Author: Khai Minh Mai
Modul: Embedded C++
Dozent: Prof. Dr.-Ing. Thomas Breuer

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Projekt:

MQTT-Protocol Implementation

Author: Khai Minh Mai
Modul: Embedded C++
Kursleiter: Prof. Dr.-Ing. Thomas Breuer

Author: Khai Minh Mai
Modul: Embedded C++
Dozent: Prof. Dr.-Ing. Thomas Breuer

Aufgabenstellung:

Entwicklung eines Software-Pakets „MQTTclient“ in C++, das das „MQTT-Protokoll“ clientseitig implementiert.
Das MQTT-Protokoll wird vorwiegend im IoT-Bereich genutzt und unterstützt robusten Datenaustausch zwischen Sensor und Aktor-Systemen. Die zu entwickelnde SW-Komponente soll auf dem Mikrocontroller einen MQTT-Client implementieren und eine API für die Anwendungssoftware bereitstellen. Anstelle des vollen Protokollumfangs reicht eine Beschränkung auf das TCP-Transportprotokoll (non secure, Port 1883) und das „Quality of Service Level“ 0 (at most once delivery) aus.

Idee:

- Einen Broker auf einem AWS EC2 Instanz selber hosten
- Der Client auf C++ implementieren

Werkzeuge:

1. Softwares:

- **CodeBlocks**: für Emulation eines virtuellen Mikrocontrollers
- **EmBitz**: für Verbindung mit dem Mikrocontroller
- **MyMQTT Android Client:** für Simulieren eines anderen Clients, und Logging

2. Hardwares:

- **Laptop**
- **Mikrokontroler STM32F769**

3. Externe Services / Bibliotheken:

- **libgcc:** Standard Bibliothek von GNU GCC
- **EmbSysLib:** als Grundlage für die Anwendung
- **Amazon Web Service:** für Hosting des eigenen Brokers
- **OpenSSL:** für Verschlüsselung der Nachrichten

Author: Khai Minh Mai
Modul: Embedded C++
Dozent: Prof. Dr.-Ing. Thomas Breuer

Anforderungen:

ID	Titel	Beschreibung	Vervollständigung	Abhängigkeiten
Technische Grundanforderungen				
TA1	TCP Verbindung	Client kann eine TCP-Verbindung mit Broker erstellen	Completed	-
TA2	MQTT Verbindung (CONNECT)	Client kann eine Verbindung mit Broker mithilfe des MQTT-Protokolls erstellen	Completed	TA1
TA3	CONACK	Client kann erkennen, wenn er eine CONACK Package von dem Broker erhält	Completed	TA2
TA4	SUBSCRIBE	Client kann zu einem beliebigen Topic abonnieren	Completed	TA2
TA5	SUBACK	Client kann erkennen, wenn er eine SUBACK Package von dem Broker erhält	Completed	TA4
TA6	PUBLISH	Client kann Nachrichten zu einem beliebigen Topic veröffentlichen	Completed	TA2
TA7	PUBACK	Client kann erkennen, wenn er eine PUBACK Package von dem Broker erhält	Completed	TA6
TA8	PUBLISH aus anderen Clients	Client kann erkennen, wenn er eine PUBLISH Package von einem anderen Client (durch den Broker) erhält	Completed	TA2
TA9	DISCONNECT	Client kann von dem	Completed	TA2

