Temperature Regulated Test Bench for 28-Pin CerDIP Carriers

- Alex Zheng

Table of Contents

List of Figures

List of Tables

No table of figures entries found.

Abstract

The objective of building a temperature regulated test bench is to keep the resistance of integrated circuits constant under test, due to a temperature variant lab. The test bench allows housing of both a fan enclosure and a heater enclosure. This test bench allows users to see the current temperature of a chip and to set a desired temperature setpoint using a rotary encoder. Temperature is measured using a thermistor [1] with a voltage divider circuit and calibrated using the Steinhart-Hart equation. Control is performed by a Raspberry Pi Pico and Micro Python. Temperature control is implemented by PID via PWM, with a PMOS for amplification. CAD (SolidWorks) was used for modeling and Cura was used to slice and 3D-print the test bench design. This included the test bench, the fan, heater enclosures, and friction fit tests. To power the 12V fan and heaters, ac/dc, dc/dc, and level shifter converters were used. The current state of the prototype is that the physical model is finished and that the test bench can read and display temperature readings of chips, though more testing is required. The physical assembly of the prototype is ~70% done; with assembling the fan enclosure, fixing components onto the test bench, soldering of a new transistor, and the rotary encoder required. The code is ~65% done, with all functionalities defined, but no implementation of PID control nor the rotary encoder has been done.

Introduction

This research project was conducted over the past 4 months with the --- worked on by an undergraduate student: Alex Zheng. The problem this project aims to solve is the undesired resistance variation of chips under test due to a non-temperature-controlled lab setting. See Figure 1 showing temperature variation.

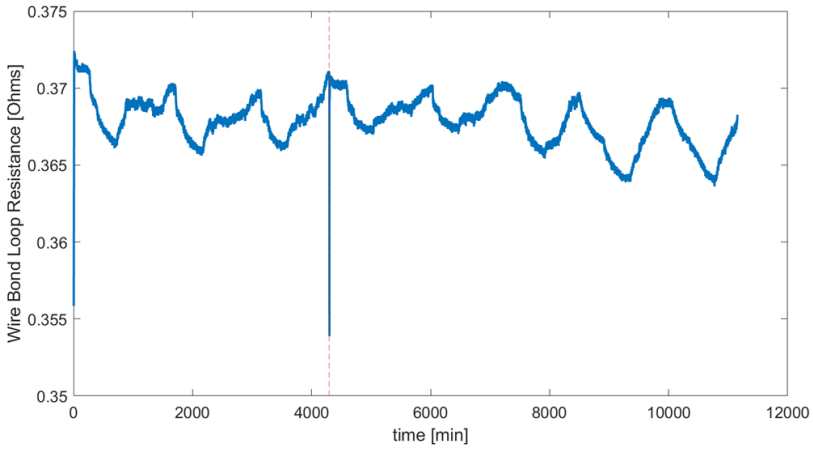

Figure 1. Image of Resistance vs. Time of a wire bond over several days due to the changing ambient temperature of the lab. Courtesy of ---

As a result, a test bench enabling temperature modulation is required. The test bench uses two 12V 70 °C PTC ceramic heaters with PID control to steadily regulate the temperature, with a desired uncertainty of ±1 °C. This control is implemented using a Raspberry Pi Pico in Micro Python. Temperature is sensed using a thermistor and a voltage divider circuit by the microcontroller. Thermistor-temperature calibrations were made using a hot plate, a reference thermocouple, and the Steinhart-Hart equation. The setpoint temperature is set by the user using a rotary encoder. The fan and heaters can be manually turned on and off, overriding the microcontroller control, using button switches. The heater enclosure uses a friction fit, as well as a wall thickness of 5 mm to decrease convection. To decrease thermal conductivity of 3D-printed plastics, "the optimal size of closure [wall thickness] was determined to be 10 mm, with no convection, and 6 mm, with possible convection", so a wall thickness of 5 mm was chosen [2]. In Figure 2, the concept model of the test bench with the heater enclosure, without heaters attached, is shown.

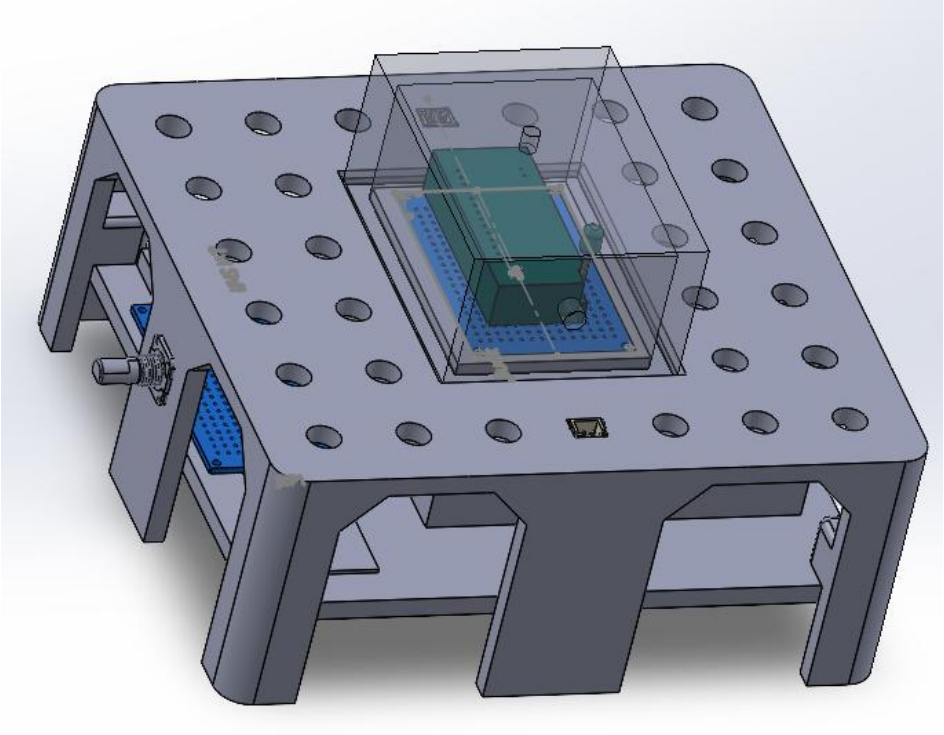

Figure 2. Concept model for the test bench in SolidWorks. Note: the heaters are not installed.

