

2nd Kibo Robot Programming Challenge Programming Manual



Version 1.1 (Release Date: April 28th, 2021)

Japan Aerospace Exploration Agency (JAXA)

General Point of Contact: Z-KRPC@ml.jaxa.jp

List of Changes

All changes to paragraphs, tables, and figures in this document are shown below;

Release Date	Revision	Paragraph(s)	Rationale
April 1 st , 2021	1.0	All	-
April 28 th , 2021	1.1	4.6.3. Setting up the Astrobeer Robot Software	Changed the reference of installation manual
		4.3. Uploading the APK and running your program	Added the contents about simulation conditions Updated figures
		4.5.1. result summary	Added the explanation about Astrobeer's task achievement Updated figures
		4.5.3. Check simulation after running	Updated figures
		4.6.3. Setting up the Astrobeer Robot Software	Updated the URL of NASA's installation manual
		5.9. Questions and information exchange	Added



Contents

1. Introduction	1
2. Setting up your machine.....	2
2.1. Requirements	2
2.2. Setting up Android Studio	2
3. Creating your application	4
3.1. Creating an Android project	4
3.2. Writing the application	5
3.3. Building your application.....	6
4. Running your program on the simulator	11
4.1. Using the simulator server	11
4.2. Login	12
4.3. Uploading the APK and running your program	13
4.4. Checking simulation while running	14
4.5. Checking the result.....	14
4.6. Running on your own machine (optional)	20
5. Programming tips	27
5.1. Do NOT write infinite loops	27
5.2. Dealing with randomness	28
5.3. About navigation errors.....	29
5.4. Attention to computing resources	29
5.5. Cautions when irradiating laser	29
5.6. Performance of Localization	29
5.7. Code review.....	30
5.8. Setting the application ID.....	30
5.9. Questions and information exchange	30
6. Simulator change log.....	31
7. Game API details.....	32
7.1. Method summary	32
7.2. Method details	33



1. Introduction

Let's start programming!

Astrobee can be controlled with an Android application called the Guest Science APK (GS APK). First, setup your machine to build your application according to the instructions in Chapter 2. Next, read Chapter 3 and create a GS APK. This chapter explains the game APIs that operate Astrobee, such as moving Astrobee and getting camera images. Then, try running the GS APK in the simulator environment. Chapter 4 describes how to use the environment.



2. Setting up your machine

First of all, set up a machine for programming.

2.1. Requirements

The machine must meet the following requirements.

- 64-bit processor
- 4 GB RAM (8 GB RAM recommended)
- Ubuntu 16.04 (64-bit version) (<http://releases.ubuntu.com/16.04/>)
or Windows 10 (64-bit version)

NOTE: If you want to run your program on your own PC, you need 8 GB of RAM (16 GB RAM recommended) and Ubuntu 16.04. For details, please refer to 4.6. Running on your own machine (optional).

2.2. Setting up Android Studio

2.2.1. Installing components (Only on Ubuntu)

If you use an Ubuntu machine, you need these components.

- openJDK8
- ADB (Android Debug Bridge)
- Gradle

Please install them with the following command.

```
sudo apt-get -y install openjdk-8-jdk adb gradle
```

2.2.2. Installing Android Studio

Please download Android Studio 3.4.1 from the Android Studio download archives page (<https://developer.android.com/studio/archive>) and extract it to your home directory.



2.2.3. Downloading additional components

To build the Guest Science APK, you need to download additional components as follows.

- 1) Launch Android Studio.
- 2) Select [Tools] -> [SDK Manager].
- 3) On the SDK Platforms tab, check "Show Package Details" and select "Android SDK Platform 25" and "Android SDK Platform 26."
- 4) On the SDK Tools Tab, check "Show Package Details" and select "25.0.3", "26.0.2" under Android SDK Build-Tools and "Android SDK Platform-Tools."
- 5) Click the [Apply] button to install these components.

3. Creating your application

3.1. Creating an Android project

To create your application, prepare a new project with the following steps.

- 1) Download the APK template (Template APK) from the download page on the web site.
 - 2) Extract the zip file to the directory where you want it.
 - 3) Launch Android Studio.
 - 4) Open the APK template folder with [File] -> [Open].
 - 5) Open [app/java/jp.jaxa.iss.kibo.rpc.defaultapk /YourService.java] in Project view.
 - 6) Write your code in runPlan1 – runPlan3 methods in the YourService.java file.
 - 7) (For the Final Round only) Replace the audio file [app/src/main/res/raw/astrobee.mp3] with the one you prepared to report the mission's completion. The file name must be astrobee.mp3. Please refer to 3.1.1. for sound file specifications
- * In the Preliminary Round, there is no need to prepare and replace the audio file.

When you open the APK template folder, the “Android Gradle Plugin Update Recommended” dialog may appear. However, you must not update the plugin because of a dependency problem, so select “Don’t remind me again for this project.”

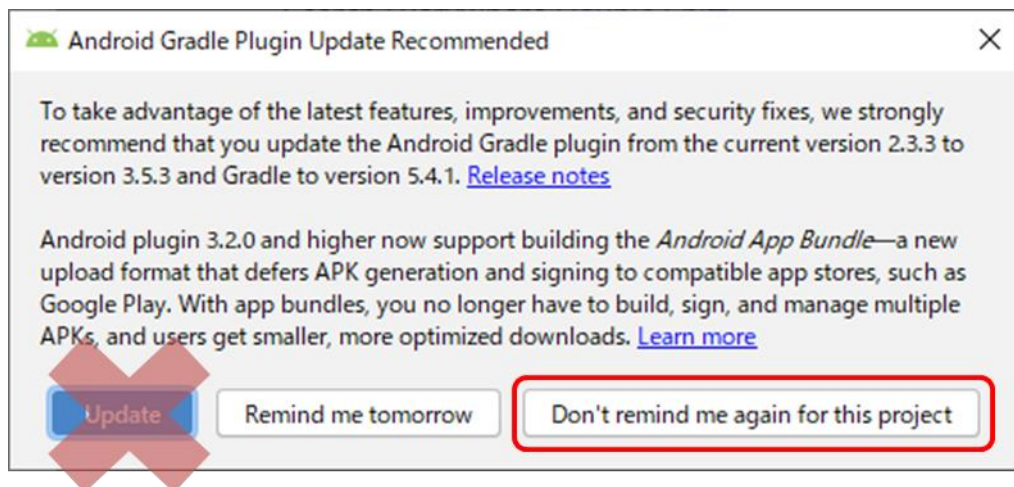


Figure 3.1.1 Android Gradle Plugin Update Recommended dialog

3.1.1. Audio file specifications

Note that the audio file you use should follow the format below.

Table 3-1 Format of a sound file

Format	MP3
Bit rate	8-320Kbps constant (CBR) or variable bit-rate (VBR)
Sampling rate	44.100kHz
Channel	Monaural
Sound Pressure Level	To Be Announced
Duration	5-15 seconds

3.2. Writing the application

You can use the game APIs shown below in the YourService.java file.

“runPlan1” is executed on the web simulator. You can choose any plan when you run the application on your own machine.

```
public class YourService extends KiboRpcService {
    // write your plan 1 here
    @Override
    protected void runPlan1(){

        // astrobee is undocked and the mission starts
        api.startMission();

        // move to Point-A
        Point point1 = new Point(11.21, -9.8, 4.79);
        Quaternion quaternion1 = new Quaternion(0f, 0f, 0f, 1f);
        api.moveTo(point1, quaternion1, true);

        (read QR, move to Point-A', read AR and aim the target)
        Bitmap image = api.getBitmapNavCam();
        String content = readQR(image);
        api.sendDiscoveredQR(content). // send the content of QR code for judge.
        ...
        ...

        // irradiate the laser
        api.laserControl(true);
    }
}
```




```
(evaluate the accuracy and retry aiming the target if necessary)
```

```
...
```

```
...
```

```
// take snapshots
```

```
// The laser accuracy is calculated from the snapshots taken by this API.
```

```
api.takeSnapshot();
```

```
// turn the laser off
```

```
api.laserControl(false);
```

```
// move to the rear of Bay7
```

```
Point point2 = new Point(10.275, -10.314, 4.295);
```

```
Quaternion quaternion2 = new Quaternion(0f, -0.7071068f, 0f, 0.7071068f);
```

```
api.moveTo(point2, quaternion2, true);
```

```
// Send mission completion
```

```
api.reportMissionCompletion();
```

```
}
```

```
protected void runPlan2() {
```

```
    // You can write your other plan here, but it's not run on the web simulator.
```

```
    // ...
```

```
}
```

```
}
```

You can find methods of the game APIs by using the code completion function of Android Studio.

