# **ТЗ · Этап 1 — Базовый конвертер (Popup, real‑time)**

## **0) Резюме**

Сделать Chrome‑расширение (Manifest V3) с **попап‑интерфейсом**, которое **офлайн** конвертирует:

* **Text → Morse** и **Morse → Text** в **реальном времени** (без кнопки Convert).
* Поддерживает три набора алфавитов для мэппинга:  
   **ITU (default)**, **ENG**, **RUS**.
* Имеет **кнопку ⓘ** с модальным окном внутри попапа: правила и таблицы кодов.
* Без бэкенда, без телеметрии, без тем. Язык UI — **EN**.

На этом этапе **нет аудио** и **нет контекстного меню** — это в следующих этапах.

## **1) Пользовательские сценарии (Acceptance)**

1. Ввожу текст в левое поле → справа мгновенно появляется корректный Морзе (разделение: **1 пробел между символами**, **/ между словами**).
2. Ввожу Морзе справа (символы . и -, разделители «пробел» и «/») → слева мгновенно появляется текст.
3. Переключаю **Alphabet** (ITU/ENG/RUS) → мэппинг меняется для обеих сторон.
4. Нажимаю ⓘ → вижу правила и таблицу соответствий выбранного алфавита.
5. Кнопки **Copy / Clear / Paste** на Text, **Copy / Clear** на Morse работают корректно.
6. Ввод до **50 000 символов** не «замораживает» попап (UI остаётся отзывчивым).
7. Всё работает **офлайн**, данные никуда не отправляются.

## **2) UI/UX (popup)**

### **2.1. Макет (без кнопки Convert)**

* **Header (узкая полоса):**
  + Слева: название Morse Code Translator (или иконка).
  + Справа: круглое ⓘ — открывает **модалку** поверх попапа.
* **Main (две колонки, равной ширины, адаптивные):**
  + **Left / Text**
    - <textarea id="textInput">
    - Под ним:  
       Paste · Copy · Clear
  + **Right / Morse**
    - <textarea id="morseInput">
    - Под ним:  
       Copy · Clear
* **Footer (одна линия):**
  + Label Alphabet + <select id="alphabetSelect"> с опциями: ITU (default), ENG, RUS.

### **2.2. Поведение**

* **Автоконвертация** на input обоих полей с **микро-дебаунсом 30–50 мс** (для плавности).
* Поля **двунаправленные**: вводишь в одно — пересчёт в другое.
* **Источник истины** — то поле, куда пользователь **последним** вводил символы.
* Paste использует navigator.clipboard.readText() (по клику).
* Copy использует navigator.clipboard.writeText() (по клику).
* Clear очищает только своё поле и **инициирует** пересчёт «в паре».
* **Модалка ⓘ** внутри попапа (без отдельной options‑page). Содержит:  
  + краткие правила разметки (см. §4.3),
  + таблицу кодов **для активного алфавита**,
  + переключатель алфавита **дублировать не нужно** (используем нижний селект).

### **2.3. Доступность**

* Все контролы имеют aria-label.
* Таб‑навигация: Header → Text → Text‑actions → Morse → Morse‑actions → Footer‑select → Info‑modal (если открыт).
* Контраст минимум WCAG AA для текста и границ.

## **3) Логика конвертации**

### **3.1. Нормализация входа**

* **Text → Morse**
  + Тримим, приводим буквы к **верхнему регистру**.
  + **ENG/ITU**: оставляем латиницу A–Z, цифры 0–9, пунктацию из поддерживаемого списка; остальные — помечаем «unknown».
  + **RUS**: оставляем кириллицу; по умолчанию **Ё → Е** (если нет явного мэппинга), пробелы сохраняем как разделители слов.
* **Morse → Text**
  + Нормализуем «точки/тире»: заменяем любые · → . и – → -.
  + Сжимаем повторные пробелы.
  + **Разделители:**
    - символ между **буквами**: **один пробел**,
    - символ между **словами**: / (также принимаем **2+ пробелов** как словораздел и конвертируем в / при нормализации).
  + Игнорим лишние недопустимые символы (кроме . - / и пробела).

