Smart Cold Storage system

A project report submitted to Dr Rekha D, SCOPE, VIT Chennai
In partial fulfilment of the requirements of the course
ECE3501 - IoT Fundamentals

DONE BY
M. MURALIKRISHNAN (22MIS1177)
SONAA RAJAGOPAL (22MIS1028)

M. Tech. (Integrated) Software Engineering, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai – 600127

Table of Contents

Chapter 5: Implementation and Coding

Chapter 6: Results and Discussion

Chapter 7: Conclusion

25

Abstract

The **Smart Cold Storage System** is an **Internet of Things (IoT)-based automation solution** designed to efficiently monitor, regulate, and maintain optimal environmental conditions for perishable goods that are highly sensitive to temperature variations. The system integrates multiple components including a **DHT11 temperature and humidity sensor**, an **ESP32 microcontroller**, and a **Peltier module** that collaboratively manage the internal thermal conditions in real time. The **DHT11** continuously measures the temperature and humidity levels within the storage unit, and these readings are processed by the **ESP32** to determine whether the environment remains within the user-defined safe range.

When the monitored temperature exceeds the pre-set threshold, the **Peltier cooling module** is automatically activated to reduce the internal temperature. Simultaneously, a **buzzer** and a **red LED indicator** are triggered to provide immediate visual and auditory alerts, ensuring timely awareness of critical temperature deviations. The system also incorporates an **LCD display module** that provides on-site visualization of the current temperature and humidity readings for quick reference and manual supervision if needed.

Beyond local monitoring, the system leverages the **Blynk IoT cloud platform** to enable **remote data visualization and control**. The architecture distinguishes between two levels of user access: an **admin dashboard**, which allows authorized personnel to modify threshold parameters and control system behavior, and a **client dashboard**, which offers real-time monitoring capabilities without control privileges. This tiered access structure ensures both operational flexibility and system security.

The proposed Smart Cold Storage System demonstrates the effective integration of **embedded sensing, IoT connectivity, and intelligent control algorithms** to maintain ideal storage conditions with minimal human intervention. This solution can be applied in various domains such as **food preservation, pharmaceuticals, agricultural storage, and laboratory environments**, where maintaining strict temperature and humidity control is critical. Overall, the system provides a scalable and cost-effective framework for modernizing traditional cold storage management using IoT technologies.

Introduction

In today's fast-paced and technology-driven world, maintaining optimal environmental conditions for temperature-sensitive goods is a major challenge faced by industries such as food preservation, pharmaceuticals, agriculture, and biotechnology. Traditional cold storage systems often rely on manual monitoring and control, which can lead to inefficiencies, human error, and potential spoilage of stored items. With the growing demand for quality assurance and sustainability, there is an increasing need for intelligent, automated systems that can ensure precise temperature regulation and real-time monitoring without continuous human supervision.

The **Smart Cold Storage System** addresses this need by leveraging the **Internet of Things (IoT)** to automate the process of temperature and humidity monitoring, control, and alerting. IoT technology enables devices to collect and exchange data over the internet, allowing users to remotely monitor and manage systems from virtually anywhere. By integrating sensors, actuators, and cloud connectivity, the proposed system ensures that the cold storage environment remains within predefined limits, thereby enhancing reliability, safety, and efficiency.

At the core of the system lies the **ESP32 microcontroller**, which serves as the main processing unit. It interfaces with a **DHT11 sensor** to acquire real-time temperature and humidity data and a **Peltier module** that functions as a thermoelectric cooling device to maintain the desired temperature. When the temperature exceeds a user-defined threshold, the ESP32 activates the Peltier module to initiate cooling. Simultaneously, a **buzzer** and **red LED** alert system notify users of the temperature breach, ensuring immediate awareness and response. The system also includes an **LCD display** that presents current environmental data for on-site monitoring.

A key feature of the system is its **integration with the Blynk IoT cloud platform**, which facilitates remote monitoring and control through a user-friendly dashboard accessible via smartphones or computers. This connectivity allows users to observe temperature and humidity trends in real time and adjust threshold settings as necessary. To ensure operational security and flexibility, the system is designed with two user interfaces: an **admin dashboard** for control and configuration, and a **client dashboard** for monitoring purposes only.

By combining automation, sensor-based feedback, and cloud connectivity, the Smart Cold Storage System minimizes human intervention, reduces energy consumption, and enhances the efficiency of cold storage management. This system represents a practical and scalable solution for modern industries seeking to improve storage reliability and maintain product quality through smart technology.

Literature Review and Related Works

In recent years, numerous researchers had explored the application of **Internet of Things (IoT)** technology in environmental monitoring and smart cold storage systems. The increasing demand for automation, energy efficiency, and real-time data accessibility led to several innovative designs integrating sensors, cloud platforms, and microcontrollers.

Sharma *et al.* [14] developed an IoT-based temperature and humidity monitoring system using the **NodeMCU microcontroller** and the **Blynk IoT platform**. Their system successfully monitored real-time data through cloud integration, demonstrating how low-cost IoT hardware could replace traditional monitoring methods. Similarly, Kumar and Singh [7] proposed a smart refrigeration model that automatically controlled cooling using IoT and provided remote feedback to users, proving the feasibility of intelligent temperature regulation.

Joseph and George [6] worked on a **warehouse environment monitoring system** using distributed DHT11 sensors connected to an IoT cloud. Their system achieved accurate real-time measurements over large storage areas, highlighting the importance of reliable data acquisition and sensor calibration in temperature-sensitive environments. In a related study, Hossain and Chowdhury [5] utilized **thermoelectric cooling (Peltier modules)** in a portable cold storage system, showcasing the potential of solid-state cooling technologies for compact and energy-efficient designs.

To further enhance performance, several researchers explored **data analytics and automation algorithms**. Chaudhary *et al.* [2] presented an IoT-based refrigeration system enhanced with **predictive analytics**, which optimized energy consumption by forecasting temperature variations. Similarly, Mishra and Patel [9] designed a **cold chain monitoring network** for perishable food supply management using the **IEEE IoT architecture**, ensuring that temperature deviations triggered instant alerts through cloud connectivity.

The concept of remote access and visualization through **IoT dashboards** was a key advancement in previous works. Rao *et al.* [13] implemented a real-time greenhouse monitoring system integrated with the **Blynk IoT platform**, allowing users to observe temperature and humidity changes from any location. Likewise, Li *et al.* [8] and Das and Roy [3] developed **cloud-based warehouse systems** where users could not only visualize data but also control connected devices remotely. These studies proved that integrating microcontrollers such as **ESP32** or **NodeMCU** with cloud interfaces significantly improved monitoring reliability and accessibility.

Patil *et al.* [11] and Nair and Thomas [10] examined **thermoelectric cooling systems** driven by IoT-based automation. Their prototypes demonstrated that Peltier modules could effectively maintain stable temperatures for small enclosures, provided that sufficient power control mechanisms were in place. Rahman *et al.* [12] expanded on this idea by designing an adaptive cooling mechanism for food preservation that adjusted cooling intensity based on stored product characteristics and environmental feedback.

In addition, Ali *et al.* [1] and Gupta and Tiwari [4] contributed to **low-cost IoT-based monitoring systems** for cold storage and pharmaceutical applications. Their work emphasized the need for **affordable, energy-efficient, and scalable** systems suitable for small businesses and local supply chains. Meanwhile, Sundar and Velmurugan [15] combined **IoT technology with data analytics** to design an intelligent cold storage prototype aimed at improving agricultural product shelf life, which inspired the conceptual direction of this present system.

Although these previous studies provided valuable insights into IoT-based environmental management, several limitations persisted. Many systems focused solely on **data monitoring** without implementing **real-time control mechanisms** such as automatic activation of cooling devices. Some utilized **costly hardware** and lacked multi-user functionality. Others provided **single-user dashboards** without distinguishing between **administrative** and **client** privileges. Furthermore, few studies explored the **use of low-cost materials** for physical prototyping, which could make such systems accessible for educational or experimental applications.

