

Пояснительная записка к работе

по теме

Разработка программного обеспечения для контроля местоположения и режима движения телеуправляемого необитаемого надводного аппарата

Санкт-Петербург

## Оглавление

Введение.....	2
Цель работы. ....	2
Задачи работы.....	2
Требования к выполнению работы.....	2
Выполнение требований к работе.....	2
Техническое задание.....	3
Протокол информационного обмена.....	4
1.    Информация о передаваемых данных.....	4
2.    Последовательность передачи данных.....	4
3.    Идентификатор пакета.....	5
Описание библиотек.....	6
Возможности программы.....	7
Листинг программы.....	8
Результат работы программы.....	10
Заключение.....	13

## **Введение.**

### **Цель работы.**

Разработка программного обеспечения (ПО) для контроля местоположения и режима движения телеуправляемого необитаемого надводного аппарата (ТННА).

Срок разработки:

### **Задачи работы.**

1. Разработка и согласование протокола информационного обмена (ПИО) используемого для получения информации от бортовой вычислительной системы (БВС) ТННА.
2. Разработка программного обеспечения для визуализации данных, поступающих от БВС ТННА.

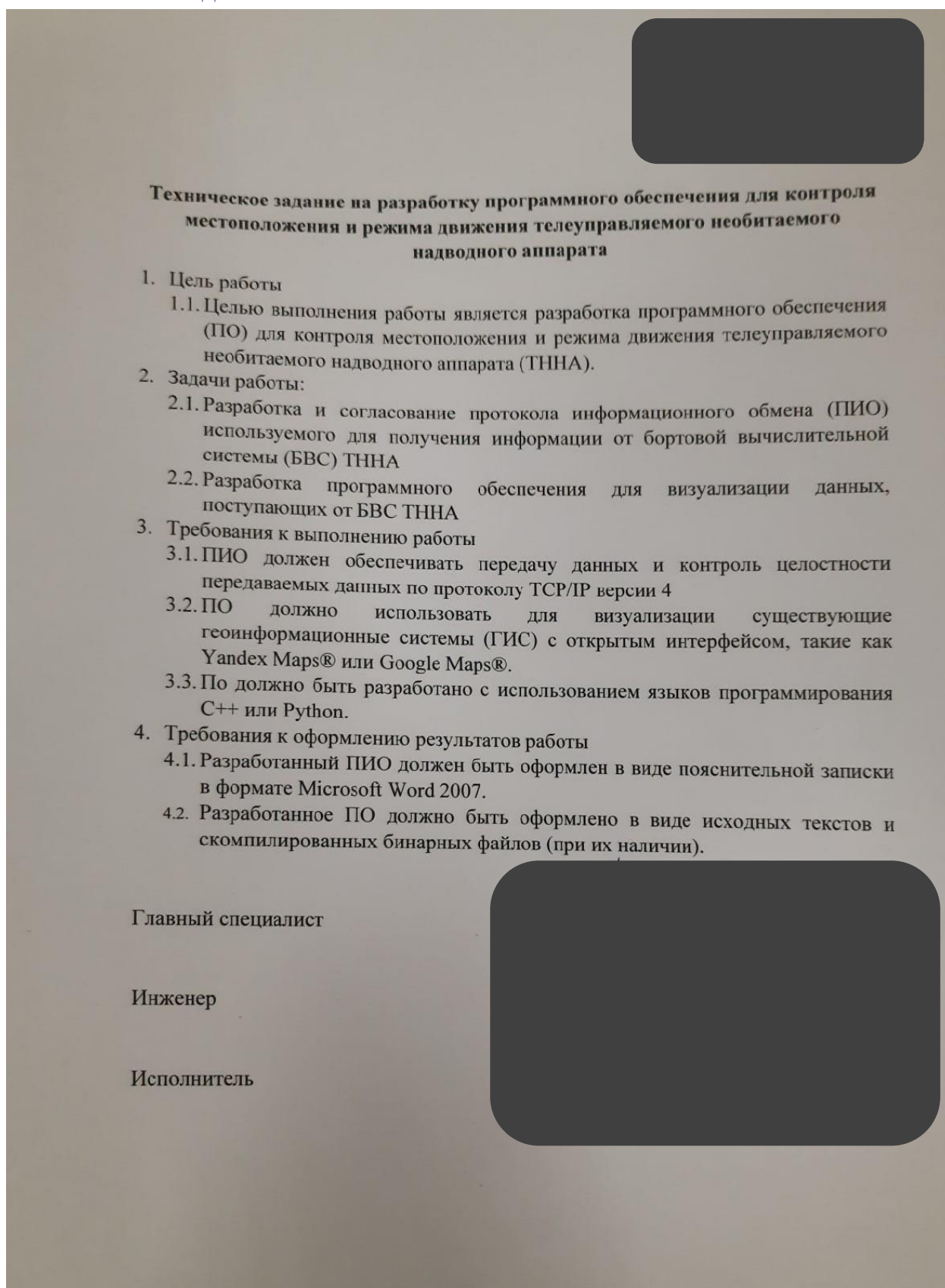
### **Требования к выполнению работы.**

1. ПИО должен обеспечивать передачу данных и контроль целостности передаваемых данных по протоколу TCP/IP версии 4.
2. ПО должно использовать для визуализации существующие геоинформационные системы (ГИС) с открытым интерфейсом, такие как Yandex Maps или Google Maps.
3. ПО должно быть разработано с использованием языков программирования C++ или Python.

### **Выполнение требований к работе.**

1. ПО осуществляет проверку правильности переданных пакетов данных с помощью алгоритма CRC-32B. Этот алгоритм использует полином 0x04C11DB7.
2. ПО использует для визуализации библиотеку folium.
3. ПО разработано с использованием языка Python 3. IDE – PyCharm.

## Техническое задание.



**Техническое задание на разработку программного обеспечения для контроля местоположения и режима движения телеуправляемого необитаемого надводного аппарата**

1. Цель работы
