Введение

На сегодняшний день большое распространение получили технологии машинного обучения, в частности — модели нейронных сетей. Они позволяют анализировать самые разнообразные данные, в том числе в условиях невозможности формализовать критерии, по которым, например, следует классификация. Однако, для реализации моделей нейронных сетей необходимы большие объемы размеченных данных (датасет).

Одним из направлений, в котором активно используются методы машинного обучения, является определение различных параметров среды на основе данных о канале связи (CSI — Channel State Information) связанных по технологии Wi-Fi устройств. Таким образом, актуальна проблема разработки автоматизированной системы сбора канальных матриц, что и является целью работы.

Основными задачам работы являются:

1. Проектирование концептуальной, логической и физической моделей базы данных.

2. Реализация физической модели.

3. Разработка приложения для взаимодействия пользователя с базой данных.

1. Анализ технического задания.

В соответствии с техническим заданием, необходимо разработать автоматизированную систему сбора канальных матриц для обучения модели нейронной сети.

Канальные матрицы — это матрицы, описывающие распространение сигнала между устройствами в пространстве. В нормальных условиях устройства используют их для предотвращения ряда нежелательных явлений, таких, как, например, приём переотраженного сигнала. Каждый элемент матрицы представляет из себя комплексную амплитуду, т. е. описывает амплитуду и фазу принятого сигнала. Структура рассматриваемых данных представлена на рисунке 1 [1].

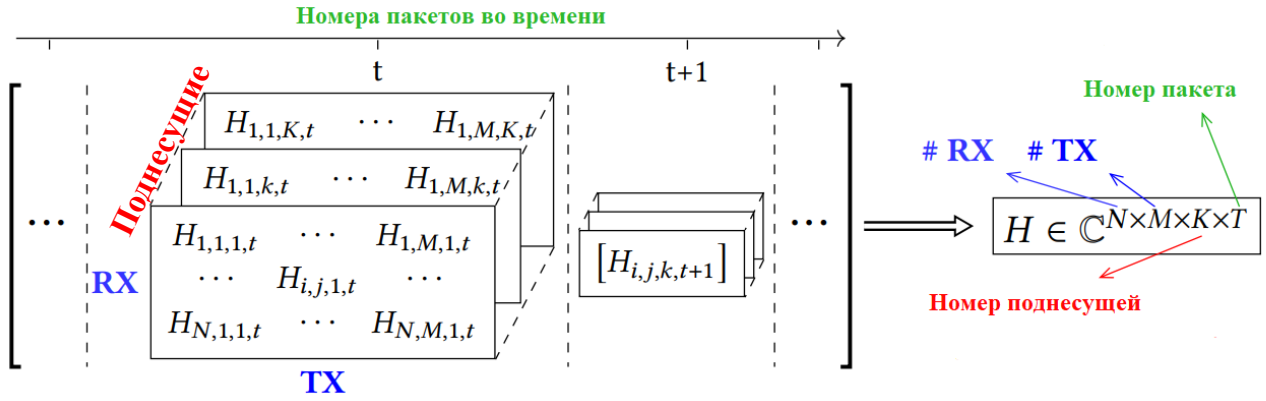

Рисунок 1 - структура канальных матриц.

Поскольку система предназначена для сбора и хранения данных для последующего их использования при обучении нейронной сети, то требуется возможность в маркировке данных. Кроме того, для возможности автоматической маркировки, и/или получения дополнительных сведений, необходима возможность автоматической фотофиксации анализируемой среды.

Для обеспечения удобства при обучении нейронных сетей, необходима реализация выгрузки датасета в формате JSON, т. к. большинство библиотек машинного обучения реализованы для сочетания языка программирования Python и библиотеки NumPy, позволяющей достаточно легко загружать данные из файлов JSON.

Поскольку уже существует система, позволяющая хранить отдельные датасеты в отдельных базах данных, то необходимо так же предусмотреть возможность импорта данных из старого формата в новую базу данных.

Т.к. данные о канале связи возможно получать из самых разных устройств (например, роутеров или микроконтроллеров серии ESP32), то необходимо также предусмотреть возможность записи используемого оборудования и расширения функциональности информационной системы путём добавления новых модулей, осуществляющих приём и предобработку данных.

Для удобства пользования системой и возможности визуализации получаемых данных, информационная система должна иметь графический пользовательский интерфейс.

2. Разработка моделей данных

2.1. Концептуальная модель

По заданию известно, что в каждом датасете применяется конкретный набор оборудования. Одно оборудование осуществляет передачу, другое — приём. Оборудование имеет две ключевые характеристики, отражающиеся на структуре принимаемых данных: количество активных антенн и количество поднесущих (соседних не пересекающихся частот). Один и тот же тип оборудования может выступать как в роли передатчика, так и в роли приёмника. Приёмник передает данные модулю информационной системы для последующих обработки и хранения. Далее представлен фрагмент концептуальной модели (рисунок 2).

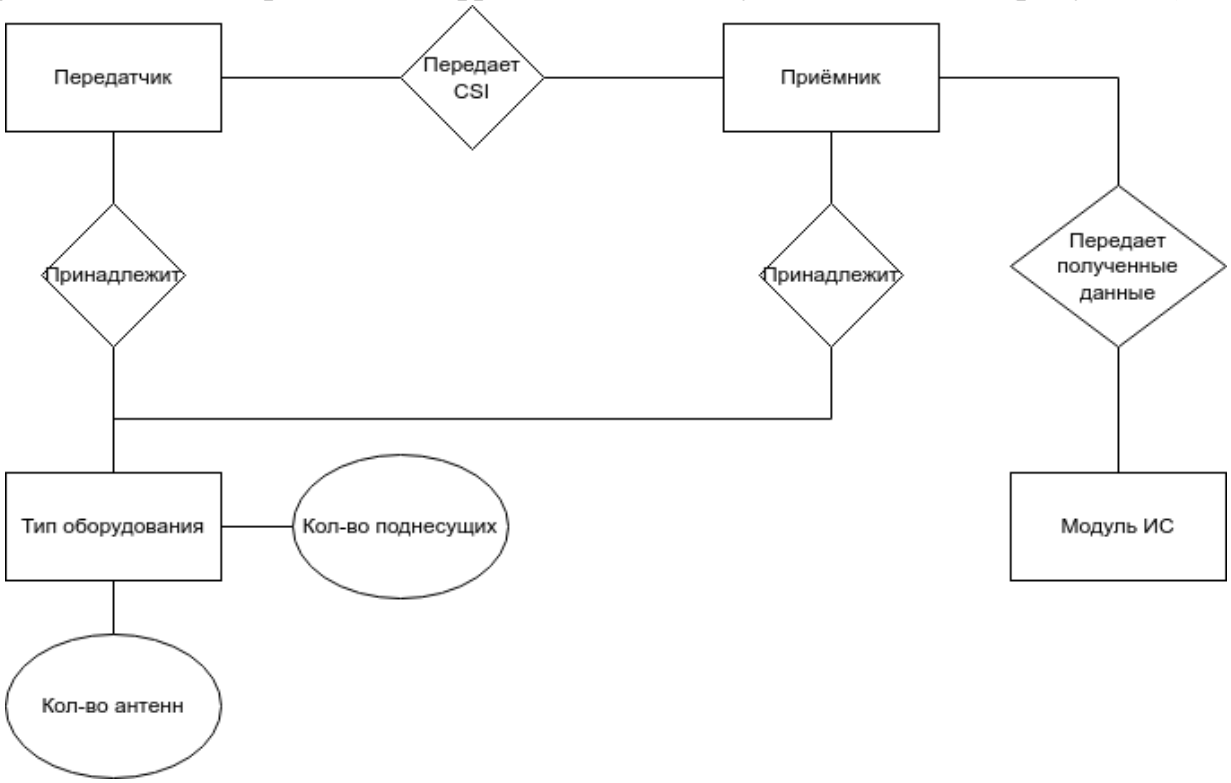

Рисунок 2 - Связь между оборудованием и ИС

Полученные модулем данные могут быть предобработаны другим модулем. Поскольку у модулей могут иметься изменяемые параметры (например, сетевой порт, на который приходят данные от приёмника), то необходима система для конфигурации этих настроек, например — хранение их в файле формата JSON, что достаточно удобно как на программном уровне, так и для пользователя. Полученные и предобработанные данные сохраняются, а затем происходит

