SENIOR DESIGN PROJECT

Smart Radiation Monitoring System

Students:

Mariam Gigauri

Tengiz Pataraia

13.05.2022

Table of Contents

Abstract

The problem our team chose to work with is the monitoring of radiation exposure. The main aim is to detect and warn people about potentially harmful radiation. We have built a hardware circuit relying on the Geiger-Muller counter and utilized an IoT server that connects the devices to the website. We have created different versions of devices: handheld, half-automated and fully automated options, and tested them all out in different conditions. As a result, we made it possible for the owner to monitor radiation exposure on the desired location from any place in the world. Moreover, statistics of radiation level in different locations can be accessible for everyone on the map on our website.

Introduction

Radiation has been around us throughout our evolution. Most of it formed naturally from minerals, which are presented in ground, soil, water, and even our bodies. Thus, our bodies are designed to deal with the low levels we're exposed to every day. But too much radiation can damage tissues by changing cell structure and damaging DNA. Everyone is exposed to radiation from benign sources such as cosmic rays, radon, microwaves and cell phones. However, higher-energy short-wave radiation in many occupational sources can penetrate and disrupt living cells and increase cancer risk, so exposure must be monitored. The risk of cancer from radiation exposure reportedly increases as the dose of radiation increases. In the 21st century, radiation has become part of our life and it will not vanish. We need to learn to live with this phenomenon, both useful and dangerous. Radiation manifests itself as invisible and imperceptible emissions, and without special devices it is impossible to detect them.

Our team decided to dedicate our time and funding to researching sophisticated and relatively inexpensive solutions to monitor the radiation level in order for people to be aware of how much radiation they are being exposed to and efficiently analyze the data received.

We address safety by implementing hardware and software modules that can detect hazards and generate appropriate responses to limit the damage caused by them. One of the most important accomplishments of our project is making safety monitoring an automated process by utilizing the Internet of Things (IoT). To our counter design, we have added modules that generate information about the safety of the area around them and send that information to the IoT cloud where appropriate response is automatically generated. As a result, owners can observe radiation level in the desired location. There are three main goals our project accomplishes:

- Hardware that efficiently monitors the radiation level.
- Software which accurately analyzes the information coming from our device.
- Generation of the appropriate response (warning message/call) if hazard is detected.

Body

Radiation Detectors

Main characteristics of detectors:

1. **Efficiency** - the ratio of the number of particles registered by the detector to the total number of particles passed through it (in fractions of unity or in percent).
2. **Spatial resolution** - the error with which the detector can fix the position of the particle in space.
3. **Time resolution** (allowing time) - a mini-small period of time between the movements of two particles through the detector, when the signals from them have not yet superimposed on each other. At smaller times, the signals will be superimposed, and two particles will be recorded by the detector as a particle.
4. **Dead time** - the time at which the detector that registered one particle manages to return to its original position.

We have done research on radiation and its detection. Relying on the characteristics listed above, we decided to build our device based on the Geiger-Muller Counter. It is one of the most efficient and accurate ones, having spatial resolution ~1, Time resolution $>10^{-9}$ s and the dead time of 10^{-4} s. Another great advantage is that Geiger-Muller detectors are not sensitive to noise - it is possible to filter each event by signal amplitude. Apart from that, it's quite cheaper and more affordable than other dosimeters.

Fig.1 demonstrates the general overview of this kind of dosimeter:

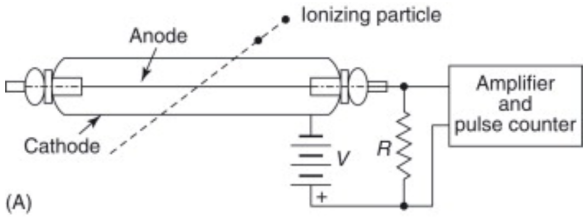


Figure 1

Overview of radiation detection using Geiger-Muller counter

If the voltage on the electrodes of the proportional counter is increased, then starting from a certain voltage, the pulse becomes independent of the energy of the primary particle, since the device begins to operate in a self-discharge mode, but this pulse increases with an increase in voltage. Such a region of voltage is called a Geiger region. The tube we are using for our device is operating in this mode. It is a gas discharge gap with a highly heterogeneous field, having a thin wire stretched along the axis of a cylindrical tube acting like a capacitor.
This electrode is enclosed in a sealed reservoir filled with gas at a pressure of 100-200 mm Hg. A voltage of several hundred volts is applied to the electrodes (in our case 440). When an ionizing particle enters the counter, free electrons appear in the gas that begin to move to the anode filament. This is how the electronic avalanche is born. Near the anode, where the intensity of the electric field is very high, the electrons formed as a result of primary ionization accelerate so much that they themselves begin to ionize the gas, intensifying the previously formed avalanche. The newly emerged electrons themselves can gain sufficient velocity to in turn cause new ionization and further enhance the electron avalanche. As a result, an independent - crown - discharge occurs, spreading along the thread. The discharge is terminated by using a gas mixture in the meter, consisting of an inert gas with a small amount of polyatomic gas vapors and haloids (Argon, in our case).

Dose

A Dosimeter counts the "dose" of radiation, it does not have a unit. X-rays are a unit of exposure dose, the "amount of radiation," which is expressed by the number of ions formed in dry air. At a dose of 1 R in 1 cm3 of air, 2.082e9 pairs of ions are formed. The main concepts in radiation measurements are dose and power. Dose is the number of elementary charges in the process of ionizing a substance, and power is the rate of dose formation per unit time. In the SI system, the exposure dose is expressed in pendants per kilogram, and this is connected with X-ray by the equation: 1 Cl/kg = 3876 R

Counts Per Minute

For the device to show newest and most accurate pulse quantity per minute, we are using the **CPM algorithm** developed by us:
We have a 60 element array, and in every minute, device is summing all the elements in array and sending them to the server and showing on the screen (handheld option). Array is filled like this: Every second it is moving on the new index and is storing pulse quantity received in that particular second there. When it fills (gets till 60) it goes back and starts over.
The advantage of our algorithm is that we always see exact quantity of the real counts per minute in our database, however, if a hazard happens, as the array is refreshed on every pulse (sometimes this time is even smaller than the second considering our tube's dead time), owner can get this information as fast as possible.

Hardware

Description of Schematics

Hardware Block Diagram

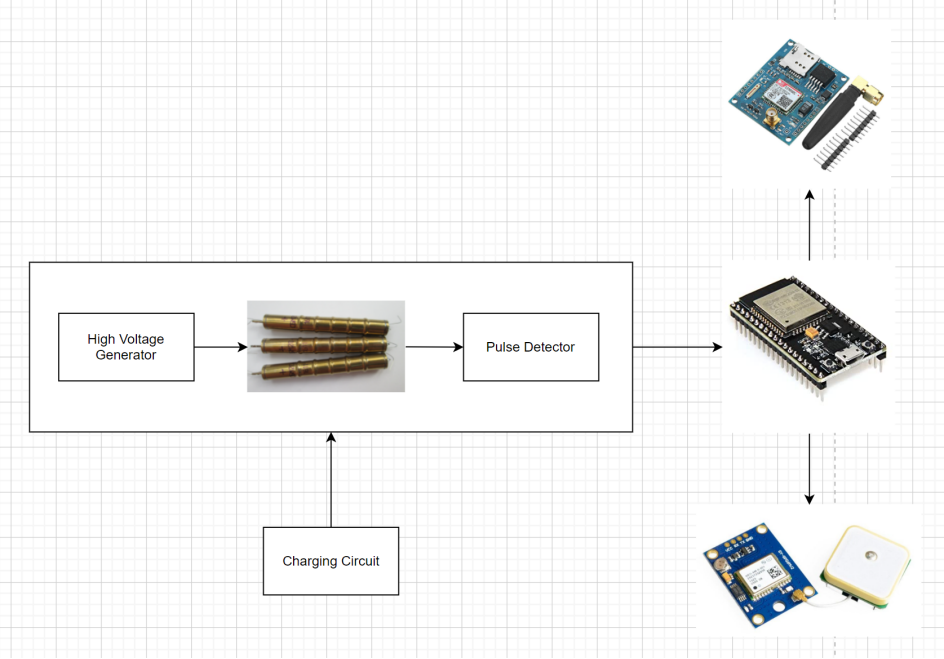


Figure 2

On the Figure 2 above one can observe block diagram of the hardware we used in the project. Our device consists of the High Voltage Generator schematics, Geiger-Muller Tube, Pulse detector which receives pulses from the tube and amplifies them in a way we can analyze them, ESP32 Microcontroller and GPS and GSM modules.

High Voltage Circuit

The heart of our circuit is the MC34063 chip, which is a DC/DC converter that is used to obtain the necessary high voltage from a low battery one. Its main advantage over simple microchip NE555 or similar pulse generators is that it can control the output voltage and adjust the parameters of capacitors and resistors to make it stable.

