ARTIK Gateway Module

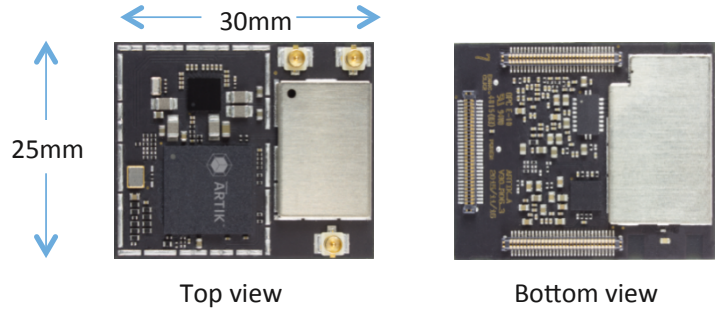
Agenda

- ARTIK Gateway Module Overview
- ARTIK Gateway Module Software Stack
- ARTIK Gateway Module Security
- ARTIK End-to-end Solution
- ARTIK Gateway Module Development
- ARTIK Use case and Ecosystem

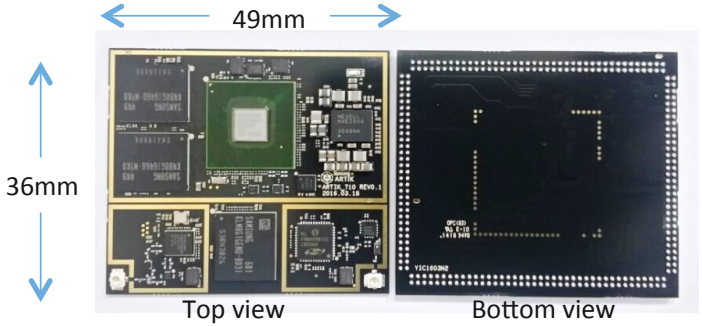
ARTIK Gateway Module Overview

ARTIK High-end module

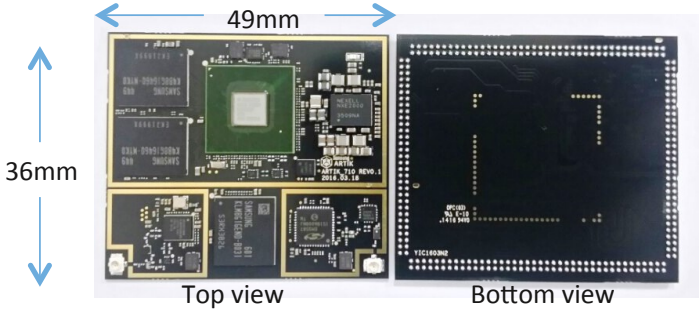
520

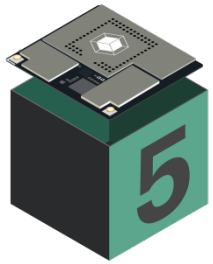


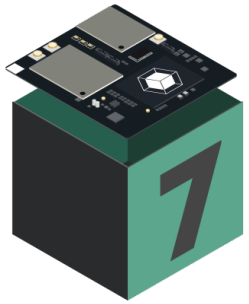
530



710







SAMSUNG ARTIK™

Samsung ARTIK™ 530/530s (512 MB, 1 GB) mid-range gateway

Secure, fully-integrated IoT solution

- Industrial and home gateways
- Voice-controlled speakers
- Building zone controllers
- Display-based healthcare monitors

SAMSUNG ARTIK™

Processor	CPU: 4x ARM® Cortex® A9 @ 1.2 GHz GPU: 3D graphics accelerator
Memory	DRAM: 512 MB/1 GB DDR3 Flash: 4 GB eMMC v4.5
Multimedia	Camera I/F: 4-lane MIPI CSI up to 5MP Display: 4-lane MIPI DSI, HDMI 1.4 a or LVDS (1280 x 720 @ 60 fps) Audio: 2x I2S audio input/output
Connectivity	WLAN (Wi-Fi): IEEE 802.11 b/g/n single-band SISO Bluetooth: 4.2+ Smart 802.15.4: Zigbee, Thread Ethernet: 10/100/1000 Base-T MAC (external PHY required)
Security	Secure element, EAL Level 5, unique device certificate and keys, PKI with mutual authentication to cloud, hardware crypto engine; secure boot*, KMS*, TEE*, *S-modules
I/O	GPIO, UART, I2C, SPI, USB Host, USB OTG, HSIC, ADC, PWM, I2S, JTAG
Temperature range	-25° to 85° (°C)
Size	36 mm W x 49 mm H x 3.4 mm D

Samsung ARTIK™ 710/710s high-end gateway

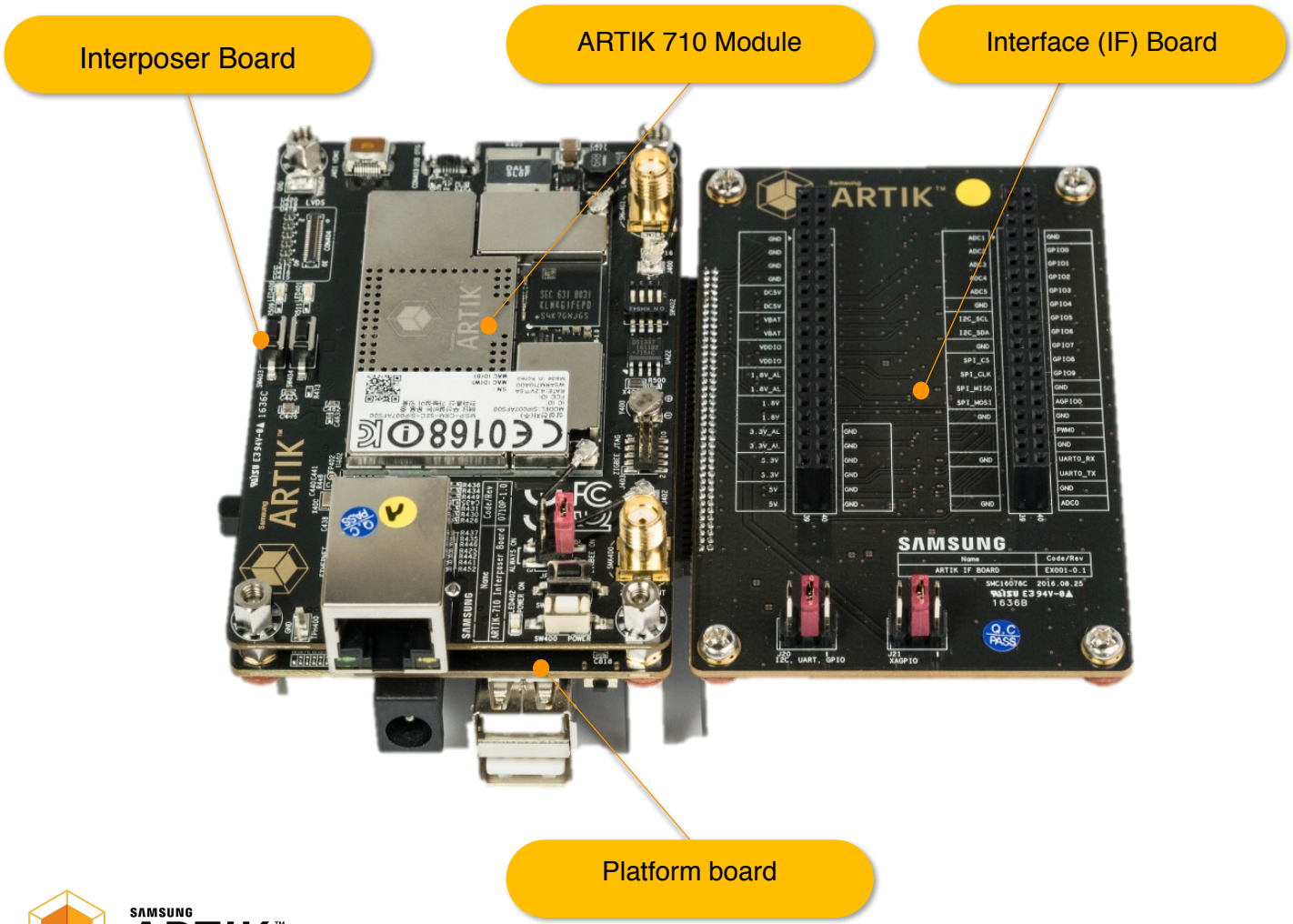
Secure, fully-integrated IoT solution

- High-end gateways
- Cameras
- Human-machine interface
- Machine learning

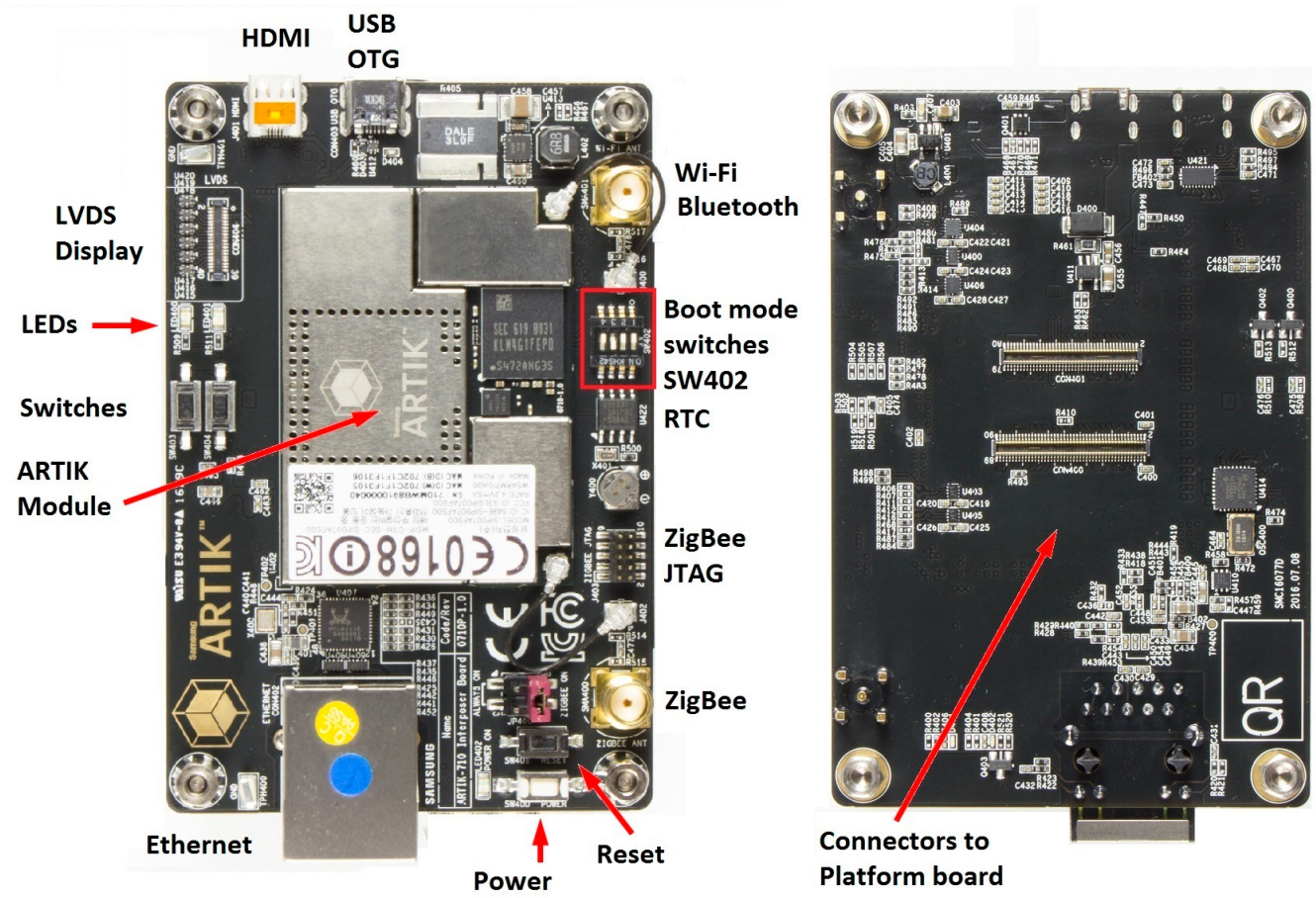
SAMSUNG ARTIK™

Processor	CPU: 8x ARM® Cortex® A53 @ 1.4 GHz GPU: 3D graphics accelerator
Memory	DRAM: 1 GB DDR3 @ 800 MHz Flash: 4 GB eMMC v4.5
Multimedia	Camera I/F: 4-lane MIPI CSI Display: 4-lane MIPI DSI up to FHD@24 bpp, LVDS, HDMI v1.4 Audio: I^2S audio interface
Connectivity	WLAN (Wi-Fi): IEEE 802.11 b/g/n/ac Bluetooth: 4.1+ Smart 802.15.4: Zigbee, Thread Ethernet: 10/100/1000 Base-T MAC (external PHY required)
Security	Secure element, EAL Level 5, unique device certificate and keys, PKI with mutual authentication to cloud, hardware crypto engine; secure boot*, KMS*, TEE*, *S-modules
I/O	GPIO, I^2C, I^2S, SPI, UART, PWM, SDIO, USB 2.0, JTAG, analog input
Temperature range	0° to 70° (°C)
Size	36 mm W x 49 mm H x 3.4 mm D

