Datenbanken und Peripherie

Die MINT-Labs unterhalten zwei Server und einige Peripherie, um diese an die "reale" Welt anzubinden. Hier erfährst du mehr.

- alfons
- berta
- charlie
- ESPnow-Bridges
- MQTT
- InfluxDB
- KNX

alfons

alfons ist ein Serverplatz bei einem deutschen Anbieter. Hier drauf läuft u.a. dieses Wiki! Weiter sind hier Datenbanken (influxDB) und weitere Servertools installiert sowie ein sogenannter MQTT-Broker. Da alfons eine öffentliche (und statische) IP hat, ist alfons weltweit erreichbar.

Bei Fragen rund um den Server wende dich am besten an Timo.

berta

"berta" war bis Ende 2023 ein Raspberry Pi 4 mit 4GB Arbeitsspeicher. Mittlerweile ist es eine VM auf dem Proxmox.

Auf berta laufen im Wesentlichen folgende Server:

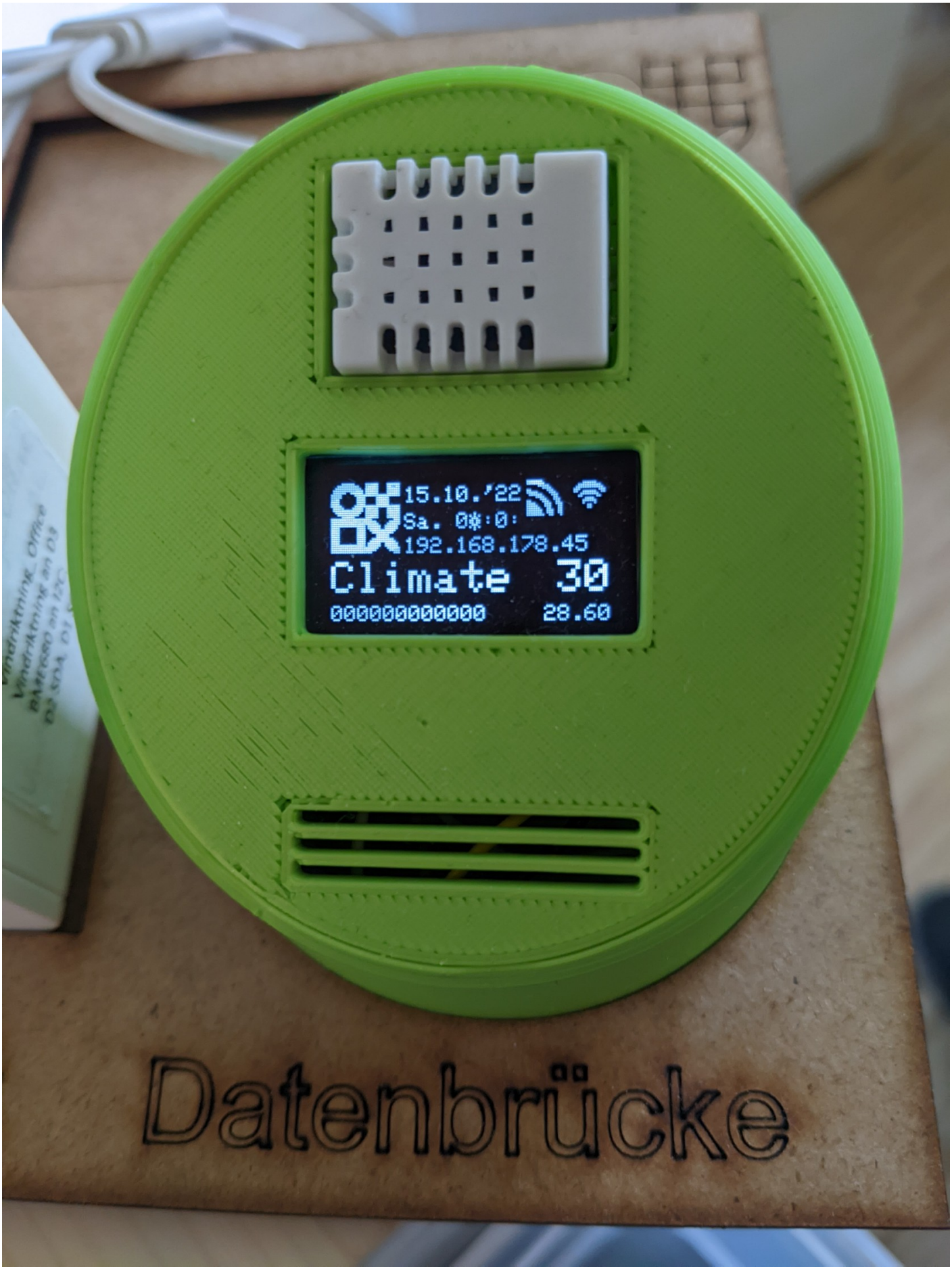
- MQTT-Broker "mosquitto": Er verteilt und empfängt Sensordaten innerhalb des Hauses, etwa von diversen IoT-Geräten. Siehe auch HIER
- Home Assistant: Smart-Home Zentrale. Es ist verlinkt mit dem hausinternen Bussystem "KNX".

