Get going with projects from testing LYSO to cooling TECs

QA/QC Jig

FOR DUMMIES®

Demystifies everything from single photons to crosstalk

A Reference for the Rest of Us!®

QA/QC Jig Documentation

Version 0.1

Adi Bornheim John Dervan Anthony LaTorre Lautaro Narvaez
Guillermo Reales Kai Svenson

April 28, 2023

Contents

Chapter 1

Introduction

The BTL QA/QC jig is designed to test up to 12 LYSO arrays at a time for quality assurance during gluing procedure at each of the assembly centers. The arrays are attached to the jig via the flex cables which attach to the panasonic connectors. The theory of operation is that when the SiPM is hit by photons it generates a current pulse which travels along the boards to the amplifier boards at the front where a transimpedance amplifier converts this current pulse into a voltage signal. That voltage signal is then amplified by a factor of approximately 4 by a second stage op amp before leaving the boards through an SMA connector. Before the final output there are two gain paths: attenuated and through. The attenuated path goes through a 4x gain reduction path constructed from a pi attenuator circuit of completely passive resistors while the through path goes through unattenuated. The purpose of these two gain paths is to allow us to measure both single photons (through path) and $\sim$ 1 MeV signals (attenuated path) with the CAEN DT5742 digitizer which has a fixed dynamic range of 1 V peak to peak.

Chapter 2

Setup

The first thing each assembly center will need to procure is a computer capable of running the software to talk to the jig and for storing and analyzing data. The minimum specs required for each machine are the following:

1. Linux compatible (to run Alma Linux 8)[1]
2. 2 TB NVME disk (important!)
3. 64-bit
4. 8 GB RAM

The next step is to install Alma Linux 8. You can do so by downloading the iso file at one of the mirrors listed at `https://mirrors.almalinux.org/isos/x86_64/8.7.html` and copying it to a USB. For example if your USB is at /dev/sdb you can run:

```
$ curl -O -L http://dist-mirror.fibernetics.ca/almalinux/8.7/isos/
   ↪ x86_64/AlmaLinux-8-latest-x86_64-dvd.iso
$ dd AlmaLinux-8-latest-x86_64-dvd.iso /dev/sdb
```

The 2 TB NVME drive is extremely important as the speed at which data can be taken may otherwise be limited by the hard drive speed.

After installing Alma Linux 8, we need to update some packages and add the current user to the wheel group so they can run sudo commands. To do this run the following commands:

[1]This means generally not using any hardware which was released too recently as it may not have drivers

```
$ su # switch to root
# dnf update -y
# dnf group install "Development Tools"
# dnf config-manager --set-enabled powertools
# dnf install epel-release
# usermod -aG wheel [username]
```

Then you need to restart the computer for the changes to take effect.

Next, you can follow the instructions in the README at `https://github.com/alatorre-caltech/dt5742` for installing the data acquisition software or run the following commands:

```
$ cd # go back to home directory
$ git clone https://github.com/alatorre-caltech/dt5742
$ cd dt5742
$ sudo dnf install hdf5 hdf5-devel python3-numpy python3-scipy python3-
   ↪ psycopg2 root python3-root python3-h5py python3-matplotlib
$ sudo make install-deps
$ make
$ sudo make install
```

2.1 Talking to the Teensy

This section will discuss getting the jig set up so you can communicate with the Teensy microcontroller.

2.1.1 Flashing the Firmware

First you need to flash the firmware on to the Teensy (this may have already been done for you, if so you can skip this part). To do this you first need to install the Arduino and Teensy software.

To install the Arduino software, download

```
$ cd
$ curl -O -L https://downloads.arduino.cc/arduino-1.8.9-linux64.tar.xz
$ tar -xvf arduino-1.8.9-linux64.tar.xz
```

You can then follow the instructions for installing Teensyduino at `https://www.pjrc.com/teensy/td_download.html` or by running the following commands:

```
$ cd
$ wget https://www.pjrc.com/teensy/td_154/TeensyduinoInstall.linux64
$ wget https://www.pjrc.com/teensy/00-teensy.rules
$ sudo cp 00-teensy.rules /etc/udev/rules.d/
$ chmod 755 TeensyduinoInstall.linux64
$ ./TeensyduinoInstall.linux64 --dir=arduino-1.8.9
$ cd arduino-1.8.9/hardware/teensy/avr/cores/teensy4
$ make
```

Next, hook up the USB cable from the computer to the Teensy and run the arduino software:

```
$ cd arduino-1.8.9
$ ./arduino
```

You should now be able to upload the sketch. You can verify that everything is working by opening the "Serial Monitor" (see Figure 2.1) and you should see some text from the Teensy (there will be some things that look like errors, but that's just because by default the Teensy expects to see 8 boards, but you most likely have less than that, see Figure 2.2 for an example).

You can immediately communicate with the board by typing in the serial monitor. For example, try "help" which will output a list of commands you can send to the board.

2.1.2 Talking over Ethernet

All of the tools like the GUI use UDP packets to communicate with the Teensy. To get that working you first need to hook up the computer and the Teensy to a local router and set up the router to give the Teensy a static IP address based on its MAC address (if you're at a

Figure 2.1: Button to open the serial monitor is on the upper right corner of the Arduino GUI.

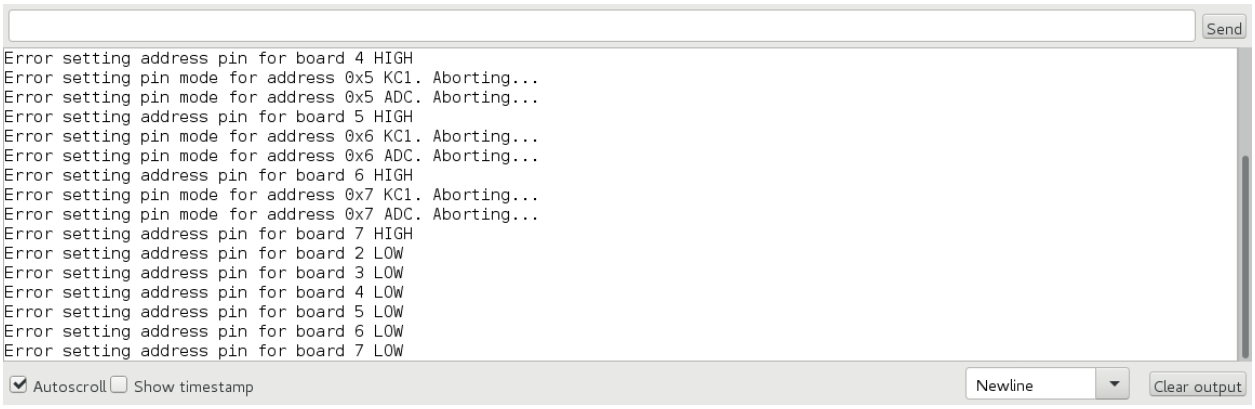


Figure 2.2: Serial monitor output on bootup

national lab, abandon all hope! You'll probably have to spend weeks or months to petition them to set up the router like so, so instead I'd recommend buying a cheap router at Best Buy or online and having a totally separate LAN).

To do this you should visit the local routers admin page (usually something like 192.168.0.1) and then going to "DHCP" and then setting it up so that any computer which connects with the MAC address AA:AA:AA:AA:AA:AA will automatically be given the IP address 192.168.1.177 (or something else, but that's the default in all the software). See Figure 2.3 for an example. Note that if you are using a different router it may look slightly different.

Once you have done this and rebooted the Teensy (make sure you don't see the first error message "Ethernet cable is not connected." in the Serial monitor), you can try to communicate with it by running:

```
$ qaqc-client --ip-address 192.168.1.177
>>> help
```

If you get a timeout then you probably didn't set up the static IP address correctly. If you correctly see the same help message as before when you were talking over USB then everything is configured correctly!

See Section 4.3 for a full list of commands.

2.2 Getting Started

To run the normal QA/QC GUI you first need to export some environment variables so that the software can upload results to the database. To do so, you can run the following commands:

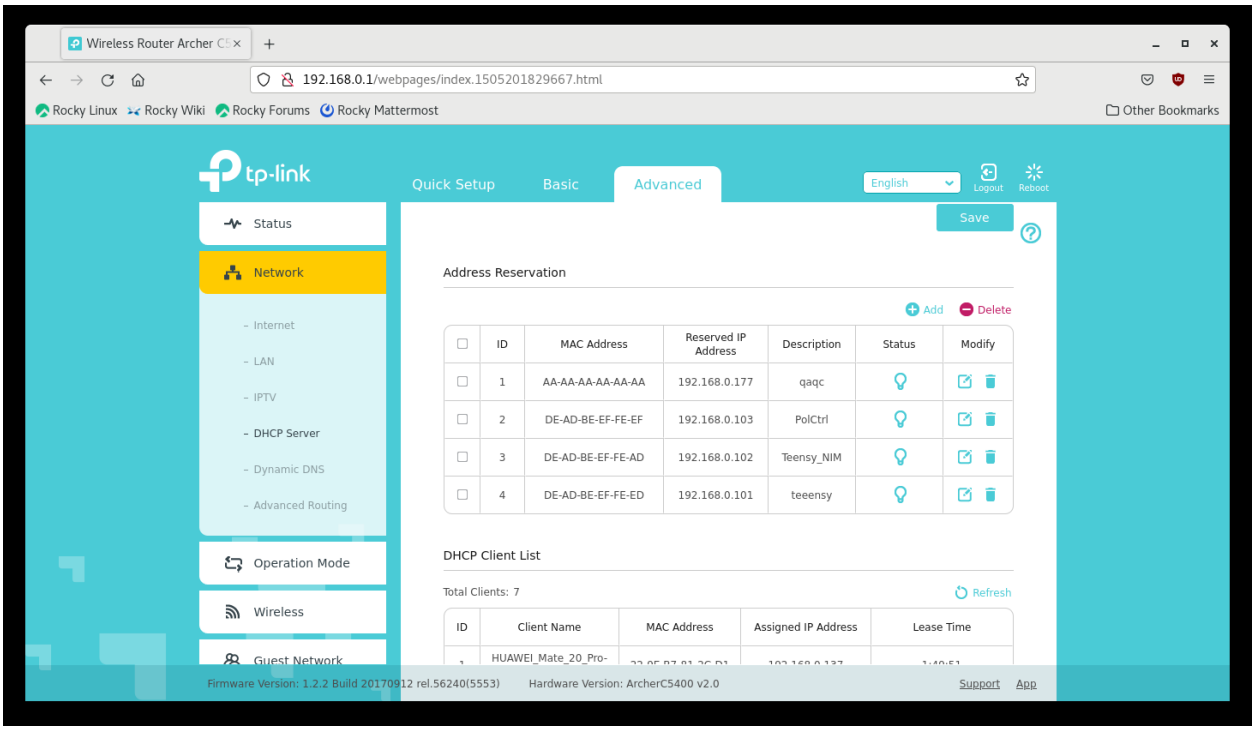


Figure 2.3: Webpage to set the DHCP IP address for the Teensy.

```
$ echo "export BTL_DB_HOST=btl-testing.hep.caltech.edu" >> ~/.bashrc
$ echo "export BTL_DB_PASS=[password]" >> ~/.bashrc
```

where you should replace [password] with the actual password. After running this command you should close and restart your terminal for the changes to take effect.

Next, to run the qaqc-gui:

```
$ qaqc-gui --ip-address 192.168.1.177
```

You should see a window like that in Figure 2.4.

First, you should make sure that the modules which you have connected are written in the GUI. The modules on the top of the GUI should be the ones closest to the control board. You can also check the “Present” checkbox to indicate which modules you wish to analyze. Finally, you can press the button labelled “Take Data” to actually take data.

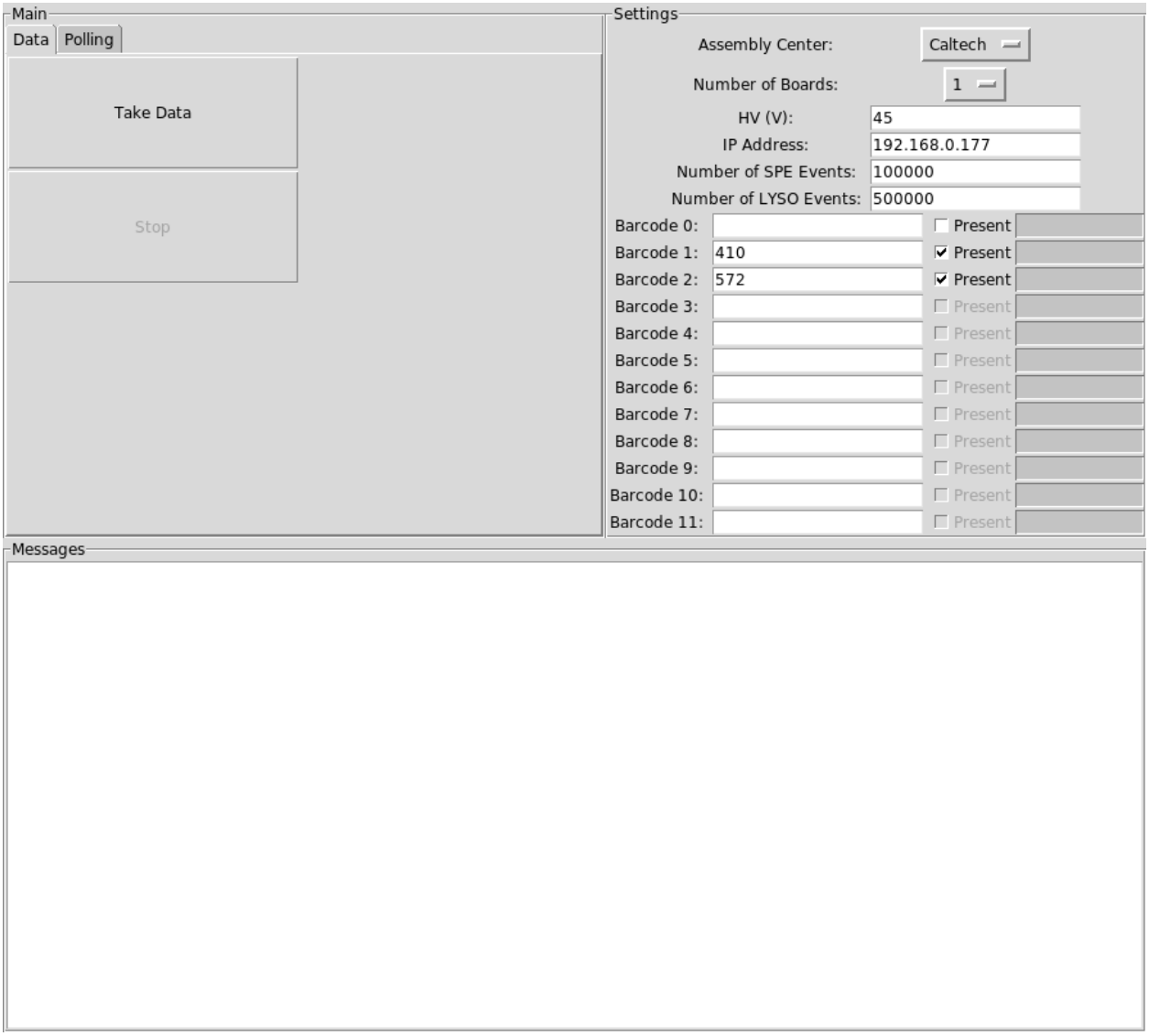


Figure 2.4: QA/QC GUI

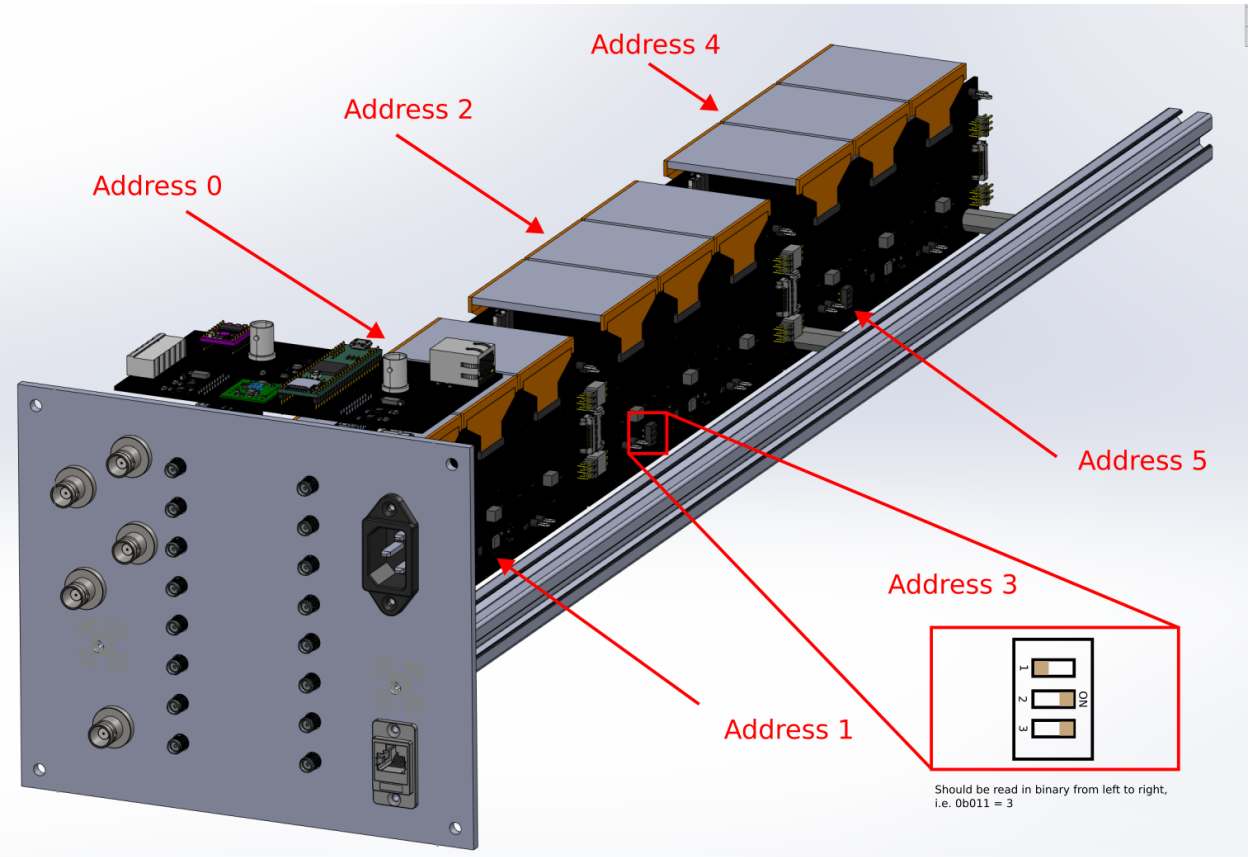


Figure 2.5: Solidworks rendering of the front of the QA/QC jig showing the correct addresses that need to be set on each board.