The **Smart Cold Storage System** developed in this project addressed these limitations by integrating a **DHT11 sensor** for continuous temperature and humidity sensing, an **ESP32 microcontroller** for real-time decision-making, a **Peltier module** for cooling control, and **Blynk IoT integration** for cloud-based monitoring and management. The system featured both **admin** and **client dashboards**, supported **real-time threshold adjustments**, and used **cost-effective materials** such as cardboard and plastic enclosures for demonstration purposes. In doing so, it extended the capabilities of prior research by combining automation, cost-effectiveness, and remote accessibility into a unified, functional prototype.

System Design and Architecture

The **Smart Cold Storage System** was designed to provide automatic control and remote monitoring of temperature and humidity within a storage environment. The architecture combined **hardware components**, **IoT connectivity**, and **cloud integration** to form a fully functional and responsive system. The following subsections describe the overall design, functional flow, and working principle of the system.

3.1 System Overview

The system architecture consisted of three main layers:

1. **Sensing Layer** – responsible for collecting environmental data through sensors.

2. **Processing Layer** – responsible for analyzing the sensor readings and making control decisions.

3. **Cloud and User Interface Layer** – responsible for data visualization, remote access, and user interaction through dashboards.

At the core of the system, the **ESP32 microcontroller** served as the central processing unit that interfaced with the sensors, actuators, and cloud services. The **DHT11 sensor** continuously measured temperature and humidity within the cold storage chamber and transmitted the readings to the ESP32 for processing. Based on the data, the ESP32 determined whether to activate the **cooling module** and **alert mechanisms**.

3.2 Architecture Diagram

Fig. 1 below represents the overall architecture of the Smart Cold Storage System.

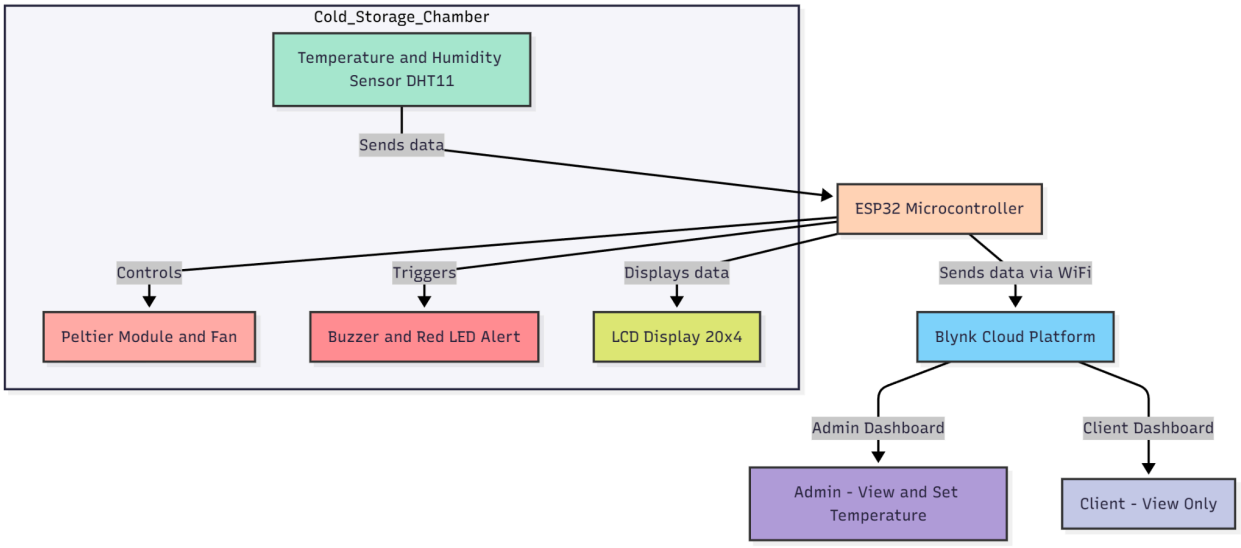


Fig. 1: *Architecture diagram of the Smart Cold Storage System.*

The diagram illustrated the relationship between the hardware modules, cloud components, and user dashboards. The architecture depicted how data flowed bidirectionally between the sensing devices and the Blynk IoT platform through the ESP32.

3.3 Flow of the Architecture

The operational flow of the Smart Cold Storage System was as follows:

1. **Data Acquisition:**
   The system started with the **DHT11 sensor**, which continuously measured the surrounding temperature and humidity inside the prototype cold storage unit.

2. **Data Processing:**
   The collected data were transmitted to the **ESP32 microcontroller**, which compared the measured temperature to a **threshold value** previously set by the administrator via the **Blynk IoT dashboard**.

3. **Decision Making:**

   - When the temperature was **less than or equal to the threshold**, the system remained **idle**, keeping the **Peltier module**, **cooling fan**, **buzzer**, and **LED indicator** turned **off**.

   - When the temperature **exceeded the threshold**, the ESP32 activated the **Peltier cooling module** and **fan** to lower the temperature. Simultaneously, the **buzzer**

and **red LED** were turned **on** to provide a local alert signal indicating a rise in temperature.

4. **Local Display and Feedback:**
   The **LCD screen** displayed the real-time **temperature**, **humidity**, and **system status** (e.g., “Cooling Active,” “Normal,” or “Threshold Exceeded”) to provide immediate visual feedback to nearby users.

5. **Cloud Connectivity:**
   The ESP32 continuously sent the live temperature and humidity readings to the **Blynk Cloud Platform** using Wi-Fi connectivity. The data were logged and displayed in real-time on the Blynk dashboard.

6. **User Interaction and Dashboard Access:**

    - The **Admin Dashboard** allowed the administrator to modify the **temperature threshold** and observe system performance remotely.

    - The **Client Dashboard** provided a **read-only view**, showing temperature and humidity trends but without control privileges.

This flow ensured that the system functioned both **autonomously** (through local automation) and **remotely** (through cloud supervision).

3.4 Functional Components in the Architecture

Each hardware and software component in the system played a distinct role within the architecture:

- **ESP32 Microcontroller:**
  Served as the system’s brain, executing logic operations, managing sensor inputs, actuator outputs, and communication with the cloud.

- **DHT11 Sensor:**
  Captured real-time data for temperature and humidity. Its low-cost and simple digital interface made it ideal for continuous monitoring.

- **Peltier Module and Fan:**
  Functioned as the thermoelectric cooling system. The Peltier module cooled the chamber when activated, while the fan dissipated heat from the hot side to maintain efficiency.

- **Buzzer and Red LED:**
  Acted as local alert mechanisms, signaling when the internal temperature rose above the threshold value.

- **LCD Display:**
  Displayed current readings and system status locally, ensuring transparency of operation.

- **Blynk IoT Platform:**
  Provided cloud storage, live data visualization, and dashboard control. Enabled real-time updates accessible via smartphones or computers.

- **Admin and Client Dashboards:**
  Differentiated access ensured security and management control. The admin could alter parameters, while the client could only monitor system performance.

3.5 Summary of Operation

The Smart Cold Storage System operated as a **closed-loop IoT-based control system**. It automatically maintained the desired temperature range using sensor feedback and actuator control, while simultaneously offering remote monitoring through the Blynk IoT platform. The integration of both **hardware automation** and **cloud-based analytics** made the system reliable, efficient, and user-friendly for both administrators and observers.

4. Working Principle and System Implementation

The **Smart Cold Storage System** was implemented as a small-scale prototype designed to automatically monitor and regulate temperature and humidity in a controlled environment. The system's working principle relied on the integration of sensors, actuators, and IoT connectivity under the control of the **ESP32 microcontroller**. The implementation process involved constructing the circuit, connecting the required components, and testing the automated temperature regulation mechanism.

4.1 Circuit Overview

The circuit of the Smart Cold Storage System was built around the **ESP32 microcontroller**, which acted as the central control unit. It processed data received from the **DHT11 sensor**, compared it against a predefined threshold value, and made decisions to activate or deactivate the **Peltier cooling module**, **fan**, and **alert mechanisms** accordingly.

