**Internet of Things: Project 1**

**Shane** **McDonnell**

**EMAIL ADDRESS**:S00272858@atu.ie

Krystian CHMIELAK

EMAIL:S00271020@atu.ie

**Links**

* **Trello Page:** https://trello.com/invite/b/67d24f2ee07c322472b0537f/ATTI5f86fd654f406a384f2e1b037c80451e30BFC77A/iot-projects
* **GitHub Team Page:** https://eur06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2FKrystian779%2FIOTProject2025&data=05%7C02%7CS00272858%40atu.ie%7Cb3b6814bde784ac9760208dd622c965d%7C4785554500bb4800a65fe79104ec0fc4%7C0%7C0%7C638774665825859687%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=2ZKnKuO8OIc7gK551uTP77t2%2B7kcKWoTicTRL%2FaF6fU%3D&reserved=0
* **Data Sources & Analysis Plan:** [Insert Link]

**Outline of the Problem to be Solved**

Many temperature sensors struggle with accuracy due to environmental interference, sensor limitations, and power fluctuations. The challenge is to design a temperature reader that provides precise and consistent readings in different environments, including indoor, outdoor, and industrial settings.

**Key Issues:**

1. **Sensor Accuracy & Calibration:** Temperature sensors can be affected by humidity, airflow, or incorrect placement, leading to inaccurate readings.
2. **Power Stability:** Inconsistent power supply can affect the performance of the temperature sensor, leading to fluctuating readings.
3. **Data Logging & Storage:** Temperature readings need to be recorded efficiently for later analysis. The challenge is deciding whether to store data locally (SD card) or remotely (cloud-based storage).
4. **Display & Accessibility:** Users need to access temperature data in real-time. Choosing an appropriate display method (LCD, OLED, web dashboard, or mobile app) is a challenge.
5. **Wireless Connectivity (Optional):** Implementing Wi-Fi or Bluetooth to send temperature readings remotely without affecting battery life.

**References**

1. https://forum.arduino.cc/t/how-accurate-is-my-temperature-sensor/648453
2. https://forum.arduino.cc/t/accurate-temperature-sensor/1054999

**Summary of the Project Solution**

### **Overview**

The proposed solution is an **Arduino-based temperature reader** designed to measure and display temperature readings accurately. The system leverages **temperature sensors, an Arduino microcontroller, and a user-friendly display interface** to provide real-time temperature monitoring. This project addresses the need for a **reliable, cost-effective, and easily deployable temperature monitoring system** for various applications, including home automation, industrial monitoring, and environmental data collection.

1. **Accurate Temperature Sensing**

Utilizes a **high-precision temperature sensor** such as the DHT11 which we are using for our project

Ensures **real-time data acquisition** with minimal response delay.

1. **User-Friendly Display and Output**

Temperature readings are displayed on an **LCD screen** or **OLED display**, ensuring clear visibility.

Optional integration with **LED indicators or buzzer alerts** for predefined temperature thresholds.

1. **Compact and Scalable Design**
   * The solution is **lightweight, portable, and easy to deploy** in various environments.
   * Can be enhanced with additional sensors or modules for **humidity, pressure, or IoT capabilities**.

**List of Project Requirements**

1. Working code
2. A display to show current temperature
3. Stable power supply
4. Must use Arduino
5. Must accurately display temperature
6. Must not use too much power

**Initial Design**

* **Proposed Code Design**

#include <Wire.h>

#include "rgb\_lcd.h"

rgb\_lcd lcd;

const int sensorPin = A0; // Grove Temperature Sensor connected to A0

const float BETA = 3950; // Beta value for the NTC thermistor

void setup() {

lcd.begin(16, 2); // Initialize the LCD

lcd.setRGB(100, 255, 0);

}

void loop() {

int analogValue = analogRead(sensorPin);

float resistance = (1023.0 / analogValue) - 1.0;

resistance = 10000.0 / resistance; // Convert to resistance

float temperatureC = 1.0 / (log(resistance / 10000.0) / BETA + 1.0 / 298.15) - 273.15;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Temp: ");

lcd.print(temperatureC);

lcd.print(" C");

// Displaying message to user if the temperature is fine or not

lcd.setCursor(0, 1);

if(temperatureC > 0 && temperatureC < 20 ){

lcd.setRGB(0, 255, 255);

lcd.print("Its Cold Out!")

}

else if (temperatureC < 0){

lcd.setRGB(0, 0, 255);

lcd.print("Its Freezing!")

