Samsung SyncMaster Monitor analysis

Francesca Madeddu
803623@swansea.ac.uk

January 31, 2015

1 Introduction

The interactive system chosen for the analysis is a Samsung SyncMaster Monitor 25". Since the menu is very complex and includes a big amount of functionalities, only a small subset of options has been selected. As shown in the Figure 1 the user can interact with the system through five different buttons located on the bottom right of the device.

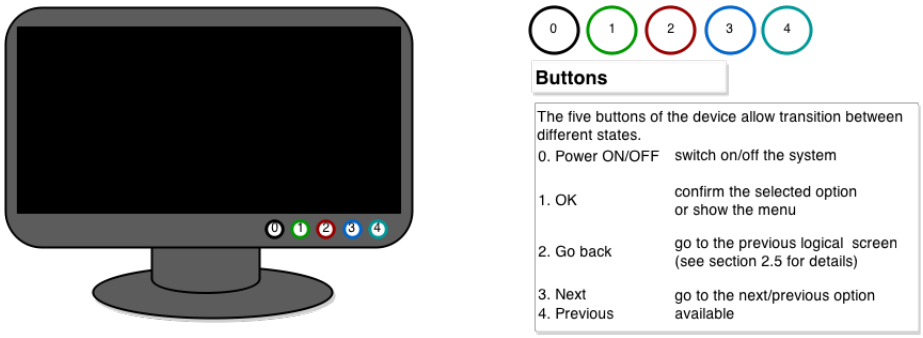

Figure 1: A sketch of the Samsung SyncMaster Monitor

The menu allows the user to navigate through the different states of the system: he can change some basic functionalities of the display, choosing within the following different options:

- brightness: can be set as 'custom' or 'standard';
- contrast: can be set as 'custom' or 'standard'.

For each possible combination[1] there is therefore a mode.

Choosing to omit most of the functionalities of the system apparently changes the nature of the system itself: however the choice is justified with the fact that focusing on a limited number of aspects allows to perform a critical analysis that keeps the "maths simple". In fact, when a system is described as a graph, its complexity can increase very sharply because of all the possible interactions the user can perform on the system.

The analysis will be performed using theorems from the graph theory, usability guidelines and the Press On Framework[4]. This approach is useful because it brings mathematics tools into the HCI world making it possible to evaluate a system in a formal and systematic way.

[1]There are two possible choices for each option, and two options are available so there are 2*2 possible configurations for the system.

