UNIVERSITY OF NEW SOUTH WALES

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Developer manual for UNSW’s Bellabot

Warnings and introduction

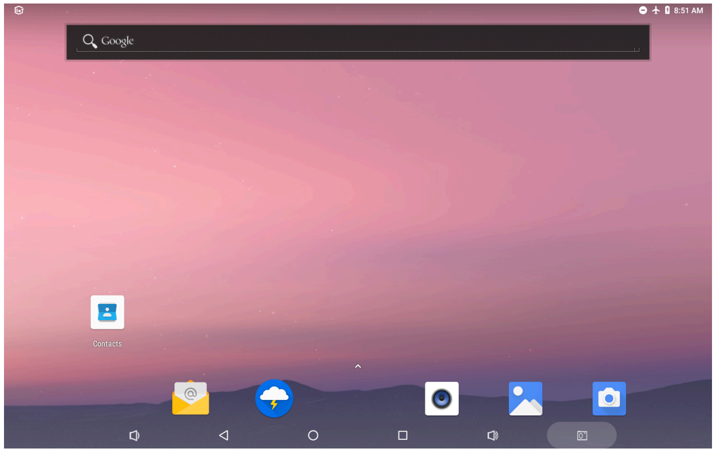
NEVER connect the bot to an internet-connected network through wifi. Nothing should happen, but there is a risk of the bot bricking itself.

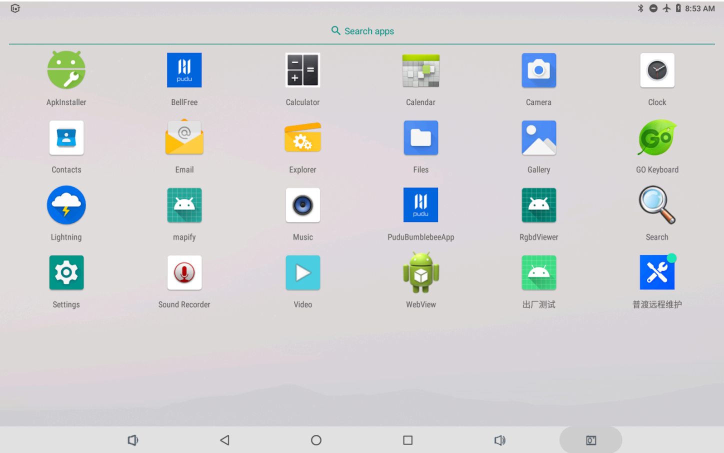
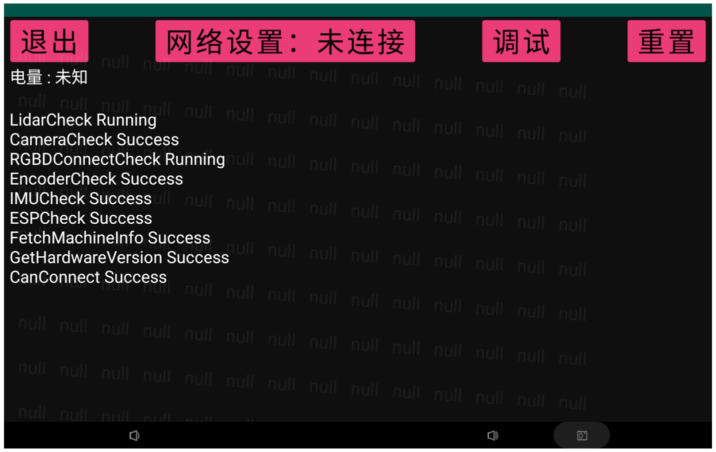
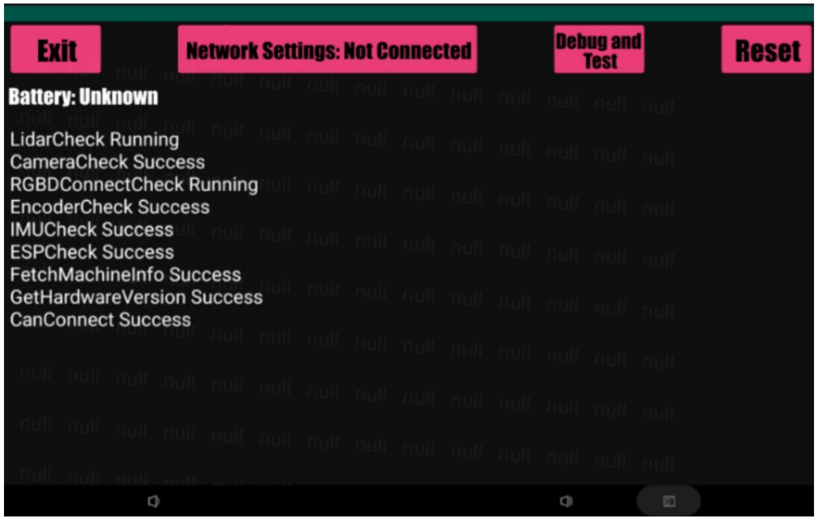
This is a manual for setting up the software and debugging tooling for any developer working on Dr Hammond Pearce's Bellabot at UNSW. If you are working with Dr Hammond, you should have access to a Dell laptop with all the tooling already set up. Refer to the rest of the document if you need a guide to setting up development on your own environment or if you do not have access to the preconfigured laptop.

