# Embedded Internet of Things Applications of SQLite Based on WinCE Mobile Terminal

Yuanhui Yu ( aiia0212@163.com )

Research

**Keywords:** SQLite, WinCE platforms, Mobile Terminal

**Posted Date:** September 11th, 2020

**DOI:** https://doi.org/10.21203/rs.3.rs-37411/v2

# Abstract

As we all know, embedded systems are becoming more and more widely used. All devices with digital interfaces, such as watches, microwave ovens, video recorders, automobiles, etc., use embedded systems, but most embedded systems are implemented by a single embedded application program to achieve the entire control logic.At present, embedded applications on the WinCE platform are extending towards microservices and miniaturization. More embedded device application data requires small, embedded database systems to organize, store, and manage. The embedded database SQLite has many advantages such as zero-configuration, lightweight, multiple interfaces, easy portability, readable source code, and open source. It is currently widely used in the WinCE embedded operating system. This article discusses the technical characteristics of the SQLite database in detail, SQLite data manipulation, SQLite transplantation under the WinCE platform, and finally implements SQLite data management on WinCE mobile terminal based on MFC programming.


# 1. Introduction

In recent years, with the continuous development of computer technology, the importance of embedded development has been continuously enhanced, and the application of databases has become more and more prominent. Compared with other databases, the characteristics of SQLite and the working environment of WinCE Features have a large degree of fit, so they are widely used. This article focuses on understanding this database and exploring its applications to some extent.

WinCE system is a small embedded operating system introduced by Microsoft. It is designed for handheld electronic devices. This operating system allows existing Windows desktop technologies to be integrated with complete mobile technology. As we all know, many Microsoft Windows operating systems use the same standard, so even if the platform changes, the system can still run, but WinCE is not like this, so it is only applicable to some specific platforms. Therefore, to achieve a variety of product requirements, WinCE is designed to be able to use a variety of different standard modes, which means that it can choose the required mode from which to customize its products. Also, its components can be some available models, which means that these modes can be selected from a set of available groups and then become standard models. Although Microsoft claims that "CE" in WinCE has no special meaning, the embedded development field has many different views on CE. One of the main points is that C in WinCE stands for consumption, pocket-size, communication ability or companion, and E stands for electronic products. It seems that there is a certain relationship between WinCE and the meaning of these words [6,7]. WinCE is a new embedded operating system developed by Microsoft. Compared with the previous desktop operating system, all relevant source code of WinCE system is developed by Microsoft, although the system interface of WinCE is taken from the previous Desktop operating system, it is still a newly developed, new information device platform based on win32 API. WinCE is a processor-independent system based on the Win32 application program interface. It also has the characteristics of structure and modularity. WinCE, like desktop operating systems such as Windows 95/98, can use programming software such as VB and VC, and the functions and interface design styles used in the

programming process have not changed much. In this way, when you want to use the application on the desktop operating system in WinCE, you only need to modify it. The system architecture of WinCE is hierarchical, as shown in Figure 1, divided into four layers from bottom to top. They are the hardware layer, OEM layer, operating system layer and application program layer.

Figure 1 WinCE System Architecture

Each layer is described in detail below:

1) Hardware Layer

The core of the hardware layer is an embedded processor, and there are many types of embedded processors, with different processing capabilities and main frequency speeds. If you want to run the WINCE operating system, the processor needs to include an MMU unit. WINCE6.0 supports four CPU architecture processors, they are ARM, MIPS, x86, SHx.Embedded hardware generally has the following characteristics:

- The architecture of the processor is not uniform.
- Hardware resources are generally limited.
- There are many types of external devices.
- Requirements for real-time performance and reliability.

2) BSP Layer

The BSP layer corresponds to the OEM layer. The BSP is tied to the operating system, and the BIOS work at the beginning is similar, but most of them are different. BSP can add a system-independent drive, and can even be developed on the upper layer, while the BIOS is not. The main function of this layer is to abstract the hardware and provide a unified interface for the operating system to interact with the hardware. It mainly includes OAL (OEM Abstraction Layer, OEM abstraction layer), bootloader, configuration files and drivers.

