

Gaining Access to Legacy I/O Devices with Android

A green arrow pointing to the right, containing the text 'How to Access I/O devices'.

How to Access I/O devices

A red arrow pointing to the right, containing the text 'What is Smart ETK'.

What is Smart ETK

An orange arrow pointing to the right, containing the text 'Implementation of Smart ETK'.

Implementation of Smart ETK

A blue arrow pointing to the right, containing the text 'Application of Smart ETK'.

Application of Smart ETK

How to Access I/O Devices

 **Android Framework and HAL**

 **Flow of I/O Access**

 **Introduction of JNI**

 **Sample**

 **Summary**

How to Access I/O Devices

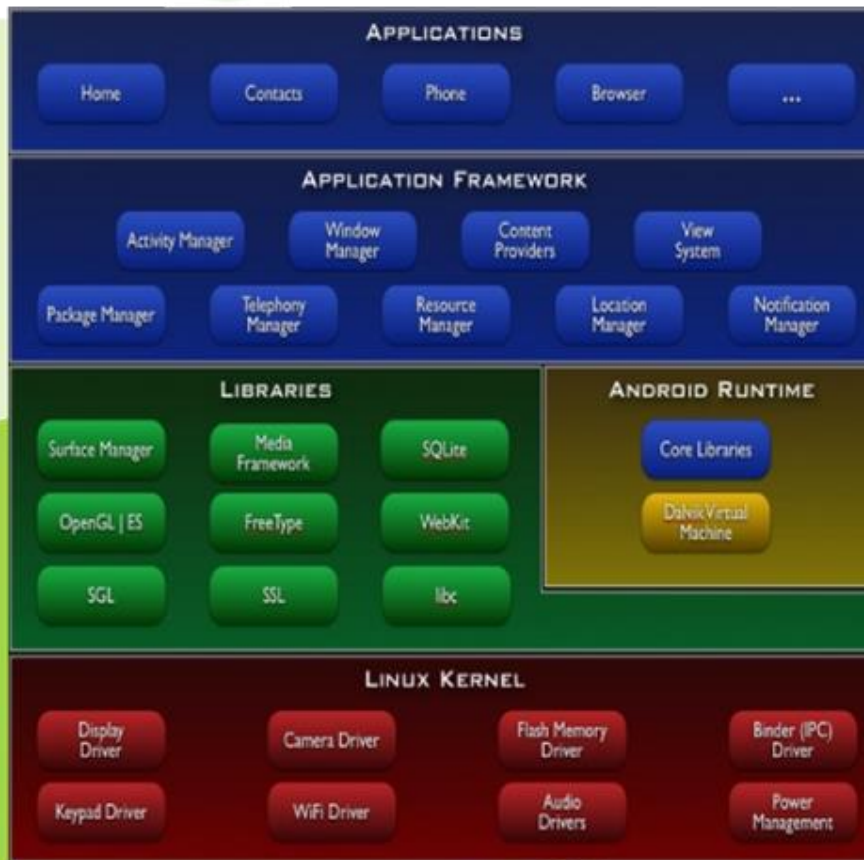




Android Framework



Conventional Four-Layer Framework



Application Layer

Provides Java-based user interface

Application Framework Layer

Provides interfaces, services and resource management for applications

System Runtime Layer

Provides shared libraries, Android runtime library and Dalvik virtual machine

Linux Kernel

Provides kernel level system services such as security, memory & process management, network stack and drivers

