Thermal Management System Utilizing DC-DC Converters and Peltier Module with Sensor-Based Temperature Measurements

Centre for Electronics Design and Technology,

Netaji Subhas University of Technology, New Delhi

Made by
Prajwal Singh, Ankit Raj, Razzaq Ali

Contents

1. Synopsis

This project focuses on developing an efficient and stable cooling system utilizing DC-DC converters to enhance the performance of a thermoelectric cooler. The system incorporates a boost converter to power a fan within a heat sink and a buck converter to regulate power to a Peltier module, optimizing its cooling efficiency. A microcontroller-based feedback mechanism is integrated to ensure precise system output control. This mechanism continuously monitors temperature through sensors measuring ambient conditions, the Peltier module's cooling plate, and the heat sink, and dynamically adjusts the system to maintain a consistent and effective temperature differential. The goal is to achieve improved cooling performance by enhancing both the stability and efficiency of the thermoelectric cooling system.

2. Introduction

Effective thermal management is critical in modern electronics to ensure reliability, efficiency, and longevity of devices. As electronic components become more powerful and compact, managing the heat they generate has become increasingly challenging. This project was motivated by the need to develop a cost-effective and efficient thermal management system that leverages DC-DC converters and thermoelectric cooling to maintain optimal operating temperatures in electronic systems. By incorporating real-time temperature monitoring and control, this project aims to enhance the performance and lifespan of electronic devices, contributing to more sustainable and reliable electronic systems.

Thermoelectric cooling, utilizing the Peltier effect, provides a non-mechanical, refrigerant-free solution for temperature management, making it ideal for electronics cooling, portable refrigeration, and precise temperature control in scientific instruments. However, its efficiency depends heavily on effective power management and precise temperature regulation. In this project, DC-DC converters are employed to enhance performance, with a boost converter powering a fan within a heat sink for heat dissipation and a buck converter regulating the power to a Peltier module. A microcontroller-based feedback mechanism monitors temperatures in real time, dynamically adjusting the system to maintain a stable temperature differential, thereby optimizing cooling performance and system stability.

We have 3-D printed a **Stevenson screen**, used in this project to house the LM35 temperature sensor which records the real-time data of the ambient temperature. The Stevenson screen works by carefully managing airflow around the temperature sensors housed within it, ensuring that the sensors measure the true ambient air temperature without interference from direct sunlight or radiated heat.

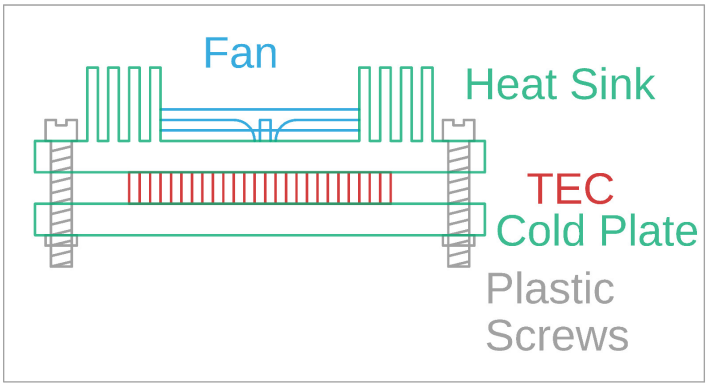


Figure 2.1: Peltier Module(TEC) Setup

This block diagram shows how the Peltier Module is set up. A heat sink with a fan is placed on the hot side of the Peltier module to efficiently dissipate the heat. An aluminum plate is placed with the cool side of the peltier module on which a thermocouple is placed to measure the temperature of cool side.