2.3 Addresses for the Boards

The control board talks with each board individually by addressing it on the I2C bus. In order for this to happen properly (i.e. to open the correct relay), it is necessary that the addresses be set properly. See Figure 2.5 to see how they should be set up. The board closest to the front panel on the left side should be address 0, the board opposite that should be address 1, the second board from the front panel on the left side should be address 2, etc. *Note that the address should be read in binary as if reading it from left to right. This may be confusing since the first dip switch is labelled 1 even though it is the 3rd binary digit.*

2.4 Hooking Channels up to the CAEN Digitizer

Figure 2.6 shows which channels on the front panel need to be hooked up to the CAEN digitizer so that the final channel numbers in the ROOT file make sense.

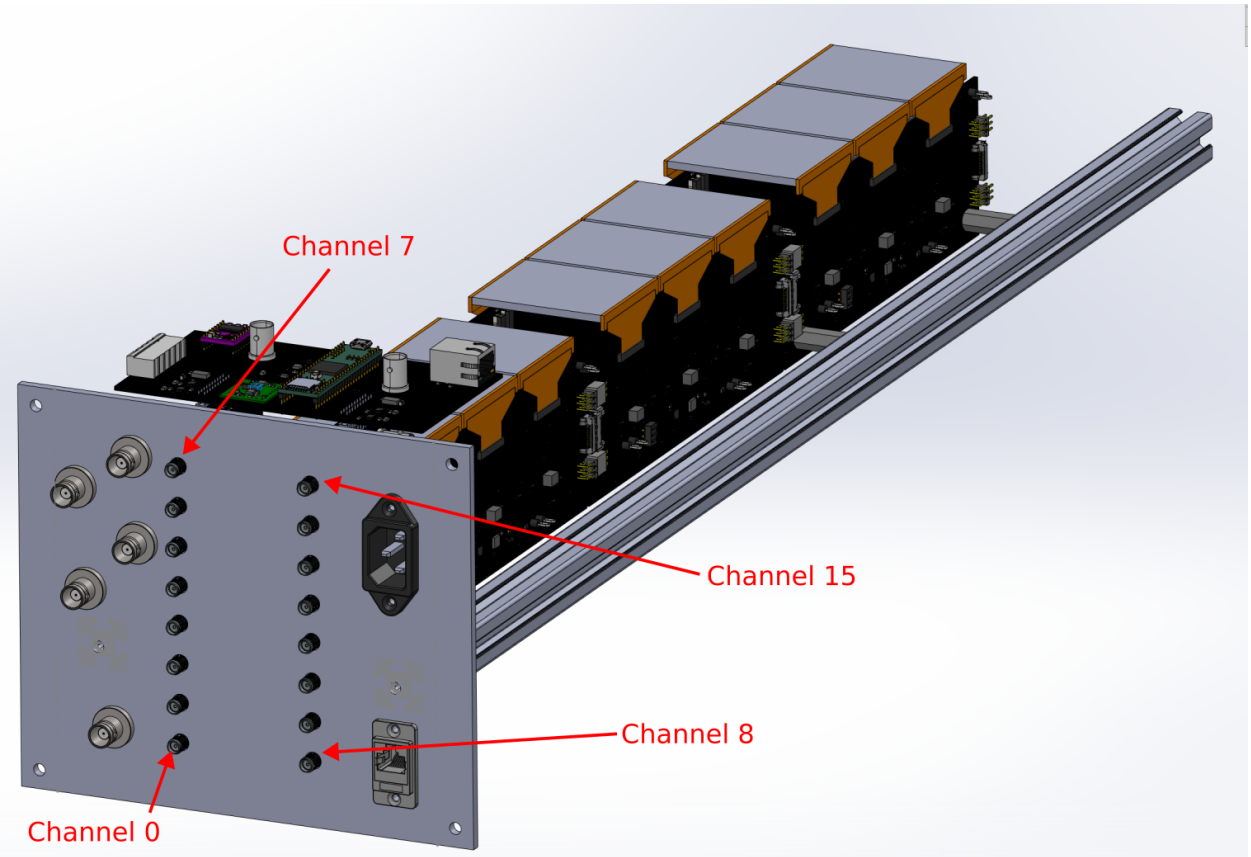


Figure 2.6: Solidworks rendering of the front of the QA/QC jig showing which channels on the front panel should be hooked up to which channels on the CAEN digitizer.

Object Name	Description
lyso_ch0;3	The final fit of the Lutetium-176 charge distribution
lyso_ch0;2	The initial fit of the Lutetium-176 charge distribution
lyso_ch0;1	The Lutetium-176 charge distribution
spe_ch0;3	The final fit of the SPE charge distribution
spe_ch0;2	The initial fit of the SPE charge distribution
spe_ch0;1	The SPE charge distribution
spe_ch0_fit;1	The TF1 object of the SPE fit
lyso_ch0_fit;1	The TF1 object of the Lutetium-176 fit
light_yield;1	The TGraph of the light yield versus channel number
pc_per_kev;1	The TGraph of the charge per unit energy deposited versus channel number
spe;1	The TGraph of the SPE charge versus channel number

Table 2.1: Table describing the various ROOT objects in the ROOT file. Only the data for channel 0 is shown in the table, but the actual ROOT file will contain histograms and function objects for all 32 channels.

2.5 Data Analysis

After running the GUI and pressing "Take Data" the script will take data and save it to a local HDF5 file. The name of this file will be `module_BARCODE.hdf5`. The size of this file will depend on the number of SPE events and LYSO events you take. For a "full" analysis, we recommend taking 100,000 SPE events and 500,000 LYSO events. This will produce an HDF5 file which is roughly 70 GB. However, we hope that as the fitting procedure is refined, for the final QA/QC the number needed will be closer to 10,000 and 50,000 respectively which should be closer to 7 GB.

After taking the data, the GUI will automatically analyze the data and produce a ROOT file called `module_BARCODE.root`. This ROOT file contains all the histograms of the SPE charge distributions and Lutetium-176 distributions, the fits for these histograms, a TGraph showing the light yield as a function of channel number, a TGraph showing the SPE charge versus channel number, and a TGraph showing the charge per unit energy deposited (from the LYSO fit) versus channel number. Table 2.1 contains a list of all the objects in the ROOT file.

Chapter 3

Hardware

The QA/QC jig should have come with the following pieces of hardware:

- 2 amplifier boards
- 2 "3 module boards" (eventually will send 4 to each institution)
- 1 control board
- 1 aluminum front panel
- 1 dark box (optional)
- 1 Intel NUC computer with a 2 TB NVME SSD (optional)
- 16 SMA to MCX connectors
- 1 DT5742 CAEN digitizer (each institution buys separately)
- 1 Meanwell RQ-125D power supply (+5,+12,-12,+24)

A Solidworks rendering of the minimal setup (without the dark box, digitizer, or power supplies) is shown in Figures 3.1 and 3.2.

3.1 Control Board

The control board is the "brains" of the setup. It contains a Teensy 4.1 microcontroller which is responsible for talking to the other boards over header pins and a I2C bus to control the HV relays, attenuation relays, and TEC relays, as well as to read out the RTD and thermistor temperatures, etc.

Power is supplied to the control board in the form of 3 voltages: +5V, -12V, and +24V (or +12V). These "raw" voltages are then put through linear regulators to supply the rest of the board's functionality. The board has a regulator to take the +5V raw signal and convert

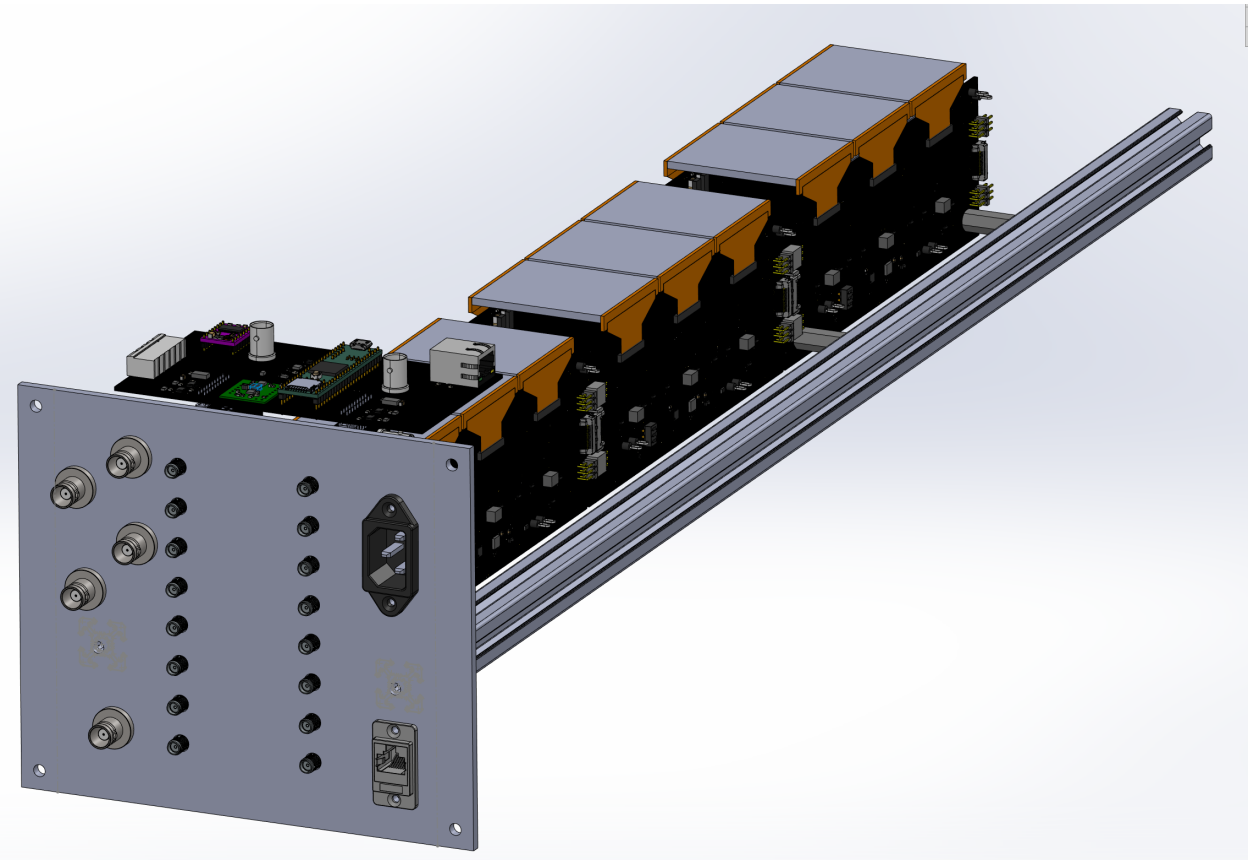


Figure 3.1: Solidworks rendering of the setup. The front panel connects to a dark box and contains BNC connections for the DC voltage rails as well as the 16 outputs which connect to the CAEN digitizer.

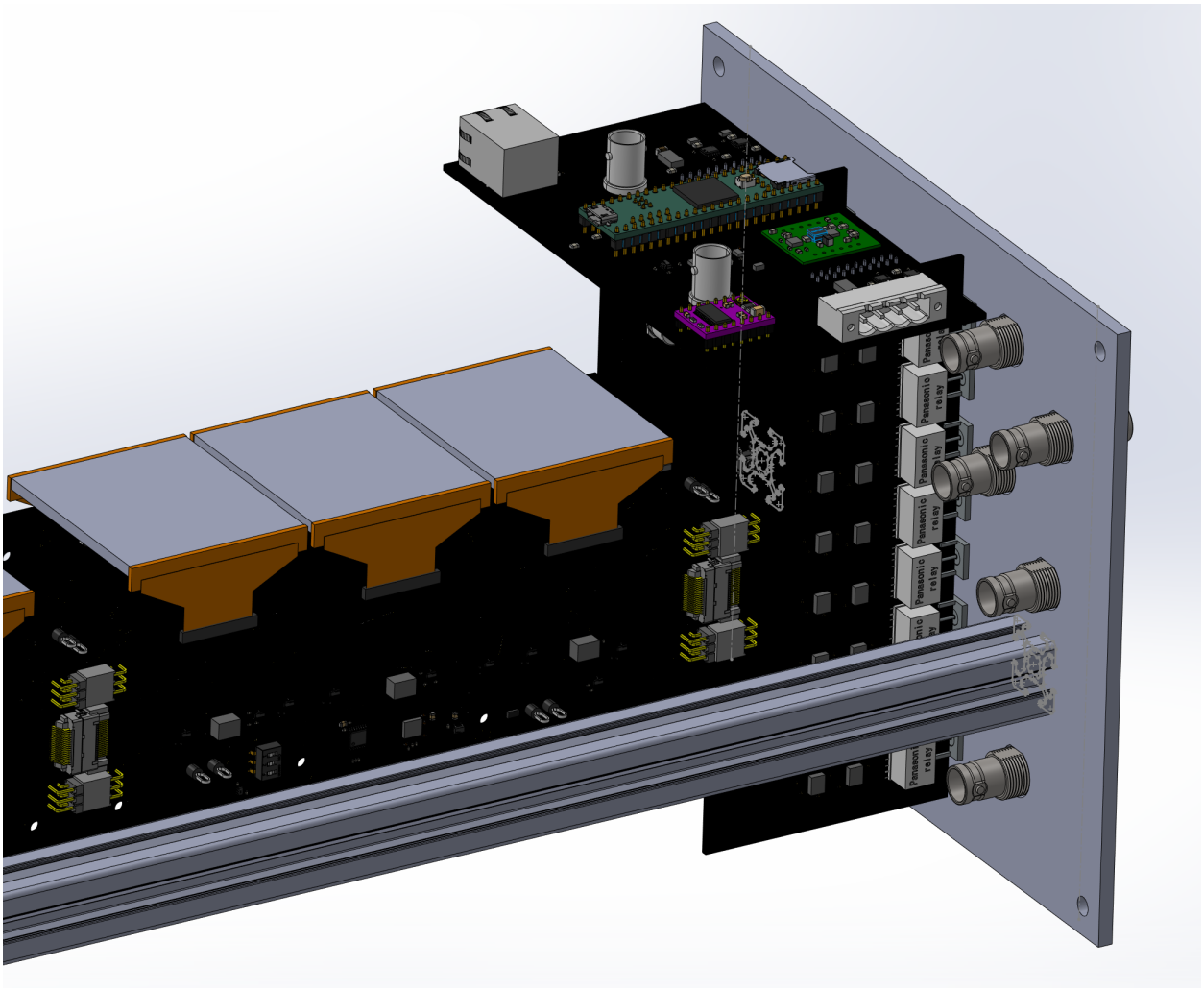


Figure 3.2: Solidworks rendering of the setup. Here you can see the three main boards: the board on top is the control board, the board connecting to the front panel is the amplifier board, and the board connected to the LYSO modules is the 3 module board. The 3 module boards can be daisy chained to support up to 12 modules (at which point we run out of address space on the I2C bus).

Input Voltage (V)	Output Voltage (V)	Purpose
+5	+5	Teensy and attenuator relays
	+3.3	HV and TEC relays
	+2	Amplifier rail
-12	-8	Amplifier rail
+24 (or +12)	> 40 V	DC/DC boost converter

Table 3.1: Summary of the various input and output voltages on the control board.

it to a "clean" +5V source for the Teensy and the attenuator relays (the big white panasonic relays on the amplifier board responsible for changing the gain of the system). The board also has a +3.3V regulator which produces the 3.3V signal which is used to turn the HV and TEC relays on and off. There are also linear regulators to convert the -12V and +5V "raw" voltages to +2V and -8V for the amplifier rails. Finally, there is a DC/DC boost converter which takes the +24V (or +12V) signal and boosts it up to more than 40V for the SiPM bias supply. Alternatively, there is a jumper which allows you to supply your own HV bias power supply. All of these are summarized in Table 3.1.

3.2 Amplifier Board

The amplifier board is responsible for taking 8 analog signals from the SiPMs and amplifying them to be able to read out by the CAEN digitizer. There are two gain paths which are controlled by a relay at the end of the board near the SMA connectors: the unattenuated high gain path and the attenuated low gain path.

To be able to detect single photons on the unattenuated high gain path, the current signal from the SiPM is about 10 microamps. This signal then goes through the first stage transimpedance amplifier with a gain of approximately 500 V/A. This turns the 10 microamp signal into a 5 mV voltage signal. This signal then goes through the second stage non-inverting amp stage with a gain of approximately 4 and we get out a 20 mV voltage signal.

3.2.1 Attenuation

The CAEN board can only detect voltage signals with a 1V dynamic range. Therefore it is necessary to attenuate the output signal when we are looking at higher energy radioactive sources. When the relay closes, the signal from the output of the second stage goes through a pi pad attenuator which reduces the signal by a factor of approximately 5.85.

Initially this pi pad attenuator was designed to have an attenuation of a factor of 4 and an impedance of 50 ohms to match the transmission line of the SMA cable. However, when testing the new higher gain SiPMs we realized that we were actually being limited by the 50 mA current limit on the second stage op amp, and so couldn't even drive the full 4V necessary from the second stage to hit the 1V limit of the CAEN after the attenuator. To solve this problem we instead designed the pi pad attenuator to have an impedance of 100 ohms. Now,

since we are not actually driving a 100 ohm load (the CAEN is fixed to 50 ohms), this means that the voltage seen at the CAEN board will no longer be a factor of 4 smaller than the input voltage, but the op amp will see a higher resistance and thus need to supply a lower current. To find out the actual voltage ratio between the output of the second op amp and what we see on the CAEN digitizer we wrote a short python script which can be found at `https://github.com/alatorre-caltech/dt5742/blob/master/pi_pad_calculator.py`.

