WIPY:独自通信フォーマット KICoH を用いた 既存黒板に後付け可能な自律型清掃ロボット

寺門宗悟† 坪井洸樹† ビンティヌルハズリダリンハナ† 成田幸和†
中村咲心† 坂井智春† 丸山智章† 奥出真理子† 市毛勝正† 周而晶†

概要：教育現場における ICT 活用が進展する中においても、黒板は即時性や視認性の高さから多くの機関で利用され続けている。しかし、手作業による黒板清掃は、多忙な教員にとって時間的制約やチョーク粉による衣服の汚損、健康被害のリスクを伴う無視できない負担となっている。既存の自動清掃システムは高価かつ大規模なものが多く導入の障壁となっていた。本研究では、既存の黒板に設置可能で、安価かつ軽量な自律型黒板清掃ロボットを提案・開発した。本システムは、汎用的な工作部品と ESP32 マイコン、Raspberry Pi を用いたサーバー・ルーターにより構成され、ネオジム磁石を搭載したクローラ機構により垂直壁面での走行を実現している。特筆すべき点として、低コストマイコンでの安定制御とネットワーク帯域の節約を目的とした独自通信フォーマット「KICoH」を設計・実装したことが挙げられる。これにより、ロボットは Web インターフェースで設定されたスケジュールに基づき、完全に自律した清掃動作を行うことが可能となった。本稿では、機体のハードウェア設計、KICoH の仕様、およびシステムの実装詳細について述べるとともに、現状の課題として制御のリアルタイム性や移動機構の信頼性向上について議論する。

1. はじめに

1.1 背景

教育現場における ICT 活用が進む現代においても、即時的な記述や視認性の高さから、多くの教育機関でチョークと黒板が利用され続けている。しかし、その維持管理、特に授業間の清掃作業は、現場の教員にとって無視できない負担となっている。

文部科学省の教員勤務実態調査[1]によれば、教員の在校時間は依然として長く、授業準備に充てられる時間は極めて限定的である。特に大学や高専などの高等教育機関では、教室移動や機材接続など、わずか 10 分程度の休み時間内に多くのタスクをこなす必要がある。特に敷地が広大になる高等教育機関では移動だけで休み時間が終わってしまうことも少なくない。加えて、清掃作業に伴う身体および衣服の汚損も深刻な課題である。手作業による黒板消しでは、飛散した微細なチョーク粒子が作業者の手指や衣服に容易に付着する。特にスーツなどの濃色衣類に付着した粉は除去が困難であり、次の授業へ向かう教員にとって、身だしなみの維持や手洗いの手間といった、本来不要であるはずの時間的・心理的コストを強いる結果となっている。

さらに、授業終了後の黒板清掃に関する明確なルールが徹底されていないケースが多く、学生が板書を消さずに退室することが散見される[2]。これは、前述した「汚れ」への忌避感も一因と推察されるが、結果として次の授業担当者が清掃を行わなければならず、貴重な準備時間がさらに圧迫されるという悪循環が生じている。

1.2 社会的課題と関連研究

手作業による黒板清掃は、時間的損失だけでなく健康面のリスクも伴う。

チョーク粉の吸入は、呼吸器系疾患やアレルギー症状を引き起こす可能性が指摘されており、教室内空気質の観点からも粉塵の飛散を最小限に抑える対策が求められている。

この課題に対し、いくつかの自動化システムが提案されている。黒板自体に回転ベルト機構を組み込んだ一体型装置[3]や、フレームに取り付けた大型のワイパーがスライドする機構[4]などが存在するが、これらは機構が大掛かりでかつ導入コストが高く、既存の黒板への後付けが困難であるという欠点を持つ。また、近年の DIY コミュニティや大学チームにおいても簡易的な清掃ロボットが提案されている[5][6]が、ネットワーク連携機能を持たず、スケジュール管理ができないため毎回手動での起動を要する点が課題となっている。