дополнительная обработка — выделение амплитуды и фазы. Далее представлен фрагмент концептуальной модели (рисунок 3).

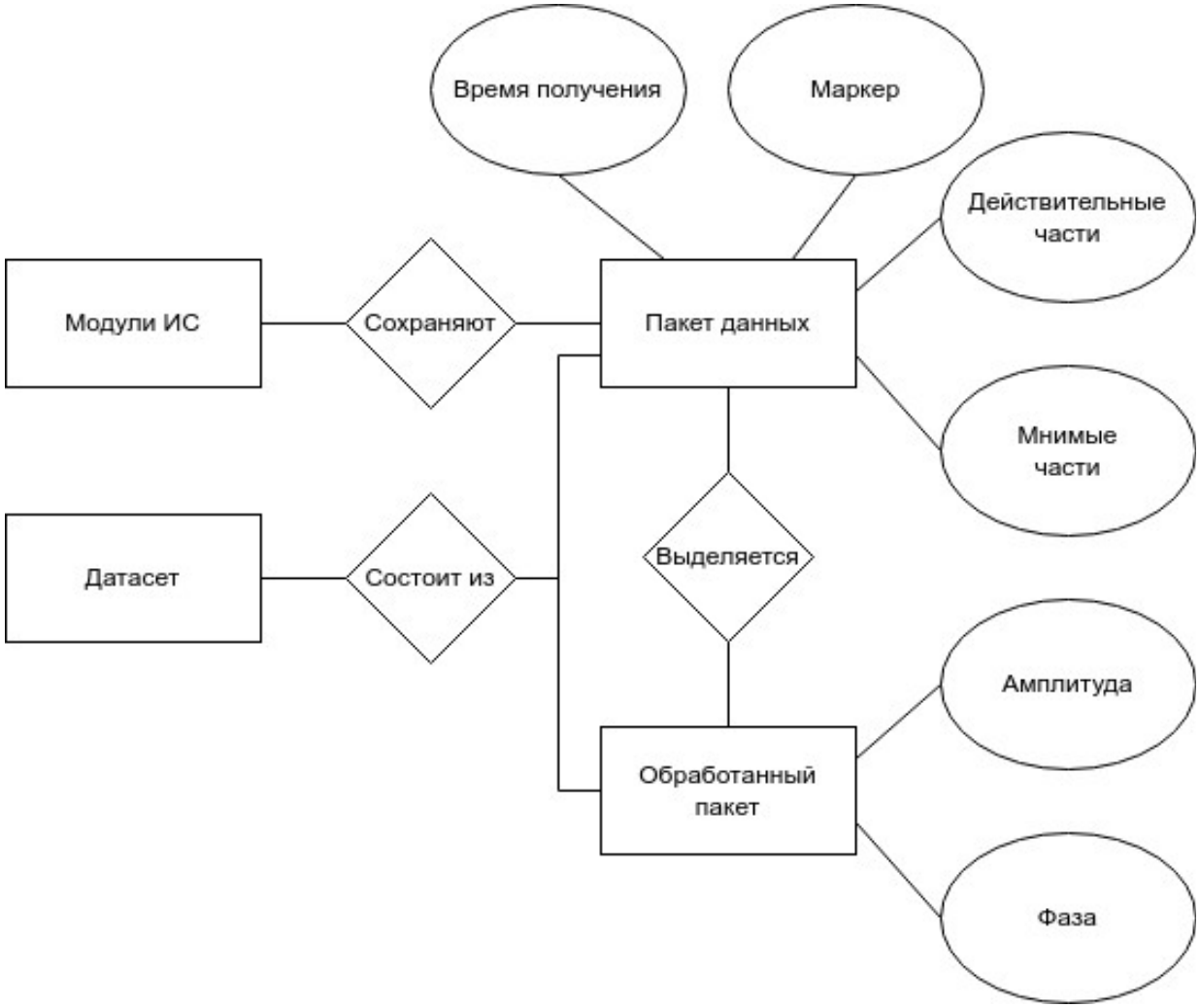

Рисунок 3 - Связь модулей и хранимых данных

Фотографии связаны с получаемыми данными в первую очередь в соответствии со временем получения, поскольку создание фотографии для каждого полученного пакета данных означало бы достаточно большой объем памяти, требуемый под них. Связь фотографий и данных представлена на фрагменте концептуальной модели (рисунок 4).

Полная схема концептуальной модели находится в приложении 1, рис. 1.

