

IoT-Labor: Smart Lock

Dokumentation

Bachelor of Science

des Studiengangs Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Tom Freudenmann, Maximilian Nagel, Marcel Fleck

26.04.2023

Bearbeitungszeitraum
Matrikelnummern, Kurs
Dozent

10.03. - 26.04.2023
6378195, 7362334, 9611872, INF20D
Hartmut Seitter

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich meine Dokumentation mit dem Thema: *IoT-Labor: Smart Lock* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Stuttgart, 26.04.2023

Tom Freudenmann, Maximilian Nagel, Marcel Fleck

Inhaltsverzeichnis

Abkürzungsverzeichnis	III
Abbildungsverzeichnis	IV
1 Einleitung	1
2 Architektur	2
2.1 Device-Layer	3
2.2 Network-Layer	3
2.3 Service-Layer	4
2.4 Application-Layer	5
3 Ausblick	6

Abkürzungsverzeichnis

BLE	Bluetooth Low Energy
GPS	Global Positioning System
HTTP	Hypertext Transfer Protokoll
IoT	Internet of Things
JSON	JavaScript Objective Notation
LED	Light Emitting Diode
LoRa	Long Range (Low Power)
LoRaWAN	Long Range Wide Area Network
MQTT	Message Queuing Telemetry Transport
TTN	The Things Network

Abbildungsverzeichnis

2.1	Architektur-Diagramm des Smart-Locks	2
2.2	Node-Red Serverarchitektur für das Smart-Lock	5

1 Einleitung

TODO: Hier anfangen zu schreiben: Buisinesscase

2 Architektur

Die entwickelte Internet of Things (IoT)-Lösung Smart Lock teilt sich in vier verschiedene Layer auf: Device-, Network-, Service- und Application-Layer. Abbildung 2.1 beschreibt den architektonischen Aufbau der Lösung genauer und enthält die verschiedenen Hardware und Software Module der Lösung.

Anhand dieser Abbildung beschreiben die folgenden Abschnitte, den architektonischen Aufbau der Komplettlösung. Hierbei wird auf die jeweiligen Architektur-Layer eingegangen und erklärt, welche Eigenschaften die abgebildeten Geräte, Protokolle und Software im Rahmen der IoT-Lösung mit sich bringen.

