Developing a control system for Peltier cells to be used in haptic applications

Panagiotis Oikonomou
Engineering Department, New York University
Abu Dhabi, UAE
po524@nyu.edu

A control system for Peltier Units was designed and implemented. The system is designed to be used for research applications in thermovinrotactile funneling illusion. The system is incorporating a peltier cell and a vibration motor to provide thermal and vibrotactile feedback. The temperature is actively measured with a thermocouple attached directly to the top of the Peltier unit. A platform independent control application was created and a C++ Arduino compatible library to control said unit was devised.

I. Introduction

In haptic applications thermoelectric coolers such as Peltier cells are increasingly utilized. A Peltier cells has the ability to relatively quickly change the direction of heat flow across its two ends, by utilizing a PN Junction design (*Fig. 1*) [1]. When current flows through in a particular direction, as the electron (e^-) and hole (e^+) flow are opposite, they will travel from one face of the cell to the other, directing the heat flow accordingly [1]. This current generates a heat flow from one side to the other, that gives rise to a temperature difference across the cell [2]. This temperature difference is proportional to the driving current (I) due to the combined effects of Joule Heating where the power, $P = RI^2$, and the heat flow ($\frac{dQ}{dt}$) to from the hot to the cold side at a rate $\frac{dQ}{dt} = K\Delta T$, where *K* is the thermal conductance of the cell [1].

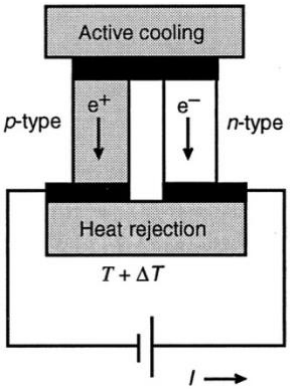


Fig. 1. Diagram of peltier cell internal structure. Two plates used for thermal ejection are thermally connected using a P-Type and an N-Type semiconductor. The semiconductors are connected in one layer and not connected in the other. Applying a potential difference across the semiconductors, will cause the flow of the charge carries (holes in the case of th N-Type, electrons in the case of the P-Type) to move towards the same direction. The heat is transferred along with the charge carriers and therefore, as one side becomes hotter, the other cools down. Adapted From [1].

This "rapid" heating and cooling ability of Peltier cells has been widely explored and utilized in the field of haptics in recent years. This utilization varies from applications in virtual reality, where they are implemented in gloves to stimulate accordingly to the thermal feedback of the scene objects [3-5], to thermal brain stimulation in comma, and potential dementia patients to aid in their computer interaction [6], to creating tactile displays using a combination of vibration and thermal feedback taking advantage of the Thermal Grill Illusion observed when a hot and cold body are in close proximity in contact with the skin [3, 7].

In this paper, the development and testing of a system responsible for controlling two types of Peltier cells is described. The first type (T1) is a miniaturized stacked 4 x 4 x 3.8 mm NL2022T unit, while the second type (T2) is a traditional 15 x 15 x 3.5 mm CP70137 unit. The system was developed with foresight to be used in haptic applications such as the above, but also in psychophysical experiments cantered around the effect of thermal and vibrotactile stimuli in haptic interactions.

II. System Overview

Two similar systems were created for the different types of Peltier cells. The general system structured, however, can be identified as follows (*Fig. 2*). The peltier cell's outputs are connected to a driver circuit that is powered by an individual power supply at the rated voltage. A thermocouple is attached to the cell's face in contact with the skin, using thermal adhesive. The outputs of the thermocouple are connected to a logic level, commercially available amplifier circuit with an embedded temperature sensor for standardization. A heatsink is used in the other face of the Peltier cell, in order to absorb the excess heat when cooling and allow it to reach low temperatures. Using a different motor driver, powered by a 4.5V external power supply, a vibration motor is attached to the Peltier cell assembly to generate vibrotactile feedback. All the outputs are then connected to a microcontroller that is further connected via USB to a computer running a platform independent controller application. The microcontroller can be powered either by an external 5-9V power supply or by the computer's own USB port. Furthermore, the system is scalable, allowing for multiple units being connected to the controller, limited by number of pins of the microcontroller used. Finally, a C++ library is developed, compatible with the Arduino IDE that allows the control of the peltier cells and thermocouple.

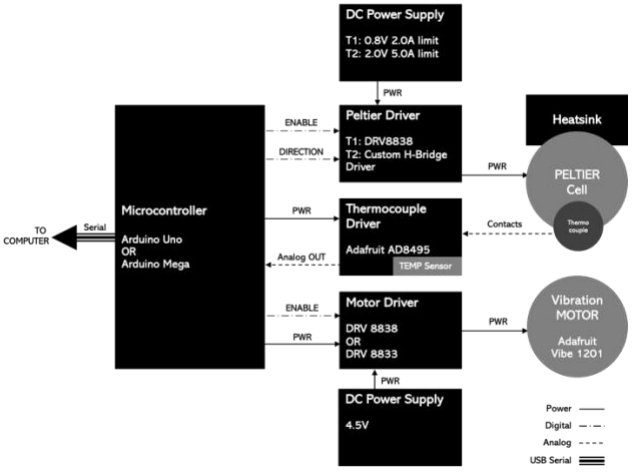


Fig. 2. A limited system diagram indicating all the systems in place to control a peltier Unit.

III. Component Characterisation

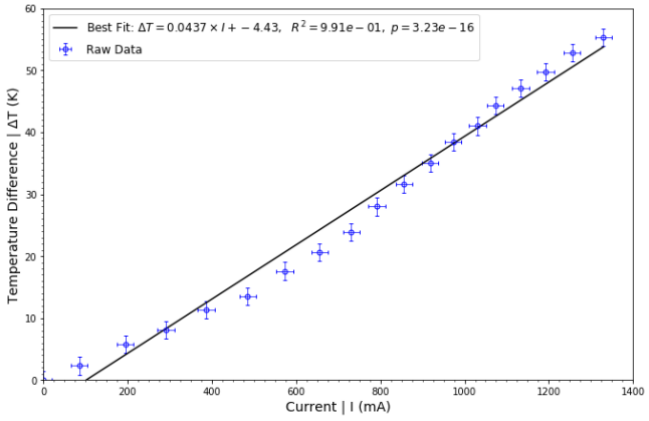
A. *Peltier Cell Type 1 (T1)*

Type 1 peltier Cell was characterized using a standardized thermocouple (see section III.C). The Cell was found to have a very high transient response, however, the area that it was occupying was very small. As a result, when the T1 was attached to the skin, the time required to heat up to a particular temperature grossly increased because the area of the T1 in contact with the skin was too small to effectively transfer thermal energy to or from the contact point.

