卒 業 研 究 報 告 書

令和5年度

研究題目

赤外線センサを用いた害獣検出および行動解析
– 通知機構と行動ビジュアライザ –

指導教員　　山口 賢一 教授

アドバイザー　　岩田 大志 准教授

氏　　名　　藤本 光

令和6年1月23日提出

奈 良 工 業 高 等 専 門 学 校　　情 報 工 学 科

要約

自然を相手にする農業従事者が抱える問題として獣害がある．獣害被害の中でも夜間に発生するものは，夜間であるが故に視界が悪いことや，農業従事者が自分の身で直接の対策が困難である．これまでの対策としては，荒らされた畑や食べられた農作物の痕跡から動物を判断し，電気柵やネットなどの設置という，動物やその行動に合った対策を対処的に行われている．しかし，柵に隙間がある場合や，そもそも想定していた動物でない，対処的な対策故に原因を特定し適切な対策を取れるまでに大きな手間と時間・費用を要している．そこで我々はこれらの問題を解決するために，畑への侵入を検出・通知し，その動物の行動や大きさを推定することで，農業従事者の獣害対策を効果的に行えるよう支援するシステムを提案する．特に本論文ではクラウドサーバを用いて，畑への侵入したという情報および推定して得られた結果を保存するデータベース，畑への侵入情報から農業従事者へ通知する Web アプリケーション，推定された行動や大きさを地図を用いて行動履歴を表示する Web アプリケーションの設計を行った．機能としては，リバースプロキシ，WebPush を用いた通知サーバ，Leaflet.js による地図を用いた行動履歴の表示 Web アプリケーション，データベースおよびデータベースアクセス制御用の内部 API があり，それぞれを Docker Compose を駆使して連携させたクラウドサーバを設計した．設計・実装後，クラウドサーバがやり取りする畑のゲートウェイに搭載される無線モジュールとの疎通確認および通信形式の確認，クラウドサーバ内の通知サーバからブラウザへの WebPush 通知の確認，クラウドサーバ内の Web アプリケーションによる行動履歴表示の確認を行った．しかし，まだシステム全体を連携させて，実際の畑で得られたセンサーデータで解析，通知・表示を行うことができていないため，実際の効果を確認するためには行わなければならない．展望では，本論文担当であるクラウドサーバについて，セキュリティや UI・UX の向上の他に，リバースプロキシとデータベースアクセス制御用の内部 API が通信量のボトルネックとなる懸念点が挙げられる．通信量のボトルネックは，本論文のシステム構成ではリバースプロキシと内部 API はデータが集中するのに全体で一つしか用意していないことが根本的な原因である．そのため，kubernetes といった分散管理システムの導入することで，更なる冗長性とスケーラビリティが実現することができる．

目 次

1 まえがき **1**

2 理論 **2**

2.1 Docker 2
2.1.1 Docker イメージ 2
2.1.2 Dockerfile 2
2.1.3 Docker ネットワーク 3
2.1.4 Docker Compose 3
2.2 Node.js 4
2.2.1 npm 4
2.2.2 Express 4
2.3 Leaflet.js 4
2.4 WebPush 5
2.4.1 サービスワーカー (Service Worker) 5
2.4.2 VAPID(Voluntary Application Server Identification) . . . 6
2.5 Nginx 6

3 システム構成 **7**

3.1 全体構成 7
3.2 サーバ構成 7
3.2.1 リバースプロキシ (Reverse Proxy) 10
3.2.2 通知サーバ (Notification-app) 10
3.2.3 Web アプリ (Visualizer-app) 10
3.2.4 データベース (Database, Database-API) 11
3.2.5 データ解析ノード (Analyzer-node) 11

4 検証実験 **12**

4.1 実験方法 12
4.1.1 ゲートウェイの LTE モジュールとの疎通実験 12
4.1.2 通知サーバの WebPush 通知実験 12
4.1.3 Web アプリの行動履歴表示実験 13
4.2 実験結果 13
4.2.1 ゲートウェイの LTE モジュールとの疎通実験 13

