07_Android App for Bluetooth Remote Control

Introduction

The aim is to develop an Android App to be able to control the robot using a phone. Java and XML programming is required. We have decided to draw a joystick on a touch screen (images below) and communicate via Bluetooth with the robot.

The app is divided in three different modes, corresponding to the current 3 modes of functioning supported by the robot.

-The first mode consists of controlling the robot directly through a Joystick. The App can show sensors results such as the distance from obstacles (ultrasonic sensor), the position of the robot and its orientation (calculated with the encoders). -In the second mode, the robot is autonomous. It can move and avoid obstacles. The App contains just an ON/OFF button to launch and stop this mode. As in the first mode, it is able to show results from the sensors. -In the third mode, the robot is placed on a map where it has to find 7 markers (S for Start, 1, 2, 3, 4, 5 and E for End). Once it has identified the location of every marker, it has to be able to move from the first one to the last one in the correct order. The App is made of a map that can be divided in lines and rows to define zones. The robot moves in the area and sends the relative positions of the marker (the zone number) to the App.

Prerequisite

Android Studio software is required: 1. Download and install it 2. Extract the Joystick_V1_0.zip folder 3. Open the software, choose “Open Project” and go to the extracted folder location 4. In Joystick_V1_0 folder, open the file: Joystick.

The software builds the project when opening, wait. Your phone needs to use Android operating system.

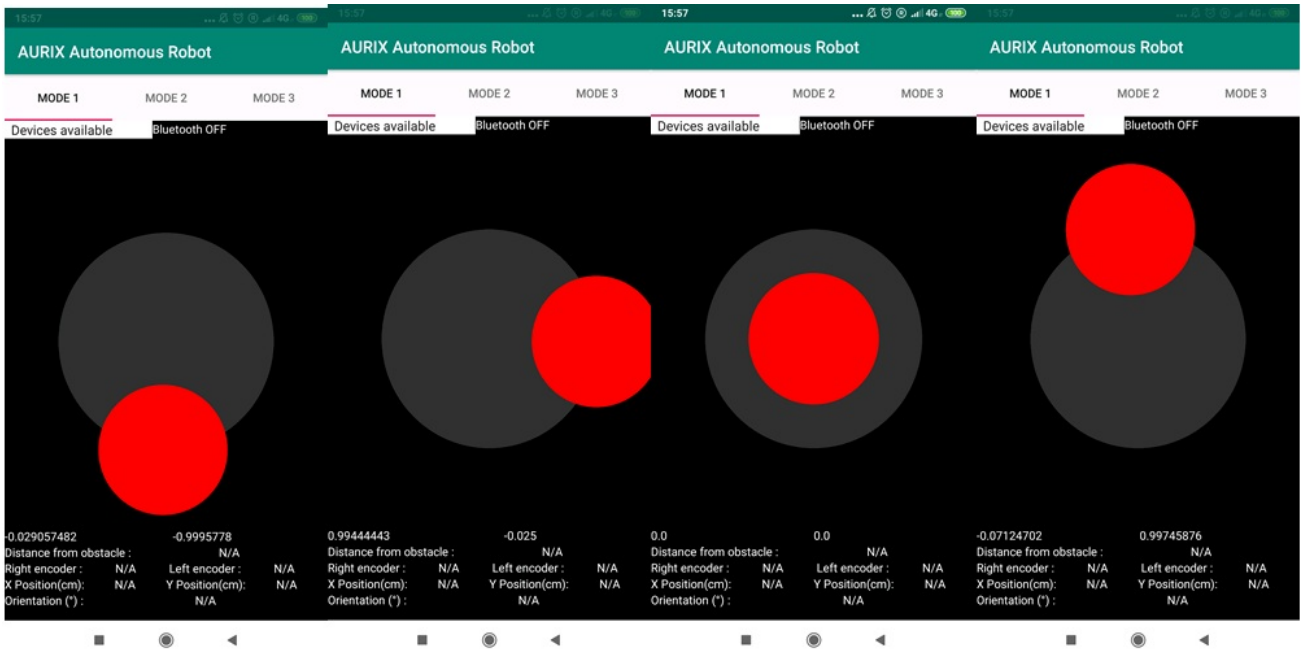
5. Put your phone on “Developer Mode” (find on Internet how to make it) and do not forget to allow “USB debugging” on your phone “Developer Options”.
6. Connect your phone to your computer and run the app from Android Studio
7. Then select your phone in the list and select “ok”.

