の設計は、スマートビルシステムアーキテクチャの原則に基づいており、将来的な拡張やアップデートも容易に行える柔軟性を持っている。これにより、公民館の運用がより効率的かつ安全になることが期待される。

4.2　システムの主要コンポーネント

4.2.1　データ共有解釈モジュール

4.2.1.1　データ共有機能

モジュール間のデータ共有: データ共有解釈モジュールは、システム内の他モジュール（データ管理モジュール、認証・認可モジュール、平常時・非常時管理モ

ジュール、デバイス制御モジュール、動作対応モジュール）と接続し、これら間でデータの共有と交換を行う。これにより、システム全体の統合性と効率性が保たれ、各モジュールが必要な情報をタイムリーに取得可能。

サービスとのデータ共有: 公民館内部の環境データや警戒情報など、外部からの重要情報を受け取り、システム内の適切なモジュールへ転送する。これにより、サービスは受け取ったデータに基づいて適切な命令を出せる。

4.2.1.2 データ解釈機能

命令の解釈と変換: サービスからの命令をシステム内のモジュールが理解しやすい形に解釈し、具体的な設定目標に変換する。例えば、抽象的な命令を具体的な操作指示に変換し、デバイス制御モジュールへ送信する。これにより、システムはより効率的かつ正確に動作可能。図 4.5 で、抽象的な命令を具体的なデバイス操作にどのように変換するかを示す。

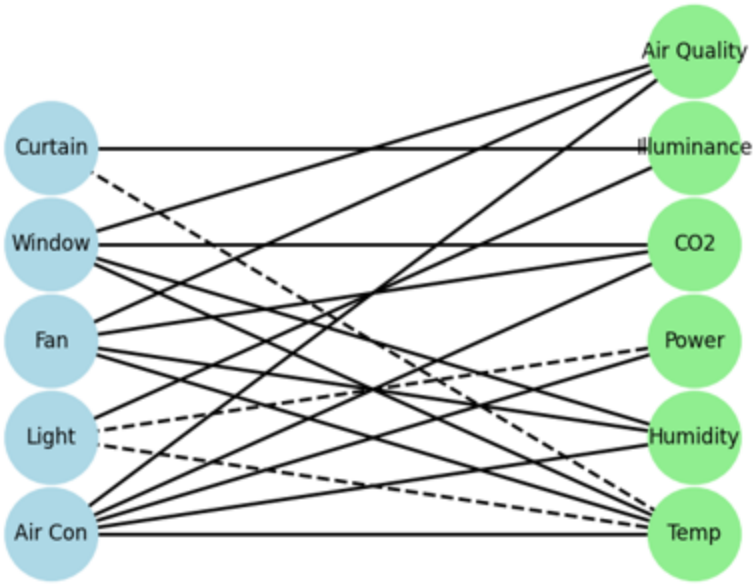

図 4.5: デバイスと環境影響の関係

エアコン: 温湿度、電力消費、CO2 濃度、空気質に影響を与える。
ライト: 照度に影響を与える。温度と消費電力に対する影響は点線で示される。
換気扇: 温湿度、空気質、二酸化炭素に影響を与える。
窓: 温湿度、空気質、二酸化炭素に影響を与える。

カーテン: 照度に影響を与える。温度に対する影響は点線で示される。

4.2.1.3　主要な役割

システム内部の情報流通の最適化: データ共有解釈モジュールは、システム内の情報流通を最適化し、各モジュール間でのデータの一貫性とアクセスの効率を保証する。これにより、システム全体の反応速度と処理能力が向上し、公民館の運用において迅速かつ適切な対応が可能になる。

外部情報との連携強化:

公民館外部からの情報（例えば地域の緊急情報や気象情報）を受け取り、内部システムに適切に統合する。これにより、公民館は外部環境の変化に迅速に対応し、利用者の安全と快適性を確保できる。

命令の具体化と実行の効率化:

抽象的な命令を具体的な操作指示に変換することで、デバイス制御モジュールや他の関連モジュールが効率的に動作できるようにする。これにより、公民館内の各種デバイスやシステムが適切に連携し、総合的な運用管理が実現される。

4.2.1.4　技術的特徴

高度なデータ処理能力:

複雑なデータセットを迅速に処理し、必要な情報を適切なモジュールに提供する能力を持つ。これにより、システム全体のデータ処理と応答時間が最適化される。

柔軟なデータ解釈アルゴリズム:

異なる形式や抽象度のデータを解釈し、具体的な操作指示に変換するための柔軟なアルゴリズムを備える。これにより、様々な種類の命令やデータを効率的に処理し、システム全体の適応性と汎用性を高める。

インターモジュール通信の最適化:

システム内の各モジュール間での通信を効率化し、データの整合性とタイムリーな情報共有を保証する。これにより、システム内の情報フローがスムーズになり、全体の運用効率が向上する。

外部情報統合の自動化:

外部からの情報を自動的に受け取り、解釈し、システム内の適切なモジュールに転送する機能を持つ。これにより、外部環境の変化に迅速に対応し、公民館の

安全性と機能性を維持する。

データ共有解釈モジュールは、MKOS の中核を成す重要なコンポーネントであり、公民館のスマート運用において不可欠な役割を果たす。このモジュールにより、公民館は日常運用から緊急時対応まで、様々なシナリオに柔軟かつ効率的に対応できるようになる。

4.2.2 デバイス制御モジュール

デバイス制御モジュールは、MKOS（Multi-Mode Kominkan Operating System）内で重要な役割を担うコンポーネントである。このモジュールの主要な機能とプロセスは以下の通り。

4.2.2.1 操作指示の受信と処理

動作対応モジュールとの連携:
デバイス制御モジュールは操作指示を受信すると、まず動作対応モジュールに対して操作デバイスの動作ポリシーをリクエストする。これにより、デバイスの動作に関する適切なポリシー情報を取得する。
ポリシーに基づく操作指示の検証:
受信した操作指示の値と動作ポリシーの値を比較し、操作指示がポリシーに準拠しているかを確認する。もし操作指示がポリシーに準拠していない場合、ポリシー中のデフォルト値に修正する。
デバイスへの命令出力:
ポリシーに基づいて修正された操作指示をデバイスへ送信し、実際のデバイス操作を行う。これにより、公民館内のデバイスが安全かつ効率的に制御される。

4.2.2.2 デバイス情報データベースの管理

デバイス情報の保持:
デバイス制御モジュールはデバイス情報データベースを持ち、公民館内の各デバイスに関する詳細情報（種類、状態、制御パラメータなど）を保持する。
データ共有解釈モジュールとの連携:
データ共有解釈モジュールは、デバイス情報データベースのデータを参照し、受け取った命令の解釈を行う。これにより、データ共有解釈モジュールは、デバイ

スの現状や能力に応じて、適切な命令を生成し、デバイス制御モジュールに送信する。

4.2.2.3　主要な役割

安全性の確保:
デバイス制御モジュールは、動作ポリシーに基づいてデバイスの操作を行うことで、公民館内のデバイス運用の安全性を確保する。不適切な操作指示はポリシーに基づいて修正され、デバイスの誤動作や安全上のリスクを最小限に抑える。

効率的なデバイス管理:
デバイス情報データベースを通じて、公民館内のデバイスの状態や能力を把握し、最適な制御を実現する。これにより、エネルギー効率の向上やデバイスの長寿命化に寄与する。

柔軟な制御対応: 状況に応じてデバイスの動作ポリシーを調整し、柔軟に制御対応を行う。これにより、公民館の様々なイベントや活動に対して適切な環境を提供する。

4.2.2.4　技術的特徴

高度な制御アルゴリズム:
複数のデバイスを同時に制御し、互いのデバイス間での干渉を避けるための高度な制御アルゴリズムを備える。これにより、複雑なシナリオでも安定したデバイス制御が可能になる。

リアルタイムデータ処理:
デバイスの状態変化や外部からの命令に迅速に対応するためのリアルタイムデータ処理能力を持つ。これにより、緊急時の迅速な対応や日常運用の効率化が実現される。

デバイス制御モジュールは、MKOS の中でデバイスの安全かつ効率的な運用を実現するための重要なコンポーネントであり、公民館のスマート運用において不可欠な役割を果たす。

4.2.3　動作対応モジュール

動作対応モジュールは、MKOS（Multi-Mode Kominkan Operating System）において、デバイスの動作ポリシーを管理し、適切な反応を調整するための重要なコンポーネントです。このモジュールの主要な機能とプロセスは以下の通りです。

4.2.3.1　デバイス動作ポリシーデータベースの管理

動作ポリシーの保存: 各デバイスの平常時および非常時の動作ポリシーをデータベースに保存し、管理する。これにより、各デバイスが特定のシナリオに応じて適切に動作することを保証する。

ポリシーの発行: デバイス制御モジュールから特定のデバイスの動作ポリシーのリクエストがある場合、適切な即時ポリシーを発行し、デバイス制御モジュールに提供する。

4.2.3.2　災害情報と警戒レベルの処理

外部情報の受け取り:

災害情報や警戒レベルなどの外部情報を外部情報管理モジュールを経由して受け取る。これにより、公民館の現在の状況をリアルタイムで把握し、適切な対応を行う。

非常時ポリシーの発行:

受け取った情報を基に、現在が平常時か非常時かを判断し、必要に応じて非常時動作ポリシーを発行する。これにより、非常時には迅速かつ適切なデバイスの動作が保証される。

サービスへの通知:

非常時情報をデータ共有解釈モジュールを経由してサービスに通知する。これにより、公民館内のサービスやシステムが非常時に適切に対応できるようになる。サービスはこの情報を基に、必要な措置を取ることができる。

4.2.3.3　主要な役割

適切なデバイス制御の確保:
各デバイスの動作ポリシーを管理することで、デバイスが状況に応じて適切に反応し、効率的かつ安全に動作することを保証する。

非常時対応の最適化:
災害情報や警戒レベルに基づいて、非常時の動作ポリシーを迅速に発行し、デバイス制御モジュールに指示を出す。これにより、非常時には公民館内のデバイスが迅速かつ適切に対応し、利用者の安全を確保する。

緊急情報の迅速な伝達:
非常時情報をデータ共有解釈モジュールを通じて他のシステムやサービスに迅速に通知する。これにより、公民館全体の緊急対応が一貫して行われ、状況に応じた迅速な措置が可能になる。

4.2.3.4　技術的特徴

柔軟なポリシー管理:
状況に応じて動作ポリシーを柔軟に調整し、変化する環境や要求に迅速に対応する。これにより、公民館の運用はより効率的かつ安全になる。

リアルタイム情報処理:
外部からの情報をリアルタイムで処理し、即座に適切な動作ポリシーを生成する。これにより、非常時の迅速な対応が可能となり、公民館の安全性が大幅に向上する。

動作対応モジュールは、MKOS の中でデバイスの適切な動作を保証し、非常時における迅速な対応を実現するための重要なコンポーネントである。このモジュールにより、公民館は日常運用から緊急時対応まで、様々なシナリオに柔軟かつ効率的に対応できるようになる。

