МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра информационных технологий в атомной энергетике (ИТАЭ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Автоматизированные системы управления технологическими процессами на объектах атомной отрасли»

Текущий контроль Светофор

Студент группы *ИКБО-50-23, Враженко Д.О.* _____
(подпись)

Преподаватель *Прорехин С.А.*

_____
(подпись)

Отчет представлен «___»_____202__г.

Москва 2025 г.

ВВЕДЕНИЕ

В рамках текущего контроля по дисциплине «Автоматизированные системы управления технологическими процессами на объектах атомной отрасли» была поставлена задача реализовать алгоритм работы светофора. Работа была выполнена на различных платформах: языки программирования общего назначения (Python, C++) и среды интерактивных игр (Terraria, Minecraft), что позволило продемонстрировать универсальность принципов проектирования алгоритмов управления.

Цель работы: закрепить навыки проектирования циклических алгоритмов управления с строгим соблюдением временных параметров и гарантией отсутствия конфликтующих состояний.

Основные требования:

- Цикл: 🔴 Красный (30 с) → 🟢 Зеленый (30 с) → 🟡 Желтый (5 с) → 🔴 Красный...
- Одновременно должен быть активен только один сигнал.
- Алгоритм должен работать бесконечно.

Для наглядной демонстрации работы всех реализаций к отчету прилагается видеообзор, где показаны запуск, цикличность работы и особенности каждой из программ.

РЕАЛИЗАЦИЯ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

Среда разработки: IDLE (Python 3.13 64-bit).

Описание алгоритма:

Алгоритм реализован в виде явного конечного автомата, где каждое состояние (красный, зеленый, желтый) активируется последовательно в бесконечном цикле `while True`. Для соблюдения временных интервалов используется блокирующая функция `time.sleep()`, которая приостанавливает выполнение программы на заданное количество секунд.

Ключевые особенности реализации:

- **Универсальная функция проверки безопасности `check_safety`:** Функция принимает три булевых параметра, где первый - ожидаемый активный сигнал, а два других — ожидаемые неактивные сигналы. Это позволяет использовать одну и ту же функцию для проверки безопасности в разных состояниях, передавая параметры в соответствующем порядке для каждого случая.
- **Последовательное переключение состояний:** Алгоритм использует простую и надежную схему, где каждое новое состояние явно устанавливается путем включения одного сигнала и выключения предыдущего. Это предотвращает возможность одновременной активации нескольких сигналов на границах переходов.
- **Обработка внешних прерываний:** Основной цикл обернут в блок `try-except` для корректной обработки прерывания программы пользователем (Ctrl+C), что обеспечивает плавное завершение работы без вывода трассировки ошибки.
- **Визуализация в консоли:** Для наглядности каждый активный сигнал сопровождается соответствующим эмодзи и текстовым описанием, что позволяет визуально контролировать работу системы в реальном времени.

Преимущества реализации:

- исключительная читаемость и прозрачность логики работы;
- универсальная система валидации состояний с детализированными сообщениями об ошибках;
- простота модификации временных параметров и добавления новых состояний.

Листинг кода приложен. Работа программы и вывод в консоль наглядно продемонстрированы в видеообзоре.

Листинг 1 — Код на языке программирования Python

```
import time

# Функция для проверки сигналов
def check_safety(true_signal, false_signal1, false_signal2):
    signals = [true_signal, false_signal1, false_signal2]
    active_count = sum(signals)
    if active_count != 1:
        raise Exception(f"КОНФЛИКТ СОСТОЯНИЙ! Активно сигналов:
{active_count}.")
    return True


print("Светофор запущен (для остановки нажмите Ctrl+C)")
print("Цикл: 🔴 Красный -> 🟢 Зеленый -> 🟡 Желтый -> 🔴 Красный...")
print("Временные интервалы: 🔴 Красный = 30 сек., 🟡 Желтый = 5 сек., 🟢
Зеленый = 30 сек.")
print()
RLight = False; RTime = 30
YLight = False; YTime = 5
GLight = False; GTime = 30
try:
    while True:
        YLight = False
        RLight = True
        if check_safety(RLight, YLight, GLight):
            print(f"🔴 Красный")
            time.sleep(RTime)
        RLight = False
        GLight = True
        if check_safety(GLight, RLight, YLight):
            print(f"🟢 Зеленый")
            time.sleep(GTime)
        GLight = False
        YLight = True
        if check_safety(YLight, GLight, RLight):
            print(f"🟡 Желтый")
            time.sleep(YTime)
except KeyboardInterrupt:
    print("\n\nРабота светофора остановлена.")
except Exception as e:
    print(f"\n\nОШИБКА БЕЗОПАСНОСТИ: {e}")
```

РЕАЛИЗАЦИЯ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++

Среда разработки: Visual Studio.

