An Automated Temperature Control System Based on the PIC18F87K22 Microprocessor

Chengyun Zhu

***Abstract*—This report outlines the design and implementation of an automated temperature control system using the PIC18F87K22 microprocessor. The system combines a keypad, an analogue temperature sensor, control mechanisms (fan and radiator), and an LCD display. The microprocessor processes data from the sensors and adjusts the control mechanisms to maintain a programmable target temperature within a programmable threshold. Real-time updates are displayed on the LCD. The fan is controlled using PWM by "Proportional plus Derivative Compensation". The software is programmed using the assembly language of the PIC18 family. This system is operational from $0°C$ to $50°C$, and the designed keypad has a theoretical response time limit of $11\mu S$.**

I. Introduction

AUTOMATED temperature-control is an important subject around us. For example, the electric radiators that helped us get through the winter need a way to regulate their temperature; the CPUs and GPUs on desktops require a robust and precise cooling system. The goal of this project is to construct a similar temperature control system using the PIC18F87K22 microprocessor on an EasyPIC PRO v7 development board.

The project emphasises the integration of hardware and software to achieve real-time temperature monitoring and control. This work not only addresses the technical challenges of designing a responsive and accurate system but also fosters a deeper understanding of modular programming, sensor interfacing, and embedded system design. By creating a practical and scalable product, this project aims to showcase the versatility of microprocessor-based solutions in addressing real-world problems.

II. Technical Design

A. High Level Design

The hardware design involves the integration of a keypad, a temperature sensor, a microprocessor, and actuators for temperature regulation. The software complements the hardware by providing efficient control logic and user feedback. The key to effective design lies in modularity.

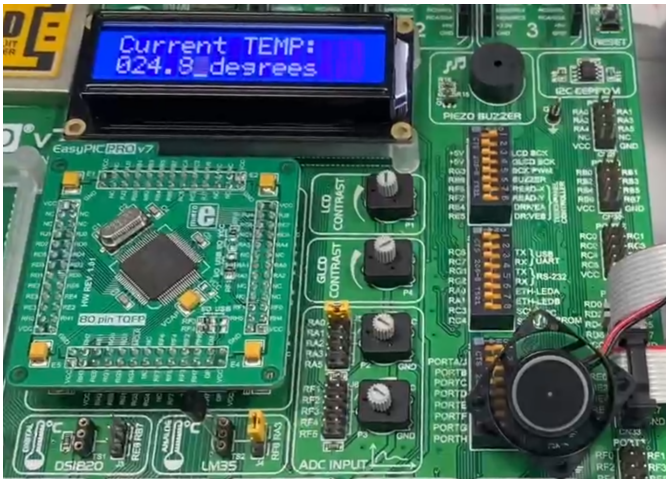

Fig. 1. The prototype of the automated temperature control system.

1) hardware design: The hardware high-level design is shown in Fig. 2. The PIC18F87K22 microprocessor serves as the central unit, interface with all other components. A 4×4 hexadecimal keypad is connected to PORTE to receive user input. The temperature sensor (LM35) is connected to the analogue input pin on PORTA (RA3), providing continuous temperature readings. The 2×16 LCD display, connected to PORTB (RB0, RB1, RB2, RB3, RB4, RB5), communicates key information such as current temperature, difference between current temperature and target temperature, and the first derivative of temperature with time. The LCD also serves as an interface when user input is detected from the keypad. The control mechanisms, which include a fan and a radiator, are connected to PORTD and PORTG, respectively, and are responsible for adjusting the ambient temperature. These actuators are controlled using Pulse Width Modulation (PWM), and a TIP3055 transistor is used as a relay. It is not possible to connect the fan and radiator directly to the microprocessor, as the microprocessor cannot provide enough power for these components. In fact, the fan we used requires $12V$, while the microprocessor can only supply $5V$.