### **3.2. Разделители в выходе**

* **Строго**: буквы — (пробел), слова — / (пробел, слэш, пробел).

### **3.3. Поведение на «unknown»**

* **Text → Morse**: для неизвестного символа вставляем маркер □ (не морзим).
* **Morse → Text**: для неизвестной последовательности — символ □.

### **3.4. Производительность**

* Конвертация — **O(n)**, итерация по строке.
* Для 50k символов — использовать **chunked‑render** (через requestIdleCallback/setTimeout(0) после 10–20k символов) или «ленивое обновление» (рендер текста одним куском, без частых DOM‑операций).

## **4) Алфавиты (мэппинги)**

### **4.1. ITU (International Morse) — базовый**

**Буквы**

A .- B -... C -.-. D -.. E . F ..-. G --. H ....

I .. J .--- K -.- L .-.. M -- N -. O --- P .--.

Q --.- R .-. S ... T - U ..- V ...- W .-- X -..-

Y -.-- Z --..

**Цифры**

0 ----- 1 .---- 2 ..--- 3 ...-- 4 ....- 5 .....

6 -.... 7 --... 8 ---.. 9 ----.

**Пунктуация**

. .-.-.- , --..-- ? ..--.. ' .----. ! -.-.-- / -..-.

( -.--. ) -.--.- & .-... : ---... ; -.-.-. = -...-

+ .-.-. - -....- \_ ..--.- " .-..-. $ ...-..- @ .--.-.

### **4.2. ENG**

* На **этом этапе** трактуем как **латиница A–Z / 0–9 / пунктуация**, **тот же мэппинг, что ITU**.
* **Цель ENG** сейчас — UX‑ясность для пользователя «только English набор».
* (**Важно**: «American Morse» *не* поддерживается в Этапе 1; добавим позже как отдельный режим, т.к. там иная длительность элементов и «внутрибуквенные паузы».)

### **4.3. RUS (кириллица)**

* Используем стандартные соответствия **русского телеграфного кода (расширение ITU)**.
* Реализовать как **отдельный словарь** (см. структуру проекта).
* Если мэппинг конкретных букв вызывает сомнения (напр. Ё, Ъ) — покрыть **юнит‑тестами** и явно задать поведение: Ё→Е по умолчанию, Ъ — редкий символ, можно не поддерживать на старте (вернуть □).
* В модалке ⓘ показать таблицу кириллицы.

Примечание: точную таблицу для RUS положить в src/morse/mapping\_ru.ts и использовать в тестах (источник — общепринятые таблицы русского Морзе; если в проекте появится legal‑требование на ГОСТ — зафиксируем ссылку и версию в комментариях).

### **4.4. Правила, показываемые в ⓘ (коротко)**

* Буквы разделяются **одним пробелом**, слова — **/**.
* Допускается ввод Morse с **многими пробелами** между словами — мы считаем 2+ пробела как «разделитель слов».
* При вводе «неизвестного» символа (в выбранном алфавите) — в другой стороне появится **□**.

## **5) Архитектура и структура файлов**

### **5.1. Стек**

* **Manifest V3**, без background в Этапе 1.
* **Vanilla TS/JS** + модульная структура ES. Сборка простой esbuild/Vite (любой, лишь бы выдавал dist/).
* Стили — простой CSS (без тем), адаптивная сетка.

