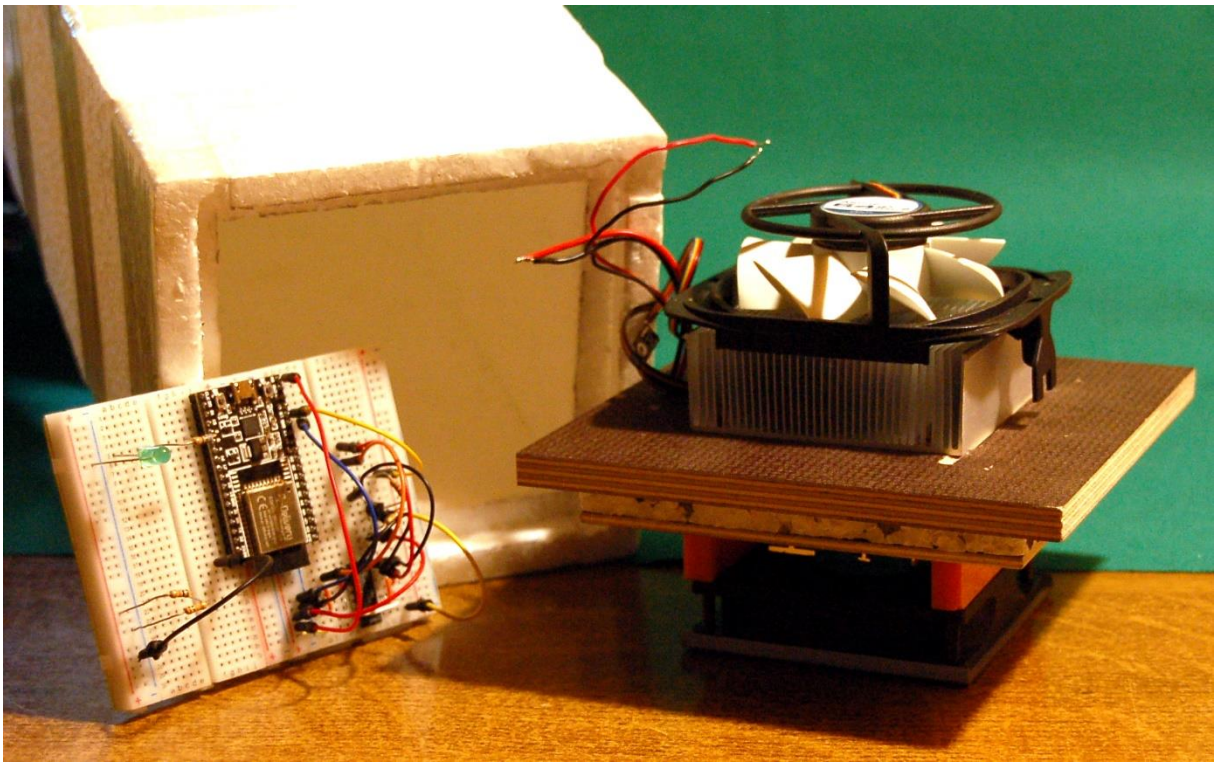

Abbildung 1: Thermobox

In the first episode of this series we built an adjustable power supply for 8A current load. Some readers will have wondered why such currents are needed in connection with microcontroller circuits. The power guzzlers are not the controllers either, but the parts that we are going to use for our bottle cooler today, the Peltier elements. With a 12V supply voltage, a current of 4A already flows through a single element. Our power supply can supply two of them in parallel. The bigger brother of the DC-DC buck converter can even supply three units with a load capacity of 12A. In today's post, I'll tell you how this works in detail and what you have to pay attention to. So welcome to part 3 of the Peltier series entitled

Peltierelements -
The bottle cooler or sub-zero temperatures from the PC power supply

Today we are dealing with the following important points on the subject of cooling.
1. How does a thermoelectric converter work as a heat shovel?
2. How do you build an efficient heat lock?
3. How can you build an inexpensive and stable cool box?
4. How can the ESP32 control and regulate voltages and currents up to 12A?
5. Where can I get the software I need and how can I use it?

The material list contains only a few larger parts, the thermocouple the DC-DC converter as well as an ESP32 and a number of small parts.

1	Thermoelektrischer Wandler 40 x 40 mm
1	DC-DC-Buck-Converter 8A oder DC-DC-Buck-Converter 12A
1	ESP32
1	LED weiß 5mm
1	LDR 5mm
1	20-25mm Messingrohr 6mmØ und 0,5mm Wandstärke
	Dichtmasse
	Schrumpfschlauch Stücke
1	Widerstand 1kΩ
1	Widerstand 10kΩ
1	NPN Standard Transistor zum Beispiel BC550
2	Zweipolige Stiftleiste
1	LED rot
1	Widerstand 560Ω
1	zweipolige Buchsenleiste
1	Breadboard
	diverse Jumperkabel

Für den Thermokopf

2	Rippen-Kühlkörper
2	dazu passende PC-Lüfter
4	Kunststoffwinkel 10 x 10 x 30mm
	Wärmeleitpaste
evtl.	einige Aluplatten Abschnitte (siehe Text)
	wasserfest verleimte Mehrschichtplatte
1	Stück Styroporplatte 126 x 126 x 10
	diverse Schrauben, Muttern,
	Zuleitungskabel für das Peltier-Element mit 1mm²

Für den Kühlbehälter

1	Styroporplatte 30mm
	Paketklebeband

It will be best to read through Chapters 2 and 3 of this manual before you get the material for the box and the thermal head. The possibilities for implementation are so diverse that I cannot possibly address all of them. Your imagination and the resources of your handicraft basement are required. Therefore, understand the following description only as a suggestion for your own approaches.

1. Some theory to warm up

In episode 2 we used the thermoelectric effect (aka Seebeck effect) for the heat flow meter. This consists in the fact that electrons migrate at the contact points between two different metals. If the contact points are heated or cooled, this leads, because of the different speeds of the electrons at the contact points, to an electrical voltage at the point where one of the two conductors is separated. Because the element itself acts as a voltage source, the technical current direction within the element is from the negative to the positive pole.

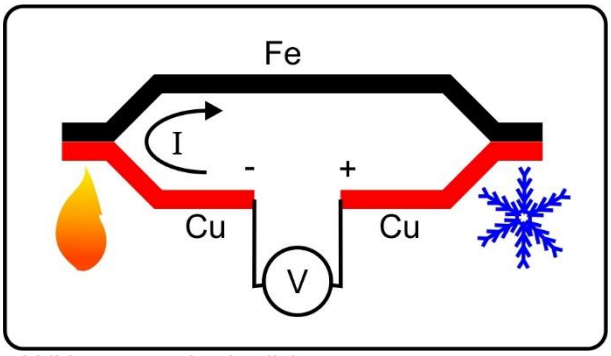

Abbildung 2: Seebeck-Effekt

If you swap cause and effect, you get the Peltier effect. This means that a current flow through the element results in a temperature difference at the contact points. With the same current direction, the contact point that would have to be heated in the Seebeck effect cools down.

