

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Московский институт электроники и математики**

Козьмин Андрей Викторович, группа БИВ247

Корсаев Артемий Батаевич, группа БИВ247

**БУДИЛЬНИК  
С ТЕХНОЛОГИЕЙ РАСПОЗНАВАНИЯ ПОЗЫ ЧЕЛОВЕКА**

Междисциплинарная курсовая работа  
по направлению 09.03.01 Информатика и вычислительная техника  
студентов образовательной программы бакалавриата  
«Информатика и вычислительная техника»

Студент \_\_\_\_\_  
подпись И.О. Фамилия

Студент \_\_\_\_\_  
подпись И.О. Фамилия

Руководитель  
Бакалавр, Старший преподаватель

\_\_\_\_\_  
И.О. Фамилия

Москва 2024 г.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

**ЗАДАНИЕ**

**на междисциплинарную курсовую работу бакалавра**  
студенту группы БИВ247 Козьмину Андрею Викторовичу

1. Тема работы

Будильник с технологией распознавания позы человека.

2. Требования к работе.

2.1 Устройство может воспроизводить звуковые сигналы.

2.2 Устройство может отправлять/получать данные по Wi-Fi.

2.3 Устройство может снимать видео.

3. Содержание работы

3.1 Написание программы для отправки данных микроконтроллером по http.

3.2 Написание программы для получения данных с камеры.

3.3 Написание программы для воспроизведения звуковых сигналов.

3.4 Проектирование и разработка устройства (электрическая схема и корпус).

4. Сроки выполнения этапов работы

Первый вариант МКР предоставляется студентом в срок до «\_\_\_» \_\_\_\_\_ 2024г.

Итоговый вариант МКР предоставляется студентом в срок до «\_\_\_» \_\_\_\_\_ 2024г.

Задание выдано «\_\_\_» \_\_\_\_\_ 2024г. \_\_\_\_\_ А.М. Елисеенко

подпись руководителя

Задание было принято

к исполнению «\_\_\_» \_\_\_\_\_ 2024г. \_\_\_\_\_ А.В. Козьмин

подпись студента

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

**ЗАДАНИЕ**

**на междисциплинарную курсовую работу бакалавра  
студенту группы БИВ247 Корсаеву Артемию Батаевичу**

1. Тема работы

Будильник с технологией распознавания позы человека.

2. Требования к работе.

2.1 Программа может обрабатывать фотографии.

2.2 Программа может отправлять/получать данные по Wi-Fi.

2.3 Программа может сравнивать фотографии.

3. Содержание работы

3.1 Написание программы загрузки данных о позе.

3.2 Написание программы для передачи данных на устройство.

3.3 Написание программы для сравнения с загруженной позой.

4. Сроки выполнения этапов работы

Первый вариант МКР предоставляется студентом в срок до «\_\_\_» \_\_\_\_\_ 2024г.

Итоговый вариант МКР предоставляется студентом в срок до «\_\_\_» \_\_\_\_\_ 2024г.

Задание выдано «\_\_\_» \_\_\_\_\_ 2024г. \_\_\_\_\_ А.М. Елисеенко  
подпись руководителя

Задание было принято

к исполнению «\_\_\_» \_\_\_\_\_ 2024г. \_\_\_\_\_ А.Б. Корсаев  
подпись студента

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

**График выполнения междисциплинарной курсовой работы бакалавра**  
студента группы Козьмина Андрея Викторовича

Тема работы

Будильник с технологией распознавания позы человека.

Дата согласования первого

варианта МКР

«\_\_» \_\_\_\_\_ 2024г.

\_\_\_\_\_

А.М. Елисеенко

подпись руководителя

Дата согласования

итогового варианта МКР

«\_\_» \_\_\_\_\_ 2024г.

\_\_\_\_\_

А.В. Козьмин

подпись студента

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

**График выполнения междисциплинарной курсовой работы бакалавра**  
студента группы Корсаева Артемия Батаевича

Тема работы

Будильник с технологией распознавания позы человека.

Дата согласования первого

варианта МКР

«\_\_» \_\_\_\_\_ 2024г.

\_\_\_\_\_

А.М. Елисеенко

подпись руководителя

Дата согласования

итогового варианта МКР

«\_\_» \_\_\_\_\_ 2024г.

