Trends in Mechanical Engineering and Technology
Volume- 16, Issue- 2, Year- 2026
Review Article
Received Date: 13 January, 2026
Acceptance Date: 7 April, 2026
Published Date: 20 June, 2026

EXPLORING the VIABILITY of DIY MOBILE ROBOTS, a TECHNICAL and ECONOMIC PERSPECTIVE

[1*]Prof. Y.S. Dighe, [2]Abhijit Gawali, [3]Ajinkya Jadhav , [4]Gaurav Khaire, [5]Nisha Pavane

[1]Assistant Professor, Department of Mechatronics Engineering, Sanjivani College of Engineering, India

[2,3,4,5]Student, Department of Mechatronics Engineering, Sanjivani College of Engineering, India

[1]digheyogeshmk@sanjivani.org.in; [2]abhijitgawali549@gmail.com; [3]ajinkyajadhav822002@gmail.com; [4]g4ur4v03@gmail.com; [5]nishapavane65@gmail.com

Correspondence Author Email: digheyogeshmk@sanjivani.org.in

Abstract

This article is intended for on ramping of novices in mobile robotics and shine a new light on feasibility of home-brew robots. Readers are expected to use this resource as a take-off point from which they will get to know where to look and to what extent, along with some of our own recommendations. The topics covered include but not limited to, hardware selection, firmware programming, feedback control, system design, simulation, algorithm selection, software. We also recommend different paths based on readers' goals. In conclusion, as mobile robotics is a vast and complex field, if possible, we recommend beginners to purchase a ready-made robot ideally with ROS support and slowly understand each subsystem by interacting with it in isolation. Else, start with a simple PID controlled differential drive robot then, learn to work with a lidar in simulation to eventually start working in the real world.

This study analyzes the balance between cost, performance, and scalability in DIY robotic systems, offering both technical and economic insights. It emphasizes the impact of open-source tools and low-cost hardware in making robotics more accessible and fostering innovation. Furthermore, it provides actionable guidance and suggested learning paths to support beginners in progressing from basic understanding to practical, real-world applications.

Keywords— ROS (Robot Operating System), System Design, Differential Drive, PID (Proportional Integral Derivative) control, lidar (Light Detection and Ranging) sensor

Arduino and ESP have ignited a new age of home-made electronics and IoT-based automation. The community of DIY drones has also started to go places and so has that of DIY manipulators, though one remains elusive and that is mobile robotics.

The Cost of Equipment still remains unattainable for many households and the synthesis of knowledge isn't quite there yet [1]. This is our attempt at bridging that gap.

I. TECHNICAL BASICS

Mobile robotics is an extremely wide and multidimensional field and is very overwhelming for novices. It consists of various concepts from software and hardware fields like control theory, path planning, communication architecture and power management. So, let's cover ground on some absolute basics that are a must to know to navigate this space of mobile robotics [2].

A. Grid Traversal Techniques

1) Depth First Traversal: Keep going down a branch till you can't go further then, look at other options.

2) Breadth First Traversal: Visit all neighbours first, put them in a visited queue and visit their neighbours which aren't in the visited queue till reaching the target, while keeping a log of cell transitions. Once the target is reached use the logged data to backtrack and find the shortest path. [Fig.1]

3) Other Notable Algorithms:

- *Dijkstra:* Breadth-First-Traversal but, with weights for different edges of graphs
- *A*:* Dijkstra but, utilized a heuristic function for choice.

