



General Purpose Frameworks

- ESP-IDF
- ESP-Jumpstart
- ESP-AT
- ESP-Hosted
- ESP-RainMaker
- Amazon:FreeRTOS

Application Specific Frameworks

- ESP-HomeKit
- ESP-Mesh
- ESP-Audio
- ESP-Who
- ESP-Skainet



ESP-IDF

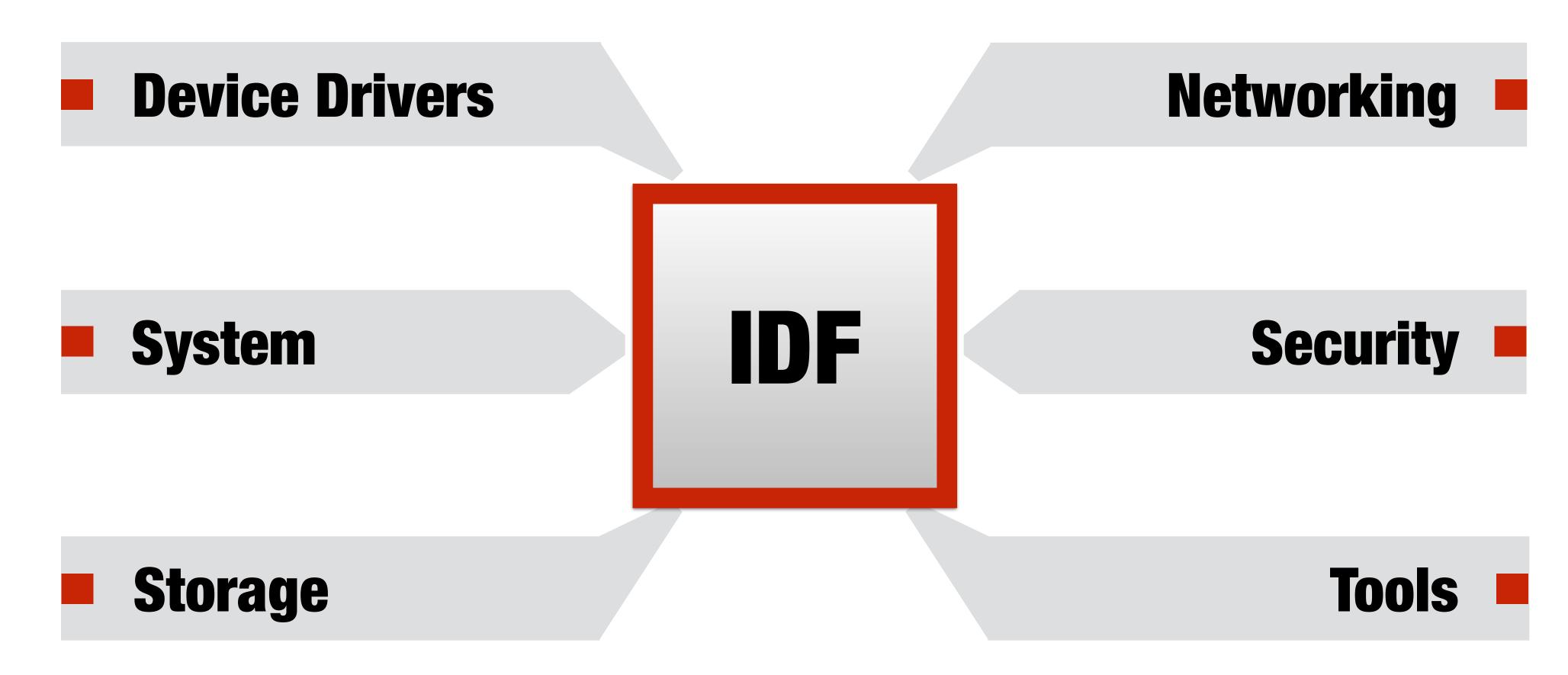


- Espressif's own SDK for C and C++ application development
- Suitable for any generic application development
- Apache 2.0 license
- Most of the components available in the source format
- Linux, Mac OS-X and Windows as a development host
- Command-line and Eclipse IDE integration for development

Get it from: https://github.com/espressif/esp-idf



IDF Components





Device Drivers

		Ethernet	BT/BLE	Wi-Fi
Network Periph				
SD	UART	I2C	I2S	GPIO
	RTC	Watchdog	ADC/DAC	SPI
Common Periph				
Crypto Acc	CAN	Sigma-Delta	RMT	Capacitive Touch
				LED Controller
ESP32 Unique Peripl	ESP3			

FreeRTOS

Boot Loader

Newlib

Heap Allocator

Core Components

PThread Layer

C++ Support (C11)

Compatibility Layer

Power Management

ULP

Power Management

Console

Misc (Logger, Panic handler, App trace)

Other

Partition Manager

EFUSE Manager

OTA Support

System Components

NVS

Object Storage

VFS Layer

Wear Levelling

FATFS

SPIFFS

File Systems

Networking

LwIP

WPA Supplicant

WiFi / Ethernet

mDNS Responder

HTTP Client / Server

CoAP

Websockets

mbedTLS

HTTP 2.0 Client

TCP/IP Based Protocols

BlueDroid (BT + BLE)

NimBLE (BLE)

ESP-BLE-Mesh

BT/BLE

SoftAP Provisioning*

BLE Provisioning*

Protocomm

JSON / Protobuf

Higher Level

Secure Boot

HW RNG

NVS Encryption

Crypto Accelerators

Platform Security

mbedTLS

OpenSSL Compat

Network Security

LibSodium

Generic Crypto

Build System (Make, CMake)