1.3 目的

そこで本研究では、既存の黒板に設置するだけで使用でき、かつ安価で軽量な自律型黒板清掃ロボットを提案する。本システムは、独自通信フォーマット「KICoH」を用いることで低コストマイコンでの安定制御を実現し、スケジュールに基づく完全自動清掃により、教員や学生の負担軽減と教室環境の改善を目指すものである。

2. 装置の概要

機体は汎用の工作部品を基に作成し、マイコンには ESP32 を採用した。汎用的な部材を用いることで工期の短縮とコストの削減、設計難易度の低減を図った。サーバー兼 Wi-Fi ルーターには Raspberry Pi 5(OS: Ubuntu 24 LTS)を採用した。

ユーザーは黒板消しの左端に本体を置き、その上で Web

† 茨城工業高等専門学校 国際創造工学科

インタフェースを用いてスケジュールと黒板のサイズを入力し、JSON 形式でバックエンドへデータを渡しバックエンドで KICoH 形式(KIC-data)に変換して DB に格納する。本体が起動時、または内部時計が日曜になったタイミングで Wi-Fi 経由で端末 ID を含んだクエリパラメータで対応するデータにアクセスし、合致するデータを取得する。

データと送信時刻(データ構造の詳細は 3 章独自通信フォーマットの節を参照)を取得した本体は、黒板の大きさの情報からルーティングし、受信した送信時刻を基にした内部クロックによる時計のカウントを始め、登録されたスケジュールの曜日時刻に達すると黒板を消す動作を始める。

3. 実装・仕様

3.1 機体

3.1.1 設計

タミヤユニバーサルプレートの上にブレッドボードで回路が実装されており、図 1 に示すように ESP32 のピンヘッダでモータードライバ DRV8835 を経由し制御されている。なお、モータードライバはモーター1 基に対して 1 基の対応対で構成される。ESP32 にはモバイルバッテリーから、モーターには単三電池 2 本直列 3.0V で電源供給されている。黒板消しはユニバーサルプレートにボルト止めされている。機体全体の構成を図 2 に示す。

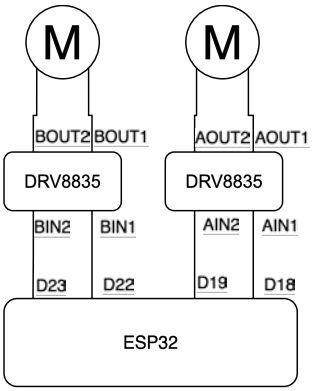

図 1　回路図

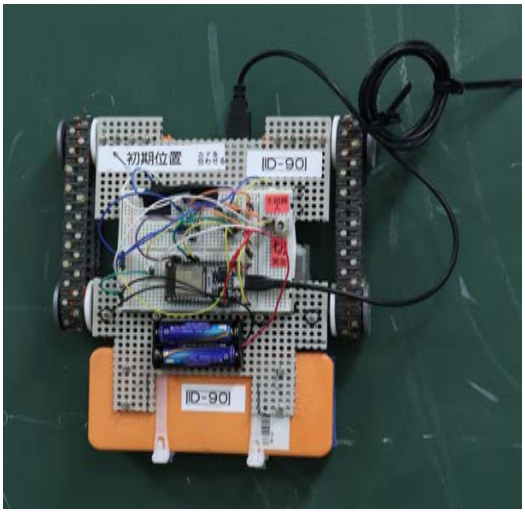

図 2　機体全体の構成

3.1.2 走行関連部分

走行関連部分はタミヤタンク工作基本セット 2 基をベースに図 3 のようにキャタピラにネオジム磁石を貼り付けることで壁での走行を可能にしている。

図 3　キャタピラ部の詳細

3.2 開発方法

本ファームウェアの開発では、なるべく ETC(Easier To Change)を維持できるように疎結合を意識したものとなっており、TDD(Test Driven Development)を採用した。

バックエンド開発には LLM(Large Language Models)をペアプログラミングパートナーとして活用するアジャイル開発手法を採用し、未知の技術スタックにおける実装工数を大幅に短縮した。用いた AI LLM エージェントは主に Gemini 2.5 Pro を用い、補助的に ChatGPT 4o を用いた。

