ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ

Активная акустическая система Bluesound PULSE CINEMA (P530)

Класс: многоканальный стриминг-процессор для пост-продакшн и систем озвучивания

ЦЕЛЕВОЕ НАЗНАЧЕНИЕ (ПРОФЕССИОНАЛЬНОЕ НЕБЫТОВОЕ ПРИМЕНЕНИЕ)

Акустическая система Bluesound PULSE CINEMA (P530) **предназначена исключительно для профессионального небытового использования**.

Основные профессиональные сценарии применения:

Сфера применения	**Тип задач**
Пост-продакшн студии (post-production)	Контроль микса и мастеринга для кинопоказа, телевидения, стриминговых сервисов
Студии дубляжа и озвучивания	Оценка диалогов и звуковых эффектов в формате Dolby Atmos
Монтажные комнаты (edit suites)	Предварительное сведение и контроль совместимости
Профессиональные демонстрационные зоны (showrooms)	Демонстрация контента в форматах высокой четкости
Конференц-залы и презентационные комнаты	Озвучивание презентаций, видеоконференций
Образовательные учреждения	Лаборатории аудиотехники, классы звукорежиссуры

Сфера применения	Тип задач
Коммерческие AV-инсталляции	Системы озвучивания в барах, ресторанах, отелях, выставочных залах

Категорически не предназначено для: бытового использования в жилых помещениях.

Ключевые особенности для профессионального применения:

1. **Поддержка Dolby Atmos для профессионального контроля:** Система обеспечивает воспроизведение многоканальных форматов Dolby Atmos, Dolby TrueHD, Dolby Digital Plus, LPCM с высотными каналами, что критически важно для контроля контента, создаваемого для кинопоказа и стриминговых платформ.
2. **Программируемость через Python API:** Поддерживает управление и автоматизацию через Python-скрипты (BluOS API).
3. **Профессиональное качество звука:** Конфигурация 3.2.2 с 16 динамиками и суммарной мощностью 500 Вт.
4. **Высокое разрешение Hi-Res Audio:** Поддержка MQA, DSD256, FLAC, ALAC, WAV, AIFF с частотой дискретизации до 192 кГц / 24 бит.
5. **Профессиональные системные интеграции:** Поддержка Crestron, Control4, RTI, Nice, URC, Lutron.

---

1. ВВЕДЕНИЕ И ОБЩИЕ СВЕДЕНИЯ

Настоящая инструкция предназначена для **квалифицированного персонала**, использующего активную акустическую систему Bluesound PULSE CINEMA (P530) в профессиональных целях. Система является многоканальным сетевым стримером с возможностью программного управления через API.

Основные технические характеристики:

Параметр	Значение
Конфигурация	3.2.2 канала (Dolby Atmos с высотными каналами)

Параметр	Значение
Количество динамиков	16 (5 твитеров + 5 среднечастотных + 2 вуфера + 4 пассивных радиатора)
Мощность усилителя	500 Вт (интеллектуальный DSP усилитель)
Динамики (твитер)	0,75″ (21 мм) × 5 шт., мощность 38 Вт каждый
Динамики (СЧ)	1,5″ × 3″ (44 × 75 мм) × 5 шт., мощность 38 Вт каждый
Вуферы	4″ (102 мм) × 2 шт., мощность 65 Вт каждый
Пассивные радиаторы	4″ (102 мм) × 4 шт.
THD+N	0,030%
Частота дискретизации	до 192 кГц
Разрядность	16–24 бит

Подключения:

Тип	Назначение
HDMI eARC	Основное подключение к источнику
Оптический (Toslink)	Цифровой вход
Аналоговый (RCA)	Линейный вход
USB Type-A	Внешние накопители
SUBW OUT (RCA)	Активный сабвуфер

Тип	Назначение
LAN (Ethernet)	Gigabit (1000 Мбит/с)
Wi-Fi	802.11ac (Wi-Fi 5), 2.4/5 ГГц
Bluetooth	5.2 aptX Adaptive

---

2. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ

Перед эксплуатацией системы ознакомьтесь со следующими требованиями:

- **Электропитание:** Используйте прилагаемый кабель питания (в комплекте предусмотрены кабели для 120В и 230В). Выберите кабель, соответствующий вашему региону. Защищайте кабель питания от повреждений.
- **Влажность:** Не допускайте попадания жидкости на корпус. Не эксплуатируйте вблизи воды.
- **Вентиляция:** Обеспечьте свободный доступ воздуха. При установке на стену используйте прилагаемый кронштейн и шаблон для точного монтажа.
- **Температура:** Эксплуатируйте при температуре от 0 до 35°C.
- **Чистка:** Используйте только сухую мягкую ткань. Не применяйте абразивные средства.