4.2.2　通知サーバの WebPush 通知実験 15
4.2.3　Web アプリの行動履歴表示実験 16
4.3　まとめ 16

5　あとがき **17**
5.1　セキュリティ対策 17
5.2　UI・UX の向上 17
5.3　サーバの冗長性 18

謝辞 **19**

参考文献 **20**

1　まえがき

自然を相手にする農業従事者が抱える問題として獣害がある．令和 4 年度の野生鳥獣による全国の農作物被害は約 156 億円で，特にシカの被害額は約 65 億円で前年の被害額と比べて増加傾向にある [1]．獣害被害の中でも夜間に発生するものは，夜間であるが故に視界が悪いことや，農業従事者が自分の身で直接の対策が困難である．これまでの対策としては，荒らされた畑や食べられた農作物の痕跡から動物を判断し，電気柵やネットなどの設置という，動物やその行動に合った対策を対処的に行われている．しかし，柵に隙間があったり，そもそも想定していた動物でない，対処的な対策故に原因を特定し適切な対策を取れるまでに大きな手間と時間・費用を要している．

我々はこれらの問題を解決するために，畑への侵入を検出・通知し，その動物の行動や大きさを推定することで，農業従事者の獣害対策を効果的に行えるよう支援するシステムを提案する．具体的な流れとしては，畑の周囲に赤外線センサを搭載したセンサノードを規則的に配置する．赤外線センサノードが検出した侵入した動物のデータを LoRa 通信および LTE 通信を用いてクラウドサーバへ収集する．クラウドサーバでは収集されたセンサデータを用いて，農業従事者に通知したり，センサデータを解析し，解明した畑内に侵入した動物についての行動履歴を地図と合わせて農業従事者に提示する．

本論文では著者が担当するクラウドサーバの解析機構以外の機能群について述べる．クラウドサーバの著者担当部分は大きく分けて，データの保存，農業従事者への通知，行動履歴の表示の三つがある．データの保存では畑に設置した赤外線センサノードから得られたセンサデータ，センサデータから解析して得られた行動履歴のデータ，登録されている畑についてのデータについて保存し，管理する．農業従事者への通知では，センサデータの送信に応答して農業従事者の個人端末へ通知を行う Web アプリケーションを実現する．行動履歴の表示では，クラウドサーバ内に保存された行動履歴のデータを用いて農業従事者の個人端末で確認可能な Web アプリケーションを実現する．以上の三つの機構について設計・実装をし，動作確認を行う．

2　理論

本章では，本研究の前提となる知見や技術について説明する．

2.1　Docker

Docker はコンテナ技術の一種で，システムの環境ごとに隔離した空間を用意して独自のディレクトリツリーを提供する．コンテナはそれ単体でシステムが完結しているため，コンテナを別のサーバにコピーして起動させることも容易であり，ポータビリティ性を有している [2]．本論文では docker を用いて機能ごとにコンテナを作成する．

2.1.1　Docker イメージ

Docker イメージは Docker コンテナ作成を支援するために用意されたアーカイブパッケージである．Docker コンテナは独立したシステム実行環境であるため，一から構築するには OS やライブラリなど全てを作り込む必要があり非常に手間がかかる．Docker コンテナ作成に必要なファイルを纏めたアーカイブパッケージが Docker イメージである [2]．Docker イメージは Docker 社が運営している Docker Hub で登録・公開されており，Docker Hub に公開されている Docker イメージをダウンロードして Docker コンテナを作成することが可能である．

Docker イメージには 2 種類あり，

- 基本的な Linux ディストリビューションだけの Docker イメージ
- アプリケーション入りの Docker イメージ

がある．

本論文では後者のアプリケーション入りの Docker イメージを用いてシステムのコンテナを作成する．

2.1.2　Dockerfile

Dockerfile は自作のイメージを作るもので，ベースとなるイメージに対してどのような変更指示をするかをまとめたファイルである．Dockerfile は「docker build」コマンドでビルドすることで Docker イメージを作成できる [2].