Furthermore, it was identified that the thermocouple was to large in relation to the unit, therefore when someone was in contact with the cell, they would mostly feel the thermocouple's pressure than the subtle temperature difference. Attempts to solder the thermocouple in the side of the peltier cell, failed as the heat required to solder the thermocouple was to high and started disordering the p-type and n-type semiconductors from their top plate conducting points that damaged the peltier cell.

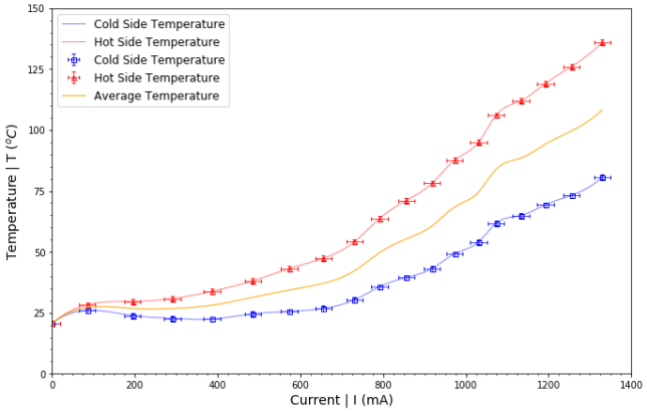
B. *Peltier Cell Type 2 (T2)*

Peltier Unit T2, was characterized using a FLIR-T62101 thermal camera and a power supply. The thermal camera recorded a temperature average of each side of the peltier cells while the current and voltage were obtained from the power supply. The voltage applied ranged per specifications from 0.0V to 1.8V (absolute max Voltage 2.0V). As seen in Graph 1 a linear pattern between the current and the temperature difference was observed to best fit the operation range of the peltier Cell.



Graph 1. Characterization of Peltier Cell using thermal camera. Temperature difference (ΔT) VS Current (I) were plotted to discover a linear trend, with best fit equation: $\Delta T = 0.0437 \times I - 4.43$. The high correlation value strongly indicated a true trend.

During characterization, however, another pattern was observed. Even though Temperature difference was strictly increasing as planned, the average temperature of the peltier was increasing as well (Graph 2). This increase was due to the fact that as heat was taken away from the cold side to the warm, but the thermal gradient established between them increased the heat flow to the cold side. As a result, positive thermal feedback was evident between the cold side heating up due to heat flow, and the Peltier's Joule Heating of the other side. This increase in temperature meant that in order to achieve a temperature difference of 50 ºC the average temperature of the peltier should be more than 80 ºC (Graph 2), which is not practical for cooling applications.



Graph 2. This grpah shows how the avearge temperature of the peltier cell, increases with increasing temperature difference. The blue and red data points are the actual values detected with a thermal camera at specific current intervals measured with the embedded power supply ammeter. The lines are the b-spline interpolation of the raw data, calculated using the scipy python library. The orange line, is an estimation of the avearge temperature, taking into account the top and bottom temperatures of the cell.

As a result, the need to have a constant temperature in one plate became apparent, in order to proceed to cooling applications as well. Per suggestion of literature [2, 8-11], a heatsink was attached to one of the faces of the Peltier Cell. In theory, this would prevent temperature buildup in one side that was enough to cause the other side to heat up, essentially achieving having one side at constant temperature.

Nonetheless, it is worth mentioning that this technique, even though very practical, after some time of continuous thermal oscillation, the heatsink has more energy than it can dissipate in time and therefore the side attached to it starts to heat up, rendering it useless. To combat this issue, one way would be to increase the heatsink size. However, since this is not always an option in haptic applications, another way would be to change the material of the heatsink to one with a lower specific heat capacity. An ideal material would have a thermal transfer rate higher than the thermal transfer rate of the Peltier Cell. Therefore, it will be able to absorb the heat and cool down faster than the thermal oscillation of the cell. Such materials, in order of effectiveness, could be (please note that they can have other practical considerations):

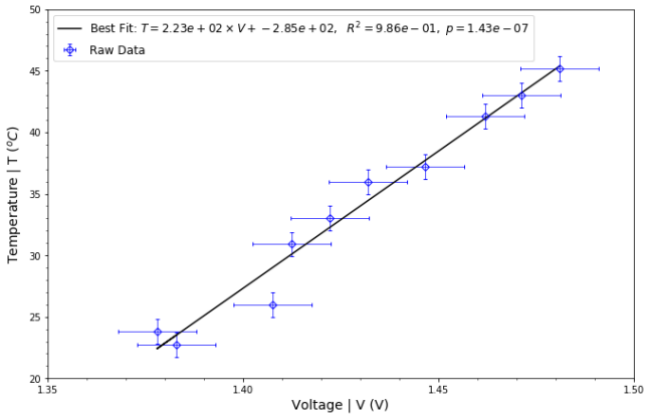
TABLE I. Materials To Replace Aluminum Heatsink [12]

Material Name	***Specific Heat Capacity c (J/Kg K)***
Lead	129
Gold	129
Tungsten	132
Tin	228
Copper	385
Aluminum	436

C. *Thermocouple*

The thermocouple was calibrated using a T2 heated up while recording it's temperature with a FLIR-T62101 thermal camera. The T2 had one side stuck to an aluminum heatsink using thermal adhesive, while the other was attached to the thermocouple that was in mechanical contact surrounded by conductive thermal paste. As current was flowing through the peltier cell, the temperature of its one side was recorded with

the thermal camera and the voltage of the thermocouple was recorded using the analog input of an Arduino Mega. The signal was converted using the internal ADC to a 1024 scalar with limits 0V (GND) to ca. 5V. However, due to the load created by driving the mosfets in the H-Bridge driver of the T2, the voltage of the Arduino would periodically fluctuate with the activation of the digital pins driving the T2. As a result, the logic Analog Reference Voltage was fed back to the Arduino's AREF pin from the thermocouple amplifier (Fig. 8*)* to standardize the analog input HIGH reference to the voltage obtained by the Arduino. The characterization is shown as a function of voltage in (Graph 3).



