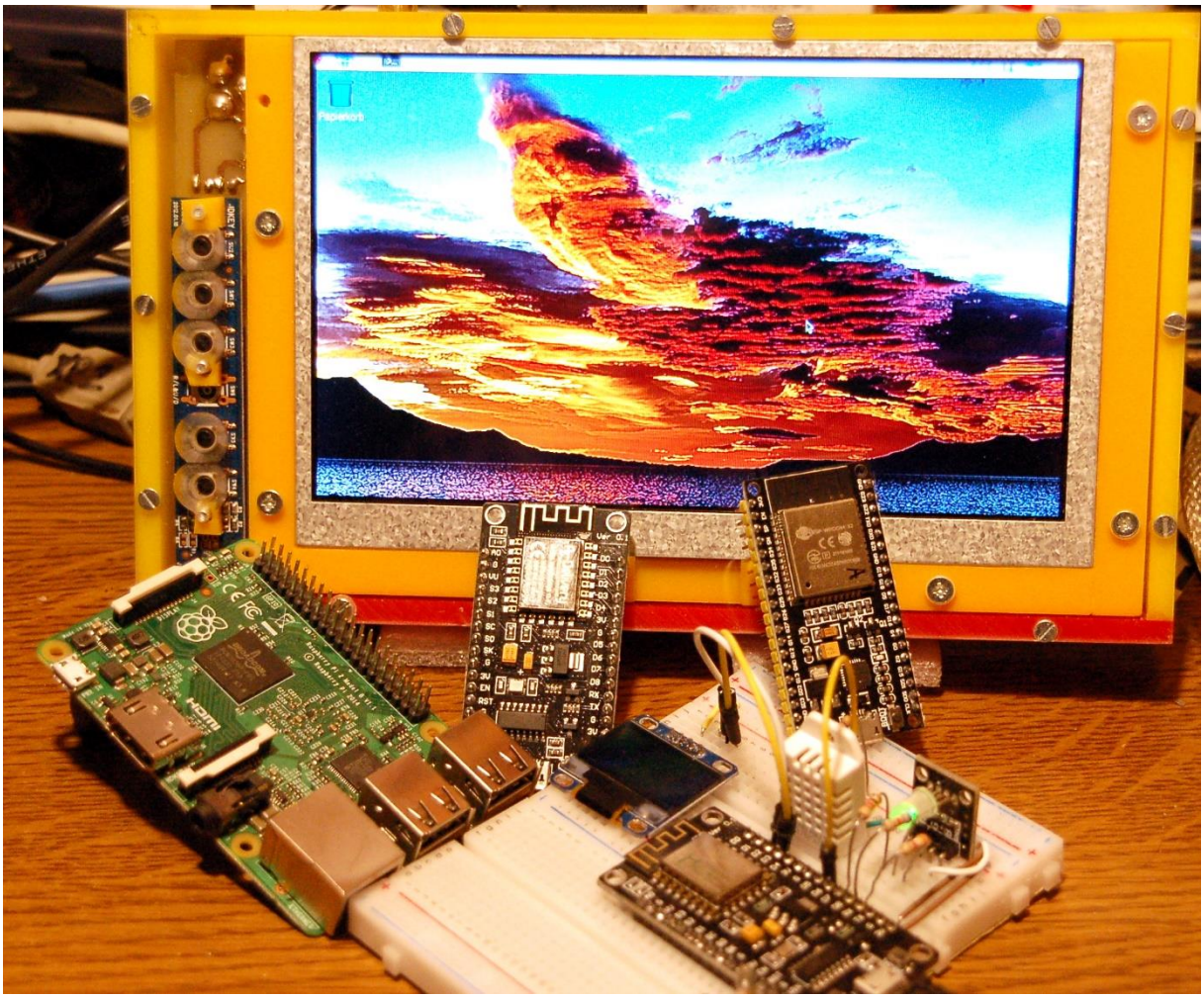

Abbildung 1: MQTT mit Raspi und ESP-Familie

Server und Clients unter MicroPython auf dem Raspi und der ESP-Familie

1. MQTT - Raspberry Pi – Mosquitto und Node-RED

Nach der Betrachtung von TCP-Webservern und dem UDP-Protokoll in verschiedenen meiner Blogposts (Frostwächter im Treibhaus 1 + 2; Geigerzähler, Mückenscheuche), nehmen wir heute den Datenaustausch via MQTT unter die Lupe. Es vereint in gewisser Weise die Vorteile des Transfers via UDP mit der Datenintegrität einer TCP-Verbindung. Letzteres Protokoll stellt die Basis für das „Message Queuing Telemetry Transport“-Protokoll dar. Das heißt, MQTT arbeitet mit einer gesicherten Verbindung. Andererseits ist der Rahmen für die Übertragung von Messdaten zu einem zentralen Server, der in diesem Zusammenhang Broker heißt, ähnlich einfach wie unter UDP. Jeder Teilnehmer kann nach Belieben senden und empfangen. Stationen, die Daten zum Broker senden, werden Publisher genannt. Clients, die umgekehrt Daten vom Broker abrufen, nennt man Subscriber oder Abonnenten. Wie bei UDP kann man beide Vorgänge auch leicht auf einer Station vereinen. Damit nichts durcheinanderkommt, gibt es Themenbereiche, unter denen man Daten veröffentlichen, oder abrufen kann, die sogenannten Topics. Aber anders als bei UDP können sich nicht zwei beliebige Netzwerkgeräte direkt miteinander

unterhalten, sondern alle Transfers laufen stets über den Broker, also den Server ab. Er sammelt Daten und verteilt sie wieder. Die Topics werden aber nicht auf dem Server festgelegt, sondern durch die Clients. Ein Thema existiert in dem Moment, in dem ein Client eine Nachricht unter diesem Thema veröffentlicht. Das macht das System flexibel und pflegeleicht.

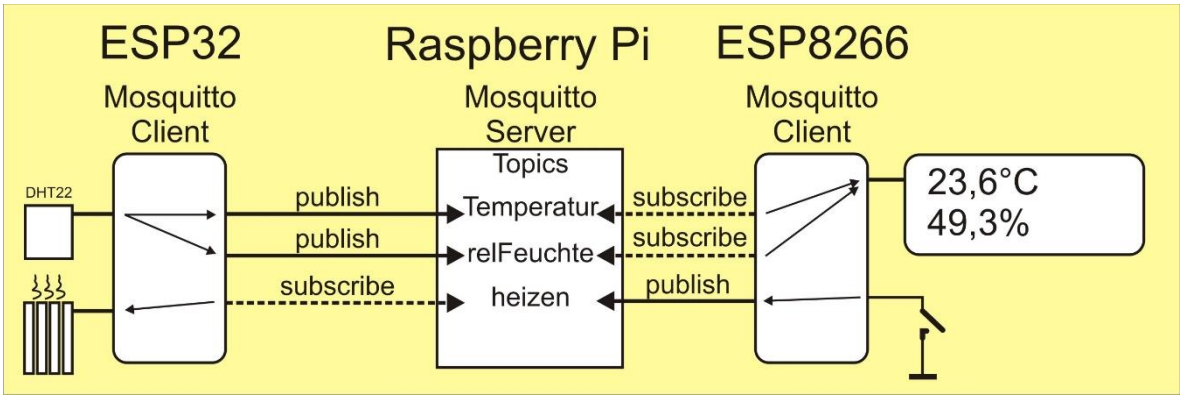

Abbildung 2: MQTT - Arbeitsweise

Als Clients kommen Messwertaufnehmer oder Anzeigeeinheiten mit einem ESP32 oder ESP8266 zur Anwendung. In Abbildung 1 ist ein typisches Scenario dargestellt. Der ESP32 stellt zum Beispiel Temperatur- und Feuchtewerte von einem DHT22 zur Verfügung und erhält vom Broker die Information, ob die Heizung einzuschalten ist. Der ESP8266 hat die Nachrichten zur Temperatur und der Luftfeuchte abonniert und erhält vom Broker die neuen Werte, sobald diese vorliegen. Ferner sendet der Controller den Schalterzustand an den Broker unter dem Topic heizen.

Eine LAN- oder WLAN-Verbindung ist Grundvoraussetzung. Nackte Arduinos mit AVR-Controller scheiden also in diesem Themenkreis alleine schon aus dem Grund aus, weil sie keine Funkverbindung bieten können und in diesem Themenkreis auch nicht unter MicroPython programmierbar sind. Die Vielfalt an Sensoren, die direkt über ADC, I2C, UART oder SPI an einem ESP andocken können, macht die ESP-Familie zusammen mit den Modulen auf der Basis von MicroPython als Clients für MQTT bestens geeignet.

