Preamble

The Liquid Fuel Engine Test Stand (LFETS) Dashboard will be SCADA-like system for displaying the state of various actuators and sensors on the LFETS to provide user interface and manual control of the hardware. It will trigger and/or safe the system that automatically controls the ignition sequencing of the LFE. The system is required in order to prep, test, and fire the various hardware required for a LFE test and eventually a rocket launch.

Technical Requirements

1. The Dashboard must be usable without an internet connection, ie in the middle of the Oregon desert.
2. The project must be maintainable by students and future PSAS members. To this end, Python is the recommended language.
3. The interface must be graphical. PyQT has been recommended.
4. The software must receive data from, and send control signals to, a Marionette DAQ-based system.
    a. Future: the Marionette may be directly attached to a Raspberry Pi, in which case the Dashboard will receive data and send commands over a LAN.
5. The system must be 'easy to understand', so that anomalies are immediately visible, without interfering with user operations (ie no modal popups)
6. The system should be scriptable, for closed-loop operations
7. The system should include events and triggers, for closed-loop operations
8. The system model shall be data-driven for maintainability.
9. The system should be able to dynamically generate network maps. NetworkX is recommended. https://networkx.github.io/

Functional Requirements

1. The interface shall graphically display the EGSE system model components.
2. The interface shall display floating-point information received from sensors.
3. The interface shall display discrete state information for sensors attached to actuators, etc.
4. The interface may have the flexibility to display external inputs (ie EGSE state estimation data) for the related components.
5. The interface may display non-EGSE information pertaining to launch operations, such as temperature, time, and wind data.
6. The interface shall respond to mouse clicks on 'interactable' objects by sending state change commands.
7. The dashboard shall respect 'command lockout' times, once a command is sent, with the exception of emergency overrides.

8. The dashboard may include a global command lockout, limiting the interface to display-only.
9. The dashboard may expect a certain response from commanded devices, such as an acknowledgement, or a report of a state change, within a given time frame.
    a. Failure to detect a state change will result in displaying an anomalous state
10. User actions shall be logged.
11. Anomalies shall be logged.
    a. Anomalies include out-of-range data, data loss, “bad inputs” (unexpected states)
12. Time series data from sensors may be aligned and logged.

Components

Device
A Device represents a physical object that may be composed of zero or more Sensors, zero or one Actuator, and a Dashboard Element.
A Device may be in one of many discrete states, including (but not limited to) ‘nominal’, ‘warning’, ‘error’, ‘disabled’, ‘valve-open’, ‘valve-closed’, and ‘changing state’. (Is the device state the responsibility of the device itself? It’s definitely based on the inputs upon the Device, ie sensors and actuator feedback)

PhysicalLinkage
A visually defined connection between two devices. Currently meant for visual reasons only, to better match the Dashboard interface with the

DashboardElement
Catch-all superclass for the View container that encompasses both the high-level information for a Device, the text and iconographic metadata for the Device, and the state of the Device.

Sensor
A component of a Device that links to a physical device via serial port or logical socket, and provides either a discrete state input, or a floating point input. A Sensor has a unit, a typical range, a warning state description, and an error state description.
The Sensor will also have a ‘data loss’ configuration, and will be contain the information required to detect if a Sensor’s input signal has not been received in a given period of time.

SensorStateDescription
A combination of a range and a time duration for a sensor to change states. For example, a temperature probe may report an error if the temperature is greater than 105 degF for more than 0.1 seconds or for more than 3 samples.

SensorTrigger
A SensorTrigger is a combination of a Sensor, a StateDescription, and a snippet of Python code that will execute if a specified sensor enters the specified state.

SensorLog
A logical class for capturing data, primarily time-series data. It shall record data with its accurate timestamp, and shall be used to perform network analysis after the test stand is used (if desired). The SensorLog may be accessed by other objects, to provide time-aggregate measurements on the dashboard (rolling avg, min/max, etc)

Actuator
A component of a Device that may be controlled by the Dashboard user, or by Dashboard scripts.

Task Priorities

The project's rough order is as follows:

1. Build model describing the Devices, the connected Actuators and Sensors, and any physical linkages between the Devices.

Definitions

Interface is specifically the user interface, or the combination of the view and the controller.

Dashboard is the system as a whole, but may be seen as the Controller and the Model pattern.

System Model - the logical devices attached to the EGSE and connections between them.

EGSE - **Electrical Ground Support Equipment**

SCADA - **Supervisory Control and Data Acquisition**. **SCADA** generally refers to an industrial computer system that monitors and controls a process. In the case of the transmission and distribution elements of electrical utilities, **SCADA** will monitor substations, transformers and other electrical assets.

Time Alignment - the process of taking multiple inputs that arrive at different times, and recording the known value at a specified time interval. This should not be confused with a reading of the sensor, but is instead a reading of the dashboard.