Kibo-RPC 日誌記錄與影像儲存教學

新增功能概述

這個版本的程式碼在基本範例的基礎上，新增了日誌記錄和影像儲存功能，這些功能對於程式除錯和開發過程非常重要。

完整程式碼

```java
package jp.jaxa.iss.kibo.rpc.sampleapk;

import jp.jaxa.iss.kibo.rpc.api.KiboRpcService;

import gov.nasa.arc.astrobee.types.Point;
import gov.nasa.arc.astrobee.types.Quaternion;

import org.opencv.core.Mat;

// new imports
import android.util.Log;

/**
 * Class meant to handle commands from the Ground Data System and execute them in
Astrobee.
 */

public class YourService extends KiboRpcService {

    // The TAG is used for logging.
    // You can use it to check the log in the Android Studio.
    private final String TAG = this.getClass().getSimpleName();

    @Override
    protected void runPlan1(){

        // Log the start of the mission.
        Log.i(TAG, "Start mission");
        // The mission starts.
        api.startMission();


        // Move to a point.
        Point point = new Point(10.9d, -9.92284d, 5.195d);
        Quaternion quaternion = new Quaternion(0f, 0f, -0.707f, 0.707f);
        api.moveTo(point, quaternion, false);

        // Get a camera image.
        Mat image = api.getMatNavCam();
        // Save the image to a file.
        api.saveMatImage(image, "test.png");
        /*
********************************************************************************
*/
        /* Write your code to recognize the type and number of landmark items in
each area! */
        /* If there is a treasure item, remember it.
*/
        /*
********************************************************************************
*/

        // When you recognize landmark items, let’s set the type and number.
        api.setAreaInfo(1, "item_name", 1);

        /* ***************************************************** */
        /* Let's move to each area and recognize the items. */
        /* ***************************************************** */

        // When you move to the front of the astronaut, report the rounding
completion.
        point = new Point(11.143d, -6.7607d, 4.9654d);
        quaternion = new Quaternion(0f, 0f, 0.707f, 0.707f);
        api.moveTo(point, quaternion, false);
        api.reportRoundingCompletion();

        /* *********************************************************** */
        /* Write your code to recognize which target item the astronaut has. */
        /* *********************************************************** */

        // Let's notify the astronaut when you recognize it.
        api.notifyRecognitionItem();

        /*
**********************************************************************************
********************* */
        /* Write your code to move Astrobee to the location of the target item
(what the astronaut is looking for) */
        /*
**********************************************************************************
********************* */

        // Take a snapshot of the target item.
        api.takeTargetItemSnapshot();
    }

    @Override
    protected void runPlan2(){
       // write your plan 2 here.
    }

    @Override
    protected void runPlan3(){
        // write your plan 3 here.
    }

    // You can add your method.
    private String yourMethod(){
        return "your method";
    }
}
```

新增的程式碼解析

1. 新增匯入

```
// new imports
import android.util.Log;
```

說明：

- 匯入 Android 的日誌記錄類別
- 用於在開發過程中記錄程式執行狀態和除錯資訊

2. 日誌標籤定義

```
// The TAG is used for logging.
// You can use it to check the log in the Android Studio.
private final String TAG = this.getClass().getSimpleName();
```

程式碼解析：

- `private final String TAG` - 宣告一個私有的常數字串作為日誌標籤
- `this.getClass().getSimpleName()` - 動態獲取當前類別的簡單名稱
- 在這個例子中，TAG 的值會是 "YourService"

用途說明：

- TAG 用於在 Android Studio 的 Logcat 中過濾和識別特定類別的日誌訊息
- 使用類別名稱作為標籤可以更容易追蹤程式執行流程

3. 任務開始日誌記錄

```
// Log the start of the mission.
Log.i(TAG, "Start mission");
// The mission starts.
api.startMission();
```

程式碼解析：

- `Log.i(TAG, "Start mission")` - 記錄一條資訊等級的日誌
- `Log.i()` 表示 Information level，是日誌的資訊等級
- 第一個參數是標籤（TAG），第二個參數是要記錄的訊息

日誌等級說明：

```
Log.d(TAG, "Debug message");    // Debug 等級
Log.i(TAG, "Info message");     // Information 等級
Log.w(TAG, "Warning message");  // Warning 等級
Log.e(TAG, "Error message");    // Error 等級
```

4. 影像儲存功能

```
// Get a camera image.
Mat image = api.getMatNavCam();
// Save the image to a file.
api.saveMatImage(image, "test.png");
```

程式碼解析：

- `api.getMatNavCam()` - 從導航相機獲取影像
- `api.saveMatImage(image, "test.png")` - 將影像儲存為檔案

影像儲存規格：

- 檔案格式：PNG
- 最大像素數：1,228,800 (1280 x 960)
- 每次模擬最多可儲存：50 張影像
- 儲存位置：Android 模擬器的 `/sdcard/data/` 目錄

實際應用建議

1. 日誌記錄最佳實務

```
// 記錄關鍵步驟
Log.i(TAG, "Moving to area 1");
api.moveTo(area1Point, area1Quaternion, false);
Log.i(TAG, "Arrived at area 1");

// 記錄識別結果
Log.i(TAG, "Found items: coin=2, shell=1");
api.setAreaInfo(1, "coin", 2);

// 記錄錯誤情況
if (image == null) {
    Log.e(TAG, "Failed to get camera image");
    return;
}
```

2. 系統化的影像儲存

```
// 為不同階段的影像命名
Mat image1 = api.getMatNavCam();
api.saveMatImage(image1, "area1_raw.png");

// 處理後的影像也可以儲存
Mat processedImage = processImage(image1);
api.saveMatImage(processedImage, "area1_processed.png");

// 目標物品影像
Mat targetImage = api.getMatNavCam();
api.saveMatImage(targetImage, "target_item.png");
```

3. 結合日誌和影像儲存的除錯策略

```
private void debugImageCapture(int areaId) {
    Log.i(TAG, "Capturing image for area " + areaId);

    Mat image = api.getMatNavCam();
    if (image != null) {
        String filename = "area_" + areaId + "_debug.png";
        api.saveMatImage(image, filename);
        Log.i(TAG, "Image saved: " + filename);
    } else {
        Log.e(TAG, "Failed to capture image for area " + areaId);
    }
}
```