2.1.3 Docker ネットワーク

Docker コンテナは独立しており，一つの Docker コンテナがサーバマシンのような役割を持っている．Docker コンテナ間が通信するために Docker 内に仮想的なネットワークを構築する必要がある．Docker には 3 種類のネットワークが用意されており，

- bridge ネットワーク
- host ネットワーク
- none ネットワーク

があるが，本論文で利用する bridge のみ説明し，他 2 種類については説明を省く．

bridge ネットワークでは IP マスカレードを利用してホスト PC が所属する新たな内部ネットワークを構築している． 図 1 に Apache のコンテナを 2 つ起動し，それぞれのポートをホスト PC の 8080 と 8081 に接続した時の例を示す．

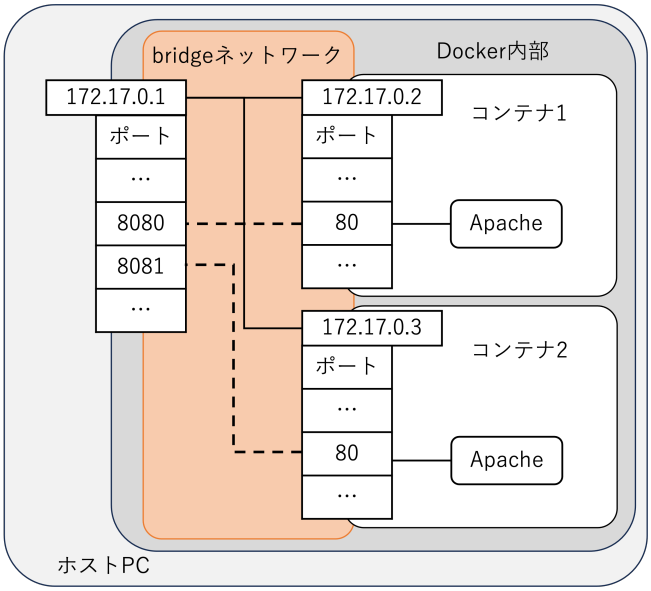

図 1: bridge ネットワーク例

2.1.4 Docker Compose

Docker Compose は複数の Docker コンテナの作成および Docker コンテナ間のネットワークの作成など，複数のコンテナの作成，停止，破棄の一連の操作をまと

クラウドサーバ？

めて実行する仕組みのことである [2]．一連のコンテナやシステムについての定義ファイル (Compose ファイル) は既定では「docker-compose.yml」というファイル名になっている．Compose ファイルを「docker-compose(docker compose)」コマンドで実行するとボリュームやネットワークが作成され，まとめてコンテナを起動でき，停止や削除するときも同様のコマンドで行うことができる．本論文で実装するクラウドサーバの一連のシステムは Docker Compose を用いて作成する．

2.2 Node.js

Node.js は JavaScript のランタイム環境であり，Web ブラウザの中でなく，単独で JavaScript のプログラムを実行できるようになっているものである [3]．ランタイム環境であることで，従来の JavaScript ではクライアント側の実装でしか使えなかったが，サーバ側でも JavaScript を利用でき，Web 開発の全てを JavaScript のみで完結することができるようになっている．Node.js は Web サーバのための機能を備えており，容易にサーバ側の実装を行うことができる．

2.2.1 npm

npm は Node.js 専用のパッケージマネージャの一種である [3]．Node.js では Node.js の機能を拡張する様々なプログラムがパッケージとして配布されている．その多種多様なパッケージをインストール・管理するシステムとして npm が用意されている．

2.2.2 Express

Expressは Node.js のフレームワークの一種で，Node.js の機能をわかりやすくし基本的な Web アプリケーション機能を比較的軽量で提供~~してくれるものである~~できる [3]．

2.3 Leaflet.js

Leaflet.js は Volodymyr Agafonkin 氏によって作成された JavaScript のマップライブラリである．マーカの表示やポップアップ，地図の拡大縮小など基本的な地図アプリの機能を提供しており，Edge や Chrome，Firefox，Safari などの基本的なブラウザをサポートしている [5]．

本論文では畑の地図表示および害獣の行動経路を線として，センサノードの位置をマーカとして図示することに用いる．

2.4 WebPush

WebPush はWeb アプリケーションがサーバからメッセージを受信できる仕組みのことを指す．Web アプリケーションがフォアグランド状態や読み込まれているかどうかに関わらず，リアルタイムな通知を提供することができる．Web アプリケーションがプッシュ通知メッセージを受信するには「サービスワーカー」が起動している必要がある [6].

WebPush の一般的なモデルを 図 2 に示す．WebPush はまず利用者が Push サービスに登録を行い，同時にアプリケーションサーバに登録情報を共有する．アプリケーションサーバは共有された登録情報を用いて Push メッセージを Push サーバを経由して利用者に Push メッセージを送り，Push 通知を実現する [7].