Please refer to 7. Game API details for more information and you can download a sample APK from the download page on the web site.

3.3. Building your application

3.3.1. On Ubuntu

To build your application, use the command shown below.

NOTE: DO NOT build your application using Android Studio to change the build task, as this may cause an error.

```
$ cd <YOUR_APK_PATH>
```

```
$ ANDROID_HOME=$HOME/Android/Sdk ./gradlew assembleDebug
```

You can find the APK file here: “<YOUR_APK_PATH>/app/build/outputs/apk/app-debug.apk”.

3.3.2. On Windows

Please build your application with the following steps.

- 1) Launch the Android Studio.
- 2) Open <YOUR_APK_PATH>.
- 3) Click app on the [Project] window.
- 4) Select [Build] -> [Make Module ‘app’].

If you find errors, please build an APK file on an Ubuntu machine.

You can find the APK file here: “<YOUR_APK_PATH>\app\build\outputs\apk\app-debug.apk”.

3.3.3. Change the application ID (optional)

You can change the application ID (*jp.jaxa.iss.kibo.rpc.sampleapk* or *jp.jaxa.iss.kibo.rpc.defaultapk* by default).

In this step, we change the application ID to “jaxa.iss.kibo.rpc.myteam” and the APK name to “myteam” with the SampleAPK project.

NOTE: This instruction is for the Final Round. Changing the application ID is not necessary and not recommended in the Preliminary Round.

NOTE: “jaxa.iss.kibo.rpc” cannot be changed.

- 1) Launch Android Studio.
- 2) Open <YOUR_APK_PATH>.
- 3) Make sure you are viewing the project in Android View.

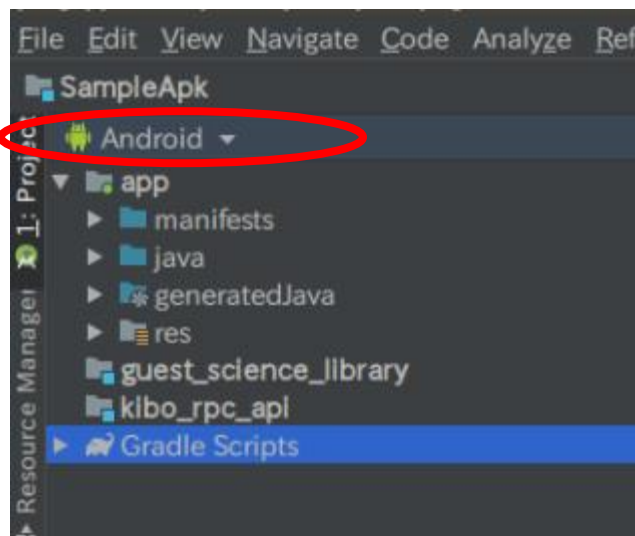


Figure 3.3.1 Android View

- 4) Click on settings (the gear icon) and deselect [Compact Empty Middle Package].

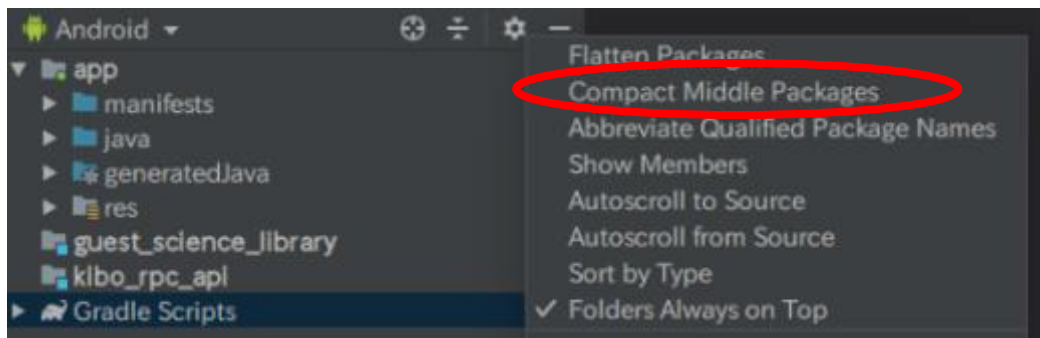


Figure 3.3.2 Unselect [Compact Empty Middle Package]

- 5) Please expand the "java" folder.

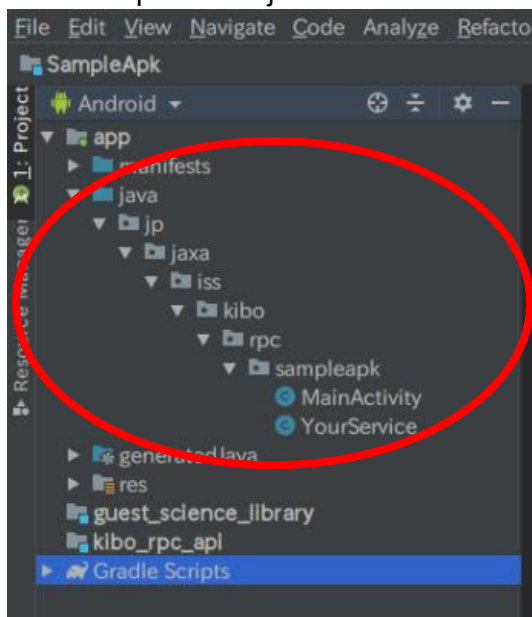


Figure 3.3.3 Expand the "java" folder

- 6) Right-click the "sampleapk" folder and select [refactor] -> [rename].
- 7) A warning will be displayed, but you want to go ahead and click [Rename Package]. After that, enter the APKname that you want. (In the picture, we rename as it "myteam".)

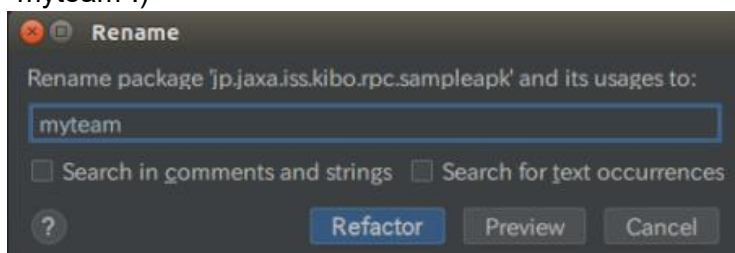


Figure 3.3.4 Rename dialog

- 8) At the bottom of Android Studio, “Refactoring Preview” will be displayed. Here, click [Do Refactor].

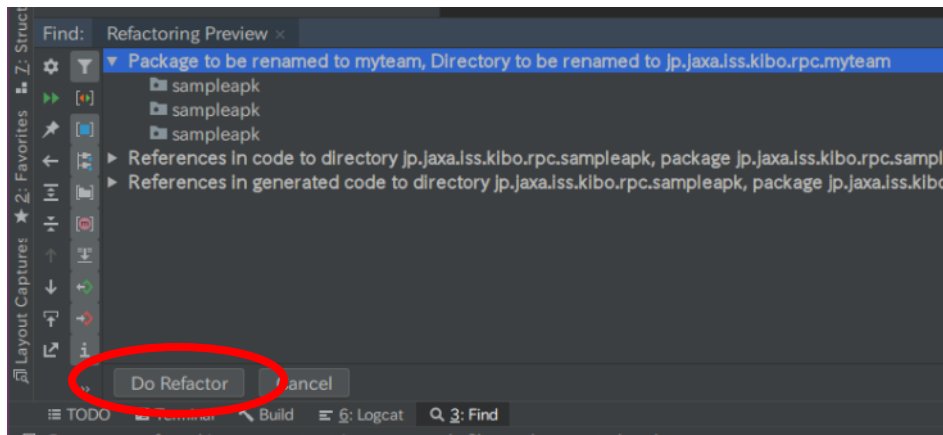


Figure 3.3.5 Refactoring Preview

- 9) Open **build.gradle (Module: app)** in Gradle Scripts on the left-side of the menu. Please change the application ID and click [Sync Now].