Als Broker sind ESPs allerdings zu schmalbrüstig, da muss mindestens ein Raspberry Pi herhalten. Wir setzen als Broker die freie Software Mosquitto ein. Sie lässt sich ohne Probleme auf Linux-Maschinen, also auch auf einem Raspberry Pi installieren. Der Mosquitto-Server sammelt die eingehenden Nachrichten, die von Clients (Publishern) unter einem Topic veröffentlicht werden und verteilt sie auf Anfrage an die Clients, welche diese Nachrichtenthemen abonniert haben (Subscriber). Die Unterschiede in der Art der Verbindung zeigen die folgenden Grafiken auf.

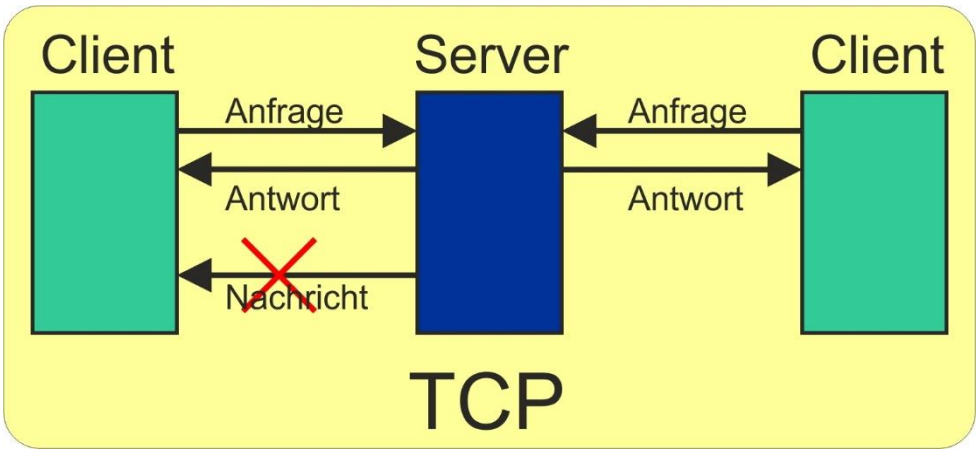

Abbildung 3: TCP - Einzelverbingen keine ad hoc-Nachrichten vom Server

TCP baut auf gesicherte Einzelverbindungen. HTTP-Clients können vom Server nur nach Anfrage kontaktiert werden. Die Datenintegrität ist gewährleistet. Monitoring des Datenverkehrs ist nicht möglich.

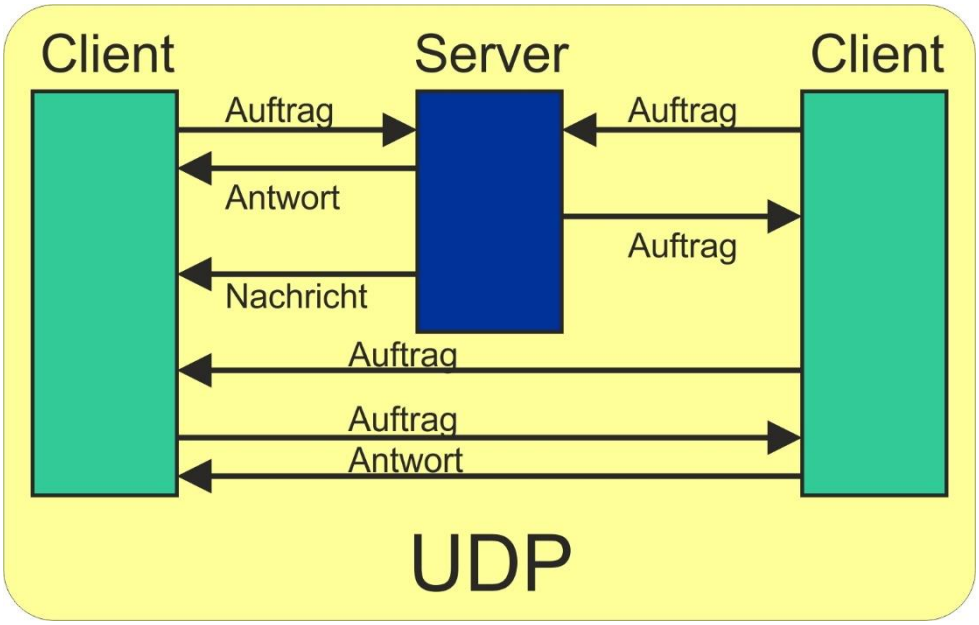

Abbildung 4: UDP - Mehrfach-Verbindungen möglich

Unter UDP sind Mehrfachkontakte zwischen den Teilnehmern möglich. Jede Station kann mit jeder anderen spontan Daten austauschen. Der Versand kann sowohl mit Broadcast an alle Teilnehmer erfolgen als auch an gezielt adressierte Einzelstationen. Die Datenintegrität ist nicht sichergestellt. Monitoring des Datenverkehrs ist nur möglich, wenn entsprechende Sendebefehle im Programm der Sendestation enthalten sind.

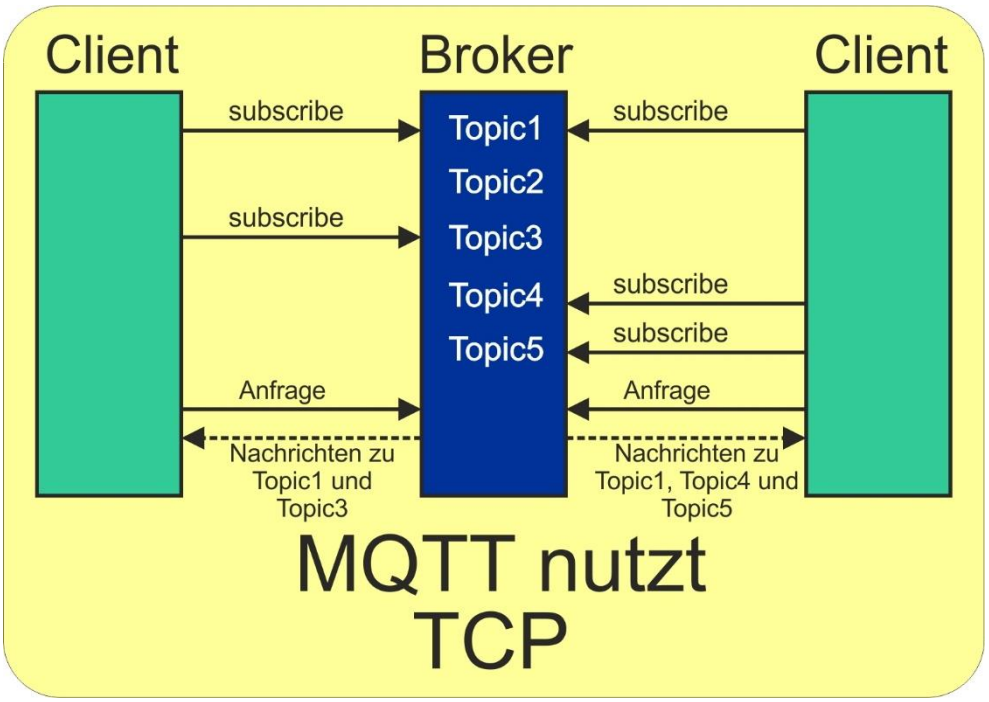

Abbildung 5: MQTT - Gezielter Nachrichtenversand nach Anfrage

MQTT setzt auf TCP auf und bietet daher gesicherte Verbindungen. Die Datenintegrität ist gewährleistet. Jeder Client kann Topics auf dem Broker einrichten, die andere Clients beliebig abonnieren können. Ist ein Topic abonniert, dann erhält der Client die neuesten Daten zu diesem Thema auf eine Anfrage beim Broker von diesem als Antwort zugestellt. Nicht abonnierte Themen werden vom Broker an den anfragenden Client nicht bedient. Monitoring des Datenverkehrs ist sehr einfach möglich, indem ein Lauscher die entsprechenden Topics abonniert.

Damit wir den Verkehr mithören können, benötigen wir also noch ein Tool, mit dem wir uns, auch als Subscriber, am Brocker anmelden können. Es ist ebenfalls ohne weiteres möglich, dass mehrere Clients dasselbe Topic abonnieren. Das ist ein Vorteil von MQTT im Vergleich zu den anderen Nachrichtenprotokollen. Wir werden diese Möglichkeit zunächst im Terminal auf dem Raspi nutzen. Später werden wir die ebenfalls freie Software Node-RED für diesen Zweck einsetzen. Das Programm, das auch auf dem Raspi installiert wird, verfügt über eine grafische Oberfläche, mit deren Hilfe auf einfache Weise browsertaugliche Schnittstellen gebaut werden können, welche die Messergebnisse von den ESP-Messknechten grafisch darzustellen erlauben. Node-RED ist standardmäßig im Installationsumfang des Betriebssystems Raspian bereits enthalten und wird in der Regel über den Softwarepool installiert. Es geht aber auch händisch über die Kommandozeile.