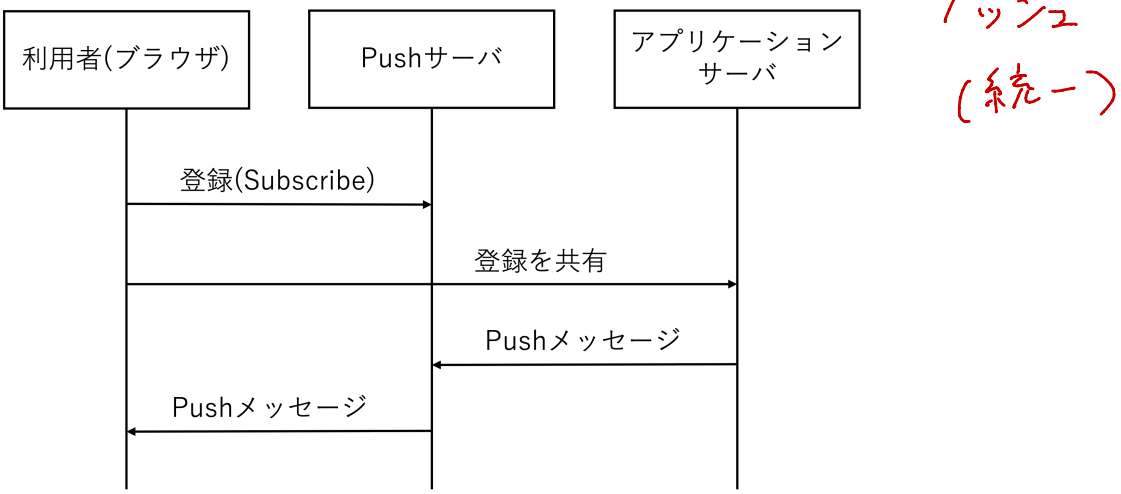

図 2: WebPush の模式図

WebPush は基本的なブラウザで保証されていて， 表 1 に示すブラウザでは保証が確認されている [6].

2.4.1 サービスワーカー (Service Worker)

サービスワーカーはあるオリジン (プロトコルとポート番号，ホストの組) とパスに対して登録されたイベント駆動型の JavaScript ファイルである [8].

表 1: WebPush が保証されているブラウザ [6]

パソコン	スマートフォン
Chrome	Chrome Android
Firefox	Firefox for Android
Opera	Opera Android
Safari	Safari on iOS
Edge	Samsung Internet

2.4.2 VAPID(Voluntary Application Server Identification)

VAPID はアプリケーションサーバを Push サービスに認証, 識別してもらうための仕組みである [9]．RFC8030[7] に記載されている WebPush にはアプリケーションサーバを認証する構造が含まれておらず，アプリケーションサーバになりすました通信が行われる問題点がある．ここで VAPID によってアプリケーションサーバの識別をすることで，アプリケーションサーバになりすました通信を防ぐことができるようになる．

2.5 Nginx

Nginx は元々 Igor Sysoev 氏が作成した HTTP サーバやリバースプロキシサーバ，メールプロキシサーバといった汎用的な TCP/UDP プロキシサーバである [10]．本論文においてはクラウドサーバ内のリバースプロキシサーバとして活用する．

3　システム構成

3.1　全体構成

本研究で作成するシステム全体のシステム図を 図 3 に示す．まず，畑内に設置した赤外線センサノード群で動物の侵入を検知し，そのデータをワイヤレス通信の LoRa 通信を用いてゲートウェイに集約する．ゲートウェイは集約されたデータを LTE 通信を用いて Internet を経由してクラウドサーバにデータを送信する．クラウドサーバではデータの管理やデータの解析，農業従事者への通知および解析結果の表示を行う．本論文ではクラウドサーバの解析アルゴリズム以外を担うため，次節でクラウドサーバの構成と詳細について述べる．