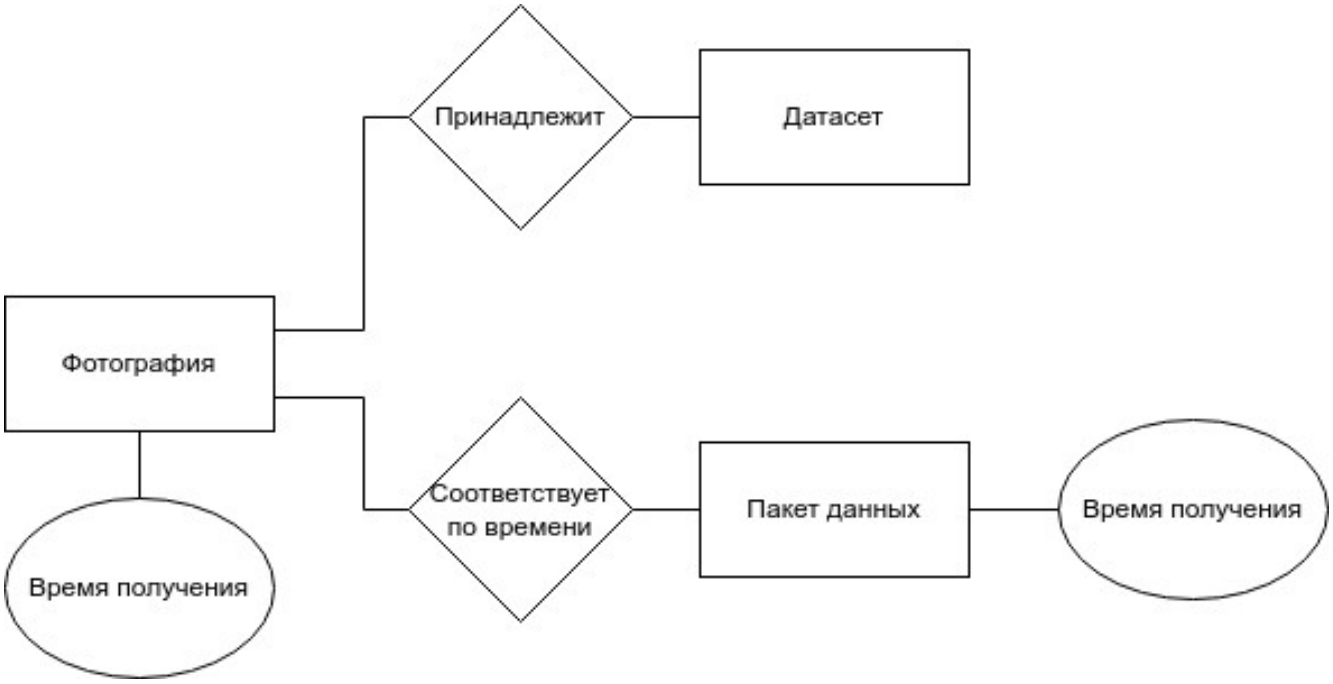

Рисунок 4 - Связь фотографий и данных

2.2. Логическая модель

В процессе анализа технического задания было определено, что база данных будет реляционной. Это означает, что сущности будут представлены в виде таблиц, а атрибуты — в качестве полей.

Рассмотрим основную сущность базы данных — датасет. Преобразуем её в таблицу, структура которой представлена на рисунке 5.

	experiment
PK, NN	**id**
NN	name
	description
FK	hardware_tx_id
FK	hardware_rx_id
	recv_handler
	preproc_handler
	config

Рисунок 5 - таблица датасета

Представлены поля имени (name) и описания (description), необходимые для различения датасетов между собой. Поля hardware_tx_id и hardware_rx_id указывают на тип используемого оборудования (передатчика и приёмника соответственно). Поля recv_handler и preproc_handler указывают на модули ИС, отвечающие за приём данных и их предобработку соответственно. Поле config хранит конфигурационный JSON-файл.

Типы используемого оборудования описаны в отдельной таблице, схема которой представлена на рисунке 6. Поля name и description, аналогично соответствующим полям в таблице experiment, задают имя оборудования и его описание. Поле antennas задает количество активных антенн у оборудования. Поле subcarriers — количество доступных поднесущих.

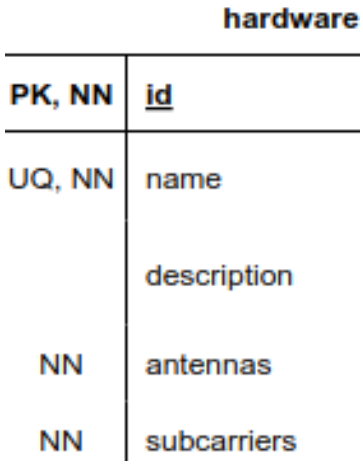

Рисунок 6 - таблица hardware

Таблицы, хранящие полученные данные, представлены на рисунке 7. Данные между собой связаны посредством идентификаторов (id). Поле timestamp — время получения пакета данных в формате Unix time. Marker — поле, соответствующее маркировке данных. num_sub — поле, обозначающее, какой поднесущей соответствует данная запись. Поля rx и tx хранят в себе значения того, на какую и с какой антенны были переданы данные о канале связи. Поля real_part и imag_part хранят в себе действительную и мнимую части комплексной амплитуды. Поля amplitude и phase хранят в себе уже выделенные амплитуду и фазу.

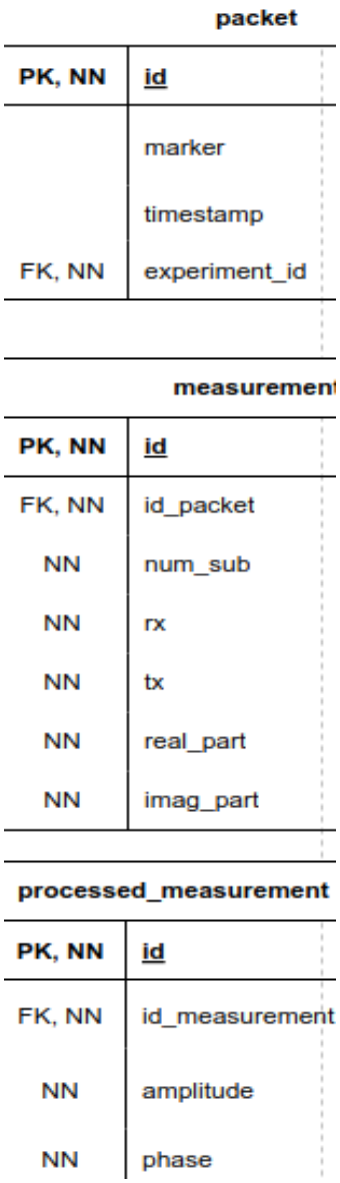

Рисунок 7 - таблицы с данными

Полная схема логической модели находится в приложении 1, рис. 2.

2.3. Физическая модель

Для создания базы данных будет использоваться движок базы данных SQLite. Это обусловлено рядом факторов:

1. Встраиваемость. SQLite встраивается напрямую в приложение, благодаря чему упрощается развертывание информационной системы.

2. Старые БД. Поскольку датасеты ранее также хранились в базах данных SQLite, то использование данного движка упростит перенос данных.

Таблицы представлены далее (рисунки 8-13).

hardware		
PK, NN	**id**	**int**
UQ, NN	name	text
	description	text
NN	antennas	int
NN	subcarriers	int

Рисунок 8 - таблица hardware

experiment		
PK, NN	**id**	**int**
NN	name	text
	description	text
FK	hardware_tx_id	int
FK	hardware_rx_id	int
	recv_handler	text
	preproc_handler	text
	config	text

Рисунок 9 - таблица experiment

image		
PK, NN	**id**	**int**
FK, NN	experiment_id	int
NN	image_path	text
NN	timestamp	int

Рисунок 10 - таблица image

packet		
PK, NN	**id**	**int**
	marker	text
	timestamp	int
FK, NN	experiment_id	int

Рисунок 11 - таблица packet

measurement		
PK, NN	**id**	**int**
FK, NN	id_packet	int
NN	num_sub	int
NN	rx	int
NN	tx	int
NN	real_part	int
NN	imag_part	int

Рисунок 12 - таблица measurement

processed_measurement		
PK, NN	**id**	**int**
FK, NN	id_measurement	int
NN	amplitude	real
NN	phase	real

Рисунок 13 - таблица processed_measurement

Полная схема физической модели находится в приложении 1, рис. 3.

3. Разработка и реализация АИС

3.1. Руководство программиста

Разработки приложения велась на языке программирования C++ в среде разработки Code::Blocks.

Использованный стек:

1. Gtkmm4 — для построения графического пользовательского интерфейса.

2. SQLiteCpp — биндинг SQLite для C++.

3. SFML Network — для разработки компонента, отвечающего за принятие данных по локальной сети.

4. OpenCV — для захвата фотографий с веб-камеры.

5. OpenGL — для ранее разработанного компонента вывода графиков.

Для разработки графического пользовательского интерфейса использовалось приложение Cambalache. Оно позволяет графически разрабатывать интерфейсы, основанные на Gtk. Затем интерфейс сохраняется в формате XML-файла, после чего он загружается в основном приложении.