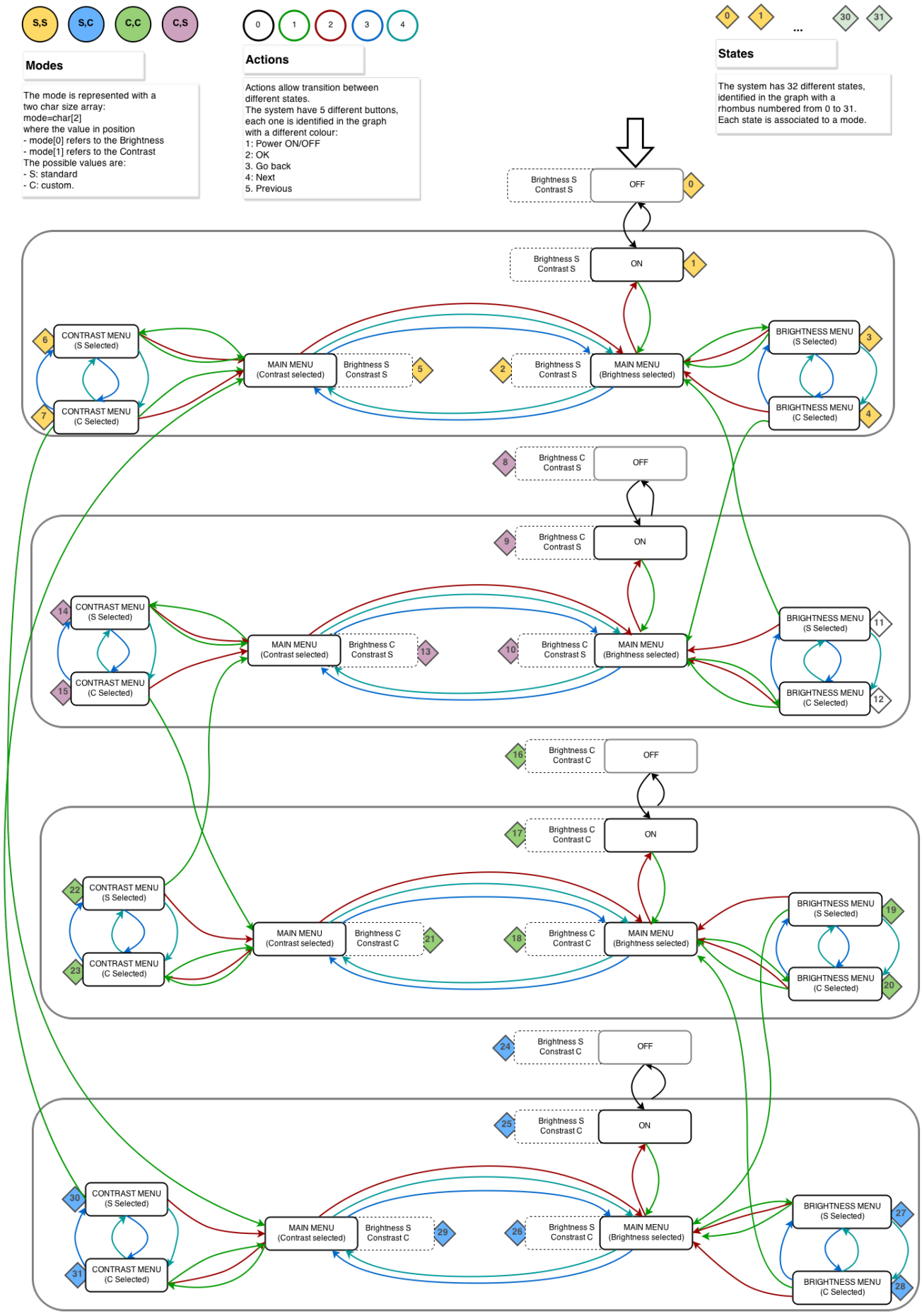

Figure 2: Graph of all possible states and actions

2 System Analysis

2.1 States and modes

Analyzing the graph makes it possible to highlight some peculiar characteristics of the device. It has 32 *states* that can be grouped in 4 different *modes* (Figure 3): in fact it has a memory, and each time it is turned on it recovers from the last saved configuration. Each mode corresponds to certain sets of states[4]: for example if the user is in a CC state he cannot see some screens of the menu of the SS mode[2]