3.3 起動方法

マイコンに電源が供給されるだけで自動的にセットアップが走るようになっている。

3.4 セットアップの内容

3.4.1 KIC-data の取得

3.4.1.1 Wi-Fi のセットアップ

Espressif が提供する Arduino 互換 ESP32 ボードコアに含まれる Wi-Fi ライブラリ（WiFi.h）に依存し，WiFi.begin(SSID, PASSWORD) により接続を試みる。一定時間経過し、接続できない場合は再試行をする。これは Wi-Fi アクセスポイント(AP)との接続を確立するまで試行される。

3.4.1.2 HTTP 通信

KIC-data の取得は，電源供給時の自動セットアップ時に行われ, HTTP 通信を用いる。Espressif が提供する Arduino 互換 ESP32 ボードコアに含まれる WString.h および HTTPClient.h に依存している。指定の URL に id クエリ

を付加した URL を用いて通信を開始し，通信が成功した場合にはペイロードを取得する。

また，通信が成功するまでリクエストは送信され続ける。

3.4.1.3 文字列からデータへの変換

i.ペイロードの分析

ペイロードから取得した文字列をプログラム上で扱えるデータとして変換する。

ペイロードからセミコロン(;)まで文字列を抽出して、プログラム内部で定められたフォーマットの識別子とバージョンが抽出された文字列と等しいか確認。

本来は、区切り文字としてセミコロンを用いた処理を行うべきではなく、まず文字列の先頭から KIC ヘッダーとバージョンとを結合した文字列の長さ分を取得し、それが、KIC ヘッダーとバージョンと一致するかを確認する必要がある。この確認を行わない場合、セミコロンが存在しない入力に対して未定義動作を引き起こす可能性があるためである。

加えて、文字列の末尾にスラッシュが含まれていることを確認する。

以上の二つの条件を満たした場合にのみ、その文字列を KICoH に準拠したものと認め、セミコロンを区切りとして文字列を分割し、トークン化を行う。

ii.トークンのデータ化

それぞれのトークンは、曜日と時刻を表すトークンであれば頭文字の数字といくつかの固定長幅 4 の数字列、黒板の高さと幅を示すトークンであれば 2 つ並びの固定長幅 4 の数字列で構成されているため、それぞれを数値型へ変換する。

また、前節 3.4.1.3.1 において KICoH に準拠した文字列であると判定した場合であっても、各トークンの長さが、あらかじめ与えられた固定長から算出可能な範囲に収まらない場合には、当該文字列は変換不可能であると判断する。その場合、本手法では直ちに変換処理を中断する方式を採用している。

3.5 スケジューラ

電源供給時のセットアップで、3.4.1 に成功した時、サーバーから受け取った KIC-data の送信時刻と日付、ハードの時刻と日付を同期する(以降、マシン時間と呼称)。CPU 内時間で 1 分経過したら、マシン時間を 1 分に相当する分だけ増加させる。現在のマシン時間の曜日において、KIC-data のスケジュールに格納されているすべての(時間, 分)とマシン時間を比較して等しくなれば KIC-data からボードサイズを取得して清掃を開始する。

3.6 清掃運動

清掃運動を図解したものを図 4 に示す。

3.6.1 清掃開始の準備

サーバーから受け取った KIC-data を解釈して
清掃対象のボードサイズを取得する。マシン本体の物理サイズから、「現在の位置から、上下方向に何 cm 移動できるか」「現在の位置から、左/右に何㎝移動できるか」を計算する。

3.6.2 現在の位置状態

マシンには 2 つの状態フラグがある。1 つ目は、現在位置から、右方向に対して清掃を行うべきかどうかを示すフラグである。2 つ目は、マシンがボードの上側に位置しているか、下側に位置しているかを示すフラグである。
これら 2 つのフラグの組み合わせによって、マシンの清掃状態は 4 通りに分かれる。

