# МЕТОДЫ ВВОДА ДАННЫХ В МИКРОЯДЕРНЫХ ОПЕРАЦИОННЫХ СИСТЕМАХ НА ПРИМЕРЕ ОПЕРАЦИОННОЙ СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ «БАГЕТ»

**А. М. Гиацинтов[а], А. Н. Еременчук[б], К. А. Мамросенко[в], К. В. Пугин[г]**
*Федеральное государственное бюджетное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт», Москва, Российская Федерация*
[а] *ORCID: https://orcid.org/0000-0002-1304-4016, giatsintov@niisi.ras.ru*
[б] *alexer@niisi.ras.ru*
[в] *ORCID: https://orcid.org/0000-0002-9889-0560, mamrosenko_k@niisi.ras.ru*
[г] *ORCID: https://orcid.org/0000-0002-0053-6230, rilian@@niisi.ras.ru*

*Аннотация:* в статье рассматривается разработка подсистемы ввода для отечественной микроядерной операционной системы реального времени (ОСРВ) «Багет». Ключевой проблемой при адаптации графических сред, таких как X Window System, для ОСРВ является недетерминированный характер стандартных решений, основанных на стеке udev/libinput. В работе предложен алгоритм «горячего подключения» (hotplug) устройств ввода, не зависящий от подсистемы udev, интегрированный в отдельный поток ввода X сервера и обеспечивающий предсказуемое время отклика. Алгоритм основан на периодическом сканировании файловой системы /dev и корректном взаимодействии с Application Programming Interface (API) X сервера через механизм обратных вызовов и синхронизации. Также представлена подсистема ввода для пользовательских приложений, обеспечивающая абстракцию над низкоуровневыми представлениями устройств, применяемая для приложений, не задействующих X сервер. В рамках работы реализовано приложение калибровки сенсорных экранов, использующее пятиточечную схему и метод наименьших квадратов для вычисления параметров аффинного преобразования, с последующей автоматической генерацией конфигурации для X сервера. Проведенные испытания подтверждают работоспособность, детерминированность и практическую применимость всех компонентов, включая интеграцию с фреймворком графического интерфейса пользователя (GUI) Nuklear. Работа демонстрирует возможность успешной адаптации сложных программных комплексов для использования в специализированных детерминированных системах.

*Ключевые слова:* ОСРВ, X сервер, графическая система, подсистема ввода, тачскрин.

*Благодарности:* публикация выполнена в рамках НИР НИЦ «Курчатовский институт» — НИИСИ по теме № FNEF-2024-0003 «Методы разработки аппаратно-программных платформ на основе защищенных и устойчивых к сбоям систем на кристалле и сопроцессоров искусственного интеллекта и обработки сигналов».

*Для цитирования:* Гиацинтов А. М., Еременчук А. Н., Мамросенко К. А., Пугин К. В. Методы ввода данных в микроядерных операционных системах на примере операционной системы реального времени «Багет». *Успехи кибернетики.* 2026;7(2):59–65.

*Поступила в редакцию: 08.04.2026. В окончательном варианте: 12.06.2026.*

# DATA INPUT METHODS IN MICROKERNEL OPERATING SYSTEMS: A CASE STUDY OF THE BAGET REAL-TIME OPERATING SYSTEM

**A. M. Giatsintov[a], A. N. Eremenchuk[b], K. A. Mamrosenko[c], K. V. Pugin[d]**
*Scientific Research Institute for System Analysis of the National Research Centre "Kurchatov Institute", Moscow, Russian Federation*
[a] *ORCID: https://orcid.org/0000-0002-1304-4016, giatsintov@niisi.ras.ru*
[b] *alexer@niisi.ras.ru*
[c] *ORCID: https://orcid.org/0000-0002-9889-0560, mamrosenko_k@niisi.ras.ru*
[d] *ORCID: https://orcid.org/0000-0002-0053-6230, rilian@@niisi.ras.ru*

*Abstract:* we developed an input subsystem for the Baget microkernel real-time operating system (RTOS). We studied the adaptation of graphical environments, including the X Window System,

to real-time systems and identified the non-deterministic behavior of conventional solutions based on the udev/libinput stack as a major challenge. To address this issue, we developed a hot-plug mechanism for input devices that operates independently of udev, is integrated into a dedicated X server input thread, and provides predictable response times. The proposed mechanism relies on periodic scanning of the /dev file system and interaction with the X server application programming interface (API) through callbacks and synchronization mechanisms. We also developed an input subsystem for user applications that abstracts low-level device interfaces and can be used independently of the X server. In addition, we implemented a touchscreen calibration application based on a five-point calibration procedure and the least-squares method to determine the parameters of an affine transformation. The application automatically generates the corresponding X server configuration. We evaluated all developed components through a series of tests. The results confirm their operability, deterministic behavior, and practical applicability, including integration with the Nuklear graphical user interface framework. The study demonstrates the feasibility of adapting complex software systems for use in specialized deterministic environments.

