1　検証実験

1.1　実験方法

1.1.1　ゲートウェイの LTE モジュールとの疎通実験

ゲートウェイが使用する LTE 通信モジュールとクラウドサーバ間での POST 形式での通信が可能か確認する．

実験手順としてはゲートウェイに搭載予定の LTE モジュールから Internet を経由させて用意したサーバにデータを送信し，通信を確認する．LTE モジュールから送信は POST 形式で HTTP 通信を用いて行う．また，送るデータは JSON 形式で送信し，データ内容としては 表 1.1 に示す．

表 1.1: LTE 通信する JSON データ

データ名	データフォーマット	注釈
timestamp	文字列 (YYYYMMDDHHMMSSSSS)	検出時刻 (年月日, 時分秒 [ms])
node	数値	センサノード番号
dir	数値	ノード内のセンサ番号
sd	(任意長)	センサデータ (サンプリングデータ)

1.1.2　通知サーバの WebPush 実験

クラウドサーバの構成要素である通知サーバ (Notification-app) から WebPush による通知ができるかどうかを確認する．

実験手順としては以下の通りである．

1. Docker Compose でサーバを起動する．
2. ブラウザアプリで起動したサーバにアクセスする．
3. アクセス時に通知を受け取るかの確認が出るため，それを許可する．
4. サーバ側から WebPush で Push 通知を送る．
5. ブラウザアプリを通じて通知がくるかを確認する．

1.1.3 Web アプリケーションの行動履歴表示実験

クラウドサーバの構成要素である Web アプリケーション (Visualizer-app) の行動履歴表示ができるかを確認する．

実験内容としては，ダミーで用意した畑の位置情報，センサノード位置情報，経路情報について正しく表示できているかを Web ページにアクセスして確認する．ダミーで用意したデータについては 表 1.2 に示す．

表 1.2: ダミーデータ

データ種類	緯度	経度	注釈
畑位置	34.64801074823137	135.75705349445346	奈良高専の校庭
センサ 1	34.64746793595177	135.75796544551852	畑の南東
センサ 2	34.64801074823137	135.75796544551852	畑の東
センサ 3	34.64852707856505	135.75796544551852	畑の北東
センサ 4	34.64852707856505	135.75705349445346	畑の北
センサ 5	34.64852707856505	135.75623810291293	畑の北西
センサ 6	34.64801074823137	135.75623810291293	畑の西
センサ 7	34.64746793595177	135.75623810291293	畑の南西
センサ 8	34.64746793595177	135.75705349445346	畑の南
経路情報	34.64879186210419	135.75744509696963	1 番目の位置
	34.64826229418025	135.75744509696963	2 番目の位置
	34.64828435957796	135.75665116310122	3 番目の位置
	34.64780333257661	135.75666189193728	4 番目の位置
	34.64724286640332	135.75664043426517	5 番目の位置

1.2 実験結果

1.1.1 ゲートウェイの LTE モジュールとの疎通実験

ゲートウェイとの LTE 通信を受信した結果を 図 1.1 と 図 1.2 に示す．

```
--- server up ---
{"timestamp":"20240123135017200","node":"1","dir":"1","sd":"1234567890"}
```

図 1.1: サーバの受信ログ

図 1.2: サーバの受信データ (body 情報の一部抜粋)

図1.1 にあるように,「{“timestamp”:“20240123135017200”,“node”:“1”,“dir”:“1”,“sd”:“1234567890”}」というデータを受け取ったことがわかる．また， 図 1.2 より「methods: { post: true }」のように POST 形式で送られていることがわかる．加えて，「[Symbol(kHesaders)]」の項目から「content-length」は 54,「content-type」は ‘application/x-www-form-urlencoded’ であることがわかる．(「host」の伏せてある部分はクラウドサーバのホスト名である．)

「content-length」は body の波括弧 ({}) と空白スペース，クォーテーションを除いた数である 54 文字と一致するため，body のデータサイズと考えられる．また,「content-type」の ‘application/x-www-form-urlencoded’ は HTML フォームでデータを送信した時と同じタイプであるため [11]，ゲートウェイからの通信は HTML フォームによるアクセスと同義に扱えると考えられる．

1.2.2　通知サーバの WebPush 実験

通知サーバの通知結果を 図 1.3 と 図 1.4　に示す．今回の実験では PC 版の Google Chrome(バージョン:120.0.6099.234（Official Build）　（x86_64）) をブラウザとして利用した．

