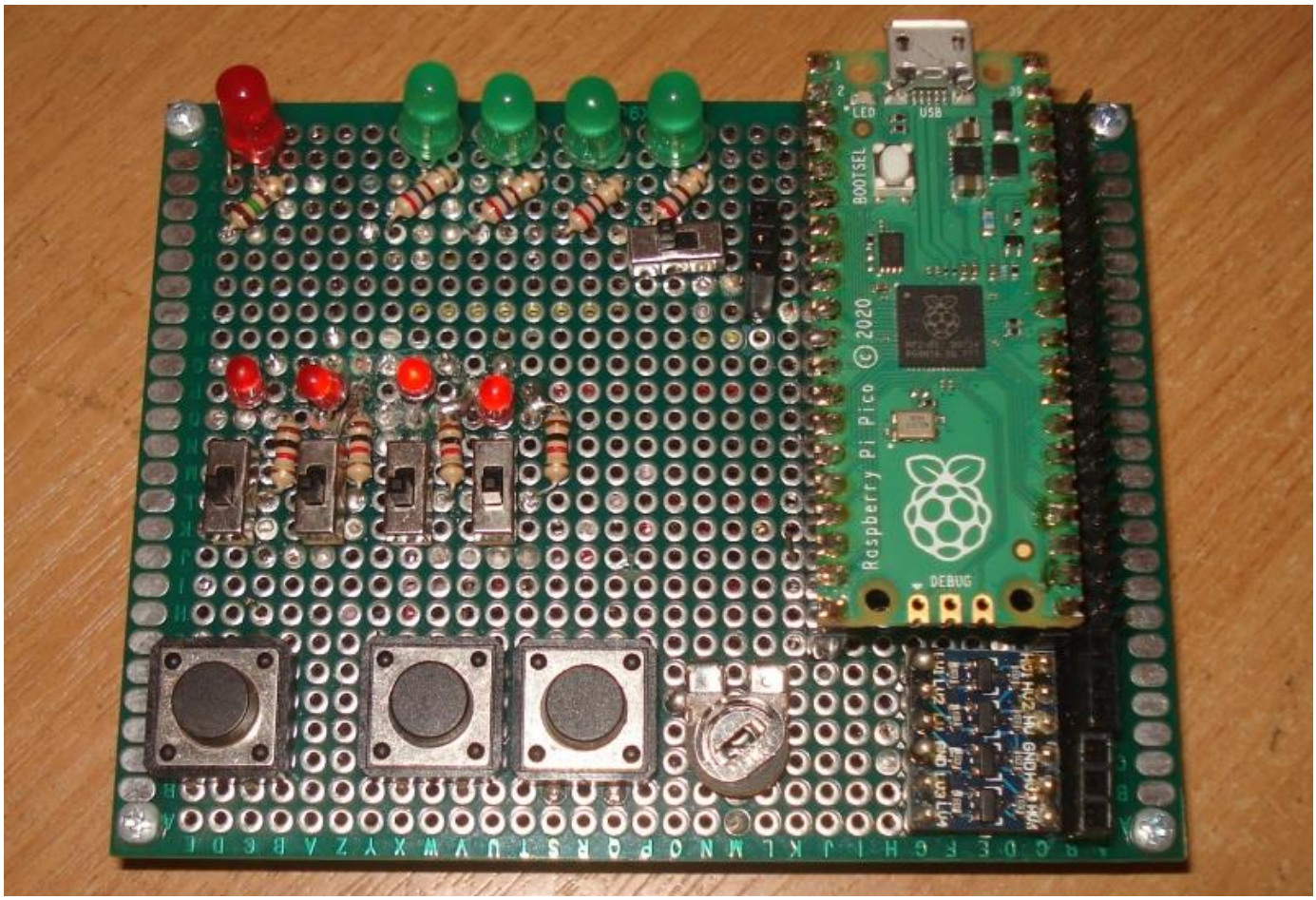
https://www.elektronik-labor.de/Raspberry/PicoTPS.html