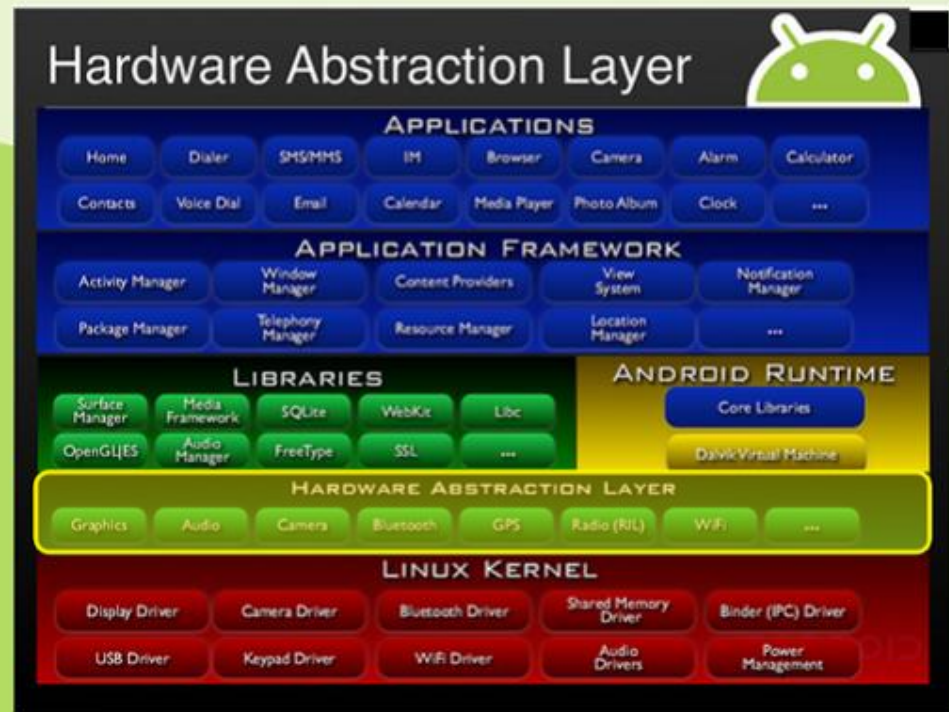


HAL



Android Framework with HAL

Package of Linux Kernel Driver, Providing interface upwards and shielding lower implement details



Flow of Access to I/O



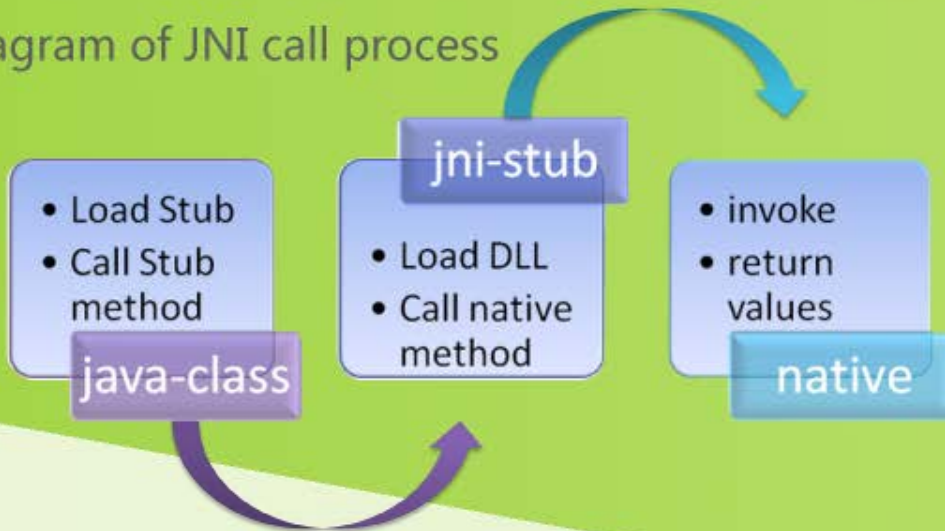


Introduction of JNI

JNI: Java Native Interface

Support mutual operation between Java and C or C++ running in JVM, or local assembly code

Diagram of JNI call process



Implement JNI by c/c++ and practice objective native function by JNI. For Java, jvm indirectly calls objective native function by loading and calling JNI



Sample

Sample of reading and writing GPIO devices by JNI

Step 1: Compile .so files for GPIO driver

- a. Add GPIO.java files into Android project
GPIO.java concludes the native interface running GPIO
- b. Use javah command on GPIO.java to generate XXX.h file
javah -classpath bin/classes -d jni com.via.vepd.GPIO

Create com_via_vepd_GPIO.c file under Linux to implement the formerly defined interface:

For example:

```
jint JNICALL Java_com_via_vepd_GPIO_setEnable(JNIEnv *env, jobject c, jboolean enable)
{
    int result;
    // do GPIO operation
    return result;
}
```

Use gcc to compile com_via_vepd_GPIO.c and generate libGPIO.so file



Sample

Sample of reading and writing GPIO devices by JNI

Step 2: Use driver library files in application

- a. Add libSmartETK.co provided by VIA under the directory of:
libs\armeabi\ in Android project
- b. Add GPIO.java into Android project

Summary



Difficulties in accessing I/O devices in Android:



Linux Kernel background required

Hardware background required

Linux driver debugging skill required

Java and C/C++ programming skills required

In Summary : Device driver development is high cost and high risk

How to access to I/O devices

What is Smart ETK

Implementation of Smart ETK

Application of Smart ETK



What is Smart ETK



VIA Smart ETK



An embedded tool kit featuring a rich set of APIs, such as GPIO, audio and LAN, etc., which help to shorten development time and speed up time to market.