### **5.2. Дерево проекта (минимум)**

/public

icon16.png

icon48.png

icon128.png

/src

popup.html

popup.ts

popup.css

info-modal.css

/morse

mapping\_itu.ts

mapping\_eng.ts // alias ITU на Этапе 1

mapping\_ru.ts

convert.ts // API: textToMorse(), morseToText()

normalize.ts // нормализация ввода и разделителей

/i18n

en.json // тексты UI и модалки (EN only)

manifest.json

package.json

tsconfig.json (если TS)

### **5.3. manifest.json (скелет)**

{

"manifest\_version": 3,

"name": "Morse Code Translator",

"version": "1.0.0",

"description": "Offline, real-time text ↔ morse converter (ITU/ENG/RUS).",

"action": {

"default\_title": "Morse Code Translator",

"default\_popup": "src/popup.html"

},

"icons": {

"16": "public/icon16.png",

"48": "public/icon48.png",

"128": "public/icon128.png"

},

"permissions": [

"storage" // сохранить выбранный Alphabet

],

"host\_permissions": []

}

На Этапе 1 **не** добавляем background, contextMenus, offscreen, clipboard\*.

## **6) Контракты функций (convert API)**

src/morse/convert.ts

export type Alphabet = 'ITU' | 'ENG' | 'RUS';

export interface ConvertOptions {

alphabet: Alphabet;

unknownChar?: string; // default: '□'

}

export function textToMorse(input: string, opts: ConvertOptions): string;

// Правила:

// - буквы через ' '

// - слова через ' / '

// - неизвестные символы -> unknownChar

export function morseToText(input: string, opts: ConvertOptions): string;

// Правила:

// - нормализация . и -

// - 2+ пробелов трактуем как ' / '

// - неизвестные последовательности -> unknownChar

src/morse/normalize.ts

export function normalizeText(input: string, alphabet: Alphabet): string;

// Uppercase, фильтрация под набор, RUS: ё->е (если нет ё)

export function normalizeMorse(input: string): string;

// Заменить '·'->'.', '–'->'-', сжать пробелы,

// 2+ пробелов превратить в ' / ' (с пробелами по краям)

src/morse/mapping\_\*.ts

export const MAP\_FWD: Record<string, string>; // 'A' -> '.-'

export const MAP\_REV: Record<string, string>; // '.-' -> 'A'

## **7) Логика в попапе**

src/popup.ts

* Получить ссылки на элементы: textInput, morseInput, alphabetSelect, кнопки.
* Прочитать alphabet из chrome.storage.local (default: ITU), проставить в селект.
* Повесить обработчики:  
  + textInput.oninput → debounce → textToMorse → записать в morseInput (без смещения курсора в text).
  + morseInput.oninput → debounce → morseToText → записать в textInput.
  + alphabetSelect.onchange → сохранить в storage → **пересчитать оба поля** из «активного источника».  
    - Хранить флаг lastEdited: 'text' | 'morse'.
  + Copy/Clear/Paste — поведение локальное для своей колонки.
  + ⓘ — открыть модалку, отрисовать таблицу шорт‑листом (A–Z/А–Я/пунктуация) для выбранного алфавита.
* Защититься от «зацикливания» обновлений (флаг isSyncing).

## **8) Тестирование**

### **8.1. Unit (Vitest/Jest допустимо)**

Покрыть convert.ts для **каждого** алфавита:

* ITU:  
  + "SOS" → "... --- ..."
  + "HELLO WORLD" → ".... . .-.. .-.. --- / .-- --- .-. .-.. -.." и обратно.
  + Пунктуация: "TEST, OK." ↔ "- . ... - --..-- / --- -.- .-.-.-"
  + Цифры: "2025" ↔ "..--- ----- ..--- .....".
* ENG: (идентичен ITU на Этапе 1).
* RUS:  
  + "ПРИВЕТ" и "СОС" (проверить прямой/обратный ход).
  + Ё → Е (если Ё не задан) / Ъ → □ (если не задан).
* Unknown:  
  + "A🙂B" → ".- □ -...".
  + "...-.-" (несуществующий) → "□".

### **8.2. Smoke E2E (Playwright опционально)**

* Открыть попап, ввести «HELLO» → увидеть Морзе.
* Переключить алфавит → пересчёт не ломается.
* Copy/Clear/Paste работают, модалка отображает таблицу.