3.3 3 Module Board

The 3 module board is responsible for connecting to the actual scintillator modules and controlling the HV relays. The way that we multiplex up to 12 modules with 384 channels to be read out by only 16 digitizer channels is by connecting multiple SiPM channels to the same output line and then selectively turning on the bias voltage for only a single channel at a time. This is done by sending a command on the I2C bus to tell a small PCA9557PW chip to set an output high which in turn starts conducting a MOSFET which causes a relay to close allowing 8 channels to see the SiPM bias voltage.

In addition, this board contains the electronics to read out the RTDs on each SiPM. The design of this system is discussed in more detail in Appendix A.

This board can also read out the current flowing through each set of TECs. To measure the total TEC resistance (which is a good indicator of whether they are broken), we use the Teensy to turn the TEC relays for a single module on for approximately 1 ms. During this 1 ms, we quickly measure the current flowing to the TEC and from this we can determine the overall resistance. We keep the time the current is flowing to 1 ms or less because as you flow more current through the TECs they will start to heat up/cool down and the resistance will change. The function which computes this can be found at `https://github.com/alatorre-caltech/dt5742/blob/master/firmware/firmware.ino#L622`.

Chapter 4

Software

There are several different software programs that have been developed to interact with the QA/QC jig which will be described in the sections below. All of the code is at `https://github.com/alatorre-caltech/dt5742`[1]. The installation is discussed in Chapter 2, but can be done simply by running:

```
$ git clone https://github.com/alatorre-caltech/dt5742
$ cd dt5742
$ make install-deps
$ make
$ sudo make install
```

During the last command, all of the scripts are installed to /usr/local/bin so that they are available via the command line without having to be in the directory anymore. You can test this out by running:

[1]The name of this repository is the model number for the CAEN digitizer we are using and was chosen because it started just to have the C code to take data from the digitizer

```
$ cd
$ wavedump --help
usage: wavedump -o [OUTPUT] -n [NUMBER] [CONFIG_FILE]
  -t, --trigger Type of trigger: "software", "external", or "self".
  -l, --label Name of hdf5 group to write data to (lyso, spe)
  --threshold Trigger threshold (volts) (default: -0.1)
  --gzip-compression-level
              gzip compression level (default: 0)
  --channel-map
              which half of the module is being recorded (default: -1)
  --help Output this help and exit.
```

4.1 wavedump

This is a C program which takes data from the CAEN digitizer and writes it out to HDF5 files which can then be analyzed from python scripts. You can find the main program at `https://github.com/alatorre-caltech/dt5742/blob/master/wavedump/src/wavedump.c`. A simple example to take 100,000 software triggered events for single photon analysis is:

```
$ wavedump -o output.hdf5 --trigger software -n 10000 -l spe
```

You can then look at the waveforms using python:

```
$ python
>>> import h5py
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> f = h5py.File("output.hdf5","r")
>>> xinc = 1/(f['spe'].attrs['drs4_frequency'] * 10**6)
>>> points = f['spe'].attrs['record_length']
>>> x = np.linspace(0, xinc * points, int(points)) - xinc * points * (1
   ↪  - f['spe'].attrs['post_trigger']/100)
>>> data = f['spe']['ch0'][:100] # get the first 100 events
>>> data -= np.median(data,axis=-1)[:,np.newaxis] # subtract baseline
>>> data /= 2**12
>>> plt.plot(x,data[0])
>>> plt.show()
```

Similarly, you can take 100,000 self triggered events to take data with a source (or to just look at the energy distribution of the internal Lutetium-176 background radiation:

```
$ wavedump -o output.hdf5 --trigger self --threshold -0.1 -n 10000 -l
   ↪ lyso
```

4.2 analyze-waveforms

Analyze-waveforms is a python script used to analyze data taken with the jig. You can find the source code at `https://github.com/alatorre-caltech/dt5742/blob/master/python/analyze-waveforms`. Specifically it does the following things:

- baseline subtraction
- integrates waveforms over a fixed window to get charge distributions
- fits single photon charge distribution to determine the single photon charge
- fits the Lutetium-176 charge distribution to determine the charge per unit energy deposited
- uploads results to the database

- writes out the light yield and all distributions and fits to a ROOT file

Supposing you took both single PE data and LYSO data with the following commands:

```
$ wavedump -o output.hdf5 --trigger software -n 100000 -l spe
$ wavedump -o output.hdf5 --trigger self --threshold -0.05 -n 500000 -l
   ↪ lyso
```

You can analyze it using the following command:

```
$ analyze-waveforms output.hdf5 -o output.root
```

This will output a ROOT file called output.root which contains all the charge distributions, fits, and measured light yields.

4.3 qaqc-client

This is a small python script used to talk to the Teensy over UDP. An example:

```
$ qaqc-client --ip-address 192.168.1.177
>>> help
```

4.3.1 Commands

The following is a list of commands you can send to the Teensy (note: for normal operations you should always use the GUI, but this is for advanced users).

tec_write CARD TEC ON/OFF This command turns the TEC relays on or off. CARD should be the binary address set on the DIP switch on the 3 module board, TEC should be 0, 1, or 2 and specifies which relay to open on the board.

hv_write CARD TEC ON/OFF This command turns the HV relays on or off.

thermistor_read CARD ADDRESS This command reads the RTD temperature in Celsius on a given board. ADDRESS should be 0, 1, or 2.

tec_sense_read CARD This command reads the low level current measurement for the TEC current draw.

tec_check CARD ADDRESS This command reads the resistance of the TEC on board CARD. ADDRESS should be 0, 1 or 2. It does this by turning on the relay for the given TEC for 1 ms and then quickly measuring the current. This makes sure that the TEC doesn't have time to heat up and change resistance. A working TEC should return about 14 ohms.

reset This command resets the board.

poll CARD ON/OFF Set up automatic polling of the temperatures and TEC current for a given board. The output is only visible in the serial monitor.

set_active_bitmask BITMASK Gives a bitmask of which boards are present. This prevents errors when resetting the board. For example, if you only have two boards present (0 and 1) you could run set_active_bitmask 0x3.

debug ON/OFF Turn debug messages on or off.

set_attenuation ON/OFF Turn on or off the attenuation. You should hear a series of clicks when you run this command.

step_home Move the stepper motor to home (not tested at all)

step STEPS Move the stepper motor a certain number of steps (not tested at all)

bias_iread Reads the bias current from the DC/DC boost converter. Note that this current is not super useful as we have diodes with bias resistors in series with the SiPMs that will draw a few milliamps whenever you have relays open.

bias_vread Reads the bias voltage from the DC/DC boost converter.

extmon_vread Reads the bias voltage independent of whether you are using the DC/DC boost converter or an external supply. Note that this value should be calibrated to be the exact value applied to the SiPM so it will be lower than what you measure on the board since there is a voltage drop across the diode in series with the SiPM.

set_hv VOLTAGE Sets the voltage output by the DC/DC boost converter.

disable_hv Disable the DC/DC boost converter.

enable_dac Enable the DAC for the DC/DC boost converter.

4.4 qaqc-gui

The QA/QC GUI is a python Tkinter script which can be found at `https://github.com/alatorre-caltech/dt5742/blob/master/python/qaqc-gui`. To run the GUI you can run:

```
$ qaqc-gui --ip-address 192.168.1.177
```

Figure 4.1 shows the QA/QC GUI window and some of the functions.

Most of the logic in the GUI is for taking data. When you press the "Take Data" button the GUI does a series of steps:

1. Opens all HV relays
2. Sets the HV to the value specified in the GUI
3. Closes the two relays on either side of the jig for the first module
4. Turns the attenuation off to take single PE data
5. Runs the wavedump program to take single PE data
6. Turns the attenuation on to take Lutetium-176 data
7. Runs the wavedump program to take Lutetium-176 data
8. Opens the two relays
9. Repeats steps 3-8 for the next set of relays for the first module
10. Analyze the data using the analyze-waveforms program
11. Repeat all the above steps for any additional modules

The tricky part in doing all this is making sure that you open the correct relays and mapping the channels in the output file so that they have some physical meaning. The mapping between physical locations on the module and the channel number in the HDF5 file is shown in Figure 4.2.

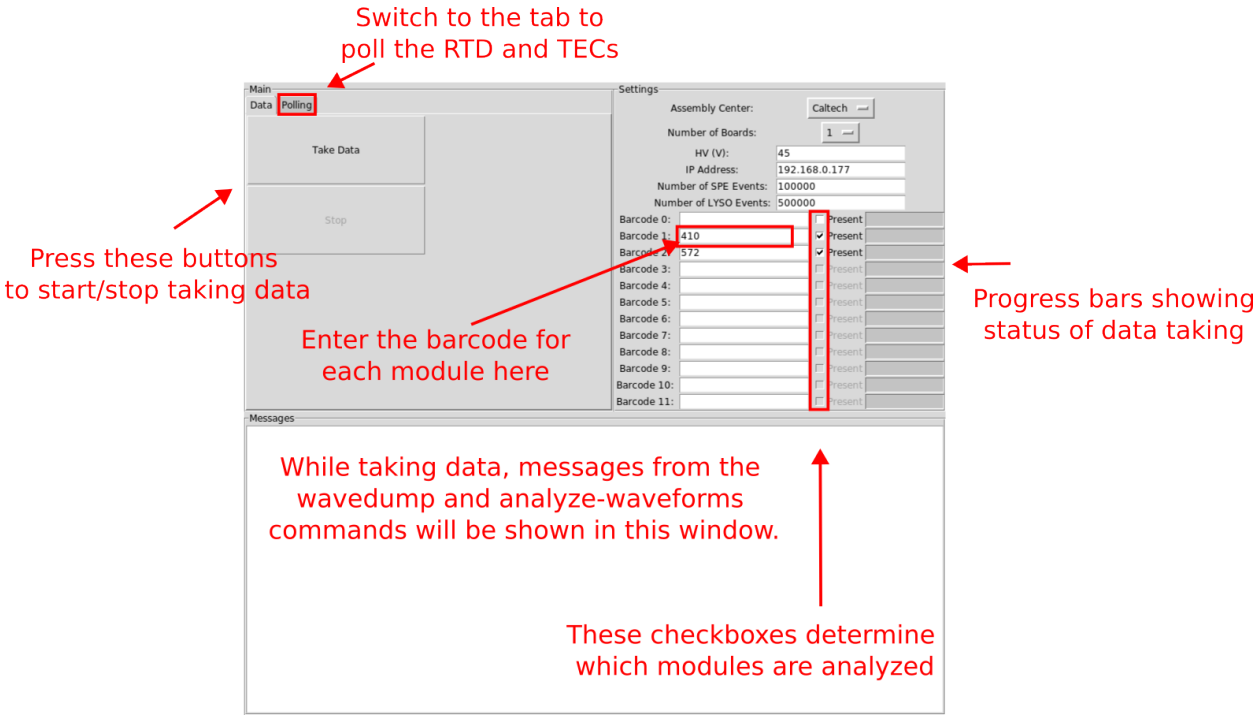


Figure 4.1: A screenshot of the qaqc-gui program along with descriptions of what the buttons and inputs do.

4.5 Website

To easily see the results of the analysis, we put together a website at `http://btl-testing.hep.caltech.edu`. You can see an example screenshot in Figure 4.3. The main page shows a list of recently tested modules along with voltage, the average light yield, and whether it passed. You can click on any one of these rows and it will then show you all the tests that have been run for that specific module. Clicking on a row here will take you to the page shown in Figure 4.4 which shows the results for a single run. Clicking on a channel at the bottom of this page will take you to the page shown in Figure 4.5 which shows the results for a single channel.

All of the code powering the website can be found at `https://github.com/alatorre-caltech/dt5742/tree/master/website`. This code is currently running on a virtual machine on the Caltech Tier2 cluster, but can in principle be run anywhere.

The main component of the webpage is the database which holds all the results. This is a postgres database whose schema can be found at `https://github.com/alatorre-caltech/dt5742/blob/master/website/btl_qa.sql`. When you take data and upload it, the data gets uploaded to the database. Then, when you go to the webpage, the server queries the database and displays the data.

The web server is a flask python script which operates behind nginx. Nginx initially takes the web request, handles any static requests (since python is relatively slow), and forwards

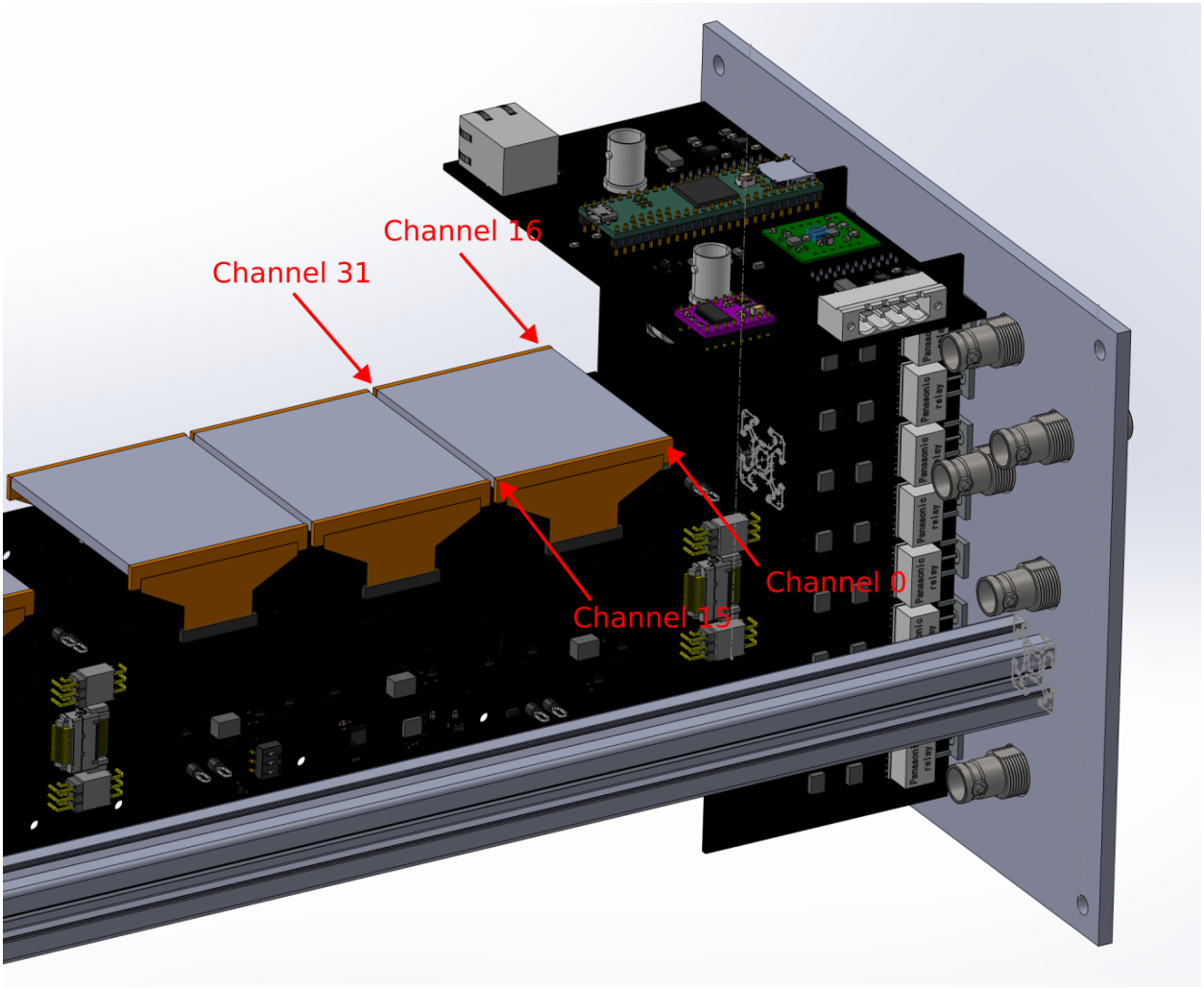


Figure 4.2: A Solidworks rendering of the back of the QA/QC jig with the mapping between the physical location on the module and channel number annotated.