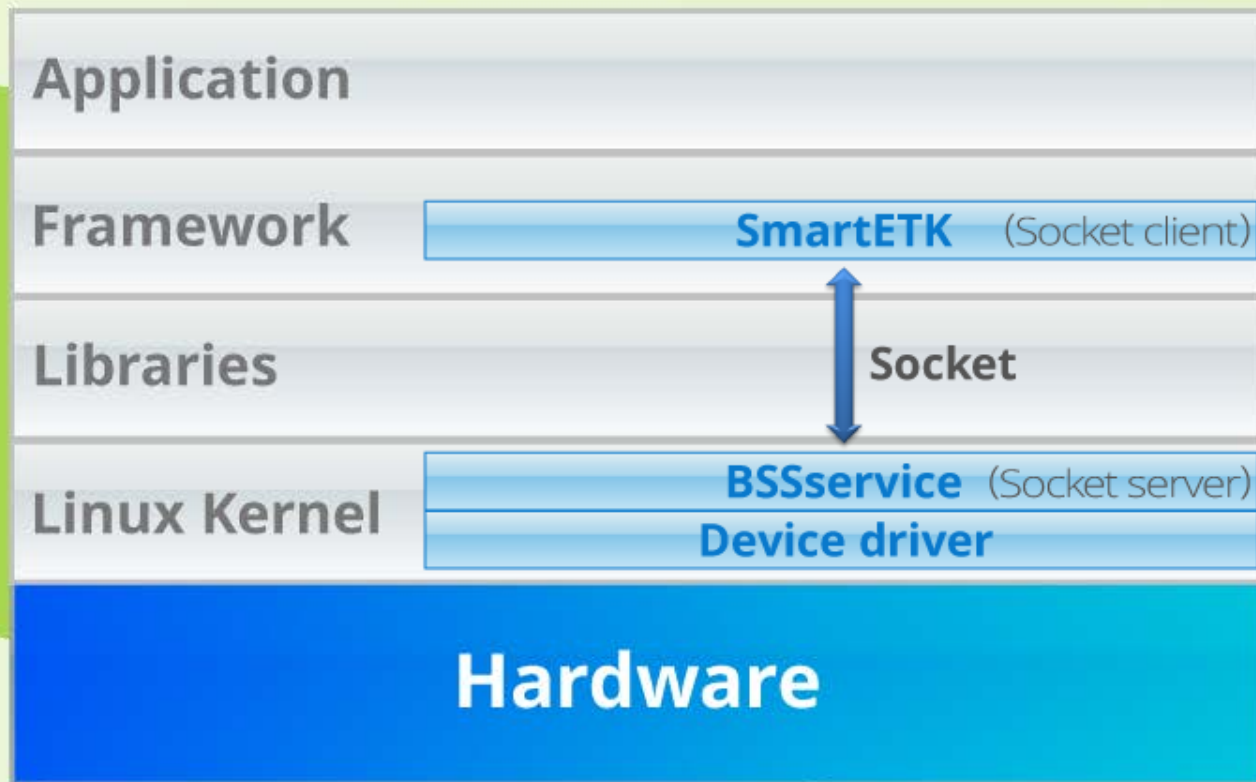
How to Access to I/O Devices

What is Smart ETK

Implementation of Smart ETK

Application of Smart ETK

Implementation of Smart ETK



Implementation of Smart ETK

Socket

Multi-threading, scalable

VS

JNI

Single task, complex syntax,
faster than socket

How to Access to I/O Devices

What is Smart ETK

Implementation of Smart ETK

Application of Smart ETK



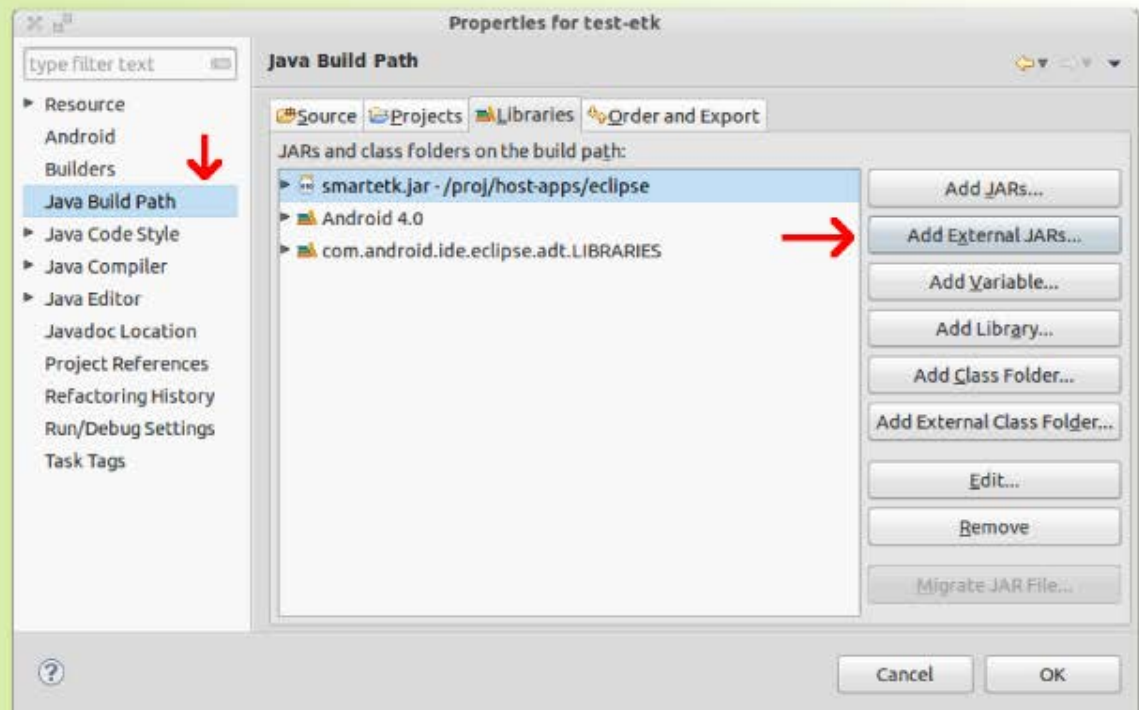
Application of Smart ETK



Sample 1. Use Smart ETK to implement reading and writing GPIO device

a. Import Smart ETK

On the properties page of Android project, visit "Add External JARs..." to import smartetk.jar into the project.





Application of Smart ETK

● Sample 1. Use Smart ETK to implement reading and writing GPIO device

b. Call APIs provided by Smart ETK in application layer programs

```
import com.via.vepd.SmartETK;
```

```
/* Declare variables to get GPIO6 values */
```

```
boolean[] bEnable = new boolean[1];
```

```
int[] nDirection = new int[1];
```

```
int[] nValue = new int[1];
```

```
GPIO gpio6 = new GPIO(6);
```

```
gpio6.setEnable(true);
```

```
gpio6.setDirection(GPIO.GM_GPO);
```

```
gpio6.setValue(1);
```

```
gpio6.getEnable(bEnable);
```

```
gpio6.getDirection(nDirection);
```

```
gpio6.getValue(nValue);
```