4.3　平常時と非常時のモード切り替え

平常時と非常時のモード切り替えは、MKOS（Multi-Mode Kominkan Operating System）において重要な機能の一つである。このモード切り替えのプロセスは以下のように行われる。

4.3.1　モード切り替えのプロセス

1. 災害情報と警戒レベルの入力: 外部情報管理モジュールは、災害情報や警戒レベルをシステムに入力する。これにより、公民館の現在の状況がシステムに伝達される。

2. 非常時状態の判断: 入力された情報に基づいて、システムは非常時状態かどうかを判断する。この判断は、公民館の安全性と運用効率を保つために重要である。

3. 非常時動作ポリシーの発行: 非常時状態と判断された場合、動作対応モジュールは非常時の動作ポリシーを発行し、デバイス制御モジュールに送信する。これにより、公民館内のデバイスは非常時に適切に動作する。

4. サービスへの通知とモード変更: データ共有解釈モジュールは非常時状態をサービスに通知し、マルチモードを持つサービスを非常時モードに切り替える。これにより、非常時に適切な命令がシステムに発送される。

5. 命令の適用範囲: 平常時モードのみのサービスは非常時に命令を発送しないが、非常時モードのみのサービスは非常時に命令をシステムに発送する。

4.3.2　重要性と機能

迅速な対応: 非常時には、公民館内のシステムとデバイスが迅速に適切な対応モードに切り替わる。これにより、利用者の安全を確保し、緊急時の混乱を最小限に抑える。

柔軟な運用: 平常時と非常時のモード切り替えにより、公民館は様々な状況に柔軟に対応できる。これにより、日常の運用効率と非常時の安全対策の両方を最適化する。

情報の正確な伝達: 外部情報管理モジュールからの正確な情報入力により、システムは適切なタイミングでモードを切り替える。これにより、非常時の対応が適切に行われ、公民館の安全性が向上する。

サービスの適応性: マルチモードを持つサービスは、状況に応じて平常時モードと非常時モードを切り替えることができる。これにより、サービスは常に最適な状態で提供される。

平常時と非常時のモード切り替え機能は、MKOS の中で公民館の運用を柔軟かつ効率的に行うための重要な要素であり、公民館の安全性と機能性を大幅に向上させる。この機能により、公民館は日常運用から緊急事態に至るまで、様々なシナリオに適切に対応できるようになる。非常時には、迅速な情報伝達とデバイスの適切な制御により、利用者の安全を最優先に考慮した対応が可能となる。また、平常時には、日々の運用に必要な機能を維持しつつ、非常時に備えた準備を常に行うことができる。これにより、公民館はより安全で快適な環境を提供し、地域コミュニティの中心としての役割を果たすことができる。

第5章　システムの機能詳細

本章では、公民館向け建物 OS の機能詳細について説明する。本システムは、建物内の環境制御と資源最適化を目的として設計されており、複数のモジュールで構成されている。各モジュールは特定の機能を担い、相互に連携して全体のシステム動作を実現する。

5.1　システムの概観

図 5.1 は、システムの全体構造とモジュール間の連携を示す UML 図である。

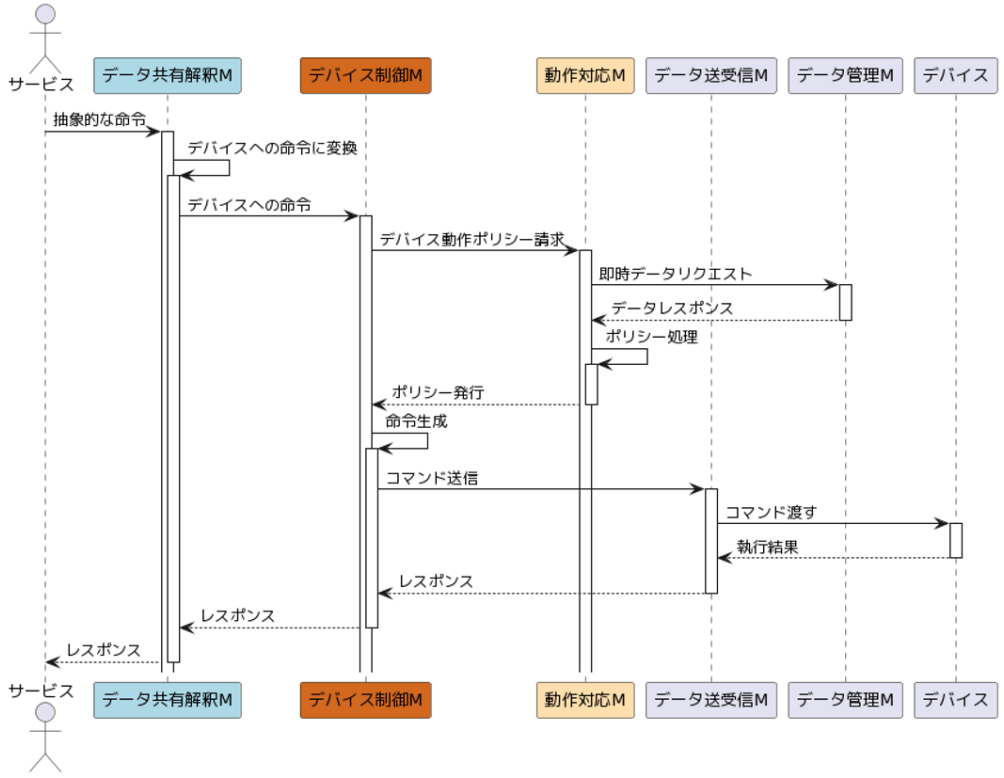

図 5.1: システムの UML 図

システムの主要な機能には、環境データの取得、抽象的な命令の解釈、動作ポ

リシーに基づくデバイス命令の生成と修正が含まれる。これらの機能は、データ共有解釈モジュール、デバイス制御モジュール、動作対応モジュールなど複数のモジュールによって実現されている。

システムの基本的な動作は、以下のように概説される。まず、データ共有解釈モジュールは、各サービスからの抽象的な命令を命令を受け取る。次に、このモジュールは受け取った命令を解析し、具体的な部屋の設定目標に変換する。変換された設定目標はデバイス制御モジュールに送られる。さらに、動作対応モジュールは、適切なデバイスを選択して、選択したデバイスの動作ポリシーを動作対応モジュールに請求する。動作対応モジュールは、現在の環境状況や平常時非常時の条件に基づいて、デバイスの動作ポリシーを発行する。デバイスモジュールは、動作ポリシーによる、設定目標の数値を修正して、具体的なデバイスへ命令を送る。このプロセスにより、公民館内の環境は常に最適化され、快適かつ安全な状態が維持される。

5.2　動作モードの切り替え

公民館は日常生活における多様なサービスの提供者として機能する一方で、非常時には地域住民の安全と支援の要となる重要な役割を担う。平常時は文化活動、教育プログラム、レクリエーションの場として活用されるが、非常時には避難所としての機能や緊急支援のコーディネートセンターへと変貌する。

この転換に対応するため、本システムでは動作対応モジュールが平常時と非常時のサービス優先度およびデバイスの設定値範囲を管理している。非常時には、例えば避難所としての機能が優先され、エンターテインメントや余暇活動関連のサービスは低優先度または停止する。動作対応モジュールは、外部からの入力信号を受けて、システム全体の動作モードを即座に切り替える。

切り替えの際には、データ共有解釈モジュールとデバイス制御モジュールに対して、現在のサービス優先度とデバイスの設定値範囲が送信される。これにより、各サービスは優先度に基づいて適切に動作し、デバイスは指定された範囲内でのみ操作される。例えば、非常時には照明は常に点灯し、避難者の安全を確保する一方で、空調はエネルギー使用を最小限に抑えるために最低限の範囲で運用される。

本システムの動作モード切り替え機能により、公民館は災害発生時に迅速かつ効果的に地域住民の安全を確保し、リソースの最適化を図ることが可能となる。これは、災害に対するレジリエンスを高め、地域コミュニティの持続可能性を支えるために不可欠である。

5.3　抽象的な命令の処理

5.3.1　抽象度に基づく命令の定義と特徴

システムにおける命令の抽象度は、直接的なデバイス制御から、より複雑なユーザーの好みや外部環境に基づく制御まで、異なるレベルを含む。表 5.1 には抽象度レベルは、低、中、高の三つに分類され、それぞれが異なるデータセットと利用シナリオを要求する。低抽象度の命令は、最も基本的なレベルで、ユーザーが特定のデバイスに対して具体的な操作命令を指定する。デバイスの技術的仕様や操作パラメーターが直接的に命令に反映される。中抽象度の命令は、外部の基準やデータを参照してデバイスの操作を決定するレベルである。平年の気候データや統計情報を基に、システムが自動的に最適な操作値を算出し提案する。高抽象度の命令は、個人の好みや過去の行動パターンを蓄積し、それを基に最適なデバイス制御を実現する最も高度なレベルである。ユーザーの居住環境やライフスタイルに最も適した環境設定を提供する。個々のユーザーに対するカスタマイズされた快適環境の実現を目指す。

	低抽象度	中抽象度	高抽象度
定義	直接的なデバイス制御パラメーターを指定する API レベル。	平年の気象データなどの外部基準を前提としてパラメーターを指定する API レベル。	個々のユーザーの好みや過去の行動データを蓄積し、それに基づいてパラメーターを自動調整する API レベル。
特徴	ユーザーが指定したパラメータに基づく直接的な制御。	平均的なデータや規範に基づいた自動化された制御。	学習した個人の好みや行動パターンに基づいた自動化された最適化。
具体的な例	「エアコンを 22 度に設定する」	「9 月の平均気温に基づいてエアコンを設定する」	「ユーザーが通常好む温度に基づいてエアコンを自動設定する」
必要なデータ	特定のデバイスやシステムの技術的仕様や制御パラメーター。	気象庁などの公的機関から提供される気候データや統計データ。	ユーザーの過去の設定データ、個人の好み、居住パターン。
利用シナリオ	ユーザーが明確な制御値を知っている場合や、特定のデバイスを特定の状態に設定する必要がある場合。	季節に応じた室温調整、平年の気温に基づいたエネルギー管理など。	個人の快適温度への自動調整、利用者の生活リズムに合わせた照明制御など。

表 5.1: 抽象度レベルごとの定義、特徴、具体的な例、必要なデータ、および利用シナリオ

5.3.2 命令の受信 API と処理

データ共有解釈モジュールは、抽象度の高い命令を具体的な設定目標に変換する役割を果たす。この機能は、低抽象度から高抽象度までの命令を受け取り、それに応じた適切なデバイス制御命令を生成するために設計されている。API では、POST メソッドを使用して、様々な抽象度の命令を受信する。以下に抽象的な命令の受信 API のコード例を示す。

```
[titleコード抽象度の命令を受信=5.1:API, frame=shadowbox]
paths:
  /command/receive:
    post:
      summary: Receive an abstract command
      requestBody:
        content:
          application/json:
            schema:
              oneOf:
                - $ref: '#/components/schemas/LowAbstractionCommand'
                - $ref: '#/components/schemas/
                    MediumAbstractionCommand'
                - $ref: '#/components/schemas/HighAbstractionCommand'
      responses:
        '200':
          description: Command processed successfully
```

