ENG ME419 Final Project

COOLING A RASPBERRY PI

Eric Sun & Nathaniel Feldman
Boston University
Prof. Goldfarb

December 16, 2016

Abstract

This report seeks to explain the heat transfer that occurs from computer components by heating and cooling a Raspberry Pi. Although not accurately represented by theoretical calculations, the results can be modeled effectively using COMSOL to demonstrate the advantages of these cooling solutions.

Contents

I Introduction

In the realm of modern computing, temperature management is a large concern. Hardware temperatures that are too high can impede performance and even cause failure. In practice, heat sinks and fans add extra cooling to a system over simply exposing components to stagnant air. Combining these cooling methods yield four different heat transfer situations to investigate and learn about.

II Theory

II.1 Natural Convection

II.1.1 Bare Chip

For natural convection, the most important non-dimensional term is the Rayleigh number:

$$Ra_L = \frac{g\beta(T_s - T_\infty)L_c^3 Pr}{\nu^2} \tag{1}$$

where g is gravitational acceleration, $\beta = \frac{1}{T_\infty}$ in Kelvin,Pr is the Prandtl number,ν is kinematic viscosity, $L_c = \frac{a}{4}$, and a is the side length

Calculating the Nusselt number will help find the convection coefficient.
For $10^4 < Ra_L < 10^7$:

$$Nu = \frac{hL_c}{k} = .39(Ra_L)^{1/4} \tag{2}$$

The rate of heat transfer is given by:

$$\dot{Q}_{conv} = hA_s(T_s - T_\infty) \tag{3}$$

II.1.2 Heat sink

For a heat sink, the Rayleigh number is calculated slightly different.
Assuming fin surface temperature is constant:

$$Ra_s = \frac{g\beta(T_s - T_\infty)S^3 Pr}{\nu^2} \tag{4}$$

where S is the spacing between fins. The Nusselt number can then be found using the following correlation:

$$Nu = \frac{hS}{k} = \left[\frac{576}{(Ra_S\frac{S}{L})^2} + \frac{2.873}{(Ra_S\frac{S}{L})^{0.5}}\right]^{-0.5} \tag{5}$$

The rate of heat transfer is given by:

$$\dot{Q}_{conv} = h(2nLH)(T_s - T_\infty) \tag{6}$$

where L is fin width, H is fin height, and n is the number of fins

II.2 Forced Convection

II.2.1 Bare Chip

Blowing air to create forced convection over the chip requires a different governing non-dimensional term, the Reynolds number:

$$Re = \frac{vL}{\nu} \tag{7}$$

Since convection varies over the length of a flat plate, an average Nusselt number, and thus an average convection coefficient can be found:

$$\overline{Nu_L} = \frac{\bar{h}L}{k} = .68(Re)^{1/2}(Pr)^{1/3} \tag{8}$$

The rate of heat transfer from the plate is given by:

$$\dot{Q}_{conv} = \bar{h}A_s(T_s - T_\infty) \tag{9}$$

II.2.2 Heat sink

Subjecting a heatsink to forced convection proves to be more complicated. As mentioned before, calculating the Nusselt number can help find the convection coefficient. In its generalized form the formula for said Nusselt number can be shown below:

$$Nu = C(Re_L)^m(Pr)^n \tag{10}$$

Where the constants C, m, and n depend on the flow situation and geometry. Because of the complex geometry, it was decided to use the simulation software Comsol to accurately predict the temperature distribution around heatsink. This was taken a few steps further and each flow situation was predicted by comsol for data comparison.

II.3 Issues and Computer Simulation

Attempting to apply the theoretical equations to the experiment yielded wide margins of error most likely resulting from critical assumptions and its small scale. The heat flux was assumed to be constant over the chip surface area with heat transfer rate equal to current draw at 5 volts. Solving for surface temperature using the assumed heat flux results in a negligible temperature difference clearly disproved by the data.