Using Smart ETK

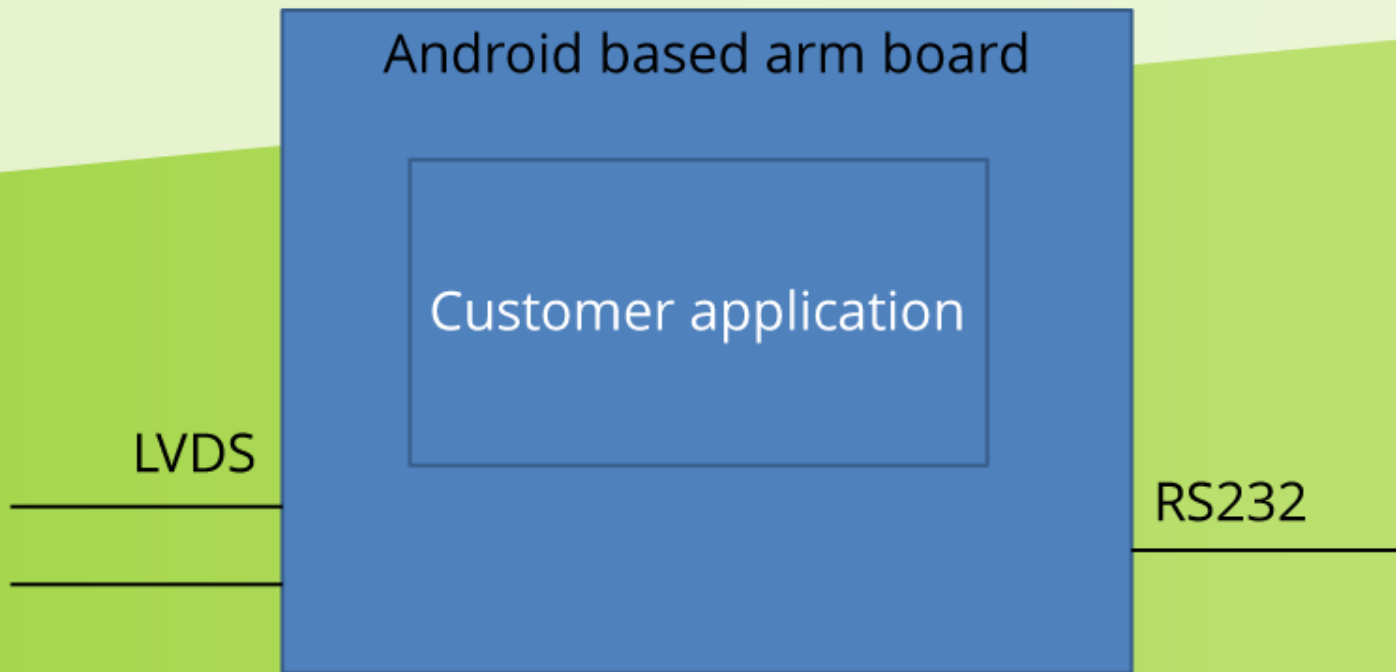
Simple, Fast

No need to develop and test drivers

Eliminates issue of APIs requiring system level authority.

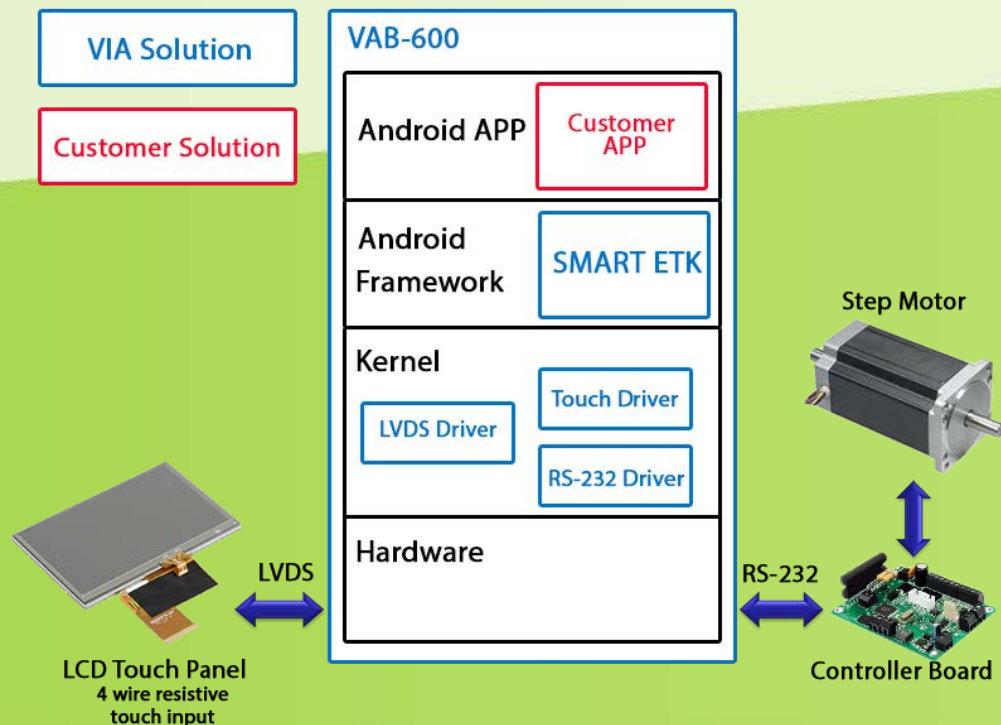
Application of Smart ETK

- **Sample 2. Industrial Automation Example:**
Customer required to use an Android application to control a step motor as well as utilize a touch LCD panel to complete certain tasks