・右方向へスライドし、かつボードの上側にいる場合は、図 4 に示す無反転の状態となる。
・左方向へスライドし、かつボードの上側にいる場合は、図 4 を左右反転した状態となる。
・右方向へスライドし、かつボードの下側にいる場合は、図 4 を上下反転した状態となる。
・左方向へスライドし、かつボードの下側にいる場合は、図 4 を左右上下反転した状態となる。

3.6.3 清掃の基本パターン

i.上下方向の清掃往復

図 4 における①、④、⑦に対応し、単純に前進と後進のどちらかを表している。

ii.横方向のスライド

i が終了し、次の列を清掃するために横へ移動する。スライド動作は複数のステップに分かれている。

3.6.1 で計算した「現在の位置から、左/右に何㎝移動できるか」この値がスライドによって発生する横方向の移動量を超えているなら 3.6.4 へ移行する。そうでないなら、まず、機体の後進方向が移動したい列に向くように、キャタピラが外れない角度分、機体を旋回させる。そして、機体を後進させ隣の列へ移動させる、ここまでが図 4 の②に値する。

次に、先ほどの斜め後進で上がってしまった重力方向の移動分を打ち消すため、前進する、ここまでが図 4 の③に値する。

これら一連のスライド動作が終わると、3.6.1 で計算した「現在の位置から、左/右に何㎝移動できるか」この値から横方向の移動量の分、減算する。

次に、後進をする。これは i に対応していて、i が終了すると、図 4 の⑤、⑥のように、移動方向や回転方向が反転するだけで先ほどと同様の事を繰り返す。

この「往復→スライド→往復」を繰り返して清掃エリアを全て埋めていく。

3.6.4 清掃終了

次の設定された清掃時刻となるまで待機。

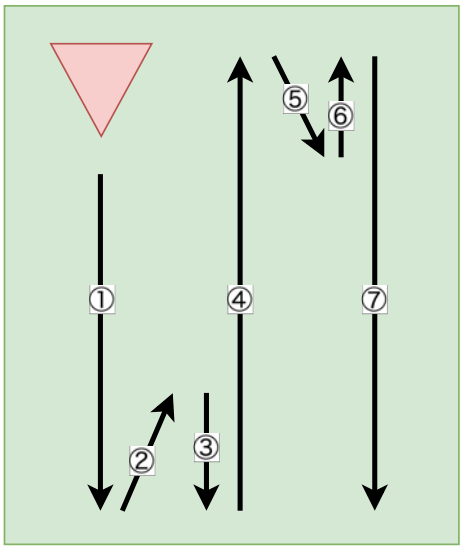

図 4　清掃のルーティング

3.7 通信用独自フォーマット

3.7.1 設計思想

IoT デバイス、特にバッテリー駆動のマイコンを用いたシステムにおいて、通信オーバーヘッドとデータ処理負荷の低減は重要な課題である。一般的によく用いられる JSON 形式は、可読性が高い反面、冗長なキー情報を含み、パース処理において動的なメモリ確保(ヒープ領域の使用)を多用するため、リソースの限られた ESP32 等のマイコンではメモリ断片化や処理遅延のリスクがある。そこで本研究では、HTTP 上で動作する軽量な独自フォーマット KICoH (KIC over HTTP) を策定した。設計においては以下の 4 点を要件とした。

i. **帯域幅の節約:** 冗長なメタデータを排除し、必要最小限のバイト数で通信を行う。
ii. **通信の確実性:** 不安定な Wi-Fi 環境下でも確実にデータを届けられる。
iii. **解読の容易性:** 文字列を機械的に解析できるような区切り方、意味付けをする。
iv. **処理の軽量性:** 文字列操作のみで解析可能とし、CPU 負荷とメモリ消費を最小化する。

