

# 放課後開発で作るミニマルな小型屋外自律移動ロボット Cub

○田中章愛, 高木洋, 吉竹大志, 渋谷拓己, 永谷智貴, 本射嘉那子

\*Akichika TANAKA, Hiroshi TAKAGI, Taishi YOSHITAKE,  
Takumi SHIBUYA, Tomoki NAGATANI, Kanako MOTOI

チーム名:CoderDojo 武蔵小杉 ロボット名:Cub

## 概要

本稿では、社会人の放課後活動として開発した小型屋外自律移動ロボット Cubについて述べる。特徴として、つくばチャレンジの課題をクリアできる仕様を持ちつつ一般家庭にも置けて一人でも電車や車で運搬できる小型でかわいらしいサイズ、私の予算のためできるだけ低コスト化したハードウェア、汎用的で分散開発が可能なシステムアーキテクチャで構成され、必要最小限・ミニマルなロボットとなっている。結果は残念ながら途中棄権となったが、短距離ながら初めての屋外自律移動を行い、将来への希望を残した。

## 1 はじめに

我々「CoderDojo 武蔵小杉」チームは主に神奈川県川崎市・横浜市在住の社会人で構成されており、小中学生向けの非営利プログラミングクラブを毎月数回運営しているエンジニアを中心とした有志団体・社会人サークルである。普段は子どもたちのゲーム作りや小学生ロボコン出場のアドバイスなどを行う中で、自分たち大人も何かプログラミングしたり一緒にチャレンジすることで相互の刺激を与えあうことができればという思い、また ROS 等モダンなロボット開発について学習を兼ねた活動として、今回社会人チームとしてつくばチャレンジに出場することを決意した。一部経験者がいるものの、未経験・初参加メンバーも多く、またロボットそのものも私の予算でのゼロからの開発となつたため企業や大学等での開発とは一味違う多くの苦労があった。本走行結果は残念ながら未達に終わったが、社会人の放課後活動ならではの制約を克服する設計を心掛けたこともあり、小型化やシステム構成においてはミニマルで一定の独自性を持ったものになった。図 1 は我々が開発した Cub の外観である。本稿では構想や具体的な実装について述べ、微力ながらコミュニティに貢献すべく知見を共有したい。

## 2 ハードウェア

本章では開発した小型ロボット Cub のハードウェアについて述べる。まず出場にあたり、放課後活動として開発する自分たちの環境や条件に合わせてロボットの要件を下記のように設定した。



図 1: Cub

- (当然)つくばチャレンジ出場要件を満たすこと
- 家庭の玄関で保管できるサイズ:400x400mm 以内
- 電車で一人で運搬できること: 重量 15kg 以内、キャリーカートで運べる高さ (800mm 以下)
- ROS 等 State of the art な技術を学習でき、モダンな開発手法にマッチしたアーキテクチャ
- 電源管理・デバッグ等、現地開発しやすい構成
- 入手性やコスト面で有利な扱いやすい部品を使う(公式のデバイス貸し出しプログラムも積極活用)
- 安全面や雨天等を考慮し、できる限り部品の露出を避けるカバーを設けること

図 3 と図 4 に具体的なハードウェアの構成を示す。なお、GPS やカメラのチルト用サーボモーターなど、今回は使用していないが将来のために搭載されているデバイスもある。



