# MultiGeiger

***Release V1.15.0***

**Mar 21, 2021**

# Contents

CHAPTER 1

# Overview

## 1.1 What is MultiGeiger?

The MultiGeiger is a radioactivity measurement device.

### 1.1.1 Main features

**multiple use cases supported**

- fixed location stations (data transfer via network)
- mobile operations (using the OLED display and speaker)
- (more are planned)

**multiple 400V Geiger-Mueller tubes supported**

- big, sensitive tubes, like the Si21g or Si22g (for measurement stations)
- smaller, less sensitive tubes, like the SBM21 (good for finding hot spots)

**Hardware**

- low parts count, simple to assemble
- inexpensive, but still good parts
- using ESP32, a modern and fast 32bit micro controller
- with WiFi (WPA2) or with WiFi + LoRA
- with OLED display

**Firmware**

- implemented in C
- using the popular arduino API

- over-the-air firmware updates

**Network and Community**

- this is an Ecocurious citizen science project
- web-based geo map for publishing measurements
- web-based archive of historic radiation data

**Free and Open**

- GPL v3 licensed
- open development and issue tracking on GitHub

## About

The **MultiGeiger** is a project of Ecocurious, your community for environment, nature and technology, see https://ecocurious.de/projekte/multigeiger/.

The goal is to establish a citizen measurement network for radioactivity in Germany.

The **MultiGeiger** hardware and software was designed by Jürgen Böhringer (http://www.boehri.de).

Reinhard/rexfue has further developed the software, the board, and is taking care of the integration of the sensors into our map https://ecocurious.de/multigeiger-karte/.

We have started the first workshops where you can assemble the components and the case with our support.

**Workshop dates can be found here:**

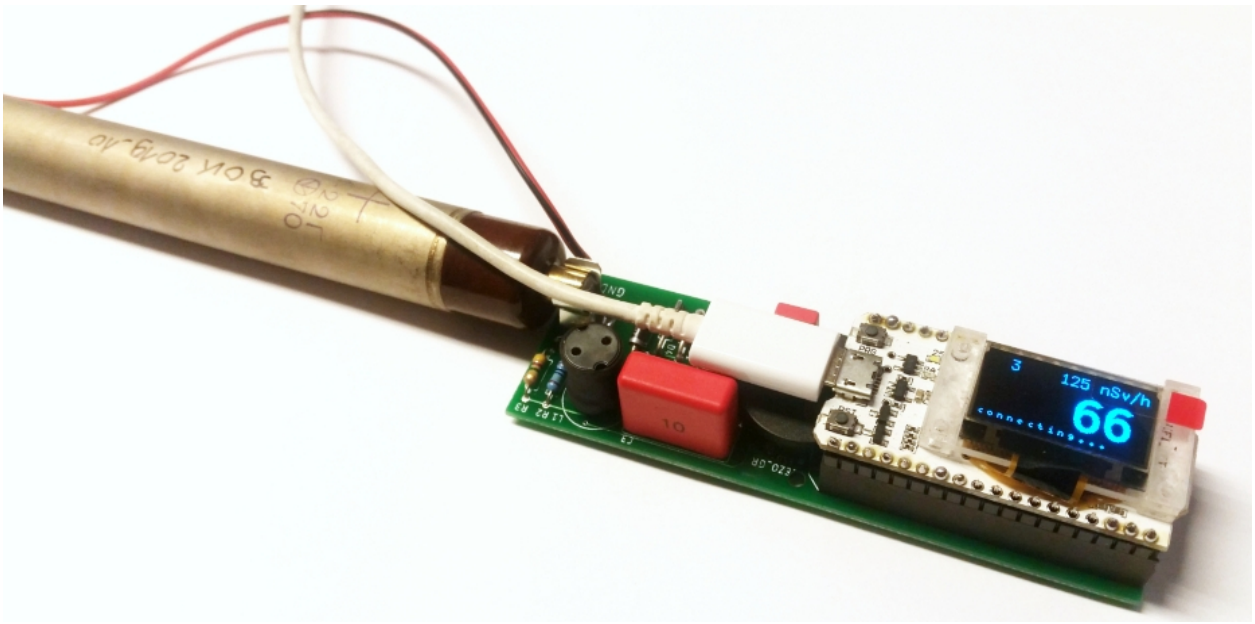
- https://www.meetup.com/de-DE/Ecocurious-deine-Umwelt-Natur-und-Technik-Community/
- https://ecocurious.de/events/

Does this sound interesting to you? Then join us, we cordially invite you!

## Assembly

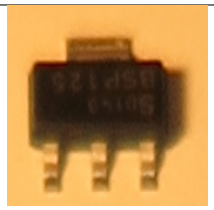
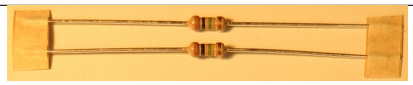
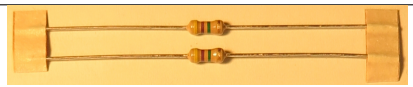
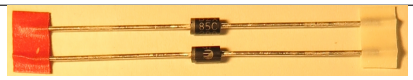
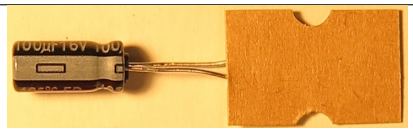
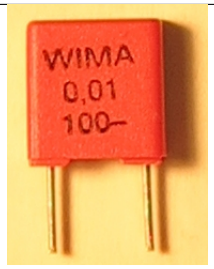
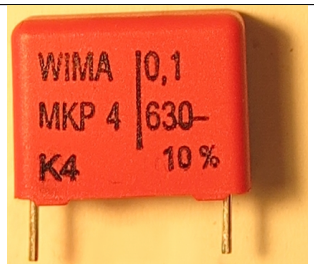
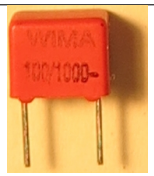
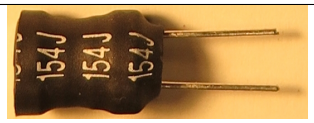
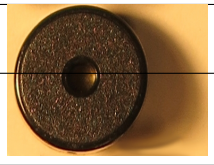
## Ergebnis

Das Multigeigerprojekt ist eine Idee der Ecocurious-Initiative. Das dazugehörige Open-Source Multigeigermessgerät zum Messen radioaktiver Gammastrahlung wird von der Community entwickelt. Die Echtzeitdaten der stationären Geräte werden per LoRaWAN oder WiFi/WLAN an einen Server gefunkt und auf einer Karte visualisiert. So entsteht ein Radioaktivitätsmessnetz in Bürgerhand, das wir hiermit aufbauen wollen. Herzliche Einladung zum Mitmachen!




### Bauteile

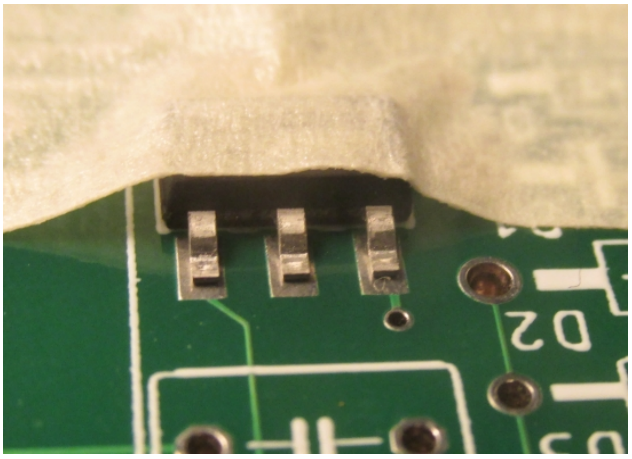
Elektronikbauteile aus (s. Stückliste) zunächst sortieren und auf Vollständigkeit prüfen:

| ID(s) | # | Bild | Typ |
|---|---|---|---|
| T1 | 1 |  | N-MOS-Transistor BSP125 |
| R1 | 1 |  | 10K Widerstand braun-schwarz-orange-gold |
| R2 R5 | 2 |  | 1,0M Widerstand braun-schwarz-grün-gold |
| R3 R4 | 2 |  | 4,7M Widerstand gelb-violett-grün-gold |
| D1 D2 | 2 |  | Z-Diode ZY 200 |
| D3 | 1 |  | Z-Diode BZX55C3V3 |
| D4 | 1 |  | Gleichrichterdiode BYV 26E |
| D6 | 1 |  | Gleichrichterdiode GP10Y |
| C1 | 1 |  | ELKO Elektrolytkondensator 100 uF, 16 V |
| C2 | 1 |  | Folienkondensator 10nF, 100V 0.01 100- |
| C3 | 1 |  | Folienkondensator 100nF, 630V 0.1 630- |
| C4 | 1 |  | Folienkondensator 100pF, 1000V 100/1000- |
| L1 | 1 |  | Spule 150uH 154J |
| Pz1 | 1 |  | Piezo-Lautsprecher |
| | |  | |

### Lötarbeiten

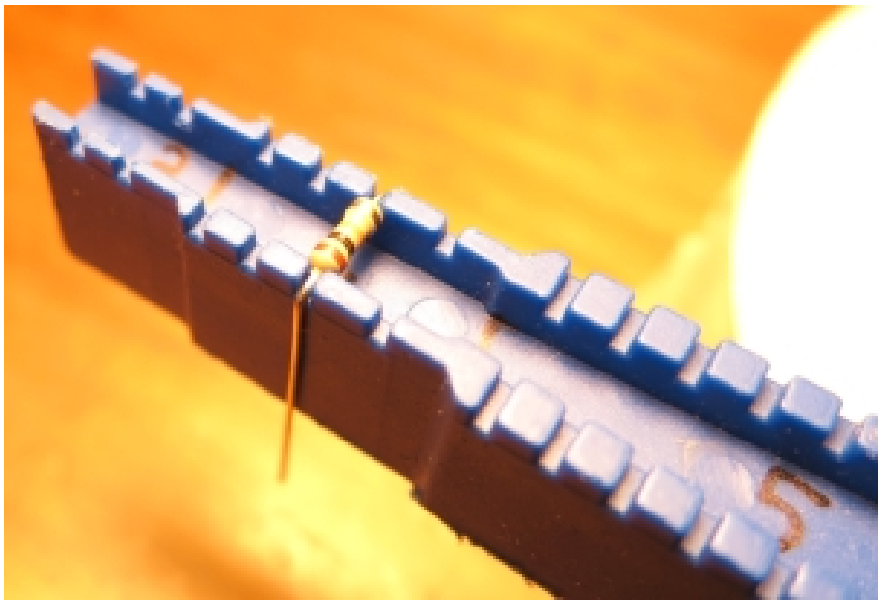
#### Feldeffekt-Transistor T1

Als erstes Bauteil wird das einzige Bauteil im SMD-Bauform montiert. Mit Klebeband (Kreppband hat sich bewährt) wird er so auf die Platine geklebt, so dass die drei Pins sichtbar sind. Dann werden die vorderen Anschlüsse verlötet, das Klebeband vorsichtig entfernt und der hintere Anschluss verlötet.