3.7.2 フォーマットの仕様

KICoH のデータパケットは、可変長のテキストデータであり、複数のフィールドが区切り文字によって結合された構造を持つ。

各フィールドはセミコロン(;)で区切られ、パケットの終端はスラッシュ(/)で示される。これにより、受信側はストリームデータの中から容易にパケットの境界を検出可能である。各フィールドの詳細な仕様を表 1 に示す。

表 1　KICoH の仕様

順序	フィールド名	データ長 (Byte)	データ型	内容・形式
1	Header	6	String	フォーマット識別子とバージョン。 例: KIC:V3
2	Timestamp	5	Integer	サーバー送信時の曜日と時刻。 形式: DHHMM(D:曜日 0-6, HH:時, MM:分)
3	BoardSize	8	Integer	黒板の物理的寸法。 形式: HHHHWWWW (縦 4 桁+横 4 桁, 単位:cm)
4	Schedule	可変長	String	清掃スケジュールのリスト。 形式: DHHMM の連続 (例: 00800=日曜 8:00)
5	Terminator	1	Char	終端記号。固定値: /

※ 曜日の定義: 0=Sunday, 1=Monday, ... , 6=Saturday

3.7.3 実装例とデータ効率

実際の通信データの例を以下に示す。

受信データ:

```
KIC:V3;01437;01140334;008001200;20700090011001330;/
```

このデータは、「バージョン 3、日曜 14:37 送信、黒板サイズ 114×334cm」であり、スケジュールとして「日曜 8:00, 12:00」および「火曜 7:00, 9:00, 11:00, 13:30」を含んでいる。

同等の情報を JSON で記述した場合、一般的な JSON 記述で約 350 バイト程度、キー名を 1 文字に短縮するなどの最適化を施した場合で約 150 バイト(キー名や括弧を含む)を要するが、KIC-data では約 55 バイトで表現可能であり、データ量を最大で約 84%削減している。また、パース処理においては、特定の区切り文字(セミコロン)を探索するだけの線形探索($O(n)$)で実装可能であり、マイコンへの負荷を極小化している。

3.8 サーバー構成

3.8.1 システム構成

システム構成図を図 5 に示す。

Raspberry Pi 5 に OS を Ubuntu24 LTS でサーバー兼 Wi-Fi ルーターとして用いた。

3.8.2 技術スタック

サーバーはバックエンドを Node.js+Express で、フロントエンドを JavaScript で記述し、DB には SQLite3 を用いた[7][8]。

Wi-Fi ルーターは netplan と hostapd, dnsmasq を用いた。ネットワーク設定を Ubuntu 標準の netplan で記述し、

hostapd は Linux で簡単な config を記述するだけでコンピューターをルータとすることができるものである。dnsmasq はローカルホストへの DNS として機能させた。

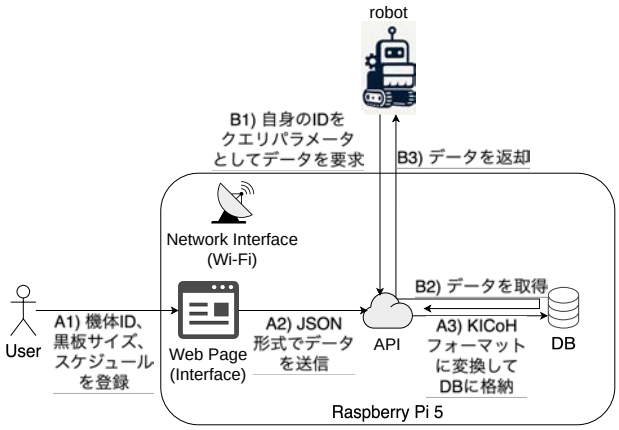

図 5　システム構成図

3.8.3 閉域ネットワークの構築

本システムでは、Raspberry Pi 5 を単なる Web サーバーとしてだけでなく、Wi-Fi アクセスポイント(AP)としても機能させている。一般的な教育機関の無線 LAN 環境(Eduroam 等)は、WPA2-Enterprise などの高度な認証方式を採用している場合が多く、ESP32 などの組み込みマイコンからの接続には証明書の管理やメモリ制約の観点で課題が残る。また、学内ネットワークのポート制限により、自作デバイス間の通信が遮断されるリスクもある。　そこで本機では、hostapd により独自の SSID を送出し、dnsmasq により DHCP および DNS サーバー機能を動作させることで、外部ネットワークに依存しない閉じたローカルネットワーク(閉域網)を構築した。これにより、設置場所のネットワーク環境に左右されず、電源を入れるだけでロボットとサーバーが即座に接続可能な Plug and Play 環境を実現している。

