# Project Work at MPIK

### Temperature control system

Stefan Dickopf

## 1 Experimental setup

A temperature sensor, a heater and a cooler are to be connected to a Arduino board. In order for it to work we need to have shifters for the voltage range to fit the inputs/outputs on the Arduino. This was first done by Vanessa Scheller using cables and a breadboard, I took her design and reworked it to be used on a milled circuit board and SMD parts.

### 1.1 Temperature sensor circuit

The circuit for the temperature sensor looks as follows:

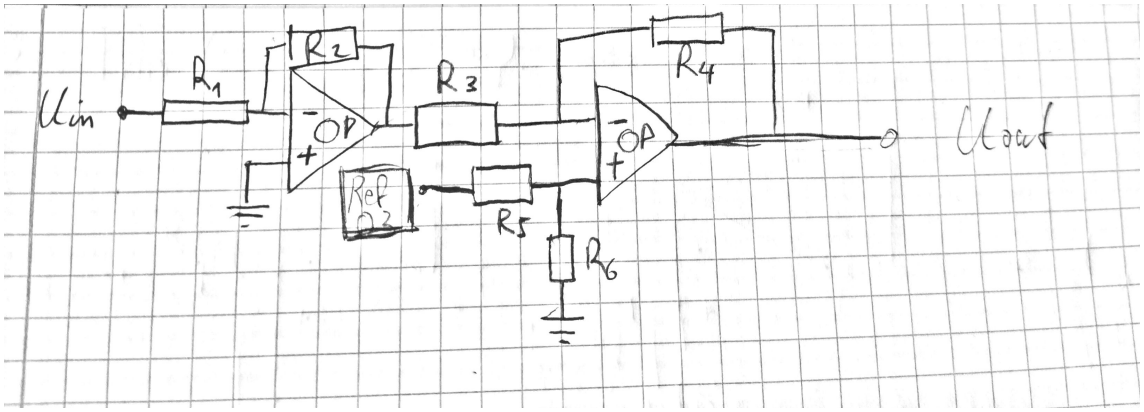



Figure 1: Temp. sensor shifter circuit

The input range from the temperature sensor we are interested in is

$$-1.6V < U_{in} < 1.6V$$

The supply voltage of the op-amps is 15V. We want to have a cutoff after the first op-amp at the input range boundaries. This can be achieved with setting the amplification factor with $R_1$ and $R_2$. The output range should equal the input range of the Arduino analogue input which goes form 0-3.3V. The values found for the resistances are as follows (in $k\Omega$):
$R_1 = 15k\Omega$
$R_2 = 133k\Omega$
$R_3 = 133k\Omega$
$R_4 = 12k\Omega + 10k\Omega$ potentiometer
$R_5 = 10k\Omega$
$R_6 = 15k\Omega + 5k\Omega$ potentiometer

This gives to following response to the input Voltage $U_{in}$

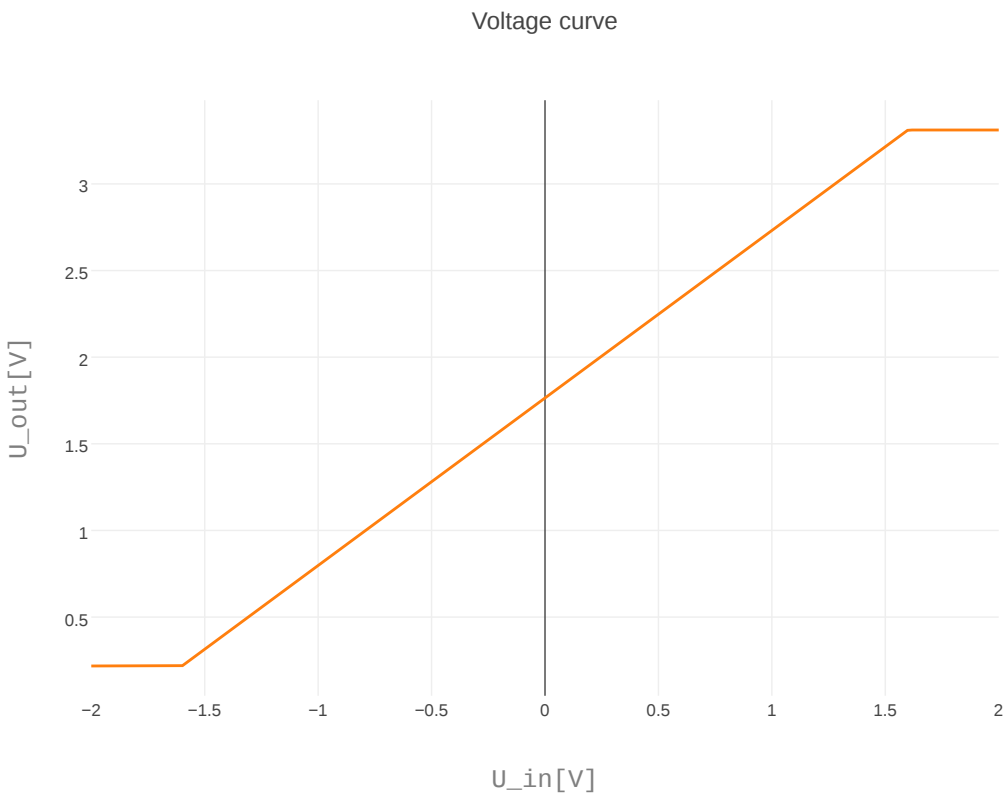



Figure 2: Theoretical response of temp. shifter

## 1.2 Heater, cooler circuit

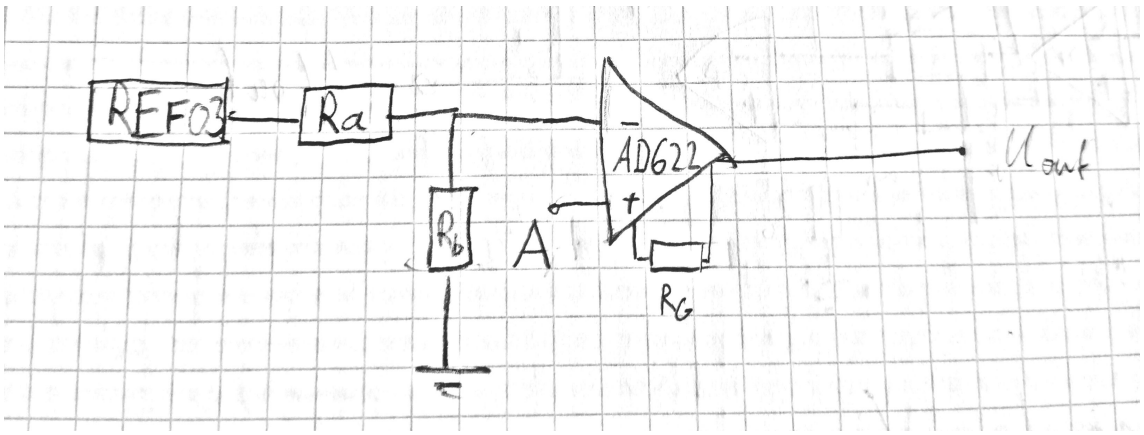



Figure 3: Heater/cooler shifter circuit

The output range from the Arduino's DAC is

$$0.55V < U_{DAC} < 2.75V$$

The regulation voltage from the heater and cooler is 0 to 10V. First the voltage from the DAC out needs to be reduced by 0.55V and then amplified such that the maximum out from the DAC corresponds to the maximum in from the heater/cooler. The amplification is set with the resistance RG. The values found by Vanessa are (in $k\Omega$):
$R_a = 3.92k\Omega + 5k\Omega$ potentiometer
$R_b = 1k\Omega$
$R_G = 27k\Omega + 10k\Omega$ potentiometer

The output here can be described by the amplified difference of the two input voltages. The amplification factor was adjusted such that it gives the needed output range. To both circuits many capacitances are added, which are only for filtering the input +15V/-15V and the ref03 voltages from AC parts. This is done with 10nF and 100nF capacitances in parallel.

## 1.3 Realization of the circuitboard

All of the circuits were realized using a smd board. The circuitboard was constructed using EAGLE Professional and made with a circuit board cutter. It has the same proportions as the used arduino due, such that it fits into the enclosure thats already existing.

