БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Ханты-Мансийского автономного округа – Югры

«Сургутский государственный университет»

Политехнический институт Кафедра автоматики и компьютерных систем

ОТЧЕТ

по лабораторной работе

дисциплина Теория Вычислительных Процессов

Тема: Модель Светофора

Выполнил: студент группы 609-21

Шумилов Илья Денисович

Принял: доцент кафедры АиКС

Брагинский М. Я.

Сургут 2025 г.

**Задание:**

1. Описать работу ЦА (управляющего двухсекционным светофором) диаграммой переходов, таблицей переходов
2. Создать программную модель ЦА (на любом ЯП) с отображением состояний, входных и выходных сигналов (таймер и лампы)

**Граф работы (рисунок 1):**

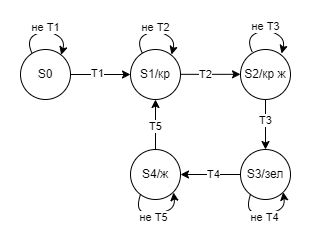


Рисунок 1 - Граф работы ЦА.

**Таблица переходов (таблица 1):**

Таблица 1 – Таблица переходов ЦА

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Время** | **Состояния** | | | | |
| S0 | S1 | S2 | S3 | S4 |
| T1 = 0 | S1 | - | - | - | - |
| T2 = 10 | - | S2 | - | - | - |
| T3 = 12 | - | - | S3 | - | - |
| T4 = 22 | - | - | - | S4 | - |
| T5 = 24 | - | - | - | - | S1 |

**Состояния:**

S0 – Светофор выключен, не горит ни один сигнал светофора;

S1 – Горит красный сигнал светофора;

S2 – Горят красный и желтый сигналы светофора;

S3 – Горит зеленый сигнал светофора;

S4 – Горит желтый сигнал светофора.

**Код программного обеспечения (листинг 1):**

Код был написан на языке программирования Typescript в среде разработки Visual Studio Code.

Листинг 1. Код программного обеспечения:

enum State {

    OFF,        // S0

    RED,        // S1

    YELLOW\_RED, // S2

    GREEN,      // S3

    YELLOW      // S4

}

enum IncomingSignal {

    RED,        // T1

    YELLOW\_RED, // T2

    GREEN,      // T3

    YELLOW,     // T4

}

class TrafficLightStateMachine {

    private \_state: State;

    public get state(): State {

        return this.\_state;

    }

    constructor() {

        this.\_state = State.OFF;

    }

    public sendSignal(signal: IncomingSignal): State {

        switch (signal) {

            case IncomingSignal.RED:

                this.handleRedSignal();

                break;

            case IncomingSignal.YELLOW\_RED:

                this.handleYellowRedSignal();

                break;

            case IncomingSignal.GREEN:

                this.handleGreenSignal();

                break;

            case IncomingSignal.YELLOW:

                this.handleYellowSignal();

                break;

        }

        return this.state;

    }

    private handleRedSignal(): void {

        if (this.\_state == State.OFF || this.\_state == State.YELLOW) {

            this.\_state = State.RED;

        }

    }

    private handleYellowRedSignal(): void {

        if (this.\_state == State.RED) {

            this.\_state = State.YELLOW\_RED;

        }

    }

    private handleGreenSignal(): void {

        if (this.\_state == State.YELLOW\_RED) {

            this.\_state = State.GREEN;

        }

    }

    private handleYellowSignal(): void {

        if (this.\_state == State.GREEN) {

            this.\_state = State.YELLOW;

        }

    }

}

class TrafficLightDomManager {

    readonly redCarLight: HTMLElement;

    readonly yellowCarLight: HTMLElement;

    readonly greenCarLight: HTMLElement;

    readonly redPedestrianLight: HTMLElement;

    readonly greenPedestrianLight: HTMLElement;

    readonly timerText: HTMLElement;

    private \_time: number;

    private stateMachine: TrafficLightStateMachine;

    constructor() {

        this.redCarLight = document.querySelector('#redCarLight') as HTMLElement;

        this.yellowCarLight = document.querySelector('#yellowCarLight') as HTMLElement;

        this.greenCarLight = document.querySelector('#greenCarLight') as HTMLElement;

        this.redPedestrianLight = document.querySelector('#redPedestrianLight') as HTMLElement;

        this.greenPedestrianLight = document.querySelector('#greenPedestrianLight') as HTMLElement;

        this.timerText = document.querySelector('#timerText') as HTMLElement;

        if (!this.redCarLight || !this.yellowCarLight || !this.greenCarLight ||

            !this.redPedestrianLight || !this.greenPedestrianLight || !this.timerText) {

            alert('Not all required elements are present on page!');