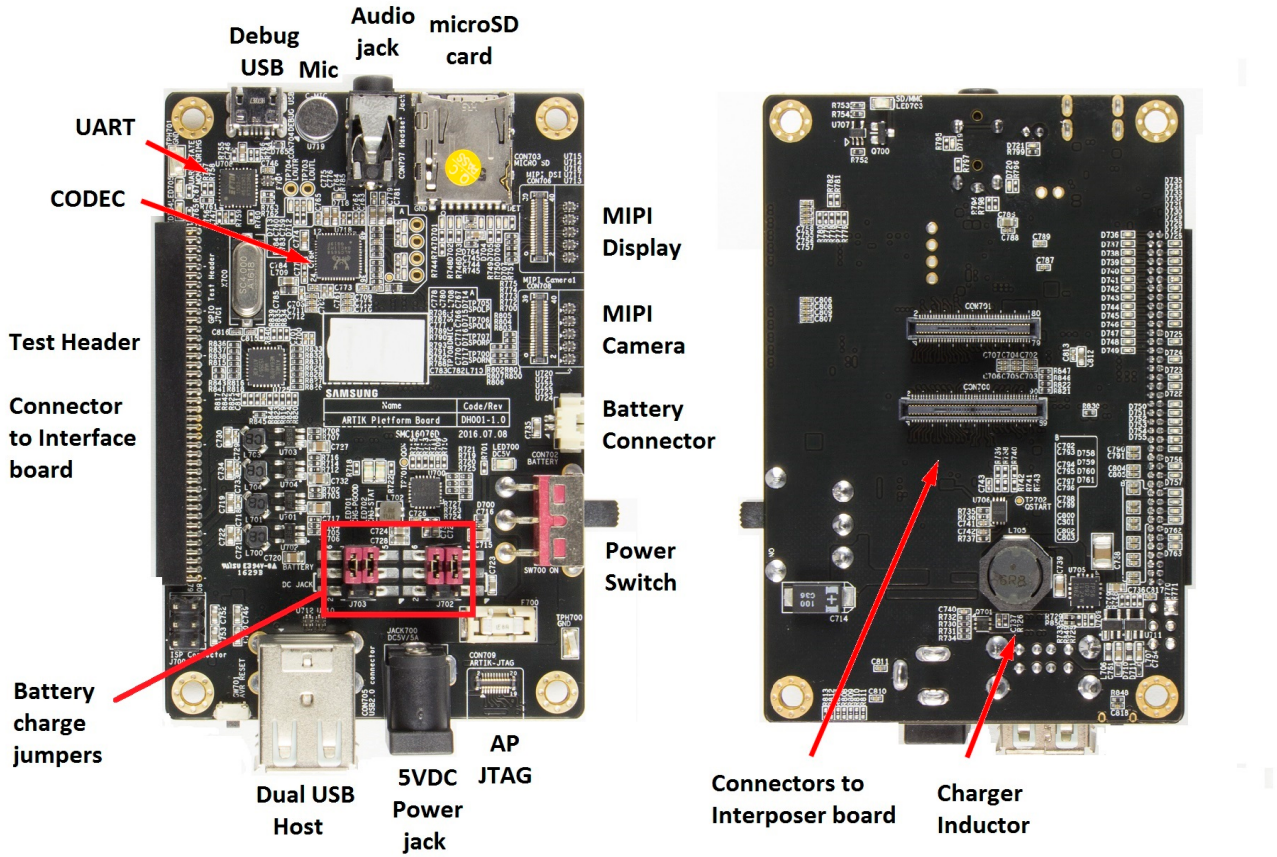
ARTIK High-end module development board



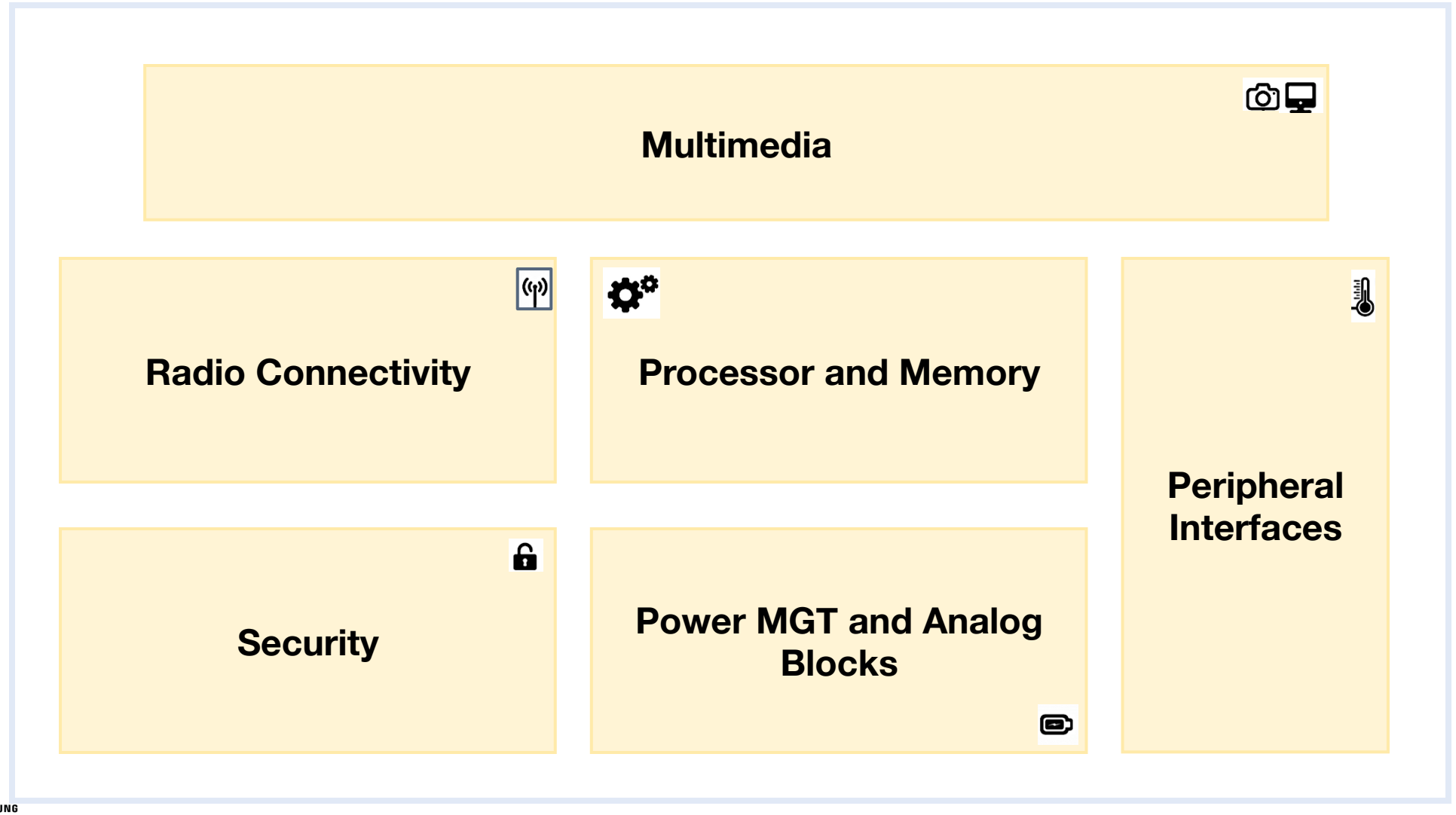
ARTIK High-End Module Interposer Board



ARTIK Gateway Module Platform Board

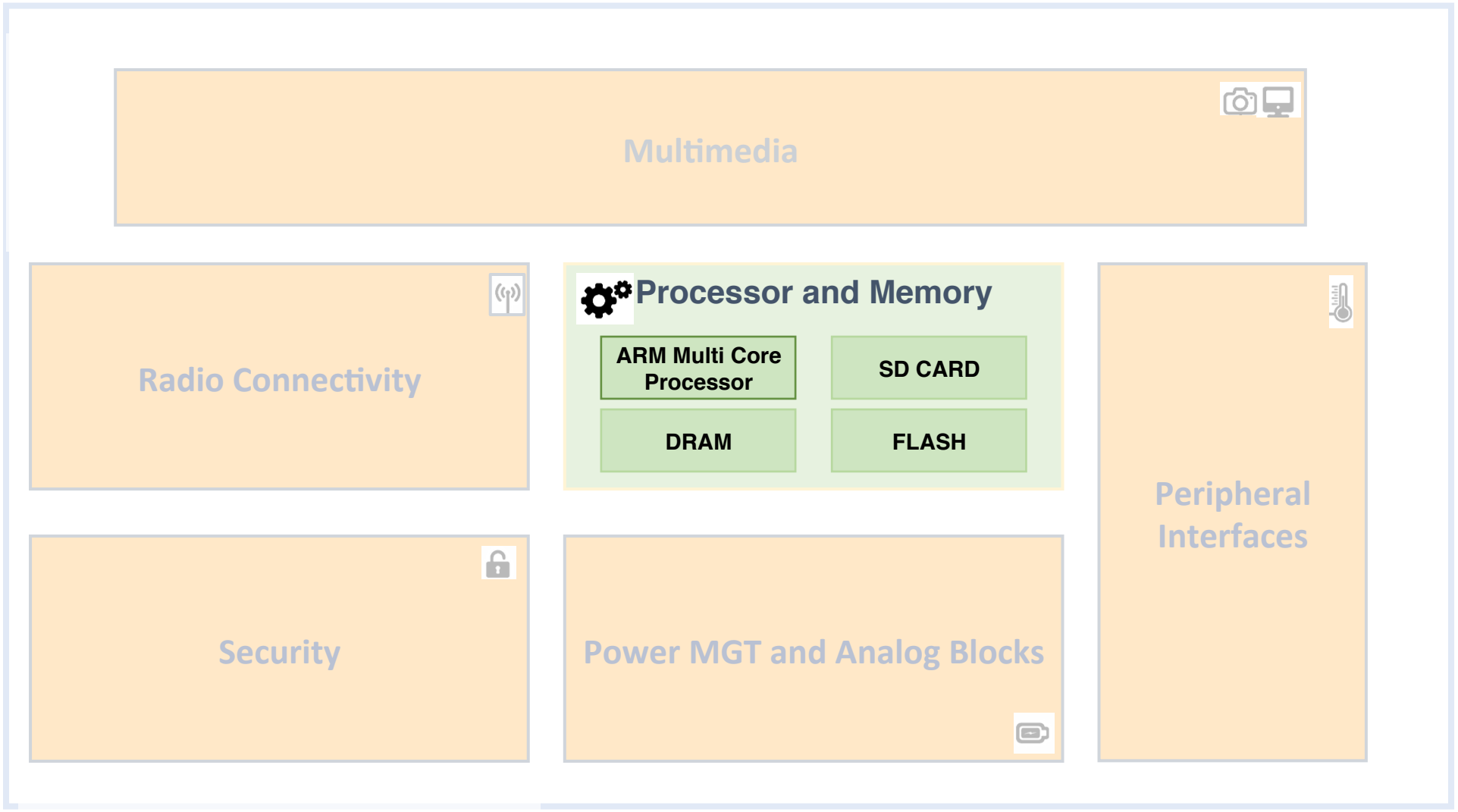


Product Details



SAMSUNG ARTIK™

Product Details – Processor and Memory



SAMSUNG ARTIK™

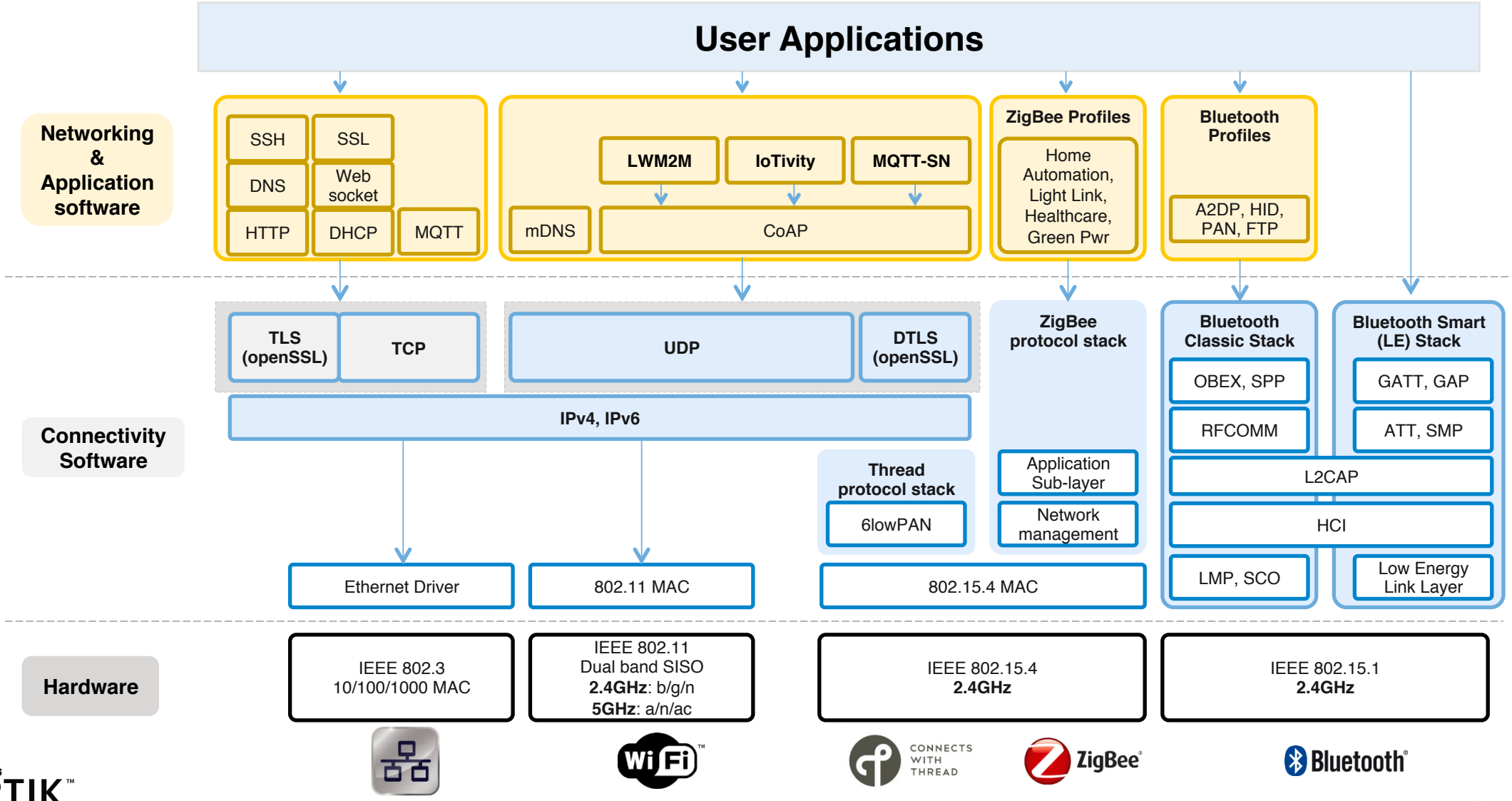
Radio Connectivity

Radio	Range	Data Rate	520	530	710
BLE	1-10m	<1Mbps	✓	✓	✓
BT	1-10m	1-3Mbps	✓	✓	✓
ZigBee	10-50m	10-100Kbps	✓	✓	✓
Thread	10-50m	10-100Kbps	✓	✓	✓
Wi-Fi	10-100m	10-100Mbps	✓	✓	✓
Ethernet			✓	✓	✓

*Z-wave and Sigfox chip set is on 520 development boards

SAMSUNG ARTIK™

Network Protocols



SAMSUNG ARTIK™

Peripheral Interfaces + Power MGT & Analog blocks

		520	530	710
Peripheral Interfaces	I2C	6	3	3
	SPI	2	2	2
	GPIO	100	107	108
	UART	2	3	3
	USB	USB 2.0*	USB 2.0	USB 2.0
Analog and Power MGT	ADC	2	6	6
	PWM	2	2	2
	PMIC	✓	✓	✓