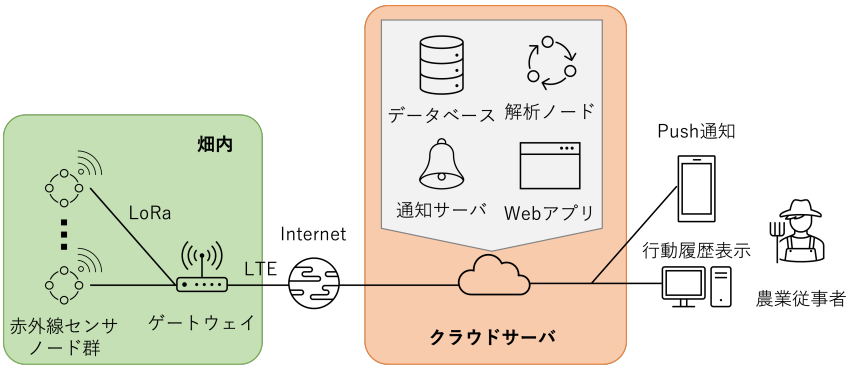

図 3: 全体のシステム図

3.2　サーバ構成

本システムのクラウドサーバは 図 3 にある通り，以下の 4 つの役割の担い，それぞれの役割が協調しながら処理を行う．

- データベース
- 解析ノード
- 通知サーバ
- Web アプリ

クラウドサーバの処理を説明するために DFD(Data Flow Diagram) を 図 4 と 図 5 に示す．クラウドサーバの処理に大きく関係するファクタとしてゲート

ウェイと農業従事者，データベースがある．ゲートウェイからは農場で得られた赤外線センサのデータがクラウドサーバへ一方的に送信されてくる．農業従事者については畑の情報や認証用の本人情報などを含んだ登録者情報を受け取り，畑に動物の侵入があったことを知らせる Push 通知とセンサデータを解析した結果得られた行動軌跡についての情報をまとめた行動履歴を提供する．データベースには畑から得られたセンサデータおよび解析後データと農業従事者から受け取った登録者情報を保管する．

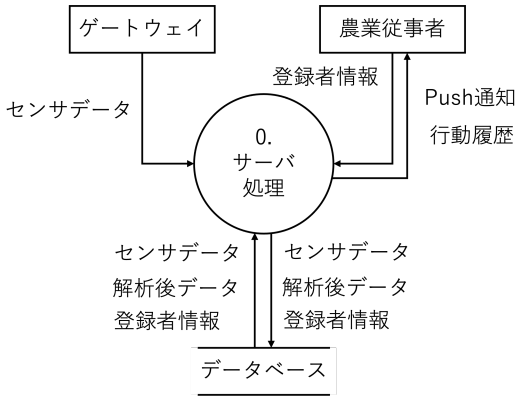

図 4: クラウドサーバの DFD(レベル 0)

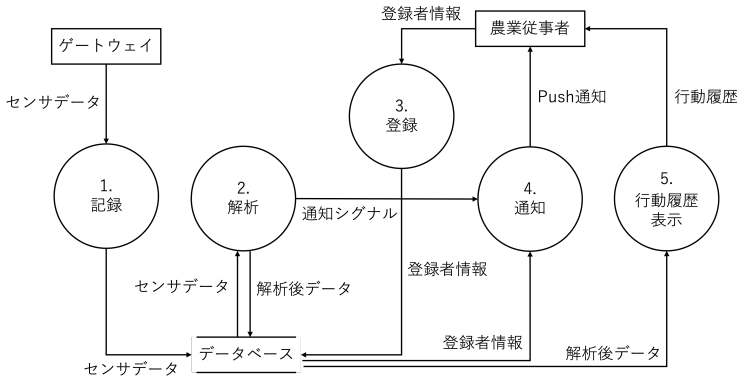

図 5: クラウドサーバの DFD(レベル 1)

本システムのクラウドサーバは文字通り，クラウド上にシステムを設置するため，開発環境との環境差を最小限にするために Docker を用いて開発を行う．また，クラウド内のシステムごとの連携を実現するために Docker Compose を用いる．

Docker Compose を用いて作成するクラウドサーバの内部ネットワークを 図 6 に示す．クラウドサーバ内部に作成する Docker コンテナは以下の 6 つである．また，Docker コンテナごとの詳細については後述する．

