Manual

Intro of the application

The application represents the printer itself, which is going to control the lights. We have three lights: the first light is used to keep track of the paper, the second represents the cyan ink, and the last one is the magenta ink.

The application simulates three printers that are consuming materials of various kinds: paper, oil, magenta, cyan, key and yellow ink. When a printer consumable reaches 0% the printer is left in an idle state. It must be reset to in order for the printer to continue working.

We will use this application to simulate how the printer is going to control the lights, with which we will alert the workers that there is a consumable near depletion. When he receives the light and color of the light he will know instantly what action to take.

Application Manual

*Starting the application and navigation*

*IPhone*

The user must install the application. In order the user to start the app, he/she must click on the icon. Then a table view will be visible with 3 printers. All in active state, as noticed from the table. Every printer has a unique name. The user then must choose the wanted from him printer. After the user chooses a printer, a detailed view will appear with the consumable of the printer. To go back to the table, with the printers, the user must click on the top of the navigation.

*IPad*

If the screen is bigger, like an iPad, the table and the printer details will be displayed both on the screen. Making it more clear to use and keep track of the printers.

*Resetting the printer consumable*

This can be done by the plus sign on the table view which is going to reset all of the values of all of the printers to 100%. If it is required to reset only one of the machines, this can simply be done by clicking the reset button on the details of the printer. Additionally, by clicking the label with the percentage we can reset the value to 100 of that current consumable.

*Using lamps with different colors to alert of an occurrencexs*

The lamps are being controlled by the first printer. The first lamp represents the paper percentage. It starts green and fades slowly through the spectrum of colors until it reaches red, of course the percentage is connected to the color of the lamp. The second lamp represents the cyan ink. The lamp remains off at the begging, but when it reaches a critical point (around 25%), it lights up. After that, around 15% the lamp starts blinking to notify that the critical section has passed to the danger point. This indicate that action must be taken, otherwise the printer may move to idle state and stop functioning when the consumable becomes 0%. For the third lamp the same process applies, but the lamp color is magenta.

Advanced Manual

We don't use any APIs for the reason that the APIs which we require are written on Swift 3. However this application is multithreaded and we use a lot of timers to control the printer consumption. The consumption is different for every printer, since since we use a separate timer for every printer. Which subtracts a random value from the printers consumable percentage. In order to make the application work with the lights we send JSON strings to the bridge to control the lamps. By controlling I mean turning them off and on, switch their color, make them fade and additionally make them blink. The blinking part does not exist with a simple JSON string command, so we create an artificial blinking by simply by running a repeat while loop on a new thread. Where we create the send on and off request with a sleep in between the two requests. The fading part of the lamp, again as the blinking, does not have a command in JSON. So I made a mathematical calculation to get the current percentage of the printer consumable and I multiply it by the difference of the percentage of the value of the hue (color). For example green is 25000 in hue value and read is 1000. So 25000 is equal 100% and 1000 is equal to 0%. The percentage difference is 250%. I multiply the printer’s current percentage with 250. This is how the printer fade from green to red. By passing JSON every time the printer change it’s percentage of the first lamp. This is done in the timers of printer one.

To get the changes from the details view I have use the delegate pattern by using protocols.

For the GUI we decide to make it look more like a printer terminal so we choose the splits view which will be displayed on a iPad to give it a more authentic view and show the power of the splits view.

In order to get access to the bridge, we have to create a username which is automatically generated through the Philips debugger on the internet browser.