		Broker auf MQTT Ebene trennen, aber noch auf TCP Ebene mit dem Broker verbunden sein		
TA10	Client aufm Bord	Die Anwendung muss auf dem gegebenen Bord (STM32F769I - DISCO) ausführbar lassen	In progress (wurde angefängt aber wegen Zeitsmangel aufgegeben)	TA2->9
Nichtfunktionale Anforderungen				
NA1	UI	Benutzer kann den Client durch eine interaktive User Interface steuern	Completed	-
NA2	UX	Benutzer bekommt Schritt für Schritt Anleitungen	Completed	NA1
NA3	DX	Die Quellcode ist für andere Developers sichtbar	Completed	-
Zusätzliche Anforderungen				
ZA1	Der Broker wird selbst gehostet	Der selbst gehosteter Broker ist funktionsfähig	Completed	-
ZA2	Authentifikation	Client kann mit Benutzername und Passwort authentifizieren lassen	Completed	ZA1
ZA3	Verschlüsselung (MQTTS)	Client kann mit einem CA Zertifikat + Public Key / Pre Shared Key eine verschlüsselte Verbindung mit dem Broker erstellen	In progress (Broker side - schon fertig; Client side - es fehlt noch kritische Hilfsmethoden)	ZA1
ZA4	QoS Niveau	Client kann ein höheres QoS (Quality of Service) Niveau wählen, damit eine Package nicht verloren werden kann	In progress (Client kann schon QoS 2 Packages aus anderen Clients erhalten)	TA6, TA8

Abweichungen von Grundanforderungen und Begründungen:

- Fast am Ende des Semesters bin ich schon mit der Implementation des Projekts auf dem Windows-Rechner fertig, aber habe ich große Schwierigkeit, dieselbe Funktionalitäten und Vorgehensweisen auf den Bord zu übertragen

    + **Bsp:**
        - NetEth Klasse unterstützt nicht die Kommunikation im öffentlichen Netz, sondern kann man die Pakete nur innerhalb eines privaten lokalen Netzes schicken.

.

        - Es existieren manche Klassen, die in der Windows Umgebung immer global verwendet werden können, sind aber auf dem Bord nicht verfügbar.

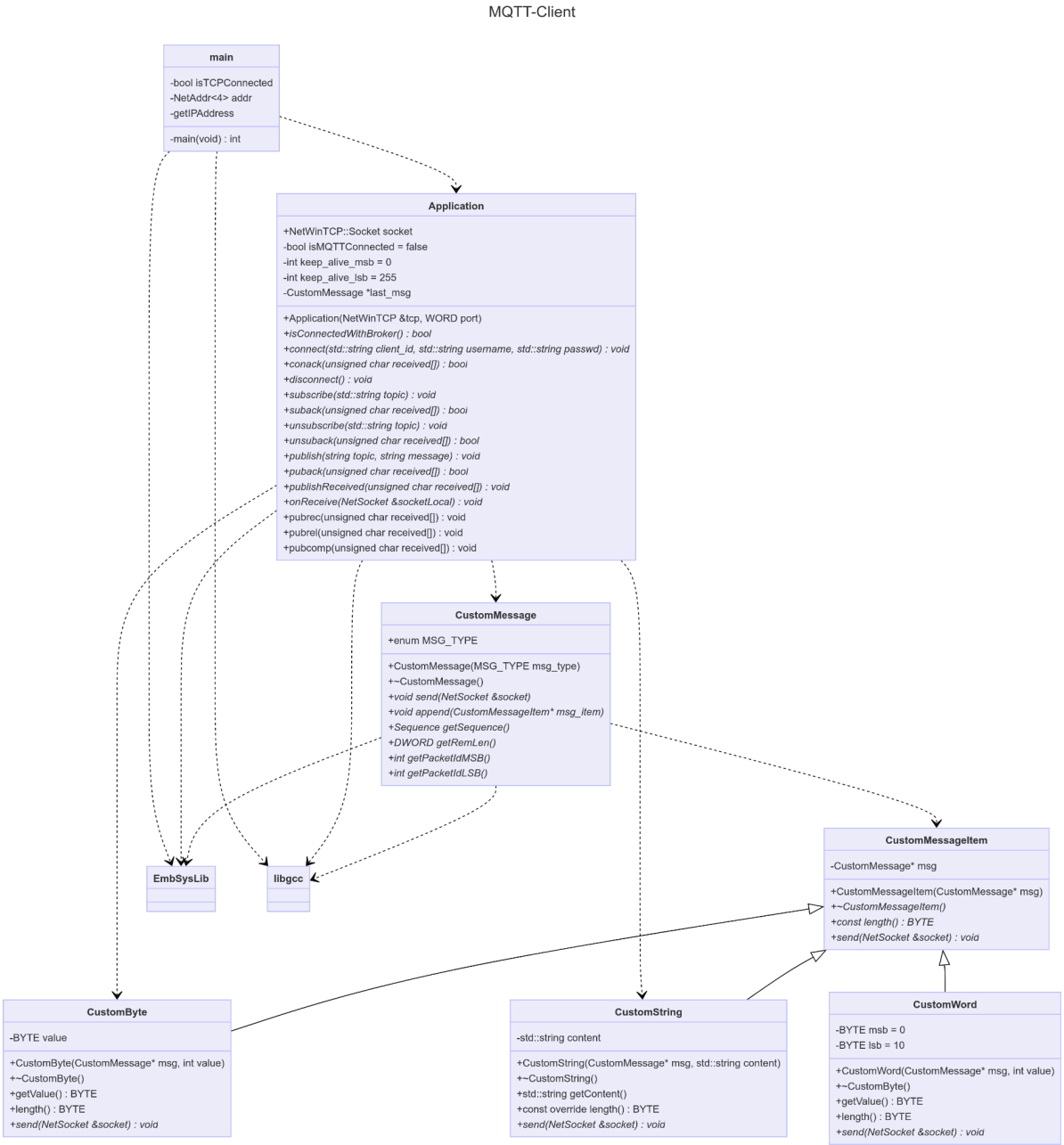
    -> Deswegen habe ich wegen dem Zeitmangel am Ende entschieden, die Übertragung aufzugeben und mit den Dokumentationen anzufangen.

