

## Einführung in IoT

Modulübergreifender Kurs Vertiefung II

## **Impressum**

5200 Einführung in IoT

Herbstsemester 2023, 2024

Institut Geomatik, Fachhochschule Nordwestschweiz, Hofackerstrasse 30, 4132 Muttenz

Autorin: Prof. Dr. Pia Bereuter

Einführung in IoT © 2024 von Pia Bereuter ist lizenziert unter [CC BY-NC 4.0](#)

Titelbild: Raspberry Pi mit Sensoren, 2022, Bereuter

# Inhaltsverzeichnis

<b>Inhalt</b>	<b>1</b>
Aufbau . . . . .	1
Kursvorbereitung . . . . .	2
Raspberry Pi Image Kontoinformationen . . . . .	2
Übungsunterlagen . . . . .	3
Repository . . . . .	3
<b>I. Theorie</b>	<b>4</b>
<b>1. Internet of Things IoT</b>	<b>5</b>
1.1. Ubiquitous Computing . . . . .	5
1.2. Internet of Things . . . . .	6
1.3. Digital Earth und IoT . . . . .	8
1.4. IoT und GIS . . . . .	11
<b>2. Sensoren</b>	<b>15</b>
2.1. Sensoren in IoT . . . . .	15
2.2. Sensoren . . . . .	16
<b>3. Datenübertragung</b>	<b>19</b>
3.1. Kommunikation . . . . .	20
3.2. MQTT - Message Queuing Telemetry Transport Protocol . . . . .	21
3.3. Cloud, Edge und Fog . . . . .	22
<b>Literatur</b>	<b>23</b>
<b>II. Praktika</b>	<b>26</b>
<b>4. Luftqualitätmessung</b>	<b>27</b>
4.1. Einführung . . . . .	27
4.2. BME688 . . . . .	28
4.3. Übungsaufbau . . . . .	29
4.4. Aufgabe 1: Sensormessungen ausführen . . . . .	31
4.5. Aufgabe 2: Berechnung der Atmosphärenkorrektur für Distanzmessungen (optional) . . . . .	33
<b>5. Spektralmessung</b>	<b>36</b>
5.1. Einführung . . . . .	36

5.2. Spektrometer . . . . .	37
5.3. AS7262 Multispektralsensor . . . . .	38
5.4. Übungsaufbau . . . . .	38
5.5. Aufgabe 1: Spektralmessungen durchführen . . . . .	39
5.6. Aufgabe 2: Spektralmessungen mit Visualisierung in der Konsole . . . . .	40
<b>6. Bewegungsmessung</b>	<b>42</b>
6.1. Einführung . . . . .	42
6.2. Beschleunigungssensoren IMU . . . . .	43
6.3. ICM20948 9DoF Motion Sensor . . . . .	44
6.4. Übungsaufbau . . . . .	45
6.5. Aufgabe 1: Bewegungsmessungen durchführen . . . . .	45
6.6. Aufgabe 2: Magnetometer . . . . .	46
<b>7. Distanzmessung</b>	<b>48</b>
7.1. Einführung . . . . .	48
7.2. VL53L5CX 8x8 Time of Flight (ToF) Array Sensor . . . . .	49
7.3. Übungsaufbau . . . . .	50
7.4. Aufgabe 1: Distanzmessung Konsole . . . . .	51
7.5. Aufgabe 2: Distanzmessung mit LCD Bildschirm . . . . .	53
<b>8. Thermale Aufnahmen</b>	<b>55</b>
8.1. Einführung . . . . .	55
8.2. MLX90640 Thermalkamera . . . . .	56
8.3. Übungsaufbau . . . . .	57
8.4. Aufgabe 1: Punktuelle Temperaturmessung . . . . .	58
8.5. Aufgabe 2: Thermale Aufnahme mit LCD Bildschirm . . . . .	58
<b>9. Biometrie</b>	<b>61</b>
9.1. Einführung . . . . .	61
9.2. MAX30101 Breakout Heart Rate, Oximeter, Smoke Sensor . . . . .	62
9.3. Übungsaufbau . . . . .	63
9.4. Aufgabe 1: Biometriemessung Konsole . . . . .	63
<b>10. IoT Monitoring</b>	<b>66</b>
10.1. Einführung . . . . .	66
10.2. Übungsaufbau . . . . .	67
10.3. Aufgabe 1: MQTT kennenlernen . . . . .	67
10.4. Aufgabe 2: MQTT mit Python verwenden . . . . .	69
10.5. Aufgabe 3: Sensordaten mit MQTT übertragen . . . . .	71
10.6. Aufgabe 4: MQTT mit Node-RED verwenden . . . . .	71
10.7. Aufgabe 5: Node-Red MQTT mit InfluxDB verwenden . . . . .	74
10.8. Aufgabe 6: InfluxDB mit Grafana verwenden . . . . .	77

10.9. Aufgabe 7: InfluxDB mit Python verwenden (optional) . . . . .	80
<b>Anhang</b>	<b>82</b>
<b>A. Raspberry Pi</b>	<b>82</b>
A.1. Raspberry Pi Image schreiben . . . . .	84
A.2. Raspberry Pi einrichten . . . . .	86
A.3. WiFi Verbindung einrichten . . . . .	86
A.4. SSH Verbindung herstellen . . . . .	87
A.5. Mobilen Hotspot nutzen . . . . .	90
A.6. Raspberry Pi Config . . . . .	90
A.7. Raspberry Pi aktualisieren . . . . .	91
A.8. Raspberry Pi im Netzwerk auffinden . . . . .	91
<b>B. Raspberry Pi Sensorbox</b>	<b>93</b>
B.1. Raspberry Pi 4 Set . . . . .	93
B.2. Sensorbox . . . . .	94
<b>C. Shell Cheat Sheet</b>	<b>96</b>
<b>D. Raspberry Pi IoT Image</b>	<b>99</b>
D.1. Raspberry Pi Zugangsdaten . . . . .	99
D.2. Installation Shell Script . . . . .	100
D.3. Konfigurationsdateien editieren . . . . .	103
D.4. Installationen testen und initialisieren . . . . .	105
D.5. Raspberry Pi Image verkleinern . . . . .	108

# Inhalt

Internet of Things ermöglicht durch die Vernetzung physischer und virtueller Objekte Infrastrukturen aufzubauen die Echtzeit Datenerfassung und Verarbeitung ermöglichen, was für die Geomatik und GIS zunehmend an Bedeutung gewinnt. Vor allem in den Bereichen Smart Cities oder Geomonitoring wird IoT eingesetzt um automatisiert Sensordaten zu erfassen, diese zu analysieren und für Entscheidungsprozesse zu nutzen. Dieser zunehmende Fokus spiegelt auch den Bedarf an Geomatiker:innen, die sich in diese Themen vertiefen und geeignete Lösungen entwickeln können.

In dieser Einführung gibt einen Überblick über die Grundlagen von IoT mit einem Fokus auf die Geomatik und Geoinformation und führt anwendungsbezogen mit praktischen Beispielen mit Raspberry Pi und unterschiedlichen Sensoren in das Thema ein. Der Anhang bietet eine Einführung in Raspberry Pi mit den wichtigsten Befehlen und Anleitungen für die Installation und Konfiguration. Zusätzlich bietet er eine praktische Anleitung für den Aufbau des vorkonfigurierten Image mit den erforderlichen Installationen wie Python, Python Bibliotheken, MQTT, Node-RED, InfluxDB und Grafana, deren Konfiguration und einfache Funktionstests.

## Aufbau

Das Skript führt in Sensorik ein und zeigt typische Anwendungen zum Thema IoT, des Weiteren zeigt es auf wie Sensordaten genutzt, gespeichert, übertragen und visualisiert werden können mittels detaillierten Beispielen und Übungen.

Die begleitenden Übungen nutzen den Einplatinenrechner Raspberry Pi mit unterschiedlichen Sensoren für die Datenerfassung, -analyse und Visualisierung. Die Programmiersprache für diesen Kurs ist Python und das genutzte Betriebssystem ist Raspberry Pi OS, eine für den Raspberry Pi angepasste Distribution von Debian (Linux).

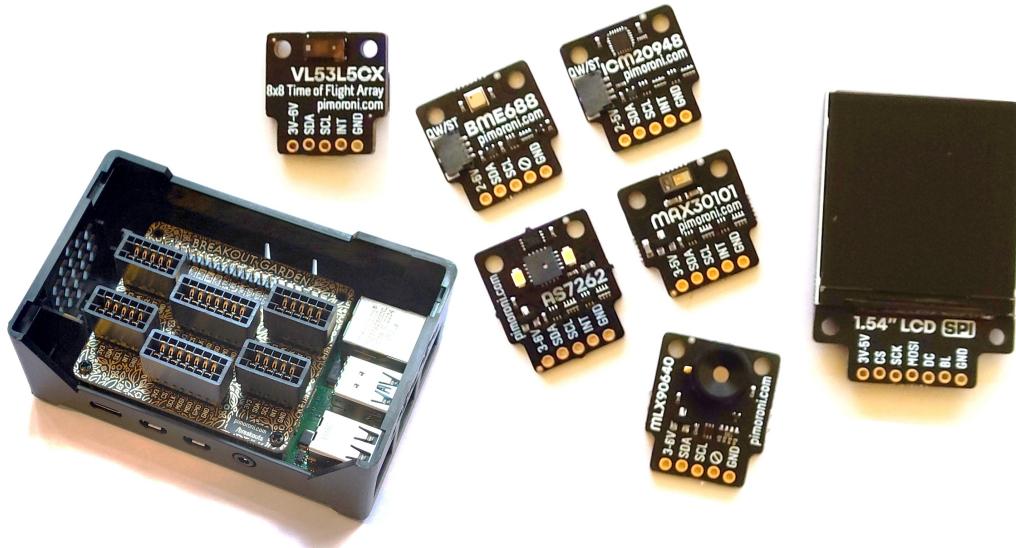
### Hinweis

#### Lernziele:

- Die Studierenden erfahren, wie IoT (Internet of Things) und Sensordaten in räumlicher Analyse eingesetzt werden können.
- Die Studierenden lernen, wie sie mit einem Einplatinenrechner Sensordaten erfassen, auswerten, kommunizieren und visualisieren können.

## Kursvorbereitung

Für den Kurs werden Raspberry Pi 4 mit dem Breakout Garden HAT und passenden Sensoren zur Verfügung gestellt. In den Computerräumen besteht Zugang zu externen Bildschirmen, Tastatur und Mäusen, mit denen die Übungen mit dem Raspberry Pi durchgeführt werden können.



**Abb. 1.:** Raspberry Pi Set: Raspberry Pi 4 mit Breakout Garden HAT und Sensoren

Für den Fernzugriff auf den Raspberry Pi und entwickeln ist ein SSH-Client (SSH Windows, Putty, Tabby), sowie eine entsprechende Entwicklungsumgebung mit Python (Anaconda, Miniconda) und Visual Studio Code empfohlen.

- SSH-Clients: [Putty](#), [Tabby](#)
- Python: [Anaconda](#), [Miniconda](#), [Python](#)

## Raspberry Pi Image Kontoinformationen

Für die Übungen wird ein Raspberry Pi Image mit vorinstallierten Paketen und Einstellungen verwendet. Die SD-Karte mit dem Image wird von der Kursleitung zur Verfügung gestellt. Das Image kann über Moodle bezogen werden.

**Tab. 1.:** Raspberry Pi Image Konto Einstellungen

Konto	User	Passwort	Kommentar
Raspberry Pi	iot	igeo@fhnw	

Konto	User	Passwort	Kommentar
influxdb	iot	igeo@fhnw	organisation: fhnw
grafana	admin	igeo@fhnw	

## Übungsunterlagen

Die Übungsunterlagen werden auf dieser Kurs Webseite Einführung in IoT zur Verfügung gestellt und werden laufend aktualisiert.

## Repository

Die Inhalte von "Einführung in IoT" sind auf der [Kurs Webseite](#) frei zugänglich und sind unter [CC BY-NC 4.0](#) lizenziert. Ideen, Änderungen und Vorschläge sind willkommen und können über das [GitHub Repository](#) eingebracht werden. Die Kurs Webseite wird mit [Quarto](#) erstellt.

# **Teil I.**

## **Theorie**

## Kapitel 1

# Internet of Things IoT

Das Internet der Dinge (IoT) bietet für die Geomatik und Geoinformationswissenschaften zahlreiche Anwendungsgebiete, durch die Automatisierung und Vernetzung von Sensoren in Echtzeit. Dies ermöglichen effiziente und präzise Überwachung von Prozessen und Umgebung (Kontext) gerade im Bereich der Smart Cities oder Geomonitoring. Beispielsweise bei Sanierungen Brücken oder bei Neubauten können Sensoren die Belastung und Verformung von Bauwerken überwachen und so frühzeitig auf Probleme hinweisen. Vorteile wie Echtzeitdaten, Automatisierung von Messprozessen, Genauigkeit und Repetition von Messungen, Kosteneffizienz durch die Reduzierung von manuellen Eingriffen und die Integration von mehreren Datenquellen unterstützen die Entscheidungsfindung in Planung und Durchführung von Projekten. Dieses Kapitel gibt eine Einführung in das Thema IoT und dessen Bedeutung für die Geomatik und Geoinformationswissenschaften.

### 1.1. Ubiquitous Computing

Aus dem Bestreben der 1980er Jahre, Technologie in den Hintergrund des Alltags einzubetten, in dem entstehenden Forschungsgebiet *Ubiquitous Computing* definierte dies Weiser *Ubiquitous Computing* (Rogers, 2006) als “die physische Welt, die reichhaltig und unsichtbar mit Sensoren, Akten, Anzeigen und Computerelementen verwoben ist, die nahtlos in die alltäglichen Objekte unseres Lebens eingebettet und durch ein kontinuierliches Netzwerk verbunden ist” mit dem Ziel im täglichen Leben unterstützend zu sein und nicht zu überladen.

**Ubiquitous Computing** Vision der Informatik allgegenwärtige Datenverarbeitung und Nutzung von Systemen ohne Bedienungsanforderung und Hardwarebelastung für die Nutzenden. Ubiquitäre Systeme agieren quasi unsichtbar im Hintergrund von Handlungsfeldern (Wiegerling, 2013).

Ethnographische Studien über den Alltag zeigen, dass der Kontext des Alltags der Menschen viel subtiler, fließender und eigenwilliger ist, als das die Theorien über den Kontext glauben machen (Salvador & Anderson, 2003). Dies erschwert eine praktische Umsetzung und Vorhersage von Bedürfnissen auf Basis von Kontextinformationen erheblich. Zusätzlich stellen sich auch ethische und soziale Fragen. Rogers (2006) plädiert für einen proaktiven Ansatz weg von *proactive Computing* zu *proactive people* und propagiert

eine fachspezifische Nutzung für bestimmte Bereiche wie beispielsweise Landwirtschaft oder Umeltsanierung und weg von der Idee von *pervasive Computing*. Zwei Technologien sind kritisch für diese Ansätze *Cloud Computing* und *Internet of Things*.

## 1.2. Internet of Things

Es existiert keine einheitliche Definition für *Internet of Things* IoT oder das Internet der Dinge. Es ist eine Bezeichnung oder Sammelbegriff für ein Netzwerk von physischen Objekten "Things", die untereinander vernetzt sind, mit Sensoren, Software und unterschiedlichen Technologien ausgestattet sind, und somit einen direkten Datenaustausch ermöglichen. Dies reicht von Monitoringsystemen, über Wildtierbeobachtungen, Smart City Anwendungen bis in die Haushalte die mit IoT Geräten ausgestattet sind.

Der Standardisierungsausschuss der International Telecommunication Union (ITU) definiert IoT als eine

.. a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) Things based on existing and evolving interoperable information and communication technologies ([ITU, 2005](#)).

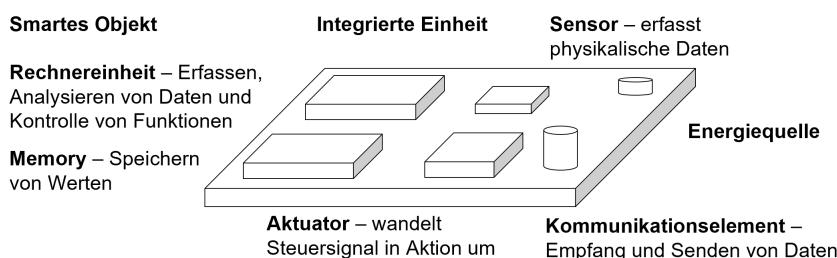
Ashton ([2009](#)) beschrieb als früher Ideengeber zu Internets der Dinge im Kontext von Supply Chain Management folgendes:

.. Ideas and information are important, but things matter much more. Yet today's information technology is so dependent on data originated by people that our computers know more about ideas than things. If we had computers that knew everything there was to know about things — using data they gathered without any help from us—we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. We need to empower computers with their own means of gathering information, so they can see, hear and smell the world for themselves, in all its random glory. RFID and sensor technology enable computers to observe, identify and understand the world—without the limitations of human-entered data. [...] The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so.

Die Kernidee von IoT ist, mit Computer Informationen, *ohne* menschliches zu tun zu erkennen. Wenn *Dinge* eigenständig Daten sammeln, diese nach Art, Messung, Messzeit und Messort geordnet werden, ermöglicht dies im Bereich der Geomatik spannende neue Ansätze, Analysen und Anwendungen. IoT ist ein multidisziplinäres Gebiet mit einem breiten Spektrum an Technologien, Protokollen, Anwendungszenarien und Disziplinen und bedingt Kenntnisse der elektronischen Komponenten, Kommunikationsprotokollen,

Echtzeitdatenanalysen, und der Lokalisierung von Objekten und Geräten (Granell et al., 2020).

Für das Konzept *Ding* oder *Thing* bestehen unterschiedliche Definitionen. Charakteristisch ist, begrenzte Ressourcen (beispielsweise geringe Rechenleistung), unzuverlässige Netzwerkverbindung, geringe Kosten für Hardware und Datenübertragung, keine Stromversorgung dafür Batterienbetrieb und nicht im Office-Umfeld sondern im Feld. Aus **Netzwerksicht** kann es als Entität mit der Möglichkeit sich mit einem Netzwerk lokal oder dem Internet zu verbinden beschrieben werden. Aus **ding-zentrierter Sicht** sind die mit den Dingen verbundenen Dienste zentral. Dienste, die Datenmengen, die von intelligenten Objekten durch deren Interaktion mit der Umgebung erfasst werden, verwalten.



**Abb. 1.1.:** Komponenten eines smarten Objekt "Thing" (Zaheeruddin & Gupta, 2020)

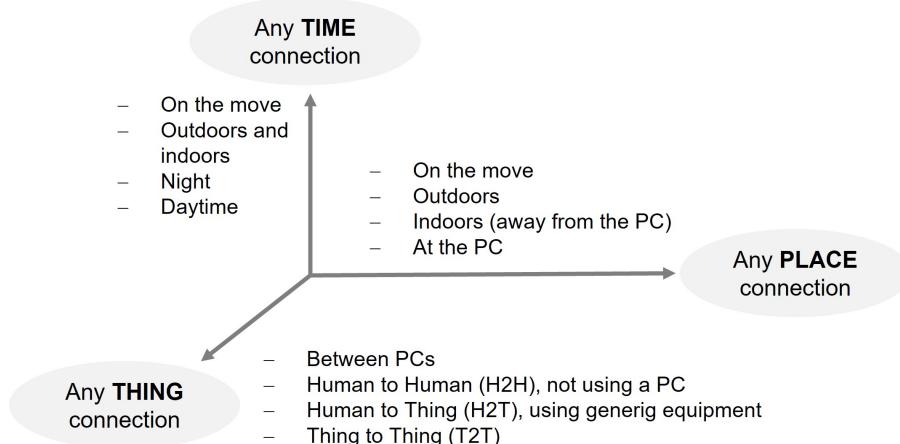
Das Internet entwickelt sich in ein Netzwerk,

- in welchem Objekte miteinander **verbunden** werden,
- Informationen aus der Umwelt gesammelt werden (**Sensorik**),
- mit der physischen Welt interagiert (**Steuerung**) werden und
- bestehende **Standards** für Dienste für den Transfer, Analyse, Anwendung und Kommunikation genutzt werden.

Die drei wesentlichen technologischen IoT Komponenten sind folglich:

- Hardware mit Sensoren, Aktoren und integrierter Kommunikationstechnologie
- Middleware bedarfsoorientierte Speicher- und Datenverarbeitungswerkzeuge für die Datenanalyse
- Präsentation mit zugänglichen und nutzerfreundlichen Visualisierungs- und Interpretationswerkzeuge über unterschiedliche Plattformen und Anwendungen

Betrachtet man die Dimensionen von IoT fügt sich aus der Sicht der Kommunikationstechnologie das *thing* mit ein (ITU, 2005). Aus Sicht der Geoinformation, sind diese drei Dimensionen sehr vertraut mit *Zeit*, *Ort* und *Objekt* dem *thing*.



**Abb. 1.2.:** Dimensionen der IoT, eine neue Dimension aus Sicht der Kommunikationstechnologie ([ITU, 2005](#))

IoT Geräte haben neben der wichtigsten Eigenschaft, dass sie mit dem Netzwerk/Internet kommunizieren können, folgende typischen Eigenschaften. Sie sind oft **funkbasiert** und oft **batteriebetrieben**. Sie lassen sich daher einfach installieren, jedoch muss im Betrieb dem Batteriebetrieb der Verbrauch berücksichtigt werden und nach Anforderung (Laufzeit vs häufige Updates) priorisiert werden<sup>1</sup>.

### 1.3. Digital Earth und IoT

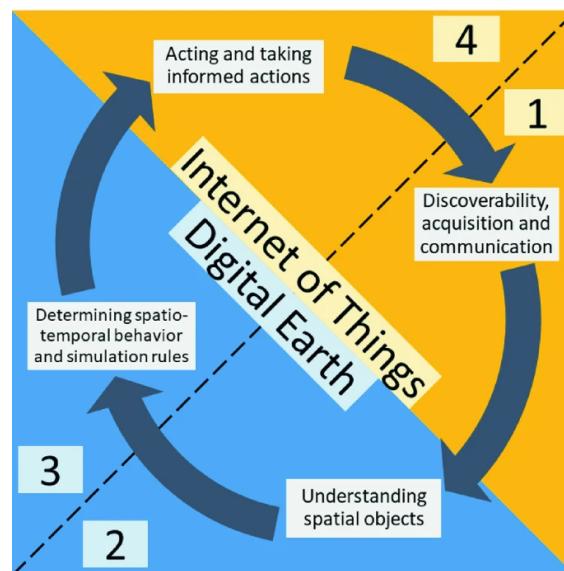
Der Vizepräsident der USA AL Gore führte 1998 in seiner Rede das Konzept der digitalen Erde ein mit der Vision die realer Erde mit einer virtuellen Nachbildung zu erweitern, einem virtuellen Zwilling ([Gore, 1998](#)).

I believe we need a “Digital Earth”. A multi-resolution, three-dimensional representation of the planet into which we can embed vast quantities of geo-referenced data.  
Gore ([1998](#))

Die Umsetzung der Vision “Digital Earth” erfordert wie auch die IoT eine entsprechende Infrastruktur, die das Auffinden, den Zugriff, die Analyse und die Verarbeitung von raumbezogenen Daten ermöglicht. Granell et al. ([2020](#)) fordern eine permanente und verstärkte Zusammenarbeit beider Bereiche. Betrachtet man die Rolle der Netzwerke und Interaktion in beiden Bereichen, kann ein Workflow von (1) der Auffindbarkeit, Erfassung und Kommunikation räumlicher Informationen, zu (2) Verständnis räumlicher Objekte und ihrer Beziehungen, dann (3) Bestimmung des raum-zeitlichen Verhaltens und der Simulationsregeln und dem (4) Handeln und Ergreifen fundierter Maßnahmen gezeichnet

<sup>1</sup>Der Raspberry Pi ist insofern mit dem verhältnismässigen hohen Stromverbrauch kein typisches IoT Device

werden. Bei (1) kann IoT mit neuen Quellen und höheren Aufnahme- und Übertragungsfrequenzen den Workflow anreichern und (2) erfordert eine kontinuierliche vertiefte Zusammenarbeit beider Infrastrukturen. Im Bereich IoT werden räumlichen Analysen wird aufgrund der aufkommenden Edge-Fog-Cloud Paradigmen zunehmen.



**Abb. 1.3.: IoT und Digital Earth Workflow (Granell et al., 2020)**

**Die Auffindbarkeit, Erfassung und Kommunikation** räumlicher Informationen erfordert das Gerät und ihre Daten auffindbar und zugänglich sind mit standardisierten Methoden in globalen zentralisierten Sammlungen, oder über dezentralen oder hierarchischen Ansätzen. Viele dieser Dienste erfordern Fachkenntnisse und einfache Zugänge zu IoT Daten fehlen. Der OGC Standard Sensor Web Enablement SWE standardisiert die Erkennung und der Zugriff auf Sensoren. Bei der räumlichen Datenerfassung in IoT haben Sensor-Metadaten eine hohe Bedeutung, wobei der SensorML-Standard einer der wichtigsten ist (Granell et al., 2020).

**Sensor Model Language SensorML-Standard** Der OGC [SensorML-Standard](#) beschreibt umfassend Sensor-Metadaten und bietet eine Schnittstelle, die das Auffinden von Sensoren und Beobachtungen folglich das Erstellen von Suchindizes erleichtert. Elemente sind Betreiber, Dienste, Standort, beobachtetes Phänomen und der zeitliche Verlauf.

**Sensor Web Enablement SWE standards suite** Der OGC [SWE Standard](#) ermöglicht die Erkennung und den Zugriff auf Sensoren und zugehörige Beobachtungsdaten über Standardprotokolle und Anwendungsprogrammerschnittstellen. Anwendung in der Erdbeobachtung, beispielsweise für das Katastrophenmanagement wie Brände, Überschwemmungen oder Vulkanausbrüche. Wobei die Unterstützung der Semantik eine Schwäche des Standards ist, *Semantic Sensor Network SSN* und Weiterentwicklungen wie *Internet of Things Ontology IoT-O* oder die *Sensor, Observati-*

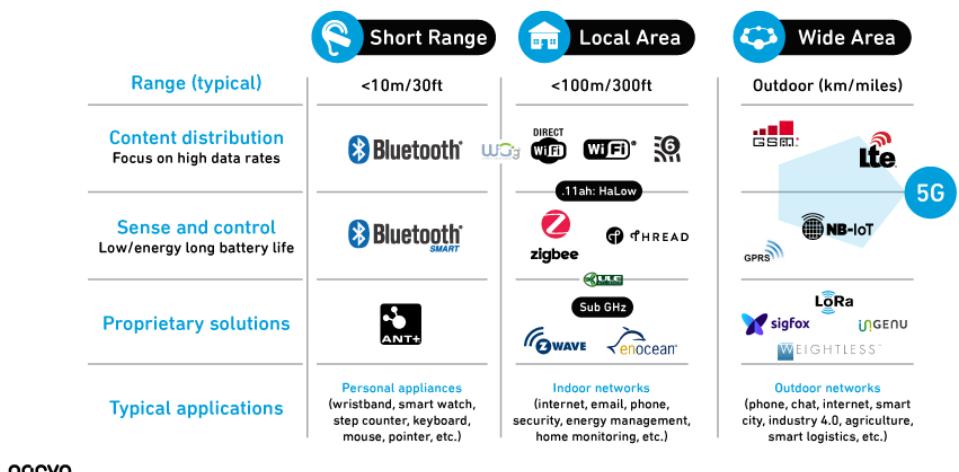
*on, Sample, and Actuator (SOSA) Ontology*, eine Zusammenarbeit zwischen W3C und OGC, nutzen Ontologien um die Semantik in IoT besser abzubilden.

SWE umfasst zwei Suchtypen das Auffinden einzelner Sensorinstanzen und Sensordienste, wobei ersteres sich auf einzelne Geräte oder Sensornetzwerke bezieht und das Zweite die Dienste, die mit dem Sensor interagieren. Die Suche kann grob in drei Gruppen unterteilt werden:

- *thematisch*: Art der Phänomene, die ein Sensor beobachtet, z. B. Temperatur, Windstärke oder Luftdruck
- *räumlich*: Ort, an dem der Sensor eingesetzt wird
- *zeitlich*: Zeitraum, in dem die Beobachtungen gemacht werden

Weiterentwicklungen im Bereich der Kommunikation zwischen *things* Geräten brachte neue Schnittstellen und über die Zeit geringere Kosten und Stromverbrauch der Kommunikationsschnittstellen wie *Bluetooth*, *Wi-Fi*, *ZigBee*, *3G-5G* oder *LORA*. Während früher einfache Datenlogger nur Daten senden konnten verfügen die Geräte heute über die Möglichkeit Daten zu Senden und zu empfangen. Dies erlaubt die Steuerung und Anpassung des Verhaltens der Geräte. Kommunikationsprotokolle die auf Maschinenkommunikation ausgerichtet sind *machine-to-machine M2M* wurden entwickelt, wie das *Advanced Message Queuing Protokoll (AMQP)*, *MQTT* oder das *streamingorientierte Protokoll STOMP*. Folgende Übersicht zeigt unterschiedliche Eigenschaften von Industriestandards für IoT über die Übertragungsdistanz, Art der Informationsübertragung und typische Anwendungsbereiche.

IoT: A Mix of Industry Standards



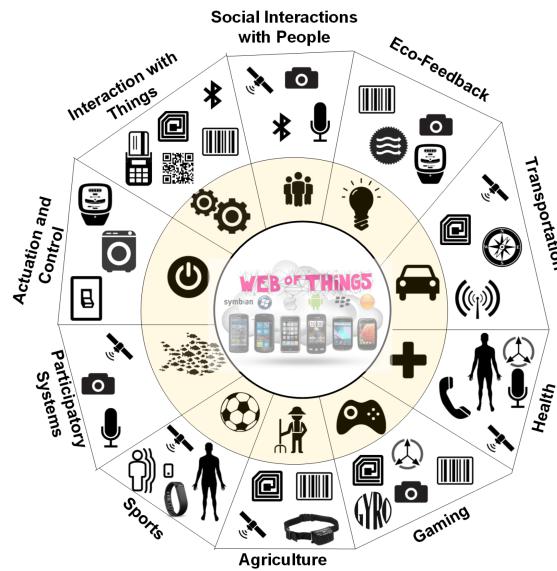
**Abb. 1.4.:** IoT Standards, Eigenschaften, Übertragungsdistanzen und Anwendungen ([Links, 2019](#))

## 1.4. IoT und GIS

*Smarte* Geräte der erzeugen grosse Datenströme mit zeitlichen und räumlichen Merkmalen, für deren Analyse die Entfernung, Fläche, Volumen oder Trajektorien entscheidend für die Datenanalyse von IoT Geräten sind. Die Vielfalt der Geräte bringt neue analytische Herausforderungen mit sich, die in der Lage sein müssen räumlich-zeitliche Echtzeitdaten mit sehr heterogenen *smarten* Geräten verarbeiten zu können (Trilles et al., 2017). Während Werkzeuge für die Analyse von Echtzeitdaten existieren, stellt die räumliche Komponente eine Herausforderung dar. Der Raum, wie der Standort, die Ausrichtung, Form und Grösse spielen eine wesentliche Rolle in IoT, da alle Geräte räumliche Eigenschaften aufweisen und miteinander räumlich zu einander in Beziehung stehen (beispielsweise der Standort, die Grösse und Orientierung von einem Auto und einem Fahrrad auf der gleichen Fahrbahn) (Granell et al., 2020). Erst über den Raum können Sensoren in Beziehung zu einander gebracht werden.

Ansätze zur räumlichen-zeitlichen Datenanalyse in Echtzeit haben noch keine standardisierten Prozeduren, die einfach und breit angewandt werden können (Granell et al., 2020). Zusätzlich sollten Standards Echtzeitanalysen ermöglichen ohne grossen zusätzlichen Rechenaufwand für Geräte, die eine beschränkte *Speicherkapazität* und *Konnektivität* haben und oft mit Batterie betrieben sind. Der OGC Sensor Observation Service SOS bedingt eine eher rechenintensive Verarbeitung von XML-Dokumenten und weitere Ansätze sind in Erarbeitung, die dem Rechnung tragen.

Kamilaris & Ostermann (2018) klassieren und geben eine Übersicht von IoT Anwendungen und Forschungsprojekte im Kontext der GIScience und IoT. Die Anwendungsbiete sind vielfältig, wie in folgender Abbildung illustriert. In fast allen aufgeführten Anwendungsbieten sind Aussagen zum Standort relevant.



**Abb. 1.5.:** Ökosystem der Anwendungen in mobile Computing im Zusammenspiel mit Sensoren von IoT Geräten ([Kamilaris & Ostermann, 2018](#))

Sie führen für die verschiedenen Anwendungsgebiete die benutzten Methodenklassen der räumlichen Analyse.

Wobei sie die Analysemethoden wie folgt gruppieren:

- **Geometric Measures:** distances and proximity of points, adjacency and connectivity
- **Data Mining:** discovering patterns from large datasets
- **Basic Analytical Operations:** methods such as buffering and overlay
- **Basic Analytical Methods:** spatial analysis of point patterns and clusters, kernels and density analysis
- **Network Analysis:** graph measures, least-cost shortest path problems and flow modeling
- **Surface Analysis & Geostatistics:** Analysis of surfaces and geostatistics deal with interpolation of surfaces and kriging.

Sie klassieren die Anwendungsgebiete und die räumlichen Analysemethoden in folgender Tabelle.

