



Systemdokumentation

Mikrowellen

Version 0.0.1, 3. November 2022

Niklas & Iwen

Inhalt

[1 Kurzbeschreibung Umgebung 2](#_Toc118355531)

[2 Systemumgebung 2](#_Toc118355532)

[2.1 Hardware 2](#_Toc118355533)

[2.1.1 M5Go 2](#_Toc118355534)

[2.1.2 Raspberry Pi 2](#_Toc118355535)

[2.2 Netzplan 4](#_Toc118355536)

[2.3 Konfiguration IoE-Umgebung 4](#_Toc118355537)

[2.3.1 Konfiguration M5 Stack 4](#_Toc118355538)

[2.3.2 Konfiguration MQTT 5](#_Toc118355539)

[2.3.3 Konfiguration InfluxDB 5](#_Toc118355540)

[2.3.4 Konfiguration Telegraf 6](#_Toc118355541)

[2.3.5 Konfiguration Kapacitor 6](#_Toc118355542)

[2.3.6 Konfiguration Chronograf 6](#_Toc118355543)

[2.3.7 Python 6](#_Toc118355544)

[2.4 Kommunikationsmatrix 7](#_Toc118355545)

[3 Testing 8](#_Toc118355546)

[3.1 Testdrehbuch 8](#_Toc118355547)

[3.1.1 Testfall 1 Lesen und Schreiben in InfluxDB mit Admin-Rechten 8](#_Toc118355548)

[3.1.2 Testfall 2 Nur Zugang zur InfluxDB telegraf mit Benutzer telegrafuser 8](#_Toc118355549)

[3.1.3 Testfall 3 Zusammenspiel Python-Script / Mosquitto / InfluxDB 9](#_Toc118355550)

[3.1.4 Testfall 4 Testen der Alerts 9](#_Toc118355551)

[3.2 Testprotokoll 10](#_Toc118355552)

[3.2.1 Testfall 1 10](#_Toc118355553)

[3.2.2 Testfall 2 10](#_Toc118355554)

[3.2.3 Testfall 3 10](#_Toc118355555)

[3.2.4 Testfall 4 10](#_Toc118355556)

# Kurzbeschreibung Umgebung

Für das Modul „162-IoE“ erstellen wir mit Hilfe von Lichtsensoren eine Oberfläche, welche Anzeigen kann welche Mikrowellen im BBC wann frei sind. Dabei werden das M5Go und ein Raspberry verwendet.

# Systemumgebung

## Hardware

Folgende Hardware wird verwendet:

### M5Go

|  |  |
| --- | --- |
| **Spezifikationen** | |
| **Typ** | **M5Go** |
| **CPU** | **ESP32 240MHz dual core, 600 DMIPS, 520KB SRAM** |
| **Firmware Version** | **V 1.8.3-en** |
| **Flash** | **4MB** |
| **IP-Adresse** | **DHCP** |
| **MAC-Adresse WLAN** | **24:0a:c4:f9:7c:f1** |
| **Sensor 1 Typ / Anschluss** | **ENV II Sensor I2C** |

### Raspberry Pi

|  |  |  |
| --- | --- | --- |
| **Spezifikationen** | | |
| **Typ** | **Raspberry Pi 3 Model B Rev 1.2** | |
| **CPU** | **Quad Core 1.2GHz ARMv7 Processor rev 4 (v7l)** | |
| **OS Version** | **Raspbian GNU/Linux 10 (buster) (Version 10.10)** | |
| **RAM** | **1GB** | |
| **Festspeicher** | **16GB** | |
| **MAC-Adresse WLAN** | **b8:27:eb:8c:81:aa** | |
| **Einstellungen** | | |
| **Aktivierte Schnittstellen** | **SSH** | |
| **IP-Adresse** | **10.0.103.157** | DHCP: Ja |
| **Subnetzmaske** | **255.255.0.0** | |
| **DNS** | **8.8.8.8** | |
| **Standard-Gateway:** | **0.0.0.0** | |

## Netzplan

## Konfiguration IoE-Umgebung

Der M5Go schickt alle paar Sekunden die gemessene Lichtstärke via MQTT an das Topic „microwaves/nr\*“. Ein Python-Script auf dem Raspberry Pi, welches auf das MQTT-Topic „microwaves/nr\*“ abonniert ist, erhält die Information, ob/welche Mikrowellen besetzt sind und schreibt sie in eine InfluxDB. Mit Chronograf werden die gemessenen Temperaturen angezeigt werden. Kapacitor würde es ermöglichen in der Zukunft auf bestimmte Messwerte mit TICKscripts zu reagieren, wird aber noch nicht genutzt.

