Designing a versatile robot framework for undergraduate robotics education

Jana Pavlasek, Polytechnique Montreal

Jana Pavlasek is an Assistant Professor at Polytechnique Montréal. She earned her PhD in Robotics at the University of Michigan. Her research interests include robotic perception and planning under uncertainty and robot learning.

Mr. Broderick Charles Riopelle, University of Michigan

Brody Riopelle is a robotics engineering technician at the University of Michigan, working on developing robots and software systems for undergraduate education. He graduated with a degree in Computer Engineering from the University of Michigan.

Mr. Abhishek Narula, University of Michigan

Abhishek Narula is an engineer, artist and educator at the Department of Robotics and the University of Michigan.

Shaw Sun, University of Michigan

Shaw Sun is a Research and Development Engineer in the Robotics Department at the University of Michigan. She earned a Master's degree in Robotics from the University of Michigan and a Bachelor's degree in Computer Science from Rutgers University. She focuses on developing MBot systems and enjoys exploring innovative techniques in robotics.

Peter Gaskell, University of Michigan Robotics Department

Peter Gaskell is a faculty member in the University of Michigan's Robotics Department, where he has spent the past decade designing and delivering hands-on curricula in robotics, embedded systems, controls, and SLAM. He led creation of the open-source MBot educational robotics platform and numerous other platforms to make advanced robotics accessible and scalable. Peter's scholarship focuses on low-cost sensor fusion and lightweight SLAM for resource-constrained platforms. Recognized with the College of Engineering's Teaching Innovation Award, he continues to build inquiry-driven learning environments that lower barriers to advanced robotics in graduate and undergraduate education.

Designing a Versatile Robot Framework for Undergraduate Robotics Education

Abstract

The growing popularity of robotics education in undergraduate engineering programs gives rise to a demand for robotic technologies to facilitate learning in the classroom. Robotics undergraduate curricula require platforms and tools that grow with the students, remaining accessible to early undergraduate levels while supporting the implementation of advanced algorithms. In this paper, we describe a software framework for educational mobile robotic platforms designed with undergraduate robotics education in mind. The MBot custom robot platform is capable of executing a vast range of robotic and machine learning algorithms using a variety of sensors. Our suite of open-source tools is designed to be accessible to students across the spectrum of undergraduate education and lends versatility to robotic platforms in the classroom. The robot can be programmed through multiple modalities designed for different levels of programming proficiency. A synchronous Application Programming Interface (API) enables the development of single-threaded applications, appropriate for CS1 level programming. The framework is supported by web-based tools which enable students to interact with the platform without extensive technical prerequisites. We describe the design criteria which enable the platform to adapt to the needs of the students throughout their journey through the engineering curriculum. We present two case studies that demonstrate how the robots are used in project-based undergraduate robotics courses at the University of Michigan: a first-year programming course and a graduate robotics laboratory. We also describe a block-based visual programming interface based on the same framework and its use in a grade school context. Finally, we present lessons learned in teaching undergraduate courses with real robots at different levels, and highlight future opportunities for development in this area.

1 Introduction

Robotics is growing rapidly in undergraduate education, with more institutions incorporating robotics in existing curricula, adding robotics concentrations, or introducing robotics majors [1, 2]. This rise in popularity creates a renewed demand for technology in the classroom to give students hands-on experience with robots. Integrating real robot platforms into the undergraduate classroom remains a significant challenge due to the technical complexity of programming robotic systems, making them inaccessible in many undergraduate contexts. The current moment lends the opportunity to reconsider the design of frameworks and tools for robotic platforms in the context of undergraduate engineering education.

Robots have a decades-long history of use in the undergraduate classroom for the instruction of classes ranging from introductory computer programming (CS1) [3, 4] to advanced special topics courses in robotics [5, 6, 7]. Many user-friendly platforms are designed with education in mind, offering interfaces which are accessible with minimal prerequisites. While these platforms offer a low burden for development, they often utilize microcontrollers for computation, limiting the ability to run many robotics and AI algorithms of relevance beyond a CS1 course (e.g. localization, mapping, computer vision). Platforms with more advanced on-board computing are capable of running these algorithms, making them suitable for modern robotics and AI algorithms. However, existing frameworks to program these platforms require advanced computing skills, limiting their use in early-year undergraduate courses or non-computing majors.

In this paper, we describe the design of a framework for a custom robot platform, the MBot [8], which enables robotic platforms to be used across a spectrum of undergraduate engineering classrooms. The design of such a robotic platform for education is a critical component of the student experience within the course. Motivated by the recent call for deeper discussion of the design elements which form a key component of teaching computing education [9], this section describes the design considerations for a robot platform suitable for undergraduate robotics education.

We identify the design criteria which govern the design of a flexible, user-friendly software framework for programming and operating a robot in the context of undergraduate education. These criteria adhere to the *low floor, high ceiling* philosophy, which strives to make an accessible system (low floor) while still being sufficiently flexible for multiple use-cases for teaching robotics in a range of undergraduate courses (high ceiling). We present the MBot Bridge, a tool which enables robots to be programmed by non-experts through a beginner-friendly Application Programming Interface (API). We also describe a suite of web-based and command line tools which provide user-friendly control and introspection of complex robot systems. The development of a block-based visual programming interface, the MBot Scratch Extension, demonstrates the flexibility of the framework. To illustrate the functionality of the system, we provide case studies of the MBot platform being used in an introductory programming course, a graduate robotics laboratory course, and a fifth grade summer camp.

