

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

Frühjahrstrimester 2023

Verfasser: Karl Scholz

Internet of Things Praktikum

Teil 1
Einrichtung der Software und der IDE
Montag, 15.05.2023

```
a@MacBook-Pro ~ % open /usr/local/etc/mosquitto/mosquitto.conf
a@MacBook-Pro ~ % □

mosquitto.conf
Apache i

- option is
899 # given multiple times, all of the files from the first instance will
- be
900 # processed before the next instance. See the man page for
- examples.
901 #include dir
902 listener 1883
903 allow_anonymous true

Zeilen: 903 Zeichen: 40.17... 40,2 KB | Unicode (UTF-8) ▲ | LF ▲
```

Inhaltsverzeichnis

Inhalt

Inhaltsverzeichnis	2
Abkürzungsverzeichnis	3
1 Mosquitto MQTT-Broker installieren.....	4
1.1 Mosquitto MQTT-Broker installieren [Windows]	4
1.1.1 Firewall konfigurieren	4
1.1.2 Broker installieren	6
1.2 Mosquitto MQTT-Broker installieren [Linux]	8
1.3 Mosquitto MQTT-Broker installieren [MAC]	10
2 MQTT-Explorer	12
2.1 Installation	12
2.2 Benutzung.....	13
3 Python installieren.....	14
4 Visual Studio Code mit Extensions für Python Entwicklung installieren....	15
5 Vorbereitung von VSCode für Mikrocontrollerprogrammierung	16
5.1 IDE	16
5.2 Extensions für die IDE	16
6 Hardwaredaten laden.....	17
7 Einrichten des Multikopter IoT Projekts.....	18
7.1 Vorbereitung.....	18
7.2 Öffnen des Projekts	18
7.3 Ordnerstruktur	19
Abbildungsverzeichnis.....	21

Abkürzungsverzeichnis

COM	serielle COMmunication Schnittstelle
ESP32	Espressif 32
IoT	Internet of Things
GPIO	General Purpose Input / Output
GND	Ground / Masse
I ² C	Inter-Integrated Circuit Bus
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit (Gyroskop und Beschleunigungsmesser)
IPv4	Internet Protocol Version 4
LED	Light Emitting Diode
LiDAR	Light Detection And Ranging
MP	Mission Pad
MQTT	Message Queuing Telemetry Transport
PIO	PlatformIO
PIP	Package Installer for Python
PWM	Pulse Width Modulation
QoS	Quality of Service (MQTT)
RGB	Red Green Blue
SDK	Software Development Kit
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VSCODE	Visual Studio Code
WiFi	Wireless Fidelity IEEE-802.11