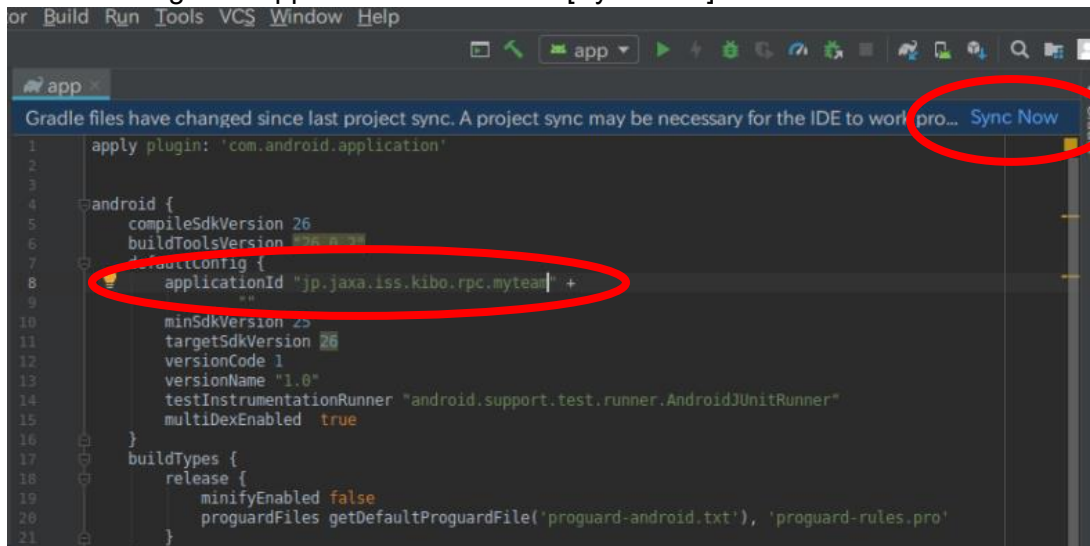


Figure 3.3.6 build.gradle (Module: app)

- 10) Open **strings.xml** in app -> res -> values on the left-side of the menu. Please change the APK name and save it.

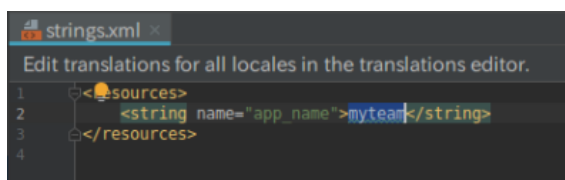


Figure 3.3.7 strings.xml

You have successfully changed the application ID in Android Studio. If you want to change the Android project name and its directory name, follow the following steps.



(1) On Ubuntu

- 11) Close Android Studio.
- 12) Please execute the following commands.

```
cd <YOUR_APK_PATH>
cd ../
mv SampleApk <YOUR_APK_NAME>
cd <YOUR_APK_NAME>
mv SampleApk.iml <YOUR_APK_NAME>.iml
```

(2) On Windows

- 10) Close the Android Studio.
- 11) Please rename a **SampleApk** folder to <YOUR_APK_NAME> with Windows Explorer.
- 12) Now rename **SampleApk.iml** to <YOUR_APK_NAME>.iml in the **SampleApk** folder with Windows Explorer.

4. Running your program on the simulator

4.1. Using the simulator server

Once you have built your application, you can run it on the web simulator provided by JAXA. To use the simulator, you need a user account issued by the Kibo-RPC secretariat. If you don't have one, please read the Kibo-RPC Guidebook to complete your application for participating in Kibo-RPC first.

Note that the actual Kibo environment is not exactly the same as the simulation environment, since there are many objects in the Kibo and the environment changes frequently. Please refer to the image and Google Map for the actual environment. (These are not the latest.)

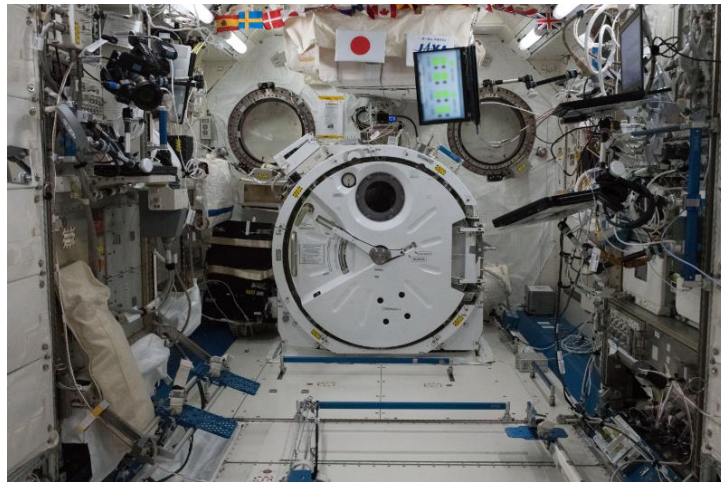


Figure 4.1.1 Image inside Kibo

Google Map: https://www.google.com/maps/@29.5604024,-95.0855631,2a,75y,205.79h,103.61t/data=!3m6!1e1!3m4!1sUA46_vlbk9kAAQvxgbyMg!2e0!7i10000!8i5000

Also, the views of Astrobees' NavCam and DockCam are different between the simulator and the real robot. Refer to the image below.

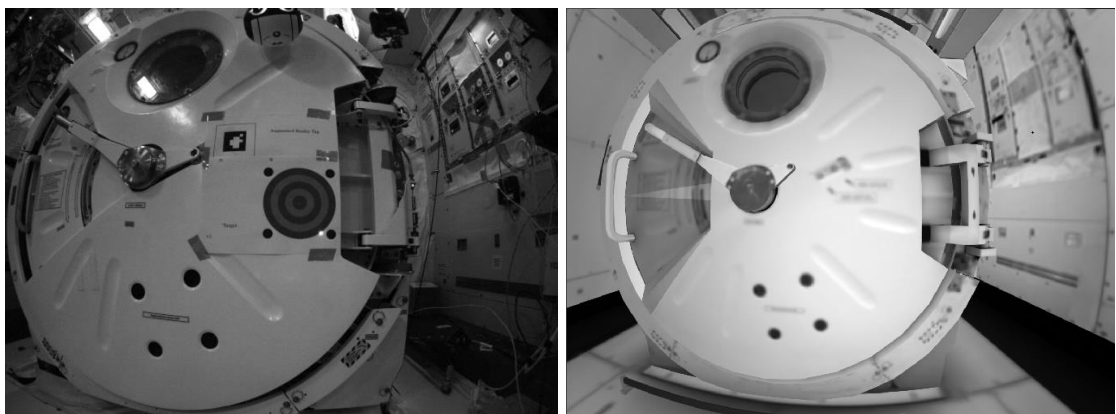


Figure 4.1.2 (Left) Actual snapshot (Right) Simulator image

4.2. Login

Access the Kibo-RPC web site (<https://jaxa.krpc.jp/>) and click “LOGIN.”



Figure 4.2.1 LOGIN tab

On the login page, enter the ID and password for your team’s account, and click the “LOGIN” button. If you have forgotten your ID, please contact the Kibo-RPC secretariat. You can reset your password by clicking the “Forgot your Password?” link.

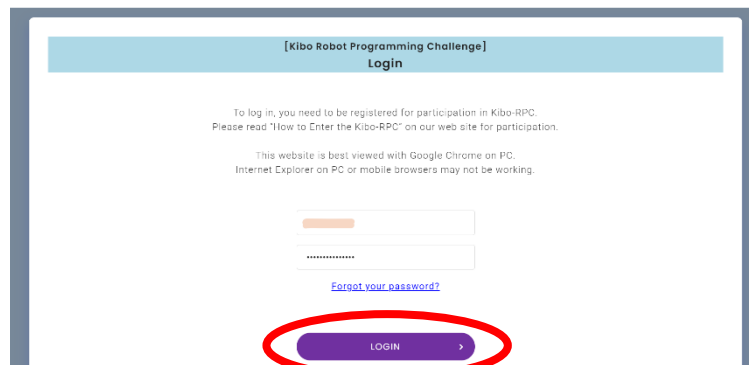


Figure 4.2.2 LOGIN button

On the home page, click “SIMULATION.”

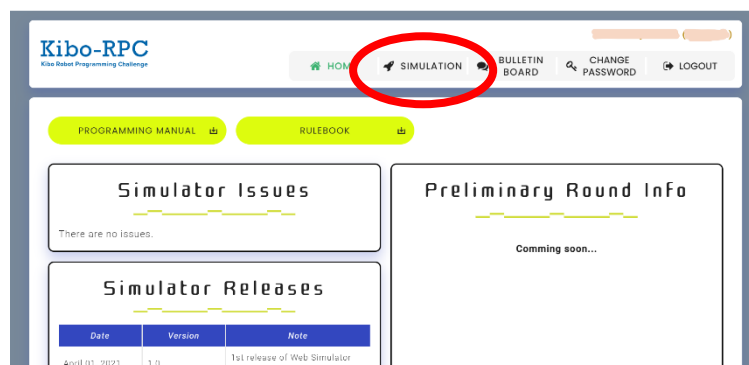


Figure 4.2.3 SIMULATION button

Now, you can access the web simulator from this page.

