1. Originality of application system

Travel is a great choice for people's leisure and entertainment. In the face of the influx of tourists, tourism groups need to consider not only the economic benefits, but also the quality of tourism services. At the same time, due to the impact of COVID-19, the park needs to take measures to prevent the epidemic, such as limiting the number of visitors.

Through the study of the specialty courses of Internet of Things, especially the practical courses of Internet of Things, we propose a park ticket checking system based on RFID technology. The system uses passive RFID tickets to realize automatic ticket checking. This system can not only greatly improve the passing ability of tourists per unit time at the entrance, greatly reduce the waiting time of tourists, improve the quality of tourism services, but also save the manpower of the park. As the reading distance of passive RFID is more than 3m, automatic identification of visitors can be realized without taking out the ticket. RFID tickets are not only convenient for tourists, but also reduce person-to-person and person-to-thing contact during the epidemic. At the same time, the system can meet the requirement of the scenic spot that one ticket is used only once through the simple tag reading and writing. The system can also realize the function of counting the number of visitors and collecting real-time statistics of the number of tourists in and out. When the number of

tourists in the scenic spot reaches the limit of the number of tourists, the system will automatically stop check-in, to realize the flow restriction during the epidemic.

2. Related technology

2.1 RFID Technology Overview

RFID (Radio Frequency Identification) technology is a non-contact information transmission through space coupling (alternating magnetic field or electromagnetic field) using radio frequency signals and identification through the transmitted information Technology. The main core component of RFID is an electronic tag with a diameter of less than 2mm. Through the radio waves emitted by sensors within a distance of several centimeters to tens of meters, the information stored in the electronic tag can be read and the identity of the articles, people and appliances represented by the electronic tag can be identified.

2.2 The Basic Components of RFID

The basic hardware architecture of RFID system consists of tag, reader, antenna, etc.

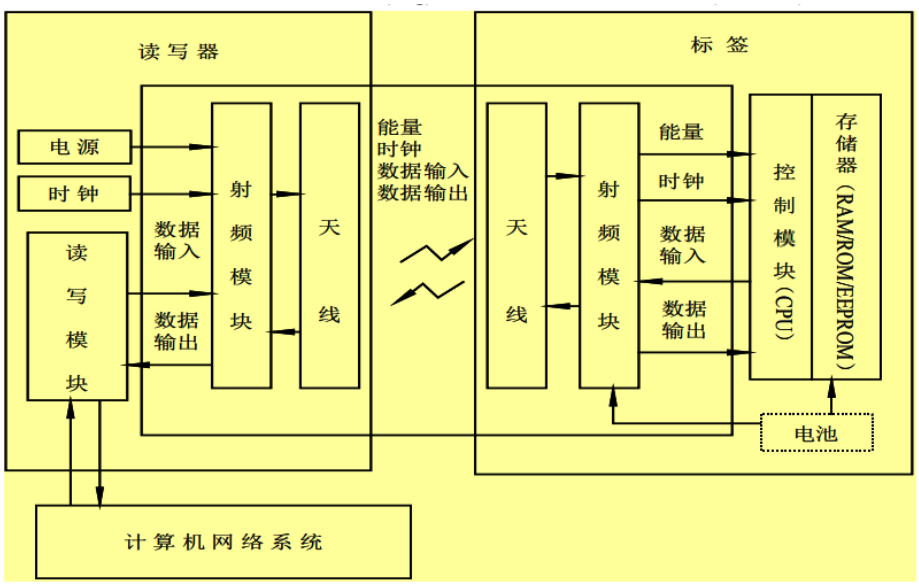

Figure. RFID System Structure

2.2.1 Tag

Tag has a certain capacity of memory for storing the information of the recognized object. The tag can be read and written under certain working environment and technical conditions. With a definite service life, no integrity of the identification and related information of the identified objects. After the data information is encoded, it can be transmitted to the reader.

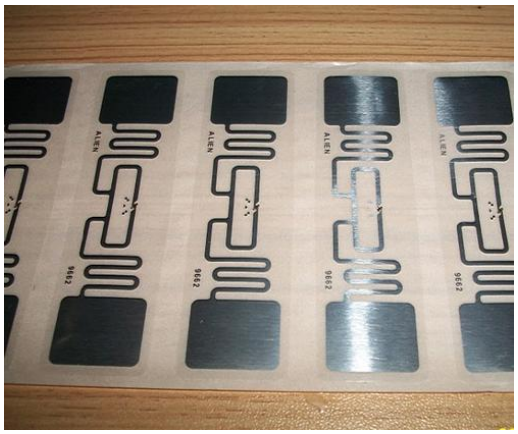

Figure. Tag

2.2.2 Reader

An RFID reader is a device that reads or writes tag information. Its task is to control the radio frequency transceiver to transmit radio frequency signals. Reader receives the encoded RF signal from the tag via the RF transceiver and decodes the tag's authentication and identification information. Reader transmits the authentication identification information along with other relevant information on the tag to the host for processing.