図 2: Cub の初期デザイン外観  
タブレットは最終的に撤去し背面にモニタを搭載した

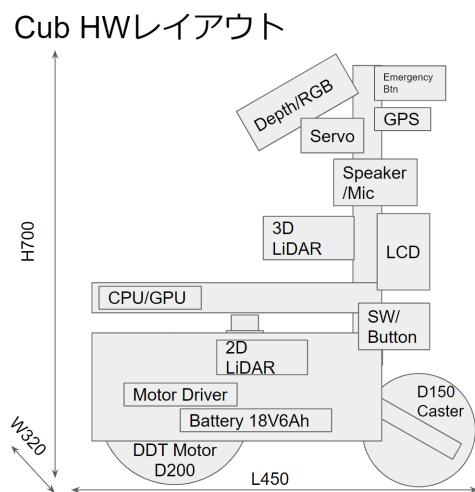


図 3: ハードウェアレイアウト図

今回初出場であったためサイズ制限等は設計自由度を下げる要因となり設計面では厳しかったが、放課後活動として活動場所が市民館等となってしまうことや電車運搬となること、保管が自宅になってしまふことなどから必要な制約となった。しかし結果的にはコンパクトなサイズで片手で持てる重量を実現でき、安全面でも見た目の恐怖感の面でも優位性のある、比較的かわいらしいデザインとすることができた。以上の要件から、下記特徴を持つロボットを設計した。

- 前輪駆動型対向二輪ロボットとし、キックボード用車輪+ダイレクトドライブモーターと車いすキャスターを用いたコンパクトな3輪構成
- 小型 LinuxPC(Jetson Orin Nano)を内蔵し LiDAR やカメラ・緊急停止ボタン等のみを上部に搭載した低背・低重心なハードウェアレイアウト
- 開発やメンテナンス性で優位な起動中の電源給電・電池交換（ホットスワップ）可能な電源構成

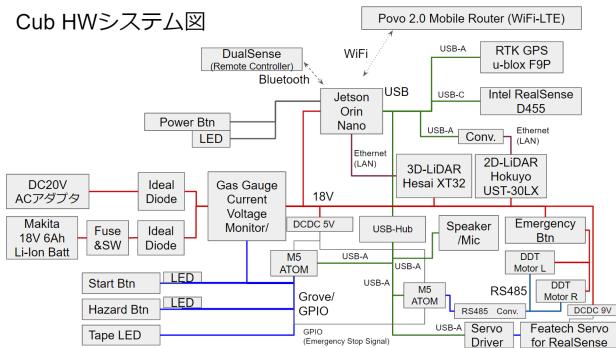


図 4: ハードウェアシステム図

今回安価な RS485 コマンド通信タイプのダイレクトドライブモーター DDT M0601C 111[1] を電動キックボード用の車輪に内蔵して使用したことで非常にコンパクトな台車を構成することができ、十分なスピードやトルク・応答性を確保することができた。制御系には M5Stack 社 ATOM Matrix[2] を使用し、既存の DDT M0601C 用ライブラリを利用できたほか、PlayStation5 用無線コントローラー DualSense とも Bluetooth にて直接通信することで早期の制御系立ち上げを行うことができた。バッテリーには電動工具用の汎用的な 18V リチウムイオン電池を使用することで入手性や容量・ピーク電流・耐久性などの性能面で十分なパフォーマンスを発揮することができた。また、理想ダイオード [3] を用いたホットスワップ回路を用いることで電源の AC 給電～バッテリー切り替え・交換をシャットダウン無しに行うことができ、開発効率を向上できた。メイン CPU については NVIDIA Jetson Orin Nano[4] を使用することでスムーズな ROS での開発を可能とし、小型軽量かつ将来の GPU 利用も含め十分な性能を確保した。フレームや構造部材はミスミのアルミフレームと 3D プリンタ部品を使用し、安価で頻繁な設計変更を可能とすることで短期に新開発の駆動系を完成させることができた。センサとして障害物回避やナビゲーション用の LiDAR には 3D の Hesai XT32(無償貸与)、2D の北陽電機 URM-40LC-EW(無償貸与)、距離センサとして Intel RealSense D455(中古品購入)を搭載した。つくばチャレンジの過去のレポートやデバイスの動向を調査しながら、参照情報の多いモダンな汎用部品を多用することで学習や立ち上げにかかる時間を短縮し、比較的小型な構成で小型な屋外自律移動ロボットを実装・構築することができた。

