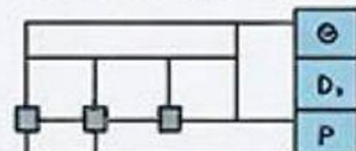


20 ппабсуапоокпные



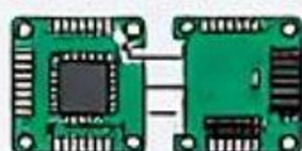
Нарвооторое



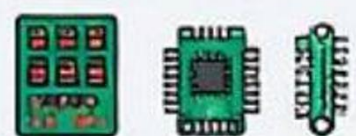
Цда ахо саснаепот

Сана	Сана
2020	2020
2020	2020
2020	2020
2020	2020
2020	2020

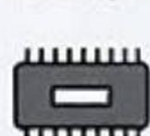
Ty tokpo зарка, CC C6



ESP8266 EPP32



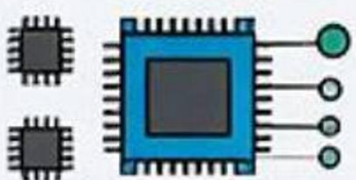
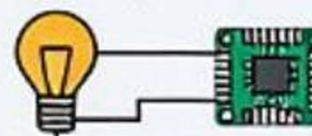
Wivieгггзeronза



Бусесрос зерасове



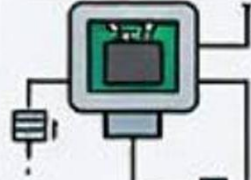
Ooderming зарка, CC C6



Eoif MOI pome



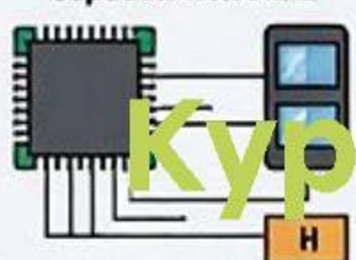
Phonnet



Hadulieteyoоkara

Сана	Сана	Сана
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020

Sepdocol'takooina



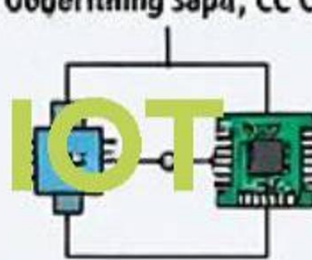
Субоопазторове



Seestowetetise



Ooderming зарка, CC C6



Sufoavacrophotee

Сана	Сана	Сана
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020

Нарвооторое



Цда ахо саснаепот



Нодовотозвон

Сана	Сана	Сана
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020

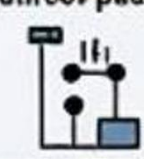
OY Fpaeronmiononae

Сана	Сана	Сана
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020

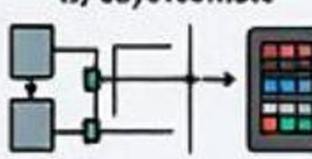
Пер Buetoile



Нахрос раеие



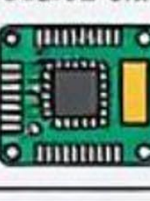
Ty cayeтеемые



Егерратеке

Сана	Сана	Сана
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020

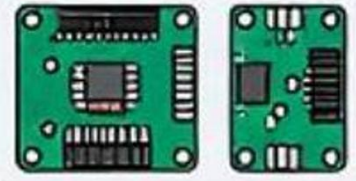
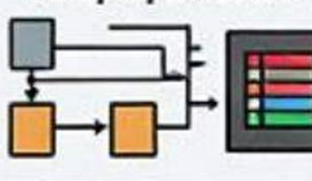
Сваблор claro



Bandapeere



Расор rapandeeato



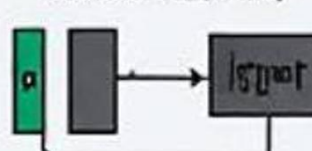
Pploeonrhure



Сое kAnuna тeуepa



Scntora appoeto,



Сана	Сана	Сана
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020
2020	2020	2020

Rodepa ppcerce



Matatak



IHeerrman



Курс создания IoT устройств с помощью esphome

Учитель Начальной

Учитель Начальной

**Курс создания IoT устройств
с помощью esphome**

«Автор»

2026

Начальной У. Ш.

