Smartbit

Marius Cerwenetz

Institut für Softwaretechnik und Datenkommunikation

2022–07–13

Agenda

Einführung

Schwierigkeiten beim Programmierenlernen

Problem ≠ Zielgruppe

- Programmier-Neulinge
- Grundkonzepte Bsp.?
- Theoretische Übungsaufgaben

Microcontroller als Alternative

- Interaktive Aufgaben
- Ausgabemöglichkeiten (LEDs, Piepser, Aktoren)
- Ausprobieren

Nachteile?

Vorteile Smartphones gegenüber Microcontroller-Schaltungen

- Hohe Verfügbarkeit
- Keine Verdrahtungsfehler
- Unabhängige Spannungsversorgung
- Zahlreiche Sensoren bereits integriert
- Drahtlose Verbindungstechnologien (WLAN, Bluetooth, UMTS/LTE)
- Zweckgebundene Ausgabemöglichkeiten

Problemstellung

Probleme

- Anbindung in Programmierumgebungen
- Smartphone-App

Lösungsansatz

Software-Lösung zur Kommunikation mit dem Smartphone

Anforderungen

Anforderungen

Aufgaben fehlen!

↓ das macht allein gar keinen Sinn

- UI-Elemente als Ausgabe
- Sensordatenübermittlung
- Geringe Latenzen
- Sicherheit

Smartbit

Aufbau der Smartbit-Lösung

Bibliothek

Middleware

Android-App

Schnittstelle zum Programmcode

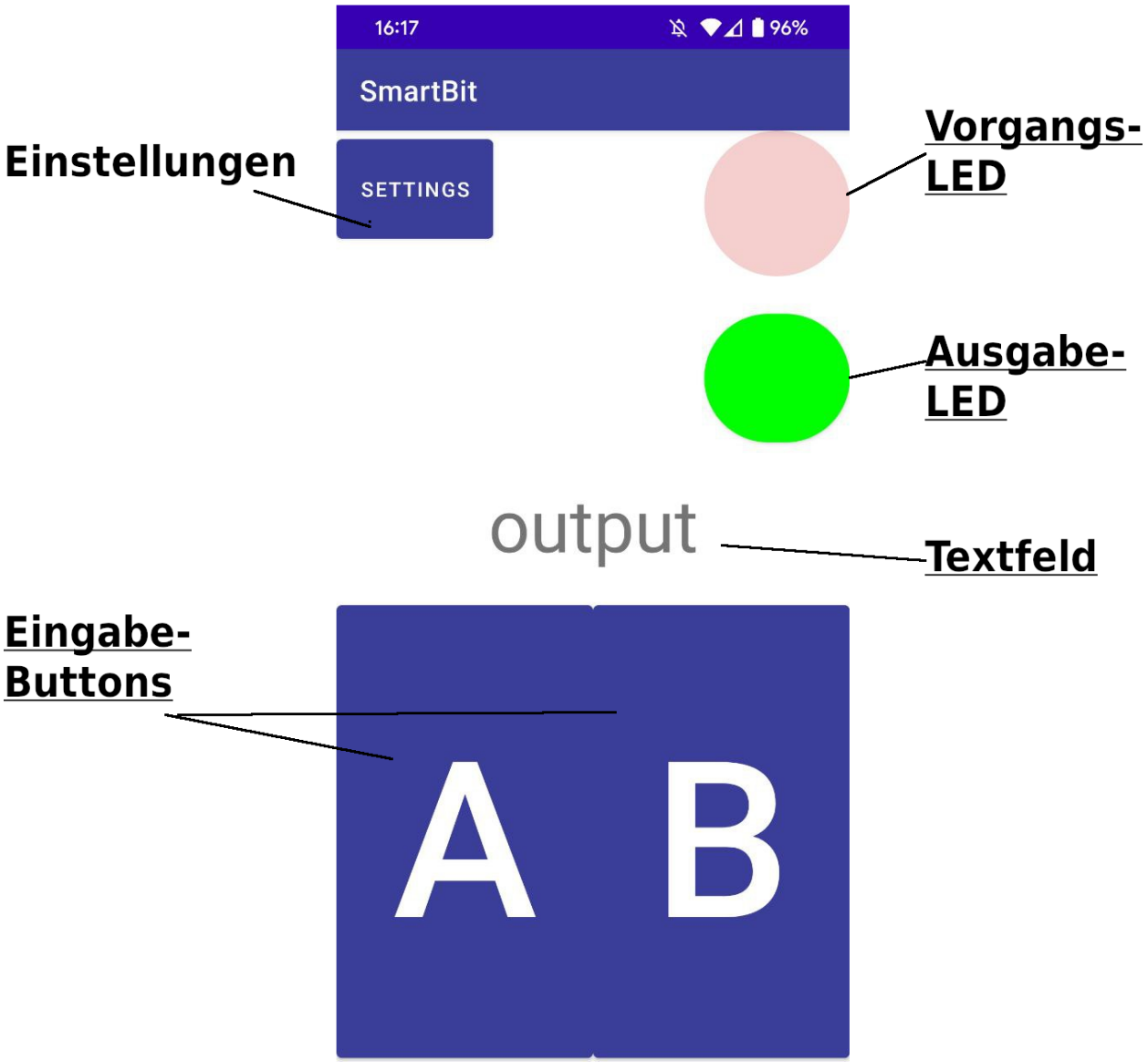
Funktionsaufrufe für Sensorwerte und Ausgaben.

