Architecture of Non-OS JavaScript-based MCU development for IoT end device

Li GuangXiang, VP engineer, Insentek.

Abstraction

This paper summarizes a layered method of structuring software for MCU develop. With a JavaScript interpreter engine being applied, the coding effort required is found decreased remarkably. As a result, the cost for development could be reduced greatly. JavaScript being instead of C-compiled binary, the executive code is allowed to be downloaded to device at runtime. Developing model is also enhanced to allow people focusing more attention on product functionality, disregarding tricky underlying details.

Introduction

IoT trends to be the significant technology next to AI. A typical IoT network, or AKA, Wireless Sensor Network(WSN), consists of large number of end devices and a few gateways. Gateway level device has strong computing power so embedded Linux is widely used in this case. IoT end device usually is based on a resource-constrained MCU system. It is single-threaded mode instead of multi-threaded OS mode. All code is packed together to form a monolithic body. So the programming is a verbose job when the project becoming bigger. This paper addresses this problem by introducing a hierarchical software structuring, then code is grouped into 3 layers. Figure 1 depicted the difference between the two architectures.

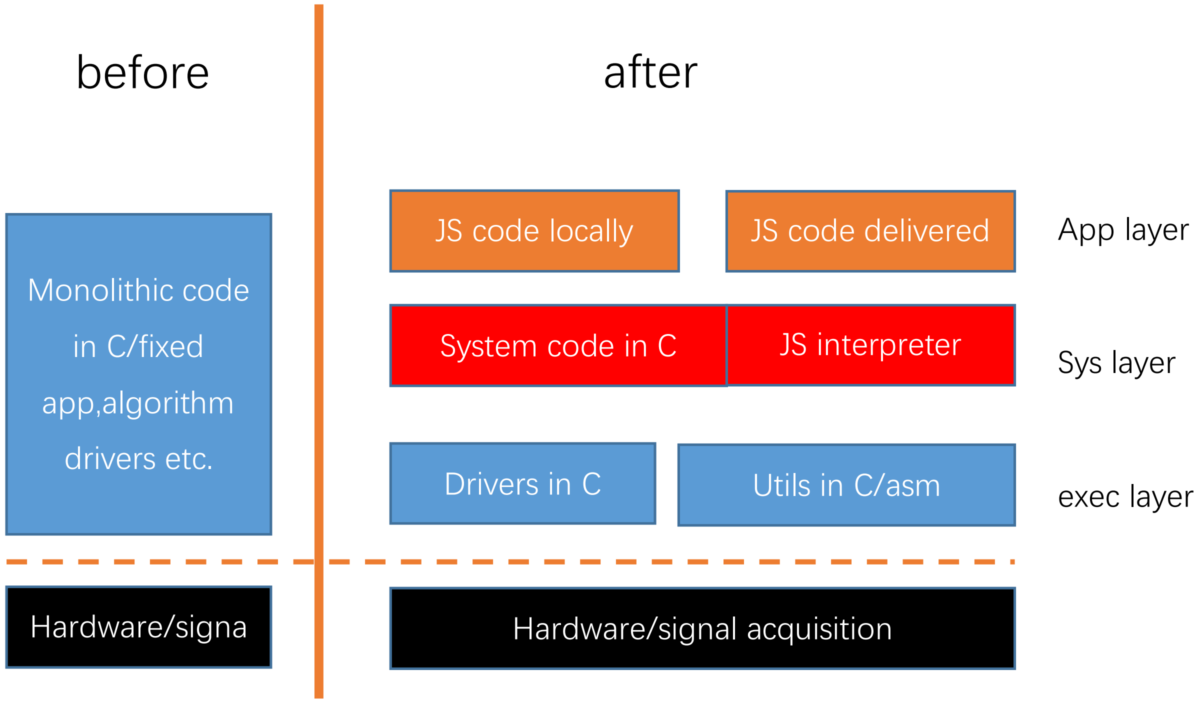


Figure 1. Difference between the two architectures

Architecture

When working, an IoT end device repeats the steps circularly at some specific intervals:

1) Acquisition of data,

2) executing algorithm on acquired data,

3) data delivery to network.

The hardware of IoT end device typically consists of signal process and wireless communication parts. The rest of the system relies on software code to organize how and when device works and sleep, etc., throughout the whole product life cycle.

