Thermoelectric Module Temperature Stability using Bidirectional Proportional-Integral-Derivative (PID) Controller (2022)

Sidney Kantor, *Electrical Engineering, University of North Dakota, Member, IEEE*

***Abstract*—Thermoelectric modules (TEMs) are solid state heat pumps that utilize the Peltier Effect to transfer heat, creating a temperature differential across two ceramic surfaces. Passing current through the device causes one side of the TEM to become cold while the other side becomes hot. This effect is reversed when the current passing through the device is reversed. A single-stage TEM can create temperature differentials as high as 70 degrees Celsius between the cold surface and hot surface of the TEM. These devices have applications where precise temperature control is required, high reliability is a factor and/or weight and space constraints exist. This paper demonstrates the design and implementation of a control system using a bidirectional proportional-integral-derivative (PID) controller, capable of both actively heating and cooling, which can maintain a setpoint temperature as well as compensate for ambient temperature disturbances.**

***Index Terms*—Compensation, control system, heat pump, Peltier effect, proportional-integral-derivative (PID), solid-state, temperature, temperature differential, thermocouple, thermoelectric device (TEM).**

I. INTRODUCTION

THERMOELECTRIC devices (fig. 1) are very interesting devices. The idea of a small solid-state device, with no moving parts, which can both heat and cool via heat transfer, by passing current through the device in one direction or another, definitely lends itself as a solution for a variety of niche applications requiring such characteristics. However, there are some things that need to be considered when developing a controller to drive these devices safely and efficiently. This paper demonstrates the design and implementation of a hardware and firmware solution for a bidirectional proportional-integral-derivative (PID) controller that can both actively heat and cool and maintain temperature stability around a setpoint as well as compensate for ambient temperature disturbances.

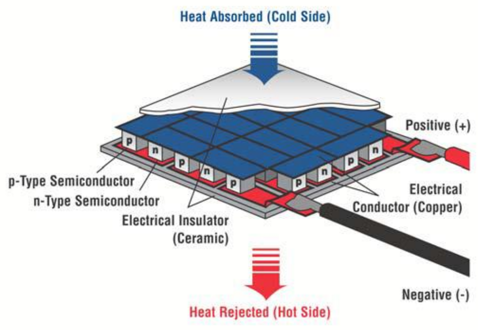

Fig. 1. Thermoelectric module.

II. ANALYSIS/REQUIREMENTS

To understand the operational behavior of TEMs, Laird Thermal Systems, a leading manufacturer of thermal management solutions, and the manufacturer of the TEM that was used in the design described in this paper, publishes the Thermoelectric Handbook (Laird Thermal Systems, 2020). This handbook contains a background on the physics of TEMs, the operation of the devices as well as approaches as to how to drive them safely and efficiently. After exhaustively reading the handbook, it was clear there were a few things that were going to need to be addressed specific to the TEM as well as the solution in general.

A. The TEM would require a bidirectional current source

TEMs will heat or cool one surface depending on the direction of the current through the device. Therefore, a hardware mechanism will be required in order to control the direction of the current.

B. TEMs require DC current with little ripple

TEMs require a DC current with little to no ripple otherwise their operational performance will be degraded as function of the ripple (1), where N is a percentage of current ripple, expressed as a decimal. Laird Technologies recommends no more than 10% ripple.

$$\frac{\Delta T}{\Delta Tmax} = \frac{1}{(1+N^2)} \tag{1}$$

C. Heat pumps require heat exchangers to dissipate the heat

TEMs are solid-state heat pumps and as such, require a heat exchanger to dissipate the heat in order create higher temperature differentials.

D. TEMs have maximum voltages, maximum currents, and maximum temperatures

To avoid damage to the TEM, there must be a mechanism to limit the maximum voltage, maximum current, and the maximum temperature of the TEM.

E. The system will require a sensor to read the temperature of the TEM

F. The system will require a user interface

The system will need to the ability to allow the user to interact with the system and set the requested setpoint temperature as well as indicate the setpoint temperature, the control temperature and any other information that may be pertinent to informing the user of the system's state.

