OpenPodcar: an Open Source Vehicle for Self-Driving Car Research

Fanta Camara[1,2], Chris Waltham[2],
David Churchill[2], and Charles Fox[1,2]

[1] *Institute for Transport Studies, University of Leeds, UK*
[2] *School of Computer Science, University of Lincoln, UK*

March 21, 2022

Abstract

OpenPodcar is a low-cost, open source hardware and software, autonomous vehicle research platform based on an off-the-shelf, hard-canopy, mobility scooter donor vehicle. Hardware and software designs are provided to convert the donor vehicle into a low-cost and fully autonomous platform. The open source platform consists of (a) hardware components: CAD designs, bill of materials, and build instructions; (b) Arduino, ROS and Gazebo control and simulation software files which provide standard ROS interfaces and simulation of the vehicle; and (c) higher-level ROS implementations of standard robot control, including the move_base interface with Timed-Elastic-Band planner which enacts command to get the vehicle move from one pose to another. The platform is large enough to transport one person at speeds up to 15km/h, for example for use as a last-mile autonomous taxi service or to transport delivery containers similarly around a city center. It is small and safe enough to be parked in a standard research lab and be used for realistic human-vehicle interaction studies. System build cost from new components is around USD7,000 in total in 2022. OpenPodcar platform thus provides a good balance between real world utility, safety, cost and research convenience.

Metadata Overview

Main design files: https://github.com/OpenPodcar
Target group: researchers and hobbyists interested in autonomous vehicle research and robotics.
Skills required: Mechanical assembly – intermediate (drilling steel); electrical assembly – intermediate (PCB soldering); Software – easy (Linux command line).
Replication: The current OpenPodcar is being used by some of the authors for human-robot interaction experiments and a second copy will be built from the documentation to improve its accuracy. The design is currently being forked for a courier-type manually-driven platform by a commercial UK vehicle manufacturer.

Keywords

Autonomous Vehicles, Open Source Hardware and Software.

---

(1) Overview

Introduction

Autonomous Vehicles (AVs, also known as 'self-driving cars'), is a fast-moving research field in both academia and the industry. Open source software (OSS) for localisation, mapping and control is available [24] but hardware vehicle platforms remain expensive and protected, making it difficult for researchers with low resources to develop algorithms or reproduce complete research systems. There is thus a need for a standard, low-cost, reproducible hardware platform, compatible with the standard open source software stack.

Open source hardware (OSH) allows for more effective and accessible sharing and collaboration among researchers [15]. By combining OSH and OSS, a standard platform can be produced for use by all members of a research community, who may then reproduce each others work in full, and contribute their new research as functional system components rather than only as reports. Such platforms may evolve from research into development and real-world applications.

To create an OSH platform for the autonomous vehicle research community, several requirements must be met: *low cost* and *easy to build* to enable the community to reproduce and use it; consumer levels of safety and reliability are not required, though *research standards of safety and reliability* are required; the system should be designed to enable *easy modification* so that it can be forked to operate with similar but different vehicles; the system should be physically *light-weight* to ease experimentation and reduce risks of damage, though large

enough for *human transport* so that it can be used in real-world applications and in research requiring realistic interactions with other human road users [9, 16].

Related Systems

SMART [35] is a design to modify an existing donor golf cart vehicle for automation research, this is of a similar size and power to OpenPodcar. Similarly, iCab (Intelligent Campus Automobile) [21] is research golf car with a ROS (Robot Operating System)-based architecture and that has been tested with Timed-Elastic Band planner [27]. However, both SMART and iCab platform designs are not open source.

Complete and built mechanical OSH designs for on-road, person-carrying, cars exist, including PixBot [36] and Tabby EVO [32]. Building these OSH vehicles is a large task for experts and may require dangerous processes such as welding, purchase of expensive components, and considerable storage space. OpenPodcar is based on a proprietary but commodity mobility scooter which is cheaper and easier to convert than performing these builds.

Several OSH RC-scale cars have been completed and built such as F1Tenth [1], AutoRally [20], BARC [22], MIT Racecar [2], [30], and [45]. These platforms are not large enough to drive on public roads or to transport people or goods like OpenPodcar.

Open Source Ecology (OSE) [23] is an ambitious project which ultimately aims to develop fully OSH vehicles including a car and tractor. OSE is optimised for reliability and for users in developing countries so it uses hydraulic power rather than electric as used in OpenPodcar. But its vehicle designs are not yet complete.

Autoware [24] is a heavyweight open source software project to construct a full ROS-based automation stack for on-road cars. Apollo [3] is an open source self-driving software stack and an open hardware interface which may be implemented on vehicles, as done in [25]. These systems could be software interfaced to run with OpenPodcar.

Some AV research can be done in simulation without the need for hardware, hence open source simulation platforms are widely available such as SUMMIT [4], Gym-Duckietown [14], CARLA [17], DEEPDRIVE [38], LGSVL Simulator [40], AirSim[42], and FLOW [46]. The USA state of Georgia provides a level 3 open-source autonomous vehicle based on a Ford-Edge[34], which can be used *gratis* in their Peachtree Corners' Curiosity Lab smart city environment.

(2) Overall Implementation and Design

Donor vehicle

A Pihsiang TE-889XLSN hard-canopy scooter (branded in UK as Shoprider Traverso, [43]) is used as a donor vehicle. It is an Ackermann-steered [29], hard-

(a) Tilting the vehicle using two axle stands, to enable access to the underside. (Note also lidar mounted to roof.)

(b) Underside with linear actuator added for steering.