Большинство взаимодействий с базой данных осуществляется через объекты-списки, разработанные с применением паттерна проектирования «Одиночка» («Singleton»).

Для расчета амплитуды и фазы сигнала используется следующий триггер:

```
CREATE TRIGGER process_measurement
AFTER INSERT
ON measurement
BEGIN
INSERT INTO processed_measurement (id_measurement, amplitude, phase)
VALUES (NEW.id, SQRT(NEW.real_part * NEW.real_part + NEW.imag_part * NEW.imag_part), atan2(NEW.imag_part, NEW.real_part));
END;
```

Объект Типа DB_Handler, отвечающий за подключение к базе данных, при старте приложения проверяет её существование, и в случае её отсутствия, создает новую с помощью SQL-запроса.

Особенностью модели базы данных является хранение изображений не в виде полей типа BLOB, а как путей до файла. Это обусловлено низкой скоростью работы SQLite с объектами типа BLOB достаточно большого размера. Это отображено на таблице 1. Значение в каждой ячейке — отношение скорости работы при хранении объекта в базе данных к скорости работы при хранении объекта в файловой системе [2].

Database Page Size	BLOB size						
	10k	20k	50k	100k	200k	500k	1m
1024	1.535	1.020	0.608	0.456	0.330	0.247	0.233
2048	2.004	1.437	0.870	0.636	0.483	0.372	0.340
4096	2.261	1.886	1.173	0.890	0.701	0.526	0.487
8192	2.240	1.866	1.334	1.035	0.830	0.625	0.720
16384	2.439	1.757	1.292	1.023	0.829	0.820	0.598
32768	1.878	1.843	1.296	0.981	0.976	0.675	0.613
65536	1.256	1.255	1.339	0.983	0.769	0.687	0.609

Таблица 1 - скорость работы SQLite с BLOB

Однако, поскольку в базе данных хранятся только пути до изображений, то удаление изображений требуется обрабатывать особым образом. Для этого был использован такой механизм SQLite, как update_hook, который позволяет задать функцию обратного вызова при обновлении, удалении или вставке в таблицу. Таким образом, при удалении записи об изображении вызывается функция приложения, удаляющая так же и файл изображения.

Для фильтрации датасетов используется используется составной запрос, что обусловлено опциональностью фильтров. При включении всех фильтров запрос принимает следующий вид:

```
SELECT id, name, description, hardware_tx_id, hardware_rx_id, config,
recv_handler, preproc_handler FROM experiment
WHERE
```

```
    id IN (SELECT experiment.id FROM experiment JOIN packet ON
packet.experiment_id = experiment.id WHERE timestamp > @min_time) AND
    id IN (SELECT experiment.id FROM experiment JOIN packet ON
packet.experiment_id = experiment.id WHERE timestamp < @max_time) AND
    hardware_tx_id = @transIdx AND
    hardware_rx_id = @recvIdx
```

Для экспорта датасета в JSON применяется следующий запрос:

```
    SELECT measurement.num_sub, measurement.rx, measurement.tx, amplitude,
phase, measurement.real_part, measurement.imag_part, packet.id
    FROM processed_measurement
    INNER JOIN measurement ON processed_measurement.id_measurement =
measurement.id
    INNER JOIN packet ON measurement.id_packet = packet.id
    INNER JOIN experiment ON packet.experiment_id = experiment.id
    WHERE experiment.id = @exp_id AND packet.marker LIKE @marker
    ORDER BY packet.id, measurement.num_sub
```

3.2. Руководство пользователя

При запуске программы открывается основное окно (рисунок 14). В данном окне производится фильтрация доступных датасетов (экспериментов), выбор активного датасета, отслеживание данных датасета.

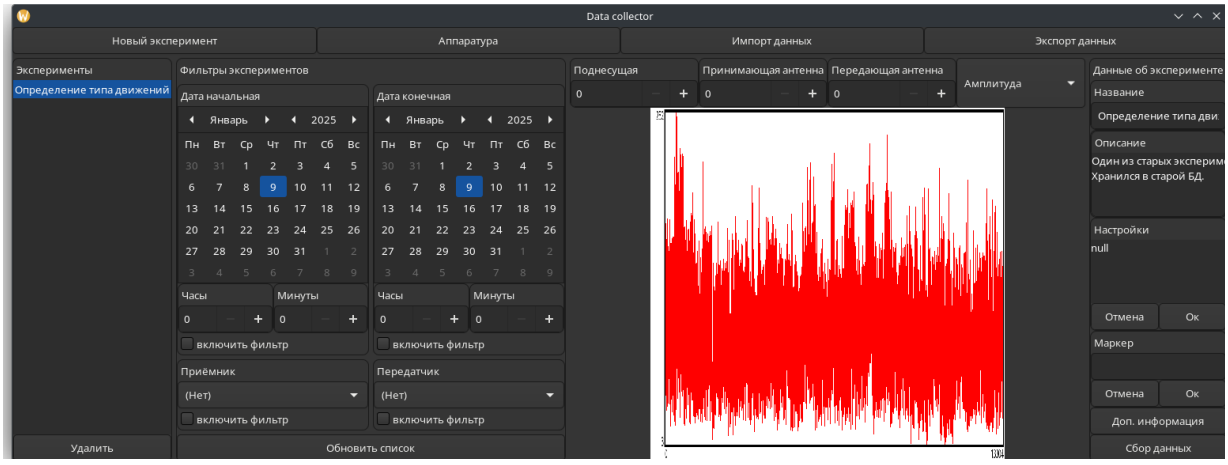

Рисунок 14 - основное окно

Из данного окна можно получить доступ ко всем остальным функциям: Созданию нового датасета (рисунок 15), управлению списком аппаратуры

(рисунок 16), импорту данных (рисунок 17), экспорту данных (рисунок 18), дополнительным данным о датасете (рисунок 19). Сбор данных и фотосъемка осуществляется при активной кнопке «Сбор данных». Для настройки модулей сбора данных используется текстовое поле «Настройки», в которое вводятся настройки в формате JSON.

