5.3 MEMORY TECHNOLOGY

Memory is one of the fastest evolving technologies in embedded systems over the recent decade. No matter how fast processors can run, there is one unchanged fact so that every embedded system needs memory to store data. Furthermore, with the rapid development of the processor, more and more data pass back and forth between the processor and the memory. The bandwidth of a memory, which is the speed of the memory, becomes the major constraint impacting the system's performance.

When building an embedded system, the designers should consider the overall performance of the memory in the system. There are two key metrics for memory performance: *write ability* and *storage permanence*. Writing in memory can be various in different memory technologies. Some kinds of memories, such as *Random-Access Memory* (RAM), require special devices or techniques for writing. A RAM device allows data items to be read and written in roughly the same amount of time regardless of the order in which data items are accessed. The two main forms of modern RAM are *Static RAM* (SRAM) and *Dynamic RAM* (DRAM). In SRAM, a bit of data is stored using the state of a six transistor memory cells. This form of RAM is more

expensive to produce, but it is generally faster and requires less power than DRAM and, in modern computers, is often used as cache memory for the CPU. DRAM stores a bit of data using a transistor and capacitor pair, which together comprise a DRAM memory cell. The capacitor holds a high or low charge (1 or 0, respectively), and the transistor acts as a switch that lets the control circuitry on the chip read the capacitor's state of charge or change it. As this form of mem- ory is less expensive to produce than static RAM, it is the predominant form of computer memory used in modern computers.

At the high end of the memory technology, we can select the memory that the processor can write to simply in a short time. There are some kinds of memories that can be accessed by setting address lines, or data bits, or control lines appropriately. At the middle of the range of memory technology, some slow written memory can be chosen. At the low end are the types of memory that require special equipment for writing.

Besides the write ability, we also need to take storage permanence into consideration. How long the memory can hold the written bits in themselves can have a key impact on the reliability of the system. In the aspect of storage permanence, there are two kinds of memory technologies: *nonvolatile* and *volatile*. The major difference is that the

nonvolatile memory can hold the written bits after power is no longer supplied, but volatile cannot. The nonvolatile memory is typically used for the task of secondary storage, or long-term persistent storage. Meanwhile, the most widely used form of primary storage to- day is volatile memory. When the computer is shut down, anything contained in the volatile memory is lost. The advanced memory technology needs to attach to the operating system. Dynamic programming is an option for heterogeneous memories' optimizations, which will be discussed later.

5.4 MOBILE EMBEDDED SYSTEMS

5.4.1 Embedded Systems in Mobile Devices

A mobile device is a typical embedded system, which is formed by a group of electronic components, such as mobile processors, storage, memory, graphics, sensors, camera, battery, and other chips. Integrating these electronic parts is to achieve a variety of desired functions for different purposes. In this section, we will use the smartphone to represent an example of a mobile embedded system. A smart phone is one of the most adopted mobile devices in contemporary people's lives. Currently, the hardware structures of most smartphones are two-processor frameworks. The two processors are the application processor

and the baseband processor, which are shown in Fig. 5.14. The *Application Processor* is in charge of running a mobile operating system and various kinds of mobile apps. It is the one that controls the whole system. Most functions provided by chips, such as the keyboard, screen, camera, and sensors, are controlled by the application processor.

Meanwhile, the *Baseband Processor* is responsible for wireless communication. This wireless communication is not the cellular or Wi-Fi network, and it is the telephone network with *Radio Frequency* (RF). The radio frequency is a rate of oscillation, which corresponds to the frequency of radio waves, and the alternating currents that carry ra- dio signals. The radio frequency module is used to send signals to the telephone network. There are two other basic modules in the *baseband processor*, which are the *Digital Baseband* (DBB) and the *Ana- log Baseband* (ABB). They modulate and demodulate the voice signal and the digital signal, encode and decode the communication channel,