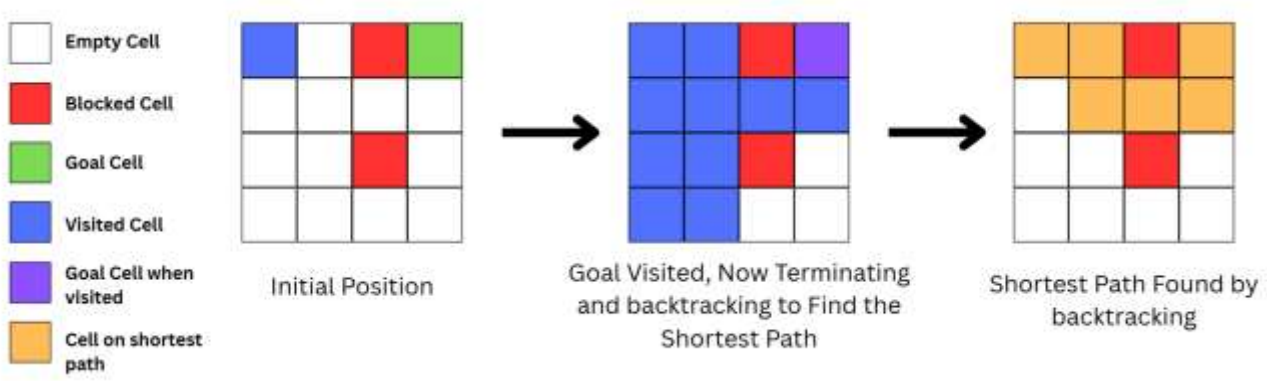

Fig. 1 Visualization of Shortest Path Finding using Breadth First Search

B. Communication Mechanisms

1) Topics : Topics are like FM radio, subscribers can choose their publishers. Topics are usually used by sensors and actuators, as they need to be non-blocking and have a simpler interface. Logging and visualization is also another use case of these [Fig 2].

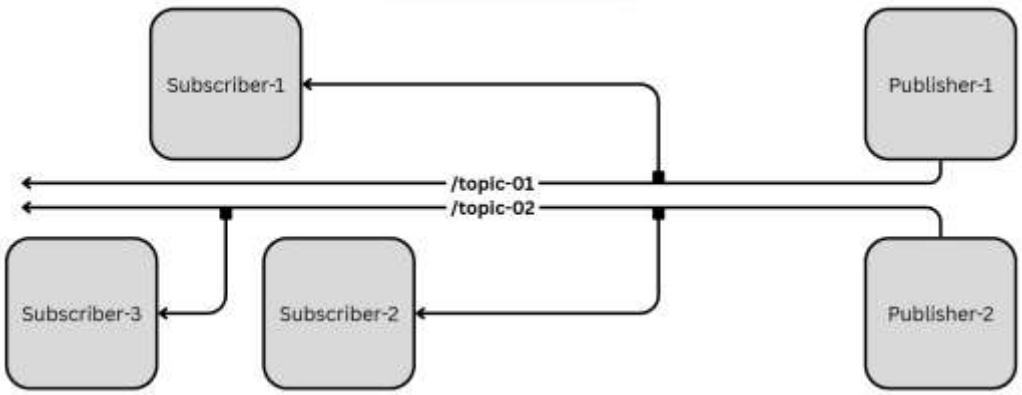

Fig. 2 Topics

2) Services: Services are just the Client-Server Model and they carry all the advantages and disadvantages of those here as well. Good for when, some processing is needed before fulfilling the request [Fig 3].

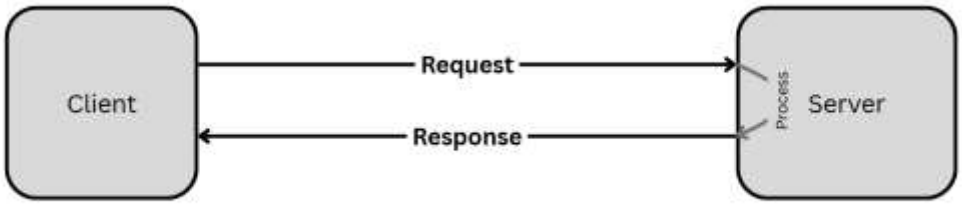

Fig. 3 Services

3) Actions: Actions are preferred for discrete behaviour and tasks that take longer to start. Actions are Heavily used in Nav2 Package [Fig 4].

