**Atlantic Technological University, Sligo**

Obrázok, na ktorom je text, vizitka, snímka obrazovky, písmo

Automaticky generovaný popis

Internet of Things

Room Monitoring System

Team members:

Jakub Lukac s00255726@atu.ie

Martin Strelec s00256070@atu.ie

Alekss Belavskis s00253368@atu.ie

GitHub link: <https://github.com/Jakub-Lukac/Iot_Project>

Trello link: <https://trello.com/b/5Fzlq8Bj/main>

Table of Contents

[1. Explaining the Problem 3](#_Toc160893553)

[2. Summary of Project Solution 4](#_Toc160893554)

[3. List of Project Requirements 5](#_Toc160893555)

[4. Initial Design 6](#_Toc160893556)

[5. Implementation Plan 8](#_Toc160893557)

[6. Testing approach 14](#_Toc160893558)

[7. Security Analysis 15](#_Toc160893559)

[8. Future Improvements 16](#_Toc160893560)

[9. List of images 17](#_Toc160893561)

[10. Reference list 18](#_Toc160893562)

# Explaining the Problem

As university students, our experiences often lead us to living in various places. For example, on university campuses. These accommodations often present unfamiliar problems and challenges. This includes the state and quality of the accommodation, which can affect our health and well-being. It is for this reason we decided to develop a device that allows the students, and most importantly, landlords, to monitor the main aspects that are essential in maintaining quality accommodation. These aspects include the amount of light that enters the room, the overall temperature, and humidity. These aspects can be controlled to prevent mold growth, or other undesirable living conditions.

Due to our firsthand experience in these environments, we know how hard it can be, and how it affects our well-being. In fact, digging deeper, we found out that it is not only us students and landlords that could benefit from this. Elderly people, who are particularly susceptible to substandard living conditions, would also hugely benefit from such a device. We found compelling evidence regarding the impact of cold housing on a sample of elderly people. This article discusses the health and social impacts of living in cold homes for older people in Ireland, highlighting the link between poor housing conditions and adverse health effects, such as respiratory and cardiovascular issues. It also emphasizes the financial strain and social isolation older individuals may face due to the challenges of heating their homes adequately. The research outlines the importance of addressing these issues through interventions, such as improving household energy efficiency, to mitigate the negative health and social consequences of living in cold homes. ( [Noëlle Cotter](https://www.emerald.com/insight/search?q=Noëlle%20Cotter) et al 2012)

Not only is this system useful in residential settings, but it can also be useful in monitoring the environments surrounding sensitive electrical equipment, such as servers. These devices are very delicate, and for their unobstructed operation, they need to be in a specific place with specific surrounding conditions. Another article mentioned their use of a similar control system to ours in a server area to enhance the efficiency of their monitoring and security. Their system can provide real-time monitoring and automatic responses to potential hazards in the server room. This solution aims to streamline monitoring tasks, improve security measures, and ensure timely responses to critical events within the server area, ensuring proper functionality. (Ahmad Roihan, Ferry Sudarto, Trengginas Cahyo Putro 2018)

# Summary of Project Solution

Our project aims to address the lack of environmental monitoring systems in various living spaces, such as university accommodation and elderly homes, that could benefit the wellbeing of its residents. Our room monitoring system will be able to track the temperature, humidity, and light levels of the room it's placed in.

This device will employ a microcontroller such as the Arduino YÚN or the ESP8266, connected to physical sensors that can send information about the environment to the microcontroller. Lastly, it will use a screen to display this information to the user, and a button or touch pad to customize what appears on the display. This screen features an RGB backlight which allows us to change the colour of the display depending on the temperature. This will help those with impaired vision to know the temperature without reading the figures on the display. All of this will be physically encased in a shell which will keep the electronic parts safe from damage and prevent temperament of the device.

For additional accessibility, we will develop an optional mobile app that will connect to the device and display all the same information available on the display. The mobile app will operate by communicating with an online database, such as Firebase, with which our microcontroller will share its sensor data.

In developing our solution, sustainability and security are both big priorities. Our project resources will be openly available on GitHub, a code sharing platform, which ensures easy collaboration between members. Additionally, as the code is public, it will allow users to have peace of mind knowing that the device isn’t doing anything malicious. It will also allow other developers to contribute code, such as new additions and fixes, to our codebase. We will ensure security of the device and mobile app by encrypting all the information sent between them. Most importantly, we will maintain the device’s security and keep it up to date.

This solution can extend not only to residential housing, but also to sensitive environments like server rooms. Looking ahead, we envision several enhancements to our device. For example, we could implement an additional display to show the forecast, which we would take from the internet. We could also add an air quality sensor to measure carbon dioxide levels, something that could occur from poor ventilation. This could notify the user that they should open a window or otherwise increase ventilation of the room and would benefit their health and well-being.