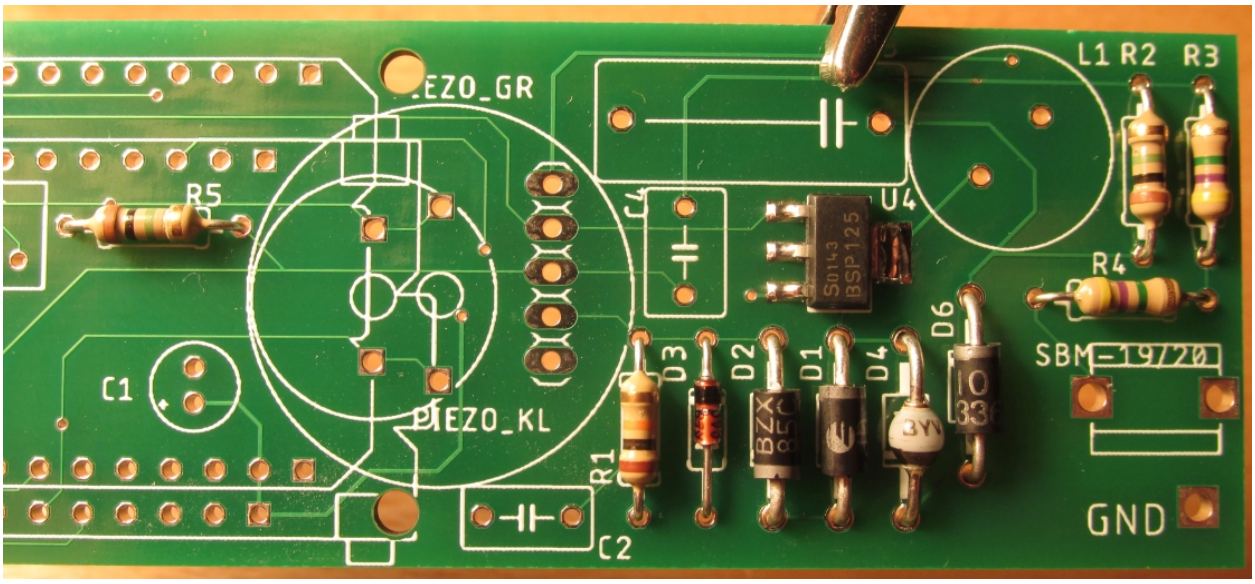


#### Widerstände und Dioden

Die Widerstände und Dioden werden alle in einer Abbiegevorrichtung (oder entsprechend mit der Hand oder einer kleinen Zange) gebogen. Die Breite ist immer vier Einheiten – sprich ca. 10 mm.

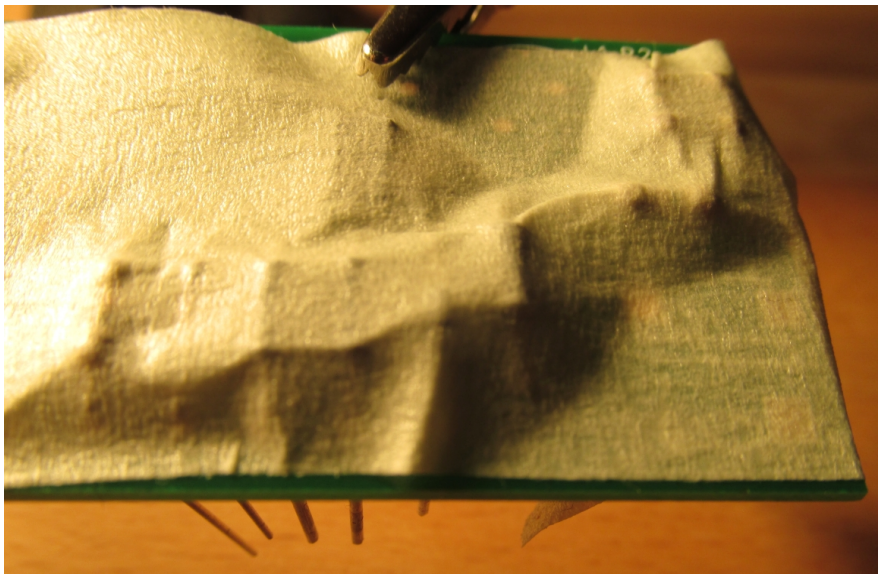


Anschließend wird die Platine mit den Widerständen und Dioden bestückt. Bei den Dioden unbedingt auf die Polarität achten! Die Kathode (Minus-Pol) ist mit einem Strich auf Platine und Bauteil markiert.




Vor dem Löten der Unterseite werden die Bauteile auf der Oberseite mit Klebeband fixiert, oder die Beinchen der Bauteile durch leichtes Auseinanderbiegen verklemmt. Dann Bauteile anlöten, Klebeband entfernen und überschüssigen Draht abschneiden.

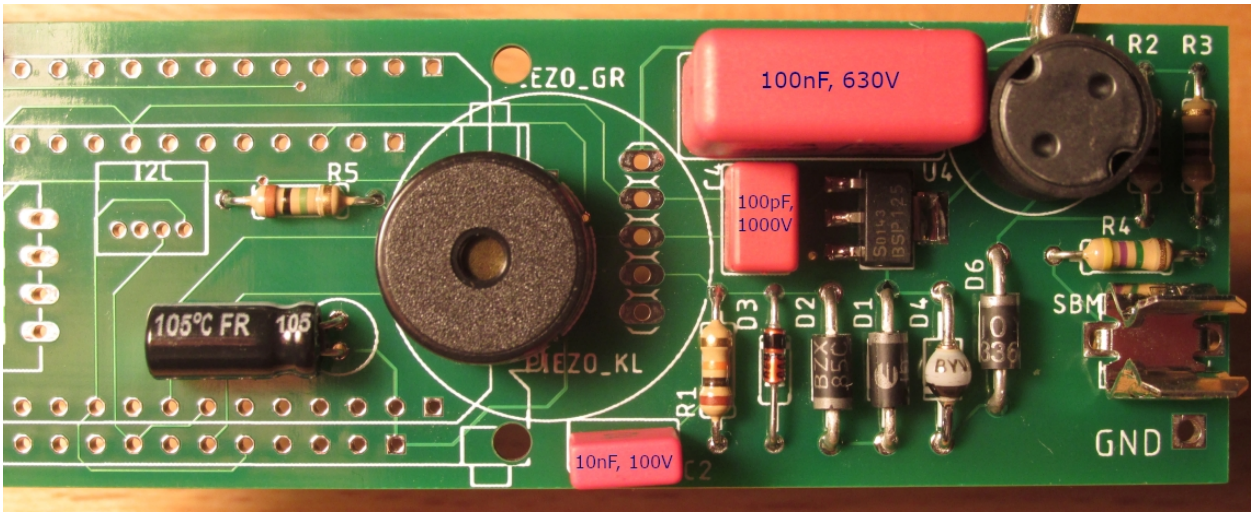
**Note:** Verletzungsgefahr: Beim Abschneiden darauf achten, dass der davonfliegende Draht niemanden verletzten kann. Möglichst festhalten beim Abschneiden!



### Kondensatoren und restliche Bauteile

Der Elektrolytkondensator (Elko) wird vorher mit einer Flachzange angewinkelt. Beim Elko unbedingt auf die Polung achten: Auf dem Elko ist der „–“-Pol markiert, auf der Platine der „+“-Pol. Die Beschriftungen müssen sich gegenüber liegen. Der Zählrohrhalter muss so herum eingesetzt werden, dass das Zählrohr nach außen steht kann – am besten testen! Dann die Bauteile mit Klebeband fixieren, anlöten und den überschüssigen Draht abschneiden.

Kondensatoren, Zählrohrhalter, Spule, Piezo-Lautsprecher sind bestückt:



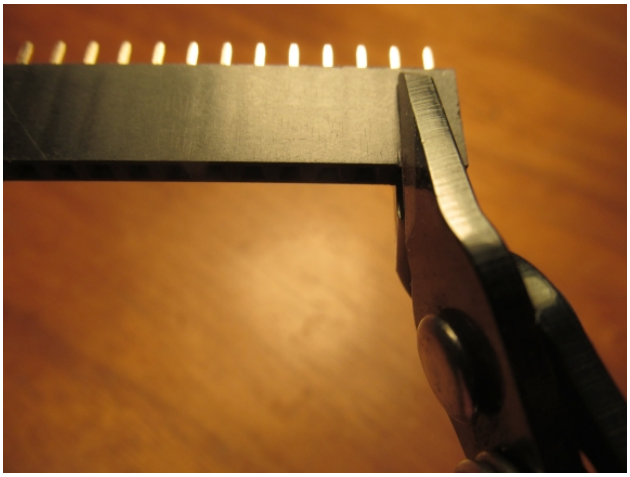

### Buchsenleisten und Mikrocontroller-Modul

Die Buchsenleiste muss vorher evtl. noch gekürzt werden.

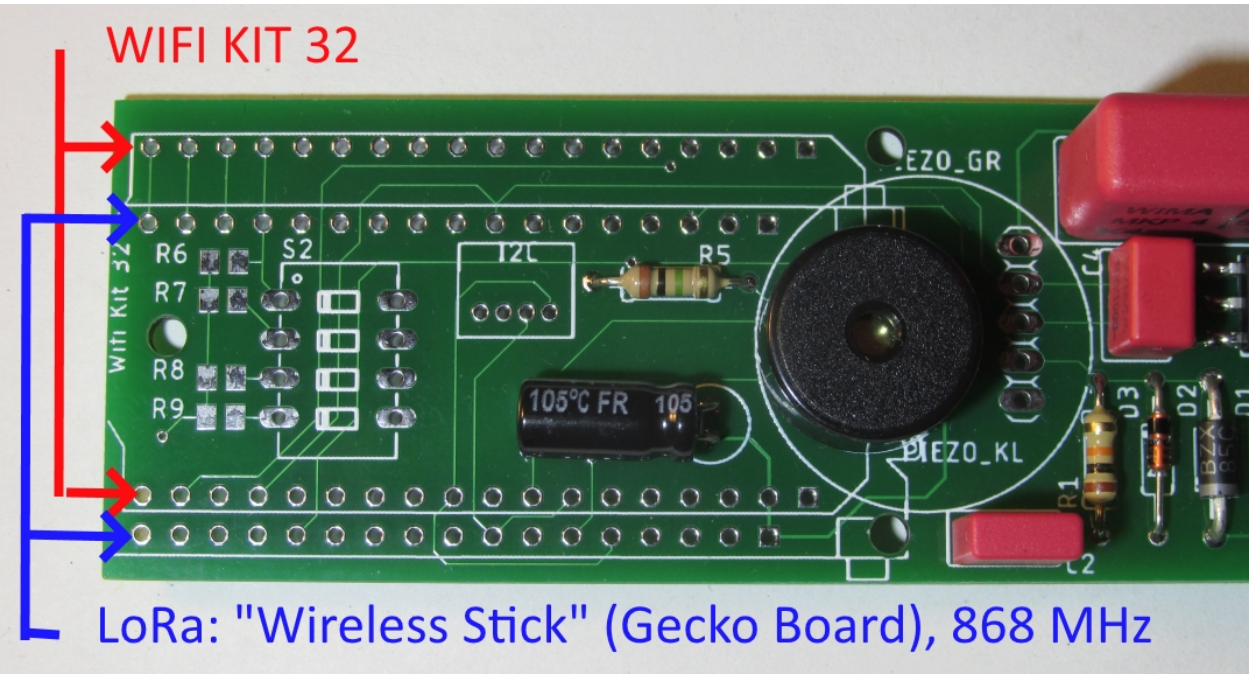
---

**Note:** Dabei muss der Seitenschneider genau auf dem **letzten nicht mehr benötigten Kontakt** angesetzt werden – **nicht dort wo man ihn eigentlich kürzen will** – ansonsten kann sie splittern. Beim Abschneiden wegfliegende Teile möglichst festhalten!