## **9) Нефункциональные и качество**

* **Оффлайн** by design (никаких сетевых запросов).
* **Производительность**: 50k символов без freeze (дебаунс, минимизация перерисовок).
* **Код‑стайл**: ESLint + Prettier (базовые правила).
* **Безопасность**: не запрашивать лишние permissions.

## **10) Критерии приёмки (Checklist)**

* Popup открывается, UI совпадает с макетом.
* Автоконвертация двунаправленно, без кнопки Convert.
* Выбор Alphabet (ITU/ENG/RUS) влияет на мэппинг.
* Модалка ⓘ показывает правила и таблицу для активного алфавита.
* Корректные разделители: символы — , слова — /.
* Unknown обрабатываются как □.
* 50k символов — UI остаётся отзывчивым.
* Всё работает без интернета.
* Unit‑тесты зелёные.

## **11) Заметки наперёд (для Этапов 2–4, не делать сейчас)**

* **Этап 2**: WebAudio Play и Save .wav (+ offscreen в манифест, если потребуется).
* **Этап 3**: «Stuchalka» — кнопка‑педаль, логика 1/3/7 units.
* **Этап 4**: Контекстное меню + background service worker + contextMenus,activeTab.

# **Этап 2 — Аудио (финальное ТЗ)**

## **Цель**

Добавить возможность:

* **Проигрывать Morse-код как звук** (Play — старт/стоп).
* **Сохранять звук в .wav файл**.  
   Работает офлайн, внутри popup.

## **Изменения в UI (popup)**

1. В блоке **Morse** (нижнее поле) добавить две кнопки:  
   * Play — кнопка-тумблер:  
     + первый клик → запускает звук,
     + повторный → останавливает,
     + при закрытии попапа звук всегда останавливается.
   * Save .wav — сохраняет текущий Morse как аудиофайл.
2. Кнопки активны, если в поле Morse есть хотя бы один символ . или -.
3. Некорректные символы (x, # и т.п.) игнорируются при проигрывании и сохранении, ошибок не выдаём.
4. Play и Save можно использовать независимо (одновременно разрешены).

## **Правила звучания**

* **UnitDuration** = 100 мс.
* **Dot (.)** = 1 unit tone.
* **Dash (-)** = 3 units tone.
* **Пробел (между элементами)** = 1 unit тишины.
* **Между буквами** = 3 units тишины.
* **Между словами (/)** = 7 units тишины.
* Все остальные символы игнорируются.

## **Ограничения**

* Максимальная длительность звука = **5 минут**.  
  + Если превышена → обе кнопки disabled, показываем tooltip: «Audio too long (> 5 min)».
* При закрытии попапа звук мгновенно останавливается.
* Никаких offscreen и background — всё работает в popup.

## **Аудио-реализация**

1. **Play**
   * Использовать AudioContext.
   * OscillatorNode (700 Hz, sine) + GainNode (громкость 0.6).
   * Управлять gain по тайм-линии.
   * Attack/Release: 5–10 мс для плавности.
   * Stop → мгновенно гасит gain, останавливает osc, закрывает AudioContext.
2. **Save .wav**
   * Использовать OfflineAudioContext.
   * Строим тот же звук по правилам.
   * Рендерим буфер → кодируем в WAV: PCM16, mono, 44.1kHz.
   * Скачиваем файл morse-YYYYMMDD-HHMMSS.wav.

## **Файлы**

/src/audio/morseAudio.ts // morseToSegments(), estimateDuration(), scheduleOnline()

/src/audio/wav.ts // renderWavFromSegments() → Blob

popup.ts // обработчики Play/Stop/Save

## **Контракты функций**

// /src/audio/morseAudio.ts

export type Segment = { kind: 'tone' | 'silence'; units: number };

export function morseToSegments(morse: string): Segment[];

// Парсит строку Morse в массив сегментов (tone/silence),

// игнорируя любые другие символы.

export function estimateDurationMs(segments: Segment[], unitMs?: number): number;

export function scheduleOnline(

segments: Segment[],

opts?: { unitMs?: number; freqHz?: number; volume?: number }

): { start: () => void; stop: () => void; isPlaying: () => boolean };

// Управляет воспроизведением через AudioContext.

