****

**Internet of Things**

**Minor Group Project**

**Team report**

**SCREEN TIME REDUCING ALARM CLOCK**

|  |  |
| --- | --- |
| Team Members and Student Numbers | Aoife Kelly (S00274039@atu.ie)  Eliška Tintěrová (S00273768@atu.ie)  Kennedy Moore (S00271816@atu.ie)  Mansura Khajra (S00265100@atu.ie)  Silvana Pereira Da Silva (S00273672@atu.ie)  Shauna McGowan (S00273772@atu.ie) |
| Class Group | Group C |
| Project Idea | Screen time reducing alarm clock |

**Table of Contents**

|  |  |
| --- | --- |
| **Section** | **Pages** |
| Data gathered | 3 |
| Outline of the problem to be solved with reference list | 4-6 |
| Summary of project solution | 7-8 |
| List of project requirements | 9 |
| Initial design | 10-11 |
| Description of how data was generated, stored, and managed | 12 |
| Implementation plan | 13- 15 |
| Testing approach | 16-18 |
| Security analysis | 19 |
| Future improvements | 20 |

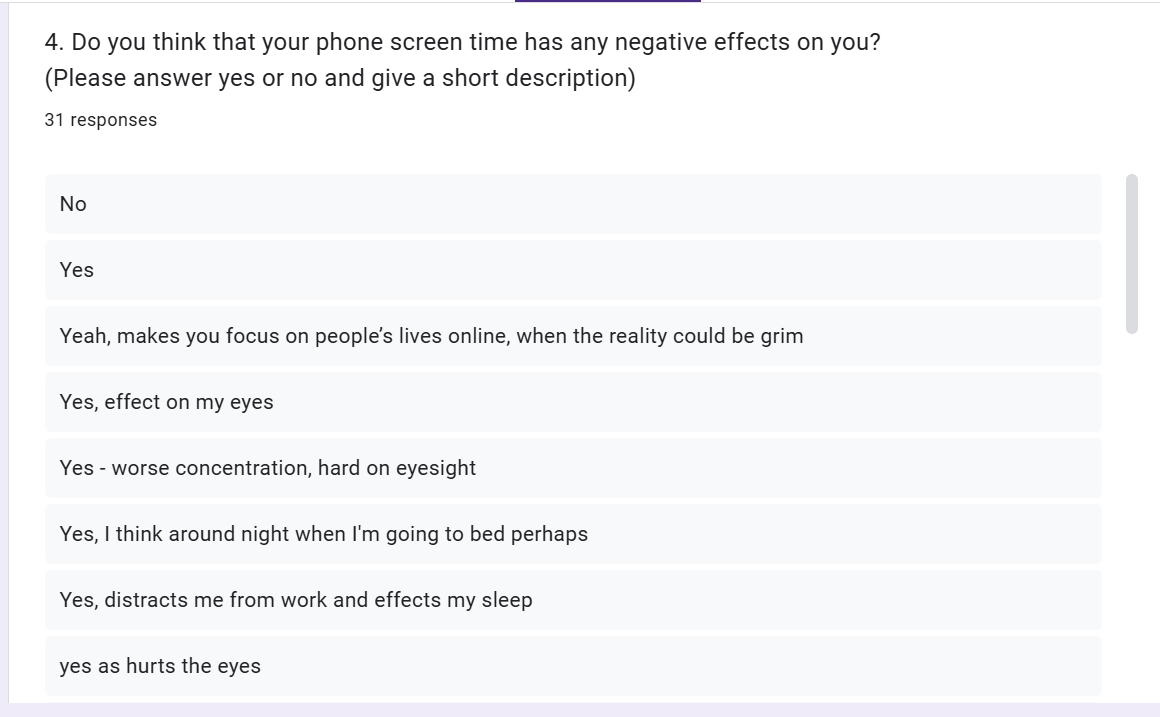
**Trello link** [**https://trello.com/invite/b/67af1084259db96e5e6cd756/ATTI77e717670b9ef81576b33d57d2cf5b96AD389854/iot-project**](https://trello.com/invite/b/67af1084259db96e5e6cd756/ATTI77e717670b9ef81576b33d57d2cf5b96AD389854/iot-project)