抽象的な命令は、具体的なデバイス操作に変換される必要がある。この変換プロセスは、命令の抽象度に応じて異なるアプローチを採用する。

低抽象度の命令では、具体的なデバイスの設定値や操作が直接指定される。例えば、「エアコンを 22 度に設定」という命令が受け取られた場合、この命令は直接デバイス制御モジュールに転送される。以下に低抽象度命令の処理コードを示す。

コード 5.2: 低抽象度命令の処理

```
function processLowAbstractionCommand(command)
    deviceId := command.deviceId
    action := command.action
    value := command.value

    if action is "set_temperature"
        sendCommandToDevice(deviceId, "setTemperature", value)
    else if action is "turn_on"
        sendCommandToDevice(deviceId, "turnOn")
    // 他のアクションに対する処理
end function
```

中抽象度の命令ては、外部の気象データを利用して具体的なデバイス制御パラメーターを決定する必要がある。この目的のために、気象庁防災情報 XML フォーマット形式の電文を使用し、PULL 型 Atom フィードから最新の気象情報を取得する。取得したデータは、中抽象度の命令に必要な情報、例えば特定の日付の気温や湿度など、実行すべきデバイス操作に直接関連するパラメーターを提供する。システムはこのデータを解析し、過去のデータや現在の気象条件に基づいて、エアコンや暖房などのデバイスに対する最適な設定値を導き出す。

コード 5.3: 気象庁のデータの獲得

```
funtion fetch_jma_atom_feed(url):
  url = "https://www.data.jma.go.jp/developer/xml/feed/extra.xml"
        response = requests.get(url)
        response.raise_for_status()
  // フィードを取得Atom

        root = ET.fromstring(response.content)
        entries = root.findall('entry')
  //レスポンスのを解析XML

        data_list = []
        for entry in entries:
            title = entry.find('{title').text
            updated = entry.find('updated').text
            content = entry.find('content')

            data = {
                'title': title,
                'updated': updated,
                'content': content.text if content is not None else
                    None
            }
            data_list.append(data)
        return data_list
end funtion
```

コード 5.4: 中抽象度命令の処理

```
function processMediumAbstractionCommand(command)
    roomCondition := command.environmentCondition
    roomId := command.room

    if roomCondition is "cool"
        currentTemperature := getCurrentRoomTemperature(roomId)
        if currentTemperature > comfortTemperature
            sendCommandToDevice("aircon", "setTemperature",
                comfortTemperature)
    else if roomCondition is "bright"
        // 照度に関する処理
    // 他の環境条件に対する処理
end function
```

高抽象度の命令では、非常に一般的な希望や感覚が表現される。例えば、「快適な環境にする」という命令は、システムが部屋の温度、湿度、照度などを総合的に考慮し、適切なデバイス設定を決定する。以下に高抽象度命令の処理コードを示す。

コード 5.5: 高抽象度命令の処理

```
function processHighAbstractionCommand(command)
    preference := command.preference
    roomId := command.room

    if preference is "comfortable"
        adjustRoomToComfortableSettings(roomId)
    else if preference is "relaxing"
        adjustRoomToRelaxingSettings(roomId)
    // 他の感覚的希望に対する処理
end function
```

5.4 部屋内の環境データとデバイス情報の提供

部屋内のデバイス情報は、サービスが効果的に環境制御を行うために重要である。データ共有解釈モジュールは、部屋の現在の環境状態や操作可能なプロパティに関する情報を提供する。以下に部屋内の環境データとデバイス情報の提供 API を示す。

コード 5.4: 部屋内の環境データとデバイス情報の提供 API

```
paths:
  /room/data:
    get:
      summary: Provide room environment data and device information
      parameters:
        - name: roomId
          in: query
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Room data retrieved successfully
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/RoomData'
```

- 部屋の環境情報（温度、湿度、照度など）と館内のスケジューリング情報は、サービスが現在の状況を理解し、適切な命令を発行するために不可欠である。
- 操作可能なプロパティに関する情報は、サービスが部屋内のデバイスをどのように制御できるかを理解するために提供される。これには、デバイスの種類、制御可能な範囲、およびデバイスの現在の状態が含まれる。

これらの機能により、データ共有解釈モジュールは、サービスが抽象的な命令を具体的なデバイス操作に変換し、効果的な環境管理を実現するための重要な役割を果たす。また、このモジュールは、部屋の状態を継続的に監視し、サービスに最新の情報を提供することで、システム全体の応答性と効率性を向上させる。

5.5　動作ポリシーに基づくデバイス命令の修正

動作対応モジジュール内には、各デバイスに対する動作ポリシーデータベースが存在し、平常時と非常時における各デバイスの設定値情報が格納されている。これらの設定値は具体的な数値ではなく、最大値（MAX）と最小値（MIN）の範囲で定義されている。デバイス制御モジュールは、受信した設定値を動作ポリシーの範囲と比較することで、適切なデバイス制御命令を生成する。

5.5.1　API

動作ポリシーに基づくデバイス命令の修正機能は、デバイス制御モジュールが生成する命令を、動作ポリシーに従って最適化するために設計されている。この機能には、動作ポリシーの照会、追加、削除を行うための API が含まれる。

動作ポリシー照会 API：

特定のデバイスに関連する動作ポリシーを取得するために使用される。これにより、デバイス制御モジュールは、現在の環境条件やユーザーの要求に基づいて、適切な設定値を選択することができる。以下に動作ポリシー照会 API を示す。

コード 5.5: 動作ポリシー照会 API

```
paths:
  /policy/query:
    get:
      summary: Query device operation policies
      parameters:
        - name: deviceId
          in: query
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Device operation policy retrieved successfully
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/DevicePolicy'
```

動作ポリシー追加 API:

動作ポリシー追加 API は、新しいデバイス動作ポリシーをシステムに追加する

ために使用される。この API を通じて、新しい種類のデバイスや特殊な運用条件に対応するポリシーを容易に追加できる。以下に動作ポリシー追加 API を示す。

コード 5.6: 動作ポリシー追加 API

```
paths:
  /policy/add:
    post:
      summary: Add a new device operation policy
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/DevicePolicy'
      responses:
        '200':
          description: Device operation policy added successfully
```

動作ポリシー削除 API:

動作ポリシー削除 API は、不要になったポリシーをシステムから削除するために使用される。これにより、デバイス制御モジュールは常に最新かつ関連性の高いポリシーに基づいて動作する。以下に動作ポリシー削除 API を示す。

コード 5.7: 動作ポリシー削除 API

```
paths:
  /policy/delete:
    post:
      summary: Delete a device operation policy
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                deviceId:
                  type: string
      responses:
        '200':
          description: Device operation policy deleted successfully
```

5.5.2 命令修正の動作

受信した設定値が動作ポリシーの範囲内にある場合、その設定値に基づくデバイス制御命令が生成され、対象のデバイスに送信される。一方、受信した設定値が現在の動作ポリシーの範囲外にある場合、範囲内で最も近い値に設定値を修正し、その修正された設定値を使用してデバイス制御命令が生成される。以下に命令修正の流れを説明する。

1. **使用可能なデバイスの解析:** デバイス制御モジュールは、最初に受信した命令を解析し、操作が可能なデバイスを特定する。この解析プロセスにより、命令が影響を与えるデバイスが明確になる。

2. **デバイスの動作ポリシー請求:** 次に、デバイス制御モジュールは動作対応モジュールに対して、特定されたデバイスの動作ポリシーを請求する。この請求は、デバイスが適切に動作するための制約条件を把握するために必要である。

3. **デバイスの動作ポリシー照会:** 動作対応モジュールは、動作ポリシーデータベースにアクセスし、特定のデバイスに関連する動作ポリシーを照会する。このポリシーには、デバイス操作の最大値（MAX）および最小値（MIN）の範囲が含まれている。

4. **命令の範囲判定と修正:** データベースからの応答に基づき、以下の処理が行われます。

   - **範囲内の場合:** 受信した設定値がポリシーの範囲内にある場合、ポリシーに従ってデバイス制御命令が生成され、デバイスに送信さる。
   - **範囲外の場合:** 受信した設定値がポリシーの範囲外にある場合、ポリシーの範囲内で最も近い値に設定値が修正され、修正された値に基づいてデバイス制御命令が生成される。

図 5.2 のプロセスにより、システムは常に最適な環境条件を維持しつつ、設定されたポリシーの範囲内でデバイスを制御する。例えば、非常時にはエネルギー消費を抑えるための設定が優先される場合があり、このような状況下でデバイス制御モジュールはエネルギー効率を最大化する設定値に基づいて命令を生成する。

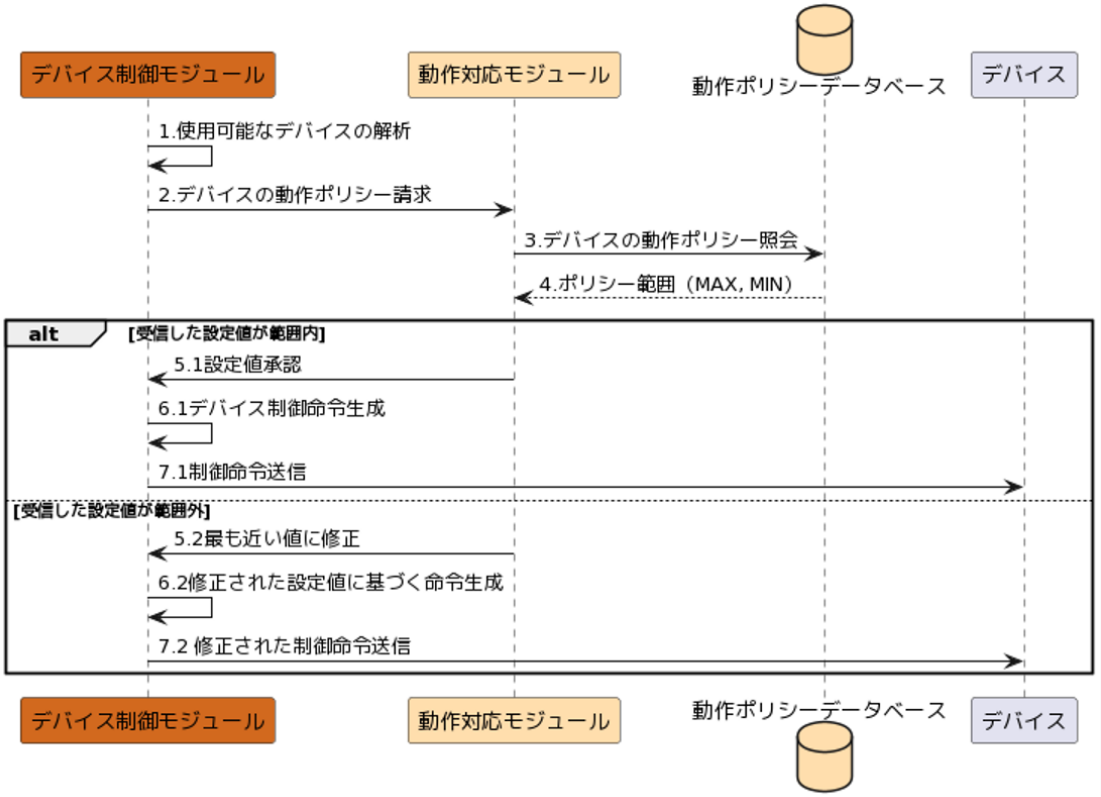

図 5.2: 命令修正のデータフロー

この機能により、システムは柔軟に異なるシナリオに対応し、常に最も適切なデバイス制御を実現する。これによって、公民館のエネルギー効率の向上、快適性の保持、および緊急時の安全性の確保が可能となる。

