**Министерство науки и высшего образования Российской Федерации**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Алгоритмы и структуры данных»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №0**

**Выполнил:**

Гачко Г. Д., студент группы N3246

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Ерофеев С.А.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(отметка о выполнении)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Содержание**

[**Введение 3**](#_heading=h.pjjgzxww1s6h)

[**Анализ станции 4**](#_heading=h.m5hsqu2eqs9u)

[**Обзор отдельных зон 5**](#_heading=h.iyne7skjewka)

[Вестибюль 5](#_heading=h.of7i439c309)

[Расположение ТК 5](#_heading=h.9nci4hgzs63x)

[Потенциальные угрозы 5](#_heading=h.dhxrc3pemzk6)

[Моделирование 6](#_heading=h.7807e69h2up0)

[Расчет ПП 8](#_heading=h.51jo5bgj5kdq)

[Пути 9](#_heading=h.ww0d6zhw1m3)

[Расположение ТК 9](#_heading=h.kgcbq44u3dp8)

[Потенциальные угрозы 9](#_heading=h.h6ve3ttr57px)

[Моделирование 9](#_heading=h.p85su0khk2h6)

[Расчет ПП 11](#_heading=h.m0gkfipjqvn)

[Входы в служебные помещения 12](#_heading=h.ir0c7amust9q)

[Расположение ТК 12](#_heading=h.xcq9e5rgp5dm)

[Потенциальные угрозы 13](#_heading=h.623znwyjx0df)

[Моделирование 13](#_heading=h.sidd1vhb9207)

[Расчет ПП 14](#_heading=h.1fcz33u7jz34)

[**Предложения по улучшению СТВН 15**](#_heading=h.b2ite1qka898)

[**Заключение 16**](#_heading=h.1nsx5egr0wrx)

[**Список литературы 17**](#_heading=h.r6hqwoi010c1)

# Постановка задачи

Цель работы – написать программу, вычисляющую корни квадратного уравнения.

Для написания программы выбран язык Python 3.11.3

# Техническое задание

Реализовать программу, вычисляющую корни квадратного уравнения (в области действительных чисел).

### Описание функционала программы

Программа может работать в двух режимах – режиме тестирования и режиме ручного управления. В режиме тестирования программа в автоматическом режиме запускает заранее установленный набор тестов, проверяя каждый из возможных исходов выполнения программы. Подробнее о тестах в разделе Тесты данного отчета.

В режиме ручного управления программа принимает на вход из stdin три числа, разделенные пробелами. Эти числа представляют собой три коэффициента квадратного управления при , и – **a**, **b** и **с** соответственно.

При получении входных данных, не соответствующих заявленному формату ввода, программа выводит сообщение об ошибке и завершает исполнение.

Далее программа проходит по алгоритму вычисления корней уравнения

описанному в разделе Блок-схема.

### 

### Блок-схема

### Код программы

import math

def tetra\_solve(a = 'error', b = 'error', c = 'error', testing\_mode = False):

    if not testing\_mode:

        try:

            # Ввод переменных.

            # float a - коэффициент при x^2

            # float b - коэффициент при x^1

            # float c - коэффициент при x^0

            a, b, c = map(float, input("Введите коэффициенты квадратного уравнения, начиная с коэффициента при x^2 (a b c):\n>>> ").split())

        except ValueError:

            print("Введите число, будьте человеком") if not testing\_mode else False

            return False

    if (a == 0):

        if (b == 0):

            if (c == 0):

                # Случай a == 0 И b == 0 И с == 0.

                # Если все коэффициенты == 0, то уравнение обращается в

                # 0 = 0

                # Следовательно, уравнение верно при любых x

                print("x - любое") if not testing\_mode else False

                return True

            else:

                # Случай a == 0 И b == 0 И с != 0.

                # Если коэффициенты при x == 0, а c != 0, то уравнение обращается в

                # с = 0, где с != 0

                # Следовательно, уравнение ложно при любых x

                print("нет корней") if not testing\_mode else False

                return {}

        else:

            # Случай a == 0 И b != 0 И с - любое

            # Если коэффициент при x^2 == 0, а b != 0, то уравнение обращается в линейное

            # bx + c = 0, где b != 0

            # Следовательно, решим линейное уравнение с помощью формулы

            # x =  -c / b

            print("уравнение обращается в линейное") if not testing\_mode else False

            x = -c / b

            print("%.4f" % x) if not testing\_mode else False

            return {x}

    else:

        # Случай a != 0 И b,c - любые

        # Если коэффициент при x^2 != 0, то уравнение является квадратным.

        # Определим количество и вид его корней через формулу дискриминанта

        # D = b\*b + (-4) \* a \* c

        D = (b\*b) + ((-4) \* a \* c) # float D - дискриминант квадратного уравнения

        if D > 0:

            # Случай D > 0

            # Если дискриминант больше нуля, то квадратное уравнение имеет два различных действительных корня

            # Обозначим их как x1 и x2

            # И найдем по стандартным формулам для корней квадратного уравнения, представленным ниже

            x1 = ( -b + math.sqrt(D) ) / (2\*a) # float x1 - первый корень квадратного уравнения

            x2 = ( -b - math.sqrt(D) ) / (2\*a) # float x2 - второй корень квадратного уравнения

            print("D > 0") if not testing\_mode else False

            print("%.4f" % x1) if not testing\_mode else False

            print("%.4f" % x2) if not testing\_mode else False

            return {x1, x2}

        elif D == 0:

            # Случай D == 0

            # Если дискриминант равен нулю, то квадратное уравнение имеет два кратных (одинаковых) действительных корня

            # Обозначим их как x1 = x2 = x

            # И найдем по стандартной формуле для кратных корней квадратного уравнения, представленной ниже

            print("D == 0") if not testing\_mode else False

            x = (-b) / (2\*a)

            print("%.4f" % x) if not testing\_mode else False

            return {x}

        elif D < 0:

            # Случай D < 0

            # Если дискриминант меньше нуля, то квадратное уравнение имеет два мнимых корня

            # В рамках нашего алгоритма мы не занимаемся их вычислением

            # И найдем по стандартной формуле для кратных корней квадратного уравнения, представленной ниже

            print('a > 10') if not testing\_mode else False

            print("нет действительных корней") if not testing\_mode else False

            return {}

mode = input("Какой режим запуска? (run [T]ests / run [M]anually)")

if mode == 'M':

    while True:

    # функция вычисления корня будет автоматически запускаться заново после конца текущего вызова

        tetra\_solve()

elif mode == 'T':

    # Семь тестов

    # 1 Неверные входные данные

# TODO !!!

    # Случай D > 0

    # Если дискриминант больше нуля, то квадратное уравнение имеет два различных действительных корня

    # Обозначим их как x1 и x2

    # И найдем по стандартным формулам для корней квадратного уравнения, представленным ниже

    assert {} == tetra\_solve(3, -4, 94, testing\_mode = True)

    print("Тест D > 0 успешен")

    # Случай D == 0

    # Если дискриминант равен нулю, то квадратное уравнение имеет два кратных (одинаковых) действительных корня

    # Обозначим их как x1 = x2 = x

    # И найдем по стандартной формуле для кратных корней квадратного уравнения, представленной ниже

    assert {3.5} == tetra\_solve(-4, 28, -49, testing\_mode = True)

    print("Тест D == 0 успешен")

    # Случай D < 0

    # Если дискриминант меньше нуля, то квадратное уравнение имеет два мнимых корня

    # В рамках нашего алгоритма мы не занимаемся их вычислением

    # И найдем по стандартной формуле для кратных корней квадратного уравнения, представленной ниже

    assert {3, -3} == tetra\_solve(-6, 0, 54, testing\_mode = True)

    print("Тест D < 0 успешен")

    # Случай a == 0 И b == 0 И с == 0.

    # Если все коэффициенты == 0, то уравнение обращается в

    # 0 = 0

    # Следовательно, уравнение верно при любых x

    assert True == tetra\_solve(0, 0, 0, testing\_mode = True)

    print("Тест a = 0, b = 0, c = 0 успешен")

    # Случай a == 0 И b == 0 И с != 0.

    # Если коэффициенты при x == 0, а c != 0, то уравнение обращается в

    # с = 0, где с != 0

    # Следовательно, уравнение ложно при любых x

    assert {} == tetra\_solve(0, 0, 4, testing\_mode = True)

    print("Тест a == 0, b == 0, с != 0 успешен")

    # Случай a == 0 И b != 0 И с - любое

    # Если коэффициент при x^2 == 0, а b != 0, то уравнение обращается в линейное

    # bx + c = 0, где b != 0

    # Следовательно, решим линейное уравнение с помощью формулы

    # x =  -c / b

    assert {-2} == tetra\_solve(0, 4, 8, testing\_mode = True)

    print("Тест линейного уравнения успешен")

else:

    print("Нет такого режима")

Рис. 8 - Моделирование камер путей при помощи сайта jvsg.com

Рис. 9 - Моделирование камер путей при помощи сайта jvsg.com

Параметры моделирования:

Камера: Axis P1365 Mk II

Формат сенсора камеры: 1/2.8’’

Фокусное расстояние: 4.5 мм.

