コンピューターのスクリーンショット

自動的に生成された説明小さい, 手押し車, 座る, 部屋 が含まれている画像

自動的に生成された説明

自立走行ロボット運用システム

プロジェクト演習4 製作課題　最終レポート

情報工学科　R4A　 R18E1002　正元　淳也

R18E1010　石橋　尚之

R18E1001　馬場　竣平

R18E1003　佐々木　祥二

内容

[概要　正元 4](#_Toc96895853)

[作品の特徴　正元 4](#_Toc96895854)

[開発動機　正元 4](#_Toc96895855)

[外部設計　正元 5](#_Toc96895856)

[ローバ　馬場 5](#_Toc96895857)

[操作用アプリケーション　正元 5](#_Toc96895858)

[メインページ 5](#_Toc96895859)

[マニュアル操作モード 6](#_Toc96895860)

[自動走行モード 6](#_Toc96895861)

[内部設計 7](#_Toc96895862)

[システム構成図　石橋 7](#_Toc96895863)

[Ngrok　石橋 8](#_Toc96895864)

[トークンの説明 9](#_Toc96895865)

[ローバ 11](#_Toc96895866)

[概要　石橋 11](#_Toc96895867)

[統括用メインプロセス　石橋 11](#_Toc96895868)

[方位センサ　石橋 12](#_Toc96895869)

[モータドライバ 石橋 13](#_Toc96895870)

[GPSモジュール 石橋 13](#_Toc96895871)

[電圧，電流計 　石橋 15](#_Toc96895872)

[部品選定 15](#_Toc96895873)

[カメラ　佐々木 16](#_Toc96895874)

[Raspberry Pi pico 石橋 17](#_Toc96895875)

[ハードウェア設計　馬場 17](#_Toc96895876)

[回路図　石橋 20](#_Toc96895877)

[使用部品　馬場 23](#_Toc96895878)

[基準局　石橋 23](#_Toc96895879)

[概要 23](#_Toc96895880)

[GPSモジュールF9Pのアンテナ測定設定 24](#_Toc96895881)

[GPSモジュールF9Pの配信設定 25](#_Toc96895882)

[RTK2GOのユーザ登録 25](#_Toc96895883)

[利用部品 25](#_Toc96895884)

[サーバ　正元 26](#_Toc96895885)

[概要 26](#_Toc96895886)

[操作用アプリケーション 26](#_Toc96895887)

[Apache2 29](#_Toc96895888)

[サーバ・ローバ間tcp通信　正元 30](#_Toc96895889)

[サーバ側 30](#_Toc96895890)

[ローバ側 31](#_Toc96895891)

[開発環境 32](#_Toc96895892)

[ローバ 32](#_Toc96895893)

[基準局 32](#_Toc96895894)

[Webサーバ 32](#_Toc96895895)

[作品評価 33](#_Toc96895896)

[RTKの測位の評価 石橋 33](#_Toc96895897)

[操作情報通信の評価 34](#_Toc96895898)

[操作用アプリケーションの評価 34](#_Toc96895899)

[ローバのハードウェア設計の評価　馬場 35](#_Toc96895900)

[まとめ 35](#_Toc96895901)

[参考資料 36](#_Toc96895902)

# 概要　正元

プロジェクト演習４の製作課題として，自立走行ロボット運用システム「panda」の開発を行った。

これの開発目的や設計について記述する。

# 作品の特徴　正元

屋外を走行するロボットを，PCやスマートフォンでどこからでも操作できるシステムである。

このシステムは，ローバと操作用 WEBアプリケーションで構成される。

WEBアプリケーションではローバの視点や位置情報をマップで確認できる。

また，ローバを前進後退・左右旋回させる遠隔操作するマニュアル操作モードや，設定した目的地までローバを自動走行させる自動走行モードがある。

ローバはGPS測位RTK法を行うことで，cm単位での自己位置測位ができる。

これにより，ローバは自動走行モードにて誤差±20cmと精密に目的地まで走行できる。

# 開発動機　正元

日本内閣府が策定したSociety5.0の方針の一つに，「ロボットや自動走行車などの技術で人の可能性が広がる社会」を目指すとある。

この方針をもとに，自動走行ロボットの研究開発が盛んにおこなわれている。

数ある自動走行ロボットの共通点は，「目的地まで自分の力で行く」ことである。これを基盤として，畑を耕したり，物を運んだりする機能を付けることによって人の可能性が広がる社会を目指している。

そこで，様々な自動走行ロボットの基盤となる「誰でもどこからでも使いやすく，目的地まで精密自立走行できるシステム」を目標としてpandaの開発に着手した。

# 外部設計　正元

屋外を走行するローバを，インターネット経由でPCやスマートフォン等様々なデバイスで，操作用アプリケーションにて遠隔でマニュアル操作やローバの目的地設定を行う。

ローバと操作用アプリケーションに分けて，見た目や操作方法について述べる。

## ローバ　馬場

悪路を走行するために，直径200mmの大きなタイヤで駆動輪は4輪駆動を採用した。

雨風を避けるために，3Dプリントした部品とアクリルで筐体の形状を箱型にした。

緊急停止ボタンで電源の入切が可能にした。

## 操作用アプリケーション　正元

操作用アプリケーションとは，PCやスマートフォン等様々なデバイスで使用できる，ローバを操作するためのWEBアプリケーションである。

操作アプリケーションは「メインページ」，ローバを遠隔で前進・後退・左右旋回させる「マニュアル操作モード」，ローバの目的地を設定し，目的地まで自動走行させる「自動走行モード」の3つのページで構成される。

各ページ上部にナビゲーションバーを配置する。ナビゲーションバーには各モード移動するためのボタンとローバの緊急停止ボタンを配置する。

ローバの緊急停止ボタンを常に配置することで，意図しない動作が行われた際にすぐに停止できるようにする。

### メインページ

上部ナビゲーションバーのpandaアイコンを押すことでメインページに移行する。

メインページには，ローバの視点と現在位置や進行方角等のステータス，そしてGoogle Mapsを表示し，ローバの現在位置をGoogle Maps上にアイコンにて示す。

