

Стабилизация Энергетического ядра Draconic Evolution через OpenComputers

Этот гайд поможет вам настроить систему автоматического контроля энергии. Программа следит за тем, чтобы в хранилище всегда оставался необходимый запас для работы реакторов, предотвращая их перегрев или остановку. Если энергии слишком много - компьютер увеличивает выход, если мало - ограничивает его.

1. Необходимые компоненты

Компьютер (OpenComputers)

Компонент	Количество	Примечание
Системный блок	1 шт	Уровень 3
Монитор	6 шт	Уровень 3 (для сборки 3x2)
Видеокарта	1 шт	Уровень 3
Процессор (ЦП)	1 шт	Уровень 3
Память (ОЗУ)	2 шт	Уровень 3.5
Жесткий диск	1 шт	Уровень 3 (4 Мб)
EEPROM	1 шт	Прошивка Lua BIOS
Дискета	1 шт	С установщиком OpenOS
Адаптер	1 шт	Для связи с устройствами
Преобразователь энергии	1 шт	Питание компьютера
Клавиатура	1 шт	Для ввода команд

Ядро энергии

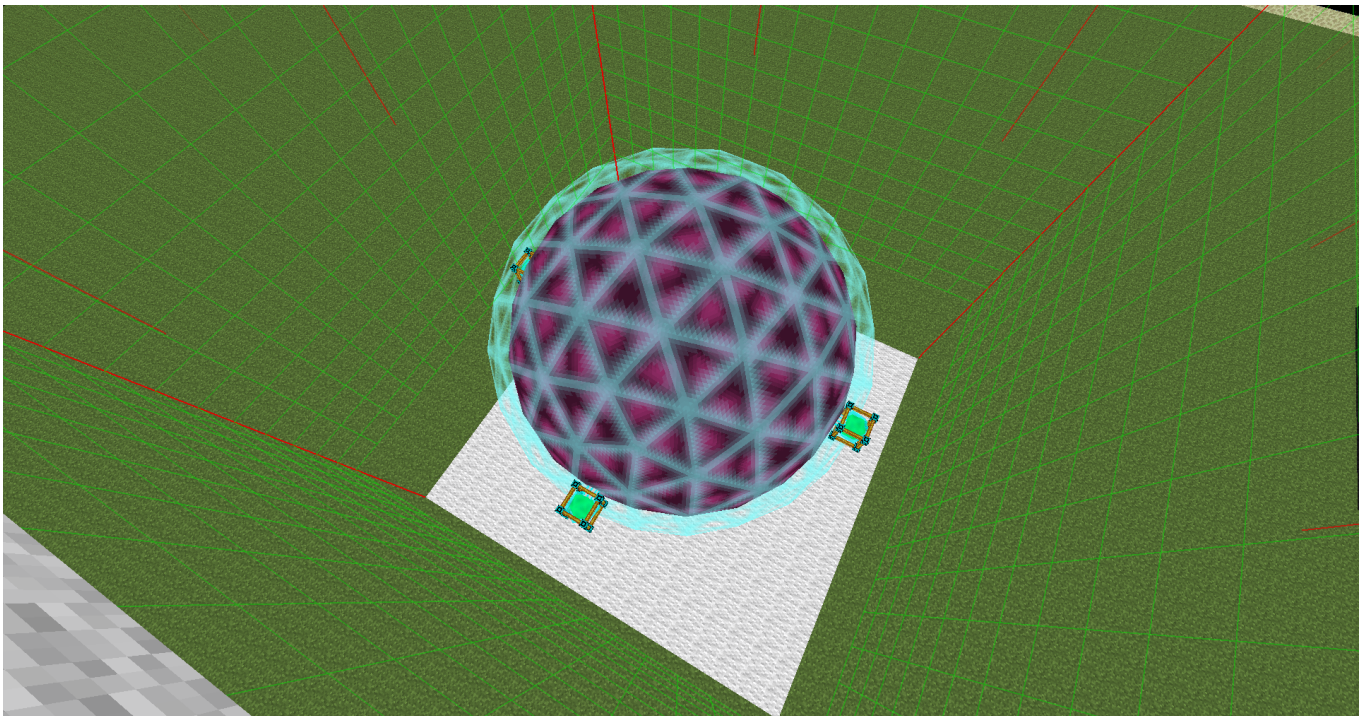
1. Само ядро энергии ([как его построить](#))
2. Флаксовый гейт - 1шт
3. Энергетический пилон - 1шт
4. Криостабилизированная флаксовая труба - смотреть по ситуации

2. Подготовка и строительство

Первым делом включите отображение границ чанков (клавиша **F9**). Крайне важно строить всю конструкцию (ядро и компьютер) **в пределах одного чанка**, чтобы избежать ошибок при загрузке мира. Инструкция по сборке ядра [тут](#). Схематика [тут](#).

