# Writing Gazebo Plugins

**Kenzie King**

**Sprint 3 Documentation of Work**

## Objective of Research

The objective of this research was to learn how Gazebo plugins can be made. These plugins are what allow movement of the models visible in the Gazebo environment, thus it is very important that we have functional plugins as there are multiple models that will need to be controlled in the simulation.

## Findings of Research

The first notable finding of this research is that Gazebo control nodes cannot be written in Python like I had hoped. While ROS is compatible with Python, the Gazebo plugins are C++ only. This does not hinder the ROS compatibility at all, however it does require more programming time on my end as I am not as comfortable with C++ as I am with Python.

Another notable finding is that these plugins are for more than just model control. This means that everything from sensors, to the GUI, to the launch of the system itself can be controlled through plugins. This means that it is also a possibility for us to have a UI that is fully integrated with the environment as opposed to simply being a separate program window overtop the environment.

What I mainly wanted to focus on in this research was model plugins. These allow for control of the individual models such as the vehicles we will have in our simulation. They are very customizable and allow for controls and data stream of all joints, links, collisions and other physical properties.

It seems to be fairly trivial to create a simple plugin that allows velocity commands to be published to the model. Obviously, there will be many more complexities added into this than just a velocity command, but now it is known that the framing of these plugins is available and relatively simple to use.

Sensors will also need this same type of C++ plugin for their control as they seemingly will not automatically broadcast the data as I had thought. This will be investigated further and hopefully tested in the next sprint.

## Research Conclusion

In conclusion of this research topic, I have learned that these plugins must be C++ and seem to be more user friendly than I had thought for simple features. In the next sprint, I will be testing the creation of these plugins for their functionality in our use case.

I have also learned that plugins have more uses than I thought. Because of this, we now have an integrated GUI option as well as an added opportunity for world, view, and launch customizations.