The main purpose of the circuit was to continuously monitor real-time temperature and humidity levels within the cold storage chamber and to maintain optimal conditions automatically. The circuit also supported cloud communication via the **Blynk IoT platform**, enabling users to remotely view live readings and modify the temperature threshold from the **Admin dashboard**.

The system consisted of the following interconnected components:

- **DHT11 sensor** for temperature and humidity measurement
- **Peltier module (TEC1-12706)** for thermoelectric cooling
- **Motor driver (L298N/L293D)** to control the cooling fan
- **Buzzer** and **LED** for local alert notifications
- **16×2 LCD display** for real-time data visualization
- **ESP32 microcontroller** for processing and cloud connectivity
- **Breadboard** for prototyping and assembling the circuit
- **Jumper wires** (male-to-male, male-to-female, female-to-female) for electrical connections
- **Dual power supply** (5V for logic components, 12V for the Peltier and fan)

- **Cardboard box** for housing the prototype system
- **Ice-cream box** as a sample storage item representing perishable goods

All modules were connected through the breadboard using color-coded jumper wires. The ESP32 read data from the DHT11 sensor, compared it with the user-defined threshold, and controlled a relay that switched the Peltier module on or off. The same logic also triggered the **buzzer** and **LED** when the temperature exceeded the limit. Simultaneously, real-time data and system status were transmitted to the **Blynk cloud** for remote monitoring.

The overall signal flow of the system was as follows:

DHT11 Sensor → ESP32 → Peltier → Fan (via Motor Driver) → Buzzer + LED → LCD → Blynk Dashboard

4.2 Components Description

4.2.1 ESP32 Microcontroller

The **ESP32** served as the core processing unit of the system. It handled input from the DHT11 sensor, performed logic operations to determine system state, and sent control signals to the cooling and alert components. It also enabled wireless communication with the **Blynk IoT platform** for cloud-based data transmission.

Key Features:

- Dual-core processor with high performance
- Built-in Wi-Fi and Bluetooth connectivity
- Multiple GPIO pins for interfacing sensors and actuators
- Low power consumption suitable for continuous operation

4.2.2 DHT11 Temperature and Humidity Sensor

The **DHT11** sensor was responsible for continuously measuring temperature and humidity inside the cold storage unit. It provided digital output to the ESP32 at fixed intervals. The sensor's reliability and affordability made it well-suited for small-scale environmental monitoring.

Key Features:

- Digital signal output for easy interfacing
- Temperature range: 0–50°C; Humidity range: 20–90%
- Accuracy: ±2°C (temperature), ±5% (humidity)
- Compact and energy efficient

4.2.3 Peltier Module (TEC1-12706)

The **Peltier module** acted as the thermoelectric cooling element. When powered, it produced a temperature gradient across its two sides—one side becoming cold (for cooling the chamber) and the other becoming hot (for heat dissipation). It was connected to the ESP32 via a relay switch for automated control.

Key Features:

- Solid-state cooling (no moving parts)
- Compact and efficient for prototype refrigeration
- Fast thermal response

4.2.4 Motor Driver

A **motor driver** (L298N or L293D) was used to control a DC cooling fan attached to the hot side of the Peltier module. It allowed the fan to operate safely by amplifying the current from the ESP32's low-power output pins and protecting the microcontroller from overcurrent.

Key Features:

- Bidirectional control for DC motors

- Supports PWM (Pulse Width Modulation) for speed control
- Integrated protection against current spikes

4.2.5 Buzzer and LED

The **buzzer** provided an audible warning while the **LED** gave a visual indication whenever the system detected a temperature higher than the threshold value. Both were connected directly to GPIO pins of the ESP32 and were activated simultaneously for immediate alerts.

Key Features:

- Fast activation and low current draw
- Clear and reliable notification system

4.2.6 16×2 LCD Display

The **LCD display** was used to present real-time temperature, humidity, and system status messages (e.g., *Cooling ON*, *Normal Temperature*, *Alert Mode*). It interfaced with the ESP32 via the **I2C communication protocol**, minimizing the number of required connections.

Key Features:

- Energy-efficient display for continuous operation
- Easy integration with microcontrollers
- Real-time visual feedback

4.2.7 Breadboard and Jumper Wires

A **breadboard** was used for assembling the prototype circuit without soldering. **Jumper wires** (M–M, M–F, F–F) provided all electrical connections among modules. Organized color-coded wiring (red for power, black for ground, and others for signals) ensured clarity and reduced wiring errors.

4.2.8 Power Supply

A **regulated dual power supply** was used in the setup:

- **5V DC** for ESP32, LCD, and logic-level components
- **12V DC** for the Peltier module and fan

A common ground was maintained between both voltage levels to ensure stable operation and accurate sensor readings.

4.2.9 Cardboard Box (Prototype Enclosure)

A **cardboard box** was used as the prototype housing for all components. The Peltier module was mounted through one wall, with the cold side facing inside and the hot side facing outward. The fan and heatsink were attached externally for efficient heat dissipation. Openings were made for the LCD, LED, and buzzer for easy visibility and access.

4.2.10 Ice-Cream Box (Sample Item)

An **ice-cream box** was used to simulate a real stored product inside the cold storage. The DHT11 sensor was positioned close to the ice-cream box (without direct contact) to measure the surrounding air temperature and humidity. This demonstrated the system's ability to preserve perishable goods effectively.

4.3 Circuit Operation Summary

During operation, the **DHT11 sensor** continuously measured temperature and humidity within the storage enclosure. The **ESP32** processed this data and displayed the readings on the **LCD screen**, while simultaneously transmitting the same data to the **Blynk cloud**.

Whenever the measured temperature exceeded the **user-defined threshold**, the ESP32 activated the **relay**, powering the **Peltier module** and **cooling fan** to lower the internal temperature. Simultaneously, the **buzzer** and **LED** were triggered, alerting users of an abnormal rise in temperature.

Once the temperature dropped back below the threshold, the system automatically deactivated the cooling module, fan, and alert mechanisms. This automated feedback loop ensured that the temperature remained stable with minimal human intervention.

4.4 Assembly Setup

All electronic components were mounted on the **breadboard** and placed securely on top of the **cardboard enclosure**. The **ESP32**, **DHT11**, and other modules were fixed using adhesive tape, while the **Peltier module** and **heatsink** were attached to the wall of the cardboard box using **thermal paste** for improved heat transfer.

Inside the box, the **ice-cream container** served as a simulated stored item, with the **DHT11 sensor** positioned nearby to record environmental changes. The overall assembly successfully demonstrated a compact, functional **IoT-based smart cold storage prototype**, combining automation, real-time data monitoring, and cost-effective implementation.

4.5 Summary

In summary, the **Smart Cold Storage System** functioned as an **automated closed-loop control system**, maintaining stable environmental conditions using sensor feedback, actuator response, and cloud-based monitoring. The implementation validated that low-cost IoT components could effectively automate temperature regulation and enhance the management of perishable goods.