We used a power MOSFET (IRF840 model) which has a big drain-source breakdown voltage and works on high frequencies. It is getting pulses and working as a switch. We used a 10mH coil to feed our circuit with current. Next step was to choose a timing capacitor, which also controls frequency, makes circuit inaudible and stabilizes voltage on the output. We used a diode and pnp transistor to defend our IC from the charges collected on the gate of MOSFET. For diode, as we were working on high frequency, we used schottky diodes. It, together with the pnp transistor ensures the safety of the remaining circuit. Lastly, we used a voltage divider to provide negative feedback for the IC and we used a capacitor to stabilize voltage on the tube.

Pulse Detector

As the heart of the pulse detector we used the microchip LM258N. It has a low supply voltage and can handle input voltage up to 32 Volts. In our case, this is unnecessary because we are lowering the voltage incoming from the tube using the voltage divider by the resistors to 5 volts, but just to ensure safety the chip which can handle higher voltages was great. LM258N in the pulse detector is put into the comparator mode, which means that it receives two voltages and outputs a digital signal indicating the larger one. In our case - the pulse. The pulses are then fed to buzzer through the 2N222A transistor, so that it makes the noise every time the radioactive particle hits the tube. Our final circuit is accessible in Appendix A

The output voltage and on capacitor tested on MultiSim can be observed on Figures 3 and 4:

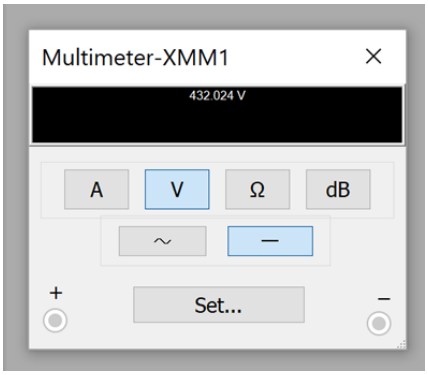


Figure 3

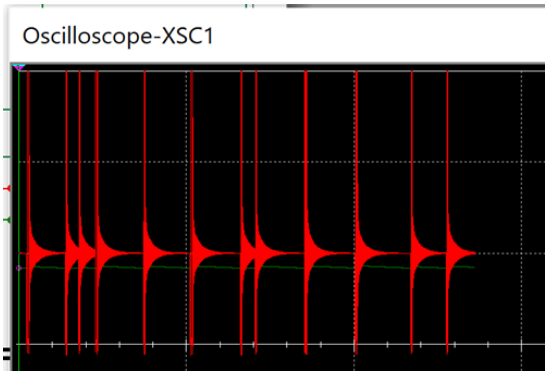


Figure 4

The output voltage tested in the lab can be observed on the Figure 5:

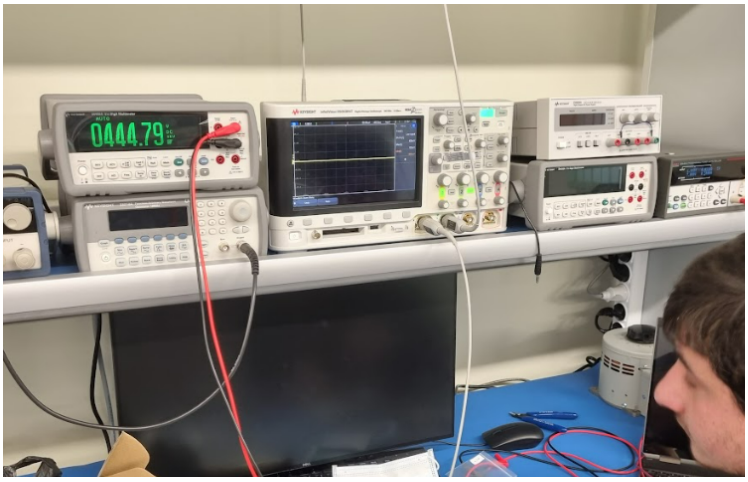


Figure 5

PCB

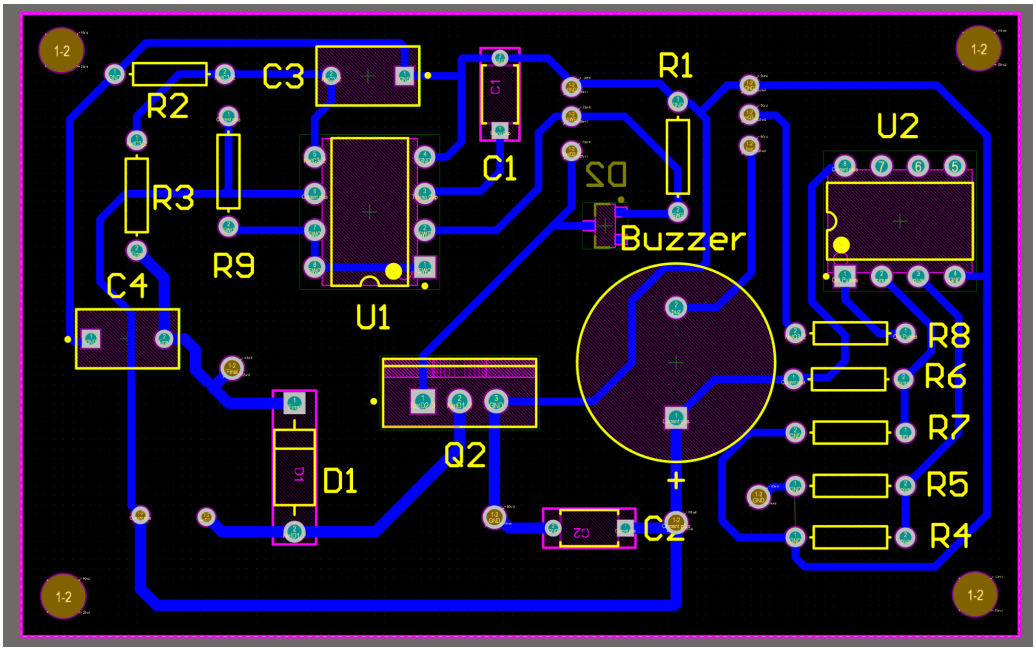


Figure 6

We have designed a one-layer PCB, so that we could print it easily. On our PCB we have placed our High Voltage Generator and Pulse detector. We have also inserted a buzzer onto it, so that when the tube “registered” radioactive particles, it makes a sound. This feature is perfect for the handheld version, as it warns the owner about the threat. On the automated device version there will be no buzzer, but a small resistor instead of it. In the Via between C4 and D1 is inserted the

+ terminal of the tube and in the Via next to the R5 is inserted the - terminal of the tube. Near the D1 are pads for the coil L1. The Vias around C2 are GND and ~+6.3 Volts correspondingly.

Description of Main Components

SMB-20

We chose the tube SMB-20. They have a low supply voltage, high speed and excellent output parameters. The body is made of stainless steel with a thickness of 50 μm. Geiger meters with metal gas containers are exceptionally good at detecting charged particles, as the particles need not even make it into the gas. They can ionize the metal casing, and the electron finds its way into the gas, triggering the electron cascade.

The tube reacts to hard beta and gamma radiation. Also, it reacts to α emission, but only at a distance of up to a few centimeters.

Characteristics:

Operating voltage range: 350-475 V

Range of recorded gamma radiation exposure dose rates: 0.004-40 mcr/s, 0.014-144 mp/h

Gamma sensitivity Ra226: 29 imp.s/mr/h

Gamma sensitivity Co60: 22 imp.s/mr/h

Maximum allowable current: 20 μA

Tube capacity: 4.2 pF

Work resource: 2x1010imp.

Recommended anode resistor: more than 5.1 MOhm

Operating temperature range: -50... 70 ° C

Operation area: ~13.8 cm^2

Figure 7 shows the Plateau plot with net pulse count in a minute at a certain voltage applied to the SMB-20. The Figure 8 indicates that the acceptable voltage range for the tube is 350-500:

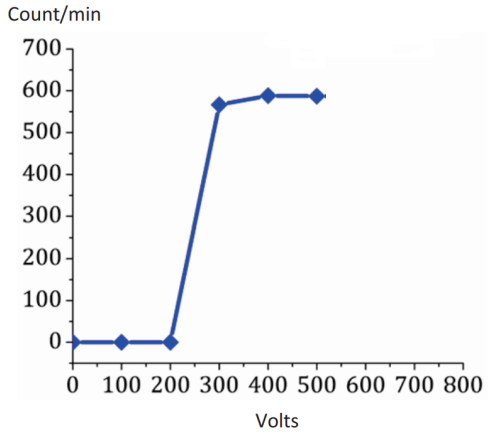


Figure 7

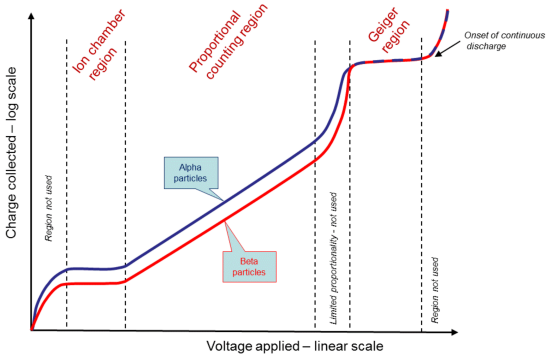


Figure 8

When beta or gamma particles hit our device, the gas that is inside the SMB-20 sensor ionizes, as a result of which a pulse is formed at the output, which enters the transistor amplifier and a click (handheld option) is heard in Buzzer. Outside the intense radiation zone, sounds follow every 1 - 2 sec.This indicates a normal, natural radiation background.