\_\_\_\_\_

А.Б. Корсаев

подпись студента

# Содержание

Введение .....	7
Актуальность .....	7
Цель работы .....	7
Задачи .....	8
Анализ существующих решений .....	8
Используемые компоненты .....	9
Глава I: Проектирование и реализация программного продукта .....	13
Постановка задачи .....	13
Анализ требований к программному продукту .....	13
Проектирование интерфейса и программной архитектуры .....	13
Выбор технологий .....	13
Реализуемая архитектура согласно паттернам проектирования .....	15
Реализация основных функций приложения .....	15
Тестирование и отладка .....	17
Глава II: Анализ результатов и возможные доработки .....	17
Оценка полученного результата .....	17
Дальнейшие шаги по улучшению .....	17
Возможности масштабирования .....	18
Проблемы, с которыми мы столкнулись .....	18
Заключение .....	18
Выводы по результатам работы .....	18
Достижение цели и выполнение задач .....	18
Личный опыт и приобретённые навыки .....	18
Перспективы продолжения работы .....	18

## Введение

### Актуальность

Сон является неотъемлемой частью жизни каждого человека. Многие пользуются будильниками и изо дня в день просыпаются под однообразную музыку, что может раздражать или надоедать (рис. 1). В связи с этим возникает необходимость создания устройства, позволяющего разнообразить ежедневную рутину пробуждения. В нашей работе мы разработали будильник, который отключается в момент, когда человек принимает необходимую позу. Это способствует более осознанному и активному пробуждению, а также уменьшает вероятность повторного засыпания.



*Рисунок 1 – Надоедливые ежедневные будильники.*

### Цель работы

Создать будильник и необходимое ПО для его функционирования, которые предоставляют следующий функционал: отключение музыки после принятия человеком необходимой позы.

## Задачи

1. Разработать приложение для удобного и интуитивного взаимодействия с будильником.
2. Разработать ПО для обработки пользовательских поз.
3. Разработать устройство и написать ПО для его функционирования.

## Анализ существующих решений

Проведя анализ открытых источников аналогичных решений найдено, не было. Но были выявлены решения, со схожими идеями:

1. Alarmy – android приложение, которое при срабатывании требует от пользователя совершение какой либо активности: решить математическую задачу, сделать фотографию заданного объекта или же небольшая физическая активность – потрясти телефон / дойти до определённого места.
2. Barcode Alarm Clock и QRAlarm – IOS и Android приложения соответственно, идея которых заключается в том, что для выключения звукового сигнала требуется просканировать определённый QR код, который пользователь заранее распечатал и поместил в помещении в определённое место.
3. Также существует множество будильников, основанных на инфракрасном датчике. Используя различные устройства – зачастую это пистолет – необходимо попасть инфракрасным лазером в приёмник, после чего будильник выключится.

Исходя из полученных данных, можно сделать вывод, что функционал нашего будильника не имеет прямых аналогов, но в то же время уже предпринимались попытки создания нестандартных решений в данной области.



## Используемые компоненты

Было принято решение разрабатывать проект, используя следующие компоненты:

Название и описание	Причины	Изображение
Микроконтроллер ESP32S3 – необходим для создания самого устройства будильника.	<ol style="list-style-type: none"> <li>1. Большое количество обучающего материала.</li> <li>2. Достаточное количество GPIO.</li> <li>3. Переходник на камеру.</li> <li>4. Достаточная производительность для многозадачной работы.</li> </ol>	
Android studio – официальная среда разработки приложений на Android.	<ol style="list-style-type: none"> <li>1. Большое количество встроенных инструментов для разработки.</li> <li>2. Наличие опыта работы в данной среде.</li> </ol>	
ESP IDF – официальный фреймворк для разработки ПО для микроконтроллеров ESP32.	<ol style="list-style-type: none"> <li>1. Обширная документация.</li> <li>2. Большое количество примеров.</li> <li>3. Наличие всех необходимых инструментов для разработки.</li> </ol>	
FastAPI – фреймворк для разработки RESP API сервисов на python.	<ol style="list-style-type: none"> <li>1. Высокая производительность.</li> <li>2. Понятная документация.</li> <li>3. Большое количество примеров и статей.</li> <li>4. Асинхронная обработка запросов.</li> </ol>	