Figure 1: Vehicle mechanical modification

canopy, electric mobility scooter. It is powered by two 12V batteries connected in series to provide 24V operating voltage and containing 75Ah. In its standard configuration, its steering is controlled by a human operated loop handle bar. The speed and braking systems are both powered by an electric motor and an electric brake via the trans-axle assembly, controlled by an AC2 digital controller receiving different voltage signals to drive forward or brake. The manual speeding and braking systems are controlled by three buttons connected in series on the handle bar. A toggle switch in parallel with a resistor (10kΩ) to choose speed mode high or low; A speed dial knob via a variable resistor (20kΩ) to choose a maximum speed value; A throttle lever connected with a potentiometer (5kΩ), 2.5kΩ to 2.6kΩ for each side to speed or brake.

Mechanical Modification for Steering

To automate steering, a Gimson GLA750-P 12V DC linear actuator with position feedback is mounted between an anchor on the underside of the chassis and the car's front axle via bearings. This actuator has a 8mm/s full load (750N) speed and 250mm stroke length (installation length is 390mm). To access the underside of the vehicle, two axle stands are used as shown in Fig. 1a. There is an existing hole in the right front wheel axle. The linear actuator was mounted via a rear hole to the left side of the front chassis and connected through the front hole of the actuator with the hole in the car's right front wheel axle via bearings as shown in Figs. 1b and 2.

Electronics

The new vehicle electronics, which require different voltage power supplies (cf. Fig. 3), are presented on a PCB (Printed Circuit Board), as shown in Figs. 4a and 4b. This is convenient as it reduces the number of small wires between the components by having them directly drawn on the board, and packages them together.

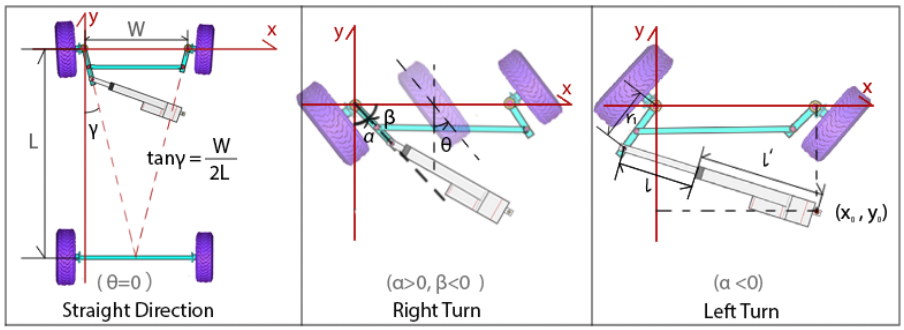


Figure 2: The bottom view of front wheels steering relationship including geometric coefficients

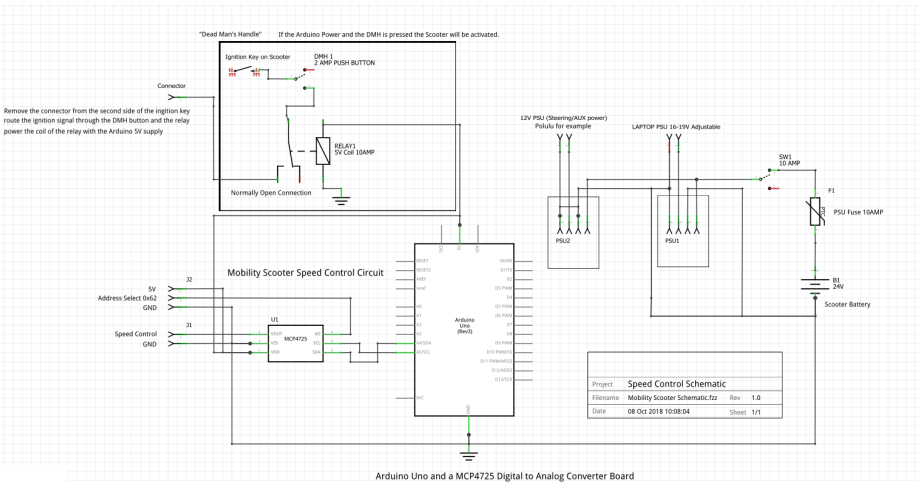
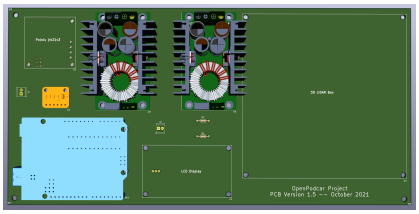


Figure 3: Circuit diagram for electronic modifications.

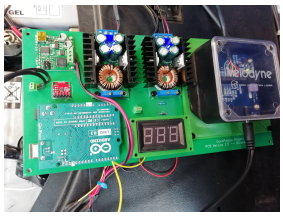
As an OSH design, the PCB hosts several daughter boards, mounted using headers. The physical structure of the large PCB comprised of these smaller components reflects the OSH design itself. There are two DC-DC Buck converters with an XL4016 regulator, an Arduino Uno, an MCP4725 DAC (Digital-Analog Converter), a Pololu Jrk 21v3 motor controller with position feedback for the linear actuator, 2 resistors (10kΩ and 100kΩ) for the potential divider and 2 terminal blocks. 3D printed parts support the mounting of the LCD and the 3D lidar to the board. A 3D printed enclosure protects the PCB board, as shown in Fig. **??**.

Steering Automation System

The front wheels are steered by a Pololu Jrk 21v3 PID controller, and it takes serial port desired positions as input. It also takes feedback position information as an analog voltage from the linear actuator as an input. It outputs analog high-power voltages to the linear actuator. A freely available Windows program



(a) PCB Design.



(b) PCB assembly currently used.

Figure 4: Electronics with PCB board design and assembly

Table 2: Linear actuator acceptable command ranges.

