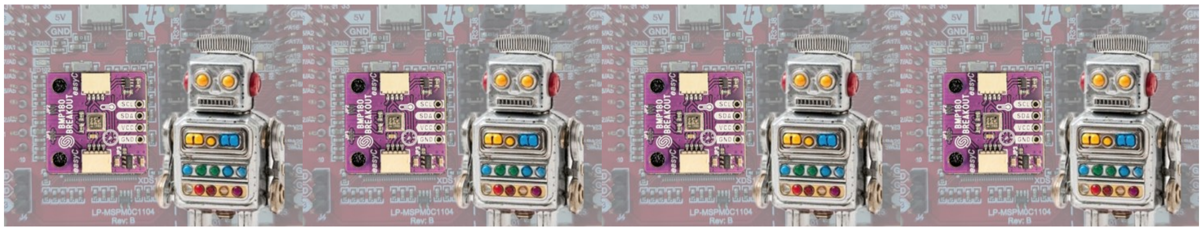
Robotics and Sensors (ENR 355)

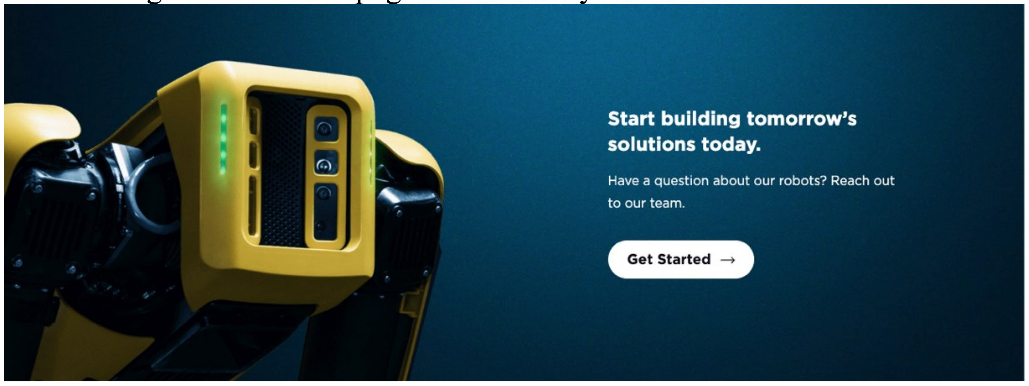


Spring, 2026 (1/14/2026 - 5/8/2026)
Day and Time: Tuesday and Thursday 9:00-9:50 AM
Location: HSC/Peterson 148
Max enrollment: 20 Credit: 1.00
Professor: Xiang Li
Contact Information: xili@coe.edu
Student Hours: walk-in (MW 9:30 am – 12:00 pm MT 1:00 pm – 3:00 pm) or by individual appointments
Department Chair: Firdevs Duru, fduru@coe.edu

Course Description:

In Nov 3 2025, the New York Times reported that Amazon is planning to replace 75% of its human workforce with robotics and automation, ETA 2033[1], *enough said*?

Or a message from the webpage of Boston Dynamics[2]:



Start building tomorrow's solutions today.

Have a question about our robots? Reach out to our team.

ENR 355 serves as an introductory course to mechatronics[3]. A good robotic application is just like any other automation: it is the system integration of mechanical, electrical, computer hardware and software.

If you have done grade-school robotics and played with Lego before, it won't be hard to put things together and make it move; and it won't be hard to power the movements with a motor. In this class, we are going to study what is under the hood: what is the design, math,

[1] https://www.nytimes.com/2025/10/21/technology/inside-amazons-plans-to-replace-workers-with-robots.html
[2] https://bostondynamics.com/ last accessed: Nov 17, 2025
[3] https://en.wikipedia.org/wiki/Mechatronics

signal and code to make movements with controls. When other kids are still fine-tuning each turning and steps manually, we can actually save lots of time by letting the bots figure it out themselves with sensors, feedback loops and computation.

Thanks to the maker culture, it is now possible to perform any robotic tasks with commercial-grade hardware and open-sourced software support. I have no doubt if you can learn how to build a servo/motor-sensor-control system in this class, you will have no issue working with, or even develop R&D or industrial robots in the future.