**Tab. 1.1.:** IoT Forschungsgebiet und räumliche Analysemethoden ([Kamilaris & Ostermann, 2018](#)).

IoT Area	Geometric Measures	Data Mining	Basic Analytical Operations	Basic Analytical Methods	Network Analysis	Surface Analysis & Geostatistics
Tourism	X	X	-	X	-	-

IoT Area	Geometric Measu- res	Data Mi- ning	Basic Analytical Operations	Basic Analytical Methods	Network Analy- sis	Surface Analysis & Geostatistics
Utility Network	X	X	-	-	X	-
Disaster Monitoring	X	-	X	-	-	X
Health and disease detection	X	X	-	X	-	X
Transportation	X	X	-	X	X	-
Logistics and assets	-	X	-	-	X	-
Wildlife monitoring	-	X	-	X	X	X
Agriculture	X	-	-	X	X	X
Crime prediction	-	-	-	X	-	-
Sports and gaming	X	-	-	X	-	-
Environment	-	-	X	X	X	X

In einer zweiten tabellarischen Zusammenfassung führen sie IoT basierte Methoden mit Beispielen von Generalisierung von Punktmessungen auf, welche Aussagen in grösseren Massstäben ermöglichen. Diese sind nach einzelnen IoT Methoden klassiert, wobei die ersten vier Methoden gerade für Generalisierung am beliebtesten sind.

**Tab. 1.2.:** IoT Methoden für die Generalisierung von Analysen ([Kamilaris & Ostermann, 2018](#)).

IoT-Based Method	Examples of Generalizations
Participatory sensing	Detecting emergency events at city scale [1], promoting neighborhood identity and local services [2], creating a noise map of a city [3], detecting outbreaks of dengue fever [4], developing heat maps from cyclists used for better city planning [5], producing a global spatial distribution of malaria risk [6].
Vehicular networks and transportation systems	Proactively performing urban traffic monitoring [7], travel planning based on real-time traffic information [8].

IoT-Based Method	Examples of Generalizations
Fixed IoT sensors	Urban decision-making assistance [9], wildlife monitoring and understanding of herd behavior [10], monitoring the area levels of air pollution [11], creating air temperature and precipitation maps [12], understanding fish-school characteristics around artificial reefs [13], estimating the level variations of the sand layer of sandy beaches or dunes [14].
Satellite imagery	Understanding how invasive species respond to landscape configuration relative to native species [15], assessing how the livestock agriculture affects the physical environment [16,17], modeling forest fire risk zones [18], earthquake risk assessment [19], planning of tsunami evacuation [20], creating digital maps with information about bacteria habitats [21], delineating groundwater potential zones in hard rock terrain [22].
Ground sensor sampling	Estimating the Grand Canyon height map [23], generating high-risk floodplain maps [24], creating soil fertility maps [25], assessing the spatial variation of groundwater quality and producing salinity hazard maps [26], assessing the heavy metal pollution in soils [27], estimating the zinc contamination concentrations around a lake [28].
Web-based IoT datasets	Estimating traffic from historical traffic flows [29], optimizing routes of public transportation based on taxi rides [30], exploring and analyzing attractive areas [31], associating assault rates to measures of population and place characteristics [32].
Combination of IoT methods	Assessing damage in Haiti by earthquake and facilitating emergency response [33], infrastructure asset management [34].

## Kapitel 2

# Sensoren

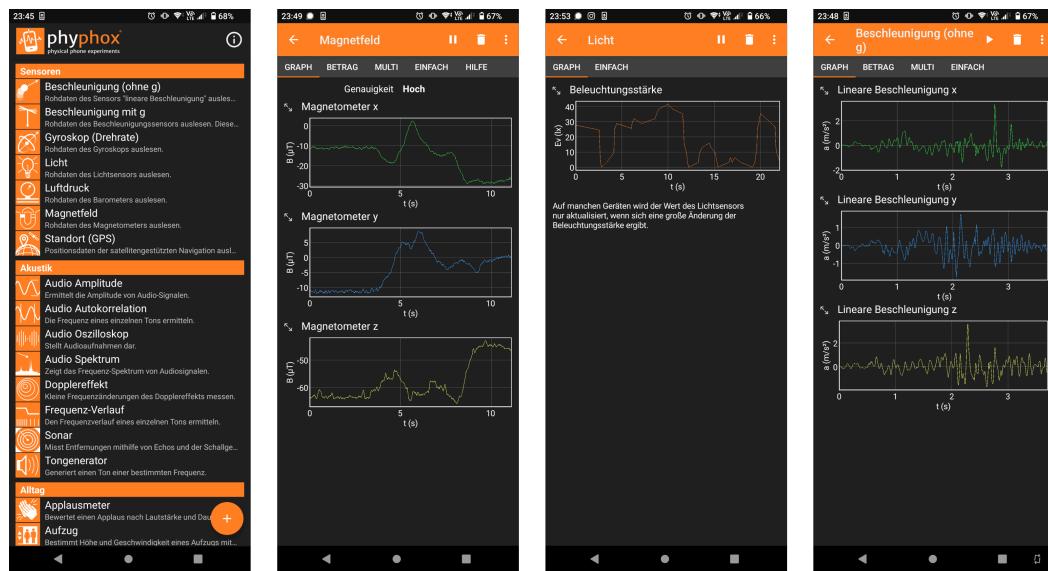
## 2.1. Sensoren in IoT

Einer der grossen Treiber von IoT ist die zunehmende Verfügbarkeit von low-cost Sensoren für viele unterschiedliche Anwendungen. Standard Sensoren sind beispielweise Umweltsensoren wie Temperatur, Luftfeuchtigkeit, Luftdruck, Bewegungssensoren mit Beschleunigung, und Magnetfeld, Neigung, Standort Sensoren wie GPS oder Kamera Licht, Schall, Gas, Flüssigkeit, wie auch Gesundheitssensoren wie Herzfrequenzvariabilität sowie GSR (galvanische Hautreaktion oder Hautleitfähigkeit). Sensoren sind in der Regel mit einem Mikrocontroller verbunden, der die Messwerte verarbeitet und an einen Computer oder ein anderes Gerät überträgt.

Der Trend geht hin zu Miniaturisierung und Multi-Sensor-Geräten mit dem Vorteil, dass über mehrere Sensoren zeitgleich(!) Messungen durchgeführt werden können und so aus der Kombination von Messwerten über Sensordatenfusion (sensor fusion) neue Erkenntnisse gewonnen werden können, beispielsweise die Kombination von GPS und Beschleunigungssensor für die Bestimmung der Position und der Geschwindigkeit (Dead Reckoning, Kalman Filter).

Sensoren können in mehrere Kategorien gruppiert werden: Umweltüberwachung (natürliche und anthropogene), Biophysikalische Sensoren und Gesundheitsüberwachung, Gebäude- und Heimautomatisierung, Automobil- und Transportanwendungen, Mobile Computing und tragbare Elektronik etc.

Viele Sensoren sind in Smartphones integriert, wie z.B. Beschleunigungssensoren, Gyroskope, Magnetometer, GPS, Lichtsensoren, Näherungssensoren, Barometer, Temperatursensoren, Feuchtigkeitssensoren, Mikrofone, Kameras, etc. Ihr Einsatz ist vielfältig, von der Unterhaltung über die Navigation bis hin zur Gesundheitsüberwachung. Mit der Applikation *phyphox - physical phone experiments* einer digitalen Experimentierbox, entwickelt von der RWTH Aachen University, können die Sensoren von Smartphones für physikalische Experimente genutzt werden.



**Abb. 2.1.:** phyphox Anwendung für physikalische Experimente mit Sensoren in Mobiltelefonen, Screenshot der Anwendung mit dem Menu und einzelnen Experimenten am Beispiel von Magnetfeld, Licht und Beschleunigung.

## Übung

### Sensorentest mit *phyphox*

Installiere auf deinem Smartphone die Applikation *phyphox* und teste die Sensoren deines Smartphones, wie die Beschleunigung, Orientierung (Magnetfeld), Standort oder Licht. Führe ein Experiment durch und dokumentiere deine Beobachtungen. Nutze die Anwendung für Vergleiche mit anderen Sensoren während der Übungen.

## 2.2. Sensoren

Sensoren sind technische Bauteile, die bestimmte physikalische oder chemische Eigenschaften messen oder eine Beschaffenheit der Umgebung qualitativ oder quantitativ erfasst. Sensoren können unterschiedlich klassiert werden, nach der Art der Erzeugung von Energie in passive und aktive Sensoren, nach dem Messprinzip oder dem Verwendungszweck.

**Tab. 2.1.:** Wirkprinzip von Sensoren Quelle: [wikipedia](#)

Wirkprinzip	Beispiel
Mechanisch	Manometer, Dehnungshebel, Federwaage, Hebelwaage, Thermometer

Wirkprinzip	Beispiel
Thermoelektrisch	Thermoelement
Resistiv	Dehnungsmessungsstreifen, Hitzdraht, Halbleiter-DMS, Pt100
Piezoelektrisch	Beschleunigungssensor
Kapazitiv	Drucksensor, Regensensor, Luftfeuchtesensor
Induktiv	Neigungssensor, Kraftsensor, Wegaufnehmer
Optisch	CCD-Sensor, Fotozelle
Akustisch	Füllstandssensor, Doppelbogenkontrolle, Ultraschall-Durchflussmesser
Magnetisch	Hall-Sensoren, Reed-Kontakt

Die Wahl von Sensoren für IoT Projekte hängt von der Anwendung ab und der dahinterstehenden Fragestellung. Die Fragestellung führt zur Bestimmung was gemessen wird und wie das was gemessen werden soll operationalisiert wird. Die Operationalisierung definiert die Messgrößen und der Wahl von einem oder mehreren Sensoren (Sensor Fusion) in einem IoT Projekt. Bei der Wahl sind mehrere Faktoren zu berücksichtigen, wie beispielsweise:

- Welche Messgröße soll gemessen werden? Welche Genauigkeit ist erforderlich? Welche Einschränkungen gibt es?
- Existiert ein Datenblatt mit den technischen Spezifikationen?
- Wie wird der Sensor angeschlossen? Wie wird der Sensor angesteuert?
- Gibt es eine Einschränkung bezüglich der Temperatur und anderen Faktoren wie Feuchtigkeit?
- Wie genau ist die Messung und welche Messabweichungen gibt es?
- Existiert es eine Library, welche die Ansteuerung des Sensors vereinfacht? Gibt es einfache Beispiele, welche die Ansteuerung des Sensors zeigen? etc.

Für die Auswahl von Sensoren sind unter anderem folgende Aspekte zu beachten:

- Anwendung (z.B. Umweltüberwachung, Gesundheitsüberwachung, Gebäude- und Heimautomatisierung, Automobil- und Transportanwendungen, Mobile Computing und tragbare Elektronik)
- Messgröße (z.B. Temperatur, Luftfeuchtigkeit, Luftdruck, Bewegung, Standort, Licht, Schall, Gas etc.) und Kombination von Messgrößen (Sensor Fusion)
- Messprinzip (z.B. mechanisch, thermoelektrisch, resistiv, piezoelektrisch, kapazitiv, induktiv, optisch, akustisch, magnetisch)
- Einschränkungen (z.B. Temperatur, Abschirmung)
- Messgenauigkeit (z.B. Auflösung, Messbereich)

- Bibliotheken (z.B. Programmiersprache, Dokumentation, Community, Beispiele, Aktualität)
- Montage (Schnittstelle, Löten, HAT)
- Datenschnittstelle (I2C, UART)
- Platzbedarf
- Stromversorgung (z.B. Batterie, Solarzelle, USB)
- Kosten
- Verfügbarkeit (z.B. Lagerbestand, Lieferzeit)
- etc.

Für Prototypenentwicklung können über verschiedene Hersteller und Distributoren Sensoren bezogen werden. Die folgende Liste gibt eine Übersicht über einige Hersteller und Distributoren von Sensoren.

- [Pi-Shop](#) ist ein Schweizer Online Händler für Raspberry Pi und Zubehör.
- [Distrilec](#) ist ein Distributor für elektronische Bauteile in der Schweiz und Europa.
- [Adafruit Industries](#) (2005, New York, USA) ist ein Hersteller von Elektronik für Maker gegründet von der MIT-Ingenieurin Limor “Ladyada” Fried.
- [Sparkfun Electronics](#) (2003, Colorado, USA) ist Open Source Elektronik Hersteller und produziert und verkauft Mikrocontroller-Plattformen, Sensoren und andere Elektronik-Komponenten.
- [Seeed Technology](#) (2008, Shenzhen, China) ist eine IoT-Innovationsplattform das sich auf Hardware-Forschung, -Produktion und -Vertrieb für Edge-Computing, Netzwerk-kommunikation und Smart-Sensing-Anwendungen spezialisiert hat und viele Produkte für Maker anbietet.
- [Pimoroni](#) 2012 ist ein Reseller von Raspberry Pi, Adafruit, Micro:bit, Arduino, Sparkfun, etc. Produkten und stellt auch eigene Elektronikprodukte und Kits her<sup>1</sup>.

## Übung

### Sensoren

Suche Dir auf den aufgelisteten Websites für elektronische Bauteile und Kits, wie dem Raspberry Pi, welche Art von Sensoren existieren.

Suche drei bis vier interessante Sensoren aus und überlege anhand der aufgeführten Aspekte zur Sensorauswahl, was alles benötigt wird um diese zu betreiben und was die Einschränkungen sind.

<sup>1</sup>Trivia: Pimoroni steht für Pirate, Monkey, Robot, Ninja.

## Kapitel 3

# Datenübertragung

Drahtlose Kommunikationstechnologien basieren heutzutage auf IEEE 802.15.4 sowie IEEE 802.15.4e Übertragungsprotokoll ([Celebi et al., 2020](#)). Das Protokoll beschreibt den die untersten beiden Schichten des OSI Modells (Abb. 3.1), die Bitübertragung und den MAC-Layer für Wireless Personal Area Networks (WPAN).

Einige der relevanten Technologien für die Datenübertragung sind, Wi-Fi, Bluetooth, ZigBee, LoRa, RFID, SigFox und Mobilfunk. Die Technologien unterscheiden sich in der Reichweite, der Datenrate, der Energieeffizienz und der Anzahl der Geräte, die angeschlossen werden können. Die Wahl der Technologie hängt von der Anwendung ab. Für die Übertragung von Sensordaten über kurze Distanzen eignet sich Bluetooth, ZigBee oder WiFi. Für die Übertragung von Sensordaten über lange Distanzen eignet sich LoRa oder Mobilfunk. Für die Übertragung von Sensordaten über kurze Distanzen mit hoher Datenrate eignet sich Wi-Fi. Für die Übertragung von Sensordaten über kurze Distanzen mit geringer Datenrate eignet sich RFID.

Je nach Anwendungsfall (Use Case) sind die Werte für die Latenzzeit (Latency), Verlässlichkeit (Reliability), Größe der Datenmenge (Data size), Reichweite (Range) für die Wahl der Kommunikationstechnologie relevant.

ISO-OSI 7 Schichtenmodell	Protokolle und Dienste (Auswahl)
Layer 7: Anwendung / Application	<b>MQTT</b> , HTTP, HTTPS, FTPS, FTP, SMTP, XMPP, LDAP, NCP...
Layer 6: Darstellung / Presentation	TLS, SSL
Layer 5: Sitzung / Session	TCP, UDP, SCTP, SPX,
Layer 4: Transport	IP, ICMP, IGMP, Ipsec, IPX
Layer 3: Vermittlung / Network	Ethernet, Token Ring, Wireless LAN
Layer 2: Sicherung / Data Link	
Layer 1: Physikalische Schicht / Physical	

**Abb. 3.1.:** Das ISO OSI Schichtenmodell mit einer Auswahl von Protokollen für die jeweilige Schicht. Das MQTT Protokoll ist in der Schichten der Darstellungs- und Anwendungsschicht angesiedelt und nutzt TCP/IP für die Datenübertragung über die IP Adressen der IoT Geräte.

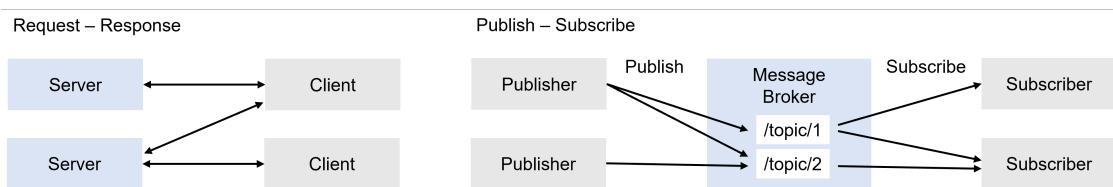
**ISO OSI Schichtenmodell** Das ISO OSI Schichtenmodell (Open Systems Interconnection Model) ist ein konzeptuelles Referenzmodell welches die Funktionsweise von Netzwerksystemen beschreibt mit dem Ziel Kommunikation über die unterschied-

lichen Technologien hinweg zu beschreiben von der Übertragung von einzelnen Bits über den Datentransport bis hin zur Anwendung beispielsweise HTTPS oder FTP ([ISO, 1994](#)). Das ISO/OSI Schichtenmodell besteht aus sieben hierarchisch aufgebauten Schichten und jeder einzelnen Schicht ist eine Aufgabe zugeordnet, wobei das Übertragungsmedium nicht definiert ist. Die Schichten eins bis vier sind transportorientierte Schichten und die Schichten fünf bis sieben sind anwendungsorientierte Schichten.

**Long Range Wide Area Network (LoRaWAN) oder LoRa** LoRaWan, Long Range Wide Area Network ermöglicht ein energieeffizientes übertragen von kleinen Datenmengen über grosse Distanzen (bis 10km).

### 3.1. Kommunikation

Zwei Kommunikationsparadigmen kommen in IoT Systemen zum Einsatz, *Publish-Subscribe* und *Request-Response*. Je nach Szenario und Anwendungsfall kommt der eine oder andere Ansatz zum Einsatz. In der *Publish-Subscribe* Kommunikation sind zwei Entitäten involviert, der Publisher, der die Daten publiziert und der Subscriber, der die Daten konsumiert, eine *one-to-many Communication*. Dies ist gerade in IoT Systemen von Vorteil, wo mehrere Geräte die Daten von einem Gerät konsumieren können ohne, dass zu jedem einzelnen Gerät eine Verbindung aufgebaut werden muss. Eine *one-to-one* Kommunikation ist hingegen eine *Request-Response* Kommunikation, in der Daten zwischen zwei Entitäten ausgetauscht werden, wobei hier der Empfänger (Adresse) der Nachricht oder der Daten bekannt sein muss. In der *Request-Response* Kommunikation ist der Server zentraler Bestandteil der Kommunikation, wohingegen in der *Publish-Subscribe* Kommunikation der Broker zentraler Bestandteil der Kommunikation ist ([Hirmer, 2023](#)). Bei *Publish-Subscribe* muss der Broker bekannt sein wohingegen die Identität der Publisher und Subscriber nicht erforderlich ist.



**Abb. 3.2.:** Kommunikationsparadigmen in der IoT Request-Response versus Publish-Subscribe

Die zentrale Komponente beim Publish-Subscribe Modells ist der Message Broker, der für das Empfangen, Zwischenspeichern und die Vermittlung der Nachrichten zu den Subscriber verantwortlich ist. Die Nachrichten oder Datenpakete werden Topics (Themen) zugeordnet die hierarchisch strukturiert sind. So können Subscriber gewisse Topics oder

Untertopics subscriben, wie beispielweise den Temperaturdaten einer Wetterstation mit *stationA/temperature*.

Message Broker können *Quality of Service* parameter definieren, die die Zuverlässigkeit der Nachrichtenübertragung definieren, ob beispielsweise die Nachricht genau oder mindestens einmal zugestellt werden soll. Dies geht jedoch zu Lasten der Performance, was gerade bei Echtzeitkommunikation relevant ist. Einer der wesentlichen Vorteile des Publish-Subscribe Modells ist, dass Publisher und Subscriber nicht gleichzeitig online sein müssen, da der Broker die Nachrichten zwischenspeichert und diese bei der nächsten Verbindung zustellt. Dies ist ermöglicht eine asynchrone Kommunikation in Echtzeit, eine wichtige Anforderung von IoT. Dies ist vor allem bei batteriebetriebenen Geräten sinnvoll, die energieeffizient arbeiten und folglich nicht kontinuierlich online sind. Eines der verbreitesten Protokolle, die *Publish-Subscribe* umsetzen ist MQTT. In **GeoMQTT** kann eine Nachricht mit einem Zeitstempel oder Interval und einer Geometrie zusätzlich dem Topic name hinzugefügt werden. Dies ermöglicht dem Subscriber zeitliche oder räumliche Filter zusätzlich zu den thematischen *Topic* Filter zu nutzen Herlé et al. (2019). Ein umfangreicherer offener Standard ist AMQP, seit 2010, der auch *request-response* Kommunikation ermöglicht<sup>1</sup>.

Eine weitere wichtige Anforderung an IoT ist die asynchrone Kommunikation in Echtzeit. Synchrone Kommunikation in Echtzeit bedingt, dass die Uhren synchron sind und beide zur gleichen Zeit kommunizieren, was in vielen Anwendungsfällen der IoT nicht wünschenswert ist.

### 3.2. MQTT - Message Queuing Telemetry Transport Protocol

Das MQTT Protokoll mit dem *Publisher-Subscriber* Ansatz ermöglicht asynchrone Kommunikation von Events in Echtzeit in dem zwischen Subscriber und Publisher ein *Broker*-Server in der Kommunikation dazwischen steht, der die Nachrichten zwischenspeichert. MQTT ist ein Protokoll für die Kommunikation zwischen Geräten und Servern und wurde ursprünglich für eine schlanke Datenübertragung über Satellitenkommunikation entwickelt. MQTT ist ein offenes Protokoll, das seit 1999 entwickelt wird, auf TCP/IP basiert und ab der Version 3.1 geöffnet wurde.

Der Publisher sendet *publish* eine Nachricht zu einem Topic (beispielsweise *gebauede1/labor1/temperature*) mit einem bestimmten *Quality of Service* (at most once, at least once, exactly once) Parameter. Der Broker speichert die Nachricht und sendet diese an alle Subscriber, die dieses Topic abonnieren *subscribe*. **Mosquitto** ist ein quelloffener Message Broker, der die MQTT implementiert. Mosquitto ist schlank und eignet sich für den Einsatz auf allen Geräten, von stromsparenden Einplatinencomputern bis hin zu kompletten Servern. Eclipse Paho ist eine quelloffene Implementierung von

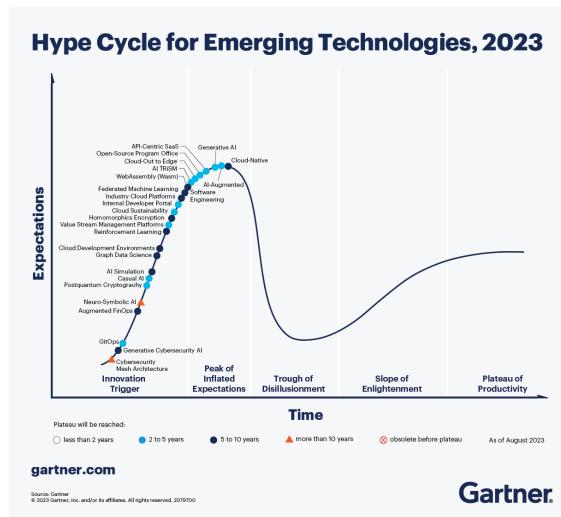
---

<sup>1</sup>Diese Protokolle sind nach dem Entwurfsmuster (Design Pattern) Beobachter (Observer) implementiert

MQTT und bietet Bibliotheken in verschiedenen Programmiersprachen wie Python, C++ oder Java an.

### 3.3. Cloud, Edge und Fog

Mit zunehmender Rechenleistung auf IoT Geräten können diese vermehrt selbst Daten prozessieren, was eine Verlagerung der Analyse ermöglicht. Die Begriffe Cloud, Edge und Fog bezeichnen im wesentlichen wo in Infrastrukturen die Datenprozessierung durchgeführt wird. In **Cloud** Infrastrukturen erfolgt die Prozessierung zentralisiert in der Cloud und grossen Recheninfrastrukturen, wohingegen **Edge**-Computing eine Infrastruktur bezeichnet, in der die lokalen Geräte selbst einen Teil der Daten dezentral und möglichst lokal verarbeiten. Fog - ein von Cisco eingeführter Begriff - bezeichnet Cloud Computing im lokalen Netzwerk, beispielsweise könnte ein Transportunternehmen die Verwaltung der Datenprozessierung in der eigenen Cloud durchführen.



**Abb. 3.3.:** Im Gartner Hype Cycle for Emerging Technologies 2023 erreicht der “Hype” *Cloud-Out to Edge* den Peak of Inflated Expectations (Gartner, 2023). Das Hype Cycle Thema “Cloud-Out to Edge” wurde im aktualisierten Hype Cycle (2024) als Hype Thema entfernt.

# Literatur

- AMS OSRAM Group. (2022). *Datasheet AS7262 6-Channel Visible Spectral\_ID Device with Electronic Shutter and Smart Interface*. AMS OSRAM Group.
- Ariza, J. Á., & Baez, H. (2022). Understanding the Role of Single-Board Computers in Engineering and Computer Science Education: A Systematic Literature Review. *Computer Applications in Engineering Education*, 30(1), 304–329. <https://doi.org/10.1002/cae.22439>
- Ashton, K. (2009). That „Internet of Things“ Thing. *RFID journal*, 22(7), 97–114.
- Celebi, H. B., Pitarokilis, A., & Skoglund, M. (2020). Wireless Communication for the Industrial IoT. In I. Butun (Hrsg.), *Industrial IoT : Challenges, Design Principles, Applications, and Security* (S. 57–94). Springer International Publishing. [https://doi.org/10.1007/978-3-030-42500-5\\_2](https://doi.org/10.1007/978-3-030-42500-5_2)
- Eclipse Foundation. (2021). Authentication Methods. In *Eclipse Mosquitto*. <https://mosquitto.org/documentation/auth-methods/>.
- Gartner. (2023). 4 Exciting New Trends in the Gartner Emerging Technologies Hype Cycle. In *Gartner*. <https://www.gartner.com/en/articles/what-s-new-in-the-2023-gartner-hype-cycle-for-emerging-technologies>.
- Gartner. (2024). Gartner 2024 Hype Cycle for Emerging Technologies Highlights Developer Productivity, Total Experience, AI and Security. In *Gartner*. <https://www.gartner.com/en/newsroom/press-releases/2024-08-21-gartner-2024-hype-cycle-for-emerging-technologies-highlights-developer-productivity-total-experience-ai-and-security>.
- Gas Sensor BME688. (2022). In *Bosch Sensortec*. <https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors/bme688/>.
- Gore, A. (1998). The Digital Earth. *Australian Surveyor*, 43(2), 89–91. <https://doi.org/10.1080/00050348.1998.10558728>
- Granell, C., Kamilaris, A., Kotsev, A., Ostermann, F. O., & Trilles, S. (2020). Internet of Things. In H. Guo, M. F. Goodchild, & A. Annoni (Hrsg.), *Manual of Digital Earth* (S. 387–423). Springer. [https://doi.org/10.1007/978-981-32-9915-3\\_11](https://doi.org/10.1007/978-981-32-9915-3_11)
- Grimm, D. (2023). *2030 Geodätische Messtechnik GMT I*.
- Herlé, S., Bill, R., & Blankenbach, J. M. (2019). *A GeoEvent-driven architecture based on GeoMQTT for the Geospatial IoT* (phdthesis RWTH-2019-10695). Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2019.
- Hirmer, P. (2023). Foundations. In P. Hirmer (Hrsg.), *Model-Based Approaches to the Internet of Things* (S. 7–15). Springer International Publishing. [https://doi.org/10.1007/978-3-031-18884-8\\_2](https://doi.org/10.1007/978-3-031-18884-8_2)
- Hollingworth, G. (2024). Raspberry Pi Connect. In *Raspberry Pi*.
- influxdata. (2023). *Get Started with InfluxDB | InfluxDB OSS v2 Documentation*. <https://docs.influxdata.com/influxdb/v2/get-started/>.

- InvenSense, T. (2021). *ICM-20948 World's Lowest Power 9-Axis MEMS MotionTracking™ Device*.
- ISO. (1994). ISO/IEC 7498-1:1994. In ISO. <https://www.iso.org/standard/20269.html>.
- ITU. (2005). *ITU Internet Reports 2005: The Internet of Things*. International Telecommunication Union.
- Jensen, K. (2020). *White Paper Chip-scale Spectral Sensing: Understanding the New Uses for Ultra-Precise Light-Source Measurement* (S. 10) [White {{Paper}}]. AMS.
- Kamilaris, A., & Ostermann, F. O. (2018). Geospatial Analysis and the Internet of Things. *ISPRS International Journal of Geo-Information*, 7(7), 269. <https://doi.org/10.3390/ijgi7070269>
- Laska, M., Herle, S., Klammer, R., & Blankenbach, J. (2018). A Scalable Architecture for Real-Time Stream Processing of Spatiotemporal IoT Stream Data—Performance Analysis on the Example of Map Matching. *ISPRS International Journal of Geo-Information*, 7(7), 238. <https://doi.org/10.3390/ijgi7070238>
- Leica Geosystems. (2022). *Leica TS60/MS60/TM60 Gebrauchsanweisung*.
- Links, C. (2019). *IoT Standards: The End Game* - Qorvo. <https://www.qorvo.com/design-hub/blog/iot-standards-the-end-game>.
- maxim integrated. (2020). *MAX30101 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health*.
- Melexis. (2019). *MLX90640 32x24 IR Array Datasheet*.
- Raspberry Pi Ltd. (2024a). Raspberry Pi Connect Beta - Access Your Raspberry Pi from Anywhere. In *Raspberry Pi*. <https://www.raspberrypi.com/software/connect/>.
- Raspberry Pi Ltd. (2024b). Raspberry Pi Documentation Config.Txt. In *Raspberry Pi*. [https://www.raspberrypi.com/documentation/computers/config\\_txt.html](https://www.raspberrypi.com/documentation/computers/config_txt.html).
- Rogers, Y. (2006). Moving on from Weiser's Vision of Calm Computing: Engaging UbiComp Experiences. In P. Dourish & A. Friday (Hrsg.), *UbiComp 2006: Ubiquitous Computing* (S. 404–421). Springer. [https://doi.org/10.1007/11853565\\_24](https://doi.org/10.1007/11853565_24)
- Salvador, T., & Anderson, K. (2003). Practical Considerations of Context for Context Based Systems: An Example from an Ethnographic Case Study of a Man Diagnosed with Early Onset Alzheimer's Disease. In A. K. Dey, A. Schmidt, & J. F. McCarthy (Hrsg.), *UbiComp 2003: Ubiquitous Computing* (S. 243–255). Springer. [https://doi.org/10.1007/978-3-540-39653-6\\_19](https://doi.org/10.1007/978-3-540-39653-6_19)
- Sliney, D. H. (2016). What Is Light? The Visible Spectrum and Beyond. *Eye*, 30(2), 222–229. <https://doi.org/10.1038/eye.2015.252>
- STMicroelectronics. (2021). *VL53L5CX - Datasheet - Time-of-Flight 8x8 Multizone Ranging Sensor with Wide Field of View*.
- STMicroelectronics. (2023). *Description of the Fields of View of STMicroelectronics' Time-of-Flight Sensors*.
- Trilles, S., Belmonte, Ò., Schade, S., & Huerta, J. (2017). A Domain-Independent Methodology to Analyze IoT Data Streams in Real-Time. A Proof of Concept Implementation for Anomaly Detection from Environmental Data. *International Journal of Digital Earth*,

- 10(1), 103–120. <https://doi.org/10.1080/17538947.2016.1209583>
- Upton, E. (2022). Supply Chain Update - It's Good News! In *Raspberry Pi*.
- Wiegerling, K. (2013). Ubiquitous Computing. In A. Grunwald & M. Simonidis-Puschmann (Hrsg.), *Handbuch Technikethik* (S. 374–378). J.B. Metzler. [https://doi.org/10.1007/978-3-476-05333-6\\_71](https://doi.org/10.1007/978-3-476-05333-6_71)
- Zaheeruddin, & Gupta, H. (2020). Foundation of IoT: An Overview. In M. Alam, K. A. Shakil, & S. Khan (Hrsg.), *Internet of Things (IoT): Concepts and Applications* (S. 3–24). Springer International Publishing. [https://doi.org/10.1007/978-3-030-37468-6\\_1](https://doi.org/10.1007/978-3-030-37468-6_1)
- Zamorano, J., García, C., Tapia, C., Miguel, A. S. de, Pascual, S., & Gallego, J. (2016). STARS4ALL Night Sky Brightness Photometer. *International Journal of Sustainable Lighting*, 18, 49–54. <https://doi.org/10.26607/ijsl.v18i0.21>

## **Teil II.**

### **Praktika**

## Kapitel 4

# Luftqualitätmessung

Diese Übung zeigt wie Sensoren mit dem Raspberry Pi genutzt werden können am Beispiel des BME688 Sensors. Dieser Sensor erfasst Temperatur, Luftdruck, Luftfeuchtigkeit und Gas Scanner VOC. Die Übung zeigt wie der Sensor angeschlossen wird, wie die Sensordaten ausgelesen werden und wie die Sensordaten in einer Datei gespeichert werden können.

