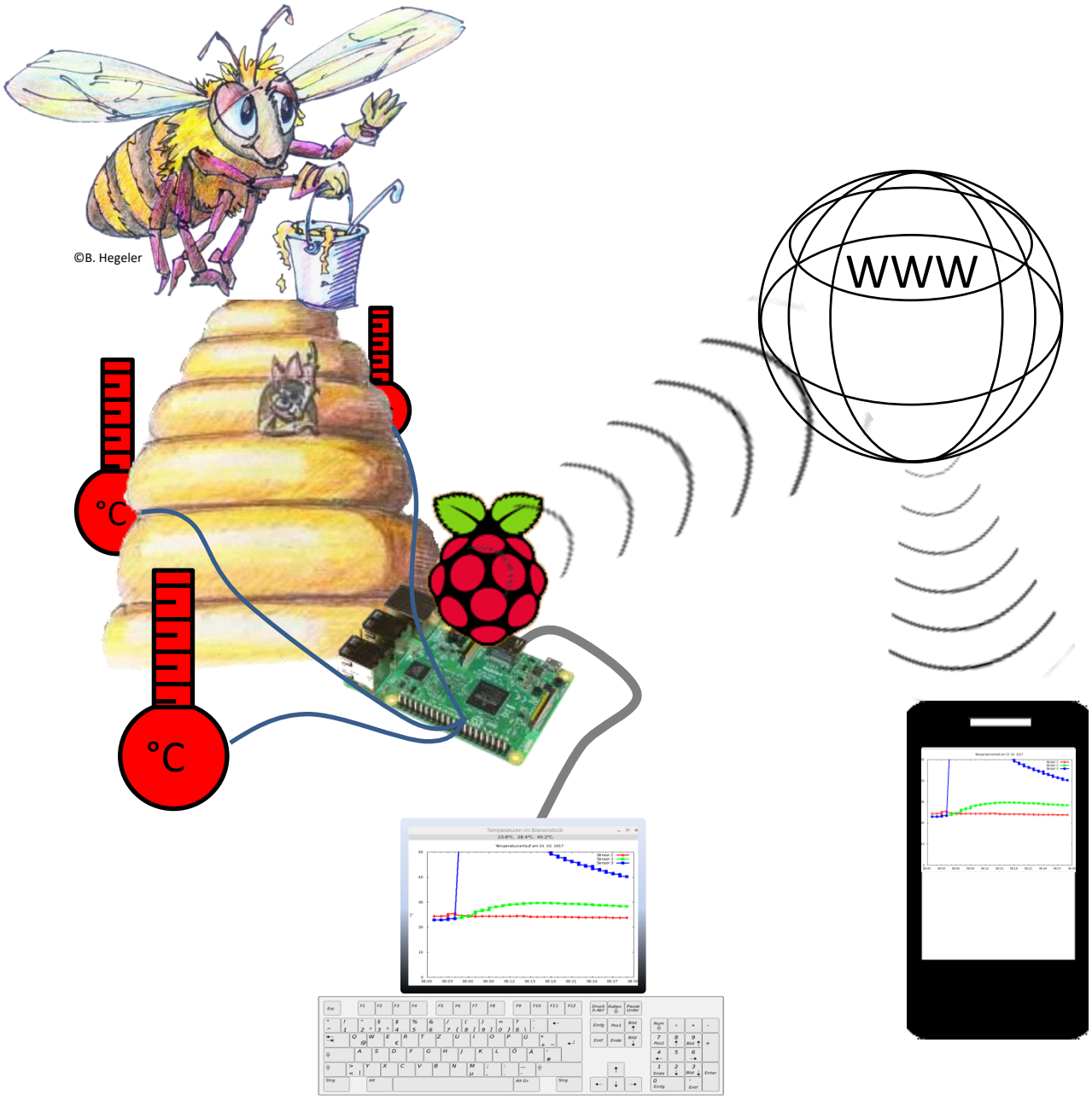
Temperaturmessung mit Raspberry Pi und DS1820-Sensoren

incl. Datenspeicherung, grafischer Darstellung und Online-Stellung für Erkundungen in Schulgarten, Freiland und Labor



igs lugA

Integrierte Gesamtschule Ludwigshafen Gartenstadt

Inhalt

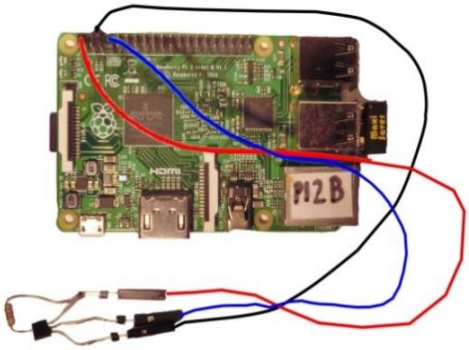
Dieses Skript ist das Zwischenergebnis eines Explore-Science-Projektes an der IGS Ludwigshafen Gartenstadt und es konnte aus Zeitgründen nicht endgültig durchgestaltet und korrigiert werden.
Dass es beim Zusammenspiel der Geräte zu Problemen kommen kann, ist im gegenwärtigen Stadium nicht zu vermeiden – und es hat den Vorteil, dass die technischen Probleme, die bei solchen Projekten auftauchen, erlebbar werden, ebenso wie die oft mühseligen und zeitaufwändigen Lösungsversuche. Die Lektüre des Skripts ohne ein entsprechendes gerätemäßiges Umfeld ist nur bedingt sinnvoll.

Ludwigshafen-Gartenstadt, im Juni 2017

Dr. Gerd Hegeler-Burkhart

I. Messtechnik mit einem Raspberry-Pi verwirklichen: Einige Anmerkungen vorweg

Rechnergestützte Mess- und Steueraufgaben im schulischen Bereich lassen sich zurzeit besonders gut mit Ardunio- und Raspberry-Pi-Systemen verwirklichen. Beide sind mit überschaubaren Kosten verbunden und sehr gut im Internet dokumentiert, man findet dort in den Foren etc. vielen nützliche Hinweise. Wesentlicher Unterschied: Arduino ist ein typisches Mikrocontroller-System, es lässt sich besonders kompakt für Aufgaben nutzen, die – einmal aufgebaut – nicht mehr verändert werden müssen. Der Raspberry Pi stellt einen kompletten Computer samt universellem Betriebssystem zur Verfügung, ist komplizierter und vielseitiger einsetzbar. Die unten dargestellten Szenarien sollten aus sich heraus deutlich machen, warum wir uns in diesen Fällen für den Raspberry Pi entschieden haben. Das sollte auch in dem Video https://youtu.be/yWQg4NyDO5g deutlich werden, passend dazu der QR-Code.



Ein Raspberry-Pi-Computer, im Folgenden oft abgekürzt als „Raspi“ bezeichnet, ist ein kompakter Einplatinencomputer. Damit er ähnlich wie ein Windows- oder Apple-Rechner benutzt werden kann, muss er mit einer Peripherie ausgestattet werden, d.h. mit Bildschirm, Tastatur, Maus etc. Es besteht auch die Möglichkeit, einen Raspi über einen anderen Computer, ein Tablet oder ein Smartphone fernzusteuern aus. Dann kommt man mit wenig Hardware aus. Details dazu später.
Raspberry-Pi-Computer gibt es mittlerweile in der 3. Generation (die nur wenig von der 2. Generation abweicht). Wir werden beide Bauarten verwenden. Das Betriebssystem, das wir nutzen – also die Entsprechung zu Windows oder iOS – ist „Raspian“ in der Variante „PIXEL“. Raspian ist eine speziell angepasste Version des Betriebssystems Linux der Variante Debian-Jessie, welches als Alternative zu Windows und iOS ganz legal benutzt werden darf, ohne dass man eine Lizenz dafür kaufen muss. Das gilt speziell auch für „PIXEL“.

Das ganze Betriebssystem samt aller Einstellungen auf einer SD-Karte
Ein Raspberry-Pi-Computer hat keine Festplatte, auf der das Betriebssystem installiert wird. Diese Aufgabe übernimmt eine Micro-SD-Karte, am besten geeignet sind 8-GB-Karte, die Lese/Schreibgeschwindigkeit ist für unsere Zwecke (keine umfangreichen Videosequenzen etc.) zweitrangig.

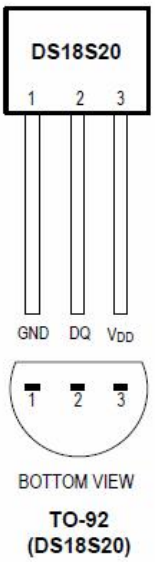
Neueinrichtung des Pixel-Betriebssystems
Das PIXEL-Betriebssystem wird mit einem andern Computer von einem Server der Raspberry-Foundation Organisation heruntergeladen, entpackt (Zip-Datei) und dann als sogenannte Image-Datei auf die SD-Karte geschrieben. Anschließend müssen dann Zeitzone, Tastaturlayout etc. auf deutsche Verhältnisse eingestellt werden und andere Einstellungen (WLAN usw.) vorgenommen werden. Außerdem müssen noch spezielle Module aus dem Internet nachgeladen werden, ebenso spezielle Software, die man nutzen will. Viel Standardsoftware (Browser, Textverarbeitung …) ist bereits in PIXEL integriert. Man muss sich informieren. Die in unserem Zusammenhang notwendige Entwicklungsumgebung für die Programmierspreche „Python“ ist auf jeden Fall schon installiert.

Kopie (Clon) einer SD-Karte eines bereits fertig eingerichteten Raspis anfertigen
Hat man einen Raspi eingerichtet, kann man eine Kopie der SD-Karte machen, zum einen, um einen Sicherheitskopie als Image-Datei zu erstellen, zum anderen, um die Einstellungen als „Clone“ auf einem anderen Raspi zu nutzen. Das ist ein enormer Vorteil, weil alle Anpassungen, Dateien, Installationen etc. übernommen werden.

Neues PIXEL **-Image oder Clone-Image mit „Win 32 Disk Imager“ schreiben und lesen**
Wichtig für beide Vorgehensweisen: Um Image-Dateien auf eine SD-Karte zu schreiben, muss man das kostenlose Programm „Win 32 Disk Imager“ (oder eine entsprechende Software) nutzen. Dieses Programm kann auch eingesetzt werden, um von einer fertigen „PIXEL“-SD eine Image als Sicherheitskopie bzw. Clone zu ziehen.

II. DS1820-Temperatursensor-Module

Pin Configurations



Pinbelegung eines DS18S20. Auch andere Typen dieser Sensorserie sind entsprechend aufgebaut.