---



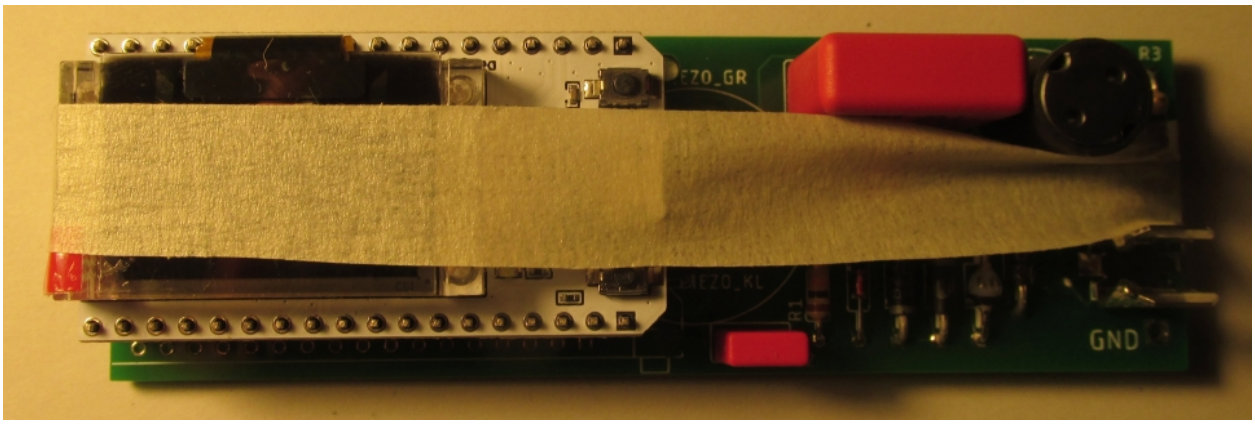
Wird der große Piezo-Lautsprecher verwendet, so sollte die Buchsenleiste mit einer Feile etwas abgefeilt werden (vorher ausprobieren). Am einfachsten geht das an der originalen, nicht bearbeiteten Seite der Buchsenleiste. Je nachdem, welches Mikrocontroller-Board verwendet wird, müssen die Buchsenleisten an den entsprechenden Stellen bestückt werden.




Die Winkel der Buchsenleiste sollten genau stimmen. Daher am besten so vorgehen:

- Die erste Buchsenleiste einstecken.
- Von unten her eine Lötstelle in der Mitte löten, die Lötstelle nochmals heiß machen und gleichzeitig die Buchsenleiste justieren (rechter Winkel, alle Pins richtig).
- Die zweite Buchsenleiste einstecken.
- In beide Buchsenleisten die Stiftleiste aus dem Mikrocontroller-Modul-Set einstecken, so dass deren längerer Teil der Stiftleiste in der Buchsenleiste steckt.
- Das Mikrocontroller-Modul aufstecken, so dass das Display sichtbar ist und die Mikro-USB-Buchse über dem Piezo-Lautsprecher liegt.
- Alles mit einem schmalen Klebeband fixieren (siehe Bild unten), so dass alle Lötstellen zugänglich sind.
- Nun können alle Kontakte gelötet werden.

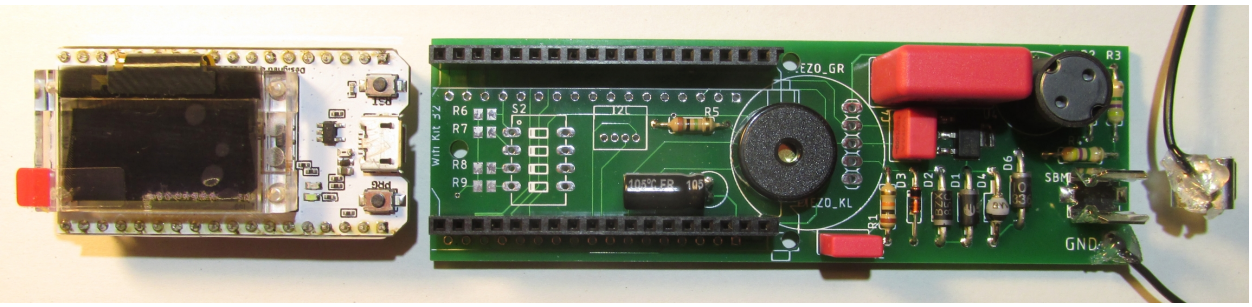
**Note:** Das OLED-Grafikdisplay ist über eine dünne Flex-Leitung angeschlossen, die nicht zu heiß werden darf. Im Zweifel lötet man die Pins in der Nähe der Flex-Leitung nicht an – sie werden nicht benötigt.



Nach dem Löten kann das Mikrocontroller-Modul mit wippenden Bewegungen vorsichtig abgezogen werden.

- Beim Mikrocontroller-Modul sind Pin-Beschriftungsaufkleber mit dabei. Diese können jetzt seitlich aufgeklebt werden. Welcher Pin wo ist, steht auf der Unterseite des Mikrocontroller-Moduls.
- Das Kabel für den Anschluss der Zählrohr-Kathode (Minus-Anschluss) an Klammer und Platine anlöten.
- Kabel mit jeweils einem Tropfen Heißkleber sichern (Zugentlastung).
- Sichtkontrolle (hast du Lötstellen vergessen, sind Lötbrücken entstanden, . . . ).
- Mikrocontroller-Modul wieder aufstecken.
- Zählrohr anklemmen: der Plus-Pol (Anode) ist markiert und muss in Richtung Platine zeigen.
- Eine erste Funktionskontrolle ist nun möglich, s. *Inbetriebnahme*.

Fertige Geigerzähler- und Mikrocontrollerplatine:




**Note:** Das dünne Glas des OLED-Grafikdisplays ist empfindlich. Bitte keine Kraft auf das Display ausüben, sondern nur auf die Mikrocontrollerplatine!

**Warning:** Verletzungsgefahr: Auf der Geigerzählerplatine wird eine **Spannung von 400 Volt** erzeugt. Diese hält sich auch noch längere Zeit **nach dem Ausschalten des Geräts**. Bei Berührung kann es zu kleineren Stromschlägen kommen, die normalerweise harmlos sind. Dennoch sollten sie vermieden werden!

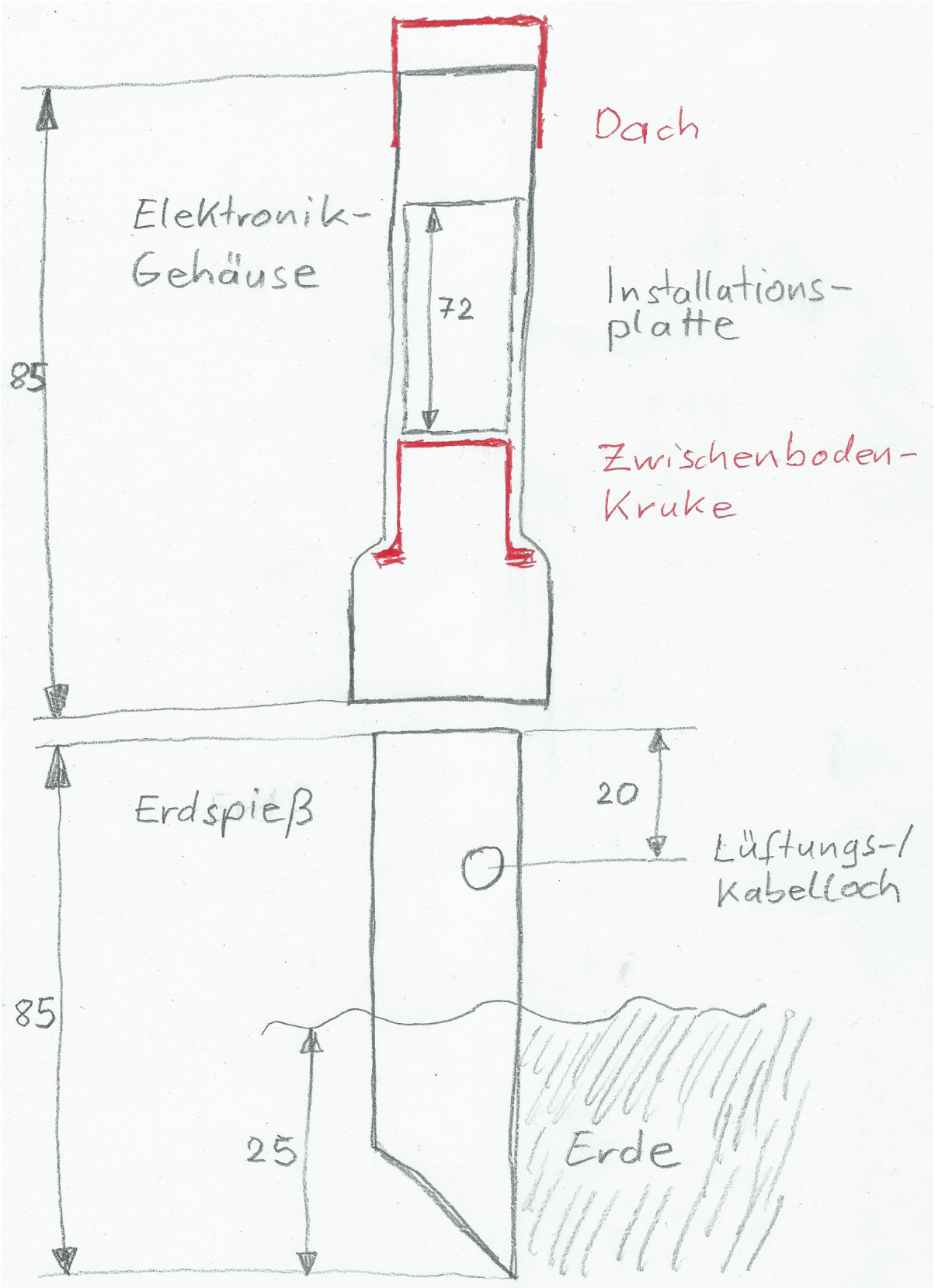

**Man sollte vor dem Arbeiten an der Schaltung den Kondensator entladen.** Am einfachsten macht man dies mit einer zweiten, ca. 30 cm langen Leitung, mit der man Anode und Kathode des Zählrohrs für etwa eine Sekunde kurzschließt. Dies schützt nicht nur vor unangenehmen Stromschlägen, sondern schützt auch die empfindliche elektronische Schaltung.

### Gehäuse

### Funktionen des Gehäuses

- Es schützt die Elektronik vor Umwelteinflüssen (z. B. Regen) und Insekten (daher sollte kein Zugang von außen zur Elektronik größer sein als 0,5 mm).
- Es hält das Zählrohr in einer Höhe von 1 m über der Wiese – somit sind unsere Messwerte (bestmöglich) vergleichbar zu denen des amtlichen ODL-Messnetzes.

### Rohre


Dach
Elektronik-
Gehäuse
72
Installations-
platte
85
Zwischenboden-
Kruke
Erdspieß
20
Lüftungs-/
Kabelloch
85
25
Erde

- Rohr für das Elektronikgehäuse: Zunächst wird das Rohr mit einer Säge auf entsprechende Länge abgesägt. Die Muffe (Aufweitung) zeigt nach unten.
- Rohr für den Erdspieß: Der noch umgesägte Teil des Restes ist das obere Ende des Erdspießes. Unten wird der Erdspieß auf entsprechende Länge abgesägt im 45-Grad-Winkel. Mit diesem schrägen Anschnitt lässt sich der Erdspieß später mit leichten Drehbewegungen gut in die Erde bohren. Alternativ kann man diesen Teil des Rohrs kurzhalten (ca. 15 cm) und in einem verzinkten Sonnenschirmhalter mit Erdspieß (Ø40mm) festklemmen.
- Lüftungs-/Kabelloch bohren: Mit einem Reißnagel o. ä. wird das Loch seitlich im Erdspieß vorgestochen. Anschließend wird es aufgebohrt. Zuletzt wird es mit einem großen Bohrer / Schälbohrer auf etwa 17 mm Durchmesser aufgebohrt.