Telegraf schreibt aktuelle Systemdaten wie RAM-, CPU-, Disk- und Netzauslastung in InfluxDB. Auch diese Werte werden per Telegraf angezeigt.

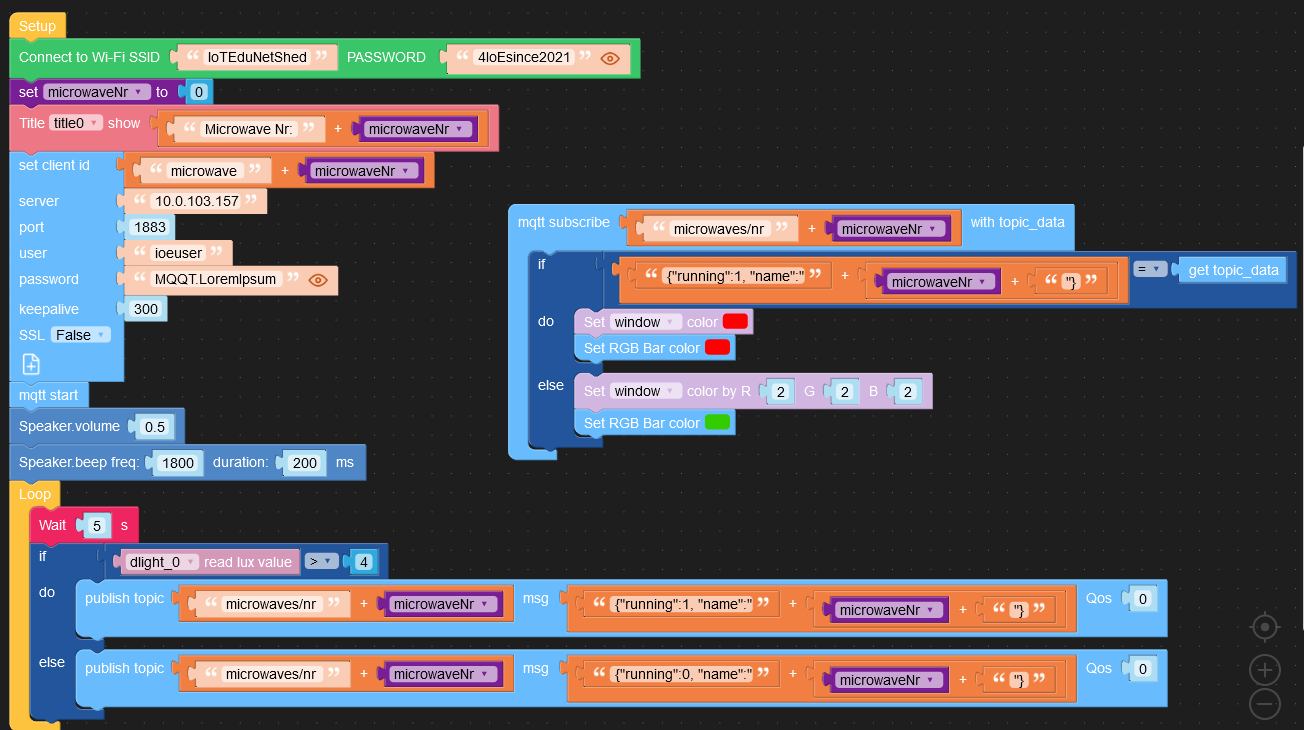
### Konfiguration M5 Stack

#### SSID

SSID: IoTEduNetShed  
Passwort: 4IoEsince2021

#### Ausgeführtes Script

microwave.m5f (als Test: MicrowaveDummy.m5f)



#### Angeschlossene Unit

DLIGHT über I2C

### Konfiguration MQTT

#### Installierte apt-Pakete:

* mosquitto
* mosquitto-clients

#### Konfigurationsdatei

/etc/mosquitto/mosquitto.conf

#### Authentifizierung

User: ioeuser  
Passwort: MQQT.LoremIpsum

#### Port

1883

### Konfiguration InfluxDB

#### Apt key

wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -

#### Apt Repository

echo "deb https://repos.influxdata.com/debian $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/influxdb.list