1.1 Related Work

Robots have a rich history in education, pioneered with the introduction of the programmable "Turtle" as part of the LOGO programming language [10]. Since then, robotics has been used extensively to teach principles of computing and engineering to students of all ages [11, 12]. In the context of undergraduate education, introductory computer science (CS1) courses have benefitted from user-friendly platforms like the LEGO ® Mindstorms robot [4, 13], which takes its name from Papert's original work [14], and the Parallax Scribbler [3, 15]. These highly impactful platforms sparked a vast array of robotic platforms and robotics competitions as an educational tool [16, 17], alongside programming environments designed to be more amenable to undergraduate teaching than common robot middleware frameworks [18, 19].

These education-focused platforms usually provide a simple way to program the robots, e.g. a custom library in the Arduino IDE, making the technical prerequisites required suitable for a CS1

audience. The platforms typically use a single microcontroller for computation, capable of running single-threaded programs and compatible with low-dimensional sensors (e.g. encoders, range finders, line detectors). This limits the sophistication of behaviors that can be programmed by the students, making these platforms unsuitable to modern robotics and AI algorithms.

Courses that focus on robotics and AI algorithms have opted to use more advanced robots, like the Sony AIBO [20], modified iRobot Roomba platforms [21, 22, 23, 24], and the Turtlebot [25]. Other courses have designed custom platforms, e.g. Duckiebot [7], MIT RaceCar, and MuSHR [26], often with open-source materials. These platforms typically include on-board computing (e.g. a Raspberry Pi or NVIDIA Jetson) and can support more complex sensors like a Lidar range finder or a camera. This hardware enables is capable of executing more sophisticated autonomous robot software including mapping, localization, and computer vision, making them well-suited to advanced undergraduate or graduate courses. These systems typically rely on frameworks such as the Robot Operating System (ROS) [27] or Lightweight Communications and Marshalling (LCM) [28] to manage to manage and communicate between processes.

Using these platforms therefore requires advanced programming skills, such as executing multiple, multi-threaded programs, and familiarity with concepts like the Linux command line, Docker, or ROS. These prerequisites limit the applicability of these platforms to undergraduate courses without advanced computing prerequisites and introduce a significant engineering burden for instructors.

The framework described in this paper is intended for use on a high-capability, Raspberry Pi-based platform, the MBot [8], a custom platform developed at the University of Michigan. The platform was initially intended for the instruction of a graduate-level course (described in Section 5.1). In this work, we describe the design of a software framework, the MBot Bridge, and associated tools which enable the platform to be accessible to early level or non-computing courses without loss of sophistication.

2 Undergraduate Robotics Design Criteria

Central to the challenge of integrating real robotic platforms into undergraduate engineering curricula is the design of software and infrastructure to support student learning. The thoughtful design of appropriate interfaces is critical to enable students to interface with these robots without allowing the complexity of the robot systems to detract from the student experience. We define the following design criteria for a robot framework suitable for undergraduate education:

DC 1 Capable of running modern robotics algorithms (e.g. localization, mapping, vision),

DC 2 Relies on technical prerequisites appropriate for a CS1 student,

DC 3 Includes visualization and introspection tools appropriate for a CS1 student.

To support modern robotics education, a robot must be capable of executing sophisticated autonomy algorithms (**DC 1**). Existing robot platforms capable of running these algorithms commonly provide a bare-bones interface which rely on advanced computing skills, which are out of reach for many undergraduate use cases. **DC 2** defines the base prerequisite level to be equivalent of an introductory programming course (CS1). A CS1 programming skill level is

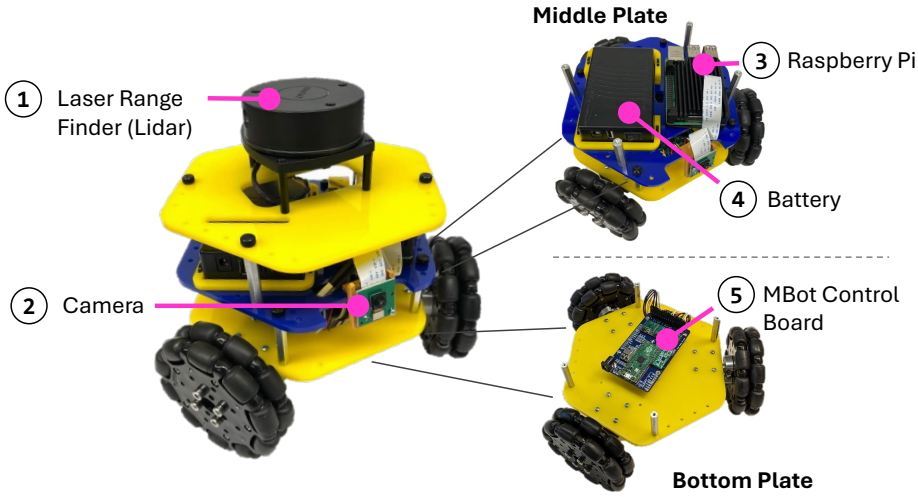


Figure 1: The MBot educational robot platform (omni-directional configuration) used in this work. The robot includes sensors (1, 2) and uses a Raspberry Pi (3) for computation. Low-level control is managed by the MBot Control Board (5) which communicates with the Raspberri Pi (3) via a serial connection. The low-cost platform is fully mobile, WiFi-enabled, and battery (4) powered.

defined as the ability to write, compile, and execute single-threaded programs. This ensures that the resulting framework matches the prerequisites for introductory classes or non-computing majors.