        }

        this.stateMachine = new TrafficLightStateMachine();

        this.handleState();

        this.\_time = -1;

    }

    public work(): void {

        setInterval((): void => {

            this.\_time++;

            this.handleTimer();

        }, 1000);

    }

    private handleTimer(): void {

        this.timerText.innerHTML = `Time: ${this.\_time}`;

        if (this.\_time == 0) {

            this.sendSignal(IncomingSignal.RED);

        }

        else if (this.\_time == 10) {

            this.sendSignal(IncomingSignal.YELLOW\_RED);

        }

        else if (this.\_time == 12) {

            this.sendSignal(IncomingSignal.GREEN);

        }

        else if (this.\_time == 22) {

            this.sendSignal(IncomingSignal.YELLOW);

        }

        else if (this.\_time == 24) {

            this.sendSignal(IncomingSignal.RED);

            this.\_time = -1;

        }

    }

    private sendSignal(signal: IncomingSignal): void {

        this.stateMachine.sendSignal(signal);

        this.handleState();

    }

    private handleState(): void {

        switch (this.stateMachine.state) {

            case State.OFF:

                this.handleOffState();

                break;

            case State.RED:

                this.handleRedState();

                break;

            case State.YELLOW\_RED:

                this.handleYellowRedState();

                break;

            case State.GREEN:

                this.handleGreenState();

                break;

            case State.YELLOW:

                this.handleYellowState();

                break;

        }

    }

    private handleOffState(): void {

        this.redCarLight.classList.add('inactive');

        this.yellowCarLight.classList.add('inactive');

        this.greenCarLight.classList.add('inactive');

        this.redPedestrianLight.classList.add('inactive');

        this.greenPedestrianLight.classList.add('inactive');

    }

    private handleRedState(): void {

        this.redCarLight.classList.remove('inactive');

        this.yellowCarLight.classList.add('inactive');

        this.redPedestrianLight.classList.add('inactive');

        this.greenPedestrianLight.classList.remove('inactive');

    }

    private handleYellowRedState(): void {

        this.yellowCarLight.classList.remove('inactive');

        this.redPedestrianLight.classList.remove('inactive');

        this.greenPedestrianLight.classList.add('inactive');

    }

    private handleGreenState(): void {

        this.redCarLight.classList.add('inactive');

        this.yellowCarLight.classList.add('inactive');

        this.greenCarLight.classList.remove('inactive');

    }

    private handleYellowState(): void {

        this.yellowCarLight.classList.remove('inactive');

        this.greenCarLight.classList.add('inactive');

    }

}

let domManager = new TrafficLightDomManager();

domManager.work();

Пример работы программы (рисунки 2-6):