# List of Project Requirements

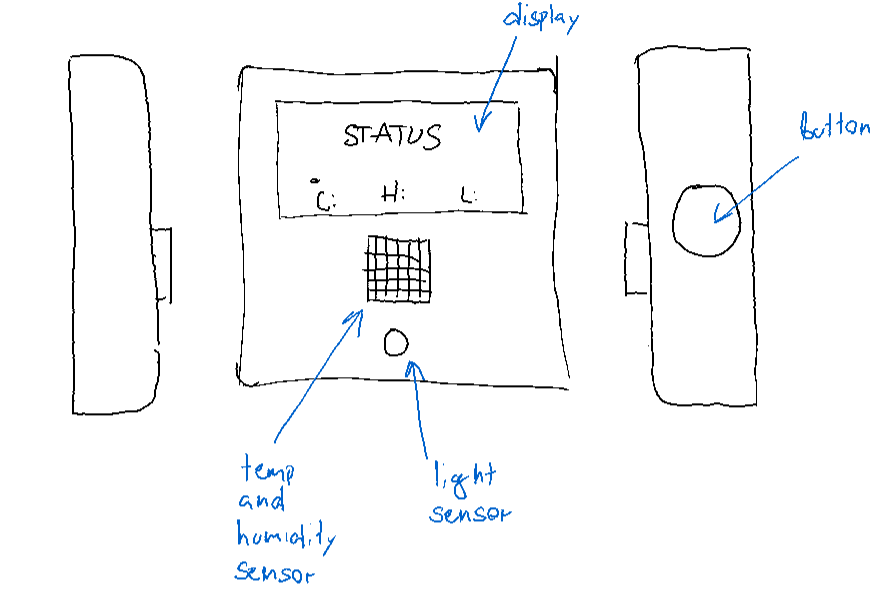
Our project had to be designed with following requirements:

The system should:

* Measure temperature.
* Measure humidity.
* Measure air quality index.
* Connect and send data through any specified wireless connection (Wi-Fi), including eduroam (WPA2 Enterprise).
* Display current state through display located on the front of the console.
* Display current state through mobile application.

# Initial Design

The first thing we needed to do was to sketch the initial design of the device. That includes the hardware setup and components used. After this we could start developing the code and sketching the initial design for the housing.



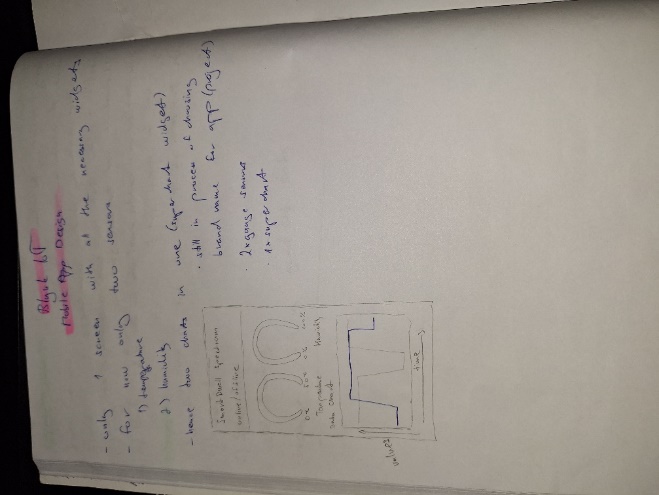
Picture 1 Initial housing design

A sketch of a cable

Description automatically generated with medium confidence

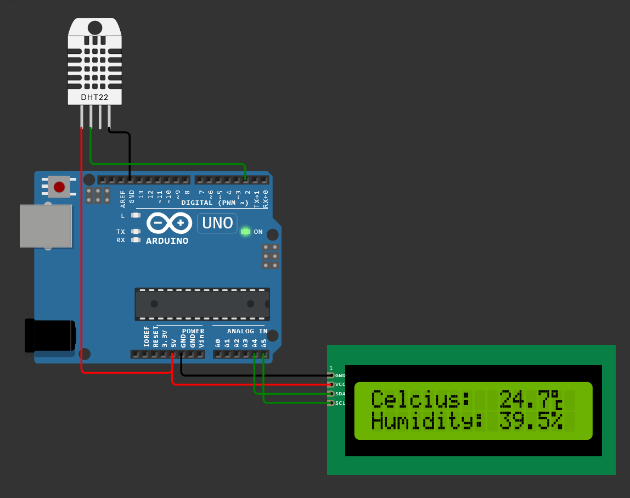
Picture 2 Dimension sketch for the components

As an initial sketch of app design, we made a low fidelity design of our app. It was then later used to form the final look of the app.