  - 1.1. Целью выполнения работы является разработка программного обеспечения (ПО) для контроля местоположения и режима движения телеуправляемого необитаемого надводного аппарата (ТННА).
2. Задачи работы:
  - 2.1. Разработка и согласование протокола информационного обмена (ПИО) используемого для получения информации от бортовой вычислительной системы (БВС) ТННА
  - 2.2. Разработка программного обеспечения для визуализации данных, поступающих от БВС ТННА
3. Требования к выполнению работы
  - 3.1. ПИО должен обеспечивать передачу данных и контроль целостности передаваемых данных по протоколу TCP/IP версии 4
  - 3.2. ПО должно использовать для визуализации существующие геоинформационные системы (ГИС) с открытым интерфейсом, такие как Yandex Maps® или Google Maps®.
  - 3.3. ПО должно быть разработано с использованием языков программирования C++ или Python.
4. Требования к оформлению результатов работы
  - 4.1. Разработанный ПИО должен быть оформлен в виде пояснительной записки в формате Microsoft Word 2007.
  - 4.2. Разработанное ПО должно быть оформлено в виде исходных текстов и скомпилированных бинарных файлов (при их наличии).

Главный специалист

Инженер

Исполнитель

Рис. 1

Техническое задание

## Протокол информационного обмена.

### 1. Информация о передаваемых данных.

Значения элементов пакета:

{:2f}{:+3.6f}{:+3.6f}{:2.1f}{:3.2f}{:2f}{:2f}{:4f}{:2.2f}{:08x}

Описание значений элементов пакета в порядке их передачи:

1. {:2f} – идентификатор пакета, число 42, будет одинаково для всех передаваемых пакетов.
2. {:+3.6f} – широта.  
Диапазон: От -180.000000 до +180.000000
3. {:+3.6f} – долгота.  
Диапазон: От -180.000000 до +180.000000
4. {:2.1f} – скорость движения катамарана.  
Диапазон: От 00.0 до 99.9
5. {:3.2f} – направление движения катамарана.  
Диапазон: От 0.00 до 359.99
6. {:2f} – день передачи пакета.  
Диапазон: От 0 до 31
7. {:2f} – месяц передачи пакета.  
Диапазон: От 0 до 12
8. {:4f} – год передачи пакета.  
Диапазон: От 2021 до 2100
9. {:2.2f} – время передачи пакета.  
Диапазон: От 00.00 до 11.59
10. {:08x} – контрольная сумма CRC32B.