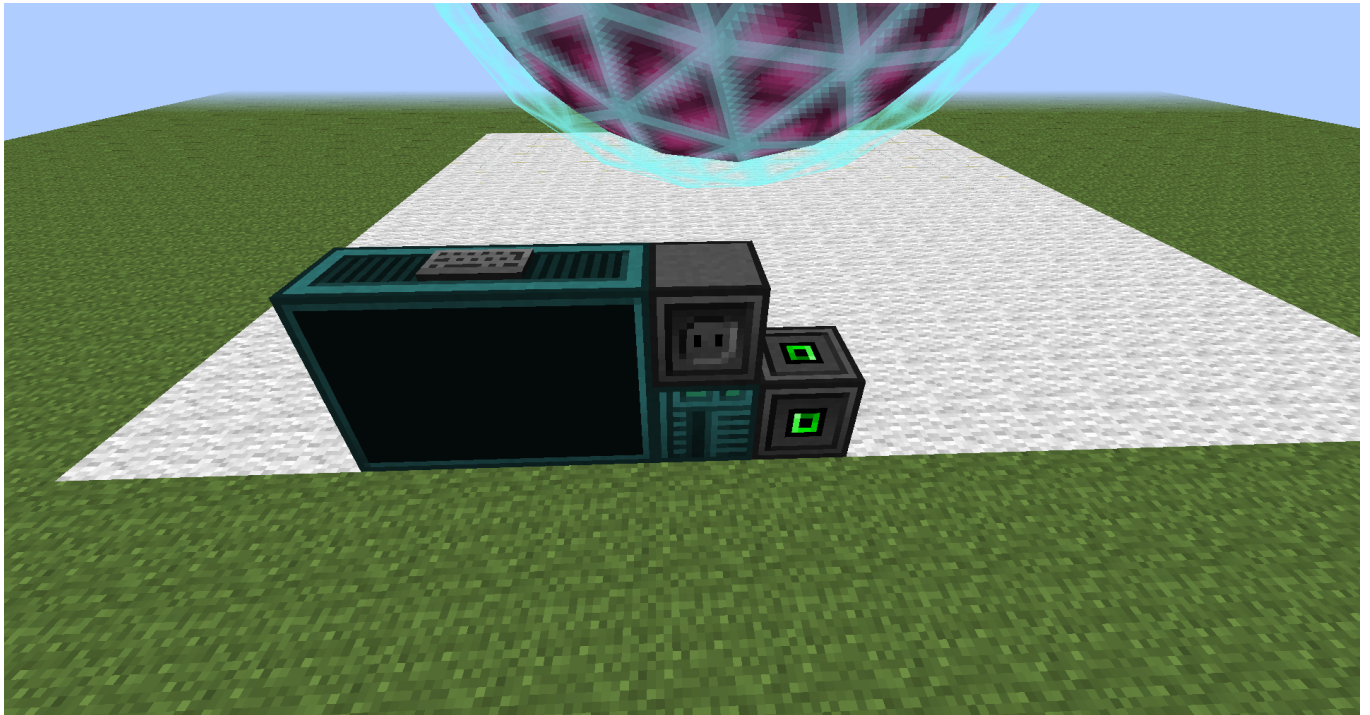
- Постройте Энергетическое ядро.

Скриншот ядра



- Установите мониторы (сеткой 3x2) и системный блок. Не забудьте прикрепить клавиатуру к монитору.

