University of Bergen

Department of Physics and Technology

A Setup for Testing of Silicon Pixel Modules for the ITk Tracker in the ATLAS Experiment at CERN

Master Thesis in Physics
by
Wai Chun Leung

June 14, 2021

Abstract

New pixel sensors will be installed to the new Inner Tracker upgrade at the ATLAS detector. These sensor will have higher radiation tolerance than the ones currently on the Inner Detector.

This thesis will concentrate on development of a test setup dedicated to testing the new pixel sensors. The Peltier Controller Temperature Monitor has been developed for this purpose. IV-curve measurements have also been done on RD53A sensor modules.

The Peltier Controller Temperature Monitor has been successfully used in stage 1 qualification test of module testing in the University of Bergen. The Peltier Controller has also been proven to work quite well.

Acknowledgement

I would like to thank my supervisor Professor Bjarne Stugu and Magne Eik Lauritzen, for the help and guidance they provided me. I also wish to thank Simon Huiberts for the help he has given me. I am grateful to my parents for their patience and support. Lastly I would like to extend my sincere thanks to my brother.

Contents

1 Introduction

The universe is composed a few building blocks called fundamental particles. These fundamental particles are governed by four fundamental forces, the strong force, the weak force, the electromagnetic force, and the gravitational force. Our best understanding of how the fundamental particles and the strong force, the weak force and the electromagnetic force are related to each other are encapsulated in the Standard Model of physics. The fundamental particles are divided into two sub categories, the fermions and the bosons. The fermions are the basic building blocks of matter and are further divided into two basic types called quarks and leptons. The bosons are what we call force carriers. Three of the fundamental forces result from the force-carrier particles. They are the strong force, the weak force, and the electromagnetic force. In 2012 ATLAS observed a new particle called the Higgs boson. This was of course a landmark discovery for the ATLAS experiment and CERN, but there are more physics for the ATLAS experiment to discover. The current Standard Model does not provide explanations the several phenomenon in particle physics. Phenomenon such as dark matter and dark energy.

The Large Hadron Collider will begin its High Luminosity upgrade in the third long shutdown in between 2025 and 2027 [1]. The upgrade will require the ATLAS detector to be upgraded. The current inner detector installed in the detector will be replaced with the inner tracker. The high luminosity upgrade will bring in new challenges to the detector. Higher luminosity will require the inner detector to be replaced with new pixel and strip sensors. These new sensors will have much higher radiation tolerance than the ones currently in the inner detector.

Norway will be part of the inner tracker upgrade and will produce and deliver the new silicon sensors and modules. The University of Bergen and the University of Oslo are collaborating in this project and encompass the Norway cluster. University of Oslo will produce the modules and University of Bergen will be testing them.

This thesis will focus on the development of a testing setup, used to test the new inner tracker sensor modules. Some IV-curve measurements have also been done on a few RD53A modules. The first part of this thesis will give a overview of the ATLAS experiment and the inner tracker upgrade. It will also explain some of the details about the new sensor modules. Then the next part will be the descriptions of IV-curves taken of RD53A modules.

The next part is focused on the development of the test setup for sensor modules. Here the development of a distrusted control system/Peltier controller called Peltier Controller Temperature Monitor will be described. A hardware interlock system, which will be used in the sensor module testing is also shown. The last part will be a overview of a qualification test that Bergen passed, where the Peltier Controller Temperature Monitor was used. Lastly, the current state of the testing setup in Bergen will be explained.

2 ATLAS and LHC

CERN stands for Conseil Européen pour la Recherche Nucléaire (European Council for Nuclear Research) and was founded in 1954. It is located at the Franco-Swiss border near Geneva. It has now 23 member states. The purpose of CERN is to study the fundamental particles. To achieve this particle accelerators and detectors were built for this purpose. Particles are made to accelerate close to speed of light and collide with each other using the particle accelerators. The collision are observed and recorded using the detectors. Such particle accelerator built are the Large Hadron Collider, which has four particle detectors: ATLAS, CMS, ALICE and LHCb [2].

2.1 Large Hadron Collider

The Large Hadron Collider (LHC) is currently the largest and most powerful particle accelerator in the world. It is a two-ring-superconducting-hadron accelerator and collider. The LHC is designed to to collide proton beams with center-mass energy of 14TeV. It can also collide heavy ions (Pb) with an energy of 2.8TeV per nucleon. The ATLAS, CMS and LHCb experiments are made to study the proton beams collisions. ATLAS and CMS is designed for a peak luminosity of $L = 10^{34} cm^{-2} s^{-1}$. LHCb is searching for B-physics and is aiming at a peak luminosity of $L = 10^{32} cm^{-2} s^{-1}$. The last experiment, ALICE, is dedicated to study the heavy ion collisions and is aiming at a luminosity of $L = 10^{27} cm^{-2} s^{-1}$ [3].

To begin the accelerator protons have to be stripped away from the hydrogen atoms. This is achieved by using electric fields to separate the proton and the electron from each other. Electric fields in the accelerators will switch from positive and negative at certain frequencies. This will pull the particles forward along the accelerator. It also ensures that the particles does not accelerate at a continuous stream, but at closely spaced bunches. The LHC also has designed metall chambers called RadioFrequency (RF) cavities. These cavities resonate at specific frequencies and allows radio waves to interact with the passing bunches. These radio waves will help accelerate the particle bunches. The particle bunches travels along a ultrahigh vacuum inside the beam pipe. This ensures that no particles collides with any gases through the accelerator [4].

The goal of LHC is to find new physics beyond the Standard Model with

center-mass energy of 14TeV. To find the number of events generated by the LHC collision, one would have to find the luminosity and the cross section. The relation is given as:

$$N_{event} = L * \sigma_{Event} \tag{2.1}$$

where L is the machine luminosity and σ_{event} is the cross section. The machine luminosity can then be written with a Gaussian beam distribution as:

$$L = \frac{N_b^2 n_b f_{rev}}{4\pi\sigma_x\sigma_y} = \frac{N_b^2 n_b f_{rev}\gamma_r}{4\pi\epsilon_b\beta*}F \tag{2.2}$$

σ_x and σ_y are the cross section of the bunch beams. ϵ_b is the normalized transverse beam emittance, $\beta*$ is the beta function at the collision point, N_b is the number of particles in a bunch, n_b is the number of bunches in a beam, γ_r is the relativistic gamma factor and f_{rev} is the revolution frequency. F is the geometric luminosity reduction factor, which are caused by crossing angle at the collision point. The factor F is expressed as:

$$F = (1 - (\frac{\theta_c\sigma_z}{2\sigma*})^2)^{1/2} \tag{2.3}$$

where θ_c is the full crossing angle at the collision point, σ_z is the root mean square of the bunch length and $\sigma*$ is the root mean square of the bunch transverse size. This expression is assuming that the beams are round, with $\sigma_z >> \beta$ and with equal beam parameters.

The integrated luminosity tells us how well the LHC performed during its run. Generally higher integrated luminosity generate more events, which means better statistics. The integrated luminosity over one run is given as:

$$L_{int} = L_0\tau_L(1 - e^{\frac{-T_{run}}{\tau_L}}) \tag{2.4}$$

where L_0 is the initial luminosity, τ_L is the luminosity lifetime and T_{run} is the total length of the luminosity run [3].

2.2 ATLAS

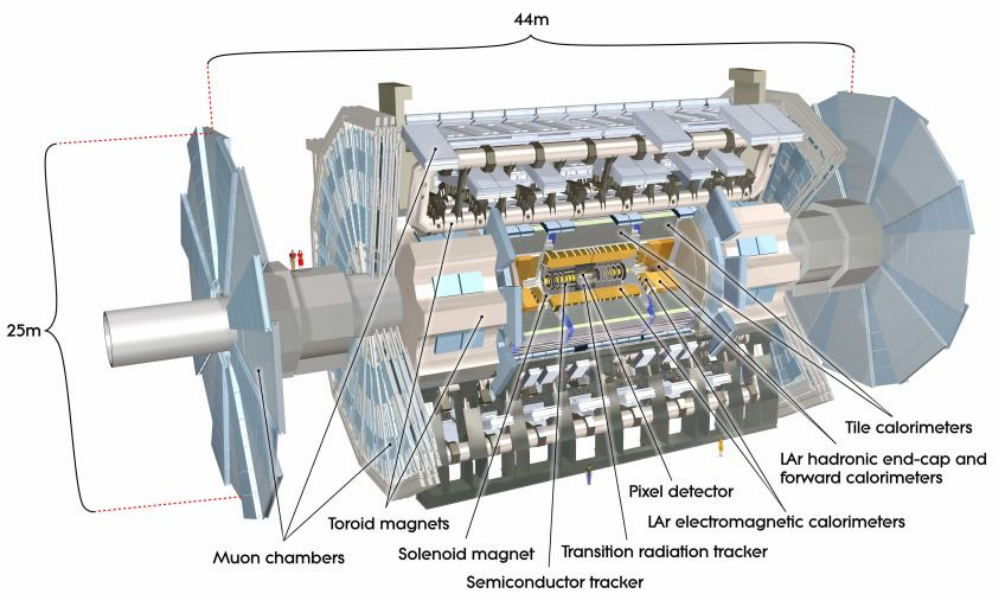


Figure 2.1: A cutaway view of the ATLAS detector. Figure is taken from [5]

ATLAS is one of the two detectors built for the purpose of probing proton - proton collisions. It stands for A Toroidal LHC ApparatuS. The ATLAS detector is 25m tall and 44m long and the weight of the detector is approximately 7000 tonnes. The detector is forward-backward symmetric with the respect of the interaction point. ATLAS is composed of a inner detector, calorimeters and muon spectrometers. The inner detector is made of discrete, high-resolution semiconductor pixel and strip detectors in the inner part. In its outer part are straw-tube tracking detectors with the capability to generate and detect transition radiation. The calorimeter consists of a high granularity liquid-argon electromagnetic sampling calorimeter. It has excellent performance in terms of energy and position resolution. The hadronic calorimeter consist of a scintillator-tile calorimeter. It is separated into a large barrel and two smaller extended barrel cylinders, one on either side of the central barrel. In the forward direction, the hadronic calorimeters is also of the liquid-argon type. The muon spectrometer is an air-core toroid system. It includes a long barrel and two inserted end-cap magnets, which generates strong bending power [5].

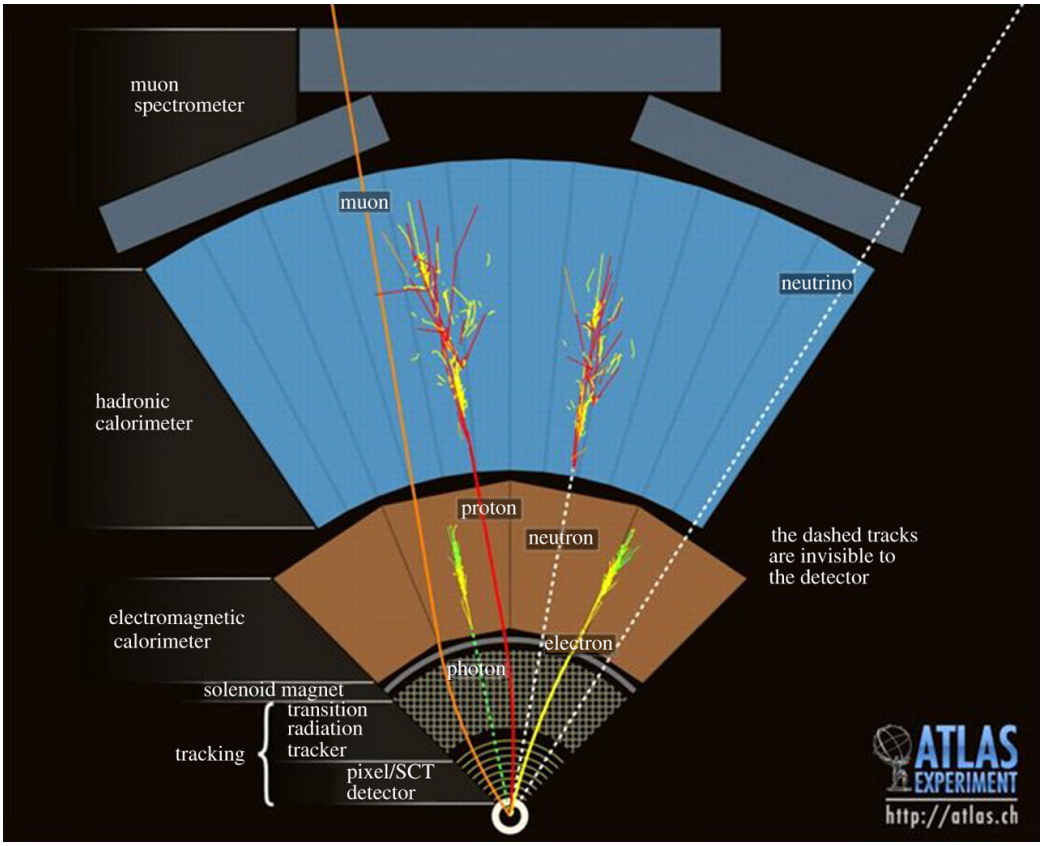


Figure 2.2: A schematic of a transverse slice of the ATLAS detector.

A thin superconducting solenoid surrounds the inner detector cavity. Three large superconducting toroids are arranged with an eight-fold azimuthal symmetry around the calorimeters. One of the toroids are for the barrel, and the two other are for the two end-caps. The solenoid surrounding the inner detector provides a magnetic solenoidal field of 2T. The different parts of the ATLAS detector are shown in Figure 2.1 [5].

Figure 2.2 shows the different pattern of energy deposits that allows the identification of different particles produced in the collisions. Charged particles such as electrons, muons and protons will leave tracks in the inner detector. The electrons will be stopped by the electromagnetic calorimeter. Hadrons will move past the electromagnetic calorimeter and will instead be stopped by the hadronic calorimeter. Muons will go all the way to the muon spectrometer. Non-charged particles such as photons and neutrons will not be visible in the inner detector. The photon will be stopped in the electromagnetic calorimeter, while the neutrons will continue past the electromagnetic calorimeter and stopped by the hardonic calorimeter. Particles

such as neutrinos are not visible in the ATLAS detector.

ATLAS uses a right handed coordinate system. The origin is at the nominal interaction point and the z-axis points along the beam pipe. The x-axis points from the interaction point to the center of the LHC ring, and the y-axis points upwards. Cylindrical coordinates are used in the transverse plane, with ϕ being the azimuthal angle around z-axis. Pseudorapidity is defined as $\eta = -\ln tan(\theta/2)$. It tells us the angle of a particle relative to the beam axis, with $\eta = 0$ being pendicular to the beam axis [6].

2.3 HL-LHC

The first phase of the LHC physics program was very successful. It reached more than 75% of its design luminosity and managed to deliver a integrated luminosity of around $25fb^{-1}$ to the ATLAS and CMS detectors. The data have yielded over 1000 publications. It have even managed to observe the Standard Models Higgs Boson in 2012. Now after the first phase of the LHC life time, the LHC will increase its luminosity. This is to maintain scientific progress and to explore the full capacity of the LHC. The LHC will change its name after the upgrade to the High Luminosity Large Hadron Collider (HL-LHC). The goal of the upgrade is to attain an integrated luminosity of $4000fb^{-1}$ [6]. This will be a increase of a factor 10 of the LHC design value [7].

The upgrade will begin during the third long shutdown between 2025 and 2027. Though, installation of the first components will begin during the second long shutdown between 2019 and 2021. The high luminosity upgrade will bring in new challenges. To accommodate the higher luminosity the magnets and optics have to be upgraded [1]

New superconducting quadrupole magnets will be installed on either side of the ATLAS and CMS experiments. This is to focus the bunches before they meet. The magnet will be made of a superconducting compound, called niobium-tin. This will make it possible to achieve a much higher magnetic field than the old niobium-titanium alloy used for the current LHC. This will increase the magnetic field to 12T from 8T. New beam optics will also be installed to ensure a constant collision rate throughout the lifespan of the beam. The new niobium-tin magnets will also replace the current bending magnets [1]. The current super conducting RF cavities will be replaced with the much better Crab Cavities [7].

The LHC is not the only part that needs to be upgraded. To maintain the performance of the ATLAS and CMS detectors, various parts have to be upgraded. The main challenges to overcome are the increased radiation damage caused by the HL-LHC and the high "pile-up" coming from the high instantaneous luminosity. To combat the new challenges the ATLAS detector will be upgrading its inner detector. The inner detector will be upgraded to a new, all-silicon Inner Tracker (ITk) [7].

3 Description of ITk and ITk Pixel

3.1 Layout of ITk

The HL-LHC will operate at an instantaneous luminosity of $L = 7.5 * 10^{34} cm^{-2} s^{-1}$. This corresponds to an average of 200 inelastic proton-proton collisions per beam crossing. In order to accommodate these new requirements the current ATLAS inner detector will be upgraded to ITk. The layout has to be optimized to maintain the same tracking performance as the present inner detector. The design of the ITk is based on a concept of "Hybrid Pixel Detector", which are currently implemented in the current inner detector. The Hybrid Pixel Detector has to go through several modifications to adapt the new HL-LHC conditions [6]. A more detailed description of the sensor modules are described in section 5.

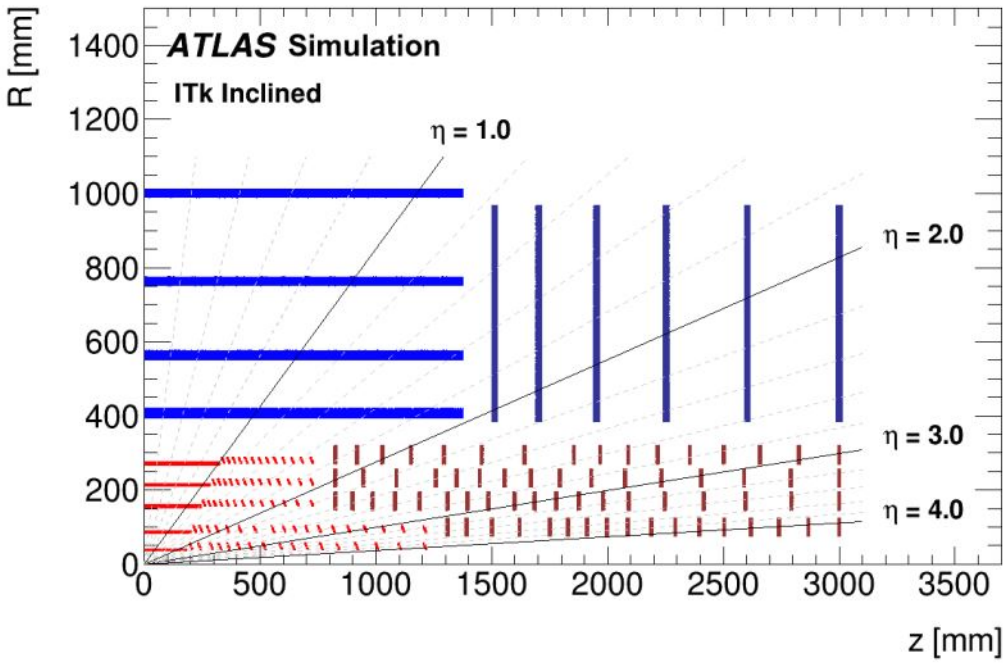


Figure 3.1: Schematic layout of the ITk. The blue are the active elemtents of the strip detector barrel and end-cap. The pixel detector barrel layer are shown in red and the end-cap rings are shown in dark red. Only one quadrant and active detector element is shown. The x-axis is the axis along the beam axis with zero being to interaction point. The y-axis is the radius measured from the interaction radius. Figure is taken from [6].

The ITk has two subsystem, a strip detector, which surrounds a pixel detector. The strip detector is composed of four barrel layers and six end-cap petal-design disk. It has a coverage of $|\eta| < 2.7$. This is increased to $|\eta| < 4$ by a 5 layer pixel detector. The pixel and strip detector are separated by a pixel support tube. Because of the harsh radiation environment of the new HL-LHC, the inner two layers of the pixel detector are replaceable. The inner two layers are separated from the three outer layers by an inner support tube. This inner support tubes facilitates the replacements of the two inner pixel layers. The pixel and strip detector provides a total of 13 hits for $|\eta| < 2.7$, except for in the barrel/end-cap transition of the strip detector. Here the hits counts are 11 hits. The pixel detector are designed to provide a least 13 hits from the end of the strip coverage for $\eta| < 4$ [6].

The inner system consist of a barrel region with two layers, layer 0 and layer 1. The two end-caps comprises of two layers of rings, ring 0 and ring 1. The layer 0 and ring 0 are equipped with 3D sensors and single chip modules. Layers 1 and ring 1 are equipped with thin planar sensors and quad modules [6].

The outer barrel are supported by staves oriented in the direction of the beam line. The staves are called "longerons" and are 2 meters long. It is divided in two sections. The central part covers $|\eta| < 1$ and are made of quad models oriented in parallel to the beam direction. The staves periphery have dual modules with sensors inclined 56°with respect to the beam direction. This inclined module concept requires two different types of longerons for layers 2 and 3 and for layers 4 [6].

The design for the outer end-caps uses three concentric set of circular crown shaped "rings". This design has two big advantages. The gap between the concentric rings can be used to route the services. The rings along the beam direction at different radii can be adjusted independently, which allows maximum coverage with reduced number of sensors. The rings are made of half rings, which consist of quad modules. These modules cannot be places side by side. Instead, they are placed on alternating side of the structure with overlap to provide maximum coverage in the azimuthal angle [6].

3.2 Fluence and Radiation Dose

The increase in luminosity will cause the radiation level in the ATLAS ITk to increase an order of magnitude compared to the current inner detector. The

outer pixel barrel and end-cap will be operated to collect a total integrated luminosity up to $4000 fb^{-1}$. The inner barrel and end-cap will be replaced after $2000 fb^{-1}$. The highest fluence in layer 0 is $1.31 * 10^{16} 1MeV n_{eq}/cm^2$. For layer 1 the highest fluence is $3.8 * 10^{15} 1MeV n_{eq}/cm^2$. The outer pixel layers (layer 2-4) will have a highest fluence of $3 * 10^{15} 1MeV n_{eq}/cm^2$ in the barrel and $3.8 * 10^{15} 1MeV n_{eq}/cm^2$ in the end-caps [6]. This means that the areas near the colliding beams has to withstand a fluene of $1.4 * 10^{16} 1MeV n_{eq}/cm^2$ [8].

4 Pixel Sensors

Since the early 1960s, the semiconductor sensors have been used in spectroscopic applications. The sensors use the same detection principle as gas-filled ionization chamber. The advantage with the semiconductor sensors is that the energy to produce a electron-hole pair is 3.6 eV in silicon, while it takes about 20eV to create ion pairs in a gas. A better energy resolution can thus be reached with the semiconductor sensors [9].