5. Program Implementation and Code Explanation

Code:

```
#define BLYNK_TEMPLATE_ID "TMPL3RdpX7u9J"
#define BLYNK_TEMPLATE_NAME "Smart Cold Storage"
#define BLYNK_AUTH_TOKEN "bKzHUdwHeS3nwM2IhVDrq1OCIoUxK6tS"

#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

// ==================== WiFi Credentials ====================
char ssid[] = "MURALI";       // your WiFi SSID
char pass[] = "MURALIKRISHNAN";       // your WiFi password

// ==================== DHT CONFIG ====================
#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// ==================== LCD CONFIG ====================
LiquidCrystal_I2C lcd(0x27, 20, 4);

// ==================== MOTOR (FAN/PELTIER) CONFIG ====================
#define IN3 32
#define IN4 33
#define ENB 18
float setTemp = 22.0;  // Default threshold temperature

// ==================== BUZZER CONFIG ====================
#define BUZZER_PIN 25   // Signal pin for buzzer (3-pin buzzer OUT pin)

// ==================== Blynk Virtual Pins ====================
// V2 -> Temperature Display
// V3 -> Humidity Display
// V4 -> Set Temperature (Slider)
```

```
// V6 -> Cooling ON/OFF LED
// V7 -> Alarm LED (High Temp Alert)

// ==================== BLYNK SLIDER CONTROL ====================
BLYNK_WRITE(V4) {
  setTemp = param.asFloat(); // Update set temperature from Blynk slider
  Serial.print("📱 Set Temp updated from Blynk: ");
  Serial.print(setTemp);
  Serial.println(" °C");
}

// ==================== SETUP ====================
void setup() {
  Serial.begin(115200);
  Serial.println("Smart Cold Storage System Starting...");

  // Initialize components
  dht.begin();
  lcd.init();
  lcd.backlight();

  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  // Initialize states
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, 0);
  digitalWrite(BUZZER_PIN, LOW);

  lcd.setCursor(0, 0);
  lcd.print("Connecting WiFi...");

  // Connect to Blynk
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

  lcd.clear();
  lcd.setCursor(0, 0);
```

```
  lcd.print("Cold Storage Ready");
  delay(2000);
}

// ===================== MAIN LOOP =====================
void loop() {
  Blynk.run();

  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println(" DHT11 Sensor Error!");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Sensor Error!");
    delay(2000);
    return;
  }

  // Display on Serial
  Serial.print("Temp: ");
  Serial.print(temperature);
  Serial.print(" °C |  Hum: ");
  Serial.print(humidity);
  Serial.print(" % | Target: ");
  Serial.println(setTemp);

  // Display on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(temperature, 1);
  lcd.print((char)223);
  lcd.print("C  S:");
  lcd.print(setTemp, 1);
  lcd.print("C");

  lcd.setCursor(0, 1);
  lcd.print("H:");
```

```
  lcd.print(humidity, 1);
  lcd.print("%");

  // Send data to Blynk
  Blynk.virtualWrite(V2, temperature);
  Blynk.virtualWrite(V3, humidity);

  // Cooling + Alarm + Buzzer control
  if (temperature >= setTemp) {
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, 255);
    Serial.println("Cooling ON");
    lcd.setCursor(0, 2);
    lcd.print("Cooling: ON ");
    Blynk.virtualWrite(V6, 1);

    // 🔴 Turn ON Alarm LED + Buzzer
    Blynk.virtualWrite(V7, 255);
    lcd.setCursor(0, 3);
    lcd.print(" ALERT: HIGH TEMP!");
    digitalWrite(BUZZER_PIN, LOW);

  } else {
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, 0);
    Serial.println(" Cooling OFF");
    lcd.setCursor(0, 2);
    lcd.print("Cooling: OFF");
    Blynk.virtualWrite(V6, 0);

    // 🟢 Turn OFF Alarm LED + Buzzer
    Blynk.virtualWrite(V7, 0);
    lcd.setCursor(0, 3);
    lcd.print("Temp Normal    ");
    digitalWrite(BUZZER_PIN, HIGH);

  delay(2000);
}
```

Code Explanation

The code for the **Smart Cold Storage System** was developed using the **Arduino IDE** and programmed into the **ESP32 microcontroller**. It integrated multiple modules — DHT11 sensor, LCD display, motor driver, buzzer, and Blynk IoT platform — to achieve real-time monitoring and automated temperature regulation.
The program logic was divided into several sections: library inclusion, configuration, initialization, and main loop operation.

5.1 Library Inclusion

```
#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#include <DHT.h>
```

The above libraries were included to provide support for:

- **WiFi.h:** handled wireless network communication for the ESP32.
- **BlynkSimpleEsp32.h:** allowed cloud-based connectivity through the Blynk IoT platform.
- **Wire.h** and **LiquidCrystal_I2C.h:** enabled I2C-based communication with the 16×2 LCD display.
- **DHT.h:** provided predefined functions to read temperature and humidity data from the DHT11 sensor.

5.2 Wi-Fi and Blynk Configuration

```
#define BLYNK_TEMPLATE_ID "TMPL3RdpX7u9J"

#define BLYNK_TEMPLATE_NAME "Smart Cold Storage"

#define BLYNK_AUTH_TOKEN "bKzHUdwHeS3nwM2IhVDrq1OCIoUxK6tS"



char ssid[] = "MURALI";

char pass[] = "MURALIKRISHNAN";
```

This section defined the **Blynk template credentials** (template ID, name, and authentication token) and **Wi-Fi credentials** (SSID and password).
These parameters authenticated the ESP32 to connect both locally (via Wi-Fi) and remotely (to the Blynk cloud server).

Through this, real-time data such as **temperature, humidity, and system state** were continuously uploaded to the Blynk dashboard.

5.3 Sensor and Hardware Pin Configuration

```
#define DHTPIN 4

#define DHTTYPE DHT11

#define IN3 32

#define IN4 33

#define ENB 18

#define BUZZER_PIN 25
```

- The **DHT11 sensor** was connected to digital pin **4** of the ESP32.
- The **motor driver** controlling the **Peltier and cooling fan** was interfaced through pins **32 (IN3)**, **33 (IN4)**, and **18 (ENB)**.
- The **buzzer** was attached to pin **25**.
- A **LiquidCrystal_I2C** object was created with the address `0x27` for displaying data.
- A variable `setTemp` (default 22°C) was used to store the temperature threshold.

These pin mappings ensured proper coordination between input (sensor) and output (actuators).

5.4 Blynk Virtual Pins

// V2 -> Temperature Display

// V3 -> Humidity Display

// V4 -> Set Temperature (Slider)

// V6 -> Cooling ON/OFF LED

// V7 -> Alarm LED (High Temp Alert)

Each virtual pin in the Blynk app corresponded to a specific function:

- **V2** and **V3** displayed live temperature and humidity data.
- **V4** controlled the threshold temperature via a **slider widget**.
- **V6** indicated the **cooling system state** (ON/OFF).
- **V7** controlled a **virtual LED** for high-temperature alerts.

5.5 Dynamic Threshold Control

```
BLYNK_WRITE(V4) {
  setTemp = param.asFloat();
  Serial.print("📱 Set Temp updated from Blynk: ");
  Serial.print(setTemp);
  Serial.println(" °C");
}
```

This **callback function** was automatically executed when the admin user adjusted the **temperature slider** in the Blynk dashboard.
The updated value replaced the previous threshold (`setTemp`), allowing real-time remote control of the system's temperature limit.

5.6 System Initialization

```
void setup() {
  Serial.begin(115200);
  dht.begin();
  lcd.init();
  lcd.backlight();
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
```

```
  pinMode(ENB, OUTPUT);

  pinMode(BUZZER_PIN, OUTPUT);

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

}
```

During system startup, all modules were **initialized**:

- The **Serial Monitor** began logging data for debugging.
- The **DHT sensor**, **LCD**, and **Wi-Fi** connection were initialized.
- The motor control pins, buzzer, and display were set to their default inactive states.
- A startup message (“Cold Storage Ready”) was displayed on the LCD after successful connection to Wi-Fi and Blynk.

5.7 Main Loop Function

```
void loop() {

  Blynk.run();

  float temperature = dht.readTemperature();

  float humidity = dht.readHumidity();

  ...

}
```

The **loop()** function continuously executed during system operation.
It performed the following major tasks in each cycle:

1. **Executed Blynk background operations** to maintain a stable connection.
2. **Read sensor data** from the DHT11 module.
3. **Displayed temperature and humidity** readings on both the LCD and Blynk dashboard.
4. **Compared** the measured temperature against the **threshold (setTemp)** to decide system state.
5. **Activated or deactivated** the cooling module, fan, LED, and buzzer accordingly.

5.8 Error Handling

```
if (isnan(temperature) || isnan(humidity)) {

  lcd.print("Sensor Error!");

  delay(2000);

  return;

}
```

If the **DHT11 sensor** failed to provide valid readings (returned NaN), an error message (“Sensor Error!”) was displayed on the LCD, and the loop was temporarily halted to prevent false actions.