1 Mosquitto MQTT-Broker installieren

1.1 Mosquitto MQTT-Broker installieren [Windows]

1.1.1 Firewall konfigurieren

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

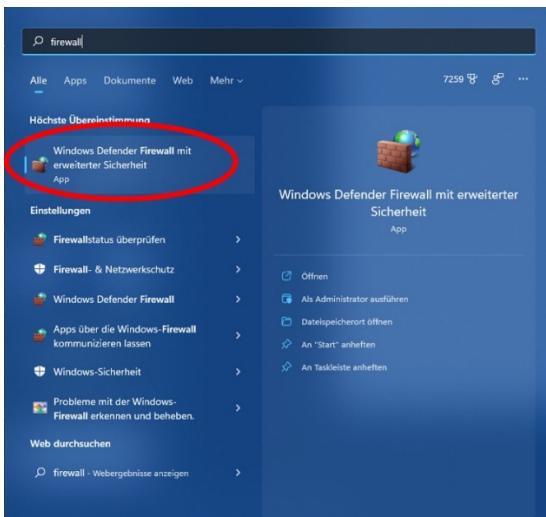


Abbildung 1: #1 Firewall Einstellungen öffnen

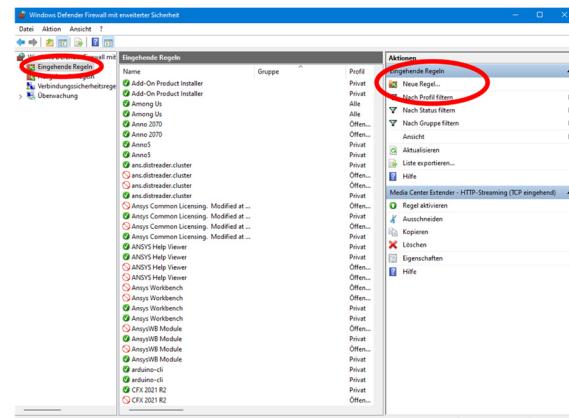


Abbildung 2: #2 Eingehende Regeln, Neue Regel

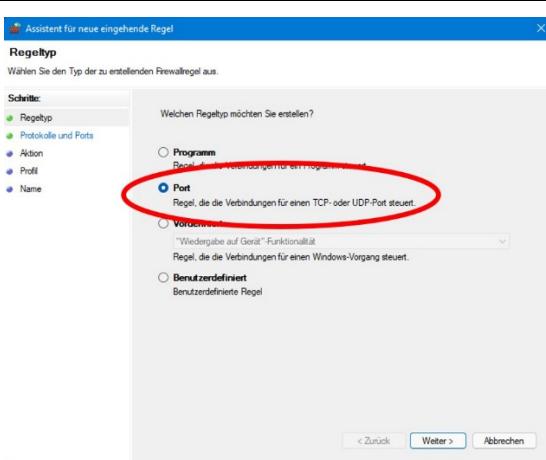


Abbildung 3: #3 Port auswählen

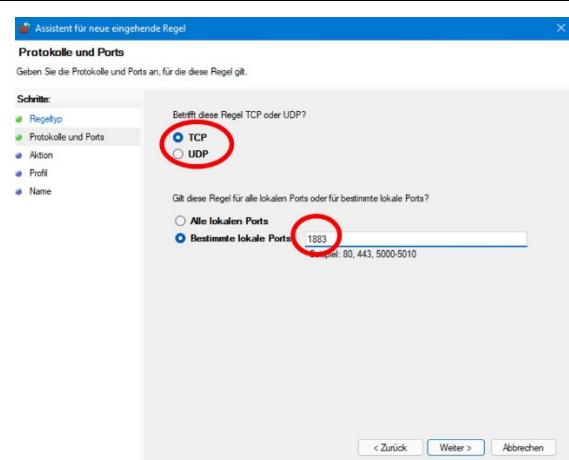


Abbildung 4: #4 TCP und 1883

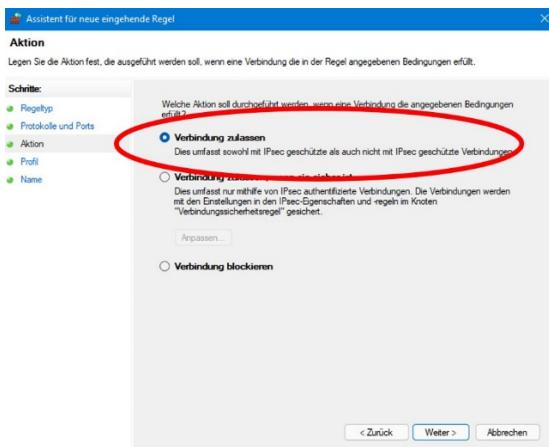


Abbildung 5: #5 Verbindung zulassen

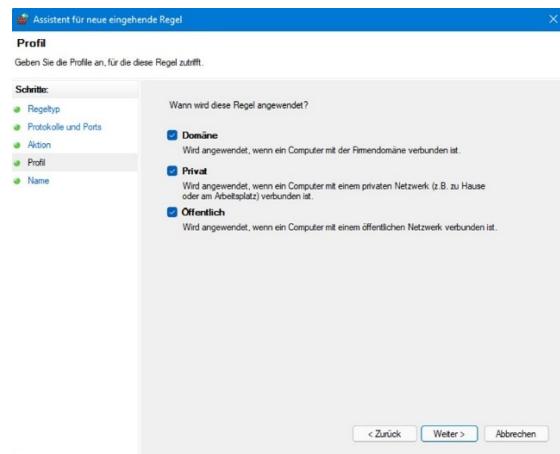


Abbildung 6: #6 Alles auswählen

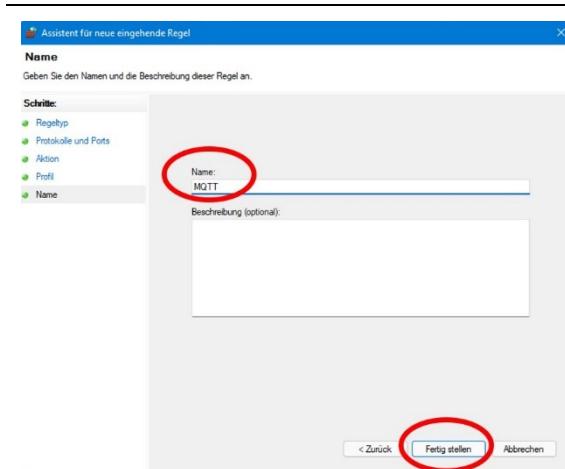


Abbildung 7: #7 Namen vergeben

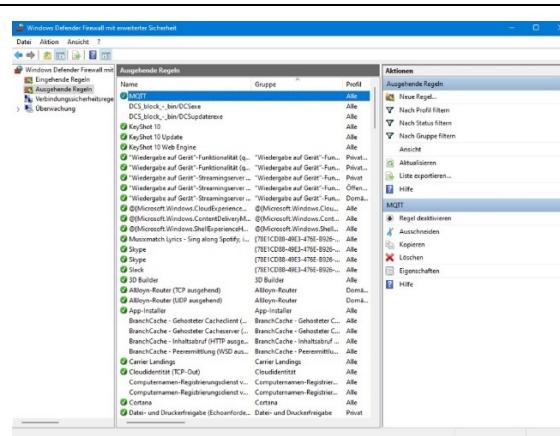
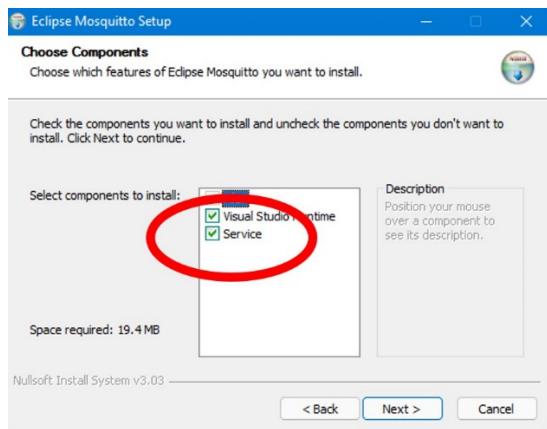