It should be noted here that BSP and BIOS are different. BIOS is an interface between hardware and software. It mainly does three things, self-checking and initializing the computer, completing program service processing and hardware interrupt processing, and processing program service requests. At the beginning of the embedded operating system startup, the BSP is responsible for the work similar to the BIOS. It is also responsible for powering up the system, initializing various devices, and loading the operating system. But BSP and BIOS are different, especially after the system is started. The difference between the two is mainly reflected in the following three aspects.

- BSP is compatible with the operating system, but the BIOS is compatible with the motherboard where it is located. The former is to allow its own hardware to support a certain embedded operating

system, while the latter is to allow all operating systems to be produced normally. Works fine on the hardware.

- Developers can modify the BSP and add some things they want to add, like drivers. But BIOS generally cannot be changed, and developers can only upgrade it or change the configuration. Relatively speaking, embedded developers have greater autonomy for BSP.
- One BSP corresponds to one hardware and one embedded operating system, that is, the same processor may correspond to multiple BSPs, and the same embedded operating system requires different BSPs for different processors. A BIOS corresponds to one hardware and multiple operating systems, which means that as long as the hardware is determined, the BIOS is determined, no matter which general operating system is used.

3) Operating System Layer

The operating system layer implements WinCE as the main function of the operating system. WinCE's process management, thread management, scheduling, physical memory, virtual memory, file system, and device management are all implemented at this level. The basic functions of the operating system layer are implemented in multiple independent processes (exe). When running, these processes mainly include:NK.EXE,GWES.EXE,FILESYS.EXE,DEVICE.EXE and SERVICES.EXE.

When the system is running, the kernel of WinCE behaves as NK.EXE process. NK.EXE is a core process that exists in all WinCE systems. It represents the core functions of process creation and loading, thoroughbred height, interrupt handling and internal management in the win32 API core. NK.EXE is composed of NK.LIB and OAL.LIB. NK.LIB is provided by Microsoft. Its code is related to the CPU instruction architecture and has nothing to do with specific peripherals. This design can make OAL as small as possible. OAL.LIB is the output of OAL code in the OEM layer after being compiled.

4) Application Layer

The application layer of the WinCE system is very similar to desktop Windows such as Windows 95/98 and is located at the highest level of the operating system hierarchy. From a system-level perspective, in WinCE, each running application is considered to be a separate process in the system, and the Win32 API is still the interface between the application and the operating system. However, the API of the WinCE operating system is only a part of the desktop system API, and because the system resources of the embedded operating system are limited, the API that occupies too many resources in the desktop API is difficult to implement or has a low utilization rate is selectively filtered. If you want to re-use these APIs, you can only write them yourself. Also, the WinCE system adds some unique APIs.

# 2. Introduction To Sqlite Database

The original idea of the SQLite database originated from D. Richard Ship, who was writing a program for the guided-missile destroyer for the U.S. Navy. Because the Informix database used at the time was too large and time-consuming, Shipp and colleagues using the GNU DBM hash library as the backend, a

simple embedded SQL database was created. Then with the next major upgrade, SQLite has developed steadily, and features and users have also grown. By mid-2001, many open source or commercial projects started using SQLite. After the extension of Java, Ruby, Python and other mainstream programming languages in the following years, the emergence of new extensions such as SQLite's ODBC interface has once again proved the widespread application and practical functions of SQLite. It has been 15 years since 2015, and SQLite has ushered in the release of SQLite3. SQLite3 is a brand new version of SQLite. Although it was developed based on SQLite 2.8.13 code, it uses database formats and APIs that are not compatible with previous versions. SQLite3 meets the requirements of some new features, such as supporting UTF-16 encoding, supporting user-defined text sorting methods, and indexing BLOBs fields. And the SQLite database version 3.0 and 2.X version APIs are very similar, but there are some important changes to note: All API interface functions and data structure prefixes have been changed from "sqlite_" to "sqlite3_". Link conflicts occur when using SQLite 2.X and SQLite 3.0. Because there is no consistent specification of what data type C language should use to store UTF-16 encoded strings. Therefore, SQLite uses ordinary void * types to point to UTF-16 encoded strings. The client can use the void * maps to any data type suitable for their system. Three functions are most commonly used in SQLite 3.0 database operations: sqlite3_open(), sqlite3_exec(), and sqlite3_close(). If you want to better control the execution of the database engine, you can use the provided sqlite3_prepare() function to compile SQL statements into bytecode, and then use the sqlite3_step() function to execute the compiled bytecode.

