Einfacher SBM-20 Geiger-Müller Zähler auf dem Raspberry Pi Pico mit MicroPython

Bernd Laquai, 22.06 2025

Nachdem mit einem Impulszähler Test in /1/ die Zweifel ausgeräumt wurden, dass die interpretierte Programmiersprache MicroPython auf einem Pi Pico Mikrocontroller zu langsam laufen würde, um einen Geiger-Müller Zähler zu bauen, wurde nun mit einem SBM-20 Geiger-Müller Zählrohr-Modul von 4N-Galaxy ein einfacher Geiger-Müller Zähler aufgebaut. Dazu wurde wieder das B-P-XPLR Pico Explorer Board von Joy-IT verwendet auf dem der Pico aufgesteckt wurde und von dem nun auch das TFT-Display zur Anzeige verwendet wurde.

Abb. 1: SBM-20 Modul auf einem Tütchen Kaliumcarbonat und dem Raspberry Pi Pico auf einem Pico Explorer Board von Joy-IT als Geigerzähler

Das SBM-20 Modul von 4N-Galaxy stellt aus der 5V-Versorgung die Hochspannung für das SBM20-Zählrohr her (ca. 400V) und erzeugt die Zählpulse. Zudem hat es einen kleinen Lautsprecher um die Zählpulse hörbar zu machen. Die Zählimpulse werden invertiert von einem Spannungspegel von 3.3V aus erzeugt, d.h. es sind „active low Pulse“ von etwa 90µs Dauer. Der Code aus /1/ muss also beim Aufsetzen des Zählimpuls Interrupts entsprechend auf die fallende Flanke und einen Pull-Up-Strom abgeändert werden. Die erzeugten Zählimpulse bedingen eine Totzeit, die im Datenblatt mit 190us spezifiziert werden. Geht man von einem maximalen Duty-Cyle von 50% aus ergibt sich die maximale Zählrate von 2600cps. Das Modul wird vom Explorer Board mit 5V versorgt, dazu muss der entsprechende USB-C Eingang zusätzlich zum Pico mit einem 5V Netzteil verbunden werden. Der Impulsausgang des Moduls wird mit dem Pin GP10 verbunden, der dem Pico als Interrupt-Impulseingang dient.

Zur Programmierung wird wie in /1/ das Python Entwicklungstool Thonny (thonny.org) verwendet. Für die Anzeige des Messergebnisses auf dem TFT-Display muss zunächst die ST7735 Library sowie der Font für die Schrift auf den Pico transferiert werden. Beides findet sich in der Zip-Datei „RB-P-XPLR_Examples-and-libraries.zip“, die man auf der Joy-It Webseite /4/ bei Downloads herunterladen kann. Um die Dateien auf den Pico zu bekommen, aktiviert man in Thonny unter „View“ den

Menüpunkt „Files“. Dann erscheint links des Code-Fensters ein einfacher Dateimanager mit dem man in den ausgepackten Ordner des Zip-Files navigiert. Unter Libraries klickt man auf die Font-Datei „font5x7.fnt“ und kopiert sie auf den Pico und unter lib klickt man auf ST7735.py und kopiert diese Datei ebenfalls auf den Pico.

Den Code aus Listing 1 kopiert man ins Code-Fenster von Thonny und speichert ihn lokal auf dem PC. Optional kann man ihn auch unter main.py auf dem Pico speichern, so dass er auch autonom abläuft, sobald der Pico mit Strom versorgt wird. Wie bereits erwähnt verwendet der Code das Impulszähler Programm, welches in/1/ beschrieben ist. Lediglich wurde in der Interrupt Vereinbarung:

intrOne = Pin(intrIn, Pin.IN, Pin.PULL_UP)

ein Pull-Up Strom entgegen der Impuls-Richtung „low-aktiv“ des SBM-20 Moduls verwendet. Aus demselben Grund wurde die Anweisung zur Ausführung des Interrupt-Handlers ebenfalls abgeändert:

intrOne.irq(trigger=Pin.IRQ_FALLING, handler=countPulse)

Das heißt, die Ausführung wird nun auf die fallende Flanke getriggert.

Um auf das TFT-Display schreiben zu können, muss zu Beginn die LCD-Display Bibliothek ST7735 importiert werden. Nach der Initialisierung mit den Zeilen ab #Initialize LCD bis lcd.begin() wird das komplette Display mit der Anweisung lcd.fill_screen(lcd.rgb_to_565(200, 200, 200)) mit grauer Farbe gefüllt und die Überschrift mit lcd.p_string(0, 0, "SBM20 Geiger counter") beginnend ab der Koordinate 0, 0 auf das Display geschrieben.