The second layer is a hardware layer to house the Raspberry Pi Pico, buck converter, level shifter, transistor, and connections. The constraint is that the test bench must be able to house the temperature control circuitry, with this extra circuitry not affecting its fundamental role as a chip test bench.

Method

Tools and Equipment: SolidWorks (latest version), Thonny IDE (latest version), Micro Python (latest version)

Functionalities:

- The test bench is self-contained, and all the hardware is self-contained to the test bench on the 2nd hardware layer.
- Thermistor sensor records the temperature of chip carrier to a desired uncertainty of 1 °C.
- Testbench provides PID temperature control to the chip carrier.
- Temperature setpoint can be set by a user using the rotary encoder. Setting the temperature is incremented and displayed on a four digit 7-segment display.
- The current temperature of the chip, sensed by the thermistor, is displayed on a four digit 7-segment display. If the rotary encoder is turned, the temperature being set is shown instead.
- Heater and fan enclosures are attached via a friction fit. Heater enclosure is 5 mm thick to reduce convection.
- Use of buttons to manually turn off power to heating/cooling elements.
- Use of epoxy, screws, and adhesives to fix components to test bench.

Test Bench States:

Below in Figure 3, the user-perceived functionalities are shown in a state diagram.

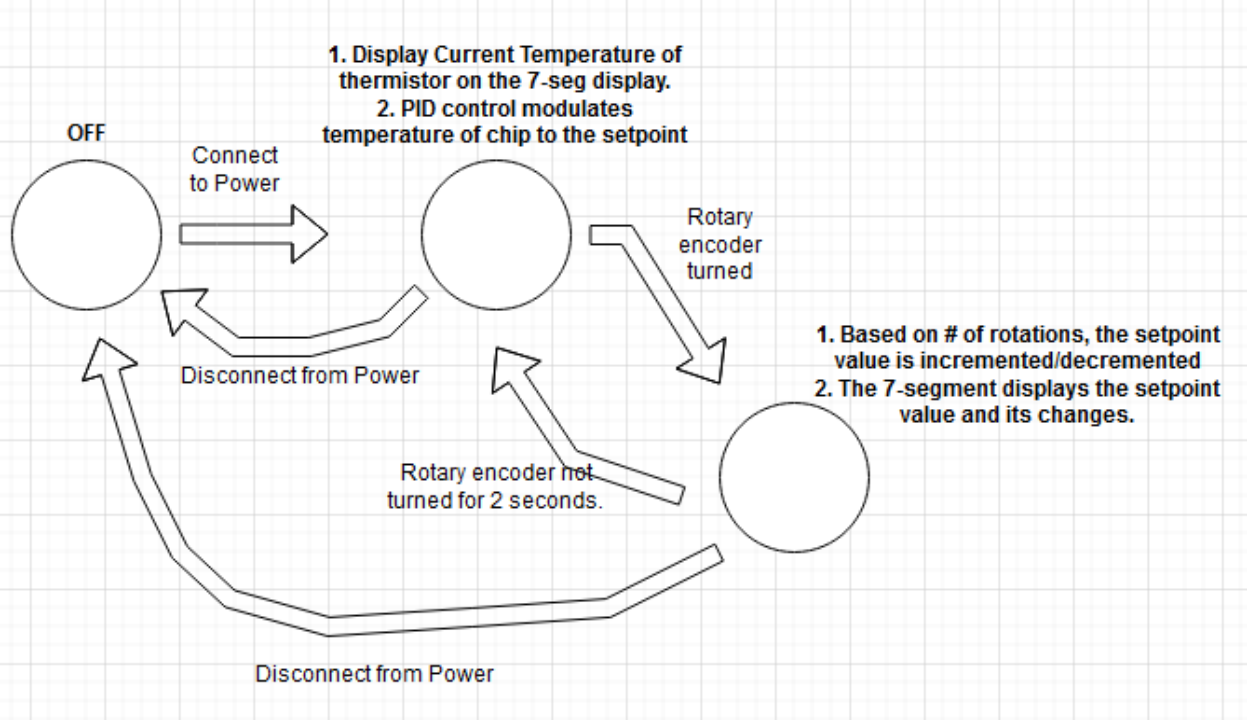

Figure 3. State diagram of testbench user functionalities

Thermistor Calibration:

Water heated up by a hot plate at various temperatures was used to gather calibration data of our thermistor. This simple method enabled higher specificity of our temperature range at which we calibrated our thermistor, allowing higher accuracy of the thermistor-temperature predictions in our proposed range of operation at 40 °C. Since our proposed set point of our heater enclosure is 40 °C, measurements near 20 °C, 40 °C, and 60 °C were taken. A thermocouple was placed in the same beaker as the thermistor, to serve as the reference temperature of the thermistor at a given resistance. This is because the temperature reading of the hot plate was inaccurate, leading to this decision to use a thermocouple as the reference instead. Using the Raspberry Pi Pico's ADC, 30 resistance values at 21 °C, 41 °C, and 67 °C were taken. The average resistance of the thermistor for the 30 measurements at each temperature is shown in Table 1.