2) software design: We use git for collaborative coding and our git repository is at https://github.com/cid02142311/microprocessors. The main development

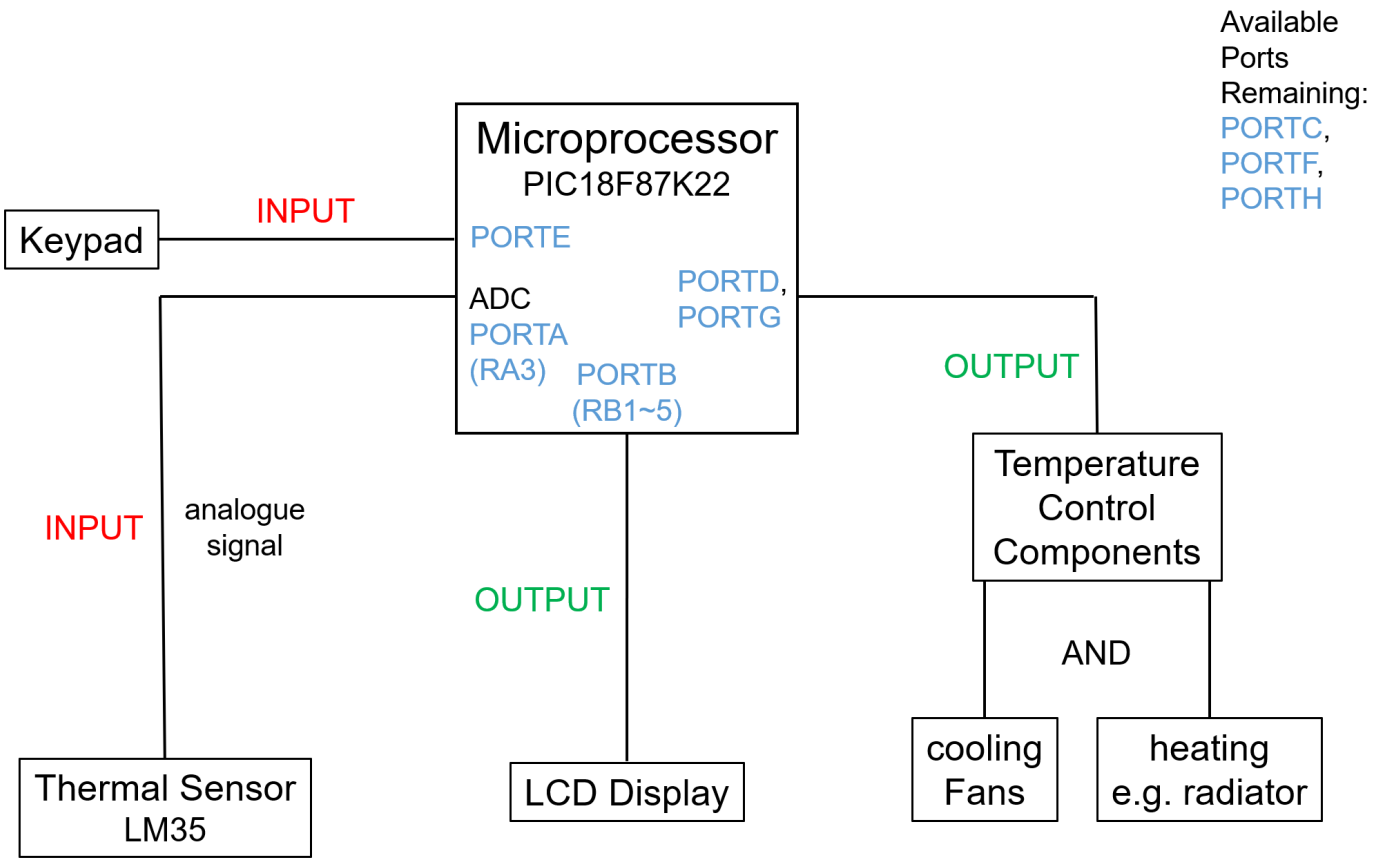

Fig. 2. Hardware block diagram. The modular design is clearly shown that each component occupies different ports.