5.9 Cooling and Alert Control Logic

```
if (temperature >= setTemp) {
```

```
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, 255);
  Blynk.virtualWrite(V6, 1);
  Blynk.virtualWrite(V7, 255);
  digitalWrite(BUZZER_PIN, LOW);
  lcd.print(" ALERT: HIGH TEMP!");
 } else {
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, 0);
  Blynk.virtualWrite(V6, 0);
  Blynk.virtualWrite(V7, 0);
  digitalWrite(BUZZER_PIN, HIGH);
  lcd.print("Temp Normal");
 }
```

This logic block governed the **core functioning** of the cooling system:

- When the **temperature ≥ threshold**:

    - The **Peltier module** and **fan** were activated via the motor driver.
    - The **buzzer** and **alert LED** (both hardware and virtual) were turned ON.
    - The LCD displayed *“ALERT: HIGH TEMP!”*
- When the **temperature < threshold**:
    - The **Peltier**, **fan**, and **buzzer** were turned OFF.
    - The system returned to *“Normal”* mode, and Blynk was updated accordingly.

This ensured continuous monitoring and responsive control, maintaining the desired environmental conditions.

5.10 System Feedback

Throughout operation, the ESP32 continuously sent **real-time temperature and humidity updates** to the Blynk dashboard.
The admin could adjust the threshold temperature, and the changes were immediately reflected in the system’s response.
The **LCD display** and **Serial Monitor** provided local feedback, while the **Blynk mobile interface** enabled remote monitoring and alerts.

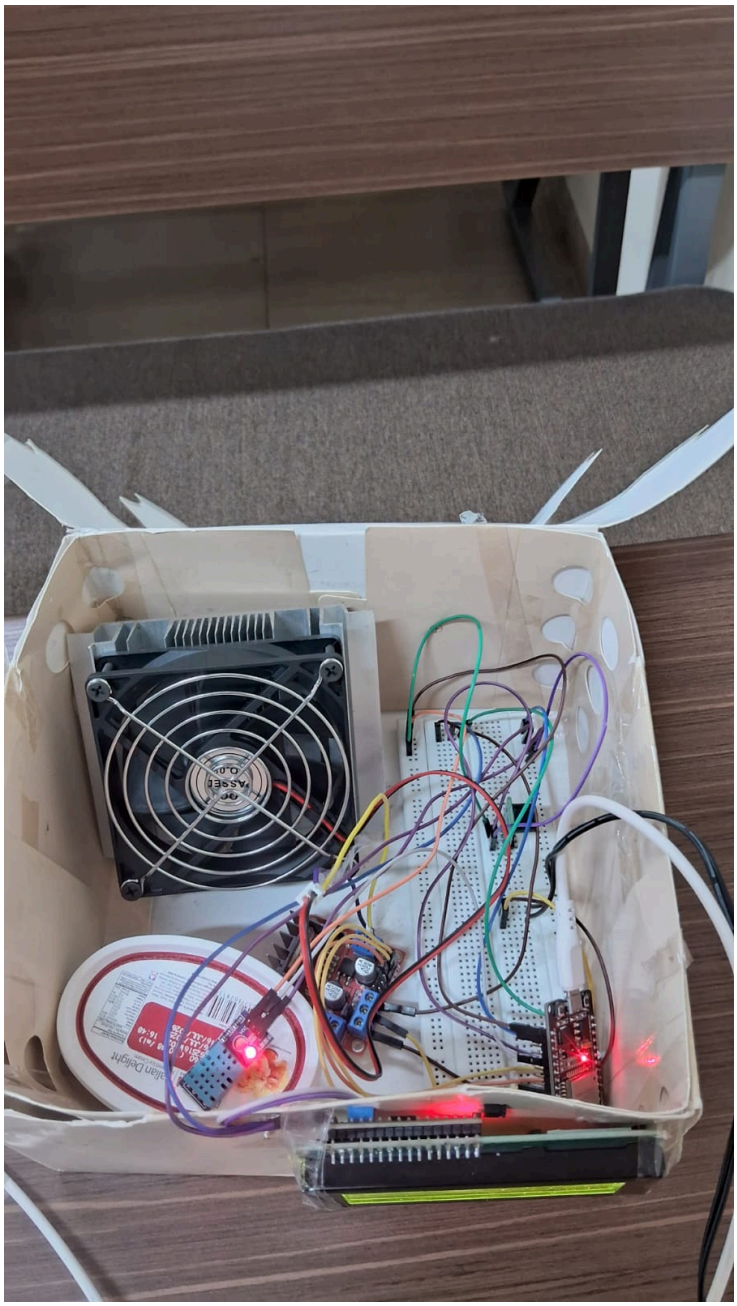
5.11 Summary

In summary, the program effectively combined sensor data acquisition, actuator control, and IoT-based connectivity.
The modular and event-driven design ensured that the system remained responsive, self-regulating, and user-friendly.
The implementation demonstrated the feasibility of integrating **IoT technology** with **environmental control mechanisms** for smart storage applications.