*Keywords:* real-time operating system, X server, graphics system, input system, touchscreen.

*Acknowledgements:* this study is a part of the NRC “Kurchatov Institute” – SRISA research project No. FNEF-2024-0003.

*Cite this article:* Giatsintov A. M., Eremenchuk A. N. , Mamrosenko K. A., Pugin K. V. Data Input Methods in Microkernel Operating Systems: A Case Study of the Baget Real-Time Operating System. *Russian Journal of Cybernetics*. 2026;7(2):59–65.

*Original article submitted: 08.04.2026.* *Revision submitted: 12.06.2026.*

**Введение**

Современные операционные системы требуют надежных и эффективных методов ввода данных, особенно в контексте специализированных систем, таких как операционная система реального времени (ОСРВ). Графическая система и подсистемы ввода являются ключевыми компонентами, обеспечивающими взаимодействие пользователя с приложениями. Многие известные научные работы сосредоточены на подсистеме вывода на экран и отображения графики на ОСРВ [1, 2], тогда как при необходимости интерактивной работы систем на базе ОСРВ требуется также и подсистема ввода, которая в ОС ОН, таких как Linux, решена через udev, libinput, systemd и другие механизмы, малоприменимые в условиях аппаратных ограничений и необходимости обработки подключения произвольных устройств ввода (так называемый механизм «hotplug»). Работы, посвященные портированию X Server на встраиваемые системы [3], часто обходят проблему динамического подключения устройств или предлагают решения, неприменимые в средах с жесткими временными ограничениями.

Настоящая работа восполняет этот пробел, представляя комплексное решение для организации подсистемы ввода в ОСРВ «Багет». В статье авторы пытаются ответить на вопрос обеспечения «отзывчивого» и детерминированного ввода данных при использовании графической подсистемы в ОСРВ, включая динамическое подключение устройств и точную работу сенсорных экранов в условиях отсутствия стандартных для ОС ОН механизмов.

По мнению авторов, научная новизна работы заключается в разработанном алгоритме «горячего подключения устройств» (hotplug) для X Server, не зависящем от стандартных подсистем Linux и обеспечивающем предсказуемое время отклика.

Практическая значимость работы состоит в следующем:

- создано программное обеспечение для калибровки тачскрина, использующее метод наименьших квадратов для компенсации систематических погрешностей, что важно для точности ввода в человеко-машинных интерфейсах;
- создан прототип подсистемы ввода для отечественной ОСРВ «Багет», обеспечивающий обработку управляющих воздействий для интерактивных приложений, использующих графическую систему данной ОС (в том числе и аппаратное ускорение на GPU).

**Анализ источников**

Современные операционные системы реального времени, особенно построенные на микроядерной архитектуре, сталкиваются с уникальной проблемой: необходимостью совмещать интер-

активность с жесткими требованиями к детерминизму и предсказуемому времени отклика [4, 5]. Разработка подсистемы ввода в таких условиях требует разработки специализированных подходов, если компромисс по снижению параметров вроде latency недопустим.

Архитектура X Window System, несмотря на известные особенности, остается практичным выбором для ОСРВ благодаря относительно низкой зависимости от подсистем ядра и удобству портирования. В отличие от Wayland, который в своей эталонной реализации libwayland требует нестандартного для POSIX системного вызова epoll() и интеграции в подсистему Direct Rendering Management, требующую полного портирования из Linux [6, 7] — X сервер использует отдельные драйверы взаимодействия с операционной системой (DDX), что особенно ценно при портировании на ОС, архитектура и функциональность которых может значительно отличаться от Linux, например, микроядерные ОСРВ.