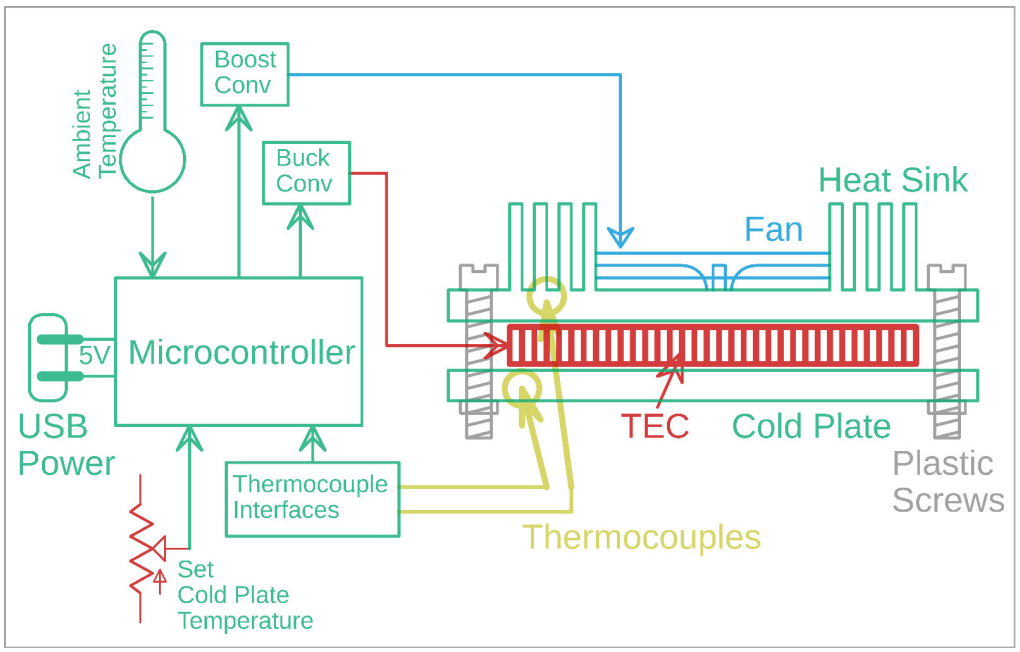


Figure 2.2: Block Diagram

This block diagram illustrates the overall structure and flow of the thermal management system. It consists of key components, including the DC-DC converters (boost and buck), the Peltier module, a microcontroller, and various sensors. The boost converter powers a fan within the heat sink, enhancing heat dissipation, while the buck converter provides regulated power to the Peltier module for efficient cooling. The microcontroller serves as the control unit, receiving real-time data and adjusting the system parameters accordingly to maintain optimal thermal conditions.

3. Circuit Schematic

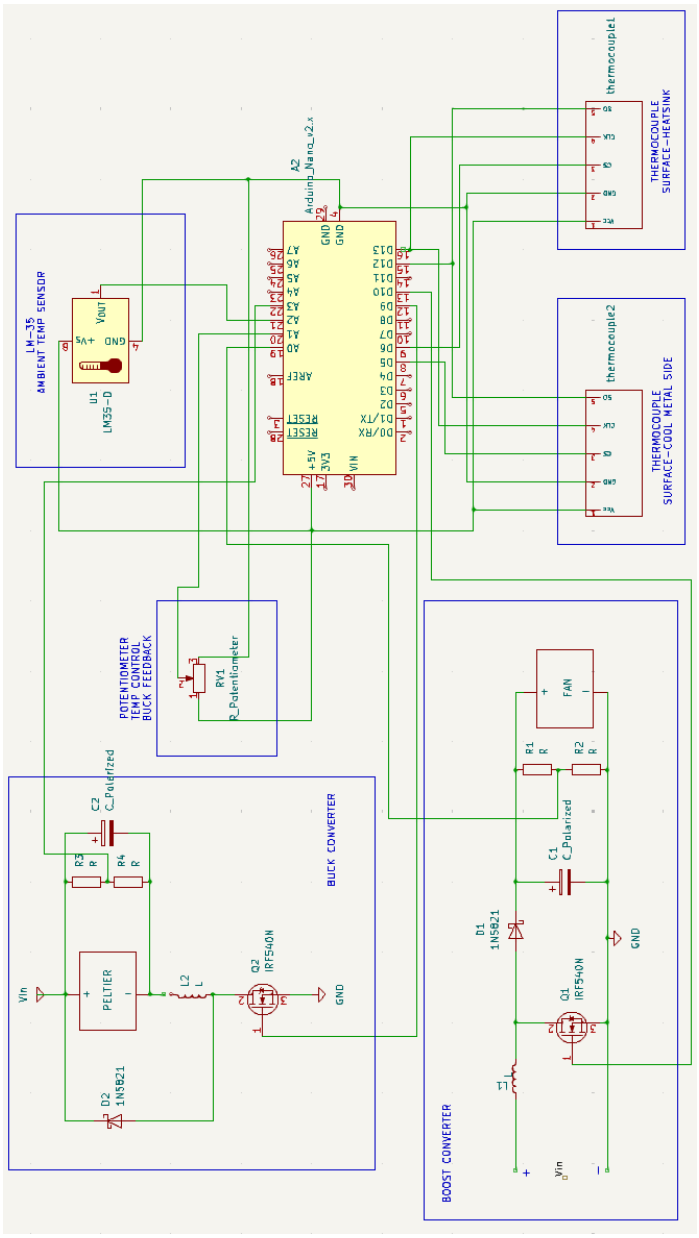


Figure 3.1: Schematic Diagram

4. Project Progression

Overview

Initial Analysis of Peltier Module

- **Current and Voltage Analysis:** The project began with a thorough analysis of the current and voltage consumption of the Peltier module to understand its power requirements.
- **Initial Power Attempt:** We attempted to power the module using only current sources. However, we encountered issues with not achieving the required current levels, leading to excessive heat generation by the transistor.