III. Prototype/Design/Implementation

A. Overview

The first step in the process was experimenting with the TEM using a bench top power supply. This allowed for getting a feel for its operational behavior. It was obvious that a heat exchanger was going to be required in order to achieve greater temperature differentials of the TEM. Once the ability to heat/cool the TEM was established, a temperature sensor was required for feedback and a thermistor was chosen for the task. In order to drive the TEM with both positive and negative currents, an H-bridge configuration of MOSFETs was used. In addition to the MOSFETs used in the H-bridge, another MOSFET was used to provide linear control of the current delivered to the TEM. At this point it was time to design a PID controller in Simulink that could interface with an Arduino's hardware. This allowed Simulink to control the heating/cooling of the TEM as well as reading the temperature from the thermistor. This also allowed for tuning the PID parameters to ensure a fast but stable response of the system. Finally, the user interface was designed which includes an LCD for display, LEDs for system status and a 10-turn potentiometer for precisely setting the setpoint temperature over a broad range of values. After experimenting with the different components of the system, a system block diagram was created (fig. 2).

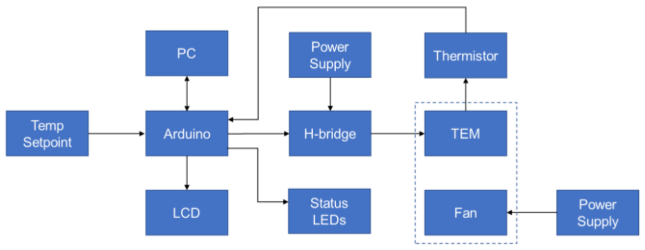

Fig. 2. System Block Diagram.

B. Mechanical

A salvaged heat sink with attached 12V DC fan worked well for the heat exchanger. The bottom side of the TEM was attached to the top of the heat sink using a homemade thermally conductive epoxy consisting of a two-part epoxy doped with aluminum oxide. Aluminum oxide is an extremely fine powder and is dangerous if inhaled, therefore, a mask and the proper precautions need to be taken when handling this powder. This ensured good heat transfer from the TEM to the heat sink. The heat sink and fan combination would require a base for support and also to ensure good air flow. The support base was designed in Fusion 360 and printed on a 3D printer (fig. 3). The support base was designed with four equally spaced horizontal 5mm holes to house LEDs which will be used for indicating the different states of the system such as cooling, heating, steady state and responding. Finally, a thermistor was attached to the top of the TEM, using the same thermally conductive epoxy, to measure the temperature for feedback.

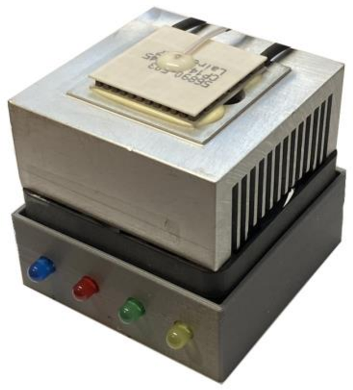

Fig. 3. Heat sink with TEM, thermistor, fan, and base with LEDs.

C. Temperature Feedback

The thermistor that was used was a 100K BAPI sensor. Thermistors are resistors that change their resistance as a function of temperature. A typical thermistor output curve is non-linear. The datasheet contains a table that maps resistances to temperatures. This lookup table could be used in with two different approaches. One approach is to use the lookup table and simply round up or down depending on what resistance was closest to the measured resistance. This approach is not a fully accurate, though. A second, and much more accurate approach, albeit more complex, is to use the lookup table but instead of rounding up or down, use linear interpolation between the points. Although either of these approaches could have been used, I chose to import the data into excel and curve fit the data to a logarithmic function (2). This proved to be the easiest approach as well as being sufficiently accurate for this project.

$$T(r) = -20.64 * \ln(\mathrm{r}) + 266.37 \tag{2}$$

D. Driving the TEM

An H-bridge circuit (fig. 4) was chosen to support the ability to reverse the current driving the TEM. The H-bridge consists of four logic level n-channel MOSFETs with the high-side MOSFETs driven by a gate driver to ensure the MOSFETs turn on fully. P-channel MOSFETs could have been used for the high-side switching without the use of the gate drivers, however, n-channel MOSFETs tend to have lower $R_{DS(on)}$ and are therefore more efficient. The H-bridge is controlled via digital outputs from an Arduino Uno and care had to be taken to ensure both branches of the H-bridge were fully off before turning on any single branch on, otherwise a short circuit would occur.