charlie

charlie ist ein Server bei Hetzner. Hier läuft im Wesentlichen unsere Nextcloud + Groupware.

ESPnow-Bridges

Derzeit stehen zwei Datenbrücken zwischen dem ESPnow- und dem WLAN-Protokoll im Haus. Eines davon an der Theke von "The Office", eines in der "X-Zone" im 1. OG.


15.10.'22
Sa. 0:0:
192.168.178.45
Climate 30
000000000000 28.60
Datenbrücke

Hardware

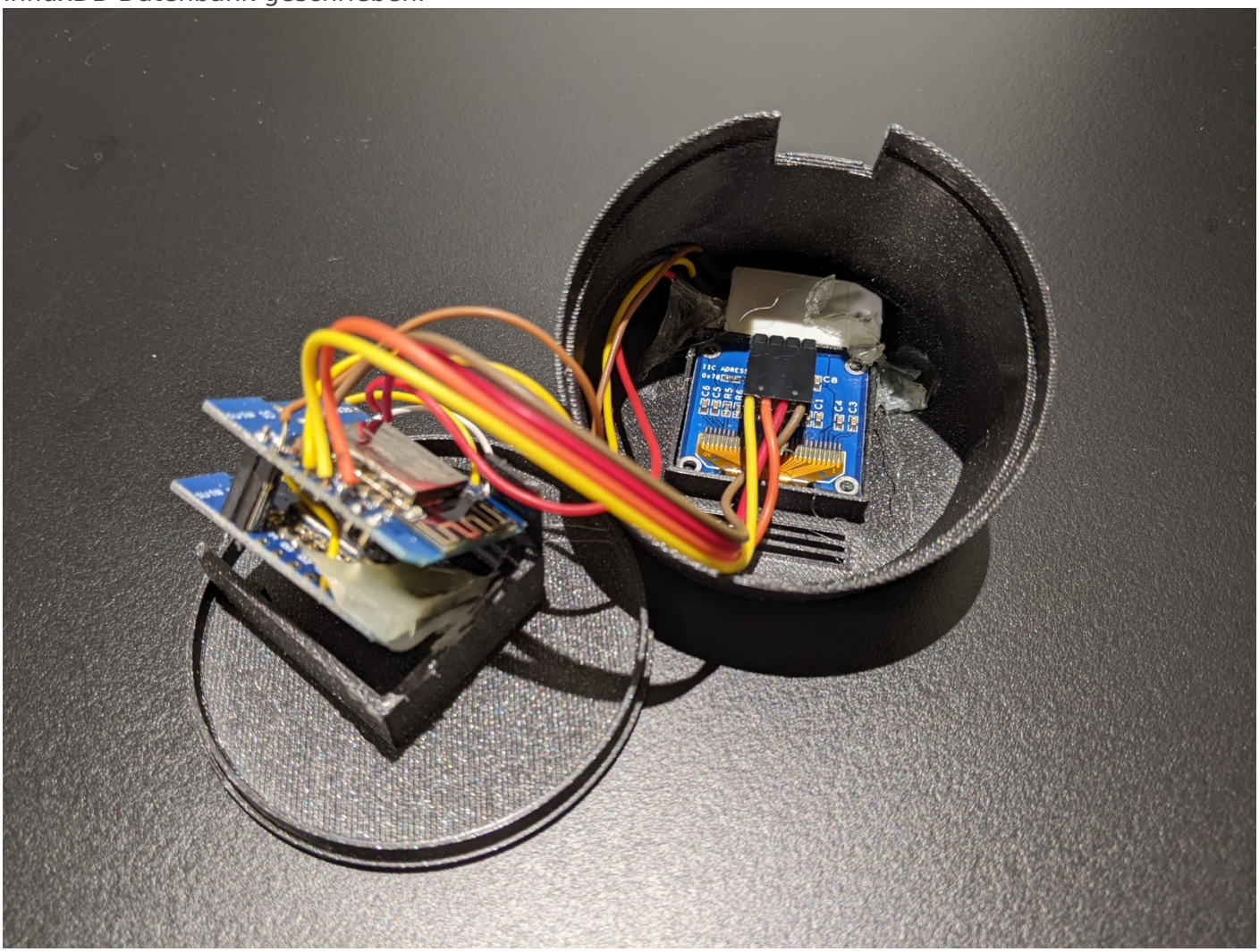
Die Bridges sind in einem kleinen 3D-gedruckten Gehäuse (aus Thingiverse). Wie auf Thingiverse beschrieben ist hier ein DHT22-Sensor (Luftfeuchte und Temperatur) und ein 0,96" OLED-Display verbaut.