FA:cmd	Effect
2500	max right i.e $\sim$ -45 deg
1900	center i.e $\sim$ 0 deg
1000	max left i.e $\sim$ +45 deg

from Pololu is required (once, at build time) to set the PID parameters for the linear actuator, the detail of this process is explained in the documentation.

The relationship between the required central turning angle θ of the pair of front wheels and extending length l of linear actuator is,

$$\theta = \alpha - \arctan(\frac{W}{2H}) \tag{1}$$

$$\beta = \alpha - \frac{\pi}{2} \tag{2}$$

$$x = r_1 \cos(\beta) \tag{3}$$

$$y = r_1 \sin(\beta) \tag{4}$$

$$l = \sqrt{(x_0 - x)^2 + (y_0 - y)^2} - L + l_0 \tag{5}$$

Where r_1, x_0, y_0, W, H and L are the geometric coefficients shown in Fig.1. Among them, the value of y_0 is negative. l_0 is the initial value of the linear actuator position feedback. Table 2 shows the acceptable commands for the linear actuator, giving commands outside this range may mechanically destroy the system.

Speed Controller Automation System

An Arduino UNO [33] is used to send electric signals to the vehicle's motor controller in place of the donor vehicle's paddle controller's potentiometer. An Adafruit MCP4725 DAC is connected to the Arduino as in Fig. 3, and is used to send clean analog speed command voltages to the donor vehicle's internal controller.

Arduino firmware source is supplied in the distribution. When uploaded to the Arduino (using the standard Arduino IDE running on the laptop), the firmware provides a simple serial port API running at 112000baud. It receives commands

Table 3: Speed commands and their corresponding output voltages.

Command	Voltage	Effect
FA:0	0	stupid fast reverse
FA:80	~ 0.9	v fast reverse (ros limit)
FA:132	~ 1.5	slowest reverse motion
		dead zone - allows ignition
FA:170	1.9	stop - zero/home position
FA:201	~ 2.3	slowest forward motion
FA:240	~ 2.7	v fast forward (ros limit)
FA:255	~ 3.0	stupid fast forward

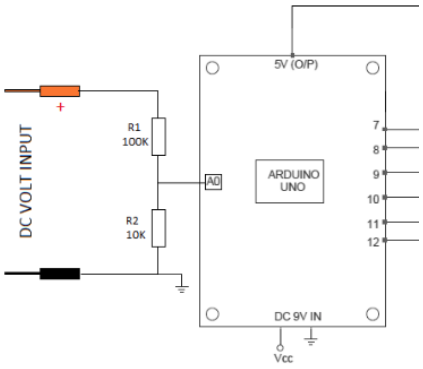


Figure 5: Potential divider linked to the battery

of the form "FA:210" as speed commands. Table 3 summarises the range of speed commands and their corresponding output voltages.

To start the ignition, the car safety system requires the control voltage to be in the dead range. A problem is that this doesn't correspond precisely to any fixed speed bytes, due to the USB power issues. But if we pick a number solidly in the center of the dead zone, such as 164, this will work for most USB supplies. (i.e. when the vehicle's battery is flatter, the voltages provided to USB power by it are lower. For example, we might send 164 and get 1.9V instead of the usual 2.26V.) This may result in the vehicle not starting – maybe by design or accident – this produces a beep as is outside the start zone.

The Arduino gets lower power e.g. max 4.9V instead of 5V, which gets divided by the DAC value. To deal with these instabilities, we added a potential divider at the battery to check the voltage and control the podcar accordingly, as in Fig. 5. We provide a "BV" command in the Arduino serial protocol which allows callers to request the current battery voltage. This can then be used by the higher-level (Python) systems to decide what speed bytes to sent, including compensating for the floating dead zone.

Software Interface (ROS)

A ROS interface to and from the physical vehicle is provided as described below. ROS is an open source operating system for robots based on a publish-subscribe pattern [37], which is the robotics community's standard interface. The ROS core and software all run on a consumer laptop computer mounted on-board the vehicle, and that could be powered from a DCDC converter from the vehicle battery, running Xubuntu 16.04 (Xenial) and ROS Kinetic.

The system expects to hear two incoming ROS control messages: /speedcmd_meterssec and /wheelAngleCmd, which contain single floats representing the desired speed in meters per second, and the desired front wheel orientation in radians respectively. These two messages are received by ROS nodes speedm2arduino and wheelAngle2Pololu, which are ROS drivers for the Arduino speed controller and the Polulo steering controller respectively. Converters from a standard ROS USB joystick driver node to the speed and angle command interface messages are provided, by joystick2speedms and joystick2wheelAngle. These use the y axis of a joystick for speed and x for steering.

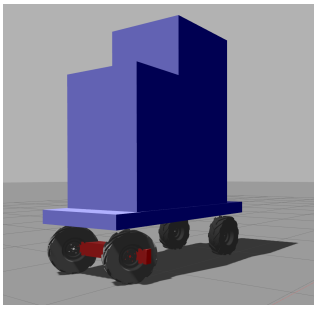
3D Lidar Sensor

A Velodyne VLP-16 lidar sensor is mounted on the vehicle roof using a Manfrotto Black Digi Table Tripod 709B. It is mounted at a 10 degree tilt downwards (to allow pedestrians to be most clearly seen in the 16 scan lines). The lidar has a ROS driver whose installation is detailed in the documentation.