Comsol is a Finite Element Analysis software that can take different physics and engineering situations and couple them into a thorough analysis. In our particular situation, Comsol will be superimposing fluid flow situations with heat transfer situations in its CFD and Heat Transfer module.

Breaking the software down, it uses differential equations (such as the Navier Stokes Equation) as well as the various other momentum, energy, and flow equations to calculate the temperature distribution around a given model. To achieve the final results, one must set various different boundary conditions on the given model. After the boundary conditions are properly set, then the equations can be solved. A mesh is created around the model, and then interpolations and iterations are done to solve for the temperature distribution. This permits the simulations of the heat transfer cases discussed above.

III Materials and Methods

This experiment required a few specific materials to gather the necessary data. At the center is the Raspberry Pi 2 Model B, a microcomputer contained on a single board. A small heat sink and thermal contact paste were required enhance convection with fins. The paste was necessary to reduce the contact resistance between the heat sink base and chip surface. For precise calculations of the power requirements, power the raspberry pi through a controlled power sources, as to know the current and voltage. Finally, a desk fan created the air stream for forced convection. Make sure to look up the volumetric flow rate of the given fan as well as the diameter of the front face as to find the velocity.

Three trials each two minutes long were conducted for every individual case. Twenty seconds were added to the beginning of the forced convection trials to indicate a steady starting temperature. A python script generates a load on each core of the CPU causing the temperature to rise. After waiting long enough for a steady temperature to develop, another script records the chip's temperature each second from the Pi's built-in sensor to a CSV file. These scripts are both listed in the appendix under section A and B, respectively.

IV Results and Analysis

IV.1 Experimental Data

Figure 1 shows the experimental data collected for the natural convection situations. Notice how the average temperature of the bare chip is around $68°C$ while the average temperature of the heatsink is around $57°C$.

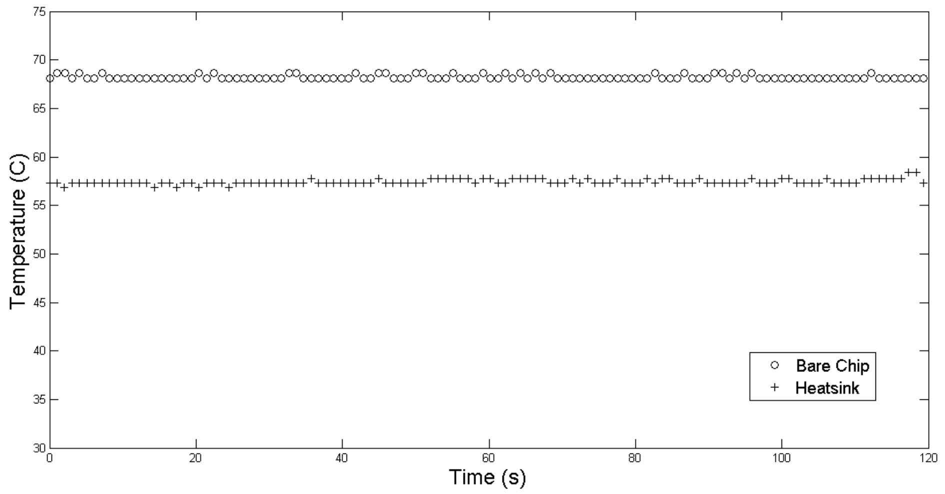

Figure 1: Natural Convection: Bare Chip vs. Heat Sink

Figure 2 shows the experimental data collected for forced convection of both bare chip and heatsink configurations. For the bare chip, the steady state temperature starts at $68°C$ and ends at $50°C$, nearly a $20°C$ difference. For the heatsink, the steady state temperature starts at $57°C$ and ends at $36°C$. Again, that is around a $20°C$ temperature difference. These two figures experimentally shows that forced convection more effectively cools the CPU than natural convection. In addition, these figures show that the heat sink more effectively transfers heat than the bare chip configuration.