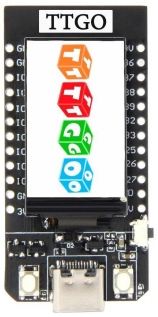
When the dosimeter approaches any object with strong radiation (a plate covered with uranium paint), clicks will become more frequent and can even merge into one continuous crack.

In the automated options, dosimeters which are placed in the desired locations are either powered only by batteries, or by the solar panel. They are sending regular information on the server and if a hazard is detected (radiation level suddenly rises) the device is sending warning messages.

ESP32

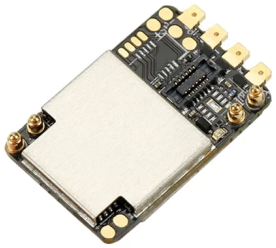
Pulses which are generated by the tube are being filtered and fed to the ESP32 which is connected to the IoT server using the MQTT protocol. ESP32 receives information from our device and sends it to the IoT cloud (we have it both: on DigitalOcean and Raspberry Pi4), which visualizes information using Grafana and sends data together with visualized tabs on our web-server.

TTGO



The TTGO LoRa32 SX1276 OLED is an ESP32 development board with a built-in LoRa chip and an SSD1306 0.96 inch OLED display. We are using it in our handheld version of the device, it automatically outputs the Counts Per Minute (times tube “registeres” radioactive particles)

GPS



We are using the GPS - satellite-based radionavigation system in order to track the location of our devices. The location in then displayed on the map on our website. We established its communication with ESP 32 using NMEA protocol, in which the data is transmitted in ASCII strings or "sentences" from one "talker" to multiple "listeners" at a time.

The GPS receiver measures the time of arrival (according to its own clock) of minimum four satellite signals. From the time of arrival and the time of transmission, the receiver forms four time of flight values, which are (given the speed of light) approximately equivalent to receiver-satellite ranges plus time difference between the receiver and GPS satellites multiplied by speed of light, which are called pseudo-ranges.

The receiver then computes its three-dimensional position and clock deviation from the four time of flights and displays the location in the NMEA monitor. We have googled the location output by the program and it appeared to be extremely accurate. Details of us using the NMEA protocol are given on the Figure 9:

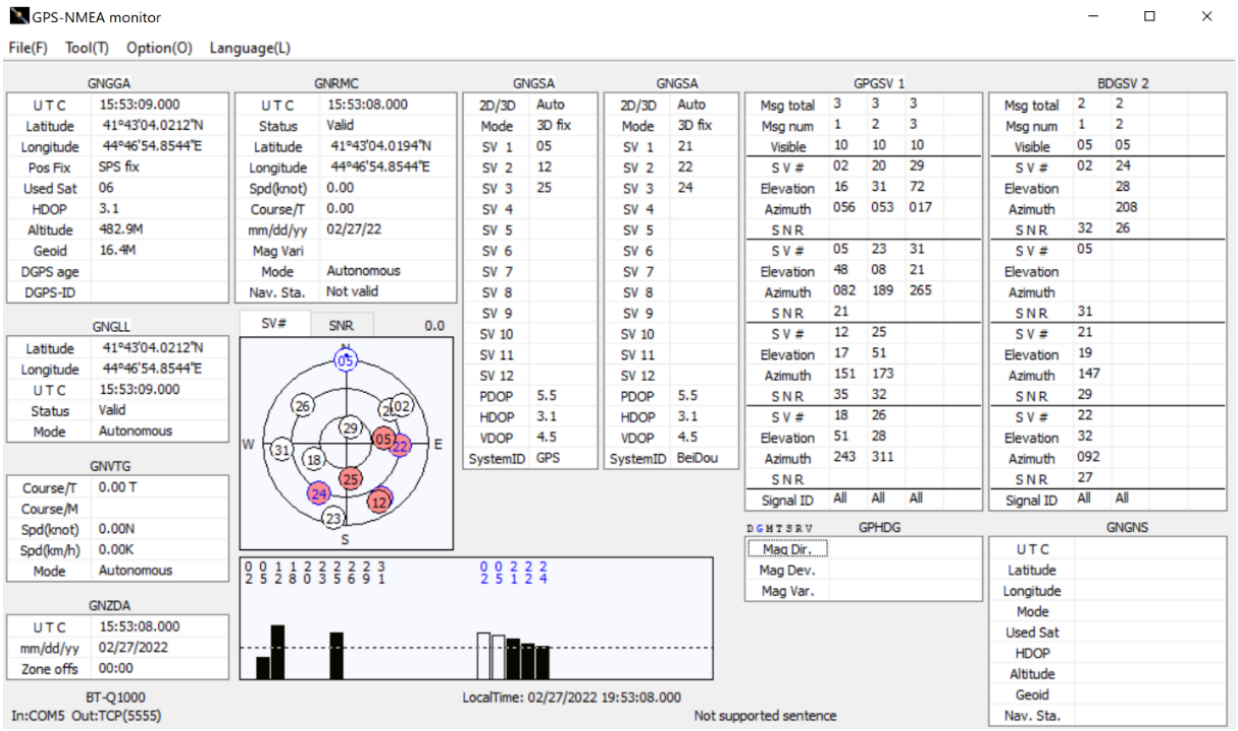


Figure 9

GSM

If our automated device which is placed far away form the owner generates a hazard, our ESP32 generates a response which is sent to owner using GSM Sim800C. It's a device that can send a message/make a call at any distance.

We tested its range and found out that we can send SMS from Tbilisi to Chicago, which can be observed on the Figure 10:

```
AT+CMGS="+12245229997"
> Greetings from Georgia
+CMGS: 8

OK
```

Figure 10

Our GSM is powered with a battery and is programmed through the ESP32. It is communicating with the server using the AT commands.

First, we check the signal quality of Signal with the command: AT+CSQ.

Next, we check if the sim card is inserted: AT+CCID.

The following step is to configure GSM so that it will send text SMS: AT+CMGF=1.

After that, we should tell the GSM cell phone number where the SMS should be sent AT+CMGS=+995xxxxxxxxx.

On the Figure 11 you can observe how we sent AT commands to GSM from a computer using Python.

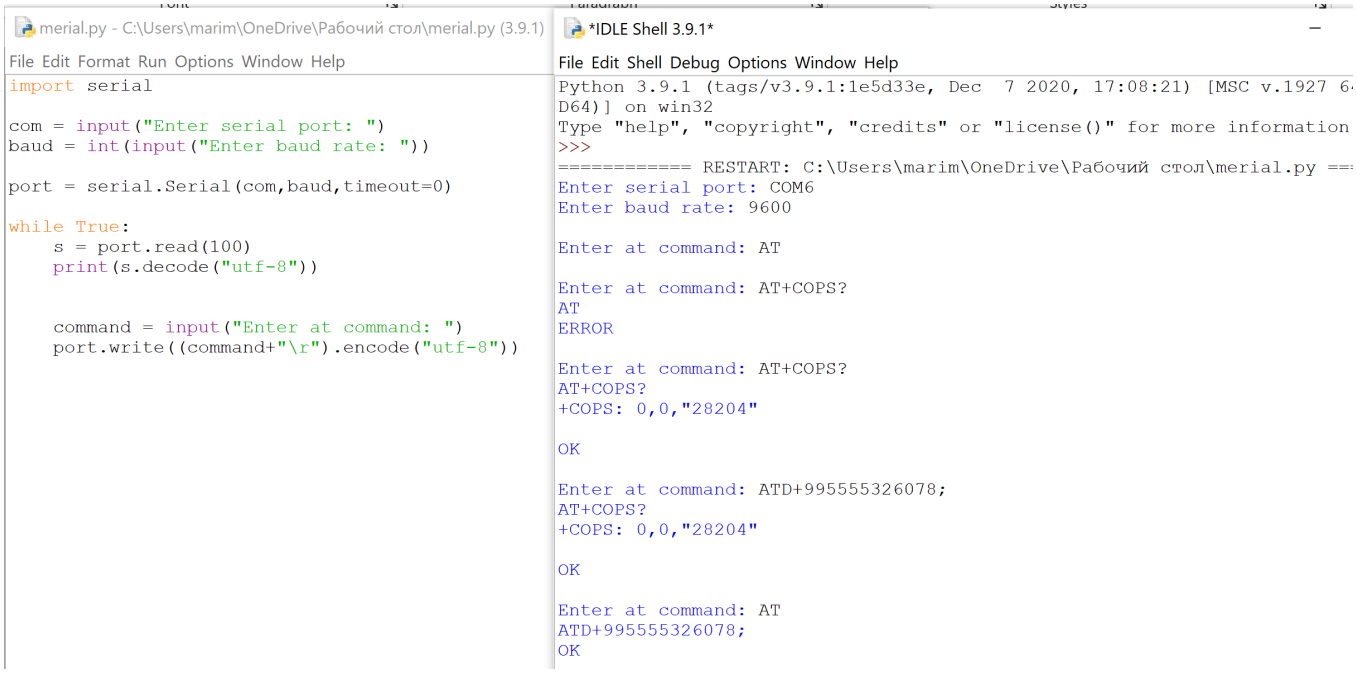
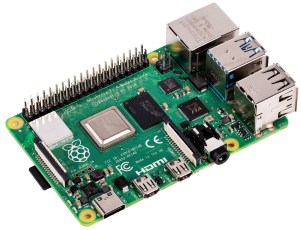


Figure 11

Raspberry Pi 4



We are using Raspberry Pi instead of the DigitalOcean. If the internet will not be available at the location, or the owner wants to secure his data without putting it out in the network, he can run everything on the Raspberry. It is working on the Linux Ubuntu OS.