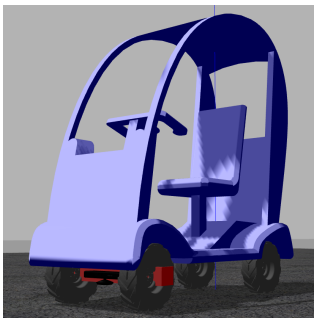
High-Level Automation Software

Localisation and Mapping System

Simultaneous Localisation and Mapping (SLAM) [44] is the robotic task of inferring the robot's location at the same time as building a map of its environment, which is a classic 'chicken and egg' problem as the two subtasks depend on one another. Solving SLAM is an NP-hard problem but many standard approximations exist. GMapping [48] is a ROS implementation of a Rao-Blackwellized Particle Filter (RBPF) in which "each particle carries an individual map of the environment". The information carried by each particle overlaps, and an estimation of a map can be built based on these relationships. As the robot moves around the environment, these estimations are stored, and when a 'feedback loop' is closed, the estimations cascade into a portion of the completed map. These maps take the form of 2D occupancy grids, and can be used later by the navigation stack to plan paths around the environment. GMapping was tested on OpenPodcar both online in 2D SLAM mode and offline for mapping purposes only with very accurate localisation and map results. To provide reliable odometry data for GMapping, ROS laser_scan_matcher package was used to serve as a stand-alone odometry estimator that matches consecutive laserscans.



(a) Physical simulation of vehicle.



(b) Visual mesh simulation of vehicle with a Kinect camera.

Figure 6: OpenPodcar 3D simulation

Path Planning and Control

Path planning is the autonomous selection of an entire desired trajectory for a robot to get from a current pose to a desired pose. Path control (or path following) is then the real-time process of executing a path plan by interactively monitoring the robot's state and sending commands to motors, to make the actualized path close to the desired path. The OpenPodcar software includes path planning and control with the standard ROS tool move_base and Timed Elastic Band (TEB) [41] plugin. These tools implement the requirement geometry of Dubins paths [18] and Ackermann steering. The values for parameters such as minimum turning radius were calculated from the technical specifications of the base vehicle [43].

Pedestrian Detection and Tracking

A pedestrian detector and tracker ROS package are included in the system. The lidar-based detections are classified by a SVM (Support Vector Machine) classifier, then a Bayesian multi-target tracker is used to track pedestrians over time. These modules re-use OSS from the EU FLOBOT project [47]. The package is integrated into OpenPodcar and is now part of the repository.

Simulation

A robotics simulation of the vehicle is provided for use in Gazebo 7 [26] under ROS Kinetic and Ubuntu 16.04 (Xenial). The simulation implements the same ROS interface as the physical vehicle system to enable plug and pay interoperability between them. The physics model is based on a simplified vehicle geometry with two large cuboids containing the vehicles' mass, as shown in Fig. 6a. Wheel geometry, friction, and motor driver parameters were measured from the physical vehicle. A detailed graphical mesh model of the vehicle is provided for display, rather than physical simulation, purposes. The main difference with the real vehicle is that the effects of the linear actuator are represented by a

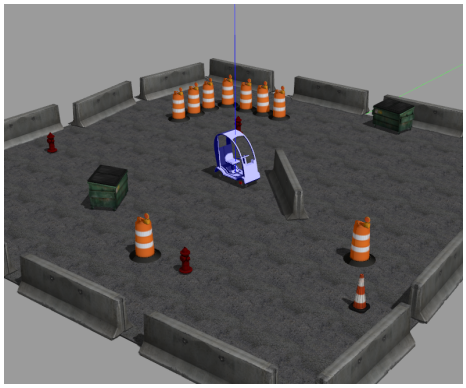
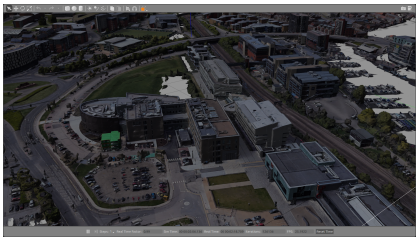
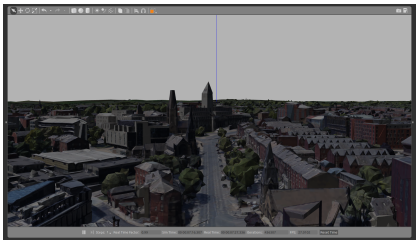


Figure 7: Default Gazebo simulation world



(a) University of Lincoln's 3D world



(b) University of Leeds' 3D world

Figure 8: OpenPodcar additional Gazebo 3D simulation worlds

tracking rod, where is mounted the Kinect sensor used in place of the lidar, as found in Fig. 6b.

A basic 3D world containing the podcar and various test objects from Gazebo libraries is provided by default as shown in Fig. 7. Moreover, the open source Blender 3D add-on, called MapsModelsImporter [28], was used to create further 3D worlds representative of the University of Lincoln campus, the testing area for the OpenPodcar, and the University of Leeds campus.

(3) Quality Control

Safety

Autonomous vehicles can present a significant hazard to humans and to the environment in which they operate. Damage to surroundings and possible injury to operators and bystanders could result from inappropriate use or malfunction. A particular risk arises from the speed controller on the donor vehicle being of 'wigwag' style, as is common in mobility scooters. This means it is an analog signal in the range 0-5V, including a dead zone around 2.5V corresponding to no motion. Above the dead zone and up to 5V are forward speed control commands of increasing speeds, below the dead zone to 0V are reverse control commands of increasing speeds. Wigwag control is potentially dangerous because a 0V signal might appear due to component failure rather than as a desired max-speed reverse command. Also, if the vehicle batteries run low, the scaling of this signal may be altered resulting in the dead zone position floating and leading to further undesired motion. The following layered safey systems are included to fully mitigate these risks:

Fusing As shown in Fig. 3, a 10A fuse is inserted between the vehicle's original 24V battery and the switch to the new electronics. This is in addition to original fusing and other safety features provided by the donor vehicle, which all remain in tact.