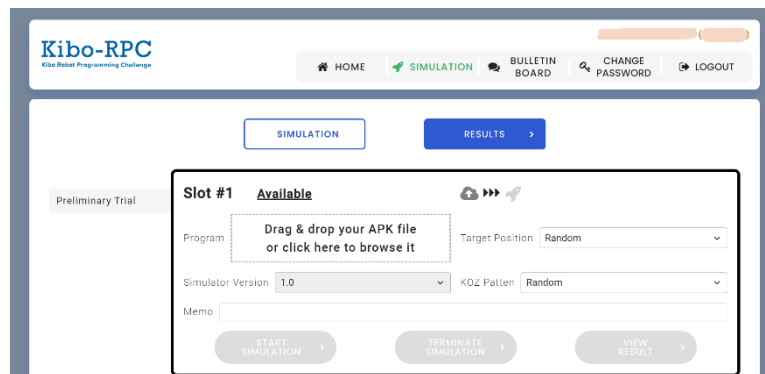


Figure 4.2.4 Web simulator page

4.3. Uploading the APK and running your program

On the simulation page, there are three slots for simulations, so that you can run at most three programs in parallel.

To start your simulation, select your APK file, the simulator version and the simulation conditions, enter a memo if desired, and click the “START SIMULATION” button.

There are two simulation conditions, the target position and the KOZ (Keep-Out Zone) pattern. You can choose “Random” or “Fixed” for the former, and “Random” or one of the eight KOZ patterns for the latter. When the target position is set to “Fixed”, the target is always put into a fixed place according to the KOZ pattern. For details of KOZ patterns, please refer to Rulebook.

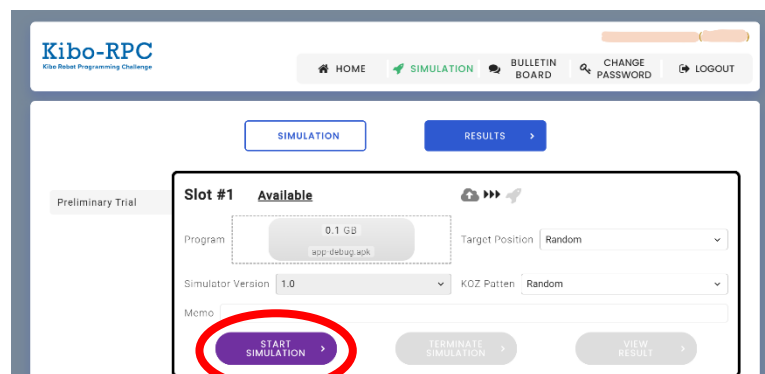


Figure 4.3.1 START SIMULATION button

A simulation may take longer than 20 minutes to run, and it does not need your attention while it runs. After starting your simulation, you can log out, get a cup of coffee, then go back to the web site.

When there is a simulation running, the slot displays its original information, and you cannot run another simulation in the same slot until it finishes.

If you want to stop your simulation, click the “TERMINATE SIMULATION” button. Note that terminating a simulation loses its game score and output files (such as rosbag and the Android Emulator’s log).

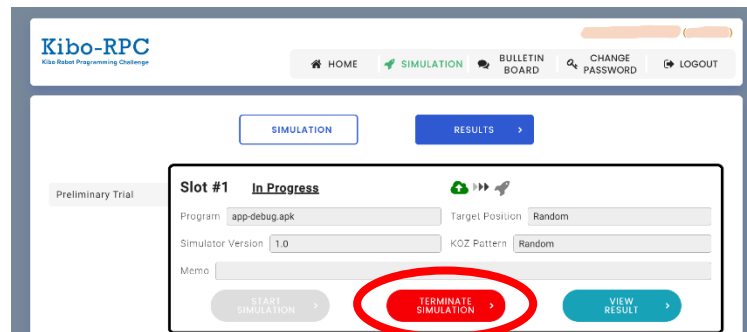


Figure 4.3.2 TERMINATE SIMULATION button

4.4. Checking simulation while running

When your simulation is running, you can log in to the simulator server (viewer) via your browser. Click the “SIMULATOR VIEWER” button to show the information for a remote connection and open the viewer in another tab by clicking the “VIEW” button.

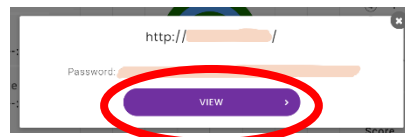


Figure 4.4.1 VIEW button

Enter the password for your remote connection to log in. Now you can use rviz to see how Astrobee moves in your simulation. This viewer is available until the simulation is finished.

The viewer displays a real-time simulation in the view-only mode for the simulation stability. You cannot operate the viewer.

4.5. Checking the result

4.5.1. Result summary

Once your simulation has started, you can check the results by clicking the “VIEW RESULT” button on the simulation page.

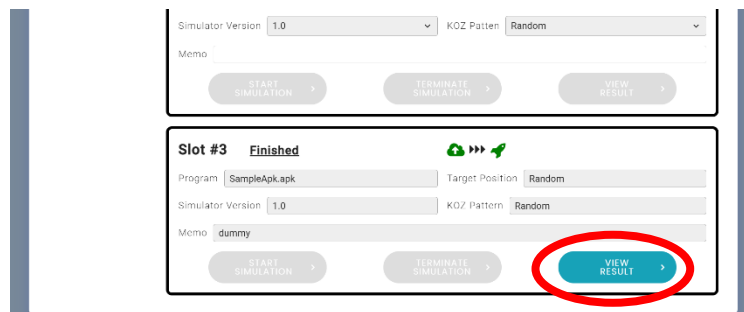


Figure 4.5.1 VIEW RESULT button

On the result page, you can see the details of your simulation, such as the game time, laser accuracy, and so on.

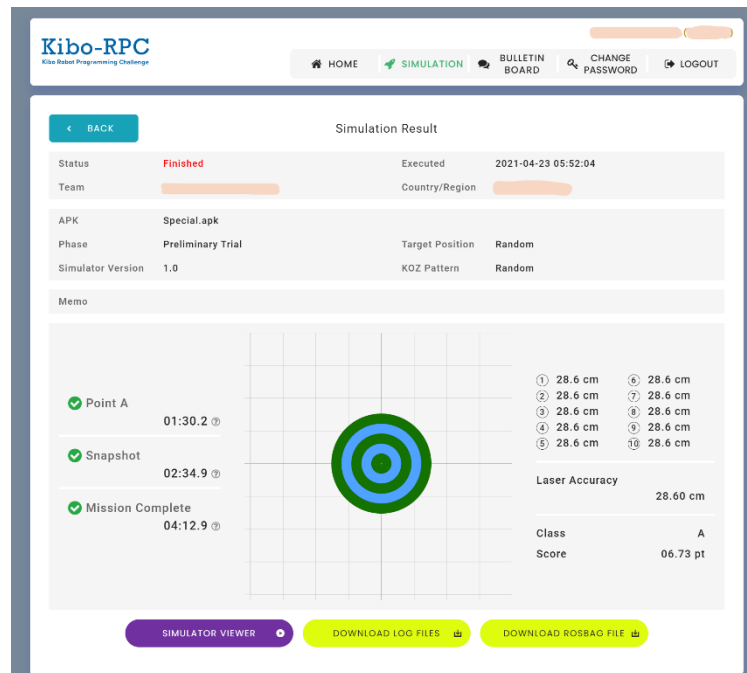


Figure 4.5.2 RESULT page

On the left of the target figure, Astrobees task achievement is displayed.

Table 4-1 Task achievement

Point A	Green check	Astrobees read the QR code at Point-A and got the correct information of Point-A'.
	Red cross	Astrobees read the QR code at Point-A, but got a wrong information of Point-A'.
	Gray minus	Astrobees did not read the QR code.
Snapshot	Green check	Astrobees took ten snapshots and all the laser shots were on the target plane.
	Red cross	Astrobees took ten snapshots, but not all the laser shots were on the target plane.
	Gray minus	Astrobees did not take ten snapshots.
Mission Complete	Green check	Astrobees executed reportMissionCompletion API at the correct position.
	Red cross	Astrobees executed reportMissionCompletion API at a wrong position.
	Gray minus	Astrobees did not try to execute reportMissionCompletion API.

4.5.2. Download ZIP file

You can get a ZIP file by clicking the “DOWNLOAD LOG FILES” button. This ZIP file contains the game score and the Android Emulator’s log. Note that some or all of these files will not be available unless your simulation finishes properly. Besides the result page, the game score also appears in a JSON file, which can be read using a text editor.

