Temperature Control and Energy Monitoring System

Muhmmad Mujtaba Siddique
National University of Science and Technology
Islamabad, Pakistan
CMS ID 411362

Ibrahim Danishmand Khan
National University of Science and Technology
Islamabad, Pakistan
CMS ID 405680

Danyal Hasan Khan
National University of Science and Technology
Islamabad, Pakistan
CMS ID 412297

Muhammad Ahmad
National University of Science and Technology
Islamabad, Pakistan
CMS ID 409888

***Abstract*—This paper focuses on documenting and presenting the results of an Electrical Engineering Project aimed at creating a thermally controlled environment in a small, insulated box. The temperature of the box is set remotely using a website and the internal temperature is raised or lowered until the desired temperature is obtained.**

I. Introduction

A fully functioning temperature-controlled environment is one where the environment can be heated and cooled using appropriate equipment until the temperature of the environment reaches the desired temperature. This process involves taking regular readings of the current temperature and making use of heating and cooling equipment that can be operated remotely and switches off automatically once the desired temperature is reached.

II. Literature Review

A. Idea

The design philosophy behind our approach to the OEL was to be able to set the desired temperature and operate the equipment using a website from any device with an internet connection to maximise versatility of usage which leads us to use an ESP32 Microcontroller as the brains of our equipment. A Piezo-electric cooler with reversible polarity is used to heat the environment and cooling fans are used to cool it. The ESP32 is connected to four relays which independently control the polarity of the piezo-electric coolers as well as powering the cooling fans.

B. Applications

The design should be usable in:

1) Pharmaceutical Industry: Pharmaceutical manufacturing requires precise temperature controlling mechanisms for their production.

2) Air Conditioners: Air Conditioning units are used to regulate and maintain set temperatures in houses, buildings, schools, offices etc.

3) Food Industry, for keeping food items at a set temperature to prevent food from going bad.

4) Automotive Industry: Temperature control systems are regularly deployed in consumer vehicles

C. Parts Used

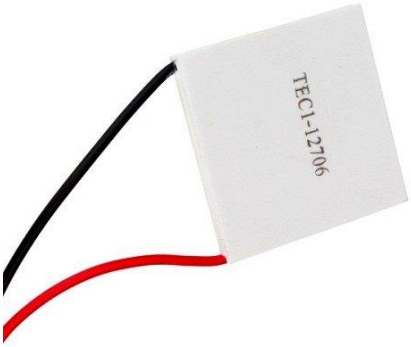
1) ESP32 Microcontroller

An Arduino based Wi-Fi microcontroller that we will use as the brains of our operation. It handles the logic for turning on and off the relays as well as reading and analysing readings from the sensors.

2) SSD1306 0.96in OLED display

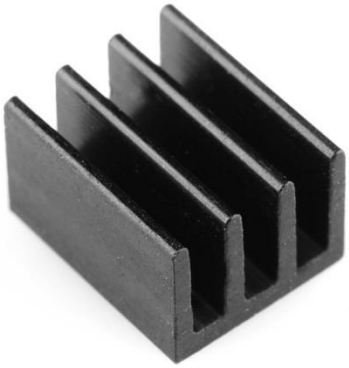
The SSD1306 is a small display that can be used with the I2C communication protocol. It has a resolution of 128 by 64 pixels.

3) TEC1-12706 Piezo-electric cooler



The TEC1-12706 is a Peltier module that can be used for both cooling and heating. The module uses the Peltier principle to heat and cool its sides simultaneously. The hot side of the module can be changed by switching its polarity.

4) Heatsinks

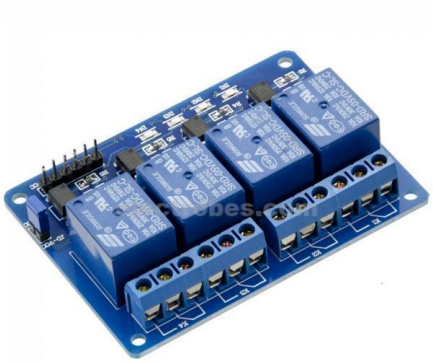


We used heatsinks to better radiate the heat coming off of the Peltier modules and to have a larger surface area for cooling.

5) Cooling Fans

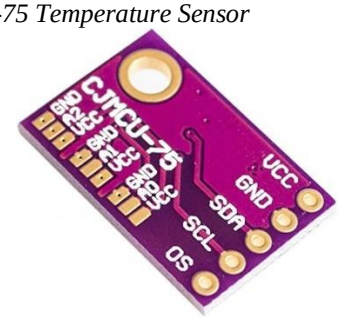
We used two regular 30mm fans to circulate air inside the room and help in heating via convection.

6) 4-Relay Module



The 4 Relay Module has four independently controllable relays that are used to control the polarity of the Peltier modules as well as controlling the fans.

7) CJMCU-75 Temperature Sensor



The CJMCU-75 Temperature Sensor operates using I2C communication and can give a reading in degrees Celsius.

8) ACS712 Current Sensor

The ACS712 is a current sensor module which returns an analogue reading that can be interpreted by the ESP32.

