# Table of Contents

# Enterprise Platform Development

- Developing an Enterprise-level Embedded software stack from scratch by identifying and integrating required components
- Flexibility and control over every part of Firmware development i.e from the Linker script level to the firmware application design.
- Use modern build systems (CMake and Meson) to make a IDE independent development environment to reduce cost of development and maintenance.
- Integrate tooling such as static analysis, clang-format, unit testing, mocking and debugging to improve the quality of code.

# Repository Structure

- Baremetal
    - Low level Linker script and learning
    - Integrate critical external components to a project
- doc
    - Component datasheeets
    - Board schematics and microcontroller datasheets
- STM32_HAL
    - STM32 HAL based examples for various microcontrollers
- Template
    - Projects built after Minimal_CMSIS in Baremetal
    - Actual application level code with various levels of tooling support
    - Takes each core part of an embedded system and creates a project around it
- third_party
    - Third Party software ZIP files used in Template projects
- tools
    - Tools used by the project, pre-installed in your system
    - Links or ZIP files

# Project Structure

Create a folder structure that would need minimal change when porting between different architectures and microcontrollers

To port our project to a different controller we would need to update these folders

- l0_lowlevel
- l1_third_party_device_specific
- l2_drivers

To port our project to a different board we would need to update these folders

- l2_board_specific
- l5_application

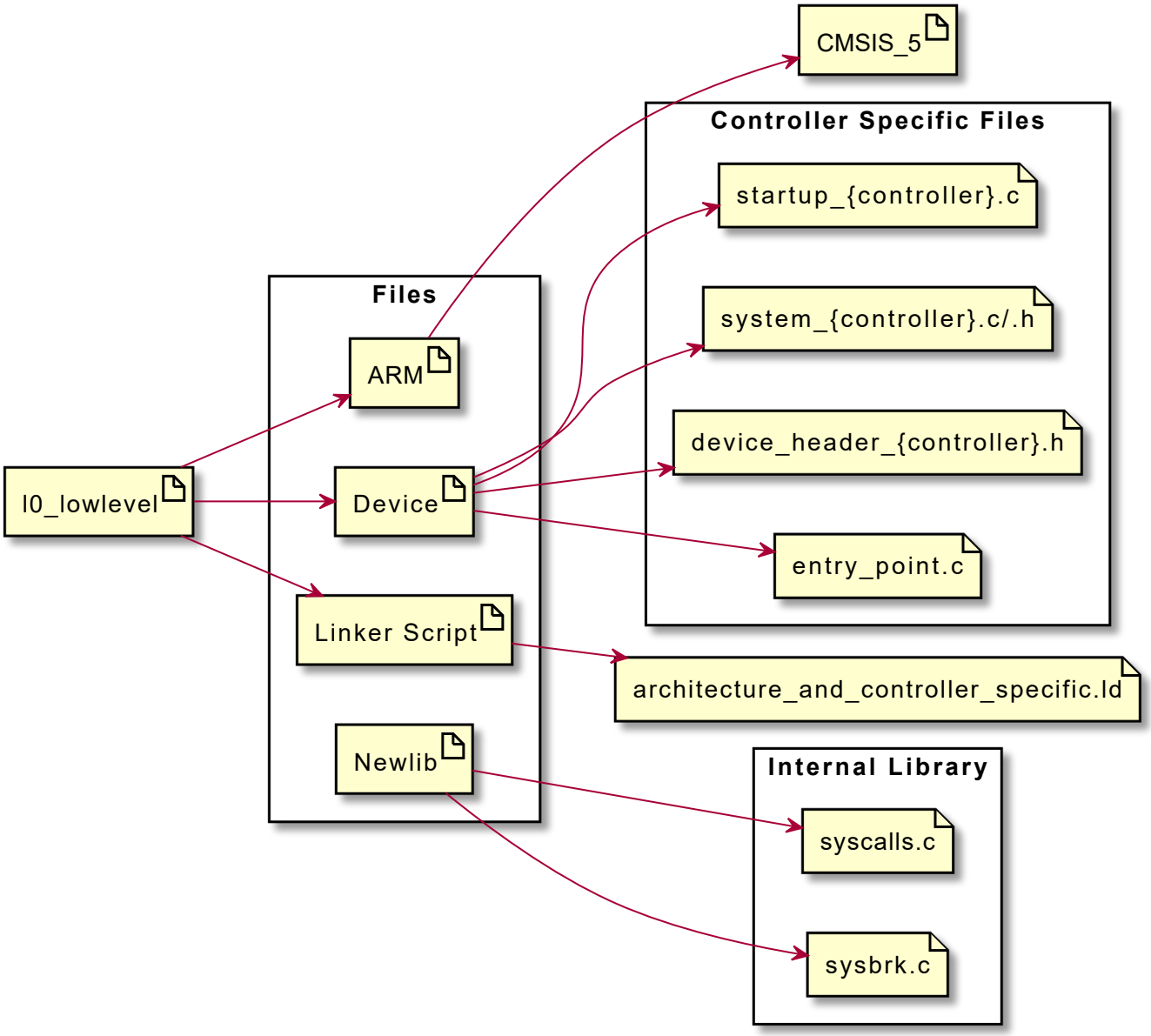
Platform independent code

- l3_functional_third_party
- l4_module (Design drivers that use the platform agnostic l2_drivers)

```
+-- l0_lowlevel
| +-- arm_cmsis
| +-- linker_script
| +-- device
| +-- toolchain_specific_syscalls
+-- l1_third_party_device_specific
| +-- RTOS
| +-- External_Libraries_device_specific
+-- l2_drivers
| +-- gpio_device_specific
| +-- uart_device_specific
+-- l2_board_specific
| +-- peripheral_initialization
+-- l3_functional_third_party
| +-- ring_buffer
+-- l4_module
| +-- sensors/actuators
| +-- technology
| +-- protocol
+-- l5_application
| +-- application_logic
| +-- main.c
```

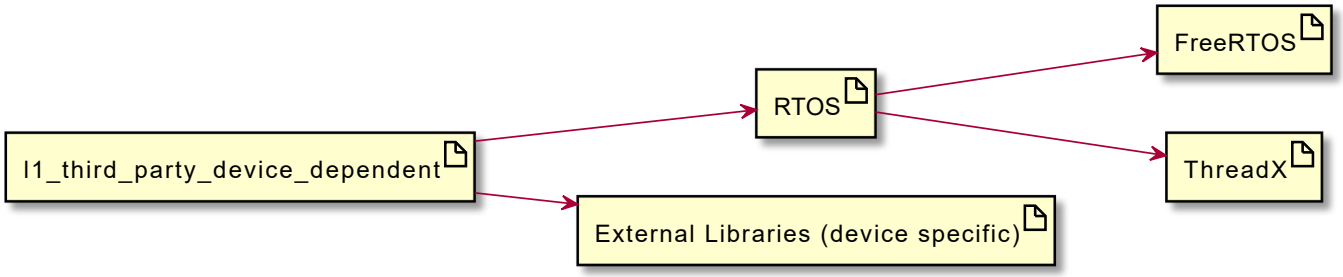
# L0 Lowlevel

- ARM CMSIS 5 has different compiler and architecture specific changes
- ARM CMSIS 5 Linker script is dependent on architecture of microcontroller
- Device Header is dependent of the Microcontroller Manufacturer
- Device Startup is dependent on ARM Architecture and Microcontroller Manufacturer



## L1 Third Party device dependent

- Certain Third Party software changes its behaviour based on the architecture and device
- For example: FreeRTOS needs to be configured differently according to different microcontroller family and architectures