Алгоритм проверки правильности переданных данных: CRC32B

Полином: 0x04C11DB7

Комментарий: при записи делимого для вычисления контрольной суммы (КС) записывается вся переданная строка символов без самой контрольной суммы.

Контрольная сумма передается в виде числа в шестнадцатеричной системе счисления.

### 2. Последовательность передачи данных.

Последовательность и вид передачи данных:

{:2f}{:+3.6f}{:+3.6f}{:2.1f}{:3.2f}{:2f}{:2f}{:4f}{:2.2f}{:08x}

Пакеты передаются один за другим по дате и времени их формирования: самый «старый» передается первым.

### 3. Идентификатор пакета.

Идентификатором пакета будет служить число 42, оно одинаково для всех пакетов и стоит на первой позиции пакета.

## Описание библиотек.

Библиотека `folium` для подключения карт и операций над ними.

Функции, используемые из этой библиотеки:

1. функция, открывающая карту

```
map = folium.Map(location=[center_lat, center_lon], zoom_start=16)
```

2. функция, наносящая точки на карту

```
folium.Marker(location=[lat, lon],  
              popup=f"Скорость:{speed}км/ч \nНаправление:{way}°  
\nДата:\n {day}.{month}.{year} \nВремя:\n {time}",  
              icon=folium.Icon(color="red")).add_to(map)
```

3. функция, соединяющая линиями точки на карте

```
folium.PolyLine(locations=[(float(lat1), float(lon1)), (lat, lon)],  
                color="red", opacity=1).add_to(map)
```

4. функция, сохраняющая изменения на карте

```
map.save("map1.html")
```

Библиотека `socket` для реализации подключения к серверу и приема от него данных.

Функции, используемые из этой библиотеки:

1. функция, создающая сокет

```
client = socket.socket()
```

2. функция, устанавливающая соединение

```
client.connect(('94.19.19.131', 5117))
```

3. функция, принимающая данные с сервера

```
data = client.recv(1024)
```

Библиотека `zlib` для вычисления контрольной суммы алгоритма CRC32-B.

Функции, используемые из этой библиотеки:

1. функция, вычисляющая контрольную сумму алгоритма CRC-32B

```
hash = zlib.crc32(delimoe, hash)
```

### **Возможности программы.**

Карта открывается на html странице после начала работы программы. Обновляя страницу можно увидеть движение ТННА.

По мере отправки сервером пакетов данных на карту заносятся маркеры красного цвета по присланным координатам (см. рис. 2).

При нажатии на маркер всплывает окно с информацией над ним со следующими данными (см. рис. 3):

1. Скорость передвижения ТННА.
2. Направление движения ТННА.
3. Дата: день, месяц, год.
4. Время, в которое ТННА находился в конкретной точке.

Точки соединены красными линиями для того, чтобы было видно траекторию движения ТННА (см. рис. 2).

Карту можно увеличивать и уменьшать. Карта динамична. (см. рис. 4)

Во время работы программы на консоль выводятся пакеты с переданными данными и информация о совпадении или несовпадении контрольной суммы (см. рис. 5).

Для каждого пакета:

- 1 строка на консоли – данные пакета.
- 2 строка на консоли – информация о контрольной сумме.

После потери соединения с сервером программа завершает свою работу и выводит последней строкой на консоль количество неправильно переданных пакетов (см. рис. 6).