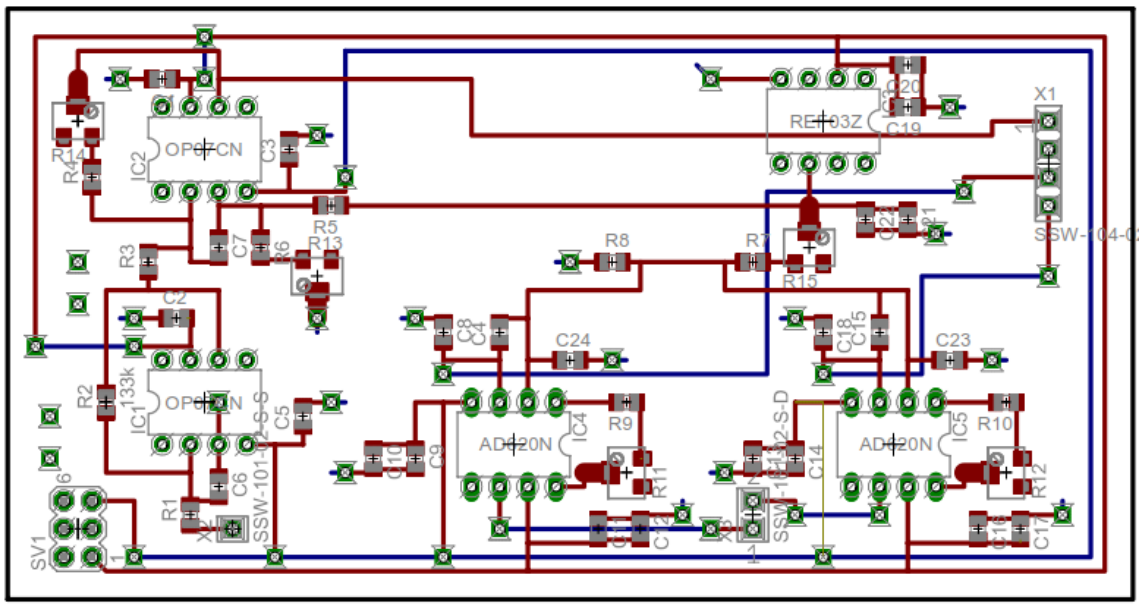



Figure 4: Circuitboard layout

The resistors and the capacitances are directly soldered onto the board, while the OP07 and the AD622 (not 620 as on the image) are put onto headers. The complete board is shown here in Figure 5.
Notice that the header for the left AD622 was placed in the opposite direction, such that the marker for the top is actually on the other side than the top is supposed to be.
Also some of the capacitances were not soldered in, on the design of the board they were made as a precaution.
It was found that the heater/cooler cables add a capacitance which lead to oscillations on the temperature measurement. We added $100\Omega$ resistors on the heater/cooler output to resolve the issue.
In Figure 6 we see the board connected to the Arduino with the Ethernet Shield 2 on top. After everything on the board is done it is put into the housing in the cabinet.