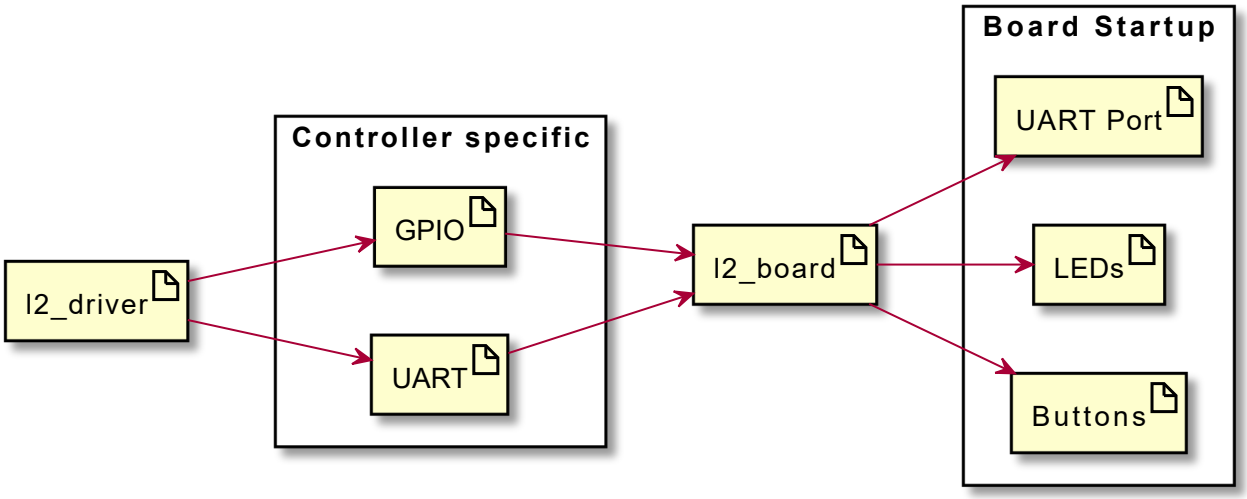



## L2 Drivers

- Basic drivers for **GPIO**, **Interrupt Handing**
- Basic protocol for **UART**, **SPI**, **I2C**
- These are microcontroller and architecture dependent