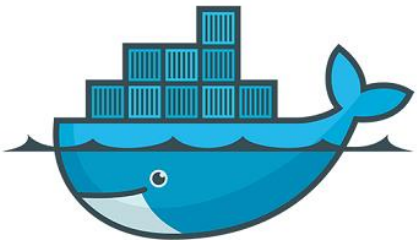



Название и описание	Причины	Изображение
Docker – инструмент для контейнеризации приложений.	<ol style="list-style-type: none"> <li>1. Наличие опыта использования данного инструмента.</li> <li>2. Обширная документация.</li> <li>3. Множество обучающих материалов и примеров.</li> <li>4. Простота развёртывания приложения.</li> </ol>	
OpenCV – мощная библиотека для обработки изображений.	<ol style="list-style-type: none"> <li>1. Наличие опыта использования этого инструмента.</li> <li>2. Популярное решение для множества задач.</li> <li>3. Хорошая производительность.</li> </ol>	
TensorFlow – фреймворк для обучения и использования моделей искусственного интеллекта.	<ol style="list-style-type: none"> <li>1. Имеется опыт использования данного фреймворка.</li> <li>2. Предоставляет большой функционал для построения моделей, обрабатывающих изображения.</li> </ol>	
Git – самая популярная система контроля версий.	<ol style="list-style-type: none"> <li>1. Большая распространённость.</li> <li>2. Имеется опыт использования этой системы контроля версий</li> </ol>	

Таблица 1 – Используемые компоненты.

## **1 Введение**

### **1.1 Актуальность**

Система “умный дом” постепенно охватывает нашу жизнь, чтобы упростить её. И в данной работе мы решили реализовать одну из его компонент – “умный” будильник.

### **1.2 Цель работы**

Цель работы – разработка комплекса ПО для взаимодействия с устройством. Пакет ПО включает себя:

1. Мобильное приложение для мобильных устройств на базе Android.
2. ПО для сервера
3. ПО для ESP32

### **1.3 Задачи**

В ходе выполнения работы и для достижения поставленных целей необходимо выполнить следующие задачи:

1. Реализация мобильного приложения для мобильных устройств на базе Android.
2. Реализация ПО для сервера.
3. Реализация ПО для ESP32.

### **1.4 Личный вклад участников**

К задачам Корсаева Артемия относятся:

1. Изучение документации tensorflow
2. Разработка сервера для обработки фотографий
3. Разработка метода сравнения фотографий
4. Поиск метода и хранения данных и его реализация, по которым происходит сравнение фотографий
5. Разработка метода построения ключевых точек позы по фотографии

## **2 Программная реализация**

### **2.1 Использование Movenet для поиска ключевых точек позы по фотографии**

Выбор пал на данную модель, как зарекомендовавшую себя своей точностью при малых аппаратных мощностях.

Полученные от Movenet точки преобразуются в вектора, выражающие собой силуэт человека.

## 2.2 Создание сервера

В качестве фреймворка был использован FastAPI. Выбор пал именно на него, так как он является наиболее лучшей опцией для построения API, с помощью которого работает передача данных между мобильным устройством на базе Android и устройством будильника.

## 2.3 Построение алгоритма сравнения поз с фотографий

Приведем пример. Алгоритм получает вектора, полученные с изображения с камеры устройства и с эталонного изображения. Далее для каждого вектора с изображения с камеры устройства берем мы ищем соответствующую пару, например, голень левой ноги с изображения с камеры устройства с голенью левой ноги с эталонного изображения. В случае, если пара не находится, текущее то позы на этих изображениях разные. Далее вычисляется угол, образующиеся между векторами. Если угол превышает пороговое значение, то позы на этих изображениях разные.

```
import numpy as np
from math import sqrt
import os

_LIMIT = 20

def estimate(pattern, detect):
    """compare detected image with pattern
    Args:
        pattern: pattern picture, etalon.
        detect: detected image by esp32.
    Returns:
        Flag - is detected image is similar with pattern."""
    mean = 0
    for i in detect:
        os.system('clear')
        detect_points = detect[i][1] - detect[i][0]
        try:
            pattern_points = pattern[i][1] - pattern[i][0]
        except Exception:
            print(f'cannot get data about edge: {i}')
            return False
        #edges consist из точки начала и конца
        dot = np.dot(detect_points, pattern_points) #скалярное умножение
        detect_norm = np.linalg.norm(detect_points, ord=None) # норма Фробениуса
        pattern_norm = np.linalg.norm(pattern_points, ord=None)
        angle = np.degrees(np.arccos(dot/(detect_norm*pattern_norm)))
        print(f'ANGLE {i}: {angle}')
        if angle > _LIMIT:
            print(f'NO. fix your {i}')
            return False
    print('YES')
    return True
```

Рисунок 2 – программная реализация функции estimate()

# Глава I: Проектирование и реализация программного продукта

## Постановка задачи

Программный продукт на вход принимает изображение и на выход возвращает результат, похожа ли поза с картинке на входе на эталонную позу.