Einbindbar in:

- C
- Java
- Python

```
1 from  smartbit import Phone
2 p = Phone()
3 accel = p.get_x_accelo()
4 p.write_text("hallo")
5
```

Smartphone-App

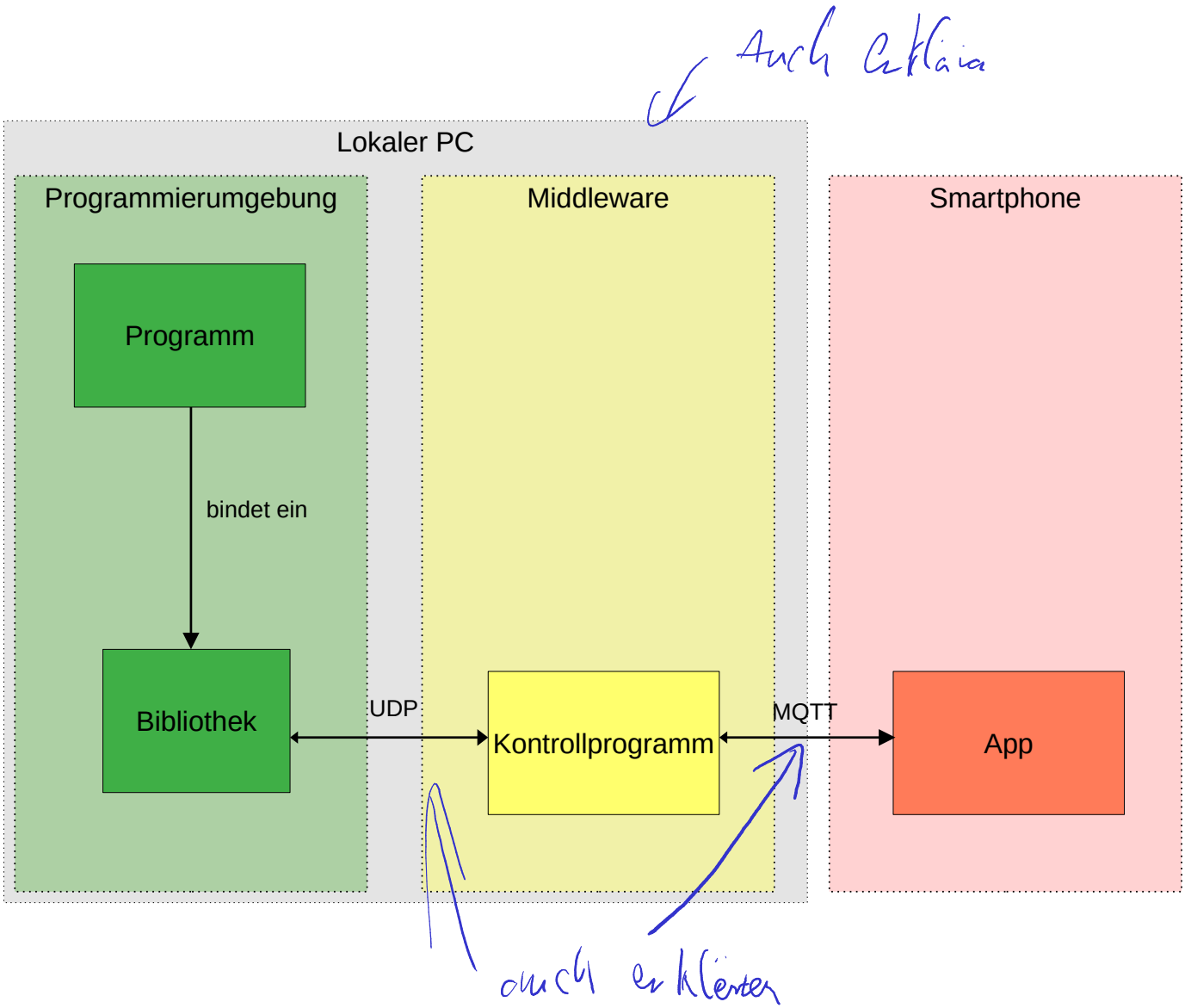


Middleware

```
swt@pb22:~/thesis01/software$ python3 ./server.py
INFO:MqttHandlerThread:trying to connect to mqtt server
INFO:MqttHandlerThread:connected to server pma.inftech.hs-mannheim.de
INFO:MqttHandlerThread:mqtt subscribed to topic: 22thesis01/test
```

Architektur

Architektur



Nachrichtenaustausch

Zwecke:

- Neuer Sensorwert
- Sensorwert-Anfrage
- Sensorwert-Antwort
- RPC-Anfrage
- RPC-Antwort

Richtige Namen und Zweck erklären

Konfigurationsdateien

- protocol.json
- config.json
- Dezentral auf alle drei Komponenten verteilt

Beispielnachricht - Neuer Sensorwert

```
{
    "type": "update_request",
    "sensor_type": "",
    "sensor_value": ""
}
```

Funktionsweise Bibliotheken

Zweck? Was macht die?

- Stub-Funktionen ← v(t) bsp
- Socket ← eigentlich egal
- Local-Loopback
- UDP, Port 5006 ← counterpart in Kontrollanwendung
- CJSON und JSON for Java

