

Universität der Bundeswehr München
ETTI 2: Verteilte Intelligente Systeme
Prof. Dr. Antje Gieraths
in Zusammenarbeit mit
WIWeB GF250

Frühjahrssemester 2022

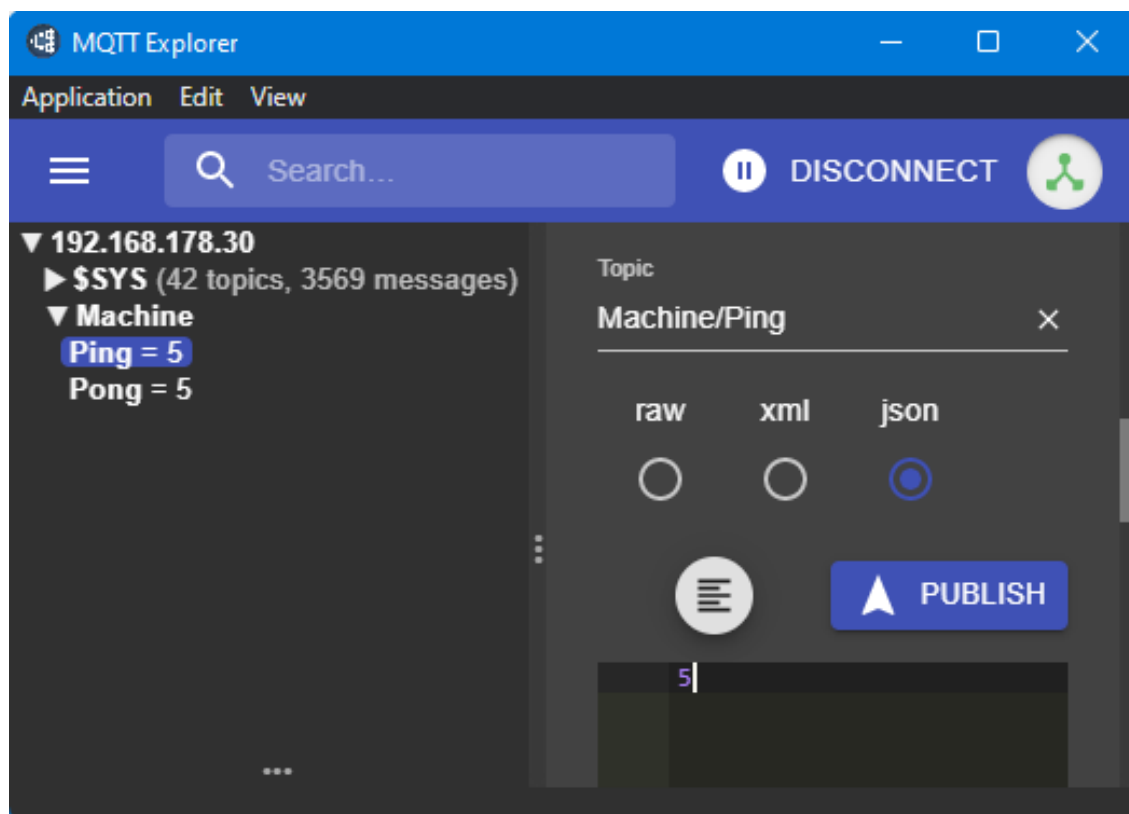
Verfasser: Karl Scholz

Internet of Things Praktikum

Teil 1

MQTT-Broker Einrichten und mit Python Benutzen

25.05.2022



Arbeitsanweisung & Praktikumsablauf

WICHTIG

Um Ihnen während des Praktikums möglichst viel fachliches Wissen vermitteln zu können, bitten wir Sie folgende Punkte, wie zum Beispiel die Einrichtung der Softwares, als Vorbereitung vor dem eigentlichen Praktikumstermin durchzuführen.

WICHTIG

Vor dem Praktikumstermin:

Arbeiten Sie im Vorfeld des Praktikums die Abschnitte bis einschließlich Punkt 5 *Visual Studio Code mit Extensions installieren* zuhause durch.

Dabei handelt es sich bis auf *Punkt 1 Theorie MQTT (Wiederholung aus der Vorlesung)* um Einrichtung der benötigten Software auf ihrem Rechner. Dies müssen Sie aktiv für ihr Betriebssystem auführen, damit Sie bei ihrem praktikumstermin sofort loslegen können.

Während des Praktikumstermins

Alle anderen Punkte.

Sollten sich bei der Vorbereitung Fragen ergeben, können Sie diese gerne im Vorfeld adressieren und per Mail an wiwebgf250@bundeswehr.org schicken. Auch zu Beginn des Praktikumstermins wird es die Möglichkeit geben einzelne Fragen zu stellen. Beachten Sie jedoch, dass das Praktikum eine gewissenhafte Vorbereitung voraussetzt.

Inhaltsverzeichnis

Inhalt

Arbeitsanweisung & Praktikumsablauf	2
Inhaltsverzeichnis	3
Abkürzungsverzeichnis	4
1 Theorie MQTT (Wiederholung aus der Vorlesung)	5
2 Mosquitto MQTT-Broker installieren	6
2.1 Mosquitto MQTT-Broker installieren [Windows]	6
2.1.1 Firewall konfigurieren.....	6
2.1.2 Broker installieren	8
2.2 Mosquitto MQTT-Broker installieren [Linux]	10
2.3 Mosquitto MQTT-Broker installieren [MAC].....	12
3 MQTT-Explorer	14
3.1 Installation.....	14
3.2 Benutzung	15
4 Python installieren.....	16
5 Visual Studio Code mit Extensions installieren	17
6 Pong Back Programm schreiben.....	19
6.1 Bibliothek einbinden	19
6.2 Callback Funktion definieren	19
6.3 Client Objekt einrichten	21
6.4 Programm ausführen und testen	23
6.5 Zusatzaufgabe: Inkrement.....	25
Abbildungsverzeichnis.....	27

Abkürzungsverzeichnis

IDE	Integrated Development Environment
IPv4	Internet Protocol Version 4
MQTT	Message Queuing Telemetry Transport
PIP	Package Installer for Python
VSCode	Visual Studio Code

1 Theorie MQTT (Wiederholung aus der Vorlesung)

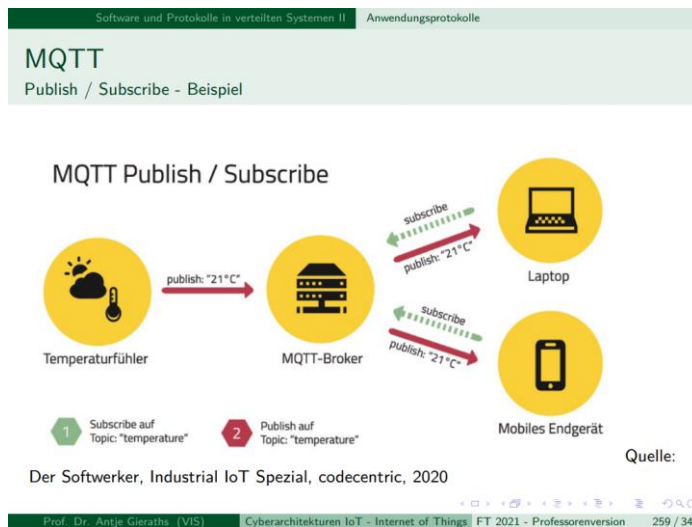


Abbildung 1: Funktionsweise MQTT, Vorlesung Gieraths S.259.

Das Internet Protokoll MQTT basiert auf dem Publisher / Subscriber Prinzip. Das heißt, dass Sender und Empfänger nichts übereinander wissen.

Hier gibt es eine Nachricht mit dem Wert „22.7“ des Topics *Temperatur*, sowie eine andere Nachricht mit dem Wert „An“ des Topics *Deckenlampe*. Beide sind allerdings Subtopics der Topic *Wohnzimmer*, welche wiederum selbst ein Subtopic von *Haus* ist.