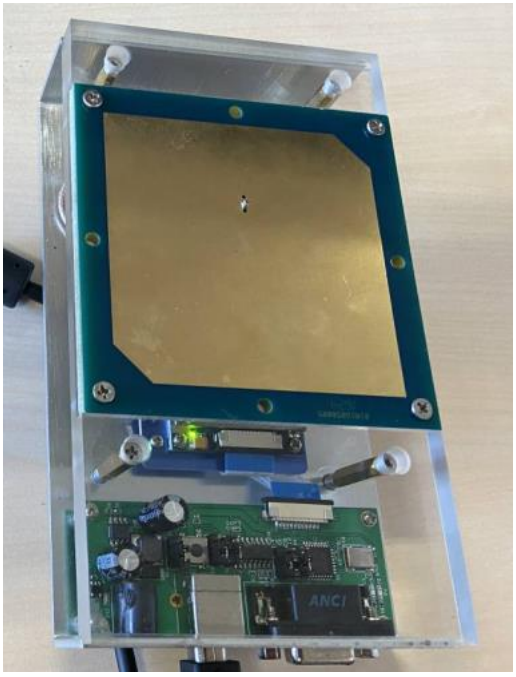

Figure. Reader

2.2.3 Antenna

The antenna transmits radio frequency signals between the tag and the reader. Any RFID system should contain at least one antenna (either

internal or external) to transmit and receive RF signals. Some of RFID systems use an antenna to transmit and receive at the same time, Other RFID systems are transmitted by one antenna and received by another antenna, depending on the specific application of the form and number of antennas.

2.3 Basic Working Principle of RFID Technology

The data transmission between the tag and the reader is carried out in the form of radio waves through an air medium.

1. The reader and writer transmit the radio carrier signal of the set data outward through the transmitting antenna.
2. When the RF tag enters the working area of the transmitting antenna, the RF tag will transmit its information code through the antenna after being activated.
3. The receiving antenna of the system receives the carrier signal from the RF tag and transmits it to the reader and writer through the modulator of the antenna. The reader decodes the received signal and sends it to the backstage computer controller.
4. According to the logic operation, the computer controller judges the validity of the RF tag, makes corresponding processing and control for different settings, and sends out command signals to control the action of the actuator.

5. The actuator acts according to the instructions of the computer.

6. Through the computer communication network, each monitoring point is connected to form the general control information platform, and different software is designed according to different projects to complete the functions to be achieved.

2.4 RFID Based Ticket System

RFID technology has the characteristics of easy and fast reading, fast identification speed, long service life (satisfied longevity), dynamic change of label data, dynamic real-time communication and so on. These characteristics make the RFID-based park ticket checking system have obvious advantages over the traditional ticket checking system.

3. System function

3.1 Requirement Analysis

The park usually uses the manual ticket checking method to check the ticket. This method is less efficient. Tourists have long waiting time and the scenic spot waste manpower. With the development of Internet of Things technology, there are more and more applications of Internet of Things. The application of Internet of Things technology in park ticket checking system not only provide better service for tourists, but also save resources for parks. RFID technology is a non-contact information

transmission through space coupling (alternating magnetic field or electromagnetic field) using radio frequency signals and identification through the transmitted information Technology. Applying RFID technology to the park ticket checking system can realize automatic ticket checking and improve the economic benefits and the quality of tourism services.

At the same time, the park ticket checking system also requires to be able to prevent multiple people to take turns to use one RFID ticket evasive behavior.

Currently, Covid-19 pandemic has changed our lives. Covid-19 disease is an acute respiratory infectious disease. It spreads through droplets and close contact. There is currently no effective treatment for Covid-19 disease, so we should pay attention to prevent infection. Parks are places where tourists congregate in large numbers. In order to reduce the gathering of tourists, the scenic spot needs to strictly limit the number of tourists. The park ticket checking system also requires that the number of tourists in the park can be automatically controlled. When the number of tourists reaches the limit set by the park, the system will automatically stop check-in to ensure that the number of tourists in the park is always within the specified limit.