Raspberry Pi Pico as TPS / MyCo

Elektronik-Labor Projekte Mikrocontroller Raspberry



Software: **Mycopico_code_2.zip**

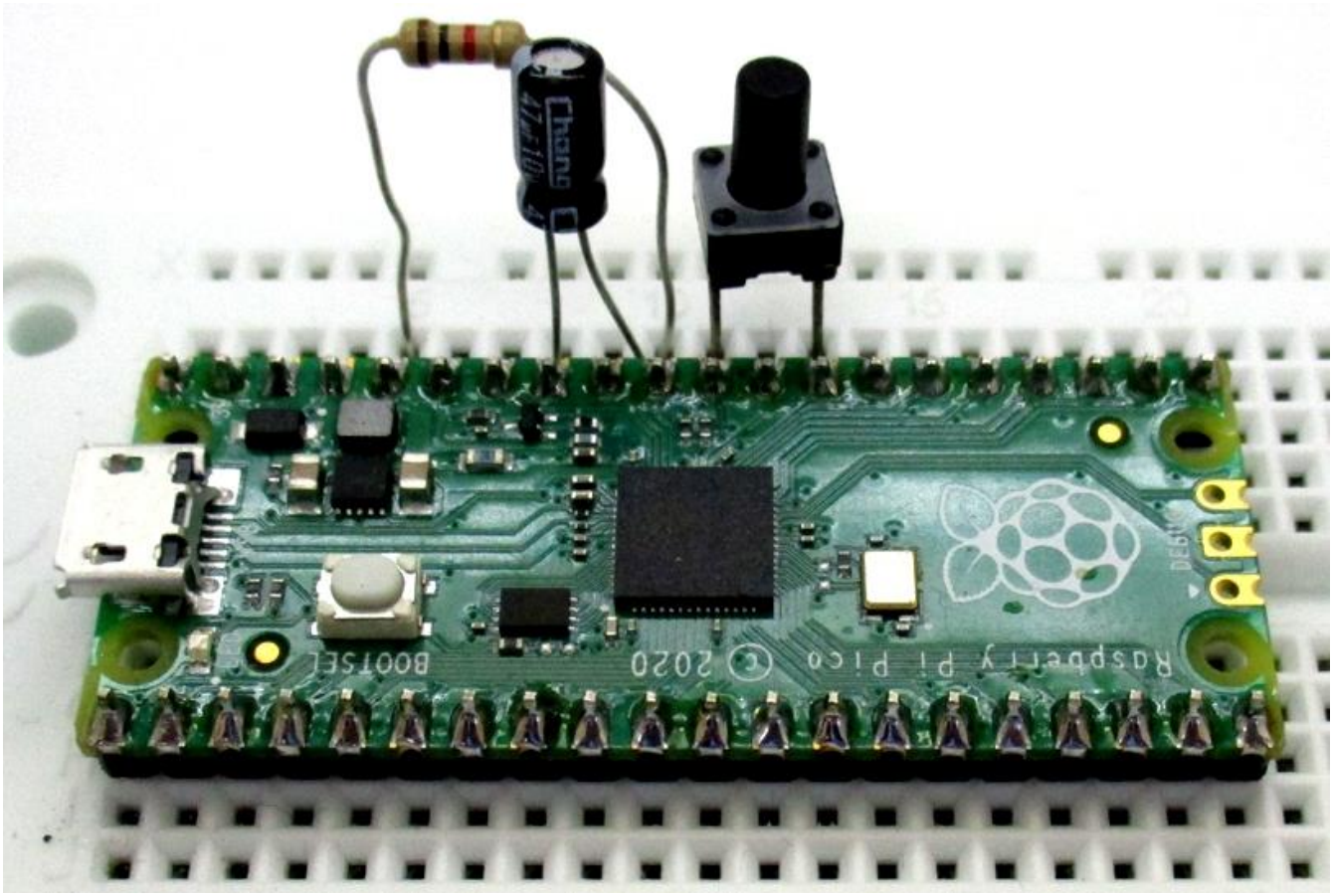
This implementation of a TPS system on a Raspberry Pico comes from Robert Mitchell in England and is called MicoPico.
Jürgen Pintaske has dealt intensively with the TPS and translated the term for the English-speaking area to MyCo. It was Juergen who drew my attention to Robert's work.
Jürgen has started a public Facebook page that reports on the progress of the TPS projects:
https://www.facebook.com/groups/269499491046124

Robert sent me the current status of the project, so that I could try out the software. I had already put my Pico on a breadboard and provided it with a reset button. The test should actually be very easy, but in the end it was a challenge that I couldn't manage without the expert help of Fabian.

Diese Implementierung eines **TPS**-Systems auf einem Raspberry Pico stammt von Robert Mitchell aus England und trägt den Namen MicoPico. Jürgen Pintaske hat sich intensiv mit der

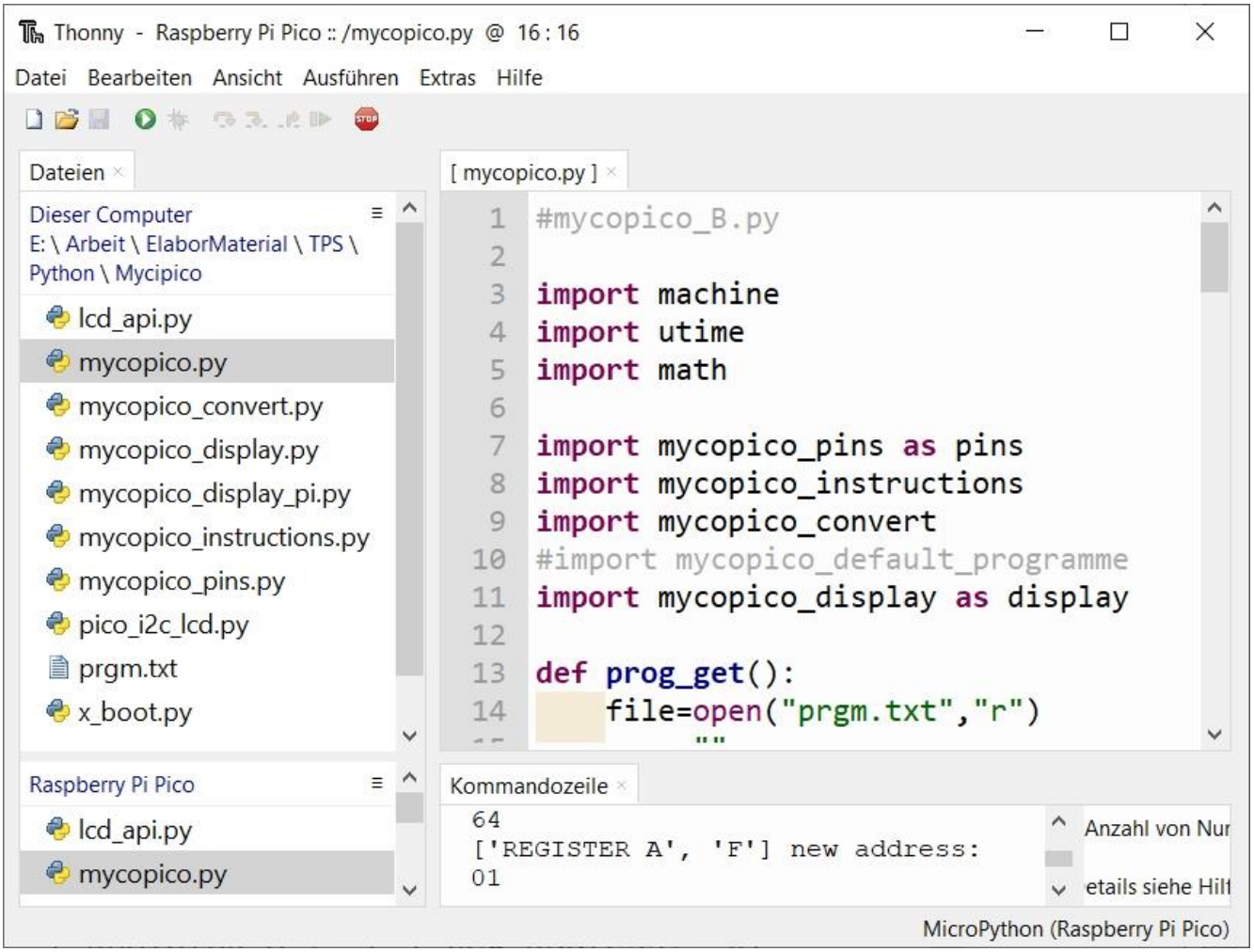
TPS beschäftigt und den Begriff für den englischen Sprachraum in MiCo übertragen. Er war es auch, der mich auf Roberts Arbeit aufmerksam gemacht hat. Jürgen hat eine öffentliche Facebook Seite gegründet, die über den Fortgang der TPS-Projekte berichtet: **https://www.facebook.com/groups/269499491046124**