- リバースプロキシ (Reverse Proxy)
- 通知サーバ (Notification-app)
- Web アプリ (Visualizer-app)
- データベース
  - データベースのアクセス制御 (Database-API)
  - データベース本体 (Database)
- 解析ノード (Analyzer-node)

また，Docker ネットワークは以下の 3 つを用意する．

- surface
- inside
- db-bus

surface ネットワークでは Internet と接続する Docker コンテナのみ所属させる．inside ネットワークには基本的な Docker コンテナ全て所属させる．db-bus ネットワークにはデータベース管理の処理を担う Docker コンテナのみ所属させる．

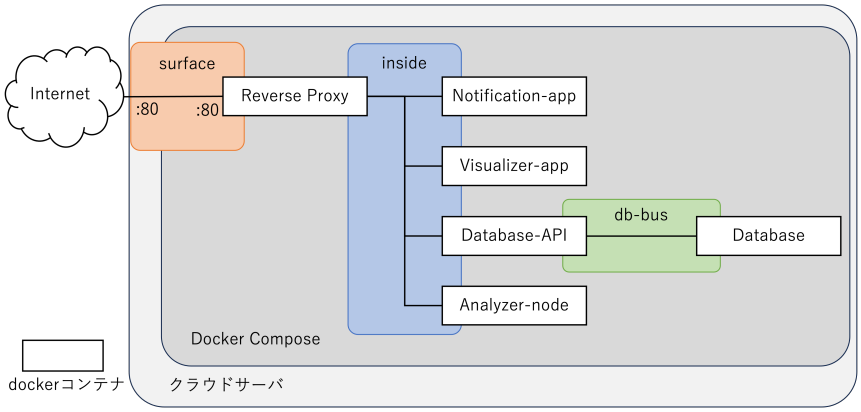

図 6: クラウドサーバのネットワーク図

3.2.1　リバースプロキシ (Reverse Proxy)

リバースプロキシではサーバにくる全てのデータアクセス (HTTP 通信) を役割ごとに適当な Docker コンテナに通信するように制御を行い，クラウドサーバの門番的な役割を担う．そのため，Docker ネットワークとしては surface ネットワークと inside ネットワークに所属する．

Nginx の Docker イメージを基本として/etc/nginx/nginx.conf を編集することによってリバースプロキシサーバとして docker コンテナを作成する．

本コンテナで行う転送処理は以下の通りである．

/notification： 通知サーバ (Notification-app)

/maps： Web アプリ (Visualizer-app)

/nodegw (POST)： データベース (Database-API)

3.2.2　通知サーバ (Notification-app)

通知サーバでは登録された畑に動物の侵入があったことを WebPush を用いて農業従事者に通知するサーバを実現する．WebPush としては VAPID を用いてサーバの登録を行い，Node.js の Docker イメージをベースとして WebPush を実現する．

通知サーバはレベル 1 の DFD(図 5) においては「4. 通知」を主に担う．そのため「2. 解析」からの通知シグナルと，データベースから登録者情報を取得するために Docker ネットワークで Analyzer-node および Database-API と同じ inside に所属する．

3.2.3　Web アプリ (Visualizer-app)

Web アプリでは畑の登録と解析データから得られた行動履歴を表示を実現する．Web アプリでは畑など地図の表示および図形の投影が必要なため，Node.js の Docker イメージをベースとして Leaflet.js を用いて実現する．本システムは複数の農家および畑を対象としてサービスを提供する．そのため，どこの畑が，誰が管理者であるかを登録する必要がある．登録ページについて開発中のページを 図 7 に示す．

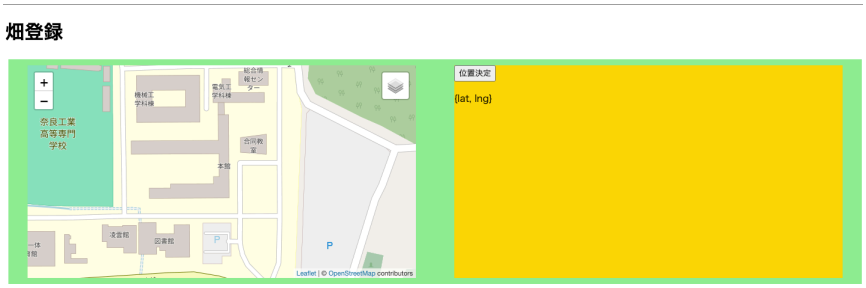

図 7: 開発中の登録ページ

Web アプリはレベル 1 の DFD(図 5) においては「3. 登録」と「5. 行動履歴表示」を主に担う．そのためデータベースへの登録者情報の保存および解析後データの取得するため，Docker ネットワークで Database-API と同じ inside に所属する．