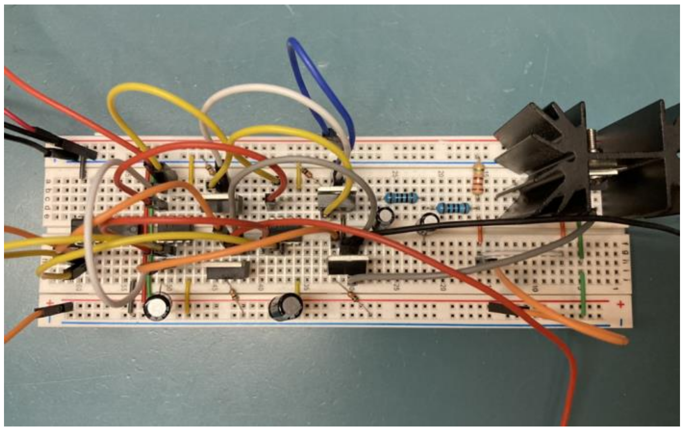
Fig. 4. H-bridge and series pass transistor.

Because the TEM needs to be driven by a DC current source with minimum ripple, a power MOSFET attached to a heat sink was used. This power MOSFET was operated linearly and allowed varying the current delivered to the TEM. This is not the most efficient approach, however for this project it will suffice.

The ATMega328P microcontroller on the Arduino does not have any built-in DACs. Therefore, a PWM signal was used to linearly control the power MOSFET. In order to smooth out the PWM signal do ensure a DC current source with minimum ripple, two cascaded RC filters were used smooth out the PWM signal.

E. Control system

Simulink was used initially to control the system. An add-on for hardware support for Arduino was installed that allows Simulink to control the output pins and read the input pins of the Arduino. A model of the system was created (fig. 5) and used a PID controller for the control systems strategy. Some additional blocks were added for signal conditioning purposes for the thermistor temperature sensor. This worked well for the intial analysis and tuning of the PID controller but this all had to ultimately be implemented in code on the Arduino.

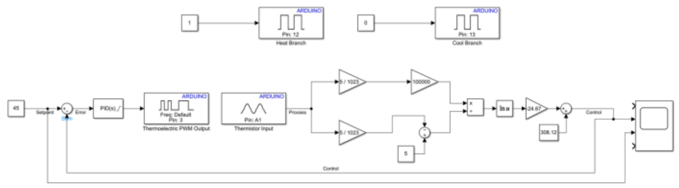

Fig. 5. Simulink block diagram.

Tuning the PID controller is challenging and requires some trial and error. What worked best was to first set the integral and derivative terms both to zero. Then adjust the proportional term until the temperature approaches its steady value without overshooting and without any oscillation. There will be some steady error at this point. The next step is to add in some integral term. This will eliminate the steady state error. Finally, tweaking the derivative and possibly the other terms slightly will allow the PID controller to reach steady state very quickly and with minimal overshoot and minimal steady state error. It's important to mention a caveat with the derivative term. If there is low level noise in the system, then the deriviatve term can cause more harm than good. Therefore, whenever using the derivative term be sure to filter the signal using a moving average (or filter type of choice) to ensure the low level noise is eliminated. The final gain values, after tuning, for the PID terms were $K_p = 10$, $K_i = 0.8$, and $K_d = 10$.