Graph 3. The thermocouple characterisation function. As temperature increased, the thermocouple voltage output was measured using a multimeter, and the temperature of the thermocouple using a thermal camera. The trend appears to be linear (High correlation of $R^2 = 0.986$). The fit equation was used to calculate the temperature of the peltiers at their operating range.

Thus, through the characterization of the thermocouple the Temperature T as a function of the voltage V is shown in (*Eq #*).

$$T = 2.23 \times 10^4 \times V - 2.85 \times 10^2 \qquad (1)$$

Where V is the potential difference detected by the Arduino in Volts, and T is the Temperature in ºC of the Thermocouple.

IV. Thermal Studies

To estimate how the enclosure should be constructed, thermal simulations were calculated. These simulations were conducted with Fusion 360's thermal simulation engine with the model directly imported. This section contains the main points and interpretation of the reports. The full documents can be found in Appendix 1 and 2.

A. Peltier Type 1 (T1) thermal simulation

The construction of the model for T1 is seen in section V.A. Two scenarios were simulated for T1, with the extreme temperatures in either side of the peltier cell. As seen in the simulation output Fig. 3.B, the thermocouple even though in contact with the base plate of the peltier effectively had the same temperature. This is very beneficial as the accurate temperature will be detected properly. The timestamp, however, when the thermocouple reached the base plate temperature was 1.0 s, meaning that the temperature reading would have at least 1 second latency to detect the temperature, therefore, a design alteration is needed to place the thermocouple in direct contact with the thermocouple's base plate. In the opposite simulation scenario (Fig. 3.A) the same situation was observed. It is also worth mentioning that in said simulation set it is assumed that the T2's cold face temperature remains constant, unaffected by the hot thermal load, in order to speed up calculation. Therefore, with a more processing power, a more accurate thermal model can be created.

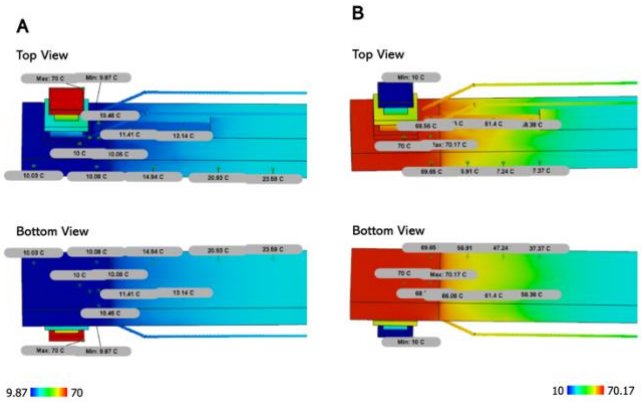


Fig. 3. Thermal results analysis of Peltier Type 1 (T1) Cell holder. Parts of the model have been omited from view to simplify the illustation of the simulation (i.e. Top cover, wires, heatsink). The thermal simulation is evaluated at equilibrium. Fig A is the scenrio with 10ºC at the bottom plate and 70ºC at the top, while Fig B is the opposite scenario. Both Top and bottom views are provided.

B. Peltier Type 2 (T2) thermal simulation

For T2, two similar simulations were carried out on the 3D model. After incorporating the closest materials, conductance values of bonds, and appropriate convection setup in standard room conditions, the peltier was loaded with 38 ºC in one side and 23º in the other, with ambient temperature 23 ºC. The loads were later inverted for the second simulation.

From the results of case B (top face at 38 ºC) (Fig. 4.B) it is possible that the excess heat created by the top plate, at equilibrium does not affect the temperature of the bottom plate, due to the size of the heatsink compared to T1. The design of the dome around the thermocouple in both cases A & B, was found to improve its response time to 0.3 s due to the reflected heat and the fact that the thermocouple is in direct contact with the peltier cell. It is important to point out, that in the equilibrium thermal distribution of case A (Fig. 4.A), the heatsink does not appear to be able to cool down solely by thermal convection with the surrounding air. This suggests that the peltier device, in order to be able to achieve a temperature variation, should not be left on, as it would not be able to cool down afterwards to a lower temperature.

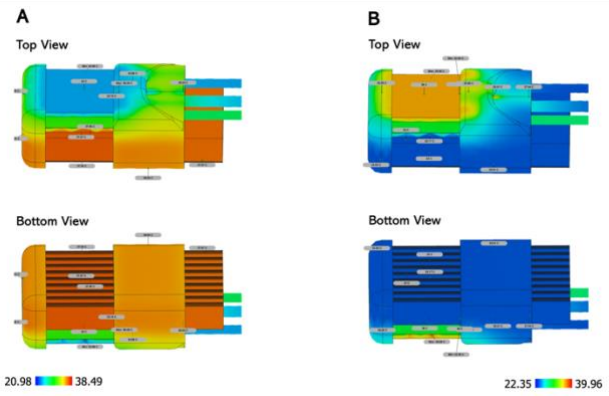


Fig. 4. Thermal results analysis of Peltier Type 2 (T2) Cell holder. Parts of the model have been omited from view to simplify the illustation of the simulation (i.e. connecting wires, wire insulation). The thermal simulation is evaluated at equilibrium. Fig A is the scenrio with 40ºC at the bottom plate

and 23ºC at the top, while Fig B is the opposite scenario. Both Top and bottom views are provided.

V. 3D Modeling

Information about all version of the 3D models, as well as Fusion 360 CAD files and STL files can be found in documentation.

