

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

ДИПЛОМНА РАБОТА

Тема: Система за анализ и идентификация на звукови сигнали в реално време

Дипломант: Виктор Александров Тодоров

Научен ръководител:

доц. д-р инж. Ст. Стефанова

СОФИЯ

2021

СЪДЪРЖАНИЕ

| | |
|--|-----------|
| Увод | 6 |
| Първа глава - Проучване на съществуващи проекти и на подходящи технологии за реализирането на системата | 8 |
| 1.1. Основни понятия, параметри и подходи, използвани при анализа на звукови сигнали | 8 |
| 1.1.1. Основни понятия и параметри | 8 |
| 1.1.2. Преобразувания на Фурье | 11 |
| 1.2. Хардуерни системи за анализ на звукови сигнали | 15 |
| 1.2.1. Комерсиални системи | 15 |
| 1.2.2. Любителски системи | 18 |
| 1.3. Софтуерни системи за анализ на звукови сигнали | 24 |
| 1.3.1. Комерсиални системи | 24 |
| 1.3.2. Системи за свободно ползване с отворен код | 26 |
| 1.4. Основни хардуерни платформи и компоненти, използвани при анализа на звукови сигнали | 31 |
| 1.4.1. Микроконтролери и модули за безжична комуникация | 31 |
| 1.4.2. Микрофони/модули (MEMS) | 41 |
| 1.4.3. Сензори за влажност и температура | 43 |
| 1.5. Основни софтуерни компоненти за обработка на звукови сигнали | 43 |
| 1.5.1. Библиотеки за обработка на звукови сигнали | 43 |
| 1.5.2. Софтуерни приложения и библиотеки за визуализация | 45 |
| 1.6. Бази данни | 47 |
| 1.6.1. InfluxDB | 47 |
| 1.6.2. PostgreSQL | 48 |
| 1.6.3. Redis | 48 |
| 1.6.4. Сравнение и анализ на БД | 49 |
| Втора глава - Основни изисквания и блокови схеми | 50 |
| 2.1. Основни изисквания към системата, проектирана в дипломната работа | 50 |
| 2.1.1. Архитектурни и комуникационни изисквания | 50 |
| 2.1.2. Функционални изисквания | 50 |
| 2.2. Основни блокови схеми | 51 |
| 2.2.1. Обща блокова схема | 51 |

| | |
|--|-----------|
| 2.2.2. По-детайлни блокови схеми на отделните блокове | 52 |
| Трета глава - Проектиране на принципна електрическа схема на системата с използване и на схеми на готови модули | 56 |
| 3.1. Избор на основни компоненти | 56 |
| 3.1.1. Избор на CAD система | 56 |
| 3.1.2. Избор на готови модули за Блок сървър | 56 |
| 3.1.3. Избор на Блок крайно устройство | 57 |
| 3.1.4. Избор на компоненти на Блок входни сигнали | 57 |
| 3.2. Проектиране на принципна електрическа схема за Блок крайно устройство | 58 |
| 3.2.1. Вариант 1 - Seeeduino LoRaWAN | 61 |
| 3.2.2. Вариант 2 - MKR WAN 1300 | 61 |
| Четвърта глава - Проектиране на печатна платка | 62 |
| 4.1. Печатни платки на използваните модули | 62 |
| 4.1.1. Raspberry Pi 3 model B+ | 62 |
| 4.1.2. RAK831 | 63 |
| 4.1.3. Seeeduino LoRaWAN | 63 |
| 4.1.3. MKR WAN 1300 | 64 |
| 4.1.4. ADMP401 модул | 65 |
| 4.1.5. BME680 pimoroni модул | 65 |
| 4.2. Проектиране на печатна платка за Блок крайно устройство | 65 |
| 4.2.1. Вариант 1 - Seeeduino LoRaWAN | 65 |
| 4.2.2. Вариант 2 - MKR WAN 1300 | 65 |
| Пета глава - Проектиране на управляващ софтуер | 66 |
| 5.1. Използван софтуер, езици и библиотеки | 66 |
| 5.1.1. LoRaWAN протокол за комуникация между Блок сървър и Блок крайно устройство | 66 |
| 5.1.2. ChirpStack и rak_common_for_gateway | 68 |
| 5.1.3. Adafruit BME680 Library | 69 |
| 5.1.4. MKRWAN.h | 70 |
| 5.1.5. LoRaWan.h | 70 |
| 5.1.6. Adafruit ZeroFFT Library | 70 |
| 5.1.7. InfluxDB | 70 |
| 5.1.8. Grafana | 71 |
| 5.2. Инсталиране на необходимия софтуер върху Блок сървър | 72 |
| 5.2.1. Инсталиране на ОС върху БС | 72 |
| 5.2.2. LoRaWAN network сървър, приложен сървър, gateway-bridge и rak_common_for_gateway | 72 |

| | |
|---|------------|
| 5.2.3. Grafana | 72 |
| 5.2.4. Influx DB | 73 |
| 5.3. Управляващ софтуер на Блок сървър | 74 |
| 5.3.1. Блокови схеми и принципни схеми на БС | 74 |
| 5.3.2. Сурс код и настройки по Grafana и ChirpStack приложния сървър на БС | 77 |
| 5.4. Управляващ софтуер на Блок крайното устройство | 97 |
| 5.4.1. Блокови схеми и принципни схеми | 97 |
| 5.4.2. Сурс код на БКУ | 100 |
| Шеста глава - Практически резултати | 104 |
| 6.1. Проблеми по време на разработката и решения | 104 |
| 6.1.1. Декодирането на пакета с данни върху приложния сървър | |
| 104 | |
| 6.1.2. Повреждане данните, взети от Аналогово-цифровия преобразувател, при инициализацията на класа Adafruit_BME680 | |
| 106 | |
| 6.1.3. Липса на обратна връзка от страна на шлюза към крайните устройства | |
| 107 | |
| 6.1.4. Свързването на MKR WAN 1300 модула към LoRaWAN мрежата | |
| 109 | |
| 6.2. Тестове и изводи | 109 |
| 6.2.1. AudioFrequencyMeter.h | 109 |
| 6.2.2. Adafruit_ZeroFFT.h | 110 |
| 6.2.3. ArduinoSound.h | 110 |
| 6.2.4. KickFFT.h | 110 |
| 6.2.5. Тест върху функцията и изправността на проектирания филтър | |
| 110 | |
| 6.2.5. Тест върху Аналогово-цифровия преобразувател и настройките му | |
| 114 | |
| 6.3. Практически резултати и изводи | 115 |
| 6.3.1. Пускане в действие и инсталиране на Rpi v.3 model B+ | 115 |
| 6.3.2. Инсталлиране на Grafana и InfluxDB и настройването им | 115 |
| 6.3.3. Изпращане на данните към приложния сървър | 117 |
| 6.3.4. Успешно записване на данните в InfluxDB | 119 |
| 6.3.5. Визуализация на данните в браузъра и изпращане на алармиращи имейли | 122 |
| 6.3.6. Прототипиране на устройството | 126 |
| Заключение | 132 |
| Използвана литература | 133 |

| | |
|---|------------|
| Приложения | 139 |
| П1. Електрическа схема на Raspberry 3 Model B+ [42] | 140 |
| П2. Електрическа схема на RAK831 [43] | 141 |
| П3. Електрическа схема на Seeeduino LoRaWAN [44] | 142 |
| П4. Електрическа схема на MKR WAN 1300 [45] | 143 |
| П5. Електрическа схема на ADMP401 [46] | 144 |
| П6. Електрическа схема на BME680 модул | 145 |
| П7. Електрическа схема и печатна платка на БКУ - вариант 1, Seeeduino LoRaWAN [47] | 146 |
| П8. Електрическа схема и печатна платка на БКУ - вариант 2, MKR WAN 1300 [48] | 147 |
| П9. MKR WAN 1300 pinout [49] | 148 |

Увод

Звукът е аналогов сигнал, който е носител на голямо количество информация. Тази информация се характеризира от неговите параметри - честота (**f**), интензитет (**I**), сила (ниво на интензитета) (**L**) и фаза (**φ**). Звуковата информация служи за комуникация между различните индивиди и дава обща представа за случващите се процеси в света около нас. Електронният анализ на звука намира все по-широко приложение в най-различни области като например звукозаписа, филмовите продукции, анализа на морското дъно, анализа на поведението и комуникацията между различните видове животни, като например поведението на пчелите в един пчлен кошер. През последния век се провеждат множество проучвания, свързани с пчелите, и някои от тези проучвания показват, че един пчлен кошер издава звукове/шумове в нискочестотната област и честотата на звуковия сигнал може да даде много информация за общото състояние и поведение на цялата колония. Други проучвания от своя страна показват, че самата пчела е способна да издава звукове до над 6.5 kHz.[1] Такъв вид информация може да бъде записана и обработена (анализирана), с което да се улесни работата на пчеларя по поддържането на кошерите.

В природата звуковите сигнали са съставени от множество звукови вълни, някои от които са носители на полезна информация, а някои са просто шумове. При изследването на някакъв конкретен обект преди всичко трябва да се знае честотния диапазон, в който той издава звукови сигнали, след което може да се отдели всеки сигнал чрез подходящо филтриране по честота, за да бъде подложен на по-нататъшна обработка. За да бъде разложен сигнала на прости честоти, може да се използва преобразуване на Фурье, което

разглежда сложната звукова вълна като съставена от множество прости сигнали.

Целта на настоящата дипломна работа е да се разработи система за анализ и идентификация на звукови сигнали - по възможност в реално време. Идеята на системата е да може по нея да се разработи по-сложна система, която по-нататък да намери реално приложение в умното отглеждане на пчелите. За тази цел системата ще се разработи така, че освен да прави Фурье анализ на звука, ще предоставя и други параметри за състоянието на околната среда - температура, влажност, замърсяване на въздуха, както и атмосферно налягане. След като бъдат събрани данните, те ще се предават от крайните устройства чрез подходяща технология и протокол към сървър, който да ги визуализира. За реализирането на тази цел ще бъде направен обзор, анализ и сравнение на съществуващи подобни системи и използваните в тях платформи и технологии. Ще бъдат разгледани подходящи хардуерни модули за реализирането на системата, както и технология за комуникация, предоставяща достатъчно голям обхват на свързване и ниска консумация. На тази база ще бъдат дефинирани конкретните изисквания към системата и ще се синтезира блоковата ѝ схема. Проектирането по същество ще се извърши чрез избор на подходяща платформа и компоненти, синтезиране на принципна електрическа схема, проектиране на печатна платка и управляващ софтуер (фърмуер). В края на дипломната работа ще бъдат представени практическите резултати, на базата на които ще се направят съответните изводи.

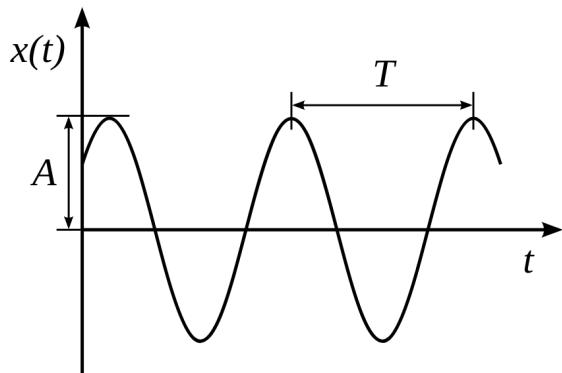
Първа глава - Проучване на съществуващи проекти и на подходящи технологии за реализирането на системата

1.1. Основни понятия, параметри и подходи, използвани при анализа на звукови сигнали

1.1.1. Основни понятия и параметри

Звукът е надлъжна механична вълна, която е резултат на промени в налягането и трептения в еластична среда (въздух, флуиди, твърди тела).[2] Звуковият сигнал се характеризира със своя период, честота, амплитуда, фаза, дължина на вълната, тембър и продължителност.

На фиг. 1.1. е показана графика на един звуков сигнал.



Фиг. 1.1.

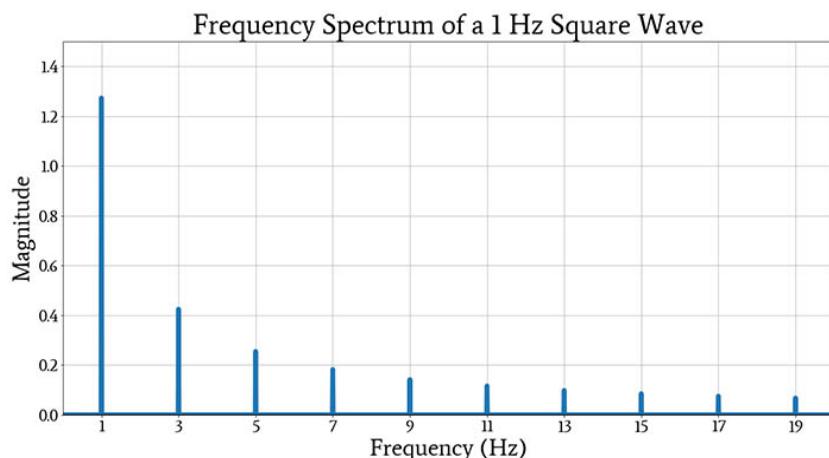
А - амплитуда на звуковата вълната.

Т - период на звуковата вълна.

Извън идеалния случай, звуковата вълна представлява времева функция съставена от много прости звуци и шумове. В този вид звуковият сигнал не може да бъде подложен на пълноценен анализ и затова трябва да бъде разложен на простите си синусоидални съставляващи чрез спектрално разлагане.[2] Идеята на спектралното разлагане е, че всеки сигнал може да бъде представен като сума от

синусоиди и косинусоиди с различни честоти. За тази цел се използва преобразуване на Фурье, което преобразува времевата функция на сигнала в неговия спектър.[3]

На фиг. 1.2. е показан честотният спектър на звукова вълна с честота 1 Hz.



Фиг. 1.2.

По абсцисата е дадена честотата f , а по ордината - амплитудата.

За да бъде подложен на цифрова обработка, звуковият сигнал трябва да бъде преобразуван в електрически посредством микрофон.

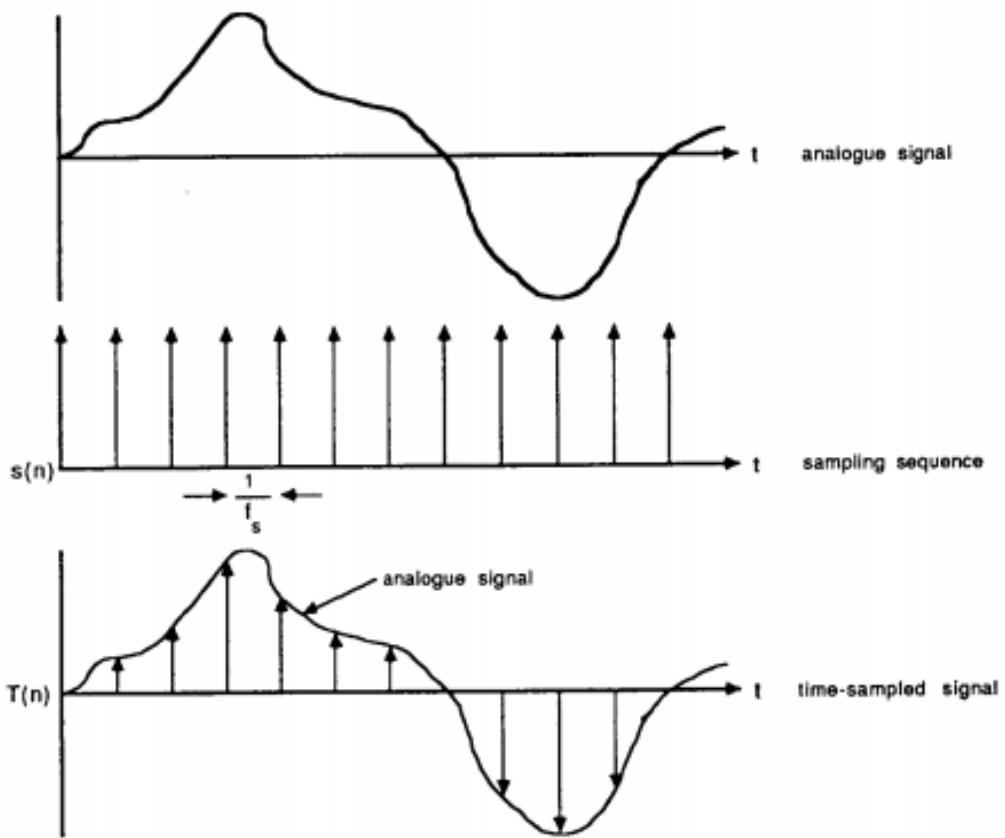
Електрическият сигнал представлява аналогов сигнал, който не подлежи директно на цифров анализ. Следователно той трябва да бъде конвертиран от аналогов в цифров вид, тоест да бъде дискретизиран, чрез аналогово-цифров преобразувател. За да бъде един сигнал дискретизиран без загуба на информацията, която носи, върху него трябва да бъде приложена теоремата на Kotelnikov-Nyquist (Котелников-Найкуист), известна още като Nyquist-Shannon (Найкуист-Шанън) или Kotelnikov-Nyquist-Shannon (Котелников-Найкуист-Шенън) за дискретизация на сигнали.[4]

Формула 1.1. показва зависимостта на честотата на дискретизация от максималната честота на изследвания сигнал.

$$f_s \geq 2f_{max} \quad (1.1.)$$

където f_s е честотата на дискретизация (честотата, с която се вземат пробите), а f_{max} е максималната честота на звука, който може да бъде записан без загуба на информация.

На фиг. 1.3. е показан аналогов сигнал и процеса на неговата дискретизация (преобразуване в цифров вид).



Фиг. 1.3.

Аналогов сигнал и вземане на преби по теорема на Котелников. Във всяка от дискретните преби се записва число, което е моментната стойност на амплитудата. Цифровата стойност на амплитудата може да бъде различна в зависимост от броя на нивата на преобразуване, но

като цяло най-разпространените формати са 8 битов (256 нива/стойности), 10 битов (1024 нива/стойности), 16 битов (65 536 стойности), 24 битов (2^{24} стойности) и 32 битов (2^{32} стойности).

Както вече беше споменато, за да бъде направен качествен анализ на сигнала, трябва да бъде изследван неговият спектър. Спектърът на аудио сигнала (звуков сигнал записан в цифров вид) може да се извлече, като се използва преобразувание на Фурье.

1.1.2. Преобразования на Фурье

Стандартна форма на преобразувание на Фурье

Преобразуванието на Фурье (ПФ) е математическа техника, която преобразува функция на времето $f(t)$, във функция на честотата $\hat{f}(\omega)$. [4]

Основната идея на анализа на Фурье е да се сравни сигнала със синусоиди с различни ъглови честоти $\omega \in R$ (измерени в Hz), където R е множеството на реалните числа. Всеки такъв синусоидален сигнал или чист тон може да се разглежда като прототип на трептене. В резултат на това сравнение за всеки разглеждан честотен параметър $\omega \in R_a$ се получава коефициент на величина $d_\omega \in R \geq 0$ (заедно с фазов коефициент $\varphi_\omega \in R$). В случай, че коефициентът d_ω е голям, има голяма прилика между сигнала и честота на синусоидата ω и сигналът съдържа периодично трептене с тази честота. В случай, че d_ω е малък, сигналът не съдържа периодичен компонент с тази честота.[3]

Формула 1.2. показва математическия запис на ПФ.

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x} dx, \quad (1.2.)$$

където x е времето, ω е ъгловата честота, по която се сравнява, а i е имагинерната съставка.

Към аудио сигнал не може да бъде приложена директно тази формула, защото тя използва метода на интегралното смятане. Поради тази причина трябва да се използва дискретното преобразуване на Фурье.

Дискретно преобразуване на Фурье

Основният проблем при пресмятането на ПФ върху машина е, че ъгловата честота ω е безкрайна величина, следователно се избира краен брой честоти $\omega = k/M$ за подходящи $M \in N$ и $k \in [0, M - 1]$, където N е множеството на простите числа. От практическа гледна точка обикновено се избира $M = N$, където N е броят на дискретните преби, които са взети за изследвания отрезък от време. Формула 1.3. показва математическия запис на дискретното преобразуване на Фурье (ДПФ).[5]

$$X(k) = \hat{f}(k/N) = \sum_{n=0}^{N-1} f(n) e^{-2\pi i n k / N} \quad (1.3.)$$

Както вече беше споменато в точка 1.1.1. сигналът $A(t)$ е дискретизиран във времето с N равномерни отчета през интервал $\Delta = 1/2f_s$, където f_s е критичната честота на Котелников (Найкуист), а спектърът се оказва ограничен, т. е. $S(f) \equiv 0$ за $f > f_s$.

Проблем на този метод е големият брой пресмятания за преминаване от времева форма или област (ВФ, Time Domain, TD) във честотна (ЧФ, Frequency Domain, FD) и паразитните ефекти на трансфер на съставките с $f > f_s$ вътре в интервала $0 < f < f_s$ (поява на фалшив спектър - aliasing), където f_s е честотата на дискретизация. Проблемът се оказва доста тежък – излишните отчети извън лентата $f > f_s$

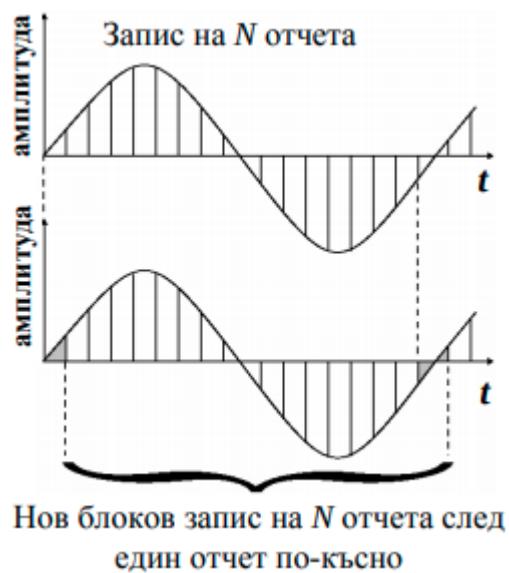
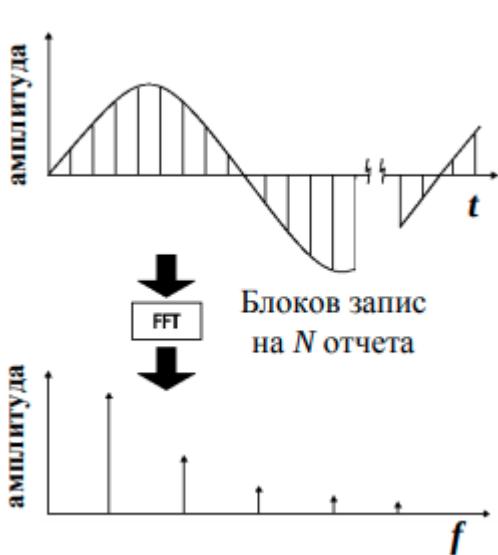
променят истинския спектър $f < f_s$ във фалшив. Това може да се избегне чрез филтриране с ниско-честотен филтър преди дискретизацията на отчетите.[6]

ДПФ има сложност от $O(N^2)$, което би отнело много време при голямо N , затова може да се приложи бързото преобразуване на Фурье, което е със сложност $O(N \times \log_2 N)$.

Бързо преобразуване на Фурье

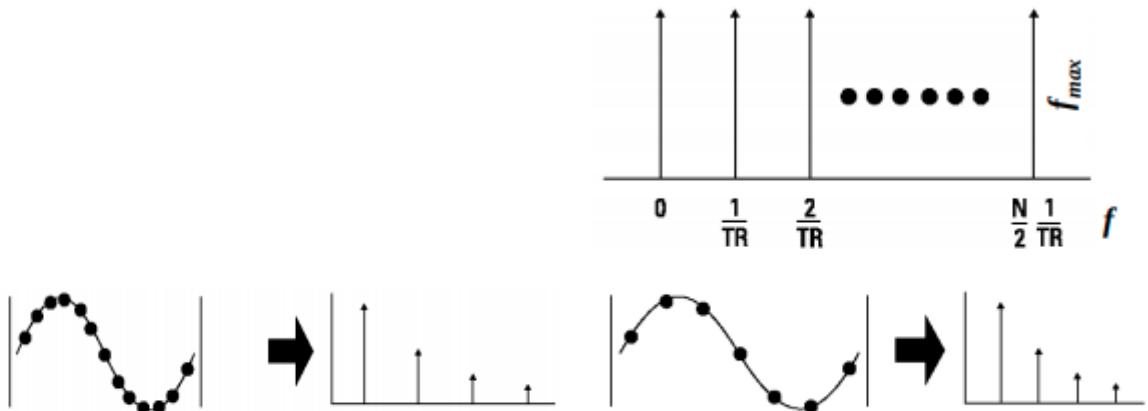
Идеята на бързото преобразуване на Фурье (БПФ) е да се правят времеви отчети само в обхвата $f < f_s$. Така вместо за всички честоти, линейно независими записи се правят само за N на брой дискретни честоти $f_n = n/N\Delta$ (където $n = -N/2, \dots, +N/2$). По този начин броят математически операции при БПФ е само $N \times \log_2 N$. Пример: ако $N = 10^6$, то ДПФ се нуждае от 10^{12} операции, а бързото БПФ – само от $2 \cdot 10^7$ операции (т.е. разликата е близо 5 порядъка!).

Важно за придобиване на представа как работи БПФ е понятието за “блоков времеви запис” (time record, TR). Идеята е следната – правят се N на брой времеви отчета в блок. Най-добре е броят записи N да е число с множител 2 (2, 4, 8, 16, 32 ...) – например 1024. Така записът от N отчета в блок се преобразува в блок от определен брой дискретни честотни линии – фиг. 1.4. Така всичките N отчета са необходими за пресмятане чрез БПФ алгоритъма на всяка честотна линия в спектъра. Този спектър е валиден до следващия блоков запис – фиг. 1.5.[6]



Описаният БПФ алгоритъм се оказва много ефективен. За да се избегне обаче появя на "фалшивият" отклик в спектъра (aliasing), трябва да се ограничи "бързината", с която се правят блоковите записи – фигури 1.6 и 1.7 по-долу. Нека TR да е периодът за блоков запис на N времеви отчета (Time Record). Тогава броят на дискретни честотни линии в спектъра е $N/2$, а максималната честотна съставка в спектъра е даден във формула 1.4. Тази честота трябва да е по-малка от критичната честота на Найкуист, т. е. $f_{max} < f_s$. Това е начинът за ограничаване и на "скоростта" на блоковия запис и на броя на записите. Понеже величината N е избрана от други съображения, ограничението всъщност е за периода на запис TR .

$$f_{max} = \frac{N}{2} \times \frac{1}{TR} < f_{sample} \quad (1.4.)$$



Пример 1: Бърз времеви запис води до широко разположени линии в спектъра

Фиг. 1.6.

Пример 2: Бавен времеви запис води до близко разположени линии в спектъра

Фиг. 1.7.

1.2. Хардуерни системи за анализ на звукови сигнали

Със стандартен осцилоскоп звукът може да се представи като функция на времето, но за да се видят отделните му съставки трябва да се използва спектроанализатор. Спектроанализаторът може да покаже както отделните съставки на сигнала, така и техните фази, амплитуда, шум и др.

По-долу ще бъдат разгледани съществуващи системи за анализ на звукови сигнали - както комерсиални, така и любителски, също така ще бъдат разгледани и техните характеристики и параметри.

1.2.1. Комерсиални системи

SM90 Class 1 Measuring Instrument

Bedrock SM50 е многофункционален акустичен измервателен уред, напълно съвместим със спецификациите на ниво 1 според стандарта IEC-61672 на международната електротехническа комисия. Той вече се предлага стандартно с модул за анализатор в реално време (1/1 и 1/3 октави). С течение на времето ще бъдат добавени още софтуерни модули чрез бесплатни надстройки. Уредът разполага с пълноцветен

сензорен еcran и може да съхранява измерените данни чрез USB.[7] На фиг. 1.8. е показано устройството.



Фиг. 1.8.

Изглед на SM90 Class 1 Measuring Instrument.

Характеристики:

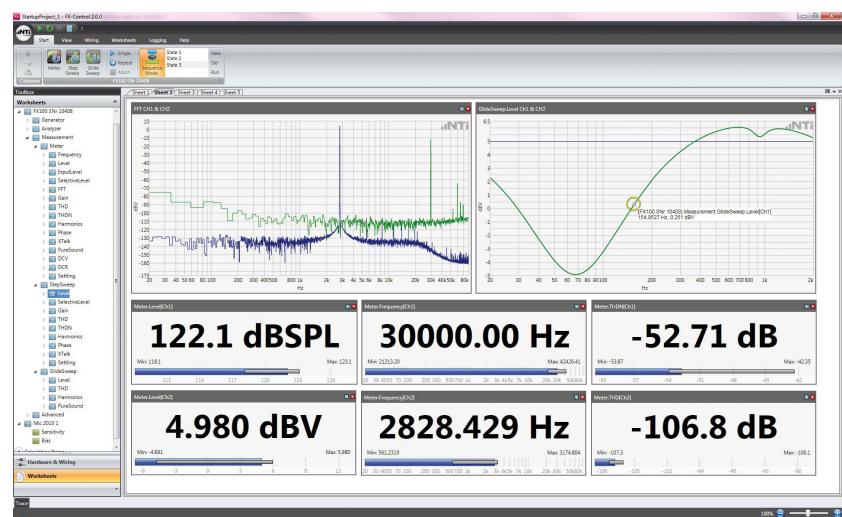
- Изключително широка функционалност;
- Интуитивен и лесен за работа;
- Възможност за запис;
- Честотна лента от 20Hz до 20kHz;
- Издръжливост на батерията между 4 и 6 часа;
- 8 GB памет, като минимум 7.5 GB могат да бъдат използвани за съхранение на измерванията;
- Захранва се с батерия;

FLEXUS FX100 Audio Analyzer

FLEXUS FX100 е устройство подходящо, както за провеждане на изследвания, така и за използването му в реален проект. Има много функционалности, може да поддържа до 14 аудио канала едновременно, има множество софтуерни функционалности и т.н. [8] На фиг. 1.9. е показано устройството и интерфейсите му. На фиг. 1.10. е показан софтуерът, който предлагат за Windows операционната система.



Фиг. 1.9.
Изглед на FLEXUS FX100 Audio Analyzer.



Фиг. 1.10.
Изглед на софтуерното приложение за уреда.

Характеристики:

- Изключително широка функционалност;
- Интуитивен и лесен за работа;
- Възможност за запис.
- Честотна лента от 5 Hz до 80 kHz.
- Максимална честота на вземане на проба е 192 kHz, като се прилага БПФ върху сигнала;
- Поддържа .NET, което му позволява да се вгражда в различни проекти;

- Може да се използва, както за анализатор, така и за генератор на аналогови и цифрови аудио сигнали.

В Табл. 1.1. е показано сравнение на описаните по-горе системи.

Таблица 1.1. Сравнение на комерсиални системи за анализ на звукови сигнали.

| Модел → Параметър ↓ | FLEXUS FX100 Audio Analyzer | SM90 Class 1 Measuring Instrument |
|------------------------------------|------------------------------------|--|
| Минимална честота | 5Hz | 20Hz |
| Максимална честота | 80kHz | 20kHz |
| Допълнителна функционалност | Да | Да |
| Връзка с компютър | Да | Да |
| Възможност за интегриране в проект | Да | - |
| Захранване | Ел. мрежа | Батерия |
| Възможност за запис | Да | Да |
| Място за запис | - | 7.5 GB |

От сравнението по-горе се вижда, че с изключение на честотния обхват и захранването, останалите функционалности на системите са сходни.

1.2.2. Любителски системи

Спектрален анализатор на базата на Ардуино с LCD дисплей

Спектралният анализатор е създаден на базата на платформата Ардуино и течнокристален графичен дисплей, и покрива честотния диапазон от 0 до приблизително 3,5 kHz. Той използва алгоритъма за бързо преобразуване на Фурье от библиотеката - "fix_fft.h". [9] На фиг. 1.11. е показана системата.



Фиг. 1.11.

Изглед на Спектрален анализатор на базата на Ардуино с LCD дисплей.

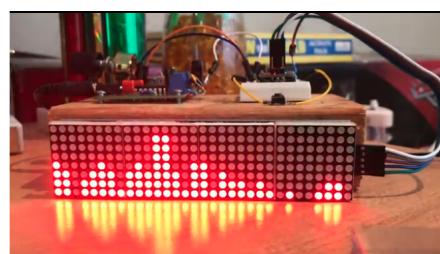
Характеристики:

- Реализиран чрез Arduino Nano;
- Използва дисплей с относително висока резолюция (ST7920 128x64 LCD);
- Сравнително добро разделение на спектъра;
- Липса на микрофон - входният сигнал се подава чрез генератор на аналогови сигнали(звукова система);
- Не е тестван реално каква честотна лента обхваща;
- Липсва филтър;
- Използва по бавната функция analogRead(), за да чете подадения сигнал;
- Не е достатъчно добре документиран;
- Използва по-бързата, но по-неточна библиотека "fix_fft.h", сравнение с "arduinoFFT.h".
- Не е реализирано запазване на данните.