GND, Pin 1: Masse (0 Volt),
DQ, Pin 2: Daten
VDD, Pin3: Betriebsspannung (beim Raspberry Pi 3,3 Volt)

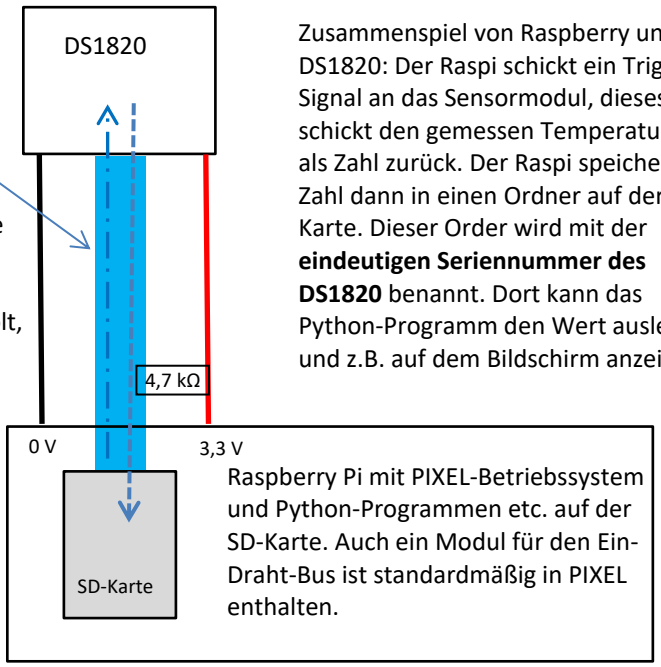
Quelle: Produktbeschreibung des Herstellers „maxim integrated“

Ein DS1820 in TO-92-Bauweise mit angeschlossenem Widerstand und als wasserdichte versiegelte Variante

Beim DS1820 handelt es sich um einen integrierten Schaltkreis in einem sogenannten TO-92-Gehäuse mit drei Beinchen – daher ist er von außen leicht mit einem Transistor zu verwechseln. Ein DS1820 enthält einen analogen Temperatursensor, einen Analog-Digitalwandler und ein1-Wire-Interface. Die drei Anschlüsse sind Masse (GND, Pin 1), Daten (DQ, Pin 2) und Betriebsspannung (VDD, Pin3). Mehrere dieser Temperatursensormodule, in Folgenden kurz „Sensoren“ genannt, können parallel angeschlossen werden. Wichtig ist, dass man einen 4,7kΩ-Widerstand zwischen die Spannungsversorgungs-Pin (3) und einen der Datenpins (2) montiert. Egal, ob man einen oder mehrere Sensoren an einen Rasperry Pi (Raspi), einen Arduino o.Ä. parallel anschließt: Ein Widerstand reicht!

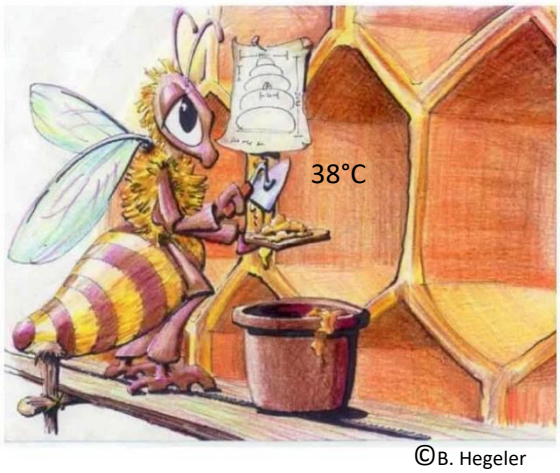
Kabel für den „1-Draht-Bus“

Die Kommunikation zwischen Raspberry Pi und DS1820 findet auf nur einem Draht statt, daher die Bezeichnung „Ein-Draht-Bus“ (One-Wire-Bus). Außerdem braucht der Sensor für seinen Betrieb noch eine Stromversorgung (Masseanschluss 0 Volt, Betriebsspannung 3,3 Volt). Darum sind insgesamt drei Kabel notwendig.



Zusammenspiel von Raspberry und DS1820: Der Raspi schickt ein Trigger-Signal an das Sensormodul, dieses schickt den gemessen Temperaturwert als Zahl zurück. Der Raspi speichert die Zahl dann in einen Ordner auf der SD-Karte. Dieser Order wird mit der **eindeutigen Seriennummer des DS1820** benannt. Dort kann das Python-Programm den Wert auslesen und z.B. auf dem Bildschirm anzeigen.

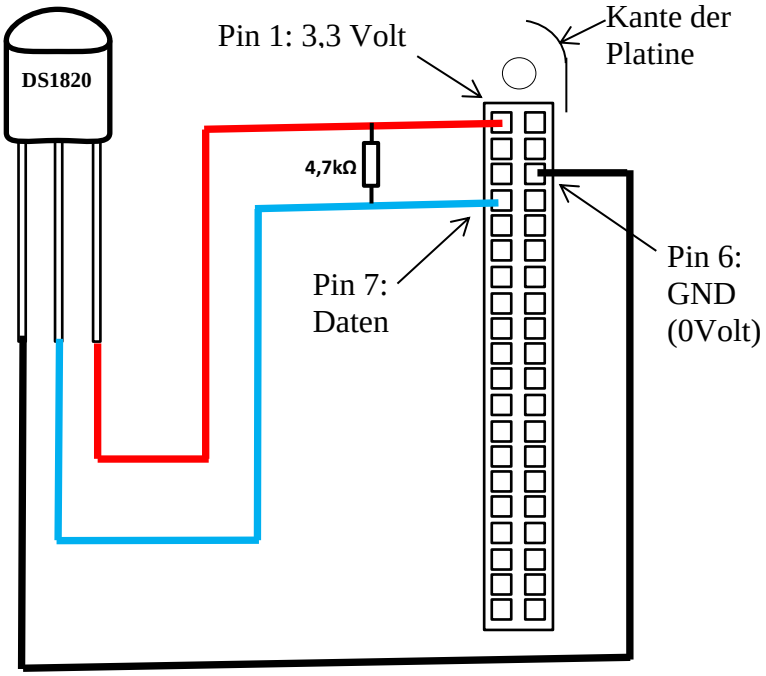
III. Dem Temperaturmanagement im Bienenstock auf der Spur

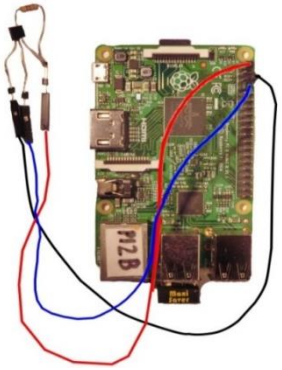


©B. Hegeler

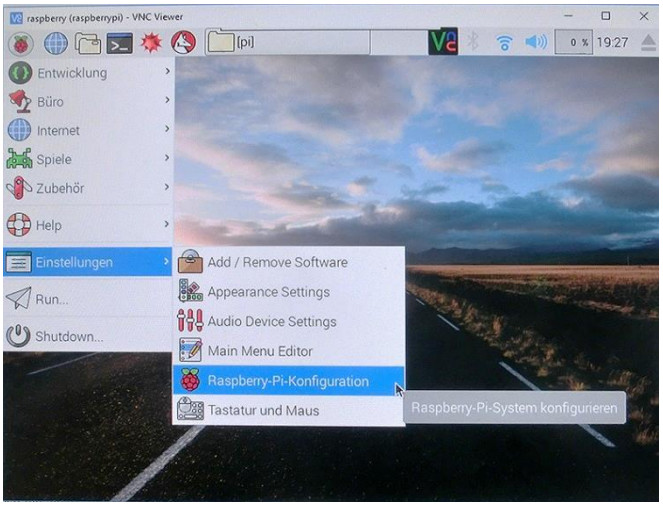
Die Garten-AG an der IGS Ludwigshafen Gartenstadt hat einen schönen Schulgarten und darin einige Bienenvölker, die von der Bienen-AG liebevoll gepflegt werden.
Hier ergab sich ein interessanter Ansatzpunkt für eine Zusammenarbeit mit dem Fachbereich Informatik: Bienenvölker beherrschen ein raffiniertes Temperatur-management. Die heranreifenden Puppen beispielsweise brauchen eine Temperatur von etwa 35° Celsius, die Wachswaben werden bei 38°C gebaut, die Honigvorräte werden kühler gelagert. In den kalten Monaten der Winterruhe dürfen die Temperaturen nie unter etwa 10° bis 5° abfallen, sonst stirbt das Volk, im Sommer sind etwa 50° tödlich.
Ein Bienenvolk löst alle diese Aufgaben selbstverständlich mit einem rein ökologischen Programm: Zur Erhöhung der Temperatur krabbeln die Arbeiterinnen dahin, wo die Wärme gebraucht wird, hängen ihre Flügel aus und lassen Ihre Flugmuskulatur auf Volllast laufen. Wird es zu warm, dann wird Wasser an die richtigen Stellen gebracht und durch emsiges Fächeln zum Verdunsten gebracht, damit es dabei das Zuviel an Wärmeenergie abtransportiert. (Quelle: Tautz, Die Erforschung der Bienenwelt, Audi Stiftung für Umwelt GmbH, Ingolstadt und Klett MINT GmbH, Stuttgart, 2014)
Extreme Temperaturen können das Bienenvolk an den Rand seiner Leistungsfähigkeit bringen, sei es durch Sonneneinstrahlung an sowieso schon extrem heißen Tage, sei es durch langanhaltende Kälteperioden. Unterstützende Maßnahmen, die Imkerinnen und Imker dann ergreifen können, sind beispielsweise zusätzlicher Sonnenschutz, bessere Wärmeisolierung des Gehäuses oder das Zufüttern von Zuckerwasser. Doch das ist an dieser Stelle nicht unser Thema. Wir kümmern uns im Messprojekt darum, wie wir angemessen Einblick erhalten in die Temperaturverteilung im Bienenstock. Klar ist, dass dazu eine ganze Reihe von Sensoren notwendig ist.

Der Temperatursensor DS18B20 und seine Anbindung an einen Raspberry Pi
Beim DS18B20 handelt es sich um einen integrierten Schaltkreis in einem sogenannten TO-92-Gehäuse mit drei Beinchen – daher ist er von außen leicht mit einem Transistor zu verwechseln. Der DS18B20 enthält einen analogen Temperatursensor, einen Analog-Digital-Wandler und ein1-Wire-Interface.
Die drei Anschlüsse sind Masse (GND, Pin 1), Daten (DQ, Pin 2) und Betriebsspannung (V_{DD}, Pin3).
Das Erfreuliche ist, dass mehrere dieser Temperatursensormodule, in Folgenden kurz „Sensoren“ genannt, parallel an einen Mikroprozessor oder einen Einplatinencomputer, z.B. einen Raspberry Pi (kurz „Raspi“), angeschlossen werden können. Das bedeutet, dass man sich beim Anschließen nur um drei Beinchen (Pins) am Raspi kümmern muss. Dabei ist wichtig, dass man die richtigen identifiziert und dass man einen 4,7kΩ-Widerstand zwischen Pin 1 und Pin 7 des Raspis montiert - egal, ob man eine oder bis zu acht Sensoren anschließt. Warum das so ist, wird in Foren im Netz beschrieben.