Picture 3 Low Fidelity Design for mobile app

A computer screen showing a circuit board

Description automatically generatedWe decided to use the code provided by Blynk to process our sensor data. In case of insufficient processing power and limited memory provided by the Arduino YÚN used as a main component in our project we thought of another approach eliminating these factors. We discovered that we could use a different controller board called ESP32 with Firebase database to store real time data thanks to Firebase API. The ESP32 was chosen instead of the Arduino YÚN because it has better support, and it is more suitable for our application.

Picture 4 Wiring diagram 2

Picture 5 Wiring diagram 1

In the event of using the ESP32 controller we had two options. Either use the first proposed Blynk app and its code to maintain the controller code ran on the ESP32 and the mobile app. Or we could develop a new mobile application using Flutter framework for both iOS and Android system with Firebase real-time database and Firebase API which overall has great compatibility with Flutter.

# Implementation Plan

To realize this idea of a project we needed a couple of things. First was the hardware and all the parts necessary for basic functionality of the system. That includes the following list of parts that contain both viable options described in the Initial Design section.

## Option A

* Arduino YÚN
* Arduino YÚN shield (provided from Grove kit v3.1)
* LCD screen (provided from Grove kit v3.1)
* Generic button (provided from Grove kit v3.1)
* Air quality sensor (provided from Grove kit v3.1)
* Temperature sensor (provided from Grove kit v3.1)

JST cables for connecting all the components (provided from Grove kit v3.1)

## Option B

* ESP32
* LCD screen (provided from Grove kit v3.1)
* Generic button (provided from Grove kit v3.1)
* Air quality sensor (sourced externally)
* DHT11 Temperature and Humidity sensor (sourced externally)
* JST cables for connecting all the components (provided from Grove kit v3.1)
* Soldering iron kit (sourced externally)

For both cases we designed a 3D printed housing that will accommodate all the hardware described in either option.

A black and blue device

Description automatically generatedA blue and silver box with a rectangular object

Description automatically generated with medium confidenceA silver square object with blue glass

Description automatically generatedA close-up of a device

Description automatically generated

Picture 6 Renders of final design

The API will be used only with option B being the Firebase API which is described in more detail in the Initial Design section.

The following code snippets will be used in the actual project. They refer to implementation of Grove kit’s temperature and humidity sensors as well as DHT sensors with build in methods for measuring temperature and humidity. Other code snippets also include implementation of LCD display and setting up the connection between Arduino YÚN and Blynk mobile application, and lastly connecting the Arduino YÚN to Wi-Fi.

## Code Example #1

#define BLYNK\_TEMPLATE\_ID "TMPL4fnh6fOfH"

#define BLYNK\_TEMPLATE\_NAME "TESTING"

#define BLYNK\_AUTH\_TOKEN "euvMzHZrgcq9EjXGBV3cYtoN-rjmSNgo"

/\* Comment this out to disable prints and save space \*/

#define BLYNK\_PRINT Serial

#include <BlynkRpcClient.h>

#include <SPI.h>

#include <WiFi101.h>

#include <BlynkSimpleWiFiShield101.h>

#include <Wire.h>

#include "rgb\_lcd.h"

// Your WiFi credentials.

// Set password to "" for open networks.

char ssid**[]** **=** "DobryDen"**;**

char pass**[]** **=** "DobryDen"**;**

BlynkTimer timer**;**

float celsius**;**

// This function is called every time the Virtual Pin 0 state changes

BLYNK\_WRITE**(**V0**)**

**{**

// Set incoming value from pin V0 to a variable

int value **=** param**.**asInt**();**

// Update state

Blynk**.**virtualWrite**(**V1**,** value**);**

**}**

// This function sends Arduino's uptime every second to Virtual Pin 2.

void myTimerEvent**()**

**{**

// You can send any value at any time.

// Please don't send more than 10 values per second.

Blynk**.**virtualWrite**(**V2**,** celsius**);**

**}**

rgb\_lcd lcd**;**

const float BETA **=** 3950**;**

void setup**()**

**{**

Serial**.**begin**(**115200**);**

Blynk**.**begin**(**BLYNK\_AUTH\_TOKEN**,** ssid**,** pass**);**

timer**.**setInterval**(**1000L**,** myTimerEvent**);**

lcd**.**begin**(**16**,** 2**);**

lcd**.**setRGB**(**255**,** 0**,** 255**);**

delay**(**1000**);**

**}**

void loop**()**

**{**

Blynk**.**run**();**

timer**.**run**();**

int analogValue **=** analogRead**(**A0**);**

celsius **=** 1 **/** **(**log**(**1 **/** **(**1023. **/** analogValue **-** 1**))** **/** BETA **+** 1.0 **/** 298.15**)** **-** 273.15**;**

lcd**.**setCursor**(**0**,** 1**);**

lcd**.**print**(**celsius**);**

**}**

## Code Example #2

#include "DHT.h"