3.8.4 データ変換と保存プロセス

サーバーの主要な役割の一つは、ユーザーフレンドリーな JSON 形式と、通信効率を重視した KICoH に適合する形式への相互変換である。ユーザーが Web インタフェース上で入力したスケジュールや黒板サイズは、標準的な JSON 形式でバックエンドに送信される。バックエンド(Express)はこれを受け取り、前述の「3.7 通信用独自フォーマット」の仕様に基づいたセミコロン区切りの文字列へと変換を行う。変換された文字列は SQLite3 データベースに格納される。この際、JSON のまま保存せず、送信直前の KIC-data 型でキャッシュしておくことで、ロボットからのアクセス時(GET リクエスト)におけるサーバー側の計算負荷を低減し、レスポンスの高速化を図っている。また、黒板サイズとスケジュール情報を別テーブルとして設計したことで、初回に黒板の大きさを登録した後の新規スケジュール登録の際、黒板のサイズあるいはスケジュールを省略したとしても問題が起こらない設計となっている。

3.8.5 通信シーケンス

ロボット(クライアント)とサーバー間の通信は、HTTP プロトコル上の単純なポーリングによって行われる。ロボットは起動時および定期的なタイミングで、config.hpp に SSID などとともに記述された固有 ID をクエリパラメータに含め、`<ipaddr>:<port>/kic-data?id=<MachineID>`へリクエストを送信する。サーバーは該当する ID の KICoH データをレスポンスボディとしてプレーンテキストで返却する。

3.8.6 フロントエンド

フロントエンドには html, css, JavaScript が用いられており、図 6 のように機体の清掃活動に必要なデータの登録の他、本システムの特徴や利点を紹介するページがフレキシブル対応で用意されている。

データ登録のフローではユーザーの入力フィールドに対し型チェックや必須項目のチェックをユーザーに対して明示するようになっていることでユーザーフレンドリーを実現している。また、ここで入力されたデータは JSON 形式に統合されて API へ送信される。

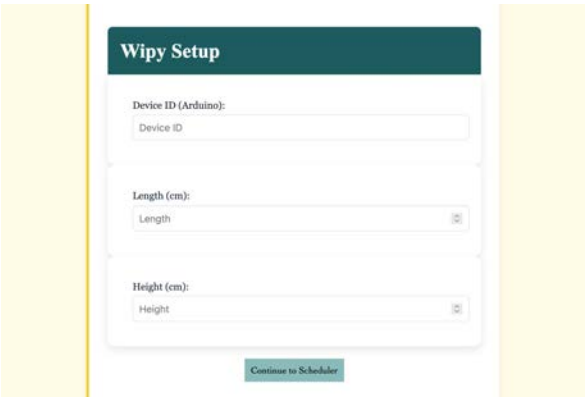

図 6　スケジュール登録ページ

4. 結果

本研究は従来の黒板消しロボットとの最大の違いとしてスケジュールによる自立清掃機能が挙げられる。しかし、この精度が悪ければ機能の効果があるとは言えない。そこでサーバーに登録されたスケジュールとマシンが実際に動き出したときのサーバー時刻との差を計測することとした。なお、マイコンの内部時計はサーバーから送られた時刻を基にするが、KICoH の仕様上、秒を除いたサーバー時刻を元に時刻のカウントを開始するため、理論上最大で 59 秒の遅れが発生する。

