2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #52982

A Progressive Robotics Curriculum: From Foundations to Machine Learning-Enabled Autonomous Systems

Dr. Stanley Baek, United States Air Force Academy

Dr. Stanley Baek is an Associate Professor in the Department of Electrical and Computer Engineering at the U.S. Air Force Academy (USAFA), which he joined in 2019. His current research interests include intelligent behavioral control of multiple robots, robust intelligence of multiple heterogeneous unmanned vehicles for navigation, searching, and tracking of multiple targets, and intelligent consensus algorithms for multiple mobile robots. He received his Bachelor's, Master's, and Ph.D. degrees from the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. From 2014 to 2019, he was an Assistant Professor at the University of Michigan, where he developed a new undergraduate robotics engineering program. From 2012 to 2014, he was a postdoctoral research engineer at the U.S. Air Force Academy, where he developed multiple-coordinated control algorithms for unmanned aerial vehicles performing a variety of tasks including intelligence, surveillance, reconnaissance, and tactical operations. In 2008, he was an electrical engineer at the U.S. Army Research Lab, where he investigated passive wing rotation of biologically inspired flapping wing robots with compliant joint mechanisms and developed design principles to mimic insect wing kinematics for optimal lift force.

A Progressive Three-Course Robotics Curriculum: From Embedded Systems to Machine Learning-Enabled Autonomous Systems

Abstract

This paper outlines a three-course robotics sequence developed at the U.S. Air Force Academy designed to build student expertise through hands-on learning and carefully scaffolded skill progression. The introductory course focuses on microcontroller programming and hardware interfacing, establishing a foundation in low-level control. The second course transitions students to Python and the Robot Operating System (ROS), applying sensor fusion, navigation, and autonomous behavior in mobile robotics projects. The advanced course shifts toward mathematical rigor, introducing neural networks, Kalman filtering for target tracking, and statistical estimation techniques including maximum likelihood and maximum a posteriori methods, all framed within machine learning theory applied to robotic systems.

Students progress from direct hardware manipulation to autonomous systems and ultimately to theoretical frameworks underpinning robotic intelligence. The curriculum balances practical implementation in early stages with analytical depth in the advanced course. Team-based projects throughout support ABET outcomes while fostering collaboration and system-level thinking. Assessment data show strong gains in technical proficiency and student confidence. We reflect on pacing, hardware choices, and the transition from implementation to theory, offering a model other institutions can adapt to prepare students for careers in autonomous systems and robotics engineering.

Introduction

Robotics education has emerged as a powerful pedagogical approach for engineering programs, offering unique opportunities to integrate multiple disciplines while engaging students through hands-on, project-based learning[1,2,3]. The interdisciplinary nature of robotics, spanning mechanical, electrical, and computer engineering along with mathematics and control theory, makes it an ideal subject for developing systems-level thinking and preparing students for modern engineering practices[4,5,6,7]. As autonomous systems become increasingly prevalent across industries from manufacturing to defense, the demand for engineers with comprehensive robotics expertise continues to grow[8,9].

Despite widespread recognition of the value of robotics education, significant challenges persist in curriculum design[10,11]. Programs must balance theoretical foundations with practical

implementation skills, manage the progression from basic concepts to advanced topics, and select appropriate hardware and software platforms within budget constraints[12,13,14]. The rapid evolution of robotics technologies—particularly in machine learning, artificial intelligence, and industry-standard frameworks such as the Robot Operating System (ROS)—further complicates curriculum development[15,16,17,18], as programs must remain current while maintaining pedagogical coherence. Additionally, robotics courses often require substantial laboratory resources, dedicated faculty mentoring, and careful coordination across prerequisite courses[19]. Programming proficiency, particularly in languages suited to robotics applications, has been identified as a critical prerequisite for student success[20].

Several institutions have reported successful robotics curriculum implementations, though most focus on either introductory courses or senior capstone experiences rather than comprehensive multi-course sequences[21,22,23]. Ahlgren *et al.*[1] documented effective practices including robot competitions for motivation, hands-on design projects, and team-based research, demonstrating significant achievement of ABET learning outcomes. Pack *et al.*[24] described senior design robotics projects, showing how robotics provides opportunities for students to apply accumulated knowledge across electrical engineering, computer engineering, and software development. Multi-disciplinary capstone projects involving unmanned aerial vehicles have demonstrated the effectiveness of robotics for integrating diverse engineering disciplines[25,26,27]. Neural network implementations in hardware have shown promise for analog electronics instruction[28]. Versatile frameworks continue to be developed to support diverse pedagogical approaches[29]. However, these reports primarily address individual courses rather than coordinated multi-course progressions that systematically build student expertise from foundational concepts through advanced theoretical frameworks.

This paper presents a comprehensive three-course robotics curriculum developed at the United States Air Force Academy that addresses the progression challenge through carefully scaffolded skills development. The curriculum begins with embedded systems programming and low-level hardware control, advances to mobile robotics using industry-standard Robot Operating System (ROS) frameworks, and culminates in mathematically rigorous machine learning methods for autonomous systems. This progressive structure allows students to experience the full robotics development stack—from direct register-level programming through high-level algorithmic design—while building competencies systematically across multiple semesters.