5.6 サービス間の命令衝突回避

命令衝突のシナリオ

公民館における複数のサービスが同時に作動している状況では、異なる命令が室内の環境属性に対して影響を与える可能性がある。例えば、サービス A がエアコンを 22 度に設定する命令と、サービス B が同時に窓を開ける命令を出す場合、この二つの命令はエネルギー効率とセキュリティの観点から衝突する。

優先度に基づく命令の処理

データ共有解釈モジュールは各サービスに優先度を割り当てる機能を有している。平常時と非常時で異なる優先度を持つことが可能であり、動作対応モジュー

ルから提供される優先度情報に基づいて命令を処理する。これにより、複数の命令が発生した場合には、優先度の高い命令が選択され、実行される。

衝突回避のプロセス

命令衝突が発生した場合、データ共有解釈モジュールは受信した命令の優先度を比較する [11]。この比較に基づき、最も優先度が高い命令が選ばれる。選ばれた命令はデバイス制御モジュールへと送信され、実行される。これにより、命令衝突を効率的に回避し、公民館内のシステムが最適な状態で動作することを保証する。図 5.3 にサービス間の命令衝突回避の処理フローを示す。

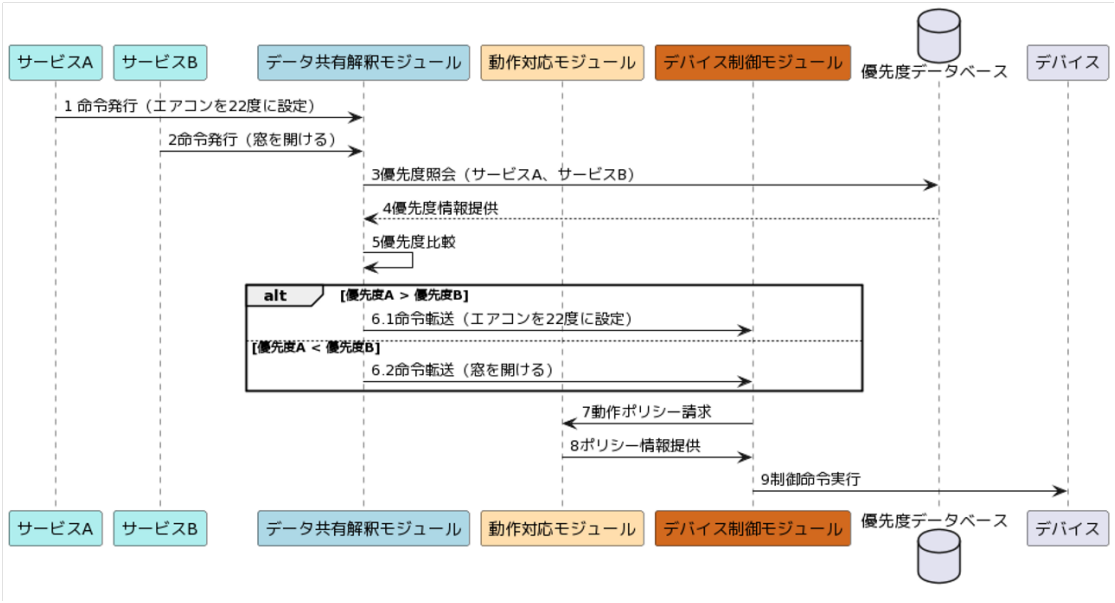

図 5.3: サービス間の命令衝突回避

このプロセスは、公民館の運用において重要な役割を果たす。複数のサービスが同時に動作している状況で、効率的かつ効果的なデバイス制御を実現するための重要な機能であり、公民館が地域コミュニティの安全かつ快適な場として機能するために不可欠である。

第6章　実装実験

第 4 章にて、多様な利用シナリオをサポートするスマートビルのための Multi-Mode Kominkan Operating System（MKOS）を提案した。本章では、当該システムの設計概念を具体化し、提案された MKOS の各機能についての実装実験を通じてその有効性と実用性を検証する。

6.1　実験対象

本研究では、Multi-Mode Kominkan Operating System（MKOS）の一部として、データ共有解釈モジュール、デバイス制御モジュール、動作対応モジュールの連携に焦点を当てた実験を行う。図 6.1 は MKOS の動作例を示し、クラウドに存在するサービスから MKOS に命令が送信された後のデータ処理の流れを表している。

抽象度	命令の内容
高抽象度	「快適な室内環境を保つ」
中抽象度	「特定日の気温設定を参照し、調整する」
低抽象度	「照明を点灯する」、「空調を 22 度に設定する」

表 6.1: サービスからの命令の例

表 6.1 に示されたように、サービスからの命令は抽象度によって高、中、低の 3 つに分類される。高抽象度命令は複雑な要求や一般的な指示を含み、中抽象度命令は特定の条件やパラメータを伴う要求を含み、低抽象度命令は具体的なデバイス操作指示を含む。

入力	出力
サービスからの命令 請求されたサービスの優先度	解析されたデバイス制御命令 サービスの優先度

表 6.2: データ共有解釈モジュールの入出力

データ共有解釈モジュールの入出力は表6.2に示されており、入力としてサービスからの命令と請求されたサービスの優先度が含まれ、出力としてサービスの優先度とデバイス制御命令が含まれる。

入力	出力
デバイス制御命令 即時の設定値範囲（動作ポリシー）	操作するデバイスの設定値範囲を請求 デバイスへの操作命令

表 6.3: デバイス制御モジュールの入出力

デバイス制御モジュールの入出力は表6.3に示されており、入力としてデバイス制御命令と即時の設定値範囲（動作ポリシー）が含まれ、出力として操作するデバイスの即時設定値範囲を請求すると共に、デバイスへの操作命令が含まれる。

動作対応モジュールの入出力は表6.4に示されており、入力として動作モード切り替え請求、操作設定値範囲の請求、サービス優先順位請求が含まれ、出力としてサービス優先度情報とデバイス設定値範囲が含まれる。

以上の内容は、本章の主要な構成要素であり、各モジュールの機能と相互作用を詳細に説明する。本研究では、これらのモジュールがどのように連携して効率的なデータ処理を実現するかを探求し、MKOSの性能と有用性を評価する。本章ではシステムの評価に関する詳細は述べないが、実験の設計と結果の概要を提示する。

6.2 実験概要

6.2.1 構築したシステム全体図

図6.1は、本実験で使用したシステムの全体図を示す。本システムは、Multi-Mode Kominkan Operating System（MKOS）の動作を模擬するために、複数のモジュールで構築さされた。実験環境は、４台のラズベリパイと１台のレイヤ２スイッチを使用して構成されている。各ラズベリパイは、MKOSの異なるモジュールを実行するために設定され、これらはレイヤ２スイッチを介して相互に通信する。プログラミング言語としてはPythonが選択され、その汎用性と豊富なライブラリにより、複雑なネットワーク通信やデータ処理が容易に行える。このシステム構成により、MKOSの各モジュール間の通信やデータの流れ、さらには全体の動作プロセスを効果的に模擬することが可能となる。

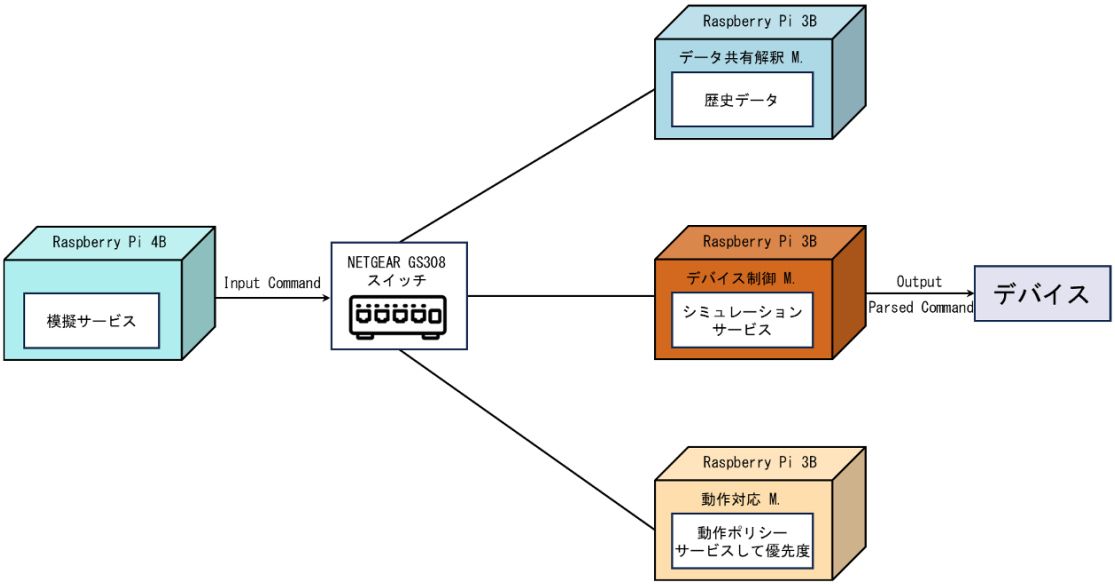

図 6.1: 本実験で使用したシステムの全体図

6.2.2 実験環境

次に実験を行なった環境について述べる。以下の表 6.4, 6.5, 6.6 の通りとなっている。

デバイス	CPU	メモリ	数量
Raspberry Pi 4B	ARM Cortex-A72	4GB	1 台
Raspberry Pi 3B	ARM Cortex-A53	1GB	3 台
Netgear GS308	-	-	1 台

表 6.4: 実験環境

デバイス	OS バージョン
Raspberry Pi 4B	Raspberry OS 64bit (Debian 12)
Raspberry Pi 3B	Raspberry OS 32bit (Raspbian 11)

表 6.5: OS のバージョン

言語/ライブラリ	バージョン
Python	3.11.2
Pandas	1.5.3
Requests	2.28.1
Flask	1.1.2
Openpyxl	3.0.9

表 6.6: Python とライブラリ

6.2.3 システムの設定と連携方法

本実験では、Multi-Mode Kominkan Operating System（MKOS）の各モジュールを表 6.7 のように Raspberry Pi に割り当て、それぞれのモジュール間で HTTP を通じて JSON メッセージを伝送する形式で連携を行った。