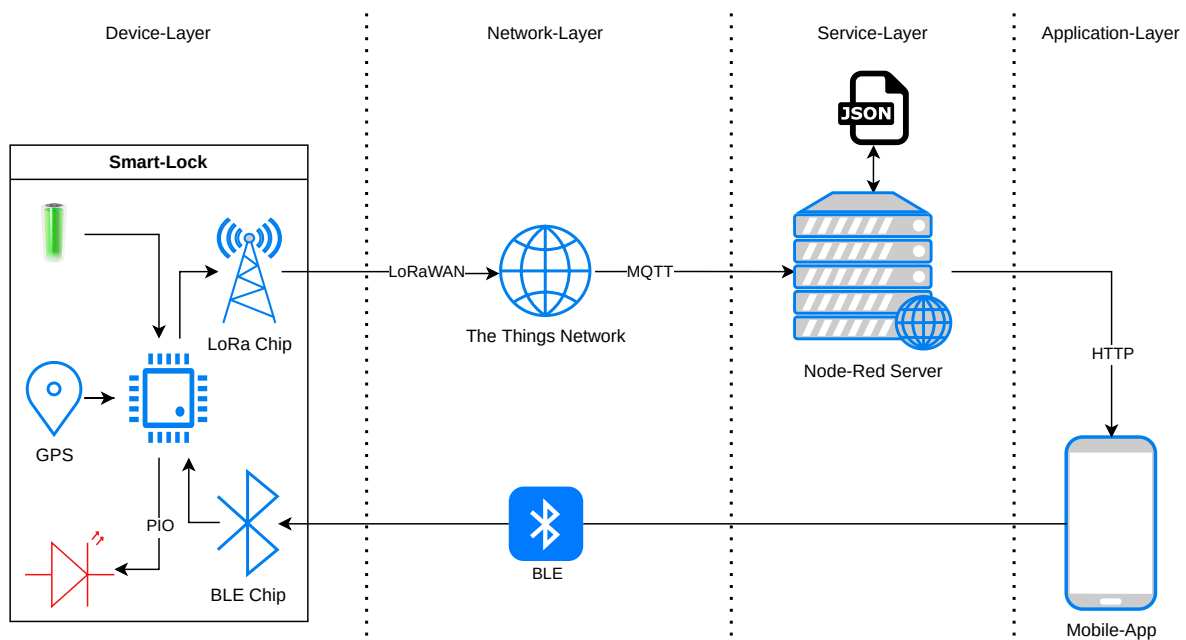


Abbildung 2.1: Architektur-Diagramm des Smart-Locks

2.1 Device-Layer

Im Device-Layer finden sich alle Sensoren und Aktoren der IoT-Lösung wieder. Zu den Sensoren gehört ein Global Positioning System (GPS)-Chip, der GPS-Daten empfängt, um den Standort des Smart-Locks festzustellen. Eine Light Emitting Diode (LED), die den Zustand des Smart-Locks, also ob geschlossen oder offen, darstellt, gehört zur Gruppe der Aktoren.

Des Weiteren befinden sich zwei Netzwerkschnittstellen in Form zwei gesonderter Chips im Device-Layer der IoT-Lösung. Ein Long Range (Low Power) (LoRa)-Chip dient dem Senden und dem Empfangen von LoRa-Nachrichten mit Hilfe von Long Range Wide Area Network (LoRaWAN) an das The Things Network (TTN). Ein Bluetooth Low Energy (BLE)-Chip ermöglicht eine Verbindung mit einem Mobilgerät und dient dem Empfangen von Befehlen, die den Zustand des Smart-Locks ändern können. Diese Netzwerkprotokolle werden in Kapitel 2.2 genauer beschrieben.

2.2 Network-Layer

Das Network-Layer beschreibt die Protokolle und Netzwerk-Server, die die IoT-Lösung zum Austausch der Daten verwendet. Dabei wird in der vorgestellten Lösung auf vier verschiedene Übertragungstechnologien gesetzt.

Für die Übertragung der GPS-Daten wird **LoRaWAN** verwendet und sendet die Daten an ein Gateway. Das Gateway ist Teil des TTNs, ein Zusammenschluss aus vielen öffentlichen LoRaWAN-Gateways. Im TTN werden die Daten über einen Parser in ein JavaScript Objective Notation (JSON)-Format gebracht und können über herkömmliche Netzwerkprotokolle, wie Hypertext Transfer Protocol (HTTP)-WebSockets oder Message Queuing Telemetry Transport (MQTT) an einen Server übertragen werden.

Die vorgestellte Smart Lock Lösung verwendet für die weitere Übertragung an das Service-Layer **MQTT**. Das hat den Vorteil, dass sich Nachrichten sobald sie empfangen werden, an den Server übertragen lassen und nicht über ein Request-Response Verfahren abgefragt werden müssen.

Zusätzlich zu LoRaWAN und MQTT wird in der Lösung **HTTP** für die Kommunikation von App und Server verwendet. Hierfür sendet die Benutzer-App eine HTTP-Request an den Node-Red Server, der die Anfrage verarbeitet und die Device-Daten an die App zurücksendet.

Zuletzt wird **BLE** für das Öffnen und Schließen des Schlosses und somit für die direkte Kommunikation von App zu Device verwendet. BLE hat zwei entscheidende Vorteile: geringer Energieverbrauch und geringe Reichweite. Ersteres ist notwendig, damit das Schloss möglichst lange Akkulaufzeiten hat und zweiteres damit das Schloss nur aus unmittelbarer Nähe geöffnet oder geschlossen werden kann.

Zusammengefasst, deckt das Network-Layer die Kommunikation von allen verwendeten Hardware- und Software-Modulen ab und verwendet hierfür LoRaWAN über das TTN als Gateway zu MQTT, sowie HTTP und BLE für die Kommunikation zur Benutzer-App.