}

else if (temperatureC > 20 && temperatureC < 30) {

lcd.setRGB(0, 255, 0);

lcd.print("Its Nice Outside");

}

else if(temperatureC > 30 && temperatureC < 40){

lcd.setRGB(255, 150, 0);

lcd.print("Wear Sunscreen!");

}

else {

lcd.setRGB(255, 0, 0);

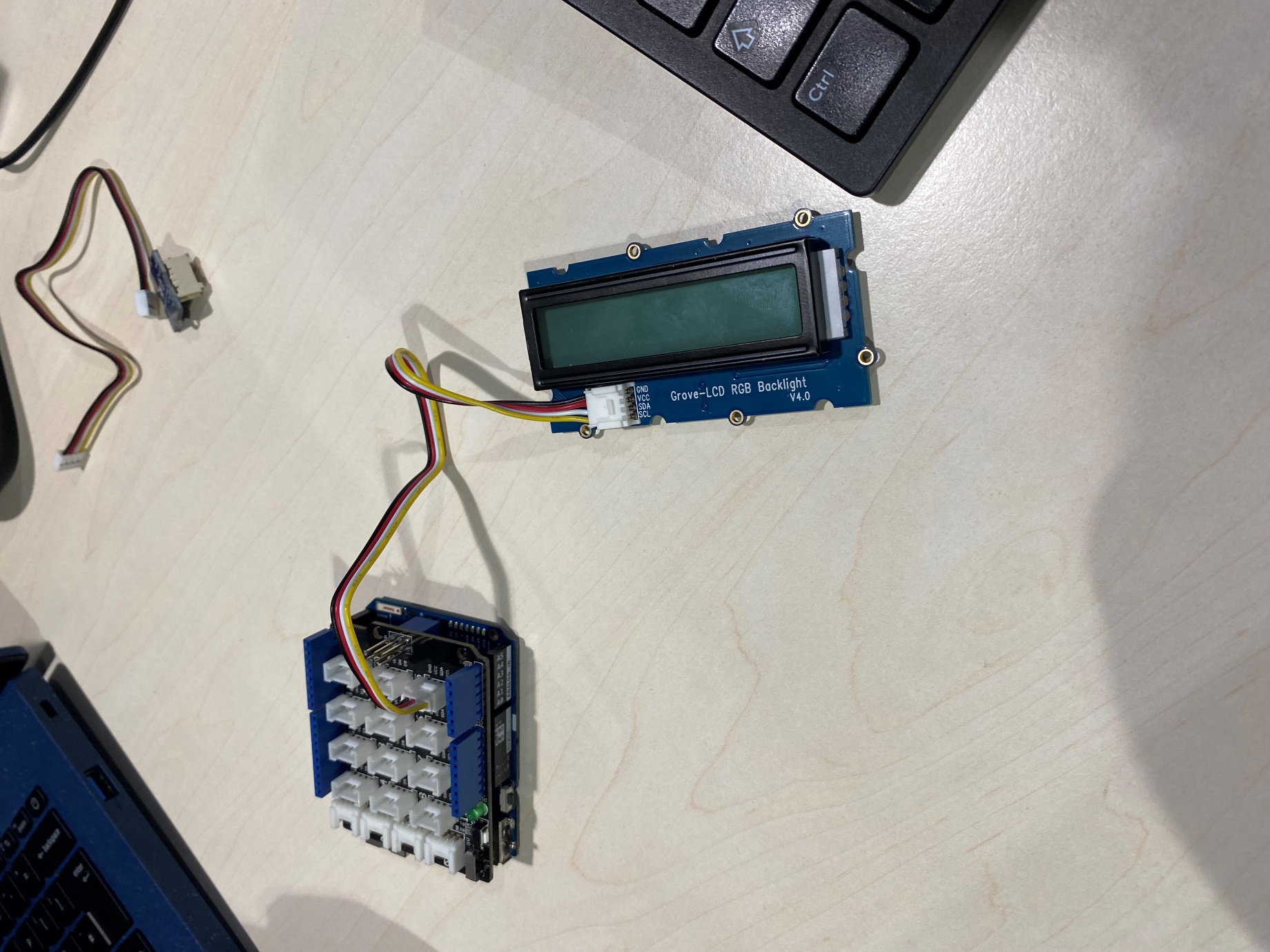
lcd.print("Stay indoors!");

