Hardware

Raspberry	32€
3.5“ Display	29€
SD Karte 32GB	15€
SDR Stick NooElec NESDR Nano 3 OTG Bundle	33€
Mikrotaster	???
gesamt	**109€**
Zubehör	
Netzgerät	9€
Power Bank	ab ca. 20€

Software

Basis-Installation

Raspberry Sonden-Image von
http://www.qsl.net/oe1ffs/Sondenpage/Sondenraspi/Sondenraspi.html

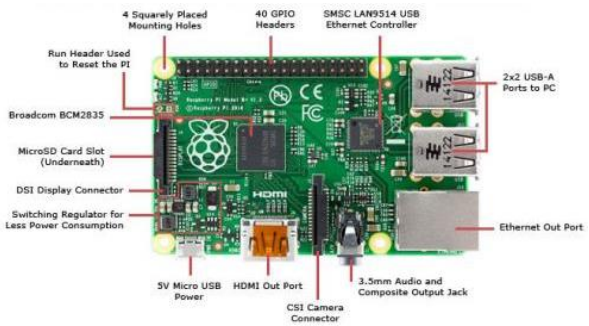
Erweiterungen

Bedienung über I/O Pins

Damit der Raspberry „im Feld“ ohne Maus und Tastatur bedient werden kann, werden drei Taster auf GPIOs angeschlossen und damit folgende Funktionen gesteuert (Bezeichnungen wie im Python Script):

1. Button01: Starten der Sonden-Decodierung bzw. mit jedem Tastendruck Weiterschalten des Frequenzbereichs um 2 MHz. Damit kann der gesamte relevante Frequenzbereich nacheinander mit nur einem SDR Stick durchgeschaltet werden.
2. Button02: Reserve, z.B. Starten/Beenden von APRSmap
3. Button03: Shutdown des Raspberry zum kontrollierten Herunterfahren statt einfachem Ausstecken des Akkupacks

Die drei Taster (Typ „normally open“) werden zwischen Pin 34 (Ground) und Pins 36 (GPIO16), 38 (GPIO20), 40(GPIO21) angeschlossen. Die GPIO Pins haben interne pull-up Widerstände, daher ist keine weitere Beschaltung notwendig.



Die Taster werden mit dem Python-Script `/home/pi/dxlAPRS/io_control.py` abgefragt:

```
import RPi.GPIO as GPIO
import subprocess
import time
import os

GPIO.setmode(GPIO.BCM)
GPIO.setup(21, GPIO.IN, pull_up_down=GPIO.PUD_UP) # button #1
GPIO.setup(20, GPIO.IN, pull_up_down=GPIO.PUD_UP) # button #2
GPIO.setup(16, GPIO.IN, pull_up_down=GPIO.PUD_UP) # button #3

iStarted = 0
# status button 01: 0 = nothing started, 1=sondecfg0_0, 2=sondecfg0_1
iStatus_Button01 = 0

# status button 02: 0 = nothing started, 1=aprsmap started
iStatus_Button02 = 0

while True:
    Button01 = GPIO.input(21)
    Button02 = GPIO.input(20)
    Button03 = GPIO.input(16)

    if Button01 == False:
        if iStatus_Button01 == 0:
            print("starting 401.0 - 403.0 MHz")
            subprocess.call("/home/pi/dxlAPRS/aprs/strsonde_wo_srv_0.sh")
            time.sleep(1)
            iStatus_Button01 = 1
        elif iStatus_Button01 == 1:
            print("starting 403.0 - 405.0 MHz")
            subprocess.call("/home/pi/dxlAPRS/aprs/strsonde_wo_srv_1.sh")
            time.sleep(1)
            iStatus_Button01 = 2
        else:
            print("stopping decoding")
            subprocess.call("/home/pi/dxlAPRS/aprs/endsonde.sh")
            time.sleep(1)
            iStatus_Button01 = 0

    if Button02 == False:
        if iStatus_Button02 == 0:
            print("Reserve")
            # start application

            time.sleep(1)
            iStatus_Button02 = 1
        else:
            print("Reserve")
            # stop application

            time.sleep(1)
            iStatus_Button02 = 0

    # Button03 is to shutdown the raspberry
    if Button03 == False:
        print("shutting down...")
        subprocess.call("/home/pi/dxlAPRS/shutdown.sh")
        time.sleep(1)
        iStarted = 0
```