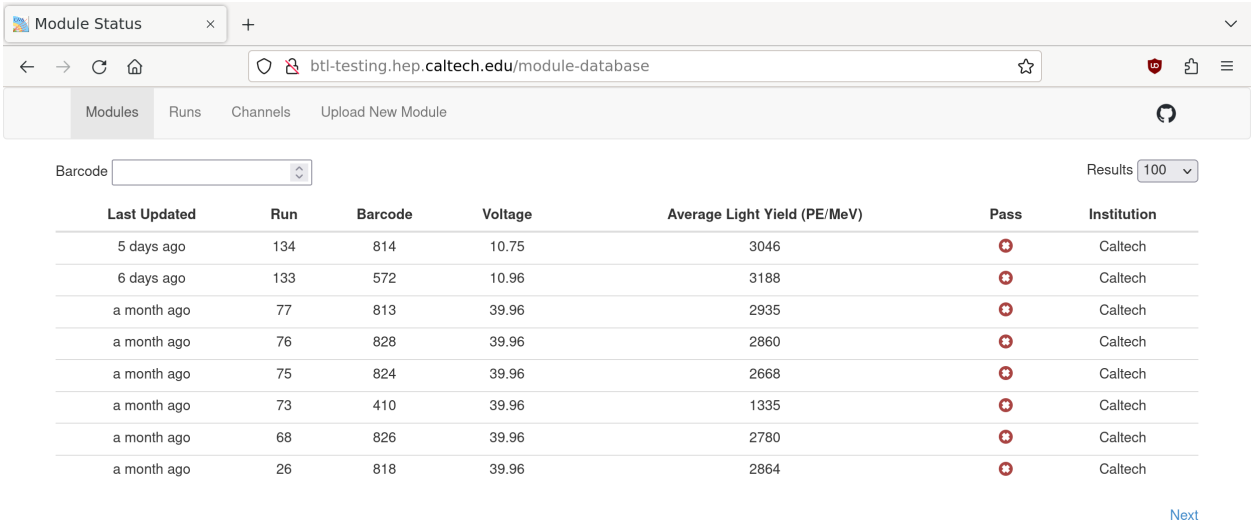
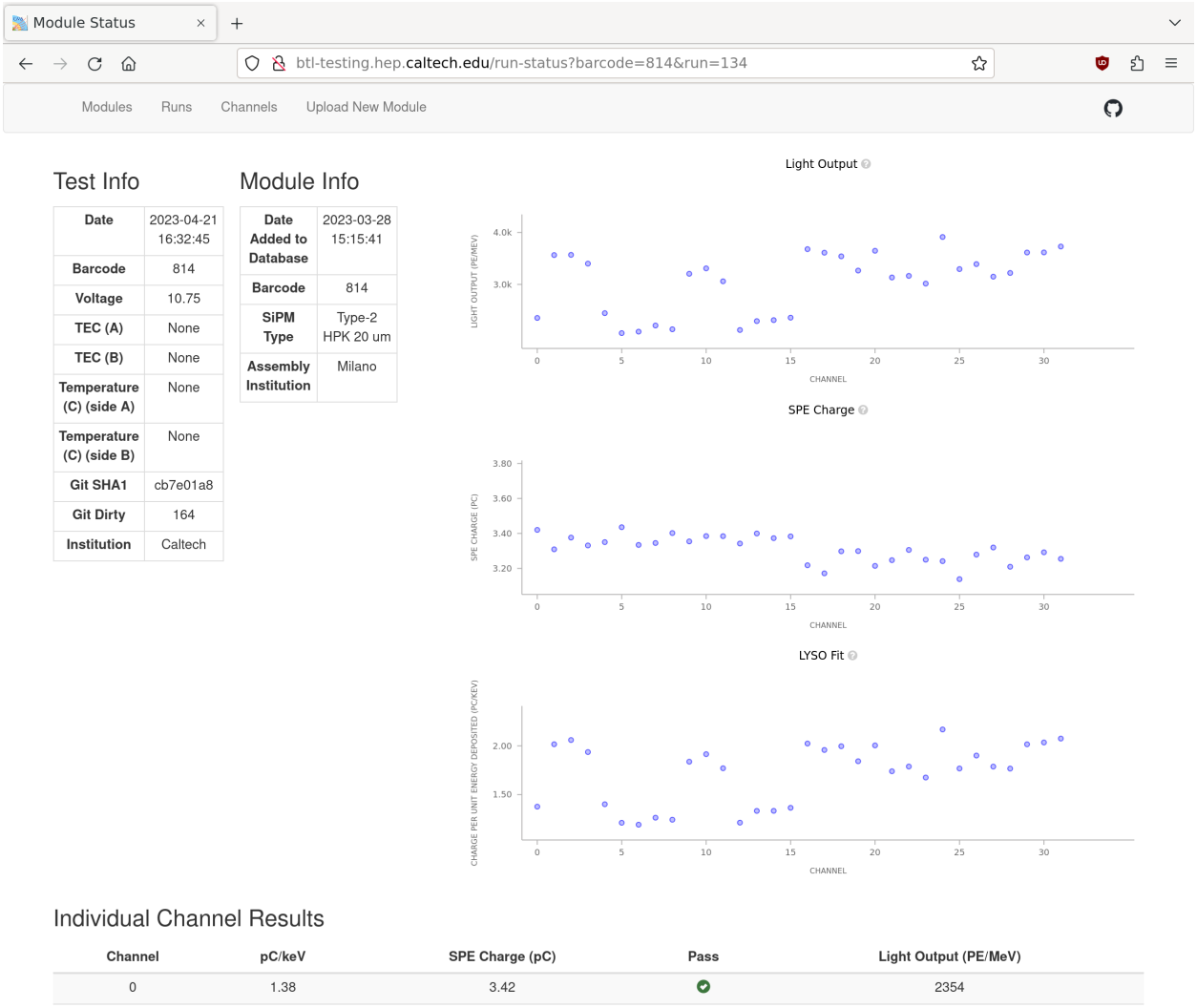


Figure 4.3: Screenshot of the main page.

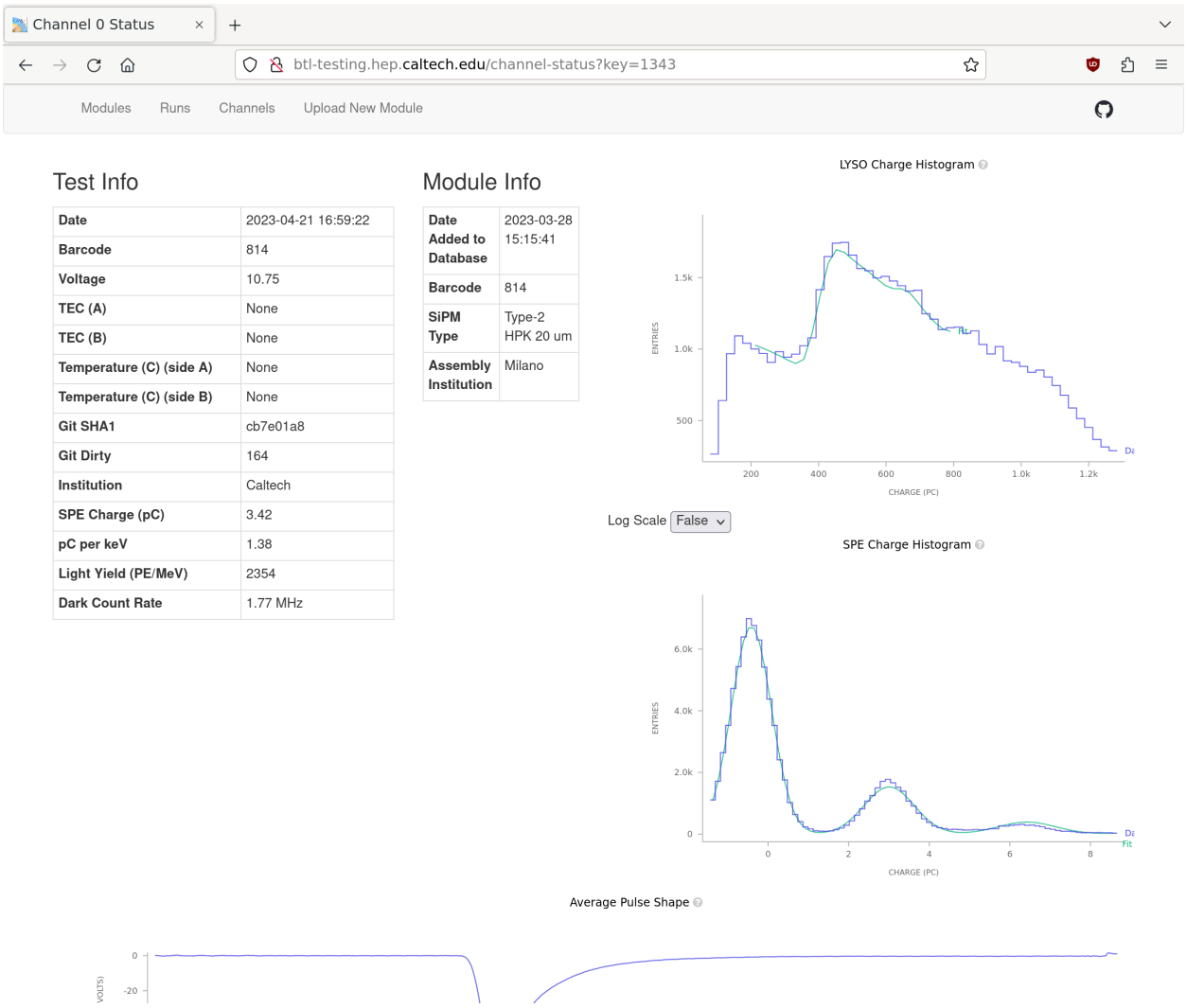


Date	2023-04-21 16:32:45
Barcode	814
Voltage	10.75
TEC (A)	None
TEC (B)	None
Temperature (C) (side A)	None
Temperature (C) (side B)	None
Git SHA1	cb7e01a8
Git Dirty	164
Institution	Caltech

Date Added to Database	2023-03-28 15:15:41
Barcode	814
SiPM Type	Type-2 HPK 20 um
Assembly Institution	Milano

Channel	pC/keV	SPE Charge (pC)	Pass	Light Output (PE/MeV)
0	1.38	3.42	✔	2354

Figure 4.4: Screenshot of the page showing the status of 1 run. The upper right corner shows a plot of the light output vs channel.



Test Info	
Date	2023-04-21 16:59:22
Barcode	814
Voltage	10.75
TEC (A)	None
TEC (B)	None
Temperature (C) (side A)	None
Temperature (C) (side B)	None
Git SHA1	cb7e01a8
Git Dirty	164
Institution	Caltech
SPE Charge (pC)	3.42
pC per keV	1.38
Light Yield (PE/MeV)	2354
Dark Count Rate	1.77 MHz

Module Info	
Date Added to Database	2023-03-28 15:15:41
Barcode	814
SiPM Type	Type-2 HPK 20 um
Assembly Institution	Milano

Figure 4.5: Screenshot of the page showing the results of a single channel. The upper right corner shows a plot of the Lutetium-176 charge distribution and the fit and the plot below that shows the single photoelectron charge fit.

any dynamic requests (to fetch data from the database for example) to the flask app. This architecture also allows us to use multiple flask processes to increase the bandwidth. In principle, it should support 100s of concurrent connections.

Chapter 5

Fitting LYSO Intrinsic Spectrum

This chapter summarizes the model we use to fit the intrinsic LYSO radioactivity spectrum for the Caltech QA/QC module tester.

5.1 Introduction

LYSO has an instrinsic unstable isotope of ^{176}Lu which undergoes a beta decay with endpoint 593 keV to ^{176}Hf[1]. After undergoing beta decay, the Hf nucleus then de-excites emitting gammas with energies of 88 keV, 202 keV, and 307 keV. We can use this intrinsic radioactivity to determine the light yield in terms of charge per unit energy deposited.

Ideally we would like to make the analysis simple by looking simply for the gamma lines so we only have to fit a gaussian distribution. However, when we tested our ability to do this by looking for coincidences between the initial beta decay in one bar and then looking for the gammas to travel to a neighboring bar there were two big problems. First, if we looked at bars two neighbors down from the bar which had the beta decay, the rate for observing the 202 keV and 307 keV gamma were so low that it would take many hours to get enough statistics to barely be able to make out the gamma lines. Second, if we look in the direct neighboring bar, although we are able to see the gamma lines much faster, we were worried that the optical crosstalk would make a measurement of the light yield unreliable. In addition, the fact that we required coincidences meant that we could no longer determine the light yield of a bar independently of the other bars. Therefore, we decided to fit the full beta decay spectrum by looking at the charge in a single bar and not looking for coincidences.

5.2 Fitting the Intrinsic LYSO Spectrum

We can calculate the expected charge distribution as follows:

$$P(q \mid \hat{y}) = \int_E P(q \mid E, \hat{y}) P(E) \mathrm{d}E \tag{5.1}$$

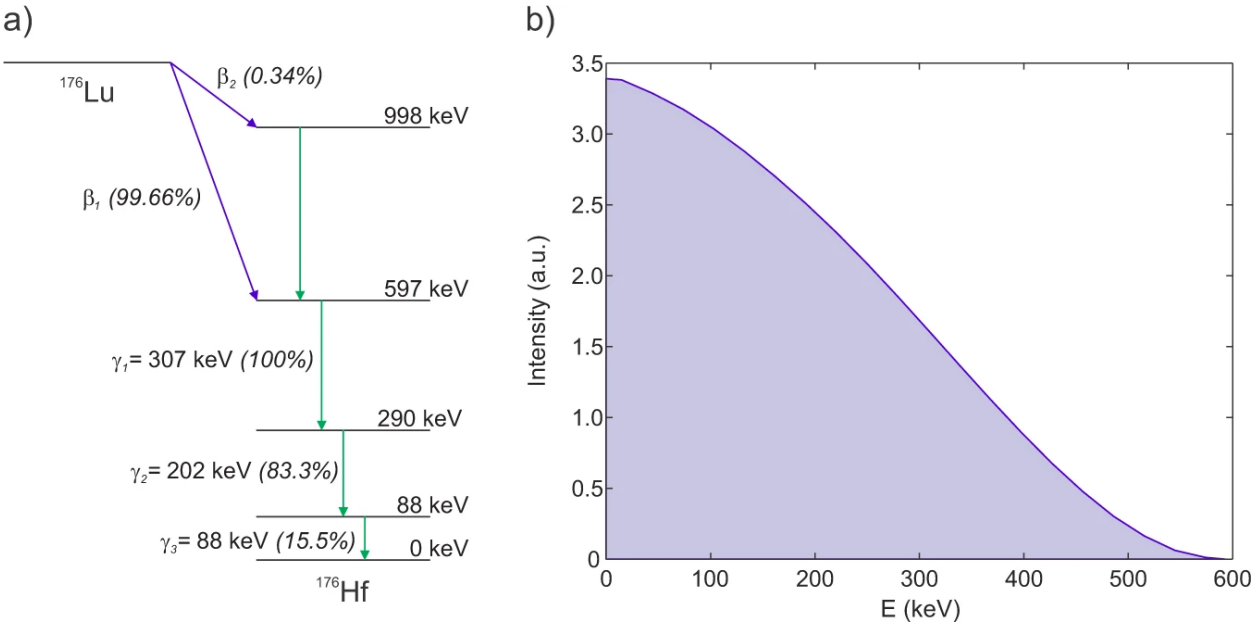


Figure 5.1: LYSO decay scheme from Reference [1]

where $P(q \mid \hat{y})$ is the probability of observing charge q given an average charge per energy deposited $\hat{y}$ (the average charge per energy deposited is assumed to be at the center of the bar), and $P(E)$ is the probability of observing an energy E in keV deposited in the crystal. $P(E)$ is given by a weighted sum of beta decay distributions offset by the possible gamma captures, i.e.

$$P(E) = \alpha_{88}\beta(E - 88\text{keV}) + \alpha_{202}\delta(E - 202) + \alpha_{290}\beta(E - 290\text{keV}) + \alpha_{307}\delta(E - 307) + \alpha_{395}\beta(E - 395\text{keV}) + \alpha_{509}\delta(E - 509) + \alpha_{597}\beta(E - 597\text{keV}) \quad (5.2)$$

where $\beta(E)$ is the beta decay distribution for ^{176}Lu, α_{88} is the probability of the beta decay and the 88 keV gamma capturing, α_{290} is the probability of the beta decay and the 88 keV and 202 keV gamma capturing, $\alpha 202$ is the probability of only observing the 202 keV gamma, etc. We do not include any terms in which the 88 keV gamma doesn't capture since that is very unlikely[1].

Next, we integrate over the light yield at different points along the bar. Although this is likely not perfectly symmetric in the real bars, we assume it is here for simplicity of fitting since we don't expect a more complicated model to have any noticeable effect on the final fit.

$$P(q \mid \hat{y}) = \int_E P(E) \int_y P(q \mid E, y, \hat{y}) P(y \mid \hat{y}, E) \mathrm{d}y \mathrm{d}E \quad (5.3)$$

We can now make some simplifications. Since the light yield doesn't depend on the energy E

$$P(y \mid \hat{y}, E) \equiv P(y \mid \hat{y}) \tag{5.4}$$

Also, the probability of observing a charge q is independent of $\hat{y}$ once y is specified

$$P(q \mid E, y, \hat{y}) \equiv P(q \mid E, y) \tag{5.5}$$

Therefore,

$$P(q \mid \hat{y}) = \int_E P(E) \int_y P(q \mid E, y) P(y \mid \hat{y}) \mathrm{d}y \mathrm{d}E \tag{5.6}$$

Now, in principle we would sum over the number of photoelectrons produced, but we would like this fit to be independent of the single photoelectron (SPE) charge. Instead, we assume that the charge distribution for a given energy E and light yield y is given by

$$P(q \mid E, y) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(q-Ey)^2}{2\sigma^2}} \tag{5.7}$$

where σ is an approximation made by assuming an average SPE charge

$$\sigma(y) \approx \sqrt{\frac{Ey}{q_{\mathrm{SPE}}}} q_{\mathrm{SPE}} \tag{5.8}$$

Therefore

$$P(q \mid \hat{y}) = \int_E P(E) \int_y \frac{1}{\sqrt{2\pi\sigma(y)}} e^{-\frac{(q-Ey)^2}{2\sigma(y)^2}} P(y \mid \hat{y}) \mathrm{d}y \mathrm{d}E \tag{5.9}$$

Finally, we make the simplifying assumption that $P(y \mid \hat{y})$ is a constant over some range $\hat{y} - \Delta y$ to $\hat{y} + \Delta y$:

$$P(q \mid \hat{y}) = \frac{1}{2\Delta y} \int_E P(E) \int_{\hat{y}-\Delta y}^{\hat{y}+\Delta y} \frac{1}{\sqrt{2\pi\sigma(y)}} e^{-\frac{(q-Ey)^2}{2\sigma(y)^2}} \mathrm{d}y \mathrm{d}E \tag{5.10}$$

The second integral can be written in the following closed form solution:

$$\begin{aligned} P(q \mid \hat{y}) = \frac{1}{2\Delta y} \int_E P(E) \\ \frac{1}{2E} \left[-\mathrm{Erf}\left(\frac{-q + (\hat{y} - \Delta y)E}{\sqrt{2(\hat{y} - \Delta y) q_{\mathrm{SPE}} E}} \right) + \mathrm{Erf}\left(\frac{-q + (\hat{y} + \Delta y)E}{\sqrt{2(\hat{y} + \Delta y) q_{\mathrm{SPE}} E}} \right) \right. \\ \left. + e^{\frac{2q}{q_{\mathrm{SPE}}}} \left(\mathrm{Erf}\left(\frac{q + (\hat{y} + \Delta y)E}{\sqrt{2(\hat{y} + \Delta y) q_{\mathrm{SPE}} E}} \right) - \mathrm{Erf}\left(\frac{q + (\hat{y} - \Delta y)E}{\sqrt{2(\hat{y} - \Delta y) q_{\mathrm{SPE}} E}} \right) \right) \right] \mathrm{d}E \end{aligned} \tag{5.11}$$

Note that computing this is very susceptiple to floating point problems. The two terms with $e^{\frac{2q}{q_{\mathrm{SPE}}}}$ should be grouped together to avoid problems.