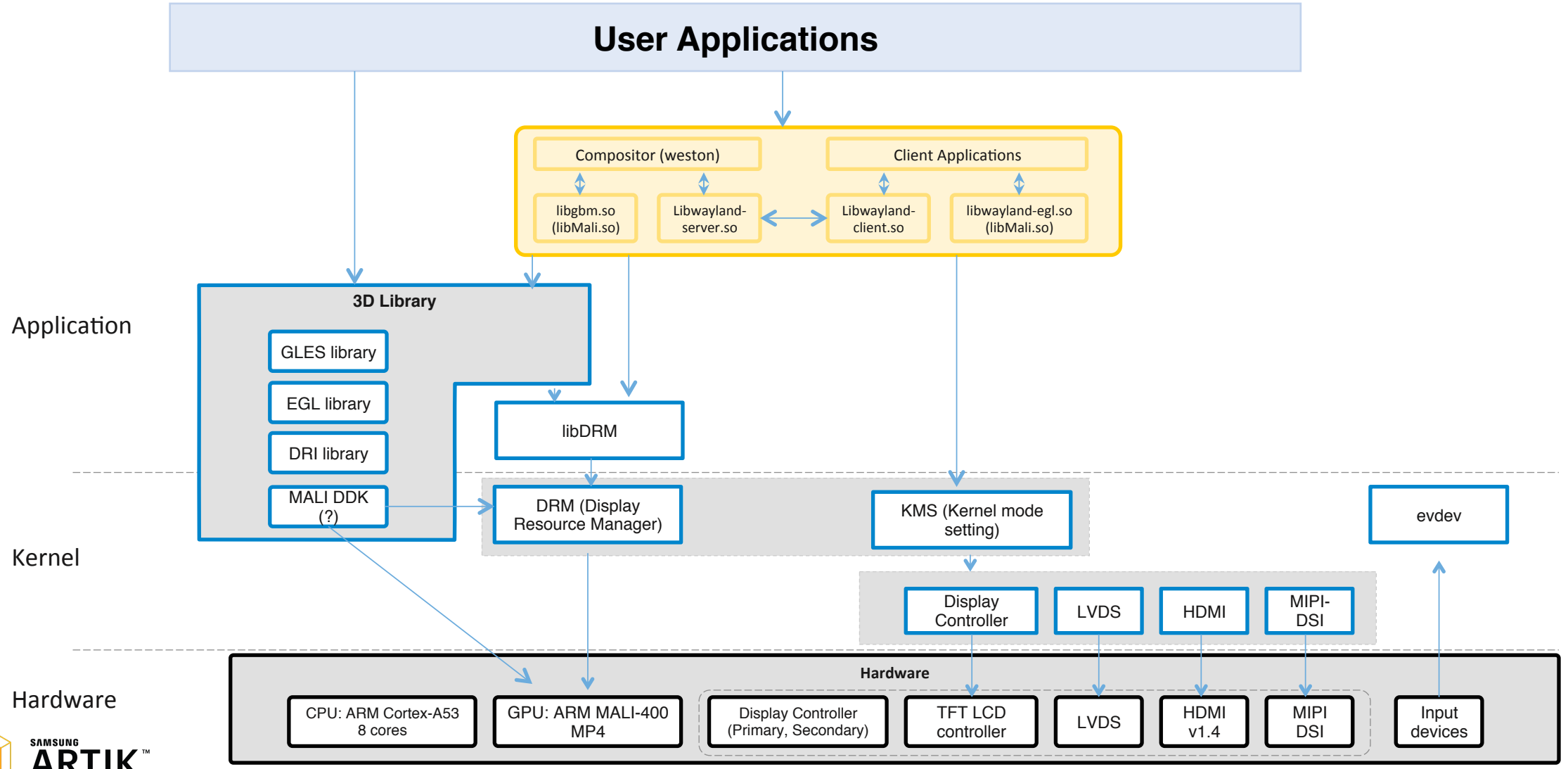
*USB device mode only for 520, rest of the module is both device and host mode.

SAMSUNG ARTIK™

Product Details - Multimedia

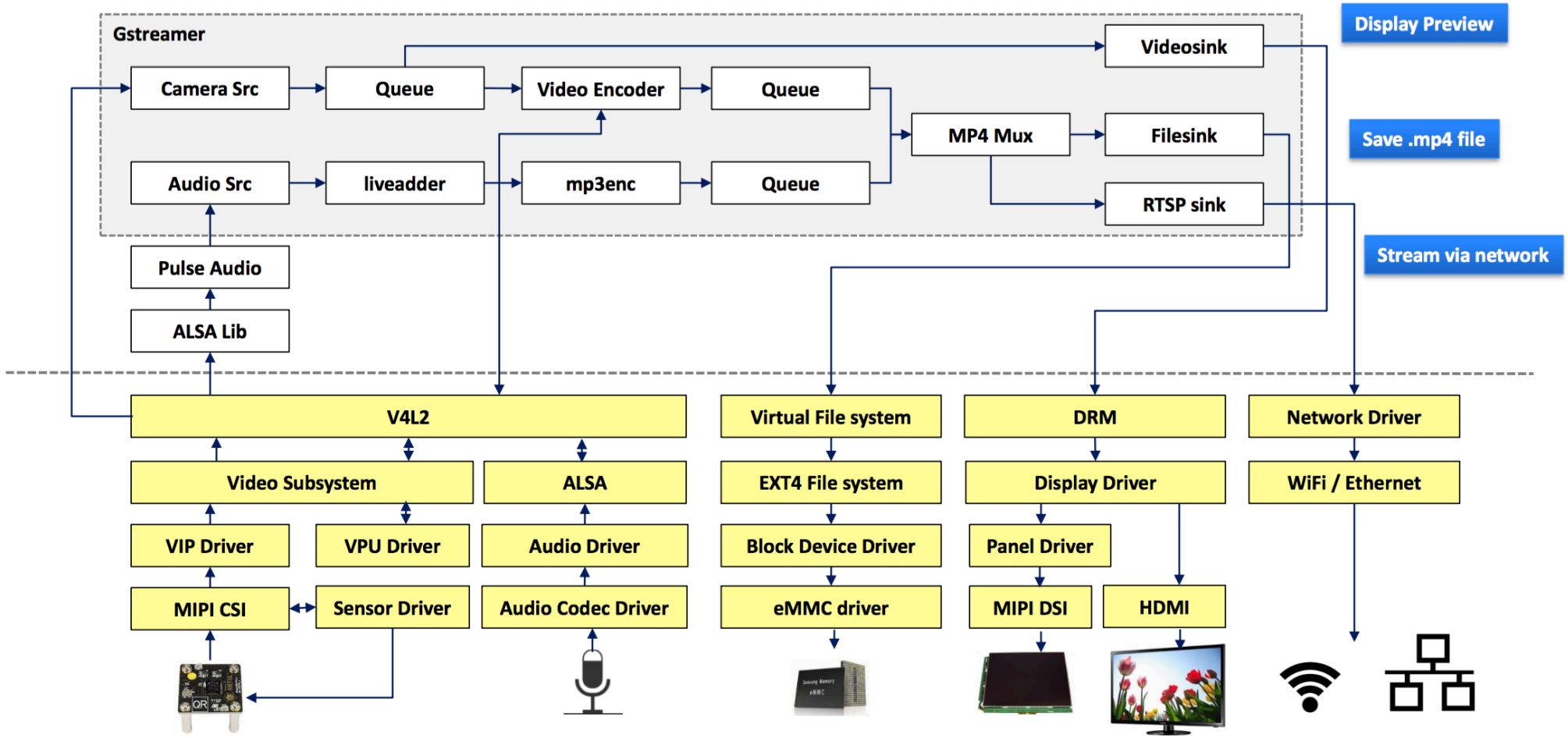
	520	530	710
I2S	1x	2x	2x
HDMI	n/a	1080p @ 60fps	1080p @ 60fps
MIPI – DSI	2-lane 540p @ 24bpp	4-lane 1080p @ 60fps	4-lane 1200p @ 24bpp
MIPI – CSI	2-lane 3MP @ 30fps	4-lane 1080p @ 30fps	4-lane 1080p @ 30fps
LVDS	n/a	720p @ 60fps	720p @ 60fps

Graphics



SAMSUNG ARTIK™

ARTIK Multimedia Framework



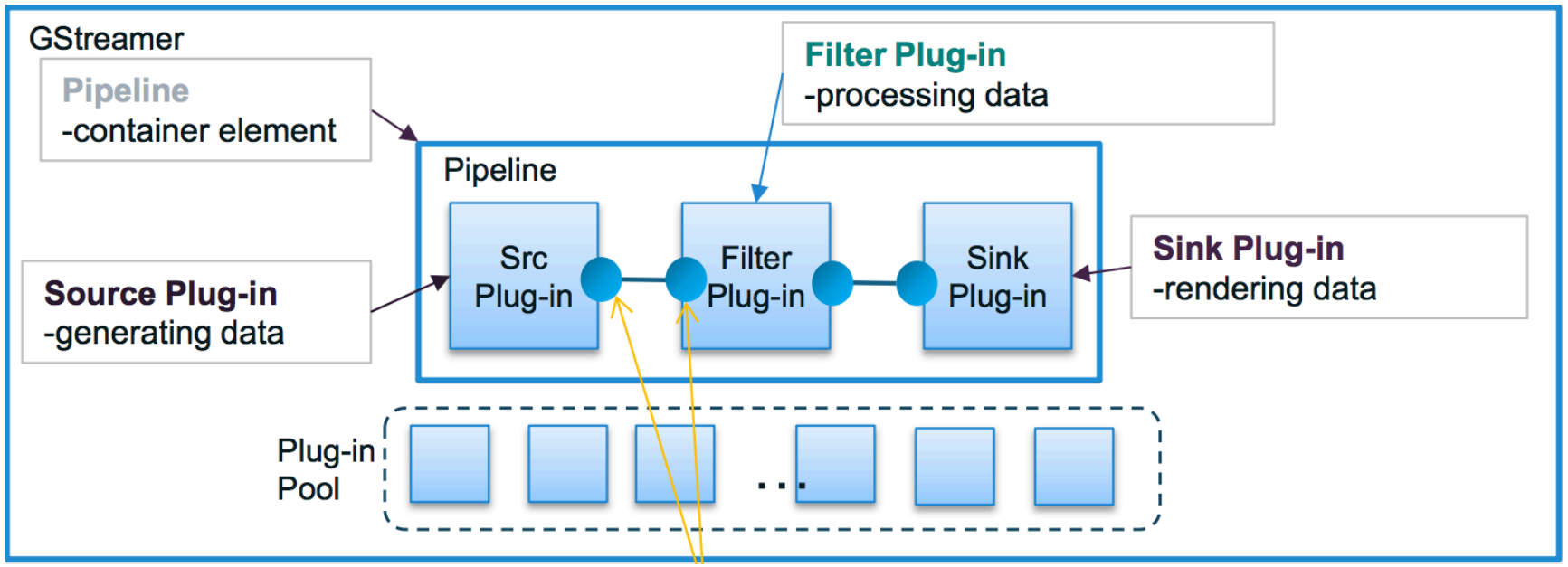
V4L2

Video for Linux Version 2 (V4L2)

- A collection of device drivers and API for supporting real-time video capture on Linux systems
- Supports many USB webcams, TV tuners, and related devices, standardizing their output

GStreamer

- Pipeline-based multimedia framework that links together a wide variety of media processing system to complete complex workflows (https://gstreamer.freedesktop.org/)

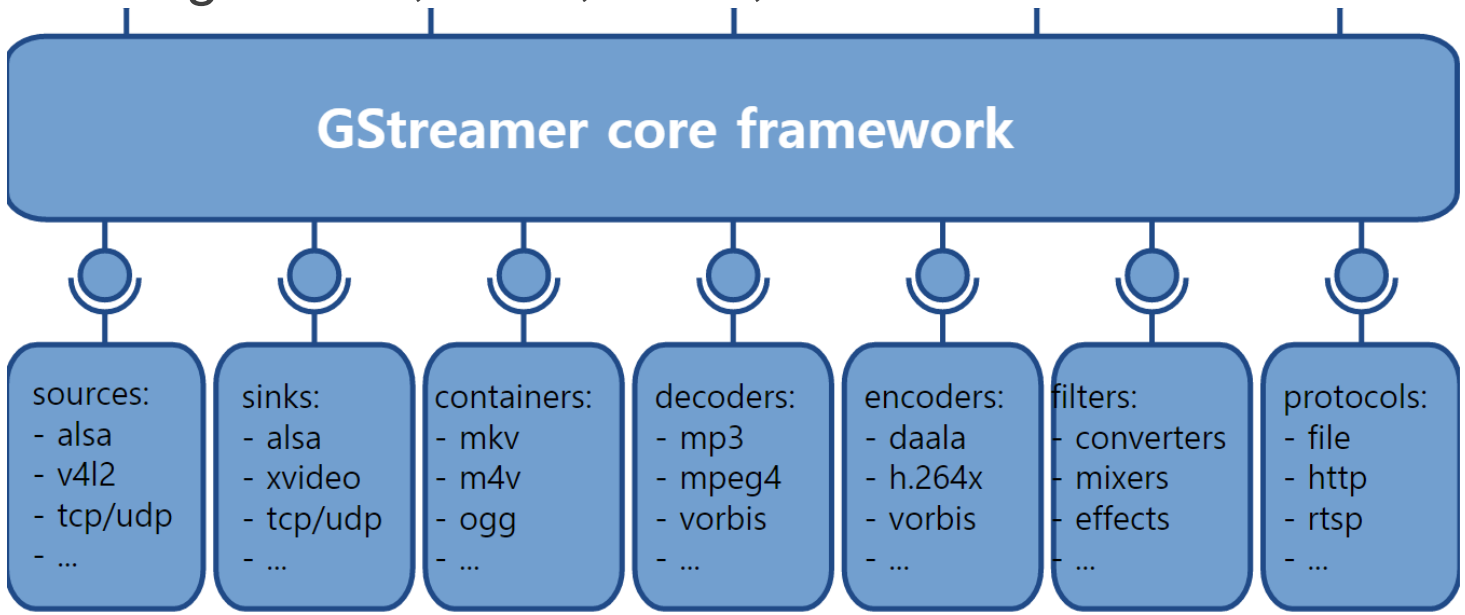


GStreamer

- Supports a wide variety of media handling components
  - Simple audio playback
  - Audio and video playback
  - Recording
  - Streaming and editing
- Serves as a base to create many types of multimedia application such as video editors, transcoders, streaming media broadcaster, media player

GStreamer Plugins

- GStreamer uses a plug-in architecture which makes the most of Gstreamer's functionality implemented as shared libraries.
- Various types including source, sinks, filters, etc.