## Листинг программы.

```
import socket
import folium
import zlib

# создаем сокет
client = socket.socket()
# устанавливаем соединение
client.connect(('94.19.19.131', 5117))
# вспомогательное число
for_center_of_map = 1
# подсчет неправильных контрольных сумм
wrong_ks = 0

# основной цикл
while True:
    # принимаем данные
    data = client.recv(1024)
    # проверяем на пустую строку
    if data == b'':
        break
    # печатаем данные
    print(data)
    # делимое - присланная строка без присланной контрольной суммы
    delimoe = data[0:47]
    # декодировка строки
    data = data.decode('utf-8')

    # делим строку на подстроки
    ident = data[0:2]
    lat = data[2:13]
    lon = data[13:24]
    speed = data[24:28]
    way = data[28:34]
    day = data[34:36]
    month = data[36:38]
    year = data[38:42]
    time = data[42:47]
    crc_in = data[49:57]
    # делаем буквы контрольной суммы заглавными
    crc_in = crc_in.upper()

    # вычисляем свою контрольную сумму
    hash = 0
    hash = zlib.crc32(delimoe, hash)
    crc_solve = "%08X" % (hash & 0xFFFFFFFF)

    # цикл с картой, если КС совпадают и идентификатор пакета 42, то отмечаем
    # координаты
    if (crc_in == crc_solve) and (ident == '42'):
        # цикл для открытия карты, для считывания первой координаты
        if for_center_of_map == 1:
            # запоминаем координату открытия карты
            center_lat = lat
            center_lon = lon
            # запоминаем предыдущую первую координату для соединения
            # координат линиями
            lat1 = center_lat
            lon1 = center_lon
            # открываем карту
            map = folium.Map(location=[center_lat, center_lon],
                              zoom_start=16)
            # меняем значение вспомогательной переменной, так как карту мы уже
```

```

открыли
    for_center_of_map = 2

    # отмечаем точку на карте
    folium.Marker(location=[lat, lon],
                  popup=f"Скорость:{speed}км/ч \nНаправление:{way}°
\nДата:\n {day}.{month}.{year} \nВремя:\n {time}",
                  icon=folium.Icon(color="red")).add_to(map)

    # записываем результат на карту
    map.save("map1.html")

    # соединяем координаты линиями
    folium.PolyLine(locations=[(float(lat1), float(lon1)), (lat, lon)],
                    color="red", opacity=1).add_to(map)

    # запоминаем предыдущую координату, чтобы соединить ее и следующую
линией
    lat1 = lat
    lon1 = lon

    print('КС совпадают')
else:
    print('КС не совпадают')
    wrong_ks = wrong_ks + 1

# печатаем количество неверных КС
print('Количество неверных КС - ', wrong_ks)

