Precision Temperature Control

An Arduino-Based System for Engineering Students

Dr Yi-Kuen Lee
HKUST

Fr Engineering Students

Mechancal Engineering, Electrical Engineering • Applied Physics

Why Master Temperature Control? Applications in Research and Industry

Criticality in Engineering Research

Many experiments—chemical reactions, material testing, biological incubators—require precise, stable temperature environments for reproducible results.

Industrial Relevance

Temperature control is the backbone of HVAC systems, process automation, data center cooling, and smart manufacturing ecosystems.

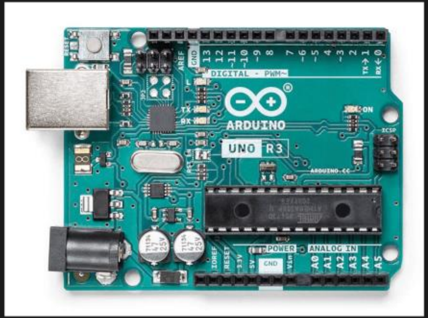
System Overview

This project demonstrates a simple ON/OFF (Bang-Bang) control system using three core components: Sensing (LM35), Processing (Arduino Uno), and Actuation (Relay).

Learning Outcome

Students gain hands-on experience in analog signal processing, digital control logic, and interfacing high-power loads with microcontrollers.

The Arduino Uno: Bridging the Physical and Digital Worlds



Microcontroller

Based on the **ATmega328P**, operating at 5V logic. It is the processing unit that executes the control algorithm.

Analog Input Pins (A0–A5)

Essential for reading the continuous voltage signal from the LM35 sensor with high precision.

Digital I/O Pins (D0–D13)

Used to send a simple ON/OFF signal to the relay module (specifically **D7** in this project).

10-Bit ADC Resolution

Provides **1024 discrete levels** (0 to 1023) for a 0V to 5V input range, crucial for temperature accuracy (~0.49°C resolution).

LM35: Linear, Calibrated, and Ready-to-Use Temperature Sensing

Precision IC Sensor

Unlike non-linear thermistors, the LM35 is a precision integrated-circuit sensor that outputs a voltage directly proportional to temperature.

Linear Scale Factor

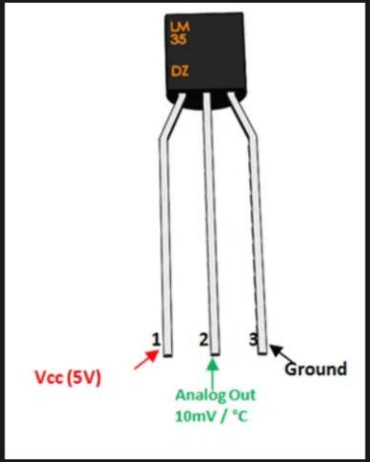
The sensor provides a linear output of **10 mV per degree Celsius (10mV/°C)**. This simplifies the conversion code significantly.

Operating Range

Capable of measuring temperatures from **–55°C to 150°C**, covering most common lab and industrial environments.

Simple Interface

Connects to the Arduino's **Analog Pin A0**. The three-pin interface (VCC, GND, OUT) requires no external calibration or signal conditioning.



The Relay Module: Safely Controlling High-Power Loads

Function

An electrically operated switch that allows the low-power Arduino (5V, ~20mA) to control high-power AC or DC loads (e.g., 240V AC, 10A).

Electrical Isolation

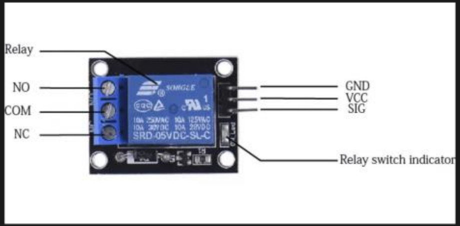
The module includes an **optocoupler**, providing complete electrical separation between the sensitive Arduino circuit and the high-voltage load circuit, ensuring safety.

Active-LOW Trigger

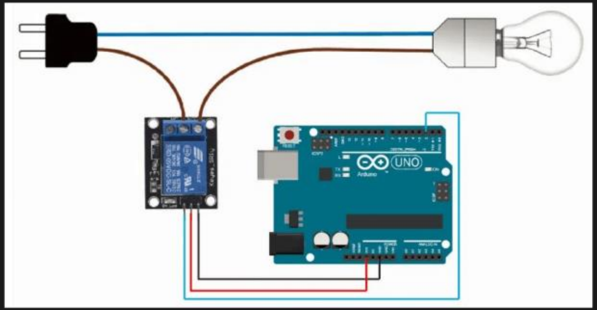
Most Arduino-compatible relays are **Active-LOW**, meaning the relay is activated (switches ON) when the Arduino pin (D7) is set to **LOW (0V)**.

Terminal Configuration

The load is connected to the **Common (COM)** and **Normally Open (NO)** terminals, ensuring the load is OFF by default (fail-safe).



Interconnection: Mapping Components to the Arduino Uno



Connection Specifications

Component	Pin	Connection	Function
LM35 Sensor	OUT	Arduino A0	Analog Temperature Signal
Relay Module	IN	Arduino D7	Digital Control Signal
LM35/Relay	VCC	Arduino 5V	Power Supply
LM35/Relay	GND	Arduino GND	Ground Reference

From Raw ADC Value to Calibrated Temperature (°C)

① ADC Reading

The Arduino reads a raw value (analogValue) between 0 and 1023 from analog pin A0.