ローバの現在位置などのステータスは1秒ごとに更新される。

グラフィカル ユーザー インターフェイス, Web サイト

自動的に生成された説明

図 1　メインページ

### マニュアル操作モード

上部ナビゲーションバーのゲームパッドアイコンを押すことでマニュアル操作モードに移行する。

マニュアル操作モードでは，メインページに上下左右ボタンを追加する。

上下左右ボタンを押すことで，ローバに前進・後退・左右旋回の動作をさせる。

より直感的な操作を実現するために，ボタンを押している間は各種動作を継続して行い，ボタンを離すことでローバを停止させる。

グラフィカル ユーザー インターフェイス

自動的に生成された説明

図 　マニュアル操作モード

### 自動走行モード

上部ナビゲーションバーのピンアイコンを押すことで自動走行モードに移行する。

自動走行モードでは，メインページに目的座標を設定するためのテキストボックスやボタンを追加する。

Google Maps上をタップするか，テキストボックスに緯度経度を入力した後にsetボタンを押すことで，ローバの目的地を設定できる。

また，目的地がどこか分かりやすくするために，Google Maps上にアイコンを立てる。

Google Mapsを操作するうちにアイコンが何処にあるか分からなくなることが想定されるので，それぞれのアイコンを中心座標としてGoogle Mapsを表示しなおすための，roverボタンとtargetボタンを設置する。



図 　自動走行モード

ローバ起動方法　石橋

ローバの起動方法は，GPIO22とGNDをジャンパピンでショートした状態で起動することである。

そして，WEBアプリケーションにカメラ映像を配信するためにSLACKのチャンネルにngrok\_onコマンドを送信する必要がある。

シャットダウンの方法はプルアップしてあるGPIO22をGNDに3秒以上接続したあとにHIGHに変化させることで行えるがカバーによって操作しにくいため，SSH接続したコンソールからシャットダウンコマンドを入力するべきである。

コンソールよる操作ができないときは，緊急シャットダウンボタンがある。これはRaspberry Piやモータドライバなどに対する電源電圧の供給を切断する機能がある。

# 内部設計

## システム構成図　石橋

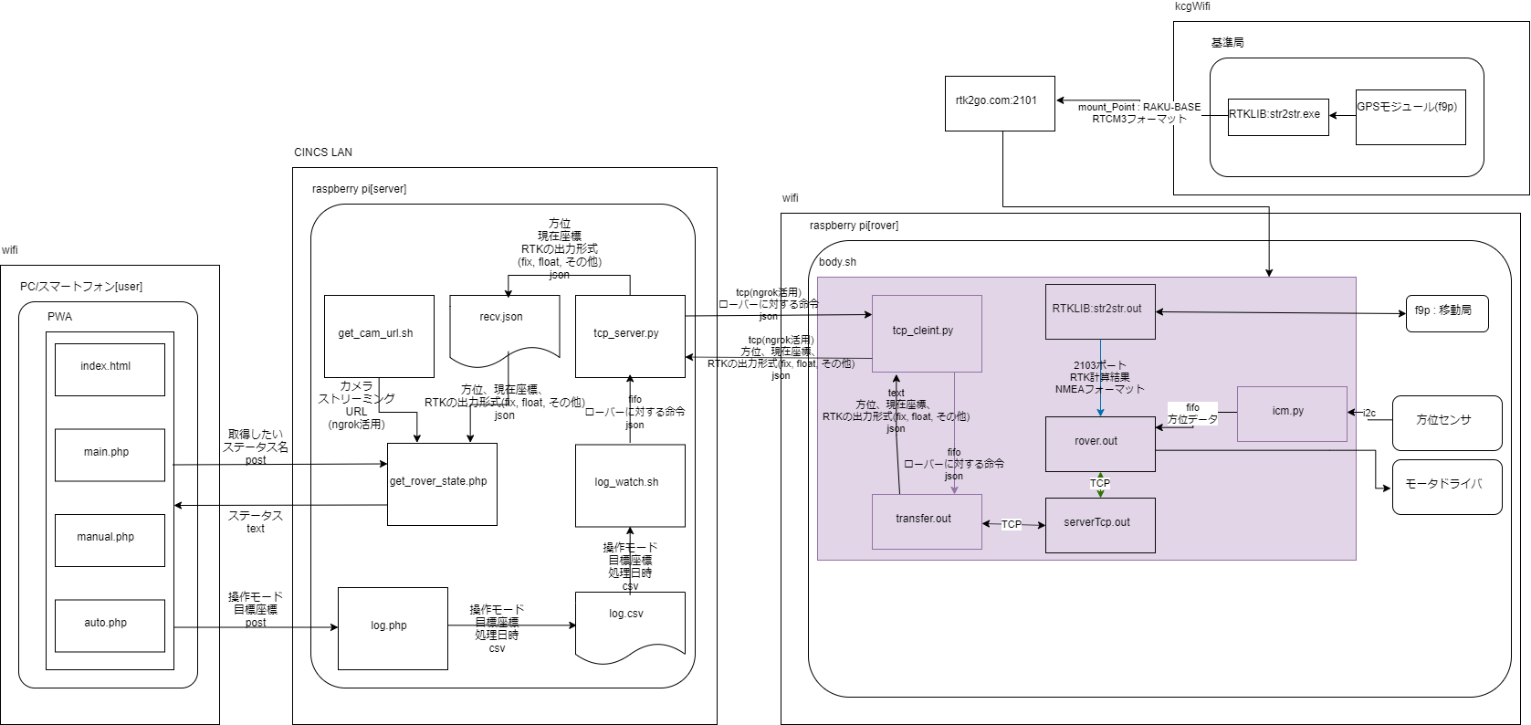


図 　システム構成図

## Ngrok　石橋

屋外のどこからでも操作用ページに接続できるようにするためにNgrokを利用した。

無料版NgrokはURLを固定することができないので，slackを活用して，操作用ページのURLを共有するようにした。

無料版Ngrokはメールアドレスを登録することで，ランダムなURL，ポートでのHTTP，TCPトンネルを作ることができ，ssh接続を行いコロナウイルスが流行する中でも，自宅から作業を行うことができた。

Ngrokサーバ一つにつき1つのNgrokプロセスを通して，4つのトンネルを作ることができる。

Ngrokサーバは日本を含めて7つ存在する。よってNgrokはメールアドレスを登録することで最大28のトンネルを用意することができる。

下記は実際にNgrokを通して利用したポートである。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| rover | | | | | |
| mySSH\_rover | localhost:60022 | | | |
| remote\_rover(reomote Desktop) | localhost:3389 | | | |
| camera\_rover | <http://localhost:8080> | | | |
| Server | | | | |
| mySSH\_server | | | | localhost:22 |
| tcp1\_server(サーバローバ間接続) | | | | localhost:8000 |
| Http\_server | | | | <http://localhost:80> |
| Https\_server(操作画面) | | | | <https://localhost:443> |
| Base | | | | |
| mySSH\_base | | localhost:22 | | |
| tcp1\_base(基準局データ) | | | localhost:2102 | |
|  | | |  | |

表 　Ngrok公開ポート一覧

Ngrokは起動するたびにURLとポート番号が変化してしまうため，slackを活用してURLを共有できるようにした。

モニター画面に映る文字

自動的に生成された説明

図 　slack\_bot 動作例

上記は使用例である。

ngrokURLを記録する掲示板としてSlackを利用時に必要となるのは，bot用アクセストークンとメッセージ書き込み用トークン，書き込み先チャンネルIDである。