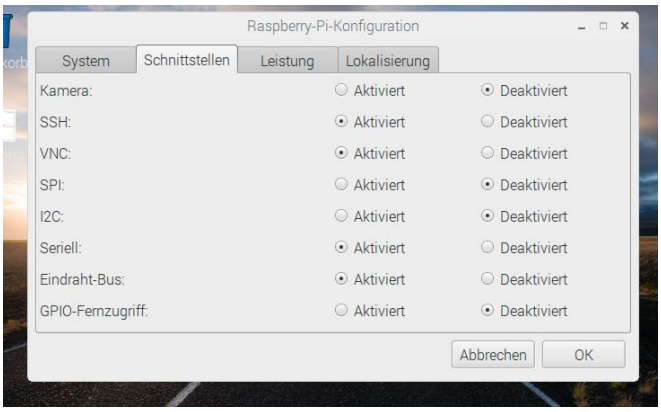


Oben: Anschlussschema für einen DS18B20 an einen Raspberry Pi
Rechts: Tatsächliche Schaltung (ohne Bildschirm, Tastatur etc.) mit dem Raspberry Pi



Nun einige grundsätzliche Anmerkungen dazu, wie ein Computer auf die Messwerte des DS1820-Sensors zugreift: Jeder Sensor hat eine einmalige Seriennummer. Dank dieser kann der Raspi für jeden Sensor eine eigene Datei anlegen, die über diese Seriennummer identifiziert wird und den aktuellen Messwert enthält.
Um den Raspi für den geschilderten Zweck zu betreiben, empfiehlt sich die Nutzung des Betriebssystems „PIXEL“ , das kostenlos im Netz verfügbar ist und auf eine Micro-SD-Karte übertragen werden muss.
Anleitungen zur Inbetriebnahme eines Raspis gibt es online.

Damit der Raspi, natürlich ausgestattet mit Tastatur und Bildschirm und mit richtiger Datums- und Zeiteinstellung, für unsere Zwecke genutzt werden kann, muss noch eine Einstellung vorgenommen werden, damit auf den Sensor bzw. die Sensoren zugegriffen werden kann, siehe dazu beide Bilder links.



Wenn die Freischaltung des Eindraht- bzw. 1-Wire-Buses erledigt ist, sollte das folgende Python-Programm „ds1820temp.py“ die richtigen Messwerte auslesen und anzeigen können. Es lässt sich über einen USB-Stick auf den Raspi kopieren. (Der Stick erscheint unter „media“.) Natürlich können die Werte auch in Dateien gespeichert und zu Grafiken umgewandelt werden. Das komplette Programm „beetemp.py“, das wir für unseren Bienenstock an der IGS Luga nutzen, befindet sich im Anhang.

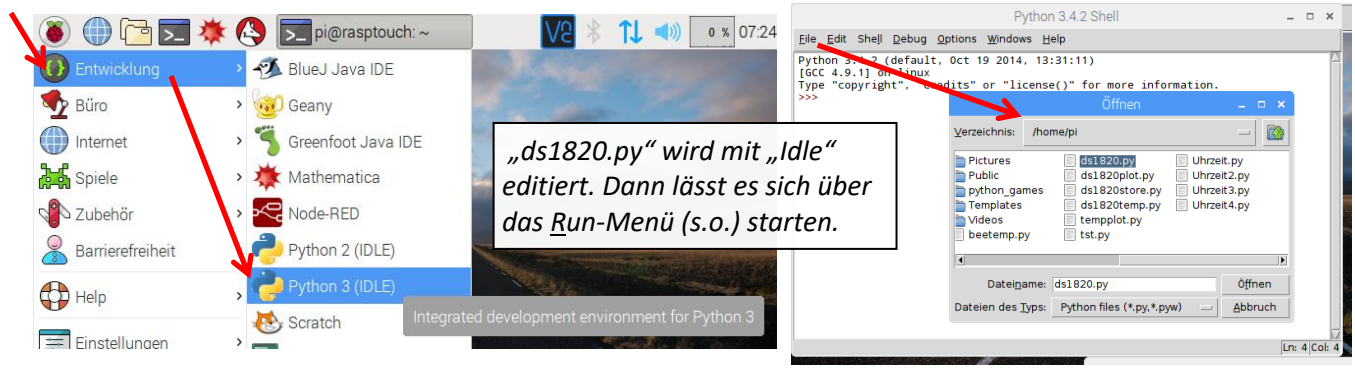
```
File  Edit  Format  Run  SubCode  Options  Window  Help
 1 from time import *
 2 def Sensorenlesen_Schreiben():
 3     Sensorenliste = open('/sys/devices/w1_bus_master1/w1_master_slaves')
 4     w1_slaves = Sensorenliste.readlines()
 5     Sensorenliste.close()
 6     for line in w1_slaves:
 7         w1_slave = line.split("\n")[0];print(w1_slave)
 8         einSensor = open('/sys/bus/w1/devices/' + str(w1_slave) + '/w1_slave')
 9         einSensordaten = einSensor.read();einSensor.close()
10         Messwert = einSensordaten.split("\n")[1].split(" ")[9]
11         temperature = float(Messwert[2:])/1000; temperature = round (temperature,1)
12         temp=str('%5.1f' % temperature)+"°C, "; print(temp)
13 while 1:
14     Sensorenlesen_Schreiben()
15     sleep(10)
```

```
File  Edit  Shell  Debug  Options
Python 3.4.2 (default, Oct 1
[GCC 4.9.1] on linux
Type "copyright", "credits"
>>> ========================
=====
>>>
10-0008030a5463
  22.9°C,
10-0008030a5463
  22.9°C,
10-0008030a5463
  22.9°C,
10-0008030a5463
  22.9°C,
10-0008030a5463
  27.4°C,
10-0008030a5463
```

Links das Programm “ds1820temp.py”, wie es im Fenster der Python-Entwicklungsumgebung “Idle” dargestellt wird, daneben die Ausgabe, die es auf dem Bildschirm erzeugt.(Code zum Kopieren im Anhang S.14)

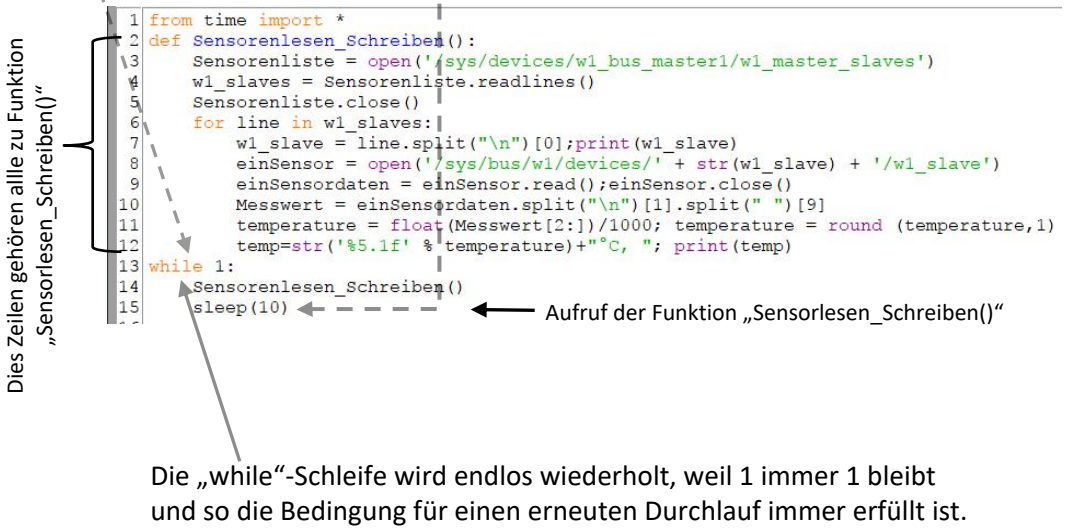
Wie kann „ds1280temp.py“ auf den Raspberry Pi gestartet werden?

Ein Programm kann aus der Programmierumgebung (IDE) „Idle“ heraus über den Run-Befehl oder mit dem LX-Terminal-Befehl „python3 ds1820temp.py“ gestartet werden. Es muss im Verzeichnis „/home/pi/“ des Raspis liegen. Sowohl das LX-Terminal-Programm als auch Idle gehören zu Grundausstattung des Raspis ebenso wie der Python3-Interpreter, also das Programm, das die Ausführung von Python-Programmen übernimmt.
(Alternativ: Das Programm wird in den Desktop-Ordner kopiert und mit Doppelkick gestartet. So öffnen sich „Idle“ und „ds1820temp.py“ automatisch.)



Erläuterungen zum Python-Programm „ds1820temp.py“

Das Modul „time“ wird mit allen seinen Funktionen, darum der Stern (*) importiert , damit die „while“-Schleife zeitgesteuert (hier im 10-Sekunden-Takt) aufgerufen werden kann. (Code zum Kopieren im Anhang S.14)




```
1  from time import *
2  def Sensorenlesen_Schreiben():
3      Sensorenliste = open('/sys/devices/w1_bus_master1/w1_master_slaves')
4      w1_slaves = Sensorenliste.readlines()
5      Sensorenliste.close()
6      for line in w1_slaves:
7          w1_slave = line.split("\n")[0];print(w1_slave)
8          einSensor = open('/sys/bus/w1/devices/' + str(w1_slave) + '/w1_slave')
9          einSensordaten = einSensor.read();einSensor.close()
10         Messwert = einSensordaten.split("\n")[1].split(" ")[9]
11         temperature = float(Messwert[2:])/1000; temperature = round (temperature,1)
12         temp=str('%5.1f' % temperature)+"°C, "; print(temp)
13 while 1:
14     Sensorenlesen_Schreiben()
15     sleep(10)
```

Die „while“-Schleife wird endlos wiederholt, weil 1 immer 1 bleibt und so die Bedingung für einen erneuten Durchlauf immer erfüllt ist.