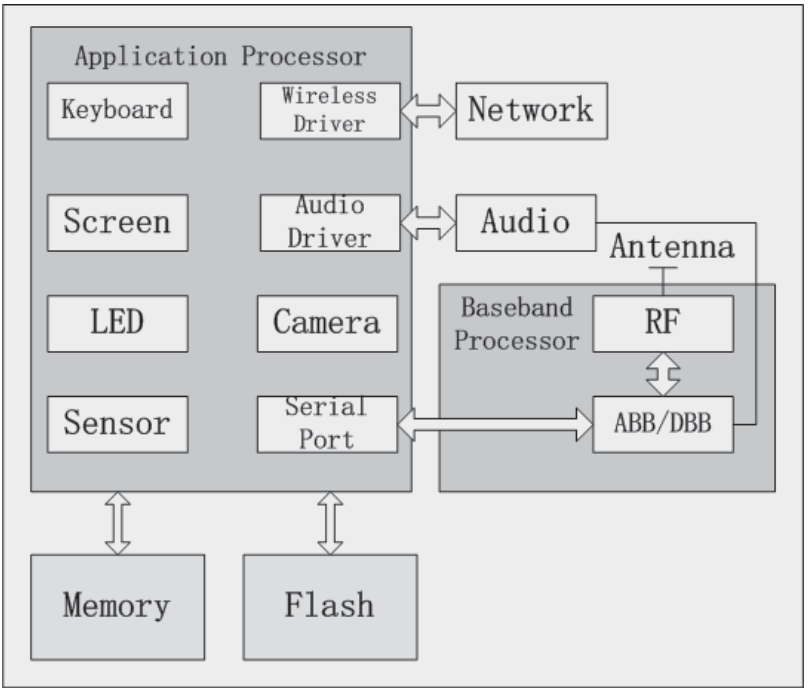

Figure 5.14 Hardware structure of a smartphone.

and control the wireless modem (modulator-demodulator). The application processor communicates with the baseband processor via the serial port, USB, and others.

5.4.2 Embedded Systems in Android

After introducing the hardware structure of the smartphone, we will take Android as an example to explain the Kernel inside Android and show how the Kernel works. As discussed in Chapter 1, Android is based on the Linux Kernel, and the Linux Kernel is an abstract layer between the hardware and the software. The basic functions of An-

droid are provided by the Linux Kernel core system service, such as file management, memory management, process management, network stack, and drivers. The Linux Kernel also provides drivers to support all the hardware related to the mobile embedded system. As shown in Fig. 5.15, there are display driver, keyboard driver, audio driver, power management, Wi-Fi driver, camera driver, and other sensor drivers. We will list some of them and explain what they do.

Display Driver. It is based on the framebuffer driver in Linux. The framebuffer offers a mechanism that allows the application to directly control the change of the screen.

Keyboard Driver. It is the driver for buttons on the mobile device, such as the Home button, the Menu button, the Return button, and the Power button.

Wi-Fi Driver. It is the driver for Wi-Fi connection based on IEEE 802.11.

Sensor Driver. Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy, and are useful if you want to monitor three-dimensional device movement or positioning, or you want

to monitor changes in the ambient environment near a device.

For example, a game might track readings from a device's gravity sensor to infer complex user gestures and motions, such as tilt, shake, rotation, or swing. Likewise, a weather application might use a device's temperature sensor and humidity sensor to calculate and report the dewpoint, or a travel application might use the geomagnetic field sensor and accelerometer to report a compass bearing.

Above the Linux Kernel is the hardware abstraction layer, which provides an easy way for applications to discover the hardware on the system. The "abstract" of the hardware abstraction layer does not mean the real operations of the hardware, and the operations of the hardware are still achieved by drivers. However, the interfaces offered by the hardware abstraction layer make it simple for developers to "use" the hardware.

Hardware Abstraction Layer Hardware abstraction layer is a software subsystem for UNIX-based operating systems providing hard- ware abstraction. The purpose of the hardware abstraction layer is to allow application to discover and use the hardware of the host system through a simple, portable, and abstract Application Programming Interface (API), regardless of the type of the underlying hardware.

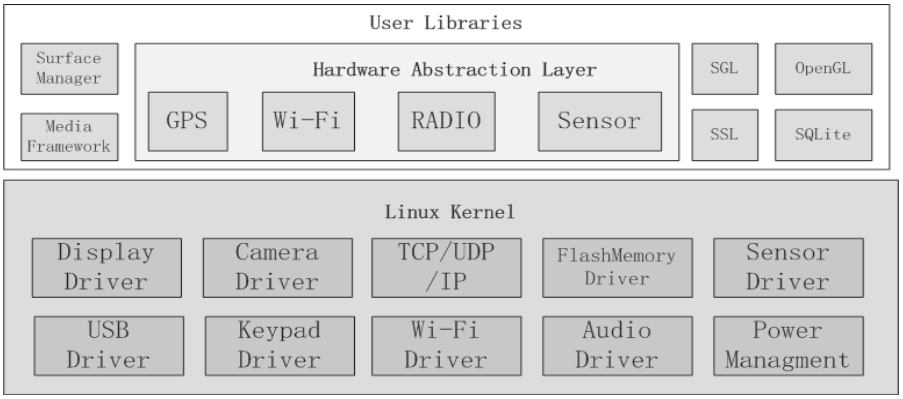

Figure 5.15 Linux Kernel of Android.