The lectures and knowledge base will be delivered based on the three major milestones:

- 3RPS parallel manipulator with stepper motor and position sensor.
- Based on micromouse[4]: maze-solving small wheeled robots since 1977.
- Build a robotic system powered by reinforced learning (RL) on NVIDIA Isaac Sim.

We are also going to weave many hands-on sessions into the course:

- 3D CAD for mechanical design and rapid prototyping
- 3D printing for rapid prototyping
- Laser cutter for rapid prototyping
- Simple power tools
- PCB soldering
- PCB digital design
- Robotic simulation

Measurable Learning Objectives:

For overall course objectives, the students will be able to:

- Integrate fundamental mechanical, electrical, and computer hardware/software principles to create a functional mechatronic system.
- Acquire mathematical concepts and skill sets for modern robotics.
- Apply control theory (PID) and math models (inverse kinematics), including sensor feedback loops and computation, to create autonomous robotic behaviors.
- Justify the selection of specific sensors, actuators, and fabrication methods for a given robotic task.
- Create a basic robotic simulation (NVIDIA Isaac Sim) to develop a robotic policy before physical implementation.
- Experience such learning loop: design, build, test, make mistakes, reflect, revise, and demonstrate growth.

Here're the objectives break-down based on milestones:

Module 0: Digital Fabrication & Mechanical Design

- Design a multi-part mechanical assembly in 2D or 3D CAD software.
- Select appropriate fabrication methods (3D printing vs. laser cutting) based on a part's design constraints.

[4] Intro by Veritasium: https://www.youtube.com/watch?v=ZMQbHMgK2rw&t=305s

- Fabricate designed parts in 3D printing and laser cutting, accounting for real-world tolerances (e.g., resolutions, fabrication confinement, print settings, etc.).
- Assemble a 1-DOF mechatronic system, ensuring mechanical rigidity and a low-friction pivot. Understand how a stepper motor operates. how to control the angular steps and convert it to required motions.
- To recall and apply the following fundamental mathematical concepts: coordinate systems (Cartesian, mostly); vector and matrices (define matrices as arrays, matrix operations, determinant and inverses of a matrix), and describing the geometric transformations with vector mapping.

Module 1: 3RPS parallel manipulator with stepper motor and position sensor (Ball Balancer)

- Explain the operational principles, hardware components and pin functions of a stepper motor and a driver.
- Can calculate and realize the power consumption and safety measurement (voltage divider or buck convertor) for motor and microcontroller.
- Implement coding (with or without library) to control a stepper's position, speed, and acceleration.
- Describe the operation principle of a distance sensor and identify its key specs and limitations.
- Implement coding to read raw sensor data, and use filter and calibration map to convert the raw data to position measurement.
- Mathematically define the position and orientation of the platform, apply transformation matrices (Jacobian matrix) to enable the platform control.
- Implement PID: explain the specific function of proportional, integral and derivative components of the PID controller.
- Integrate the whole system (sensor, motor and controller) to a single main loop.
- PID system tuning and testing.
- Inverse kinematics: on a 3 DOF model, identify and measure key parameters and key geometric relationship to convert motor angles into desired tilting movements.
- Based on the previous 1 DOF model, use the same or different position sensing mechanism to map a 2D coordinate system.
- Model the ball balancing problem with inputs and control loops.
- Integrate the whole system (sensor, motor and controller) to achieve 3 DOF balancing of a ball.
- Analyze and debug common system failures, distinguishing between mechanical (e.g., backlash), sensor (e.g., noise), and software (e.g., loop speed) issues.

Module 2: Micromouse (Maze Solver)

- Design and fabricate a small, 2-wheel differential drive robot chassis.
- Integrate DC motors with encoders or stepper motors for precise wheel control.
- Implement a PID control loop for accurate forward movement (odometry) and point turns.
- Calibrate and deploy an array of IR or Time-of-Flight (ToF) sensors for wall detection. Implement a continuous feedback loop to avoid wall crashing.

- Use vectors to represent the position of the robot, and use homogeneous transformation matrices for localization and mapping.
- Implement an encoder to record the movements (odometry).
- Implement a "wall-following" algorithm as a basic navigation strategy.
- Analyze and implement a complete maze-solving algorithm (e.g., Flood Fill) to find the optimal path from a start to a center square.