This integral is performed numerically using numpy's trapz function.

5.3 Cuts

The likelihood function described in the previous section assumes no crosstalk between channels. The actualy LYSO crystals will have a significant (approximately 15%) amount of crosstalk between neighboring crystals. This crosstalk is caused primarly by photons leaking from one SiPM to the next in the glue layer between the SiPMs and the LYSO. Modelling this crosstalk in the likelihood function is possible but tricky because we don't know the crosstalk amount a priori and it may be more or less than the average depending on the location of the event and how well everything was glued.

To cut events which are likely to contain crosstalk we cut events in which there is a significant amount of charge in neighboring crystals (see `https://github.com/alatorre-caltech/dt5742/blob/ee9e61a448b6ab8e9ddb229288ae692085de7eb6/python/analyze-waveforms#L539`. The idea here is to cut any events in which a gamma escaped the current crystal and interacted in a neighboring crystal. This works surprisingly well and produces fits which fit surprisingly well (see Figure 5.2). The only problem is with channels 7, 8, 23, and 24 which are near the middle of the module. For these channels, there is always an adjacent channel which is not powered when we are taking data. This means that it is possible for a beta decay to happen in an unpowered bar and a gamma to scatter in the powered bar. In this case, we will get a lot of crosstalk from the adjacent unpowered bar. In the default fit, we actually fix the gamma peak parameters to 0 since they should be negligible. For these specific channels, we do allow the gamma peak parameters to float and this seems to work ok (it still doesn't address the fact that we don't model the crosstalk).

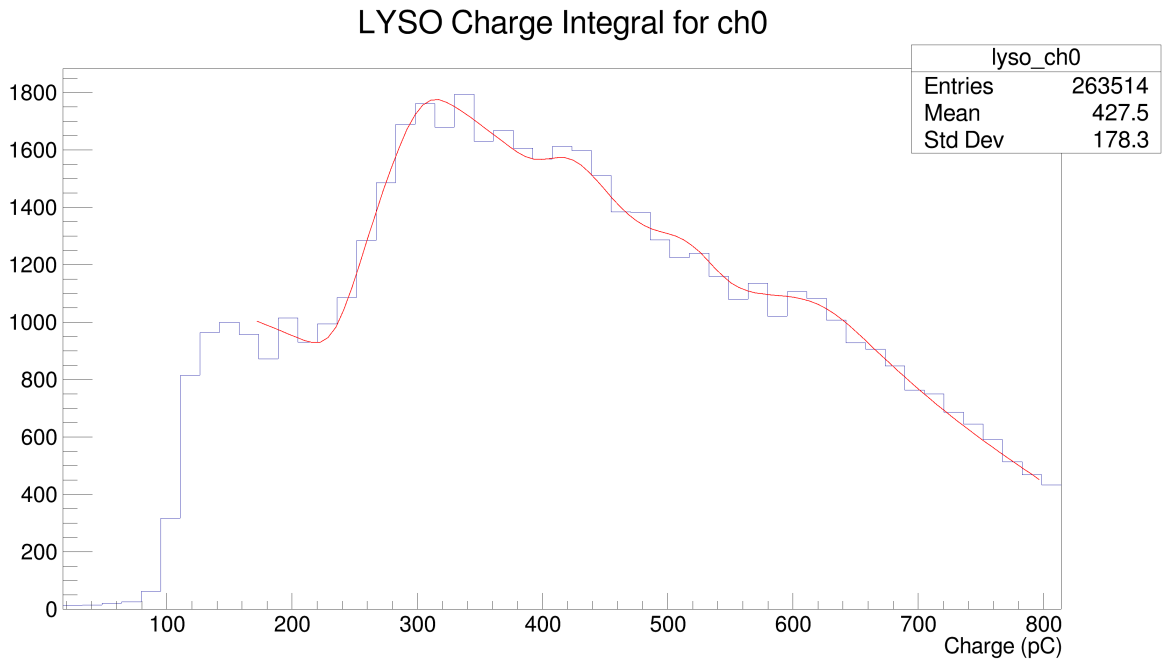


Figure 5.2: Fit to the Lutetium-176 distribution. The sharp edge on the left is due to the trigger threshold, so we don't fit that low. The highest peak is the start of the beta decay spectrum offset by 290 keV (88 keV and 202 keV gamma absorbed).

Chapter 6

Fitting Single Photoelectron Charge

The fitting of the SPE charge histogram presents a challenge because of the relative infrequency of dark SPEs and weakness of their signal. Thus, the SPE charge histogram does not always yield discernible peaks (as can be seen in Figure 6.1), requiring that a mathematical model be used to fit the histogram and extrapolate the SPE charge. Our model includes seven parameters to account for all the major sources of variability and noise. We first assume that the charge of an SPE follows a Gaussian distribution with mean μ and standard deviation σ, which we denote $G_x(\mu, \sigma)$, where G_x is the probability density of measuring charge x. The charge distribution of n SPEs is then G convolved with itself n times, which is $G_x(n\mu, \sigma\sqrt{n})$ since $G_x(\mu_1, \sigma_1) * G_x(\mu_2, \sigma_2) = G_x(\mu_1 + \mu_2, \sqrt{\sigma_1^2 + \sigma_2^2})$.

We now consider the probability that n SPEs are detected in a given integration window. If the detection of SPEs were independent events, n would follow a Poisson distribution[1]. However, according to Agnes et al. [2], there is a large enough probability that one SPE be correlated to the production of another (which can arise from SiPM pixel cross-talk) for this not to be the case. A distribution to model such behavior was conceived by Vinogradov et al. [3]. Given the mean number of SPEs detected λ and the probability that an SPE trigger a secondary SPE p, the probability of detecting n SPEs is given by[2]

$$V_n(\lambda, p) = \frac{e^{-\lambda}}{n!} \sum_{i=0}^{n} B(i, n) \cdot (\lambda(1-p))^i \cdot p^{n-i} \tag{6.1}$$

where

$$B(i, n) = \begin{cases} 1 & \text{if } i = 0 \text{ and } n = 0 \\ 0 & \text{if } i = 0 \text{ and } n > 0 \\ \frac{n!(n-1)!}{i!(i-1)!(n-i)!} & \text{otherwise} \end{cases} \tag{6.2}$$

[1]For a given number of SPEs produced, each of them have a probability of falling within the integration window. If we assume that each SPE has the same probability of being in this window, then n would follow a binomial distribution. However, we make the reasonable assumption that the total number of SPEs being produced is very large, and the probability of an SPE being detected is very small. In this limit, the binomial distribution approaches a Poisson distribution.

[2]To evaluate V_n in the case where $p = 0$, we define $V_n(\lambda, 0) \equiv \lim_{p\to 0} V_n(\lambda, p) = \frac{\lambda^n}{n!} e^{-\lambda}$, which is a Poisson distribution, as we would expect.

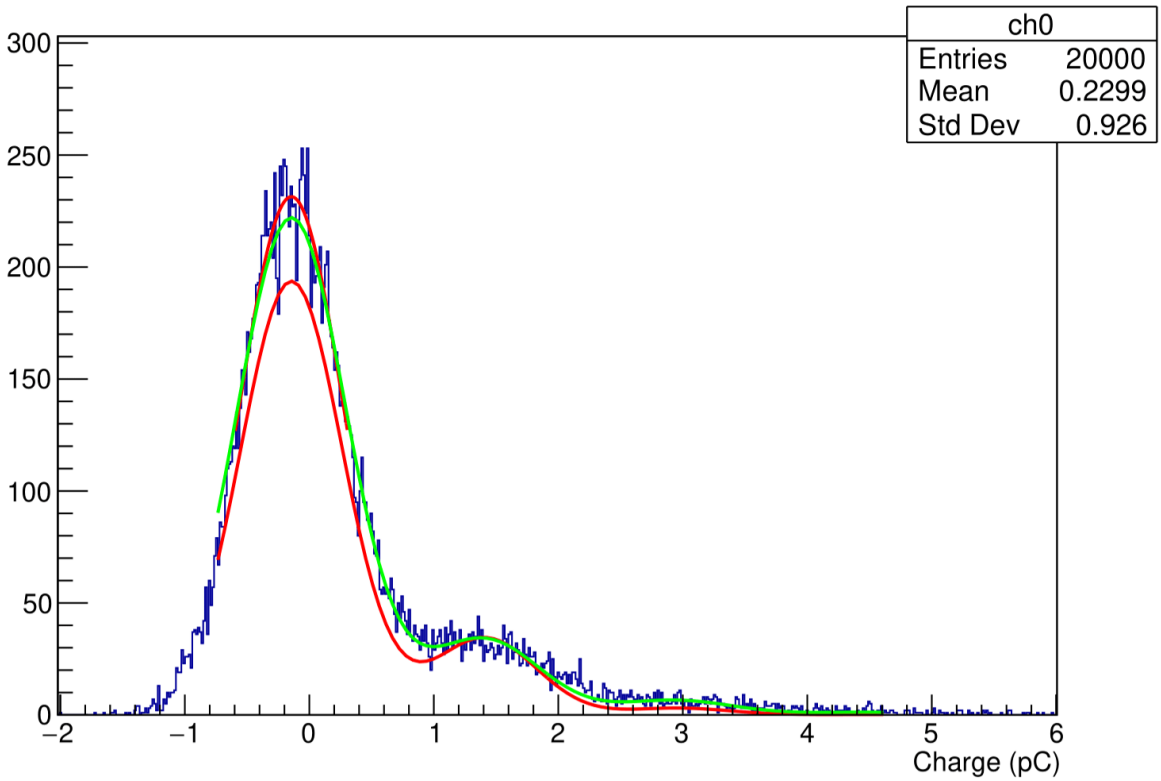


Figure 6.1: SPE charge histogram without discernible peaks that was able to be fit with my implementation of the fitting algorithm. Preliminary fit curves in red, final fit curve in green. Data acquired with the 3-module and 8TIA boards using the CAEN digitizer.

If we assume that a given region of integration contains an integer number of SPEs, then the combined SPE charge distribution for all n is the sum of all $G_x(n\mu, \sigma\sqrt{n})$, each weighted by the $V_n(\lambda, p)$.

$$\sum_{i=0}^{\infty} V_i(\lambda, p) \cdot G_x(i\mu, \sigma\sqrt{i}) \tag{6.3}$$

Finally, every waveform includes noise, mainly from the digitizer. We assume that this noise is centered at 0 V, and its charge distribution is Gaussian with a standard deviation of Ω. Our complete model is the convolution of the noise distribution with the SPE charge distribution.

$$G_x(0, \Omega) * \sum_{i=0}^{\infty} V_i(\lambda, p) \cdot G_x(i\mu, \sigma\sqrt{i}) \tag{6.4}$$

where '$*$' denotes the convolution operator. Since convolutions are linear, we have

$$\sum_{i=0}^{\infty} V_i(\lambda, p) \cdot G_x(i\mu, \sqrt{\Omega^2 + i\sigma^2}) \tag{6.5}$$

Finally, we add a simple horizontal offset parameter h, and vertical scale parameter s, making our final model:

$$M_x(s, h, \lambda, \mu, \Omega, \sigma, p) = s \cdot \sum_{i=0}^{\infty} V_i(\lambda, p) \cdot G_{x-h}(i\mu, \sqrt{\Omega^2 + i\sigma^2}) \tag{6.6}$$

where M_x is the probability density of measuring a charge x from an SPE waveform. In reality, we truncate this sum after at most 20 terms since $V_i(\lambda, p)$ diminishes quickly as a function of i.

This model is implemented in `https://github.com/alatorre-caltech/dt5742/blob/master/python/btl/fit_511_funcs.py`, and uses CERN's data analysis library ROOT to fit the model to any given SPE charge histogram. However, with seven parameters, ROOT is unable to fit this model accurately without some of the parameters starting off close enough to their true values. Therefore, we estimate the values of several of the parameters using a generalized algorithm before performing a fit with ROOT. The process is described below.

1. p is temporarily fixed to zero. This effectively makes V_n a Poisson distribution.

2. Try to find the offset of the zero peak by looking for the peak in the charge distribution less than 0.

3. The histogram gets fit with a Gaussian curve initialized with the mean equal to the peak found in the previous step. This fit provides a good estimate of Ω.

4. The location and standard deviation of the noise peak is used to estimate λ. Approximately 95% of all the events that detected no SPE lie within $h \pm 2\Omega$. Taking all the histogram entries in this range and dividing by the total number of entries gives V_0. With our assumption that $p \approx 0$, we can solve for λ.

5. Using a mixture of Gaussians method [4], the previous estimate give us enough information to also estimate μ. For any normalized linear combination of Gaussian distributions:

$$S = \sum_{i=0}^{k} \alpha_i G_x(\mu_i, \sigma_i) \tag{6.7}$$

where $\sum_{i=0}^{k} \alpha_i = 1$, the mean of S is equivalent to $\sum_{i=0}^{k} \alpha_i \mu_i$. In our situation, $\alpha_i = V_i(\lambda, p)$, which we can already estimate for all i. Furthermore, $\mu_i = i\mu + h$ (note that $G_{x-h}(\mu, \sigma) = G_x(\mu + h, \sigma)$). Thus

$$\text{Total mean of the SPE histogram} = \sum_{i=0}^{k} V_i(\lambda, p)(i\mu + h) \tag{6.8}$$

where k is an arbitrary integer we use to set the degree of precision. In Equation 6.8, μ is only parameter which does not have an estimate yet, allowing us to solve for it.

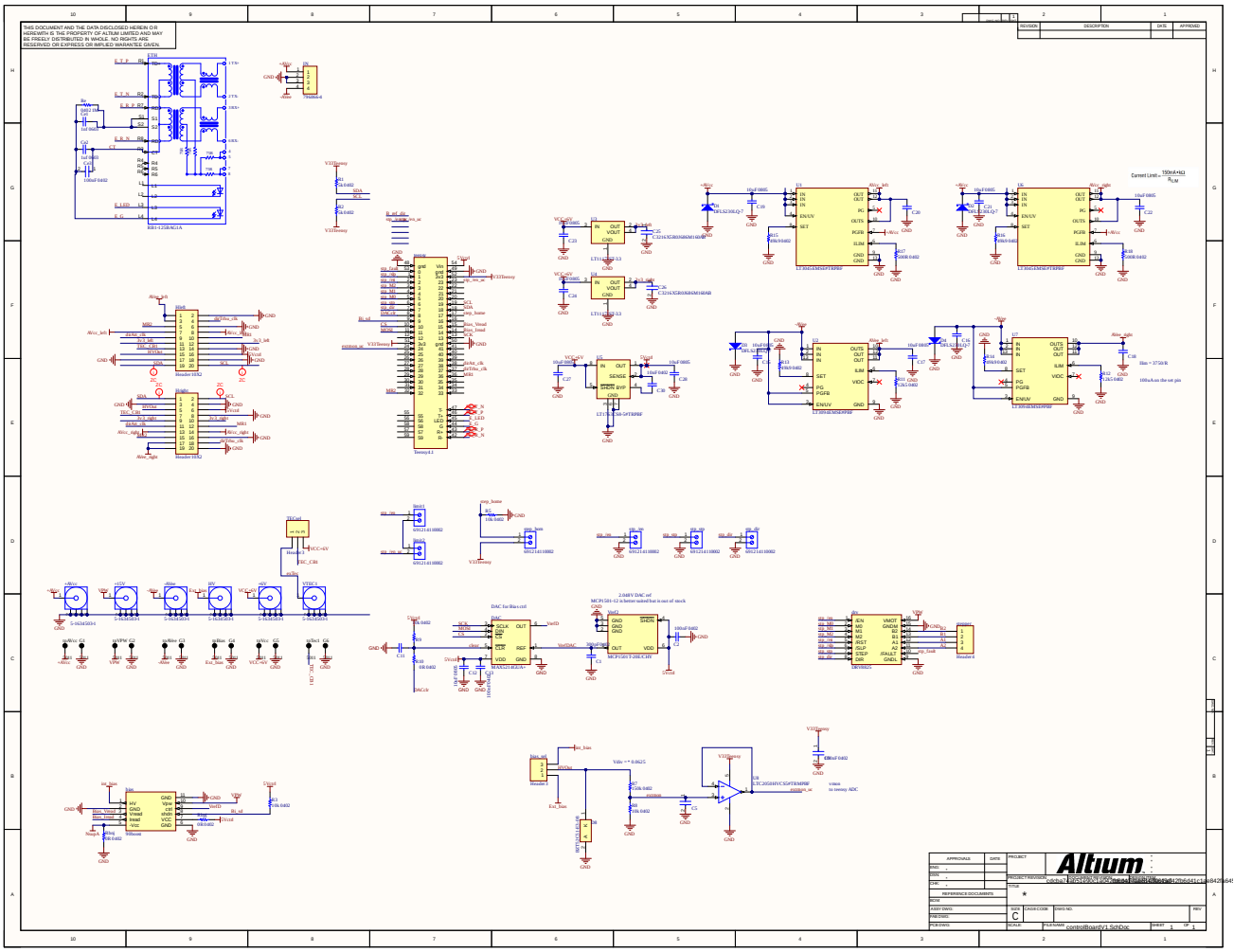
6. The only parameter that has yet to be estimated is σ, which we initialize to zero. For one, we hypothesize that it is much less significant than Ω, and two, this encourages ROOT's fit algorithm to maintain discernible peaks in the model.