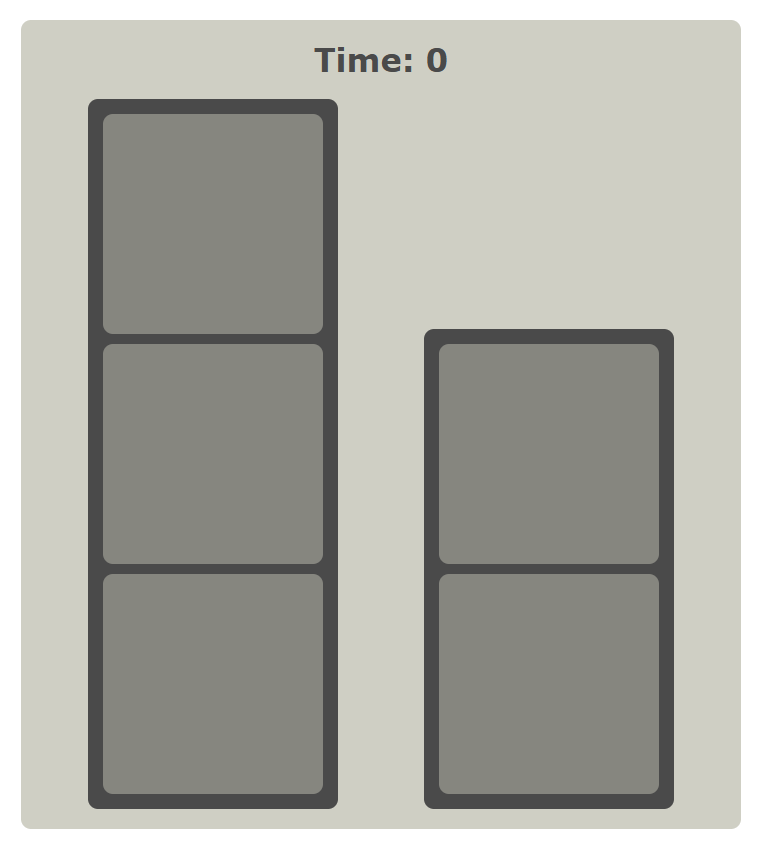


Рисунок 2 – Светофор в состоянии S0

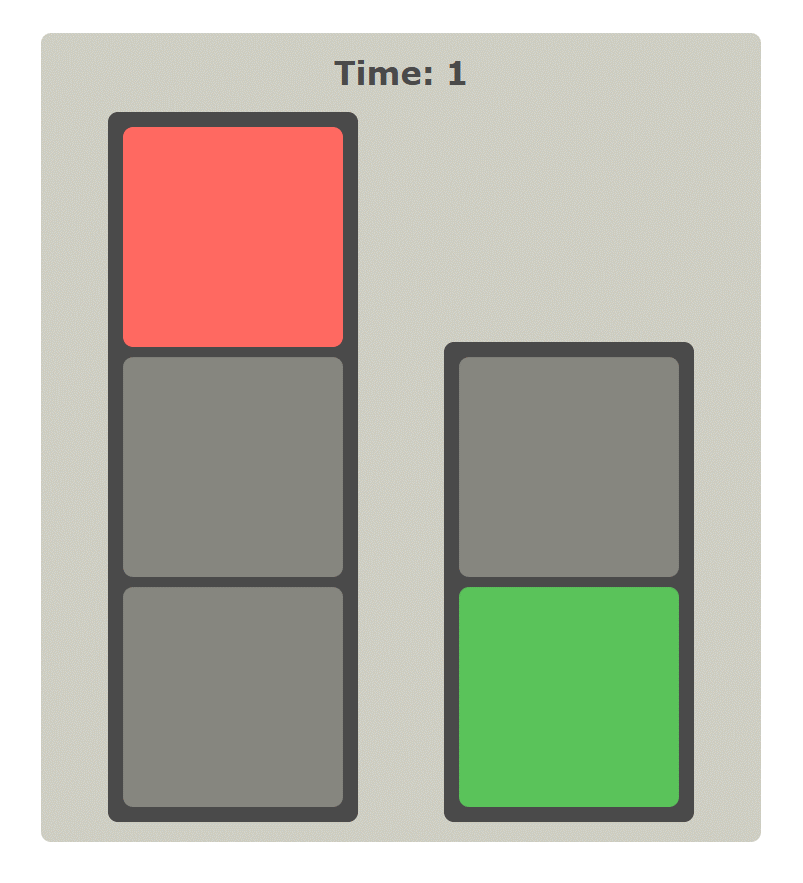


Рисунок 3 – Светофор в состоянии S1

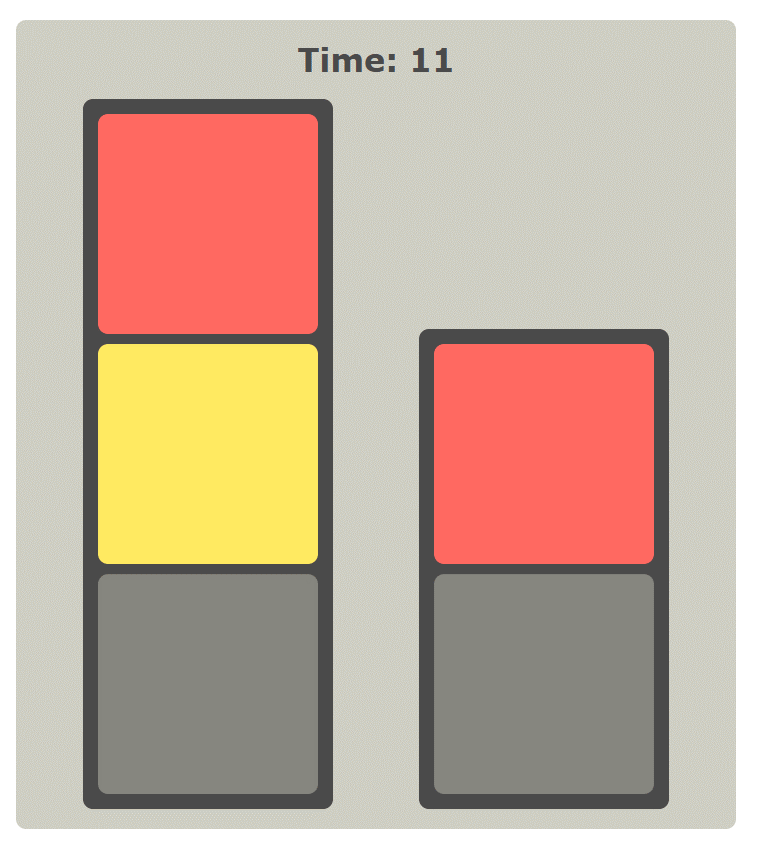


Рисунок 4 – Светофор в состоянии S2