具体的には、1 回目の動作時刻では正常に時刻を受け取り解釈することができているかを検証し、2 回目と 3 回目の動作時刻でマイコンの内部時計の精度を検証する。これらを 1 セットとし、計 4 セット計測する。

検証の結果は表 2 に示す。1 回目の動作では、スケジュールに登録した時刻と初回の実際に動作した時刻が最大でも 59 秒と理論値内に収まっている。また、2 回目以降の動作開始時刻を見ると初回から 1 秒以上のズレが無いことが

分かる。このことからこのシステムは十分に機能を果たす能力があると言えると考える。

表 2　設定時刻と実際の実行時の時刻

1st set

time	result
19:50	19:50:06
19:54	19:54:06
19:58	19:58:06

2nd set

time	result
20:05	20:05:50
20:09	20:09:50
20:13	20:13:50

3rd set

time	result
16:40	16:40:51
16:44	16:44:51
16:48	16:48:51

4th set

time	result
16:52	16:52:59
16:56	16:56:59
17:00	17:00:59

5. 今後の課題

本研究で開発した自律型黒板清掃ロボットおよび通信フォーマットは、一定の有用性を示したが、実運用に向けて解決すべきいくつかの課題が明らかになった。

第一に、移動機構の信頼性と自由度の向上である。採用したクローラ(キャタピラ)機構は、斜め移動や旋回時の負荷により履帯が脱落するリスクがあるほか、構造上、真横への並進移動が不可能である。これにより、清掃経路の生成に制約が生じている。今後は、全方向移動が可能なオムニホイールの採用や、磁石ではなく吸盤を採用する[9]などの吸着機構と移動機構の統合的な見直しを行うことでより柔軟かつ堅牢な移動システムを検討する必要がある。

第二に、運用負担の更なる軽減とフォーマットの最適化である。現状では、清掃対象となる黒板の寸法を利用者が手動で計測・入力する必要があり、設置時の負担となっている。超音波センサや ToF(Time of Flight)センサを搭載し、黒板の端点を自律的に検知・計測する自動キャリブレーション機能の実装が望まれる。また、独自フォーマット KICoH については、通信安定性は確保できたものの、多数の機体を同時接続する環境を想定し、データペイロードのさらなる圧縮やバイナリ形式の検討など、通信負荷を極小化するフォーマットへの改良を継続する。

参考文献

[1] 文部科学省: "令和 4 年度教員勤務実態調査"
https://www.mext.go.jp/a_menu/shotou/hatarakikata/1417539_00004.htm
(参照 2025/12/12)

[2] 帯広畜産大学 大学教育センター: "授業における板書・プレゼンテーションの留意点"
https://www.obihiro.ac.jp/edu-affairs-presentation
(参照 2025/12/12)

[3] S. Vignesh, K. Vinith, K. Mouleswaran, H. M. Sanjay, M. Logeshwaran: "A Review On Automatic Blackboard Cleaner", International Journal of Research in Engineering, Science and Management (IJRESM), Vol. 4, No. 4, pp. 103-106, 2021.
(参照 2025/12/12)

[4] 特許: "黒板消去装置及び自動消去機能を備える黒板", 特開 2010-005865 (JP2010005865A).(参照 2025/12/12)

[5] Gugen: "黒板消しロボット KESPIE",
https://gugen.jp/subscriptions/work/871
(参照 2025/12/12).

[6] swiper: "swiperWHITE: World's first automatic whiteboard cleaner", YouTube, 2019
(https://youtu.be/srbTaPVX19o?si=58Rdj4askc2l8gSf)
(参照 2025/12/12)

[7]本研究の ESP32(マイコン)の制御ソースコード(GitHub)
https://github.com/Tomarun029831/BlackBoardCleaner
(参照 2025/12/12)

[8]本研究のサーバープログラムソースコード(GitHub)
https://github.com/sg-tera/wipyserver
(参照 2025/12/12)

[9] Yu Yoshida, Shugen Ma: "Design of a wall-climbing robot with passive suction cups", 2010 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1513-1518, 2010.
(参照 2025/12/12)