Raspberry Pi を用いたラリー競技車両用 ナビゲーションシステムの開発

石原実*　新美和正*　伊藤晴太郎*　北条直樹*　三川侑良*
黒木優汰*　石間伏彗太*　安田綜一郎*　鈴木遼平*　中嶋健二*
藤原真由美**　山口こころ***　笠井正三郎****

Development of Navigation System for Rally Competition Vehicles using Raspberry Pi

Minoru ISHIHARA*　Kazumasa NIIMI*　Seitaro ITO*　Naoki HOJYO*　Yura MIKAWA*
Yuta KUROKI*　Keita ISHIMABUSHI*　Soichiro YASUDA*　Ryohei SUZUKI*　Kenji NAKAJIMA*
Mayumi FUJIWARA**　Cocoro YAMAGUCHI***　Shozaburo KASAI****

Keywords: *Raspberry Pi, Navigation System, Rally, CAN network, GPS*

1. はじめに

モータースポーツのカテゴリーの中にラリーという競技がある. ラリーとは複数区間の走行時間を計測し, その合計で勝敗を競う競技である. 移動区間はリエゾンと呼ばれ閉鎖された一般の公道を使用する. またラリー最大の特徴としてドライバーとコドライバーと呼ばれるナビゲーターが競技車両に乗車する. コドライバーにはリエゾン区間の案内や車両の状態をドライバーに伝える役目があるが, その手段は現状アナログであり, 車両情報･位置情報などを総合的に得るシステムは存在していない.

今回, コドライバーを助ける第 3 のドライバーとして本システムを開発する運びとなり電子工学科の学生 10 名で 2018 年 7 月プロジェクトチームが発足した. 本システムは, ドライバー･コドライバーから要望を聞き, 以下の 4 つの機能を実現するものを検討し, 製作できたので, 報告する.

- ･競技中の車両の状態をドライバーに伝える
- ･リエゾン区間で現在地をドライバーに伝える
- ･競技区間の天気情報をドライバーに伝える
- ･競技中の車載動画を録画する

2. システム概要

本システムは, 競技車両に搭載し, 邪魔にならず必要な情報がドライバーに確認しやすくかつ操作が容易である必要がある. 今回は, 競技車両のカーナビと置き換えて装着し, ディスプレイ上に必要な情報を表示させ, 操作もディスプレイをタッチすることにより実現させるものとした. システム開発のベースには Raspberry Pi を用いた.

2.1　ハードウェア

図 2-1 にハードウェア構成を示し, 図 2-2 に実際の車両搭載写真, 図 2-3 に背面からの写真を示す.