Results

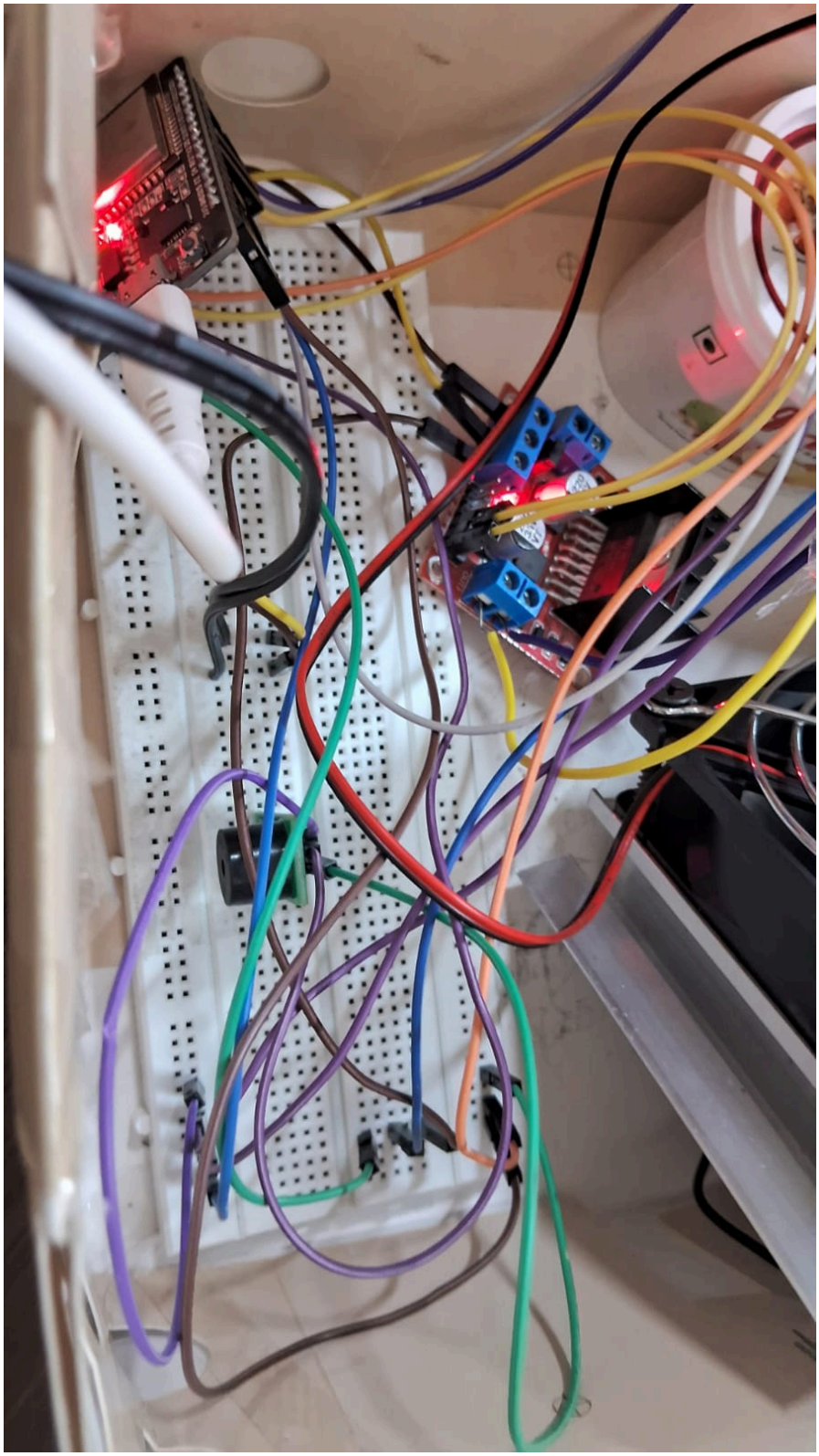
A. Photos of the Components

1. Complete Smart Cold Storage System



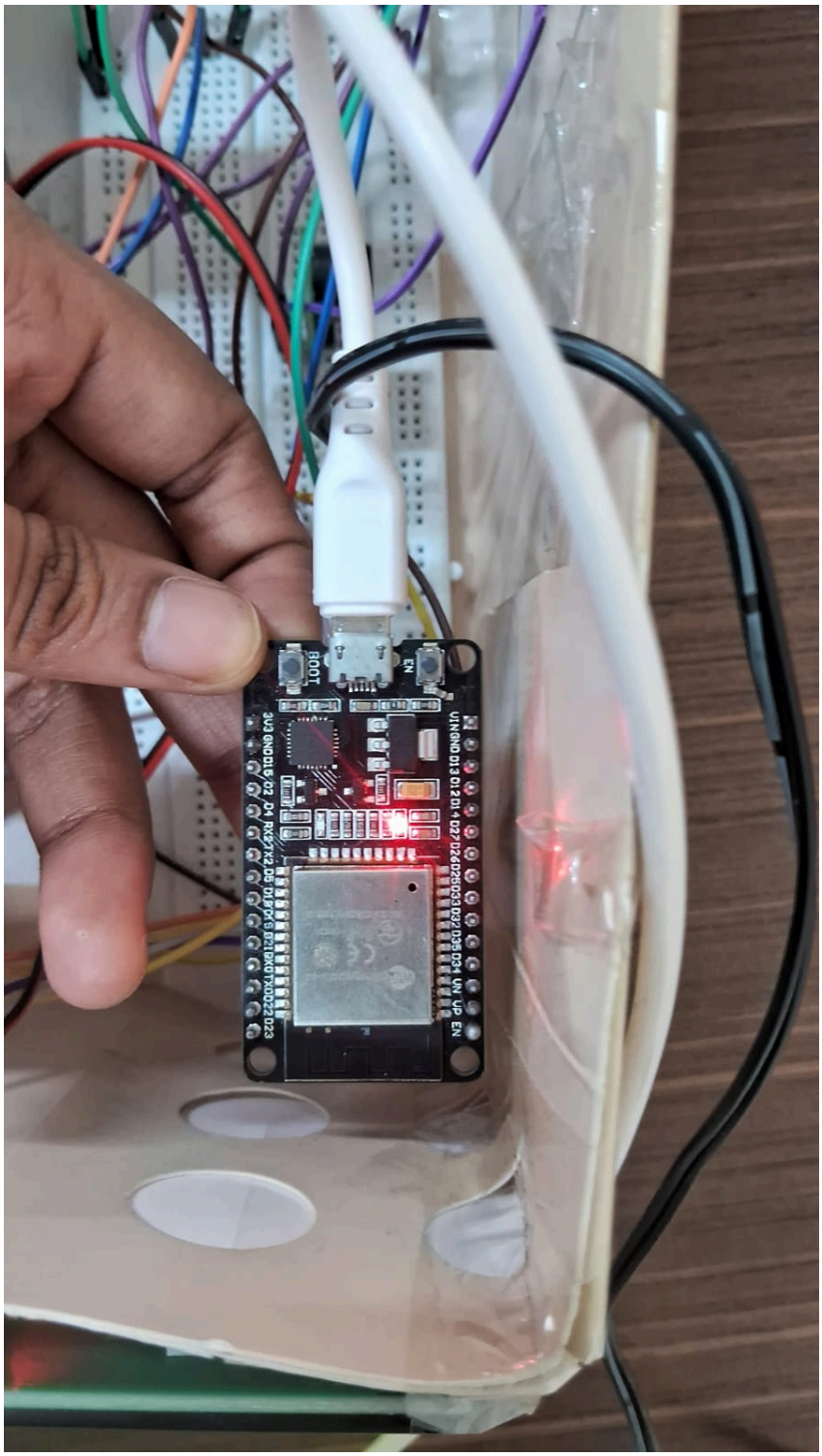
2. Peltier when the fan is turned on

3. Breadboard depicting all the connections



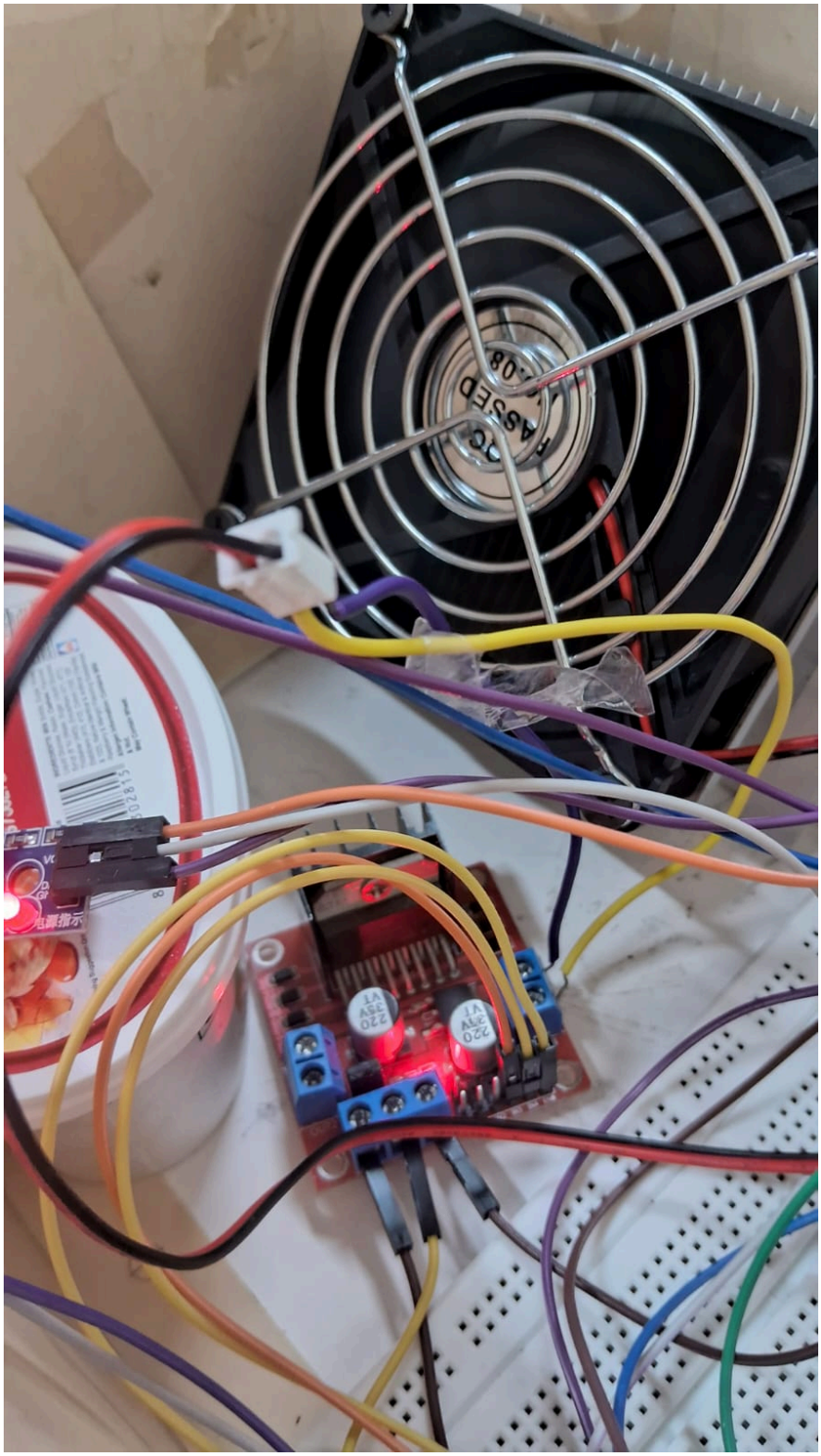