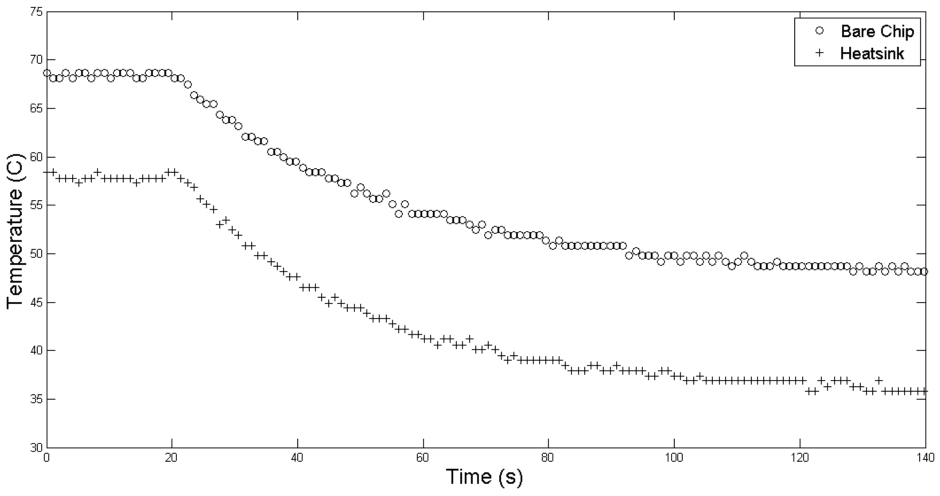

Figure 2: Forced Convection: Bare Chip vs. Heat Sink

These two figures experimentally shows that forced convection more effectively cools the CPU than natural convection. In addition, these figures show that the heat sink more effectively transfers heat than the bare chip configuration.

IV.2 COMSOL Model

The figure below shows the results of a Comsol heat transfer simulation of the bare chip under a natural convection. In this simulation, it shows that the surface of the chip is $80°C$ under steady state conditions.

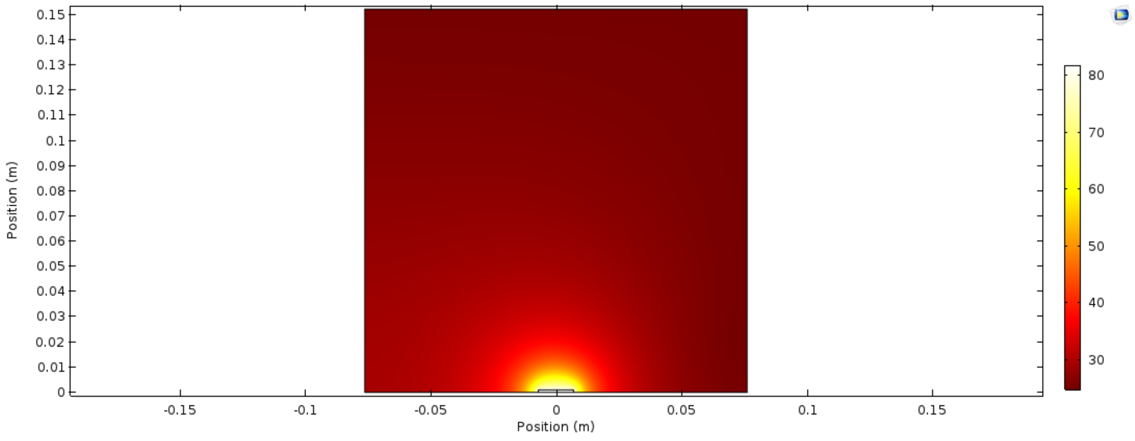

Figure 3: Natural Convection: Bare Chip

Figure 4 shows the natural convection of the heatsink in a 2D plane. Notice how the steady state temperature is slightly above 55°C, almost exactly matching the experimental data shown in Figure 1.