#### Installierte apt-Pakete:

influxdb

#### Konfigurationsdatei

/etc/influxdb/influxdb.conf

#### Benutzer

|  |  |  |
| --- | --- | --- |
| **Benutzer** | **Password** | **Rechte** |
| Admin | Influx.LoremIpsum | Alle Rechte auf allen Datenbanken |
| telegrafuser | Influx.LoremIpsum | Alle Rechte auf Datenbank telegraf |

#### Datenbanken

* ioe : Temperaturmesswerte Shedhalle, Keine Retention Policy (default)
* telegraf : Systemdaten CPU, RAM, Network usw. des Raspberry Pi. Retention Policy von 4 Tagen.

#### Port

8086

### Konfiguration Telegraf

#### Installierte apt-Pakete:

telegraf

#### Konfigurationsdatei

/etc/telegraf/telegraf.conf

### Konfiguration Kapacitor

#### Installierte apt-Pakete:

kapacitor

#### Konfigurationsdatei

/etc/kapacitor/kapacitor.conf

#### Port

9092

#### Account Pushover

None

#### TICKscripts

None

### Konfiguration Chronograf

#### Installierte apt-Pakete:

chronograf

#### Port

8888

### Python

#### Installierte pip3 Pakete

Folgende pip3 Pakete müssen mit pip3 install installiert werden:

* paho-mqtt
* influxdb

## Kommunikationsmatrix

Tabellarische Ansicht der Kommunikationswege in der IoT-Umgebung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Quelle | Protokoll | Ziel (Adresse) | Zielservice / Port | Zusatzinfos |
| M5 Stack Core | MQTT | Raspberry Pi (10.0.103.157) | Mosquitto / 1883 | Schreibt ob die Mikrowelle an ist auf microwaves/nr\* |
| MQTT-Broker | MQTT | Raspberry Pi | Python Script / 1883 |  |
| Chronograf | http | Raspberry Pi (10.0.103.26) | InfluxDB / 8086 | Chronograf bezieht Daten zum Darstellen von InfluxDB über HTTP. |
| Telegraf | http | Raspberry Pi (10.0.103.26) | InfluxDB / 8086 | Telegraf speichert Daten der aktivierten Telegraf-Plugins in InfluxDB. |
| Kapacitor | n/a | Raspberry Pi (10.0.103.26)  https://pushover.net | n/a | In Kapacitor sind noch keine Scripts für Alerts definiert. |
| microwave\*.py | MQTT | Raspberry Pi (10.0.103.26) | InfluxDB / 8086  Mosquitto / 1883 | Das Python-Script meldet sich dem Topic microwave/nr\* an und schreibt die erhaltenen Daten in InfluxDB. |

# Testing

## Testdrehbuch

Die Tests können der Reihe nach von 1-4 durchgeführt werden. Allenfalls benötigte Logindaten sind dieser Dokumentation zu entnehmen.

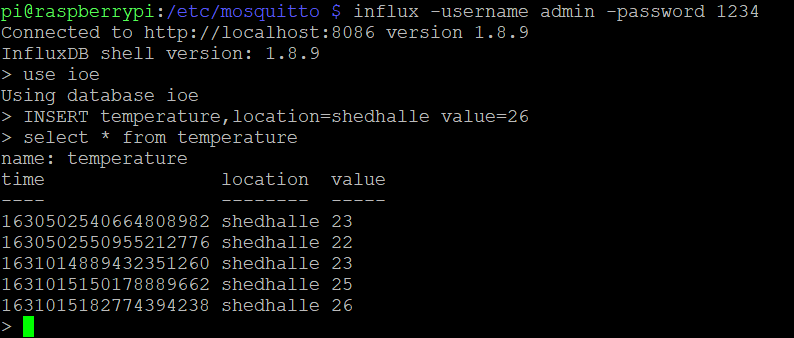
### Testfall 1 Lesen und Schreiben in InfluxDB mit Admin-Rechten

In diesem Testfall wird getestet, ob mit dem Admin-Benutzer in InfluxDB geschrieben und gelesen werden kann.

Auf Kommandozeile des Raspberry Pi folgende Befehle eingeben:

influx –username admin –password Influx.LoremIpsum  
  
> use ioe  
> INSERT temperature,location=shedhalle value=26  
> select \* from temperature

Die Ausgabe sollte den neu hinzugefügten Wert mit den bereits existierenden Werten anzeigen:



### Testfall 2 Nur Zugang zur InfluxDB telegraf mit Benutzer telegrafuser

In diesem Testfall wird getestet, ob der Benutzer telegrafuser nur Lese- und schreibrechte auf die Datenbank telegraf hat.

