Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ Международных образовательных программ_____

КАФЕДРА _____Программное обеспечение ЭВМ и информационные технологии_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

Моделирование программного обеспечения управления умного дома на основе сетей Петри

Студент ___ИУ7И-86Б_____ _____ ___ Нгуен Фыок Санг__
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР _____ ___ Рудаков И. В._
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

Нормоконтролер _____ _____
(Подпись, дата) (И.О.Фамилия)

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7
(Индекс)
_____ Рудаков И.В.
(И.О.Фамилия)
« ___ » _____ 20__ г.

З А Д А Н И Е
на выполнение выпускной квалификационной работы бакалавра

Студент группы ИУ7И-86Б

Нгуен Фыок Санг
(фамилия, имя, отчество)

Тема квалификационной работы Моделирование программного обеспечения управления умного дома на основе сетей Петри

Источник тематики (НИР кафедры, заказ организаций и т.п.)
НИР кафедры

Тема квалификационной работы утверждена распоряжением по факультету ИУ № 03.02.01-04.03/18 от « 11 » декабря 2021 г.

Часть 1. Аналитический раздел
Провести рассмотрение типов сетей Петри а также моделирование на основе вложенных сетей Петри. Проанализировать характеристики компонентов системы умного дома.

Часть 2. Конструкторский раздел

Спроектировать и описать систему умного дома и спроектировать взаимосвязь между ее компонентами. Разработать схему сетей Петри

Часть 3. Технологический раздел

Осуществить выбор языка и среды среда реализации ПО. Описать структуру разрабатываемого ПО метода. Провести функциональное тестирование.

Часть 4. Исследовательский раздел

Исследовать эффективность разработанного программного обеспечения, а также синхронизацию и взаимодействие объектов в подсистемах умного дома.

Оформление квалификационной работы:

Расчетно-пояснительная записка на _50-70_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания « » _____ 2021 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до « __ » _____ 2022_ г.

Руководитель квалификационной работы _____ ____Рудаков И. В._____
(Подпись, дата) (И.О.Фамилия)

Студент _____ ____Нгуен Ф. С. ____
(Подпись, дата) (И.О.Фамилия)

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ УТВЕРЖДАЮ

КАФЕДРА ИУ-7 Заведующий кафедрой ИУ-7

ГРУППА ИУ7У-76Б _____ И. В. Рудаков

«1» декабря 2021 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения выпускной квалификационной работы

студента: Нгуен Фыок Санг

Тема квалификационной работы Моделирование программного обеспечения управления умного дома на основе сетей Петри

№ п/п	Наименование этапов выпускной квалификационной работы	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Должность	ФИО, подпись
1.	Задание на выполнение работы. Формулирование проблемы, цели и задач работы	_____ *Планируемая дата*		Руководитель ВКР	
2.	1 часть. Аналитический раздел	_____ *Планируемая дата*		Руководитель ВКР	
3.	Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач	_____ *Планируемая дата*		Заведующий кафедрой	
4.	2 часть. Конструкторский раздел	_____ *Планируемая дата*		Руководитель ВКР	
5.	3 часть. Технологический раздел	_____ *Планируемая дата*		Руководитель ВКР	
6.	4 часть. Исследовательский раздел	_____ *Планируемая дата*		Руководитель ВКР	
7.	1-я редакция работы	_____ *Планируемая дата*		Руководитель ВКР	
8.	Подготовка доклада и презентации	_____ *Планируемая дата*			
9.	Заключение руководителя	_____ *Планируемая дата*		Руководитель ВКР	
10.	Нормоконтроль	_____ *Планируемая дата*		Нормоконтролер	
11.	Внешняя рецензия	_____ *Планируемая дата*			
12.	Защита работы на ГЭК	_____ *Планируемая дата*			

Студент _____ _____
(подпись, дата)

Руководитель работы _____
(подпись, дата)

РЕФЕРАТ

Расчетно-пояснительная записка содержит 54 с., 4 ч., 27 рис., 6 табл., 10 источников.

Тема: Моделирование программного обеспечения управления умного дома на основе сетей Петри

Ключевые слова: моделирование, Петри, сеть Петри, умный дом

Объектом исследования работы является методы, позволяющее запускать подозрительные исполняемые файлы на замаскированных виртуальных машинах.

Цель работы – разработать метод моделирования программного обеспечения управления умного дома на основе сетей Петри.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Разработка инструментов моделирования и анализа поведения распределенных систем является актуальной задачей, особенно в связи с постоянным увеличением сложности проектируемых и исследуемых систем.

Сети Петри - один из наиболее распространенных современных формализмов моделирования и анализа параллельных распределенных систем.

В настоящее время проводится большое количество как теоретических исследований в этой области, так и реализации практических инструментов для разработки распределенных алгоритмов на основе сетей Петри.

Также разрабатывается ряд программных комплексов на основе сетей Петри, дополненных конструкциями объектно-ориентированного программирования.

В то же время добавление определенных языковых конструкций часто производится на основе требований приложений без определения их формальной семантики.

Цель работы: Разработать метод моделирования программного обеспечения управления умного дома на основе сетей Петри.

Требуется решить следующие задачи:

1. Провести анализ характеристик сети Петри, характеристик компонентов системы умного дома;
2. Формулировать модели ПО на основе сетей Петри;
3. Разработать алгоритм управления умного дома;
4. Разработать структуру ПО метода;
5. Провести исследование метода.

1. Аналитическая часть

В данном разделе будет провести рассмотрение типов сетей Петри а также моделирование на основе вложенных сетей Петри. Проанализировать характеристики компонентов системы умного дома

1.1.Классические сети Петри

Сети Петри — аппарат для моделирования динамических дискретных систем (преимущественно асинхронных параллельных процессов).

1.1.1. Понятие сети Петри

Сеть Петри определяется как четверка **<P, T, I, O>**

где P = {p1, p2, …, pn} — конечные множества позиций,

T = {t1, t2, …, tm} — конечные множества переходов,

I и O — множества входных и выходных функций (отображение из переходов в комплекты позиций).

C = (P, T, I, O)

P = {p1, p2, p3, p4, p5}

T = {t1, t2, t3, t4}

X	T1	T2	T3	T4
I(x)	{p1}	{p2, p3, p5}	{p3}	{p4}
O(x)	{p2, p3, p5}	{p5}	{p4}	{p2, p3}

1.1.2. Графы Сети Петри

Сеть Петри представляет собой двудольный ориентированный граф, в котором позициям соответствуют вершины, изображаемые кружками, а переходам — вершины, изображаемые утолщенными черточками; функциям I соответствуют дуги, направленные от позиций к переходам, а функциям O — от переходов к позициям.

На рисунке 1 показана сеть Петри , соответствует пример 1.

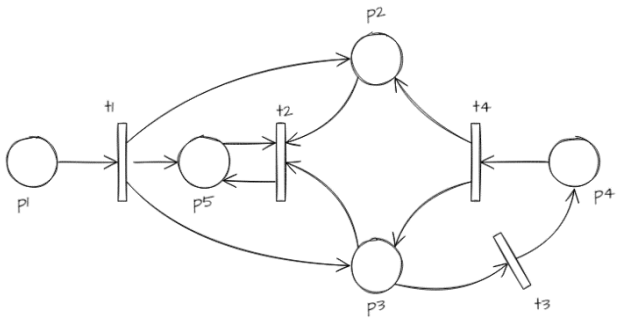


Рисунок 1 сеть Петри

Позиция, у которых нет входящих дуг, называются входными.

Позиция, у которых нет исходящих дуг, называются выходными.

1.1.3. Маркировка

Каждая позиция сети Петри может содержать ноль или более маркеров. Все маркеры считаются одинаковыми и неотличимыми друг от друга. Распределение маркеров по позициям называют маркировкой (μ). Маркировка и может быть также определена как n-вектор $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$, $n = |P|$.

На рисунке 2 показан пример сеть Петри с маркировкой μ = (0, 2, 0, 5, 1).

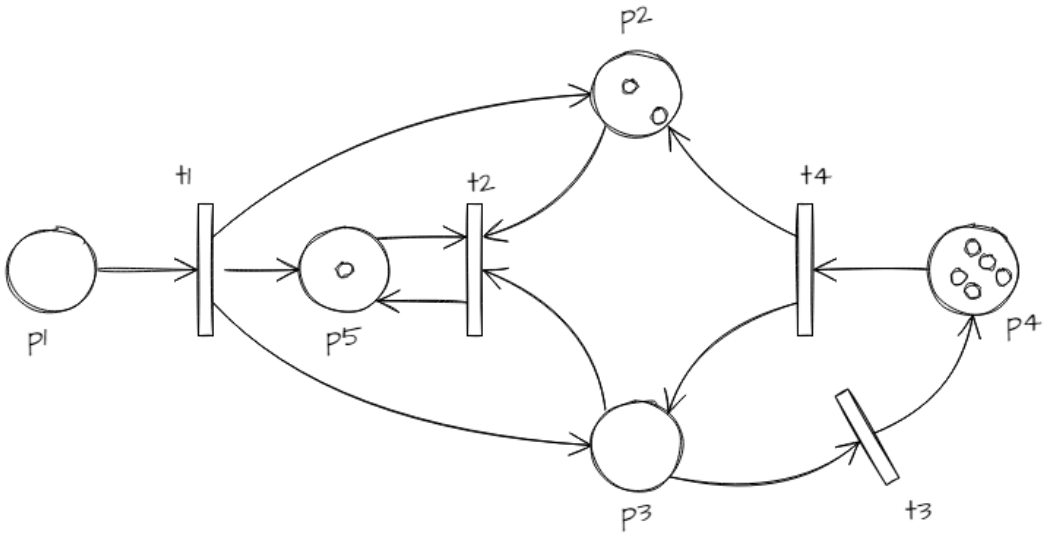


Рисунок 2 сеть Петри с маркировкой μ = (0, 2, 0, 5, 1)

1.1.4. Правила выполнения сетей Петри

Выполнением сети петри управляют количество и распределение маркеров в сети.

Переход запускается удалением маркеров из его входных позиций и образованием новых маркеров, помещаемых в его выходные позиции.

Перенос маркеров выполняется по следующей схеме:

- Переход t является ***активным***, если каждая его входная позиция содержит по крайней мере одну метку (более точно — по одной метке на каждую входящую в этот переход дугу);
- Активный переход может сработать, при срабатывании переход поглощает по одной метке с каждой своей входной позиции и размещает по одной метке на каждая свая выходная позиция (по одной метке на каждую исходящую дугу);
- В каждый момент времени для срабатывания из всех активных переходов недетерминированным образом выбирается один. Если активных переходов нет, то работа сети на этом завершается;

Последовательность переходов σ, в которой i-ый переход - сработавший на i-ом шаге работы сети, называется последовательностью срабатываний сети Петри.

Последовательность срабатываний однозначно определяет последовательность маркировки μi , где μ0 является начальной маркировкой после срабатывания t-го перехода, маркировка μ преобразуется в маркировку μ', будем обозначать кратко: $\mu \xrightarrow{t} \mu'$.

На рисунке 3 показан пример работы сети Петри для последовательности срабатываний σ = [t1, t3].

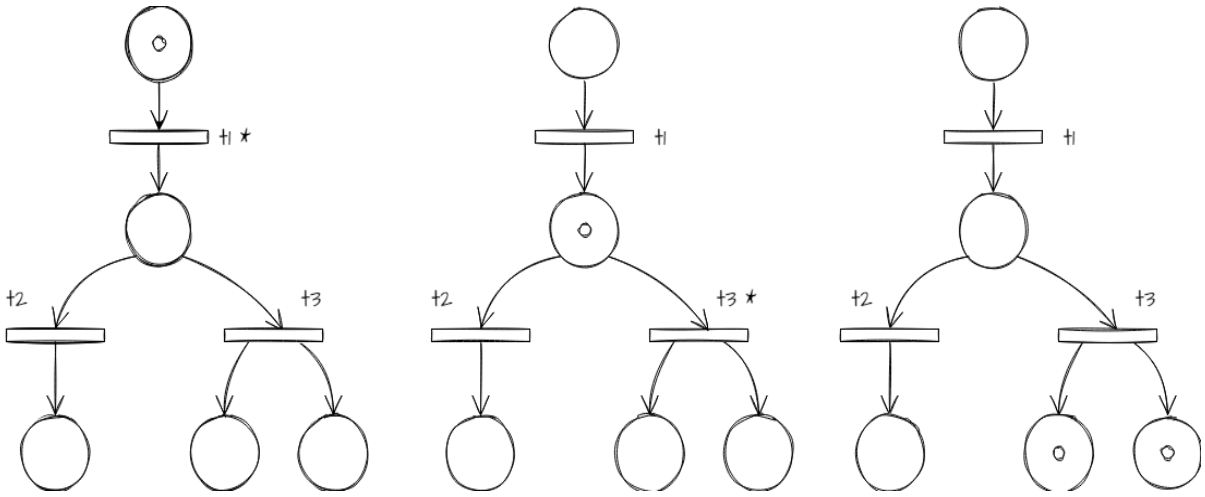


Рисунок 3. работы сети Петри для σ = [t1, t3].

1.1.5. Пространство состояний сети Петри

Состояние сети Гетри определяется ее маркировкой. Запуск перехода изменяет состояние сети Петри посредством изменения маркировки сети.

Пространство состояний сети Петри, обладающей i позициями, есть множество всех маркировок. Изменение в состоянии, вызванно запуском перехода, определяется функцией изменения, которую мы назовем функцией следующего состояния. Когда эта функция применяется к маркировке p (состоянию) и переходу to она образует новую маркировку (состояние), которая получается при запуске перехода i, в маркировке p. Так как i, может быть запущен только в том случае, когда он разрешен, то функция не определена, если t не разрешен в маркировке p. Если же 1, разрешен, то б(p, 1, u", где p" есть маркиров ка, полученная в результате удаления фишек из входов 1; и добав ленння фишек в выходы 1.