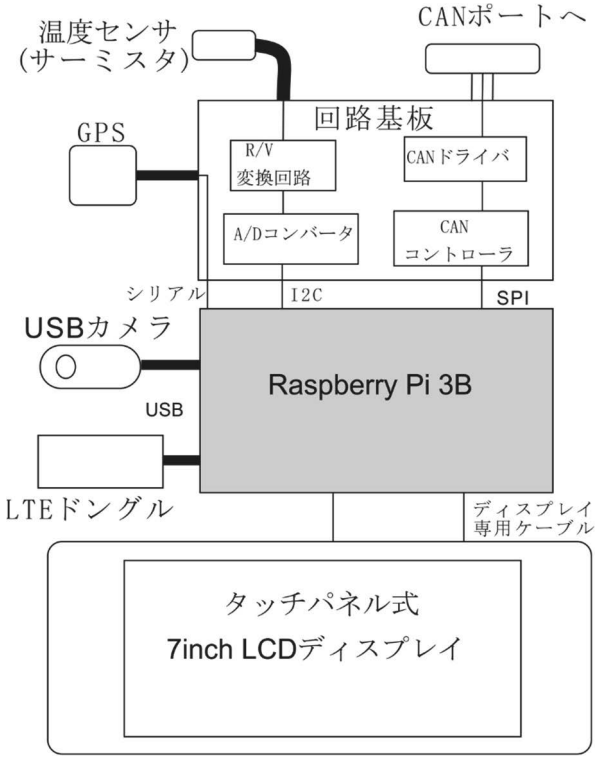

図 2-1　ハードウェア構成

*　電子工学科　学生　**　電子工学科技術職員
***電子工学科　非常勤講師　****電子工学科　教授

図 2-2　車両搭載時の写真

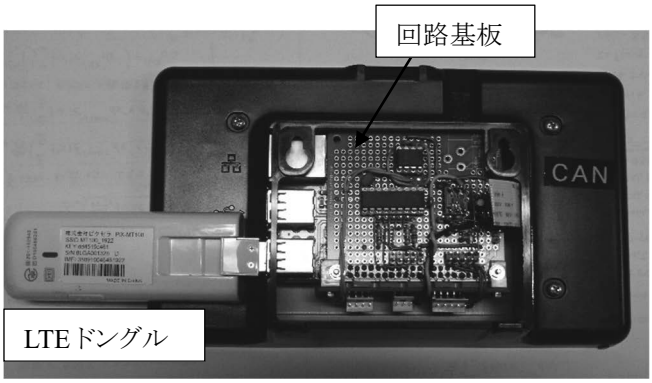

図 2-3　背面からの写真

自動車本体の多くの情報は CAN バス上に流れているため，CAN バスに接続して情報を得ることができる．この通信には，MICROCHIP 社の CAN コントローラ(MCP2515)と CAN ドライバー(MCP2551)を用いて実現している．Raspberry Pi と MCP2515 は SPI により通信している．CAN から得られない情報として，エンジンオイルの温度が必要であり，これは専用のサーミスタ温度センサをオイルパンに取り付けている．サーミスタは温度により抵抗値が変化するため，抵抗値を電圧に変換し，その電圧を A/D 変換して取り込んでいる．A/D 変換器は，MICROCHIP 社の MCP3425 を用いている．この IC は 16 ビットの分解能をもち，Raspberry Pi とは I^2C で通信を行うことができる．これらの回路は Raspberry Pi の GPIO コネクタに回路基板を取り付けて実現している．その他，インターネットと接続するために LTE ドングル(PIXELAPIX-MT100)を，走行中の映像を記録するためにカメラ(Logicool C270)を USB ポートに接続している．

位置情報を取得するための GPS は YIC の GT-902PGG を用いており，GPS/QZSS の L1 C/A 信号を受信し，1PPS で位置情報がシリアルデータとして出力される．この信号を Raspberry Pi のシリアルポートと接続し，取り込んでいる．

なお，本システムの電源は，モバイルバッテリ(imuto Taurus X6L)とし，競技車両とは分けている．

2.2　ユーザーインターフェース

競技中のドライバーの負担にならないようユーザーインターフェースは全てワンタッチで操作できるように作成した．図 2-4 は Raspberry Pi 起動時の画面である．各機能をアイコン化しアイコンをタッチすることでプログラムを実行させている(画面中央部の4つのアイコン)．