The challenges of handling uncertain conditions are central to the field of robotics and an important learning tool for students interested in studying robotics. However, dealing with hardware in real-world conditions inevitably introduces failure modes and diagnosing issues on a robot can be frustrating to students. **DC 3** specifies that a robot software framework must include tools that can be used for visualization and troubleshooting that are accessible from a CS1 level.

In this paper, we describe the design of a software framework and tools for the MBot platform that meet the defined criteria.

3 Preliminaries: Robot Platform and Software

We extend the MBot, a low-cost, flexible platform for robotics education at the undergraduate and graduate levels designed at the University of Michigan [8]. The platform was originally designed to teach a graduate-level robotics lab covering mapping, localization, and planning. The MBot hardware and software are open-source and the platform has been used to teach multiple classes at the University of Michigan and at three collaborating universities.

Robot Hardware. The framework described in this paper assumes the use of a mobile robot platform equipped with a single-board computer and sensors, including low-level sensors (e.g. encoders, IMU) and high-level sensors (e.g. camera, Lidar). We use the MBot, a custom platform that can be built with a differential drive or omni-directional drive configuration (see MBot Omni configuration in Figure 1). The motors are controlled via a custom circuit board featuring a

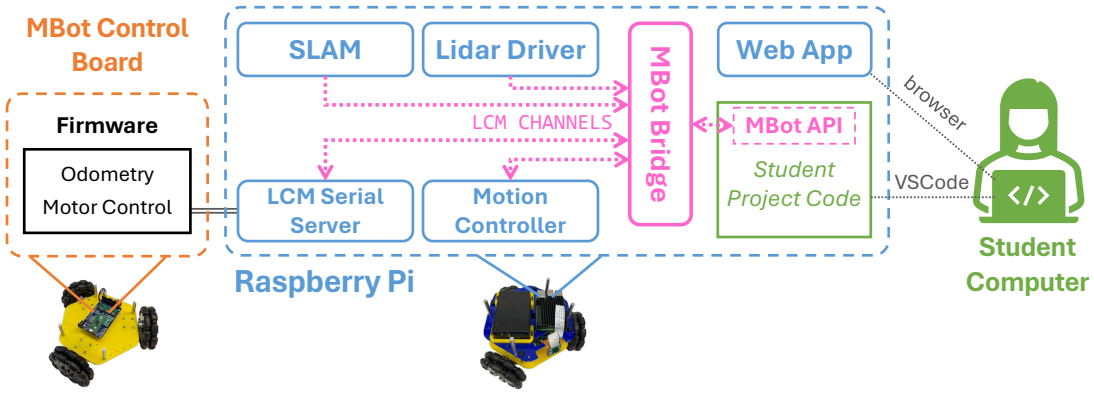


Figure 2: Example software architecture for the MBot framework. The MBot Control Board (left) runs firmware which accomplishes low-level tasks including motor control and odometry. The board communicates with the robot software stack on the Raspberry Pi (center) through a serial connection. The MBot Bridge translates information from the processes (e.g. sensor drivers, localization, planning) to a custom API which students can use to program the robot. Students access the robot through a remote interface and through the web tools in a browser (right).

microcontroller, the MBot Control Board, which drives motors in response to commands over a serial connection. A Raspberry Pi single-board computer interfaces with a Lidar and camera, and executes high-level robot software. For a full description of the robot hardware design, see [8].

Robot Software. The MBot software architecture is comprised of multiple parallel processes which run on the Raspberry Pi and control the low-level robot functionality. These processes include system utilities, sensors drivers (e.g. Lidar), and a serial server for communication with the control board. The processes communicate via the Lightweight Communications and Marshalling (LCM) message passing framework [28]. The control board is equipped with a microcontroller running custom firmware that reads low-level sensor information (e.g. encoders, odometry, IMU) and executes control commands passed from the serial server process via a serial connection. This software architecture is illustrated in Figure 2.

4 Designing a Framework for Undergraduate Robotics Education

Upper-level or graduate robotics courses typically require students to implement code by interfacing directly with the existing operating system and processes on the robot, as described in Section 3. Programming these robots typically involves the effective use of message-passing frameworks, which requires competence with multi-threaded and object-oriented programming. Building and executing the programs required in these systems requires competency with the Linux operating system and heavy use of multiple command line sessions. It follows that such a framework is suitable only to courses which assume strong computing prerequisites.

This section describes a framework which enables the implementation of algorithms on a real robot that is accessible to a broad range of undergraduate engineering students. We introduce the *MBot Bridge*, a program which enables *single-threaded* programs that can interface with

sophisticated robot software stacks and high-dimensional sensors, satisfying **DC 2** and **DC 1** from Section 2. We also develop web-based and command line tools for visualization and debugging which students can use in the classroom for control and troubleshooting the system, satisfying **DC 3**. We additionally develop a block-based visual programming extension that uses Scratch and the MBot Bridge to provide a beginner-friendly control interface. The software and tools described in this section are open-source, designed to be compatible with the open-source hardware for the MBot platform.

4.1 MBot Bridge

We introduce a new tool, the MBot Bridge, which can be used to program end-user applications on the MBot without LCM. The MBot Bridge can be used to write synchronous, single-threaded programs suitable for students at a CS1 level, while also being sufficiently flexible for more sophisticated use-cases. The design of the MBot Bridge is inspired by Rosbridge, a tool which enables non-ROS users to program ROS-based robot platforms [29].