Kconfig

Footprint Analyzer

Flash Programmer

OpenOCD JTAG

Serial Monitor

SystemView

App Tracer

GCov

Core Dump

NVS Image Creator

Mass Mfg Tool



Cloud Agents







- Amazon AWS IoT https://github.com/espressif/esp-aws-iot
- Google Cloud IoT https://github.com/espressif/esp-google-iot
- Azure IoT https://github.com/espressif/esp-azure
- Production quality examples available for each



ESP Jumpstart

https://github.com/espressif/esp-jumpstart



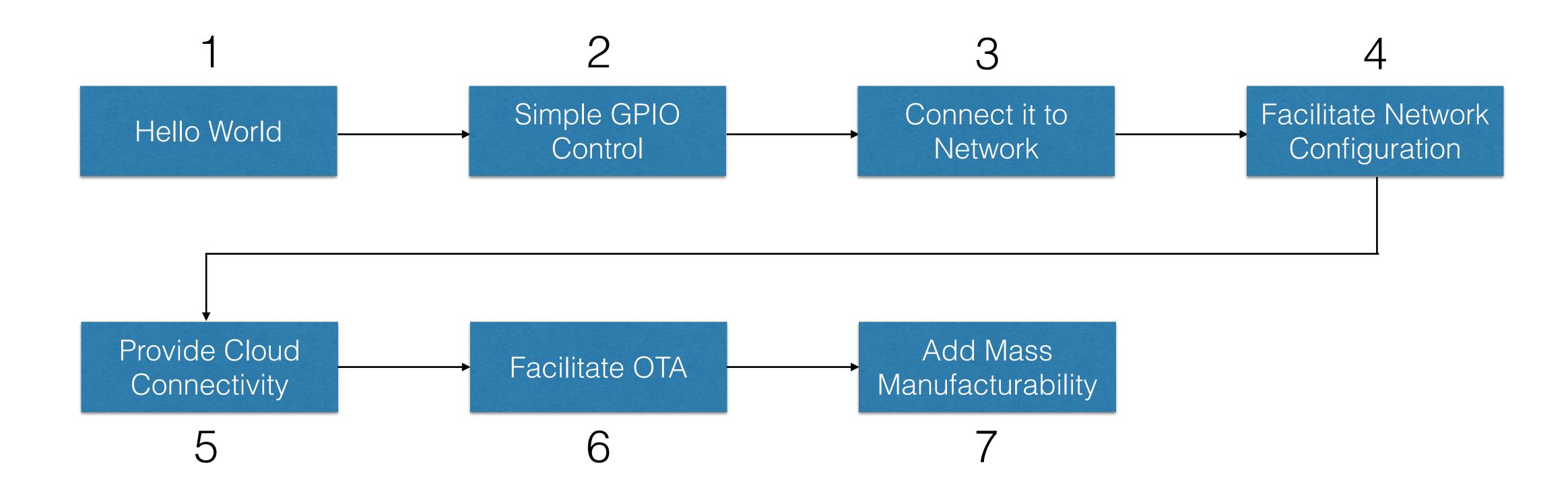
Why ESP-Jumpstart?

- All our customers need not reinvent the wheel when it comes to common product use-cases
- Building production ready firmware can be hard and requires multiple decisions to be taken
- Ready reference and boilerplate code can significantly simplify this
- However it can't be a black-box and step-by-step understanding is required for easy changes based on the use-cases



What is ESP-Jumpstart?

• ESP-Jumpstart provides a step-by-step tutorial with production quality boilerplate code for following steps:





ESP-AT

- A WiFi and BLE connectivity solution for products having external host microcontroller
- No programming required on ESP32 side
- AT Command set (documentation can be found <u>here</u>)
- Available transports UART, SPI (master) and SDIO (slave)

Get it from: https://github.com/espressif/esp32-at



- System Commands (reset, power-save, file-system, diagnostics etc.)
- WiFi Commands (mode, scan, connect/disconnect, softAP configuration etc.)
- TCP/IP Commands (socket, open, send, recv, TLS configuration etc.)
- BLE Commands (scan, advertise, GATT server, GATT client, SMP configuration etc.)