5.4.3 Power Management of Android

Android supports its own power management (on top of the standard Linux power management) designed with the premise that the CPU should not consume power if no applications or services require power. As shown in Fig. 5.16, Android requires that applications and services request CPU resources with wake locks through the Android application framework and native Linux libraries. If there are no active wake locks, Android will shut down the CPU. The wake locks are used by applications and services to request CPU resources. The power management uses wake locks and time-out mechanism to switch the state of system power, so that system power consumption decreases.

Currently, Android only supports screen, keyboard, buttons backlight, and the brightness of the screen. As shown in Fig. 5.17, when a user application acquires full wake lock or a screen/keyboard touch

activity event occurs, the machine will enter “awake” state. If timeout happens or the power key is pressed, the machine will enters the “notification” state. If partial wake locks are acquired, it will remain in “notification”. If all partial locks are released, the machine will go into “sleep.

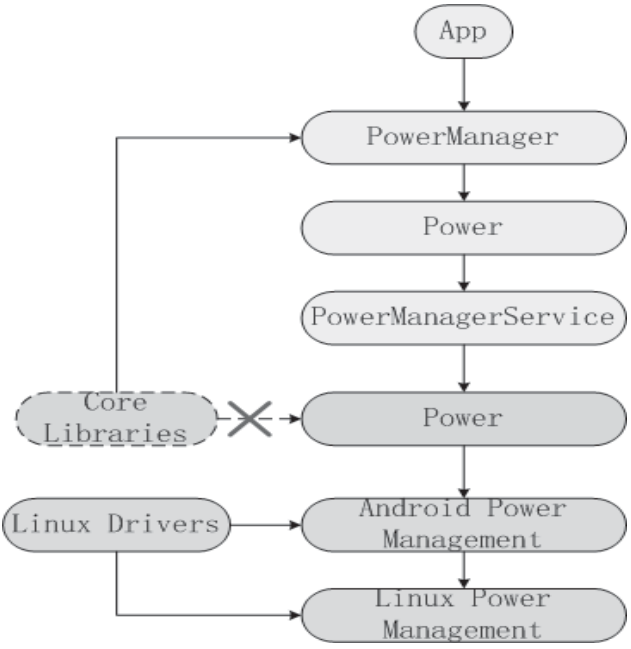

Figure 5.16 Power management of Android.

5.4.4 Embedded Systems in Mobile Apps

The mobile embedded systems are under the layer of mobile operating systems. The mobile embedded systems cannot directly used by mobile apps, and they only can be used through mobile operating systems, such as iOS and Android. We will take Android as an example.

Android is already an embedded operating system, and its roots are

derived from embedded Linux. The main hardware platform for Android is the Acorn RISC Machine (ARM) architecture. ARM is a family of instruction set architectures for computer processors based on a reduced instruction set computing architecture. An approach that is based on reduced instruction set reduces costs, heat, and power consumption. Such reductions are desirable traits for light, portable, battery-powered devices, and other embedded systems. Android devices incorporate many optional hardware components, including cameras, GPS, orientation sensors, dedicated gaming controls, accelerometers, gyroscopes, barometers, magnetometers, proximity sensors, thermometers, and touch-screens.

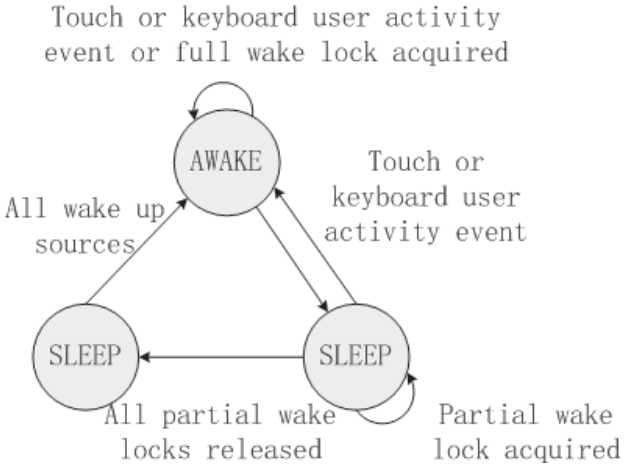

Figure 5.17 A finite-state machine of the Android power management.

We can use Android *Software Development Kit* (SDK) to develop our own mobile apps, and via the methods that are already imple-

mented to use the embedded systems inside a mobile device. For example, developers only can use the camera of a mobile device through calling methods encapsulated in the Android SDK. This design method makes the process of developing mobile apps much simper than old methods. The developers do not need to spend time designing the interaction with embedded systems inside an Android device, and they only need to know what functions Android SDK can provide. We will introduce more knowledge about Android SDK and developing technologies in the next chapter.