Incorporating Cooling Mechanisms

- **First Heat Sink and Fan Integration:** To mitigate the heat on the hot side, a heat sink was added, and a fan was placed on top of it. This setup successfully reduced the temperature, but it did not bring the temperature down to the desired level.
- **Upgraded Heat Sink:** To further enhance cooling, a new heat sink with an integrated fan was introduced. This provided a more efficient cooling solution, though it still required further optimization.

Development of Power Converters

- **Need for Dual Output Values:** The project required two different output values from the same input source. To achieve this, a buck and boost converter was designed and built from scratch.
- **Integration and Performance Enhancement:** After integrating the buck and boost converters with the rest of the system, we observed improved performance. However, we faced challenges with fluctuating voltage regulation during operation.

Microcontroller-Based Feedback Mechanism

- **Stabilizing Output Voltage:** To address the issue of varying voltage, a microcontroller-based feedback mechanism was developed. This system actively monitored the output voltage of the DC-DC converters and maintained it consistently throughout operation.

Real-Time Temperature Monitoring:

- Three temperature sensors were integrated into the system for real-time monitoring. These sensors tracked the temperature of the ambient environment, the heat sink, and the aluminum plate on the cool side of the Peltier module. This ensured that the temperature differential was maintained effectively, optimizing the performance of the cooling system.

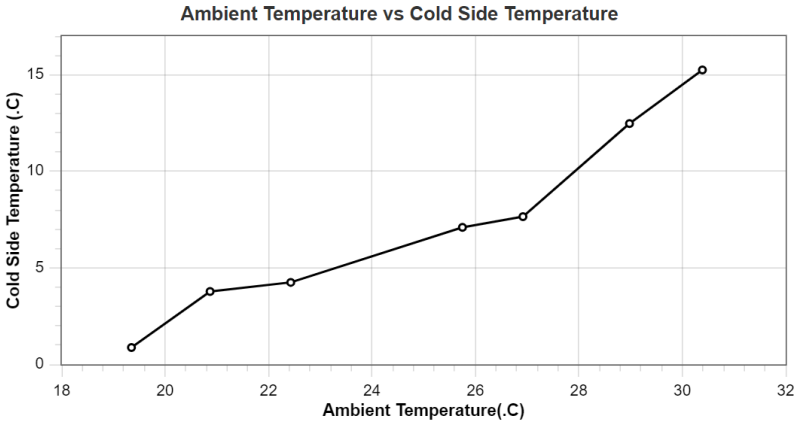
5. Working

5.1 Pseudo Code

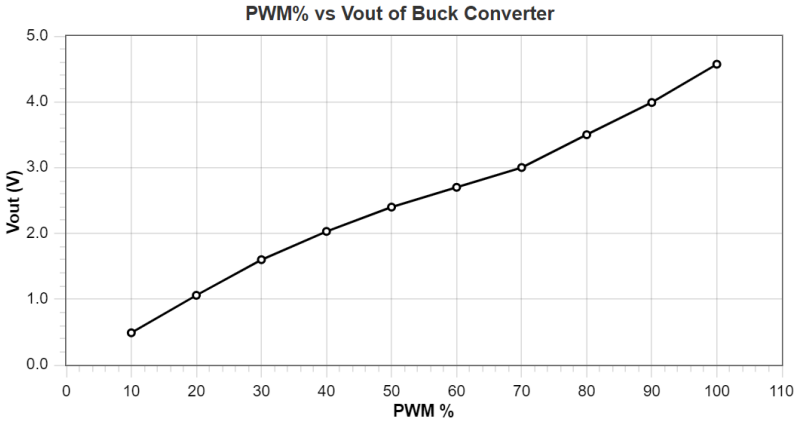
The following pseudo code outlines the primary functions and logic implemented in the system:

```
1. Setup:
   - Initialize serial communication for data logging.
   - Configure PWM pins for the buck and boost converters.
   - Set up the timers to generate a 37.5 kHz PWM signal.

2. Loop:
   - Boost Converter Control:
     - Read reference voltage.
     - Map reference voltage to PWM duty cycle.
     - Adjust duty cycle based on the difference from the desired value.
     - Update the PWM signal for the boost converter.

   - Buck Converter Control:
     - Read temperature control input.
     - Map input to PWM duty cycle.
     - Update the PWM signal according to the potentiometer.

   - Data Logging (every 10 seconds):
     - Read temperatures from sensors.
     - Log and calculate duty cycles and output voltages.
     - Send data to the serial monitor.

3. Functions:
   - read_Temperature():
     - Average readings from the LM35 sensor to calculate ambient temperature.

   - buckConverterControl():
     - Adjust PWM duty cycle for the buck converter based on input.

   - boostConverterControl():
     - Adjust PWM duty cycle for the boost converter based on reference voltage.
```