Once the final system was running entirely on the Arduino, the system was hit with an arbitrary step function and the input, output and time was logged. This data was imported into MATLAB's System Identification app and MATLAB was then able to estimate a system transfer function (3) with an accuracy of around 98%.

$$T(s) = \frac{40.12s^2+173.7s+245.4}{s^3+1538s^2+742s+244.5} \tag{3}$$

The bode magnitude and phase diagrams for (3) can be seen in fig. 7. The root locus and the response plot for (3) can be seen in fig. 8. The time domain repsonse plot for (3) can be seen in fig. 9 and shows a fast response with some overshoot and no steady state error. Rise time was 2.91 seconds, overshoot was 15.1 % and steady state was reached in 13.3 seconds. The response was slightly underdamped with slight overshoot, however, this allowed for faster rise and fall times and therefore, was an acceptable compromise.

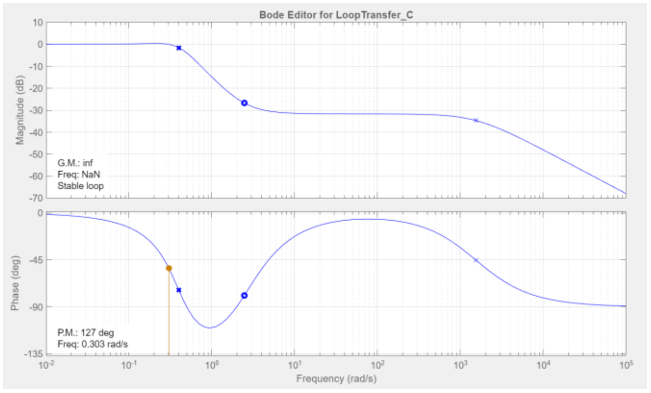

Fig. 7. Bode magnitude plot and phase diagram.

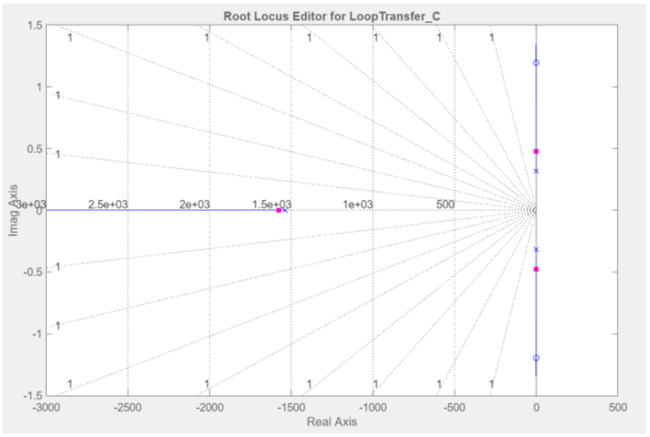

Fig. 8. Root locus plot.

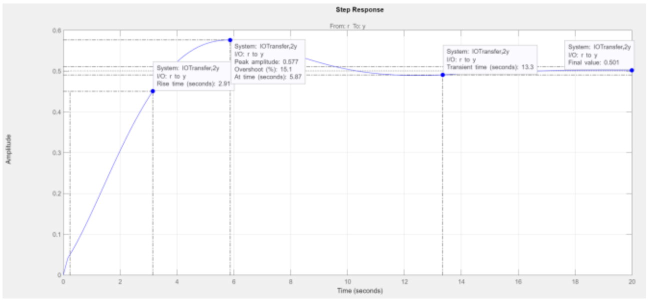

Fig. 9. Step response.

F. Human machine interface (HMI) (fig. 10)

1) **10-Turn potentiometer**
   To set the temperature setpoint.

2) **Status LEDs**
   Four different LEDs were used consisting of a blue (cooling), red (heating), green (steady state), and yellow (responding). Different color LEDs have different forward voltage drops; therefore, each LED required a unique current limiting resistor.

3) **Liquid Crystal Display (LCD)**
   To display the setpoint temperature, the actual temperature, the mode, and the steady state status.