Charging Circuit

In order for our circuit to be energy efficient (92% efficiency) we are making it work on one lithium-ion battery 3.7V. This is not enough for the circuit to function properly, thats why we added a buck-boost converter after the battery. It boosts 3.7 V to 5 V and makes input voltage more stable. Model of our converter can be observed on the Figure 12:

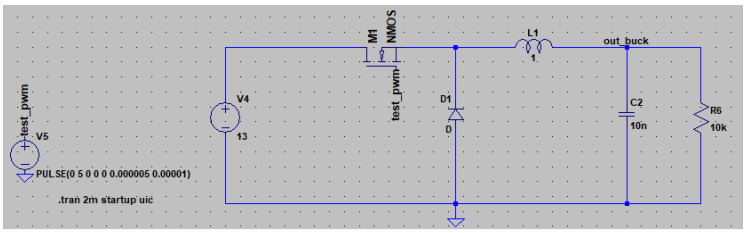
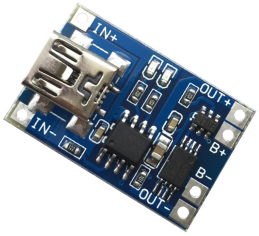
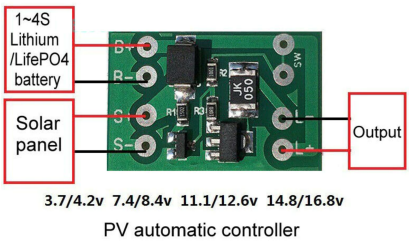


Figure 12



To charge the battery we are using a charger module with current protection. We chose this one, as its based on the TP4056 charger IC and DW01 battery protection IC. Furthermore, when the battery voltage drops below 2.4V the protection IC will switch the load off to protect the cell from running at too low of a voltage – and also protects against over-voltage and reverse polarity connection.



For the autonomous device installed in another location, we are using a small solar panel in order to keep the device charged. For the solar panel we need an MPPT controller, which regulates the charge of our battery bank.

The module also employs various protection functions, such as battery/solar panel reverse polarity protection, output over temperature and over current/short circuit protection, which greatly improves the stability and safety of the system.

Photovoltaic effect in solar cells

In most photovoltaic applications the radiation is sunlight, and the devices are called solar cells. In our case of a semiconductor p–n junction solar cell, illuminating the material creates an electric current because excited electrons and the remaining holes are swept in different directions by the built-in electric field of the depletion region. The AC PV is operated at the non-equilibrium conditions and AC current is also produced when a flashing light is illuminated at the interface. The AC PV effect does not follow Ohm's law, being based on the capacitive model that the current strongly depends on the frequency of the chopper, but voltage is independent of the frequency. The peak current of AC at high switching frequency can be much higher than that from DC. The magnitude of the output is also associated with the light absorption of materials.

Final Block Diagram of the autonomous device Charging Circuit:

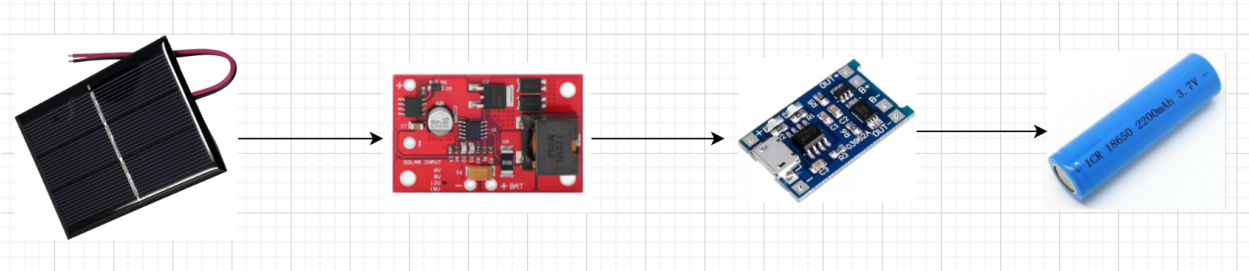


Figure 13

Software

Software Block Diagram

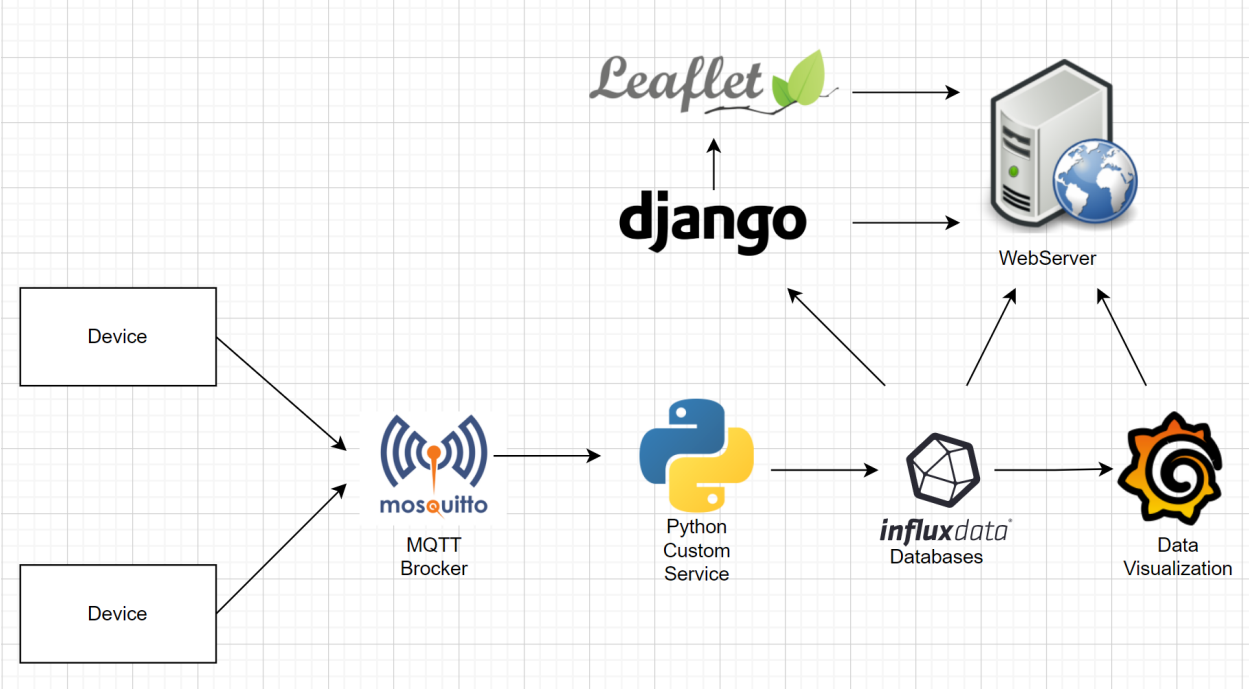


Figure 14

IOT Server

Our project uses IoT to connect our device, GSM and GPS modules and our web application. The modules and IoT system exchange information via the Message Queuing Telemetry Transport (MQTT) protocol. This protocol operates at the TCP/IP layer of the internet, supporting bi-directional communication between devices. The devices connected with the MQTT protocol can publish and subscribe to information and it is an excellent option for sending high volumes of sensor messages to analytics platforms and cloud solutions.

MQTT protocol distinguishes between two kinds of network elements: a broker and a client. MQTT broker is a server or cloud that receives messages from publisher devices and routes them to the subscriber clients. Clients can be any devices that are connected to MQTT broker with MQTT protocol. The information exchanged by devices is labeled with topics. Devices can

publish data to a particular topic, sending a message to the broker a control 6 message (Topic). The broker distributes that information to devices that have subscribed to that topic. At first, clients should establish a connection with the broker.

