**Development Environment Setup for Gutter Cleaning Robot Project**

**July 6, 2024**

**Decision Affecting:** Computation, the ability externals, accessing existing libraries.

**Overview:** To facilitate a consistent and controlled development environment for the Gutter Cleaning Robot Project, we are considering the use of Visual Studio Code (VS Code) with a development container. This approach aims to manage dependencies effectively and ensure compatibility across different development setups.

**Hardware Compatibility Considerations:**

* **ESP32**: While capable for many embedded applications, the ESP32 may be limited for running full ROS 2 due to resource constraints. Micro-ROS is a potential alternative for integrating ROS 2 functionalities in resource-constrained environments.
* **Raspberry Pi**: As a more resource-abundant platform, the Raspberry Pi is suitable for running full ROS 2, providing ample processing power for complex computations and connectivity options for various peripherals.
* **STM32**: The use of micro-ROS can enable ROS 2 compatibility, balancing between resource constraints and computational needs.

**Development Container:** Utilizing Docker with the VS Code Remote - Containers extension allows for the creation of a controlled development space that isolates the development environment from the host system. This setup supports:

* **Ease of Setup**: Simplifies the process of configuring development tools and libraries.
* **Consistency**: Ensures that all team members work within the same environment, avoiding the "works on my machine" problem.
* **Flexibility**: Allows for easy updates and changes to the development tools without affecting the host system.

**Configuration Steps:**

1. **Base Image Selection**: Depending on the chosen hardware and software approach, an appropriate Docker base image (e.g., Ubuntu with ROS 2 for comprehensive applications, or a lighter base for simpler tasks) is selected.
2. **Tools Installation**: Essential development tools such as PlatformIO, GCC for C/C++, and Python are installed within the container.
3. **Container Integration**: Ensures the container has necessary access to hardware interfaces, such as USB ports for device programming.

**Decision Rationale:** Choosing a development container is driven by the need for a reproducible and consistent development environment that can adapt to different hardware capabilities and software requirements. This setup not only facilitates the initial development phase but also supports scalability and adaptability to different computational platforms (ESP32, Raspberry Pi, STM32) depending on the project's evolving needs.

This approach ensures that the project is equipped to handle integration issues and supports external module access effectively, making it an ideal choice for managing the complex software and hardware interactions in the Gutter Cleaning Robot Project.