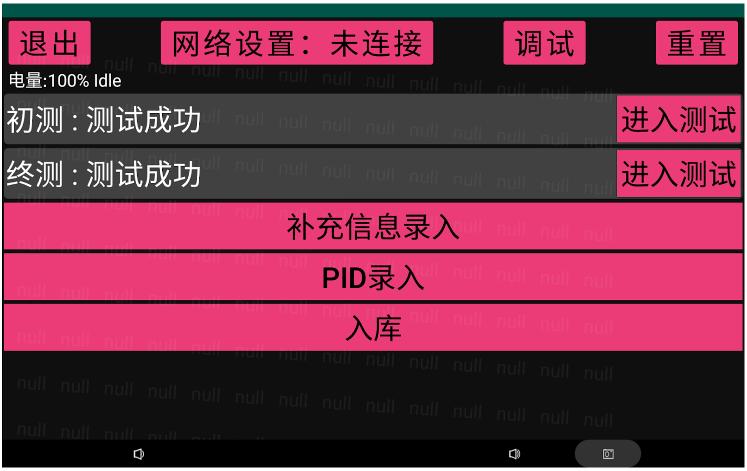
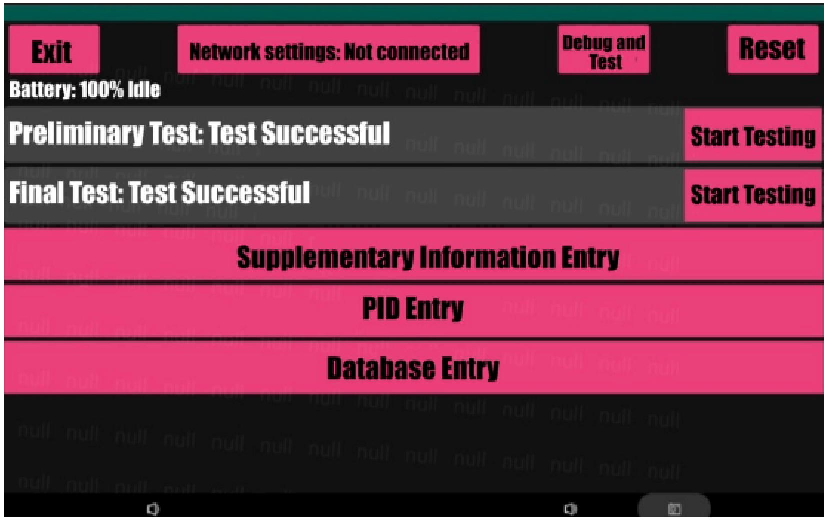
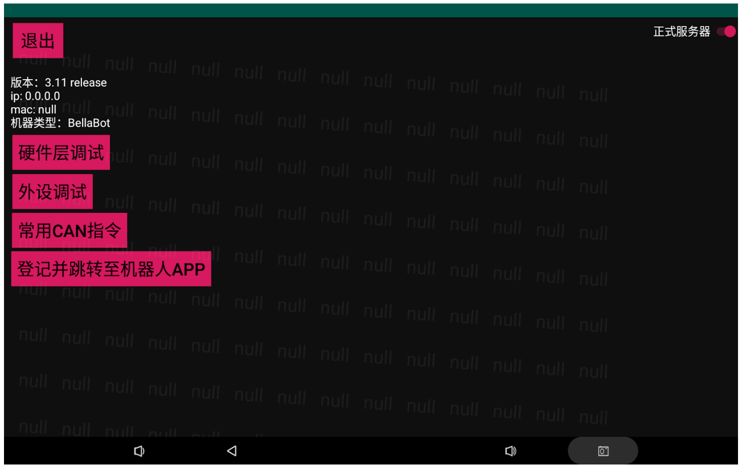
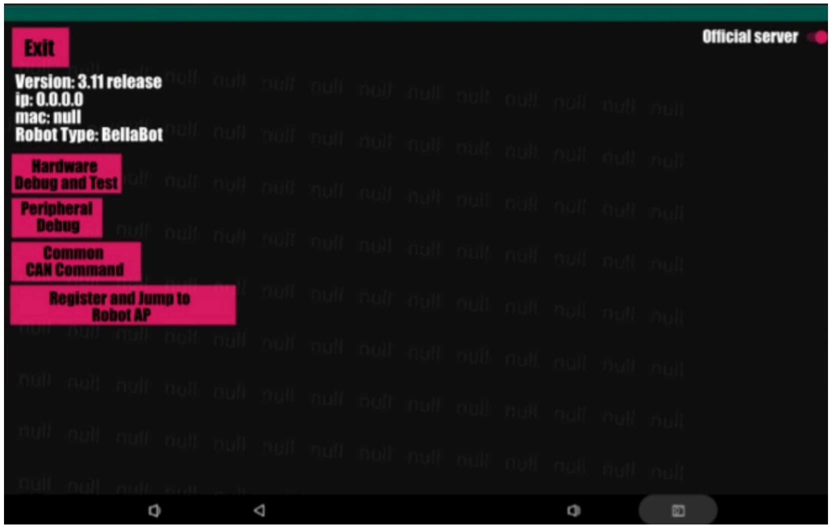
Table of contents