## Анализ требований к программному продукту

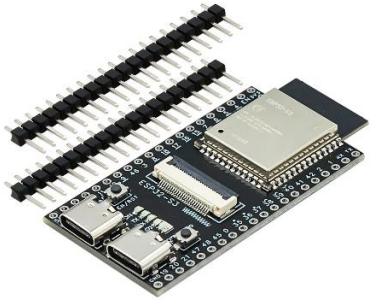

Это who?



## Проектирование интерфейса и программной архитектуры

Это who?

## Выбор технологий

Было принято решение разрабатывать проект, используя следующие технологии:

Название и описание	Причины	Изображение
Микроконтроллер ESP32S3 – необходим для создания самого устройства будильника.	5. Большое количество обучающего материала. 6. Достаточное количество GPIO. 7. Переходник на камеру. 8. Достаточная производительность для многозадачной работы.	
Android studio – официальная среда разработки приложений на Android.	3. Большое количество встроенных инструментов для разработки. 4. Наличие опыта работы в данной среде.	

ESP IDF – официальный фреймворк для разработки ПО для микроконтроллеров ESP32.	4. Обширная документация. 5. Большое количество примеров. 6. Наличие всех необходимых инструментов для разработки.	
FastAPI – фреймворк для разработки REST API сервисов на python.	5. Высокая производительность. 6. Понятная документация. 7. Большое количество примеров и статей. 8. Асинхронная обработка запросов.	

Название и описание	Причины	Изображение
Docker – инструмент для контейнеризации приложений.	5. Наличие опыта использования данного инструмента. 6. Обширная документация. 7. Множество обучающих материалов и примеров. 8. Простота развёртывания приложения.	
OpenCV – мощная библиотека для обработки изображений.	4. Наличие опыта использования этого инструмента. 5. Популярное решение для множества задач. 6. Хорошая производительность.	

TensorFlow – фреймворк для обучения и использования моделей искусственного интеллекта.	3. Имеется опыт использования данного фреймворка. 4. Предоставляет большой функционал для построения моделей, обрабатывающих изображения.	
Git – самая популярная система контроля версий.	3. Большая распространённость. 4. Имеется опыт использования этой системы контроля версий	

Таблица 2 – Используемые технологии.

Реализуемая архитектура согласно паттернам проектирования  
(а что писать?)

Реализация основных функций приложения

Одной из важнейших функций обработки изображения является `movenet()`, которая отвечает за нахождение ключевых точек на изображении.

```

model_name = "movenet_lightning"
input_size = 256
interpreter = tf.lite.Interpreter(model_path="./CV/model.tflite")
interpreter.allocate_tensors()

def movenet(input_image):
    input_image = tf.cast(input_image, dtype=tf.uint8)
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()
    interpreter.set_tensor(input_details[0]['index'], input_image.numpy())
    interpreter.invoke()
    keypoints_with_scores = interpreter.get_tensor(output_details[0]['index'])
    return keypoints_with_scores

```

Рисунок 3 – листинг функции *movenet()*

Эта функция используется уже в *get\_keypoints()*, которая преобразует изображения в вид, необходимый для функции *movenet()*. Так, в список преобразований входит возможность получения на вход картинки как строки, символизирующей путь к изображению, так и с *numpy*-массивами. Вход преобразуется тип данных *Tensor()*, используемых библиотекой *tensorflow*, а затем проходит масштабирование *Tensor()*, который посылается уже в функцию *movenet()*, результат которой возвращается из *get\_keypoints()*.

```
def get_keypoints(image, from_narray=False):
    """get keypoints on skeleton.
    Args:
        image: a detected picture by web camera. Can be a nparray and path to picture/
        from_narray: how parse image
    Returns:
        keypoints"""
    if from_narray:
        image = tf.convert_to_tensor(image, dtype=np.uint8)
        # image = tf.image.decode_jpeg(image)
        pass
    else:
        image_path = image
        image = tf.io.read_file(image_path)
        image = tf.convert_to_tensor(image)
        image = tf.image.decode_jpeg(image)

    input_image = tf.expand_dims(image, axis=0)
    input_image = tf.image.resize_with_pad(input_image, input_size, input_size)
    keypoints_with_scores = movenet(input_image)
    return keypoints_with_scores
```