Author: Khai Minh Mai
Modul: Embedded C++
Dozent: Prof. Dr.-Ing. Thomas Breuer

Implementation:

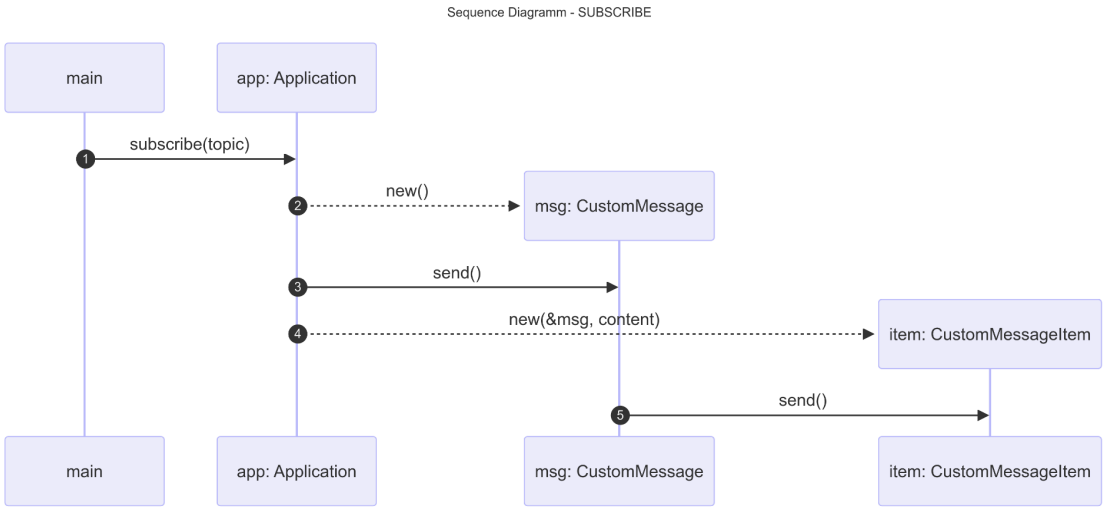
1. Client

1.1. UML Diagramm

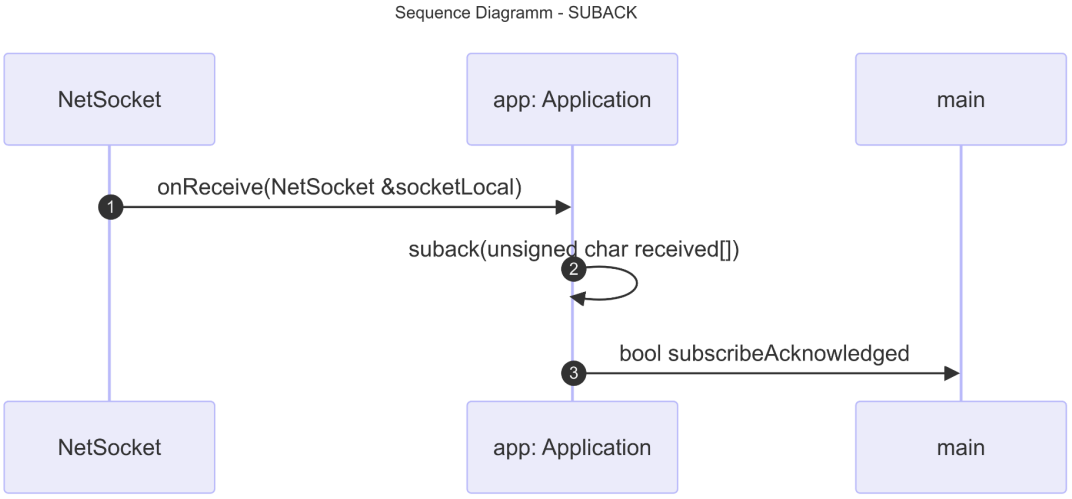


1.2. Sequenzdiagramm

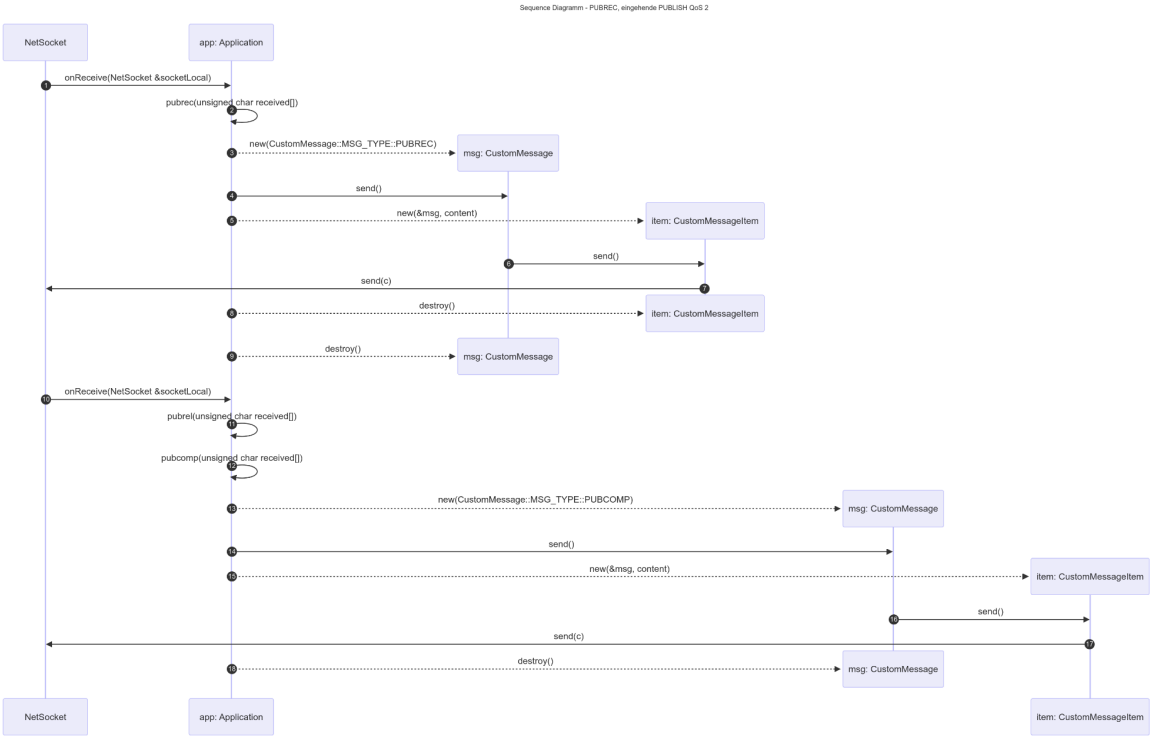
1.2.1. SUBSCRIBE - ausgehende Package



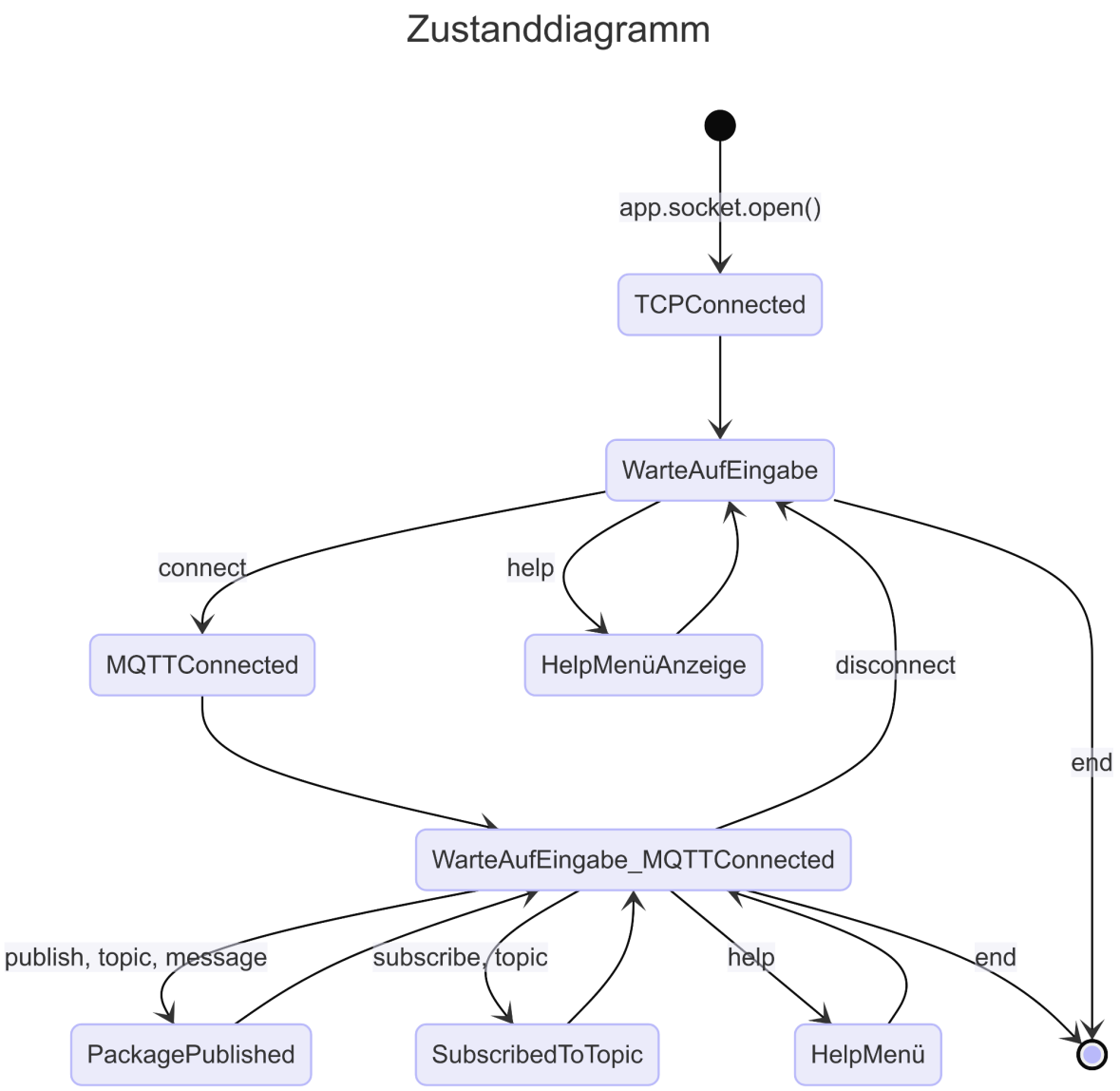
1.2.2. SUBACK - eingehende Package