The paper makes several contributions to robotics engineering education. First, we present a complete curriculum design with detailed course descriptions, learning progressions, and assessment strategies that other institutions can adapt to their contexts. Second, we provide concrete evidence of effectiveness through multi-year assessment data demonstrating consistent achievement of learning objectives and ABET student outcomes. Third, we share practical implementation details including hardware platform selections, laboratory configurations, and course materials that address common resource constraints. Finally, we offer candid discussion of challenges encountered and lessons learned, including student workload management, the transition from hands-on to theory-intensive instruction, and strategies for maintaining student engagement across varying pedagogical approaches.

The remainder of this paper is organized as follows. The Curriculum Design Philosophy section describes the three courses in detail, explaining the pedagogical rationale behind topic selection

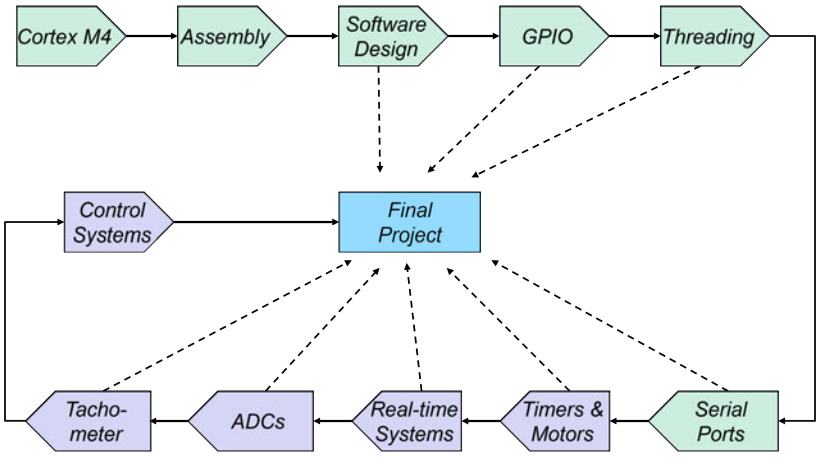


Figure 1: Progressive topic flow in embedded systems course showing how concepts build toward the final project

and progression. Implementation Details addresses practical considerations including hardware platforms, laboratory facilities, and course materials. The Pedagogical Approach section examines teaching methods, project structures, and assessment strategies employed across the sequence. Assessment and Outcomes presents multi-year data demonstrating learning objective achievement and student performance. Challenges and Lessons Learned candidly discusses difficulties encountered and solutions implemented. The Discussion section synthesizes findings, identifies best practices, and provides recommendations for other institutions. We conclude with reflections on the curriculum's effectiveness and future directions.

Curriculum Design Philosophy

Embedded Systems: Foundational Course

The embedded systems course serves as the foundation for the robotics curriculum, introducing students to microcontroller programming and hardware interfacing. This course targets junior electrical and computer engineering students who have completed introductory programming courses and a logic design course. The primary objective is to develop skills to design, implement, test, and debug microcontroller-based systems through hands-on experiential learning.

The course utilizes the Texas Instruments MSP432 ARM Cortex-M4 microcontroller as the hardware platform. This 32-bit processor provides sufficient complexity to teach modern embedded systems concepts while remaining accessible to undergraduate students. Students develop programs in both C and ARM assembly language, gaining exposure to both high-level and low-level programming paradigms.

Course content progresses systematically through fundamental embedded systems topics, as illustrated in Figure 1. Early labs focus on GPIO operations for interfacing with LEDs and switches, advancing to timing and interrupt concepts, serial communication protocols (UART, I2C, SPI), analog-to-digital conversion for sensor data acquisition, and motor control using PWM and H-bridge circuits. The final laboratory introduces proportional and integral control

Table 1: Introduction to Robotics Course Topic Progression

Week	Topic	Content
1-2	Linux & ROS2	Command-line interface, ROS2 nodes, topics, messages
3-4	Python & Gamepad	Python programming for ROS2, teleoperation control
5-6	TurtleBot & IMU	Robot motion control, IMU-based navigation
7-8	Transformation & LiDAR	Coordinate transformations, LiDAR-based perception
9-10	SLAM	Simultaneous Localization and Mapping
11-13	Computer Vision & AprilTag	Image processing, object detection, fiducial markers
14-16	Final Project	Autonomous navigation mission

fundamentals. The final project challenges student teams to design and implement an autonomous wall-following robot integrating infrared distance sensors, bumper switches, tachometers, and DC motors with control algorithms to navigate a maze.

Assessment emphasizes both individual understanding and practical implementation skills through homework assignments, laboratory demonstrations submitted via GitHub, and final project technical reports. Students consistently report that this course significantly strengthens their ability to identify and solve complex engineering problems (Outcome 1), apply engineering design (Outcome 2), and communicate effectively (Outcome 3).

Introduction to Robotics: Intermediate Course

The introduction to robotics course transitions students to higher-level mobile robotics concepts through Robot Operating System 2 (ROS2). This course targets junior and senior students who have completed programming fundamentals with Python. The course goal is for students to design, implement, test, and debug robotics-based systems by developing Python programs incorporating ROS2 functions and successfully interfacing the microcomputer with the external world.