Курс создания IoT устройств с помощью esphome /
У. Ш. Начальной — «Автор», 2026

Практический курс по созданию IoT-устройств на базе ESPHome — от первого YAML до готовых проектов умного дома. В 20 главах разобраны выбор железа (ESP8266, ESP32 и современные варианты C3/C6), установка среды, синтаксис конфигураций, работа с Wi-Fi и Ethernet, сенсоры, реле, светодиоды, дисплеи, шины I²C/SPI/UART, автоматизации на устройстве, Bluetooth, deep sleep для батарейных узлов, безопасность и отладка. Отдельные главы посвящены интеграции с Home Assistant, OTA-обновлениям и двум полным проектам: умному выключателю и метеостанции. Книга содержит десятки готовых примеров кода, рекомендации по питанию, масштабированию парка устройств и лучшим практикам. Подходит новичкам без опыта C++ и тем, кто уже собирает DIY-автоматизацию и хочет систематизировать знания. Акцент на локальном контроле, надёжности и реальном применении в домашней автоматизации.

Содержание

ГЛАВА 1. ВВЕДЕНИЕ В IoT И ESPHome	5
ГЛАВА 2. АППАРАТНАЯ ПЛАТФОРМА: ESP8266 И ESP32	7
ГЛАВА 3. УСТАНОВКА И НАСТРОЙКА СРЕДЫ РАЗРАБОТКИ	10
Конец ознакомительного фрагмента.	12

Учитель Начальной

Курс создания IoT устройств с помощью esphome

ГЛАВА 1. ВВЕДЕНИЕ В IoT И ESPHome

1.1. Что такое Internet of Things (IoT)

Internet of Things (Интернет вещей) — это сеть физических объектов, оснащённых датчиками, исполнительными механизмами, микроконтроллерами и средствами связи. Эти устройства собирают данные об окружающей среде, обмениваются ими и принимают решения с минимальным участием человека.

Типичные сферы применения:

- Умный дом: освещение, климат, безопасность, энергомониторинг
- Промышленность: мониторинг оборудования, предиктивное обслуживание
- Сельское хозяйство: влажность почвы, теплицы, полив
- Медицина: носимые датчики, удалённый мониторинг пациентов
- Городская инфраструктура: освещение улиц, парковки, качество воздуха

Ключевые требования к IoT-устройству:

- Надёжность (стабильная работа месяцами без перезагрузки)
- Энергоэффективность (особенно для батарейных узлов)
- Безопасность (защита от несанкционированного доступа)
- Локальный контроль (работа без обязательного облака)
- Простота интеграции в существующую систему автоматизации

1.2. Почему именно ESPHome

ESPHome — open-source система, которая превращает микроконтроллеры ESP8266, ESP32 (и ряд других платформ) в полноценные умные устройства. Вместо написания прошивки на C++ вы описываете поведение устройства в YAML-файле. ESPHome компилирует конфигурацию в оптимизированную прошивку и загружает её на чип.

Преимущества ESPHome:

- Нет необходимости писать C++ для 95% типичных задач
- Нативная интеграция с Home Assistant — сущности появляются автоматически
- Обновление прошивки по воздуху (OTA) без физического доступа
- Сотни готовых компонентов: сенсоры, дисплеи, реле, климат, медиа
- Локальный контроль: устройства работают в вашей сети без облака
- Активное сообщество, регулярные релизы, подробная документация
- Поддержка Arduino и ESP-IDF framework
- Возможность писать lambda (вставки C++) при необходимости

Альтернативы и сравнение:

- Tasmota — тоже YAML/веб-конфиг, силён в готовых прошивках для реле/розеток
- ESPEasy — гибкий, но сложнее в освоении
- Чистый Arduino / ESP-IDF — полный контроль, но больше кода и поддержки
- PlatformIO + свои скетчи — гибко, но интеграция с HA делается вручную

ESPHome особенно удобен, если у вас уже есть (или планируется) Home Assistant.