Fassen wir die Hard- und Software für unser Projekt zusammen.

Hardware:

1	Raspberry Pi Bundle mit Pi 1 B+, Pi Pico und Zubehör oder
1	Raspberry Pi (1B+, 2, 2B, 3, 3B) ff. als Raspi bezeichnet
1	SD-Karte 8-16GB passend zum ausgewählten Raspi
1	Breadboard
diverse	Jumperkabel
1	Raspi Netzteil, 5V, 2,5A
1	Monitor (nur zur Einrichtung des Raspi)
1	Tastatur (nur zur Einrichtung des Raspi)
1	Maus (nur zur Einrichtung des Raspi)
1	HDMI-Kabel (nur zur Einrichtung des Raspi)
1	Netzwerkkabel

Software

Software für ESP32/8266:
Thonny oder
µPyCraft

Firmware:
MicroPython passend zum ESP32 oder ESP8266

MicroPython Module und Programmfiles:
umqttsimple.py
client.py

Software für den Raspi:
Raspian-Image
Imager.exe Brennprogramm für die SD-Card
Mosquitto-Broker
Node-RED

Software für die Windowsmaschine
xming
putty

MicroPython - Sprache - Module und Programme

Zur Installation von Thonny finden Sie hier eine ausführliche Anleitung. Darin gibt es auch eine Beschreibung, wie die MicropythonFirmware (Stand 15.12.2021) auf den ESP-Chip gebrannt wird.

MicroPython ist eine Interpretersprache. Der Hauptunterschied zur Arduino-IDE, wo Sie stets und ausschließlich ganze Programme flashen, ist der, dass Sie die MicroPython-Firmware nur einmal zu Beginn auf den ESP32 flashen müssen, bevor der Controller MicroPython-Anweisungen versteht. Sie können dazu Thonny, µPyCraft oder esptool.py benutzen. Für Thonny habe ich den Vorgang hier beschrieben.

Sobald die Firmware geflasht ist, können Sie sich zwanglos mit Ihrem Controller im Zwiegespräch unterhalten, einzelne Befehle testen und sofort die Antwort sehen, ohne vorher ein ganzes Programm kompilieren und übertragen zu müssen. Genau das stört mich nämlich an der Arduino-IDE. Man spart einfach enorm Zeit, wenn man einfache Tests der Syntax und der Hardware bis hin zum Ausprobieren und Verfeinern von Funktionen und ganzen Programmteilen über die Kommandozeile vorab prüfen kann, bevor man ein Programm daraus strickt. Zu diesem Zweck erstelle ich auch gerne immer wieder kleine Testprogramme. Als eine Art Makro fassen sie wiederkehrende Befehle zusammen. Aus solchen Programmfragmenten entwickeln sich dann mitunter ganze Anwendungen.

Autostart

Soll das Programm autonom mit dem Einschalten des Controllers starten, kopieren Sie den Programmtext in eine neu angelegte Blankodatei. Speichern Sie diese Datei unter boot.py im Workspace ab und laden Sie sie zum ESP-Chip hoch. Beim nächsten Reset oder Einschalten startet das Programm automatisch.

Programme testen

Manuell werden Programme aus dem aktuellen Editorfenster in der Thonny-IDE über die Taste F5 gestartet. Das geht schneller als der Mausklick auf den Startbutton, oder über das Menü **Run**. Lediglich die im Programm verwendeten Module müssen sich im Flash des ESP32 befinden.

Zwischendurch doch mal wieder Arduino-IDE?

Sollten Sie den Controller später wieder zusammen mit der Arduino-IDE verwenden wollen, flashen Sie das Programm einfach in gewohnter Weise. Allerdings hat der ESP32/ESP8266 dann vergessen, dass er jemals MicroPython gesprochen hat. Umgekehrt kann jeder Espressif-Chip, der ein kompiliertes Programm aus der Arduino-IDE oder die AT-Firmware oder LUA oder … enthält, problemlos mit der MicroPython-Firmware versehen werden. Der Vorgang ist immer so, wie hier beschrieben.

Vorbereiten des Raspi

Raspian besorgen

Raspian ist vom Debian-Linux abgeleitet und in verschiedenen Releases verfügbar. Die hier verwendete Version taugt für alle Raspi-Boards der Versionen 1B+, 2, 2B, 2B+, 3 und 3B.

Firmware-Releases mit verschiedenem Umfang werden hier zum Download angeboten https://www.raspberrypi.com/software/operating-systems/#raspberry-pi-os-32-bit. Ich arbeite in diesem Beitrag mit einem Raspberry 2B und einem Image, das von dem Programm **Imager** automatisch heruntergeladen wird und mit wenigen

Klicks sehr einfach zu installieren ist. Benötigt wird dafür eine SD-Karte der Kategorie 10 mit 8GB aufwärts. Auf meiner 8GB-Karte sind nach der Einrichtung weiterer Software jetzt noch 45% verfügbar.

Das Vorbereiten der Karte ist denkbar einfach. Wir laden den Imager herunter und starten die Datei durch Doppelklick.

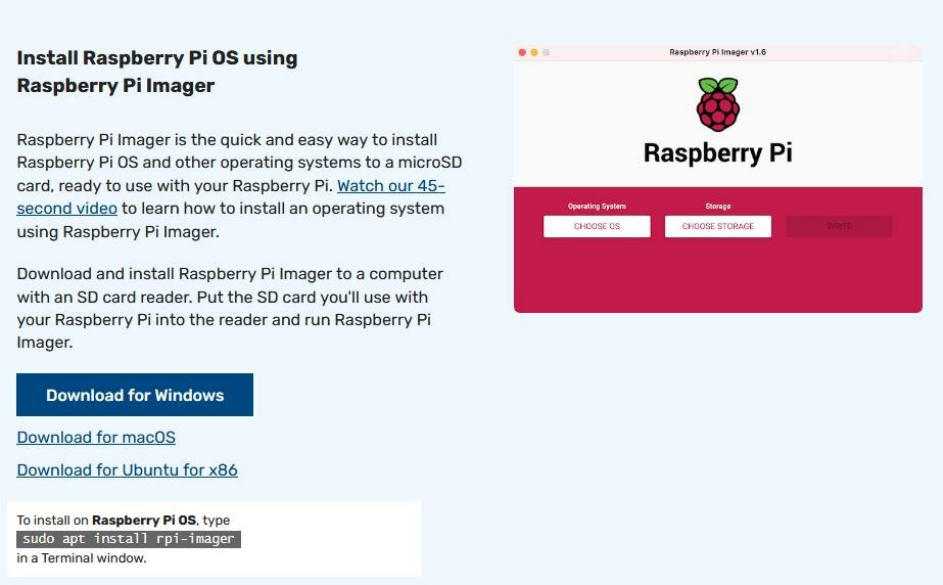

Abbildung 6: Raspberry Imager herunterladen

Abbildung 7: Install Imager

Nach ein paar Sekunden ist die Installation in den Ordner C:\Program Files (x86)\Raspberry Pi Imager abgeschlossen. Wir können die Anwendung auch gleich starten.

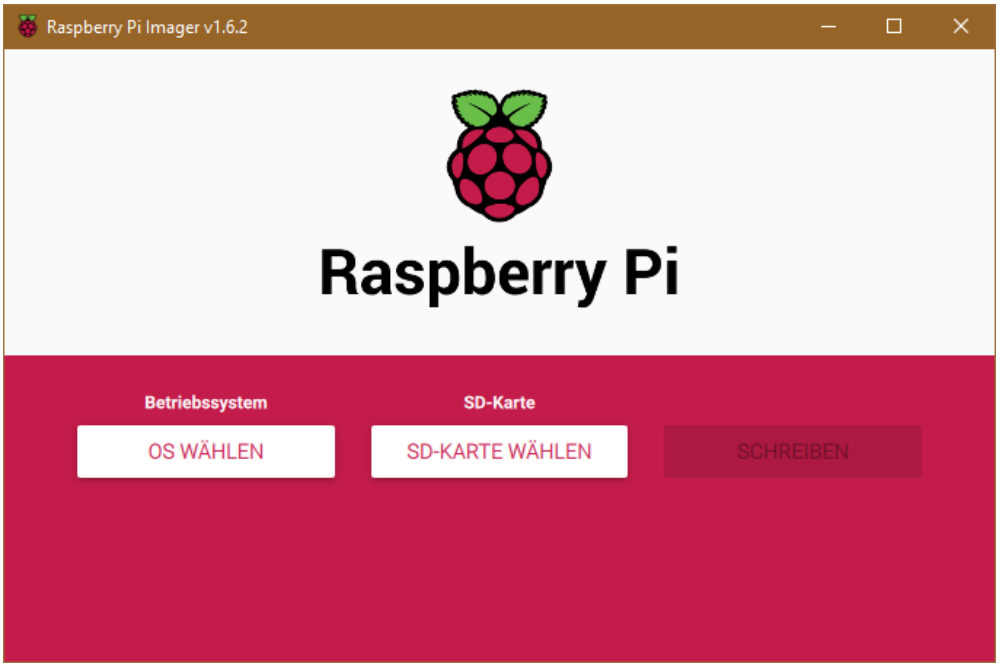

Abbildung 8: Imager starten

Zuerst muss ein Betriebssystem gewählt werden -> **OS WÄHLEN**.
Ich habe das empfohlene System ausgewählt.