⚠ **ПРОФЕССИОНАЛЬНОЕ ПРИМЕЧАНИЕ:** Система является сетевым аудиоустройством и зависит от правильно настроенной локальной сети.

---

3. РАСПАКОВКА И ПРОВЕРКА КОМПЛЕКТАЦИИ

1. Извлеките компоненты из упаковки. Сохраните оригинальную упаковку для транспортировки.
2. Проверьте комплектацию:

Компонент	Описание
PULSE CINEMA (P530)	Основной блок
Кабель питания 120 В	Для регионов с напряжением 120 В
Кабель питания 230 В	Для регионов с напряжением 230 В
Кабель Ethernet	Для проводного подключения к сети
Кабель HDMI	Для подключения eARC
Кронштейн для настенного монтажа	С крепежом и шестигранным ключом
Шаблон для настенного монтажа	Для точной разметки отверстий
Стопорные язычки (L и R)	Для дополнительной фиксации
Руководства	Краткое руководство, гарантия, инструкция по безопасности

3. Осмотрите корпус на предмет повреждений, полученных при транспортировке.

---

4. УСТАНОВКА И РАЗМЕЩЕНИЕ (ПРОФЕССИОНАЛЬНАЯ КОНФИГУРАЦИЯ)

4.1. Габаритные размеры

Вариант установки	Размеры (Д × В × Г)
Настольная (на ножках)	1200 × 74 × 140 мм
Настенный монтаж	1200 × 140 × 84,5 мм

Вес: 6,69 кг

4.2. Варианты размещения

Система оснащена **встроенным акселерометром**, который автоматически определяет ориентацию (настенная или настольная) и соответствующим образом настраивает DSP.

А. Настольная установка:

- Установите систему горизонтально на резиновые ножки
- Интерфейс верхней панели должен быть обращен к задней части продукта
- Оптимальное размещение — на уровне ушей оператора

Б. Настенный монтаж:

- Используйте прилагаемый кронштейн и шаблон для разметки
- Крепление возможно только в одной ориентации
- Система должна монтироваться максимально близко к стене

В. Универсальный монтаж:

- Совместима с универсальными и артикулирующими кронштейнами
- Используйте стопорные язычки (Mount Locking Tabs) для дополнительной безопасности

---

5. ПРОГРАММИРОВАНИЕ НА PYTHON (BLUOS API)

5.1. Общие сведения

Bluesound PULSE CINEMA поддерживает управление через **BluOS API** — RESTful веб-сервис, позволяющий программно управлять всеми функциями системы. API доступен через HTTP-запросы к встроенному веб-серверу устройства.

Базовые параметры подключения:

- **Протокол:** HTTP
- **Порт:** 80
- **IP-адрес:** IP-адрес устройства в локальной сети
- **Формат ответа:** XML или JSON (зависит от метода)

5.2. Определение IP-адреса устройства

Перед началом программирования необходимо определить IP-адрес PULSE CINAMA в сети:

```
python
import socket

def discover_bluesound_devices():
    """
    Обнаружение Bluesound устройств в локальной сети
    использует UPnP обнаружение (SSDP)
    """
    import requests
    import xml.etree.ElementTree as ET

    # SSDP запрос для обнаружения Bluesound устройств
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
    sock.settimeout(5)

    # Multicast адрес для SSDP
    MCAST_ADDR = "239.255.255.250"
    MCAST_PORT = 1900

    # Поисковый запрос для устройств Bluesound
    search_message = (
        "M-SEARCH * HTTP/1.1\r\n"
        "HOST: 239.255.255.250:1900\r\n"
        "MAN: \"ssdp:discover\"\r\n"
        "MX: 3\r\n"
        "ST: urn:schemas-upnp-org:device:MediaRenderer:1\r\n\r\n"
    ).encode()

    sock.sendto(search_message, (MCAST_ADDR, MCAST_PORT))

    devices = []
    try:
```

```
        while True:
            data, addr = sock.recvfrom(1024)
            response = data.decode()
            if "Bluesound" in response or "bluesound" in response.lower():
                # Извлечение Location URL
                for line in response.split('\r\n'):
                    if line.lower().startswith('location:'):
                        location = line.split(':', 1)[1].strip()
                        devices.append({'ip': addr[0], 'location': location})
                        break
    except socket.timeout:
        pass
    finally:
        sock.close()

    return devices

# Пример использования
devices = discover_bluesound_devices()
for device in devices:
    print(f"Найдено устройство: {device['ip']}")
```