The MBot Bridge is composed of two main parts: the *MBot Bridge Server*, the backbone of the framework which interfaces between the robot software stack and user applications, and the *MBot Bridge API*, which provides a user-friendly abstraction layer to read and write data, available in Python, C++, and JavaScript. The MBot Bridge employs websocket connections and a custom protocol to pass data between client programs and the server, and provides a *bridge* to the LCM message passing system on the robot. The websocket-based design of the system enables easy integration into web-based applications, which are highly user-friendly as they can be accessed with no prerequisites from any browser. The software architecture is shown in Figure 2.

The MBot Bridge allows data from LCM processes to be queried synchronously, enabling CS1-friendly single-threaded programming. In contrast, message passing frameworks rely on subscription callbacks in multiple threads, requiring the user to manage thread safety. The user-friendly benefit of the MBot Bridge comes at the cost of added latency, so time-critical algorithms should interface directly with the message passing framework when possible.

The MBot Bridge Protocol. The MBot Bridge defines a custom protocol in JSON to communicate over websockets using JSON messages. There are six types messages available: `REQUEST`, `RESPONSE`, `PUBLISH`, `SUBSCRIBE`, `UNSUBSCRIBE`, and `ERROR`. The types are listed and described in Table 1.

The most common applications are to send data to a channel with a `PUBLISH` message (e.g. to send control commands to drive the robot) or to read data from a specific channel with a `REQUEST` method (e.g. to read sensor data or state information). When a `REQUEST` is sent over a websocket connection, the server sends back a `RESPONSE` message on the same websocket connection with the latest data on the given LCM channel in the `data` field. The data is sent as a JSON object, or as raw bytes when the `as_bytes` boolean is set.

The `SUBSCRIBE` message type results in the server sending all data on a given channel back along the same websocket connection, until the connection is closed or an `UNSUBSCRIBE` message is sent. This functionality exists for completeness and is primarily useful for JavaScript applications that cannot directly subscribe to LCM channels.

Table 1: Message Type Definitions in the MBot Bridge Protocol

Type	Description	Keys
`REQUEST`	A request to read data.	`channel`, `dtype`, `as_bytes`
`RESPONSE`	A response from the server.	`channel`, `dtype`, `data`
`PUBLISH`	A request to publish data.	`channel`, `dtype`, `data`
`SUBSCRIBE`	A request to send all data on a certain LCM channel back from the websocket.	`channel`, `dtype`, `as_bytes`
`UNSUBSCRIBE`	A request to unsubscribe from a channel on the given websocket connection.	`channel`
`ERROR`	An error response from the server.	`data`

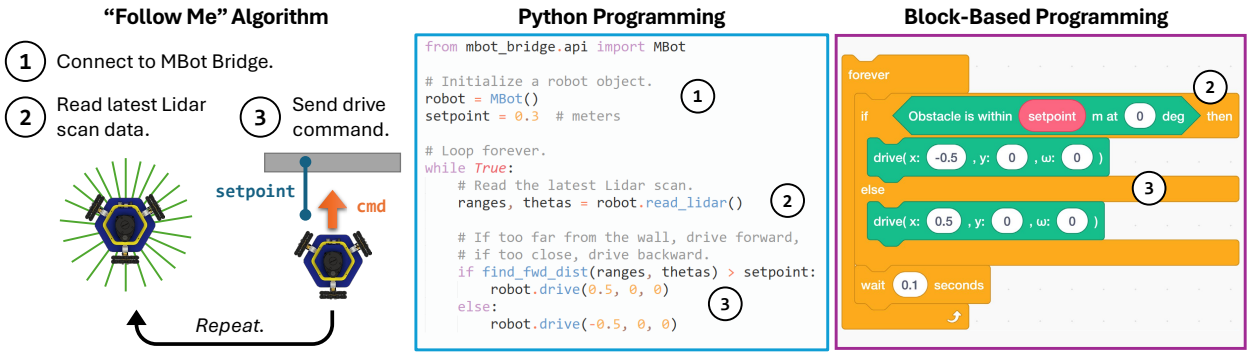


Figure 3: An illustration of a program to control the MBot using the MBot Bridge API (middle) and the Scratch extension. Both programs implement a "Follow Me" algorithm, which maintains a setpoint distance to the obstacle in front of the robot. Once connected to the MBot Bridge server (1), the algorithm reads the Lidar scan (2), then sends a drive command to move forward if the distance to the obstacle is greater than the setpoint, and backward if the distance is less than the setpoint.

The `ERROR` message is used to report errors in reading or processing data, and includes an error message in the `data` field. An error can be returned if the client requests data on a channel that does not exist, if a data type cannot be decoded, or if messages are incorrectly formatted.

The MBot Bridge Server. The MBot Bridge Server is a process which monitors all the LCM channels that exist on the robot and exposes a websocket server to listens for queries. Each client query opens a unique websocket and sends a JSON message using the defined protocol. If a response is required or an error is raised, it is sent to the client over the same websocket, which is subsequently closed (except in the case of a `SUBSCRIBE` message). The server can convert between JSON objects and all available LCM message types.

The MBot Bridge API. The MBot Bridge provides a custom API which allows a user to send commands and read data using the custom protocol. The API is a user-friendly interface which allows the user program the robot without interacting directly with LCM or websockets. The API is the primary way students interact with the robot and functions are provided for the common use-cases (e.g. driving the robot, reading Lidar or pose). The API is available in Python, C++, and JavaScript. An example program using the Python API is provided in Figure 3.