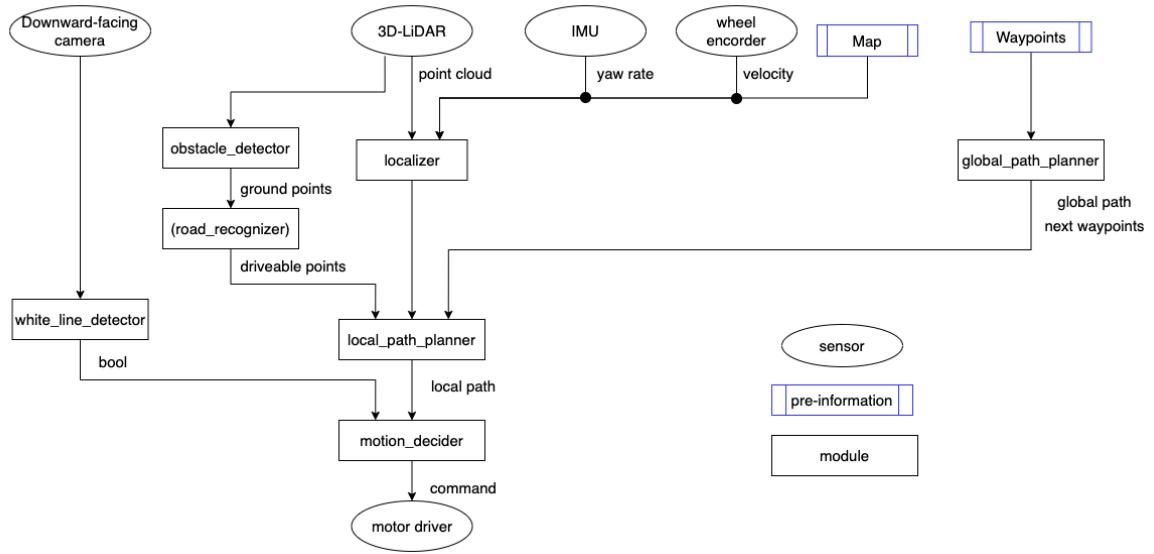


図 5: System architecture

### 3 ソフトウェア

本章では、作成したナビゲーションシステムのソフトウェアについて記述する。図 5 はシステム図である。

システム全体のフレームワークとしては ROS1 Noetic[5] を使用し、内界センサ (IMU・ホイールのオドメトリ) の情報と 3D-LiDAR のスキャン点群と点群地図による NDT[6] スキャンマッチングの情報を拡張カルマンフィルタで統合し自己位置推定を行った。また、3D-LiDAR の計測点群から走行可能領域を認識し、事前に設定したウェイポイントを障害物を回避しながら順番に巡回することで自律移動を行う。

#### 3.1 内界センサ・自律航行系の構成

自己位置推定・自律航行を行うためのオドメトリ取得に使用した内界センサデバイスを(表 1)に示す。

表 1: 内界センサとして使用したデバイス

型番	ベンダー	機能
M5Atom	M5Stack	オドメトリ取得及びモーター制御
DDT M0601C 111	Direct Drive Tech	モーター・エンコーダ
BNO055	Bosch	IMU

コマンド通信タイプのモータードライバからの回転角情報を活用し、既存の Arduino ライブラリ [7] を用いて M5Atom 側で計算、速度制御及び位置情報を取得しオドメトリ計算を行った。当初はモータードライバの速度情報を元に積分によるオドメトリ計算を行っていたが、誤差蓄積が大きく使い物にならなかつたため、位置情報を元に計算する方法に切り替えた。また

Yaw 角の精度向上のため IMU を追加しカルマンフィルタによるジャイロオドメトリで位置・方向ともに自律走行が可能な精度を達成した。ROS1 のシステムへの合わせこみに関しては、rosserial ライブラリ [8] を活用した。