3.2 Function Description

The system uses passive RFID technology to realize automatic ticket checking. Visitors use RFID tickets and set up readers at the entrance to the park. Reader will automatically read the RFID ticket information as visitors enter and exit to verify the validity of the ticket. After the ticket has been used once, the information on the ticket is modified, the ticket is invalid and cannot be used again.

The system can also set the upper limit of the number of tourists and collect real-time statistics of the number of tourists in and out. When the number of tourists in the park reaches the upper limit, the system will automatically stop the check-in and realize the flow restriction during the epidemic period.

4. Design and Implementation of system

4.1 Overall Design

The Java based system control the ticket gate, which control the on/off and calculate the actual inside number of tourists. There is a reader in the ticket gate, reading the tag and check the ticket. Whenever a ticket has been checked, its value would be modified that can only be used to exit. The counting number will be displayed whenever the number changed.

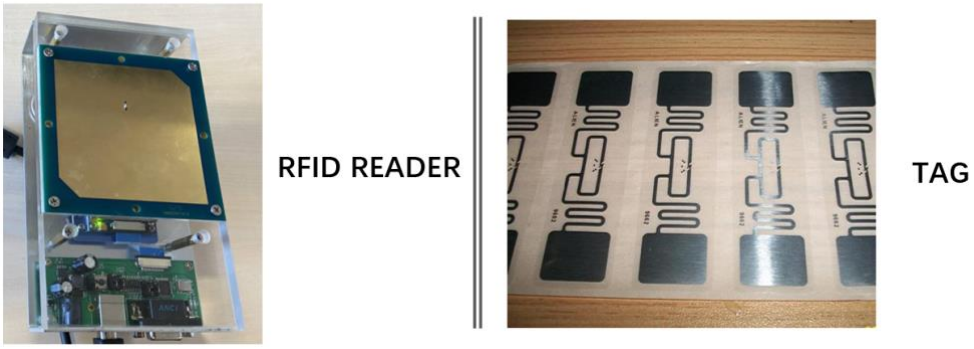

Figure. Overall Design

4.2 Hardware design

To sum up, the hardware we used in this system includes, passive tags, a reader, an USB-cable and a laptop.

RFID tag is composed of coupling element and chip, each RFID tag has unique electronic code, attached to the object to identify the target object. After the tag enters the magnetic field, it receives a radio frequency signal from the reader, and sends out the value stored in the chip based on the energy obtained by the induced current.

The application used RFID to interact with computer – COM4 is connected to the computer using the USB-cable, which acts as a reader/writer in this program. The reader receive data from tag, and process after analysis. The data can also be re-write to marked, presenting the validation of tickets. The connecting method is shown as above.

4.3 Software Design

Based on the object-oriented principle, we chose Java as our main programming language. The whole process of implement the remote control of RFID reader includes inventory test, read data, write data, close. Based on the basic program, we mainly encapsule them into well-round cycles. The cycle and if logic phases decided how tickets are read, modified and accepted, displaying current number of tourists in the park at the current stage. To make it visible, we printed every useful information whenever changes made.

Despite the software we designed, extra software also becomes useful in our programming procedure. We used “RadioTools” to inventory tags, read tags, write tags. This is mainly for comparation – checking the values and connection states when designing program.

4.4 Code Design

Main process-

Start from some initial steps: get inventory area, set inventory area, start testing and finishing the inventory step, everything is ready for the next processing steps. We embedded all the rest functions in a method, userReadSync, passing necessary variable and start counting. It is not until the end of inventory that the process keep recycling till reaching the maximum value.

```
while(true) {
    if (i == 0) {

        //System.out.println("connect success");
        getInventoryArea();// 盘点区域获取 getInventoryArea
        setInventoryArea();// 盘点区域设置 setInventoryArea
        startInventory();// 开始盘点测试 startInventory
        stopInventory();// 停止盘点测试 stopInventory

        while(count<MAX){
            int n=userReadSync(0, 1, count);        //READ THE TICKET
            count=n;
        }
    } else {
        System.out.println("connect failed");
    }
}
```

Access Gate-

The authentication step in this experiment is to verify the password.