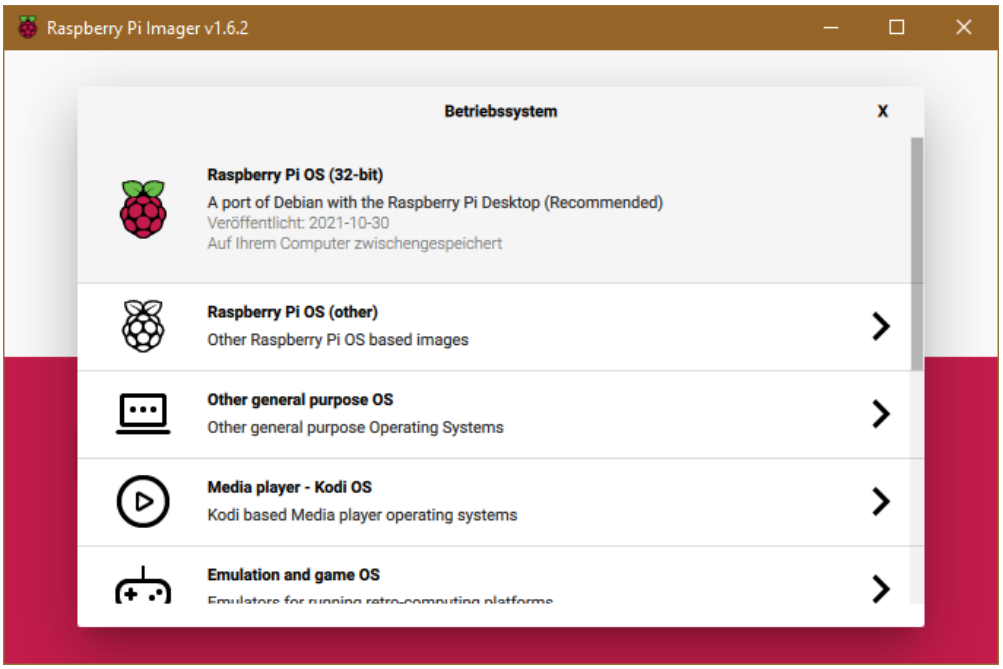

Abbildung 9: Auswahl des Betriebssystems

Die SD-Karte muss sich im Kartenslot befinden und vom System erkannt worden sein. Jetzt kommt der gefährlichste Teil des Ganzen, die Auswahl des Speichermediums. Gefährlich ist das deswegen, weil der Inhalt der Speicherkarte gelöscht wird. Sie müssen sich also ganz sicher sein, dass auf der Karte keine wichtigen Daten gespeichert sind. Die wären nach dem Start des Brennvorgangs unwiederbringlich futsch!

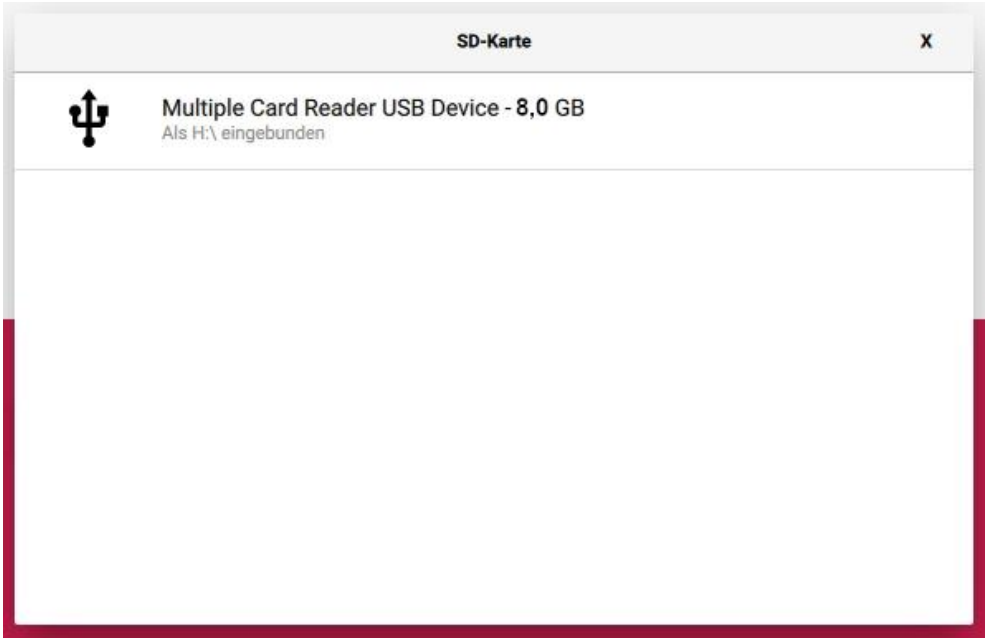

Abbildung 10: Kartenauswahl - hier ganz einfach

Wir wählen die Karte aus und quittieren die Sicherheitsabfrage mit JA. Der Download der Firmware beginnt, und danach startet der Schreibvorgang, der je nach der ausgewählten Firmware gut eine halbe Stunde aufwärts dauern kann.

Abbildung 11: System wurde erfolgreich geschrieben

Als Nächstes erzeugen wir mittels eines Editors eine leere Datei mit dem Namen ssh und kopieren diese auf die Karte. Falls Windows die eben erstellte Karte formatieren möchte, lassen Sie das nicht zu. Nun entnehmen wir die Karte aus dem Leserschacht und setzen sie am Raspi ein.

Für die weiteren Aktionen ist es günstig, vorübergehend eine Tastatur, eine Maus und einen Monitor am Raspi anzuschließen. Ferner brauchen wir eine freie IP-Adresse im lokalen Netzwerk, die wir dem Raspi zuweisen können. Weil wir darauf Server betreiben wollen, ist es besser dem Raspi eine feste IP-Adresse zuzuweisen. Der einfachste Weg dahin führt über den Raspi selbst. Alle weiteren Schritte können wir dann auch über eine SSH-Verbindung von einem anderen Rechner aus erledigen. Verbinden wir den Kleinen also rasch noch mit einem Netzwerkkabel mit dem Router.

Nun starten wir den Raspi. Nach ein paar Einstellungen befinden wir uns auf dem Raspian-Desktop - Next, Set Country -> Germany, Next, Change Password -> ein eigenes eingeben und bestätigen, Next, Update Software, Skip, Done.

Bei meinem Mini-TFT-Display war es jetzt nötig die Bildschirmgröße anzupassen.

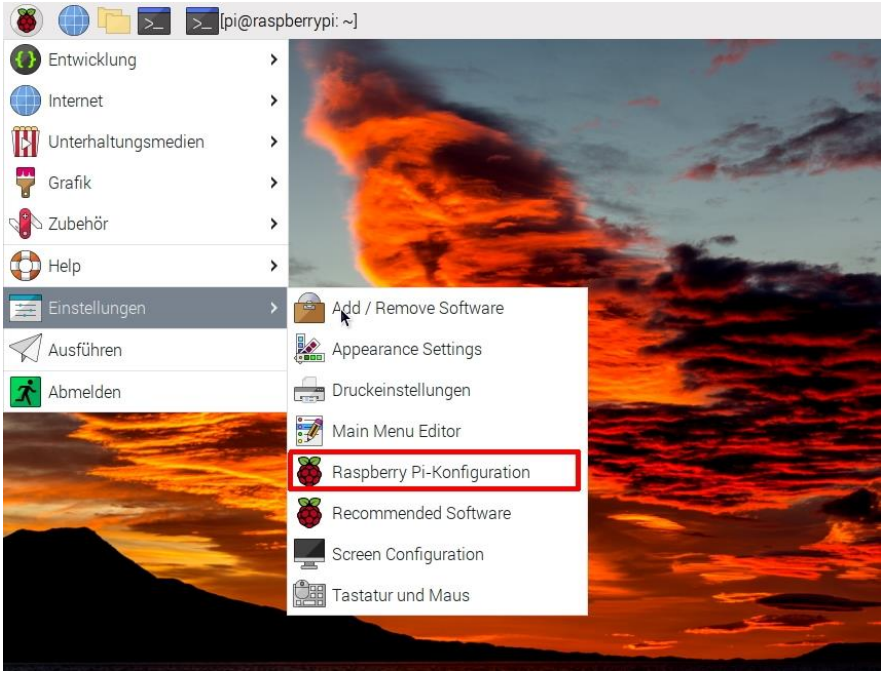

Abbildung 12: Raspi - Einstellungen aufrufen

Über **Headless Resolution** kann man auch direkt die Auflösung einstellen.

