УДК 656:004.31

СИСТЕМА КОНТРОЛЯ ДОРОЖНОГО ТРАНСПОРТА И БЕЗОПАСНОСТИ С ИСПОЛЬЗОВАНИЕМ МИКРОПРОЦЕССОРОВ

Сулейменов Аскер Раймбекович
kabae@List.ru
Студент 3 курса образовательной программы 6B06103 – Администрирование систем и сетей
Научный руководитель – **Габдулова А.Е.**
сеньор-лектор, магистр
Атырауский университет им.Х.Досмухамедова, г.Атырау, Республика Казахстан

Введение

Современные транспортные средства представляют собой сложные технологические комплексы, требующие постоянного контроля и оперативного реагирования на изменения в их техническом состоянии. С ростом интенсивности дорожного движения и усложнением транспортной инфраструктуры вопрос безопасности становится первостепенным. Для повышения уровня безопасности дорожного движения разрабатываются системы мониторинга, способные не только отслеживать критические параметры, такие как скорость и уровень топлива, но и своевременно сигнализировать о возникновении потенциально опасных ситуаций.

Приложение «Система мониторинга транспорта и безопасности» разработано на языке Python с использованием библиотеки Tkinter для создания удобного графического интерфейса. Основной функционал приложения заключается в имитации работы микропроцессорной системы, которая непрерывно контролирует состояние транспортного средства. При обнаружении несоответствий нормальным эксплуатационным показателям, таких как превышение установленной скорости или снижение уровня топлива ниже безопасного минимума, система автоматически активирует сигнализацию. Такой подход позволяет оперативно информировать водителя или оператора о возможных аварийных ситуациях, что способствует своевременному принятию мер по обеспечению безопасности.

Особое внимание в разработке уделено параллельному выполнению процессов. Для обеспечения отзывчивости графического интерфейса процесс мониторинга запускается в отдельном потоке, что позволяет избежать блокировки основного окна и гарантировать непрерывное обновление данных в реальном времени. Это особенно важно в условиях динамичного изменения параметров транспортного средства, когда задержки в отображении информации могут привести к неверной оценке ситуации.

Таким образом, представленное приложение демонстрирует возможности современных программных решений в области транспортного мониторинга и безопасности. Оно служит не только учебным примером реализации систем контроля и сигнализации, но и потенциалом для дальнейшей интеграции с реальными датчиками и системами управления транспортными средствами. Внедрение подобных решений способствует повышению уровня безопасности на дорогах, снижению риска аварийных ситуаций и созданию более эффективных систем управления движением.

1. Функциональные возможности приложения

Приложение выполняет следующие функции:

1.1 Мониторинг параметров транспортного средства:
При каждом цикле работы система обновляет данные о текущей скорости и уровне топлива. Данные генерируются случайным образом для имитации реальной работы датчиков.

1.2 Проверка условий безопасности:
система анализирует полученные параметры и проверяет, не превышает ли скорость заданный порог или не опускается ли уровень топлива ниже минимального значения. При выявлении нарушений система выводит предупреждения.

1.3 Активация сигнализации:

при обнаружении опасных ситуаций (например, превышение скорости или низкий уровень топлива) система активирует сигнализацию. Это служит сигналом для оператора или водителя о необходимости принятия мер.

1.4 Графический вывод данных:

результаты работы системы выводятся в отдельном окне с использованием графического интерфейса на базе Tkinter. Такой подход позволяет в реальном времени отслеживать изменения и получать визуальные уведомления о состоянии транспортного средства.

1.5 Параллельное выполнение процессов:

для обеспечения отзывчивости пользовательского интерфейса процесс мониторинга выполняется в отдельном потоке. Это предотвращает блокировку главного окна приложения, позволяя пользователю наблюдать динамику системы без задержек.