### 2.1 SQLite Meaning

SQLite is a lightweight database and a relational database management system that complies with ACID. It is contained in a relatively small C library and implements a self-sufficient, serverless, zero-configuration, transactional SQL database engine. Just like other databases, the SQLite engine is not a separate process and can be statically or dynamically connected as required by the application. SQLite directly accesses its stored files.

### 2.2 SQLite Features

SQLite is an embedded database, which is similar to Microsoft Access, but a .db format file. But unlike Access, it does not require any software to be installed and is very lightweight. It is used by many embedded applications, including Tencent QQ, Thunder (you can see a sqlite3.dll file in the Thunder installation directory, that is it), and now the well-known Android and so on. SQlite3 is its third major version. Specifically, the SQLite database has the following characteristics:

1) SQLite database management is simple and very stable

SQLite is a single file, so it is easy to manage. The file format is very stable on many major versions. The characteristic of SQLite is that the database exists as a single file and has a stable format. Also, SQLite is easy to configure. The functionality of SQLite can be managed in two ways: compilation flags and PRAGMA statements (runtime configuration). There is no such thing as a configuration file, you just need

to build the library you want and then configure the runtime options when you create the database connection.

2) Scalability and controllability of SQLite database

Since SQLite is embedded in the application, it runs in the same address space and can execute the application code on your behalf. Whether it is the Python standard library SQLite driver pysqlite or the optional driver aspw, it can provide APIs that define custom SQL functions, aggregation functions, and classifications. aspw goes one step further and provides APIs that can also be used to define virtual tables and virtual file systems.

3) SQLite database is very fast

SQLite is very fast. Because it runs in the same address space, there is no line protocol, serialization, and no need to communicate through UNIX sockets. And it runs on the same computer, so it has no network burden when executing queries or reading results. Also, SQLite can run on mobile devices.

4) SQLite integration of BerkeleyDB

BerkeleyDB only needs to lock individual pages, not the entire database, so it can give application developers the concurrent database access required even better performance. Another benefit of BerkeleyDB is that it uses fewer system resources to improve system efficiency.

### 2.3 Technical Analysis of SQLite Database

1) Design goals of SQLite3

The design goal of SQLite is to achieve an embedded, secure, efficient, stable, easy-to-manage database that takes up very low system resources. Especially in embedded devices, only a few hundred K of memory may be needed to ensure its smooth operation.

2) The internal structure of SQLite3

Inside the system, SQLite3 is mainly composed of the following components: SQL compiler, kernel, backend, and accessories. In an implementation, SQLite3 is a database that uses the virtual machine and VDBE (Virtual Database Engine) to debug, extend and modify the kernel of SQLite3. SQLite3 supports databases up to 2TB in size. Disk files storing each database can be moved between computers in different orders. SQLite3 obtains its database permissions based on the file system. In SQLite3, all SQL statements are compiled into human-readable assemblies that can be executed in the SQLite3 virtual machine [1]-[8].

# 3. Methods

### 3.1 Create Operation

In SQLite, as with SQL Server and MySQL, the creation operations include creating tables, views, and indexes. Details as follows:

1) Create a table

create table name (field name 1 field type 1, field name 2 field type 2, ...);

// You can not specify the field type

We tend to specify types.

E.g:

CREATE TABLE person1 (a, b, c); // The end of this statement must use ";"

CREATE TABLE person2 (id int primary key autoincrement, name varchar (20));

Note: primary key: This field is the primary key. It cannot be duplicated if it has a unique value.

create table if not exists table name (field name 1 field type 1, field name 2 field type 2, ...);

CREATE [[TEMP | TEMPORARY] TABLE table_name (column_definitions, [constraints]);

Square brackets indicate optional items. Vertical bars mean one of the multiple options. {} Contains a list of options, indicating that one must be selected

TEMP: declares that the created table is a temporary table and only lives in the current session. Once the connection is disconnected, it will be automatically deleted.

table_name: Enter the table name here.

column_definitions: Represents a comma-separated column definition (also called a field list), each field includes a field name, field constraints and field type. If there are multiple constraints, the constraints are separated by spaces.

