Critical Software: ECSS Hackathon Challenge

The focus of this challenge will be to create a working simulation of an in-home Smart Meter display. One important reason for the current move in the UK to fit every UK home with a smart energy meter is to provide a greater understanding of the way energy is used. In-home displays allow anyone to see exactly how much energy is being used at a given moment in time, as well running totals for amount and cost of the energy being used. By allowing people to understand which of their appliances use the most energy, and to see what using those appliances actually cost to run, people will naturally reduce the amount they use these appliances.

The aim of this challenge is to build a simulation of an in-home display that shows exactly which appliances are running in a house at any given time. In order to simulate a house using energy over time, this should run as a time-based simulation. The application you build should start at a fixed point in time, say the current time and date for example, and then count forward in hours from that time. Each hour that passes, any appliance in the house which is switched on will contribute towards a running total of the amount of energy used. The amount of energy each appliance uses will be set out below, measured in Kilowatts. The running total of energy used should be measured in Kilowatt Hours. A Kilowatt hour is a measurement of energy that is commonly used by energy companies to bill households for how much energy they use. It represents the amount of energy in Kilowatts that an appliance will use if run continuously for one hour.

In order to complete this challenge, you will be creating a Graphical User Interface (GUI) application. This kind of application makes it easier and more intuitive for users to interact with. The GUI you will be creating will display a floor plan of a house, onto which you will place icons that represent different appliances. Some example images for both appliances and floor plans will be provided, but feel free to use your own! Each appliance must include a button or toggle, so it can be turned on or off.

You are free to complete the challenge using any programming language and framework you prefer. One commonly used language for Hackathons is Python, as it is straightforward, easy to get started with, and features support for a large amount of 3rd party libraries and modules. You can see the documentation for Python at <https://docs.python.org/3/>. For the GUI part of the challenge, one possible library to use is TKinter - <https://docs.python.org/3/library/tk.html>. TKinter can be used in a normal Python script by simply importing it at the top of the script.

There are some examples of floor plans available, as well as some example icons to use. There may not be an icon for each type of appliance, so feel free to create or download your own images to use instead.

Challenge Specification

In order to complete the challenge brief, you must build a software system that meets the following criteria:

* Upon being run/launched, a GUI window is displayed to the user.
  + The background to this window should be an image representing the floor plan of a house.
  + The window should contain at least three appliances that use electricity, and at least two that use gas.
  + Each appliance should contain a label or button indicating whether it is turned on or off, and allowing this to be changed.
  + Each appliance uses either electricity or gas, and contributes towards the relevant total when calculations are made.
  + Appliance positions are controlled by the script of the application.
  + Labels showing the running combined total of gas and electricity used, and running combined cost of the energy for both of these. The unit for total energy for both electricity and gas is Kilowatt Hour (kWh), and for cost is GBP (£).
  + A label showing the current time and date, from the moment the application launched.
* Once launched, the application will begin a timer, simulating the passage of time within the house.
  + Each hour that passes in the simulation, the energy used by each running appliance in the previous hour is calculated and added to the running totals of energy and cost.
  + Appliances that are switched off before the end of the current interval don’t contribute to the energy usage.
* The amount of energy used by each appliance is specified in Kilowatts, and is given in the table after this specification. These values should be hard-coded into the script of the application.
* The amount of energy used during each interval should be calculated by adding together all ‘on’ appliances, then adding this to the running total. Different
* The cost of the energy used per interval should be calculated by taking the amount of each fuel used, and multiplying by the rate for that fuel. Rates for fuels are:
  + Gas: 8.706p per kWh
  + Electricity: 20.156p per kWh

Appliance usage rates:

|  |  |  |
| --- | --- | --- |
| **Appliance Name** | **Fuel type** | **Usage rate (kW)** |
| Refrigerator | Electricity | 5 |
| Kettle | Electricity | 100 |
| Oven | Gas | 3 |
| Lamp | Electricity | 0.5 |
| Boiler | Gas | 10 |
| Television | Electricity | 25 |
| Games Console | Electricity | 30 |
| Cooking Hob | Gas | 6 |
| PC | Electricity | 60 |
| Stereo System | Electricity | 15 |
| Radiator | Gas | 30 |

Extension Tasks

If your team manages to complete the above specification, there are a number of suggestions for ways the application could be improved. These can be done in any combination and any order.

* Instead of using buttons to toggle each appliance on and off, modify the icon used to display each appliance, so that clicking an appliance toggles between an ‘on’ and an ‘off’ icon. One could be darker than the other, or have another way to distinguish which represents which state.
* Add the ability to save and load data about the state of the simulation. The current usages of each type of fuel, as well as the total cost for each, should be saved in some kind of file, possibly a .CSV file. The same file can be loaded again by the user, which will reset the simulation to the state when that file was saved.
* Make the other built-in constants of the system configurable. Add the ability to load a file specifying the energy rating of each appliance, or the cost of each type of fuel.
* Add a second window to the GUI, containing all the data labels and any buttons added for saving/loading the state of the simulation. This window should update based on the appliances in the main window, and should be able to control all the functionality of the application.