Robert hat mir den aktuellen Stand des Projekts geschickt, sodass ich die Software ausprobieren konnte. Ich hatte ja schon meinen Pico auf ein Steckboard gesetzt und mit einem Reset-Taster versehen. Eigentlich sollte der Test also ganz einfach sein, aber es war letztlich doch eine Herausforderung, die ich nicht ohne die fachkundige Hilfe von **Fabi** geschafft habe.



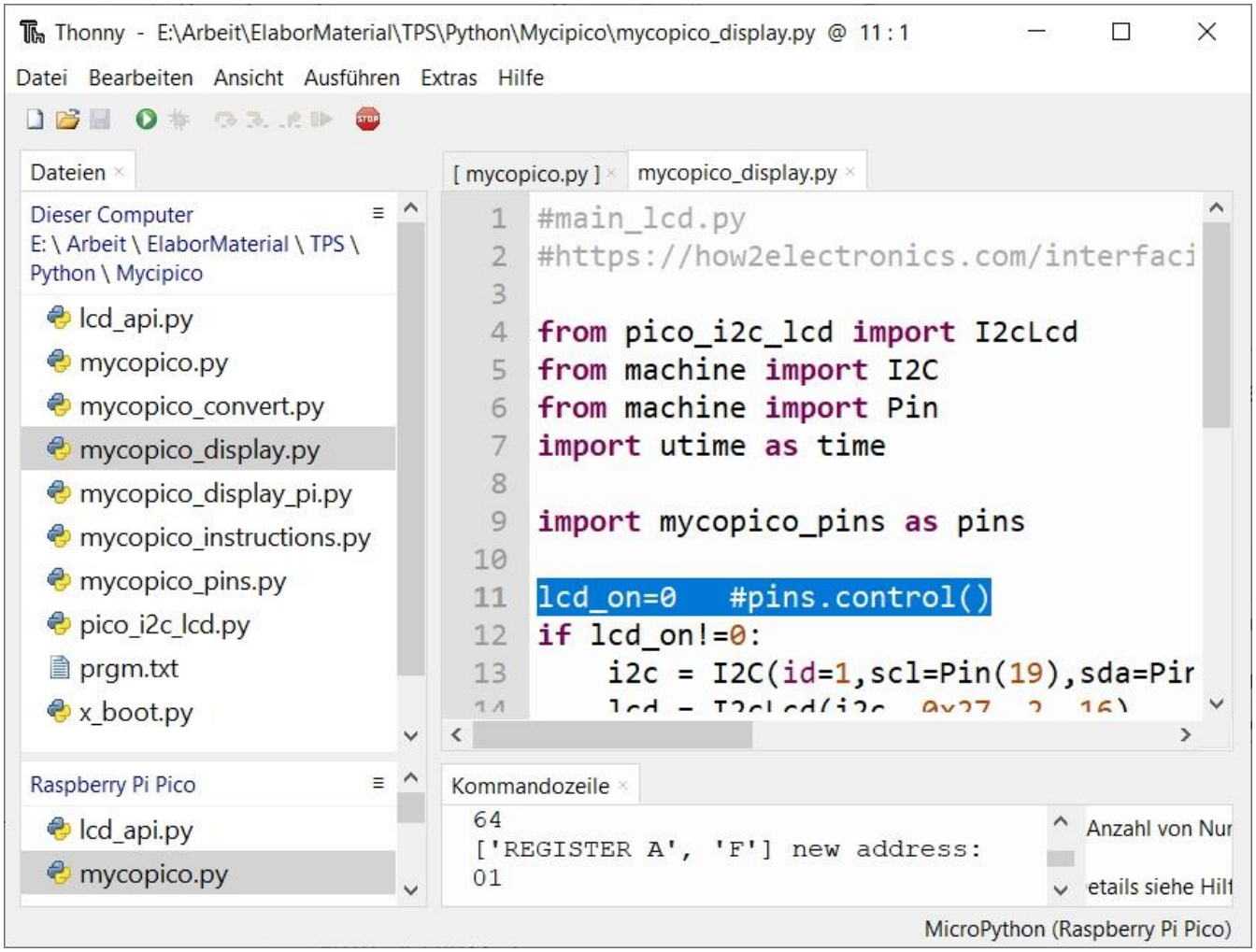
Robert split the project into several files, which makes it very clear.
So far, I had only worked with MicroPython programs where everything was in one file.
So, I had no idea how to deal with the files to be imported - an embarrassing rookie situation.
Fabian showed me how to proceed:
You first have to transfer all files to the Pico, only from there it can reload them.