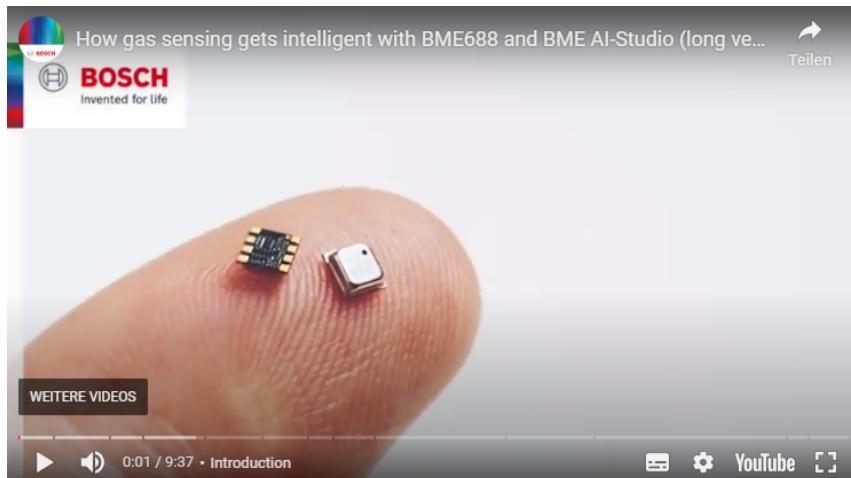
### 4.1. Einführung

Ziel dieser ersten Übung ist es den BME688 Sensor kennen zu lernen und die Sensordaten auszulesen. Der BME688 ist ein 4-in-1 Sensor für Temperatur, Luftdruck, Luftfeuchtigkeit und Gas Scanner VOC. Der Sensor verfügt über eine I2C Schnittstelle, die mit der Python Library [bme680-python](#) angesteuert und die Sensordaten ausgelesen werden können.

**Unterlagen:** [E01\\_Luftqualitaet.zip](#)

#### Vorbereitung

- Lest das Kapitel im Anhang zu [Raspberry Pi](#)
  - Konzentriere Dich auf die wichtigsten Details zur Inbetriebnahme des Raspberry Pi
- Lest die Dokumentation zum BME688 Sensor „Gas Sensor BME688“ ([2022](#))
  - Konzentriere Dich auf die Beschreibung der Schnittstelle und technische Spezifikation auf dem Datenblatt



**Abb. 4.1.:** Youtube Video: How gas sensing gets intelligent with BME688 and BME AI-Studio [Youtube Video](#)

---

### Unterlagen

---

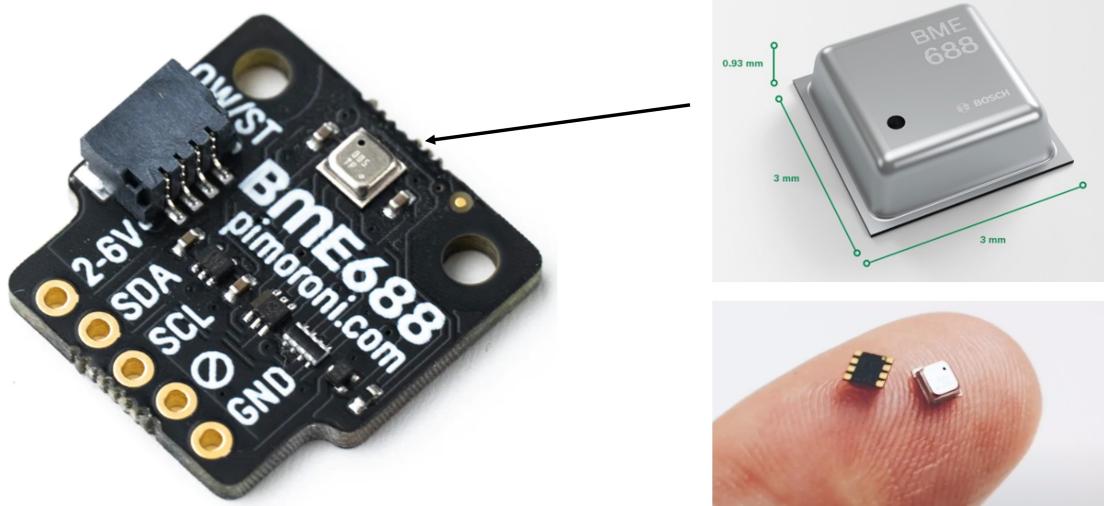
Produkt	<a href="#">BME688 4-in-1 Air Quality Breakout</a>
Datenblatt	<a href="#">Bosch Datasheet BME 688</a>
GitHub	<a href="#">bme680-python Library mit Beispielen</a>
Tutorial	<a href="#">Getting started with BME680 Breakout</a>

---

## 4.2. BME688

BME688 - Bosch Sensor für Temperatur, Luftdruck, Luftfeuchtigkeit, Gas Scanner VOC (Abb. 4.2)

- Temperatur +/-0.5°C (-40° .. -85°)
- Luftdruck +/-0.12hPa (300...1100hPa)
- Luftfeuchtigkeit +/-3% (0 ...100%)
- Gas Scanner VOC, VSCs (AI)
- Python, C Library
- Raspberry Pi Pins 1,3,5,7,9



**Abb. 4.2.:** BME688 Bosch Sensor für Luftqualitätsmessung mit Referenzbild für den Grössenvergleich

### 4.3. Übungsaufbau

- Schliesse den Raspberry Pi an Monitor, Keyboard und Maus an oder verbinde Dich mit diesem über SSH (und SFTP).
- Erstelle auf dem Raspberry Pi im `Documents` Ordner einen neuen Ordner `BME688`, in welchem Du Änderungen und neue Dateien für diese Übung speichern kannst.
- Schliesse den Sensor **BME688** an den Raspberry Pi über die Breakout Garden **I2C** Schnittstelle an.

### Kontrolle der Hardware

Kontrolliere mit dem Befehl `i2cdetect -y 1` ob der Raspberry Pi mit dem Sensor verbunden ist und der Raspberry Pi Zugriff auf den Sensor hat. Erscheint eine Zahl, dann hat der Raspberry Pi den Sensor auf dem I2C Bus erkannt. Falls Du mehr über das Program und den Befehl wissen möchtest, kannst Du mit dem Befehl `man i2cdetect` das Manual `man` aufrufen.

```
i2cdetect -y 1
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          - - - - - - - - - - - -
10:          - - - - - - - - - - - -
20:          - - - - - - - - - - - -
30:          - - - - - - - - - - - -
40:          - - - - - - - - - - - -
```

## Hinweis

### Sensor Ausrichtung beachten

Beim Anschliessen der Sensoren in die Schnittstellen des Breakout Garden **unbedingt** die korrekte Ausrichtung beachten! Die Beschriftung der Anschlüsse auf dem Sensor und dem Breakout Garden müssen übereinstimmen!



**Abb. 4.3.:** Sensor links korrekt angeschlossen, rechts falsch ausgerichtet angeschlossen.

```
50: ---  
60: ---  
70: --- 76 ---
```

i2cdetect installieren, falls das Programm nicht existiert:

```
sudo apt-get update  
sudo apt install python3-smbus  
sudo apt install i2c-tools
```

**Hinweis:** Damit der Befehl `i2cdetect` funktioniert, muss der I2C Bus aktiviert sein. Dies kann mit dem Befehl `sudo raspi-config` im Menü Interfacing Options und I2C aktiviert werden.

## Kontrolle der Installation

Aktiviere die virtuelle Python Environment (hier unter `~/.env`) und teste ob die für die Übung erforderlichen Python Libraries installiert sind. Aktiviere über folgenden Befehl die virtuelle Umgebung, welche für diese Übungen im Homeordner `~/.env` erstellt wurde<sup>1</sup>.

<sup>1</sup>In dieser Übung wird die Python Virtual Environment `.env` im Homeordner `~/` erstellt und aktiviert. Diese kann auch an einem Ort erstellt werden, beispielweise projektbasiert. Erstellen einer Virtual Environment:  
`python -m venv ~/Development/Projekt1/env`

Die aktivierte Umgebung wird durch die Anzeige des Umgebungsname in der Kommandozeile angezeigt, hier mit (`.env`).

```
source ~/ .env/bin/activate
```

Teste ob die für die Übung erforderlichen Python Libraries installiert sind. Ein einfacher Test zur Kontrolle, ob ein Modul installiert ist, ist über das Import Statement des Moduls über die Python Konsole im Terminal. Falls kein Fehler auftritt, ist das Modul in der virtuellen Environment installiert.

```
python -c "import math"  
0  
python -c "import numpy"  
Traceback (most recent call last):  
  File "<string>", line 1, in <module>  
    ImportError: No module named numpy
```

(1)

(2)

(1) **math** Modul existiert

(2) **numpy** Modul existiert nicht

Installiere das Modul mit folgendem Befehl, falls es nicht installiert ist.

```
pip install bme680
```

## Kopiere (Clone) die Library mit den Beispielen auf den Raspberry Pi

Wechsle in den Ordner *Documents* und kopiere mit folgenden Befehlen die Library auf Deinen Raspberry Pi.

```
cd Documents  
git clone https://github.com/pimoroni/bme680-python  
cd bme680-python/examples
```

## 4.4. Aufgabe 1: Sensormessungen ausführen

Teste das Beispiel `read-all.py` im Ordner `examples`. Dieses Beispiel gibt die Temperatur, Luftdruck und Luftfeuchtigkeit des Sensors BME 688 aus.

```
python read-all.py
```

Mit `Ctrl+c` kann das Script wieder gestoppt werden. Die Ausgabe sollte in etwa so aussehen (gekürzt):

```
# Output Beispiel
read-all.py - Displays temperature, pressure, humidity, and gas.
Press Ctrl+C to exit!

Calibration data:
par_gh1: -10
...
Initial reading:
gas_index: 0
gas_resistance: 1338124.79581836
heat_stable: False
humidity: 44.397
meas_index: 0
pressure: 990.82
status: 32
temperature: 28.89

Polling:
28.89 C,990.82 hPa,44.39 %RH
28.91 C,990.82 hPa,44.37 %RH,5684.846331497602 Ohms
28.94 C,990.80 hPa,44.31 %RH,5684.846331497602 Ohms
28.97 C,990.81 hPa,44.24 %RH,5684.846331497602 Ohms
29.00 C,990.81 hPa,44.19 %RH,5684.846331497602 Ohms
29.03 C,990.82 hPa,44.12 %RH,5684.846331497602 Ohms
```

Folgendes Code Snippet zeigt eine gekürzte Version des `read-all.py` Python Beispiels, der die *Temperatur*, *Luftdruck* und *Luftfeuchtigkeit* mit der BME680 Library ausgibt.

```
#!/usr/bin/env python
import bme680
try:
    sensor = bme680.BME680(bme680.I2C_ADDR_PRIMARY) (1)
except (RuntimeError, IOError):
    sensor = bme680.BME680(bme680.I2C_ADDR_SECONDARY)

# Oversampling Einstellungen
sensor.set_humidity_oversample(bme680.OS_2X) (2)
sensor.set_pressure_oversample(bme680.OS_4X)
sensor.set_temperature_oversample(bme680.OS_8X)
sensor.set_filter(bme680.FILTER_SIZE_3)
```

```
print('Sensordaten:')

try:
    while True:
        if sensor.get_sensor_data():
            output = '{0:.2f} C, {1:.2f} hPa, {2:.3f} %RH'.format(
                sensor.data.temperature,
                sensor.data.pressure,
                sensor.data.humidity)
            print(output)
except KeyboardInterrupt:
    pass
```

(3)

- ① Testen der beiden möglichen I2C Adressen
- ② Oversampling Einstellungen können Messungen durch Mitteln verbessern und das Rauschen und Drifts reduzieren
- ③ Sensordaten auslesen

### Übung 4.1. BME 688

Studiert die Python Skripte und online Tutorial zum Sensor

- Wie wird auf den Sensor zugegriffen?
- Wie reagiert der Feuchtigkeitssensor auf Änderungen?
- Wie hoch sind die Werte für den Luftdruck, was sind Vergleichswerte?
- Wie könnt Ihr einfach Sensorwerte in eine Datei schreiben?

## 4.5. Aufgabe 2: Berechnung der Atmosphärenkorrektur für Distanzmessungen (optional)

Geodätische Distanzmessverfahren wie bei der Tachymetrie (Totalstationen) und Laser-scanning benötigen eine Korrektur der Messwerte, um die Distanz zwischen zwei Punkten auf der Erde zu berechnen. Über eine *Massstabskorrektur* (ppm [mm/km]) können distanzproportionale Reduzierungen berücksichtigt werden, die *Projektionsverzerrung*, *Reduktion auf Meereshöhe* und die *Atmosphärenkorrektur* berücksichtigen. Die Luftfeuchtigkeit beeinflusst Distanzmessungen in feuchtem und heissen Klima. Diese muss den zur Messzeit geltenden atmosphärischen Bedingungen angepasst werden.

Die Korrektur wird als *Atmosphärenkorrektur* bezeichnet und ist abhängig von der *Temperatur*, dem *Luftdruck* und der *Luftfeuchtigkeit*. Die Korrektur wird in ppm (parts per million [mm/km]) angegeben und kann mit folgender Gleichung 4.1 berechnet werden (Grimm, 2023; Leica Geosystems, 2022, S. 87) und gilt für die Distanzmessung mit sichtbaren roten Laser

$$\Delta D_1 = 286.338 - \left[ \frac{0.29535 \cdot p}{(1+\alpha \cdot t)} - \frac{4.126 \cdot 10^{-4} \cdot h}{(1+\alpha \cdot t)} \cdot 10^x \right] \quad (4.1)$$

$\Delta D_1$	Atmosphärische Korrektur	[ppm]
$p$	Luftdruck	[mbar]
$t$	Lufttemperatur	[°C]
$h$	relative Luftfeuchte	[%]
$\alpha$	= $\frac{1}{273.15}$	
$x$	= $(7.5 \cdot \frac{t}{237.3 + t}) + 0.7857$	

Folgende Funktion in Python berechnet die Atmosphärenkorrektur für Geodätische Distanzmessverfahren basierend auf der Gleichung 4.1

```
import math

def calculate_atmospheric_correction(temperature, air_pressure, humidity):
    # Constants
    ALPHA = 1 / 273.15
    X = (7.5 * temperature / (237.3 + temperature)) + 0.7857

    # Interim results
    denominator = 1 + ALPHA * temperature

    formula0 = 286.338
    formula1 = 0.29535 * air_pressure / denominator
    formula2 = 4.126 * 10 ** (-4) * humidity / denominator
    formula3 = 10 ** X

    # ppm-calculation
    correction_ppm = round(formula0 - (formula1 - formula2 * formula3), 2)

    # Return important values as a dictionary
    result = {
        "temperature": temperature,
        "air_pressure": air_pressure,
        "humidity": humidity,
        "correction_ppm": correction_ppm
    }
    return result
```

Für Distanzmessungen höchster Genauigkeit sollte die atmosphärische Korrektur auf 1ppm genau bestimmt werden, folglich sollten die Messwerte mindestens folgenden Genauigkeit aufweisen: die Temperatur auf 1°C, der Luftdruck auf 3mbar (1Millibar =

1hPa) und die Luftfeuchtigkeit auf 20 genau bestimmt werden ([Leica Geosystems, 2022, S. 87](#)).

### Übung 4.2. Atmosphärenkorrektur

Nutzt die Funktion `calculate_atmospheric_correction` um eine atmosphärische Korrektur mit den momentan gemessenen amtmosphärischen Bedingungen zu berechnen.

- Wie hoch ist die Korrektur mit den Werten des BME688 Sensors?
- Wie hoch ist dieselbe Korrektur bei doppelter Luftfeuchtigkeit?
- Wie hoch ist die Korrektur bei 20°C, 1000hPa und 50% sowie 100% Luftfeuchtigkeit?
- Erfüllt der BME688 Sensor die Anforderungen für Distanzmessungen höchster Genauigkeit?

## Kapitel 5

# Spektralmessung

Diese Übung führt einen Spektralsensor ein und zeigt wie Spektralmessungen durchgeführt werden können. Der AS7262 Spektralsensor misst über 6-Kanäle im sichtbaren Bereich und kann für spektroskopische Messungen eingesetzt werden.

## 5.1. Einführung

Ziel dieser Übung ist es den AS7262 Spektralsensor kennen zu lernen und die Sensordaten auszulesen. Der AS7262 ist ein Spektralsensor, der über eine I2C Schnittstelle mit dem Raspberry Pi verbunden wird und einer Python Library angesteuert werden kann.

**Unterlagen:** [E02\\_Spektralmessung.zip](#)

### Vorbereitung

- Schaut das Video von AMS zum [AS7262 Spektralsensor](#) (bis Minute 2:20)
- Lest das White Paper von Jensen ([2020](#)) über Spektralsensoren
  - Konzentriere Dich da auf die Beschreibung der *Multi-channel spectral sensors* und in welchen Gebieten diese Sensoren eingesetzt werden.
- Studiert das Datenblatt zum AS7262 Spektralsensor AMS OSRAM Group ([2022](#))
  - In welchen Temperaturbereichen kann der Sensor eingesetzt werden?



**Abb. 5.1.:** ams Spectral ID iSPI Eval Kit [Youtube Video](#)

---

#### Unterlagen

---

Produkt	<a href="#">AS7262 Breakout</a>
Datenblatt	<a href="#">AS7262</a>
GitHub	<a href="#">as7262-python</a>
Video	<a href="#">ams Spectral ID iSPI Eval Kit</a>

---

## 5.2. Spektrometer

Spektrometer, sind Sensoren, die das Licht auf einzelne Bänder der Wellenlänge (in Nanometer nm) messen. Für das menschliche Auge wird das sichtbare Spektrum zwischen 360nm bis 830nm angegeben, wobei sich der Bereich unter optimalen Bedingungen in UV und NIR erstrecken kann ([Sliney, 2016](#)).

Auf kleinen Chip verbaute Spektralsensoren ermöglichen zahlreiche neue Anwendungen, die früher mit grossen Spektroskopen durchgeführt wurden. Die Spektralsensorsysteme finden Anwendung wie Farbbestimmung, Authentifizierung und Spektralanalyse von Substanzen, Materialien, Lebensmitteln und Flüssigkeiten und werden in den Bereichen von Konsumgütern, Industrie und Medizin eingesetzt.

Ein Multispektralsensor liefert die Antwort auf die Frage, ob eine orangefarbene Probe eine Mischung aus Rot und Gelb oder ein reines Orange ist. Multispektralsensoren teilen das gewählte Spektrum in Spektralkanäle auf (siehe Abb. 5.2) und sind so angeordnet, dass die Spektralkanäle kontinuierlich gut abgedeckt sind. Die Messung erfolgt im sicht-

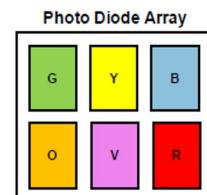
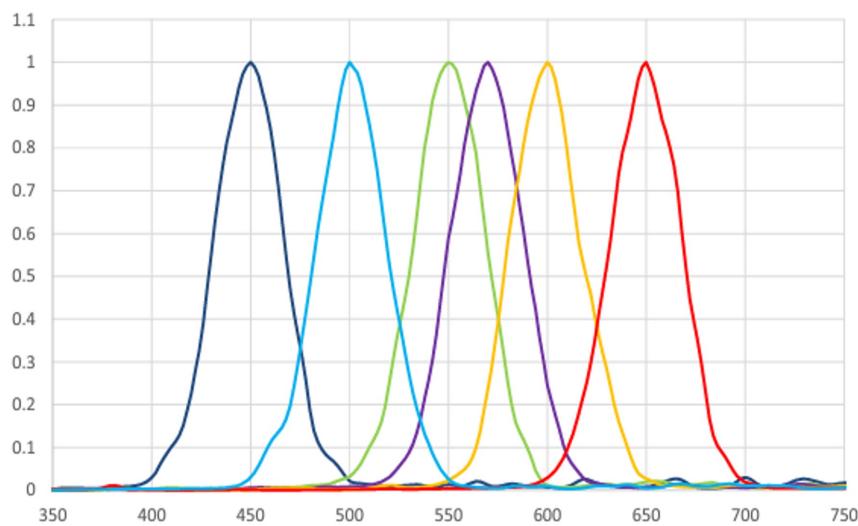
baren Bereich radiometrisch ([Jensen, 2020](#)). Das heißt, der Sensor misst die spektrale Leistungsverteilung der Messung.

### 5.3. AS7262 Multispektralsensor

Der AS7262 ist ein sehr kompakter Multispektralsensor, der über 6 Spektralkanäle im sichtbaren Bereich misst. Der Sensor eignet sich für spektroskopische Messungen wie *Farbmessung, Absorption, Bestrahlungsstärke, Reflexion* und *Transmission*. Der Sensor ist auf einem Breakout Board verbaut und kann über die I2C Schnittstelle mit dem Raspberry Pi verbunden werden, zusätzlich sind zwei LEDs verbaut, die den Messbereich für Messungen beleuchten können.

AS7262 - 6-Kanal Spektral Sensor (Abb. 5.2)

- 6 Spektralkanäle (450, 500, 550, 570, 600, 650nm)
- 2 on-board Beleuchtungs-LEDs
- I2C interface (address: 0x49)
- Python Library
- werkseitig kalibriert



**Abb. 5.2.:** links: Spektrale Sensitivität des AS7262, oben: AS7262 Breakout von Pimoroni, unten: AS7262 Photodioden Array

### 5.4. Übungsaufbau

- Schliesse den Raspberry Pi an Monitor, Keyboard und Maus an oder verbinde Dich mit diesem über SSH (und SFTP).

- Erstelle auf dem Raspberry Pi im `Documents` Ordner einen neuen Ordner **AS7262**, in welchem Du Änderungen und neue Dateien für diese Übung speichern kannst.
- Schliesse den Sensor **AS7262** an den Raspberry Pi über die Breakout Garden **I2C** Schnittstelle korrekt an (siehe [E01 Luftqualität](#)), so dass die Beschriftung der Anschlüsse am Sensor und bei der Schnittstelle übereinstimmen.
- Kontrolliere mit dem Befehl `i2cdetect -y 1` ob der Raspberry Pi mit dem Sensor verbunden ist. Der Sensor sollte auf der Adresse `0x49` erkannt werden.
- Aktiviere die virtuelle Environment von Python mit `source ~/.env/bin/activate` und kontrolliere, ob die Library `as7262` installiert ist mit `python -c "import as7262"`. Bei einer Fehlermeldung muss die Library in der aktivierte virtuellen Environment mit `pip install as7262` installiert werden.

Wechsle in den Ordner `Documents` und kopiere mit folgenden Befehlen die Library auf Deinen Raspberry Pi.

```
cd Documents
git clone https://github.com/pimoroni/as7262-python
cd as7262-python/examples
```

## 5.5. Aufgabe 1: Spektralmessungen durchführen

Teste das Beispiel `spectrum.py` im Ordner `examples`. Dieses Beispiel gibt die Messungen der einzelnen Spektralkanäle aus.

Startet das Script mit `python spectrum.py`<sup>1</sup>. Mit `Ctrl+c` kann das Script wieder gestoppt werden. Die Ausgabe sollte in etwa so aussehen (gekürzt):

```
python spectrum.py
Orange: 30.22943878173828
Red:    14.262250900268555
Orange: 22.67207908630371
Yellow: 14.271308898925781
Green:   13.56598949432373
Blue:    6.527451515197754
Violet:  7.2663960456848145
```

Folgendes Code Snippet zeigt eine gekürzte Version des `spectrum.py` Python Beispiels für die Ausgabe der Spektralmessungen der 6 Kanäle:

---

<sup>1</sup>Nicht vergessen zuerst die korrekte virtuelle Environment mit den installierten Libraries über `source ~/.env/bin/activate` zu starten

```
#!/usr/bin/env python
from as7262 import AS7262
as7262 = AS7262()
as7262.set_gain(64)                                     ①
as7262.set_integration_time(17.857)
as7262.set_measurement_mode(2)
as7262.set_illumination_led(1)

try:
    while True:
        values = as7262.get_calibrated_values()
        print(""""
Red: {}
Orange: {}
Yellow: {}
Green: {}
Blue: {}
Violet: {}""".format(*values))

except KeyboardInterrupt:
    as7262.set_measurement_mode(3)
    as7262.set_illumination_led(0)
```

- ① Sensor konfigurieren, setzen des Gains, der Integrationszeit, des Messmodus und der Beleuchtungs LED
- ② Kalibrierte Messwerte auslesen und darstellen

### Übung 5.1. AS7262 Messwerte interpretieren

Führt einige Messungen mit unterschiedlichen Lichtquellen und Materialien durch und notiert Euch die Messwerte. Was beobachtet Ihr?

- Messung mit Tageslicht
- Messung mit LED-Taschenlampe des Smartphones
- Messung mit LED-Taschenlampe und Farbfilter (Mäppli)
- Messung mit weissem Papier

## 5.6. Aufgabe 2: Spektralmessungen mit Visualisierung in der Konsole

Das Beispiel `bargraph.py` zeigt die Messwerte der Spektralkanäle in einem Balkendiagramm an. Die Messwerte werden in der Konsole ausgegeben, nach einem Weissabgleich. Der Weissabgleich wird benötigt damit der Sensoren die Messwerte korrekt in

Referenz zu Weiss interpretiert.

Für diese Aufgabe benötigt Ihr ein weisse Blatt Papier, mit welchem der Sensor den Weissabgleich durchführt. Dafür hält ihr dieses circa 5cm vor den Sensor, drückt eine Taste und der Sensor führt den Weissabgleich durch. Danach könnt Ihr das Papier wieder wegnehmen und die Messungen durchführen.

**Übung 5.2. Verteilung der Spektralkanäle untersuchen**

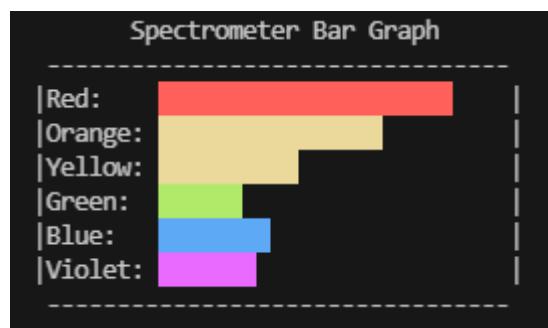
- Führt verschiedene Messungen durch mit unterschiedlichen Lichtquellen und Materialien und interpretiert die Ausgabe des `bargraph.py`.
- Studiert den Code und versucht die Funktionsweise zu verstehen.

```
python bargraph.py
```

```
Setting white point baseline.
```

```
Hold a white sheet of paper ~5cm in front of the sensor and press a key...
```

```
Baseline set. Press a key to continue...
```



**Abb. 5.3.:** Beispiel Ausgabe von `bargraph.py` Spectrometer Bar Graph

## Kapitel 6

# Bewegungsmessung

Bewegungssensoren werden heute fast allgegenwärtig in vielen Bereichen eingesetzt, wie in Smartphones oder auch vielen Messgeräten der Geomatik. Diese Übung führt in die Funktionsweise von MEMS Bewegungssensoren ein und zeigt wie Bewegungsmessungen mit einem ICM20948 9DoF Motion Sensor durchgeführt werden können.

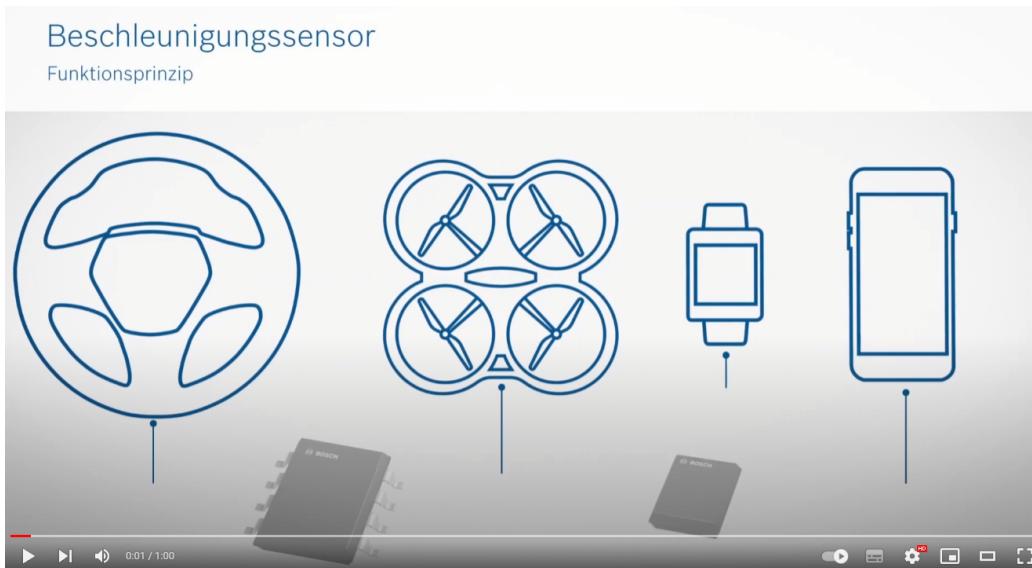
## 6.1. Einführung

Ziel dieser Übung ist es Bewegungsmessung mit inertialen Messeinheiten (IMU) über den ICM20948 Bewegungssensor kennen zu lernen und die Sensordaten auszulesen und testen. Der *ICM20948* ist ein 9DoF Bewegungssensor, der über eine I2C Schnittstelle mit dem Raspberry Pi verbunden wird und einer Python Library angesteuert werden kann.

**Unterlagen:** *E03\_Bewegungsmessung.zip*

### Vorbereitung

- Schaut das Video zur Funktionsweise von MEMS Bewegungssensoren
  - Video: [How MEMS Accelerometer Gyroscope Magnetometer Work](#) (bis Minute 2:50), sowie folgendes
  - Video: [Bosch Funktionsprinzip eines Beschleunigungssensors](#)
- Studiere das Datenblatt zum ICM-20948 ([InvenSense, 2021](#))
  - In welchen Temperaturbereichen kann der Sensor eingesetzt werden?
- Installiere auf deinem Smartphone die Applikation [phypox](#) und teste die Beschleunigungssensoren deines Smartphones.



**Abb. 6.1.:** How MEMS Accelerometer Gyroscope Magnetometer Work & Arduino

Tutorial [Youtube Video](#)

---

#### Unterlagen

---

Produkt [ICM20948 Breakout](#)

Datenblatt [ICM 20948](#)

GitHub [icm20948-python](#)

---

## 6.2. Beschleunigungssensoren IMU

Beschleunigungssensoren, oder inertiale Messeinheit (inertial measurement unit IMU) messen die Beschleunigung von Objekten mit dem Messprinzip der Trägheit und erfassen die Kraft die auf die Masse des Objekt wirkt, wenn dieses beschleunigt wird.

Für die Erfassung der sechs kinematischen Freiheitsgrade werden drei Achsen der Beschleunigung (Accelerometer) und drei Achsen der Rotation (Gyroskop) gemessen, die die Beschleunigungsmessung und Winkelgeschwindigkeit der Drehraten ausgeben. Für die Erfassung der Orientierung im Raum wird ein Magnetometer eingesetzt, welches die Ausrichtung des Objekts im Magnetfeld der Erde misst, um die Ausrichtung im Raum zu bestimmen. Wenn alle drei Sensoren kombiniert werden, spricht man von 9DoF Motion Sensoren, der neun Freiheitsgrade (9 Degrees of Freedom) misst.

Beschleunigungssensoren werden in vielen Anwendungen eingesetzt, wie z.B. in der Automobilindustrie (Auslösen von Airbags), der Luft- und Raumfahrt, der Medizintechnik (Beschleunigungssensoren in Herzschrittmachern) und der Unterhaltungselektronik (Smartphones für die Ausrichtung des Bildschirms).

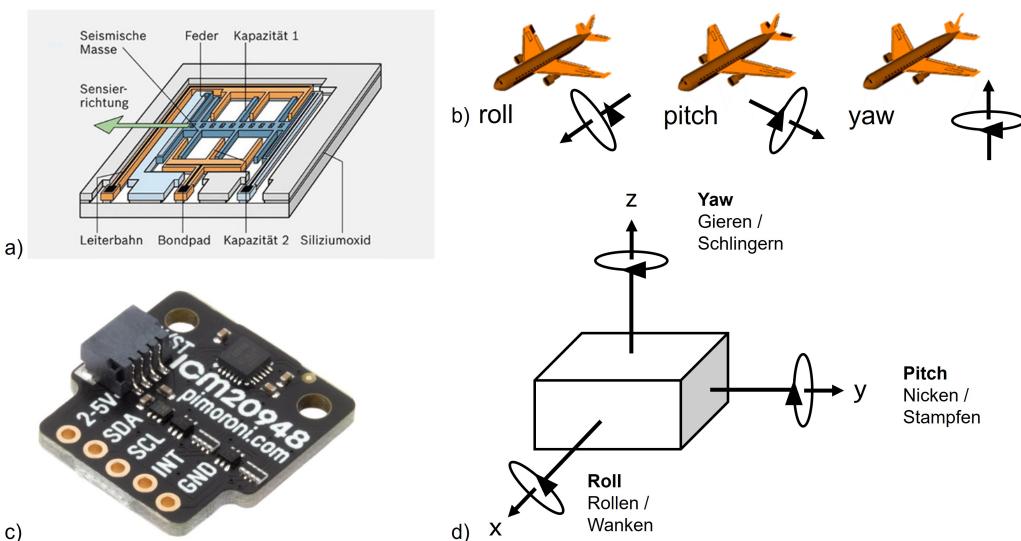
### 6.3. ICM20948 9DoF Motion Sensor

Der ICM-20948 von [TDK InvenSense](#) (Abb. 6.2) ist ein 9-Achsen MEMS Bewegungssensor, mit einem 3-Achsen Gyroskop, einem 3-Achsen Beschleunigungssensor und einem 3-Achsen Magnetometer und sehr geringem Stromverbrauch. Er enthält zwei Chip, einen für die Bewegungsmessung mit Gyroskop und Beschleunigungssensor und einen zweiten für das Magnetometer.