Множество достижимых Маркировка R : наименьшее множество маркировок, удовлетворяет двум условиям:

1. $M \in R(\mu)$

2.Если $\mu' \in R(\mu)$, для некоторого перехода t: $\mu' \xrightarrow{T} \mu''$ То $\mu'' \in R(\mu)$

1.1.6. Некоторые свойства сетей Петри

Безопасность: СП называется безопасной если все позиции сети в безопасности. позиция в сети безопасен, если количество фишек в ней никогда не превышает 1.

K-ограниченность: СП называется k-ограниченным если все его позиции k-ограничены. Позиция является k-ограниченным, если количество фишек в нем не может превышает целое число k. СП безопасная - 1-ограниченная.

$$\forall \mu \in M, \forall p \in P: \mu(p) \leq k \tag{1}$$

СП называется ***строго сохраняющим***, если общее количество фишек в сети остается постоянным.

$$\forall \mu_1, \mu_2 \in M, \sum_{i=1}^{n} \mu_1(p_i) = \sum_{i=1}^{n} \mu_2(p_i) \tag{2}$$

Живая сеть петри: если все его переходы живы.

Переход называется живым, если из любого состояния, достижимого из начального, возможен переход в любое другое достижимое состояние. Достижимость сп заключается в возможности достижения заданных маркировок.

$$\forall \mu \in M, \exists \mu' \in M, \mu \xrightarrow{t} \mu' \tag{3}$$

Перегруженность: маркировка μ, достижаемая из маркировки μ_0, считается мертвым, если нет перехода, который можно запустить, начиная с μ.

$$\forall \mu \in M, \nexists t \in T, \mu \xrightarrow{t} \mu', \mu' \in M \tag{4}$$

Обратимость: СП называется обратимой если каждая маркировка μ получается из μ_0, то μ_0 также получается из μ.

$$\forall \mu \in R(\mu_0), \exists t \in T: \mu \xrightarrow{t} \mu_0 \text{ (или } \mu_0 \in R(\mu)) \quad (5)$$

1.2.Параллельные процессы

Более адектватной во многих случаях являются интерпретации сетей Петри, в которых процессы могут обрабатываться одновременно. В стандартной модели такой сети предполагается, что время срабатывания всех переходов является одинаковым.

Конфликтом в сети Петри называется ситуация, когда сразу несколько активных переходов претендуют на одну метку некоторого места. При последовательном срабатывании переходов конфликты никак не учитываются, однако при параллельной интерпретации требуется некоторый способ их разрешения.

Пример конфликта показан на рисунке 0.4, на котором переходы t1 и t2 конфликтуют из-за общей метки в месте p.

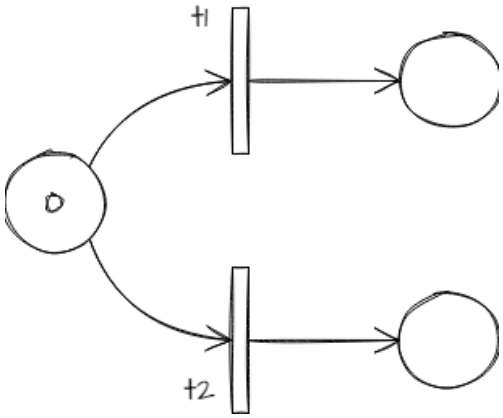


Рисунок 4. Пример конфликта

Стандартная схема параллельной обработки переходов в сетях Петри выглядит следующим образом: из всех активных в данный момент времени переходов выбирается некоторое их бесконфликтное подмножество (никакие два из выбранных переходов не имеют взаимного конфликта); все эти переходы срабатывают одновременно. Как и выше, если активных переходов нет, то сеть завершает свою работу.

1.3. Расширения сетей Петри

В настоящее время имеется большое количество различных вариаций сетей Петри, мы рассмотрим несколько наиболее стандартных расширений — сети с приоритетами, ингибиторные, цветные и временные сети Петри.

1.3.1. Сети с приоритетом

Сеть Петри с приоритетами — это стандартная сеть Петри, каждому переходу t которой поставлено в соответствие некоторое число Prt, называемое приоритетом этого перехода. Приоритеты используются для более полного управления разрешением конфликтных ситуаций. Если два перехода t1 и t2 конфликтуют из-за некоторого общего ресурса, то преимущество получит тот, который имеет больший приоритет. В частности, если все переходы данной сети имеют разные приоритеты, то конфликтов в такой сети вообще не будет. На практике, однако, достаточно определить только несколько приоритетов для потенциально конфликтных переходов.

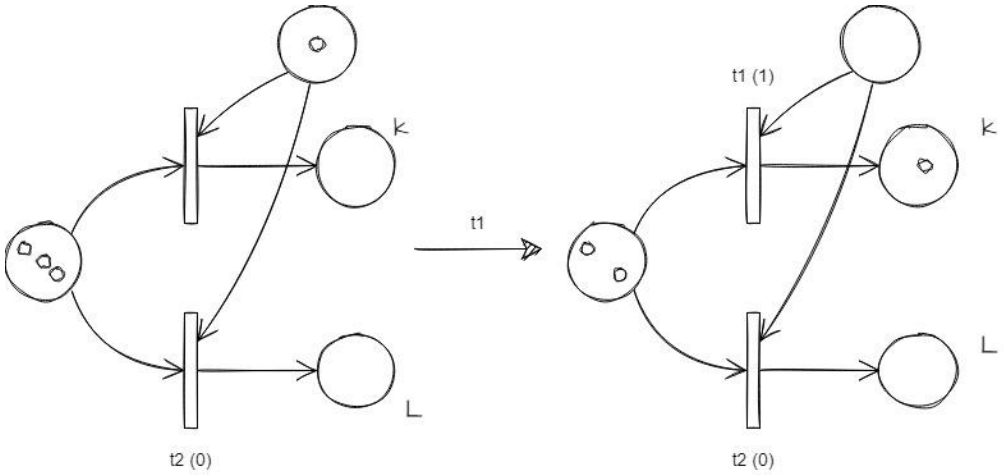


Рисунок 5 Представление сети Петри с приоритетом

1.3.2. Ингибиторные сети

В ингибиторных сетях Петри к стандартным дугам, ведущим из мест в переходы, добавляется специальный вид ингибиторных (тормозящих) дуг. Такая дуга, при наличии в соответствующем месте хотя бы одной метки, препятствует

активации соответствующего перехода. Поэтому, такой переход будет активизирован только тогда, когда в данном месте закончатся все метки. Таким образом, тормозящие дуги позволяют явным образом выполнять проверку на отсутствие меток в заданном месте.

На рисунке 6 показана реализация логических операций с использованием (пустое место соответствует логическому нулю, а место с одной меткой — логической единице).

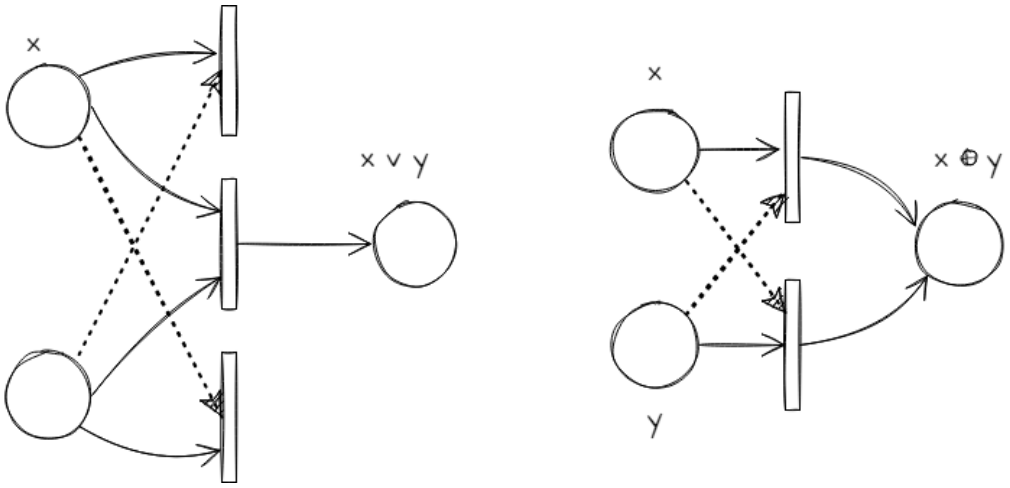


Рисунок 6 Реализация логических операций с использованием

1.3.3. Цветные сети Петри

В традиционных сетях Петри все метки по определению являются одинаковыми, следовательно, неразличимыми. Однако в тех системах, которые моделируются сетями Петри, метки часто представляют собой различные объекты. В цветных сетях Петри каждая метка имеет свое значение (цвет), что дает возможность отличать одни метки от других. Значение метки может быть простым, или любого сколь угодно сложного типа.

Переходы в цветных сетях Петри определяют отношения между значениями входных меток и выходных меток. Для этого входным дугам каждого перехода приписываются предусловия, которые определяют, метки с какими значениями поглощаются данным переходом. Выходные дуги перехода определяют (с помощью выражений) значения меток, которые будут помещены в соответствующие места.

1.3.4. Временные сети Петри

Для моделирования систем, в которых отдельные подпроцессы имеют различную продолжительность, можно использовать временные сети Петри. В таких сетях каждая метка получает дополнительный атрибут - время задержки.

Задержки, которые назначаются меткам, приписываются дугам, ведущим от переходов. Соответственно, все метки, поставляемые по какой-либо дуге, будут получать одну и ту же задержку, которая приписана этой дуге.

1.4. Сети Петри и параллельные вычисления

Сети Петри по своему определению обладают встроенным параллелизмом, поэтому они традиционно используются для моделирования разнообразных параллельных процессов.

Стандартная интерпретация сетей Петри с точки зрения параллельных вычислительных процессов заключается в том, что метки представляют собой данные, места - память, в которых хранятся данные, а переходы - подпрограммы, которые обрабатывают эти данные. Каждая подпрограмма ожидает, когда будут готовы все ее входные данные, после чего производит необходимые вычисления и помещает результаты на свои выходные места. Если какие-либо два перехода не конфликтуют из-за данных, то при некоторых условиях соответствующие им подпрограммы могут быть выполнены одновременно, т. е. параллельно.

На рисунке 7 показана сеть Петри, моделирующая две стандартные операции fork и join для управления параллельными процессами.

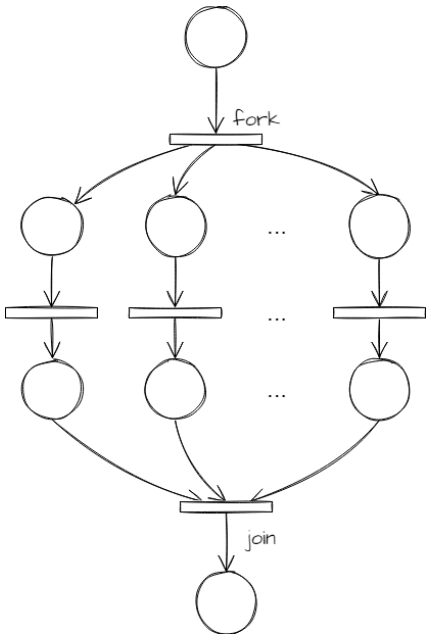


Рисунок 7 Моделирование операций fork и join сетью Петри

Так же легко сети Петри позволяют моделировать и конвейерную обработку данных.

Для организации взаимодействия параллельных процессов применяются различные схемы и механизмы синхронизации, критические секции, семафоры, операции обмена и т. д., которые относительно просто и наглядно моделируются с помощью сетей Петри.

На рисунке 8 показан фрагмент сети Петри, который реализует механизм взаимного исключения, применяемый для обеспечения корректного доступа нескольких процессов к одному разделяемому ресурсу. В качестве такого ресурса может выступать какая-нибудь структура данных (например, файл) или устройство (принтер). Часть процесса, которая используется для доступа и модификации разделяемого объекта, называется критической секцией. Выполнение процессом критической секции должно блокировать выполнение другими процессами своих критических секций, связанных с тем же разделяемым объектом.

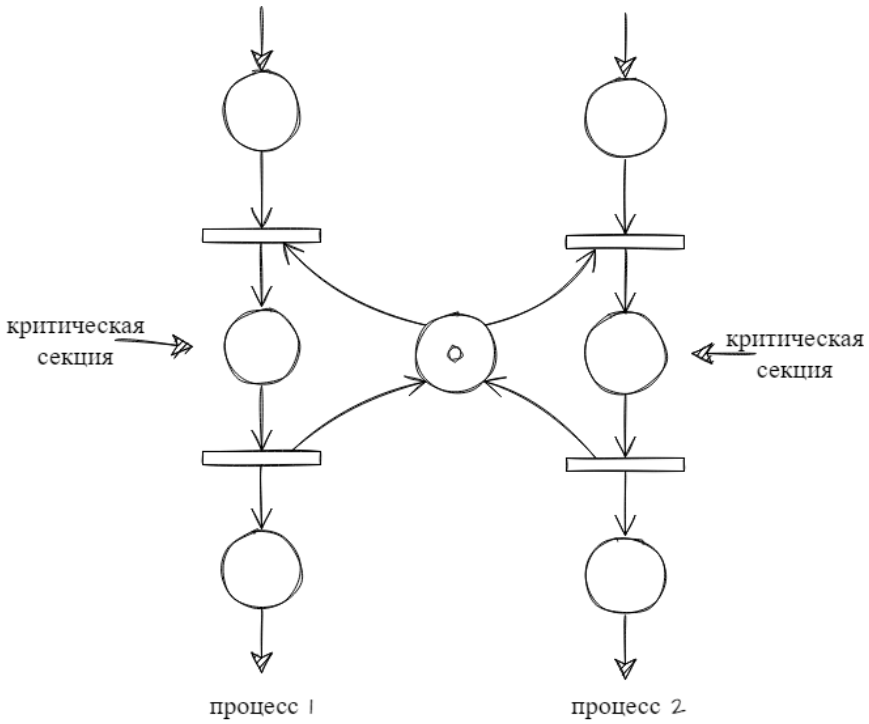


Рисунок 8. Моделирование процесса взаимного исключения

1.5. Вложенные сети Петри

Во вложенных сетях, фишки в позициях сети сами могут быть сетями Петри.