ラズパイ番号	デバイス	機能	IP アドレス
1	Raspberry Pi 4B	模擬サービス	192.168.100.10
2	Raspberry Pi 3B	データ共有解釈モジュール	192.168.100.100
3	Raspberry Pi 3B	デバイス制御モジュール	192.168.100.101
4	Raspberry Pi 3B	動作対応モジュール	192.168.100.102

表 6.7: システム構成

各モジュールが有線接続を通じてネットワークに接続され、指定された IP アドレスを介して互いに通信を行う。模擬サービス（Raspberry Pi 4B）は、事前に定義された命令を含む Excel ファイルから命令を読み取り、これをシステムに送信する役割を担う。表 6.8,6.9 は、Excel ファイル内に保存される模擬用の命令例を示している。これらはサービスからの命令としてシステムで模擬される。

抽象度	命令例
低抽象度	カーテン閉じる
中抽象度	石川県能美市の 9 月 27 日で、多少涼しめに運転
高抽象度	普段の生活よりも暖かく、少し汗ばむ程度に運転

表 6.8: 抽象的な命令例

抽象度	JSON 形式の命令
低抽象度	{ ”device”: ”curtain”, ”setting”: 0 }
中抽象度	{ ”location”: ”Nomi, Ishikawa”, ”date”: ”2023-09-27”, p̈reference”: ”slightly cool” }
高抽象度	{ ”preference”: ”warmer than usual, slightly sweaty” }

表 6.9: 抽象的な命令の JSON 表示

システム内のデータ共有解釈モジュール、デバイス制御モジュール、動作対応モジュールは HTTP を通じて JSON メッセージを伝送し、相互に連携する。この連携により、命令は適切に解釈され、必要なデバイス制御命令に変換されてデバイスに送信される。最終的に、システムは効率的に命令を処理し、デバイスに適切な操作命令を発行する。

6.3 実験ユースケース

本研究における実験は、構築した Multi-Mode Kominkan Operating System（MKOS）の機能と性能を評価することを目的としている。以下に、主要な実験内容を述べる。

6.3.1 抽象度命令の処理実験

本実験では、構築したシステムが抽象的な命令（抽象度低、中、高）をどの程度正確に処理できるかを評価する。サービスから送信された命令を受信した後、システムはこれらの命令をデータ共有解釈モジュール、デバイス制御モジュール、動作対応モジュールを経由して処理し、最終的に適切なデバイス制御命令を発行する。

実施方法:

実験では、抽象度が異なる複数の命令（抽象度低、中、高）をシステムに送信し、各モジュールがこれらの命令をどのように処理し、最終的なデバイス制御命令にどのように変換するかを観察した。各命令の処理時間と正確性を記録し、システムの効率性と信頼性を評価した。

実験結果:

実験結果は、表 6.10, 表 6.11 で示される。

低抽象度の命令は、変換を必要とせず、直接にデバイス制御モジュールに送信される。これらの命令は、システムによって具体的なデバイス向けの操作命令として扱われ、デバイスのポリシーに基づいて適切なアクションを実行する。例えば、「カーテンを閉じる」という命令は、カーテンを操作するデバイスに直接命令を発送することに相当します。

中抽象度の命令では、命令に含まれる条件（例：特定の日付や場所）を歴史データと照合し、その条件下で過去に実行された適切な低抽象度の命令を選択する。これにより、システムはより動的な状況に対応し、過去のデータに基づいて最適なデバイス操作を行うことができる。

高抽象度の命令に関しては、データ共有解釈モジュールが特定のキーワードを発見する機能を持っています。特定のキーワードが発見されると、それに関連するデフォルトの設定目標に基づいて低抽象度の命令に変換される。この変換プロセスを通じて、システムは一部高度な指示を具体的なデバイス操作に落とし込み、デバイス制御モジュールに送信することができできる。

評価:

システム導入後の命令処理時間を評価した結果、図 6.2 および 6.3 に示される通り、命令の実行数に対する処理時間が明らかになった。横軸は命令実行数、縦軸は処理時間（秒）で、この図からシステムが一定数の命令を効率的に処理できることが確認できる。

実験結果によると、コマンド数の増加に伴い、所要時間は直線的に増加する傾向が見られる。Raspberry Pi 3b を使用した場合、各コマンドの処理には平均 200〜300 ミリ秒かかることが観察された。システムが複数のコマンドを連続して処理する際も、所要時間は直線的に増加する。

また、コマンドやポリシー、優先度情報を Excel ファイルとして保存し、これを読み込む際にパフォーマンスが影響を受けることが判明した。Excel ファイルの読み込みにより、コマンド処理時間には最大で約 100 ミリ秒の差が生じていた。しかしながら、市民ホールのような実際の使用環境では、この遅延はほとんど影響しないと考えられる。将来的には、Raspberry Pi 3b よりも高性能なデバイスを使用することで、コマンド処理時間がさらに短縮される可能性がある。

抽象度	命令例	処理結果
低抽象度	エアコンが 24 度に設定する	成功
低抽象度	照明オン、明るさ 70 %	成功
低抽象度	窓閉める	成功
低抽象度	カーテン閉じる	成功
中抽象度	石川県能美市の 9 月 27 日で、多少涼しめに運転	成功
中抽象度	午後 3 時から 4 時まで日差しを遮る	成功
中抽象度	夜間、静かに運転	成功
中抽象度	夏期、エネルギー効率よく運転	成功
中抽象度	石川県能美市の 9 月 27 日で、多少涼しめに運転	成功
高抽象度	普段の生活よりも暖かく、少し汗ばむ程度に運転	成功
高抽象度	快適でリラックスできる環境	成功
高抽象度	活動的な雰囲気	失敗
高抽象度	静かで集中しやすい環境	成功
高抽象度	落ち着いた照明でリラックス	失敗

表 6.10: 抽象度別命令の処理

抽象度	Input	Output
低抽象度	{ ”device”: ”air con”, ”setting”: 24 }	{ ”device”: ”air con”, ”setting”: 24 }
低抽象度	{ ”device”: ”light”, ”setting”: 70 }	{ ”device”: ”light”, ”setting”: 70 }
低抽象度	{ ”device”:”fan”, ”setting”: 30 }	{ ”device”:”fan”, ”setting”: 30 }
低抽象度	{ ”device”: ”window”, ”setting”: 0 }	{ ”device”: ”window”, ”setting”: 0 }
低抽象度	{ ”device”: ”curtain”, ”setting”: 0 }	{ ”device”: ”curtain”, ”setting”: 0 }
中抽象度	{ ”location”: ”Nomi, Ishikawa”, ”date”: ”2023-09-27”, ”preference”: ”slightly cool” }	{ ”device”: ”air con”, ”setting”: 23 }
中抽象度	{ ”timestart”: ”15:00”, ”timeend”: ”16:00”, ”action”: ”sunlight block” }	{ ”device”: ”curtain”, ”setting”: 0 }
中抽象度	{ ”time”: ”night”, ”operation”: ”quiet” },	{ ”device”: ”window”, ”setting”: 0 }
中抽象度	{ ”time”: ”morning”, ”temperature”: ”warm” }	{ ”device”: ”air con”, ”setting”: 24 }
中抽象度	{ ”season”: ”summer”, ”efficiency”: ”high” }	{ ”device”:”fan”, ”setting”: 70 }
高抽象度	{ ”preference”: ”warmer than usual, slightly sweaty” }	{ ”device”: ”air con”, ”setting”: 26 }
高抽象度	{ ”preference”: ”comfortable and relaxing” }	{ ”device”: ”air con”, ”setting”: 22 }
高抽象度	{ ”preference”: ”quiet and focused” }	{”device”: ”window”, ”setting”: 0 }