Im Bild sieht man eine solche Datenbrücke mit Datum, Wochentag, nicht funktionierender Uhrzeit (Bug, muss gefixt werden!), MQTT-Verbindung (erstes Icon rechts oben), WLAN-Empfang (zweites Icon rechts oben), der eigenen IP-Adresse (hier 192.168.178.45). Es folgt ein Datenpaket wie es via MQTT weitergegeben wird. Der Sender ist ein Klimasensor "Climate" welcher drei Daten sendet: 30 (Luftfeuchtigkeit in %), 28.6 (Temperatur) sowie 000000000000 (Batteriespannung; es gibt in diesem Gerät keine Batterie). Der Sender ist in diesem Fall das eigene Gerät mit dem eingebauten DHT22-Sender.

Innen werkeln zwei D1-mini ESP8266. Einer ist dabei im "normalen" WLAN, ein weiterer horcht nur auf ESPnow-Nachrichten. Sie sind via UART verbunden. Auf ihnen laufen je ein Arduino-Skript, welches größtenteils von MrDIY übernommen wurde.

1. Auf dem D1-mini für ESPnow befindet sich "gateway_receiver_MLR.ino"; dieses Programm "horcht" ständig auf ESPnow-Kommunikation und hat die Software-Mac-Adresse 12:34:56:78:9A:BC
2. Auf dem D1-mini für die WLAN-Kommunikation befindet sich "gateway_min_MLR.ino", an ihm sind auch das OLED und der DHT22 angeschlossen.

Der zweite D1-mini sendet alle ESPnow-Daten, die vom ersten D1-mini kommen, mittels MQTT weiter an den hausinternen Server "berta". Auch nimmt er einmal pro Minute die Messwerte des DHT22-Sensors und schickt diese ebenso an "berta". berta kümmert sich dann um die Weiterleitung an den großen externen Server "alfons". Alle Daten werden in die dort installierte influxDB-Datenbank geschrieben.



Warum eigentlich ESPnow???

