A Report on Issues Rising from Locally Stored Undocumented Code in the Previous Year and How it is Solved by This Year's Implementation

Table of Contents

Introduction & Overview

This article will discuss how Gyrodos G2.0's code was undocumented and locally stored, the issues leading to and caused by it, and how it is being improved in Gyrodos G2.2. G2.0's code had no documentation at all, and it only existed on one computer, which meant that changes in the code base are both impossible to track and to understand without the committer.

Problems Regarding G2.0's Code

G2.0's code was undocumented and stored locally. This was probably due to the need for development velocity and lack of communication. Development was rushed and the need for proper code documentation was overshadowed by the frequent changes in the code base.

Github was also not used since pull requests take time, and storing everything on one device is relatively efficient in terms of time. The problem is that this led to increasingly unmanageable code. With each iteration, the code gets increasingly unreadable, which makes documenting the code increasingly harder, making the developers less incentivised to care, and the vicious cycle continues.

In fact, taken from the PyCharm records of the 2024 G2.0, there were 434 lines of Python code for the Lucario Interface's onshore component (MacBook), 116 lines of C++ code for the Lucario Interface's interim component (Arduino Uno) and 449 lines of C++ code for the Gyrodos G2.0 ROV. Development of code was still barely manageable, however with the G2.2's numerous upgrades planned which include a much smarter thrust vectoring, autostabilization and faster communication features, even a base version that has no communication function has 1759 lines of code (Lucario 1.0.0), and we predict when we add in communication and functions, it will go up into the 5000s, making development by 1 person impossible. To add to that amount, this year we will have to make a Floater (Magikarp series), and that will require autonomous data-sharing and thus even more effort.

An Analysis on Code Documentation

Code documentation is essential in a code base because it allows developers to quickly understand what the code is doing and how to work with it. Though writing cleaner and more self-explanatory code can reduce the need of in-line documentation (i.e. comments), it does not provide a wider overview of what the code does. Without code documentation, it takes tremendous amounts of time and effort to understand every commit.

Clear and concise code documentation is required in collaboration projects, allowing new developers to understand the code base easier and faster.

Measures to be taken

We recommend all members on the Coding team use GitHub to establish good file management practice so as to manage the ever-increasing amount of code present related to Gyrodos's development, and that Coding should work hand-in-hand with the Mechanical Engineering department to know more about their functional updates and how they affect the craft's performance. It is paramount that inter-department integration be done, otherwise using GitHub will just induce extra bureaucratic inefficiency and our members will fail to reap its benefits.

References

https://swimm.io/learn/code-documentation/code-documentation-benefits-challenges-and-tips-for-success