Microfluidic Chip Heater Design Notebook

Author: Swagat Bhattacharyya

I. Introduction

The Purdue iGEM microfluidic (MF) chip has a reaction chamber that must be held steady at certain temperatures to properly react reagents. Conventional temperature control mechanisms, such as hot plates and ovens are bulky and expensive. Since a compact, low-cost solution is desired to make Purdue iGEM's MF platform more accessible, we have developed a minimalistic heating mechanism that can quickly heat up and regulate the temperature of a MF reaction chamber to within 1°C of any desired temperature between 30°C and 90°C.

II. Heat Transfer

Heat transfer characteristics are important for determining the required output flux of the heating mechanism and determine the local temperature gradient. Initially, we assume that the MF chip will be placed atop a circular, metallic heating element with radius R as shown in Fig. 1a. We assume that the circular region of the MF chip in direct contact with the heating element has a temperature of T_{BC} and that the surface temperature falls to room temperature (T_{RT}) at some radial distance ϵ from the center of the heating element. Then, the steady-state surface temperature function, $\phi(x,y)$, with $R < r = \sqrt{x^2+y^2} < \epsilon$ will follow Laplace's equation: $\nabla^2\phi(x,y) = 0$ assuming no heat loss. The boundary value problem can be solved over the annulus in the complex plane [1] to yield: $\frac{(T_{BC}-T_{RT})\log r + \log \frac{R^{(T_{RT})}}{\epsilon^{(T_{BC})}}}{\log \frac{R}{\epsilon}}$. To summarize, the surface temperature distribution will be given by:

$$\phi(x,y) \approx \begin{cases} T_{BC} & ;\ when\ r < R \\ \frac{(T_{BC}-T_{RT})\log r + \log \frac{R^{(T_{RT})}}{\epsilon^{(T_{BC})}}}{\log \frac{R}{\epsilon}} & ;\ else \end{cases} \tag{1}$$

An example of a surface temperature distribution with $T_{BC} = 50$°C and $R = 0.5$ is shown in Fig. 1b. We observe that the surface temperature rapidly drops near room temperature within 2 cm of the reaction chamber for sensible values of ϵ.

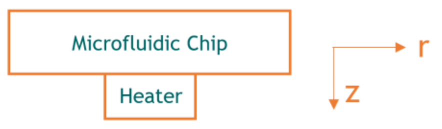
An important criterion for the design of the overall system is the heater temperature (T_H) required to maintain T_{BC}. This can be estimated by back-calculation from the finite-flux Neumann condition using a finite difference method. Since the heater has finite output heat flux, the following Neumann condition holds:

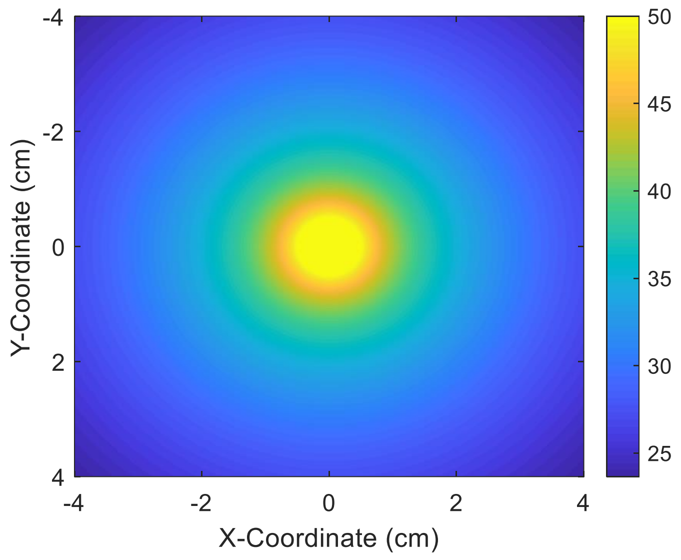
$$k_{PDMS}\frac{\partial\phi}{\partial z} = \frac{q}{\pi R^2} \tag{2}$$

In Eq. 2, k_{PDMS} denotes the thermal conductivity of the PDMS (about $0.0015\frac{W}{cm\cdot K}$) used to construct the MF chip and q denotes the thermal power dissipation (TDP) of the heater. The TDP of the heater in turn is given by: $q = V^2/R_H$, where V denotes the voltage drop across the heating element and R_H denotes the heating element resistance. If we substitute the finite difference approximation: $\frac{\partial\phi}{\partial z} \approx \frac{T_H - T_{BC}}{\Delta z}$ into Eq. 2, we obtain:

$$T_H \approx T_{BC} + \frac{\frac{V^2}{R_H}\Delta z}{\pi R^2 k_{PDMS}} \tag{3}$$