ESP-HOSted

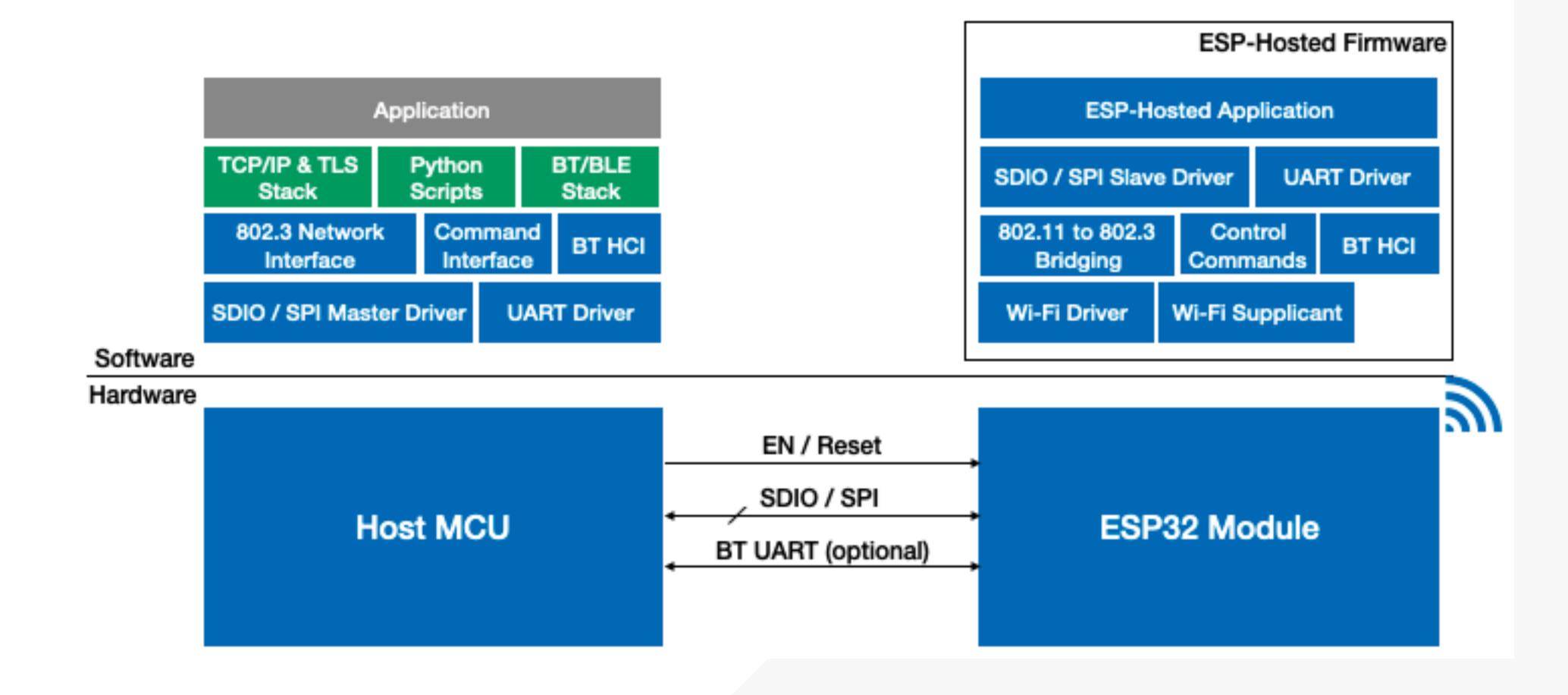


ESP Hosted

- ❖ A solution to use ESP32 with a separate host MCU or MPU
- Solution includes ESP32 firmware, Linux Driver, STM32 reference implementation and host command library
- Supports SDIO and SPI based transport
- Data Path 802.3 network interface on host side with packet bridging
- Control Path Google Protobuf based command-response protocol over SDIO/SPI or separate UART
- ❖ BT/BLE HCI interface available over SDIO/SPI or separate UART



Hardware and Software





Additional Information

Open-source release available - https://github.com/espressif/esp-hosted



ESP-RainMaker



A Typical IoT Product

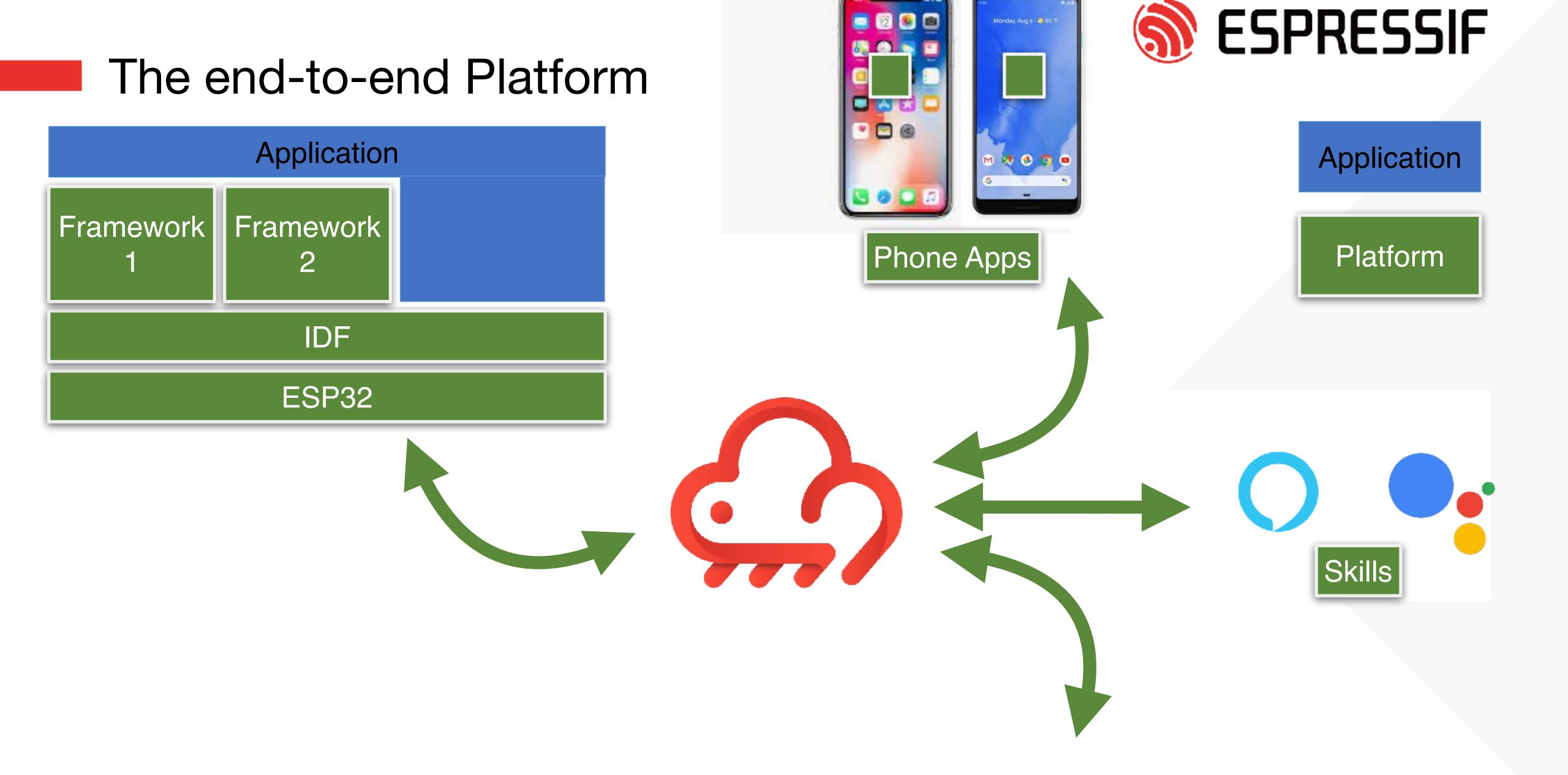
A Typical IoT Product Requires 3 Software Development Efforts