#define DHTPIN 2

#define DHTTYPE DHT22

DHT dht**(**DHTPIN**,** DHTTYPE**);**

#include "LiquidCrystal\_I2C.h"

LiquidCrystal\_I2C lcd**(**0x27**,** 20**,** 4**);**

uint8\_t deg**[**8**]** **=** **{**

0b11100**,**

0b10100**,**

0b11100**,**

0b00000**,**

0b00111**,**

0b00100**,**

0b00100**,**

0b00111**,**

**};**

uint8\_t percent**[**8**]** **=** **{**

0b11000**,**

0b11001**,**

0b00010**,**

0b00100**,**

0b01000**,**

0b10011**,**

0b00011**,**

0b00000**,**

**};**

void setup**()** **{**

Serial**.**begin**(**9600**);**

Serial**.**println**(**F**(**"DHTxx test!"**));**

dht**.**begin**();**

lcd**.**init**();**

lcd**.**backlight**();**

lcd**.**setCursor**(**0**,** 0**);**

lcd**.**print**(**"Celsius:"**);**

lcd**.**setCursor**(**0**,** 1**);**

lcd**.**print**(**"Humidity:"**);**

**}**

void loop**()** **{**

delay**(**2000**);**

float h **=** dht**.**readHumidity**();**

float t **=** dht**.**readTemperature**();**

float f **=** dht**.**readTemperature**(true);**

float hif **=** dht**.**computeHeatIndex**(**f**,** h**);**

float hic **=** dht**.**computeHeatIndex**(**t**,** h**,** **false);**

uint8\_t deg2**[**8**]** **=** **{**0**};**

uint8\_t percent2**[**8**]** **=** **{**0**};**

**for** **(**int i **=** 0**;** i **<** 8**;** i**++)** **{**

deg2**[**i**]** **=** deg**[**i**];**

lcd**.**createChar**(**3**,** deg2**);**

percent2**[**i**]** **=** percent**[**i**];**

lcd**.**createChar**(**4**,** percent2**);**

delay**(**100**);**

**}**

delay**(**500**);**

lcd**.**setCursor**(**10**,** 0**);**

lcd**.**print**(**t**);**

lcd**.**setCursor**(**14**,** 0**);**

lcd**.**print**(**"\x03"**);**

lcd**.**setCursor**(**10**,** 1**);**

lcd**.**print**(**h**);**

lcd**.**setCursor**(**14**,** 1**);**

lcd**.**print**(**"\x04"**);**

**}**

## Code Example #3

#include "LiquidCrystal\_I2C.h"

LiquidCrystal\_I2C lcd**(**0x27**,** 20**,** 4**);**

const float BETA **=** 3950**;**

void setup**()** **{**

Serial**.**begin**(**9600**);**

lcd**.**init**();**

lcd**.**backlight**();**

lcd**.**setCursor**(**1**,** 0**);**

lcd**.**print**(**"Temperature"**);**

**}**

void loop**()** **{**

int analogValue **=** analogRead**(**A0**);**

float celsius **=** 1 **/** **(**log**(**1 **/** **(**1023. **/** analogValue **-** 1**))** **/** BETA **+** 1.0 **/** 298.15**)** **-** 273.15**;**

lcd**.**setCursor**(**7**,** 1**);**

lcd**.**print**(**celsius**);**

**}**

## Code Example #4

#define BLYNK\_TEMPLATE\_ID "TMPL4fnh6fOfH"

#define BLYNK\_TEMPLATE\_NAME "TESTING"

#define BLYNK\_AUTH\_TOKEN "euvMzHZrgcq9EjXGBV3cYtoN-rjmSNgo"

/\* Comment this out to disable prints and save space \*/

#define BLYNK\_PRINT Serial

#include <BlynkRpcClient.h>

#include <SPI.h>

#include <WiFi101.h>

#include <BlynkSimpleWiFiShield101.h>

#include <Wire.h>

#include "rgb\_lcd.h"

// Your WiFi credentials.

// Set password to "" for open networks.

char ssid**[]** **=** "DobryDen"**;**

char pass**[]** **=** "DobryDen"**;**

BlynkTimer timer**;**

float celsius**;**

// This function is called every time the Virtual Pin 0 state changes

BLYNK\_WRITE**(**V0**)**

**{**

// Set incoming value from pin V0 to a variable

int value **=** param**.**asInt**();**

// Update state

Blynk**.**virtualWrite**(**V1**,** value**);**

**}**

// This function sends Arduino's uptime every second to Virtual Pin 2.

void myTimerEvent**()**

**{**

// You can send any value at any time.

// Please don't send more that 10 values per second.