Average Resistance of Thermistor at 21 °C	Average Resistance of Thermistor at 41 °C	Average Resistance of Thermistor at 67 °C
4.866 ohms	2.694 ohms	1.332 ohms

Table 1. Average resistance of the thermistor at 3 temperatures points for 30 replicates.

Using these 3 resistance and temperature pairs and the Steinhart-Hart equation https://rusefi.com/Steinhart-Hart.html , the calibration equation of the thermistor is given by:

$$T = \frac{1}{0.0028427564207573488 + 0.00033827116557912 * \ln R_t + 0.00000555342539196426 * (lnR_t)^3} \text{ [K]}$$

Then considering the dissipation factor of the thermistor [1], which is caused by its own resistive heating, the final temperature equation was obtained for the thermistor.

$$T = \frac{1}{0.0028427564207573488 + 0.00033827116557912 * \ln R_t + 0.00000555342539196426 * (lnR_t)^3} - \frac{1000 * I^2 R}{8} \text{ [K]}$$

- Where I is the current through the thermistor and R is the resistance of the thermistor. This equation is included in the code section as well.
- 8 mW/K is the dissipation factor of our thermistor.
- Multiplying by a factor of 1000, since our dissipation factor is in milliwatts.

CAD:

Doing the CAD and the physical design was the most time-consuming portion of the project.

Below in Figure 4 and Figure 5, one can see the initial concept sketches for the test bench. Note that these are very rough, preliminary sketches. The hardware is now on another bottom layer instead of on the side and the heater and fan enclosures are separate now.

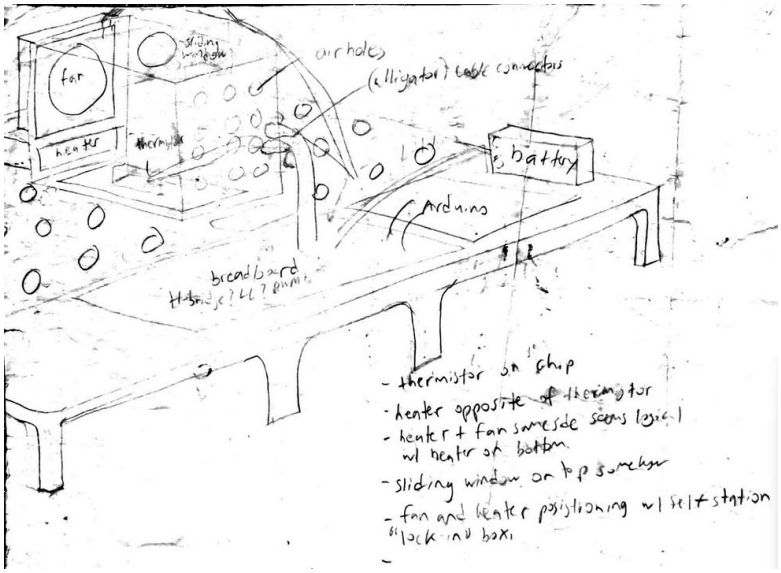

Figure 4. Isometric view of initial concept sketch of test bench design. In this initial design the fan and heater aren't separated, and the hardware is on the side of the testbench as opposed to under it.

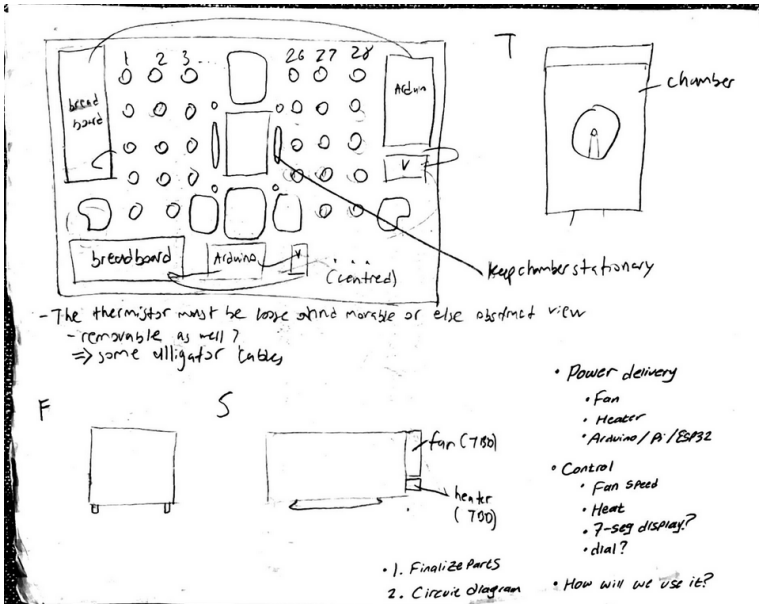

Figure 5. Top view of the initial test bench concept design.

Four preliminary models were made for the test bench. Below are the initial four concept designs for the 28-hole layout.