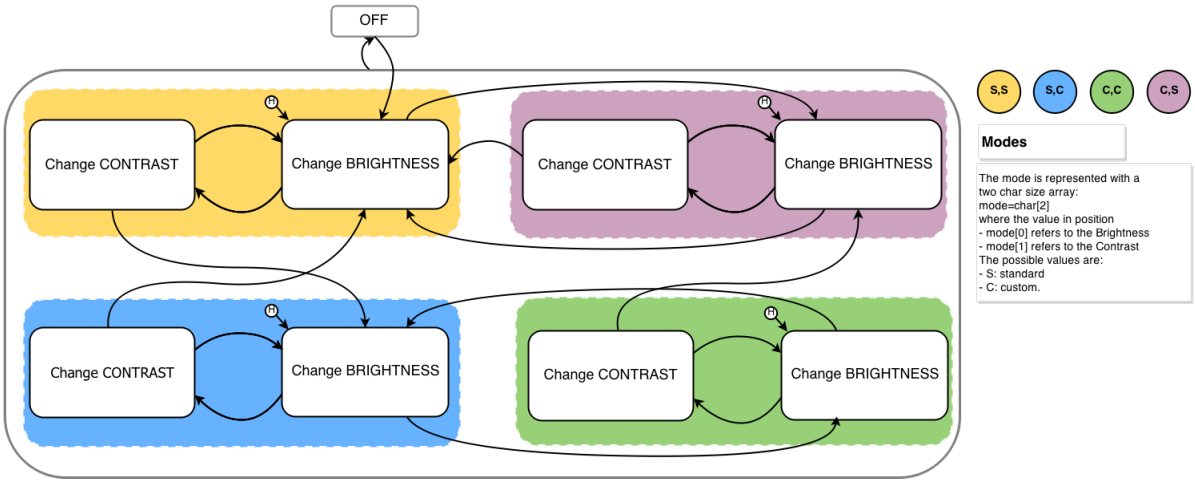

Figure 3: Statechart of the system

2.2 Indicators

As seen before the system has 32 different states: indeed it would require $\log_2 32 = 5$ indicators to distinguish between all of them. The device has actually only one physical indicator: a green led close to the power button that displays if the device is in a ON/OFF state. This indicator is very important as it lets the user to discern between two different states which would result into the same feedback:

- a switched-off monitor;
- a switched-on monitor with a black background.

A distinction within the actual possible states of the system and the modes had been made: 32 are the possible states, 4 are the modes in which the device can be set and that actually correspond to the user goal. In fact the user needs to know and change modes rather than states: the navigation between different states is just the way the user has to reach his purpose.

Therefore it would appear reasonable for the device not to have five different indicators. However it means the user is forced each time to navigate almost the entire system in order to know in which configuration the device is and which options are available. In fact it is an impossible job (maybe an expert user could) to understand in which mode the system is only looking at the light emitted from the monitor. The user hence cannot use his *deduction* profitably.

A possible solution could be developing an alternative version of the menu where all information is present at the same time so that the user can immediately catch the current configuration.

[2]CC means Brightness = custom, Contrast = custom; SS means Brightness =standard, Contrast = standard. Look at the Figure 3 for a deeper explanation of the mode syntax.

Figure 4: Two alternative designs for the menu

Using a design like the one shown in the Figure 4 would decrease the *cost of the worst case* (when the user wants to go from one configuration to the opposite one, for example from CC to SS) from 11 to 4.

This kind of solution follows the basic usability principle which says that it is crucial to help the user *memory* [3] offering a visual reference to the all possible actions. However it would collide with the *closure principle*[3] according to which users usually prefer small tasks and feel confused and overwhelmed when too many options are directly available.

2.3 Connectivity and usability

Hinges and *bridges* are crucial part of a graph, as they include critical information: when deleted the graph has been disconnected. Translated into the world of a interaction system it means the user hasn't any way to reach a specific functionality of the system any more.

Hinges have to be clear because if the user is not aware of their existence then the navigation through the entire system is limited. As common the system has the ON/OFF hinge: but there is also another hinge corresponding to the main menu of each mode: in fact states 1, 8, 16, 24 are hinges (see Figure 5). The connectivity[4], not considering the ON/OFF hinge, is then equal to 1 and it does not fit any connectivity requirement as the connectivity should always be equal or greater than 2. In fact, without pressing the 'OK button' the user will never reach the main menu.

The graph includes some *cliques*[5], corresponding to the *main menu* and the *brightness/contrast sub-menus* of each mode (ie. [2,3,4][5,6,7] are cliques for the first mode). Inside a clique is it possible to univocally associate an action to a button. In fact within the quoted cliques buttons have a unique meaning:

- ON/OFF: switch ON/OFF the device;
- OK button: confirm a selection;
- Previous/Next button: go to the Previous/Next option;
- Go back: return to the previous logic screen.

This is no longer possible when it is needed to navigate between different cliques: in that case, for example, the 'OK button' is used with multiple purposes:

- to confirm a selection;
- to show the main menu.

A more interesting information comes from the investigation of the independent set[6] since it gives to the designer the difficulty of the user to do something. There has to be kept in mind that even if is not true that if the size of independent set is small the user will do his job easily, it is definitively true that if it is high the job will be hard. The size of the independent set of the system is equal to 12 (Figure 5).

[3]From the Jakob Nielsen website:[2] "(..) *minimize the user's memory load by making objects, actions, and options visible*".
[4]Defined as the smallest number of vertices have to be deleted to disconnect the graph.
[5]A clique can be defined as sub-graph that is complete.
[6]Indipendent set can be seen as the shortest path getting from anywhere to anywhere that takes at least two steps.