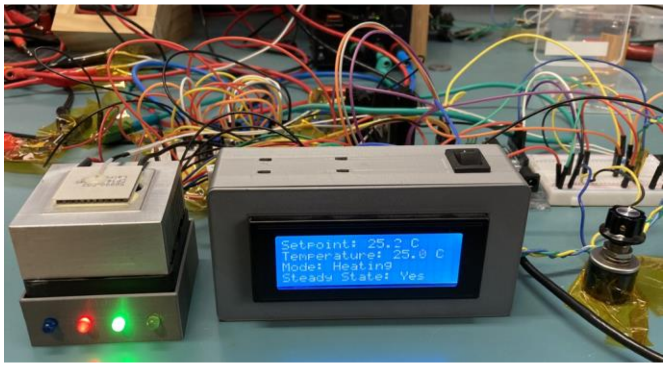

Fig. 10. System interface.

G. Power supply rails

Three different power rails were required (fig. 11). An 8V power supply rail with current limiting was used to power the TEM. An 8V to 12V power rail was used to power the heat exchanger fan and finally, a 5V power supply rail was used for all 5V logic requirements. Linear power supplies were used for all of the rails to ensure clean power sources.

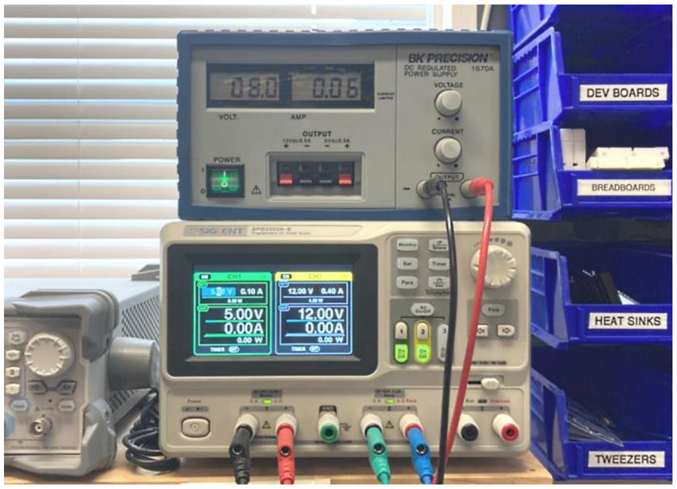

Fig. 11. Power supplies.

IV. Results

The results of the simulated step response, using the system transfer function (3), as identified by the System Identification application from the actual control system data, can be seen in fig. 12 and fig. 13. The results show a fast response with some overshoot and no steady state error. These responses are highly correlated to the actual system response which was reassuring.

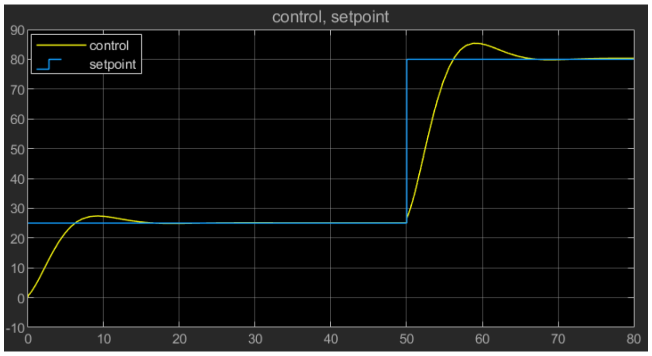

Fig. 12. Step response from 25.0° C to 80.0 ° C.

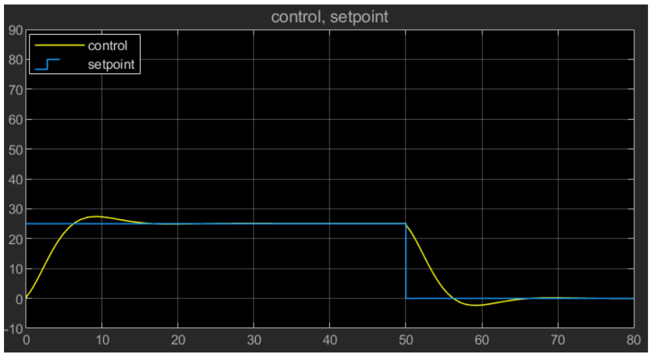

Fig. 13. Step response from 25.0° C to 0.0 ° C.