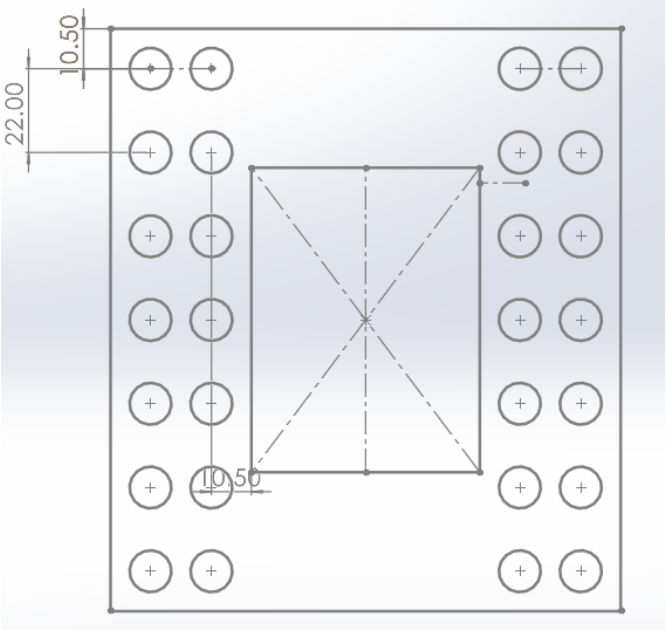

Figure 6. Option 1: Test bench layout

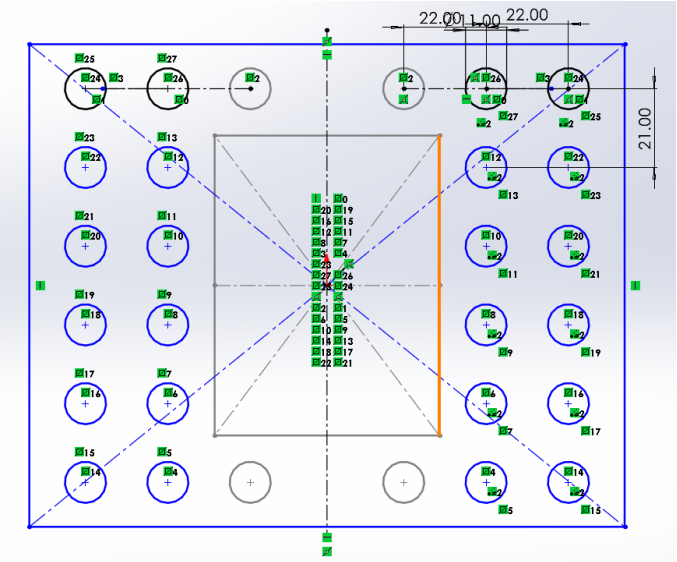

Figure 7. Option 2: Test bench layout

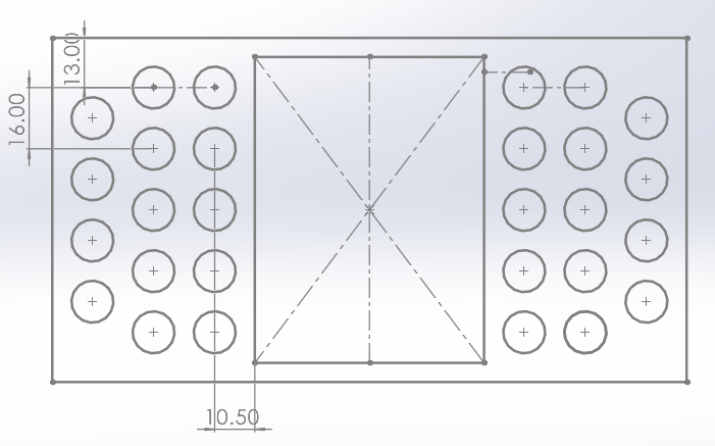

Figure 8. Option 3: Test bench layout

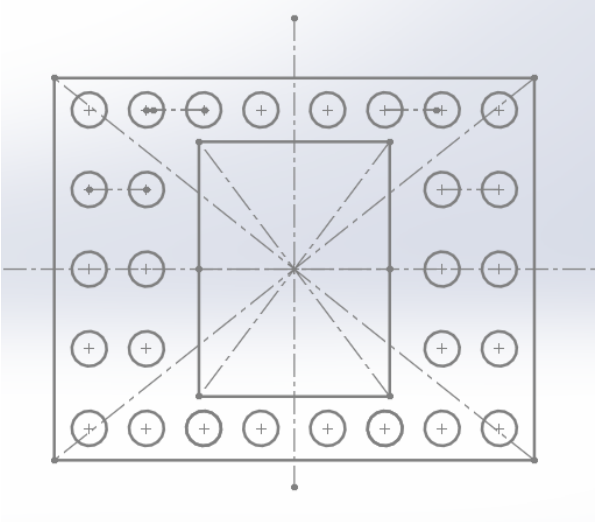

Figure 9. Option 4: Test bench layout

Option 2 of Figure 7 was chosen as our test bench layout due to its minimization of area, yet still logical numbering of the 28 holes for the CerDIP carriers.

The fan and heater enclosures were modeled next.

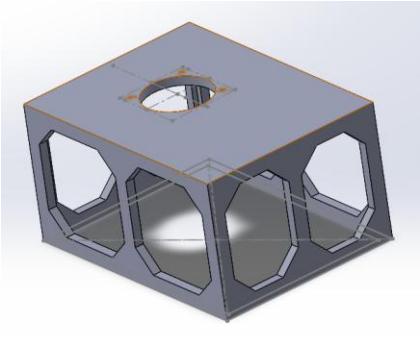

Figure 10. Fan enclosure

The fan enclosure features the fan hole on top and convection openings on its sides. The convection openings are aesthetically pleasing and feature bridging. Additionally, 3D-printing bridging tests were performed to test the capabilities of bridging.

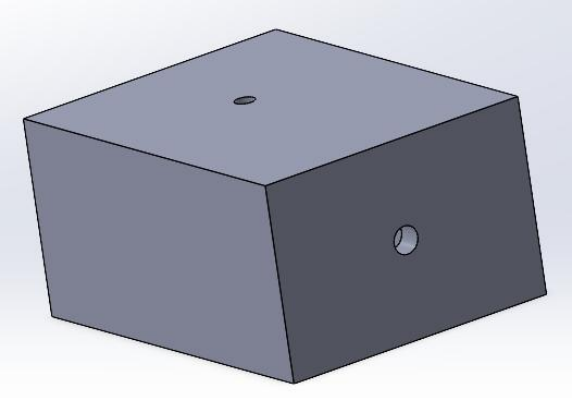

Figure 11. Heater enclosure

The heater enclosure features the thermistor insertion hole on top and a heater-wiring hole on the side. The thermistor hole was carefully measured such that it would be centered on the chip carrier when in use.