Dead Man's Handle It is essential that a suitable emergency stop system is implemented in all autonomous vehicles. Given the research nature of the OpenPodcar, a safety mechanism which stops the vehicle under fault conditions is an especially important part of the design. A wired dead-man's handle (DMH) is included which is required to be pressed by a human experimenter at all times, in order for a hardware relay to actively continue to supply power from the vehicle's batteries to all other systems. The relay connects to the donor vehicle's keyed ignition switch and will naturally cut out if these signals are absent for any reason, including failures in the safety systems themselves. A photograph of the installed system is shown in Fig. 9.

Heartbeat Signal The serial protocol linking the Arduino to ROS includes a heartbeat signal, in which the Arduino code will shut down the motors unless

Figure 9: Steering console showing the newly added relay (with lit LED)

a correctly formatted and timestamped serial command is received within 0.1 seconds. This requires ROS code to actively check and confirm its own status and to send positive confirmation, for example if ROS or Linux go down then this heartbeat will cut off.

Steering Control System Limiter Limitations are placed on the steering controller for the linear actuator commands, to only allow the vehicle to accept and execute input values within the range that will keep the mechanical mounting safe.

General testing

The OpenPodcar platform was developed and our own build was heavily tested between March 2018 and March 2022. With its first automated test drives taking place since summer 2018 and an estimated 100km or more driven to date, the vehicle design has thus proven robust enough for autonomous vehicle research.

For new builds, a series of sub-components (e.g. Pololu, DAC), components (e.g. PCB, lidar), and system integration tests are defined and included in the build instructions. Components and sub-components are tested individually and then after their integration to the vehicle. Below is a summary of these tests.

At the component level, an external power supply, a multi-meter, a clamp meter and a breadboard with some wires are frequently used to recreate smaller electronic circuits in order to check the voltages, currents and the good functioning of each component during the build. For instance, a circuit with an external power supplying 5V to the Arduino connected to the DAC is used to test the Arduino code and its communication with the DAC whose output voltage should be set by default within the desired range of 1.9V–2.1V. Similarly, another circuit can be created with an external power supplying 12V to the Pololu connected to the linear actuator to send direct commands via the Windows app that is used to fix the PID parameters. These simple tests are essential to the success of components' integration to the vehicle and make things easier later.

At the system integration level, udev rules are used to facilitate testing with the creation of simlinks, i.e. dynamic assignments for the laptop USB ports connected to the Arduino and the Pololu, using their respective product and

vendor IDs. This helps in being able to physically interchange the USB ports without having any impact at the software level. For the speed control, the vehicle wheels are lifted from the ground using jacks to stop them from driving off. This technique helps to test and fix the Arduino and ROS speed control code whilst staying in the same place. The vehicle steering is first tested using the Windows app that allows direct commands to be directly sent to the linear actuator. This helps verify and fix the linear actuator mounting as desired. Similarly, using Pololu's C++ API, direct commands are sent from a terminal to the linear actuator, but this time for testing at the software level.

Driving tests are initially performed in the manual joystick control mode in order to ensure that both hardware and software stack work well together. In particular, the LCD on the PCB board helps with checking in real-time the voltage received for each speed command and the LEDs colors displayed on the Pololu also give useful indications about the steering control.

The autonomous driving tests with move_move and TEB are performed with the vehicle speed controls dial know set to "5", corresponding to about 0.2m/s. This relatively low speed is chosen because these tests may be performed in a shared and cluttered research lab around people. Also, a large inflation distance is set in the planner to prevent the vehicle from close contacts with both static and dynamic obstacles. At first, simple and short goals are sent to move_base such as "drive one meter forward and keep your current orientation" or "drive three meters forward and keep your current orientation". Once the vehicle is able to execute and reach these simple goals, more complex goal commands are sent. Once a goal is reached, it is possible to resend immediately another goal without having to turn off the system, which is very convenient for example when one wants to ask the vehicle to return to its starting position or go somewhere else.

Setting a very high accuracy for goals such as 0.01cm and 0.01rad is achievable on the vehicle and is tested for short drives in the lab. However, in these cases, the short drives may end up taking a lot of time, for example it can take up to three minutes to simply reach a one meter forward goal. This is due to the planner's oscillating behaviour around the goal. To fix this, more tolerance should be given for the goal accuracy, for example, 15cm and 0.15rad give an acceptable vehicle behaviour. During these driving tests, ROS topics and RViz (ROS visualization tool) are particularly monitored to get informed about the vehicle behaviour in real-time.

(4) Application

Use cases

Autonomous Vehicle Research

Many AV researchers cannot currently afford the acquisition of a self-driving hardware platform for their work. The OpenPodcar is primarily designed for this purpose, as a low-cost and an all-in-one, software and hardware platform for researchers and hobbyists. Thus, giving them not only the opportunity to

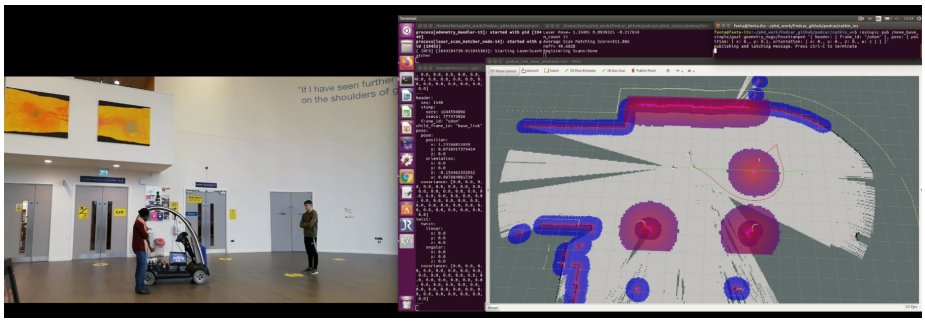


Figure 10: OpenPodcar test drive with GMapping SLAM, ROS move_base with TEB planner and obstacle avoidance.