Factory test (“出厂测试”) translation and reference

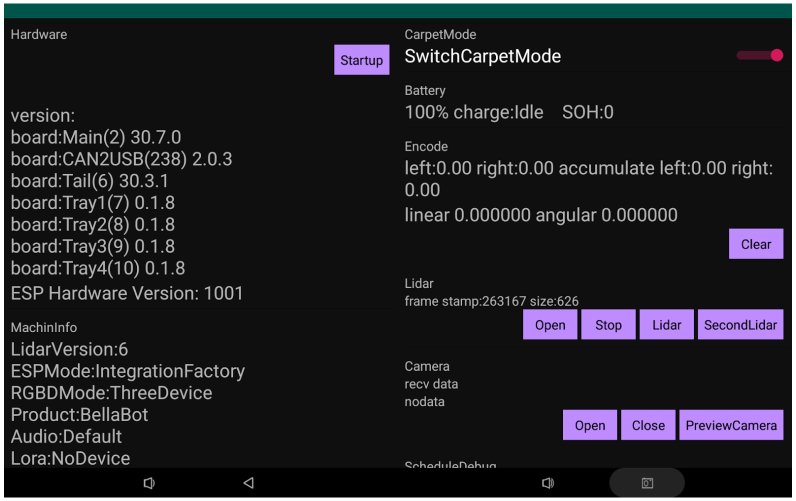
Below is a walkthrough of the factory test app that came with the Bellabot, which provides the ability to individually test the bot’s hardware modules. A translation into English is provided for reference.

Steps	Displayed Screen	Translation
1. Start the robot by pressing the start button for 3 seconds, make sure the battery is on and the cover is well placed. The home page will be displayed.		

2. Apps available will be displayed.	 8:53 AM Search apps ApkInstaller BellFree Calculator Calendar Camera Clock Contacts Email Explorer Files Gallery GO Keyboard Lightning mapify Music PuduBumblebeeApp RgbdViewer Search Settings Sound Recorder Video WebView 出厂测试 普渡远程维护	出厂测试 (5th app in the 4th row) - Factory test 普渡远程维护 (last app in the 4th row) - Pudu Remote Maintenance Enter the Factory test app for the privilege control and access to the robot operation data. I.e. All the screens displayed in the following are the displays for the Factory test app.
3. After entering the factory test app, it will go through an auto-check for all the hardware and software in the system.	 电量：未知 LidarCheck Running CameraCheck Success RGBDConnectCheck Running EncoderCheck Success IMUCheck Success ESPCheck Success FetchMachineInfo Success GetHardwareVersion Success CanConnect Success	 Exit Network Settings: Not Connected Debug and Test Reset Battery: Unknown LidarCheck Running CameraCheck Success RGBDConnectCheck Running EncoderCheck Success IMUCheck Success ESPCheck Success FetchMachineInfo Success GetHardwareVersion Success CanConnect Success