それぞれの取得方法と役割は下記のとおりである。

### トークンの説明

・メッセージ書き込み用トークン

後述の書き込み先チャンネルIDを利用して，Botプログラムの起動状態を確認や自作プログラムからbot操作用メッセージの送信を行うことができる。

・slack送受信用Token取得方法

1. appsの作成

1. api管理画面URLにアクセス。

2. [Create New App]実行

3. [From scratch]選択

4. [App Name]に好きなAPP名を入力(後から変更可能？)

5. [Pick a workspace to develop your app in:]でAppsを作成したいワークスペースを選択。

(作成したいワークスペースがない場合は「ログイン方法」を確認。)

6. [Create App]でAppsの作成

1. トークンに対する許可の制御

下記の二つそれぞれに[Add an OAuth Scope]から下記の4つの権限を与える。

・ 許可の場所

- OAuth & Permissions > Scopes > Bot Token Scopes

・ 許可の内容

- channels:history

- channels:read

- chat:write

- chat:write.customize

2. トークン発行

1. F5などで再度画面を読み込み

2. [OAuth Tokens for Your Workspace > Install to Workspace]からトークン発行

3. [OAuth Tokens for Your Workspace > User OAuth Token] から実際のトークを取得

・Bot用アクセストークン

Botプログラムはこのトークンを利用して，slackに投稿されたメッセージを読み込み，それに対応した動作やslackに返信を行う。

・Botトークン発行

1. アプリを検索するから，HuBotを検索し，インストール
2. 設定を追加を追加
3. ユーザ名を登録後，Hubotインテグレーションの追加をクリック
4. APIトークンをメモ

・書き込み先チャンネルID

メッセージ書き込み用トークンをサポートし，プログラムによって送信されたメッセージの送信先を決定する。

メッセージ書き込み用トークンと同一ワークスペースであることが必須である。

チャンネルID取得には，slackアプリで取得したいチャンネルを右クリック後，チャンネル詳細を開いた後に下にスクロールでチャンネルIDを取得できる。

・slackbotののコマンド入力時による操作

入力コマンドは下記を想定していますが，helpを含む一部コマンドは全てのbotが反応している。

1.コマンド[スペース][送信先botのモード]

利用可能なコマンドの詳細はslackにhelpと書き込むことで確認できる。

## ローバ

### 概要　石橋

ローバはサーバから受け取った操作命令に従って，動作する。

基準局からGPSデータを受取ったRTCM3データを使い，搭載しているGPSモジュールにRTK測位を行わせる。

RTKの計算結果や方位データをサーバに返す。

それぞれどのように機能を実現しているか述べる。

### 統括用メインプロセス　石橋

統括用プロセスとして，モータの制御や目的地への移動の計算を行うrover.outとサーバとの送受信を中継するローバサーバが上げられる。  
この２つに分けた理由は開発当初はサーバとの通信形式やrtk測位の結果の受信方法などの仕様が未定だったことや，アプリケーション用サーバが未完成のときにローバを動作させるプログラムをrover.outから分離して作成したかったからだ。  
通信方式としてfifoも選択肢に上げられたが，接続するプロセスの数が未定だったことから，選択肢から外した。  
通信に利用するデータの形式を統一することで，データサイズが大きくなったが，一部のデータ形式を変更したことによる通信エラーの可能性を排除できた。メインプロセスを設計するに当たって，高い拡張性を持つように考えた。

ログファイルの保存のためにパスを通す必要がある。

環境変数を永続化するには， /etc/profile などに下記を追加するとよい。

PRO4\_ROVER\_DIR\_PATH="/home/pi/\*\*\*/rover/"

### 方位センサ　石橋

icm20948は3軸ジャイロセンサ，3軸加速度センサ，3軸磁気センサを搭載しており，水平に固定して磁気を測定することで方位を取得した。

理論的には磁気のx軸とy軸の値をatanによって取得できる。

しかし，実際は中心点がx=0，Y=0を中心点にしていない上に，正円ではない。

ローバは精密な旋回を行えないため，数度のズレを許容すると考えれば円形に近いので正円出ないことは問題にはならないが，中心点がずれていることは問題である。

そこで，測定値に事前に測定した値から用意したオフセットを加算することで対策を取ることにした。

オフセット値はX軸とY軸ともにセンサを水平にしたときの最大値と最小値を用いて下記の式から計算できる。

OFFSET =（MAX - MIN)/2 - MAX

この計算方法の問題点はセンサがz軸方法の揺れを考慮していないことであるが，センサは水平に固定されていることと，ローバが精密な旋回を行えないことから，対処すべき問題ではないと考えた。

方位センサのデータを取得するにあたって，開発期間短縮のためにpythonの配布ライブラリを利用した。

モータなどを動作させているメインプロセスとの通信はfifoを利用しており，ループのたびに「[方位データ][，]」の形式で送信している。

そして，メインプロセスは最後にデータを最新の方位データとして読み込む。fifoにデータが存在しなければ，前回の方位から変化していないと認識する。

グラフ

自動的に生成された説明

図 方位センサ測定結果と編集結果

### モータドライバ 石橋

モータを制御するPWM出力にはLAP方式とSMB方式が存在する。今回は制御性が低く消費電流も少ないSMB方式ではなく，制御性が高く消費電流が多いLAP方式を利用した。LAP方式では回転の方向を決定するDIRピンにPWM信号を流し，PWMピンに旋回方向を決めるHIGH， LOW信号を出す。

GPIOのPWM出力はwiringPiを用いた。利用したraspberry piは4Bだったため，wiringPiをアップデートしてバージョンが2.52であることを確認後に利用した。

### GPSモジュール 石橋

GPSモジュールには2周波のGPS信号に対応できるだけでなく，RTK計算用エンジンを内蔵しているF9Pを利用した。

RTKLIBv2.4.3のstr2strを活用することで，USBコネクタ単独でGPSモジュールへの基準局データの送信とtcp通信によるRTK計算結果配信を行うことができる。

実際に利用したコマンドは下記のとおりである。

str2str -in ntrip://rtk2go.com:2101/RAKU-BASE -out serial://ttyACM0:115200#2103

コマンド末尾の[#2103]がtpcポート2103からRTK計算結果を配信することを示している。この機能はv2.4.3から実装されたため，ドキュメントに記載がないので注意が必要である。

Rover.outはtpcポートからエンコードが必要なデータやNMEAフォーマットを含む文字列を受信する。$GNGGAから始まるnmeaフォーマットのデータを解析することで，RTK測位の結果を知ることができる。RTK測位の結果としてfixかfloat，その他かどかはGNGGAの7番目のデータで知ることができる。