Robert hat das Projekt in mehrere Dateien aufgeteilt, was es sehr übersichtlich macht. Ich hatte bisher aber nur mit MicroPython-Programmen gearbeitet, bei denen alles in einem File lag. Deshalb hatte ich keine Vorstellung davon, wie man mit den zu importierenden Dateien umgehen musste, ein peinlicher Anfängerfehler. Fabi hat es mir gezeigt: Man muss alle Dateien zuerst in den Pico übertragen, erst von da kann er sie nachladen.



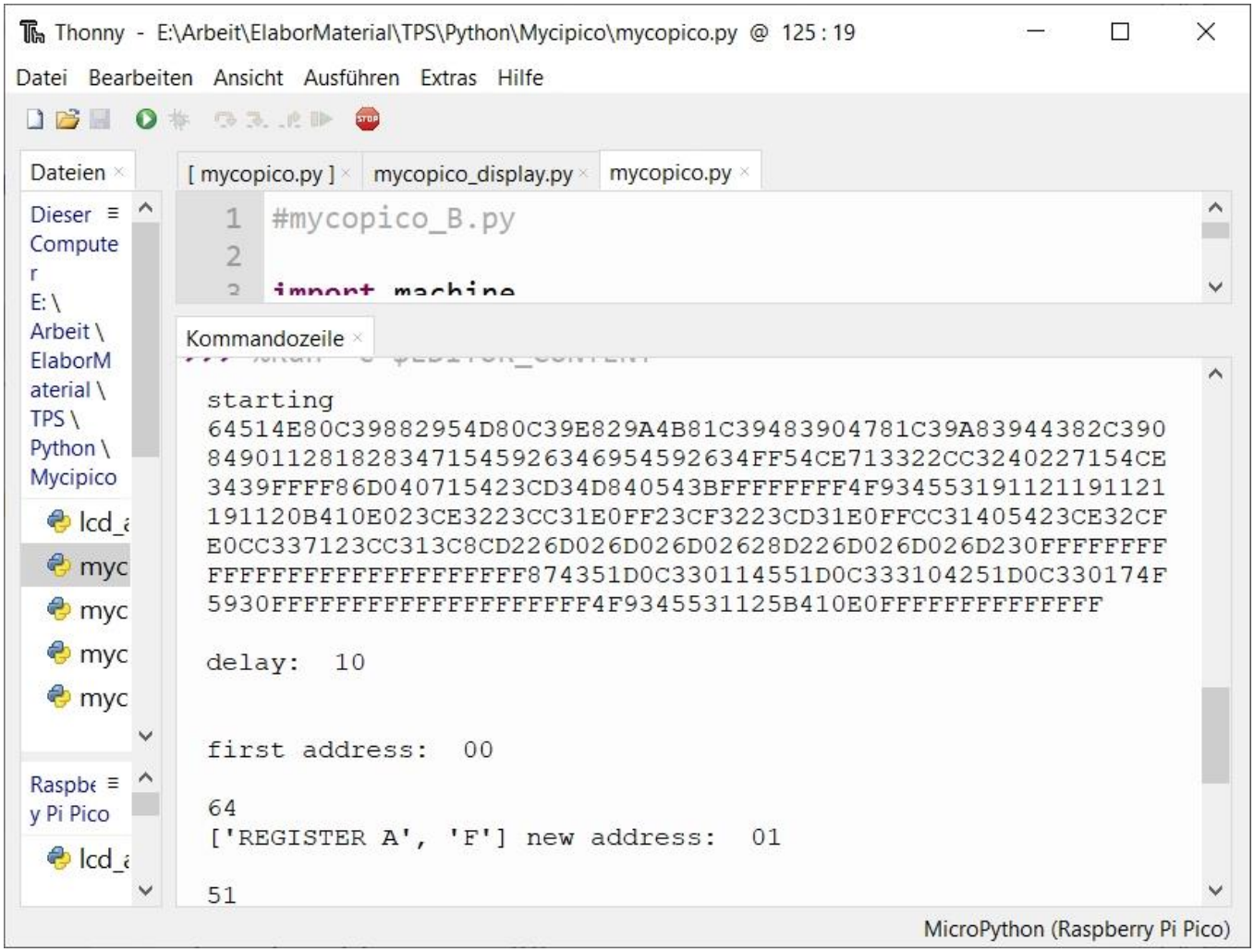
The second difficulty was that Robert used an I2C LCD but I didn't have one.
In mycopico.display.py I was able to switch it off at a central point: lcd_on = 0

Die zweite Schwierigkeit bestand darin, dass Robert ein I2C-LCD verwendet hat, ich aber kein solches hatte. In mycopico.display.py konnte ich es aber an einer zentralen Stelle abschalten: lcd_on=0