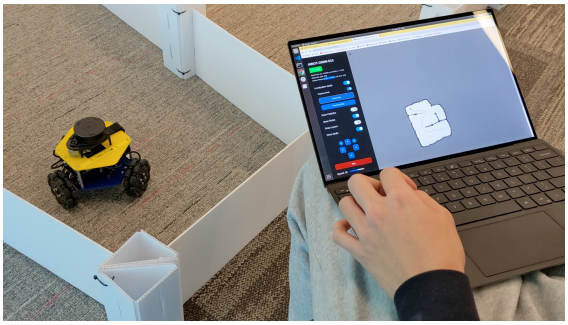


Figure 4: The MBot software framework includes a web-based application which can be used to drive the robot, control the mapping system, and visualize the map, robot position, and Lidar data.

4.2 MBot Visualization and Troubleshooting Tools

The MBot includes multiple peripheral devices and multiple independent processes, increasing the number of possible failure modes. Additionally, robotics as a field is still in its infancy, and algorithms and hardware that are robust to the full range of real-world environments have yet to be developed. To compensate for this challenge, we argue that tools for basic control and diagnosing are key components of an educational robotic platform. In this section, we describe the design of tools developed as part of the MBot software framework intended for undergraduate student use.

Web Tools. Web-based applications make ideal user-friendly interfaces to robot platforms. Once installed on the robot, these applications can be accessed through a web browser without installing additional software or running programs manually. Web-based tools are developed using the MBot Bridge JavaScript API. The MBot framework includes two main web applications: the primary MBot Web App and the MBot LCM Monitor.

The MBot Web App is the primary interface for visualization and basic control the the robot. It can be used to teleoperate the robot and to toggle mapping functionality. It also displays the robot location, Lidar data, and map (see Figure 4). The LCM Monitor web application displays all the current LCM channels on the robot, the latest message contents and the publication frequency. This tool provides introspection into the robot software stack and can be used to troubleshoot a number of common problems, e.g. malfunctioning sensors or missing processes.

Command-Line Tools. The MBot features a suite of command-line tools designed to streamline interaction with the robot and provide introspection into its systems. These tools provide functionality for system diagnostics, communication analysis, service management, and hardware monitoring through simple one-line commands. These intuitive commands allow students to focus on robotics concepts and algorithm development without being overwhelmed by technical complexities.

Each tool serves a distinct purpose. For example, MBot-info extracts detailed system- and robot-specific information from system files to evaluate the robot's operational state. MBot-lcm-msg and MBot-lcm-spy facilitate communication framework analysis, allowing students to monitor message traffic with a single command. MBot-service simplifies service management by leveraging Linux tools such as systemctl and journalctl, avoiding complex

Table 2: Overview of Robot Control Blocks in the MBot Scratch Extension

Block Name	Output	Description
Detect Obstacle	Boolean	Returns true if a Lidar scan detects an obstacle within a specified threshold and angle
Drive	None	Sends a drive command with a specified linear and angular velocity
Stop	None	Sends a drive command with 0 linear and angular velocity
Reset Position	None	Resets the odometry to its origin
Angle to Nearest Obstacle	Number	Returns the angle of the shortest ray in a Lidar scan
Distance to Nearest Obstacle	Number	Returns the length of the shortest ray in a Lidar scan
Get X Position	Number	Returns the relative displacement in the robot's x-direction
Get Y Position	Number	Returns the relative displacement in the robot's y-direction
Get Heading	Number	Returns the relative angular displacement about the robot's z-axis

command-line inputs while still offering students insights into background processes. MBot-status provides centralized real-time monitoring of hardware components, offering a roadmap for identifying and isolating potential failures. Together, these tools create an integrated ecosystem that supports development, testing, and experimentation for students.

4.3 MBot Scratch Extension

The MBot Scratch Extension is built on top of the MBot Bridge JavaScript API and provides a block-based visual programming interface suitable for early programming education. It requires a fork of the Scratch GUI and the Scratch Virtual Machine. The MBot Scratch GUI is the front-end application that enables the user to develop Scratch block programs. The MBot Scratch Virtual Machine is where the programming blocks are defined and implemented.

By default, Scratch provides access to blocks for event-driven (asynchronous) programming, math and logical operators, iteration, and control flow, among others. The MBot Scratch Extension provides a range of blocks categorized into command, boolean, and reporter functions that facilitate comprehensive control and feedback from the robot.

Using these blocks specific to MBot in conjunction with the Scratch default set of blocks, users can create complex programs that manage robot behaviors and interactions within its environment. This combination enhances not only the educational experience by making robotics more accessible but also the capabilities of developers without a thorough programming experience exploring robotics. An illustration of a program using the MBot Scratch Extension is shown in Figure 3.

5 Curriculum Case Studies

To demonstrate the effectiveness of the educational tools described in the previous section, we describe case studies consisting of courses developed at the University of Michigan which have

Checkpoints	Contents
0. Robot Setup	Linux Basics, Microcontrollers, Embedded Systems
1. Motion Controller	Wheel Speed Calibration, Odometry, PID for wheel speed and body velocity, Motion Controller design
2. SLAM Development	Occupancy Grid Construction, Monte Carlo Localization, SLAM System Integration, LCM Channels
3. Path Planning and Exploration	Map Construction, A* Path Planning, Autonomous Exploration

Table 3: Curriculum description for the graduate robotics systems laboratory at the University of Michigan.

