Getting Started with AWS IoT

e-Yantra Team
Embedded Real-Time Systems Lab
Indian Institute of Technology Bombay

January 3, 2018

Table of contents

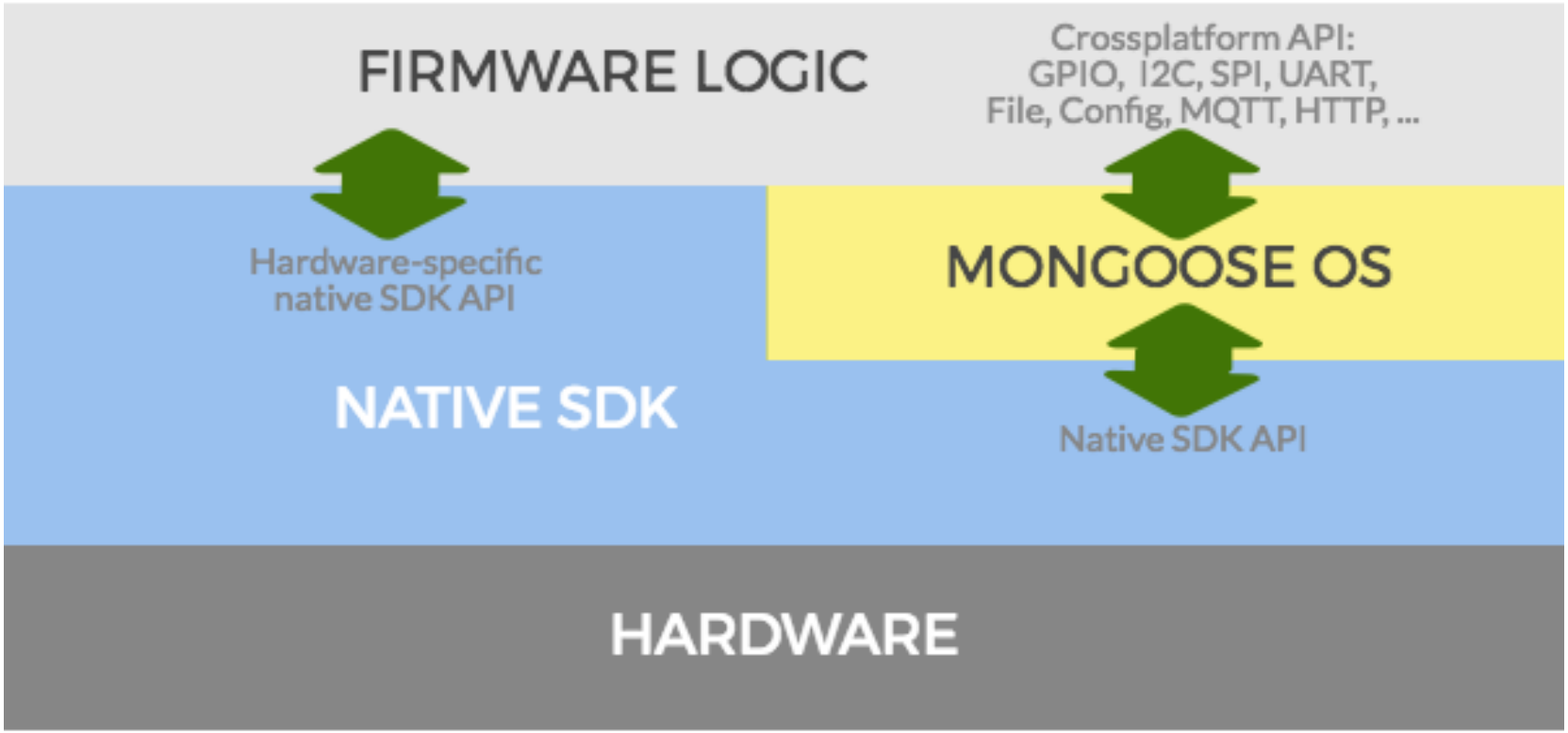
Mongoose OS introduction

An open source operating system for hardware that support javascript. It follows event-driven architecture of javascript with a non-blocking I/O model.

Features

- Supported microcontrollers: ESP32, ESP8266, CC3220, CC3200
- based on Mongoose library for networking
- APIs for GPIO, PWM and other peripherals
- Built in support for IoT cloud integration
- Manage devices using Remote Procedure Call(RPC)
- Write firmware in Javascript or in C
- Supports Over the Air(OTA) firmware update

Mongoose OS architecture



[0]Image Courtesy: https://mongoose-os.com/docs/book/images/arch1.png

Mongoose OS components

- mos tool - device management and firmware building
- build toolchain - Docker image for building a mongoose OS app (normally happens in cloud unless specified otherwise)
- ready-to-use apps and libraries

Before beginning hands on

- Hardware required: ESP8266 based device
- Software required: mos tool

Installing mos tool

Follow instruction in given link to install mos tool
https://mongoose-os.com/software.html

Getting the demo-js app

- Get the demo code from example repo: https://mongoose-os.com/docs/reference/apps.html

Mongoose OS app structure

- mos.yml
- fs/init.js
- fs/conf0..9.json
- src/main.c

Parts of Configuration File

- config schema
- filesystem
- src
- libs

init.js code walk-through

- Loading the library
- Variable definitions
- Helper functions
- Initialising peripherals and its handlers
- Running codes periodically using Timer function
- wifi network handler

Using MOS tool

- Command for building project
  mos build –arch=esp8266
- flashing the code
  mos flash

More mos commands and uses

- mos –help
- mos console
- mos config-get
- mos ls
- mos config-set mqtt.enable=true

Running the aws-dht-js app

Link for cloning the example app
https://github.com/sanamshakya/interfacing-AWS-IoT