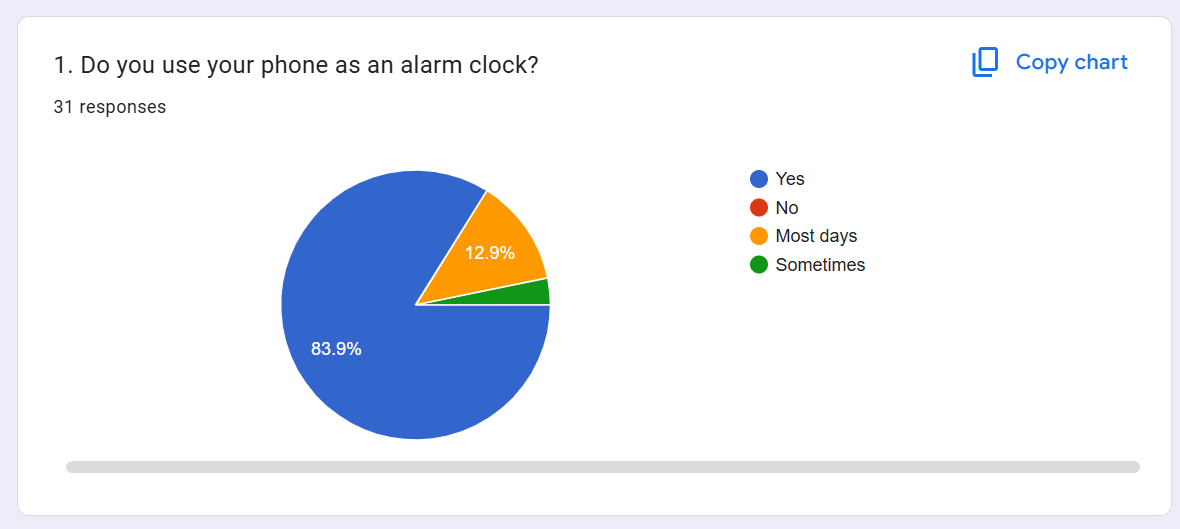
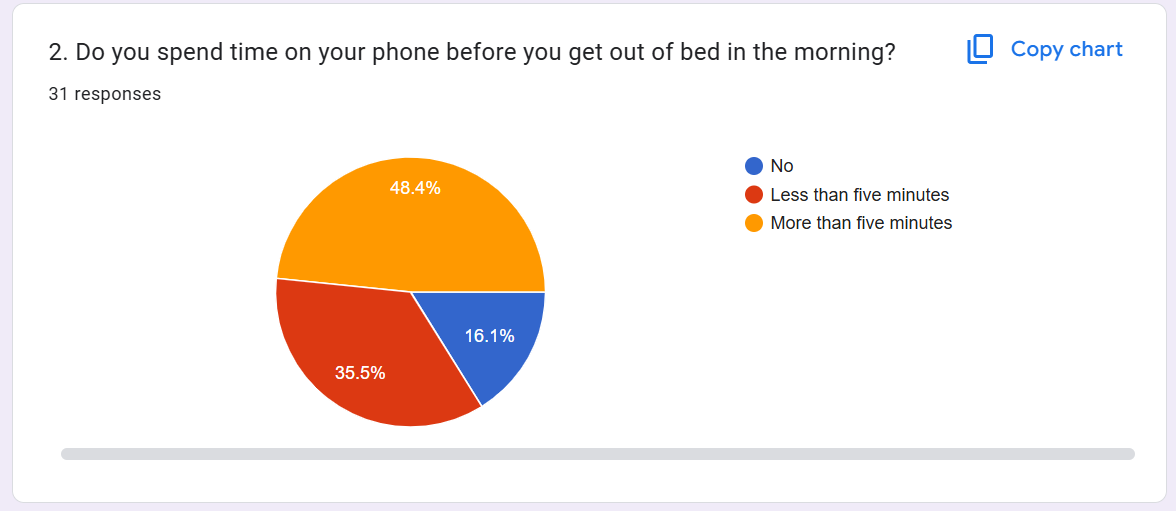
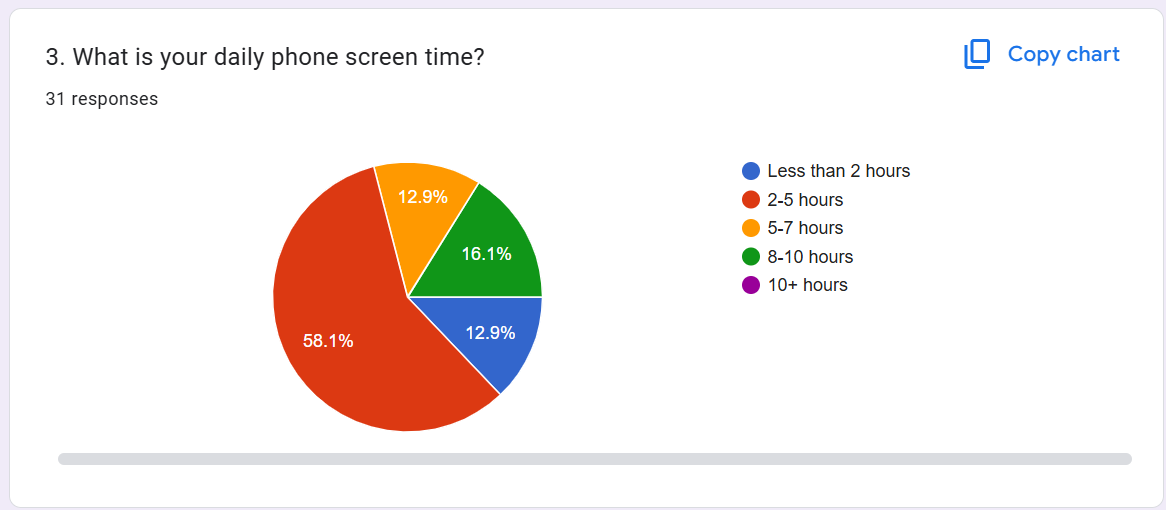
**Github link**

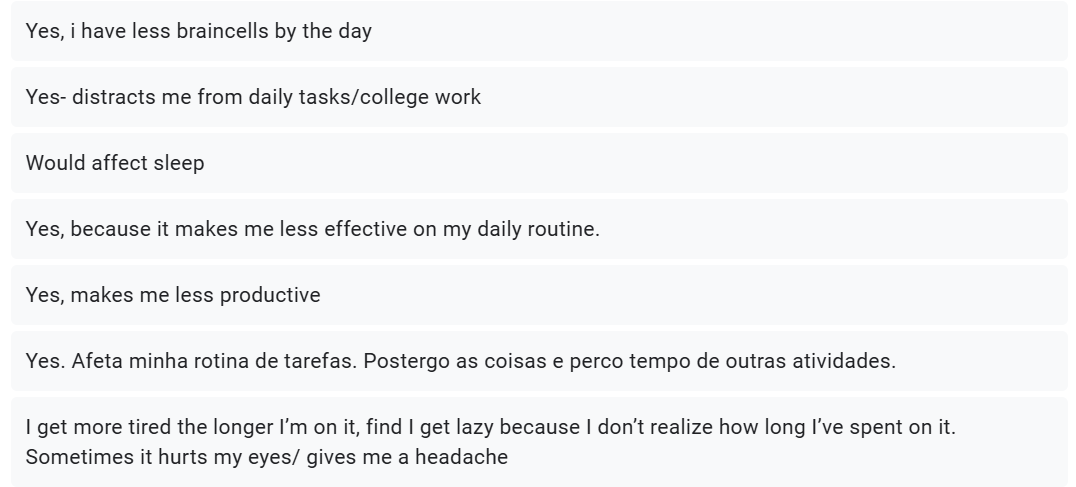
[**https://github.com/kennedy-mr/InternetIOfThings.git**](https://github.com/kennedy-mr/InternetIOfThings.git)