Wayland по умолчанию опирается на event-driven модель с неявной синхронизацией через vsync и commit-буферы, что вносит джиттер задержки ввода [8] и даже в случае применения протокола linux-explicit-synchronization-unstable-v1 требует от системы поддержки многих интерфейсов, имеющихся только в Linux, что, по сравнению с X11, ограничивает его применимость в микроядерных ОСРВ. Но портирование X Server тоже не является бесшовным — важной задачей становится подсистема ввода. В дистрибутивах Linux и других схожих ОС ОН она реализована через многоуровневый стек: ядро (evdev), демон udev и библиотека libinput, которая обеспечивает универсальную обработку событий от всех устройств ввода [9–11]. Такая архитектура подсистемы ввода является принципиально недетерминированной из-за своего событийного подхода, что делает ее неприменимой на микроядерных ОСРВ. В ОС, где «все, кроме управления памятью, планирования и IPC, вынесено в пользовательское пространство» [12], таких как NuttX, используются собственные, сильно упрощенные подсистемы ввода, ориентированные на статическую конфигурацию, что подчеркивает пробел в решениях, обеспечивающих динамичность и детерминизм. В средах без динамического заполнения базы устройств, аналогичной udev, hotplug должен быть реализован на уровне драйвера, но конкретные решения для ОСРВ не предлагаются [11, 13]. RTOS на базе Linux (например, Enea Linux) предлагают утилиты для статической настройки устройств для отхода от использования udev, но динамический, детерминированный hotplug остается малоисследованной областью. В работе предлагается алгоритм периодического детерминированного сканирования, интегрированный в архитектуру InputThread, что обеспечивает предсказуемость поведения системы и тем самым способствует созданию hotplug-подсистем для ОСРВ.

**Определение доступных устройств ввода**

Одним из базовых требований к современным системам общего назначения является поддержка технологии hotplug, позволяющей прозрачно подключать и отключать периферийные устройства без перезагрузки системы или перезапуска оконной системы.

Однако в среде ОСРВ «Багет» использование подобных механизмов зачастую неприемлемо. Их работа носит недетерминированный, асинхронный и событийно-управляемый характер, что может вносить непредсказуемые задержки, нарушающие временные ограничения реального времени. Таким образом, требуется создание реализации hotplug для X Server, которая, с одной стороны, соответствует стандартной архитектуре и логике работы X.org, а с другой — отвечает жестким требованиям ОСРВ на детерминированное выполнение.

Стандартная точка интеграции кода hotplug в X Server тесно связана с его внутренним механизмом многопоточной обработки ввода. Следовательно, успешная реализация требует не только создания нового детерминированного алгоритма опроса, но и его корректного внедрения в существующую архитектуру потоков X Server.

Решение данной проблемы было найдено в переходе на многопоточную архитектуру с использованием выделенного потока ввода. Данная функциональность активируется флагом *--enable-input-thread* при сборке сервера. В этом режиме создается отдельный поток, который берет на себя роль обработчика ввода: он непрерывно в цикле опрашивает файловые дескрипторы всех устройств, читает поступающие события и помещает их в очередь для последующей обработки главным потоком. Это позволяет совместить преимущества обоих подходов: низкую задержку (поскольку поток ввода практически всегда готов к чтению) и отсутствие ограничений сигнальной

безопасности (чтение происходит в контексте обычного потока, а не обработчика сигнала).

Для ОСРВ «Багет» задача реализации hotplug сводится к созданию механизма, способного:

1. Обнаруживать факт подключения или отключения устройства ввода.
2. Идентифицировать тип устройства (мышь, клавиатура).
3. Уведомлять X Server об изменении конфигурации устройств.
4. Обеспечивать детерминированное время отклика.
5. Интегрироваться в существующую архитектуру потоков X Server.

Предложенное решение основано на создании подсистемы мониторинга (device monitor), которая осуществляет периодическое сканирование файловой системы */dev* для обнаружения изменений. Алгоритм использует следующие ключевые структуры данных:

```
/* Статус устройства в системе мониторинга */
enum devmon_device_status{
        DEVMON_DEVICE_STATUS_UNKNOWN = 0,// Статус не определен
        DEVMON_DEVICE_STATUS_OFF = 1,// Устройство отсутствует
        DEVMON_DEVICE_STATUS_ON = 2,// Устройство присутствует
};
struct devmon_data {
        InputInfoPtr *pInfo; // Указатель на информацию об устройстве (X Server)
        DeviceIntPtr dev; // Указатель на логическое устройство (X Server)
        char* device_path; // Путь к устройству в файловой системе
};
```

Процесс работы подсистемы можно разделить на три этапа.