9) 12V Li-ion Battery

We chose a 12V battery with a high current output to power the entire circuit so that the Peltier modules can be used to their full effect.

10) Voltage Regulator

A dynamic voltage regulator was used to step down the voltage from 12V to 5V for the ESP32 and sensors.

III. Designing the Circuit

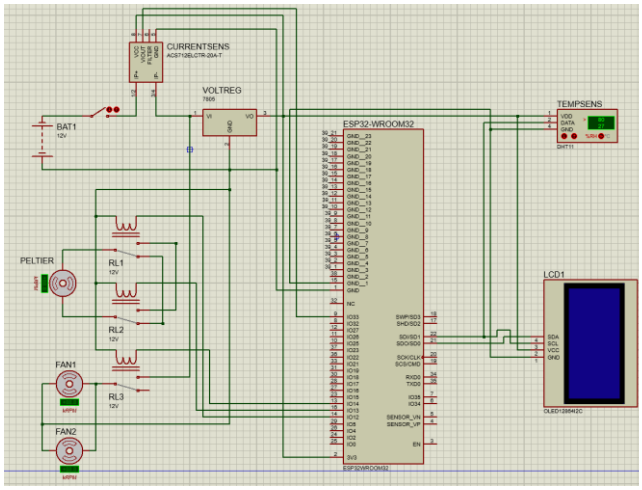


Figure 1: Final Circuit Design

To power the ESP32, we need a voltage between 5V and 8V going into its built-in regulator to ensure safe operation with minimum power loss. The sensors on the other hand require exactly 5V across them for ideal operation. Therefore, it was necessary to provide a 5V power line to both the ESP32 and the sensors using a Voltage Regulator. The fans and cooler we used require a 12V input therefore they have to run on a separate circuit connected directly to the battery and be controlled by the relays.

A. Peltier Cooler

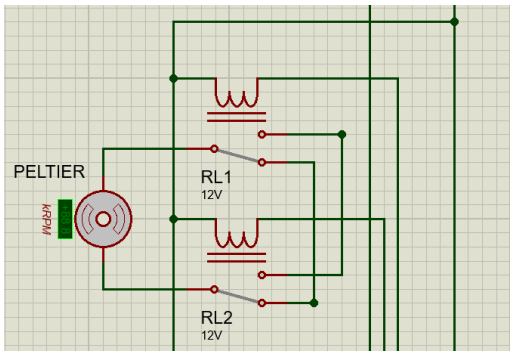


Figure 2: Peltier Module (Shown as a Motor due to lack of schematic availability)

The Peltier module is connected to the system via two relays, with both sharing ground and power pins. However, either relay connects to a different terminal of the Peltier which allowed us to switch the polarity of the module to initiate either heating or cooling.

B. Fan

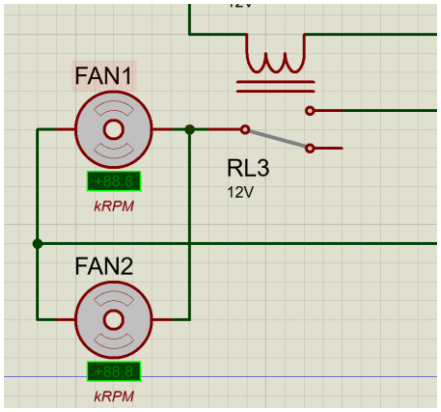


Figure 3: Fans Connection

The fans are controlled by a relay that draws power from the 12V battery and is operated by a digital pin on the ESP32.

C. *Temperature Sensor and Display:*

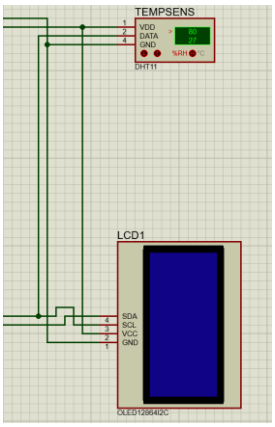


Figure 4: Temperature Sensor and Display Connection

The CJMCU-75 and OLED both require 5V from the voltage regulator and can communicate with the ESP32 with I2C communication. Therefore, they share the same data lines but operate at different addresses, which allows the microcontroller to communicate with multiple devices simultaneously.

D. *Battery and Voltage Regulation:*

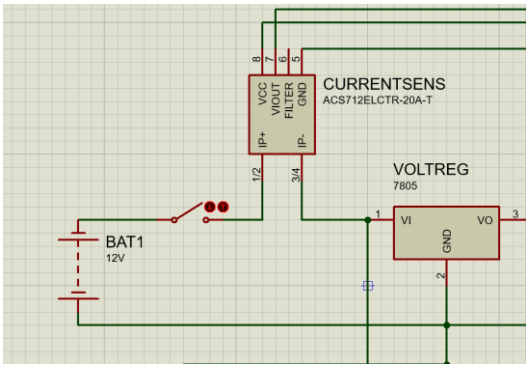


Figure 5: Battery, Current Sensor, and Voltage Regulator connection

Since we need to calculate the total power of the system, we need to measure the current drawn from the battery using a sensor. The battery's positive terminal goes through the current sensor and into the voltage regulator which then sends power to the 5V power rail for the ESP32 and sensors.