**Survey**

[**Phone Screen Time**](https://docs.google.com/forms/d/1wtenDGqmxgHJtg9f5e3JVgWPTyt4OIORKKwPOl2jV9s/viewform?edit_requested=true)

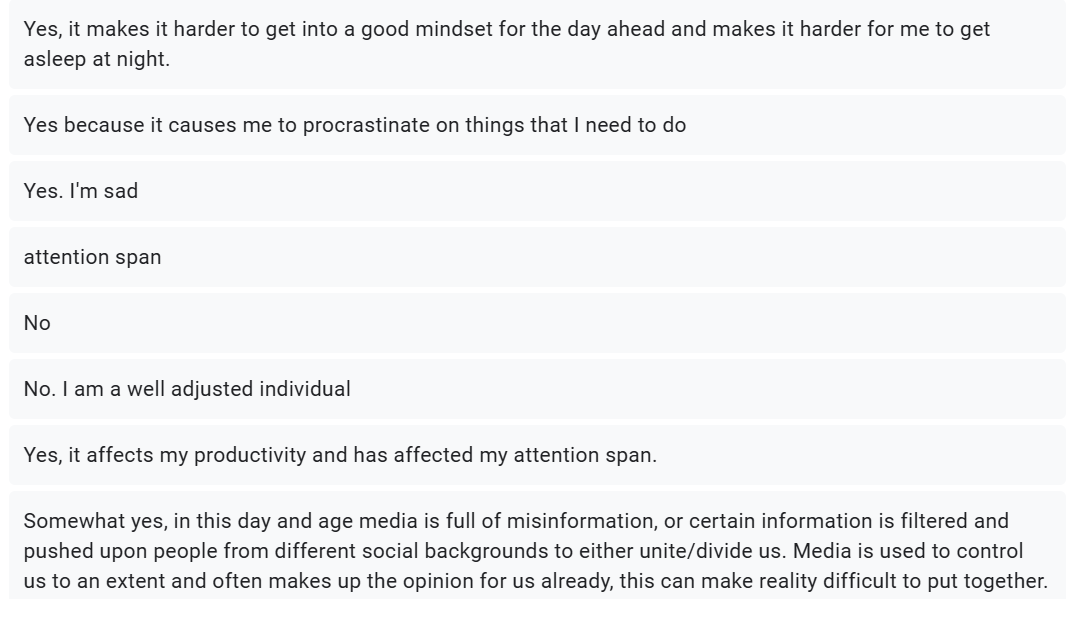
**Data gathered**

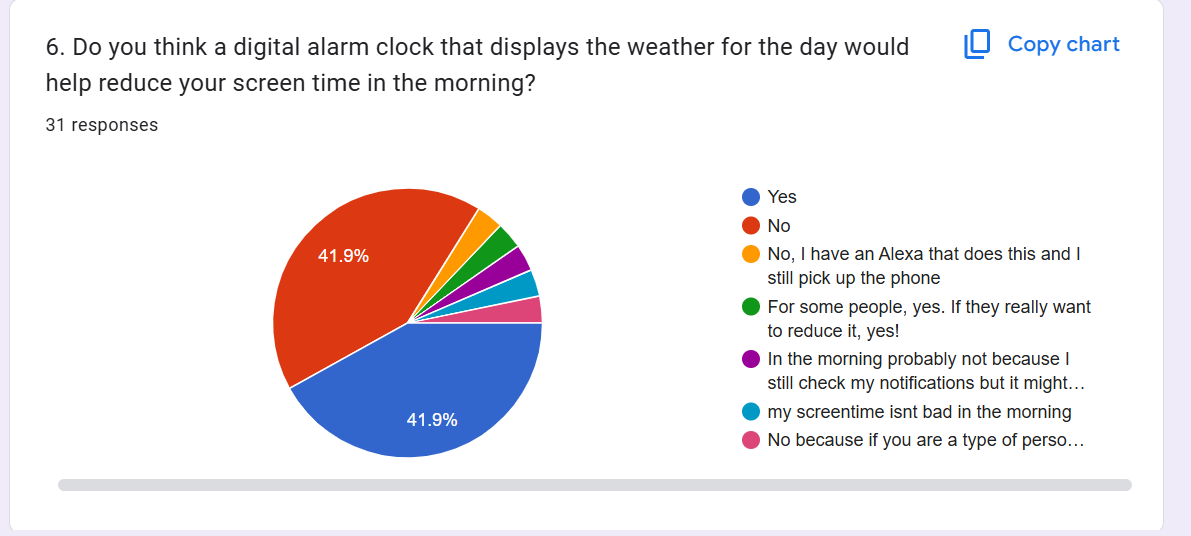


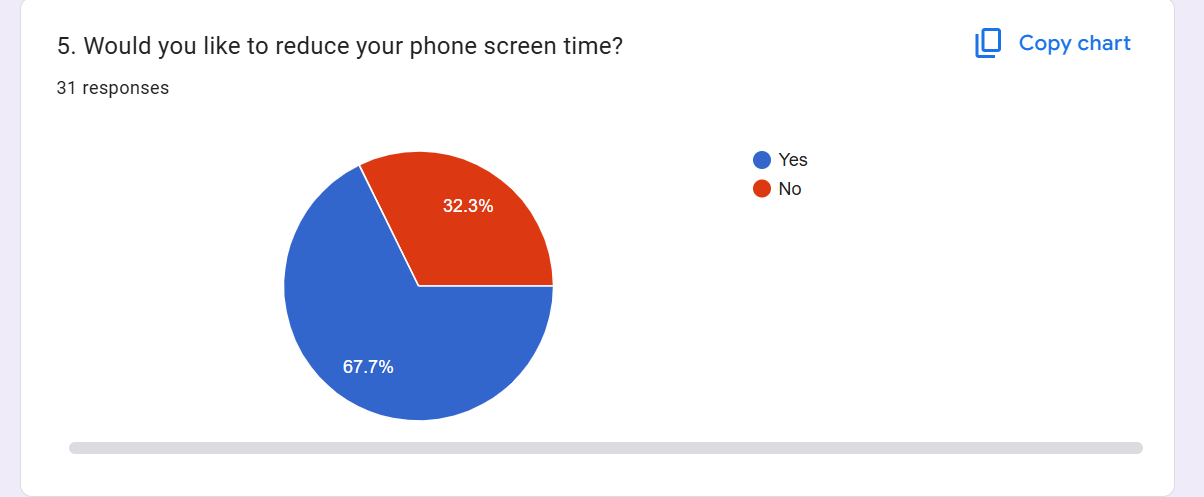


**A screenshot of a social media post

AI-generated content may be incorrect.**







**Outline of the problem to be solved**

Screen time refers to the amount of time spent looking at screens such as televisions, laptops, or phone screens (Merriam-Webster, 2019). In their 2024 study, Deyo et al. found that the American college students in their sample spent an average of seven hours per day on hand-held devices such as phones and tablets. High screen times can have very negative effects on physical and mental health.

Spending time looking at screens can lead to sedentary behaviour. Wang et al. (2019) found a link between sedentary behaviour as a result of screen time leading to depressive symptoms when screen time was greater than 2 hours per day. This could be caused by people withdrawing socially in favour of screen time based activity (Wang et al., 2019). Choosing screen time over interaction with people may also lead to cognitive decline. Neophytou et al. (2021) point out that social bonds and in-person interactions could delay cognitive decline as we age, while screen time is associated with a greater risk of cognitive decline and dementia.

Sleep is negatively affected by phone use. Screen time before sleeping leads to poor sleep quality, insomnia, delayed sleep onset and shorter sleep duration (Christensen et al. 2016). This in turn can lead to adverse effects such as poorer performance in school, college, or work, and increased risk of obesity and cardiovascular disease (Christensen et al. 2016).

We conducted a survey of thirty friends and family and found that the most prevalent phone screen time was 2-5 hours with 56.7 % of the participants clicking this option. 16.7% of participants indicated that they spend 8-10 hours a day on their phone. 70% of the participants admitted that they would like to reduce their phone screen time. We asked participants to share any negative effects that their phone screen time might have on them. Some responses included distraction from responsibilities such as chores and college work, increasing procrastination and reducing productivity, and eye pain and headaches. In total, 86.7% of participants had a screen time of 2 or more hours per day, which can be associated with increased risk for depressive symptoms (Wang et al., 2019).

Given the negative effects caused by high screen times we decided to work on a project that could help people to reduce their screen time. As a group, we thought about what leads us to looking at our phones when we wake up and agreed that phone alarms and checking the weather for the day were two of the main reasons. We felt that starting off on the right foot each morning with a digital alarm clock could be a good step in the right direction. We feel that this project idea strongly aligns with Sustainable Development Goal Number 3: Good Health and Well-Being.