- Device Firmware A device side business logic, peripheral control, cloud connectivity
- Phone Apps Initial Wi-Fi configuration, user account creation, device control and monitoring
- Cloud Application A conduit for device data for control and monitoring, voice assistants integration, data logging, cloud side business logic





ESP-Rainmaker is a platform that allows developers to build connected devices with ESP32 and ESP32-S2 without requiring any cloud or phone application side development





Rainmaker Agent

- Rainmaker Device Agent is an SDK component
- It allows device description to the cloud, define different characteristics, their types, limits and assign UI elements
- The complexity of cloud communication is hidden from the application
- Rainmaker Agent also abstracts out workflows such as Wi-Fi configuration, user-to-device association.



Rainmaker Agent

```
/* Create a device and add the relevant parameters to it */
esp_ra_create_device("Light", common_callback);
esp_ra_device_add_type("Light", "esp.device.lightbulb");

esp_ra_device_add_param("Light", "output", esp_ra_bool(false), NULL, PROP_FLAG_READ | PROP_FLAG_WRITE);
esp_ra_param_add_ui_type("Light", "output", "esp-ui-toggle");

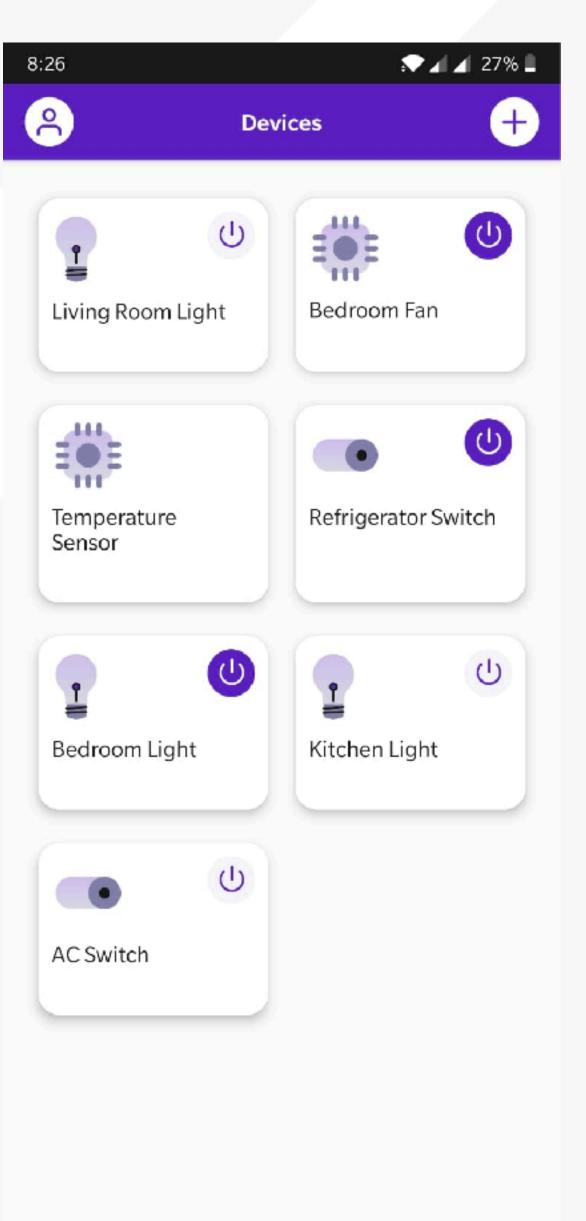
esp_ra_device_add_param("Light", "brightness", esp_ra_int(50), NULL, PROP_FLAG_READ | PROP_FLAG_WRITE);
esp_ra_param_add_ui_type("Light", "brightness", "esp-ui-slider");
esp_ra_param_add_bounds("Light", "brightness", esp_ra_int(0), esp_ra_int(100), esp_ra_int(0));

/* Start the ESP RainMaker Agent */
esp_ra_start();
```

Rainmaker Phone Apps

- Rainmaker Phone Apps are self-adapting to display device defined parameters
- They support User sign-up, login, Wi-Fi configuration, user-device association and device control
- Both iOS and Android apps available in full source format







Rainmaker Cloud

- Rainmaker cloud is built using AWS Serverless architecture
- Users strong X.509 based mutual authentication with devices
- It is completely transparent for device to phone apps and device to voice assistant skills communication
- With Serverless architecture and careful design it provides high levels of scalability, security and availability at low cost
- The service is free for makers and can be deployed in customers' AWS accounts for production





Additional Information

- Cloud RESTful APIs and Python bindings for third-party service integration available
- Amazon Alexa integration and Google Assistant integration available
- Website: http://rainmaker.espressif.com
- Available on GitHub: https://github.com/espressif/esp-rainmaker