**Note:** Verletzungsgefahr: Mit der Klinge/dem Bohrer immer nach außen, d. h. vom Körper weg, schneiden oder bohren!

### Dach und Zwischenboden

Für das Dach kann man einen Rohrabschlussstopfen oben vorsichtig über das Rohr schieben, s. Bild rechts.

Als Zwischenboden verwenden wir eine Kruke, also eine Plastikdose, die in Apotheken zum Abfüllen von Salben verwendet wird. Der rote Deckel kann als Material für eine kabeldurchführungsverkleinernde Zwischenbodenauflage dienen.

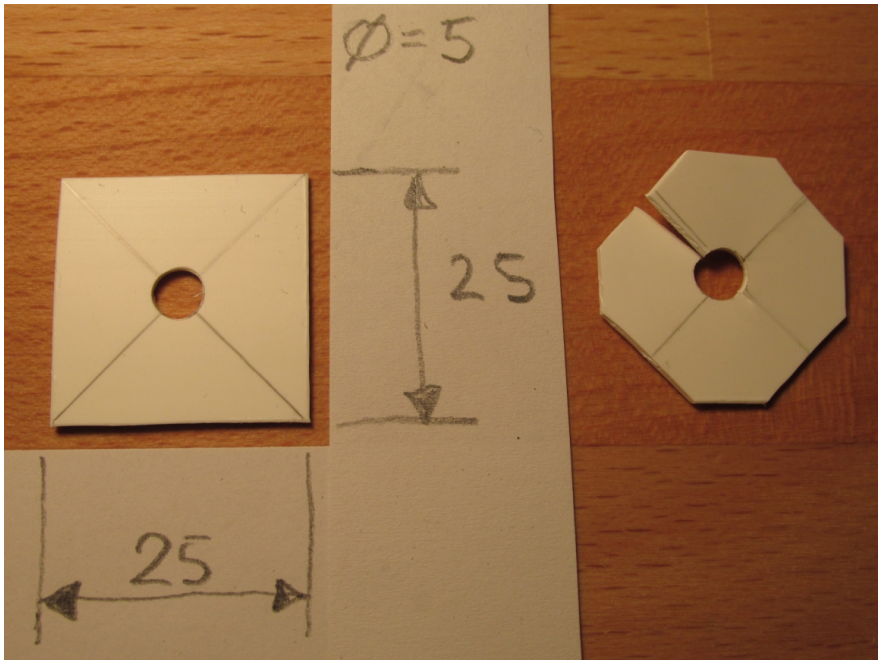
Für den Zwischenboden wird lediglich das Gewinde vorsichtig mit der Säge abgesägt, s. Bild links. Der „Kragen“ muss bestehen bleiben, da er als Anschlag dient.

In den Zwischenboden wird in der Mitte ein Loch zur Kabeldurchführung mit 10 mm gebohrt. Der Stecker des Kabels muss durch das Loch passen.

Eine Zwischenbodenauflage dient zum Schutz vor Insekten, da sie das Loch der Kabeldurchführung verkleinert. Man kann dafür z. B. den roten Deckel der Kruke oder ein ähnliches Stück Kunststoff verwenden.

Es wird die Mitte markiert und dort ein Loch mit dem Durchmesser des Stromkabels (ca. 5 mm) gebohrt. Anschließend werden die Ecken abgeschnitten. Zusätzlich wird ein Verbindungsschlitz von außen zum Loch geschnitten.

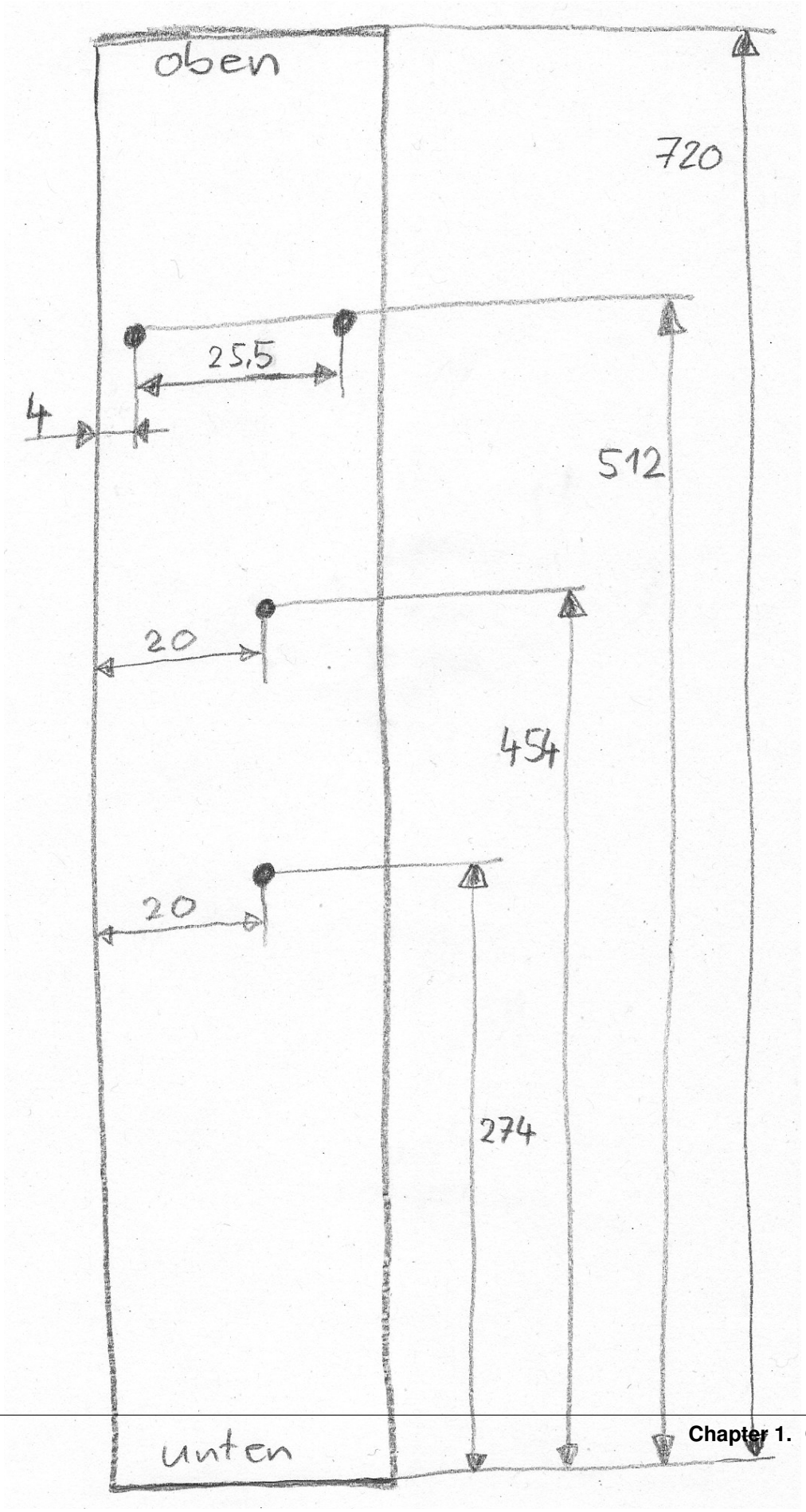
Aufbau der Zwischenbodenauflage (in zwei Arbeitsschritten):




### Installationsplatte

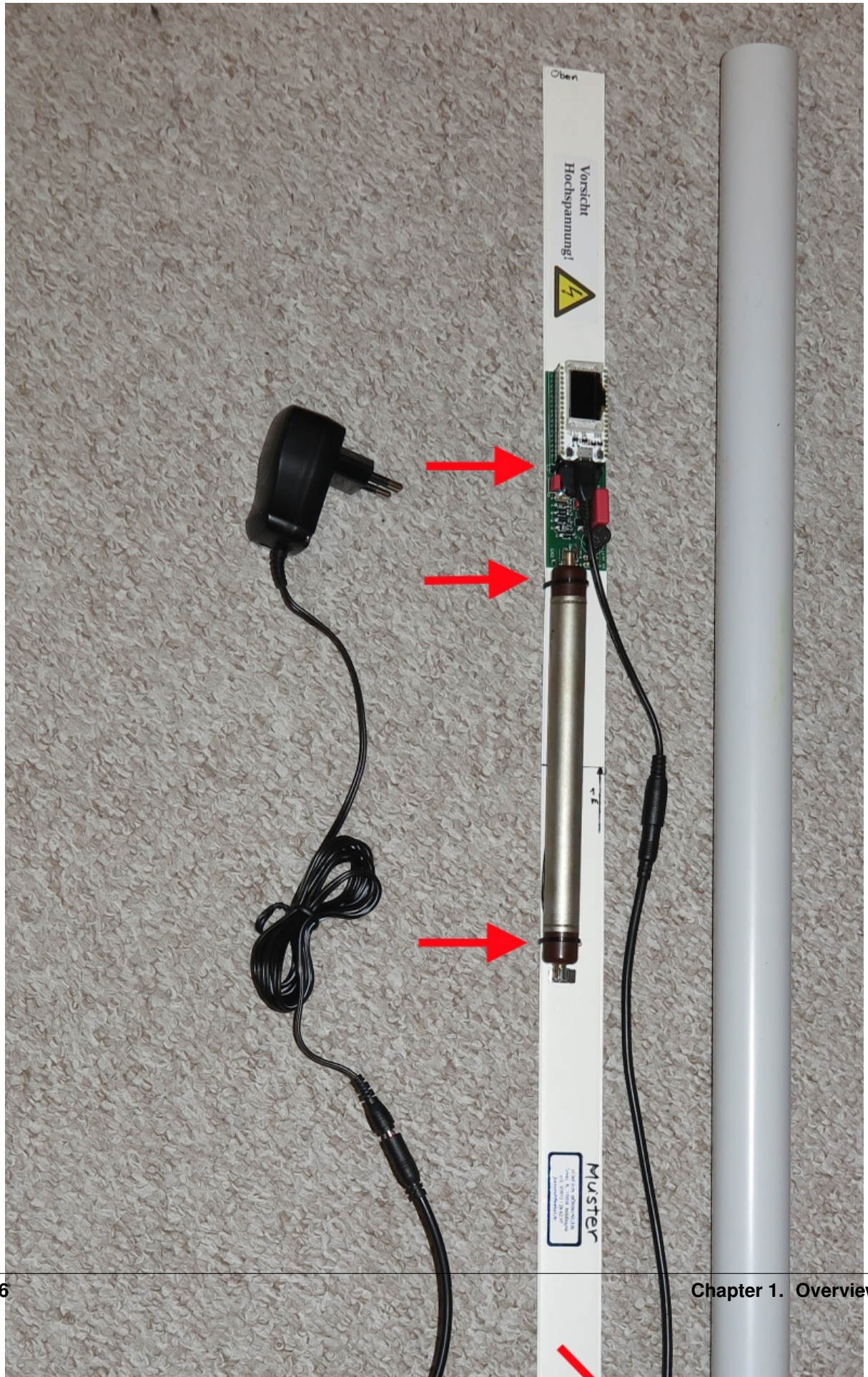
- Die Installationsplatte kann aus dem Ober- oder Unterteil eines Kabelkanals gefertigt werden. Dazu werden die Seitenteile vorsichtig mit einem Messer abgeschnitten. Danach wird der dadurch entstandene, biegsame Kunststoffstreifen auf die richtige Länge gebracht. Hierzu kann eine starke Schere verwendet werden.
- Löcher durch Installationsplatte bohren: Elektronik und Zählrohr werden so mit Kabelbindern auf Installationsplatte fixiert, dass das Zählrohr sich auf 1 m Höhe befindet, wenn der Erdspieß 25 cm in der Erde steckt. Das Zählrohr zeigt nach unten. Die Löcher werden mit dem Reißnagel vorgestochen und dann auf 3 mm aufgebohrt, s. Bild „Bohrplan Installationsplatte“.
- Die Platine wird lediglich mit einem Kabelbinder auf der Platte montiert, der durch beide Löcher gezogen wird. Er verläuft unterhalb des USB-Steckers.