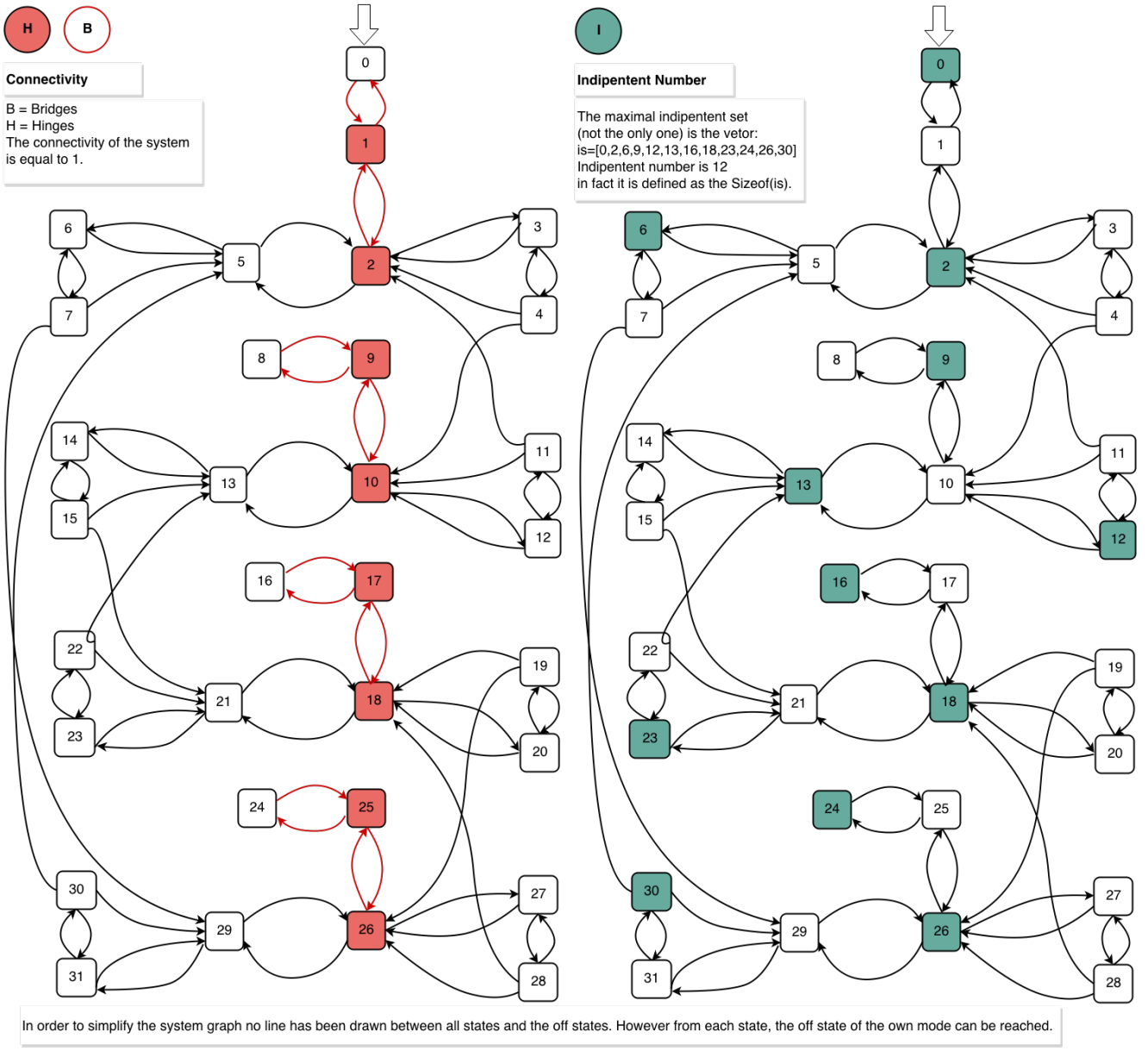

Figure 5: Hinges and bridges and Indipendent set

2.4 Coherence

The designer has adopted a *conceptual model* where to change the mode to another the 'OK button' has to be pressed. This choice seems to be justified by the desire of keeping the system coherent: in fact the idea of *confirmation* is always associated to the 'OK button'. But, because the 'OK button' takes the user back to the previous logical screen, this means that the system is not showing the feedback properly: in fact it does not give any chance to the user to change or amend his choice in a 1-step action. Showing the feedback when the user selects the option rather than wait for the 'OK button' to be pressed, probably would undermine the coherence of the system but would definitively improve the error prevention[7].