2. Архитектура и структура кода

```
import tkinter as tk
import time
import random
import threading

class VehicleMonitor:
    def __init__(self, text_widget, max_speed=120, min_fuel=10):
        """
        Инициализация параметров транспортного средства и текстового виджета для вывода.
        :param text_widget: виджет Tkinter для отображения логов
        :param max_speed: максимальная допустимая скорость (км/ч)
        :param min_fuel: минимальный допустимый уровень топлива (%)
        """
        self.speed = 0        # текущая скорость (км/ч)
        self.fuel = 100       # текущий уровень топлива (%)
        self.max_speed = max_speed
        self.min_fuel = min_fuel
        self.alarm_active = False
        self.text_widget = text_widget

    def log(self, message):
        """Вывод сообщения в текстовый виджет."""
        self.text_widget.insert(tk.END, message + "\n")
        self.text_widget.see(tk.END)

    def update_state(self):
        """
        Обновление состояния транспортного средства: скорость и уровень топлива.
        Используется генерация случайных значений для имитации работы датчиков.
        """
        self.speed = random.randint(0, 150)
        self.fuel = max(0, self.fuel - random.randint(0, 5))
        self.log(f"Скорость: {self.speed} км/ч, Уровень топлива: {self.fuel}%")

    def check_safety(self):
        """
```

```
        Проверка условий безопасности: превышение скорости или низкий уровень топлива.
        При нарушении условий активируется сигнализация.
        """
        if self.speed > self.max_speed:
            self.log("Предупреждение: Обнаружено превышение скорости!")
            self.trigger_alarm("Превышение скорости")
        if self.fuel < self.min_fuel:
            self.log("Предупреждение: Низкий уровень топлива!")
            self.trigger_alarm("Низкий уровень топлива")

    def trigger_alarm(self, reason):
        """
        Активация системы сигнализации с указанием причины.
        :param reason: причина срабатывания сигнализации
        """
        if not self.alarm_active:
            self.log(f"СИГНАЛИЗАЦИЯ! Причина: {reason}. Примите меры безопасности!")
            self.alarm_active = True
        else:
            self.log("Сигнализация уже активирована.")

    def reset_alarm(self):
        """
        Сброс сигнализации (например, после устранения опасной ситуации).
        """
        if self.alarm_active:
            self.log("Сброс сигнализации.")
            self.alarm_active = False

def monitoring_loop(monitor: VehicleMonitor):
    """
    Главный цикл мониторинга: периодически обновляет состояние транспортного средства, проводит проверки и включает/сбрасывает сигнализацию.
    """
    while monitor.fuel > 0:
        monitor.update_state()
        monitor.check_safety()
        time.sleep(1)
        if monitor.alarm_active:
            time.sleep(2)
            monitor.reset_alarm()

def start_monitoring(monitor):
    monitoring_loop(monitor)

def main():
    # Создание основного окна Tkinter
    root = tk.Tk()
    root.title("Система мониторинга транспорта и безопасности")

    # Создание текстового виджета для вывода логов
```

```
    text_widget = tk.Text(root, wrap='word', width=80, height=20)
    text_widget.pack(padx=10, pady=10)

    # Инициализация системы мониторинга с передачей текстового виджета
    monitor = VehicleMonitor(text_widget)

    # Запуск мониторинга в отдельном потоке, чтобы не блокировать графический интерфейс
    threading.Thread(target=start_monitoring, args=(monitor,), daemon=True).start()

    # Запуск основного цикла обработки событий GUI
    root.mainloop()

if _name_== "_main_": main()
```

Приложение построено по модульному принципу, что обеспечивает удобство поддержки и масштабирования:

2.1 Класс Vehicle Monitor

Главный класс системы отвечает за:

- **Инициализацию параметров:**

В конструкторе задаются начальные значения скорости (0 км/ч) и уровня топлива (100%). Также устанавливаются пороговые значения для скорости (например, 120 км/ч) и уровня топлива (например, 10%).

- **Методы обновления состояния:**

Метод update_state() генерирует новые значения скорости и уровня топлива, имитируя работу датчиков, и выводит результат в графический виджет.

- **Проверку условий безопасности:**

Метод check_safety() анализирует текущие параметры и вызывает активацию сигнализации через метод trigger_alarm(), если обнаружено превышение скорости или низкий уровень топлива.

- **Управление сигнализацией:**

Методы trigger_alarm() и reset_alarm() отвечают за включение и последующий сброс сигнализации. При активации выводится сообщение с указанием причины срабатывания.

- **Логирование в интерфейс:**

Метод log() отвечает за вывод сообщений в текстовый виджет графического окна, что позволяет пользователю получать обратную связь о текущем состоянии системы.

2.2 Основной цикл мониторинга

Функция monitoring_loop() реализует непрерывное обновление состояния транспортного средства. Основные этапы цикла включают:

- Обновление параметров транспортного средства.
- Проверку условий безопасности.
- Вывод данных и сообщений в графический интерфейс.
- Управление временем обновления с использованием функции time.sleep() для имитации реального времени работы.

2.3 Графический интерфейс (Tkinter)

Для визуализации данных используется библиотека Tkinter, что позволяет создать простое и интуитивно понятное окно:

- **Окно приложения:**

Основное окно создается с помощью метода Tk(), задаются заголовок и параметры окна.

