Inhaltsverzeichnis

Telefon- Nummernschalter mit dem Raspberry Pi prüfen 3
Anforderungen 3
Installation 3
Die Ausgabe 5
Bekannte Probleme 5

Telefon- Nummernschalter mit dem Raspberry Pi prüfen

Falls man sich z.B. einen MFV-Konverter gebaut hat und der aber nicht so richtig funktionieren will, dann kann das schnell daran liegen, das der Nummernschalter nicht die Impulsfolgen produziert, die der IWV→MFV Konverter erwartet - so auch in meinem Fall, und auch wenn ich am Ende mit meinem Konverter gescheitert bin, so ist dabei immerhin ein Prüfprogramm abgefallen, was noch nicht so ganz rund läuft, aber wenigstens einen ersten Eindruck vermitteln kann, in wie weit der Nummernschalter sich überhaupt noch rührt.

Anforderungen

Das Programm ist für den Raspberry geschrieben, weil dieser mit seinen GPIOs die notwendigen Anschlüße für den Nummernschalter gleich frei Haus mitliefert.

Installation

Aktuell ist das Programm auf GPIO24 (Pin18) und den Masseanschluß gleich daneben (GND, Pin 20) konfiguriert. An diese beiden Pins schliesst man den NSI und NSA in Reihe an.

Weiterhin lädt man das folgende Programm auf den Raspi und startet es mit

```
python3 nummernschalterpruefer.py
```

nummernschalterpruefer.py

```
#!/usr/bin/python

# sudo apt-get install python3-rpi.gpio
from RPi import GPIO
from datetime import datetime
import time

ss = 0 # start time stamp
ls = 0 # last time stamp

# returns the elapsed milliseconds since the start of the program
def millis():
    dt = datetime.now()
    ms =  dt.minute * 60 *1000 + dt.second * 1000 + dt.microsecond /
1000
    return int(ms)

def mark(tick):
```

```
    print("\t", end="")
    for i in range(tick):
        if i > 100:
            break
        if i == 34:
            print (">", end="")
        elif i== 42:
            print ("<", end="")
        elif i== 56:
            print (">", end="")
        elif i== 68:
            print ("<", end="")
        elif i== 100:
            print ("%", end="")
        else:
            print (".", end="")
    print()




channel = 24
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP)


def main():
    global ss,ls,channel
    try:
        while True:
            success = GPIO.wait_for_edge(channel, GPIO.FALLING, timeout
= 50)
            if success is None:
                continue
            print("start")
            ss=0
            while not success is  None:
                ls = millis()
                if ss == 0:
                    ss= ls
                # wait for up to 5 seconds for a rising edge (timeout
is in milliseconds)
                success = GPIO.wait_for_edge(channel, GPIO.RISING,
timeout=1000)
                ts1 = millis()

                print ('H {} {} ' . format(ts1-ss, ts1-ls), end="")
                mark(ts1-ls)
                ls = ts1
                if not success is None:
                    success = GPIO.wait_for_edge(channel, GPIO.FALLING,
```