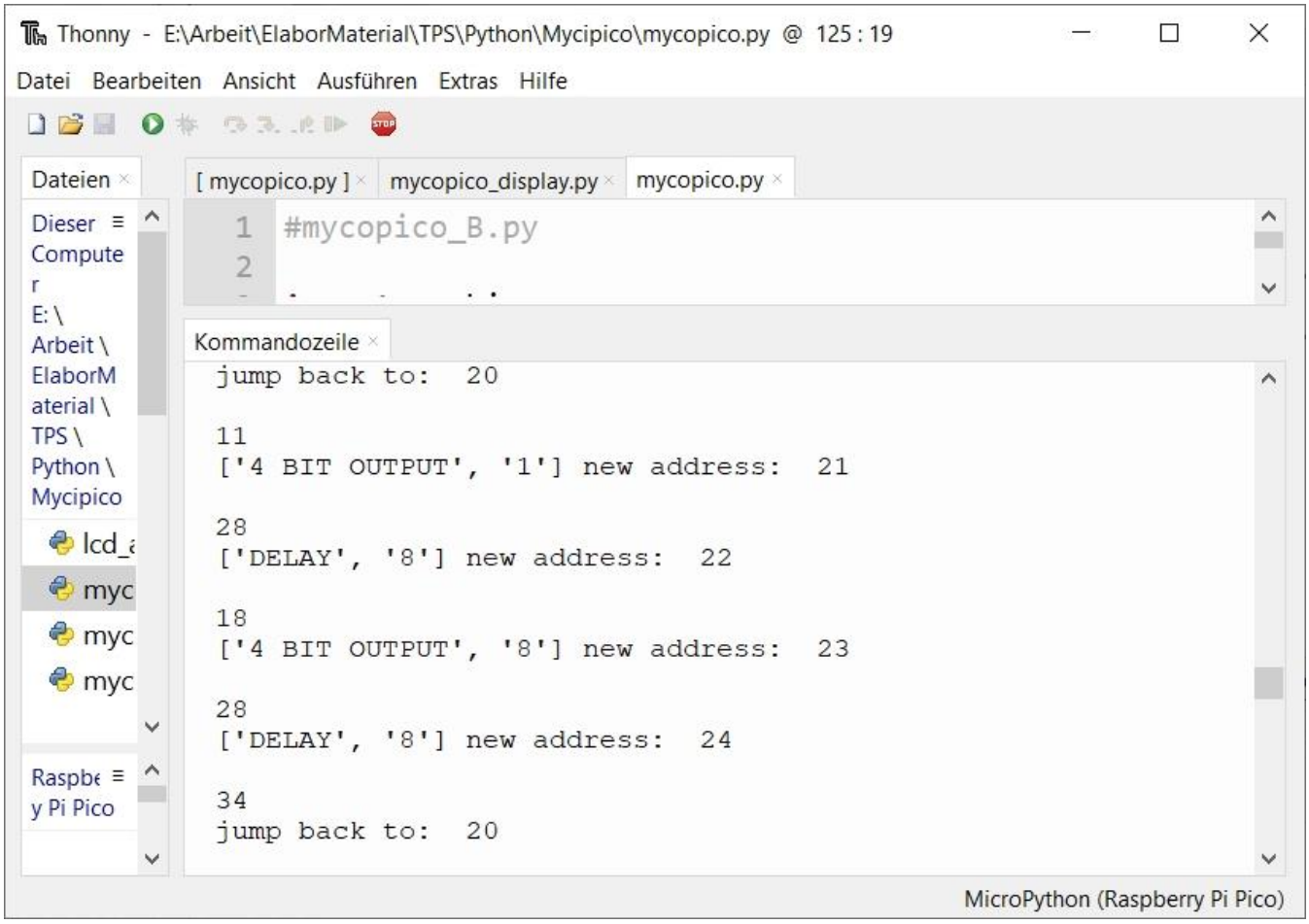
Now the software is running!
First, the entire collection of sample programs and sub-programs is loaded from the original TPS and then executed.

Jetzt läuft die Software! Zuerst wird die gesamte Sammlung der Beispielprogramme und Unterprogramme aus der originalen TPS geladen und dann ausgeführt.