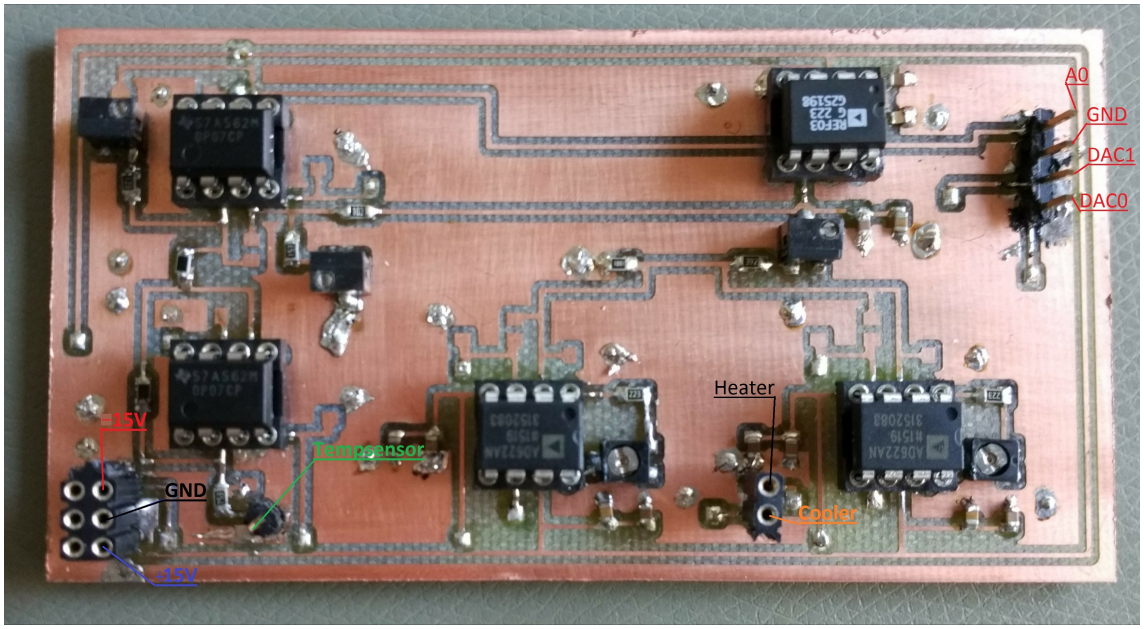



Figure 5: Circuitboard picture

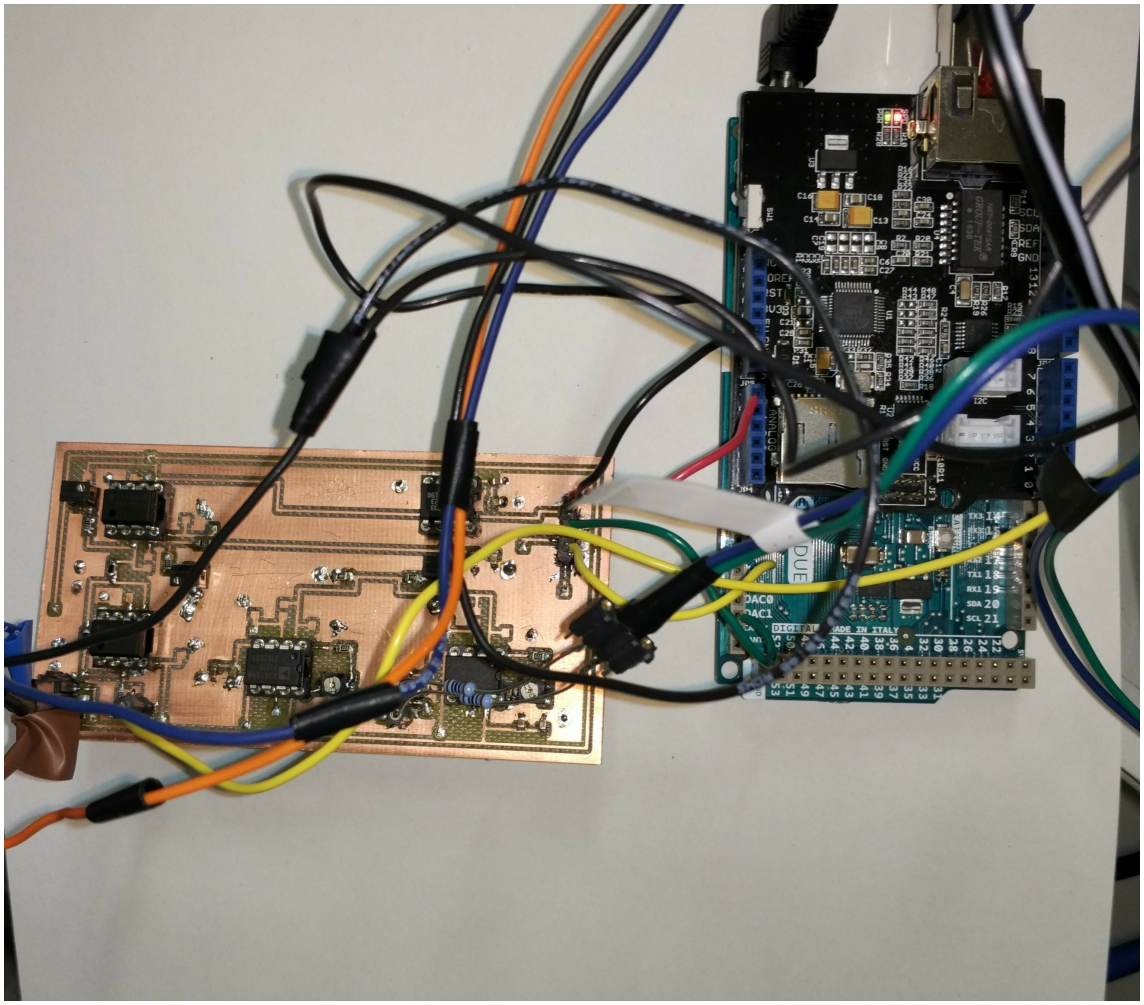


Figure 6: Board connected to Arduino

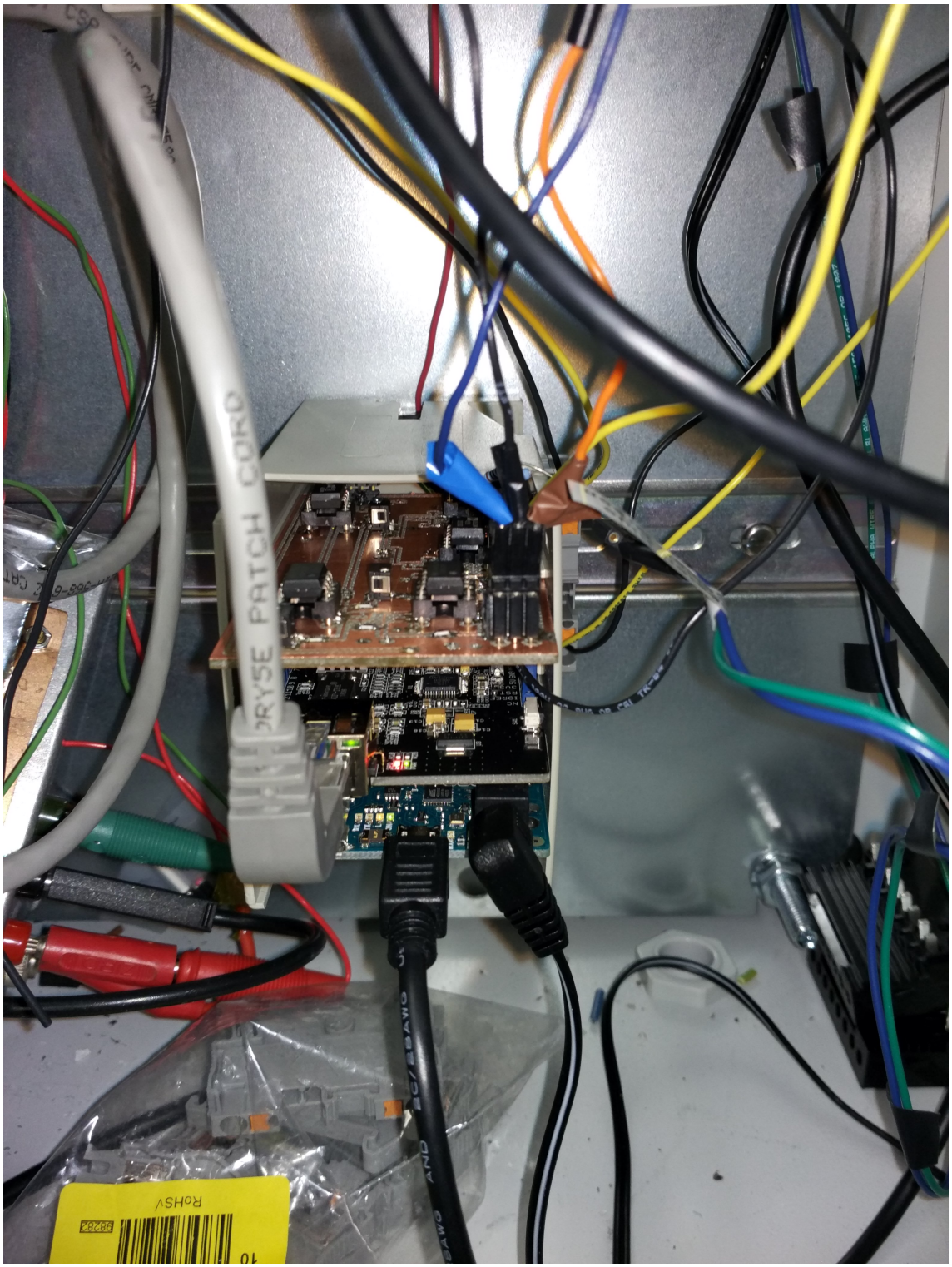
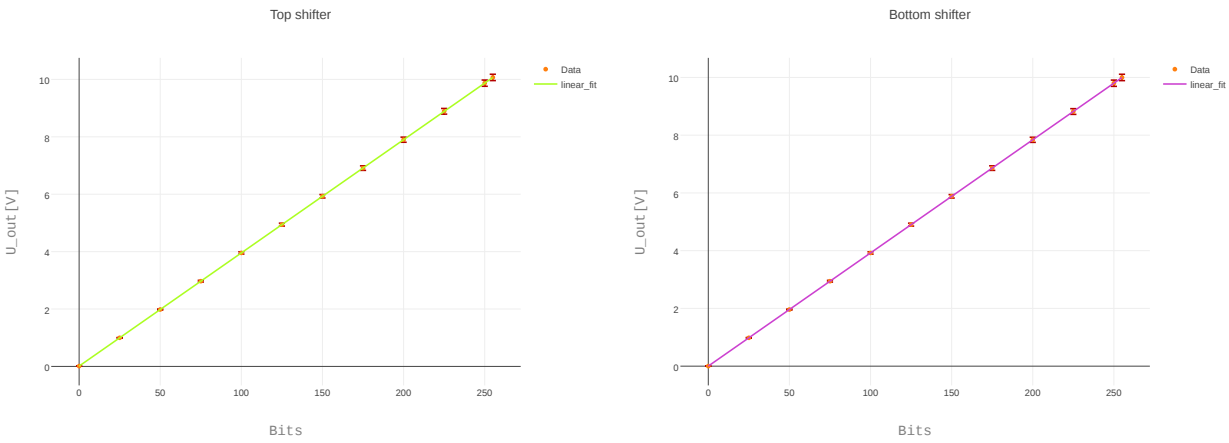



Figure 7: Board and Arduino in the housing

# 2 Measurements

## 2.1 Measurements for the heater/cooler shifter circuit

I measure the output voltage from the heater and cooler shifter while changing the input through the Arduino DAC outputs. The DAC outputs use 8Bit resolution, so we measure from 0 to 255 Bits.




(a) (b)

Figure 8: Heater/cooler shifter calibration

It's visible that the components act as expected, i.e. linearity is achieved. The minimum out voltages are near enough to 0 and the maximum voltages are close enough to 10V.

## 2.2 Measurements of temperature sensor circuit

We first measure the voltages (and set them with the variable resistances) on the output for the min/max values -1.6V/1.6V and make sure that above/below these voltages the output voltage does not change anymore. Then we measure the values inbetween.

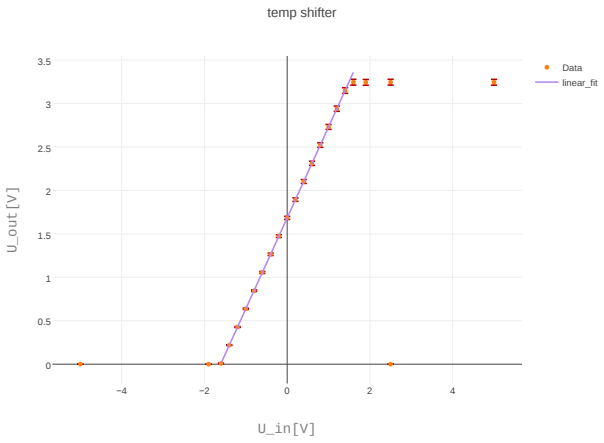



Figure 9: Response of the temp. shifter

We see that we have linearity between -1.6V and 1.6V (slight drop before 1.6V) and afterwards the desired plateaus.