カメラ機能についてはユーザーの操作は不要とし Raspberry Pi を起動した時点で自動的に撮影を開始することとした．

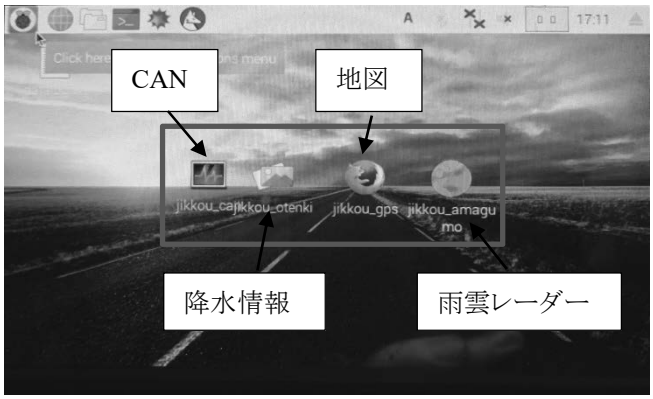

図 2-4　Raspberry Pi 起動画面

3. 機能

3.1　車両状態の把握

3.1.1　概要　ラリー競技において競技車両は日に数 100km 走行するため，車両の消耗や状態を把握することは競技を進めていく上で重要である．車両の状態は CAN 通信を用い車両の各コンピュータでやり取りを行っている．通常であればユーザーはその情報を得ることはできないが CAN バスに接続することにより車両の状態を知ることができる．本システムでは CAN 通信により車両情報を読み出すとともにエンジンオイル温度を温度センサで計測して，ディスプレイに表示する．

3.1.2　CAN 通信　CAN 通信とは，自動車内部の情報を自動車の各部でやり取りするための通信規格である．やり取りされる情報には，エンジン回転数，エンジン水温，トルク，故障情報などが含まれており，情報の種類ごとに ID 番号により管理され切り分けて読み取ることが可能となる．

データは"フレーム"という構造によって伝送される．このフレームには 8byte の自動車データ，フレームに入っているデータの種類を表す ID，その他フレームの伝送に必要な情報が含まれる．図 3-1 に CAN フレーム内自動車データの一例を示す．

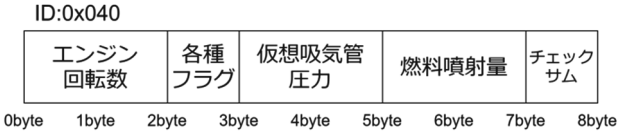

図 3-1　CAN データの一例

自動車から取得する CAN データは 0, 1 の並びからなるビット列で構成されているが, CAN 読み取りライブラリである "canTools"を使用することによって情報の種類ごとに切り分けて読み取ることが可能となる.

今回これらの情報を取り込んで表示するために, 自動車の CAN ネットワークと接続するための回路, および情報をユーザーに分かりやすく整理して表示するプログラム, ユーザーインターフェースを開発した.

3.1.3 **開発内容** CAN の情報を読み取り次第, 辞書型変数を使用し, データ値をデータ名と紐付けの上, コンピュータ内に保存する. これらの情報を UI に表示していく. この際, 負荷軽減の為に画面の更新は1秒毎に設定した. また, 視認性を上げるために背景は黒, 文字は白とした. 図 3-2 に表示例を示す. エンジン水温など情報が数値で伝送されているものは数値で表示し, 画面下部には故障情報を表示した. 故障情報はフラグでデータが送られてくるため, 車両の状態が正常な場合を青文字, 故障情報はオレンジ文字で表示した.

開発は, python3.7 をベースに, CAN 通信には canTools を用い, GUI 構築には tkinter を用いた.

なお, エンジンオイル温度(油温)は, CAN データにはないため, 2.1 で説明した通り, 別途情報取得を行い, 合わせて表示している.