The course employs Raspberry Pi single-board computers running Linux and ROS2, representing a significant shift from bare-metal microcontroller programming. Mobile robot platforms equipped with LIDAR, camera, and inertial measurement unit provide the physical testbed for algorithm development. Students program exclusively in Python, leveraging ROS2 libraries for robot control and perception.

Course content introduces students to the ROS2 ecosystem and publish-subscribe architecture, as summarized in Table 1. Early labs teach ROS2 fundamentals including nodes, topics, and messages. The curriculum progresses to sensor integration with IMUs and coordinate transformations, LiDAR-based perception, and SLAM concepts. Students implement autonomous behaviors including wall following, waypoint navigation, and goal-seeking. Computer vision using OpenCV enables stop sign detection and AprilTag recognition for visual localization. The three-week final project requires teams to implement autonomous navigation through a maze, culminating in formal demonstrations and technical reports.

This course effectively addresses ABET student outcomes 1 (problem solving), 3 (communication), 5 (teamwork), 6 (experimentation and data analysis), and 7 (lifelong learning).

The hands-on nature of mobile robotics projects generates high student engagement and motivation.

Advanced Robotics: Theory-Intensive Course

The advanced robotics course represents a significant departure from implementation-focused earlier courses, emphasizing mathematical rigor and theoretical foundations. This course targets senior students with strong mathematical preparation in linear algebra, probability theory, and statistics. The course goal is to provide fundamental knowledge and skills to design and develop machine learning algorithms for robotic problems.

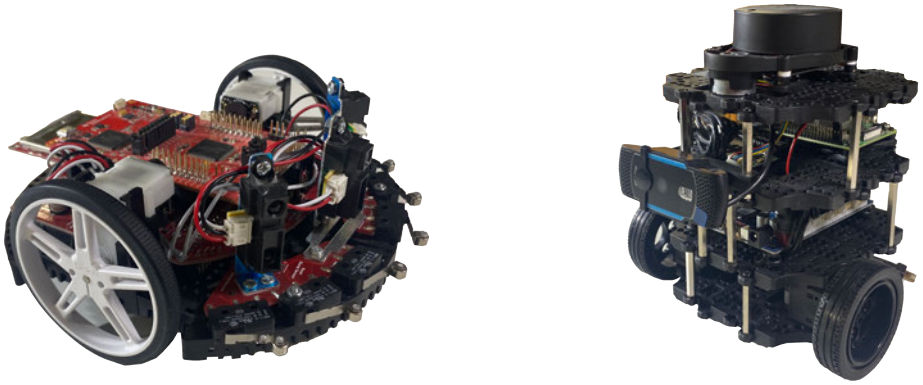
Unlike previous courses emphasizing hardware platforms, the advanced robotics course focuses on mathematical modeling and algorithm development. Students use Python with scientific computing libraries (NumPy, SciPy, scikit-learn) to implement and test algorithms. This design philosophy reflects progression from hands-on implementation to analytical reasoning, preparing students for advanced research and development roles.

Table 2: Advanced Robotics Course Biweekly Schedule

Weeks	**Topics**	**Contents**
1-2	Linear Algebra Foundations	ML fundamentals, basis, eigenvalues, eigenvectors
3-4	Probability & Estimation	LSE, Joint distributions, optimal and recursive estimation
5-6	Kalman Filtering	Derivation, implementation, applications, Project 1
7-8	Bayesian Inference & MLE	Bayes' theorem, maximum likelihood estimation
9-10	Regression Methods	MAP estimation, linear regression, gradient descent
11-12	Classification & Validation	Logistic regression, naive Bayes, regularization
13-14	Advanced Methods	SVM, neural networks, backpropagation
15-16	Final Project	Project development, implementation, presentations

The course is structured in two major modules, as shown in Table 2. The first module establishes mathematical foundations including linear algebra, probability theory, optimal estimation, and Kalman filtering. The second module focuses on machine learning algorithms including Bayesian inference, MLE/MAP estimation, supervised learning methods, and neural networks. This progression mirrors the curriculum's philosophy of building complexity systematically while deepening theoretical understanding.

Course content emphasizes derivation over application. Students derive algorithms from first principles, prove theoretical properties, and implement algorithms without pre-built library functions. This approach ensures deep understanding and prepares students to adapt algorithms for novel problems. Assessment includes homework requiring derivations and software implementations, two midterm exams, and two major projects. The theory-intensive nature represents significant adjustment but develops ability to apply mathematical knowledge (ABET outcome 1), engineering design (Outcome 2), communication (Outcome 3), and experimentation (outcome 6).



(a) TI-RSLK MAX (b) TurtleBot3 Burger

Figure 2: Robot platforms: (a) TI-RSLK MAX for embedded systems, (b) TurtleBot3 Burger for introduction to robotics

Implementation Details

Hardware and Software Platforms

Table 3 summarizes hardware and software platforms used across the three-course sequence, reflecting careful consideration of educational effectiveness, cost, industry relevance, and availability.