Спектрален анализатор на базата на Ардуино с LED дисплей

Спектралният анализатор (спектрометър) е базиран на платформата Ардуино и светодиоден дисплей, и покрива честотен диапазон

приблизително от 200 Hz до 19,32 kHz. Използва алгоритъма за бързо преобразуване на Фурье от библиотеката е "arduinoFFT.h". [10]
Видът на системата е показан на фиг. 1.12.



Фиг. 1.12.

Изглед на Спектрален анализатор на базата на Ардуино с LED дисплей.

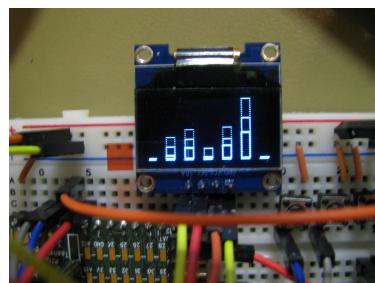
Характеристики:

- Проектът е реализиран върху платформите Arduino Uno и Arduino Nano;
- Голяма ширина на покриваната честотна лента;
- Използва честота на дискретизация от 38.64 kHz, което му позволява да възпроизведе сигнал до 19.32 kHz;
- Взема 64 преби на всеки блоков времеви запис;
- Директно взема резултатите на пробите чрез настройки на регистрите, което е значително по-бързо от функцията `analogRead()`;
- Добре документиран проект.
- Дисплей с малка резолюция - 32x8 LED матричен дисплей;
- Малък брой съставки на спектъра, изобразявани на дисплея - 32 честоти;
- Липсва филтър;
- Звукът се предава чрез звукова система (генератор на сигнали) - липсва микрофон;

- Използва библиотеката "arduinoFFT.h", която дава по-прецизни резултати.
- Не е реализирано запазване на данните.

OLED спектрален анализатор на базата на Arduino Pro Mini

Спектрален анализатор използваш платформата Arduino Pro Mini, OLED дисплей с разрядност 28x641, микрофон MAX9812 и използва филтъра MSGEQ7, който филтрира звука на 7 предефинирани честоти (63Hz, 160Hz, 400Hz, 1kHz, 2.5kHz, 6.25kHz и 16kHz), като следи за тяхната амплитуда.[11] На фиг. 1.13. е показана системата.



Фиг. 1.13.

Изглед на OLED спектрален анализатор на базата на Arduino Pro Mini.

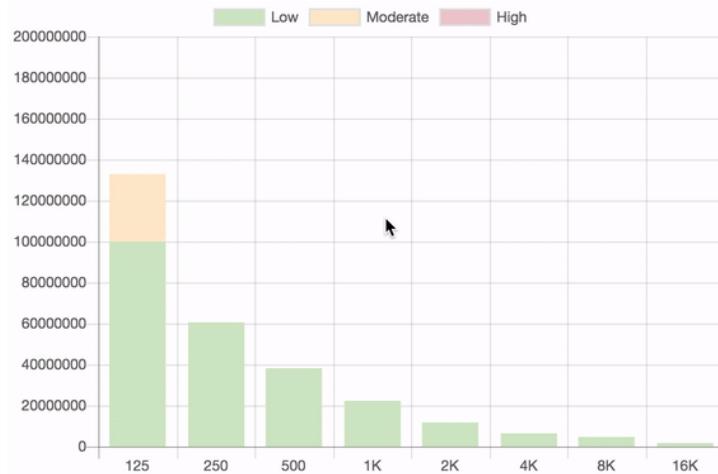
Характеристики:

- Проектът е реализиран върху Arduino Pro Mini, което значи, че кодът е съвместим с Arduino Uno, Nano и т.н.;
- Директно филтриране чрез MSGEQ7, което спестява натоварването върху микроконтролера;
- Използва OLED дисплей;
- Взема звуковия сигнал чрез микрофон;
- Добре документиран проект.
- Малък брой съставки на спектъра изобразявани на дисплея - 7 предварително зададени честоти;
- Статичност на спектъра (честотите винаги са едни и същи);

- Не е реализирано запазване на данните.

Спектрален анализатор на базата на ESP32

Спектрален анализатор реализиран с помощта на платформата ESP32, която визуализира спектъра в браузъра посредством Web Socket комуникация.[12] На фиг. 1.14. е показана системата:



Фиг. 1.14.

Изглед на Спектрален анализатор на базата на ESP32.

Характеристики:

- Проектът има интернет свързаност чрез Web Socket-и;
- Филтрацията се извършва върху микроконтролера с помощта на Фурье преобразуването, като използва библиотеката "arduinoFFT.h" за тази цел;
- Получава звуковия сигнал посредством микрофон;
- Използва голямо мащабиране в браузър средата;
- Проектът е зле документиран - липсват схеми и код за приложението в браузъра;
- Не е реализирано запазване на данните.

В Табл. 1.2. е дадено сравнение на разгледаните в тази подточка системи.

Таблица 1.2. Сравнение на любителски системи за анализ на звукови сигнали.

| Система, базирана на: → Параметър ↓ | Ардуино с LCD дисплей | Ардуино с LED дисплей | Ардуино с OLED дисплей | ESP32 |
|---|-----------------------------|--------------------------|----------------------------------|------------------|
| Минимална честота | ≈0Hz | ≈200Hz | 63Hz | ≈125Hz |
| Максимална честота | ≈3.5kHz | ≈19.32kHz | 16kHz | ≈16kHz |
| Непрекъснат спектър | Да | Да | Не | Да |
| БПФ посредством | fix_fft.h | arduinoFFT.h | MSGEQ7 | arduinoFFT.h |
| Визуализация | ST7920 128x64 LCD | LED 32x8 | OLED 28x641 | Браузър |
| Четене на вход | Вградена функция | Регистри | Дискретизиране на хардуерно ниво | Вградена функция |
| Документация | Лоша | Добра | Добра | Непълна (лоша) |
| Интернет свързаност | Не | Не | Не | Да |
| Запис на данните | Не | Не | Не | Не |
| Честота на дискретизация | - | 38.46kHz | - | 40kHz |
| Източник на сигнал | Генератор | Генератор | Микрофон | Микрофон |
| Филтър | Не | Не | Да | - |

Сравнението в таблицата показва, че системите се различават както по използвана хардуерна платформа, така и по спектрален обхват, начин на обработка и визуализация на сигнала, Интернет свързаност на системата други подобни. Това от своя страна позволява да се види и оцени по-широко многообразие от възможности за практическа реализация на подобна система.

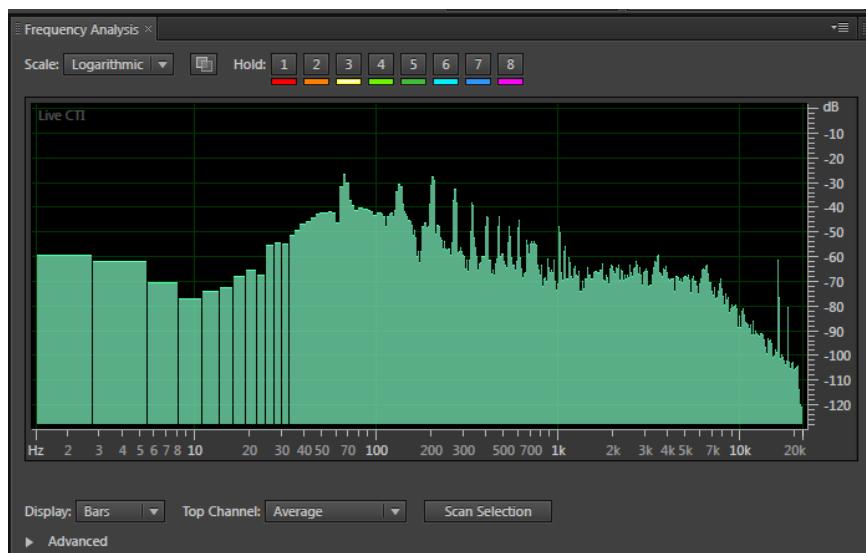
1.3. Софтуерни системи за анализ на звукови сигнали

Освен със хардуерни системи спектрален анализ на звука може да се прави и със софтуерни приложения. В тази точка ще бъдат разгледани комерсиални системи за спектрален анализ и системи с отворен код.

1.3.1. Комерсиални системи

Adobe Audition

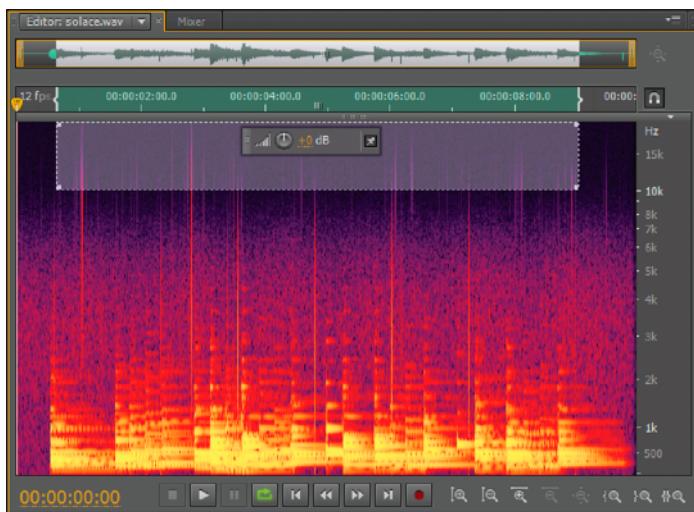
Audition е пълен набор от инструменти, включващ опция за множество записи, форма на вълната и спектрален дисплей за създаване, смесване, редактиране и възстановяване на аудио съдържание. Тази мощна аудио работна станция е създадена за ускоряване на работните процеси за продукция на видео и аудио изглажддане, както и за доставянето на усъвършенстван микс с чист звук.[13] На фиг. 1.15. е показана честотно-амплитудната характеристика на сигнала.



Фиг. 1.15.

Изглед на спектъра генериран от приложението.

На фиг. 1.16. е показана спектограмата на сигнала.



Фиг. 1.16.

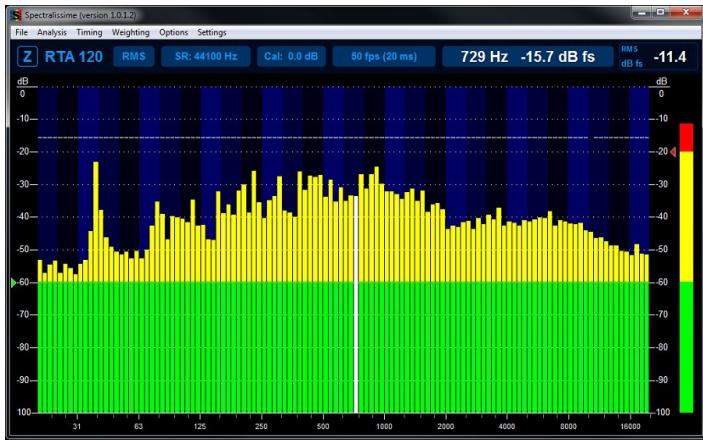
Спектрограмата генерирана от приложението.

Характеристики:

- Редактор с много функционалности;
- Интуитивен и лесен за работа;
- Възможност за запис.
- Честоти на дискретизация 44.1kHz, 48kHz, 88.2kHz, 96 kHz и др.

Spectralissime

Spectralissime е самостоятелно приложение за спектрален анализ, базирано на високопрецизна банка от лентови филтри, адаптирано към звуковото възпроизвеждане на човека. Приложението дава по-последователен анализ на спектъра и позволява по-добро разбиране на всякакви звукови явления благодарение на постоянната си точност, от 20 Hz до 20 kHz.[14] На фиг. 1.17. е показано приложението.



Фиг. 1.17.

Спектърът генериран от приложението.

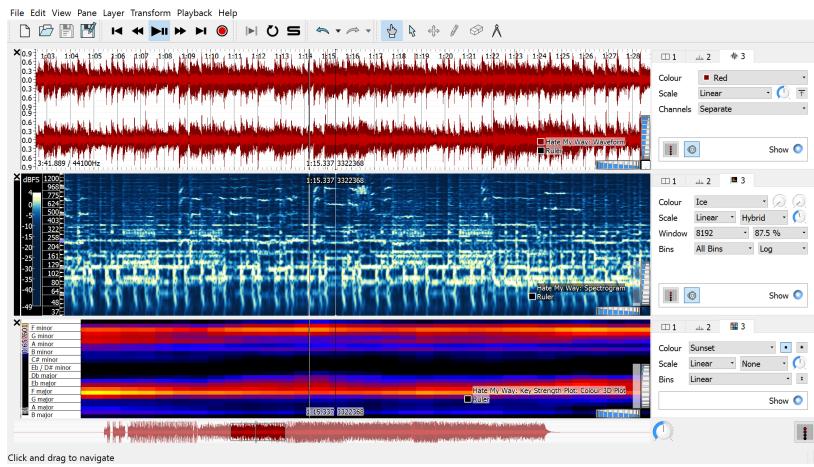
Характеристики:

- Безплатен;
- Много точен спектрален анализ в честотната лента (20Hz - 20kHz);
- Интуитивен и лесен за работа;
- Четири честоти на вземане на преби (44.1, 48, 88.2 или 96kHz);
- Възможност за работа в реално време;
- Направен е изцяло за Windows;

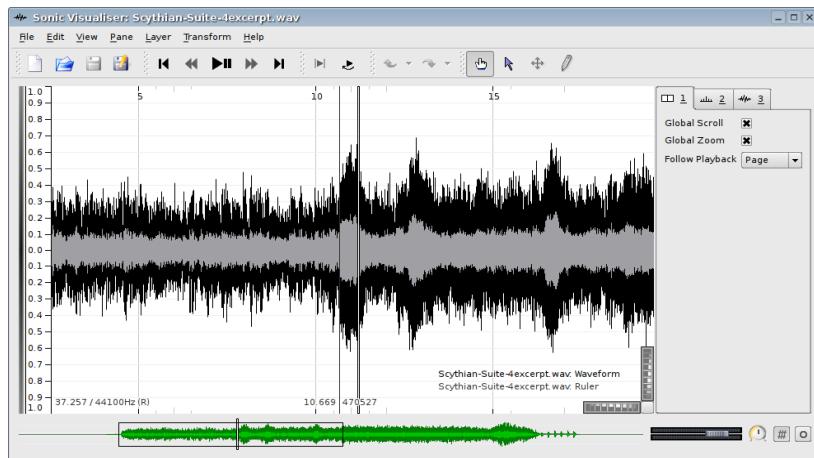
1.3.2. Системи за свободно ползване с отворен код

Sonic Visualiser

Sonic Visualiser е аудио редактор, поддържащ множество файлови формати и предоставящ множество функции за обработка на аудио. Разпространен е под GNU лиценза. [15] На фиг. 1.18. и 1.19. са показани част от функционалностите, които системата предоставя.



Фиг. 1.18.



Фиг. 1.19.

Изглед на функционалностите на приложението.

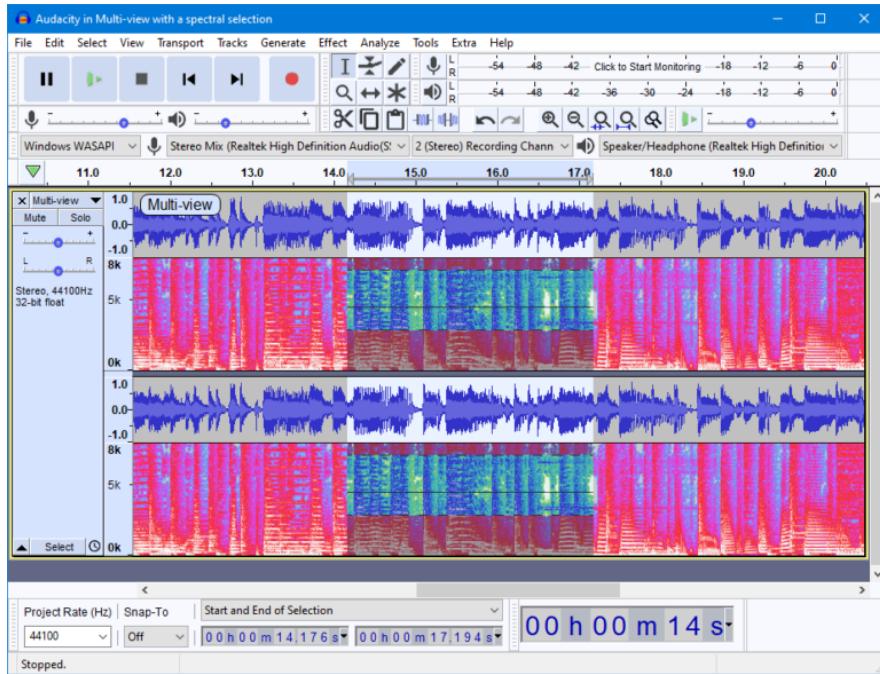
Характеристики:

- Многофункционален редактор;
- Интуитивен и лесен за работа;
- Възможност за запис.
- Измерването в реално време е трудно осъществимо.

Audacity

Audacity е лесен за използване, многоканален аудио редактор и рекордер за Windows, macOS, GNU / Linux и други операционни

системи. Разработено от група доброволци като отворен код.[16] На фиг. 1.20. е показана програмата.



Фиг. 1.20.

Изглед на функционалностите на приложението.

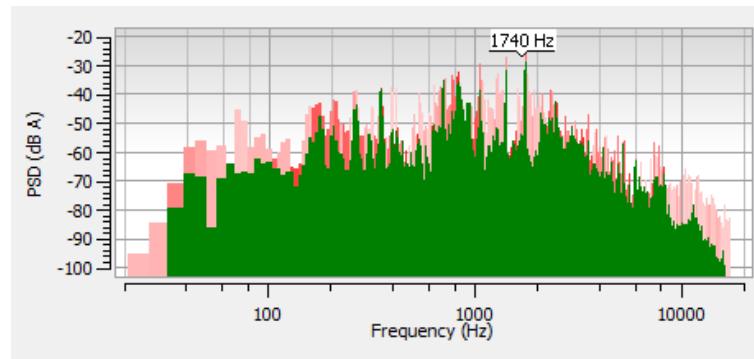
Характеристики:

- Многофункционален редактор;
- Възможност за запис.
- С отворен код;
- Има възможност някои от функционалностите му да работят в реално време.

Friture

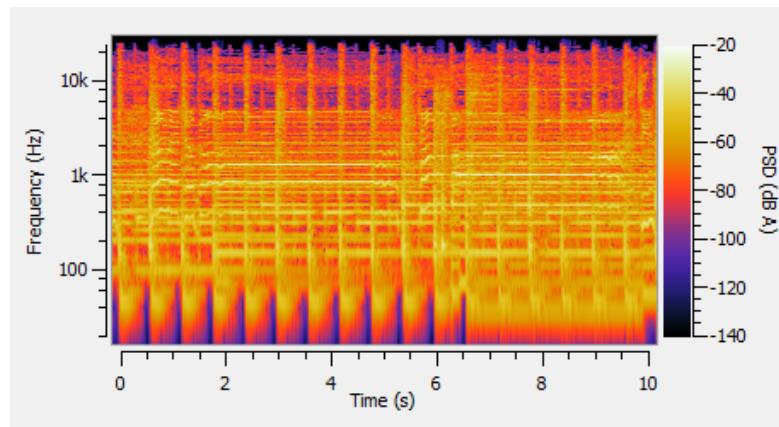
Friture е спектрален анализатор с отворен код и с възможност за визуализация на FFT спектър, октавен спектър, 2D спектrograma и т.н. и да управлявате съответните им настройки за анализ на аудио спектъра в реално време. За анализ на спектъра на FFT можете да

зададете максимална и минимална честота, размер на FFT, минимална и максимална амплитуда в dB и др. Също така може да генерира сигнали и да представя времевата функция на звука. Една от отличителните черти на този софтуер за аудио анализ е, че поддържа изглед с много прозорци, т.е. можете да добавите нови прозорци към интерфейса/менюто на приложението, за да анализирате различни спектрални изгледи едновременно. [17] На фиг. 1.21, 1.22 може да видите как изглеждат част от тези функционалности.



Фиг. 1.21.

Спектър генериран от приложението.



Фиг. 1.22.

Спектрограма генерирана от приложенито.

Характеристики:

- Анализатор, който може и да генерира определени синусоиди;
- 48kHz честота на дискретизация;
- Поддържа БПФ, Октаен спектър, осцилоскопен изглед, спектрограма;
- 2 канала за входен сигнал;
- Възможност за работа в реално време;
- Възможност за следене на множество сигнали едновременно;

В Табл. 1.3. е показано сравнение на разгледаните софтуерни системи за анализ, както комерсиални, така и свободно ползвани.

Таблица 1.3. Сравнение на софтуерни системи за анализ и обработка на звукови сигнали.

| Модел → Параметър ↓ | Sonic Visualiser | Audacity | Friture | Adobe Audition | Spectralissime |
|--------------------------------|---------------------------|----------------------------|---------------------------|---|--------------------------------------|
| Използване | Свободно / Отворен код | Свободно / Отворен код | Свободно / Отворен код | Платено | Опционални лицензи |
| Основна функция | Редактор | Редактор | Анализатор | Редактор | Анализатор |
| Честоти на дискретизация | - | 44.1kHz 48kHz и др. | 48kHz | 44.1kHz 48kHz 88.2kHz 96kHz и др. | 44.1kHz 48kHz 88.2kHz 96kHz |
| Големина на пробите | - | 16-bit 24-bit 32-bit | - | 16-bit 24-bit 32-bit | - |
| Работа в реално време | Не/Трудно | Възможно | Възможно | - | Да |
| Операционна система | Windows Linux macOS | Windows Linux macOS | Windows Linux macOS | Windows macOS | Windows |

Основните разлики между сравнените в таблицата системи освен начина на тяхното използване са по основната им функция, честотите

на дискретизация, работата в реално време и поддръжката на различни операционни системи.

1.4. Основни хардуерни платформи и компоненти, използвани при анализа на звукови сигнали

В тази точка ще бъдат разгледани различни платформи, базирани на микроконтролери, както и модули, които са подходящи за анализ на звукови сигнали, и ще бъде направено съответното сравнение и анализ. При това ще бъдат включени и част от компонентите, използвани в системите, описани в т. 1.2.2., за които има повече информация.

1.4.1. Микроконтролери и модули за безжична комуникация

За да бъде направен какъвто и да било звуков анализ е необходима съответна изчислителна мощност, особено ако микроконтролерът, който приема и записва звуковия сигнал в реално време, извършва и първоначалната му обработка. От друга страна визуализацията на първоначалния анализ и последващото му задълбочаване, макар и не в реално време, предполага използването на по-мощен микроконтролер. Поради тази причина по-долу ще бъдат разгледани платформи, базирани на различни по производителност микроконтролери. Ще бъдат разгледани и модули, които осигуряват безжична комуникация и позволяват връзка между различни микроконтролерни платформи.

Raspberry Pi 3

В тази точка ще бъде разгледан моделът Raspberry Pi 3 Model B+, който е показан на фиг. 1.23.[18]



Фиг. 1.23.
Изглед на Raspberry Pi 3.

Основни характеристики:

- Дължина - 85 mm;
- Ширина - 56 mm;
- Процесор - Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz;
- Работно напрежение - 3.3V;
- Захранващо напрежение - 5V DC (2.5A);
- Цифрови входове/изходи - 40;
- Поддържа USB интерфейс версия 2;
- Има цифрови интерфейси за видео и звук;
- Слот за външна памет - на базата на SD карта;
- SDRAM - 1GB LPDDR2;
- Връзка с Wi-Fi - 2.4 GHz и 5 GHz;
- Bluetooth 4.2, BLE;
- Gigabit ethernet;
- Работна температура - 0 - 50°C;

SX127X GPS HAT (Dragino LoRa/GPS HAT)

SX127X GPS HAT е модул за Raspberry Pi позволяващ използването на GPS и LoRa безжична комуникация.[19] На фиг. 1.24. е показан модълът.



Фиг. 1.24.
Изглед на SX127X GPS HAT.

Характеристики:

- Предварително конфигуриран с една от следните честоти - 868 MHz, 433 MHz, 915 MHz;
- Съвместимост с Raspberry Pi 2 Model B и Raspberry Pi 3 model B;
- Максимален бюджет на връзката: 168 dB;
- Вграден температурен сензор и сигнализатор за изтощена батерия;
- Максимална големина на пакет: 256 bit (с включено и CRC);
- Динамичен обхват RSSI: 127 dB;
- Ток протичащ на RX извода: 10.3 mA;
- Ток при запаметяване в регистър: 200 nA;
- Чувствителност: до -148 dB;
- Едноканален концентратор.

RAK831

RAK831 е многоканален високоэффективен приемо-предавателен модул, проектиран да приема няколко LoRa пакета едновременно, използвайки различни фактори на разпространение по множество канали. Модулът

предоставя възможност за осигуряване на стабилна комуникация между шлюз и огромно количество LoRa крайни възли, разпределени в широк диапазон от разстояния.[20] На фиг. 1.25. е показан модълът.



Фиг. 1.25.
Изглед на RAK831.

Характеристики:

- Предварително конфигуриран с една от следните честоти - 433 MHz, 470 MHz, 868 MHz, 915 MHz;
- Съвместимост с Raspberry Pi 3;
- Максимален бюджет на връзката: 162 dB;
- Максимална консумация до 2300 mA;
- Чувствителност: до -142.5 dB;
- Многоканален концентратор.

Arduino

Arduino е хардуерна платформа с отворен код и схеми, базирана на лесен за използване хардуер и софтуер. За да се програмира се използва специализиран език за програмиране на Arduino и софтуерната среда Arduino IDE.

През годините Arduino е мозъкът на хиляди проекти, от битови и ежедневни обекти до сложни научни инструменти. Световна общност от създатели - студенти, любители, художници, програмисти и професионалисти - се е събрала около тази платформа с отворен код,

техният принос е добавил невероятно количество достъпни знания, които могат да бъдат от голяма помощ както за начинаещи, така и за експерти.[21] Тук ще бъде разгледан модела Arduino Uno, който можете да видите на фиг. 1.26.



Фиг. 1.26.

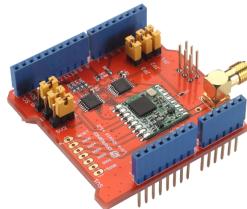
Изглед на Arduino Uno.

Основни характеристики:

- Дължина - 68.6 mm;
- Ширина - 53.4 mm;
- Тегло - 25 g;
- Микроконтролер - ATmega328P;
- Работно напрежение - 5V;
- Захранващо напрежение (препоръчително) - 7-12V;
- Захранващо напрежение (лимит) - 6-20V;
- Цифрови входове/изходи - 14;
- Шим входове/изходи - 6;
- Аналогови входове - 6;
- Ток през вход/изход (макс) - 20mA;
- Ток през 3.3V пин (макс) - 50mA;
- Flash памет - 32 KB, като 0.5 KB се използват от bootloader(бутлодъра);
- SRAM - 2 KB;
- EEPROM - 1 KB;
- Тактова честота - 16 MHz;

SX127x Shield за Arduino

SX127x Shield е модул за Arduino позволяващ използването на LoRa безжична комуникация. [22] На фиг. 1.27. е показан модулът.



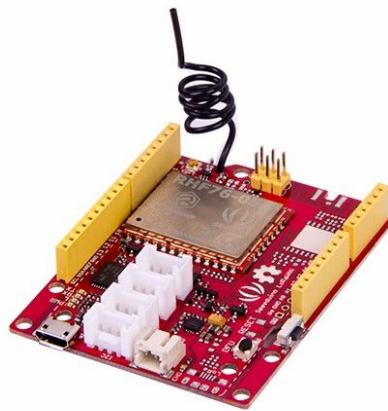
Фиг. 1.27.
Изглед на SX127x Shield за Arduino.

Подробности:

- Предварително конфигуриран с една от следните честоти - 868, 433, 915 MHz;
- Съвместим с Arduino Leonardo, Uno, Mega, DUE;
- Максимален бюджет на връзката: 168 dB;
- Максимална големина на пакет: 256 bit (с включено и CRC);
- Динамичен обхват RSSI: 127 dB;
- Ток протичащ на RX извода: 10.3 mA;
- Ток при запаметяване в регистър: 200 nA;
- Чувствителност: до -148 dB;

Seeeduino LoRaWAN

Seeeduino LoRaWAN е Arduino съвместима платформа с вградена LoRa комуникация. [23] Модулът е показан на фиг. 1.28.



Фиг. 1.28.
Изглед на Seeeduino LoRaWAN.

Основни характеристики:

- Дължина - 68 mm;
- Ширина - 53 mm;
- Тегло - 19.6 g;
- Микроконтроллер: ATSAMD21G18, 32-Bit ARM Cortex M0+;
- Работно напрежение - 3.3V;
- Захранващо напрежение: 3.7 V, чрез батерия;
- Цифрови входове/изходи - 20;
- Шим входове/изходи - Всички освен 2 и 7;
- Аналогови входове - 7 (6 с 12 битов ЦАП и 1 с 10 битов ЦАП);
- Ток през вход/изход - 7 mA;
- Flash памет - 256 KB;
- SRAM - 32 KB;
- Тактова честота: 48 MHz;
- Вграден микрочип за управление на литиева батерия;
- Вградена LoRa;
- Максимален бюджет на връзката: 160dB;
- Чувствителност: -140dBm;
- Двулентов, разработен за честоти: 434/470MHz и 868/915MHz;

MKR WAN 1300

MKR WAN 1300 е Arduino платформа съчетаваща функционалността на Arduino MKR Zero и LoRa комуникацията. [24] Модулът е показан на фиг. 1.29.



Фиг. 1.29.

Изглед на MKR WAN 1300.

Основни характеристики:

- Дължина - 67.64 mm;
- Ширина - 25 mm;
- Тегло - 32 g;
- Микроконтроллер: SAMD21 Cortex-M0+ 32bit low power ARM MCU;
- Работно напрежение - 3.3V;
- Захранващо напрежение: 5V;
- Цифрови входове/изходи - 20;
- Шим входове/изходи - Всички освен 2 и 7;
- Аналогови входове - 7 (6 с 12 битов ЦАП и 1 с 10 битов ЦАП);
- Ток през вход/изход - 7 mA;
- Flash памет - 256 KB;
- SRAM - 32 KB;
- Тактова честота: 32.768 kHz (RTC), 48 MHz;
- Вградена LoRa;
- Максимален бюджет на връзката: 168dB;
- Чувствителност: -140dBm;
- Разработен за честоти: 433/868/915 MHz;

ESP32

ESP32 е микроконтролер предназначен за IoT проекти с Wi-Fi и Bluetooth свързаност.[25] На фиг. 1.30. е показана NodeMCU хардуерна платформа с ESP32 микроконтролер.



Фиг. 1.30.
Изглед на ESP32.

Характеристики:

- Захранващо напрежение: 2.6 - 3.6V (препоръчително над 3.3V);
- Процесор Tensilica Xtensa 32-bit LX6;
- Тактова честота: до 240 MHz;
- Wi-Fi свързаност: 2.4 GHz (до 150 Mbit/s);
- Bluetooth: версия 4.2 и поддържа BLE;
- ROM: 448 KiB
- SRAM: 520 KiB
- Вградена флаш памет: поддържа само на чипа ESP32-D2WD chip (2 MiB) и на модула ESP32-PICO-D4 SiP module (4 MiB);
- Поддържа външна флаш памет и SRAM: до 16 MiB за код и до 8 MiB за данни;
- GPIO входове/изходи: 34;
- Работна температура: -40 - 125°C;

Сравнение

В табл. 1.4. е показано сравнение на микроконтролерите, които разглеждахме в тази точка.