MEMS sind mikroelektromechanische Systeme, die aus mikroskopisch kleinen mechanischen und elektrischen Komponenten bestehen. Diese werden meist aus Silicium hergestellt und sind im Falle von Beschleunigungssensoren sehr kleine Massen, die sich bei Beschleunigung bewegen und die Änderung der elektrischen Kapazität messen.

9DoF Motion Accelero-, Gyro-, Magnetometer

- $\pm 2/\pm 4/\pm 8/\pm 16$  g 3-axis accelerometer
- $\pm 250/\pm 500/\pm 1000/\pm 2000$  DPS (degrees per second) 3-axis gyroscope
- 3-axis compass with wide range up to  $\pm 4900 \mu\text{T}$
- Python, C Library
- I2C interface (address: 0x68 or 0x69)
- Qwiic/STEMMA QT connector
- I2C interface (address 0x68/0x69 (cut trace))



**Abb. 6.2.:** a) schematische Darstellung eines MEMS Beschleunigungssensors Quelle: Bosch, b) roll, pitch, roll bei Flugzeugen, c) ICM-20948 Breakout von Pimoroni, d) Orientierung von IMU Sensoren.

## 6.4. Übungsaufbau

- Schliesse den Raspberry Pi an Monitor, Keyboard und Maus an oder verbinde Dich mit diesem über SSH (und SFTP).
- Erstelle auf dem Raspberry Pi im `Documents` Ordner einen neuen Ordner `ICM20948`, in welchem Du Änderungen und neue Dateien für diese Übung speichern kannst.
- Schliesse den Sensor **ICM20948** an den Raspberry Pi über die Breakout Garden **I2C** Schnittstelle korrekt an (siehe [E01 Luftqualität](#)), so dass die Beschriftung der Anschlüsse am Sensor und bei der Schnittstelle übereinstimmen.
- Kontrolliere mit dem Befehl `i2cdetect -y 1` ob der Raspberry Pi mit dem Sensor verbunden ist. Der Sensor sollte auf der Adresse `0x68` erkannt werden.
- Aktiviere die virtuelle Environment von Python mit `source ~/.env/bin/activate` und kontrolliere, ob die Library `icm20948` installiert ist mit `python -c "import icm20948"`. Bei einer Fehlermeldung muss die Library in der aktivierten virtuellen Environment mit `pip install icm20948` installiert werden.

Wechsle in den Ordner `Documents` und kopiere mit folgenden Befehlen die Library auf Deinen Raspberry Pi.

```
cd Documents
git clone https://github.com/pimoroni/icm20948-python
cd icm20948-python/examples
```

## 6.5. Aufgabe 1: Bewegungsmessungen durchführen

Teste das Beispiel `read-all.py` im Ordner `examples`. Dieses Beispiel gibt die Messungen der einzelnen Bewegungsmessungen aus, der Beschleunigung, Winkelgeschwindigkeit und Orientierung mit dem Accelerometer, Gyrometer und Magnetometer.

Startet das Script mit `python read-all.py`<sup>1</sup>. Mit `Ctrl+C` kann das Script wieder gestoppt werden. Die Ausgabe sollte in etwa so aussehen (gekürzt):

```
python read-all.py
read-all.py
Reads all ranges of movement: accelerometer, gyroscope and compass heading.
Press Ctrl+C to exit!

Accel: 01.01 -0.02 00.01
Gyro: -0.42 01.73 00.01
```

<sup>1</sup>Nicht vergessen zuerst die korrekte virtuelle Environment mit den installierten Libraries über `source ~/.env/bin/activate` zu starten

Mag: -86.85 57.45 34.05

Accel: 01.01 -0.02 00.02

Gyro: -0.40 01.50 -0.16

Mag: -85.35 55.50 34.80

Folgendes Code Snippet zeigt eine gekürzte Version des `read-all.py` Python Beispiels für die Ausgabe der Beschleunigungsmessung.

```
#!/usr/bin/env python
import time
from icm20948 import ICM20948

imu = ICM20948()

while True:
    x, y, z = imu.read_magnetometer_data()                                ①
    ax, ay, az, gx, gy, gz = imu.read_accelerometer_gyro_data()           ②

    print(""" # <3>
Accel: {:05.2f} {:05.2f} {:05.2f}
Gyro:  {:05.2f} {:05.2f} {:05.2f}
Mag:   {:05.2f} {:05.2f} {:05.2f}""".format(
        ax, ay, az, gx, gy, gz, x, y, z
    ))                                                                      ③

    time.sleep(0.25)                                                       ④
```

- ① Auslesen des Magnetometers (x,y,z)
- ② Auslesen des Accelerometers (ax,ay,az) und Gyrometers (gx,gy,gz)
- ③ Messwerte auf der Konsole ausgeben
- ④ Warten 0.25 Sekunden (damit die Ausgabe nicht zu schnell ist)

## 6.6. Aufgabe 2: Magnetometer

Das Beispiel `bargraph.py` zeigt die Messwerte des Magnetometers in einem Balkendiagramm an und zeigt die Orientierung des Sensors an je nach dem über welche Achse gemessen wird. Der Befehl `python bargraph.py --help` zeigt die Optionen des Skripts an.

Die Ausgabe sollte für die Option `--graph` in etwa so aussehen:

## Übung 6.1. Bewegungsmessung

- Führe das Beispiel `read-all.py` aus und beobachte die Messwerte.
- Versuche den Sensor jeweils leicht in eine Richtung zu bewegen, zu drehen, zu kippen und beobachte die Messwerte.
- Versuche den Sensor zu leicht zu schütteln und beobachte die Messwerte.
- Vergleiche die Messwerte mit den Messwerten deines Smartphones mit der App *phyphox*.
- Versucht zu eruieren wie die Achsen orientiert sind und vergleicht mit anderen Gruppen.
- Schreibe die Messwerte in eine Datei und visualisiere diese mit einem Plot, modifiziere dazu das Beispiel `read-all.py` und speichere die Datei als `read-csv.py`, so dass die Messungen zeilenweise mit einem Separator gespeichert werden. Ausgaben aus einem Plot können mit dem Befehl `python read-csv.py > imu_horizontal.csv` in eine Datei geschrieben werden. Nun könnt ihr verschiedene Versuche mit der IMU durchführen und in einer Datei speichern. Die Datei könnt ihr beispielsweise mit *LibreOffice Calc* oder *Excel* öffnen und die Daten visualisieren.

```
python bargraph.py --graph
bargraph.py - Convert raw values to heading

Rotate the sensor through 360 degrees to calibrate.

Press Ctrl+C to exit!

043.5 [██████████]
```

## Übung 6.2. Bar graph

- Kalibriere den Sensor und vergleiche die Orientierung des Sensors mit den Himmelsrichtungen
- Vergleiche die Ausgaben auch mit der Orientierung des Smartphones und den Messwerten der App *phyphox*.
- Studiert den Code und versucht die Funktionsweise zu verstehen.

## Kapitel 7

# Distanzmessung

Distanzmessungen sind essentiell im Bereich der Geomatik. Unterschiedliche Messverfahren ermöglichen Distanzmessungen unter anderem Time-of-Flight Sensoren, die in dieser Übung praktisch genutzt und getestet werden.

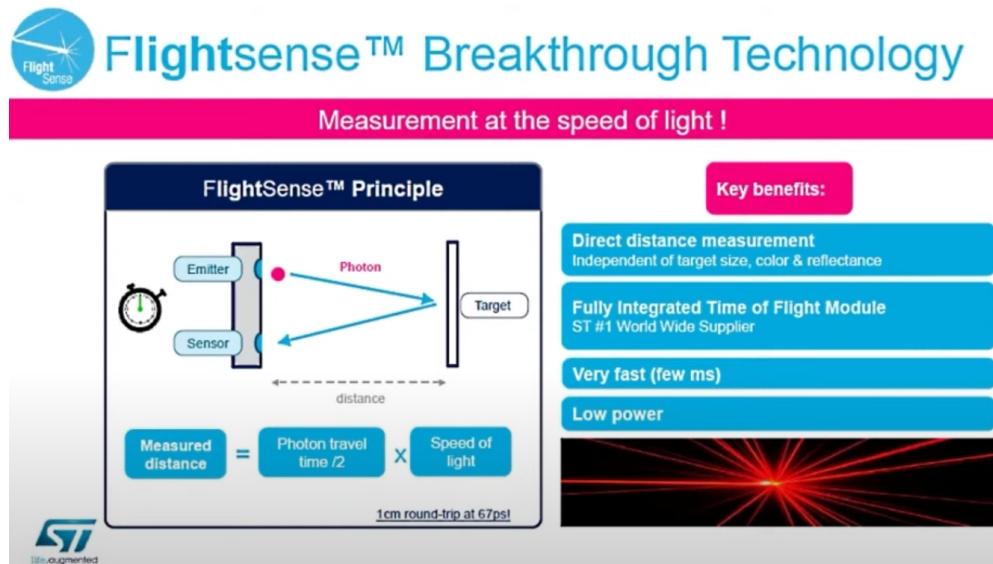
## 7.1. Einführung

Ziel dieser Übung ist es Distanzmessungen mit dem VL53L5CX Sensor kennen zu lernen und die Sensordaten auszulesen und testen. Der *VL53L5CX* ist ein 8x8 Time of Flight (ToF) Array Sensor, der über eine I2C Schnittstelle mit dem Raspberry Pi verbunden wird und einer Python Library angesteuert werden kann.

**Unterlagen:** *E04\_Distanzmessung.zip*

### Vorbereitung

- Schaut folgende von Video von Adafruit Industries zur Funktionsweise des VL53L5CX Sensors an: [EYE ON NPI - ST VL53L5CX Time-of-Flight Ranging Sensor](#)
- Studiere das Datenblatt zum VL53L5CX ([STMicroelectronics, 2021](#)), sowie die *Application Note* ([STMicroelectronics, 2023](#))
  - In welchen Temperaturbereichen kann der Sensor eingesetzt werden?
  - Welches ist die höchste Abtastrate für die Distanzfeldmessungen?
  - Was sind Anwendungsgebiete für diesen Sensor?



**Abb. 7.1.:** Adafruit - EYE ON NPI - ST VL53L5CX Time-of-Flight Ranging Sensor

[Youtube Video](#)

### Unterlagen

Produkt [VL53L5CX Breakout](#)

Datenblatt [VL53L5CX](#)

GitHub [vl53l5cx-python](#)

## 7.2. VL53L5CX 8x8 Time of Flight (ToF) Array Sensor

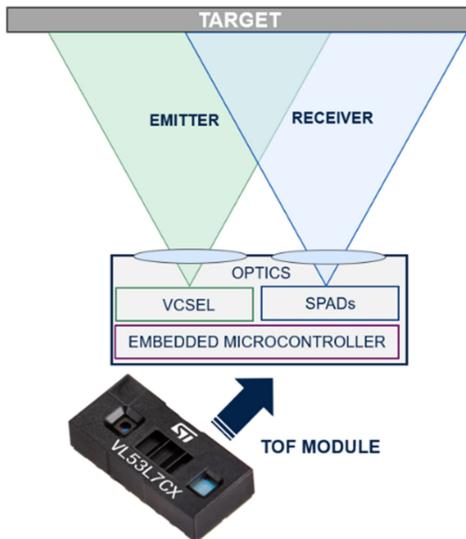
Der VL53L5CX ist hochentwickelter Distanzsensor mit einer 8x8-Multizonenmessung und einem großen Sichtfeld, ideal für Roboter und fortschrittliche Bewegungserkennung. Der misst die Entfernung mit Time of Flight (ToF), also mit der Laufzeit von Licht, indem er einen Infrarotlaser mit geringer Leistung auf ein Ziel schickt und die Zeit misst, die das Licht benötigt, um zurückzukehren.

Dieser Sensor hat eine hohe Genauigkeit und Abtastfrequenz (bis zu 60 Hz) und einen großen Erfassungsbereich (von 2 cm bis 4 m). Besonders interessant ist, dass der Sensor nicht nur eine Messung durchführt, sondern eine 8x8 Matrix mit Messwerten zurückgibt. Das bedeutet, dass Bewegungen aus bestimmten Richtungen erkannt werden können oder der Sensor benutzt werden kann um Kollisionen zu vermeiden oder Objekte zu verfolgen ohne dass mehrere Sensoren benötigt werden.

VL53L5CX 8x8 Time of Flight (ToF) Array Sensor Breakout

– 8x8 Multizone readings

- Distance 2cm - 4m
- I2C interface, with address: 0x52
- Python, C Library



**Abb. 7.2.:** links: VL53L5CX Breakout von Pimoroni, rechts: schematische Darstellung TOF Moduls Quelle: STMicroelectronics (2023)

### 7.3. Übungsaufbau

- Schliesse den Raspberry Pi an Monitor, Keyboard und Maus an oder verbinde Dich mit diesem über SSH (und SFTP).
- Erstelle auf dem Raspberry Pi im `Documents` Ordner einen neuen Ordner `VL53L5CX`, in welchem Du Änderungen und neue Dateien für diese Übung speichern kannst.
- Schliesse den Sensor **VL53L5CX** an den Raspberry Pi über die Breakout Garden **I2C** Schnittstelle korrekt an (siehe [E01 Luftqualität](#)), so dass die Beschriftung der Anschlüsse am Sensor und bei der Schnittstelle übereinstimmen.
- Kontrolliere mit dem Befehl `i2cdetect -y 1` ob der Raspberry Pi mit dem Sensor verbunden ist. Der Sensor sollte auf der Adresse `0x29` erkannt werden.
- Aktiviere die virtuelle Environment von Python mit `source ~/.env/bin/activate` und kontrolliere, ob die Libraries `vl53l5cx_ctypes` und `st7789` installiert sind mit `python -c "import st7789"` und `python -c "import vl53l5cx_ctypes"`. Bei einer Fehlermeldung muss die jeweilige fehlende Library in der aktivierte virtuellen Environment mit `pip install st7789` oder `pip install vl53l5cx_ctypes` installiert werden.

Wechsle in den Ordner `Documents` und kopiere mit folgenden Befehlen die Library auf Deinen Raspberry Pi.

```
cd Documents
git clone https://github.com/pimoroni/vl53l5cx-python
cd vl53l5cx-python/examples
```

## 7.4. Aufgabe 1: Distanzmessung Konsole

Teste das Beispiel `test.py` im Ordner `examples`. Dieses Beispiel liest die Werte der 8x8 Time of Flight Messung aus mit Werten zu `motion`, `distance`, `reflectance` und `status` aus.

### Hinweis

Das Script startet langsam, da die Library jeweils die Firmware beim Starten lädt.

Startet das Script mit `python test.py`. Mit `Ctrl+C` kann das Script wieder gestoppt werden. Die Ausgabe sollte in etwa so aussehen (gekürzt):

```
python3 test.py
Uploading firmware, please wait...
Done!
[[ 36   26 2356  543]
 [ 38   62 1943  3847]
 [ 27   68  530  6744]
 [ 14   18   66  458]] [[1254   308   406 2042   365   377   314   237]
[1275   351   397   413   404   403   354   228]
[1297   357   375   432   427   422   391   241]
[1250   348   385   389   415   437   398   315]
[1273   358   358   385   405   429   400   363]
[1238   336 2240   368   424   417   379   336]
[1262   226 2215   180   188   218   202   190]
[ 108   110   113  117   116   120   120 124]] [[23   1   2 30 11   8 27 41]
[16   1   3   4   4 17 43 35]
[23   1   2   6   5 21 58 32]
[21   1   2   4   6 32 62 52]
[27   1   3   5   9 36 65 71]
[27   1 59   5 16 47 52 46]
[26   1 39   6   9 20 20 21]
[13 13 14 14 15 16 17 20]] [[False False False  True False False  True  True]
[ True False False False  True  True  True]
[ True False False False False  True  True  True]
[ True False False False False  True  True  True]
[ True False  True False  True  True  True]
[ True False False  True  True  True  True]
[ True  True  True  True  True  True  True]]
```

Folgendes Code Snippet zeigt eine gekürzte Version des `test.py` Python Beispiels für die Ausgabe der Distanzmatrix.

```
import time
import numpy
import vl53l5cx_ctypes as vl53l5cx
from vl53l5cx_ctypes import STATUS_RANGE_VALID, STATUS_RANGE_VALID_LARGE_PULSE

print("Uploading firmware, please wait...")
vl53 = vl53l5cx.VL53L5CX()                                     ①
print("Done!")
vl53.set_resolution(8 * 8)                                       ②
vl53.enable_motion_indicator(8 * 8)
# vl53.set_integration_time_ms(50)
# Enable motion indication at 8x8 resolution
vl53.enable_motion_indicator(8 * 8)
# Default motion distance is quite far, set a sensible range
# eg: 40cm to 1.4m
vl53.set_motion_distance(400, 1400)
vl53.start_ranging()                                              ③

while True:
    if vl53.data_ready():
        data = vl53.get_data()                                     ④
        # 2d array of motion data (always 4x4?)
        motion =
        ↵ numpy.flipud(numpy.array(data.motion_indicator.motion[0:16]).reshape((4,
        ↵ 4)))
        # 2d array of distance
        distance = numpy.flipud(numpy.array(data.distance_mm).reshape((8, 8)))
        # 2d array of reflectance
        reflectance = numpy.flipud(numpy.array(data.reflectance).reshape((8,
        ↵ 8)))
        # 2d array of good ranging data
        status =
        ↵ numpy.isin(numpy.flipud(numpy.array(data.target_status)).reshape((8, 8)),
        ↵ (STATUS_RANGE_VALID, STATUS_RANGE_VALID_LARGE_PULSE))
        print(motion, distance, reflectance, status)
        time.sleep(0.1)                                            ⑤
```

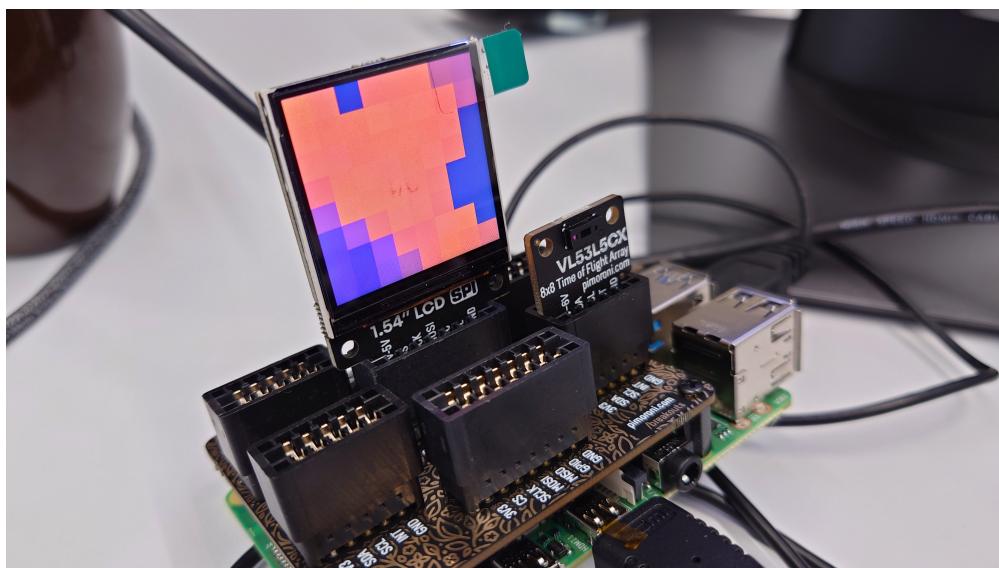
- ① Sensor initialisieren und Firmware laden
- ② Sensor konfigurieren (Auflösung, Bewegungserkennung, Messbereich)
- ③ Messung initialisieren
- ④ Warten 0.1 Sekunden (damit die Ausgabe nicht zu schnell ist)

### Übung 7.1. Bewegungsmessung

- Führe das Beispiel `test.py` aus und beobachte die Messwerte.
- Führe unterschiedliche Tests durch, indem Du ein Objekt vor den Sensor hältst und bewegst.
- Vergleiche die Messwerte und kontrolliere die gemessenen Distanzen.
- Untersuche die einzelnen Matrizen und versuche die Bedeutung der einzelnen Werte zu verstehen.

## 7.5. Aufgabe 2: Distanzmessung mit LCD Bildschirm

Folgende Aufgabe nutzt den 1.54" LCD Bildschirm mit einer 240x240 Pixel Auflösung. Die Library `vl53l5cx_ctypes` enthält mehrere Beispiele, die die Distanzmatrizen für die *Distanz-, Bewegungs- und Reflektanzmessung* auf dem Bildschirm anzeigen. Die Beispiele sind im Ordner `examples` zu finden.



**Abb. 7.3.:** Aufbau der Versuchsanordnung für die Distanzmessung mit dem LCD Bildschirm montiert im hinteren SPI Slot

### Vorbereitung

- Kontrolliert mit `python -c "import st7789"` ob die Library `st7789` installiert ist. Testet auch, ob die Bibliotheken `numpy` und `matplotlib` installiert sind und installiert diese ansonsten mit `sudo apt install python3-matplotlib python3-numpy`.
- Kontrolliere, ob der Raspberry Pi den *Breakout Garden HAT* mit den 2 SPI Anschlüssen und 4 I2C Anschlüssen bestückt ist (Abb. 7.3).
- Montiere den Bildschirm im hinteren SPI Slot des *Breakout Garden HATs* wie in Abb. 7.3, da er sonst die Messung des VL53L5CX Sensors verdeckt.

Führt nun folgende Scripts aus und beobachtet die Ausgabe auf dem LCD Bildschirm.  
Auch hier braucht es etwas Geduld, da die Firmware beim Starten geladen wird.

- python distance\_240x240\_lcd.py
- python motion\_240x240\_lcd.py
- python reflectance\_240x240\_lcd.py
- python object\_tracking.py

### Übung 7.2. Distanzmessung

- Führe die Beispiele aus und beobachte die Ausgabe auf dem LCD Bildschirm.
- Untersuche die einzelnen Matrizen und versuche die Bedeutung der einzelnen Werte zu verstehen.
- Führe unterschiedliche Tests durch, indem Du ein Objekt vor den Sensor hältst und bewegst.
- Was passiert wenn Du ein Objekt vor den Sensor hältst? Ist die Form erkennbar?
- Was passiert wenn Du ein Objekt vor den Sensor hältst und bewegst?
- Studiere den Code der Beispiele und versuche die Funktionsweise zu verstehen.
- Überlege Dir Anwendungsfälle für diesen Sensor allgemein und im speziellen für die Geomatik.

## Kapitel 8

# Thermale Aufnahmen

In der Fernerkundung sind Thermalkameras ein wichtiges Instrument zur Erfassung von Wärmebildern, beispielsweise für die Zählung von Tierbeständen oder auch für Untersuchungen von Gebäudeisolationen. Diese Übung nutzt die MLX90640 Thermalkamera und zeigt wie thermale Aufnahmen mit dem Raspberry Pi gemacht werden können.

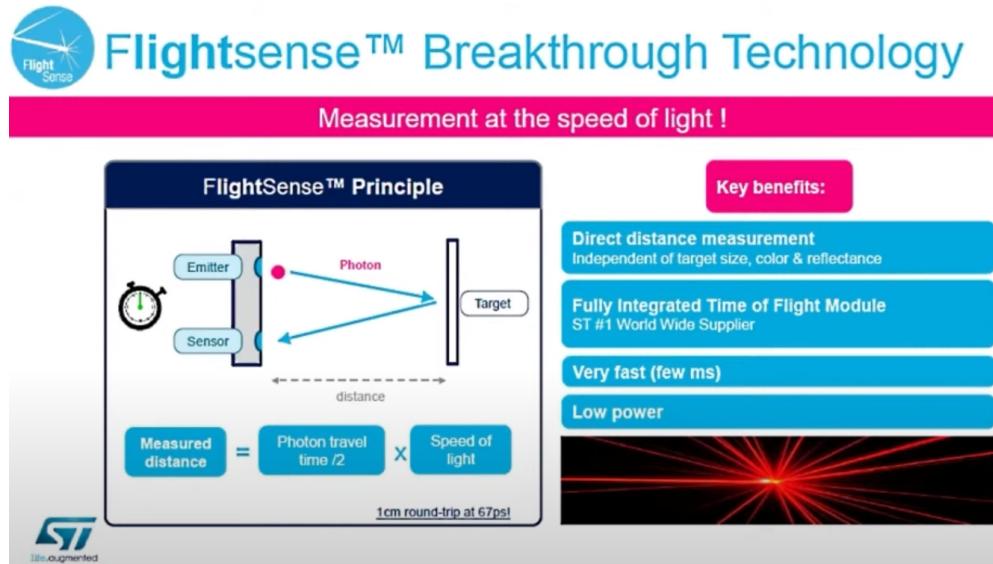
### 8.1. Einführung

Ziel dieser Übung ist es thermale Kamera mit der MLX90640 Kamera kennen zu lernen und die thermalen Bilddaten auszulesen und testen. Die *MLX90640* ist eine weitwinkel Kamera mit einer Auflösung von 24x32 Pixel. Sie wird über eine I2C Schnittstelle mit dem Raspberry Pi verbunden und kann über eine Python Library angesteuert werden.

**Unterlagen:** *E05\_Thermalkamera.zip*

#### Vorbereitung

- Schaut folgende von Video von Melexis zur Funktionsweise der MLX90640 Thermal-kamera an: [Far Infrared \(IR\) Thermal Sensor Array 32x24 RES \(MLX90640\)](#)
- Studiere das Datenblatt zum VL53L5CX ([Melexis, 2019](#)) und beantworte folgende Fragen:
  - In welchen Temperaturbereichen kann der Sensor eingesetzt werden?
  - Welches ist die höchste Abtastrate für die Distanzfeldmessungen?
  - Was sind Anwendungsgebiete für diesen Sensor?



**Abb. 8.1.:** Far Infrared (IR) Thermal Sensor Array 32x24 RES (MLX90640) [Youtube Video](#)

### Unterlagen

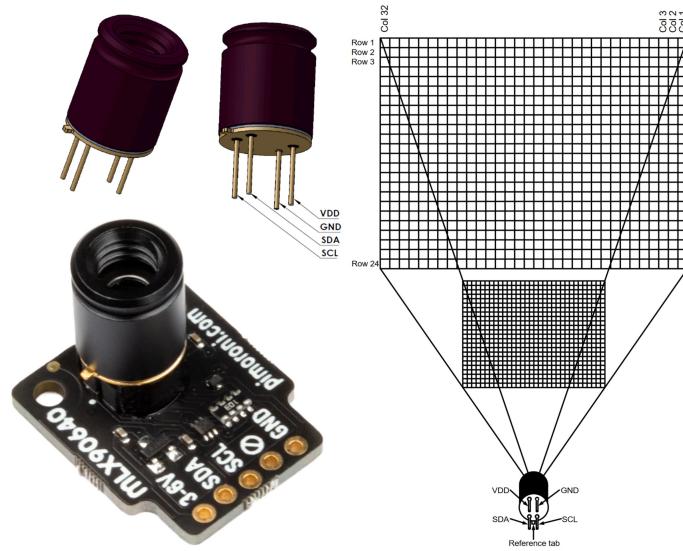
Produkt	MLX90640 Breakout
Datenblatt	MLX90640
GitHub	<a href="#">mlx90640-library</a> , Adafruit MLX90640

## 8.2. MLX90640 Thermalkamera

Der MLX90640 ist eine Thermalkamera mit einer Auflösung von 32x24 Pixel mit einem Sichtfeld von 55°. Die Kamera misst in einem Temperaturbereich von -40° - 300°C mit einer Genauigkeit von etwa 1°C und mit einer Aufnahmerate von bis zu 64 FPS. Die Anwendungsbereiche sind vielfältig, von der Temperatur der Kaffeetasse, Hitzeentwicklung in elektronischen Geräten bis hin zur Überwachung von Gebäuden und Anlagen.

### MLX90640 Thermal Camera

- Melexis MLX90640 far-infrared sensor array
- Brennweite: 55°
- 32x24 pixels
- I2C interface (address 0x33)



**Abb. 8.2.:** links: MLX90640 Thermalkamera, oben: schematische Darstellung der Kamera und ihrer Anschlüsse rechts: Pixelpositionaufbau Quelle: STMicroelectronics (2023)

### 8.3. Übungsaufbau

- Schliesse den Raspberry Pi an Monitor, Keyboard und Maus an oder verbinde Dich mit diesem über SSH (und SFTP).
- Erstelle auf dem Raspberry Pi im `Documents` Ordner einen neuen Ordner `MLX90640`, in welchem Du Änderungen und neue Dateien für diese Übung speichern kannst.
- Schliesse den Sensor **MLX90640** an den Raspberry Pi über die Breakout Garden **I2C** Schnittstelle korrekt an (siehe [E01 Luftqualität](#)), so dass die Beschriftung der Anschlüsse am Sensor und bei der Schnittstelle übereinstimmen.
- Kontrolliere mit dem Befehl `i2cdetect -y 1` ob der Raspberry Pi mit dem Sensor verbunden ist. Der Sensor sollte auf der Adresse `0x33` erkannt werden.
- Aktiviere die virtuelle Environment von Python mit `source ~/.env/bin/activate` und kontrolliere, ob die Libraries `adafruit-blinka`, `adafruit-circuitpython-mlx90640` und `RPI.GPIO` installiert sind mit `python -c "import adafruit-blinka"`, `python -c "import adafruit-circuitpython-mlx90640"` und `python -c "RPI.GPIO"`. Bei einer Fehlermeldung müssen die fehlenden Libraries in der aktivierte virtuellen Environment mit `pip install adafruit-blinka adafruit-circuitpython-mlx90640 RPI.GPIO` installiert werden.
- Die Python Scripts für die Thermalkamera werden in den *Unterlagen* zu dieser Übung bereitgestellt.

Wechsle in den Ordner `Documents` und kopiere in diesen den Ordner `MLX90640` mit den Code Beispielen.

## 8.4. Aufgabe 1: Punktuelle Temperaturmessung

Teste das Beispiel `average_temperature.py` im Ordner `examples`. Dieses Beispiel liest die Werte aus der Matrix der Thermalkamera und gibt den Mittelwert der Temperatur aus. Die Ausgabe sollte in etwa so aussehen (gekürzt):

```
python average_temperature.py
Average MLX90640 Temperature: 23.6C (74.5F)
```

Folgendes Code Snippet zeigt eine gekürzte Version des `average_temperature.py` Python Beispiels für die Ausgabe der Distanzmatrix.

```
import time,board,busio
import numpy as np
import adafruit_mlx90640

i2c = busio.I2C(board.SCL, board.SDA)                                     ①
mlx = adafruit_mlx90640.MLX90640(i2c)
mlx.refresh_rate = adafruit_mlx90640.RefreshRate.REFRESH_2_HZ

frame = np.zeros((24*32,))                                                 ②
while True:
    try:
        mlx.getFrame(frame)
        break
    except ValueError:
        continue

# print out the average temperature from the MLX90640
print('Average MLX90640 Temperature: {0:2.1f}C ({1:2.1f}F)'.\
      format(np.mean(frame), (((9.0/5.0)*np.mean(frame))+32.0)))          ⑤
```

- ① Setup von I2C, Initialisierung des MLX90640 Sensors und Setzen der Abtastrate auf 2 Hz
- ② Numpy Array für das Speichern der 768 Temperaturwerte (24x32 Pixel) erstellen
- ③ MLX Temperaturwerte in das Numpy Array speichern
- ④ Falls ein Fehler eintritt, nochmals versuchen den Sensor auszulesen
- ⑤ Temperaturmittelwert ausgeben

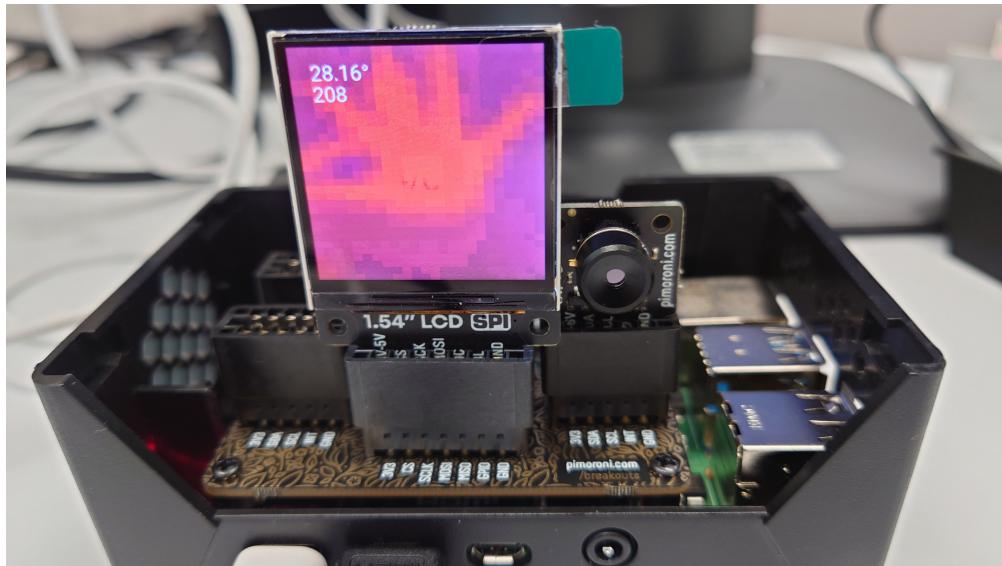
## 8.5. Aufgabe 2: Thermale Aufnahme mit LCD Bildschirm

Folgende Aufgabe nutzt den 1.54" LCD Bildschirm mit einer 240x240 Pixel Auflösung. Der Ordner enthält mehrere Beispiele zur Thermalkamera, die die Temperaturmatrix auf

### Übung 8.1. Einzelne Temperaturmessung

- Führe das Beispiel `average_temperature.py` aus und beobachte die Messwerte.
- Teste das Script mit unterschiedlichen warmen Objekten (z.B. Kaffeetasse, Hand, etc.).

dem Bildschirm anzeigen. Die Beispiele sind im Ordner `MLX90640` zu finden.