Subtopics haben einen gleichen Anfang bis einschließlich eines Schrägstrichs. Im Code sehen Topics bzw. Subtopics so aus:

```
strLivingroomTempTopic = "Haus/Wohnzimmer/Temperatur"
```

Sie dienen nicht nur der Übersichtlichkeit, sondern auch der Organisation:

```
strAllLivingroomTopics = "Haus/Wohnzimmer/#"
```

Mit einem „#“ kann man hier zum Beispiel alle IoT Geräte im Wohnzimmer auf einmal abonnieren. Zudem dient ein „+“ als Platzhalter in der Mitte:

```
strAllIndoorTempTopics = "Haus/+ /Temperatur"
```

Hiermit lassen sich alle Temperatursensoren im Haus abfragen.

Die Zeit spielt auch eine wichtige Rolle. Nachrichten werden vom Broker immer nur an die Clients verteilt, die zum Zeitpunkt der Veröffentlichung dieses Topic abonniert haben. Sollte ein Client erst nach dem Veröffentlichen einer Nachricht das Topic einer Nachricht abonnieren, so wird er nicht vom Broker über diese Nachricht informiert.

```
▼ 127.0.0.1
▼ Haus
  ▼ Wohnzimmer
    Temperatur = 22.7
    Deckenlampe = An
  ▼ Kueche
    Temperatur = 28.3
```

Abbildung 2: Nachrichten und Topics

Eine Nachricht besteht nur aus ihrem Inhalt und einem Topic, um den Inhalt zu kategorisieren.

2 Mosquitto MQTT-Broker installieren

2.1 Mosquitto MQTT-Broker installieren [Windows]

2.1.1 Firewall konfigurieren

Falls Sie Windows nutzen müssen Sie ihre Firewall erst für *MQTT* konfigurieren.