ESPnow ist ein vom ESP-Hersteller Espressif entwickeltes Protokoll. Bei "normalem" WLAN müssen sich auch IoT-Geräte immer beim Router anmelden und die Verbindung halten; das kostet aber Strom, was nicht jeder IoT-Sensor gerne hat. Etwa bei autark arbeitenden Wetterstationen, die nur via Solar betrieben werden, gilt es immer, Strom zu sparen. Daher schalten sich diese IoT-Sensoren auch ab, wenn sie keine Daten nehmen oder senden. Nun müsste sich so ein Sensor aber bei jedem Start erneut mit dem Router verbinden, was durchaus mehrere Sekunden dauern kann - Zeit, während der er mit voller WLAN-Leistung funkt! Bei einem ESP32 sprechen wir da von bis zu neun Sekunden und einem Stromverbrauch von bis zu 120mA! Dagegen funktioniert das ESPnow-Protokoll viel einfacher. Der Sender "ruft" einmal kurz seine Datenpakete entweder einem ESPnow-Partner zu (dazu muss er aber die MAC-Adresse des Partners kennen), oder er ruft an alle

verfügbaren Geräte (sog. Broadcasten). Dieses "Rufen" dauert nur wenige zehn bis einhundert Millisekunden! Der Stromverbrauch reduziert sich also drastisch.

MQTT

MQTT, Message Queueing Telemetrie Transport, ist ein offenes Netzwerkprotokoll um vorwiegend kleine Datenpakete (etwa Telemetriedaten) zwischen Geräten ermöglicht.

Hierbei gibt es immer einen Hauptserver, den sogenannten Broker. Ein solcher ist etwa auf berta installiert, in diesem Fall handelt es sich um mosquitto. Dieser Broker sammelt Daten der Client-Geräte ein und verteilt sie wiederum. Dafür können sich Client-Geräte beim Broker für gewisse Themen ("topics") anmelden ("subscriben"). Ändert sich ein Parameter, also etwa ein Messwert in einem topic, sendet der Broker diese neuen Werte an alle Subscriber. Topics sind ähnlich einer Ordnerstruktur geordnet, etwa "house/livingroom/temperature" bezeichnet den Temperatursensor im Wohnzimmer des Hauses. Die Ordnung kann/darf man sich selbst aufbauen, sie sollte nur konsequent für alle beteiligten Geräte bleiben.

Konkret: ein Client könnte etwa eine kleine Wetterstation draußen sein. Diese sammelt jede Stunde Temperatur-, Luftfeuchte- und Luftdruckdaten sowie Daten zur Sonneneinstrahlung. Diese Daten sendet es dann als Datenpaket(e) mit dem topic "weather" an den Broker. Der Broker registriert diese und schickt sie an alle, die sich für das topic "weather" subscribed haben. So jemand könnte zum Beispiel ein Gerät sein, welches die Rolläden je nach Sonneneinstrahlung steuert. Vielleicht interessieren dieses Gerät dann auch gar nicht die anderen Daten der Wetterstation (Luftdruck, Luftfeuchte, Temperatur) und es ist nur am topic "weather/sun" interessiert.

Besonderheiten

- "Last will": ein "last will" eines Clients wird im Broker gespeichert. Stirbt der Client (etwa weil Batterie leer, Abbruch der WiFi-Verbindung, ...) sendet der Broker diesen "last will" an alle Subscriber des Clients. Der "last will" könnte etwa lauten "client xyz disconnected". Andere Clients könnten dann auf diesen letzten Willen entsprechend reagieren.
- "Wildcard subscriptions": das "+"-Zeichen oder das "#" können als sog. Wildcards verwendet werden. Gibt es das topic "temperature" etwa in allen Räumen eines Hauses (house/livingroom/temperature, house/bathroom/temperature, house/kitchen/temperature, house/sleepingroom/temperature...) und man möchte alle Temperaturen auf einmal abfragen, könnte dies so aussehen: house/+/temperature. Ebenso können alle sub-topics einer Hierarchie angesehen werden durch Verwendung von "#": "house/#" gibt alle topics wieder, die unter "house" ablaufen.

Unser Zugang

der User lautet mqtt_user, das Passwort findet sich im MINT-Labs Bitwarden (frage Fabian).

InfluxDB

Unsere InfluxDB-Datenbank ist auf alfons installiert und daher weltweit zugänglich.

KNX

Wir haben einen KNX-Bus im Haus. Es handelt sich um ein Enerix KNX/IP-Gateway.

Das zuständige Gateway hängt im LAN: 192.168.178.54

Die Adressdaten bekommt ihr von Fabian.