

IoT Geiger Counter

Geigerzähler als Basis einer Internet-Wetterstation

Malte Pöggel

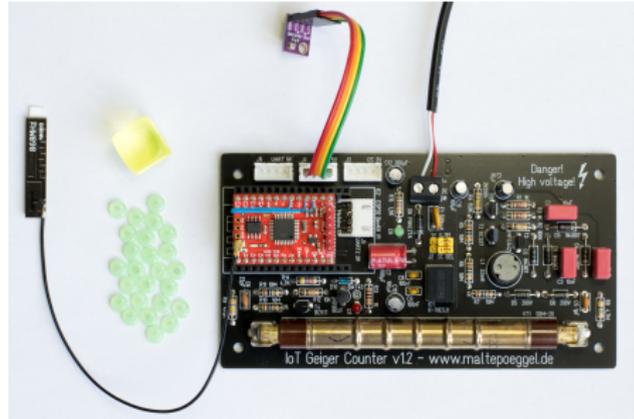
Maker Faire Hannover

19.08.2023



Inhalt

- ① Einführung
- ② Hardware
- ③ Firmware
- ④ Software
- ⑤ Abschluss



Einführung

Bestehendes System

- WiFi Weather Sensor (2016)
- Strahlungsmonitor ist eine spannende Ergänzung
- uRadmonitor Kit1
 - LAN Port unflexibel
 - Bug bei fehlendem Internet
- Nicht einheitlich
- Austausch durch Neuentwicklung?

Messnetzwerk des Bundesamtes für Strahlenschutz (BfS)

- 1700 Messstellen
- Erfasst Ortsdosisleistung
- Daten frei einsehbar
- Interessante Infos
- <https://odinfo.bfs.de/>

Messstellen in Deutschland



Abbildung: Screenshot Webseite

Sonde einer ODL Messeinrichtung



Abbildung: Ausgemusterte Sonde mit zwei Zählrohren

Gefahren am Beispiel des Goiânia-Unfalls 1987

- Gerät zur Strahlentherapie aus stillgelegter Klinik gestohlen
- Strahlenquelle zerlegt und unter Freunden verteilt
- 4 Todesfälle
- 249 Personen kontaminiert
- 500 Personen leiden auch 2017 noch an Spätfolgen



Abbildung: Strahlungskopf, IAEA (CC BY 2.0), <https://flic.kr/p/dYJ5QW>

¹Quelle: <https://de.wikipedia.org/wiki/Goi%C3%A2nia-Unfall>

Gefahren am Beispiel Wuppertal 2016

- Gerät zur Werkstoffprüfung
- Strahlenquelle steckte in Ausfahrtschlauch fest
- Schlauch aufgeschnitten
- Strahlerhülle beschädigt
- Anstieg der Dosisleistung unbemerkt - $80\mu\text{Sv/h}$
- Erhöhte Messwerte an 6 Stationen des DWD
- Aufklärung dauerte



Abbildung: Screenshot Webseite, abgerufen Juli 2023

²Quelle: <https://fragdenstaat.de/a/269416>

Gefahren am Beispiel Radonbelastung

- Radioaktives Gas
- Kommt im Erdreich in Gestein vor
- Durchdringt Mauerwerk
- Ansammlung in Gebäuden möglich
- Alphastrahler
- Radioaktive Zerfallsprodukte z.T. Beta- und Gammastrahler
- Risiko für Lungenkrebs



Abbildung: Uraninit,
Wikimedia User Foreade
(CC BY 4.0)

³Quelle: <https://www.bfs.de/DE/themen/ion/umwelt/radon/einfuehrung/einfuehrung.html>

Zusammenfassung

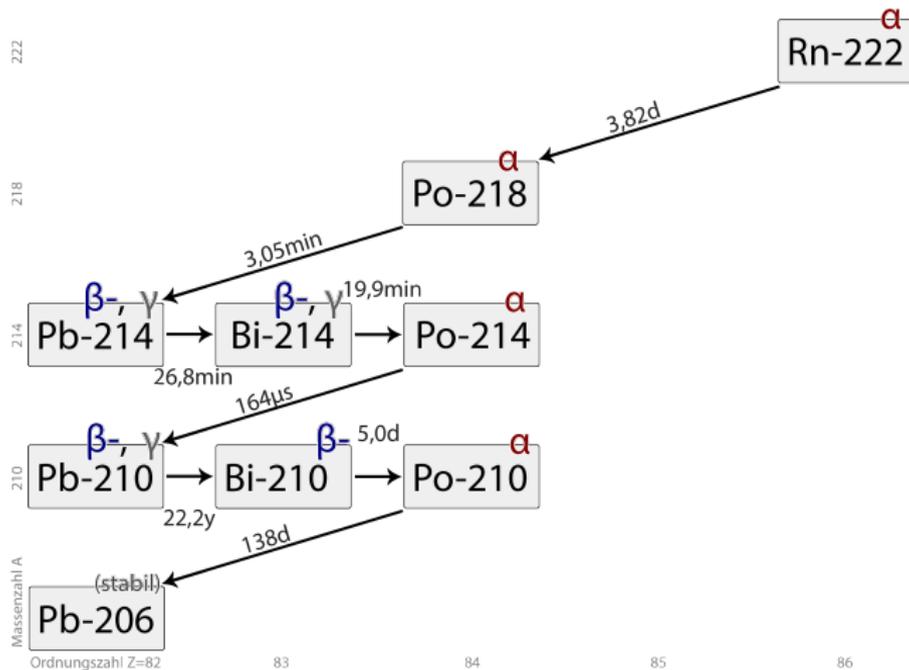
- Auch kleinere Vorfälle sind möglich
 - Flächendeckendes Monitoring findet bereits statt
 - Ein eigenes Gerät ersetzt kein wissenschaftliches Messgerät
- ABER
- Selbstbau fördert Interesse an Technik & Naturwissenschaften!