4.1 Fundamental Sensor Properties

4.1.1 Charge Carrier Concentration

An intrinsic semiconductor is a semiconductor with the concentration of impurities that are negligible compared to thermally generated free electrons and holes. The number of free n electrons can be calculated as:

$$n = \int_{E_c}^{\infty} N(E)F(E)dE \tag{4.1}$$

where N(E) is the density of states in the conduction band, F(E) is the Fermi-Dirac function and E_c is the energy of the lower bound conduction band arbitrary set to zero. N(E) is can be calculated as:

$$N(E)dE = 4\pi(\frac{2m_n}{h^2})^{2/3}E^{1/2}dE \tag{4.2}$$

where m_n is the effective mass of the electrons and h is the Planck constant. The Fermi-Dirac function is defined as:

$$F(E) = \frac{1}{1 + e^{(E-E_F)/kT}} \tag{4.3}$$

Here, E_f is the Fermi energy, k is the Boltzmann constant and T is the temperature in Kelvin. The Fermi function can be approximated by an exponential function in the conduction band:

$$F(E) \approx e^{-(E-E_F)/kT} \tag{4.4}$$

Putting together Equation 4.2 and 4.3 in Equation 4.1, and calculating the integral, we get the concentration of free electrons:

$$n = 2(\frac{2\pi m_n kT}{h^2})^{3/2} e^{-(E_C-E_F)/kT} = N_C e^{-(E_C-E_F)/kT} \tag{4.5}$$

and free holes:

$$p = 2(\frac{2\pi m_p kT}{h^2})^{3/2} e^{-(E_F-E_V)/kT} = N_V e^{-(E_F-E_V)/kT} \tag{4.6}$$

where m is the effective mass of the charge carriers, E_C is the energy of the conduction band, E_V is the energy of the valence band. Multiplying Equation 4.5 with 4.6 gives us:

$$np = n_i^2 = N_C N_V e^{-(E_g)/kT} \tag{4.7}$$

E_g is the energy gap defined as $E_C - E_V$. Because in intrinsic semiconductors the number of free electrons are equal to the number of holes, Equation 4.7 can be used to calculate the intrinsic carrier concentration [9].

4.1.2 Doping

Materials used in semiconducting devices are usually not intrinsic, but are doped with a small fraction of other materials. This leads to changes in the materials conductivity. Elements used to dope silicon are from the third or the fifth group in the periodic table. They have either one valence electron less or more than silicon. If the material has more valence electron than silicon, it is called a donor. The extra valence electron gets released into the conduction band. This leads to an increase of electron concentration, while the hole concentration decreases. A silicon doped with donors is called n-doped material. Here the electron are the majority carriers and holes are the minority carriers. Majority carries are the charge carriers that carry most of

the electric current, while minority carriers carry a smaller amount of current [9].

If one would dope a silicon with e.g boron instead, a material with one less valence electron, the silicon would produce a free hole. Boron has one less electron than silicon and therefore will take an electron from the valence band and establish a covalent bond with the neighbouring silicon atom. Boron and materials with the same property are therefore called an acceptor. A semiconductor doped with an acceptor is called a p-doped material. These materials have holes as the majority carriers while the electron is the minority. Semiconductors can contain both donors and acceptors. Though, in most case, one of the dopant are dominant [9].

4.1.3 P-N Junction

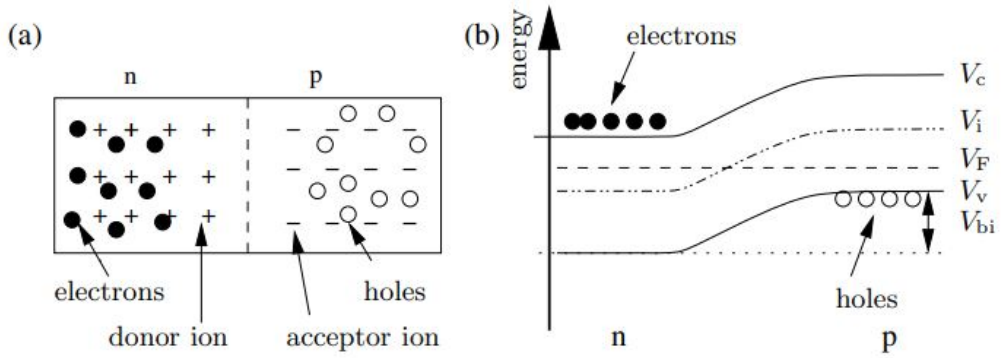


Figure 4.1: A figure illustrating the p-n junction (a) and band diagram (b), which illustrates the formation of built-in voltage. Figure is taken from [9].

If a n-doped and p-doped material is put together, some of the electrons from the n-doped side will diffuse into the differently doped side. This is due to the concentration differences. The charged carriers will recombine with the majority carriers. This will produce a zone near the junction, where there are no free charge carriers. This zone is called a depletion zone. In this depletion zone the acceptor ions and donor ions are left without their reversely charged carriers. Thus the region is electrically charged and is called the space charge region. The space charge region causes a electrical field, which counteract the diffusion of charge carriers. The space charge region can be characterized as the built-in voltage:

$$V_{bi} = \frac{kT}{e} \ln(\frac{n_{0,n} p_{0,p}}{n_i^2}) \approx \frac{kT}{e} \ln(\frac{N_D N_A}{n_i^2}) \tag{4.8}$$

where $n_{0,n}$ is the electron concentration in the n-doped side and $p_{0,p}$ is the hole concentration in the p-doped side. A complete ionization of donors and acceptors can be assumed. The electron and hole concentration can then be replaced with concentration of donors, N_D, and acceptors, N_A [9].

4.1.4 Reverse Bias

If an external voltage is applied in the same direction as the built-in voltage, the space charge region will increase as more of the majority carrier from both sides are removed. The junction are now reversely biased. The following equations will follow the convention that the voltage are positive for reverse bias, and negative for forward bias. This is due to that sensors are operated reversely biased. The width of the depletion zone can be calculated as:

$$W = x_n + x_p = \sqrt{\frac{2\varepsilon_0 \varepsilon_{Si}}{e}(\frac{1}{N_A} + \frac{1}{N_D})(V - V_{bi})} \tag{4.9}$$

W is the total width of the depletion zone, x_n and x_p is the width of the n-side and the p-side respectively. V is the applied external voltage and V_{bi} is the built-in voltage. Often, silicon sensor is using highly doped p-doped silicon with low-doped n-material. This means that the $\frac{1}{N_A}$ can be neglected. The built-in voltage is also much smaller than the typical operational voltage. Therefore the built-in voltage can also be neglected. These two assumptions lead to:

$$W \approx x_n \approx \sqrt{\frac{2\varepsilon_0 \varepsilon_{Si}}{eN_D} V} \tag{4.10}$$

This equation is widely used for calculating the depletion depth. If the bias voltage increases, the width of the depletion zone will increase. When the bias voltage reaches the full depletion voltage V_{depl}, which is when the width of the depletion zone covers the whole thickness of the device. This V_{depl} often defined as the minimum operation voltage. If the bias voltage exceed the full depletion voltage the device will be overdepleted [9].

Current signal that are caused by radiation can be detected by a charge-sensitive amplifier. When there are no radiation, there are always a steady leakage current or dark current. This leakage current can be caused by free charge carriers from the undepleted volume diffusing into the space charge region. The main cause of leakage current is thermal generation at generation-recombination centers at the surface and in the depleted volume. The thermal generation of electrons can be described as:

$$J_{vol} = -e\frac{n_i}{\tau_g}W \tag{4.11}$$

where n_i is the intrinsic carrier concentration, τ_g is the generation lifetime and W is the width of the depletion zone. The temperature dependence is hidden in the intrinsic carrier concentration (see Equation 4.7). If one would further increase the bias voltage, electrical breakdown will occur [9].

4.1.5 Radiation Damage

Radiation damage is divided into bulk and surface defects. The bulk defects are caused by displacement of crystal atoms. The surface defects includes effects covering dielectrics and the interface region. Main effects of radiation damage are increase of the leakage current, change of the space charge in the depleted zone, increase of full depletion voltage and charge trapping [9].

When high energy particle passes the sensor, it will not exclusively interact with the electron cloud producing the electrical signal. Hadrons, such as pions and protons will also interact with the nuclei. This will often cause them to displace them out of the lattice position. This will produce crystal imperfection, which may be electrically active. Consequently it will change the electric properties of the material. Thus the effective doping concentration will change and the full depletion voltage will increase. Energy levels in the band gap caused by the crystal defects can act as generation-combination centers. This will cause a decreace of generation lifetime and thus increase in the volume generation current. Therefore it will cause an increase of leakage current. These defects are not stable, but are able to move through the crystal. This movement causes annealing if the defects meets during their migration through the crystal. Radiation-induced effects also causes charge trapping. These traps are unoccupied in the depletion zone due to lack of charge carriers and can hold parts of the signal charge for a time longer than

the charge collection time. Thus the signal height will be reduced [9].

4.2 3D Sensors

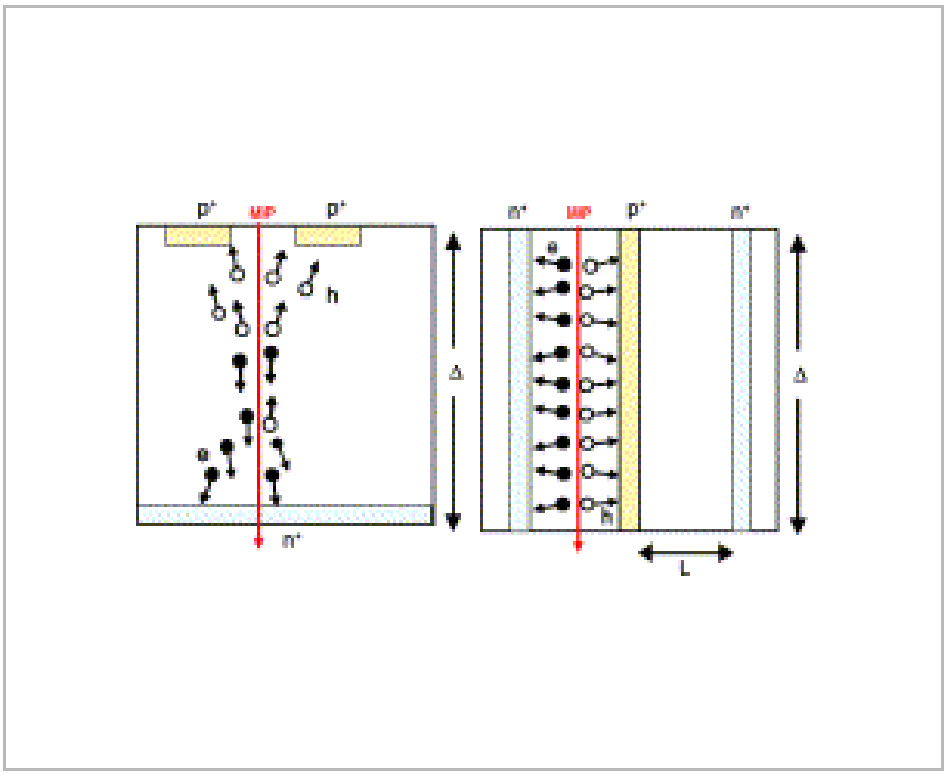


Figure 4.2: A schematic cross-section of planar (left) and 3D sensor (right). Δ is the active thickness and L is the collection distance in 3D sensor.

3D silicon detectors are more radiation tolerant than other sensor technologies. This is because of the distance between the columnar electrodes, which penetrate the sensor bulk perpendicular to the surface. It is therefore decoupled from the sensor thickness and can be chosen to be smaller than the thickness of standard planar sensors. The reduced distance between the electrodes leads to less charge trapping from radiation induced defects and also lowers the operational voltage. This will lead to lower power dissipation after irradiation. The baseline design of the ATLAS ITk detector the active thickness will be $150\mu m$, with an extra $100\mu m$ passive support wafer to facilitate handling and bump-bonding. The 3D sensor technology will be used

in the layer 0 and ring 0 of the ITk detector [6]. The Figure 4.2 shows the difference between a regular planar sensor and a 3D sensor.

5 Sensor Modules

The design for the pixel module is called a hybrid pixel module. This hybrid pixel module is made out of two parts: a passive high resistivity silicon sensor and a front-end read-out chip, and a flexible PCB. The former is called a bare module and the latter is called the module flex. The schematic of a bare pixel module is shown in Figure 5.1 and a protoype triplet ring module can been seen on Figure 6.15. The bare module and the flex module are joined together using a high density connection technique called "flip-chip bump-bonding" [6].

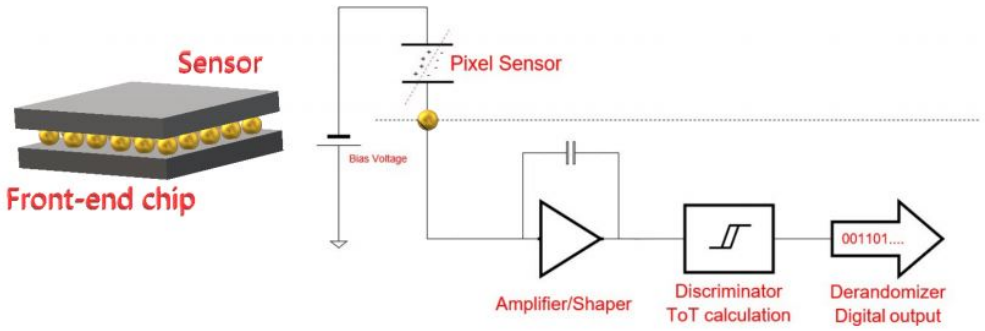


Figure 5.1: A schematic view of a bare pixel module. Source: [6].

5.1 Bare Module Assembly Process

The hybridization process are split into four main stages:

1. Bump deposition
2. Sensor wafer processing
3. Wafer thinning and dicing
4. Flip-chip bonding

Bump deposition is the deposition of solder or indium bumps on the front-end wafer. Sensor wafer processing includes the deposition of a solderable

metal layer on the sensor pixel pad. This known as the under-bump metallization. The front-end chip wafer will be thinned to $150\mu m$ in the wafer thinning step. After dicing the wafers, the dies are sorted and only the good ones are used for flip chip. The flip chip bonding is a process where the front-end chips are joined face to face with the sensor chip. The process requires precise alignment of the parts. Then pressure and heat is applied to form an electrical and mechanical bond [6].

5.2 RD53 Collaboration

The RD53 collaboration was established with the goal of developing technology and design platform for pixel chips needed by ATLAS and CMS at the HL-LHC. The RD53A integrated circuit validates the design environment and basic circuit blocks, which will be reused in the production chip. The physical size of the RD53A chip is half the size of the production chip. This is to lower the cost of prototyping, while being able to validate "large chip" effects in both design and performance [6]. The next generation chip called RD53B are complete full size pixel chips. The RD53B chips are made specifically for each experiment (ATLAS and CMS), as each chip size needs to be adapted to the specific needs to each experiment. The chip has the same basic architecture as RD53A, but has extra features not present in RD53A generation chip [10].

5.2.1 RD53A and RD53B

The RD53A chip is 20mm wide and 12mm tall. It contains 76 800 pixels and 240 M transistors [6]. The chip uses a 9 metal layer stacks, which consist of 7 thin, 1 thick and 1 ultra thick metal layers. The sensitive area of the chip is placed at the top of the chip. It is arranged in a matrix of 192 x 400 pixel and has a size of $50\mu m$ x $50\mu m$. The pixel matrix is built up of 8 x 8 pixel cores. The 64 front ends within a core is placed as 16 "analog islands" with 4 front end each. These analog islands are surrounded a synthiezed digital sea. An illustration of the analog island and digital sea can be seen in Figure 5.2 [11]. These digital pixel cores handles all the processing of the front end binary output. This includes masking, digital injection, charge digitization using Time over Threshold (ToT), storage of ToT values, latency timing, trigging and read-out [6].

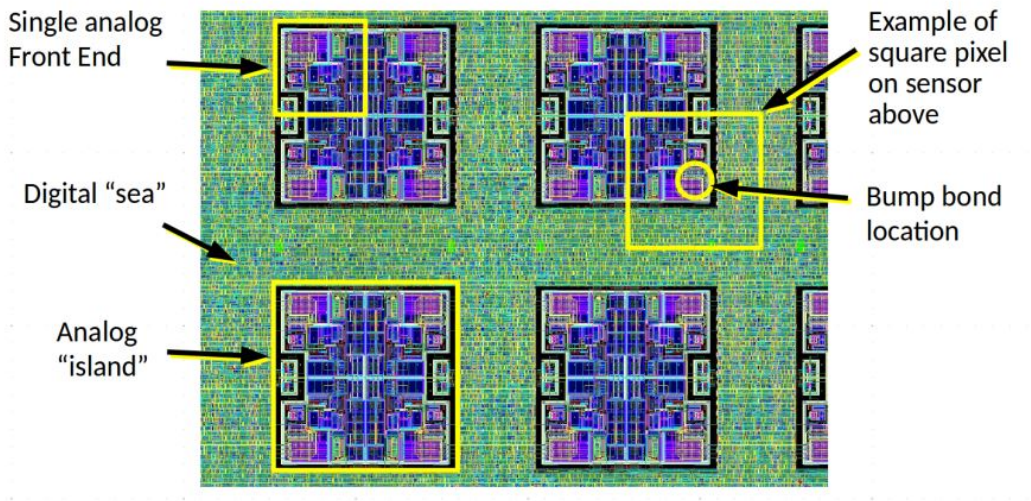


Figure 5.2: An illustration of the concept of analog islands surrounded by the digital sea. The analog islands are in blue and the digital sea is the green surrounding it. Figure is taken from [6].

RD53A is designed to support serial powered operation. The chip will assume a constant current, instead of constant voltage power supply. To achieve this Shunt Low Drop Out (ShuSLDO) are used to generate internal constant voltage rails while drawing a constant current from an external power source. The chip contains two ShuLDO control circuits. One of the circuits are for the analog internal rails and the second is for the digital internal rails [11].

The RD53A chip has three different front end designs. This is to allow detail performance comparisons. The three front ends are: Differential, Linear and Synchronous front ends. The arrangement of the three different front ends are illustrated in Figure 5.3. The differential front end are made of purely analog circuit. This means that it contains no memory latches, flip-flops or counters. The front end uses a differential gain stage in front of the discriminator and implements a threshold by unbalancing the branches. The linear front end uses a linear pulse amplification in front of the discriminator. This will compare the pulse to a threshold voltage. The synchronous front end implements a baseline "auto-zeroing" scheme. This will require periodic acquisition of a baseline instead of pixel-pixel threshold trimming. It is not possible to have equal area of all front ends because the 400-pixel wide matrix is made out of 8 by 8 synthesized cores. As seen in Figure 5.3 two of the front ends, differential and linear front ends, have a 17 core width, while

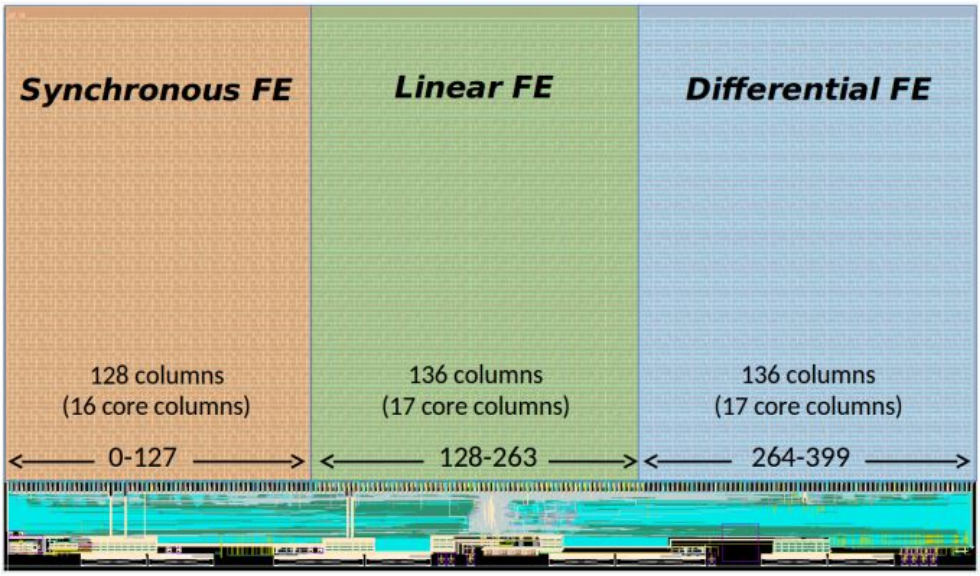


Figure 5.3: Photograph of the RD53A chip. It shows the three different flavor front ends. Figure is taken from [6].

the synchronous front end has a 16 core width. The synchronous front end was chosen to have 16 core width because of it was simulated to use slightly more power. The differential and linear front ends are placed beside each other since they have similar functionality. This allows the largest possbile area to have an uniform response [11].

As said before, the RD53B is a full size pixel chips. The RD53B has the same pixel core size as RD53A (8 x 8 pixel cores). The pixel matrix is made of 384 x 400 pixels with size of $50\mu m$ x $50\mu m$. Serial powering is improved from the RD53A, with improvement such as start-up, add over-current and overcharge protection. RD53B has only one front end. ATLAS choose the differential front end, while CMS chose the linear type. RD53B still uses the same analog island in a digital sea design as RD53A [12].

5.3 Characterization and IV-curves

IV-curves are powerful tool for sensor testing. It can show many problems, such as problems during sensor production process and effects from the radiation damage. IV-curves are measurements of the leakage current. Multiple measurements are done over various different temperatures.

5.3.1 Setup for IV-Curve Measurements

An environmental chamber was used to control the environment temperature. This will allow us to control the temperature down to around -60°C. For the bias voltage a Keithley 2400 sourcemeter was used. A TTI PL303QMD Quad-mode dual power supply was used to power the sensor chip when needed. The environmental chamber had a hole drilled to the side. The hole was connected to a dry air supplier. This will ensure that no dew was formed on the sensor. To control the Keithley 2400 sourcemeter, a PC with a python script was used. The python script was automated, where it will automatically step in voltage and take current measurements. The voltage was turned on and off at every step increase. This was to take account for the sensor self heat. The current was also measured with a sense wire setup. A sensor to monitor the relative humidity was also used. The temperature measurements are taken directly from the temperature shown by the environmental chamber. IV-measurements of four RD53A sensors was taken, D59-2, D62-1, D67-1 and D67-2.