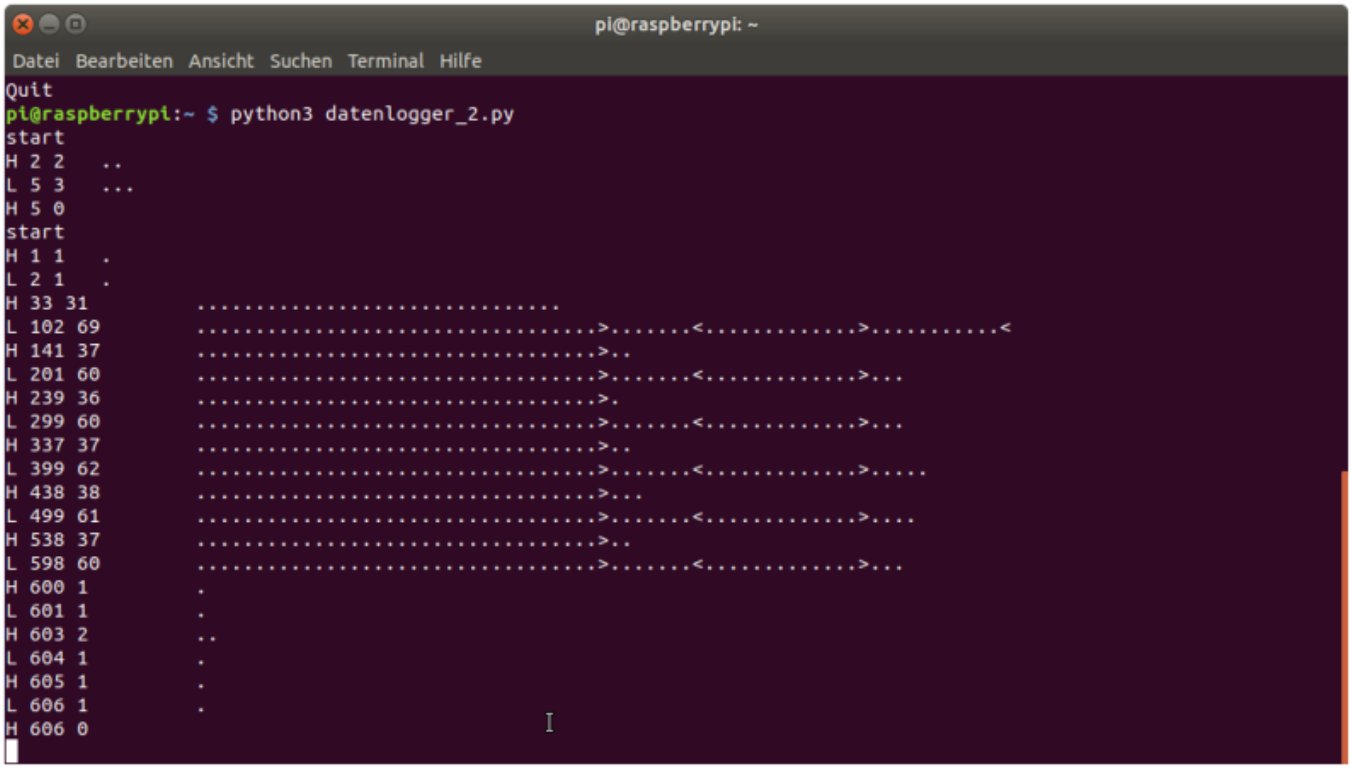
```
timeout=1000)
                    if not success is None:
                        ts2 = millis()
                        print ('L {} {} ' . format(ts2-ss, ts2-ls),
end="" )
                        mark(ts2-ls)

    except KeyboardInterrupt: # does not work if it runs in background.
        print ("\nQuit")

if __name__ == '__main__':
    main()
    GPIO.cleanup()
```

Die Ausgabe

Das Programm versucht die einzelnen Flanken zu erkennen und zeigt ihre Länge in Millisekunden an. Der besseren Optik wegen werden die Längen auch als Punkte ausgegeben, wobei dabei dann die standardisierten Impulslängen von 62ms und 38 ms +/- 10% durch < und > optisch angezeigt werden.



Bekannte Probleme

Die GPIO-Library ist nicht besonders gut dafür geeignet, Pin- Änderungen mit Timestamps zu koppeln, wenn Schalterprellen mit im Spiel ist. Wartet man nämlich auf eine beliebige Flanke, dann sagt einem

die Funktion nicht, welche Art Flanke nun getriggert hat. Die muß man dann erst separat abfragen, aber die kann sich natürlich durchs Schalterprellen in der Zwischenzeit schon wieder geändert haben. Wartet man jedoch auf bestimmte Flanken, dann hängt man auf einmal voll im Schalterprellen, denn wenn man den zusätzliche Bounce-Parameter benutzt, dann ist man mit der Entprellzeit schon schnell im Toleranzbereich des Nummernschalter-Signals selber, und dementsprechend verschlechtert sich dann auch die Genauigkeit der Anzeige und damit die Aussagekraft der ganzen Messung. Darum ist das Programm in seiner jetzigen Form nur ein Hilfsmittel, aber noch kein echt verläßliches Messinstrument.