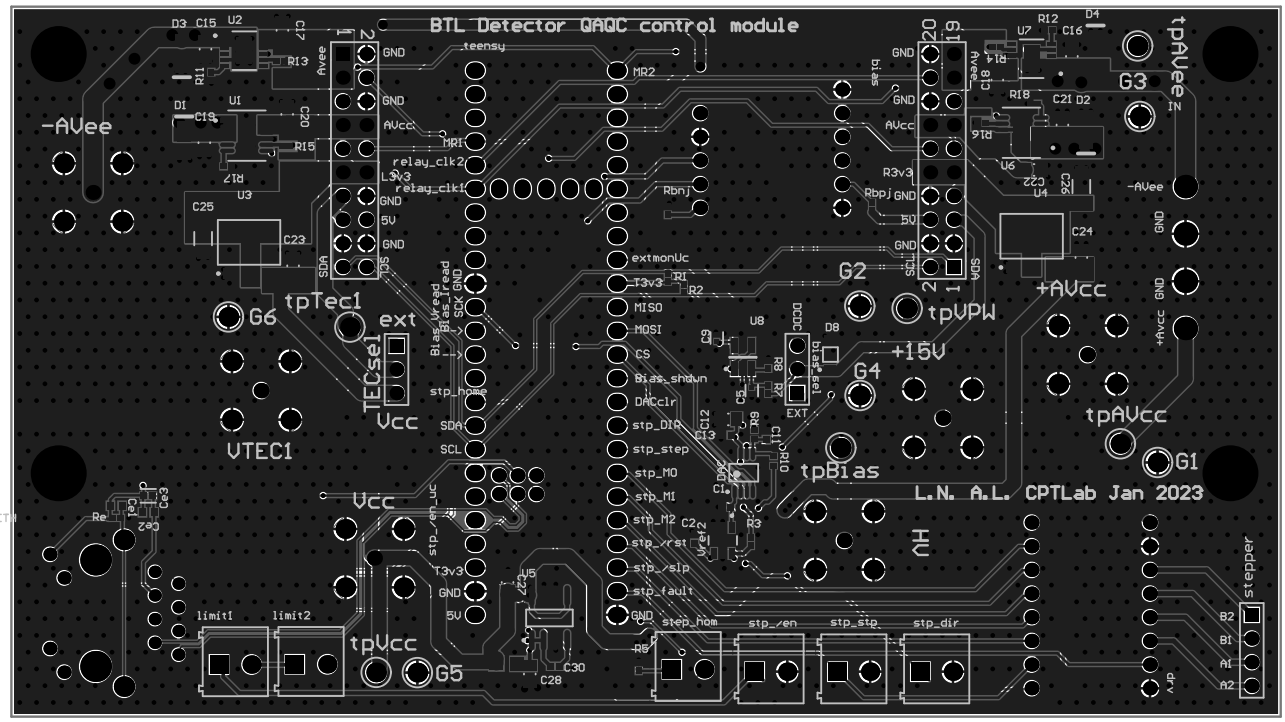
After all the parameters have been estimated, ROOT performs two fits of the model. The first leaves all parameters fixed except λ and μ since these are the parameters that have the greatest influence on the relative height and positioning of the peaks, and are most important to have as accurate as possible. The second fit unfixes all parameters except h as a final, all-encompassing, estimation of all the parameters. h is left fixed since there is no mathematical discrepancy between it and the mean of the fitted Gaussian in step 3. The charge of an SPE is recorded as μ after the final fit is performed.

Chapter 7

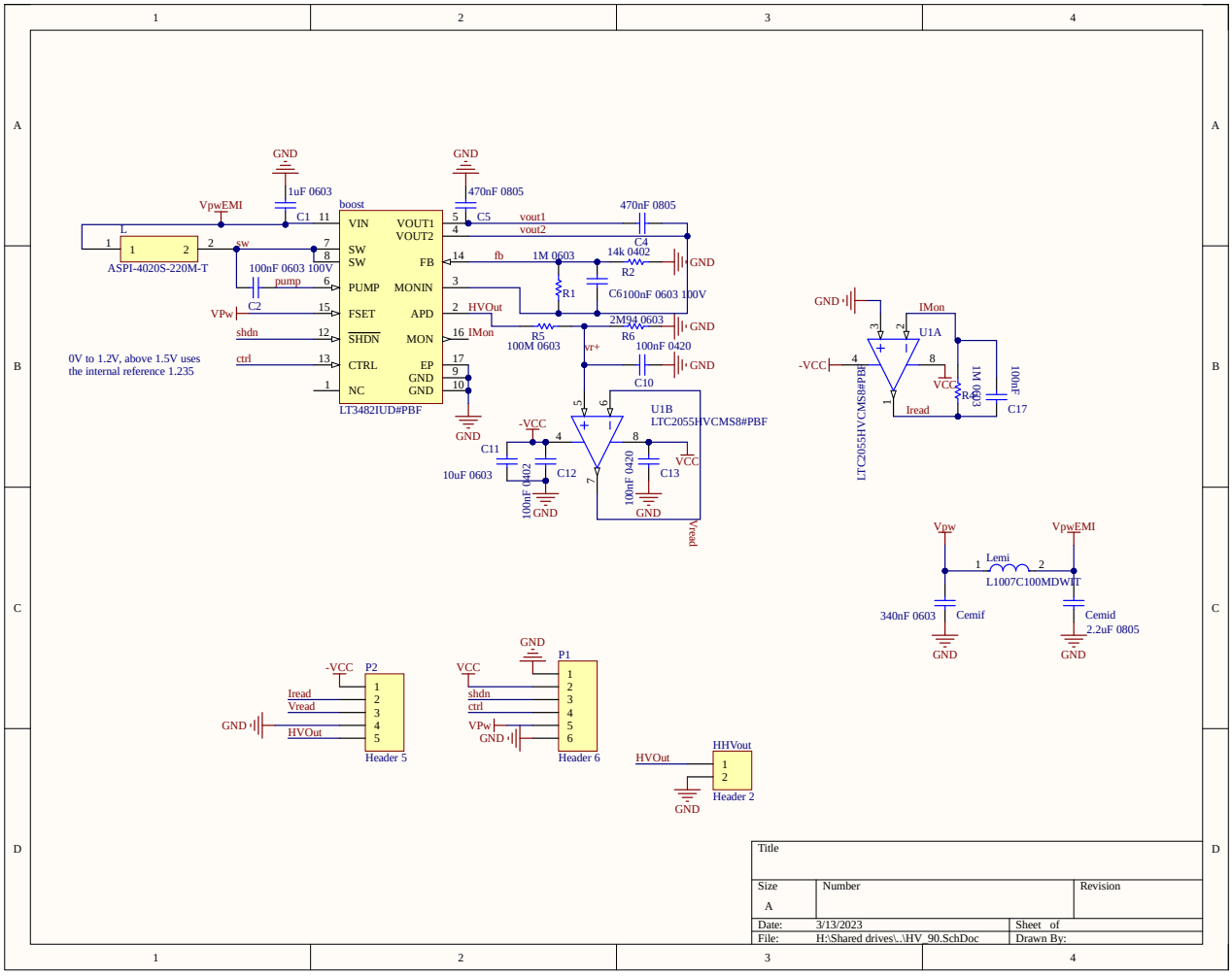
Schematics

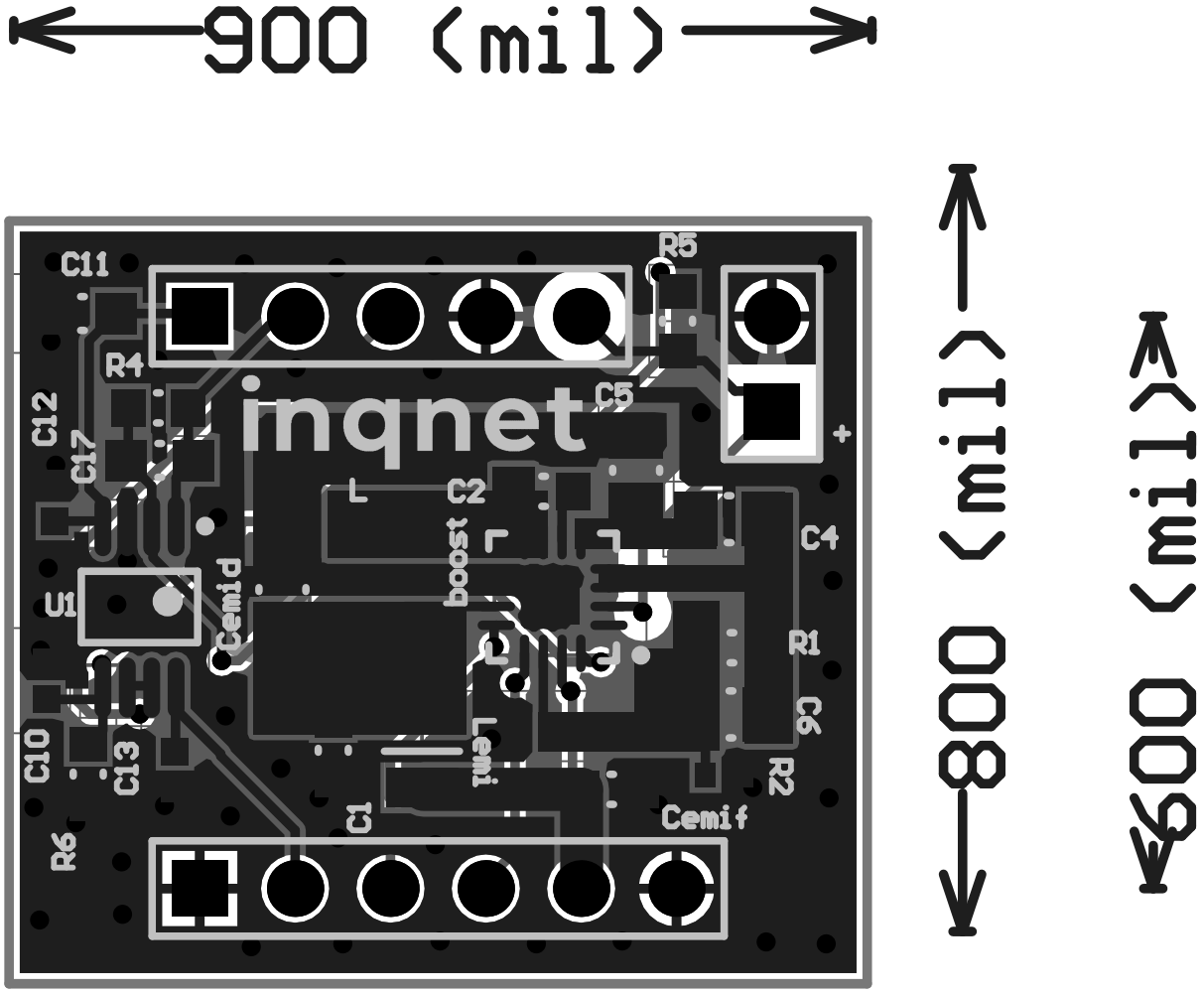
7.1 Control Board



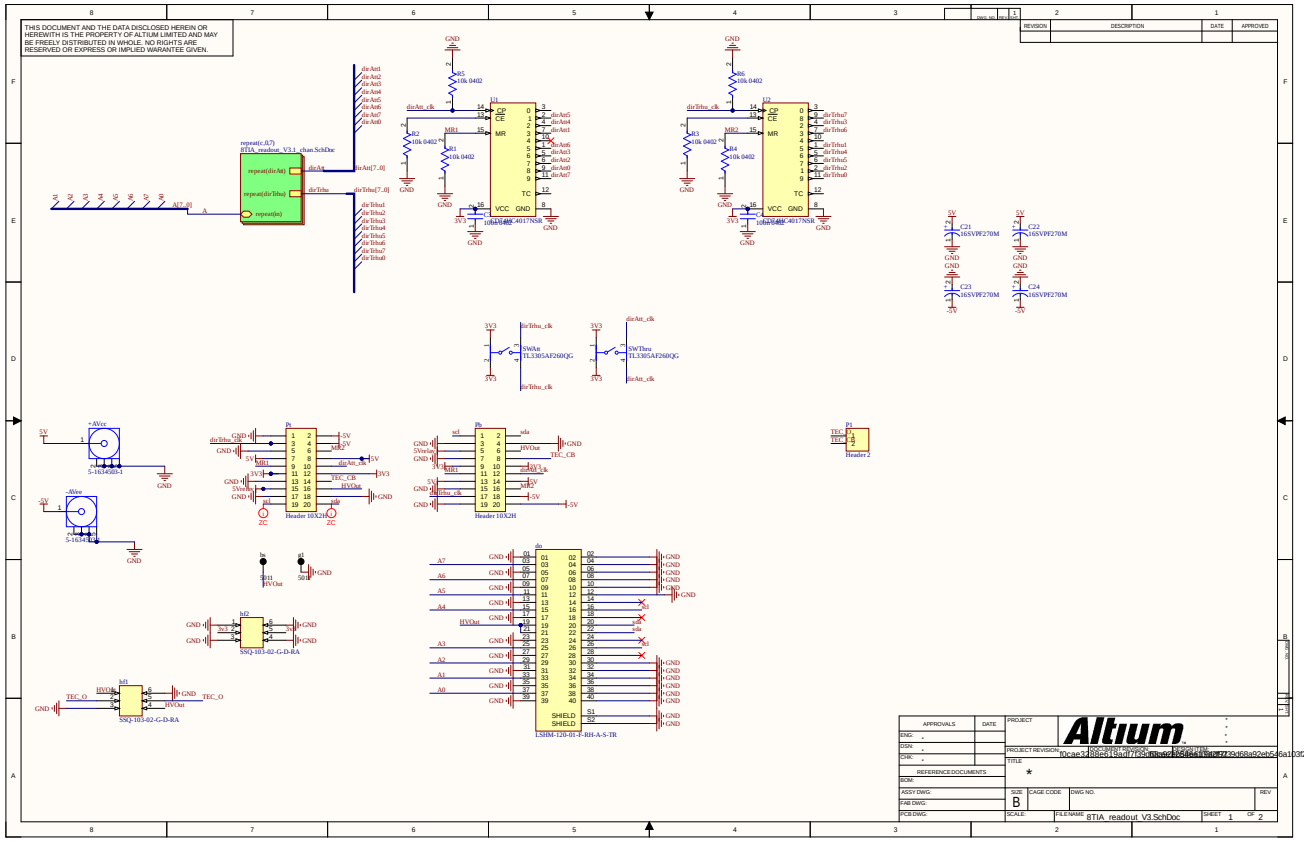

BTL Detector QAQC control module
-AVee
tpTec1
G6
ext
TECsel
Vcc
VTEC1
tpVcc
G5
limit1
limit2
teensy
relay_clk2
relay_clk1
L3v3
GND
5V
SDA
SCL
Bias_Vread
Bias_Iread
SCK
stp_home
stp_en_uC
MR2
MR1
Rbnj
extmonUc
T3v3
MISO
MOSI
CS
Bias_shdwn
DACclr
stp_DIR
stp_step
stp_M0
stp_M1
stp_M2
stp_/rst
stp_/slp
stp_fault
step_hom
stp_/en
stp_stp
stp_dir
bias
AVcc
R3v3
Rbpj
G2
tpVPW
+15V
G4
tpBias
EXT
+AVcc
tpAVcc
G1
G3
tpAVee
IN
L.N. A.L. CPTLab Jan 2023
HV
Stepper
B2
B1
A1
A2

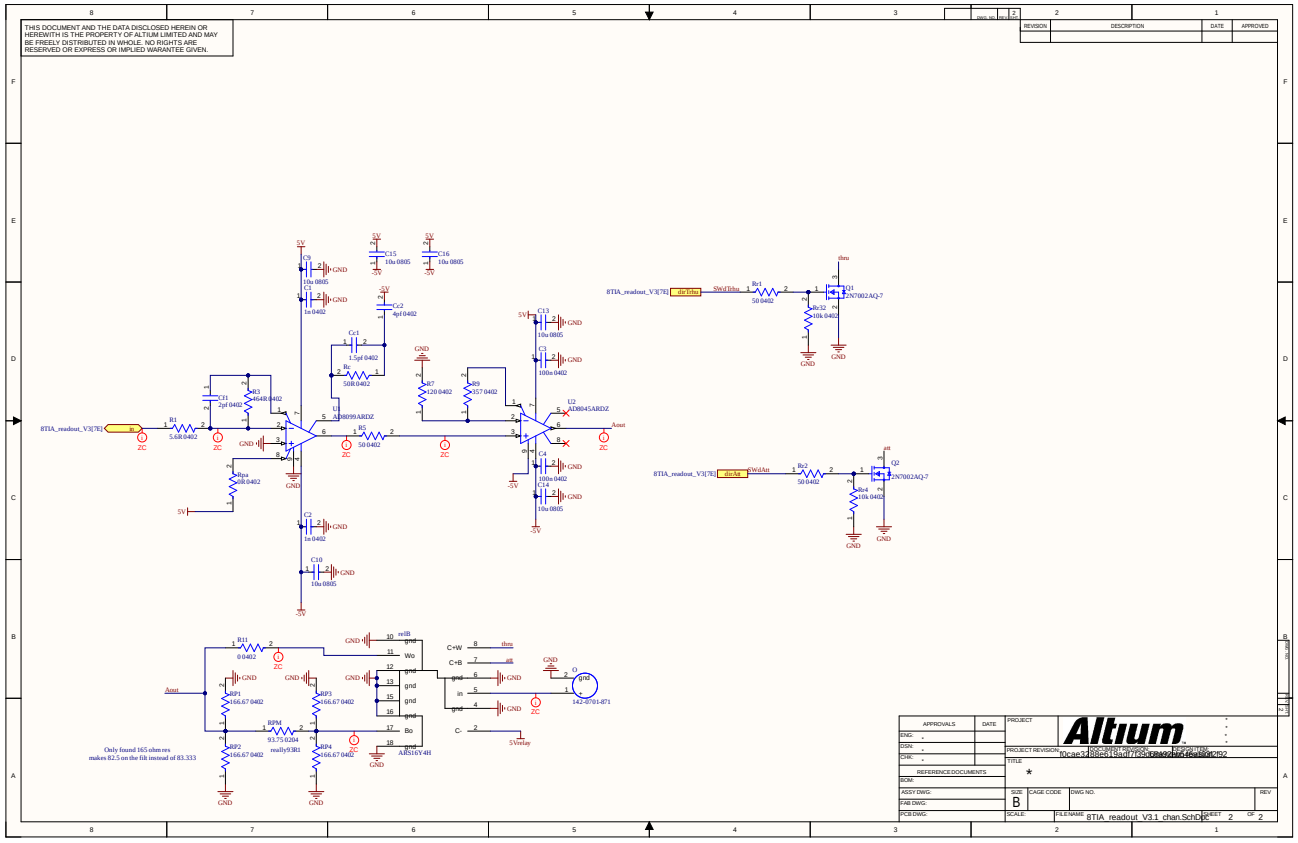
7.2 HV Elevator

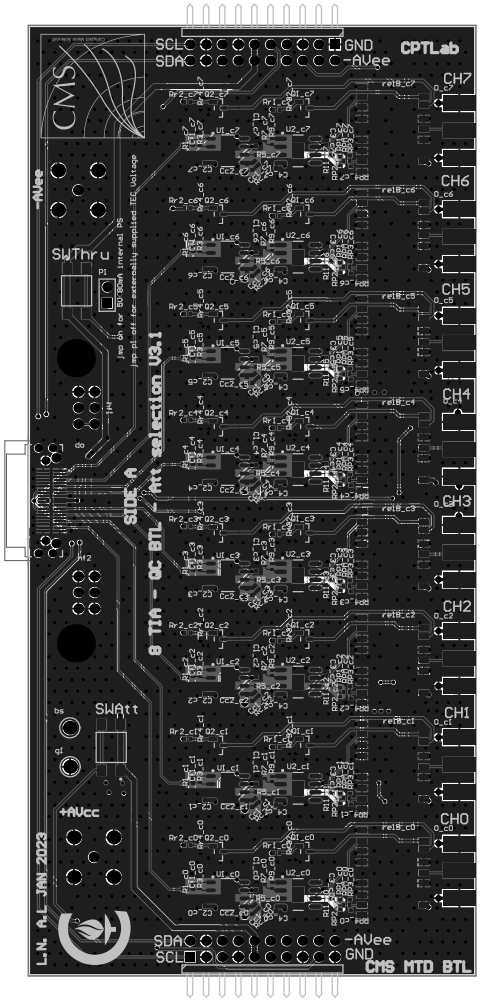



900 (mil)
800 (mil)
600 (mil)
inqnet
C11
R5
R4
C5
C12
C17
C2
C4
U1
Cemid
boost
R1
C3
C10
C13
Lemi
R2
C1
Cemif
R6