Разрешение камеры: 1920x1080

Высота установки камеры: 0.8 м.

Высота цели: 1 м.

Расстояние до цели: 57 м.

### Расчет ПП

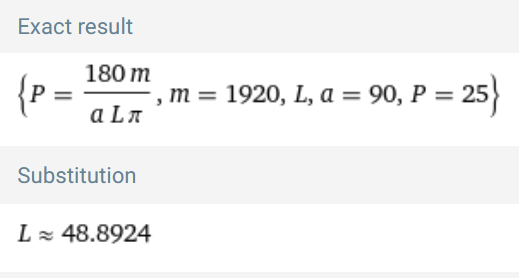


Рис. 10 - Расчёт максимальной дистанции выполнения задачи СТВН с использованием wolframalpha.com

Расчет по формуле показывает, что для используемых камер задача наблюдения будет выполняться на расстоянии до 48.9 метров.

## Входы в служебные помещения

Рис. 11 - Схема входов в служебные помещения на ст. м. “Новочеркасская”

### Расположение ТК

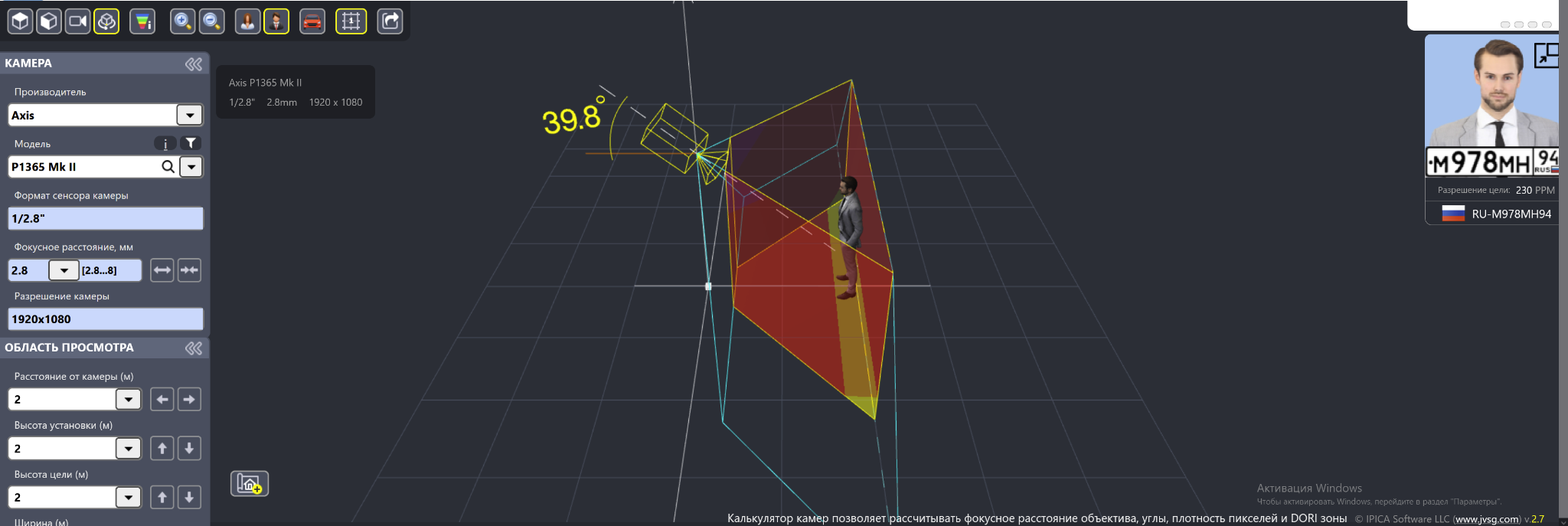
На входе в служебные помещения в дальнем торце вестибюля расположено 10 ТК (на Рис. 11 пронумерованы как ТК 4-1, 4-2, и т.д. до ТК 4-10).

### Потенциальные угрозы

1. доступ в служебные помещения посторонних лиц
2. конфликтные ситуации, правонарушения и экстренные ситуации - хулиганство, кражи, физическое насилие, терроризм, необходимость срочного оказания медицинской помощи

Исходя из приведённых угроз, для ТК в зоне служебных помещений была выбрана задача **идентификации**.