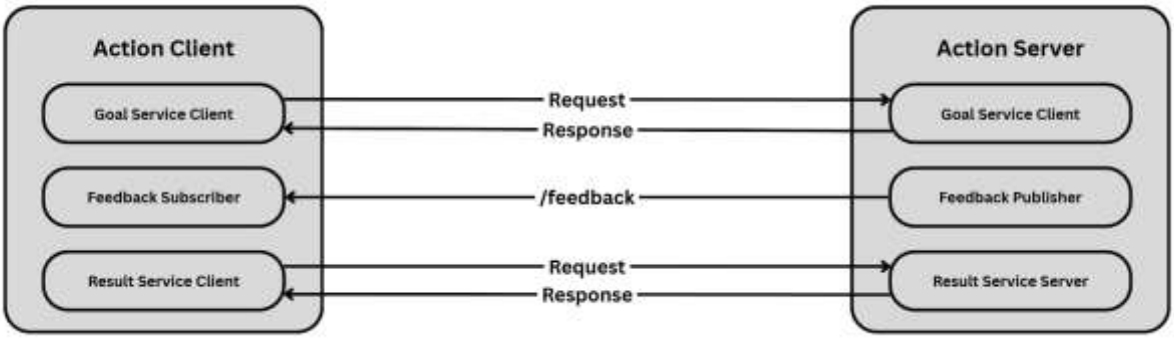

Fig. 4 Actions

C. Control System & Algorithm

System is a collection of elements/components connected together to achieve a certain output from a certain input. When such a system is used for controlling some parameters like position, speed, flow, etc it is called a control system [Fig 5].

1) Types of Control systems:

- *Open loop:* These systems don't have feedback, they are simple, economical and prone to mistakes in availability of environmental noise. These are usually not robust enough for navigation or robotics but do shine in simple tasks where interference can be predicted beforehand and compensated.
- *Closed Loop:* These systems have feedback, they are relatively sophisticated than their open loop counterparts. These are heavily used in robotics because of their robustness towards noise as noise is abundant in real-world scenarios.

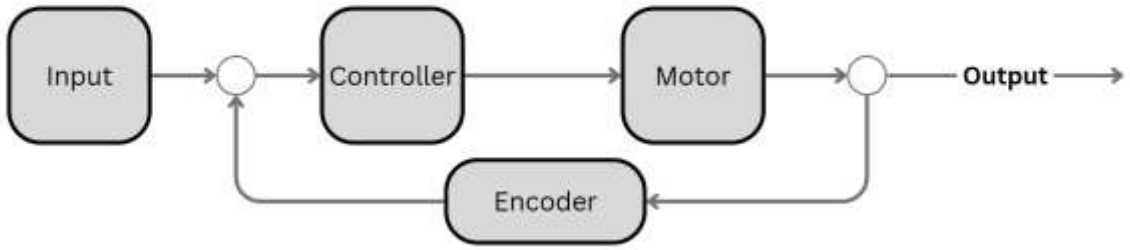

Fig. 5 Block Diagram of Closed Loop System

2) Feedback Control Strategies:

- *Proportional-Integral-Derivative (PID):* These systems provide appropriate error corrections according to a simple expression that is dependent on only the current error, previous error and sum of errors. Don't confuse this simplicity for lack of application though. It is the most commonly used method in robotics and sufficient for almost all cases. [Fig.6]

  Kp - Response Time & may introduce overshoot

  Ki - Opposes Steady State error & may introduce integral windup

  Kd - Opposes overshooting (Acts like friction to control variable as it approaches target)

$$u(t) = K_p.e(t) + \int_0^t e(\tau).d\tau + K_d \frac{d}{dt} e(t)$$

Fig. 6 Expression to calculate PID Control Variable

- *Linear Quadratic Regulators (LQR):* Here, system dynamics are in the form of linear differential equations and cost function is quadratic.
- *Model Predictive Control (MPC):* An optimal control strategy that uses current estimates to produce an output that reduces the overall cost based model of the plant.

D. Sensor Fusion

The idea behind sensor fusion is that of combining two imperfect measurements to reach an estimate that is better than that of any single measurement. Many types of algorithms/filters are used for this process, some important ones for robotics are as follows [4].

1) Complementary Filter: Just a simple combination of both outputs in different percentages. E.g., in drone Accelerometer and Gyroscope is used for better roll and pitch. In mobile robotics, Gyroscope and Magnetometer and sometimes even encoders are combined for better estimation of yaw. Or Accelerometer and encoders are combined for estimating odometry. [Fig.7]

$$\theta = \theta_{acc}.\alpha + \theta_{gyr}.(1-\alpha)\alpha$$

Fig. 7 Expression for Tilt using Complementary Filter used in Aerial Robots

2) Kalman Filters: These are more sophisticated than the simple complementary filter but the idea is the same, i.e., combine multiple flawed measurements for better estimates. There are also different types of Kalman filters like Extended Kalman Filter (EKF), Unscented Kalman Filter(UKF).