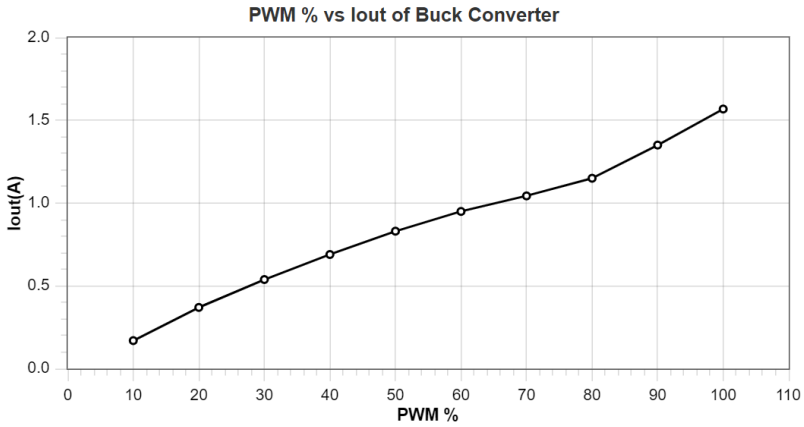
6. Observations



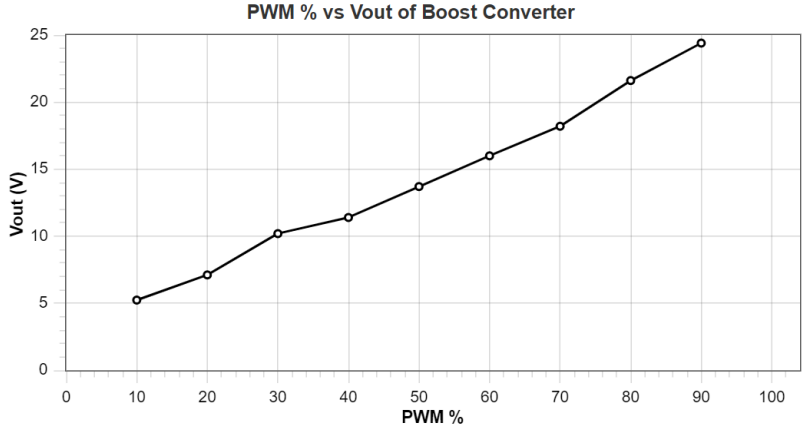
The graph titled "Ambient Temperature vs Cold Side Temperature" illustrates the relationship between the ambient temperature and the temperature on the cold side of the Peltier module. As the ambient temperature increases, there is a corresponding rise in the cold side temperature, depicted by a steadily increasing trend in the graph. The average differential maintained between the ambient temperature and the cold side temperature, as depicted in the graph, is approximately **17.3**°C.



This graph shows the relationship between the PWM percentage and the output voltage (Vout) of the buck converter. Similar to the current, as the PWM percentage increases, the output voltage rises steadily. This graph highlights the capability of the buck converter to regulate the output voltage in response to changes in the PWM duty cycle, which is crucial for maintaining stable voltage levels in various applications.



This graph illustrates the relationship between the Pulse Width Modulation (PWM) percentage and the output current (Iout) of the buck converter. As the PWM percentage increases, the output current also increases, demonstrating a near-linear relationship. This indicates that by adjusting the PWM duty cycle, the buck converter effectively controls the current supplied to the load.