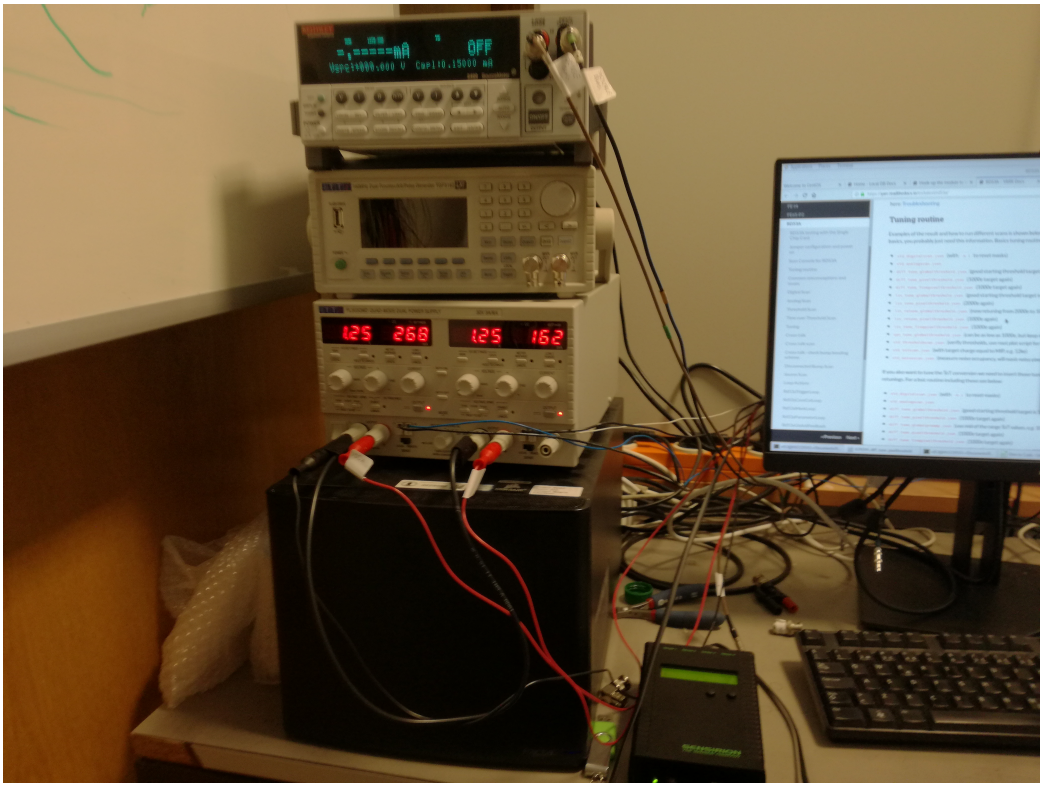


Figure 5.4: Photograph of the power supplies used in IV-measurements.

5.3.2 IV-curves

Figure 5.5 shows the IV measurement of the RD53A-D67-2 sensor. The environmental chamber was not covered up during the measurements. We was not able to power up the chip. The chip was probably damaged by the radiation as the sensor has been highly irritated with $1.5 * 10^{16} Neq/cm^2$. The sensor does not seems to show any noticeable breakdown. The leakage current was measured from -60°C to -10°C. At -10°C the leaked current seems to have a different curve than at other temperature. The curve seems to be very linear from 0V to 60V. It could be that the breakdown happened at a very low voltage and it acts now as a constant resistor.

The RD53A-D67-1 sensor shows a more understandable curve as seen in Figure 5.6. We can see that the breakdown voltage is at around 6V. The curve is nearly linear after the breakdown voltage. The leakage current was measured only from two temperatures: -30°C and -60°C. The leakage current are little higher at -30°C. This is consistent as the leakage current will increase with temperature. Like the D67-2 the D67-1 sensor was also irradiated with $1.5 * 10^{16} Neq/cm^2$. During the measurement the chamber was not covered up and the chip was not powered.

Figure 5.7 shows the IV-curve of the RD53A-D62-1 sensor. We can see that it has a normal IV-curve, where the breakdown is at around 110V and the sensor is fully depleted at around 5V. The leakage current was measured at -10°C and at -25°C. The measurement was also done by regular ramping up the voltage, and also switching the voltage off and on after each increase in step. There was not much of a difference between the two methods. This time, the chamber was covered up during the IV-measurement and the chip was off. This was to ensure that the light did not affect the leakage current. To help with the cooling of the sensor, a fan was placed under the sensor.

D59-2 sensor seems to be completely broken. We can see from Figure 5.8 that the breakdown voltage is very close to 0V. It seems to reach breakdown voltage immediately. The graph is linear all the way to 6V. The compliance of the Keithley 2400 sourcemeter was set to 0.110mA. At 6V the leakage current reached the compliance and any higher voltage will result in 0.110mA. The leakage current was measured from -60°C to -10°C.

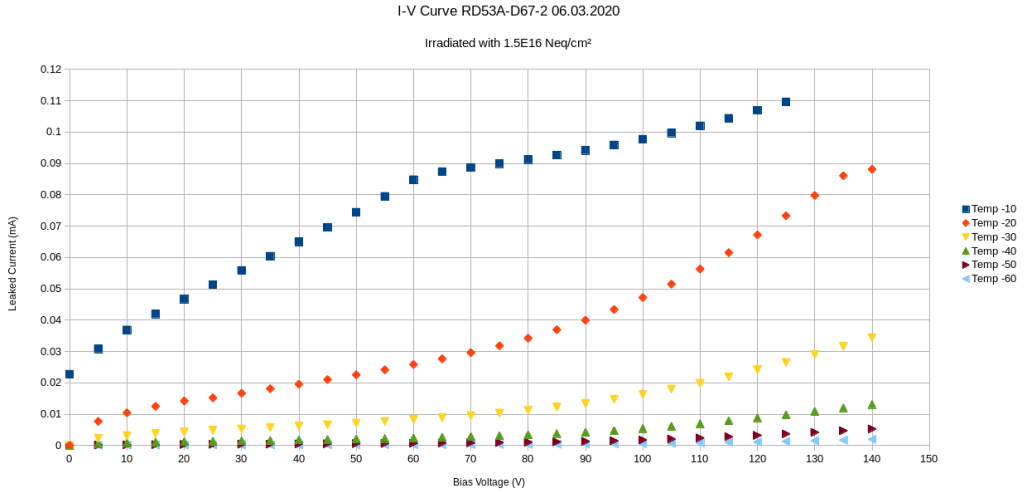


Figure 5.5: IV-curve of the RD53A-D67-2 sensor. The sensor was irradiated with $1.5 * 10^{16} Neq/cm^2$. The x-axis shows the bias voltage and the y-axis show the leaked current [mA].

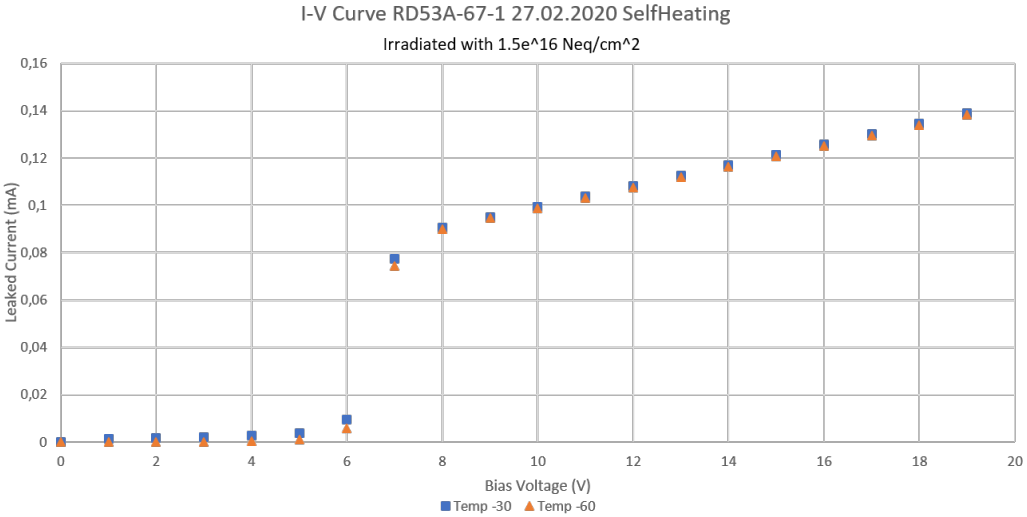


Figure 5.6: IV-curve of the RD53A-D67-1 sensor. The sensor was irradiated with $1.5 * 10^{16} Neq/cm^2$. The x-axis shows the bias voltage and the y-axis show the leaked current [mA].

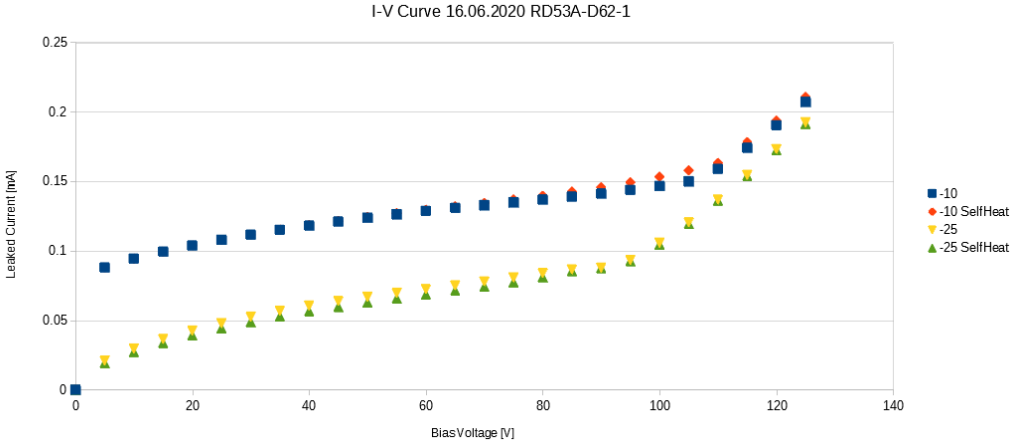


Figure 5.7: IV-curve of the RD53A-D62-1 sensor. The sensor was irradiated with $10^{16} Neq/cm^2$ The x-axis shows the bias voltage and the y-axis show the leaked current [mA]. The IV-curve was measured with and without taking account of self heating.

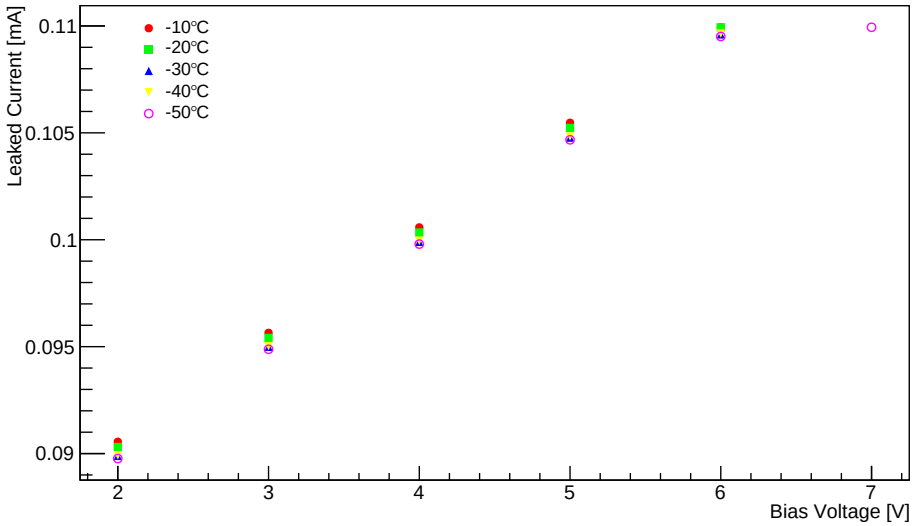


Figure 5.8: IV-curve of the RD53A-D59-2 sensor. The sensor was irradiated with $5 * 10^{15} Neq/cm^2$ The x-axis shows the bias voltage and the y-axis show the leaked current [mA].

5.3.3 Performance Measurement

The ATLAS detector can not operate with modules sensors that draws uncontrollably too much current. Therefore, the stability of a RD53A sensors was also measured. This was done using the same setup as IV measurement. The leakage current was measured in a 24 hour period with a constant bias voltage. A fan was also placed under the sensor to help with the cooling. The test was done at -10°C and -25°C. -25°C is the expected temperature at the start of the run, and -10°C is the end of life temperature. A bias voltage of -75V was also chosen as it is the voltage where the efficiency is highest after an irradiation of $10^{16} Neq/cm^2$. The long time measurement was done on the RD53A-D62-1 sensor, as it has the best IV-curve. It also is fully depleted at -75V and is highly irradiated with $10^{16} Neq/cm^2$. This makes it a good candidate to test the performance of the sensor.

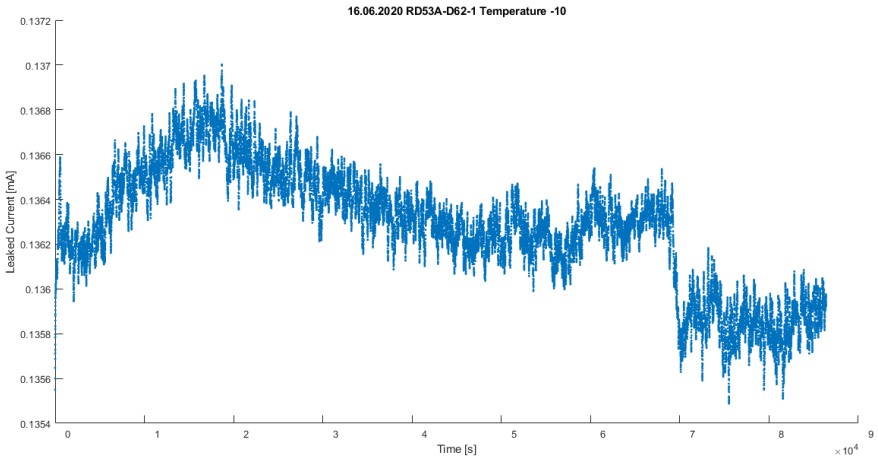


Figure 5.9: Performance measurement of the D62-2 sensor at -10°C. The measurement was done over 24 hours. The x-axis shows the time in seconds and y-axis the leakage current in mA. The measurement was done with a bias voltage of -75V.

The leakage current at -10°C is very stable. As one can see in Figure 5.9 the leakage current fluctuate with around 0.0004mA. We can also see that the leakage current starts at around 0.1356mA and after a couple of seconds, starts to increase to 0.1362mA. This is because of the sensor heating it self up, thus increasing the leakage current. At around 20 hour the leakage current seems to have a noticeable drop. This could be caused by the fluctuation of the temperature in the environmental chamber. Other changes in the leakage

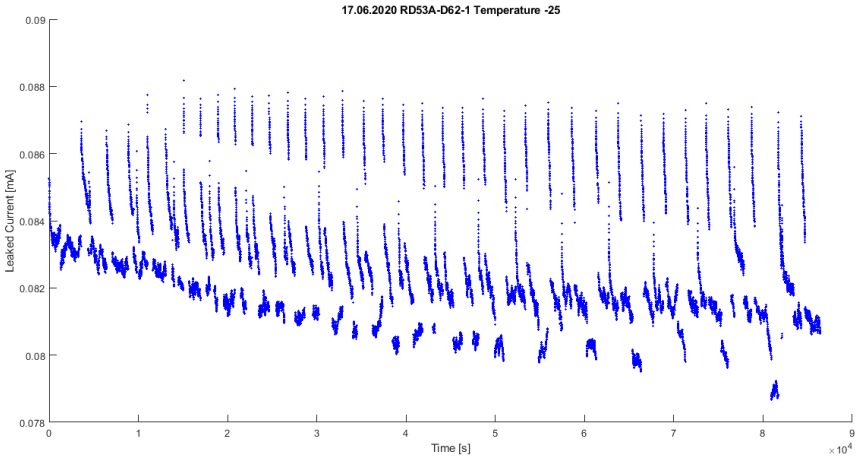


Figure 5.10: Performance measurement of the D62-2 sensor at -25°C. The measurement was done over 24 hours. The x-axis shows the time in seconds and y-axis the leakage current in mA. The measurement was done with a bias voltage of -75V.

current is probably caused by small temperature changes in the chamber.

Figure 5.10 shows the measurement done at -25°C. Here the leakage current fluctuate with 0.004mA. This is larger than at -10°C, but still very small. Overall the leakage current does not change much and is stable. The leakage current seems to swing up and down periodically. This may be caused by that the environmental chambers fan also stops and starts periodically. If this is the case the temperature could fluctuate enough to cause these periodic cycles of the leakage current.

6 Test Setup for Sensor Modules

6.1 Testing Procedures of RD53A Modules

During the production flow the module must go through several test. These tests include visual inspection of the module, weighing, basic electrical test and full electrical test at different temperatures. Figure 6.1 describes the production flow and the test which should be done at each step. The temperature during testing will be what is measured on the module negative temperature coefficient (NTC) sensor. No electrical test can be done until after wirebonding. It is then important to perform full electrical test after wirebonding at high and low temperatures. The high and low temperatures are defined as 20°C and -25°C respectively for 3D module sensors. A final full electrical test will be done at the end of production. Between the steps basic electrical test will be done to ensure the module is still functioning [13].

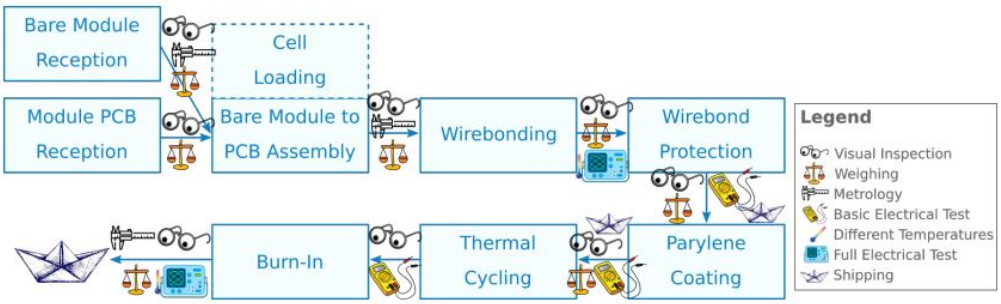


Figure 6.1: Schematic of assembly flow. Figure is taken from [13].

After receiving the bare module package, the packaging will have to be checked for damages and anomalies. Evaluation of any data logger such as temperature, humidity and impact/shock must be done after opening the package. Visual inspection of the module includes checking for damages/anomalies, e.g., loose wirebonds, corrosion. The inspection have to be documented with photographs and uploaded to a database [13].

The full electrical test includes IV test for checking sensor damages. During the IV test the front end should be powered off. The test requires a light-tight environment with a temperature at -25°C with a tolerance of ±2°C for 3D sensors. The relative humidity must be below 50% and the dew point should not go above -10°C. The front end also has to be tested. It is required that the front end must be able to start in Shunt Low-dropout (SLDO) mode

at high temperature (20°C) and at -35°C. To check for pixel failures the front end must be tuned. The module must have 30V applied during the tuning procedure. Source scan will also be done [13].

Basic electrical test will be done after each stage to verify if the module is still functional. The electrical test will be a subset of the full electrical test done at high temperature. The test include IV test with the Front End turned off and tuning [13].

The modules will be subjected to around 100 cycles down to -45°C and in worst case one cycle down to -55°C during operation. Because of this, the module also has to go through thermal cycling during production. Each module must go through 10 cycles from -45°C to 40°C and one cycle from -55°C to 60°C. The module will not be powered on during the procedure. The ramp rate must not go faster than 15°C/minute and the preferred ramp rate is 10 to 14°C/minute [13].

6.2 Peltier Controller Temperature Monitor (PCTM)

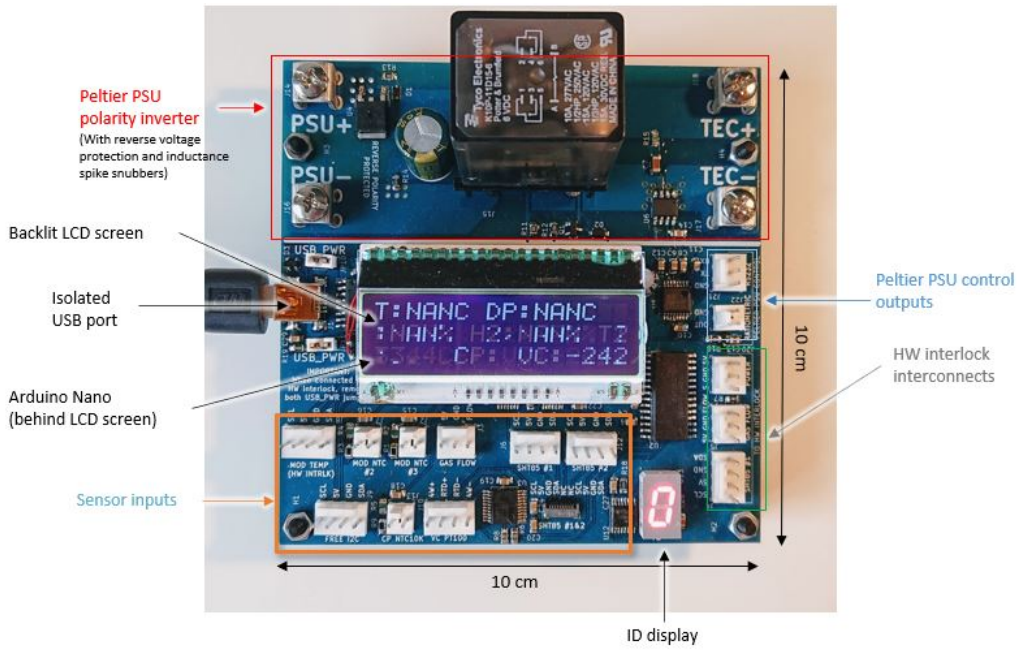


Figure 6.2: A picture and an overview of the PCTM. Source: [14]

During QC testing the module has to be tested at different temperatures.

It is also required that relative humidity and dew point stays below or above certain level. (See section 6.1). This means that the temperature, relative humidity and dew point has to be monitored during electrical test and thermal cycling. The temperature to the module also has to be controlled. To control the temperature, Peltier elements will be used. Therefore one must also have a way to control the Peltier.

The Peltier Controller Temperature Monitor (PCTM)[1] is made to monitor temperature and relative humidity during QC testing. Regular Peltier controller exist but are expensive, and in addition to needing a laboratory power supply, the cost can be high. Therefore the PCTM aims to lower the cost by also functioning as a Peltier controller. The PCTM can also monitor different parameters during QC testing such as, purge gas flow, vacuum chuck temperature, cold plate temperature and the current to the Peltier. The PCTM consist of two part, a PCB and a python GUI.