Another example where respecting the coherence doesn't seem the best solution regards the connectivity of the system. As said the system is poorly connected as its connectivity is equal to 1: if the user does not press the 'OK button' from the state "ON" he will never discover what the system can do. A possible solution to overtake the issue, would be showing the main menu to the user independently from which button has been pressed. On one hand this could affect the coherence of the system (different buttons are been used for the same action), on the other hand, the importance of making hinge visible could justify the design choice.

[7]From the Jakob Nielsen website:[2]"(..) *a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action*".

2.5 Error recovery

The system is tolerant to *overrun errors* [8], in fact the the average cost is equal to 0.83 while the worst case scenario is 2. However the situation changes in case of *single-press errors* [9]: while the cost for the worst case seems acceptable even if equal to 6 (in fact it corresponds to the case the user accidentally presses the 'ON/OFF button') the percentage of errors that can be undone directly is too high (54.38%). This happens because the system has a 'Go back button', but its job is not to *undo* last action (that is to go back to the last state); instead when the 'Go back button' is pressed, the workflow goes back to the previous logical screen. The navigation through the menu is organized in two logical levels of depth (Figure 6):

- The *main menu* (Level 1): where is possible to select brightness or contrast;
- The *brightness* or the *contrast submenus* (Level 2): where is possible to change the value from custom to standard or vice versa.

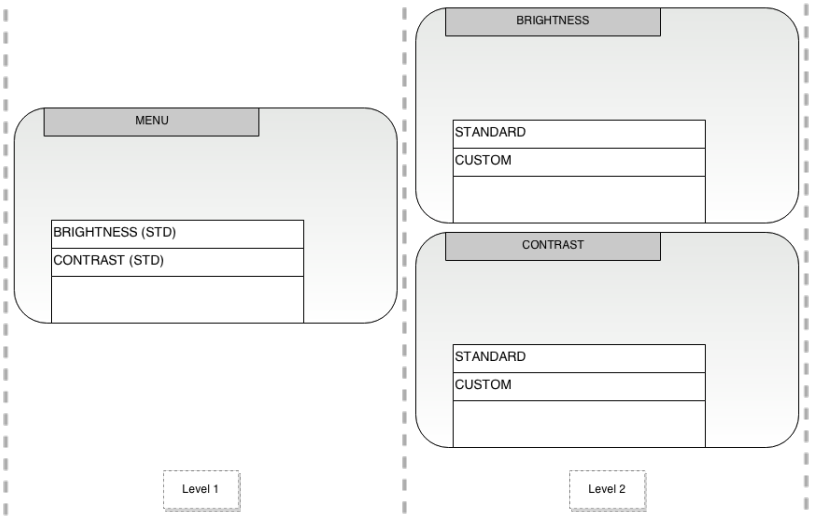

Figure 6: Different logic levels of the menu

Whenever the 'Go back button' is pressed from a *Level n,* it goes in a state of a *Level(n-1)* instead of going to the previous state.

2.6 Costs

So far in the analysis of the system no distinction has been made between edges: actions have been treated equally. However a weight can be assigned to each edge in order to give a cost to a specific action. According to the Keystroke Level Model[1] the system has been evaluated towards a specific goal: change the mode from CC to CS [10]. It allows to retrieve some information about the time that is required to accomplish a specific task. To reach the goal the user has only one method, described in Figure 7. To be noted that the time for each operator is the one defined by the Keystroke Model [11] and that the reaction time has been considered as equal to zero, since the device has an immediate response.

[8] An overrun error happens when the correct button is accidentally pressed twice instead of once.
[9] Single-press error happen when a button is pressed in a random way.
[10] Change the mode from CC to CS means to change the value of the contrast from custom to standard.
[11] The KTM defines the keystroke or button press to 0.2s, the pointing the target to 1.1s and the mental preparation to 1.35[1].