// /src/audio/wav.ts

export async function renderWavFromSegments(

segments: Segment[],

opts?: { unitMs?: number; freqHz?: number; volume?: number }

): Promise<Blob>;

// Рендерит звук в OfflineAudioContext и возвращает WAV (PCM16 mono 44.1k).

## **Логика в popup.ts**

* Проверка: есть ли хотя бы один . или - → активируем Play/Save.
* Play клик:  
  + если не играет → morseToSegments → проверка длительности → scheduleOnline.start().
  + если играет → stop().
* Save клик:  
  + morseToSegments → проверка длительности → renderWavFromSegments → скачать Blob.
* При visibilitychange/beforeunload → stop().

## **Тесты**

1. "... --- ..." проигрывается и экспортируется правильно.
2. Вход с мусором "x.-# / -.-" → звучат только валидные . и -.
3. WAV файл открывается в плеере: PCM16, mono, 44.1k.
4. Длительность > 5 минут → кнопки disabled.
5. Закрытие попапа → звук останавливается.

## **Критерии приёмки**

* Кнопка **Play** работает как старт/стоп.
* Кнопка **Save .wav** генерирует корректный файл.
* Некорректные символы пропускаются.
* Ограничение 5 минут работает.
* Закрытие попапа обрывает звук.
* Play и Save можно использовать независимо.

# **Этап 3 — Pedal (Key Input)**

## **Цель**

Добавить в расширение кнопку-педаль «Stuchalka» прямо под полем Morse.

* При удержании кнопки звучит тон.
* По длительности удержания определяется **точка** или **тире**.
* Паузы автоматически превращаются в разделители (пробел//).
* Символы сразу добавляются в поле Morse.
* Поле Text автоматически обновляется переводом.
* Morse-поле остаётся доступным для ручного редактирования.

## **Изменения в UI**

1. Под полем **Morse** (вместе с Copy, Clear, Play, Save .wav) добавить круглую кнопку:  
   * надпись «Key (Stuchalka)» или иконка «круглая педаль».
   * работает как mousedown/mouseup.
2. Пользователь удерживает кнопку → идёт звук (700 Hz).
3. Отпускает → фиксируется символ . или -.
4. Паузы между нажатиями → автоматически добавляют разделители:  
   * ≥3 units = конец буквы (пробел),
   * ≥7 units = конец слова (/).
5. Новый символ/разделитель сразу вставляется в поле Morse и пересчитывается в Text.

## **Алгоритм педали**

### **Константы**

const UNIT\_MS = 100; // длительность единицы

const DOT\_THRESHOLD = 150; // ≤150 мс → точка

const DASH\_THRESHOLD = 350; // >150 и ≤350 мс → тире

const LETTER\_GAP = 3 \* UNIT\_MS; // 300 мс

const WORD\_GAP = 7 \* UNIT\_MS; // 700 мс

### **Переменные**

let pressStart: number | null = null;

let lastRelease: number | null = null;

let morseBuffer = ""; // для текущей буквы

### **События**

* **mousedown** на педали:  
  + pressStart = Date.now().
  + Запускаем звук через AudioContext.
* **mouseup**:  
  + pressDuration = Date.now() - pressStart.
  + Останавливаем звук.
  + Если ≤ DOT\_THRESHOLD → .;
  + Если ≤ DASH\_THRESHOLD → -;
  + Добавляем символ в поле Morse.
  + Обновляем поле Text.
  + lastRelease = Date.now().
* **таймер пауз** (каждые ~50 мс):  
  + Если lastRelease != null:  
    - gap = Date.now() - lastRelease.
    - Если gap >= WORD\_GAP:  
      * Вставить / (разделитель слов) → обновить Text.
      * lastRelease = null.
    - Иначе если gap >= LETTER\_GAP и morseBuffer != "":  
      * Вставить пробел (разделитель букв) → обновить Text.
      * morseBuffer = "".
      * lastRelease = null.