Hardware

Arten radioaktiver Strahlung

- Alphastrahlung
 - Teilchenstrahlung
 - Doppelt positiv geladene Heliumkerne
 - Leicht abzuschirmen: einige cm Luft, ein Blatt Papier
- Betastrahlung
 - Teilchenstrahlung
 - Elektronen (β^-) oder Positronen (β^+)
 - Abschirmbar mit einigen mm Aluminium
- Gammastrahlung
 - Elektromagnetische Strahlung
 - Photonen
 - Abschirmbar mit Blei (nicht vollständig)

Radioaktiver Zerfall



Entwicklung des Prototypen: Hochspannungserzeugung

- Stammt aus dem Netz
- Betrieb mit 5V
- Geringe Stromaufnahme
- Aufgebaut ca. 2012

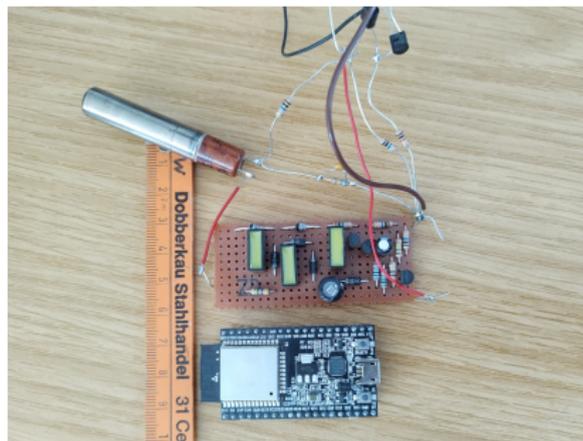


Abbildung: Musteraufbau des Hochspannungsteils

⁴Quelle: <https://www.mikrocontroller.net/topic/anschluss-und-verwendung-von-geiger-mueller-sbm-20>

Entwicklung des Prototypen: Signalauswertung

- Anschluss an Kathode
- Spannungsteiler und Darlington-Transistor
- LED zur Kontrolle
- Pegelwandlung auf 3.3V