# 3 The temperature control system

## 3.1 Controlling the PID

The temperature value gets read out on a arduino due platform. It then checks with the temperature reference and then calculates control values for the heater / cooler with a PID controller, which was realized as a software running on the arduino. This was originally developed by Vanessa Scheller. I worked with her log book and report.
One of my tasks was to install the Ethernet Shield 2 on the Arduino in order for the PID controller to be abled to communicate more easily with a computer. The existing possibility to communicate with the Arduino was to do it over the usb serial interface of the Arduino which allows to read out data and also send data to the Arduino, but requires a computer to be connected to the Arduino at all times. The advantage of a network interface is that it just needs a connected ethernet cable to the Arduino.
The Ethernet Shield 2 is just connected to the Arduino like any other Arduino shield, by just plugging it on top of the Arduino. The Arduino acts as a server which communicates with a client program running on the tritium-vserver.
The client sends commands directly to the Arduino which get handled there in a defined time intervall. The Arduino sends back a response including information.
The client program runs in the background and issues the command to receive the serial output from the arduino and stores those directly in a file as is, but also creates a THee file with only the time and the temperature values.
The client program can be controlled with the controller script. Running the script with a command line argument makes the client program send the argument as a command to the Arduino. The answer from the arduino is stored in a log file and also printed on the console.
The commands are to be send as follows:

**Getting the serial output**
Send the command ***g*** to receive the serial output. The values received are:

1. Measurement number (resets every new running day)
2. Mean temperature value over 1 minute. (in 10 bit resolution from 0 to 1023)
3. Difference of current temperature to set value of temperature
4. Difference of mean temperature to set value of temperature
5. Heater set value (in 8 bit resolution from 0 to 255)
6. Cooler set value (in 8 bit resolution from 0 to 255)
7. Days running

**Getting the PID parameters and the temperature set value**
Send the command ***gpid*** to receive the values $K_P$, $T_N$ and $T_V$ for both the heater and the cooler as well as the current temperature set value (which is titled sollwert). Also get the state of the PID (1 for turned on, 0 for turned off).

**Setting the PID parameters and setting fixed heater/cooler set values**
Send the command ***xy###***, where *x* is either *h* or *c* for heater/cooler and *y* is either *p*,*i*,*d* for the respective part of the regulator. *y* can also be *f* for a fixed value of the heater/cooler set value (this turns of the PID control). ### corresponds to the wanted value (again in 8 bits from 0 to 255). The Arduino programm checks whether the value is in bounds.

**Setting the temperature reference**
Send the command ***so###*** to set the temperature reference to the value ### (in 10 bits).

**Turning PID off and back on**
Send the command ***pidon / pidoff*** to turn the PID on or off. If the PID is turned off it will use the fixed values for the set values of the heater and cooler (default is zero for both).

The calibration for the temperature values of the sensor is shown in Figure 10

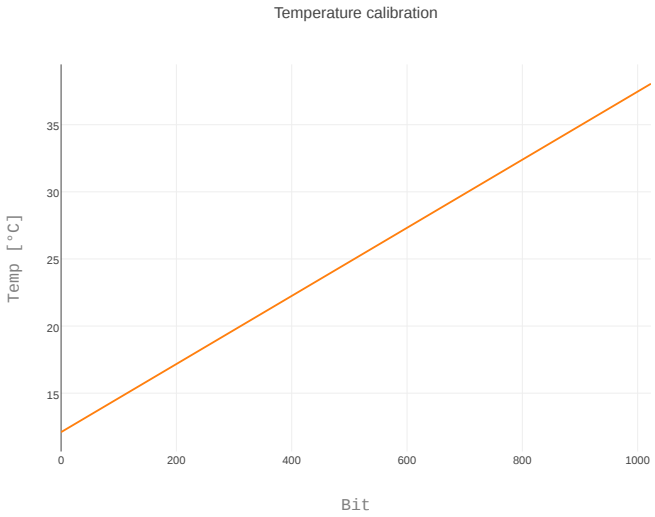



Figure 10: Calibration of temperature sensor

## 3.2 Testing the temperature control system

The board is now connected to the arduino, the temperature sensor and the heater/cooler. Now I measure for a constant temperature reference the temperature with already existing sensors (i.e. top window, top door and the pid temperature sensor).

### 3.2.1 Holding the temperature constant

We want to see now, how well the temperature stabilization works for keeping the temperature constant over the day (18.02.2017). We get the data for one day from the THee-Files located in the tritium-drive (the pid-client program also produces Thee-files there). See Figure 11 (a-c) for the temperatures. Also a histogram of the data at the PID-sensor is plotted in Figure 11 (d).
The mean of the histogram is at $\mu = 23.515K$ and the standard deviation at $\sigma = 0.057K$. We see that for the sensor the PID uses to regulate we have a good nearly constant temperature (fluctuations $\sim 0.5°C$ and $\sigma = 0.057K$). The other two sensor show a fast

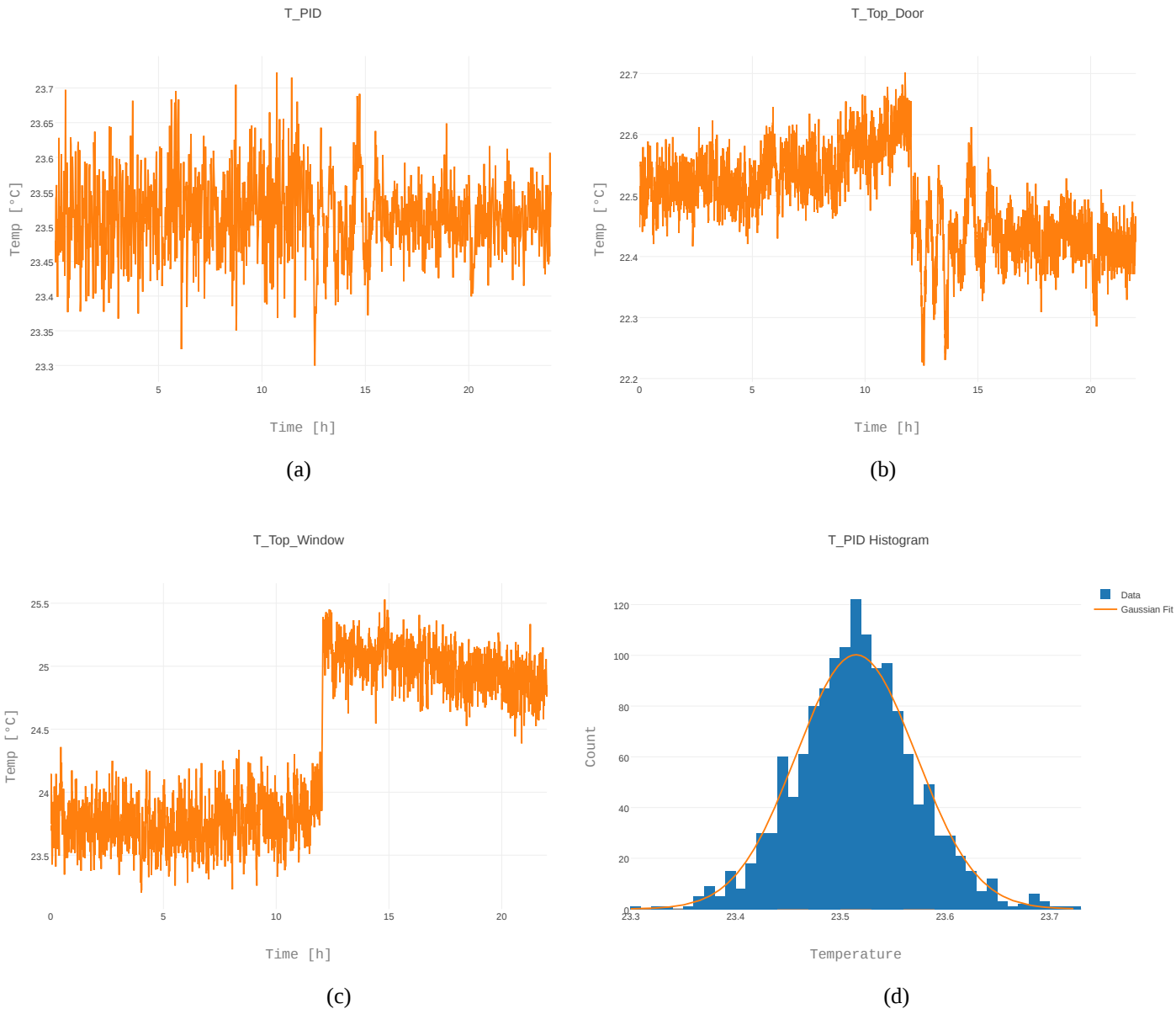
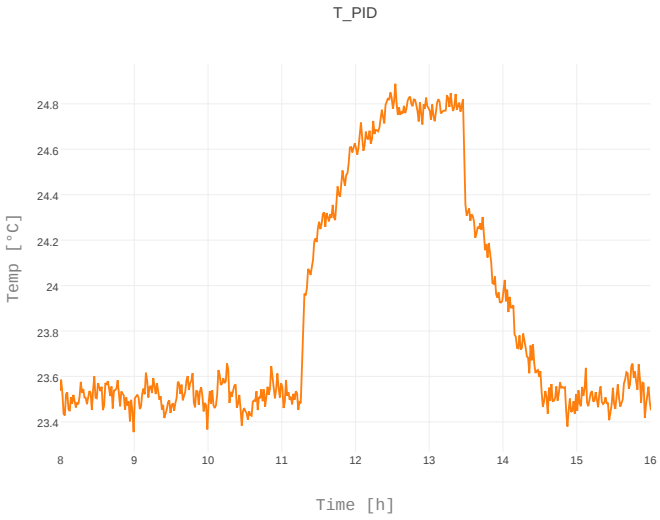



Figure 11: Temperature stabilization over one day with temp. from sensors T_PID, T_Top_Door, T_Top_Window.