[, constraints]: Represents constraints. For example, you can use UNIQUE constraints to specify that the value of a field in all records should be different.

2) Create a view

E.g:

sqlite> CREATE VIEW Westview AS SELECT * FROM testable WHERE first_col> 50;

// Create the simplest view

sqlite> CREATE TEMP VIEW tempview AS SELECT * FROM testtable WHERE first_col> 50; // Create a temporary view

Create index

E.g:

sqlite> create index test_idx on mytable (value);

### 3.2 Modify Table Structure Operation

Similar to the SQL server and MySQL, SQLite's modification table is also an operation to add, delete, modify, and check the tables in the database. Often these operations also apply to the where operation, that is, the use of conditional statements. If you just want to update or delete certain table fixed records, you must add some conditions after the DML statement to achieve. The following table 1 lists some common conditional statements and common formats:

Table 1 SQLite Modified Table Structure Common Conditional Statements

| Common conditional statements | note |
|---|---|
| where Column name = value | Cannot use two = |
| where Column name is value | is Equivalent to = |
| where Column name != Some value | != Means not equal |
| where Column name is not value | is not Equivalent to != |
| where Column name > value | Means greater than a certain value |
| where Column name 1 = value and Field 2 > value | and Equivalent to && in C |
| where Column name 1 = value or Field 2 = value | or Equivalent to \|\| in C |

### 3.3 Data Query Operation

1) Exact query

Exact lookup, that is, using the exact conditions to find the corresponding data in the table or view or index.

E.g:

select * from table1 where id = 1; // This means to find the data with id 1 in the test table

sprintf (show, "select word from history where username = '% s';", Classmsg.name);

2) Fuzzy query

The fuzzy query is usually to give a range of data and then use like followed by conditions that require data to find related data.

select * from <table name>; // all data;

select * from table1 where name like "s%"; //% means any number of characters

select * from table1 where name like "s_"; // _ means any character

select * from table2 limit 0,3; // 0 means starting from // first line, 3 means querying three records

select * from student where age> 10; // conditional query

The select statement is relatively complex in SQL, and it is also the essence of SQL. Syntactically, the select statement is composed of a series of words, and each word can complete a specific operation. The select statement except select Except for the sentence, all the remaining words are optional.

### 3.4 Delete Operation

1) Delete data

delete from <table name> where condition;

If the sentence does not have a where part, then executing the statement will delete the entire table's data.

E.g:

delete from table name;

delete from table1 where id = 1; // Delete the data with id 1 in the test table

delete from table1 where score <90; // Delete data in test table with a score less than 90

In the delete operation, you can also use where and like to perform precise deletion and fuzzy deletion. The use of where and like in delete is the same as that. Therefore, no specific description is given here, but only a brief description in the above example. The same applies to views and indexes.

2) Delete the table (the same method for views and indexes)

E.g:

sqlite> DROP TABLE test; // Delete the test table

sqlite> DROP TABLE IF EXISTS test;

### 3.5 Increase Operation

The insert statement is used to increase the data. When using this statement, pay attention to the column names and values must be corresponding, you can not confuse the order, otherwise, the data insertion is prone to errors.

insert into <table name> (field list) values (corresponding values); // when using a string, you can add double or single quotes

E.g:

insert into test (name, age) values ("mike", 21); // Insert information named mike, age 21

sqlite> insert into test (id, value) values (3, 'Rose'); // Insert information with id 3 named Rose

### 3.6 Update Data

update table name set field 1 = value of field 1, field 2 = value of field 2, ...;

E.g:

update test set name = 'Mike', age = 22; // Change the name of all records in the test table to Mike and age to 22.