### Моделирование

Рис. 12 - Моделирование камеры над входом в служебные помещения при помощи сайта jvsg.com

Моделирование показало, что задача **идентификации** выполняется на дистанции до 1.8 метра от камеры.

### Расчет ПП

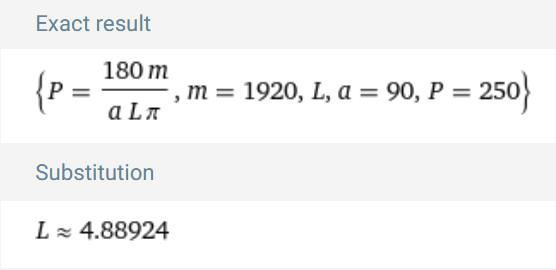


Рис. 13 - Расчёт максимальной дистанции выполнения задачи СТВН с использованием wolframalpha.com

Расчет по формуле показывает, что для используемых камер задача наблюдения будет выполняться на расстоянии до 4.8 метров.

# Предложения по улучшению СТВН

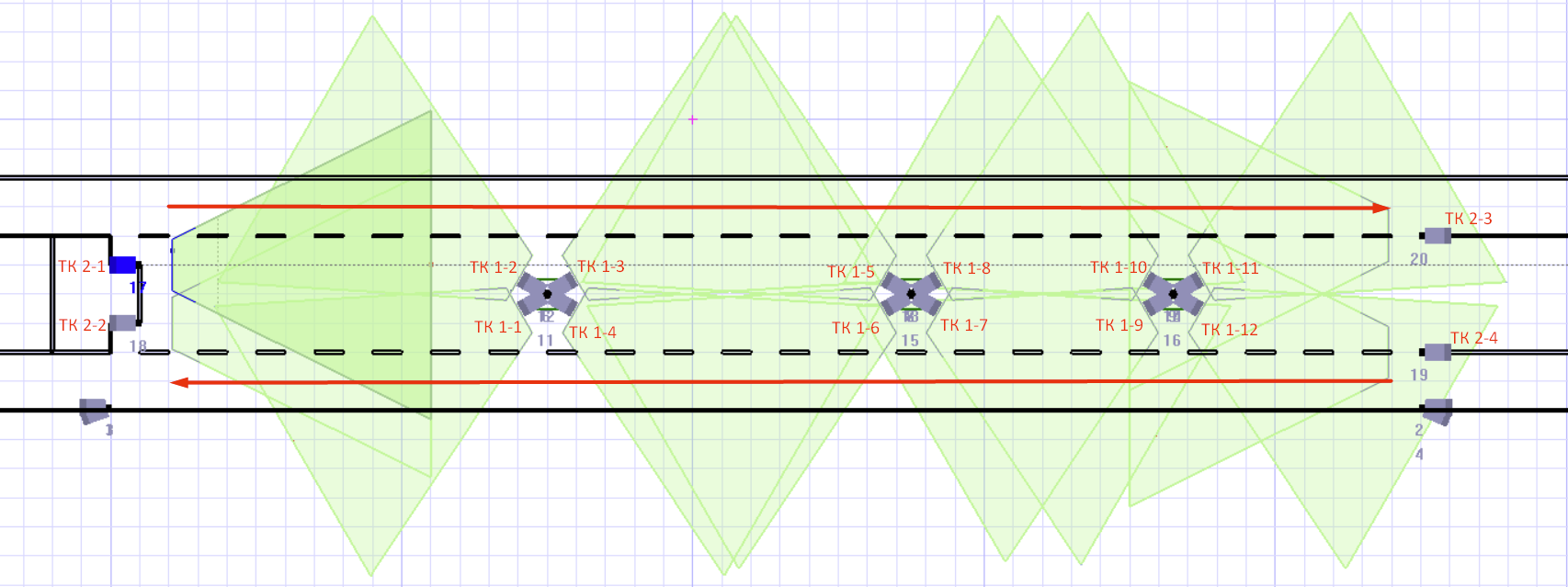
В зоне вестибюля ст.м., задача **наблюдения** выполняется не на всем пространстве вестибюля.

