Code Management and Generative AI Best Practices
A Collaborative Review
EE450 Military Robotic Applications

Cadet 1
United States Military Academy
West Point, NY
First.Last@westpoint.edu
Cadet 2
United States Military Academy
West Point, NY
First.Last@westpoint.edu

***Abstract*—This is a collection of best-practice proposals by the cadets of EE450 Military Robotic Applications. The cadets use this document to share their perspectives, experiences, and recommendations on the use of software IDEs (specifically, Visual Studio Code and the Arduino IDE), robotics project code structure, and the use of generative artificial intelligence in completing robotics projects. This is a "by-cadets-for-cadets" living record that both encourages reflection and guides future cadets in their robotics endeavors.**

I. Instructions to Contributors

This document is divided into four parts:

1) Getting Started with Arduino Programming
2) Code Management and Structure
3) Integrating Generative AI
4) Testimonials and Examples

You may contribute to any or all of these sections as much or as little as you like. It is critical that your contributions remain *high quality* and *easy to understand*. Do your best to place your inputs in the correct sections and subsections, but feel free to create your own sections or subsections if you think you need to. You must not, under any circumstances, provide direct answers to any of the course projects, mini-projects, quizzes, or other assignments — the purpose is to guide other cadets on their own problem-solving, not to solve the problem for them.

Contributions to this document must be made via a *pull request* to the appropriate GitHub repository. This first requires you to *fork* the project repository into a new *branch*. If you need guidance on how GitHub works, or how to submit a pull request, you can check the provided references. [1] [2]

II. Getting Started with Arduino Programming

A. Visual Studio Code

B. Arduino IDE

III. Code Management and Structure

A. Incorporating the Sense-Decide-Act Paradigm

B. Implementing States and Using State Diagrams

C. File Structure and Version Management — Avoid Drowning in Code

IV. Integrating Generative AI

A. Registering for Copilot Pro

B. Managing and Integrating Generated Code

C. Documentation and Citation of Generated Code

V. Testimonials and Examples

Feel free to add any other guidance, examples (good or bad!), or advice to other cadets here.

A. Advice from CDT X

If I had to give one piece of advice, it's to research the sensors I'm using and understand how they work, *then* build my state diagram and think about what I actually want the robot to do in terms of sensors and actuators, *then* think about how I want to prompt the generative AI. Starting by inputting the problem statement into Copilot never worked for me a single time — it always gave me something I didn't understand and had no idea how to fix, causing problems later.

References

[1] GitHub Docs. About branches. https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-branches, 2025. Accessed 2025-11-14.
[2] GitHub Docs. About pull requests. https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests, 2025. Accessed 2025-11-14.