Abbildung 8: #8 Alle Schritte wiederholen für Ausgehende Regeln

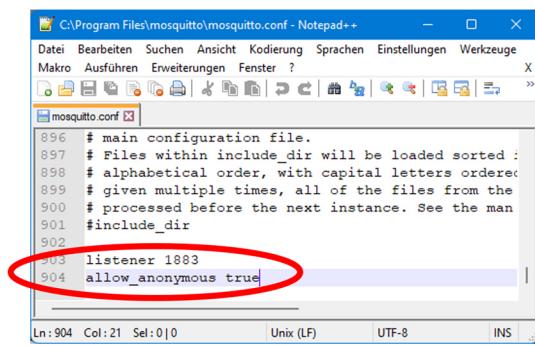
1.1.2 Broker installieren



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 9: Mosquitto Installationswizard



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

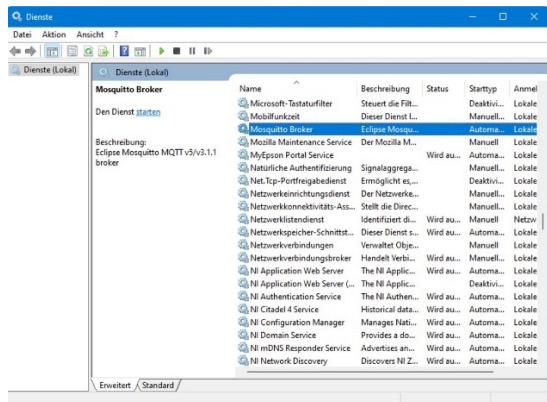
oder:

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

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

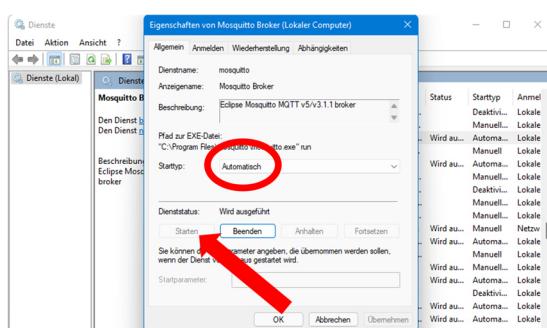
Erfordert eventuell Admin Berechtigungen -> Notepad++ im Admin Mode ausführen.

Abbildung 10: mosquitto.conf



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

Abbildung 11: Windows Systemanwendung "Dienste"



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

Abbildung 12: Mosquitto Broker Dienst

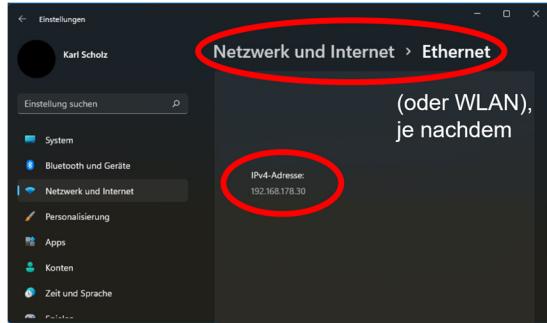
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\ >netstat -na | find "1883"
  TCP  127.0.0.1:1883      0.0.0.0:0          ABHÖREN
  TCP  [:1]:1883           [:]:0              ABHÖREN

C:\Users\ >
```

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

Abbildung 13: Port 1883 wird abgehört



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

Abbildung 14: 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 15: IPv4 Adresse in ipconfig /all

1.2 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.

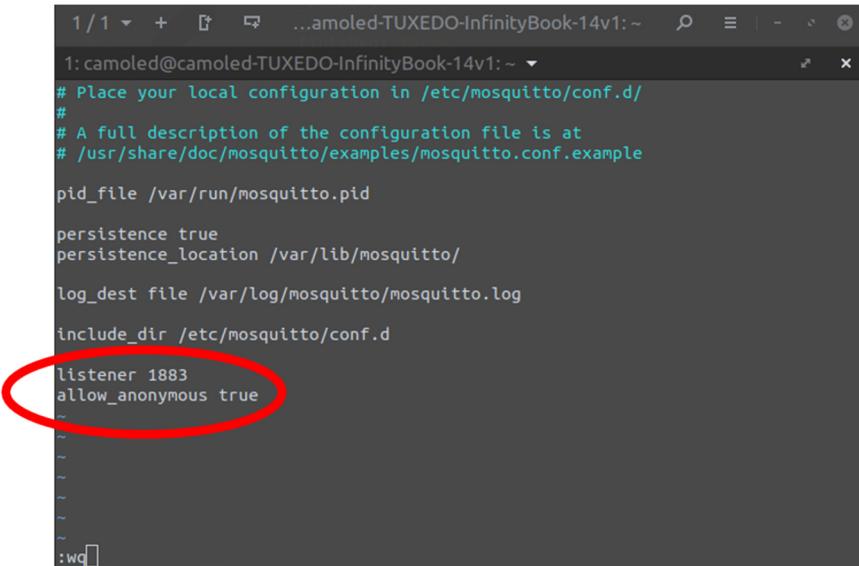
5. Vim installieren

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

6. Mosquitto config datei mit VIM bearbeiten

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

listener 1883
allow_anonymous true