Таблица 1.4. Сравнение на различни микроконтролери и развойни среди подходящи за реализирането на системата.

| Модул → Параметър ↓ | Raspberry Pi 3 model B+ | Arduino Uno | Seeeduino LoRaWAN | MKR WAN 1300 | ESP32 |
|--------------------------------|--|--|---|---|---|
| Размери | 85mm x 56mm | 68.6mm x 53.4mm | 68mm x 53mm | 67.64mm x 25mm | - |
| Тегло | - | 25g | 19.6g | 32g | - |
| Захранващо напрежение | 5V | 7-12V | 3.7V | 5V | 2.6-3.6V |
| Работно напрежение | 3.3V | 5V | 3.3V | 3.3V | 3.3V |
| Входове/изходи | 40 GPIO | 14 цифрови вход/изход и 6 аналогви входа | 20, 7 от които аналогови входа и 18 ШИМ | 20, 7 от които аналогови входа и 12 ШИМ | 34 GPIO |
| SRAM/SDRAM | 1GB LPDDR2 | 2 KB | 32 KB | 32 KB | 520 KiB |
| EEPROM/ROM | - | 1 KB | - | - | 448 KiB |
| Външна памет/Флаш памет | Външна SD карта | 32 KB | 256 KB | 256 KB | 0 MiB, 2 MiB или 4 MiB (зависи от модела) |
| Безжична връзка | Вградена Wi-Fi и Bluetooth Може да се добави LoRa | Може да се добави | Вградена LoRa | Вградена LoRa | Вградена Wi-Fi и Bluetooth |
| Тактова частота | 1.4GHz | 16Mhz | 48Mhz | 32.768 kHz (RTC), 48 MHz | до 240 MHz |
| Работна температура | 0 ~ 50°C | -40 ~ 85°C | - | - | -40 ~ 125°C |

1.4.2. Микрофони/модули (MEMS)

ADMP401

ADMP401 е модул с вграден MEMS микрофон, импеданс преобразувател и усилвател. Подходящ е както за изследване на близки разстояния, така и за изследване на далечни разстояния.[26] На фиг. 1.31. е показан модулът.



Фиг. 1.31.

Изглед на ADMP401.

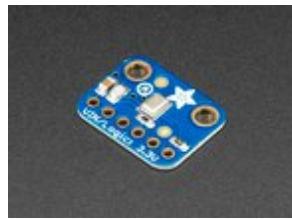
Характеристики:

- Захранващо напрежение: от 1.5 V до 3.3 V;
- Максимален ток на захранването (входен ток): 250 μ A;
- Честотен диапазон: 100 Hz до 15 kHz;
- Отношение сигнал-шум: 62 dBA;
- Работна температура: от -40°C до $+85^{\circ}\text{C}$;
- Динамичен обхват: 88 dB;
- Вече не се произвежда;

Adafruit I2S MEMS Microphone Breakout - SPH0645LM4H

Модул с вграден MEMS микрофон и I2S интерфейс за комуникация.

Предназначен за Cortex-M базирани устройства.[27] Модулът е показан на фиг. 1.32.



Фиг. 1.32.

Изглед на Adafruit I2S MEMS Microphone Breakout - SPH0645LM4H.

Подробности:

- Захранващо напрежение: от 1.62 V до 3.6 V;
- Максимален ток на захранването (входен ток): 600 μ A;
- Честотен диапазон: 50 Hz to 15 kHz;
- Отношение сигнал-шум: 65 dBA;
- Работна температура: от 0°C до 45°C;

В табл. 1.5 е онагледено сравнението между микрофоните, които бяха разгледани.

Таблица 1.5. Сравнение на различни микрофоni подходящи за реализиране на системата.

| Модул → Параметър ↓ | ADMP401 | Adafruit I2S MEMS Microphone |
|-------------------------|------------------|------------------------------|
| Технология на микрофона | MEMS | MEMS |
| Захранващо напрежение | 1.5 V до 3.3 V | 1.62 V до 3.6 V |
| Макс. входен ток | 250 μ A | 600 μ A |
| Честотен диапазон | 100 Hz до 15 kHz | 50 Hz to 15 kHz |
| Отношение сигнал-шум | 62 dBA | 65 dBA |
| Работна температура | -40°C ~ +85°C | 0°C ~ 45°C |
| Динамичен обхват | 88 dB | - |
| В производство | Не | Да |

1.4.3. Сензори за влажност и температура

BME680 модул

ВМЕ680 е сензор, който измерва температурата, атмосферно налягане, влажността на околната среда и наличие на летливи органични съединения. Осигурява висока надеждност, стабилност, ниска консумация и т.н. [28] На фиг. 1.33. е показан сензорът.



Фиг. 1.33.

Изглед на ВМЕ680 модул.

Параметри:

- Обхват на измерване на влажност: 0-100 %;
- Обхват на измерване на температура: -40-85 °C;
- Обхват на измерване на налягане: 300-1100 hPa;
- Интерфейс за комуникация: I2C;

1.5. Основни софтуерни компоненти за обработка на звукови сигнали

1.5.1. Библиотеки за обработка на звукови сигнали

Вече беше разгледана същността на ПФ и БПФ, а сега ще бъдат разгледани готови библиотеки, които да извършват преобразуванието на мястото на наш алгоритъм. Също така ще бъдат засегнати и библиотеки, които могат да послужат за последващ анализ.

arduinoFFT.h

Любителска библиотека създадена за Arduino, специално за прилагане на БПФ.[29]

Подробности:

- Автор: Enrique Condes;
- Текущ разработчик: Enrique Condes;
- Изчисленията са с точност след десетичната плаваща запетая, което прави съответната имплементация по-бавна, но сравнително точна;
- Съвместима с всички Arduino платформи.

fix_fft.h

Любителска библиотека създадена за Arduino, специално за прилагане на БПФ.[30]

Подробности:

- Автор: Dimitrios P. Bouras;
- Текущ разработчик: Enrique Condes;
- Изчисленията се закръгляват до цели числа, което прави преобразуванията много по-бързи, но по-неточни;
- Съвместима с Arduino Micro, Arduino Leonardo, Arduino Mega, Arduino Nano, Arduino Uno и Arduino Yún.

DSP библиотеки за Cortex M3 и други ARM процесори

Библиотеки създадени за цифрова обработка на сигнали (Digital Signal Processing) специално за Cortex M3 микроконтролери, но може да се използва върху други ARM контролери. [31]

Подробности:

- Поддържа БПФ;
- Добре документиран;
- Библиотеката притежава много функционалности;

CMSIS-DSP библиотека

Библиотека предназначена за обработка на сигнали върху Cortex-M и Cortex-A базирани устройства. [32]

Подробности:

- Предназначена е за Cortex-M и Cortex-A базирани устройства;
- Поддържа БПФ, базови математически функционалности, бързи математически функции, комплексни функции, филтриращи функции и други.

NumPy

NumPy (Numerical Python) е една от най-добрите библиотеки, снабдени с полезни ресурси, за да помогнат на учените да превърнат Python в мощен инструмент за научен анализ и моделиране. Популярната библиотека с отворен код се предлага под лиценза BSD. Това е основната библиотека на Python за изпълнение на задачи в научните изчисления. NumPy е част от по-голяма базирана на Python екосистема с инструменти с отворен код, наречена SciPy. Библиотеката предоставя на Python значителни структури от данни за безпроблемно създаване на многомерни масиви и извършване на изчисления на матрици. Освен използването си при решаване на уравнения на линейна алгебра и други математически изчисления, NumPy се използва и като универсален многоизмерен контейнер за различни видове общи данни. [33]

1.5.2. Софтуерни приложения и библиотеки за визуализация

Една от важните стъпки към осъществяването на системата е начина за визуализацията на спектъра, затова сега ще бъдат разгледани системи, с които могат да се визуализират данните и евентуално да се анализират или да се следи поведението на системата.

Grafana

Grafana е софтуер с отворен код, който позволява визуализиране, изследване, анализиране и други операции върху бази с данни. Също така има механизми за алармиране при необичайни явления (по SMS, Email и други). Безплатен за употреба върху личен сървър (по дефиниция/предварителна настройка се стартира на порт 3000), като може да се заплати за техен сървър, който дава и допълнителни възможности (като съхранение на log файлове и т.н.). Съвместима е с множество системи за бази данни сред които InfluxDB, PostgreSQL, MySQL, Redis и др. Има възможност да се инсталира върху Raspberry Pi.[34] На фиг. 1.34. е показано табло с графики създадено в Grafana.

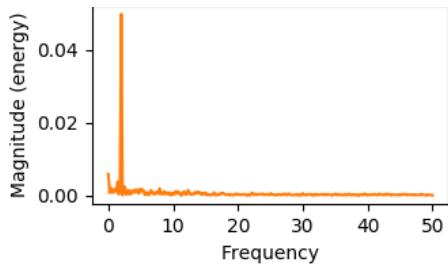


Фиг. 1.34.

Изглед на уеб приложението Grafana.

Matplotlib

Matplotlib е библиотека за създаване на статични, анимирани и интерактивни визуализации в Python. Библиотеката е проектирана да бъде толкова използваема, колкото MATLAB, с възможността да използва Python и има предимството да бъде бесплатна и с отворен код.[35] На фиг. 1.35. е показана графика, построена чрез помощта на библиотеката.



Фиг. 1.35.

Изглед на спектър генериран чрез Matplotlib.

Audacity

Audacity е софтуер, който беше разгледан в т. 1.3.2. Той позволява да бъде контролиран чрез външен процес. Може да се управлява от скриптови езици, които могат да използват именувани буфери на процеси, именувани канали (named pipe) като Python и Perl. Приложението може да се управлява и чрез макроси (macros), но те предоставят по-малко функционалности. Не е препоръчително да се използва върху уеб сървър, защото приложението не е подсигурено.[36]

1.6. Бази данни

Данните за анализ на звукови сигнали и параметри на околната среда, които ще се приемат, трябва да се записват и съхраняват в подходящ вид в съответната база с данни (БД), за да може впоследствие лесно и бързо да бъде извършен необходимия анализ върху тях. Поради това по-долу ще бъдат разгледани някои подходящи за целта системи за управление на база данни (СУБД).

1.6.1. InfluxDB

InfluxDB е база данни с времеви редове с отворен код, разработена от InfluxData. Той е написан на Go и е оптимизиран за бързо съхранение с

висока наличност и извличане на данни от времеви редове в области като мониторинг на операции, метрики на приложения, данни от сензори за IoT и анализи в реално време.

Базата използва два езика - Flux и InfluxQL за управление на данните. Има бесплатна версия, но предоставя и платени версии (облачна услуга и Enterprise услуга).[37] InfluxDB често се използва заедно със софтуера за визуализация и следене на данните Grafana, който беше разгледан в т. 1.5.2.

1.6.2. PostgreSQL

PostgreSQL е мощна обектно-релационна база данни с отворен код, която използва и разширява езика SQL, комбиниран с много функции, които безопасно съхраняват и мащабират най-сложните натоварвания от данни. Софтуерът е напълно безплатен, дори за комерсиални цели, като освен това се използва вече над 30 години.[38]

1.6.3. Redis

Redis е софтуер с отворен код (лицензиран според BSD лиценза), който се използва за хранилище за структури от данни в оперативната памет, използвана като посредник между друга база данни, кеширане на информацията и съобщенията. Той поддържа структури от данни като низове, хешове, списъци, набори, растерни карти, хиперлогове и др. Основната ѝ цел е да се ускори процеса на извлечане на данните от конкретната база данни. Съществува и платена версия на Redis - Redis Enterprise. [39]

1.6.4. Сравнение и анализ на БД

Табл. 1.6. Сравнителна таблица на различни СУБД, чрез които може да се реализира системата.

| Име → Параметър ↓ | InfluxDB | PostgreSQL | Redis |
|--|---|--|--|
| Вид | Времева база данни | Релационна база данни | Използва ключ-стойност модел |
| SQL | SQL подобен език | Да, с разширение | Не |
| Лиценз | MIT-License, налична е ѝ платена версия | BSD | BSD 3-Clause, налична е ѝ платена версия |
| Година на поява | 2013 | 1989 | 2009 |
| Операционна система | Linux OS X | FreeBSD HP-UX Linux NetBSD OpenBSD OS X Solaris Unix Windows | BSD Linux OS X Windows |
| ACID (Атомарност, последователност, изолираност, стабилност/трайност) принципи | Няма | Да | Частично покритие |
| Поддръжка на тригери | Няма | Да | Няма |
| Поддръжка на функции | Няма | Да | Могат да се пишат на Lua |
| Метод за достъп до данните | HTTP API JSON по UDP протокола | ADO.NET JDBC ODBC и др. | RESP - REdis Serialization Protocol |

По-детайлно сравнение на трите СУБД може да се намери на [40].

От таблицата по-горе може да се направи извода, че InfluxDB е добър компромис/баланс между трите разгледани системи.

Втора глава - Основни изисквания и блокови схеми

2.1. Основни изисквания към системата, проектирана в дипломната работа

Въз основа на направения обзор в първа глава по-долу са дефинирани основните изисквания към дипломната работа, като продължение и развитие на първоначалното задание, като те са разделени на няколко групи за по-голяма яснота.

2.1.1. Архитектурни и комуникационни изисквания

- Системата да се състои от две хардуерни платформи, изпълняващи ролята съответно на сървър и крайно устройство;
- Между двата вида устройства трябва да има комуникационна връзка, по която периодично да се обменя информация;
- Сървърната платформа трябва да може да поддържа връзка с повече от едно крайни устройства едновременно;
- Сървърът трябва да позволява отдалечен достъп за потребителя.

2.1.2. Функционални изисквания

- Крайното устройство трябва да приема и да обработва звуковата информация;
- В допълнение, крайното устройство трябва да следи температурата и влажността;
- Обработената информация трябва да се изпраща периодично на сървъра;
- Сървърът трябва да приема изпратената информация и да я записва в подходящ вид в съответна база данни;

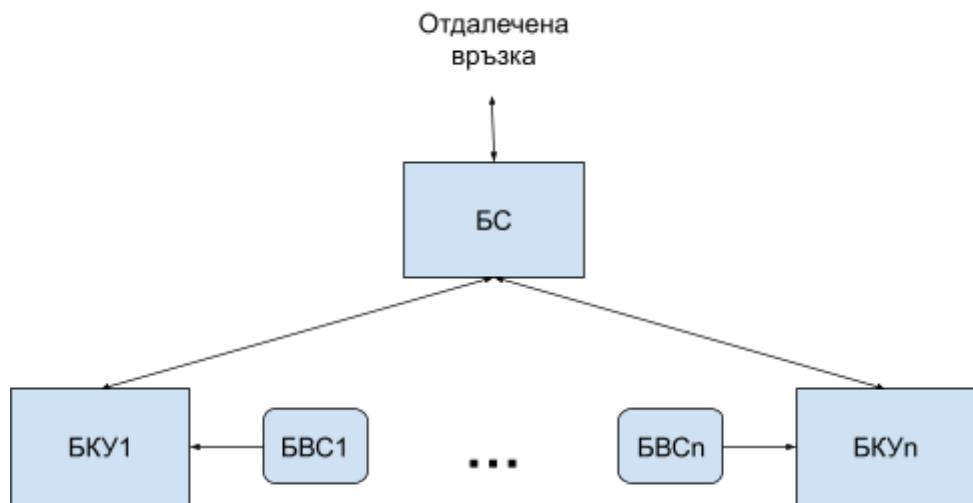
- Записаната информация трябва да може допълнително да се обработва и визуализира, като резултатите от тях трябва да са достъпни отдалечно;
- Сървърът трябва да може да изпраща конфигурационна информация на крайните устройства.

2.2. Основни блокови схеми

В следващите точки ще бъдат представени и описани основните блокови схеми на системата, взаимните връзки между блоковете и тяхната функционалност.

2.2.1. Обща блокова схема

На фиг. 2.1. е показана общата блок схема на проекта. Входните сигнали от сензорите (БВС, Блок входни сигнали) се подават на крайното устройство (БКУ, Блок крайно устройство), което извършва първоначалната обработка на данните. БКУ могат да бъдат повече от един. След първоначалната обработка всяко крайното устройство предава данните на сървър (БС, Блок сървър), който ги записва в съответна БД, и впоследствие може да извърши подходящ анализ и визуализация, както и да осигури отдалечен достъп на външен потребител до тях.



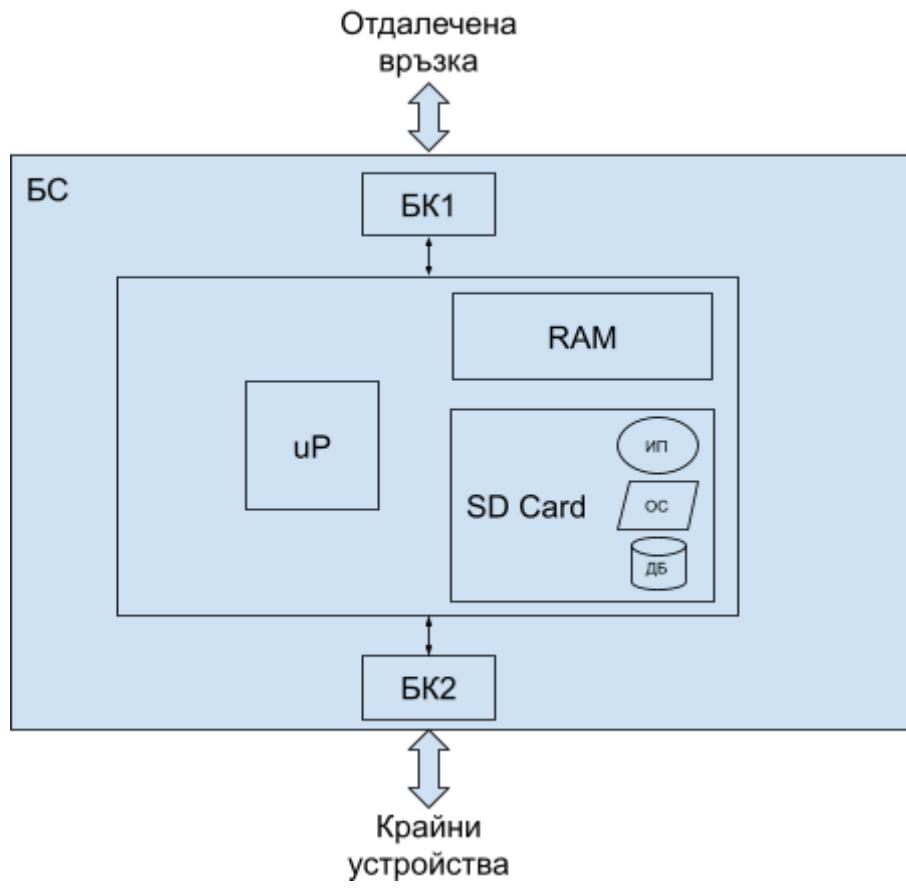
Фиг.2.1.
Основна блокова схема на системата.

2.2.2. По-детайлни блокови схеми на отделните блокове

В тази точка ще бъдат разгледани детайлно отделните блокове, които са елементи на общата блокова схема.

Блок сървър (БС)

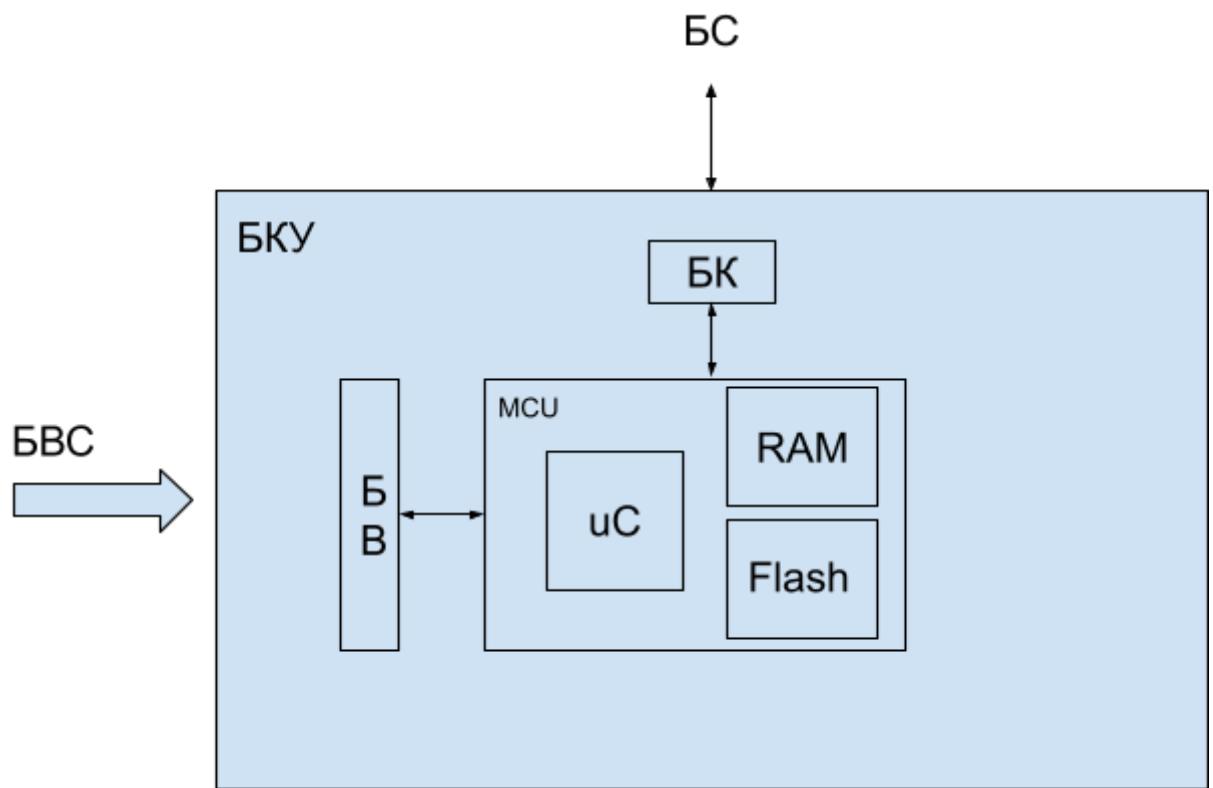
На фиг. 2.2. е показана по-подробна блокова схема на БС. Микропроцесорът (uP) е ядрото на БС, като нейните задачи са да приема информацията от различните БКУ чрез БК1 (Блок Комуникации 1), да я обработва, записва, а впоследствие да я анализира, визуализира и да осигурява отдалечен достъп до нея чрез втория комуникационен блок (БК2). За текущите данни uP използва оперативна памет (RAM), а операционната система (ОС), СУБД, както и изпълнимите програми (ИП) са записани в енергонезависима постоянна памет (ЕНПП).



Фиг. 2.2.
Блокова схема на Блок Сървър.

Блок крайно устройство (БКУ)

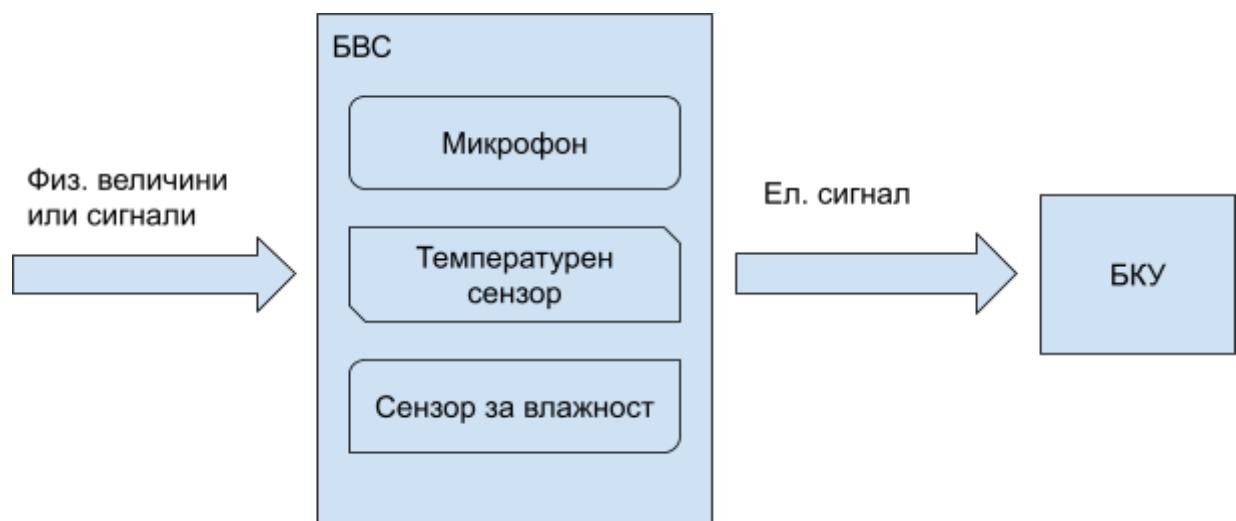
На фиг. 2.3. е показана по-детайлна блокова схема на Блок крайно устройство. Блок входове (БВ) приема входните сигнали от различните видове сензори от Блок входни сигнали (БВС) и ги подава на блок микроконтролер (uC). Микроконтролерът извършва първоначалната обработка и запис на данните в оперативната памет (Блок RAM), и впоследствие се предават на БС чрез Блок комуникации (БК). В постоянната памет (в случая Flash) се съхранява програмата, която изпълнява микроконтролера, както и някои постоянни данни. От страна на сървъра чрез БК към uC могат да постъпят конфигурационни данни и някои параметри.



Фиг. 2.3.
Блокова схема на Блок крайно устройство.

Блок входни сигнали (БВС)

На фиг. 2.4. е показана по-детайлна блокова схема на Блок входни сигнали. На входа са подадени различни физични величини и сигнали като звук, температура на околната среда и влажност на околната страна. Сигналите биват преобразувани от сензорите (микрофон, сензор за температура и сензор за влажност) в електрически (аналогови или цифрови) сигнали, които биват предадени към БКУ.



Фиг. 2.4.
Блокова схема на Блок входни сигнали.

Трета глава - Проектиране на принципна електрическа схема на системата с използване и на схеми на готови модули

3.1. Избор на основни компоненти

3.1.1. Избор на CAD система

Circuit Maker

CircuitMaker е професионална, бесплатна CAD система с отворен код разработена от Altium, която разполага с мощно маршрутизиране, иерархично въвеждане на схеми, автоматично маршрутизиране и интуитивна 3D™ технология. Освен това CircuitMaker разполага с обширна общност от професионалисти и ентузиасти, които я разработват или използват. CircuitMaker върви под операционната система Windows. Системата няма ограничения по отношение на сложността на проектирани схеми и платки, за разлика от други подобни системи. [41]

3.1.2. Избор на готови модули за Блок сървър

Raspberry Pi 3 model B+

За сървърен компонент ще бъде използвано Raspberry Pi 3 model B+. Съществуващата готова принципната електрическа схема е показана на Приложение 1 (П1) и е достъпна на сайта на производителя. [42]

RAK831

За LoRa безжична комуникация ще бъде използван разширителния модул RAK831. Съществуващата готова принципната електрическа схема е показана в П2, а повече детайли има на страницата на производителя. [43]

3.1.3. Избор на Блок крайно устройство

Тук ще бъде направен опит да се реализира системата не само с повече от едно крайно устройство, но и с крайни устройства, които са базирани на различни хардуерни платформи. Този опит ще бъде полезен за събирането на полезни данни за поведението и изпълнението на поставените задачи от различните хардуерни платформи.

Вариант 1 - Seeeduino LoRaWAN

Едното крайно устройство ще бъде базирано на модула Seeeduino LoRaWAN, който съчетава Ардуино платформата с вграден LoRa модул за комуникация. В П3 са показани съществуващите принципни електрически схеми, които могат да се намерят на сайта на производителя. [44]

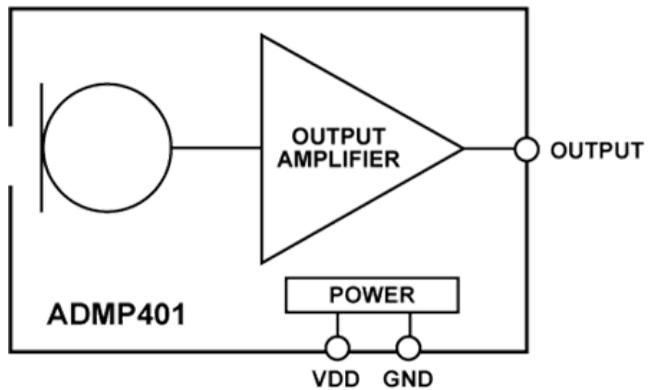
Вариант 2 - MKR WAN 1300

Второто крайно устройство ще бъде базирано на модул Arduino MKR WAN 1300, който беше разгледан в т. 1.4. Съществуващите принципни електрическите схеми на устройството могат да бъдат намерени в П4 или на сайта на производителя. [45]

3.1.4. Избор на компоненти на Блок входни сигнали

ADMP401 модул

За микрофон ще бъде избран ADMP401 модулът, на фиг. 3.1. е показана блок схемата му. В П5 е показана съществуващата принципна електрическа схема на модула, която може да бъде намерена и онлайн.[46]



Фиг. 3.1.

BME680 Breakout

За отчитане на влажност и температура ще бъде използван модул базиран на BME680, като в П6 е показана съществуващата принципна електрическа схема на модула. Схемата не е публикувана онлайн, но беше получена след изрично запитване към производителя на модула.

3.2. Проектиране на принципна електрическа схема за Блок крайно устройство

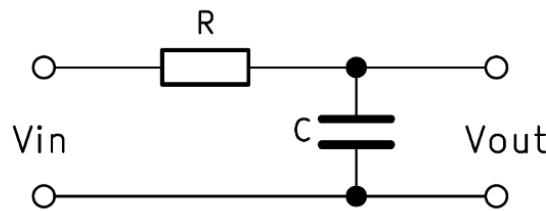
Връзката между микроконтролера и BME680 модула се осъществява по интерфейса I²C, а ADMP401 модула подава звуковия сигнал към аналоговия вход на микроконтролера. В тази точка ще бъдат разгледани двата варианта на електрическата схема на крайните устройства в зависимост коя платформа се използва. На схемата има два допълнителни извода, чиято идея е по тях да се подава захранващото напрежение към другите компоненти от веригата.

Според различни проучвания пчелата като самостоятелен индивид може да издава звукове до 6.5 kHz, следователно към системата беше проектиран нискочестотен пасивен RC филтър от първи ред с ограничаваща честота от приблизително 8 kHz. Филтърът работи на базата на делител на напрежение, като (реактивната част на)

импеданса на кондензатора се влияе от честота на сигнала (виж формула 3.1.).

$$X_c = -\frac{1}{2\pi f C} \quad (3.1.)$$

На фиг. 3.2. е показана принципната схема на свързване на филтъра.

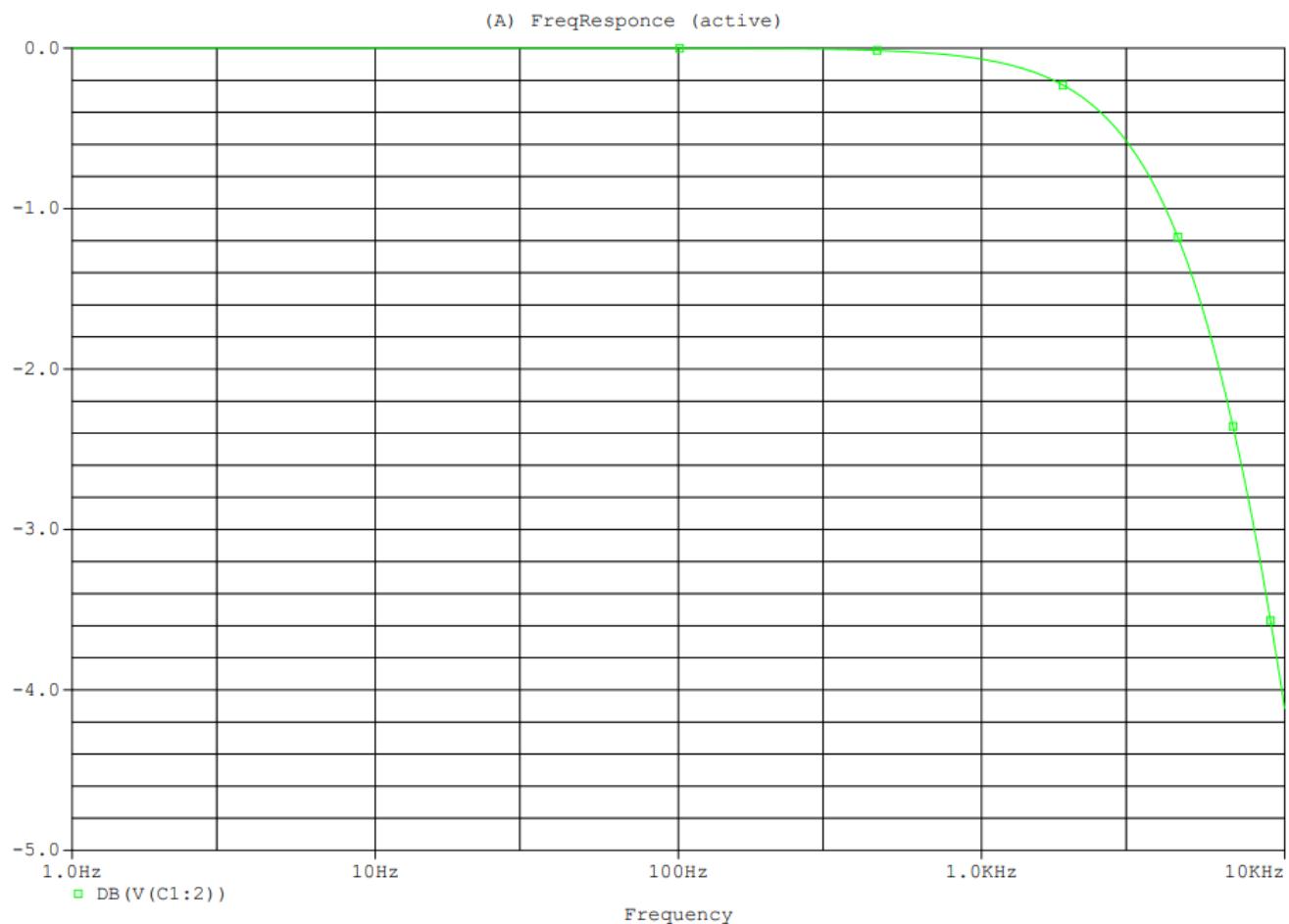


Фиг. 3.2.