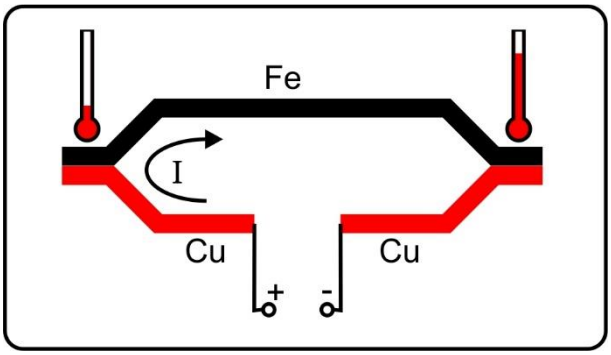

Abbildung 3: Peltier-Effekt

In principle, this also works with iron and copper. However, similar to the Seebeck effect, the effect is considerably increased if semiconductor material is used instead of metals. Our Peltier elements are positively and negatively doped bismuth telluride. 127 small cuboids made of this material are connected in series and lined up between two ceramic plates. The PN junctions are thus parallel on one plate (top), the NP contacts on the other. This adds up the effects of the Peltier effect, one side cools while the other is heated. That leads us straight to point 2 of the preliminary considerations.

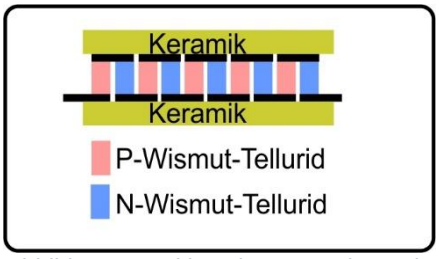

Abbildung 4: Peltier-Element_schematisch

2. How do you build an effective heat lock?

The purpose of our application should be cooling. Cooling simply means that internal energy has to be withdrawn from a body. This energy, together with the not inconsiderable amount of electrical work to be applied, is delivered to the hot side of the element. So that our converter does not reach temperatures that are too high, the solder of the contact points melts at 125 ° C, we need effective heat dissipation, which we ensure by using a PC heat sink with fan. Because there is also a simple principle. The lower the temperature of the waste heat exchanger, the lower the temperatures we reach in the cold store or, in other words, the less electrical energy we have to use for the same cold store temperature. Figure 5 shows a possible setup in which the warm-side fan is still missing. The thermocouple lies between two 4 and 6 mm thick aluminum plates. Please ensure that surfaces are as flat as possible so that good thermal contact can be established between the Peltier element and metal, especially on the warm side. The use of thermal paste is also a very good idea in this context.

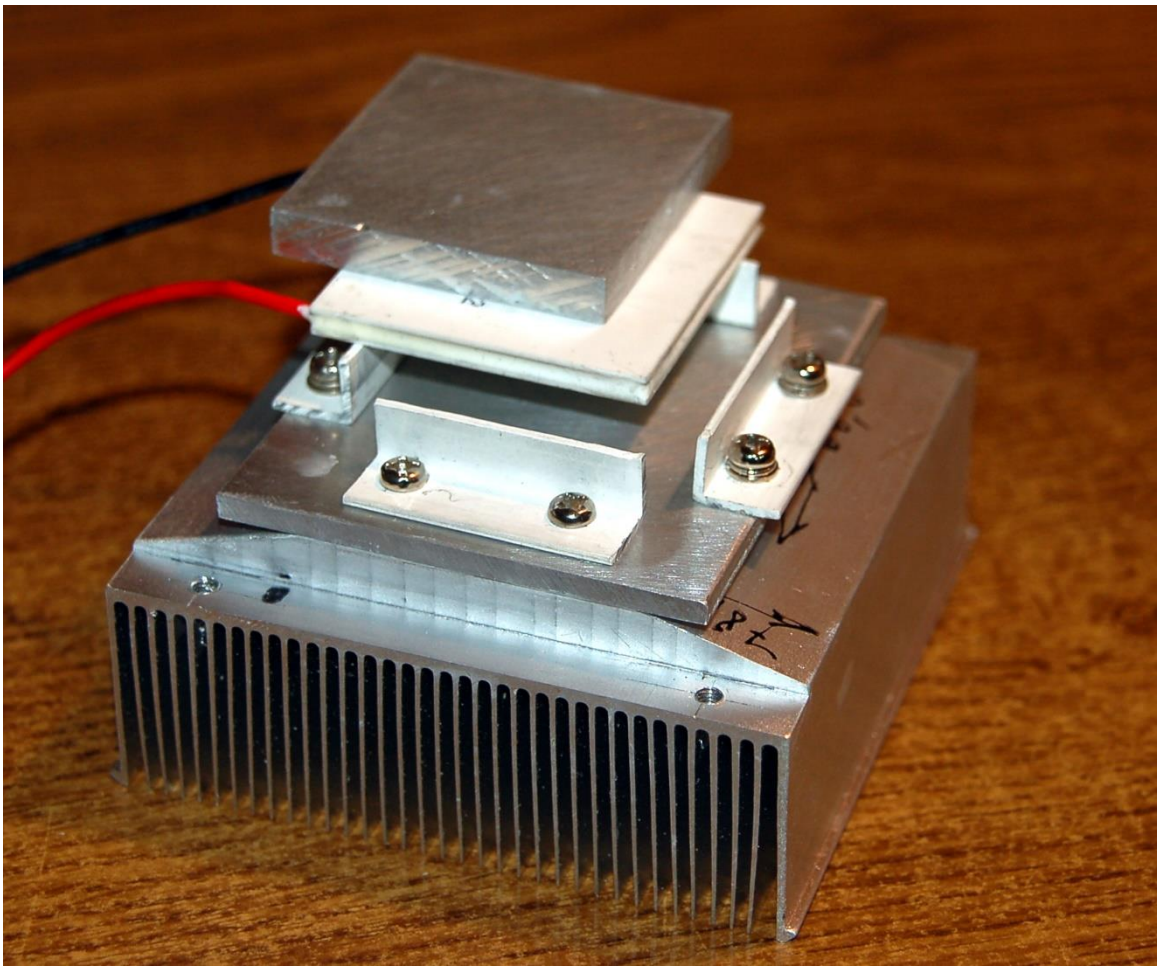

Abbildung 5: Anordnung der Teile

How and from what you build the unit largely depends on what you can find in your craft kit or at the scrap dealer. But one thing is important. The ceramic plates of the Peltier element must at least be covered by the metal bodies, or better slightly covered. That was the reason why I screwed an aluminum plate onto the heat sink when I was building it. The surface of the heat sink was namely 3mm too narrow.

So that the Peltier element cannot slip during assembly, it is held in position by the four plastic brackets. Aluminum angles are unsuitable because they represent thermal bridges. This is exactly where we have the greatest temperature differences in our structure. Oh yes, and the difference between the hot and cold side may not exceed 65 ° C, regardless of the maximum temperature of 125 ° C, according to the element's data sheet.