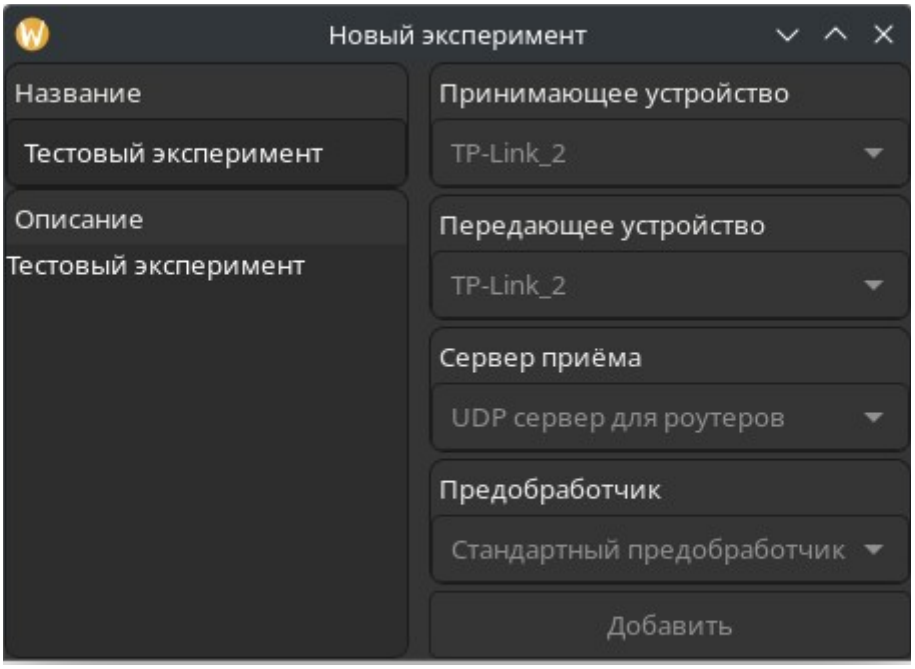

Рисунок 15 - Окно создания датасета

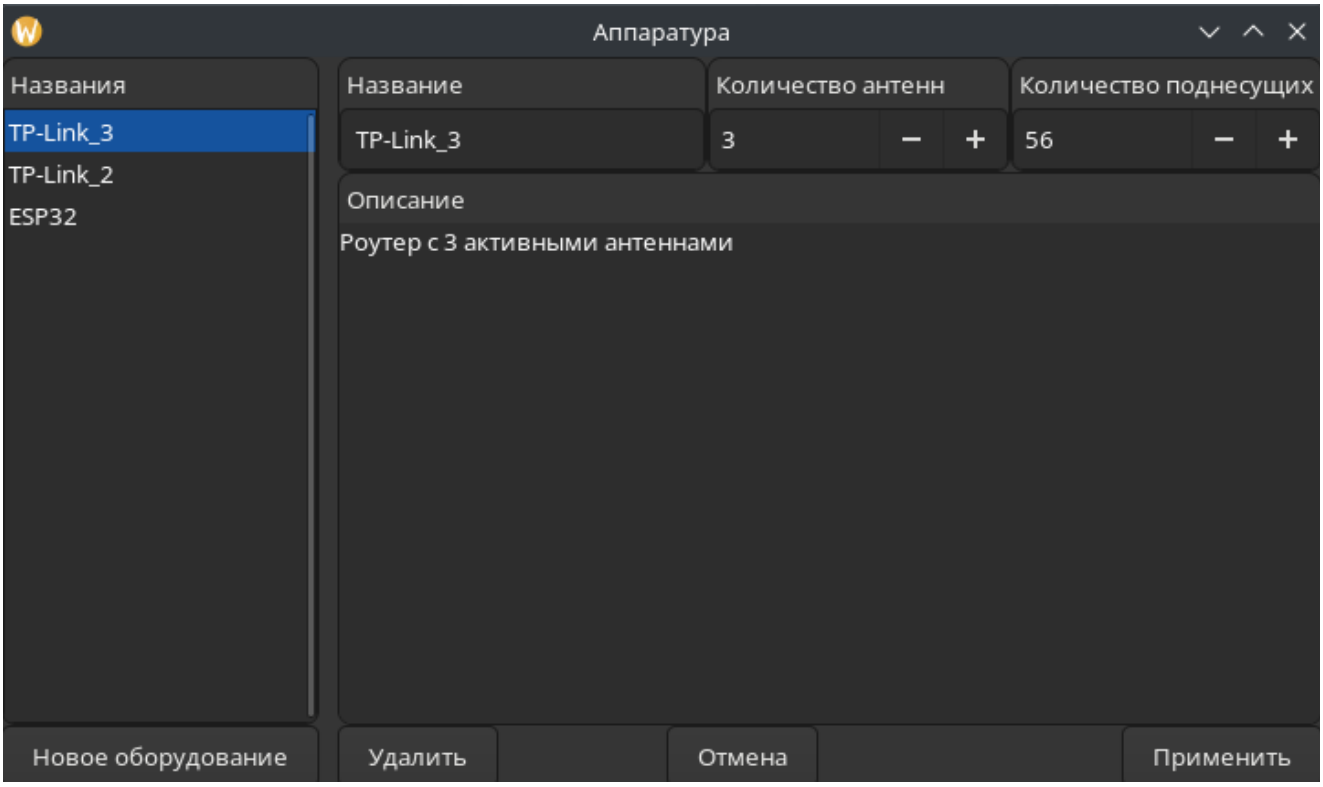

Рисунок 16 - окно управления аппаратурой

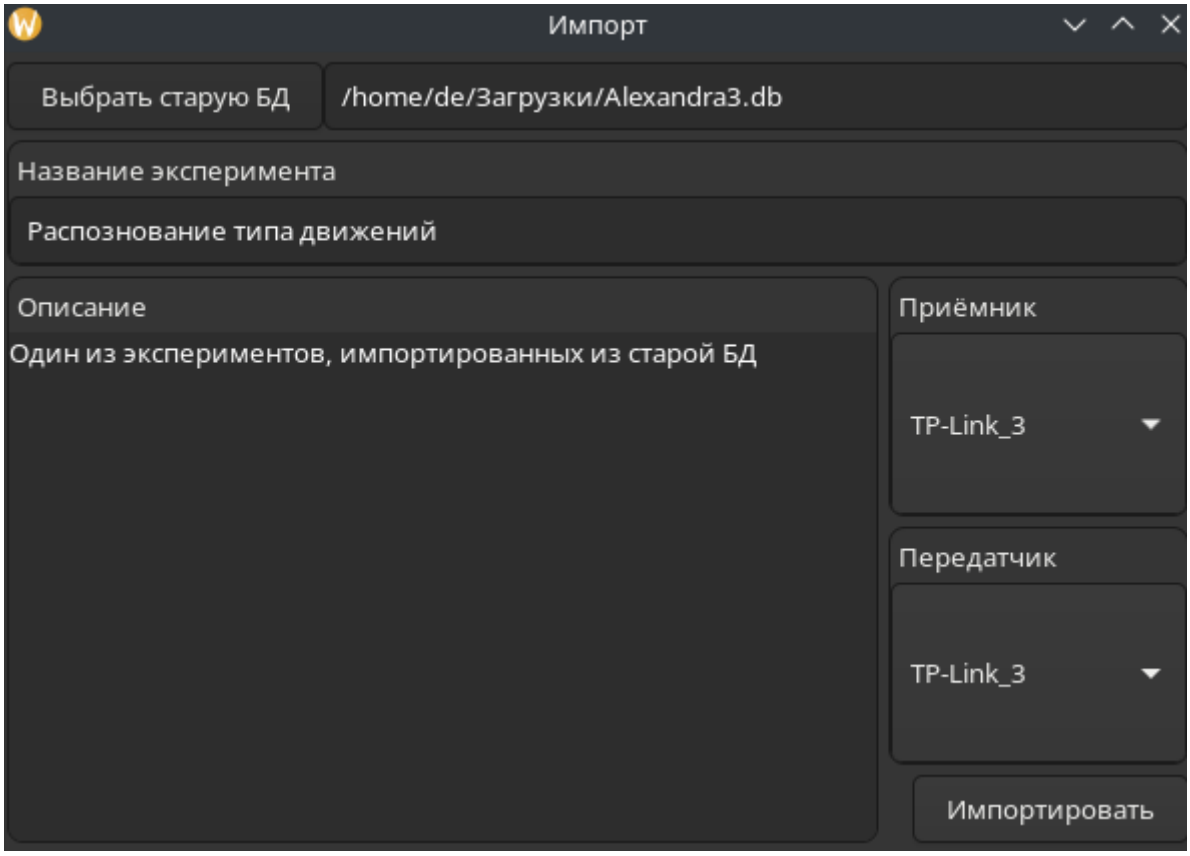

Рисунок 17 - окно импорта данных

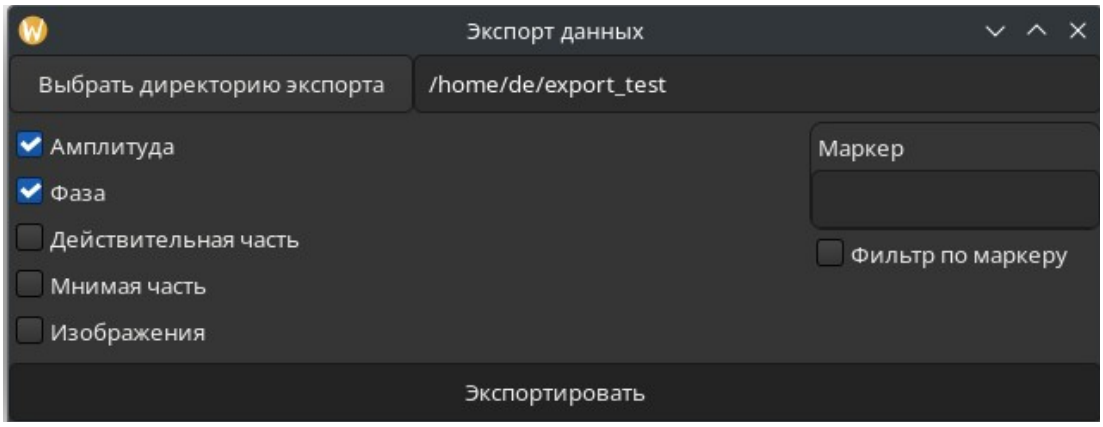

Рисунок 18 - окно экспорта данных

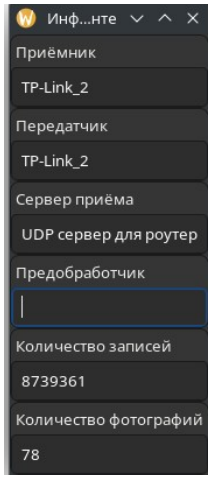

Рисунок 19 - окно дополнительной информации

4. Тестирование АИС

В ходе и по завершение разработки любой системы необходимо её тестирование в различных сценариях. Это же относится и к информационным системам. Программа должна корректно работать в случае любых действий пользователя, а в случае исключительных ситуаций недопустимо аварийное завершение работы программы. Это достигается путём повсеместной проверки ввода пользователя и обширного использования конструкции try-catch с последующим выводом текста исключений в поток stderr.

На рисунках 20-22 представлено тестирование функции фильтрации датасетов. На рисунке 23 представлен тест экспорта данных.