Максималната честота на сигнала (изразващата честота, cut-off frequency) f_c , която филтърът пропуска може да се изчисли по формула 3.2.

$$f_c = \frac{1}{2\pi RC} \quad (3.2.)$$

В уравнението има две неизвестни R и C , затова за удобство бива избиран за C стандартна стойност от 100nF , при което се получава $R = 198.9435\Omega$, което може да се приближи до 200Ω или два резистора със стандартна стойност от по 100Ω . Амплитудно-честотната характеристика на филтъра е показана на фиг. 3.3.



Фиг. 3.3.

По абсцисата е честотата в логаритмичен мащаб, а по ординатата - затихването в децибели.

За проектиране на филтъра в OrCAD беше използван "VAC" източник като "DC Offset" беше зададен да е 1.6V, защото на изхода на микрофона има приблизително 1.6V при липса на звукови сигнали. За амплитуда на сигнала беше избран 1Vpp, а честота беше зададена в интервала от 1Hz до 10kHz.

Вижда се, че граничната честота на филтъра (на -3 dB) е около 8 KHz (по формула 3.2. при зададените по-горе стойности на компонентите се получава 7.957kHz).

3.2.1. Вариант 1 - Seeeduino LoRaWAN

Seeeduino LoRaWAN се захранва от JST2.0 конектор, затова на схемата има 4 извода за захранване, два предназначени да се върже JST2.0 конектора към тях и два, по които да се подаде захранващото напрежение. Принципната електрическа схема може да се намери в П7. Проектът от своя страна може да бъде намерен в CircuitMaker.[48]

3.2.2. Вариант 2 - MKR WAN 1300

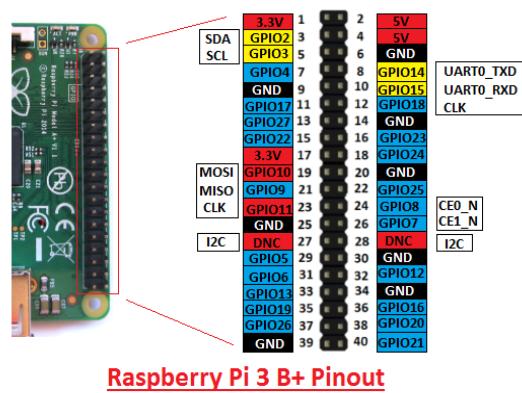
Принципната електрическа схема на този вариант може да се види в П8. Проектът от своя страна може да бъде намерен в CircuitMaker.[49]

Четвърта глава - Проектиране на печатна платка

4.1. Печатни платки на използваните модули

4.1.1. Raspberry Pi 3 model B+

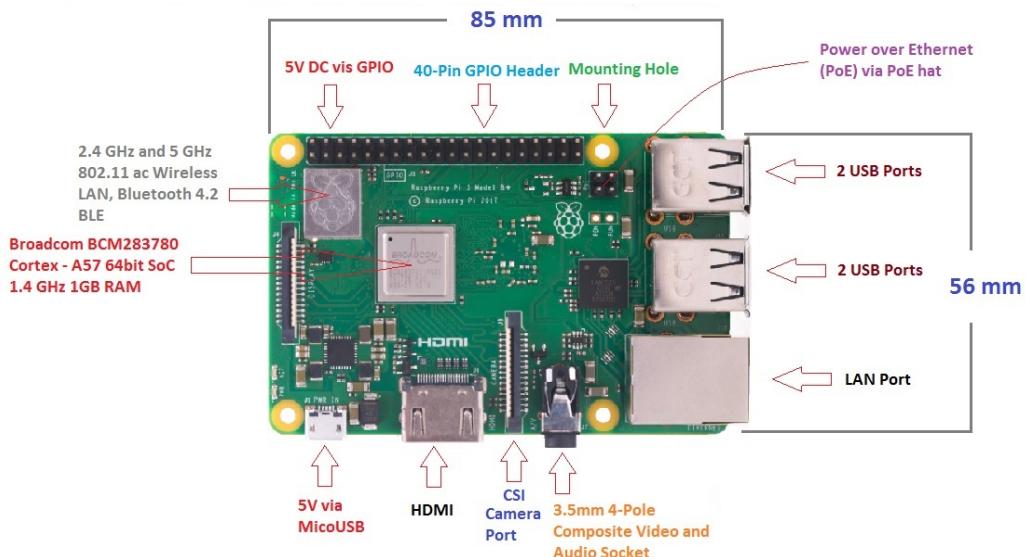
На фиг. 4.1. са показани изводите на хардуерната платформа. Платката е показана на фиг. 4.2.



Raspberry Pi 3 B+ Pinout

Фиг. 4.1.

Изводи на Raspberry Pi 3 model B+.

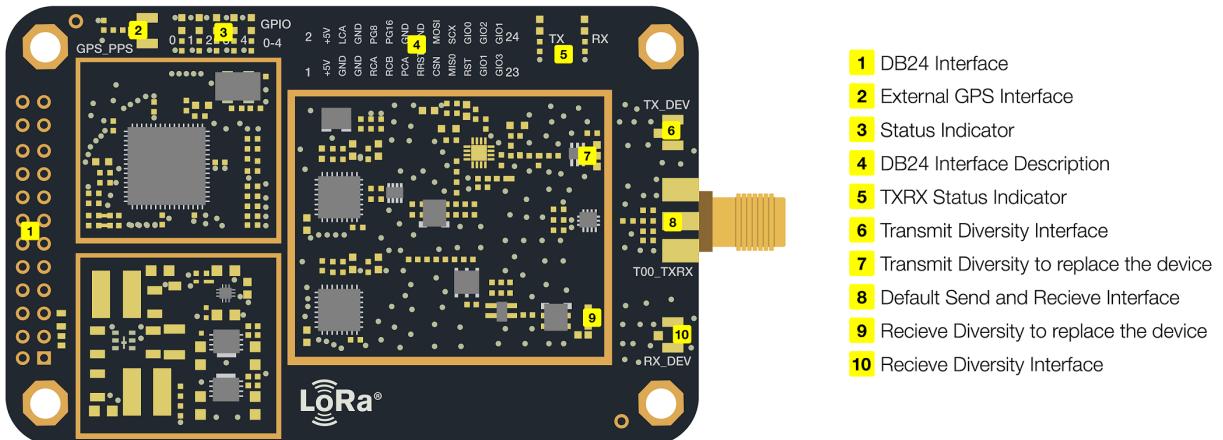


Фиг. 4.2.

Печатна платка на Raspberry Pi 3 model B+.

4.1.2. RAK831

На фиг. 4.3. Е показана печатната платка с изводите ѝ.

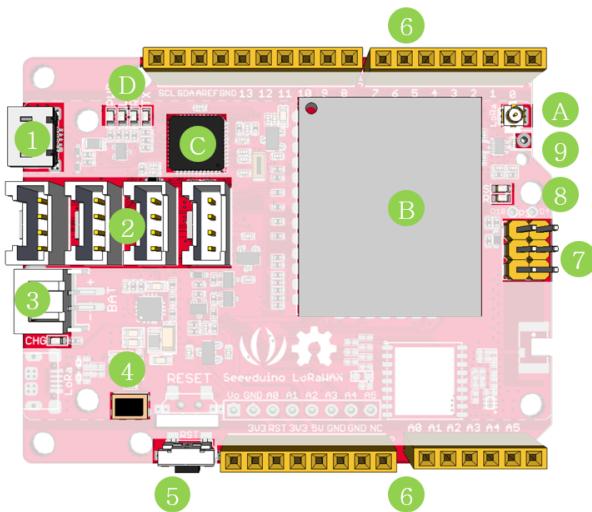


Фиг. 4.3.

Печатна платка на RAK831.

4.1.3. Seeeduino LoRaWAN

На фиг. 4.4. са дадени изводите на устройството. Модел с размери и 3D дизайн на платката може да се намери на сайта на производителя. [45]



Фиг. 4.4.

Печатна платка на Seeeduino LoRawan.

Означенията на печатната платка са следните:

1. Micro USB - Чрез него може да се захранва и програмира устройството;
2. Grove конектори
3. JST2.0 вход за липо батерия (3.7V) и диод за индикация
4. DFU Button - Бутон за режим на фърмуера
5. Reset бутон
6. Стандартни изходи на Ардуино
7. ICSP изводи
8. Светодиод за режим на фърмуера
9. Проводникова антена
 - A. uFL антена
 - B. RF модул - RHF76-052AM
 - C. ARM Cortex M0 процесор - ATSAMD21G18
 - D. Светодиоди
 - RX/TX - светодиод индикиращ трансфер по UART интерфейса
 - L - светодиод свързан към D13
 - PWR - светодиод индикиращ захранване

4.1.3. MKR WAN 1300

Модел на печатна платка може да се намери на сайта на производителя, в zip проект, проектиран на EAGLE.[24] Отделно може да бъде намерен pdf с модел на платката и разположението на изводите в П9 и на сайта на производителя.[50]

4.1.4. ADMP401 модул

Модел на печатната платка може да бъде намерен на сайта на производителя, откъдето може да се изтегли EAGLE проект с електрическа схема и печатна платка. [51]

4.1.5. BME680 pimoroni модул

Печатната платка на конкретния модул не беше намерена, но на сайта на производителя са предоставени размерите му - 19x19x2.75 mm (LxWxH), по които може да се направи изображение (корпус).

4.2. Проектиране на печатна платка за Блок крайно устройство

В тази точка ще бъдат представени двата варианта на печатни платки на крайните устройства в зависимост от това коя платформа се използва.

4.2.1. Вариант 1 - Seeeduino LoRaWAN

Проектираната печатна платка на вариант 1 може да се види в П7. Проектът от своя страна може да бъде намерен в CircuitMaker.[48]

4.2.2. Вариант 2 - MKR WAN 1300

Печатната платка на вариант 2 може да се види в П8. Проектът от своя страна може да бъде намерен в CircuitMaker.[49]

Пета глава - Проектиране на управляващ софтуер

5.1. Използван софтуер, езици и библиотеки

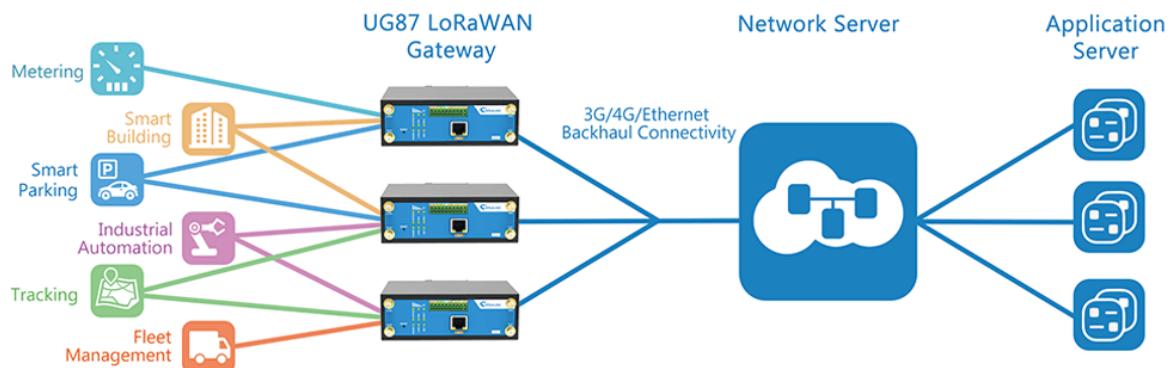
В тази точка ще бъдат разгледани използваните протоколи, библиотеки, системи за управление на бази данни и друг софтуер използван в конкретната дипломна работа.

5.1.1. LoRaWAN протокол за комуникация между Блок сървър и Блок крайно устройство

Данните между Блок крайно устройство и Блок сървър се изпращат по LoRa (Long Range) технологията, която е техника за модулация на разширен спектър, базирана на разширяване на спектъра чрез метода на линейната честотна модулация (Chirp spread spectrum).

Комуникацията между Блок крайно устройство и Блок сървър е реализирана посредством LoRaWAN протокола, чиято структура е показана на фиг. 5.1. [51][52]

Application Example



Фиг. 5.1.
Структура на LoRaWAN протокола.

Крайните устройства изпращат данни посредством LoRa към LoRaWAN шлюзове/концентраторите (gateways), които разпределят получените данни към LoRaWAN мрежови сървър. Мрежовият сървър се грижи да отсее повтарящите се данни и да ги предаде на приложния сървър. Важно е да се отбележи, че крайното устройство не изпраща данните си на конкретен шлюз, а ги разпространява към всички (broadcast) и данните се приемат от шлюзовете в обсега на сигнала.

Има два начина за автентикация при LoRaWAN, това са OTAA (Over the air activation) и ABP (Activation by personalization).

При OTAA крайното устройство и сървъра пазят DevEUI (64 битов уникален идентификатор на устройството), AppEUI (64 битов уникален идентификатор на приложението) и AppKey, който се използва за генериране на ключ на сесията между крайното устройство и мрежовия сървър (Network session key, NwkSKey) и ключ на сесията между крайното устройство и сървъра (Application session key, AppSKey).

NwkSKey и AppSKey са ключове, които се генерират по време на автентикация, пазят се от сървъра и от крайното устройство и служат за криптиране и декриптиране на информацията между устройството и сървъра. Също така се създава и DevAddr, който има подобна функция на IP адрес в една Ethernet мрежа.

При ABP крайното устройство и сървъра предварително имат създадени NtwSKey, AppSKey и DevAddr, което позволява на крайното устройство директно да изпраща данните си на приложния сървър, без да е необходимо да премине през процедурата за генериране на ключове.

Системата, която е разгледана в тази дипломна работа, използва метода OTAA.

В LoRaWAN има три вида реализация на крайни възли или по правилно е да се нарекат три класа устройства. Преди описанието им е важно да се отбележи, че всеки следващ клас трябва да имплементира възможностите на предишния. Класовете са съответно A, B и C.

Клас А (Class A) устройство изпраща данните си и след това има два прозореца, времеви периода, в които може да получи данни от концентратора. Ако концентратора не изпрати данни точно в интервала от време, в който устройството приема, устройството ще "заспи" и ще може да получи данни от концентратора, чак когато отново изпрати данни към него и след това отвори двета "прозореца", в които приема данни. Клас А са най-икономични, защото работят най-малко време и този клас устройство е проектиран в текущата дипломна работа.

Клас В (Class B) устройство работи като клас А, но освен двета задължителни прозореца може да отваря още "прозорци" като между него и концентратора съответно се предава синхронизираща информация, за да може концентратора да "знае" кога крайния възел "слуша".

Клас С (Class C) устройство "слуша" почти през цялото време, като затваря "прозорците" само когато трябва да изпраща съобщения и следователно консумира най-много енергия.

На посочения линк в литературата може да намерите 1.0.2. спецификацията на LoRaWAN протокола, която е имплементирана в конкретната дипломна работа.[53]

5.1.2. ChirpStack и rak_common_for_gateway

ChirpStack е система с отворен код и MIT лиценз, която съчетава софтуер за комуникация между шлюза и мрежовия сървър, мрежови сървър и приложен сървър. [54]

rak_common_for_gateway е хранилище, което предоставя, от RAKWireless, който в съчетание с gateway-bridge софтуера на ChirpStack приема данните и ги изпраща на мрежовия сървър. [55]

5.1.3. Adafruit BME680 Library

Това е библиотеката, с помощта на която се четат данните от BME680.

Хранилището, в което има линк към документация и описание на необходимия допълнителен софтуер, може да намерите на посочения линк. [56]

В библиотеката има един клас - Adafruit_BME680, ако на конструктора не се подадат параметри класът се инициализира с връзка по I²C интерфейса. На създадения обект трябва да се извика методът begin() с адрес 0x76 в конкретния случай на дипломната работа.

```
while (!bme680->begin(0x76)) {  
    delay(5000);  
}  
bme680->setTemperatureOversampling(BME680_OS_4X);  
bme680->setHumidityOversampling(BME680_OS_2X);  
bme680->setPressureOversampling(BME680_OS_4X);  
bme680->setIIRFilterSize(BME680_FILTER_SIZE_3);  
bme680->setGasHeater(320, 150); // 320°C for 150 ms
```

Фиг. 5.2

Извадка от сурс кода на системата, отговарящ за инициализация на BME680 модула.

В извадения пример от фиг. 5.2. от разработката на дипломната работа има указател към обекта, на който се извикват методите. Това, което се случва е, че се задава адреса да е 0x76, който е на съответния модул и се задава честота на семплиране на температурата 4 пъти по-голяма от минималната (по теоремата на Котелников/Найкуист, виж формула 1.1.), честота на семплиране на влажност 2 пъти по-голяма от минималната и честота на семплиране 4 пъти по-голяма за налягането. Задава се филтьрът да работи с 3 семпъла/проби и се задава сензора за газ да работи на 320°C и забавянето му е 150 ms.

5.1.4. MKRWAN.h

Това е библиотеката, необходима на Arduino MKR WAN 1300, за да осъществи LoRaWAN комуникация. [57] Библиотеката няма контрол върху различните предавателни канали, защото те са строго зададени върху фърмуера(firmware) на CMWX1ZZABZ-78 модула.

5.1.5. LoRaWan.h

Това е библиотеката, необходима на Seeeduino LoRaWAN, за да осъществи LoRaWAN комуникация. [58] Библиотеката има контрол върху различните предавателни канали.

5.1.6. Adafruit ZeroFFT Library

Това е библиотеката, която извършва БПФ преобразуванието върху събранные аудио преби.[59]. Тя има три основни функции:

- FFT_BIN(num, fs, size) - връща честотите, по които "сравнява", които са базирани на дискретизиращата честота;
- FFT_INDEX(freq, fs, size) - връща честотата, по даден индекс;
- int ZeroFFT(q15_t *source, uint16_t length) - взима масив от аудио преби и изчислява индекс, стойност за всяка честота.
Стойностите се записват в масива с преби, следователно информацията за пребите бива загубена;

5.1.7. InfluxDB

За съхранение на данните от крайните устройства се използва InfluxDB.[60] В InfluxDB се поддържат два езика - InfluxQL и Flux, в рамките на дипломната работа ще се използва InfluxQL. В InfluxDB данните се записват по специфичен течен протокол "write protocol" или "line protocol", който има следната структура изобразена на фиг. 5.3.

```
<measurement>[,<tag_key>=<tag_value>[,<tag_key>=<tag_value>]]  
<field_key>=<field_value>[,<field_key>=<field_value>]  
[<timestamp>]
```

Фиг. 5.3.

Протокол за запис на данни в InfluxDB.

За записване на данните от приложния LoRaWAN сървър в базата с данни е използвана клиентската библиотека за python на InfluxDB.[61]

За да има съвместимост между версия 1.8., която се използва в дипломната работа и версия 2.0. и нейните подобрения на InfluxDB, данните се изпращат на адрес /api/v2/write. Върху InfluxDB предварително бива създаден администраторски потребител admin и евентуално друг потребител, който да използва базата с данни.

Потребителите, които не са администратори, може да се създадат с ограничени права, като например им се отнеме правото да изтриват измервания и точки (points) - подобни са на редовете в релационните бази данни. Също така трябва да бъде създадена база с данни, в която да се записват данните и като “bucket” в конкретната дипломна работа е подадено “test/”, не е подадено “retention policy”, защото не е създавано такова и се ползва създадено по подразбиране. Клиентската библиотека позволява данните да бъдат подадени по пет начина:

- Като низ директно под формата на “write protocol”;
- Като специфична структура Point;
- Като речник, това е методът, който се използва в конкретната дипломна работа;
- Като “Observable stream”;
- Като “Pandas DataFrame”;

5.1.8. Grafana

За визуализация и следене на данните се използва Grafana, чиито предимства бяха разгледани в точка 1.5.2.

5.2. Инсталиране на необходимия софтуер върху Блок сървър

В тази точка ще бъдат разгледани стъпките през инсталацията на необходимия софтуер върху Блок сървър и необходимите настройки.

5.2.1. Инсталиране на ОС върху БС

За свързване и инсталиране и настройване на RPi се следва следния справочник. [62] В конкретната система е използвана 16 GB SD карта и е инсталирана препоръчаната версия от Raspberry Pi Imager.

5.2.2. LoRaWAN network сървър, приложен сървър, gateway-bridge и rak_common_for_gateway

Приложените програми се инсталират върху Raspberry Pi, който ползва Debian базирана операционна система, затова за инсталацията на ChirpStack приложенията се следва главата “Quickstart Debian or Ubuntu” [63]. Накратко трябва да се изтеглят - mosquitto, mosquitto-clients, redis-server, redis-tools и postgresql, чрез командата sudo apt install. Тези приложения са необходими за вътрешната работа на отделните компоненти на ChirpStack. Трябва също да се изтеглят Gateway-bridge, мрежовия сървър и приложния сървър. Трябва да се изтегли и управляващия софтуер за съответното разширение за RPi разгледано в точка 3.1.1. Инструкциите за инсталiranе на софтуера могат да се намерят в хранилището.[55]

5.2.3. Grafana

Grafana трябва да се инсталира върху системата, след което да се свърже към “източник”, тоест да се свърже към базата с данни.

В Grafana могат да се създадат потребители с права единствено да гледат определени табла. Таблата могат да се групират в директории, което също е удобно, защото различните устройства на различните

потребители могат да бъдат групирани. Неудобно е, че за всеки потребител трябва да се създават и определят правата ръчно. За всяко различно устройство трябва да се променят много от параметрите, като например devEUI, което е индексирано таг поле в базата с данни, по което се идентифицират различните устройства в конкретната разработка на дипломната работа.

След като бъде активиран smtp протокола в конфигурационния файл и бъде създаден имейл профил, от който да се изпращат имейли и бъдат въведени данните на този имейл профил в конфигурационния файл, могат да бъдат изпращани имейли към потребителите. Например в дипломната работа биват изпращани известия, предупреждения при много високи честоти или много ниски честоти. Същото е направено и за амплитудните измервания. Може да намерите линк с обяснение как се настройва smtp протокола в литературата. [64]

5.2.4. Influx DB

Преди инсталацията системата и пакетите ѝ биват обновени до най-нова версия. След което се добавя ключът на хранилището на InfluxDB, което ще позволи на RPI да търси пакетите необходими за инсталацията на субд-то. След което се добавя и самото хранилище в “източници”, “sources” на пакетния мениджър, добавя се Buster версията. Препоръчително е системата отново да се обнови. InfluxDB се инсталира чрез пакетния мениджър, след което се активира “service” файла на субд-то, което му позволява да се стартира автоматично при всяко пускане на системата. На фиг. 5.5. могат да се видят командите използвани за инсталацията и конфигурацията в разглежданата система.

```
$ sudo apt update
$ sudo apt upgrade
$ wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key
add -
$ echo "deb https://repos.influxdata.com/debian buster stable" | |
sudo tee /etc/apt/sources.list.d/influxdb.list
$ sudo apt update
$ sudo apt install influxdb
$ sudo systemctl unmask influxdb
$ sudo systemctl enable influxdb
$ sudo systemctl start influxdb
```

Фиг. 5.5.

Команди за инсталация на InfluxDB.

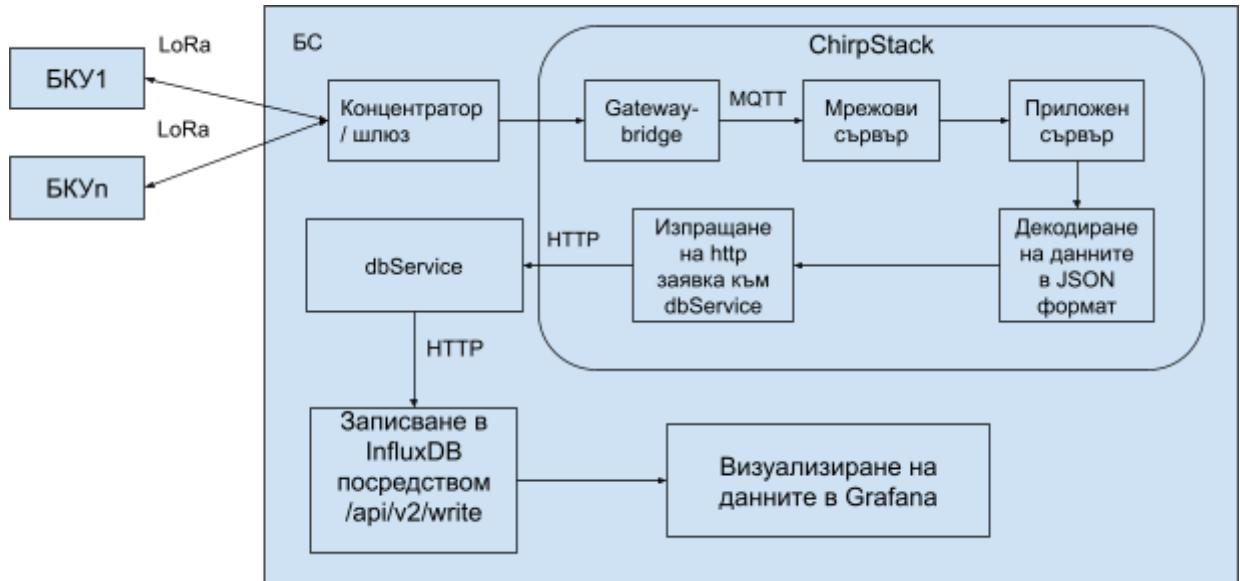
След като се инсталира субд-то, трява да се създаде администраторски профил, на който да се дадат привилегии. Трябва да се създават съответните бази данни, върху които ще се съхраняват данните. Линк към използвания справочник по време на инсталацията може да бъде намерен в литературата.[65]

5.3. Управляващ софтуер на Блок сървър

В тази точка ще бъде разгледана по-подробно как работи системата върху Блок сървър.

5.3.1. Блокови схеми и принципни схеми на БС

Принципната схема, която показва връзките между различните компоненти на системата е показана на фиг. 5.6.



Фиг. 5.6.
Принципна схема на системата.

LoRa не е предназначена за пренос на голямо количество информация, също така количеството на информацията, която може да се пренася по LoRaWAN технологията има регионални ограничения, които трябва да се спазват. Затова е имплементиран протокол, който е с променлива големина на полезната информация. Тоест ако трябва да се из pratят данни само за честота и амплитуда ще бъдат из pratени само 8 байта информация, ако трябва да се из pratят данни за всички параметри, които системата следи - температура, влажност, налягане, съпротивление на газовия сензор, амплитуда, честота и тестови брояч полезната информация ще е общо 26 байта. Към тези изчисления винаги трябва да се добави още един байт, който е първи в протокола и всеки негов бит е флаг за това кои параметри са из pratени по протокола.

На фиг. 5.7. и 5.8. е изображен процесът по формиране на протокола.

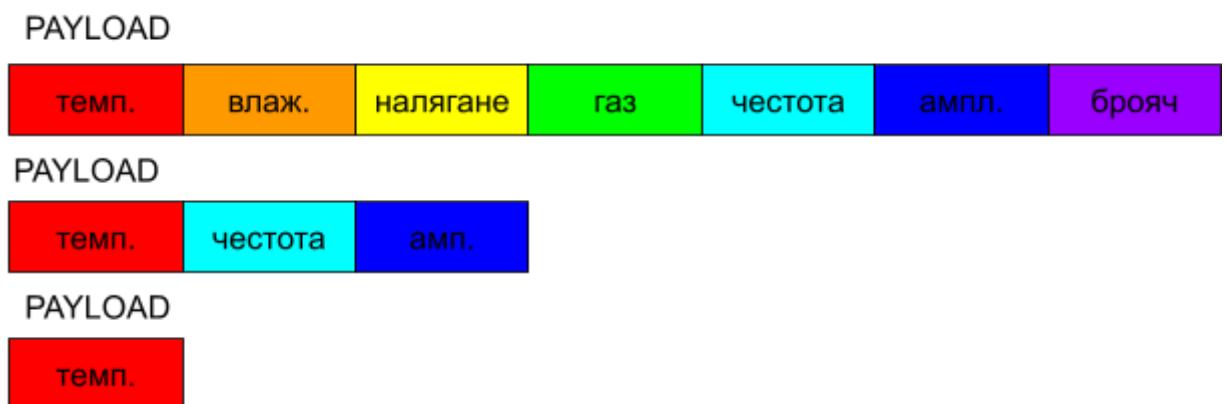


01111111

Легенда

- | | |
|---|-----------------------------------|
| ■ | - празен бит |
| ■ | - флаг за температура |
| ■ | - флаг за влажност |
| ■ | - флаг за налягане |
| ■ | - флаг за газ |
| ■ | - флаг за честота |
| ■ | - флаг за амплитуда |
| ■ | - флаг за брояч - тестови функции |

Фиг. 5.7.



Фиг. 5.8.

Имплементация на протокол за пренос на полезна информация по LoRaWAN.

Ако по протокола се пренася информация за някой от параметрите, неговият флаг ще е “1”, ако не се пренася информация за този параметър флагът му ще е “0”. Първият бит от заглавния байт е празен, което позволява на протокола да бъде разширен в бъдеще.

5.3.2. Сурс код и настройки по Grafana и ChirpStack приложния сървър на БС

В тази подточка е показан кодът, който декодира данните, които се пренасят по протокола обяснен в точка 5.3.1. Декодиране се осъществява чрез побитово отместване, а функцията е реализирана на езика JavaScript, защото това е езикът, който поддържа интерпретатора на декодиращата функция интегриран в приложния LoRaWAN сървър. Конкретната имплементация на функцията е изложена на фиг. 5.9.

```
// Decode decodes an array of bytes into an object.
// - fPort contains the LoRaWAN fPort number
// - bytes is an array of bytes, e.g. [225, 230, 255, 0]
// - variables contains the device variables e.g. {"calibration": "3.5"}
// (both the key / value are of type string)
// The function must return an object, e.g. {"temperature": 22.5}
function Decode(fPort, bytes, variables) {
    var json0bject = new Object();
    // FLAGS used to find what is in payload
    var TEMP_FLAG = 64;      // 01000000
    var HUM_FLAG = 32;       // 00100000
    var PRES_FLAG = 16;      // 00010000
    var GAS_FLAG = 8;        // 00001000
    var FREQ_FLAG = 4;       // 00000100
    var AMP_FLAG = 2;        // 00000010
    var TEST_FLAG = 1;        // 00000001
    var offset = 0;

    if (bytes[0] & TEMP_FLAG) {
        json0bject.temperature = (bytes[offset+4] << 24) |
        (bytes[offset+3] << 16) | (bytes[offset+2] << 8) | bytes[offset+1];
        json0bject.temperature /= 100;
        json0bject.temperature.toFixed(2)
        offset += 4;
    }
    if (bytes[0] & HUM_FLAG) {
        json0bject.humidity = (bytes[offset+4] << 24) | (bytes[offset+3]
```

```

<< 16) | (bytes[offset+2] << 8) | bytes[offset+1];
        jsonObject.humidity /= 100;
        jsonObject.humidity.toFixed(2)
        offset += 4;
    }
    if (bytes[0] & PRES_FLAG) {
        jsonObject.pressure = (bytes[offset+4] << 24) | (bytes[offset+3]
<< 16) | (bytes[offset+2] << 8) | bytes[offset+1];
        jsonObject.pressure /= 100;
        jsonObject.pressure.toFixed(2)
        offset += 4;
    }
    if (bytes[0] & GAS_FLAG) {
        jsonObject.gas = (bytes[offset+4] << 24) | (bytes[offset+3] <<
16) | (bytes[offset+2] << 8) | bytes[offset+1];
        jsonObject.gas /= 1000;
        jsonObject.gas.toFixed(2)
        offset += 4;
    }
    if (bytes[0] & FREQ_FLAG) {
        jsonObject.frequency = (bytes[offset+4] << 24) | (bytes[offset+3]
<< 16) | (bytes[offset+2] << 8) | bytes[offset+1];
        offset += 4;
    }
    if (bytes[0] & AMP_FLAG) {
        jsonObject.amplitude = (bytes[offset+4] << 24) | (bytes[offset+3]
<< 16) | (bytes[offset+2] << 8) | bytes[offset+1];
        offset += 4;
    }
    if (bytes[0] & TEST_FLAG) {
        jsonObject.counter = (bytes[offset+2] << 8) | bytes[offset+1];
    }

    return jsonObject;
}

```

Фиг. 5.9.