Table 3: Hardware and Software Platforms by Course

Course	**Hardware Platform**	**Software/Languages**	**Cost/Unit**
Embedded Systems	TI-RSLK MAX	C, ARM Assembly, Code Composer Studio	$200-250
Intro to Robotics	TurtleBot3	Python, ROS2 Humble, Ubuntu 22.04	$750-900
Advanced Robotics	laptop/desktop	Python, TensorFlow/Scikit-Learn	$0 (Own laptop)

The embedded systems course employs the TI-RSLK MAX (Figure 2a), integrating the MSP432P401R LaunchPad with ARM Cortex-M4 processor. This platform provides complete mobile robot with integrated sensors (infrared distance, Hall effect, light detection) and actuators, eliminating mechanical construction and allowing immediate focus on embedded programming. Students interface directly at register level with GPIO, timers, UART, I2C, SPI, ADC, and PWM peripherals. The department provides kits that students check out for the semester. The Valvano textbook[30] serves as primary resource, with supplementary materials at https://usafa-ece.github.io/ece382/.

The introduction to robotics course utilizes TurtleBot3 Burger platforms (Figure 2b) with Raspberry Pi 4 (8GB RAM) running ROS2 Humble on Ubuntu 22.04. This represents significant architectural shift from bare-metal programming to Linux-based environments. ROS2 has

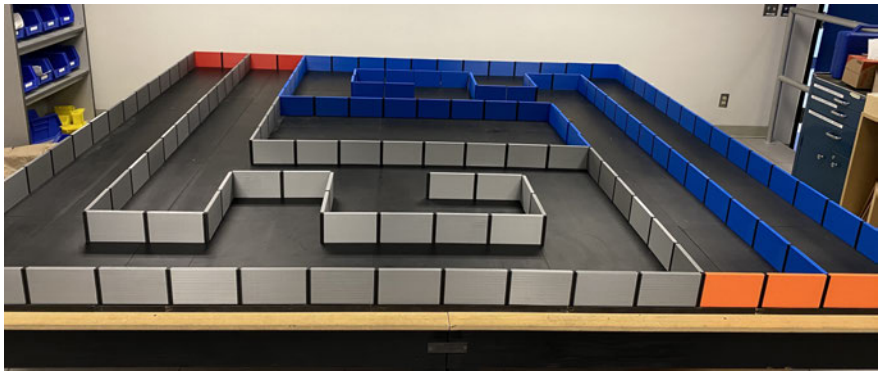


Figure 3: Custom-designed reconfigurable maze (2.88 m $\times$ 2.88 m) for autonomous robots

become the de facto standard framework for robotics research and industry. The platform provides 360-degree LIDAR, IMU, wheel encoders, and USB webcams for computer vision. Laboratory materials include tutorials on Linux, ROS2, Python, SLAM, and computer vision at https://stanbaek.github.io/ece387/.

The advanced robotics course requires only standard computing resources, eliminating additional hardware costs. Students use personal laptops with open-source tools including Python, MATLAB, TensorFlow, PyTorch, and scikit-learn. This software-focused approach emphasizes that mathematical concepts transcend specific platforms. Course materials are available at https://stanbaek.github.io/ece487/.

Laboratory Facilities and Resources

The robotics laboratory features a custom-designed reconfigurable maze system with modular blocks (18 cm length, 10 cm height, 1.3 cm width) in a 16×16 configuration creating a 2.88 m $\times$ 2.88 m testing arena (Figure 3). For TurtleBot3 exercises, blocks stack two-high for reliable LIDAR detection. The modular design allows reconfiguration between semesters while maintaining consistency within each term.

The laboratory provides 24 workstations with dual monitors and HP docking stations enabling quick laptop connections. Intel NUC computers serve as ROS2 master computers for the introduction to robotics course. Microsoft Teams channels enable students to ask questions and share insights outside class hours, with instructors monitoring but encouraging peer-to-peer learning.

Course Materials

The embedded systems course uses *Embedded Systems: Introduction to Robotics* by Valvano[30] as primary textbook, with laboratory assignments that reinforce and extend the material. The introduction to robotics course has no required textbook, relying on ROS documentation and online tutorials. The advanced robotics course similarly has no required textbook. While Pattern Recognition and Machine Learning by Bishop[31] and Probabilistic Machine Learning: An Introduction by Murphy[32] offer comprehensive treatments of the subject, both texts are too advanced for most undergraduates. Instead, two more accessible books are recommended:

Machine Learning Fundamentals: A Concise Introduction by Jiang[33] and Introduction to Machine Learning: From Math to Code by Wang[34]. The course relies primarily on instructor-developed lecture notes.

All courses maintain publicly accessible websites with syllabi, schedules, lab instructions, assignments, and resources: https://usafa-ece.github.io/ece382/, https://stanbaek.github.io/ece387/, and https://stanbaek.github.io/ece487/. GitHub repositories provide starter code and submission mechanisms, exposing students to version control practices.

Student Workload and Time Commitment

Each course carries three credit hours with roughly 40 hours in-class instruction for the semester. Out-of-class workload typically ranges from six to nine hours weekly. All three courses implement a "grace day" policy—students receive five self-granted extension days per semester (maximum two per assignment), accommodating scheduling conflicts or longer debugging sessions while maintaining accountability through eventual deadlines and late penalties (10–25% per day) after grace days are exhausted.

Student feedback consistently emphasizes starting assignments early. Representative quotes include "Don't even think about starting any assignments the night before" and "DO NOT wait to start homework/labs/projects." This led to implementing "early bird" incentives in some sections, providing bonus points for submissions 24–48 hours before deadlines.

