Разработка графического интерфейса создания дашбордов и визуализация событий в эксперименте BAIKAL-GVD

Студент : Сафронова М.А.
Научный руководитель: д.ф.-м.н. Наумов Д.В.

Цель работы

- обзор подводного нейтринного телескопа Байкал;
- ознакомление с пакетом NTSim;
- разработка удобных сервисов для работы с детектором;

Введение

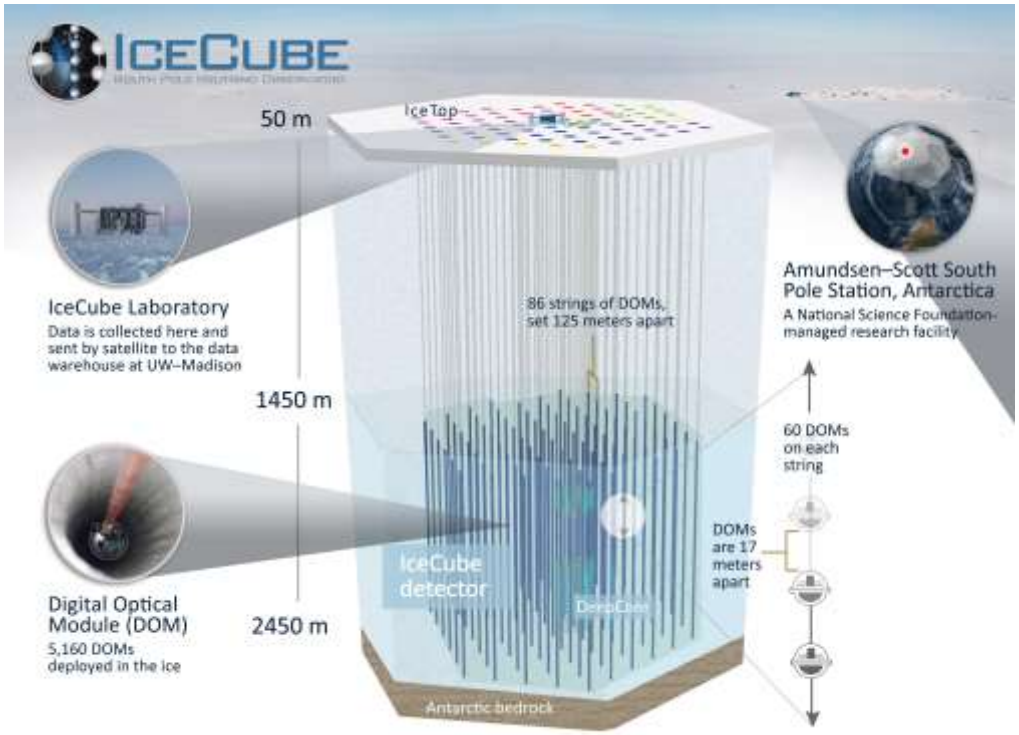


Рис.1 Эксперимент IceCube.

Идея создания гигатонного нейтринного телескопа в Северном полушарии появилась после обнаружения астрофизических нейтрино энергией около 1000 ТэВ детектором IceCube (расположен в Южном полушарии)

В результате создана глобальная нейтринная сеть обнаружения нейтрино со всей небесной сферы

Основная цель эксперимента - регистрация потоков астрофизических нейтрино от источников, расположенных в Северном полушарии Земли