A. Peltier Cell Type 1 (T1) Attachment cover

To more appropriately control the peltier cells, holding all the components together, the following enclosures have been designed in cad. The *"Peltier Base MK2"* for T1 (Fig. 5) was the model used in the thermal simulations. It contains a heatsink on the top side to dissipate the Peltier's heat, and a small metal plate at the other side designed to be in contact with the skin. Attached with thermal adhesive to the metal plate is the thermocouple. The outer shell of the part is designed to be 3D printed in PLA or ABS (Thermally tested as ABS) and is separated into 3 components that can be printed without support material (Print quality tested in UM2). The part can be then glued together with the T1 the thermocouple and the sink and adds structural rigidity to the peltier as well as provides thermal conductance and convection with the environment to cool down as needed. Technical drawing can be found in Appendix 3.

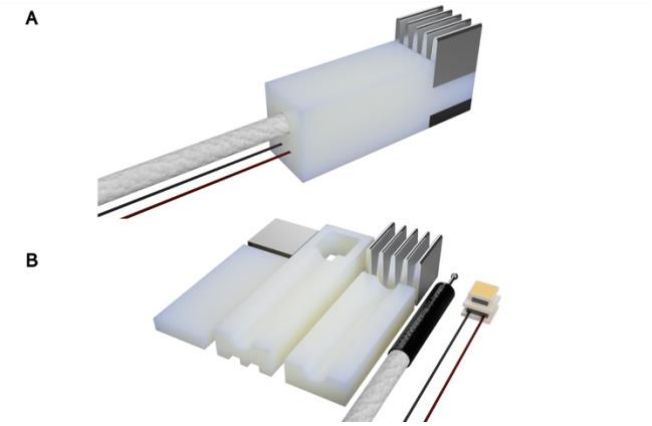


Fig. 5. Rendering of Peltier Cell Type 1 (T1) cover. The materials viewed are the correct materials of the scene. The original CAD files can be found in the documentation while the engineering drawings in Appendix 3. Fig A is an isometric view of the model, while Fig B is an explosion of all of its components.

B. Peltier Cell Type 1 (T1) Funnelling Illusion Attachment

For T1 a system of cascading multiple peltier cells linearly together and controlling them (see circuit id section VIII.A) using a single microcontroller to investigate thermal funneling was created. This system consists of two components, a peltier cell housing with dovetailed ends (Fig. 6) that slots directly into a 2-rail ruled base in order to be arranged properly. In this design, the thermocouple is directly attached to the top face of the peltier device for better thermal response time.

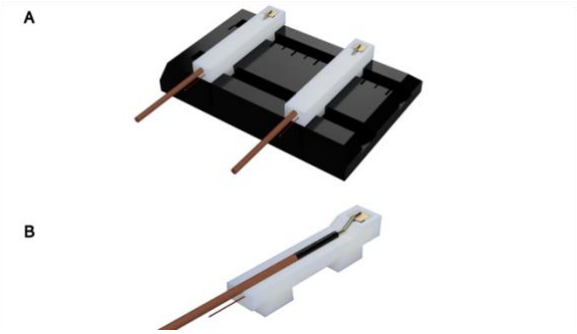


Fig. 6. Rendering of Peltier Cell Type 1 (T1) cover and support assembly. The materials viewed are the correct materials of the scene. The original CAD files can be found in the documentation. Fig A is an isometric view of the model, while Fig B is a detailed isometric view of one peltier cell holder used (Top cover has been ommited).

C. Peltier Cell Type 2 (T2) Attachment Cover

For the T2, a different type of enclosure was designed. It incorporates two 3D printable parts in ABS or PLA (Thermally tested as ABS) with supports. The front part (Fig. 7) is used as an end stop to hold the T2 in place along with the heat sink. The second part slides over the heatsink and T2, and it contains a pocket to house the thermocouple in contact with the T2's front face so as to get an accurate temperature reading. The bezels on top of the peltier are less than 0.8 mm in height to that when the peltier unit is touched to the skin it has maximum contact with it, furthermore, the cover of the thermocouple provides an additional barrier preventing the tip of the sensor to touch the hand and confuse the temperature measurement, as, that way, a fluctuation by the skin temperature will be induced. The second part also contains a recessed designed to fit an *Adafruit Vibration Motor #1201* that is used to stimulate vibrotactile feedback with the skin.

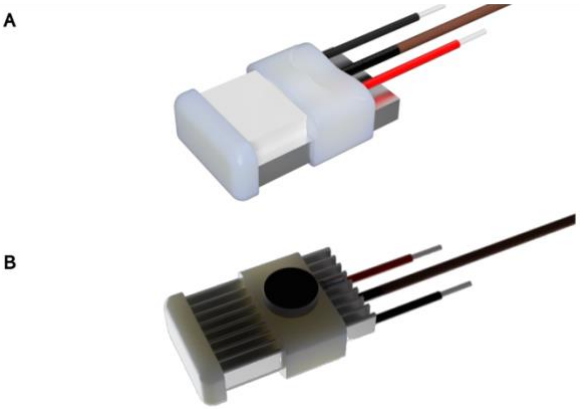
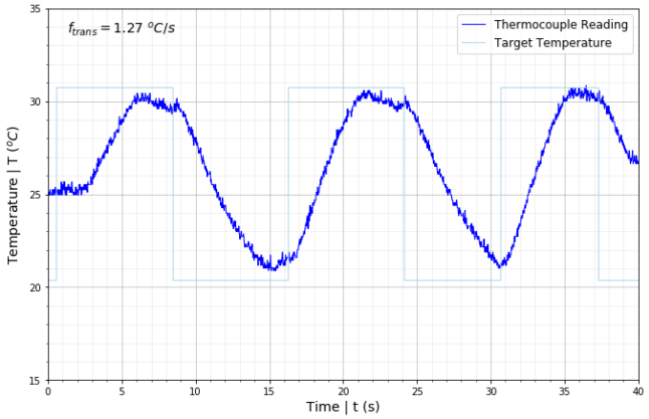


Fig. 7. Rendering of Peltier Cell Type 2 (T2) cover. The materials viewed are the correct materials of the scene. The original CAD files can be found in the documentation while the engineering drawings in Appendix 4. Fig A is an isometric view of the model, while Fig B is an inverse isometric vew of the model, to illustrate how the mottor is attached. The top face of the cell (Fig. A) is in contact with the skin.