The PCB part of PCTM consist of a polarity inverting part, a backlit LCD screen, an Arduino Nano, sensor inputs, Peltier PSU control output and Hardware Interlock interconnections. The different parts can be seen in Figure 6.2. The Arduino Nano[2] is a micro controller. It is the "brain" of PCTM, everything is controlled by the microcontroller. It is connected to the PC through a USB port. It can either receive power through the USB from a PC or it can be powered by the hardware interlock. A Python GUI application can then be used to communicate with the PCTM. The PCTM also has a small ID display. It is used to differentiate other PCTM from each other, since multiple PCTM can be connected to the same PC.

6.2.1 PCTM Sensor Inputs

The PCTM has three NTC inputs. The first NTC input will be connected to the hardware interlock using a I2C serial communication protocol. Reason for this, is that we want to monitor the same NTC value that the hardware interlock also receives. The hardware interlock also has high precision resistor, which means that the hardware interlock will have a more accurate reading of the NTC. The microcontroller receives a 12bit digital value from the hardware interlock. The 12bit value represents the ratio between the measured voltage from the NTC and the supply voltage. To calculate the

[1] PCTM is custom made by Magne Lauritzen

[2] Arduino Nano is distributed by the Arduino Company

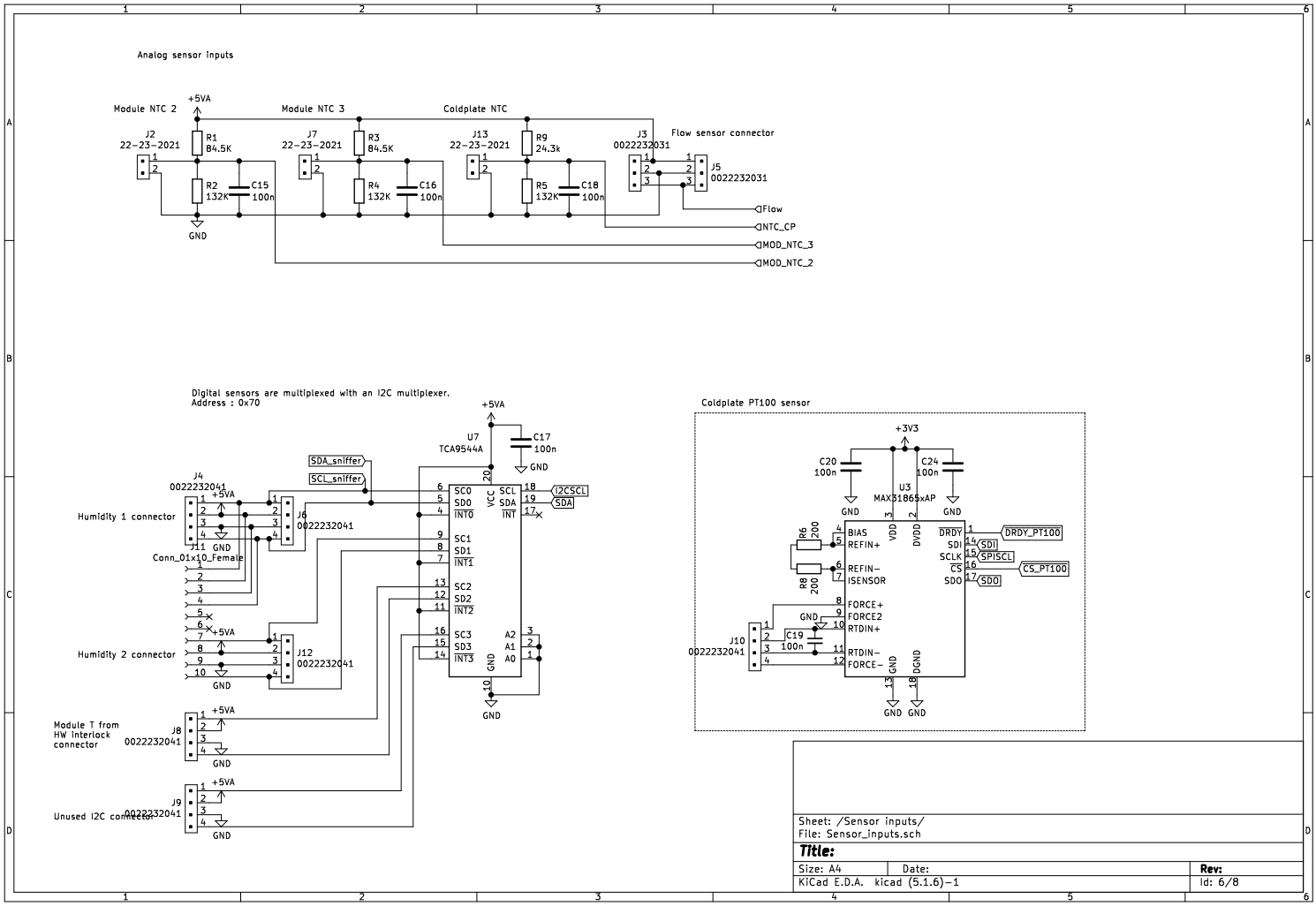


Figure 6.3: A schematic of the PCTM sensor inputs

voltage we use Equation 6.2 described further down in this section (dividing by 4096 instead of 1023). The rest of the NTC inputs (NTC1, NTC2 and Cold plate) are read out using a linearisation resistor network. The resistor values are shown in Figure 6.3. Using the linearisation resistor network we can simply calculate the resistance of the NTC using a simple equation:

$$R_{NTC} = \frac{V_{NTC} * R_1}{V_S - V_{NTC} * (1 + \frac{R_1}{R_2})} \tag{6.1}$$

where V_{NTC} is the voltage read out by the microcontroller and V_S is the supply voltage. R_1 and R_2 are the resistor values from the resistor network. From Figure 6.3 we obtain the value for the linearisation network, where $R_1 = 84.5K\Omega$ and $R_2 = 132K\Omega$. The cold plate resistor network has different values, because they are tuned to a higher temperature range than the module NTCs. The values for cold plate linearisation network are $R_1 = 24.3K\Omega$ and $R_2 = 132K\Omega$. The microcontroller measures the voltage based on the ratio between its supply voltage and the voltage measured. The value the microcontroller outputs is a digital 10bit value going from 0 to 1023, where 0 means 0V and 1023 means 5V. To translate that back to voltage we use:

$$V_{NTC} = \frac{V_{Digital}}{1023} * V_S = V_{ratio} * V_S \tag{6.2}$$

Here $V_{Digital}$ is the 10bit value and $V_{ratio} = \frac{V_{Digital}}{1023}$. Putting this in Equation 6.1 we obtain:

$$R_{NTC} = \frac{V_{ratio} * R_1}{1 - V_{ratio} * (1 + \frac{R_1}{R_2})} \tag{6.3}$$

We can see that the voltage supply is cancelled out. This is good since the voltage supply can vary from 4.5V to 5.5V depending on how the Arduino microcontroller was manufactured. We can then calculate the NTC resistance using only the 10bit digital value obtained from the microcontroller. All NTC inputs will come from 10K NTC. 10K NTC are NTC where the NTC resistance is $10K\Omega$ when the temperature is 25°C. The NTC temperature is then calculated using a fit over the NTC table obtained from *NTC Chip Thermistor Temperature Sensing Device Datasheet* [15]. The fit is shown in Figure 6.4 and is fitted very well with the NTC table.

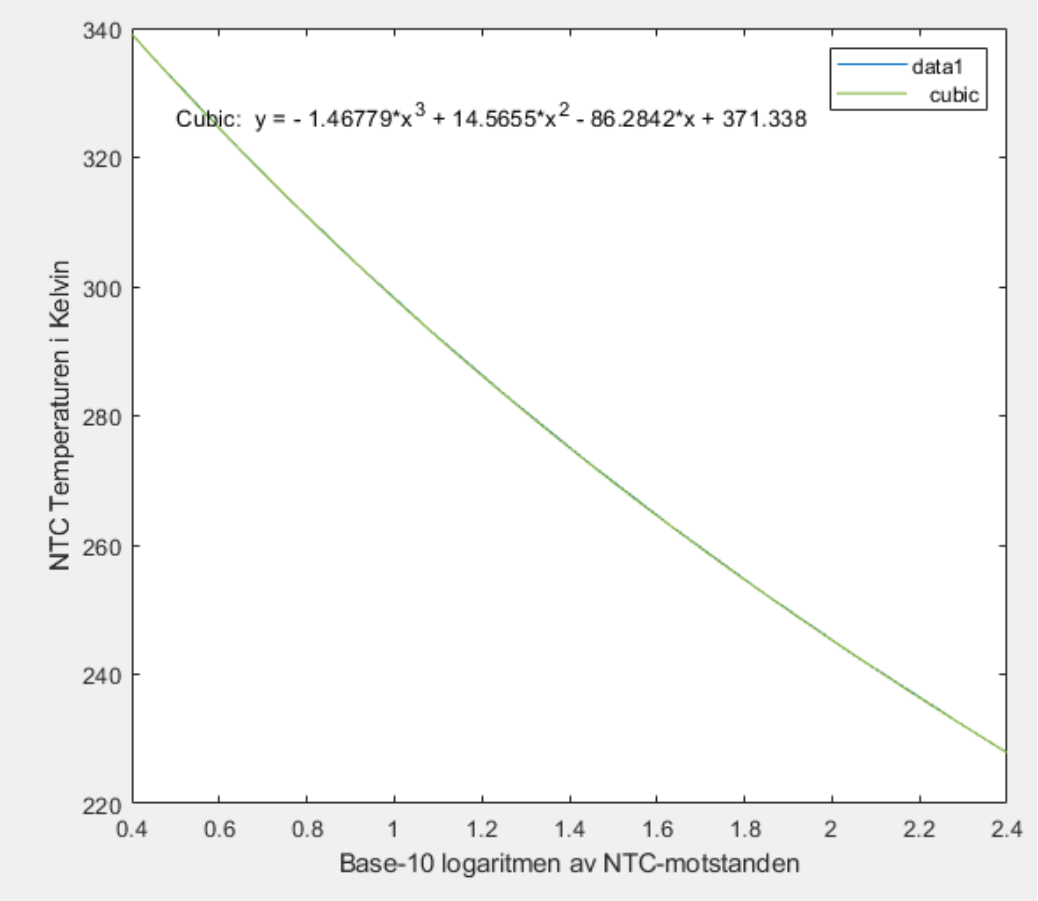


Figure 6.4: A graph of the cubic fit (green line) done over NTC table (blue line). The blue line is hiden under the green line. Y-axis is the temperature in Kelvin and X-axis is the NTC-resistance in Base 10 logarithmic scale.

The PCTM can also be connected to SHT85 humidity sensors through I2C serial communication protocol. In Figure 6.3 we can see that two SHT85 inputs are connected to a I2C multiplexer before it is connected to the microcontroller. The two SHT85 have the same I2C address and therefore one can not communicate with the sensors separately only using I2C. An I2C multiplexer is used to first select which channel (Channel 0 to Channel 3) it will connect to first, and then the microcontroller can connect to the device. This will allow us to communicate with each I2C device/sensor separately regardless of what I2C address it has. With the I2C multiplexer the microcontroller can selectively choose between what SHT85 input it wants to read out. In Figure 6.3 we see the NTC1 input is connected to Channel 2 and the SHT85 inputs is connected to Channel 0 and 1. Channel 3 is currently

not used and are connected to the free I2C input. The PCTM also allows a flex cable to connect to both SHT85 as input. SHT85 sensors measures the gas-purge temperature and relative humidity. To calculate the dew point, we use a approximation based on temperature and relative humidity taken from [16]:

$$H = \frac{log_{10}RH - 2}{0.4343} + \frac{17.62 * T}{243.12 + T}$$

$$Dp = \frac{243.12 * H}{17.62 - H} \tag{6.4}$$

RH is the relative humidity, T is the gas-purge temperature and Dp is our calculated dew point.

There are, however some problem with the current design. I2C also requires a pull up resistors, which is not currently included in the design. The pull up resistors have to be manually soldered on the underside of the PCB for each Channel inputs on the I2C multiplexer. A pull up resistor of 3.3KΩ was chosen. This value was the optimal resistance for using I2C Fast mode. Running in I2C in Fast mode means that the I2C will communicate with a speed of 400KHz. Normally the I2C runs at 100KHz. The resistor value was selected to ensure that we can run the I2C at normal mode or fast mode.

The gas-flow input measures the flow of purge gas in L/Minutes. This input is connected to a radiometric gas flow sensor, which outputs a value of 0V to the supply voltage value (Usually 5V). In our testing setup we use a 2L/Minutes radiometric gas-flow sensor. This means the gas-flow will be calculated as:

$$G_{flow} = \frac{V_{flow}}{V_S} * 2 \tag{6.5}$$

where G_{flow} is the gas-flow in L/Minutes, V_{flow} is the output of the gas-flow sensor and V_S is the voltage supply. How microcontroller handles voltage measurements are explained earlier in this section. Using Equation 6.2 (with V_{flow} instead of V_{NTC}) and putting in this Equation we obtain:

$$G_{flow} = V_{ratio} * 2L/Minutes \tag{6.6}$$

Like before the voltage supply is cancelled out and we obtain a nice equation, with only a variable V_{ratio} that can change.

The vacuum chuck temperature is measured from a PT100 element. A PT100 element is a platinum element which acts as a positive temperature thermometer. The resistance of a PT100 element is 100Ω when the temperature is 0°C. PT elements often have small changes when exposed to large change in temperature. This means that we have to have accurate measurement of the PT100 resistance. We use a digitalizing resistance measurement integrated circuit (IC) MAX31865 designed for measuring PT100 elements. In addition 4-sense wire will be used for maximizing accuracy. The MAX31865 communicates with the microcontroller through a SPI communication protocol and output a 15bit digital value representing the ratio between the PT100 resistance and the reference resistance, which is chosen to be 400Ω. This digital value will range from 1 to 32768. The PT100 resistance is then calculated as:

$$R_{PT100} = \frac{IC_{Digital}}{32768} * 400\Omega \tag{6.7}$$

where $IC_{Digital}$ is digital value coming from the MAX31865. To determine the PT100 temperature we use a calculation from [17]:

$$T_{PT100} = \frac{Z_1 + \sqrt{Z_2 + Z_3 * R_{PT100}}}{Z_4} \tag{6.8}$$

where R_0 is the reference resistance and:

$$Z_1 = -A = -3.9083 * 10^{-3}$$

$$Z_2 = A^2 - 4 * B = 17.58480889 * 10^{-6}$$

$$Z_3 = \frac{4 * B}{R_0} = -23.10 * 10^{-9}$$

$$Z_4 = 2 * B = -1.155 * 10^{-6}$$

6.2.2 Polarity Power Supply Polarity Inverter

Most laboratory power supplies does not come with the ability to invert the polarity of its output. A Peltier element cools or heat down a object depending on the polarity of the current that enters the Peltier. Therefore we are required to find a way to invert the polarity that works with all power supplies. The top part of the PCTM shown in Figure 6.2 consist of a circuit which allows the microcontroller to change the polarity at will.

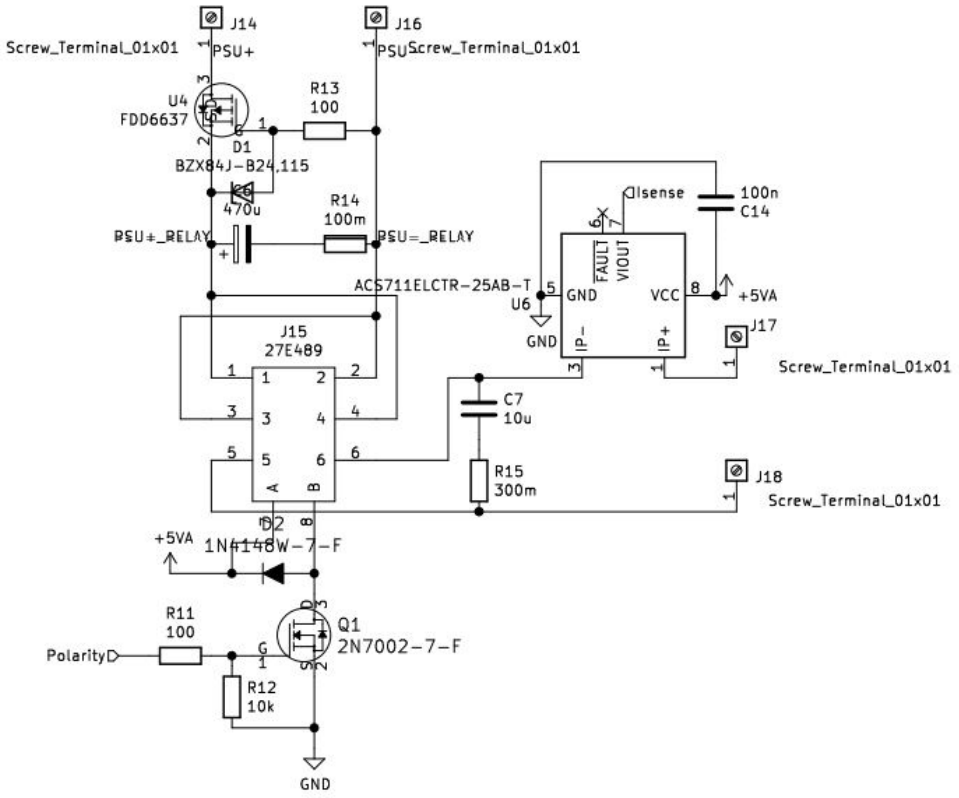


Figure 6.5: A schematic of the polarity inverting part of the PCTM

The way it works is that we have a general purpose relay, the big object at the top of the PCTM (see Figure 6.2), have the ability to change the path of the circuit and thus changing its polarity. This relay can be controlled by the microcontroller, where a logical HIGH signal will activate the relay. The polarity will then be reversed. If sending a logical LOW signal the relay will revert back to its original position and the polarity will be reversed back. Sudden changes to the polarity will cause inductance spikes, which can cause

damage to the power supply. Therefore a part of the circuit is dedicated to dampen the inductance spikes to a safe level.

To measure the current going into the Peltier element, a Hall Effect Linear Current Sensor is connected to the Peltier's negative terminal. The sensor outputs 0V to 5V proportional to the current flow. It can measure both current direction. Thus an output of 2.5V means 0A flowing. The output will be under 2.5V when there is negative current and over 2.5V when positive current is flowing. The sensor that was chosen has an optimized accuracy range of $\pm 25A$, and a sensitivity of 55 mV/A. The sensitivity value is for a voltage supply value of 3.3V. We use a voltage supply of around 5V. Hence we have to scale the sensitivity up by a factor of $\frac{V_S}{3.3V}$, where V_S is our supply voltage. The current can then be calculated as:

$$I_{Peltier} = \frac{3.3V * V_{output}}{55mV/A * V_S} - 2.5V \tag{6.9}$$

where V_{output} is the sensor output and $I_{Peltier}$ is the current flowing into the Peltier element. We can then use Equation 6.2 (with V_{output} instead of V_{NTC}) to calculate the current:

$$I_{Peltier} = \frac{3.3V * V_{ratio}}{55mV/A} - 2.5V \tag{6.10}$$

We can see that the voltage supply is once again cancelled out and we are left with the only changing variable, V_{ratio}. Using Equation 6.10 we obtained a slight deviation at 0A. This is because the sensor output is not exactly 2.5V when 0A is flowing. To fix this, we have to do a small calibration. Equation 6.10 then becomes:

$$I_{Peltier} = \frac{3.3V * V_{ratio}}{55mV/A} - V_{zero}$$

where V_{zero} is the part that needs to be calibrated. V_{zero} is found by calculating $\frac{3.3V * V_{ratio}}{55mV/A}$ at 0A and was calculated to be around 2.560V.

6.2.3 Hardware Interlock Connections

One of the PCTM feature is that it can connect to a hardware interlock without affecting it. The PCTM will "silently" receive both module temperature and relative humidity from the interlock. Reason for this is because we want to measure the same data coming from the hardware interlock. As said before in section 6.2.1, the PCTM receives the module temperature via I2C communication from the hardware interlock, but it can also receive data from the SHT85 connected to the hardware interlock. To achieve that we implemented a I2C sniffer to sniff the data packet between the hardware interlock and the SHT85 sensor. The PCTM can also receive power from the hardware interlock.

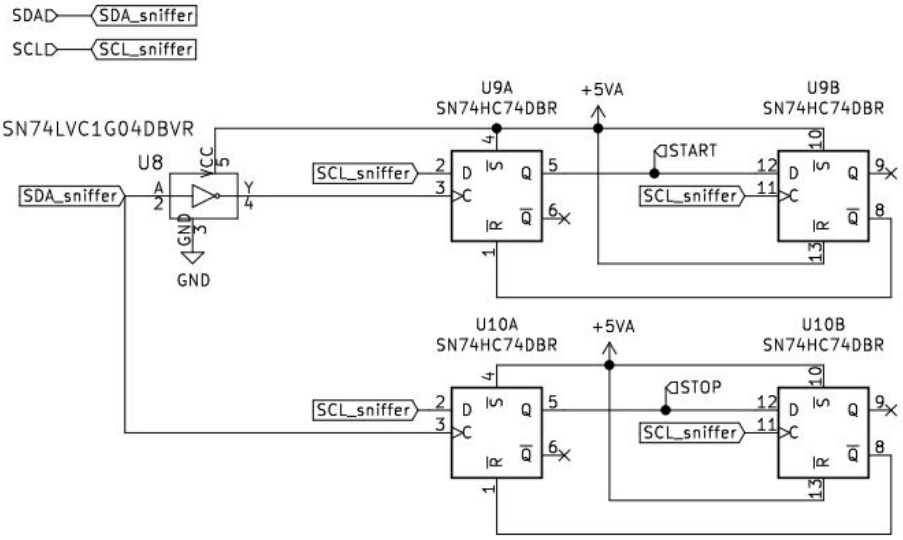


Figure 6.6: A schematic of the I2C sniffer part of the PCTM

The PCTM has the ability to sense that it is connected to a hardware interlock. When the PCTM is connected to a hardware interlock, it will go to a hardware interlock mode. In this mode the PCTM will immediately stop communicating with the first SHT85 connection and start the I2C sniffer.

The I2C protocol is initiated by a start condition. The start condition is defined as when the SDA (data line) is transitioning from HIGH to LOW while the SCL (clock line) is HIGH. Likewise a I2C protocol is ended when the SDA is transitioning from LOW to HIGH while SCL is HIGH. This defines the stop condition. After a start condition a 7Bit address is sent followed by

a single Write/Read bit. To be able to monitor the I2C communication, one would need to identify the start and stop conditions.