Die Funktion „Sensorlesen_Schreiben()“ öffnet zunächst die Datei „/sys/devices/w1_bus_master1/w1_master_slaves“, die im Zusammenspiel vom Betriebssystem des Raspberry Pis und dem Sensormodul DS1820 automatisch an genau diesem Ort und mit genau dieser Bezeichnung angelegt wird
Der Inhalt dieser Datei ist eine Liste, die der Variablen „w1_slaves“ zugeordnet wird.
Die Zeilen dieser Liste stellen die Seriennummern der gefundenen Sensoren dar; diese wird Liste mit der Bezeichnung „w1_slaves“ Zeile für Zeile in den Arbeitsspeicher geladen.
Zu den Seriennummern wurden automatisch Order angelegt,
aus denen nun die Messwerte durch „split“-Befehle „herausgeschnitten“ werden.
Diese Messwerte werden dann noch gerundet, formatiert und der als Textvariable „temp“ angefügt, welche in Zeile 6 zunächst als leer (leerer String: „ “) definiert wurde.
So wird pro Durchlauf (also alle drei Sekunden) ein String
mit den Temperaturwerten mittels „print“-Befehl ausgegeben.

Auf **https://www.kompf.de/weather/pionewiremini.html** **(15.12.2016)** haben wir folgende Zusammenstellung von detaillierteren Informationen gefunden: „Die Module [der Software] legen im Verzeichnis /sys/bus/w1/devices jeweils ein Unterverzeichnis für jeden gefundenen Sensor an. Der Name des Verzeichnisses setzt sich aus dem Family-Code des Sensors und seiner eindeutigen Identifikationsnummer zusammen. Sensoren vom Typ DS1820 und DS18S20 haben den Family-Code 10, DS18B20 den Code 28 und DS1822 die 22. In jedem Unterverzeichnis gibt es die Datei w1_slave, die Sensorstatus und gemessenen Temperaturwert enthält:

cd /sys/bus/w1/devices
cd 10-000801b5*
cat w1_slave

```
0f 00 4b 46 ff ff 06 10 0c : crc=0c YES
0f 00 4b 46 ff ff 06 10 0c t=7375
```

Die Datei besteht aus zwei Zeilen, die jeweils den hexadezimalen Registerdump des Sensor-ICs enthalten. Am Ende der ersten Zeile steht die Prüfsumme (CRC) und die Information, ob es sich um einen gültigen Messwert handelt (YES). Die zweite Zeile endet mit dem Temperaturmesswert in tausendstel Grad Celsius. Im Beispiel beträgt die Temperatur also 7,375 °C. Die Genauigkeit auf drei Stellen hinter dem Komma ist natürlich nur scheinbar; dem Datenblatt des DS18S20 entnimmt man zum Beispiel, dass die Messgenauigkeit ±0,5 °C beträgt. Die tatsächliche Temperatur liegt also irgendwo zwischen 6,8 und 7,9 °C.“

IV. DS1820-Temperaturwerte in eine Tabelle schreiben und eine Grafik daraus erstellen

1. In eine Tabelle schreiben

Nachdem wir den Raspberry-Pi dazu gebracht haben, die Temperaturwerte vom Sensor abzuholen und auf dem Bildschirm anzuzeigen, ist die nächste Programmieraufgabe, ihn diese Werte mit der Uhrzeit in eine Tabelle zu schreiben zu lassen. Das erledigt das Programm „ds1820store.py“: (Code zum Kopieren im Anhang S.14)

File Edit Format Run SubCode Options Window Help

```
1  import csv
2  from time import *
3  lt=localtime(); jahr,monat,tag,stunde,minute,sekunde = lt[0:6]
4  DateiName = "Bienenstock-1-Temperaturen-"+str(jahr)+"-"+str(monat)+"-"+str(tag)+".csv"
5
6  def Sensorenlesen_Schreiben():
7      global DateiName
8      lt=localtime(); stunde,minute = lt[3:5]
9      Temp_Proto_Datei = open(DateiName,"a")
10     Temp_Proto_Datei.write("%02i"%(stunde)+":%02i"%(minute)+",")
11     for line in w1_slaves:
12         w1_slave = line.split("\n")[0];print(w1_slave)
13         einSensor = open('/sys/bus/w1/devices/' + str(w1_slave) + '/w1_slave')
14         einSensordaten = einSensor.read();einSensor.close()
15         Messwert = einSensordaten.split("\n")[1].split(" ")[9]
16         temperature = float(Messwert[2:])/1000; temperature = round (temperature,1)
17         print(temperature)
18         Temp_Proto_Datei = open(DateiName,"a")
19         Temp_Proto_Datei.write(str('%5.1f' % temperature+","))
20     Temp_Proto_Datei.write("\n"); Temp_Proto_Datei.close()
21     sleep(5)
22
23 Sensorenliste = open('/sys/devices/w1_bus_master1/w1_master_slaves')
24 w1_slaves = Sensorenliste.readlines()
25 Sensorenliste.close()
26
27 while True:
28     Sensorenlesen_Schreiben()
```

Als Ergänzung zu „ds1820.py“ wurde zunächst das Modul „csv“ benötigt.
Bei diesem Quellcode sollte man versuchen zu verstehen, wie dem Bezeichner „Dateiname“ in Zeile 4 ein String zugeordnet wird, der aus Thema, Zeitangabe und Dateitypen-Bezeichnung (.csv) zusammensetzt wird - ganz ähnlich, wie einer Variablen ein Wert zugeordnet wird.
Dieser Bezeichner wird dann in der Funktion „Sensorenlesen_Schreiben()“ in Zeile 13 genutzt, um eine Datei (ein Dateiobjekt) mit diesem Namen zu erstellen, indem sie geöffnet wird.
Der Zugriff darauf ist möglich über den Bezeichner „Temp_Proto_Datei“. Als erste Methode wird „.write(…)“ für das Schreiben der Uhrzeit in diese Datei genutzt. Doch zuvor muss im Anweisungsteil ab Zeile 23 – wie bereits in „ds1820.py“ – die Datei mit den Dateinamen für die Tabellen mit den Sensorwerten ausgelesen werden, damit „Sensorenlesen_Schreiben()“ darauf zugreifen kann. Und schließlich wird in Zeile 27 die Hauptschleife definiert, in der immer wieder die o.g. Funktion aufgerufen wird.
Die zeitliche Taktung erfolgt über den Aufruf der sleep()-Funktion in Zeile 21. Wenn die Hardware richtig angeschlossen ist und das Programm aus der Programmierumgebung (IDE) „Idle“ heraus oder mit dem LX-Terminal-Befehl „python3 ds1820store.py“ gestartet wird, sollte sich nun die Datei „Bienenstock-1-Temperaturen-2017_xx_xx.csv“ mit Zeit- und Temperaturwerten füllen; sie liegt im gleichen Verzeichnis „/home/pi/“ wie auch „ds1820store.py“.

2. Grafik erzeugen

Nachdem es gelungen ist, die CSV-Datei zu erstellen, können wir ein weiteres nützliches Programm nutzen, das wir allerdings nicht direkt einbinden können. Stattdessen muss zunächst das Modul „subprocess" eingebunden werden, welches dann zum Aufruf des eigenständigen Programms „gnuplot" genutzt werden kann, nachdem es mittels „sudo apt-get install gnuplot-x11" über den LX-Monitor installiert wurde.

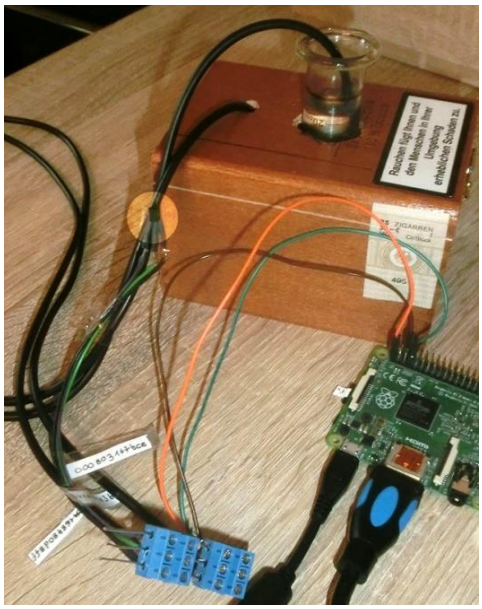
Das Programm „ds1820plt.py", welches auf der nächsten Seite gezeigt wird, nutzt diese Möglichkeiten. Die Bilddatei wird in unserem Fall immer wieder überschrieben, damit es nach dem Einbinden in eine Webseite immer wieder auffrischt werden kann. (Code zum Kopieren im Anhang S.14)

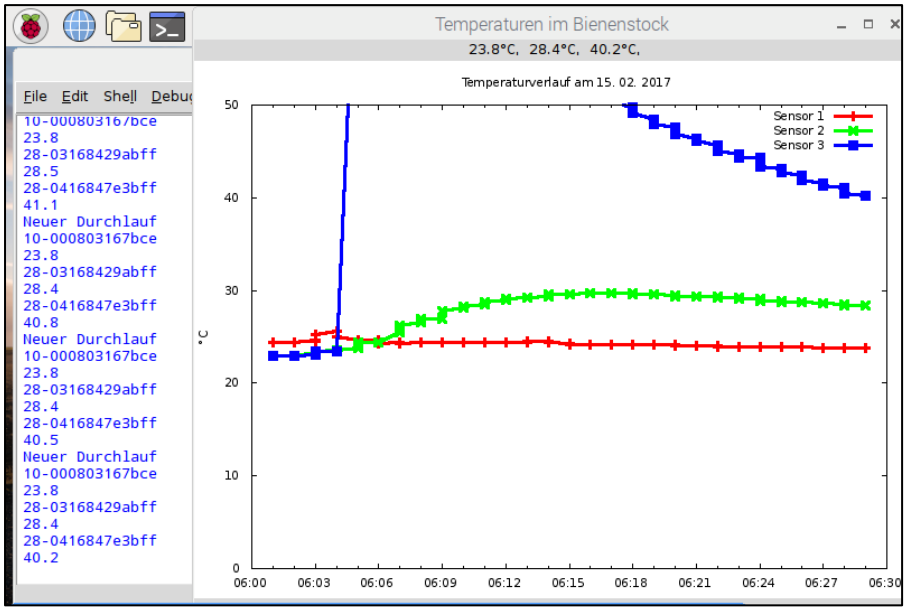
ds1820plot.py - \\RASPTOUCH\Pi-Share1\home\pi\ds1820plot.py (3.5.1)