Bohrplan Installationsplatte:


oben
720
25,5
4
512
20
454
20
274
unten

### Einbau

Einbau des Geräts (die roten Pfeile markieren Kabelbinder):


Oben
Vorsicht
Hochspannung!
Muster

- Platine und Zählrohr werden mit Kabelbinder auf die Installationsplatte montiert.
- In die Micro-USB-Buchse wird der passende Adapter gesteckt.
- Das „10 m Verlängerungskabel“ wird richtig herum zunächst durch den Erdspieß und dann durch den Zwischenboden gefädelt. Anschließend wird er mit dem Adapter verbunden.
- Danach wird die Zwischenbodenauflage oberhalb des Zwischenbodens vorsichtig auf das Kabel geschoben.
- Oberhalb der Zwischenbodenauflage wird ein Kabelbinder auf das Kabel gezurrt, so dass eine Zugentlastung entsteht.
- Erster Test des Gehäuses: Wir schieben erst vorsichtig die Installationsplatte in das Elektronikgehäuse. Anschließend folgen Zwischenboden und der Erdspieß.
- Wenn alles passt wird noch die Zwischenbodenauflage mit Silikon-Kleber auf den Zwischenboden geklebt und mit Kreppband (bis zum Aushärten) fixiert.
- Der Warn-Aufkleber „Vorsicht Hochspannung“ wird auf die Installationsplatte geklebt.

### Inbetriebnahme

WLAN-Variante: Das Gerät baut einen eigenen WLAN-Accesspoint (AP) auf. Die SSID des AP lautet ESP32-xxxxxxxx, wobei die xxx die Chip-ID des WLAN-Chips sind (Beispiel: ESP32-51564452). Bitte diese Nummer notieren, sie wird später für die Anmeldung benötigt, z. B. bei https://devices.sensor.community/. Der Access-Point bleibt für 30 s aktiv. Danach versucht das Gerät, sich mit dem (früher) eingestellten WLAN zu verbinden. Dieser Verbindungsversuch dauert ebenfalls 30 s. Kommt keine Verbindung zustande, wird wieder der eigene AP für erzeugt. Das standardmäßig vergebene WLAN-Kennwort ist „*ESP32Geiger*“ und sollte zeitnah geändert werden.

Weitere Informationen: *Deployment* und *Usage*.

**Wir wünschen viel Freude mit dem neu gebauten Gerät und hoffen dass es niemals unnatürliche bzw. gesundheitsschädliche Ausschläge messen wird!**

### Setup

Download and unpack the latest release from GitHub ( https://github.com/ecocurious/MultiGeiger/releases ) as source code (zip) or source code (tar.gz). In the new directory, open the directory *multigeiger* and load the file *multigeiger.ino* with the Arduino IDE.

The board supports two different Heltec devices, different counting tubes and optionally a sensor for ambient temperature, air pressure, and humidity (BME280 or BME680). The software can send data via network to different services.

- **Heltec WiFi Kit 32** This MCU has WiFi, a large display and plugged into the longer female connectors on the board.
- **Heltec Wireless Stick** This MCU has a very small display, and provides LoRa beside WiFi. It must be plugged into the shorter female headers on the board.

To select the Heltec boards in the Arduino IDE, the following steps must be taken:

- Add the file: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json in Preferences->Additional Boards Manager URLs.
- Then, the ESP32 boards (name “esp32 by Espressif Systems”) can be installed under Tools->Board->Boards Manager. Choose the correct Heltec board under Tools->Board.
- Select the **Heltec wireless Stick** for **both** boards.

- If you use arduino-esp32 >= 1.0.5, you might need to apply a patch to get back the "partition scheme" menu needed in the next step, see the `misc/arduino-esp32/paritition-menu-*.*` files in our git repo.
- Select under Tools **Flash size: "4MB(32Mb)"** and **Partition Scheme: "Minimal SPIFFS (Large APPS with OTA)"**. The software recognizes automatically which board is equipped.

Various software settings can be made via the following files (see comments inside):

- **./multigeiger/userdefines.h** (always necessary, an example is provided in userdefines-example.h)
- **./platformio.ini** (only for platformio, an example is provided in platformio-example.ini)

All external libraries are required, which are listed in the file

```
platformio-example.ini
```

under the section

```
lib-deps =
```

Please install the latest version via platform.io / Libraries.

**Caution:** If the Arduino IDE is used, please check that in the file project_config/lmic_project_config.h (in the top level in this library) the correct configurations are set. The file must look like this:

```
// project-specific definitions
#define CFG_eu868 1
//#define CFG_us915 1
//#define CFG_au921 1
//#define CFG_as923 1
// #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP /* for as923-JP */
//#define CFG_in866 1
#define CFG_sx1276_radio 1
//#define LMIC_USE_INTERRUPTS
```

The specified versions of the libraries are the minimum requirements. We always test with the latest versions too, so please always install and use the latest versions. If the compiler reminds other libraries, please install them in the Arduino IDE via *Sketch -> Include Library -> Manage Libraries ...*

### Procedure after startup

The device establishes its own WiFi access point (AP). The SSID of the AP is **ESP32-xxxxxx**, where the xxx are the chip ID (or MAC address) of the WiFi chip (example: **ESP32-51564452**). **Please write down this number, it will be needed later.** This access point remains active for 30 sec. After that the device tries to connect to the (previously) defined WiFi netwword. This connection attempt also takes 30sec. If no connection could be established, the own AP is created again and the process starts again and again.

### Configuring the device via WiFi

After the WiFi AP of the device appears on your cell phone or computer, connect to it. The connection asks for a password, it is **ESP32Geiger**. The start page of the device opens usually **automatically**. If the start page does not appear, you have to call the address **192.168.4.1** with a browser. The start page appears, where you can't miss the link to the **configure page** , click on it and you enter the settings page.

The settings page has the following lines:

- Geiger accesspoint SSID This is the SSID of the built-in AP and can be changed. If the sensor was already registered with this number at sensor.community, a new registration is mandatory.

- Geiger accesspoint password This is the password for the built-in AP. It **MUST** be changed the first time. If desired, the default password **ESP32Geiger** can be used again. The field must not be left blank. Save the password to your favourite password manager.
- WiFi client SSID Here you have to enter the SSID of the WLAN you want to connect for network/internet access.
- WiFi client password And here the corresponding password.

For more security, it is recommended to use a separate WiFi network (e.g. guest network) to ensure an isolated communication from the normal network.

If everything is entered, press **Apply** and the data are stored in the internal EEPROM. Leave this page via **Cancel**, because only in this way the program closes the Config-Mode and connects to the local WiFi network. If there is no **Cancel** Button, go back to the WiFi settings of the device and type in the normal home network parameters again.

**CAUTION**. **When updating to version 1.13, the WiFi settings must be re-entered**. In future versions this step shall become obsolete.

Furthermore, the following options can be defined on the settings page:

- Start melody, speaker tick, LED tick and display on/off.
- Send data to sensor.community or/and to madavi.de
- If LoRa hardware is available: the LoRa parameters (DEVEUI, APPEUI and APPKEY) can be entered here.

The firmware on the MultiGeiger can be updated with the link **Firmware update** at the End of the settings page. Download the .bin file, select it via **Browse...** and click **Update**. It will take roughly 30sec for uploading and flashing the firmware. If you see **Update Success! Rebooting...**, the MultiGeiger will reboot and the new firmware will be active.

If **Update error: ...** appears, the update did not work. The previous firmware is still active.

The settings page can be called up from your own WiFi at any time. To do this, just enter in the address bar of the browser: http://esp32-xxxxxxx (xxxxx is the chip ID – see above). If it does not work with this hostname, use the IP address of the Geiger counter instead. The Ip address can be found in the devices list in your router. If successful, the login page appears. Enter **admin** as username and the chosen password (see above). Now you will see the settings page as described.

### Server for measured data

The pulses are counted for one measuring cycle at a time, from which the "counts per minute" (cpm) are calculated. After each cycle the data is sent to the servers at *sensor.community* and at *madavi.de*.

At *sensor.community* the data is stored and made available for retrieval the next day as CSV file. This file can be found at http://archive.sensor.community/DATE/DATE_radiation_si22g_sensor_SID.csv), where DATE = date in format YYYY-MM-DD (both times equal) and SID is the sensor number of the sensor (**not** the ChipID). For other sensors, replace the counting tube name **si22g** with the corresponding name (e.g.: sbm-20 or sbm-19).

At *madavi* the data is stored in a RRD database and can be accessed directly as a graph via this link: https://www.madavi.de/sensor/graph.php?sensor=esp32-CHIPID-si22g. Here CHIPID is the ChipId (the digits of the SSID of the internal access point).

During the transmission of the data to the servers, the name of the server is briefly shown in the status line (bottom line) of the display.

### Login to sensor.community

In order to send the measuered data to sensor.community, it is mandatory to have a valid account and the sensor is registered. Both can be done at https://devices.sensor.community. Create an account if you do not have one via the *Register* button and log in. To register a new sensor click *Register new sensor*. Fill in the form:

- Sensor ID: Enter the number (only the numbers) of the SSID of the sensor (e.g. for the sensor ESP-51564452 enter 51564452).
- Sensor Board: Select *esp32* (by the small arrows on the right)
- Basic information: Enter the address and the country. The internal name of the sensor can be assigned arbitrarily, but must be entered. Please check **Indoor sensor** as long as the sensor operates not outdoor.
- Additional information: Can be left blank, but its nice to provide further information.
- Hardware configuration: Select the sensor type **Radiation Si22G** (or accordingly). The value for the second sensor can remain DHT22, as it is irrelevant in this context.
- Position: Please enter the coordinates as accurate as possible. You can use the right button to calculate the coordinates. They are needed to show your sensor on the map.

Finish the settings by clicking *Save settings*. At the overview page for this sensor go to *Data*. Here you see amongst others the ID of the sensor. Please remember: the ID mandatory for the queries at sensor.community or multigeiger.citysensor.de.

### Setup (LoRA)

The MultiGeiger can be connected with the followng steps to TTN ("The Things Network"):

- Create the TTN device in your profile at *The Things Network*
- Transfer the parameters to the multi-pointer
- Login at *sensor.community* (former luftdaten.info)
- HTTP integration

### Creating a TTN device

The device must be registered with TTN. To do this, an account must first be created at TTN (if one does not already exist).