**References**

Christensen MA, Bettencourt L, Kaye L, Moturu ST, Nguyen KT, Olgin JE, et al. (2016) Direct Measurements of Smartphone Screen-Time: Relationships with Demographics and Sleep. PLoS ONE 11(11): e0165331. <https://doi.org/10.1371/journal.pone.0165331>

Deyo, A., Wallace, J. and Kidwell, K.M. (2024). Screen time and mental health in college students: Time in nature as a protective factor. *Journal of American College Health*, 72(8), pp.1–8. doi:<https://doi.org/10.1080/07448481.2022.2151843>.

Merriam-webster (2019). *Definition of SCREEN TIME*. [online] Merriam-webster.com. Available at: <https://www.merriam-webster.com/dictionary/screen%20time>.

‌Neophytou, E., Manwell, L.A. & Eikelboom, R. Effects of Excessive Screen Time on Neurodevelopment, Learning, Memory, Mental Health, and Neurodegeneration: a Scoping Review. *Int J Ment Health Addiction* 19, 724–744 (2021). <https://doi.org/10.1007/s11469-019-00182-2>

**Summary of the project solution**

This project aims to provide a gadget designed to function as an alarm clock while also reducing screen time. It offers an alternative to the first thing users do in the morning being to look at their phones.

This project (screen-time reducing alarm clock) involves using an Arduino to create a device that can utilize the Met Éireann API to display a weather forecast, a positive message for the day, or a reminder for the day. The integration of the Met Éireann API allows the device to provide real-time weather updates, enhancing its functionality and user experience. Using the Met Éireann API improves its usefulness and makes it more user-friendly.

In our project, we chose to utilize an LCD (Liquid Crystal Display) as the central component to improve user interaction. This allows users to see the time, view a weather forecast or in the future read a positive message for the day. The LCD is connected at Base Shield, through Grove Cables.

In the first stage of the project, we are building an LCD screen that displays the time. The time is hard-coded in a 24-hour format to ensure the alarm does not go off twice a day. There are two numbers separated by a colon, both being two-digit numbers. The second number increments by one every 60 seconds. The first number changes every time the second number hits 00, which occurs after the second number is reset (when it reaches 60, it automatically resets to 00). The first number resets when it reaches 24.

In the second stage, the LCD will display the forecast for the day and show the user a message about their day, based on their schedule (input in a spreadsheet or notes app). In the future we plan to include suggestions and little reminders like taking daily pills, health supplements, or tasks from their to-do list for that day.