Funktionsweise der Smartphone-App

SensorEventListener

- Neue Sensormessdaten
- Übermittlung an Kontrollanwendung

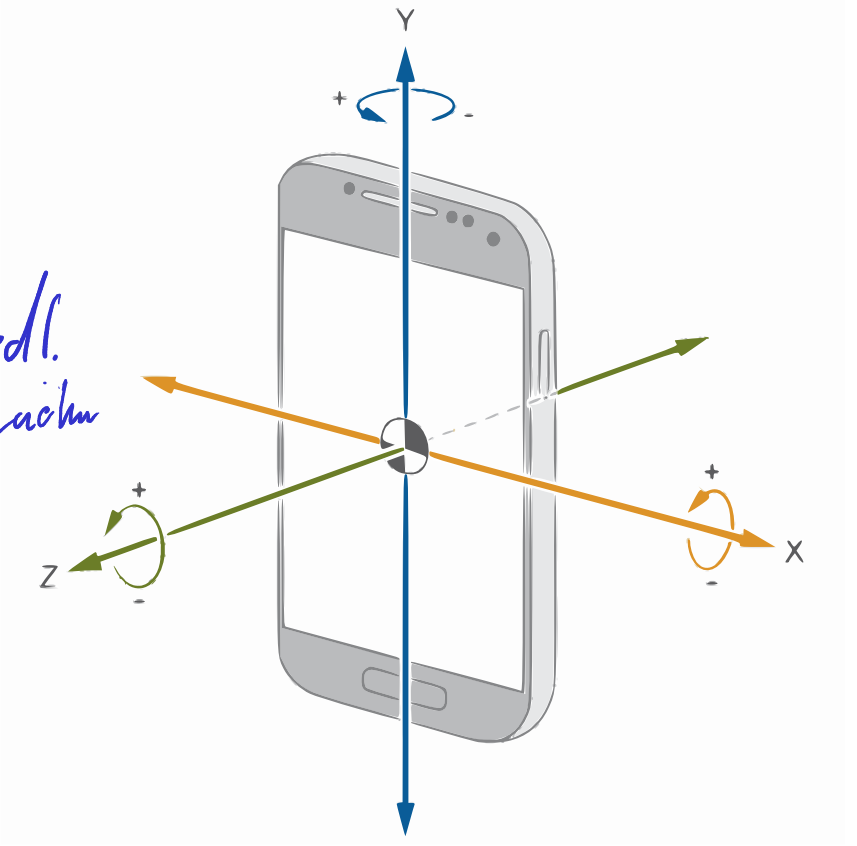
MessageListener

- Warten auf Nachrichten
- Reaktion Ausgabe
- Ggf. RPC-Antwort

Sensoren

- Beschleunigungssensor
- Gyroskop
- Annäherungssensor