Blynk**.**virtualWrite**(**V2**,** celsius**);**

**}**

rgb\_lcd lcd**;**

const float BETA **=** 3950**;**

void setup**()**

**{**

Serial**.**begin**(**115200**);**

Blynk**.**begin**(**BLYNK\_AUTH\_TOKEN**,** ssid**,** pass**);**

timer**.**setInterval**(**1000L**,** myTimerEvent**);**

lcd**.**begin**(**16**,** 2**);**

lcd**.**setRGB**(**255**,** 0**,** 255**);**

delay**(**1000**);**

**}**

void loop**()**

**{**

Blynk**.**run**();**

timer**.**run**();**

int analogValue **=** analogRead**(**A0**);**

celsius **=** 1 **/** **(**log**(**1 **/** **(**1023. **/** analogValue **-** 1**))** **/** BETA **+** 1.0 **/** 298.15**)** **-** 273.15**;**

lcd**.**setCursor**(**0**,** 1**);**

lcd**.**print**(**celsius**);**

**}**

Final Code

#define BLYNK\_PRINT Serial

#include "secrets.h"

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

#include "esp\_wpa2.h"

#include <DHT.h>

#include <Wire.h>

#include "rgb\_lcd.h"

#define MQ135PIN A4 // Analog pin for MQ-135 air quality sensor

#define DHTPIN A0 // Digital pin for DHT-11 temperature/humidity sensor

#define DHTTYPE DHT11

#define CONTROLLERPIN A27 // Analog pin for button/rotary controller to control the RGB LCD

DHT dht**(**DHTPIN**,** DHTTYPE**);** // Types: DHT11, DHT22 (AM2302, AM2321), DHT21 (AM2301)

rgb\_lcd lcd**;**

BlynkTimer timer**;**

uint8\_t degreeSymbol**[**8**]** **=** **{** 0b11100**,** 0b10100**,** 0b11100**,** 0b00000**,** 0b00111**,** 0b00100**,** 0b00100**,** 0b00111 **};**

float celcius**;**

float humidity**;**

int airQuality**;**

int button **=** 0**;**

String getWifiStatus**(**int status**)** **{**

**switch(**status**)** **{**

**case** WL\_IDLE\_STATUS**:** **return** "WL\_IDLE\_STATUS"**;**

**case** WL\_SCAN\_COMPLETED**:** **return** "WL\_SCAN\_COMPLETED"**;**

**case** WL\_NO\_SSID\_AVAIL**:** **return** "WL\_NO\_SSID\_AVAIL"**;**

**case** WL\_CONNECT\_FAILED**:** **return** "WL\_CONNECT\_FAILED"**;**

**case** WL\_CONNECTION\_LOST**:** **return** "WL\_CONNECTION\_LOST"**;**

**case** WL\_CONNECTED**:** **return** "WL\_CONNECTED"**;**

**case** WL\_DISCONNECTED**:** **return** "WL\_DISCONNECTED"**;**

**}**

**}**

void setBacklightColour**(**float input**,** int min**,** int mid**,** int max**)** **{**

int r**,** g**,** b**;**

**if** **(**input **<=** min**)** **{**

r **=** 0**;**

g **=** 0**;**

b **=** 255**;**

**}** **else** **if** **(**input **<=** mid**)** **{**

r **=** 0**;**

g **=** map**(**input**,** min**,** mid**,** 0**,** 255**);**

b **=** 255 **-** g**;**

**}** **else** **{**

r **=** 255**;**

g **=** map**(**input**,** mid**,** max**,** 255**,** 0**);**

b **=** 0**;**

**}**

lcd**.**setRGB**(**r**,** g**,** b**);**

**}**

void printValueToLCD**(**int value**,** String title**,** String symbol**,** int min**,** int mid**,** int max**)** **{**

lcd**.**clear**();**

setBacklightColour**(**value**,** min**,** mid**,** max**);**

lcd**.**setCursor**(**0**,** 0**);**

lcd**.**print**(**title**);**

lcd**.**setCursor**(**2**,** 1**);**

lcd**.**print**(**value**);**

lcd**.**setCursor**(**5**,** 1**);**

lcd**.**print**(**symbol**);**

**}**

void mainTimer**()** **{**

// celcius = 15.6;

celcius **=** dht**.**readTemperature**();**

// humidity = 40.1;

humidity **=** dht**.**readHumidity**();**

**if** **(**isnan**(**celcius**)** **||** isnan**(**humidity**))** **{**

Serial**.**println**(**"Failed to read from DHT sensor!"**);**

**}**

// airQuality = 30;

airQuality **=** analogRead**(**MQ135PIN**);**

Blynk**.**virtualWrite**(**V2**,** celcius**);**

Blynk**.**virtualWrite**(**V3**,** humidity**);**

Blynk**.**virtualWrite**(**V4**,** airQuality**);**

Serial**.**print**(**"Celcius: "**);**

Serial**.**print**(**celcius**);**

Serial**.**println**(**" 'C"**);**

Serial**.**print**(**"Humidity: "**);**

Serial**.**print**(**humidity**);**

Serial**.**println**(**" %"**);**

Serial**.**print**(**"Air Quality: "**);**

Serial**.**print**(**airQuality**);**

Serial**.**println**(**" AQI"**);**

// button = 0;

button **=** analogRead**(**CONTROLLERPIN**);**

**if** **(**button **<** 80**)** **{**

printValueToLCD**(**celcius**,** "Temperature"**,** "\x03"**,** **-**10**,** 15**,** 50**);**