One of the sensors used in the prototype is the buzzer. It unmutes and wakes the user at the scheduled time. Initially, the alarm time is pre-programmed, but the goal is to allow the user to set the desired time themselves. The buzzer operates by comparing the current time with the alarm time, and when they match, it activates.

The buzzer is deactivated by another sensor – a button. When the button is pressed, it stops the buzzer. We also track the last state of the buzzer to prevent the alarm from going off twice within one minute, as the code runs in a loop. Currently, the button is used to deactivate the alarm. Meanwhile, efforts are underway to enhance its functionality, including allowing the user to set the alarm time.

To ensure the time aligns with the Ireland time zone and eliminates the user's concerns about adjusting for daylight saving time, we will include an additional sensor in the project. This RTC (Real-Time Clock) sensor was not part of the initial Grove Kit. It allows the alarm clock to display real time without requiring the time to be hardcoded in the alarm clock code.

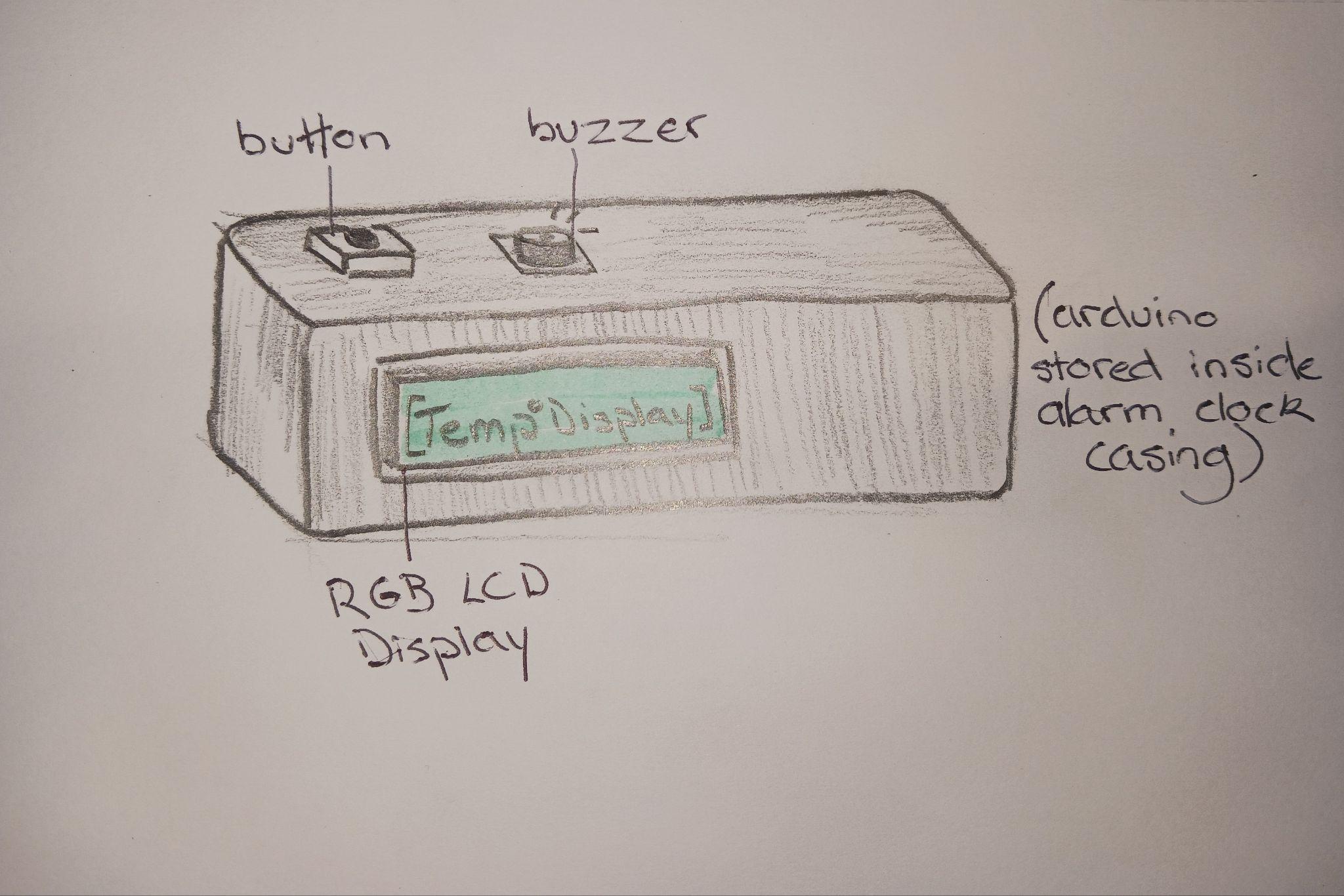
The last part of the solution involves the weather forecast. For this, OpenWeatherMap was used as the source. The code retrieves the information directly from the website and displays it on the LCD screen. For this project, the city for which the weather is shown is hardcoded to be Sligo.