The structure in Figure 1 looks similar to the modern OS model of “kernel-system call-application code”, but actually no any OS code existing here. The JavaScript main loop code gets running in the upper layer, that belongs to the application domain. Code in the bottom layer handles all hardware details, including communication stack, mostly wireless stack here. In the middle layer an appropriate tiny java-script code interpreter is integrated to translate JS code at runtime. The middle layer does isolate the codes from up and down, of JS and C respectively. The JavaScript interpreter acts as an important role that contributes to form a “standard” runtime environment. As an example of my application from this new architecture, the total binary code size for a stm32 MCU is found taking up several hundreds of kilobytes.

Taking advantage of the architecture, it comes true that, different sensor device product would only depend on different JS codes in the upper layer in Figure 1.

Code Independency

Code independency is considered as one of important objective of developing large-scale software. But when programming MCU this rule is usually ignored because the develop scale is small comparatively. Recently this issue is noticed again when the IoT end device becomes more rich functional, more interactional between devices and with network. So the architecture mentioned in this paper is implemented, as shown as in Figure 2.

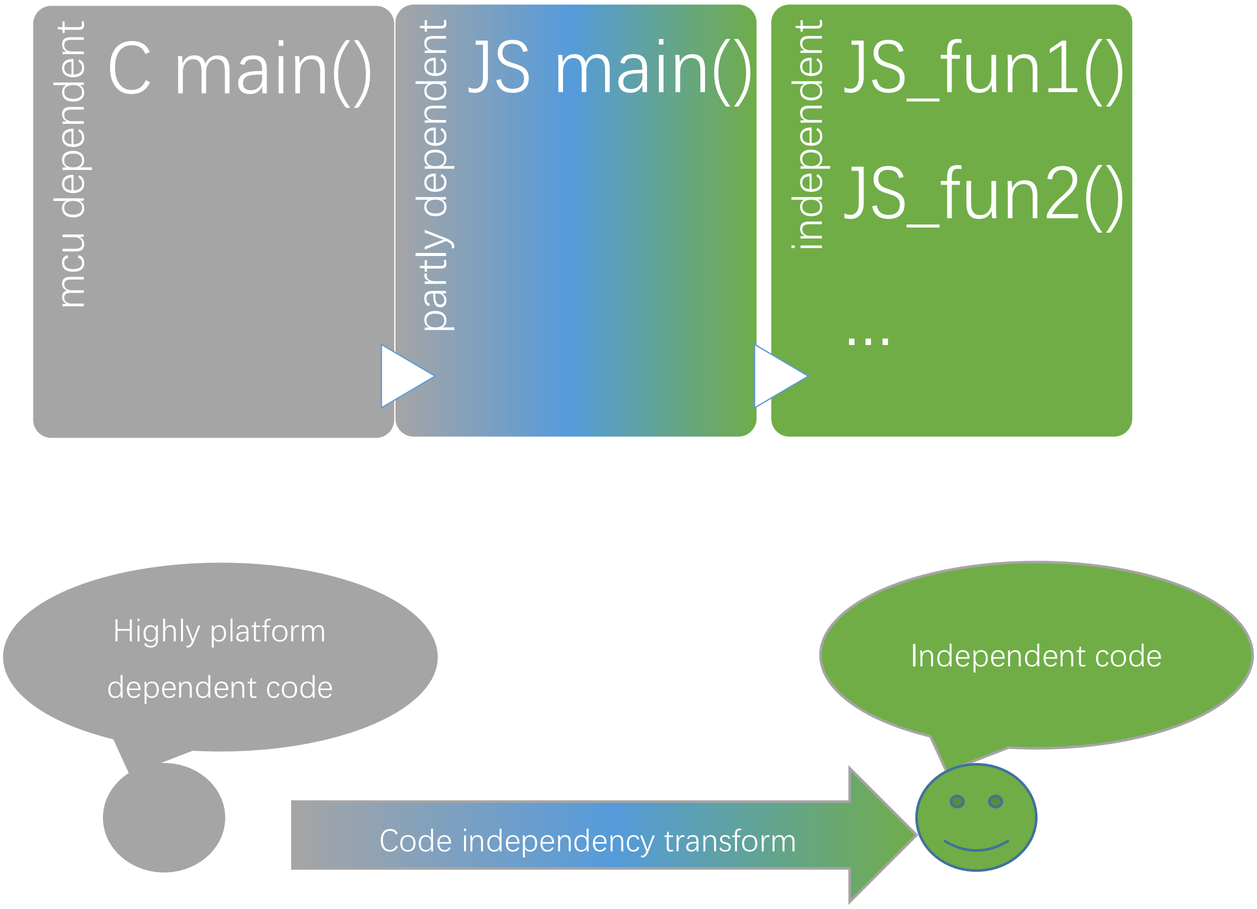


Figure 2. code independency

Flexibility

Under this architecture, some special funciton code could be implemented either in underlying C code or JS code. For example, the Kaman filtering algorithm has ever been implemented either in upper or lower layer respectively. It’s tested and proved that they performed the same logical effect but a little different efficiency, according to some extra overhead in JS-C transfer. With communication stack integrated, application in JS code does have the flexible ability to talk to its neighbor or cloud server. Under the architecture in this paper, the interaction among device nodes is possible to realize by delivering of pieces of JS code. Similar to the slogan of “Java code wrote once, running everywhere on internet”, now we have it realized in domain of Internet of Thing(IoT).