After the authentication step, we guaranteed the basic reading method

unchanged.

```
//READ TICKET FUNCTION
public static int userReadSync(int position, int length, int count) {
    while(true){
        RwData rwData = new RwData();
        byte[] password = StringUtils.stringToByte("00000000");
        int status =
                Linkage.getInstance().readTagSync(password, 3, position, length, 3000, rwData);
        //添加循环验证，避免读取失败 Add loop validation to avoid read failure
        if (status == 0) {
            String result = "";
            String epc = "";
            if (rwData.status == 0) {
                if (rwData.rwDataLen > 0) {
                    result = StringUtils.byteToHexString(rwData.rwData,
                            rwData.rwDataLen);
                }
                if (rwData.epcLen > 0) {
                    epc = StringUtils
                        .byteToHexString(rwData.epc, rwData.epcLen);
                }
                int re = Integer.parseInt(result);
                if (re==0000){
                    userWriteSync(0,1,"1111");
                    count += 1;
                    System.out.println("Allow to Access.");
                    System.out.println("count="+count);
                }
                else{
                    System.out.println("Ticket is not available");
                }

                return count;
            }
        }
        //System.out.println("user read failed");
    }

}
```

We defined, unused ticket has a “0000” value in the tag, while used value contains “1111” and “2222” that the former one indicates the tourist is allowed to in but has not quitted, the latter one presents completely invalid tickets. Whenever one is allowed to in, the counter added 1 in the total number.

```
Allow to Access.
count=1
Ticket is not available
Ticket is not available

Allow to Access.
count=2
Allow to Access.
count=3
Ticket is not available
Ticket is not available
```

```
Ticket is not available
Allow to Access.
count=4
Ticket is not available
Ticket is not available

Allow to Access.
count=5
Ticket is not available
Ticket is not available
Ticket is not available

Allow to Access.
count=6
Ticket is not available
Ticket is not available
Ticket is not available
```

Exit Gate-

totally opposite to the accessing program. Whenever there's someone out of the park, the counter -1 the guarantee the accuracy. As mentioned above, we labeled 2222 to mark such state.

```
int re = Integer.parseInt(result);
if (re==1111){
    userWriteSync(0,1,"2222");
    count -= 1;
    System.out.println("Exit success.");
    System.out.println("count="+count);
}
else{
    System.out.println("Ticket is not available");
}
```

```
Exit success.
count=99
Ticket is not available
Ticket is not available
Ticket is not available
Ticket is not available
```

In terms of the public method we used, in JavaUhf, the device / terminal connection and the port / deinit port fall into the link category, and users only need to move the number associated with the port and call the low-contact method in the device library. During the process of readiness, reading, writing, etc., you need to call the communication

method, and finally the portable method of removing the device. The UhfDemo class has methods for listing, reading, writing, sorting / finding. startInventory retrieves basic data in counting mode. Therefore, in this example, the Start Inventory method uses a stylus to process the information received, calculate the time available, and wait for it to return to print the most recent information received.

*The complete source code can be viewed in the appendix.

4.5 Issues to Consider in System Deployment

4.5.1 Distribution of Reader Design

To begin with, we can further consider the reader position – whether one machine for in and another for out, or whether each of machine are able to handle in and out recording?

For the first possibility, if the check-in and check-out machine are set separately instead, the whole system would be better. As for the basic implementation, all codes in this system are posited in a certain packet. There will be a class that stores the static overall variable, recoding the total heads in the park. Two distinct classes that responsible for in and out separately, invoke the count variable by reading the static overall value. In this way, one is for in(add), while another is for out(minus). To be more specific, when tourists come in (the machine read the tag), the entrance's

gate open and add 1 to the count number, changing tag value to 1111, which represent the ticket has been used. On the contrary, when someone exit from another gate, the machine would remove the head from the total count, changing the tag value to 2222, labelling that the ticket is invalid any more.

For the other updating possibility, the checking machine can also be combined together. Each time, only one person allowed, or the count will easier get wrong. That is to say, to guaranteed that tag detected is belonged to one certain person, the algorithm should certainly do something in order to prevent from occurring convention.

4.5.2 Flow control

The reader is so sensitive that sometimes it can be a kind of disadvantage. As a case in point that the detected tag is just someone holding nearby, but they didn't use to enter, it will be a mistake. In order to solve such problem, the reader should not only limit the distance they identify, but also the number of tags they process. In the code, once a tag has been detected, rather than handle it directly, the algorithm should stop accepting new tourists and check the distance first. If the tag is in valid distance, then the machine does some processing steps, else, restart and accepting other tags.

4.5.3 To solve deadlock problem

One of the deadlock problems notices that both A and B are trying to modify the value of C memory, and they requested simultaneously. To solve such kind of deadlock problem, we can set a certain priority to them, or to set some deadlock protection. We distribute all request into threads- each need to be locked when attempt to modify the count value. After finishing modification, then unlock it. When each thread is queued, waiting for previous threads to be finished one by one, and follow the same locking order separately, we ensure no more than one thread is processing the count number.