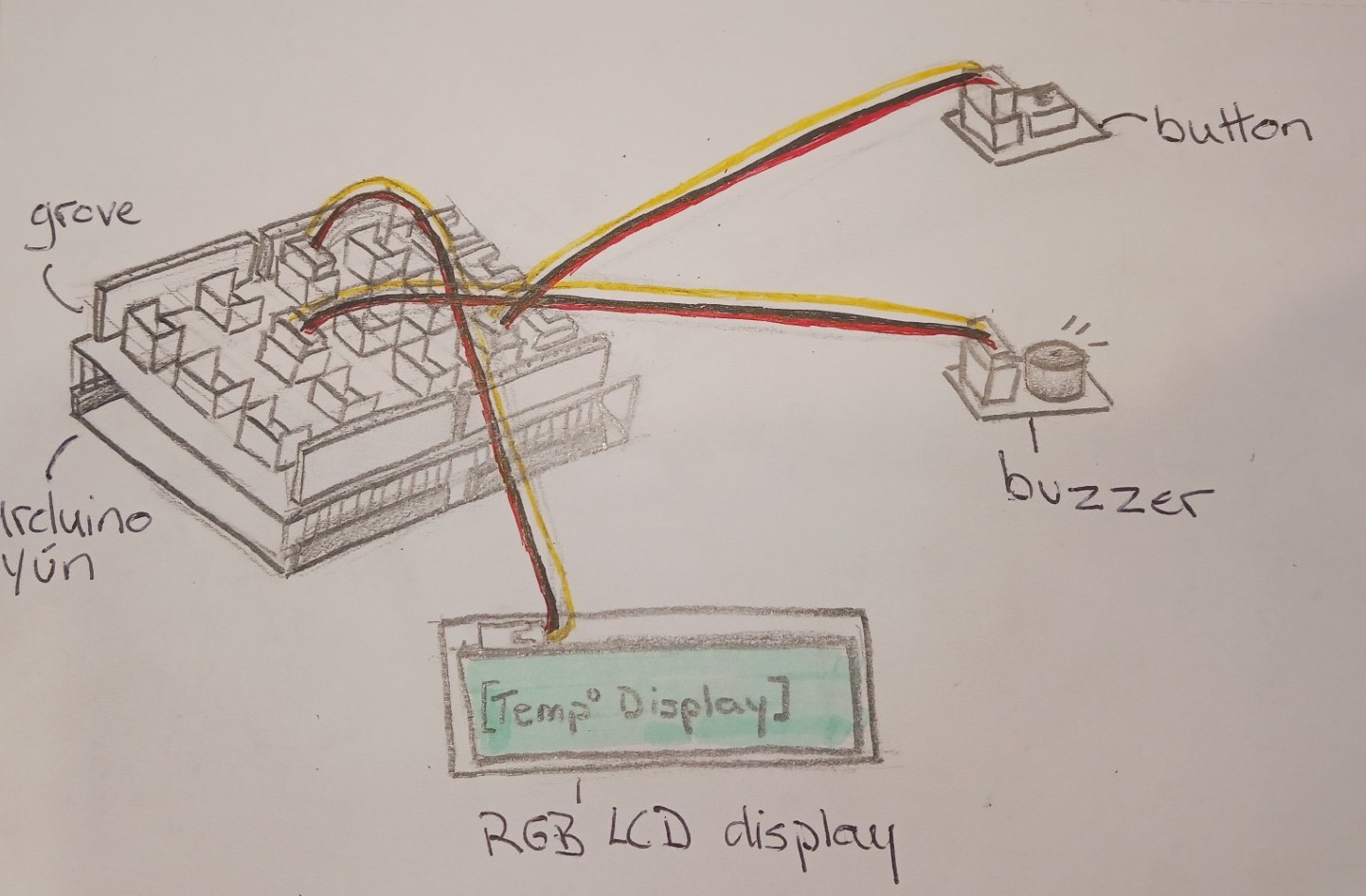
**List of Project Requirements**

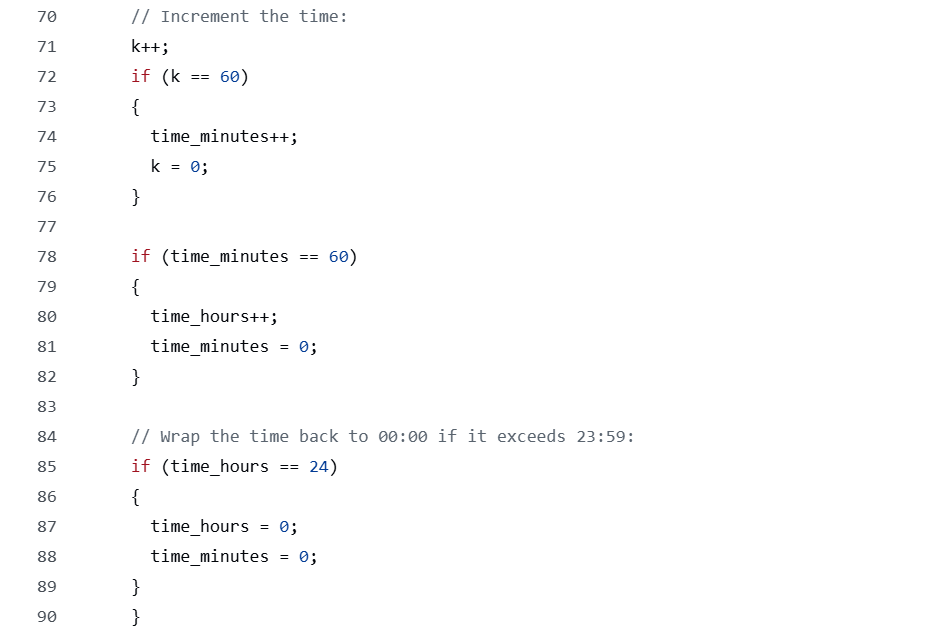
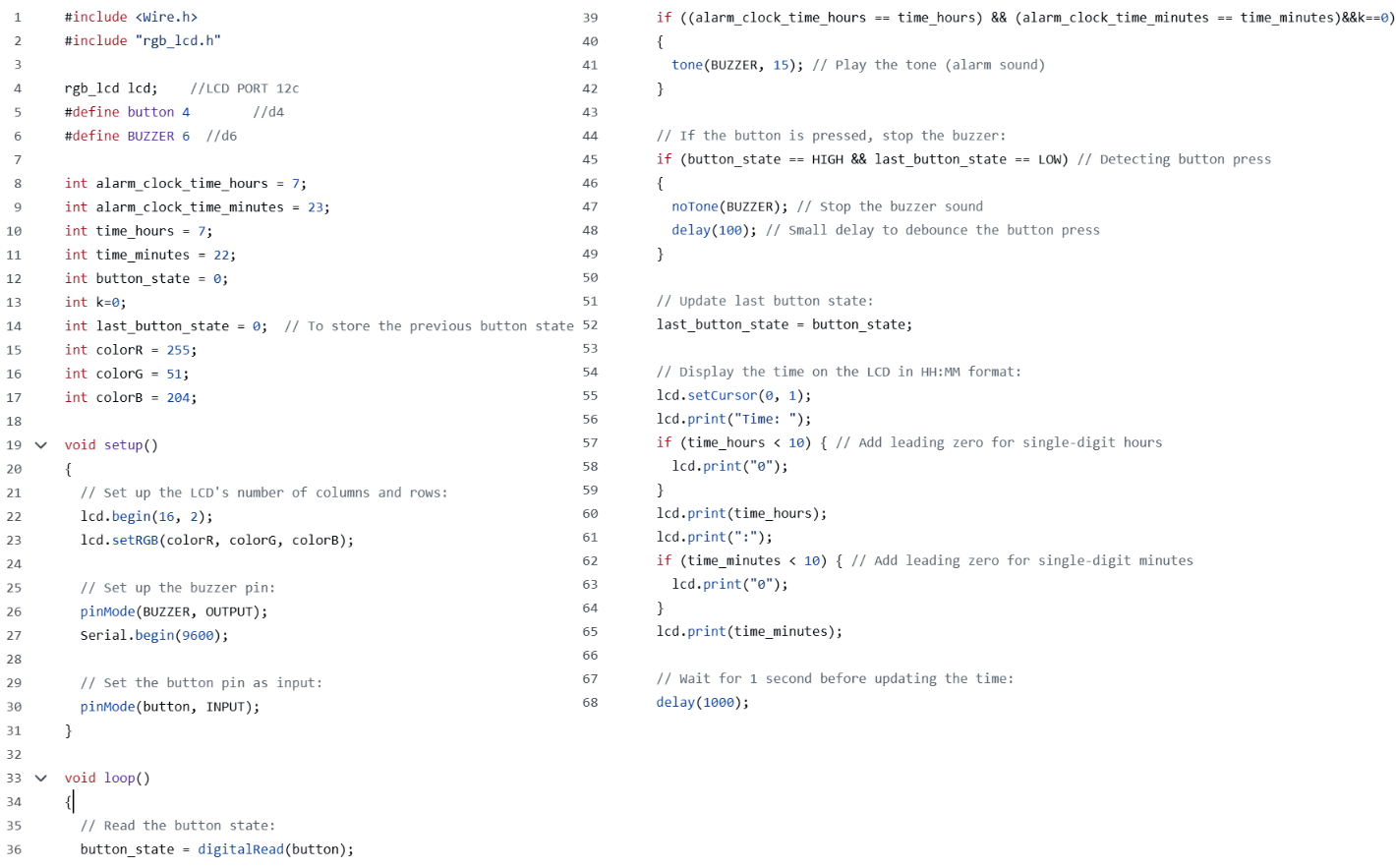
1. Buzzer goes off at the correct time
2. Alarm stops when button is pressed
3. Time is displayed on LCD
4. Weather forecast is displayed on LCD
5. LCD switches between displaying time and weather forecast
6. Correct weather details are displayed on LCD

**Initial Design**

**Sketch of Proposed Device:**



**Sketch of Proposed Hardware Setup:**

**Proposed Code:**

The Met Éireann weather API will be used in order to display the temperature/weather of the day to the user whenever the alarm goes off.

**Description of data generated, datasets or APIs used and how you plan to store and manage data**

For this project, we have one API being used, OpenWeatherAPI. API stands for “Application Programming Interface”, which is our code that states the set of rules and protocols, allowing different software applications to communicate with each other by exchanging data and functionality. It is a bridge between our Arduino alarm clock and the weather service we used.

In our API, we set up the LCD screen. This API requests weather data for Sligo from a weather service and then displays the temperature outside on our LCD screen, upon waking up and turning the alarm off.

At the current moment, we are not storing data from the user. In the future, we would like to have data collected and stored in a spreadsheet. Examples of such data would include how long it takes the user to turn off the alarm and more in-depth information about the weather.