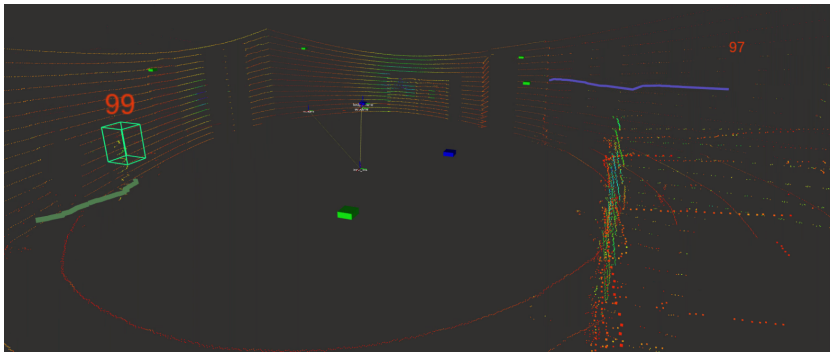


Figure 11: Pedestrian detection and tracking output from RViz.

reproduce, develop and test algorithms on a physical hardware platform but also to extend its capabilities with new features.

The "Related Systems" section showed that there are many open source software stacks without related open hardware platform, the OpenPodcar thus fills this gap and offers the opportunity not only to deploy Autoware or other types of AV software but also to extend the hardware capabilities to the point where OpenPodcar could become a standard test bed for the AV research community. For example, this platform could be useful to test different SLAM and planning algorithms, parallel and valet parking methods. The objective being that both hardware and software can be tested regularly in real-world conditions and contribute towards the deployment of AVs. The OpenPodcar can avoid both static and dynamic obstacle using the integrated feature in move_base and TEB planner. Fig. 10 shows the OpenPodcar test drive with GMapping, move_base and TEB planner in action when it encounters an obstacle on its path.

Human-Robot Interactions

Understanding human behaviour and interaction strategies are of upmost importance nowadays for autonomous systems. There is a general growing interest from the robotics and autonomous vehicle research communities to tackle the

Figure 12: OpenPodcar test drive with remote control.

numerous challenges posed by human interactions. Social robots as well as autonomous vehicles need better models of human behaviour [5, 6]. Some of the authors (FC and CF) are particularly interested in improving autonomous vehicles' decision-making using a game theoretic approach for road-crossing scenarios [19]. Several empirical studies e.g. [7, 8, 13], were performed in highly safe lab environments and found that human participants were not interacting realistically with the other agent. A similar experiment performed in a VR environment showed a more realistic behaviour from the participants [9, 10]. An additional model of human proxemics (i.e., interpersonal distances) has been developed and is being combined with the game theory model [11, 12]. In future work, the OpenPodcar will be used to extend these human experiments using a real physical platform. The pedestrian detection and tracking feature will be particularly useful for this task, since the AV needs to know where the pedestrian is to make a decision. Fig. 11 shows an example output of the pedestrian detection and tracking using the OpenPodcar.

Transportation e.g. Last-Mile Delivery

Covid-19 outbreak exposed one of the most promising use cases for AVs, that is the transportation of people and goods around urban areas. For example, autonomous last-mile delivery minimises the chances of transmission of a disease to vulnerable people via human contact andthe OpenPodcar can carry at least 76kg, more than standard autonomous vehicle research platforms. Another important application is the removal of HGVs from city centers, which aims to reduce car traffic and air pollution by replacing them with smaller and faster vehicles with greener energies like the OpenPodcar. Fig. 12 shows an example of a passenger using the platform in remote control transportation mode. A similar use case is possible in the full autonomous last-mile delivery, for instance transporting people from the train station to their office. This may need more automation software, and would be explored as part of future work.

Reuse potential and adaptability

The OpenPodcar design is intended so that the mechanical, electronics and software components can be easily ported to other vehicles/platforms and only require small changes on the software side to adapt it and fix some parameters specific to the new vehicle requirements. This could include future deeper OSH vehicles as well as additional commercial donor vehicles. Cheaper sensors such as depth cameras or stereo cameras could be used instead of the 3D lidar. Such modifications would typically require an advanced rather than intermediate designer/builder.

(5) Build Details

Availability of materials and methods

The design is made under CERN-OSH-W licence which allows for the use of commercially available proprietary components such as the off-the-shelf donor vehicle. However the design is intended to be easily modifiable for transfer to other base vehicles, including those which are OSH at lower levels. The PCB can be manufactured by many online PCB manufacturers. The additional mechanical and electronics used are common parts available from standard online vendors.

Ease of build

The vehicle modification requires the use of common hand tools for assembly: spanners, screwdrivers, and pliers. Additionally, a 3D printer is needed to fabricate some components. Basic soldering skills are needed for assembling the PCB.

Operating software and peripherals

The system requires open source software: Arduino IDE, Ubuntu 16.04, ROS Kinetic, Gazebo, KiCad (PCB Design), ROS GMapping, ROS move_base, and Velodyne lidar driver. It also requires the Pololu Configuration Utility Manager software which is available gratis from the manufacturer website. The on-board laptop should have minimal specifications of amd64 3GHz quad-core, 8GB RAM, 250Gb hard-disc, USB and Ethernet ports. The system might also work on lower specifications. Step-by-step instructions for installation of these software dependencies, and the new system software components, are provided in the repository.

Hardware documentation and files location:

Archive for hardware documentation, build files and software

Name: GitHub TODO should we move to gitlab for this now?