If we return to the previous example (shown in Fig. 1b), we estimate that $\frac{\partial\phi}{\partial z} \approx 6063\ ℃/cm$. If we take $\Delta z = 0.1\ mm$, we estimate that the heater element should be at 110℃ to heat the reaction chamber to 50℃. This represents a large (vertical) thermal gradient which makes it very difficult to heat the reaction chamber through the PDMS walls. The large thermal gradient is caused by the low thermal conductivity of PDMS, which is three to four orders of magnitude lower than that of most metals. Since the MF chip is intended to be disposable, the simplest and cheapest workaround for efficient heat transfer and temperature control is to embed a thin metallic insert (eg. Staple, push pin, etc.) through the bottom layer into the reaction chamber, such that the heating element can transfer heat through the insert into the reaction chamber. Special care should be taken to ensure that this insert does not cause Purdue iGEM's MF chip to leak; however, this is unlikely at the low pressure differentials present.





(a) (b)

Fig. 1: (a) Diagram of heater placement and sign conventions (b) Chip surface temperature (ºC) in proximity of the heating element [$T_{BC} = 50℃$ and $R = 0.5$]

III. Experimental Setup

An experimental test setup (shown in Fig. 2) using a steel wool-based heater was created in order to better understand different aspects of heater design and test algorithms. Although there are expected to be some differences between the prototype and the final product, it is expected that the results presented in this work will translate well to the final product. The mean TDP of the heater is controlled via an algorithm running on an Arduino microcontroller. The microcontroller reads the heater temperature via a negative temperature coefficient (NTC) thermistor voltage divider, performs control decisions, and then outputs a pulse-width modulation (PWM) signal to the gate of a n-channel Metal-Oxide Field-Effect Transistor (nFET) to regulate the heater (see Fig. 2). A large duty cycle on the PWM output corresponds to a high TDP, and a low duty cycle on the PWM output corresponds to a low TDP. In general, the mean TDP is: $q = Duty \cdot V^2/R_H$.

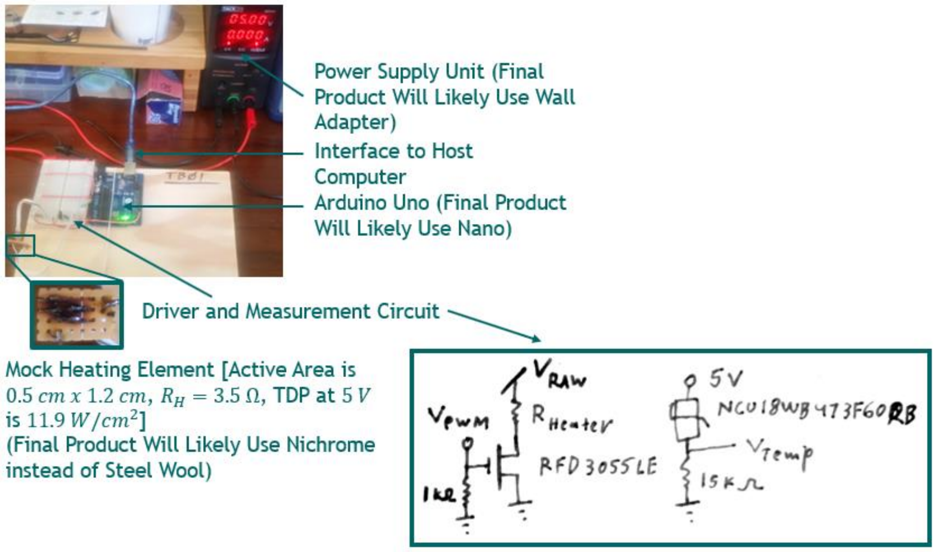

Power Supply Unit (Final Product Will Likely Use Wall Adapter)
Interface to Host Computer
Arduino Uno (Final Product Will Likely Use Nano)
Driver and Measurement Circuit
Mock Heating Element [Active Area is $0.5\,cm \times 1.2\,cm$, $R_H = 3.5\,\Omega$, TDP at $5\,V$ is $11.9\,W/cm^2$]
(Final Product Will Likely Use Nichrome instead of Steel Wool)
V_{RAW}
R_{Heater}
V_{PWM}
$1k\Omega$
RFD3055LE
5V
NCU18WB473F60RB
V_{Temp}
$15k\Omega$

Fig. 2: Diagram of heater system experimental test setup

IV. Heater Control Algorithm

Iteration 1:

The first iteration of the control algorithm had two distinct controllers: (1) bang-bang (2) proportional-integral (PI). The central idea behind the control mechanism was to use a bang-bang controller to output the maximum duty cycle (or turn off) to change the reaction chamber temperature from the current temperature to another setpoint (i.e. target temperature). Then, when the heater temperature is within a certain neighborhood of the setpoint, a PI controller is used to robustly offset heat loss from the reaction chamber (flowcharts summarizing the control algorithms in more detail can be seen in Fig. 3); the output of the PI controller is the estimated heat flux (q/A_{Eff}) required, where A_{Eff} denotes the effective total surface area of the heater and reaction chamber. If we assume that convective heat loss dominates in our application, the estimated heat flux required to hold the reaction chamber temperature constant at the setpoint $(T_{SetPoint})$ would be approximately: $h(T_{SetPoint} - T_{RT})$, where h denotes the convective heat loss constant, which is about $1.5\,\frac{mW}{cm^2 \cdot ^\circ\mathrm{C}}$ for stagnant air. This value is used to initialize the PI controller output (i.e. $u_0 = 1.5 \times 10^{-3}\,\frac{W}{cm^2 \cdot ^\circ\mathrm{C}} \cdot (T_{SetPoint} - T_{Atm})$. The PI controller was formulated in its iterative velocity form:

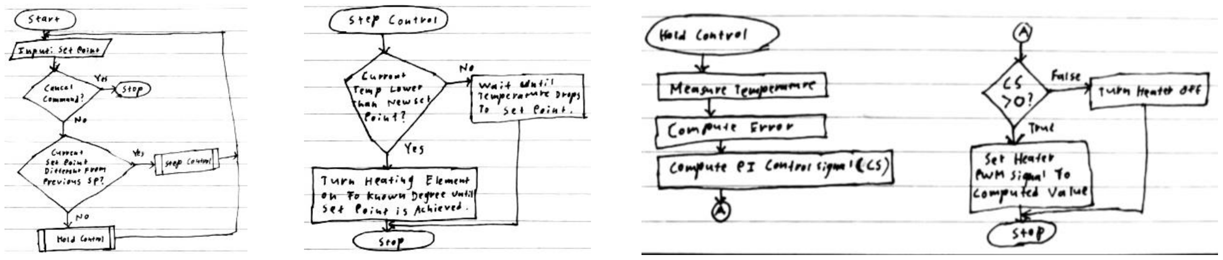
$$u_k = u_{k-1} - \Delta u_k; \quad \Delta u_k = k_p \left(e_k - e_{k-1} + \frac{T_k - T_{k-1}}{\tau_i} \cdot e_{k-1} \right) \tag{4}$$

In Eq. 4, k_p, τ_i, T_k, and e_k denote the proportional constant, the integral time constant, the time at step k, and the error from the set point at step k. The error at time step k is given by: $e_k = T_{Meas_k} - T_{SetPoint}$, where T_{Meas_k} denotes the measured reaction chamber temperature at time step k. The PI controller output is then converted to a duty cycle using:

$$Duty_k = \frac{R_H \cdot u_k}{V^2 \cdot A_{Eff}} \tag{5}$$

Note that the maximum and minimum duty cycle are limited in the actual implementation. Experimental measurements of the system response (heating element temperature) to a step change in the set point is shown in Fig. 4. There are three key observations to be made in Fig. 4. First, there is a significant amount of sensor noise, which can negatively affect stability. Second, the temperature dips significantly when the controller switches over from bang-bang to PI control; this is because the steady-state heat loss estimates were off. Third, there is a large overshoot above the setpoint that seems to be caused by integral windup (i.e. the integrator accumulating the error terms despite the heater being turned on to its upper or lower limit).

A Python-Arduino interface was designed for data logging, data analysis, and high-level control of the MF chip. This interface allows for two levels of abstraction, which makes things easier for users and future developers. The Python script running on a host computer (i.e. the clinician's computer) passes hardware parameters to the Arduino at the appropriate time and can analyze/visualize data sent from the Arduino. The Arduino performs hardware control at the lowest level with control algorithms. The communication between Python and Arduino is structured. The Python code iterates through the available serial ports on the host computer, sending a specific packet to find the port where the Arduino is connected. Once the Python code finds the proper serial port and establishes proper communications, a user-specified script is run. This user-specified script typically includes an assortment of serial communications tasks. In a standard exchange, Python will first send three bytes to the Arduino, where the first byte will correspond to an instruction (eg. turn on the heater, move a motor, etc.), and the second and third bytes will be auxiliary numerical information to execute the instruction (eg. set point, motor speed, target motor position, etc.). The, the Arduino will send tab delimited information (eg. temperature, fluorescence, etc.) back to the host computer.