Скриншот установки ПК и мониторов



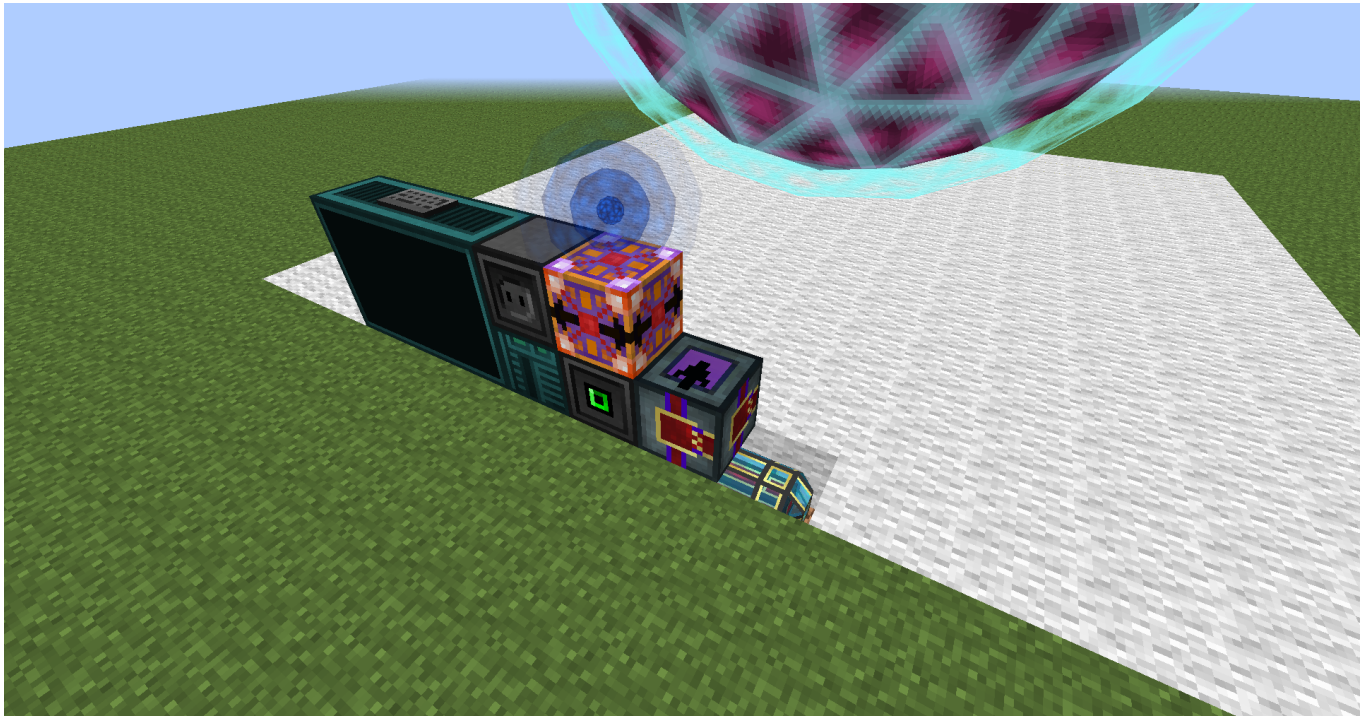
Нюансы сборки системного блока

1. **Lua BIOS:** Обычный чистый EEPROM нужно объединить в сетке крафта с "Руководством OpenComputers".
2. **OpenOS:** Обычную дискету также соедините с "Руководством OpenComputers", чтобы получить загрузочный диск с операционной системой.



3. Подключение компонентов

Скриншот подключения к ядру

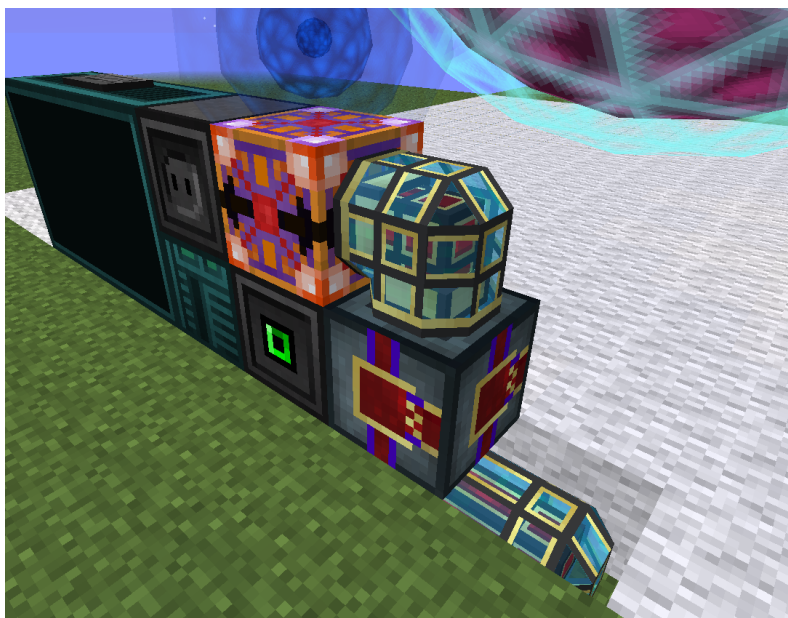


Настройка пилон и гейта

Важное уточнение: стекло на пилон следует ставить с той стороны пилон (верх, низ) которая смотрит в сторону центра ядра энергии. Это для тех, кто собирает схему не четко по инструкции.