\

4. ESP32

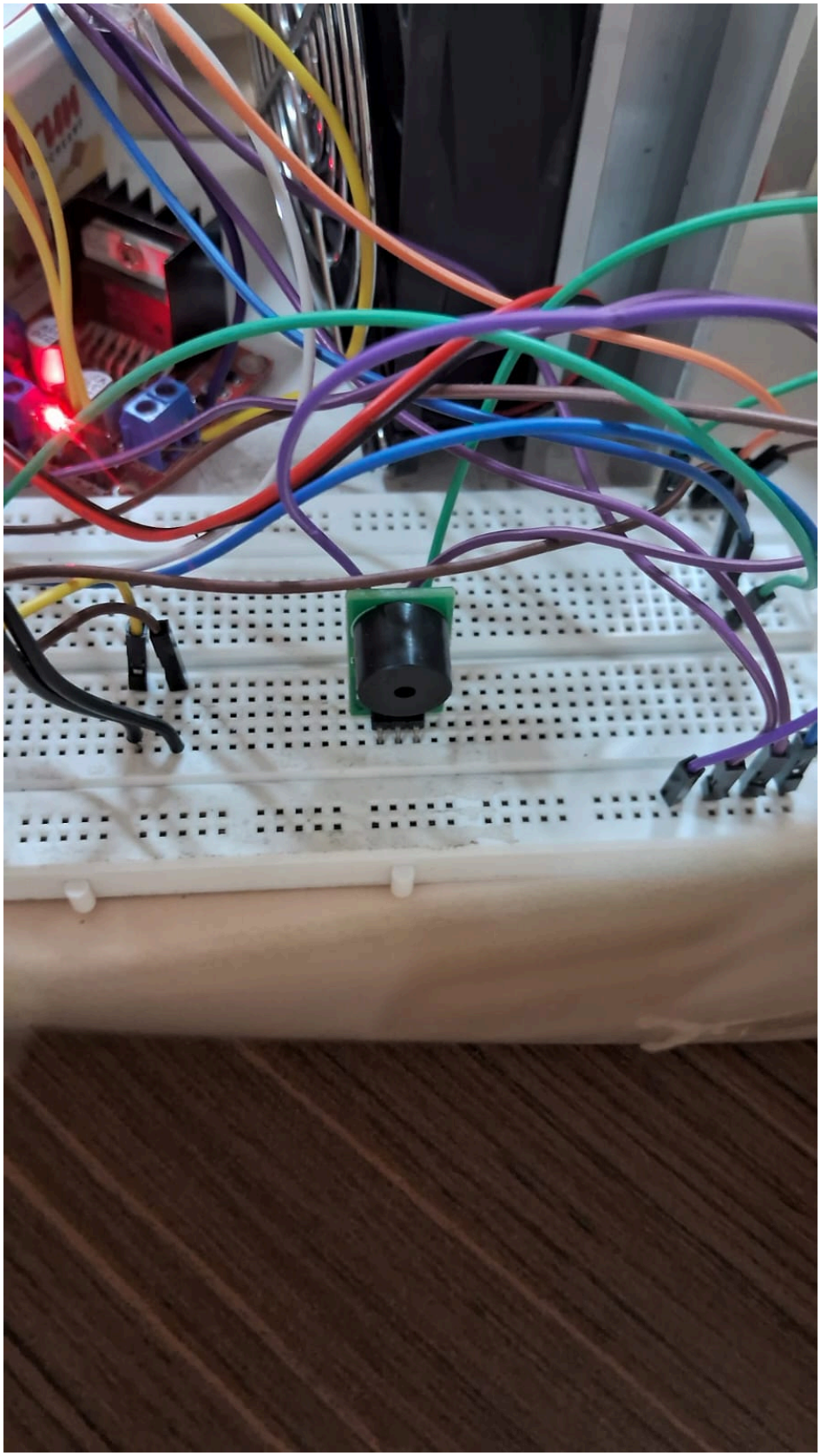


5. DHT11 Sensor

6. Motor Driver



7. Buzzer

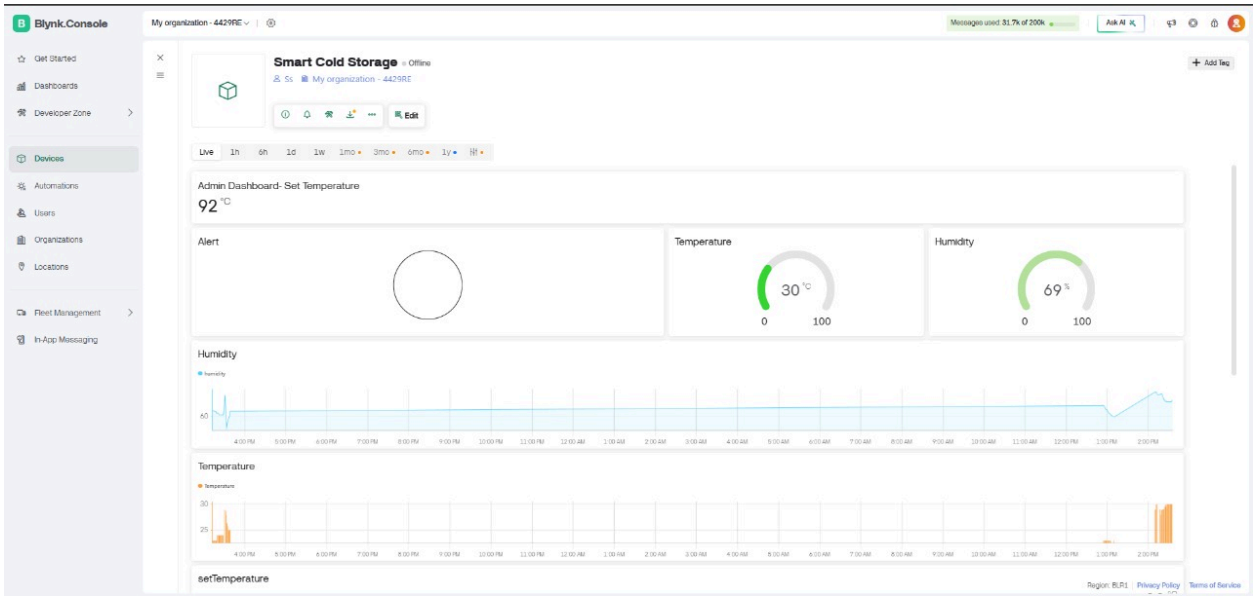


8. LCD Display with Readings

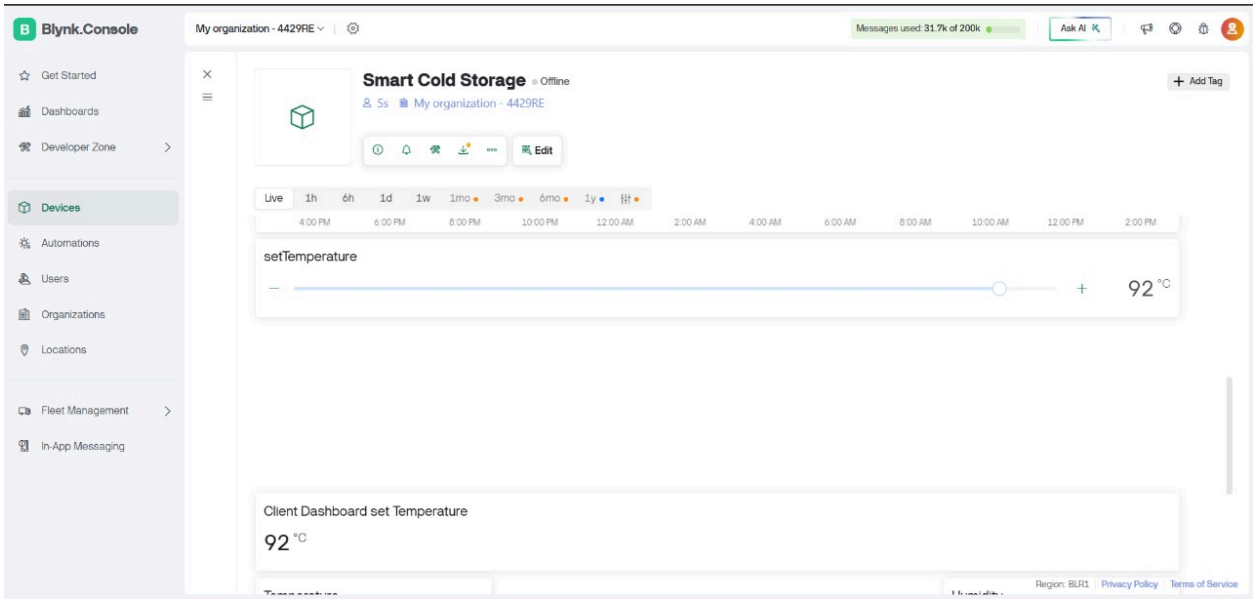
9. Icecream Box - Depicting a sample item in a Cold Storage System

