UML Overview of Project from a software engineering perspective.

* RocketDriver
  + Logs and compresses data sensor data to be sent over the CAN Bus (Data Frames)
  + Responds to Command Frames to tell ValueControl to actuate values which drive the rocket physically
  + Respond Command Frames by sending a StateReport Frame to the GUI Driver
* CommandGUI
  + Receives 8 logs sensor data in DataFrames from RocketDriver
  + Feed sensor data into Sensors, then takes interpreted data and displays it on the GUI
  + Receives UserInput and converts it into CommandFrames for directing the rocket
* Sensor
  + Can be declared as any type of sensor
  + Each sensor has a unique ID which exists as an enum in the SensorTable
  + Supports compression with BitChopper into MiniPacket
  + Supports hardware level measurements from AOC
  + Supports measurements from external sensors (data is passed in)
  + Can be created from MiniPacket and return an InterpretedSensorReading
* Value Control
  + Encapsulates the physical state of the rocket
  + Produces StateReport frames
  + Handles opening/closing solenoids
  + Converts high-level state commands to value configurations
* RocketLogger
  + Logs and compresses sensor data
* Data Frame
  + Contains sensor data from the rocket
* Command Frame
  + Contains commands which change(?) the local state RockerDriver and physically change the state of the packet
* State Report Frame
  + Contains information about state of ValveControl and potentially other important state info
* Abstract CAN Packet
  + Handles packing of MiniPackets into CAN Packet
  + Also contains info about sending node 8 what type of message this is
* MiniPacket
  + Contains an ID and payload data to be sent over CAN Bus
* CanBitBuffer
  + Treats CAN Packet as continuous bit buffer
* BitChopper
  + Used for bit level compression