Этап 1. Инициализация.

1. Вызывается функция *devmon_initialize()* для инициализации библиотеки мониторинга с заданными параметрами (максимальное количество callback-функций и т.д.).

2. На основе конфигурационных файлов X Server определяются количество и типы устройств (мыши, клавиатуры), для которых требуется мониторинг.

3. Выделяются массивы для хранения идентификаторов мониторов (*mouse_monitors[]*, *kbd_monitors[]*) и массивы ассоциированных с ними данных (*mouse_data[]*, *kbd_data[]*).

Этап 2. Регистрация и запуск мониторинга.

Для каждого устройства вызывается функция *devmon_monitor_device()*, которая:

- принимает путь к устройству в */dev*, указатель на функцию обратного вызова и указатель на пользовательские данные (структуру *devmon_data*);
- возвращает уникальный идентификатор монитора ($\geq$1) при успешной регистрации или -1 в случае ошибки. Запускается отдельный поток мониторинга, который выполняет периодическое сканирование.

Этап 3. Цикл сканирования и обработки.

В бесконечном цикле с детерминированным интервалом (например, 1 секунда) поток мониторинга выполняет:

1. Открытие и чтение содержимого директории */dev*.

2. Для каждого зарегистрированного устройства проводится проверка его наличия: проверяется каждое устройство в директории; если имя устройства в директории совпадает с требуемым, то устройство считается включенным.

3. Если состояние устройства изменилось, вызывается зарегистрированная callback-функция с соответствующим статусом (*DEVMON_DEVICE_STATUS_ON* или *DEVMON_DEVICE_STATUS_OFF*).

Для интеграции с X сервером применяется функция обратного вызова *device_callback()*.

При получении статуса *DEVMON_DEVICE_STATUS_ON* callback-функция:

- записывает в журнал обнаружение нового устройства;
- вызывает *NewInputDeviceRequest()* для регистрации нового устройства в X Server, что приводит к его инициализации и добавлению в список активных устройств.

При получении статуса *DEVMON_DEVICE_STATUS_OFF* callback-функция:

• записывает в журнал удаление устройства;

• вызывает *DeleteInputDeviceRequest()* для корректного удаления устройства из внутренних структур X Server и освобождения ресурсов.

Важно отметить, что все вызовы API X Server (NewInputDeviceRequest, DeleteInputDeviceRequest) должны выполняться в главном потоке сервера. Поскольку наш мониторинг работает в отдельном потоке, необходима корректная синхронизация, например, с помощью механизма отложенных действий или очереди сообщений, чтобы избежать состояний гонки.

Алгоритм включает комплексную обработку ошибок на всех этапах: проверка возвращаемых значений системных вызовов, валидация указателей, корректная обработка критических сбоев. При завершении работы X Server или отключении модуля вызываются функции *devmon_ release_ monitoring()* для каждого монитора и *devmon_ release()* для финального освобождения всех ресурсов библиотеки мониторинга.

**Поддержка тачскрина и калибратор**

Еще одной задачей была поддержка тачскрина в X сервере на ОСРВ «Багет». Теоретически точка прикосновения $P'$, регистрируемая контроллером сенсора с координатами ($X'$, $Y'$), должна точно соответствовать точке $P$ с координатами ($X$, $Y$) на дисплее. На практике между ними существует систематическая ошибка, обусловленная тремя основными факторами:

1. Масштабные коэффициенты $(\boldsymbol{k_X}, \boldsymbol{k_Y})$. Возникают из-за различия в разрешении дисплея и сенсорного контроллера. Например, для дисплея 1024 × 768 и 12-битного АЦП контроллера (4096 отсчетов) коэффициенты составят $k_X = 1024/4096 = 0.25$ и $k_Y = 768/4096 = 0.1875$.
2. Смещение $(\boldsymbol{\Delta X}, \boldsymbol{\Delta Y})$. Вызвано относительным сдвигом центров координатных систем дисплея и сенсора.
3. Поворот $(\boldsymbol{\Delta\theta})$. Обусловлен угловым смещением осей сенсора и дисплея.