Mosquitto Broker

For the broker, we are using Mosquitto as it is lightweight and is suitable for use on all devices from low power single board computers to full servers (we are using both). Mosquitto is receiving MQTT messages coming from sensors and is sending them to connected clients. Configuration of our Mosquitto Broker is given in the Figure 15:

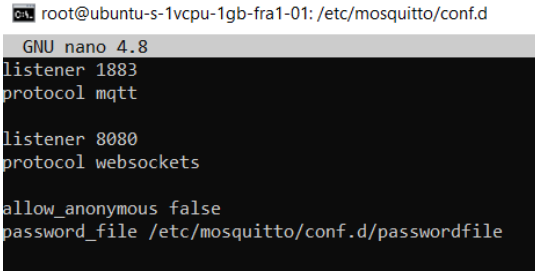
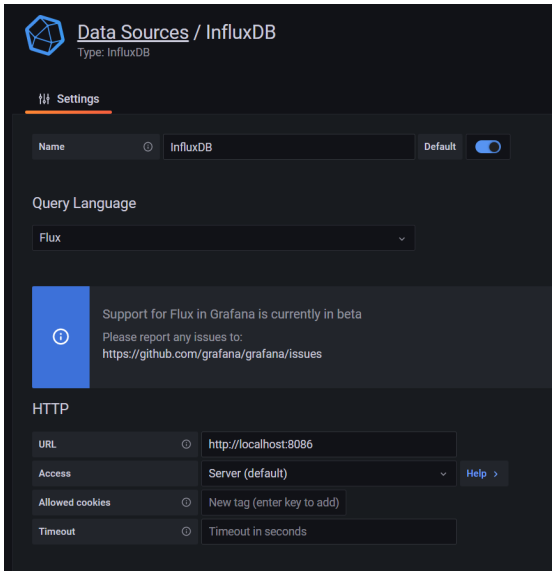


Figure 15

Python Custom Service

Our python custom service registered itself in the MQTTbroker as the client and is receiving messages from the bucket sent from the ESP32. Then, it is parsing received data and is storing it in the InfluxDB database. One can find our codes in the folder attached to the report. Path is indicated in the Specifications part given below.



InfluxDB Database

InfluxDB is perfect for real-time applications for analytics, IoT and cloud-native services. It is fast, elastic, serverless real-time monitoring platform, dashboarding engine, analytics service and event and metrics processor.

Django

Django is a web-server and we are using it to host our website. Together with that, it is getting the data from the InfluxDB Databases and feeding that data to the Leaflet.

Leaflet

Leaflet is an open source JavaScript library used to build web mapping applications. Figure 16 shows how does the map look like on the website.

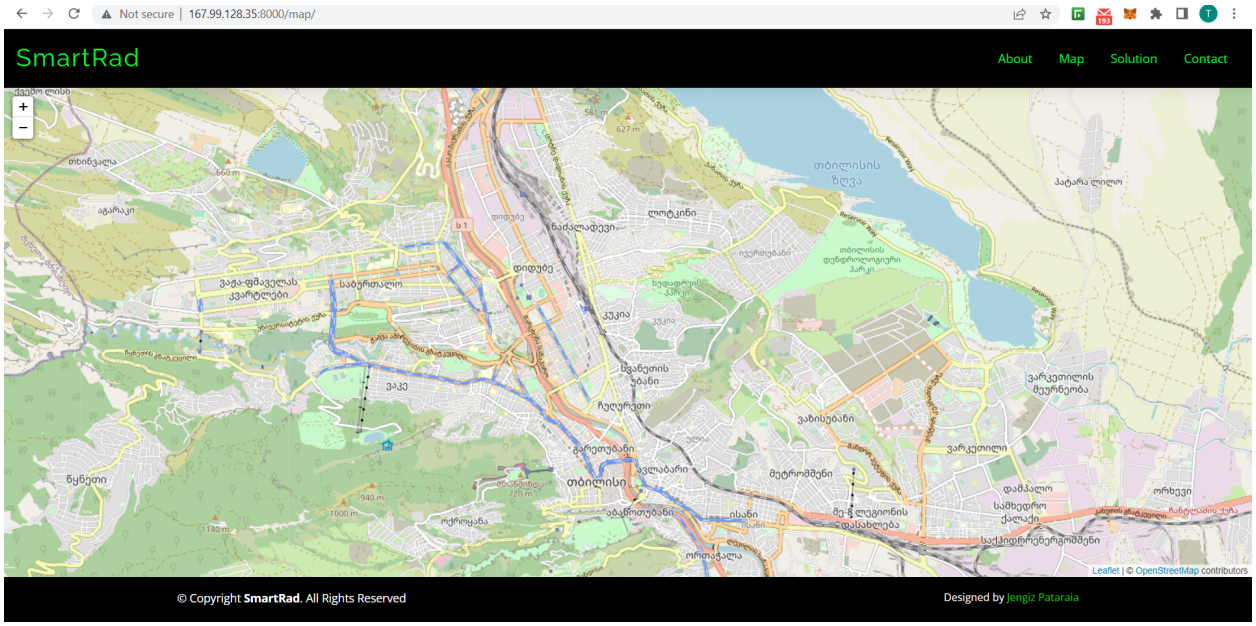


Figure 16

When a device with GPS is turned on, the droplet of its location appears on the map, together with Counts per Minute of the device.

Grafana

We are using Grafana to visualize data. It takes data from InfluxDB database and is visualizing it on the graphs showing the radiation level. It can be zoomed in and out and also the time interval can be chosen if someone is interested in particular data. Another feature is that one can filter out

devices by location. One device can be chosen from the map and Grafana will show only that devices graph. Figure 17 is showing how does Grafana function:

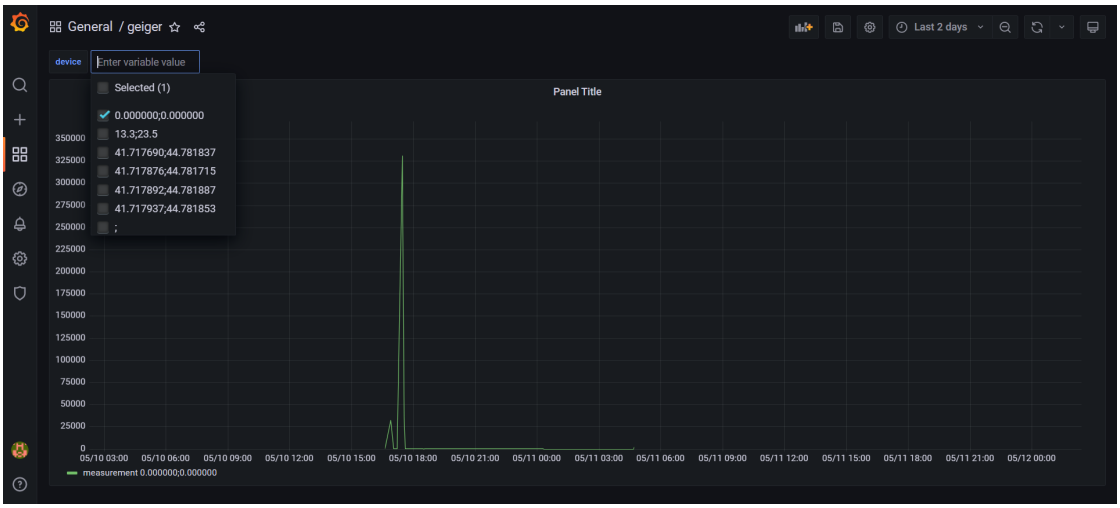


Figure 17

WebSite

Our website is hosted on the DigitalOcean platform and anyone interested can find it. We have designed the Frond-End of the website using HTML, CSS and JavaScript. The website has an about page, the map, the solution (which describes the process of making it) and the contact page for the potential customers. The part of the About page is observable on the Figure 18:

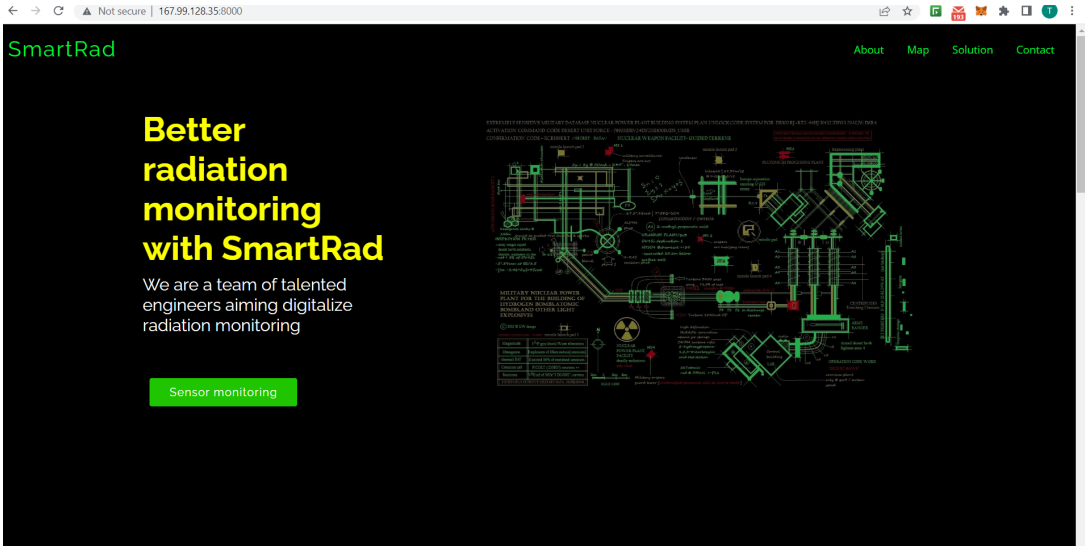


Figure 18

Final Assembly

Figure 19 shows how does the final assembly of the handheld option looks like without the box:

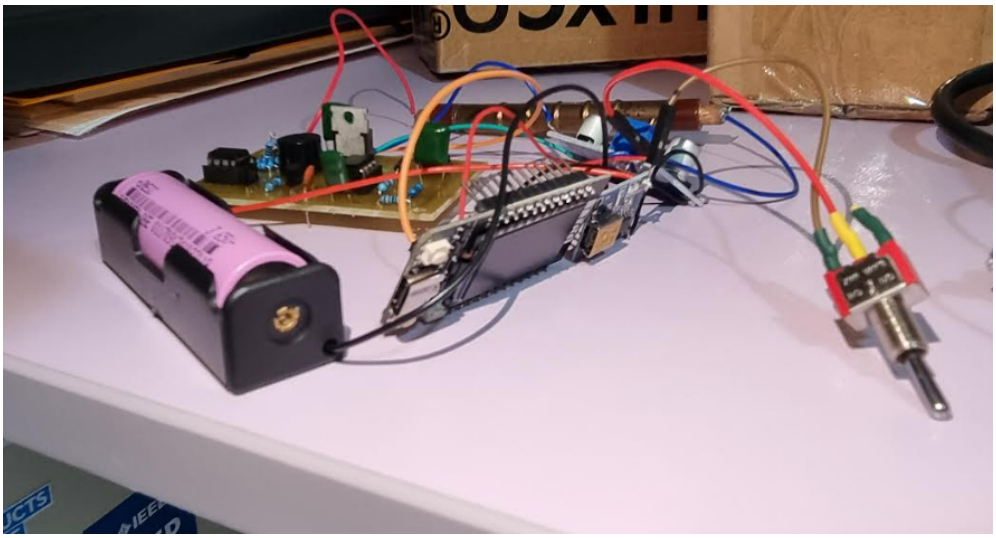


Figure 19

Figure 20 shows how does the final assembly of the handheld option looks like with the box made of wood:

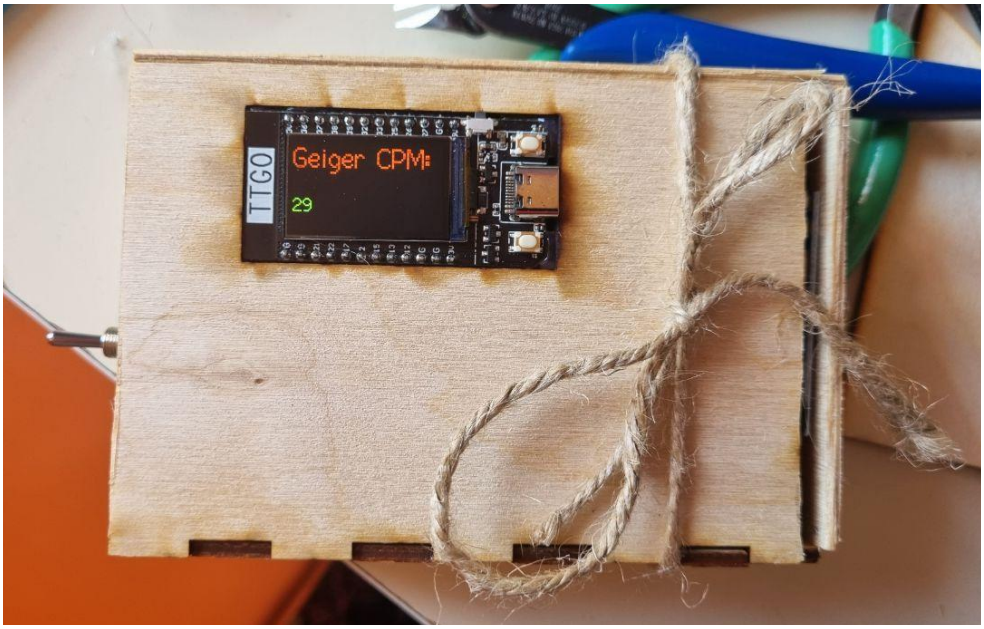


Figure 20

The box was designed in the CorelDraw and cut out from wood manually by us.

Testing

Geiger meters are excellent for qualitative radiation detection in that they give no information concerning the energy or type of radiation. However, they do provide a dependable method for measuring the volume of radiation, i.e. the number of gamma rays and beta particles, collectively referred to as events, per unit of time. For each radiation event, there is some probability α that the event makes contact with a gas molecule and starts the detectable electron avalanche. For a tube with known α, it is possible to extrapolate measurements for an accurate estimate of the true radiation.

The radiative flux of a source is given by:

$$\Phi = \frac{P}{4\pi(r+\frac{1}{\sqrt{4\pi}})^2} \quad (1)$$

Where P is the activity of the source and r is the distance from the source. A factor $\frac{1}{\sqrt{4\pi}}$ is included to adjust for smaller r. Multiplying the flux by the cross sectional area of the Geiger tube gives a theoretical value for events per unit time.

Using a Poisson distribution based model, we adjusted our theoretical values to account for the Dead Time of a tube. Consider a length of time , the length of a time of T_{DT} as the length of a tube's deadtime. Say that k events pass through the tube during that time, and there is some probability α that and event ionizes a gas molecule. Figure 21 illustrates the process.

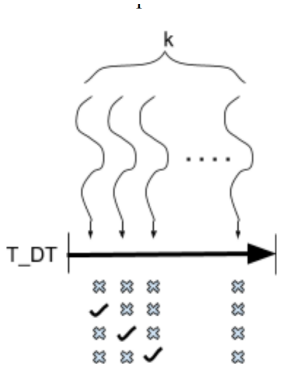


Figure 21

Within this time period, as each event arrives, it can be detected or not detected. Detection leads to a deadtime and failure to detect the remaining events. The probability of completely missing all k events is $(1 - \alpha)^k$. Otherwise, at most one event will be detected during this time period. Modeling the radiation source as a Poisson distribution allows us to account for the possibility of any k value. For some time T_{DT} the probabilities of detection are

$$P_0 = \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} (1 - \alpha)^k \quad (2)$$

$$P_1 = 1 - P_0 \quad (3)$$

where λ is the flux adjusted for the time T_{DT}. Multiplying P_1 by $\frac{60}{T_{DT}}$ gives the expected value

per minute accounting for the deadtime. The series collapses nicely giving final expression for counts per minute is given by:

$$CPM = \frac{60}{T_{DT}} (1 - e^{-\lambda - \lambda(\alpha - 1)}) \quad (4)$$

CPM in this case is the empirically measured value with known error. With some algebraic manipulation, we can solve for efficiency.

$$\alpha = -\frac{1}{\lambda} ln(1 - CPM(\frac{60}{T_{DT}})) \quad (5)$$

Once α is known, we need only divide future background measurement to get an estimate of the actual radiation present.

We have tested our device without any radioactive sample in the background and then with a glass (185gr.) covered in coloring made from Uran-238. The coloring is considered to be 2% of the mass, so the radioactivity can be said to be 1.23μCi.

We recorded the pulses per minute for two situations: without the sample and with the sample put in 15 cm. Data can be observed in the Appendix B. Then, we calculated the theoretical events per minute and divided the measured by the theoretical for an α value. Inserting the data in

MATLAB (Appendix D), we received this histograms indicated on Figure 22, on which you can observe data with and without sample:

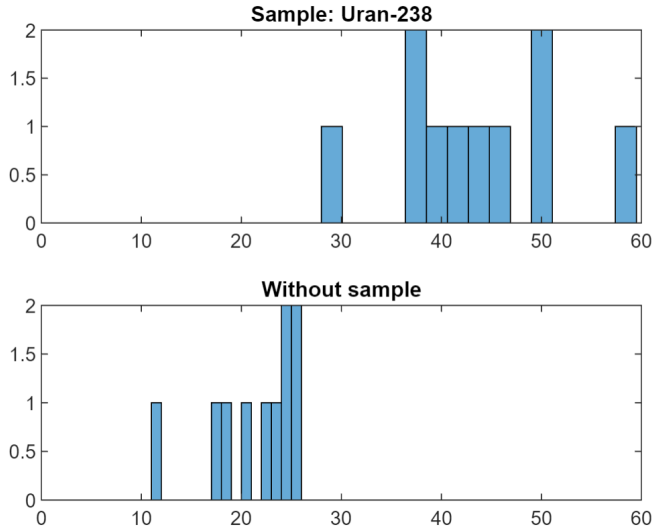


Figure 22

At 15 cm, we considered the deadtime issue to be negligible. The probability of missing one or more due to deadtime, in some time frame equal to a dead time, is given by:

$$P_{missing \geq 1} = \sum_{k=2}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \quad (6)$$

Where λ is the average arrival rate (total flux) for this time period.