Then, the testbench model was made with Option: 2. Figure 12 shows the testbench model. The fits for the heater and fan enclosures, the fit for the proto board, the hardware layer, the JST connector holes, and the rotary encoder hole is implemented in this test bench model.

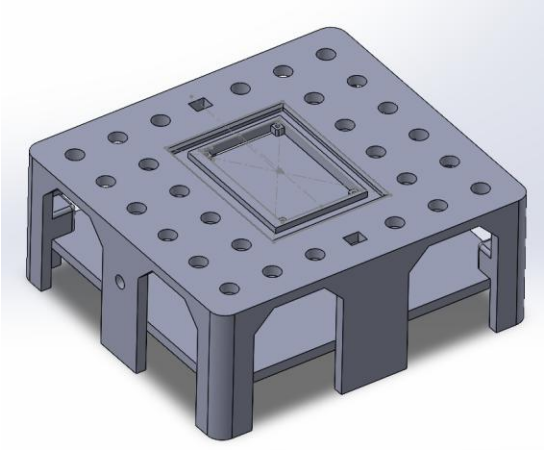

Figure 12. Current testbench model

After assembling the reference parts with the test bench in an assembly. The test bench model is shown in Figure 13. Figure 13 is the same as Figure 2.

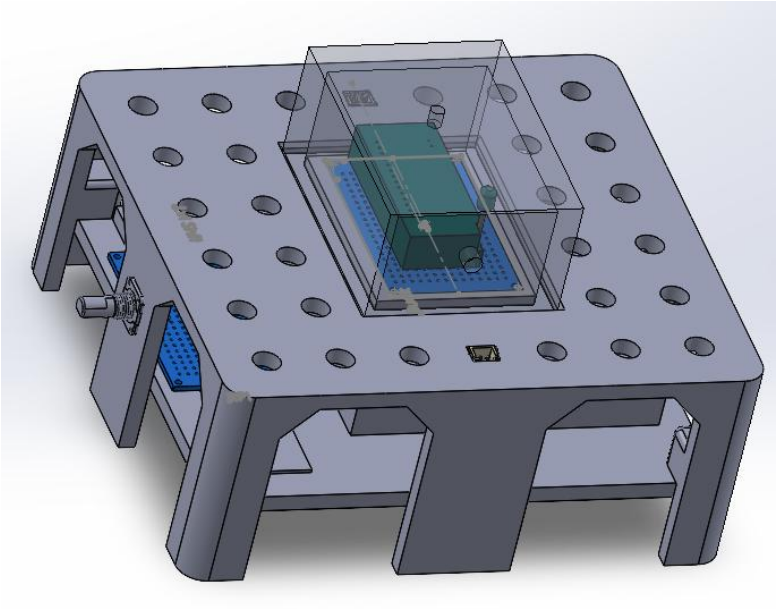

Figure 13. Current test bench final model.

The final model of the test bench.

Fabrication

Fabricating the project was divided into 3d-printing the test bench, soldering connections for the hardware components, attaching the hardware to the physical test bench, and implementing temperature control.

- The following challenges arose when trying to 3D-print the test bench:
    a. Ensuring ideal friction-fit tolerances of the enclosure-testbench, JST connectors, rotary encoder, and hardware layer friction fits. As a result, test/scrapped prints were needed to ensure an ideal fit and design.
    b. The protoboard has 14 holes along its width and the chip carrier takes up 9 holes. See Figure 14. This asymmetry required a horizontal translation of the protoboard on the testbench of 1.27 mm or 0.05", due to the pitch size of 2.54 mm of the protoboard.

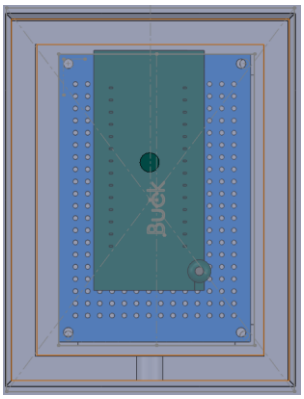

Figure 14. The asymmetry of the chip carrier on the proto board.

c. Structural weaknesses. Un-chamfered corners on the test bench broke off initially when force was applied to the test bench. Chamfered corners were added as a result. See Figure 15.

Figure 15. Chamfered edges of test bench.

- The following challenges arose when soldering the connections of the hardware:
    a. The amplification transistor initially used was a surface mount transistor. This was extremely difficult to solder to loose wires, as the metal pins of the transistor were pulled off. The SMD transistor was then attached to a protoboard to be more easily fixed onto a solderable connection, but the transistor did not work in testing, and it is believed to have been fried.
    b. The wiring may cause shorts with the CerDIP carriers if not properly tucked or fixed onto the hardware layer.
    c. The next step is to finish soldering the remaining components, such as the rotary encoder, and begin attaching the hardware to the test bench.

- The following challenges arose from attaching the hardware to the test bench:
    a. The chamfers (Figure 19) on the inside edges of the test bench interfere with the nuts of the CerDIP carriers, requiring an adhesive, such as epoxy to properly secure.
    b. The heater enclosure requires a material able to withstand ~70 °C. Since PLA has a glass transition temperature of "about 60 °C". An external print using PETG was printed for the heater enclosure (---) [3].

c. The abundance of wires for the hardware may cause shorts unexpectedly, these must be properly dealt with.