4.5.4 Safety

Safety should be fully considered, in case that someone attempt to modify the tag value again and again, and even use it as profitable illegal business. In order to solve this, we do some protection either by further designing of code or use the RFID blocking mechanism.

The first one, code designing, has similar principle with that in internet accounts' password designing – the more active the password is, as well as the high random sequence with complicated combination, the more safety it would be. Thus, code for each tag could be even more complex (such as Ae7fp rather than 1111) and generated in random. Whenever the

new code generated, they would be stored in the database, and once read, they change actively again and again.

As for the RFID blocking, Usually, we can use RFID block cards (passive or active) to cooperated: passive shielding or protector can absorb or deflect RFID signals. Active RFID shields or protectors use microchips. They usually emit disturbing signals. Effectively pretend to be another card that causes the card reader to collide. Or it can consume the power of the transmission signals needed to power the chips in the card. FID block fabric is used to produce contactless block card. These are ultrafine cards that protect all contactless cards from the blue tooth, Wi-Fi, RFID and NFC, 3 and 4G cellular signals. The RFID block card protects and protects ID cards and payment cards from intrusion, theft and cloning by the most powerful RFID and NFC readers at 13.56Mhz and 125Khz.

5. Extension of the system

In the future, the system can also achieve the management and statistical analysis of tourists, using RFID to track and record the tourists in the entire park area in each area of the situation and consumption. The system can display the current number of people in each area in real time, locate the location of everyone at any time, and count the personnel in each area. At the same time, it is necessary to realize the entrance detection and alarm of person who does not have ticket, and to animate

the walking path of tourists.

Gates are set in each area of the park, and relevant information will be recorded when visitors pass through the gate. Once a tourist enters the system, he/she will be considered to be in the area until he/she passes through the gate again. Meanwhile, the ID and relevant information of the tourist will be recorded when he/she makes consumption in the area.

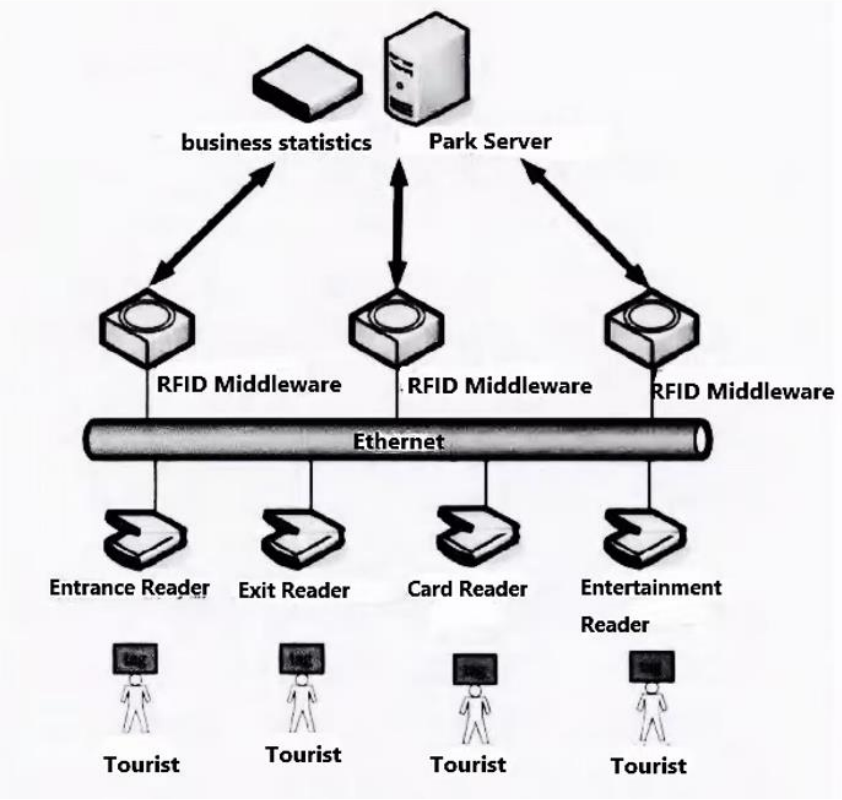

Figure. Extension of the System

6. Analysis of Internet of Things technology

The Internet of Things technology, as the name suggests, is the existence between software and hardware. Perhaps people have long discovered that the ultimate goal of pure computer research and scientific

considerations is to project into life, that is, to combine software and hardware and bring convenience to people's lives. Therefore, the emergence and popularity of the Internet of Things technology is the general trend. Common applications include a wider range of applications such as smart home, temperature and humidity monitoring, security systems, logistics processes, and agro-industry. Looking at the above, we can say that the essence of the Internet of Things technology is the interaction between software and sensors and monitors, quantifying the interaction information into digital information, and then processing on this basis to respond to human requests.