What applies to the warm side is also important for the cold side. So that the thermal energy can be collected in the cooling container as effectively as possible, a cooling body with a fan is also provided here.

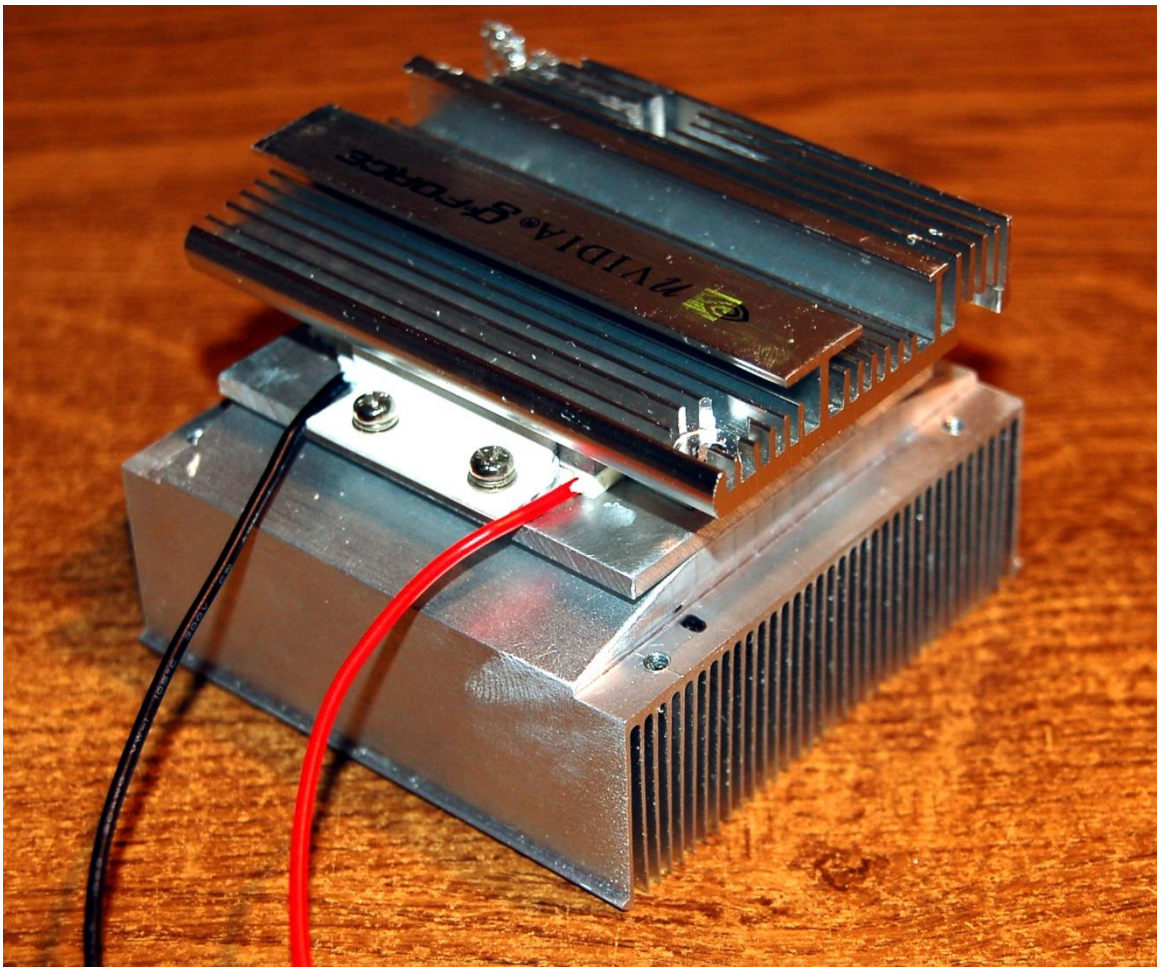

Abbildung 6: Thermokopf komplett (ohne Lüfter)

- The entire heat exchanger head, still without a fan, is shown in Figure 6. The individual layers are shown schematically in Figure 7. Because of the multitude of heat sinks and fans, your result will very likely be different. In all cases, only three things are important.
-
- • Good thermal conductivity transitions between the layers in the heat flow
- • At least 1 cm thick insulation between the heat exchangers on the hot and cold sides
- • Sufficiently dimensioned cooling on the warm side

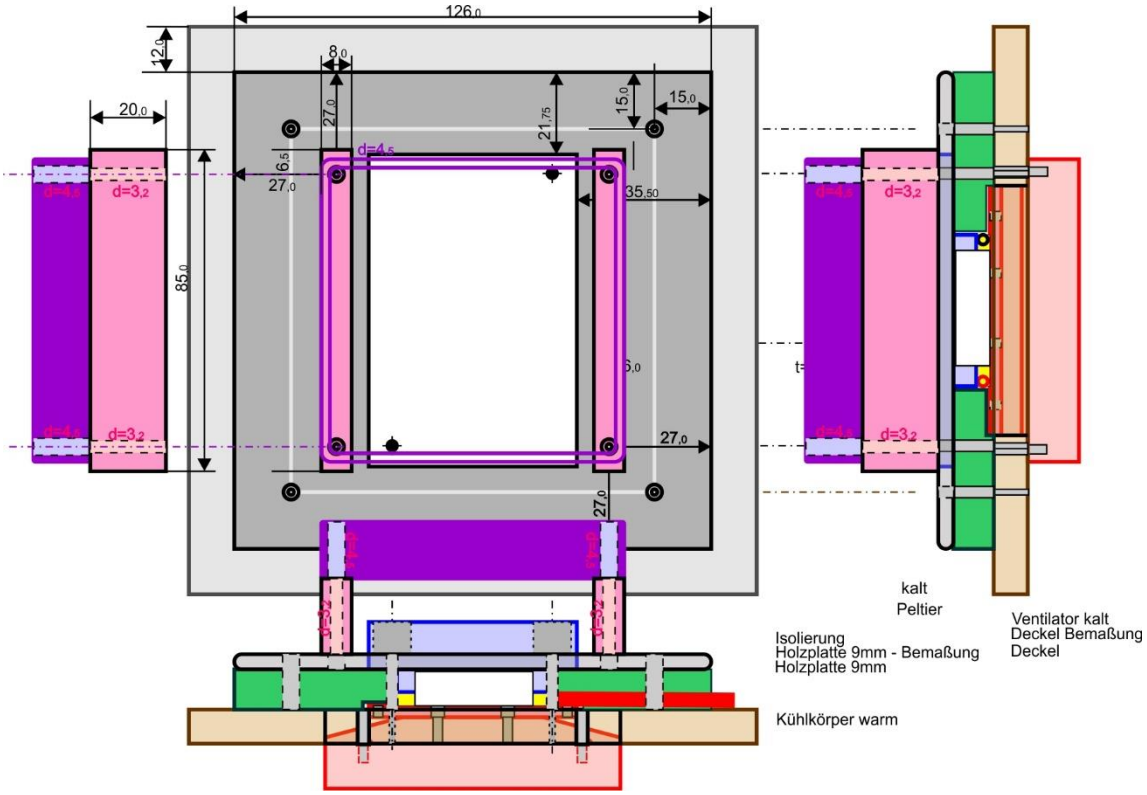

Abbildung 7: Lüfter Kaltseite

By clicking on Figure 7 you can download the archive of all work drawings for the thermal head. This includes 6 pages DIN A4, which show the career on a 1: 1 scale.