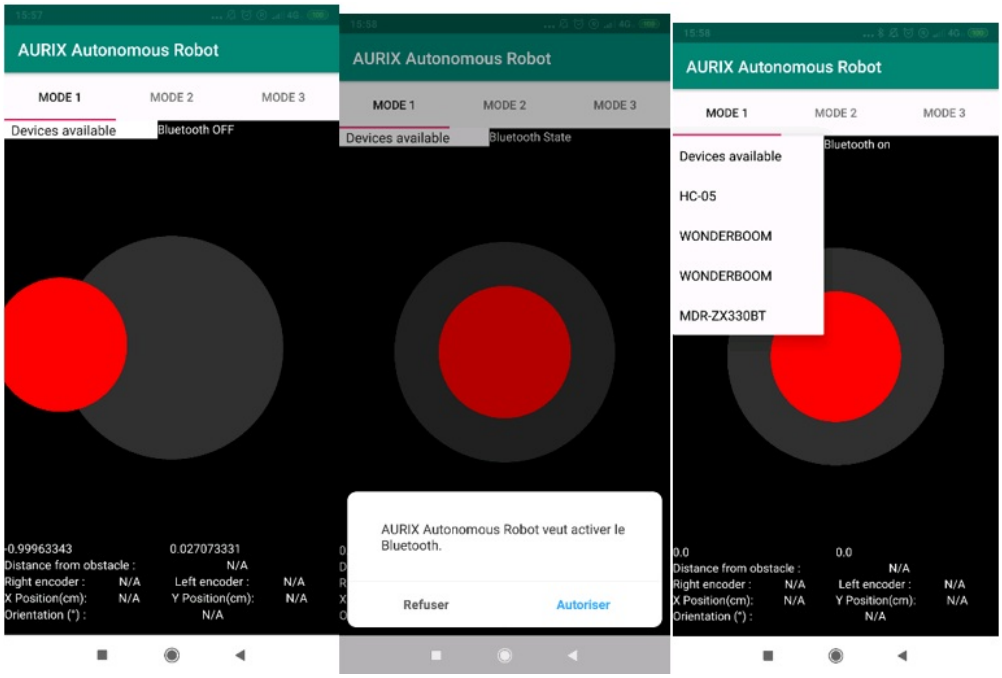
An App Item called ‘Joystick’ will be created on your phone.

Table of Contents

First Mode

This joystick is made of two circles placed in the middle of the screen (it is a relative position calculated with the size of the app layout, and depends on the size of the phone). The user can move the red circle until its center reaches the perimeter of the gray one. We get the x and y value of the center of the red circle to know the joystick position and work in a grey circle of radius 1. X= [-1, 1] and Y= [-1, 1].





A drop-down menu (called spinner on Android Studio) contains the Bluetooth paired devices. The textView on the right upper side shows the Bluetooth state (“Bluetooth On”, “Bluetooth Off”, “Turning On Bluetooth…”, “Turning Off Bluetooth…”).