The design for generating the start and stop condition are taken from [18]. It uses a dual D-type positive edged flip flops to generate a start/stop signal whenever a start or a stop condition has occurred. From Figure 6.6 we can see that two dual D-type flip flops is used, one for each condition. The start and stop signal are connected to the D2 and D3 pins on the microcontroller. When a start or stop condition has occurred, the start/stop signal will transition from LOW to HIGH. After the PCTM is connected to a hardware interlock it will start a hardware interrupt on the start signal. A hardware interrupt is a routine where the microprocessor will stop whatever it is doing and execute a another function instead called Interrupt Service Routine (ISR). After an ISR, the microprocessor will return to whatever it is doing. The hardware interrupt on the start signal will start another hardware interrupt on the SCL line and on the stop signal. Whenever the SCL line goes HIGH a hardware interrupt is activated and SDA line is sampled. The hardware interrupt on the stop signal will deactivate the hardware interrupt on the start signal, on the stop signal and on the SCL line. The sampled SDA data is then analyzed afterwards. There are two reason for this. The first is that each ISR much take as little time to execute as possible. Other parts of the microprocessor can not happen while in a hardware interrupts. The second reason is that the I2C standard speed is at 100KHz. This means that the SCL line is going HIGH every $10\mu s$. An ISR can not be longer than $10\mu s$ otherwise we will lose the data.

Most of the problem with the I2C sniffer is to make the ISR as short as possible. The Arduino compiler has special functions to make ISR (attachInterrupt), but these functions goes through extra instruction to execute an ISR. According to [19] it will take $2.9375\mu s$ to start an ISR and $2.1875\mu s$ to leave it. The SDA line changes (from HIGH to LOW or vice versa) only when the SCL is LOW. Since the SCL line on a 100KHz clock speed is only HIGH for $5\mu s$, we are left with only around $2\mu s$ to sample the data. There is not enough time to sample the data. Luckily there is a different method of defining an ISR. This use a more "low level" method to create ISR, but are much faster. Using the other method it will take $1.4375\mu s$ to enter an ISR and to $1.1875\mu s$ to exit. This will leave us enough time to sample the data and leave the ISR before $10\mu s$. The Arduino uses a function called digitalRead to read what state the pins are on. Unfortunately this functions takes about $5\mu s$ to execute, which means this function can not be used to read out the SDA line. A another method is used, by simply accessing the

register that holds the state of the pin the SDA line is connected to. The microprocessor to the Arduino Nano has different registers, one of them are the registers that holds the information of what state the pins are. This registers are in binary form and we can use bitwise operations on the registers to find the state of the SDA. Using this method we reduced the time to read out the pin to around $1\mu s$. Another problem with the ISR is that each hardware interrupt has a priority order. The hardware interrupt on the pins are on a lower priority. This means that sometime, an another interrupt can trigger before the hardware interrupt on the SCL line. This will delay the ISR on the SCL line. This is easily fixed as the microprocessor has a special flag one can set on our hardware interrupt. This flag will increase the priority the hardware interrupt to another level of priority. This level is on the top of the priority order and can even interrupt other hardware interrupts.

6.2.4 Python GUI

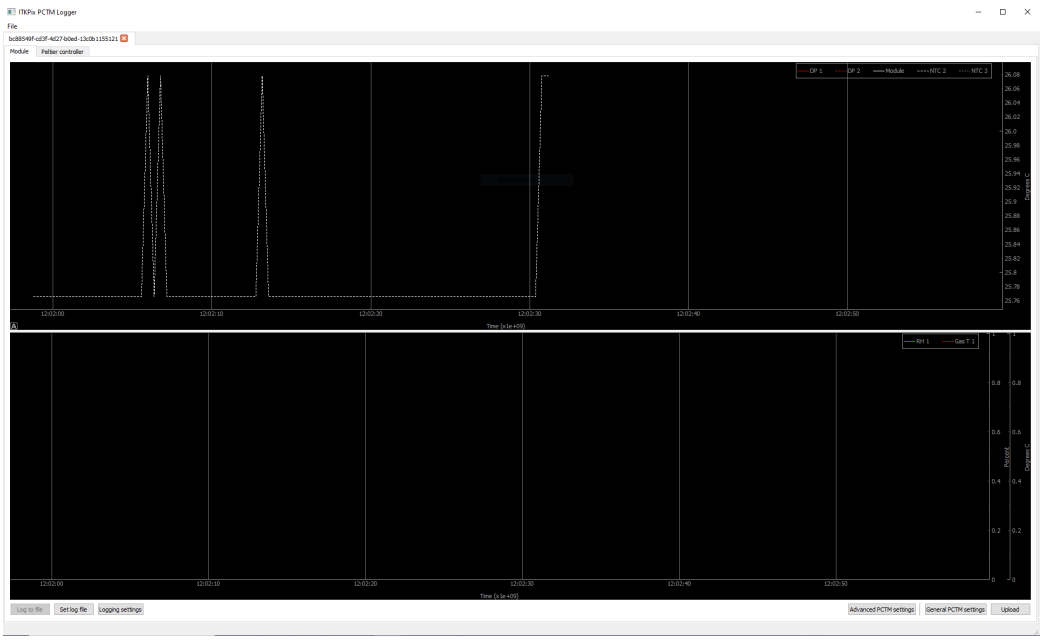


Figure 6.7: A screenshot of the Python GUI. The screenshot shows two graph. The top graph is used to monitor the temperature and dew point. The bottom graph shows the relative humidity. On the bottom there are buttons to configure the PCTM such as sampling rate. The GUI also has two tabs, where the second tab shows the settings and graphs used in the Peltier controller part of the PCTM.

A Python GUI was created to control and read out the PCTM data. The GUI includes options to control the sampling rate for the different parameters monitored and to control the Peltier controller part. The GUI can upload the different PID parameters and also setting the PID controller to manual or automatic. It also includes real-time graphing of the different parameters and can save the data in a HDF5 format. The GUI is made in mind that multiple PCTM can be connected to the same PC. Therefore it can also set the ID display, so each PCTM can be identified separately. A overview of the GUI can be seen on Figure 6.7.

6.2.5 Peltier Controller

As mentioned before in section 6.2 the temperature to the module is controlled by a Peltier element. This is achieved by using a Proportional-Integral-Derivative loop (PID loop). A PID loop is a control loop that calculates a correction based on the error, a proportional term, an integral term and a derivative term. The error is defined as the difference between a desired set-point (SP) and the measured process variable (PV). This correction will hopefully minimize the error, which means that our PV will be getting closer and closer to the SP. The algorithm for the PID loop is given as:

$$CO(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{6.11}$$

where $CO(t)$ (Controller Output) is the output of the PID loop, K_p is the proportional term, K_i is the integral term, K_d is the derivative term and $e(t)$ is the error. The proportional, integral, derivative part must be tuned for the process one would like to control. To find the optimal value one must be able to tune the PID. There are different methods of tuning PID which are based on several factors.

To tune PID loops one will have to first identify what kind of process we want to control. Different type of processes requires different methods to tune them. There are three types of processes. The first process is called self-regulating process (also called non-integrating process). These types of process are self-regulating in a way that for a change in CO, the PV will stabilise at a fixed value. The second process is called an integrating process. An integrating process PV does not stabilise at a fixed value, but will instead increase linearly for a change in CO. These processes are therefore also

called non self-regulating processes. There are another process called near-integrating process. These processes are self-regulating process, but with a long time constant. The time constant is long enough that it will begin to behave more like a integrating process than a self-regulating process. The third process is called a runaway process. These processes are like integrating process, but instead of the PV increasing linearly with a change in CO, the PV will accelerate. Thus the process will runaway and go out of control.

Our test setup uses a Peaktech P1535 power supply which has a 5V ratiometric input. The power supply is used to control the Peltier elements. It has a range of 1V-32V and 0A-20A. The power supply is connected to the PCTM and are controlled using a pin which has Pulse Width Modulation (PWM). The Arduino controls the voltage level of the pin using a value of 0 - 255. This value controls the duty cycle to the PWM and thus the pin is able to output a value between 0V - 5V. During testing and tuning the PID a dummy module was used and was used on a assembled cooling unit[3]. A Julabo Circulating Chiller is connected to the cooling unit and was set to -10°C during testing and tuning.

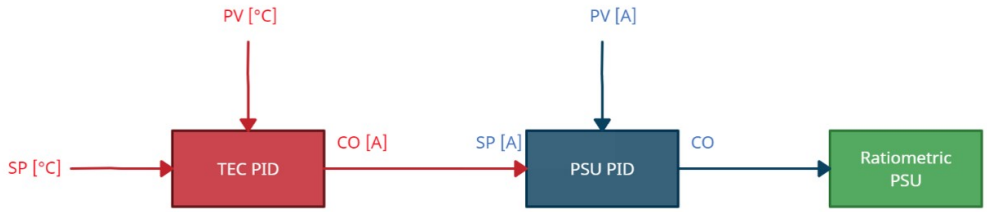


Figure 6.8: A diagram of the PID structure to the PCTM

It is mentioned in Section 6.2.2 that the current to the Peltier is measured. We can use this to implement a PID loop on the power supply. The motivation for also implementing a PID loop on the current is that ratiometric power supplies are often inaccurate and difficult to control with the ratiometric mode. The PID loop will ensure that we can set the current going to the Peltier more accurately and also the PID will help stabilise the current. This means that we will have two PID loops, one for the power supply and one for controlling the temperature. Figure 6.8 shows a diagram of how the PID are connected together. The PID loop for the power supply is called PSU PID and the PID loop for control of Peltier is called TEC PID. The CO to the TEC PID is connected to SP to PSU PID. This means that

[3]Details about the cooling unit and overall setup can be found on section 6.5

the TEC PID loop CO will be a variable with a unit of Ampere. The CO of the TEC PID will be the set-point to the PSU PID. Thus the current the Peltier receives will be the output of the TEC PID loop, and the PSU PID is there to ensure that the power supply output the right value.

To identify and tune the process a step response is used. The step response procedure is simple. One will need to make a change in CO, and observe how the PV changes. Using a step response one can find the time constant and the dead time. If the time constant is larger than 4 times the dead time, it is usually considered to be a near-integrating processes and has to be tuned as an integrating one. The time constant is defined as the time the PV takes to reach 0.63 of its total PV change. The dead time is defined as the time between the change in CO and when the PV is reacting. The dead time and the time constant will be used to tune the PID.

Dead Time [s]	0.649
Time Constant [s]	0.137
Change in PV [A]	7.4492
Change in CO	99

Table 6.1: Results from a step respons on PSU PID

Doing a step response on the PSU PID we found that the dead time was several times larger than the time constant. In fact from Table 6.1, we can see that the dead time is around 6 times larger than the time constant. Regular tuning rules like Ziegler-Nichols has poor performance on dead time dominating process. Some modification must be done to the Ziegler-Nichols rules. We use a dead time dominating tuning rule from [20] seen in Table 6.2. In Table 6.2 the gp is the process gain and are defined as $gp = \frac{\Delta PV}{\Delta CO}$. The SM part the stability margin and is a detuning variable. It is there to ensure stability and can be any value from 1 to 4. We can use the results from Table 6.1 and tuning rules from Table 6.2 to tune our PID loop. The tuning variables that are given is controller gain, integral time and derivative time. To convert it back to the parameters we use these relations:

$$\begin{aligned} Kc &= Kp \\ Ki &= \frac{Kc}{Ti} \\ Kd &= Kc * Td \end{aligned} \tag{6.12}$$

The results from the tuning can be found on Table 6.5

Kc	0.36 / (gp * SM)
Ti	td / 3
Td	0

Table 6.2: Dead time dominating tuning rules from [20]. gp is the process gain and is defined as $gp = \frac{\Delta PV}{\Delta CO}$. SM is the stability margin.

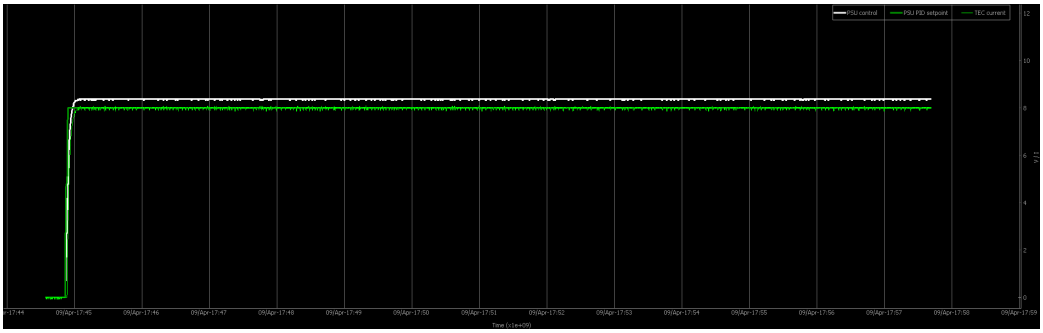


Figure 6.9: Results from tuning the PSU PID with a setpoint manually set to 8A. The light green line is the setpoint and the dark green line is the measured current. The x-axis is the time in minutes and is in the range from 17:45 to 17:59. The y-axis is current and is in the range from 0A to 12A

Figure 6.9 shows the results from the tuning the PSU PID. The setpoint was set to 8A and we can see that the current takes around 1s to reach 8A and stabilise at that value. There are no undershoots or overshoots. The tuning works really well. Figure 6.10 shows that the tuning works also when the TEC PID is controlling the setpoint. We can see that the light green line (measured current) is following the setpoint very closely. It also exhibit no undershoots or overshoots.

A step response on the TEC PID shows a very large time constant. In fact it is more than 4 times as large as the dead time. This means that TEC PID processes is a near-integrating process and will have to be tuned as an integrating process. A procedure to tune an integrating process is to wait until the process is stable, then make a step change in CO and wait until a visible slope can be calculated. Afterwards change the CO back to its original value and wait until a steady slope. The dead time is found by finding the time between the step change and where the two slopes intersect. We will then calculate the difference between the first and the second slope and the process integration rate:

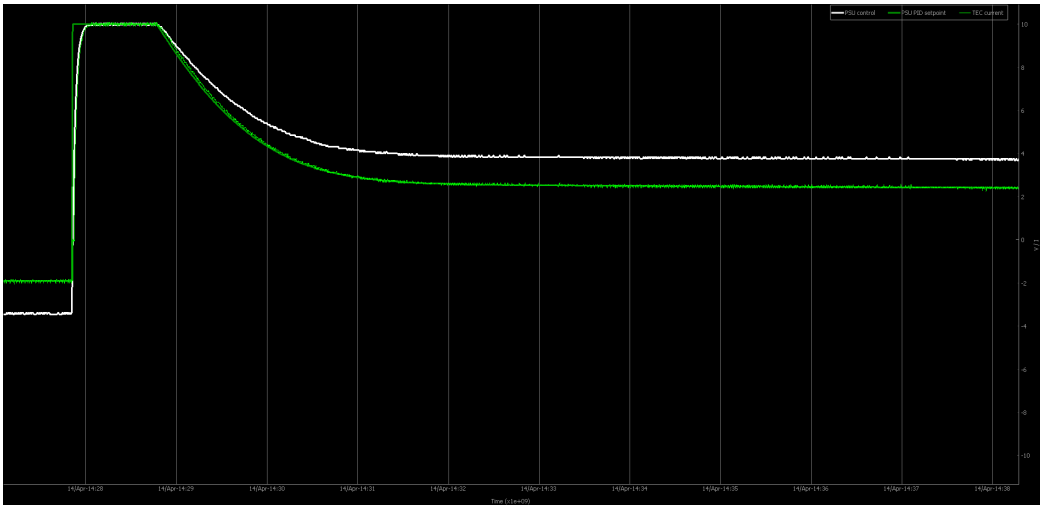


Figure 6.10: Results from tuning the PSU PID with the TEC PID controlling the setpoint. The light green line is the setpoint and the dark green line is the measured current. The x-axis is the time in minutes and is in the range from 14:25 to 14:38. The y-axis is current and is in the range from -10A to 10A

$$\begin{aligned} DS =& Slope_1 - Slope_2 \\ ri =& \frac{DS}{\Delta CO} \end{aligned} \tag{6.13}$$

where ri is the integration rate and ΔCO is the step change done during the procedure. The procedure used is described in [21]. While tuning the PID it was discovered that the process behaves differently when cooling and heating the Peltier. Therefore two set of tuning was done, one for cooling and one for heating. The results from the procedure can be found in Table 6.3. We can use the results to calculate the tuning constants using the tuning rules from Table 6.4. The tuning constants can then be converted back to the constant we use by using Equation 6.12. The results can be found on Table 6.5.

	Slope Difference	Integration Rate	Dead Time [s]
TEC PID (Cooling)	0.134	0.067	21
TEC PID (Heating)	0.19	0.094	26

Table 6.3: Results from doing the procedure described in [21].

Kc	0.9 / (SM x ri x td)
Ti	3.33 x SM x td
Td	0

Table 6.4: Tuning rules for integrating processes. SM is the stability margin. ri is the process integral rate. td is the dead time.

	Kp	Ki	Kd
PSU PID	2.4	10.91	0
TEC PID (Cooling)	0.32	0.002	0
TEC PID (Heating)	0.182	0.004	0

Table 6.5: Results from tuning the PSU PID and the TEC PID. A stability margin value of 2 was chosen for all results.

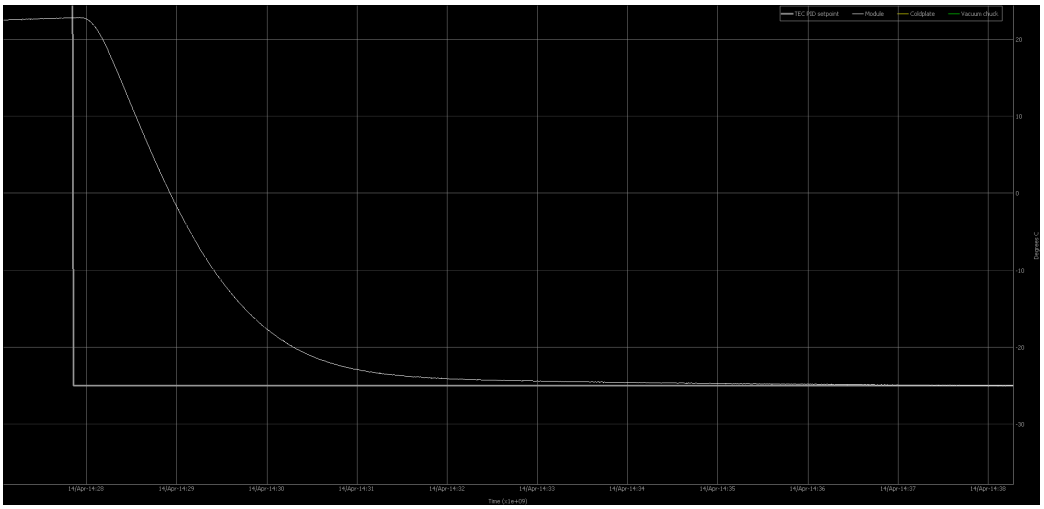


Figure 6.11: Results from tuning the TEC PID. The graph starts at 23°C and the setpoint was set to -25°C. The white line is the module temperature and the grey line is the setpoint. The x-axis is the time in minutes and is in the range from 14:28 to 14:38. The y-axis is module temperature in Celsius and is in the range from -30°C to 20°C

Figure 6.11 shows the module temperature starting from around 23°C. The setpoint was set to -25°C. It uses the tuning constants obtain from Table 6.5 (TEC PID Cooling). We can see that the temperature will first decrease quite fast before reaching around -24°C. From there it will take a little while to settle down to -25°C. No overshoot or undershoot was observed. The ramp rate was around 20°C/min, which is according to Section 6.1 is too

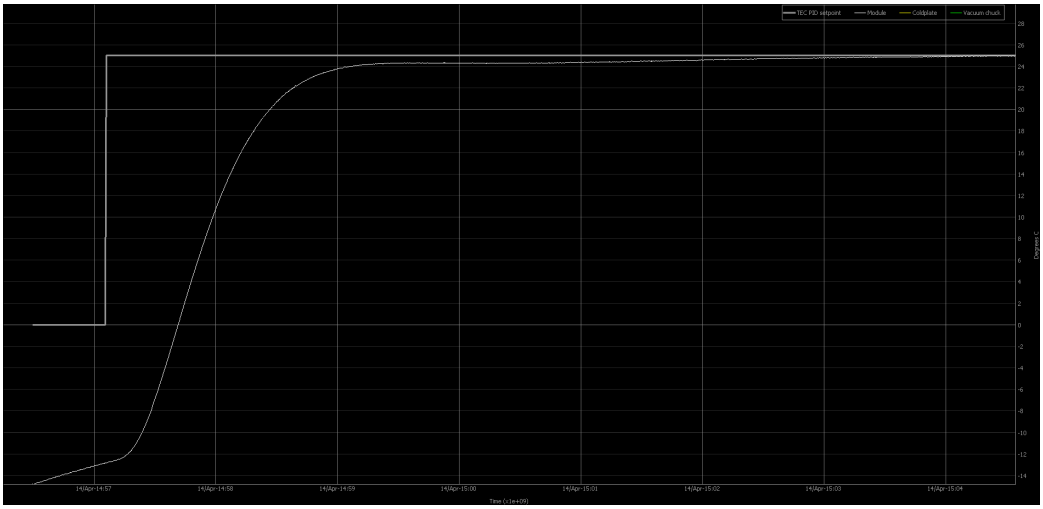


Figure 6.12: Results from tuning the TEC PID. The graph starts at -13°C and the setpoint was set to 25°C. The white line is the module temperature and the grey line is the setpoint. The x-axis is the time in minutes and is in the range from 14:57 to 15:04. The y-axis is module temperature in Celsius and is in the range from -14°C to 28°C

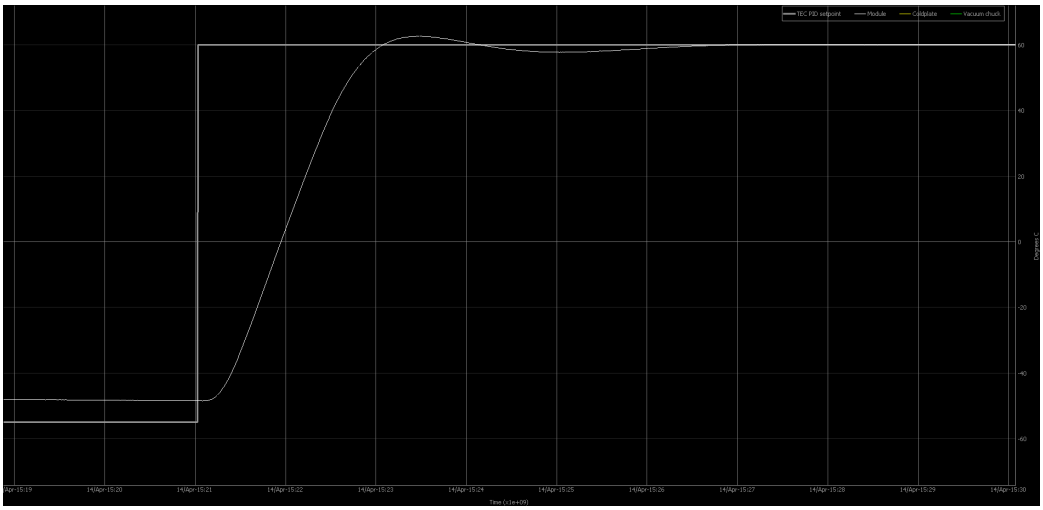


Figure 6.13: Results from tuning the TEC PID. The graph starts at -48°C and the setpoint was set to 60°C. The white line is the module temperature and the grey line is the setpoint. The x-axis is the time in minutes and is in the range from 15:19 to 15:30. The y-axis is module temperature in Celsius and is in the range from -60°C to 60°C

high. Figure 6.12 shows a graph of the module temperature starting from -13°C, with a setpoint of 25°C. Tuning parameter used was the heating one from Table 6.5. It behaves somewhat like Figure 6.11, where it will slow down at 24°C before taking some time settling to 25°C. There are also no overshoot or undershoot. The ramp rate was calculated to be around 30°C/min which is again, too high. Figure 6.13 shows the results from the tuning (TEC PID (Heating) values). The module temperature starts at -48°C and the setpoint was set to 60°C. Here it shows an overshoot of about 3°C, before settling at 60°C. This time the ramp rate was 80°C, which is way over the recommended limit. Overall the tuning was successful and the results were good, but the ramp rate has to be fixed. This can be done by simply ramping the setpoint. This way the module temperature can not exceed the setpoint ramp rate.