Pedagogical Approach

Progressive Skill Development

The three-course sequence employs deliberate progression in complexity and abstraction across three dimensions: programming abstraction level, system complexity, and theoretical depth.

Programming abstraction progresses from register-level embedded C and assembly through Python with ROS2 middleware to mathematical algorithm implementation in scientific libraries. This exposes students to multiple paradigms while demonstrating appropriate tool selection. System complexity increases from individual peripheral interfaces to complete robotic systems to high-abstraction platforms for deploying machine learning algorithms. Theoretical depth increases from minimal mathematical formalism through moderate mathematical content to rigorous derivation-based instruction, recognizing that students require concrete experiences before engaging productively with abstract theory.

Project-Based Learning

All three courses employ project-based learning with varying scope and structure. The embedded systems final project (three weeks) challenges students to design autonomous wall-following robots integrating all course concepts. The introduction to robotics final project (three weeks) requires autonomous maze navigation with specific waypoints, permitting diverse solutions and rewarding novel approaches. The advanced robotics course includes two major projects: Kalman

filtering implementation for state estimation, and an two-week final project enabling deep exploration of topics including neural networks and support vector machines, with substantial technical reports (15–20 pages).

Assessment Methods and Grading Strategies

Assessment employs multiple methods aligned with learning objectives. Homework assignments serve different purposes: embedded systems reinforces C programming and peripheral configuration; introduction to robotics emphasizes ROS2 and Python proficiency; advanced robotics requires mathematical derivations and algorithm implementation. All courses return homework with detailed feedback within 2–4 days. Laboratory assignments require working demonstrations with GitHub submissions. Examinations assess individual theoretical understanding, with recent averages approximately 75% across all courses, indicating appropriate difficulty levels.

Assessment and Outcomes

Course Objective Achievement

Multi-year assessment data demonstrate consistent achievement of learning objectives across all three courses. Tables 4, 5, and 6 summarize objective achievement over recent academic years.

Table 4: ECE 382 (Embedded Systems) Course Objective Achievement (2020-2024)

Objective	**2024**	**2023**	**2022**	**2021**	**2020**
Obj 1: Utilize microcontroller units	97.1%	97.7%	97.2%	93.0%	92.8%
Obj 2: Write assembly language	86.8%	93.8%	80.2%	82.3%	82.1%
Obj 3: Write C programming	89.4%	92.2%	88.7%	89.7%	88.1%
Obj 4: Communicate technical work	94.5%	93.0%	91.3%	84.9%	86.8%
Obj 5: Debug and modify programs	95.6%	96.7%	95.4%	90.2%	89.6%
Obj 6: Interface hardware	93.0%	95.8%	91.7%	89.2%	86.9%

The embedded systems course shows strong performance across all objectives, with consistent improvement from 2020 to 2024 in most areas. Objective 2 (assembly language programming) exhibits the greatest variability, ranging from 80.2% in 2022 to 93.8% in 2023. This variability likely reflects adjustments in exam difficulties and emphasis on assembly versus C programming. The 2024 decrease to 86.8% followed increased emphasis on higher-level conceptual questions rather than detailed register configurations, representing a deliberate pedagogical shift toward deeper understanding over memorization.

The introduction to robotics course was offered as a pilot for two years prior to 2024, during which no formal assessment data were collected. The two-year assessment period shown in Table 5 demonstrates solid performance across all objectives, with Objective 2 (utilizing Linux OS) showing consistently strong achievement at 92.0% and 90.7%. Objectives 4–6 remain stable in the 84–86% range. However, Objectives 1 and 3 show notable declines, with Objective 3

Table 5: ECE 387 (Introduction to Robotics) Course Objective Achievement (2024-2025)

Objective	**2025**	**2024**
Obj 1: Understand basic robotics concepts	85.4%	91.8%
Obj 2: Utilize Linux OS for robotic applications	92.0%	90.7%
Obj 3: Write and run ROS code in Python	85.0%	88.2%
Obj 4: Interface sensors and actuators	86.3%	86.4%
Obj 5: Implement robot algorithms in software	86.4%	85.4%
Obj 6: Evaluate laboratory outcomes in writing	85.8%	84.1%

(ROS/Python programming) decreasing from 88.2% to 85.0%. The course assessment report identifies Python programming proficiency as a significant challenge, with many students lacking experience in object-oriented programming and third-party library usage despite having completed introductory programming courses. This programming gap likely contributes to the moderate achievement rates in implementation-focused objectives.

Table 6: ECE 487 (Advanced Robotics) Course Objective Achievement (2023-2025)

Objective	**2025**	**2024**	**2023**
Obj 1: Understand ML concepts	86.3%	90.1%	89.6%
Obj 2: Develop equations for ML methods	90.8%	92.9%	90.0%
Obj 3: Analyze ML problems	84.7%	94.8%	86.3%
Obj 4: Implement ML algorithms	94.0%	96.1%	89.1%
Obj 5: Evaluate algorithm performance	93.4%	94.7%	86.6%

The advanced robotics course shows strong overall performance with notable year-to-year variations. The 2024 cohort demonstrated exceptional performance across all objectives, averaging 93.7% compared to 89.8% in 2025 and 88.3% in 2023. Course assessment reports attribute this variation partially to differences in student academic preparation—the 2025 cohort had significantly lower average GPA (3.49) compared to 2024 (3.70), limiting direct comparability. Objective 3 (analyzing and solving ML problems) shows the greatest variation, from 94.8% in 2024 to 84.7% in 2025, possibly reflecting increased difficulty in graded review problems or reduced student preparedness.