**Abb. 8.3.:** Aufbau der Versuchsanordnung für die Distanzmessung mit dem LCD Bildschirm montiert im *hinteren* SPI Slot

### Vorbereitung

- Kontrolliert mit `python -c "import st7789"` ob die Library `st7789` installiert ist. Installiere die Library mit in der aktivierten virtuellen Environment `source ~/.env/bin/activate` mit `pip install st7789`, falls sie nicht installiert ist. Testet auch, ob die Bibliotheken `numpy` und `matplotlib` installiert sind und installiert diese ansonsten mit `sudo apt install python3-matplotlib python3-numpy`.
- Kontrolliere, ob der Raspberry Pi den *Breakout Garden HAT* mit den 2 SPI Anschlüssen und 4 I2C Anschlüssen bestückt ist (Abb. 8.3).
- Montiere den Bildschirm im vorderen SPI Slot des *Breakout Garden HATs* wie in Abb. 8.3, oder passe das Script, an so dass die Werte der hinteren SPI Schnittstelle verwendet werden.
- Teste die Beispiele im Ordner `MLX90640` und beobachte die Ausgabe auf dem LCD Bildschirm.

**Übung 8.2. Experimente mit der Thermalkamera und dem LCD Bildschirm**

- Teste mit unterschiedlichen Objekten die Temperaturmessung und beobachte die Ausgabe auf dem LCD Bildschirm.
- Wie gut lassen sich Objekte mit unterschiedlichen Temperaturen unterscheiden?
- Wie gut lassen sich Objekte erkennen, die sich in der Temperatur nur wenig oder stark unterscheiden?
- Studiere den Code der Beispiele und versuche die Funktionsweise zu verstehen.
- Überlege Dir, wie Du die Beispiele erweitern könntest.

## Kapitel 9

# Biometrie

Biometrische Messungen ermöglichen die Erfassung von physiologischen Daten wie Herzfrequenz oder Sauerstoffsättigung. Während diese Sensoren nicht typische Sensoren für IoT Projekte sind, können sie für spezielle Anwendungen wie Gesundheitsüberwachung oder Fitness eingesetzt werden. Diese Übung führt in die biometrischen Messungen mit dem MAX30101 Sensor ein und zeigt wie die Herzfrequenz gemessen werden kann.

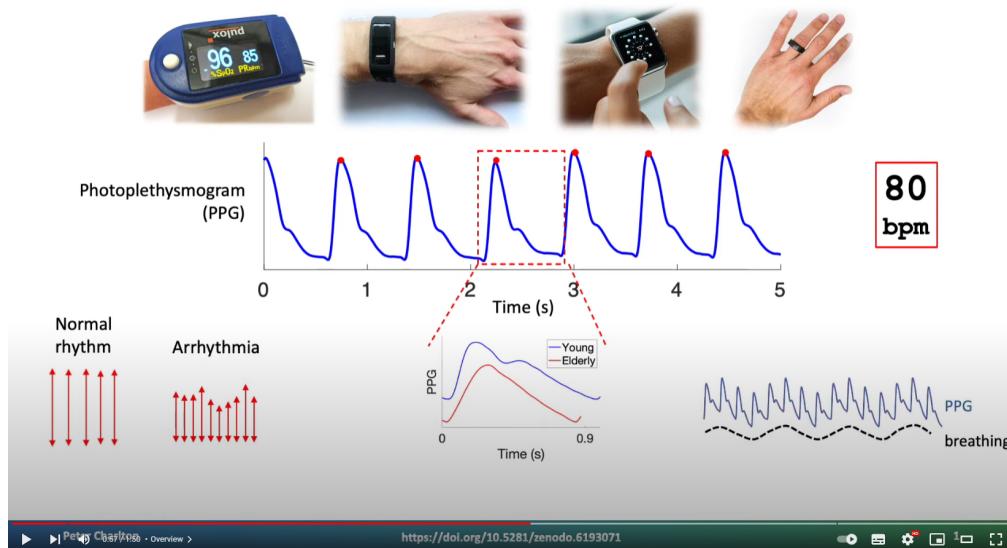
### 9.1. Einführung

Ziel dieser Übung ist es biometrische Messungen mit dem *MAX30101* Sensor durchzuführen und den Sensor in seiner funktionsweise zu untersuchen. Der *MAX30101* ist ein Sensor zur Messung der Herzfrequenz und der Sauerstoffsättigung im Blut und ein Rauch-/Partikelsensor. Er verfügt über eine I2C Schnittstelle und kann über eine Python Library angesteuert werden.

**Unterlagen:** *E06\_Biometrie.zip*

#### Vorbereitung

- Schaut Euch folgendes von Video von Peter Charlton zur Funktionsweise der Herzfrequenzmessung an: [Photoplethysmography in a minute \(and a bit\) - Peter Charlton](#)
- Studiere das Datenblatt zum MAX30101 ([maxim integrated, 2020](#))
  - In welchen Temperaturbereichen kann der Sensor eingesetzt werden?
  - Welches ist die höchste Abtastrate für die Sauerstoffmessungen?
  - Was sind Anwendungsgebiete für diesen Sensor?



**Abb. 9.1.:** Photoplethysmography in a minute (and a bit) - Peter Charlton [Youtube Video](#)

---

### Unterlagen

---

Produkt [MAX30101 Breakout](#)

Datenblatt [MAX30101](#)

GitHub [max30105-python](#)

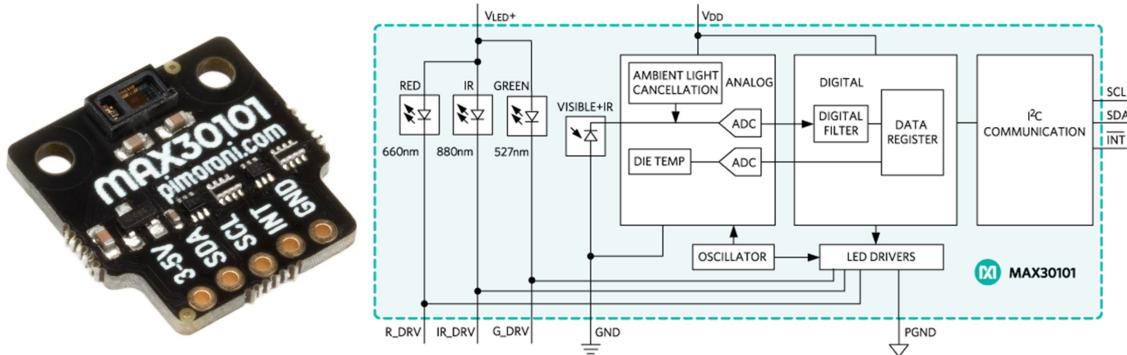
---

## 9.2. MAX30101 Breakout Heart Rate, Oximeter, Smoke Sensor

Der MAX30101 ist hochentwickelter Herzfrequenz-, Oximeter- und Rauch-/Partikelsensor. Der Sensor verfügt über drei LEDs (rot, grün, IR) und Photodetektoren. Mit der Photoplethysmographie (photoplethysmography PPG) kann über die Farbveränderung der Haut bei jedem Herzschlag dieser detektiert werden, wenn der Sensor leicht auf den Finger gedrückt wird. Der Sensor kann auch dazu benutzt werden um Partikel in der Luft wie rauch zu erkennen, in dem er die Lichtmenge, die von Partikeln in der Luft zurückgeworfen wird misst.

MAX30101 Breakout - Heart Rate, Oximeter, Smoke Sensor Breakout

- MAX30101 - heart rate, oximeter, smoke sensor
- Green, red, and infra-red LEDs
- Photodetectors
- Ambient light rejection
- Temperature sensor



**Abb. 9.2.:** links: MAX30101 Breakout von Pimoroni, rechts: funktionale Diagramm des MAX30101 Moduls Quelle: maxim integrated (2020)

### 9.3. Übungsaufbau

- Schliesse den Raspberry Pi an Monitor, Keyboard und Maus an oder verbinde Dich mit diesem über SSH (und SFTP).
- Erstelle auf dem Raspberry Pi im `Documents` Ordner einen neuen Ordner `MAX30101`, in welchem Du Änderungen und neue Dateien für diese Übung speichern kannst.
- Schliesse den Sensor **MAX30101** an den Raspberry Pi über die Breakout Garden **I<sup>2</sup>C** Schnittstelle korrekt an (siehe [E01 Luftqualität](#)), so dass die Beschriftung der Anschlüsse am Sensor und bei der Schnittstelle übereinstimmen.
- Kontrolliere mit dem Befehl `i2cdetect -y 1` ob der Raspberry Pi mit dem Sensor verbunden ist. Der Sensor sollte auf der Adresse `0x57` erkannt werden.
- Aktiviere die virtuelle Environment von Python mit `source ~/.env/bin/activate` und kontrolliere, ob die Library `v15315cx_ctypes` installiert ist mit `python -c "import v15315cx_ctypes"`. Bei einer Fehlermeldung muss die Library in der aktivierte virtuellen Environment mit `pip install v15315cx_ctypes` installiert werden.

Wechsle in den Ordner `Documents` und kopiere mit folgenden Befehlen die Library auf Deinen Raspberry Pi.

```
cd Documents
git clone https://github.com/pimoroni/max30105-python
cd max30105-python/examples
```

### 9.4. Aufgabe 1: Biometriemessung Konsole

Teste das Beispiel `read-heartbeat.py` im Ordner `examples`. Dieses Beispiel liest die Pulsschläge pro Minute in PPM (beats per minute). Ein erkannter Pulsschlag wird mit

einem <3 angezeigt.

Startet das Script mit `python read-heartbeat.py`. Mit `Ctrl+c` kann das Script wieder gestoppt werden. Die Ausgabe sollte in etwa so aussehen (gekürzt):

```
python read-heartbeat.py
NOTE! This code should not be used for medical diagnosis.

...
Starting readings in 10 seconds...

BPM: 0.00 AVG: 0.00
BPM: 0.00 AVG: 0.00
BPM: 0.00 AVG: 0.00
BPM: 0.00 AVG: 0.00
BPM: 45.55 AVG: 45.96
<3 BPM: 55.75 AVG: 59.90
    BPM: 55.75 AVG: 59.90
<3 BPM: 59.64 AVG: 70.65
    BPM: 59.64 AVG: 70.65
```

Folgendes Code Snippet zeigt eine gekürzte Version des `read-heartbeat.py` Python Beispiels für die Ausgabe des Herzschlags.

```
#!/usr/bin/env python

# NOTE! This code should not be used for medical diagnosis. It's
# for fun/novelty use only, so bear that in mind while using it.

import time
from max30105 import MAX30105, HeartRate

max30105 = MAX30105()                                     ①
max30105.setup(leds_enable=2)                                ②
max30105.set_led_pulse_amplitude(1, 0.2)
max30105.set_led_pulse_amplitude(2, 12.5)
max30105.set_led_pulse_amplitude(3, 0)
max30105.set_slot_mode(1, 'red')
max30105.set_slot_mode(2, 'ir')
max30105.set_slot_mode(3, 'off')
max30105.set_slot_mode(4, 'off')

def display_heartrate(beat, bpm, avg_bpm):                  ③
    print("{} BPM: {:.2f} AVG: {:.2f}".format("<3" if beat else "  ",
                                                bpm, avg_bpm))
hr = HeartRate(max30105)                                    ④
```

```
delay = 10  
print("Starting readings in {} seconds...\n".format(delay))  
time.sleep(delay)  
  
try:  
    hr.on_beat(display_heartrate, average_over=4)  
except KeyboardInterrupt:  
    pass
```

- ① Sensor initialisieren
- ② Sensor konfigurieren (LED Pulse Amplitude, Slot Mode der LED)
- ③ Funktion zur Darstellung des Herzschlags in der Konsole
- ④ Initialisierung des Herzschlagdetektors
- ⑤ 10 Sekunden warten und dann die Herzschläge ausgeben
- ⑥ Bei einem erkannten Herzschlag wird die Funktion `display_heartrate` aufgerufen und der Herzschlag wird über 4 Sekunden gemittelt in der Konsole ausgegeben.

### Übung 9.1. Pulsmessung

- Führe das Beispiel `read-heartbeat.py` aus und beobachte die Messwerte.
- Führe unterschiedliche Tests durch und behandle den Sensor freundlich.
- Vergleiche die Messwerte und kontrolliere die Werte in dem Du den eigenen Puls misst oder mit einem Sportuhr vergleichst.
- Studiere den Code der Beispiele und versuche die Funktionsweise zu verstehen.

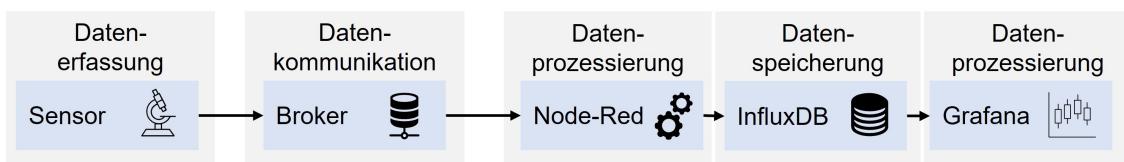
## Kapitel 10

# IoT Monitoring

IoT Monitoring ermöglicht die Erfassung von Sensordaten, die Kommunikation und Visualisierung von Daten in Echtzeit. Dies erfordert eine Architektur, die die Daten erfasst, Echtzeitkommunikation, asynchroner Kommunikation, Datenspeicherung und Visualisierung ermöglicht. Diese Übung baut einen typischen IoT Stack auf mit MQTT, Node-Red, InfluxDB und Grafana und zeigt wie Sensordaten über MQTT publiziert und abonniert werden können.

### 10.1. Einführung

Ein IoT Datenfluss erstreckt sich über verschiedene Instanzen, die für die einzelnen Prozesse zuständig sind, von der Erfassung von Sensormesswerten im IoT Gerät über die Kommunikation der Messwerte bis zur Datenverarbeitung, Speicherung und Visualisierung (Abb. 10.1). Hierbei können alle Schritte auf einem Gerät durchgeführt werden oder jeder einzelne über ein anderes Gerät oder Server. In der Abbildung aufgezeigt sind typische Softwarekomponenten, die in IoT eingesetzt werden, wie Node-Red für die Datenverarbeitung, die InfluxDB Datenbank, welche für das Speichern von Zeitreihendaten entwickelt wurde und Grafana eine Visualisierungsplattform, die für Messdaten optimiert ist. Es sind Werkzeuge die über ihre graphische Oberfläche einen guten Einstieg ermöglichen, sogenannte *low-code* Tools, die sich gut eignen für die Entwicklung von Prototypen mit geringem zeitlichen Aufwand. Je nach Anwendung können die einzelnen Prozessschritte auch gut in einer Scriptssprache wie Python durchgeführt oder eine andere Datenbank verwendet werden.



**Abb. 10.1.:** Typischer IoT Datenfluss und Verarbeitung über diverse Instanzen, von dem IoT Gerät mit Sensorik, Datenkommunikation mit MQTT und dem MQTT Broker, zur Datenprozessierung mit Node-Red, Datenspeicherung und Visualisierung.

Ziel dieser Übung ist es MQTT näher kennenzulernen und die MQTT Kommunikation mit dem Raspberry Pi zu testen. MQTT ist ein leichtgewichtiges Kommunikationsprotokoll, welches das *Publish-Subscribe* Muster verwendet und sich gut für die Anwendung in IoT Projekten geeignet.

**Unterlagen:** [E07\\_MQTT.zip](#)

## 10.2. Übungsaufbau

- Schliesse den Raspberry Pi an Monitor, Keyboard und Maus an oder verbinde Dich mit diesem über SSH (und SFTP).
- Erstelle auf dem Raspberry Pi im `Documents` Ordner einen neuen Ordner `mqtt`, in welchem Du Änderungen und neue Dateien für diese Übung speichern kannst.
- Schliesse den Sensor **BME688** an den Raspberry Pi über die Breakout Garden **I2C** Schnittstelle korrekt an (siehe [E01 Luftqualität](#)), so dass die Beschriftung der Anschlüsse am Sensor und bei der Schnittstelle übereinstimmen.
- Kontrolliere mit dem Befehl `i2cdetect -y 1` ob der Raspberry Pi mit dem Sensor verbunden ist. Der Sensor sollte auf der Adresse `0x76` erkannt werden.
- Kontrolliere, ob die Library `bme680` installiert ist mit `python -c "import bme680"`. Installiere die Library mit in der aktivierte virtuellen Environment `source ~/.env/bin/activate` mit `pip install bme680`, falls sie nicht installiert ist.
- Wechsle in den Ordner `Documents` und erstelle einen Ordner `mqtt` für diese Übung.

## 10.3. Aufgabe 1: MQTT kennenlernen

[Mosquitto](#) ist eine kompakter open-source Message Broker der Eclipse Foundation, welcher das MQTT Protokoll implementiert und auf Produktionsserver, wie auch auf stromsparenden Geräten wie dem Raspberry Pi eingesetzt werden kann. Wir testen die Kommunikation mit dem MQTT Broker und den Clients auf dem Raspberry Pi. Der Mosquitto Broker ist auf dem Raspberry Pi Image vorkonfiguriert und läuft als Service im Hintergrund. Die Clients `mosquitto_pub` und `mosquitto_sub` sind ebenfalls installiert und können für das Testen der Kommunikation MQTT verwendet werden. Mosquitto Dokumentation: <https://mosquitto.org/documentation>



**Abb. 10.2.:** MQTT Device publiziert an den MQTT Broker über ein definiertes Topic Messdaten. Der Broker publiziert die Daten über die definierten Topics. Diese werden vom Subscriber abonniert und empfangen mit dem Vorteil, dass keines der Geräte zur gleichen Zeit synchron senden und empfangen muss.

Erstelle einen Subscriber der für das Topic `iot/temperature` eine Subscription erstellt.

```
mosquitto_sub -h 127.0.0.1 -v -t 'iot/temperature'
```

Öffne ein zweite Shell und erstelle einen Publisher für dasselbe Topic

```
mosquitto_pub -h 127.0.0.1 -t 'iot/temperature' -m 'Aussentemperatur: 22°
↪ Celsius'
```

**MQTT Message** Die MQTT Nachrichten (Messages) bestehen aus einem *Topic* (Thema) und einer *Payload* dem Nachrichteninhalt.

**MQTT Topics** MQTT verwendet Themen (Topic) und hierarchische Themenebenden getrennt über einem Schrägstrich / ähnlich einem Ordnersystem auf einem Computer. Dies ermöglicht eine Strukturierung und auch Filterung der Messages. Topics sind case-sensitive, sollten nicht mit einem /, \$ beginnen und keine Leerzeichen enthalten, sowie kurz und aussagekräftig sein. Filtern von Topics ist mit dem Wildcard + für ein einzelnes Thema und # für alle Themen möglich<sup>1</sup>.

**MQTT Payload** Die Nachrichten (Messages) die über MQTT übertragen werden, werden als Payload bezeichnet. Diese Payload ist nicht an eine bestimmte Struktur gebunden und kann ein Text- oder Binärformat sein. Eine festgelegte Struktur für die Payload ist jedoch sinnvoll für eine reibungslose Datenverarbeitung beispielsweise im JSON Format. Übliche Fromate sind Text, JSON, Binärdaten, Hex Strings oder auch Protocol Buffer.

Falls der Mosquitto Brokers und Clients nicht installiert sind, können diese mit folgenden Befehlen installiert werden.

```
sudo apt install mosquitto -y
sudo apt install mosquitto-clients -y
```

<sup>1</sup>Topics, die mit \$ beginnen sind reserviert für interne Informationen des Brokers und Clientstatistiken, oft in der Form \$SYS/. Beispielsweise zeigt \$SYS/broker/clients/connected die Anzahl der verbundenen Clients. Mit dem Wildcard # Filter könnt Ihr mit \$SYS/# alle Systeminformationen der Brokers anzeigen lassen.

### Übung 10.1.

- Teste nun den MQTT mit unterschiedlichen Topics und Nachrichten.
- Teste einen weiteren Mosquitto Server auf einem anderen Raspberry Pi und teste die Kommunikation. Hierbei muss die IP Adresse entsprechend angepasst werden.
- Einigt Euch auf einen MQTT Broker und eine Topic-Struktur beispielweise `iot/temperature` und `iot/humidity` und publiziert und abonniert Nachrichten. Was fällt Euch auf, wenn Ihr die Nachrichten empfängt?
- Welche Angaben sollten zusätzlich zu den Messwerten in der Payload der Nachrichten enthalten sein?

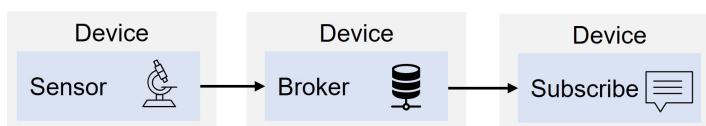
```
sudo systemctl enable mosquitto.service
sudo systemctl status mosquitto
```

#### Hinweis

MQTTBox und [MQTT Explorer](#) sind zwei MQTT Clients, die für die Entwicklung und das Testen von MQTT Applikationen verwendet werden können. Jedoch scheinen diese nicht mehr oder eher sporadisch weiterentwickelt zu werden.

## 10.4. Aufgabe 2: MQTT mit Python verwenden

Nun verwenden wir die Bibliothek `paho-mqtt` in Python um MQTT Messages zu senden oder empfangen. Folgende zwei Code Snippets zeigen wie ein Publisher und Subscriber in Python minimal implementiert werden können.



**Abb. 10.3.:** Device sendet Sensordaten und Topic an den MQTT Broker, ein weiteres Gerät abonniert (subscribe) dieses Topic und kann so die gesendeten Daten an das entsprechende Topic empfangen. In dieser Übung über ein Python Script für *Publish* und *Subscribe*

**Script 1:** `mqtt_sub.py` - Subscriber

```

import paho.mqtt.client as mqtt
ip = "127.0.0.1"                                     ①
topic = "iot/temperature"                                ②

# Callback Funktion für den Verbindungsauftbau
def on_connect(client, userdata, flags, rc):           ③
    print("Connected - code: "+str(rc))
    client.subscribe(topic)                             ④

# Callback Funktion für eingehende Nachrichten
def on_message(client, userdata, msg):                  ⑤
    print(msg.topic+" "+str(msg.payload))

# Erstellen des MQTT Clients
client = mqtt.Client()                                 ⑥
client.on_connect = on_connect
client.on_message = on_message
client.connect(ip, 1883, 60)                           ⑦
client.loop_forever()

```

- ① IP Adresse des MQTT Brokers
- ② Topic auf welches der Subscriber hört
- ③ Callback Funktion on\_connect wird ausgeführt, wenn die Verbindung steht
- ④ Subscription für das Topic
- ⑤ Callback Funktion on\_publish wird ausgeführt, wenn eine Nachricht empfangen (publish) wird
- ⑥ Erstellen eines MQTT Clients und Zuweisung der Callback Funktionen
- ⑦ Verbindung zum MQTT Broker herstellen und auf eingehende Nachrichten warten (loop\_forever)

### Script 2: mqtt\_pub.py - Publisher

```

import paho.mqtt.publish as publish
ip = "127.0.0.1"                                     ①
topic = "iot/temperature"
publish.single(topic, "22.0", hostname=ip, port=1883)   ②

```

- ① IP Adresse des MQTT Brokers und Topic definieren
- ② Nachricht mit Topic an den MQTT Broker (ip,port) senden

Falls die Library paho-mqtt nicht installiert ist, kann diese in der aktivierte virtuellen Environment source ~/env/bin/activate mit pip install paho-mqtt.

**Übung 10.2.**

- Speichere die beiden Dateien `mqtt_pub.py` und `mqtt_sub.py` im Ordner `mqtt` ab.
- (Optional: Passe die IP Adresse des MQTT Brokers an, falls ein anderer MQTT Broker genutzt werden soll.)
- Öffne zwei Terminals und führe diese mit `python mqtt_pub.py` und `python mqtt_sub.py` aus.

## 10.5. Aufgabe 3: Sensordaten mit MQTT übertragen

Passe nun das Script `mqtt_pub.py` an, so dass die Temperatur vom Sensor `BME688` ausgelesen und über MQTT übertragen wird. Die Temperatur soll alle 10 Sekunden übertragen werden. Nutze hierfür das Script zum Auslesen der Sensordaten aus der Übung [E01 Luftqualität](#) und passe dieses an.

**Übung 10.3.**

- Speichere das `mqtt_pub.py` als `mqtt_pub_bme688.py` im Ordner `mqtt` ab.
- Ergänze das Script um die Funktionen zum auslesen der Temperatur des BME688 Sensors.
- Ergänze das Script für das Auslesen der Luftfeuchtigkeit und Luftdruck und ergänze die topics mit `iot/humidity` und `iot/pressure`.
- Erweitere das Script `mqtt_sub.py` um die Subscription für die Topics `iot/humidity` und `iot/pressure` und speichere es als `mqtt_sub_bme688.py` ab.
- Öffne zwei Terminals und führe diese mit `python mqtt_pub_bme688.py` und `python mqtt_sub_bme688.py` aus.

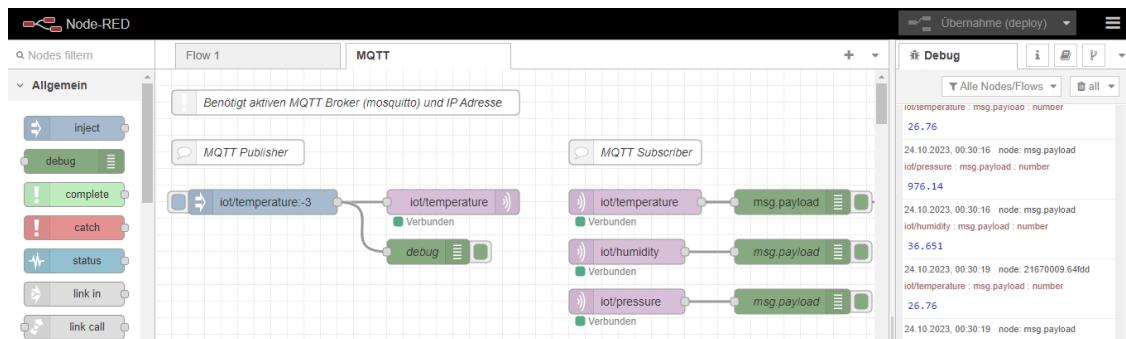
## 10.6. Aufgabe 4: MQTT mit Node-RED verwenden

Node-Red ist ein grafisches Entwicklungswerkzeug für IoT, ein low-code Tool für *event-driven applications*. Es bietet eine browserbasierte und datenstromorientierte “Flow” Programmierung für die Verarbeitung von Sensordaten (ähnlich wie FME oder graphische Modellierungswerzeuge in GIS). Die Implementation von Node-Red ist in JavaScript und basiert auf node.js. Datenverarbeitungsflows können gespeichert und wiederverwendet werden. Im Arbeitsbereich (Abb. 10.4) können sogenannte Nodes, die unterschiedliche Funktionen erfüllen, zu einem Daten-“Flow” werden. Nodes können auch selbst erstellt werden. Es gibt eine Vielzahl von Nodes, die von der Community entwickelt wurden, die

über den Node-Red Palette Manager installiert werden können. Core Nodes von Node-Red sind:

- **Inject Node:** Kann einen Flow über den Button direkt auslösen oder in regelmässigen Abständen mit Zeitstempel oder vordefinierten Nachrichten (msg) senden.
- **Debug Node:** Kann die [Nachrichten](#) (msg) in Debug Sidebar anzeigen lassen.
- **Function Node:** Kann mit JavaScript [Funktionen](#) den Inhalt der Nachrichten (msg) verändern.
- **Change Node:** Ermöglicht das Ändern von Eigenschaften einer Nachricht (ohne den Funktion Node zu nützen) um beispielsweise Eigenschaften zu setzen, ändern oder löschen.
- **Switch Node:** Kann Nachrichten (msg) anhand von Regeln auswerten in verschiedene Ausgänge leiten (wie ein Switch Case in der Programmierung).
- **Template Node:** Kann über Eigenschaften einer Nachricht und einer Vorlage (Template) neue Nachricht nach Vorlage erstellen: Nachricht: `{ {payload} }!` wird Nachricht: `1570439577309 !`.

Node-Red kann über den Browser auf dem Raspberry Pi oder via Laptop gestartet werden. Öffne den Browser und gebe die IP Adresse des Raspberry Pi mit dem Port 1880 ein: <http://<ip-adresse>:1880>. Führt nun das Kurztutorial zu Node-Red aus: [Node-Red Getting Started: First Flow](#). In diesem Tutorial wird ein Flow erstellt, der eine Nachricht mit einem Zeitstempel ausgibt.

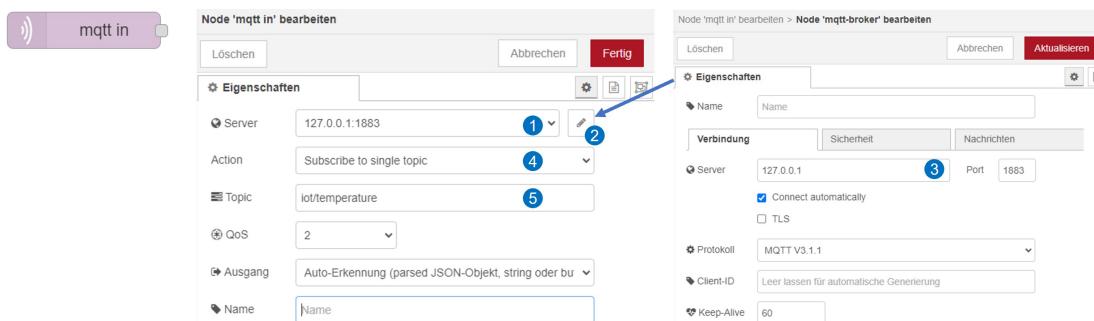


**Abb. 10.4.:** Node-Red Flow der Daten in das Topic `iot/temperature` published, und Nachrichten aus dem Topic `iot/temperature` subscribed.

### Übung 10.4.

Erstelle einen Flow der Nachrichten an den MQTT Broker senden und Nachrichten vom MQTT Broker empfangen kann.

1. Füge hierfür einen Inject Node, einen Debug Node und einen MQTT Output Node hinzu. Öffne die Einstellungen des MQTT Output Nodes (siehe Abb. 10.5) und setze die IP Adresse des MQTT Brokers und das Topic auf `iot/temperature`. Starte den Flow und überprüfe, ob die Nachrichten an den MQTT Broker gesendet werden.
2. Erstelle nun einen weiteren Flow, der die Nachrichten vom MQTT Broker empfängt und in der Debug Sidebar anzeigen. Füge hierfür einen MQTT Input Node (siehe Abb. 10.5) und einen Debug Node hinzu. Öffne die Einstellungen des MQTT Input Nodes und setze die IP Adresse des MQTT Brokers und das Topic auf `iot/temperature`. Starte den Flow und überprüfe, ob die Nachrichten vom MQTT Broker empfangen werden.
3. Ergänze den Flow um einen MQTT Input Node für die Topics `iot/humidity` und `iot/pressure` und je einen Debug Node. Starte den Flow und überprüfe, ob die Nachrichten vom MQTT Broker empfangen werden.



**Abb. 10.5.:** Node-Red MQTT Node Einstellungen setzen, (1) Name angeben, (2) Server Einstellungen öffnen und (3) IP Adresse des MQTT Brokers setzen, (4) subscribe to single topic wählen und (5) Topic `iot/temperature` setzen.

#### Hinweis

Flows stoppen: Um einen Flow in der Ausführung zu stoppen, kann dieser über den Tab Flow (rechte Maustaste) deaktiviert werden (disable flow) und mit einem erneuten Übernehmen (Deploy) wird der Flow gestoppt.