Для перехода от «сырых» координат сенсора $(X', Y')$ к скорректированным координатам дисплея $(X, Y)$ применяется аффинное преобразование, позволяющее учесть все перечисленные погрешности единой системой линейных уравнений [14]:

$$X = \alpha_X \times X' + \beta_X \times Y' + \Delta X$$

$$Y = \alpha_Y \times X' + \beta_Y \times Y' + \Delta Y,$$

где коэффициенты $\alpha_X, \beta_X, \alpha_Y, \beta_Y, \Delta X, \Delta Y$ являются функциями от масштабных коэффициентов, угла поворота и смещения $(k_X, k_Y, \Delta\theta, \Delta X, \Delta Y)$. Именно вычисление этих шести коэффициентов является целью процедуры калибровки.

Для нахождения шести неизвестных коэффициентов требуется система уравнений, составленная на основе измерений в нескольких контрольных точках.

Трехточечная калибровка. Минимально необходимое количество точек — три, не лежащие на одной прямой. Решение сводится к вычислению обратной матрицы. Данный метод эффективен в системах с низким уровнем шума.

Многоточечная калибровка ($n > 3$). Для повышения устойчивости алгоритма к шумам измерения применяют калибровку по большему числу точек, что приводит к переопределенной системе уравнений. Оптимальное решение, минимизирующее среднеквадратичную ошибку, находится с помощью метода наименьших квадратов через псевдообратную матрицу Мура–Пенроуза [15]. Этот подход статистически усредняет погрешности измерений, повышая точность и надежность результата.

В рамках работы создано приложение калибровки, архитектура которого построена по модульному принципу и включает три компонента: модуль управления процессом калибровки, вычислительный модуль и графический интерфейс пользователя.

Практическая реализация алгоритма включает следующие шаги:

1. На дисплей выводятся $n$ калибровочных меток с известными координатами.
2. Пользователь последовательно касается каждой метки.
3. Для каждого касания фиксируются усредненные «сырые» координаты.

4. После сбора данных вызывается вычислительная процедура: при $n = 3$ вычисляется точное решение системы линейных уравнений. При $n > 3$ применяется метод наименьших квадратов для нахождения оптимальных коэффициентов.

5. Коэффициенты сохраняются в энергонезависимой памяти для последующей коррекции координат каждого касания в реальном времени.

В реализованном приложении используется $n = 5$ (т.н. пятиточечная система), что обеспечивает более высокую точность по сравнению с меньшими значениями $n$.

Для решения системы уравнений применяется библиотека матричных вычислений, реализованная в вычислительном модуле: расчет производится через псевдообратную матрицу. После успешного завершения калибровки приложение генерирует конфигурационный фрагмент для файла **xorg.conf** с рассчитанными $(\alpha_X, \beta_X, \alpha_Y, \beta_Y, \Delta X, \Delta Y)$, обеспечивая тем самым постоянную коррекцию ввода на системном уровне. Контроль качества калибровки происходит при помощи подсистемы проверки пользовательского ввода, включающей определение двойных и ошибочных касаний с последующим автоматическим сбросом процедуры.

**Результаты и обсуждение**

Для проверки работоспособности подсистемы ввода, используемой вместе с графической системой, была произведена интеграция с библиотекой GLFW, используемой для создания 3D-приложений на ОСРВ «Багет». Также было создано тестовое приложение, использующее библиотеку GLFW и GUI-фреймворк Nuklear, позволяющее вводить текст, отслеживающее и отображающее нажатие клавиш клавиатуры, а также определяющее перемещение и нажатие клавиш мыши. Данное приложение позволило отладить работу подсистемы ввода: на данный момент работает обработка мыши и нажатия клавиш клавиатуры, а также все стандартные методы ввода текста, кроме обработки зажатых клавиш.

Предложенный механизм hotplug для устройств ввода в X Server обходит зависимость от стандартного стека udev/libinput за счет реализации алгоритма периодического сканирования файловой системы.

Ключевыми особенностями решения являются:

1. Детерминированность: алгоритм с предсказуемым временем выполнения, что критично для ОСРВ.

2. Интеграция с потоком ввода: реализация корректно встроена в многопоточную архитектуру X Server, требующую сборки с флагом *--enable-input-thread*.

3. Надежность: включает комплексную обработку ошибок и корректное управление ресурсами.

4. Детерминированный hotplug: поддерживает динамическое добавление и удаление устройств, число которых не влияет на предсказуемость времени выполнения алгоритма.