File Edit Format Run SubCode Options Window Help

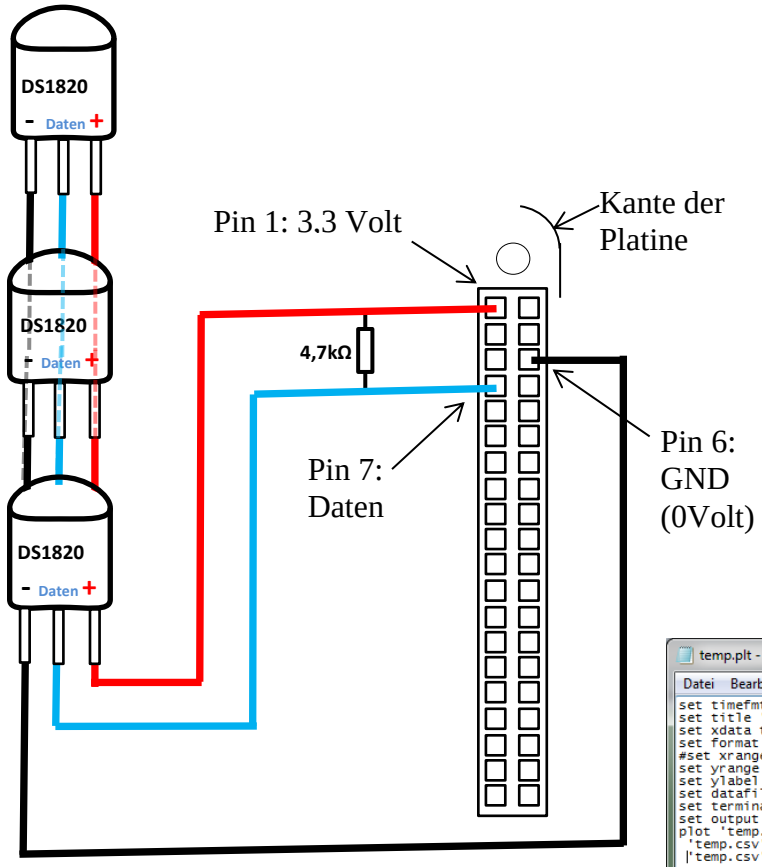
```
import subprocess, csv
from time import *
lt=localtime(); jahr,monat,tag,stunde,minute,sekunde = lt[0:6]
DateiName = "Bienenstock-1-Temperaturen-"+str(jahr)+"-"+str(monat)+"-"+str(tag)+".csv"

def Sensorenlesen_Schreiben():
    global DateiName
    lt=localtime(); stunde,minute = lt[3:5]
    Temp_Proto_Datei = open(DateiName,"a")
    Temp_Proto_Datei.write("%02i"%(stunde)+":%02i"%(minute)+",")
    for line in w1_slaves:
        Temp_Proto_Datei = open(DateiName,"a")
        w1_slave = line.split("\n")[0];print(w1_slave)
        einSensor = open('/sys/bus/w1/devices/' + str(w1_slave) + '/w1_slave')
        einSensordaten = einSensor.read();einSensor.close()
        Messwert = einSensordaten.split("\n")[1].split(" ")[9]
        temperature = float(Messwert[2:])/1000; temperature = round (temperature,1)
        Temp_Proto_Datei.write(str('%5.1f' % temperature+","))
        temp=str('%5.1f' % temperature)+"°C, "; print(temp)
    Temp_Proto_Datei.write("\n"); Temp_Proto_Datei.close()
def Grafik_Erzeugen():
    a = subprocess.call(["sudo", "cp", DateiName, "temp.csv"])
    a = subprocess.call(["gnuplot","temp.plt"])
    a = subprocess.call(["sudo","cp","temp.png",'/home/pi/Desktop'])
    sleep(5)

Sensorenliste = open('/sys/devices/w1_bus_master1/w1_master_slaves')
w1_slaves = Sensorenliste.readlines()
Sensorenliste.close()

while True:
    Sensorenlesen_Schreiben()
    Grafik_Erzeugen()
```

V. Mehrere DS1820-Sensoren am Raspberry Pi: Temperaturprofile im zeitlichen Verlauf erstellen

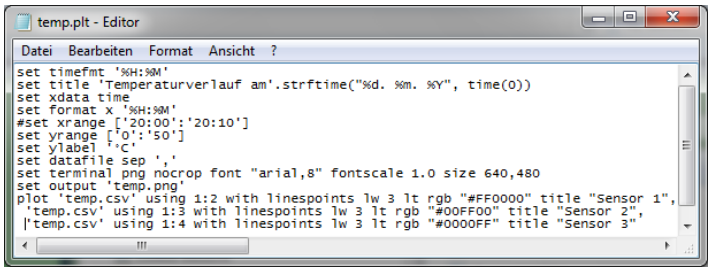




Hat man mehrere der o.g. Sensoren an den Raspberry Pi angeschlossen und liest man die Daten mit dem Programm „beetemp.py“ (siehe Anhang; Programm wird später genauer erklärt) aus, dann können die Daten auch gleich als Diagramme entsprechend der Abbildung oben rechts dargestellt werden. Die Liste in der Mitte erscheint im Protokollfenster der Python-Entwicklungs-umgebung „Idle“, wenn in dieser das Programm „beetemp.py“ editiert (geöffnet) und dann gestartet wurde.
Das Schaltungsschema ist einfach: Der eine 4,7kΩ-Widerstand wird (wie bei dem einzelnen Sensor) angeschlossen, die weiteren Sensoren werden einfach parallel verbunden – wie unten zu sehen.



Besondere Beachtung sollte die Montage der einzelnen Sensormodule finden. Erfreulicherweise gibt es fertig konfektionierte, wasserdicht versiegelte DS18B20-Temperatursensoren mit 1m-Kabel sowie „3-Pin-5,08-mm-Pitch“-Blöcke, wie sie im Foto oben zu sehen sind. So lassen sich die Verbindungen übersichtlich gestalten und sind gut gegen Kurzschlüsse geschützt.
Zusätzlich zum Python-Programm „beetemp.py“ muss das „Gnuplot“-Programm „tmp.plt“ im Programmordner „pi“ des Raspis gespeichert werden und Gnuplot muss auf dem Rechner installiert sein, wenn die Diagrammerstellung automatisch ablaufen soll.
(Code zum Kopieren im Anhang S.14)

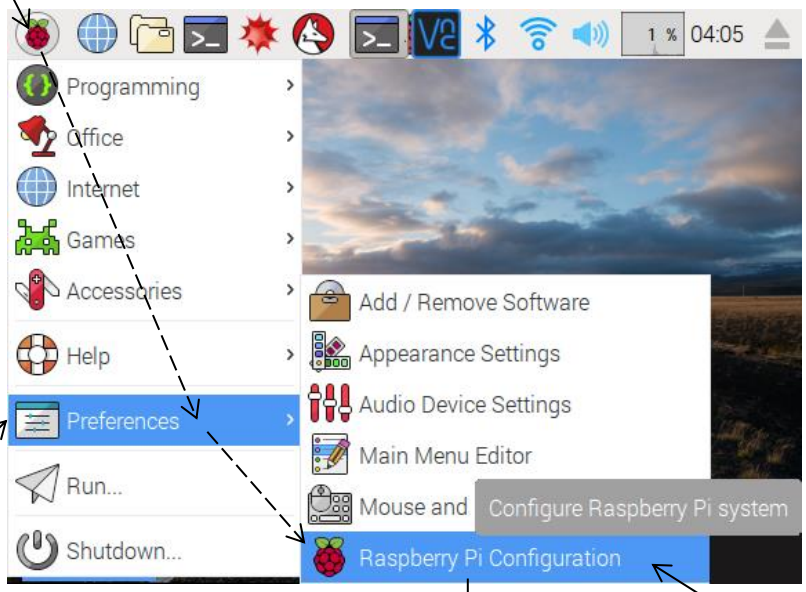


VI. Den Raspberry Pi über Smartphone, Tablet oder Windows-Rechner fernsteuern

a) Vorbereitungen am Raspberry Pi – den VNC-Server in Betrieb nehmen

Weil der Raspberry-Pi (Raspi) so klein und billig ist, kann er in Anlagen eingebaut werden und dort bleiben, ohne dass man übertrieben Angst haben muss, dass der ausgebaut und gestohlen wird, wenn er nicht ständig überwacht wird. Und im eingebauten Zustand müssen nicht einmal seine Anschlussbuchsen nach außen zugänglich sein, denn wir können über WLAN* auf ihn zugreifen und ihn fernsteuern. Dazu ist auf dem Raspi (beim Betriebssystem PIXEL) als ein sogenannter VNC-Server gleich in der Grundausstattung dabei. Bei der Einrichtung muss dieser allerdings freigeschaltet werden. Dazu muss der Raspi zunächst mit Tastatur und HDMI-Monitor betrieben werden. Vom PIXEL-Desktop aus geht man folgendermaßen vor:

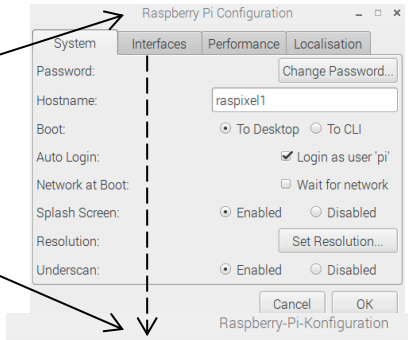
1. Auf die Himbeere klicken!

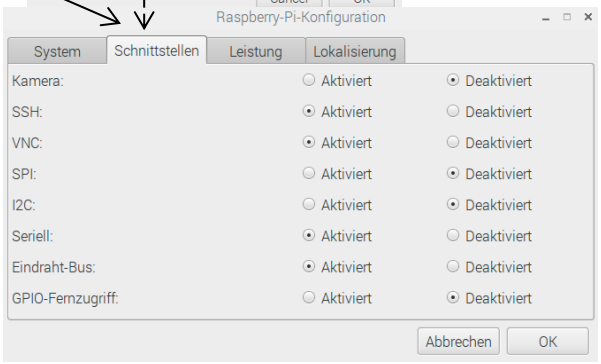


2. Mauszeiger über „Preferences" ziehen!

3. Mauszeiger weiter über „Raspberry Pi Configuration" ziehen und dann klicken!

4. In den „Raspberry Pi Configurations" gibt es einen Reiter mit der Bezeichnung „Interfaces". Wird dieser Reiter geklickt, kommt man zur Übersicht der Schnittstellen. („Schnittstellen" ist die Übersetzung von „Interfaces".) Wir müssen SSH und VNC aktivieren, damit die Fernsteuerung funktioniert. Den Eindrahtbus (engl. 1-Wire bzw. One-Wire-Bus) aktivieren wir auch, weil wir ihn später für die Temperatursensoren benötigen.