5.3. Базовые API запросы

BluOS API использует HTTP GET запросы к эндпоинту /Sync:

```
python
import requests
import json
from typing import Dict, Any, Optional

class BluesoundController:
    """
    Класс для управления Bluesound PULSE CINEMA через Python
    """

    def __init__(self, ip_address: str):
        """
        Инициализация контроллера

        Args:
            ip_address: IP-адрес устройства в локальной сети
```

```
    """
    self.ip = ip_address
    self.base_url = f"http://{ip_address}:80"
    self._sync_endpoint = "/Sync"

def _send_command(self, command: str, params: Optional[Dict] = None) -> Dict[str, Any]:
    """
    Отправка команды к устройству

    Args:
        command: Команда API
        params: Параметры команды

    Returns:
        Ответ в виде словаря
    """
    url = f"{self.base_url}{self._sync_endpoint}"
    payload = {'cmd': command}
    if params:
        payload.update(params)

    try:
        response = requests.get(url, params=payload, timeout=5)
        response.raise_for_status()

        # API возвращает данные в формате "key=value" на каждой строке
        result = {}
        for line in response.text.strip().split('\n'):
            if '=' in line:
                key, value = line.split('=', 1)
                result[key] = value
        return result
    except requests.exceptions.RequestException as e:
        print(f"Ошибка связи с устройством: {e}")
        return {'error': str(e)}

# === Управление воспроизведением ===

def play(self) -> Dict[str, Any]:
    """Запуск воспроизведения"""
    return self._send_command("play")

def pause(self) -> Dict[str, Any]:
    """Пауза"""
```

```
    return self._send_command("pause")

def stop(self) -> Dict[str, Any]:
    """Остановка"""
    return self._send_command("stop")

def next_track(self) -> Dict[str, Any]:
    """Следующий трек"""
    return self._send_command("skip")

def previous_track(self) -> Dict[str, Any]:
    """Предыдущий трек"""
    return self._send_command("back")

# === Управление громкостью ===

def set_volume(self, volume: int) -> Dict[str, Any]:
    """
    Установка уровня громкости

    Args:
        volume: Значение громкости (0-100)
    """
    volume = max(0, min(100, volume))
    return self._send_command("volume", {"level": volume})

def volume_up(self, step: int = 5) -> Dict[str, Any]:
    """Увеличение громкости"""
    current = self.get_volume()
    if 'volume' in current:
        new_volume = min(100, int(current['volume']) + step)
        return self.set_volume(new_volume)
    return {'error': 'Cannot get current volume'}

def volume_down(self, step: int = 5) -> Dict[str, Any]:
    """Уменьшение громкости"""
    current = self.get_volume()
    if 'volume' in current:
        new_volume = max(0, int(current['volume']) - step)
        return self.set_volume(new_volume)
    return {'error': 'Cannot get current volume'}

def mute(self) -> Dict[str, Any]:
    """Отключение звука"""
```

```
    return self._send_command("mute")

def unmute(self) -> Dict[str, Any]:
    """Включение звука"""
    return self._send_command("unmute")

# === Получение информации ===

def get_volume(self) -> Dict[str, Any]:
    """Получение текущего уровня громкости"""
    return self._send_command("volume")

def get_state(self) -> Dict[str, Any]:
    """Получение состояния проигрывателя"""
    return self._send_command("state")

def get_status(self) -> Dict[str, Any]:
    """Получение полного статуса устройства"""
    return self._send_command("status")

def get_now_playing(self) -> Dict[str, Any]:
    """Получение информации о текущем треке"""
    return self._send_command("NowPlaying")

# === Управление входами ===

def select_input(self, input_name: str) -> Dict[str, Any]:
    """
    Выбор входного источника

    Args:
        input_name: Имя источника (hdmi, optical, analog, bluetooth)
    """
    input_map = {
        'hdmi': 'hdmi_arc',
        'optical': 'optical',
        'analog': 'analog_in',
        'bluetooth': 'bluetooth'
    }

    mapped_input = input_map.get(input_name.lower(), input_name)
    return self._send_command("select", {"input": mapped_input})

# === Управление режимами звука ===
```

```python
def set_listening_mode(self, mode: str) -> Dict[str, Any]:
    """
    Установка режима прослушивания

    Args:
        mode: Режим (movie, music, night)
    """
    return self._send_command("listeningMode", {"mode": mode})

def set_subwoofer(self, enabled: bool) -> Dict[str, Any]:
    """
    Включение/выключение сабвуфера

    Args:
        enabled: True - включен, False - выключен
    """
    state = "1" if enabled else "0"
    return self._send_command("subwoofer", {"state": state})

# === Управление предустановками ===

def press_preset(self, preset_number: int) -> Dict[str, Any]:
    """
    Активация предустановки

    Args:
        preset_number: Номер предустановки (1 или 2)
    """
    if preset_number not in [1, 2]:
        return {'error': 'Preset number must be 1 or 2'}
    return self._send_command(f"preset{preset_number}")

# === Управление группировкой (multi-room) ===

def group_devices(self, slave_ips: list) -> Dict[str, Any]:
    """
    Группировка нескольких устройств Bluesound

    Args:
        slave_ips: Список IP-адресов устройств для группировки
    """
    result = {}
    for slave_ip in slave_ips:
```

```
            group_url = f"http://{slave_ip}:80/Sync?cmd=group&master={self.ip}"
            try:
                response = requests.get(group_url, timeout=5)
                result[slave_ip] = response.status_code == 200
            except requests.exceptions.RequestException as e:
                result[slave_ip] = False
                result[f"{slave_ip}_error"] = str(e)
        return result

    def ungroup_devices(self) -> Dict[str, Any]:
        """Отмена группировки"""
        return self._send_command("ungroup")
```