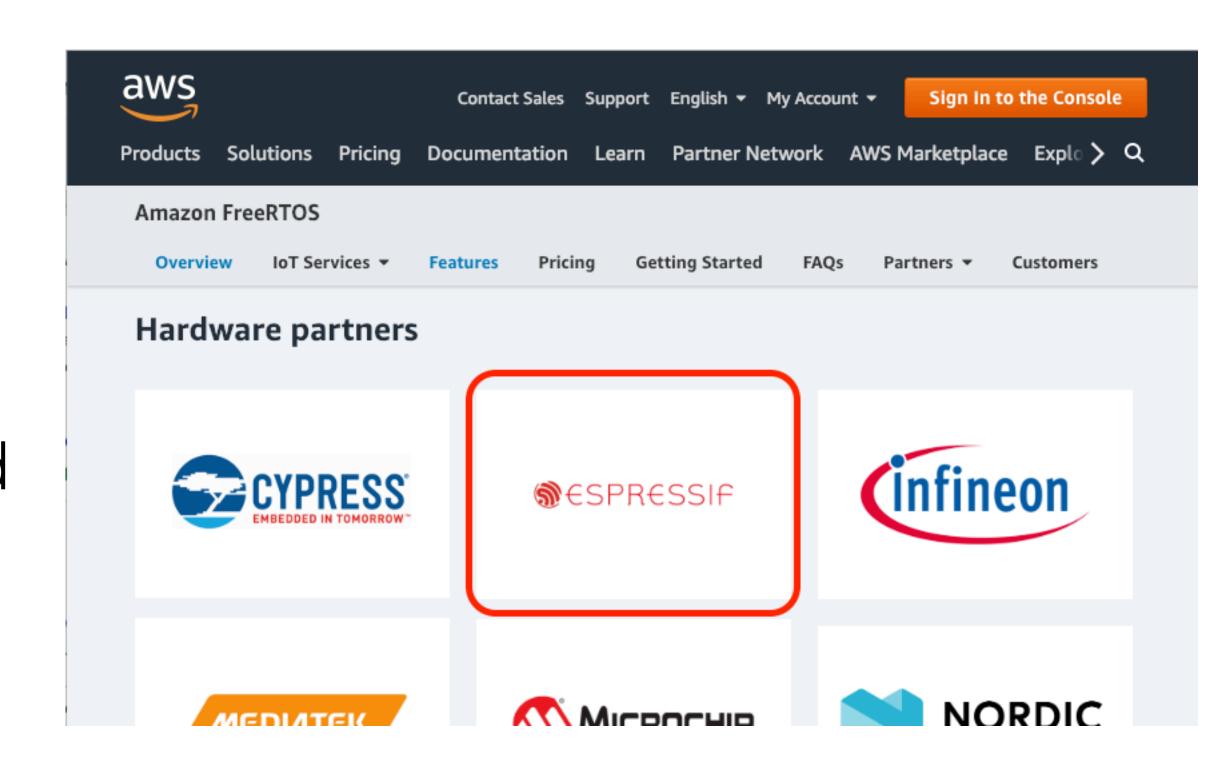
Amazon:FreeRT0S

https://github.com/aws/amazon-freertos/



Amazon:FreeRTOS on ESP32

- Amazon's hardware partner for Amazon:FreeRTOS with its ESP32 chipset
- Multiple Modules and Devkits qualified for AFR
- Support for single-core ESP32 configuration