Module 3: RL robotics

- Isaac Sim fundamentals: import robot assets and pre-configured training environment to go through a training session.
- Design your own robotic system in CAD.
- Simply and convert the CAD design file to urdf format and import to Isaac Sim.
- Configuring the design with physics properties to simulate realistic interactions.
- Implement control mechanisms with OmniGraph and ROS 2.
- Adding sensors and data streaming pipeline to ROS 2.
- Use Script Editor to write and run scripts to simulate the scene.
- Use Markov decision process to define a task and run a training.
- Transfer robot learning policies from simulation to reality (sim-to-real).

Cross-Disciplinary Skills (From Hands-on Sessions)

- Design a simple, custom Printed Circuit Board (PCB) in a digital design tool (e.g., KiCad, EasyEDA, Fusion or Altium) to solve an integration problem (like a motherboard for ball balancer, or a drive train for a micromouse.
- Assemble and solder through-hole (and maybe basic surface-mount) components onto a PCB.
- Utilize simple power tools (e.g., drill, and Dremel) for modifying and assembling mechanical parts.
- Document project progress and results in clear, professional "slide decks" for presentation.

Required Texts and/or Course Materials:

- Lecture slides and resources will be posted and maintained on Moodle and GitHub.
- There is no traditional textbook; curated online references and readings will be provided throughout the term.
- Recommended textbook if you like to read one. In this course, contents from chapter 2-6 will be covered to do classical robotics.
  Lynch, Kevin M., and Frank C. Park. *Modern robotics*. Cambridge University Press, 2017. Free online link: https://hades.mech.northwestern.edu/index.php/Modern_Robotics
- The links to recommended book chapters and tutorials will be provided on Moodle and GitHub.

Course Policies:

Official catalog description: Introduces the basics of robotics and automation; mechanical design, principles of motion and kinematics for automated devices. The students will complete projects while learning about the system requirements identification, sensor, and microcontroller integration. The course exposes students to the current research in robotics research such as autonomous vehicles, assistant robots, and field robots for extreme conditions. Prerequisite: Electromagnetism (PHY-265).

Schedule of major due dates: Homework assignments typically due one week later it's assigned.

Almost none of the assignments have point values, the ones do are given 0 (Retry) or 1 (exemplary/success) points. There are no partial credit and no averaging.
The work you turn in will be evaluated against quality standards that will be made clear on each assignment. If your work meets the standard, you will receive full credit for it. You will also get feedback, a chance to revise and update your grading.

The goal is to create a learning loop: try something, make mistakes, reflect, revise, and try again—without penalty for mistakes, as long as you demonstrate growth.
This process is mirror how evaluation of work happens post college. In your futural jobs, your manager (or PI in academia) will provide you feedback, and you will then demonstrate growth. A good project team will always move forward by such iteration.

The individual types of assignments are listed as follows:

Assignment	**Basis for grading**	**What will be recorded**
Foundation skills	Overall correctness and degrees of completion	Success/Retry
Applications	Completeness and overall correctness	Exemplary/Success/Retry
Reports	Completeness, overall correctness, and writing	Exemplary/Success/Retry
Advanced skills	Completeness and efforts	Exemplary/Pass/No pass
Presentation	Writing, documentation, and presentation	Pass/No pass

Method for determining final grade: Based on the nature of each section (concepts, math, coding, debugging), we will have tasks in the form of

- Homework (typically due one week later, can be anything from work out bugs to literature review and report).
- In-class pop quiz (work out a math problem with pen and paper, or debugging a code with all the help you can get.
- In-class real-time demo (build a project/server/notebook and bring it online).
- In-class presentation (introduce or showcase a theory, a model, a project … to the classroom)

Your grade is assigned with the table listed below. Each row indicates the minimum number of successful results needed to satisfy the requirement for that grade. For example, you will be required to earn 10 or more success mark on foundation skills to get grade of B.