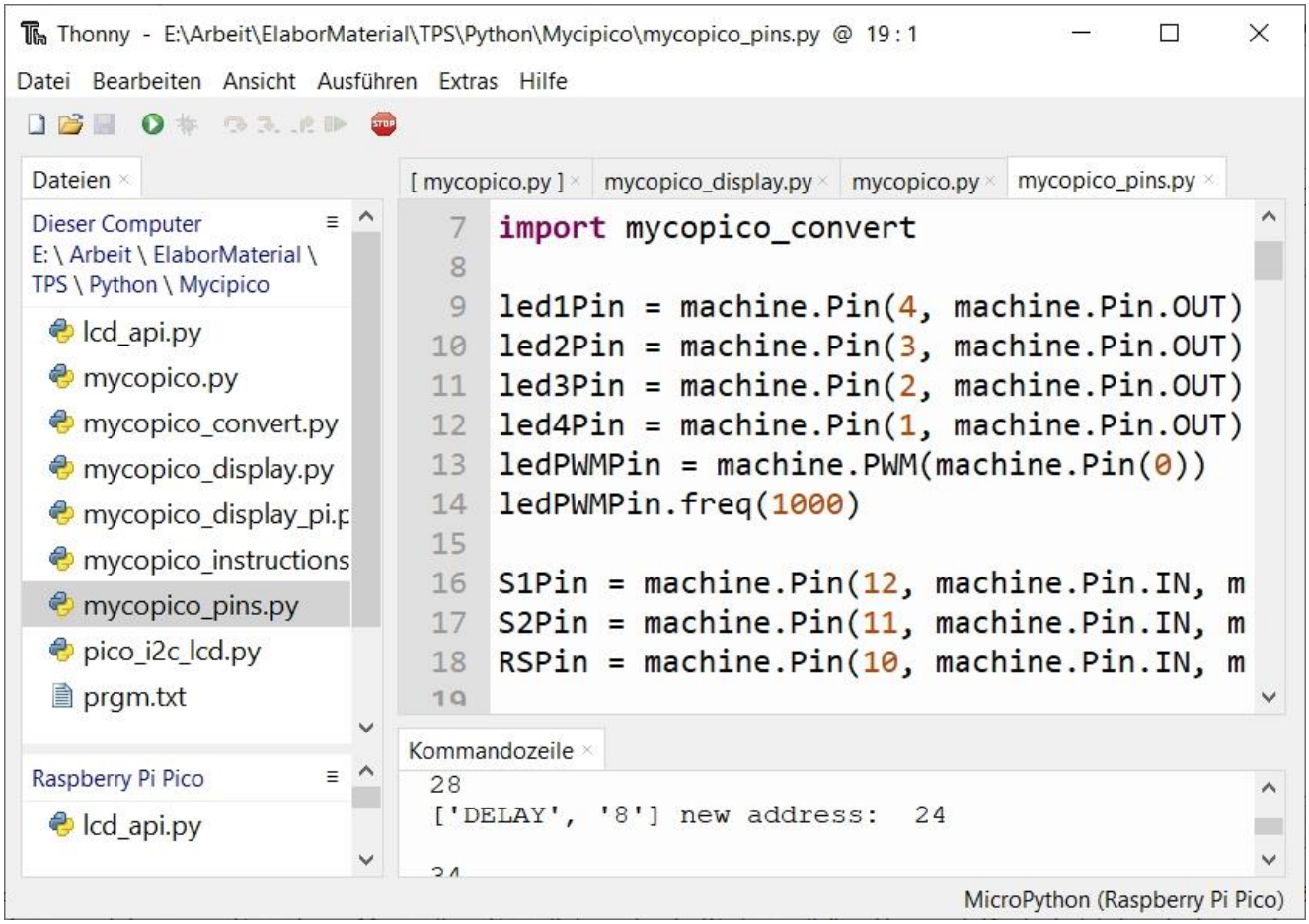
And as in the original TPS, the first example program now runs, an alternating flasher with outputs "1" and "8".
But unlike in the original TPS, you now get output for every line and you can watch how the program starts over and over again after returning to address 20.

Und wie in der originalen TPS läuft nun das erste Beispielprogramm ab, ein Wechselblinker mit den Ausgängen "1" und "8". Aber anders als in der originalen TPS erhält man nun Ausgaben zu jeder Zeile und kann so beobachten, wie das Programm nach einem Rücksprung auf die Adresse 20 immer wieder von vorn beginnt.



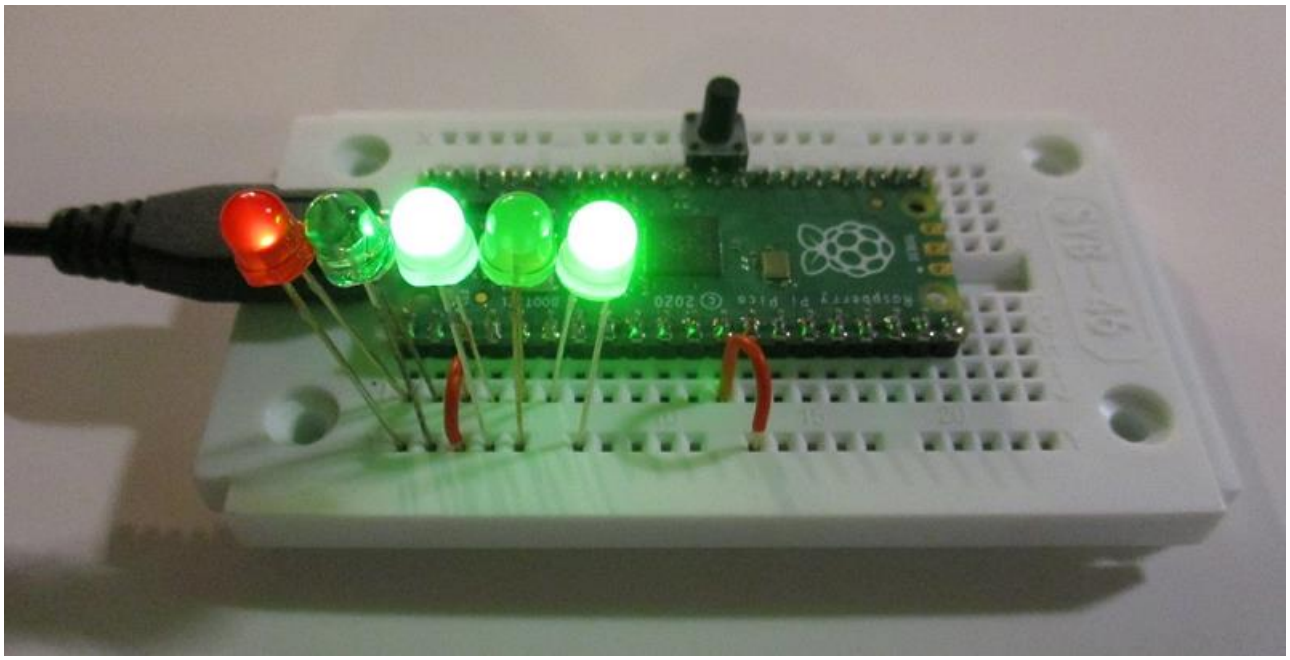
Now only the LEDs at the outputs are missing, then the Pico-TPS is almost complete. The mycopice_pins.py file provides an overview.
The outputs are at pins 1 to 4.
The LEDs with internal series resistors have now been connected.

Jetzt fehlen nur noch die LEDs an den Ausgängen, dann ist die Pico-TPS fast vollständig. Den Überblick verschafft die Datei mycopice_pins.py. Die Ausgänge liegen an den Pins 1 bis 4. Dort wurden nun die LEDs mit internen Vorwiderständen angeschlossen.