② Voltage Conversion

The voltage output by the LM35 is calculated using:

> Voltage (V) = (analogValue / 1024) × 5.0

This converts the discrete ADC value to a continuous voltage between 0V and 5V.

③ Temperature Conversion

Since the LM35 has a scale factor of 10 mV/°C (or 0.01 V/°C), the temperature is:

> Temperature (°C) = Voltage (V) / 0.01

Dividing by the scale factor converts voltage directly to degrees Celsius.

④ System Resolution

The system's theoretical resolution is approximately 0.49°C per ADC step, calculated as:

> Resolution = (5000 mV / 1024) / 10 mV/°C ≈ 0.49°C

The Control Algorithm: Simple Threshold-Based Switching

1

Set Point (Threshold)

A desired temperature (setPoint) is defined in the code (e.g., 30°C). This is the target temperature the system aims to maintain.

2

Control Loop

The system continuously compares the measured temperature (currentTemp) with the setPoint in the main Arduino loop, executing thousands of times per second.

3

Cooling Logic (Example)

If the system is controlling a cooling fan:

> **IF**currentTemp > setPoint, **THEN** activate relay (set D7 **LOW**) → turn fan **ON**
> **ELSE** (if currentTemp ≤ setPoint), **THEN** deactivate relay (set D7 **HIGH**) → turn fan **OFF**

4

Code Structure

The logic is implemented within the Arduino loop() function, ensuring continuous monitoring and rapid response to temperature changes.

Hysteresis: Ensuring System Stability and Component Longevity

① The Problem of Chattering

Without hysteresis, if the temperature hovers at the **setPoint** (e.g., 30.0°C), the relay can **rapidly switch ON and OFF** as tiny fluctuations occur — this is called **"chattering."**

② The Solution: Hysteresis Band

Introduce a **Hysteresis Band** (e.g., ±1.0°C) to create two thresholds and prevent rapid switching:

> **Turn ON Threshold:** setPoint + Hysteresis (e.g., 31.0°C)
>
> **Turn OFF Threshold:** setPoint − Hysteresis (e.g., 29.0°C)

③ Improved Control Logic

The fan turns **ON only when temperature exceeds 31.0°C** and stays ON until it drops below 29.0°C, creating a **2°C dead band** where the relay is stable.

④ Benefits: Stability and Longevity

- **Reduced Wear:** Eliminates rapid relay switching, extending lifespan.
- **System Stability:** Prevents oscillation and ensures smooth operation.
- **Energy Efficiency:** Fewer actuations lower power use.
- **Fundamental Principle:** Hysteresis is core to many control systems.

Building the System: From Breadboard to Working Prototype

1 Hardware Assembly

Securely connect all components on a breadboard or custom PCB according to the wiring diagram.

LM35 sensor → Arduino A0

Relay module → Arduino D7

Power and ground connections verified

2 Software Setup

Install the Arduino IDE and upload the provided code to the Arduino Uno.

Download temperature_controller.ino

Connect Arduino via USB cable

Select correct board and COM port in IDE

Click Upload to program the device

3 Configuration

Modify the code variables to match your experimental requirements.

Set setPoint (target temperature)

Define Hysteresis band (±1.0°C typical)

Adjust relay trigger logic if needed

Re-upload modified code to Arduino

4 Testing & Debugging

Verify system operation and validate threshold behavior.

Open Arduino Serial Monitor (9600 baud)

Observe real-time temperature readings

Verify relay clicks at upper/lower thresholds

Connect load device to relay terminals

Future Enhancements: Moving Towards Advanced Control

① User Interface Enhancement

Integrate an **LCD or OLED display** to show real-time temperature and set point without relying on the Serial Monitor. This enables standalone operation and provides immediate visual feedback to users in the field.

② Wireless Monitoring & Control

Add a **Wi-Fi (ESP8266) or Bluetooth module** to log data and allow remote monitoring/control via a web dashboard or mobile app. Enable real-time alerts and historical data analysis for long-term trend monitoring.

③ Advanced Control Algorithm

Replace the simple ON/OFF (Bang-Bang) logic with a **PID (Proportional-Integral-Derivative) controller** for much smoother, more precise, and faster temperature regulation. PID eliminates overshoot and reduces settling time significantly.

④ Data Logging & Analysis

Implement **SD card logging** to record temperature data over long periods for experimental analysis. Enable post-processing of data for statistical analysis, trend detection, and system performance validation.

Summary: A Foundation in Applied Control Systems

Achieved Goal

Successfully designed and understood a **functional, safe, and stable temperature control system** using low-cost, open-source components. This system demonstrates the fundamental principles of embedded systems, sensor integration, and real-world control applications.

Key Takeaways

- **Analog Signal Processing:** Converting raw ADC values to meaningful physical quantities (temperature) requires understanding of sensor specifications and mathematical conversion.
- **Digital Actuation:** Microcontrollers can safely control high-power loads through relay modules with electrical isolation, enabling practical real-world applications.
- **Hysteresis in Control:** Introducing a dead band around the set point prevents system instability and component wear—a critical concept in all control systems.

Next Challenge: PID Control

Apply the **PID (Proportional-Integral-Derivative) control algorithm** to this system to achieve professional-grade temperature stability. PID control provides smoother regulation, faster response times, and better handling of disturbances compared to bang-bang control.

Resources & References

The full code, wiring diagram, component specifications, and documentation are available on the **original GitHub repository**: github.com/mikelix/arduino-temperature-controller. Additional resources include Arduino IDE docs, the LM35 datasheet, relay module specs, and PID tutorials for further study.

Questions & Discussion