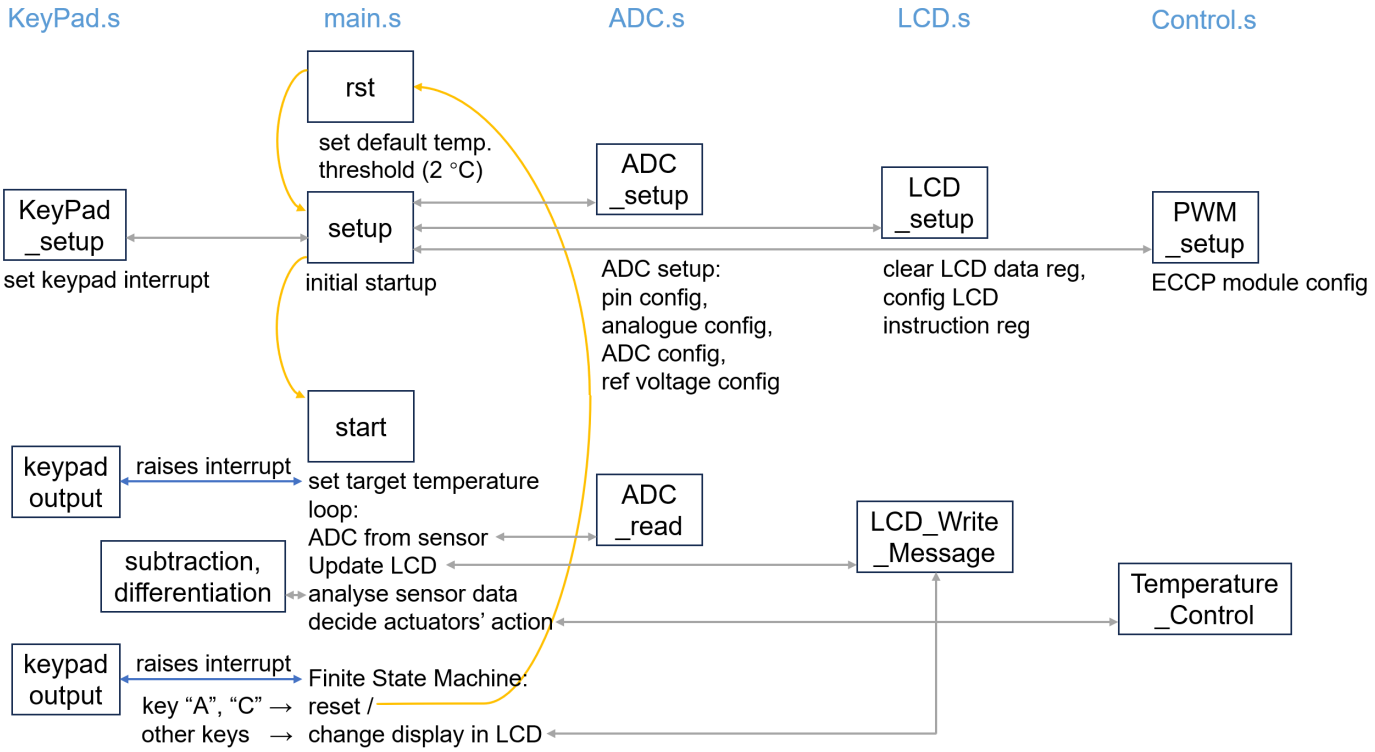

Fig. 3. Software flow chart. Grey lines represent instructions CALL or BRA, and orange lines represent instruction GOTO. Blue line represents operations raised by interruption.

takes place in the master branch. Fig. 3 is a brief flow chart of the code.

When the system is turned on, the microprocessor first processes the function “rst” and “setup” in the main.s file. At this stage, the temperature threshold is set to $2°C$ by default. We do not ask the user to set this threshold at startup, as $2°C$ is tested as the most appropriate. However, this threshold is still user-programmable later. In the “setup” function, all the components used are configured. The keypad interrupt is set using the internal clock “T0CON”. The Analogue-to-Digital Converter (ADC) for temperature sensor is configured, by setting pin RA3 as the conversion pin and $4.096V$ as the reference voltage [1]. This reference voltage is selected after careful calculations so that the output after conversion is exactly the voltage (in mV) produced by the temperature sensor. The initial setup of the LCD involves clearing its data registry and configuring its instruction registry. The PWM is also configured using the internal Enhanced Capture/Compare/PWM (ECCP) module. This involves setting parameters such as “CCP2CON”, “CCPR2L”, and another internal timer “T2CON”.

Once the initial setup is complete, the programme transitions to the “start” function, where user input is captured. At this stage, the LCD displays the prompt “Enter TEMP:” on its first line, prompting the user to set the target temperature by pressing three keys sequentially. For example, to set a target temperature of $25°C$, the user presses “0”, “2”, and “5”. The LCD updates its second line to display “025.0 degrees.” This process is managed using a keypad counter, which is initially set to 3. Each detected key press decreases the counter by 1. When the counter reaches 0, the programme progresses to the next section. The counter is essential because the keypad detection process relies on interrupts, and the microprocessor cannot predict when an input will occur.