Вложенная сеть состоит из системной сети и элементных сетей, представляющих сетевые фишки.

1.5.1. Элементарные сети Петри.

В элементарных сетях Петри разрешаются только разметки, в которых каждая позиция содержит не более одной фишки. Соответственно, срабатывание перехода может перемещать из некоторой позиции или добавлять в некоторую позицию не более одной фишки. Поэтому веса всех дуг в элементарной сети одинаковы и равны 1.

В позиции p может быть маркер, а может и не быть. Следовательно, позиция p можно рассматривать как некоторое условие. Условие может быть истинным (позиция p содержит фишку) или ложно (позиция p пуста).

Элементарные сети Петри обладают свойствами Безопасности.

1.5.2. Сети Петри высокого уровня.

Сети Петри высокого уровня характеризуются следующими особенностями:

- разметка сети задается с помощью индивидуальных, т. е. Различимых между собой фишек;
- позиции могут содержать кортежи (n-ки) индивидуальных фишек; размер кортежа определяется арностью позиции;
- переходы могут срабатывать в различных режимах, удаляя фишки из одних позиций и добавляя их в другие, при этом единственным априорным ограничением является требование локальности, т. Е. В любом режиме переход может удалять фишки только из своих входных позиций и добавлять только в выходные позиции.

В сетях Петри высокого уровня дугам приписываются выражения, содержащие переменные. Различные режимы срабатывания переходов задаются различными означиваниями переменных в этих выражениях. При этом переменной в качестве значения приписывается фишка некоторого типа, а значением выражения является мультимножество фишек.

Сетью Петри высокого уровня называется набор

BHL = (P, T, I, O, D, Ф, W, K, M0),

Где

P - множество позиций (при каждой позиции $p \in P$ сопоставлено натуральное число $n \geq 1$, называемое арностью позиции p);

T - множество переходов;

I и O - множества входных и выходных функций (отображение из переходов в комплекты позиций);

D - функция, сопоставляющая каждой позиции $p \in P$ некоторый непустой домен D(p);

$M : P \rightarrow M(D(p))$ – маркировка;

$K : P \rightarrow M(D(p))$ — функция емкости позиций;

Ф — функция, сопоставляющая каждому переходу t некоторое непустое множество режимов срабатывания Ф(t);

W(p,t) ∈ M(D(p)×Ф(t)) - мультимножество фишек, удаляемых из позиции p;

W(t,p) ∈ M(Ф(t) × D(p)) - мультимножество фишек, добавляемых в позицию p;

M0 - начальная разметка.

На рисунке 9 показана сеть Петри высокого уровня, с 3 типами фишек A, B, C.

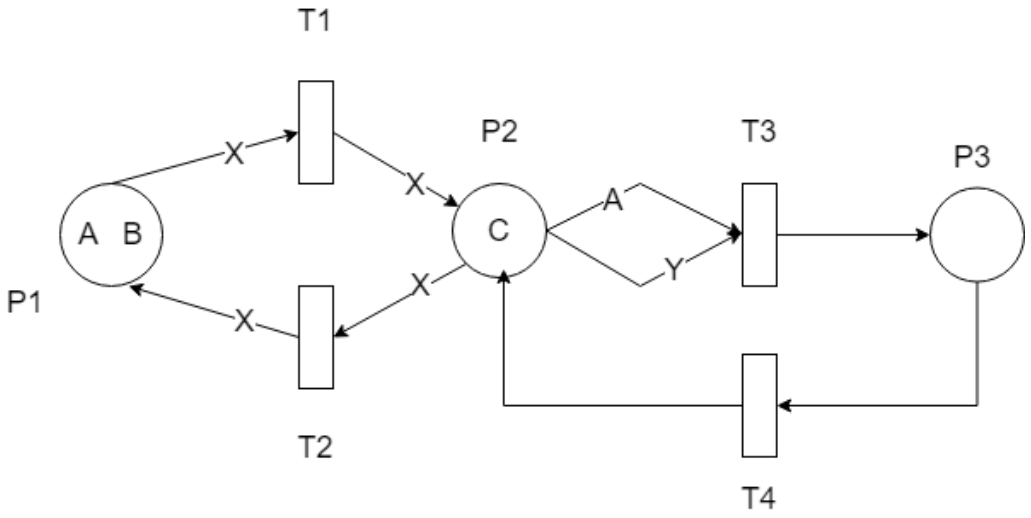


Рисунок 9. сеть Петри высокого уровня

Переменная x может принимать любое из значений A, B и C. Три возможных означивания переменной x задают три режима срабатывания для каждого из переходов t1 и t2. выражение на дуге от P2 к T3 заменено на выражение A. Это означает, что T3 не будет активирован, пока в P2 не будет фишки A.

1.5.3. Вложенные сети Петри

Вложенная сеть Петри - это сеть Петри высокого уровня (системная сеть*)*, где каждая фишка представляет собой маркированная элементная сеть Петри (сетевая фишка).

Во вложенных сетях предусмотрен механизм синхронизации срабатывания переходов в системной и элементных сетях.

Два вида синхронизации:

- Вертикальная, когда переход в системной сети срабатывает одновременно с некоторыми переходами в элементных сетях;
- Горизонтальная, когда одновременно срабатывают два перехода в элементных сетях (находящихся в одной и той же позиции системной Сети).

Для этого некоторые переходы системной сети помечены специальными метками. Переходы в элементной сети, которые должны срабатывать одновременно с переходом с меткой M в системной сети, помечены дополнительной к M меткой $\bar{M}$;

Var = {v, . . .} — множество имен переменных.

Con_{atom} - множество атомарных констант.

Con_{net} – множество сетевых констант.

$Con = Con_{atom} \cap Con_{net}$ - множество имен констант.

A_{net} - множество атомарных фишек.

Язык выражений L(Atom) над множеством атомов Atom определим следующим образом:

- atom ∈ Atom является выражением в L(Atom);
- Если atom1, . . . , atomn ∈ Atom, то кортеж (atom1, . . . , atomn) является выражением размерности n в L(Atom);
- Если e1, e2 ∈ Expr(Atom) — выражения одинаковой размерности n, то (e1+ e2) является выражением размерности n в Expr(Atom).

Двухуровневой вложенной сетью Петри (NPN) называется набор:

NPN = (Atom, Lab, SN,(EN1, . . . , ENk),Λ),

Где

Atom = Var ∪ Con — множество атомов;

$Lab = Lab_v \cup Lab_h$ — множество меток (Lab_v для вертикальной, Lab_h - для горизонтальной);

EN1, . . . , ENk - элементные сети вложенной сети NPN;

SN = (N,L, U, W, M0) - сеть Петри высокого уровня;

L - язык выражений;

U = (A, I) - модель языка L;

$A = A_{net} \cup A_{atom;}$

Функция I: Con →A - задает некоторую интерпретацию констант языка;

Λ - частичная функция пометки переходов, помечающая некоторые переходы системной сети метками из Labv и некоторые переходы в элементных сетях метками из множества Lab.

Разметка вложенной сети определяется как разметка ее системной сети.

Переход t сети SN является активным при некоторой разметке M и означивании b в том и только том случае если всякая входная для t позиция p содержит мультимножество, являющееся значением приписанного соответствующей входной дуге выражения W(p,t).

На рисунке 10-11 показана вложенная сеть петри с системными (SN) и элементными сетями (EN).

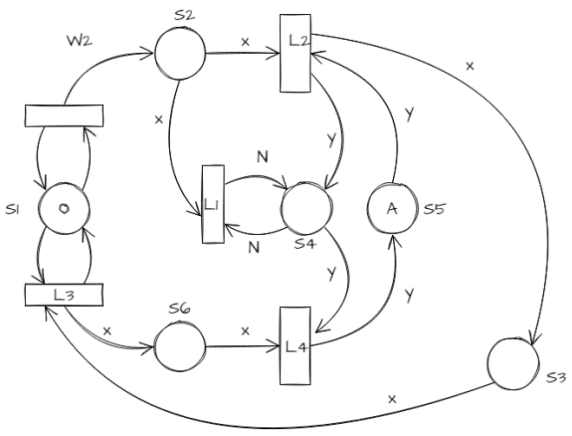


Рисунок 10. Системная сеть (SN)

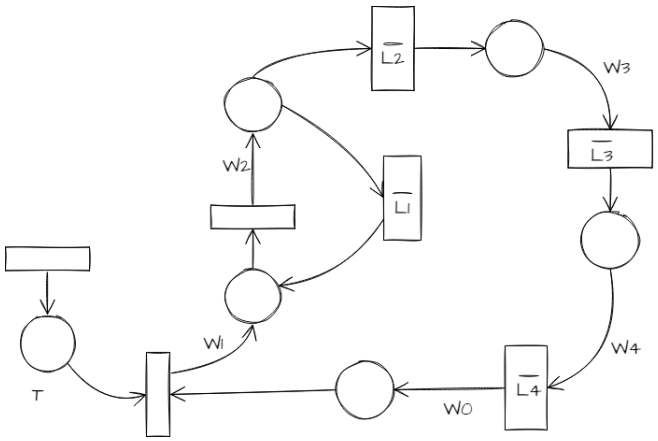


Рисунок 11. Элементная сеть EN

Для вложенной сети петри определяются следующие четыре вида шагов срабатывания:

1. ***Системно-автономный шаг:***

Если t — непомеченный переход системной сети и является активным при разметке M и означивании b, то срабатывание перехода t в сети SN называется системно-автономным шагом.

2. ***Элементно-автономный шаг:***

Пусть (α1, . . . , αn) ∈ M(p) — кортеж фишек, находящийся в позиции p системной сети при разметке M, αi = (EN, m) — сетевая фишка в этом кортеже, тогда локальное срабатывание непомеченного активного перехода t в сетевой фишке αi, при этом сеть EN остается в той же позиции системной сети SN, называется элементно-автономным шагом.

3. ***Шаг горизонтальной синхронизации:***

Пусть t1, t2 - дополнительные активные переходы в сетевых фишках αi и αj соответственно. Одновременное локальное срабатывание двух переходов t1 и t2, при этом сети EN1 и EN2 остаются в той же позиции системной сети SN, называется шагом горизонтальной синхронизации.

4. ***Шаг вертикальной синхронизации***

Одновременное срабатывание перехода t в системной сети SN и дополнительных к нему переходов в элементарной сети α1, . . . , αk, называется шагом вертикальной синхронизации.

1.5.4. Вложенные сети Петри с охранами на переходах:

Во вложенной сети Петри с охраной, наложить на переход соответствующее условие – охрану.

G : T → L — функция охраны, приписывающая каждому переходу t ∈ T некоторое выражение булевского типа.

Переход t может сработать только при истинном функции охраны. С помощью охраны можно накладывать условие как на структуру сетевой фишки, так и на ее разметку.

1.6. Анализ систем умного дома

Система умного дома должна включать следующие основные подсистемы:

- Подсистема климат-контроля;
- Подсистема освещения;
- Подсистема бытовой техники;
- Подсистемы безопасности и охраны.

Каждая из подсистем отвечает за мгновенную реакцию на срабатывание датчиков, сигнализирующую об изменении соответствующего входного параметра системы Умный Дом, с целью дальнейшей коррекции системы в заданной области (областях).

Система Умного дома также должна включать

- Удаленное управление умным домом;
- Внутренний модуль управления умным домом;
- Модуль центрального управления;
- Контроллеры подсистем Умного дома.

1.6.1. Подсистема климат-контроля

Подсистема климат-контроля управляет устройствами, предназначенными для обогрева, вентиляции, кондиционирования воздуха в помещениях, а также подогревом пола, в зависимости от времени суток и года.

Когда пользователя нет дома, система работает на минимальной мощности для поддержания климата в доме.

Когда пользователь приходит домой, система автоматически регулирует температуру, влажность и качество воздуха в соответствии с предпочтениями пользователя (по заданным параметрам).

1.6.2. Подсистема освещения

Система управления освещением позволяет регулировать яркость и яркость всех типов источников света, включая лампы накаливания, компактные люминесцентные, галогенные и светодиоды, до заданных уровней для достижения большей экономии энергии, обеспечения визуального интереса, повышения безопасности и создания настроения для определенных случаев. При управлении системой домашней автоматизации работа домашнего освещения может быть синхронизирована с другими подсистемами. Это дает еще большие преимущества; например, свет может включаться и выключаться в соответствии с настройками системы безопасности.

Лампа работает в режимах: “Вкл”, “Выкл”. Ее яркость может варьироваться на разных уровнях (например, очень низкая, низкая, нормальная, высокая, очень высокая).

В подсистеме освещения имеются светоизмерительные датчики. Когда датчик показывает, что уровень освещенности низкий, система посылает сигнал включения на лампы.

1.6.3. Подсистемы безопасности и охраны

Подсистема безопасности непрерывно контролирует и реагирует на все тревоги и сработки в доме, обеспечивает удаленный контроль за событиями в доме, контролирует пожарную безопасность и безопасность в сфере коммунальных домовых систем.

В систему безопасности входят охранные датчики и громкоговорители. Когда эти датчики обнаруживают злоумышленника, он отправит сигнал системе освещения и громкоговорителям для предупреждения, в дополнение к отправке уведомлений пользователям, которые были настроены заранее.