3.2.4 データベース (Database, Database-API)

データベースでは MySQL を用いたデータベースコンテナと node.js を用いたデータベースコンテナ管理用の API コンテナの二つで構成する．データベースコンテナだけでなく API コンテナを作ることで，データベースアクセスをカプセル化して不正なアクセスを抑制している．

Docker ネットワークとしては db-bus ネットワークに所属して，他コンテナからのデータベースアクセスを受けるために Database-API コンテナのみ，inside ネットワークにも所属する．

3.2.5 データ解析ノード (Analyzer-node)

データ解析ノードでは python イメージを主体としてデータベースに蓄えられる赤外線センサのデータを解析する役割を持つ．本コンテナでは Database-API コンテナと通信してデータの取得および解析後のデータの保存を行い，通知サーバに通知シグナルを送信する．そのため，Docker ネットワークとしては inside ネットワークに所属する．

解析の詳細に関しては本論文範囲外であるため割愛する．

4　検証実験

4.1　実験方法

4.1.1　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 2 に示す．

表 2: 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)

4.1.2　通知サーバの WebPush 通知実験

クラウドサーバの構成要素である通知サーバ (Notification-app) による Web-Push 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，許可する．
4. サーバ側から WebPush 通知を送る．
5. ブラウザアプリを通じて通知が来るかを確認する．

4.1.3　Web アプリ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 (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 3 に示す．

表 3: 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

4.2　実験結果

4.2.1　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 8 と 図 9 に示す．

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

図 8: サーバの受信ログ

図 9: サーバの受信データ (body 情報の一部抜粋)

図8 にあるように,「{”timestamp”:”2024012212445004 ”,”node”:”1”,”dir”:”1”,”sd”:”1234567890”}」というデータを受け取ったことがわかる．(それ以外の「{”node”:”1”,”dir”:”1”,”sd”:”1234567890”}」と「{”timeStamp”:”Jikan”,”NodeID”:”ID”,”Direction”:”dir”,”sampledata”:”data dayo!”}」,「{”name”:”andesite”,”age”:”20”}」はサーバが正しく起動しているかどうかのチェックとして curl コマンドや Javascript を用いて送ったものである．) また， 図 9 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”tk2-101-53177.vs.sakura.ne.jp” はクラウドサーバとして利用したさくらインターネットの VPS のドメインである．)

「content-length」は body の波括弧 ({}) と文字列部分以外の空白スペース，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と考えて差し支えないと思われる．

4.2.2　通知サーバのWebPush通知実験

通知サーバの通知結果を 図10 と 図11　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．

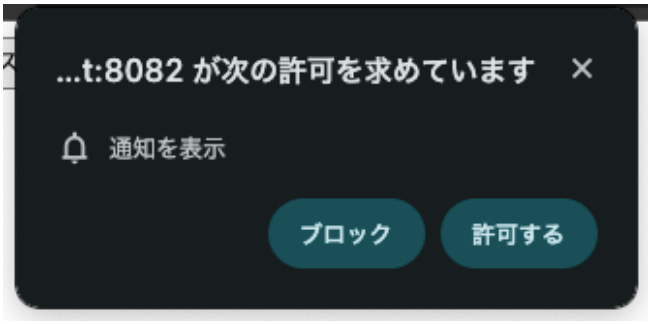

図 10: WebPushの通知許可への確認ダイアログ

図 11: WebPushの通知

図10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10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11がデスクトップに表示された． 図11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通知だと判断される．

4.2.3　Web アプリの行動履歴表示実験

Web アプリの行動履歴表示について 表 3 を表示した Web ページを 図 12 に示す.