```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 1883
allow_anonymous true
```

Abbildung 16: Bearbeitung von mosquitto.conf mit VIM

Klicken Sie die Taste *i* und schreiben Sie die beiden rot umrahmten Zeilen am Ende dazu. Klicken Sie anschließend *ESC* und speichern/schließen sie durch tippen von *:wq* (mit Doppelpunkt, siehe Abbildung 16).

1.3 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

INTEL

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

ARM

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

listener 1883

allow_anonymous true

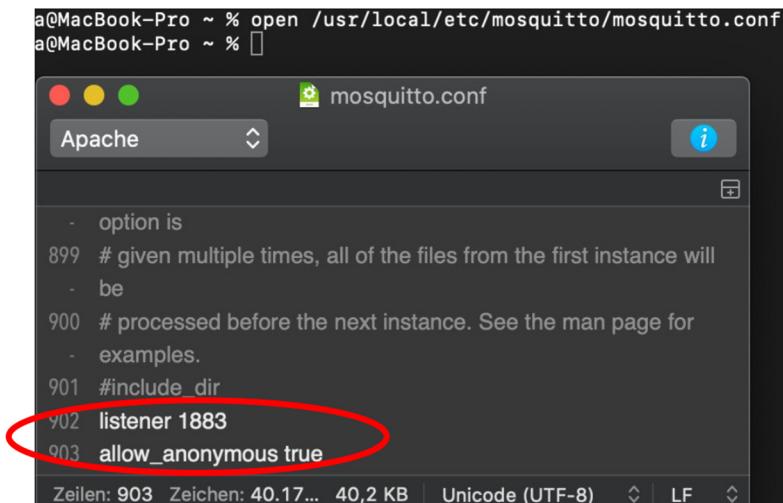


Abbildung 17: Konfigurieren von Mosquitto auf MAC

Speichern Sie die Datei und schließen sie den Texteditor.

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

INTEL

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

ARM

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

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



Abbildung 18: 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 0xffffffff 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**.

2 MQTT-Explorer

2.1 Installation

Installieren Sie das Programm *MQTT-Explorer*.

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

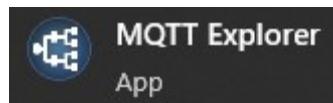


Abbildung 19: MQTT-Explorer Logo

- **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

2.2 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.

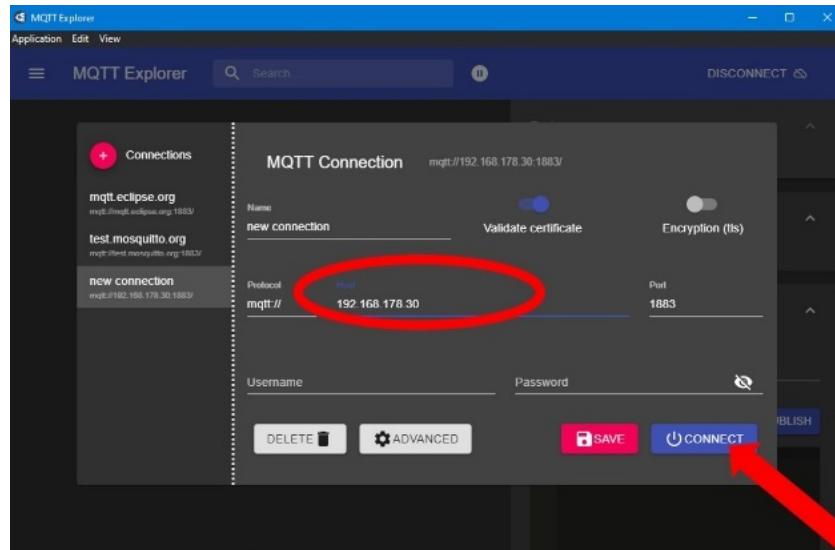


Abbildung 20: MQTT-Explorer: Anmeldung beim Broker

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.

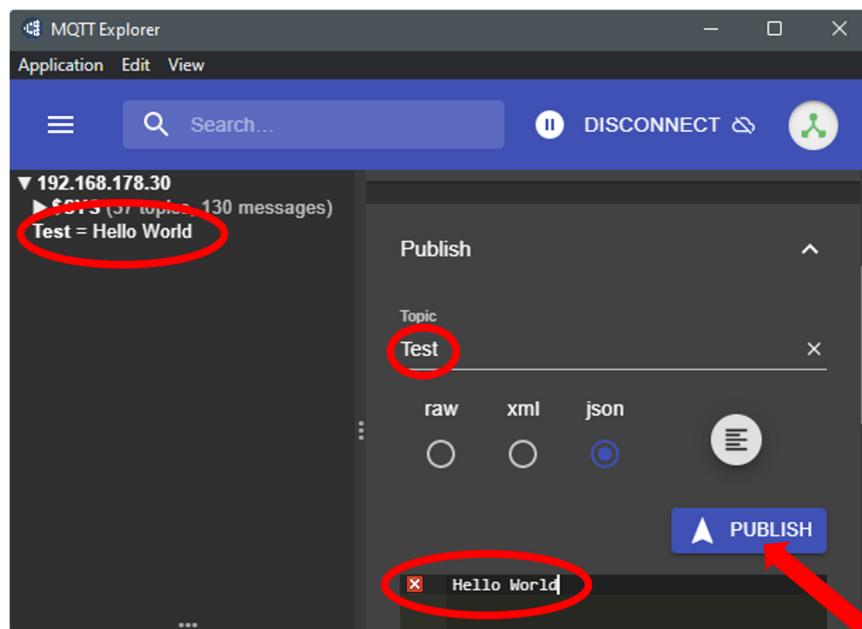


Abbildung 21: Erste Veröffentlichung einer Nachricht

3 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.

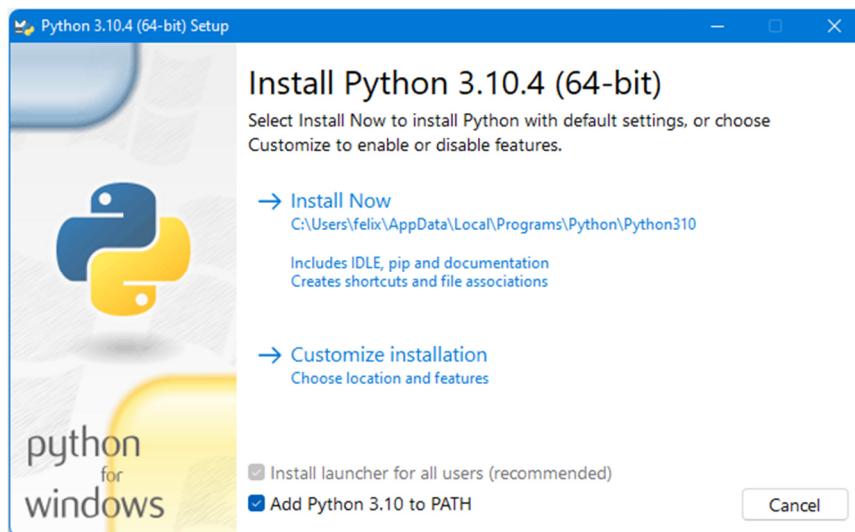


Abbildung 22: Python Installationswizard

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
```