Da man auf das Display einiges an Text unterbringen kann, wird nicht nur der aktuelle Messwert angezeigt, sondern auch die 4 vorangegangenen Werte um eine Änderung besser sichtbar zu machen. Die 5 Strings dafür werden in der Liste StrArry[] gespeichert, wovon StrArray[0] den aktuellen Messwert darstellt, der nach dem Scroll-Vorgang auf die y-Koordinate 10 geschrieben wird (x=20). Die 4 vorigen Messwerte werden nach dem Scrollen mit for i in range(numLines-1, 0, -1) und strArr[i] = strArr[i-1] auf die y-Koordinate mit y = i*10+10 und x = 20 geschrieben. Nach den Anweisungen zur Ausgabe auf das TFT-Display wird der Interrupt-Handler wieder scharf geschaltet, nachdem er unmittelbar nachdem Erreichen der erwarteten Anzahl an Zählimpulsen maxcnt deaktiviert wurde. Die Auswertung der Zählrate ist identisch wie in /1/. Sobald der Zähler an einem Referenzort kalibriert wurde und damit die Kalibrierfaktoren convOfst und convSlope bekannt sind, kann auch alternativ die Dosisleistung berechnet und ausgegeben werden. Dazu muss nur in strArr[0] = round(rate,2) die Variable rate durch die Variable dose ersetzt werden, die dann auf zwei Nachkommastellen gerundet ausgegeben wird.

```
from machine import Pin, SPI
import time
import ST7735
convOfst = 0  #in cps
convSlope = 1 #in (uSv/h)/cps
maxcnt = 100
intrIn = 10 #interrupt input
counter = 0
numLines = 5

# Initialize interrupt
intrOne = Pin(intrIn, Pin.IN, Pin.PULL_UP)

# Define interrupt handler functions
def countPulse(pin):
    global counter
    counter += 1
    #print(counter)
```

```
# Attach interrupt handler to interrupt input
# Initialize LCD
spi = SPI(0, baudrate=8000000, polarity=0, phase=0, sck=Pin(18), mosi=Pin(19),
miso=Pin(16))
lcd = ST7735.ST7735(spi, rst=6, ce=17, dc=3)
backlight = Pin(2, Pin.OUT)
# Turn backlight on
backlight.high()
lcd.reset()
lcd.begin()
lcd.fill_screen(lcd.rgb_to_565(200, 200, 200))
lcd.p_string(0, 0, "SBM20 Geiger counter")
strArr =[0.0 for i in  range (numLines)]
for i in range(numLines):
    lcd.p_string(20, i*10+10, str(strArr[i]))
#setup interrupt
intrOne.irq(trigger=Pin.IRQ_FALLING, handler=countPulse)
oldTime = time.ticks_ms()
while True:
    if (counter >= maxcnt):
        intrOne.irq(handler=None) #detach interrupt
        newTime = time.ticks_ms()
        dt = time.ticks_diff(newTime, oldTime) #measure time difference
        rate = float(maxcnt)*1000.0/float(dt) #in cps
        dose = (rate-convOfst)/convSlope; #in uSv/h
        oldTime = newTime;
        counter = 0;
        # Display content on the LCD
        # scroll
        for i in range(numLines-1, 0, -1):
            strArr[i] = strArr[i-1]
        strArr[0] = round(rate,2)
        for i in range(numLines):
            lcd.p_string(20, i*10+10, str(strArr[i]))
        intrOne.irq(trigger=Pin.IRQ_RISING, handler=countPulse)
```

Listing 1: MicroPythonCode für den SBM-20Geiger-Müller Zähler

Nach Fertigstellung der Programmierung mit MicroPython in Thonny wurde der Code in Gang gesetzt und zunächst die Zählrate zur Hintergrundstrahlung ohne Probe bestimmt (Abb. 2a). Man kann eine leichte Streuung erkennen, die auf Grund der 100 Zählimpulse auf die gewartet wird, bei etwa 10% liegen sollte, sofern man sehr viele Messungen macht. Bei nur 5 Messungen streut der Schätzwert für die Streuung (Standardabweichung) ebenfalls. Der Mittelwert der 5 angezeigten Messungen beträgt etwa 0.45cps.

Wenn nun beispielsweise einen Versuch für den Unterricht mit Schülern zum Thema Radioaktivität zu gemacht werden soll, der völlig ungefährlich ist, kann man sich das früher viel genutzte Backtriebmittel „Pottasche“ (Kaliumcarbonat) besorgen, das vorwiegend in der Weihnachtszeit in Reformhäusern und großen Lebensmittelmärkten in den Regalen für Backzutaten zu finden ist. Es enthält das auf natürliche Weise radioaktive Kalium auf das auch ein SBM-20 Zählrohr gut anspricht (vor allem auf die Betastrahlung). Kaliumcarbonat hat eine spezifische Radioaktivität von etwa 16Bq/g, das reine Kalium liegt bei 31Bq/g. Wird das Kalium vom Körper durch Lebensmittel aufgenommen regelt der körpereigene Stoffwechsel die Kaliummenge auf etwa 2g Kalium pro kg Körpergewicht, so dass die Kalium-bedingte Strahlung im Körper nie mehr als 4000-5000Bq an Aktivität ausmacht. Dieser Wert gilt als völlig unbedenklich und ist auch die Voraussetzung für eine gesunde Ernährung.