Grade	**Foundation**	**Applications**	**Reports**	**Advanced skills**	**Cross-Disciplinary**
A	12	6	2	1	1
B	10	5	1	1	1
C	8	3	1	0	1
D	6	2	0	0	0

A grade of “F” is assigned if not all of the requirement for a “D” are met.
Plus grades: A "plus" is given on a letter grade if you satisfy all the requirements plus one of the following:

- More than half of the recorded grades are exemplary, when applicable.
- Complete two categories in the grade table for the next grade level up, no including presentation.

Minus grades: no “minus” will be given on a letter grade.

Here’re the learning objectives grouped by category:

Foundation skills

1. **[ ]** (Module 0) Design a multi-part assembly in 3D CAD software.
2. [ ] (Module 0) Design a laser cutting pattern in 2D CAD software.
3. [ ] (Module 0) Math applications of coordinate systems (Cartesian, spherical and cylindrical).
4. [ ] (Module 0) Math applications of vector and matrices (define matrices as arrays, matrix operations, determinant and inverses of a matrix).
5. [ ] (Module 0) Describing the geometric transformations with vector mapping.
6. **[ ]** (Module 1) Explain the operational principles of a stepper motor and a driver.
7. **[ ]** (Module 1) Identify all key hardware components and their pin functions for 1-DOF actuator system.
8. **[ ]** (Module 1) Calculate and set the correct current limit on the driver.
9. **[ ]** (Module 1) Wire the complete 1-DOF actuator system, matching the voltage for the driver and control system.
10. **[ ]** (Module 1) Describe and demonstrate the time-of-flight operating principle of an ultrasonic sensor and its limitations.
11. **[ ]** (Module 1) Interface a distance sensor with the 3.3V Raspberry Pi Pico.
12. **[ ]** (Module 1) Explain the specific function of the Proportional (P), Integral (I), and Derivative (D) components of a PID controller.
13. **[ ]** (Module 1) Explain the math foundation (geometrical calculation) of inverse kinematics.
14. **[ ]** (Module 1) Identify and measure the key geometric parameters of the 3-DOF platform for its kinematic model.
15. **[ ]** (Module 1) Explain the position sensor system wither with distance sensor or touch screen sensor.
16. **[ ]** (Module 1) Explain how the X-axis PID (pitch) and Y-axis PID (roll) work together.
17. **[ ]** (Milestone 2) Integrate motors and an array of IR/ToF sensors for wall detection.
18. **[ ]** (Milestone 3) Reinforcement Learning (RL) Theory: Define "State," "Action," and "Reward" in the context of a robotic balancing task
19. **[ ]** (Milestone 3) Implement robotic simulation with pre-made assets and training environment.
20. **[ ]** (Cross-Skill) Utilize simple power tools for modifying and assembling parts.
21. **[ ]** (Cross-Skill) Design a simple, custom Printed Circuit Board (PCB) in a digital design tool.
22. **[ ]** (Cross-Skill) Assemble and solder through-hole components.

2. Applications (Demos)

1. **[ ]** (Module 0) Fabricate and assemble a functional 1-DOF mechatronic system with a low-friction pivot.
2. **[ ]** (Module 0) Implement codes to control a stepper's position, speed, and acceleration.

3. **[ ]** (Module 0) Implement codes to read, filter, and calibrate raw sensor data into a reliable 1D position.
4. **[ ]** (Module 0) Integrate and tune gains to achieve a stable 1-DOF balance.
5. **[ ]** (Module 1) Implement a basic robotic application with microcontroller: code, wirings, read sensor input and control an actuator.
6. **[ ]** (Module 1) Design, fabricate, and assemble the 3-DOF Stewart platform mechanism.
7. **[ ]** (Module 1) Implement codes to read, filter, and map touch screen inputs to a 2D (X, Y) coordinate system.
8. **[ ]** (Module 1) Integrate and tune the dual PID controllers to achieve a stable 3-DOF balance.
9. **[ ]** (Milestone 2) Implement a PID control loop for accurate wheeled odometry and point turns.
10. **[ ]** (Milestone 2) Implement a "wall-following" algorithm.
11. **[ ]** (Milestone 2) Realize a "wall-following" robotic system.
12. **[ ]** (Milestone 2) Implement a "maze-solving" algorithm.
13. **[ ]** (Milestone 2) Realize a " maze-solving " robotic system.
14. **[ ]** (Milestone 3) Implement python scripts to set up Isaac Sim simulation.
15. **[ ]** (Milestone 3) Successfully train a balancing policy in NVIDIA Isaac Sim.
16. **[ ]** (Cross-Skill) Create a basic robotic simulation (in NVIDIA's Isaac Sim) of a robo-car or robo-arm by following tutorials and walkthroughs.