5.4. Примеры использования

Пример 1: Базовое управление

python

```
# Создание экземпляра контроллера
controller = BluesoundController("192.168.1.100")

# Установка громкости
controller.set_volume(45)

# Выбор HDMI входа
controller.select_input("hdmi")

# Установка режима Movie для контроля киноконтента
controller.set_listening_mode("movie")

# Воспроизведение
controller.play()

# Получение информации о текущем треке
info = controller.get_now_playing()
print(f"Сейчас играет: {info.get('title', 'Неизвестно')}")

# Пауза
controller.pause()
```

Пример 2: Мониторинг состояния с логированием

```python
import time
from datetime import datetime

class BluesoundMonitor:
    """Класс для мониторинга состояния Bluesound устройства"""

    def __init__(self, ip_address: str, log_file: str = "bluesound_log.txt"):
        self.controller = BluesoundController(ip_address)
        self.log_file = log_file

    def log_state(self):
        """Логирование текущего состояния"""
        state = self.controller.get_state()
        volume = self.controller.get_volume()

        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        with open(self.log_file, "a") as f:
            f.write(f"[{timestamp}] Состояние: {state.get('state', 'unknown')}\n")
            f.write(f"[{timestamp}] Громкость: {volume.get('volume', 'unknown')}%\n")
            f.write("-" * 50 + "\n")

    def monitor(self, interval_seconds: int = 60):
        """Запуск мониторинга с заданным интервалом"""
        try:
            while True:
                self.log_state()
                time.sleep(interval_seconds)
        except KeyboardInterrupt:
            print("Мониторинг остановлен")

# Пример использования
monitor = BluesoundMonitor("192.168.1.100")
monitor.log_state()  # Однократное логирование
```