Student Performance and Persistence

Enrollment and completion data reveal strong student engagement across the sequence. The embedded systems course, required for electrical and computer engineering majors, typically enrolls 35–45 students per semester with completion rates exceeding 96%. The introduction to robotics course, offered as an elective, attracts 20–28 students per offering with similar high completion rates. The advanced robotics course, offered to seniors with strong mathematical backgrounds, enrolls 12-18 students annually with 100% completion in recent years.

Grade distributions indicate appropriate difficulty levels across courses. The embedded systems course maintains final grade averages around 91.1%, with approximately 45% of students earning A- or higher. The introduction to robotics course shows similar patterns with average grades

around 90.1%. The advanced robotics course, despite greater mathematical rigor, maintains comparable grade distributions (average 90.6% in 2025), though with greater standard deviation reflecting varying student mathematical preparation.

Workload data from student surveys indicate consistent time commitments of 6–9 hours per week outside class time across all three courses. Despite efforts to reduce embedded systems course load by approximately 30% based on student feedback, survey results indicate no corresponding decrease in time commitment, suggesting that students expand effort to match available time or that baseline difficulty remains substantial regardless of assignment quantity.

Student Feedback and Qualitative Assessment

Course evaluation data and informal student feedback provide valuable qualitative insights complementing quantitative performance measures. Consistent themes emerge across multiple semesters and courses.

Students consistently appreciate the hands-on nature of robotics projects, reporting higher engagement and motivation compared to traditional lecture-based courses. Comments such as "finally got to see theory applied to real systems" and "watching my robot successfully navigate the maze was the highlight of my semester" reflect the intrinsic motivation generated by tangible accomplishments. However, students also acknowledge frustration when debugging hardware issues consumes disproportionate time, particularly in embedded systems and introduction to robotics courses where hardware failures or configuration errors can block progress for hours.

The transition from hands-on robotics to theory-intensive machine learning in the advanced course generates mixed reactions. Some students embrace the mathematical rigor, appreciating deeper understanding of algorithms they previously used as "black boxes." Others struggle with the abstraction shift, expressing desire for more physical robot applications. Course modifications in response to this feedback include explicitly connecting mathematical concepts to robotics applications and, when feasible, demonstrating algorithms on physical platforms after theoretical development.

Regarding programming challenges, the introduction to robotics course assessment report identified Python proficiency gaps as significant obstacles. While students complete introductory programming courses, many lack experience with object-oriented programming, data structures beyond basic lists, and third-party library documentation. Troubleshooting skills also require development—students often request instructor assistance before systematically interpreting error messages or consulting documentation. Future offerings will include focused Python refresher modules early in the semester and explicit instruction in debugging strategies.

Challenges and Lessons Learned

Curriculum Pacing and Content Coverage

Balancing breadth of topics against depth of understanding presents ongoing challenges across all three courses. Initial embedded systems offerings attempted comprehensive peripheral coverage,

resulting in superficial treatment. Subsequent refinements reduced peripheral coverage while increasing depth on fundamental concepts, producing better student outcomes. Similarly, the introduction to robotics course initially attempted broad sensor integration coverage but now focuses more narrowly on core ROS2 competencies and computer vision applications, producing stronger learning outcomes as evidenced by improved assessment scores.

Coordination across courses requires ongoing attention. Students occasionally enter courses with weaker prerequisite skills than assumed due to time elapsed since foundational courses. Implementing prerequisite refresher modules early in each semester helps address these predictable gaps.

Software Tool Evolution

Rapid evolution of robotics software tools presents ongoing curriculum maintenance challenges. The introduction to robotics course transitioned from ROS1 to ROS2 between 2024–2025, requiring complete revision of all materials. Texas Instruments' December 2024 release of VS Code-based Code Composer Studio similarly necessitates updates to embedded systems materials. Establishing sustainable processes for curriculum maintenance proves essential—dedicating one to two weeks during summer break for updates and maintaining detailed change logs help manage software evolution impacts, though ongoing time commitment remains substantial at approximately 10–15% of instructor effort.

Transition from Implementation to Theory

The progression from hands-on courses to theory-intensive machine learning presents significant adjustment challenges. Students entering advanced robotics have strong practical skills but variable mathematical preparation, particularly in probability theory and linear algebra. Mathematical prerequisite gaps manifest most clearly in Kalman filtering content requiring fluent manipulation of multivariate Gaussian distributions and matrix operations.

Bridging strategies include concrete numerical examples before abstract formulations, computational exercises demonstrating algorithmic behavior, and explicit connections between mathematics and robotics applications. Beginning Kalman filtering instruction with specific numerical examples tracking robot position from noisy GPS measurements, developing intuition through visualization, and only subsequently generalizing to abstract formulations improves student comprehension. Students who implement algorithms from scratch develop significantly deeper understanding than those who merely apply library functions, though implementation assignments consume substantial time.