If the audio playback fails, “*Internal error has occurred. Unable to play sound*” will be logged to the Android Emulator’s log (adb.log). If no exception is shown in the log, the audio playback was successful.

Table 4-2 Example of result.json

```
{
  "Mission Time": {
    "start": "19700101 000021632",
    "finish": "19700101 000822824"
  },
  "QR": {
    "0": {
      "result": true,
      "timestamp": "19700101 000423792",
      :
    },
    :
  },
  "Approach": {
    "0": {
      "direction": true,
      "x": 1.22,
      "y": -3.44,
      "r": 4.12,
      "timestamp": "19700101 000646960",
      :
    },
    :
  },
  "Report": {
    "try": true,
    "success": true
  },
  "illegal": false
}
```

“Mission Time” is the difference between the “start” time and the “finish” time.

“result” is true if the value of the QR code is correct.

“direction” is true if the laser shot is on the Target plane.

“r” is the distance between the center of Target and the laser shot.

“0”, “1”, ... and “9” correspond to the 1st, 2nd, ... and 10th snapshot.

The average distance is referred to as “Laser Accuracy”.

“try” is true if you execute reportMissionCompletion API.

“success” is true if you execute reportMissionCompletion API at the correct position.

“illegal” is for server internal use.

You can also get a rosbag as a ZIP file by clicking the “DOWNLOAD ROSBAG FILE” button. The size of this file will be so large that it may take a long time to download it.

4.5.3. Check simulation after running

To check previous simulations, click the “Results” button on the simulation page. The results page lists your past simulations. This list can hold up to 20 simulations.

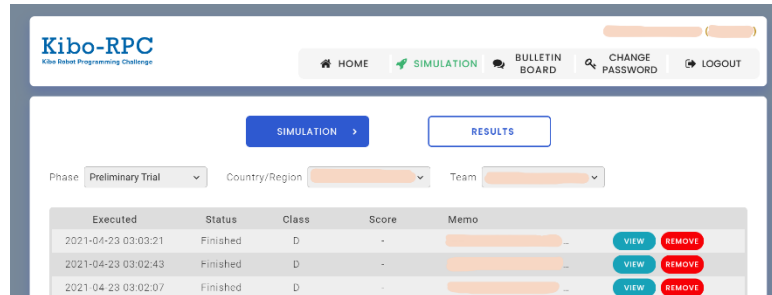


Figure 4.5.3 Results list page

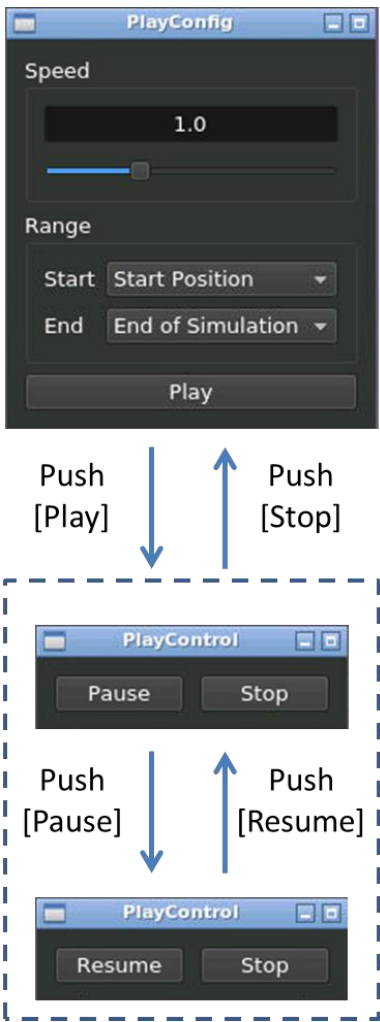
The “VIEW RESULT” button is the same as the one on the simulation page. Please be careful when you click the “REMOVE RESULT” button; it removes the output files of the selected simulation and these results will be lost.

You can play the rosbag (simulation result) at 0.5x – 3x speed with the viewer. You can change rosbag replay settings and rviz settings. The details are, described in the sections below.

4.5.4. rosbag replay settings

You can change rosbag replay settings using Rosbag Player.

Figure 4.5.4 Rosbag Player



Type	Description
Speed Slider	Select replay speed.
Range Selector	Select replay range.
Play Button	Start replay and open rviz window. If rviz already has opened, it will restart.
Pause Button	Pause replay.
Resume Button	Resume replay.
Stop Button	Stop replay and back to PlayConfig window.

4.5.5. rviz settings

You can change the display settings for the rviz window.

Table 4-3 rviz configuration

Item	Check box in the “Displays” tab
Planning trajectory	[Visualize]->[PlanningTrajectory]
Trajectory	[Visualize]->[Trajectory]
KeepInZone/KeepOutZone	[Visualize]->[Zones]
NavCam	[Sensors]->[NavCam]
DockCam	[Sensors]->[DockCam]
HazCam	[Sensors]->[HazCam]

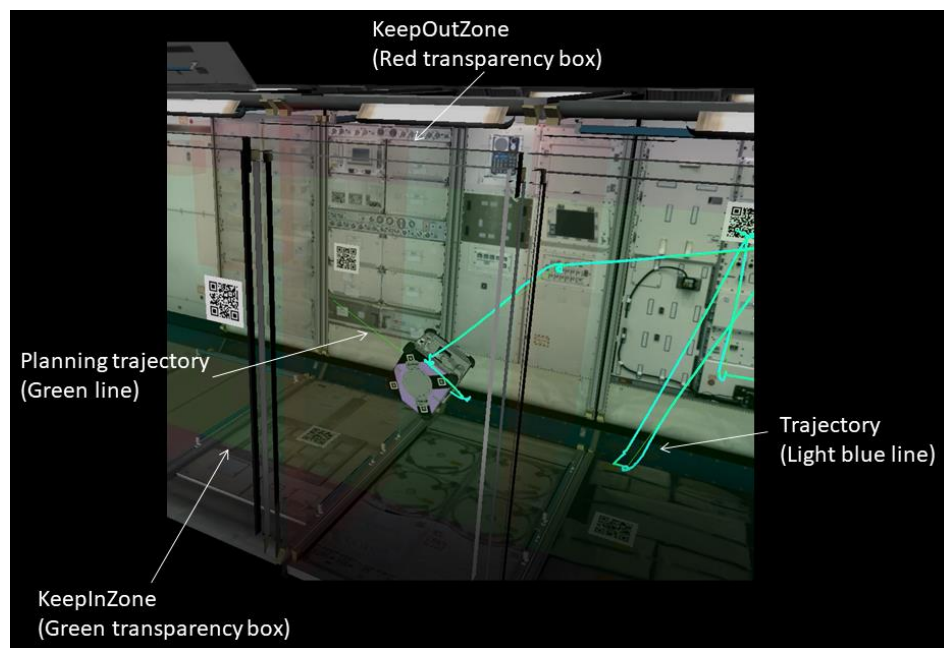


Figure 4.5.5 rviz configuration description



4.6. Running on your own machine (optional)

You can also run the program on your own machine. This chapter provides a procedure to set up the Astrobeer Simulator. You get a simple simulation environment without randomness modules (Air flow simulator and Object's randomness generator) or a judge a module.

4.6.1. Differences between web simulator and local simulator

External Modules for Local Simulation Environment do not include random factor modules (object position, airflow, and navigation error).

You can test and debug your program using a local simulator, but you need to evaluate it on a web simulator server in order to obtain a high score in the Preliminary Round.

4.6.2. Requirements

The following requirements are needed to set up a simulation environment on your machine.

- 64-bit processor
- 8 GB RAM (16 GB RAM recommended)
- Ubuntu 16.04 (64-bit version) (<http://releases.ubuntu.com/16.04/>)

4.6.3. Setting up the Astrobeer Robot Software

Clone code from GitHub (<https://github.com/nasa/astrobeer>) and install Astrobeer Robot Software according to the installation manual. (<https://nasa.github.io/astrobeer/html/install-nonNASA.html>)

NOTE: Since the web simulator runs Astrobeer Robot Software v0.13.0 and Android submodule v0.8.0, please execute the following command after cloning the Android software repository.

```
pushd $SOURCE_PATH
git checkout c8f3da73c5a4b3cff09f1b9095647f0dab795372
popd
pushd $ANDROID_PATH
git checkout 5b07e4d626781a6f7e0a9cdf4397375cbe509803
popd
```

After building the source code, execute the following commands in order.

```
pushd $BUILD_PATH
source devel/setup.bash
popd
roslaunch astrobee sim.launch dds:=false robot:=sim_pub rviz:=true
```

Is the image below displayed on your screen? If so, installation is complete!