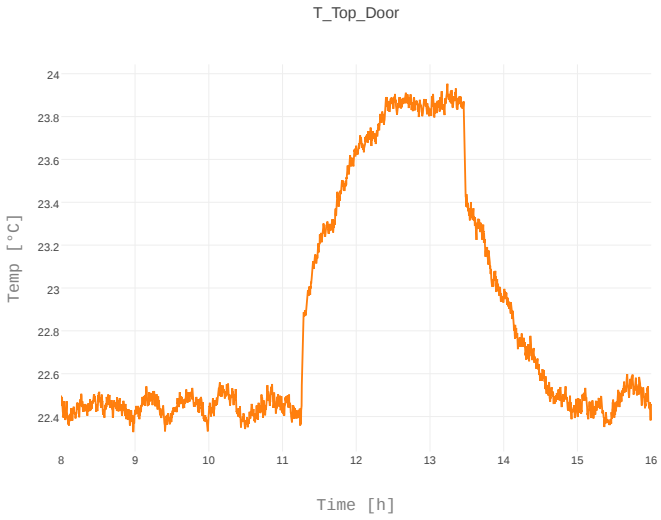
rise(window) and a fall(door) in temperature at around 12 o'clock. I can now look at the cooler/heater values and see that the cooler was shut off up until nearly 12 o'clock and then turned on (probably due to rise in outside temperature), while the heater value decreased. The heater is located closer to the window and the cooler closer to the door. The rise in outside temperature could be due to sun coming in the windows thus raising the temperature there fast. The drop at the door should then be due to the cooler being turned on.

### 3.2.2 Change in temperature reference

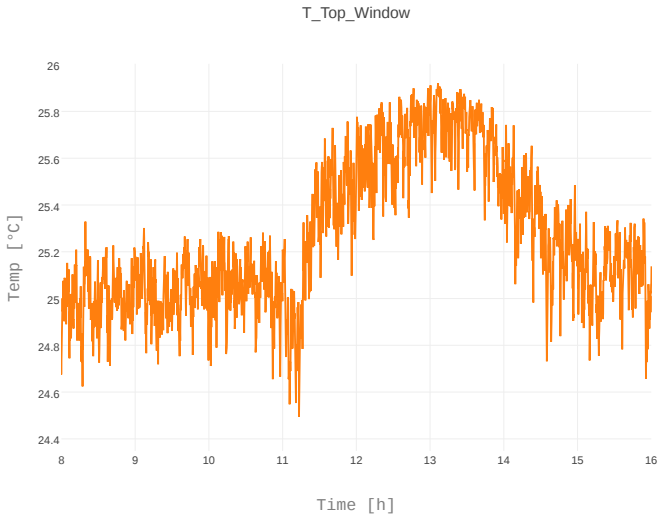
To see how the system reacts to rapid changes in temperature it is much easier to look at a change in the reference temperature. For this we change the value of the reference temperature from 450 Bits ($\sim 23.5°C$) to 500 Bits ($\sim 24.8°C$).
Again we look at the temperature values from the 3 sensors.




(a)




(b)




(c)

Figure 12: Respose to change in intemperature reference

It's visible that temperature at the door and at the PID sensor react very similarly, the temperature at the window seems to react a little slower.
I can now compute the time constant of the system with the following fit:

$$f(t) = B + A(1 - e^{-t/\tau})$$

for the upwards slope and

$$f(t) = B + Ae^{-t/\tau}$$

for the downwards slope.
For the fit to work we need errors for the temperature values. For this we assume a constant

error of 0.1 °C. The bounds are set to the time the temperature reference was set.

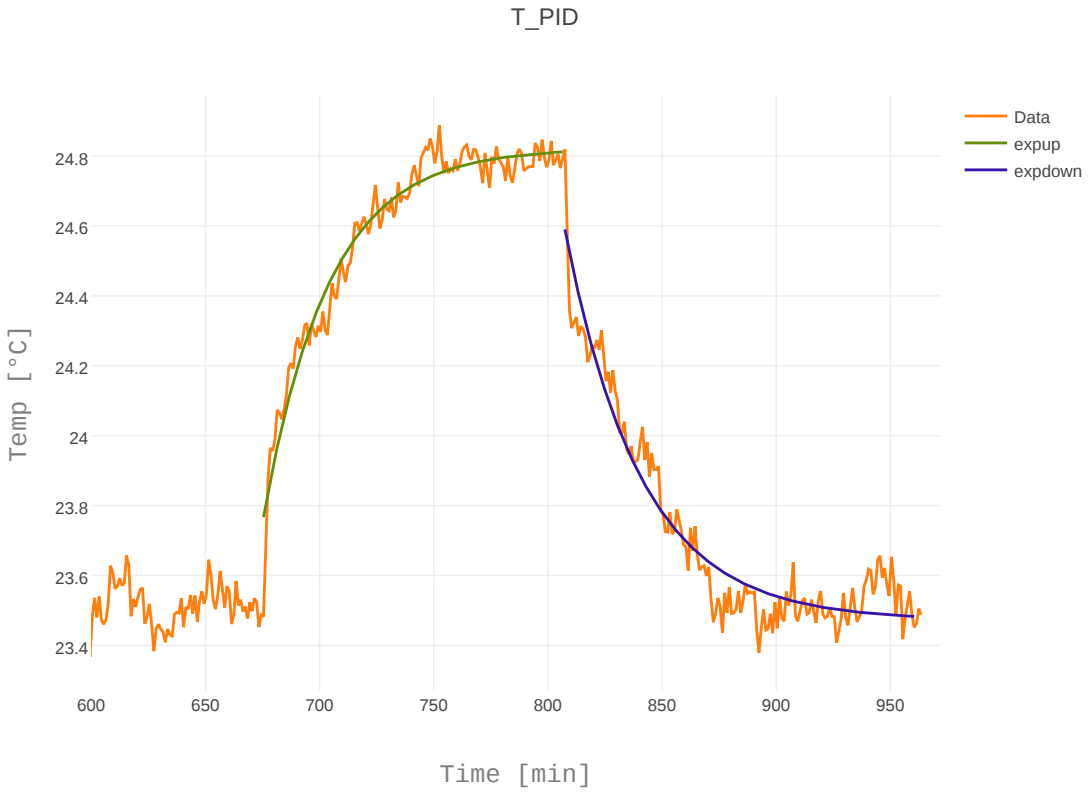



Figure 13: Response of the PID-controller for a change of 50 bits in temperature

The time constants are:

$$\tau_{up} = (28.6 \pm 2.3)min, \quad \tau_{down} = (33.1 \pm 2.4)min$$

It is also visible that the temperature does not overshoot and does not strongly oscillate.