Кроме того, в системе также есть датчики пожарной сигнализации и спринклеры, когда датчик пожарной сигнализации обнаруживает возгорание (из-за повреждения электрооборудования, забыв выключить газовую плиту,...), система автоматически активирует спринклеры, при этом также отправка сообщений предопределенным пользователям.

1.7.Выводы из аналитического раздела

В данном разделе было рассмотрено типов сетей Петри а также их характеристики. Проанализированы характеристики компонентов системы умного дома.

2. Конструкторский раздел

В данном разделе будут рассмотрены требования к моделированию , описание компоненты системы умного дома и спроектирование взаимосвязь между ими

2.1. Требования к моделированию

Моделирование должно соответствовать следующим требованиям:

- Должен быть обеспечен комфорт пользователя;
- Свести к минимуму действия пользователя, или другими словами, максимально увеличить степень автоматизации умного дома;
- Максимально сократить потребляемую энергию.

2.2. Проектирование взаимосвязь между компонентами

Для представления взаимосвязи (синхронизации) между компонентами было предложено присваивать каждому переходу метку. На основе присвоенных меток можно узнать, какие переходы синхронизированы друг с другом.

По соглашению вертикальные синхронные переходы помечаются буквами «l» и «-l» (дополнительно). Эти типы меток означают, что они должны быть активированы одновременно.

Для некоторых переходов не требуется, чтобы все они были активны одновременно, но связь между ними представляет собой «если, то». Должен использовать другой метод присвоения меток для представления этой связи. Я предлагаю использовать следующий метод:

Переход T2 активен, если активен переход T1 (или если T1 то T2), то мы присваиваем переходу T2 метку со значением «*», добавляя метку перехода T1.

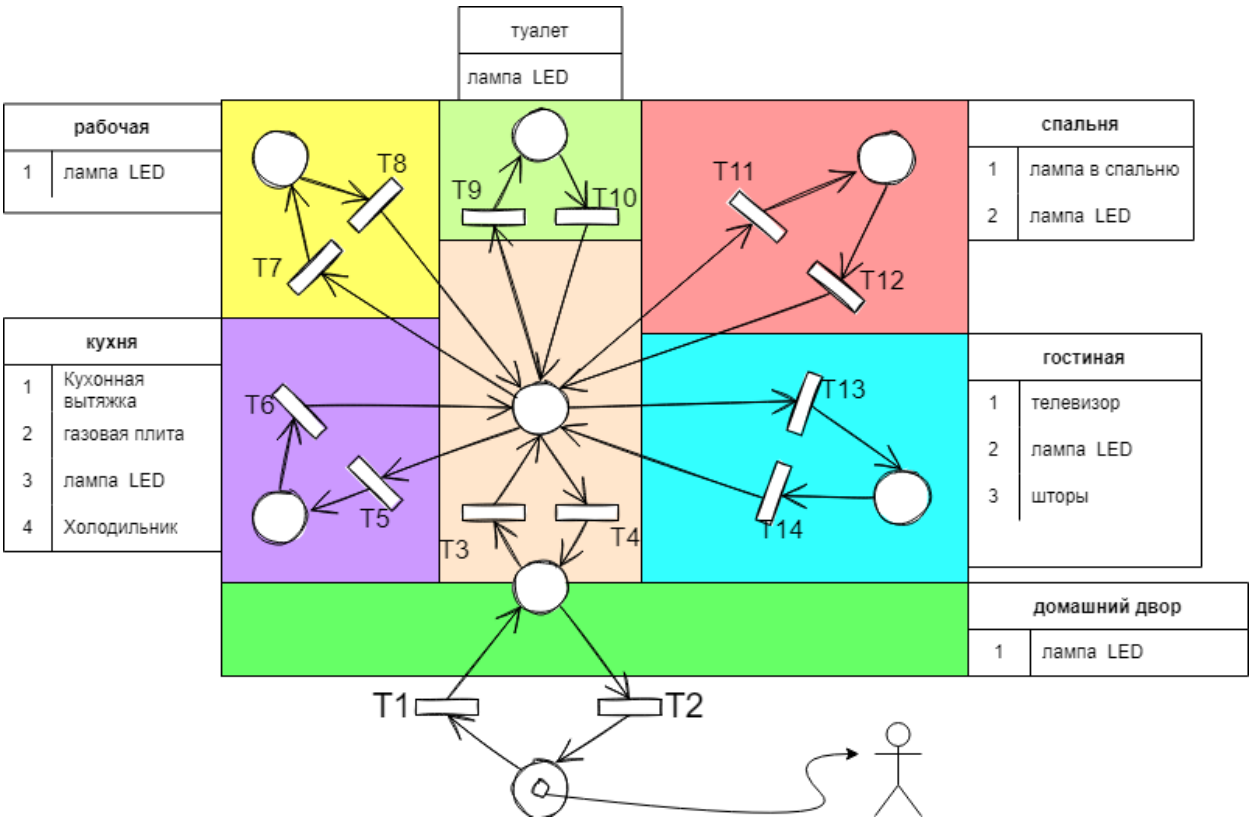


Рисунок 12. Сеть Петри, представляющая местоположение пользователя

В сеть Петри на рисунке 17 каждый жетон представляет одного человека. В начале пользователь отсутствует. Сценарий этой сети заключается в том, что пользователь вводит пароль (или сканирует магнитную карту), происходит аутентификация (как описано в сетке Петри для двери), а затем пользователь войдет в дом (в центральной позиции). Пользователь может перемещаться между комнатами (через центральную позицию)

2.3.Описание компонентов системы

В этом разделе формализации сетями Петри подсистем системы умного дома.

2.3.1. Описание системы освещения:

Система освещения включает лампы и шторы.

Простая лампа обладают двумя состояниями “включено” и “выключено”. Есть два перехода между ними.

На рисунке 12 сеть Петри для описания простой лампы.

P1 – “включено”, P2- “выключено”

T1 – включить, T2 - выключить

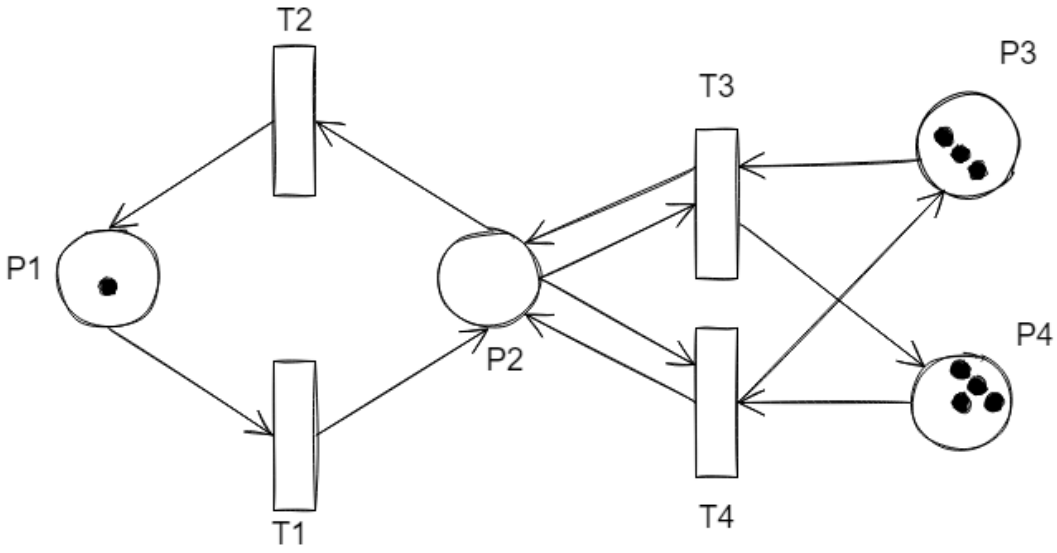


Рисунок 13. сеть Петри для описания простой лампы

Количество точек в местах P3 и P4 указывает количество уровней интенсивности света, которые можно увеличить (для P3) и уменьшить (для P4). Переходы T3 и T4 представляют действие увеличения и уменьшения соответственно, яркости лампочки. Увеличение (уменьшение) яркости выполняется только при включенной лампе и имеет возможность увеличения (уменьшения) яркости.

Шторы имеют два состояния: закрытые или открытые.

На рисунке 14 сеть Петри для шторы. S1 соответствует закрытому состоянию штор. S2 - открытое состояние.

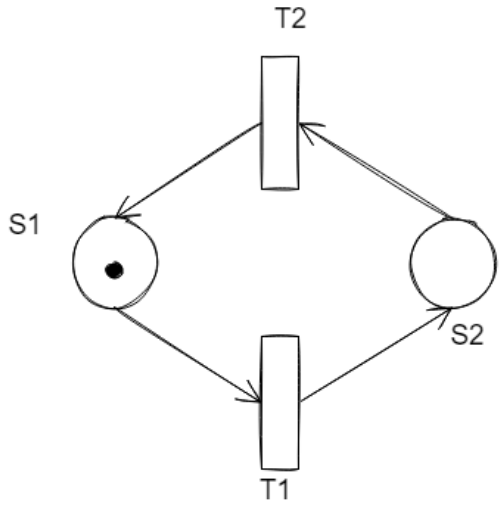


Рисунок 14. сеть Петри для шторы

2.3.2. Описание системы климат-контроля

На рисунке 15 сеть Петри для кондиционеров

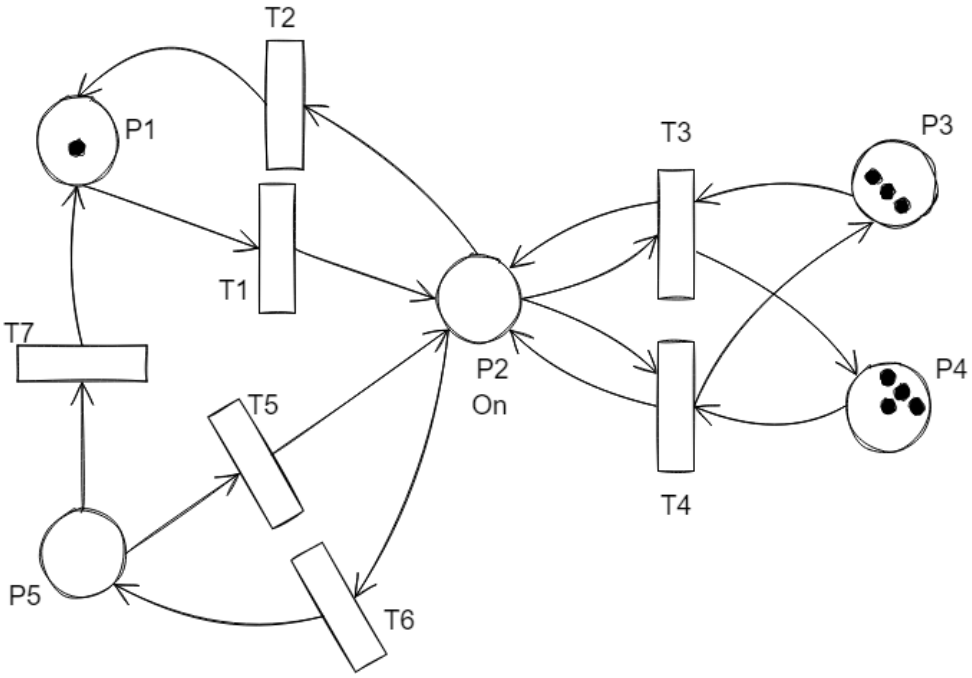


Рисунок 15. Сеть Петри для кондиционеров

Кондиционер может находиться в одном из трех состояний: “выключен”, “включен”, “ожидания”, соответствующие положениям P1, P2 и P5. Аналогично лампе, две позиции P3 и P4 имеют функцию управления увеличением или уменьшением мощности кондиционера.

Очиститель воздуха описывается простой сетью Петри, показанной на рисунке 16. Он находится в “выключенном” или “включенном” состоянии.