Общая схема установки

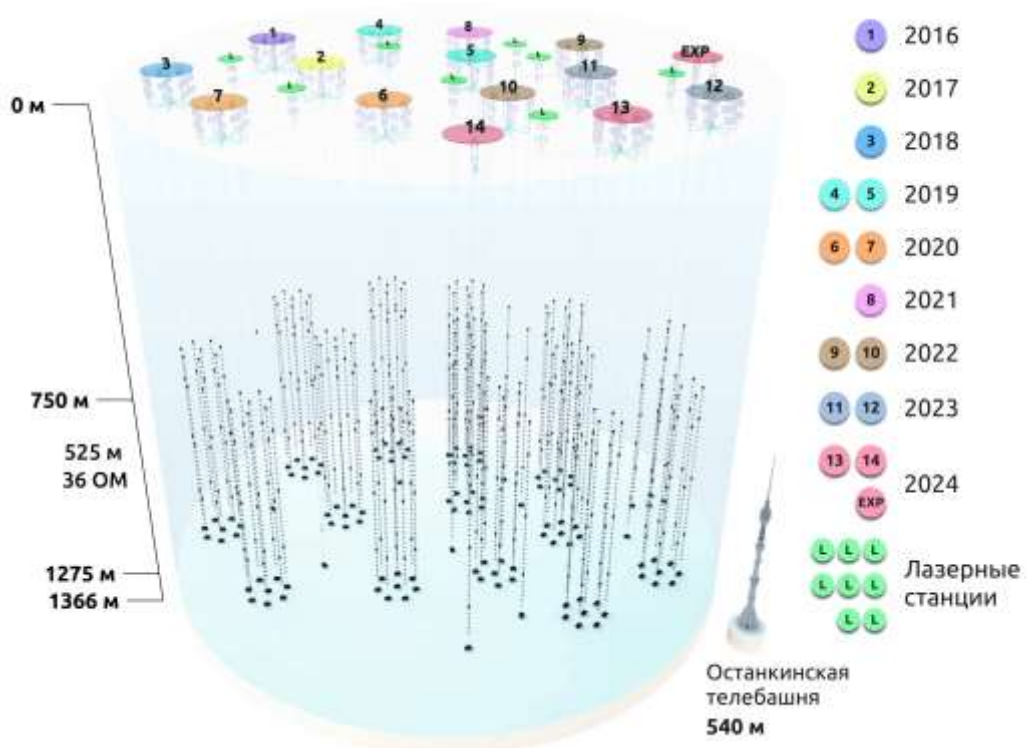


Рис3. Общая схема нейтринного телескопа Байкал

Основной структурной единицей установки являются кластеры. Каждый кластер представляет собой набор из восьми километровых струн, расположенных в виде семиугольника

Кластеры управляются береговой станцией и работают независимо друг от друга. Возможно проводить многокластерный и однокластерный анализ

Каждая крепится с одной ко дну озера, а с другой к буям на поверхности, и содержит 36 оптических модулей, расположенных на расстоянии 15 м друг от друга начиная с глубины 90м. Таким образом, они покрывают глубину около 525м