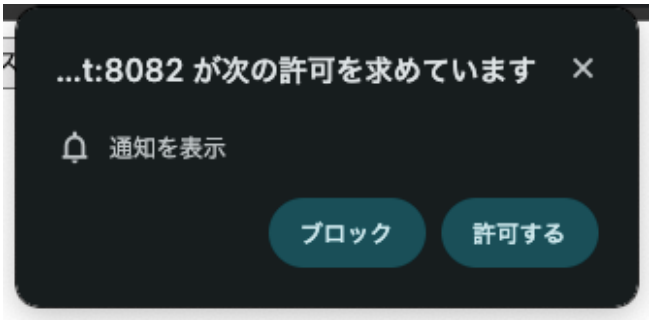

図 1.3: WebPush の通知許可への確認ダイアログ

図 1.4: WebPush の通知

図 1.3 は WebPush を実施するアプリケーションサーバにアクセスした時のダイアログである．図 1.3 の選択肢で「許可する」を選択することで，WebPush の通知が受け取れるようになる．「許可する」を選択後，サーバに WebPush 送信用のメソッドを起動するようにシグナルを送ると，図 1.4 がデスクトップに表示された．図 1.4 の通り，Google Chrome からの通知であることがわかり，Push 通知をクリックすると Google Chrome に遷移させられたため，Google Chrome による Push 通知だと判断できる．

1.2.3　Web アプリケーションの行動履歴表示実験

Web アプリケーションの行動履歴表示について 表 1.2 を表示した Web ページを 図 1.5 に示す．

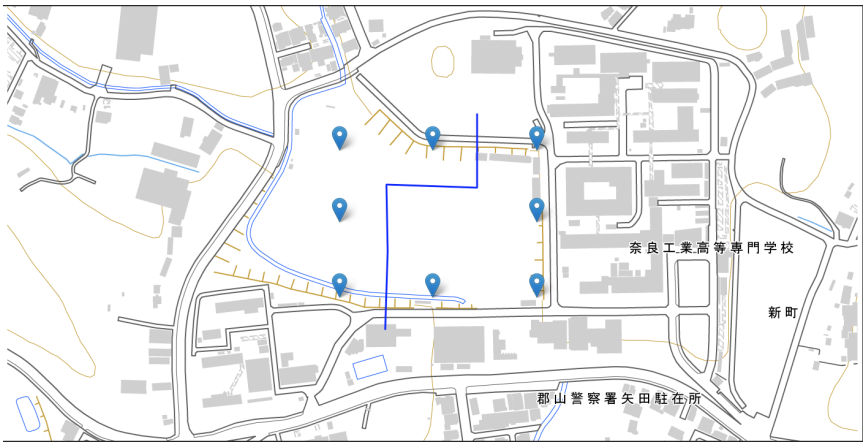

図 1.5: Web アプリケーションの行動履歴表示 (ダミーデータ)

図 1.5 より，奈良高専の校庭を中心とした 8 つのセンサノード (ピン) と，行動経路 (青い折れ線) が描画されていることがわかる．そのため正しくダミーデータ (表 1.2) 通りの描画が行えていると判断できる．

1.3 まとめ

本章の検証実験により，ゲートウェイとの疎通確認および通信方法の特定，通知サーバによる WebPush の通知の確認，Web アプリケーションによる行動履歴表示の確認が行うことができた．

本論文ではできなかったがシステムとして効果を確認するためには，赤外線センサノードおよびゲートウェイを実際の畑に設置し，動物が畑に侵入した時の実際のセンサデータを取得，クラウドサーバへ送信したのち，行動解析アルゴリズムを用いて侵入した動物の行動を推定，そうして得られたデータから PC やスマートフォンに対して WebPush による通知，ブラウザを通した行動履歴の表示する，という一連の流れを検証する必要がある．

参考文献

[1] 農林水産省, “野生鳥獣による農作物被害状況の推移”, https://www.maff.go.jp/j/seisan/tyozyu/higai/hogai_zyoukyou/attach/pdf/index-31.pdf, 2024年1月18日参照.

[2] 大澤文孝, 浅井尚, “触って学ぶクラウドインフラ docker 基礎からのコンテナ構築”, 日経BPマーケティング, 2020年.

[3] 掌田津耶乃, “Node.js 超入門”, 株式会社 秀和システム, 2017年.

[4] StrongLoop, IBM, “Express - Node.js web application framework”, https://expressjs.com/, 2024年1月18日参照.

[5] Volodymyr Agafonkin, “Leaflet - a JavaScript library for interactive maps”, https://leafletjs.com, 2024年1月18日参照.

[6] Mozilla Foundation, “プッシュ API - Web API | MDN”, https://developer.mozilla.org/ja/docs/Web/API/Push_API, 2024年1月18日参照.

[7] M. Thomson, E. Damaggio, B. Raymor, Ed., “Generic Event Delivery Using HTTP Push”, https://datatracker.ietf.org/doc/html/rfc8030, RFC8030, December 2016, 2024年1月21日参照.

[8] Mozilla Foundation, “サービスワーカー API - Web API | MDN”, https://developer.mozilla.org/ja/docs/Web/API/Service_Worker_API, 2024年1月18日参照.

[9] M. Thomson, P. Beverloo, “Voluntary Application Server Identification (VAPID)”, https://datatracker.ietf.org/doc/html/rfc8292, RFC8292, November 2017, 2024年1月21日参照.

[10] Nginx, “nginx”, https://nginx.org/en/, 2024年1月18日参照.

[11] Mozilla Foundation, “POST - HTTP | MDN”, https://developer.mozilla.org/ja/docs/Web/HTTP/Methods/POST, 2024年1月22日参照.