Декодираща функция.

Декодиращата функция се “качва” върху приложния сървър, има специално отделено място, където да се качи. След което приложния сървър се грижи за интерпретацията и извикването на функцията. Функцията декодира данните от бинарен вид във вид на JSON формат,

след което данните се препращат по HTTP към dbService, който се грижи за записването им в InfluxDB.

Данните се записват в InfluxDB в следния формат:

- Измерването се създава при първия запис в субд-то и името му в конкретния случай на разглежданата система е "SoundMeasurementSystem";
- като таг ключ се записва devEUI на устройството;
- като ключове на полете се записват съответно - "amplitude", "counter", "frequency", "gas", "humidity", "pressure", "temperature";
- Времето на всяка точка е текущото време на системата в момента на записа;

За имплементиране на dbService е използвана библиотеката http.server към python езика. Също е използван API интерфейса предоставен от ChirpStack.[66][67]

В приложния сървър първо трябва да се създаде организация, конфигурацията в конкретната дипломна работа е изложена на фиг. 5.10.

[Organizations](#) / diploma

DASHBOARD **CONFIGURATION**

Organization name *

diploma

The name may only contain words, numbers and dashes.

Display name *

Diploma project

Gateways

Organization can have gateways

When checked, it means that organization administrators are able to add their own gateways to the network. Note that the usage of the gateways is not limited to this organization.

Max. number of gateways *

0

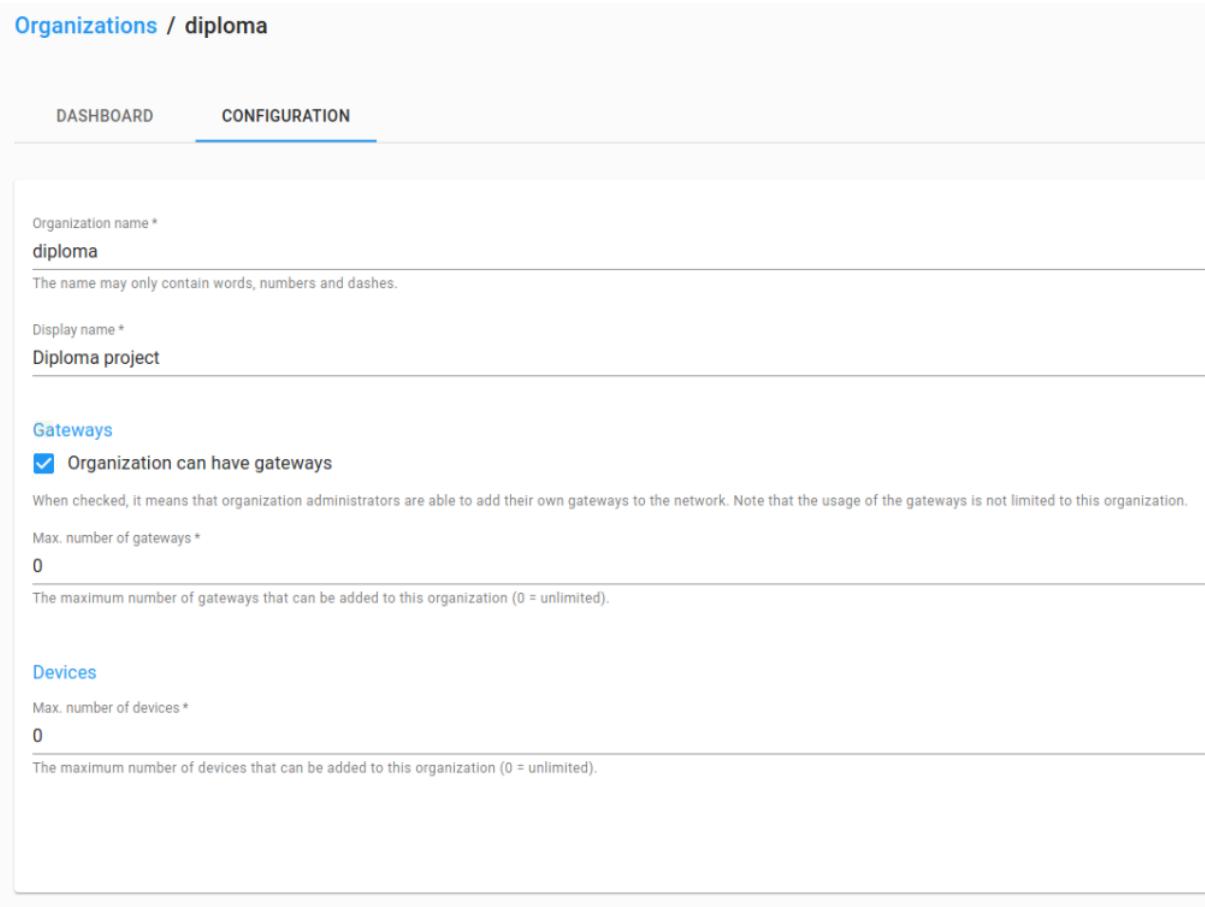
The maximum number of gateways that can be added to this organization (0 = unlimited).

Devices

Max. number of devices *

0

The maximum number of devices that can be added to this organization (0 = unlimited).



Фиг. 5.10.

Създаване на организация.

След това трябва да бъде създаден Service-Profile, настройките използвани в текущата система са показани на фиг. 5.11.

Service-profile name *

ServiceProfile

A name to identify the service-profile.

Add gateway meta-data

GW metadata (RSSI, SNR, GW geoloc., etc.) are added to the packet sent to the application-server.

Enable network geolocation

When enabled, the network-server will try to resolve the location of the devices under this service-profile. Please note that y support.

Device-status request frequency

0

Frequency to initiate an End-Device status request (request/day). Set to 0 to disable.

Minimum allowed data-rate *

0

Minimum allowed data rate. Used for ADR.

Maximum allowed data-rate *

5

Maximum allowed data rate. Used for ADR.

Фиг. 5.11.
Поле за създаване на Service-profile.

Трябва също да се създаде и Network-server, настройките на текущата имплементация са показани на фиг. 5.12.

GENERAL GATEWAY DISCOVERY TLS CERTIFICATES

Network-server name *

LoRaServer

A name to identify the network-server.

Network-server server *

localhost:8000

The 'hostname:port' of the network-server, e.g. 'localhost:8000'.

Фиг. 5.12.
Създаване на сървър.

След като бъдат изпълнени горепосочените стъпки може да се създаде “шлюз” и “апликация” (gateway, application). Те представляват абстрактно ниво съответно на апликацията и шлюза върху приложния сървър. На фиг. 5.13. е показано полето за настройка на “шлюза”.

| GENERAL | TAGS | METADATA |
|---|------|----------|
| Gateway name * The name may only contain words, numbers and dashes. | | |
| Gateway description * | | |
| Gateway ID * | | |
| Network-server * Select network-server Select the network-server to which the gateway will connect. When no network-servers are available in the dropdown, make sure a service-profile is created and assigned to the gateway. | | |
| Gateway-profile Select gateway-profile Optional. When assigning a gateway-profile to the gateway, ChirpStack Network Server will attempt to update the gateway according to the gateway-profile settings. | | |
| <input type="checkbox"/> Gateway discovery enabled When enabled (and ChirpStack Network Server is configured with the gateway discover feature enabled), the gateway will send out periodical probe messages. | | |
| Gateway altitude (meters) * 0 When the gateway has an on-board GPS, this value will be set automatically when the network has received statistics from the gateway. | | |

Фиг. 5.13.
Прозорец за създаване на шлюз.

В конкретната имплементация концентратора се създава автоматично от софтуера предоставен от RAKWireless. Gateway-profile не е създаден в разглежданата система. Софтуерът на RAKWireless позволява да се правят промени по настройките на концентратора, но такива не са правени в конкретната дипломна работа.

Преди да бъде създадена апликация, трябва да се създаде и Device-profile. На фиг. 5.14., фиг. 5.15. и фиг. 5.16. са показани настройките на Device-profile.

GENERAL

Device-profile name *

DEVPROF-EU868-OTAA

A name to identify the device-profile.

LoRaWAN MAC version *

1.0.2

The LoRaWAN MAC version suitable for this device.

LoRaWAN Regional Parameters

A

Revision of the Regional Parameters.

ADR algorithm *

Default ADR algorithm

The ADR algorithm that will be used by the device.

Max EIRP *

16

Maximum EIRP supported by the device.

Uplink Interval (seconds) *

30

The uplink interval in seconds.

Фиг. 5.14.

Задаване на настройки на Device-Profile.

GENERAL **JOIN (OTAA / ABP)**

Device supports OTAA

Фиг. 5.15.

Метод на активация е зададен OTAA.

Payload codec

Custom JavaScript codec functions

By defining a payload codec, ChirpStack Application Server can encode and decode the binary device payload for you.

```

1 // Decode decodes an array of bytes into an object.
2 // - fPort contains the LoRaWAN fPort number
3 // - bytes is an array of bytes, e.g. [225, 230, 255, 0]
4 // - variables contains the device variables e.g. {"calibration": "3.5"} (both the key / value are of type string)
5 // The function must return an object, e.g. {"temperature": 22.5}
6 function Decode(fPort, bytes, variables) {
7     var jsonObject = new Object();
8
9     // FLAGS used to find what is in payload
10    var TEMP_FLAG = 64;      // 01000000
11    var HUM_FLAG = 32;       // 00100000
12    var PRES_FLAG = 16;      // 00010000
13    var GAS_FLAG = 8;        // 00001000
14    var FREQ_FLAG = 4;       // 00000100
15    var AMP_FLAG = 2;        // 00000010
16    var TEST_FLAG = 1;       // 00000001
17
18    var offset = 0;
19

```

The function must have the signature `function Decode(fPort, bytes)` and must return an object. ChirpStack Application Server will convert this object

```

1 // Encode encodes the given object into an array of bytes.
2 // - fPort contains the LoRaWAN fPort number
3 // - obj is an object, e.g. {"temperature": 22.5}
4 // - variables contains the device variables e.g. {"calibration": "3.5"} (both the key / value are of type string)
5 // The function must return an array of bytes, e.g. [225, 230, 255, 0]
6 function Encode(fPort, obj, variables) {
7     return [];
8 }

```

Фиг. 5.16.

Прозорецът, в който поставя декодиращата функция.

След като всички стъпки са изпълнени може да се създаде апликация, в която вече да се създават различни устройства.

Съответно при създаване на апликация трябва да се въведе име, описание на апликацията и Service-profile. (виж фиг. 5.17)

Applications / Create

Application name *

The name may only contain words, numbers and dashes.

Application description *

Service-profile *

Select service-profile

The service-profile to which this application will be attached. Note that you can't change this value after the application has been created.

CREATE APPLICATION

Фиг. 5.17.

Създаване на апликация.

След като е създадена апликация може да се създаде устройство към тази апликация. За тази цел устройството трябва да има име, описание, devEUI, който може да е генериран от сървъра или да е вграден в устройството и Device-profile. На фиг. 5.18. е показано полето за създаване на устройство.

GENERAL VARIABLES TAGS

Device name *

The name may only contain words, numbers and dashes.

Device description *

Device EUI *

MSB C

Device-profile *

Select device-profile

Disable frame-counter validation

Note that disabling the frame-counter validation will compromise security as it enables people to perform replay-attacks.

Device is disabled

ChirpStack Network Server will ignore received uplink frames and join-requests from disabled devices.

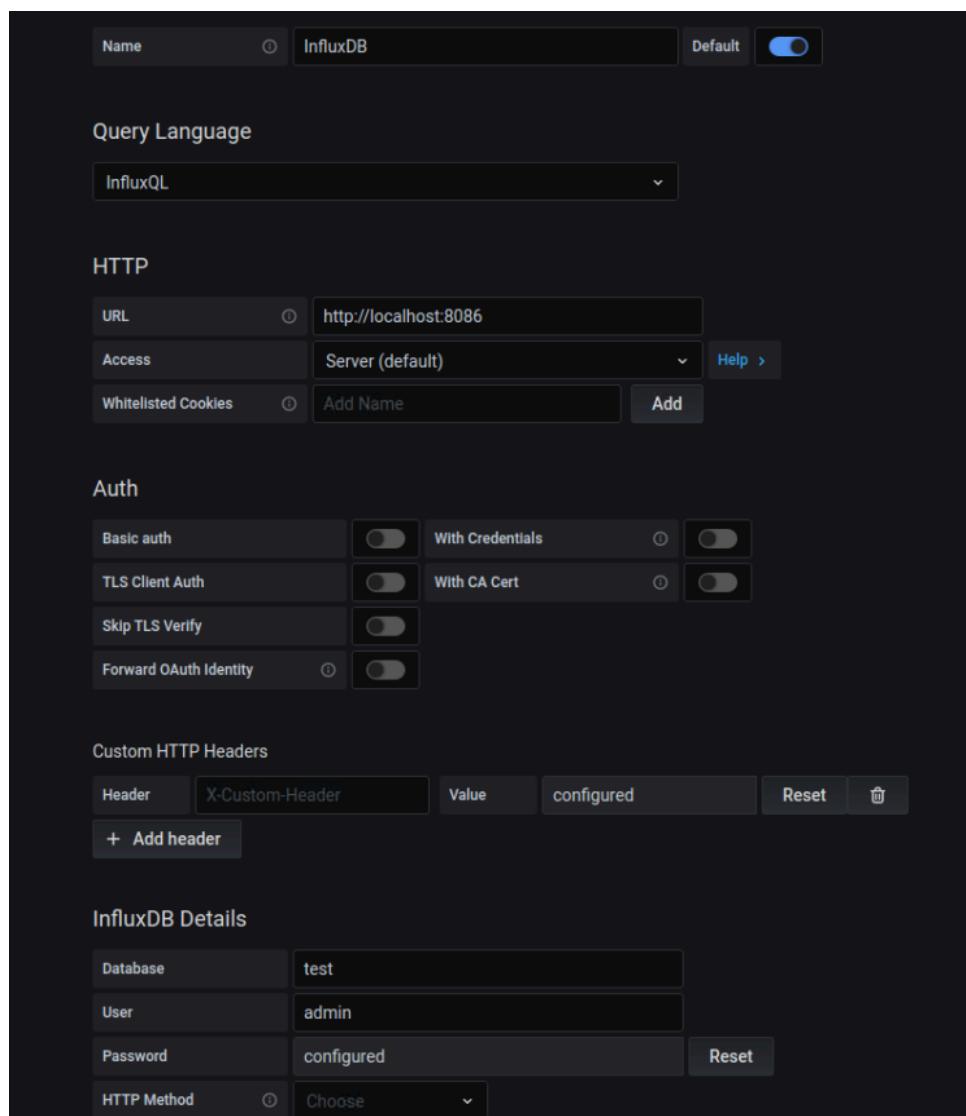
CREATE DEVICE

Фиг. 5.18.

Прозорец за създаване на устройство.

В конкретната реализация на дипломната работа е създадена апликация, която се казва “SoundMeasurementSystem” и две устройства - “Seeeduino” и “MKRWAN_1300”.

Относно Grafana, първо трябва да бъде създадена връзка между Grafana и базата с данни (в случая InfluxDB). В разглежданата система е избран InfluxQL за език за заявки към базата и базата и Grafana са на една машина, съответно Grafana се свърза към localhost:8086 (адреса на InfluxDB по подразбиране). Избрана е база данни ‘test’ и потребител admin. На фиг. 5.19. са показани настройките.



Фиг. 5.19.

Прозорец за връзка към база с данни.

След това се създават, поканват по имейл потребители, с права само да преглеждат таблица. (виж фиг. 5.20.)

| Login | Email | Name | Seen | Role | |
|-------------|------------------------------------|--------|------|--------|----------------|
| admin | admin@localhost | | 4m | Admin | x |
| VikoTodorov | viktor.a.todorov.2016@elsys-bg.... | Viktor | 13d | Viewer | x |

Фиг. 5.20.

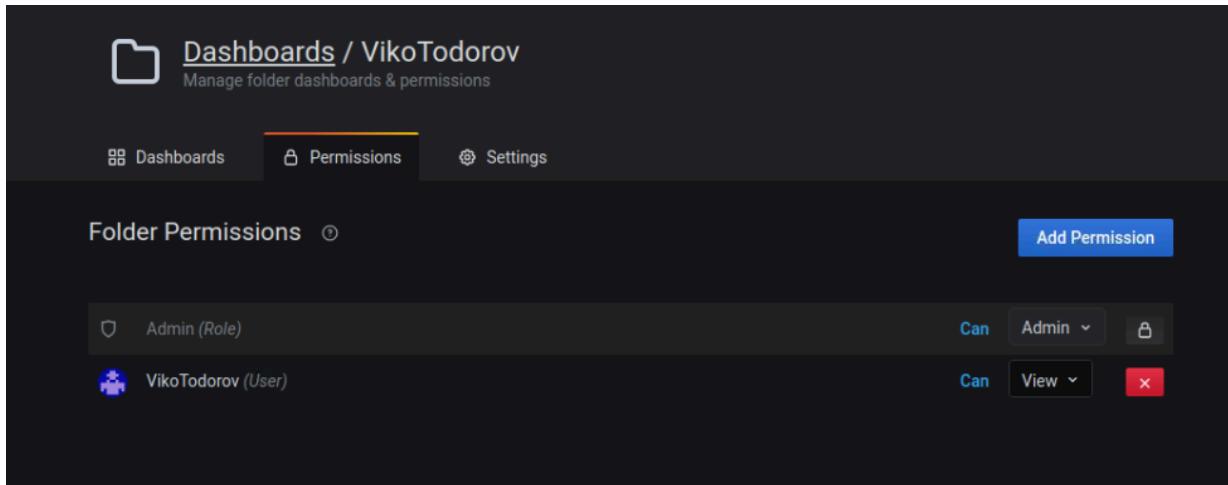
Прозорец за поканване на нови хора и задаване на права.

След това за всеки потребител се създава папка с неговите устройства, настройките са показани на фиг. 5.21. и фиг. 5.22.

The screenshot shows a dark-themed user interface for managing dashboards. At the top, there's a header with a folder icon and the text 'Dashboards / VikoTodorov'. Below the header, there are three tabs: 'Dashboards' (selected), 'Permissions', and 'Settings'. Underneath the tabs, there's a search bar with the placeholder 'Search dashboards by name' and two blue buttons: 'New Dashboard' and 'Import'. Further down, there are two filter dropdowns: 'Sort (Default A-Z)' and 'Filter by starred'. A 'Filter by tag' dropdown is also present. The main content area displays a single dashboard entry: 'Seeeduino' under 'VikoTodorov'. The entire interface has a clean, modern design with a focus on functionality.

Фиг. 5.21.

Папка с таблица.



Фиг. 5.22.

Прозорец за даване на права на потребителите за дадена папка.

На всяко табло от системата се създават две графики съответно за амплитудата и за честотата. Важно е да се отбележи, ако още не е станало ясно, че под амплитуда се има предвид измерването на АЦП-то, а не амплитуда спрямо човешкия слух. За температура, налягане, влажност и съпротивление на газ се създават четири статистики - текуща, средна, максимална и минимална стойност в измервания период от време. На фиг. 5.23., фиг. 5.24., фиг. 5.25., фиг. 5.26., фиг. 5.27. и фиг. 5.28. са изобразени настройките, които са направени за да се извлекат и изчислят четирите статистики на температурата. Аналогичен е процеса и при извличането на налягане, съпротивление на газ и налягане. Само полето, по което се изличат стойностите трябва да се промени.

A

| | | | | | | | |
|-----------|---------------------|------------------------|-------|--------|---|------------------|---|
| FROM | default | SoundMeasurementSystem | WHERE | devEUI | = | 47cb55800017003a | + |
| SELECT | field (temperature) | last () | + | | | | |
| GROUP BY | time (\$__interval) | fill (null) | + | | | | |
| FORMAT AS | Time series | | | | | | |
| ALIAS BY | Temperature | | | | | | |

Фиг. 5.23.

Извличане на последната измерена стойност.

A

| | | | | | | | |
|-----------|---------------------|------------------------|-------|--------|---|------------------|---|
| FROM | default | SoundMeasurementSystem | WHERE | devEUI | = | 47cb55800017003a | + |
| SELECT | field (temperature) | mean () | + | | | | |
| GROUP BY | time (\$__interval) | fill (null) | + | | | | |
| FORMAT AS | Time series | | | | | | |
| ALIAS BY | Temperature | | | | | | |

Фиг. 5.24.

Извличане на средната сума от измерени стойности.

A

| | | | | | | | |
|-----------|---------------------|------------------------|-------|--------|---|------------------|--|
| FROM | default | SoundMeasurementSystem | WHERE | devEUI | = | 47cb55800017003a | |
| SELECT | field (temperature) | min () | + | | | | |
| GROUP BY | time (\$__Interval) | fill (null) | + | | | | |
| FORMAT AS | Time series | ▼ | | | | | |
| ALIAS BY | Temperature | | | | | | |

Фиг. 5.25.

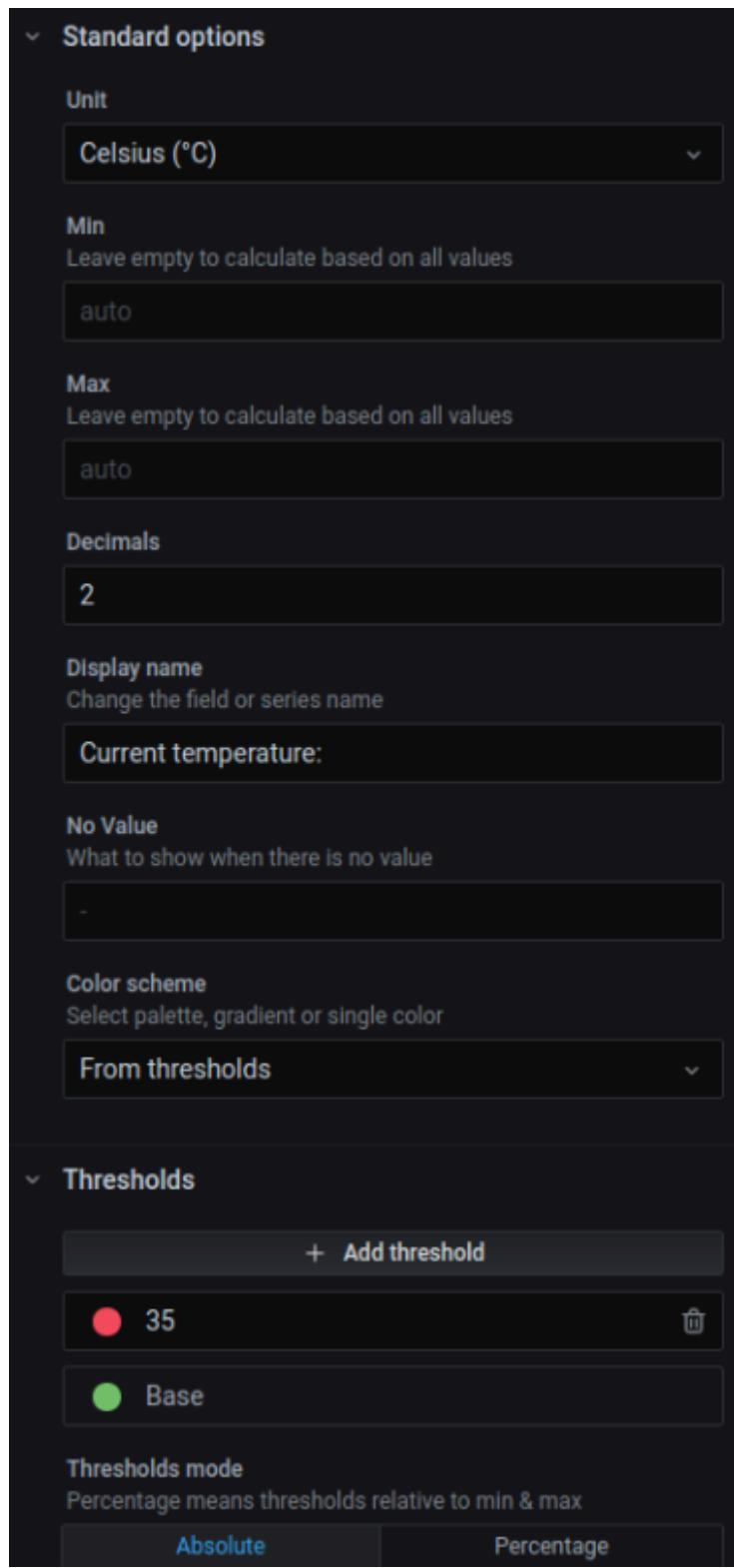
Извличане на най-малката измерена стойност.

A

| | | | | | | | |
|-----------|---------------------|------------------------|-------|--------|---|------------------|--|
| FROM | default | SoundMeasurementSystem | WHERE | devEUI | = | 47cb55800017003a | |
| SELECT | field (temperature) | max () | + | | | | |
| GROUP BY | time (\$__Interval) | fill (null) | + | | | | |
| FORMAT AS | Time series | ▼ | | | | | |
| ALIAS BY | Temperature | | | | | | |

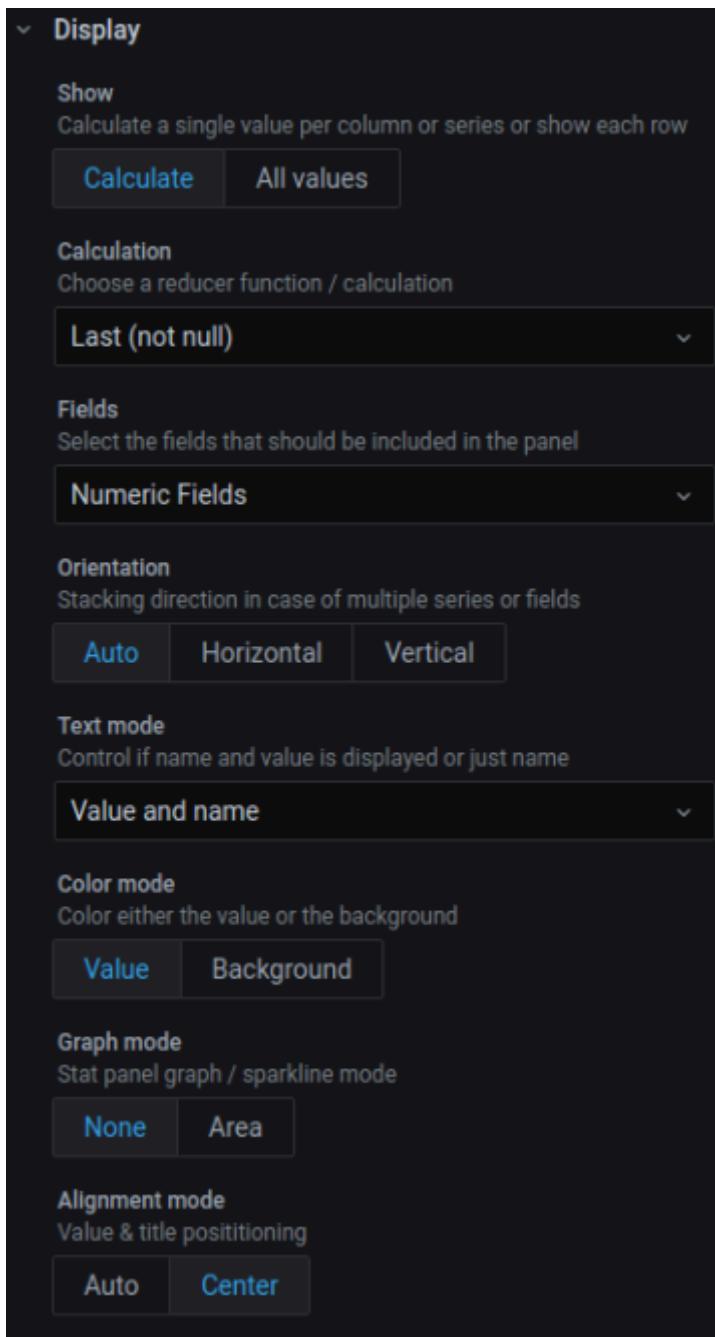
Фиг. 5.26

Извличане на най-голямата измерена стойност.



Фиг. 5.27.

Задаване на прагове, мерни единици и др.



Фиг. 5.28.

За всяка от статистики на "Calculation" да се зададе правилната опция:
last, max, min или mean.

За графиките настройките са показани на фигурите от фиг. 5.29 до
фиг. 5.33.

The screenshot shows a Grafana query editor interface. The query is named 'A' and consists of the following parameters:

- FROM:** default SoundMeasurementSystem
- WHERE:** devEUI = 47cb55800017003a
- SELECT:** field (frequency) mean ()
- GROUP BY:** time (\$__interval) fill (null)
- FORMAT AS:** Time series
- ALIAS BY:** Frequency

Фиг. 5.29.

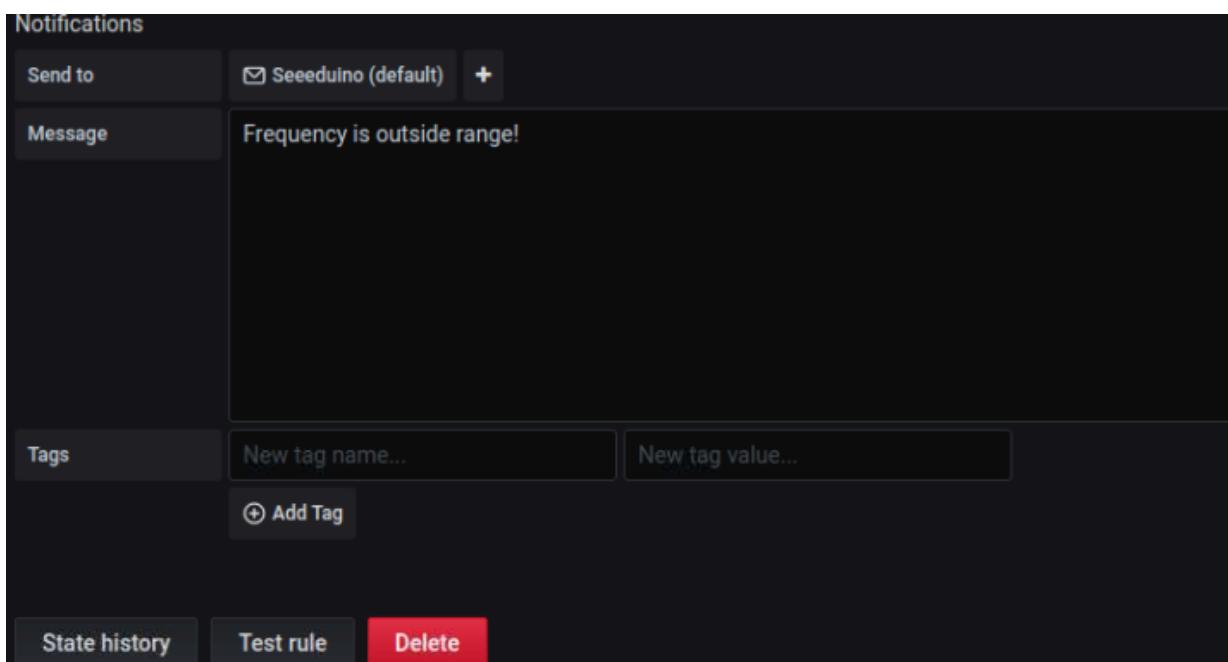
Извличане на средната честота за даден интервал.

The screenshot shows a Grafana rule configuration window titled 'Rule'. The rule is named 'Frequency_alert' and has the following settings:

- Evaluate every:** 1m
- For:** 3m
- Conditions:** avg () OF query (A, 5m, now) IS OUTSIDE RANGE 100 TO 900
- No Data & Error Handling:**
 - If no data or all values are null: SET STATE TO Keep Last State
 - If execution error or timeout: SET STATE TO Alerting

Фиг. 5.30.

Създаване на предупреждение при честоти извън интервала 100-900Hz.