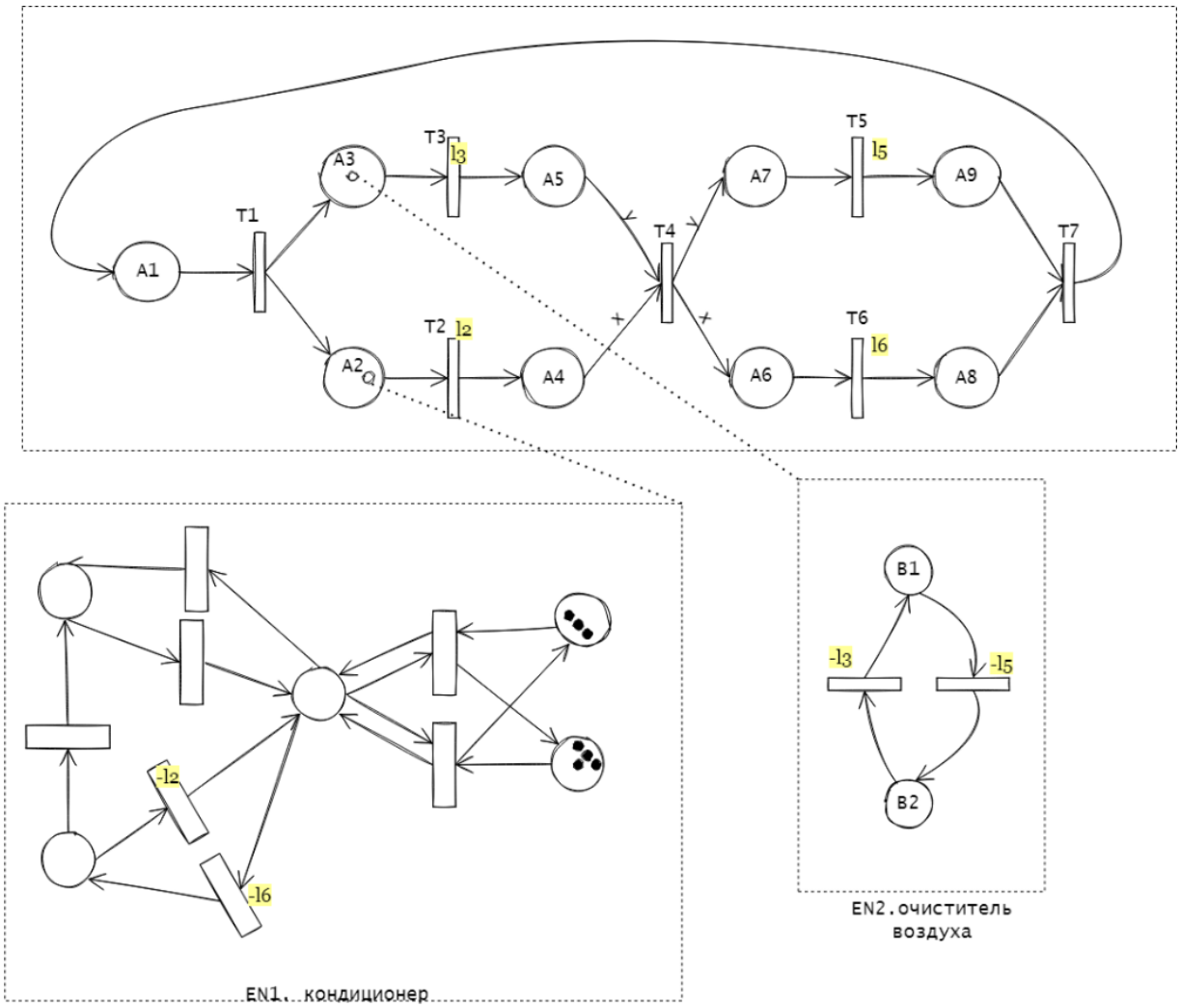


Рисунок 16. Сеть Петри для системы климат-контроля

Позиция	Переходы
A2 - содержит кондиционер	T1 - пользователь приходит
A3 - содержит очиститель воздуха	T2 - включить кондиционер
A4 - кондиционер включен	T3 - включить очиститель воздуха
A5 - очиститель воздуха включен	T4 - пользователь уходит
A6 - содержит кондиционер	T5 - выключить кондиционер
A7 - содержит очиститель воздуха	T6 - выключить очиститель воздуха
A8 - кондиционер в режиме ожидания	
A9 - очиститель воздуха выключен	

B1 - очиститель воздуха включен	
B2 - очиститель воздуха выключен	

Таблица 1. описание позиций/переходов сети Петри для системы климат-контроля

В этой сети Петри, когда пользователь приходит домой, активируется переход T1, что приводит к появлению токена сети в P2 и другого токена сети в P3. За этим следует активация переходов T2 и T3, где переходы помечены как l2 и l3 в сетевых токенах, активируются, в результате включается кондиционер и очиститель воздуха. На некоторых ребрах помечены x и y: x обозначает кондиционер, y обозначает очиститель воздуха.

2.3.3. Описание системы безопасности и охраны

Для описания двери, она может иметь следуюшие четыре состояния: заблокирована, разблокирована, открыта и закрыта.

На рисунке 15 показана сеть Петри для двери.

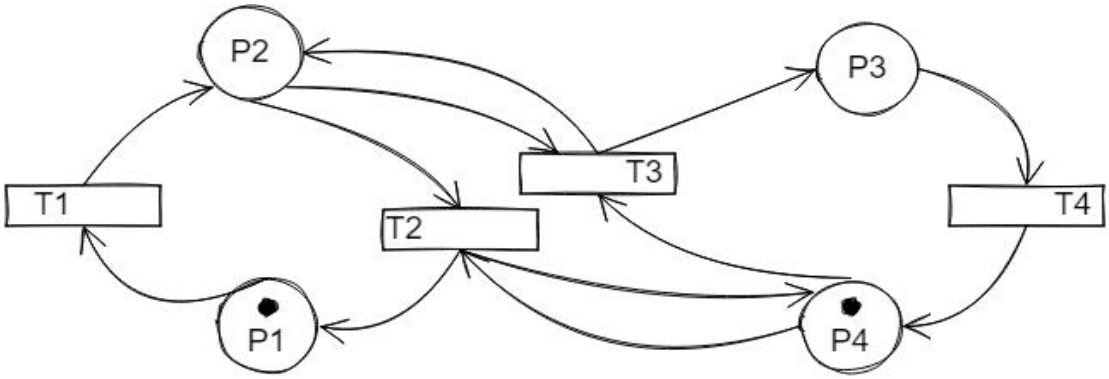


Рисунок 17. Сеть Петри для двери

Позиция	Переходы
P1 - дверь в заблокированном состоянии;	T1 - разблокировать дверь
P2 - дверь в разблокированном состоянии;	T2 - блокировать дверь
P3 - дверь открыта;	T4 - открыть дверь
P4 - дверь закрыта	T5 - закрыть дверь

Таблица 2. описание позиций/переходов сети Петри для двери

Когда разблокировать дверь (T1), точка из позиции P1 будет переведена в позицию P2, дверь будет разблокирована. Дверь разблокирована, его можно открыть или закрыть (с помощью перехода T3 или T4). Дверь блокируется только тогда, когда она закрыта, то есть ее нельзя заблокировать, пока она открыта.

Описание процесса аутентификации пользователя.

От пользователя требуется ввод пароля (либо сканирование магнитной карты, распознавание лица). Затем проверяем пароль, если пароль правильный, то посылаем сигнал на дверь (описано выше, соответствует переходу т1 на картинке), дверь открывается, пользователь помечается как действительный. И наоборот, если пароль неверный, пользователь может повторно ввести пароль. Если ввести неверный ввод больше указанного количества раз (по умолчанию 3 раза), то сигнал о взломе постороннего.

На рисунке 16 и таблице 2 показана сеть Петри для процесса аутентификации пользователя.

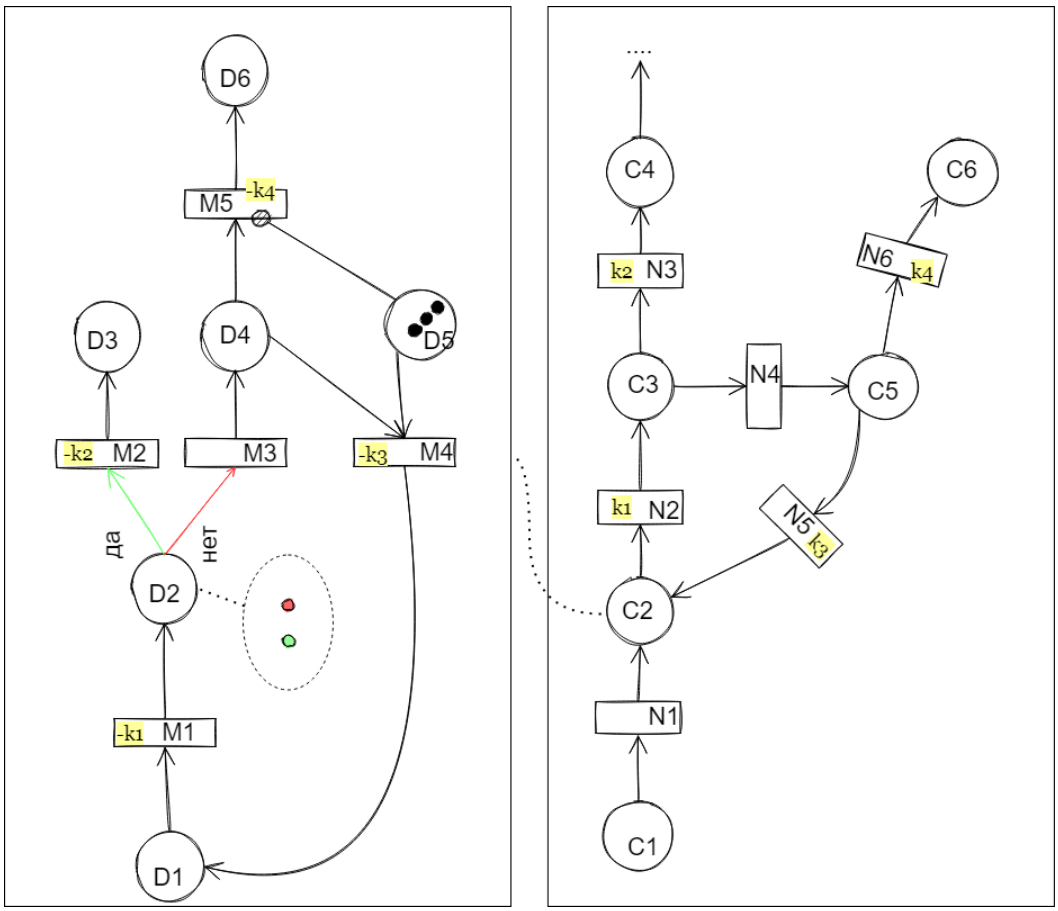


Рисунок 18. сеть Петри для процесса аутентификации пользователя

Позиция	Переходы
D1 - пароль	T1 - проверка пароля
D2 - результат проверки пассворка	T2 - пометить пользователя как действительного
D3 - пользователь ввел неправильный пароль	T4 - пометить пользователя как недействительного
D4 - пользователь недействителен	T5 - повторный ввод пароля
D5 - количество попыток	T6 - отправить уведомление
D7 - уведомление	N1 - пользователь приходит
C1 - неизвестный пользователь	N2 - процесс аутентификации
C2 - начинает аутентификацию	N3 - успешная аутентификация
C3 - завершает аутентификацию	N4 - аутентификация не удалась

C4- аутентифицированный пользователь	N5 - пользователь хочет повторить
C5 — пользователь не прошел аутентификацию	
C6 - нарушитель	

Таблица 3. описание позиций/переходов сети Петри для процесса аутентификации пользователя

Сеть Петри справа представляет пользователей. Сеть слева — это сетевой токен, который представляет собой процесс аутентификации пользователя.

При срабатывании перехода T1, на выходе P2 появится жетон зеленого или красного цвета (соответствующий истинности или ложности пароля). Каждый тип жетона будет иметь разное направление движения (в соответствии с цветовой меткой дуг и цветом жетона). Более того, дуга начинается с позиции M5 до перехода D5, отмечено стрелкой с кружком. Эта дуга отличается от обычного, он активен только тогда, когда в позиции D5 вообще нет жетонов. Это означает, что нет возможности повторить попытку ввода пароля.

2.4.Сеть Петри, описывающая пользователей

После процесса аутентификации, описанного в предыдущем, пользователь был аутентифицирован. Ниже приводится описание взаимодействия пользователя с домашними устройствами.

На рисунке 19 показано описание состояний пользователя.

C6 по C9 описывают состояния пользователя на кухне.

C6 - пользователь на кухне	N7/N8 - входит/выходит из кухни
C7 - пользователь готовит	N9/N10 начать/закончить приготовление
C8 - пользователь использует газовую плиту	N11/N12 - включить/выключить газовую плиту
C9 - пользователь ест еду	

Таблица 4. описание позиций/переходов сети Петри

Метки q1, q2 выполняют синхронизацию газовой плиты с вытяжкой. Вытяжка включается/выключается в зависимости от газовой плиты.

В спальне пользователь находится в двух состояниях: спит или бодрствует. Если пользователь находится в спящем состоянии, то выключить лампу и включить ночник. Когда пользователь просыпается, ночник автоматически выключается. Эти отношения представлены метками q3 и q4.

Позиция C14 указывает на то, что пользователь находится в режиме кино (затемнение, шторы закрыты). C15 - в режиме встречи (увеличение яркости). Для этого необходимо синхронизироваться с системой освещения.

В следующем абзаце будут описаны действия пользователей.

Как показано выше, каждому пользователю соответствует один токен в сети Петри, как показано на рисунке 19.

