PCB Heat Testing After Washing

Contents

[Overview 1](#_Toc25772513)

[Experimental 1](#_Toc25772514)

[Results and Discussion 2](#_Toc25772515)

[Appendix A – Setup 4](#_Toc25772516)

[Appendix B – Python Script 5](#_Toc25772517)

# Overview

Testings have been conducted on the PCBs (Printed Circuit Boards), also referred to as SSUs (Smart Sensing Units) or simply 'boards', to gauge their performances after washing. The boards initially had large error margins, i.e. jumping to 6mV readings at 0.5mV input when heated to high temperature (80-90˚C). It was suggested to clean the boards with an alcohol (washing) as a remedy for this error. The testings described in this report confirmed that the washing indeed made drastic improvements of the accuracy of the results.

# Experimental

Each PCB reports two sets of readings: temperature and voltage drop. Each PCB, fully dried after washing, was connected to the Central Unit, sometimes known as TCP-CAN. The CU was then connected to a computer, which hosts a webUI (browser user interface) for displaying the data from the CU in real time. A 0.5mV voltage drop was placed on the PCB using electrical connector clips. The PCB was placed in a cardboard box, which acted as an easy oven.

After the setup, each PCB was allowed to operate at ambient temperature for about one minute. Then, hot air was blown into the oven from the top. Effort was made to swirl the hot air around the PCB, so that it received even heat. But it was not perfect. The temperature of the board was brought up to between 80˚C and 90˚C, which mimics the operating temperature of anode busbar where the PCBs would be installed. It was attempted to keep the temperature within this range for 2-3min, but again, fluctuations were inevitable.

Photos of the setup can be found in Appendix A. The data is collected on the computer and written into a .txt file. 67 PCBs were tested in this manner.

A close up of a keyboard

Description automatically generated

Figure Snippet of .txt file containing data for PCB No. 23. “sid” is the PCB ID number, "s\_smp" is the time stamp, "v\_in" is the voltage drop reading, "t\_in" is an unused parameter, "t\_brd" is the temperature.

Note in Figure 1, “sid” 172 is also present, because No.172 was also connected to the CU, but it was not being tested, i.e. it was outside of the oven.

# Results and Discussion

67 plots were obtained using a Python script found in Appendix B. The time span was usually 400 seconds, except for a few boards (“26rosak", "142", "172") which were monitored for up to 800 seconds, to investigate their behaviour over a long period of time. The y-axis of bottom graph was temperature, whereas the y-axis of top graph was voltage drop 0.3mv to 0.7mV. Note this was quite a small range. All plots are found in the .zip file submitted with this report.

Looking at the plot for board No.26, the voltage drop reading showed sudden jumps during the heating period. This behaviour was common among all boards tested. The behaviour following that, i.e. during stable temperature was more important. No. 26 displayed a more fluctuated reading at high temperature compared to room temperature, but the average was roughly unchanged. In other words, high temperature affected the accuracy of the reading, but to a much smaller extent than before washing.

A screenshot of a cell phone

Description automatically generated

Figure Plot of No. 26

Many PCBs showed good stability like No. 26. They are No. 4, 17, 23, 25, 26,29, 45, 48, 101, 131, 132, 229. However, some showed larger fluctuations, namely No. 21, 28, 88, 141, 168, 218. It has been inferred that these errors were due to bad contact between the electrical clips and PCB electrodes. Such errors were highly likely because of the angle in which the clips had to be placed in order to fit on the PCB (refer to Appendix A). Notably, even with the bad contact, the readings were still stable compared to those before washing.

A screenshot of a cell phone

Description automatically generated

Figure No.28 showed large fluctuations

No. 35, interestingly, zeroed out for a few seconds. This error was likely due to its firmware and would be fixed with updated stable firmware.

{"sid":35,"s\_smp":310.500000,"v\_in":0.000000,"t\_in":0.000000,"t\_brd":0.000000}

A screenshot of a cell phone

Description automatically generated

Figure No. 35 zeroed out for a few seconds around 320s

# Appendix A – Setup

A picture containing indoor

Description automatically generated

A close up of text on a whiteboard

Description automatically generated

A person sitting at a desk with a computer

Description automatically generated

# Appendix B – Python Script

func.py

import matplotlib.pyplot as plt

def process(f, sid, time, vd, temp):

#extracts all IDs, vd and temp

for line in f:

data = line.split(',')

#print(int(data[0].split(':')[1]))

sid.append(int(data[0].split(':')[1]))

#print(float(data[2].split(':')[1]))

vd.append(float(data[2].split(':')[1]))

#print(float(data[4].split(':')[1].split("}")[0]))

temp.append(float(data[4].split(':')[1].split("}")[0]))

#check if need to remove 172

for i in range(0, 4):

if sid[i] is not 172:

remove\_172(sid, vd, temp)

break

#print(vd)

construct\_time(sid, time)

def construct\_time(sid, time):

i = 0

for j in sid:

time.append(i)

i += 0.5

def remove\_172(sid, vd, temp):

i = 0

while (i < len(sid)):

#177 was measured with negative voltage drop

if (sid[i] is 177):

vd[i] = vd[i] \* -1

if ((sid[i] is 172) or (sid[i] is 0)):

sid.pop(i)

vd.pop(i)

temp.pop(i)

i -= 1

i += 1

def generate\_plot(file):

sid = []

temp = []

vd = []

time = []

with open(file, 'r') as f:

process(f, sid, time, vd, temp)

SSU = file.split('.')[0].split('t')[1]

plt.figure()

plt.suptitle('SSU No.' + SSU)

plt.subplot(211)

plt.plot(time, vd, 'r-', linewidth=0.5)

plt.ylabel('Voltage drop / mV')

#plt.axis([xmin, xmax, ymin, ymax])

if (len(time) > 1200):

plt.axis([0, 800, 0.3, 0.7])

else:

plt.axis([0, 400, 0.3, 0.7])

plt.subplot(212)

plt.plot(time, temp, 'b-', linewidth=0.5)

plt.ylabel('Temperature / C')

plt.xlabel('Time / s')

if (len(time) > 1200):

plt.axis([0, 800, 0, 100])

else:

plt.axis([0, 400, 0, 100])

#plt.show()

plt.savefig(SSU + '.png', bbox\_inches='tight')

#generate\_plot('heat55.txt')