4. When all the testing has been done successfully, the display will allow users to get access to the hardware and software data.	 退出 网络设置：未连接 调试 重置 电量:100% Idle 初测：测试成功 终测：测试成功 	 Exit Network settings: Not connected Debug and Test Reset Battery: 100% Idle Preliminary Test: Test Successful Final Test: Test Successful Supplementary Information Entry PID Entry Database Entry
5. Entering the debug and test, control settings for the robot hardware and peripherals are available.	 版本：3.11 release ip: 0.0.0.0 mac: null 机器类型：BellaBot 硬件层调试 外设调试 常用CAN指令 登记并跳转至机器人APP	 Version: 3.11 release ip: 0.0.0.0 mac: null Robot Type: BellaBot Hardware Debug and Test Peripheral Debug Common CAN Command Register and Jump to Robot AP

5.1 Hardware testing
This part is originally displayed in English. No need for translation.



Hardware

version:
board:Main(2) 30.7.0
board:CAN2USB(238) 2.0.3
board:Tail(6) 30.3.1
board:Tray1(7) 0.1.8
board:Tray2(8) 0.1.8
board:Tray3(9) 0.1.8
board:Tray4(10) 0.1.8
ESP Hardware Version: 1001

MachinInfo
LidarVersion:6
ESPMode:IntegrationFactory
RGBDMode:ThreeDevice
Product:BellaBot
Audio:Default
Lora:NoDevice

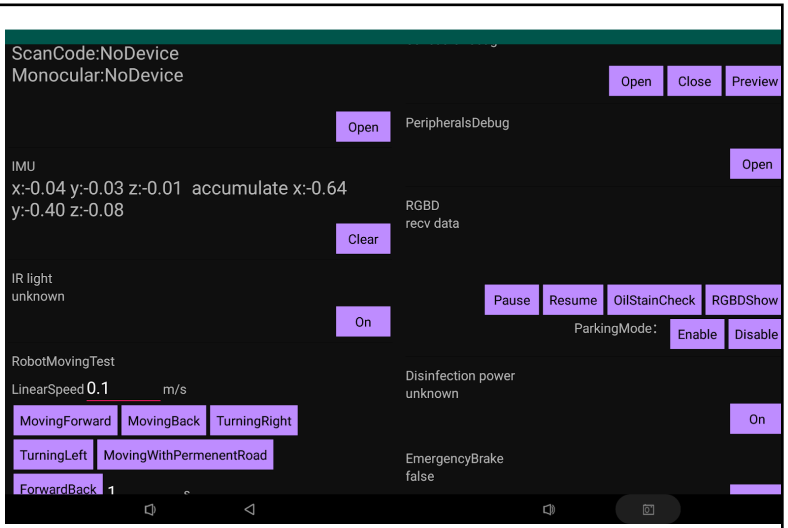
CarpetMode
SwitchCarpetMode

Battery
100% charge:Idle SOH:0

Encode
left:0.00 right:0.00 accumulate left:0.00 right: 0.00
linear 0.000000 angular 0.000000

Lidar
frame stamp:263167 size:626

Camera
recv data
nodata



ScanCode:NoDevice
Monocular:NoDevice

IMU
x:-0.04 y:-0.03 z:-0.01 accumulate x:-0.64 y:-0.40 z:-0.08

IR light
unknown

RobotMovingTest
LinearSpeed 0.1 m/s

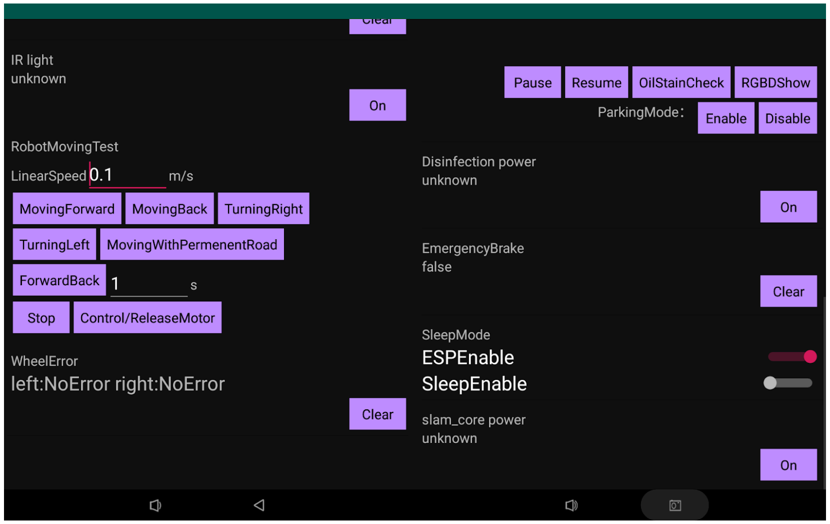
PeripheralsDebug

RGBD
recv data

ParkingMode:

Disinfection power
unknown

EmergencyBrake
false



IR light
unknown

RobotMovingTest
LinearSpeed 0.1 m/s

ForwardBack 1 s

WheelError
left:NoError right:NoError

ParkingMode:

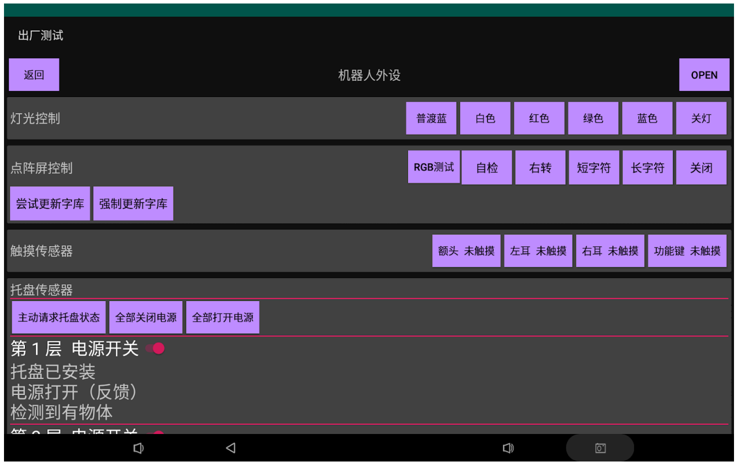
Disinfection power
unknown

EmergencyBrake
false

SleepMode
ESPEnable
SleepEnable

slam_core power
unknown

5.2 Peripherals testing



出厂测试

返回 机器人外设 OPEN

灯光控制 普渡蓝 白色 红色 绿色 蓝色 关灯

点阵屏控制 RGB测试 自检 右转 短字符 长字符 关闭

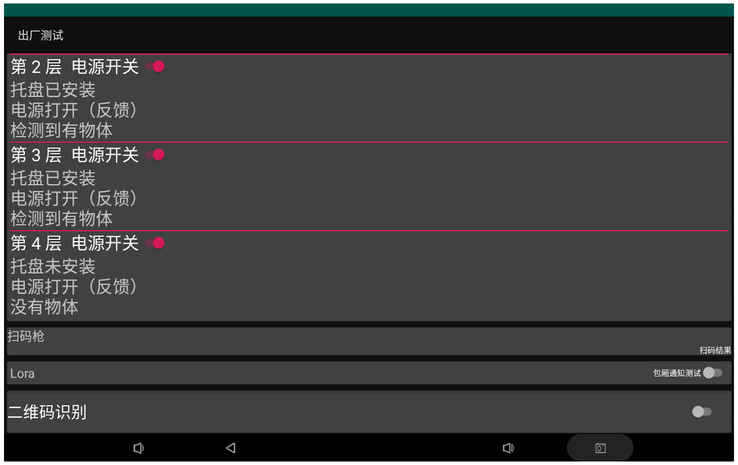
尝试更新字库 强制更新字库

触摸传感器 额头 未触摸 左耳 未触摸 右耳 未触摸 功能键 未触摸

托盘传感器

主动请求托盘状态 全部关闭电源 全部打开电源

第 1 层 电源开关
托盘已安装
电源打开（反馈）
检测到有物体



出厂测试

第 2 层 电源开关
托盘已安装
电源打开（反馈）
检测到有物体

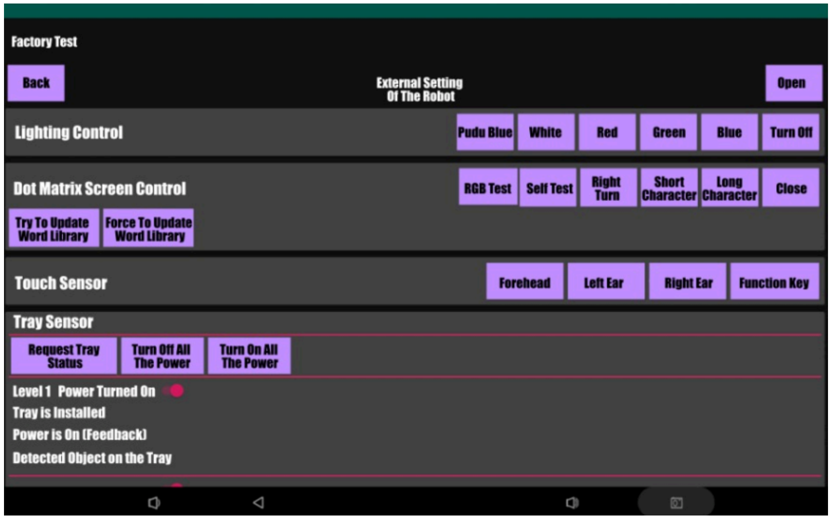
第 3 层 电源开关
托盘已安装
电源打开（反馈）
检测到有物体

第 4 层 电源开关
托盘未安装
电源打开（反馈）
没有物体

扫码枪 扫码结果

Lora 包厢通知测试

二维码识别



Factory Test

Back External Setting Of The Robot Open

Lighting Control Pudu Blue White Red Green Blue Turn Off

Dot Matrix Screen Control RGB Test Self Test Right Turn Short Character Long Character Close

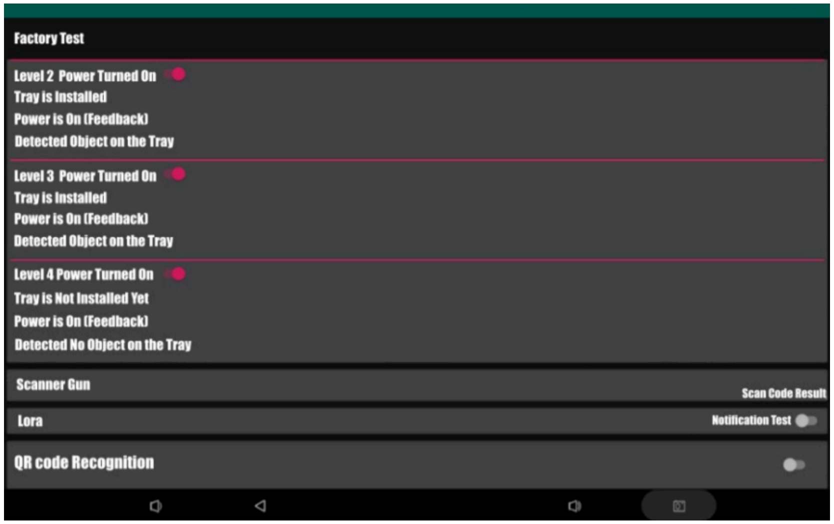
Try To Update Word Library Force To Update Word Library

Touch Sensor Forehead Left Ear Right Ear Function Key

Tray Sensor

Request Tray Status Turn Off All The Power Turn On All The Power

Level 1 Power Turned On
Tray is Installed
Power is On (Feedback)
Detected Object on the Tray



Factory Test

Level 2 Power Turned On
Tray is Installed
Power is On (Feedback)
Detected Object on the Tray

Level 3 Power Turned On
Tray is Installed
Power is On (Feedback)
Detected Object on the Tray

Level 4 Power Turned On
Tray is Not Installed Yet
Power is On (Feedback)
Detected No Object on the Tray

Scanner Gun Scan Code Result

Lora Notification Test

QR code Recognition

Android Debug Bridge (adb)

Introduction

Android Studio has a suite of tools that is very useful for debugging and working on the bot. In particular, Android Debug Bridge (adb) is especially crucial for the ability to view logs on the bot.

Prerequisites

- A Mac or Linux machine.
- Android Studio or the Android command line tools installed (see https://developer.android.com/studio for more details).
- Root Explorer installed on the bot.
- Developer options and USB debugging enabled on the bot.

Connection via USB (for Linux only)

Connection via USB also requires a USB-C to USB-A cable. Connect the development machine and the bot with the USB cable (the USB-C side must go into the bot, and currently the front back plate must be taken off).