Persistent identifier: e.g. DOI, etc.

Project repository: https://github.com/OpenPodcar

Licence: CERN-OHL-W

Date published: dd/03/2022

The project is structured as two separate formal OSH designs, each licenced as CERN-OSH-W. The first covers all components which are easily transferable to other vehicles without modification. The second contains all components which are specific to the mobility scooter donor vehicle. This structure enables the first design to be used as sub-component of closed products while also preventing closed modifications of it.

(6) Discussion

Conclusions

The OpenPodcar is a multi-purpose hardware and software platform for autonomous vehicle research. It provides the required hardware and software tools to carry out research in this field. The platform has a lower-level stack, a higher-level stack and a simulator for initial testing. It has several safety features to prevent hazards. The general testing carried on the vehicle shows a robust and safe design. Several use cases have been identified and successfully tested. The OpenPodcar is open source to allow further improvements and extensions of its capabilities from the community.

In the current setup, the lidar has limited perception of obstacles that are too close and not as high as the lidar. This is generally fine, because people or objects would be seen before, but this can be problematic with objects such as desks and chairs that are not detected by the laserscans and can create unexpected collisions.

Future Work

The OpenPodcar opens up the opportunity for extensions, both at the hardware and software levels. For example, a low-cost alternative to lidar is to use a stereo camera for point cloud sensing. In this option, a StereoLabs ZedCam is mounted similarly on the vehicle roof. In the future, we may develop a hardware platform that uses an open source motor controller such as OSMC [39] or ODrive [31]. As time goes, more open source software for AV research will be tested on this platform, including more advanced SLAM algorithms and human interaction methods. The replication of this work on a 2nd vehicle will help identify issues and improve the documentation. When this work started, ROS1 was the widely used version of ROS and the most stable, but at the time of writing of this paper, the robotics community projects are shifting to ROS2 for its security features, we therefore aim to upgrade the software stack to ROS2, and will share some

of our lessons learnt and insights from this process, which may be useful for the community.

Paper author contributions

Fanta Camar performed the real physical podcar automation work including the mechanical design, the electronics with the printed circuit board (PCB), ROS localisation, path planning, all the driving tests, wrote the documentation for the physical podcar and the manuscript. Chris Waltham designed the initial electronics circuit including the deadman handle and the Pololu motor controller, wrote the Arduino code for the speed control and participated to the initial remotely-controlled driving test. David Churchill developed the podcar simulator with path planning within ROS/Gazebo and wrote the documentation for the simulator. Charles Fox supervised the work, wrote the manuscript, and wrote some ROS code. The manuscript was improved by comments from all the co-authors.

Acknowledgements

The authors would like to thank Jacob Lord for creating the vehicle graphical mesh model, Yao Chen for scoping Dubins path methods and the mechanical design, Gabriel Walton for scoping simulation tools, Yicheng Zhang for helping with the 3D printer and many other useful tools for the PCB board, Zak Burrows for the PCB enclosure and assistance during some autonomous driving tests.

Funding statement

This project has received funding from EU H2020 grant 723395 interACT, and from InnovateUK grant 5949683 C19-ADVs.

Competing interests

The authors declare that they have no competing interests.

References

[1] F1tenth. https://f1tenth.org/.

[2] MIT RaceCar. https://mit-racecar.github.io/.

[3] ApolloAuto. Apollo. https://github.com/ApolloAuto/apollo.

[4] Panpan Cai, Yiyuan Lee, Yuanfu Luo, and David Hsu. Summit: A simulator for urban driving in massive mixed traffic. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4023–4029. IEEE, 2020.

[5] Fanta Camara, Nicola Bellotto, Serhan Cosar, Dimitris Nathanael, Matthias Althoff, Jingyuan Wu, Johannes Ruenz, André Dietrich, and Charles W. Fox. Pedestrian models for autonomous driving Part I: low-level models, from sensing to tracking. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[6] Fanta Camara, Nicola Bellotto, Serhan Cosar, Florian Weber, Dimitris Nathanael, Matthias Althoff, Jingyuan Wu, Johannes Ruenz, André Dietrich, Anna Schieben, Gustav Markkula, Fabio Tango, Natasha Merat, and Charles W. Fox. Pedestrian models for autonomous driving Part II: high-level models of human behavior. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[7] Fanta Camara, Serhan Cosar, Nicola Bellotto, Natasha Merat, and Charles W. Fox. Towards pedestrian-av interaction: method for elucidating pedestrian preferences. In *IEEE/RSJ Intelligent Robots and Systems (IROS) Workshops*, 2018.

[8] Fanta Camara, Serhan Cosar, Nicola Bellotto, Natasha Merat, and Charles W. Fox. *Continuous Game Theory Pedestrian Modelling Method for Autonomous Vehicles*. River Publishers, 2020.

[9] Fanta Camara, Patrick Dickinson, and Charles Fox. Evaluating pedestrian interaction preferences with a game theoretic autonomous vehicle in virtual reality. *Transportation Research Part F: Traffic Psychology and Behaviour*, 78:410–423, 2021.

[10] Fanta Camara, Patrick Dickinson, Natasha Merat, and Charles W. Fox. Towards game theoretic av controllers: measuring pedestrian behaviour in virtual reality. In *IEEE/RSJ International Conference on Intelligent Robots and Systems Workshops*, 2019.

[11] Fanta Camara and Charles Fox. Space invaders: Pedestrian proxemic utility functions and trust zones for autonomous vehicle interactions. *Int J of Soc Robotics*, 2020. https://doi.org/10.1007/s12369-020-00717-x.