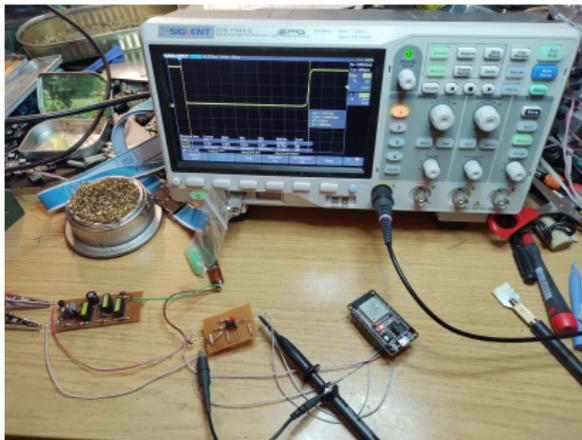


Abbildung: Überprüfung der Zählimpulse am Oszilloskop

Entwicklung des Prototypen: Impulszähler

- Geeignet sind

Mikrocontroller mit

Hardware Zähler
- ESP32: Pulse Counter

(PCNT)
- AVR: Timer-Hardware



Abbildung: Simulation von Impulsen mittels Signalgenerator

Entwicklung des Prototypen: CPU & Datenübertragung

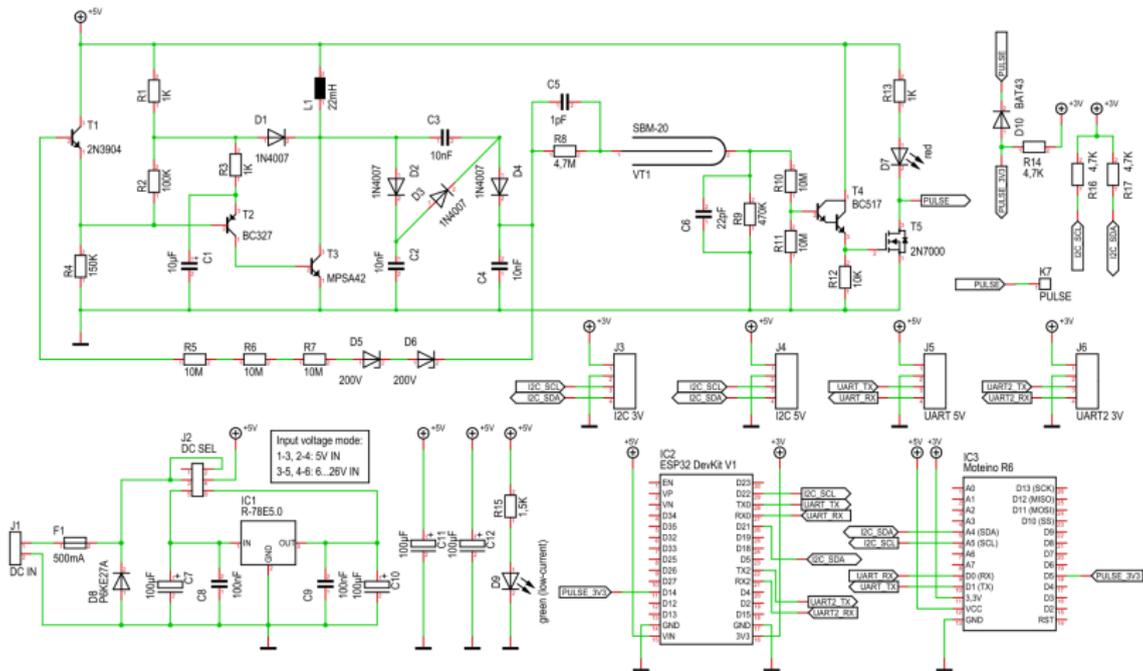
WiFi

- ESP32 DevKit V1 (15 Pin)
- 4MB Flash, 520kB SRAM
- 240MHz Taktfrequenz
- 2,4GHz WLAN
- Reichweite 50-100m
- Erfordert WLAN Router
- Keine Beschränkungen
- Stromaufnahme etwa 80-200mA

LoRa

- Moteino R6 (ATMega328)
- 32kB Flash, 2kB SRAM
- 8 oder 16MHz Taktfrequenz
- 868MHz LoRaWAN
- Reichweite 500-1000m
- Erfordert LoRa Gateway
- Frequenznutzung beschränkt (Duty Cycle)
- Stromaufnahme $\leq 10\text{mA}$

Schaltplan



Leiterplatte in der Entwicklung

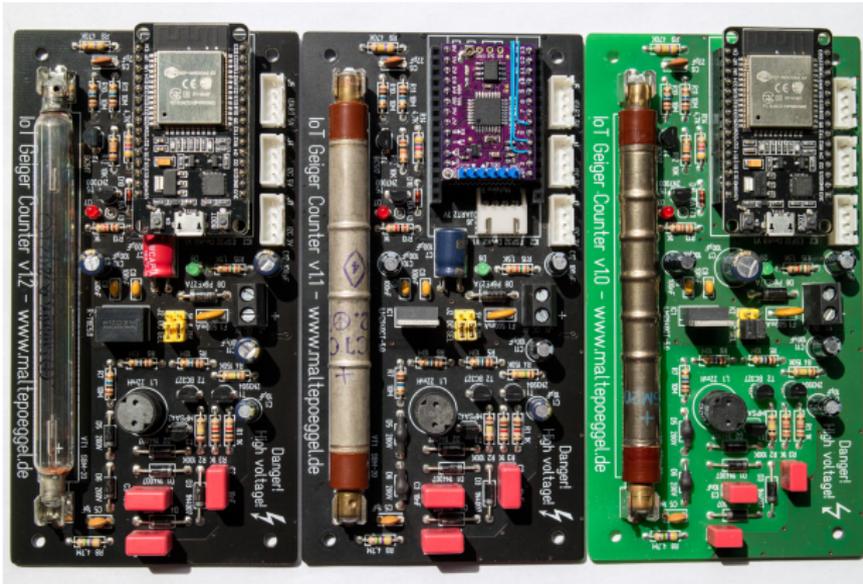


Abbildung: Verschiedene Platinen-Revisionen

Leiterplatte im Detail

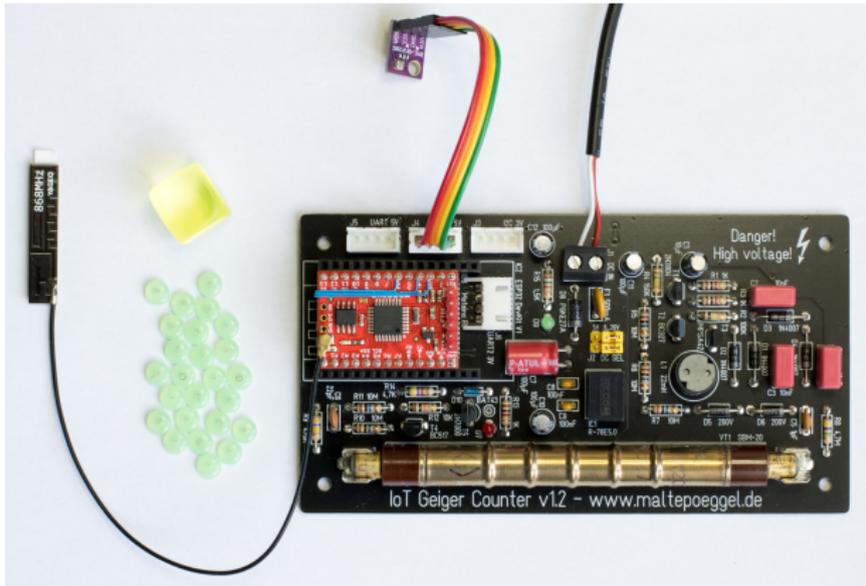


Abbildung: v1.2 mit Moteino, BME280 und Uranglasperlen

Gehäuse



Abbildung: Einbau im IP65 Gehäuse mit Kabelverschraubung, Mehrfachdurchführung und Druckausgleichsmembran