Start
Input: Set Point
Cancel Command?
Yes
Stop
No
Current Set Point Different From Previous SP?
Yes
Step Control
No
Hold Control
Step Control
Current Temp Lower than New set Point?
No
Wait Until Temperature Drops To Set Point.
Yes
Turn Heating Element on To Known Degree Until Set Point is Achieved.
Stop
Hold Control
Measure Temperature
Compute Error
Compute PI Control Signal (CS)
A
A
CS >0?
False
Turn Heater Off
True
Set Heater PWM Signal To Computed Value
Stop

(a) (b) (c)

Fig. 3: (a) Flowchart of overall control algorithm (b) Bang-bang controller for switching between set points (c) PI controller for holding a set point

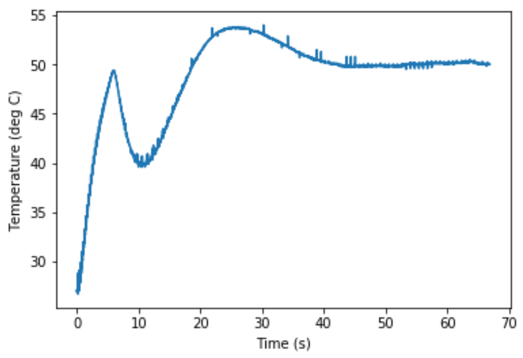


Fig. 4: Experimental measurements of controller (Iteration 1) with setpoint of 50°C

Iteration 2:

The final iteration of the heater control algorithm sought to mitigate the issues observed from the previous iteration. To ensure smoothness in the system response, only one controller was used (no switching action). A proportional-integral-derivative (PID) controller with anti-windup was used. Due to the presence of a derivative term, special care was taken to reduce sensor noise. Sensor noise reduction was performed using an exponential moving average (EMA) filter, which is defined by the following iterative equation:

$$T_{Meas,Filt_k} = \alpha T_{Meas_k} + \left(1 - T_{Meas,Filt_k}\right);\ \ T_{Meas,Filt_0} = T_{Meas_0} \tag{6}$$

In Eq. 6, $0 < \alpha < 1$ denotes the memory factor of the EMA filter, and $T_{Meas,Filt_k}$ denotes the filtered temperature measurement. The PID controller here is defined by:

$$u_k = u_{k-1} - \Delta u_k;\quad u_0 = 1.5 \times 10^{-3}\,\frac{W}{cm^2\cdot°C}\cdot(T_{SetPoint} - T_{RT}) \tag{7}$$

Where the increment is given by:

$$\Delta u_k = k_p\left(e_k - e_{k-1} + \frac{T_k - T_{k-1}}{\tau_i}\cdot e_{k-1}\cdot\epsilon + \frac{\tau_d}{T_k - T_{k-1}}\cdot(e_k - 2e_{k-1} + e_{k-2})\right) \tag{8}$$

$$e_k = T_{Meas,Filt_k} - T_{SetPoint} \tag{9}$$

In Eq. 8, T_d denotes the derivative time constant and ϵ denotes the anti-windup switch. The anti-windup switch is defined by:

$$\epsilon = \begin{cases} 0;\ u_{k-1} < 0\ \ or\ \ u_{k-1} > Duty_{Max}\cdot\frac{V^2}{R_H}\cdot A_{Eff} \\ 1;\qquad\qquad\qquad\qquad\qquad\qquad else \end{cases} \tag{10}$$

In Eq. 10, $Duty_{Max}$ denotes the maximum permissible duty cycle (for safety reasons). The controller defined by Eq. 6-10 works well as demonstrated by experimental measurements of the system response to a step change in the set point shown in Fig. 5. There is low measurement noise, and the system responds quickly but does not overshoot.

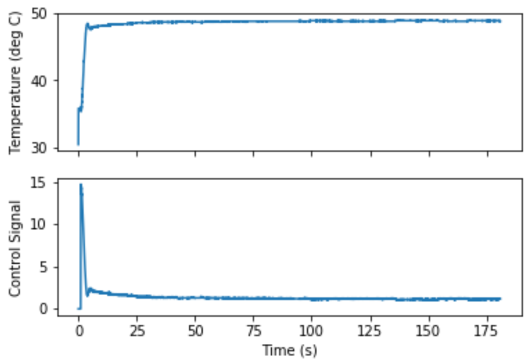


Fig. 5: Experimental measurements of controller (Iteration 2) with setpoint of 50°C

V. Results and Discussion

The PID control algorithm written was tested for setpoints between 30-90°C as shown in Fig. 6. As seen in Fig. 6, the controller approached the setpoint fast without significant overshoot for all setpoints tested. Furthermore, there was no asymptotic error and low noise. When the controller activates, the spike in the P+D components of the controller output may cause the duty cycle to be capped to $Duty_{Max}$, pushing the heater toward the setpoint rapidly. When this happens, the integrator is turned off until the duty cycle returns within the bounds. Ultimately, the integral term is responsible for holding the heater at the setpoint as $t \rightarrow \infty$. However, the temperature that the sensor measures may not be the true temperature – there may be $1 - 2$°C of error in the sensor measurement itself.