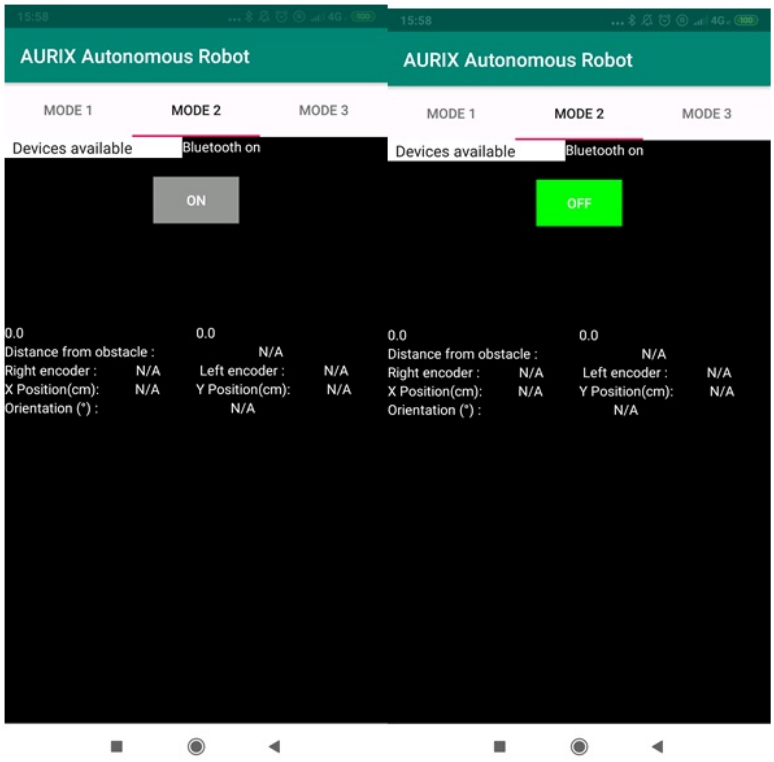
⚠ Devices considered as paired does not mean they are available for Bluetooth connection. It just means the phone knows them and can engage a socket.

Under the the Joystick, two text boxes (called textView on Android Studio) show and update the X and Y values of the Joystick position. Then, textViews for the distance from the obstacle, the values of the encoders, the X and Y position of the robot since its first position and its orientation are displayed.

⚠ In some versions of the robot project code, there is no value sent, this is why “N/A” is displayed.

Second Mode

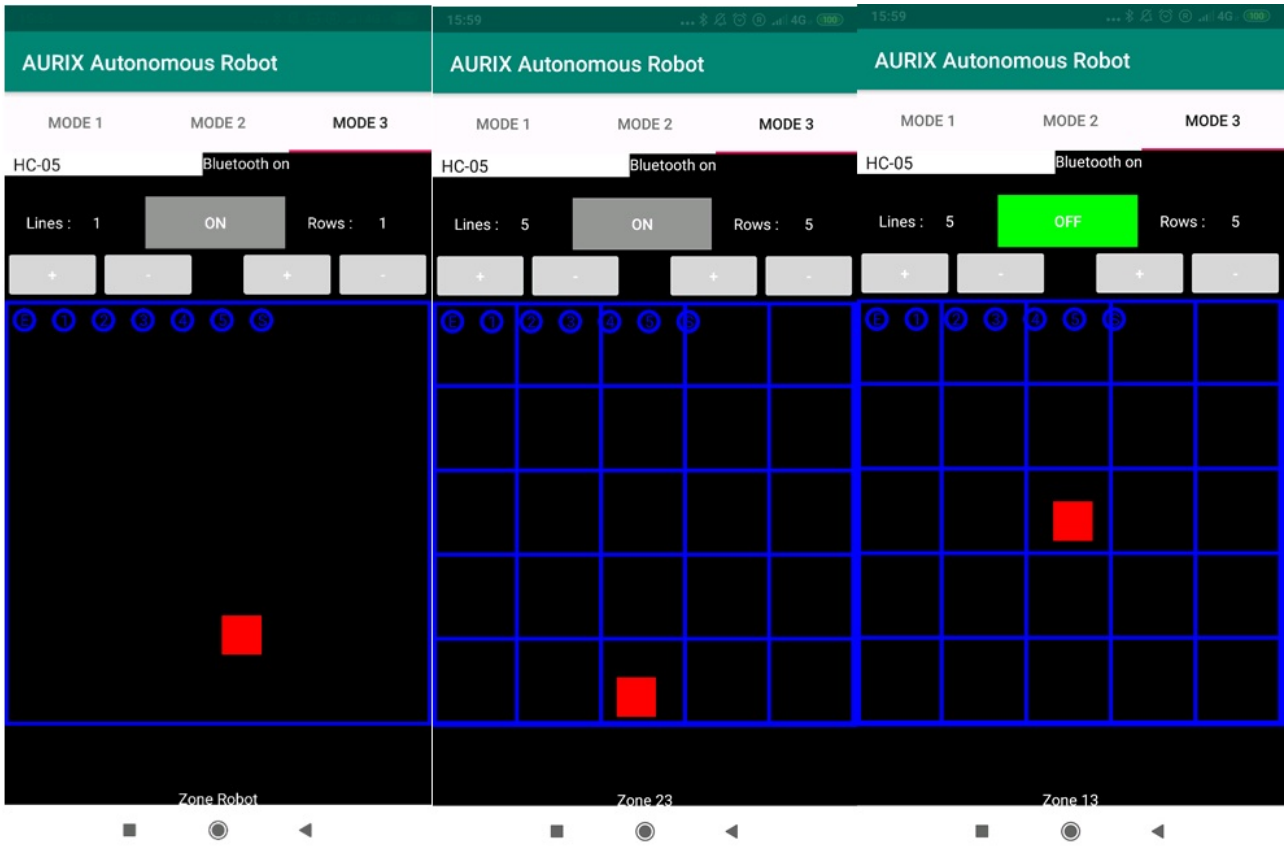
The second mode is just composed of an ON/OFF button. Once the Bluetooth connection is working with the robot, the user can start and stop the autonomous driving.