Gehäuse



Abbildung: Im Außenbereich verhindert ein Druckausgleichselement das Eindringen von Feuchtigkeit (dieses war hier nicht vorhanden)

⁵Infos: <https://www.bopla.de/technische-daten/technische-informationen/druckausgleichselemente>

Firmware

ESPHome Firmware - Einführung

- ESP32 mit WiFi
- Firmware zur Heimautomatisierung
- Einfache Einbindung von Sensoren
- Viele Schnittstellen zur Ausgabe
- Direkte Integration in HomeAssistant
- Konfiguration über YAML Files
- Keine Programmierkenntnisse nötig
- PlatformIO Buildsystem

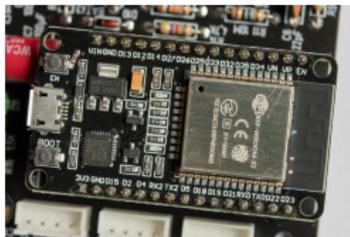
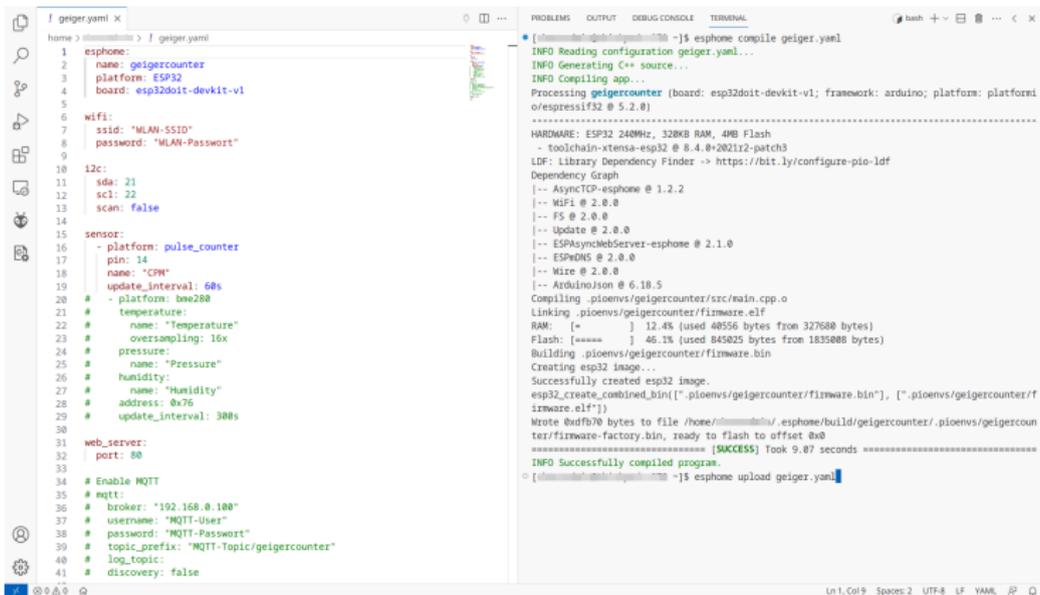


Abbildung: ESP32
DevKit V1

⁶Link: <https://esphome.io/>

ESPHome Firmware - Konfiguration



```
1 esphome:
2   name: geigercounter
3   platform: ESP32
4   board: esp32doit-devkit-v1
5
6 wifi:
7   ssid: "MLAN-SSID"
8   password: "MLAN-Passwort"
9
10 12c:
11   sda: 21
12   scl: 22
13   scan: false
14
15 sensor:
16   - platform: pulse_counter
17     pin: 14
18     name: "CPM"
19     update_interval: 60s
20   # - platform: bme280
21     # temperature:
22     #   name: "Temperature"
23     # oversampling: 16x
24     # pressure:
25     #   name: "Pressure"
26     # humidity:
27     #   name: "Humidity"
28     # address: 0x76
29     # update_interval: 300s
30
31 web_server:
32   port: 80
33
34 # Enable MQTT
35 # mqtt:
36 #   broker: "192.168.0.100"
37 #   username: "MQTT-User"
38 #   password: "MQTT-Passwort"
39 #   topic_prefix: "MQTT-Topic/geigercounter"
40 #   log_topic:
41 #   discovery: false
```

```
bash -] esphome compile geiger.yaml
INFO Reading configuration geiger.yaml...
INFO Generating C++ source...
INFO Compiling app...
Processing geigercounter (board: esp32doit-devkit-v1; framework: arduino; platform: platform/
ofrespressif32 @ 5.2.0)
-----
HARDWARE: ESP32 240MHz, 320KB RAM, 4MB Flash
- toolchain-xtensa-esp32 @ 8.4.0:202112-patch3
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf
Dependency Graph
|-- AsyncTCP-esphome @ 1.2.2
|-- WiFi @ 2.0.0
|-- FS @ 2.0.0
|-- Update @ 2.0.0
|-- ESPAsyncWebServer-esphome @ 2.1.0
|-- ESPmDNS @ 2.0.0
|-- Wire @ 2.0.0
|-- ArduinoJson @ 6.18.5
Compiling pioenvs/geigercounter/src/main.cpp.o
Linking .pioenvs/geigercounter/firmware.elf
RAM: [=====] 12.4k (used 48556 bytes from 327680 bytes)
Flash: [=====] 46.1k (used 845025 bytes from 1835008 bytes)
Building .pioenvs/geigercounter/firmware.bin
Creating esp32 image...
Successfully created esp32 image.
esp32_create_combined_bin([".pioenvs/geigercounter/firmware.bin"], [".pioenvs/geigercounter/f
irmware.elf"])
Wrote 0x0dfb70 bytes to file /home/maltepoggel/.esphome/build/geigercounter/.pioenvs/geigercount
er/firmware-factory.bin, ready to flash to offset 0x0
===== [SUCCESS] Took 9.07 seconds =====
INFO Successfully compiled program.
bash -] esphome upload geiger.yaml
```