First time connection/setup

1. Set the USB connection mode in BellaBot's Settings > Developer Options to MTP (Media Transfer Protocol).
2. Copy the adbkey.pub from the development machine to the bot. The current collection of adb keys reside in the SD card/data/adb_keys folder.
3. Rename the adbkey.pub file on the bot to pd_adb_key_(your initials). You may also edit the existing pd_adb_key (a group of keys) and append the contents of adbkey.pub into a new line in pd_adb_key.

adb authorisation

Note that these steps need to be repeated every time you wish to connect to the bot via adb. If you disconnect the USB cable and then reconnect, these steps need to be repeated.

1. Open Root Explorer on the bot.
2. From the SD card/data/adb_keys folder, copy your pd_adb_key_(your initials) (or the group of keys) to /data/misc/adb. /data/misc/adb is the path from device root, not from the device storage.
3. Rename the new file to pd_adb_key if it is not named that already, to replace the existing one.

4. On your development device, you may now start the adb server by running any adb command in your command line.
5. If the adb server has already started before doing these steps, you must restart the adb server by running “adb kill-server” before any other adb command.
6. If the adb server is connected, you can verify this by typing “adb devices” and seeing “rockchip-bumblebee   device” under the list of devices.

Connection via network (for Mac and Linux)

The same First time connection/setup steps apply if you have not connected before. This means that you need to find some way to put the development device’s adb key onto the bot through USB somehow, which may be an issue for Mac machines. See if you can transfer that key to a Linux machine, which can then copy the key on to the device.

Connection via network requires both the Bellabot and the development device to be connected to the same network. As the Bellabot should not be connected to wifi, this means that it should be connected to a non-wifi connected router.

adb authorisation

Note that these steps need to be repeated every time you wish to connect to the bot via adb.

1. On the bot, swipe down and hold down the wifi icon to select the network to connect to.
2. Connect to the same network on the development device.
3. Open Root Explorer on the bot.
4. From the SD card/data/adb_keys folder, copy your pd_adb_key_(your initials) to /data/misc/adb.
5. You can view the bot’s IP address from Settings > About Phone > Status. On your development device, run “adb connect 192.168.x.x”, where 192.168.x.x is the bot’s IP address.
6. Once you wish to disconnect from the Bellabot, run “adb kill-server”.

ROS Integration

Introduction

ROS (Robot Operating System) is primarily used in robotics for development and control.

- ROS can be installed and run on Android devices, allowing them to act as robot controllers or to perform various tasks related to robotics.
- Installing ROS on Android involves adapting it to the ARM architecture and the Android operating system. This can be done through ROSJava, a set of software libraries and tools that allow ROS to be used with Java-based programs, such as those developed for Android. This enables communication with ROS nodes on other systems.
- With ROS on Android, devices can perform tasks such as sensor data acquisition (like camera, IMU data, motor speeds), processing, and sending commands to other ROS-enabled robots or systems.

Prerequisites

Before installing ROS on Android, ensure your development environment meets these prerequisites:

1. **Operating System**: Your development machine should be running Ubuntu 16.04 Xenial Xerus natively or through WSL (Windows Subsystem for Linux).

2. **ROS Kinetic**: ROS Kinetic must be installed on your development machine. This allows interaction with ROS nodes on the Bellabot or other ROS-enabled devices.

3. **Android Studio**: Install Android Studio on your development machine. Android Studio is essential for developing, maintaining, and uploading code to the Bellabot.

4. **Java JDK 10**: JDK 10 or older is essential for working with ROSjava, as it is very outdated. Android Studio will automatically download the correct Gradle version when it is set to use JDK 10 under Gradle settings.

Developing the App

Step 1: Install ROS on Your Development Machine

- Follow the official ROS installation instructions for Ubuntu 16.04 Xenial Xerus. This will provide the necessary tools and libraries to interact with ROS nodes.

Step 2: Set Up Android Studio

- Download and install Android Studio from the official website. This IDE is used for developing Android applications and will be essential for building and deploying your ROS-enabled Android app.

Step 3: Develop the Android App

- Use Android Studio to open the **ros-bellabot** project. This project includes all the necessary assets, Java code and Gradle files to develop and build the app.
- Under Settings > Build, Execution, Deployment > Build Tools > Gradle, change the Gradle JDK from the default (usually 17.0) to 10. From the dropdown menu, select Add JDK and select the directory where you installed JDK 10.
- Sync your project to apply the Gradle changes. If there are any errors, retry sync.
- Add, change or refactor code as desired.

Step 4: Build and Deploy

- Once you are happy with your changes, build the Android app.
- If build completes with no errors, hover over the main menu (4 horizontal lines) on the top left and open the Build dropdown menu. Then, under Build App Bundle(s) / APK(s), select Build APK(s). This will compile the project into the APK format that can be installed on Bellabot.