unterschiedl. Einheiten nachm



Allgemein

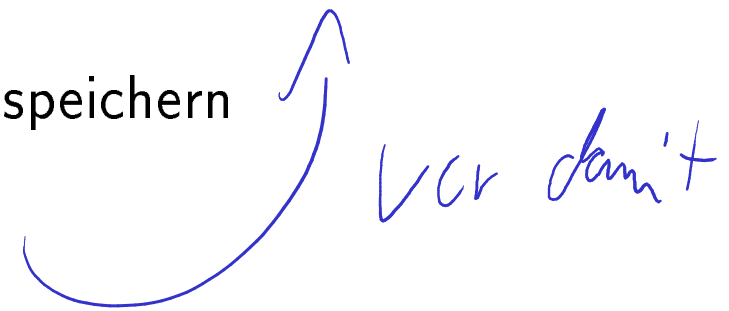
Zweck? Name? Um was geht's?

- Mehrere Threads
- Kommunikation per MQTT und UDP
- Thread-Kommunikation durch Python-Queues

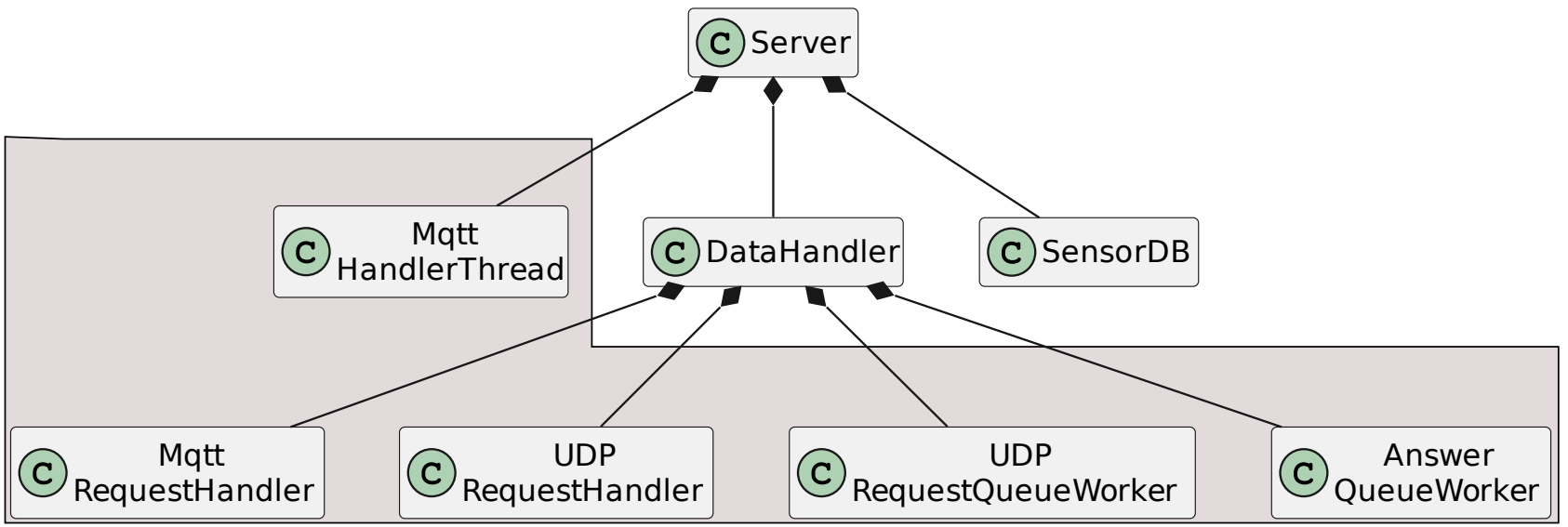
Funktionsweise der Middleware

Aufgaben:

- Sensormesswerte annehmen und speichern
- Sensor-Anfragen beantworten
- RPC-Anfragen weiterleiten
- RPC-Antworten weiterleiten

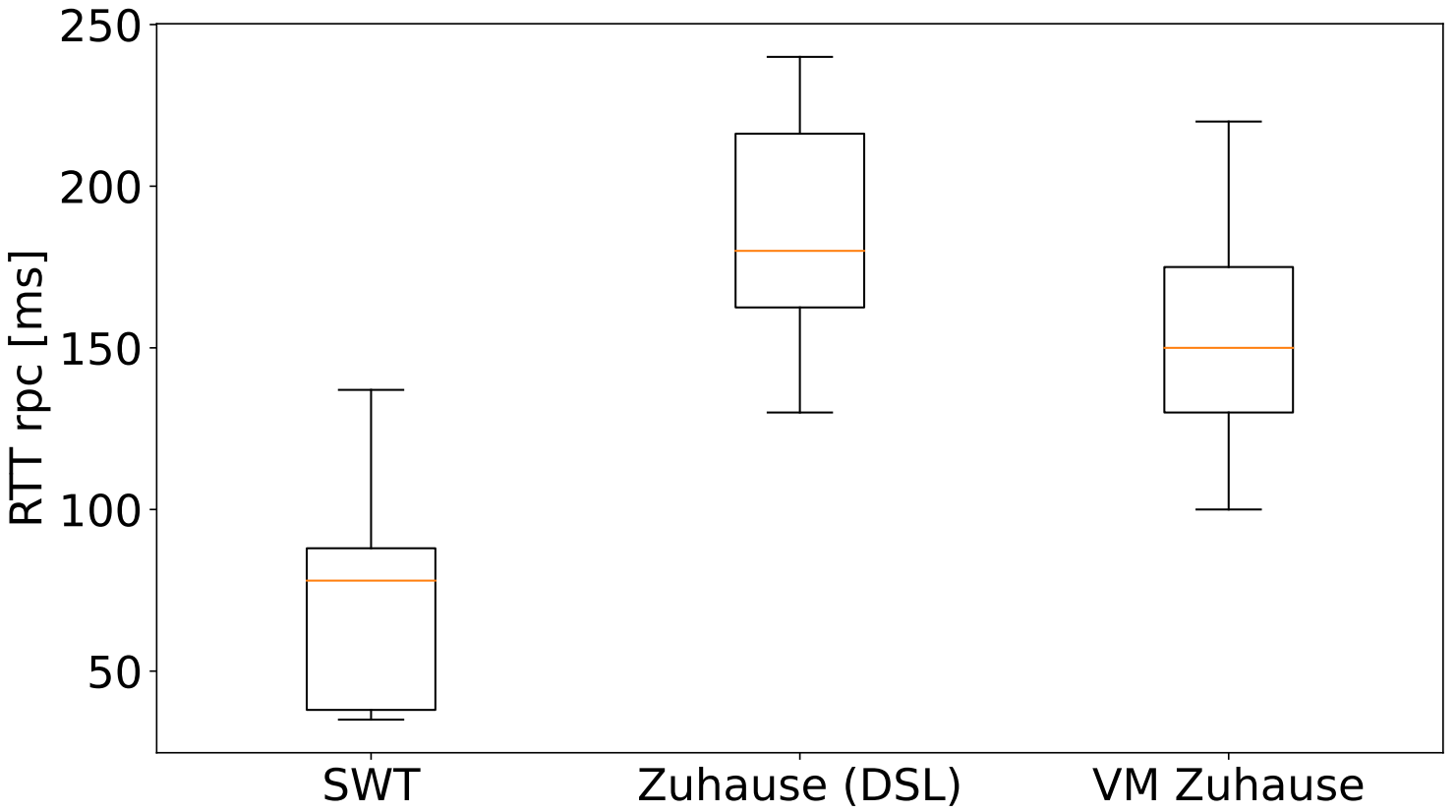


Aufbau der Kontrollanwendung



Evaluation

Latenzen



Sonstiges

- Portabilität
- Latenzverschleierung durch Caching
- ~~TLS-Verschlüsselung MQTT~~
- Transparenter Ablauf durch Logging