Пример 3: Интеграция с профессиональной системой управления (Crestron/Control4)

```python
import asyncio
import websockets
import json

class BluesoundWebSocketController:
```

```
"""
WebSocket контроллер для реального времени
Используется для интеграции с профессиональными системами управления
"""

def __init__(self, ip_address: str):
    self.ip = ip_address
    self.websocket_url = f"ws://{ip_address}:80/ws"
    self.websocket = None
    self.callbacks = []

async def connect(self):
    """Установка WebSocket соединения"""
    self.websocket = await websockets.connect(self.websocket_url)

async def subscribe_events(self):
    """Подписка на события устройства"""
    subscribe_msg = json.dumps({"type": "subscribe", "events": ["state", "volume", "track"]})
    await self.websocket.send(subscribe_msg)

async def listen_events(self):
    """Прослушивание событий"""
    async for message in self.websocket:
        event = json.loads(message)
        for callback in self.callbacks:
            await callback(event)

def add_event_handler(self, callback):
    """Добавление обработчика событий"""
    self.callbacks.append(callback)

async def set_volume_ramp(self, target_volume: int, duration_seconds: float):
    """
    Плавное изменение громкости (ramp)
    Полезно для профессиональных сценариев (постепенное затухание)

    Args:
        target_volume: Целевая громкость (0-100)
        duration_seconds: Длительность изменения в секундах
    """
    controller = BluesoundController(self.ip)
    current = int(controller.get_volume().get('volume', 0))
    steps = 20
    step_volume = (target_volume - current) / steps
```

```
        step_time = duration_seconds / steps

        for i in range(steps):
            new_volume = current + int(step_volume * (i + 1))
            controller.set_volume(new_volume)
            await asyncio.sleep(step_time)

# Пример асинхронного использования
async def main():
    ws_controller = BluesoundWebSocketController("192.168.1.100")
    await ws_controller.connect()
    await ws_controller.subscribe_events()

    # Плавное снижение громкости за 3 секунды
    await ws_controller.set_volume_ramp(20, 3.0)

# asyncio.run(main())
```

Пример 4: Создание сценариев автоматизации

python

```
class BluesoundAutomation:
    """Автоматизация профессиональных сценариев"""

    def __init__(self, ip_address: str):
        self.controller = BluesoundController(ip_address)

    def recall_session(self, session_preset: dict):
        """
        Восстановление сессии из предустановки

        Args:
            session_preset: Словарь с параметрами сессии
        """
        # Восстановление громкости
        if 'volume' in session_preset:
            self.controller.set_volume(session_preset['volume'])

        # Восстановление входа
        if 'input' in session_preset:
            self.controller.select_input(session_preset['input'])

        # Восстановление режима звука
        if 'listening_mode' in session_preset:
```

```
        self.controller.set_listening_mode(session_preset['listening_mode'])

    # Восстановление состояния сабвуфера
    if 'subwoofer' in session_preset:
        self.controller.set_subwoofer(session_preset['subwoofer'])

def create_mixing_session(self):
    """Создание сессии для сведения"""
    session = {
        'volume': 75,
        'input': 'hdmi',
        'listening_mode': 'movie',  # Максимальная точность для киноконтента
        'subwoofer': True
    }
    self.recall_session(session)

def create_voiceover_session(self):
    """Создание сессии для озвучивания (диалогов)"""
    session = {
        'volume': 65,
        'input': 'optical',
        'listening_mode': 'music',  # Более нейтральное звучание для диалогов
        'subwoofer': False
    }
    self.recall_session(session)

def create_calibration_sequence(self, reference_level_db: int = 85):
    """
    Создание калибровочной последовательности

    Args:
        reference_level_db: Опорный уровень звукового давления (дБ)
    """
    # Примечание: Для полной калибровки требуется измерительный микрофон
    # Здесь приведена логика установки уровня для калибровочных сигналов

    calibration_tones = [
        {"frequency": 1000, "duration": 5, "description": "Опорный тон 1 кГц"},
        {"frequency": 250, "duration": 3, "description": "Низкочастотный тон"},
        {"frequency": 8000, "duration": 3, "description": "Высокочастотный тон"},
        {"frequency": 40, "duration": 2, "description": "Инфранизкий тон"}
    ]

    # Установка опорного уровня (требует ручной настройки с SPL метром)
```

```
        print("Установите опорный уровень громкости (85 дБ SPL)")
        print("Используйте измерительный микрофон для калибровки")

        return calibration_tones

# Пример использования автоматизации
automation = BluesoundAutomation("192.168.1.100")
automation.create_mixing_session()
```