The Internet of Things technology has a wide range of applications, including wireless sensor networks, RFID, and information processing. Especially RFID, logistics information in online shopping cannot be separated from it, rapid detection cannot be separated from it during physical examination, and a series of security measures during the outbreak of new coronary pneumonia cannot be separated from it. In this system, RFID has been applied to the ticket checking system. Inspired by the pandemic of the new crown pneumonia virus in 2020. Not only in the park, but also in many scenes, it is necessary to limit the current to ensure the health of the crowd.

As students in the computer field, our eyes shouldn't just stay on the surface: function is just the realization of code, and we should consider

how to make it better. In recent years, human-computer interaction has become more and more closely related to the Internet of Things. Although this has not received strong attention in China, researchers need to pay more attention to how to bring more convenience to people based on the realization of functions. How to avoid a lot of problems? How to maintain safety? Just like the system we made, the experimental framework and code we built is just an empty shell. To truly become a product used in life, we must solve the problems in the previous chapter one by one and integrate them into it.

CODE – ACCESS

```
package com.uhf.demo;

import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.lang.*;
import com.uhf.detailwith.InventoryDetailWith;
import com.uhf.linkage.Linkage;
import com.uhf.structures.InventoryArea;
import com.uhf.structures.RwData;
import com.uhf.utils.StringUtils;



public class UhfDemo {
    @SuppressWarnings("static-access")
    public static void main(String[] args) {
        int i = Linkage.getInstance().initial("COM4");// 初始化连接设备,参数：端口号
        int count = 0;
        // function：init， parameter：The port number
        while(true) {
            if (i == 0) {

                //System.out.println("connect success");
                getInventoryArea();// 盘点区域获取 getInventoryArea
                setInventoryArea();// 盘点区域设置 setInventoryArea
                startInventory();// 开始盘点测试 startInventory
                stopInventory();// 停止盘点测试 stopInventory

                while(count<MAX){
                    int n=userReadSync(0, 1, count);            //READ THE TICKET
                    count=n;
                }
            } else {
                System.out.println("connect failed");
            }
        }
    }
    public static void userWriteSync(int position, int length, String data) {
        while(true){
            byte[] password = StringUtils.stringToByte("00000000");
            byte[] writeData = StringUtils.stringToByte(data);
            RwData rwData = new RwData();
```

```
            int status =
                    Linkage.getInstance().writeTagSync(password, 3, position, length,
writeData, 500, rwData);//调用 linkage 中的 user 写入函数 注意参数  Invoking the user
writing function in linkage and note the arguments
            //添加循环验证，避免读取失败 Add loop validation to avoid write failure
            if (status == 0) {
                if (rwData.status == 0) {
                    String epc = "";
                    if (rwData.epcLen > 0) {
                        epc = StringUtils
                                .byteToHexString(rwData.epc, rwData.epcLen);
                    }
                    //System.out.println("epc====" + epc);
                    //System.out.println("user write success");
                    return;
                }
            }
            //System.out.println("user write failed");
        }
    }

    //READ TICKET FUNCTION
    public static int userReadSync(int position, int length, int count) {
        while(true){
            RwData rwData = new RwData();
            byte[] password = StringUtils.stringToByte("00000000");
            int status =
                    Linkage.getInstance().readTagSync(password, 3, position, length, 3000,
rwData);//调用 linkage 中的 user 读取函数 注意参数  Invoking the user reading function in
linkage and note the arguments