6.2.6 Problems with PCTM

There are several design problems with the PCTM. This includes the missing pull up resistors required by the I2C protocol (See Section 6.2.1). There are other problems with the I2C multiplexer. The different channels of the I2C multiplexer should be separate from each other, but it turns out it has a small connection between each other. The PCTM also has a regular multiplexer (MCP23017), as there are not enough pins on the Arduino Nano. The MCP23017 also uses I2C communication. The small connection from the I2C multiplexer causes the SCL and SDA line to get pull down a small amount. In fact it will sometime pull the lines below 4V. According to the MCP23017 datasheet [22] the multiplexer HIGH requirements are 4V. This means that if the SCL or the SDA goes below 4V, the MCP23017 do not know what state it is in. This causes the microcontroller to fail. This is fixed by adding $100pF$ capacitors to the SDA and SCL lines and adding extra $3.3K\Omega$ resistors. An another problems is that pin A7, which controls the ratiometric power supply is not a PWM pin. Hence, we have to add a botch wire connecting pin A7 to a another unused pin which has PWM. This pin was chosen to be pin D9.

The tuning process for the TEC PID loops has also been problematic. There are many tuning rules and methods for tuning PID. A lot of them are good for certain processes and are bad for others. Most of the time was spent looking for good tuning rules that eliminates overshoots and undershoots. Figure 6.14 shows one of the earlier attempt to tune the TEC PID. A different kind of tuning rules was used, called lambda tuning. The graph exhibit some

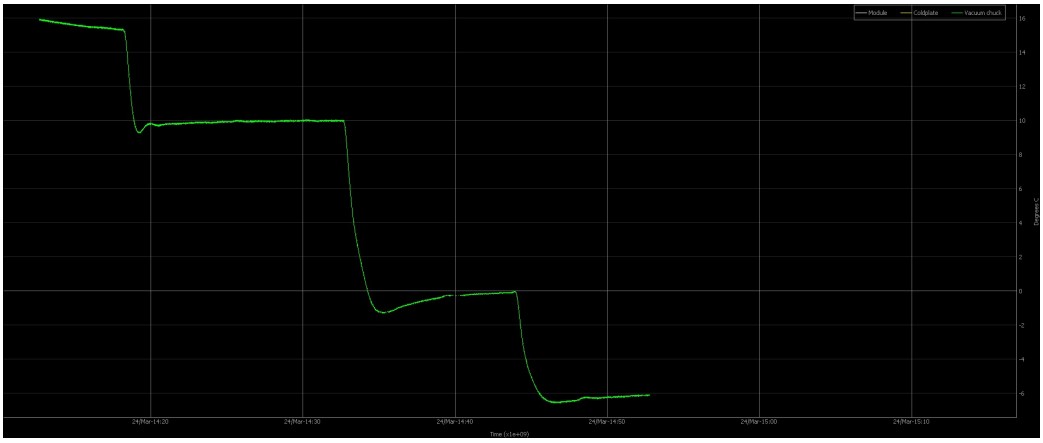


Figure 6.14: Earlier results from tuning the TEC PID. A method called lambda tuning rules was used. The green line is the module temperature despite it says Vacuum chuck. The x-axis is the time in minutes and is in the range from 14:20 to 15:10. The y-axis is module temperature in Celsius and is in the range from -6°C to 16°C

overshoot of around 1.5°C at 14:35. This is not a big overshoot, but we are only stepping 10°C. With lambda tuning the overshoot grows bigger as the setpoint is stepped larger. This is problematic as in thermal cycling of the module, we will have to step as large as 115°C. The process was also not correctly identified as near-integrating process. This causes a much longer time to tune the PID as we will have to wait for the processes to settle. The time constant can be as high as 20 minutes. This tuning rules should be good for both integrating and self-regulating processes, but we was unsuccessful to eliminate the overshoot with the tuning rule. Instead a better tuning methods was found, which uses a modified version of Ziegler-Nichols rules made for integrating processes. Correctly identifying the processes not only made it faster to tune, but also the tuning rules works much better.

6.3 Stage 1 Qualification Test

To ensure that the different testing sites are able to test the ITk pixel the sites have to go through a qualification test. The qualification test is occurred in three stages. Bergen has currently managed to successfully pass the stage 1 qualification test. Simon Huiberts has done most of the work, which includes reading out the module (tuning and scans) and using the local database. Graham Lee did the visual inspection part. Wai Chun Leung helped with the

monitoring and controlling the temperature. In this section details about the qualification test will be explained and results from the test will be showed. The stage 1 qualification test requirements are:

- Use a digital module
- Some visual inspection
- Readout of NTC module
- Keep the module below 55°C with passive cooling (or active, if necessary)
- Probing and trimming VDDA/D to 1.2V
- Full tuning and scans using 3 single readout adapters
- Usage of local database

We received a D4 digital ring triplet module to qualify for the stage 1 test. The ring triplet consist of three chip, in a ring configuration. The cooling unit in Bergen has only vacuum chuck for linear triplet modules. This means that active cooling will be difficult. Since the module only needs to be kept under 55°C, passive cooling through the circulating chiller will be enough. The ring triplet does not fit with the vacuum chuck. The module carrier is instead placed on top of the vacuum chuck. Vacuum is applied to the carrier and the vacuum chuck. The temperature are read out from the NTC data coming from the flex cable. The flex cable are connected to the adapter cards. From the adapter cards the three NTC output coming out of the module is read out by the PCTM. The NTC from chip 3 is connected to the hardware interlock. NTC from chip 2 is connected to the NTC2 input and NTC from chip 1 is connected to the NTC3 input of the PCTM. The circulating chiller was set to 0°C. Dry air was applied to the module with a flow rate of around $1L/min$.

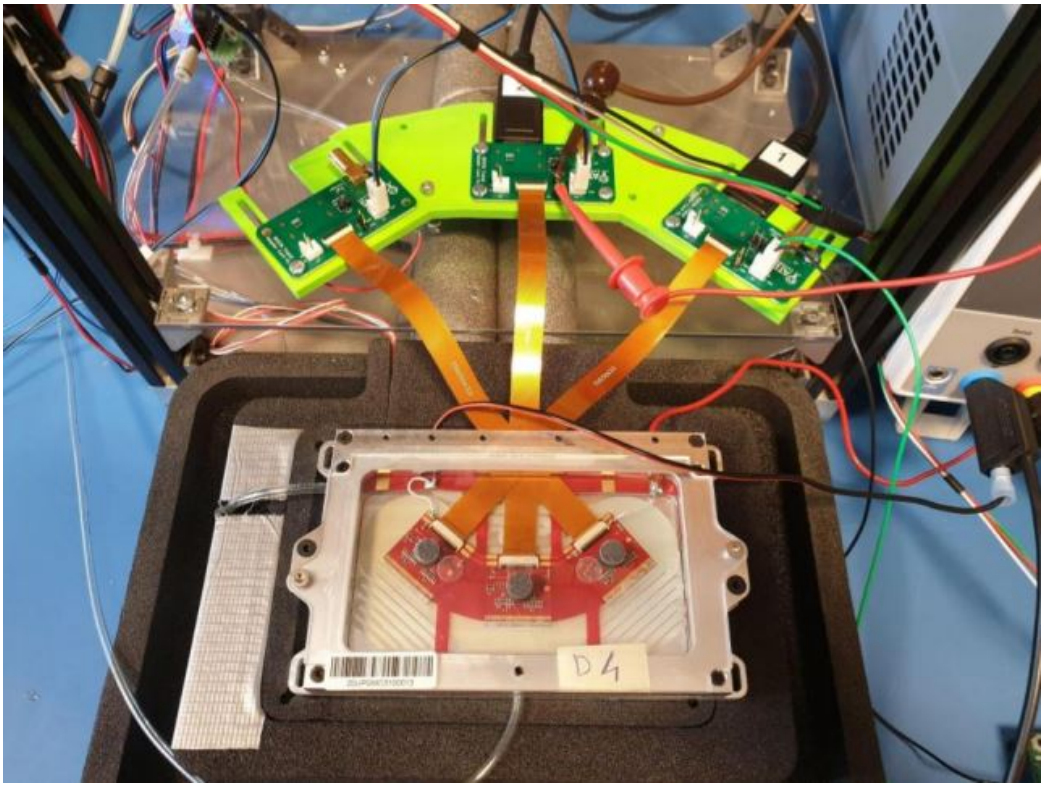


Figure 6.15: A picture of the ring triplet module sitting on the cooling unit. The module carrier is sitting on top of the vacuum chuck. The dry air can be seen coming in from the left. The vacuum are supplied from the front. Both the module carrier and the vacuum chuck are supplied with vacuum.

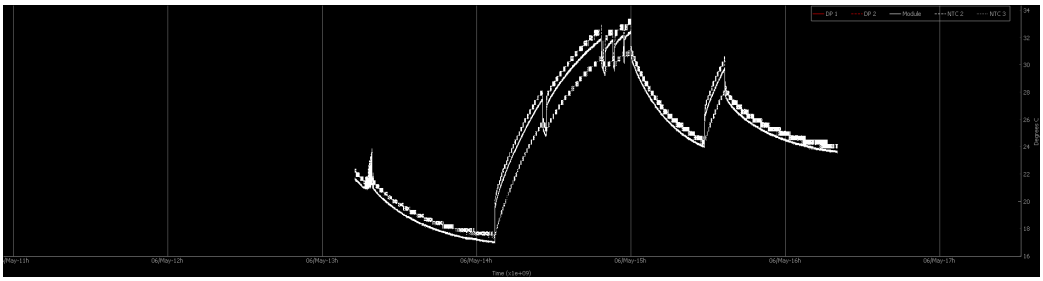


Figure 6.16: A graph of the three NTC temperature during the stage 1 qualification test. The x-axis is the time in hours and is in the range from 11:00 to 17:00. The y-axis is module temperature in Celsius and is in the range from 16°C to 34°C. The graph shows the temperature during the VI-scan and the trimming procedure.

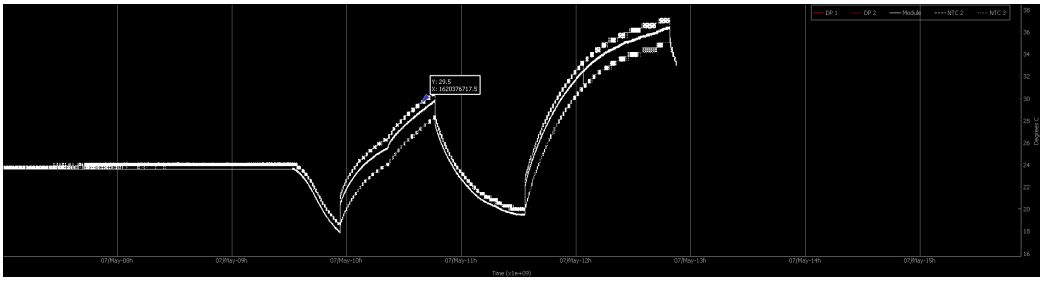
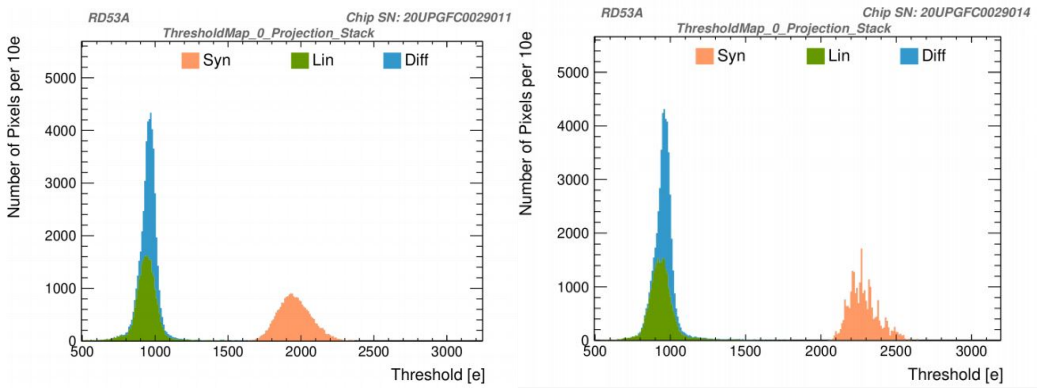


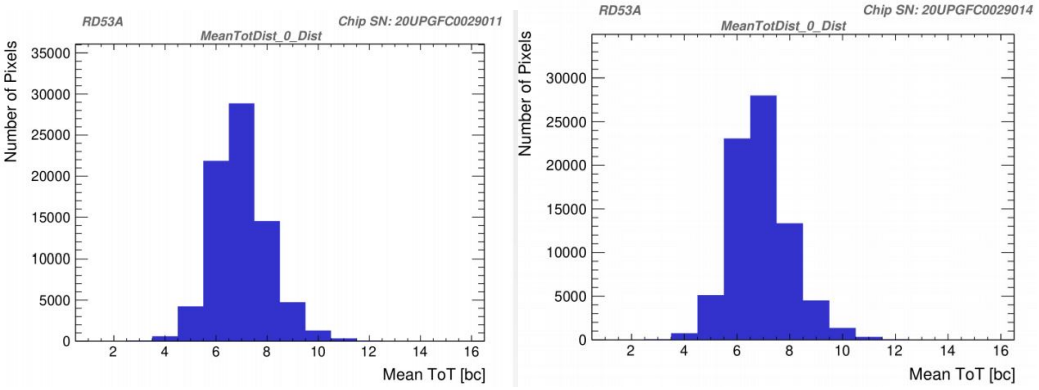
Figure 6.17: A graph of the three NTC temperature during the stage 1 qualification test. The x-axis is the time in hours and is in the range from 08:00 to 15:00. The y-axis is module temperature in Celsius and is in the range from 16°C to 38°C. The graph shows the temperature during the tuning and communication trials.

Figure 6.16 and 6.17 shows the temperature during the stage 1 qualification. Readout of the NTC module is therefore successful. The small spike in the beginning in Figure 6.16 is the temperature during the VI-scan. Afterwards the graph shows the temperature during trimming of the chips. Tuning and communication of the chip was done the day after the VI-scan. The temperature during the tuning can be seen in Figure 6.17. We can see that the whole time the temperature never went above 38°C. Thus the requirement of keeping the module temperature under 55°C is fulfilled.



(a) Chip Nr: 3 **(b)** Chip Nr: 2

Figure 6.18: Threshold tuning results of chip 2 and 3.



(a) Chip Nr: 3 **(b)** Chip Nr: 2

Figure 6.19: ToT tuning results of chip 2 and 3.

The D4 ring triplet had the chip 1 disconnected because of a short circuit. Thus the module only had two working chips: chip 2 and 3. Simon Huiberts managed to communicate with the two chips and successfully managed to tune the chips. The target threshold for the synchronous front end was set to 2000e and 1000e for the differential and linear front ends. The synchronous threshold target was set to higher, because of earlier reported issue with the front end. The threshold target was successfully tuned on chip 3 for all front end. The synchronous threshold target did not reach 2000e for chip 2. Results can be found on Figure 6.18. The ToT (Time over Threshold) was successfully tuned on both chips. The target was 7 for 10Ke. The results can be seen on Figure 6.19.

Visual inspection of the module was done by Graham Lee. The inspection was done using EPSON Perfection 0600 photo scanner and Olympus microscope with Canon EOS MK II camera. The images was taken manually and stitched together using a image stitching software. Visual inspection of the wirebonds can be seen on Figure 6.20.

Figure 6.20: A picture of the wirebonds on the module chips. This was done using a Olympus microscope with Canon EOS MK II camera.

6.4 Hardware Interlock

During module testing the modules have to be powered and cooled in a dry atmosphere. Modules can be damaged during testing. Damages such as bump bond disconnect caused by excessive temperature swings, corrosion caused by high dew point and front end damage caused by overvoltage or overcurrent. DCS (Digital Control System) should be able to monitor the module during testing and prevent damages to happen. A DCS is a software interlock. However, software interlock are complex and cannot always be relied upon. There are many causes for failure for instance, crashes or halting and loss of monitoring or control of power supplies.

A hardware interlock is used as a safety net in case the DCS system fails to protect the modules. The hardware interlock should monitor a few important parameters, has least amount of components to function and has minimal possible reliance on digital logic. The limits on the hardware interlock as to be higher than the DCS system, as the DCS must be able to react before the hardware interlock.

Magne Eik Lauritzen designed and developed the hardware interlock, which will be used in module testing. The hardware interlock consist of three parts: A trigger board, a logic board and power relay boards. Currently two version of the trigger board has been developed and tested and one version of the logic board. The hardware interlock was populated and tested by Wai Chun Leung.

6.4.1 Trigger Board V0.1

The trigger board monitors different parameters described in Table 6.6. When a parameters exceed the limit, a trigger will be sent to the logic board. The interlock signal will then go from the logic board to different relay boards to cut off e.g power to the module or Peltier. The trigger board uses a reverse logic, where LOW signal means a trigger signal and a HIGH signal is when the parameters are under the limits.

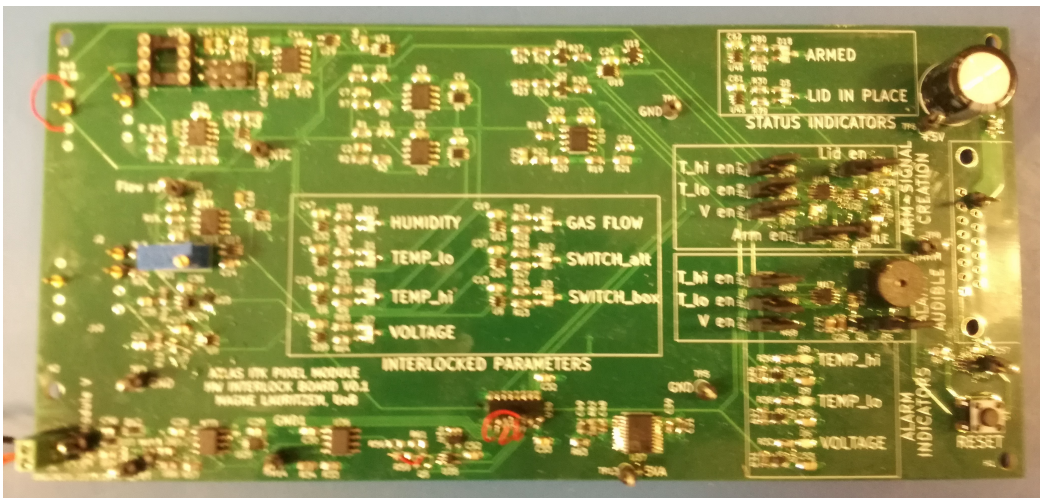


Figure 6.21: A picture of a Trigger Board V0.1.

The hardware interlock has an input protection part where the input current from the module is limited. The purpose of this is to prevent damage to components. Unfortunately the design of the input protection is not working as intended. The design should clamp the input voltage between 5V and Ground, but was unable to do so. The voltage is monitored through the use of voltage comparators. The comparator compares the voltage of 1.8V and the input voltage, and will trigger a LOW signal if input voltage goes over 1.8V. The 1.8V is set by using simple voltage divider. The reference

Parameter	Limits
Module Voltage	1.8V
Module Temperature	-60°C, +65°C
Relative Humididty	30%
Dry Air Flow	Set by user
Enclosure switch	Open (normally closed)
Alternate switch	Optional

Table 6.6: A table of the different parameters and its limits monitored by the trigger board V.01

voltage should be 5V, but it was measured to be around 4.6V. This means that the limit set by the voltage divider was lower than expected. Except for the small error on the reference voltage, the voltage comparator parts works. The voltage comparator parts have its own ground plane. This means a digital isolator has to be used, connecting the two grounds. This digital isolator was not correctly connected and had to be reconnected.

The humidity part uses a ATtiny 85 8-bit microcontroller to monitor the relative humidity. The microcontroller will send out a 0V-5V signal, which will then be compared with a comparator. This was not tested on trigger board V0.1, but tested in V0.3.

The temperature is monitored nearly the same way as the voltage. Comparators are used, and the NTC inputs are connected to the comparators. Two comparators were used, one for high temperature limit and one for the low temperature limit. The limits are defined in Table 6.6. The temperature signal is first conditioned by a linearisation network. The NTC temperature will then be a voltage that can be compared. To test the circuit a potentiometer was used to simulate the NTC resistance. The temperature comparators were found to be working as intended. The trigger board trigged on limits over 65°C and under -60°C.

The monitoring of the box switch is rather simple. If the box switch to the enclosure is disconnected, the input will be pulled to ground and will trigger a signal. A Schmitt trigger is used to get a clean signal. The box switch was tested to trigger successfully.

The air flow part should be connected to a ratiometric gas flow sensor. The input will then be a voltage that can be compared. Since different kind of ratiometric sensor can be used the limit can be manually set by using a

potentiometer. The compared voltage is again set by using a voltage divider. The potentiometer is a part of the voltage divider and thus one can freely set the compared voltage. This was also tested and found to be working.

