Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Теория Систем

Лабораторная работа №1

Синтез управляющего конечного автомата

Выполнил:

Маликов Глеб Игоревич

Группа № P3324

Преподаватель:

Русак Алена Викторовна

Санкт-Петербург

2025

**Содержание**

[Задание 3](#_Toc190438347)

[Реализация 4](#_Toc190438348)

[Вывод 9](#_Toc190438349)

# Задание

Требуется спроектировать управляющий конечный автомат. Пример: конечный автомат, управляющий стиральной машиной

# Реализация

Реализованный конечный автомат симулирует работу кондиционера. Автомат управляет режимами работы кондиционера, включая различные активные режимы (охлаждение, обогрев, ЭКО, вентилятор), а также паузу, остановку, и состояние ошибки или поломки кондиционера (состояние "broken").

**Состояния:**

* off: Кондиционер выключен.
* on: Кондиционер включен, но режим работы не выбран.
* eco\_active: Активный режим ЭКО.
* eco\_pause: Режим ЭКО на паузе.
* fan\_active: Активный режим вентилятора.
* fan\_pause: Режим вентилятора на паузе.
* heating\_active: Активный режим обогрева.
* heating\_pause: Режим обогрева на паузе.
* cooling\_active: Активный режим охлаждения.
* cooling\_pause: Режим охлаждения на паузе.
* stop: Остановка работы по истечении таймера.
* broken: Кондиционер вышел из строя.

**Переходы:**

* power\_on: Переводит автомат из состояния 'off' в 'on'.
* power\_off: Переводит из состояний 'on', 'stop' или 'broken' в 'off'.
* set\_eco, set\_fan, set\_heating, set\_cooling: Переход из состояния 'on' в соответствующие активные режимы работы.
* timeout: По истечении рабочего таймера активное состояние (eco\_active, fan\_active, heating\_active, cooling\_active) переходит в 'stop'.
* pause: Приостановка работы активного режима – переход в соответствующее состояние 'pause'.
* resume: Возобновление работы из режима паузы – возвращение в активное состояние.
* random\_break: Случайный сбой, переводящий автомат из активного состояния в состояние 'broken'.
* repair: Переход из состояния 'broken' в 'on' после проведения ремонта.
* restart: Перезапуск работы после остановки – переход из 'stop' в 'on'.

Конечный автомат выполнен на языке Python с использованием библиотеки "transitions". Ниже представлены основные части кода автомата.

states = [

'off',

'on',

'eco\_active',

'eco\_pause',

'fan\_active',

'fan\_pause',

'heating\_active',

'heating\_pause',

'cooling\_active',

'cooling\_pause',

'stop',

'broken'

]

transitions\_list = [

# Переходы питания.

{'trigger': 'power\_on', 'source': 'off', 'dest': 'on'},

{'trigger': 'power\_off', 'source': ['on', 'stop', 'broken'], 'dest': 'off'},

# Переходы из состояния 'on' в рабочий режим.

{'trigger': 'set\_eco', 'source': 'on', 'dest': 'eco\_active'},

{'trigger': 'set\_fan', 'source': 'on', 'dest': 'fan\_active'},

{'trigger': 'set\_heating', 'source': 'on', 'dest': 'heating\_active'},

{'trigger': 'set\_cooling', 'source': 'on', 'dest': 'cooling\_active'},

# По истечении таймера работы активные состояния переходят в 'stop'.

{'trigger': 'timeout', 'source': ['eco\_active', 'fan\_active', 'heating\_active', 'cooling\_active'], 'dest': 'stop'},

# Переходы на паузу из активных режимов в состояние 'pause'.

{'trigger': 'pause', 'source': 'eco\_active', 'dest': 'eco\_pause'},

{'trigger': 'pause', 'source': 'fan\_active', 'dest': 'fan\_pause'},

{'trigger': 'pause', 'source': 'heating\_active', 'dest': 'heating\_pause'},

{'trigger': 'pause', 'source': 'cooling\_active', 'dest': 'cooling\_pause'},

# Переходы возобновления работы из состояния паузы в активное состояние.

{'trigger': 'resume', 'source': 'eco\_pause', 'dest': 'eco\_active'},

{'trigger': 'resume', 'source': 'fan\_pause', 'dest': 'fan\_active'},

{'trigger': 'resume', 'source': 'heating\_pause', 'dest': 'heating\_active'},

{'trigger': 'resume', 'source': 'cooling\_pause', 'dest': 'cooling\_active'},

# Случайный сбой может произойти в любом активном рабочем состоянии.

{'trigger': 'random\_break', 'source': ['eco\_active', 'fan\_active', 'heating\_active', 'cooling\_active'], 'dest': 'broken'},

# Ремонт неисправного кондиционера.

{'trigger': 'repair', 'source': 'broken', 'dest': 'on'},

# Перезапуск после остановки.

{'trigger': 'restart', 'source': 'stop', 'dest': 'on'},

]

class AC(GraphMachine):

def \_\_init\_\_(self):

super(AC, self).\_\_init\_\_(states=states, transitions=transitions\_list, initial='off', auto\_transitions=False)

# Дескрипторы таймеров (используются в рабочих режимах).

self.\_working\_timer = None

self.\_random\_break\_timer = None

Далее в классе AC описаны «Колбэк» функции, выполняющиеся при входе/выходе из состояний, как например запуск или отмена таймеров и вывод логов.

ac = AC()

log(f"Начальное состояние: {ac.state}", level="info")

ac.power\_on()

log("Состояние после включения: {}".format(ac.state), level="info")

ac.set\_cooling()

log("Состояние после установки режима охлаждения: {}".format(ac.state), level="info")

time.sleep(3)

ac.pause()

log("Состояние после постановки на паузу: {}".format(ac.state), level="info")

time.sleep(2)

ac.resume()

log("Состояние после возобновления работы: {}".format(ac.state), level="info")

log("Ожидание срабатывания таймеров (истечение времени или случайный сбой)...", level="info")

time.sleep(20)

if ac.state == 'broken':

ac.repair()

log("Состояние после ремонта: {}".format(ac.state), level="info")

log("Конечное состояние: {}".format(ac.state), level="info")

Результат работы данной симуляции представлен ниже.