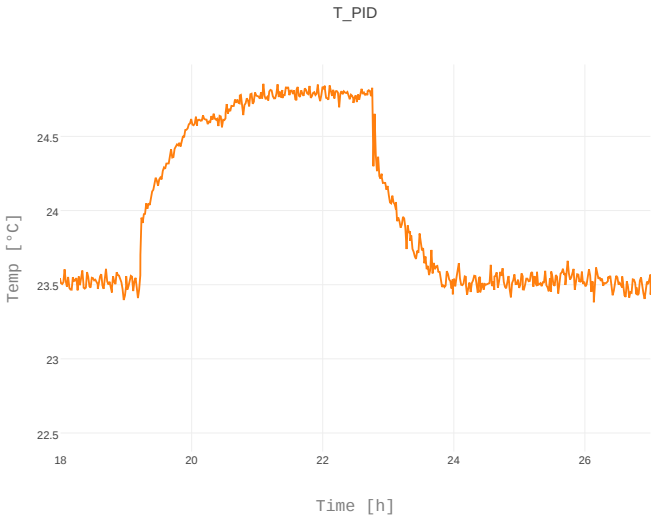
I do another measurement to confirm the behaviour. This one is done at night (Temperature to 500 Bits at 19:11 and back to 450 Bits at 22:45) (see Figure 14 ).
I also repeat the fitting procedure (Figure 15 ). We get a similar value for the upwards slope. The downwards slope is slightly faster than before, which might be cause by lower outside temperature. We see here that some time after the temperature reaches the lower value again it starts to oscillate heavier. This is also visible for the Door temperature, which we can also fit (Figure 16 ).
Here we get

$$\tau_{up} = (32.5 \pm 2.0)min, \quad \tau_{down} = (25.4 \pm 1.5)min$$

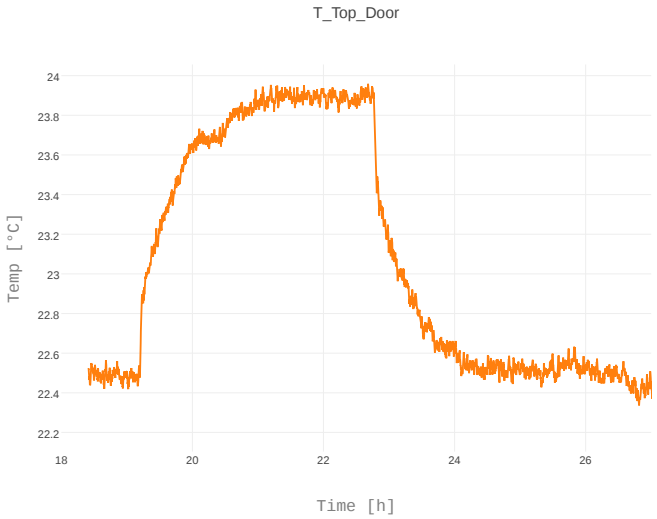
for the PID-sensor and

$$\tau_{up} = (34.7 \pm 2.0)min, \quad \tau_{down} = (32.7 \pm 1.6)min$$

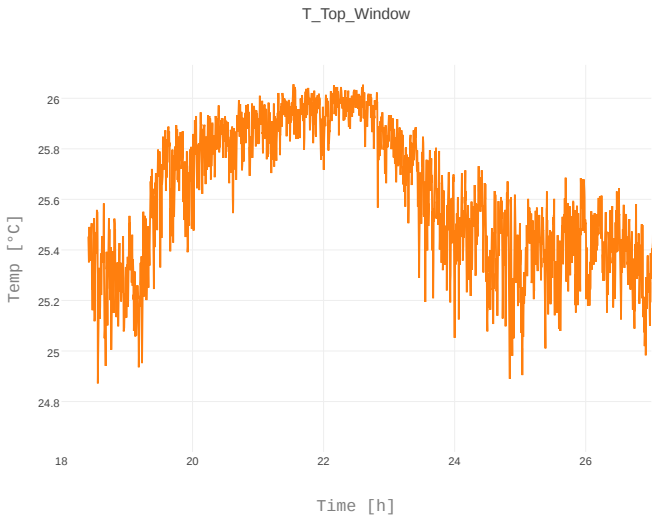
We get a higher value for the upwards slope and a slightly faster downwards slope than before, which might be cause by lower outside temperature. We see here that some time after the temperature reaches the lower value again it starts to oscillate heavier.
This is also visible for the Door temperature, which we can also fit.




(a)




(b)




(c)