[12] Fanta Camara and Charles Fox. Extending quantitative proxemics and trust to hri. *31st IEEE International Conference on Robot & Human Interactive Communication (RO-MAN) (under review)*, 2022.

[13] Fanta Camara, Richard Romano, Gustav Markkula, Ruth Madigan, Natasha Merat, and Charles Fox. Empirical game theory of pedestrian interaction for autonomous vehicles. In *Measuring Behavior 2018: 11th International Conference on Methods and Techniques in Behavioral Research*. Manchester Metropolitan University, March 2018.

[14] Maxime Chevalier-Boisvert, Florian Golemo, Yanjun Cao, Bhairav Mehta, and Liam Paull. Duckietown environments for OpenAI Gym. https://github.com/duckietown/gym-duckietown, 2018.

[15] Fisher Daniel K and Gould Peter J. Open-source hardware is a low-cost alternative for scientific instrumentation and research. *Modern instrumentation*, 2012, 2012.

[16] Patricia R DeLucia. Effects of size on collision perception and implications for perceptual theory and transportation safety. *Current directions in psychological science*, 22(3):199–204, 2013.

[17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[18] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[19] Charles W. Fox, Fanta Camara, Gustav Markkula, Richard Romano, Ruth Madigan, and Natasha Merat. When should the chicken cross the road?: Game theory for autonomous vehicle - human interactions. In *VEHITS 2018: 4th International Conference on Vehicle Technology and Intelligent Transport Systems*, January 2018.

[20] Brian Goldfain, Paul Drews, Changxi You, Matthew Barulic, Orlin Velev, Panagiotis Tsiotras, and James M Rehg. Autorally: An open platform for aggressive autonomous driving. *IEEE Control Systems Magazine*, 39(1):26–55, 2019.

[21] D Gomez, P Marin-Plaza, Ahmed Hussein, A Escalera, and J Armingol. Ros-based architecture for autonomous intelligent campus automobile (icab). *UNED Plasencia Revista de Investigacion Universitaria*, 12:257–272, 2016.

[22] Jon Gonzales. *Planning and Control of Drift Maneuvers with the Berkeley Autonomous Race Car*. PhD thesis, University of California at Berkeley, 2018.

[23] Marcin Jakubowski. Open Source Ecology. https://www.opensourceecology.org/, 2003.

[24] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018. https://github.com/Autoware-AI/autoware.ai.

[25] Tobias Kessler, Julian Bernhard, Martin Buechel, Klemens Esterle, Patrick Hart, Daniel Malovetz, Michael Truong Le, Frederik Diehl, Thomas Brunner, and Alois Knoll. Bridging the gap between open source software and vehicle hardware for autonomous driving. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1612–1619, 2019.

[26] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ Inter-*

national Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), volume 3, pages 2149–2154. IEEE, 2004.

[27] Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera. Global and local path planning study in a ros-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, 2018, 2018.

[28] Elie Michel. Maps Models Importer. https://github.com/eliemichel/MapsModelsImporter.

[29] William F Milliken, Douglas L Milliken, et al. *Race car vehicle dynamics*, volume 400. Society of Automotive Engineers Warrendale, PA, 1995.

[30] Naohiro Nakamoto and Hiroyuki Kobayashi. Development of an open-source educational and research platform for autonomous cars. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 6871–6876, 2019.

[31] ODrive Robotics. ODrive. https://odriverobotics.com/.

[32] Open Motors. Tabby EVO. https://www.openmotors.co/evplatform/.

[33] Jonathan Oxer and Hugh Blemings. *Practical Arduino: cool projects for open source hardware*. Apress, 2011.

[34] Peachtree Corners. Curiosity Lab. https://www.curiositylabptc.com/.

[35] Scott Pendleton, Tawit Uthaicharoenpong, Zhuang Jie Chong, Guo Ming James Fu, Baoxing Qin, Wei Liu, Xiaotong Shen, Zhiyong Weng, Cody Kamin, Mark Adam Ang, et al. Autonomous golf cars for public trial of mobility-on-demand service. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1164–1171. IEEE, 2015.

[36] PixMoving. Pixbot. https://gitlab.com/pixmoving/pixbot.

[37] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[38] Craig Quiter. Deepdrive. https://github.com/deepdrive/deepdrive.

[39] Robot Power. Open Source Motor Control (OSMC). http://www.robotpower.com/products/osmc_info.html.

[40] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. *arXiv preprint arXiv:2005.03778*, 2020.

[41] Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Efficient trajectory optimization using a sparse model. In *2013 European Conference on Mobile Robots*, pages 138–143. IEEE, 2013.

[42] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.

[43] Shoprider. Shoprider flagship luxury scooter model: Te-889xlsn user manual. https://www.usermanual.uk/shoprider/te-889xlsn/manual, 2016.

[44] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[45] Bastien Vincke, Sergio Rodriguez Florez, and Pascal Aubert. An open-source scale model platform for teaching autonomous vehicle technologies. *Sensors*, 21(11), 2021. https://github.com/BastienV-SATIE/AutonomousCar/.

[46] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M. Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *CoRR*, abs/1710.05465, 2017. https://flow-project.github.io/.

[47] Z. Yan, S. Schreiberhuber, G. Halmetschlager, T. Duckett, M. Vincze, and N. Bellotto. Robot perception of static and dynamic objects with an autonomous floor scrubber. *Intelligent Service Robotics*, 13(3):403–417, 2020. https://github.com/LCAS/FLOBOT.

[48] Barry Loh Tze Yuen, Khairul Salleh Mohamed Sahari, and Zubaidi Faiesal Mohamad Rafaai. Improved map generation by addition of gaussian noise for indoor slam using ros. *Journal of Robotics, Networking and Artificial Life*, 4(2):118–123, 2017.

Copyright notice