**Software and API details**

Subject to Change, Look for Updates

Reference the LFRE github for visuals and code as a guide (<https://github.com/izukaike/LFRE>) , copy and paste if needed DO NOT HESITATE TO ASK QUESTIONS AND ASK FOR HELP:)

1. Display all 7 sensors data ((1)thrust, (5) pressure,(1) temperature, at the same time at an appropriate
   1. What framework you use for GUI is up to you I don’t know much about unity but you can try that out and see
2. Visual indicator of receiving feedback
   1. Doesn't have to look like this but the logic is the same

“ If bluetooth.recieve\_data() != NULL:

Turn on off little light on GUI

Else:

Turn light on GUI “

1. Display **refresh rate (how often data is refreshed/ calculating)**
2. Send a on signal to start control and reading data **so this is similar to the start plot button on the first gui but now start plot also includes being able to open and close valves**
3. Have a start button
4. A spark frequency enter on start up
   1. Ike during the com, start phase the use enters a number like 1 and that it sparks every 10ms or something like that
5. This just means you are reading 7 sensor’s data each ( >= 12 bits)

Considerations

* Calibration
* Speed/Refresh Rate
* How you want the data-> (ints, ascii, etc .)
* Read 4 pressure transducers
* Read 2 RTD’s/Thermocouples
* Read 1 Thrust sensor

1. Control 2 valves 2 buttons
   1. Look something like “engine.open\_valve()” and “engine.close\_valve()”
2. Receive status data. Not just read pressure but reading bits for needed feedback
   1. Like a message of “1010101010” and each bit is if a valve is open or an abort signal is triggered or aborts may just be handled on your side
3. The gui needs to have a visual indicator if it got feedback like how in the old GUI if the button was pressed or not the button color changed that and we need that
4. Receive debugging bits (don’t need to interpret just put it in a variable)
   1. Like “degug\_register = comm\_object.read()” and just don’t much after that
      1. I may do more about how I can add this.
5. Save data either automatically or if a button is pressed
   1. This just means that when the test is going on we can save the data
      1. What I did before was read in a value and add it to an array and plotted
      2. Example: “while true:

pressure\_trans\_data = comm\_obj.read()

Pt\_data\_array.append(pressure\_trans\_data)

\*\* after test\*\*\*

For index in range(len(Pt\_data\_array)):

Excel\_file.write(pt\_data\_array[index]) ”

* 1. ^^^You may come up with a better way to do this and not take a lot of time to process but that’s basically the strategy that I took for

1. Test Sequence
   1. Just a way to automate the physical button presses like open and close valves and be able to react to pressure changes
      1. Ex: While time < 5 seconds:

Engine.spark()

if pressure\_transducer\_data > COMBUSTION\_PRESSURE:

break out of loop and go to next phase

#phase 2

While time < 2 seconds:

If pressure\_transducer\_data > MAX\_PSI:

Engine.close\_valve()

Etc.

1. A button that tells if its real data HITL (Hardware-in-the-loop) fake data -> default is real data

|  |
| --- |
| COIL |

|  |
| --- |
| HITL |

|  |
| --- |
| VALVE |



API Details-> **TBC**

I plan on giving you a library or module that you can import and should help with development and interfacing

Ex: import LFRE\_HW as engine (engine is obj name)

\*your code\*

Def open\_valve\_button():

Engine.open\_valve() #this is provided by me

LFRE\_HW API (library)

My Notes:

<https://raspberrypi.stackexchange.com/questions/71216/cant-install-pybluez>

Actions:

Class members (variables you have access to)

Class functions (functions you have access to)

We can change these to specific values, but you can user place holders then use “replace all”

Constructor

HW \_Class\_Object obj( bluetooth info) -> TBC

Control Functions (Tx, Sending out)

Open\_valve\_1()

Open \_valve 2()

Open\_valve\_3()

Close \_valve\_1()

Close\_valve\_2()

Close\_valve\_3()

Start\_sparking() \*this should be used in the test sequence\*

Stop\_sparking() \*this should be used in the test sequence\*

Spark\_frequency(int \*time in ms\*)

Hitl\_testing()

Start\_plot()

Calibrate()

\*Some bluetooth set up stuff\* TBC

Data Function (Rx, receiving)

Note for Izuka: make struct to properly pass data in function

Read\_data\_frame() returns an array

* + [Start] | [Status] | [pt1] | [pt2] | [ pt3] | [pt4] | [thrust] | [debug] | [stop]

8 32 16 16 16 16 32 8 8

Helper Functions

D\_to\_V( dataframe df): data to values

* returns array of pt1 ducers psi : [ pt1, pt2, pt3, pt4, thrust]
* These values should be used for plotting on gui and saving data

D\_to\_S( dataframe df): data to status

* Returns array of status bits: [valve1, valve2, valve3, coil, abort1, abort2, abort3, abort 4]
* These values should be used for visual indicators of feedback

D\_to\_D(dataframe df): data to debug \*Don't worry about this function\*

* + Returns array of electronics status for debugging and hitl testing -> TBC

Variables