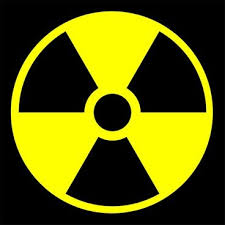
**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ КЪМ ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**Проект: Radiation Monitoring System**

****

Тема: Наблюдение нивата на радиация в рискови зони

Изготвил: Ивайло Русинчовски

София 2025

**Въведение**

Цел на проекта

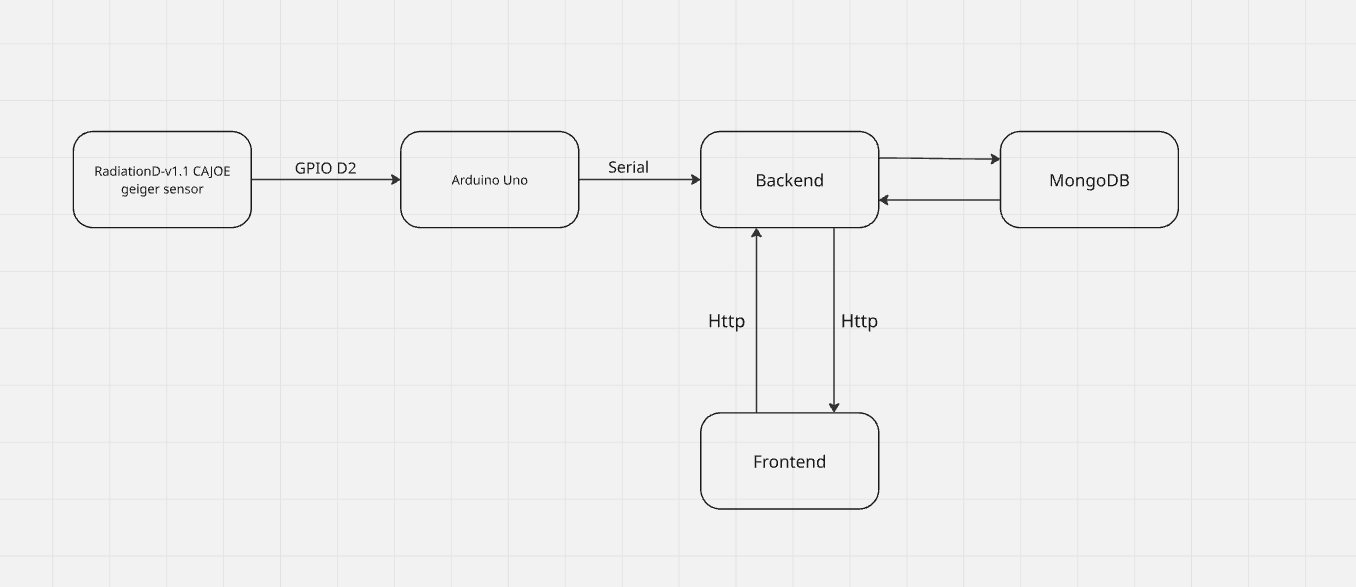
Целта на проекта е да се създаде приложение, което да улеснява наблюдението на нивата на радиация и предупреждаване при опасни такива. Достъпът до приложението е сравнително лесен, като е нужно да се премине през регистрация и да е на лице гайгеров сензор и микроконтролер, така че приложението е подходящо за използване от техници в атомни електроцентрали, еколози, хора, подготвящи бункери в случай на ядрена зима и др. Уеб приложението предоставя графика с измерените нива на радиация спрямо точки от времето, както и визуално предупреждение при надвишаване на безопасните нива.

Ключови функционалности

* Интегриран гайгеров брояч, данните от който се събират от микроконтролер Arduino uno, който е свързан с машината, на която е пуснато приложението.
* Web socket real time ъпдейти за максимална точност на визуализацията на измерените данни.
* JWT authentication за защитен достъп
* Контейнеризиране за по-лесно опериране и deployment

**Технически спецификации**

Архитектура



Използвани технологии

* Frontend - NextJS + Tailwind- Тъй като предлага multi routing out of the box
* Backend - NodeJS + Express- тъй като Експрес е стандарта в индустрията
* Database - MongoDB- понеже няма нужда от SQL, поради не релационната натура на данните
* Контейнеризация- Docker, понеже е най-масово достъпната технология за тази цел
* RadiationD-v1.1 CAJOE- тъй като е бюджетен гайгеров сензор
* Arduino UNO- тъй като предлага проста интеграция с уеб приложението чрез Serial

**Инструкции за стартиране**

1. Клонираите гитхъб репозиторито
   1. Линк: <https://github.com/Ivailo2707/Radiation-Monitoring-System>
2. Уверете се, че имате инсталирани всички гореспоменати софтуерни технологии.
3. Инсталирайте dependenci-тата във frontend-a и backend-a чрез npm install
4. Ако притежавате необходимите хардуерни компоненти
   1. Отворете backend/[server.js](http://server.js)
   2. На ред 46 сложете порта, на който е plug-нато Вашето Arduino UNO
5. Ако не притежавате необходимите хардуерни компоненти
   1. Отворете backend/[server.js](http://server.js)
   2. Закоментирайте ред 45-65
6. Преименувайте backend/.env.example на .env
7. Върнете се в главната директория и run-нете командата docker-compose up --build
8. Отворете localhost:3000 във Вашия browser

**Frontend детайли**

* Използвани библиотеки/framework-ове:
  + Next JS
  + Tailwind
* Основна структура на компонентите
  + src
    - app
      * dashboard
      * login
      * signup
      * layout
      * page
    - components
      * common
        + ProtectedRoute
    - context
      * AuthContext
    - lib
      * auth
      * readings
    - themes
      * theme
  + Dockerfile
* Стилизиране
  + Tailwind utility classes
* HTTPS
  + dev-https

**Backend детайли**

* Използвани технологии
  + NodeJS
  + Express
  + JWT
  + WebSocket Integration
  + Rate Limiting
  + Serial Port Data Ingestion
* API routes
  + Post/signup
  + Post/login
  + Post/readings
  + Get/readings
* Mongo models
  + User
    - username:String
    - password:String
  + Reading
    - value:Number
    - timestamp:Date

**Тестване**

* Thunderclient- API тестване
* test js- За тестване на предупрежденията

**Проблеми**

* Deployment
  + наличието на хардуер е предпоставка за проблеми при deploy-ването
  + Идея за решение: Използването на автоматично засичане на порта за ардуиното или операции с приложението изцяло на микрокомпютър
* Слабо security
  + За момента сигурността е обикновена signup/login система
  + Идеи за решения: Имплементиране на 2fa и Captcha и HTTPS

**Бъдещи подобрения**

* Deploy-ване в сървър или cloud
* Подобряване на графичния интерфейс
* Подобряване на сигурността
* Мащабизиране на системата
* Разрешаване на гореспоменатите проблеми