

The background features a complex network of thin, light-colored lines and dots, forming a web-like structure. Overlaid on this are several translucent triangles of various sizes and orientations, some of which are slightly offset from the main network, creating a layered, geometric effect.

# ESP-IDF Build System

---

Brief Overview

---

# ESP-IDF Build System Overview

- Review of Espressif [Documentation](#)
    - Build System Concepts
    - Using the Build System & CMake
    - Example CMake Project Setup
    - Project CMakeLists File
    - Minimal Component CMakeLists
-

---

# Build System Concepts

## ESP-IDF Build System Concepts [Documentation](#)

- Project → a directory that contains all files and configuration to build a single “app”.
  - Project Configuration → is held in a single file called `sdkconfig` in the root directory of the project.
  - app → an executable which is built by ESP-IDF.
  - Components → are modular pieces of standalone code which are compiled into static libraries (.a files) and linked into an app.
  - Target → the hardware for which an application is built.
-

---

# Using the Build System & CMake

- `idf.py`
    - CMake → configures the project to be built.
    - Ninja → builds the project.
    - `esptool.py` → used for flashing the target.
  - CMake Directly
    - `idf.py` is a wrapper around CMake for convenience.
  - CMake in an IDE
    - Use an IDE with CMake integration. *We'll use the Espressif, Eclipse-based IDE.*
-

---

# Example CMake Project Setup

## Example CMake Project Setup [Documentation](#)

- Top-level Project CMakeLists.txt File
- “sdkconfig” Project Configuration File
- Optional “components” Directory
- “main” Directory
- “build” Directory
- Kconfig.projbuild

```
- myProject/  
  - CMakeLists.txt  
  - sdkconfig  
  - components/  
    - component1/  
      - CMakeLists.txt  
      - Kconfig  
      - src1.c  
    - component2/  
      - CMakeLists.txt  
      - Kconfig  
      - src1.c  
      - include/  
        - component2.h  
  - main/  
    - CMakeLists.txt  
    - src1.c  
    - src2.c  
  
  - build/
```

---

# Project CMakeLists File

## Project CMakeLists File [Documentation](#)

- Each project has a single top-level CMakeLists.txt file that contains build settings for the entire project. By default, the project CMakeLists can be quite minimal.

### Minimal Example CMakeLists

Minimal project:

```
cmake_minimum_required(VERSION 3.16)
include($ENV{IDF_PATH}/tools/cmake/project.cmake)
project(myProject)
```

---

---

# Minimal Component CMakeLists

## Minimal Component CMakeLists [Documentation](#)

- The minimal component CMakeLists.txt file simply registers the component to the build system using `idf_component_register`:

```
idf_component_register(SRCS "foo.c" "bar.c"  
                      INCLUDE_DIRS "include"  
                      REQUIRES mbedtls)
```

The background is a light cream color with a complex geometric pattern. On the left side, there is a dense network of thin, light brown lines connecting small circular nodes, forming a web-like structure. Scattered across the entire background are numerous triangles of various sizes and orientations, some outlined in thin brown lines and others in thin yellow lines. Some of these triangles are filled with a very light yellow color. In the upper right corner, there are several small, faint clusters of dots, resembling a starry sky or a distant galaxy. The overall aesthetic is clean, modern, and mathematical.

# Next lesson

---