GStreamer plugins

GStreamer includes over 150 plugins

software plugins are executed on the host CPU

hw accelerated plugins shunt execution to acceleration DSPs and the result back

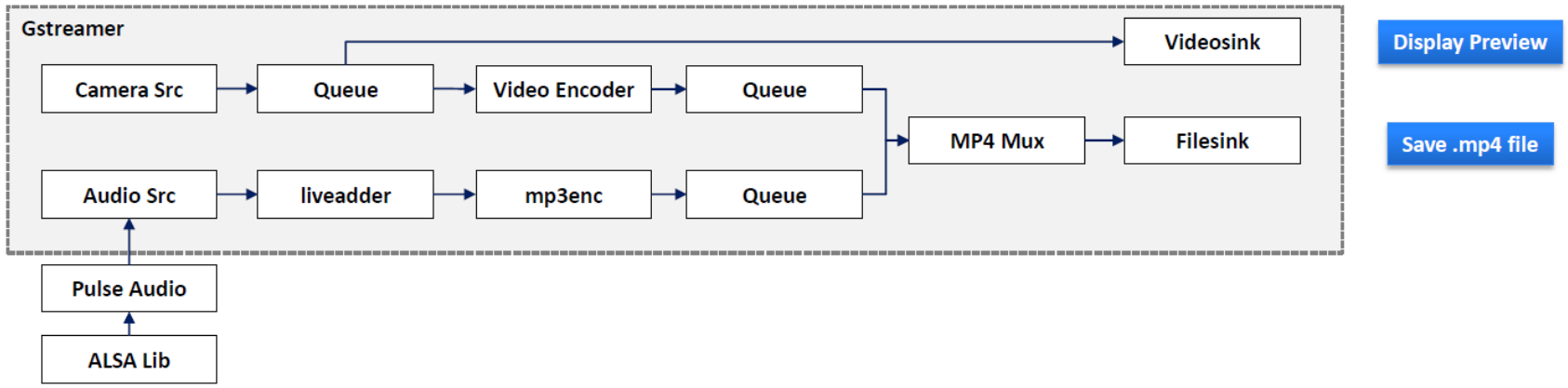
3rd party plugins

Texas Instuments, Motorola, et al.

SAMSUNG ARTIK™

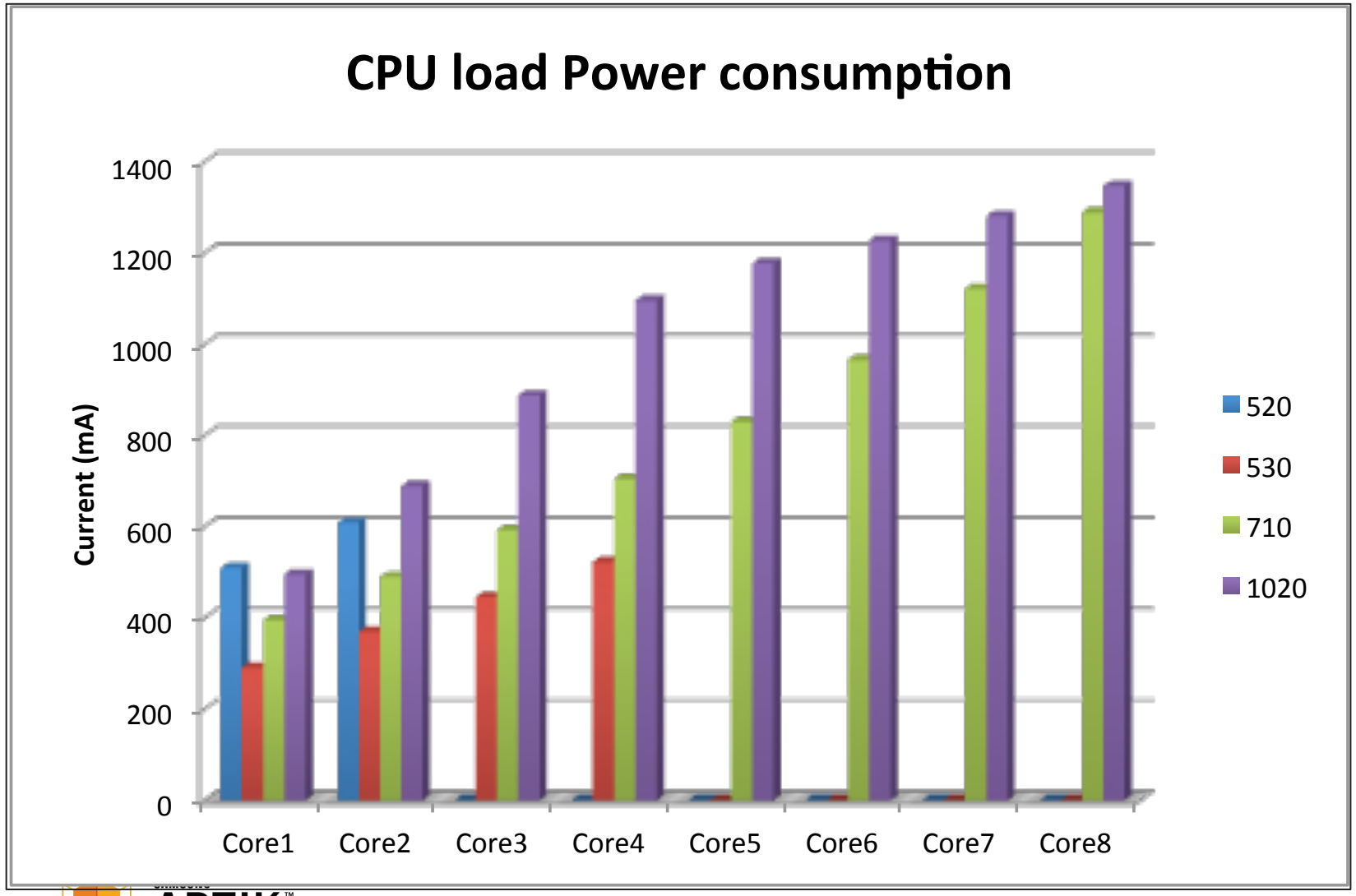
GStreamer Pipeline Example for ARTIK710/530

- Record a HD MP4 movie with sound and simultaneous preview on display



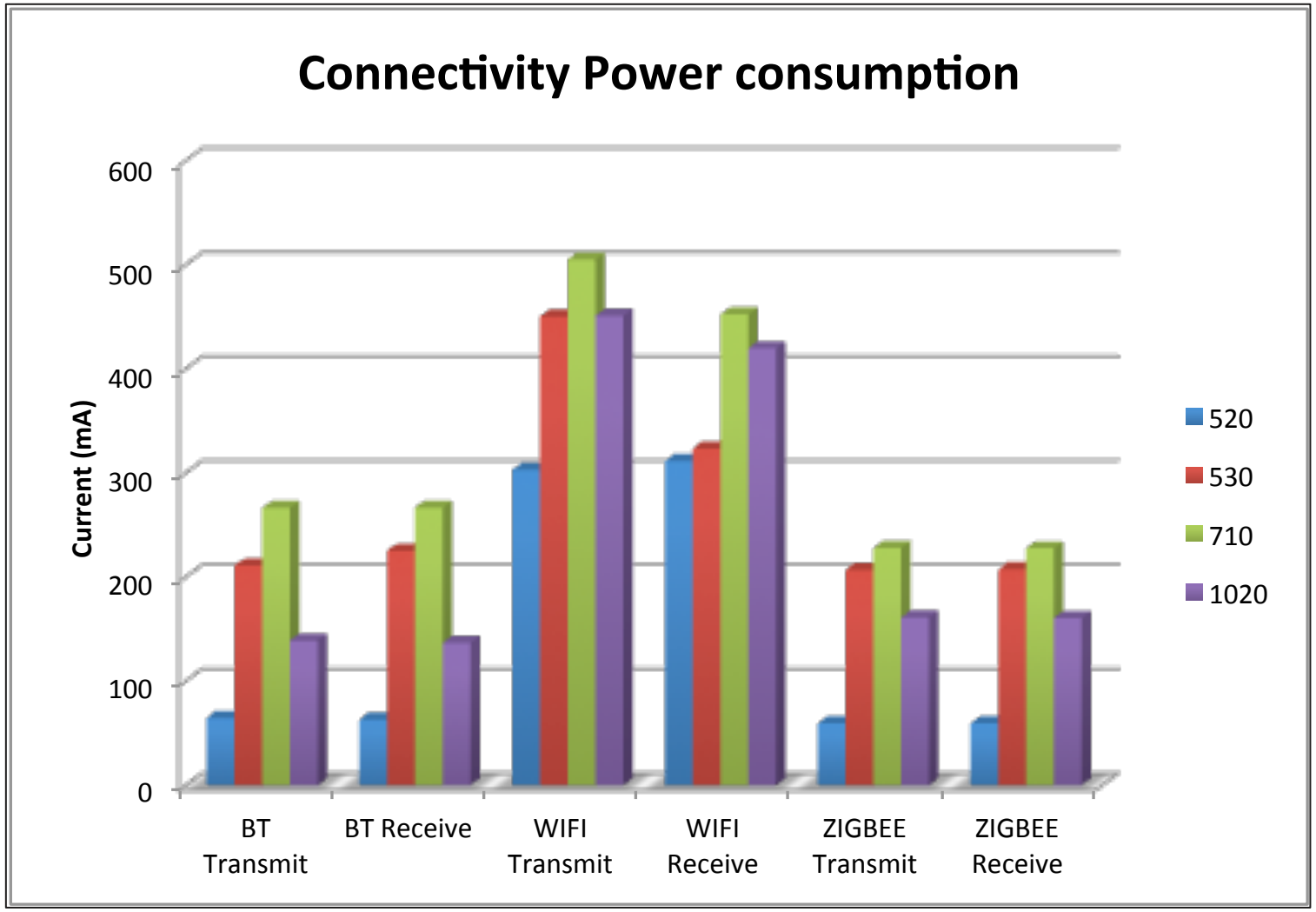

```
gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! tee name=t \
t. ! queue ! nxvideoenc bitrate=12000000 ! queue ! mux. \
autoaudiosrc ! liveadder start-time-selection=2 start-time=600000000! lamemp3enc ! queue ! mux. \
t. ! nxvideosink \
mp4mux name=mux ! filesink location=result_a.mp4
```

Module Power Consumption – CPU Load



ARTIK™

Module Power Consumption - Connectivity

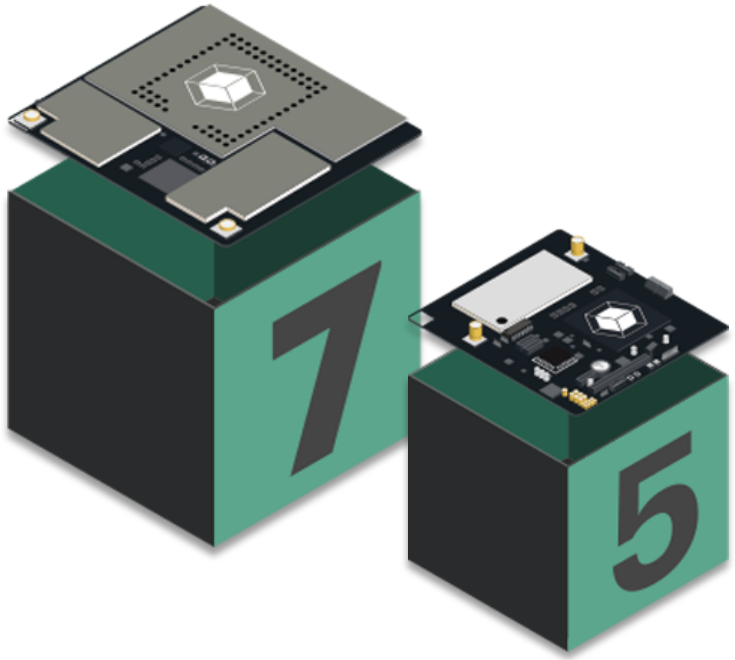


- Power measurements include the processor as well, i.e. not just radios
- BT Tx/Rx
  - Transfer/receive test file using Obex FTP between ARTIK and Android phone
- WI-FI Tx/Rx
  - Transfer/receive packet using iperf3 (802.11ac)
- ZigBee
  - Transfer/receive packet using Ember tool