1. **Энергетический пилон:** Установите его и поставьте блок стекла сверху для активации. Нажмите **ПКМ** по синему ядру внутри пилон, чтобы стрелки были направлены **наружу**. Это режим отдачи энергии.
2. **Флаксовый гейт:** Ставьте строго по направлению потока. Фиолетовая сторона (вход) должна смотреть на пилон, оранжевая (выход) - в сторону ваших потребителей.
3. **Соединение:** Соедините пилон и вход гейта Криостабилизированной трубой (у нее нет лимита передачи).
4. **Адаптер:** Поставьте адаптер вплотную к Флаксовому гейту, чтобы компьютер мог им управлять.

Важно: Следите, чтобы ядро не переполнялось. Если реакторам некуда будет отдавать энергию, произойдет критический перегрев и взрыв. Выход гейта должен быть подключен к мощному потребителю (например, к Генераторам материи через конвертер).



4. Настройка программной части

Включите компьютер кнопкой в системном блоке.



Сначала мы видим загрузку, после чего у нас появляется вот такое окно

```
OpenOS 1.6.1 (2048k RAM)
Screens can display Unicode - paste the special chars or use unicode.char.
```

```
/home #
```

Установка ОС

После загрузки с дискеты введите команду:

```
install
```

```
/home # install
Install OpenOS? [Y/n]
```

Нажимайте **Enter**, подтверждая установку. В конце система предложит перезагрузку - снова нажмите **Enter**.

```
Installation complete!
Reboot now? [Y/n]
```

Безопасность и автозапуск

Добавьте себя в список администраторов (команда чувствительна к регистру):

```
useradd ник
```

```
/home # useradd Des0ope
/home #
```

Это мы делали для того чтобы ПК могли пользоваться только вписанные в него пользователи.

Настройте автозапуск программы при включении ПК:

```
edit .shrc
```

и дописываем в самом конце такую строку

```
gate.lua
```

```
alias l="ls -lh"
alias ..="cd .."
alias df="df -h"

alias grep="grep --color"
gate.lua
```

И нажимаем `Ctrl + S` (сохранить) и `Ctrl + W` (закрыть).

Создание кода

Создайте файл программы:

```
edit gate.lua
```

Вставьте код. Код очень длинный, вставляйте его частями (по 250 строк) из-за ограничений буфера обмена.

Код (v2.0)

```
local component = require("component")
local event = require("event")
local computer = require("computer")
local unicode = require("unicode")
local term = require("term")
local keyboard = require("keyboard")
local gpu = component.gpu
```

```
-----
-- [КОНФИГУРАЦИЯ И ВЕРСИЯ]
-----
```

```

local VERSION = "v 2.0"

local CFG = {
    targetEnergy = 1000 * 10^9, -- Цель: 1 T RF
    aggressiveness = 0.03,      -- Плавность реакции стабилизации

    -- Настройки кнопок
    btnStep = 10^9,             -- Шаг (1B)
    btnStepCtrl = 10 * 10^9,    -- Шаг с Ctrl (10B)
    btnStepShift = 100 * 10^9   -- Шаг с Shift (100B)
}

local COLORS = {
    bg = 0x111111, tileBg = 0x222222, tileHeader = 0x333333,
    text = 0xEEEEEE, label = 0x999999,
    good = 0x55FF55, warn = 0xFFAA00, bad = 0xFF5555,
    energy = 0x00AAFF, sat = 0xFFAA00, fuel = 0xFFFF55,
    btn = 0x444444, btnActive = 0x006699,
    graphLine = 0x444444,
    black = 0x000000
}

-- Проверка компонентов
if not component.isAvailable("draconic_rf_storage") then
    print("ОШИБКА: Нет ядра (draconic_rf_storage)!")
    os.exit()
end

if not component.isAvailable("flux_gate") then
    print("ОШИБКА: Нет гейта (flux_gate)!")
    os.exit()
end

local rfStorage = component.draconic_rf_storage
local fluxGate = component.flux_gate

local W, H = gpu.maxResolution()
gpu.setResolution(W, H)