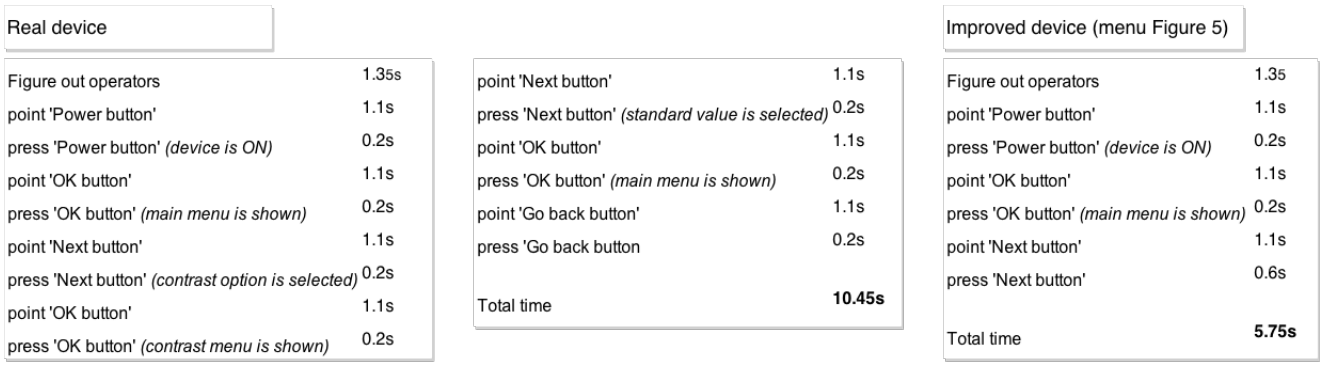

Figure 7: Time needed to go from a CC mode to a CS mode

According to results, the user needs a considerable time to reach his goal. A different implementation of the menu (as shown in Figure 4) would decrease the requested time to 10.45s to 5.75.

3 Extending the framework

A Json object [12] has been created in order to analyze the system using the framework . A new variable has been added to the Json object: *stateImages*. It is an array of strings and has to be the same size of the *states* variable. It contains the path to each image associated to each state. It allows to dynamically change images while interacting with the system.

To support the new variable few new lines have been added to the `makeFrom` function and to the `displayState`.

```
function makeForm(d){
  ...
  docwrite("<tr>
    <td align=right></td>
    <td align=center><img src=\""+d.stateImages[d.state]+"\"
    name=images height=200></td>
    </tr>");
  ...
function displayState(d) {
   ...
   document.forms["info"]["images"].src = d.stateImages[d.state];
}
```

4 Limits and conclusion

The analyzed device seems to have problems related to the error recovery that could be in fact optimized and to the connectivity since some states should be made more visible; furthermore the time needed to accomplish even an easy task appears too high. However when looking at the results it has to be kept in mind that the fact the device has been simplified could change the meaning of data. For example, the 'Next button' seems superfluous when no more than two options are available, however in the real device more than two options are actually present.

The limit of this kind of approach in fact is that even if the graph gives interesting and formal ways to analyze the system, it requires a complex, long and deep analysis in order to retrieve meaningful and trusted results and a critical mind to deeply understand how abstraction can impact on results.

[12] See Appendix.