Пример 5: REST API сервер для удаленного управления

```
python
from flask import Flask, request, jsonify

app = Flask(__name__)

# Конфигурация устройств
DEVICES = {
    "studio_a": "192.168.1.100",
    "studio_b": "192.168.1.101",
    "control_room": "192.168.1.102"
}

@app.route('/api/device/<device_name>/volume', methods=['GET', 'POST'])
def handle_volume(device_name):
    """Управление громкостью через REST API"""
    if device_name not in DEVICES:
        return jsonify({"error": "Device not found"}), 404

    controller = BluesoundController(DEVICES[device_name])

    if request.method == 'GET':
        volume = controller.get_volume()
        return jsonify(volume)
    elif request.method == 'POST':
        data = request.get_json()
        if 'level' not in data:
            return jsonify({"error": "Missing 'level' parameter"}), 400
        result = controller.set_volume(data['level'])
        return jsonify(result)

@app.route('/api/device/<device_name>/play', methods=['POST'])
def handle_play(device_name):
    """Запуск воспроизведения"""
```

```
    if device_name not in DEVICES:
        return jsonify({"error": "Device not found"}), 404

    controller = BluesoundController(DEVICES[device_name])
    result = controller.play()
    return jsonify(result)

@app.route('/api/device/<device_name>/input', methods=['POST'])
def handle_input(device_name):
    """Выбор входного источника"""
    if device_name not in DEVICES:
        return jsonify({"error": "Device not found"}), 404

    data = request.get_json()
    if 'source' not in data:
        return jsonify({"error": "Missing 'source' parameter"}), 400

    controller = BluesoundController(DEVICES[device_name])
    result = controller.select_input(data['source'])
    return jsonify(result)

@app.route('/api/devices', methods=['GET'])
def list_devices():
    """Список всех устройств"""
    return jsonify(DEVICES)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

5.5. Справочник команд BluOS API

Команда	Описание	Параметры
volume	Управление громкостью	level (0-100)
mute	Отключение звука	-
unmute	Включение звука	-

Команда	Описание	Параметры
play	Воспроизведение	-
pause	Пауза	-
stop	Остановка	-
skip	Следующий трек	-
back	Предыдущий трек	-
state	Состояние проигрывателя	-
status	Полный статус	-
NowPlaying	Информация о текущем треке	-
select	Выбор входа	input (hdmi_arc, optical, analog_in, bluetooth)
group	Группировка устройств	master (IP мастер-устройства)
ungroup	Отмена группировки	-
preset1	Предустановка 1	-
preset2	Предустановка 2	-

6. ОРГАНЫ УПРАВЛЕНИЯ (СЕНСОРНАЯ ПАНЕЛЬ)

Элемент	Функция
Preset 1 / Preset 2	Две программируемые кнопки
Volume + / -	Сенсорные кнопки регулировки громкости
Play/Pause	Многофункциональная кнопка
Status LED	Светодиодный индикатор состояния

Цветовая индикация Status LED:

Код	Описание состояния
Зеленый (постоянный)	Режим «Hotspot» (готовность к подключению)
Зеленый (мигающий)	Подключение к сети
Синий (постоянный)	Подключен к сети — готов к работе
Белая вспышка	Доступно обновление ПО
Красный (постоянный)	Режим обновления
Чередование красный/зеленый	Обновление ПО в процессе

Код	Описание состояния
Красный (мигающий)	Сброс к заводским настройкам

---

7. ПОДКЛЮЧЕНИЯ И ИНТЕРФЕЙСЫ

7.1. HDMI eARC

- **Назначение:** Подключение к источнику с поддержкой eARC/ARC
- **Отображение в приложении:** HDMI ARC или eARC

7.2. Оптический вход (Toslink)

- **Назначение:** Подключение цифровых источников
- **Отображение в приложении:** Optical Input

7.3. Аналоговый вход (RCA)

- **Назначение:** Подключение линейного выхода
- **Отображение в приложении:** Analog Input