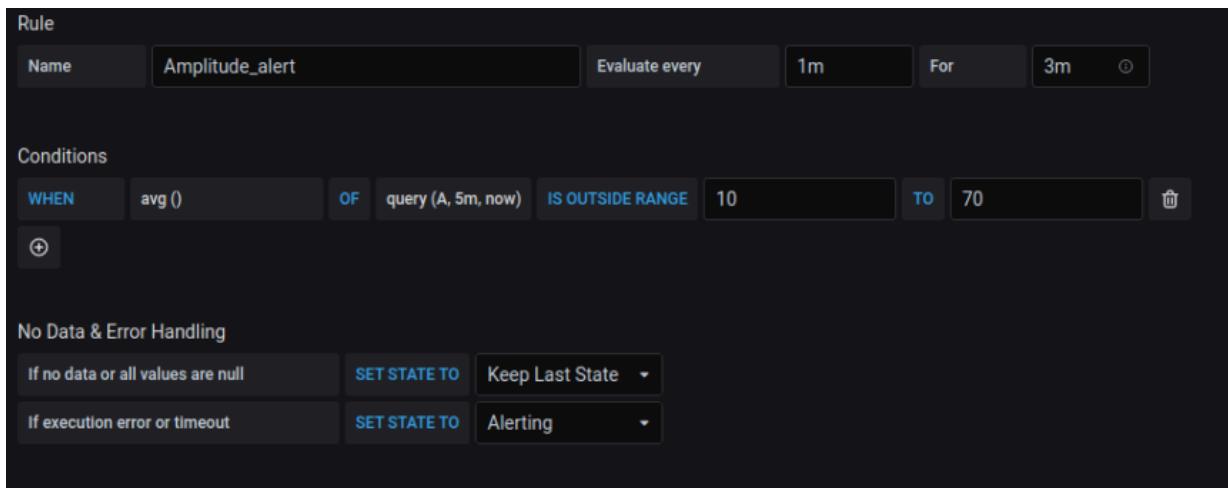
Фиг. 5.30.

Съобщението, което се изпраща при предупреждение за съмнителна честота.

The screenshot shows a query builder interface with a search bar labeled 'A'. The query is defined by several fields:
- FROM: default SoundMeasurementSystem
- WHERE: devEUI = 47cb55800017003a
- SELECT: field (amplitude) mean ()
- GROUP BY: time (\$__interval) fill (null)
- FORMAT AS: Time series
- ALIAS BY: Amplitude

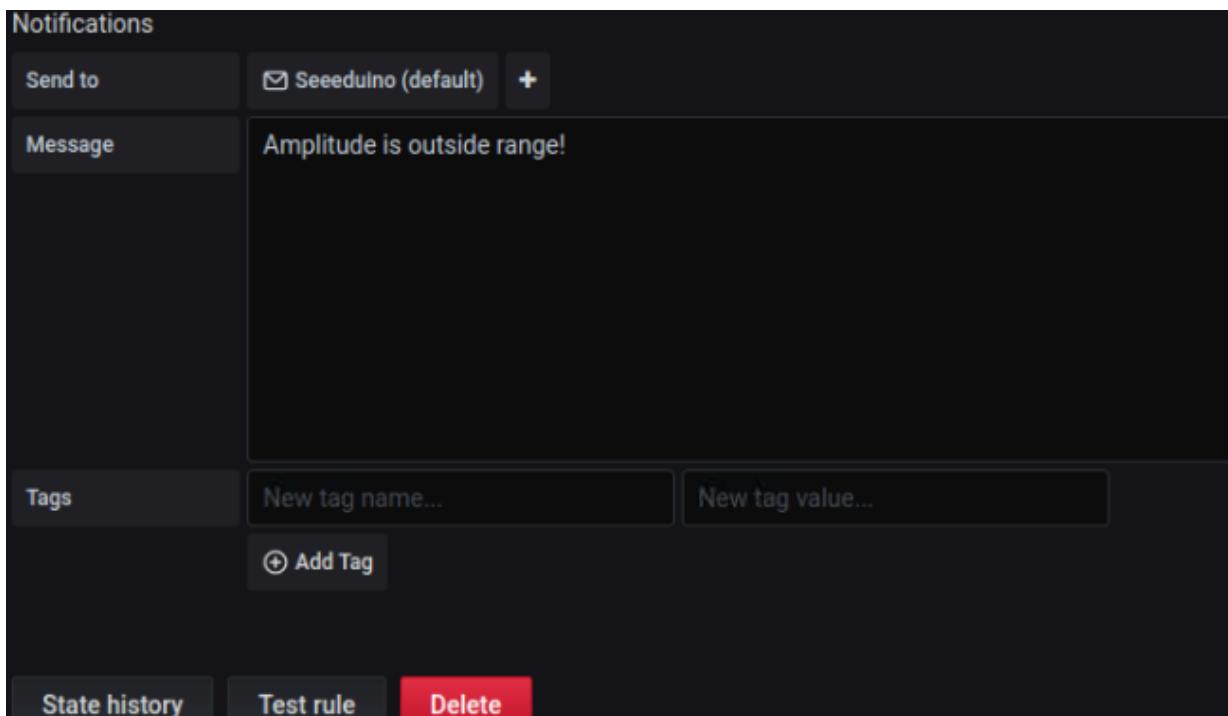
Фиг. 5.31.

Извличане на средната амплитуда за даден интервал.



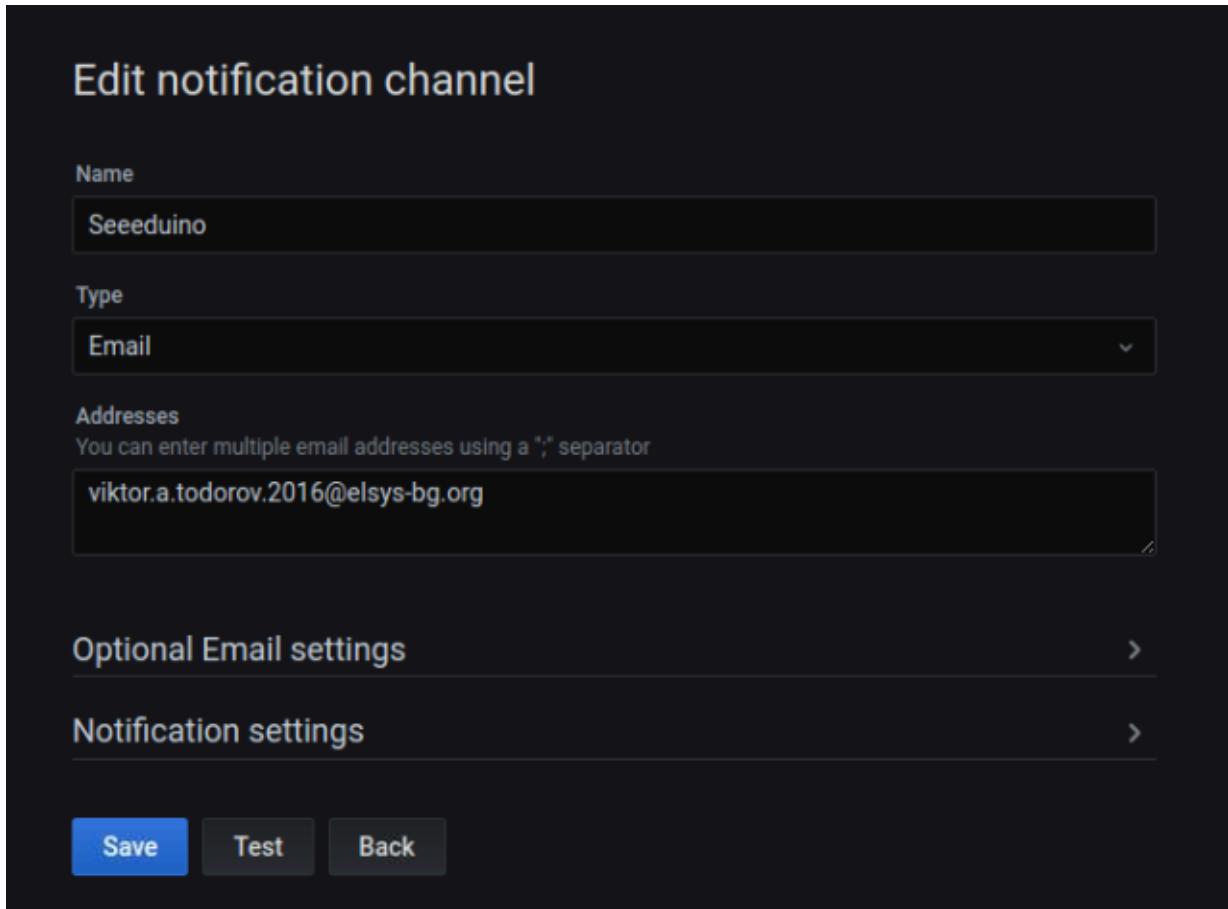
Фиг. 5.32.

Създаване на предупреждение при амплитуди извън интервала 10-70 единици.



Фиг. 5.33.

Съобщението, което се изпраща при предупреждение за съмнителна амплитуда.



Фиг. 5.34.

Всеки “Alert Rule” трябва да има “Notification channel”, на който се въвежда метода за известяване и потребителите, до които да достигне.

5.4. Управляващ софтуер на Блок крайното устройство

В тази точка ще бъдат разгледани по-подробни блокови схеми и функционалности на Блок крайно устройство.

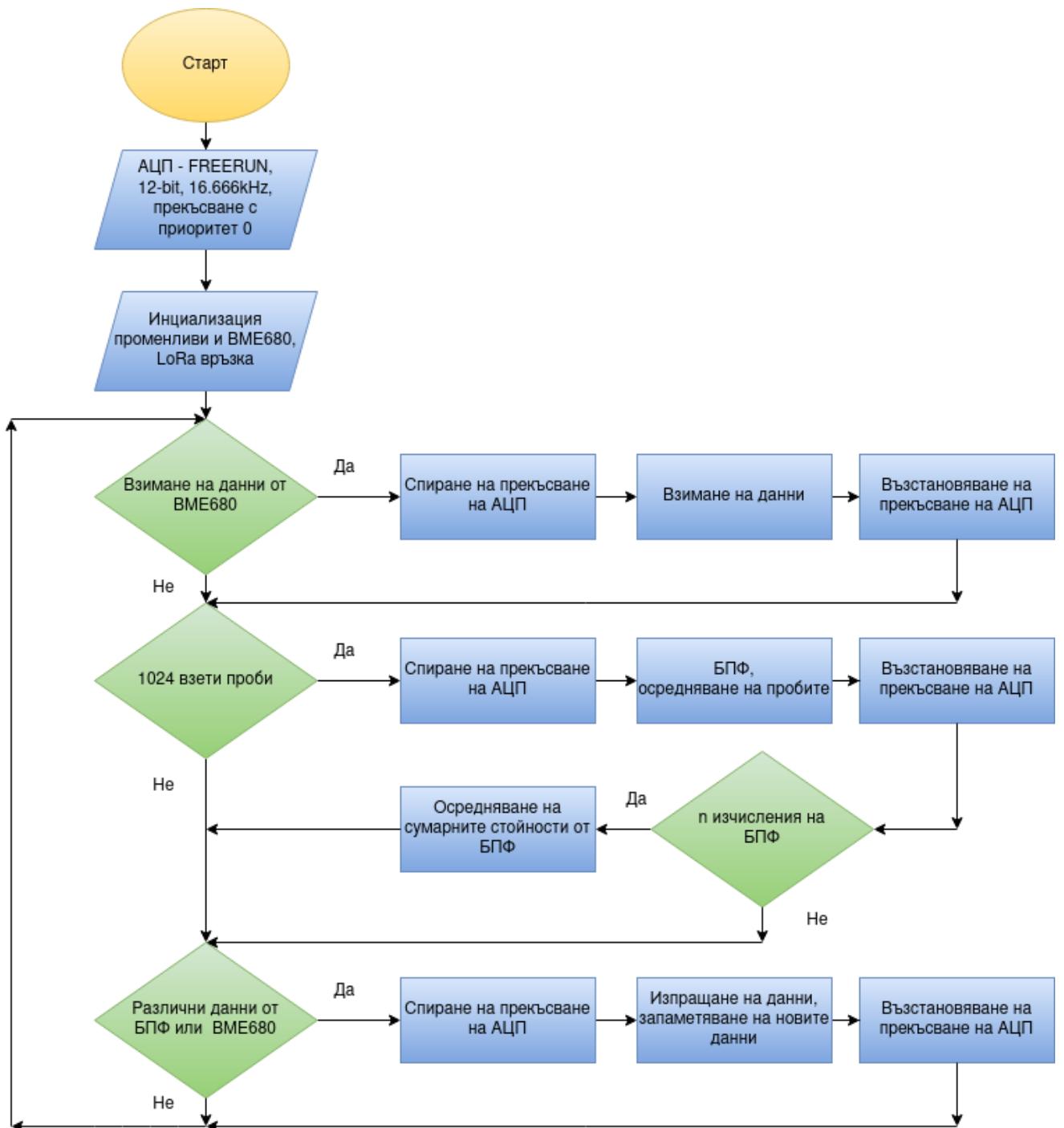
5.4.1. Блокови схеми и принципни схеми

Върху крайното устройство се инициализира LoRaWAN връзка със сървъра. Създава се инстанция на класът Adafruit_BME680 и се инициализира и настройва АЦП-то с честота 16.666 kHz, 12 битова резолюция. АЦП-то се настройва да има прекъсване с приоритет 0, най-големия потребителски предназначен приоритет на прекъсване.

Данните от BME680 не се събират постоянно, а на определено време, когато настъпи моментът за събиране на данни от BME680, прекъсването на АЦП-то се изключва, данните се събират и прекъсването се възобновява.

БПФ не се изчислява постоянно, а на определен брой взети преби, по време на изчислението прекъсването на АЦП-то се изключва. Дори да се вземат 1024 преби, времето което изследваме така е изключително малко, ако на всяка преба и трябват 60us, то времето, което се изследва така е само 61.44ms. Поради тази причина може да се сумират няколко изчисления на БПФ и на определен брой да се осредняват. Ако за 1024 преби се изследва време от 61.44ms и БПФ от тези преби се сумира n четни пъти: например 64 пъти е сумирано БПФ, то времето което се обхваща от анализа така е равно на 3.93216s или приблизително 4s. n се избира така, че да може осредняването да се извърши чрез побитово отместване, тоест n е степен на 2-ката (16, 32, 64 и тн.).

Когато има нови данни, данните се сравняват с предишните и ако има значителна разлика между тях, новите данни се записват на мястото на предишните и се изпращат към сървъра по LoRaWAN протокола. На фиг. 5.5. е показана принципна блокова схема на процеса.



Фиг. 5.35.

Блокова схема на фърмуера управляващ крайните устройства.

5.4.2. Сорс код на БКУ

В тази точка ще бъдат представени извадки от кода, който контролира Блок крайно устройство.

На фиг. 5.36. е показан извадка от кода на крайното устройство, който отговаря за пресмятането на БПФ.

```
if (sampleCounter == dataSize) {  
    uint16_t amp = 0;  
    NVIC_DisableIRQ(ADC_IRQn); // stop ADC_IRQHandler to fill values in array  
    removeDCOffset(aDCVal, dataSize);  
    for (int i=0; i < dataSize; i++){  
        amp += abs(aDCVal[i]);  
    }  
    soundMeasurement.amplitude += amp >> 9;  
    ZeroFFT((q15_t*)aDCVal, dataSize);  
    for (int i=0; i < dataHalfSize; i++){  
        if (aDCVal[i] > aDCVal[indexFFT]) {  
            indexFFT = i;  
        }  
    }  
    soundMeasurement.frequency += (uint32_t)FFT_BIN(indexFFT, sampleRate, dataSize);  
    countFFT++;  
  
    if (countFFT == fftCycles) {  
        soundMeasurement.amplitude >>= fftCyclesDiv;  
        soundMeasurement.frequency >>= fftCyclesDiv;  
        if (soundMeasurement.amplitude - prevSoundMeasurement.amplitude >= 10  
            || soundMeasurement.amplitude - prevSoundMeasurement.amplitude <= -10) {  
            prevSoundMeasurement.amplitude = soundMeasurement.amplitude;  
            ampFlag = true;  
        }  
        if (soundMeasurement.frequency - prevSoundMeasurement.frequency >= 45  
            || soundMeasurement.frequency - prevSoundMeasurement.frequency <= -45) {  
            prevSoundMeasurement.frequency = soundMeasurement.frequency;  
            freqFlag = true;  
        }  
        soundMeasurement.frequency = 0;  
        soundMeasurement.amplitude = 0;  
        countFFT = 0;  
    }  
  
    indexFFT = 0;  
    sampleCounter = 0;  
    NVIC_EnableIRQ(ADC_IRQn); // enable ADC_IRQHandler again  
}
```

Фиг. 5.36.

Извадка от фърмуера, отговаряща за БПФ преобразуванието.

На фиг. 5.37. е изложена извадката, която създава пакета с данни, който после бива препратен по LoRaWAN протокола.

```
if (tempFlag) {
    packetBuffer[0] |= (1 << TEMP_HEAD);
    memmove(packetBuffer+offset, &bmeMeasurement.temp,
    sizeof(bmeMeasurement.temp));
    tempFlag = false;
    offset += 4;
    newMessage = true;
}
if (humFlag) {
    packetBuffer[0] |= (1 << HUM_HEAD);
    memmove(packetBuffer+offset, &bmeMeasurement.humidity,
    sizeof(bmeMeasurement.humidity));
    humFlag = false;
    offset += 4;
    newMessage = true;
}
if (pressFlag) {
    packetBuffer[0] |= (1 << PRESS_HEAD);
    memmove(packetBuffer+offset, &bmeMeasurement.pressure,
    sizeof(bmeMeasurement.pressure));
    pressFlag = false;
    offset += 4;
    newMessage = true;
}
if (gasFlag) {
    packetBuffer[0] |= (1 << GAS_HEAD);
    memmove(packetBuffer+offset, &bmeMeasurement.gas,
    sizeof(bmeMeasurement.gas));
    gasFlag = false;
    offset += 4;
    newMessage = true;
}

if (freqFlag) {
    packetBuffer[0] |= (1 << FREQ_HEAD);
    memmove(packetBuffer+offset, &prevSoundMeasurement.frequency,
    sizeof(prevSoundMeasurement.frequency));
    freqFlag = false;
    offset += 4;
    newMessage = true;
}
if (ampFlag) {
    packetBuffer[0] |= (1 << AMP_HEAD);
```

```

        memmove(packetBuffer+offset, &prevSoundMeasurement.amplitude,
        sizeof(prevSoundMeasurement.amplitude));
        ampFlag = false;
        offset += 4;
        newMessage = true;
    }
    if (testFlag) {
        packetBuffer[0] |= (1 << TEST_HEAD);
        memmove(packetBuffer+offset, &testCounter,
        sizeof(testCounter));
        testFlag = false;
    }
}

```

Фиг. 5.37.

Извадка от сурс кода, отговаряща за структурирането на протокола.

На фиг. 5.38. е изложен част от кода, който настройва АЦП.

```

// --> Need of 12.5 kHz freq = 80us
// --> ADC time = prop. delay(p. 785) + Adj.ADC sample(p. 798) - half
cycle(the sample, p. 787)
// --> Using 8MHz system clock with division factor of 1
// --> DIV64 -> ADC generic cloak 8MHz/64 = 125 000 -> time = 1/125 000
= 8us (one cycle)
// --> Prop. delay = (6 + 0(GAIN_1x))/125 000 = 48us
// --> 80us = 32us + Adj. ADC sample - 4us -> Adj. ADC sample = 36us
// --> Adj. ADC sample = (samplen+1)*(cycle/2) = (samplen+1)*4us ->
samplen = 8
// This function sets up the ADC, including setting resolution and ADC
sample rate

void aDCSetup() {
    // set vref for ADC to VCC
    REG_ADC_REFCTRL = ADC_REFCTRL_REFSEL_INTVCC1;

    // average control 1 sample
    // samplen = 8
    REG_ADC_AVGCTRL |= ADC_AVGCTRL_SAMPLENUM_1;
    REG_ADC_SAMPCTRL = ADC_SAMPCTRL_SAMPLEN(8);

    // Input control and input scan, gain 0, positive to A0, negative to
gnd
    REG_ADC_INPUTCTRL |= ADC_INPUTCTRL_GAIN_1X |
ADC_INPUTCTRL_MUXNEG_GND | ADC_INPUTCTRL_MUXPOS_PIN0;
    while (REG_ADC_STATUS & ADC_STATUS_SYNCBUSY);
}

```

```

// set the divide factor, 8 bit resolution and freerun mode
ADC->CTRLB.reg |= ADC_CTRLB_RESEL_12BIT | ADC_CTRLB_PRESCALER_DIV64
| ADC_CTRLB_FREERUN;
while (REG_ADC_STATUS & ADC_STATUS_SYNCBUSY);

// Disable window monitor mode
ADC->WINCTRL.reg = ADC_WINCTRL_WINMODE_DISABLE;
while(ADC->STATUS.bit.SYNCBUSY);

// start ADC when event occurs
ADC->EVCTRL.reg |= ADC_EVCTRL_STARTEI;
while (ADC->STATUS.bit.SYNCBUSY);

// enable and set ADC to run in standby
ADC->CTRLA.reg |= ADC_CTRLA_ENABLE;
while (ADC->STATUS.bit.SYNCBUSY);
}

// This function sets up an ADC interrupt that is triggered
// when an ADC value is out of range of the window
// input argument is priority of interrupt (0 is highest priority,
// except RESET, -2 and -1)
void setUpInterrupt(byte priority) {

    // enable ADC ready interrupt
    ADC->INTENSET.reg |= ADC_INTENSET_RESRDY;
    while (ADC->STATUS.bit.SYNCBUSY);

    // enable ADC interrupts
    // set priority of the interrupt
    NVIC_EnableIRQ(ADC_IRQn);
    NVIC_SetPriority(ADC_IRQn, priority);
}

```

Фиг. 5.38.

Линк към хранилището, където се намира целия сорс код, както за Блок Сървър, така и за Блок крайно устройство може да се намери в литературата.[68]

Шеста глава - Практически резултати

6.1. Проблеми по време на разработката и решения

В тази глава ще бъдат разгледани основните срещнати проблеми по време на разработката на дипломната работа и техните решения, ако са били намерени по време на разработката.

6.1.1. Декодирането на пакета с данни върху приложния сървър

- Проблем:

Първоначално данните се предаваха във формат

“t21.33h53.90p934.28g475.94f8000a10t10”, където ‘t’ е флаг за температура, ‘h’ за влажност, ‘p’ за налягане, ‘g’ за съпротивление на сензора, ‘f’ за честота, ‘a’ за амплитуда, ‘t’ беше флаг за секунди, който в по нататъшното развитие на дипломната работа се превърна във флаг за тестови брояч. Проблемът се изразяваше в това, че беше реализиран алгоритъм използващ “lookahead” регулярен израз за декодирането на данните от низов формат във масив от низове “t21.33”, “h53.90”... По време на тестовете javascript функцията работеше перфектно във браузъра, но когато беше качена върху приложния сървър не искаше да работи. Първоначалния вид на функцията е показан на фиг. 6.1.

```
// Decode decodes an array of bytes into an object.  
//   - fPort contains the LoRaWAN fPort number  
//   - bytes is an array of bytes, e.g. [225, 230, 255, 0]  
//   - variables contains the device variables e.g.  
{"calibration": "3.5"} (both the key / value are of type  
string)  
// The function must return an object, e.g. {"temperature":  
22.5}  
function Decode(fPort, bytes, variables) {  
    var jsonObject = new Object();
```

```

var temp;
var humidity;
var pressure;
var gas;
var frequency;
var amplitude;
var seconds;
var message = atob(bytes);
message = message.split(/(=?[thpgfas])/g);

for (i of message) {
    switch(i[0]) {
        case 't':
            var str = i.slice(1);
            temp = parseFloat(str);
            break;
        case 'h':
            var str = i.slice(1);
            humidity = parseFloat(str);
            break;
        case 'p':
            var str = i.slice(1);
            pressure = parseFloat(str);
            break;
        case 'g':
            var str = i.slice(1);
            gas = parseFloat(str);
            break;
        case 'f':
            var str = i.slice(1);
            frequency = parseFloat(str);
            break;
        case 'a':
            var str = i.slice(1);
            amplitude = parseFloat(str);
            break;
        case 's':
            var str = i.slice(1);
            seconds = parseFloat(str);
            break;
    }
}

if (!isNaN(temp)) {
    jsonObject.temp = temp;
}

```

```

if (!isNaN(humidity)) {
    jsonObject.humidity = humidity;
}
if (!isNaN(pressure)) {
    jsonObject.pressure = pressure;
}
if (!isNaN(gas)) {
    jsonObject.gas = gas;
}
if (!isNaN(frequency)) {
    jsonObject.frequency = frequency;
}
if (!isNaN(amplitude)) {
    jsonObject.amplitude = amplitude;
}
if (!isNaN(seconds)) {
    jsonObject.seconds = seconds;
}
return jsonObject;
}

```

Фиг. 6.1.

Първоначалния вид на декодираща на функция.

- Решение

След допитване във форума на ChirpStack се оказа, че проблема се изразява в интерпретатора, който използва ChirpStack за декодиращата функция. От този форум дойде и идеята за сегашния формат на пакета с данни.[69]

6.1.2. Повреждане данните, взети от Аналогово-цифровия преобразувател, при инициализацията на класа Adafruit_BME680

- Проблем:

Фурие анализа като самостоятелна програма работеше и даваше сравнително точни резултати. При опит за връзването на логиката на BME680 модула и Фурие анализа, резултатите от Фурие анализа ставаха изключително неточни.

- Решение

Първоначално беше пренаписана библиотеката на Adafruit за BME680 сензора, където има функция `delay()` се замени от цикъл, който цикли докато не мине определен интервал от време. След като този подход не помогна, библиотеката беше преинсталала и върната в първоначалния си вид. Накрая проблема беше решен като инициализацията на класа `Adafruit_BME680` се направи да е след настройването на регистрите на Аналогово-цифровия преобразувател. В рамките на дипломната работа не стана ясно от какво е причинен проблема, предположенията са, че настройва регистър, който после се достъпва в настройките при АЦП-то, който не е занулен предварително и има вдигнат флаг за `write-protection` или промяна по регистрите на родовия брояч (`generic clock`), който използва АЦП-то.

6.1.3. Липса на обратна връзка от страна на шлюза към крайните устройства

- Проблем:

Според LoRaWAN стандарта всеки концентратор трябва да поддържа 8 канала, за да може да поддържа множество устройства.

Концентраторът, чрез който беше направен първоначален опит за реализация на системата, е едноканален (`Dragino LoRa/GPS HAT`).

Проблемът с едноканалния концентратор е, че ако му се подаде информация на честота, която не следи, тя просто няма да бъде засечена. Друг проблем е, че мрежовите сървъри предлагани на пазара постепенно започват да спазват стандарта все по-строго, което значи, че в един момент ще спрат да поддържат едноканални концентратори.

Разбира се, дори и едноканален, софтуерно може да се реализира концентратора да има двупосочна комуникация. Беше изтеглен "уж" двуканален "packet forwarder", който е базиран на едноканален. За него авторът и някои други хора твърдят, че може да се направи да

поддържа двупосочна връзка, съответно той беше изтеглен и тестван. Двупосочна комуникация не беше реализирана, на няколко пъти бяха променяни настройките по конфигурационния файл и дори веднъж по сурс файловете, но нямаше никакъв ефект. За инсталация и конфигурация беше следван справочника предоставлен от Dragino.[70] На фиг. 6.2. е показана конфигурацията на концентратора.

```
{^M
  "SX127x_conf": ^M
  {^M
    "freq": 868100000, ^M
    "freq_2": 868100000, ^M
    "spread_factor": 7, ^M
    "pin_nss": 6, ^M
    "pin_dio0": 7, ^M
    "pin_nss_2": 6, ^M
    "pin_dio0_2": 7, ^M
    "pin_RST": 3, ^M
    "pin_led1": 4, ^M
    "pin_NetworkLED": 22, ^M
    "pin_InternetLED": 23, ^M
    "pin_ActivityLED_0": 21, ^M
    "pin_ActivityLED_1": 29 ^M
  }, ^M
  "gateway_conf": ^M
  {^M
    "ref_latitude": 0.0, ^M
    "ref_longitude": 0.0, ^M
    "ref_altitude": 10, ^M

    "name": "your name", ^M
    "email": "a@b.c", ^M
    "desc": "Dual channel pkt forwarder", ^M

    "interface": "eth0", ^M

    "servers": ^M
    [^M
      {^M
        "address": "localhost", ^M
        "port": 1700, ^M
        "enabled": true ^M
      }, ^M
      {^M
        "address": "localhost", ^M
        "port": 1700, ^M
        "enabled": false ^M
      } ^M
    ] ^M
  } ^M
} ^M
```

Фиг. 6.2.

Промените направени върху файла са: беше променен порта да е 1700, който да отговаря на този на gateway-bridge софтуера на ChirpStack и адреса да отговаря на локалния за машината.

- Решение:

Системата беше препроектирана като едноканалният Dragino LoRa/GPS НАТ беше заменен с многоканалния RAK831.

6.1.4. Свързването на MKR WAN 1300 модула към LoRaWAN мрежата

- Проблем:

Проблемът се изразяваше в това, че MKR WAN 1300 изпращаше съобщения, когато се опитва да се свърже по OTAA метода, но когато беше направен опит да се направи ABP активация, не се получаваше нищо от страна на концентратора. Дори log файла на "packet forwarder" не показваше да има никакви изпратени съобщения.

- Решение:

След като концентратора беше заменен с RAK831 и активацията беше реализирана по OTAA метода, проблемът беше решен.

6.2. Тестове и изводи

В тази подточка ще бъдат разгледани различните проведени експерименти в рамките на дипломната работа и ще се направят съответните изводи за състоянието и поведението на системата. Ще се разгледат библиотеките, с които системата е направена и другите библиотеки, които бяха тествани в рамките на дипломната работа.

6.2.1. AudioFrequencyMeter.h

AudioFrequencyMeter.h е първата библиотека, с която бяха направени тестове, но библиотеката поддържа много малко високи честоти, затова

беше изоставена. Тя е официалната Ардуино библиотека. Линк към взетите резултати и сурс кода може да намерите в литературата. [71]

6.2.2. Adafruit_ZeroFFT.h

Adafruit_ZeroFFT.h е текущата библиотека, която се използва в дипломната работа. Тя е базирана на CMSIS-DSP, разработена е от Adafruit. Линк към взетите резултати и сурс кода може да намерите в литературата. [72]

6.2.3. ArduinoSound.h

ArduinoSound.h е библиотеката, която поддържа I²S комуникация, но резултатите не бяха особено добри, затова системата беше оставена с проектирания вече модел с аналогов микрофон. Линк към взетите резултати и сурс кода може да намерите в литературата. [73]

6.2.4. KickFFT.h

KickFFT.h е библиотека за имплементиране на дискретна трансформация на Фурье върху масив от входни данни. Тази библиотека използва справочни таблици за тригонометричните функции, за да намали процесорната мощност и да увеличи ефективността на кода. Тестовете на библиотеката бяха разочароваващи, резултатите могат да се намерят в хранилището в директория TestOfModulesAndOtherStuff/Samples/.

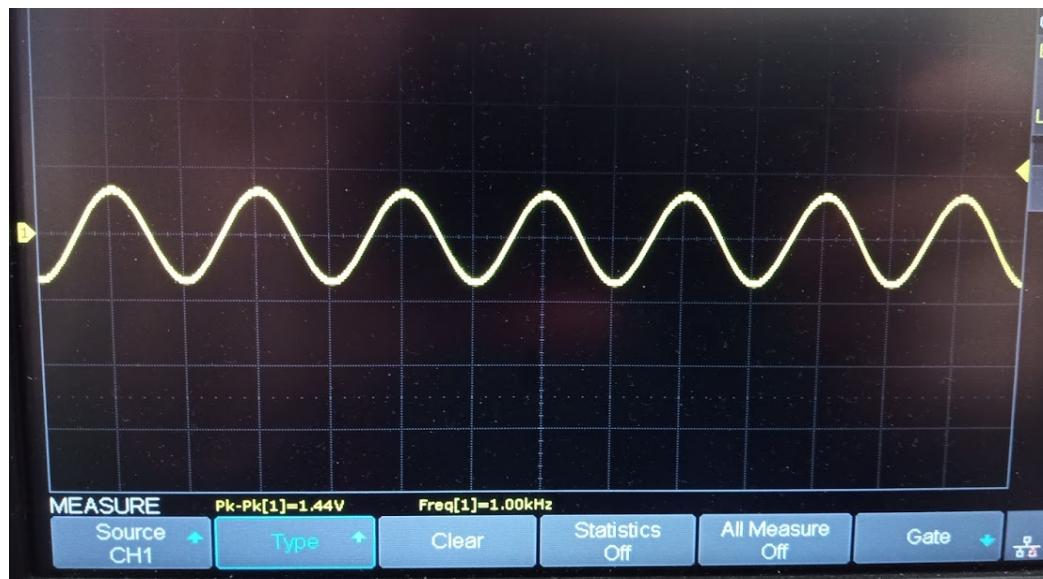
6.2.5. Тест върху функцията и изправността на проектирания филтър

Беше направена постановка с генератор и осцилоскоп, на която беше тестван проектирания филтър в т. 3.2. Постановката е показана на фиг. 6.3.