Рисунок 4 – листинг функции *get\_keypoints()*

Одной из ключевых функций является *keypoints\_and\_edges\_for\_display()*, которая на вход получает ключевые точки, возвращая информацию для представления позы на изображении в виде линий(костей).



```

keypoints, edges, for_display=keypoints_with_scores,
height,
width,
keypoint_threshold=11, names=None):

"""Returns high confidence keypoints and edges for visualization.

Args:
    keypoints_with_scores: A numpy array with shape [1, 1, 17, 17] representing
        the keypoint coordinates and scores returned from the Roost model.
    height: height of the image in pixels.
    width: width of the image in pixels.
    keypoint_threshold: minimum confidence score for a keypoint to be
        visualized.

Returns:
    (keypoints_xy, edges_xy, edge_colors) containing:
    * the coordinates of all keypoints of all detected entities;
    * the coordinates of all skeleton edges of all detected entities;
    * the colors in which the edges should be plotted.

"""
    keypoints_all = []
    keypoints_edges_all = []
    edge_colors = []
    num_instances = ... = keypoints_with_scores.shape
    for idx in range(num_instances):
        kpts_x = keypoints_with_scores[idx, :, :, 0]
        kpts_y = keypoints_with_scores[idx, :, :, 1]
        kpts_scores = keypoints_with_scores[idx, :, :, 2]
        kpts_absorb_xy = np.stack(
            [kpts_x, kpts_y], axis=0)
        [kpts_x, kpts_y] = np.array(kpts_x, kpts_y[0], axis=0)
        kpts_above_thresh_absorb_xy = kpts_absorb_xy[
            kpts_scores > keypoint_threshold, :]
        keypoints_all.append(kpts_above_thresh_absorb_xy)
        edges_with_names = dist(
            for_edge_pair, color = KEYPOINT_EDGE_INDS_TO_COLOR[name])
        if keypoints_with_scores[idx, :, :, 2] > keypoint_threshold and
            keypoints_with_scores[idx, :, :, 2] > keypoint_threshold:
            x_start = kpts_absolute_xy[edge_pair][0], 0
            y_start = kpts_absolute_xy[edge_pair][1], 0
            x_end = kpts_absolute_xy[edge_pair][2], 0
            y_end = kpts_absolute_xy[edge_pair][3], 0
            line_seg = np.array([x_start, y_start], [x_end, y_end])
            keypoints_all.append(line_seg)
            edge_colors.append(color)
        edges_with_names.append(KEYPOINT_EDGE_INVERTED[edge_pair][1])
    return keypoints_xy, edges_xy, edge_colors

def keypoints_all(
    keypoints_xy = np.concatenate(keypoints_all, axis=0)
else
    keypoints_xy = np.zeros((0, 2, 2))

if keypoints_edges_all:
    edges_xy = np.stack(keypoints_edges_all, axis=0)
else
    edges_xy = np.zeros((0, 2, 2))
if names:
    return keypoints_xy, edges_xy, edge_colors, edges_with_names
else
    return keypoints_xy, edges_xy, edge_colors

```

Рисунок 4 – листинг функции keypoints\_and\_edges\_for\_display()

(пихнуть код по мобилке функции и есп)

## Тестирование и отладка

Для тестирования модели распознавания позы на начальном этапе была написана программа для её отладки. В качестве камеры, откуда берется поза, используется веб-камера компьютера.

Следующим этапом тестирование модели распознавания позы было использования камеры устройства на базе ESP32S3 вместо веб-камеры компьютера. (рассказать тест мобильного приложения и есп)

## Глава II: Анализ результатов и возможные доработки

## Оценка полученного результата

(12/10. А вообще, нужно что-то написать)

## Дальнейшие шаги по улучшению

Планируется сделать загрузку пользователем собственного шаблона позы, поддержку распознавания нескольких поз на фотографии, улучшение графического

интерфейса, а также улучшить функцию сравнения позы: например, чтобы функция оценивала не просто картинку, а видео длиной в 5 секунд.

### Возможности масштабирования

(А вообще, нужно что-то написать)

### Проблемы, с которыми мы столкнулись

Были проблемы с определением функции оценки позы и подбором порогового значения для неё. (А вообще, нужно что то написать)

## Заключение

### Выводы по результатам работы

(12/10. А вообще, нужно что-то написать)

### Достижение цели и выполнение задач

Мы считаем, что разработка минимально жизнеспособного продукта проведена успешно. Проект имеет большой потенциал за счёт интеграции с системой “умный дом”. (нужно что-то еще написать)

### Личный опыт и приобретённые навыки

Артемий Корсаев приобрел навыки работы в команде, работа с фреймворком FastAPI, а также разработкой с помощью инструментов tensorflow(А вообще, нужно что то написать)

### Перспективы продолжения работы

(А вообще, нужно что то написать)