The main part of the software operates in three key stages: sensor input processing, LCD output communication, and control logic execution. Input from the sensors is analogue and is proportional to temperature; it is first converted to digital signal using the built-in Analogue-to-Digital Converter (ADC). Then, this information is processed by the microprocessor, recording not only the temperature readings but also its first derivative and the difference between real and ideal temperature, by calling individual modules named “subtraction” and “differentiation”. Since the microprocessor operates at a relatively fast speed, the first derivative is simply considered to be the difference from the last reading. The LCD is updated with real-time data, providing clear and concise feedback to the user. The control logic employs “Proportional plus Derivative Compensation” (PD) control, adjusting the power of the fan or radiator based on the extent of the temperature deviation from the desired range and the rate of temperature change. We used PWM to control a relay.

B. Finite State Machine

The main programme functions as a basic Finite State Machine (FSM). Users can press any key (except “A” and “C”) to view three parameters on the LCD: the current temperature, the difference between the target temperature and the current temperature, and the temperature derivative with respect to time. Since this report is intended for readers with a background in physics, a brief introduction to FSM concepts is provided below.

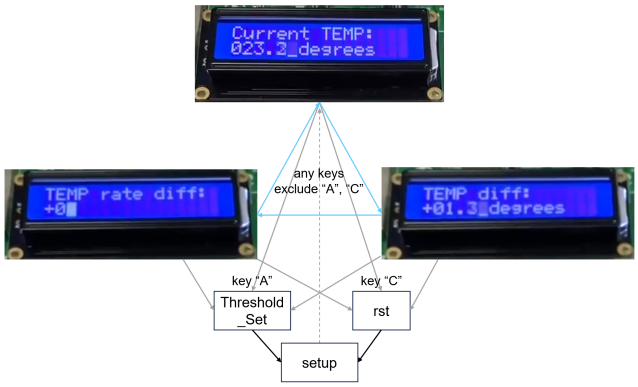

Fig. 4. Schematic of this basic finite state machine.

A Finite State Machine (FSM) is a computational model used to design systems that transition between a finite number of states based on external inputs. At any given time, the system exists in one state, and specific conditions trigger transitions to other states. FSMs are widely used in engineering and computational applications for tasks such as control systems, user interfaces, and signal processing.

In this project, the FSM is employed to manage the programme’s operational flow, facilitating the response to user input and enabling the display of various temperature data. Fig. 4 illustrates the relationships between the different stages of the FSM. The keypad used is a hexadecimal keypad; the keys “A” through “F” can be assigned to additional functionalities, as user input for temperature settings is restricted to decimal keys “0” through “9”. The key “C” is used for system reset, and the key “A” is used to change the allowed temperature threshold. The other 4 character keys are not assigned yet. They can be used for additional functionalities in the future.

C. Proportional plus Derivative Compensation Control

Proportional-Integral-Derivative (PID) control is a widely used feedback control mechanism in automation and process control systems. It combines three control actions: the proportional term (P) reacts to the current error by producing a control action proportional to the magnitude of the error, the integral term (I) accounts for the accumulation of past errors to eliminate steady-state offset, and the derivative term (D) predicts future error trends by considering the rate of change of the error. This is somewhat similar to the idea of second order differential equations: the integral term serves as the damping term, and the derivative term serves as the forcing term. By tuning these three parameters, a PID controller can achieve precise and stable control of a system, making it suitable for a wide range of applications, including temperature regulation, motor control, and industrial automation.

In this project, PID control is used to fine-tune the system's response and maintain the desired target temperature effectively. PID control does not require all three components; in this project, we used the proportional term and the derivative term. The integral term is not used because we set a threshold around the target temperature; besides, the fan can stop very quickly when its power supply is cut off. There is very little chance for the temperature to pass over the target temperature.

D. Keypad

The keypad we used is a 4×4 hexadecimal keypad, shown in Fig. 5. It features 8 pins: 4 pins connect the rows and the remaining 4 connect the columns. As a passive device, the keypad establishes a connection between a row and a column when a key is pressed.

In operation, low voltage is supplied to the columns and high voltage is applied to the rows. If a key in row 1 is pressed, the corresponding row will register a low voltage due to its connection to a low-voltage column. By alternating the voltage supply between rows and columns, the system can also determine the specific column of the pressed key, enabling accurate key detection. Given that the microprocessor operates significantly faster than the human action of pressing a key, nearly every key press can be detected with high accuracy. However, the keypad does have a theoretical performance limit, which is discussed in further detail in the performance section.