The direction of the heat transport is important for the installation of the Peltier element. If you put the red wire to the positive terminal and the black wire to the negative terminal of a 5V source that can deliver at least 2 A for a few seconds, you will find that the printed side gets cold. The other ceramic surface becomes increasingly warmer. Do not prolong the experiment too long, as you know that temperatures that are too high are harmful to the element. The experiment revealed which area is the warm side. This side belongs to the large heat sink.

3. How can you build an inexpensive and stable cool box?

So that the box is also light, affordable and easy to build ... we take a 30 mm styrofoam sheet and cut or saw it into four 17 cm wide strips. We provide each of the strips with cutouts, as can be seen in Figure 8. A step is cut out at the upper end so that the thermal head can be lowered down to the carrier plate in the finished container. We cut a square with an edge length of 11 cm from the rest of the styrofoam sheet. It fits exactly between the side parts and fixes them in place when we tie them together with the adhesive tape. As far as possible, there should be no gaps between the plates. The outer base plate with 17 x 17 cm is fixed several times with adhesive strips on the side surfaces, and then we are already done. The interior with 11 x 11 x 47 cm can accommodate 2L plastic bottles up to 11 cm in diameter.

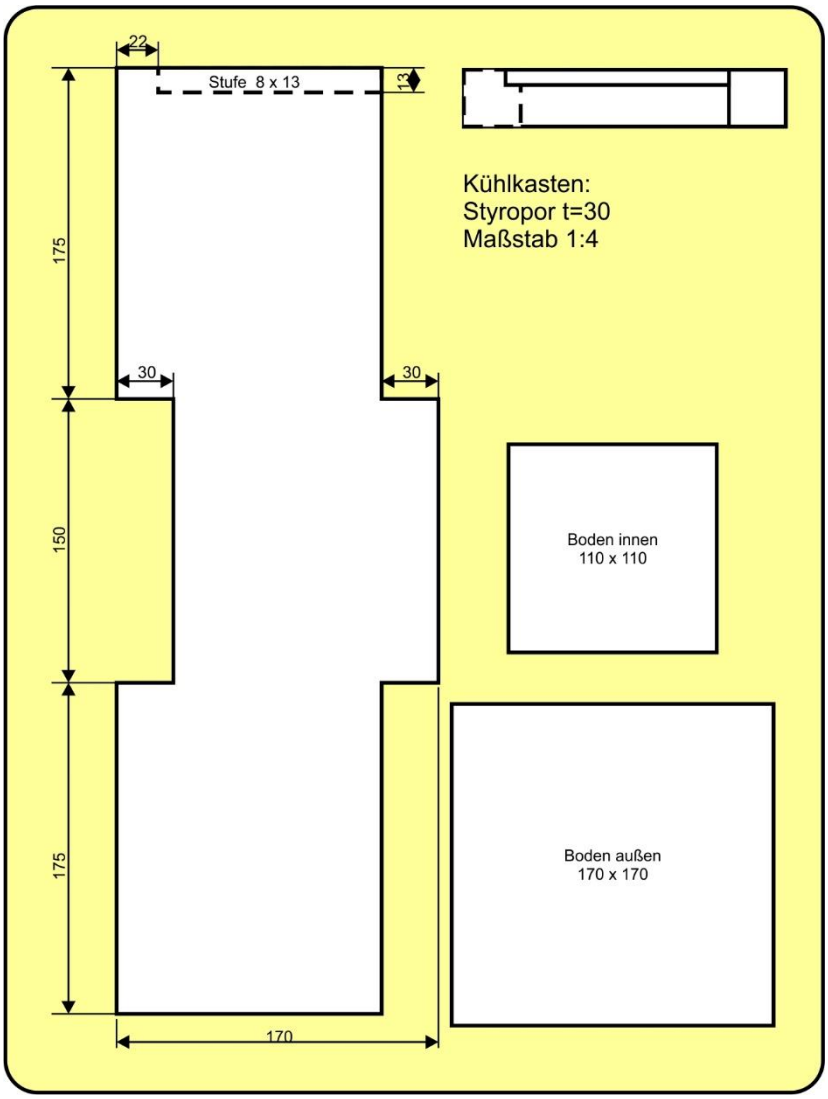

Abbildung 8: Thermobox_Schnittmuster

4. How can the ESP32 control and regulate voltages and currents up to 12A?

The cooling performance depends essentially on two factors. One is the temperature of the heat exchanger on the warm side and the second is the strength of the current through the Peltier element. The latter in turn depends on the applied voltage, because the internal resistance of the Peltier element is, roughly speaking, a constant. Thus, according to the resistance formula I = U / R.

If we want to keep the temperature in the cold room at a certain level, we only have to suck out the exact amount of heat that penetrates through the wall. If, in another case, we also want to remove the amount of heat that is in the drink and in the bottle, we of course need a significantly higher use of electrical energy.

A regulation can now simply consist in switching the current through the Peltier element on and off at certain time intervals. However, you can also set the current strength so that a constant temperature results or, in the second case, first set it to

full power and then reduce it after the target temperature has been reached. I chose the more interesting second solution.

But how can an ESP32 handle voltages up to 16 V and currents up to 12 A? One tool is a DC-DC step down converter (aka buck converter), which can regulate voltages very well, which can be specified using a trimming potentiometer. - Oh yes, of course - and the ESP32 then turns the potentiometer or what? No, he doesn't have to. However, it can control the brightness of an LED via PWM pulses ... which in turn illuminates a light-sensitive resistor (LDR Light Dependent Resistor), ... which is switched in parallel to the trimming potentiometer on the controller board.