Over the course of a minute, our desired sample time, we found the probability of missing one or more radiation event due to deadtime to be 0.4530. On its face, this value is not negligible. However, if we evaluate the poisson distribution at k=2 , we find the probability of missing just one event in a minute to be ~0.025. The probability of missing a significant number of events is unlikely. Even though the probability of missing one event is somewhat likely, a single event is negligible relative to the random variations in radiation. Relying on our code, α value with 95% confidence interval is :

$$\alpha = 0.040443523227688 \pm 0.008993521741057$$

The efficiency is close to 100%.

Conclusion

Our project addressed the 21st century's one essential issue: Radiation detection and Warning citizens. Our primary purpose was to make those two commodities affordable and readily available to everyone, living in a world where radiation has become part of our life.
We endured safety by implementing hardware and software modules that can monitor radiation, detect hazards (excess radiation) and generate appropriate responses to limit the damage caused by it and protect people. One of the most important accomplishments of our project is the whole automatisation of the device, by utilizing the Internet of Things (IoT) and charging independence by using solar panel. To our counter design, we have added modules that generate information about the safety of the area around them and send that information to the IoT cloud where appropriate response is automatically generated and the data is visualized. As a result, our project has turned a simple dosimeter into a smart digital dosimeter, data of which can be accessed from every point of the world. The databases storing the data can be used to find the correlation between radiation level and other parameters.
This was our first project which means that it is not perfect, and certain things could have been done better. However, we are certainly going to develop it further. First thing to change is implementing an algorithm that will make the device specify radiation levels and ranges indicated by WHO. Another point to ameliorate is PCB -it would be great to put all components on the PCB so that we did not have to use all those jumpers which made a mess. We would also design a controlled buck boost converter on our own, because fabricated one could not handle all the modules connected together.
Our future plan is to solve all the issues and also design a small UAV which will be equipped with our device. As it already has GPS, it would be great to have an ability to tell UAV the exact location where it should go to measure radiation level there, and it would land on the exact coordinate. Another great part of it is that scientists can measure radiation in places not accessible for people because of extremely high radiation level there (such as Chernobyl).
Overall, we attempted to generate a solution to the issue of radiation detection. We accomplished this by implementing an IoT system to generate information about the state of the radiation level

and added peripherals, allowing us to monitor radiation level even from the faraway point. Our project turns a simple, ordinary dosimeter into a smart, modern and digital one.

Specifications

All materials that are required to replicate our project accessible in the folder we are sending together with this report.

In the folder, you can see a section named PCB. It contains the .PrjPCB and Gerber files for our PCB. We designed one-layer and two-layer options, both are accessible. You can also access schematics of MC34063 in the folder. As it was not included in Altium designer library, we had to create it manually.

In the folder, you can see a section named Invoices. It contains bills of all of the components that we purchased.

You can see PowerPoint presentations, video demonstrations, duties chart, poster, and project proposal.

In the Source codes folder, there are all the codes that were used for our project:

- Folder named MATLAB: contains a code for analyzing data caught by the SMB-20 tube
- Folder named Geiger_TTGO: contains .ino code for the device with option of TTGO (handheld one)
- Folder named Geiger_GSM: contains .ino code for the automated device, which is using the GSM to hotspot an internet
- Folder named Geiger_Wi-Fi: contains .ino code for the device, which is working on the internet connection (we are using 4G/5G modem for it)
- Folder named Libraries: contains all of the .h libraries we have used while developing our project
- Folder name MQTT_Connect: contains Python custom service code
- Folder name WebServer: contains the Django code for the webserver and the HTML, CSS and JavaScript (Leaflet library) for the website

Team Member Contribution:

Mariam Gigauri

My responsibilities in the process of developing SmartRad was to design the schematics for the high voltage generator and pulse detector, and choose all the right components for this circuit. Together with schematics, I was also working on the PCB for our project. I designed it in the Altium designer. It took me 2 tries to make a good version. I will certainly ameliorate it in the future!

One of my responsibilities was to research the particle physics and understand the radiation phenomenon, in order to apply the theoretical knowledge to the experimental one and compare theoretical values to the experimental (which were generated by our code).

I created a code which counts pulses and outputs the sum to the serial monitor every minute using the ESP32. I have also written the program on the TTGO which counts and outputs quantity of pulses.

For my further explorations I was running the program for 15 minutes without the sample around it and I re-run it after for another 15 minutes, but with the sample of glass covered with Uran 238 coloring put in 15 cm from it.

I implemented the MATLAB code which shows the histograms of our device without the sample around it and with the sample, and those histograms are quite comparable and easy to understand. My code also generated the probability of error, which was relatively small.

My next responsibility was to connect GPS and make it work. I explored the NMEA protocol and first have powered the module with the FTDI module, then with Atmega (as I was acquainted with it), then with Python and lastly with ESP32.

I connected the GSM and explored the AT commands in order to “talk” to the GSM. Just like in the GPS case, I first run it on the FTDI, then on the Python and lastly on the ESP32.

I was in charge of managing the project schedule, team presentations, project report, poster and final assembly of the device.

Tengiz Pataraia

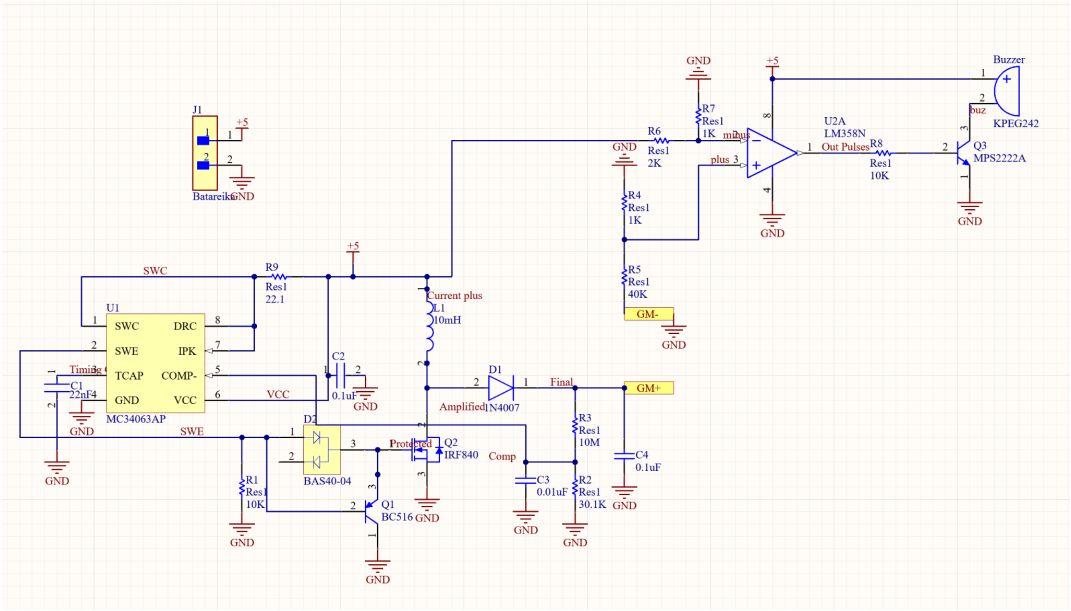
My responsibilities in the process of developing SmartRad was to mix software and hardware. I have summed up the final codes for all 3 options of our devices.
I came up with the CountsPerMinute algorithm, which refreshes the array of stored data with every pulse.
My next responsibility was building the website. To do the Front-end part, I learned the HTML, CSS and JavaScript in order to implement it. Then, I explored the Leaflet, to which I connected the data coming from GPS case, so that the device location could be seen on the map.
As for the Back-end, I have set up the Django webserver and connected it to the Leaflet.
To connect this all with the ESP32 code, I have built the python custom service, which registers itself in the Mosquitto as the client and is receiving messages from the bucket sent from the ESP32. Then, it is parsing received data and is storing it in the InfluxDB database.
I have also set up the DigitalOcean - I installed Ubuntu on the remote computer and have set up the InfluxDB database and Gafana on it. InfluxDB is storing all the data coming from the device into databases, and the Grafana visualizes the data onto graphs, which is quite convenient when exploring radiation level. In the InfluxDB databases, I created databases, in which weekly, monly and yearly radiation average can be calculated. In addition to that, Grafana is also outputting the desired time interval onto the graphs and user can see whatever information he is looking for. Connecting that all to Leaflet and GPS, Grafana also can filter out data by locations.
I did the same on the Raspberry Pi - I manually configured the Ubuntu OS on my computer and then have downloaded it onto the microSd card of the Raspberry. Then, I did everything like on the DigitalOcean. Raspberry will give us ability to set up our server wherever there is no internet connection.
My other responsibility was to build up an application, which enables the user to connect the device to the bluetooth (handheld option) and observe the data on his phone.
Apart from the software, I was in charge of parts procurement, solving all the technical issues and managing the budget.