SAMSUNG ARTIK™

ARTIK Gateway Module Software Stack

ARTIK Gateway Software Stack



Application

ARTIK API

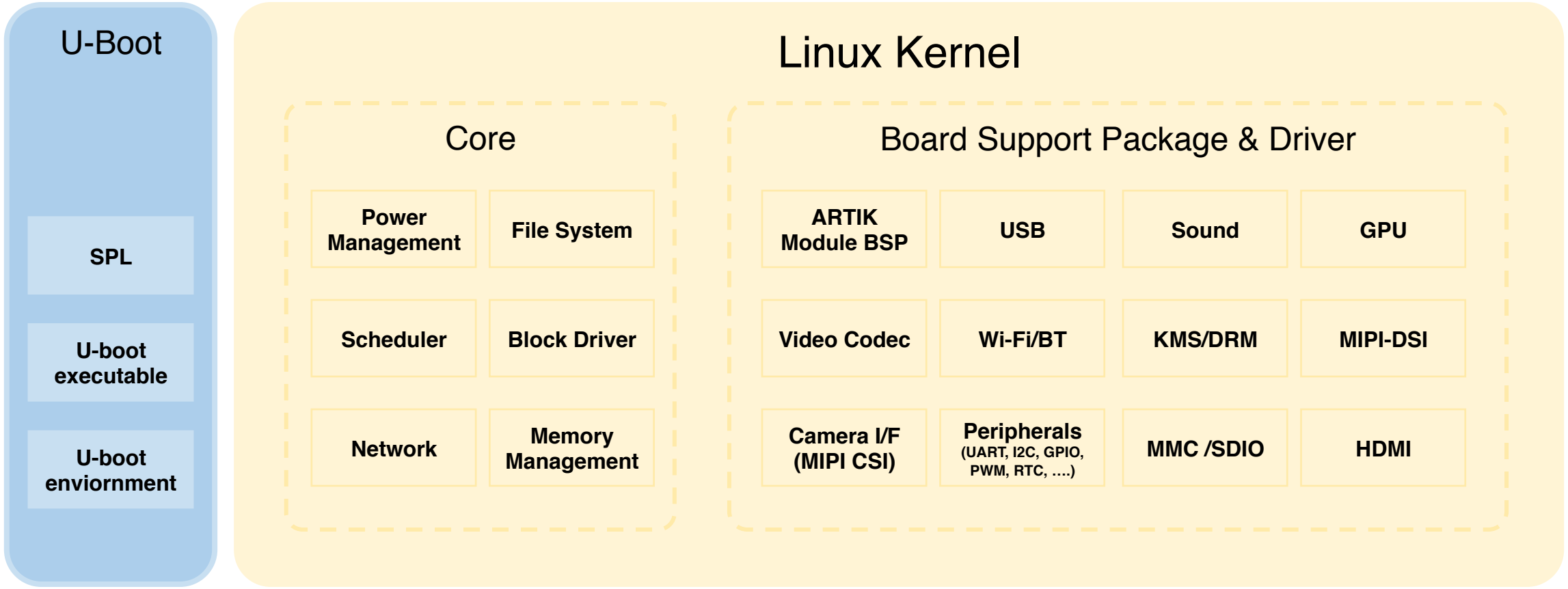
Root FS

Kernel

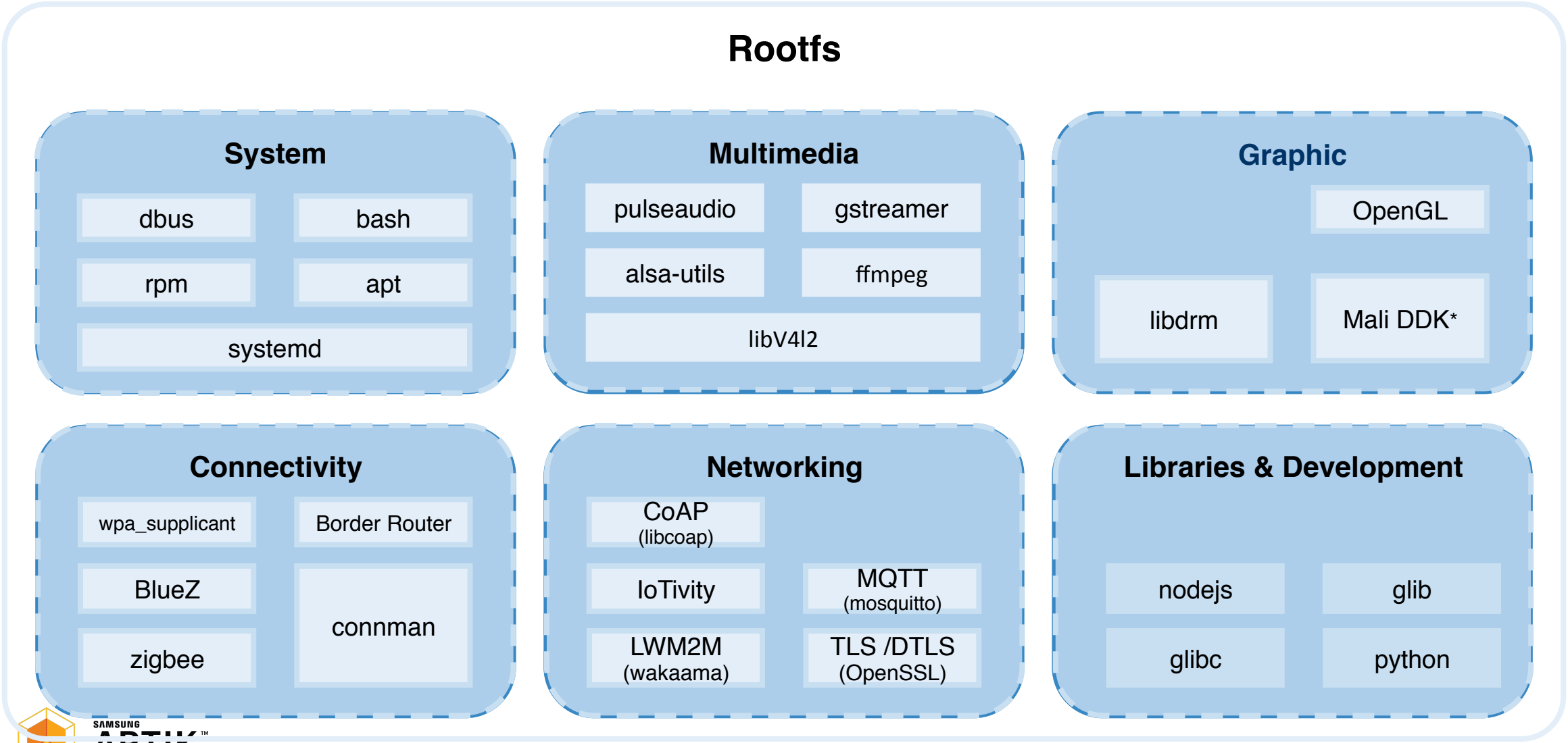
Hardware

SAMSUNG ARTIK™

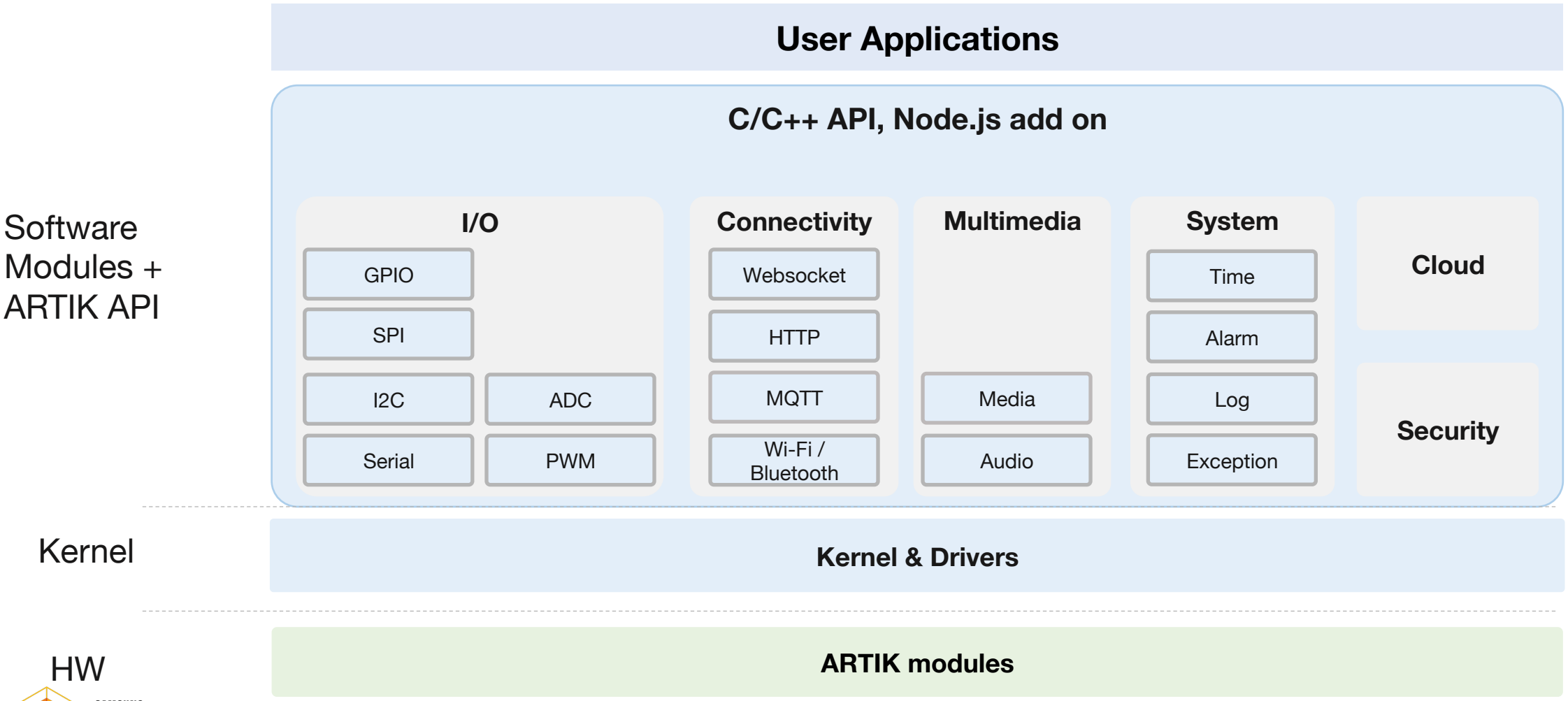
U-Boot and Linux Kernel Architecture



Architecture of Rootfs



ARTIK SDK (5, 7 series)



SAMSUNG ARTIK™

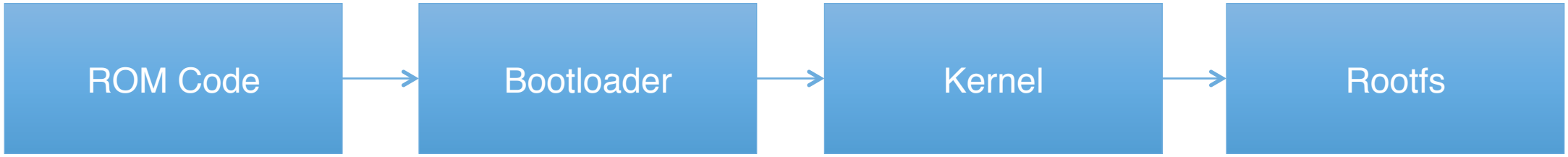
ARTIK Gateway Module Security

ARTIK Security

		ARTIK module (05x, 5, 7)	ARTIK S-module	Comments
Secure communication	Per device unique key & certificate	✓	✓	Uniquely identifies device
	Key stored in HW secure element	✓	✓	Secure key storage
	PKI infrastructure: Mutual authentication of device and cloud	✓	✓	Device talks to authorized cloud and vice versa
	Post Provisioning		✓	Provision with your own keys and certificates
Device protection/ secure code execution	KMS infrastructure for code signing		✓	Key Management Service
	Code verification key in HW		✓	Secure key storage
	Secure boot (check for authorized code)		✓	Boot image verification
	JTAG access locked		✓	Lock out debug access
Data protection/ Secure storage	Secure OS (separate normal & secure operations)		✓	Hardware enforced secure applications via TEE
	Security Lib API (27 API calls)	Limited(random number generator, get cert and signature)	✓	Key Manager, Authentication, Secure Storage, Post Provisioning, Encrypt/Decrypt
	Secure storage		✓	Encrypt data stored on Flash

SAMSUNG ARTIK™

Secure Boot - Revisit



ROM Code

- Need a way to store the public keys which will be used to decrypt the signature of the bootloader. The mechanism has to be tamper-proof.
- Basic flow:

1. Loads the bootloader in a secure space to avoid physical attack
2. Loads the embedded public key
3. Checks the hash of the public key against the hash table in OTP
4. Uses this verified public key to check the signature of bootloader
5. Executes the bootloader binary

SAMSUNG ARTIK™

Kernel Authentication

- U-Boot has Device Tree Blob(DTB) support
- DTB can also be used to store a public key, which is used to verify the signature of kernel image
- FIT Image that includes all the images needed to boot a system
- `mkImage` provides built-in support for signing of binaries hash

Key Generation

```
openssl genrsa -out my_key.key 4096

openssl req -batch -new -x509 -key my_key.key -out my_key.crt
```

mkImage requires certificate and private key files have the same name.

DTB creation

```
/dts-v1/;
/ {
    model = "Keys";
    compatible = "vendor,board";
    signature {
        key-my_key {
          required = "image";
          algo = "sha1,rsa4096";
          key-name-hint = "my_key";
        };
    };
};
```

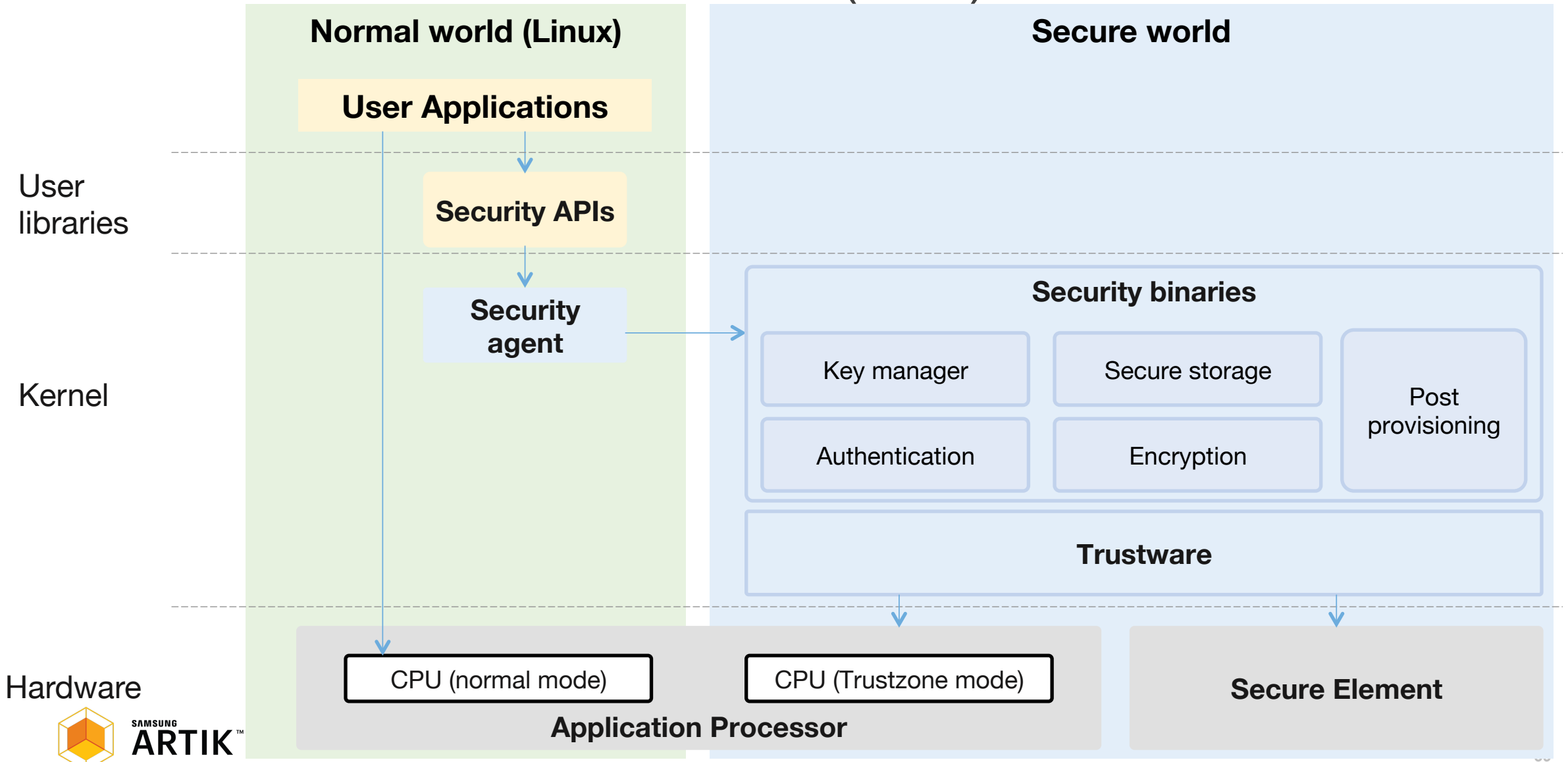
fitImage.its

```
/ {
    description = "fitImage for Foo revA and revB";
    #address-cells = <1>;
    images {
        kernel@1 {
            description = "Linux kernel";
            data = /incbin/("zImage");
            type = "kernel";
            arch = "arm";
            os = "linux";
            compression = "none";
            load = <0x10008000>;
            entry = <0x10008000>;
            signature@1 {
                algo = "sha1,rsa4096";
                key-name-hint = "my_key";
            };
        };
        fdt@1 {
            description = "DTB for Foo revA";
            data = /incbin/("foo-reva.dtb");
            type = "flat_dt";
            arch = "arm";
            compression = "none";
            signature@1 {
                algo = "sha1,rsa4096";
                key-name-hint = "my_key";
            };
        };
        fdt@2 {
            description = "DTB for Foo revB";
            data = /incbin/("foo-revb.dtb");
            type = "flat_dt";
            arch = "arm";
            compression = "none";
            signature@1 {
                algo = "sha1,rsa4096";
                key-name-hint = "my_key";
            };
        };
    };
    configurations {
        default = "conf@1";
        conf@1 {
            kernel = "kernel@1";
            fdt = "fdt@1";
        };
        conf@2 {
            kernel = "kernel@1";
            fdt = "fdt@2";
        };
    };
};
```

U-boot and fitImage Creation

```
#DTB compiled out-of-tree because we need to add the public key with \code{mkimage}
dtc u-boot_pubkey.dts -O dtb -o u-boot_pubkey.dtb
make CROSS_COMPILE=arm-linux-gnueabihf- foo_defconfig
make CROSS_COMPILE=arm-linux-gnueabihf- tools
tools/mkimage -f fitImage.its -K u-boot_pubkey.dtb -k /path/to/keys -r fitImage
make CROSS_COMPILE=arm-linux-gnueabihf- EXT_DTB=u-boot_pubkey.dtb
```

Trusted Execution Environment(TEE)



Secure Storage

- eMMC file system (Flash-based)
  - Uses the same storage as the normal operating system. However, a specific partition is managed by the TrustZone-based Secure OS.
  - All data in this partition is encrypted with a unique key generated at run time, and is stored as a file unit of 32KB with a maximum of 1024 files that may be stored.
- Secure Element – an isolated storage device that supports the storage of up to 16 AES 128-bit keys.
  - The Secure Element provides high levels of security as hardware with anti-tamper measures.
  - All communication from the Secure Element to the processor is secured and encrypted.