Appendix

Json Object


```
{
author: "Francesca Madeddu",
notes: "External Display",
modelType: "Samsung SyncMaster",
buttons: ["ON/OFF","OK","<--",">>","<<"],
stateNames:
        ["OFF(s,s)","ON(s,s)","MAIN-MENU_bright(s,s)","BRIGHT-MENU_std(s,s)",
        "BRIGHT-MENU_cstm(s,s)","MAIN-MENU_contr(s,s)","CONTR-MENU_std(s,s)",
        "CONTR-MENU_cstm(s,s)","OFF(s,c)","ON(s,c)","MAIN-MENU_bright(s,c)",
        "BRIGHT-MENU_std(s,c)","BRIGHT-MENU_cstm(s,c)","MAIN-MENU_contr(s,c)",
        "CONTR-MENU_std(s,c)","CONTR-MENU_cstm(s,c)","OFF(c,c)","ON(c,c)",
        "MAIN-MENU_bright(c,c)","BRIGHT-MENU_std(c,c)","BRIGHT-MENU_cstm(c,c)",
        "MAIN-MENU_contr(c,c)","CONTR-MENU_std(c,c)","CONTR-MENU_cstm(c,c)",
        "OFF(c,s)","ON(c,s)","MAIN-MENU_bright(c,s)","BRIGHT-MENU_std(c,s)",
        "BRIGHT-MENU_cstm(c,s)","MAIN-MENU_contr(c,s)","CONTR-MENU_std(c,s)",
        "CONTR-MENU_cstm(c,s)" ],
fsm: [[1,0,0,0,0],[0,2,1,1,1],[0,3,1,5,5],[0,2,2,4,4],[0,26,2,3,3],[0,6,1,2,3],
        [0,5,5,7,7],[0,13,5,6,6],[9,8,8,8,8],[8,10,9,9,9],[8,11,9,13,13],
        [8,10,10,12,12],[8,18,10,11,11],[8,15,9,10,10],[8,5,13,15,15],[8,13,13,14,14],
        [17,16,16,16,16],[16,18,17,17,17],[16,20,17,21,21],[16,10,18,20,20],
        [16,18,18,19,19],[16,23,17,18,18],[16,29,21,23,23],[16,21,21,22,22],
        [25,24,24,24,24],[24,26,25,17,17],[24,28,25,29,29],[24,2,26,28,28],
        [24,26,26,27,27],[24,30,25,26,26],[24,29,29,31,31],[24,21,29,30,30],],
startState: 0,
state: 0,
manual: ["Device is OFF","Device is ON","MENU (Brightness)","BRIGHTNESS-MENU (standard)",
        "BRIGHTNESS-MENU (custom)","MAIN-MENU (Constrast)","CONTRAST-MENU (standard)",
        "CONTRAST-MENU (custom)", "Device is OFF","Device is ON","MENU (Brightness)",
        "BRIGHTNESS-MENU (standard)","BRIGHTNESS-MENU (custom)","MAIN-MENU (Constrast)",
        "CONTRAST-MENU (standard)","CONTRAST-MENU (custom)","Device is OFF","Device is ON",
        "MENU (Brightness)","BRIGHTNESS-MENU (standard)","BRIGHTNESS-MENU (custom)",
        "MAIN-MENU (Constrast)","CONTRAST-MENU (standard)","CONTRAST-MENU (custom)",
        "Device is OFF","Device is ON","MENU (Brightness)","BRIGHTNESS-MENU (standard)",
        "BRIGHTNESS-MENU (custom)","MAIN-MENU (Constrast)","CONTRAST-MENU (standard)",
        "CONTRAST-MENU (custom)"],
action: ["Press","Pressed","Pressing"],
errors: "never",
graphics: "",
stateImages:
        ["stateImages/0.png","stateImages/1.png","stateImages/2.png",
        "stateImages/3.png","stateImages/4.png","stateImages/5.png",
        "stateImages/6.png","stateImages/7.png","stateImages/8.png",
        "stateImages/9.png","stateImages/10.png","stateImages/11.png",
        "stateImages/12.png","stateImages/13.png","stateImages/14.png",
        "stateImages/15.png","stateImages/16.png","stateImages/17.png",
        "stateImages/18.png","stateImages/19.png","stateImages/20.png",
        "stateImages/21.png","stateImages/22.png","stateImages/23.png",
        "stateImages/24.png","stateImages/25.png","stateImages/26.png",
        "stateImages/27.png","stateImages/28.png","stateImages/29.png",
        "stateImages/30.png","stateImages/31.png"],
}
```