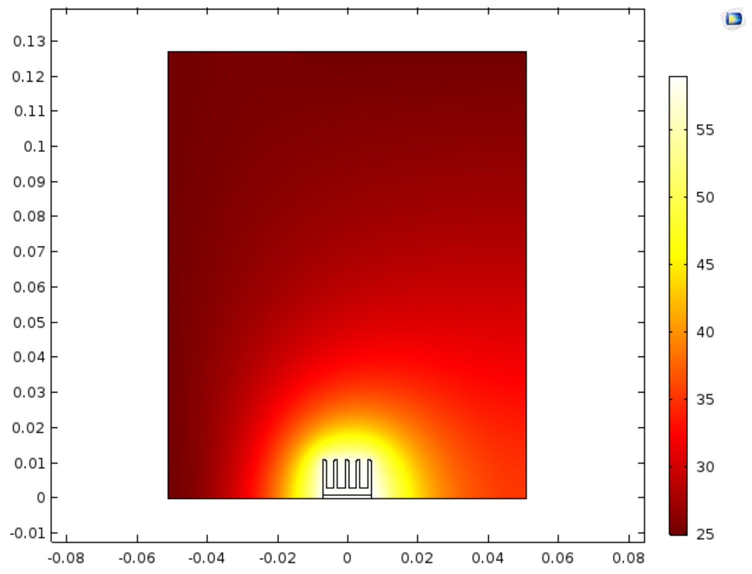

Figure 4: Natural Convection: Heat Sink

Figure 5 shows the forced convection of the bare chip. The steady state tempature of the chip is shown to be 65°C.

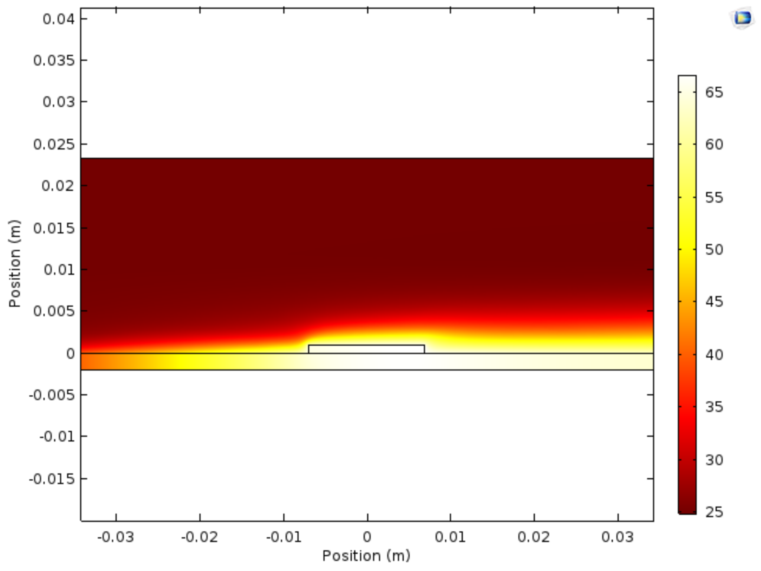

Figure 5: Forced Convection: Bare Chip

Although both Figure 5 and Figure 3 show a small amount of error with the chip temperature, the temperature difference between the natural and forced simulations is the around the same temperature difference shown in by the experiemental data for the bear chip.

Figure 6 shows the 3D temperature distribution of the heatsink under forced convection. The darker area indicates a cooler temperature due to the inlet of airflow. Overall, the whole plot ranges between $27°C$ and $31.5°C$, with the chip temperature likely being $31.5°C$. This is very close the the experimental data of the heatsink.