Secure Element

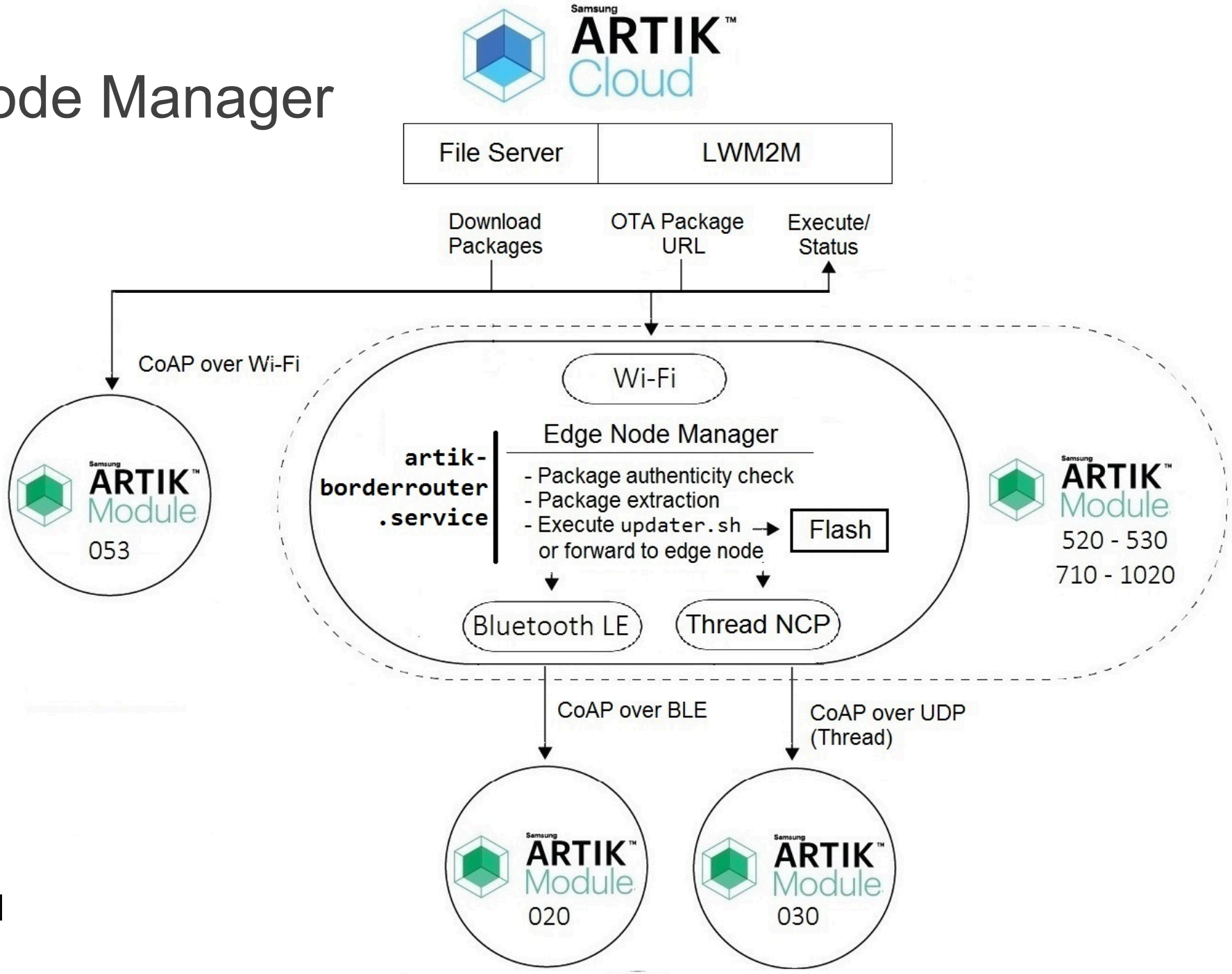
- The Secure Element provides services for the secure storage of cryptographic material and other protected content.
- It includes cryptographic services such as random-number generation, key/data secure storage, and certificates handling and processing.
- Secure Element uses Smart Shield, Smart Sensor, Smart Core technologies to achieve the highest level of security and protection.
- The Secure Element meets the Common Criteria (CC) certification for security and for Evaluation Assurance Level (EAL) 5+.

ARTIK End-to-end solution

ARTIK Gateway Onboarding

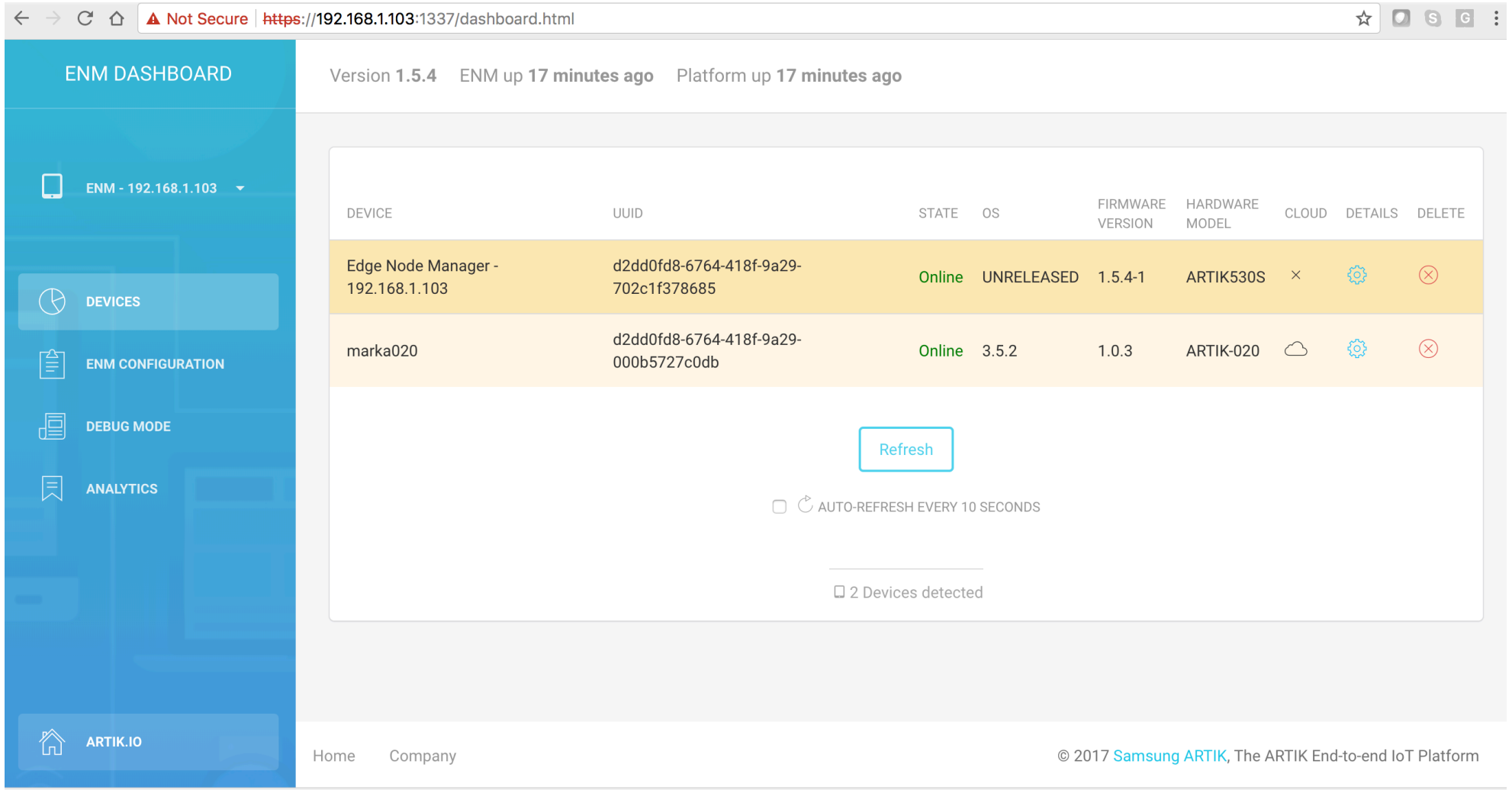
- Support BLE onboarding on Gateway modules
  - QR code contains device information like MAC address, through which the mobile app learns the BLE service UUID of the ARTIK controller.
  - Mobile App scans for BLE devices and matches the service UUID with the one from the QR code.
  - Connection Manager connects ARTIK Gateway to WiFi.
  - Mobile App interacts with BLE on-boarding service through BLE GATT profile.