Abbildung: Erstellung des YAML Codes und Kompilieren der Firmware

ESPHome Firmware - Webinterface



geigercounter

Name	State	Actions
CPM	27.00 pulses/min	
Humidity	35.2 %	
Pressure	1026.1 hPa	
Radiation	0.154 uSv/h	
Temperature	19.1 °C	

Scheme  

OTA Update

```
Time      level Tag      Message
18:29:03 [D] [sensor:127] 'Temperature': Sending state 19.11000 °C with 1 decimal
18:29:03 [D] [sensor:127] 'Pressure': Sending state 1026.14795 hPa with 1 decimal
18:29:03 [D] [sensor:127] 'Humidity': Sending state 35.28605 % with 1 decimal
18:29:07 [D] [pulse_counter:174] 'CPM': Retrieved counter: 30.00 pulses/min
18:29:07 [D] [sensor:127] 'CPM': Sending state 30.00000 pulses/min with 2 decimal
18:29:07 [D] [sensor:127] 'Radiation': Sending state 0.17101 uSv/h with 3 decimal
18:30:06 [D] [pulse_counter:174] 'CPM': Retrieved counter: 15.00 pulses/min
18:30:06 [D] [sensor:127] 'CPM': Sending state 15.00000 pulses/min with 2 decimal
18:30:06 [D] [sensor:127] 'Radiation': Sending state 0.08550 uSv/h with 3 decimal
18:31:07 [D] [pulse_counter:174] 'CPM': Retrieved counter: 17.00 pulses/min
18:31:07 [D] [sensor:127] 'CPM': Sending state 17.00000 pulses/min with 2 decimal
18:31:07 [D] [sensor:127] 'Radiation': Sending state 0.09690 uSv/h with 3 decimal
18:32:07 [D] [pulse_counter:174] 'CPM': Retrieved counter: 23.00 pulses/min
18:32:07 [D] [sensor:127] 'CPM': Sending state 23.00000 pulses/min with 2 decimal
18:32:07 [D] [sensor:127] 'Radiation': Sending state 0.13111 uSv/h with 3 decimal
18:33:06 [D] [pulse_counter:174] 'CPM': Retrieved counter: 27.00 pulses/min
18:33:06 [D] [sensor:127] 'CPM': Sending state 27.00000 pulses/min with 2 decimal
18:33:06 [D] [sensor:127] 'Radiation': Sending state 0.15391 uSv/h with 3 decimal
```

Abbildung: Weboberfläche im Browser

LoRaWAN Firmware - Einführung

- Moteino R6 (ATMega328)
- Selbst entwickelte Firmware
- Basiert auf Arduino Bibliotheken
- MCCI Arduino LMIC für LoRaWAN
- OTAA und ABP Modus
- Getestet mit TTN und Chirpstack
- BME280 Sensor optional
- PlatformIO Buildsystem



Abbildung: Moteino

⁷Link: <https://github.com/MalteP/iotgeiger-lora>

LoRaWAN Firmware - Konfiguration

The image shows two side-by-side screenshots. The left screenshot displays the 'Overview' page for a device named 'iotgeiger-demo' in the 'eu1.cloud.thethings.network' console. The 'General information' section includes: End device ID: iotgeiger-demo; Frequency plan: Europe 863-870 MHz (SF9 for 8K2 - recommended); LoRaWAN version: LoRaWAN Specification 1.0.2; Regional Parameters version: RP005 Regional Parameters 1.0.2; Created at: Jul 24, 2023 17:19:14. The 'Activation information' section shows: AppEUI: 0xCA, 0xFE, 0x20, 0x67, 0x2E, 0x...; DevEUI: 0xCA, 0xFC, 0x05, 0x00, 0x76, 0x...; AppKey: 0xC3, 0x31, 0xCA, 0x14, 0xB, 0xB... The 'Session information' section states: This device has not joined the network yet. The 'MAC data' section has a 'Download MAC data' button.