- **Текстовый виджет:**

Создан текстовый виджет, в котором отображаются логи работы системы (обновление параметров, предупреждения и сообщения о сигнализации). Виджет обеспечивает автоматическую прокрутку для удобства восприятия информации.

2.4 Параллельное выполнение с помощью потоков

Чтобы интерфейс оставался отзывчивым во время выполнения непрерывного мониторинга, используется модуль threading. Запуск функции monitoring_loop() в отдельном потоке позволяет параллельно работать с графическим интерфейсом:

- **Преимущества:**

Такой подход предотвращает зависание интерфейса, так как основная нить отвечает за события GUI, а отдельный поток – за фоновый мониторинг.

2.5 Запуск приложения

При запуске кода приложения «Система мониторинга транспорта и безопасности» происходит последовательность следующих этапов:

1. Инициализация системы

- Приложение начинает работу с создания объекта класса VehicleMonitor, который представляет собой микропроцессорную систему для контроля состояния транспортного средства. В этот момент:
- Устанавливаются начальные значения скорости, уровня топлива и состояния сигнализации.
- Создаются методы для обновления параметров и проверки условий безопасности.

2. Запуск графического интерфейса

- Программа использует библиотеку Tkinter для создания удобного окна управления, где отображаются основные параметры транспортного средства:
- Скорость автомобиля (изменяется динамически).
- Уровень топлива (имитируется случайное снижение).
- Индикатор тревожных ситуаций (включается при опасных условиях).

3. Параллельный запуск мониторинга

- Для обеспечения постоянного обновления данных создаётся отдельный поток, который выполняет:
- Обновление параметров: скорость и уровень топлива изменяются случайным образом, моделируя реальное поведение автомобиля.
- Проверку условий безопасности: если скорость превышает допустимый порог или уровень топлива падает ниже критической отметки, включается система сигнализации.
- Обновление интерфейса: все изменения мгновенно отображаются в окне Tkinter, обеспечивая мониторинг в реальном времени.

4. Отображение и реагирование на опасные ситуации

- Если скорость превышает 120 км/ч, активируется аварийный сигнал.
- Если уровень топлива падает ниже 10%, появляется предупреждение.
- Включённая тревога сигнализирует водителю о необходимости принять меры (например, сбавить скорость или заправить топливо).

5. Ожидание действий пользователя

- Пиложение работает в интерактивном режиме, позволяя пользователю наблюдать за изменением параметров и реагировать на них. Графический интерфейс остаётся активным до закрытия окна пользователем.
- Таким образом, код демонстрирует базовые принципы мониторинга транспортного средства и безопасности, обеспечивая обновление данных в реальном времени и оперативное реагирование на критические ситуации.