1.3. Как устроен процесс разработки

Типичный цикл:

1. Создаёте YAML-конфигурацию (через wizard, Device Builder или вручную)
2. ESPHome проверяет синтаксис и собирает прошивку (PlatformIO под капотом)
3. Загружаете прошивку на устройство:
 - первый раз по USB (UART)
 - далее — по OTA (Wi-Fi)
4. Устройство подключается к Wi-Fi, поднимает Native API
5. Home Assistant обнаруживает устройство и создаёт сущности
6. Настраиваете автоматизации в HA или прямо в ESPHome

Способы работы:

- Home Assistant Add-on (самый простой путь)
- ESPHome Device Builder (веб-UI, можно в Docker)
- Командная строка: `esphome run / compile / upload / logs`
- CI/CD (проверка конфигураций в git)

1.4. Что вы узнаете в этом курсе

Курс построен от простого к сложному:

- Главы 1–5 — фундамент (железо, установка, YAML, сеть)
- Главы 6–11 — компоненты (сенсоры, актюаторы, шины, дисплеи)
- Главы 12–16 — продвинутые возможности и эксплуатация
- Главы 17–18 — два полноценных проекта
- Главы 19–20 — кастомные компоненты, масштабирование, ресурсы

ГЛАВА 2. АППАРАТНАЯ ПЛАТФОРМА: ESP8266 И ESP32

2.1. Семейство ESP

Микроконтроллеры Espressif Systems стали стандартом DIY IoT благодаря:

- Низкой цене
- Встроенному Wi-Fi (и Bluetooth у большинства ESP32)
- Достаточному количеству GPIO и периферии
- Огромной экосистеме и поддержке сообществом

ESPHome официально поддерживает ESP8266, ESP32 и множество вариантов.

2.2. ESP8266

Характеристики (типичные):

- Одноядерный Tensilica L106, 80/160 МГц
- ~80 КБ RAM, доступной приложению
- Flash: обычно 1–4 МБ
- Wi-Fi 802.11 b/g/n
- Нет Bluetooth
- Ограниченное число безопасных GPIO

Популярные платы:

- NodeMCU v2 / v3
- Wemos D1 Mini
- ESP-01 / ESP-01S (очень мало пинов)
- Sonoff Basic (готовое реле с ESP8266)

Рекомендация ESPHome: для НОВЫХ проектов лучше брать ESP32-C3 (сопоставимая цена, больше RAM и flash, BLE). ESP8266 продолжают поддерживаться, существующие устройства работают стабильно.

Особые пины ESP8266 (нельзя использовать свободно):

GPIO0, GPIO2, GPIO15 — влияют на режим загрузки
GPIO1, GPIO3 — UART0 (TX/RX)
GPIO6–11 — подключены к flash

2.3. ESP32 и варианты

Оригинальный ESP32:

- Двухъядерный Xtensa LX6, до 240 МГц
- ~320 КБ RAM, часто 4 МБ flash (+PSRAM на некоторых платах)
- Wi-Fi + Bluetooth Classic/BLE
- Богатая периферия (ADC, DAC, touch, hall, CAN и т.д.)
- Самый зрелый и рекомендуемый вариант для большинства проектов

Варианты:

ESP32-S2

Wi-Fi, без BLE, USB OTG, хорош для устройств с USB

ESP32-S3

Двухъядерный, Wi-Fi + BLE 5, больше AI/ускорителей, часто с PSRAM

ESP32-C3

Одноядерный RISC-V, Wi-Fi + BLE 5, позиционируется как замена ESP8266

Меньше GPIO, ниже потребление, отличный выбор для простых узлов

ESP32-C6

RISC-V, Wi-Fi 6 + BLE + Thread/Zigbee (OpenThread в ESPHome)

Хороший выбор для новых проектов с Mesh/Thread

ESP32-H2

BLE + Thread/Zigbee, без Wi-Fi — для чистых Thread-устройств

ESP32-P4

Высокопроизводительный RISC-V, для сложных задач (видео, UI)

Рекомендации ESPHome (актуальные):