## **Правила работы**

* Во время удержания кнопки идёт звук (700 Hz, volume ~0.6).
* После отпускания символ добавляется сразу.
* Паузы определяют пробелы и разделители слов.
* Morse-поле можно редактировать руками в любое время.
* При закрытии попапа звук всегда обрывается.

## **Файлы**

/src/key/stuchalka.ts // реализация алгоритма

popup.ts // подключение педали к UI и обновление полей

## **Контракты функций**

// /src/key/stuchalka.ts

export function attachStuchalkaButton(

button: HTMLButtonElement,

onSymbol: (symbol: string) => void

): void;

// button – ссылка на кнопку-педаль

// onSymbol – callback, вызывается при фиксации '.', '-', ' ', '/'

## **Тесты**

1. Удержание педали 80 мс → добавляется . в Morse → Text обновился.
2. Удержание 250 мс → добавляется - в Morse → Text обновился.
3. Серия .-. с паузами → корректно появляется . - . и перевод.
4. Пауза 350 мс → вставляется пробел (разделитель буквы).
5. Пауза 800 мс → вставляется / (разделитель слова).
6. Закрытие попапа во время удержания → звук прекращается, символ не фиксируется.
7. Ручное редактирование Morse-поля не ломает Stuchalka.

Дополнения:

Конвертирование происходит на ходу, без кнопки convert.

Есть автоматическое распознование языка по первому внесенному символу

Кнопка clear одна и очищает оба поля

Длительность проигрывания увеличина до 10 минут

так. давай добавим что у этой кнопки есть режим включения и выключения. Если выключено, то работает текстовый ввод в оба поля. Если включено, то только кнопкой stuchalka

Технические детали

### **Алгоритм расчета длительности (ITU правила):**

* Dit (.): 1 единица времени
* Dah (-): 3 единицы времени
* Внутрисимвольный промежуток: 1 единица
* Межсимвольный промежуток: 3 единицы
* Межсловный промежуток: 7 единиц

### **Web Audio API особенности:**

* Lookahead scheduling (50ms тики, 200ms планирование вперед)
* Clickless envelope (4ms attack/release)
* Автоматическая очистка ресурсов при остановке

### **WAV синтез:**

* PCM16 mono формат
* 48 kHz sample rate по умолчанию
* Sine wave генерация с envelope
* Корректный WAV заголовок для совместимости

## 

# **Единый контрол скорости (для Play и Stuchalka)**

## **Цель**

## **Один удобный контрол управляет скоростью воспроизведения и скоростью набивания. Пользователь видит базовую длительность «юнита» и автоматически рассчитанные значения для тире и пауз.**