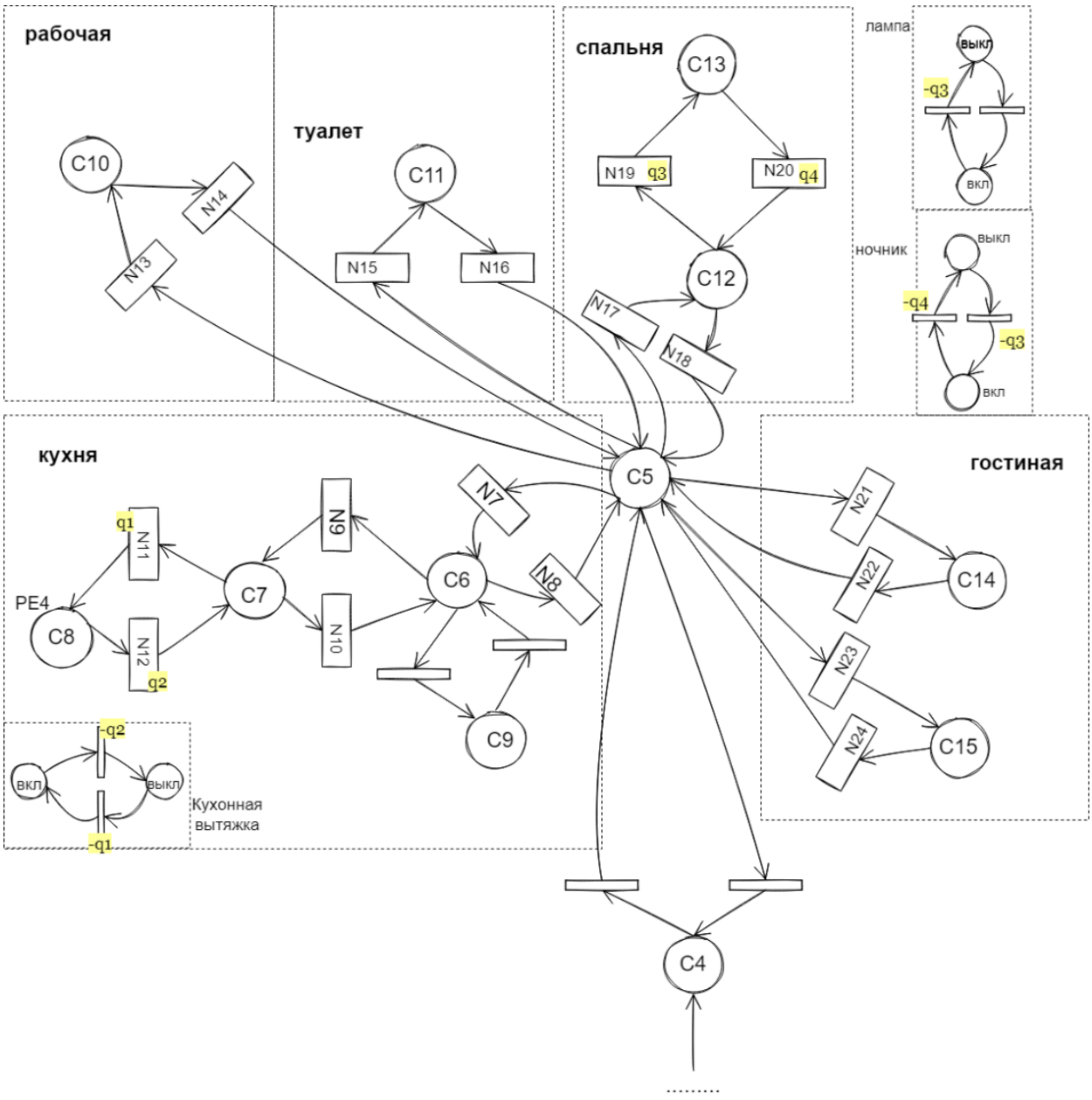


Рисунок 19. Сеть Петри для пользователей

Позиция	Переходы
P1 - неизвестный пользователь	T1 - пользователь вводит пароль
P2 - в процессе проверки	T2 — аутентификация не удалась
P3 - нарушитель	T3 — успешная аутентификация
P4 - подтвержденный пользователь	T4 - пользователь уходит из дома
P5 - пользователь находится в помещении	T5 - пользователь входит в дом
P6 - пользователь готовит	t6 — пользователь выходит из дома

P7 - пользователь смотрит фильм	T7 - пользователь начинает готовить
P8 - пользователь спит	T8 — пользователь заканчивает готовить

Таблица 5. описание позиций/переходов сети Петри для пользователей

3. Технологический раздел

В этом разделе осуществить выбор языка и среды среда реализации ПО. Описать формат входных и выходных данных и структуру разрабатываемого ПО. Провести функциональное тестирование, описать пользовательский интерфейс.

3.1.Выбор средств программной реализации

В связи с необходимостью совместного использования расширенных типов сетей Петри, языки (включая языки программирования и языки моделирования), доступные для моделирования сетей Петри, не являются удовлетворительными.

Следовательно, необходимо разработать модель для моделирования сетей Петри. Язык Python был выбран в качестве языка программирования, поскольку он обладает следующими преимуществами:

- язык программирования высокого уровня;
- с поддержкой объектно-ориентированного программирования;
- платформ независимый.

При выборе среды разработки рассматриваются и другие возможные среды, которые помогут автоматизировать процесс разработки. Для разработки данной программы необходимо обеспечение средой следующих возможностей:

- удобные инструменты для отладки и поиска ошибок, в случае их возникновения;
- разработка более гибкой и надежной программы путем обработки различных исключительных ситуаций, возникающих в результате некорректной работы программы;
- использование всплывающих подсказок во время написания кода программы, что обеспечивает значительное экономию времени и повышения уровня продуктивности.

Среда разработки PyCharm поддерживает все эти возможности, а также многие другие, такие как:

- инструменты для запуска тестов и анализа покрытия кода, включая поддержку всех популярных фреймворков для тестирования;
- умное автодополнение, инструменты для анализа качества кода, удобная навигация, расширенные рефакторинги и форматирование;
- окно Свойства для настройки свойств и событий элементов управления в пользовательском интерфейсе.

Среду разработки PyCharm стоит рассматривать как интеллектуальную среду разработки, понимающую код. В процессе его написания программистом она занимается построением синтаксического дерева, определением особенностей размещенных ссылок, анализом возможных путей исполнения операторов и передачи данных.

3.2.Структура разработанного ПО

ПО состоит из следующих основных модулей:

- модуль сети Петри;
- подсистема климат-контроля;
- подсистема освещения;
- подсистема бытовой техники;
- подсистемы безопасности;
- модуль умного дома.

Каждый модуль содержит классы, представляющие устройства, принадлежащие системе, описываемой модулем.

Модуль сети Петри содержит основные элементы, составляющие сеть Петри. К ним относятся: позиции, дуги, переходы.

В модуле освещения, включая некоторые ламы, и наличие дополнительного контроллера, управляющего ламы по заданному расписанию.

На рисунке 20 представлена схема взаимосвязей модулей ПО.

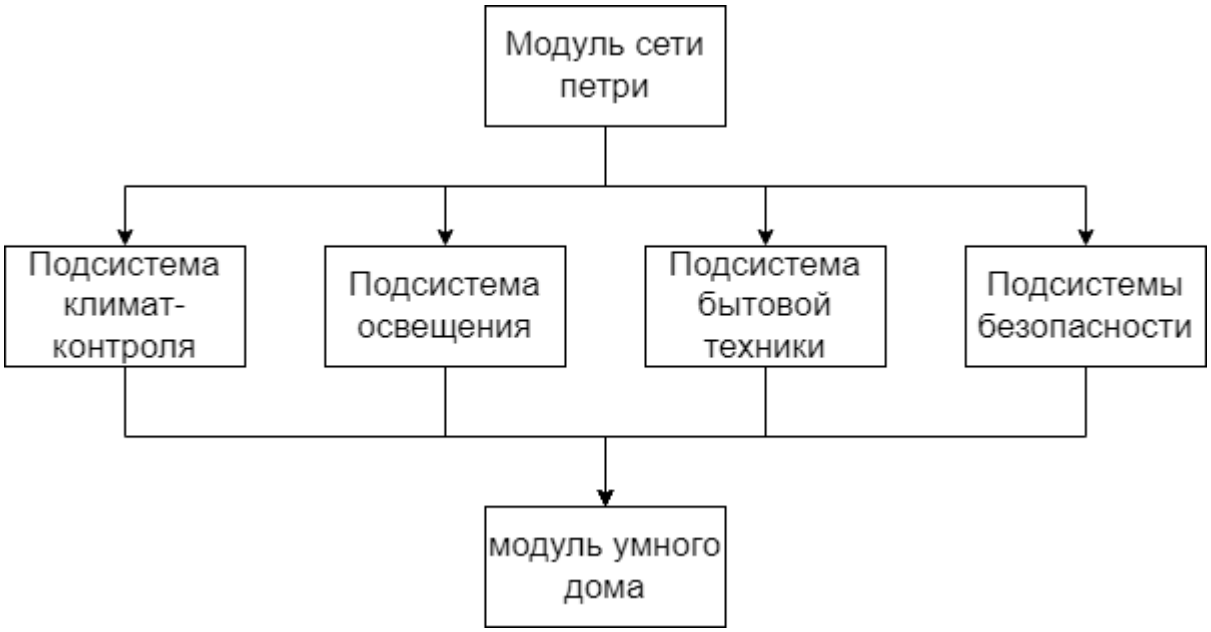


Рисунок 20. схема взаимосвязей модулей ПО

На рисунке 21 представлена диаграмма классов модуля сети Петри

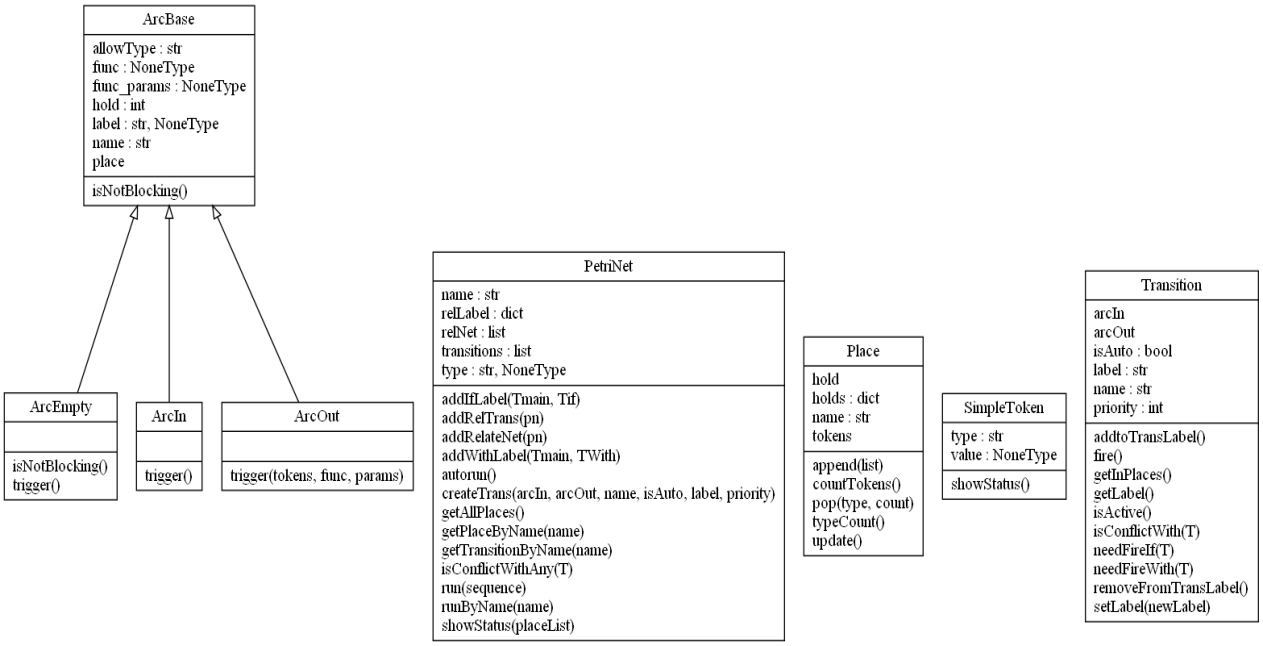


Рисунок 21. диаграмма классов модуля сети Петри

Класс singletoken используется для представления токена.

Класс Place используется для представления для должности. Он содержит токены (либо простые токены, либо сетевые токены), а также количество токенов каждого типа.

Абстрактный класс ArcBase используется для представления дуг. Он состоит из положения, начальной или конечной точки дуги. Класс также содержит дополнительную информацию, такую как тип и количество токенов, которые могут проходить через эту дугу, и дополнительно функцию обработчика токенов (при необходимости).

Класс ArcIn представляет собой дугу, которая войти в переход.

Класс ArcOut представляет собой дугу, которая выйти из перехода.

Класс ArcEmpty представляет торможеный дуг. Этот класс имеет метод isNotBLocking, в отличие от двух классов ArcIn и ArcOut, возвращает истину, если в его позиции нет токенов.

Класс Transition представляет переход. Он содержит входящие дуги и исходящие дуги. В целях синхронизации переходов класс также включает метку, которая присваивается переходам. Чтобы решить проблему конфликта, добавляется число, которое называется приоритетом перехода.

Сеть Петри представлена классом PetriNet. Он состоит из набора переходов. В классе также есть метод, активирующий один или несколько переходов (если они являются активированными).

Каждый объект (лампа, кондиционер, ...) будет содержать внутри себя сеть Петри, описывающую себя. Когда необходимо изменить состояние объекта или происходит событие, связанное с объектом, нам нужно только активировать один или несколько переходов соответственно.

3.3.Функциональное тестирование

Для каждого объекта содержит сеть Петри, активируем последовательные переходы, а затем сравниваем текущее состояние сети Петри с рассчитанным ранее.

Для проверки взаимодействия между объектами, делая то же самое, также активируем последовательные переходы, а затем сравниваем состояния связанных сетей Петри, с результирующими состояниями.

3.4.Выводы

В качестве языка программирования был выбран язык Python , в среде разработки pycharm. На этом основании, реализован программый метода моделирования. Было произведено тестирование работы системы.

4. Исследовательский раздел

В этом разделе исследовать эффективность разработанного программного обеспечения, а также синхронизацию и взаимодействие объектов в подсистемах умного дома.

4.1.Факторы окружающей среды

Из-за аппаратных ограничений среди факторов окружающей среды учитываются только температура окружающей среды и время захода солнца.

Изменение температуры обновляется ежечасно.

Влияние температуры окружающей среды на температуру в помещении рассчитывается по следующей формуле:

$$\Delta T = T_{\text{среды}} * \left(T_{\text{среды}} - T_{\text{помещении}}\right) * k * \Delta t \qquad (6)$$

Где

$T_{\text{среды}}$ - температура окружающей среды

$T_{\text{помещении}}$ - температура в помещении

k = 0.05 - коэффициент теплопередачи

Δt - промежуток времени

4.2.Параметры оборудования

Для системы освещения, всего было использовано 6 лампочек. Каждый имеет мощность 15 вт, с коэффициентом рассеивания тепла 0,3.