Similarly, conditional statements can also be used when updating data. The method of use is to add conditional statements after the statement. The method is similar to the previous one, so it will not be described

### 3.7 Other SQLite output commands

As shown in Table 2 below [9-10].

Table 2 Other SQLite output commands

| SQLite output command | Effect |
|---|---|
| .schema [table] | Structure of the display table |
| .tables | Get all tables and views |
| .indices [table ] | Get the index list of the specified table |
| .output [filename ] | Export database to SQL file |
| .read [filename ] | Import database from SQL file |
| .output [filename.csv ] | Format output data to CSV format |
| .import [filename.csv ] newtable | Import data from a CSV file into a table |
| sqlite3 [database] .dump > [filename] | Backup database |
| sqlite3 [database ] < [filename ] * | Restore the database |

# 4. Experimental

### 4.1 WinCE Platform Introduction

The Win CE platform is a member of many Windows families. Everyone understands that Windows is the most used operating system in the world, has the most mature GUI, and many resources required for application development. Win CE deserves these advantages. The WinCE platform is the cornerstone of Microsoft's embedded, mobile computing platform. It is an open, embedded operating system applied to small devices. WinCE is a streamlined Win 95, and the graphical user interface of Win CE is very advantageous and outstanding. To flexibly adapt to the needs of a wide range of products, Win CE can adopt different standard models, which means that different products can be customized by the forecast.

After customization, WinCE can achieve the minimum mode required by existing systems, thereby reducing the storage of scripts and the operation of the operating system. WinCE is a compact, intact and scalable embedded operating system. It is a multi-threaded, priority-based preemptive operating system. It is especially suitable for hardware platforms with relatively few resources. The system is not safe, reducing the risk of damaging the system due to abnormal applications face to face. Its kernel can also be used for streamlining and customization. The products on the WinCE platform are roughly divided into three product lines: Pocket PC (handheld computer) Handheld PC (handheld PC) and Auto PC. In recent years, the market share of embedded products has become higher and higher. The WinCE platform can be used in consumer electronics equipment, network terminals, Internet access equipment, industrial data acquisition controllers, mobile handheld devices and embedded devices, and provides modular software systems for them. The WinCE operating system is not inferior to the current mainstream Android operating system in the market, which can be reflected in the following four aspects:

1. Good security. The system is mature, well-closed, long used, stable, and easy to integrate with Windows systems.
2. Application software compatibility is better.
3. It still has a large market share.
4. Compared with the Android system, it supports more hardware platforms, such as x86, ARM, MIPS, SuperH and other mainstream CPU structures in the embedded field.

### 4.2 Transplanting of SQLite on WinCE Platform

SQLite porting will require different projects depending on the platform. Here is an example of EVC to briefly introduce the transplantation process.

First, download the SQLite source code from the http://www.sqlite.org/ website, which includes files in two different formats, .c and .h, and extract the two files into the same directory. Secondly, you need to install and configure the WinCE platform compiler. This requires you to install a software development kit customized by your development version in a certain development environment, such as Windows. After the installation, you will find support under the compiler selection in EVC. Options for this development board.

Next, it is to create a suitable project file and create a new project in the Ethernet virtual link. Here we take the dynamic link library as an example for analysis. First, select the DLL in the project type, use "SQLite" as the project file name, and then select ARMV4I in the "CPU type" option. The next step is to generate the final library file. Add the downloaded source file first, and then write the interface definition module, definition file, and remove some external library files. Take version 4.3.1 as an example. Shell. c, tclSQLite. c, ICU. c is the three files that need to be removed. The first file is mainly used to generate management tools. In the project of building DLL, this file does not affect. The second file is mainly used to support TCL. The script, but it is not used in embedded programming, so there is no need to exist. The third file is

to fuse the ICU library and the SQLite library. In this process, the ICU library is not needed, so it needs to be used. Removed. The last step is to optimize the library files according to specific needs.