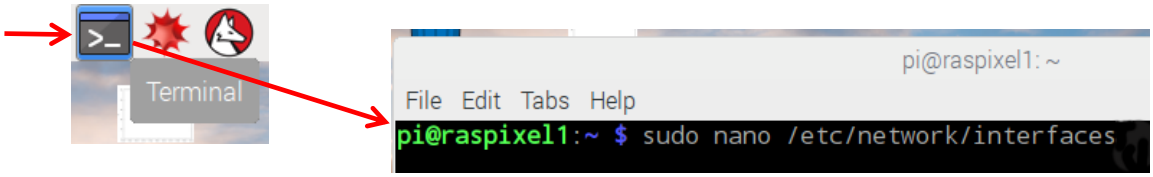


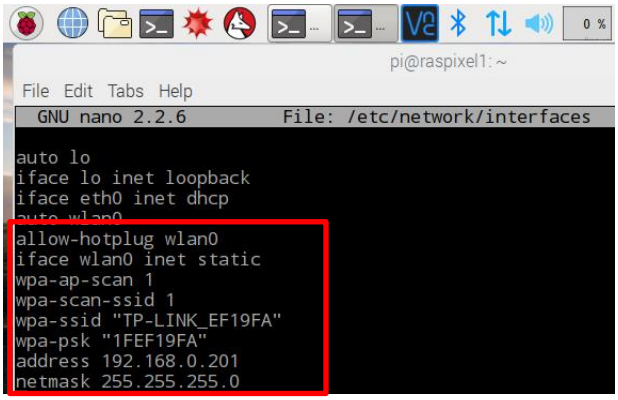


* Für die WLAN-Einrichtung auf dem Raspi gibt es Extraanleitungen. Der Raspi 3 hat schon ein Modul eingebaut, ältere Raspis muss man eine WLAN-Stick nachrüsten. Ist das WLAN eingerichtet, kann man sich dem Rechner, Tablet oder Smartphone zuwenden.

b) Raspi zwecks VNC-Zugriff mit einer statischen IP-Adresse versehen für WLAN und Ethernet (oder USB bei Android-Geräten)

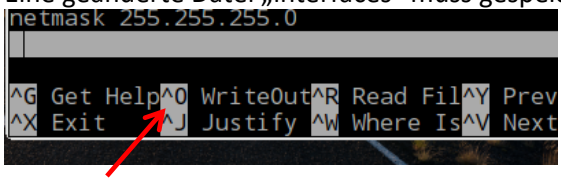
Damit am Bienenstock, im Gelände etc. der Zugriff auf den Raspberry- Pi (Raspi) möglich ist, kann ein kleiner Router wie z.B. der TP-Link TL-WR702N benutzt werden. Das WLAN, welches dieser Router dann bereitstellt, sobald er über Mikro-USB-Anschluss mit Strom versorgt wird, hat einen Namen wie z.B. hier „TP-LINK_EFF19A“ und verlangt zum Einloggen ein Passwort wie z.B. „1FEF19FA“.
Zur Stromversorgung bietet sich eine aufladbare Powerbank an.
Sollen Daten auch gleich ins Internet hochgeladen werden, muss der Router mit einer SIM-Karte bestückt sein. Dieses kann z.B. das HUAWEI-E5330.
Damit ein Raspi in dem mobilen WLAN eine feste IP-Adresse bekommt, die einen VNC-Zugriff über ein Tablet, Smartphone oder Rechner ermöglicht, müssen WLAN-Name und -Passwort in der Datei „interfaces“ eingetragen sein, welches sich im Verzeichnis /etc/network/ befindet und mit dem Befehl „sudo nano /etc/network/ interfaces“ geöffnet und dann geändert werden kann, wenn man diesen zunächst in ein geöffnetes Terminalfenster eingetragen hat.

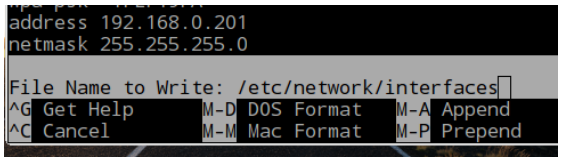




An den Einträgen dort lässt sich ablesen:
Das WLAN bekommt nicht über das DHCP (Dynamic Host Configuration Protocol) sondern statisch(static) die IP-Adresse zugewiesen
und die Zugangsdaten mitgeteilt.
Bei der Wahl der IP-Adresse muss darauf geachtet werden, dass sie nicht im Bereich liegt, der vom DHCP verwaltet wird. Bei unseren Router ist das – bezogen auf die letzte Zahl - der Bereich bis 199, darum wurde 201 gewählt. Die Gruppe der ersten drei Zahlen bleibt unverändert 192.168.0.

Eine geänderte Datei „interfaces“ muss gespeichert werden. Das geschieht so:



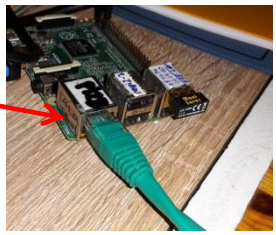


Mit „Strg o“ wird das Eingabefeld verlassen und der Name der Datei, in die gespeichert werden soll, wird angezeigt. Da der Name „interfaces“ bleiben soll, muss nur „Enter“ gedrückt werden.

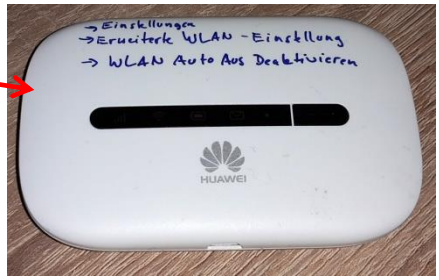
Dann muss das Netzwerk neu gestartet werden mit „sudo service networking restart“
Mit „ip addr“ wird dann überprüft, ob die neue IP-Adresse tatsächlich eingestellt wurde.

Wenn auch der Ethernet- sowie der USB-Aschluss für den VNC-Zugriff nutzbar sein sollen, bieten sich für unsere Zwecke folgende Einträge in „interfaces“ an, wobei hier als Beispiel für das WLAN der „HUAWEI-E5330“-Router eingesetzt wurde:

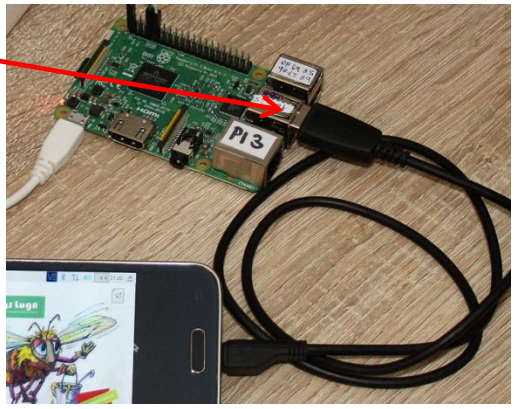
```
auto eth0
allow-hotplug eth0
iface eth0 inet static
address 169.254.175.201
auto lo
iface lo inet loopback
iface eth0 inet dhcp
```



```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
wpa-ap-scan 1
wpa-scan-ssid 1
wpa-ssid "HUAWEI-E5330-9B17"
wpa-psk "a734h2m5"
address 192.168.8.210
```




```
allow-hotplug usb0
auto usb0
iface usb0 inet static
address 192.168.42.42
netmask 255.255.255.0
network 192.168.42.0
broadcast 192.168.42.255
```

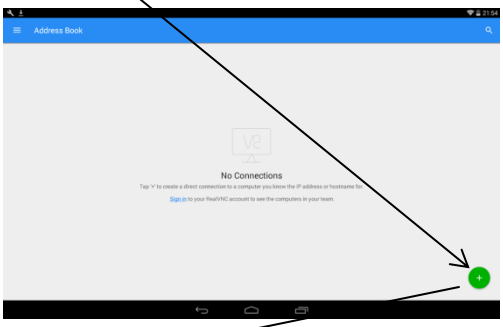


c) Zugriff mittels VNC-Viewer, der auf dem Smartphone, Tablet oder Rechner installiert ist

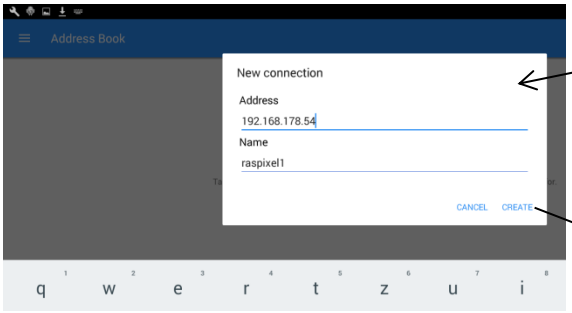
Für den Schulungseinsatz im Gelände ist ein Tablet besonders geeignet, da es einen guten Kompromiss zwischen Größe und Ablesbarkeit darstellt. Auf dem Smartphone oder dem Rechner läuft die Software dann ganz ähnlich.

1. Den VNC-Viewer gibt im Playstore für Android und auf der Seite https://www.realvnc.com/download/viewer/ für Rechner zum Download für die Installation.

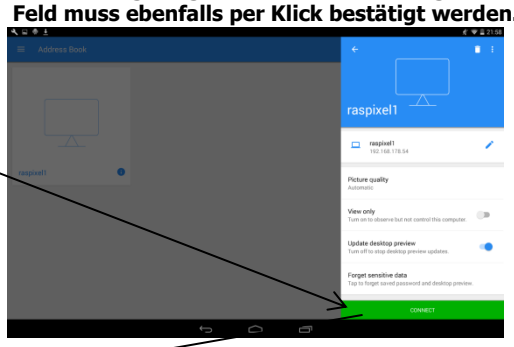
2. Wird der Viewer das erste Mal gestartet, wird zunächst noch keine Verbindung angezeigt. Um eine Verbindung herzustellen, muss auf den grünen Punkt mit dem weißen + geklickt werden.



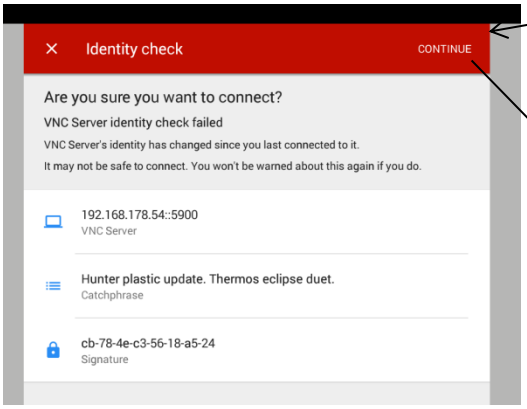
3. Zur Einrichtung der neuen Verbindung muss die IP-Adresse des Raspis bekannt sein. Wie diese festgelegt oder ermittelt wird, steht in der WLAN-Anleitung. Außerdem sollte man einen gut merkbaren Namen für die Verbindung gewählen.