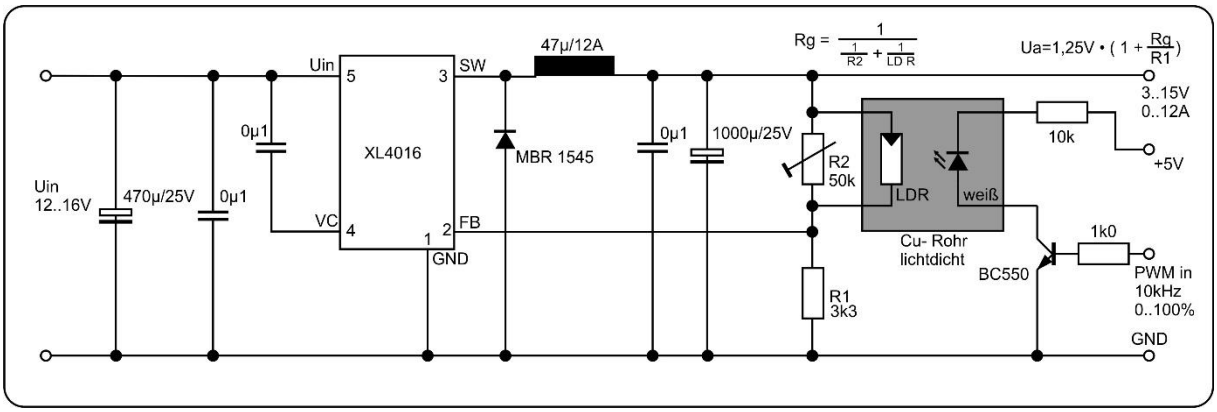

Abbildung 9: Buck-Converter Schaltplan

We set the potentiometer to the right stop and then turn it back so far that the voltage just begins to decrease when the LDR is completely darkened (Umax) or reaches the maximum permissible value of 16V. Then we can control the voltage at the output of the controller between approx. 3V and Umax via the ratio of pulse length to period of the PWM signal (aka duty cycle) - unfortunately not linear. In the next episode we will also teach the ESP32 how to measure currents, and then we will be able to regulate currents via the voltage control. This makes the controller a real all-rounder. How to get the problem with linearity under control I will also reveal in the next episode.

LDR and the white LED together form a self-made optocoupler. The two components come in a piece of 6mm brass tube to shield outside light. The wall thickness of 0.5mm results in a clear width of 5mm. We file off the thicker edge of the LED so that it fits completely into the tube, the LDR can be countersunk from the start without any problems. We solder short pieces of cable to the connections of both components and slide pieces of shrink tubing over them for insulation. Then we close the openings with modeling clay and equip the cable ends with a two-pole pin header.

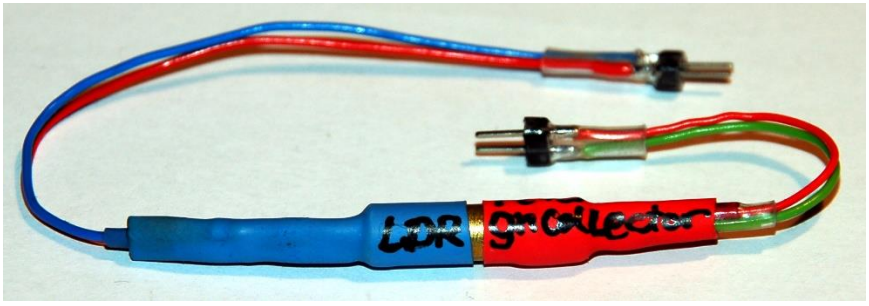

Abbildung 10: Spezieller Optokoppler

The LDR has some properties that are very useful to us. The response time is 1 to 3 ms. This "smears" the pulses of the PWM signal into a smooth resistance value if the frequency is greater than 1000Hz. The light resistance is 200 ohms, the dark resistance over 2 MΩ. Unfortunately, the area in between cannot be fully used for the following reasons.

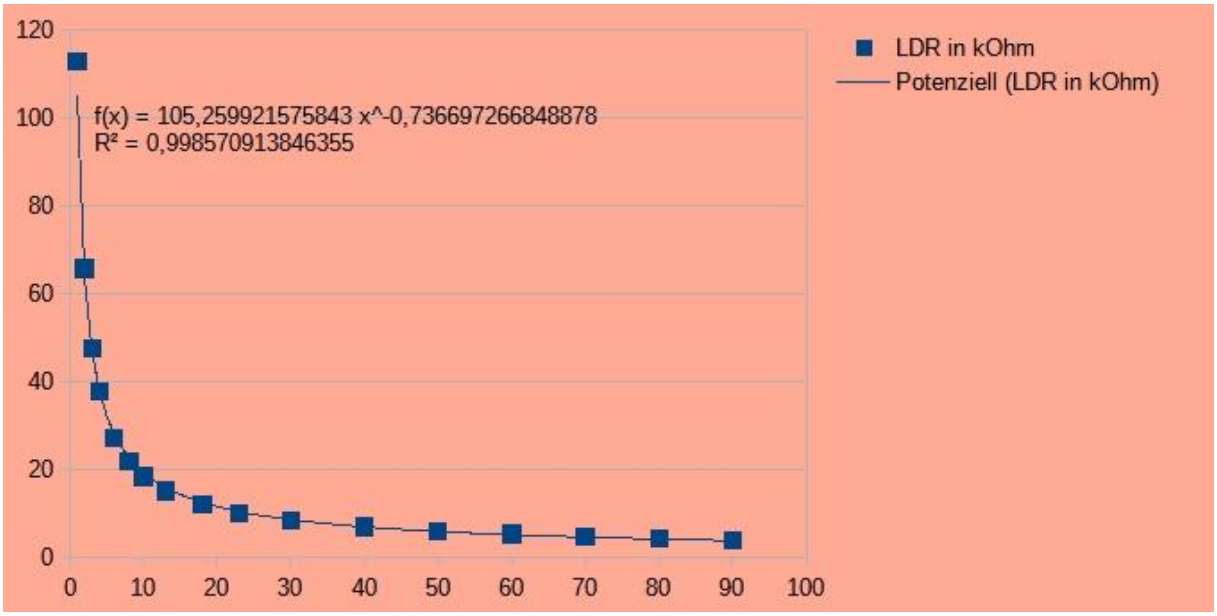

Abbildung 11: LDR-Widerstand gegen PWM-Duty-Cycle

The resistance of the LDR decreases linearly with the illuminance, but the latter is not linearly related to the duty cycle of the PWM signal. This is confirmed by a series of tests, the result of which can be seen in Figure 11. In order to brake the rapid drop in the characteristic curve at low duty cycles, I increased the series resistance of the LED from 560 ohms to 10 kΩ. This means that the illuminance does not increase as quickly at low percentages, and the curve becomes flatter. At the end of the scale, however, less illuminance also leads to a higher residual resistance in the LDR because less light now arrives.

Calculating the resistance of the parallel circuit from R2 and the LDR doesn't make things any less complex. The equation of the power function, which the trend line follows, is therefore also included in the relationship between output voltage and duty cycle. And at 100% duty cycle, the higher LDR value results in a minimum voltage of approx. 3V. For this application, however, we can live with it.