Теплоэкзотермическая мощность лампы рассчитывается по формуле:

$$\Delta W_{\text{л}} = W * k \qquad (7)$$

Где

W – мощность лампы

K - коэффициентом рассеивания

Для системы климат-контроля мы используем кондиционер с электрической мощностью 1500 Вт и мощностью охлаждения 745,5 Вт. При снижении энергопотребления по кубическому закону $y = x^2$. Холодопроизводительность кондиционера равна:

$$\Delta W_c = \frac{level}{100} * W * \Delta t \quad (8)$$

Где

Level - уровень работы кондиционера

W – мощность лампы

Δt - промежуток времени

4.3.Планирование действий пользователя

Действия пользователя планируются в порядке доступности, с разной продолжительностью, принимаемой по нормальному распределению.

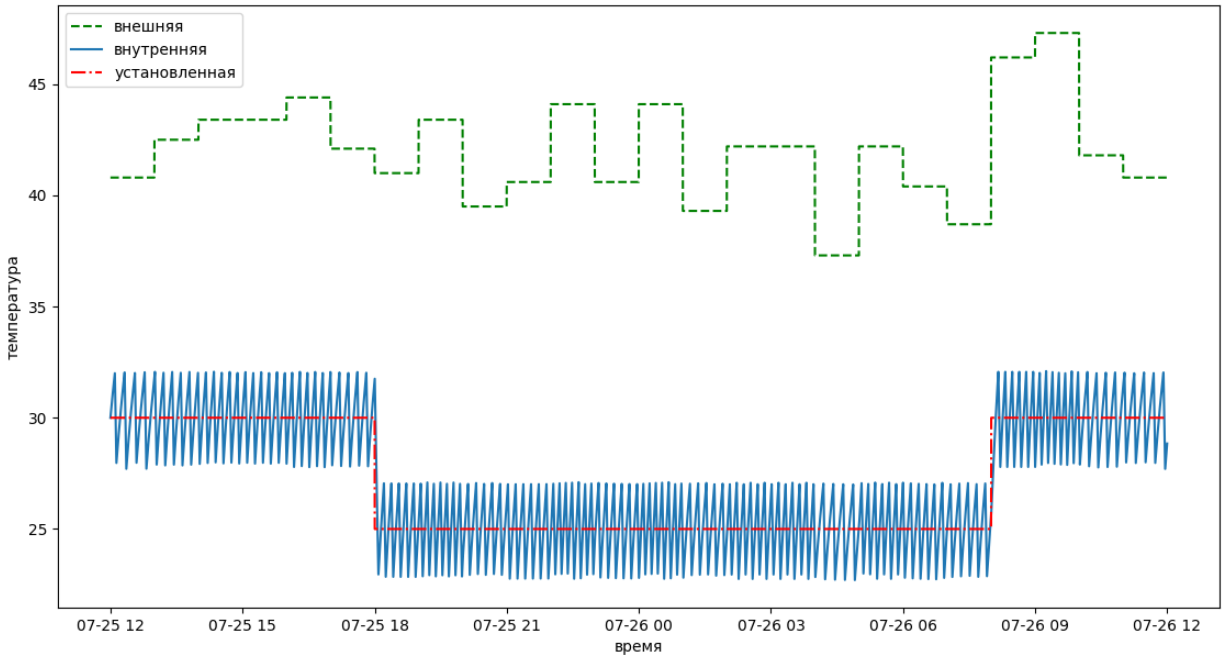
Действия	Продолжительность (мин)	
проснуется	5	3
пойдет в туалет	10	5
приготовит еды	30	10
завтркает	15	5
уидет на работу	8*60	30
воидет домой	0	0
переидет на гостиной	60	30
приготовит еды	60	30
ужинает	30	10
Перейти в рабочую комнату	3*60	30
пойти в туалет	15	10
Иди в спальню	0	0
Идти спать	8*60	60

Таблица 6. Действия пользователя

4.4.Результаты исследования

Моделирование выполняется с использованием Вкл/Выкл и ПИД-регулятора. Отсюда мы черпаем оптимальный метод использования энергии умного дома.

На рисунках представлены результаты сравнения внутренней и наружной температуры с использованием обоих типов регулятора.



,

Рисунок 22. температуры с использованием Вкл/Выкл-регулятора

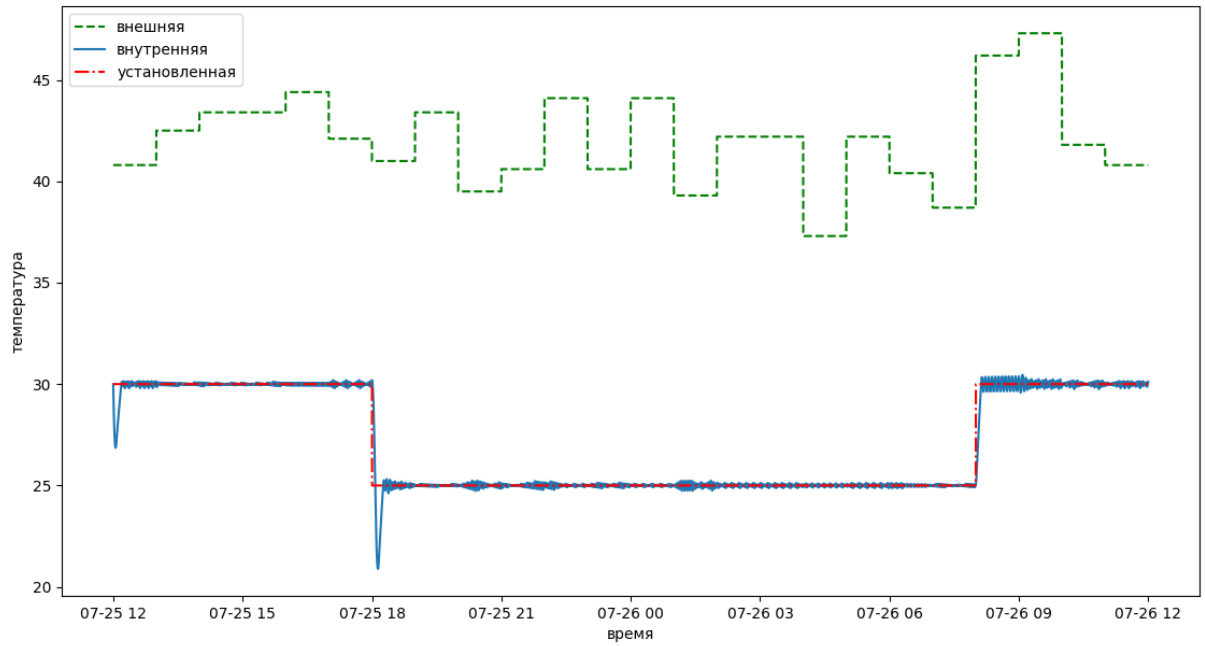


Рисунок 23. температуры с использованием ПИД-регулятора

Из двух приведенных выше рисунков видно, что использование ПИД-регулятора дает значительно меньшую ошибку температуры по сравнению с ошибкой, вызванной использованием вкл/выкл-регулятора.

Энергопотребление всего дома рассчитывается по общей энергии, используемой кондиционерами, лампами и вытяжками. На рисунках показано энергопотребление умного дома с течением времени.

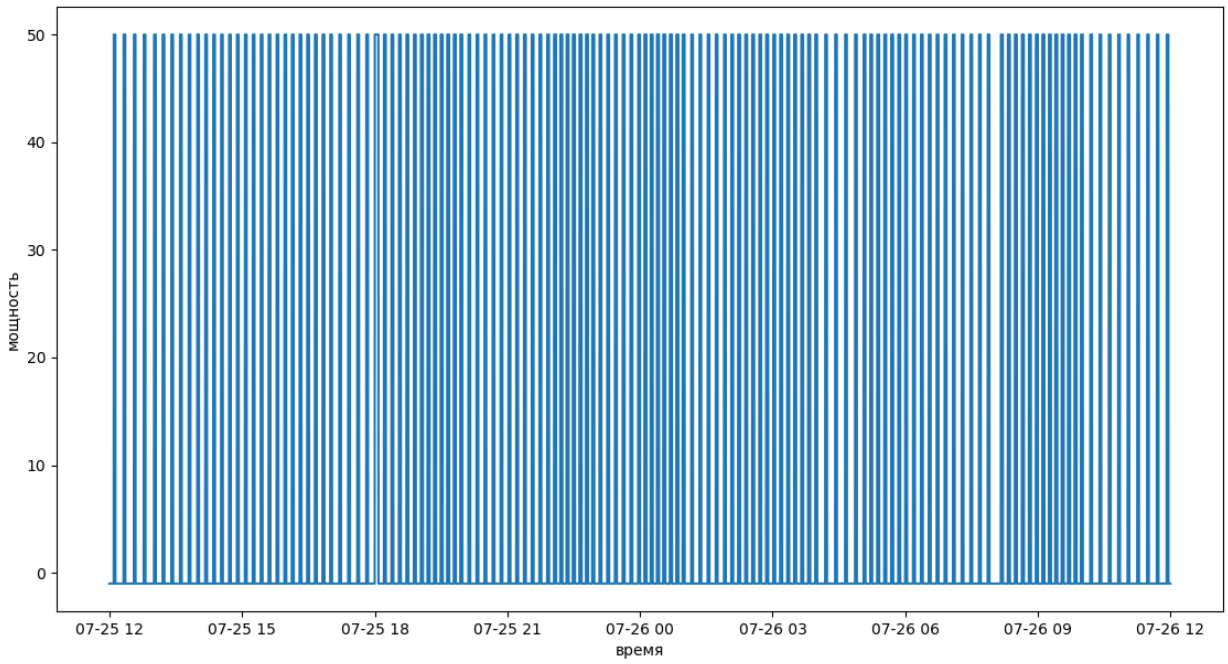


Рисунок 24. энергопотребление умного дома с течением времени (вкл/выкл-регулятора)

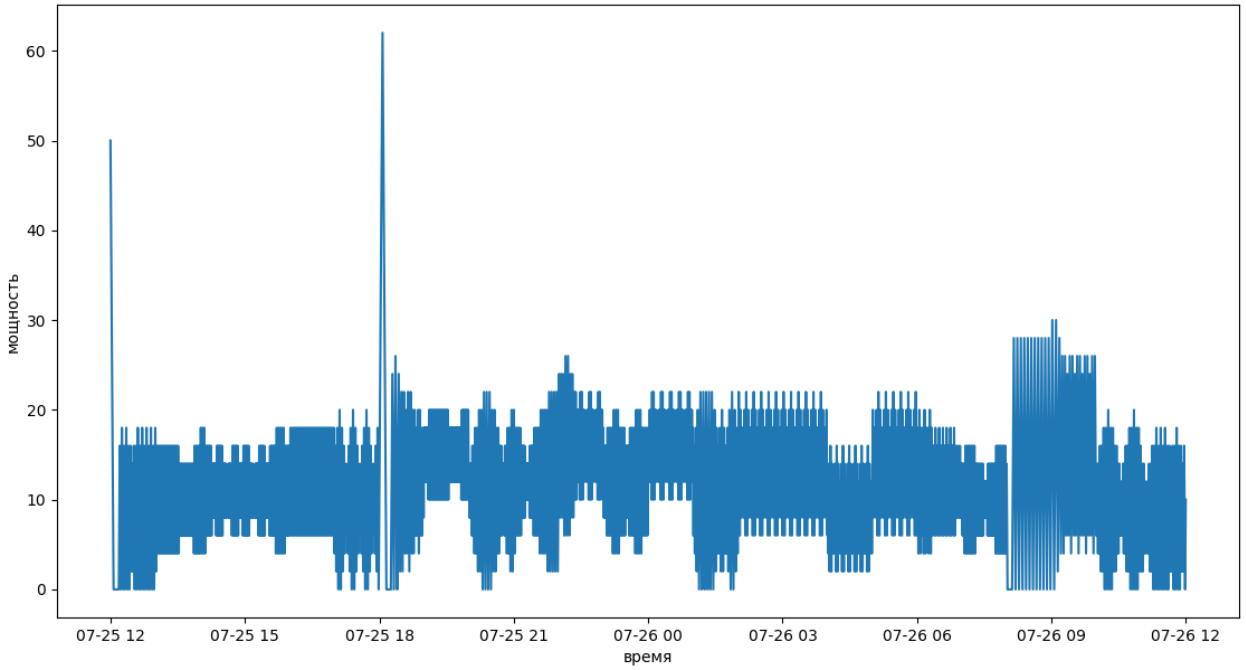


Рисунок 25. энергопотребление умного дома с течением времени(ПИД-регулятора)

На рисунке 24 энергетические линии более разрежены из-за включения и выключения кондиционера.

На рисунках 36, 27 указано общая энергия, которая была использована с момента запуска.