Discussion

Effectiveness of Progressive Approach

Assessment data provide substantial evidence for the effectiveness of the curriculum design. Consistent achievement of learning objectives across all three courses, with most exceeding 85%, demonstrates successful competency acquisition. Student performance on ABET outcomes shows steady improvement in embedded systems from 87.7% (2020) to 93.7% (2024) for

problem-solving, suggesting ongoing refinements enhance learning effectiveness. Introduction to robotics assessment reveals strong performance on integration objectives (86.2% average), indicating successful skill transfer from embedded systems. Advanced robotics students demonstrate high implementation proficiency (94.0%) despite mathematical rigor, suggesting prior programming provides solid foundations.

Comparison with literature provides additional context. Single-course robotics offerings cannot achieve the depth of specialized expertise developed through multi-course sequences. Our three-course progression enables students to experience low-level embedded programming, systems-level frameworks, and advanced algorithmic development—preparing them for diverse autonomous systems careers. Programs emphasizing only senior capstone projects risk shallow understanding, while those focusing exclusively on theory may produce students who struggle with implementation. The balance through progressive hands-on then theory-intensive courses addresses both concerns.

Best Practices and Generalizable Lessons

Several practices emerge as particularly effective, supported by assessment data and student feedback. Early and sustained hands-on experience develops engineering intuition—Objective 1 (utilizing microcontroller units) consistently exceeding 95% validates extensive hands-on practice effectiveness. Explicit skill progressions that build on prior knowledge enable students to tackle increasingly complex challenges—introduction to robotics integration objectives averaging above 85% demonstrates strong development of these skills. Project-based learning with authentic challenges generates substantially higher engagement, with student comments emphasizing that successful robot missions provide powerful motivation.

The deliberate progression from bare-metal programming through middleware to algorithmic machine learning contrasts with programs maintaining consistent abstraction levels, exposing students to the full software stack. Substantial emphasis on mathematical rigor, including derivations from first principles, exceeds typical undergraduate coverage, yet performance data (Objective 2 averaging 91.2%) demonstrate undergraduates can achieve sophisticated mathematical understanding when properly supported.

Several lessons generalize beyond this specific curriculum. Progressive design requires explicit attention to prerequisites and systematic competency building. Balancing hands-on and theoretical content across sequences enables addressing both practical skills and conceptual foundations. Sustainable robotics programs require substantial ongoing investment in equipment maintenance, laboratory support, and curriculum updates—not merely initial purchases. Student time commitment inherently exceeds traditional courses; acknowledging this through credit allocation or coordination with other courses proves preferable to attempting unsustainable reductions.

Conclusions

This paper has presented a comprehensive three-course robotics curriculum at the U.S. Air Force Academy that systematically develops student expertise from embedded systems foundations

through machine learning-enabled autonomous systems. Assessment data spanning multiple academic years demonstrate consistent achievement of learning objectives across all courses, with most objectives exceeding 85% and several surpassing 95%. ABET student outcomes data show strong performance in problem-solving (93.7%), engineering design (94.6%), and communication (94.5%), validating robotics education's effectiveness for developing core engineering competencies. The progressive structure—moving from low-level hardware control through systems-level robotics frameworks to rigorous mathematical algorithmic development—enables students to experience the full autonomous systems development stack while building competencies through carefully scaffolded learning progressions. Key best practices include early hands-on engagement that develops essential intuition, clearly defined skill progressions that build on prior knowledge, project-based learning with authentic problems that generate high motivation, and integration of theoretical and practical content that maintains relevance.

For institutions seeking to develop or enhance robotics curricula, this paper offers both inspiration and practical guidance. The progressive three-course model proves effective but requires substantial resource investment including dedicated laboratory space, equipment budgets accommodating ongoing maintenance, technician support, and sustained faculty effort. Institutions should calibrate implementation to their specific contexts rather than attempting direct replication. As autonomous systems proliferate across industries, engineers who understand both practical implementation and theoretical foundations will remain in high demand. The model presented here, grounded in multi-year assessment evidence and refined through continuous improvement, offers one effective approach validated by consistent learning outcomes and strong student performance. We encourage robotics educators to adapt these ideas to their contexts and collaborate in advancing effective robotics pedagogy to meet society's growing needs for qualified autonomous systems engineers.

Disclaimer

PA#: USAFA-DF-2026-177. The views expressed in this article are those of the author and do not necessarily reflect the official policy or position of the United States Air Force Academy, the Air Force, the Department of Defense, or the U.S. Government.

References

[1] David J. Ahlgren, Igor M. Verner, Daniel Pack, and Steve Richards. Effective practices in robotics education. In *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition*. ASEE, 2004.

[2] Fred G. Martin. *Robotic Explorations: A Hands-On Introduction to Engineering*. Prentice Hall, 2001. ISBN 978-0130895684.

[3] Illah Nourbakhsh, Emily Hamner, Baker Dunlavey, David Bernstein, and Kevin Crowley. Educational robotics and engineering education. In *NSF Workshop on Research in K-12 Robotics Education*. National Science Foundation, 2004.

[4] Frank Klassner and Steven D. Anderson. A case study of lego mindstorms' suitability for artificial intelligence and robotics courses at the college level. *ACM SIGCSE Bulletin*, 35(1):8–12, 2003.