Der hier gezeigte Geigerzähler reagiert dann auch deutlich auf das Tütchen Pottasche, wenn man es direkt unter die Platine des Zählrohr-Moduls legt. Der Mittelwert aus den 5 Werten auf der Anzeige beträgt nun 1.15cps, daher liegt die Zählrate mit der Probe etwa um einen Faktor 2.5 höher, was signifikant ist.

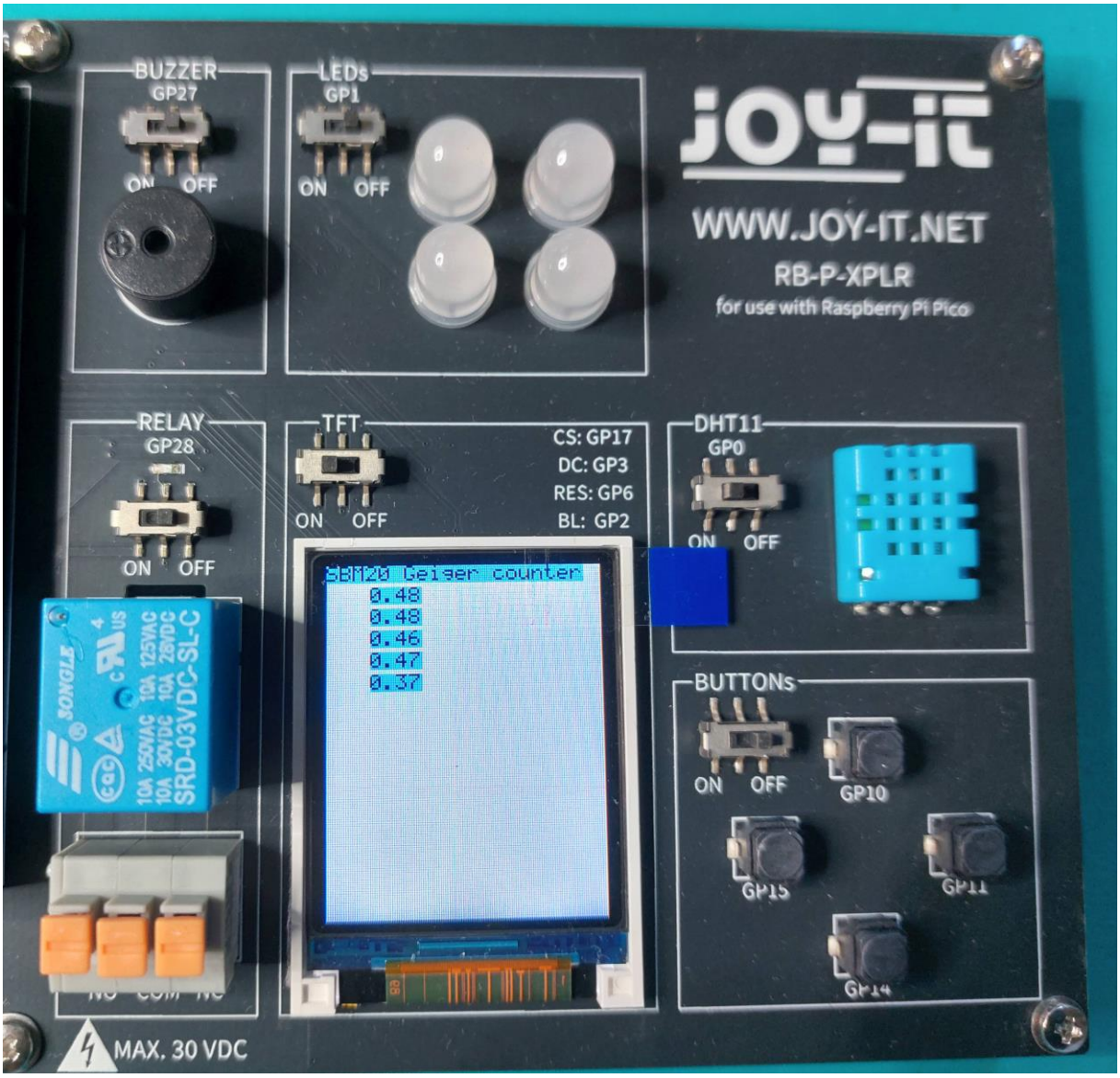

Abb. 2a: Zählraten zur Hintergrundstrahlung (ohne Probe)