B. Blynk Screenshots

10. Admin Dashboard



11. Temperature Regulator



12. Client Dashboard

Results Explained

The results clearly showed that the **Smart Cold Storage System** achieved its primary objectives — maintaining optimal environmental conditions for stored goods while providing real-time visibility through IoT connectivity.
The combination of **ESP32**, **DHT11**, and **Blynk** created an efficient, scalable, and user-friendly solution.

While the current prototype was designed for small-scale use, the architecture could easily be expanded for **commercial cold rooms**, **food storage**, or **medical applications** by using higher-power cooling units and more precise sensors.
The integration of **data logging**, **energy optimization**, and **AI-based predictive control** could further enhance the system's efficiency and usability in future iterations.

7. Conclusion

The **Smart Cold Storage System** successfully demonstrated the potential of **Internet of Things (IoT)** technology to automate, optimize, and enhance the efficiency of temperature-controlled storage environments. The prototype effectively monitored **real-time temperature and humidity levels**, responded automatically to threshold changes, and provided users with **remote visibility and control** through the **Blynk IoT platform**.

By integrating sensors, actuators, and cloud-based connectivity into a single intelligent framework, the system achieved seamless coordination between hardware and software components. The **DHT11 sensor** provided continuous environmental readings, while the **ESP32 microcontroller** processed data and made automated decisions to activate or deactivate the **Peltier cooling module** and **fan**. This automation eliminated the need for constant human supervision and ensured a stable temperature range for sensitive goods.

The inclusion of **visual (LED)** and **audible (buzzer)** alerts further enhanced system responsiveness and user safety. These alert mechanisms provided immediate warnings in case of temperature rise, reducing the risk of product degradation or equipment failure. The **LCD display** offered clear, on-site visibility of environmental conditions, while the **Blynk mobile dashboard** extended monitoring and control capabilities to remote users in real time.

An important design choice was the implementation of **role-based access control** within the IoT dashboard:

- **Administrators** were allowed to configure or adjust temperature thresholds.
- **Clients** were limited to viewing live environmental data.

This separation of roles improved both **security** and **operational efficiency**, ensuring that only authorized personnel could modify system parameters while general users could still observe system performance.

From an operational standpoint, the system demonstrated notable advantages:

- **Automation:** Reduced manual intervention by maintaining temperature automatically.
- **Energy Efficiency:** The system activated cooling only when required, conserving power.
- **Reliability:** Consistent and stable readings with quick response to environmental fluctuations.
- **Accessibility:** Remote monitoring and control through the Blynk IoT cloud.

These results collectively validated that **IoT-driven cold storage management** can provide a sustainable, intelligent, and user-friendly alternative to traditional manual monitoring systems.

However, while the project achieved its objectives as a proof of concept, certain limitations were identified. The **DHT11 sensor**, though effective for prototyping, offered moderate accuracy and a limited operating range. Likewise, the **Peltier module** provided sufficient cooling for small-scale applications but would need to be upgraded for larger storage environments. Addressing these limitations could significantly expand the system's industrial viability.

Looking forward, the project offers vast scope for enhancement and scalability. By integrating **predictive analytics**, **machine learning algorithms**, and **cloud-based data logging**, future iterations could predict temperature variations and optimize power usage more intelligently. Additionally, the adoption of more advanced sensors, automated ventilation systems, and energy-efficient cooling mechanisms could transform this model into a **commercial-grade IoT cold storage solution**.

In summary, the Smart Cold Storage System stands as a strong demonstration of how IoT can revolutionize environmental management in storage facilities. It successfully bridged automation, connectivity, and control, providing a foundation for next-generation smart refrigeration systems suitable for **industrial, agricultural, and biomedical applications**.

References

[1] R. Ali, S. Khan, and A. Mehmood, “IoT-based cold chain monitoring system for pharmaceutical storage using MQTT protocol,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 2, pp. 112–118, 2022.

[2] P. Chaudhary, R. Verma, and D. Bhattacharya, “Energy-efficient IoT-enabled refrigeration system using predictive analytics,” *Journal of Smart Systems and Technologies*, vol. 8, no. 3, pp. 45–53, 2022.

[3] B. Das and A. Roy, “Smart temperature control in refrigeration systems using cloud computing,” *Journal of Embedded Systems and IoT Applications*, vol. 5, no. 1, pp. 33–40, 2020.

[4] S. Gupta and P. Tiwari, “Design of low-cost IoT-based real-time cold storage monitoring system,” *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, vol. 9, no. 4, pp. 87–93, 2021.

[5] M. Hossain and T. Chowdhury, “Thermoelectric cooling in portable cold storage using Arduino and IoT,” *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 9, no. 9, pp. 1152–1159, 2020.

[6] A. Joseph and M. George, “Warehouse temperature and humidity monitoring using IoT-enabled sensor networks,” *International Journal of Emerging Technologies in Engineering Research*, vol. 9, no. 5, pp. 98–104, 2021.

[7] A. Kumar and R. Singh, “Design and implementation of smart refrigeration control using IoT,” *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 8, no. 10, pp. 2041–2047, 2020.

[8] X. Li, Y. Chen, and L. Zhao, “Cloud-based smart warehouse system using ESP32 and temperature sensing modules,” *Journal of Intelligent Systems*, vol. 31, no. 6, pp. 289–296, 2022.

[9] D. Mishra and R. Patel, “IoT-assisted cold chain monitoring for perishable food supply management,” *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7650–7658, 2021.

[10] J. Nair and K. Thomas, “Design of IoT-enabled smart refrigeration system using thermoelectric modules,” *Proceedings of the International Conference on IoT Applications in Engineering*, vol. 3, no. 1, pp. 27–34, 2021.

[11] V. Patil, S. Deshmukh, and P. Kale, “Automated temperature and humidity regulation using thermoelectric cooling,” *International Journal of Electronics and Communication Systems*, vol. 15, no. 4, pp. 65–72, 2021.

[12] M. Rahman, T. Hossain, and S. Alam, “IoT-enabled intelligent food preservation system using adaptive cooling mechanisms,” *Asian Journal of Engineering and Applied Sciences*, vol. 10, no. 2, pp. 77–85, 2021.

[13] N. Rao, L. Devi, and V. Prasad, “Real-time greenhouse environment control using Blynk IoT platform,” *International Journal of Engineering Research & Technology*, vol. 9, no. 8, pp. 150–156, 2020.

[14] P. Sharma, K. Gupta, and A. Jain, “IoT-based temperature and humidity monitoring system using NodeMCU and Blynk application,” *International Research Journal of Engineering and Technology*, vol. 6, no. 7, pp. 2210–2215, 2019.

[15] R. Sundar and S. Velmurugan, “Implementation of smart cold storage using IoT and data analytics for agricultural products,” *International Journal of Smart Technology and Research*, vol. 11, no. 2, pp. 53–61, 2023.