The Dot Graph

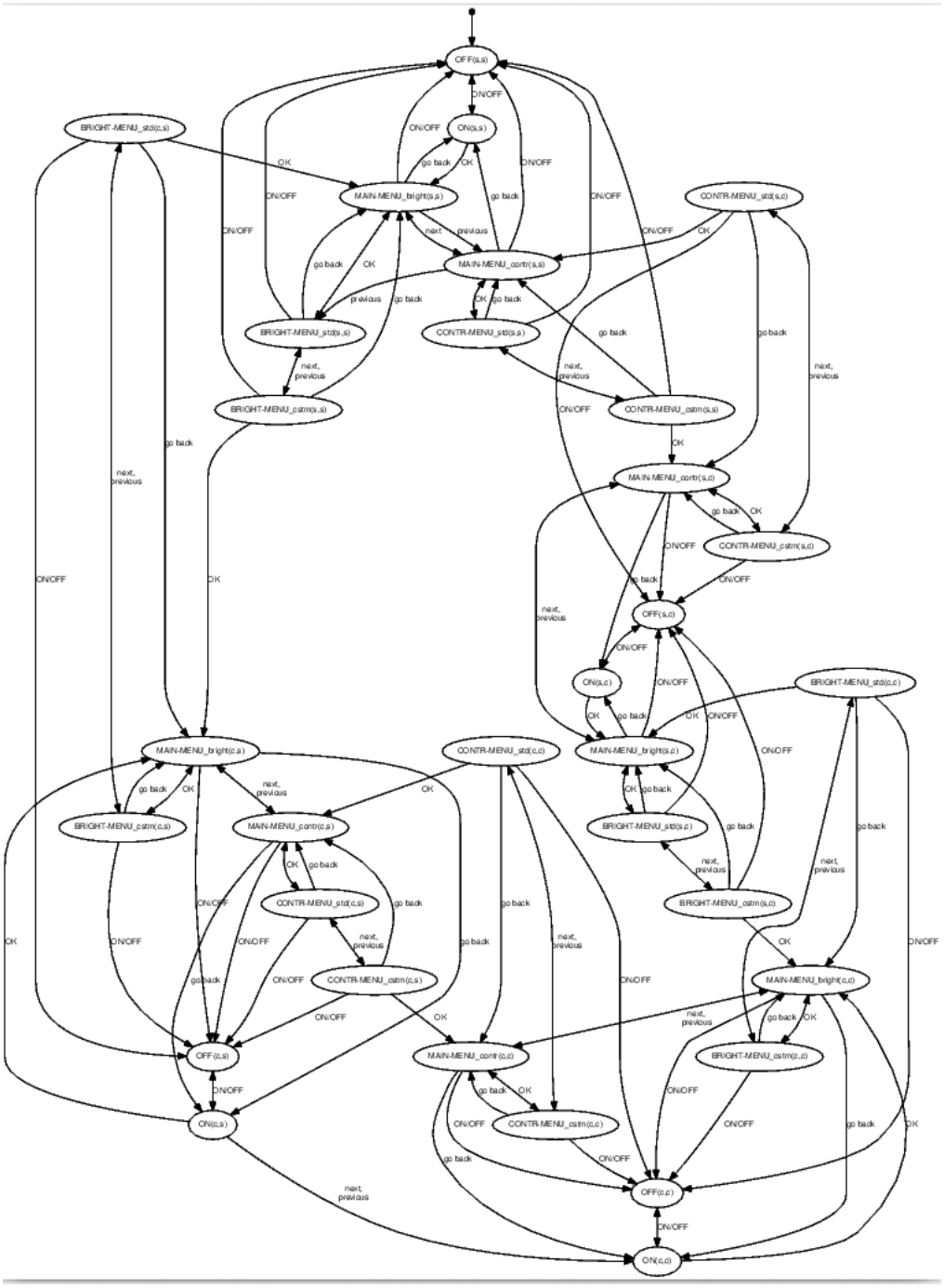

Figure 8: The Dot graph of the system

External file

The complete code of the extended framework and the images corresponding to each state can be downloaded @ Press On Framework extended

References

[1] Stuart K Card, Thomas P Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, 1980.

[2] Jakob Nielsen. 10 Usability Heuristics for User Interface Design, 1995. http://www.nngroup.com/articles/ten-usability-heuristics/.

[3] Ben Shneiderman. Human factors experiments in designing interactive systems. *Computer*, 12(12):9–19, 1979.

[4] Harold Thimbleby. *Press On: Principles of Interaction Programming.* Mitt Press, 2010.