#### 移動局GPSモジュールの設定

GPSモジュールの設定はu-centerの[view] - [Message View]を利用して行う。後述の設定はRTKを行う上で最小限の設定ではなく，RTKの実行とu-centerを利用して多くのパラメータを確認するための設定である。

1.[UBX] - [CFG] - [GNSS]

以下のようにGNSSデータを受信する衛星の選択を行う。

GPS[L1C/A， L2C]，Galileo[E1， E5b]，BeiDou[B1， B2]，QZSS[L1C/A， L2C]，GLONASS[L1，L2]

2.[UBX] - [CFG] - [DGNSS]

演算で求めた搬送波の数をできるだけ固定する。

3 = RTK fixed Ambiguities are fixed whenever possible

3.[UBX] - [CFG] - [MSG]

出力するメッセージを選択する

01-35 NAV-SAT 受信している衛星の位置や信号レベル

01-07 NAV-PVT 相違情報

02−15 RXM-RAWX 搬送波位相などの観測データ

02-13 RXM-SFRBX 放送歴や航行の情報

F0-00 NMEA GxGGA 緯度，経度，UTC時刻

F0-04 NMEA GxRMC 緯度，経度，移動方位，速度

4.[UBX] - [CFG] - [NAV5]

Dynamic Model : 0-Portable

Fix Mode : 3 Auto 2D/3D

UTC Standard : 0-Automatic

Min SV Elevation : 15~25

5.[UBX] - [CFG] - [PRT]

この項目では3で選択したデータを出力する形式を選択することができる。

Target : USB においてUBXを選択肢から外すと，USB接続においてu-centerの設定が行えなくなるという記述を見たので注意が必要になる。

Target : 3-USB

Protocol In : 0+1+5 UBX + NMEA + RTCM3

Protocol out : 0+1+5 UBX + NMEA + RTCM3

6. [UBX] - [CFG] – [CFG]

設定を電源が切断された後も保存するために，Save Current Configurationを選択する。

### 電圧，電流計 　石橋

システムとプログラム的に完全に独立した機器として，電圧電流計INA260を搭載している。この機器によってモータドライバに流れる電源の電圧と電流をリアルタイムに測定することができる。このセンサはI2Cアドレスをはんだ付けで変更することができるため，複数搭載することもできる。

この機器を搭載したことで，バッテリーの電流値が足りないことに気がつくことができた。

### 部品選定

Raspberry Pi 4B 8G

できるだけ処理効率を高めるため，メモリ用が多い８Gにした。

Qwiic - ICM-20948搭載 9DoF IMUモジュール

方位が測定できるだけでなく，余裕があれば加速度などの処理も行うことで，精確な移動を行いたかったため。

Qwiic - ZED-F9P搭載 GPS-RTK2モジュール

RTK測位にはできるだけ，多くの衛星信号を取得した方が有利なため，2周波であり多くのデータを取得できるf9pを採用した。

INA260 電流センサ max15A

プログラム実行中の電圧と電流の値を連続で取得するため。

Raspberry Pi Pico

低価格であり，arduinoIEDを利用して過去のライブラリなどを利用できると聞き，実際に活用できるかを試したかったから。

vl53l0x

障害物検知を行うときに，センサと平行でなくても正確な距離を測ることがtofセンサであったため。

Cytron 10A 5～30V デュアルチャネル DCモータドライバ

高い電流値に耐えられるモータドライバとして採用した。

### カメラ　佐々木

ローバ視点を共有するためのカメラ選定と，カメラ映像配信方法について述べる。

#### 選定

ローバの自律性発展を考慮し，深度画像を取り入れられることを要件とし調査したところ，intel realSenseが適していると判断した。

realSenseの機種を比較した所，以下のものが候補として挙がった。

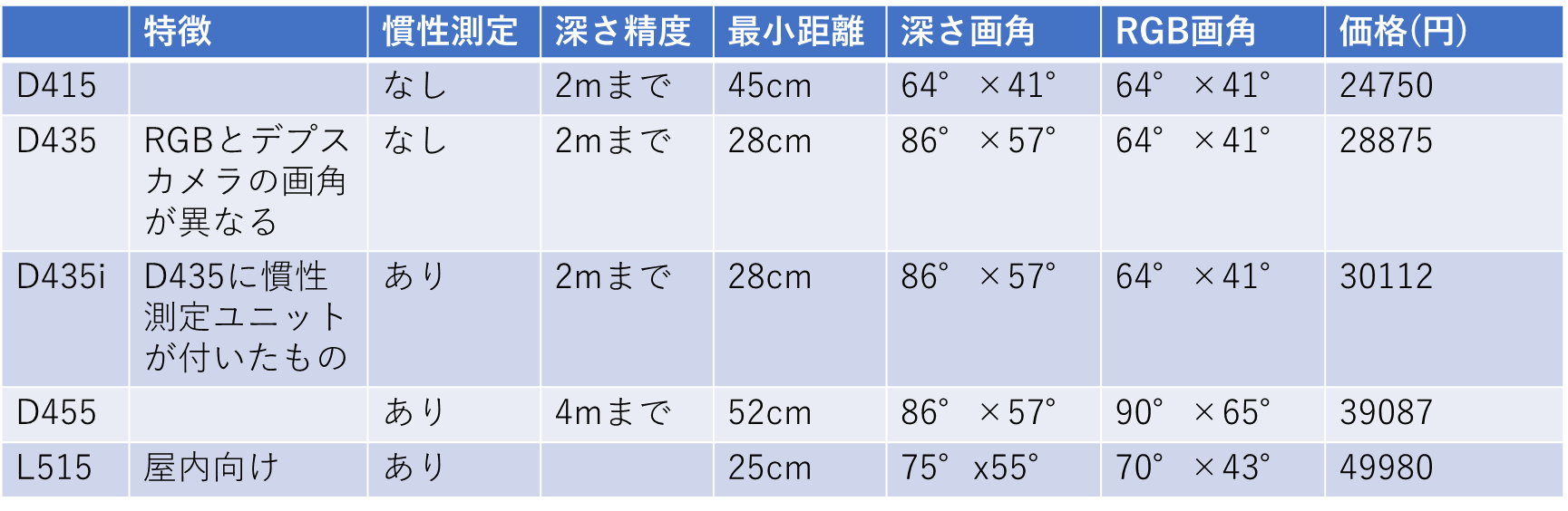


表 　realSense比較

この中から，画角が広く，屋外で使用でき，慣性ユニットで車体角度検出が期待できるD435iを選定した。

#### 配信方法

カメラ映像の配信はmjpg-streamerで行い，配信用ページをsngrokで外部公開し，slackbotを利用して操作用アプリケーションに配信した。

### Raspberry Pi pico 石橋

Picoは一系統のi2cラインに５つ以上のtofを接続した場合，i2cアドレスを取得できない。そこで，picoは2系統のi2cを持っていることを活かし，i2cに各3つのtofを接続することで6つのtofのデータを取得可能にいた。