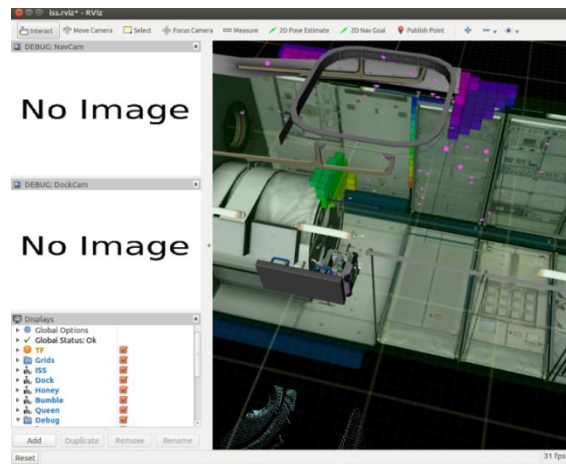


Figure 4.6.1 Setup result

4.6.4. Creating the AVD (Android Virtual Device)

Create an AVD (Android Virtual Device) as follows.

- 1) Launch Android Studio.
- 2) Select [Tools] -> [AVDManager].
- 3) In the Android Virtual Device Manager window, click [+ Create Virtual Device ...].
- 4) Select device **Nexus 5** (Resolution 1080x1920) and click [Next].
- 5) Select the [x86 Images] tab, choose **Nougat/API Level 25/ABI x86_64/Android 7.1.1(NO Google APIs)**, then click [Next].

NOTE: Download the system image now if you need it.

- 6) Set the AVD name to “AstrobeeAndroidSim.”
- 7) Click [Finish].

In the Android Virtual Device Manager window, you will see “AstrobeeAndroidSim” in the list.

Click the Play button in the Action column. If the AVD launches successfully, you will see the following image.

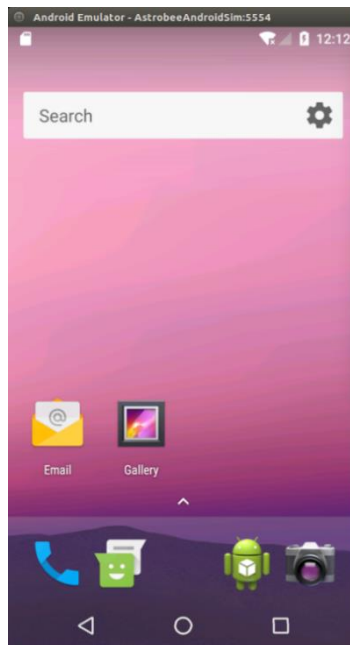


Figure 4.6.2 Android Emulator screen

See https://github.com/nasa/astrobee_android/blob/master/emulator.md for details.

4.6.5. Building the Guest Science Manager APK

To run your program, you must install the Guest Science Manager APK. (see details at https://github.com/nasa/astrobee_android/blob/5b07e4d626781a6f7e0a9cdf4397375cbe509803/guest_science_readme.md)

Execute the following commands in order to build the Guest Science Manager APK.

```
cd $ANDROID_PATH/core_apks/guest_science_manager
$ ANDROID_HOME=$HOME/Android/Sdk ./gradlew assembleDebug
```

4.6.6. Setting up the network

Setup the network between the Astrobee Simulator and the Android Emulator.

See https://github.com/nasa/astrobee_android/blob/master/emulator.md for details.

(1) Setting the HOST network

Execute the following command to open the host file.

```
sudo nano /etc/hosts
```

Add the three lines below to the host file and save.

```
10.42.0.36    hlp
10.42.0.35    mlp
10.42.0.34    llp
```

(2) Setting the environment variables

Execute the following commands to set the environment variables.

```
export ANDROID_PATH="${SOURCE_PATH}_android"
export EMULATOR=$HOME/Android/Sdk/tools/emulator
export AVD="AstrobeeAndroidSim"
```

Note that you need to execute the above commands whenever you open a terminal. If you write these commands in your `.bashrc` file, you don't have to execute them.

(3) Setting up the Android network and starting the Android Emulator

Execute the following commands to set up the Android network and launch the Android Emulator.

```
cd $ANDROID_PATH/scripts  
./launch_emulator.sh -n
```

See https://github.com/nasa/astrobee_android/blob/master/emulator.md for details.

* If the Android network does not work, try turning off Wi-Fi in the Android Emulator.

4.6.7. Installing APKs

If the Android Emulator is not running, execute the following commands to start it.

```
cd $ANDROID_PATH/scripts  
./launch_emulator.sh -n
```

In another terminal, execute the following commands to install the Guest Science Manager APK and your GS APK.

```
cd $ANDROID_PATH/core_apks/guest_science_manager  
adb install -g -r activity/build/outputs/apk/activity-debug.apk  
cd <YOUR_APK_PATH>  
adb install -g -r app/build/outputs/apk/app-debug.apk
```

4.6.8. Setting QR codes, an AR tag, and the target

Set QR codes, an AR tag, and the target in the Astrobee Simulator in the following steps.

- 1) Download Kibo-RPC_SimExtMod.zip from the Download page on the Web site.
- 2) Extract the zip file to the directory you want.
- 3) Execute the following commands.

```
cd <SETUP MODULE DIR>  
chmod +x setup.sh  
./setup.sh
```

4.6.9. Running your program

It's time to run your program!

(1) Launching the Android Emulator

Execute the following commands to launch the Android Emulator.

```
cd $ANDROID_PATH/scripts  
./launch_emulator.sh -n
```

(2) Starting the Astrobees Simulator

Before starting the Astrobees Simulator, execute the following commands to set the ROS environment variables on the other terminal.

```
export ROS_IP=$(getent hosts llp | awk '{ print $1 }')  
export ROS_MASTER_URI=http://${ROS_IP}:11311
```

Execute the following command to start the Astrobees Simulator.

```
roslaunch astrobees sim.launch dds:=false robot:=sim_pub rviz:=true
```

(3) Running the Guest Science Manager APK and GS APK

Execute the following commands to set the ROS environment variables on the other terminal again.

```
export ROS_IP=$(getent hosts llp | awk '{ print $1 }')  
export ROS_MASTER_URI=http://${ROS_IP}:11311
```

Execute the following commands to start the Guest Science Manager APK and to launch the GDS simulator.

```
$ANDROID_PATH/scripts/gs_manager.sh start  
cd $SOURCE_PATH/tools/gds_helper/src  
python gds_simulator.py
```



Operate the GDS simulator to run your GS APK.

- 1) Press any key to grab control
- 2) Select your GS APK.
- 3) Type **b** and press **Enter** to start the GS APK.
- 4) Type **d** and press **Enter** to send a custom guest science command.

Now Astrobees starts to locate the leak!

5. Programming tips

5.1. Do NOT write infinite loops

You **must not** write any infinite loops in your code because no one can stop Astrobee while the loop is executing.

Double check that you use finite loops with a defined counter value, as shown below.

```
// NG
while(!result.hasSucceeded()){
    // do something
}

// OK
final int LOOP_MAX = 5;
int loopCounter = 0;
while(!result.hasSucceeded() && loopCounter < LOOP_MAX){
    // do something
    ++loopCounter;
}
```



5.2. Dealing with randomness

You must consider the randomness of the environment.

When you want to move the robot, refer to the commands below...

```
// move to point 1
api.moveTo(point1, quaternion1, true);
// move to point 2
api.moveTo(point2, quaternion2, true);
// move to point 3
api.moveTo(point3, quaternion3, true);
```

If there is no randomness in the environment, this code works well.

However, AstrobEE may be faced with errors such as **tolerance violations**, and your code will not work, so you have to provide redundant code, as we see below.

Remember, **Do NOT** allow any infinite loops in your code!

```
Result;
final int LOOP_MAX = 5;

// move to point 1(first try)
result = api.moveTo(point1, quaternion1, true);

// check result and loop while moveTo api is not succeeded.
// Do NOT write infinite loop.
int loopCounter = 0;
while(!result.hasSucceeded() && loopCounter < LOOP_MAX){

    // retry
    result = api.moveTo(point1, quaternion1, true);
    ++loopCounter;

}
// move to point 2
//...
```

5.3. About navigation errors

The real world always has uncertainties. Navigation error is one of them and the Kibo-RPC simulator server simulates it.

However, modeling and simulating navigation errors are highly complicated, and this increases the calculation load. Therefore, random error following Gaussian distribution is used generally.