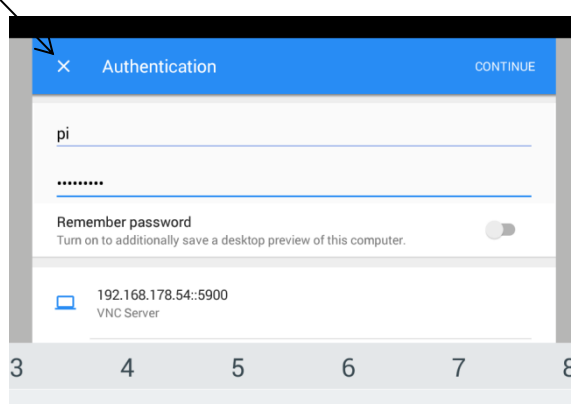
4. Nach dem Klick auf „CREATE" wird dieser Bildschirm gezeigt. Das „CONNECT" im grünen Feld muss ebenfalls per Klick bestätigt werden.



5. Falls es zu einen Problem beim „Identity check" kommt, muss „CONTINUE" geklickt werden. Es ist aber auch möglich, dass man gleich zu „Authentication" gelangt.



5. Auf dem Raspi ist man immer als Nutzer (user) aktiv. Wenn nicht geändert wurde, heißt dieser „pi" und das (noch nicht geänderte Passwort dazu „raspberry". Über „CONTINUE" geht es weiter zum Desktop des Raspis.

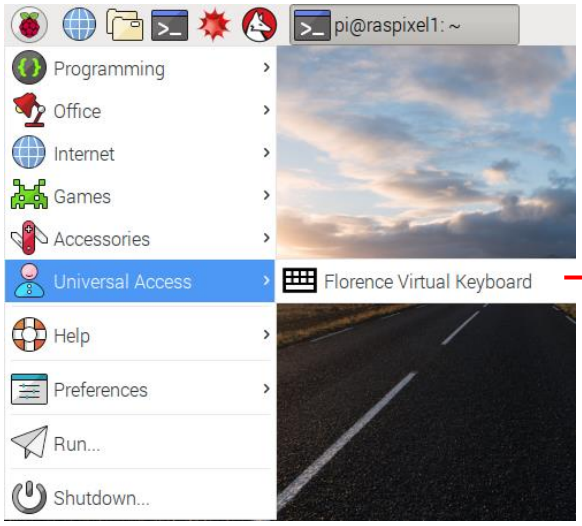


d) Bedienung des VNC-Viewers zu Steuerung des Raspis

Da man beim Tablett ohne Maus und Keyboard auskommen muss, ist die Bedienung nicht ganz einfach.
Zum Glück gibt es Hilfe-Funktion. Für viele Situationen ist es gut, die Mausfunktion aktiv zu schalten und die Stecknadel ebenfalls. Das geht durch Tippen auf das halbtransparente Bedienfeld. Achtung: Der Mauszeiger ist so eingerichtet, dass er nicht unter, sondern abseits des Fingers erscheint und geführt wird. Viel Spaß beim Testen! (Die Fernsteuerung über den Rechner mit echter Tastatur und entsprechender Mauszeiger-Bedienung ist die Bedienung komfortabler, aber man muss ggf. sein Notebook dabei haben.)

So ähnlich sieht der Desktop aus, wenn man Zugriff über ein Android-Tablet hat. Wichtig ist die Bedienleiste, die halb transparent erscheint.
In diesem Beispiel sind die Feststell-Stecknadel für den Mauszeiger und die Maus aktviert. Das X ermöglicht es, den VNC-Viewer komplett zu schließen.

Falls sich die VNC-Viewer-Tastatur nicht öffnen lässt, kann man mittels
„sudo apt-get install at-spi2-core florence“ auf dem Raspberry eine Software-Tastatur installieren. Sie ist dann über den Hauptmenüpunkt „Universal Access“ aufrufbar und kann über das Werkzeugsymbol eingestellt werden, z.B. über „Layout“, dass auch die Navigationstasten sichtbar sind oder – falls die Tastatur doppelt Zeichen setzt – dass der Timer hochgesetzt wird.

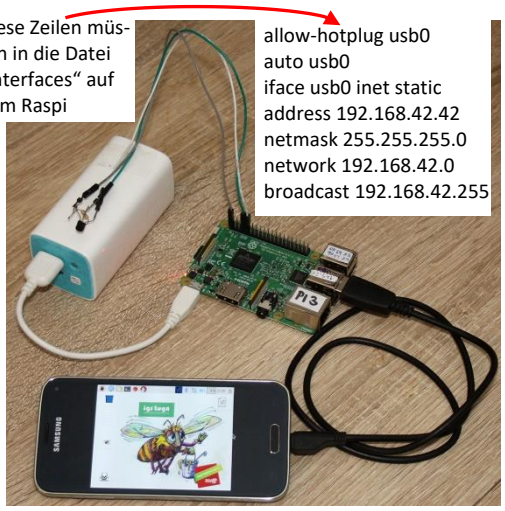


e) Geländegängig: Raspi über USB an Android-Smartphone koppeln - falls auf WLAN- Einbindung und Upload auf einen Server verzichtet werden kann

Ein Raspberry („Raspi“, Typ 2 oder 3) zusammen mit einer 8-GB-SD-Karte und einer USB-Powerbank ist für etwa 45 € erhältlich. Wenn man dann ein nicht allzu altes Android-Gerät hat, lässt sich damit dieser Raspi mit PIXEL-Betriebssystem so nutzen, dass die Geräte auch für Wald und Flur, Schulgarten und Labor im Wortsinn tragbar sind.

Für diesen Zweck muss der Raspi allerdings an einem größeren Laptop oder anderen Windows-Rechner vorbereitet werden. Dafür gibt es genauere Anleitungen weiter vorne in diesem Skript.

Zentral ist, dass in der Datei „interfaces“ die Zeilen im Kasten rechts oben im Bild eingetragen sind. Wie man sie dorthin bekommt, wird ebenfalls weiter oben erklärt. (Außerdem muss der VNC-Server des Raspis aktiviert sein, was er in der Regel ist.)



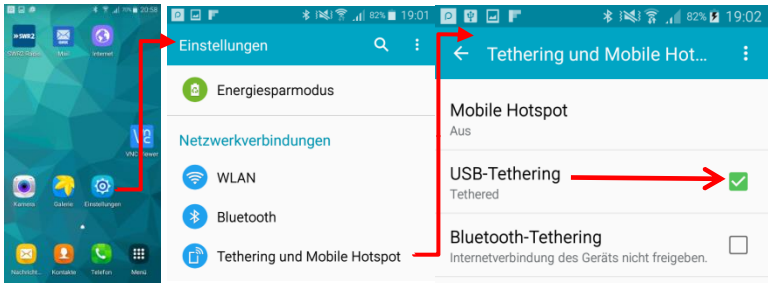
Auf das Smartphone muss über Google-Play eine VNC-Viewer-App installiert worden sein.

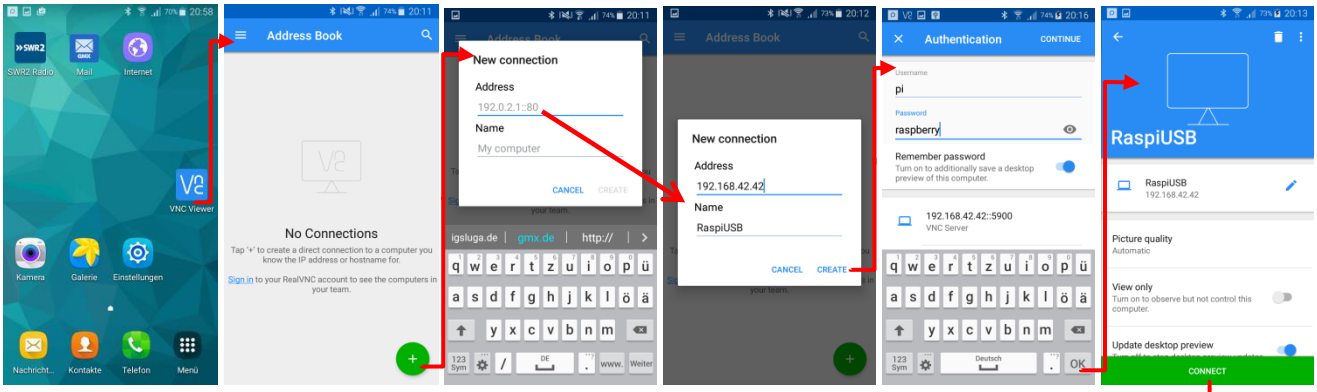
Wenn das alles geschehen ist, muss der Raspi zunächst über den Mikro-USB-Anschluss auf der Raspi-Platine an eine Powerbank angesteckt werden, sodass er hochfährt, was man am Blinken erkennt. Nun muss das Smartphone mit einem USB-Kabel an einen der großen USB-Buchsen auf dem Raspi angeschlossen werden. Weiter muss am Android-Gerät über **„Einstellungen => Netzwerkverbindungen => Tehtering“** das **„USB-Tethering“** aktiviert werden, was nur gelingt, wenn der Raspberry tatsächlich angeschlossen ist. Anschließend geht es zurück zum Startbildschirm.

Dort wird der **VNC-Viewer** (nach der USB-Tethering-Aktivierung) gestartet und eine „new connection“ hinzugefügt, wo als „address“ lediglich „192.168.42.42“ eingetragen wird sowie ein Name, der frei wählbar ist, z.B. „RaspiUSB“. Nun auf den Connect-Button tippen, ggf. einen Sicherheitshinweis bestätigen und dann als Username „pi“ und als Passwod „raspberry“ eintragen. Außerdem sollte man die Passwort-Speicherung aktivieren und „OK“ tippen.

Nun sollte sich der **Raspi-Desktop** auf dem Smartphone öffnen - und das Üben mit der Navigation auf dem kleinen Bildschirm kann beginnen!