The right screenshot shows the Visual Studio Code editor with the 'config.h' file open. The code defines various LoRaWAN parameters for the 'noteino16mhz' board. The code is as follows:

```
src > C:\config.h > cat APPKEY
1 #pragma once
2
3 // Only one of these settings must be defined
4 #define USE_OTAA
5 // #define USE_ABP
6
7 #ifdef USE_OTAA
8
9 // LoRaWAN OTAA AppEUI, application EUI in little endian format (LSB)
10 static const u1_t PROGMEM APPEUI[8] = {0xCA, 0xFE, 0x20, 0x67, 0x2E, 0xC6, 0x07, 0x2E};
11
12 // LoRaWAN OTAA DevEUI, device EUI in little endian format (LSB)
13 static const u1_t PROGMEM DEVEUI[8] = {0xCA, 0xFC, 0x05, 0x00, 0x7E, 0x05, 0xB3, 0x70};
14
15 // LoRaWAN OTAA AppKey, application key in big endian format (MSB)
16 static const u1_t PROGMEM APPKEY[16] = {0xC3, 0x31, 0xCA, 0x14, 0xB2, 0xE6, 0x44, 0x10, 0x20, 0x19, 0x
17
18 #endif
19
20 #ifdef USE_ABP
21
22
```

The terminal window below the code shows the compilation output:

```
avrduide: safedone: Fuses OK (E:00, H:00, L:00)
avrduide done. Thank you.
===== [SUCCESS] Took 32.32 seconds =====
Environment   Status   Duration
-----
noteino16mhz SUCCESS 00:00:32.323
-----
1 succeeded in 00:00:32.323 -----
Terminal will be closed automatically. To prevent this from happening, please see first tab to the right.
PlatformIO: Upload
```

Abbildung: Einrichtung der Parameter und Kompilieren der Firmware

LoRaWAN Firmware - LoRaWAN Konsole

The screenshot displays the TTN LoRaWAN console interface. The top navigation bar shows 'Overview', 'Applications', 'Gateways', and 'Organizations'. The 'Applications' section is active, showing 'iotgeiger-demo'. The left sidebar contains navigation options like 'Overview', 'End devices', 'Live data', 'Payload formatters', 'Integrations', 'Collaborators', 'API keys', and 'General settings'. The main content area shows a list of data messages with columns for Time, Type, and Data preview. The data preview shows decoded JSON payloads with fields for cpm, humidity, pressure, temperature, and uvv_h.

Time	Type	Data preview
18:07:29	Forward uplink data message	DevAddr: 26 08 20 79 Payload: { cpm: 28, humidity: 65.95, pressure: 996.69, temperature: 22.33, uvv_h: 0.114 }
18:07:29	Successfully processed data message	DevAddr: 26 08 20 79
18:02:29	Forward uplink data message	DevAddr: 26 08 20 79 Payload: { cpm: 21, humidity: 66.375, pressure: 996.64, temperature: 22.33, uvv_h: 0.12 }
18:02:29	Successfully processed data message	DevAddr: 26 08 20 79
17:57:29	Forward uplink data message	DevAddr: 26 08 20 79 Payload: { cpm: 19, humidity: 63.546, pressure: 996.36, temperature: 22.33, uvv_h: 0.108 }
17:57:29	Successfully processed data message	DevAddr: 26 08 20 79
17:52:29	Forward uplink data message	DevAddr: 26 08 20 79 Payload: { cpm: 20, humidity: 63.076, pressure: 996.36, temperature: 22.33, uvv_h: 0.114 }
17:52:29	Successfully processed data message	DevAddr: 26 08 20 79
17:47:29	Forward uplink data message	DevAddr: 26 08 20 79 Payload: { cpm: 24, humidity: 63.647, pressure: 996.21, temperature: 22.33, uvv_h: 0.137 }
17:47:29	Successfully processed data message	DevAddr: 26 08 20 79
17:42:29	Forward uplink data message	DevAddr: 26 08 20 79 Payload: { cpm: 22, humidity: 63.291, pressure: 996.21, temperature: 22.33, uvv_h: 0.125 }
17:42:29	Successfully processed data message	DevAddr: 26 08 20 79
17:37:30	Forward uplink data message	DevAddr: 26 08 20 79 Payload: { cpm: 21, humidity: 63.308, pressure: 996.1, temperature: 22.23, uvv_h: 0.12 }
17:37:30	Successfully processed data message	DevAddr: 26 08 20 79
17:32:30	Forward uplink data message	DevAddr: 26 08 20 79 Payload: { cpm: 21, humidity: 62.894, pressure: 996.5, temperature: 22.31, uvv_h: 0.12 }
17:32:30	Successfully processed data message	DevAddr: 26 08 20 79

Abbildung: Decodierte Nutzdaten in der TTN Konsole

Software

Benötigte Softwarekomponenten

- Übertragung
 - Gemeinsame Schnittstelle
- Erfassung
 - Sammeln & an DB senden
- Speicherung
 - Datenbanksystem
- Visualisierung
 - Diagramm erzeugen
- Installation auf Server oder VM



Abbildung: Grafana Dashboard

Datenübertragung

- Es wurde das MQTT Protokoll gewählt
- Message Queue Telemetry Transport
- Sowohl mit WiFi als auch LoRaWAN Variante kompatibel
- Client-Server Protokoll für M2M-Kommunikation
- Seit 2013 standardisiert für das Internet der Dinge

Datenübertragung

- Daten können unter Themen (Topic) veröffentlicht werden
- Clients können dieses Thema abonnieren (Subscribe)
- Daten werden den jeweiligen Abonnenten zugestellt (Publish)
- WiFi mit ESPHome Firmware
 - Zusätzlicher MQTT Server nötig
- LoRaWAN mit eigener Firmware
 - LoRa Stack stellt MQTT Server bereit