2.3 Service-Layer

Das Service-Layer besteht aus einem Node-Red Server, der in einem Docker-Container läuft. Das hat den Vorteil, dass die Anwendung beliebig umgezogen oder skaliert werden kann. Zusätzlich empfängt der Server die Daten aus dem TTN über MQTT und speichert sie local in der *Context*-Variablen und persistent als JSON-Dokument ab. Dadurch können die Daten auch nach einem Neustart der Anwendung weiter verwendet werden. Abbildung 2.2 zeigt die folgenden drei Prozesse in Node-Red:

- **Initialisierung:** Der erste *Flow*, der beim Serverstart einmalig ausgeführt wird. Er lädt das gespeicherte JSON-Dokument und setzt die *Context*-Variablen auf die gespeicherten Werte.
- **Verbindung zum TTN-Server:** Dieser Prozess wird für jede im TTN empfangene und weitergeleitete Nachricht ausgeführt und speichert die empfangenen GPS- und Sensor-Informationen im JSON-Dokument bzw. Context des Servers.
- **App Anfrage:** Für jede empfangene HTTP-Anfrage an die Route *server-adresse:1880/device-id* wird der *Flow* ausgeführt. Er lädt die Daten für die mitgelieferte *device-id* und schickt sie als Antwort zurück an die App.

Außerdem wird der Server in der Azure-Cloud, auf einem Ubuntu-Server gehostet. Dadurch ist der Server aus dem Internet zugänglich und die Daten können jeder Zeit vom Benutzer abgerufen werden.

Zusammengefasst, deckt der Server das Service-Layer mit persistenter Speicherung ab, indem er Nachrichten aus dem Network-Layer empfängt und an das Application-Layer weiterleitet. Zusätzlich werden die Daten im Server gespeichert, um sie jeder Zeit abfragen zu können, auch wenn der Server neu gestartet wurde. In Zukunft lässt sich das System

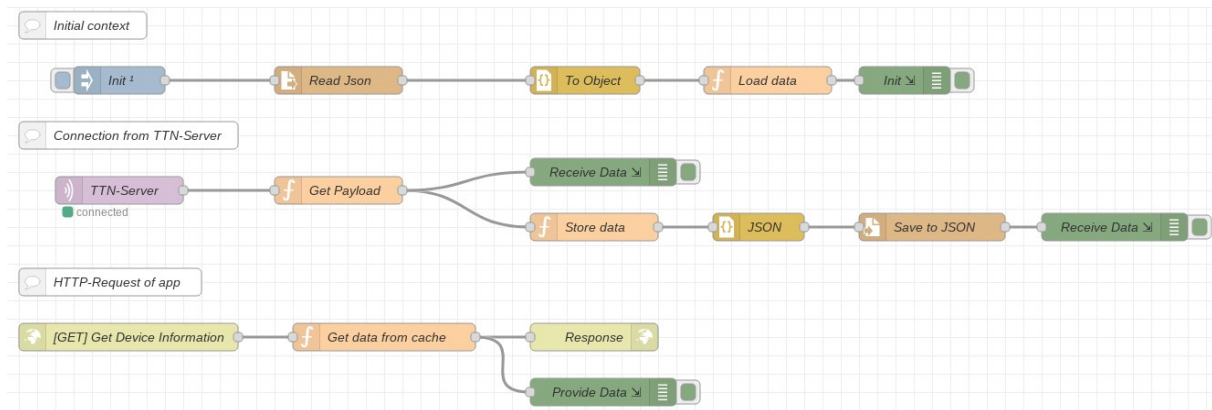


Abbildung 2.2: Node-Red Serverarchitektur für das Smart-Lock

weiter skalieren oder auf einem eigenen Server hosten, da es in einem Docker-Container läuft.

2.4 Application-Layer

TODO: Hier anfangen zu schreiben

3 Ausblick

TODO: Hier anfangen zu schreiben