Sample program 2 is now running here, a counter with the 4 output LEDs, and the quasi-analog output to the PWM LED (red) in parallel.
The wire bridge on the right between input 1 and GND started it.
After RESET the Input status is read in, and 1110 is the indicattor to start example program 2.
Everything as usual and just like on the original TPS.

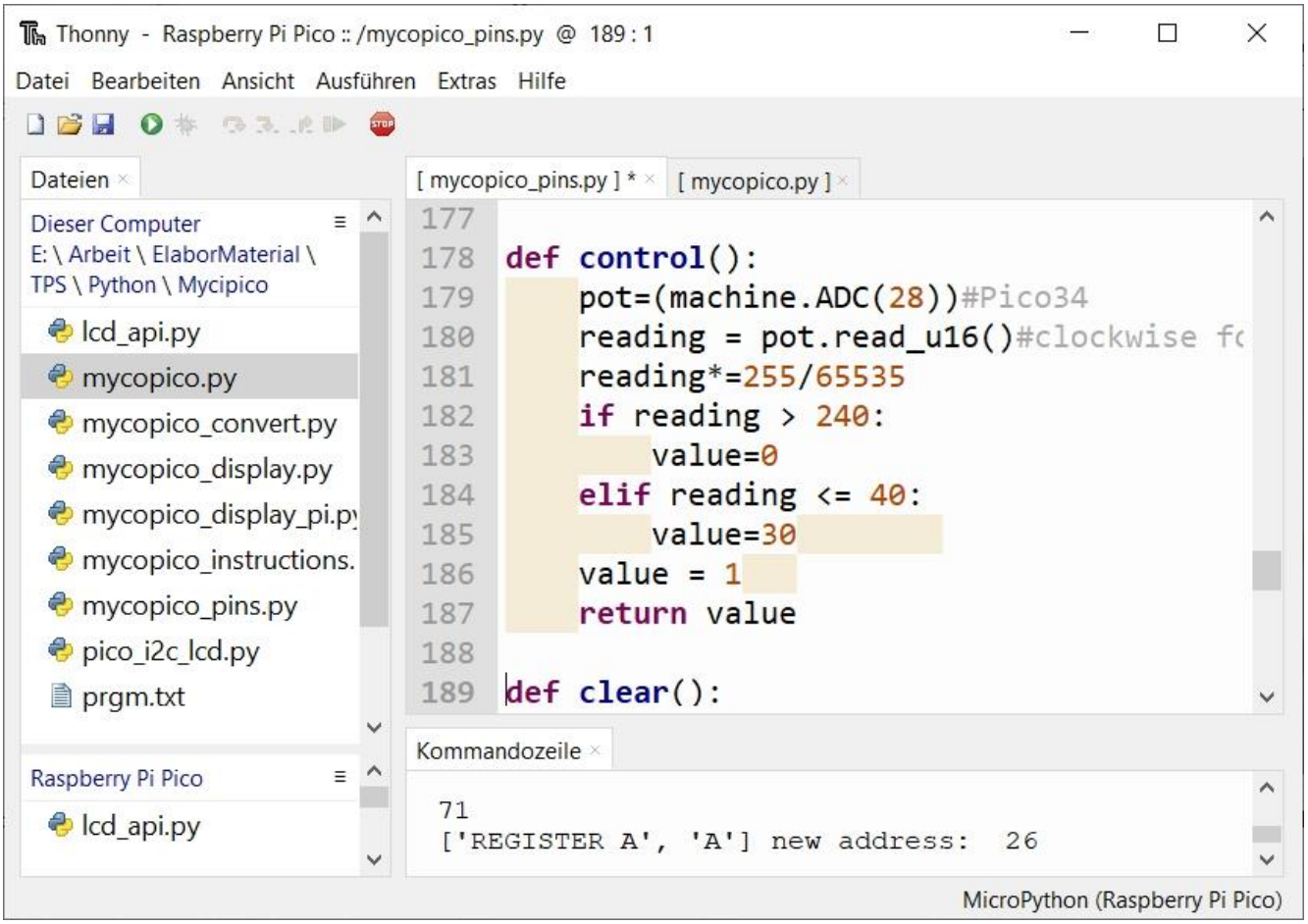
Hier läuft nun das Beispielprogramm 2, ein Aufwärtszähler mit PWM-Ausgabe (rot). Die rechte Drahtbrücke zwischen Eingang 1 und GND hat es gestartet. Alles wie gewohnt und genauso wie an der originalen TPS.



Another issue I had to overcome: The software uses a potentiometer with which you can set the