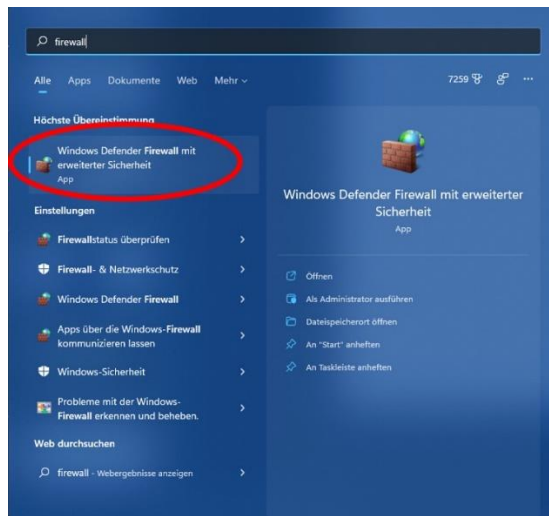


Abbildung 3: #1 Firewall Einstellungen öffnen

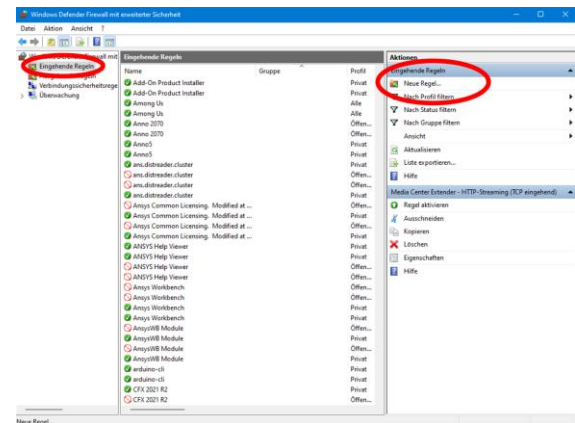


Abbildung 4: #2 Eingehende Regeln, Neue Regel

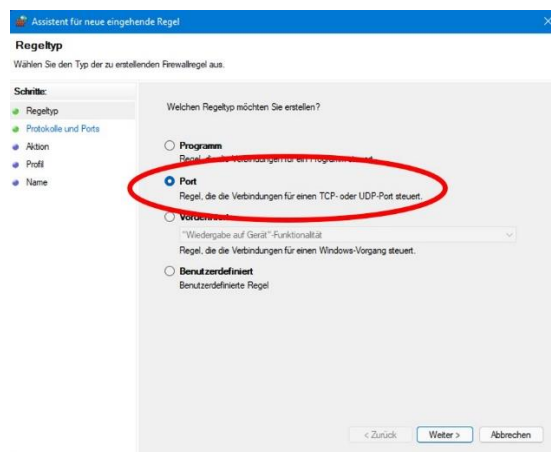


Abbildung 5: #3 Port auswählen

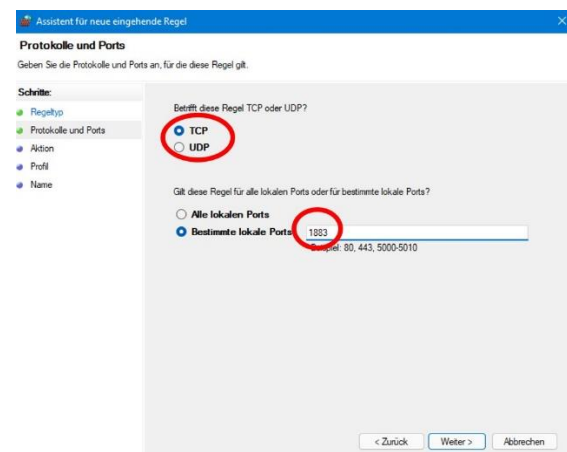


Abbildung 6: #4 TCP und 1883

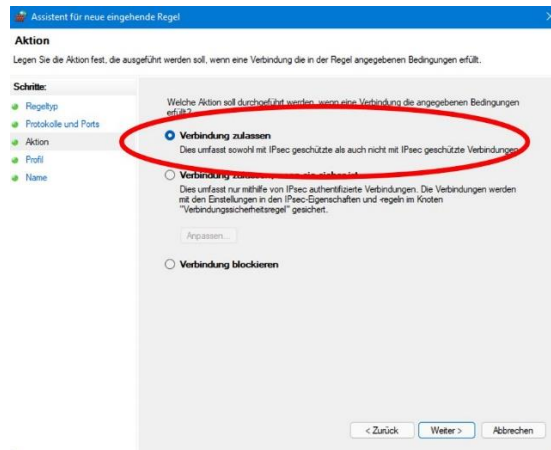


Abbildung 7: #5 Verbindung zulassen

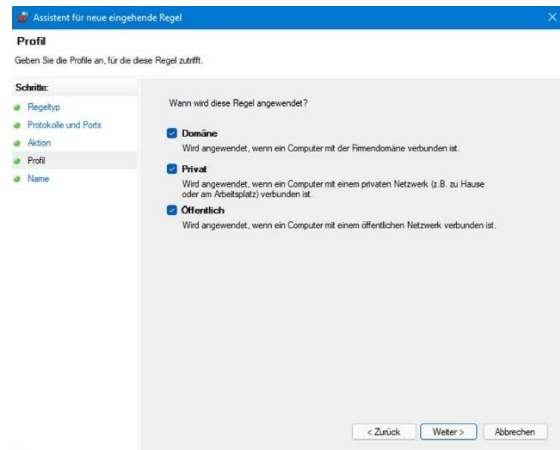


Abbildung 8: #6 Alles auswählen

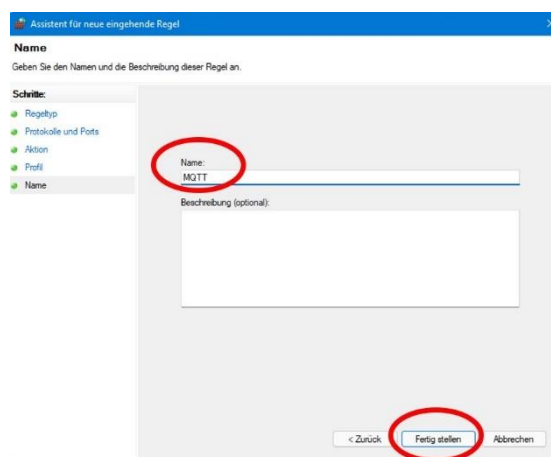


Abbildung 9: #7 Namen vergeben

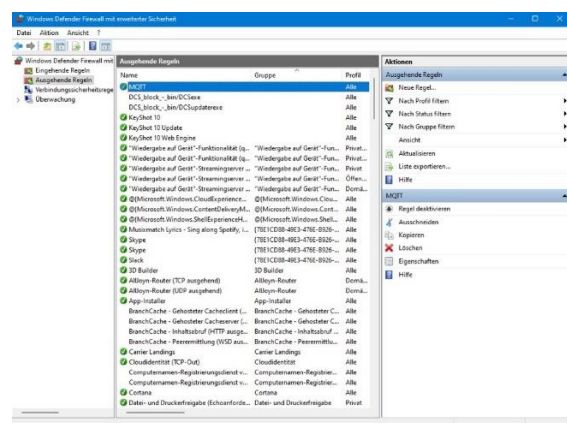
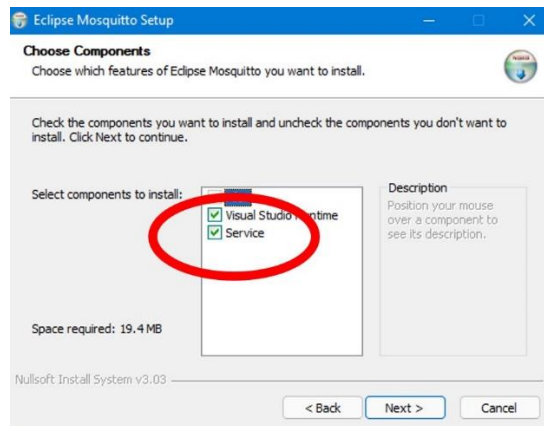


Abbildung 10: #8 Alle Schritte wiederholen für Ausgehende Regeln

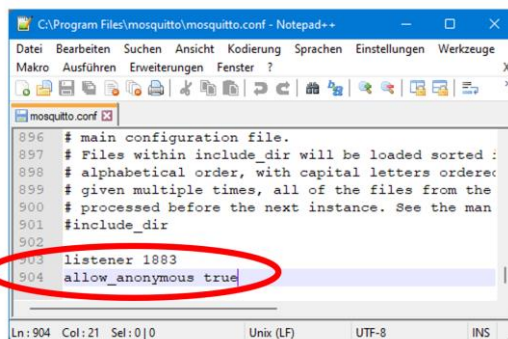
2.1.2 Broker installieren



Laden Sie sich den entsprechenden Installer (nicht den Source Code) für ihre Betriebssystem von <https://mosquitto.org/download/> herunter.

Stellen Sie sicher, dass Sie den Service mitinstallieren.

Abbildung 11: Mosquitto Installationswizard



„C:\Program Files\mosquitto\mosquitto.conf“

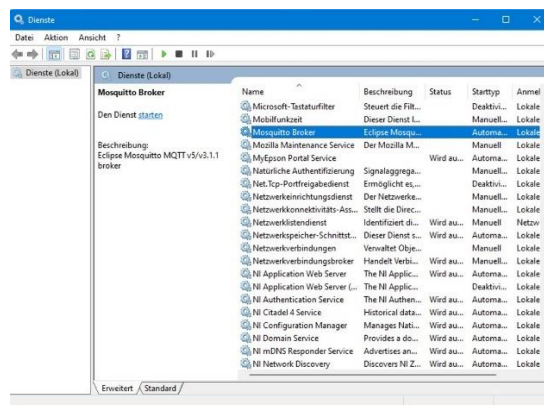
oder:

„C:\Programme\mosquitto\mosquitto.conf“

mit Texteditor öffnen und die unteren zwei Zeilen hinzufügen, speichern!

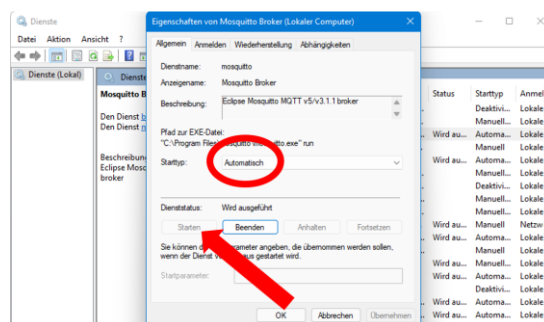
Erfordert eventuell Admin Berechtigungen -> Notepad++ im Admin Mode ausführen.

Abbildung 12: mosquitto.conf



Öffnen Sie die Windows Systemanwendung „Dienste“ (eng. „Services“) (einfach im Startmenü eintippen) und suchen Sie dort nach „Mosquitto Broker“.

Abbildung 13: Windows Systemanwendung "Dienste"



Bearbeiten Sie den Dienst und stellen Sie sicher, dass der Starttyp auf „Automatisch“ eingestellt ist. Starten Sie danach den Dienst.

Abbildung 14: Mosquitto Broker Dienst

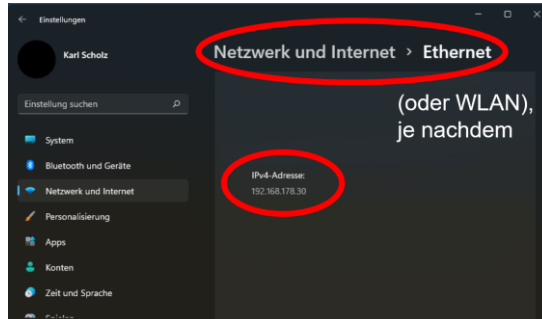

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\ >netstat -na | find "1883"
TCP    127.0.0.1:1883      0.0.0.0:0          ABHÖREN
TCP    [::]:1883          [::]:0             ABHÖREN

C:\Users\ >
```

Öffnen Sie ein neues Terminal und überprüfen Sie mit „`netstat -na | find „1883“`“, ob der Port abgehört wird.

Abbildung 15: Port 1883 wird abgehört



Finden Sie in den Windows Einstellungen heraus, wie ihre IPv4 Adresse lautet.

Abbildung 16: IPv4 Adresse in der App „Einstellungen“

```
Ethernet-Adapter Ethernet:
Verbindungsspezifisches DNS-Suffix:
Beschreibung. . . . . :
Physische Adresse . . . . . :
DHCP aktiviert. . . . . :
Autokonfiguration aktiviert . . . . . :
IPv6-Adresse. . . . . :
Temporäre IPv6-Adresse. . . . . :
Verbindungslokale IPv6-Adresse . . . . . :
IPv4-Adresse . . . . . : 192.168.178.30(Bevorzugt)
Subnetzmaske . . . . . :
Lease erhalten. . . . . :
Lease läuft ab. . . . . :
Standardgateway . . . . . :

DHCP-Server . . . . . :
DHCPv6-IAID . . . . . :
DHCPv6-Client-DUID. . . . . :
DNS-Server . . . . . :

NetBIOS über TCP/IP . . . . . :
```

Alternativ können Sie auch in einem Terminal Fenster „`ipconfig /all`“ eingeben und den von Ihnen benutzten Adapter überprüfen.

Abbildung 17: IPv4 Adresse in ipconfig /all

2.2 Mosquitto MQTT-Broker installieren [Linux]

1. Apt updaten:

```
wiweb@ubuntu:~$ sudo apt-get update && sudo apt-get upgrade -y
```

2. Mosquitto und mosquitto clients installieren

```
wiweb@ubuntu:~$ sudo apt-get install mosquitto mosquitto-clients
```

Nach dieser Zeile ist Mosquitto installiert und wird automatisch gestartet, auch nach jedem neuen Neustart.

3. Net-tools (*ifconfig*) installieren