Figure 14: Response 2 to change in intemperature reference

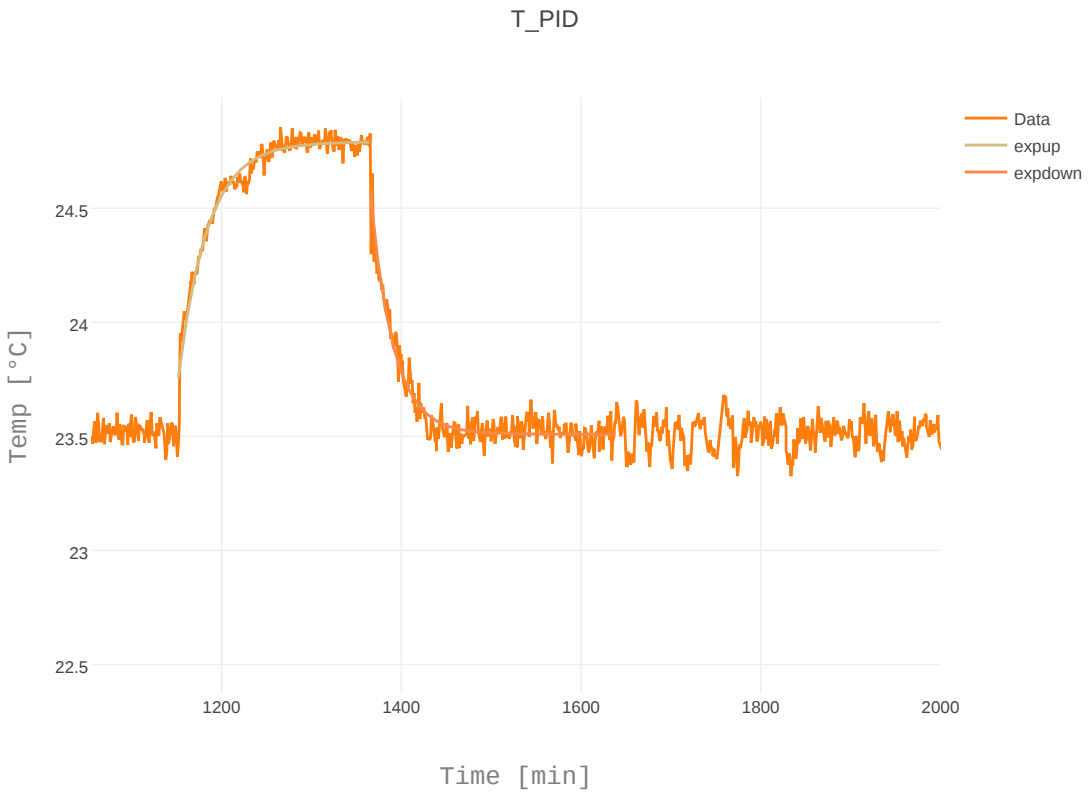



Figure 15: Step response 2 of PID on PID temp. sensor

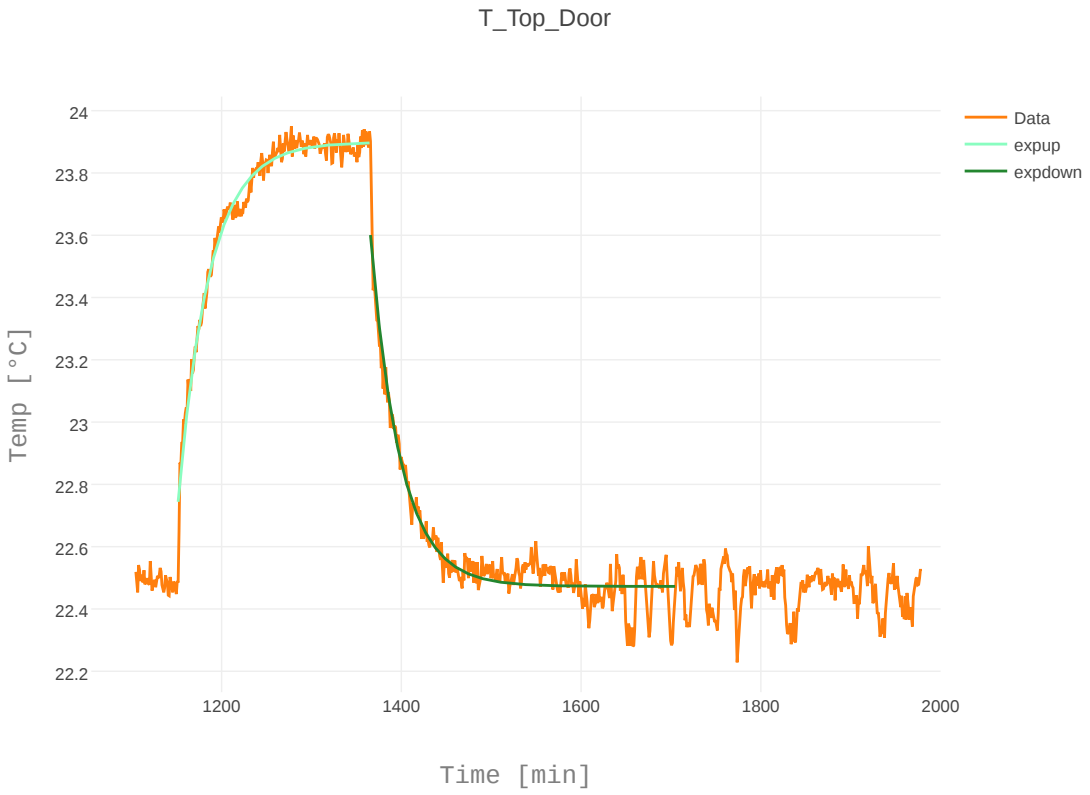
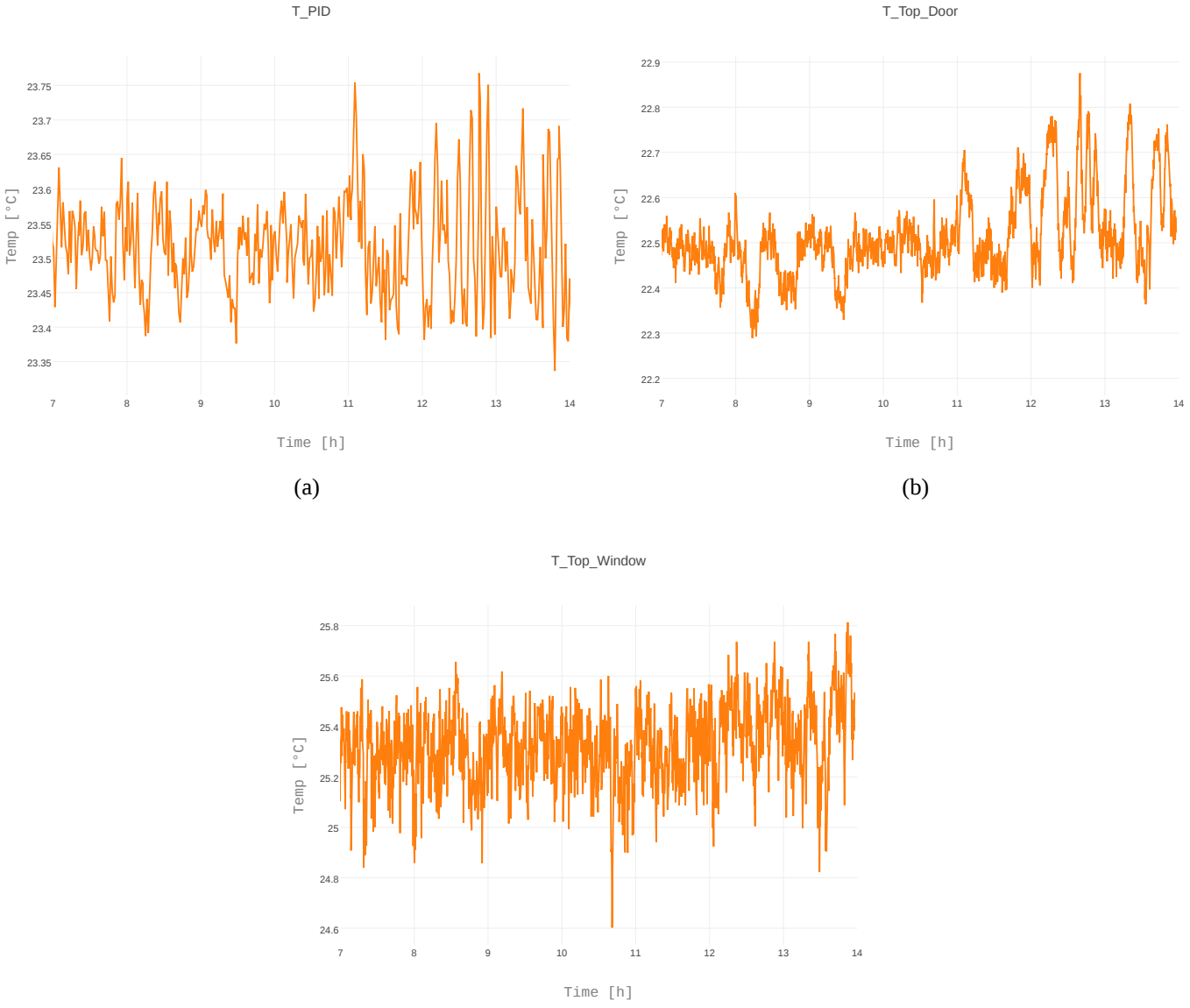



Figure 16: Step response 2 on Top_Door temperature sensor

### 3.2.3 Turning the lights on

At 11 o'clock I turn the lights on in the control room. The reaction at the different sensors is shown below (Figure 17 ), also the set values of the heater/cooler are shown (Figure 18 ).

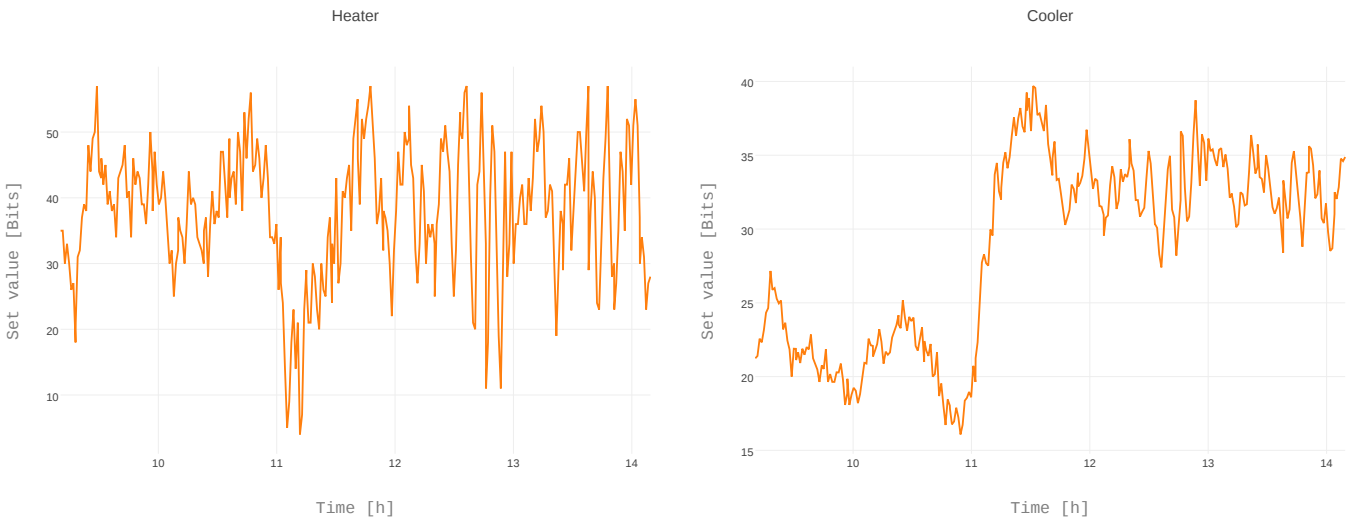



(a) (b)

(c)

Figure 17: Temperature response to light being turned on

Figure 17 shows that the temperature does oscillate more after turning the lights on, especially more to the higher temperatures. Below we see that while the heater set value nearly stays the same the cooler starts cooling more after turning the lights on.