As we all know, the WinCE platform is a relatively popular and widely used operating system in the embedded devices of smart mobile terminals. This system is characterized by small storage space, so when developing applications in the WinCE system, such as Microsoft's SQLServer database and Sybase's SQL Anywhere database and Oracle's Oracle database [11], it takes up a little more resources. Traditional databases are no longer suitable for this system. Based on the above analysis, this article will introduce a relatively compact and high-performance embedded database SQLite [2,12]. This database can more easily implement the storage and management of data in the system so that it can be efficiently developed. Applications are suitable for this system [5].

# 5. Results And Discussion Section

The characteristics of SQLite make it a good application in WinCE to achieve the storage and management of data. Below we will use Visual Studio2005 and SQLite database to connect with the mobile smart terminal on the WinCE platform and apply the database simply. Here are the steps for the application example:

1. Establish an MFC smart device application, import the library file generated in the previous migration process into the project and link accordingly.
2. The main application we implemented in the experiment is to implement a simple application of the smart device squadron database. Therefore, the design of the program is relatively simple. The MFC application we built uses the form of a dialog box. Therefore, consider the dialog box A button is added to call the corresponding database operation. Three buttons are added to create a database, create a table, and a query table, and a text box is used to display the contents of the table.
3. SQLite in this experiment is an ultra-lightweight open source database. The source downloaded from the official website (http://www.sqlite.org/) is a file written in C, but it is inevitable when called in C++ There will be some problems. Based on this, this experiment uses CppSQLite3. CppSQLite3 is a C++ class library after the second package of SQLite. The download address of CppSQLite3 is https://github.com/lmmir/CppSQLite3.

The code of the corresponding control in this experiment is as follows:

```
void CSQLiteDlg::OnBnClickedButton1()

{

    CppSQLite3DB db;

    db.open("\\D:\\Work\\4\\WinCEAPI\\SQLiteTest\\SQLite1\\test.db");

    messageBox(_T("SUCCESS"),_T("INFORMATION"),MB_OK);
```

```
    db.close();

}

void CSQLiteDlg::OnBnClickedButton2()
{
    CppSQLite3DB db;
    db.open("\\D:\\Work\\4\\WinCEAPI\\SQLiteTest\\SQLite1\\test.db"");
    CppSQLite3Query q=db.exceQuery(L"create table student(sno int ,sname nvarchar(6),sage int)");
    q=db.exceQuery(L”insert into student values(1,'hello',20);
    messageBox(_T("SUCCESS"),_T("INFORMATION"),MB_OK);
    db.close();

}

void CSQLiteDlg::OnBnClickedButton3()
{
    CppSQLite3DB db;
    db.open("\\D:\\Work\\4\\WinCEAPI\\SQLiteTest\\SQLite1\\test.db"");
    CppSQLite3Query q=db.exceQuery(L" select * from student");
    CString strTemp;
    While(!q.cof())
    {
            strTemp.Format(_T("%s-%s-%s"),q.fieldValue(0),q.fieldValue(1),q.fieldValue(2));
```

```
            m_list.AddString(strTemp);
            q.nextRow();
        }
        q.finalize();
        db.close();

}
```

4. The application of the database is mainly realized by responding to the click event of the button control, and adding event response programs to different buttons, to apply the database according to different events during the operation. The results are shown in Figure 2:

# 6. Conclusions

The previous application mainly refers to the application of databases to smart mobile devices based on the WinCE platform, but technology must have its practical value to receive attention. In practical applications, the application of databases has received more and more attention, especially Driven by big data, the application, and processing of data are becoming increasingly important. Also, the development prospects of embedded programming have become broader, with a large gap. Therefore, the application of databases in embedded programming has become more and more important. SQLite has its unique characteristics in embedded programming. It is widely used in programming. In this regard, the predecessors have done a lot of research and attempts and achieved certain results in combination with hardware processing.

There are two main types of mobile terminal applications. One is high-performance, which is only applicable to some industries. The other is a low-end platform that performs simple data collection, processing, and interaction. With the development of computers, the performance of high-end equipment has continued to increase and the price has decreased, and low-end platforms have gradually been replaced by low-end handheld computers. The "high-end" part tends to the fields that require precision and rigor, such as military and medical, while the "low-end" is more to meet the needs of individuals, communities, some enterprises, and fields.