Step 5: Install the App

- Connect Bellabot to your development machine via a USB cable. Copy the APK into its DCIM directory.
- On the Bellabot, open the Explorer app and find the DCIM directory. You should see your APK there. Click on the file, and select Install.

Step 5: Start ROSBellabot

- On the Bellabot, open the app that you just installed. Under **Advanced Options**, select **New Public Master**.
- On your development machine, connect to the same WiFi network as Bellabot. Then, open a new Terminal and run `roscore.`
- In a new Terminal window, run `export ROS_MASTER_URI=http://<IP_ADDRESS>:11311`. Replace <IP_ADDRESS> with Bellabot's IP address.
- Finally, run `rostopic list` to see a list of ROS topics published by Bellabot, and `rostopic echo <topic>` to subscribe to a topic and view its contents.

For detailed guidance and troubleshooting, refer to the documentation provided by ROS and Android Studio, as well as the **rosjava_core** and **android_core** git repositories.

Possible two-way communication:

While the current ROS app publishes sensor data, it might be possible to set it up to receive updates instead. The following link contains code for a rosjava listener, which could work on Bellabot too:
https://github.com/rosjava/rosjava_core/blob/kinetic/rosjava_tutorial_pubsub/src/main/java/org/ros/rosjava_tutorial_pubsub/Listener.java

BellaBotApp Development

Introduction

BellaBotApp is the currently created app that interfaces best with the API developed to get data from and set data to the BellaBot. This section of the developer's manual will go through the steps to set up your environment to work on this application

Prerequisites

- A regular development environment.
- Android Studio or the Android command line tools installed (see https://developer.android.com/studio for more details).
    - Gradle should come with Android Studio, our development used version 8.5
    - You should have an emulator through Android Studio, set up with the operating system Android Oreo 3.1. You can use the Pixel Tablet as this is the device most similar to the BellaBot's tablet.
- Node (we used version 22.4.0)
- Java (we used version 17.0.11)

Note: To set up Expo Go with Android Studio, see: https://docs.expo.dev/workflow/android-studio-emulator/

Setting Up The Project

For this application, you need two different programming environments, one for the frontend and one for the backend.

To get started with the frontend:

1. Change directories into the root of the application, specifically BellaBotApp/
2. Run the command `npm i` to install all relevant packages
3. Assuming that Android Studio is set up correctly (following steps from the developer manuals above), run the command `npm run android` to start up the project.
    a. You should see a QR code in your terminal. This is a sign that Expo Go started up properly
    b. In a couple seconds, the app should open up in your emulator. If not, press a when prompted to open it explicitly

To get started with the backend:

1. In Android Studio, select File -> Open…, and then choose the BellaBotApp/android/ folder as the root of your Android Studio project.
2. Allow the project to build in Android Studio
3. To validate that the project was built correctly, press the hammer icon to build the project.

If you want to develop on the backend, you will have to use Android Studio. This is because the backend code requires functionality that is only produced by this IDE, such as automatic generation of packages to be used, like the com.facebook.react package.

Deploying To BellaBot

To deploy to the bot, ensure that you have ADB access to the robot. You will also need a tool called ‘eas’. To download this, run ‘npm i -g eas-cli’. The global install is so that other developers won’t have to use this tool, as it requires a free account to be made to use it.

1. Create a free account with EAS. Guides are available online for this step
2. If you’ve made changes to the metadata of the application, run `npx expo prebuild –platform android` to regenerate this information for the upcoming build.
3. While connected to the internet, run `eas build –platform android –profile production –local`. This will start the build process and after a while will produce a .apk file.
4. Finally, connect to BellaBot through ADB and send the apk to the bot
    a. If you’re using the network to transfer the file, the command should be of the form `adb -s BOT_IP_ADDRESS install APK_NAME.apk`
5. The terminal should produce the output ‘Performing Streamed Install’ and then ‘Success’ once downloaded.

Note: If there is a currently existing BellaBotApp on the bot, you must uninstall it before downloading a new one onto the bot.

Running Cypress Tests For BellaBotApp

Cypress testing doesn’t natively work with React Native, it’s instead built mainly for web applications.

1. Open the BellaBotApp on the web by running ‘npm run web’
    a. This should open up a web browser and show a somewhat augmented version of the app
2. Run ‘npx cypress run’ to run the tests