E. *Web Control:*

The ESP32 hosts a local webserver that can be accessed by any device on the same network. Alternatively, the ESP32 can host its own Wi-Fi connection that the device can connect to. On connecting to the webserver, the ESP32 pushes an HTML file that is displayed by the browser. Then the ESP32 sends asynchronous update events to the client, which are deciphered by the Javascript on the website and used to change its contents. This allows us to simply update the values on the website without having to repeatedly refresh the website. Alternatively, an AJAX server can be used to quickly update the contents of the client's website but may be more processor intensive, therefore we opted for an easier and less taxing option.

IV. Methodology

A. *Construction:*

We created a wooden box from composite 1-inch wood. The box is 10in x 10in x 10in and has a cutout on top to create room for the Peltier modules and heatsinks. The module and heatsinks are then bolted to the top of the box. A breadboard with the ESP32 is then added to the box along with the relay and voltage regulator. Then the battery is added, connected to a switch, and resting on top of the box.

B. *Code*

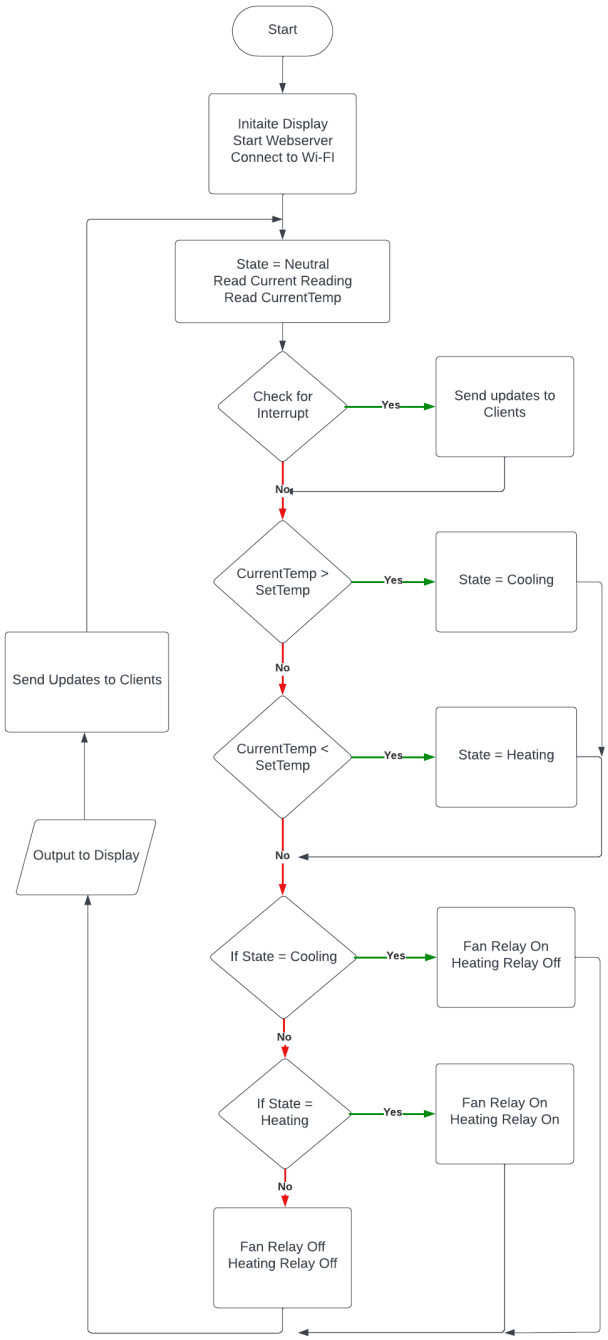


Figure 6: Flowchart showing the logic for the ESP32.

V. Results and Discussions

A. Heating

We recorded a high temperature of 32 degrees Celsius with a maximum current draw of 1830 mA and a max power of 21.9 W.

B. Cooling

We recorded a low temperature of 23 degrees Celsius with a maximum current draw of 612 mA and a max power of 7.34 W.

State	Temperature (ºC)	Max Current Draw (mA)	Max Power Draw (W)
Cooling	23*	612	7.34
Heating	32	1830	21.9

**The Peltier failed to cool more than a few degrees when going sub-ambient.*

A problem we noticed working with Peltier modules is that both its side start to heat up when under immense heat, therefore negating any cooling effects it would have originally had. This results in sub-optimal cooling and is therefore extremely slow. To improve on this, we need to dissipate heat on the hot side of the Peltier faster, so that it doesn't significantly affect the cooling.

VI. Conclusion

Manually maintaining a set temperature is inefficient since its always prone to human error and can be unreliable at times. However, a system that automatically can stabilize itself at the given temperature is very reliable and is extremely versatile.

With the ESP32 as the brains of our system, we created a system that can be controlled from anywhere in your house or can even be expanded to host on a publicly open server that can be accessed from anywhere on the internet. However, our project isn't perfect and can be further improved upon which leaves the door open for further improvements such as using a better Peltier module for more consistent cooling.