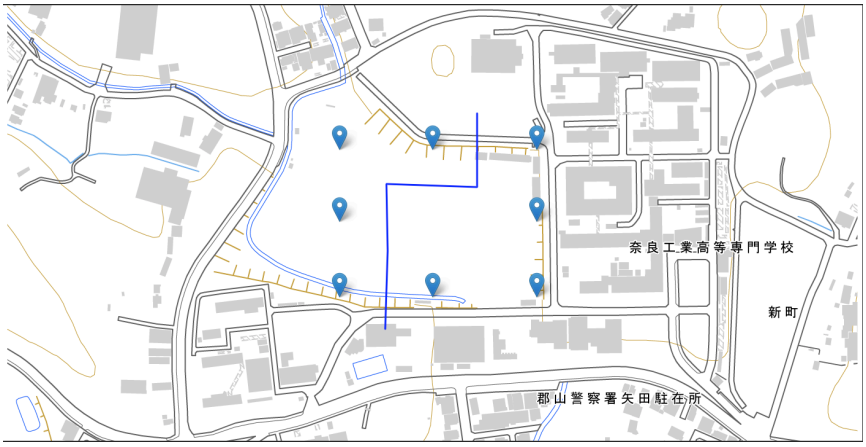

図 12: Web アプリの行動履歴表示 (ダミーデータ)

図 12 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 3) 通りの描画が行えていると判断される.

4.3　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害獣の行動を推定，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.

5　あとがき

本論文では，害獣検出と行動解析を行うシステムの中で，クラウドサーバに関する部分について設計，述べてきた．畑に設置予定のゲートウェイとのデータ通信や通知アプリの動作検証，Web アプリによる行動履歴の表示といった，クラウドサーバとして必要な要素についての設計・検証を行ってきたが，~~また~~まだ試験的な部分が大きく，実際に利用するには至らない部分が多い．特に優先的に憂慮しなければならないこととして，以下に示すことが挙げられる．

- セキュリティ対策
- UI・UX の向上
- サーバの冗長性

5.1　セキュリティ対策

クラウドサーバではゲートウェイからの受信に HTTP 通信を用いている．そのため，データ通信が十分に暗号化されておらず，通信中に盗み見られたり，悪意を持った攻撃者がデータを装って POST 送信をしてくる恐れが考えられる．これに関しては，ゲートウェイに搭載できるモジュールやマイコンの性能に左右されるため，ゲートウェイ設計担当である共同研究者とも話し合っていく必要があると考えられる．

また，クラウドサーバへのアクセスは基本的にクラウドサーバ自体のファイアウォールや Docker Compose 内のリバースプロキシ，利用するクラウドサービス独自のセキュリティシステムを考えている．ただ，本論文のシステムではクラウドサーバ内で，農業従事者の情報を入力し，登録するフォームを設けたり，それらの個人情報を取り扱うこととなるため，十分に堅牢なシステムが必要だと考えられる．そのためデータ管理や通信手段を https などで暗号化したり，登録方法を別の手段を利用して本クラウドサーバ上で個人情報のやり取りを最小限にするなどの対策することが重要である．

5.2　UI・UX の向上

本論文のクラウドサーバは実験結果の通り，機能を確かめるための仮組みの設計でしかない．故に，対象ユーザとして農業従事者と定めている以上，基本

的にパソコンなどの操作に慣れていない可能性を考慮する必要が~~出る~~ある.

5.3 サーバの冗長性

本クラウドサーバは常時畑に設定したゲートウェイからセンサ情報を受け取ることが予想される．そのため，登録される畑が増えれば，クラウドサーバへのデータ送信は肥大化してゆく．特に対象としているのが夜間の獣害被害であるため，そのデータ送信も夜間に集中する可能性がある．そうなったときに，本論文のクラウドサーバ構成は，通信制御を行うリバースプロキシとデータベースがボトルネックとなることが考えられる．

そのような場合に備えて将来的にはボトルネックとなる，Docker コンテナを複数個並列して用意したり，負荷を分散させる機構 (kubernetes のようなロードバランサー) を用いることが必要になると考えられる．

謝辞

本研究を行うにあたって，親身かつ丁寧にご指導してくださりました，山口賢一教授，岩田大志准教授に最大限の感謝したします．

研究を行う際に助言や相談に乗っていただいた，山口賢一研究室，岩田研究室の皆様，心よりお礼申し上げます．特に共に研究を行なった山口賢一研究室の山口璃桜さん，岩田研究室の松尾慧一さんには感謝の意を表します．

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.