Рис. 14 - Текущее покрытие вестибюля

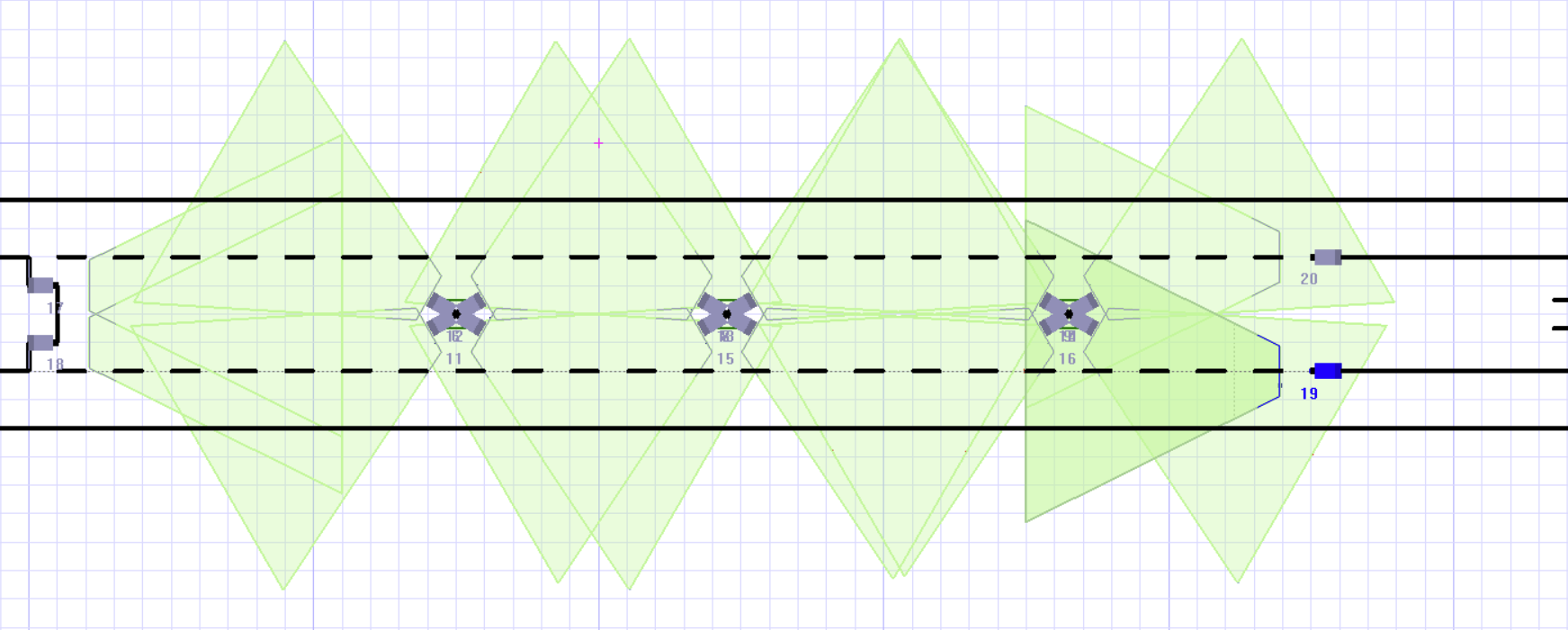
На схеме видно, что пространство под первым слева столбом для ТК не просматривается в достаточном качестве, равно как и пространство сверху и снизу от этого столба. Перенос центрального столба **на 6 метров влево** (как показано на Рис. 14) позволит покрыть эти “слепые зоны”, не жертвуя покрытием остальной части вестибюля.

Рис. 15 - Покрытие вестибюля после улучшений

# Заключение

В данной работе выполнено:

* проанализирована установка СТВН на ст.м. “Новочеркасская” г. Санкт-Петербурга
* выделены различные зоны этой ст.м., требующие контроля с помощью СТВН
* оценены угрозы согласно выбранным зонам
* определены задачи СТВН для выбранных зон согласно предполагаемым угрозам
* рассмотрена текущая конфигурация СТВН
* внесены предложения по улучшению конфигурации СТВН

Также приведены схемы, планы и расчёты, использованные в работе.

# Список литературы

1. Волхонский, В.В. СТВН-1 ТВН студентам 221123. - URL: https://drive.google.com/drive/folders/1\_xi5O5XQ05PyMC9M\_aR-hKENz1IU6Ts5
2. Волхонский, В.В. СТВН-1 ТВН студентам 211123. - URL: https://drive.google.com/drive/folders/1\_xi5O5XQ05PyMC9M\_aR-hKENz1IU6Ts5
3. Прогулки по метро : сайт - URL: https://www.metrowalks.com (дата обращения: 17.01.2024)
4. Video Surveillance Design Apps | JVSG : сайт - URL: https://www.jvsg.com/ (дата обращения: 17.01.2024).
5. Петербургский метрополитен | Метро 2-х столиц : сайт - URL: https://metro2.org/spb (дата обращения: 17.01.2024).