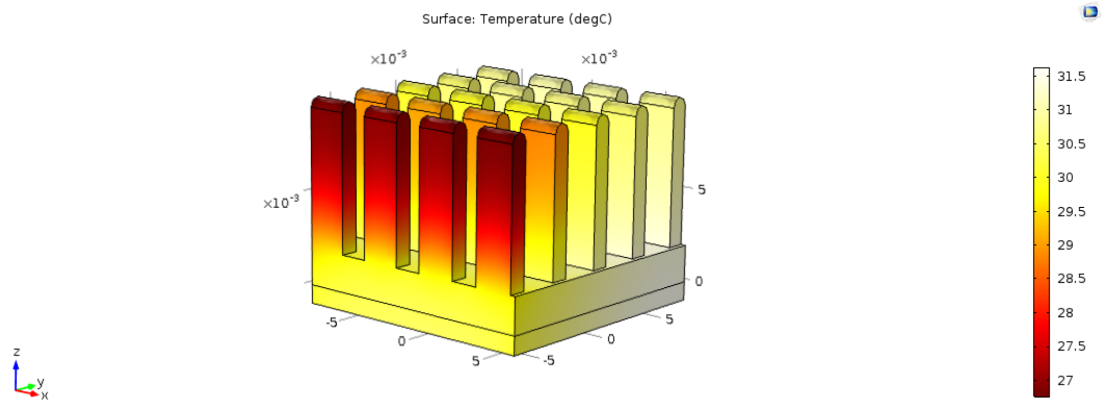

Figure 6: Forced Convection: Heat Sink

Below is a table of the exact temperature values for each situation. The calculated velocity from the fan and power to the circuit used for the Comsol simulations are $57 ft/min$ and $1.15W$, respectively

Table 1: Table of steady state temperatures for different configurations

	Experiment ($°C$)		**Comsol** ($°C$)	
	Natural Convection	Forced Convection	Natural Convection	Forced Convection
Bare Chip	68.2	48.2	82.3	66.1
Heatsink	57.4	35.8	57.4	31.5

V Conclusion

Heat sinks and fans are proven methods of reducing the temperature of heat sensitive components. Through this experiment, their ability to improve heat transfer was not only demonstrated, but also explained theoretically. However, further investigation of critical assumptions and appropriate correlations is necessary to build an more accurate theoretical model of the experiment. Thankfully, powerful computer simulations such as COMSOL can model real world conditions to produce similar results and corroborate the experimental data.

VI Bibliography

Ada, Lady. "Introduction to Raspberry Pi 2 - Model B." (n.d.): n. pag. Adafruit.com. Adafruit Industries, 09 Feb. 2015. Web. 13 Dec. 2016.

Ahmadi, Pakdaman, and Bahrami. "Pushing the Limits of Vertical Naturally-cooled Heatsinks; Calculations and Design Methodology." International Journal of Heat and Mass Transfer 87 (2015): 11-23. Print.

Cengel, Yunus A., and Afshin J. Ghajar. Heat and Mass Transfer: Fundamentals & Applications. 5th ed. New York: McGraw-Hill, 2011. Print.

Finlayson, Bruce A., and Ebrary Provider. Introduction to Chemical Engineering Computing. Second ed. 2014. Web.

Introduction to Multiphysics Modeling. Suecia: COMSOL, 2009. Comsol.com. Comsol Multiphysics, 2016. Web. 12 Dec. 2016.

VII Appendix

VII.1 Python Scripts

CPU Load Generator

```
import multiprocessing as mp

def f(x):
    while True:
        x*x

if __name__ == '__main__':
    pool = mp.Pool(4)
    pool.map(f,range(4))
```

Pi Temperature Monitor

```
from subprocess import check_output
from re import findall
import time
import csv

def get_temp():
    temp = check_output(["vcgencmd","measure_temp"]).decode("UTF-8")
    temp = float(findall("\d+\.\d+",temp)[0])
    return(temp)

run_time = input("Enter run time in seconds: ", )

with open("cpu_temp.csv", "a") as csvfile:
    start_time = time.time()
    while time.time() - start_time <= int(run_time):
       temp = get_temp()
       elasped_time = time.time() - start_time
       log = csv.writer(csvfile, delimiter=' ')
       log.writerow([str(elasped_time), str(temp)])
       time.sleep(1)
```