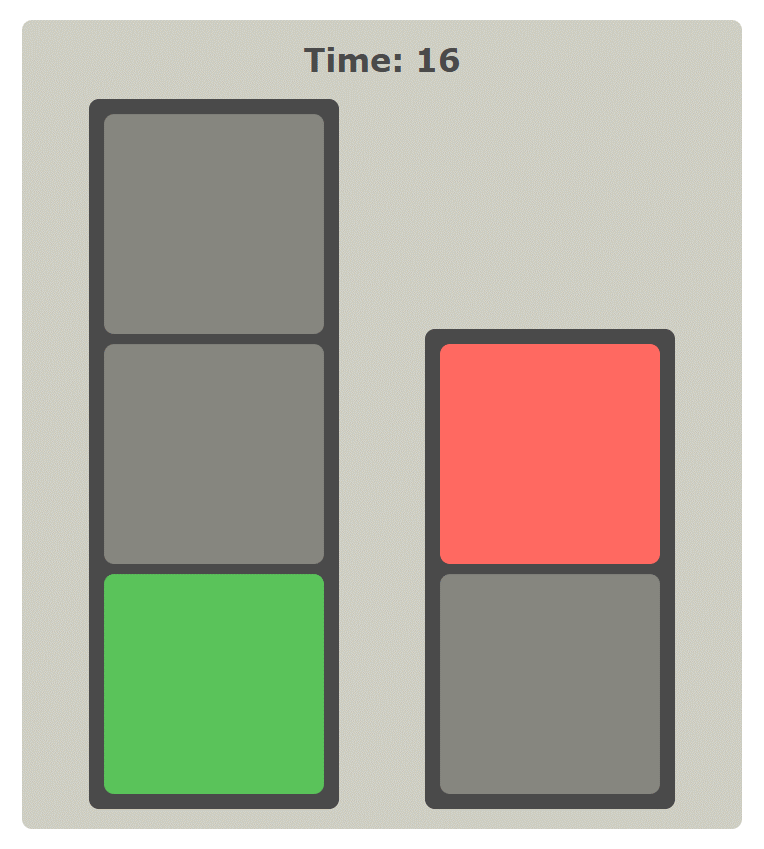


Рисунок 5 – Светофор в состоянии S3

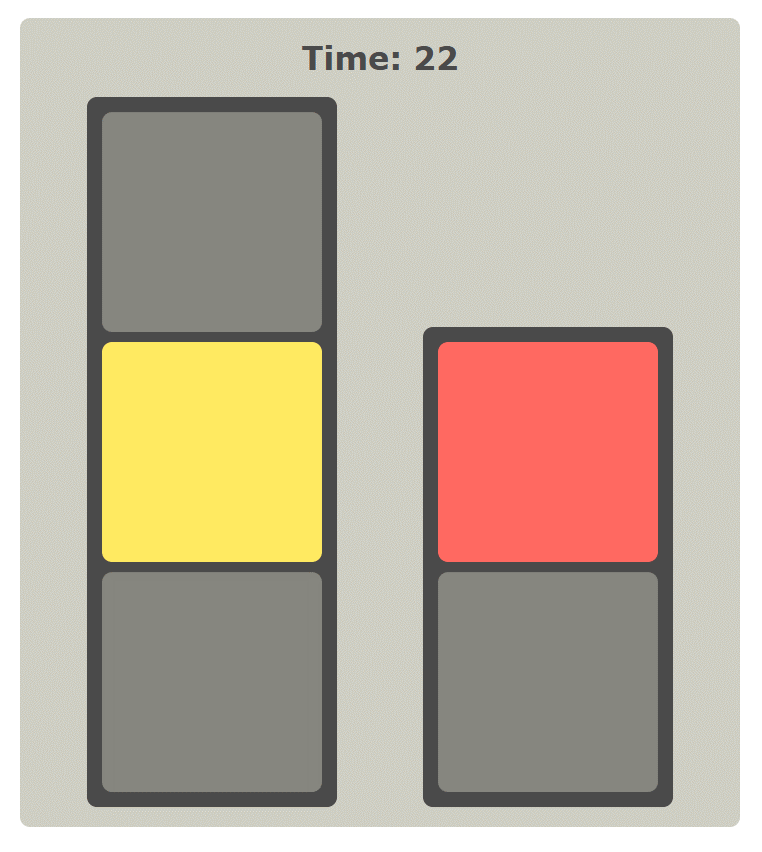


Рисунок 6 – Светофор в состоянии S4

Выводы: в ходе лабораторной работы была создана программная реализация цифрового автомата модели светофора.