The trigger board has a latching system. If any parameters goes above the limit and the board is armed, the board will latch. This means that the trigger board will stay triggered, until it is disarmed. The trigger board is disarmed by pressing the reset button. The trigger board will not trigger unless it is armed. The board uses a self arming system. If the temperature goes under 15°C, above 30°C or the voltage goes above 0.51V, the trigger board will self arm. The self arming of the different parameters can be turned of by removing the jumpers connected to each parameter. If the parameters goes beyond the self arming limit, and the trigger board is not armed, an audible alarm will sound. This can, like the self arming, be disable at will for each parameter by removing some jumpers. The self arming system proved to be problematic, since it self does not have a latch system. Thus the trigger board will self arm, but not stay armed in some cases. For example, if the voltage goes above the limit 1.8V. The trigger board will self arm, since it is above 0.51V. The trigger from the trigger board should go to the relay board, which would break the power to the module. This of course means that the voltage will drop down to 0V. Now the trigger board will turn off the self arming. This will also turn off the latch. In this case the trigger signal will disappear and the power to the module will be turned back on. Then the voltage will go above 1.8V again, thus power to the module will switch off and on indefinitely. The self arming system of the trigger board is therefore not functional and must be scrapped or redesigned.

Indicator is also on the trigger board (see Figure 6.21). On the bottom right, the lights indicate if the temperature are under 15°C and above 30°C and the voltage is above 0.51V. The lights are usually green if everything is normal, but will light red if triggered. On the top right light indicate if the trigger board are armed or not and if the lid is open. The light in the middle of the board indicate if any parameter monitored has triggered. All the indicators on the trigger board V0.1 is functional.

6.4.2 Trigger Board V0.3

Trigger Board V0.3 has many improvements and fixes a lot of the problem present in version 0.1. The board has been significantly reduced in size. The new version of the trigger board has multiple new functions:

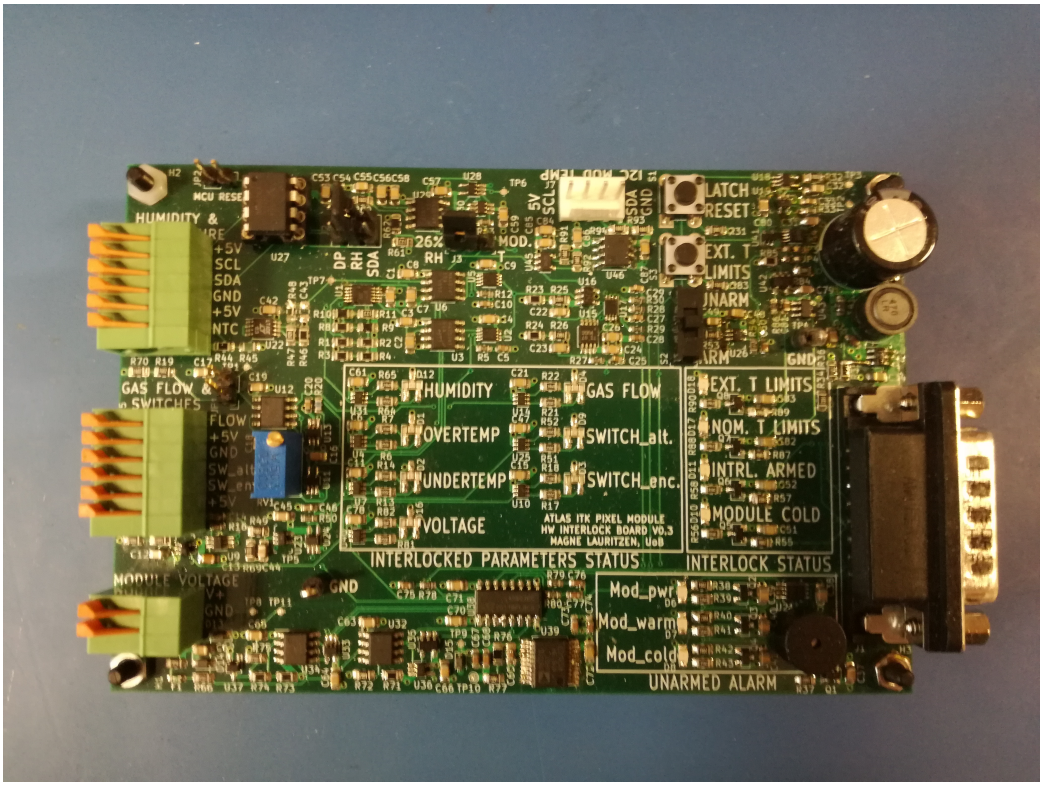


Figure 6.22: A picture of a Trigger Board V0.3.

- The ability to switch the temperature limits
- The self arming has been removed and replaced by a simple arming switch
- A circuit to digitize the voltage value from the NTC. It can then be used to connect to a DCS system (E.g PCTM) through I2C communication
- Voltage regulator. The trigger board will be supplied with 7V-15V and then regulated to 5V
- Limits have been changed slightly

The input protection part is fixed in this version. The voltage comparator are also working as intended. The voltage limit has been changed to 1.943V from 1.8V. Gas flow part has not been changed and thus are working as designed. All the indicators on the board are also working. The alarm is also functioning.

Parameter	Limits
Module Voltage	1.943V
Extended Module Temperature	-60°C, +65°C
Nominal Module Temperature	-40°C, +40°C
Relative Humididty	26%
Dry Air Flow	Set by user
Enclosure switch	Open (normally closed)
Alternate switch	Optional

Table 6.7: A table of the different parameters and its limits monitored by the trigger board V0.3

The ability to change the temperature limit is done by pressing a button. This will change the temperature limit between nominal and extended limits. A jumper configuration has to be done first. The temperature limits is tested to change successfully, but the indicator that indicate what temperature limit the trigger board is currently operating on shows the wrong limit. When nominal limit are active, it will show that extended limit is active instead and vice versa. The NTC conditioner part has also a 1.7147 gain amplifier added to it, which is functional. There is however a problem with the RC-filter connected to the output of the amplifier. It turns out that if the voltage between the two inputs of the comparator are over around 1.4V there will be a current flow between the two inputs. Between each input there are $4.1K\Omega$ resistors. This means that if the voltage between the inputs are over 1.4V the resistor in the RC-filter will be a part of a voltage divider. This will lower the the voltage of the NTC output and thus the comparators will receive wrong NTC value. This is fixed by removing the RC-filter, but this will make the signal more noisy. Other than the problem with the RC-filter the comparators are functioning as intended. The new NTC digitizer has been tested and been successfully communicated with the PCTM. The NTC digitizer will sample the NTC voltage and output a 12bit digital value. This 12bit digital value will be sent over to a DCS using I2C communication. Unfortunately the digital isolator between the voltage digitizer and the trigger board output was connected wrong. It was fixed by soldering the isolator upside down.

There are no problem with the new arming system. Instead of the trigger board arming it self, now the user have to arm it manually. Alarms will sound and indicator will light red if the parameters goes beyond the safe limit. The latch are now reset by using the latch reset button.

The trigger board are now supplied with 7V-15V instead of 5V. To bring

the voltage down to 5V a voltage regulator is being used. The voltage regulator was not designed correctly, as some extra components was unnecessary. Removing the excess components allows the regulator to function as designed. The voltage is supplied by the logic board, which are connected using a D-sub connection. The different trigger signal are also sent over using the D-sub.

The relative humidity part was tested on the V0.3 trigger board. An ATtiny 85 micro controller was programmed to communicate with a SHT85 sensor through I2C communication. The relative humidity was calculated as described in section 6.2.1. The micro controller then output a signal of 0V - 5V in one of its PWM pins. A value of 5V corresponds to 100% relative humidity and 0V to 0%. The value is then compared to a voltage value corresponding to 26% humidity. The trigger board can be changed to compare dew point instead by changing a jumper configuration, but the relative humidity part was only tested. There was no problem with communicating with the SHT85 sensor. No problem was also found with the comparator.

6.4.3 Logic Board V0.1

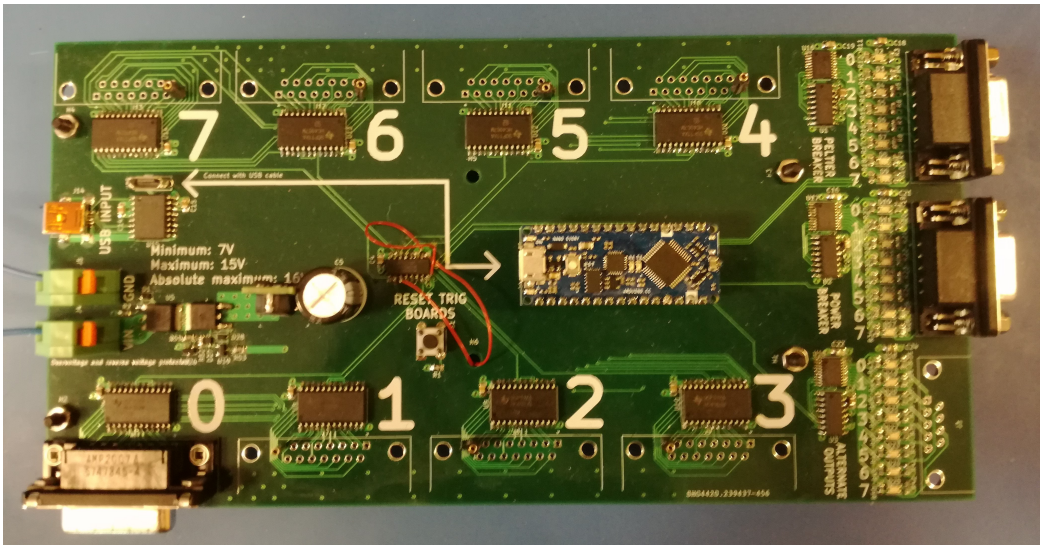


Figure 6.23: A picture of a Logic Board V0.1.

The purpose of the logic board is to handle all the trigger signal from the trigger board. The trigger board then will send signals to the relay boards to shut down e.g power to the module. The actions taken by the logic board are depended on the status of the monitored parameters. The criteria are

described in Table 6.8. The output of the logic board is a True/False signal to the power-cutoff relay boards.

Output type	Criteria
Module power cutoff	IF Overvoltage OR Overtemp OR Humidity_issue
Peltier power cutoff	IF Overtemp OR Undertemp OR Humidity_issue

Table 6.8: Critera for the logic board to take action. Humidity_issue = Enclosure_switch OR Gas_flow OR Humidity.

The logic board has to be supplied with at least 7V and a maximum 15V. It has a voltage regulator, which will regulate the voltage down to 5V to the rest of the board. A total of 8 trigger boards can be connected simultaneously. The signals from the trigger boards will be connected to a multiplexer. The multiplexer is then connected to a PISO shift register. An Arduino microcontroller handles all the trigger signals. First the microcontroller will choose a channel from the multiplexer. The different channels are connected to the trigger signals corresponding to the different parameters. The output of the multiplexer is then read out by the microcontroller serially using the PSIO shift register. A total of five channels have to be read out by the microcontroller. The microcontroller will translate the signal from the multiplexer channel to a 8bit value. Each bit corresponds to each trigger board connection. The five 8bit values are then compared using basic bitwise operations. The microcontroller is then connected to two SIPO shift register. Each shift register is connected to the two D-sub output, which are then connected to the relay boards. The micro controller compares the 8bit values as described in Table 6.8 and outputs a True/False signal. The signals are then sent to the power-cutoff relay boards through the SIPO shift registers. A test firmware to the microcontroller has been successfully developed and the logic board has been tested to function. Some modifications have to be done as the PISO shift register connections was connected wrong (Can be seen in the middle of Figure 6.23).

The logic boards have also indicators to indicate if any of the trigger boards have warranted an action to the relay boards. These lights will usually light green, and will turn red when trigged. The lights have been tested to work. There are however some problem with the design. The lights will light up randomly red and green if there are no trigger board connected to the inputs. This is because the voltage levels will float if there are no connection. We want the lights to be green at all time if there are nothing wrong, as it will be easier to notice if the lights turn red. The problem can be fixed by simply

adding a switch to the different trigger board inputs. With the switch, the user can turn on and off the different inputs to the trigger boards.

6.5 Current State of Module Testing Setup at Bergen

The University of Bergen (UoB) and the University of Oslo(UoO) composes the Norway cluster. UoO will produce linear triplet modules and ship them to UoB where they will be tested. Then the modules will be sent to assembly.

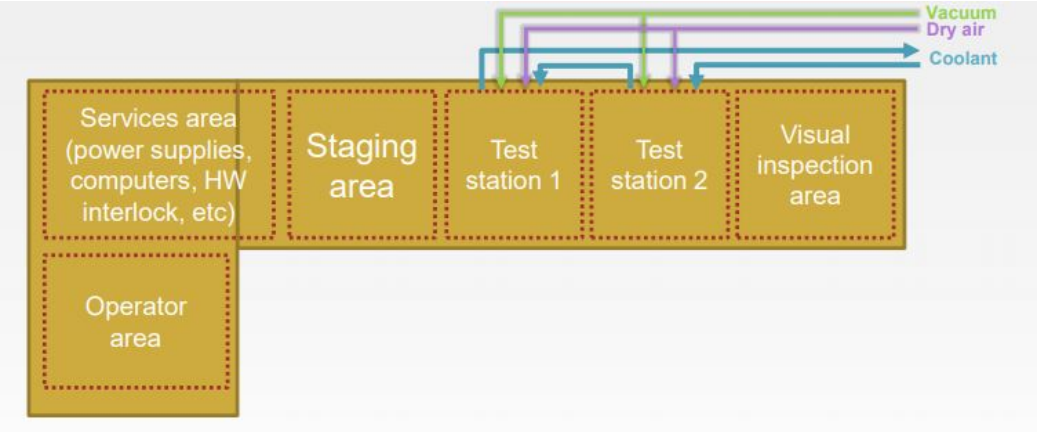


Figure 6.24: An overview of the module testing setup in Bergen. Source [23]

Bergen has currently two test setup capable of testing two modules in parallel. This is expected to sufficient, but can be expanded to include a third setup. Our testing setup will be in a regular lab, not in a cleanroom. This will simplify the development of the setup and if needed can be replicated in a cleanroom later. Figure 6.24 shows an overview of the current setup in Bergen. A test station consist of:

- Triplet cooling unit
- Keithley SMU 2400-Series Sourcemeter high voltage power supply
- Rohde Schwarz HMP4040 low voltage power supply
- Peaktech 1535 low voltage Peltier power supply
- Module power adapters

- Module data adapters
- Dry air flow sensor
- Dry air and vacuum valves
- Hardware interlock boards
- PCTM

The test stations also have a scaffolding where module power adapters, module data adapters, dry air flow sensor, dry air and vacuum valves and PCTM can be attached.

The cooling unit is illustrated by Figure 6.25. It consist of a coldplate, which is connected to a circulating chiller. Two Peltier elements are stacked on top of each other, with thermal interface materials in between. The vacuum chuck sits on top of the Peltiers and has a epoxy pottet PT100 temperature connected. Vacuum is also supplied to the chuck. The module carrier, with the module inside will sit on top of the vacuum chuck. The bottom of carrier is removed and the module with sit directly on the chuck. Dry air is also coming in through the module carrier. The Peltier will help pump the heat away from the module onto the coldplate. The coldplate heat from the coldplate will then be pumped away with the circulating chiller. We are using a Julabo DD1000F circulating chiller. The chiller will give us the ability to test up to 8 modules in parallel. The cooling unit should be able to take the module down to -55°C. This allows us to perform the QC test at a single station.

During QC testing the cooling units must be flushed with dry air. In Bergen we are using Peak Scientific PG28L air dryer. It is capable of dry air flow at maximum 28 L/min, and dew point of -70°C.

For visual inspection we have a modified Epson V600 flatbed scanner and an Olympus microscope with Canon EOS MK II camera. The flatbed scanner can be used to take overview images. The microscope with the camera is used to take detail images and perform manual visual inspection.

To monitor the enclosed temperature and humidity, a SHT85 sensor mounted on a small PCB is used. The small PCB can be attached to the module carrier. The SHT85 sensor sits directly in front of the exhaust hole of the module carrier. This configuration will give us excellent airflow and can directly sample the enclosure temperature.

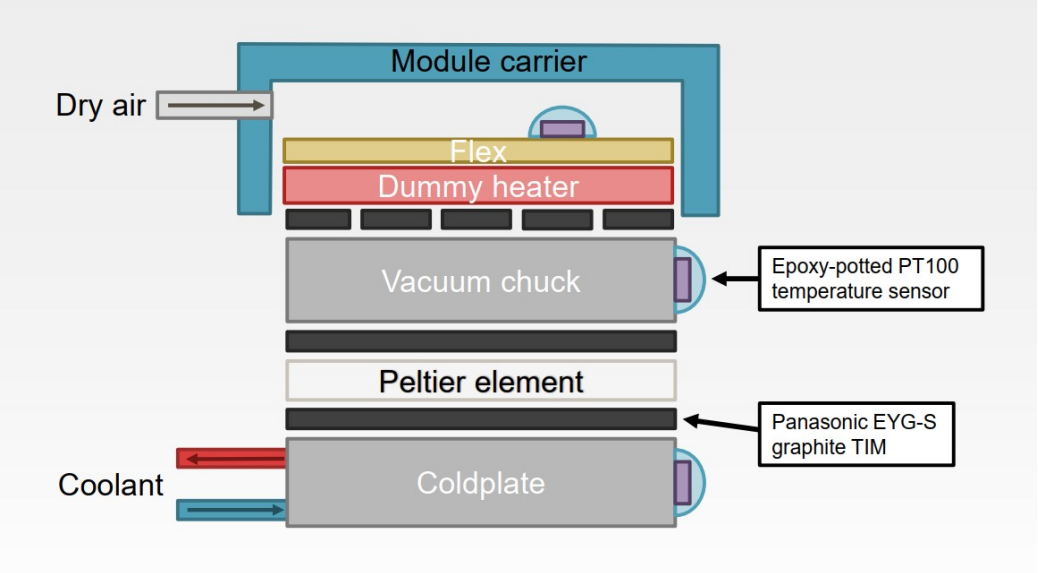


Figure 6.25: A illustration of a cooling unit.

The PCTM and the hardware interlock will be used to monitor the module temperature. Triplet modules have three NTC, which are accessed though the data cable. Two of the NTC inputs will be digitized by the PCTM. One of the NTC input will be digitized by the hardware interlock, which has higher accuracy. The PCTM can then transmit the three NTC measurement, enclosure temperature, humidty and dew point to the DAQ (Data Acquisition) PC. The DAQ PC will also be able to communicate with the modules through display port cables. YARR (Yet Another Rapid Readout) will be used to read and communicate with the chips.

7 Conclusion and Outlook

We have developed a working DCS system and a Peltier Controller. The PCTM has been shown to be able to read the NTC temperature from a D4 ring triplet module during the first qualification test. The PCTM is also able to graph the temperature. All DCS parameters that are monitored during testing is also successfully monitored by the PCTM. This includes, relative humidity, dew point, NTC temperature, coldplate temperature, gas flow, and vacuum chuck temperature. The Peltier Controller part of PCTM has been shown to work well enough to control the temperature quite accurately. When cooling down the modules, the temperature showed no overshoots, but heating it up to 60°C did produce some overshoots. This may be eliminated when the ramp rate restriction is implemented. The hardware interlock has been shown to work in some degree. All the crucial part is working successfully, but some minor design faults are still present in version 0.3. The logic board has also been tested successfully.

The PCTM is currently still in development. It is in a working state, but with minor bugs that have to be fixed. There are features that are needed in the test setup that are not implemented. Stage 2 qualification test requires the DCS data to be uploaded to the local database. Therefore, the PCTM has to be able to upload the DCS data into an influx database, which is then uploaded to a local database. Of course improvement to the Pelter Controller also need to implemented. A form of ramp control need to be in place for the thermal cycling part of the sensor module test. This of course also means testing of thermal cycling also needs to be performed. Later on, automation of thermal cycling would be necessary. PCTM currently does not have any form of software interlock implemented. This will also be necessary in the future. Version 0.4 of the trigger board is currently in development. This version will fix the problems present in version 0.3. The logic board will also be upgraded to version 0.2. The new version will fix the design faults in the prior version and also add some new features. The current test setup in Bergen lacks some features. A kind of source scan capability needs to be developed.

In the future the PCTM will be used by the University of Bergen and University of Oslo. In both locations the PCTM will be used to monitor the temperature, relative humidity and dew point during module testing. The PCTM will also be used considerably to control the temperature under testing, especially thermal cycling.

References

[1] *High-Luminosity LHC*. URL: https://home.cern/resources/faqs/high-luminosity-lhc (visited on 2021).

[2] *About CERN*. URL: https://home.cern/about (visited on 2021).

[3] Philip Bryant Lyndon Evans. "LHC Machine". In: *Journal of Instrumentation* (2008). DOI: 10.1088/1748-0221/3/08/S08001. URL: https://iopscience.iop.org/article/10.1088/1748-0221/3/08/S08001/pdf.

[4] *How an accelerator works*. URL: https://home.cern/science/accelerators/how-accelerator-works (visited on 2021).

[5] The ATLAS Collaboration et al. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* (2008). DOI: 10.1088/1748-0221/3/08/S08003. URL: https://iopscience.iop.org/article/10.1088/1748-0221/3/08/S08003/pdf.

[6] *Technical Design Report for the ATLAS Inner Tracker Pixel Detector*. Tech. rep. Geneva: CERN, Sept. 2017. URL: https://cds.cern.ch/record/2285585.

[7] Burkhard Schmidt. "The High-Luminosity upgrade of the LHC: Physics and Technology Challenges for the Accelerator and the Experiments". In: *Journal of Physics: Conference Series* (2016). DOI: 10.1088/1742-6596/706/2/022002. URL: https://iopscience.iop.org/article/10.1088/1742-6596/706/2/022002/pdf.

[8] O. Dorholt et al. "Beam tests of silicon pixel 3D-sensors developed at SINTEF". In: *Journal of Instrumentation* (2018). DOI: 10.1088/1748-0221/13/08/P08020.

[9] Leonardo Rossi et al. *Pixel Detectors From Fundamentals to Applications*. Springer, 2006.

[10] Maurice Garcia-Sciveres, Flavio Loddo, and Jorgen Christiansen. *RD53B Manual*. Tech. rep. Geneva: CERN, Mar. 2019. URL: https://cds.cern.ch/record/2665301.

[11] Maurice Garcia-Sciveres. *The RD53A Integrated Circuit*. Tech. rep. Geneva: CERN, Oct. 2017. URL: https://cds.cern.ch/record/2287593.

[12] Maurice Garcia-Sciveres. *RD53B Design Requirements*. Tech. rep. Geneva: CERN, Feb. 2019. URL: https://cds.cern.ch/record/2663161.

[13] Lingxin Meng. *RD53A Module Testing Document*. Tech. rep. Latest version on gitlab: https://gitlab.cern.ch/lmeng/rd53amoduletestingdoc. Geneva: CERN, Nov. 2019. URL: https://cds.cern.ch/record/2702738.