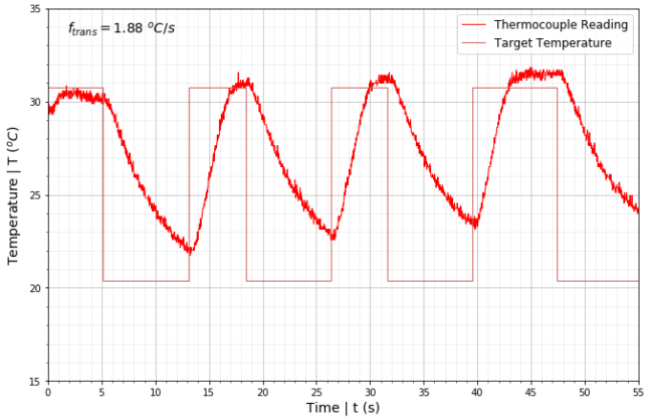
VI. T2 Final Transient Response

The transient response of the Peltier cell T2, was measured by connecting it to the system and commanding the temperature to follow a square wave oscillation pattern with different duty cycles up to the point when the T2 barely reached the target temperature at the end of the cycle. During this process it was observed that the transient response of T2 towards cold temperatures is different than the one towards hot. As a result, the square wave given varied according to the

category tested: i.e. to investigate the cold transient frequency f_{tCold} (Graph 4), the peltier was heater for a prolonged period of time to ensure that it reached the maximum temperature and then cooled for decreasing time intervals over a temperature difference of 10 ºC (30 ºC to 20 ºC). The opposite occurred to calculate the hot transient frequency f_{tHot} (Graph 5).



Graph 4. Transient response of Type 2 Peltier Cell aiming for cold temperature. The wave is a sample. The response rate was calculated by using the pulse width (pw) at the lower temperature. Specifically, $f_{trans} = \frac{20^oC}{pw} = 1.27^oC/s$.



Graph 5. Transient response of Type 2 Peltier Cell aiming for hot temperature. The wave is a sample. The response rate was calculated by using the pulse width (pw) at the lower temperature. Specifically, $f_{trans} = \frac{20^oC}{pw} = 1.88^oC/s$.

Specifically, the transient response frequencies are as follows:

$$f_{tCold} = 1.27\ ^oC/s$$

$$f_{tHot} = 1.88\ ^oC/s$$

As a result, a faster heat up rate was observed. Furthermore, it is worth noting, that as the T2 attached heatsink increases in temperature the f_{tCold} decreases as well, up to the point that it is no longer able to reach temperatures below a particular threshold. More close examination is required, however, to characterize such behavior.

VII. Control Code

To control the Peltier cell assembly a library was developed in C++ compatible with the Arduino API. This library contains 4 main classes each responsible for controlling one element of the assembly (i.e. Peltier Cell, Thermocouple, Motor) while the fourth class is used to connect the Arduino with the assembly to the control application developed in processing (Seen in Section VIII.A). A complete set of the methods and implementations can be found in the documentation, this section, briefly outlines the basic parts of the library's operation

A. *Thermocouple*

The Thermocouple class is used to control a thermocouple connected to an AD-8495 thermocouple amplifier. The output of the amplifier is connected through a voltage divider to an analog input pin of the Arduino board. The temperature is calculated using the method *void readTemp()* that takes the analog signal converts it to the equivalent voltage using *Eq #* and then uses the thermocouple characteristic equation (*Eq #*) to convert the voltage into temperature.

$$V = A_{in} \times \frac{A_{ref}}{ADC_{res}^2 - 1} \tag{2}$$

Where V is the thermocouple voltage, $A_{in} \in [0,1023] \cap N$ is the analog input from the corresponding Arduino Pin, A_{ref} is the Voltage reference of the analog input of the Arduino (Logic HIGH ~ 5.0V), and ADC_{res} is the Arduino's Analog to Digital Converter Resolution, in this case it is fixed to 10 as it is a 10-bit ADC.

Furthermore, the thermocouple contains a rolling average function. The user can define the maximum number of data points (N) to be included in the average in the constructor (default is 5), and a dynamically allocated array is created to store the last specified number of data points. When an the method *readTemp()* is run, the new value measured is pushed in the array, as the first element, while the last gets discarded. The average value changes iteratively according to *Eq #*.

$$\bar{T} = \frac{T_0 - T_{N-1}}{N} \tag{3}$$

Another method *void predict(int t)* is used to predict the temperature of the thermocouple according to its gradient in time *int t* measured in milliseconds. The gradient is calculated using *Eq #*.

$$\frac{\Delta T}{\Delta t} = \frac{T_0 - T_{N-1}}{\delta t} \tag{4}$$

Where $\delta t = N * t_{res}, t_{res} = 10 \frac{ms}{element}$, and T_0 is the current temperature.

As a result, the predicted temperature in time t milliseconds is calculated by *Eq #*.

$$T(t) = \frac{\Delta T}{\Delta t} \times t + T_0 \tag{5}$$

B. *Peltier*

The Peltier class is used to control both types of Peltier cells (i.e. T1 and T2). It incorporates a Boolean variable *H-Bridge* that when *true* it is able to drive a peltier connected to the custom H-Bridge driver (Section VIII, part #) specifically 2 *Direction* Pin Driver, and when *false* it is able to drive a peltier that is connected to a driver with *Enable* and *Direction* pins, like the DRV8838 used in the final system for controlling T1.

Each instance of the Peltier Class contains a reference to a Thermocouple object, which corresponds to the thermocouple physically attached to the Peltier cell. To heat up the cell, the method *void heat(float targetTemp)* has the following pseudocode (The class variable *THRESHOLD* is defined with the instantiation of the object):

```
START heat
    IMPORT targetTemp
    temp <- Thermocouple.read()

    BOOLEAN peltierState <- |targetTemp - temp| > THRESHOLD
    BOOLEAN reverseState <- targetTemp < temp

    IF H-Bridge IS TRUE
            SET peltierPin TO peltierState
            SET reversePin TO reverseState

    ELSE
            SET peltierPin TO peltierState*reverseState
            SET reversePin TO peltierState*(NOT reverseState)
END heat
```

This code allows the heating up of the Peltier unit for both types of drivers. As a result, the following Truth table (table #) can be constructed for Driver Carriers (*H-Bridge* = false) and H-Bridge drivers (*H-Bridge = true*), showing the direction of the motor according to the target temperature T_{target} and the current temperature T_0.