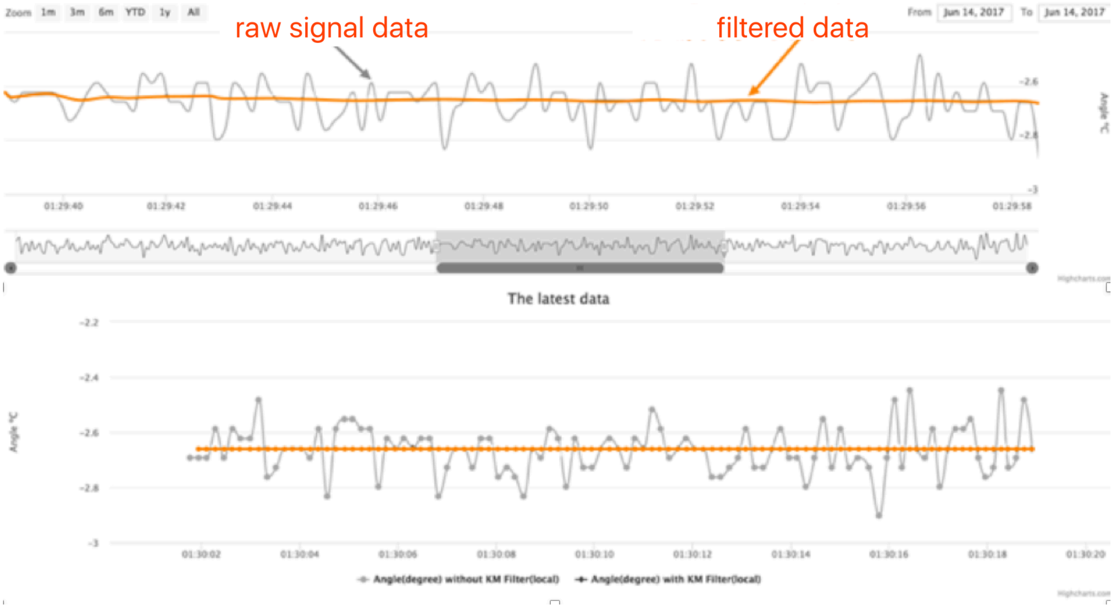
An IoT sensor device typically has small size, simple structure, 24-7 working, long lifetime battery supply, etc. A new software structuring such as in this paper is necessary to be developed to meet the needs.

Drivers

A set of functions is written to provide the Hardware Abstraction Layer(HAL). It includes encryption/decryption, data transfer over UART/SPI, firmware upgrader, and any more code which should be here. Some of them are encapsulated as JS-API functions to be called by JavaScript application code.

Software oscillography

During developing a sensor device, you often keep inspecting signal data very frequently. It’s inconvenient for people to operate under low level programming environment. So a simulated oscillography was implemented using JavaScript code in web server as shown in Figure 3. The signal develop board is connected to LAN via wireless Ethernet. Data is reported to a node.js server. Opening any PC and connecting to the server via web browser, shows the chat as the following Figure 4. In this picture it portrays how Kaman filtering works.



Conclusion

This paper demonstrates how to design and implement a layered software structuring on MCU(stm32 based) programming even without OS. It follows one of the principles of modern OS methodology, that's the model of "kernel-syscall-userland". It shows the rule of "code independency" can be fully implemented on MCU. This programming model is helpful to IoT end device develop. Additionally, a software oscillography is also implemented to help inspect signal processing in this paper.

About the author

With years of experience coding Linux kernel, system level coding on either RISC SoC or CISC architecture, embedded system, algorithms, wireless communication protocol, and general software develop. Such as ever porting Linux and android ver2.0 onto a MIPS platform, X86 BIOS hack, etc. Now is concentrating on develop of sensor/IoT products cooperating with top mechanics academic institution, for early warning of landslides, also find sensors data association using deep learning library to help on understanding physical phenomena. The author can be contacted by guangxiang.li@insentek.com or 1370058716@qq.com.