Auf Kommandozeile des Raspberry Pi folgende Befehle eingeben:

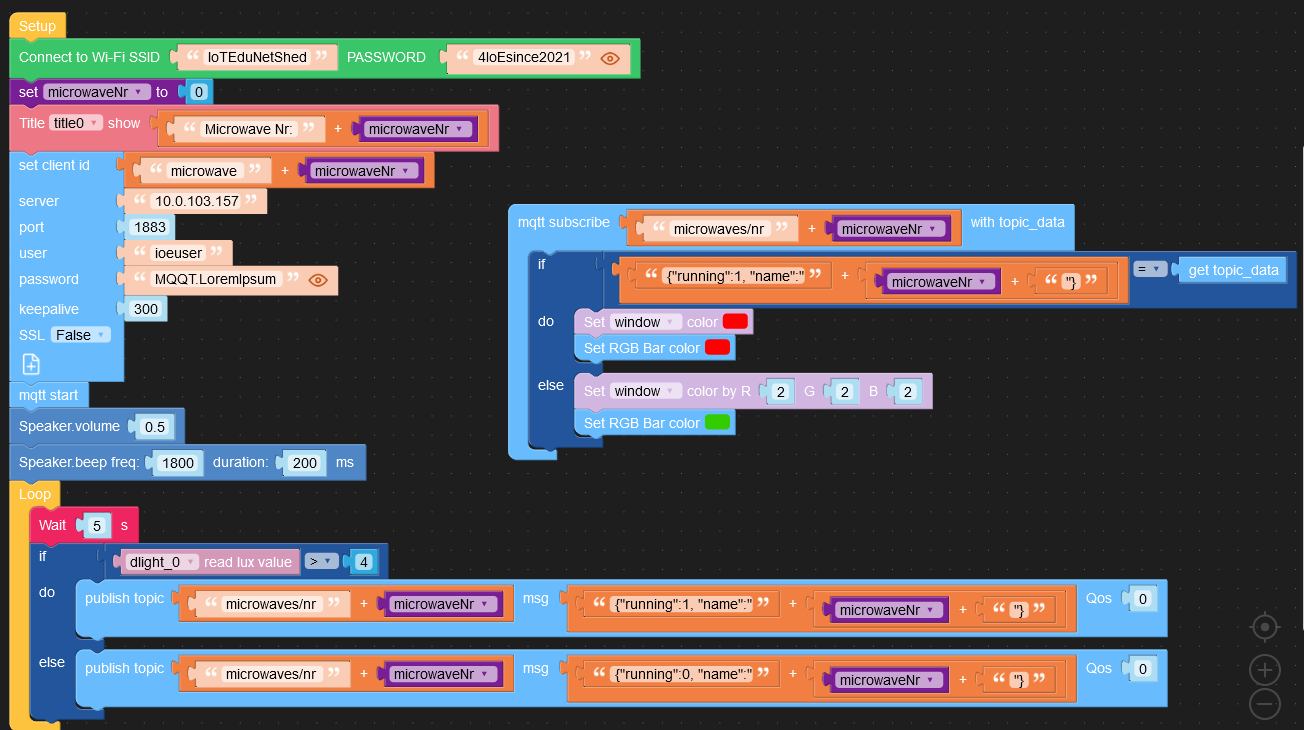
influx –username telegrafuser –password 1234  
  
> show databases

Die Ausgabe sollte nur die Datenbank telegraf anzeigen, die Datenbank ioe darf nicht aufgelistet sein:

### Testfall 3 Zusammenspiel Python-Script / Mosquitto / InfluxDB

In diesem Testfall schickt der M5Stack Mikrowellen daten an das Topic microwaves/nr0. Das Pyt hon-Script abonniert das Topic und schreibt die Mikrowellendaten in die InfluxDB ioe.

UI-Flow-Script „microwave.m5f“ (alternativ kann auch microwaveDummy.m5f benutzt werden. Button A Mikrowelle an, Button B Mikrowelle aus.) auf <https://flow.m5stack.com/> laden, microwaveNr auf 0 setzen und Unit DLIGHT hinzufügen. Anschliessend auf M5Stack ausführen.