E. Transformations

Coordinate transformations are a mapping between two links in multidimensional space. Transformations are expressed in the relative position of the child node with respect to the parent node, usually in 3-Translations and 3-Rotations. There are mainly two types of transformations.

1) Static/Fixed: The transformation is fixed forever i.e., no relative movement is allowed between links.

2)Dynamic: These allow for relative movement in usually one dimension but multi-degree-of-freedom transforms are also allowed. [Fig.8]

* The following figure shows a transformation of 10 units in X-Axis, 2 units in Z-Axis and a rotation of -90 degrees in Y-Axis, in that order.

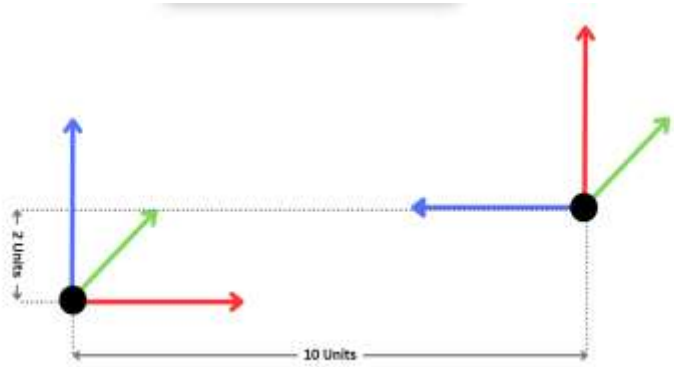

Fig. 8 An Example of Transformation

III. SYSTEM DESIGN & HARDWARE

A. System-Design

Low-level control is off-loaded to the controller instead of the processor which is the proper way to do it [Table 1]. This way the processor can completely focus on computation and communications alone also, isolates the expensive components like processor and lidar from the high voltage components like motors and drivers [Fig 9]. Processor communicates with the controller via serial communication (SPI is an alternative) [5].