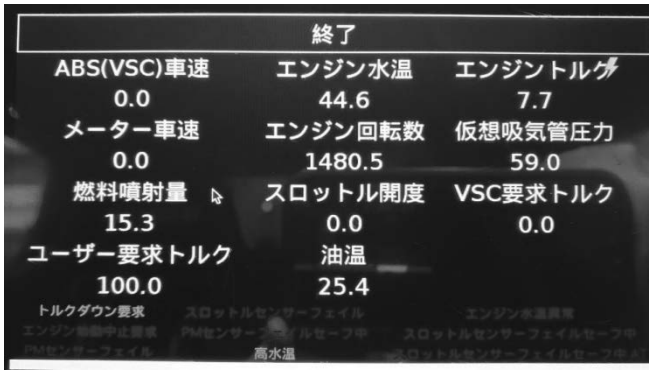

図 3-2 車両データ表示:実行画面

3.2 GPS による位置情報の取得と地図表示

3.2.1 **概要** ラリーではコドライバーによるナビゲーションを元にリエゾン区間を移動し競技を行う. コドライバーはコマ図と呼ばれる競技独自の案内図を見てナビゲーションを行うため実際に走行している位置を取得し, 現在地に関連する情報を知ることは競技を優位に進めていく上で必要となる. 位置情報に関しては, GPS を用いることにより取得することが可能である. 取得した位置情報より, 周辺地図および現在位置を表示させたい. 走行場所によってはインターネットへの接続が難しい場合があるため, インターネットからの地図情報をそのまま利用することは不安定である. そこで, オフライン環境で現在地の情報を表示することができる地図システムを構築した. このことにより, 安定した地図データを利用することができるばかりでなく, 通信負荷の軽減を図ることができる.

3.2.2 **GPS 通信** 今回使用する GPS モジュールは, 受信した GPS データをシリアル通信により GPIO ピンを使い Raspberry Pi に取り込んでいる. 受信したシリアルデータを Python で使用するため, pySerial(1)という Python のモジュールを使用した. これを使用すると Raspberry Pi の Python でシリアル通信が可能になる.

GPS モジュールから送られてくるデータは NMEA-0183 というフォーマットの文字列である. GPS モジュールは「みちびき」等の衛星から発信されている信号を受信し, 時刻, 緯度, 経度, 海抜, 高度, 測位に利用した衛星の数や ID, それぞれの衛星の位置(方位角と仰角)などの情報を NMEA-0183 フォーマットの文字列として送出している. この文字列をプログラムで扱いやすいものに変換するため, Python ライブラリである micropyGPS(2)を利用した. micropyGPS に GPS データを入力すると, それを解析して GPS オブジェクトにデータを追加, 更新していく. GPS オブジェクトのデータとして時刻, 緯度, 経度, 測位に利用した衛星の数や ID などの情報が得られる.

3.2.3 **オフラインマップ** 今回, オフラインマップを作成するために, Web 地図サービスで広く使われているオープンソースの JavaScript のライブラリである Leaflet を用いて, マップを HTML で出力した. これにより, タイルベースの Web 地図の作成を容易に行うことができた. また, オープンソース GIS である QGIS3.4 を用いて, 国土地理院の地図サービスの地理院地図の地図タイル(3)を保存した. こうすることで, オフラインでの地図情報の利用を可能にしている.

3.2.4 **地図システム** 図 3-3 に地図システムの構成図を示す. 前項で述べたように, 現在地の GPS 情報は Raspberry Pi に取り付けた GPS モジュールで受信しているが, このデータは HTML 上で扱うことができない. そこで,

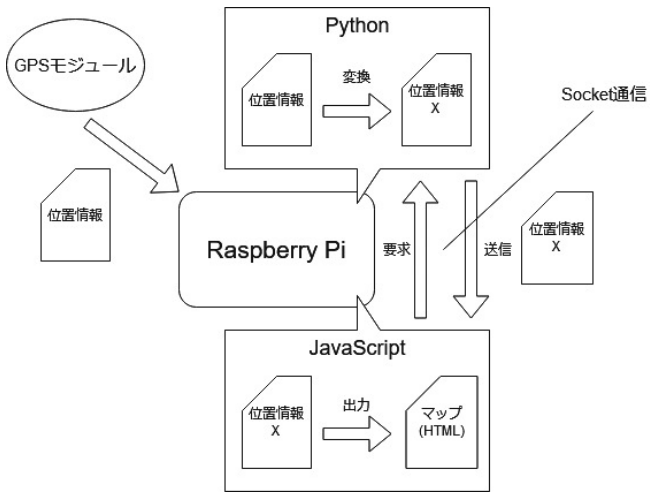

図 3-3 地図システムの構成

この位置データを Python で変換し, Socket 通信によって JavaScript に送信している. ここでは, Python でサーバー側のコードを, JavaScript でクライアント側のコードを記述した. また, WebSocket のモジュールとして, Python では websockets を, JavaScript では socket.io を用いた. 図 3-4 は本校の中庭でオフラインマップを使用した際に表示された画面である. 現在地を中央(地図上赤丸部分)とし, 周辺のマップ情報が表示されていることがわかる. 実際に使用する際は, マウス等は用いずタッチパネルでの操作を行うため, 左下のズームボタンや右上の終了ボタンは大きめに配置している.