            //添加循环验证，避免读取失败 Add loop validation to avoid read failure
            if (status == 0) {
                String result = "";
                String epc = "";
                if (rwData.status == 0) {
                    if (rwData.rwDataLen > 0) {
                        result = StringUtils.byteToHexString(rwData.rwData,
                                rwData.rwDataLen);
                    }
                    if (rwData.epcLen > 0) {
                            epc = StringUtils
                                .byteToHexString(rwData.epc, rwData.epcLen);
                    }
                    int re = Integer.parseInt(result);
```

```
                    if (re==0000){
                        userWriteSync(0,1,"1111");
                        count += 1;
                        System.out.println("Allow to Access.");
                        System.out.println("count="+count);
                    }
                    else{
                        System.out.println("Ticket is not available");
                    }

                    return count;
                }
            }
            //System.out.println("user read failed");
        }

    }

    public static void startInventory() {// 开始盘点 startInventory
        InventoryArea inventory = new InventoryArea();
        inventory.setValue(2, 0, 6);
        Linkage.getInstance().setInventoryArea(inventory);
        InventoryDetailWith.tagCount = 0;// 获取个数 Get the number
        Linkage.getInstance().startInventory(2, 0);
        InventoryDetailWith.startTime = System.currentTimeMillis();// 盘点的开始时间 Start
time of Inventory

        while (InventoryDetailWith.totalCount < 100) {

            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

        stopInventory();// 进行停止盘点 stopInventory

        /*for (Map<String, Object> _map : InventoryDetailWith.list) {
            System.out.println(_map);
            System.out.println("天线号(antennaPort)：" + _map.get("antennaPort"));
            System.out.println("epc 码：" + _map.get("epc"));
```

```
            System.out.println("TID/USER 码：" + _map.get("externalData"));
            System.out.println("次数(count)：" + _map.get("count"));
            System.out.println("Rssi：" + _map.get("rssi"));
        }*/

        long m_lEndTime = System.currentTimeMillis();// 当前时间 The current time
        double Rate = Math.ceil((InventoryDetailWith.tagCount * 1.0) * 1000
                / (m_lEndTime - InventoryDetailWith.startTime));

        long total_time = m_lEndTime - InventoryDetailWith.startTime;
        String dateStr = StringUtils.getTimeFromMillisecond(total_time);
        int tag = InventoryDetailWith.list.size();
        //System.out.println("盘点速率(Inventory rate)：" + Rate);

        /*if (tag != 0) {
            System.out.println("盘点时间(Inventory time)：" + dateStr);
        } else {
            System.out.println("盘点时间(Inventory time)：" + "0 时 0 分 0 秒 0 毫秒");
        }
        System.out.println("标签个数(The number of tag)：" + tag);*/

    }

    public static void stopInventory() {// 停止盘点 stopInventory
        Linkage.getInstance().stopInventory();
    }

    // 盘点区域获取 getInventoryArea
    public static void getInventoryArea() {
        InventoryArea inventoryArea = new InventoryArea();
        int status = Linkage.getInstance().getInventoryArea(inventoryArea);
        if (status == 0) {
            //System.out.println("area:" + inventoryArea.area);
            //System.out.println("startAddr:" + inventoryArea.startAddr);
            //System.out.println("wordLen:" + inventoryArea.wordLen);
            //System.out.println("getInventoryArea success");
            return;
        }
        //System.out.println("getInventoryArea failed");
    }

    // 盘点区域设置 setInventoryArea
    public static void setInventoryArea() {
        InventoryArea inventoryArea = new InventoryArea();
```

```
            inventoryArea.setValue(2, 0, 6);// 2 为 epc+user
            int status = Linkage.getInstance().setInventoryArea(inventoryArea);
            if (status == 0) {
                //System.out.println("setInventoryArea success");
                return;
            }
            //System.out.println("setInventoryArea failed");
        }

}
```