### Create TTN account

At https://account.thethingsnetwork.org/register you have to enter a **USERNAME**, the **EMAIL ADDRESS** and a **PASSWORD**. Then right down over **Create account** create the account. After that you can log in to the console with the new data (https://account.thethingsnetwork.org/users/login).

### Create application

After you have logged in successfully, create the new application via **APPLICATIONS** and **add applications**. The following fields must be filled in:

- **Application ID:** Any name for this application, but it must not yet exist in the network (e.g.: geiger_20200205).

- **Description:** Any description of the application can be entered here.
- **Application EUI:** Remains empty, the number is generated by the TTN system.
- **Handler registration:** The pre-filled value (ttn-handler-eu) is already correct and remains.

Now add the application with **Add application** in the lower right corner.

### Create device

Finally, the device has to be created. To do this, select the newly created application in the overview of applications (click). Then in the middle area at **DEVICES** start the creation of a new device via **register device**. The following fields must be filled in:

- **Device ID:** Any name for the device. It must be unique within the application (e.g.: geiger_01) and consist only of lower-case letters.
- **Device EUI:** Click once on the symbol on the far left of the line, then the text appears that this number is generated by the system. We don't have to enter anything else.
- **App Key:** No input necessary
- **App EUI:** Stays like this

Click on **Register** in the lower right corner. Congratulations, the device is created.

### Modifying the LoRa parameters

- After the registration was completed, the LoRa parameters can be transferred to the program.
- They can be set up at the configuration site of the Geiger counter (see above).
- Go through the configuration site until the settings of the LoRa parameters are displayed. Type in the 3 parameters from the TTN console (**APPEUI, DEVEUI, APPKEY**). They can be found in your TTN account for each device (see above). The HEX values must be entered **without** spaces as they appear in the TTN.

Example: | The TTN console reads

```
Device EUI 00 D0 C0 00 C3 19 7C E8
```

Then the following must be entered:

```
00D0C000C3197CE8
```

This is also applies to **APPEUI** and **APPKEY** .

### Logging data into sensor.community (formerly luftdaten.info)

If you want the MultiGeiger to pass recorded data to *sensor.community* via TTN, you have to register it. The registration is similar to the TTN registration described above. In the following, only the changes are explained:

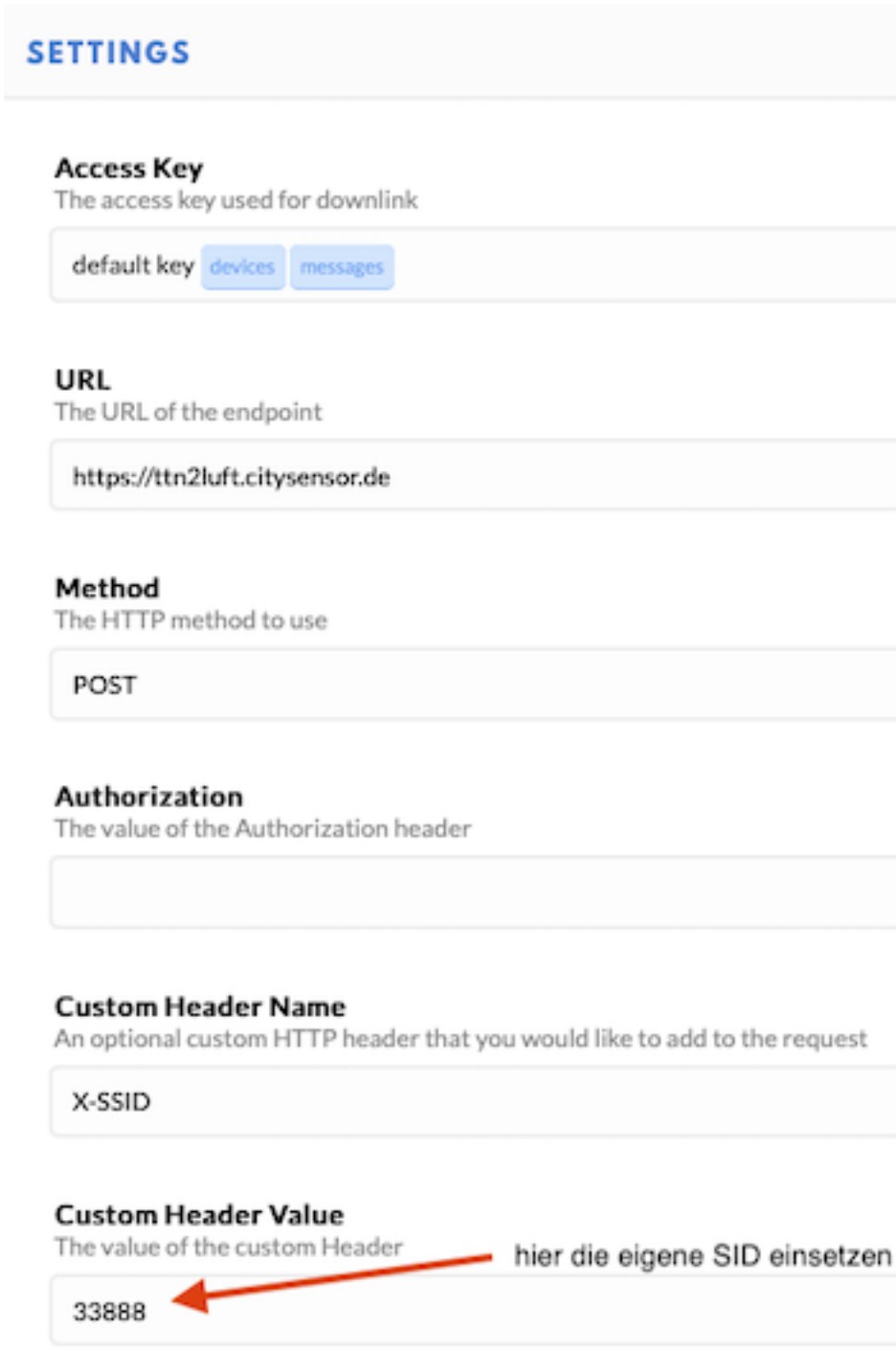
- Sensor ID: Enter the last 4 bytes of the DEVEUI in left to right order (e.g. if the DEVEUI is *00 D0 C0 00 C3 19 7C E8*, so enter *C3197CE8*), but converted to decimal, not in HEX (finally: 3273227496).
- Sensor Board: Select **TTN**, using the small arrows on the right.

### HTTP integration

To get the data from TTN to *sensor.community* you have to enable the HTTP integration at TTN. In the TTN console click *Applications* and then click on the application of the GeigerCounter (e.g. *geiger_20200205*). On the top right in the bar with *Overview*, *Devices*, *Payload Formats*, *Integrations*, *Data*, *Settings* click **Integrations**. Then select **HTTP Integration** via **add integration**.

**Now fill in the displayed fields:**

- **Process ID** Enter any name for this integration here **Access Key:** Click here once and select the *default key*
- **URL:** Enter the URL for the ttn2luft program: https://ttn2luft.citysensor.de
- **Method:** If it reads already *POST*, don't touch it
- **Authorization:** remains empty
- **Custom Header Name:** here comes the text **X-SSID** pure
- **Custom Header value:** Enter the SSID of the sensor (the number you got when you registered at sensor.community, *NOT* the chip ID).
- Click **Add integration** in the lower right corner to confirm the changes.




See this example how the form should look like:

### TTN payload (example)

In order to get readable values in the TTN console instead of solely data bytes, a small script can be inserted as payload decoder. Go to the TTN website, log in, click **Applications** to find the application you created above. Select the tab **Payload Formats** in the menu bar and paste the following code into the field. Existing code will be overwritten):

```
function Decoder(bytes, port) {
  // Decode an uplink message from a buffer
  // (array) of bytes to an object of fields.
  var decoded = {};
  if(port == 1) {
  decoded.counts = ((bytes[0]*256 + bytes[1]) * 256 + bytes[2]) * 256 + bytes[3];
  decoded.sample_time = (bytes[4] * 256 + bytes[5]) * 256 + bytes[6];
  decoded.tube = bytes[9];
  var minor = (bytes[7]&0xF)+(bytes[8]>>4) ;
  decoded.sw_version="" + (bytes[7]>>4) + "." + minor + "." + (bytes[8]&0xF);
  }
  if (port === 2) {
    decoded.temp = ((bytes[0] * 256 + bytes[1]) / 10) + "°C";
    decoded.humi = bytes[2] / 2 + "%";
    decoded.press = ((bytes[3] * 256 + bytes[4]) / 10) + "hPa";
  }
  return decoded;
}
```

### Usage

### OLED display

### Top line

Left: Time since power-on (not shown on small displays).

Right: Overall average radiation since power-on.

### Middle area

Current CPM (counts per minute) displayed using a rather big font.

### Bottom line

This is a status display with 8 positions, numbered 0..7:

Rules of thumb:

- `.` usually means “off” or “unused”.
- if you see some *number* (`0` .. `7`) within the status display line, something went wrong.

Positions:

- 0: WiFi
  - `A`: AccessPoint active
  - `w`: WiFi client trying to connect
  - `W`: WiFi client connected
  - `0`: some error happened
- 1: sensors.community transmission

  - `.`: off (not configured, not enabled)
  - `?`: init (enabled, before 1st transmission)
  - `S`: sending
  - `s`: idle (shown after successful sending)
  - `1`: sending failed (shown after trying to send)
- 2: madavi transmission
  - `.`: off (not configured, not enabled)
  - `?`: init (enabled, before 1st transmission)
  - `M`: sending
  - `m`: idle (shown after successful sending)
  - `2`: sending failed (shown after trying to send)
- 3: TTN ("The Things Network")
  - `.`: off (not configured, not enabled, no LoRa hardware)
  - `?`: init (enabled, before 1st transmission)
  - `T`: sending
  - `t`: idle (shown after successful sending)
  - `3`: sending failed (shown after trying to send)
- 4: BLE (Bluetooth® Low Energy)
  - `.`: off (not enabled)
  - `?`: init (enabled, before setup of BLE service)
  - `B`: connected and sending notifications, if requested by connected device
  - `b`: connectable (advertising and ready to connect)
  - `4`: BLE error
- 5: unused
- 6: unused
- 7: High-Voltage Capacitor charging
  - `H`: OK
  - `7`: failure to charge HV capacitor

### ESP32 buttons

The MultiGeiger ESP32 microcontroller board has 2 buttons:

- RST: reset (restarts device)
- PRG: program (press and hold PRG, press RST momentarily, upload firmware)

Often, using PRG is not needed for flashing firmware, because it "just works" when using the right tools.

### DIP Switches

Optionally, the MultiGeiger can be equipped with a 4-contact DIP switch under the ESP32 microcontroller board.

The firmware makes use of these 4 switches as defined in the `Switches` data type in `switches.h`:

- SW0: Speaker on
- SW1: Display on
- SW2: LED on
- SW3: BLE on

### BLE - Bluetooth® Low Energy

### BLE Heart Rate Service

The MultiGeiger provides a Bluetooth® Low Energy (BLE) service to allow the collection of the Geiger-Mueller count rate via a GATT Heart Rate Service (Service UUID 0x180D). The following characteristics are used:

- 0x2A37 ('Heart Rate Measurement Characteristic'):
  - The first byte is a collection of status flags, according to the service's standard
  - 'Heart Rate Measurement' as 16 bit value (little endian), corresponds to Geiger-Mueller counts per minute (CPM)
  - 'Energy Expenditure' as 16 bit value (little endian), represents a rolling packet counter
- 0x2A38 ('Heart Rate Sensor Position Characteristic')
  - 'Sensor Position' as 8 bit value, corresponds to TUBE-TYPE, allowing the conversion of CPM to radiation rate
- 0x2A39 ('Heart Rate Control Point Characteristic')
  - Write characteristic, required by service's standard to reset Energy Expenditure to 0. Writing 0x01 resets the rolling packet counter to 0.