- The following challenges arose with implementing temperature control:
    a. To calibrate the thermistor, the Steinhart-Hart equation was used. To obtain the resistance values of the thermistor at three appropriate temperatures, a hot plate and a thermocouple (with an uncertainty of 1 °C) were used as the reference value. Water heated using the hot plate, at three different temperatures were measured and referenced with the thermistor and thermocouple. This presents some inaccuracies, which are presumed to be within the desired uncertainty of 1 °C for our system.
    b. When testing the thermistor at room temperature (~21 °C), reading the thermistor value using our voltage divider circuit and Raspberry Pi Pico gave us a value of ~30 °C. After discussion of this problem with ---, it was revealed by them that it was due to the dissipation factor of our thermistor [1]. The thermal energy that the thermistor generated from itself was not accounted for in the original thermistor formula; it had to be subtracted. After accounting for the dissipation factor, the thermistor gave much more accurate readings, though still needing additional attention. The temperature calibration equation in Method shows the dissipation factor, but further research is needed to verify this, as the temperature readings are still inaccurate.

<u>Programming Code Outline:</u>

The code consists of only 1 function only to functionalities such as data collection, data recording, the rotary encoder, and PID, requiring the setting of variables and the use of time that both must be used later and will otherwise be lost in the scope of a function or needlessly unnecessary to use a function for.

Functionalities:

- Display on 4 digit 7-segment display.
    - This function takes a 4 digit number representing the value to be displayed on the 4-digit 7-seg display and displays XX. XX.
- Data collection from thermistor.
    - This functionality reads the thermistor and generates the corresponding calculated temperature value.

- Data recording.
    - This functionality takes the values of temperature, resistance, and time, and writes it into a file onto the Raspberry Pi to use for PID tuning (to be done).
- Rotary encoder.
    - This functionality reads changes in the signal from the rotary encoder and changes the setpoint value (for the PID control). This functionality uses the "display" functionality to allow users to see their changes in the setpoint value.
- PID
    - This functionality calculates the area and rate of change of the error at a given rate (currently 0.5 s). Micro Python does not support SymPy so manual calculation of area (integral) and rate of change (derivative) is required. (PID is not tuned yet).