}

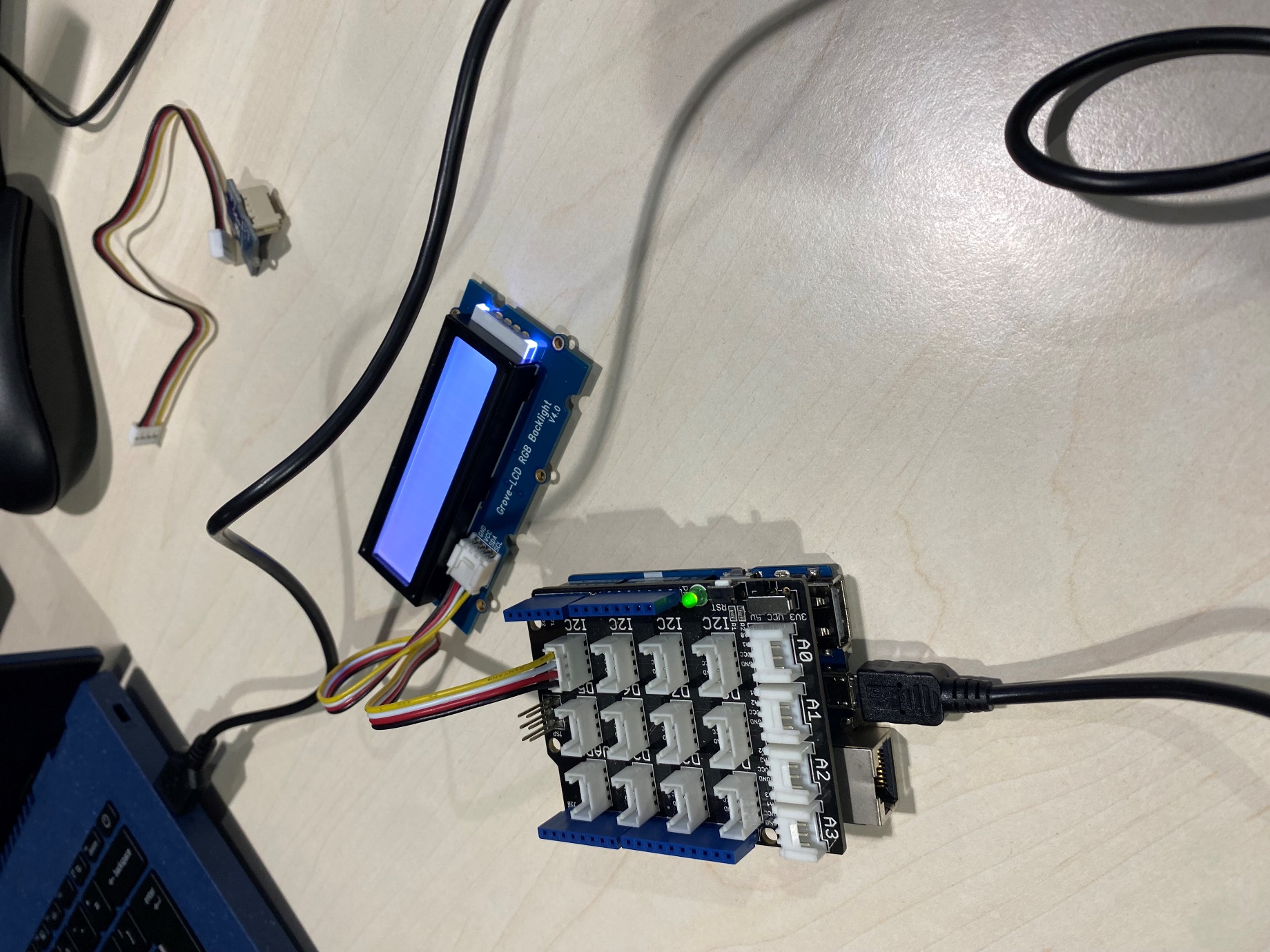
delay(2000);

}

* **Proposed Hardware Setup**







* **Description of APIs & Data Processing**

An **API (Application Programming Interface)** is a set of protocols and tools that allow different software applications to communicate with each other.