国土地理院地図より

図 3-4　オフラインマップの使用中の画面

3.3 天気情報

3.3.1　概要　ラリーではコースが広範囲に渡るため各地点での天気情報, 主に降水量を把握することが勝敗の鍵を握る. そのため登録した各地点と現在地の降水強度を表示するシステムを作成した. 瞬間的な雨の強さを 1 時間あたりに換算した雨量を降水強度と言い, 単位は"mm/h"で表す. システムの流れを図 3-5 に示す. また, 降水強度だけでは, 特定の地点での雨の強さを予測しやすいが, コースの途中での雨の状況等は予測しにくい. そのため, Yahoo! JavaScript マップ API[4]を用いて雨雲レーダーも作成した.

3.3.2　内容　まず, Yahoo!が提供している気象情報 API[5]に各地点, システムの GPS モジュールから取得した現在地点のデータを渡す. そうすると, API はその地点の情報を xml または json 形式で出力するので, それをシステム上でユーザーが必要なデータのみ取り出し, GUI に表示する. 本システムでは json 形式を用いた. システム開発には Python3.5.3 と Python 用の GUI ライブラリである Tkinter を使用した. Yahoo!の気象情報 API へのリクエスト部分を示す.

```
url_d="https://map.yahooapis.jp/weather/V1/place?coordinates={longitude}.{latitude}&appid={key}&output=json"
```

この部分で気象情報 API に欲しい座標地点の情報のリクエストを行っている. このリクエストのパラメータを以下に示す.

・longitude:少数点 3 桁以上で表した経度
・latitude:少数点 3 桁以上で表した緯度
・key:Yahoo!JAPAN から提供されたアプリケーション ID

情報を取得する際はこの url_d に各パラメータを入力して, リクエストを送信する.

実際に使用している際の画面を図 3-6 に示す. 表示は上から,
・あらかじめ設定した 3 地点の名前, 現在の降水強度
・現在地点の座標, 現在の降水強度, 5 分後の降水強度
となっている.

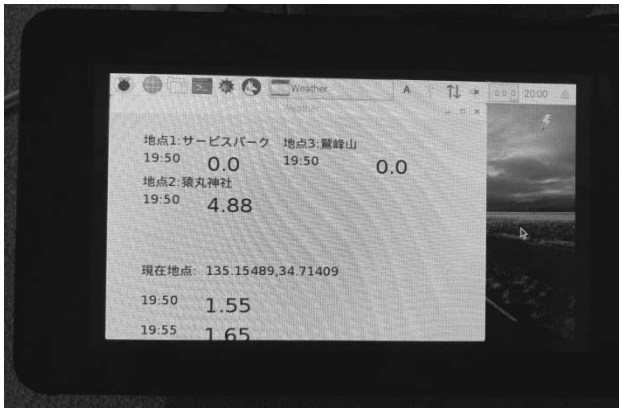

図 3-6　降水強度システム使用中の画面

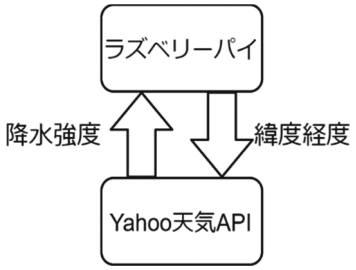

図 3-5　降水強度システムの概要

次に, 図 3-7 に雨雲レーダーの構成図を示す. 雨雲レーダーは, Yahoo! JavaScript マップ API を用いて, 現在地周辺の雨雲情報を 1 分ごとに取得している. 前項で述べた地図システムと同様に, 現在地の GPS 情報を HTML 上で扱えるように Python で変換し, Socket 通信で読み込み Java Script で扱えるようにしている.