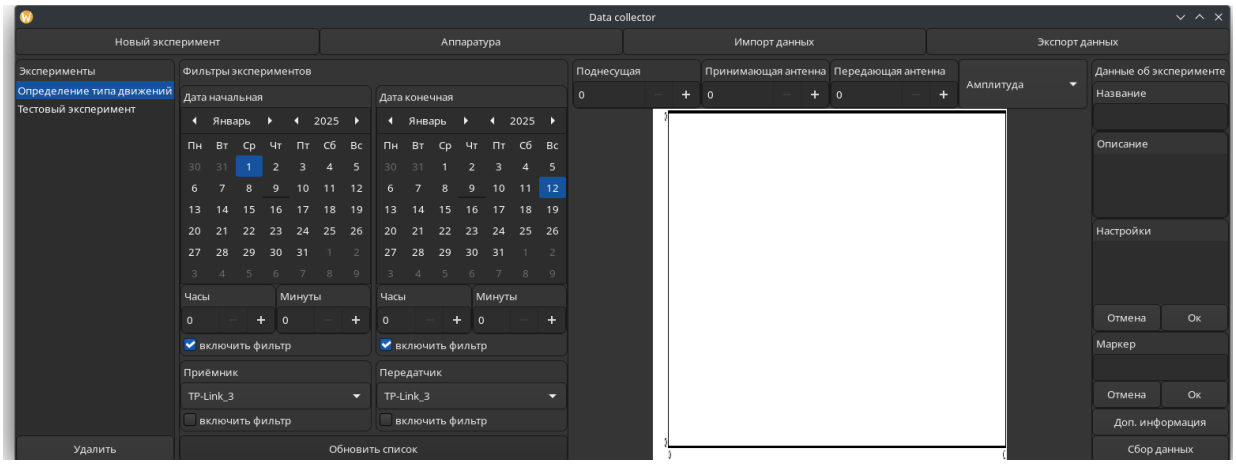

Рисунок 20 - Тест фильтрации для всех допустимых датасетов

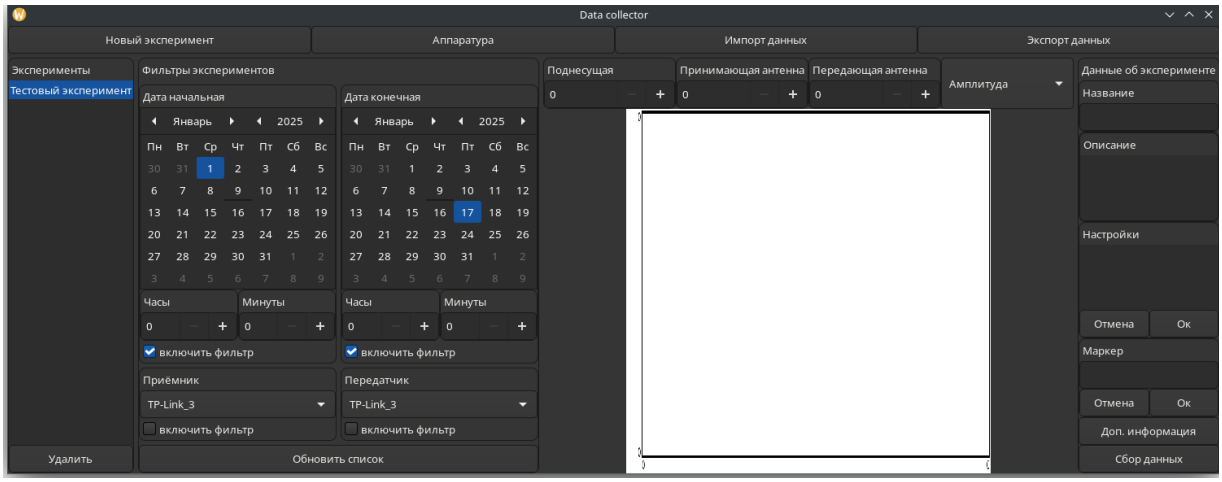

Рисунок 21 - Тест фильтрации для одного допустимого датасета

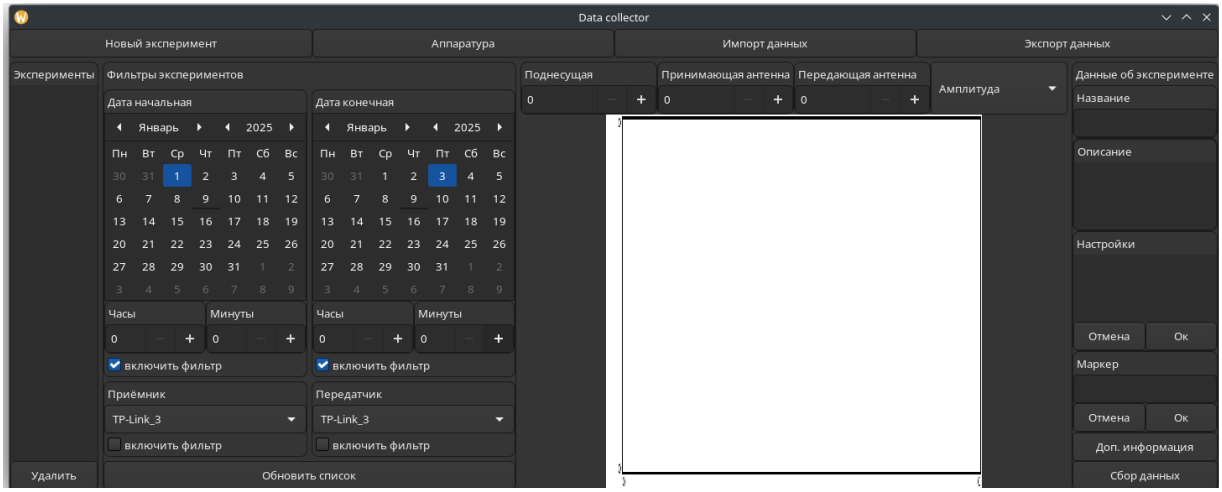

Рисунок 22 - Тест фильтрации для не одного допустимого датасета

```
de@de-laptop:~/export_test$ tree
.
├── ampls
│   ├── 0_0.json
│   ├── 0_1.json
│   ├── 1_0.json
│   └── 1_1.json
├── phase
│   ├── 0_0.json
│   ├── 0_1.json
│   ├── 1_0.json
│   └── 1_1.json
└── photos
    ├── _1736372813.jpg
    ├── _1736372814.jpg
    ├── _1736372815.jpg
    ├── _1736372992.jpg
    ├── _1736373005.jpg
    ├── _1736373006.jpg
    ├── _1736373007.jpg
    ├── _1736373008.jpg
    ├── _1736373009.jpg
    ├── _1736373010.jpg
    ├── _1736373011.jpg
    ├── _1736373012.jpg
    ├── _1736373013.jpg
    ├── _1736373014.jpg
    ├── _1736373015.jpg
    ├── _1736373016.jpg
    └── _1736373040.jpg
```

Рисунок 23 - Тест экспорта данных

Заключение

В ходе данной курсовой работы была разработана автоматизированная информационная система сбора канальных матриц. Для достижения поставленной цели были разработаны структура базы данных и приложение для работы с ней.

Список используемой литературы

1. Yongsen Ma, Gang Zhou, and Shuangquan Wang. WiFi Sensing with Channel State Information: A Survey // ACM Comput. Surv., 2019, V. 52, I. 3, No. 46. — 4 P. DOI:10.1145/3310194

2. Документация SQLite: [Электронный ресурс]. https://www.sqlite.org/intern-v-extern-blob.html (Дата обращения: 07.11.2024).