The Kibo-RPC simulator also implements a Gaussian distribution and the parameters are as follows;

Regarding position;

x: mean = 0 m and 3sigma = 0.1 m

y: mean = 0 m and 3sigma = 0.1 m

z: mean = 0 m and 3sigma = 0.1 m

Regarding orientation;

x: mean = 0 degree and 3sigma = 3 degree

y: mean = 0 degree and 3sigma = 3 degree

z: mean = 0 degree and 3sigma = 3 degree

You have to consider that self-positioning and self-orientation obtained from the APIs (getRobotKinematics and getTrustedRobotKinematics) includes these errors.

5.4. Attention to computing resources

If the computing loads are high, Astrobees might be overloaded and not work on orbit. The specifications of Astrobees real robot's HLP are as follows. Note that available resources are different by other software working on HLP. Multithreading is not recommended.

CPU: Qualcomm Snapdragon 820 (4 cores, 2.2GHz)

RAM: 4GB

5.5. Cautions when irradiating laser

Please use the laser only to illuminate the Target. Due to safety reasons, it is prohibited to point the laser at the crew. It is required to inform the intended timing and target of laser irradiation in advance to both flight controllers and the crew. Therefore, create a program where the laser does not hit the crew and does not illuminate a place unnecessarily. DO NOT irradiate the laser blindly, for example, when reading QR/AR is failed. Please check your code not to proceed the processing despite the direction to aim is uncertain

5.6. Performance of Localization

There is a possibility of losing Astrobees's self-position on-orbit. Once the self-position is lost, Astrobees may not be able to recover on its own. In this case, it means it is a game over. Be aware that this incident is not in the simulator. It is well known that the technology Astrobees uses is likely to lose its self-position when the camera view gets too close to wall, floor, airlock, and so on because the camera cannot capture enough features at those places. Please note the above when creating your program.

5.7. Code review

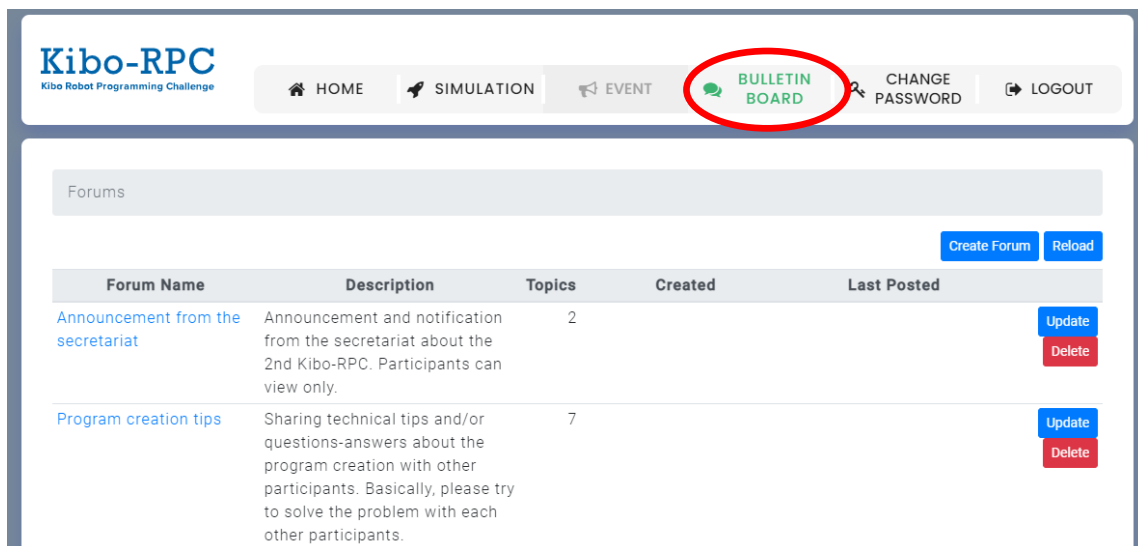
In the Final Round, before the submitted APK is uplinked to Astrobee on orbit, JAXA / ARC performs a code review for safety confirmation in advance. In the code review, if there is an inappropriate code in the submitted APK, we might delete it or instruct the participants to rewrite it.

5.8. Setting the application ID

Each Final Round APK must have a unique application ID to avoid conflict when installing on Astrobee in the ISS. The application ID will be specified for the finalists later.

5.9. Questions and information exchange

You can post questions, share programming tips and exchange information with other teams on the bulletin board. Make effective use of it to create your program!



Forum Name	Description	Topics	Created	Last Posted
Announcement from the secretariat	Announcement and notification from the secretariat about the 2nd Kibo-RPC. Participants can view only.	2		
Program creation tips	Sharing technical tips and/or questions-answers about the program creation with other participants. Basically, please try to solve the problem with each other participants.	7		

Figure 5.9.1 BULLETIN BOARD page

6. Simulator change log

Ver.1.0	Initial Release
Ver.1.1	You can select simulation conditions, Target Position and KOZ Pattern.

- * The version used in the Preliminary Round will be the latest version.

7. Game API details

Details of the Kibo-RPC's game APIs are listed below.

7.1. Method summary

Table 7-1 Method summary

Modifier and Type	Method and Description
gov.nasa.arc.astrobee.Result	<u>flashlightControlBack</u> (float brightness) Set the brightness of the back flash light
gov.nasa.arc.astrobee.Result	<u>flashlightControlFront</u> (float brightness) Set the brightness of the front flash light
android.graphics.Bitmap	<u>getBitmapDockCam</u> () Get Bitmap image of DockCam.
android.graphics.Bitmap	<u>getBitmapNavCam</u> () Get Bitmap image of NavCam.
double[][]	<u>getDockCamIntrinsics</u> () Get camera matrix and distortion coefficients of DockCam.
jp.jaxa.iss.kibo.rpc.api.types.ImuResult	<u>getImu</u> () Get IMU telemetry
org.opencv.core.Mat	<u>getMatDockCam</u> () Get Mat image of DockCam.
org.opencv.core.Mat	<u>getMatNavCam</u> () Get Mat image of NavCam.
double[][]	<u>getNavCamIntrinsics</u> () Get camera matrix and distortion coefficients of NamCam.
jp.jaxa.iss.kibo.rpc.api.types.PointCloud	<u>getPointCloudHazCam</u> () Get PointCloud data of HazCam.
jp.jaxa.iss.kibo.rpc.api.types.PointCloud	<u>getPointCloudPerchCam</u> () Get PointCloud data of PerchCam.
gov.nasa.arc.astrobee.Kinematics	<u>getRobotKinematics</u> () Get current data related to positioning and orientation for Astrobee.



Modifier and Type	Method and Description
gov.nasa.arc.astrobeekinematics	<u>getTrustedRobotKinematics()</u> Get trusted data related to positioning and orientation for Astrobee
gov.nasa.arc.astrobeekinematics.Result	<u>laserControl</u> (boolean state) Turn on/off the laser pointer. If the current state is the same as the input value, nothing happens.
gov.nasa.arc.astrobeekinematics.Result	<u>moveTo</u> (gov.nasa.arc.astrobeekinematics.types.Point goalPoint, gov.nasa.arc.astrobeekinematics.types.Quaternion orientation, boolean printRobotPosition) Move Astrobee to the given point and rotate it to the given orientation.
gov.nasa.arc.astrobeekinematics.Result	<u>relativeMoveTo</u> (gov.nasa.arc.astrobeekinematics.types.Point goalPoint, gov.nasa.arc.astrobeekinematics.types.Quaternion orientation, boolean printRobotPosition) Move Astrobee to the given point using a relative reference and rotates it to the given orientation.
boolean	<u>reportMissionCompletion</u> () Report mission completion, blink the lights, and play sound you prepared.
void	<u>sendDiscoveredQR</u> (java.lang.String value) Send the given QR code data for judge
boolean	<u>startMission</u> () Astrobee undocks and start counting the mission time
void	<u>takeSnapshot</u> () Take snapshots for judge. This API takes about 10-11 seconds.

7.2. Method details

• flashlightControlBack

```
public gov.nasa.arc.astrobeekinematics.Result flashlightControlBack
(float brightness)
```

Set the brightness of the back flash light

Parameters:

brightness - Brightness percentage between 0 - 1.

**Returns:**

A Result instance carrying data related to the execution.
Returns null if the command is NOT executed because of an error

- **flashlightControlFront**

```
public gov.nasa.arc.astorbee.Result flashlightControlFront
(float brightness)
```

Set the brightness of the front flash light

Parameters:

brightness - Brightness percentage between 0 - 1.

Returns:

A Result instance carrying data related to the execution.
Returns null if the command is NOT executed because of an error

- **getBitmapDockCam**

```
public android.graphics.Bitmap getBitmapDockCam()
```

Gets Bitmap image of DockCam.