また，tofのアドレスはtofがシャットダウンするたびにデフォルトアドレスに戻るため，picoが再起動するたびにtofのi2cアドレスを書き換える必要がある。

そして，6つのtofからローバの前後左右の一定距離に障害物があるかどうかを探索する。

picoとraspberry pi間の通信は4本の信号線をつなぐことで行った。例として，前に障害物があるときはpicoの1番の信号線をHIGHにする。そして，raspberry piが1番の信号線の変化を認識する。例に上げたような簡易的な方法でraspberry piは周囲の障害物を認識している。

Picoの開発はarduino IDEを利用して行った。そのため，tofを操作にはarduino用ライブラリが利用できた。

### ハードウェア設計　馬場

足回りに台車のタイヤを利用していて，それに対応するシャフトが無かった為，M16の六角ボルトの頭を加工し，モータのシャフトに固定する。タイヤは元々付属していたベアリングを取り外し， 代わりに3Dプリンタで作製した固定具に変更する。それらをワッシャとM16のナットで固定した。

フレームは，強度を上げる為に底面にアルミフレームを加工して作製した。

外装は，見た目にこだわりアクリルと3Dプリンタで作製したパーツを使い，箱型になるように作製した。

筐体に熱が籠りにくいように筐体の後方下側に網戸の網のようなもので熱が逃げるようにした。また，これによって網より大きい石や土等が侵入してくることが無くなった。

アクリルの加工は，レッザーカッタを利用した。障害物センサ(ToF)，カメラ(RealSense)，緊急停止ボタンを取り付ける為の穴をAutoCADで作製し，AutoCAD 2018 DXF(\*.dxf)ファイルで保存する。

コンピューターのスクリーンショット

自動的に生成された説明

図 　外装アクリル加工例

グラフ

低い精度で自動的に生成された説明

図 　3Dプリンタで筐体の支柱を印刷する時

印刷設定

・オブジェクト自動旋回　高品質モード

・プロファイル　Tough PLA (Default)

・一般

　内部充填密度　低 (15％)

　インフィルタイプ　蜂の巣

・サポート

　サポート，ラフトを有効にする

ダイアグラム が含まれている画像

自動的に生成された説明

図 　レーザーカッターを使用する場合

Fusion360から図面をdxfファイルに変換することができるが，SmartDIYsCreatorに変換したファイルを利用するとなぜかおかしくなっていたので，AutoCADで作製するようにしていた。

設定は，今回利用したアクリルは2mmだが，2mm設定がないので，プリセットをアクリル(黒)3mm切断にして切り切れない可能性がある為，マニュアルで回数を12回に変更して切断した。

### 回路図　石橋

ダイアグラム, 概略図

自動的に生成された説明

図 　全体回路図

ダイアグラム, 概略図

自動的に生成された説明

図 　 Raspberry Pi周辺回路

ダイアグラム, 概略図

自動的に生成された説明

図 　 Raspberry Pi Pico周辺のTof回路

グラフ

中程度の精度で自動的に生成された説明

図 　電源，モータ中継回路

ダイアグラム, 概略図

自動的に生成された説明

図 　電圧変換回路

### 使用部品　馬場

Raspberry Pi 4B 8G \*1

Qwiic - ICM-20948搭載 9DoF IMUモジュール \*1

Qwiic - ZED-F9P搭載 GPS-RTK2モジュール \*1

ANN-MB-00 2周波対応GPSアンテナ \*1

SMA/u.FLコネクタ変換ケーブル \*1

INA260 電流センサ max15A \*1

Raspberry Pi Pico \*1

vl53l0x \*6

Cytron 10A 5～30V デュアルチャネル DCモータドライバ \*1

型番不明12VDCモータ \*4

エアー式キャスター部品（タイヤ） 200MMグレー \*4

アルミフレーム \*6

クリサンデー アクリル板 黒 EX502 180mm×320mm 2mm厚 SS-2 \*1

アクリサンデー アクリル板 黒 EX502 320mm×545mm 2mm厚 S-2 \*2

3Dプリンタで作ったパーツ \*18  
M３プラスチック高ナット 6mm \*6

M３プラスチックネジ 4mm \*24

## 基準局　石橋

### 概要

アンテナの座標と取得した衛生の観測データをRTCM3フォーマットでRTK2GO.comに送信している。

アンテナの座標は単独測量を２４時間行った平均値を座標としている。

24時間測定したのは，GNSS衛生は２４時間の周期で航行しているので上記変動を含めて測量するには２４時間以上の測量が必要だからである。

・他のアンテナ座標の測定方法

１.最寄りの基準局の座標を利用してRTK即位を行う。

この方法は安定的に動作している基準局がなかったため，却下した。

2.国土地理院から提供される電子基準点情報を使う。

この方法ならば最も正確な座標を取得できる ため，不特定多数に利用してもらうためには恋の方法が最適である。しかし，リアルタイムRTK測量は成功させることができたが，後処理RTK測量には失敗し続けた。そのため，この方法を採用することができなかった。

### GPSモジュールF9Pのアンテナ測定設定

GPSモジュールの設定は常にu-centerというソフトで行う。

U-center のメニューから [View] - [Message View] を選択し，F9Pの設定画面を出し，下記の設定を行う。

1.[UBX] - [CFG] - [GNSS]

以下のようにGNSSデータを受信する衛星の選択を行う。

GPS[L1C/A， L2C]，Galileo[E1， E5b]，BeiDou[B1， B2]，QZSS[L1C/A， L2C]，GLONASS[L1，L2]

2.[UBX] - [CFG] - [DGNSS]

3=RTK fixed : Ambiguities are fixed whenever possible

3. [UBX] - {GFG] - [MSG]

この項目ではUSBやシリアルポートから出力するデータを選択でき，下記の項目をUSBから出力できるように設定する。

01-35 NAV-SAT 観測データ

F5-05 1005 基準極座標

F5-4D 1077 GPSの観測データ

F5-57 1087 GLONASSの観測データ

F5-61 1097 Galileoの観測データ

F-7F 1127 BeiDouの観測データ

F5-E6 1240 GLONASSの補正データ

4. [UBX] - [CFG] - [PRT]

この項目では2で選択したデータを出力する形式を選択することができる。

Target : USB においてUBXを選択肢から外すと，USB接続においてu-centerの設定が行えなくなるという記述を見たので注意が必要になる。

Target : 3-USB

Protocol In : 0+1+5 UBX + NMEA + RTCM3

Protocol out : 0+1+5 UBX + NMEA + RTCM3

5.[UBX] - [CFG] -[TMOOD3]

