Title	センサIoTデバイスにおけるエミュレーション抽象化
Author(s)	広瀬，太志
Citation	
Issue Date	2019-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/15909
Rights	
Description	Supervisor:篠田　陽一，先端科学技術研究科，修士（情報科学）

Abstract of Emulation for Sensor IoT Devices

1710171 Futoshi Hirose

IoT (Internet of Things) makes it easy to measure and collect a wide range of environmental information. IoT devices equipped with various sensors are assumed to use in urban areas, farms, and the like. However, if the devices distribute widely, it is difficult to repair with the relocation after installation. Such a device has low power and uses a wireless communication module of a very narrow band. Therefore, updating with OTA (On-The-Air) is difficult. Although verification before installation is important, it is inefficient to prepare the necessary number of devices and perform the verification work on site. Therefore, this research considers verifying the IoT device system by emulation using a computer. Emulation of the sensor IoT device needs to execute a large number of device emulators at once. It requires many computing resources.

Therefore, in this research I propose to abstract a part of the functions of the emulator, depending on the type of applications and sensors to be verified. The emulator abstraction has a kind of trade-off relationship. In general, it is possible to reduce emulation calculation request resources by emulator abstraction. Conversely, it causes a decrease in fidelity as compared with the actual machine. Due to changes in fidelity, functions necessary for testing may not be faithful. In this case, the abstracted emulation cannot satisfy the test requirement. Also, if you do not implement an appropriate abstracted emulator, you will develop an emulator that uses more computation resources than necessary. Moreover, as the computation time of the emulator increases, the real-time performance of the execution gets worse. Therefore, this research clarified the trade-off relationship between fidelity and test requirements when abstracting the emulator and evaluated the effectiveness of the emulator abstraction.

I defined a four-layer model to clarify the trade-off relationship between emulator abstraction types. This base on the fact that emulation execution significantly changes the abstraction type at four points. The four points are layers of the application, library, system call, and hardware. For each layer, this research showed the abstraction target, the interface for abstraction, and organized the verifiable items. It also shows the advantages and disadvantages of emulation of these layers. For example, in the emulation of the application layer, it is possible to confirm the algorithm and protocol used by the software under test. However, since many layers including the library layer are abstracted, there are very few items that can verify. In another example, the hardware layer emulation imitates hardware exceptionally accurately, so many items can verify as much as an actual device. In exchange

for this, it requires extensive calculation resources. If using emulation with these emulation layer models must carefully consider the characteristics of the software under test and the IoT system.

Experiments were conducted to confirm the effectiveness of emulation abstraction based on the defined four-layer model. As a verification environment, a processor emulator using ARM architecture which is assumed to apply in embedded devices prepare. Sensor IoT device to be tested has a processor and some sensors and communicates between them with SPI (Serial Peripheral Interface). Software for sensor IoT device cannot verify without installing such I/O interface in the emulator. However, although an interface is necessary, it is not required to implement a protocol or the like faithfully. Therefore, abstract SPI communication function. There is no problem in verifying the operation of the application of the IoT device. In this research, I designed two kinds of emulators. The first is "SPI register model" which performs more faithful SPI communication processing. The SPI module has dedicated registers. By setting or getting data in this register, it communicates with the sensor connected to the wire. We defined the SPI hardware register, developed it by software, and implemented the emulator. The second is "Exodus model" which does not execute SPI communication processing faithfully. The Exodus model that realizes library level abstraction does not have SPI hardware module. To do SPI communication without using a module, I implemented a unique instruction word "exd" in the processor emulator. Besides, I rewrote the SPI library to use this exd. When processing related to the SPI communication in the library call, the mnemonic of exd interprets, and the host OS processes the communication processing instead. As a result, changing from the processing of the emulator with the significant overhead to the Linux native read/write system call.

In the evaluation, I conducted three experiments. First, Quantitatively confirm the change of computation request resource by abstraction of emulator by measuring emulator execution time. Compared to the SPI register model with high fidelity, the Exodus model emulator evaluated that the calculation resource requirement reduces because the 9.95% execution time was short. Second, by measuring the time only for SPI communication processing, the influence of abstraction on SPI communication processing was confirmed in detail. By the abstraction of SPI communication, the execution time of the emulator with the fidelity lowered at the SPI library level tended to shorten. Finally, I examined the amount of requested memory at emulation execution. In IoT device emulation, it is necessary to execute a large number of devices simultaneously. The results show that in the emulator used in the experiment, the memory usage is sufficiently small in the modern

computer. Although the required memory size is a problem that there may be a problem depending on the mounting method of the emulator. These results showed that it is possible to shorten the calculation time using the abstracted emulator and to reduce the requested resources of emulation.

IoT device emulation discussed the role to do in IoT system development. In the IoT system development process, I examined at what stage the verification by emulation should be designed and executed.

In this research, I showed the trade-off between emulation abstraction and confirmed that the abstraction of the emulator contributes to the reduction of request calculation resources in IoT device emulation.