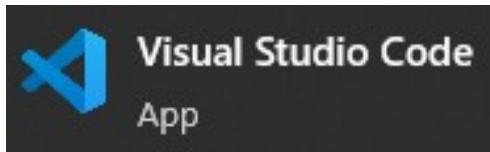
```
C:\>pip install paho-mqtt
Collecting paho-mqtt
  Using cached paho-mqtt-1.6.1.tar.gz (99 kB)
    Preparing metadata (setup.py) ... done
Using legacy 'setup.py install' for paho-mqtt, since package 'wheel' is not installed.
Installing collected packages: paho-mqtt
  Running setup.py install for paho-mqtt ... done
Successfully installed paho-mqtt-1.6.1

C:\>pip list
Package    Version
-----
paho-mqtt  1.6.1
pip        22.0.4
setuptools 58.1.0

C:\>
```

Abbildung 23: Bibliothekinstallation mit PIP

4 Visual Studio Code mit Extensions für Python Entwicklung installieren



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

Abbildung 24: Visual Studio Code Logo

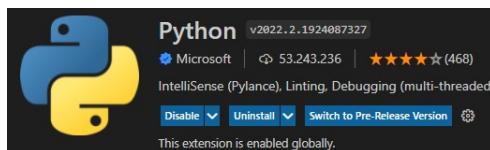


Abbildung 25: Python Extension

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 26: Code Runner Extension

Installieren Sie außerdem die *Code Runner* Extension.

