State Machine Workshop

HYPED

October 8th, 2022

1 Introduction

This is an introductory workshop to the state machine module. It is based on the codebase from the previous year but simplified for the sake of this workshop. Please familiarize yourself with the code provided, refer to the diagram below and attempt every question. If you have any doubts, feel free to ask one of us. Good luck!

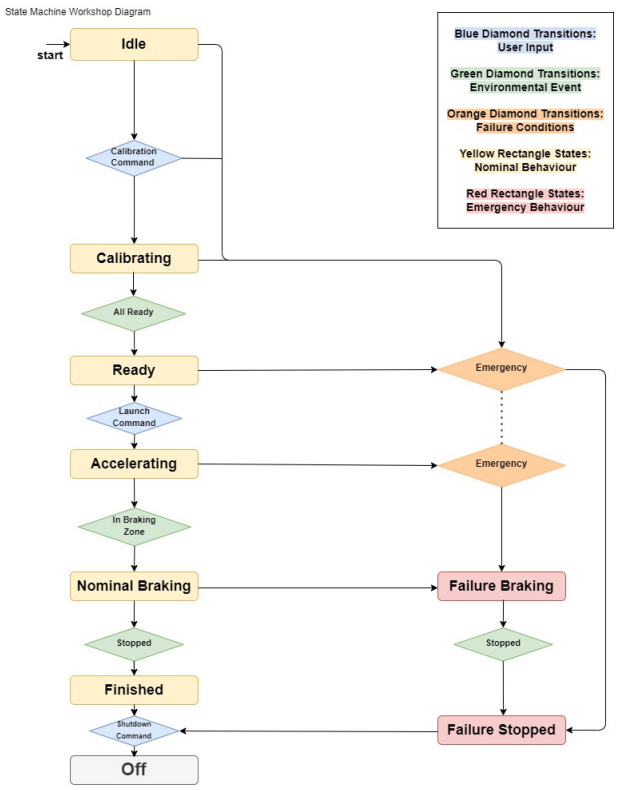

Figure 1: The State Machine Diagram

2 What is a State Machine?

In very simple terms, a state machine reads a set of inputs and changes to a different state based on those inputs. In each individual state, the system (in our case being the pod itself), behaves differently and waits to execute a transition once certain conditions are met. The system can only be in one state at a time (i.e it's deterministic).

If you are curious and would like to learn more, this is a good place to start. What we have is a deterministic finite state-machine, though there are non-deterministic ones too. Looking into the variants of the turing machine could send you down on an interesting state-machine rabbit hole!

3 Installing the Workshop Files Locally

To start with, head over here (* need to change link to the stm-workshop repo directly once its there *) and download the repo to your machine. Refer to the git workshop if you have any troubles with this part.

4 Implementing Transitions

After you have had a chance to look around the files (focus on the *state_machine* and *data* folder for now), head to *transitions.cpp*. Tasks 1-5 are labelled in the file and will give you the opportunity to implement transition methods (i.e methods that help take you from one state to another). Do refer to *data.hpp* and *transitions.hpp* to get a better idea if you get stuck.

5 Implementing States

Hopefully you got through implementing transitions just fine, though don't worry if not either - focus on familiarizing yourself with the code (its structure, style, etc) and you will be just fine!

Now you get to implement some states! Go over to *state.cpp*, and you will notice that the Calibrating, Accelerating and FailureStopped states are missing. Using the transition methods from before, create these states. Use the states already defined as a guide.

6 Testing

At the start, testing would not work since there are chunks of code missing. Once you have completed ***all*** challenges, run the below commands in the *stm_workshop* repo to test!

```
$ mkdir build
$ cd build
$ cmake ..
$ make -j
$ ctest
```

7 Few Things to Consider

Now that you have gone through and implemented parts of our pods state machine, you are officially ready to join the state machine team!

It is important to note that anyone part of this team will have to work closely with the rest of the software team and be actively involved in ensuring correct behaviour is maintained in each and every state of the pod (this also involves creating relevant tests to verify behaviour).

Some things to ask yourself after having completed this workshop would be: What was missing from the code?, What is the point of checking for emergency in every state?, Why are there two emergency states?, and What does state machine even achieve? If you are confused about any of these, or would like to discuss your ideas - feel free to approach us about it.

Hopefully you enjoyed this workshop and are eager to get started. We are definitely excited to welcome you on board.

Stay HYPED < 3.