表 6.11: 抽象的な命令の Input と Output

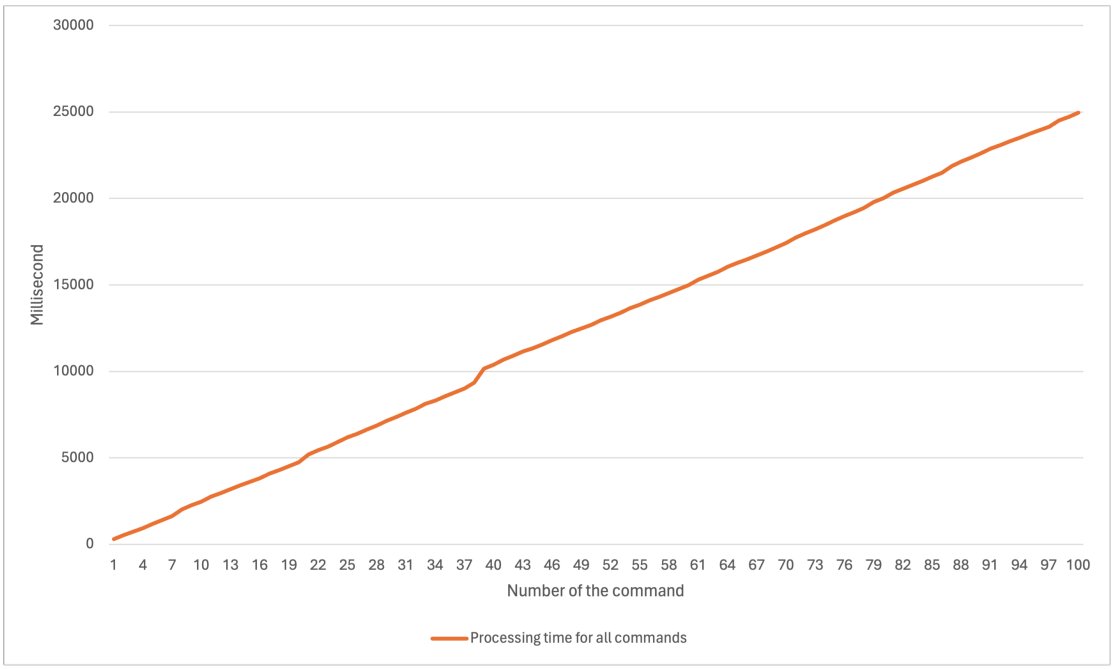

図 6.2: 複数コマンドの処理に要する時間

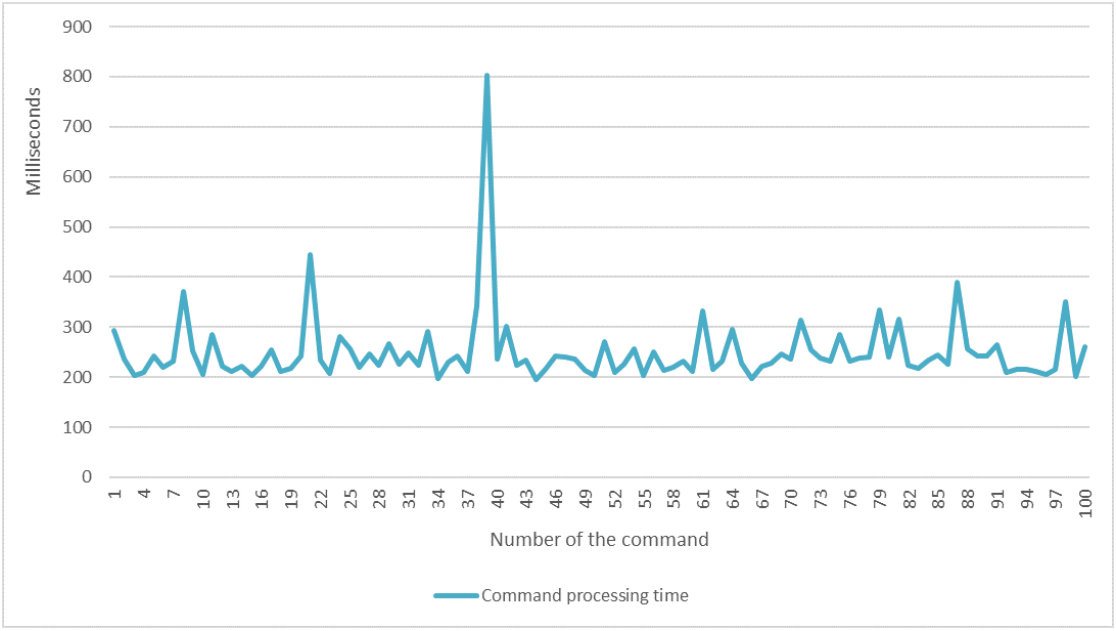

図 6.3: 各コマンドの処理に要する時間

6.3.2　平常時モードと非常時モードの切り替え実験

この実験では、システムが平常時モード（Normal mode）と非常時モード（Emergency mode）の間でどのように切り替えを行うかを評価する。モードに応じて、システムは異なるデバイス制御ポリシーを適用する。この実験では、モード切り替えの際のシステムの反応速度と適応性を評価する。

実施方法:

この実験では、動作対応モジュールが搭載されたRaspberry Piを使用し、平常時モード（Normal mode）と非常時モード（Emergency mode）の間での切り替えを実施した。モジュールのRaspberry Piに接続されたキーボードを通じて、実験担当者が手動でモードの切り替えを行う。同じ命令をこれら2つの異なるモードで処理し、各モードでの出力される命令の違いを観察する。実験では、特定の命令（例えば、「空調を24° Cに設定する」）が平常時モードと非常時モードでどのように異なって処理されるかを詳細に記録する。このプロセスを通じて、システムが異なるモードに基づいて適切なデバイス制御ポリシーを適用する能力を評価する。

実験結果:

この実験では、平常時モードと非常時モードでのデバイス制御命令の変更を評価し、それぞれのモードでのデバイス設定範囲と実際にデバイスに送信される設定値を比較した。以下の表は、異なるモードでのデバイス制御命令の入出力を示している。表6.12から、非常時モードでは、デバイスの設定値がより制限された範囲に修正され、平常時モードよりも厳格な制御が行われることが分かる。修正マークが「Yes」となっている項目は、システムが自動的に設定値を修正し、適切な範囲内に収めたことを示している。

評価:

本実験において、システムは平常時モードと非常時モードの間での切り替えを効率的に行うことが確認された。すべての命令は、現在のモードに対応した設定範囲に基づいて適切に処理され、デバイスに送信された。特に非常時モードでは、デバイスの設定値が厳しく制限され、より安全かつ適切な制御が実施されることが観察された。

この機能は緊急時や特定の状況下でのシステムの適応性を高めるために重要で

制御するデバイス	設定値	平常時モードの設定範囲	デバイスへの設定値	修正
空調	22 ° C	22 ° C〜26 ° C	22 ° C	No
空調	21 ° C	22 ° C〜26 ° C	22 ° C	Yes
空調	28 ° C	22 ° C〜26 ° C	26 ° C	Yes
照明	70%	50%〜100%	70%	No
照明	55%	50%〜100%	55%	No
照明	20%	50%〜100%	50%	Yes
カーテン	閉じる	開閉	閉じる	No
カーテン	開ける	開閉	開ける	No
窓	開ける	開閉	閉じる	No

表 6.12: 平常時モードにおけるデバイス制御命令の入出力

制御するデバイス	設定値	非常時モードの設定範囲	デバイスへの設定値	修正
空調	22 ° C	23 ° C〜25 ° C	23 ° C	Yes
空調	21 ° C	23 ° C〜25 ° C	23 ° C	Yes
空調	28 ° C	23 ° C〜25 ° C	25 ° C	Yes
照明	70%	60%〜80%	70%	No
照明	55%	60%〜80%	60%	Yes
照明	20%	60%〜80%	60%	Yes
カーテン	閉じる	開のみ	開ける	Yes
カーテン	開ける	開のみ	開ける	No
窓	開ける	閉のみ	閉じる	Yes

表 6.13: 非常時モードにおけるデバイス制御命令の入出力

ある。非常時モードでは、デバイスの設定値が厳しく制限されることで、安全性が向上し、システム全体の安定性を保持することができる。また、平常時モードでは、より広範な設定範囲を許容することで、日常的な運用において必要な柔軟性を提供する。

6.3.3　命令の衝突回避機能の実験

本実験では、データ共有解釈モジュールが複数のサービスから受け取った命令を低抽象度の命令に解釈した後、同じデバイスに対する命令が衝突する場合の処理方法を評価する。

実施方法:

この実験では、模擬サービスを表す複数の Raspberry Pi を使用し、異なるサービスが同じデバイスに対して異なる設定値を持つ命令をデータ共有解釈モジュールに送信する。各サービスは異なる抽象度と優先度を持ち、これによりシステムは複数の命令を受け取った際の衝突回避能力を試される。データ共有解釈モジュールはこれらの命令を解析し、適切なデバイス制御命令に変換してデバイス制御モジュールに送信する。実験の重要な部分は、デバイス制御モジュールの出力を観察し、どの命令が最終的にデバイスに送信されるかを確認することである。もし優先度が最も高いサービスからの命令が選択され、デバイスに送信される場合、システムの衝突回避機能は正常に機能していると判断される。

実施結果:

複数のサービスから同じデバイスに対する命令が発行された際、システムはサービスの優先度を比較してして、優先度高いのサービスの命令を選択し、デバイス制御モジュールに送信することに成功した。表 6.14 は、複数のサービスが同じデバイスを制御しようとした際の命令処理の例を示している。この表にサービスの優先度を示す、数値大きいの方は優先度高いの意味である。

評価:

本実験により、システムが複数のサービスからの命令を効率的に処理し、衝突を回避する能力を持っていることが確認された。特に、複数のサービスが同じデバイスに対して異なる命令を出した場合、システムはサービスの優先度に基づいて適切な命令を選択し、デバイスに送信する。重要な点として、サービスの優先度は平常時モードと非常時モードによって変化することが観察された。非常時モードでは、特定のサービスがより高い優先度を持ち、緊急状況下での適切なデバイス制御を確保する。一方、平常時モードでは、優先度は通常の運用条件に基づいて設定される。このように、システムは状況に応じて優先度を動的に調整することができ、異なるシナリオに柔軟に対応する能力を持つ。

サービス	制御するデバイス	設定値	優先度	モード	出力命令
サービス A	空調	24 ° C	7	平常時	{ ”device”: ”air con”, ”setting”: ”24 ° C” }
サービス B		22 ° C	5	平常時	
サービス A	照明	70%	3	平常時	{ ”device”: ”light”, ”setting”: ”50%” }
サービス C		50%	5	平常時	
サービス A	空調	24 ° C	6	非常時	{ ”device”: ”air con”, ”setting”: ”22 ° C” }
サービス B		22 ° C	8	非常時	
サービス A	照明	70%	6	非常時	{ ”device”: ”light”, ”setting”: ”50%” }
サービス C		50%	8	非常時	

表 6.14: 命令の衝突回避機能の実験結果

第7章　考察

本研究の課題

本研究で開発した建物 OS は、現段階では広く普及していないという重要な限界を持つ。また、実験に使用したデータは仮想的に生成されたものであり、実際の建物内デバイスの制御命令の多様性や複雑性を完全には反映していない。このため、実際の建物環境での OS の適用性や有効性については、さらなる検証が必要である。

現在の公民館などの公共施設におけるデバイス管理は、まだシステム化されていないか、限定的な範囲でのみ行われている。このため、建物 OS が提供するリソース管理と制御の機能を最大限に活用するためには、建物内のデバイスをシステムと連携させる必要があるが、これが現実的な課題となっている。

今後の展開

将来的な研究の方向性として、高抽象度の命令処理の改善が挙げられる。大規模言語モデル（LLM）や自然言語処理技術を活用し、より洗練された命令解釈モデルの開発を目指す。これにより、人間の自然な言語で与えられた命令を効率的に解析し、適切なデバイス制御命令に変換する能力を向上させることができる。

また、実際の建物環境でのシステム適用に関する研究も重要である。実際のデバイスとの連携や、より現実的な環境でのシステムの性能と効果を評価することで、システムの実用性を高めることができる。このような研究は、建物 OS の実世界での応用を促進し、公共施設の運営を効率化するための重要な一歩となるだろう。

第8章　結論

本研究は、公民館向けの新しい建物OSである「Multimode Kominkan Operating System (MKOS)」の設計と実装を通じて、建物管理システムの新たな可能性を探求した。MKOSは、日常のコミュニティ活動支援と災害時の避難所機能の両方をサポートすることを目的とし、これまでの建物OSの限界を克服することを試みた。特に、データ共有解釈モジュールを通じて複数のサービスからの命令を効率的に処理し、デバイス制御の衝突を回避する能力が示された。

本研究で開発されたMKOSは、公民館のような地域コミュニティ施設に特化した建物管理システムとしての新しいスタンダードを提案する。明確で使いやすいAPIの提供により、開発者やプログラマがシステムを容易に利用でき、公民館の運用管理の効率化と災害時の迅速な対応が可能になる。また、システムの設計原理と主要コンポーネントの詳細な検討により、建物OSの設計における新たな方向性を提示した。

模擬実験を通じて、MKOSの機能性と効果を実証し、特に高抽象度の命令処理、モード切り替え機能、命令の衝突回避機能の有効性が確認された。これにより、MKOSは公民館の日常的な運用と非常時の対応の両方をサポートする有能なシステムであることが示された。

結論として、本研究は公民館のような地域コミュニティ施設における建物管理システムの新たなアプローチを提供し、建物OSの概念を拡張することに成功した。今後の研究では、実際の公民館環境でのMKOSの適用と、その効果のさらなる検証が求められる。