**}** **else** **if** **(**button **<=** 160**)** **{**

printValueToLCD**(**humidity**,** "Humidity"**,** "%"**,** 0**,** 50**,** 100**);**

**}** **else** **if** **(**button **>** 160**)** **{**

printValueToLCD**(**airQuality**,** "Air Quality"**,** "AQI"**,** 0**,** 250**,** 500**);**

**}**

**}**

void connectToWifi**()** **{**

int status **=** WL\_IDLE\_STATUS**;**

Serial**.**print**(**"\nConnecting to network: "**);**

Serial**.**println**(**WIFI\_SSID**);**

Serial**.**println**(**getWifiStatus**(**status**));**

WiFi**.**disconnect**(true);** // Disconnect from Wi-Fi to set new Wi-Fi connection

WiFi**.**mode**(**WIFI\_STA**);** // Set Wi-Fi mode

WiFi**.**begin**(**WIFI\_SSID**,** WPA2\_AUTH\_PEAP**,** EAP\_ANONYMOUS\_IDENTITY**,** EAP\_IDENTITY**,** EAP\_PASSWORD**);**

// WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

int counter **=** 0**;**

**while** **(**WiFi**.**status**()** **!=** WL\_CONNECTED**)** **{**

**if** **(**counter**++** **>=** 60**)** **{** ESP**.**restart**();** **}**

status **=** WiFi**.**status**();**

Serial**.**print**(**getWifiStatus**(**status**));**

pinMode**(**CONTROLLERPIN**,** INPUT**);**

Serial**.**println**(**"..."**);**

delay**(**500**);**

**}**

Serial**.**println**(**""**);**

Serial**.**println**(**F**(**"Connected!"**));**

Serial**.**println**(**F**(**"Local ESP32 IP: "**));**

Serial**.**println**(**WiFi**.**localIP**());**

**}**

void setup**()** **{**

Serial**.**begin**(**115200**);**

connectToWifi**();**

pinMode**(**CONTROLLERPIN**,** INPUT**);**

lcd**.**begin**(**16**,** 2**,** 0**);**

lcd**.**setRGB**(**0**,** 255**,** 0**);**

lcd**.**createChar**(**3**,** degreeSymbol**);**

dht**.**begin**();**

Blynk**.**begin**(**BLYNK\_AUTH\_TOKEN**,** WIFI\_SSID**,** WIFI\_PASSWORD**);**

timer**.**setInterval**(**3000L**,** mainTimer**);**

**}**

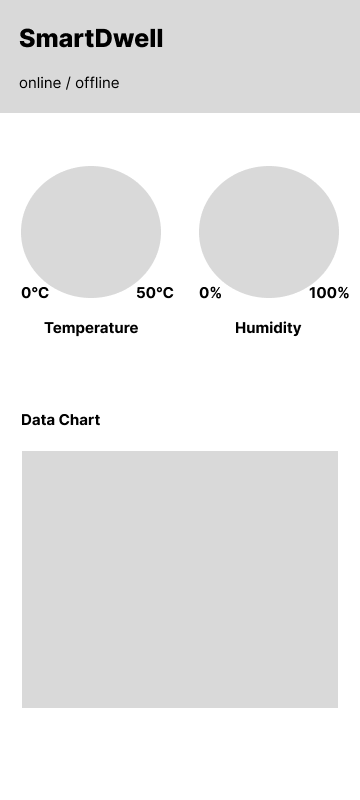
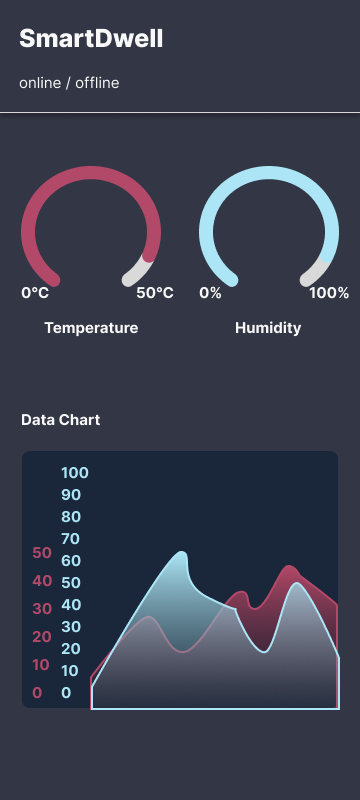
void loop**()** **{**