- Лучший общий выбор: ESP32 (классический)
- Бюджетный/простой узел: ESP32-C3
- Новый проект с перспективой Thread: ESP32-C6
- Не рекомендуют для новых проектов: ESP8266

2.4. Выбор конкретной платы

На что смотреть:

- Чип и объём flash/PSRAM
- Наличие USB-UART (CP2102, CH340, встроенный USB у C3/S2/S3)
- Распиновка и удобство макетирования (DevKit vs модуль)
- Ethernet (Olimex ESP32-POE, WT32-ETH01)
- Дисплей на борту (LilyGO T-Display, M5Stack)
- Питание: 5 В USB, 3.3 В, Li-ion

Примеры board ID для ESPHome (PlatformIO):

esp32dev — универсальный ESP32

nodemcu2 — NodeMCU ESP8266

esp32-c3-devkitm-1

esp32-s3-devkitc-1

mhetesp32devkit

lolin_s2_mini

Если точной платы нет в списке — почти всегда работает generic:

esp32dev / esp01_1m / esp32-c3-devkitm-1

2.5. Питание и надёжность

- Стабильные 3.3 В (не ниже 3.0 В под нагрузкой Wi-Fi)
- Condensators 100 nF + 10–100 μ F рядом с модулем
- Для реле и мощных нагрузок — отдельное питание, общая земля
- Избегайте длинных тонких проводов питания
- Brownout detector на ESP32 может перезагружать устройство при просадках

ГЛАВА 3. УСТАНОВКА И НАСТРОЙКА СРЕДЫ РАЗРАБОТКИ

3.1. Способы установки

1) Home Assistant Add-on

Самый простой путь. В НА: Settings → Add-ons → Add-on Store → ESPHome.
После установки открывается веб-интерфейс Dashboard.

2) Docker

```
docker run -d --name esphome -v /path/to/config:/config \
-v /etc/localtime:/etc/localtime:ro --network host \
ghcr.io/esphome/esphome
```

3) pip (командная строка)

```
python3 -m venv esphome-venv
source esphome-venv/bin/activate # Windows: esphome-venv\Scripts\activate
pip install esphome
esphome version
```

4) ESPHome Device Builder (standalone веб-UI)

Можно запускать локально или в контейнере, удобен без полного НА.

3.2. Первая конфигурация через wizard

```
esphome wizard my_device.yaml
```

Мастер спросит:

- Имя устройства (hostname, латиница, без пробелов)
- Платформу (ESP32 / ESP8266)
- Плату (board)
- SSID и пароль Wi-Fi

Будет создан базовый YAML. Далее редактируете его и запускаете:

```
esphome run my_device.yaml
```

Команда скомпилирует, предложит способ загрузки (USB / OTA) и сразу покажет логи.

3.3. Структура каталога проектов

Рекомендуемая структура:

```
esphome-configs/
├── secrets.yaml
```

```
|— packages/
|   |— base_wifi.yaml
|   |— base_api.yaml
|   |— sensors_common.yaml
|— livingroom.yaml
|— bedroom_switch.yaml
|— weather_station.yaml
```

secrets.yaml (никогда не коммитьте в публичный репозиторий):

```
wifi_ssid: "MyHomeWiFi"
wifi_password: "supersecret"
api_encryption_key: "сгенерированный_ключ_base64"
ota_password: "ota_secret"
```

Генерация ключа шифрования API:

```
openssl rand -base64 32
```

3.4. Полезные команды CLI

```
esphome config device.yaml # только проверка конфигурации
esphome compile device.yaml # сборка без загрузки
esphome upload device.yaml # только загрузка
esphome logs device.yaml # логи (USB или API)
esphome run device.yaml # compile + upload + logs
esphome clean device.yaml # очистка сборочного кэша
esphome wizard new.yaml # мастер новой конфигурации
```

3.5. Первая прошивка по USB

1. Подключите устройство USB-кабелем (data-кабель, не charge-only)
2. На многих ESP32: зажмите BOOT, нажмите RESET, отпустите RESET, затем BOOT
3. Выберите порт в меню esphome run
4. Дождитесь окончания загрузки
5. После успеха устройство само подключится к Wi-Fi

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.