TABLE II. MOTOR DIRECTION TRUTH TABLE

Scenario	***Driver Carrier***		***H-Bridge Driver***		***OUT***
	Peltier Pin	***Reverse Pin***	***Peltier Pin***	***Reverse Pin***	
$T_{target} < T_0$ AND $\lvert T_{target} - T_0 \rvert > THR$	HIGH	LOW	LOW	HIGH	COOL
$T_{target} > T_0$ AND $\lvert T_{target} - T_0 \rvert > THR$	HIGH	HIGH	HIGH	LOW	HEAT
$T_{target} > T_0$ AND $\lvert T_{target} - T_0 \rvert < THR$	LOW	LOW	LOW	LOW	IDLE
$T_{target} < T_0$ AND $\lvert T_{target} - T_0 \rvert < THR$	LOW	HIGH	LOW	LOW	IDLE
$T_{target} = T_0$ AND $\lvert T_{target} - T_0 \rvert < THR$	LOW	LOW	LOW	LOW	IDLE

The Peltier Class also implements a method that would allow the cell to linearly ramp from a one temperature T_i to another T_f in time *t* milliseconds. This functionality is implemented in runtime without *delay()* so that it can be non-intrusive to the rest of the program running in the Arduino. A function *void rampSet(float Ti, float Tf, int t)* was implemented so that it can be called once and set up the parameters for a particular ramp. Then, in the body of the loop, the function *void ramp()* is called in order to heat the peltier to the appropriate value at the time it's called. The pseudocode for the *ramp()* function is seen below:

```
START ramp
    IF onRamp IS TRUE  //class variable onRamp is set true when a
                       //ramp is set, and false when a ramp is
                       //finished
    currentTemp <- Thermocouple.read()
    tempDifference <- Tf – Ti

    // MILLIS() is a function of the Arduino API that returns the time
    // from the moment the program started running in milliseconds
    // class variable initialTime is set to the MILLIS() when the
    // rampSet() function was called.
    targetTemp <- ((MILLIS() – initialTime) * tempDifference) / t

    heat(targetTemp)

    IF (targetTemp < toTemp) OR (tempDiff > 0)
            onRamp <- false
END ramp
```

C. Motor

The Motor Class is a very simple driver class for a motor connected directly to one PWM pin of the Arduino with an NPN Logic level transistor or a motor driver that is controlled by one pin (i.e. A driver carrier whose Direction or Phase pin is pulled to logic LOW or logic HIGH) connected to it.

The motor class contains a *void drive(int PWM)* method, that drives the writes the equivalent PWM signal to the motor pin.

D. Controller

The controller Class is designed to standardize the way of communication between the control application and the Arduino. The communication at both ends happens through the Serial port of the computer that the Arduino is connected to. This sketch is designed to read and interpret string commands of particular format send in the serial port by the control application (See control application in Section IX), and then send the target temperature of all the peltier cells, as well as their current temperature to be graphed.

The way that the interpretation of commands work is by decoding a very specific structure. The command structure and function is shown in table #.

TABLE III. COTROLLER COMMANDS

Command	***ID***	***Function***	***Callback***
H <peltier> T <temp> E	H	Heats a the peltier cell int: <peltier> to temperature float: <temp>	heat()
I <peltier> E	I	Set peltier int: <peltier> to idle mode	Idle()
R <peltier> F <Ti> T <Tf> I <time> E	R	Ramp peltier int: <peltier> from float: <Ti> to float: <Tf> in time int:<time> ms	ramp()
A <peltier1> <peltier2> H <temp1> C <temp2> T <time> E	A	Alternate Rampling between <peltier1> and <peltier2> in time int: <time> ms	autoRamp()
V <peltier> T <PWM>	V	Vibrate the motor of peltier int:<peltier> at int: <PWM>	vibrate()

As it can be seen in *table #* the controller commands all have a unique 1-letter identifier at the beginning used to callback the relevant function that later decodes the rest of the command String and performs the appropriate action. All the commands are terminated with the character E, so that the Arduino will only empty its *Serial Input Buffer* once it has detected a complete command. This way if more or less than one commands are flushed in the buffer during the time the Arduino is checking for it, it will act appropriately to either wait or discard the command. The commands are parsed with string methods, by identifying a control character (e.g. 'T', 'F', etc.), then reading the value from the first space (' ') after it to the second one. The substring is then parsed to the equivalent data type using native methods, and then it is used in the context of the appropriate command.

The controller class contains dynamically allocated arrays of 4 types:

1) Peltier: That contains Peltier instances corresponding to the peltiers connected to the arduino

2) Thermocouple: That contains Thermocouple instances corresponsing to the thermocuples connected to each peltier

3) Motor: That contains Motor instances corresponding to the motors at each peltier cell aassembly

4) PeltierControl: this is an array of a custom struct that contains the appropriate variables to control each peltier (the struct is seen in the code below.)

```
//Assistive data structure for the Controller Class
struct PeltierControl{
  bool heat;
  bool idle;
  bool autoRamp;
  bool wait;
  float targetTemp;
```

It is worth mentioning that the dynamic allocation is not implemented using the keywords *new* and *delete*, as they are not supported by the Arduino API. Instead, the functions malloc() and free() are used to assign and free memory space to pointers.

VIII. Circuits

A. Circuit for T1 Arrangment

To control T1 an Arduino Mega Microcontroller was preferred because of its high amount of Pulse Width Modulation (PWM) Enabled pins, that can be used for driving the motors. Each motor needs only one PWM pin to control it that is directly connected through a 1KΩ resistor to the base of a 2N2222 logic level NPN transistor. The collector is connected to one end of the motor, while the other end is connected to an external 4.8V power supply. The emitter is connected directly to the common ground of the circuit. A diode was added between the terminals of the motor to prevent back voltage (Fig. 8). The motor circuit can be repeated as many times as needed for the number of peltier units connected to the microcontroller.