The input of the keypad is detected by raising the interrupts. Instead of telling the microprocessor to look at the keypad all the time, it is better for the microprocessor to take action until a change is detected.

Fig. 5. The 4×4 hexadecimal keypad used. It has 8 pins connected to PORTE.

The keypad provides very useful user input: for example, now we can set the target temperature, the temperature threshold as we wish. In addition, key "C" is assigned to the system reset, and key "A" is assigned to set the temperature threshold. (This is another process, because the threshold is set to $2°C$ by default, and the system reset does not change this.)

E. Temperature Sensor

The temperature sensor we used is LM35, which outputs an analogue signal. It is installed in a specific slot on the EasyPIC PRO v7 development board, which is actually pin RA3. The voltage of its output is proportional to the temperature: its output is $0mV + 10.0mV/°C$ [2]. If the room temperature is $0°C$, the output is $0mV$. If the room temperature is $20°C$, the output is $200mV$. The signal is first converted to a digital signal using an Analogue-to-Digital Converter (ADC).

The digital signal produced by the ADC is in hexadecimal form. In order to show the measured results on the LCD in an understandable way, a conversion from hexadecimal to decimal is required. This is done by performing several 24 bit × 8 bit multiplications. The 24 bit number is $0x418A$. After this conversion, the digits are added with 0x30, becoming ASCII codes, and then sent to the LCD display.

F. LCD Display

The LCD we used is a 2×16-character LCD. It has data registry and instruction registry; normally the instruction registry is not changed, except for some situations, i.e. changing writing position from first line to second line.

G. Temperature Control Components

The fan used is model 02510SS-12M-AA, produced by NMB Technologies. Fig. 6 shows the fan used.

Fig. 6. NMB 02510SS-12M-AA fan used.

This is actually a very small fan, nearly impossible to have any real cooling effect. However, since we are only making a prototype, this fan is acceptable at this stage. Also, because we had already used a transistor as a relay, it is very simple to switch to bigger fans, as long as the voltage requirement is also $12V$ DC.

The radiator can simply be a resistor. It also requires a relay and an external power supply.

H. Relay and Pulse Width Modulation

For the temperature control components, relays and external power supplies are needed. This is because the microprocessor does not have the ability to deliver a large power output. In real practice, after applying a workload to one pin (ideally $5V$), the voltage of that pin may drop to $3V$ or lower, depending on the workload. Thus, a relay is needed to control these powerful components.

Initially, we used the TIP3055 transistor as the relay. This is a bipolar NPN transistor that allows us to control the external $12V$ power supply using a $5V$ pin. However, a significant distortion was observed in the output signal. After a discussion with professor Neil, we switched to a MOSFET transistor 2N7000.

PWM control is achieved using the internal ECCP module. After configuring the parameters "CCP2CON", "CCPR2L", and the internal timer "T2CON", the PWM is ready to use. "CCPR2L" is the duty cycle registry. For example, if "CCPR2L" is assigned by 0xFF, the fan is on all the time. If "CCPR2L" is assigned by 0x7F, the fan is on and off at a given frequency determined by the timer setting.

Fig. 7. Graph illustrating PWM. Note: this graph has some trigger problem.

Fig. 7 shows one situation in which the PWM control is running. Please note that this graph has some trigger problem. However, the PWM feature is clearly shown in this graph: the yellow line shows the voltage at pin RD0, and the green line shows the voltage measured at the fan. The microprocessor produces a periodic on and off pattern, and the experiential start and decay pattern caused by the eddy current in the fan is clearly shown. The less the duty cycle, the slower the fan.

III. Results and Performance

Combining PWM and PID control, the automated temperature control system works perfectly as planned. A short video recording is provided here. (College account required to visit this sharepoint link.)

The calculated PID result is a number between 0x00 and 0xFF, and is passed into the duty cycle registry. If the temperature difference is greater than $10°C$, the duty cycle registry is always 0xFF. If the temperature difference is less than the threshold (default $2°C$), the duty cycle is 0x00. Because the microprocessor works at a relatively fast speed, the derivative term is assigned with much greater weight.

The performance of the system has several theoretical limits: keypad response time, LCD response time, working temperature, average power, maximum power, cooling & heating power, energy efficiency. The limits that can be calculated at this stage are listed below.