```

```

-- [ПЕРЕМЕННЫЕ И РАСЧЕТ МАКЕТА]
-----

local running = true
local buttons = {}

-- === НАСТРОЙКА СИММЕТРИИ ===
local M = 2 -- Отступ (Margin) со всех сторон
local panelH = 9 -- Высота верхних панелей

-- Расчет координат
local totalW = W - (M * 3)
local panelW = math.floor(totalW / 2)

-- Левая панель
local x1 = M
local y1 = M

-- Правая панель
local x2 = x1 + panelW + M
local y2 = M

-- График
local graphX = M
local graphY = y1 + panelH + 1
local graphW = W - (M * 2)
local graphH = H - graphY - M - 2

local energyHistory = {}
for i = 1, graphW do energyHistory[i] = 0 end

local maxCoreCapacity = rfStorage.getMaxEnergyStored()

-- Фильтр (Плавность)
local filVal = 0
local function expRunningAvg(newVal)
    if math.abs(newVal - filVal) > 10^15 then filVal = newVal end
    filVal = filVal + ((newVal - filVal) * 0.05)
    return filVal
end
end

```

```

-----
-- [ГРАФИКА - ФУНКЦИИ]
-----

local function rect(x, y, w, h, col)
    gpu.setBackground(col)
    gpu.fill(x, y, w, h, " ")
end

local function text(x, y, str, fg, bg)
    if bg then gpu.setBackground(bg) end
    gpu.setForeground(fg)
    gpu.set(x, y, str)
end

local function center(x, y, w, str, fg, bg)
    if bg then gpu.setBackground(bg) end
    gpu.fill(x, y, w, 1, " ")
    local px = x + math.floor((w - unicode.len(str))/2)
    if fg then gpu.setForeground(fg) end
    gpu.set(px, y, str)
end

-- ВНИМАНИЕ: Эта функция очищает всю строку под собой.
local function textRight(x, y, w, str, fg, bg)
    if bg then gpu.setBackground(bg) end
    gpu.fill(x, y, w, 1, " ")
    local len = unicode.len(str)
    local px = x + w - len
    if fg then gpu.setForeground(fg) end
    gpu.set(px, y, str)
end

local function fmt(num)
    if math.abs(num) >= 10^12 then return string.format("%.3f T", num/10^12)
    elseif math.abs(num) >= 10^9 then return string.format("%.3f B", num/10^9)
    elseif math.abs(num) >= 10^6 then return string.format("%.3f M", num/10^6)
    elseif math.abs(num) >= 10^3 then return string.format("%.3f K", num/10^3)
    else return string.format("%d", math.floor(num)) end
end

```

```
end
```

```
local function progressBar(x, y, w, pct, colBar, colBg)
```

```
    rect(x, y, w, 1, colBg)
```

```
    local barW = math.floor(w * (pct/100))
```

```
    if barW > w then barW = w end
```

```
    if barW > 0 then rect(x, y, barW, 1, colBar) end
```

```
end
```

```
local function btn(id, x, y, w, h, str, bg, fg)
```

```
    rect(x, y, w, h, bg)
```

```
    center(x, y + math.floor(h/2), w, str, fg or COLORS.text, bg)
```

```
    buttons[id] = {x=x, y=y, w=w, h=h}
```

```
end
```

```
-----  
-- [ИНТЕРФЕЙС]  
-----
```

```
local function drawStaticInterface()
```

```
    gpu.setBackground(COLORS.bg)
```

```
    term.clear()
```

```
    -- Левая панель
```

```
    rect(x1, y1, panelW, panelH, COLORS.tileBg)
```

```
    rect(x1, y1, panelW, 1, COLORS.tileHeader)
```

```
    center(x1, y1, panelW, "СОСТОЯНИЕ ЯДРА", COLORS.text, COLORS.tileHeader)
```

```
    text(x1+2, y1+2, "НАКОПЛЕНО:", COLORS.label, COLORS.tileBg)
```

```
    text(x1+2, y1+4, "ЦЕЛЬ СТАБ.:", COLORS.label, COLORS.tileBg)
```

```
    -- Правая панель
```

```
    rect(x2, y2, panelW, panelH, COLORS.tileBg)
```

```
    rect(x2, y2, panelW, 1, COLORS.tileHeader)
```

```
    center(x2, y2, panelW, "ЭНЕРГИЯ", COLORS.text, COLORS.tileHeader)
```

```
    text(x2+2, y2+2, "ТЕКУЩИЙ ПОТОК:", COLORS.label, COLORS.tileBg)
```

```
    text(x2+2, y2+4, "ВЫХОД ГЕЙТА:", COLORS.label, COLORS.tileBg)
```

```

-- График
rect(graphX, graphY, graphW, 1, COLORS.tileHeader)
center(graphX, graphY, graphW, " ГРАФИК ПОТОКА ", COLORS.label, COLORS.tileHeader)
rect(graphX, graphY+1, graphW, graphH, COLORS.bg)

-- Подвал
local footerY = H - 1
btn("exit", M, footerY, 10, 1, "ВЫХОД", COLORS.bad)

-- === ДОБАВЛЕНО: Версия слева внизу ===
local verX = M + 12 -- Отступ кнопки выхода + пробел
text(verX, footerY, VERSION, COLORS.label, COLORS.bg)

-- Кредиты
local crX = W - 32
local crY = H - 3
text(crX, crY, "Разработчик: Des0ope", COLORS.label, COLORS.bg)
text(crX, crY+1, "Проект: McSkill", COLORS.label, COLORS.bg)
text(crX, crY+2, "Сервер: HitechClassic 1.7.10", COLORS.label, COLORS.bg)
end

local function drawGraph(delta)
    table.insert(energyHistory, delta)
    while #energyHistory > graphW do table.remove(energyHistory, 1) end

    local gTopY = graphY + 1
    local gBottomY = graphY + graphH
    local midY = gTopY + math.floor(graphH/2)

    if graphH < 2 then return end

    local maxVal = 1
    for _, v in ipairs(energyHistory) do
        if math.abs(v) > maxVal then maxVal = math.abs(v) end
    end

    for i = 1, graphW do
        local val = energyHistory[i] or 0
        local x = graphX + i - 1