### Testing BLE

Any decent Bluetooth® scanning app should be able to connect to the device and show the notification values. Known to work well is the nordic® nRF Connect app, which should be available for iOS and Android. More infos about the Android app can be found here on github.

Any heart monitor app / device should be able to connect to the MultiGeiger, too, but that was not tested yet. Any input is appreciated.

- Figure 1: Connect to the MultiGeiger with the name 'ESP32-<id>', where <id> should be the same ID (7 to 8 digits) as with the access point to set the device up initially.
- Figure 2: Find the correct service (UUID 0x180D) and, if needed, load / open characteristics.
- Figure 3: Find the Heart Rate Measurement Characteristic (UUID 0x2A37) and click on the icon to start notifications.

While testing, please keep in mind that an update packet is sent only every ~10s (along with a display refresh, if enabled). If a WiFi transmission is blocking display and BLE updates, the interval may be even longer. So it might take a while until a reaction with actual data can be seen in the app.

*Fig. 1*

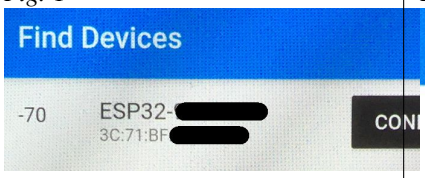



*Fig. 2*

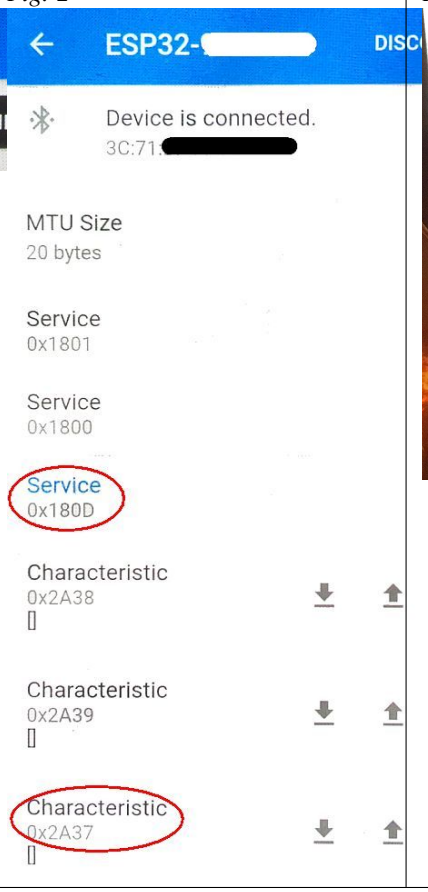
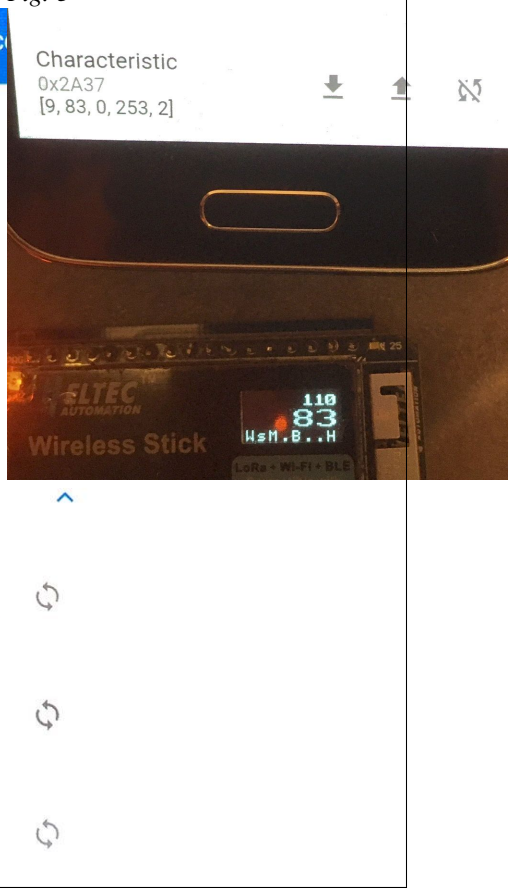



*Fig. 3*




### Deployment

When deploying the MultiGeiger measurement station, you need to consider some things.

Of course, for us private persons, there rarely will be a 100% perfect deployment place. But still, we can be aware of the parameters which may have a negative impact.

### Requirements for the deployment place

### Choosing a deployment place

We want to measure radioactive substances (dust), also known as "fallout". They come from e.g. nuclear explosions or reactor desasters.

In a very weak variant, they also naturally come from the atmosphere due to the decay of the radioactive noble gas radon. This "radon fallout" causes a small, but measurable peak on our devices, which degrades with a half life of 45 minutes.

With the help of these "radon peaks", we can see whether our device is deployed to a good place and is sensitive also for "real" fallout.

### Irrigation area

We need a bigger irrigation area in the direct vicinity of the measurement station. This area should also have the capability to hold and store radioactive particles.

A lawn or a meadow is perfect for this.

On the other hand, some asphalt can not do that, because particles are washed away directly after raining down.

### Wind direction

The area should be unobstructed from the main wind direction by other buildings, bushes or trees.

### Sun light

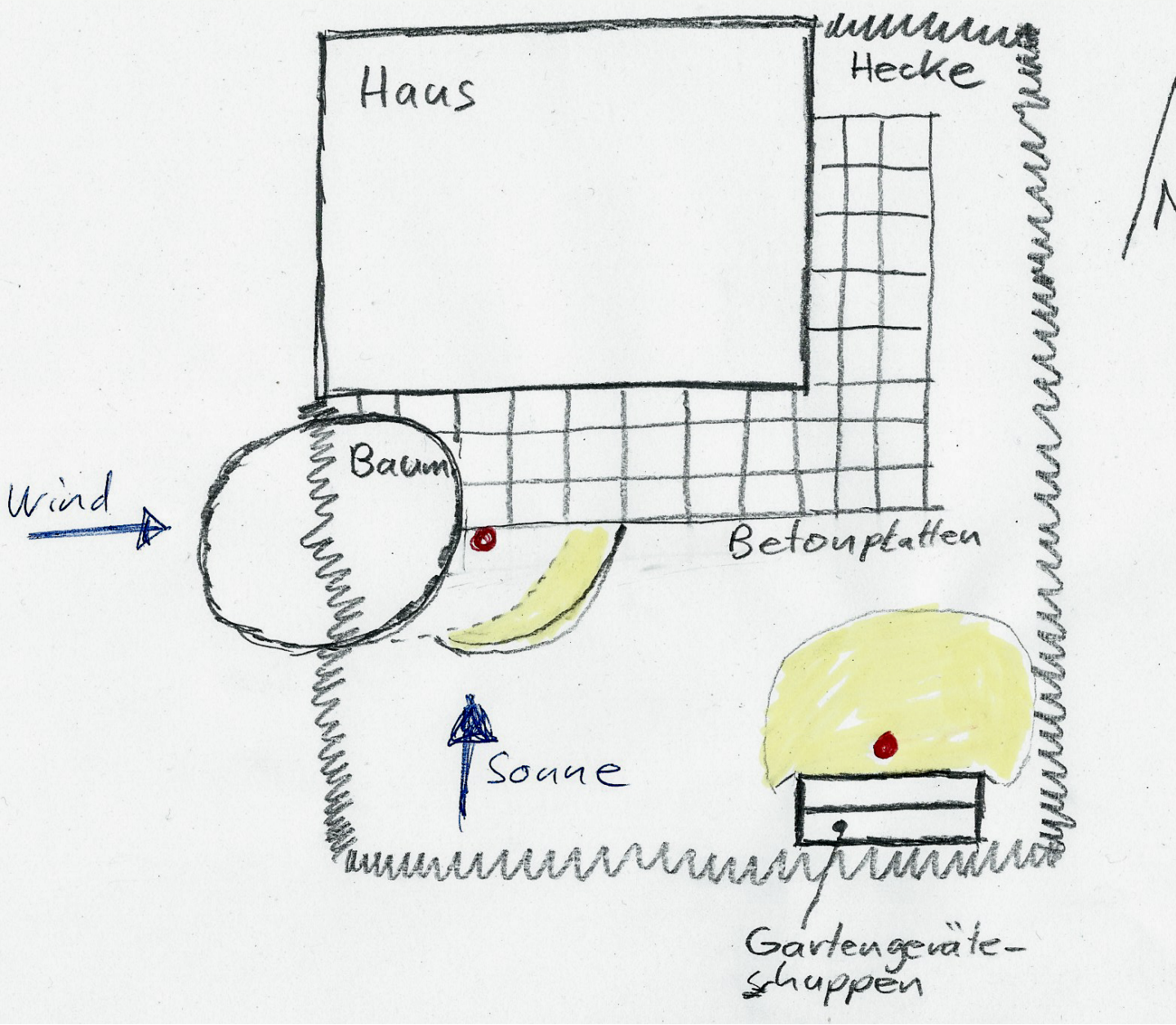
The power supply cable as well as the MultiGeiger device itself, should be in the shadows most of the time.

Direct sunlight can have two negative influences:

- Heat: The temperature inside the device can rise to up to 60 deg celsius in direct sunshine (45 deg celsius in the shadows). We did practical temperature tests showing that the device works ok up to 70 deg celsius. But the GM tube is only specified to up to 50 deg celsius.
- UV damage: Over time, the cable and case can be damaged by UV radiation and get porous.

### Examples

Examples of good and bad deployment places:




Comparison of two deployment places. Yellow is the “usable irrigation area”.

Bad deployment place (red dot at the left)

Reasons:

- Meadow: a bit part of the area around the GM device is made of concrete platter. Incapable of holding/storing fallout.
- Wind: the main wind direction is from the west and the tree obstructs fallout raining down close to the device.
- Sun: Sunlight comes primarily from the south and strongly shines on the devices.

Good deployment place (red dot at the right)

Reasons:

- Meadow: a big part of the area around the GM device is meadow, which can hold/store fallout.

- Wind: no obstruction
- Sun: the GM device is in the shadow due to the garden shack.

#### More Information

Description of the BFS-ODL measurement network (german):

- Messnetz
- ODL-Sonde

Video interview about the BFS-ODL measurement network (german):

- Interview über das BFS-ODL-Netzwerk

### Frequently Asked Questions

#### General

#### Radiation doses, are there practical examples?

There's a great XKCD about that, see there:

https://xkcd.com/radiation/

### Resources

This is a collection of additional resources that are somehow related to MultiGeiger.