Returns:

Bitmap image of DockCam(1280 px x 960 px) or null if an internal error occurs. Format:Bitmap.Config.ARGB_8888

- **getBitmapNavCam**

```
public android.graphics.Bitmap getBitmapNavCam()
```

Gets Bitmap image of NavCam.

Returns:

Bitmap image of NavCam(1280 px x 960 px) or null if an internal error occurs. Format:Bitmap.Config.ARGB_8888

- **getDockCamIntrinsics**

```
public double[][] getDockCamIntrinsics()
```

Get camera matrix and distortion coefficients of DockCam. Different values are returned on orbit and in the simulator.

Returns:

Array of camera parameters [camera matrix, distortion coefficients] for DockCam. The array of the camera matrix and distortion coefficients is as follows.

Camera Matrix : $[f_x, 0, c_x, 0, f_y, c_y, 0, 0, 1]$

Distortion Coefficients : $[k_1, k_2, p_1, p_2, k_3]$

- **getImu**

```
public jp.jaxa.iss.kibo.rpc.api.types.ImuResult getImu()
```

Gets IMU telemetry

Returns:

ImuResult data or null if an internal error occurs.

- **getMatDockCam**

```
public org.opencv.core.Mat getMatDockCam()
```

Gets Mat image of DockCam.

Returns:

Mat image of DockCam(1280 px x 960 px) or null if an internal error occurs. Format:CV8UC1

- **getMatNavCam**

```
public org.opencv.core.Mat getMatNavCam()
```

Gets Mat image of NavCam.

Returns:

Mat image of NavCam(1280 px x 960 px) or null if an internal error occurs. Format:CV8UC1

- **getNavCamIntrinsics**

```
public double[][] getNavCamIntrinsics()
```

Get camera matrix and distortion coefficients of NavCam. Different values are returned on orbit and in the simulator.

Returns:

Array of camera parameters [camera matrix, distortion coefficients] for NavCam. The array of the camera matrix and distortion coefficients is as follows.

Camera Matrix : $[f_x, 0, c_x, 0, f_y, c_y, 0, 0, 1]$

Distortion Coefficients : $[k_1, k_2, p_1, p_2, k_3]$

- **getPointCloudHazCam**

```
public jp.jaxa.iss.kibo.rpc.api.types.PointCloud getPointCloudHazCam()
```

Gets PointCloud data of HazCam.

Returns:

PointCloud data of HazCam(224 px x 171 px) or null if an internal error occurs.



- **getPointCloudPerchCam**

```
public jp.jaxa.iss.kibo.rpc.api.types.PointCloud getPointCloudPerchCam()
```

Gets PointCloud data of PerchCam.

Returns:

PointCloud data of PerchCam(224 px x 171 px) or null if an internal error occurs.

- **getRobotKinematics**

```
public gov.nasa.arc.astorbee.Kinematics getRobotKinematics()
```

Gets current data related to positioning and orientation for Astorbee. Note that the data cannot be trusted when the confidence is POOR or LOST.

Returns:

Current Kinematics.

- **getTrustedRobotKinematics**

```
public gov.nasa.arc.astorbee.Kinematics getTrustedRobotKinematics()
```

Gets trusted data (i.e., data with a GOOD confidence) related to positioning and orientation for Astorbee. If no data with a Confidence of GOOD can be obtained, the timeout occurs in 30 seconds.

Returns:

Trusted current Kinematics or null if an internal error occurs or request timeout

- **laserControl**

```
public gov.nasa.arc.astorbee.Result laserControl(boolean state)
```

Turns power on/off Laser Pointer. If it is same state as input parameter, nothing happens.

Parameters:

state - Set a laser pointer true:power on / false:power off.

Returns:

A Result instance carrying data related to the execution. Returns null if the command is NOT executed because of an error

- **moveTo**

```
public gov.nasa.arc.astorbee.Result moveTo(gov.nasa.arc.astorbee.types.Point goalPoint,
gov.nasa.arc.astorbee.types.Quaternion orientation,
boolean printRobotPosition)
```

Moves Astorbee to the given point and rotates it to the given orientation.

**Parameters:**

goalPoint - Absolute cardinal point (xyz)

orientation - An instance of the Quaternion class. You may want to use CENTER_US_LAB or CENTER_JEM as an example depending on your initial position.

printRobotPosition - Flag whether to print robot positions in log or not.

Returns:

A Result instance carrying data related to the execution. Returns null if the command is NOT executed because of an error

- **relativeMoveTo**

```
public gov.nasa.arc.astrobeer.Result relativeMoveTo(gov.nasa.arc.astrobeer.types.Point goalPoint,
gov.nasa.arc.astrobeer.types.Quaternion orientation,
boolean printRobotPosition)
```

Moves Astrobeer to the given point using a relative reference and rotates it to the given orientation.

Parameters:

goalPoint - The relative end point (relative to Astrobeer)

orientation - The absolute orientation

printRobotPosition - Flag whether to print robot positions in log or not.

Returns:

A Result instance carrying data related to the execution. Returns null if the command is NOT executed because of an error

- **reportMissionCompletion**

```
public boolean reportMissionCompletion ()
```

Report mission completion, blink the lights, and play sound you prepared.

Returns:

Returns True if the execution is successful. Returns false if the command is NOT executed because of an error.

- **sendDiscoveredQR**

```
public void sendDiscoveredQR(String value)
```

Send a QR code data for judge.

Parameters:

value - string you read from QR code.

- **startMission**

```
public void startMission()
```

Astrobee undocks and start counting the mission time.

- **takeSnapshot**

```
public void takeSnapshot()
```

Take snapshots for judge. This API takes about 10-11 seconds.

7.2.1. Type information

7.2.2. Summary

Table 7-2 Type information summary

Type	Description
<code>jp.jaxa.iss.kibo.rpc.api.types.ImuResult</code>	IMU telemetry data.
<code>jp.jaxa.iss.kibo.rpc.api.types.PointCloud</code>	Point cloud data.

7.2.3. Details

Details of Kibo-RPC's types are bellow.

(1) `jp.jaxa.iss.kibo.rpc.api.types.ImuResult`

- **getAngularVelocity**

```
public gov.nasa.arc.astrobee.types.Vec3d getAngularVelocity()
```

Returns:

Angular velocity data.

- **getAngularVelocityCovariance**

```
public double[] getAngularVelocityCovariance()
```

Returns:

Angular velocity Covariance data.



- **getLinearAcceleration**

```
public gov.nasa.arc.astrobee.types.Vec3d getLinearAcceleration()
```

Returns:

Linear acceleration data.

- **getLinearAccelerationCovariance**

```
public double[] getLinearAccelerationCovariance()
```

Returns:

Linear acceleration covariance data.

- **getOrientation**

```
public gov.nasa.arc.astrobee.types.Quaternion getOrientation()
```

Returns:

Orientation data.

- **getOrientationCovariance**

```
public double[] getOrientationCovariance()
```

Returns:

Orientation covariance.

(1-1) jp.jaxa.iss.kibo.rpc.api.types.PointCloud

- **getWidth**

```
public int getWidth()
```

Returns:

Width of point cloud data.

- **getHeight**

```
public int getHeight()
```

Returns:

Height of point cloud data.

- **getPointArray**

```
public gov.nasa.arc.astrobee.types.Point[] getPointArray()
```

Returns:

A point cloud data array of the size of width x height.

(2) Reference

Please refer below for information about other Types.

Type	URL
gov.nasa.arc.astrobee .Kinematics	https://github.com/nasa/astrobee_android/blob/5b07e4d626781a6f7e0a9cdf4397375cbe509803/astrobee_api/api/src/main/java/gov/nasa/arc/astrobee/Kinematics.java
gov.nasa.arc.astrobee .Result	https://github.com/nasa/astrobee_android/blob/5b07e4d626781a6f7e0a9cdf4397375cbe509803/astrobee_api/api/src/main/java/gov/nasa/arc/astrobee/Result.java
gov.nasa.arc.astrobee .types.Vec3d	https://github.com/nasa/astrobee_android/blob/5b07e4d626781a6f7e0a9cdf4397375cbe509803/astrobee_api/api/src/main/java/gov/nasa/arc/astrobee/types/Vec3d.java
gov.nasa.arc.astrobee .types.Quaternion	https://github.com/nasa/astrobee_android/blob/5b07e4d626781a6f7e0a9cdf4397375cbe509803/astrobee_api/api/src/main/java/gov/nasa/arc/astrobee/types/Quaternion.java
gov.nasa.arc.astrobee .types.Point	https://github.com/nasa/astrobee_android/blob/5b07e4d626781a6f7e0a9cdf4397375cbe509803/astrobee_api/api/src/main/java/gov/nasa/arc/astrobee/types/Point.java