Fazit und Ausblick

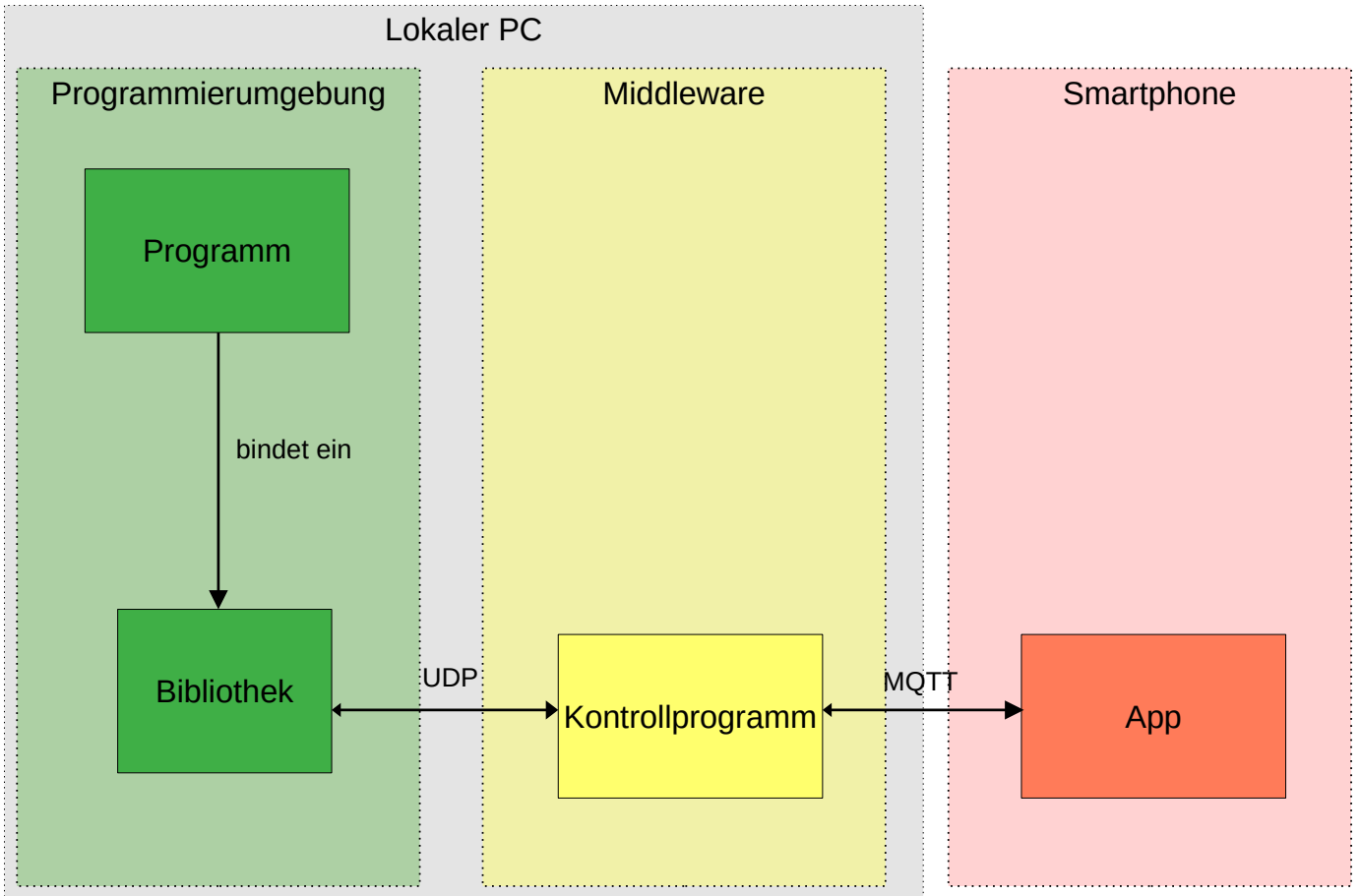
Fazit

- Alle Anforderungen implementiert
- JSON-Parser
- Latenzzeiten tolerabel
- Objekt-Orientierung
- Verfügbarkeit

Ausblick

- QR-Code für Konfigurationen
- Mehrere Sensorwerte zwischenspeichern
- Burst-Mode / Selektive Sensormessung
- Group-Sessions

Zusammenfassung



Demo