#### Hardware / Parts

MultiGeiger supported ESP32 boards:

- Heltec WiFi Kit 32 (WiFi-only, infos from board manufacturer) https://heltec.org/project/wifi-kit-32/
- Heltec Wireless Stick (WiFi+LoRa, infos from board manufacturer) https://heltec.org/project/wireless-stick/

#### Software

- Windows driver for the USB -> UART chip CP2102: http://esp32.net/usb-uart/#SiLabs

### Changelog

#### V1.15.0 2021-03-21

New features:

- add bluetooth (BLE) support, #78

Fixes:

- improve LoRaWAN stability (work around LMIC bug #677, add LMIC polling from loop()), #373

- do async NTP/clock setup, #316
- speaker: init "duty_mode" member in MCPWM config
- avoid using IotWebConf 3.0.0 for now, #357, PR #370

Other changes:

- patch: restore partition scheme menu for arduino-esp32 1.0.5
- move CI from travis CI to github workflow
- start screen cleanups, #335
- code / naming style fixes
- remove dates in file names, commit relevant versions to git
- add drill files, #354
- docs:
  - use transifex / sphinx / readthedocs.org for translatons (en/de for now)
  - document docs/translation workflow in development docs
  - added assembly and deployment guide
  - document esp32 board buttons, #129
  - document dip switch usage, #128
  - move README-{de,en}.* contents into the .rst docs
  - BLE usage documentation update with some images, #338
  - added links to map, ecocurious, assembly room
  - markup, rendering, spelling fixes, cleanups
  - fix unclear version / date in Aufbauanleitung, #110
  - moved links to docs -> resources, #223
  - add xkcd about radiation doses to FAQ, #310

### V1.14.0 2020-05-16

New features:

- implement status line on OLED display (see docs), #257
- also support BME680 sensor for temperature, humidity, pressure
- display time up to 60s / 60m / 24h / 99d, then roll over
- speaker/LED: timer-driven sequencer, hw PWM sound, #35
- TLS support
  - add clock module, use NTP to set the clock
  - use persistent per-server HTTPClient instances
  - use connection: keep-alive for web requests
  - add https capability (can be used for sending data)
  - note: transmission to sensors.community and madavi is still using http!

Fixes:

- fixed GM pulse debouncing, #248
- pulse counting: deal with microseconds uint32 overflow, #273
- check WiFi status before trying to transmit
- fix race condition, #286

Other changes:

- dip switches: only read once at boot time, #207
- new font (u8x8 uses 8px width anyway)
- slow down main loop
- toilet -> custom server, add comments about toilet usage, #214
- refactor/simplify pulse counting ISR, bookkeeping in main loop, #220
- refactor big main loop into smaller functions with local bookkeeping.
- misc. other code cleanups
- loraWan: removed unused/not needed code, #212, #234
- removed meeting notes, #294
- docs:
  - README improvements (board name, flash size, partition scheme, passwords, LoRa)
  - update development/release docs (create/test binaries, IDE settings, . . . )

### V1.13.0 2020-04-14

- auto-detect hardware (STICK vs. WIFI) by hardware pin
- use config web page for more values (userdefines.h has the defaults), #140
- try both adresses of BME280
- LoRa payload changes, e.g. to fulfill 'TTN Fair Access Policy'
- send additional data to servers
- send to MADAVI in one single request both geiger and thp data
- new logging with DEFAULT_LOG_LEVEL configuration
- integrated travis-ci:
  - for compile checks (platformio, wifi and stick build)
  - for style checks (using the "astyle" CPP checker)
- source: modularization, cleanups, less globals, . . . (quite huge internal changes, please help testing!)
- building:
  - platformio-based build: suppress lmic_project_config.h usage
  - arduino-ide-based build: you still need to edit that file
- use bump2version tool for project version bumps, #169
- docs:

  - added upgrade hints for 1.13 in README on github
  - https://multigeiger.readthedocs.io/ == the beginning of new (sphinx / reST-markup based) online docs, #163
  - add a basic, short README in English (also for online docs)
  - include infos about project name, #121
  - moved changelog.md to docs/source/changes.rst
  - updated/fixed development docs, #46
  - update docs about new 5V power supply / cabling, #122
  - description of LoRa Payload updated
  - other docs improvements / fixes

### V1.12.0 2020-01-18

- simple OTA (Over-The-Air) updates via web browser based upload, #120
- use less charge pulses in loop() for timing, more in setup() for initial charging, #134
- output error msg on Serial if HV charging fails
- tag log output with "GEIGER: ", #85
- add TUBE_UNKNOWN 0 to have a specific value for experimenting
- adapted platformio.ini to pull all dependencies
- send CR and LF on serial
- changed default tube from sbm-20 to si22g
- semantic versioning, version numbers now like x.y.z
- changed building of revString and lora_version
- docs updated / improved
- explain SBM-19/SBM-20 conversion factor
- removed IotWebconf bundled&patched code, used as a lib now.

### V1.11.1 2019-12-16 rxf

- change luftdaten.info to sensor.community

### V1.11.0 2019-12-16 rxf

- defaults in userdefines-example.h changed
- Software version for LoRa now 2 Bytes
- Display start screen for Wireless stick fixed
- changed to semantic versioning

### V1.10 2019-12-13

- conversion factor for Si22G tube fixed
- char variables changed to int
- isr routines shielded with portMUX
- debug serial out formatting improved
- sequence of counting and dispaying and hv charging improved
- speaker and led tick fixed
- many calls to millis() consolidated

### V1.9 2019-11-12

- structure for different counter tubes
- LoRa payload changed again
- hv pulse every second
- calculate and display cpm value every 10 seconds
- fixed div by 0 if there's no tube
- Readme corrected

### V1.8 2019-11-04

- indentation/spacing, refactor OLED functions, fix conversion factor
- MEASUREMENT_INTERVAL 150sec
- changed LoRa payload

### V1.7 2019-10-21

- PINs rearranged, so we can use new Wifi-Kit-32 and WiFi Stick Light
- Hardware-Layout V1.4 and up
- use switch for speaker tick and display off

### V1.61 2019-09-30

- default measuring interval is now 2.5min

### V1.6 2019-09-13

- some rearrangement of files
- userdefine.h for user changable #defines
- test with dip-switch (needs pullup resistors!)

- Hardware layout V1.3 and lower - OLD Wifi-Kit-32!

### V1.5 2019-09-11

- added BME280 (uses same I2C as display)
- Support for display on Wireless Stick
- For LoRa-Devices added LoRa functionality

### V1.4 2019-09-03

- default configuration with measurement interval of 10min

### V1.3 2019-09-03

- building of ESP-ID out of MAC address is now identical to 'Feinstaubsensor'

### V1.2 2019-09-02

- sending to madavi corrected

### V1.1 2019-09-01

- Library IoTWebConfig changed -> function 'setThingName' added
- Move this (IoTWebConfig) library to source path
- building the SSID from the MAC corrected: first 3 Bytes of MAC build SSID
- LoRa autodetection removed

### V1.0 2019-08-19 rxf

- added detection of LoRa device
- WiFiManager to enter WLAN data and other configs
- send to luftdaten.info every 2.5 min

### V0.3 2019-05-12 jb

- added bug fix for the "Double-Trigger-Problem". This was caused by the rising edge falsly triggering an other pulse recording. The Problem is that there is no Schmitt-Trigger available in the controller.
- simplified serial printing modes
- made seconds in Display as inverse to be able to separate it from minutes
- cleaned up the code
- Fixed overflow bug in Minute-Count+

### V0.2 2019-04-26 jb

- added 1 Minute RS232 (USB) logging mode

### V0.1 2019-03-25 jb

- first version for ESP32 board

## Development

This chapter will get you started with MultiGeiger development.

MultiGeiger is written in C using Arduino-ESP32.

## Contributions

. . . are welcome!

Some guidance for contributors:

- discuss about changes on github issue tracker
- make your PRs against the `master` branch
- do clean changesets:
  - focus on some topic, resist changing anything else.
  - do not do style changes mixed with functional changes.
  - run the automatic code formatter before committing
  - try to avoid refactorings mixed with functional changes.
  - if you need to fix something after commit/push:
    - if there are ongoing reviews: do a fixup commit you can merge into the bad commit later.
    - if there are no ongoing reviews or you did not push the bad commit yet: edit the commit to include your fix or merge the fixup commit before pushing.
  - have a nice, clear, typo-free commit comment
  - if you fixed an issue, refer to it in your commit comment
- make a pull request on github and check on the PR page what the CI system tells about the code in your PR
- wait for review by other developers

## Building a development environment

TODO

### Automatic Code Formatter

We use astyle for automated code formatting / formatting checks.

Run it like this:

```
astyle --options=.astylerc 'multigeiger/\*'
```

## Documentation

### Building the docs with Sphinx

Documentation is written in English and translated to other languages from that source (initially German).

The documentation (in reStructuredText format, .rst) is in `docs/source/`, `index.rst` is the starting point there.

To build the docs, you need to have Sphinx installed and run:

```
cd docs/
make html
```

Then point a web browser at `docs/build/html/index.html`.

The website is updated automatically by ReadTheDocs through GitHub web hooks on the main repository.

After changes of the (english) master docs, the translation master files (`*.pot`) need updating (adding/removing/updating strings in there):

```
cd docs/build/gettext
sphinx-build -b gettext ../../source .
```

Then, these changes need to get pushed to transifex, so translators can comfortably translate on the web:

Translation is organised via [transifex](https://www.transifex.com/thomaswaldmann/multigeiger/), you need to have an account or at least login there and fire a "join team" request. Then translate the missing parts and notify the developers (e.g. via issue tracker).

```
tx push --source
```

Later, after translators did their part, updated translations need to get pulled from transifex:

```
tx pull --all
```

Now we have changes in our git workdir and we need to commit them:

```
git add locales/
git commit -m "updated translations"
git push
```

This will trigger a build of the docs and their translation(s) on readthedocs.io.

### Flashing devices / creating binaries

Arduino IDE:

- do a git checkout of the wanted release, e.g. `git checkout V1.13.0`

- use the default userdefines.h (available as userdefines-example.h)
- IDE settings:
  - Device: Heltec WiFi Stick (always use this, even if you have a WiFi Kit 32)
  - Flash size: 4MB (32Mb)
  - Partition scheme: minimal SPIFFS (large APPS with OTA) - this fits onto 4MB devices.
- `Arduino IDE -> Sketch -> Upload`

  This is to test whether the compiled code actually works after USB-flashing to your device.
- `Arduino IDE -> Sketch -> Export compiled binary`

  This creates a .bin file for OTA updating. Test whether OTA updating using that file works.

### Creating a new release

Checklist:

- make sure all issues for this milestone are closed or moved to the next milestone
- check if there are any pending fixes for severe issues
- check whether some CA certificate (see `ca_certs.h`) will expire soon and whether we already can add their next valid cert.
- find and fix any low hanging fruit left on the issue tracker
- close release milestone on Github
- update `docs/source/changes.rst`, based on `git log $PREVIOUS_RELEASE..`
- `bump2version --new-version 1.23.0 release` - this will:
  - update versions everywhere
  - auto-create a git tag
  - auto-create a git commit
- review the automatically generated changeset
- create a github release for this tag:
  - create a binary (see above) and attach to the github release
  - add a link to the relevant `changes.rst` section to the github release

### Authors

- Juergen Boehringer (see www.boehri.de)
- Reinhard X. Fuerst (see feinstaub.rexfue.de)
- Thomas Waldmann <twaldmann@thinkmo.de>
- and others

## License

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
```

(continues on next page)

(continued from previous page)

```
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.
```

(continues on next page)

(continued from previous page)

```
  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
```

(continues on next page)

(continued from previous page)

```
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
```

(continues on next page)

(continued from previous page)

```
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
```

(continues on next page)

(continued from previous page)

```
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
```

(continues on next page)

(continued from previous page)

```
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.
```

(continues on next page)

(continued from previous page)

```
  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
```

(continues on next page)

(continued from previous page)

```
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.
```

(continues on next page)

(continued from previous page)

```
  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.
```

(continues on next page)

(continued from previous page)

```
  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
```

(continues on next page)

(continued from previous page)

```
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```