# Software

## Overall information about the code

Our first idea was to create a library for every room, door and control panel. The library should implement *“Init()”* and *“Update()”* functions which will be called in main function. In addition to those them we also implemented some functions which control the state of the room directly, for example they turn the light on and off. This approach made code more maintainable and readable, therefore we used it in our final build.

Main function photo

We had to come up with a solution to how buttons should operate in rooms where there is possibility to change brightness or color of the light. After few discussions we sticked with idea last action takes effect. This means that if brightness was set to half of its potential by clicking the button the light is turned off (light is set to maximum brightness after clicking button, only when brightness is zero), even though the potentiometer’s value stays unchanged. This solution turned out to be very handy when we started working on control panel.

When a button is pressed the change from not being pressed and being pressed isn’t immediate, there are always some fluctuations which are result of mechanical construction of a button (some reference to this topic). This can be solved by connecting capacitor to a button pin and ground. But, we didn’t do that and since PCB was already printed we decided to solve this issue by adding delay every time a button is pressed, this resulted in functions for every button which check status of a button and if it is pressed they start 100 ms delay.

Add photo of such function

## Room 1

Room 1 has one button for turning the LED on and off and one potentiometer for controlling the brightness of the LED. We implemented functions for controlling and retrieving information about the LED’s status and brightness.

Photo of room\_1.h

In *“Init()”* we initialized ADC and PWM using Alin’s libraries(reference to his libraries), and also set button pin as input with pull up resistor.

*“Update()”* function checks if potentiometer values changed since last measurement, if yes update LED’s intensity accordingly. Then it checks if button is pressed and wasn’t pressed previously, if this statement is true then the status of LED is inverted.

Photo of update\_room\_1()

## Room 2

This is the simplest room it has only two buttons and one LED. Since we wired buttons in parallel, from the Arduino perspective they act as a one button. Again, we added functions for getting and setting status of the LED. Logic in *“Update()”* and *”Init()”* is very similar to Room 1, just without potentiometer and LED is controlled using registers not using the PWM library.

Photo of room 2 update

## Room 3

Room 3 contains one potentiometer for color changing, a button and one LED. We decided that potentiometer will affect only one property of the light for example red or green.

We were aware of limitations of this approach and decided to implement configuration of a potentiometer’s controlled value into control panel, in final version it is possible to choose which part of the light’s color is controlled by a potentiometer. We also added HSV color model because it is much more easier to work with for people. So you can configure red, green, blue, hue, saturation, value whilst red is default.

Photo of update function

Potentiometer is read the same way as in the room 2 and LED is controlled using PWM 3 from Alin’s library (Reference to his library), turning on and off using the switches between full white and completely off.

## Door

Door’s LED is controlled by PIR sensor and LDR, therefore there is no button. In addition to *“Init()”* and *“Update()”* functions we implemented functions for detecting movement, getting current value of LDR and setting threshold for LDR.

In *“Init()”* function we just initialized ADC using Alin’s library (Reference to Alin’s library) and pin for LED as output and pin for PIR sensor as input using registers.

Every time the movement is detected and LDR has higher value than threshold LED is turned on and timer is reset. Timer is just one variable which is incremented by one every time the *“Update()”* function is called, when the LED is turned on the variable is set to zero, when this variable reaches particular value LED is turned off.

Photo of update function.

## Control panel

Even though it is not a room, it works very similarly to one, because buttons function the same as in the rooms and control panel and background logic is just a more complicated version of a LED.

Photo of init function.

In addition to physical components, which make up control panel, we had to implement back-end logic for the menu responsible for storing and navigating in menu elements, we also implemented some type of a link or connection between control panel and rooms, we called it mediator.

Photo of update function

Mediator’s responsibilities are to retrieve and change information for room associated with menu element for example for element showing status of LED in room 1, mediator will return status of led in room 1 and element will be updated, when value in menu element is changed mediator ensures that room is updated as well.

### Menu

When we were designing a menu, we took inspiration from files and folders in a computer, every element in a menu can be file or folder.

Folder contains more folders or files, for example at the top level of the menu there are four folders (room 1, room 2, room 3, door) after selecting one of the folders its content will be displayed on control panel. Returning back to previous folder is done by clicking “back” button.

Files hold values which can be linked to room or it can be some arbitrary value which affects other files for example file “Color format” can have value “RGB” or “HSV” and this value affect how is color of the light in room 3 represented on control panel. After selecting file its value can be changed by clicking “up” and “down” buttons, changes take effect instantaneously. File can be deselected by clicking “back” button.

Files and Folders are contained in “Menu\_element” struct (reference C structs), this struct has a variable of enumeration type(reference enums in C) to determine if current menu element is file or folder, after this variable is union(reference C unions) with file or folder struct in it. Only differences between folder and file struct are that file has also value, type of data stored in it and variable which determines if it is selected.

Photo of these structs

Our first idea for storing menu elements was to use multidimensional array, but we quickly realized that it would be difficult to work with it, because we can’t determine the size of an array in C and we were risking going out of bounds of it. For that reason, we stored all the elements in one dimensional array and location of the element we encoded in its “ID”.

ID acts as a path to the element for example after clicking second folder on first level first element on second level will have ID equals to 12, path is inverted (first level is last digit and last level is first digit) so it is easier to search in array of elements.

Image of menu structure in drawio

### Display

To simplify the use of display we created helper library, which has functions for updating whole display with menu elements, which are passed to the function as a parameter, and for updating one particular element, so display is not refreshed every time a value in one element is changed only that element is refreshed.

Photo of display.h

I addition to displaying current menu elements, display also mark hovered element with “\*” character and selected element with “>” character.

## Control panel connection to other rooms

As it was mentioned in previous chapters to update rooms using control panel we are using Mediator. Mediator’s API consists of two functions which return pointer to a function one returns a function for changing value and the second one returns a function for getting a value. Which function to return is determined by menu element ID.

Photo of mediator.h

When changing a value we went with approach to change a value by a step

## Problems