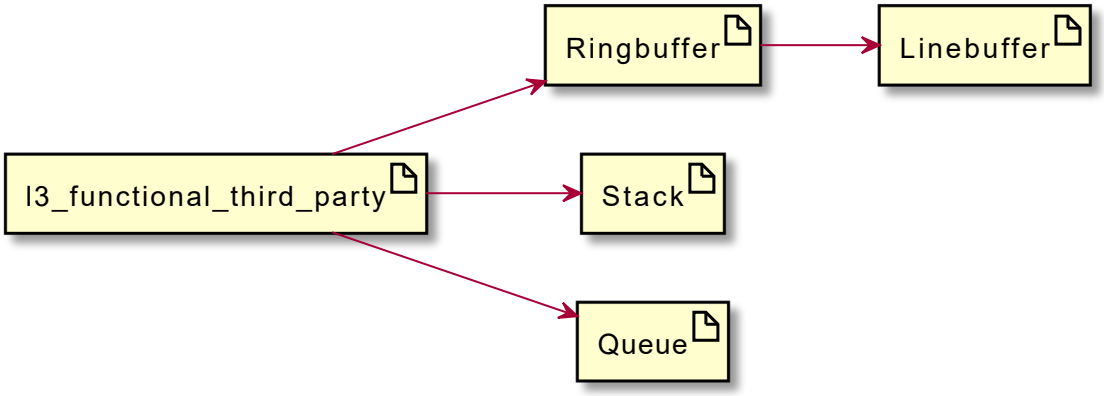
## L2 Board

- For internal board_specific initialization
- Syscalls based externed functions



# L3 Functional third party library

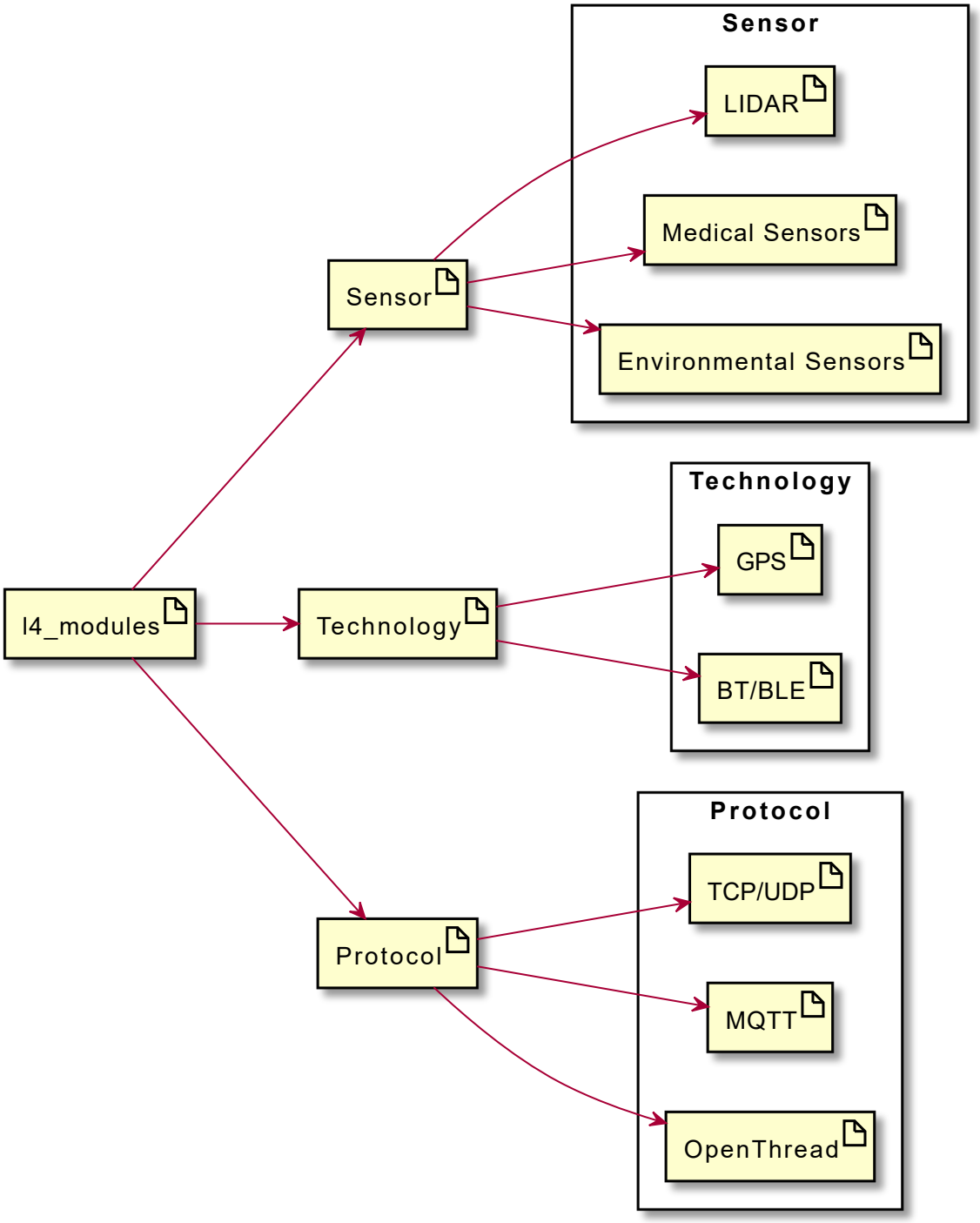
- Functional Third Party code integrated into the project
- For example: **Ring Buffer**, **JSON Library**