```

```

local h = 0
if math.abs(val) > 0 then
    h = math.ceil( (math.abs(val) / maxVal) * (graphH/2) )
    if h > (graphH/2) then h = math.floor(graphH/2) end
end

local barTop, barBottom
local color

if val > 0 then
    barBottom = midY + 1
    barTop = barBottom - h
    if barTop < gTopY then barTop = gTopY end
    color = COLORS.good
elseif val < 0 then
    barTop = midY + 1
    barBottom = barTop + h
    if barBottom > gBottomY then barBottom = gBottomY end
    color = COLORS.bad
else
    barTop = midY
    barBottom = midY + 1
    color = COLORS.bg
end

if barTop > gTopY then
    gpu.setBackground(COLORS.bg)
    gpu.fill(x, gTopY, 1, barTop - gTopY, " ")
end

local barH = barBottom - barTop
if val ~= 0 and barH > 0 then
    gpu.setBackground(color)
    gpu.fill(x, barTop, 1, barH, " ")
elseif val == 0 then
    gpu.setBackground(COLORS.bg)
    gpu.setForeground(COLORS.graphLine)
    gpu.set(x, midY, "-")
end

```

```

        end

        if barBottom < gBottomY then
            gpu.setBackground(COLORS.bg)
            local clearH = gBottomY - barBottom + 1
            gpu.fill(x, barBottom, 1, clearH, " ")
        end
    end
end

local function updateLiveDisplay(stored, target, delta, gateFlow)
    local valWidth = 16
    local valX = x1 + panelW - valWidth - 2

    -- === ЛЕВАЯ ПАНЕЛЬ ===
    center(valX, y1+2, valWidth, fmt(stored).. " RF", COLORS.energy, COLORS.tileBg)
    center(valX, y1+4, valWidth, fmt(target).. " RF", COLORS.warn, COLORS.tileBg)

    local btnW = 3
    local btnPlusX = valX - btnW - 1
    btn("inc", btnPlusX, y1+4, btnW, 1, "+", COLORS.btn)
    local btnMinusX = btnPlusX - btnW - 1
    btn("dec", btnMinusX, y1+4, btnW, 1, "-", COLORS.btn)

    local maxPct = (stored / maxCoreCapacity) * 100
    if maxPct > 100 then maxPct = 100 end

    local barX = x1 + 2
    local barW = panelW - 4

    center(barX, y1+6, barW, string.format("%.2f%% (от макс. объема)", maxPct), COLORS.text,
COLORS.tileBg)
    text(barX, y1+6, "0%", COLORS.label)

    local str100 = "100%"
    local x100 = barX + barW - unicode.len(str100)
    text(x100, y1+6, str100, COLORS.label)

    progressBar(barX, y1+7, barW, maxPct, COLORS.energy, 0x111111)

