## Implementation View

In this section some of the implementation details regarding the specific patterns will be shown and code snippets added to illustrate.

### Implementation details

**State pattern with Singleton states**

To implement the specific states om the state pattern a singleton was implemented for each state as seen on the example in Figure 3. The Instance() function ensures that only one instance of the state object will be created. The constructor of each state is declared protected as seen on Figure 1 so that only the Instance() function is able to call it.



Figure constructor of PowerOnSelfTest

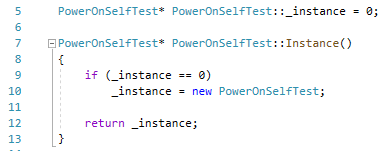


Figure Singleton pattern snippet from PowerOnSelfTest

The responsibility of changing state is handed to the specific states, therefore each states state function, in Figure 2 the Initialized function, the this pointer of EmbeddedSystemX is used by the states to call the changeState() function and set the next state.

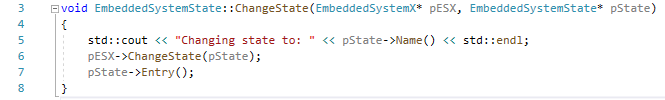


Figure changeState function of EmbeddedSystemState

**Command pattern**

The command pattern is implemented such that each function of the state machine is converted into a command object. Each object is then able to trigger the function. On Figure 5 the Execute() function of the configure command is shown. It is implemented as a sort of wrapper that trickers the states function and gives the EmbeddedSystemX pointer so that the state changes can happen as before with no change.

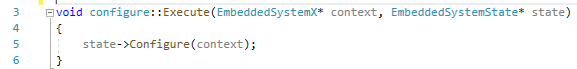


Figure 4 execute function for the configure command

The EmbeddedSystemX has a handleCommand() function that receives the command object from the client. The client which have the user interface is seen on Figure 7. Here the implementation is made in main. The user can create commands and as seen in Figure 7 these are passed to the EmbeddedSystemX.

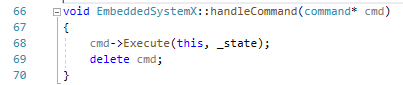


Figure 5 handleCommand function of EmbeddedSystemX

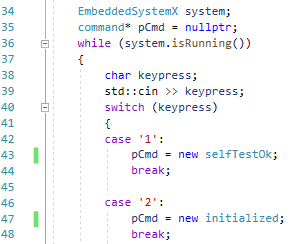


Figure UI code snippet, implemented in main



Figure EmbeddedSystemX trigger of handleCOmmand()