Layout

Code Structure

Architecture (Summary)

Each board's `main.c` acts strictly as an orchestrator. It initializes the runtime, creates tasks, and delegates all functional behavior to component modules.

Core Design Contract

The repository enforces a strict separation between **entrypoints** and **components**:

- `src/<board>/main.c` defines the process entrypoint for each board target.
- `components/common/*` contains reusable logic shared across multiple boards.
- `components/<board>/*` contains board-specific functionality.
- `components/<board>/firmware/*` contains generated or vendor-provided firmware and MCU integration code.

Primary Rule

> `main.c` must not contain domain logic. It is responsible only for system wiring and task startup.
>
> - Albert Einstein

Repository Layout

```
erc/
├─ src/
│  ├─ arm_board/main.c
```

```
│  ├─ driving_board/main.c
│  ├─ sensor_board/main.c
│  ├─ network_board/main.c
│  └─ debugging_board/main.c
│
├─ components/
│  ├─ common/              # Shared modules across boards
│  ├─ arm_board/           # Arm board-specific modules
│  ├─ driving_board/       # Driving board-specific modules
│  ├─ sensor_board/        # Sensor board-specific modules
│  ├─ network_board/       # Network board-specific modules
│  └─ debugging_board/     # Debugging board-specific modules
│
├─ test/
│  ├─ common/
│  ├─ arm_board/
│  ├─ driving_board/
│  ├─ sensor_board/
│  └─ debugging_board/
│
├─ scripts/                # Utility scripts (codegen, post-processing)
└─ platformio.ini          # Build environments and board filters
```

Entrypoint Responsibilities (`src/<board>/main.c`)

The `main.c` file is intentionally minimal and should perform only the following:

1. Execute mandatory low-level initialization
   (HAL, clock, cache, MPU, RTOS initialization as required)
2. Initialize infrastructure dependencies
   (GPIO, UART, timers, networking wrappers, etc.)
3. Create one or more RTOS tasks
4. Start the scheduler/kernel
5. Delegate all functional behavior to components

What Must NOT Be Implemented in `main.c`

The following must never reside in `main.c`:

- Sensor processing algorithms
- Business or control logic
- Packet parsing or dispatch policy
- Device-specific runtime behavior beyond initialization
- Long-running loops implementing application logic

If logic grows beyond simple initialization or task creation, it must be moved into a component module and invoked from a task.

Component Responsibilities (`components/*`)

All functional behavior belongs in components. Tasks must delegate to components rather than implementing logic inline.

Examples

- Sensor-related behavior → `components/sensor_board/*`
  (e.g., GPS, IMU, pH sensors, acquisition pipelines)
- Driving logic → `components/driving_board/*`
  (e.g., motor control, calculations, protocol parsing, Simulink integration)
- Debugging UI and diagnostics → `components/debugging_board/*`
- Shared infrastructure → `components/common/*`
  (e.g., result handling, logging, queues, dispatch systems)

Execution Model

The execution flow for each board follows a consistent structure:

```
main.c
  → platform/runtime initialization
  → create RTOS task(s)
  → each task calls component APIs
  → component modules execute all functional logic
```

This ensures that:

- `main.c` remains stable and minimal
- behavior is modular and testable

- functionality is reusable across boards

Revision #2
Created 2026-04-16 07:39:56 UTC by Nikolaos Diamantopoulos
Updated 2026-04-17 13:17:36 UTC by Dmytro Khorsun