Integrating Visual Studio Code into our Botball Development Workflow

Markus Weberndorfer
Linzer Technikum - HTL Paul-Hahn-Straße
Litec
A-4020 Linz
40146720210298@litec.ac.at

Manuel Zöttl
Linzer Technikum - HTL Paul-Hahn-Straße
Litec
A-4020 Linz
40146720210096@litec.ac.at

***Abstract*—Our team sought to improve our software development process for the ECER 2024 Botball challenge by integrating Visual Studio Code as our primary integrated development environment (IDE). The built-in KISS IDE provided limitations such as lagging performance, potential data loss from crashes, and a lack of modern developer tooling. By utilizing Visual Studio Code in conjunction with an SSH connection and a custom build script, we were able to leverage its robust feature set for writing, debugging and maintaining our robot control code, while still preserving the ability to compile and run programs on the Wombat robot controllers. This paper outlines the motivation behind adopting Visual Studio Code, the implementation details involved, the specific benefits realized, and discusses potential future improvements to further streamline this development workflow.**

I. Introduction

Applying effective software development tools and practices is crucial for maximizing productivity, especially on complex projects like those encountered in educational robotics competitions. While the provided KISS IDE offers a simple, self-contained development environment tailored specifically for the Botball kit, our team recognized several shortcomings that motivated us to explore alternative solutions as part of our preparations for ECER 2024.

The primary issues we faced with the KISS IDE included lackluster performance at times, particularly when working on larger files, lack of advanced editing capabilities like code completions, and most critically, the risk of losing work due to browser crashes with no way to recover unsaved changes. These limitations significantly hindered our coding workflow and ability to iterate quickly.

Given our team's familiarity with modern development tools like Visual Studio Code from other coding projects, we hypothesized that integrating a professional-grade IDE in place of KISS could help mitigate the issues above and generally elevate our skills by exposing us to industry tools used by countless developers worldwide.

II. State of the Art

Robotics-oriented programming environments have traditionally coupled relatively basic code editors with custom toolchains and libraries tailored for their specific hardware and software stack. While simplifying setup, this approach forgoes the wealth of advanced functionalities offered by professional developer tools.

In recent years, however, there has been a push in some domains to leverage powerful third-party code editors and IDEs that integrate well with robotic frameworks, rather than using the proprietary built-in alternatives. Prominent examples include web-based IDEs like Gitpod for working with Robot Operating System (ROS), as well as desktop IDEs like Visual Studio Code featuring comprehensive ROS support through extensions.

The advantages of adopting industry-standard tools in educational robotics contexts are multi-fold:

1. Students gain exposure to the same tools used by professional developers, better preparing them for future careers.
2. Projects can benefit from advanced IDE capabilities like intelligent code completions, inline debugging, rich extensions, and much more.
3. IDEs with a thriving developer community, regular updates, and cross-platform support avoid potential stagnation or discontinuation.

Given the active development of Visual Studio Code by Microsoft, its rich extension ecosystem, and deep Git integration, we selected it as the ideal professional IDE to integrate into our Botball workflow.

III. Concept / Design

Our main requirements were:

1. Use Visual Studio Code as the primary code editor
2. Preserve ability to compile and run code on the Wombat controllers
3. Leverage Git for version control directly within the IDE

To meet these, we decided to connect Visual Studio Code to the Wombat over SSH and use a custom script to handle compiling code and deploying/executing it on the robot.

IV. Implementation

A. *Setting up the Environment*

We installed Visual Studio Code and the Remote-SSH extension to enable editing files directly on the Wombat controllers over an SSH connection. Auto-completions were limited since VS Code lacks native Botball library support, but we explored possibilities to generate downloadable code intelligence data.

B. Compiling and Running

While we could edit files remotely via SSH, we still needed a way to trigger the existing Botball compiler toolchain to build our code into an executable binary, transfer it to the Wombat robot filesystem, and run it on the controller.

To accomplish this, we created a custom bash script that automated these steps:

1. Compile all .c source files together with required Botball libraries into an executable
2. Run the executable

This script was configured as a build task in Visual Studio Code, allowing us to compile and execute our code on the robot with a simple keyboard shortcut.

C. Version Control

Visual Studio Code provided integrated support for Git version control directly within the IDE interface. We initialized a Git repository, committed our code changes, and leveraged capabilities like branching, pushing/pulling, and merge conflict resolution. This streamlined collaboration and change tracking immensely.

V. Results / Conclusion

A. Benefits

- Rich code editing: Completions, navigation, advanced find/replace
- Debugging: Setting breakpoints, inspecting variables, step execution
- Git integration: All version control workflows in the IDE UI
- Extensions: Code linters, formatting utilities, custom themes, etc.
- Cross-platform: Consistent experience on Windows, Mac, Linux

While there was an initial setup investment, the productivity gains made it worthwhile. Using Visual Studio Code elevated our skills beyond the KISS IDE.

B. Future directions

- Tighter integration with Botball toolchain and API documentation
- Official support for Botball C libraries (code intelligence, etc.)

Overall, Visual Studio Code successfully modernized our development experience and better prepared us for real-world software practices.