(a) (b)

Figure 18: Heater/cooler set values after turning on the lights

### 3.2.4 Temperature stabilization over several days

Over the weekend of the 24th to 27th I could make a measurement over two and a half days. The measurement starts on the 24th at around 16:30 and ends at the 27th at around 4:30. Here I can also do a histogram for the door and window temperatures, as there is no big jump in them (unlike Figure 11 )
From the histograms (Figure 19 (d-f) ) we get the mean and standard deviations for the PID sensor:

$$\mu = 23.514K \quad \sigma = 0.075K$$

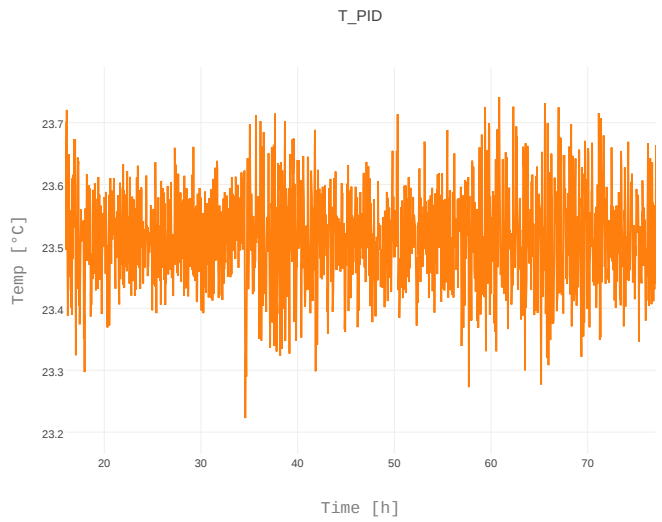
for the Top_Door sensor:

$$\mu = 22.652K \quad \sigma = 0.073K$$

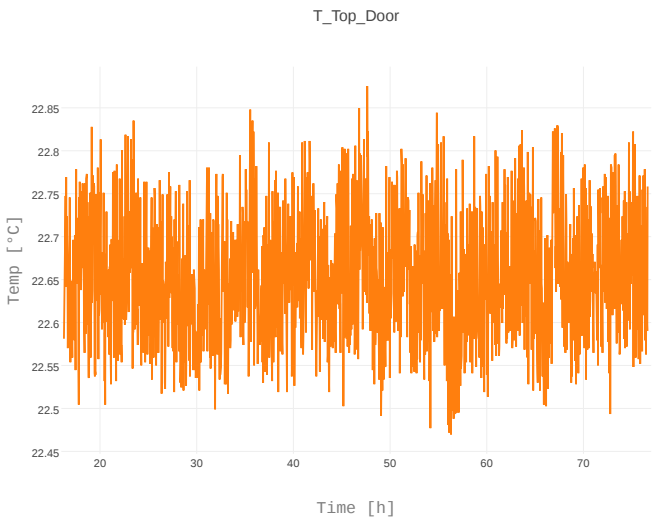
and for the Top_Window sensor:

$$\mu = 24.927K \quad \sigma = 0.267K$$

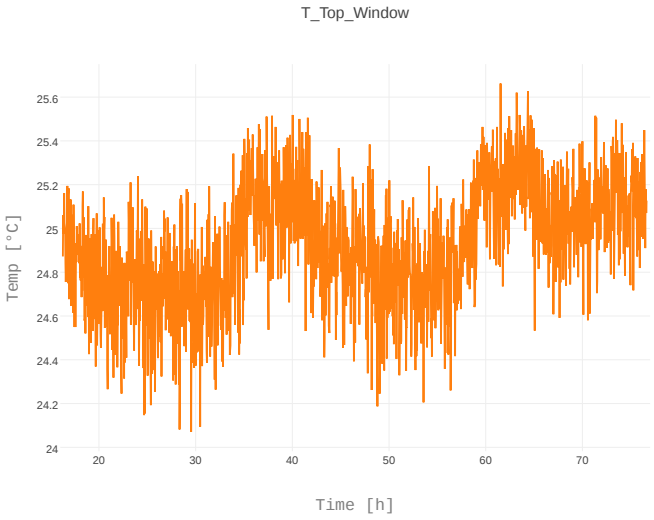
We see that the temperature is fairly stable near the door and the PID temperature sensor ($\sigma = 0.076K$ slightly higher than before), with not even half a degree deviations. The temperature near the window varies more, with a standard deviation of $\sigma = 0.267K$.
Its also visible that the PID sensor seems to show more oscillation on daytime and at the window there is a clear rise in temperature over the day.
What can also be seen is that the door temperature seems to be rather colder than the mean value, meaning it oscillates more to the cold side and the opposite is the case for the window temperature. This can also be seen by calculating the weighted difference of points to the right to points to the left of the mean. If that number is positive it means that we have more points to the left.
Calculation yields the following values:
PID: $9.67\text{e}{-04}$
Door: $1.77\text{e}{-02}$
Window: $-1.05\text{e}{-02}$
which agree with the asumptions.



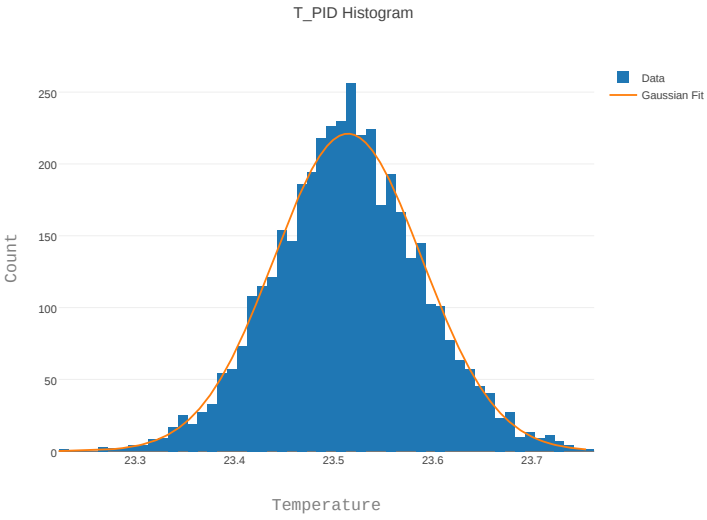

(a)



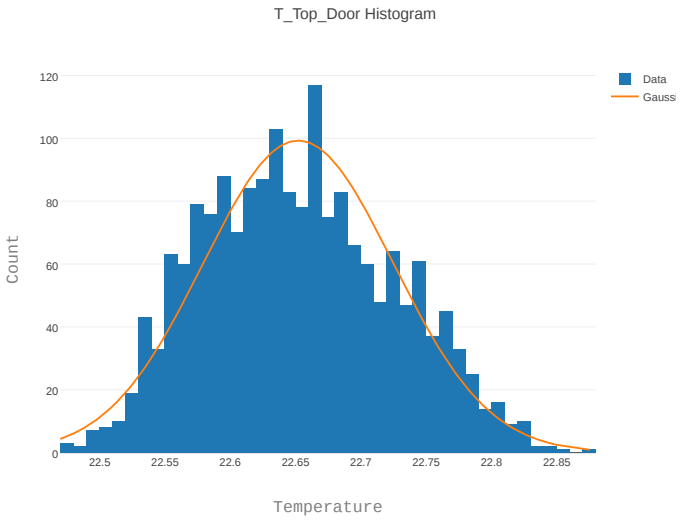

(b)



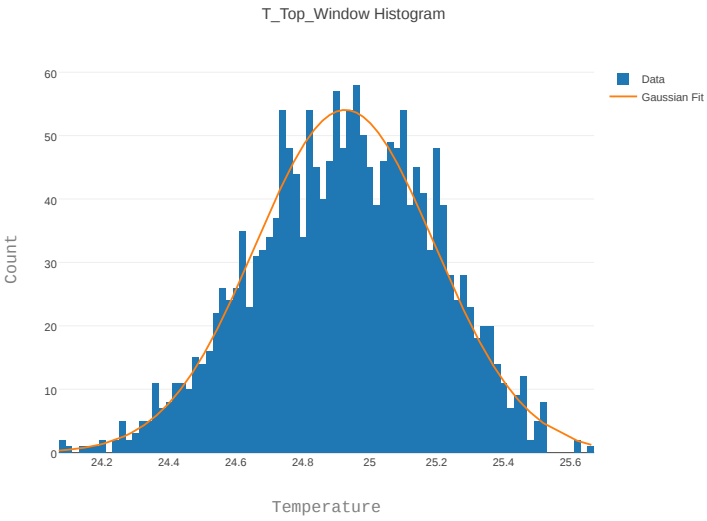

(c)




(d)




(e)




(f)

Figure 19: Temperature stabilization over several days with temp. from sensors T_PID, T_Top_Door, T_Top_Window.