Third Mode

In this mode, the user can define zones in the map where the robot moves. From 1 line / 1 row (1 zone) to 7 lines / 7 rows (49 zones). Before the user launch the test, he can move the robot (represented with the red square) to its zone departure by clicking on it. Then, autonomous driving can be started with by clicking on the ON/OFF button.

⚠ With the current version, the robot does not send its position. There is already a method to get some data that need to be improve to get position/speed (or others), and then a method is needed to update the draw of the robot position in real time.



JoystickView.java class

This class creates the Joystick described in the Introduction, it extends SurfaceView and implements SurfaceHolder.Callback and View.OnTouchListener.

- SurfaceView is a java class that manages the drawing surface of the App. More details on: https://developer.android.com/reference/android/view/SurfaceView
- SurfaceHolder.Callback and as we are creating a Joystick, we need to have listeners to be able to know how and where our object was moved. More details on: https://developer.android.com/reference/android/view/SurfaceHolder.Callback https://developer.android.com/reference/android/view/View.OnTouchListener

The most important methods:

- private void setupDimensions():

Get the size of the screen (width and height) to determine the center, where the joystick is built. We also determine the two circles radius. Values are dynamic to adapt the size of the joystick with the phone used.

- private void drawJoystick(float newX, float newY):

Draw the Joystick on the current Canvas (link below).

- public boolean onTouch(View v, MotionEvent e):

This method is called when user moves uses the touch screen. The drawJoystick method is called depending on the location of the finger and values taken by X and Y are limited by the size of the gray circle (the base radius).

- public interface JoystickListener:

This interface is a Listener that calls the “void onJoystickMoved(float xPercent, float yPercent, int id);” method to get the X and Y values in the main activity.

More details about Canvas on: this link.

MapView.java class

This class was created for the 3rd mode. It is used to display and update the view of the map where the robot moves. It is divided in lines and rows to create zones where the markers (S, E and numbers) are placed during the robot motion. It contains 3 constructors.