## 10.7. Aufgabe 5: Node-Red MQTT mit InfluxDB verwenden

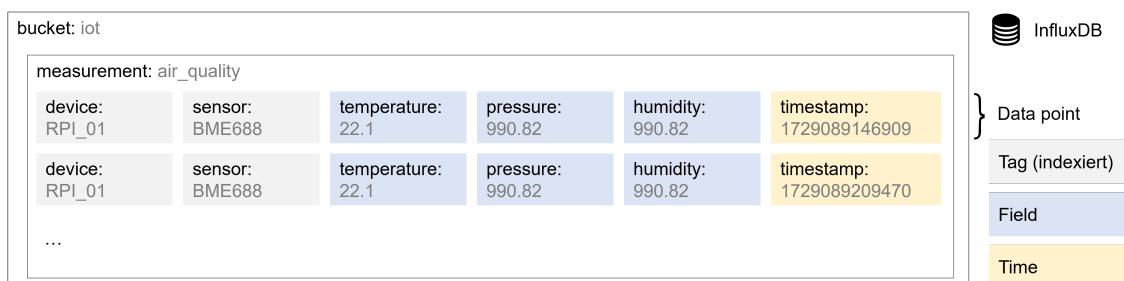
InfluxDB ist eine Datenbank, die für die Erfassung, Speicherung, Verarbeitung und Visualisierung von Zeitreihendaten entwickelt wurde. Zeitreihendaten sind Datenpunkte, die in zeitlicher Sequenz erfasst wurden und bestehen in der Regel aus aufeinanderfolgenden Messungen aus derselben Quelle, wie beispielsweise die Temperaturdaten des BME688.



**Abb. 10.6.:** IoT Datenfluss mit Node-Red MQTT und InfluxDB. Python Script publiziert (publish) die Sensormesswerte und das Topic an den MQTT Broker, NodeRed abonniert (subscribe) ein oder mehrere Topics und speichert die Sensormesswerte in einer InfluxDB Datenbank.

InfluxDB organisiert die Zeitreihendaten in *Buckets* (anstatt Datenbanken) und *Messungen* (Measurements). Ein Bucket kann mehrere Messungen enthalten, wobei Messungen Felder und Tags enthalten können ([influxdata, 2023](#)). Eine Messung enthält *Felder* mit key-value Paaren von Messwerten, die sich über die Zeit ändern. *Tags* sind key-value Paare, die sich nicht über die Zeit ändern und für die Filterung und Gruppierung verwendet werden können.

Das Tutorial [Getting Started](#) zeigt die ersten Schritte mit InfluxDB<sup>2</sup>.



**Abb. 10.7.:** Datenorganisation in der InfluxDB, über Buckets und Measurements, mit Tags für Metadaten, Feldern für die Speicherung der Messwerte und dem Zeitstempel.

<sup>2</sup>Die Dokumentation über die [InfluxDB key concepts](#) und das [InfluxDB Schema Design](#) führen übersichtlich die Konzepte Best Practices des Schema Designs und effiziente Abfragen ein.

## Glossary

**Bucket** Ein Bucket ist ein benannter Container zur Speicherung der Zeitreihendaten. Buckets können mehrere Messungen enthalten und sind die oberste Ebene der Organisation in InfluxDB.

**Measurement** Eine Messung ist eine logische Gruppierung von Zeitreihendaten, mehreren Einzelmessungen (Data Point, points). Alle Messungen sollten dieselben Tags enthalten. Messungen können mehrere Tags und Felder enthalten

**Data Point** Ein Datenpunkt ist eine Einzelmessung, die zu einem bestimmten Zeitpunkt erfasst wurde. Ein Datenpunkt besteht aus einem *mesurement*, *field set*, *tag set* und einem *timestamp*.

**Tags** Tags sind key-value Paare, die sich nicht (oder selten) über die Zeit ändern und

für die Filterung und Gruppierung verwendet werden können. Tags können für die Organisation und Strukturierung der Daten verwendet werden und werden indiziert. Die Indexierung ermöglicht eine effiziente Abfrage und Aggregation der Daten.

**Fields** Felder sind key-value Paare von Messwerten, die sich über die Zeit ändern, wie Temperatur, Luftdruck etc. Felder können für die Speicherung von Messwerten verwendet werden. Felder werden nicht indiziert.

**Timestamp** Der Zeitstempel ist der Zeitpunkt, zu dem die Messung erfasst wurde. Der Zeitstempel wird in InfluxDB automatisch hinzugefügt, wenn er nicht explizit in der Messung definiert ist.

Die graphische Oberfläche von InfluxDB kann über den Browser auf dem Raspberry Pi oder via Laptop gestartet werden. Öffnet nun den Browser und gebe die IP Adresse des Raspberry Pi mit dem Port 8086 ein: <http://<ip-adresse>:8086>.

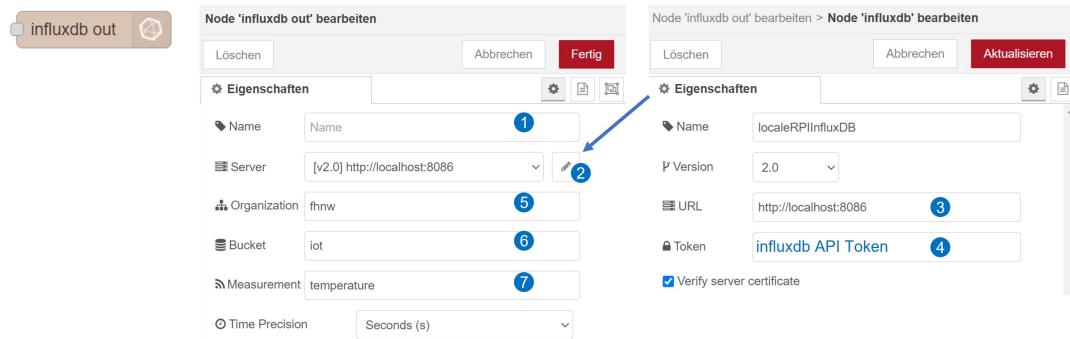
Erstellt über die linke Menuleiste unter *Load Data* einen neuen Bucket mit dem Namen *iot*, falls dieser nicht schon existiert. Kopiert unter *load Data / API Tokens* den Token für den Zugang zur Datenbank.

Diese Angaben (*bucket* und *API Token*) werden für den InfluxDB Node in Node-Red benötigt. Die Einstellungen des InfluxDB Node benötigen die Angaben, wohin die Daten in der Datenbank gespeichert werden (links in Abb. 10.8) mit Angaben zum Bucket (Datenbankname), Organisation<sup>3</sup> und measurement und die Serververbindung (rechts in Abb. 10.8) mit der IP Adresse und das Token für den Zugang zur Datenbank.

### Hinweis

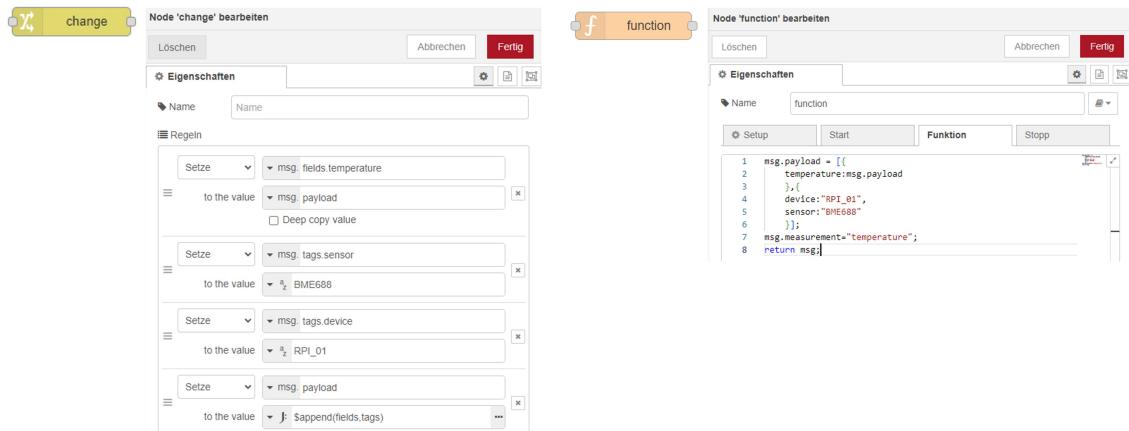
Falls keine Nodes für InfluxDB in Node-Red vorhanden sind, kann die Erweiterung `node-red-contrib-influxdb` über das Hauptmenü *Palette verwalten* im Tab *Installation* installiert werden.

<sup>3</sup>wird bei der Erstellung der Datenbank definiert



**Abb. 10.8.:** Einstellungen der InfluxDB Nodes in Node-Red, links: Einstellungen zur Datenbank (1) mit Angaben zur Organisation (5), Bucket (6) und Namen der Messung *measurement* (7), rechts: Einstellungen zur Datenbankverbindung (2) mit der URL und Port der InfluxDB (3) und dem API Token (4), für den Zugang zur Datenbank.

Die Messwerte können direkt in die InfluxDB geschrieben werden, jedoch fehlen da noch die Tags, die für die Filterung und Gruppierung verwendet werden können. Diese können über den Change Node hinzugefügt werden, wie in Abb. 10.9 gezeigt. Hierbei wird der Wert der Payload in ein Feld mit dem Namen *temperature* geschrieben und die Tags *device* und *sensor* werden hinzugefügt. Dieselbe Struktur kann alternativ auch mit dem Function Node erstellt werden, siehe Abb. 10.9 rechts. Hierbei wird der Wert der Payload in ein Feld mit dem Namen *temperature* geschrieben und die Tags *device* und *sensor* werden hinzugefügt. Die Bezeichnung der Messung *temperature* wird im InfluxDB Node definiert oder kann im Function Node überschrieben werden.



**Abb. 10.9.:** Über den Change Node, wie auch den Function Node kann mit JavaScript der Inhalt der Payload verändert werden und diesen für den Import in die InfluxDB angepasst werden.

```

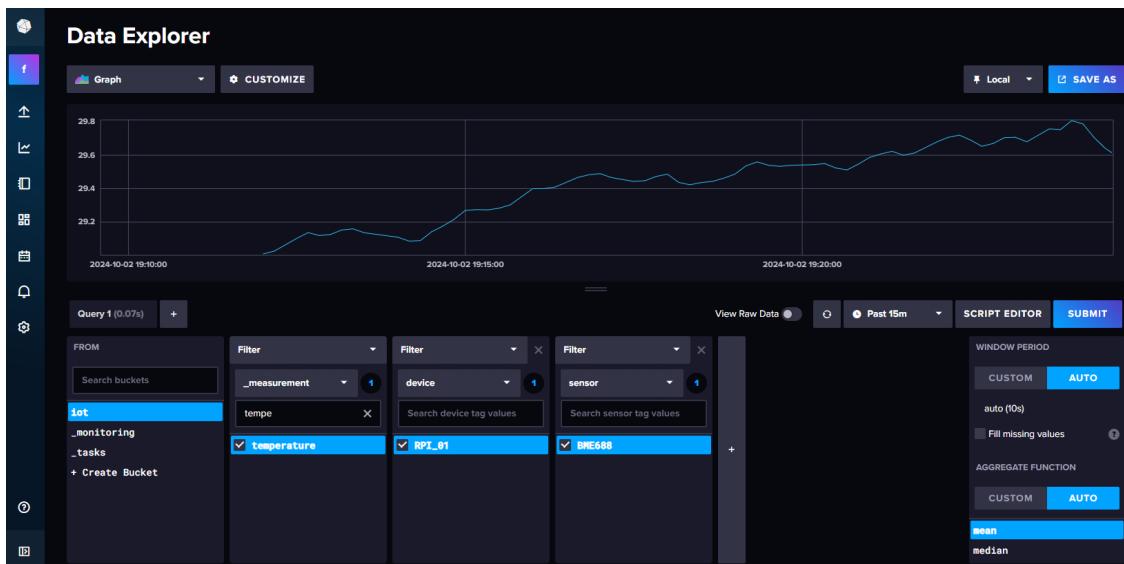
msg.payload = [
  {
    temperature:msg.payload
  }
];
  
```

(1)

```
}, { device:"RPI_01", sensor:"BME688" }];
msg.measurement="temperature";
return msg;
```

- ① Werte der Payload in Payloadstruktur für die InfluxDB schreiben
- ② Tags der Messung zuweisen
- ③ Optional *measurement* kann im InfluxDB Node oder im Funktionsknoten definiert werden

InfluxDB bietet über das Menu *Data Explorer* eine einfache Möglichkeit die Daten zu visualisieren. Der Query Builder (Abb. 10.10) hilft bei der Erstellung von Abfragen mit der über die Fields und Tags gefiltert werden kann, die dann über den Button *Submit* ausgeführt und dargestellt werden können. Die erstellte Query kann über den *Script Editor* (Abb. 10.10) angezeigt werden.



**Abb. 10.10.:** Über den *Data Explorer* können in InfluxDB einfach Abfragen zusammengestellt und über *submit* visualisiert werden

## 10.8. Aufgabe 6: InfluxDB mit Grafana verwenden

Grafana ist eine Open-Source Anwendung für die grafische Darstellung von Daten aus den verschiedenen Datenquellen, wie Postgres, SQLite oder InfluxDB. Über Grafana können interaktive Dashboards mit vielen Visualisierungsansätzen erstellt werden. In dieser Übung wird Grafana mit der InfluxDB Datenbank verwendet.

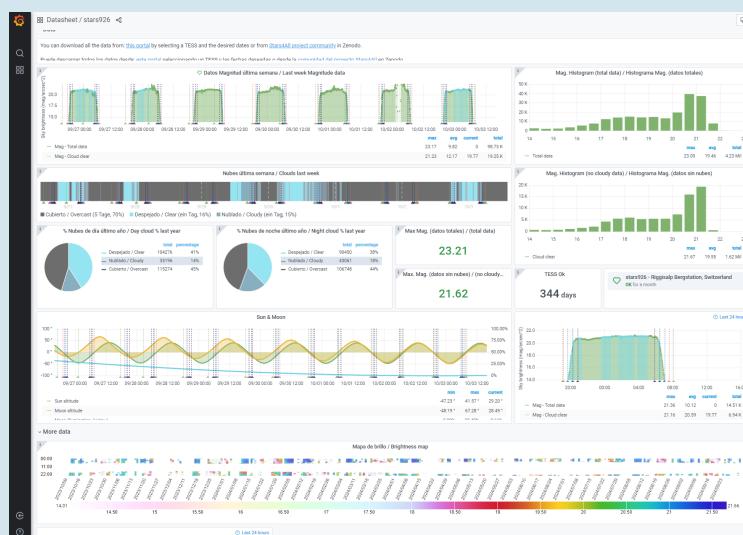
Öffnet nun den Browser und gebt die IP Adresse des Raspberry Pi mit dem Port 3000 ein: <http://<ip-adresse>:3000>. Unter Connections / Data Sources können Grafana

### Übung 10.5.

- Erstellt nun einen Flow der die Nachrichten des Topics `iot/temperature` vom MQTT Broker empfängt und in die InfluxDB schreibt.
- Erstellt erst einen flow der nur die Nachrichten ohne Tags in die InfluxDB schreibt
- Ergänzt den Flow mit einem Change oder Function Node um die Tags `device` und `sensor` hinzuzufügen.
- Visualisiert den Temperaturverlauf des BME688 in der InfluxDB mit dem *Data Explorer* und studiert die Flux Abfragesprache.
- Erstellt einen Flow der die Nachrichten der Topics `iot/humidity` und `iot/pressure` vom MQTT Broker empfängt und in die InfluxDB schreibt.
- (Optional) erstellt im Menu *Dashboard* eine Visualisierung der Messwerte.

### Hinweis

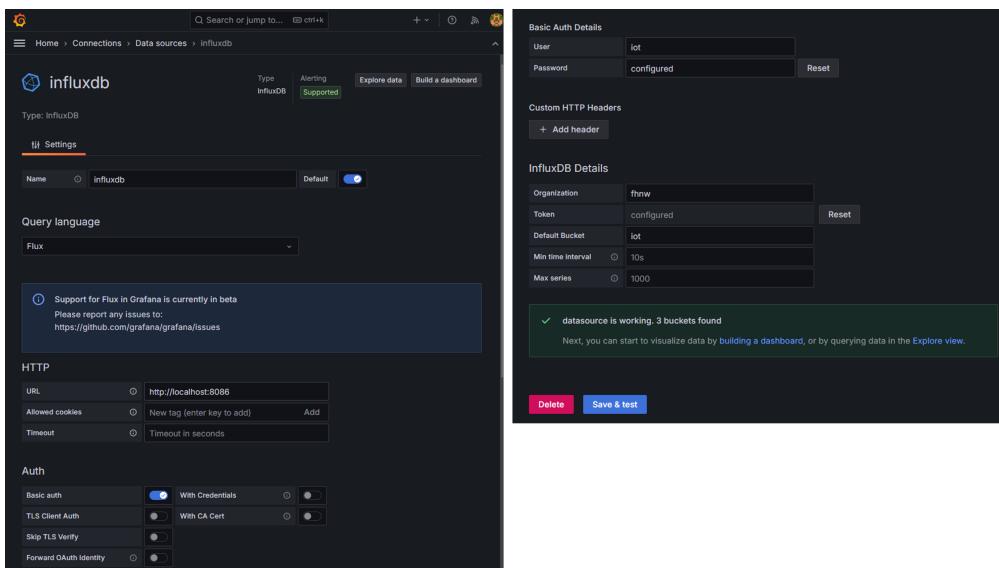
**Monitoring der Lichtverschmutzung am Nachthimmel** TESS (Telescope Encoder and Sky Sensor) ist ein Photometer ([Zamorano et al., 2016](#)) für das Stars4All Citizen Science Projekt mit dem Ziel die Lichtverschmutzung weltweit in der Nacht zu vermessen, welche nicht nur für die Umwelt problematisch ist, sondern auch für astronomische Beobachtungen. Die TESS Geräte senden von verschiedenen Standorten weltweit ihre Messungen welche über der folgende Dashboard mit Grafana der Öffentlichkeit zur Verfügung gestellt werden. Dashboard: <https://tess.dashboards.stars4all.eu/>, Messungen der Lichtverschmutzung am Beispiel des Naturparks Gantrisch in der Schweiz. [https://tess.dashboards.stars4all.eu/d/datasheet\\_stars926/stars926?orgId=1](https://tess.dashboards.stars4all.eu/d/datasheet_stars926/stars926?orgId=1)



**Abb. 10.11.:** Grafana Dashboard der Messdaten des TESS Photometes im Naturpark Gantrisch dem ersten Dark Sky Park der Schweiz

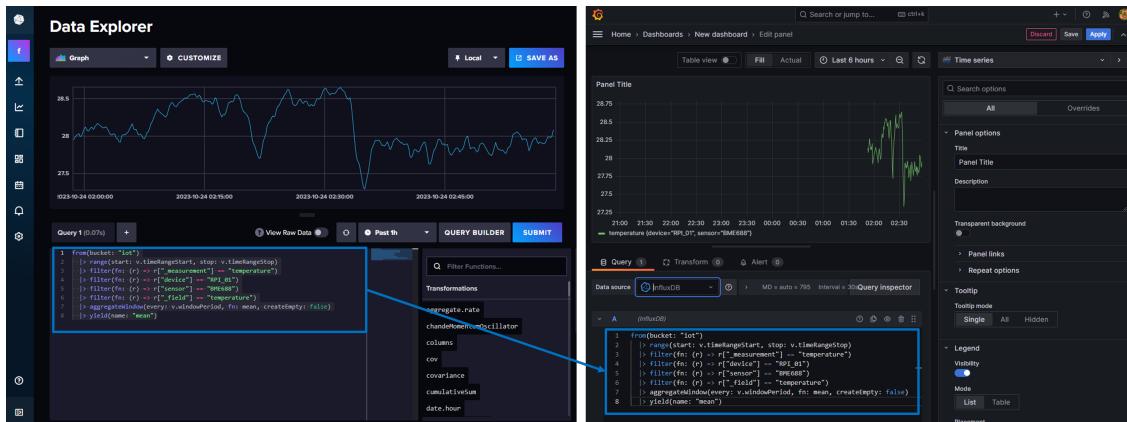
Datenquellen hinzugefügt werden, wählt nun *InfluxDB* um Grafana mit der InfluxDB zu verbinden. Setzt folgende Einstellungen (Abb. 10.12) und speichert diese:

- *Query Language*: Wählt in den Einstellungen zu *Query Language* die Abfragesprache Flux.
- *HTTP*: Setzt die IP Adresse des Raspberry Pi und den Port 8086: `http://localhost:8086`.
- *Auth*: aktiviert *Basic Auth*
- *Basic Auth Details*: Setzt den InfluxDB Benutzer und das Passwort für die InfluxDB
- *InfluxDB Details*: Setzt die InfluxDB Organisation *fhnw*, den *API Token*, und den InfluxDB Bucket *iot*.



**Abb. 10.12.:** InfluxDB als Datenquelle in Grafana hinzufügen mit den entsprechenden Angaben.

Ist die Verbindung erstellt, können neue Dashboards erstellt werden. Die Queries, die für die Visualisierung verwendet werden sollen, können über den *Query Builder* in InfluxDB erstellt und in Grafana beim Erstellen der Panels (Abb. 10.13) reinkopiert werden.



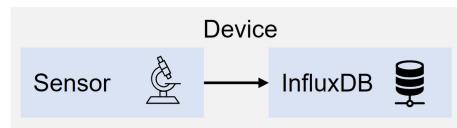
**Abb. 10.13.:** Flux Queries der Datenabfragen im Data Explorer können in Grafana für die Visualisierung kopiert und genutzt werden.

### Übung 10.6.

- Erstellt eine Verbindung zwischen Grafana und der InfluxDB.
- Erstellt ein Dashboard mit einer Visualisierung der Temperatur, Luftfeuchtigkeit und Luftdruck des BME688 Sensors.

## 10.9. Aufgabe 7: InfluxDB mit Python verwenden (optional)

Anstatt mit Node-Red können die Daten des Sensors direkt in die InfluxDB geschrieben werden. Folgender Code zeigt wie die Daten des BME688 Sensors mit Python ausgelesen und in die InfluxDB geschrieben werden können. Teste erst, ob der `influxdb-client` installiert ist und installiere diesen falls nicht mit dem Befehl `pip3 install influxdb-client`. Folgendes Tutorial zeigt wie die Daten mit Python in die InfluxDB geschrieben werden können: [Getting Started with Python and InfluxDB v2.0](#).



**Abb. 10.14.:** Direktes Speichern der Sensormessdaten in der Influxdb ohne MQTT Protokoll. Hierbei gilt zu beachten, dass beide Geräte eine aktive Verbindung haben.

```

from datetime import datetime
import time

from influxdb_client import InfluxDBClient, Point, WritePrecision
  
```

```
from influxdb_client.client.write_api import SYNCHRONOUS

# Generieren ein Token in der InfluxDB UI unter dem Tab "Data / Tokens Tab"
token = "<influxdb-token>"
org = "fhnw"
bucket = "iot"

client = InfluxDBClient(url="http://localhost:8086", token= token, org= org)
write_api = client.write_api(write_options = SYNCHRONOUS)

temperature = 22.0
data =
    Point("measures").tag("sensor","BME688").tag("device","RPI_01").field("temperature",
    temperature)
write_api.write(bucket = bucket, record = data)
```

**Übung 10.7.** Schreibe nun mit Hilfe des Tutorials und dem Beispielcode ein Python Script, welches die Temperatur, Luftfeuchtigkeit und Luftdruck des BME688 Sensors ausliest und in die InfluxDB schreibt.

## Anhang A

# Raspberry Pi

Open Source Einplatinenrechner werden in Bildung und Forschung eingesetzt und haben eine breite Anwendung in weiteren Bereichen wie IoT. SBCs haben grosse Bedeutung in der Ingenieur- und Informatikausbildung, da sie sich gut für Hands-on Vermittlung komplexer Themen eignen, und verbessern technische Fähigkeiten, stärken das Interesse und Motivation ([Ariza & Baez, 2022](#)).

**System on Chip SOC** integriert die meisten Funktionen und Komponenten auf einem Chip (CPU, RAM, GPU, Schnittstellen) ursprünglich für eingebettete (embedded) Systeme, heute Mobile Geräte (Mobiles, Tablets), Edge Geräte (IoT) etc. → Größenreduktion, verbesserte Leistung, tiefer Stromverbrauch, kosteneffizient, jedoch lange Entwicklungszeiten und keine Modularität.

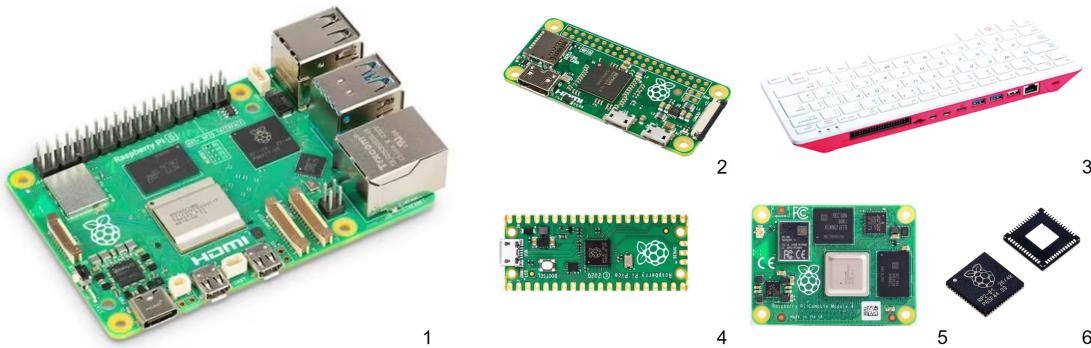
**Single Board Computer SBC** ist ein Computer auf einer Leiterplatine mit allen erforderlichen Komponenten und Anschlüssen wie CPU, RAM, GPU I/O Schnittstellen etc. → Einfach zu nutzen, tiefer Stromverbrauch, kosteneffizient, getestete Hardware, tendenziell günstig, jedoch schwierig Anpassungen vorzunehmen, weniger Möglichkeiten als «Multi Board Computer».

Der [Raspberry Pi](#) ist ein preisgünstiger Einplatinencomputer (Single Board Computer SBC) in Kreditkartenformat zum Experimentieren und Erlernen des Programmierens. Die Motivation für die Entwicklung des Raspberry Pi war die sinkende Anzahl Informatikstudierender an der Universität Cambridge und der Rückgang der Informatikkenntnisse der Studierenden. Der Raspberry Pi wird von der britischen [Raspberry Pi Foundation](#) entwickelt seit 2009 entwickelt.

Url: <https://www.raspberrypi.com>

Raspberry Pi Modelle im Überblick:

- Raspberry Pi 5 (mit 4/8GB RAM ab November 2023)
- Raspberry Pi 4 Model B (mit 1/2/4/8GB RAM)
- Raspberry Pi 3 Model B+
- Raspberry Pi 2 Zero günstige und kompakte Version (mit/ohne Bluetooth und WiFi)
- Raspberry Pi 400 Unit (Tastatur mit integriertem Raspberry Pi)
- Raspberry Pi Pico Series flexible und kompakte Microcontroller mit dem RP2040 Chip
- Raspberry Pi Compute Module 4 (CM4 mit/ohne eMMC)



**Abb. A.1.:** Raspberry Pi Modelle: 1. Raspberry Pi 5, 2. Raspberry Pi 2 Zero, 3. Raspberry Pi 400, 4. Raspberry Pi Pico, 5. Raspberry Pi Compute Module 4, 6. RP2040 Chip

Die Raspberry Pi verfügen über General Purpose Input Output GPIO sogenannte Pins, die für die Verbindung mit Sensoren und Aktoren verwendet werden können. Die Stiftleiste mit den GPIO Pins sind in unterschiedliche Gruppen unterteilt, die unterschiedliche Funktionen haben. Die GPIO Pins können über eine Programmiersprache wie Python oder C angesprochen werden und ermöglichen die Interaktion mit Sensoren oder Akto- ren und weiteren Komponenten.

**Raspberry Pi Pinout**

**I2C - Inter Integrated Circuit**

**GPIO 2 and GPIO 3 -** the Raspberry Pi's I2C1 pins - allow for two-wire communication with a variety of external sensors and devices.

The I2C pins include a fixed 1.8 kΩ pull-up resistor to 3.3V. They are not suitable for use as general purpose IO where a pull-up might interfere.

I2C is a multi-drop bus, multiple devices can be connected to these same two pins. Each device has its own unique I2C address.

You can verify the address of connected I2C peripherals with a simple one-liner:

```
1. sudo apt-get install i2c-tools
2. sudo i2cdetect -y 1
```

You can then access I2C from Python using the smbus library:

```
1. import smbus
2. DEVICE_BUS = 1
3. DEVICE_ADDR = 0x15
4. bus = smbus.SMBus(DEVICE_BUS)
5. bus.write_byte_data(DEVICE_ADDR, 0x00, 0x01)
```

**Legend**

- GPIO (General Purpose IO)
- SPI (Serial Peripheral Interface)
- I<sup>2</sup>C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- PCM (Pulse Code Modulation)
- Ground
- 5V (Power)
- 3.3V (Power)

**Details**

- [More Information](#)

**Revision:** a020d3  
**SoC:** BCM2837  
**RAM:** 1GB  
**Storage:** 16GB eMMC  
**USB ports:** 4 (of which 0 USB3)  
**Ethernet ports:** 1 (300Mbps max. speed)  
**Wi-Fi:** True  
**Bluetooth:** True  
**Network ports (CSI):** 1  
**Display ports (DSI):** 1

**J8:**

3V3	(1)	(2)	5V
GPIO2	(3)	(4)	5V
GPIO3	(5)	(6)	GND
GPIO4	(7)	(8)	GPIO4
GPIO5	(9)	(10)	0x1015
GPIO17	(11)	(12)	GPIO18
GPIO27	(13)	(14)	GND
GPIO22	(15)	(16)	GPIO23
GPIO10	(19)	(20)	0x1014
GPIO11	(21)	(22)	GPIO25
GPIO13	(23)	(24)	GPIO8
GPIO19	(25)	(26)	GPIO7
GPIO26	(27)	(28)	GPIO1
GPIO10	(29)	(30)	GND
GPIO13	(31)	(32)	GPIO12
GPIO11	(33)	(34)	GND
GPIO16	(35)	(36)	0x1016
GPIO26	(37)	(38)	GPIO20
	(39)	(40)	GPIO21

**P0E:**

TR01	(1)	(2)	TR00
TR03	(3)	(4)	TR02

For further information, please refer to <https://pinout.xyz/>

**Abb. A.2.:** links: Die Website <https://pinout.xyz> bietet eine interaktive Übersicht und Dokumentation der Pinouts der einzelnen Raspberry Pi Modellen, wie beispielsweise welche Pins die Schnittstelle I2C ansteuern. rechts: Der Befehl `pinout` liefert direkt über das Terminal das Pinout des Raspberry Pi

Raspberry Pi war von der weltweiten Chip- und Halbleiterknappheit betroffen und hatte seit 2020 tiefe Lagerbestände von elektronischen Komponenten. Die Nachfrage nach

Raspberry Pi ist seit 2020 stark gestiegen und die Produktionskapazitäten konnten nicht erhöht werden. Die Produktionsengpässe werden sich voraussichtlich ab Ende 2023 verbessern ([Upton, 2022](#)). Die Knappheit führte zu einem Preisanstieg von 50% bis 100% bei den Raspberry Pi Modellen und zu einer Verknappung der Raspberry Pi Modelle. Der Dienst [Raspberry Pi Stock Scraper](#) führt eine Übersicht der Verfügbarkeit der weltweiten Raspberry Pi Lagerbeständen von online Shops. Der [Pi-Shop](#) hat teilweise Kits und Boards in der Schweiz an Lager.

## A.1. Raspberry Pi Image schreiben

Der Raspberry Pi führt keinen internen Speicher mit sich, sondern benötigt eine SD-Karte. Auf dieser SD-Karte wird das Betriebssystem installiert, respektive als Image auf die MicroSD Karte geschrieben. Die SD-Karte muss mindestens 8 GB Speicherplatz haben idealerweise 16 GB oder mehr.

Software: [Raspberry Pi - Software](#)

Tutorial: [Raspberry Pi - How to set up Raspberry Pi](#)

**Raspberry Pi Imager** ist die von Raspberry Pi erstellte Software für das Schreiben von Images auf eine SD Karte. Der Imager stellt direkt auch weitere Images wie Ubuntu, Medienzentren, 3D Druck etc. zur Verfügung. Alternative *Image Burner* sind [Balena Etcher](#) und [Win32DiskImager](#).

Das Betriebssystem **Raspberry Pi OS** ist ein auf Debian basiertes Betriebssystem mit Anpassungen für die Hardware auf dem Raspberry Pi. Es gibt eine **Lite** Version ohne Desktop und eine **Desktop** Version mit Desktop. Die **Desktop** Version ist für den Einstieg empfehlenswert, da sie die grafische Oberfläche bietet und die Konsole. Die **Lite** Version ist für den produktiven Einsatz empfehlenswert, da sie weniger Ressourcen benötigt und somit mehr Ressourcen für die Anwendungen zur Verfügung stehen. Dazu muss der Raspberry Pi über das Netzwerk erreichbar sein, beispielsweise über SSH.