```
wiweb@ubuntu:~$ sudo apt-get install net-tools
```

4. Mit *ifconfig* IPv4-Adresse herausfinden

```
wiweb@ubuntu:~$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Lokale Schleife)
    RX packets 1462 bytes 127682 (127.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1462 bytes 127682 (127.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.8.132 netmask 255.255.255.0 broadcast 192.168.8.255
    inet6 fe80::8af7:43e7:4d8:c682 prefixlen 64 scopeid 0x20<link>
    inet6 fd74:c14f:dc74:1800:4551:4ade:1f1a:b7c3 prefixlen 64 scopeid 0x0<global>
    inet6 fd74:c14f:dc74:1800:99c4:9bdf:3667:a5a3 prefixlen 64 scopeid 0x0<global>
    inet6 2a01:598:b890:8f7c:46cf:9f99:9f62:5353 prefixlen 64 scopeid 0x0<global>
    inet6 2a01:598:b890:8f7c:74c1:4fdc:7418:5 prefixlen 128 scopeid 0x0<global>
    inet6 2a01:598:b890:8f7c:f934:affa:3461:195d prefixlen 64 scopeid 0x0<global>
    ether f8:94:c2:ca:69:e4 txqueuelen 1000 (Ethernet)
    RX packets 5773 bytes 5881345 (5.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3509 bytes 618466 (618.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

In diesem Fall ist die IPv4-Adresse *192.168.8.132* .

Im Normalfall ist Mosquitto auf Linux standardmäßig passend eingerichtet, sollten Sie später dennoch Probleme haben, führen Sie noch die Schritte 5 und 6 aus.

5. Vim installieren

```
wiweb@ubuntu:~$ sudo apt-get install vim
```

6. Mosquitto config datei mit VIM bearbeiten

```
wiweb@ubuntu:~$ sudo vi /etc/mosquitto/mosquitto.conf
```


2.3 Mosquitto MQTT-Broker installieren [MAC]

1. Terminal Fenster öffnen und homebrew installieren:

```
(base) gf250@MacBook-Pro ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

2. Mosquitto und mosquitto clients installieren

```
(base) gf250@MacBook-Pro ~ % brew install mosquitto
```

3. Mosquitto Broker konfigurieren

```
(base) gf250@MacBook-Pro ~ % open /usr/local/etc/mosquitto/mosquitto.conf
```

4. Im Texteditor, der sich öffnet folgendes ganz unten einfügen:

listener 1883

allow_anonymous true

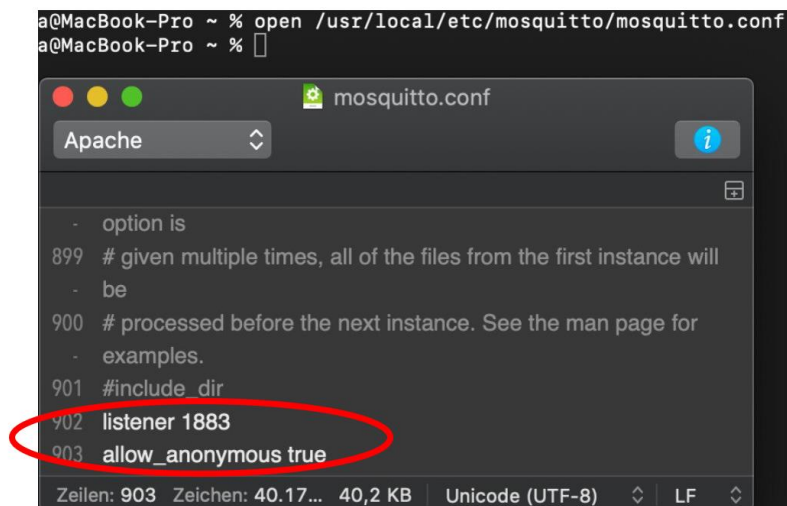


Abbildung 19: Konfigurieren von Mosquitto auf MAC

Speichern Sie die Datei und schließen sie den Texteditor.

5. Mosquitto Broker starten (muss jedes Mal eingegeben werden)

```
(base) gf250@MacBook-Pro ~ % /usr/local/sbin/mosquitto -c /usr/local/etc/mosquitto/mosquitto.conf
```

Falls Sie mit folgender Sicherheitswarnung konfrontiert werden, akzeptieren Sie diese.



Abbildung 20: Sicherheitswarnung von MAC

6. Mosquitto Broker beenden

Ctrl+C in das offene Mosquitto Terminal eingeben oder folgenden Befehl ausführen:

```
(base) gf250@MacBook-Pro ~ % killall mosquitto
```

7. Mit *ifconfig* IPv4-Adresse herausfinden

```
(base) gf250@MacBook-Pro ~ % ifconfig
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=400<CHANNEL_IO>
ether 20:c9:d0:86:58:3f
inet6 fe80::18ad:f625:b9cf:a329%en0 prefixlen 64 secured scopeid 0x4
inet 192.168.137.62 netmask 0xffffffff00 broadcast 192.168.137.255
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: active
```

In diesem Fall ist die *IPv4*-Adresse *192.168.137.62* .

3 MQTT-Explorer

3.1 Installation

Installieren Sie das Programm *MQTT-Explorer*.

- Windows: Laden Sie sich das Programm hier herunter: <http://mqtt-explorer.com/>

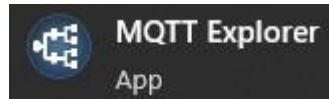


Abbildung 21: MQTT-Explorer Logo

- Linux

1. Apt updaten:

```
wiweb@ubuntu:~$ sudo apt-get update && sudo apt-get upgrade -y
```

2. Snap store installieren:

```
wiweb@ubuntu:~$ sudo apt-get install snapd
```

3. MQTT-Explorer installieren:

```
wiweb@ubuntu:~$ sudo snap install mqtt-explorer
```

4. MQTT-Explorer öffnen mit:

```
wiweb@ubuntu:~$ mqtt-explorer
```

- MAC

1. MQTT-Explorer installieren

```
(base) gf250@MacBook-Pro ~ % brew install mqtt-explorer
```

2. In Finder/Programme MQTT Explorer öffnen

3.2 Benutzung

Geben Sie die IPv4 Adresse des Computers, auf dem der *MQTT*-Broker läuft, ein. Falls das der gleiche PC ist, auf dem Sie *MQTT-Explorer* benutzen wollen, können Sie auch die IPv4-Adresse *127.0.0.1* verwenden, dies sind immer Sie selbst.

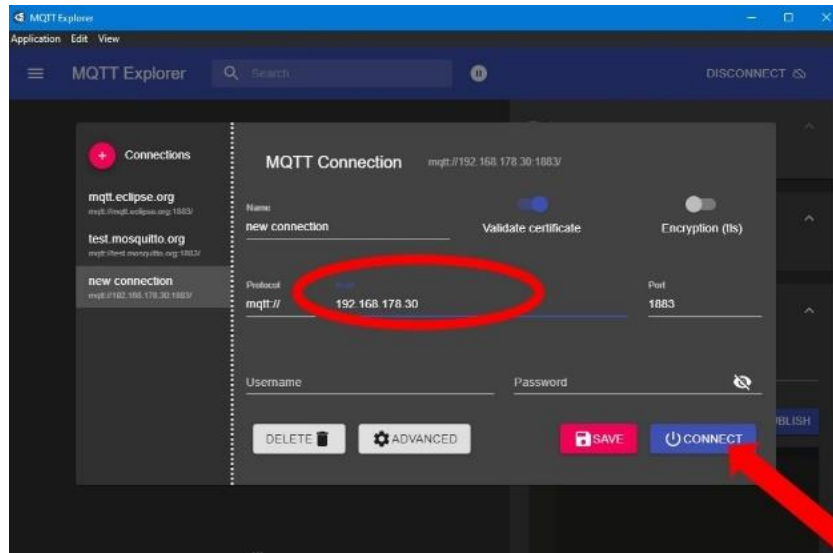


Abbildung 22: MQTT-Explorer: Anmeldung beim Broker

Der *MQTT-Explorer* abonniert beim Anmelden alle Topics. Somit können Sie alle Nachrichten sehen, die beim Broker veröffentlicht werden.

Veröffentlichen Sie eine Nachricht unter dem Topic „Test“ und überprüfen Sie den Empfang.

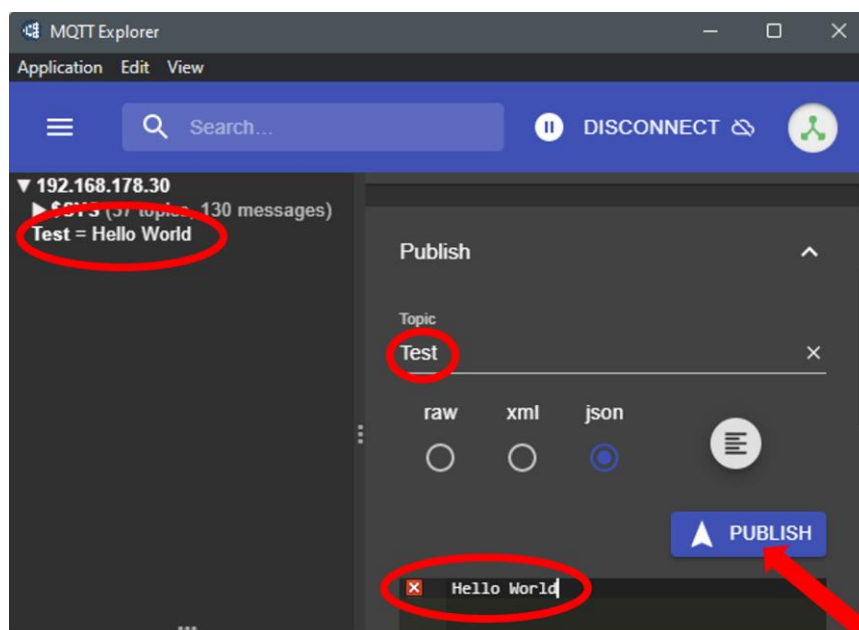


Abbildung 23: Erste Veröffentlichung einer Nachricht

4 Python installieren und einrichten

Linux und MAC haben Python standardmäßig vorinstalliert. Wenn Sie Windows nutzen, laden Sie sich Python von <https://www.python.org/> herunter und führen Sie den Installer aus. Der Haken bei *Add Python to PATH* soll gesetzt sein.

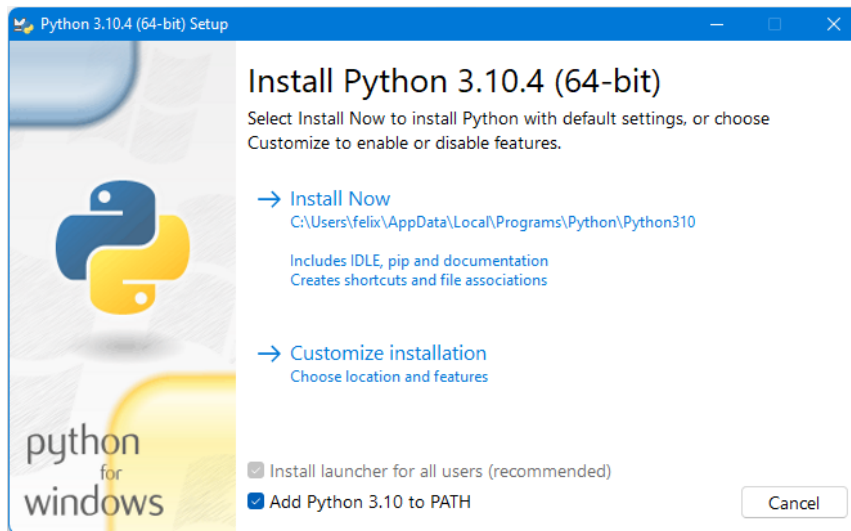


Abbildung 24: Python Installationswizard

Öffnen Sie ein neues Terminal Fenster und installieren Sie die paho-mqtt-Bibliothek über den *Package Installer for Python (PIP)*

```
pip install paho-mqtt
```

Stellen Sie danach sicher, dass die Bibliothek installiert ist.