Mode : 1-Suvey-in

Minimum Observation Time 96400 s

Required Position Accurary 0.01 m

これらの方法でF9Pはアンテナ座標の測定を行う。

結果の確認は[UBX] - [NAV] - {SVIN]を確認することで，XYZ座標を得られる。

### GPSモジュールF9Pの配信設定

1.[UBX] - [CFG] -[TMOOD3]

Mode : 3-Fixed Mode

Fixed Position : 上記の方法で取得した基準局の座標をできるだけ細かく設定する。

2. [UBX] - [CFG] – [CFG]

設定を電源が切断された後も保存するために，Save Current Configurationを選択する。

・GPSモジュールのデータをRaspberry Piから配信

配信にはRTKLIBのsr2strを活用した。入出力の詳細なフォーマットはコマンド$ str2str –h で確認できる。後述の方法でRTK2GOにマウントポイントを登録し，RTK2GOに基準局のデータを送信した。

str2str -in serial://ttyAMA0 -out ntrips://:パスワード＠rtk2go.com:2101/RAKU-BASE

### RTK2GOのユーザ登録

登録にはユーザ名，メールアドレス，マウントポイント名が必須になる。マウントポイント名は既存のものと重複しては行けない。パスワードは空の場合ランダムで生成されるため，選択しませんでした。

登録確認メールに返信を行わなければ，数時間でマウントポイントが利用できなくなるので注意が必要になる。正式に登録が完了するまでは仮パスワードを利用する必要があり，仮パスワードはユーザ登録ページに記載されている。過去のブログと仮パスワードが違うことがあるので，利用時に確認する必要がある。正式登録が完了すると仮パスワードが利用できなくなり，利用していたシステムがエラーを出力するので，注意が必要になる。

### 利用部品

Raspberry Pi \*1

Semoic HG-GOYH 7151，測量GNSSアンテナ \*1

Qwiic - ZED-F9P搭載 GPS-RTK2モジュール \*1

ANN-MB-00 2周波対応GPSアンテナ \*1

SMA/u.FLコネクタ変換ケーブル \*1

同軸変換コネクタ SMAオス - TNCメス \*1

## サーバ　正元

### 概要

サーバは，操作用アプリケーションの配信，操作用アプリケーションの操作情報の受け取りと後処理，またローバへ動作命令の送信とローバから座標等の状態の受信を行う。

それぞれどのように機能を実現しているか述べる。

### 操作用アプリケーション

操作用アプリケーションはPCやスマートフォン等様々なデバイスからローバを遠隔操作するためのアプリケーションである。

様々なデバイスで操作するために，WEBアプリケーションとする。

また，PWAとすることによって，各デバイスにインストールしてホーム画面に追加できるようにし，アクセス性を高める。

#### PWA

PWAは，基本はWEBアプリケーションだが，デバイスにインストールしてオフラインでも動作でき，端末に通知を送ることができる。

今回は端末に通知を送ることも，オフラインでの動作も想定していないが，デバイスにインストールしてホーム画面に追加できるという点が本システムのコンセプトである，「どこでも誰でもどこからでも使いやすい」という点にマッチすると考え採用した。

WEBアプリケーションをPWAとするためには，アプリアイコンやアプリ名を定義するmanigest.jsonと，キャッシュをどのように残すかを定義するservice-worker.jsが必要となる。

これらは，後述するmonacaのPWAテンプレートにて生成されたものを修正して実装した。

また，PWAとしてアプリケーションを配信するためには，httpsで提供する必要がある。

https通信では証明書が必要だが，ngrokで配信することによってngrokの証明書が使われるため，不要となった。

#### Monaca

Monacaは，HTML5とJsでハイブリットアプリを製作できる開発ツールである。

今回，WEBアプリケーションを作成するのが初めてでノウハウもない為，monacaのテンプレートを利用することにより開発手順削減を図った。

また，monacaのテンプレートにはonsenUIというスマートフォン向きのUIを手軽に実装できるライブラリが入っているため，デザイン性と操作性の向上につながった。

#### 操作情報送信

操作用アプリケーションはサーバに操作情報を送信するために，HTTPのPOSTメソッドを利用する。

送信情報は以下の表の通りとする。

|  |  |  |
| --- | --- | --- |
| state | 動作モード | 0:停止，1:前進，2:後退，3:右旋回，4:左旋回，5:自動走行 |
| ymd | 動作開始日 | 即時実行：0000/00/00 |
| time | 動作開始時間 | 即時実行：00:00:00 |
| lat | 目標緯度 | float |
| lng | 目標経度 | float |

表 　操作用アプリケーションからサーバへの送信情報

POSTで送られた情報はlog.phpにて処理し，log.csvに追記する。

log.csvに追記する内容は以下の表のとおりとする。

|  |  |  |
| --- | --- | --- |
| nowDay | 処理日 | xxxx/xx/xx |
| nowTime | 処理時刻 | xx:xx:xx |
| state | 動作モード | 0:停止，1:前進，2:後退，3:右旋回，4:左旋回，5:自動走行 |
| ymd | 動作開始日 | 即時実行：0000/00/00 |
| time | 動作開始時間 | 即時実行：00:00:00 |
| lat | 目標緯度 | float |
| lng | 目標経度 | float |

表 　サーバでの保存情報

以上が操作用アプリケーションからサーバへの操作情報送信の流れである。

#### Google maps API

アプリケーション上にGoogle mapsを表示し操作するために，javascriptにてGoogle maps APIを利用した。

Google maps APIを利用するためには，APIキーの取得が必要となる。

APIキー取得手順を記述する。

Googleアカウントの作成

↓

Google Cloud Platfotm(<https://console.cloud.google.com/home/>)にアクセスし，新規プロジェクトを作成

↓

[APIとサービス]メニュー＞APIとサービスの有効化＞Maps JavaScript APIを有効化

↓

[APIとサービス]メニュー＞認証情報>認証情報を作成>APIキーを選択

この手順で生成されたAPIキーをウェブページにAPIキーを埋め込むことで，Google Maps APIを利用できるようになる。

#### マニュアル操作モード

マニュアル操作モードでは上下左右ボタンを押している間ローバを動作させ，ボタンを離したら動作を停止させる処理を実現するにあたり，PCとスマートフォンでクリックアクションの違いがあった。

PCではonmousedownにてボタンを押したことを，onclickにてボタンを押し終わったことを検知した。

スマートフォンでは，ontouchstartでボタンをタッチしていることを，ontouchcancelとontouchendにてボタンから手が離れたことを検知した。

<button id="up" ontouchstart="move\_forward\_rover();" onmousedown="move\_forward\_rover();" onclick="stop\_rover();" ontouchcancel="stop\_rover();" ontouchend="stop\_rover();">↑</button>