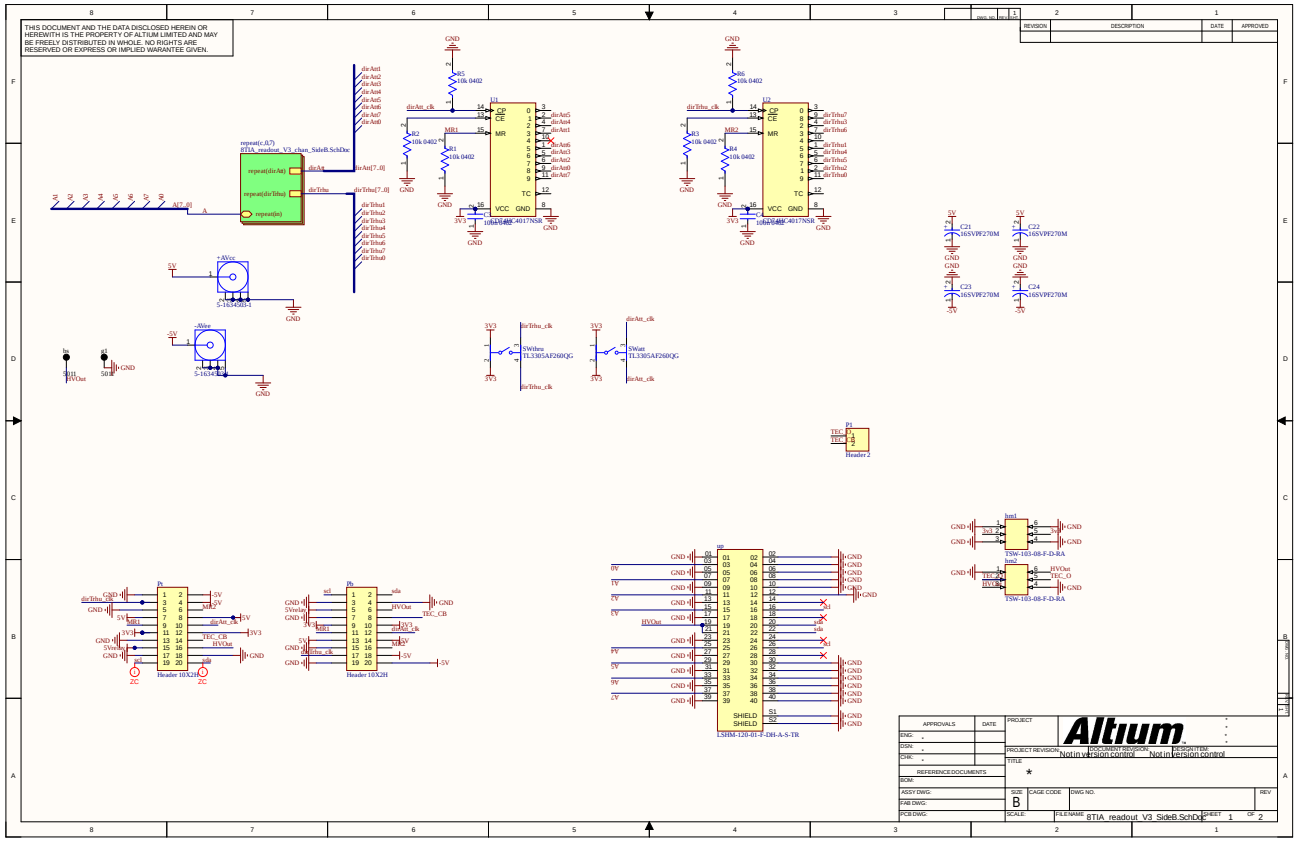
7.3 Amplifier Board

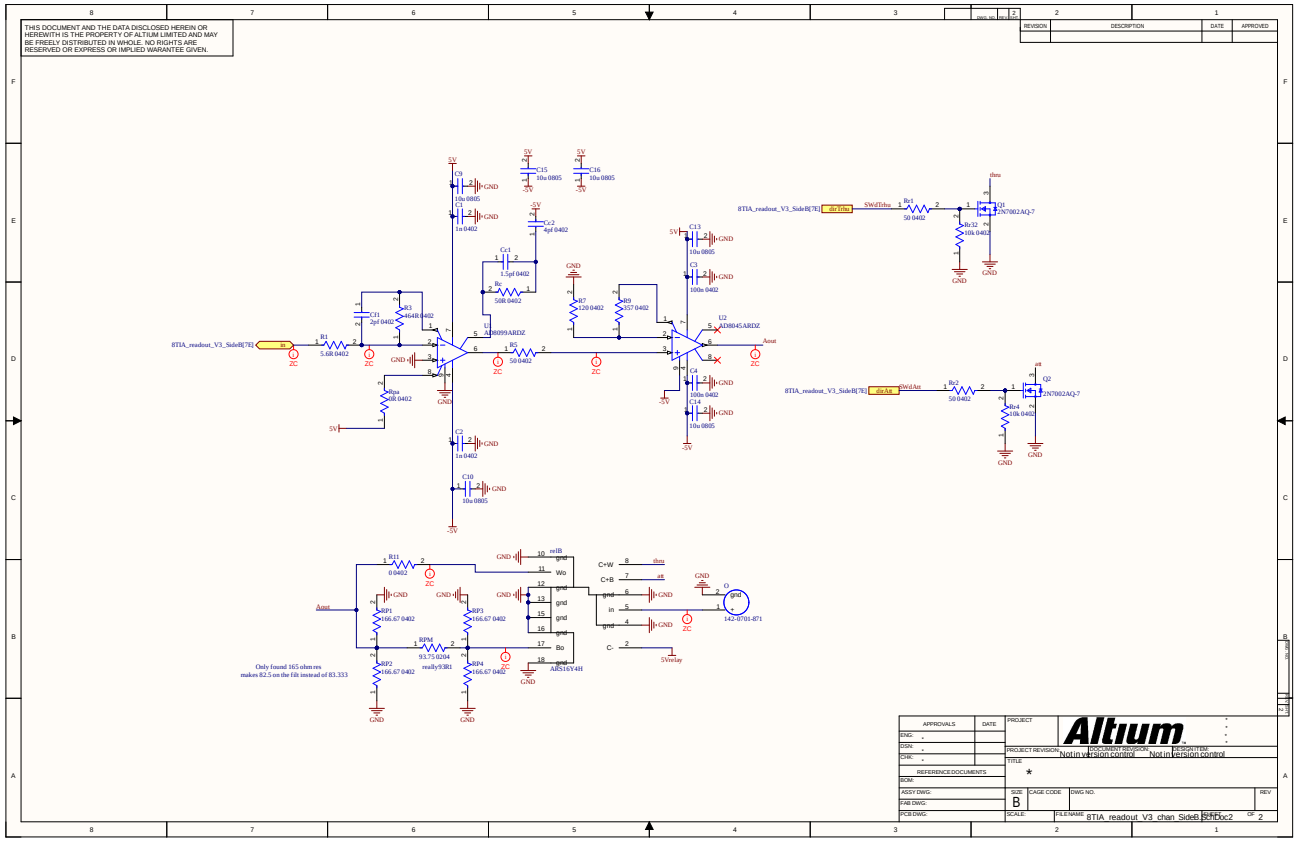


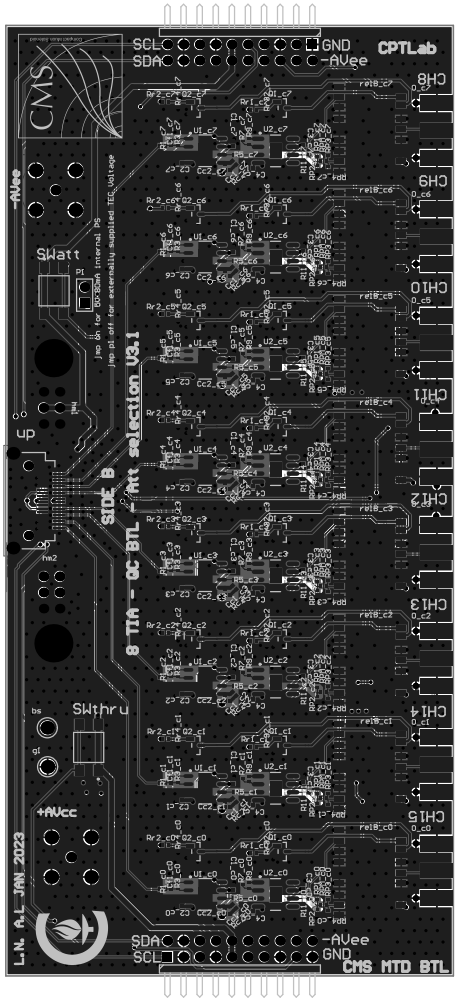

THIS DOCUMENT AND THE DATA DISCLOSED HEREIN OR HEREWITH IS THE PROPERTY OF ALTIUM LIMITED AND MAY BE FREELY DISTRIBUTED IN WHOLE. NO RIGHTS ARE RESERVED OR EXPRESS OR IMPLIED WARANTEE GIVEN.
REVISION
DESCRIPTION
DATE
APPROVED
8TIA_readout_V3[78]
Ain
Aout
AD8099ARDZ
AD8045ARDZ
2N7002AQ-7
thru
dirThru
SWdThru
dirAnt
SWdAnt
Only found 165 ohm res makes 82.5 on the filt instead of 83.333
APPROVALS
DATE
PROJECT
Altium
REFERENCE DOCUMENTS
TITLE
ASSY DWG:
FAB DWG:
PCB DWG:
SIZE
B
CAGE CODE
DWG NO.
REV
SCALE:
FILENAME
8TIA_readout_V3.1 chan.SchDoc
SHEET
2
OF
2


SCL
SDA
GND
-AVee
CPTLab
CH7
CH6
CH5
CH4
CH3
CH2
CH1
CH0
SWThru
SWAtt
+AVcc
-AVee
SIDE A
8 TIA - QC BTL - Att selection V3.1
L.N. A.L. JAN 2023
CMS MTD BTL

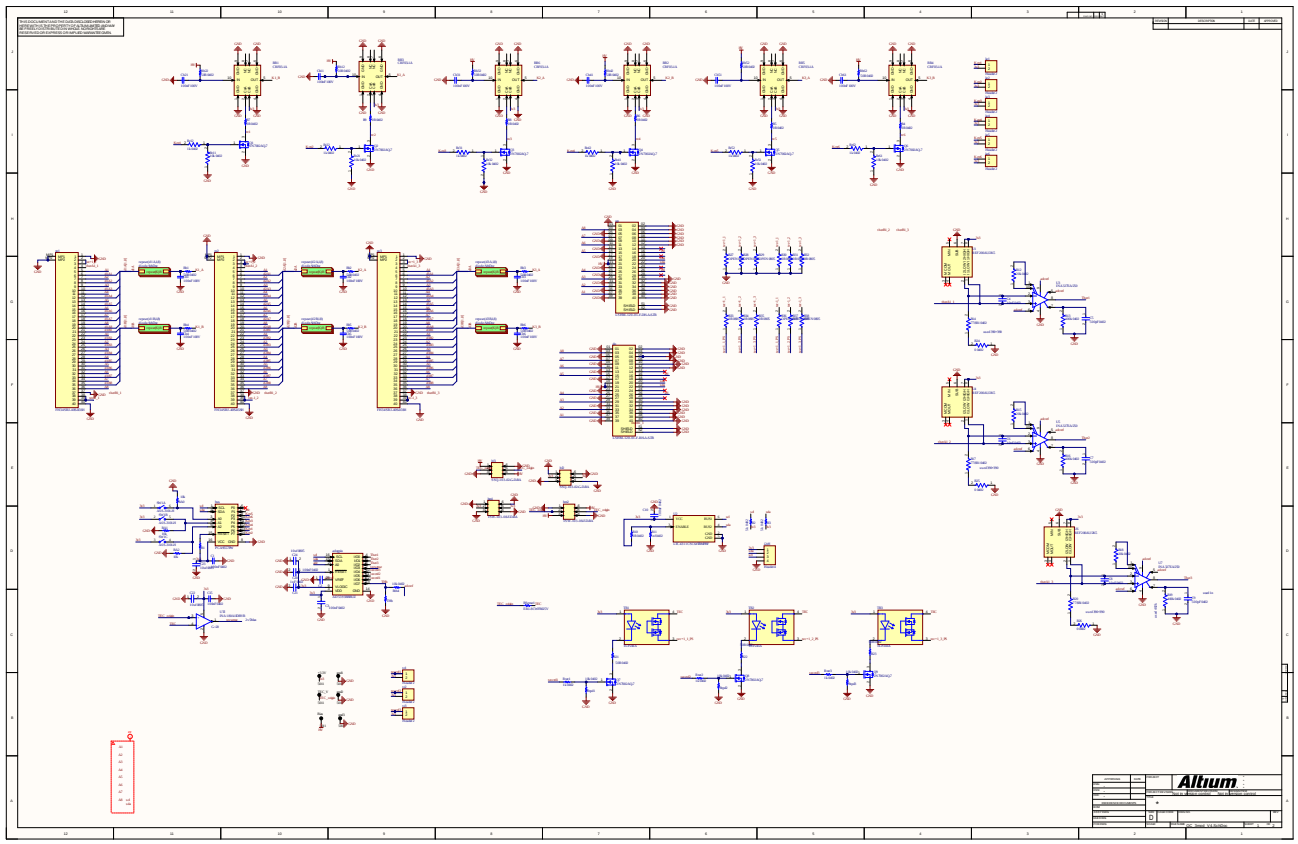
7.4 Amplifier Board (side B)