```

```

-- === ПРАВАЯ ПАНЕЛЬ ===
local valX2 = x2 + panelW - valWidth - 2

local dColor = COLORS.text
local sign = ""
if delta > 0 then dColor = COLORS.good; sign = "+"
elseif delta < 0 then dColor = COLORS.bad; sign = "" end

center(valX2, y2+2, valWidth, sign..fmt(delta).. " RF/t", dColor, COLORS.tileBg)
center(valX2, y2+4, valWidth, fmt(gateFlow).. " RF/t", COLORS.text, COLORS.tileBg)

-- === БОЛЬШОЙ СТАТУС БАР (ВНИЗУ) ===
local statusStr = ""
local statusBg = COLORS.btn
local statusFg = COLORS.black

-- НОВАЯ ЛОГИКА СТАТУСОВ
if math.abs(stored - target) < 10^9 then
    -- Энергия в норме (равна цели)
    statusStr = "СТАБИЛЬНО"
    statusBg = COLORS.good
else
    if stored < target then
        -- Нехватка энергии
        if delta > 0 then
            -- Но она растет = Хорошо
            statusStr = "ВОСПОЛНЕНИЕ"
            statusBg = COLORS.energy -- Синий
        else
            -- Она падает = Плохо
            statusStr = "ВНИМАНИЕ: РАЗРЯД"
            statusBg = COLORS.bad -- Красный
            statusFg = 0xFFFFFF
        end
    else
        -- Избыток энергии
        if delta < 0 then
            -- Она падает (возвращается к цели) = Хорошо

```

```

        statusStr = "СБОРОС ИЗЛИШКОВ"
        statusBg = COLORS.warn -- Оранжевый
    else
        -- Она растёт дальше = Плохо
        statusStr = "КРИТИЧЕСКОЕ ПЕРЕПОЛНЕНИЕ"
        statusBg = COLORS.bad -- Красный
        statusFg = 0xFFFFFF
    end
end
end

local stX = x2 + 2
local stW = panelW - 4
local stY = y2 + 6
local stH = 3

gpu.setBackground(statusBg)
gpu.fill(stX, stY, stW, stH, " ")
center(stX, stY + 1, stW, statusStr, statusFg, statusBg)

drawGraph(delta)
end

local function handleTouch(x, y)
    for id, b in pairs(buttons) do
        if x >= b.x and x < b.x + b.w and y >= b.y and y < b.y + b.h then
            if id == "exit" then
                running = false
            elseif id == "inc" or id == "dec" then
                local step = CFG.btnStep
                if keyboard.isControlDown() then step = CFG.btnStepCtrl
                elseif keyboard.isShiftDown() then step = CFG.btnStepShift end

                if id == "dec" then CFG.targetEnergy = CFG.targetEnergy - step end
                if id == "inc" then CFG.targetEnergy = CFG.targetEnergy + step end

                if CFG.targetEnergy < 0 then CFG.targetEnergy = 0 end
                computer.beep(1000, 0.05)
            end
        end
    end
end

```

```

        rect(b.x, b.y, b.w, b.h, COLORS.good)
        os.sleep(0.05)
        rect(b.x, b.y, b.w, b.h, COLORS.btn)
    end
end
end
end

-----
-- [ГЛАВНЫЙ ЦИКЛ]
-----

drawStaticInterface()

-- Мы искусственно отнимаем 1 секунду от времени запуска,
-- чтобы первый кадр отрисовался моментально.
local prevTime = computer.uptime() - 1.0
local prevEnergy = rfStorage.getEnergyStored()

while running do
    -- Опрашиваем клики по экрану часто (каждые 0.1 сек), чтобы кнопки работали быстро
    local sig = {event.pull(0.1, "touch")}
    if sig[1] == "touch" then
        handleTouch(sig[3], sig[4])
    end

    local nowTime = computer.uptime()
    local dt = nowTime - prevTime

    -- Математика, график и обновление гейта срабатывают ровно 1 раз в секунду
    if dt >= 1.0 then
        local nowEnergy = rfStorage.getEnergyStored()
        local rawDiff = (nowEnergy - prevEnergy) / (dt * 20)

        local smoothedDiff = expRunningAvg(rawDiff)

        local currentGate = fluxGate.getFlow()
        local errorVal = nowEnergy - CFG.targetEnergy
        local correction = errorVal * CFG.aggressiveness
        local newGate = currentGate + smoothedDiff + correction
    end
end

```

```
if newGate < 0 then newGate = 0 end

-- Защита от спама в сеть
if math.abs(newGate - currentGate) > 500 then
    fluxGate.setSignalLowFlow(math.floor(newGate))
end

updateLiveDisplay(nowEnergy, CFG.targetEnergy, smoothedDiff, newGate)

prevTime = nowTime
prevEnergy = nowEnergy
end
end

gpu.setResolution(W, H)
gpu.setBackground(0x000000)
gpu.setForeground(0xFFFFFFFF)
term.clear()
```