付 録A　実験のデータ

command num	Send Time	Receive Time	Delay Seconds	Command processing time	Processing time for all commands
1	1706522436	1706522436	0.294252634	294.252634	294.252634
2	1706522436	1706522436	0.236847878	236.8478775	531.1005116
3	1706522436	1706522437	0.202967882	202.9678822	734.0683937
4	1706522437	1706522437	0.210142851	210.1428509	944.2112446
5	1706522437	1706522437	0.24143815	241.4381504	1185.649395
6	1706522437	1706522437	0.21908474	219.0847397	1404.734135
7	1706522437	1706522438	0.231610537	231.6105366	1636.344671
8	1706522438	1706522438	0.370497704	370.4977036	2006.842375
9	1706522438	1706522438	0.251778603	251.7786026	2258.620977
10	1706522438	1706522438	0.206392765	206.392765	2465.013742
11	1706544669	1706544669	0.285662889	285.6628895	2750.676632
12	1706544669	1706544670	0.221461296	221.4612961	2972.137928
13	1706544670	1706544670	0.211122513	211.1225128	3183.260441
14	1706544670	1706544670	0.222318172	222.3181725	3405.578613
15	1706544670	1706544670	0.202722788	202.7227879	3608.301401
16	1706544670	1706544670	0.221384525	221.3845253	3829.685926
17	1706544670	1706544671	0.254240513	254.2405128	4083.926439
18	1706544671	1706544671	0.211819887	211.8198872	4295.746326
19	1706544671	1706544671	0.217186928	217.1869278	4512.933254
20	1706544671	1706544671	0.243212938	243.2129383	4756.146193
21	1706545262	1706545263	0.445269346	445.2693462	5201.415539
22	1706545263	1706545263	0.233273983	233.273983	5434.689522
23	1706545263	1706545263	0.206672668	206.6726685	5641.36219
24	1706545263	1706545264	0.282139301	282.1393013	5923.501492
25	1706545264	1706545264	0.255895376	255.8953762	6179.396868
26	1706545264	1706545264	0.220169067	220.1690674	6399.565935
27	1706545264	1706545264	0.246027946	246.0279465	6645.593882
28	1706545264	1706545265	0.222905636	222.9056358	6868.499517
29	1706545265	1706545265	0.2663486	266.3486004	7134.848118
30	1706545265	1706545265	0.224910736	224.9107361	7359.758854
31	1706545334	1706545335	0.248063326	248.0633259	7607.82218
32	1706545335	1706545335	0.223554134	223.5541344	7831.376314
33	1706545335	1706545335	0.291372538	291.3725376	8122.748852
34	1706545335	1706545335	0.197290182	197.2901821	8320.039034
35	1706545335	1706545336	0.230315208	230.3152084	8550.354242
36	1706545336	1706545336	0.241683483	241.6834831	8792.037725
37	1706545336	1706545336	0.212523937	212.5239372	9004.561663
38	1706545336	1706545336	0.342561007	342.5610065	9347.122669
39	1706545336	1706545337	0.803632975	803.6329746	10150.75564
40	1706545337	1706545337	0.236996889	236.9968891	10387.75253
41	1706545361	1706545361	0.300955772	300.9557724	10688.70831
42	1706545361	1706545362	0.223757029	223.7570286	10912.46533
43	1706545362	1706545362	0.233155489	233.155489	11145.62082
44	1706545362	1706545362	0.19436574	194.3657398	11339.98656
45	1706545362	1706545362	0.215644598	215.644598	11555.63116
46	1706545362	1706545362	0.242267847	242.2678471	11797.89901
47	1706545362	1706545363	0.240365744	240.3657436	12038.26475
48	1706545363	1706545363	0.237140656	237.1406555	12275.40541

49	1706545363	1706545363	0.212617636	212.6176357	12488.02304
50	1706545363	1706545363	0.202464581	202.4645805	12690.48762
51	1706545453	1706545454	0.26999259	269.99259	12960.48021
52	1706545454	1706545454	0.209747314	209.7473145	13170.22753
53	1706545454	1706545454	0.225061417	225.0614166	13395.28894
54	1706545454	1706545454	0.25556016	255.5601597	13650.8491
55	1706545454	1706545454	0.203439951	203.4399509	13854.28905
56	1706545454	1706545455	0.24973321	249.7332096	14104.02226
57	1706545455	1706545455	0.214427471	214.4274712	14318.44974
58	1706545455	1706545455	0.219856977	219.8569775	14538.30671
59	1706545455	1706545455	0.231662035	231.662035	14769.96875
60	1706545455	1706545456	0.210771322	210.7713223	14980.74007
61	1706545488	1706545488	0.332010269	332.0102692	15312.75034
62	1706545488	1706545488	0.216584444	216.584444	15529.33478
63	1706545488	1706545488	0.232027054	232.0270538	15761.36184
64	1706545488	1706545489	0.295686722	295.6867218	16057.04856
65	1706545489	1706545489	0.227588654	227.5886536	16284.63721
66	1706545489	1706545489	0.19817996	198.1799603	16482.81717
67	1706545489	1706545489	0.222284317	222.284317	16705.10149
68	1706545489	1706545490	0.228173971	228.1739712	16933.27546
69	1706545490	1706545490	0.245924711	245.9247112	17179.20017
70	1706545490	1706545490	0.23671174	236.7117405	17415.91191
71	1706545526	1706545527	0.312936783	312.9367828	17728.8487
72	1706545527	1706545527	0.255310535	255.3105354	17984.15923
73	1706545527	1706545527	0.237469912	237.4699116	18221.62914
74	1706545527	1706545527	0.231422663	231.4226627	18453.05181
75	1706545527	1706545528	0.285712481	285.7124805	18738.76429
76	1706545528	1706545528	0.232367039	232.3670387	18971.13132
77	1706545528	1706545528	0.23732233	237.3223305	19208.45366
78	1706545528	1706545528	0.239517212	239.5172119	19447.97087
79	1706545528	1706545529	0.335035563	335.0355625	19783.00643
80	1706545529	1706545529	0.23996973	239.9697304	20022.97616
81	1706545682	1706545683	0.31520462	315.2046204	20338.18078
82	1706545683	1706545683	0.223916054	223.9160538	20562.09683
83	1706545683	1706545683	0.218561649	218.5616493	20780.65848
84	1706545683	1706545683	0.234423399	234.423399	21015.08188
85	1706545683	1706545684	0.244664907	244.6649075	21259.74679
86	1706545684	1706545684	0.225081682	225.0816822	21484.82847
87	1706545684	1706545684	0.389349461	389.3494606	21874.17793
88	1706545684	1706545684	0.256213665	256.213665	22130.3916
89	1706545684	1706545685	0.242921829	242.9218292	22373.31343
90	1706545685	1706545685	0.243075848	243.0758476	22616.38927
91	1706545743	1706545743	0.26439476	264.3947601	22880.78403
92	1706545743	1706545743	0.208596706	208.5967064	23089.38074
93	1706545743	1706545744	0.215364695	215.3646946	23304.74544
94	1706545744	1706545744	0.215211153	215.211153	23519.95659
95	1706545744	1706545744	0.211664915	211.6649151	23731.6215
96	1706545744	1706545744	0.206084967	206.0849667	23937.70647
97	1706545744	1706545744	0.215825319	215.8253193	24153.53179
98	1706545744	1706545745	0.350621939	350.6219387	24504.15373
99	1706545745	1706545745	0.201010704	201.010704	24705.16443
100	1706545745	1706545745	0.261557341	261.5573406	24966.72177