**Implementation Plan**

**Equipment Needed**

==================================================================

* **Computer**: ATU Desktop.
* **Internet Connection**: Mobile Data, eduroam
* **IDE (Integrated Development Environment)**: https://www.arduino.cc/en/software.
* **Version Control System**: Git, GitHub, or Bitbucket for version control.
* **Testing Tools**: Any specific tools required for testing your project.

**List of Products**

==================================================================

Ø Grove – Buzzer - The buzzer will go off at the set time.

Ø Grove – Button - When the button is press, the alarm will stop

Ø Grove – LCD RGB Backlight - it will display the time e the weather forecast

Ø Arduino – Grove Real time clock – RTC, DS1307 - It will display the time, using the code through RTC MODEL.

Ø BBC Microbit

Ø API - [One Call API 3.0 - OpenWeatherMap](https://openweathermap.org/api/one-call-3)

[api 7ECBB0C17a028641084777724799ed61]

|  |  |
| --- | --- |
| Base Shield V2 | Grove – Buzzer |
| Grove – Button | Grove – LCD RGB Backlight |
| Arduino – Grove Real time clock – RTC, DS1307 | BBC Microbit; |
| API  7ECBB0C17a0286410847777254799ed61 | [One Call API 3.0 - OpenWeatherMap](https://openweathermap.org/api/one-call-3) |

**CODE 2 – Update code | Objective:**

1. Set the current time. 2. Activate the alarm.

3. Trigger the buzzer. 4. Stop the buzzer when the button is pressed.

==================================================================

**Libraries’s code -** Bridge.h and HttpClient.h: Used for HTTP requests.

Wire.h - Enables I2C communication.