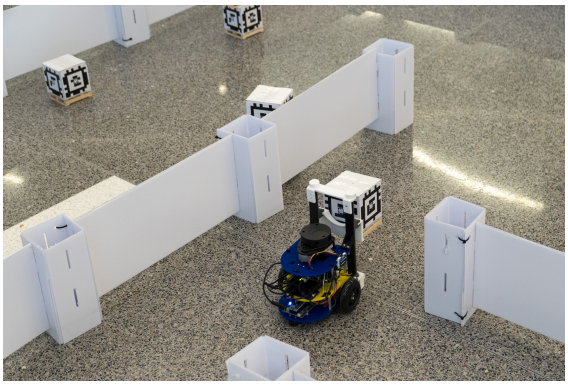


Figure 5: Example of a final competition involving a warehouse navigation task using the MBot in the graduate robotics lab at the University of Michigan.

employed our framework and platform in real classroom settings. It is worth noting that the MBot can be employed in other courses spanning topics in engineering, robotics, and AI, beyond those described here.

First, we describe a graduate robotic systems laboratory course, for which the MBot was initially developed, demonstrating how the robot is integrated in a course when advanced technical skills can be assumed. Next, we describe an introductory programming course taught using the MBot Bridge to program the robot requiring only CS1-level computing skills. Finally, we discuss a preliminary case study using the MBot Scratch extension for a fifth grade summer camp.

5.1 Graduate Robotics System Laboratory

The graduate robotic systems laboratory at the University of Michigan exposes students to mobile robotics concepts through hands-on tasks centered around the MBot platform. The curriculum progresses from hardware assembly and embedded programming to the development of advanced robotic systems, culminating in a fully integrated Simultaneous Localization and Mapping (SLAM) system. Checkpoints, designed as scaffolded challenges, allow students to incrementally build their knowledge and skills in hardware setup, motion control, and SLAM development, detailed in Table 3. The course concludes with a competition where students showcase their ability to navigate and map environments autonomously (see Figure 5).

The course projects require students to program using LCM over custom channels to efficiently

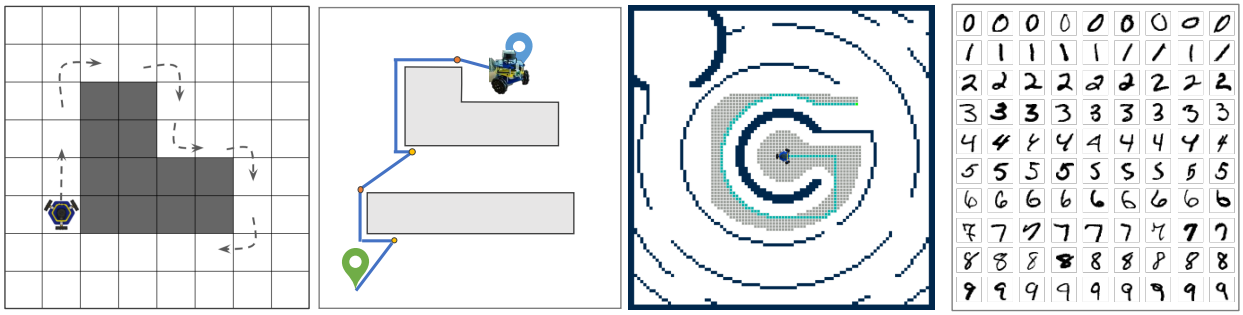
send and receive data between multiple processes. Students leverage the MBot web-based tools and command line tools to validate and troubleshoot their robots. By the end of the course, students have hands-on experience in essential robotics concepts such as localization, perception, and motion planning, as well as robot software and hardware systems, preparing them for real-world applications and challenges in mobile robotics.

5.2 Introductory Programming for Robotics

An introductory programming course through the lens of robotics and AI for first year engineering students was developed and taught at the University of Michigan. The course has also been taught as a special topics course to computer science and engineering students at three collaborating US universities. The course is intended to introduce students to computational thinking and foundational concepts in robotics which they have the opportunity to explore in greater depth later in their studies. The course is composed as a collection of modules, each of which culminate in a project implemented on the MBot using the framework and tools described in this paper.

The course modules are designed for students to practice programming skills by completing projects on a real robot. The early projects cover reactive control through wall following and bug navigation tasks. The final projects provide a high-level introduction to path planning and image recognition, relying on the concepts of graph search and machine learning, and also include a real robot component. The course projects are visualized in Figure 6.

While the first projects rely on computing concepts typical of CS1 courses which utilize robots, e.g. feedback control and state machines, the projects increase in complexity throughout the semester. Projects 3 and 4 require students to utilize mapping, localization, and vision, necessitating more sophisticated robot capabilities. The MBot Web App allows students to control the robot and make maps through their web browsers, and the MBot Bridge API allows them to fetch robot state information and send commands. This design illustrates how students interface with modern robotic algorithms within single-threaded CS1-level programming assignments.



Project 1: **Wall Follower** | Project 2: **Bug Navigation** | Project 3: **Path Planning** | Project 4: **Image Classification**

Figure 6: An illustration of the four primary projects students complete as part of the introductory programming course using the MBot.

5.3 Middle School Robotics Summer Camp

A two-session robotics camp was designed in partnership with a local non-profit after school literacy program to inspire younger students to pursue education in STEM fields, using the MBot Scratch extension. In addition, this camp tested if the MBot interface could be adapted for students without any programming knowledge using block-based programming. The students in this camp were in the fifth grade and participated in two sessions. The first session introduced the students to the MBot, explaining the sensors and computers on board and allowing them to teleoperate the robot from a laptop. The second session gave the students a design challenge: use the block coding interface to design a program to autonomously navigate the MBot through a simple maze.