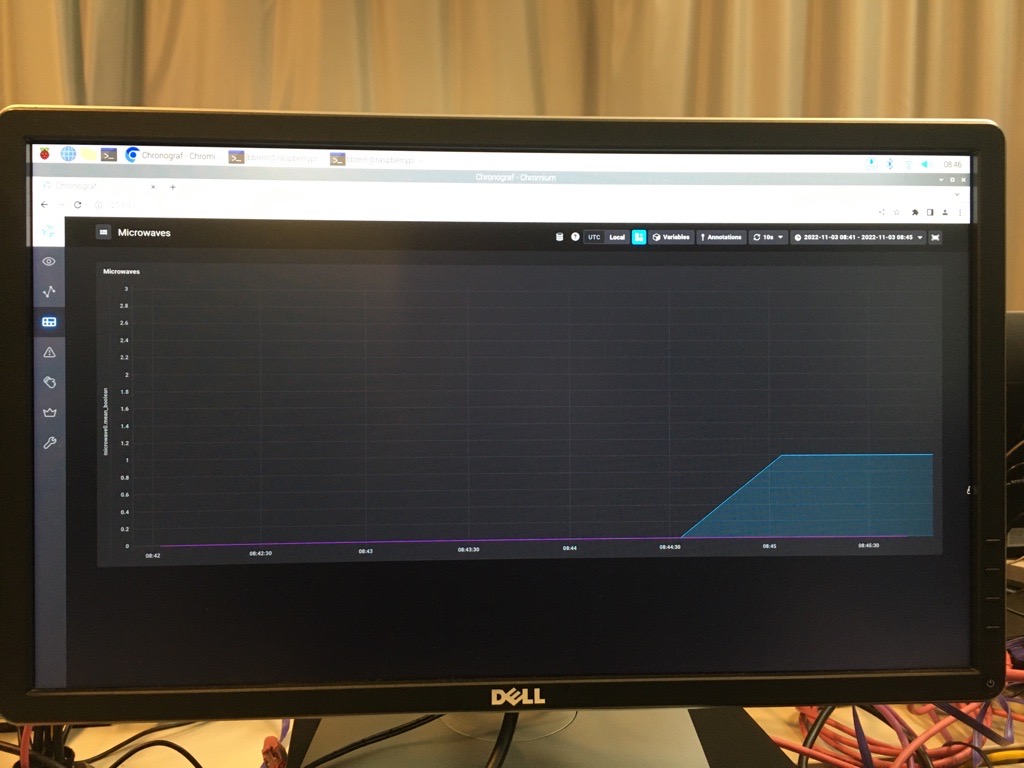


Python-Scripte microwave\*.py auf dem Raspberry ausführen mit Hilfe vom Bashscript sts.sh (/home/bbrein/pythonForTICK/sts.sh , pip3 Module paho-mqtt und influxdb müssen installiert sein).

Bei Chronograf (<http://10.0.103.26:8888>) das Dashboard „Microwaves“ öffnen. Mit der Hand den Lichtsensor blockieren oder ins Licht halten (Update alle 5s)

Es sollte ein Liniendiagramm angezeigt werden, welches die aktuelle Nutzung der Mikrowellen anzeigt.

Das Ein-/Ausschallten der Mikrowelle sollte klar ersichtlich sein.



### Testfall 4

In diesem Testfall erhält die React Seite Daten über die JSON Database vom Python Formatter (original vom Broker) aus. Die Seite soll die Korrekte Anzahl Mikrowellen Anzeigen und den Graph entsprechend aktualisieren.

Das React Script muss im korrekten Directory mit dem script «start» über npm ausgeführt werden. Dazu müssen der Broker (Raspberry), die M5Go’s mit den korrekten Scripts und das Formatter Script eingeschalten/gestartet sein.

Sobald das Formatter Script ein Objekt ins Terminal printen sollte die React-Seite sofort neu geladen werden. Im grossen weissen Text in der Mitte des Bildschirms sollte nun die richtige Anzahl ausgeschalteter Mikrowellen Anzeigen. Der Graph sollte innerhalb weniger Sekunden dieselbe Zahl anzeigen.

## Testprotokoll

Testfälle wurden durchgeführt von: Niklas Breit

### Testfall 1

Erfolgreich

### Testfall 2

Erfolgreich

### Testfall 3

Erfolgreich

### Testfall 4

Erfolgreich