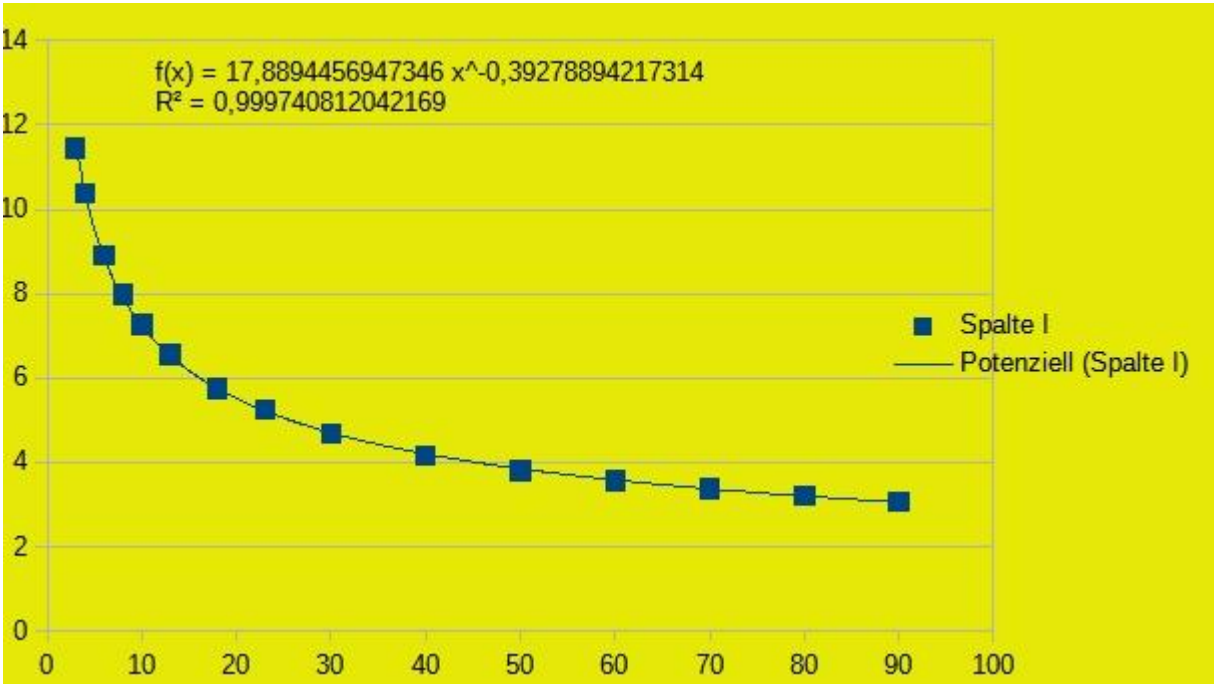

Abbildung 12: Ausgangsspannung gegen Dutycycle

The equation of the trend line for the output voltage is a fractional rational power function. We will incorporate this knowledge into the control program for the ESP32 in the next episode.

$$Ua = 17{,}8894 \cdot \left(\frac{1}{D}\right)^{0{,}39279}$$

Abbildung 13: Gleichung der Trendlinie

Of course, the parameters, factor and exponent in the equation depend on the individual circumstances of the structure of the optocoupler. That is why we will soon develop a program with the help of which we can record the corresponding characteristic curve. But first the circuit has to be created on the breadboard, otherwise there is nothing to measure. The places where pieces of wire to the LDR have to be soldered to the board are marked in yellow in the illustration.

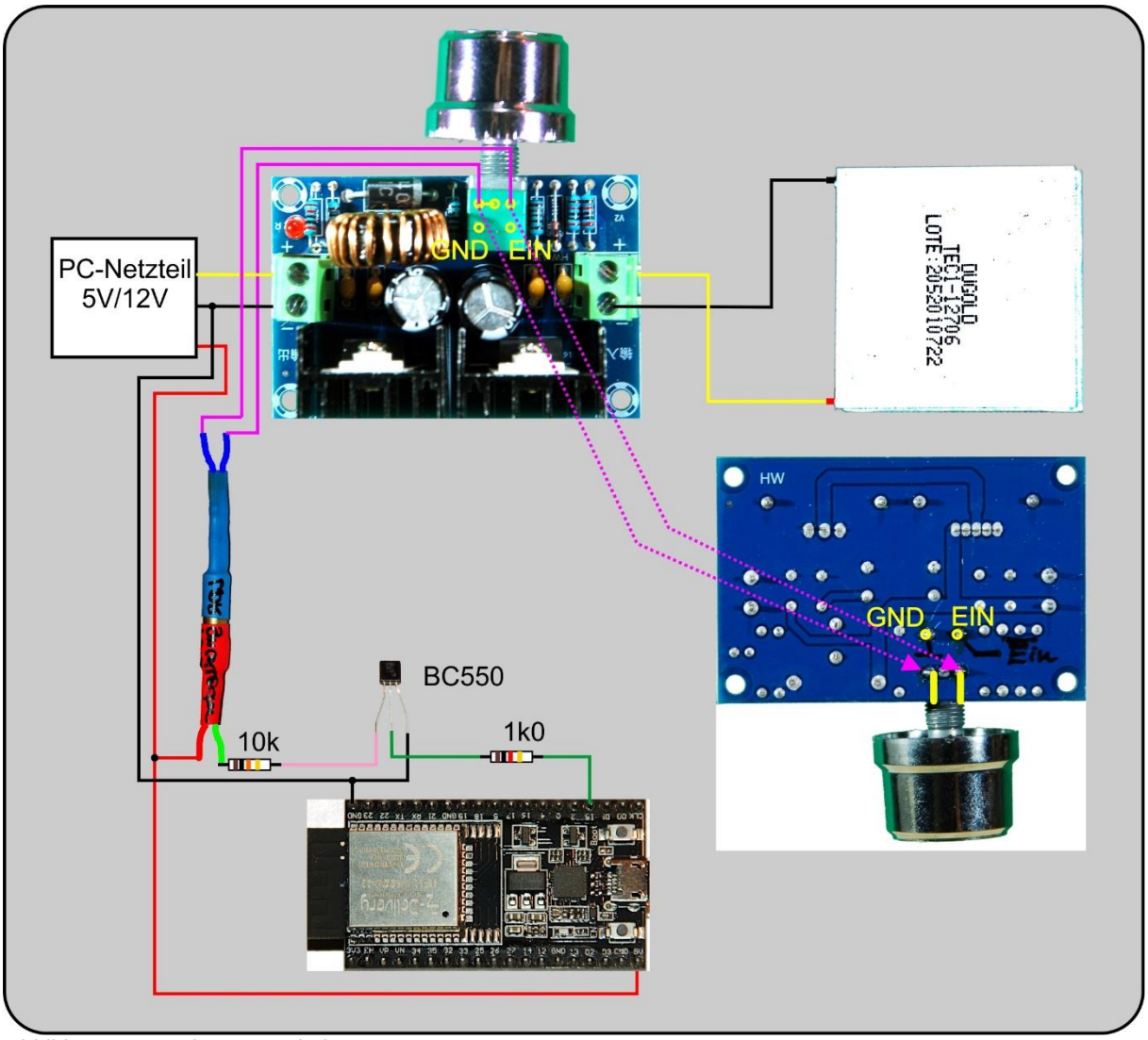

Abbildung 14: Regler 8A - Schaltung

We are now ready to collect the data for the characteristic curve. We do this in idle mode, which means that we have not yet connected the Peltier element in the thermal head.