Описание алгоритма:

Алгоритм представляет собой конечный автомат с тремя состояниями, реализованный в виде бесконечного цикла `while (true)`. Логика построена на последовательном переключении трех булевых переменных (RLight, YLight, GLight), отвечающих за состояния сигналов. Для соблюдения временных интервалов используется функция приостановки выполнения текущего потока `std::this_thread::sleep_for()` с указанием времени в секундах.

Ключевые особенности реализации:

- **Универсальная функция проверки безопасности `checkSafety`:** Функция принимает три булевых параметра, где первый - ожидаемый активный сигнал, а два других — ожидаемые неактивные сигналы. Это позволяет использовать одну и ту же функцию для проверки безопасности в разных состояниях, передавая параметры в соответствующем порядке для каждого случая.
- **Использование исключений:** Основной цикл обернут в блок `try-catch` для корректного перехвата и обработки возможных ошибок безопасности, что обеспечивает аварийную остановку программы с выводом диагностического сообщения.
- **Поддержка кириллицы:** Использование `setlocale(LC_ALL, "Russian")` обеспечивает корректный вывод русскоязычных сообщений в консоль.

Преимущества реализации:

- исключительная читаемость и прозрачность логики работы;
- универсальная система валидации состояний с детализированными сообщениями об ошибках;
- простота модификации временных параметров и добавления новых

состояний.

Листинг кода приложен. Пример работы скомпилированной программы показан в видеообзоре.

Листинг 2 — Код на языке программирования C++

```
#include <iostream>
#include <thread>
#include <string>

// Функция для проверки сигналов
bool checkSafety(bool true_signal, bool false_signal1, bool false_signal2) {
    int activeCount = (true_signal ? 1 : 0) + (false_signal1 ? 1 : 0) +
(false_signal2 ? 1 : 0);
    if (activeCount != 1)
        throw std::runtime_error("КОНФЛИКТ СОСТОЯНИЙ! Активно сигналов: " +
std::to_string(activeCount));
    return true;
}

int main() {
    setlocale(LC_ALL, "Russian");
    std::cout << "Светофор запущен" << std::endl;
    std::cout << "Цикл: Красный -> Зеленый -> Желтый -> Красный..." <<
std::endl;
    std::cout << "Временные интервалы: Красный = 30 сек., Желтый = 5 сек.,
Зеленый = 30 сек." << std::endl;
    std::cout << std::endl;
    bool RLight = false; int RTime = 30;
    bool YLight = false; int YTime = 5;
    bool GLight = false; int GTime = 30;
    try {
        while (true) {
            YLight = false;
            RLight = true;
            if (checkSafety(RLight, YLight, GLight)) {
                std::cout << "Красный" << std::endl;
                std::this_thread::sleep_for(std::chrono::seconds(RTime));
            }
            RLight = false;
            GLight = true;
            if (checkSafety(GLight, RLight, YLight)) {
                std::cout << "Зеленый" << std::endl;
                std::this_thread::sleep_for(std::chrono::seconds(GTime));
            }
            GLight = false;
            YLight = true;
            if (checkSafety(YLight, GLight, RLight)) {
                std::cout << "Желтый" << std::endl;
                std::this_thread::sleep_for(std::chrono::seconds(YTime));
            }
        }
    }
    catch (const std::exception& e) {
        std::cerr << "ОШИБКА БЕЗОПАСНОСТИ: " << e.what() << std::endl;
        return 1;
    }
    catch (...) {
        std::cerr << "Неизвестная ошибка" << std::endl;
        return 1;
    }

    return 0;
}
```

TERRARIA

Платформа: Игра Terraria с использованием механизмов игры.

Описание реализации:

В игре Terraria логика светофора была реализована с помощью системы проводов, логических элементов и таймеров.

Визуальные сигналы: В качестве сигналов использовались окрашенные «Глуповатые шарики» в соответствующие для светофора цвета (красный/жёлтый/зелёный).

Логика и таймеры: Использовались Логические элементы «И», «ИЛИ», «НЕИ» и Таймеры самой игры. Настройка таймеров производилась таким образом, чтобы они активировали соответствующие сигналы через заданные интервалы (30с, 5с, 30с).

Цикличность: Цепочка таймеров была закольцована, обеспечивая бесконечную работу.

Безопасность: Схема была построена таким образом, что активация одного сигнала автоматически гасила предыдущий, что исключает конфликт состояний.

Визуальная работа схемы, процесс переключения сигналов и обзор логики с помощью проводов подробно показаны в видеообзоре.

MINECRAFT

Платформа: Игра Minecraft с использованием системы командных блоков и счетчиков.

Описание реализации:

В Minecraft для создания светофора была использована продвинутая система командных блоков и scoreboard-механик, что позволило реализовать точный временной контроль и проверку безопасности состояний.