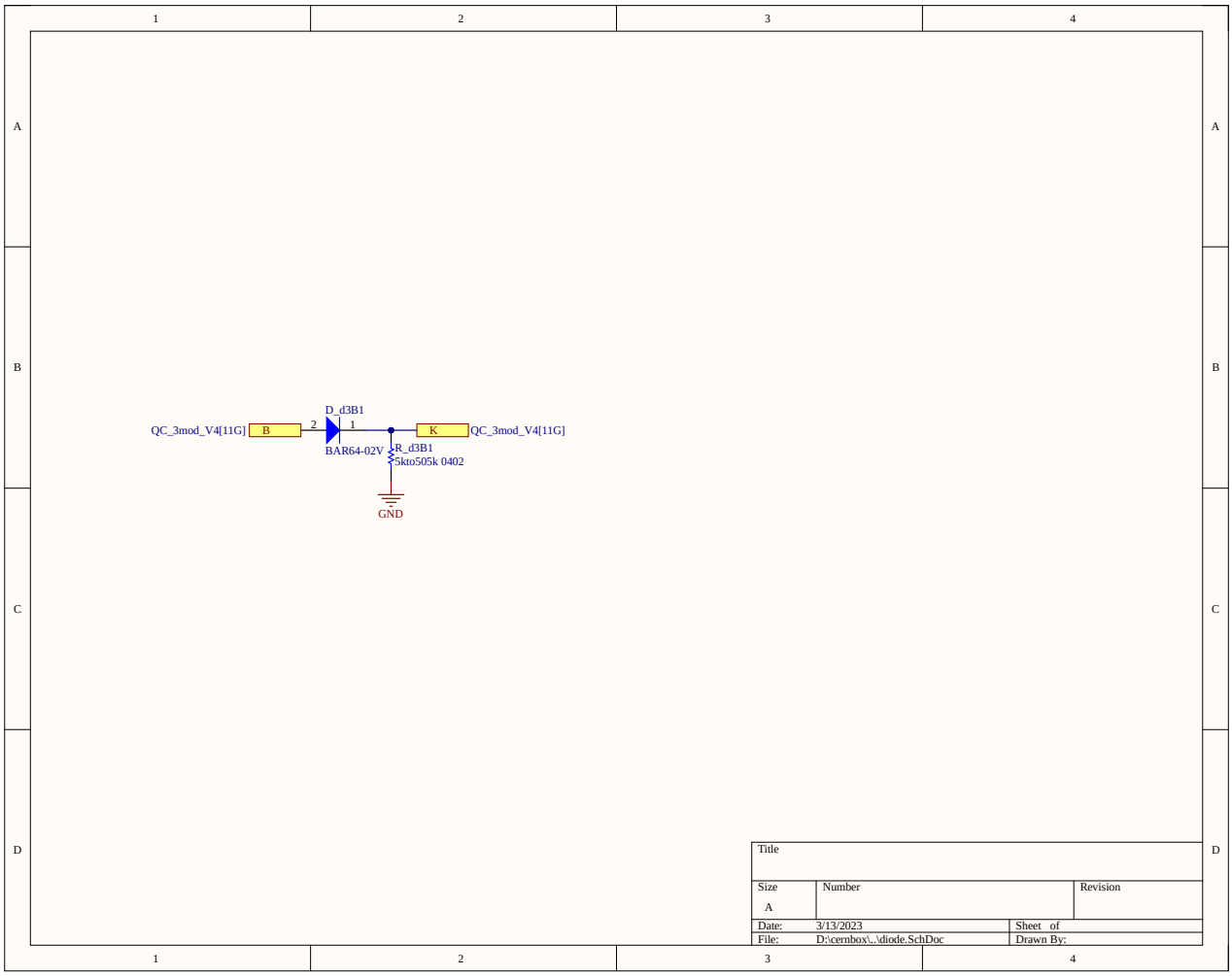


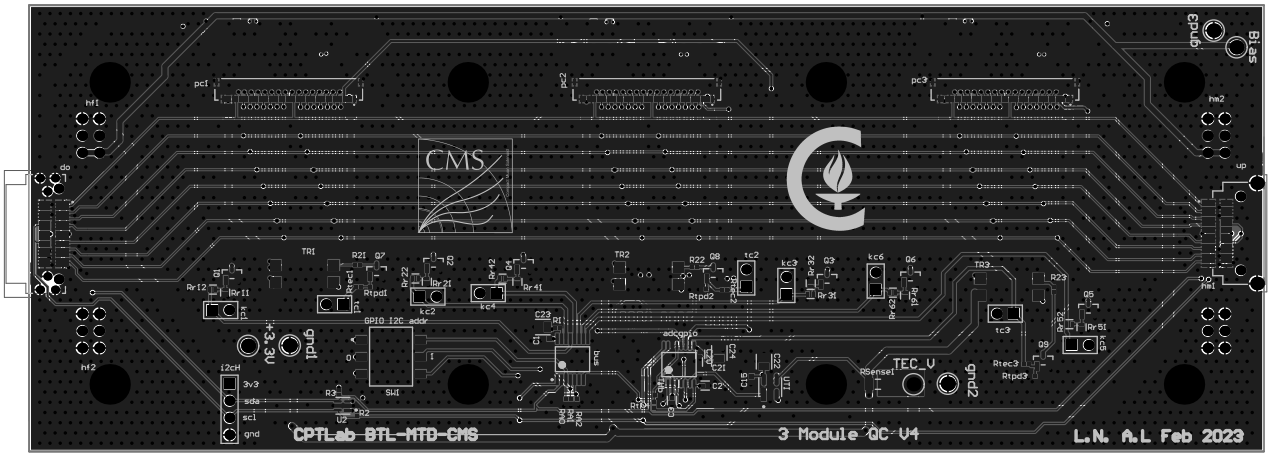

THIS DOCUMENT AND THE DATA DISCLOSED HEREIN OR HEREWITH IS THE PROPERTY OF ALTIUM LIMITED AND MAY BE FREELY DISTRIBUTED IN WHOLE. NO RIGHTS ARE RESERVED OR EXPRESS OR IMPLIED WARANTEE GIVEN.
REVISION
DESCRIPTION
DATE
APPROVED
8TIA_readout_V3_SideB[7:0]
U1 AD8099ARDZ
U2 AD8045ARDZ
Rf 50 0402
Rz 50R 0402
R1 5.6R 0402
R3 464R 0402
Cf1 2pf 0402
Cz1 1.5pf 0402
Cz2 4pf 0402
R7 120 0402
R9 357 0402
Rbia 50R 0402
C4 100n 0402
C13 10u 0805
C14 10u 0805
C15 10u 0805
C16 10u 0805
C9 10u 0805
C8 1u 0402
C5 1u 0402
C10 10u 0805
C3 100n 0402
5V
-5V
GND
Aout
dirTrbu
dirAn
SWdTrbu
SWdAn
Rr1 50 0402
Rr32 10k 0402
Rr2 50 0402
Rr4 10k 0402
Q1 2N7002AQ-7
Q2 2N7002AQ-7
thru
att
RA1 0 0402
RP1 166.67 0402
RP2 166.67 0402
RP3 166.67 0402
RP4 166.67 0402
RPM 93.75 0204
Only found 165 ohm res
makes 82.5 on the filt instead of 83.333
really 9.9R1
relB
Wo
Bo
AHS16Y4H
C+W
C+B
C-
in
5Vrelay
142-0701-871
APPROVALS
DATE
PROJECT
ENG:
DSN:
CHK:
REFERENCE DOCUMENTS
BOM:
ASSY DWG:
FAB DWG:
PCB DWG:
Altium
PROJECT REVISION:
Not in version control
CONFIGITEM:
Not in version control
TITLE
*
SIZE
B
CAGE CODE
DWG NO.
REV
SCALE:
FILENAME
8TIA_readout_V3_chan_SideB_SCHDoc2
SHEET
2
OF
2


SCL
SDA
GND
-AVee
CPTLab
CMS
-AVee
SWatt
Jmp sh for 50 Ohm internal PD
Jmp pl off for externally supplied TEC voltage
up
SIDE_B
8 TIA - QC BTL
Att selection V3.1
SWthru
+AVcc
L.N. A.L JAN 2023
SDA
SCL
-AVee
GND
CMS MTD BTL
CH8
CH9
CH10
CH11
CH12
CH13
CH14
CH15

7.5 3 Module Board




1
2
3
4
A
B
C
D
QC_3mod_V4[11G]
B
2
D_d3B1
1
K
QC_3mod_V4[11G]
BAR64-02V
R_d3B1
5kto505k 0402
GND
Title
Size
A
Number
Revision
Date:
3/13/2023
Sheet of
File:
D:\cernbox\..\diode.SchDoc
Drawn By:


gnd3
Bias
pc1
pc2
pc3
hf1
hm2
CMS
up
TR1
TR2
TR3
R21
Q7
Rtpd1
kc2
kc4
tc2
kc3
R22
Q8
Rtpd2
Q3
kc6
Q6
tc3
R23
Q5
Q9
Rtpd3
GPIO I2C addr
+3.3v
gnd1
adcgpio
TEC_V
RSense1
gnd2
i2cH
3v3
sda
scl
gnd
SW1
U2
hf2
CPTLab BTL-MTD-CMS
3 Module QC V4
L.N. A.L Feb 2023

7.6 Teensy

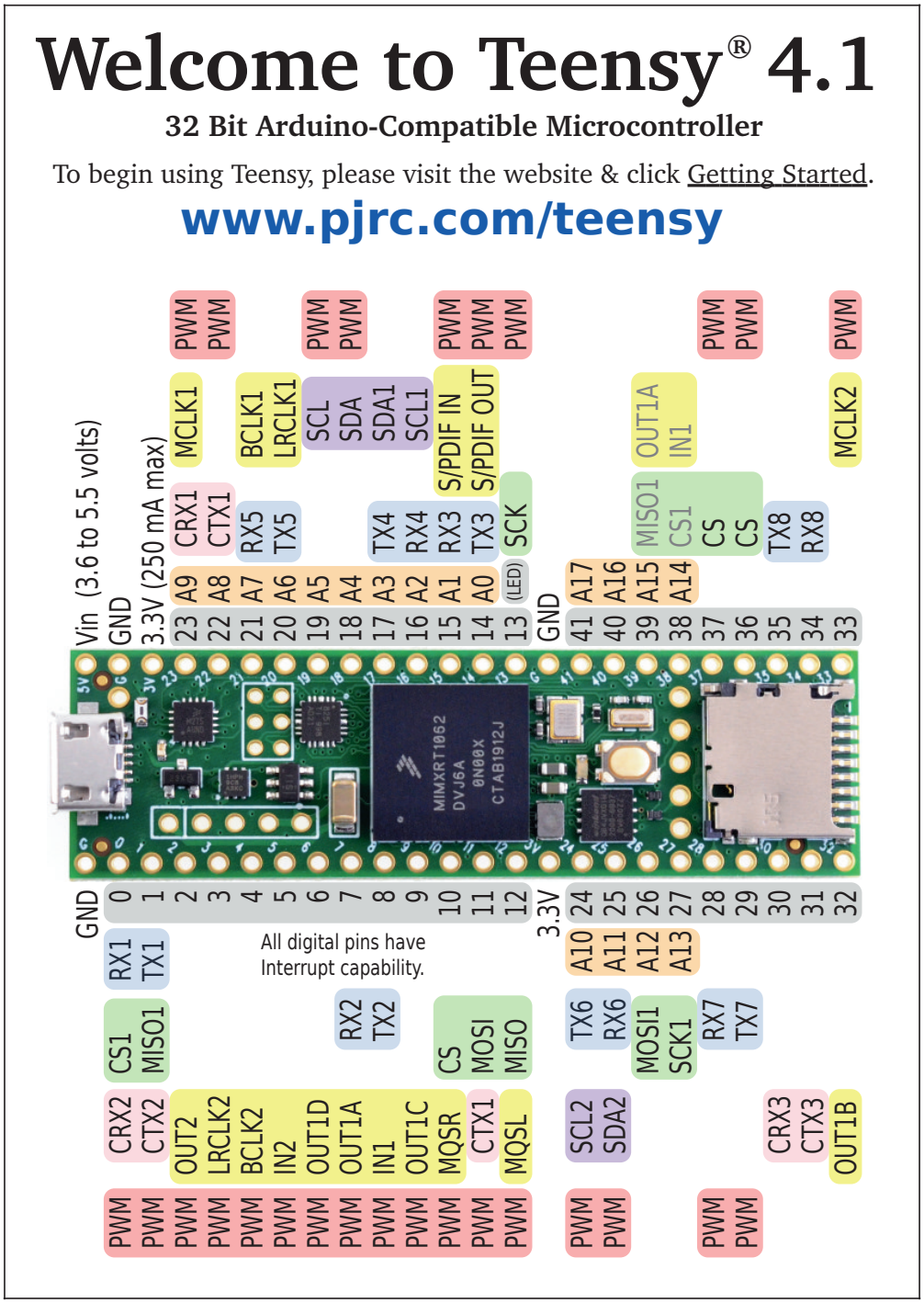


Figure 7.1: Teensy 4.1 pinout

Bibliography

[1] H. Alva-Sánchez et al. "Understanding the intrinsic radioactivity energy spectrum from 176Lu in LYSO/LSO scintillation crystals". In: (2018). DOI: 10.1038/s41598-018-35684-x.

[2] P. Agnes et al. "Performance of the ReD TPC, a novel double-phase LAr detector with silicon photomultiplier readout". In: *The European Physical Journal C* 81.11 (Nov. 2021). DOI: 10.1140/epjc/s10052-021-09801-6. URL: https://doi.org/10.1140%2Fepjc%2Fs10052-021-09801-6.

[3] S. Vinogradov et al. "Probability distribution and noise factor of solid state photomultiplier signals with cross-talk and afterpulsing". In: *2009 IEEE Nuclear Science Symposium Conference Record (NSS/MIC)*. 2009, pp. 1496–1500. DOI: 10.1109/NSSMIC.2009.5402300.

[4] Minus One-Twelfth (https://math.stackexchange.com/users/643882/minus-one-twelfth). *Calculating the mean and standard deviation of a Gaussian mixture model of two curves*. Mathematics Stack Exchange. URL: https://math.stackexchange.com/q/3689169.

Appendix A

RTD Readout Calculations

This document is designed to show the calculations and schematic for the readout of the PT1000 RTDs on the SiPM packages.

A.1 Requirements

1. Be able to calculate the temperature in the range -50 C - 100 C

2. Be able to distinguish temperatures around room temperature with a precision of 0.1 degree C

The output voltage will be read by the AD5593R (datasheet here: `https://www.analog.com/media/en/technical-documentation/data-sheets/ad5593r.pdf`) which has a 12 bit ADC. The resolution should theoretically be:

$$\delta = \frac{2.5\text{V}}{2^{12}} \approx 1\text{mV} \tag{A.1}$$

A.2 Schematic

Figure A.1 shows a schematic of the RTD readout scheme. This scheme is suggested at `https://www.ti.com/lit/ug/tidu969/tidu969.pdf`. The amplifier used is the INA326 whose datasheet can be found at `https://www.ti.com/lit/ds/symlink/ina326.pdf`, and the two current sources are provided by the REF200 which can be found here: `https://www.ti.com/lit/ds/symlink/ref200.pdf`.

Note that right now the two current sources are powered by the 3.3 V rail and the op amp is powered by the 2.5 V voltage reference provided by the AD5593R. I don't think either of these is critical. I chose to use the 2.5 V reference just because it's conveniently close and the 3.3 V rail is easier to access so the REF200 chips can be placed more freely on the board.

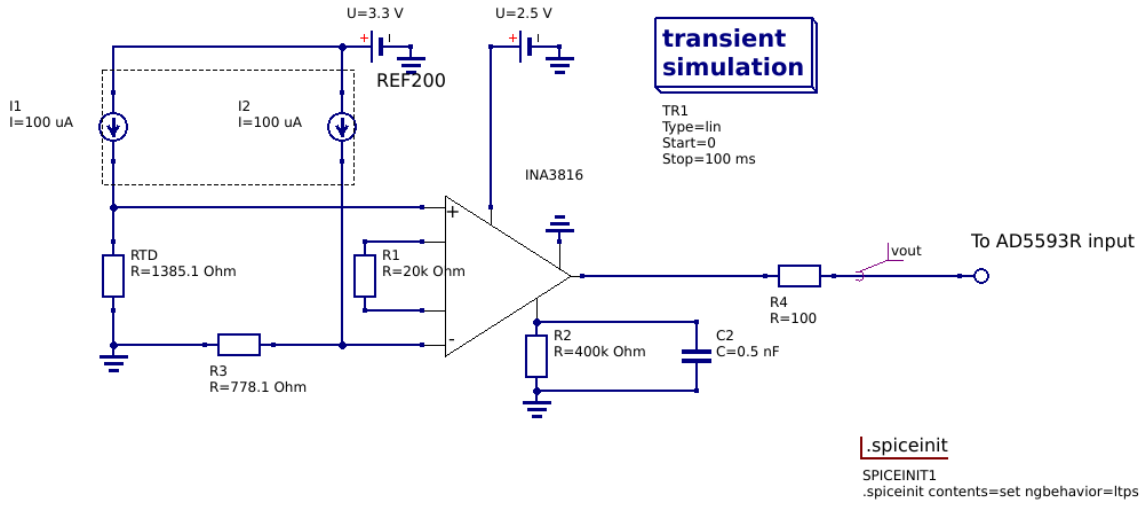


Figure A.1: Schematic of RTD readout scheme

A.3 Calculation

The PT1000 RTD should have the following resistances at the extreme temperatures we want:

$$\mathrm{R}(-50\mathrm{C}) = 803.1\Omega \tag{A.2}$$

$$\mathrm{R}(100\mathrm{C}) = 1385.1\Omega \tag{A.3}$$

Next, we can calculate the required gain:

$$G = \frac{\mathrm{V_{out,max}} - \mathrm{V_{in,max}}}{(\mathrm{RTD_{max}} - \mathrm{RTD_{min}})\, I_{\mathrm{ref}}} = \frac{2.5\mathrm{V} - 0.1\mathrm{V}}{(1385.1\Omega - 803.1\Omega)\, 100\mu\mathrm{A}} = 41.23\mathrm{V/V} \tag{A.4}$$

From Table 2 in `https://www.ti.com/lit/ug/tidu969/tidu969.pdf` we see a standard value for R1 and C2 is 20k and 0.5 nF. We can calculate R2 as:

$$\mathrm{R2} = \frac{\mathrm{GR1}}{2} = \frac{41.2320\mathrm{k}}{2} = 412.3\mathrm{k} \tag{A.5}$$

and we can just choose a standard value of 400k.

So our actual gain will be

$$\mathrm{G} = 2\frac{\mathrm{R2}}{\mathrm{R1}} = \frac{400\mathrm{k}}{20\mathrm{k}} = 40 \tag{A.6}$$

Finally, we can calculate the value of R3:

$$\mathrm{R3} = \frac{\mathrm{G} \cdot \mathrm{I_{ref}} \cdot \mathrm{RTD_{min}} - \mathrm{V_{out,min}}}{\mathrm{GI_{ref}}} = \frac{40 \cdot 100\mu\mathrm{A} \cdot 803.1\Omega - 0.1\mathrm{V}}{40 \cdot 100\mu\mathrm{A}} = 778.1\Omega \tag{A.7}$$

Now, we can check if we are able to actually distinguish 0.1 degrees C.

$$\mathrm{R(20C)} = 1077.9\Omega \tag{A.8}$$
$$\mathrm{R(20.1C)} = 1078.3\Omega \tag{A.9}$$

The voltage across the RTD is

$$\mathrm{V_{RTD}} = \mathrm{I_{ref}} \cdot \mathrm{R_{RTD}} \tag{A.10}$$

while the voltage across R3 is

$$\mathrm{V_3} = \mathrm{I_{ref}} \cdot \mathrm{R_3} \tag{A.11}$$

Therefore the output voltage is

$$\mathrm{V_{out}} = \mathrm{G} \cdot (\mathrm{V_{RTD}} - \mathrm{V_3}) = \mathrm{G} \cdot \mathrm{I_{ref}} \cdot (\mathrm{R_{RTD}} - \mathrm{R_3}) \tag{A.12}$$

So the difference in voltages will be

$$\mathrm{V_{out}(20C)} = 40 \cdot 100\mu\mathrm{A} \cdot (1077.9 - 778.1) = 1.1992 \tag{A.13}$$
$$\mathrm{V_{out}(20.1C)} = 40 \cdot 100\mu\mathrm{A} \cdot (1078.3 - 778.1) = 1.2008 \tag{A.14}$$

The difference is about 1 mV which is just barely within our ability to resolve.

A.4 Notes

As mentioned in the guide at `https://www.ti.com/lit/ug/tidu969/tidu969.pdf`, all components should be 1% thin film resistors and X7R ceramic capacitors.