#### 3.2 大域的経路計画

今回作成した Dijkstra 法 [9] による大域的経路計画について述べる。使用する 3 次元点群を参照し、事前に分岐点や通過したい地点などにウェイポイントを設定し、各ウェイポイント間の距離を Dijkstra 法のコストとして指定することで、指定したウェイポイントを最短で通過するような経路を生成する。

ウェイポイント設定には、RViz と ChatGPT を活用した。まず RViz にクリック対象の地面とする十分に大きな空の Map と三次元地図とを配置し、Publish Point 機能で座標を Topic として出力するよう設定した。この Topic をスクリプトによりウェイポイントを表す yaml ファイルに保存し利用した。この仕組みにより、座標の手打ちを避けられ手間を削減出来た。これに必要なスクリプトは全て ChatGPT を用いて要件・仕様のテキストからコードを生成したものを僅かに手直しして実現したものであり、開発の手間・時間も削減出来た。

#### 3.3 局所的経路計画

今回利用した局所的経路計画について述べる。まず、3D-LiDAR の計測点群から高さ情報を元に障害物

に関する点群を抽出し、抽出した障害物に関する点群を用いて二次元の格子地図を作成する。作成した周辺地図と推定した自己位置の結果、次に目指すウェイポイントの位置を元に State Lattice Planner[10] を用いて、障害物にぶつからないような軌道を生成しすることで、安全に各ウェイポイントを巡回する。

### 3.4 白線・物体検出

ここでは白線・物体検出の実装と本番での稼働状況について記述する。検出に使用したデバイスを(表2)に示す。

表2: 白線・物体検出に使用したデバイス

型番	ベンダー	機能
Realsense D455	Intel	光学カメラ、深度カメラ
Jetson Orin Nano	Nvidia	物体検出
Windows PC	HP	開発・デバッグ環境

白線・物体検出は、Realsense と Jetson および Windows PC 上の Python プログラムによって行った。Realsense は SDK の Python ラッパーである pyrealsense2 を使用することで RGB 画像、深度画像、IMU 情報を取得することができる。RGB 画像は各色 8 ビット、深度画像は 16 ビットとし、それそれを位置合わせした状態で受け取る。ホスト側の処理能力を考慮して、リフレッシュレートは 5 Hz とし、解像度は最大の  $1280 \times 720 \text{ pixels}^2$ とした。単純に取得した深度画像は欠損が多く、そのままでは使いにくいので、SDK の Post Processing 機能である Hole Filling によって補間している。なお IMU の情報は今回は使用していない。

白線および物体の検出には RGB 画像を用いた。視野の下半分のみ注目領域とし、入力画像をトリミングした。細かいテクスチャにより県検出精度が落ちないようにガウシアンぼかしを適用した。画像内から白い領域だけを抽出するために、RGB 色空間から HSV 色空間に変換し、 $0 \leq S \leq 50$ かつ $200 \leq V \leq 255$ のピクセルのみを 255 とし、残る領域を 0 とするマスク画像を作成した。OpenCV の Canny 法によってマスク画像から輪郭を抽出し、同様にハフ変換にて直線を検出した。ハフ変換のパラメーターの一例として次を使用した。 $\rho = 1, \theta = \text{np.pi}/180, \text{threshold} = 100, \text{minLineLength} = 200, \text{maxLineGap} = 20$ 。白線に対して機体が正対していることを前提とし、視野内で水平な直線だけを抽出するために、画像の横方向を角度  $\phi = 0^\circ$  として、水平なもの ( $-10^\circ \leq \phi + 180^\circ \times n \leq 10^\circ$ )

のみを残した。最後に白線の中心座標を算出し、深度画像を参照することで、白線の検出と白線までの距離を取得した。

物体認識は Ultralytics 社の YOLOv5[11] を使用し、事前学習モデルは yolov5s を使用した。特に変更は加えず、そのまま使用した。十分に遠い物体を検出から除外するために、画像全体に対して検出された矩形の面積が  $1/64$  以上ある物体のみを残した。検出された矩形領域の中心  $8 \times 8 \text{ pixels}^2$  に対し、深度画像から平均距離を取得した。物体の矩形領域全体の平均を計算すると、背景に引っ張られて正しい距離を取得できない。