```
pip list
```

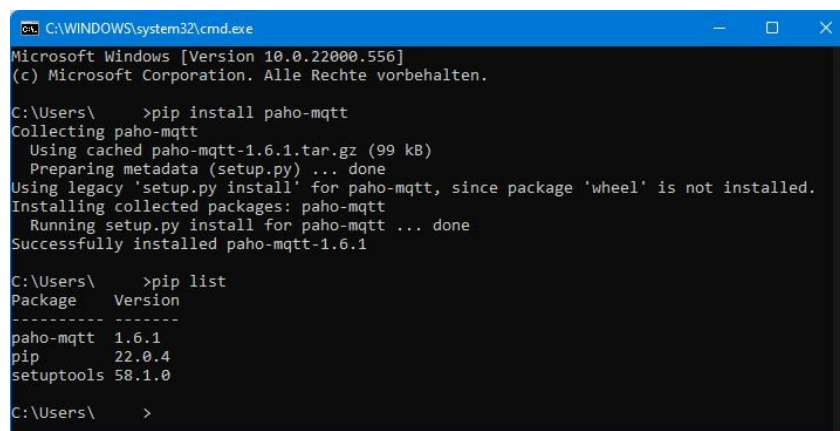


Abbildung 25: Bibliothekinstallation mit PIP

5 Visual Studio Code mit Extensions installieren

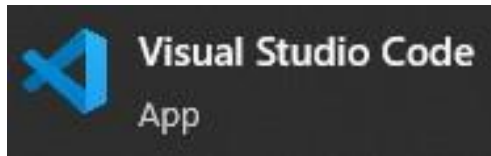


Abbildung 26: Visual Studio Code Logo

Installieren Sie das Programm *Visual Studio Code*. Dieses können Sie von <https://code.visualstudio.com/> herunterladen.

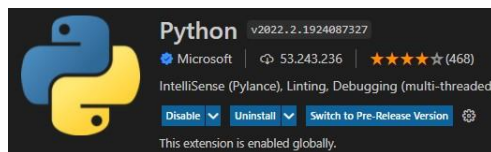


Abbildung 27: Python Extension

Navigieren Sie (links am Rand) in *VSCode* zum Punkt *Extensions* und suchen sie nach *Python*, installieren sie diese und alle weiteren Extensions, die Ihnen während der Installation vorgeschlagen werden.



Abbildung 28: Code Runner Extension

Installieren Sie außerdem die *Code Runner* Extension.

Diese fügt ihrer *IDE* einen Knopf zum Ausführen von Programmen hinzu.

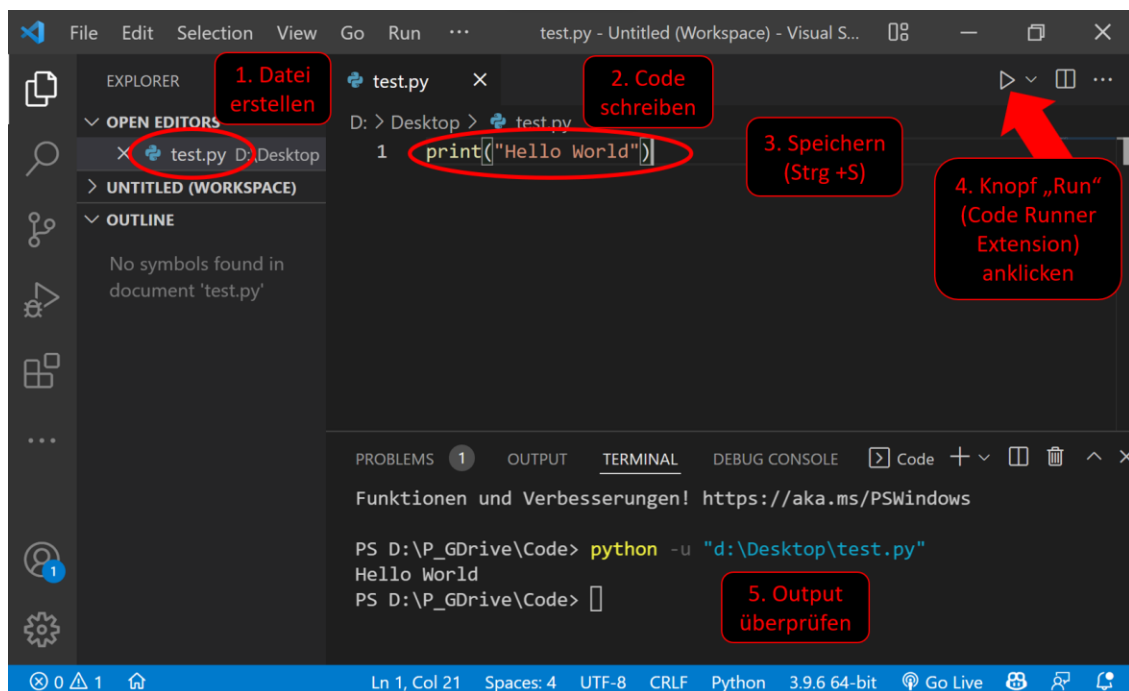


Abbildung 29: Testen der Umgebung via „Hello World“ Programm

Bis hierher sollten Sie das Praktikum im Vorfeld vorbereiten. Die nachfolgenden Aufgaben sind während des Praktikums zu bearbeiten. Überprüfen Sie anhand folgender Liste, ob Sie alle Punkte abgearbeitet haben.

- Sie sind mit den Grundlagen von MQTT vertraut ☐
- Sie haben den *MQTT*-Broker „Mosquitto“ installiert ☐
- Sie wissen, wie Sie ihre *IPv4*-Adresse herausfinden können ☐
- Sie haben das *MQTT* Debugging Programm *MQTT-Explorer* installiert ☐
- Sie haben Python installiert ☐
- Sie haben *Visual Studio Code* mit den Extensions installiert ☐

6 Pong Back Programm schreiben

In dieser Aufgabe werden Sie ein Programm in der Programmiersprache *Python* umsetzen, welches Nachrichten, die von Ihnen manuell in dem Subtopic *Ping* veröffentlicht werden auch in der Subtopic Pong zu veröffentlichen.

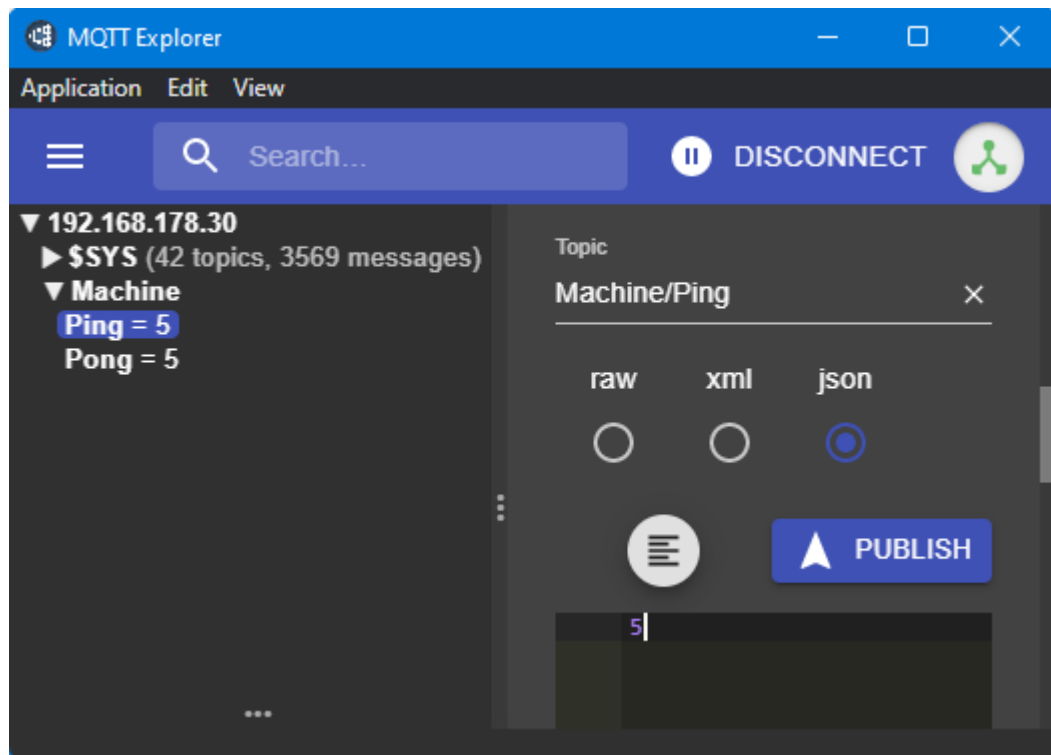


Abbildung 30: PongBack-Maschine

Erstellen Sie dazu eine leere Datei und speichern sie als *.py*-Datei ab, zum Beispiel:

PongBack.py

6.1 Bibliothek einbinden

Binden Sie die Bibliothek ein, es empfiehlt sich immer einen Alias zu nutzen, hier *mqtt*.

```
import paho.mqtt.client as mqtt
```

6.2 Callback Funktion definieren

Um Funktionen im Code zu benutzen, müssen sie bekannt und damit vor der Benutzung im Code definiert sein.

Definieren Sie deshalb eine Funktion

```
def Callback():
```

mit den Parametern

```
client, userdat, message
```

Diese wird später von der Bibliothek immer dann aufgerufen, wenn eine neue Nachricht auf ein von Ihnen abonniertes Topic ankommt.

Im Körper der Funktion müssen wir als erstes die Nachricht in Topic und Payload aufteilen und dekodieren.

Erstellen Sie eine Variable für den Inhalt der angekommenen Nachricht

```
strMessage =
```

und weisen Sie ihr den Wert von *message.payload* mit *utf-8* dekodiert und als String konvertiert zu.

```
str(message.payload.decode("utf-8"))
```

Erstellen Sie eine weitere Variable für das Topic der Nachricht

```
strTopic =
```

und weisen Sie ihr direkt den Wert von *message.topic* zu. Hier ist keine Typkonvertierung oder Dekodierung notwendig.

Im folgenden Abschnitt soll, sofern das Topic wirklich das *Ping*-Subtopic ist, dieselbe Nachricht im *Pong*-Subtopic veröffentlicht werden.

Prüfen sie dementsprechend ob

```
strTopic == "Machine/Ping"
```

wahr ist.

Falls ja, veröffentlichen Sie eine Nachricht mit der Funktion *publish()* des Objektes *client*, was wir noch erstellen werden.