Rechts: Aktivierung des USB-Tetherings auf dem Android- Gerät
Mitte: Herstellung der VNC-Verbindung zum Raspi
Unten: Raspi-Arbeitsplatz





VII. Code zum Kopieren

#Code ds1820temp.py

```
from time import *
def Sensorenlesen_Schreiben():
    Sensorenliste = open('/sys/devices/w1_bus_master1/w1_master_slaves')
    w1_slaves = Sensorenliste.readlines()
    Sensorenliste.close()
    for line in w1_slaves:
        w1_slave = line.split("\n")[0];print(w1_slave)
        einSensor = open('/sys/bus/w1/devices/' + str(w1_slave) + '/w1_slave')
        einSensordaten = einSensor.read();einSensor.close()
        Messwert = einSensordaten.split("\n")[1].split(" ")[9]
        temperature = float(Messwert[2:])/1000; temperature = round (temperature,1)
        temp=str('%5.1f' % temperature)+"Â°C, "; print(temp)
while 1:
    Sensorenlesen_Schreiben()
    sleep(10)
```

#Code ds1820store.py

```
import csv
from time import *
lt=localtime(); jahr,monat,tag,stunde,minute,sekunde = lt[0:6]
DateiName = "Bienenstock-1-Temperaturen-"+str(jahr)+"-"+str(monat)+"-"+str(tag)+".csv"

def Sensorenlesen_Schreiben():
    global DateiName
    lt=localtime(); stunde,minute = lt[3:5]
    Temp_Proto_Datei = open(DateiName,"a")
    Temp_Proto_Datei.write("%02i"%(stunde)+":%02i"%(minute)+",")
    for line in w1_slaves:
        w1_slave = line.split("\n")[0];print(w1_slave)
        einSensor = open('/sys/bus/w1/devices/' + str(w1_slave) + '/w1_slave')
        einSensordaten = einSensor.read();einSensor.close()
        Messwert = einSensordaten.split("\n")[1].split(" ")[9]
        temperature = float(Messwert[2:])/1000; temperature = round (temperature,1)
        print(temperature)
          Temp_Proto_Datei = open(DateiName,"a")
          Temp_Proto_Datei.write(str('%5.1f' % temperature+","))
    Temp_Proto_Datei.write("\n"); Temp_Proto_Datei.close()
    sleep(5)

Sensorenliste = open('/sys/devices/w1_bus_master1/w1_master_slaves')
w1_slaves = Sensorenliste.readlines()
Sensorenliste.close()

while True:
    Sensorenlesen_Schreiben()
```

#Code ds1820plot.py

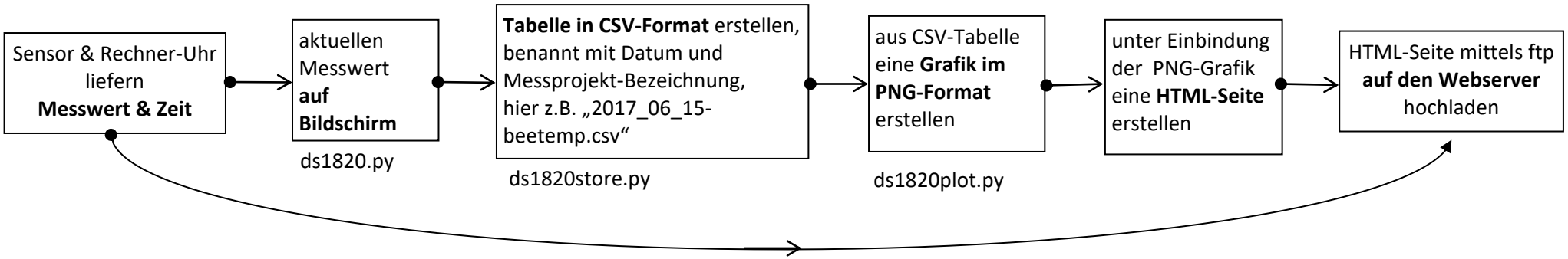
```
import subprocess, csv
from time import *
lt=localtime(); jahr,monat,tag,stunde,minute,sekunde = lt[0:6]
DateiName = "Bienenstock-1-Temperaturen-"+str(jahr)+"-"+str(monat)+"-"+str(tag)+".csv"

def Sensorenlesen_Schreiben():
    global DateiName
    lt=localtime(); stunde,minute = lt[3:5]
    Temp_Proto_Datei = open(DateiName,"a")
    Temp_Proto_Datei.write("%02i"%(stunde)+":%02i"%(minute)+",")
    for line in w1_slaves:
        Temp_Proto_Datei = open(DateiName,"a")
        w1_slave = line.split("\n")[0];print(w1_slave)
        einSensor = open('/sys/bus/w1/devices/' + str(w1_slave) + '/w1_slave')
        einSensordaten = einSensor.read();einSensor.close()
        Messwert = einSensordaten.split("\n")[1].split(" ")[9]
        temperature = float(Messwert[2:])/1000; temperature = round (temperature,1)
        Temp_Proto_Datei.write(str('%5.1f' % temperature+","))
        temp=str('%5.1f' % temperature)+"Â°C, "; print(temp)
    Temp_Proto_Datei.write("\n"); Temp_Proto_Datei.close()

def Grafik_Erzeugen():
    a = subprocess.call(["sudo", "cp", DateiName, "temp.csv"])
    a = subprocess.call(["gnuplot","temp.plt"])
    a = subprocess.call(["sudo","cp","temp.png",'/home/pi/Desktop'])
    sleep(5)

Sensorenliste = open('/sys/devices/w1_bus_master1/w1_master_slaves')
w1_slaves = Sensorenliste.readlines()
Sensorenliste.close()

while True:
    Sensorenlesen_Schreiben()
    Grafik_Erzeugen()
```

```
#temp.plt fuer 3 Sensoren
set timefmt '%H:%M'
set title 'Temperaturverlauf am '.strftime("%d. %m. %Y", time(0))
set xdata time
set format x '%H:%M'
#set xrange ['20:00':'20:20'] falls nur eine bestimmte Zeitspanne geplotte werde soll
set yrange ['0':'50']
set ylabel '°C'
set datafile sep ','
set terminal png nocrop font "arial,8" fontscale 1.0 size 640,480
set output 'temp.png'
plot 'temp.csv' using 1:2 with linespoints lw 3 lt rgb "#FF0000" title "Sensor 1"
'temp.csv' using 1:3 with linespoints lw 3 lt rgb "#00FF00" title "Sensor 2",
'temp.csv' using 1:4 with linespoints lw 3 lt rgb "#0000FF" title "Sensor 3"
```

#Code Beteemp.py komplett

```
import os,sys,subprocess, csv,ftplib;from time import *;from tkinter import *;import RPi.GPIO as GPIO
global DateiName,tempLabel,ftpLabel,bildLabel,root,tkinter,bild
lt=localtime(); jahr,monat,tag,stunde,minute,sekunde = lt[0:6]
root = Tk();root.geometry("640x530+100+100");root.title("Temperaturen im Bienenstock")
tempLabel= Label(root,text="Warten auf Daten!");tempLabel.pack()
bild=PhotoImage(file="temp.png")
bildLabel=Label(root,image=bild);bildLabel.pack()
ftpLabel= Label(root,text=" ");ftpLabel.pack();root.update_idletasks()
DateiName = "Bienenstock-1-Temperaturen-"+str(jahr)+"-"+str(monat)+"-"+str(tag)+".csv"
print(DateiName)
def Sensorenlesen_Schreiben_in_Datei():
   lt=localtime(); stunde,minute = lt[3:5]
   global DateiName,tempLabel,ftpLabel,bildLabel,root,tkinter
   Temp_Proto_Datei = open(DateiName,"a")
   Temp_Proto_Datei.write("%02i"%(stunde)+":%02i"%(minute)+",")
   Sensorenliste = open('/sys/devices/w1_bus_master1/w1_master_slaves')
   w1_slaves = Sensorenliste.readlines()
   Sensorenliste.close()
   temp = " "
   for line in w1_slaves:
      w1_slave = line.split("\n")[0];print(w1_slave)
      einSensor = open('/sys/bus/w1/devices/' + str(w1_slave) + '/w1_slave')
      einSensordaten = einSensor.read();einSensor.close()
      Messwert = einSensordaten.split("\n")[1].split(" ")[9]
      temperature = float(Messwert[2:])/1000; temperature = round (temperature,1)
      Temp_Proto_Datei.write(str('%5.1f' % temperature+","))
      temp=temp + str('%5.1f' % temperature)+"°C, "; print(temperature)
   Temp_Proto_Datei.write("\n"); Temp_Proto_Datei.close()
   tempLabel.config(text=temp);tempLabel.pack();root.update_idletasks()
def HTML_Erzeugen():
   reader = csv.reader(open(DateiName))
   htmlfile = open("Bienenstock-1-Temperaturen-"+str(jahr)+"-"+str(monat)+"-"+str(tag)+".html","w")
   htmlfile.write('<table border = "1">')
   rownum = 0
   try:
      for row in reader:
         for column in row:
            htmlfile.write('<td>' + column + '</td>')
         htmlfile.write('</tr>');rownum += 1
      htmlfile.write('</table>');htmlfile.close()
   except:
      print ("Umwandlungsfehler!")
      pass
def Grafik_Erzeugen():
   global tempLabel,ftpLabel,bildLabel,root,tkinter
   a = subprocess.call(["sudo", "cp", DateiName, "temp.csv"])
   a = subprocess.call(["gnuplot","temp.plt"])
   a = subprocess.call(["sudo","cp","temp.png",'/home/pi/Desktop'])
   bild=PhotoImage(file="temp.png")
   bildLabel.config(image=bild);bildLabel.pack();root.update_idletasks()
   sleep(20)
def Dateien_Hochladen():
   global tempLabel,ftpLabel,bildLabel,root,tkinter
   try:
      uploadfile = "Bienenstock-1-Temperaturen-"+str(jahr)+"-"+str(monat)+"-"+str(tag)+".html"
      file=open(uploadfile,'rb')
      session=ftplib.FTP('igsluga.lima-ftp.de','igsluga','bPLj7bvkGg')
      session.storbinary('STOR /igsluga.lima-city.de/'+uploadfile,file);file.close()
      uploadfile = "temp.png";file=open(uploadfile,'rb')
      session.storbinary('STOR /igsluga.lima-city.de/temp-server.png',file); file.close();session.quit
      ftpLabel.config(text="Upload ok!"); ftpLabel.pack(); root.update_idletasks()
   except:
      ftpLabel.config(text="Upload-Fehler!"); ftpLabel.pack();root.update_idletasks(); print("FTP-Fehler")
      pass
def messchleife():
   global tempLabel,ftpLabel,bildLabel,root,tkinter
   Sensorenlesen_Schreiben_in_Datei(); Grafik_Erzeugen(); HTML_Erzeugen(); Dateien_Hochladen()
   print("Neuer Durchlauf")
   root.after(10,messchleife)
root.after(10,messchleife)
root.mainloop()
```

VIII. Datenfluss beim „Temperatur-im-Bienenstock-Projekt“



Messwerte in Tabelle speichern und als Grafik in HTML-Seite auf Webserver

wird komplett von „beetemp.py“ erledigt