CODE – EXIT

```
package com.uhf.demo;

import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.lang.*;
import com.uhf.detailwith.InventoryDetailWith;
import com.uhf.linkage.Linkage;
import com.uhf.structures.InventoryArea;
import com.uhf.structures.RwData;
import com.uhf.utils.StringUtils;



public class UhfDemoExit {
    @SuppressWarnings("static-access")
    public static void main(String[] args) {
        int i = Linkage.getInstance().initial("COM4");// 初始化连接设备,参数：端口号
        int count = 100;
        // function：init， parameter：The port number
        while(true) {
            if (i == 0) {

                //System.out.println("connect success");
                getInventoryArea();// 盘点区域获取 getInventoryArea
                setInventoryArea();// 盘点区域设置 setInventoryArea
                startInventory();// 开始盘点测试 startInventory
                stopInventory();// 停止盘点测试 stopInventory

                while(true){
                int n=userReadSync(0, 1, count);            //READ THE TICKET
                count=n;
                }
            } else {
                System.out.println("connect failed");
            }
        }
    }
    public static void userWriteSync(int position, int length, String data) {
        while(true){
            byte[] password = StringUtils.stringToByte("00000000");
            byte[] writeData = StringUtils.stringToByte(data);
            RwData rwData = new RwData();
```

```
            int status =
                    Linkage.getInstance().writeTagSync(password, 3, position, length,
writeData, 500, rwData);//调用 linkage 中的 user 写入函数 注意参数  Invoking the user
writing function in linkage and note the arguments
            //添加循环验证，避免读取失败 Add loop validation to avoid write failure
            if (status == 0) {
                if (rwData.status == 0) {
                    String epc = "";
                    if (rwData.epcLen > 0) {
                        epc = StringUtils
                                .byteToHexString(rwData.epc, rwData.epcLen);
                    }
                    //System.out.println("epc====" + epc);
                    //System.out.println("user write success");
                    return;
                }
            }
            //System.out.println("user write failed");
        }
    }

    //READ TICKET FUNCTION
    public static int userReadSync(int position, int length, int count) {
        while(true){
            RwData rwData = new RwData();
            byte[] password = StringUtils.stringToByte("00000000");
            int status =
                    Linkage.getInstance().readTagSync(password, 3, position, length, 3000,
rwData);//调用 linkage 中的 user 读取函数 注意参数  Invoking the user reading function in
linkage and note the arguments
            //添加循环验证，避免读取失败 Add loop validation to avoid read failure
            if (status == 0) {
                String result = "";
                String epc = "";
                if (rwData.status == 0) {
                    if (rwData.rwDataLen > 0) {
                        result = StringUtils.byteToHexString(rwData.rwData,
                                rwData.rwDataLen);
                    }
                    if (rwData.epcLen > 0) {
                            epc = StringUtils
                                .byteToHexString(rwData.epc, rwData.epcLen);
                    }
                    int re = Integer.parseInt(result);
```

```
                    if (re==1111){
                        userWriteSync(0,1,"2222");
                        count -= 1;
                        System.out.println("Exit success.");
                        System.out.println("count="+count);
                    }
                    else{
                        System.out.println("Ticket is not available");
                    }

                    return count;
                }
            }
            //System.out.println("user read failed");
        }

    }

    public static void startInventory() {// 开始盘点 startInventory
        InventoryArea inventory = new InventoryArea();
        inventory.setValue(2, 0, 6);
        Linkage.getInstance().setInventoryArea(inventory);
        InventoryDetailWith.tagCount = 0;// 获取个数 Get the number
        Linkage.getInstance().startInventory(2, 0);
        InventoryDetailWith.startTime = System.currentTimeMillis();// 盘点的开始时间 Start
time of Inventory

        while (InventoryDetailWith.totalCount < 100) {

            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

        stopInventory();// 进行停止盘点 stopInventory

        /*for (Map<String, Object> _map : InventoryDetailWith.list) {
            System.out.println(_map);
            System.out.println("天线号(antennaPort)：" + _map.get("antennaPort"));
            System.out.println("epc 码：" + _map.get("epc"));
```

```
            System.out.println("TID/USER 码： " + _map.get("externalData"));
            System.out.println("次数(count)： " + _map.get("count"));
            System.out.println("Rssi： " + _map.get("rssi"));
        }*/

        long m_lEndTime = System.currentTimeMillis();// 当前时间 The current time
        double Rate = Math.ceil((InventoryDetailWith.tagCount * 1.0) * 1000
                / (m_lEndTime - InventoryDetailWith.startTime));

        long total_time = m_lEndTime - InventoryDetailWith.startTime;
        String dateStr = StringUtils.getTimeFromMillisecond(total_time);
        int tag = InventoryDetailWith.list.size();
        //System.out.println("盘点速率(Inventory rate)： " + Rate);

        /*if (tag != 0) {
            System.out.println("盘点时间(Inventory time)： " + dateStr);
        } else {
            System.out.println("盘点时间(Inventory time)： " + "0 时 0 分 0 秒 0 毫秒");
        }
        System.out.println("标签个数(The number of tag)： " + tag);*/

    }

    public static void stopInventory() {// 停止盘点 stopInventory
        Linkage.getInstance().stopInventory();
    }

    // 盘点区域获取 getInventoryArea
    public static void getInventoryArea() {
        InventoryArea inventoryArea = new InventoryArea();
        int status = Linkage.getInstance().getInventoryArea(inventoryArea);
        if (status == 0) {
            //System.out.println("area:" + inventoryArea.area);
            //System.out.println("startAddr:" + inventoryArea.startAddr);
            //System.out.println("wordLen:" + inventoryArea.wordLen);
            //System.out.println("getInventoryArea success");
            return;
        }
        //System.out.println("getInventoryArea failed");
    }

    // 盘点区域设置 setInventoryArea
    public static void setInventoryArea() {
        InventoryArea inventoryArea = new InventoryArea();
```

```
            inventoryArea.setValue(2, 0, 6);// 2 为 epc+user
            int status = Linkage.getInstance().setInventoryArea(inventoryArea);
            if (status == 0) {
                //System.out.println("setInventoryArea success");
                return;
            }
            //System.out.println("setInventoryArea failed");
        }

}
```