For example, it is applied to the functions of phone, information and address book of ordinary smartphones; some functions of the mobile business hall. These have achieved great results. Also, SQLite based on the WinCE platform has appeared in the design of related tour guide systems. For example, some people in China have conducted in-depth research on tour guides in the field of tourism

and found that their freedom and comprehensiveness in obtaining travel information have defects. Transplanted the SQLite embedded database application, and realized the optimization of the museum guide. Many other scholars have also carried out different levels of optimization research on tour guides. Also, many scholars have been involved in the research of this database synchronization system. It can be seen that, in practical applications, SQLite is very important in WinCE embedded development. [13-14]

# Abbreviations

Embedded Compact (CE)

Windows Embedded Compact (Win CE)

Original Equipment Manufacturer (OEM)

OEM Adaptation Layer (OAL)

User Interface (UI)

Operating System (OS)

Board Support Package (BSP)

Basic Input / Output System (BIOS)

Open Database Connectivity (ODBC)

Structured Query Language (SQL)

# Declarations

Availability of supporting data: We can provide the data.

### Competing interests

These no potential competing interests in our paper. And all authors have seen the manuscript and approved it to submit to your journal. We confirm that the content of the manuscript has not been published or submitted for publication elsewhere.


### Funding

This research was supported by the Department of Higher Education of the Ministry of Education of the People's Republic of China for its financial support for the cooperative education project of industry-university cooperation under the contract 201702038005.


### Author’s contributions

All authors take part in the discussion of the work described in this paper. These authors contributed equally to this work and should be considered co-first authors.

### Acknowledgments

The authors thank the editor and anonymous reviewers for their helpful comments and valuable suggestions.

### Author Details

**Yuanhui Yu:** associate professor, master tutor, received his M.S. degree from the University of Electronic Science and Technology of China in 2002 and worked at Jimei University. His research interests include image processing, intelligent information processing, mobile agent computing, and data fusion. contact address: 185 Yinjiang Rd., Jimei Zone, Computer Engineering College, Jimei University Xiamen, Fujian Province, 361021, P. R. China.

# References

1. Z. Wang, Y. L. Shen, C. C. Su, “Design of Embedded Data Acquisition Integrated System Based on SQLite Database”, IEEE TRANSACTIONS ON COMPUTERS, vol 69, issue 5, pp. 666-678, 2020.
2. Liu, T., Zhang, M., Zhu, J. et al. ACCP: adaptive congestion control protocol in named data networking based on deep learning. Neural Comput & Applic 31, 4675–4683 (2019).
3. Y. Shen, Y. J. Shen, Z. L. Shao, “An Efficient LSM-Tree-Based SQLite-Like Database Engine for Mobile Devices,” IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, vol. 38, issue 9, pp. 1635-1647, 2019.
4. Zhang, S. G. Hao, Q. X. Zhang, “Recovering SQLite data from fragmented flash pages,” Annals of Telecommunications, vol. 74, issue 7-8, pp. 451-460, 2019.
5. Partha Sarathi Banerjee, Satyendra Nath Mandal, Debashis De, Biswajit Maiti. RL-Sleep: Temperature Adaptive Sleep Scheduling using Reinforcement Learning for Sustainable Connectivity in Wireless Sensor Networks. Sustain. Comput. Informatics Syst. 26: 100380 (2020)
6. Qureshi, K.N., Bashir, F. & Abdullah, A.H. Distance and signal quality aware next hop selection routing protocol for vehicular ad hoc networks. Neural Comput & Applic 32, 2351–2364 (2020).
7. Lee Suchul, Lee Sungil, Lee Jun-Rak, “SPaRe: Efficient SQLite Recovery Using Database Schema Patterns,” KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS, vol. 11, issue 3, pp. 1557-1569, 2017.
8. L. Liu, T. Mi, “Design and Implementation of Embedded Home Server Based on SQLite in Smart Home System,” 2017 4TH INTERNATIONAL CONFERENCE ON INFORMATION SCIENCE AND CONTROL ENGINEERING (ICISCE), pp. 942-946, 2017.
9. Lee Sungkwang, Lee Taemin, Park Hyunsun, “Differential Write-Conscious Software Design on Phase-Change Memory: An SQLite Case Study,” ACM TRANSACTIONS ON DESIGN AUTOMATION OF

ELECTRONIC SYSTEMS, vol. 21, issue 3,pp. 333-339, 2016.