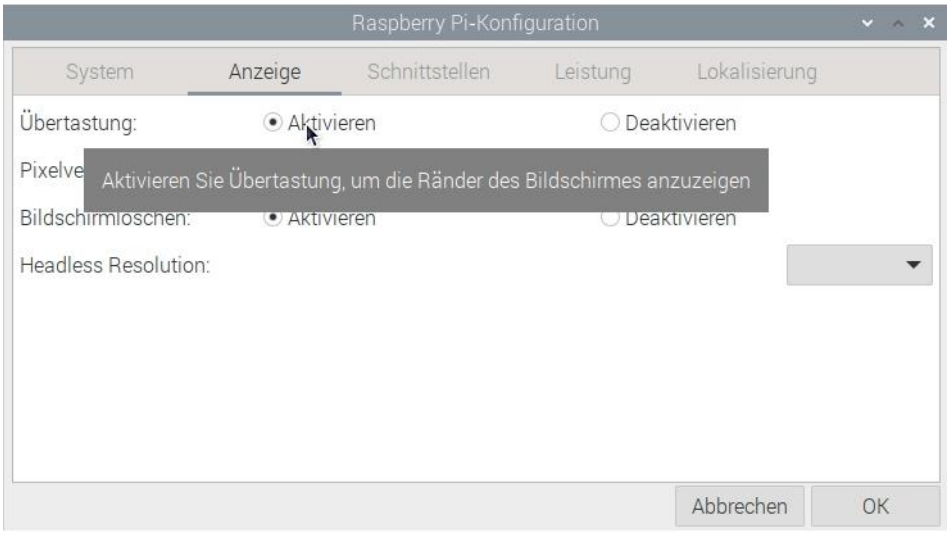

Abbildung 13: Raspi - Bildschirm einstellen

In einem Abwasch geht auch noch die Aktivierung/Kontrolle des SSH-Zugriffs, damit wir vom Netzwerk her auf den Raspi zugreifen können. Bildschirm, Tastatur und Maus am Gerät werden dann obsolet. Aber – noch sind wir nicht so weit.

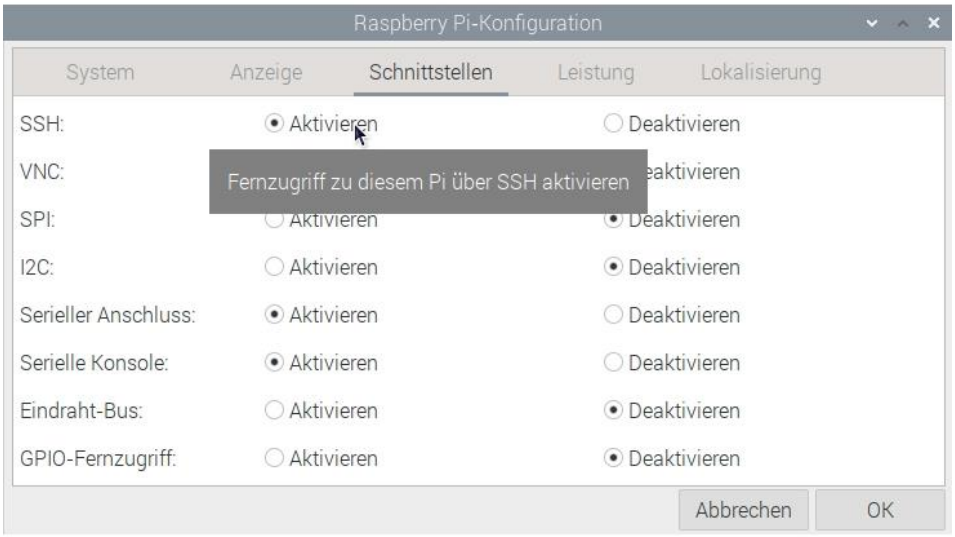

Abbildung 14: Raspi - SSH aktivieren

Die Konfiguration beenden wir mit **OK**.

Das Meiste, was noch zu tun ist, können wir von der Kommandozeile aus erledigen. Dazu öffnen wir ein Terminal mit der Tastenkombination **Strg + Alt + T**. Das Standard-Image enthält den Editor **nano**, den wir gleich mit folgendem Befehl starten.

sudo nano /etc/dhcpcd.conf

Suchen Sie die folgenden Zeilen im unteren Teil der Datei **dhcpcd.conf**. Entfernen Sie die "#" am Zeilenbeginn und setzen Sie die Werte gemäß ihrem Heimnetzwerk.

Example static IP configuration:
#interface eth0
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1

Diese Stelle ändern wir ab in:

 # Example static IP configuration:
interface eth0
static ip_address=10.0.1.99/24
static routers=10.0.1.25
static domain_name_servers=10.0.1.25

Mit **Strg + O** speichern Sie die Änderung ab und mit **Strg + X** beenden Sie den Editor. Starten Sie den Raspi jetzt neu.

Nach der Anmeldung öffnen wir wieder ein Terminal und prüfen mit dem folgenden Befehl, ob unsere Änderungen gefruchtet haben.

ifconfig

Die Ausgabe sollte etwa so aussehen:

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet **10.0.1.99 netmask 255.255.255.0 broadcast 10.0.1.255**
        inet6 fe80::9f0d:6726:b163:74c0 prefixlen 64 scopeid 0x20<link>
        inet6 2003:f3:7722:6200:d933:6671:26db:57ec prefixlen 64 scopeid 0x0<global>

Wir überprüfen auch sofort, ob die Netzverbindung ins Internet funktioniert. Ping an eine externe URL oder IP klärt uns darüber auf.

pi@raspberrypi:~ $ **ping regensburg.de**
PING regensburg.de (62.116.156.60) 56(84) bytes of data.
64 bytes from 60-156-116-62.rev.customer-net.de (62.116.156.60): icmp_seq=1 ttl=59 time=15.8 ms
64 bytes from 60-156-116-62.rev.customer-net.de (62.116.156.60): icmp_seq=2 ttl=59 time=15.5 ms
64 bytes from 60-156-116-62.rev.customer-net.de (62.116.156.60): icmp_seq=3 ttl=59 time=15.3 ms

Abbruch mit **Strg + C,** und wenn wir schon gerade dabei sind, testen wir auch gleich noch den SSH-Zugang. Im Terminal geben wir folgenden Befehl ein.

ssh localhost

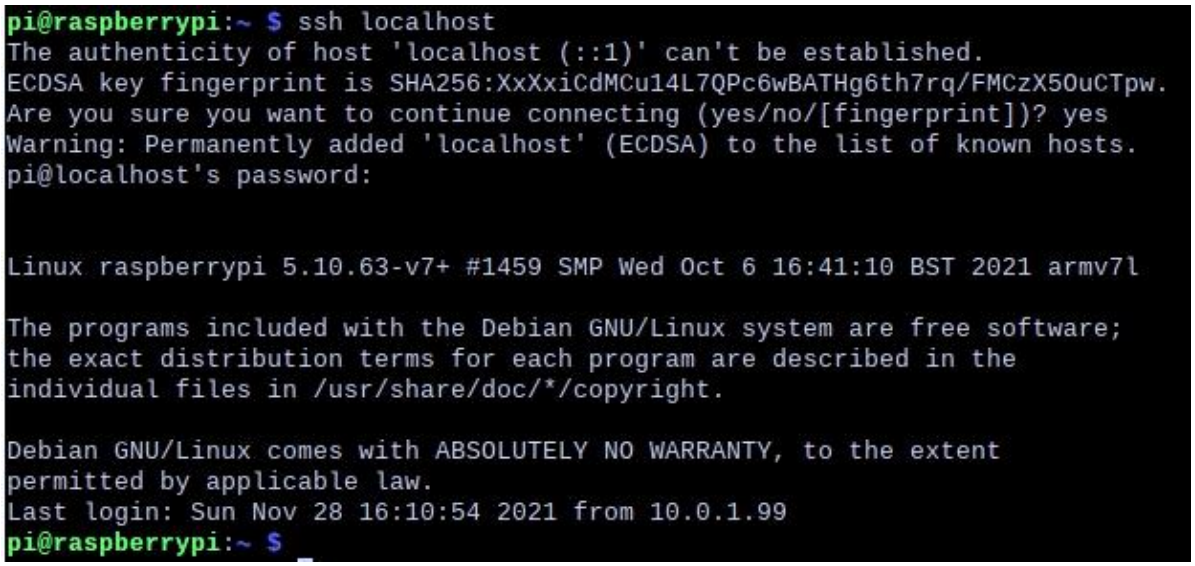

Abbildung 15: Raspi - SSH-Test

Wir müssen den "Angriff" mit "yes" bestätigen und uns dann mit dem Passwort, das wir oben vergeben haben anmelden. Wir sind jetzt praktisch von unserem System durch den Hausierereingang in unser System eingedrungen, ohne es vorher verlassen zu haben. OK - so etwas geht nur auf einem PC. Wenn das aber geklappt hat, dann geht das auch von einem anderen Rechner aus.