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

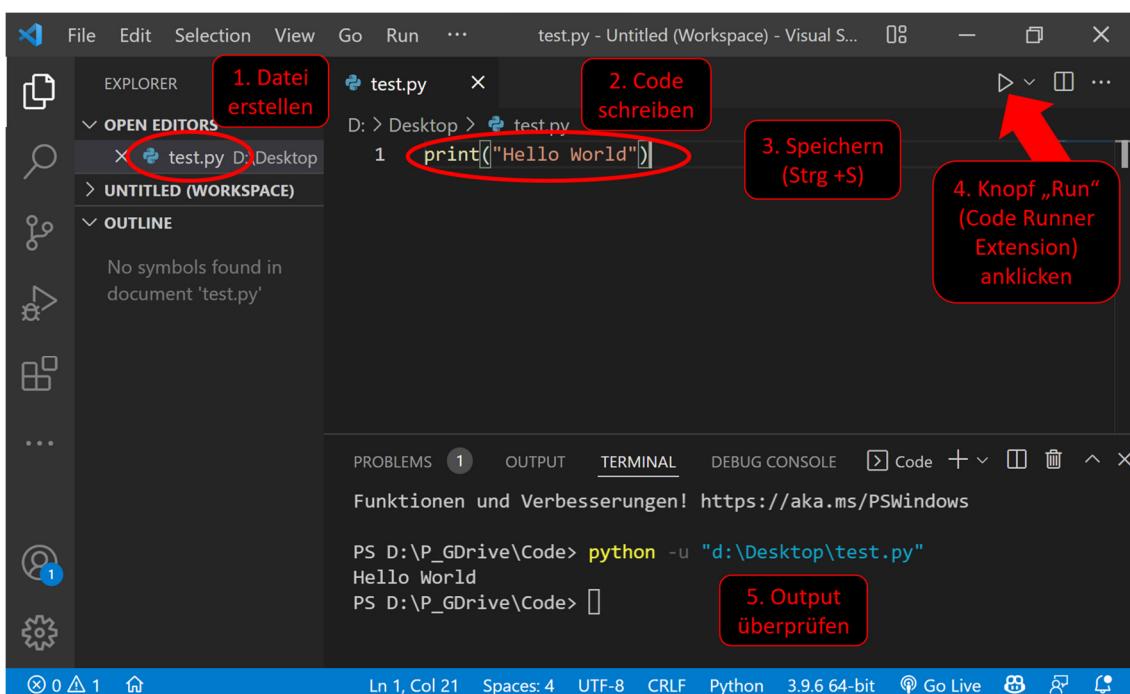
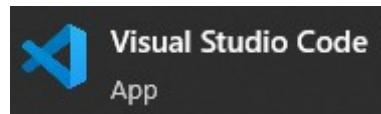


Abbildung 27: Testen der Umgebung via „Hello World“ Programm

5 Vorbereitung von VSCode für Mikrocontrollerprogrammierung

5.1 IDE

Nutzen Sie einen Rechner mit *Windows*, *MacOS* oder *Linux* und installieren sich das kostenlose *Integrated Development Environment (IDE) Visual Studio Code (VSCode)*.



Download → <https://code.visualstudio.com/>

Abbildung 28: Logo von Visual Studio Code

5.2 Extensions für die IDE

Navigieren Sie (links am Rand) in *VSCode* zum Punkt *Extensions* und suchen Sie nach *PlatformIO (PIO)*, installieren Sie diese und alle weiteren Extensions, die Ihnen während der Installation von *PIO* vorgeschlagen werden. (*C/C++*, *IntelliSense*, etc.)

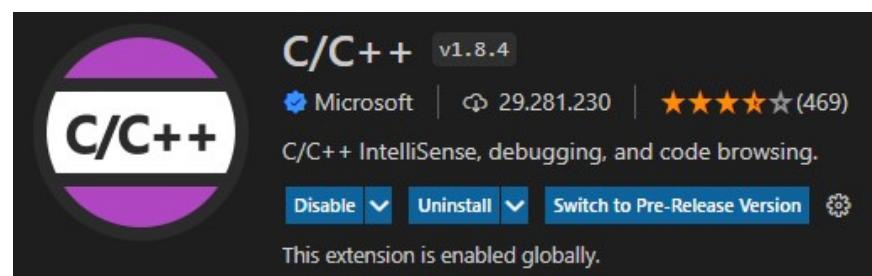
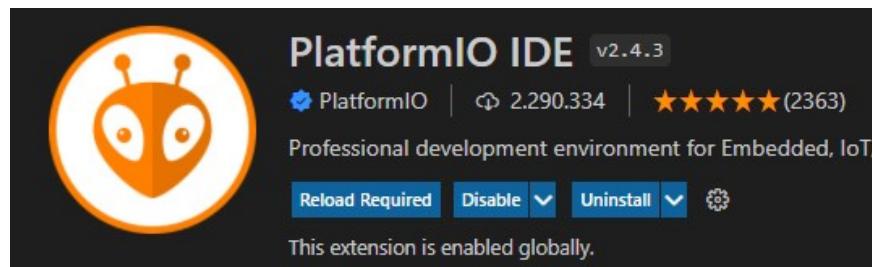