Datenübertragung

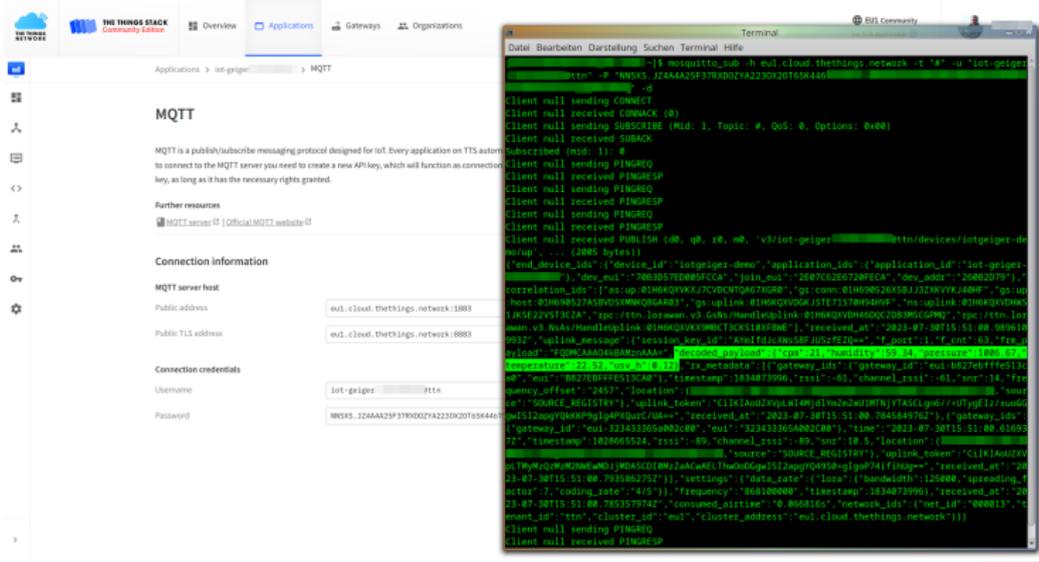
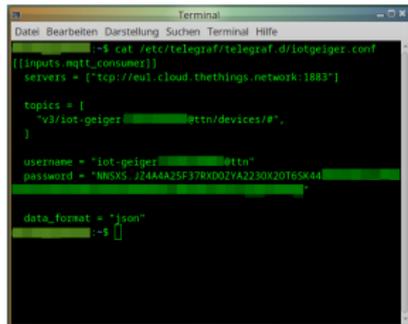


Abbildung: Verbindungstest zum The Things Network MQTT Server

Datenerfassung

- Es wurde Telegraf gewählt
- Tool gehört zu InfluxDB
- Sammelt Daten ein
- Kann u.a. als MQTT Client dienen
- Schreibt in die Datenbank



```
Terminal
Datei Bearbeiten Darstellung Suchen Terminal Hilfe
~$ cat /etc/telegraf/telegraf.d/iotgeiger.conf
[[inputs.mqtt_consumer]]
  servers = ["tcp://eu1.cloud.thethings.network:1883"]

  topics = [
    "vs/iot-geiger/#####@ttn/devices/#",
  ]

  username = "iot-geiger"
  password = "MNSX5_JZ4AA25F37RXD02YA2230X20T6SK44"

  data_format = "json"
  ->
```

Abbildung:
Beispielkonfiguration der
MQTT Verbindung

⁸Link: <https://www.influxdata.com/time-series-platform/telegraf/>

Datenspeicherung

- Es wurde InfluxDB gewählt
- Datenbank speziell für Zeitreihen
- Kein festes Schema
- Hohe Flexibilität
- Key-Value Datenbank
- Organisation der Datensätze nach Zeitstempel
- Gruppierung mittels Measurements und Tags

⁹Link: <https://www.influxdata.com/>

Datenvisualisierung

- Es wurde Grafana gewählt
- Grafische Darstellung von Daten
- Verschiedene Datenquellen
- Frontend browserbasiert
- Frei konfigurierbare Dashboards
- Erweiterbar über Plugins