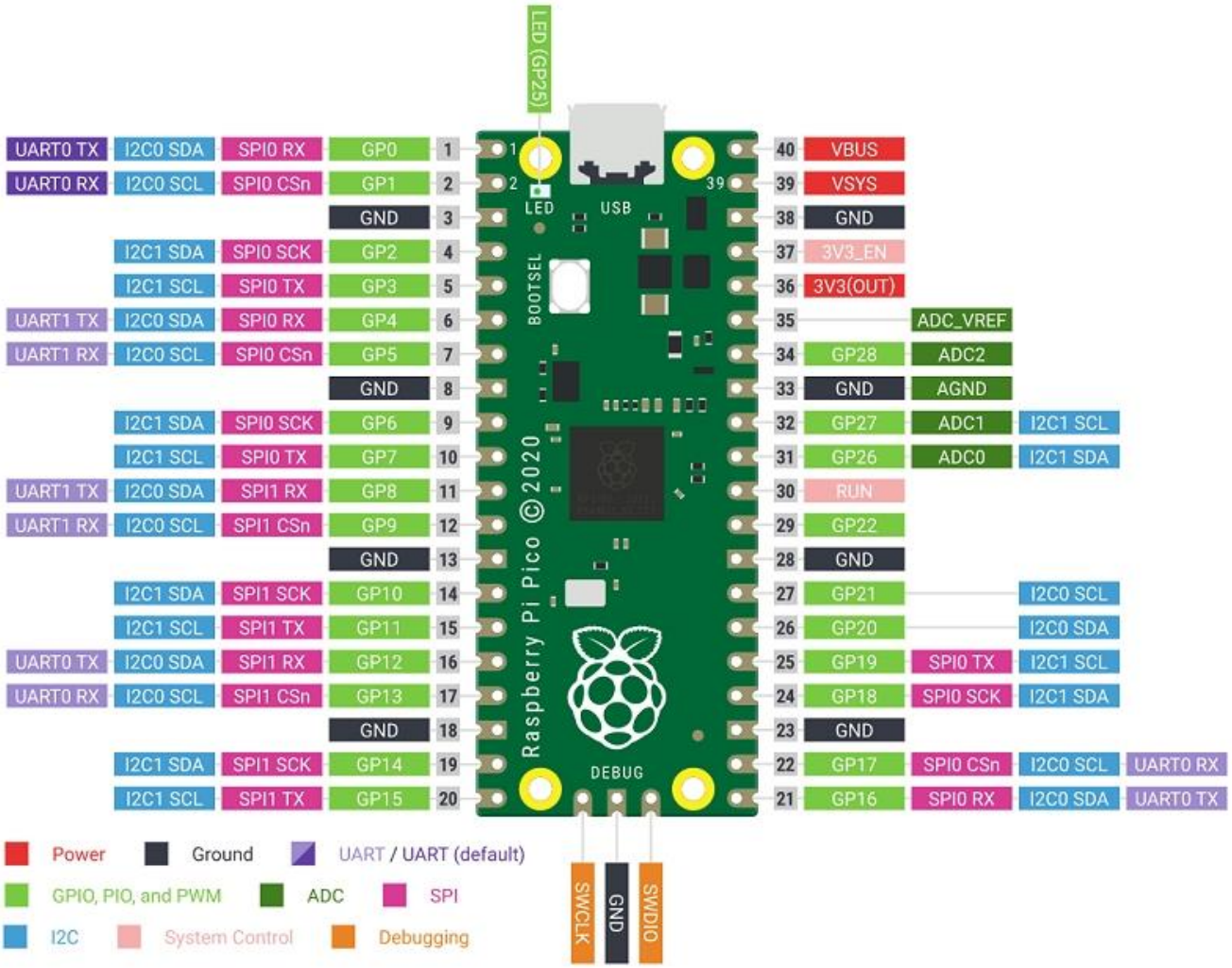
working speed, this potentiometer can be seen in the photo of Robert's board at the top.
Because I didn't have one connected, everything went too slowly.
Therefore, I have fixed the delay value value = 1 in the control () function,

Noch ein Problemchen: Die Software verwendet ein Poti, mit dem man die Arbeitsgeschwindigeit einstellen kann, zu erkennen auf dem Foto von Robert ganz oben. Weil ich keins angeschlossen hatte, lief alles zu langsam. Deshalb habe ich den Delay-Wert value = 1 in der Funktion control() fest vorgegeben,



So far everything is working well, and I've learned a lot about Python.
Much remains to be explored. For example, I should try to enter my own programs.
With the print output I can then follow exactly what each command is doing.

Soweit läuft es nun, und ich habe einiges über Python dazugelernt. Vieles ist noch zu erforschen. Zum Beispiel sollte ich mal probieren, eigene Programme einzugeben. Mit den Print-Ausgaben kann ich dann genau verfolgen, was jeder Befehl macht.


LED (GP25)
UART0 TX I2C0 SDA SPI0 RX GP0 1
UART0 RX I2C0 SCL SPI0 CSn GP1 2
GND 3
I2C1 SDA SPI0 SCK GP2 4
I2C1 SCL SPI0 TX GP3 5
UART1 TX I2C0 SDA SPI0 RX GP4 6
UART1 RX I2C0 SCL SPI0 CSn GP5 7
GND 8
I2C1 SDA SPI0 SCK GP6 9
I2C1 SCL SPI0 TX GP7 10
UART1 TX I2C0 SDA SPI1 RX GP8 11
UART1 RX I2C0 SCL SPI1 CSn GP9 12
GND 13
I2C1 SDA SPI1 SCK GP10 14
I2C1 SCL SPI1 TX GP11 15
UART0 TX I2C0 SDA SPI1 RX GP12 16
UART0 RX I2C0 SCL SPI1 CSn GP13 17
GND 18
I2C1 SDA SPI1 SCK GP14 19
I2C1 SCL SPI1 TX GP15 20
40 VBUS
39 VSYS
38 GND
37 3V3_EN
36 3V3(OUT)
35 ADC_VREF
34 GP28 ADC2
33 GND AGND
32 GP27 ADC1 I2C1 SCL
31 GP26 ADC0 I2C1 SDA
30 RUN
29 GP22
28 GND
27 GP21 I2C0 SCL
26 GP20 I2C0 SDA
25 GP19 SPI0 TX I2C1 SCL
24 GP18 SPI0 SCK I2C1 SDA
23 GND
22 GP17 SPI0 CSn I2C0 SCL UART0 RX
21 GP16 SPI0 RX I2C0 SDA UART0 TX
LED
USB
BOOTSEL
Raspberry Pi Pico ©2020
DEBUG
SWCLK
GND
SWDIO
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI
I2C
System Control
Debugging