This graph shows the relationship between the Pulse Width Modulation (PWM) percentage and the boost converter's output voltage (Vout). As the PWM percentage increases, the output voltage of the boost converter rises linearly, indicating that the PWM duty cycle directly influences the output voltage. The graph demonstrates that higher PWM values lead to a higher output voltage, essential for controlling the boost converter's performance in temperature regulation applications.

7. Bill of Materials (BOM)

The following table shows the list of the components used in this project.

Item No.	Component	Quantity
1	Arduino Nano	1
2	Peltier Module (TEC)	1
3	Heat Sink with Cooling Fan	1
4	Metallic Cold Plate	1
5	MAX6675	2
6	Type-K Thermocouple	2
7	IRF540N-MOSFET	2
8	Inductors	2
9	Capacitors	2
10	Resistors	4
11	Schottky Diode	2
12	Potentiometer	1
13	LM-35 Temp Sensor	1
14	Connectors	2
15	Zero Board	1
16	Connecting Wires	As required

8. Result/Conclusion

- We can achieve the required temperature differential between the ambient temperature and the cool side of the Peltier module.
- With the help of the Arduino Nano-based feedback mechanism, we are able to get constant output voltage for the DC-DC converters.
- The temperature monitoring system using the three sensors is seen to be beneficial for analysis and safe working of the project.

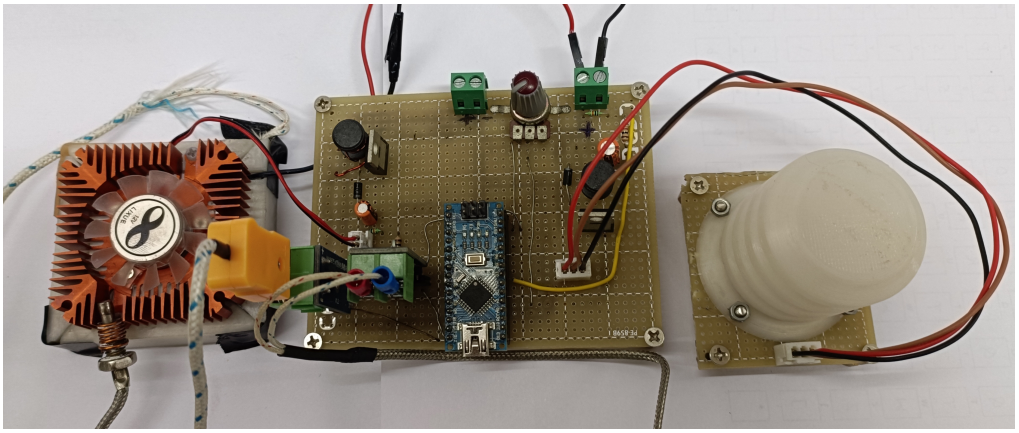


Figure 8.1: Implementation

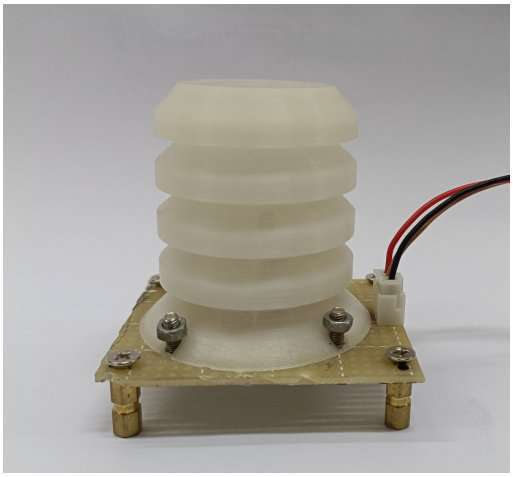


Figure 8.2: Stevenson Screen

References

[1] Paul Horowitz and Winfield Hill , *The Art of Electronics*, 1980.

[2] Dhananjay V. Gadre, "Tiny Microcontroller Hosts Dual DC/DC-Boost Converters", *Design Ideas*, 15 May 2008.

