Design Laboratory

Documentation of the project
“Smarthome”

Created by: **Bąba Marcin, Kozak Mikołaj**

Leader: **Koryciak Sebastian**

1. The aim of the project

The purpose of the project was to create a home control center on the STM32 microcontroller in which users can set selected functionalities in each room or a garden. Also the microcontroller will download an current outside temperature and will display it to the user.

2. Project description

In this project we used a microcontroller STM32F469IDISCOVERY with touchscreen and software for STM microcontrollers:
- TouchGFXDesigner - to create a graphics GUI
- STM32CubeIDE - to edit a C++ source code

STM32F469IDISCOVERY

To create graphics design we used thenounproject.com for icons and archon.pl for house design. We also used GIMP for editing colours, combining images etc.

To create an application which downloads current outside temperature we used Python language with special libraries for sending data by UART and for downloading weather.

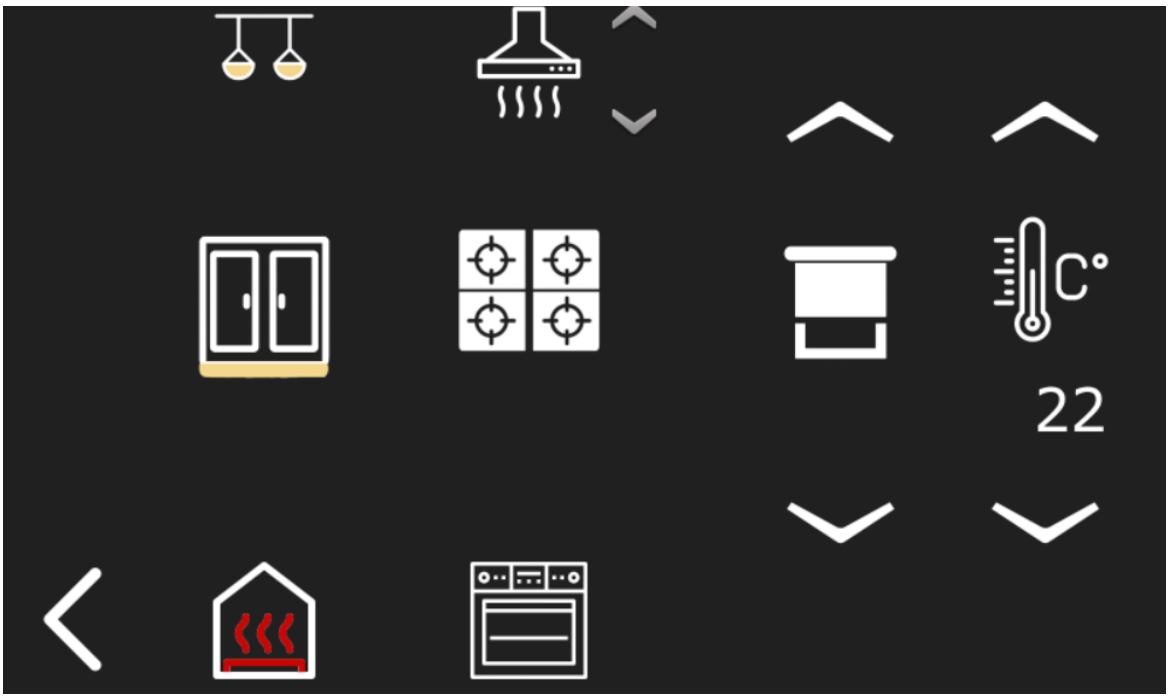
3. Presentation of the project functionality

At the beginning of the program, it shows us the start screen that provides a choice of which part of the home we want to enter.

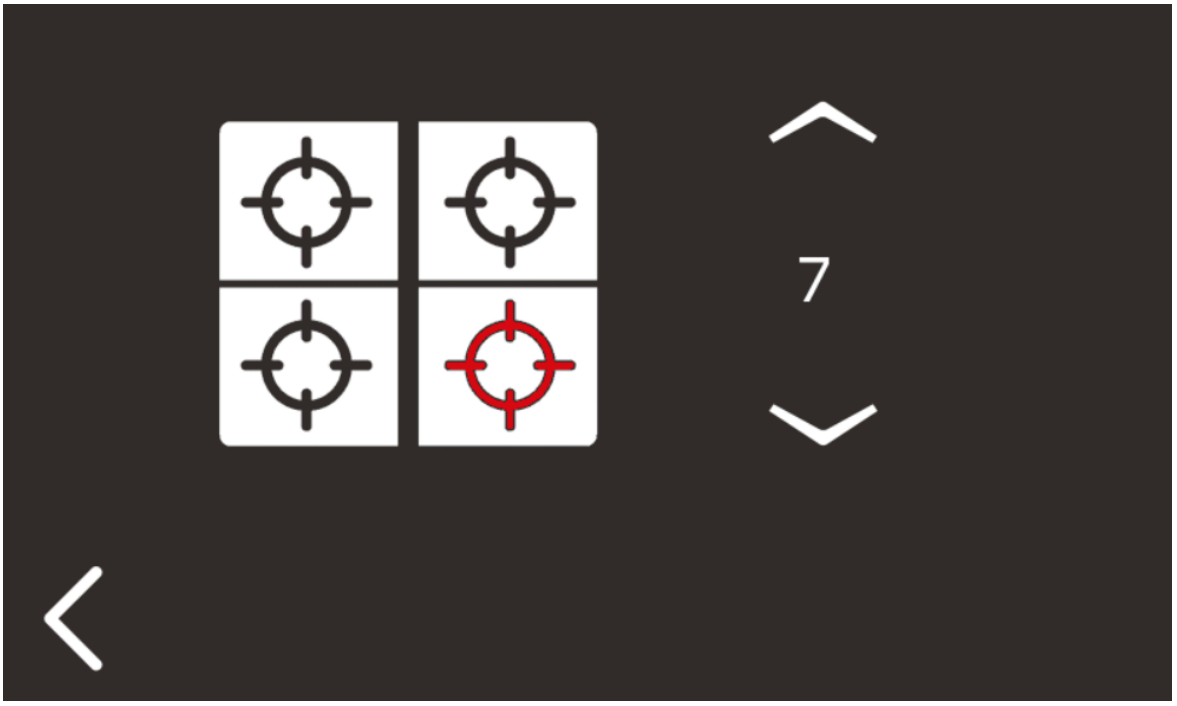
The current temperature from Krakow is displayed in the top right corner. This data is sent via UART from the Python application.