¹⁰Link: <https://grafana.com/>

Datenvisualisierung

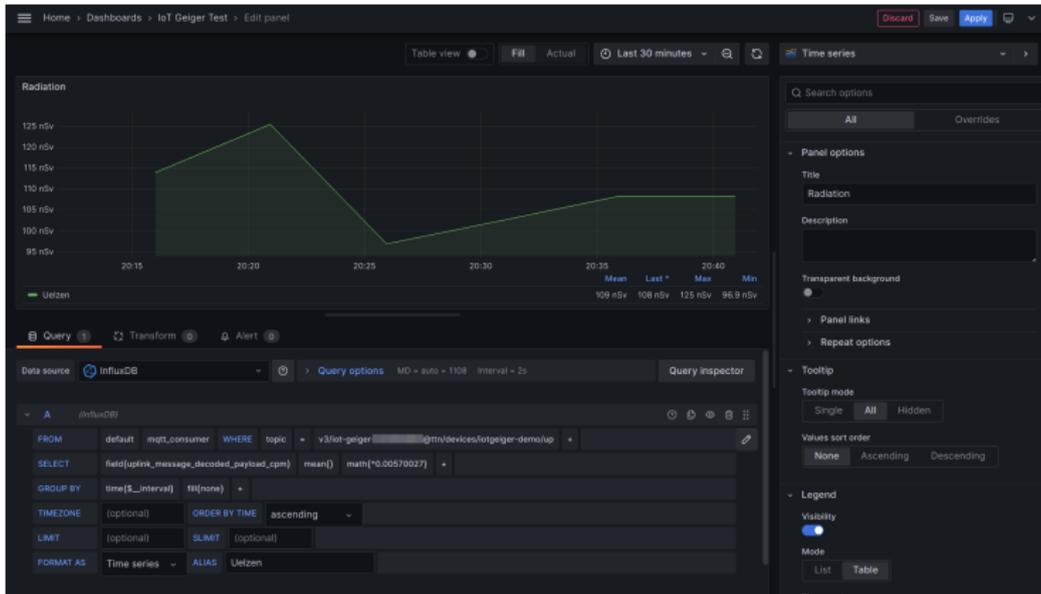


Abbildung: Konfiguration eines Panels in Grafana

Datenvisualisierung

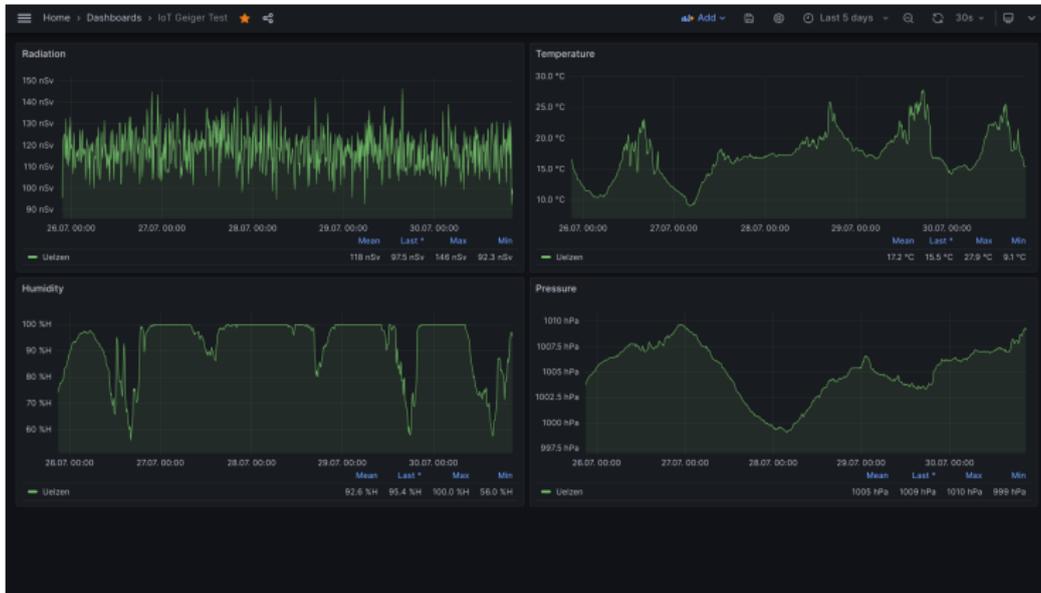


Abbildung: Ansicht des Grafana Dashboards

Zusammenfassung

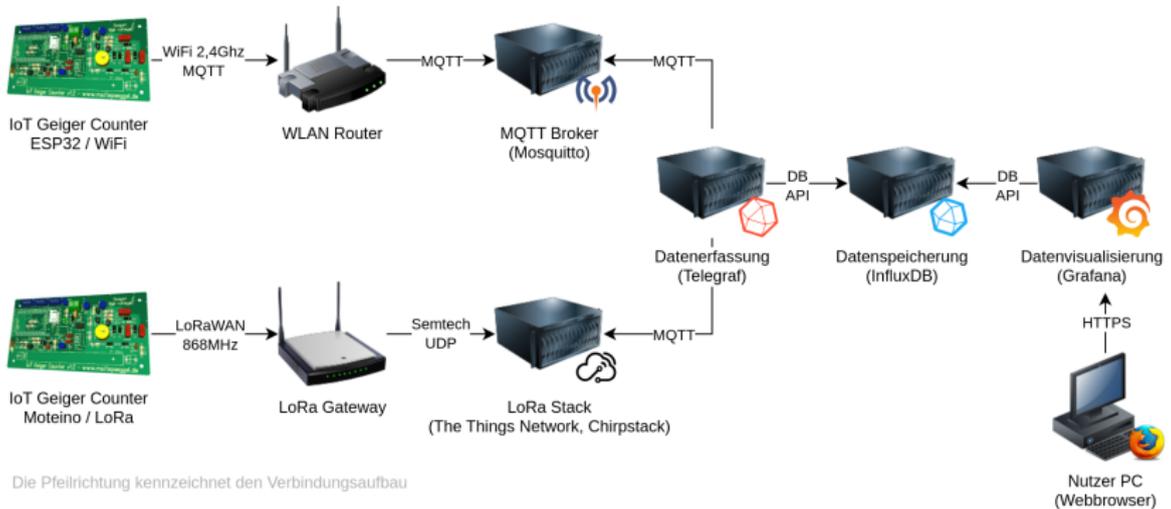


Abbildung: Übersicht der Hard- und Software Komponenten

Fragen?

Weitere Infos & Download der Folien:



<https://www.maltepoeggel.de/?site=iotgeiger>