В настоящее время установка состоит из 14 кластеров

Оптические модули

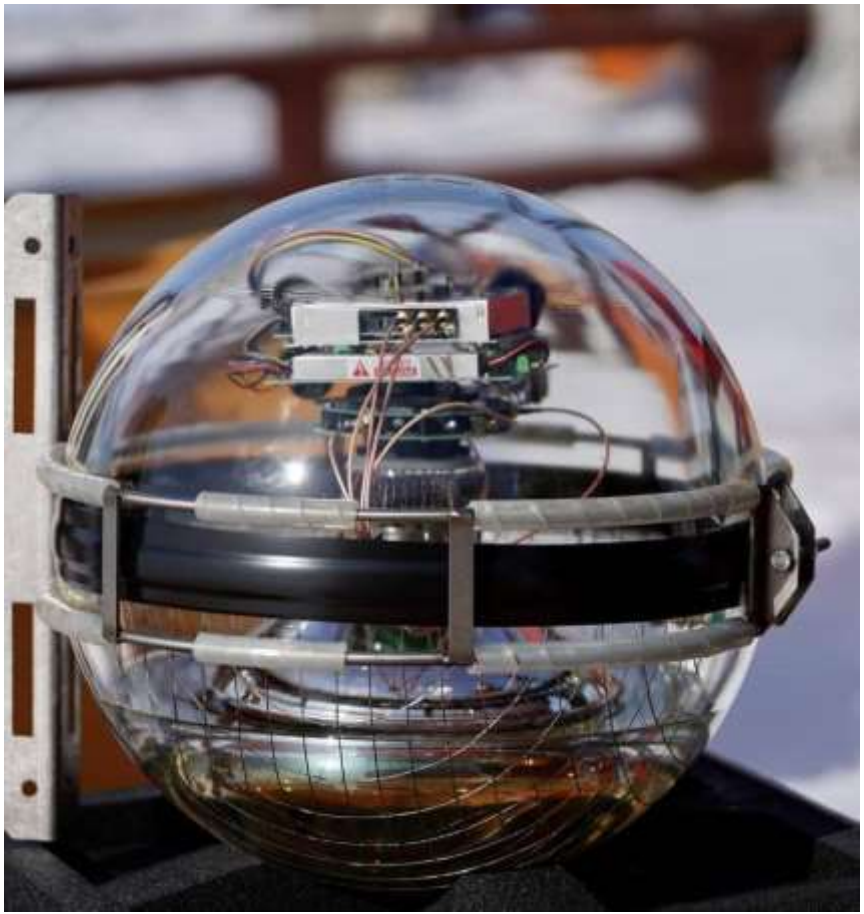


Рис4. Оптичский модуль Байкальского нейтринного телескопа

Оптический модуль — основной элемент установки. Нейтрино не регистрируются непосредственно, при взаимодействии с веществом они образуют заряженные частицы. Одна из наиболее распространённых реакций:

$$p + \nu_\mu \rightarrow \mu + n \quad (1)$$

В силу высоких энергий нейтрино скорости мюонов превосходят скорость света в воде. В результате возникает черенковское излучение, которое регистрируется ФЭУ, расположенных внутри герметичных и устойчивых к высокому давлению стеклянных сфер

Помимо ФЭУ оптические модули содержат большое количество датчиков (давления, температуры, влажности воздуха и тд)

Разработка веб-сервисов.

Детектор BAIKAL-GVD собирает данные пока идет время сеанса работы - рана. Для взаимодействия со всем процессом существует сервис, который позволяет клиенту взаимодействовать с раном через браузер. Задачей данной работы было создание удобного программного обеспечения.

Программное обеспечение, использованное в работе:

1) Flask
2) socket

Разработка веб-сервисов.

```python
import socket
from flask import Flask, jsonify, render_template


def run_service():
    HOST = '127.0.0.1'
    PORT = 5002

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT))
        s.listen()
        print(f"Socket service запущен на {HOST}:{PORT}")

        while True:
            conn, addr = s.accept()
            with conn:
                print('Socket client подключён:', addr)
                data_to_send =  "HTTP/1.1 200 OK\r\nContent-Type:
                conn.sendall(data_to_send.encode())
                conn.close()




if __name__ == '__main__':
    run_service()
```

Рис.5 Скрипт создания первого сервиса.

Первый сервис создан с помощью пакета socket. Создается TCP - сокет, который привязывается к порту, а далее ждет подключения. При появлении подключения сокет отправляет сообщение другому сервису.

Разработка веб-сервисов.

```
69  from flask import Flask, render_template_string
70  import socket
71  import threading
72  from queue import Queue
73
74  app = Flask(__name__)
75
76  messages = Queue()
77
78  HTML_TEMPLATE = """
79  <!DOCTYPE html>
80  <html>
81  <head>
82      <title>Socket Service</title>
83      <style>
84          body { font-family: Arial, sans-serif; margin: 20px; }
85          .message { padding: 10px; margin: 5px; background: #f0f0f0; border-radius: 5px; }
86      </style>
87  </head>
88  <body>
89      <h1>Полученные сообщения:</h1>
90      <div id="messages">
91          {% for msg in messages %}
92          <div class="message">{{ msg }}</div>
93          {% endfor %}
94      </div>
95  </body>
96  </html>
97  """
```

Рис. 6 Скрипт создания второго сервиса.