By touching one of the buttons, we can access either the first or second floor.

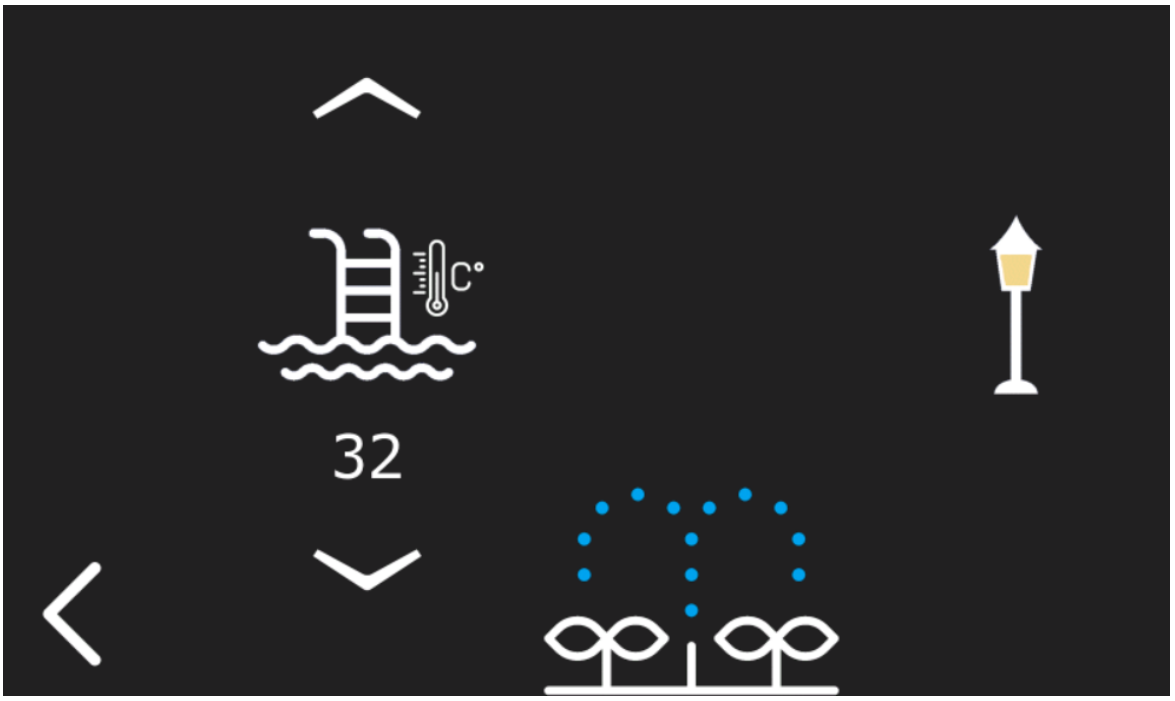
For example, this screen displays the kitchen and the devices that we can turn on or off. Additionally, each room has the possibility to adjust the temperature.



When clicking on the induction cooker, four segments appear that allow us to choose and adjust the heat settings.



Moving forward, the garden option allows for changing the temperature of the water in the pool and turning the sprinkler and outside lighting on or off.



4. Problems

The main problem of this project was saving functions selected by the user. For instance when the user turns on floor heating and exit room screen it should be set until the user turns it off.
To solve this problem we created special variables which contain information about current state and also we created setters and getters for them.

For example to make the temperature setting work we created:

- variable (randomly generated at the beginning), setter and getter in Model.cpp
- setter and getter in presenter.cpp files
- function snprintf which types our variable to the text buffer on screen while screen is starting up and special functions which increases or decreases our variable (when user clicks buttons on the screen) in View.cpp files