ESP-HomeKit

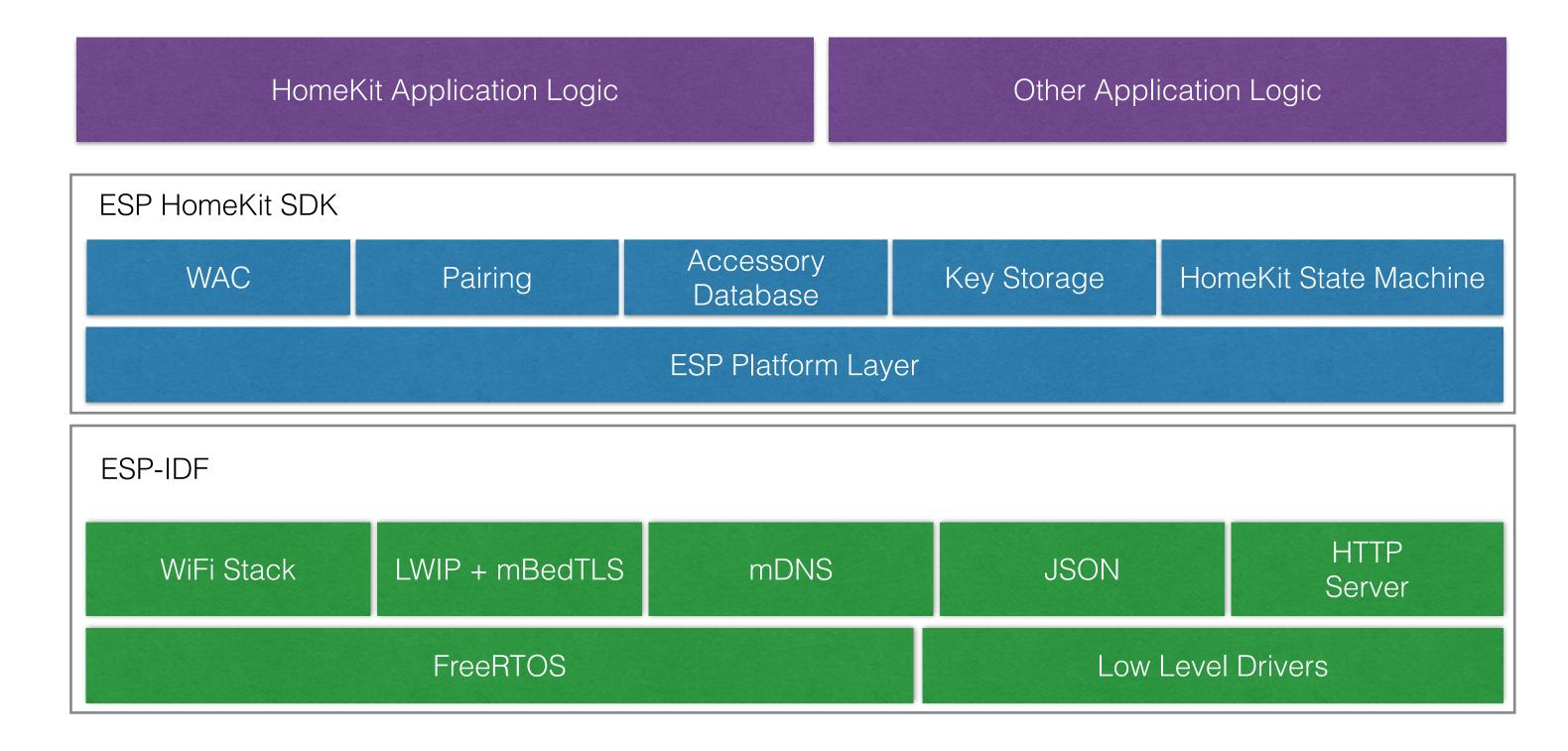
What's HomeKit

- Apple's smart-home protocol that creates a common language for device communication
- Works over WiFi or BLE protocol
- Wireless Accessory Configuration Easy network provisioning for WiFi accessories
- •iOS based control iPhone, iPad, Apple Watch, HomePod can control accessories
- Remote control possible via iPad, Apple TV or HomePod



ESP HomeKit SDK



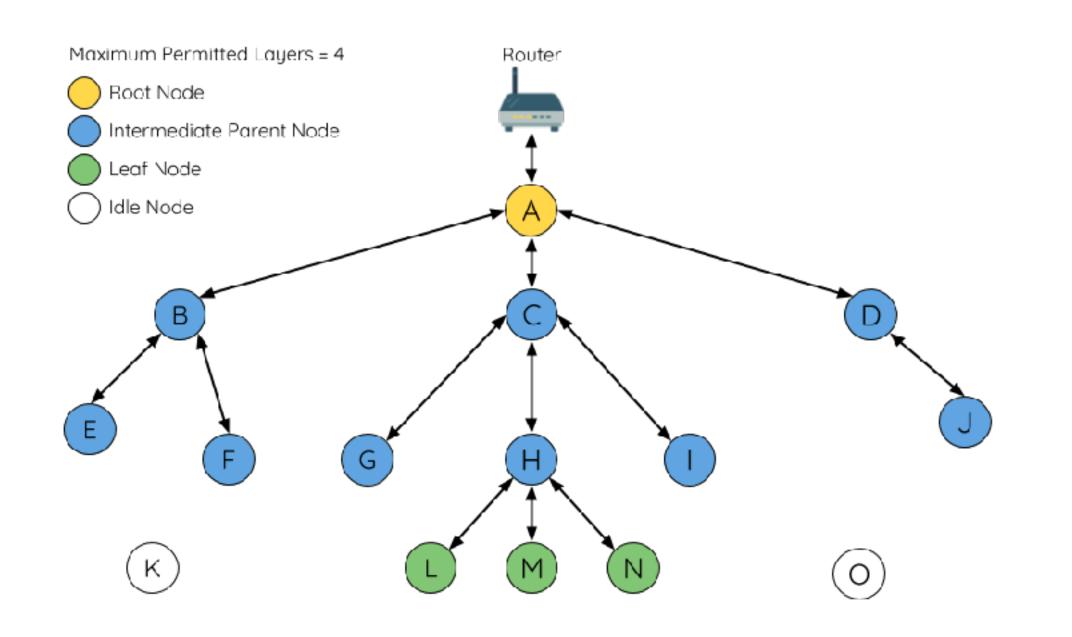


- Based on ESP-IDF
- C based memory-efficient implementation
- Compliant with latest R13 version of HomeKit Accessory Protocol Specification
- Supports MFi co-processor based or Software Authentication



ESP-Mesh

- Self-forming, self-healing network
- Based on standard WiFi protocol
- No special infrastructure requirement



- High Throughput Upto 10 Mbps per hop
- BLE based configuration
- Auto or manual formation support
- Multicast and Broadcast messaging support
- Low-Power operation supported for battery operated leaf-nodes



Application Use-cases

- Range extension for smart-home devices to cover WiFi blind spots
- Larger networks without special infrastructure parking places, public lighting, building automation etc.