Фиг. 6.3.

На фиг. 6.4. е показан сигнал с 1 kHz честота и пиково напрежение 1.44 V , а на фиг. 6.5. е показан сигнал с честота 8 kHz и пиково напрежение 640 mV. На фиг. 6.6. е показан сигнал с честота 9 kHz, пиково напрежение 600 mV .На фиг. 6.7. - честота 10 kHz, пиково напрежение 520 mV. На фиг. 6.8. - честота 12 kHz, пиково напрежение 480 mV.



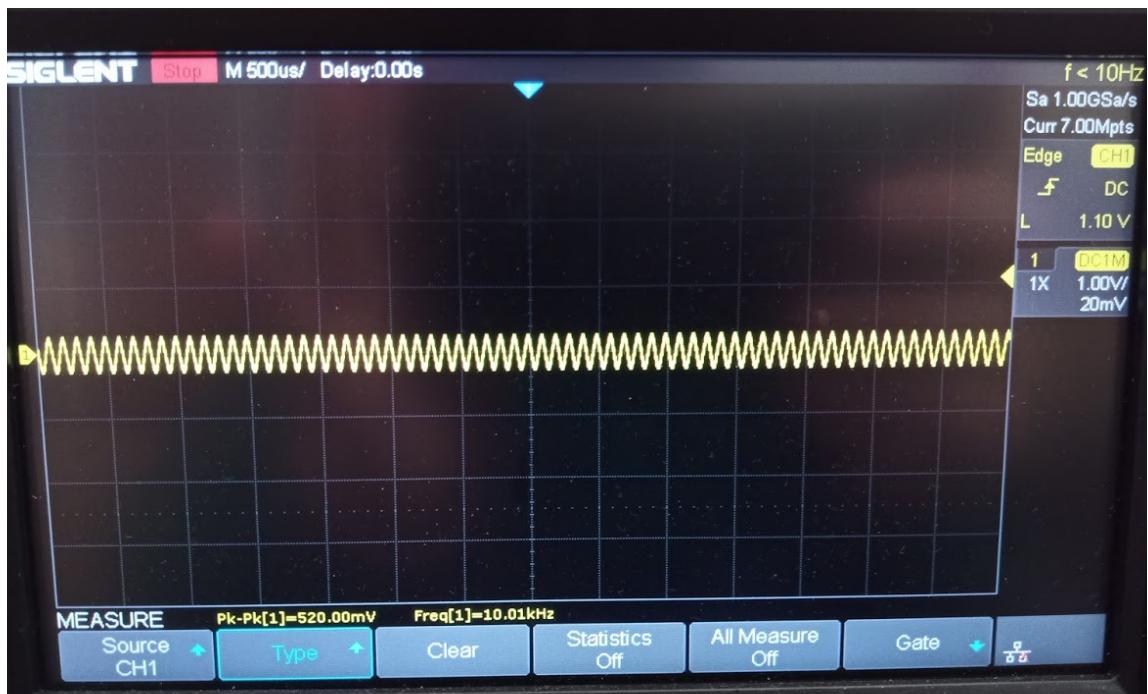
Фиг. 6.4.
1 kHz - честота, 1.44 V - напрежение.



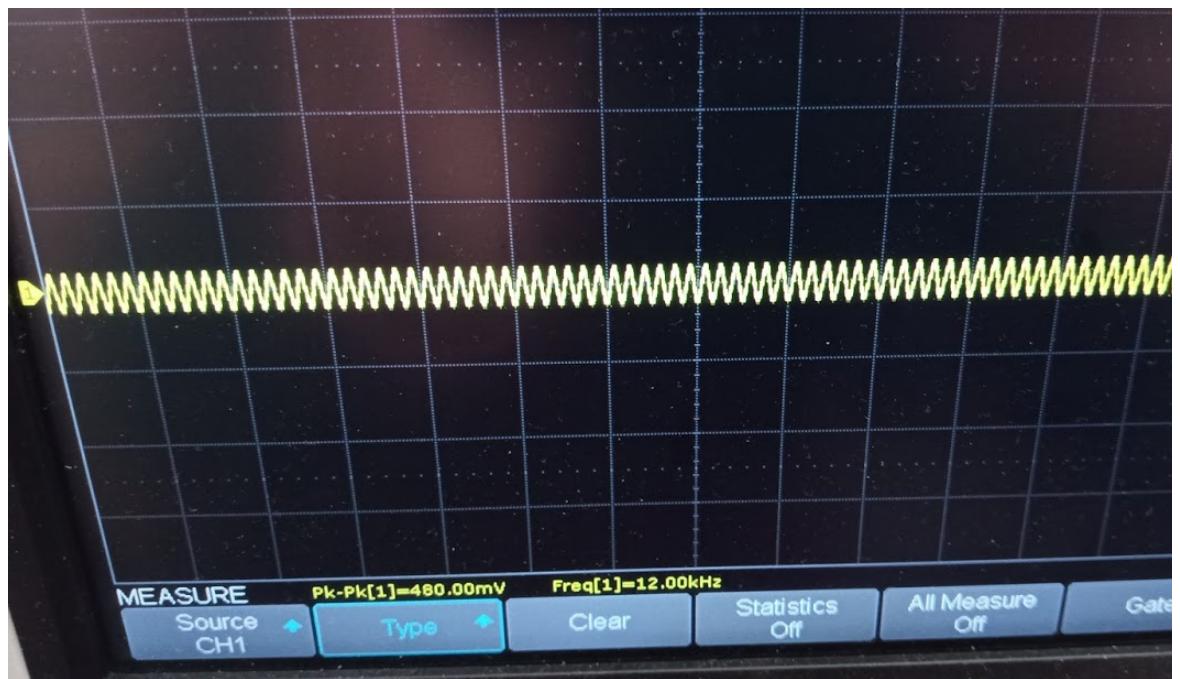
Фиг. 6.5.
8 kHz - частота, 640 мВ - напряжение.



Фиг. 6.6.
9 kHz - частота, 600 мВ - напряжение.



Фиг. 6.7.
10 kHz - частота, 520 mV - напрежение.



Фиг. 6.8.
12 kHz - частота, 480 mV - напрежение.

На база на взетите резултати може да се твърди, че филтъра функционира правилно и според очакванията.

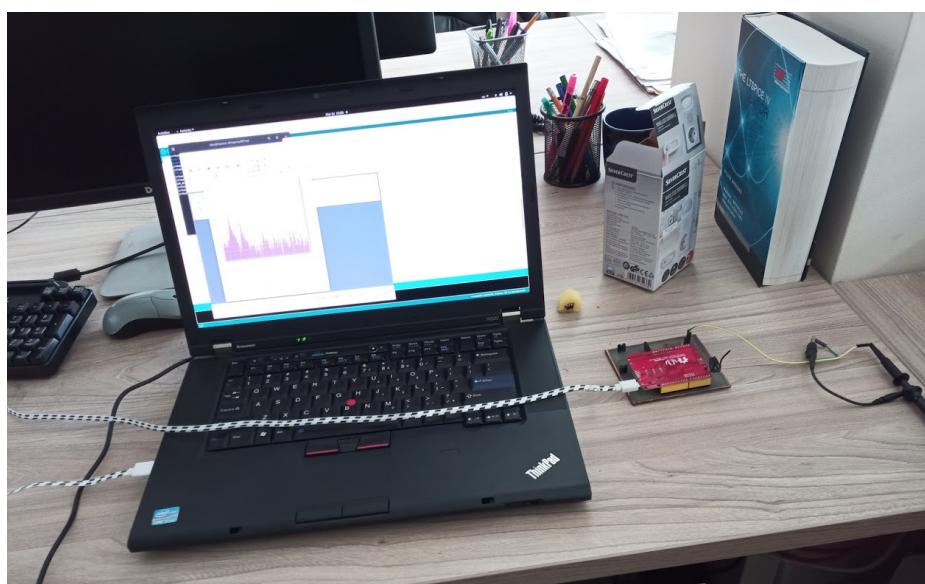
6.2.5. Тест върху Аналогово-цифровия преобразувател и настройките му

Беше проведен тест върху работата на АЦП-то с цел наблюдение на поведението му при подаване на правилна синусоида от генератор на сигнали. Постановката е показана на фиг. 6.9. и фиг. 6.10.



Фиг. 6.9.

Генераторът подава сигнал към АЦП-то на микроконтролера.



Фиг. 6.10.

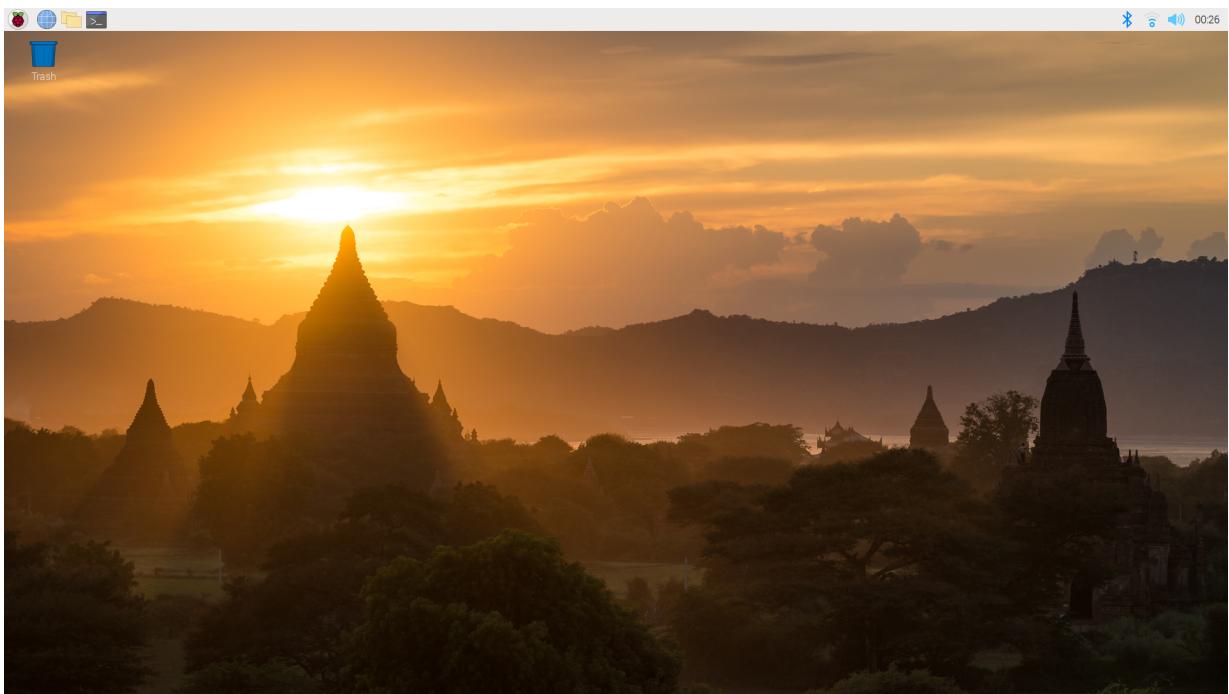
Взетите измервания биват тествани в Audacity, за да се потвърди правилното извличане на резултата.

6.3. Практически резултати и изводи

В тази глава ще бъдат разгледани постигнатите резултати стъпка по стъпка и ще бъдат направени съответните изводи и забележки към поведението на реализирана система в рамките на срока на дипломната работа.

6.3.1. Пускане в действие и инсталиране на Rpi v.3 model B+

Като беше следван справочника посочен в точка 5.2.1 беше инсталрирана ОС върху Rpi. Операционната система е показана на фиг. 6.11.



Фиг. 6.11.

Начален прозорец на операционната система на Raspberry Pi.

6.3.2. Инсталриране на Grafana и InfluxDB и настройването им

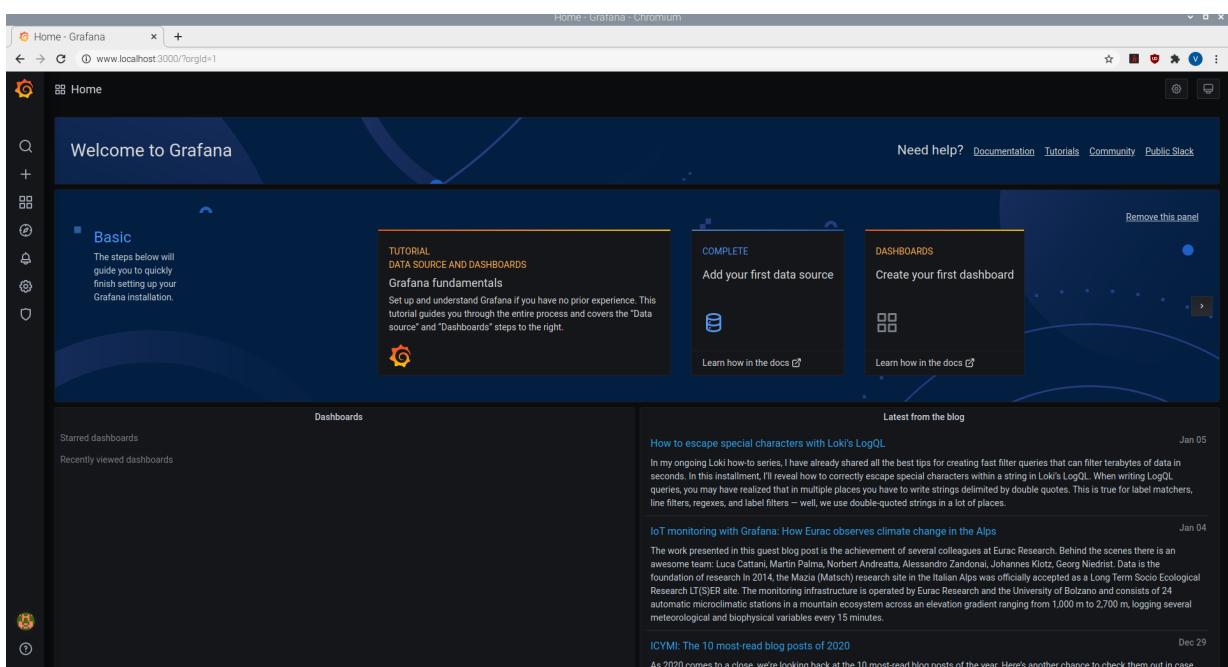
На фиг. 6.12, фиг. 6.13 и фиг. 6.14 са показани “screenshot” на екрана, които показват успешното инсталриране и пускане на Grafana и InfluxDB.

```

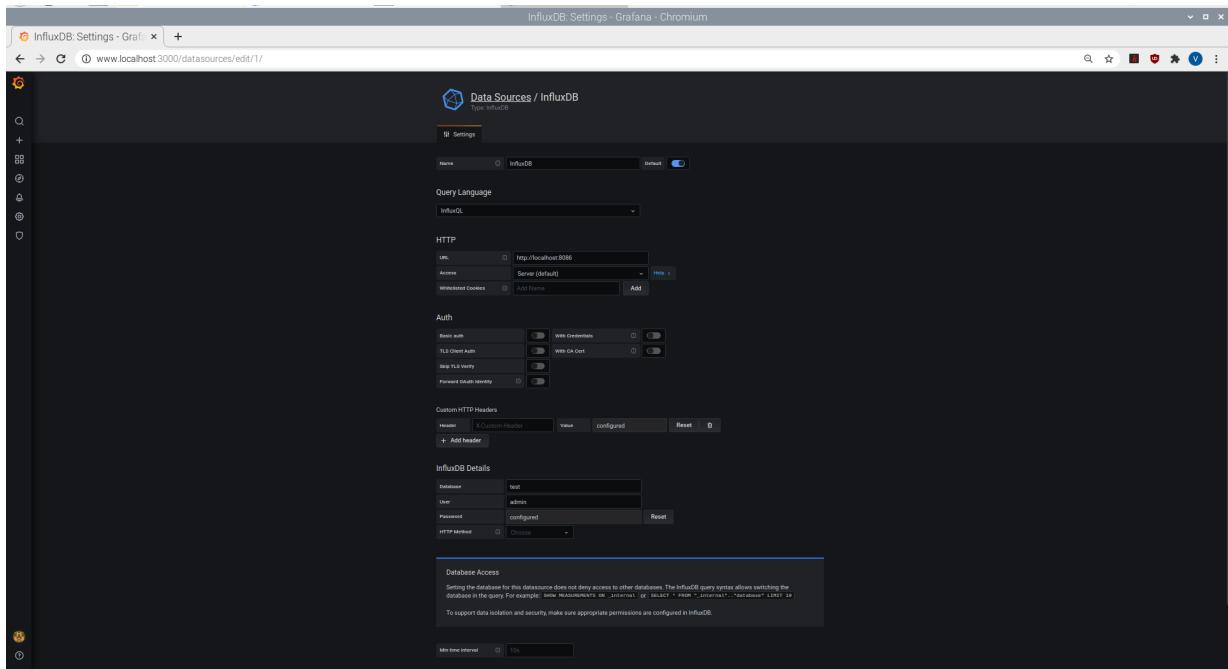
Connected to http://localhost:8086 version 1.8.3
InfluxDB shell version: 1.8.3
> show databases
name: databases
name
-----
_internal
test
>
> 

```

Фиг. 6.12.
Успешно инсталиране и стартиране на InfluxDB.



Фиг. 6.13.
Успешно инсталиране и стартиране на Grafana.

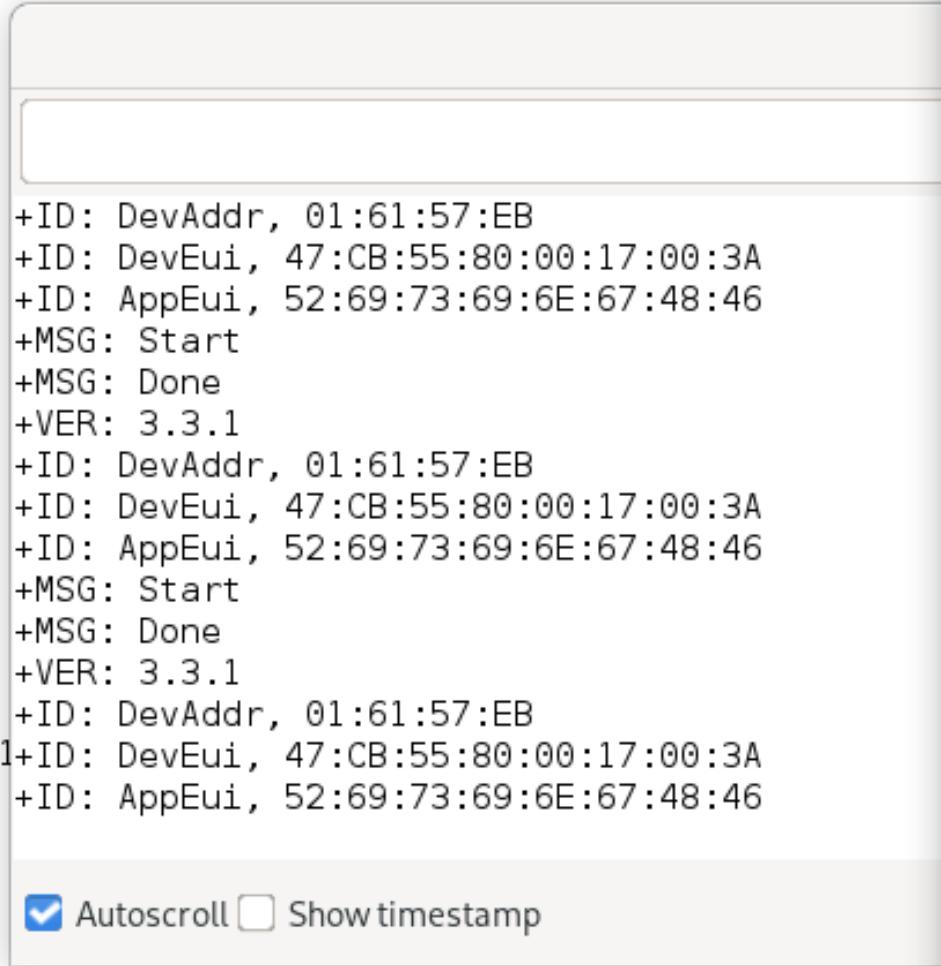


Фиг. 6.14.

Успешно свързване на Grafana към InfluxDB.

6.3.3. Изпращане на данните към приложния сървър

На фиг. 6.15. и фиг. 6.16. може да се види успешното изпращане на данните по LoRaWAN мрежата от крайното устройство до приложния сървър на ChirpStack.



The screenshot shows a terminal window with a light gray background and a white text area. The text displays several lines of data, likely LoRa frames, starting with '+ID' and followed by device addresses and other parameters. At the bottom of the window, there is a toolbar with two checkboxes: one checked ('Autoscroll') and one unchecked ('Show timestamp').

```
+ID: DevAddr, 01:61:57:EB
+ID: DevEui, 47:CB:55:80:00:17:00:3A
+ID: AppEui, 52:69:73:69:6E:67:48:46
+MSG: Start
+MSG: Done
+VER: 3.3.1
+ID: DevAddr, 01:61:57:EB
+ID: DevEui, 47:CB:55:80:00:17:00:3A
+ID: AppEui, 52:69:73:69:6E:67:48:46
+MSG: Start
+MSG: Done
+VER: 3.3.1
+ID: DevAddr, 01:61:57:EB
1+ID: DevEui, 47:CB:55:80:00:17:00:3A
+ID: AppEui, 52:69:73:69:6E:67:48:46

 Autoscroll  Show timestamp
```

Фиг. 6.15.

Успешно изпращане на данни към LoRa приложния сървър.

| DETAILS | CONFIGURATION | KEYS (OTAA) | ACTIVATION | DEVICE DATA | LORAWAN FRAMES | FIRMWARE |
|--|---------------|-------------|------------|-------------|----------------|----------|
| ? HELP PAUSE ⬇️ DOWNLOAD ✖️ CLEAR | | | | | | |
| 2:36:48 PM | up | | | | | ▼ |
| 2:36:18 PM | up | | | | | ▼ |
| 2:35:33 PM | up | | | | | ▼ |
| 2:33:34 PM | up | | | | | ▼ |
| 2:33:19 PM | up | | | | | ^ |
| <pre> applicationID: "1" applicationName: "SoundMeasurementSystem" deviceName: "Beedulino" devEUI: "47cb55800017003a" rxInfo: [] 0 items ▼ txInfo: {} 3 keys frequency: 868100000 modulation: "LORA" ▼ LoRaModulationInfo: {} 4 keys bandwidth: 125 spreadingFactor: 7 codeRate: "4/5" polarizationInversion: false adr: true dr: 5 fCnt: 3 </pre> | | | | | | |

Фиг. 6.16.

6.3.4. Успешно записване на данните в InfluxDB

На фиг. 6.17., фиг. 6.18. и фиг. 6.19. е показана успешна връзка между крайното устройство и сървъра и записване на данните в InfluxDB.

The screenshot shows a terminal window with the following text output:

```
0M
dN
on
e+MSGHEX: Done
o+MSGHEX: Start
ow+MSGHEX: Done
ov+MSGHEX: Start
:+MSGHEX: Done
+MSGHEX: Start
dn+MSGHEX: Done
le+MSGHEX: Start
:+MSGHEX: Done
+MSGHEX: Start
97+MSGHEX: Done
1e+MSGHEX: Start
:+MSGHEX: Done

97it  Autoscroll  Show timestamp
asected to http://localhost:8086 version 1.8.3
```

Фиг. 6.17.

Серийният монитор показва успешно изпратени съобщения.

фиг. 6.18.

Успешно получаване на данните от приложния сървър.

```

> select * from SoundMeasurementSystem
name: SoundMeasurementSystem
time           amplitude counter devEUI          frequency gas   humidity pressure temperature
----           -----
2021-01-30T02:09:29.742238673Z 80      0     47cb55800017003a 4440    8065.16 100    728.67  33.95
> select * from SoundMeasurementSystem
name: SoundMeasurementSystem
time           amplitude counter devEUI          frequency gas   humidity pressure temperature
----           -----
2021-01-30T02:09:29.742238673Z 80      0     47cb55800017003a 4440    8065.16 100    728.67  33.95
2021-01-30T02:10:34.673444764Z 80      1     47cb55800017003a 4440    211.64 56.822  926.76  18.95
2021-01-30T02:11:39.604338952Z 80      2     47cb55800017003a 4440    65.53   56.963  926.76  18.95
2021-01-30T02:12:44.512323384Z 80      3     47cb55800017003a 4440    87.95   56.754  926.76  18.94
2021-01-30T02:13:49.426446628Z 80      4     47cb55800017003a 4440    103.59  56.642  926.74  18.92
2021-01-30T02:14:54.346633098Z 80      5     47cb55800017003a 4440    115.96  56.605  926.7   18.9
2021-01-30T02:15:59.265315455Z 80      6     47cb55800017003a 4440    127.06  56.622  926.7   18.87
2021-01-30T02:17:04.176651571Z 80      7     47cb55800017003a 4440    134.8   56.663  926.68  18.85
> []

```

Фиг. 6.19.

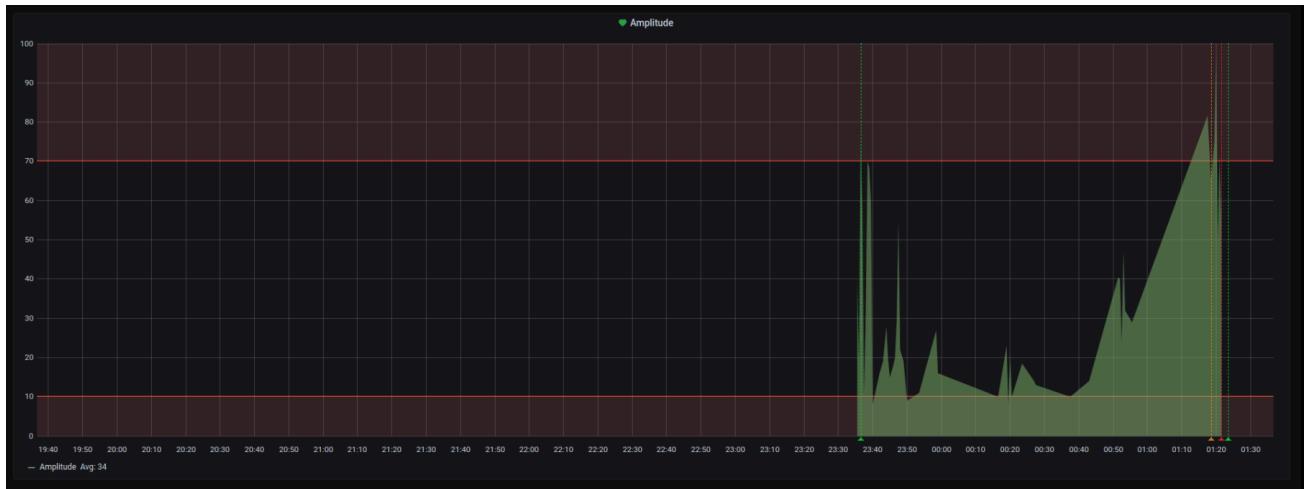
Успешно записване на данните в InfluxDB.

6.3.5. Визуализация на данните в браузъра и изпращане на алермиращи имейли

На по долните фигури са показани реализираните практически резултати, които бяха постигнати по време на дипломната работа.



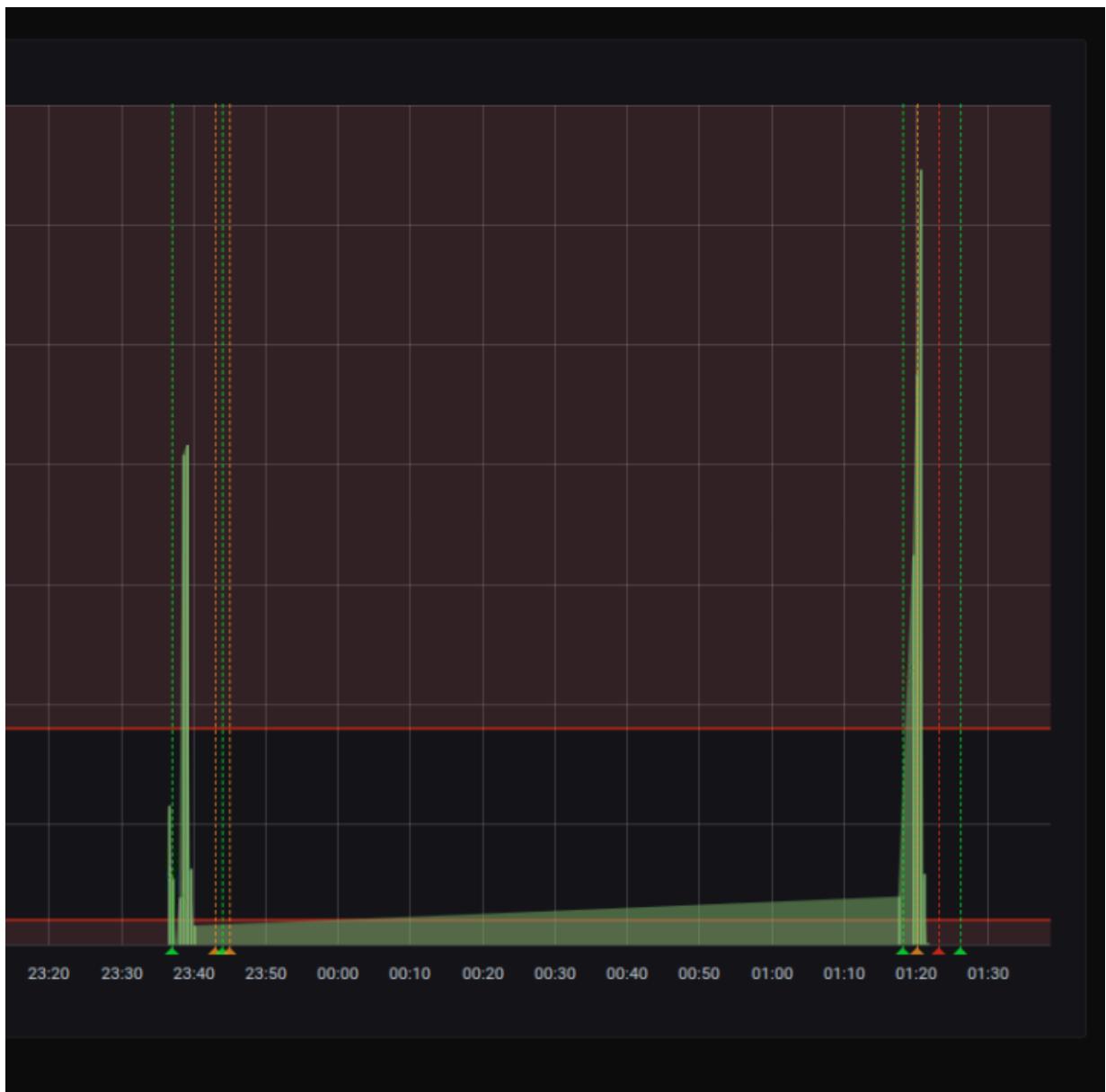
Фиг. 6.20.
Изглед на цялото табло.



Фиг. 6.21.
Амплитудният граф - узголемен.