実機に Realsense を取り付け、Windows PC 上で Python を実行しながら検出のテストをしたところ、タイルの継ぎ目などで白線の誤検出が発生したので、目的の白線に接近した場合のみ検出結果を有効にする必要が分かった。しかしながら、本走行までに自律移動が可能な状態に至らず、Jetson Orin Nano を使用した完全な状態での動作確認およびパラメーターのチューニングまで完了しなかった。今後、継続的に開発が必要な要素となっている。

## 4 つくばチャレンジ EX 向けた改修

今回初出場ということもあり、できるだけ現場経験値を積むためつくばチャレンジ EX にも参加することにした。特につくばチャレンジ with PLATEAU は地図データの活用の点で過去の蓄積のない自分たちにとっても得るもののが大きいと考え挑戦することとした。

### 4.1 PLATEAU データの変換

G 空間情報センターの PLATEAU つくば市データ[12]をダウンロードし、PLATEAU SDK for Unity[13]を使って OBJ データを作成、その後 PCL mesh2pcd[14]を使い OBJ から点群データに変換した後で座標の不整合を Cloud Compare というツールで修正して ROS 用の点群地図ファイルを作成した。当初つくばチャレンジ EX with PLATEAU のチュートリアル[15]を参考に FME というデータ変換ソフトウェアを利用しようとしたが、ライセンスの取得に時間がかかることが分かったため使用できなかった。Unity 上では不要なデータを細かいオブジェクト単位で編集・削除することができるため、結果的にはロボット視点で不要なデータをすべて削除することができ、必要最低限のデータセットを構築することができた。

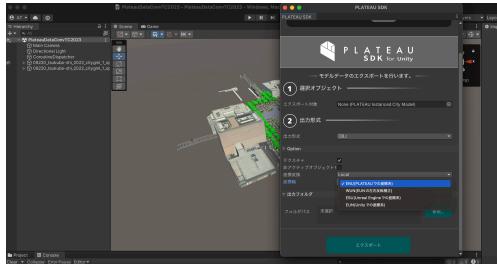


図 6: PLATEAU SDK for Unity でのデータ加工

## 4.2 ROS 対応

PLATEAU データから変換した点群データは、左右反転した状態であったため、点群データに変換後に cloudcompare を用いて修正を加えた。また、変換した点群データの点密度が低かったため、3D-LiDAR 点群とのマッチングが取れない場面が多く見受けられた。それを対応するために、変換した点群を複数コピーしそれぞれ数 mm ずらして重ね合わせたものを一つの点群地図として扱うことで点密度を向上させ、スキャンマッチング精度の向上を図った。

## 5 実験結果



図 7: つくばチャレンジ EX with PLATEAU の様子

本機体は表 3 のとおり、本走行の 1 回とつくチャレンジ EX の 2 回に参加した。それぞれの結果と考察や感想を記述する。

表 3: Cub 参加イベント (記録認定のあるイベントのみ)

日程	イベント名
8/17-18	つくばチャレンジ EX @イースつくば
11/19	本走行
12/3-4	つくばチャレンジ EX with PLATEAU @つくばセンター広場

はじめに、各イベントでの結果を記す。「つくばチャレンジ EX@イースつくば」および「本走行」で

は、スタートすることができなかった。「つくばチャレンジ EX with PLATEAU@つくばセンター広場」では、数十メートルの自律移動に成功したものの、途中でコースアウトにより緊急停止、棄権・失格となつた。