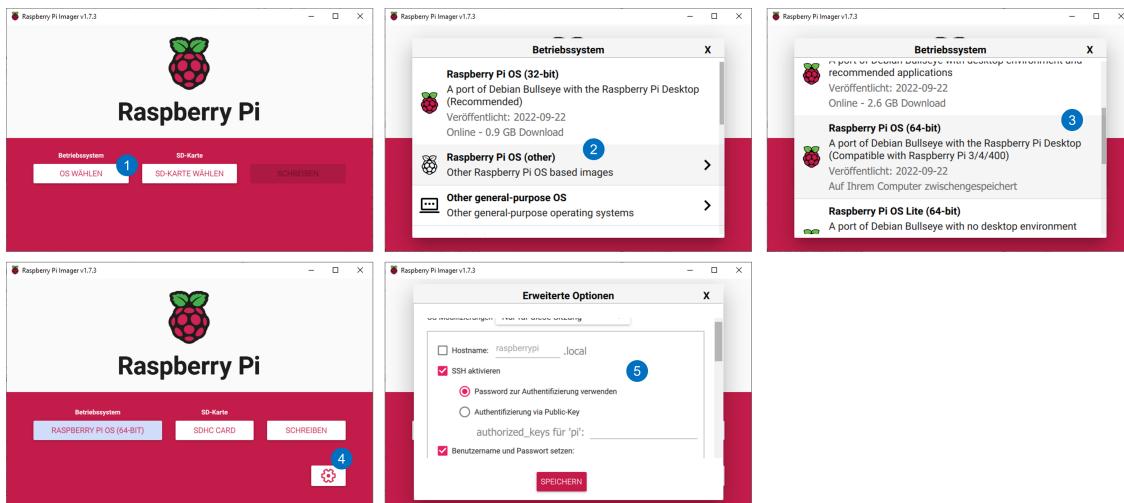
### Hinweis

Für die Nutzung am Campus nutzen wir die **Desktop** Version, da wir die grafische Oberfläche für das aktivieren des Internetzugangs benötigen.

Image schreiben mit **Raspberry Pi 64-bit** (Desktop Version) und mit den erweiterten Optionen Voreinstellungen zu **SSH** (Ja), **WiFi** (Heimnetzwerk), **Konto** und unter Sprache die **Zeitzone** Europe/Zurich und das das **Tastaturlayout** CH setzen.

### Hinweis

Die Einstellungen können später in der Konsole mit dem Programm `raspi-config` geändert werden.



**Abb. A.3.:** Raspberry Pi Imager ausführen mit (1) Betriebssystem wählen, (2) Raspberry Pi OS (Other), (3) Raspberry Pi OS 64-bit, (4) erweiterte Einstellungen wählen und unter diesen Voreinstellungen zum Konto, SSH, Zeitzone und Tastaturlayout vordefinieren.

### Hinweis

Für Testzwecke kann es sinnvoll sein den Standardkontonamen `pi` und das Standardpasswort `raspberry` zu verwenden. Jedoch sollte der Kontonamen und das Passwort für den produktiven Einsatz geändert werden, vor allem wenn der Raspberry Pi mit dem Internet verbunden ist, beispielsweise mit einem aktivierten SSH Zugang.

### Übung

#### Raspberry Pi Image auf SD-Karte schreiben

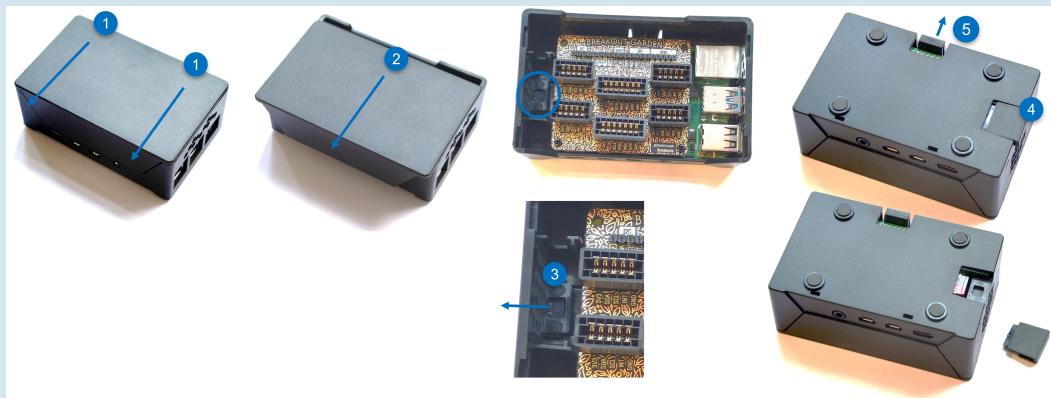
1. Installiere den Raspberry Pi Imager
2. Schreibe Raspberry Pi OS 64-bit auf die SD Karte
3. Aktiviere SSH und setze die Spracheinstellungen (WiFi Name und Password des Netzwerks von Zuhause oder dem Mobile Hotspot)
4. SD Karte in den Raspberry einsetzen
5. Maus, Tastatur und Monitor anschliessen und dann mit dem Netzteil mit Strom versehen

## A.2. Raspberry Pi einrichten

Für das Einrichten des Raspberry Pi wird die SD-Karte mit dem Image in den Raspberry Pi eingesetzt. Für das erstmalige Einrichten des Raspberry Pi OS ist ein Bildschirm, eine Tastatur und eine Maus erforderlich. Verbunden wird der Raspberry Pi über die HDMI-Schnittstelle mit dem Bildschirm und über die USB Schnittstelle mit der Tastatur und der Maus, dann wird die Stromversorgung über USB-C angeschlossen.

### Gehäuse öffnen und SD Karten Slot öffnen

Den Deckel durch leichtes Drücken mit zwei Finger bei (1) horizontal leicht in Pfeilrichtung drücken, nach einem Klick lässt sich das Gehäuse aufschieben. Der SD Karten-Slot lässt sich mit einem Stift in der Einbuchtung (3) mit leichtem Druck in Pfeilrichtung öffnen (4). Das Raspberry Pi Board kann durch ein leichtes Klicken in Pfeilrichtung (5) aus dem Gehäuse entfernt werden.



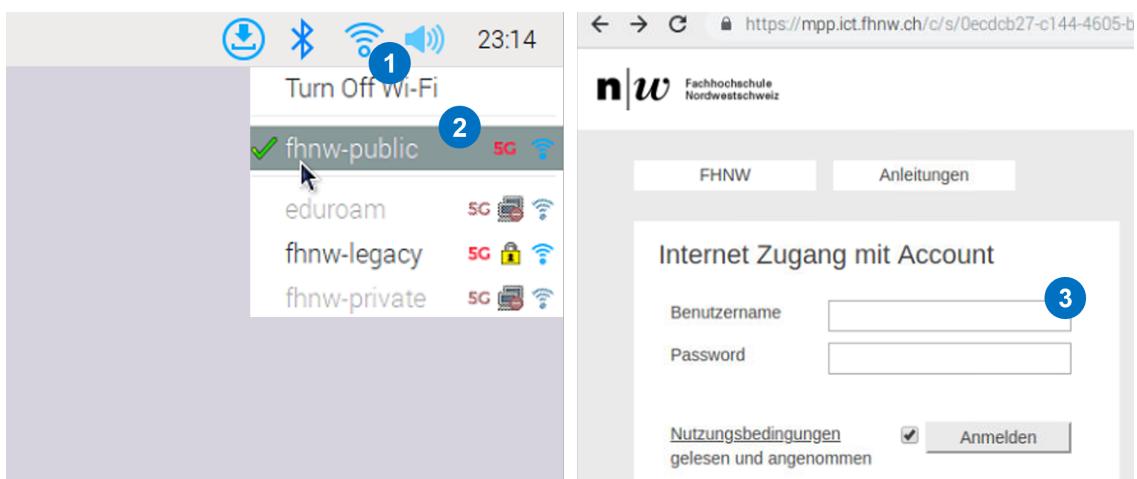
**Abb. A.4.:** Gehäuse öffnen, SD Karte wechseln und das Board aus dem Gehäuse entfernen.

## A.3. WiFi Verbindung einrichten

Das WiFi Network kann auch über den Desktop eingerichtet werden. Das WiFi Symbol in der oberen rechten Ecke des Desktops zeigt die verfügbaren WiFi Netzwerke an. Je nach Einstellungen muss hierbei noch das WiFi Land CH ausgewählt werden. Die WiFi Einstellungen können auch über die Konsole mit dem Programm `raspi-config` vorgenommen werden.

**Hinweis**

Für die Verbindung mit dem Internet am Campus muss das Netzwerk *fhnw-public* ausgewählt werden. Im Browser kann über die Website <https://mpp.ict.fhnw.ch> mit dem FHNW Login oder über das Mobiltelefon die Verbindung hergestellt werden.



**Abb. A.5.:** WiFi Verbindung über den Desktop auf dem Campus herstellen. Das Netzwerk *fhnw-public* wählen und auf der Website <https://mpp.ict.fhnw.ch> die Verbindung mit dem FHNW Account oder via Mobile herstellen.

#### A.4. SSH Verbindung herstellen



**Abb. A.6.:** Shell über die Softwareleiste öffnen

Das SSH Protokoll ermöglicht den direkten auf den Raspberry Pi über das Konsolenfenster. Hierfür wird die IP Adresse des Raspberry Pi benötigt. Die IP Adresse kann über das Programm `ifconfig` oder `ip addr` in der Konsole abgefragt werden.

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.0.247 netmask 255.255.255.0 broadcast 192.168.0.255
                inet6 2a02:aa12:9102:780:feb9:b074:8e31 prefixlen 128 scopeid 0x0<global>
                inet6 2a02:aa12:9102:780:a3cd:afa2:31fb prefixlen 64 scopeid 0x0<global>
                inet6 fe80::753c:73b:9edc:9007 prefixlen 64 scopeid 0x20<link>
                        ether b8:27:eb:3a:fc:27 txqueuelen 1000 (Ethernet)
                        RX packets 7141 bytes 1182587 (1.1 MiB)
                        RX errors 1 dropped 3 overruns 0 frame 1
                        TX packets 8127 bytes 3473107 (3.3 MiB)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

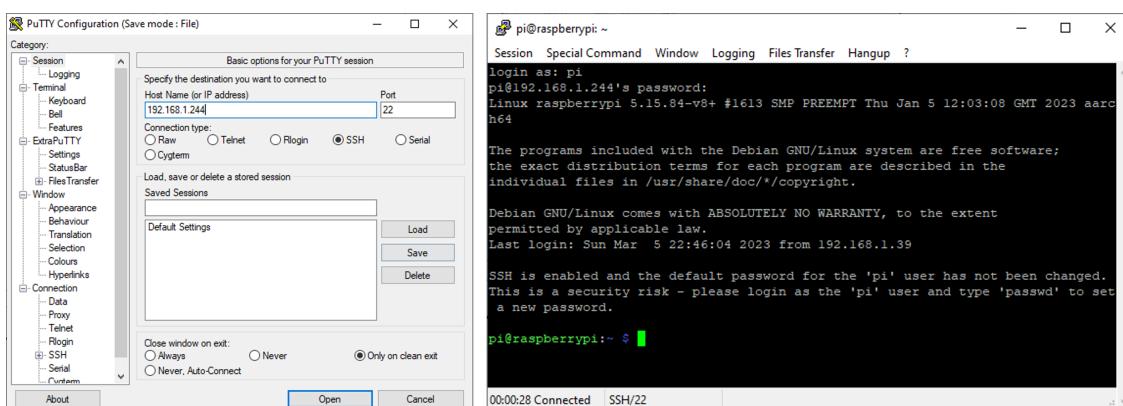
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
                inet6 ::1 prefixlen 128 scopeid 0x10<host>
                        loop txqueuelen 1000 (Lokale Schleife)
                        RX packets 9929 bytes 4827194 (4.6 MiB)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 9929 bytes 4827194 (4.6 MiB)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.0.51 netmask 255.255.255.0 broadcast 192.168.0.255
                inet6 2a02:aa12:9102:780:545:58b6:db3f:10c0 prefixlen 64 scopeid 0x0<global>
                inet6 fe80::8fe1:1525:c108:70ef prefixlen 64 scopeid 0x20<link>
                        ether b8:27:eb:6f:a9:72 txqueuelen 1000 (Ethernet)
                        RX packets 13 bytes 2120 (2.0 KiB)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 30 bytes 5301 (5.1 KiB)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

**Abb. A.7.:** Die IP-Adresse für das Wi-Fi ist unter `wlan0` bei `inet` aufgeführt. Falls der Raspberry Pi mit einem Ethernet Kabel mit dem Netzwerk verbunden ist, wird die IP Adresse unter `eth0` bei `inet` aufgeführt.

Für die Verbindung mit SSH zu Raspberry Pi existieren unterschiedliche Clients:

- Direkt über die Eingabeaufforderung in Windows mit `ssh`
- PuTTY <https://putty.org> (win) ein SSH und telnet client für Windows (auch portable Nutzung ohne Installation möglich)
- Tabby <https://tabby.sh> (win, mac, linux) ein modernes open source Terminal für lokale Shell, Serielle Schnittstelle, SSH und Telnet Verbindungen, File Transfer mit SFTP und Konfigurationsmöglichkeiten (auch portable Nutzung ohne Installation möglich)

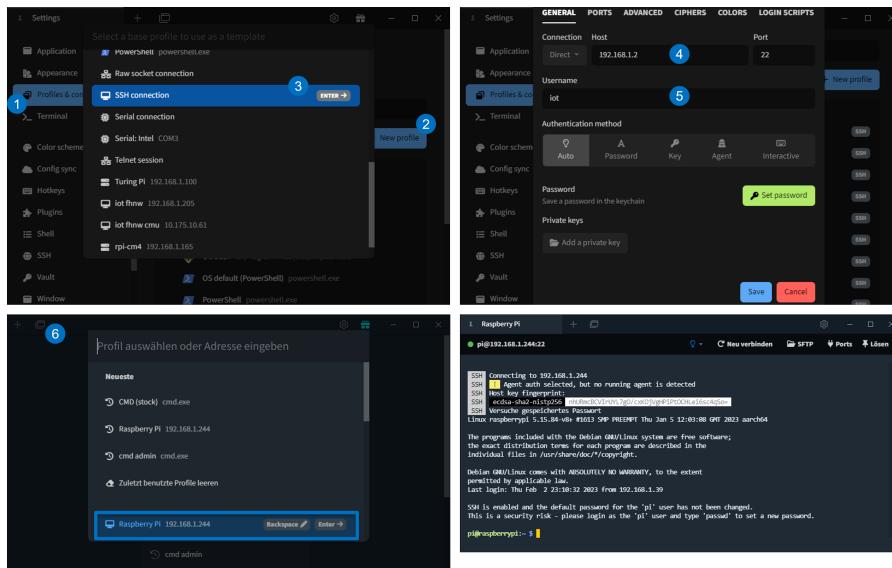


**Abb. A.8.:** SSH Verbindung mit Putty einrichten

## Tabby

Neues SSH Verbindungsprofil mit Tabby erstellen mit der IP Adresse des Raspberry Pi erstellen.

1. Unter Einstellungen / Profile & Verbindungen ein Neues Profil anlegen
2. In der Auswahl SSH-Verbindung wählen
3. Unter Host die IP Adresse Benutzername des Raspberry Pi setzen und speichern

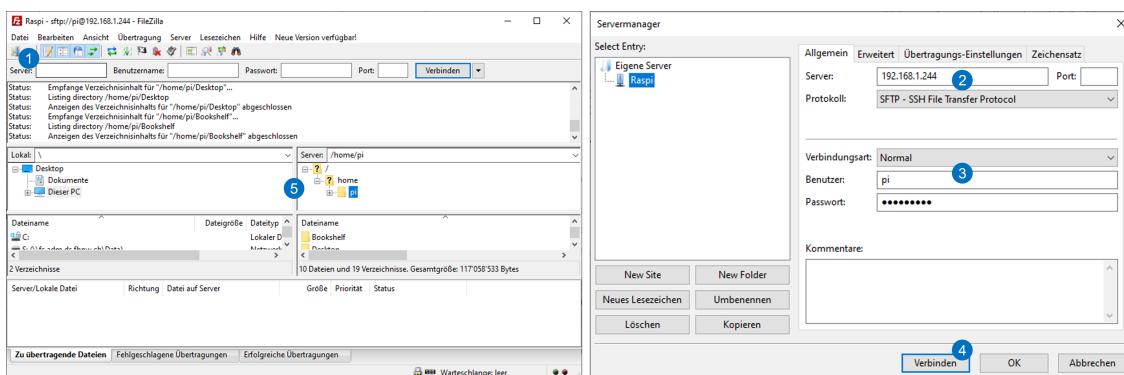


**Abb. A.9.:** SSH Verbindung mit Tabby einrichten (1-5) und anwenden (6).

## Datentransfer mit SFTP

Remote Datentransfer mit SFTP dem SSH File Transfer Protokoll ermöglicht ein einfaches Verwalten und auch sichern der Daten. Hierbei eignet sich die Software [FileZilla](#) oder [WinSCP](#) für den Datentransfer.

In Filezilla eine SFTP Verbindung über Datei/Servermanager herstellen mit Server: <IP Adresse Raspberry Pi>, Protokoll: SFTP, und bei der Verbindungsart normal wählen und dort Benutzername und Passwort des Raspberry Pi Users angeben.



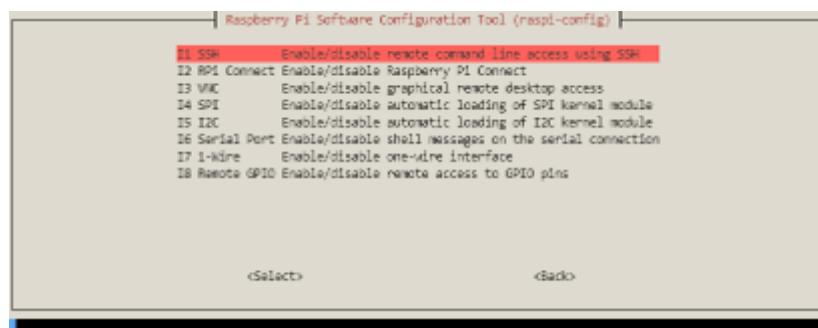
**Abb. A.10.:** Datentransfer mit FileZilla. Eine neue Verbindung im Servermanager (1) erstellen unter Server die IP-Adresse des Raspberry Pi setzen mit dem SFTP Prokoll, und bei der Verbindungsart *normal* wählen und Benutzername und Passwort des Raspberry Pi angeben und mit “Verbinden” (4) die SFTP Verbindung aufzubauen und im Hauptfenster (5) Daten transferieren.

## A.5. Mobilen Hotspot nutzen

Auf Windows kann ein mobiler Hotspot eingerichtet werden, um den Raspberry Pi mit dem Internet zu verbinden. Hierfür wird die Internetverbindung des Computers über das WLAN mit dem Raspberry Pi geteilt. Einen Mobilen Hotspot auf Windows einrichten, den Raspberry Pi mit dem Netzwerk verbinden und die IP Adressen der verbundenen Raspberry Pi's werden im Reiter Mobiler Hotspot aufgeführt.

## A.6. Raspberry Pi Config

Die Grundlegenden Einstellungen des Raspberry Pi können mit dem Programm `raspi-config` vorgenommen werden, entweder über die Konsole oder über den Desktop. Beispielsweise kann dort SSH aktiviert werden, das Tastaturlayout geändert werden oder die Zeitzone angepasst werden. Oder weitere Schnittstellen wie I2C, SPI oder UART aktiviert werden.



**Abb. A.11.:** Über die Konsole `raspi-config` starten und über das Konsolenmenu die Einstellungen anpassen. Im Beispiel wird über das Interface Options Menu SSH aktiviert.

## A.7. Raspberry Pi aktualisieren

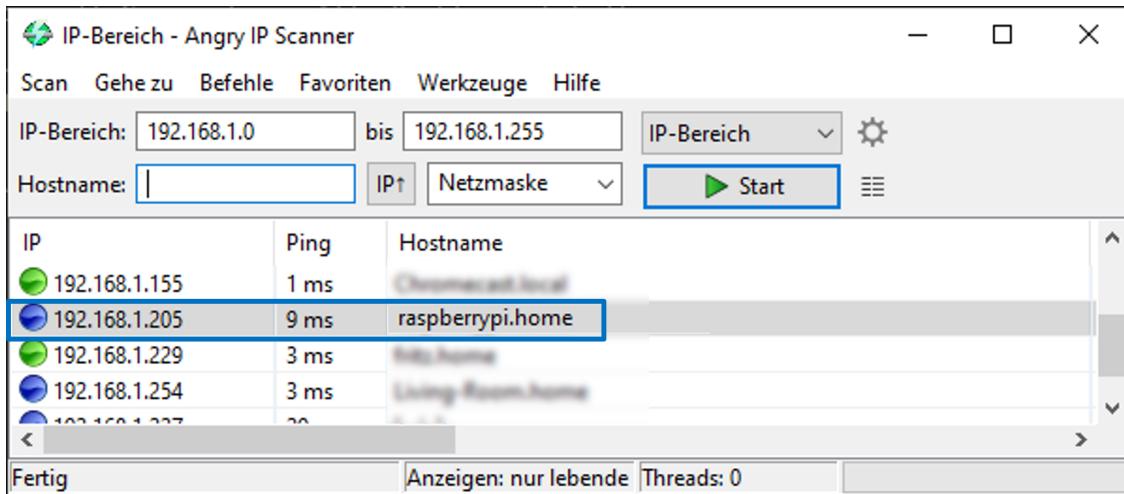
Generell empfiehlt es sich den Raspberry Pi aktuell zu halten und vor jeder Installation von Softwarepaketen folgende Befehle durchzuführen

```
sudo apt update
sudo apt upgrade
sudo apt-get clean
```

## A.8. Raspberry Pi im Netzwerk auffinden

Sobald der Raspberry Pi und auch andere Geräte mit dem Netzwerk verbunden sind kann deren IP Adresse mit Scanner Tools wie [Angry IP Scanner](#), `arp -a`[<sup>a</sup>] `arp -a` listet die IP- und die MAC Adressen der einzelnen Geräte. Es kann ohne das Aufführen von Hostnamen jedoch aufwendig werden, die einzelnen Einträge durchzugehen um zu eruieren. Tipp: `arp -a` vor dem Booten des Raspberry Pi ausführen und ein zweites mal nachdem der Raspberry Pi vollständig gebootet hat nochmals und eruieren welche IP Adresse hinzugekommen ist.] oder `nmap` gefunden werden. Für Android und iOS existieren diverse Scanner Apps wie [Fing](#) oder [Net Analyzer](#).

Dienste, die dass Netzwerk durchleuchten wie der Angry IP Scanner sind in grösseren Netzwerken wie bei Firmen oder Universitäten aus Sicherheitsgründen blockiert. Für den Nutzen im lokalen Netzwerk zu Hause ist der Angry IP Scanner jedoch ein nützliches Tool.

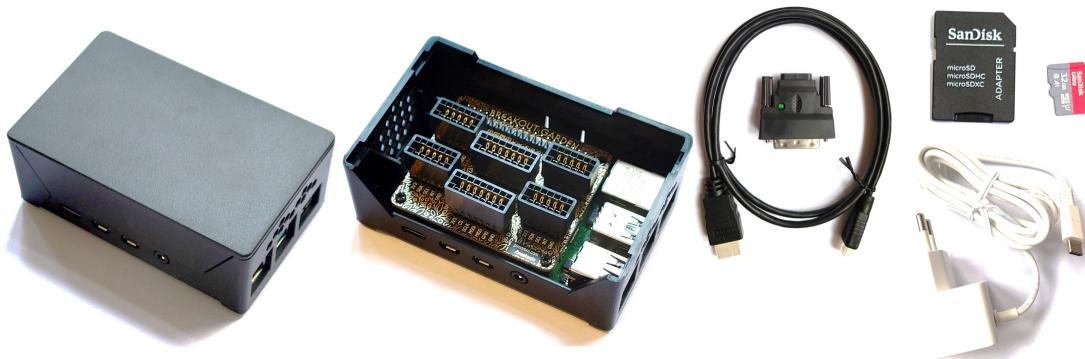


**Abb. A.12.:** Beispiel Scan eines lokalen Netzwerks mit dem Angry IP Scanner

**Anhang B**

# Raspberry Pi Sensorbox

## B.1. Raspberry Pi 4 Set



**Abb. B.1.:** Inhalt des Raspberry Pi 4 Set, mit dem Brekaout Garden HAT, HDMI-HDMI Mini Kabel, Netzteil mit USB-C Anschluss und MicroSD Karte mit Adapter

**Tab. B.1.:** Inhalt des Raspberry Pi 4 Set

---

### Inhalt / Stückliste

---

Raspberry Pi 4B - 4G	<a href="#">Raspberry Pi</a>
Pimoroni Breakout Garden HAT (I2C+SPI)	<a href="#">Pimoroni</a>
HighPi Pro Case for Raspberry Pi 4	<a href="#">HiPi</a>
Raspberry Pi 15W USB-C Netzteil	<a href="#">Raspberry Pi</a>
HDMI-HDMI Mini Kabel <sup>1</sup>	
MicroSD Karte mit SD Adapter - 32Gb	
HDMI-DVI Adapter	

---

<sup>1</sup>HDMI Mini Anschluss schnell defekt

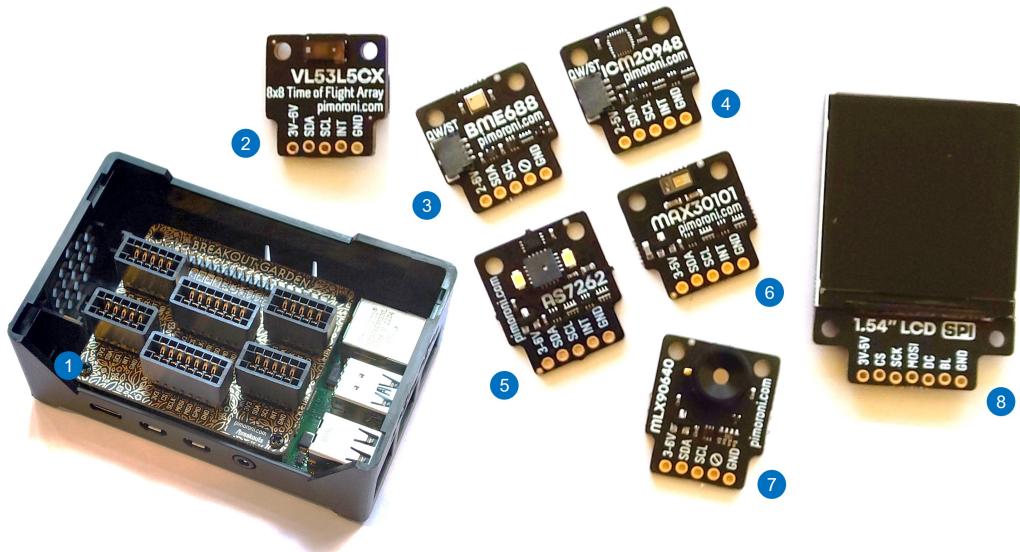
## B.2. Sensorbox

Folgender Abschnitt gibt eine Übersicht der in diesem Kurs verwendeten Sensoren. Die Sensoren sind auf kleinen Platinen (Pimoroni-Breakouts) mit Randanschlüssen montiert und können einfach auf der *Breakout Garden HAT* Erweiterung auf den Raspberry Pi ohne löten und verkabeln montiert werden.

Die Sensorbox enthält mehrere typische IoT Sensoren, die in vielen IoT Geräten Verwendung finden und im Geomatikkontext von Interesse sind.

**Tab. B.2.:** Sensorbox Inhaltsübersicht

Sensor	Beschreibung	Produkt, Datenblatt, Library
BME688	Temperatur, Luftdruck, Luftfeuchtigkeit & Gas	<a href="#">BME688 Breakout</a> , <a href="#">Bosch BME688</a> , <a href="#">bme680-python</a>
ICM20948	9DoF Motion Accelero-, Gyro-, Magnetometer	<a href="#">ICM20948 Breakout</a> , <a href="#">ICM 20948</a> , <a href="#">icm20948-python</a>
VL53L5CX	Time of Flight ToF – 8x8 Multizone	<a href="#">VL53L5CX Breakout</a> , <a href="#">VL53L5CX</a> , <a href="#">vl53l5cx-python</a>
AS7262	Spektral Sensor 6 Kanäle: 450, 500, 550, 570, 600, 650 nm	<a href="#">AS7262 Breakout</a> , <a href="#">AS7262</a> , <a href="#">as7262-python</a>
MAX30101	Herzfrequenz, Oximeter, Rauchsensor	<a href="#">MAX30101 Breakout</a> , <a href="#">MAX30101</a> , <a href="#">max30105-python</a>
MLX90640	Thermal Kamera 32x24pixel Breakout – Wide angle (110°) 1.54'' LCD LCD Bildschirm SPI 240x240pixels	<a href="#">MLX90640 Breakout</a> , <a href="#">MLX90640</a> , <a href="#">mlx90640-libraryAdafruit MLX90640</a> <a href="#">1.54" SPI Colour Square LCD (240x240)</a> , <a href="#">st7789-python</a>
	Adafruit I2C Hub Qwiic/Stemma QT 5 Port Hub	<a href="#">I2C 5 Port Hub</a>



**Abb. B.2.:** Raspberry Pi mit der Breakout Garden HAT Erweiterung (1) von Pimoroni und den Sensoren, (2) VL53L5CX Time of Flight, (3) BME688 Temperatur, Luftfeuchtigkeit und Gas, (4) ICM20948 9DoF Motion Accelero-, Gyro-, Magnetometer, (5) AS7262 Spektral Sensor, (6) MAX30101 Herzfrequenz, Oximeter, (7) MLX90640 Thermal Kamera, (8) 1.54" LCD Bildschirm Rauchsensor

### Hinweis

#### Sensor Ausrichtung beachten

Beim Anschliessen der Sensoren in die Schnittstellen des Breakout Garden **unbedingt** die korrekte Ausrichtung beachten! Die Beschriftung der Anschlüsse auf dem Sensor und dem Breakout Garden müssen übereinstimmen!



**Abb. B.3.:** Sensor links korrekt angeschlossen, rechts falsch ausgerichtet angeschlossen.

**Anhang C**

# Shell Cheat Sheet

Einen unvollständige Übersicht gängiger Konsolen Befehle in Linux.

- **Raspberry Pi runterfahren:** sudo shutdown now
- **Raspberry Pi neu starten:** sudo reboot now

**Tab. C.1.:** Nützliche Linux Befehle für die Kommandozeile (Shell)

Kommando	Kommentar
sudo shutdown now	System jetzt runterfahren / beenden
sudo reboot now	System jetzt neustarten
man <Befehl>	Dokumentation / Manual von Paketen
<Befehl> --help   less	Ruft die Optionen von Befehlen auf, die Option <code>-less</code> ermöglicht, dass mit den Cursortasten in den in längeren Hilfeseiten geblättert werden kann.
whoami	User Informationen anzeigen
pwd	Name des aktuellen Verzeichnisses aufzeigen
ls	Inhalt des aktuellen Verzeichnis zeigen
ls -l	Auflisten aller Dateien und deren Details aktuellen Verzeichnis, wie Schreib- und Leseberechtigungen
ls -a	Alle Dateien inklusive der versteckten anzeigen
ls -lh	Filegrößen menschenfreundlich darstellen
ls -lah	Alle Dateien anzeigen
find	Suche von Dateien mit sehr vielen Optionen für die Suche
cd	Verzeichnis wechseln ( <b>change directory</b> ) Beispiel: <code>cd .. /Documents</code> wechsle in den übergeordneten Ordner mit “..” und gehe in das Verzeichnis “Documents”
mkdir	Erstelle ein Verzeichnis <code>mkdir Projekt</code>
cp	Kopiere eine Datei/Ordner <code>cp &lt;quell-pfad&gt; &lt;ziel-pfad&gt;</code>
mv	Verschieben/Umbenennen einer Datei/Ordner <code>mv &lt;quell-pfad&gt; &lt;ziel-pfad&gt;</code>
rm	Dateien löschen

Kommando	Kommentar
rm -r	Ordner und Inhalte löschen (alternativ <code>rmdir</code> ), Option <code>-f</code> ohne Bestätigung
chmod	Datei- und Verzeichnisrechte ändern (r Lese-, w Schreib- und x Ausführrechte) für Nutzer und Nutzergruppen. Beispiel: <code>chmod 777 Datei.txt</code> gibt allen Nutzern Lese-, Schreib- und Ausführrechte.
zip -r <zip file name> <folder to zip>	Dateien in einem Ordner in eine .zip Datei komprimieren. (Installationsbefehl, falls das Paket <code>zip</code> nicht installiert ist: <code>sudo apt install zip unzip</code> )
unzip datei.zip	Zip-Datei entzippen

**Tab. C.2.:** Nützliche Linux Befehle zur Softwareverwaltung und Installation

Kommando	Kommentar
sudo <Befehl>	Einmalig einen Befehl als su ( <b>super user</b> =Administrator) ausführen (=super user do..) wie beispielsweise eine Installation von Paketen. Beim Ausführen wird das Passwort des su benötigt
sudo apt install <Paketname>	Software über die Software-Verwaltung APT (Advanced Package Tool) installieren
sudo apt remove <Paketname>	Software deinstallieren
sudo apt purge <Paketname>	Konfigurationsdateien nach der Deinstallation entfernen
sudo apt autoremove	Alle nicht mehr benötigten Pakete / Software
sudo apt-cache showpkg <Paketname>	Zeigt alle Informationen über Pakete, Version, Abhängigkeiten und Installation an. <code>apt-cache</code> bietet mit <code>sudo apt-cache dump</code> eine Liste aller installierten Pakete, <code>sudo apt-cache stats</code> die Anzahl der installierten Pakete und deren Abhängigkeiten.
sudo apt-cache search <Paketname>	Ermöglicht die Suche nach Paketen, beispielsweise <code>sudo apt-cache search minesweep</code>
sudo apt update	Laden der aktuellen Paketliste für das Betriebssystem
sudo apt dist-upgrade	Aktualisieren des Betriebssystems (nach <code>sudo apt-get update</code> ausführen)
sudo apt autoremove sudo apt autoclean	Entfernt nicht mehr benötigte Dateien und Pakete
dpkg --get-selections dpkg -l	Alle Softwarepakete auflisten

Kommando	Kommentar
dpkg --get-selections   grep <Paketname>	Gewünschtes Softwarepaket mit grep filtern
sudo dpkg -S <Paketname>	Installationsort von einem Softwarepaket anzeigen
dpkg -l   grep <Keyword>	Nach einem installieren Softwarepaket suchen
dpkg -s <Paketname>   grep Version	Version der installierten Software auflisten
which <Paketname>	Pfad zu den installierten Binaries anzeigen
apt list <Paketname>	Version von einem Package anzeigen
apt list <Paketname> -a	Alle verfügbaren Versionen von einer Software in diesem Repository auflisten
apt-cache policy <Paketname>	Metadaten von einem Softwarepaket auflisten
apt-cache madison <Paketname>	Metadaten von einem Softwarepaket auflisten
sudo apt search <Paketname>	Suche, ob ein Softwarepaket verfügbar ist, ev zuerst die Paketliste mit sudo apt update aktualisieren.
sudo apt-cache search <Paketname>	

**Anhang D**

# Raspberry Pi IoT Image

Für eine möglichst reibungslose Durchführung der Übungen ist ein vorkonfiguriertes Image hilfreich. Dies gewährleistet, dass alle Studierenden mit der gleichen Software arbeiten und die Übungen nicht durch Installationsprobleme verzögert werden.