RGB\_lcd.h - Controls the Grove LCD RGB Backlight.

* + - 1. The Grove LCD is initialized.
      2. The backlight is set to green (lcd.setRGB(1, 255, 100)).
      3. Displays "Weather Fetcher" for 2 seconds

ArduinoJson.h - Used to parse the JSON response from the OpenWeatherMap API.

* Requests weather data from OpenWeatherMap API

i. Builds a URL using city = "Sligo" and apiKey = "7ecbb0c17a028641084777254799ed61".

ii. Sends an HTTP GET request using HttpClient.

* Processes the API response

i. Reads and prints the response to the Serial Monitor.

ii. Parses the JSON response with ArduinoJson to extract the temperature.

iii. If deserialization fails, it waits for 60 seconds and retries.

1. Displays the temperature on the LCD
2. Repeats every 60 seconds.

**Testing approach**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case | Test Case Title | Requirement No. | Test Steps | Test Data/Screenshots | Expected Result | Test Pass/Fail | Tested By | Date |
| 1 | Buzzer goes off at the correct time | 1 | 1)Run code  2)Wait for buzzer to go off  3)Ensure buzzer goes off at time specified in code |  | Buzzer goes off at the correct time | Pass | Eliška & Shauna | 27/02/25 |
|
|
| 2 | Alarm stops when the button is pressed | 2 | 1)Run code  2)Wait for buzzer to go off  3)Press button  4)Ensure buzzer stops when button is pressed |  | Buzzer stops when the button is pressed | Pass | Eliška & Shauna | 27/02/2025 |
|
|
|
| 3 | Time is displayed on LCD | 3 | 1)Run code  2)Check LCD to see if time is displayed |  | Time is displayed on LCD | Pass | Eliška & Aoife | 6/3/2025 |
|
| 4 | Weather forecast is displayed on LCD | 4 | 1) Run code  2)Check LCD to see if weather data is displayed |  | Weather forecast is displayed on LCD when code is run | Pass | Eliška & Mansura | 6/3/2025 |
|
|
| 5 | LCD switches between displaying time and weather forecast | 5 | 1)Run code  2)Check LCD to see if time is displayed  3)Wait 5 seconds and check to see if LCD is displaying weather data  4)Wait another 5 seconds to see if time is displayed again |  | Weather forecast is displayed on LCD after 5 seconds and time is displayed on LCD after another 5 seconds before returning to displaying time | Pass | Eliška & Mansura | 6/3/2025 |
|
|
| 6 | Correct weather details are displayed on LCD | 6 | 1) Run code  2)Check that weather shows on LCD  3)Check that the weather details displayed are correct based on Open Weather API |  | Weather forecast is displayed on LCD  The weather data that is shown is correct according to the Open Weather API | Fail:  Incorrect details shown on LCD | Eliška & Silvana | 6/3/2025 |
|
|
|

**Security Analysis**

Up until this point in the development process of the project, there has not been much opportunity to add security measures to our alarm clock. This is because we are not currently collecting and storing user data in any way.

A level of security that we have included is a password for the Arduino Yún. This means that those who do not know the password do not have access to the alarm clock that we have created. It prevents people from making changes to the Yún such as what Wi-Fi network it is connected to and what the password is.

In the future, we plan on using two spreadsheets as part of this project. One spreadsheet will be used to store user data such as their tasks for the day and the other will be used to store data about the user such as how long it took them to turn off the alarm and what time they set the alarm for. We will protect these spreadsheets with passwords. This will prevent anyone gaining unauthorised access to the spreadsheets so that no one can see, use, alter, or steal this information about the user.

We will ensure that the devices we use to access the Yún, code, and spreadsheets are updated to the latest specifications and that we have anti-virus software installed. This will add another layer of security in protecting user data.

**Future improvements**

In the future, we hope to improve our alarm clock by using an RTC (Real Time Clock) module. This will mean that the current date and time will appear on the LCD screen and be used by the code. We will not have to hard code in a time and instead will be able to set an alarm and it will go off when it is actually that time. We ordered an RTC module online, but we are currently waiting for this module to arrive. When we have an RTC module we will have to change our code so that it works with the RTC module instead of the time we set for it.

We also hope to add a light to the alarm clock. The plan is to code the light so that it comes on at the same time as the sun rises. This will mimic natural light and make it easier for the person using the alarm to wake up. We plan to use the Met Éireann API to get access to data about what time the sun rises. The API only provides access to the weather data for the previous hour. This will not work as by the time we find out when the sun is rising it will be too late. Instead, we plan to find the time that the sun rose on the previous day and use this to code the light to turn on.

As it stands, we can only set the time for the alarm to go off within the code for the alarm clock. In the future we hope to be able to use buttons to set the alarm time. This would further reduce the screen time of the user as they will not need to access a laptop or phone to set the alarm but will be able to use the buttons instead.

We also plan to add a snooze button to the alarm so that the user will have the option to continue sleeping for another five minutes before the alarm goes off again.

We want to send some data about the user of the alarm clock to a spreadsheet. We will use PushingBox to do this. We want to collect data about things such as how long it takes the user to turn off the alarm, the weather forecast, the time the sun rises, the temperature, and if the alarm was snoozed and how many times it was snoozed. We think it would be interesting to see how factors such as the day of the week, weather, and the time the sun rises might affect how the user uses the alarm (such as whether they snooze the alarm or not) and whether it has any influence on how quickly they turn off the alarm.

Another idea we are exploring is to allow the user to create a spreadsheet of their weekly tasks and connect this to the alarm clock so that a message will appear on the LCD in the morning to remind them of their responsibilities for the day.