The program **reglertest**.py that we need for this is very simple. We import a couple of classes, create a pin object for the flash button and an output pin for an LED. The blink () function defined below allows us to easily program blink signals for status messages. If the LED is on the plus rail and is pulled from the pin to ground in order to light up, the parameter inverted must be set to True. The default value False assumes that the LED is on GND level and is switched on by a HIGH level on the pin.

```
# reglertest.py
# Author: J. Grzesina
# Rev: 1.0
# Stand: 2021-08-24
# ******************************************************
from machine import Pin,PWM
from time import sleep
import os,sys
```

```
taste=Pin(0,Pin.IN,Pin.PULL_UP)
blinkLed=Pin(2,Pin.OUT)

# Pintranslator fuer ESP8266-Boards
# LUA-Pins     D0 D1 D2 D3 D4 D5 D6 D7 D8
# ESP8266 Pins 16  5  4  0  2 14 12 13 15
#                 SC SD

pwmPin=15               # PWM-Pin zur Spannungssteuerung
pwm=PWM(Pin(pwmPin))
pwm.freq(10000)         # PWM-Frequenz
pwm.duty(1023)          # volles Rohr für minimale Spannung

# ******************************************************
# Funktionen
# ******************************************************


def setDuty(d):
    pwm.duty(d)


def blink(pulse,wait,inverted=False):
    if inverted:
        blinkLed.off()
        sleep(pulse)
        blinkLed.on()
        sleep(wait)
    else:
        blinkLed.on()
        sleep(pulse)
        blinkLed.off()
        sleep(wait)


while True:
    r=input("Dutycycle %:")
    r=int(float(r)/100*1023)
    setDuty(r)
    print(r)
    if taste.value()==0:
        print("Mit Flashtaste abgebrochen")
        sys.exit()
    blink(0.1,0.9, inverted=True)
    #sleep(1)
```

One of the key points of this sample program is how to define and set up the PWM pin. With the ESP32, all output pins are also PWM-compatible. The PWM frequency can be up to 40MHz. The duty cycle is specified by a value between 0 and 1023, i.e. not in percent.

```
pwmPin = 15 # PWM pin for voltage control
pwm = PWM (Pin (pwmPin))
pwm.freq (10000) # PWM frequency
pwm.duty (1023) # full pipe for minimal tension
```

The XL4016 on the controller board always sets the output voltage Ua so that the voltage from the voltage divider consisting of R2 and R1 at the feedback input FB is 1.25 V. The minimum voltage results when the parallel connection of LDR and R2 reaches the lowest value. Because R2 is no longer changed, this is the case with maximum brightness of the LED and thus with 100% duty cycle. However, because the parameter value has to be selected from the range 0 to 1023, the percentage values must be converted after the entry. This happens in the endless loop, which can be left after an entry by pressing the Flash button.

```
r = input ("Dutycycle%:")
r = int (float (r) / 100 * 1023)
setDuty (r)
```

Now, after entering various percentage values, we can measure the resulting output voltage with a digital multimeter and enter it together in a table. The graphic evaluation with LibreOffice Calc finally provides us with the desired formula with the parameters.

With the same program we can of course also test our thermal head. So that the Peltier element does not overheat, we also operate the fan on the warm side at the same time. It is connected directly to 12V so that it can always start properly, regardless of the voltage on the Peltier element.

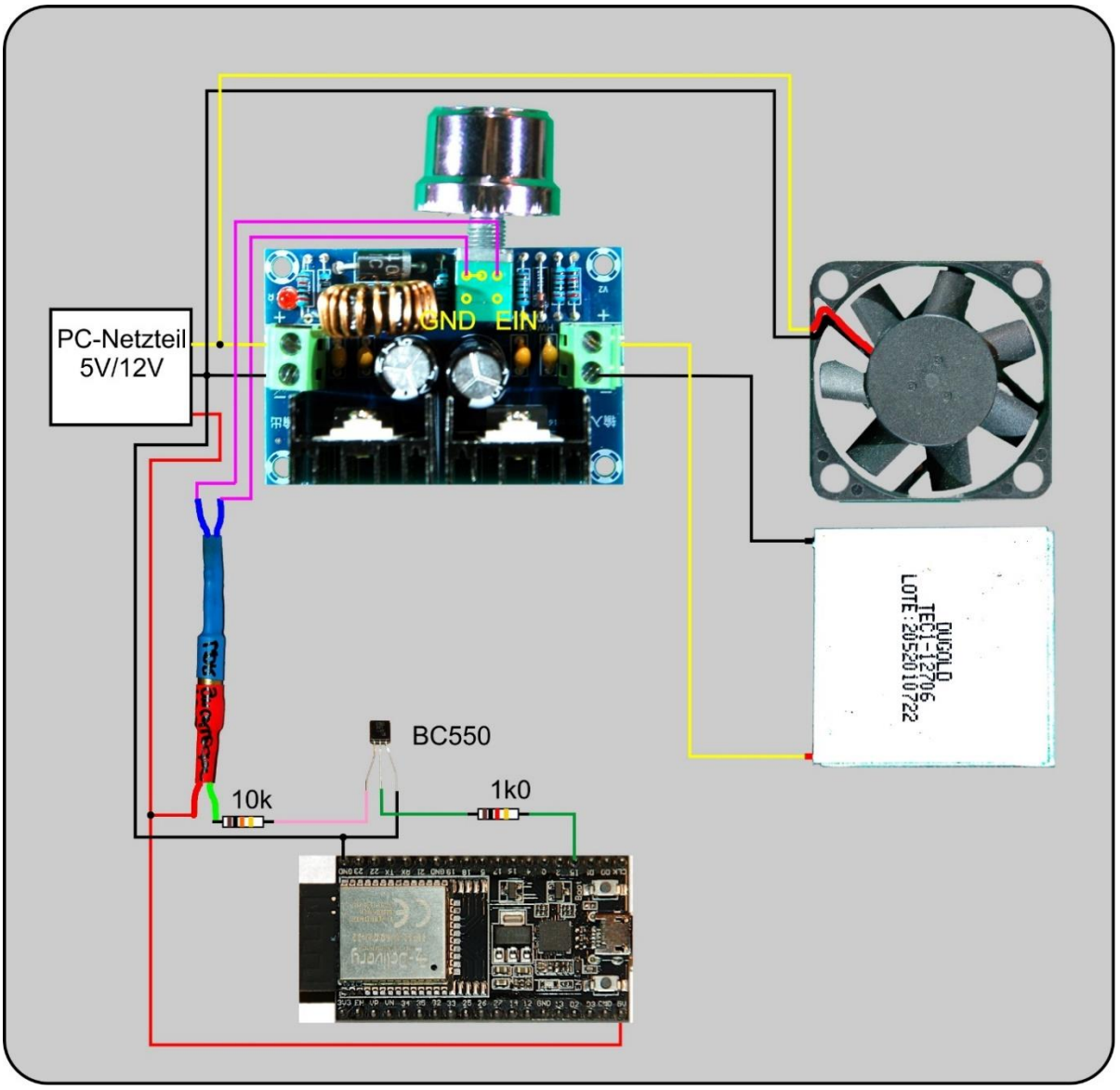

Abbildung 15: Regler 8A - Schaltung mit Lüfter

With my setup, I was able to determine a temperature of -6 ° C on the heat sink on the cold side in about 5 to 6 minutes, while the warm side heat sink reached and held about hand heat.