このようにして取得した位置情報を中心とした地図に, Yahoo! JavaScript マップ API で取得した雨雲情報のレイヤを重ねることで, 地図上に雨雲を表示している.

実際に, 本校の中庭で雨雲レーダーを使用した際の実行画面を図 3-8 に示す. 現在地を中心に雨雲情報が表示されていることが分かる. 左上にズームバー, 右上に終了ボタンを配置している.

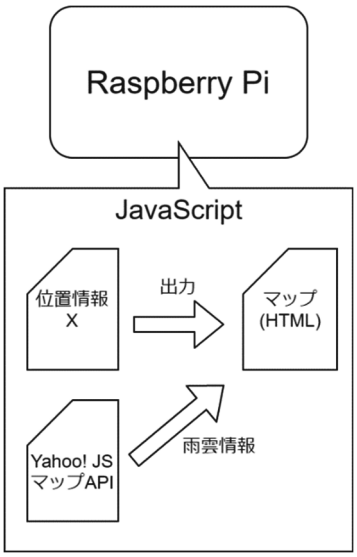

図 3-7　雨雲レーダーの構成

図 3-8　雨雲レーダー使用中の画面

3.3.3 **今後の予定**　今回作成したプログラムでは，あらかじめ入力した地点と現在地点の降水強度しか出力できないため，将来的にはプログラム実行中に GUI 内で座標や住所を入力できるようにする機能や UI を改良することでユーザーがさらに利用しやすいアプリケーションにしていきたいと考えている．また，上記で述べたように降水強度は 60 分後まで予測できるので今後更に表示できるようにしていきたいと考えている．

3.4 カメラ

3.4.1 **概要**　競技車両の乗車員以外からは，実際に走っているときのドライバーの様子やシステム上の他の機能を直接見ることができない．そこで，車内にカメラを搭載することにより，計器の動作状況や走行状況を撮影することで，競技終了後に改善点を見つけやすくする．

3.4.2 **内容**　RaspberryPi の電源を ON にして OS が起動したときに，後述のコマンドが自動で実行されるように設定し，接続した Web カメラ(Logicool C270)で動画を撮影する．撮影した動画は RaspberryPi に接続した USB メモリ内の指定のフォルダに保存する．走行終了後，USB メモリから動画ファイルを取り出す．撮影時に使用するコマンドを以下に示す．

```
sleep 30s
sudo ffmpeg ¥
      -f v4l2 -input_format mjpeg -video_size 640*480
       -framerate 30 -i /dev/video0 ¥
      -c:v copy ¥
     -f segment_time 900 -segment_wrap 30
       ¥/media/pi/USB メモリ内のディレクトリ名
       /output%02d.mkv
```

以下に ffmpeg コマンドの詳細を示す．

- 3-4 行目　映像の取得関係(mjpeg から入力，映像サイズ 640*480，フレームレート 30/s)
- 5 行目　映像のコーデック(copy:エンコード無しで生データを保存する)
- 6 行目　ファイルの分割関係(1 つの動画ファイルの時間 900s，最大ファイル数 30 個)
- 7-8 行目　保存する場所のパス/保存するファイル名
  ファイル名は output の後に撮影順に数字がつけられたものになる．(始めに保存されるファイル名は output00.mkv)

このコマンドを/etc/rc.local の最終行 exit 0 以前に書き込むことで，RaspberryPi 起動時にコマンドが実行される．ffmpeg の前に sleep を入れているのは RaspberryPi 起動時に USB メモリを認識するまで ffmpeg を実行するのを止めるためである．また，コマンドを実行するディレクトリとファイルを保存するディレクトリが違うため，パスの指定には注意が必要である．問題点として，一度電源を切った際には，撮影したものは別のフォルダに移す必要がある点がある．これは，起動時に録画が開始され，保存する動画ファイル名が前回と同様となる仕様上，上書き保存されてしまうためである．