Application of Smart ETK

Sample 2. Industrial Automation Example:
Customer required to use an Android application to control a step motor as well as utilize a touch LCD panel to complete certain tasks



Smart ETK Advantages

With Smart ETK

1. Application uses APIs in agreed style to control all peripheral devices, making it easy and convenient
2. VAB-600+ Smart ETK
Low investment of human resources, increased product reliability

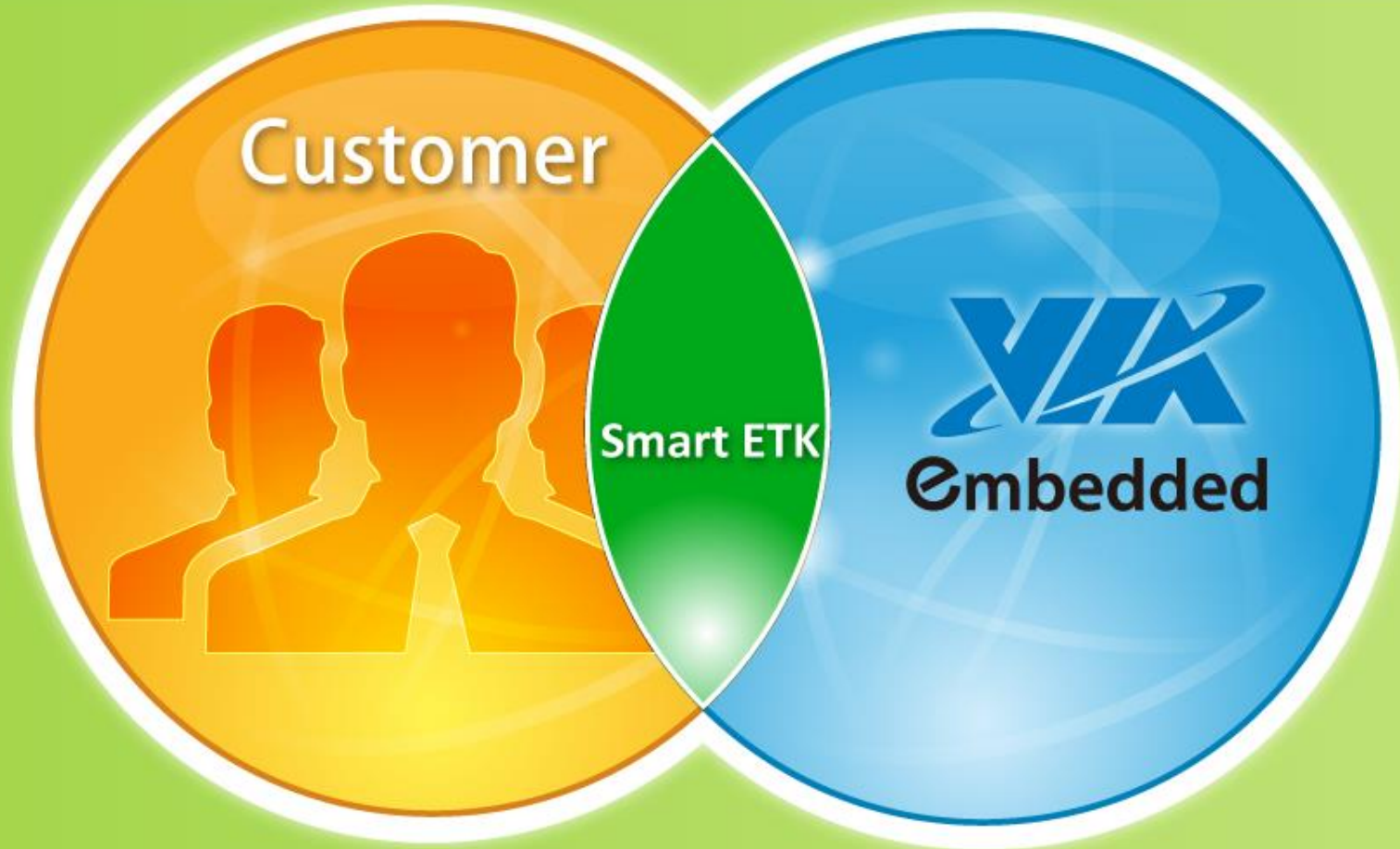


Without Smart ETK

1. Need to develop and modify drivers for all peripheral devices
2. Android BSP needs to be modified
3. Harder to integrate software and hardware