```

## Результат работы программы.

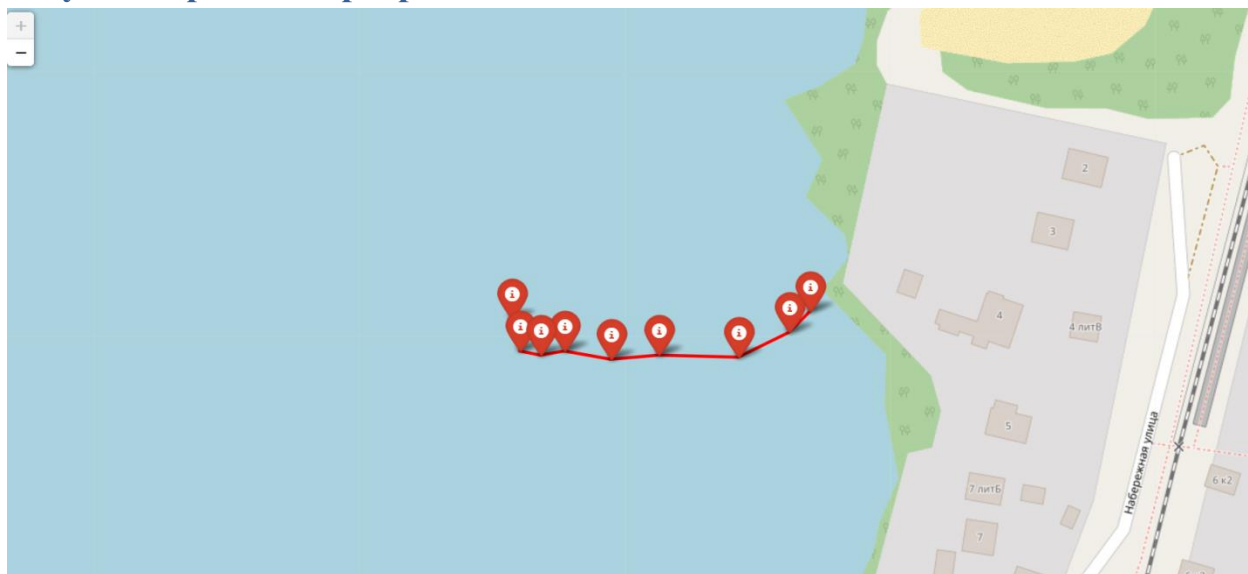


Рис. 2

## Маркеры на карте

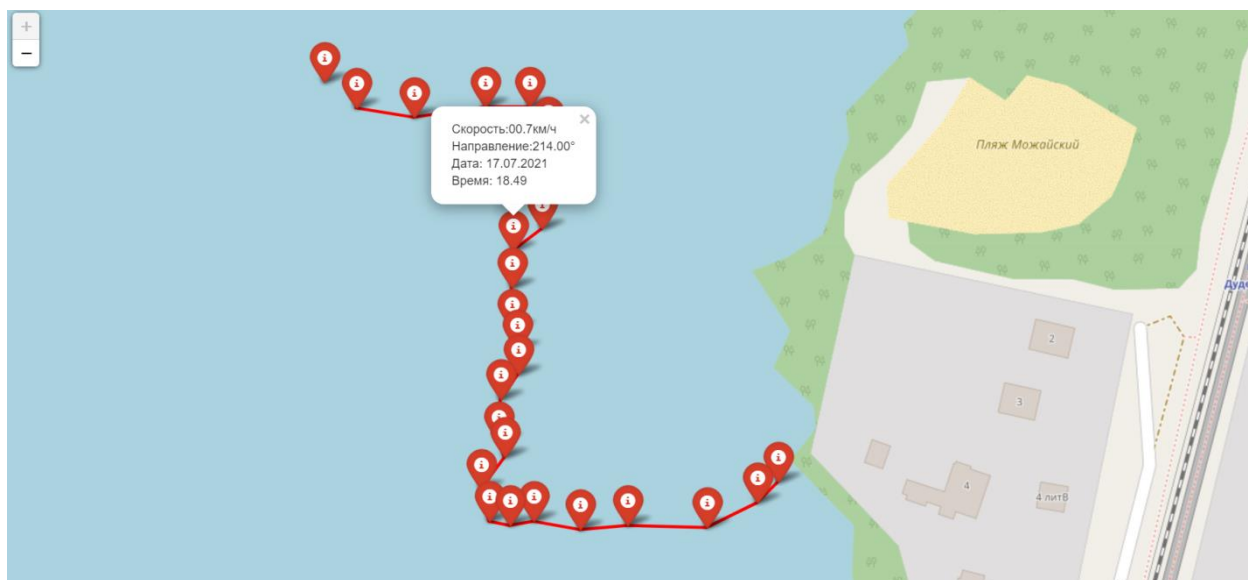


Рис. 3

## Окно с информацией

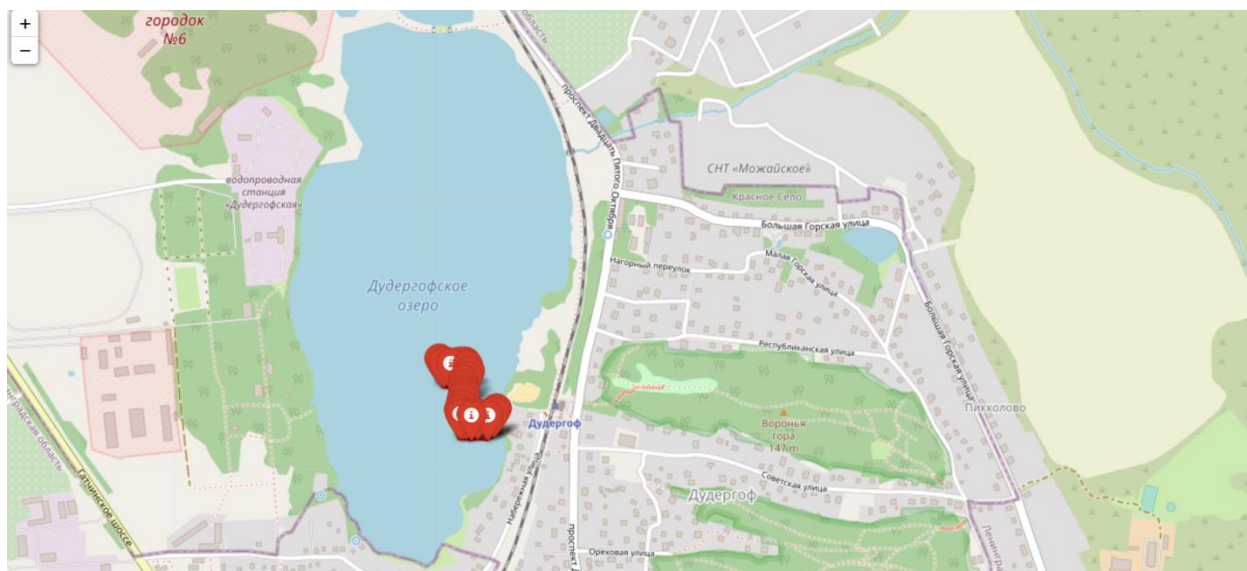


Рис. 4

Увеличенная карта

```

КС совпадают
b'42+059.701651+030.11548500.0150.001707202118.460xdd7aa662\n'
КС совпадают
b'42+059.701659+030.11507200.9156.001707202118.460xb160932d\n'
КС совпадают
b'42+059.701648+030.11482300.3172.001707202118.460x2ef1352d\n'
КС совпадают

```

Рис. 5

Информация на консоли во время работы программы

```

b'42+059.701846+030.11518900.5267.001707202119.170xa5a10a73\n'
КС совпадают