**Description of Data & Data Management Plan**

* **Datasets or APIs Used**

We used Cloud apis to store data online

* **Storage & Management Strategy**

We store data using a cloud api

**Implementation Plan**

* Equipment Needed

Arudino

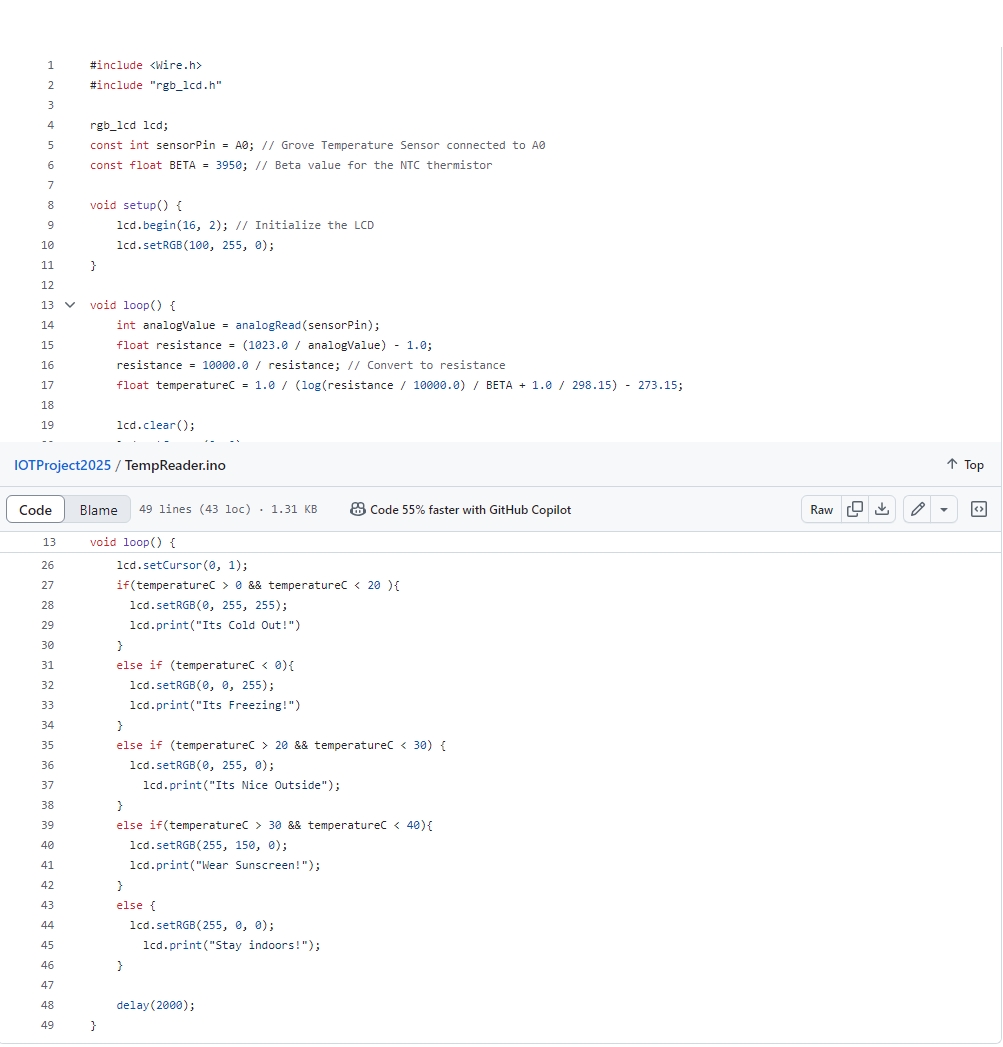
Grove-LCD RGB backlight

Male to female and male to male cables

* **APIs to be Used**

**Cloud api to store data**

* **Code Samples**



**Testing Approach**

* **Planned Software & Hardware Testing**

Me and Krystian test the software and hardware using the Arduino app and the Arduino itself. We would often find that we would have issues uploading our code to the arduino

* **Evidence of Tests Conducted**



**Security Analysis**

**Access Control**: Limit physical access to the Arduino board by placing it in a locked enclosure.

**Tamper Detection**: Use sensors (e.g., microswitches or tilt sensors) to detect unauthorized access attempts.

**Power Surge Protection**: Protect the board from electrical surges with capacitors and voltage regulators.

**Future Improvements & Next Steps**

1. Improvements to Hardware  
Switch to high-precision sensors (DS18B20, DHT22).  
Install sensors for air quality, pressure, or humidity.  
Use sleep modes to maximize power usage.

2. Enhancements in Software and Security  
Apply data smoothing techniques (Kalman filters, moving averages).  
Secure data transfer (MQTT with TLS, AES encryption).  
Use an app or web dashboard to enable remote configuration.

3. Cloud Integration & Connectivity  
Send information to Thingspeak, AWS IoT, or Firebase.  
Include notifications and alerts (SMS, email, and app).  
Reduce your reliance on the cloud by using edge computing.

4. Reliability and Testing  
Perform extensive field testing in various settings.  
Put a watchdog timer in place for automatic recovery.  
For data validation, use anomaly detection and checksums.

5. Completed Product & Implementation  
Create a robust, water-resistant enclosure.  
Examine target industries and market potential.  
PCB design should be optimized for mass manufacturing.