[5] Deepak Kumar and Lisa Meeden. Robotics in computer science education. *ACM SIGCSE Bulletin*, 36(2):1–5, 2004.

[6] M. A. Gennert, T. Padir, et al. Designing an undergraduate robotics engineering curriculum: Unified robotics I and II. In *ASEE Annual Conference Proceedings*, 2018.

[7] C. A. Berry, M. A. Gennert, and R. M. Reck. Practical skills for students in mechatronics and robotics education. In *Proceedings of 2020 ASEE Virtual Annual Conference*, June 2020.

[8] George A. Bekey. *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT Press, 2005. ISBN 978-0262025782.

[9] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer, 2nd edition, 2016. ISBN 978-3319325507.

[10] Soo Hyun Kim and Jong Wook Jeon. Educational robotics for all ages. *International Journal of Advanced Robotic Systems*, 12(5):1–10, 2015.

[11] A. Al-Emran et al. Research and education in robotics: A comprehensive review, trends, challenges, and future directions. *Machines*, 14(4), 2025.

[12] Daniel J. Pack, Robert Avanzato, David J. Ahlgren, and Igor M. Verner. Fire-fighting mobile robotics and interdisciplinary design: Comparative perspectives. *IEEE Transactions on Education*, 47(3):369–376, 2004.

[13] Alessandro Valentini, Giuliano Donzellini, and Donatella Ponta. Educational robotics: A learning tool for all school levels. *International Journal of Online Engineering*, 5(3), 2009.

[14] L. Cehovin Zajc, A. Rezelj, and D. Skocaj. Low-cost open-source robotic platform for education. *IEEE Transactions on Learning Technologies*, 16(1):18–25, February 2023.

[15] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010. ISBN 978-0136042594.

[16] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005. ISBN 978-0262201629.

[17] L. Ma, Y. Wang, C. Xu, and X. Li. Introducing ROS-projects to undergraduate robotic curriculum. In *Proceedings of 2023 ASEE Annual Conference & Exposition*, Baltimore, Maryland, June 2023.

[18] S. R. Oca and B. Hament. Implementing and using ROS in undergraduate robotics curricula. In *Proceedings of 2024 ASEE Annual Conference & Exposition*, Portland, Oregon, June 2024.

[19] Robert Avanzato. Educational robotics: A snapshot of the college programs. In *Proceedings of the 2004 American Society for Engineering Education Annual Conference*. ASEE, 2004.

[20] M. Agheli, G. Lewin, and M. Nemitz. WIP: The necessity of an RBE-tailored first-year programming course in the robotics engineering curriculum. In *Proceedings of 2024 ASEE Annual Conference & Exposition*, Portland, Oregon, June 2024.

[21] Eileen Wings. Robotics as a gateway to higher learning. In *Proceedings of the 2004 American Society for Engineering Education Annual Conference*. ASEE, 2004.

[22] Jenay M. Beer, Arthur D. Fisk, and Wendy A. Rogers. Towards a framework for levels of robot autonomy in human-robot interaction. *Journal of Human-Robot Interaction*, 3(2):74–99, 2014.

[23] M. Ciletti and G. Plett. Piloting a balanced curriculum in electrical engineering introduction to robotics. In *Proceedings of 2005 ASEE Annual Conference*, Portland, Oregon, June 2005.

[24] J. P. Perlin, Daniel J. Pack, Barry E. Mullins, and R. E. Speakman. Senior capstone design experience: Hovering robot. In *Proceedings of the 2003 ASEE Annual Conference*, Nashville, TN, 2003. ASEE.

[25] Jeffrey E. Hubbard, Daniel J. Pack, and Jack S. Elston. A multi-disciplinary senior design project using cooperative unmanned aerial vehicles (uavs). In *Proceedings of the 2006 ASEE Annual Conference*. ASEE, 2006.

[26] Stephen P. Trimble, Daniel J. Pack, and Barry E. Mullins. Multi-disciplinary capstone design project: An unmanned aircraft system (uas) for vehicle tracking. In *Proceedings of the 2010 ASEE Annual Conference*. ASEE, 2010.

[27] J. L. Moore, Daniel J. Pack, Jack S. Elston, and Brian Reese. Assessment of a multi-university unmanned systems capstone design project. In *Proceedings of the 2012 ASEE Annual Conference*. ASEE, 2012.

[28] Joshua Alspector, A. Jayakumar, and S. Luna. A simple hardware implementation of neural networks for instruction in analog electronics. In *Proceedings of the International Joint Conference on Neural Networks*, pages 679–684. IEEE, 1991.

[29] B. Riopelle, A. Narula, S. Sun, and P. Gaskell. Designing a versatile robot framework for undergraduate robotics education. In *Proceedings of 2025 ASEE Annual Conference*, 2025.

[30] Jonathan W. Valvano. *Embedded Systems: Introduction to Robotics*. Jonathan Valvano, 2019. ISBN 978-1074544300.

[31] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 978-0387310732.

[32] Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022. ISBN 978-0262046824.

[33] Hui Jiang. *Machine Learning Fundamentals: A Concise Introduction*. Cambridge University Press, 2021. ISBN 978-1108835132.

[34] Ruye Wang. *Introduction to Machine Learning: From Math to Code*. Springer, 2023. ISBN 978-3031338024.