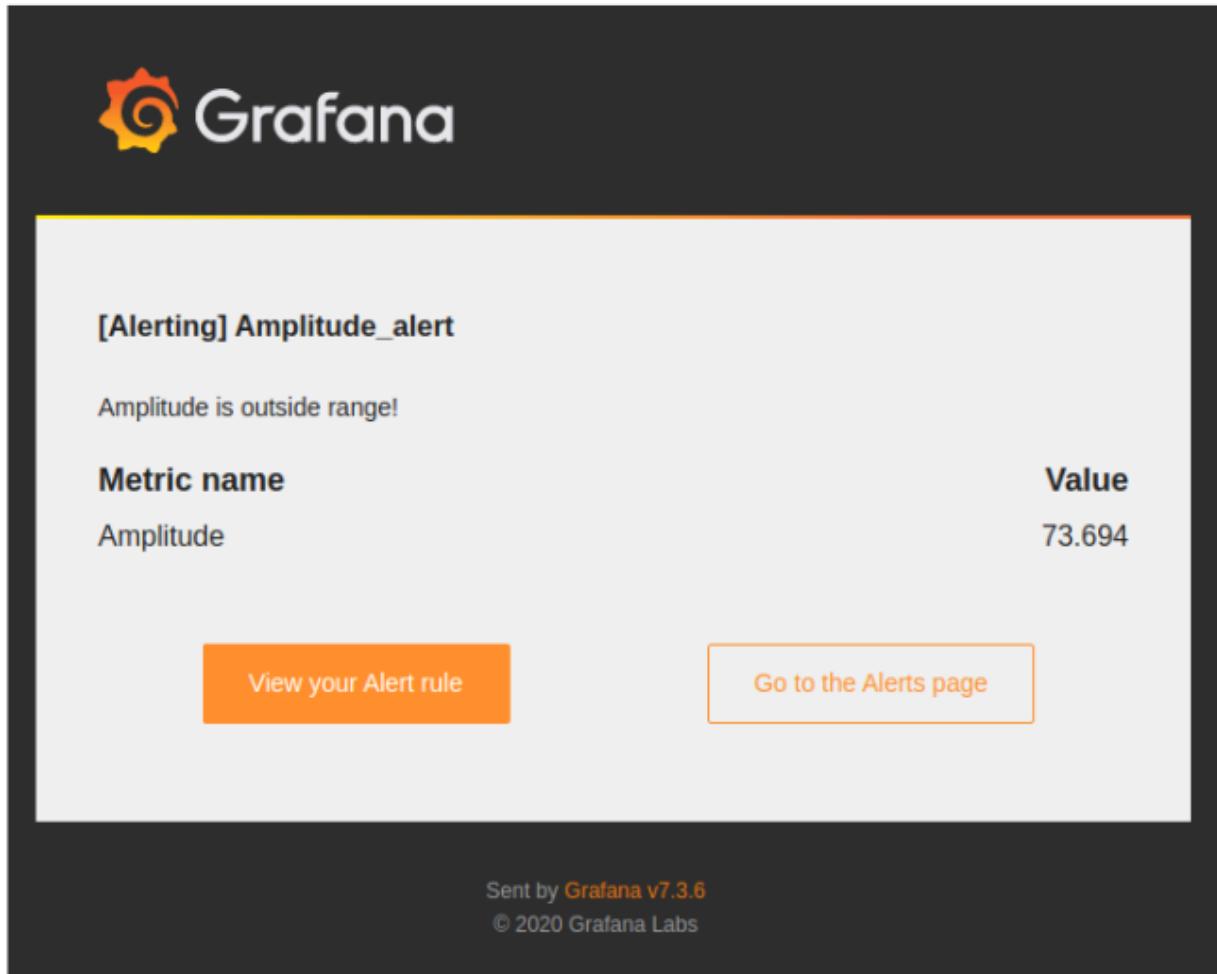
Фиг. 6.22.
Честотният граф - узголемен.



Фиг. 6.23.

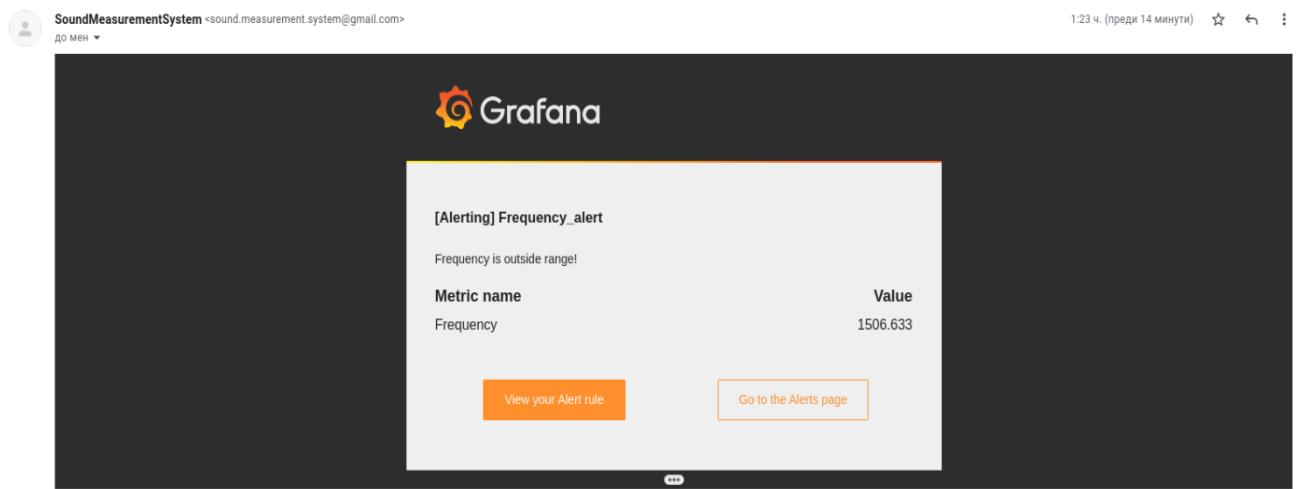
Честотният граф - изрязване от активната област.

Когато между две измервания няма стойности се свързват двете стойности като празното място се запълва със стойността на първата от двете стойности.



Фиг. 6.24.

Предупреждение изпратено по имейл, което известява за странно поведение на амплитудата.

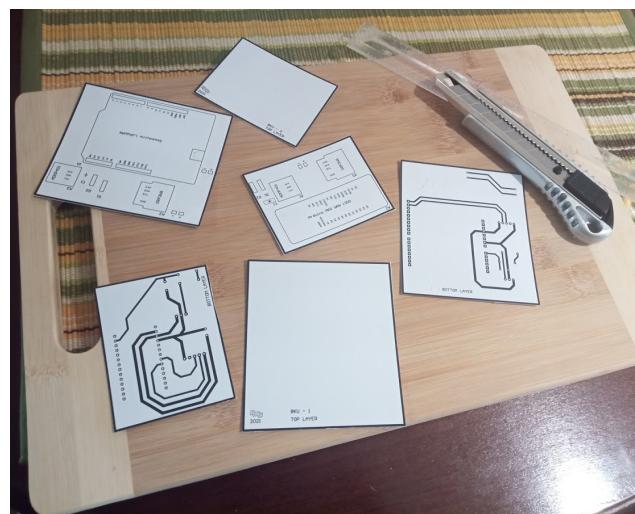


Фиг. 6.25.

Предупреждение, изпратено по имейл, което съобщава за странно поведение на честотата.

6.3.6. Прототипиране на устройството

За прототипиране на устройството беше използвана техниката на тонер трансфер от гланцова хартия върху текстолит и за ецващ разтвор беше използван железен трихлорид. Процесът на прототипирането е показан на фиг. 6.26. - фиг. 6.36.



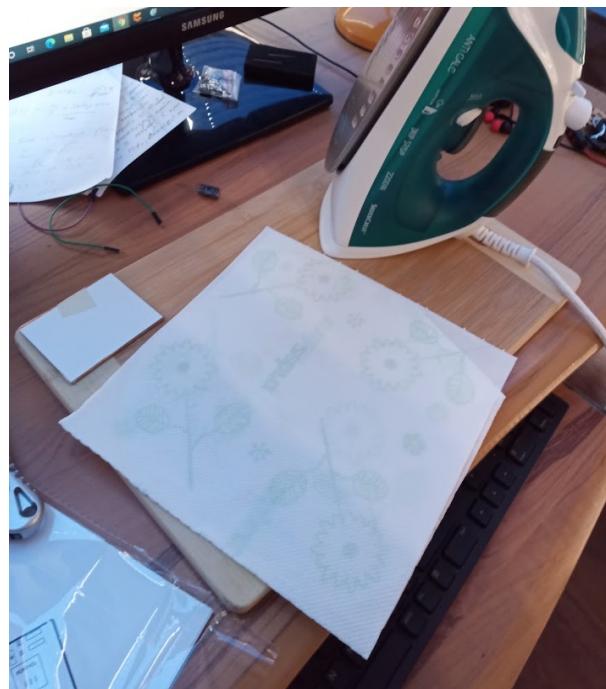
Фиг. 6.26.

Подготвяне на моделите за отпечатване.



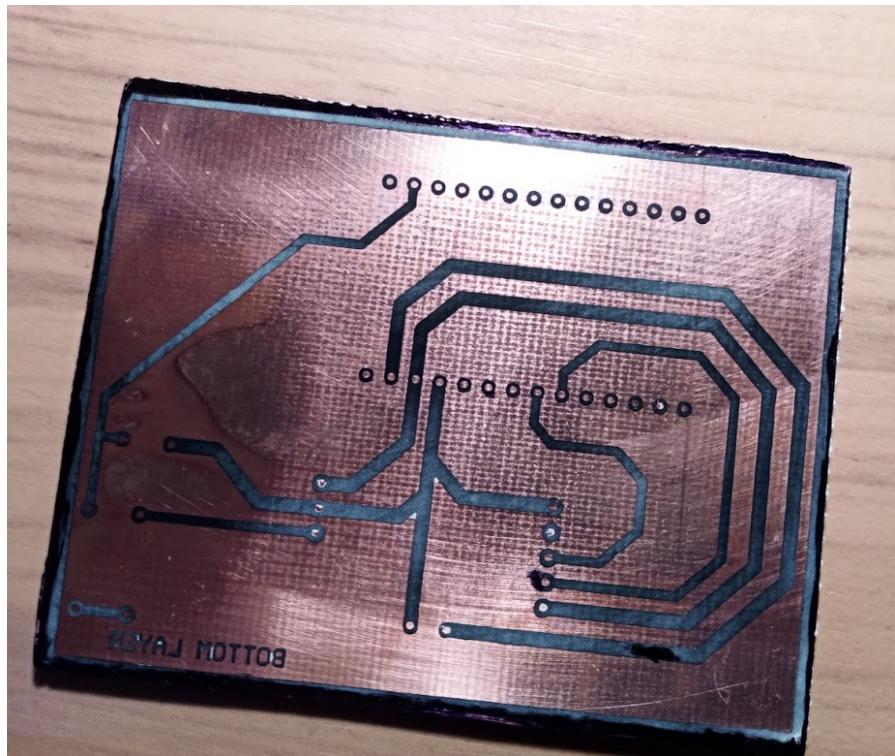
Фиг. 6.27.

Изрязване на текстолитна плоча с подходящи размери.



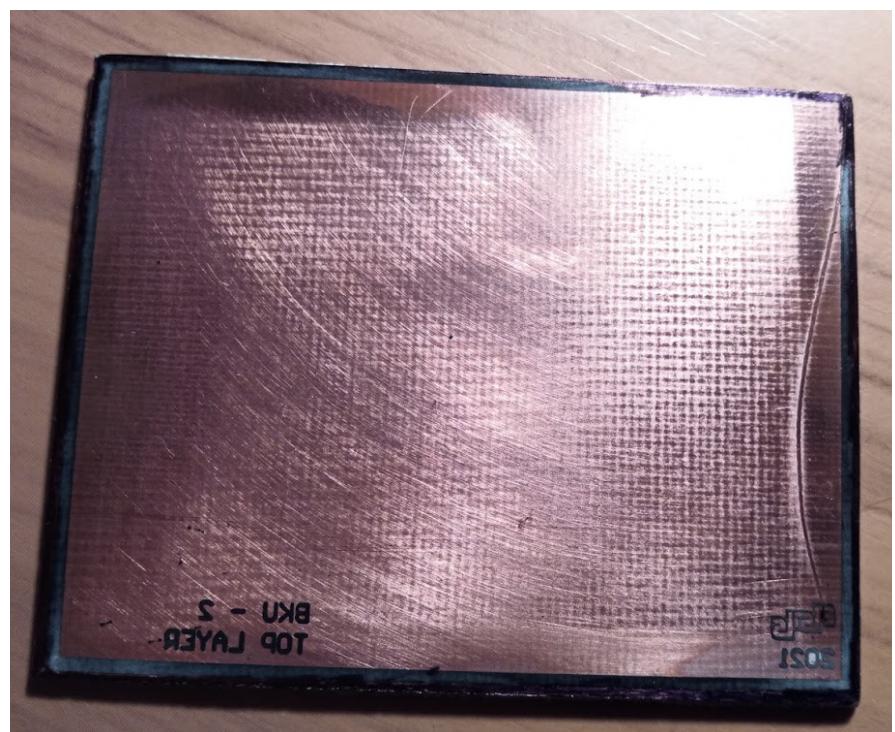
Фиг. 6.28.

Пренасяне на тонера от моделите върху плочата.



Фиг. 6.29.

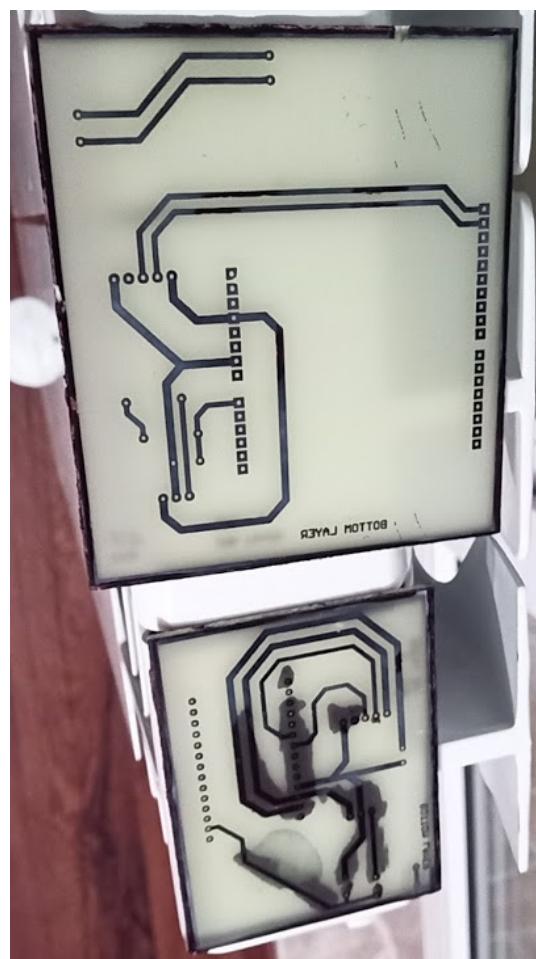
Резултат от тонер трансфера. Надписите не бяха забелязани навреме и излязоха огледално.



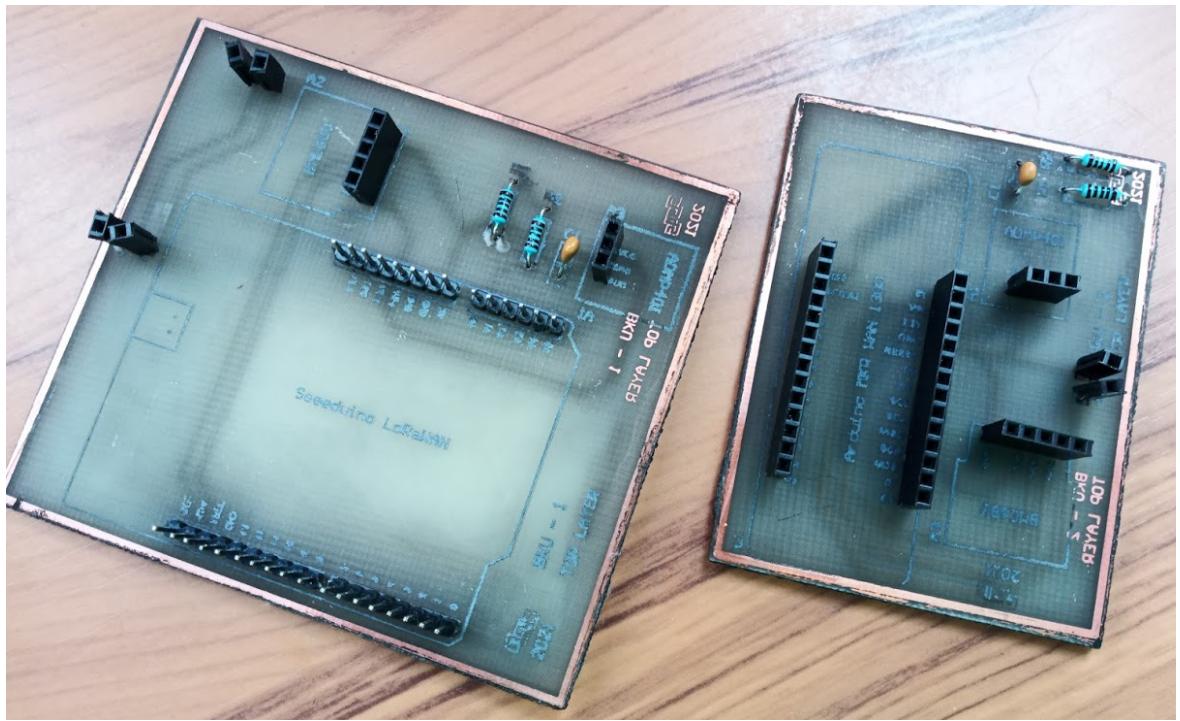
Фиг. 6.30.



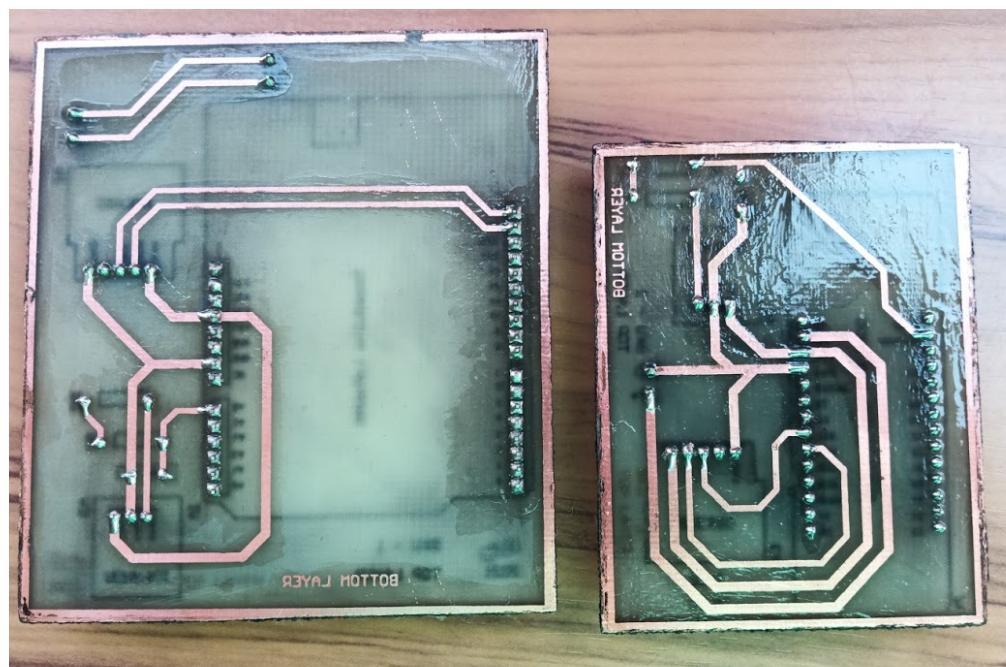
Фиг. 6.31.
Процес на ецване на печатните платки.



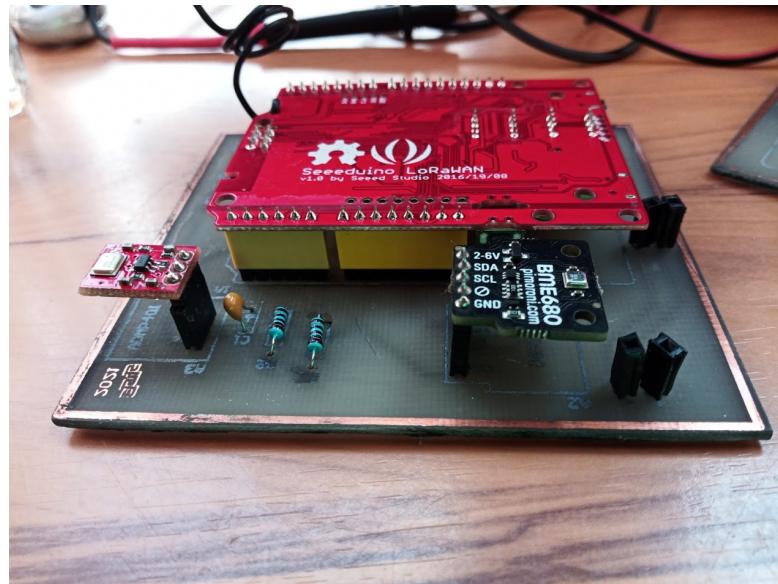
Фиг. 6.32.
Резултат след ецването на печатните платки.



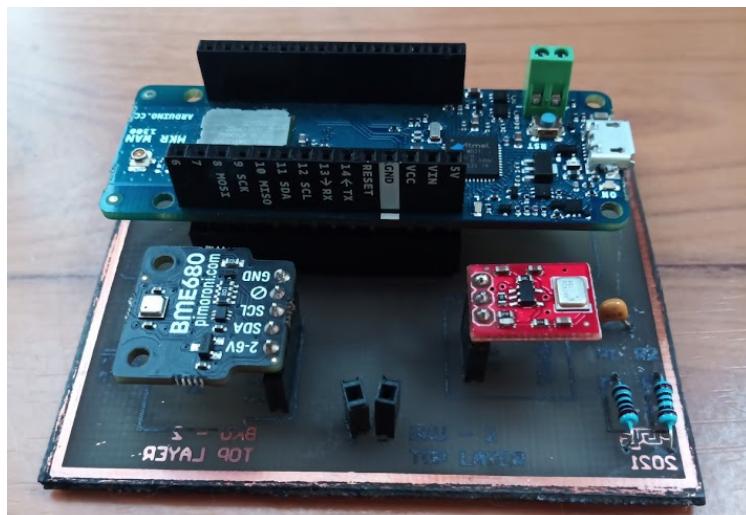
Фиг. 6.33.
Монтиране на компонентите върху печатните платки.



Фиг. 6.34.
Пистите на платките бяха лакирани с цел предпазване от корозия.



Фиг. 6.35.



Фиг. 6.36.

Заключение

В настоящата дипломна работа е представена система за анализ и идентификация на звукови сигнали, която освен това може да измерва температура, атмосферно налягане, влажност на въздуха и наличие на някои замърсявания на околната среда. Постигната е двупосочна LoRaWAN връзка от крайните устройства към сървъра, която поддържа повече от едно устройство едновременно.

В първа глава е представено проучване на методите за анализ на звукови сигнали, на някои съществуващи системи за анализ на такива сигнали, както и на подходящи платформи, модули и технологии за реализиране на подобна система.

Във втора глава са дефинирани основните изисквания към системата и са синтезирани общата ѝ блокова схема, по-детайлните схеми на отделните блокове, както и техните взаимовръзки и функционалност.

В трета глава са представени електрическите схеми на използваните модули и компоненти и са проектирани принципните електрически схеми на крайните устройства.

В четвърта глава са дадени печатните платки на използваните модули и са проектирани печатните платки на крайните устройства.

В пета глава са синтезирани функционалните диаграми и алгоритми на работа на управляващия софтуер, като са представени конфигурации и сурс код на отделните модули на софтуера на системата.

В шеста глава са изложени някои проблеми, възникнали по време на практическата реализация на дипломната работа, както и техните решения, ако са намерени такива. Представени са проведените тестове и постигнатите практически резултати.

Въз основа на всичко описано по-горе може да се твърди, че целта на настоящата дипломна работа е изпълнена.

Бъдещото развитие на системата може да бъде в следните насоки:

- Създаване на захранващ модул, съдържащ слънчев панел и акумулаторна батерия с цел да се поддържа по-дълготрайна и независима работа на системата;
- Анализиране на получените от крайните устройства данни посредством подходящи алгоритми и изкуствен интелект с цел прогнозиране на определени събития.

Използвана литература

1. Гласът на медоносните пчели,
http://www.sama.ru/~yerko/yerko_bg/articl_10.htm
2. Fundamental of acoustics,
https://www.who.int/occupational_health/publications/noise1.pdf
3. The Fourier Transform in a Nutshell,
https://www.researchgate.net/publication/290440858_The_Fourier_Transform_in_a_Nutshell
4. Nyquist-Shannon sampling theorem,
https://home.strw.leidenuniv.nl/~por/AOT2019/docs/AOT_2019_Ex13_NyquistTheorem.pdf
5. Introduction to the Fourier Transform,
<https://lpsa.swarthmore.edu/Fourier/Xforms/FXformIntro.html>
6. Измерване Измерване на спектри спектри на сигнали сигнали.
Спектроанализатори. ,
http://www.phys.uni-sofia.bg/~dankov/P%20Dankov_Lecture%20materials/Microwave%20measurements/MWM_Lecture4_Spectrum.pdf
7. SM90 Class 1 Measuring Instrument,
<https://bedrock-usa.com/sm90-class-1-measuring-instrument/>
8. FLEXUS FX100 Audio Analyzer,
<https://www.nti-audio.com/en/products/flexus-fx100-audio-analyzer>
9. Спектрален анализатор “Направи си сам” на базата на Ардуино - 1,
<https://create.arduino.cc/projecthub/mircemk/diy-fft-audio-spectrum-analyzer-ca2926>
10. Спектрален анализатор на базата на Ардуино - 2,
<https://create.arduino.cc/projecthub/shajeeb/32-band-audio-spectrum-visualizer-analyzer-902f51>
11. OLED Spectrum Analyzer W/arduino & MSGEQ7,

[https://www.instructables.com/OLED-Spectrum-Analyzer-Warduino-MSGE
Q7/](https://www.instructables.com/OLED-Spectrum-Analyzer-Warduino-MSGE-Q7/)

12. ESP32/ ESP-EYE: Browser Based Spectrum Analyzer,

[https://blog.squix.org/2019/08/esp32-esp-eye-browser-based-spectrum-a
nalyzer.html](https://blog.squix.org/2019/08/esp32-esp-eye-browser-based-spectrum-analyzer.html)

13. Adobe Audition,

<https://www.adobe.com/bq/products/audition.html>

14. Spectralissime, the Spectrum Analyzer - VB-Audio Software

<https://vb-audio.com/Spectralissime/>

15. Sonic Visualiser,

<https://www.sonicvisualiser.org/>

16. Audacity,

<https://www.audacityteam.org/>

17. Friture,

<http://friture.org/features.html>

18. Raspberry Pi 3 Model B+,

<https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>

19. dragino - SX127X GPS HAT,

<http://www.dragino.com/products/lora/item/106-lora-gps-hat.html>

20. RAK831,

[https://www.rakwireless.com/en-us/products/lpwan-gateways-and-concen
trators/rak831](https://www.rakwireless.com/en-us/products/lpwan-gateways-and-concen
trators/rak831)

21. Arduino - Arduino Uno,

<https://store.arduino.cc/arduino-uno-rev3>

22. SX127x Shield за Arduino,

<https://www.dragino.com/products/lora/item/102-lora-shield.html>

23. Seeeduino LoRaWAN,

<https://www.seeedstudio.com/Seeeduino-LoRaWAN-p-2780.html>

24. MKR WAN 1300,

<https://store.arduino.cc/arduino-mkr-wan-1300-lora-connectivity-1414>

25. ESP32,

<https://www.espressif.com/en/products/socs/esp32>

26. ADMP401 микрофон,

<https://www.analog.com/en/products/admp401.html>

27. Adafruit I2S MEMS Microphone Breakout - SPH0645LM4H,

<https://www.adafruit.com/product/3421>

28. BME680 модул,

<https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors-bme680/>

29. arduinoFFT.h,

<https://www.arduino.cc/reference/en/libraries/arduinofft/>

30. fix_fft.h,

https://www.arduino.cc/reference/en/libraries/fix_fft/

31. DSP библиотеки за Cortex M3 и други ARM процесори,

<https://www.embeddedsignals.com/ARM.htm>

32. CMSIS-DSP библиотека,

<https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html>

33. NumPy,

<https://numpy.org/>

34. Grafana,

<https://grafana.com/>

35. Matplotlib,

<https://matplotlib.org/>

36. Audacity Scripting,

<https://manual.audacityteam.org/man/scripting.html>

37. InfluxDB,

<https://www.influxdata.com/products/influxdb/>

38. PostgreSQL,

<https://www.postgresql.org/about/>

39. Redis,

<https://redis.io/>

40. Сравнителна таблица на ДБ,

<https://db-engines.com/en/system/InfluxDB%3BPostgreSQL%3BRedis>

41. CircuitMaker,

<https://circuitmaker.com/About>

42. Raspberry Pi 3 model B+ електрическа схема,

https://www.raspberrypi.org/documentation/hardware/raspberrypi/schemas/rpi_SCH_3bplus_1p0_reduced.pdf

43. RAK831 документация,

<https://docs.rakwireless.com/Product-Categories/WisLink/RAK831/Datasheet/>

44. Seeeduino LoRaWAN документация,

https://wiki.seeedstudio.com/Seeeduino_LoRAWAN/#schematic-online-viewer

45. MKR WAN 1300 схема,

https://content.arduino.cc/assets/MKRWANV1.0_sch.pdf

46. ADMP401 електрическа схема,

<https://www.sparkfun.com/datasheets/BreakoutBoards/ADMP401-Breakout-v13.pdf>

47. CircuitMaker SeeeduinoLoRaWAN линк,

<https://workspace.circuitmaker.com/Projects/Details/ViktorTodorov/Seeeduino-LoRaWAN>

48. CircuitMaker MKRWAN_1300 линк,

<https://workspace.circuitmaker.com/Projects/Details/ViktorTodorov/MKRWAN1300>

49. MKR WAN 1300 pinout,

https://content.arduino.cc/assets/Pinout-MKRWAN1300_latest.pdf

50. ADMP401 модул Sparkfun,

<https://www.sparkfun.com/products/9868>

51. Какво е LoRa и LoRaWAN,

<https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/>

52. LoRa Alliance,

<https://lora-alliance.org/>

53. LoraWAN 1.0.2 спецификация,

https://lora-alliance.org/resource_hub/lorawan-specification-v1-0-2/

54. ChirpStack,

<https://www.chirpstack.io/>

55. rak_common_for_gateway хранилище,

https://github.com/RAKWireless/rak_common_for_gateway

56. Adafruit BME680 Library,

https://github.com/adafruit/Adafruit_BME680

57. MKRWAN.h,

<https://www.arduino.cc/en/Reference/MKRWAN>

58. LoRaWAN.h,

<https://github.com/VikoTodorov/sound-lora-system/tree/main/DocsAndSpecs/LoRaWan>

59. Adafruit ZeroFFT,

https://github.com/adafruit/Adafruit_ZeroFFT

60. InfluxDB 1.8. документация,

<https://docs.influxdata.com/influxdb/v1.8/>

61. influxdb-client-python,

<https://github.com/influxdata/influxdb-client-python>

62. Raspberry Pi наръчник за инсталация,

<https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up>

63. Quickstart Debian or Ubuntu,

<https://www.chirpstack.io/project/guides/debian-ubuntu/>

64. Grafana smtp конфигурация,

<https://techexpert.tips/grafana/grafana-email-notification-setup/>

65. InfluxDB инсталиране,

<https://pimylifeup.com/raspberry-pi-influxdb/>

66. ChirpStack http интеграция,

<https://www.chirpstack.io/application-server/integrations/http/>

67. ChirpStack API хранилище - python,

<https://github.com/brocaar/chirpstack-api/tree/master/python/src>

68. sound-lora-system,

<https://github.com/VikoTodorov/sound-lora-system>

69. Решение на проблема с декодиращата функция,

<https://forum.chirpstack.io/t/work-with-strings-and-regex-in-decode-function-issue-why/10253/4>

70. LoRa_GPS_Hat User Manual,

http://www.dragino.com/downloads/downloads/LoRa-GPS-HAT/LoRa_GPS_HAT_UserManual_v1.0.pdf

71. AudioFrequencyMeter,

https://github.com/VikoTodorov/sound-lora-system/tree/main/TestOfModulesAndOtherStuff/AudioFrequencyMeter_test

72. ZeroFFT_test,

https://github.com/VikoTodorov/sound-lora-system/tree/main/TestOfModulesAndOtherStuff/ZeroFFT_test

73. ArduinoSound_test,

https://github.com/VikoTodorov/sound-lora-system/tree/main/TestOfModulesAndOtherStuff/I2S_with_ArduinoSound.h

Приложения

П1. Електрическа схема на Raspberry 3 Model B+ [42]

П2. Електрическа схема на RAK831 [43]

П3. Електрическа схема на Seeeduino LoRaWAN [44]

П4. Електрическа схема на MKR WAN 1300 [45]

П5. Електрическа схема на ADMP401 [46]

П6. Електрическа схема на ВМЕ680 модул

**П7. Електрическа схема и печатна платка на БКУ - вариант 1,
Seeeduino LoRaWAN [47]**

**П8. Електрическа схема и печатна платка на БКУ - вариант 2,
MKR WAN 1300 [48]**

П9. MKR WAN 1300 pinout [49]