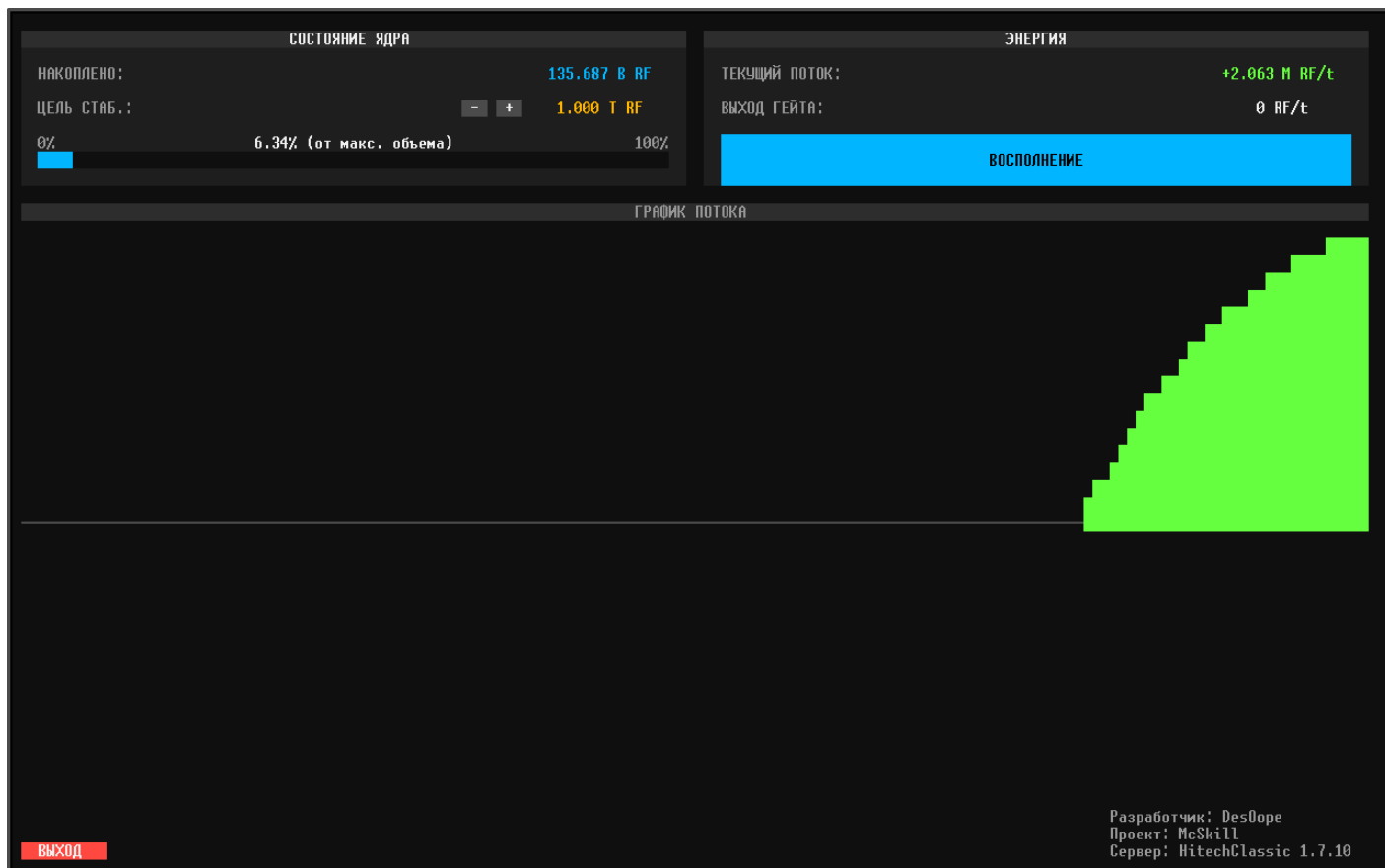
И нажимаем **Ctrl + S** (сохранить) и **Ctrl + W** (закреть).

Перезагрузите компьютер командой:

```
reboot
```

5. Управление программой

После перезагрузки откроется интерфейс стабилизации. Вы увидите текущее состояние ядра и настройки лимитов.



Регулировка целевого уровня энергии:

- **ЛКМ** по кнопкам "+" или "-": изменение на 1 млрд (1B).
- **Ctrl + ЛКМ**: изменение на 10 млрд (10B).
- **Shift + ЛКМ**: изменение на 100 млрд (100B).

Логика работы:

- Если энергии в ядре меньше установленной цели - Флаксовый гейт закрыт (энергия не уходит).
- Как только цель достигнута - гейт открывается.
- Если уровень энергии продолжает расти выше цели - программа автоматически повышает пропускную способность гейта, пока ситуация не стабилизируется.

Revision #7

Created 2026-03-17 19:53:46 UTC by DesOope

Updated 2026-06-12 11:52:36 UTC by DesOope