各イベントに対し、考察や感想を含めて詳細に記す。「つくばチャレンジ EX@イースつくば」では、ハードウェアはおおむね完成しており、走行装置をラジコン操作できる状態であったが、ソフトウェアの開発が追いついておらず、LiDAR などのデバイスの立ち上げや ROS の完全な構築ができていなかった。そのため出走できずやむなくリタイアとなった。感想としては、今年初めて参加した本チームから見て、他の参加チームの機体の完成度が驚くほど高く、ノウハウの蓄積の優位性を痛感した。特に、施設の入場口付近で 4 台の機体がお見合いになる難しい状況に、各自が自律的に回避や後退を行って走行を続ける場面が印象的で、自律移動を行うことのハードルの高さを感じた。

「本走行」では、事前に一定距離の自律移動はできるようになっていたものの、直前のメンテナンスが原因でスタートラインを越えることなくタイムアップでリタイアとなった。本来であれば、ナビゲーションや自己位置推定を担当する Jetson とモーターを制御する M5Stack 相互に指令やオドメトリをやり取りするが、直前に M5Stack のプログラムを書き換えたことで両者の接続が不安定になりスタート後正常動作できなかった。このミスの背景には、システム全体の完成度が上がっていなかったために、エラーに対して素早くリカバリーする体制が整っていなかったこともあった。

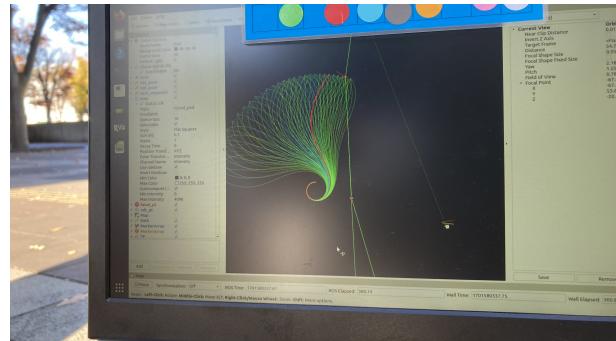


図 8: EX 走行失敗時の経路生成の様子（原因解析中）

「つくばチャレンジ EX with PLATEAU@つくばセンター広場」では、各デバイスは正常につながり、自律移動も開始できたが、経路生成の異常が発生したために、コースアウトすることとなった。その時の経路の

様子を図 8 に示す（緑線：候補軌道、青線：衝突回避軌道、赤線：選択軌道）。なお、まだ原因は特定できておらず、継続的に調査を進める必要がある。

## 6 今回活用できなかったアイテム

今回つくばチャレンジ初出場にあたり、公式の貸出しプログラムを最大限活用させていただいた。しかし、実際の開発においては時間制約やスキル、またシステムとの相性なども含め十分に活用できなかったアイテムが多数発生してしまった。ここに原因を含めて紹介しつつ、無償提供していただいたことへの心からの感謝と十分活用できなかったことへのお詫びの気持ちを表したい。

- ・北陽電機様:Lidar 「URM-40LC-EW」を貸与いただいたが、ROS 向け設定で手間取り(結果的に単純ミスと判明) システム統合ができなかったこと、また 3D LiDAR での障害物回避で走行自体はできたことから活用できなかった。今後エリア設定機能を使った衝突防止センサとして安全機能向上に活用したい。
- ・マップフォーランジ様:3 次元 SLAM ソフトウェア「MAP IV Engine Cloud」のライセンスを貸与いただいたが、結果的にはチーム内経験者の知見を活かし全体を ROS1 で開発することとなり、また事前配布された地図や PLATEAU を活用したことから使用できないままに終わってしまった。次回以降は改めて 3DSLAM で独自地図を作成したい。