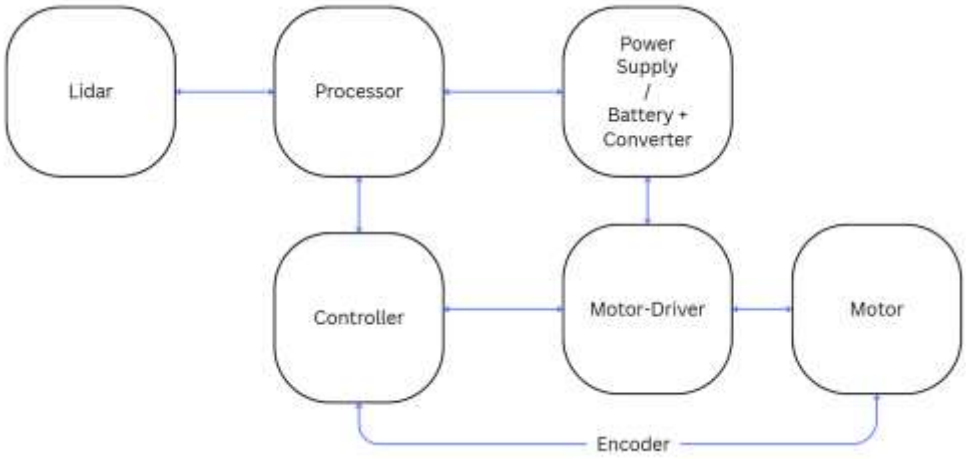

Fig. 9 System Design

B. Hardware

TABLE I

HARDWARE SELECTION

Component	**Recommendation**	**Reasoning**	**Alternatives or Considerations**
1. Microcontroller	Arduino Uno R3 (~ 300 Rs for clone)	Community support, learning resources & convenience	STM32F103C8T6 (BluePill) (~ 200 Rs) ESP32 (~ 500 Rs)
2. Microprocessor	Raspberry Pi 5 - 4 GB RAM (~6K Rs)	Community support, learning resources & up-to date	BeagleBoneBlack Rev C3 (~5.5K Rs) Raspberry Pi 4 - 4 GB RAM (~5.5K Rs)
3. Motors (Pair)	12V-N20-52 RPM (~1K Rs)	Torque & Encoders	Any as per chassis and weight
4. Motor Driver	TB6612FNG (~ 200 Rs)	Less Power Consumption	L293D (~ 200 Rs) L298N (~ 200 Rs)
5. Lidar	RPLidar-A1 (~ 8.5K Rs)	Price, Examples & Learning Resources	Any as per your budget
6. Battery	Battery: 3S Lithium Polymer (~1-2K Rs)	Capacity and Discharge Rate	Lithium-Ion Battery packs (~100 Rs per Cell)
7. Buck Converter	XL4015 5A (~ 100 Rs)	Cheap	Any, but make sure supports adequate current (~5A)
8. Wheels	a. Castor Wheel (~50 Rs) b. Driven Wheels [Pair] (~ 300 Rs) c. Couplings for Driven Wheels [Pair] (~ 200 Rs)	Necessity	Pick with appropriate sizes and compatibility
9. Chassis	Acrylic (~200 Rs) Plywood (~50-200 Rs)	Easy to work with consumer tools and yet firm enough	Hard-Foam

* Few tools that aren’t part of the robot but, recommended

- Drill
- Power Supply (can be extracted from an old Computer for free)

* *NOTE:* Novices should stick with our recommended controller and processor, since the project itself has plenty of complexity it isn't advised to compound that by choosing core hardware that doesn't have a ton of community support or learning material, alternatives are mentioned for awareness.

IV. SOFTWARE

A. Communication Framework

1) Robot Operating System: is an ecosystem of compatible tools that streamline robotics development. It provides powerful visualization support in the form of Rviz and is also supported by most off the shelf robots and simulators. Python and C++ are supported languages [6]. Though it now supports windows as well, we highly recommend beginners to stick with linux due to currently available learning material.

* Some Important Concepts:
  - *Universal Robot Description Format (URDF):* Used to define a robot using an xml file
  - *XML Macros (XACRO):* Utilized to reduce repetition in URDF files
  - *Communication Mechanisms:* Topics, Services and Actions (As introduced in Section II.B)

* Some Important Packages of Note:
  - *robot-state-publisher:* for static transforms
  - *joint-state-publisher:* for dynamic transforms
  - *teleop-twist-keyboard:* for teleoperation

2) Alternatives for Communication: These can used but, novices are still advised to stick with ROS
  - *ZeroMQ*
  - *Mobile Robot Programming Toolkit (MRPT)*

B. Simulator

1) Gazebo: Physics simulator that is open-source and most widely used by the ROS-community. Its file format is SDF(Simulation Description Format) URDF can be easily translated to this. There are two versions, classic is end-of-life but most learning resources still use it and should be given some consideration but, for future compatibility the current version is recommended [7].

2) Alternatives:
  - Webots: Cross-platform, even support web
  - Nvidia-Isaac: Better compatibility with Nvidia Components
  - AWS-RoboMaker: For Cloud features
  - Mujoco: Cross-platform & simple

V. ROS2-CONTROL

ROS2-Control is a realtime kernel for interaction of ROS2 with real-world hardware. It abstracts away the need to specifically write drivers for each and every component specifically.

* There are three types of Hardware:
  - *System:* reads and writes and collections of joints

- *Sensor:* Only reads, and single joint
- *Actuator:* Only writes and single joint

* Hardware Component Lifecycle:

- *Unconfigured (on_init, on_cleanup):* Hardware is Initialized, Communication isn't
- *Inactive (on_configure, on_deactivate):* Communication is Initialized
- *Active (on_activate):* Hardware has Electricity flowing through it
- *Finalized (on_shutdown):* Interface is ready for unloading

* Mock Components: Trivial simulations of hardware components. Very useful for a quick sanity check and improves iteration time greatly [8].

* Firmware: Some are available on the web, we recommend *ros_arduino_bridge* by *Josh Newans* though it's highly recommended to try writing your own. To write your own, just write an UART interface with a PID controller for a differentially driven system.