ESP-Audio

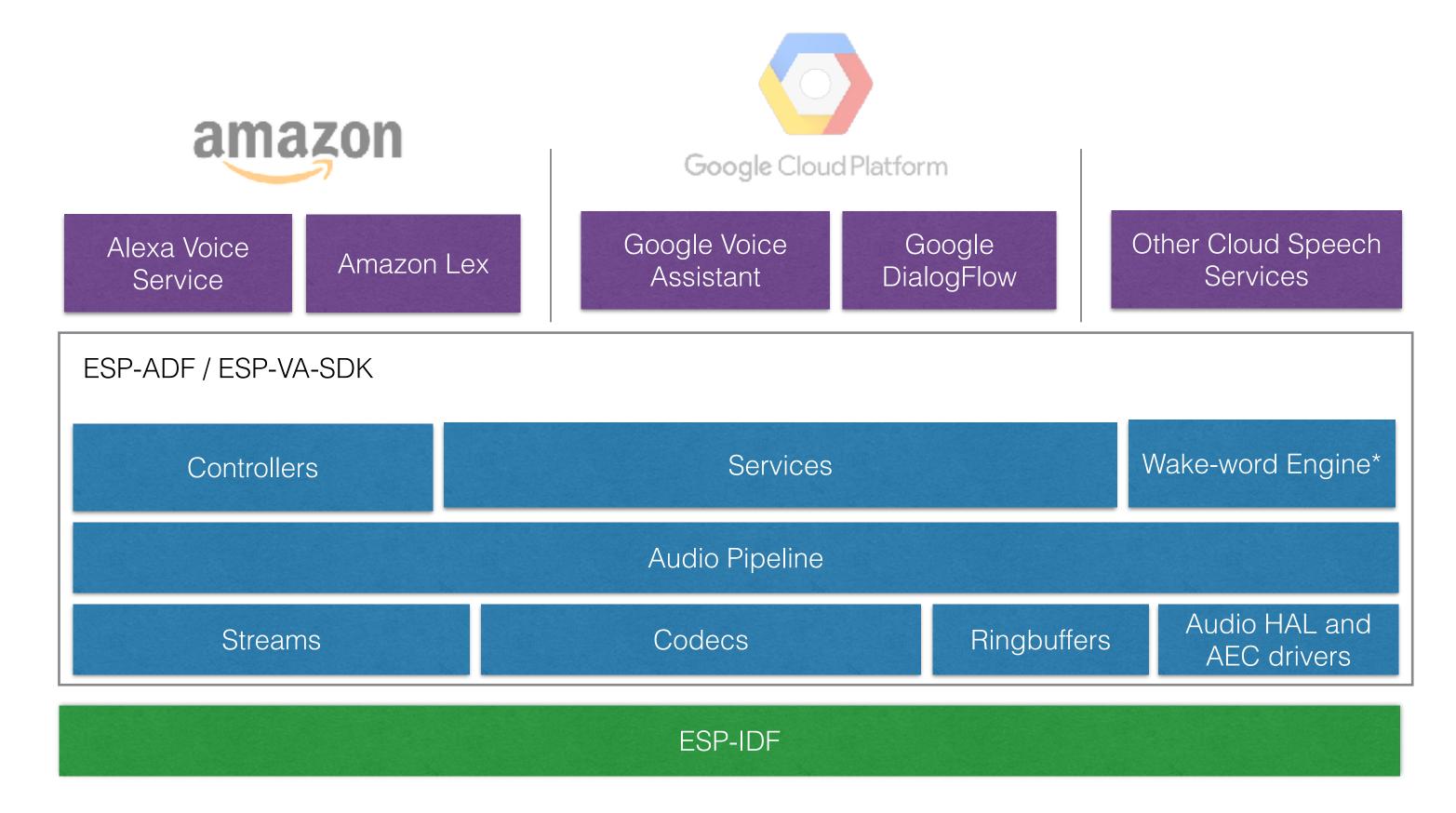


Audio Frameworks

- ESP-ADF Generic Audio SDK (https://github.com/espressif/esp-adf)
- ESP-VA-SDK Voice Assistants SDK (https://github.com/espressif/esp-va-sdk)



Software Architecture

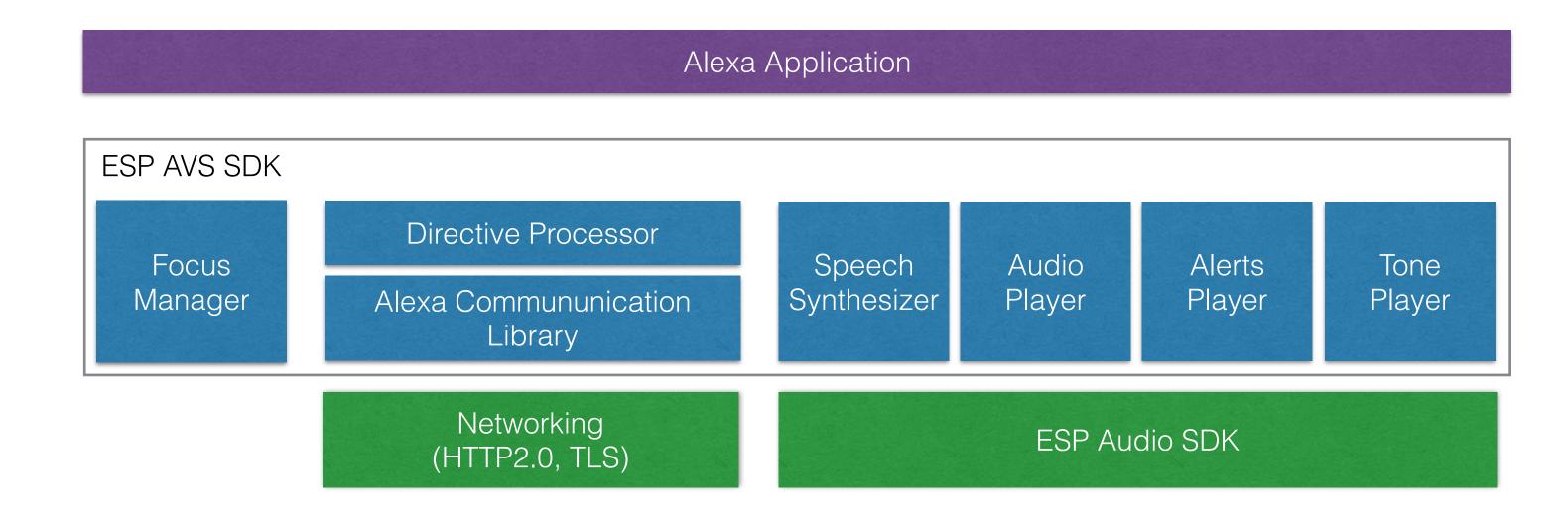


- Based on open-source ESP-IDF
- Audio Pipelining Support
 - Loosely equivalent to GStreamer in Linux
 - Memory optimised implementation
- Codecs and Containers
 - Commonly used codecs, containers and playlists are well supported
 - MP3, AAC, Opus, Vorbis, AMR
 - M4A, TS, Ogg
 - M3U8, PLS
- Streams
 - I2S, HTTP, HLS, file-system streams to source or sink audio
- Services
 - Airplay, DLNA, Bluetooth A2DP Sink, Tone player
- Abstraction layer to easily support newer DSPs and hardware codecs