Abbildung 29: Logos der PlatformIO und C/C++ Extensions

6 Hardwaredaten laden

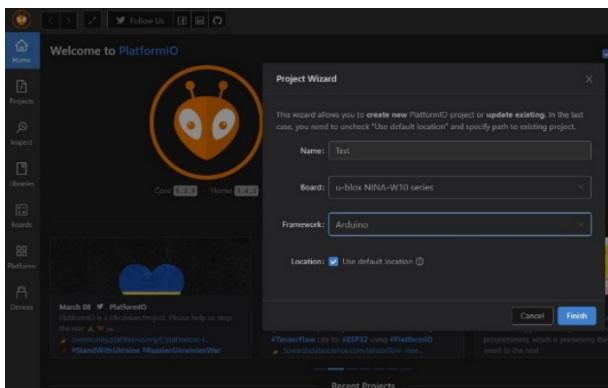


Abbildung 30: PlatformIO Project Wizard

Nach beendeter Installation von *PIO* klicken Sie unten links auf das kleine Haus-Symbol um die Startseite *PIO:HOME* zu öffnen. Dort erstellen Sie über *New Project* ein neues Projekt mit beliebigem Namen. Wählen Sie als Board „*u-blox NINA-W10 series*“ („*Espressif 32*“ Plattform) und bei Framework „*Arduino*“ aus.

```

main.cpp x
testbumms > src > main.cpp > ...
1 #include <Arduino.h>
2 void setup() {
3 | // put your setup code here, to run once:
4 }
5 void loop() {
6 | // put your main code here, to run repeatedly:
7 }

PROBLEMS 6 OUTPUT TERMINAL DEBUG CONSOLE
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [ ] 4.0% (used 13224 bytes from 327680 bytes)
Flash: [ == ] 15.4% (used 201216 bytes from 1310720 bytes)
===== [SUCCESS] Took 1.15 seconds
Terminal will be reused by tasks, press any key to close it.

```

Abbildung 31: erste erfolgreiche Kompilierung

Das Erstellen kann eine Weile dauern, da nun die hardwarespezifischen Daten heruntergeladen werden. Kompilieren Sie anschließend mit dem Haken (*Build*) unten links das gerade erstellte Projekt.

Anmerkung: Dieses Projekt wurde nur erstellt, damit die Hardwaredaten geladen werden, da PlatformIO beim Importieren vorhandener Projekte teilweise Probleme damit hat. Sie benötigen dieses Projekt nach erfolgreicher Kompilierung nicht mehr und können es löschen.

7 Einrichten des Multikopter IoT Projekts

7.1 Vorbereitung

In diesem Praktikum im Rahmen der Vorlesung *Internet of Things* wird der Schwerpunkt auf die Kommunikation bzw. Netzwerkeinbindung gelegt. Zusätzlich werden Sie im Rahmen des Themengebiets *Edge Computing* lokale Logik programmieren. Daher werden alle für Sie im Kontext der Vorlesung nicht relevanten Teile bereits fertig ausprogrammiert sein. Diese haben Sie in Form einer „*Starterprojects*“ als ZIP-Datei bekommen, oder von Github heruntergeladen.

<https://github.com/UniBw-ETTI-2-IoT/DroneLabCourse-Starter.git>

Entpacken Sie diese und verschieben Sie den darin befindenden Ordner an einen beliebigen Ort in Ihrem Dateisystem. Dieser Ordner ist das Projekt.

7.2 Öffnen des Projekts

Öffnen Sie VSCode und gehen Sie auf die Startseite von *PIO* (Haus ganz links unten) und wählen den Punkt *Open Project*. Navigieren Sie IN das Starter Projekt hinein auf Ebene der *platformio.ini* – Datei. Bestätigen Sie nun das Öffnen.

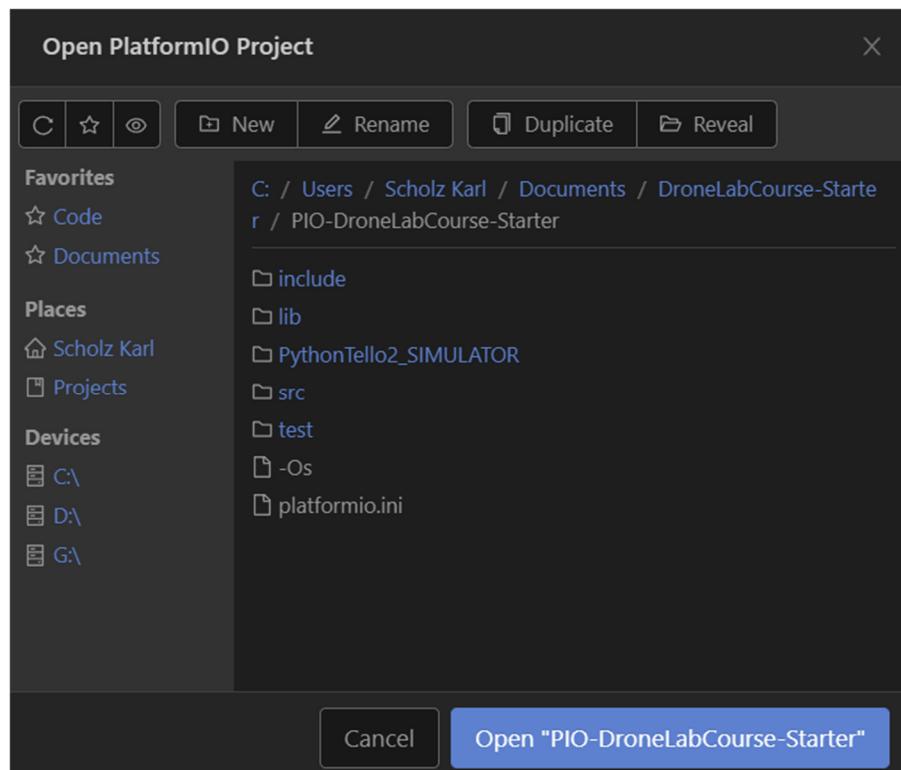


Abbildung 32: PlatformIO: Open Project

7.3 Ordnerstruktur

```

    ▼ PIO-DroneLabCourse-Star...
        > .pio
        > .vscode
    ▼ include
        C background.h
        C main.h
        ⓘ README
    ▼ lib
        > pubsubclient-master
        > RMTT_Libs-master
        ⓘ README
    ▼ PythonTello2_SIMULATOR
        ⚒ SimulatedTello2.py
    ▼ src
        C background.cpp
        C main.cpp
        > test
        E -Os
        ⚙ .gitignore
        📜 platformio.ini

```

Abbildung 33: Ordnerstruktur des Projekts in PlatformIO

Nach öffnen des Projekts öffnet sich links der Dateibaum des Projekts.