# L4 Modules

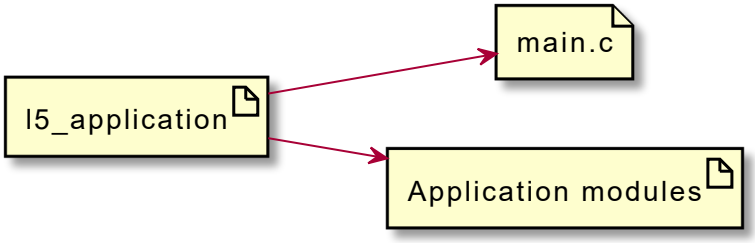
Writing code for various hardwares i.e

- Communication
- Sensors
- Actuators

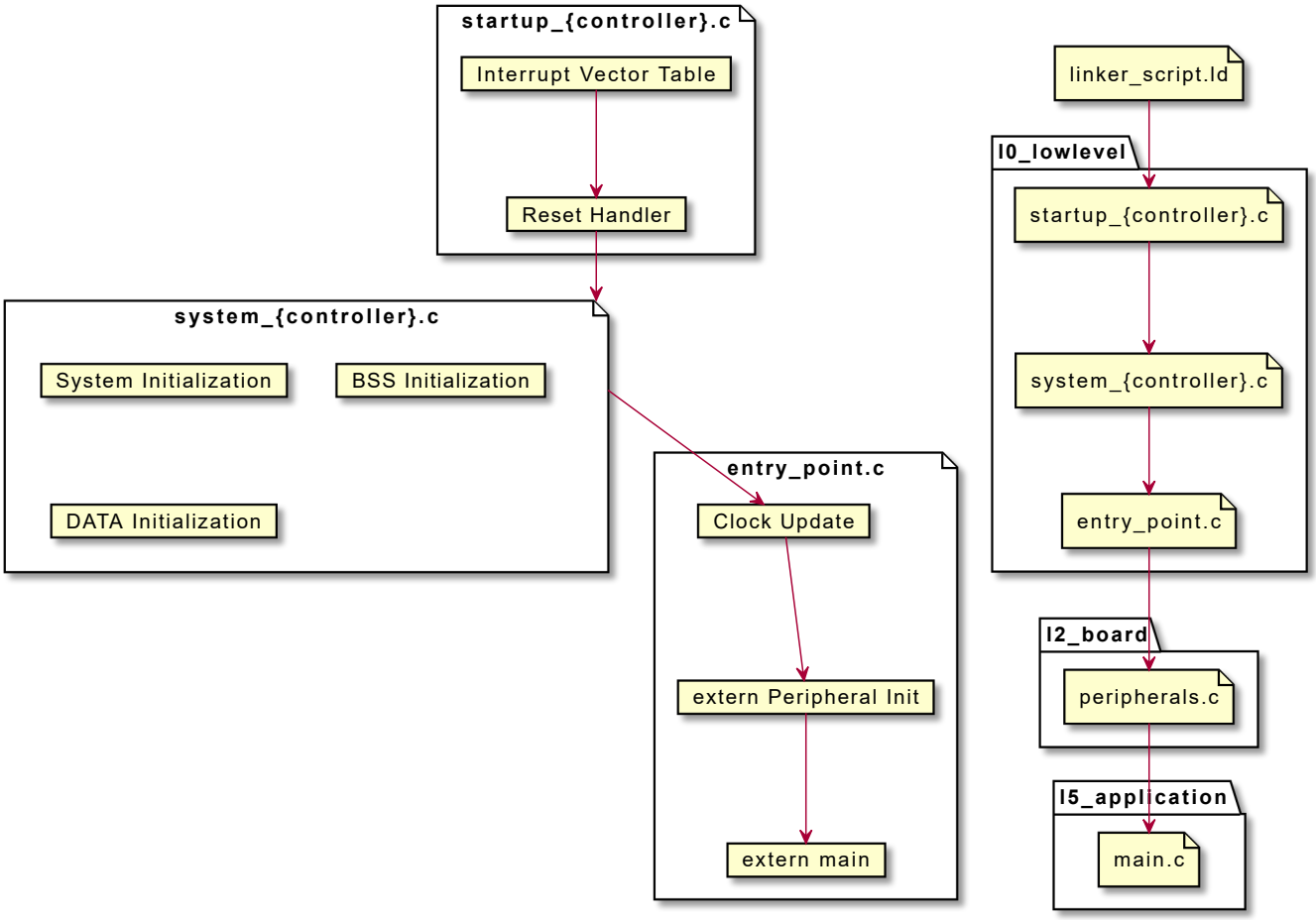


## L5 Application

- Application Logic for the project
- main.c resides on the top level




# Startup to Main (Use-case)



# Startup to Main (Sequence)