Mosquitto installieren

Die letzte Aktion, die wir von der Kommandozeile aus durchführen, ist die Installation des Mosquitto Brokers.

sudo apt-get update

Eine Menge an Paketen wird geladen und installiert, das dauert ein paar Minuten.

sudo apt install -y mosquitto mosquitto-clients

Wieder wird eine Reihe von Paketen geholt, entpackt und eingerichtet. Dann sorgen wir dafür, dass Mosquitto jedes Mal beim Booten des Systems gleich mit gestartet wird.

sudo systemctl enable mosquitto.service

Der folgende Befehl verrät uns die Version des Programms und, dass der mosquitto-Server auf dem Port 1883 lauscht.

mosquitto -v
1638040611: **mosquitto version 2.0.11 starting**
1638040611: Using default config.
1638040611: Starting in local only mode. Connections will only be possible from clients running on this machine.

1638040611: Create a configuration file which defines a listener to allow remote access.
1638040611: For more details see https://mosquitto.org/documentation/authentication-methods/
1638040611: **Opening ipv4 listen socket on port 1883**.
1638040611: Error: Address already in use
1638040611: Opening ipv6 listen socket on port 1883.

Die Fehlermeldung, dass die Adresse bereits benutzt wird (Error: Address already in use) können wir getrost ignorieren, weil sie uns nur sagt, dass Mosquitto bereits läuft.

Als Vorbereitung für den nächsten Blogpost sollten wir aber noch eine Kleinigkeit an der Konfiguration des Mosquitto-Deamons ändern. Würden wir jetzt versuchen, von einem anderen Netzwerkgerät, etwa unserem ESP32 auf den Broker zuzugreifen, dann bekämen wir eine Abweisung. Das liegt daran, dass anonyme Zugriffe, also ohne Benutzerauthentifikation und Passwort, standardmäßig verboten sind. Das wollen wir, zumindest in der Testphase ändern.

Wechseln wir also auf dem Raspi in das Verzeichnis **/etc/mosquitto/conf.d** und legen wir dort eine Datei an mit dem Namen **anonymous.conf**.

cd /etc/mosquitto/conf.d
sudo nano anonymous.conf

Nun geben wir die folgenden beiden Zeilen ein, speichern die Datei ab und beenden nano.

listener 1883
allow_anonymous true

Dann starten wir Mosquitto neu

systemctl restart mosquitto

Hintergrund:
Jede Datei im Verzeichnis **/etc/mosquitto/conf.d** mit der Endung **.conf** wird als Konfigurationsdatei aufgefasst und beim Start von mosquitto über die Datei **/etc/mosquitto/mosquitto.conf** aufgerufen. Hier können wir also unsere Erweiterungen zur Konfiguration unterbringen.

Installation von Node-RED

Auf dem Raspi fehlt jetzt nur noch das Paket Node-RED. Starten wir die Installation. Ich habe wieder den Weg über die Kommandozeile gewählt, da weiß man, was man tut.

sudo apt install nodered

Die Installation dauert wieder etwas länger. Schließlich meldet sich der Kommandopromt wieder, und wir können dafür sorgen, dass auch der Node-RED-Deamon mit dem Start des Systems hochfährt.

sudo systemctl enable nodered.service

Ausgabe:
Created symlink /etc/systemd/system/multi-user.target.wants/nodered.service → /lib/systemd/system/nodered.service.

Damit ist die Installationssession am Raspi beendet. Wir starten den Kleinen neu durch. Nachdem nun all das, was auf dem Raspi laufen soll, seinen Dienst tut und korrekt arbeitet, können wir beruhigt Tastatur, Monitor und Maus vom Raspi abziehen.

Software für die Windowskiste

Während dessen holen wir uns ein Terminalprogramm auf den Windows-Rechner mit dem wir über das Netzwerk Kontakt mit dem Raspi aufnehmen können. Weit verbreitet ist putty. Wir laden die Datei für unser System herunter. Die exe-Datei ist ohne Installation lauffähig. Wir speichern sie in einem beliebigen Verzeichnis und legen uns einen Link auf den Desktop. Mit ein paar Klicks ist das Terminalprogramm nach dem ersten Start eingerichtet. Unter **Connection - X11** schalten wir **X11 forwarding** ein.

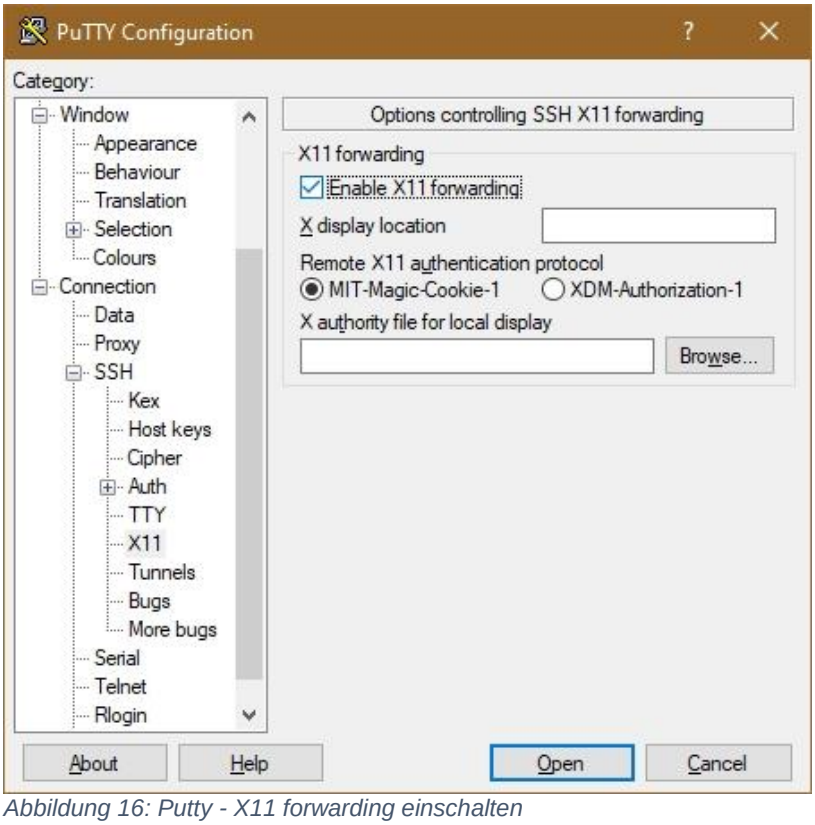

Abbildung 16: Putty - X11 forwarding einschalten

Unter Session geben wir die IP unseres Raspis ein, der Port bleibt auf 22 stehen. Dann geben wir der Verbindung einen sprechenden Namen und speichern die Konfiguration ab.

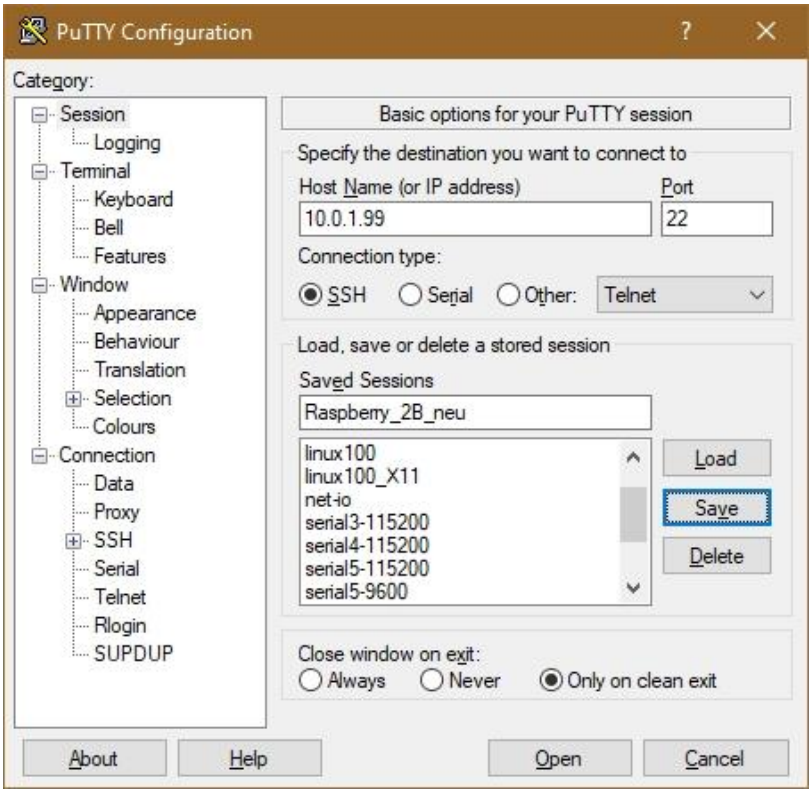

Abbildung 17: Putty - Einrichtung sichern

Mit Open öffnen wir ein Terminal zum Raspi. Es sieht ähnlich aus wie das, mit dem wir schon am Raspi gearbeitet haben. Lediglich die Menüzeile fehlt.