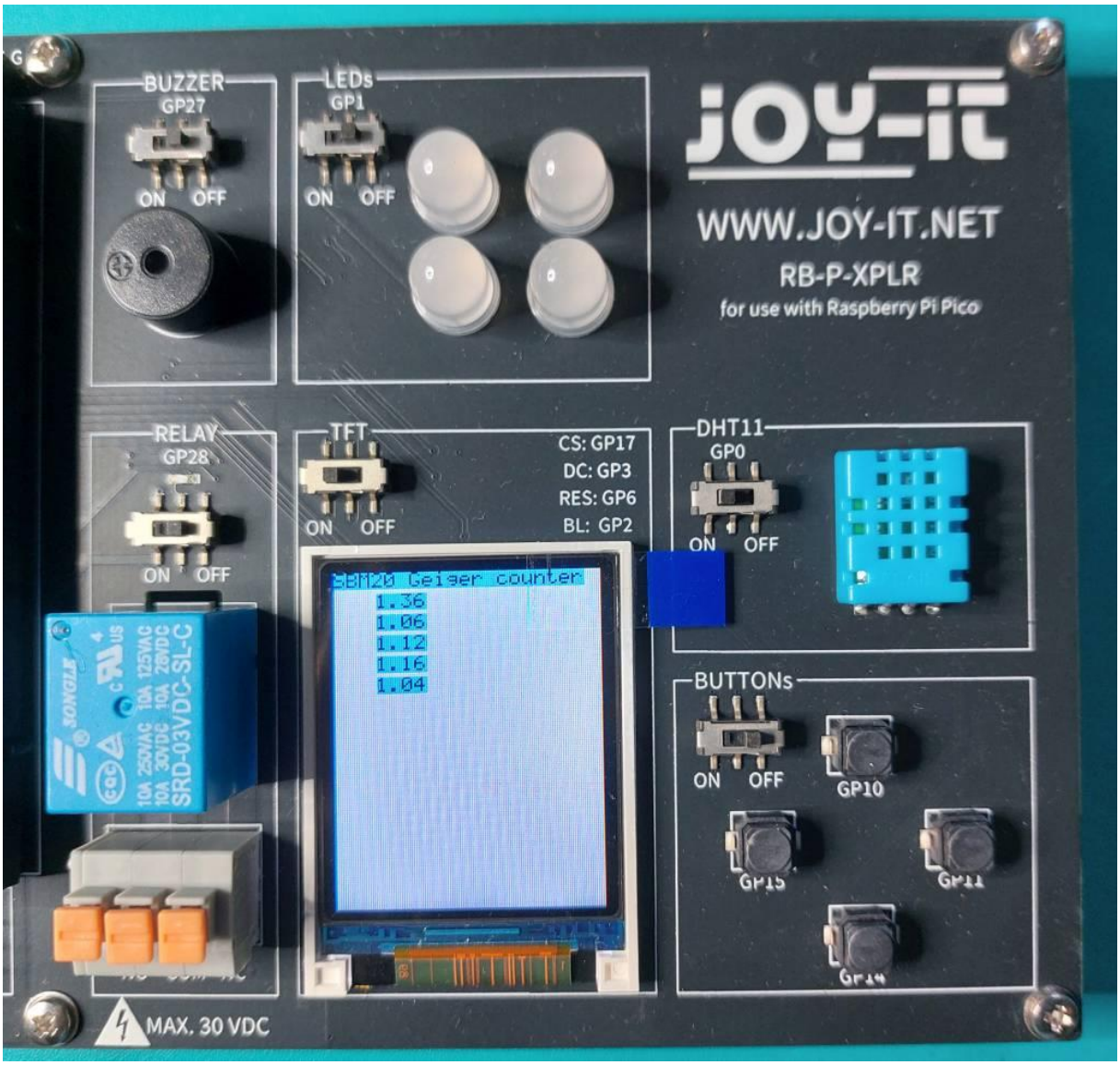

Abb. 2b: Zählraten mit einer Kaliumcarbonat Probe (Tütchen mit 30g Inhalt)

Literatur

/1/ Bernd Laquai; Impulszähler mit MicroPhython auf dem Raspberry Pi Pico
http://opengeiger.de/Impulsz%C3%A4hlerMicroPython.pdf

/2/ 4N Galaxy SBM-20 Driver-Modul
https://www.4n-gx.de/R02_de.html

/3/ Bernd Laquai; Das Zählrohr-Modul von 4N-Galaxy (SBM-Driver)
http://www.opengeiger.de/4NGalaxyModul.pdf

/4/ Joy-It Explorer Board
https://joy-it.net/de/products/RB-P-XPLR