Приложение 1. Модели данных

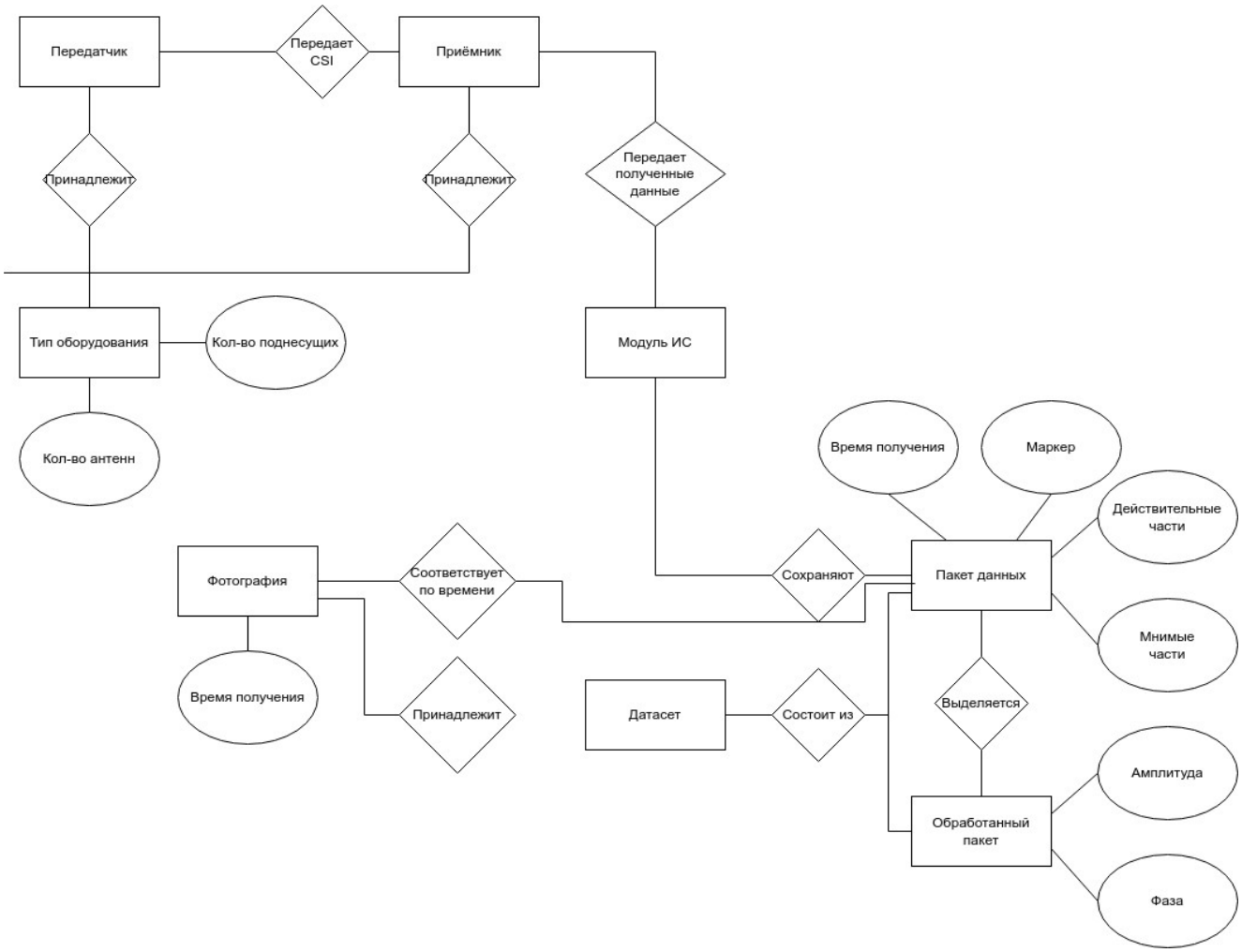

Рисунок 1 - Концептуальная модель

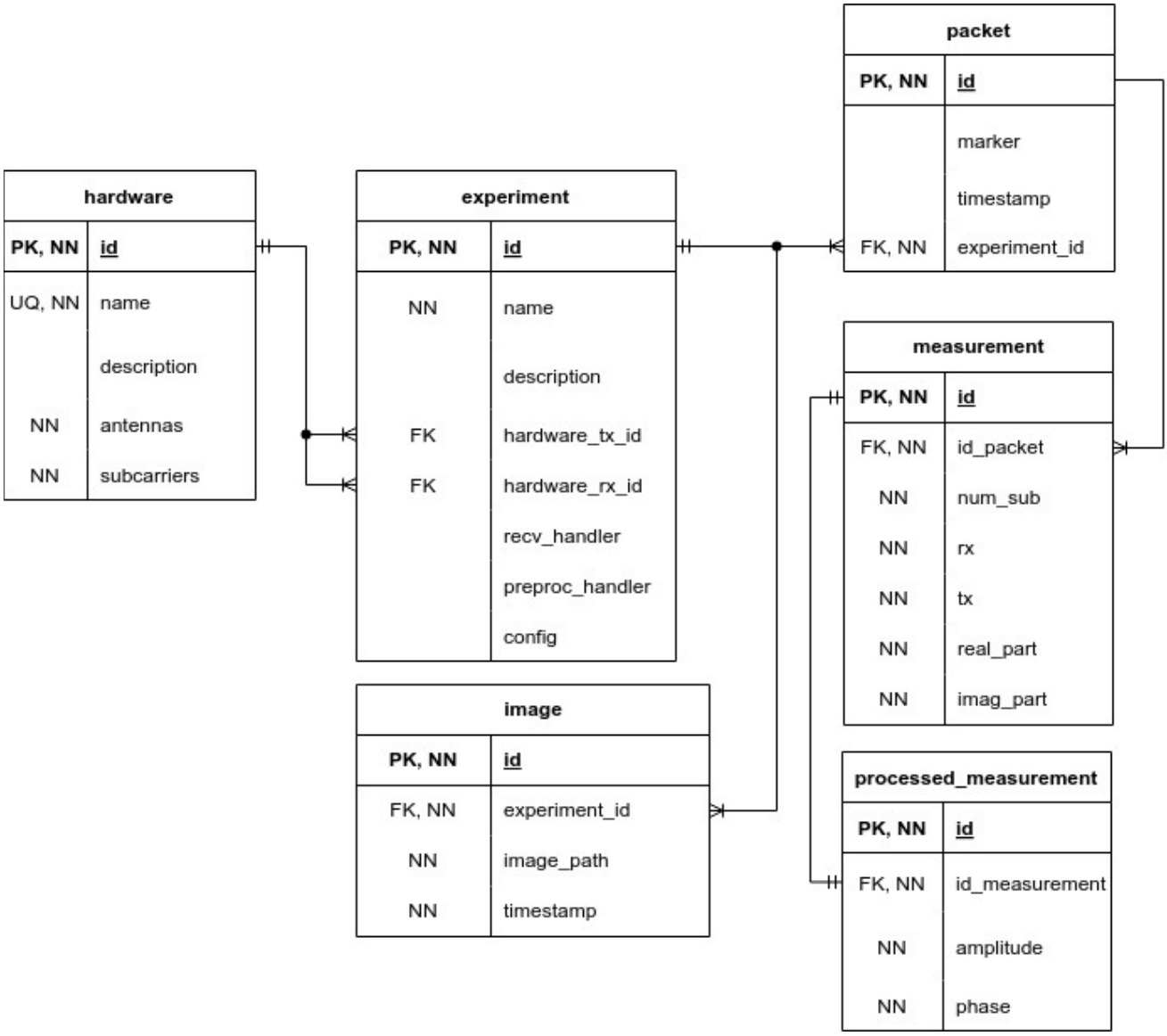

Рисунок 2 - Логическая модель

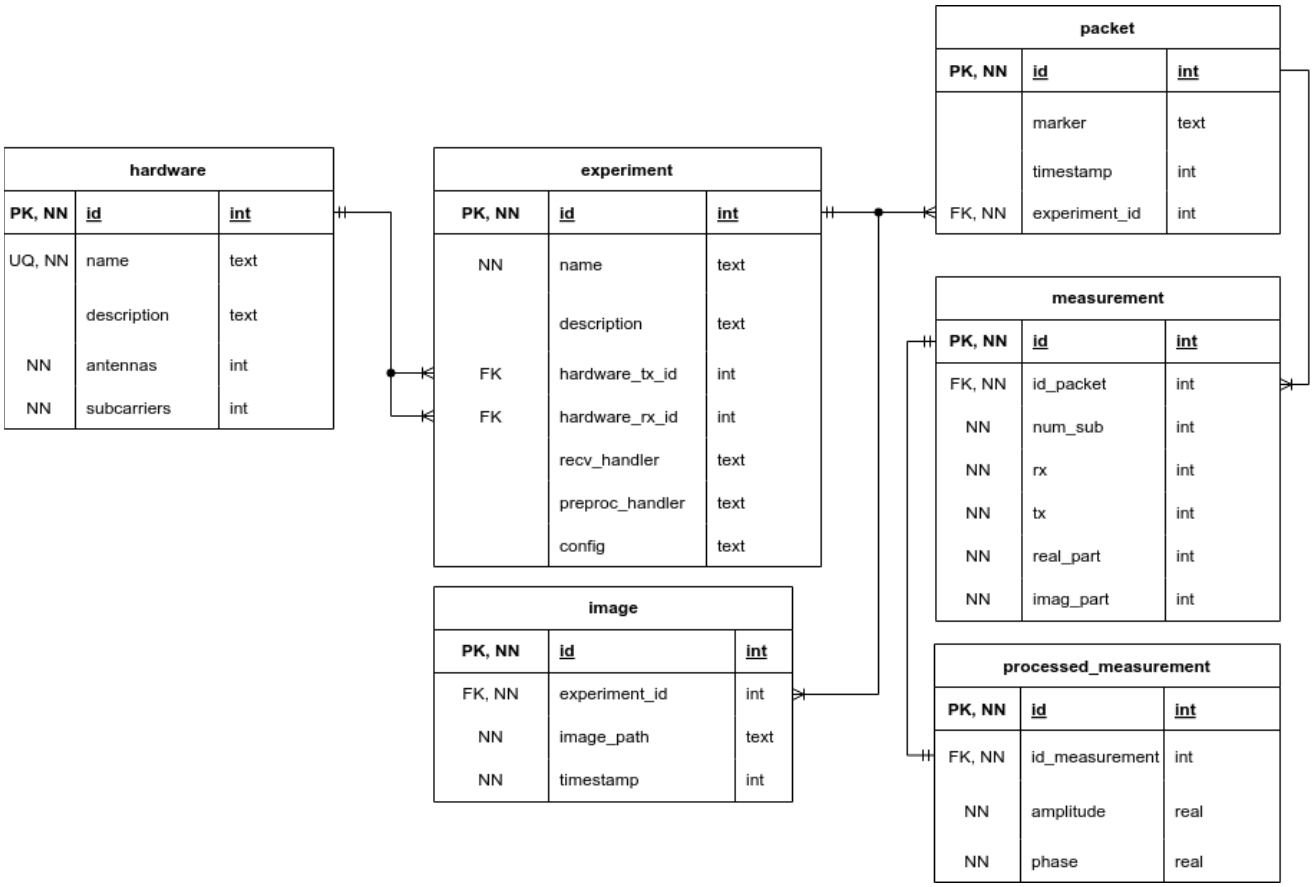

Рисунок 3 - Физическая модель

Приложение 2. Текст программы

Текст программы выложен на платформу GitHub и доступен вместе со всеми материалами по ссылке https://github.com/Umaler/DB_Course