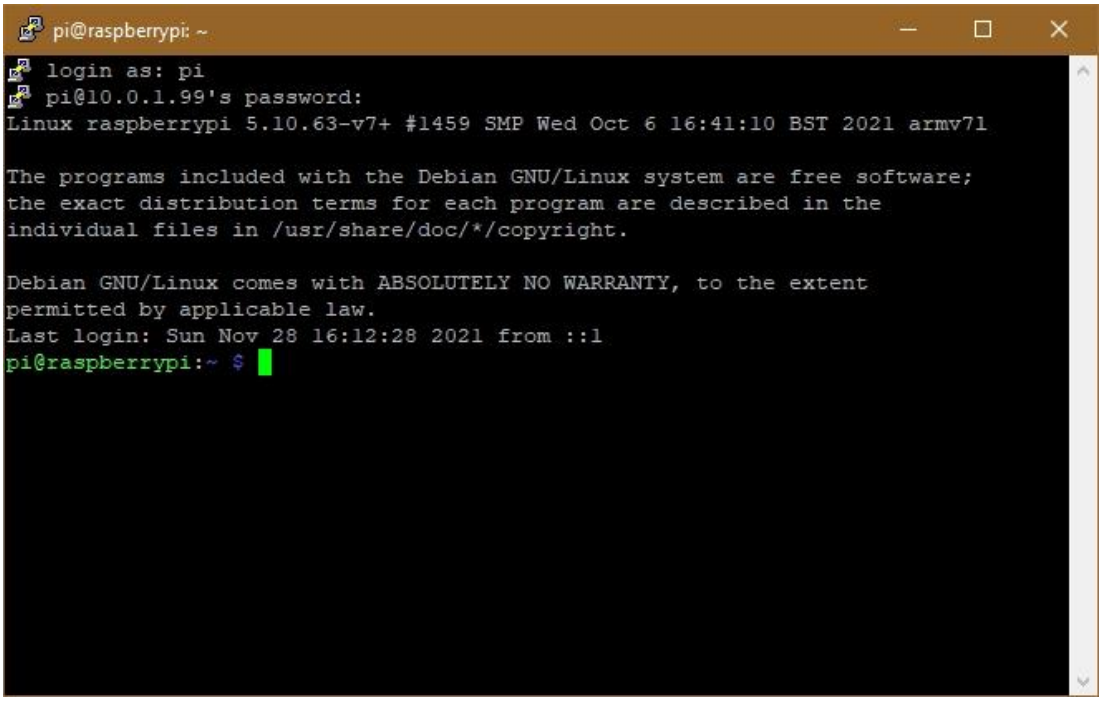

Abbildung 18: Putty - Terminalfenster

Testen wir einmal den Mosquitto und schauen nach, ob er auch sticht. Zu diesem Zweck brauchen wir zwei Terminalfenster, von denen jedes einen Mosquitto-Client emuliert. Öffnen wir also ein zweites Terminal.

Im ersten Terminal geben wir folgenden Befehl ein, um als Subscriber zu fungieren. Dieses Fenster soll also Nachrichten vom Brocker empfangen.

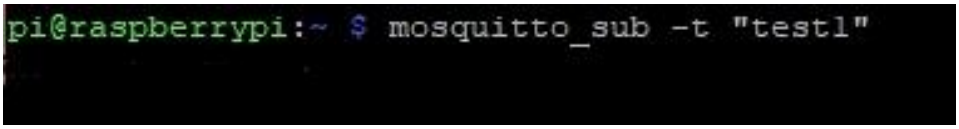

Abbildung 19: Mosquitto - Subscriber lauscht

Im zweiten Fenster lassen wir von einem Publisher eine Nachricht an den Broker senden.

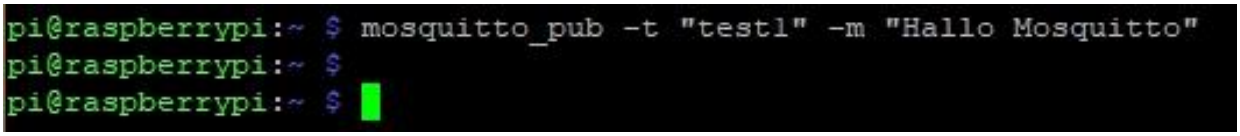

Abbildung 20: Mosquitto – Publisher – Nachricht wird gesendet

Mit dem Absenden der Nachricht erscheint diese im Fenster des Subscribers.

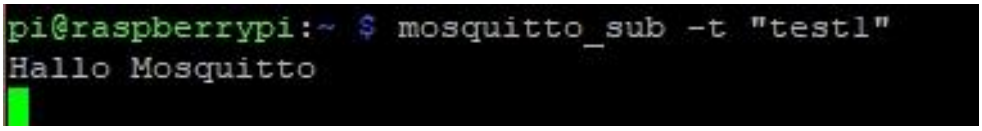

Abbildung 21: Mosquitto - Subscriber - Nachricht ist angekommen

Während Mosquitto über die Kommandozeile angesprochen wird, meldet sich Node-RED als Weboberfläche in einem Browser. Wir testen auch diese Installation. Node-RED sollte ja beim letzten Bootvorgang automatisch mit gestartet worden sein. So sollte das in etwa aussehen, wenn wir Node-RED mit der IP unseres Raspis und dem **Port 1880** aufrufen.