9. Code

```
#include "max6675.h"

// Pin assignments for thermocouples
int thermocoupleDataPin1 = 12;
int thermocoupleCSPin1 = 5;
int thermocoupleClockPin1 = 13;
int thermocoupleDataPin2 = 12;
int thermocoupleCSPin2 = 6;
int thermocoupleClockPin2 = 13;
// Pin for LM35 temperature sensor
int lm35Pin = A3;
// Thermocouple objects for temperature measurements
MAX6675 thermocouple1(thermocoupleClockPin1, thermocoupleCSPin1,
    thermocoupleDataPin1);
MAX6675 thermocouple2(thermocoupleClockPin2, thermocoupleCSPin2,
    thermocoupleDataPin2);
// Analog input pins for feedback and control
int buckFeedbackPin = A4;
int buckTempControlPin = A1;
int boostReferenceVoltagePin = A0;

int boostPWMDutyCycle = 0;
float boostReferenceVoltage = 85.00;

unsigned long previousMillis = 0;
const long logInterval = 10000;  // Interval for logging data (10 seconds)

void setup() {
  Serial.begin(9600);
  // Configure PWM pins
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);

  // Set up Timer1 for Fast PWM at 37.5 kHz
  noInterrupts(); // Temporarily disable interrupts

  TCCR1A = 0; // Clear Timer1 control registers
  TCCR1B = 0;
  TCNT1 = 0;  // Reset Timer1 counter

  // Set Fast PWM mode with ICR1 as top
  TCCR1A |= (1 << WGM11);
  TCCR1B |= (1 << WGM13) | (1 << WGM12) | (1 << CS10);

  // Set ICR1 value to achieve 37.5 kHz frequency
  ICR1 = 425;

  // Enable PWM output on pins 9 (OC1A) and 10 (OC1B)
  TCCR1A |= (1 << COM1A1) | (1 << COM1B1);

  interrupts(); // Re-enable interrupts
}
void loop() {
  boostConverterControl();
  buckConverterControl();
  // Read and map feedback from buck converter
  int rawBuckFeedback = analogRead(buckFeedbackPin);
  int mappedBuckFeedback = map(rawBuckFeedback, 100, 1000, 42, 0);
  // Log sensor readings and system state at regular intervals
  if (millis() - previousMillis >= logInterval) {
```

```
    previousMillis = millis();
    // Log data to serial
    Serial.print("Ambient Temperature: ");
    Serial.print(read_Temperature(lm35Pin));
    Serial.print("C_aluminium = ");
    Serial.print(thermocouple1.readCelsius());
    Serial.print(" || C_heat_sink = ");
    Serial.println(thermocouple2.readCelsius());

    Serial.print(" || Boost Duty Cycle : ");
    Serial.print((OCR1B * 100) / 425);
    Serial.print("%");

    Serial.print(" || Boost Voltage : ");
    Serial.print(((4.0 / 425.0) * OCR1B) * 11);

    Serial.print(" || Buck Duty Cycle : ");
    Serial.print(100.00 - ((rawBuckFeedback / 1023.00) * 100));
    Serial.print("%");

    Serial.print(" || Buck Output Voltage : ");
    Serial.println(mappedBuckFeedback * 0.11);
  }
}
//Function to get average ambient temperature
float read_Temperature(int pin) {
  long sum = 0; // To accumulate the total value of samples
  // Take multiple samples and accumulate the total
  for (int i = 0; i < 4; i++) {
    int sensorValue = analogRead(pin);
    sum += sensorValue; // Accumulate the sensor values
    delay(10); // Short delay between samples to allow the sensor to
        stabilize
  }
  // Compute the average value
  float averageValue = sum / 4;
  float voltage = averageValue * (5.0 / 1023.0); // Convert the average
      value to voltage
  float temperature = voltage * 100.0; // Convert the voltage to
      temperature in Celsius
  return temperature;
}
// Control logic for the buck converter
void buckConverterControl() {
  float tempControlValue = analogRead(buckTempControlPin);
  float pwmValue = map(tempControlValue, 0, 1023, 0, 425);
  OCR1A = pwmValue;  // Set PWM duty cycle for buck converter
}
// Control logic for the boost converter
void boostConverterControl() {
  float referenceVoltage = analogRead(boostReferenceVoltagePin);
  float pwmValue = map(referenceVoltage, 0, 1023, 0, 425);
  if (pwmValue < 10) {
    boostPWMDutyCycle = 0;
  } else {
    if (boostReferenceVoltage > pwmValue) {
      boostPWMDutyCycle = boostPWMDutyCycle + 1;
      boostPWMDutyCycle = constrain(boostPWMDutyCycle, 1, 424);
    } else if (boostReferenceVoltage < pwmValue) {
      boostPWMDutyCycle = boostPWMDutyCycle - 1;
      boostPWMDutyCycle = constrain(boostPWMDutyCycle, 1, 424);
    }
  }
  OCR1B = boostPWMDutyCycle;  // Set PWM duty cycle for boost converter
}
```