Although camp sessions were designed for middle school students, the block-based programming interface for MBot provides an intuitive and accessible approach to developing computational thinking and programming skills for undergraduates and professionals who lack extensive programming experience. The field of robotics is uniquely diverse and includes many individuals who may not have a strong programming background. For these students and professionals for whom programming is a supplementary skill, the block-based programming environment offers a gentle introduction to coding. This approach allows them to quickly create functional programs and build confidence before transitioning to more complex, text-based languages if necessary.

6 Discussion

Teaching robotics involves unique challenges in terms of the tools and infrastructure used, which can have a significant impact on student experience. In this section, we describe key takeaways from our experience developing and teaching with the MBot platform, and discuss future work.

6.1 Ingredients for an Undergraduate Educational Robotics Platform

Below, we summarize suggestions for features a robot for an undergraduate course should have, drawing from our experience teaching with the MBot ecosystem.

Low floor, high ceiling. The call for a platform which is both accessible (low floor) and extendable to advanced tasks (high ceiling) is repeated throughout robotics education literature [3, 6, 8]. In this work, a "high ceiling" robot is employed, capable of running localization, mapping, planning, and vision applications on top of low-level control. Our software framework based on the MBot Bridge strives to accomplish a "low ceiling." This enables the robot to be used at levels ranging from introductory programming to graduate studies.

Reliability and Diagnosability. User tools and interfaces which provide control, visualization, and diagnosis information are essential for robotics education at the undergraduate level. Platforms designed for K-12 place a significant emphasis on user-centered design, whereas higher-education platforms tend to be more similar to research platforms which rely primarily on Linux-based command line interfaces. One promising line of work to mitigate this is in the

development of web-based tools, which require no installation, are machine-agnostic, and center visualization [29]. Introspection tools are central to the MBot framework.

Community and support. A key factor that impacts student and faculty experience in a robotics course is the availability of quality instructional material and documentation for a platform. Additionally, community support is critical for troubleshooting issues that arise. The MBot hardware and software is open-source, with guides available for staff and students. Adding more, diverse voices to the development community as robotics education scales will help grow these resources.

6.2 Limitations & Future Work

In practice, it is extremely challenging to achieve both a *low floor* and *high ceiling* in a robotic platform. While the MBot framework described here has been effective in a CS1-level course and a middle school context, integrating the system in a new course and configuring a robot fleet remains a technical burden on faculty and course staff. Additionally, students and instructors require training on using and troubleshooting robots, adding to the preparation time for faculty. The development of more training materials and further web-based diagnosis tools would alleviate issues dealing with errors that arise. Furthermore, developing an open-source community around the platform is a key area of future improvement. A positive correlation between strong community and successful curriculum adoption has been demonstrated in previous successful cases of education innovation [30].

Another avenue of future work is to evaluate the effectiveness of the system design quantitatively by designing user studies in the classroom. We are interested in studying the effectiveness of the platform as a learning and a motivational tool for undergraduate engineering students, including student tolerance for handling errors in the system.

7 Conclusion

In this paper, we present a framework intended to enable the MBot to be integrated into the undergraduate engineering classroom. We describe the design of a comprehensive suite of open-source software tools which allow non-expert users to program sophisticated robot systems. Central to this framework is the MBot Bridge, which acts as an interface between a multi-process message passing-based robot software stack. The MBot Bridge enables the development of web-based introspection and visualization tools which are accessible to undergraduate students, and visual programming interfaces accessible to beginner programmers. This framework aims to advance educational learning tools for robots in the classroom with robotics education in mind, and has been employed within several real classroom settings to date. This work opens further avenues for investigation, including studies of the educational impact of robotics platforms in the classroom and development of more tools and curricula based on the MBot.

Acknowledgments

This work was supported in part by the Sloan Foundation, the Ford Motor Company, and Amazon, Inc. The development of the Robotics curriculum at the University of Michigan is due

to the efforts and leadership of Jessy Grizzle, Chad Jenkins, Elliott Rouse, and Dawn Tilbury. The authors thank Mark Guzdial for his contributions to undergraduate robotics education with the MBot. We acknowledge the contributions of Ed Olson, Justin Tesmer, Shane DeMeulenaere, and Collin Johnson, whose work developing early robotic platforms for the University of Michigan graduate robotics program led to the development of the MBot. We also thank the faculty who were early users of the MBot for their feedback and collaboration, notably Jasmine Jones, Wei Wu, Jan Pearce, Dwayne Joseph, Todd Shurn, Katie Skinner, and Xiaoxiao Du, and the instructional aides and graduate student instructors who supported teaching with the MBots. This paper employed AI-assisted writing tools for minor editing and grammar improvements.

References

[1] M. A. Gennert and C. B. Putnam, "Robotics as an undergraduate major: 10 years' experience," in *ASEE Annual Conference & Exposition*, 2018.

[2] O. C. Jenkins, J. Grizzle, E. Atkins, L. Stirling, E. Rouse, M. Guzdial, D. Provost, K. Mann, and J. Millunchick, "The Michigan Robotics undergraduate curriculum: Defining the discipline of robotics for equity and excellence," *arXiv preprint arXiv:2308.06905*, 2023.

[3] T. Balch, J. Summet, D. Blank, D. Kumar, M. Guzdial, K. O'hara, D. Walker, M. Sweat, G. Gupta, S. Tansley, *et al.*, "Designing personal robots for education: Hardware, software, and curriculum," *IEEE Pervasive Computing*, vol. 7, no. 2, pp. 5–9, 2008.