Edge Node Manager



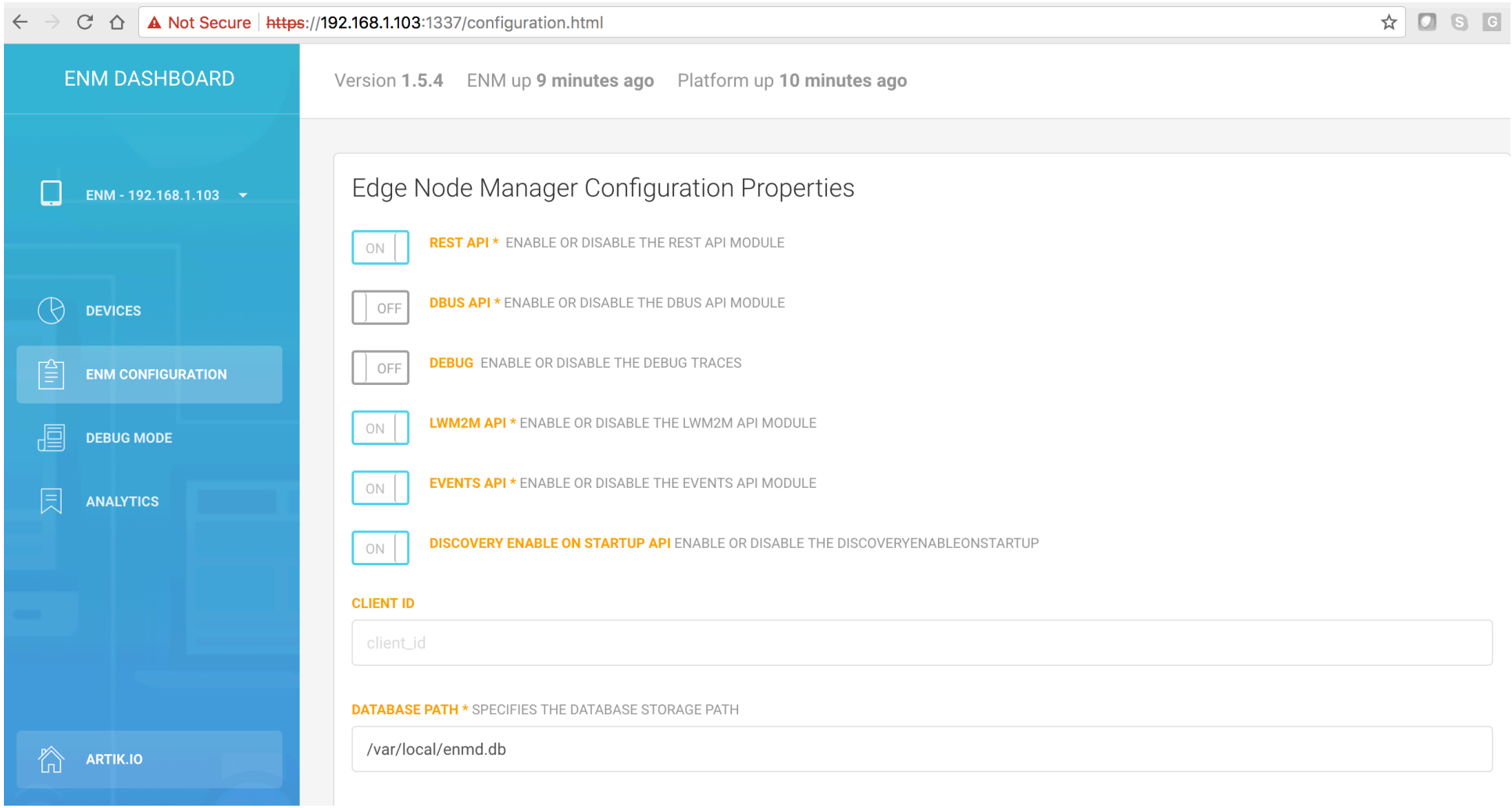
SAMSUNG ARTII

Edge Node Manager Dashboard



SAMSUNG ARTIK™

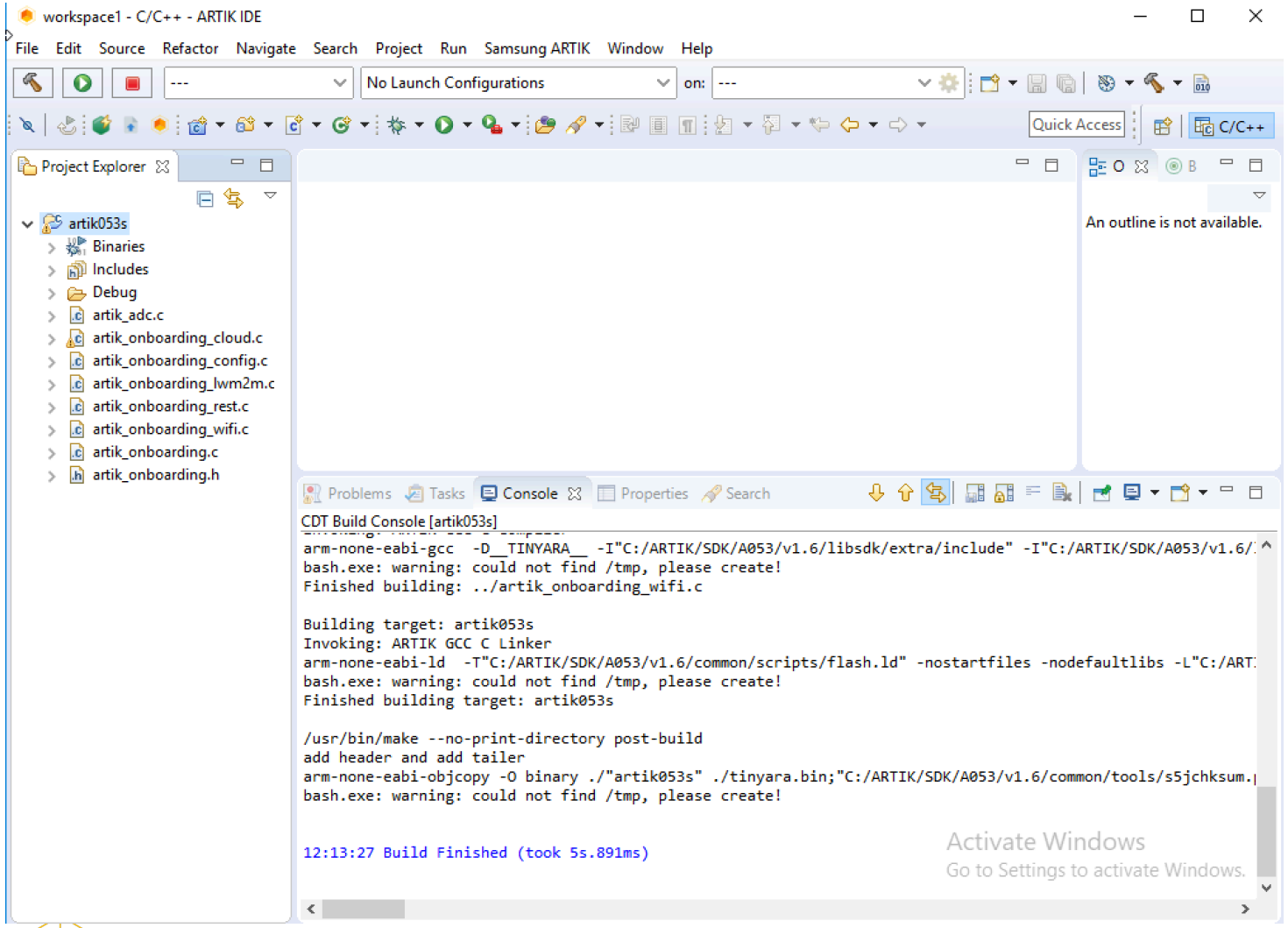
Edge Node Manager Dashboard (Cont.)



SAMSUNG ARTIK™

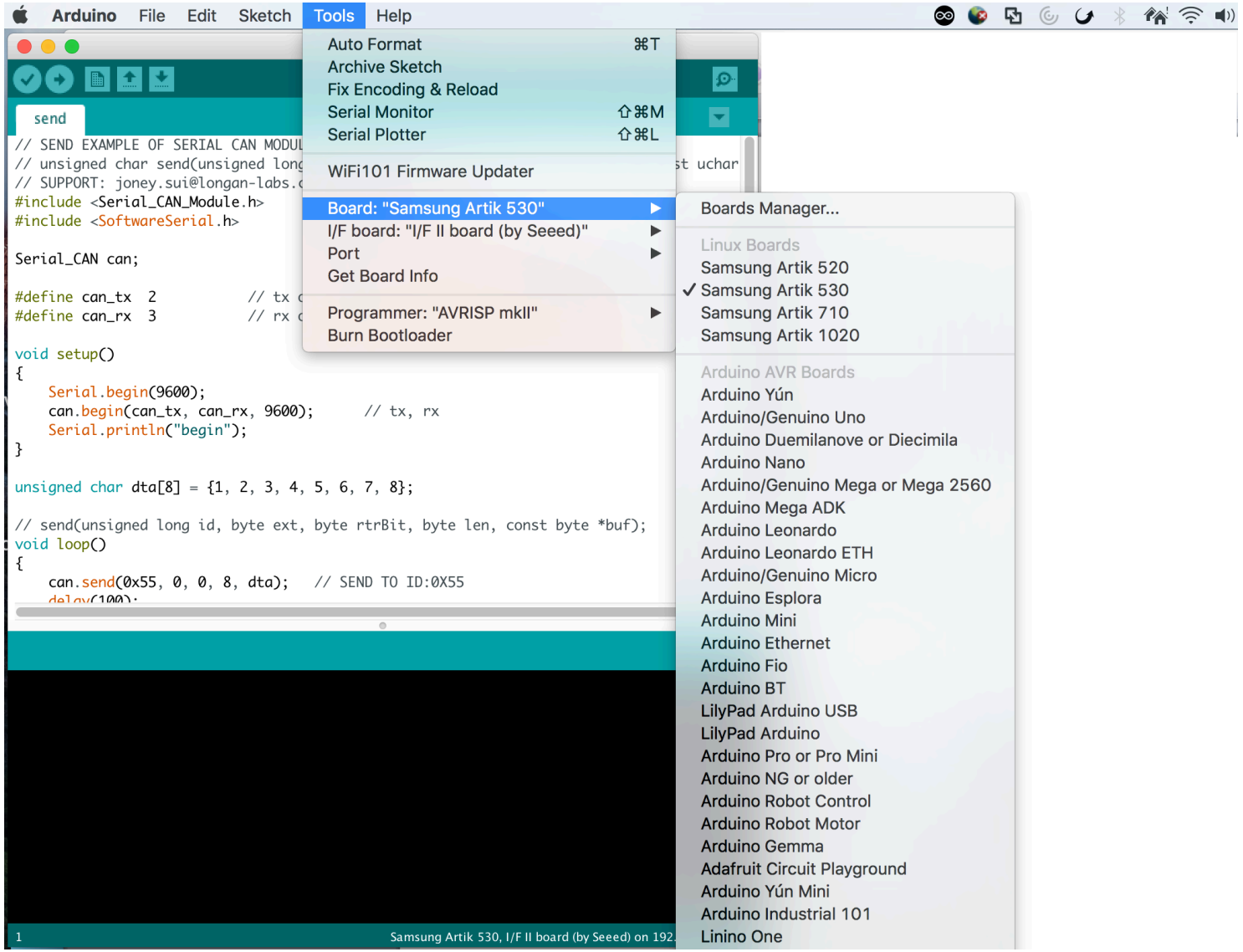
ARTIK Gateway Module Development

ARTIK IDE



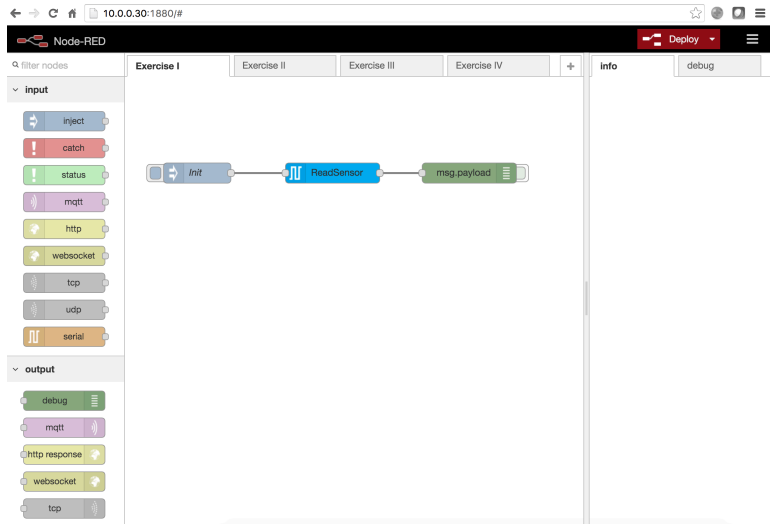
gcc-arm-linux-gnueabihf
and
aarch64-linux-gnu

Arduino



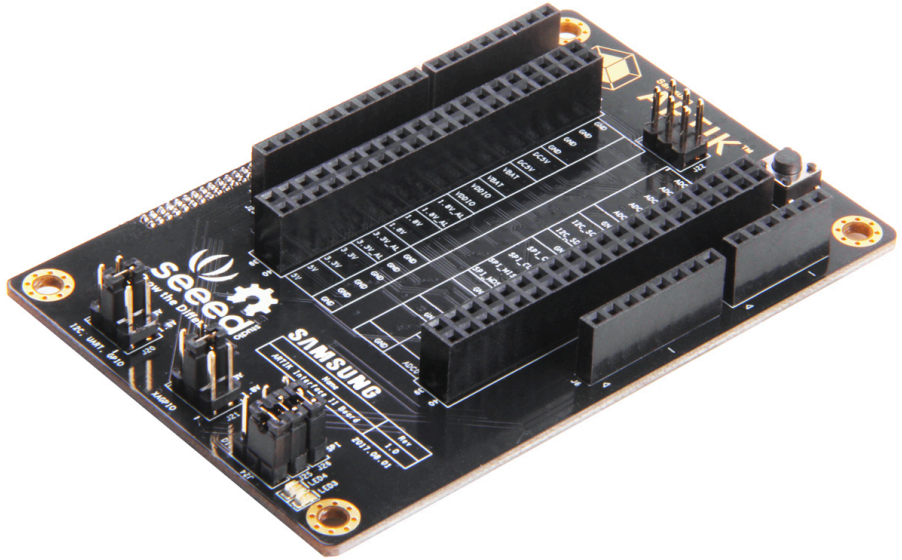
Node-RED

- A visual tool for wiring the internet of things, based on Node.js
- Utilizes flow programming technique
- Construct program flow by drag-and-drop
- You have the option not to write code
- Growing ecosystem
- Cloud-based solutions: IBM Bluemix

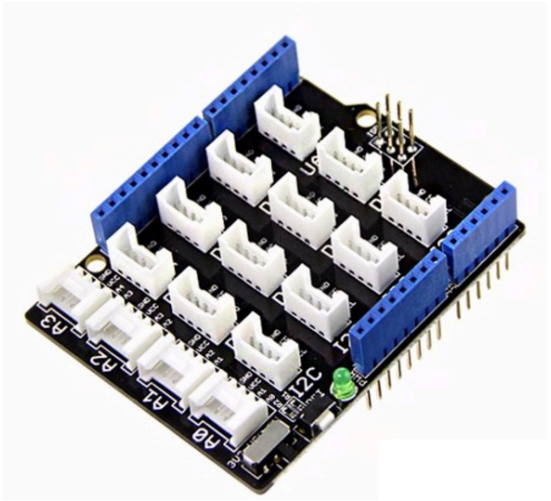


SAMSUNG ARTIK™

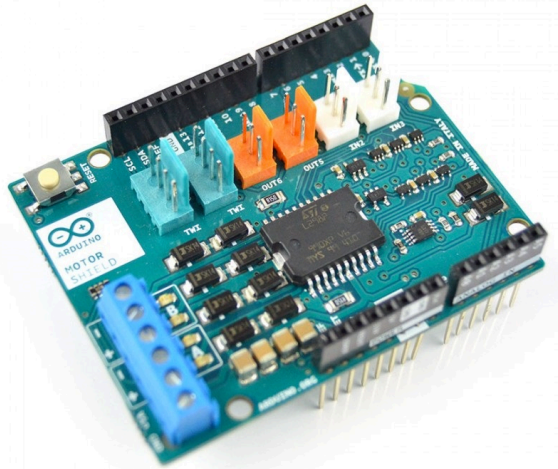
Arduino Shields



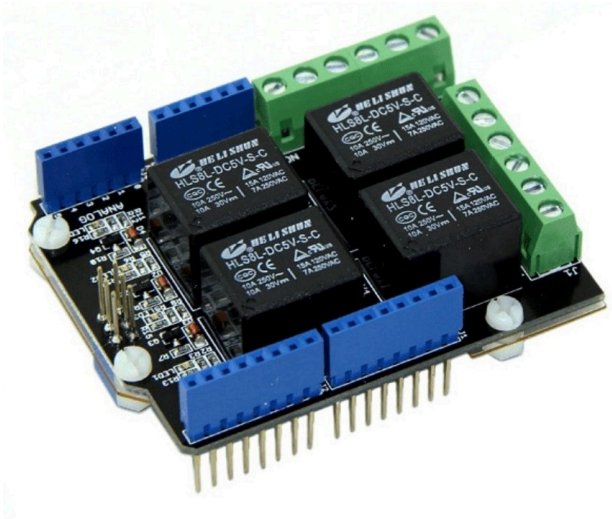
Arduino IF II board



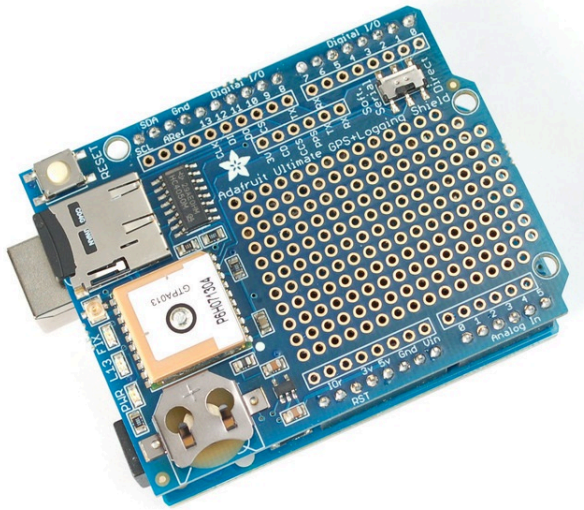
Base Shield



Motor Shield



Relay Shield



GPS Logger Shield

Native Development

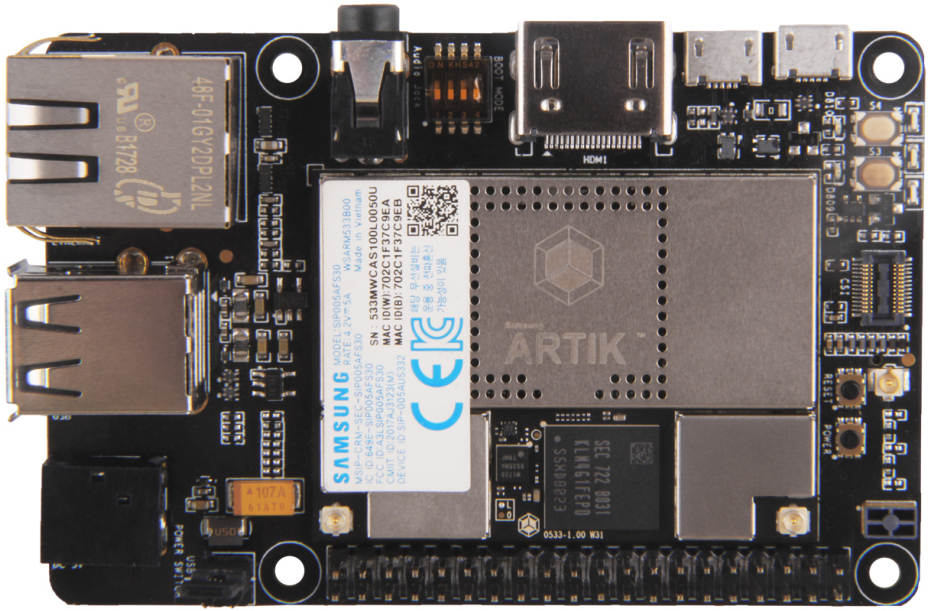
- C: Most popular programming languages for embedded devices. e.g, ARTIK SDK
- Python:
- JS: Node.js is the most popular JavaScript runtime for high-end IoT devices.
- Java: JDK is required.