b'42+059.701770+030.11585500.7081.001707202119.170x9e637a2b\n'
КС совпадают
Количество неверных КС - 0
Process finished with exit code 0

```

Рис. 6

Информация на консоли после завершения работы

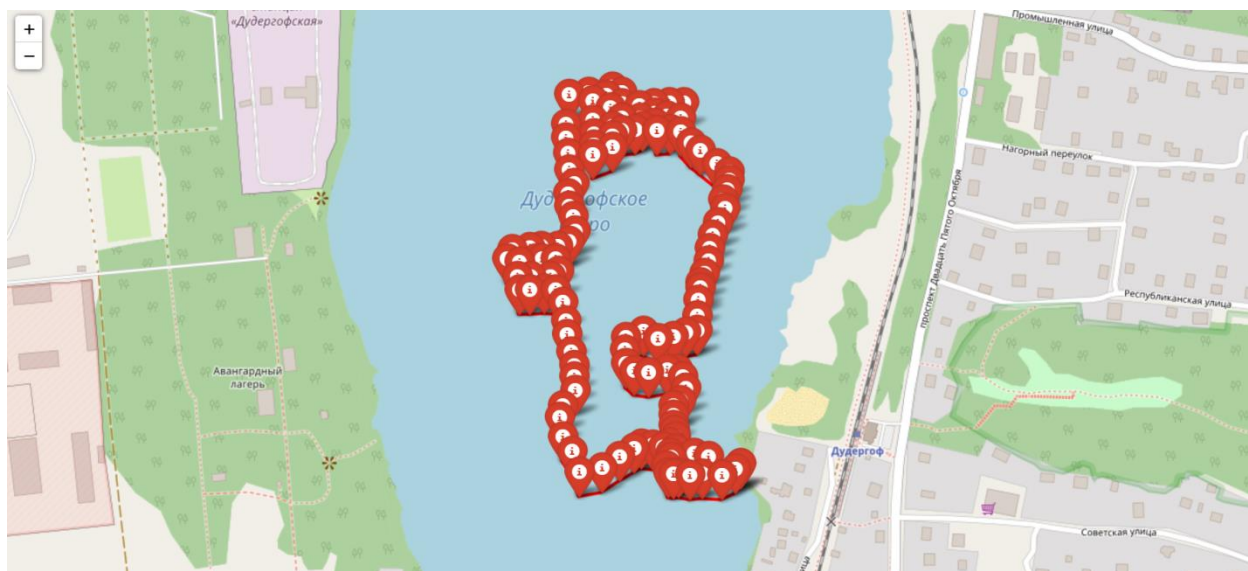


Рис. 7

Карта после завершения работы программы



Рис. 8

Траектория движения ТНА

### **Заключение.**

Разработано программное обеспечение (ПО) для контроля местоположения и режима движения телеуправляемого необитаемого надводного аппарата (ТННА) с возможностью наблюдать траекторию движения и информацию о скорости и направлении движения ТННА. Также описан протокол информационного обмена.