Software

For flashing and programming the ESP32:
Thonny oder
µPyCraft
packetsender for testing the ESP32 as a TCP/UDP server
ncat im Paket nmap For testing the controller as a TCP/UDP Client

Used Firmware:

MicropythonFirmware
Please choose a stable version

MicroPython-Programs:
reglertest.py

MicroPython - Language - Modules and Programs

You can find detailed instructions for installing Thonny here. There is also a description of how the Micropython firmware is burned onto the ESP chip.

MicroPython is an interpreter language. The main difference to the Arduino IDE, where you always and exclusively flash entire programs, is that you only have to flash the MicroPython firmware once at the beginning on the ESP32 before the controller understands MicroPython instructions. You can use Thonny, µPyCraft or esptool.py for this. I have described the process for Thonny here.

As soon as the firmware is flashed, you can have a casual conversation with your controller, test individual commands and immediately see the answer without first having to compile and transfer an entire program. This is exactly what bothers me about the Arduino IDE. You simply save an enormous amount of time if you can do simple tests of the syntax and hardware through to trying out and refining functions and entire program parts via the command line before you knit a program out of it. For this purpose I also like to create small test programs over and over again. As a kind of macro, they combine recurring commands. From such program fragments, entire applications can develop.

Autostart

If the program is to start autonomously when the controller is switched on, copy the program text into a newly created blank file. Save this file under boot.py in the workspace and upload it to the ESP chip. The program starts automatically the next time it is reset or switched on.

Testing programs

If the program is to start autonomously when the controller is switched on, copy the program text into a newly created blank file. Save this file under boot.py in the workspace and upload it to the ESP chip. The program starts automatically the next time it is reset or switched on.

In between, Arduino IDE again?

If you want to use the controller together with the Arduino IDE again later, simply flash the program in the usual way. However, the ESP32 / ESP8266 then forgot that it ever spoke MicroPython. Conversely, every Espressif chip that contains a compiled program from the Arduino IDE or the AT firmware or LUA or ... can easily be provided with the MicroPython firmware. The process is always as described here.

Ausblick

In der nächsten Folge werden wir ein MicroPython-Programm für den ESP32 entwickeln, wodurch er die Versorgung von bis zu drei der heute vorgestellten Kühleinheiten übernehmen kann. Für drei Einheiten brauchen wir dann allerdings einen leistungsfähigeren Buck-Converter, der bis zu 12 A liefern kann.

Bis dahin wünsche ich viel Freude beim Aufbau und Test der Kühlbox.

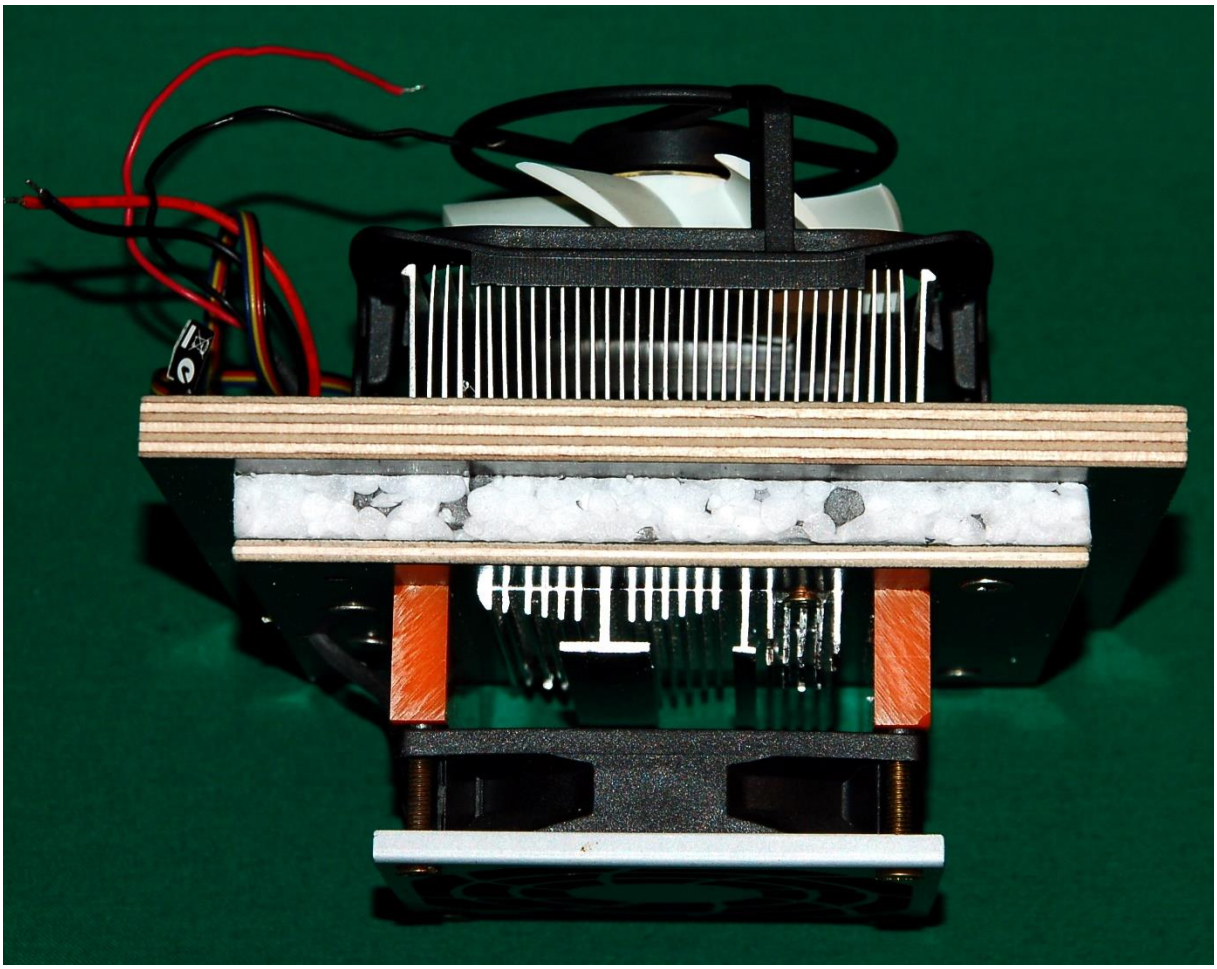

Abbildung 16: Thermokopf komplett