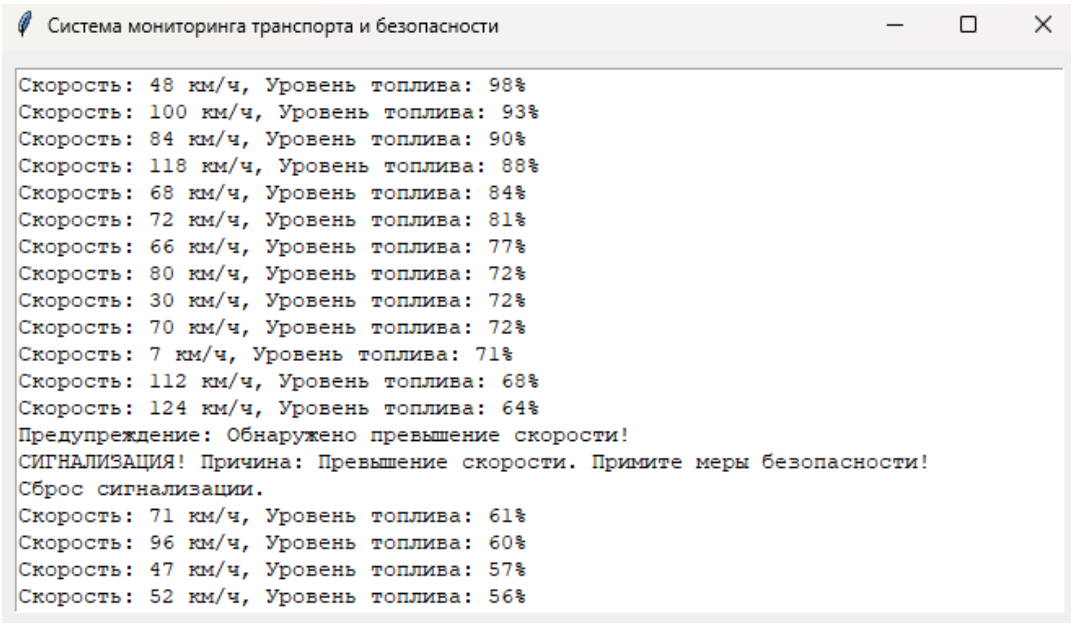

Рисунок 1 – Симуляция разный ситуаций

3. Преимущества и возможности расширения

Приложение демонстрирует базовый прототип системы мониторинга транспортного средства. Среди его преимуществ можно выделить:

Простота и наглядность:

Простой дизайн и понятный вывод данных позволяют легко оценить работоспособность системы.

Модульность кода:

Разделение на отдельные функции и классы облегчает дальнейшую поддержку и масштабирование.

Использование потоков:

Параллельное выполнение задач позволяет реализовать более сложные сценарии работы системы без снижения производительности графического интерфейса.

Интеграция с реальными датчиками:

Подключение к аппаратному обеспечению для получения реальных данных о состоянии транспортного средства.

Асинхронное программирование:

Переход к использованию асинхронных вызовов для повышения масштабируемости и отзывчивости системы.

Расширение функционала GUI:

Добавление дополнительных элементов управления, таких как кнопки для настройки параметров, графики изменения скорости и уровня топлива, а также уведомления в виде всплывающих окон.

Хранение и анализ данных:

Логирование результатов работы системы в базу данных для последующего анализа и прогнозирования потенциальных неисправностей.

Заключение

Разработанное приложение демонстрирует эффективность интеграции программных решений для контроля технического состояния транспорта и обеспечения безопасности дорожного движения. Реализованная система, разработанная на языке Python и библиотеке Tkinter, позволяет непрерывно отслеживать ключевые параметры, такие как скорость и уровень топлива, и оперативно реагировать на отклонения от установленных норм.

Применение случайной генерации данных для имитации работы датчиков иллюстрирует принцип работы микропроцессорной системы в условиях моделирования реальных ситуаций. Такой подход предоставляет возможность не только для обучения, но и для дальнейшей разработки реальных систем мониторинга.

Ключевым аспектом приложения является использование параллельных вычислений посредством потоков, что обеспечивает постоянную отзывчивость графического интерфейса. Это позволяет вести мониторинг в режиме реального времени без задержек, что критически важно при обнаружении аварийных ситуаций. Благодаря модульной структуре, реализованной в классе Vehicle Monitor, приложение легко расширяется и адаптируется под различные сценарии эксплуатации, включая интеграцию с реальными датчиками и внешними системами безопасности.

Модульность кода и четкая логика работы системы открывают широкие возможности для дальнейшего развития проекта. Планируемые направления совершенствования включают расширение функционала графического интерфейса, внедрение асинхронного программирования для повышения масштабируемости, а также интеграцию с базами данных для накопления и анализа эксплуатационных данных. Эти доработки позволят создать более надежную и адаптивную систему, способную обеспечить высокий уровень безопасности и повысить оперативность реагирования на критические изменения в состоянии транспортного средства.

В заключение можно отметить, что предложенное решение является важным шагом на пути к созданию комплексных систем мониторинга и управления транспортом. Оно демонстрирует потенциал использования современных технологий для повышения безопасности дорожного движения и создания эффективных методов контроля, что имеет решающее значение в условиях современного развития транспортной инфраструктуры. Дальнейшее развитие и интеграция подобных систем способствуют не только снижению риска аварийных ситуаций, но и повышению общего уровня безопасности на дорогах.

Список использованной литературы

1. Баландин, О. В. Интеллектуальные транспортные системы. — СПб.: Питер, 2019. — 320 с.
2. Горбачев, В. А. Программирование на Python: от основ к искусственному

интеллекту. — М.: Бином, 2021. — 416 с.

3. Кнут, Д. Искусство программирования. Том 1. Основные алгоритмы. — М.: Диалектика, 2020. — 752 с.

4. Тэненбаум, Э. Компьютерные сети. — СПб.: Питер, 2018. — 960 с.

5. Документация Python: https://docs.python.org/ru/3/

6. Документация библиотеки Tkinter: https://docs.python.org/3/library/tkinter.html

Документация по многопоточности в Python: https://docs.python.org/3/library/threading.html