The cooldown time is trickier to experimentally characterize but can be motivated theoretically. It can be shown that (if we assume convective heat loss is the dominant heat transfer mechanism) the reaction chamber temperature during cooling can be approximated by:

$$\frac{\partial T}{\partial t} \approx \frac{h(T_{RT}-T)A_{Eff}}{m_w C_v + m_H C_p} \tag{11}$$

In Eq. 11, m_w denotes the mass of the fluid inside the reaction chamber, c_v denotes the specific heat at constant volume for the fluid (about 4 $\frac{J}{g \cdot K}$ assuming the fluid is mostly water), m_H denotes the mass of the heating element (0.4 $\frac{J}{g \cdot K}$ for steel), and C_p denotes the specific heat at constant pressure of the heating element (). Solving this ODE for $T_{RT} < T$ yields: $T = T_{RT} + (T_0 - T_{RT})e^{-\frac{hA_{Eff}}{m_w C_v + m_H C_p}t}$, where T_0 is the initial heater temperature. Hence, the time required to cool to a temperature T_{Cool} would be given by: $t_{Cool} = \frac{m_w C_v + m_H C_p}{hA_{Eff}} \ln\left(\frac{T_0 - T_{RT}}{T_{Cool} - T_{RT}}\right)$. From this formula, we find that for most MF chip-heater configurations, we can say $A_{Eff} \sim R_{Chamb}^2$ (i.e. the effective surface area is dominated by the surface area of the thin cylindrical reaction chamber) and notice that $m_w C_v \gg m_H C_p$ and $m_w \sim R_{Chamb}^2$. Thus, we find that the cooling time is largely independent of the reaction chamber radius (R_{Chamb}). An example cooling time computation with some semi-arbitrary parameters is shown in Exp. 1.

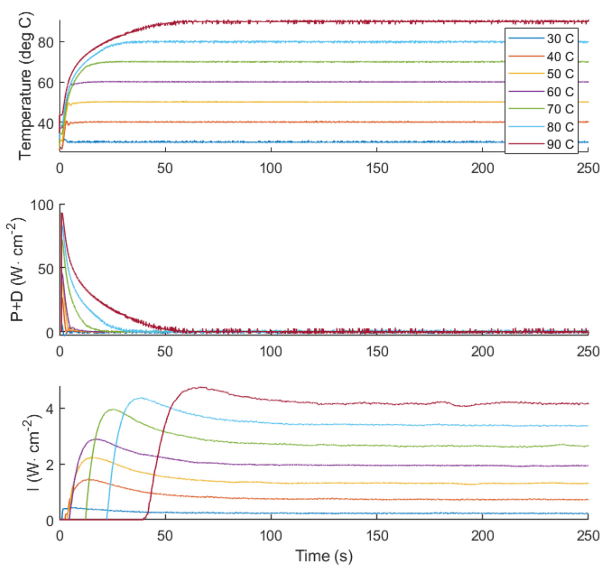


Fig. 6: Experimental measurements of PID controller with setpoints between 30-90°C

Exp. 1: Demonstration of cooling time estimation

We want to find the time it will take for 70 mm^3 of water ($m_w = 0.07\ g$) to cool from $T_0 = 80$°C to $T_{Cool} = 37$°C if $T_{RT} = 25$°C, $h = 1.5 \times 10^{-3} \frac{W}{cm^2 \cdot {}^\circ C}$, $C_v = 4 \frac{J}{g \cdot K}$, $A_{Eff} = 1\ cm^2$, $C_p = 0.4 \frac{J}{g \cdot K}$, and $m_H = 0.1\ g$.

$$t_{Cool} = \frac{m_w C_v + m_H C_p}{h A_{Eff}} \ln\left(\frac{T_0 - T_{RT}}{T_{Cool} - T_{RT}}\right) \approx \mathbf{325\ s\ (5.4\ mins)}$$

Note that the cooling time would have been about 41 s if there was no water in the reaction chamber. These numbers are generally an overestimate because there are other cooling mechanisms besides convection.

References

[1] Fundamentals of Complex Analysis by E.B. Saff & A.D. Snider