```
client.publish()
```

Als Parameter sollen Sie zuerst das Topic

```
"Machine/Pong"
```

und danach den als String ausgelesenen Inhalt der empfangenen Nachricht übergeben.

Lassen Sie sich zu Debugging Zwecken mit der Funktion *print()* ausgeben, was gerade passiert.

Zur Erinnerung: *print()* benutzen Sie so:

```
var = 3
print("1", str(2), var)
```

Output:

1 2 3

6.3 Client Objekt einrichten

Erstellen Sie eine neue Variable

```
client =
```

und initialisieren diese mit einem neuen Objekt der Klasse

```
mqtt.Client()
```

Diese benötigt von ihnen eine *Client-ID*, d.h. einen Namen, mit dem Sie sich beim Broker später identifizieren kann. Dieser ist frei wählbar, zum Beispiel:

```
"PongBack-Machine"
```

Führen Sie anschließend die Funktion

```
connect()
```

des Objekts aus und übergeben Sie ihr als erstes einen String mit der IPv4-Adresse ihres Brokers ohne Angabe des Ports und als zweiten Parameter, als Integer, den Port. Dies sieht zum Beispiel so aus:

```
"127.0.0.1", 1883
```

Damit das Subtopic *Ping* im *MQTT-Explorer* erscheint, soll jetzt eine erste Nachricht unter diesem Topic veröffentlicht werden. Benutzen Sie dazu die gleiche Funktion, die Sie auch in der Callback Funktion *on_message* am Ende benutzt haben. Stellen Sie aber sicher, dass Sie die Nachricht unter dem Topic

```
"Machine/Ping"
```

veröffentlichen. Den Inhalt können Sie beliebig wählen.

Nach dem Veröffentlichen sollen sie das Topic abonnieren („subscribe“). Führen Sie dazu die Funktion

```
subscribe()
```

des *client*-Objektes aus und übergeben Sie als Parameter einen String mit dem *Ping*-Subtopic.

Als letztes müssen Sie ihrem *client*-Objekt noch ihre Callback-Funktion bekannt machen.

Setzen sie dazu die Variable

```
on_message =
```

ihrer *client*-Objektes auf den Namen ihrer Callback Funktion ohne „()“.

Jetzt ist das Objekt eingerichtet und die Callback Funktion fertig implementiert. Geben Sie sich mit einem Aufruf der *print()* Funktion aus, dass das Programm jetzt läuft.

Danach rufen Sie die Funktion

```
loop_forever()
```

ihres *client*-Objektes auf.

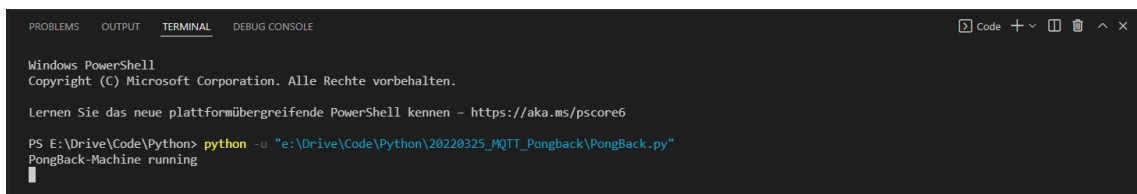
Die PongBack-Maschine ist fertig implementiert und kann ausgeführt werden.

6.4 Programm ausführen und testen

Öffnen Sie zuerst das Programm *MQTT-Explorer* und verbinden sich, wie unter 3 *MQTT-Explorer*

beschrieben, mit ihrem Broker, da sie sonst nicht mitbekommen, wenn Nachrichten veröffentlicht werden.

Führen Sie ihr Programm wie in Kapitel 5 *Visual Studio Code mit Extensions* in *Abbildung 28: Code Runner Extension* beschrieben, aus.



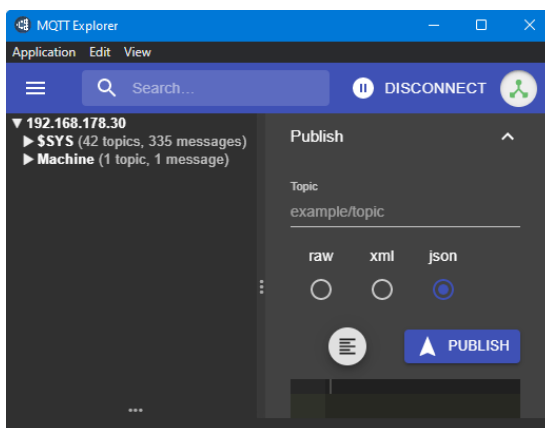
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Lernen Sie das neue plattformübergreifende PowerShell kennen - https://aka.ms/pscore6

PS E:\Drive\Code\Python> python -u "e:\Drive\Code\Python\20220325_MQTT_Pongback\PongBack.py"
PongBack-Machine running
  
```

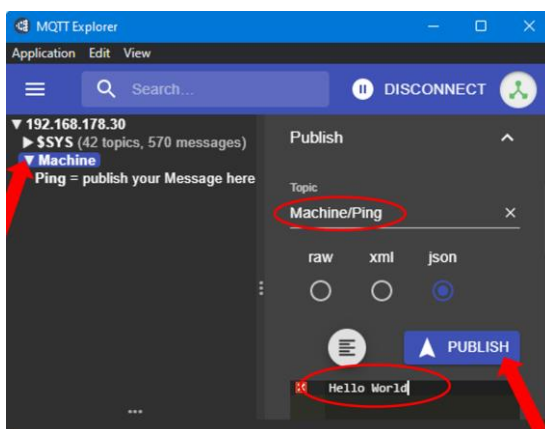
Abbildung 31: PongBack Programm läuft



Sie sollten nun ein neues Terminal bekommen, in dem Sie den Inhalt ihrer Ausgabe, die Sie vor *client.loop_forever()* geschrieben haben, sehen können.

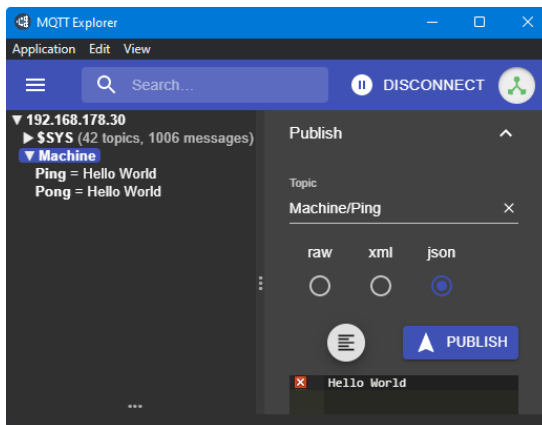
Zusätzlich sollten Sie nun im *MQTT-Explorer* ein neues Topic *Machine* sehen.

Abbildung 32: MQTT-Explorer nach Programmstart



Durch Klicken auf das Topic *Machine* klappen Sie die Subtopics auf. Hier sollte schon eine Nachricht auf das Subtopic *Ping* veröffentlicht worden sein. Veröffentlichen Sie eine beliebige Nachricht in dem Subtopic *Ping*.

Abbildung 33: Subtopics von Machine



Nach dem Veröffentlichen sollte ihre Nachricht erst bei dem Subtopic *Ping* und kurz darauf auch bei dem Subtopic *Pong* auftauchen. Zusätzlich sollten Sie in dem Terminal, in dem ihr Python Skript läuft ihre Ausgabe aus der Callback Funktion sehen.

Abbildung 34: Nachricht auch von Python veröffentlicht

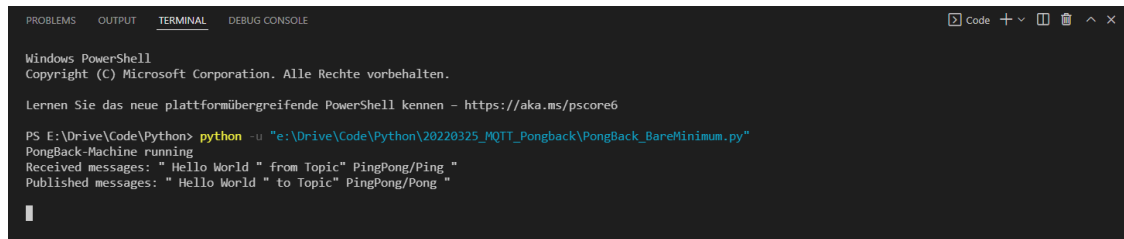


Abbildung 35: Ausgabe aus ihrer Callback Funktion im Python Terminal

Beenden Sie das Programm, in dem Sie zur Laufzeit Strg+C in die Konsole eingeben.