付 録B　ユースケースとデータの 検討

カテゴリ	ユースケース名	目的	アプリケーションの具体的な動作（基本フロー）	必要なデータ	必要なセンサー	必要なアクチュエータ	稼働状況 （平常時のみ、 非常時のみ、 平常時・非常時）	平常時○・非常時△の 状況に応じたユースケースの厳選化	
1. 施設管理	1.1 温熱管理	室内の温熱を快適な状態に管理する目的	1. 温度データを取得する 2. 設定した温度と比較し，暖房，冷房，除湿を判断する 3. エアコン等のアクチュエータに動作指示をする	室内温度，湿度 室外温度，湿度	温度センサー 温度センサー	エアコン，換気扇，電動窓	平常時・非常時	○利用者の快適性を維持する △緊急時に備え，室温を適切に保つ	
	1.2 空気質管理	室内の空気に外気を入れ換えることや， 空気清浄機を用いることで， 室内の空気質を管理する事が目的	1. 空気質データを取得する 2. 基準値と比較し，空気清浄の必要性を判断する 3. 換気・空気清浄機に動作指示する	室内温度，湿度 CO2 濃度 PM2.5濃度	温度センサー 温度センサー CO2 センサー PM2.5センサー	エアコン，空気清浄機	平常時・非常時	○室内の空気を清潔に保つ △有害物質の拡散を防止する	
	1.3 照明管理	効率的なエネルギー使用，環境への配慮，快適な照明環境の提供	1. 照明データを取得する 2. 設定した照明基準と比較し，照明のオン・オフや強度を判断する 3. 照明に動作指示する	室内照度 人体検知情報 タイマー情報	照度センサー 人感センサー	照明器具	平常時・非常時	○利用者の利便性と省エネのバランスを重視 △利用者の安全と防災を重視	
	1.4 電子錠を用いた施設の管理	カードキーを権利者に付与することで，部屋の鍵を管理する目的	1.権利者の情報を登録する 2.カードキーを発行する 3.カードキーを認証する	利用者の氏名、ID カードキーID アクセス許可された部屋の情報 解錠・施錠状況の静止画・動画	カードリーダー カメラ	電子錠	平常時・非常時	○不正者の侵入や不正利用を防止する △利用者が安全に施設を利用できるようにする	
	1.5 電子錠・スマホを用いた鍵権限の付与・管理	部屋の利用者に，一定期間のみ解錠できる権限をスマホに付与する事で，部屋の鍵管理や利用者の利便性を向上する目的	1.利用者の情報を登録する 2.スマホアプリをインストールする 3.鍵権限を付与する 4.鍵権限を解除する	利用者の氏名、ID スマホの識別情報 アクセス許可された部屋の情報	カードリーダー	電子錠	平常時・非常時	○不正者の侵入や不正利用を防止する △利用者が安全に施設を利用できるようにする	
	1.6 キーボックスの鍵管理	キーボックスの鍵を適切で管理することで，複数の鍵を管理する目的	1.キーボックスに鍵を登録する 2.鍵を貸し出す 3.鍵を返却する	設置場所情報 鍵の情報（数量，種類） 貸出・返却情報（期間，利用者）	-	キーボックス	平常時・非常時	○鍵の管理を強化し，不正利用を防止する △		
	1.7 在室人数の管理	感染症発生時に密閉空間を防ぐ目的	1.在室人数を計測する 2.在室人数の上限を設定，更新する 3.在室人数の上限を超えた場合，入場を制限する	部屋の情報（番号，位置，人数上限） 人数計測情報	人感センサー カメラ	入退室管理システム モニター	平常時・非常時	○利用者の利便性向上のために利用状況を把握する △感染症など の拡大を防止するために，密集を避ける	
2. 災害・防犯監視	2.1 火災検知	早期に火災を検出し，その情報を適切な人々やシステムに迅速に通知	1. 火災データを取得する 2. 火災の発報を判断する 3. 建物内の人に通知する。 4. 関連施設へ、消防へ通知する。	火災情報 ・熱の有無，位置情報 ・煙の有無，位置情報	熱センサー 煙センサー	ディスプレイ スピーカー	平常時・非常時	検知のみのため特になし	
	2.2 水害監視	浸水や雨量や地下水位の異常上昇を検出し、洪水や水害のリスクを早期に把握	1. 水位データを取得する 2. 基準以上の水位であり，浸水の危険性があるか判断する 3. 建物内の人に通知する。 4. 関連施設へ通知する。	水位	水位センサー	ディスプレイ スピーカー	平常時・非常時	検知のみのため特になし	
	2.3 雪害監視	大雪や雪による施設建物や駐車場の積雪量監視が目的	1. 施設の積雪データを取得する 2. 基準以上の積雪量であるか判断する 3. 建物内の人に通知する 4. 関連施設へ通知する 5. 除雪依頼をする	積雪量	積雪センサー	ディスプレイ スピーカー	平常時・非常時	検知のみのため特になし	
	2.4 風害監視	異常な風速または突風を検出し、風による損壊や危険のリスクを早期に把握	1. 風速データを取得する 2. 基準以上の積雪量であるか判断する 3. 建物内の人に通知する 4. 関連施設へ通知する	風速	風速センサー	ディスプレイ スピーカー	平常時・非常時	検知のみのため特になし	
	2.5 豪雨監視	豪雨や大雨による被害の早期発見と予防が目的，異常な降水量を検出	1. 雨量データを取得する 2. 基準以上の雨量であるか判断する 3. 建物内の人に通知する 4. 関連施設へ通知する	雨量	雨量センサー	ディスプレイ スピーカー	平常時・非常時	検知のみのため特になし	
	2.6 不審者への防犯監視	不審者への防犯を目的とし，建物周辺を監視する．	1. 静止画，動画を取得する 2. 人物が，施設内へ侵入したか判断する 3. 建物内の人に通知する 4. 関連施設へ通知する	静止画，動画	カメラ	ディスプレイ スピーカー	平常時・非常時	検知のみのため特になし	
	2.7 害獣対策としての監視	害獣から施設を守ることを目的とし，建物周囲を監視する．	1. 静止画，動画を取得する 2. 害獣がいるか判断する 3. 建物内の人に通知する。 4. 関連施設へ通知する	静止画，動画	カメラ	ディスプレイ スピーカー	平常時・非常時	検知のみのため特になし	
3. 健康・医療	3.1 健康診断	住民の健康状態を管理する目的	1. 住民の健康情報を測定する 2. 健康情報を保存する 3. データを管理する	住民ID 氏名 身長 体重 血圧 骨密度	カードリーダー 身長計 体重計 血圧計 骨密度計	-	平常時のみ	-	
	3.2 オンライン診療向け個室整備	診察時には医者の個人情報を扱うため，第三者が室内にいない状態をつくるなど「プライバシーを確保」することが目的	1. 住民が指定の利用登録をする 2. 利用時にユーザー認証する 3. 電子錠を解錠し，指定を提供する	住民ID	カードリーダー	電子錠	平常時のみ	-	
	3.3 体温モニタリング	公民館の訪問者や職員の健康状態を監視し、発熱などの症状を早期に検出	1. 体温データを取得する 2. 基準値と比較し，発熱の有無を判断する 3. ディスプレイ，スピーカーに動作指示し，通知する	体温データ	体温計 サーモグラフィーカメラ	ディスプレイ	平常時・非常時	○感染症の拡大を防ぐ △感染症の拡大を防ぐ事と合わせて，適切な医療機関へ繋げる事が重要	
	3.4 メンタルヘルスケア	個々の人の精神的健康状態を把握し、必要に応じて適切なケアやサポートを提供する	1. ユーザーのメンタルヘルスケアに関するアンケートデータを取得する 2. アンケートデータからスコアを算出 3. 基準値以上であれば，専門医へ情報を通知する．ケアやサポートへ繋げる	メンタルヘルスアンケートデータ	-	-	平常時・非常時	○定期的なデータを取ることが重要 △災害時はメンタルの急激な変化があるため，適切な医療機関へ繋げる事が重要	
	4.1 避難者名簿作成	避難者やマイナンバーカードでのチェックインを行い名簿を作成し，避難した人ととどうていない人を管理する目的	1.避難者の避難所入室時，マイナンバーカード等でユーザー認証を行う（ユーザー識別） 2.アプリケーションが避難者を登録する． 3.避難者をアプリケーション端末や外部から閲覧できるようにする．	避難者データ ・住民ID（マイナンバーカード） ・避難グループ人数	カードリーダー	ディスプレイ	非常時のみ	-	

4. 避難所利用	4.2 仮設居住場所の割り当て	顔写真やマイナンバーカードでチェックインした避難者に，仮設居住場所を割り当てる目的	1. 避難者の避難所入居時，人数登録をする 2. アプリケーションが避難者の人数と仮設居住場所データの収容可能人数を照らし合わせ，割当先の仮設居住場所を決定する 3. 避難者に割当先の仮設居住場所を提示する	仮設居住場所データ ・居住場所ID ・収容人数 避難者データ ・住民ID ・避難グループ人数	カードリーダー	ディスプレイ スピーカー	非常時のみ	-	
	4.3 備蓄品在庫管理	避難所として公共施設を利用する際の備蓄品の在庫管理をする目的	1. 避難所内の備蓄品データの登録を行う 2. 備蓄品データを権限を持つ第三者が閲覧できるようにする 3. 使用時に備蓄品データの更新を行う	避難所内備蓄品データ ・品名 ・数量	-	ディスプレイ	平常時・非常時	○備蓄品を適切に保管する △備蓄品を適切に提供する	
	4.4 倉庫から各避難所への備品管理	倉庫での備蓄品管理 倉庫から各避難所へ備蓄品を郵送する際の数量の管理目的	1. 倉庫内の備蓄品データの登録を行う 2. 各避難所内の備蓄品データと避難者データを参照して，倉庫内の備蓄品の受け渡し管理を行う。	倉庫内備蓄品データ ・品名 ・数量 各避難所内備蓄品データ ・避難所名 ・品名 ・数量	-	ディスプレイ	非常時のみ	-	
	4.5 分散避難時の物資配給拠点	避難所ではなく各世帯にて避難する分散避難時に食料や水，生活用品といった物資を配給する場所として利用する目的 マイナンバーカードなどの認証と物資の分配をする機能を持つ	1. 物資配給データを登録する。 2. 物資配給を目的として避難所へ来た住民の認証を行う。 3. 住民情報を元に物資の分配を行う。	避難者データ(受け取り者) 住民ID 世帯人数 物資配給データ ・品名 ・数量	-	ディスプレイ	非常時のみ	-	
	4.6 非常時の電力管理	電力供給が中断された場合にも、重要なシステムやデバイスの運用を維持する	1. 電力供給対象となる重要システムを登録する。 2. 電力停止時に電力共有する対象システムへ電力供給を行う。	電力供給対象システム ・システム名 ・供給先	-	UPS等	非常時のみ	-	
	4.7 避難者の位置把握	各部屋や避難経路上のユーザーを検知し，ユーザー位置を把握する。 避難者の位置状況把握や救助隊への現場状況連絡へ用いる	1. ユーザー位置データを取得する。 2. ユーザー位置データを救助隊への通知する。	人体検知データ ・在不在 ユーザー位置情報データ ・氏名 ・位置	-	ディスプレイ スピーカー	非常時のみ	要避難者の年齢，性別，体格といった情報もあると，更に適切な救助活動が可能になると考えられる	
	4.8 避難経路支援	火災場所といった通れない場所の地点情報を元に，建物内外の安全な避難経路をユーザーに通知する。	1. 避難経路情報と部屋毎の災害情報を取得する。 2. 避難経路情報を更新する。 3. 最新の避難経路情報をユーザーへ通知する。	避難経路 災害情報 ・熱の有無，位置情報 ・煙の有無，位置情報	熱センサー ・熱の有無 ・設置位置 煙センサー ・煙の有無 ・設置位置	ディスプレイ スピーカー	非常時のみ	ユーザーの位置情報や身体情報を活用できれば，ユーザー毎に最適な避難経路を選択，提示することが可能になると考えられる	

謝 辞

本論文の執筆に際して、多くの方々からの貴重な助言とサポートを頂きました。ここに深く感謝の意を表します。主指導教員である丹康雄教授に、研究テーマの選定からゼミでの資料作成に至るまで、丁寧かつ熱心なご指導を賜りましたことに深く感謝申し上げます。また、副指導教員であるリム勇仁准教授には、ゼミでの発表に対する貴重なアドバイスと知識の指導を頂き、心より感謝致します。Sioutis 先生には、実験に関するご指導を頂き、この研究への理解を深めることができました。また、Pham 先生には、研究に関する質問への丁寧な回答や修論の作成におけるご支援を頂き、深く感謝しております。同期であり友人でもある草野さんには、公民館に関する共同研究において多大な助力を頂きました。また、リム研究室の皆様には、日常生活における友情とサポートを頂き、感謝の念に堪えません。最後に、遠く離れた中国の両親には、絶えず精神的および経済的な支援を頂き、ここに深い感謝の意を表します。皆様のご支援があってこそ、本論文を完成させることができました。

参考文献

[1] 清水建設（株）. 建物運用のデジタル変革を支援する建物 os「dx-core」を商品化. https://www.shimz.co.jp/company/about/news-release/2020/2020025.html, 2020.

[2] 平田あや and 田中隆文. 地域防災における公民館の役割について. 中部森林研究, 68:83–86, 2020.

[3] 愛媛県教育委員会事務局管理部社会教育課. ちょっと自慢の公民館事業. https://ehime-c.esnet.ed.jp/shougai/seijinkyoiku/kominkan/toppi.html.

[4] 公益社団法人全国公民館連合会 事務局次長 村上英己. 防災に活かす公民館. page 21. 内閣府（防災担当）普及啓発・連携参事官室, 平成 30 年.

[5] 館山市役所. 公民館とは…（公民館ってどんなところ？）. https://www.city.tateyama.chiba.jp/kouminkan/page100069.html.

[6] 公民館・図書館等社会教育施設のデジタル活用促進について. 文部科学省 総合教育政策局地域学習推進課, 令和 4 年 6 月 14 日.

[7] 内閣府（防災担当）. 防災気象情報と警戒レベルとの対応について. https://www.jma.go.jp/jma/kishou/know/bosai/alertlevel.html.

[8] Motoaki Yamazaki, Keiichi Hirose, and Michihito Shiraishi. 建物デジタルプラットフォーム 「dx-core (建物 os)」 の開発. 清水建設研究報告, *volume=99, pages=23–28, year=2021,*.

[9] 独立行政法人情報処理推進機構. スマートビルシステムアーキテクチャガイドライン. 2023.

[10] 陳 翔・草野 清重・リム 勇仁・丹 康雄. 公民館向け建物 os における非常時の動作変更方法に関する研究.

[11] Marios Sioutis, Junsoo Kim, Azman Osman Lim, and Yasuo Tan. A home service deployment platform with support for detection and resolution of physical resource conflicts. In *The 1st IEEE Global Conference on Consumer Electronics 2012*, pages 333–336, 2012.