- **Keypad Response Time:** The keypad interrupt is raised approximately each $1.6\mu S$, since the clock rate is $62.5kHz$. The theoretical keypad reading processing time based on the length of the code is approximately $9.4\mu S$. This is a rough estimate,

considering that the frequency of the microprocessor is $64MHz$ [1]. The combined keypad response time is approximately $11.0\mu S$.

- **LCD Response Time:** The Liquid Crystal Display (LCD) itself requires longer delays, because it cannot update very quickly. The calculated response time for 1 byte of information is approximately $40.5\mu S$. For 16 characters, the response time is approximately $650\mu S$.
- **Working Temperature:** This is the union set of working temperature of all components. The ambient temperature under bias for the microprocessor is $-40°C$ to $+125°C$ [1]. The working temperature for the LM35 temperature sensor is from $-55°C$ to $150°C$ [2]. The ambient temperature for the fan used is from $-10°C$ to $70°C$ [3]. The typical working temperature for LCD is from $0°C$ to $50°C$. Overall, this system is operational from $0°C$ to $50°C$.

The factors such as energy efficiency, maximum power cannot be determined at this stage. This is because the temperature control components (fan and radiator) must be replaced with different ones.

IV. Updates, Modifications and Improvements

As an alpha prototype, our current system successfully demonstrated its availability. There are still a lot of places in hardware and software that require further improvement.

In terms of hardware, multiple temperature sensors can be added. Currently, there is only one temperature sensor, and its reading cannot be cross-referenced with other sources. In addition, the fan and radiator need to be replaced with more powerful ones.

In terms of software, the idea of modularity requires further improvement. Currently, there are still 507 lines of code in "main.s". Although most of them make up the FSM, some lines should be moved to their relevant submodules.

In the future, using the idea of FSM, more states and more information can be displayed on the LCD; there are still 4 character keys on the keypad that are not assigned yet. Other information, such as system alerts, can be displayed.

With more functionality and complexity, a Failure Mode and Effects Analysis (FMEA) should be done. This is a systematic approach used to identify potential failure modes within a system, assess their impact on performance, and prioritise corrective actions to mitigate risks. By examining each component or process step, the FMEA helps uncover weaknesses, predict their consequences, and implement strategies to improve reliability and safety. This method is widely used in engineering, manufacturing, and quality management to proactively address vulnerabilities and improve the robustness of the system.

V. Conclusion

The automated temperature control system represents a comprehensive application of the PIC18F87K22 microprocessor to solve a practical problem. By combining precise sensor data, proportional plus derivative compensation control, and an intuitive user interface, the system provides an effective solution to maintain stable environmental conditions. This system is operational from $0°C$ to $50°C$, and the designed keypad has a theoretical response time limit of $11\mu S$. This project not only highlights the potential of microcontroller-based systems, but also serves as a foundation for future enhancements. The design is aligned with the principles of modularity, efficiency, and scalability, providing significant educational and technical value.

VI. Product Specifications

- **Microprocessor:** PIC18F87K22
- **Development Board:** EasyPIC PRO v7
- **Keypad:** 4×4 hexadecimal keypad
- **Temperature Sensor:** LM35
- **Display:** 2×16-character LCD
- **Fan:** NMB 2510SS-12M-AA
- **Relay:** MOSFET transistor 2N7000
- **Working Temperature:** $0°C \sim 50°C$
- **Programmable Factors:** target temperature, temperature threshold

Acknowledgment

I acknowledge the use of ChatGPT 4o (Open AI, https://chat.openai.com). It is used for initial project idea inspiration and proofreading my final draft. I also acknowledge the use of writefull, a build-in language checker in Overleaf LaTeX editor.

I acknowledge the reuse of some sentences from my previous proposal on this project.

I must express my thanks to Professor Neil and the demonstrators in the lab. They helped us a lot with this project, and with the assembly language in particular.

Thanks to Amiya. She accompanies me all the time during vacation, providing invaluable emotional support during my down times.

REFERENCES

[1] Microchip Technology Inc., “PIC18F87K22-Family-Data-Sheet”, DS30009960G, revised 15 April 2024.
[2] Texas Instruments, “LM35 Precision Centigrade Temperature Sensors”, SNIS159H, August 1999 – Revised December 2017.
[3] NMB Technologies, “DC Axial Fans 02510SS”, MinebeaMitsumi.
[4] Microchip Technology Inc., “N-Channel Enhancement-Mode Vertical DMOS FET”, DS20005695A, revised 28 February 2020.
[5] Zhu Chengyun, “Automated Temperature Control System”, 25 November 2024.