Blynk**.**run**();**

timer**.**run**();**

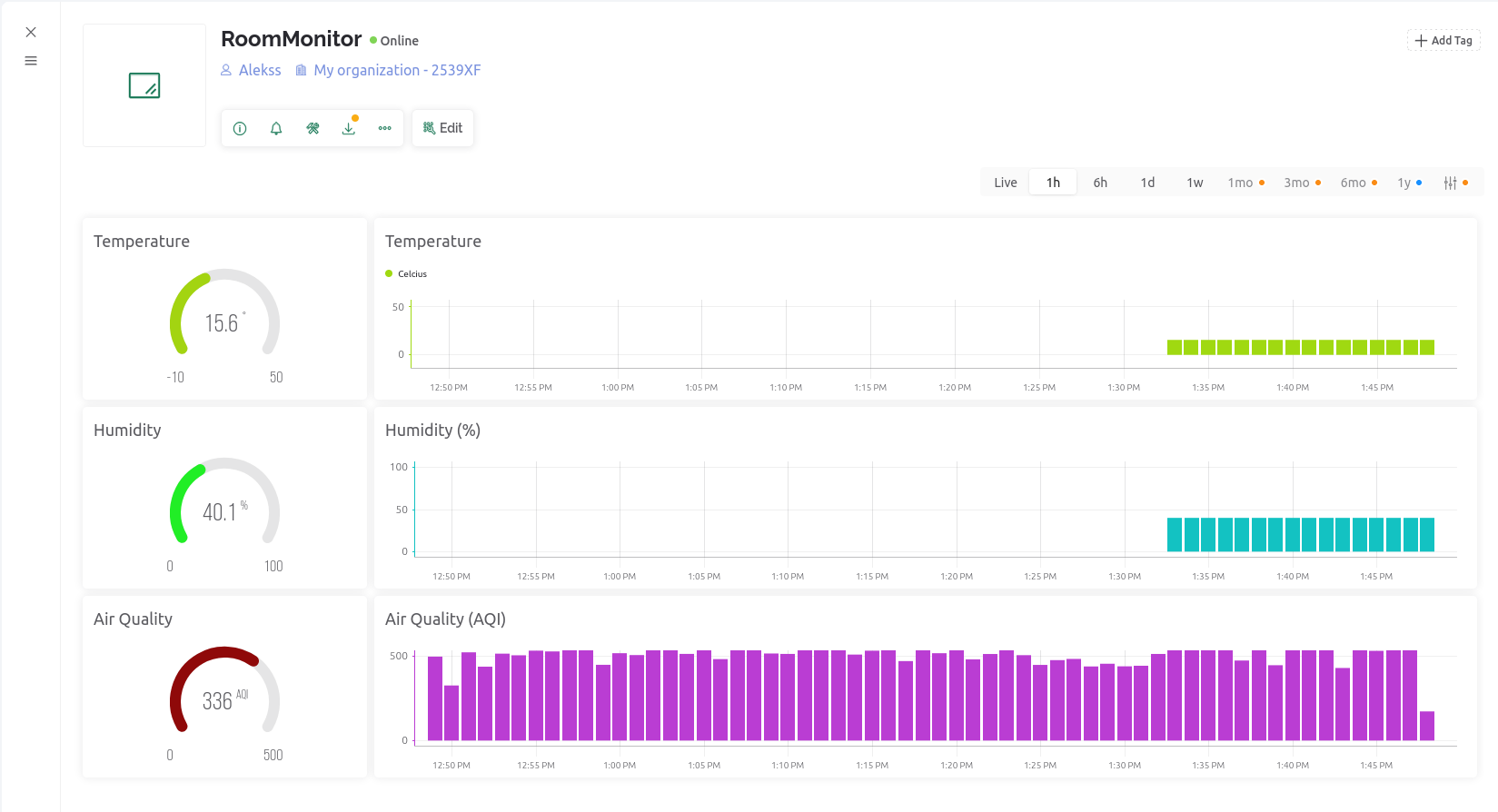
**}**

We made sure to properly read through the Blynk development platform documentation to get the better understanding of how this framework works and what widgets are available to us to develop the mobile application. We found that we are very limited when it comes to the variety of widgets, as we can only use Guage and Chart widgets. Based on that we developed very simple user interface, but at the same time we wanted to aim for great user experience, we came up with our own colour schemes to enhance the UI. After the whole team approved the Low-Fidelity prototype of the Blynk mobile application, we decided to move forward with the prototype and designed the Mid-Fidelity and the High-Fidelity prototype.