The system allowed the user to set the temperature setpoint via a 10-turn potentiometer to a temperature ranging from 0° C to 80° C. The LED indicators and the LCD displayed the status of the system as expected with one exception; the serial protocol for communicating with the LCD had a bug and every so often it would fail, requiring a reset. This occurred infrequently and did not affect any of the trials. The system quickly tracked, met, and maintained any temperature set via the potentiometer as well as compensating for any disturbances. A thermal imaging camera was used to ensure the temperatures were reasonably accurate and can be seen in fig. 14 and fig. 15.

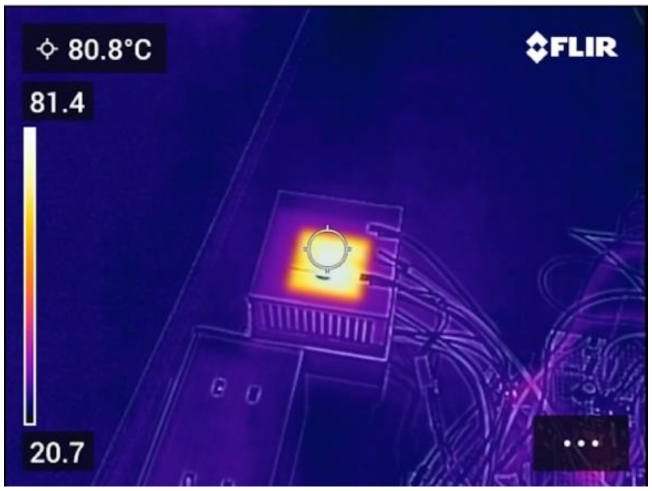

Fig. 14. Step response from ambient to 80.0 ° C (thermal image).

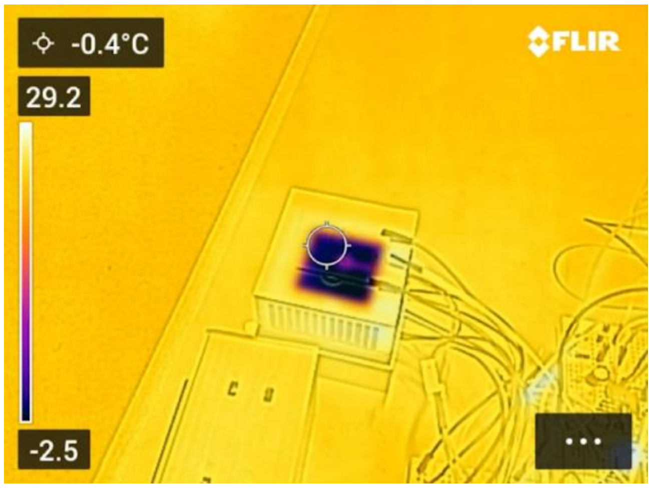

Fig. 15. Step response from ambient C to 0.0 ° C (thermal image).

V. Discussion/Conclusion

A. Improvements

Although the design did meet the requirements initially defined for the project, there were lessons learned along the way and if doing the project over, the following improvements could be made.

- Use of high frequency PWM with LC filter for efficient driving of the TEM
- Design and use a PCB
- More ergonomic user interface
- Ability to manage higher current for greater temperature differential
- Use of a digital temperature sensor to eliminate noise and for greater accuracy
- Proper fuses for safety
- All voltage rails derived from a single input power source
- Add over voltage, over current and over temperature protection at the hardware level

References

[1] *Thermoelectric Handbook*, Laird Thermal Systems, Morrisville, NC, USA, 2020.

Sidney Kantor has over 25 years' experience in the software industry as a software engineer. He held a role at his current company working in a research and development position, investigating the potential application of artificial intelligence within the context of the company's needs. Within this role he implemented a neural network using Python and Jupyter Notebook using only the native capabilities of the language. The neural network was trained on image recognition and was able to accurately identify handwritten numerical values zero through nine, regardless of the orientation. He is currently studying towards earning a Bachelor of Science (B.S) in electrical engineering from the University of North Dakota.