Below is the up-to-date code:

```
from machine import Pin, PWM, Timer, ADC
import time, math

# function displays the temperature or setpoint, up to 2 decimal places, on the 4 digit 7-segment display
def display(temp):

    if(temp>9999):
        return

    D4 = temp % 10
    D3 = (temp//10) % 10
    D2 = (temp//100) % 10
    D1 = temp//1000

    dp = Pin(0, Pin.OUT,Pin.PULL_DOWN)
    d1 = Pin(1, Pin.OUT,Pin.PULL_DOWN)
    d2 = Pin(2, Pin.OUT,Pin.PULL_DOWN)
    d3 = Pin(3, Pin.OUT,Pin.PULL_DOWN)
    d4 = Pin(4, Pin.OUT,Pin.PULL_DOWN)
    a = Pin(5, Pin.OUT)
    b = Pin(6, Pin.OUT)
    c = Pin(7, Pin.OUT)
    d = Pin(8, Pin.OUT)
    e = Pin(9, Pin.OUT)
    f = Pin(10, Pin.OUT)
    g = Pin(11, Pin.OUT)

    pins = [g,f,e,d,c,b,a]

    values = {0:[0,1,1,1,1,1,1],
            1:[0,0,0,0,1,1,0],
            2:[1,0,1,1,0,1,1],
            3:[1,0,0,1,1,1,1],
            4:[1,1,0,0,1,1,0],
            5:[1,1,0,1,1,0,1],
            6:[1,1,1,1,1,0,1],
            7:[0,0,0,0,1,1,1],
            8:[1,1,1,1,1,1,1],
            9:[1,1,0,1,1,1,1]
        }

    d3.value(1)
    d1.value(1)
    d2.value(1)
    d4.value(1)

    for index, value in enumerate(values[D1]):
        pins[index].value(value)
        d1.value(0)
        dp.value(0)

    time.sleep(0.002)

    d3.value(1)
```

```
    d1.value(1)
    d2.value(1)
    d4.value(1)

    for index, value in enumerate(values[D2]):
        pins[index].value(value)
        d2.value(0)
        dp.value(1)

    time.sleep(0.002)

    d3.value(1)
    d1.value(1)
    d2.value(1)
    d4.value(1)

    for index, value in enumerate(values[D3]):
        pins[index].value(value)
        d3.value(0)
        dp.value(0)

    time.sleep(0.002)

    d3.value(1)
    d1.value(1)
    d2.value(1)
    d4.value(1)

    for index, value in enumerate(values[D4]):
        pins[index].value(value)
        d4.value(0)
        dp.value(0)

    time.sleep(0.002)


# Below initializes variables and constants

adc = ADC(Pin(28))

# Resistance of the fixed voltage in the voltage divider circuit for the thermistor
R_fixed = 20
# Accurate value of the voltage feeding into the thermistor
V_p = 3.3265

# previous value variable for incremental encoder
prev = 0;
# The pin reading value A of the incremental rotary encoder signal
A = Pin(27, Pin.IN, Pin.PULL_DOWN)
# The pin reading value B of the incremental rotary encoder signal
B = Pin(26, Pin.IN, Pin.PULL_DOWN)

# Increment/decrement values for the rotary encoder; can change the value to increment/decrement more heavily/lightly
lookup_table = {(0,1): 0.5, (0,2): -0.5,(1,0): -0.5, (1,3): 0.5,(3,1): -0.5, (3,2): 0.5,(2,0): 0.5, (2,3): -0.5}

# Initial setpoint of the desired temperature
setpoint = 40
# boolean value allowing the buffer to determine between Displaying vs. Setting temperature
settingMode = False

# The pin outputting the signal to the field effect transistor (FET).
fet = Pin(22)
# frequency of the transistor pin. Note: extra research is necessary to determine optimal frequency.
fet.freq(732)

# The following two variables stores time values to get changes in time to record data for Temp vs Time
temp_time = 0
startTime = time.time()
prevTime = 0

# PID constants
area = 0
# previous error; used to get changed in error for D
PrevError = 0
# previous time value taken; used to get area time*temp for I
prevPIDTime = 0
K = 1
Ti = 1
Td = 1

while True:

    # Below variables read the thermistor
    V_Rfixed = (adc.read_u16()/65535) * V_p
    R_thermistor = R_fixed* ((V_p/V_Rfixed) - 1)

    # Dissipation factor calculation
    current = V_p/(R_fixed + R_thermistor)
```

```
    res_heating = 1000*(current**2*R_thermistor)/8


    # Steinhart-hart calibration for NTC thermistor, 1 measurement/second
    if(time.time()-temp_time > 1):

        Temp = (1 / (0.0028427564207573488 + 0.000338027116557912 * math.log(R_thermistor) + 0.00000555342539196426 *
(math.log(R_thermistor))**3)) - res_heating - 273.15
        temp_time = time.time()

    tempTime = time.time()


    # Recording Temperature, Resistance, and Time for tuning and data collection; records ever 0.5 sec.
    if((time.time() - prevTime) > 0.5 or prevTime == 0):

        with open('data.txt', 'w') as file:
            file.write(str(Temp) + ", " + str(tempTime-startTime) + ", " + str(R_thermistor) + "\n")

        prevTime = time.time()


    # Reading and processing from the rotary encoder
    AB = (A.value() << 1) | B.value()

    # if the rotary encoder is rotated ...
    if(AB != prev):  # Note: Do not set the setpoint to a negative value.

        settingMode = True

        # increment/decrement the setpoint
        setpoint = setpoint + lookup_table[(prev, AB)]

        # display the setpoint value
        display(round(setpoint*100))

        prev = AB
        lastSetTime = time.time()

    # Displays the setpoint temperature for 2 seconds after the rotary encoder is rotated...
    elif(settingMode):

        display(round(setpoint*100))

        if(time.time()-lastSetTime > 2): # Can change duration
            settingMode = False

    # otherwise, display temperature of thermistor ... (Default)
    else:

        display(round(Temp*100))


    # Calculating the duty cycle for our heater using PID control. PID tuning not done yet, so PID constants are undeclared
    # -- The time iteration for integral and derivative must be greater than the time per cycle of while loop.
    currentError = Temp - setpoint

    # Every 0.1 seconds the PID calculation is taken
    if((time.time() - prevPIDTime) > 0.1 or prevPIDTime == 0):

        timeChange = (time.time() - prevPIDTime)
        area = area + currentError * timeChange
        d = (currentError - PrevError) / timeChange
        prevPIDtime = time.time()
        PrevError = currentError

    output = round(K*(currentError + (1/Ti)*a + Td*d))

    duty = int(output)
    fet.duty_u16(duty)


# ASIDE:Below is the code used to calibrate the thermistor.
# with open('tempCalibration3.txt', 'w') as file: #cal1 at 41 C,  #cal2 at 67 C, # cal3 at 21 C
    #for i in range(30):
        # V_Rfixed = (adc.read_u16()/65535) * V_p
        # R_thermistor = R_fixed* ((V_p/V_Rfixed) - 1)
        # print("R: " + str(R_thermistor))
        # file.write(str(R_thermistor) + "\n")
        # time.sleep(1)
# break
```

Results and Discussion

The current state of the project is that the heaters can be powered, and the test bench can read temperature readings of a chip. See Figure 16, Figure 17, Figure 18, Figure 19 for pictures of the current state.

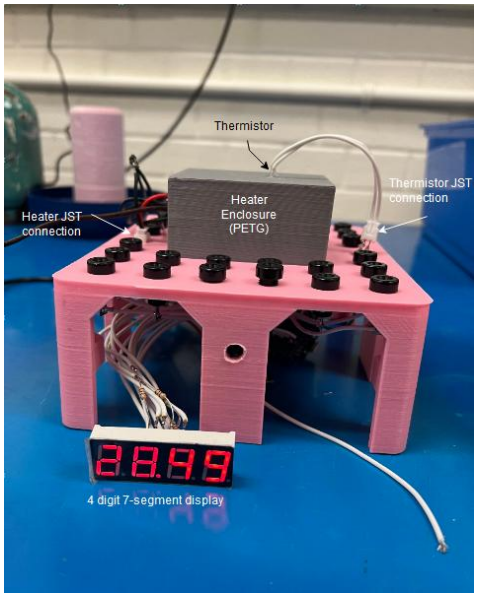

Figure 16. Temperature reading of chip carrier within enclosure when the heater is currently heating on.

Figure 17. Temperature reading from thermistor vs. thermocouple. Additional thermistor calibration is necessary.