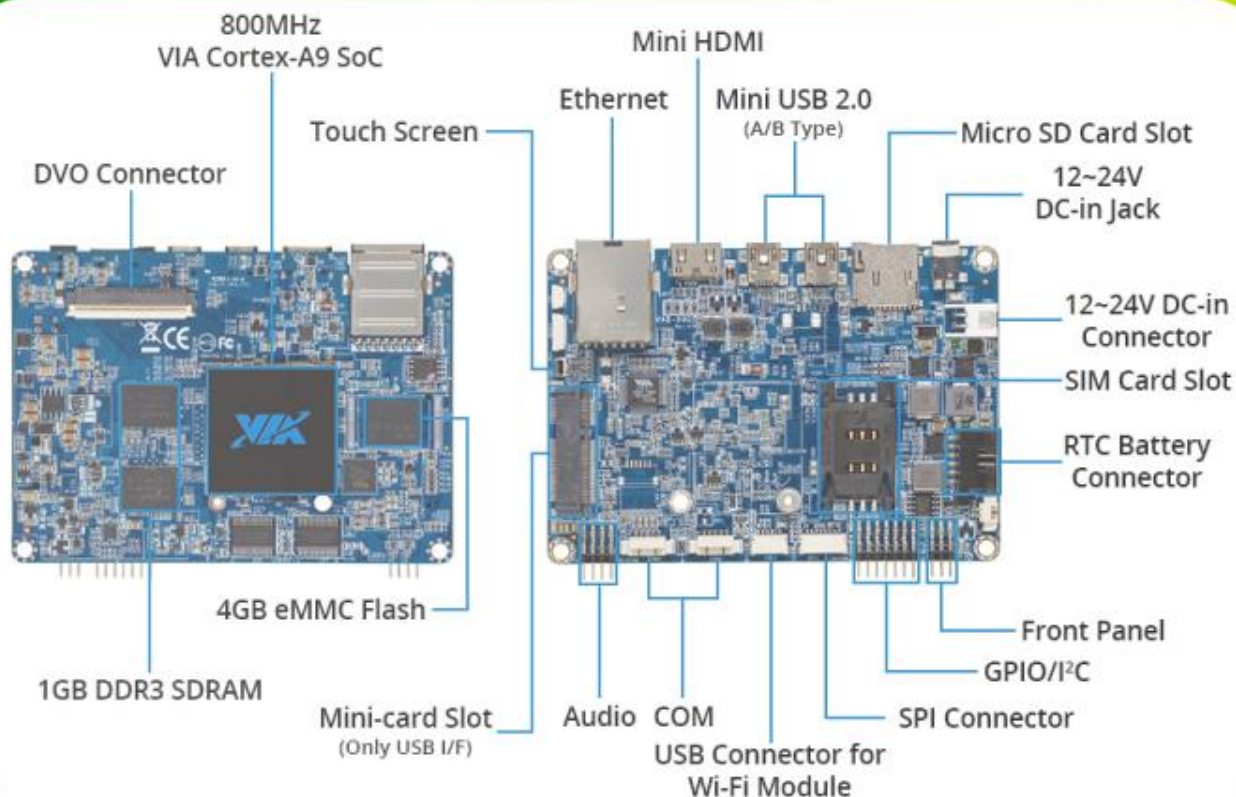
Creating Value for Both Sides



Smart ETK helps to reduce risk and cost of development

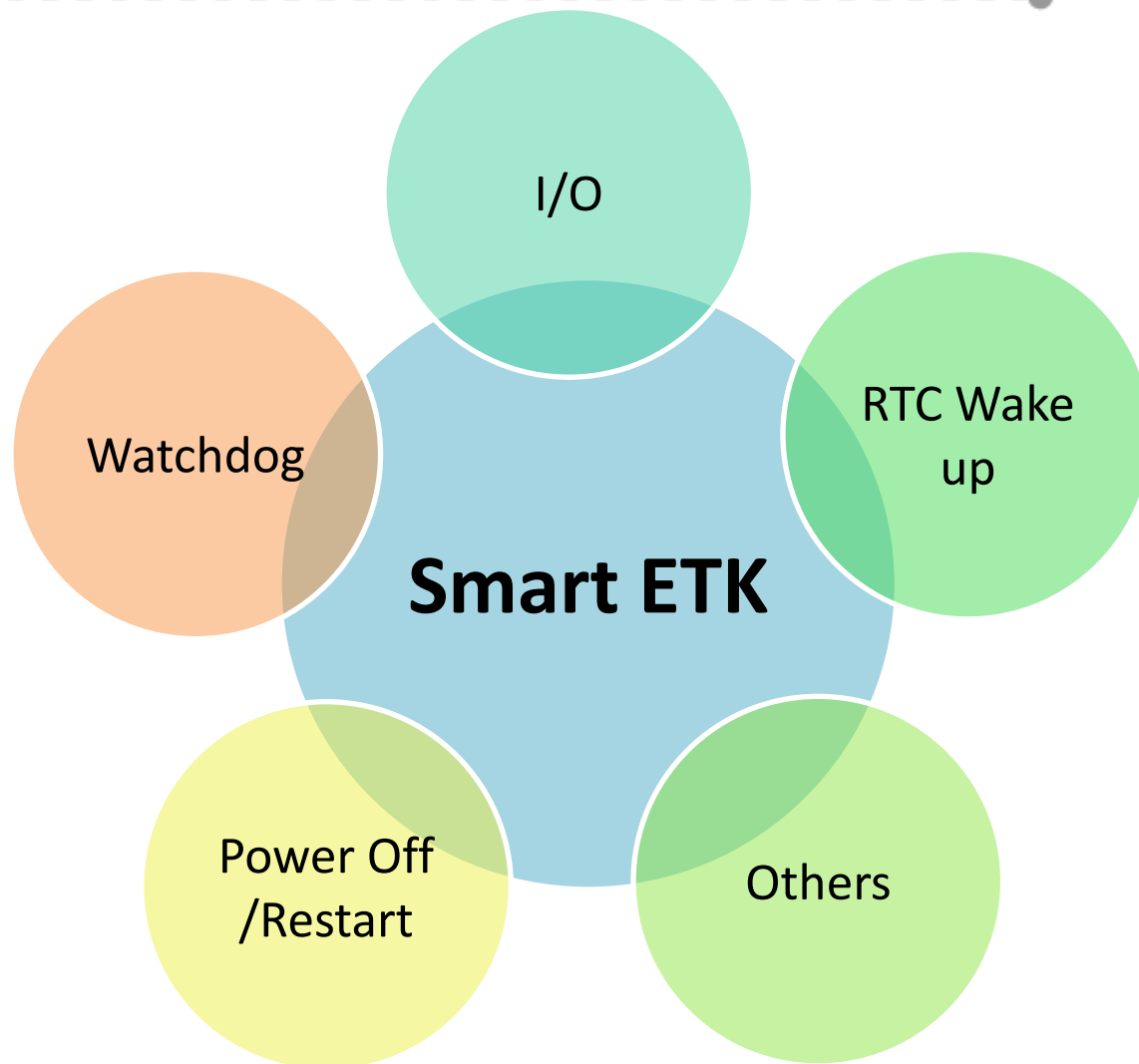
VAB-600

Ultra-compact 10cm x 7.2cm Pico-ITX form factor with rich I/O feature set



<http://www.viaembedded.com/cn/boards-cn/pico-itx/vab-600/>

Smart ETK Full Function Display



RTC Wake Up Function

Community Signage examples:

Android 4.0.3-based signage player

Automatic daily startup at 7:00

Automatic daily shutdown at 19:00





Better
Performance

More functions

Open, standard

Future Smart ETK



© 2015 VIA Technologies, Inc All Rights Reserved.

- *VIA reserves the right to make changes in its products without notice in order to improve design or performance characteristics.*
- *This publication neither states nor implies any representations or warranties of any kind, including but not limited to any implied warranty of merchantability or fitness for a particular purpose. No license, express or implied, to any intellectual property rights is granted by this document.*
- *VIA makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication or the information contained herein, and reserves the right to make changes at any time, without notice. VIA disclaims responsibility for any consequences resulting from the use of the information included herein.*
- *VIA C7®, VIA C7®-D, VIA C7®-M, and VIA Eden™ are trademarks of VIA Technologies, Inc.*