Основные механизмы реализации:

1. Система отсчета времени:
   - для отсчета времени используется счетчик cycle_step, который непрерывно увеличивается на 1 каждый тик (0,05 секунды);
   - при достижении значения 1300 (65 секунд полного цикла) счетчик сбрасывается до 0, обеспечивая цикличность.
2. Визуальное представление и тайминг:
   - 🔴 Красный (30 секунд = 600 тиков): Блок на позиции (3,60,0) становится красным, а жёлтый погасает;
   - 🟡 Жёлтый (5 секунд = 100 тиков): Блок на позиции (3,59,0) становится жёлтым, а зелёный погасает;
   - 🟢 Зелёный (30 секунд = 600 тиков): Блок на позиции (3,58,0) становится зелёным, а красный погасает.
3. Система проверки безопасности:
   - реализована расширенная система диагностики состояний;
   - при нормальной работе для каждого состояния выводится сообщение "✅ Светофор в норме";
   - при обнаружении конфликтов (одновременное включение нескольких сигналов) выводятся соответствующие сообщения об ошибках:
     - "🚨 ОШИБКА: Активны ВСЕ три сигнала!";
     - "⚠ ОШИБКА: Конфликт красный-желтый!";

- "⚠ ОШИБКА: Конфликт красный-зеленый!";
- "⚠ ОШИБКА: Конфликт желтый-зеленый!".

Преимущества данного подхода:

- высокая точность таймингов (с точностью до игрового тика);
- встроенная система мониторинга и самодиагностики состояния;
- отсутствие конфликтующих состояний гарантируется логикой команд;
- наглядность работы через сообщения в чате.

Полная демонстрация работы командных блоков, переключения состояний и системы диагностики представлена в видеообзоре.

Листинг 3 — Команды, используемые в командных блоках

```
scoreboard objectives add traffic_light dummy

scoreboard players set cycle_step traffic_light 0

scoreboard players add cycle_step traffic_light 1

fill 3 60 0 3 58 0 black_concrete

execute if score cycle_step traffic_light matches 1300.. run scoreboard
players set cycle_step traffic_light 0

execute if score cycle_step traffic_light matches 0 run setblock 3 60 0
red_concrete

execute if score cycle_step traffic_light matches 600 run setblock 3 58 0
lime_concrete

execute if score cycle_step traffic_light matches 1200 run setblock 3 59 0
yellow_concrete

execute if score cycle_step traffic_light matches 0 run setblock 3 59 0
minecraft:black_concrete

execute if score cycle_step traffic_light matches 600 run setblock 3 60 0
minecraft:black_concrete

execute if score cycle_step traffic_light matches 1200 run setblock 3 58 0
minecraft:black_concrete

execute if block 3 60 0 red_concrete if block 3 59 0 black_concrete if block
3 58 0 black_concrete run say ✅ Светофор в норме

execute if block 3 60 0 black_concrete if block 3 59 0 yellow_concrete if
block 3 58 0 black_concrete run say ✅ Светофор в норме

execute if block 3 60 0 black_concrete if block 3 59 0 black_concrete if
block 3 58 0 lime_concrete run say ✅ Светофор в норме

execute if block 3 60 0 red_concrete if block 3 59 0 yellow_concrete if block
3 58 0 black_concrete run say ⚠ ОШИБКА: Конфликт красный-желтый!

execute if block 3 60 0 red_concrete if block 3 59 0 black_concrete if block
3 58 0 lime_concrete run say ⚠ ОШИБКА: Конфликт красный-зеленый!

execute if block 3 60 0 black_concrete if block 3 59 0 yellow_concrete if
block 3 58 0 lime_concrete run say ⚠ ОШИБКА: Конфликт желтый-зеленый!

execute if block 3 60 0 red_concrete if block 3 59 0 yellow_concrete if block
3 58 0 lime_concrete run say 🚨 ОШИБКА: Активны ВСЕ три сигнала!
```

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Все реализации были протестированы.

Временные интервалы соблюдаются точно.

Конфликтующие состояния (включение нескольких сигналов одновременно) отсутствуют во всех реализациях.

Алгоритм работает циклически и не останавливается до принудительного прерывания.

Результаты тестирования и корректная работа всех четырех светофоров в реальном времени продемонстрированы в видеообзоре.

ЗАКЛЮЧЕНИЕ

В ходе работы были успешно реализованы алгоритмы управления светофором на четырех различных платформах. Несмотря на различия в инструментарии, основополагающие принципы АСУ ТП были соблюдены:

- цикличность работы;
- строгое соблюдение временных характеристик;
- гарантия единственности активного состояния.

Работа демонстрирует, что ключевые инженерные понятия — конечный автомат, тайминг и безопасность — являются универсальными и могут быть применены в самых разных средах, от высокоуровневых языков программирования до виртуальных игровых миров.

Наглядным подтверждением корректности всех реализаций служит прилагаемый видеообзор, в котором можно наблюдать параллельную работу всех светофоров, их временные циклы и бесконфликтное переключение состояний.