VI. NAVIGATION-2

Navigation-2 is a ROS based navigation stack that makes it very easy to get started with mobile robot navigation. Actions mentioned in Section.II.B are Heavily used in Nav-2 Package [9]

A. Transforms of Note for First-Time Robot Setup [Fig 10]

- *base_link:* Rotational Center of the body
- *odom:* Fixed frame relative to starting position of robot
- *map:* World Fixed frame for globally-consistent representations of distance

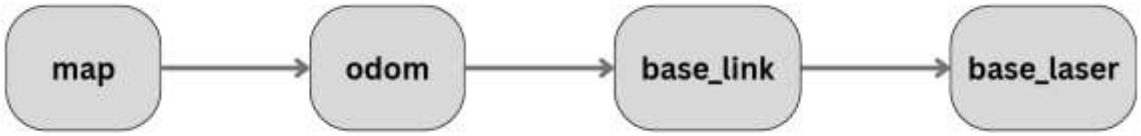

Fig. 10 Transform Tree for Nav-2

B. Planner Server

Planner Server is responsible to compute the robot's path. Following are some important plugins and algorithms supported by Nav-2.

- *NavFn planner:* Djikstra or A*
- *Smac 2D planner:* 2D A* using 4 or 8 neighborhoods with smoother & multi resolution query
- *Theta Star Planner:* Uses line of sight to create non-discreetly oriented path segments

C. Slam Toolbox

Slam Toolbox is a package that provides a set of tools and capabilities for 2D SLAM *(Simultaneous Localization and Mapping)* that is highly compatible and commonly utilized with Nav-2

CONCLUSIONS

DIY mobile robotics has come a long way in terms of user friendliness but it's still not near enough to the likes of DIY IoT, Drones or even CNC. The platform developers & maintainers are also currently more focused on implementing latest cutting-edge features than user comfort as they themselves are working in research organizations and corporations with those demands.

But, the future looks promising as the hardware could be purchased at about 20K - 25K Rs (about the cost of a budget 3d-printer) and software is getting more and more user friendly [10] . As for learning paths, learners with software and algorithm concerns should start working with simulations first. learners with more of a hardware concern should get a ready-made bot if possible and learn about sub-systems in isolation else, they should start with a simple differential drive PID controlled robot on a microcontroller and slowly graduate to a microprocessor by adding a lidar and using ROS2, ROS2-Control and Nav-2.

REFERENCES

1. Build a Mobile Robot with ROS. [ONLINE]. Available: https://articulatedrobotics.xyz/category/ build-a-mobile-robot-with-ros
2. Self Driving and ROS 2 - Learn by Doing! Plan and Navigation. [ONLINE]. Available: https://www.udemy.com /course/self-driving-and-ros-2-learn-by-doing-plan-navigation
3. ROS 2 Nav2 [Navigation 2 Stack] - with SLAM and Navigation. [ONLINE]. Available: https://www.udemy.com /course/ros2-nav2-stack/
4. ROS-2 Documentation. [ONLINE]. Available: https://docs.ros.org/en/jazzy/index.html
5. Nav-2 Documentation. [ONLINE]. Available: https://docs.nav2.org/
6. ROS-2 Control Documentation. [ONLINE]. Available: https://control.ros.org/jazzy/index.html
7. Choi JY, Ahn S, Kim D, Heo J, Yun WJ, Hong S, Bae S, Ahn SH. Exploring challenges and opportunities in manufacturing and intelligence for future robotics. International Journal of Precision Engineering and Manufacturing. 2025 Sep;26(9):2203-22.
8. Cavallo F, Limosani R, Manzi A, Bonaccorsi M, Esposito R, Di Rocco M, Pecora F, Teti G, Saffiotti A, Dario P. Development of a socially believable multi-robot solution from town to home. Cognitive Computation. 2014 Dec;6(4):954-67.
9. Woern AL, Pearce JM. Distributed manufacturing of flexible products: Technical feasibility and economic viability. Technologies. 2017 Oct 30;5(4):71.
10. Pedersen SM, Fountas S, Have H, Blackmore BS. Agricultural robots—system analysis and economic feasibility. Precision agriculture. 2006 Sep;7(4):295-308.