The T1 was controlled with a low voltage, high current motor driver carrier, such as the dual channel DRV8833 or the single channel DRV8838. In the current schematic the DRV8833 is shown (Fig. 8). The ends of the T1 are connected to the output pins of the one channel of the driver and two digital pins were connected to the input channel as shown in Fig. 8. One of the pins is controlling the phase of the T1 (i.e. the direction of current flow), the other is controlling the power (i.e. Turning the motor on or off).

Finally, an AD8495 thermocouple amplifier was used to read the thermocouple output so that it can be read by the analog pins of the Arduino. It is worth mentioning that the reason why the motors are connected to external power supplies is because the Arduino by itself had a power limit that was reached when driving the Motors to logic High. This meant that the voltage of the microcontroller would momentarily drop to maintain the current flow. This drop would affect the voltage coming to the amplifier of the thermocouples introducing an extra instability in the output. As a result, whenever the motors were in use the thermocouple output would introduce large fluctuations (±5 ºC) in their readings. Therefore, initially, a variable capacitor was placed between the +V and GND terminals of the microcontroller to smooth out the fluctuations. However, even though the voltage was not fluctuating, it would still drop an average of 0.1 V when the motor was used. As a result, it was decided to use an external power supply for the motors and that fixed the fluctuation problem.

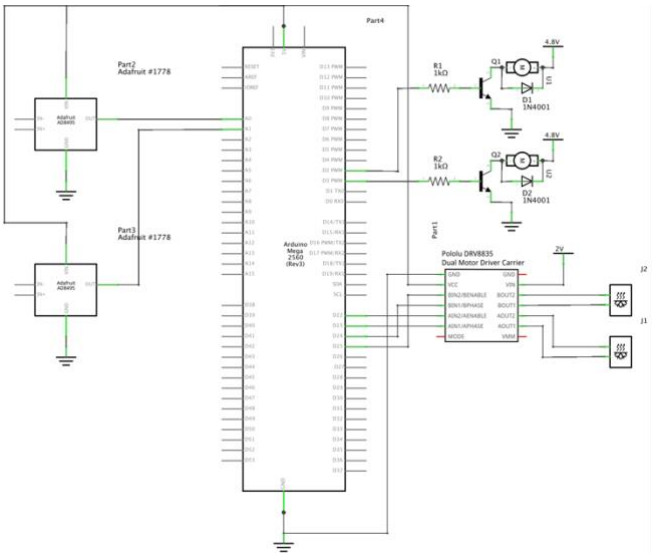


Fig. 8. Circuit schematic for Peltier Cell Type 1 (T1) control. All the elements are included except the thermocouples that are connected directly to the inputs of the two AD8495 amplifier units. The amplifier has an embedded temperature sensor for standardising so that the thermocouple does not need to be diped in ice cold water for reference. The cirucit shown is for two peltier Elements with motors and tehrmocouples.

B. Custom H-Bridge Driver for T2

Peltier Cell T2 can draw current up to 7.0 A at 2.0 V, this characteristic meant that it was really hard to find a commercial driver that could deliver such a high current at these low voltages. Therefore, a custom driver was designed using N-Type logic level Mosfets (IRLB8721). At this point is worth mentioning that, ideally, replacing two of the four IRLB8721 Mosfets with 2 PNP Mosfets with a threshold voltage at around 1.4 ~ 2.0 Volts ($V_{th} \in [1.4, 2.0]\ V$), as they can be driven with the voltage output of the peltier unit power supply (Examples of such Mosfets include: SPU09P06PL, SFT1341-E, or 2SJ681(Q)). Since there was a shortage of such specific components the circuit seen in Fig. 9 was designed and built. The H-Bridge driver works at logic level with two pins from the Arduino, that drive the cell as seen in Truth *table #*. A PCP was created for the circuit a, as seen in Fig. 9.

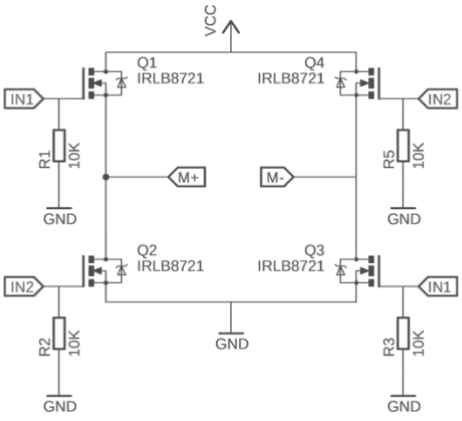


Fig. 9. Custom H-Bridge Driver cirucit. The 4 Mosfters are logic level and can be directly controlled through the arduino Digital Pins. Each pin output is attached with a pulldown resistor that connects the gate to the ground.

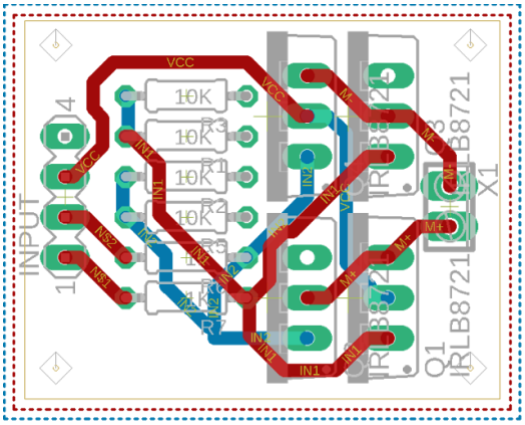


Fig. 10. The PCB CAD output of the circuitboard of the circuit shown in Fig. 9. The board has a GND poor in both sides so that to minimise the connections needed. The poor is not shown in the diagram.

C. *Circuit for T2 Arrangement*

For the circuit of T2 the thermocouples and motors are connected in a similar fashion in seen in Fig. 9. The peltier cells, however, are driven using the custom H-Bridge driver seen in *Section # Part #*, where the 2.0 V power Supply is connected to the VCC pin, the ground of the power supply is connected to the ground of the Arduino and the GND pin of the driver. Then two digital pins from the are connected directly from the Arduino to the drivers IN1 and IN2 pins. The T2's terminals are connected to the M+ and M- of the driver.