2025-02-14 11:48:31.129 Начальное состояние: off

2025-02-14 11:48:31.137 Кондиционер включен.

2025-02-14 11:48:31.147 Состояние после включения: on

2025-02-14 11:48:31.151 Переход в режим ОХЛАЖДЕНИЕ.

2025-02-14 11:48:31.154 Запуск таймера работы на 10 секунд.

2025-02-14 11:48:31.160 Запуск таймера случайного сбоя на 6.43 секунд.

2025-02-14 11:48:31.182 Состояние после установки режима охлаждения: cooling\_active

2025-02-14 11:48:34.188 Выход из режима ОХЛАЖДЕНИЕ.

2025-02-14 11:48:34.201 Режим ОХЛАЖДЕНИЕ на паузе.

2025-02-14 11:48:34.220 Состояние после постановки на паузу: cooling\_pause

2025-02-14 11:48:36.229 Переход в режим ОХЛАЖДЕНИЕ.

2025-02-14 11:48:36.234 Запуск таймера работы на 10 секунд.

2025-02-14 11:48:36.242 Запуск таймера случайного сбоя на 9.33 секунд.

2025-02-14 11:48:36.250 Состояние после возобновления работы: cooling\_active

2025-02-14 11:48:36.255 Ожидание срабатывания таймеров (истечение времени или случайный сбой)...

2025-02-14 11:48:45.578 Произошел случайный сбой! Кондиционер вышел из строя.

2025-02-14 11:48:45.582 Выход из режима ОХЛАЖДЕНИЕ.

2025-02-14 11:48:45.587 Кондиционер вышел из строя! Пожалуйста, отремонтируйте его.

2025-02-14 11:48:56.259 Кондиционер включен.

2025-02-14 11:48:56.265 Состояние после ремонта: on

2025-02-14 11:48:56.272 Конечное состояние: on

С помощью встроенных возможностей библиотеки *transitions*, в частности класса GraphMachine, была сгенерирована диаграмма переходов конечного автомата, представленная на рисунке 1.

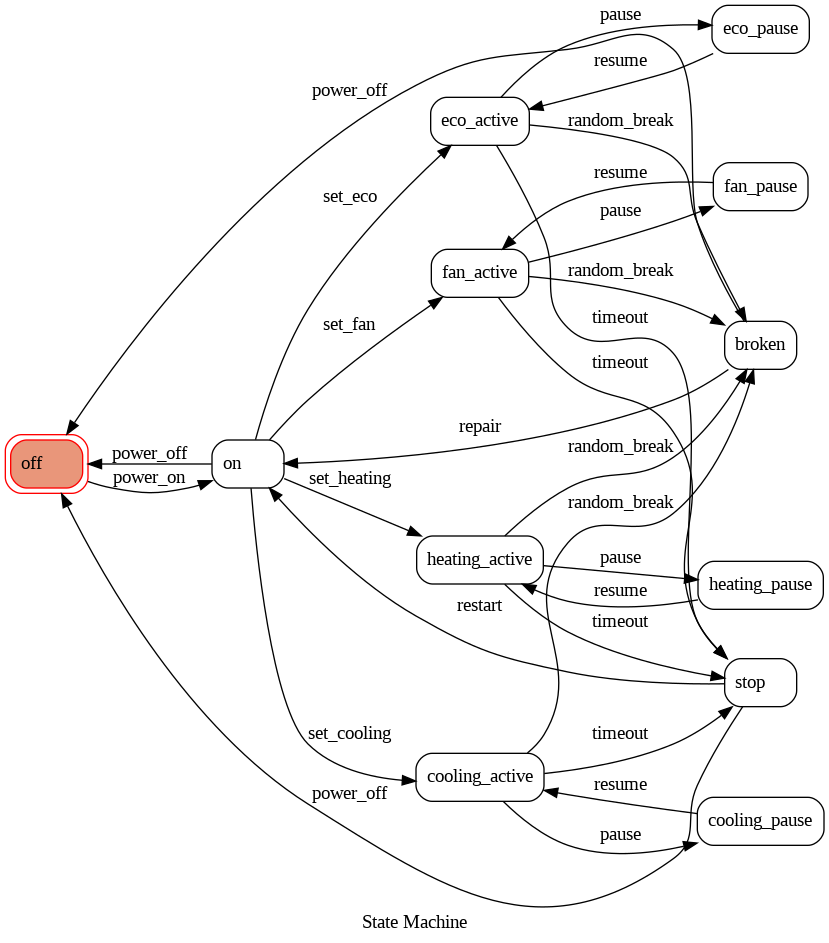


Рисунок 1 - Диаграмма переходов

# Вывод

В ходе лабораторной работы был спроектирован и реализован конечный автомат для управления работой кондиционера. В результате были четко определены состояния работы устройства, что позволило структурировать логику функционирования системы. Реализованные переходы между состояниями обеспечивают возможность выбора режима, постановку на паузу, автоматическую остановку по истечении заданного времени и обработку сбоев. Применение таймеров для моделирования автоматических переходов, таких как истечение времени работы или случайный сбой, продемонстрировало важность асинхронных процессов в управлении состояниями. Кроме того, использование библиотеки *transitions* значительно упростило реализацию и визуализацию конечного автомата, что является важным аспектом при разработке подобных систем.