3. Reports

1. **[ ]** Analyze the 1-DOF system's response (e.g., oscillation, drift) and relate it to specific PID gains in a report.
2. **[ ]** (Module 2) Analyze and debug 3-DOF system failures, distinguishing between mechanical, sensor, and software issues in a report.
3. **[ ]** (Milestone 3) Analyze and document the drive train and control system (keep in lane) of your micromouse.
4. **[ ]** (Milestone 3) Analyze and document the performance of your wall-following algorithm.
5. **[ ]** (Milestone 3) Analyze and document the performance of your maze-solving algorithm.
6. **[ ]** (Milestone 4) Successfully train a robotic policy in NVIDIA Isaac Sim and describe the challenges of "Sim-to-Real" transfer.

4. Advanced skills

1. **[ ]** (Module 2) Derive and explain the geometric relationships for the 1-DOF system, successfully implement such system.
2. **[ ]** (Module 2) Write codes that *successfully* implements the IK, converting pitch and roll into three motor angles.
3. **[ ]** (Module 2) Model the 3-DOF balancing problem in the simulation.
4. **[ ]** (Milestone 3) Analyze and implement a maze-solving algorithm with simulation.
5. **[ ]** (Milestone 3) Implement a complete maze-solving robotic system.
6. **[ ]** (Milestone 3) Implement probability and statistics to model sensor noise and filter out erratic readings.

5. Cross-Disciplinary Skills

1. **[ ]** Document or introduce in clear, professional "slide decks" and deliver a timely and well-prepared presentation.
2. **[ ]** During teamwork, made a major contribution to the project.

3. **[ ]** Take leadership roles (such as project manager and coordinator, principle designer, or main dev) in a group project and coach other students to acquiring new skills.

Participation expectations, if applicable:

Participate actively in each class will get the most of the course and avoid spending extra time self-teaching. Attendance is tracked but not graded. I will follow up with student with excessive absences, but not direct penalty will be incurred. You do not need to seek permission to miss a class. *Is soloing and crunching possible?* Yes, it is. However, teamwork is encouraged and heavily valued for your final grade. If you know how to do it alone, do it with a teammate and show them how it's done.

Late work policy, if applicable:
You will start the semester with 5 tokens. You will spend 1 token to:

- Extend the deadline on any submissions by 36 hours
- Retry a second time (i.e., submission for a third time).

Deadline extension must be request before the original deadline. The policy will be adjusted based on the classroom feedback and general pacing.

Acceptable methods for submitting work: Microsoft PowerPoints/google slides will be the preferred submission format via Moodle submission portal. Hand writing, hand sketch, or multimedia format will be acceptable only if it's comprehensive to me, but won't earn an "Exemplary" grade. "Slide deck skill" will be essential for your future career, now it's a good time to practice!

Course management system /website, if applicable: Lecture slides and resources will be posted and maintained on Moodle and GitHub.

Information about Final Exam and course meetings through the end of the term: No final exam will be planned. Since each milestone will delivery of a functional robotic system, your final grates will reflect your personal growth, your effects in all topics, and how well you work within a team across the whole semester.

Any additional information or class policies on ChatGPT/AI/generative technology:

General rule-of-thumb: AI tools are roller skates, not crutches.

AI tools (like ChatGPT or Copilot) may be used as long as they support your learning and do not replace your independent thought or effort.
Citation rule: You must append an "AI Appendix" to any assignment where AI was used.

- Code: Wrap AI-generated blocks in comments:
  ```
  ### Begin AI Code ###
      AI code here
  ### End AI Code ###
  ```
- Images: Caption clearly with examples like [Image generated by XXX generator].
- Prompt Log: Include the specific prompts you used at the end of your submission as supplementary log.