## 7 おわりに

今回社会人サークルとして初めてつくばチャレンジに参加したが、とても楽しく参加でき学びも非常に大きかった一方、実際にはロボット開発だけでなく当初想定上に開発環境や体制の構築に苦労した。なによりまずは開発場所の確保として近隣の市民館を利用したが会議室の予約が取れないといった初步的なところからの苦労も多かった。また、チームメンバーも基本的にはなかなか普段会えないためリモート開発と集合開発を組み合わせたが、実機が 1 つしかないため最終的なすり合わせが結果的に現場での実験に依ってしまった点は開発の大幅な遅れにもつながった。これらの点は経験値不足からの点もあり、今後改善していくたい。いずれにしても、今回既存のロボットを使わずに自作でコンパクトなロボットを開発できたことで、一般的なつくばチャレンジ出場ロボットより低コストか

つコンパクトと思えるミニマルなものができ、家庭での保管や電車での移動にも対応するなど社会人チーム・放課後開発に向いたロボットの参考例をつくることができた。自分たちとしてもこれをバネに今後の出場や完走を目指すとともに、ぜひこれを読んでいただいた社会人の方・サークルのみなさんにも参考にしていただければ本望である。つくばチャレンジというリアルな実験と開発を通じて得られた実学的な学びは何物にも代えがたいということは最後に申し添えたい。

## 8 謝辞

応援してくれたサブメンバー福林さん、快く開発場所を貸していただいた川崎市立中原市民館・かわさき市民活動センターのみなさま、応援してくれたCoderDojo 武蔵小杉の参加者のみなさん、そして背中を押してくれた家族へ心から感謝の意を表します。また最後に、つくばチャレンジという素晴らしい挑戦の場を企画・運営・支援いただいた運営委員会・スタッフ・つくば市のみなさまに心からお礼申し上げます。

## 参考文献

- [1] Direct Drive Tech DDT M0601C 111, (2024-1 閲覧). <https://www.switch-science.com/products/7646>.
- [2] M5Stack ATOM Matrix, (2024-1 閲覧). <https://www.switch-science.com/products/6260>.
- [3] LM74610 スマート・ダイオードモジュール, (2024-1 閲覧). <https://strawberry-linux.com/catalog/items?code=17461>.
- [4] NVIDIA Jetson Orin Nano, (2024-1 閲覧). <https://www.nvidia.com/ja-jp/autonomous-machines/embedded-systems/jetson-orin/>.
- [5] Ros noetic nujemys, (2024-1 閲覧). <https://wiki.ros.org/noetic>.
- [6] Peter Biber and Wolfgang StraBer. The normal distributions transform: A new approach to laser scan matching. **Proc. of IROS.IEEE 2003**, pp. 2743–2748, 2003.
- [7] M5stack ddt motor m06/m15 library, (2024-1 閲覧). [https://github.com/takex5g/M5\\_DDTMotor\\_M15M06](https://github.com/takex5g/M5_DDTMotor_M15M06).
- [8] Rosserial arduino library, (2024-1 閲覧). <https://www.arduino.cc/reference/en/libraries/rosserial-arduino-library/>.
- [9] E. W. Dijkstra. **A note on two problems in connection with graphs**. Numerische Mathematik, 1959.
- [10] amslabtech/state\_lattice\_planner, (2024-1 閲覧). [https://github.com/amslabtech/state\\_lattice\\_planner.git](https://github.com/amslabtech/state_lattice_planner.git).
- [11] Yolo v5, (2024-1 閲覧). <https://github.com/ultralytics/yolov5>.
- [12] 3D 都市モデル (Project PLATEAU) つくば市 (2022 年度), (2024-1 閲覧). <https://www.geospatial.jp/ckan/dataset/plateau-08220-tsukuba-shi-2022>.
- [13] Plateau sdk for unity, (2024-1 閲覧). <https://github.com/Project-PLATEAU/PLATEAU-SDK-for-Unity/>.
- [14] mesh2pcd, (2024-1 閲覧). <https://github.com/PointCloudLibrary/pcl/blob/master/tools/mesh2pcd.cpp>.
- [15] つくばチャレンジ EX, (2024-1 閲覧). <https://tsukubachallenge.jp/2023/about/extr>.