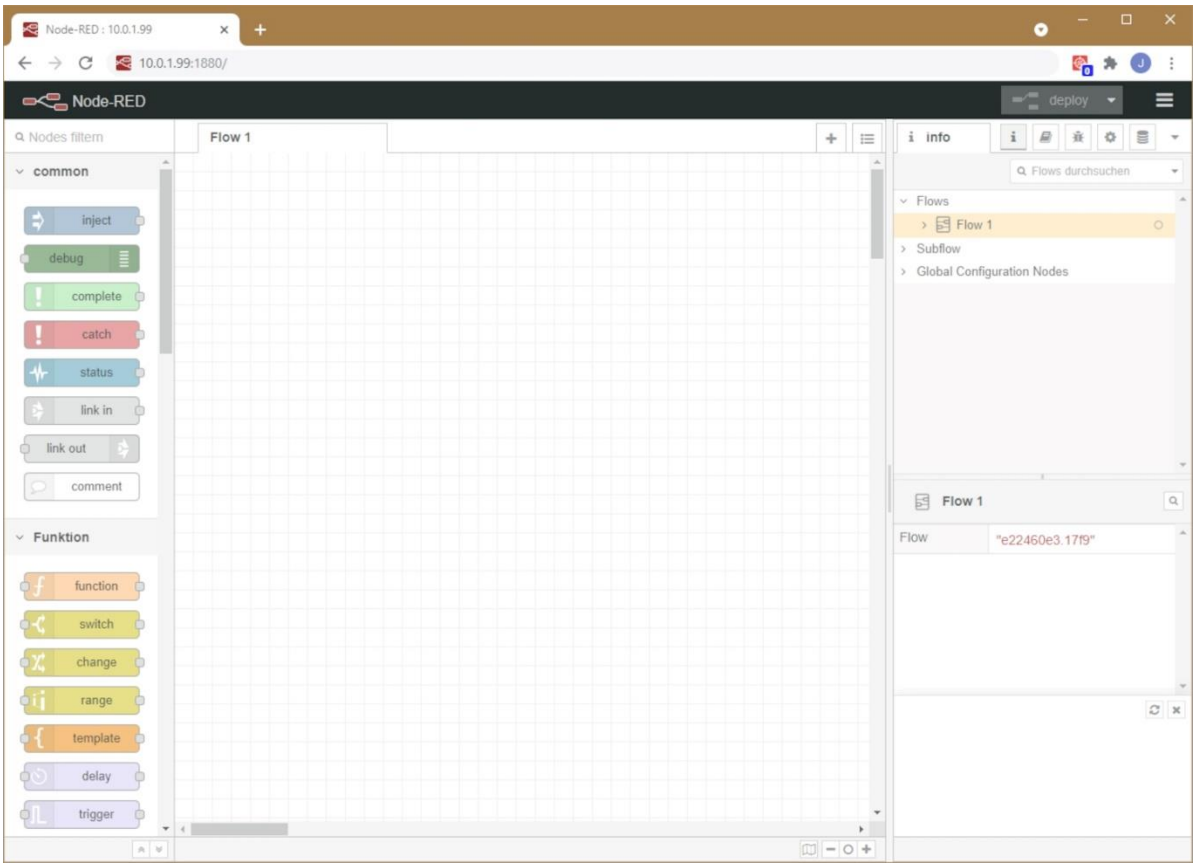

Abbildung 22: Node-RED am PC in Chrome

Jetzt fehlt für heute nur noch eine Kleinigkeit. Der Raspi hat ja nun keinen Bildschirm mehr. Um auf der Kommandozeile zu arbeiten, können wir uns über Putty einloggen. Aber was machen wir, wenn wir mit der grafischen Oberfläche des Raspi etwas machen wollen oder müssen. Wir haben bei der Einrichtung von Putty die X11-Weiterleitung aktiviert. Die können wir nutzen, wenn wir auf der Windowskiste einen X11-Server installieren. Das zuständige Tool heißt xming und kann vom Sourceforge-Net heruntergeladen werden. Wir starten das Setup mit Doppelklick, ignorieren die Meldung von Windows und folgen dann dem Assistenten mit "Next".

Abbildung 23: xming - Setup

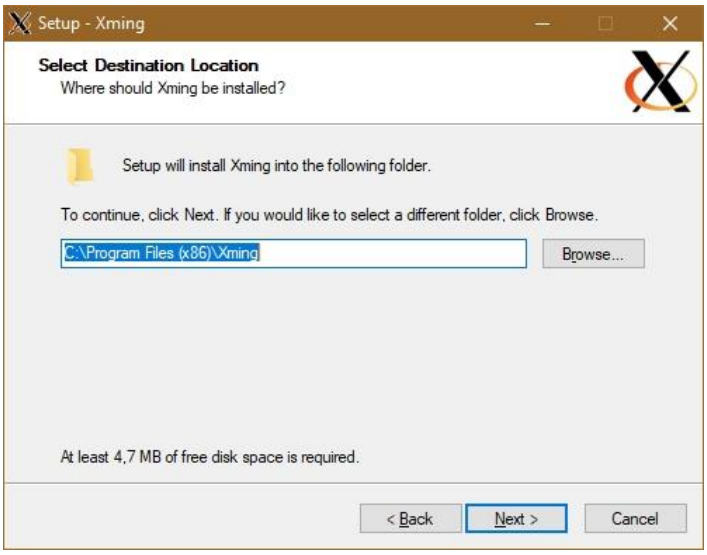

Abbildung 24: xming - Zielort

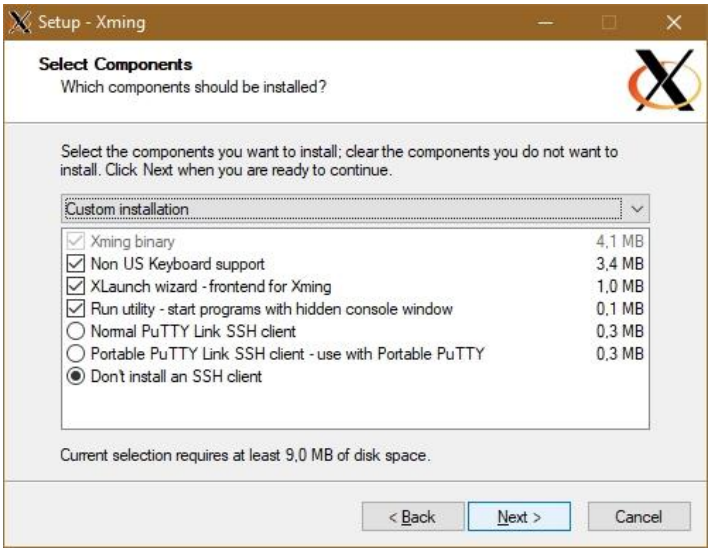

Abbildung 25: xming - Komonentenauswahl

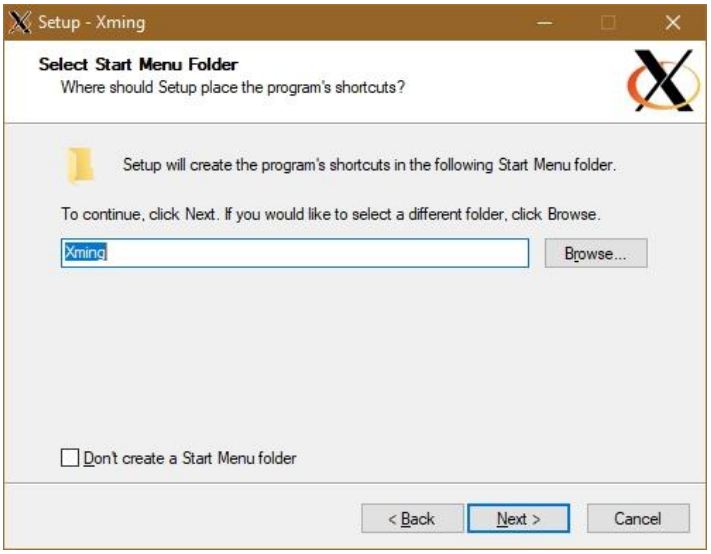

Abbildung 26: xming - Start-Ordner auswählen

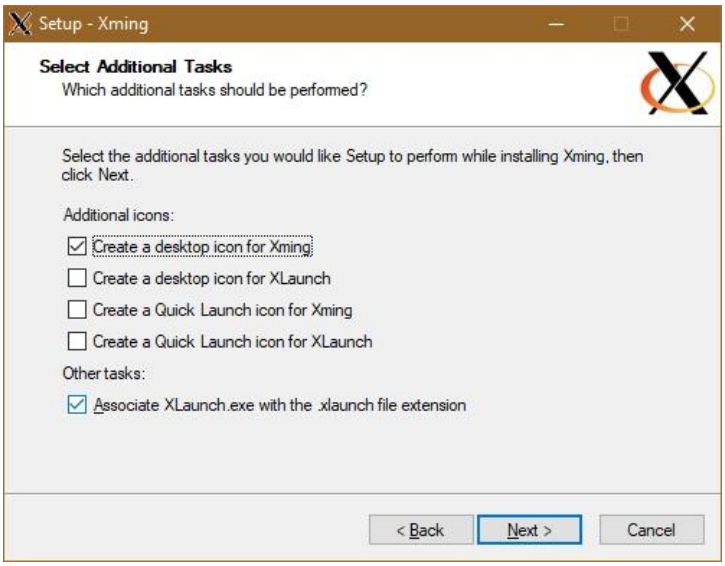

Abbildung 27: xming - Verknüpfungen festlegen

Abbildung 28: xming - Fertigstellung

Vom Windows Desktop aus starten wir über einen Rechtsklick auf das xming-Icon das Programm **als Administrator**. Wir müssen die Änderungen am System erlauben, dann zieht sich xming in den Hintergrund zurück. Wenn wir nun ein Putty Terminal öffnen, und den folgenden Befehl auf die Kommandozeile schreiben, dann öffnet sich auf dem Windowsbildschirm nach einigen Sekunden die Thonny-IDE von unserem Raspi.

thonny & disown

Durch das Anhängsel "& disown" können wir das Terminal für andere Zwecke weiterverwenden. Ohne den Zusatz wäre das Terminal blockiert.

Wir können auch in Erfahrung bringen, ob Python auf dem Raspi läuft.

python3

Ausgabe:

```
Python 3.9.2 (default, Mar 12 2021, 04:06:34)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Siehe da, es ist bereits die neueste Version 3.9, die uns hier freundlich mit dem Pythonprompt begrüßt. Es muss allerdings gesagt werden, dass es sich nicht um MicroPython sondern um die ausgewachsene Version CPython handelt. Es gibt daher eine Reihe von Unterschieden im Vergleich zum ESP32/ESP8266, was den Sprachumfang aber auch die Hardware angeht.

Die Grundlagen für unser MQTT-Projekt sind hiermit geschaffen. Im nächsten Beitrag werden wir einen ESP32 / ESP8266 als MQTT-Client einrichten. Durch die phantastischen Eigenschaften von Node-RED wird es dann später ein Leichtes, ansprechende Anwendungen für die Überwachung von Messstellen zu kreieren.

Dieser Blogpost ist auch als PDF-Dokument verfügbar.