[4] I. M. Souza, W. L. Andrade, L. M. Sampaio, and A. L. S. O. Araujo, "A systematic review on the use of LEGO® robotics in education," in *IEEE Frontiers in Education Conference (FIE)*, pp. 1–9, IEEE, 2018.

[5] E. Welton, S. Hutchinson, and M. Spong, "A modular, interdisciplinary approach to undergraduate robotics education," in *IEEE Frontiers in Education Conference (FIE)*, pp. 714–719, IEEE, 1993.

[6] D. S. Touretzky, "Seven big ideas in robotics, and how to teach them," in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pp. 39–44, 2012.

[7] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, *et al.*, "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," in *International Conference on Robotics and Automation (ICRA)*, pp. 1497–1504, IEEE, 2017.

[8] P. Gaskell, J. Pavlasek, T. Gao, A. Narula, S. Lewis, and O. C. Jenkins, "MBot: A modular ecosystem for scalable robotics education," in *International Conference on Robotics and Automation (ICRA)*, 2024.

[9] R. B. Shapiro, K. DesPortes, and B. DiSalvo, "Improving computing education research through valuing design," *Communications of the ACM*, vol. 66, no. 8, pp. 24–26, 2023.

[10] C. Solomon, B. Harvey, K. Kahn, H. Lieberman, M. L. Miller, M. Minsky, A. Papert, and B. Silverman, "History of LOGO," *Proceedings of the ACM on Programming Languages*, vol. 4, no. HOPL, pp. 1–66, 2020.

[11] M. M. McGill, "Learning to program with personal robots: Influences on student motivation," *ACM Transactions on Computing Education (TOCE)*, vol. 12, mar 2012.

[12] C. A. Berry, S. L. Remy, and T. E. Rogers, "Robotics for all ages: a standard robotics curriculum for k-16," *IEEE Robotics & Automation Magazine*, vol. 23, no. 2, pp. 40–46, 2016.

[13] B. Fagin, "Using Ada-based robotics to teach computer science," in *SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pp. 148–151, 2000.

[14] S. Papert, "Mindstorms: Children, computers, and powerful ideas," 1980.

[15] J. Summet, D. Kumar, K. O'Hara, D. Walker, L. Ni, D. Blank, and T. Balch, "Personalizing CS1 with robots," *ACM SIGCSE Bulletin*, vol. 41, no. 1, pp. 433–437, 2009.

[16] S. Evripidou, K. Georgiou, L. Doitsidis, A. A. Amanatiadis, Z. Zinonos, and S. A. Chatzichristofis, "Educational robotics: Platforms, competitions and expected learning outcomes," *IEEE access*, vol. 8, pp. 219534–219562, 2020.

[17] B. T. Fine, J. Denny, N. Dix, and A. Frazier, "Oh the robots that you can choose: A technical review of mobile robot platforms," *Journal of Computing Sciences in Colleges*, vol. 35, no. 8, pp. 126–135, 2020.

[18] D. Blank, L. Meeden, and D. Kumar, "Python robotics: An environment for exploring robotics beyond LEGOs," *ACM SIGCSE Bulletin*, vol. 35, no. 1, pp. 317–321, 2003.

[19] D. S. Touretzky and E. J. Tira-Thompson, "Tekkotsu: A framework for AIBO cognitive robotics," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20, p. 1741, Citeseer, 2005.

[20] M. M. Veloso, P. E. Rybski, S. Lenser, S. Chernova, and D. Vail, "CMRoboBits: Creating an intelligent AIBO robot," *AI magazine*, vol. 27, no. 1, pp. 67–67, 2006.

[21] B. C. Dickinson, O. C. Jenkins, M. Moseley, D. Bloom, and D. Hartmann, "Roomba Pac-Man: Teaching autonomous robotics through embodied gaming," in *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*, pp. 35–39, 2007.

[22] M. Conbere and Z. Dodds, "Toys and tools: Accessible robotics via laptop computers," American Association for Artificial Intelligence, 2007.

[23] M. Lapping-Carr, O. C. Jenkins, D. H. Grollman, J. Schwertfeger, and T. Hinkle, "Wiimote interfaces for lifelong robot learning," in *AAAI Spring Symposium: Using AI to Motivate Greater Participation in Computer Science*, pp. 61–66, 2008.

[24] B. Tribelhorn and Z. Dodds, "Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education," in *International Conference on Robotics and Automation (ICRA)*, pp. 1393–1399, IEEE, 2007.

[25] R. Amsters and P. Slaets, "Turtlebot 3 as a robotics education platform," in *Robotics in Education: Current Research and Innovations 10*, pp. 170–181, Springer, 2020.

[26] S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, J. R. Smith, S. Chouhury, C. Mavrogiannis, and F. Sadeghi, "MuSHR: A low-cost, open-source robotic racecar for education and research," *CoRR*, vol. abs/1908.08031, 2019.

[27] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "ROS: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, p. 5, Kobe, Japan, 2009.

[28] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4057–4062, 2010.

[29] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: ROS for non-ROS users," in *Robotics Research: The 15th International Symposium ISRR*, pp. 493–504, Springer, 2017.

[30] C. L. Hovey, D. P. Bunde, Z. Butler, and C. Taylor, "How do I get people to use my ideas? Lessons from successful innovators in CS education," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pp. 841–847, 2023.