7.4. Выход сабвуфера (SUBW OUT)

- **Тип разъема:** RCA моно
- **Частота кроссовера:** 90 Гц

7.5. USB (Type-A)

- **Назначение:** Подключение внешних накопителей
- **Форматирование:** FAT32 или NTFS

7.6. LAN порт (Ethernet)

- **Скорость:** Gigabit (1000 Мбит/с)

---

8. НАСТРОЙКИ ЗВУКА

8.1. Режимы прослушивания

Режим	Применение
Movie	Оптимизирован для киноконтента, подчеркивает эффекты и диалоги
Music	Нейтральный режим для музыкального контента
Late Night	Снижает динамический диапазон

8.2. Функции обработки звука

Функция	Описание
Subwoofer	Включение/выключение выхода сабвуфера
Surround Upmixer	Преобразование в объемное звучание
Virtualizer	Расширение звуковой сцены

Функция	Описание
Volume Leveller	Поддержание постоянного уровня громкости

---

9. ПРОФЕССИОНАЛЬНЫЕ ФУНКЦИИ

9.1. Автоматическая ориентация (акселерометр)

Встроенный акселерометр автоматически определяет способ установки и переназначает каналы для корректного воспроизведения Dolby Atmos.

9.2. ИК-обучение (IR Learning)

Позволяет обучить систему работать со сторонним ИК-пультом управления.

9.3. Автоматическое включение/выключение

При использовании HDMI eARC система автоматически включается и выключается вместе с источником.

9.4. Bonded Speaker Link

Дополнительный Wi-Fi модуль для связи с беспроводными тыловыми динамиками и сабвуфером.

9.5. Профессиональные системные интеграции

Поддержка систем управления: **Crestron, Control4, RTI, Nice, URC, Lutron**.

---

10. BLUOS APP (УПРАВЛЕНИЕ)

- **Поддерживаемые платформы:** iOS, Android, Windows, macOS
- **Управление многокомнатной системой (multi-room)**
- **Доступ к 23+ музыкальным сервисам**
- **Воспроизведение локальной медиатеки**

Поддерживаемые аудиоформаты:

Формат	Поддержка
MQA	Полная поддержка
DSD	DSD256
FLAC	До 192 кГц / 24 бит
ALAC	До 192 кГц / 24 бит
WAV	До 192 кГц / 24 бит
AIFF	До 192 кГц / 24 бит

---

11. ПОИСК И УСТРАНЕНИЕ НЕИСПРАВНОСТЕЙ

Проблема	Решение
Нет звука	Проверьте питание. Убедитесь, что выбран правильный вход. Проверьте настройки ARC/eARC на источнике

Проблема	Решение
Не удается подключиться к сети	Проверьте LED индикатор. Для Wi-Fi: проверьте пароль. Для Ethernet: проверьте кабель
Нет эффекта Dolby Atmos	Убедитесь, что контент имеет поддержку Dolby Atmos. Проверьте настройки источника
API не отвечает	Проверьте IP-адрес устройства. Убедитесь, что устройство находится в той же подсети
Ошибка Python запроса	Проверьте сетевое соединение. Убедитесь, что порт 80 открыт

---

12. СБРОС К ЗАВОДСКИМ НАСТРОЙКАМ

1. Убедитесь, что система включена.
2. Нажмите и удерживайте кнопку **Play/Pause** на верхней панели.
3. Удерживайте до тех пор, пока LED не начнет **мигать красным**.
4. Отпустите кнопку.
5. Система перезагрузится. LED станет **зеленым** (режим Hotspot).

Внимание: Сброс удаляет все пользовательские настройки (сеть, предустановки, настройки звука).

---

13. ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ И ГАРАНТИЯ

- **Уход за корпусом:** Протирайте мягкой сухой тканью
- **Обновление ПО:** Производится автоматически через BluOS App
- **Транспортировка:** Используйте оригинальную упаковку
- **Сервис:** При повреждениях обращайтесь к авторизованному дилеру

Гарантийный срок: 24 месяца

ПРИЛОЖЕНИЕ: ПОЛНЫЕ ТЕХНИЧЕСКИЕ СПЕЦИФИКАЦИИ

Параметр	Значение
Модель	PULSE CINEMA (P530)
Артикул	P530BLKUNV
Цвет	Черный (Black)
Габариты (настольная)	1200 × 74 × 140 мм
Габариты (настенный)	1200 × 140 × 84,5 мм
Вес	6,69 кг
Процессор	ARM Cortex-A53, Quad-Core, 1.8 ГГц
THD+N	0,030%
Частота кроссовера сабвуфера	90 Гц
API протокол	HTTP REST (порт 80)
Программирование	Python через BluOS API