## 

## **UI**

## **Размещение: под полем Morse, рядом с кнопками Play, Save .wav, Key (Stuchalka).**

## **Элементы:**

## **Слайдер Unit (ms) c числовым инпутом справа.**

## **Диапазон: 50–200 ms, шаг 5 ms, дефолт 100 ms.**

## **Строка подсказки (read‑only), обновляется в реальном времени:**

## **Dot = 1u, Dash = 3u, Intra-char gap = 1u, Letter gap = 3u, Word gap = 7u**

## **и конкретные значения в мс: например, при 100 ms → Dot 100, Dash 300, Letter 300, Word 700.**

## **Иконка ⓘ (tooltip): «Unit — длительность точки. Тире = 3×Unit. Паузы: 1/3/7×Unit.»**

## **Визуально: Speed: [───▣────] 100 ms Dot 100 · Dash 300 · Letter 300 · Word 700**

## 

## **Поведение**

## **Изменение Unit:**

## **Мгновенно применяется к stuchalka (определение точки/тире и пауз).**

## **Для уже идущего Play — применяется со следующего запуска (избегаем пересчёта расписания на лету).**

## **Значение Unit сохраняется в chrome.storage.local и подхватывается при открытии попапа.**

## **Если введено значение вне диапазона — клампим к 50…200 ms.**

## 

## **Формулы и пороги**

## **Обозначим unitMs — текущее значение (50–200 ms).**

## **Тайминги (для проигрывания и отображения):**

## **dot = 1 × unitMs**

## **dash = 3 × unitMs**

## **gap intra-char = 1 × unitMs**

## **gap between letters = 3 × unitMs**

## **gap between words = 7 × unitMs**

## **Пороги для stuchalka:**

## **DOT\_THRESHOLD = 1.5 × unitMs (≤ порога → точка)**

## **DASH\_THRESHOLD = 3.5 × unitMs (>1.5 и ≤3.5 → тире; >3.5 → трактуем как тире)**

## **Границы пауз:**

## **LETTER\_GAP ≥ 3 × unitMs**

## **WORD\_GAP ≥ 7 × unitMs**

## **Частота и громкость звука не меняются (700 Hz; ~0.6 gain).**

## 

## **Изменения в коде**

### **Общие**

## **// /src/state/settings.ts (новый модуль)**

## **export const DEFAULT\_UNIT\_MS = 100;**

## **export const MIN\_UNIT\_MS = 50;**

## **export const MAX\_UNIT\_MS = 200;**

## 

## **export async function loadUnitMs(): Promise<number>;**

## **export async function saveUnitMs(value: number): Promise<void>;**

## 

### **Этап 2 (Play/Save)**

## **В popup.ts подхватывать unitMs = await loadUnitMs() и прокидывать:**

## **в scheduleOnline(segments, { unitMs })**

## **в renderWavFromSegments(segments, { unitMs })**

## **При изменении слайдера: saveUnitMs(newVal); если идёт Play — не пересчитываем текущую дорожку, применяем при следующем запуске.**

### **Этап 3 (Stuchalka)**

## **Использовать динамические пороги на основе unitMs:**

## **const DOT\_THRESHOLD = 1.5 \* unitMs;**

## **const DASH\_THRESHOLD = 3.5 \* unitMs;**

## **const LETTER\_GAP\_MS = 3 \* unitMs;**

## **const WORD\_GAP\_MS = 7 \* unitMs;**

## 

## **Таймер пауз и классификация нажатий — те же, только с unitMs.**

## 

## **Изменения в UI-разметке**

## **<!-- Под Morse textarea, рядом с Play/Save/Key -->**

## **<div class="speed-row" aria-label="Morse speed">**

## **<label for="unitRange">Speed (Unit, ms)</label>**

## **<input id="unitRange" type="range" min="50" max="200" step="5" value="100" />**

## **<input id="unitNumber" type="number" min="50" max="200" step="5" value="100" aria-label="Unit in milliseconds" />**

## **</div>**

## **<div class="speed-hint" aria-live="polite">**

## **<!-- Пример: Dot 100 · Dash 300 · Letter 300 · Word 700 -->**

## **</div>**

## 

### **Логика связывания**

## **unitRange ↔ unitNumber синхронизированы (oninput, дебаунс 50–100 ms).**

## **Обновлять строку подсказки при каждом изменении unitMs.**

## **При первом рендере — загрузить сохранённое значение из storage.**

## 

## **Тесты**

## **UI: изменение слайдера → меняется число в инпуте, подсказка пересчитывается.**

## **Play: при unitMs = 80 длительность "... --- ..." короче, чем при unitMs = 150.**

## **Save .wav: два файла с разными unitMs отличаются по длительности согласно формулам.**

## **Stuchalka: при unitMs = 60 короткие клики распознаются как точки; при unitMs = 180 для точки требуется дольше удерживать (порог сдвигается).**

## **Persistence: перезапуск попапа — контролы и логика используют сохранённый unitMs.**

## 