図 3-9 は実際の走行中に撮影された動画から切り取ったものである．今回は保存容量を考慮して画質を落としているため，細かな文字を読み取ることは難しいが，走行中の車内の様子を確認することができた．

図 3-9　走行中の車内映像

4. 結果

本システムを搭載し 2020 年 11 月 14 日 15 日に京都府綴喜郡宇治田原町で開催された 2020 年 JMRC 近畿 SS ラリーシリーズ第 4 戦「やましろのくにラリーin 宇治田原」に出場した. 図 4-1 に出場時の様子を示し, 実戦で本システムを使用した感想及び今後の要望を下記にまとめる.

図 4-1　やましろのくにラリーin 宇治田原出場の様子

4.1 CAN について

4.1.1 感想

・競技中の車両の情報を細かく知ることができ有効
・フェールセーフフラグのオンオフが色分けされており視覚的に判断しやすい
・競技当日以外にも車両作りから使用できるので便利

4.1.2 今後の要望

・ラベルと数値のフォントサイズを変更しメリハリをつけると表示が見やすい
・現在, 数値は白で表記されているが設定値を設け設定した値を超えると色が変わるなどすればより有効

4.2 GPS について

4.2.1 感想

・リエゾン区間で現在地を把握することができナビゲーションに有効
・以前はコドライバーだけが現在地を確認しながら進んでいたが地図が常時表示されているのでドライバーも現在地点を把握することができる
・以前は道に迷った際 Google map を開いていたが常時表示されているので手間がかからない

4.2.2 今後の要望

・リエゾン区間は同じ道を走ることが多いので軌跡が残ると便利
・事前に任意の個所を地図上にマーキングできると便利

4.3 天気情報について

4.3.1 感想

・競技中に設けられた整備の時間に次の走行場所の降水量を知ることができるので天候に合わせたタイヤの選択ができ有効
・設定した地点の情報を一度に見ることができ便利

4.3.2 今後の要望

・風速や気温等の情報もわかるとなお良い

4.4 カメラについて

4.4.1 感想

・Raspberry Pi の電源を入れるだけで競技終了時点まで撮影でき他の操作の必要が無く手軽
・フレームレートも車両の速度とあっており後日, 車両の状況を見直す際に使用できる

4.4.2 今後の要望

・カメラ本体を見直し解像度を上げる
・出力形式が mkv になっていたので任意のファイル形式を選択できると便利
・映像の臨場感が増すため音声があると良い

5. まとめ

本システムにより今までは全く不明であった競技中の車両の情報がドライバーに伝わるようになった. また, 撮影動画を見直すことにより車両に乗車していないユーザーも車両情報を共有することができた.

また競技中において現在位置や天気情報をリアルタイムに確認することができ競技の助けとなった.

本システムを実現したことはラリー競技を進める上で有効的な手段だと言える.

今後は, ドライバーからの意見をフィードバックし, より有効な情報をより使いやすく提供できるように改善に努めたい.

参考文献

(1) Welcome to pySerial’s documentation - pySerial 3.0 documentation
https://pythonhosted. org/pyserial/
(2) GitHub - inmcm/micropyGPS: A Full Featured GPS NMEA-0183　sentence parser for use with Micropython and the PyBoard embedded platform
https://github.com/inmcm/micropyGPS
(3) 地理院地図 / GSI Maps｜国土地理院
https://maps.gsi.go.jp/
(4) YOLP(地図):Yahoo! JavaScript マップ API - Yahoo!デベロッパーネットワーク
https://developer.yahoo.co.jp/webapi/map/openlocalplatform/v1/js/
(5) 気象情報 API - Yahoo!デベロッパーネットワーク
https://developer.yahoo.co.jp/webapi/map/openlocalplatform/v1/weather. html