Picture 8 High-Fidelity Design

Picture 7 Mid-Fidelity Design



Picture 9 Web Dashboard

# Testing approach

To test our room monitoring system we decided to test it in real life environment that is our accommodation. This would provide useful and accurate data that we will later use to calibrate and improve our entire system. We believe that this approach will be the most beneficial as it is closely related to the project’s targeted problematic.

For accurate testing we decided to use the following testing scheme:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test case# | Title | Test Steps | Expected Result | Pass/Fail | Date |
| 1 | Measure  Temperature | 1. turn on device  2. wait | Temperature displayed in console | PASS | March 2024 |
| 2 | Measure Humidity | 1. turn on device  2. wait | Humidity displayed in console | PASS | March 2024 |
| 3 | Measure  Air Quality Index | 1. turn on device  2. wait | Light value displayed in console | PASS | April 2024 |
| 4 | Connect and send data to API | 1. turn on device  2. wait | Blynk is showing the data received from device | PASS | April 2024 |
| 5 | Display current state through LCD display | 1. turn on device  2. wait | Temperature displayed on RGB LCD | PASS | April 2024 |
| 6 | Display current state on mobile app | 1. turn on device  2. open mobile app | Climate data displayed on mobile app | PASS | April 2024 |

# Security Analysis

To prevent unauthorized breaches into our device we need to address two ways to access our device. Physical and remote. The physical part resembles the intruder physically connecting to the device via cable. This is prevented by enclosing the entire system in a 3D printed case assembled with screws. The intruder will not gain access to the device unless he is in possession of a screwdriver or will use excessive amount of force to damage or destroy the casing of the device to gain access to the Arduino YÚN ports.

Regarding the remote access the Arduino will be secured by the password, meaning that if by any chance the intruder would want to gain access to our device, he would firstly need to know the Wi-Fi password. Another way of defending against malicious thread is the implementation of authorization of the Blynk mobile application when connecting via secured Wi-Fi to Arduino YÚN. In the Arduino YÚN code we defined three very important lines of code, which are the Blynk Template ID, Blynk Template Name and mostly importantly the Blynk Authorization token, which represents the core of the authorization.

We made use of a ‘secrets.h’ file to store sensitive credentials used in the main code. A ‘.gitignore’ file is used to prevent the ‘secrets.h’ from being uploaded to github. We also included a ‘secrets.h-example’ file to show how the secrets file should be formatted. This prevents our sensitive credentials for Blynk, Wi-Fi, etc, from showing up on the internet.

# Future Improvements

We plan to implement a second display that will be used to show the user forecast for the next 24 hours. This will expand the functionality of the device. To implement this, however, we will have to do some user testing before making this improvement to evaluate if this improvement will be useful in the long run.

The design of the case will also be updated to make the casing smaller and more convenient to put it into a regular household. In its current state the device cannot be mounted otherwise than just facing upwards laying down on the table. This will also be reworked as we plan to mount it to the wall.

Regarding the App design we plan to purchase a subscription plan for Blynk app development to widen our possibilities and make the app more functional than it is right now. We can also enhance the user experience by creating modern and pleasant user interfaces using the wide variety of widgets which are only available with subscription. Another approach would be to develop our own app from scratch using the previously mentioned flutter framework and Firebase real-time database with Firebase API. This will allow us to, as developers, to have more and almost full control of the entire life cycle of the app.

Other improvements include maintaining the device and keeping it up to date with security standards and other requirements.

# List of images

[Picture 1 Initial housing design 6](#_Toc160981258)

[Picture 2 Dimension sketch for the components 6](#_Toc160981259)

[Picture 3 Low Fidelity Design for mobile app 7](#_Toc160981260)

[Picture 4 Wiring diagram 1 7](#_Toc160981261)

[Picture 5 Wiring diagram 2 7](#_Toc160981262)

[Picture 6 Renders of final design 9](#_Toc160981263)

[Picture 7 Mid-Fidelity Design 14](#_Toc160981264)

[Picture 8 High-Fidelity Design 14](#_Toc160981265)

Picture 9 Web Dashboard………………………………………………………. ……………………………………………………..18

# Reference list

Noëlle Cotter, Eugene Monahan, Helen McAvoy, Patrick Goodman 2012, ‘Coping with the cold – exploring relationships between cold housing, health and social wellbeing in a sample of older people in Ireland’ *emerald* [Online]. Available from: <https://www.emerald.com/insight/content/doi/10.1108/14717791211213607/full/html>  
[Viewed 8 March 2024].

Ahmad Roihan, Ferry Sudarto, Trengginas Cahyo Putro 2018, ‘Internet of Things on Monitoring and Control System in Server Area’ *ieeexplore* [Online]. Available from: <https://ieeexplore.ieee.org/abstract/document/8549759>  
[Viewed 8 March 2024].