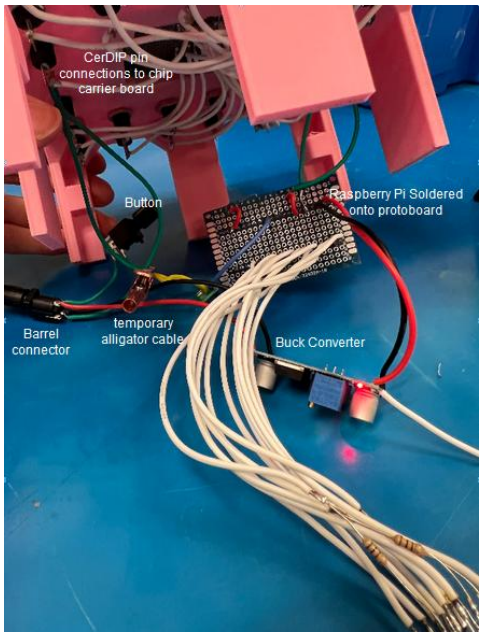

Figure 18. Underside of testbench.

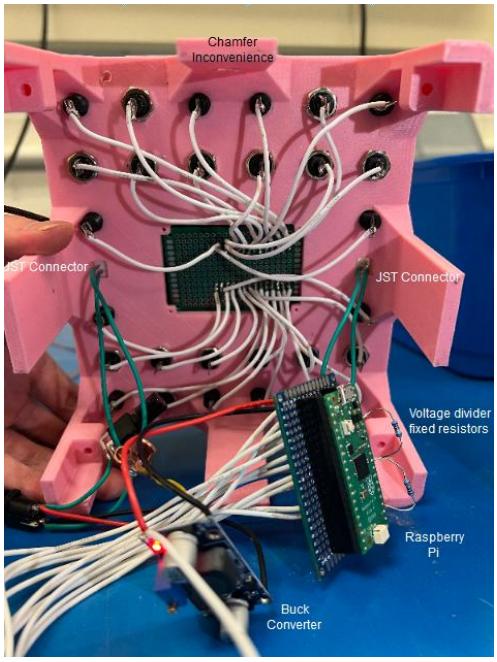

Figure 19. Underside of test bench (full bottom view).

From Figure 17, one can see the difference between the temperature reading from the thermistor vs. the thermocouple. This unexpected difference is something that must be further investigated.

Based on the current state of the prototype, I **do** think that this prototype can achieve the desired functionality of temperature control. There are currently no big problems that are bottlenecking us. After this

is done, I would ideally recommend that a final finished prototype be made if time allows. I think that this prototype possesses inconveniences such as chamfers interfering with the CerDIP carriers (Figure 19), a high potential for shorts due to un-heatshrinked wiring, and a general unaesthetic appearance due to soldering mistakes made. I think these inconveniences can be solved quickly or ignored, but if time allows, a second product would be nice.

The limitations and challenges faced were primarily in the CAD. The CAD was extremely time-consuming, taking up more than half of the time spent on the test bench. This took away time from implementing all the functionalities of the hardware, control, and testing more rigorously, leading to unfinished work which is not tested properly. The PID control, something critical to temperature regulation, was unfortunately not completed. Additionally, 3D-printing the test bench was resource intensive due to my personal inexperience.

For future research, the following areas must be researched and implemented (in no order):

- Investigation of discrepancy between thermistor reading and the actual reading (thermocouple). This may be still due to the dissipation factor, and I think additional research is needed in this area.
- Obtain a new transistor and resolder it for the connection between the Raspberry Pi Pico and the power supply for temperature control.
- Adhering the CerDIP carriers affected by chamfers to the testbench using epoxy.
- Tuning the heater for PID control and researching an optimal pin frequency for the duty cycle.
- Testing PID control.
- Fixing hardware components to the testbench (protoboard, JST) and the hardware layer (level shifter, buttons, barrel connector, 7-segment display) in an aesthetic and functional manner. Verify that shorts cannot occur; potential resoldering may be necessary.
- 3D-printing the fan enclosure and screwing the fan on.

Skills developed include:

- SolidWorks modeling. Designing and modeling the test bench made me heavily more familiar with SolidWorks features, assemblies, and parametric design.

- 3D-printing. Printing many test prints to test tolerances, and discussion of 3D-printers and how the properties of 3D-printing materials affect our project (glass transition temperature of PLA) with --- greatly increased my knowledge of 3D-printing.
- PID control. Learning about temperature control and how PID could be used to solve the temperature regulation problem helped me learn about tuning PID controllers and the theory behind the concept.
- Soldering. I have become very familiar with soldering by soldering the connections of the test bench. I have learned how to effectively tin connections, use flux, and change the temperature of the iron to get an ideal soldered connection.
- Microcontrollers. Through learning more about the Raspberry Pi Pico and programming it in Thonny using Micro Python, I have learned about microcontrollers as well as their capabilities and methods to using them.
- Electronics. I learned much more about various circuit parts and their properties while exploring parts that could be used in our circuitry to power our heaters and fans.
- Ability to learn new things. Being exposed to so many different areas through this project, I expanded and tested my ability to learn new things.
- Teamwork and collaboration. I have become much more acquainted with how to work on a project in a group through my meetings and work sessions with ---

Conclusion

The temperature regulated test bench for the 28-Pin CerDIP carriers helps the lab test integrated circuits in a temperature-controlled manner that keeps resistance constant. The physical model of the test bench is finished. Hardware that powers the heater and that senses the temperature within the heater enclosure is implemented. Functionalities relating to temperature control: PID tuning, the rotary encoder, the amplification transistor, must be researched and further implemented. Further work necessary is included in the
Results and Discussion. I recommend that a resoldering of unaesthetic /unideal soldered connections be made on top of the further work necessary.

References

[1] Vishay, "NTCLE100E3," Vishay, 2024.

[2] J. K. Beata Grabowska, "The Thermal Conductivity of 3D Printed Plastic Insulation Materials—The Effect of Optimizing the Regular Structure of Closures," MDPI, Wroclaw, 2020.

[3] T. Polygenis, "A Comprehensive Guide to PLA Melting Point and How it Influences 3D Printing," Wevolver, 2023.