[14] Magne Lauritzen. *Peltier Controller and Test Monitor in Bergen*. 2021. URL: https://indico.cern.ch/event/1017141/contributions/4268868/attachments/2209251/3739644/PCTM%20V0.2.pdf.

[15] *NTC Chip Thermistor Temperature Sensing Device Datasheet*. NTCG103JF103FTDS. TDK Corporation. 2021. URL: https://media.digikey.com/pdf/Data%20Sheets/TDK%20PDFs/NTCG103JF103FTDS_Spec.pdf.

[16] *Application Note Dew-point Calculation*. SHTxx. Rev. 1.2. SENSIRON The Sensor Company. 2006. URL: http://irtfweb.ifa.hawaii.edu/~tcs3/tcs3/Misc/Dewpoint_Calculation_Humidity_Sensor_E.pdf.

[17] Toru Fukushima Grayson King. *RTD Interfacing and Linearization Using an ADuC8xx MicroConverter*. Tech. rep. AN-709. Rev. 0.0. Analog Devices, Inc, 2004. URL: https://www.analog.com/media/en/technical-documentation/application-notes/AN709_0.pdf..

[18] Aimee Kalnoskas. *How To Monitor I2C Communications Through RS232*. 2014. URL: https://www.microcontrollertips.com/monitor-i2c-communications-rs232/.

[19] Nick Gammon. *Interrupts*. 2012. URL: http://gammon.com.au/interrupts.

[20] Jacques F. Smuts. *Tuning Rule for Dead-Time Dominant Processes*. 2010. URL: https://blog.opticontrols.com/archives/275.

[21] Jacques F. Smuts. *Level Controller Tuning*. 2011. URL: https://blog.opticontrols.com/archives/697.

[22] *16-Bit I/O Expander with Serial Interface*. MCP23017. Rev. C. MicroChip. 2016. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf.

[23] Magne Lauritzen. *Module testing plans at Bergen*. 2020. URL: https://indico.cern.ch/event/981791/contributions/4147203/attachments/2161453/3646990/Module%20testing%20plans%20at%20Bergen_Magne%20Lauritzen.pdf.

Appendices

Appendix I

The PCTM project can be found on https://gitlab.cern.ch/itk-pixel-test/bergen-pctm. There, one can find how to install the PCTM GUI and how to upload the firmware to the Arduino. The GUI is in the Application folder, the firmware in the firmware folder and the hardware folder includes a list of the parts needed to build a PCTM.

There are some modification that has to be done to the PCTM PCB that are currently not written in the Wiki. $3.3K\Omega$ pull up resistors has to be soldered on the inputs that uses I2C communication. This includes the inputs of NTC1, both of SHT85 inputs and the free I2C input. In addition $3.3K\Omega$ pull up resistors and 100pF capacitors also has to be soldered on the microcontroller SDA and SCL pins. In addition, Pin A7 has to be connected to Pin D9 on the microcontroller.

Using the PCTM is quite straight forward. The PCTM can be powered up by the USB connection or the hardware interlock. If one should power the PCTM through the hardware interlock, the jumpers near the USB input will have to be removed. After powering up the PCTM it should read all the inputs immediately. The various parameters monitored is shown on the small LCD screen on top of the PCB. The USB cable can connect to the PC and communicate with the PCTM GUI. After starting the GUI it should automatically connect to the microcontroller. When the microcontroller is connected it should start to graph the various parameters. One should not need to do anything other than connecting the SHT85, PT100s and NTCs to the inputs.

In the GUI one can change various settings in the PCTM settings button. Here one can change the different sampling frequency. A button to calibrate the current sensor is also present. One would need to press the Upload button to upload the new setting to the microcontroller. The GUI has two tabs. The second tabs handles the Peltier Controller part. Here settings for the TEC PID and PSU PID can be changed. The PID loops can be turned on and off by a button on the bottom right. Be sure to press Upload to upload the new changes to the microcontroller.

The TEC PID and PSU PID loop settings includes option to change the tuning parameters for the PID loop. One can also set the delay for the PID loop and also the limits. Manual mode can be set to control the two controller output manually. This will be useful when tuning the PID loops. The Peltier elements are connected to the Peltier side of the polarity power supply inverter. The power supply should be connected to the PSU side. If using ratiometric PSU, one would need to connect the ratiometric output to the PSU. The current sensor has to be first calibrated before tuning the PSU PID loop.

Appendix II

Listing 1: Arduino Code for the hardware interrupts. Used mainly for the I2C sniffer

```
/*
 InterrruptServiceRoutines.h
 Definitions of Interrupt Service Routines for ↩
    hardware interrupts
 */

#ifndef InterruptServiceRoutines_h
#define InterruptServiceRoutines_h

#include <arduino.h>
#include "Pctm.h"

extern PCTM pctm;
extern Sampler s;

ISR (PORTA_PORT_vect)
/*
 ISR for INTERLOCK_CONNECTION and I2C_SNIFFER_START ↩
    interrupts
 INTERLOCK_CONNECTION and I2C_SNIFFER_START pins ↩
    share the same PORT
 */
{
  /*
   HW interlock interrupt
   */
  if ((VPORTA.INTFLAGS & 0b00000010) >> 1)
  {
    VPORTA.INTFLAGS = VPORTA.INTFLAGS | 0b00000010; ↩
       //Clear flag on EXPANDER_INTB
    pctm.delay_time = millis() - pctm.before_time;
    if (pctm.delay_time < 100)
    {
      return;
    }
    pctm.interlock_connection = !pctm.↩
```

```
    interlock_connection;
  s.interlock_connection = !s.interlock_connection;

  if (pctm.interlock_connection == true)
  {
    PORTA.PIN0CTRL = 0b00000010; //Interrupt ↩
        enabled with sense on rising edge on pin D2 ↩
        (I2C_SNIFFER_START)
  }
  else
  {
    VPORTF.INTFLAGS = VPORTF.INTFLAGS | 0b00100000;↩
        //Clear flag on I2C_SNIFFER_STOP
    VPORTA.INTFLAGS = VPORTA.INTFLAGS | 0b00000001;↩
        //Clear flag on I2C_SNIFFER_START
    VPORTC.INTFLAGS = VPORTC.INTFLAGS | 0b01000000;↩
        //Clear flag on I2C_SNIFFER_SCL
    PORTA.PIN0CTRL = 0b00000000; //Disable ↩
        interrupt on pin D2 (I2C_SNIFFER_START)
    PORTC.PIN6CTRL = 0b00000000; //Disable ↩
        interrupt on pin D4 (I2C_SNIFFER_SCL)
    PORTF.PIN5CTRL = 0b00000000; //Disable ↩
        interrupt on pin D3 (I2C_SNIFFER_STOP)
  }
  pctm.before_time = millis();
}

/*
 I2C start signal interrupt
 */
if (VPORTA.INTFLAGS & 0b00000001)
{
  VPORTA.INTFLAGS = VPORTA.INTFLAGS | 0b00000001; ↩
      //Clear flag on I2C_SNIFFER_START
  PORTC.PIN6CTRL = 0b00000010; ////Interrupt ↩
      enabled with sense on rising edge on pin D4 (↩
      I2C_SNIFFER_SCL)
  PORTF.PIN5CTRL = 0b00000010; ////Interrupt ↩
      enabled with sense on rising edge on pin D3 (↩
      I2C_SNIFFER_STOP)
  s.i2c_buffer = 0;
```

```
    s.i2c_ready = false;
  }
}

ISR (PORTC_PORT_vect)
/*
 ISR for I2C_SNIFFER_SCL
 */
{
  s.sda_buffer = (VPORTB.IN & 0b00000100) >> 2; //↩
     DigitalRead on pin D5 (I2C_SNIFFER_SDA)
  s.i2c_buffer = (s.i2c_buffer << 1) | s.sda_buffer;
  VPORTC.INTFLAGS = VPORTC.INTFLAGS | 0b01000000; //↩
     Clear flag on I2C_SNIFFER_SCL
}

ISR (PORTF_PORT_vect)
/*
 ISR for I2C_SNIFFER_STOP
 */
{
  VPORTF.INTFLAGS = VPORTF.INTFLAGS | 0b00100000; //↩
     Clear flag on I2C_SNIFFER_STOP
  VPORTA.INTFLAGS = VPORTA.INTFLAGS | 0b00000001; //↩
     Clear flag on I2C_SNIFFER_START
  VPORTC.INTFLAGS = VPORTC.INTFLAGS | 0b01000000; //↩
     Clear flag on I2C_SNIFFER_SCL
  s.i2c_ready = true;
  s.i2c_sniffer_time = millis();
  PORTA.PIN0CTRL = 0b00000000; //Disable interrupt on↩
      pin D2 (I2C_SNIFFER_START)
  PORTC.PIN6CTRL = 0b00000000; //Disable interrupt on↩
      pin D4 (I2C_SNIFFER_SCL)
  PORTF.PIN5CTRL = 0b00000000; //Disable interrupt on↩
      pin D3 (I2C_SNIFFER_STOP)
}

#endif
```

Listing 2: Snippet of Arduino Code for the Peltier controller class.

```
void Peltiercontroller::analogCompute()
{
  if (TECPID->Compute())
  {
    setPSUsetpoint(_TEC_Output);
  }
}

void Peltiercontroller::currentCompute()
{
  if (PSUPID->Compute())
  {
    setPSURatiometricOutput((int)_PSU_Output);
  }
}

void Peltiercontroller::setTECsetpoint(double ↩
   setpoint)
{
  _TEC_Setpoint = setpoint;
  sampler.pid_info.TEC_setpoint = _TEC_Setpoint;
}

void Peltiercontroller::setPSUsetpoint(double ↩
   setpoint)
{
    _PSU_Setpoint = setpoint;
    sampler.pid_info.PSU_setpoint = _PSU_Setpoint;
}

void Peltiercontroller::setPSUAbsoluteOutput(double ↩
   value)
// Sets the PSU output value. Unit: Amperes
{
  if (_psu_comm == 0)
  {
    if (_psu_max == 0){return;} // Cannot set an ↩
       absolute output with ratiometric control if ↩
       the PSU max current has not been given
```

```
    setPSURatiometricOutput((int)(255 * value / ↩
        _psu_max));
  }else{
    // Here we must call a function that sets the PSU↩
        output via RS232.
  }
}

void Peltiercontroller::setPSURatiometricOutput(int ↩
    value)
// Sets the ratiometric control output value. Unit: ↩
   PWM counts (-255 to +255)
{
  value = min(max(value, -255), 255); // Enforce ↩
      valid value
  if (value < 0)
  {
    mcp.digitalWrite(POLARITY, HIGH);
  }

  else if (value >= 0)
  {
    mcp.digitalWrite(POLARITY, LOW);
  }

  analogWrite(PSU_RATIOMETRIC, abs(value));
  sampler.pid_info.PSU_control_pwm = value;
  sampler.pid_info.PSU_control = (float)(value)*↩
      _psu_max/255;
}

void Peltiercontroller::setPSUmax(double psu_max)
{
  _psu_max = psu_max;
  EEPROM.put(PSUMAX_EEPROM_ADDR, psu_max);
  EEPROM.put(PSUMAX_FLAG_EEPROM_ADDR, 0);
}

void Peltiercontroller::disable()
// Disable the PID loop. The PCTM boots to disabled ↩
   mode to prevent accidents.
```

```
{
  TECPID->SetMode(MANUAL);
  PSUPID->SetMode(MANUAL);
  setPSUAbsoluteOutput(0);
  _enabled = false;
}

void Peltiercontroller::enable()
// Enable the PID loop.
{
  _applyPreferredSettings();
  _enabled = true;
}

void Peltiercontroller::setMode(pid_settings cmd, ↩
   float value)
{
  switch (cmd)
  {
    case CC:
      _preferredSettings.TECPID = MANUAL;
      _preferredSettings.PSUsetpoint = value;
      break;

    case CV:
      /* // The PCTM cannot yet measure the peltier ↩
         voltage
       *
       */
      break;

    case CCTRL:
      _preferredSettings.PSUPID = MANUAL;
      _preferredSettings.PSUabsout = value;
      break;

    case PID_TEC:
      _preferredSettings.TECPID = AUTOMATIC;
      break;

    case PID_PSU:
```

```
        _preferredSettings.PSUPID = AUTOMATIC;
        break;
    }

    if (_enabled)
    {
      _applyPreferredSettings();
    }
}

void Peltiercontroller::_applyPreferredSettings()
{
  TECPID->SetMode(_preferredSettings.TECPID);
  if (_preferredSettings.TECPID == MANUAL){↩
     setPSUsetpoint(_preferredSettings.PSUsetpoint);}
  PSUPID->SetMode(_preferredSettings.PSUPID);
  if (_preferredSettings.PSUPID == MANUAL){↩
     setPSUAbsoluteOutput(_preferredSettings.↩
     PSUabsout);}

}

void Peltiercontroller::setSampletime(int ↩
   new_sampletime)
{
  _sampletime = new_sampletime;
  TECPID->SetSampleTime(_sampletime);
}

void Peltiercontroller::setILimit(double new_limit)
{
  _i_limit = new_limit;
  //TECPID->SetOutputLimits(-_limit, _limit);
}

void Peltiercontroller::setVLimit(double new_limit)
{
  _v_limit = new_limit;
  //TECPID->SetOutputLimits(-_limit, _limit);
}
```

```
void Peltiercontroller::setKp(double new_Kp, ↩
   pid_settings which_pid)
{
  switch (which_pid)
  {
    case PID_TEC:
      _Kp = new_Kp;
      EEPROM.put(TECKP_EEPROM_ADDR, _Kp);
      TECPID->SetTunings(_Kp, _Ki, _Kd);
      break;
    case PID_PSU:
      _PSU_Kp = new_Kp;
      EEPROM.put(PSUKP_EEPROM_ADDR, _PSU_Kp);
      PSUPID->SetTunings(_PSU_Kp, _PSU_Ki, _PSU_Kd);
      break;
  }
}

void Peltiercontroller::setKi(double new_Ki, ↩
   pid_settings which_pid)
{
  switch (which_pid)
  {
    case PID_TEC:
      _Ki = new_Ki;
      EEPROM.put(TECKI_EEPROM_ADDR, _Ki);
      TECPID->SetTunings(_Kp, _Ki, _Kd);
      break;
    case PID_PSU:
      _PSU_Ki = new_Ki;
      EEPROM.put(PSUKI_EEPROM_ADDR, _PSU_Ki);
      PSUPID->SetTunings(_PSU_Kp, _PSU_Ki, _PSU_Kd);
      break;
  }
}

void Peltiercontroller::setKd(double new_Kd, ↩
   pid_settings which_pid)
{
  switch (which_pid)
  {
```

```
      case PID_TEC:
        _Kd = new_Kd;
        EEPROM.put(TECKD_EEPROM_ADDR, _Kd);
        TECPID->SetTunings(_Kp, _Ki, _Kd);
        break;
      case PID_PSU:
        _PSU_Kd = new_Kd;
        EEPROM.put(PSUKD_EEPROM_ADDR, _PSU_Kd);
        PSUPID->SetTunings(_PSU_Kp, _PSU_Ki, _PSU_Kd);
        break;
    }
}
```

Listing 3: Snippet of Arduino Code for the sampler class. Handles the sampling of the PCTM inputs

```
sample Sampler::_readModuleTemperature(int module_num↩
    )
{
  int ntc_value;
  double ntc_ratio;
  sample temp_struct;

  switch(module_num)
  {
    case ss_NTC_1:
      _i2c_select(2);
      Wire.requestFrom(0x48, 2);
      if(Wire.available())
      {
        int ntc_msb = Wire.read();
        ntc_msb = ntc_msb << 8;
        ntc_value = Wire.read() | ntc_msb;
        ntc_ratio = ntc_value / (4096 * 1.7147);
      }
      else
      {
        ntc_value = NAN;
        ntc_ratio = NAN;
      }
      break;
    case ss_NTC_2:
      ntc_value = analogRead(NTC_2);
      ntc_ratio = ntc_value / 1023.0;
      break;
    case ss_NTC_3:
      ntc_value = analogRead(NTC_3);
      ntc_ratio = ntc_value / 1023.0;
      break;
  }

  double ntc_res = (ntc_ratio * 84.5) / (1 - ↩
     ntc_ratio * (1.0 + (84.5 / 132.0)));
```

```
  double ntc_res_log = log10(ntc_res);
  double ntc_temp = -1.46779 * pow(ntc_res_log, 3.0) ↩
     + 14.5655 * pow(ntc_res_log, 2.0) - 86.2842 * ↩
     ntc_res_log + 371.338;
  double ntc_temp_cels = ntc_temp - 273.15;

  float sample_time  = _time();

  temp_struct.val = ntc_temp_cels;
  temp_struct.s = sample_time;
  temp_struct.counts = ntc_value;

  return temp_struct;
}

sample Sampler::_readColdPlateTemperature()
{
  sample temp_struct;

  double ntc_value = analogRead(NTC_CP);
  double ntc_ratio = ntc_value / 1023.0;
  double ntc_res = (ntc_ratio * 24.3) / (1 - ↩
     ntc_ratio * (1.0 + (24.3 / 132.0)));

  double ntc_res_log = log10(ntc_res);
  double ntc_temp = -1.46779 * pow(ntc_res_log, 3.0) ↩
     + 14.5655 * pow(ntc_res_log, 2.0) - 86.2842 * ↩
     ntc_res_log + 371.338;
  double ntc_temp_cels = ntc_temp - 273.15;

  float sample_time = _time();

  temp_struct.val = ntc_temp_cels;
  temp_struct.s = sample_time;
  temp_struct.counts = ntc_value;

  return temp_struct;
}

sample Sampler::_readGasFlow()
{
```

```
  sample gasflow_struct;
  double PLACEHOLDER_gasflow_max = 5.0;

  double gasflow_value = analogRead(FLOW);
  double gasflow_ratio = gasflow_value / 1023.0;

  double gasflow = gasflow_ratio * ↩
     PLACEHOLDER_gasflow_max;

  float sample_time = _time();

  gasflow_struct.val = gasflow;
  gasflow_struct.s = sample_time;
  gasflow_struct.counts = gasflow_value;

  return gasflow_struct;
}

humidity_struct Sampler::_readHumidity(int sensor)
{
  _i2c_select(sensor);
  sample relhum_struct;
  sample dew_struct;
  sample temp_struct;
  humidity_struct humidity_struct;

  if (sht->readSample())
  {
    double temp = sht->getTemperature();
    double relhum = sht->getHumidity();

    double H = (log10(relhum) - 2.0) / 0.4343 + ↩
       (17.62 * temp) / (243.12 + temp); // SHTxx Dew↩
       -point Calculation from http://irtfweb.ifa.↩
       hawaii.edu/~tcs3/tcs3/Misc/↩
       Dewpoint_Calculation_Humidity_Sensor_E.pdf
    double dewPoint = (243.12 * H) / (17.62 - H);

    float sample_time = _time();

    relhum_struct.val = relhum;
```

```
    relhum_struct.s = sample_time;

    temp_struct.val = temp;
    temp_struct.s = sample_time;

    dew_struct.val = dewPoint;
    dew_struct.s = sample_time;

    humidity_struct.rel = relhum_struct;
    humidity_struct.dp = dew_struct;
    humidity_struct.t = temp_struct;

    return humidity_struct;
  }
  else
  {
    float sample_time = _time();

    relhum_struct.val = NAN;
    relhum_struct.s = sample_time;

    temp_struct.val = NAN;
    temp_struct.s = sample_time;

    dew_struct.val = NAN;
    dew_struct.s = sample_time;

    humidity_struct.rel = relhum_struct;
    humidity_struct.dp = dew_struct;
    humidity_struct.t = temp_struct;

    return humidity_struct;
  }
}

humidity_struct Sampler::_readHumidity_sniffer()
{
  sample relhum_struct;
  sample dew_struct;
  sample temp_struct;
  humidity_struct humidity_struct;
```

```
uint16_t humidity_MSB = (i2c_buffer >> 20) & 0↩
    b11111111;
uint16_t humidity_LSB = (i2c_buffer >> 11) & 0↩
    b11111111;
uint16_t humidity_value = (humidity_MSB << 8) | ↩
    humidity_LSB;
double relhum = (humidity_value / (65536.0 - 1.0)) ↩
    * 100.0;

uint16_t temperature_MSB = (i2c_buffer >> 47) & 0↩
    b11111111;
uint16_t temperature_LSB = (i2c_buffer >> 38) & 0↩
    b11111111;
uint16_t temperature_value = (temperature_MSB << 8)↩
     | temperature_LSB;
double temperature = -45.0 + (temperature_value / ↩
    (65536.0 - 1.0)) * 175.0;

double H = (log10(relhum) - 2.0) / 0.4343 + (17.62 ↩
    * temperature) / (243.12 + temperature); // ↩
    SHTxx Dew-point Calculation from http://irtfweb.↩
    ifa.hawaii.edu/~tcs3/tcs3/Misc/↩
    Dewpoint_Calculation_Humidity_Sensor_E.pdf
double dewPoint = (243.12 * H) / (17.62 - H);

float sample_time = i2c_sniffer_time - sync;

relhum_struct.val = relhum;
relhum_struct.s = sample_time;

temp_struct.val = temperature;
temp_struct.s = sample_time;

dew_struct.val = dewPoint;
dew_struct.s = sample_time;

humidity_struct.rel = relhum_struct;
humidity_struct.dp = dew_struct;
humidity_struct.t = temp_struct;
```

```
  return humidity_struct;
}

sample Sampler::_readPeltierCurrent()
{
  sample peltier_struct;

  double peltier_value = analogRead(PELTIER_CURRENT);
  double peltier_ratio = peltier_value / 1023.0;
  double peltier_current = ( 3.3 * peltier_ratio / ↩
     0.055 ) - _current_zero;

  float sample_time = _time();

  peltier_struct.val = peltier_current;
  peltier_struct.s = sample_time;
  peltier_struct.counts = peltier_value;

  return peltier_struct;

}

unsigned long Sampler::_time()
{
  //It would be nice to return a double here, but ↩
     apparently on arduinos all doubles are actually ↩
     floats.
  //Floats only have sub-10ms accuracy up to 36 hours↩
     .
  unsigned long t = millis() - sync; //Milliseconds ↩
     since last PC sync
  return t;
}

void Sampler::_i2c_select(int channel)
{
  uint8_t sendByte;
  switch (channel)
  {
    case 0:
      sendByte = 0b00000100;
```

```
      break;
    case 1:
      sendByte = 0b00000101;
      break;
    case 2:
      sendByte = 0b00000110;
      break;
    case 3:
      sendByte = 0b00000111;
      break;
  }

  Wire.beginTransmission(0x70);
  Wire.write(sendByte);
  Wire.endTransmission();
}
```