С помощью html разметки создано оформление веб-страницы.

```python
def socket_service():

    host = '127.0.0.1'
    port = 5002

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((host, port))
        print(f"Сокет-сервер запущен на {host}:{port}")

        response = b""

        while True:
            chunk = s.recv(4096)
            if not chunk:
                break
            response += chunk

        headers_end = response.find(b"\r\n\r\n")
        if headers_end == -1:
            raise ValueError("Invalid HTTP response")

        raw_headers = response[:headers_end].decode()
        body = response[headers_end + 4:].decode()

        if body:
            messages.put(body)

        status_line = raw_headers.split("\r\n")[0]
        status_code = int(status_line.split()[1])

        headers = {}
        for line in raw_headers.split("\r\n")[1:]:
            if ": " in line:
                key, value = line.split(": ", 1)
                headers[key] = value

        return {
            "status_code": status_code,
            "headers": headers,
            "body": body,
        }
```

Рис. 7 Скрипт создания второго сервиса.

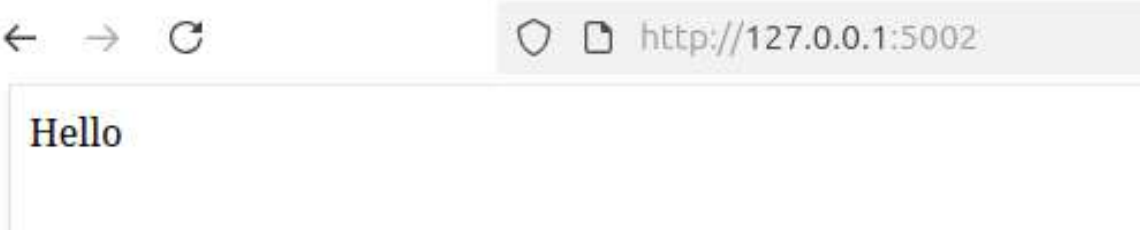
Для второго сервиса написана функция взаимодействия с первым сервисом. Функция подключается к первому сервису по определенному порту, после чего получает сообщение с помощью метода recv().

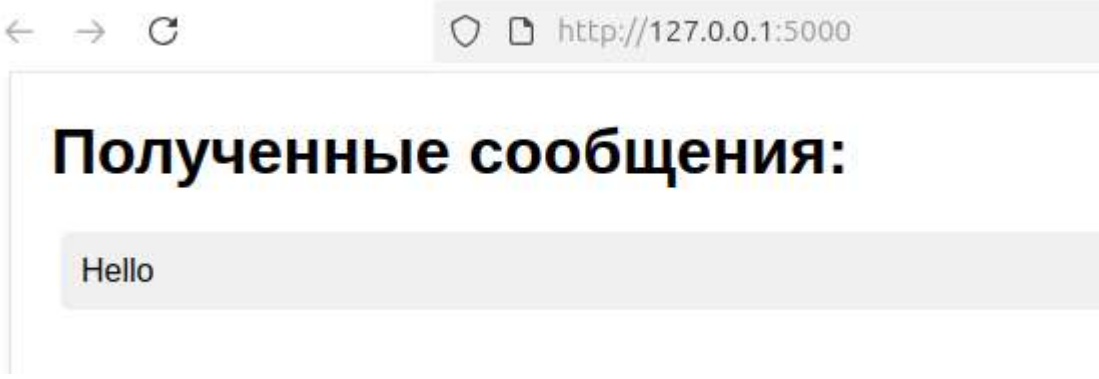
В случае успешного получения данных, recv() возвращает байтовый объект, содержащий полученные данные.

```python
@app.route('/')
def index():
    current_messages = list(messages.queue)
    return render_template_string(HTML_TEMPLATE, messages=current_messages)

if __name__ == '__main__':

    socket_thread = threading.Thread(target=socket_service)
    socket_thread.daemon = True
    socket_thread.start()


    app.run(port=5000, debug=True)
```

Рис. 8 Скрипт создания второго сервиса.

Пример работы сервисов





4. Заключение

- В ходе работы было проведено ознакомление с основными нейтринными экспериментами, также были изучены принципы работы нейтринных детекторов, таких как IceCube, KM3NeT, BAIKAL-GVD.
- В рамках данной научной работы был изучен и использован пакет NTSim.
- Также в данной работе реализованы два веб-сервиса, взаимодействие между которыми организованно с помощью сокетов.