1.2.3. PUBREC - für Erhalten eines PUBLISH Package mit QoS-Niveau 2



1.3. Zustandsdiagramm

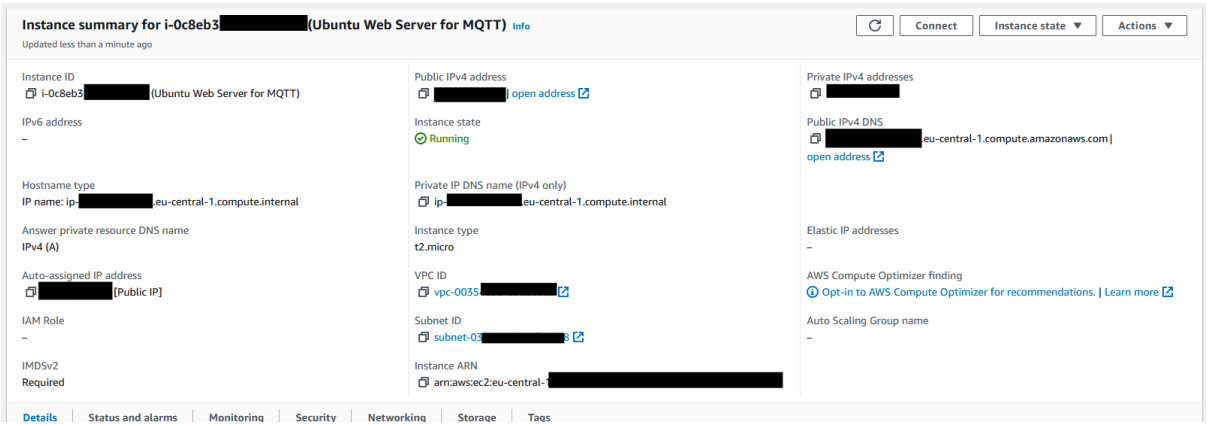


2. Broker

2.1. Die AWS EC2 Instanz

Author: Khai Minh Mai
Modul: Embedded C++
Dozent: Prof. Dr.-Ing. Thomas Breuer

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences



2.2. Konfigurationen

```
# Default port
listener 1883

# Logging
connection_messages true
log_type all
log_dest topic

# Encrypted port
listener 8883
cafile /etc/mosquitto/conf.d/ca.crt
certfile /etc/mosquitto/conf.d/server.crt
keyfile /etc/mosquitto/conf.d/server.key
tls_version tlsv1.2

# Password
allow_anonymous true
password_file /etc/mosquitto/conf.d/passwd
```

Vorgehensweise:

1. Aufgabenstellung durchlesen, über das Thema recherchieren

1.1. Aufgabenstellung:

- https://lea.hochschule-bonn-rhein-sieg.de/goto.php?target=file_1539542_download&client_id=db_040811