IX. Control Application

To effectively control the peltier Cells a control application was developed in Java, so that it is platform independent. The application, "Grapher MK3," includes a graph element at the top that directly reads the output from the Arduino through the Serial port. These values are the reading from the thermocouple of all the connected Peltier Cells, and the target temperature of all the cells. At the left side there are numbered circular elements that change transparency according to the corresponding vibration motor.

Below there is a series of control elements that set the temperature of each peltier accordingly. Below that there are two graphs that can be altered so that to send a real-time temperature and/or vibration change to the Arduino. The pulses sent, are saved as .txt files using a native file dialog and can be loaded back for easy and fast access during psychophysical experiments. The application also possesses a limited command line interface for the commands shown in Appendix 5, that are useful in controlling various features of the application. The application is fully customizable by setting up the port using the *PORT* command, and by setting up the number of cells connected to the Arduino using the *PELTIERNUM* command.

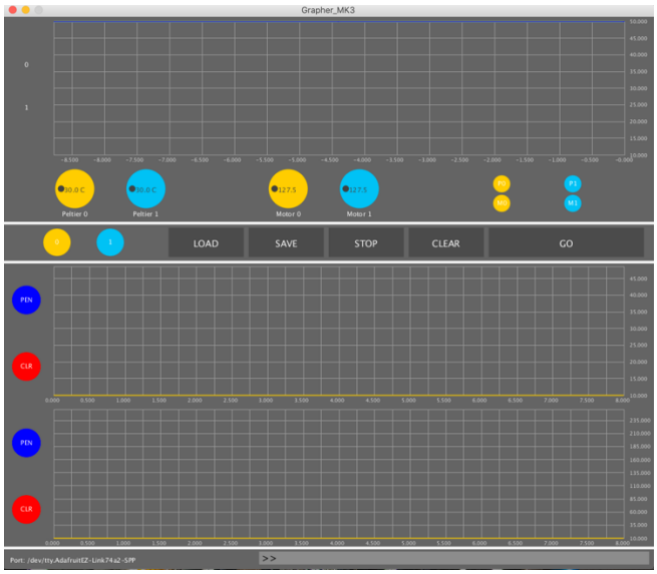


Fig. 11. A screencast of the Control Application Graphical User Interface (GUI). At the top is the part where the output from the arduino is projected, using the graph and the numbered lights. The elements at the top (i.e. sliders, buttons, etc.) are used to directly control the attributes of the connected Peltier Cells or Vibration Motors. The bottom section is used to create graphs designed to be send to the arduino directly. The lines snap to the grid to ease with the design process.

X. Implications for Further Research

To further develop this project, it is suggested that a driver for the large Peltier Cells is created using the PNP Mosfets previously described with two optocouplers controlling them to isolate the circuit. Furthermore, a substitution of the heatsink size and/or material would be appropriate to prevent the self-destructive behavior of the peltier Unit. Furthermore, the application should be further optimized after undergoing some usage testing during true experimental procedures, to optimize the workflow, and minimize the setup time. Also, an alternative for the thermocouple as a means of measuring temperature should be explored to minimize the top ledge footprint of the unit. It is suggested that either the thermocouple is heated and pushed thought the plastic insulation on the underside of the top face of the Peltier Cell.

XI. Reference

[1] F. J. DiSalvo, "Thermoelectric Cooling and Power Generation," *Science*, vol. 285, no. 5428, p. 703, 1999, doi: 10.1126/science.285.5428.703.

[2] C. Alaoui and Z. M. Salameh, "Solid state heater cooler: Design and evaluation," 2001: Institute of Electrical and Electronics Engineers Inc., pp. 139-145, doi: 10.1109/LESCPE.2001.941640.

[3] G. Chernyshov, K. Ragozin, C. Caremel, and K. Kunze, "Hand motion prediction for just-in-time thermo-haptic feedback," S. N. Spencer, Ed., 2018: Association for Computing Machinery, doi: 10.1145/3281505.3281573.

[4] G. Gioioso, M. Pozzi, M. Aurilio, B. Peccerillo, G. Spagnoletti, and D. Prattichizzo. *Using Wearable Haptics for Thermal Discrimination in Virtual Reality Scenarios*, vol. 535, pp. 144-148, 2019.

[5] H. Morimitsu and S. Katsura, "Two-degree-of-freedom robust temperature control of peltier device based on heat disturbance observer," (in English),

Electr Eng Jpn, Article vol. 184, no. 1, pp. 66-74, 2013, doi: 10.1002/eej.22282.

[6] J. Gabriel, A. Silva, M. T. Restivo, and I. Pinheiro, "Brain stimulation using an haptic thermal device," 2013, pp. 32-35, doi: 10.1109/ExpAt.2013.6703025.

[7] T. H. Yang, G. H. Yang, D. S. Kwon, and S. C. Kang, "Implementing compact tactile display for fingertips with multiple vibrotactile actuator and thermoelectric module," 2007, pp. 580-581, doi: 10.1109/WHC.2007.72.

[8] H. Morimitsu and S. Katsura, "Frequency response analysis of observer-based thermal control system of peltier device," 2011, pp. 256-261, doi: 10.1109/HSI.2011.5937375.

[9] Y. Osawa, H. Morimitsu, and S. Katsura, "Control of thermal conductance with detection of single contacting part for rendering thermal sensation," (in English), *IEEJ J. Ind. Appl.*, Article vol. 5, no. 2, pp. 101-107, 2016, doi: 10.1541/ieejjia.5.101.

[10] A. Savin, A. Floca, M. Trifanescu, N. Ionescu, and A. Visan, "Designing a cold plate used in rapid freeze prototyping technology," *Advanced Materials Research*, vol. 1036, pp. 648-51, / 2014, doi: 10.4028/www.scientific.net/AMR.1036.648.

[11] R. A. Taylor and G. L. Solbrekken, "Comprehensive system-level optimization of thermoelectric devices for electronic cooling applications," *IEEE Transactions on Components and Packaging Technologies*, vol. 31, no. 1, pp. 23-31, 2008, doi: 10.1109/TCAPT.2007.906333.

[12] Upper Canada District School Board. "Specific Heat Capacity Table." ucdsb.on.ca. http://www2.ucdsb.on.ca/tiss/stretton/database/Specific_Heat_Capacity_Table.html (accessed JUL 30, 2019).