References:

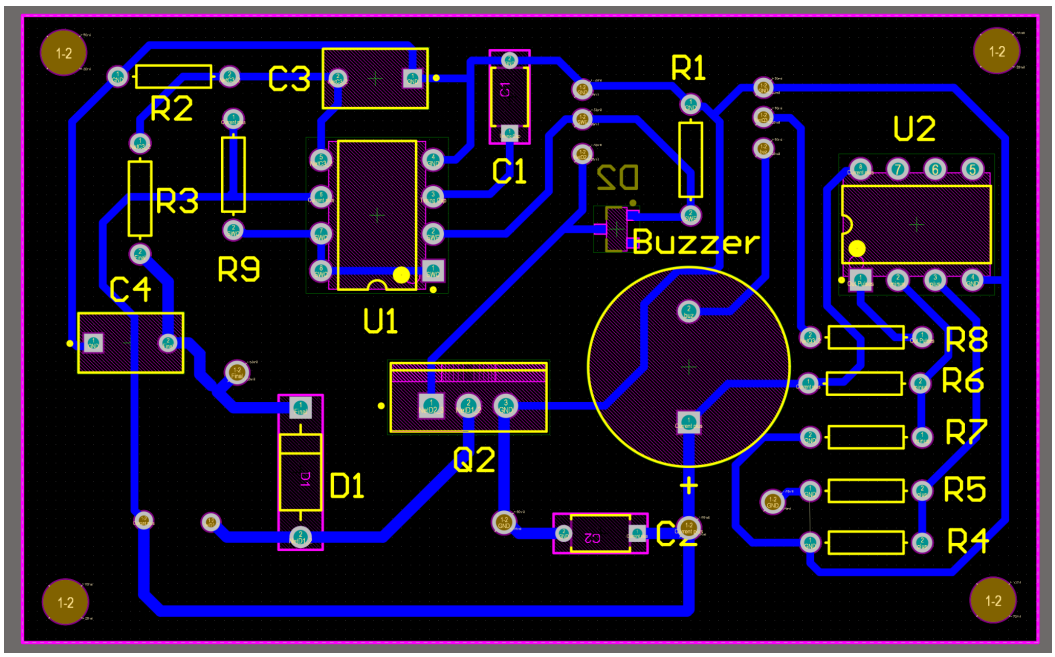
1. At Commands Reference Guide (2006). Retrieved from: https://www.sparkfun.com/datasheets/Cellular%20Modules/AT_Commands_Reference_Guide_r0.pdf
2. Turner, J. “Atoms, Radiation, and Radiation Protection”
3. Sivukhin D.V. General course of physics.Volume V - Particle Physics
4. Kiger, P. “How do Geiger counters work?”, https://science.howstuffworks.com/geiger-counter.htm
5. Gakstatter, E. “What exactly is NMEA?”, https://www.gpsworld.com/what-exactly-is-gps-nmea-data/#:~:text=Today%20in%20the%20world%20of,and%20match%20hardware%20and%20software.
6. NMEA reference manual. Retrieved from: https://www.gpsworld.com/what-exactly-is-gps-nmea-data/#:~:text=Today%20in%20the%20world%20of,and%20match%20hardware%20and%20software.
7. TinyGPS++ Library for the ESP, http://arduiniana.org/libraries/tinygpsplus/
8. InfluxDB OSS 2 Documentation, https://docs.influxdata.com/influxdb/v2.2/
9. Leaflet documentation, https://leafletjs.com/reference.html
10. Clark, J. “What is IOT?”, Retrieved from: https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/

Appendices

Appendix A - Schematics



(a) Schematics of the High Voltage generator together with Pulse detector



(b) PCB for the schematics above

Appendix B - Excel Table for the data without and with sample

Without Sample	With Sample in 15cm
26	50
11	39
24	59
26	37
23	43
22	46
17	51
20	28
24	37
18	42

Appendix C - MATLAB Code for trimming rows

```
function out=trimRows(rowMat)
%trims NaN of column vector
out=rowMat;
for i=1:length(rowMat)
if isnan(rowMat(i))
out(i:end)=[];
break
end
end`
end
```

Appendix D - MATLAB code for Analysis

```
clc;
format long;
%% Readings from the Excel, filled with data received on the ESP32 in the time range of 15
minutes
Geiger_Raw_Table = readtable('Geiger_Data.xlsx','Sheet','Sheet1','Range','A1:B42');
```

```
Geiger_Raw = table2array(Geiger_Raw_Table);
%giving matlab data from columns
Geiger_without_sample=Geiger_Raw(:,1);
Geiger_Uran238=Geiger_Raw(:,2);
% trimming columns using trimRows function
Geiger_without_sample=trimRows(Geiger_without_sample);
Geiger_Uran238=trimRows(Geiger_Uran238);
%% Theoretical Values
r=[0 0.15]; %meters
P=1.23e-6; %Curies considering that the glass we tested is 185 gr and uranium coloring is 2% of
the glass.
countsPerCi = 4.468e10; %counts/s/Ci
flux = P*countsPerCi./(4.*pi.*(r+(1./sqrt(4.*pi))).^2); % counts/s/m^2
Geiger_Area = 0.01*0.1;
Geiger_flux = 60*flux*Geiger_Area; %counts/m
%% Analysis
Detected_Avg_Geiger=[mean(Geiger_without_sample), mean(Geiger_Uran238)]; %averages
Detected_STD_Geiger=[std(Geiger_without_sample), std(Geiger_Uran238)]; %standart
deviations
error_Geiger=[Detected_Avg_Geiger-2.*Detected_STD_Geiger;Detected_Avg_Geiger+2.*Dete
cted_STD_Geiger];
%% Theoretical Values with Poisson Correction
Geiger_DeadTime = 10e-4; %s dead time
Geiger_flux_per_DeadTime = Geiger_flux*(1/60)*Geiger_DeadTime; % lambda 02, flux per
dead time
%% Real Data Analysis
Error_Correction=sqrt(2*(Detected_STD_Geiger(2)).^2+2*(Detected_STD_Geiger(1)).^2);
Difference_Correction=(Detected_Avg_Geiger(2)-Detected_Avg_Geiger(1));
Geiger_Corrected_15cm=[Difference_Correction + Error_Correction;
Difference_Correction-Error_Correction]; %with sample-without sample
alpha=Difference_Correction./Geiger_flux(2);
Error_alpha=abs(alpha).*sqrt((Difference_Correction./Error_Correction).^2+(2*sqrt(Geiger_flux
(2))./Geiger_flux(2)).^2);
alpha_95_confidence=[alpha+ Error_alpha;alpha - Error_alpha]
nbins=15;
subplot(2,1,1)
histogram(Geiger_Uran238,nbins)
```

```
title("Sample: Uran-238")
xlim([0 60])
subplot(2,1,2)
histogram(Geiger_without_sample,nbins)
title("Without sample")
xlim([0 60])
```

Appendix E - Bill of Materials

Components	Price in $ - Purchased Abroad	Price in GEL – Purchased in Georgia
MC34063API IC (x10)	0.72$	-
IRF840 MOSFET (x10)	1.80$	-
LM358 IC (x10)	0.65$	-
BAS40 Diode (x40)	1.87 $	-
1N4007 Diode	0.77$	-
0.01uF Capacitor (x20)	1.08$	-
0.1uF Capacitor (x20)	1.62$	-
TO-92 pnp Transistor	0.95$	-
Sim800C GSM IREX	3.61$	-
Sim800C Quad-Band	6.93$	-
Li-Po Charging Module (x5)	0.92$	-
Solar Panel	7.65$	-
Neo 7M GPS	5.40$	-
MPPT controller	3.33$	-

0.5 Watt Resistors	4.86$	-
0.25 Watt Resistors	3.09$	-
STM32 Programator	8.62$	-
STM32	5.13$	-
WiFi router	25.45$	-
Bluetooth HC-06	4.62$	-
STM32Discovery	49.99$	-
BSIDE DMM	20.87$	-
TS100 Soldering Iron	71.22$	-
GPS Shenzhen Anxie (x10)	39.15$	-
TTGO T-Display	12.11$	-
ESP32 (x6)	24.21$	-
Geiger-Muller Tubes (x3)	62$	-
PCBs	12.51$	-
Hantek Oscilloscope	-	350₾
LM350 IC	-	6.80₾
Potentiometer 10K	-	1.30₾
Potentiometer 10K	-	4.70₾
Banana Plugs	-	8₾
Cables	-	10₾
Banana Sockets	-	5.20₾
Resistors (x26)	-	1.8₾
BC516 Transistor (x9)	-	5.4₾
BC637 Transistor (x5)	-	2.5₾
2N222A Transistor (x5)	-	1.5₾

Ceramic Capacitors (x22)	-	5.1₾
Boost Converter XL6019 (x4)	-	24₾
Step down converter (x3)	-	15₾
Microchip nest (x4)	-	1.2₾
2 sided circuit board	-	5₾
Voltmeter-Ampermeter	-	33₾
Li-Ion Battery Socket (x4)	-	13.50₾
Li-Ion Battery (x3)	-	36₾
MBR2545 Diode (x5)	-	17.5₾
Tweezers (x2)	-	16.2₾
Magnifying glasses	-	40₾
Glue Gun	-	70₾
Silicon Glue (x5)	-	7.5₾
Caliper	-	55₾
Textolite for PCB	-	13₾
Isolating organizer for soldering	-	54₾

The Total Cost is 690$