3rd party Libraries/APIs

- Multimedia: PyAudio, OpenCV, Speech Recognition etc.
- Communication Protocols/Frameworks:
  - MQTT(Eclipse Mosquitto/Paho)
  - OPC-UA(Eclipse Milo, open62541)
  - LWM2M(Eclipse Wakama, Eclipse Leshan)

ARTIK Gateway Module Use Case

Seeed Eagleye 530s



- Powered by the Samsung ARTIK™ IoT platform. Raspberry-Pi form factor
- Incorporates ARTIK 530s 1GB SoM, a Quad Core Cortex® A9 running @ 1.2 GHz.
- Includes 40 pin GPIO and accessory interface.
- Support for Micro SD, Ethernet 10/100/1000, Wi-Fi 802.11 a/b/g/n, Bluetooth BLE 4.2 802.15.4, and ZigBee/Thread.
- Supports full HDMI, MIPI camera interface, video, and audio media.

ARTIK Ecosystem

Open Source Frameworks, Solutions

Gateway Solutions:

AWS Greengrass

kura™

EDGE X FOUNDRY™

Communication Protocols/Frameworks:

MQTT.ORG

IoTivity

Wakaama

eclipse smarthome

OPC UA™

Modbus

thingworx®

amazon web services

Microsoft Azure

SAP® Certified
Built on SAP Cloud Platform

vmware®

IBM Bluemix™

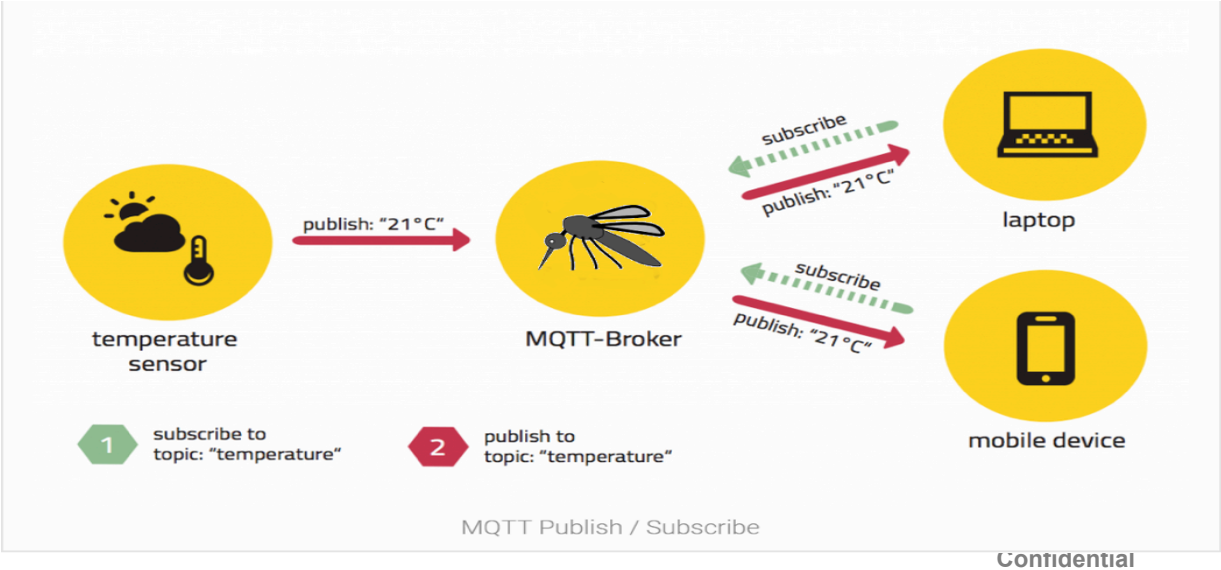
SAMSUNG ARTIK™

Lab Sessions

Lab 6: MQTT

Message Queue Telemetry Transport (MQTT)

- MQTT history
- Light-weight messaging protocol, rides on TCP
- Broker / Clients architecture
- Publication / Subscription messaging model
- No pre-defined format for payload



MQTT Publish / Subscribe

Message Queue Telemetry Transport (MQTT)

- The **publish/subscribe** message pattern to provide **one-to-many message distribution** and decoupling of applications
- Publish/subscribe is **event-driven** and enables messages to be pushed to clients
- The central communication point is the MQTT **broker**, it is in charge of dispatching all messages between the senders and the rightful receiver
- Each client that publishes a message to the broker, includes a **topic** into the message. **The topic is the routing information for the broker**
- Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client
- Therefore, the clients don’t have to know each other
- This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers

Message Queue Telemetry Transport (MQTT)

- A Message Queue stores message until they are consumed. In MQTT, it is possible the message is not processed by any client.
- In Message Queue, a message will only be consumed by one client
- MQTT has more flexibility compared to a message queue

Message Queue Telemetry Transport (MQTT)

- MQTT client includes publisher or subscriber
- In general, a MQTT client can be both a publisher & subscriber at the same time
- A MQTT client can run on any device from a micro controller up to a server. MQTT C client code only takes 30KB, Java code is about 100KB.
- MQTT client libraries are available for a huge variety of programming languages, e.g, C/C++, Arduino, Java, JavaScript, Android, iOS, C#, .NET

  https://github.com/mqtt/mqtt.github.io/wiki/libraries

- MQTT client: Eclipse Paho

  MQTT.fx(available for Win/MacOS/Linux) etc.

Message Queue Telemetry Transport (MQTT)

- MQTT Broker is responsible for receiving all messages, filtering them, and sending the messages to all subscribed clients.
- It holds the session of all persistent clients including subscriptions and missed messages
- Authentication and authorization of clients.
- Self Hosted MQTT brokers:

  Eclipse Mosquitto 

  HiveMQ(licensed) 

- Cloud based MQTT brokers:

  AWS 

  Microsoft Azure 

  IBM Bluemix 

  Eclipse Mosquitto (test.mosquitto.org) 

  HiveMQ (broker.hivemq.com) 

Message Queue Telemetry Transport (MQTT)

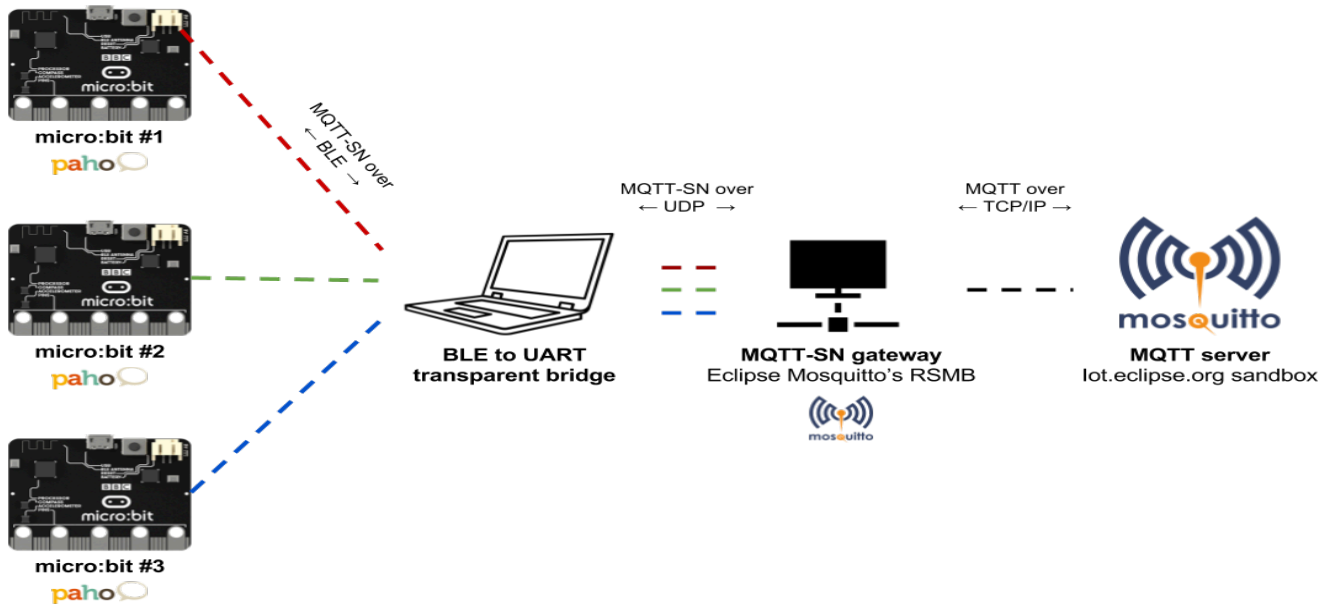
- MQTT supports TLS(Transport security Layer) SSL
- TLS and SSL are cryptographic protocols which use a handshake mechanism to negotiate various parameters to create a secure connection between the client and the server
- After the handshake is completed, an encrypted communication between client and server is established and no attacker can eavesdrop any part of the communication
- Servers provide a X509 certificate, typically issued by a trusted authority, which clients use to verify the identity of the server
- For Authentication, MQTT specification leaves it to individual brokers to implement.

Message Queue Telemetry Transport (MQTT)

- Topics can get very long.
- Runs on TCP, requires TCP/IP stack.
- How about non TCP-IP connections?

Message Queue Telemetry Transport (MQTT)

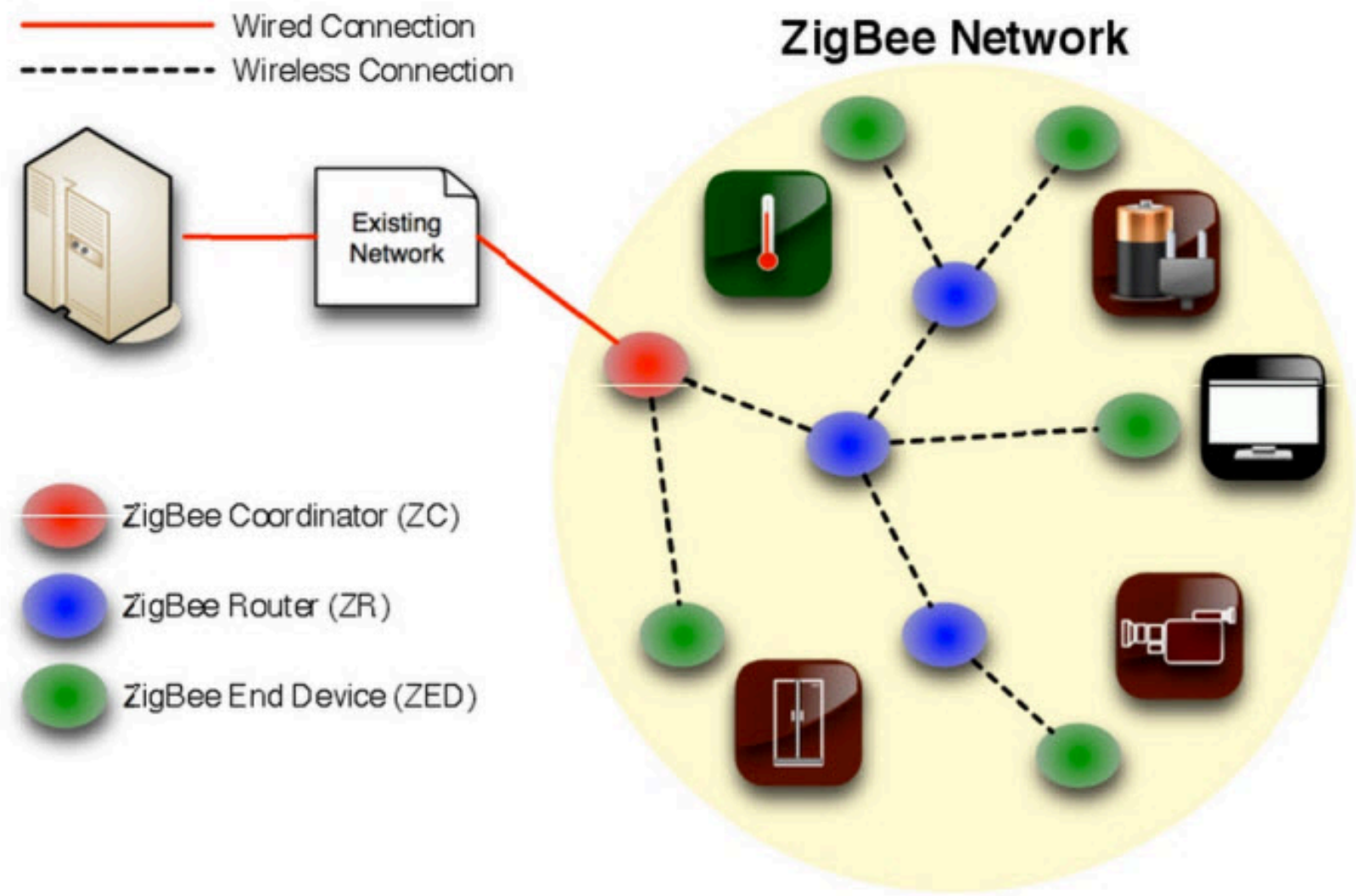
- MQTT-SN supports **topic ID** instead of using a topic name. Saves media bandwidth and device memory
- Topic ID to topic name can be preconfigured in MQTT-SN gateway
- MQTT-SN does not require TCP/IP stack, can be used over UDP. Supports protocols like ZigBee, Z-Wave, BLE.



SAMSUNG ARTIK™

Lab 7: ZigBee

ZigBee Architecture



APPENDIX