10. Jain Vineeta, Gaur M.S., Laxmi Vijay, “Detection of SQLite Database Vulnerabilities in Android Apps,” Lecture Notes in Computer Science, vol. 10063, pp. 522-531, 2016.
11. T. Zhao, Z. Q. Wei; Y. Q. Yang, “Research on SQLite Database Encryption Technology in Instant Messaging Based on Android Platform,” Lecture Notes in Computer Science, vol. 9243, pp.316-325, 2015.
12. Kang Johyeon, Kim Jinho, “Enhancing the performance of Currency Control in SQLite Database Management System Utilizing Lightweight Locking Method,” The Journal of Information Technology and Architecture, vol. 12, issue 2, pp. 249-258, 2015.
13. Park, Sanghyun, “A Novel Recovery Scheme for SQLite Based on Logical Logging,” Journal of Korean Institute of Information Technology, vol. 12, issue 11, pp. 181-192, 2014.
14. Xie, S. J. Song, “Design of Door Monitoring Interface Based on Qt4 and SQLite,” Applied Mechanics and Materials, vol. 556-562, pp.1592-1596, 2014.

# Figures

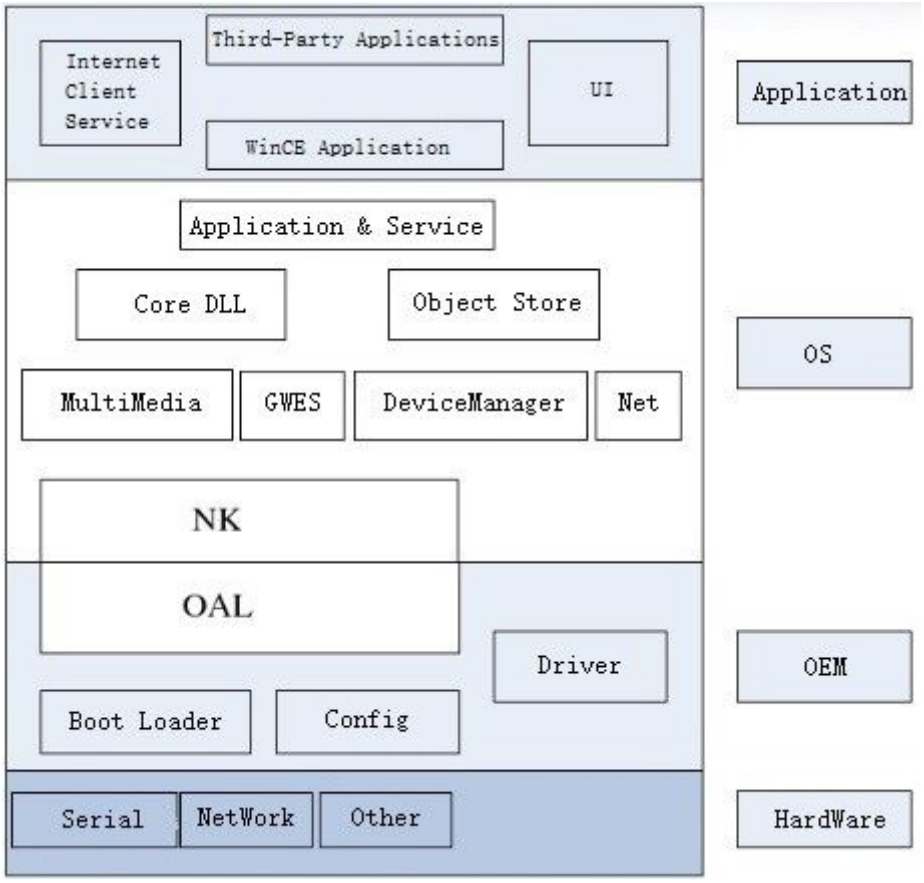


**Figure 1**

WinCE System Architecture


## Figure 2

Effect