#### 自動走行モード

マップ上をタップすることで目的地を分かりやすく設定できるように，Google mapsにクリックイベントを追加した。

クリックイベントでは，タップした場所の座標を取得できるため，目的地マーカの座標をタップした座標に設定し再表示することで，タップした位置にマーカを配置することにした。

#### ローバステータスの取得

サーバのget\_rover\_state.phpに各ステータスを読み取る関数が定義されている。

POSTにて取得したいデータに対応した関数名を送信することにより，各種ステータスを取得している。

ローバの現在座標・進行方位・RTKステータスはローバの現在ステータスが格納されているrecv.jsonを読み取り，カメラURLはslackbotを利用して取得している。

### Apache2

Webページを公開するために，Apache2でWEBサーバを構築した。

Apache2のインストール方法を以下に示す。

$ sudo apt-get install apache2

$ sudo service apche2 start

以上で，httpとhttpsで/var/www/html/index.htmlの内容を配信するWEBサーバが構築できる。

Apache2をインストールすると，/var/www/htmlディレクトリが生成される。

デフォルトでは，サーバにアクセスすると，/var/www/html/index.htmlが開かれる。

開発当初，アクセス時に開くファイルの変更方法が分からなかったため，index.htmlに開いたらmain.phpにジャンプする記述をすることでアプリケーションを開くようにした。

## サーバ・ローバ間tcp通信　正元

ngrokで公開したサーバのtcpポートをslackbotで共有し，そこにローバが接続して双方向通信を行っている。

送信と受信でポートを分けることなく，1つのプログラムで双方向通信を実現した。

### サーバ側

tcp\_server.pyにて8000番ポートを公開し，クライアントの接続を待ち，接続されたら双方向通信を開始する。

サーバは以下の内容をローバに送信する。

|  |  |  |
| --- | --- | --- |
| サーバ→ローバ | | アプリケーション操作時に送信 |
| mode | 動作モード | 0:停止，1:前進，2:後退，3:右旋回，4:左旋回，5:自動走行 |
| lat | 目標緯度 | float |
| lng | 目標経度 | float |

表 　サーバからローバへの送信情報

これらのデータはjsonフォーマットで送信しているため，要素が増えても通信バッファ1024Byte以内であれば問題ない。

送信データは，log\_watch.shにてlog.csvの更新内容を1行ずつ読み取り，json形式でPIPEに書き込むことでtcp\_server.pyに受け渡しをしている。

当初，PIPEではなく，標準ファイルに入出力することでデータの受け渡しをしていたが，遅延が大きかったためPIPEにてデータの受け渡しをすることにした。

PIPEはブロッキングされるため，操作情報が更新されないと読み取りで処理が停止し，次に進まないという問題がある。

そこで，送信と受信でタスクを分けてマルチタスク処理することで，動作の停止を回避した。

また，別の対策として，pythonではなくC言語で実装しなおし，PIPEを開く際にノンブロッキングオプションを入れることが考えられる。

また，受信タスクでは受信したデータをrecv.jsonに上書きすることで，現在ステータスを常時更新している。

### ローバ側

Ngrokにて公開されているサーバのtcpポートをslackbotで取得し，アクセスすることで双方向通信を開始する。

ローバは以下の内容をサーバに送信する。

|  |  |  |
| --- | --- | --- |
| ローバ→サーバ | | 1s周期で送信 |
| mode | RTKステータス | 4:FIX，5:FLOAT，else:error |
| dir | 進行方位 | 0~360 float |
| lat | 現在緯度 | float |
| lng | 現在経度 | float |

表 　ローバからサーバへの送信情報

送信データはjsonフォーマットで送信しているため，要素が増えても通信バッファ1024Byte以内であれば問題ない。

サーバとは違い，ローバの送信内容の即時性はそれほど重要ではないので，PIPEではなく標準ファイル入出力でローバプログラムからtcp\_client.pyに受け渡しをしている。

また，受信は常時行い，jsonファイルに書き込むことでローバに現在の操作情報を受け渡している。

## 開発環境

### ローバ

Raspberry Pi 4 Model B / 8GB

Raspbian 10.11

GNU bash, バージョン 5.0.3

RTKLIB v2.4.3

wiringPi 2.52

slackbot for python v1.0.0

sparkfun-qwiic-icm20948 v0.0.1 for python

ngrok version 2.3.40

nlohmnn-jsonn

Raspberry Pi pico

Arduino IDE

VL53L0X library for Arduino

Fusion360

AutoCAD

XYZPrinting da Vinchi Jr. 1.0A

XYZprint

FABOOL Laser Mini

SmartDIYsCreator

### 基準局

Raspberry Pi 4 Model B / 8GB

Raspbian 10.11

GNU bash, バージョン 5.0.3

RTKLIB v2.4.3

### Webサーバ

Raspberry Pi 4 Model B / 8GB

Raspbian 10.11

Python3.7

Apache/2.4.38

Html5

PHP 7.3.31

GNU bash, バージョン 5.0.3

slackbot for python v1.0.0

# 作品評価

## RTKの測位の評価 石橋

GPSもジュールを固定して数分間連続測定した結果は下記の通りになった。この結果から，4x5cmの範囲内に収まっていることから数センチ単位の測定に成功していると考えられる。

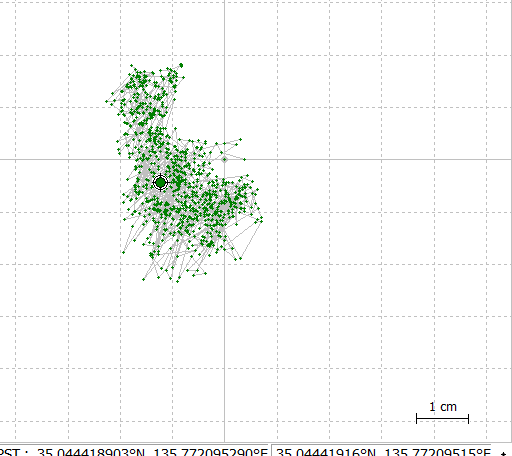


図 　GPS測定結果１

下記のテーブルの四方にそって移動させた時も測定結果が四角形になっていることがわかるので，移動しても数センチ単位の測定が成功していると考える。測定結果が70x85cmなのはモジュール台が正方形ではなかったため，長方形に近い移動をしたためだと考えられる。



図 　GPS測定結果２

下記の図はローバを四角形に移動させたRTK結果の記録である。開始位置と終了位置が20cmずれている。数センチ単位での移動をローバが行うことができないため，大きめの誤差を許容する必要があるためである。