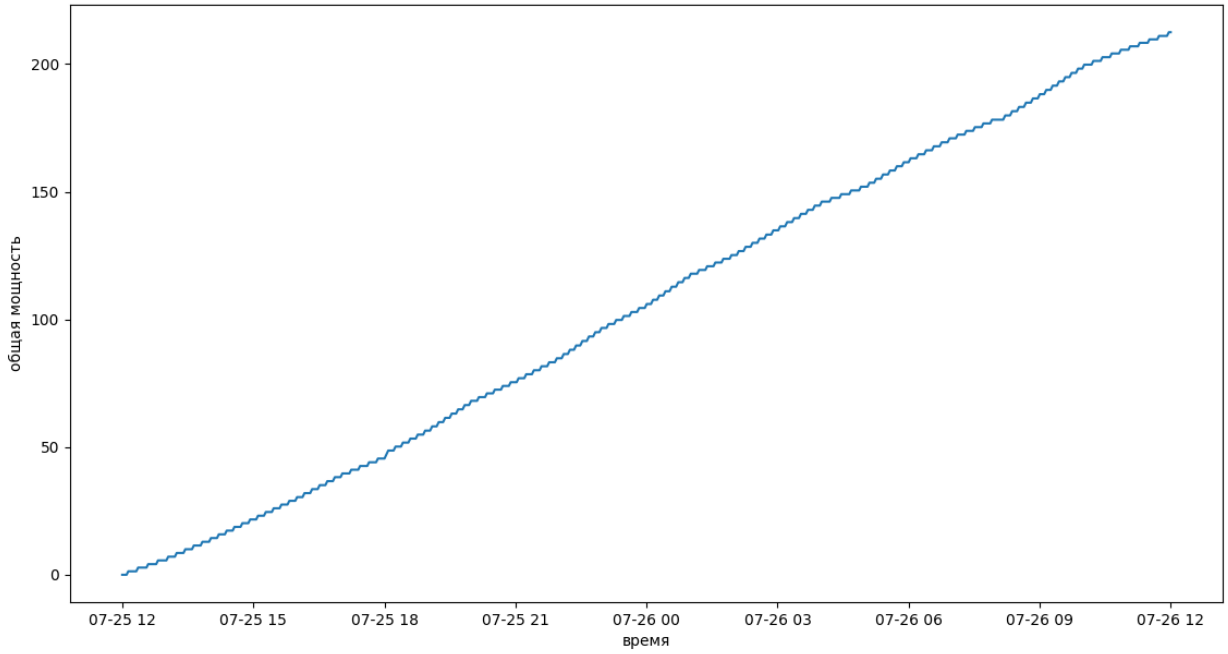


Рисунок 26. общая энергия (вкл/выкл-регулятора)

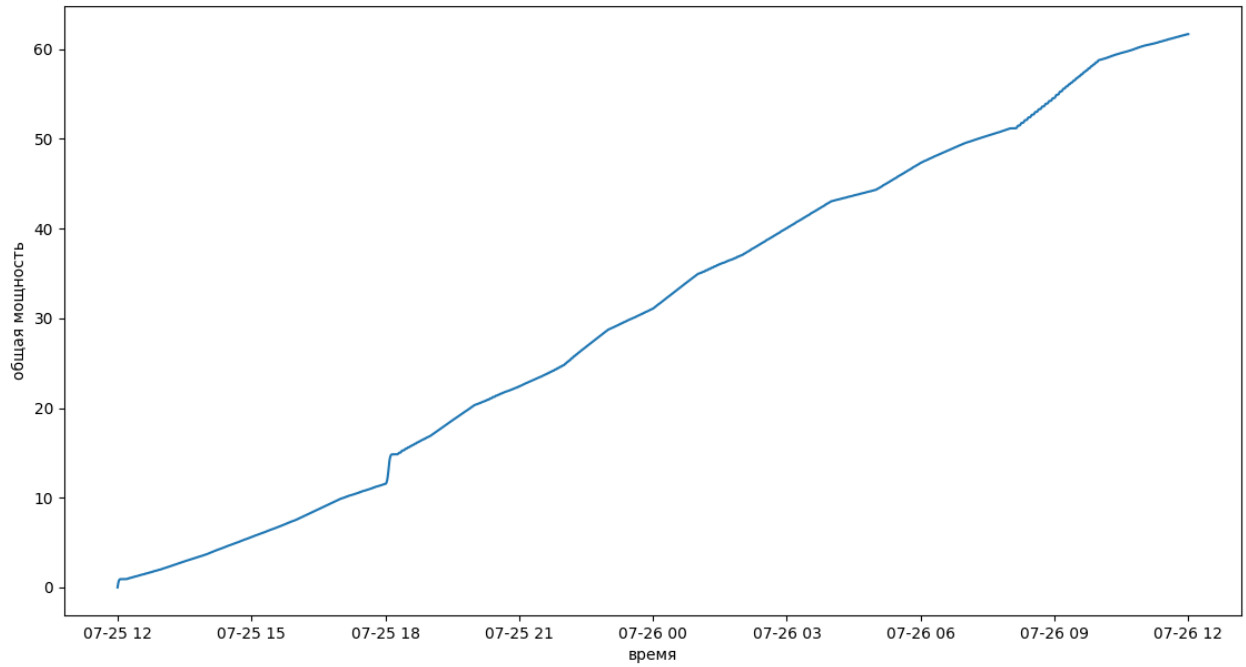


Рисунок 27. общая энергия (ПИД-регулятора)

В течение дня с такими же погодными условиями и деятельностью человека, вкл/выкл-контроллер потребляет более 200 кВтч энергии, что примерно в 3 раза больше энергии, используемой ПИД-регулятором (около 60 кВтч).

Это происходит из-за нелинейности между охлаждающей способностью и электрической мощностью. Электрическая мощность увеличивается быстрее, чем мощность охлаждения, или, другими словами, если мощность охлаждения увеличивается в два раза, электрическая мощность увеличивается не вдвое, а более чем в два раза. Это приводит к тому, что включение и выключение попеременно потребляет больше электроэнергии, чем регулировка уровня работы кондиционера в соответствии с температурой в помещении.

4.5. Выводы

По результатам проведенных исследований можно сделать выводы:

- Использование ПИД-регулятора не только снижает погрешность, но и помогает снизить энергопотребление.
- Однако использование ПИД-регулятора в моделировании требует больше времени из-за необходимости активировать большее количество Transitons, а также времени на соответствующий анализ настройки.

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Питерсон Дж. Теория сетей Петри и моделирование систем, стр. 15-34.
2) Учебный курс МГТУ им. Баумана “Основы САПР. Моделирование”. Сети Петри. Анализ сетей Петри. [Электронный ресурс]. Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=110_Simul/3018.mod/?cou=140_CADedu/CAD.cou
3) Вестник МГТУ МИРЭА 2015 № 1 МОДЕЛИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СЕТЕЙ ПЕТРИ. [Электронный ресурс]. Режим доступа: https://rtj.mirea.ru/upload/medialibrary/941/02-kudj.pdf
4) Некоторые свойства сетей Петри и их приложения (на вьетнамском языке). [Электронный ресурс]. Режим доступа: https://repository.vnu.edu.vn/bitstream/VNU_123/8349/1/01050001007.pdf
5) Course Software Modeling, VNUHCM-University Of Science
6) AUTOMATION OF THE SMART HOUSE SYSTEM-LEVEL DESIGN. [Электронный ресурс]. Режим доступа: http://31.186.81.235:8080/api/files/view/14898.pdf
7) Вложенные сети Петри и моделирование распределенных систем [Электронный ресурс]. Режим доступа: http://skif.pereslavl.ru/psi-info/psi/psi-publications/e-book-2004/e-book/1-4/02-Lomazova-Vlozhennye-seti-p-337.pdf
8) Вложенные сети Петри: моделирование и анализ распределенных систем с объектной структурой. [Электронный ресурс]. Режим доступа: https://www.researchgate.net/publication/264732214_Vlozennye_seti_Petri_modelirovanie_i_analiz_raspredelennyh_sistem_s_obektnoj_strukturoj
9) Nested Petri Nets: Multi-level and Recursive Systems. [Электронный ресурс]. Режим доступа: https://www.researchgate.net/publication/220444866_Nested_Petri_Nets_Multi-level_and_Recursive_Systems

10) Василий Теслюк, Василий Береговский. Development of smart house system model based on colored Petri nets. [Электронный ресурс]. Режим доступа: https://www.researchgate.net/publication/261299505_Development_of_smart_house_system_model_based_on_colored_Petri_nets.

ПРИЛОЖЕНИЕ А

Листинг 1. PetriNet.py

```
from const import *


TransLabels = {}

class SimpleToken:
    def __init__(self, type = NONE, value = None):
        self.type = type
        self.value = value

    def showStatus(self,):
        pass

class Place:
    def __init__(self, tokens = [] ,  name = ''):
        self.name = name
        self.tokens = tokens
        self.holds = self.typeCount()
        self.hold = sum(self.holds.values())

    def countTokens(self,):
        return self.hold

    def typeCount(self,):
        res = {NONE: 0,
               LIGHT_TYPE:0}

        for i in self.tokens:
            if str(i.type) in res:
                res[str(i.type)] += 1
            else:
                res[str(i.type)] = 1
        return res

    def update(self,):
        self.holds = self.typeCount()
        self.hold = sum(self.holds.values())

    def pop(self, type = 0, count = 1):
        c = count
        tokens = []
        index = 0
        #for index, value in enumerate(self.tokens):
        while (index < len(self.tokens) and (c > 0 or c == -1)):
            value = self.tokens[index]
            if (c == -1 or c > 0) and value.type == type:
                tokens.append(self.tokens[index])
                self.tokens.pop(index)
                if (c > 0):
                    c -= 1
            else:
                index += 1
        self.update()
        return tokens
```

```
class ArcBase:
    def __init__(self, place : Place, hold = 1, label = None, allowType = NONE,
name = '', func = None, params = None):
        self.place = place
        self.hold = hold
        if (label == None):
            self.label = NONE
        else:
            self.label = label
        self.allowType = allowType
        self.name = name
        self.func = func
        self.func_params = params

    def isNotBlocking(self,):
        rc = str(self.allowType) in self.place.holds and
self.place.holds[str(self.allowType)] >= self.hold
        return rc

class ArcIn(ArcBase):
    def trigger(self):
        tokens = []
        self.place.holds[str(self.allowType)] -= self.hold

        tokens = self.place.pop(type = self.allowType, count = self.hold)

        if self.func:
            for t in tokens:
                self.func(t, self.func_params)
        return tokens

class ArcOut(ArcBase):
    def trigger(self, tokens, func = None, params = None ):
        append_tokens = []
        c = 0
        for i in tokens:
            if (c < self.hold or self.hold == -1):
                if i.type == self.allowType:
                    append_tokens.append(i)
                    c += 1
        self.place.holds[str(self.allowType)] += self.hold

        if func != None:
            func(append_tokens, params)
        self.place.append(append_tokens)
        return

class ArcEmpty(ArcBase):
    def __init__(self, place, hold = 0, label = None, allowType = NONE, name =
''):
        ArcBase.__init__(self,place, hold = 0, label = None, allowType = NONE,
name = '')

    def isNotBlocking(self,):
        return self.place.countTokens() == 0

    def trigger(self,):
        return []
```

```
class Transition:
    def __init__(self, arcIn, arcOut, name = '', isAuto = False, label = '',
priority = 100,):
        self.name = name
        self.arcIn = arcIn
        self.arcOut = arcOut
        self.priority = priority
        self.isAuto = isAuto
        if label == '':
            label = NONE
        self.label = label
        self.addtoTransLabel()


        for i in arcIn:
            i.name = "[{} -> {}]".format(i.place.name, name)

        for i in arcOut:
            i.name = "[{} -> {}]".format(name, i.place.name)

    def addtoTransLabel(self,):
        global TransLabels
        if self.getLabel() not in TransLabels:
            TransLabels[self.getLabel()] = {'main':[], '*': []}
        if self.label.startswith('*'):
            TransLabels[self.getLabel()]['*'].append(self)
        else:
            TransLabels[self.getLabel()]['main'].append(self)

    def removeFromTransLabel(self,):
        trans = TransLabels[self.getLabel()]
        if self in trans['main']:
            trans['main'].remove(self)
        if self in trans['*']:
            trans['*'].remove(self)

    def getLabel(self):
        if self.label.startswith("*"):
            return self.label
        if self.label.startswith('not'):
            return self.label[3:]
        return self.label
    def setLabel(self, newLabel):
        self.removeFromTransLabel()
        self.label = newLabel
        self.addtoTransLabel()

        self.label = newLabel

    def needFireWith(self, T):
        return self.label == "not" + T.label or "not" + self.label == T.label

    def needFireIf(self, T):
        return self.label == "*" + T.label

    def isActive(self, ):
        for i in self.arcIn:
            if not i.isNotBlocking():
                return False

        return True

    def getInPlaces(self):
        res = []
        for i in self.arcIn:
            res.append(i.place)
        return set(res)
```

```
class PetriNet:
    def __init__(self, transitions = [], name = '', type = None):
        self.name = name
        self.transitions = transitions
        if type == None:
            self.type = self.name
        else:
            self.type = type


    def createTrans(self, arcIn, arcOut, name = '', isAuto = False,
                    label = None, priority = 100,):
        t = Transition(arcIn, arcOut, name, isAuto, label, priority)
        self.transitions.append(t)

    def getAllPlaces(self, ):
        place = []
        for i in self.transitions:
            for j in i.arcIn + i.arcOut:
                place.append(j.place)

        place = list(set(place))
        place.sort(key = lambda x: x.name)
        return place

    def showStatus(self, placeList = None):
        if placeList == None:
            placeList = self.getAllPlaces()
        print("\n\n")
        print(self.name)
        for i in placeList:
            print("{:^10}".format(i.name), end = '')
        print("\n")
        for i in placeList:
            print("{:^10}".format(i.hold), end = '')
        print("\n\n")


    def getTransitionByName(self, name):
        for i in self.transitions:
            if i.name == name:
                return i
        return None
    def getPlaceByName(self, name):
        places = self.getAllPlaces()
        for i in places:
            if i.name == name:
                return i

    def isConflictWithAny(self, T):
        res = []
        for t in self.transitions:
            if t != T and T.isConflictWith(t):
                continue

    def runByName(self, name):
        tran = self.getTransitionByName(name)

        return self.run([tran])
```

```
    def run(self, sequence = None):
        count = 0
        global TransLabels
        if (sequence == None):
            return count

        for i in sequence:
            if (i.label == NONE):
                relatedTrans = [i,]
            else:
                relatedTrans = TransLabels[i.getLabel()]['main']
            isActives = [t.isActive() for t in relatedTrans]

            if (False in isActives):
                continue

            for t in relatedTrans:
                t.fire()
            count += 1
        return count

    def autorun(self):
        res = 0
        self.showStatus()
        while True:
            count_Active = 0

            for i in self.transitions:
                if i.isAuto:
                    rc = self.run([i,])
                    if (rc == 1):
                        count_Active += 1
                        res += 1
                        self.showStatus()
                        break
            if count_Active == 0:
                return
```