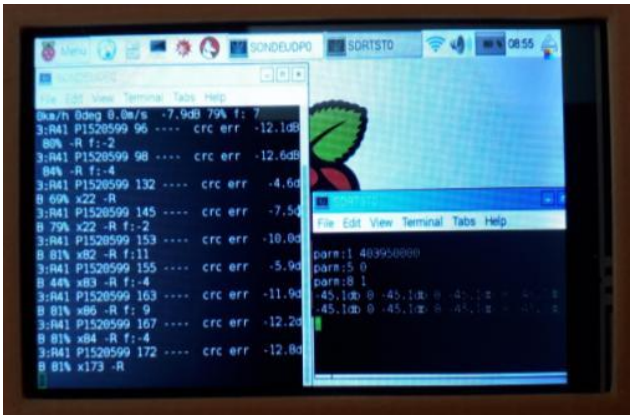
Damit dieses Script nach dem Hochfahren automatisch gestartet wird, muss folgende Zeile in /etc/rc.local hinzugefügt werden:

```
sudo python /home/pi/dxlAPRS/io_control.py
```

Folgende weitere Files müssen angelegt werden:

`/home/pi/dxlAPRS/aprs/strsonde_wo_srv_0.sh`	Tool-Konfiguration zum Starten der Sonden-Decodierung mit strcfg0_0.txt Kopie vom Original-File, aber im Bereich „SDRTST0“ ist sdrcfg0.txt durch sdrcfg0_0.txt“ ersetzt. <CALL-12>, <CALL-SSID> und <APRS-ID> sind entsprechend der Anleitung von OE1FFS anzupassen.
`/home/pi/dxlAPRS/aprs/strsonde_wo_srv_1.sh`	Tool-Konfiguration zum Starten der Sonden-Decodierung mit strcfg0_1.txt Kopie vom Original-File, aber im Bereich „SDRTST0“ ist sdrcfg0.txt durch sdrcfg0_1.txt“ ersetzt. <CALL-12>, <CALL-SSID> und <APRS-ID> sind entsprechend der Anleitung von OE1FFS anzupassen.
`/home/pi/dxlAPRS/aprs/sdrcfg0_0.txt`	Kopie vom Original-File strcfg0.txt mit Konfiguration der Sonden für den ersten 2MHz Bereich. Die Bereiche können beliebig definiert werden, z.B. auch zuerst 404.0MHz-406.0MHz und im sdrcfg0_1.txt ein Bereich darunter. So können die Frequenzen der Sonden in der Nähe bereits beim Starten, und erst beim Weiterschalten die Frequenzen der anderen Sonden ausgewählt werden.
`/home/pi/dxlAPRS/aprs/sdrcfg0_1.txt`	Kopie vom Original-File strcfg0.txt mit Konfiguration der Sonden für den nächsten 2MHz Bereich
`/home/pi/dxlAPRS/shutdown.sh`	Script zum Herunterfahren des Raspberry

Die Fenstergrößen und Positionen sind in `strsonde_wo_srv_<x>.sh` bereits so eingestellt, dass alle relevanten Informationen (Pegel, Sonden-Koordinaten) gleichzeitig am 3.5“ Display sichtbar sind.



Display Konfiguration

Damit die konfigurierten Fenstergrößen auch richtig angezeigt werden, muss in `/boot/config.txt` folgende Einstellung hinzugefügt werden (bzw. das File mit dieser Information angelegt werden):

```
# 3.5 inch display settings
max_usb_current=1
hdmi_group=2
hdmi_mode=87
hdmi_cvt 800 480 60 6 0 0 0
```

Gehäuse

Das Gehäuse besteht aus vier Teilen, die 3D gedruckt wurden:

1. Unterteil: hier kommt der Raspberry mit Display hinein. An der Oberseite (beim Scharnier) ist genug Platz, um auch noch den NooElec Nano 3 SDR Stick, ein paar SMA Adapter und die beiliegende Behelfsantenne unterzubringen.
   Druckzeit: ca. 3h
2. Deckel: kann mittels Scharnier mit dem Unterteil verbunden werden. Als Achse passt ein Draht mit 1.5mm^2 Durchmesser. Der Deckel hat vorne einen Schnapper, der in eine Nase am Unterteil einrastet und damit beim Transport zu bleibt.
   Druckzeit: ca. 1.5h
3. Tastenpaneel: Hier passen 3 Mikro-Taster hinein. Die Taster habe ich mit Superkleber zusammengeklebt und in die Ausnehmung gesteckt. Die Drähte zu den GPIOs sind an die Taster direkt angelötet und mit Stiftleistensteckern an die GPIO Leiste des Raspberry angesteckt
   Druckzeit: ca. 15min
4. Rahmen für Display: Hält das Display in seiner Position.
   Druckzeit: ca. 20min

In etwa 5h können also alle Teile gedruckt werden, dazu wird ca. 25m Filament verbraucht (Materialpreis: ca. 1,5€). Die Teile sind so konstruiert, dass sie eng zusammenpassen und müssen beim Zusammenbau mit ein paar vorsichtigen Feil-Strichen an den Ecken (dort drückt sich der Kunststoff beim Drucken etwas auseinander) angepasst werden. Dafür kann das ganze Gehäuse nur durch Zusammenstecken und ohne einer einzigen Schraube zusammengebaut werden. Der Zusammenbau dauert ca. 1h.