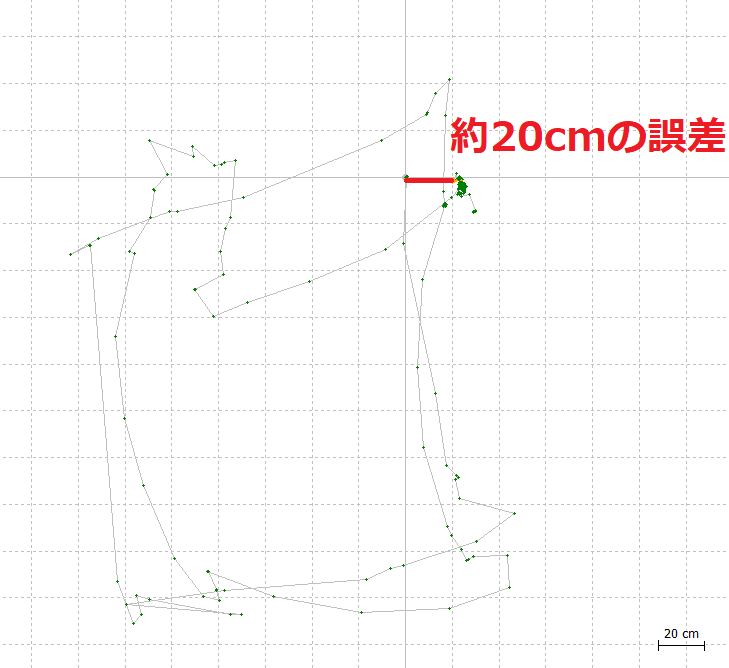


図 　ローバ走行時のGPSログ

これらの結果から，ローバは移動の制御性が低いため，RTK測位の性能を最大限引き出すことができていないと考えられる。しかし，RTK測位による数センチ単位の測定そのものには成功していると考えられる。

## 操作情報通信の評価

Ngrokとslackbotを利用することで，グローバルIPが割り当てられていない環境でもシステムの動作ができるようになった。これにより，コンセプトの1つである，「どこからでも使える」ということを実現できた。

しかし，ngrokにて1度サーバを経由しているためか，マニュアル操作してからローバが動作するまでに約0.5sの遅延が発生しているため操作性は悪くなってしまった。

しかし，本システムのメイン機能は自動走行モードであり，マニュアル操作は予備機能であるため大きな問題にはならないと考える。

## 操作用アプリケーションの評価

操作用アプリケーションはPWAとすることにより，PCやスマートフォンにインストールできるようになったため，アクセス性は高くできた。

また，Google mapsの航空写真を見ながら目的地周辺の様子を確認し，マップ上をタップすることで目的地を設定できるため，コンセプトの「誰でも使いやすく」を実現することができた。

しかし，ローバのステータスをコンソールのように表示しており，まだ開発者目線の設計となっているので，そこは改善の必要がある。

また，WEBアプリケーションを開発するのが初めてということもあり，手探りで開発に着手したことから，ソースコードの構成が分かり辛いことが反省点である。

## ローバのハードウェア設計の評価　馬場

アルミフレームを使っている為，フレーム部の強度はしっかり出来た。また，ある程度の雨風を防げる筐体を作ることが出来た。

旋回する時に筐体全体が振動する為，色々なとこのネジが緩む。定期的に締めなおす必要がある。

改良点としては，悪路を想定しなかったらタイヤをオムニホイルにすると動きに多様性を持たせることが出来ると考えた。また，悪路想定するのであればタイヤの直径が大きくて柔らかいタイヤでサスペンションを付けるなど4つのタイヤはしっかり地面に設置するとスムーズに動作できると考えた。

アクリル板は２mmではぶつかったりすると破損したりした為，厚みを増やす必要があった。

全体的にスペースの余裕がある為，タイヤの大きさを考えると筐体のサイズを小さく出来る。

# まとめ

作品コンセプトである，「誰でもどこからでも使いやすく，目的地まで精密自立走行できるシステム」をPWAやGoogle Maps API，GPS測位RTK法を利用することにより実現することができた。

Pandaの次の段階として自律性が求められる。

現在は目的地へ一直線に行くだけだが，どのようなルートで行くのか。また，障害物があったときに，どのように回避するのかという判断をできるようにしなくては，実際に公道を走らせることはできない。

これを実現するために，カメラやLiDARを用いて周囲状況を把握することが課題である。

また，スケジューリング機能や，過去走行ログ表示，バッテリー充電管理，故障等のステータス表示，複数台のローバをサーバに接続して，運用システムとしての機能を充実させる必要がある。

まだ課題は多く残っているが，「誰でもどこからでも使いやすく，目的地まで精密自立走行できるシステム」を実現できたため，最低要件を満たすことができた。

今回得た知識を，今後の開発や業務に役立てていきたい。

# 参考資料

LAP方式とSMB方式について

[https://tattatatakemori.hatenablog.com/entry/2017/07/20/232827]

WiringPiのアップデート

http://wiringpi.com/wiringpi-updated-to-2-52-for-the-raspberry-pi-4b/

NMEAフォーマットの例

<https://www.hiramine.com/physicalcomputing/general/gps_nmeaformat.html>

ICM20948ライブラリ

<https://github.com/sparkfun/Qwiic_9DoF_IMU_ICM20948_Py>

RTKLIB V2.4.3

https://github.com/tomojitakasu/RTKLIB/tree/rtklib\_2.4.3

C++ - Vincenty 法による地球楕円体上の位置計算！

<https://www.mk-mode.com/blog/2020/10/21/cpp-geodesical-calculation-by-vincenty-2/>

センチメートルGPS測位 F9P RTKキットマニュアル　CQ出版

ラズベリーパイ上にApacheサーバを建てる

<https://arrow.hatenablog.jp/entry/2017/11/23/223454>

Googleマップを使ってみよう！

<https://www.zenrin-datacom.net/business/media/g001/>

Google Maps JavaScript API でクリックした座標の取得

<https://qiita.com/SOJO/items/d43a28a6fe27de3cc85c>

指定したファイルの更新があったらコマンドを自動実行するシェルスクリプト

<https://mizti.hatenablog.com/entry/2013/01/27/204343>

Google Maps Platform

<https://developers.google.com/maps?hl=ja>

mjpeg-streamerのインストール&設定

<https://raspi-katsuyou.com/index.php/2020/06/30/11/10/44/644/#:~:text=mjpg%2Dstreamer%E3%81%AF%E3%83%A9%E3%82%BA%E3%83%91%E3%82%A4%E3%81%A7,%E3%81%99%E3%82%8B%E3%81%9F%E3%82%81%E3%81%AE%E3%82%BD%E3%83%95%E3%83%88%E3%82%A6%E3%82%A7%E3%82%A2%E3%81%A7%E3%81%99%E3%80%82>

Googleマップに色々なマーカーを描画する方法

<https://maps.multisoup.co.jp/blog/1881/>