1.2. Referenz für die Recherche:

- Offizielle Dokumentationen von MQTT Protokol: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718090
- Dokumentationen von AWS: https://aws.amazon.com/what-is/mqtt/
- Mosquitto man page: https://mosquitto.org/man/mosquitto-8.html
- Beispiele über Implementationen von MQTT:
  + https://codepr.github.io/posts/sol-mqtt-broker/
  + https://medium.com/gravio-edge-iot-platform/how-to-set-up-a-mosquitto-mqtt-broker-securely-using-client-certificates-82b2aaaef9c8
  + https://github.com/kurdybacha/mqttcpp

2. Implementierung

2.1. Versuchen, die aufgelisteten Ideen in C++ zu implementieren

- War mit der Einstellung des Brokers erfolgreich, aber war noch über die Vorgehensweise mit dem Client verwirrt - Es war mir unklar, wie ich die Bibliothek “EmbSysLib” ausnutzen konnte.

2.2. Mit Kommilitonen und Prof. Breuer zu diskutieren

- Die meisten meiner Kommilitonen waren auch darüber leider ahnungslos, da das Thema etwa “niche” war.
- Prof. Breuer hat mir viel dabei geholfen, indem für mich die größte Hilfe war, dass ich jetzt verstehe, was die Bibliothek EmbSysLib tun kann, und wie ich diese Bib ausnutzte.

2.3. Wieder versuchen, mit den neuen Informationen (und Hilfen) die Ideen zu implementieren

- War mit den gegebenen Bibliotheken erfolgreich, den Client zu implementieren. Ich hatte aber Schwierigkeiten, dieselbe Ideen auf dem Bord in der gegebenen Zeitsmenge überzutragen.
- Habe die Entscheidung getroffen, die Anwendung auf Windows zu vervollständigen, statt versuchen, eine fehlerhafte Anwendung auf dem Bord zu haben.

3. Dokumentierung

3.1. Anforderungsanalyse

- Angeforderte Technische / Funktionale Anforderungen (steht in der Aufgabenstellung)
- Nicht funktionale Anforderungen (Benutzerfreundlichkeit)
- Zusätzliche technische Anforderungen (um die Anwendung zu verfeinern)

3.2. Anwendungsarchitekturanalyse

- UML Diagramm
- Sequenzdiagramm (ausgehende Packages, eingehende Packages, und PUBREC für eingehende Packages mit QoS 2)

Kritiken:

1. Zeitmanagement

- Zu viel Zeit für unnötige Sachen investiert haben.

2. Noch hohes Niveau von Abhängigkeit

- Ich muss noch viele Hilfen von Prof. Breuer anfragen

3. Codes noch nicht “Industriestandard”

- Noch zu überfüllte Klassen statt mehreren sichtbaren Klassen
- Codes noch nicht vollständig “sauber”

Potentielle Weiterentwicklungsrichtungen:

1. Sicherheitsmaßnahmen

- Vervollständigung der noch fehlenden Sicherheitsmaßnahmen, um eine verschlüsselte Verbindung zwischen Client und Broker sicherzustellen (TLS / PSK / Verbindung durch VPN).
- Dafür werde ich die NetSocket Klasse selber erweitern müssen.

2. Übertragung auf dem Bord

- Übertragung der Funktionalitäten an Bord STM32F769I DISCO.

3. UI verschönern

- Eine vollständige Windows MQTT Client kann ich auf diesem schon aufgebauten Grundlage weiterentwickeln.

Fazit:

Durch die Module und das Projekt konnte ich viel über C++, Embedded Systems, MQTT - Protokoll, und in weiterer Sinne, Messaging Protokolls, die auf TCP/IP basieren.
Das Projekt war im Überblick erfolgreich, indem ich am Ende ein funktionales Endprodukt habe, trotz der noch fehlenden Anforderungen.
Die Quellcodes passen zu dem Ziel des Moduls, da ich dabei viel über Themen wie C++, Objektorientierte Programmierung, Embedded Systems, Pointers, Dateneingabe & -ausgabe und Kommunikation über TCP/IP üben konnte.