Проведенные тесты подтвердили заявленные особенности: устройства типа «мышь» и «клавиатура» корректно добавляются в систему и удаляются из нее в течение заданного интервала опроса (1 секунда). Вносимые алгоритмом задержки являются полностью предсказуемыми и не оказывают негативного влияния на детерминированность выполнения задач реального времени в ОСРВ «Багет». Следовательно, данная реализация демонстрирует возможность адаптации X Server для работы в средах со строгими требованиями к детерминизму и с требованиями к возможности пользовательского ввода с графического интерфейса.

**Заключение**

Комплекс решений, представленный в данной статье, может быть применен за рамками ОСРВ «Багет», в других детерминированных средах, особенно в ОСРВ с микроядром, имеющих структурные и архитектурные сходства с ОСРВ «Багет». Статья показывает экспериментальную возможность адаптации сложных программных комплексов для работы в детерминированных средах, не основанных на базе Linux и имеющих аппаратные и программные ограничения. Этот подход может быть применен не только для X Server, но и для другого графического ПО на платформах реального времени, например, бортовых индикаторах на транспорте, промышленных контроллерах, медицинском оборудовании, носимых устройствах.

Также в работе представлен прототип программного обеспечения калибровки сенсорных экранов. Он полнофункционально применим, используется широко известная аффинная модель и метод наименьших квадратов для компенсации погрешностей, также имеет механизмы валидации ввода. Прототип прошел апробацию на реальных системах на базе ОСРВ «Багет».

Дальнейшее развитие архитектуры системы может быть направлено как на доработку текущей функциональности и снижение задержек в подсистеме ввода, так и на расширение функциональности в части поддержки устройств, а также безопасности и разделения доступа.

## ЛИТЕРАТУРА


1. Feske N., Hartig H. DOpE — A Window Server for Real-Time and Embedded Systems. *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS 2003, Cancun, Mexico)*. 2003:74–77. DOI: 10.1109/REAL.2003.1253255.
2. Giatsintov A. M., Mamrosenko K. A., Bazhenov P. A. Architecture of Graphics System with 3D Acceleration Support for Embedded Operating Systems. *Tsinghua Science and Technology*. 2024;29(3):863–873. DOI: 10.26599/TST.2023.9010045.
3. Zhadchenko A. V., Mamrosenko K. A., Giatsintov A. M. Porting X Windows System to Operating System Compliant with Portable Operating System Interface. *International Journal of Advanced Computer Science and Applications*. 2020;11(7):17–22.
4. Von Arnim C., Gessner G., Jarwitz M., Lechler A., Riedel O. Updating the Linux TAPRIO Scheduler in Deterministic Time. *Proceedings of the 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA, Stuttgart, Germany)*. 2022:1–7. DOI: 10.1109/ETFA52439.2022.9921594.
5. Yang C.-F., Shinjo Y. Obtaining Hard Real-Time Performance and Rich Linux Features in a Compounded Real-Time Operating System by a Partitioning Hypervisor. *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2020, Lausanne, Switzerland)*. 2020:59–72. DOI: 10.1145/3381052.3381329.
6. Manrique D. *X Window System Architecture Overview*. Режим доступа: https://tldp.org/HOWTO/pdf/XWindow-Overview-HOWTO.pdf.
7. *Wayland Architecture*. Режим доступа: https://wayland.freedesktop.org/architecture.html.
8. Dørum M. Wayland Input Latency. *GitLab Blog*. 2025. Режим доступа: https://gitlab.com/mort96/blog/-/blob/published/content/00000-home/00016-wayland-input-latency.md.
9. Input Subsystem Architecture. *The Linux Kernel Documentation*. Режим доступа: https://docs.kernel.org/input/input.html.
10. Udev. *ALT Linux Wiki*. Режим доступа: https://www.altlinux.org/Udev.
11. What Is libinput? *libinput Documentation*. Режим доступа: https://wayland.freedesktop.org/libinput/doc/latest/what-is-libinput.html.
12. ЭРЕМЕКС. *Разработка встроенных операционных систем реального времени*. Режим доступа: https://www.eremex.ru/upload/iblock/eb6/rtos_dev_book.pdf.
13. *Enea Linux Open Source Report*. Режим доступа: https://linux.enea.com/4.0/documentation/html/book-enea-linux-open-source/index.html.
14. Fang W., Chang T. *Calibration in Touch-Screen Systems*. Режим доступа: https://www.ti.com/lit/an/slyy137/slyy137.pdf.
15. Lewis F. L. *Optimal Estimation: With an Introduction to Stochastic Control Theory*. New York: John Wiley & Sons; 1986. 376 p.