^{*} Espressif Wake-word Engine is still work in progress and is not used in the solutions as of now



ESP AVS and AFI SDK

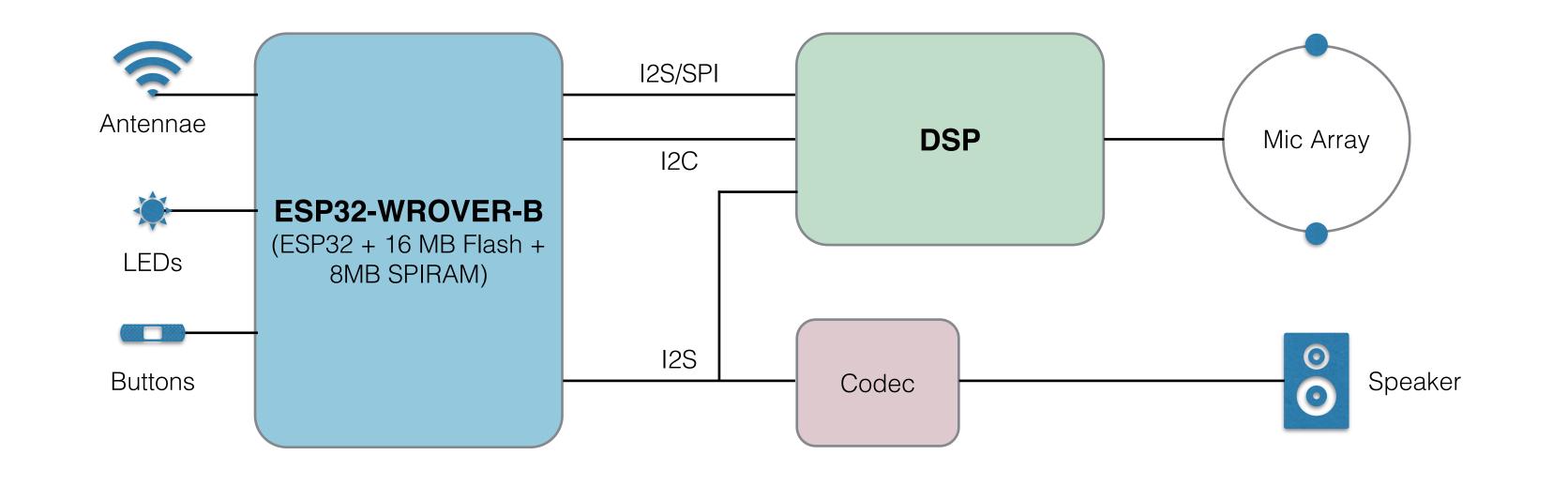


- C based implementation
- Memory and performance optimised
- Based on latest AVS protocol specifications
- Supports all standard AVS features such as
 - Conversations/TTS
 - Music Services
 - Alerts and reminders
 - Multi-turn
- Self-certification tests pass
 - Functional
 - Music
 - Security

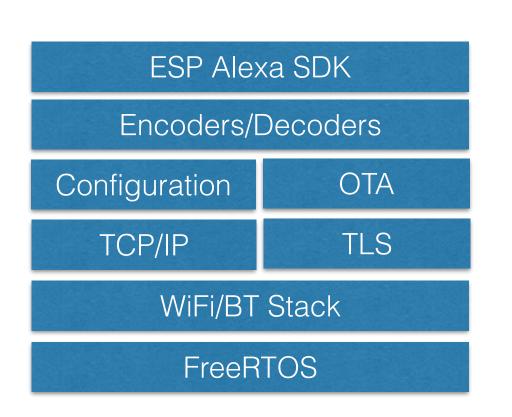


Typical Architecture

Hardware

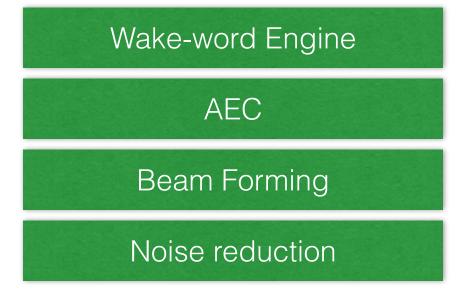






ESP32

DSP





- Smart-speakers and headphones
- Alarm clocks and gadgets
- IoT devices ranging from Smart-plugs to big appliances
- Enterprise devices



Amazon Lex / Google Dialogflow

- Custom voice chatbot services
- Pay-as-you-go pricing
- No certification requirement
- Custom wake-word possible
- Customers get to control the interaction and data



ESP-Who

https://github.com/espressif/esp-who



- Face detection and recognition SDK
- Complete offline training and processing; no cloud involved
- Face detection at 6fps and recognition at 3fps rate
- Design requires PSRAM availability



ESP-Skainet

https://github.com/espressif/esp-skainet



ESP-Skainet Overview