6.5 Zusatzaufgabe: Inkrement

Bis jetzt wurden nur Nachrichten verschickt und nicht mit Variablen interagiert. Erstellen Sie zuerst ganz oben eine neue Variable und initialisieren sie mit dem Wert 0.

```
iZahl = 0
```

Um aus der Callback-Funktion darauf zugreifen zu können müssen wir sie im Körper der Funktion noch einmal mit dem Schlüsselwort *global* davor definieren.

```
global iZahl
```

Jetzt können Sie die Variable auch in der Callback-Funktion wie gewohnt benutzen.

Schreiben Sie ihr Programm so um, sodass die Variable *iZahl*, mit der Zahl, die Sie unter *Ping* veröffentlichen erhöht wird.

Tipp: Sie können sich auch rechts am Rand über den „Trend“-Knopf den Verlauf der Nachrichten eines Topics graphisch anzeigen lassen. Somit ist *Ping* quasi $f'(x)$ von *Pong*.

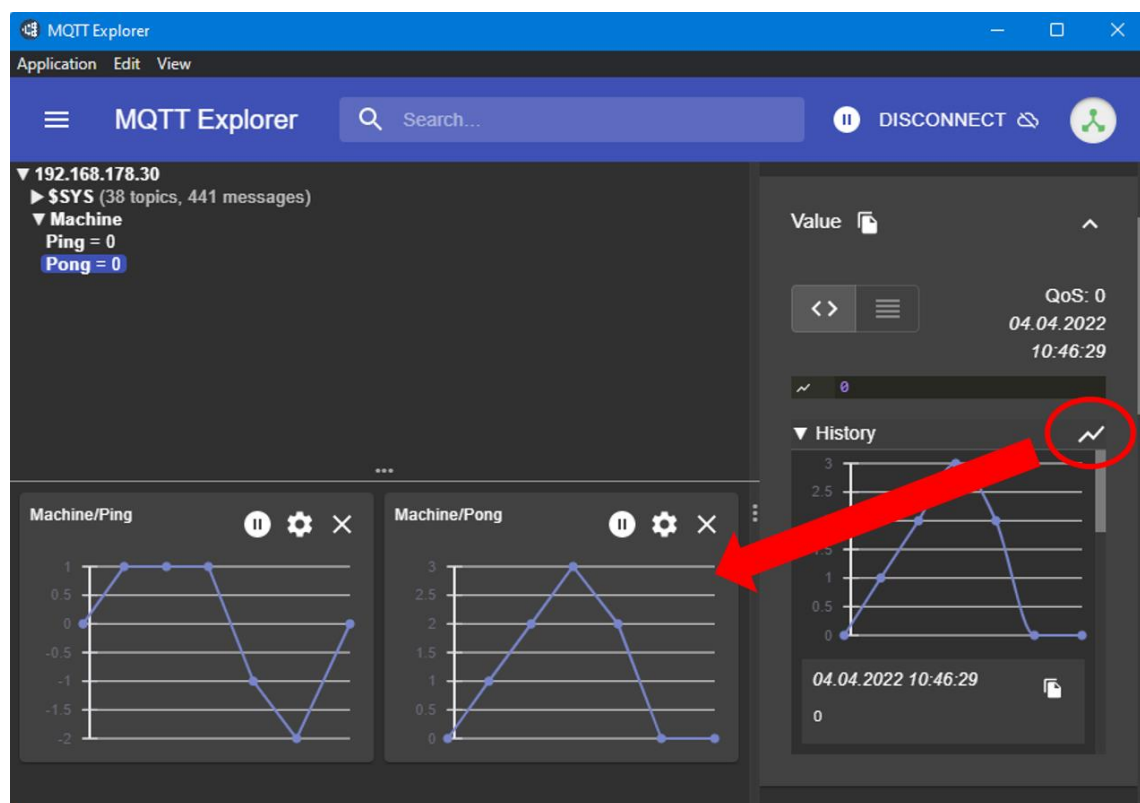


Abbildung 36: Zusatzaufgabe Inkrement: graphisch Darstellung

```
TERMINAL ... Code + - [ ] [X] < X

PS D:\P_GDrive\Code> python -u "d:\P_GDrive\Code\Python\20220325_MQTT_Pongback\PongBack_IncrementZusatzaufgabe.py"
PongBack-Increment-Machine running
Received messages: " 0 " from Topic" Machine/Ping "
Published messages: " 0 " to Topic" Machine/Pong "

Received messages: " 1 " from Topic" Machine/Ping "
Published messages: " 1 " to Topic" Machine/Pong "

Received messages: " 1 " from Topic" Machine/Ping "
Published messages: " 2 " to Topic" Machine/Pong "

Received messages: " 1 " from Topic" Machine/Ping "
Published messages: " 3 " to Topic" Machine/Pong "

Received messages: " -1 " from Topic" Machine/Ping "
Published messages: " 2 " to Topic" Machine/Pong "

Received messages: " -2 " from Topic" Machine/Ping "
Published messages: " 0 " to Topic" Machine/Pong "

Received messages: " 0 " from Topic" Machine/Ping "
Published messages: " 0 " to Topic" Machine/Pong "
```

Abbildung 37: Zusatzaufgabe Inkrement: Ausgabe im Terminal

Abbildungsverzeichnis

Abbildung 1: Funktionsweise MQTT, Vorlesung Gieraths S.259.....	5
Abbildung 2: Nachrichten und Topics	5
Abbildung 3: #1 Firewall Einstellungen öffnen.....	6
Abbildung 4: #2 Eingehende Regeln, Neue Regel.....	6
Abbildung 5: #3 Port auswählen	6
Abbildung 6: #4 TCP und 1883	6
Abbildung 7: #5 Verbindung zulassen.....	7
Abbildung 8: #6 Alles auswählen	7
Abbildung 9: #7 Namen vergeben	7
Abbildung 10: #8 Alle Schritte wiederholen für Ausgehende Regeln	7
Abbildung 11: Mosquitto Installationswizard	8
Abbildung 12: mosquitto.conf.....	8
Abbildung 13: Windows Systemanwendung "Dienste"	8
Abbildung 14: Mosquitto Broker Dienst	8
Abbildung 15: Port 1883 wird abgehört	9
Abbildung 16: IPv4 Adresse in der App „Einstellungen“	9
Abbildung 17: IPv4 Adresse in ipconfig /all	9
Abbildung 18: Bearbeitung von mosquitto.conf mit VIM	11
Abbildung 19: Konfigurieren von Mosquitto auf MAC	12
Abbildung 20: Sicherheitswarnung von MAC	13
Abbildung 21: MQTT-Explorer Logo	14
Abbildung 22: MQTT-Explorer: Anmeldung beim Broker	15
Abbildung 23: Erste Veröffentlichung einer Nachricht.....	15
Abbildung 24: Python Installationswizard	16
Abbildung 25: Bibliothekinstallation mit PIP	16
Abbildung 26: Visual Studio Code Logo	17
Abbildung 27: Python Extension	17
Abbildung 28: Code Runner Extension	17
Abbildung 29: Testen der Umgebung via „Hello World“ Programm	17
Abbildung 30: PongBack-Maschine	19
Abbildung 31: PongBack Programm läuft	23
Abbildung 32: MQTT-Explorer nach Programmstart	23
Abbildung 33: Subtopics von Machine	23
Abbildung 34: Nachricht auch von Python veröffentlicht.....	24
Abbildung 35: Ausgabe aus ihrer Callback Funktion im Python Terminal	24
Abbildung 36: Zusatzaufgabe Inkrement: graphisch Darstellung.....	25
Abbildung 37: Zusatzaufgabe Inkrement: Ausgabe im Terminal	26

Alle Abbildungen dieses Dokumentes sind Screenshots von Windows, Linux, Mac, Visual Studio Code oder MQTT-Explorer und wurden vom Verfasser erstellt. Einzige Ausnahme ist Abbildung 1, diese stammt aus dem Vorlesungsskript „Internet of Things“, Seite 259, Frau Prof. Dr. rer. nat. Antje Gieraths.