Für die Übungen mit den Sensoren sind Python und die entsprechenden Sensorbibliotheken erforderlich. Die Installation dieser Bibliotheken ist in den Übungsanleitungen beschrieben. Die Übungen mit dem IoT Stack mit MQTT, InfluxDB, Node-Red und Grafana erfordern eine Installation der Software auf dem Raspberry Pi. Die Installation dieser ist nicht in der Übungsanleitung abgedeckt.

**Hinweis**

**Hinweis:** Für das Kopieren mehrerer Images auf SD Karten können diese einzeln auf die SD Karten geschrieben werden oder empfehlenswert für das Kopieren von mehreren SD Karten lohnt es sich auch eine SD Karten Kopierstation zu verwenden, wie z.B. die [Renkforce Speicherkarten-Kopierstation](#)

## D.1. Raspberry Pi Zugangsdaten

Notiere die Zugangsdaten für den Raspberry Pi und die einzelnen Softwarekomponenten in der untenstehenden Tabelle. Die Zugangsdaten werden für die Übungen benötigt.

**Tab. D.1.:** Zugangsdaten zu den einzelnen Konten notieren, gerade bei Datenbanken geht dies leicht vergessen.

Konto	User	Passwort	Kommentar
Raspberry Pi			
influxdb			organisation:
grafana			

## D.2. Installation Shell Script

Shell Script für die Installation der erforderlichen Bibliotheken Script `install_iot_software.sh` mit dem unten aufgeführten Code erstellen und dem Script die Rechte für die Ausführung setzen mit

```
sudo chmod +x install_iot_software.sh
```

Script mit folgendem `sh install_iot_software.sh` oder `./install_iot_software.sh` Befehl ausführen:

### Hinweis

InfluxDB Version: Die Version der InfluxDB auf die neuste Version im Script anpassen.

Shell Script: `install_iot_software.sh` - Bash Script zu Installation der Software und Libraries für das fächerübergreifende Modul 5200 IoT Installation von:

- Python Libraries Beispielcode für die Pimoroni Sensoren
- Klonen der Libraries mit Beispielen in Documents/Libraries
- Jupyter Notebook
- MQTT, Mosquitto Broker und Clients
- Node-Red
- InfluxDB
- Grafana
- VNC

### Hinweis

**Zeilenumbrüche EOL** Zeilenunterbrüche von Textdateien unterscheiden sich zwischen Unix `LF` und Windows `CRLF` Systemen. Bash Script müssen auf Linux Systemen Unix Zeilenumbrüche (EOL) `LF` aufweisen. Dies ist vor allem dann wichtig, wenn das Script auf einem Windows System erstellt wird. Windows verwendet standardmäßig `CRLF` Zeilenumbrüche. In Visual Studio Code können die EOL Zeichen mit `Ctrl+Shift+P` und `Change End of Line Sequence` auf `LF` umgestellt werden oder über die Statusleiste unten rechts bei `CRLF`. In Notepad++ kann dies über `Edit/EOL Conversion/Unix (LF)` umgestellt werden.

`install_iot_software.sh`: Bash Installations-Script mit Unix Zeilenumbrüchen EOL

```
#!/bin/bash
COL='\033[0;32m' # Primary Color
NC='\033[0m' # No Color
echo "${COL}Raspberry Pi Version${NC}"
hostnamectl
cat /proc/cpuinfo | grep Model

echo "${COL}Raspberry Pi aktualisieren${NC}"
sudo apt update
sudo apt full-upgrade -y
sudo apt autoremove -y

echo "${COL}VNC Installation${NC}"
sudo apt-get install realvnc-vnc-server
sudo apt-get install realvnc-vnc-viewer

echo "${COL}i2c-tools Installation${NC}"
sudo apt install python3-smbus
sudo apt install -y i2c-tools

echo "${COL}Mosquitto Server, Clients und Python Libraries Installation${NC}"
sudo apt install mosquitto -y
sudo apt install mosquitto-clients -y
sudo systemctl enable mosquitto.service
sudo systemctl restart mosquitto

echo "${COL}Node-Red Installation${NC}"
curl -sL https://raw.githubusercontent.com/node-red/linux-
↳ installers/master/deb/update-nodejs-and-nodered -o
↳ update-nodejs-and-nodered.sh
chmod +x update-nodejs-and-nodered.sh
bash update-nodejs-and-nodered.sh --confirm-root --confirm-pi
↳ --confirm-install --no-init
rm update-nodejs-and-nodered.sh
sudo systemctl enable nodered.service
sudo systemctl start nodered.service

echo "${COL}InfluxDB Installation${NC}"
wget https://dl.influxdata.com/influxdb/releases/influxdb2-2.7.1-arm64.deb
sudo dpkg -i influxdb2-2.7.1-arm64.deb
sudo service influxdb start

echo "${COL}Grafana Installation${NC}"
sudo apt-get install -y apt-transport-https software-properties-common wget
sudo mkdir -p /etc/apt/keyrings/
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee
↳ /etc/apt/keyrings/grafana.gpg > /dev/null
```

```
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com  
↳ stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list  
sudo apt-get update  
sudo apt-get install grafana -y  
  
sudo systemctl daemon-reload  
sudo systemctl start grafana-server  
sudo systemctl enable grafana-server.service  
sudo apt autoremove -y  
  
echo "${COL}Klonen der Pimoroni Python Bibliotheken${NC}"  
echo "${COL}in Documents/Libraries${NC}"  
cd Documents  
mkdir Libraries  
cd Libraries  
git clone https://github.com/pimoroni/st7789-python  
git clone https://github.com/pimoroni/bme680-python  
git clone https://github.com/pimoroni/icm20948-python  
git clone https://github.com/pimoroni/as7262-python  
git clone https://github.com/pimoroni/max30105-python  
git clone https://github.com/pimoroni/vl53l5cx-python  
git clone https://github.com/pimoroni/mlx90640-library  
cd ..../
```

Zip Datei des Ordners Libraries erstellen für das Backup der Beispielcode der einzelnen Sensoren.

```
cd ~/Documents/  
zip -r Libraries.zip Libraries
```

## Installation der Python Libraries mit *venv*

Mit dem Upgrade auf *Raspberry Pi Bookworm* wird Python auf die Version 3.11 aktualisiert. Ab dieser Version können Python Libraries nicht mehr systemweit installiert werden. *Python Virtual Environment* sind nun erforderlich, entweder pro Projekt oder pro Userkonto siehe <http://rptl.io/venv> für weitere Informationen.

Erstelle eine Python Virtual Environment .env im Homeordner ~/ mit folgenden Befehlen für den User iot und aktiviere die Virtual Environment mit source ~/.env/bin/activate und deaktiviere die Virtual Environment mit deactivate. Wenn die Virtual Environment projektabasiert im Projektordner erstellt wird reicht es den Pfad zum Projektordner anzugeben z.B. python -m venv ~/Development/Projekt1/env und die Virtual Environment wird im Projektordner erstellt.

```
python -m venv ~/.env
source ~/.env/bin/activate
```

Requirements.txt Datei erstellen mit den Python Libraries, die in der Virtual Environment installiert werden sollen. Die Datei kann mit `nano requirements.txt` erstellt und mit `Ctrl+o` und `Ctrl+x` gespeichert werden. Python Libraries in der Virtual Environment installieren mit `pip install -r requirements.txt`. Nutze für die Installation folgende requirements.txt Datei. Kontrolliere mit `pip list` ob die Libraries installiert wurden.

*requirements.txt:* Python Libraries für die Pimoroni Sensoren

```
matplotlib
scipy
pigments
numpy
jupyter
rpi.gpio
spidev
pillow
numpy
st7789
bme680
icm20948
as7262
max30105
vl53l5cx-ctypes
RPi.GPIO
adafruit-blinka
```

## D.3. Konfigurationsdateien editieren

### Raspberry Pi Einstellungen für I2C, SPI, SSH, VNC anpassen

**Interface Options aktivieren** In den Raspberry Pi Konfigurationseinstellungen mit `sudo raspi-config` die Interface Optionen *I2C, SPI, SSH und ev VNC (Virtual Network Computing) oder RPi Connect* aktivieren. VNC, RPi Connect Hollingworth (2024)<sup>1</sup> werden nur benötigt, wenn der Raspberry Pi über Fernzugriff gesteuert werden soll.

---

<sup>1</sup>RPi Connect erlaubt über einen Relay Server die Verbindung zu einem Raspberry Pi herzustellen über WebRTC wie Browser Clients für Zoom, Slack, Microsoft Teams. Der Relay Server wird nur für die Erstellung der Verbindung benötigt und wird von Raspberry Pi kostenlos betrieben. RPi Connect installieren mit `sudo apt install rpi-connect` und mit `sudo reboot` neu starten.



**Abb. D.1.:** Über die Konsole `raspi-config` starten und über das Konsolenmenu (links) die Einstellungen in den *Interface Options* (rechts) *I2C, SPI, SSH und ev VNC* aktivieren.

## Baud rate des I2C Protokolls anpassen

Baud rate des I2C Protokolls mit `sudo nano /boot/firmware/config.txt` öffnen und folgende Zeile ergänzen. Mit `Ctrl+o` und `Ctrl+x` speichern (siehe [Raspberry Pi Ltd, 2024b](#)).

```
dtparam=i2c_arm=on, i2c_arm_baudrate=400000
```

## Mosquitto Server Konfiguration

**Mosquitto Server Konfiguration** anpassen mit `sudo nano /etc/mosquitto/mosquitto.conf` und am Ende der Datei folgendes einfügen ([Eclipse Foundation, 2021](#)).

```
listener 1883
allow_anonymous true
```

Mosquitto Server neu starten mit `sudo systemctl restart mosquitto`

**Node-Red Einstellungen initialisieren** mit `sudo node-red admin init`

- yes installation customise settings
- yes keep settings file at default location
- no setup user security
- yes enable project features
- select *manual* workflow
- select *default theme*
- yes allow function nodes to load external modules

```
Node-RED Settings File initialisation
=====
This tool will help you create a Node-RED settings file.
```

```
□ Settings file · /root/.node-red/settings.js

User Security
=====
□ Do you want to setup user security? · No

Projects
=====
The Projects feature allows you to version control your flow using a local git
↳ repository.

□ Do you want to enable the Projects feature? · Yes
□ What project workflow do you want to use? · manual - you must manually commit
↳ changes

Editor settings
=====
□ Select a theme for the editor. To use any theme other than "default", you
↳ will need to install @node-red-contrib-themes/theme-collection in your
↳ Node-RED user directory. · default
□ Select the text editor component to use in the Node-RED Editor · monaco
↳ (default)

Node settings
=====
□ Allow Function nodes to load external modules? (functionExternalModules) ·
↳ Yes

Settings file written to /root/.node-red/settings.js
```

## D.4. Installationen testen und initialisieren

### Mosquitto Broker

Testen ob der Mosquitto Broker und Clients lokal auf dem Raspberry Pi funktionieren.  
Erstelle einen Subscriber der für das Topic `iot/temperature` eine Subscription erstellt.

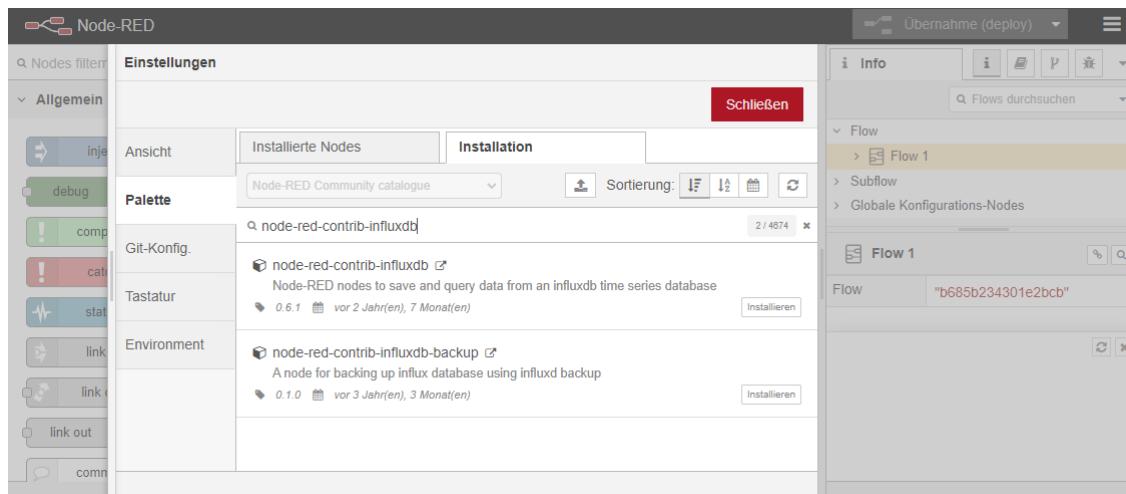
```
mosquitto_sub -h 127.0.0.1 -v -t 'iot/temperature'
```

Öffne ein zweite Shell und erstelle einen Publisher für dasselbe Topic

```
mosquitto_pub -h 127.0.0.1 -t 'iot/temperature' -m 'Aussentemperatur: 22°
↪ Celsius'
```

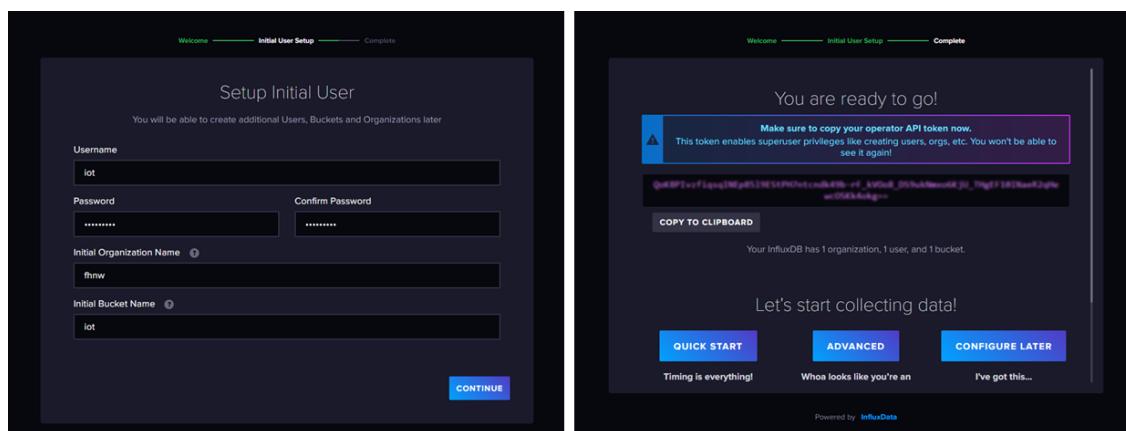
## Node-Red

Node-Red Server testen : <IP-Adresse>:1880 <http://192.168.1.205:1880> Influxerweiterung *Palette* installieren: *Hamburgermenu oben-rechts / Paletten verwalten* und unter *Palette / Installation* die Palette `node-red-contrib-influxdb` installieren.



## InfluxDB

InfluxDB Server testen: <IP-Adresse>:8086 <http://192.168.1.205:8086> Den *Initial User* erstellen mit *Username*, *Password*, *Organisation* und einem *Bucket Name*, der ersten Datenbank. Im Anschluss mit Logout/Login den Zugang verifizieren.

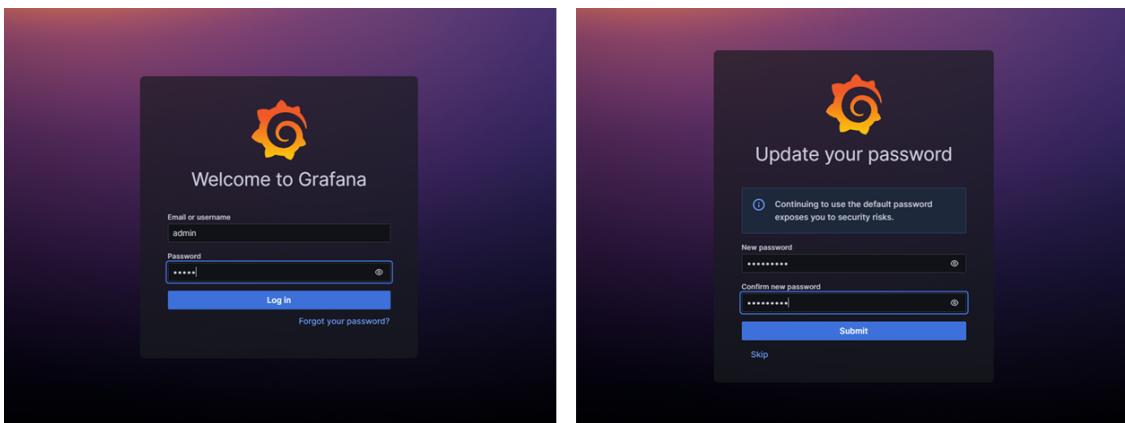


### Hinweis

**Operator API token speichern!** Das Superuser Passwort wird nur einmal angezeigt. Das **Operator API token** kopieren und an einem sicheren Ort speichern!  
Nach dem Logout/Login ist das Passwort nicht mehr sichtbar.

## Grafana

Grafana Server testen: <IP-Adresse>:3000 <http://192.168.1.205:3000> Grafana startet mit dem Default Admin User: admin und Passwort: admin. Das Passwort ändern und im Anschluss mit Logout/Login den Zugang verifizieren.



## Jupyter Notebook

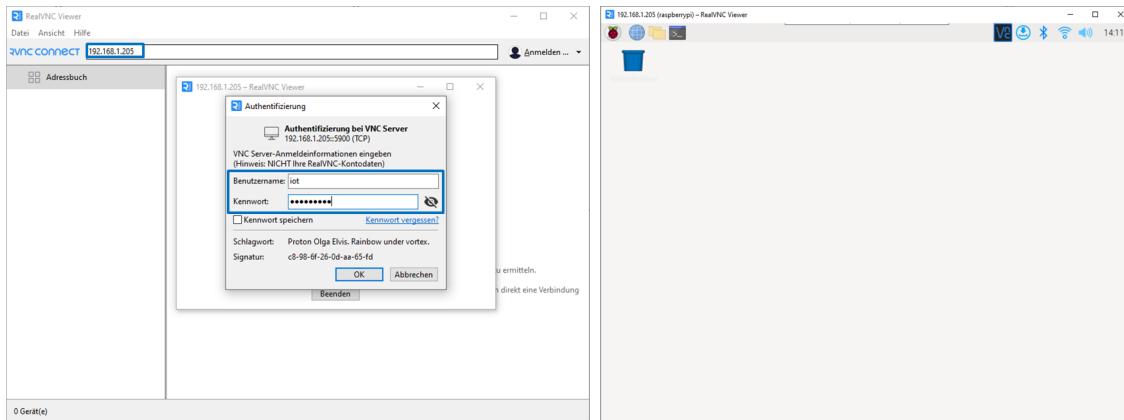
Jupyter Notebook testen <IP-Adresse>:9999 <http://192.168.1.205:9999>

```
# jupyter notebook installation testen
jupyter-notebook --no-browser --ip=192.168.1.205 --port 9999 --notebook-dir
↳ ~/Documents
```

## VNC Testen

VNC Viewer auf dem PC installieren. [Real VNC Viewer](#) (standalone .exe, d.h. es ist keine Installation erforderlich).

**Hinweis:** Beim Starten des VNC Viewers muss man sich nicht anmelden sondern nur die Option **Verwenden Sie RealVNC Viewer ohne sich anzumelden** wählen.



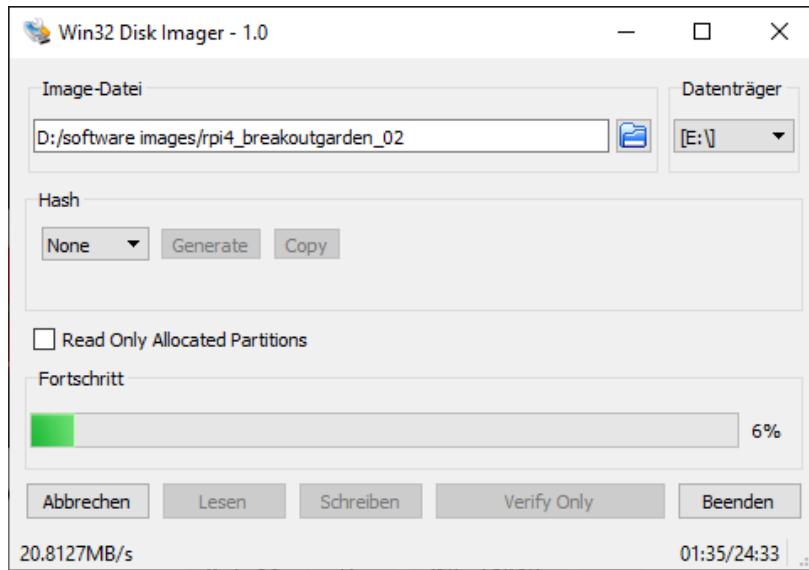
## D.5. Raspberry Pi Image verkleinern

Das Raspberry Pi Image kann mit dem Tool [PiShrink](#) verkleinert werden. PiShrink kann auf dem Raspberry Pi oder auf einen anderen Linux System wie Ubuntu oder einer virtuellen Machine mit einem Linux installiert und genutzt werden.

1. Backup Image auf den lokalen Rechner mit [Win32DiskImager](#) lesen und in eine Image Datei schreiben.
2. Virtuelle Machine mit Linux Ubuntu/Debian mit *gemeinsamen Ordner* starten
3. Mit dem Script <https://github.com/Drewsif/PiShrink> das Image verkleinern
4. SD Karte Formatieren mit [SD Card Formatter](#)
5. Image auf SD Karte schreiben mit *Win 32 Disk Imager* oder *Raspberry Pi Imager*
6. Für die Archivierung kann das Image mit 7zip zusätzlich komprimiert werden (Archive Format `.xz` und Kompressionsstufe `Ultra`)

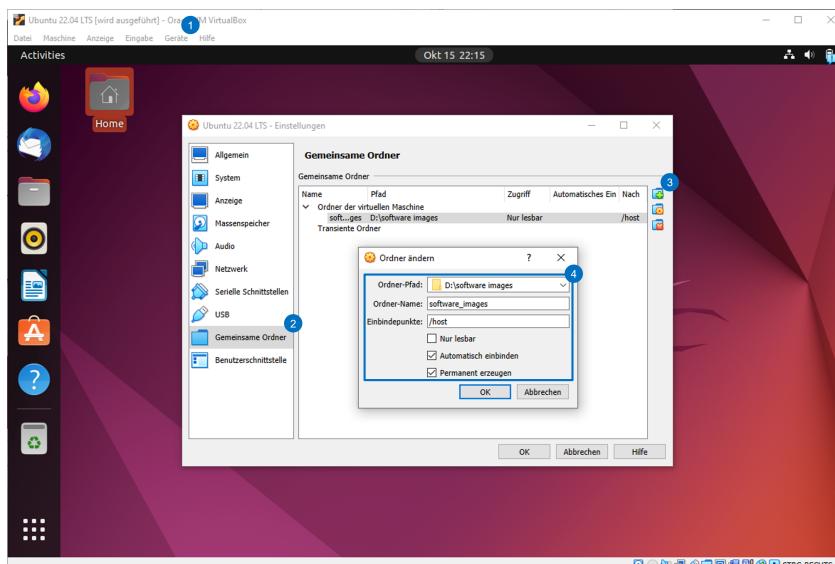
Tutorial: [How I Backup and Shrink My SD Card Images - Youtube](#)

Backup Image mit Win32DiskImager schreiben, mit der Funktion *Lesen/Read* und diese als Image Datei auf den lokalen Rechner speichern.



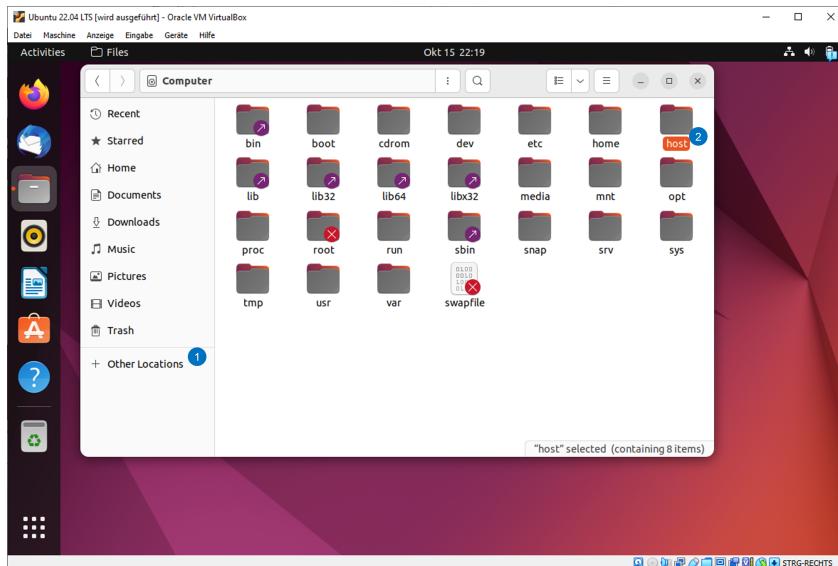
**Abb. D.2.:** Backup Image der SD Karte schreiben mit Win32DiskImager und der Funktion Lesen/Read

Virtuelle Machine mit VM [VirtualBox](#) starten und in den Einstellungen der VM über *Geräte/Gemeinsame Ordner* einen gemeinsamen Ordner zwischen VM und dem lokalen Rechner einrichten.



**Abb. D.3.:** Gemeinsamer Ordner in der VM VirtualBox einrichten

Das Image sollte nun im File Explorer unter "+ Other Locations" als Ordner *host* eingebunden und sichtbar sein.



**Abb. D.4.:** Gemeinsamer Ordner *host* auf der VM Virtualbox eingebunden

Öffne nun ein Terminal damit das Script `pishrink.sh` installiert und ausgeführt werden kann.

**Tipp:** Falls das Terminal nicht öffnet mit `Ctrl+Alt+F3` in das TTY Terminal wechseln.

**Tipp:** Falls der VM User noch keine Adminrechte hat (bei einer Neuinstallation von Ubuntu ist das `root` Passwort noch nicht gesetzt), können diese nachträglich gesetzt werden mit `sudo passwd`. Bei der Passwortabfrage das aktuelle User Passwort eingeben und das neue Root Passwort zweimal eingeben. Anschliessend als Root User anmelden mit `su -`.<sup>2</sup>

GitHub: <https://github.com/Drewsif/PiShrink>

```
# install prerequisites
sudo apt install parted xz-utils
# download the pishrink script
wget -O ./pishrink.sh
↳ https://raw.githubusercontent.com/Drewsif/PiShrink/master/pishrink.sh
# make the script executable
chmod +x ./pishrink.sh
# run pishrink on the .img file
sudo ./pishrink.sh -v /host/<rpi-imagefile>.img
# Komprimiere die Image als .xz Archive (eher langsam)
xz -z -9e /host/<rpi-imagefile>.img
```

<sup>2</sup>Einem User Adminrechte zuweisen: `sudo usermod -a -G sudo <username>`, Kontoinformationen id <username> aufzeigen.

```

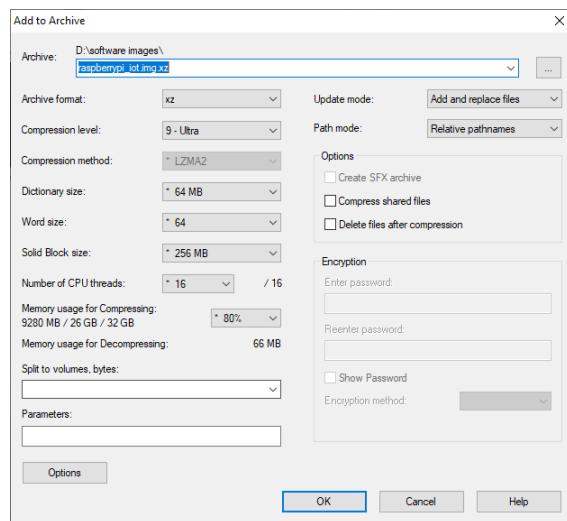
Ubuntu 22.04 LTS [wird ausgeführt] - Oracle VM VirtualBox
Datei Maschine Anzeige Eingabe Geräte Hilfe
vboxuser@vmUbuntu:~$ sudo ./pishrink.sh -v /host/rpi4_breakoutgarden_02.img
pishrink.sh v0.1.3
pishrink.sh: Gathering data ...
Creating new /etc/rc.local
pishrink.sh: Checking filesystem ...
rootfs: 157012/1925760 files (0.2% non-contiguous), 1518392/7725184 blocks
resize2fs 1.46.5 (30-Dec-2021)
pishrink.sh: Shrinking filesystem ...
resize2fs 1.46.5 (30-Dec-2021)
Resizing the filesystem on /dev/loop15 to 1824802 (4K) blocks.
Relocating blocks      XXXXXXXXXXXXXXXXXXXXXXXX
Begin pass 3 (max = 64056)
Scanning inode table  XXXXXXXXXXXXXXXXXXXXXXXX
The filesystem on /dev/loop15 is now 1824802 (4k) blocks long.

pishrink.sh: Shrinking image ...
pishrink.sh: Shrunk /host/rpi4_breakoutgarden_02.img from 30G to 7.3G ...
vboxuser@vmUbuntu:~$ 

```

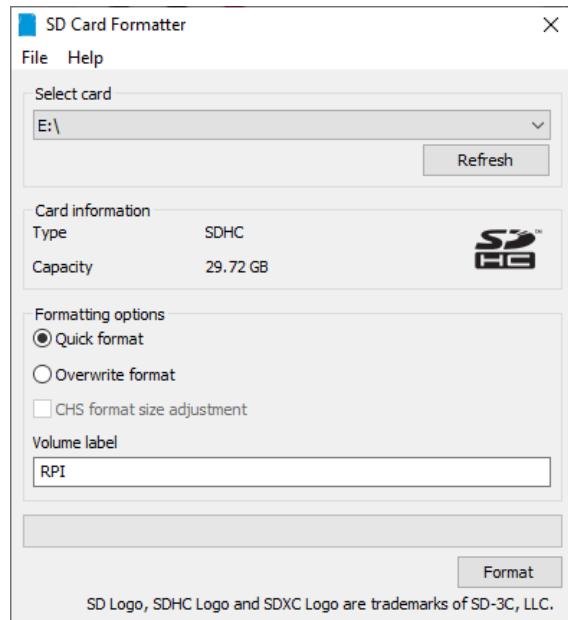
**Abb. D.5.:** Screenshot des Terminals während der Ausführung von pishrink

Image mit 7-Zip zusätzlich komprimieren File Kontextmenü 7-Zip/Add to Archive und als Archive Format .xz sowie den Compression Level 9-Ultra wählen.



**Abb. D.6.:** 7-Zip Einstellungen für die zusätzliche Kompression des Images mit .xz.

Falls die SD Karte schon früher für Raspberry Pi Images benutzt wurde und mehrere Partitionen aufweist, kann es helfen die Karte mit dem [SD Card Formatter](#) nochmals von Grund auf zu formatieren.



**Abb. D.7.:** Screenshot des SD Card Formatter

# Index

- AS7262, 38
- BME688, 28
- Digital Earth, 8
- General Purpose Input Output GPIO, 83
- Generalisierung von Messdaten, 13
- ICM20948, 44
- IMU, 42
- IoT, 6
- IoT Anwendungen, 11
- ISO/OSI Schichtenmodell, 19
- LoRaWAN, 20
- MAX30101, 62
- MEMS, 44
- MLX90640, 56
- Node-Red, 71
- phyphox, 15
- Pins, 83
- Raspberry Pi, 82
- Raspberry Pi Imager, 84
- Raspberry Pi OS, 84
- Raspberry Pi Update, 91
- raspi-config, 90
- Räumliche Analysemethoden, 12
- Sensor Fusion, 15
- Sensor Web Enablement SWE standards, 9
- Sensoren, 16
- SensorML-Standard, 9
- SFTP SSH File Transfer Protokoll, 89
- Single Board Computer SBC, 82
- SSH, 87
- System on Chip SOC, 82
- Tabby, 89
- Ubiquitous Computing, 5
- VL53L5CX, 49