In der Datei *platformio.ini* sind die Hardware, sowie Anweisungen für den Compiler definiert. Dies ist die Datei, die auch unter dem Punkt *Anmerkung Debugging* in Teil 2 angesprochen wird, da hier auch die *Baudrate* für die serielle Schnittstelle auf 115200bd gesetzt wurde.

Spezifische Bibliotheken, wie Sie hier für MQTT (*pubsubclient-master*) und die Tello Drohne (*RMTT_Libs-master*) benötigt werden, können in den Ordner *lib* eingefügt werden. Diese können dann im jeweiligen Projekt benutzt werden. Standardbibliotheken wie *Arduino.h* oder *WiFi.h* wurden von *PIO* entsprechend der Hardwaredefinition im Hintergrund bereitgestellt und können einfach über `#include <LibName>` eingebunden werden (vgl. Arduino IDE).

Im *PythonTello2_SIMULATOR* – Ordner befindet sich ein Python-Skript, das in der Lage ist die zweite fliegende Drohne zu simulieren. So können Sie den kompletten Durchlauf auch nur mit einer Drohne testen, sollten Sie hardwaretechnisch beschränkt sein.

Im *src* Ordner befinden sich die Programmdateien des Projekts. Die dazugehörigen header-Dateien befinden sich im *include* Ordner. Für Sie interessant ist nur die Datei *main.cpp*. **Nur dort müssen Sie während des Praktikumstermins etwas bearbeiten.** Sie können trotzdem in alle anderen Dateien, auch die der Bibliotheken, zum Verständnis hineinschauen, auch wenn das für die Aufgabe nicht von Nöten ist. Bitte verändern Sie dort jedoch nichts.

Überprüfen Sie anhand folgender Liste, ob Sie alle Punkte abgearbeitet haben.

- 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 für Python installiert
- Sie haben *VSCODE* mit den Extensions für Mikrocontroller installiert
- Sie haben die Hardwaredaten geladen und erfolgreich leer kompiliert
- Sie haben das Starter-Projekt geöffnet und ohne Veränderung erfolgreich kompiliert

Abbildungsverzeichnis

Abbildung 1: #1 Firewall Einstellungen öffnen	4
Abbildung 2: #2 Eingehende Regeln, Neue Regel	4
Abbildung 3: #3 Port auswählen.....	4
Abbildung 4: #4 TCP und 1883	4
Abbildung 5: #5 Verbindung zulassen.....	5
Abbildung 6: #6 Alles auswählen	5
Abbildung 7: #7 Namen vergeben	5
Abbildung 8: #8 Alle Schritte wiederholen für Ausgehende Regeln	5
Abbildung 9: Mosquitto Installationswizard	6
Abbildung 10: mosquitto.conf	6
Abbildung 11: Windows Systemanwendung "Dienste“	6
Abbildung 12: Mosquitto Broker Dienst	6
Abbildung 13: Port 1883 wird abgehört	7
Abbildung 14: <i>IPv4</i> Adresse in der App „Einstellungen“	7
Abbildung 15: <i>IPv4</i> Adresse in ipconfig /all	7
Abbildung 16: Bearbeitung von mosquitto.conf mit VIM	9
Abbildung 17: Konfigurieren von Mosquitto auf MAC.....	10
Abbildung 18: Sicherheitswarnung von MAC	11
Abbildung 19: MQTT-Explorer Logo	12
Abbildung 20: MQTT-Explorer: Anmeldung beim Broker	13
Abbildung 21: Erste Veröffentlichung einer Nachricht.....	13
Abbildung 22: Python Installationswizard	14
Abbildung 23: Bibliothekinstallation mit PIP	14
Abbildung 24: Visual Studio Code Logo	15
Abbildung 25: Python Extension	15
Abbildung 26: Code Runner Extension	15
Abbildung 27: Testen der Umgebung via „Hello World“ Programm	15
Abbildung 28: Logo von Visual Studio Code	16
Abbildung 29: Logos der PlatformIO und C/C++ Extensions	16
Abbildung 30: PlatformIO Project Wizard.....	17
Abbildung 31: erste erfolgreiche Kompilierung.....	17
Abbildung 32: PlatformIO: Open Project.....	18
Abbildung 21: Ordnerstruktur des Projekts in PlatformIO.....	19

Alle Abbildungen dieses Dokumentes sind Screenshots von Windows, Linux, Mac, Visual Studio Code oder MQTT-Explorer und wurden vom Verfasser erstellt.