- private void init(): This method initiate the visual attributes of the objects that will be placed on the map (robot, markers, rectangles)
- private void setupDimensions(): This method defines the map dimensions. It is made of a square which each side measures the width of the phone screen (minus the width of the line draw)
- protected void onDraw(Canvas canvas): This method is called when the class object is instantiated and every time the canvas is modified (e.g. the user adds lines and rows). It calls “DrawMap” (described under)
- DrawMap(Canvas canvas, float cx[], float cy[], float robotPosition[]): This method calculates and draws all the zones and save their position (middle X and middle Y of the rectangle) in a 2D array
- public boolean onTouchEvent(MotionEvent event): This method moves the rectangle representing the robot to the location chosen by the user, and update the

Zone Position of the Robot (calculation made by the method under)

- private String ZoneRobot(float XRobot, float YRobot): Calculate the zone where the robot is located
- public void AddZones(int iLines, int iRows): Update the number of rows, lines and zones that have to be drawn

Main.java class

This class calls the different objects and class used by the App. It extends AppCompatActivity and implements JoystickView.JoystickListener (the interface we have created before).

More details on: this link.

The most important methods:

- public void onCreate(Bundle savedInstanceState):

The main.java class is the first object created when app is launched. The onCreate method is then called. We call there all the objects that have to be created at the App opening : CreateJoystick(); CreateLabels(); CreateButtons(); initiate_Bluetooth(); CreateSpinner(); (methods described just below except Bluetooth => see Bluetooth Communication)

- public void CreateSpinner():

We instantiate a new spinner object, a new ArrayList of Strings and a new ArrayAdapter. The ArrayList is filled in later with the names of the available devices. The spinner cannot just copy the content of the ArrayList, we need to use an ArrayAdapter to fulfil it. *public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l)* method is overridden to either request a Bluetooth connection with the selected device in the list, or disconnect it if a connection was already on and the user selects the “Disconnect” field.

- public void CreateJoystick():

This method instantiate a new JoystickView object.

- public void CreateLabels():

This method creates all the textView objects the same way :

1) We calculate the relative position of the object depending on the size of the screen available. 2) We instantiate a new textView object. 3) We set the Color, the X and Y position, and the text. 4) We add it to the current View with the addContentView() method

(link below)

- public void CreateModeScreens(): This method instanciate the 3 mode screens and their listeners. Each time a mode is selected, we decide which elements are displayed or not
- public void UpdateLabels(float xPercent, float yPercent):

Update the value of X and Y (joystick position) in their textView.

- public void onJoystickMoved(float xPercent, float yPercent, int id):

When the Joystick is moved, the interface calls this method. 9 positions are defined with a character depending on the Joystick position : stop “q”, go straight “w”, go back “s”, turn left “a” and turn right “d”, curve left straight “e”, curve right straight “r”, curve left back “f”, curve right back “g”. We use the method sendData(String msg) to send the message via the Bluetooth socket created.

- Public void CreateButtons()

Instantiate the plus and minus buttons for the mode 3 and their listeners. These buttons are used to add or delete lines and rows to determine the zones of the mode

- public void ModeSelection(String sMode)

ModeSelection(String sMode): Function that stop the robot and send the mode selected. Used by the buttons listeners.

- protected void onDestroy():

When App is closed from the phone, Bluetooth communication is killed with the app.

More details about addContentView() on: this link.

Bluetooth Communication

The Bluetooth Communication is made of numerous methods that will be described. I recommend you to read this page about Bluetooth operating and its characteristics. Most part of the code comes from this site.

- public void initiate_Bluetooth():

This method calls the following methods or create the objects related to them.

- public boolean Bluetooth_Connection():

This method only has one goal: verifying that Bluetooth can be used on the phone, and if so, activate it on the phone. There are two ways to activate the Bluetooth. By defining the App as an administrator and activate it without the user permission, or by creating an automatic pop-up to ask the permission for the Bluetooth activation. The second possibility was chosen.

- private final BroadcastReceiver bluetoothReceiver = new BroadcastReceiver() {public void onReceive(Context context, Intent intent){}}:

_We instantiate a new BroadcastReceiver that watches the Bluetooth State and calls Current_State_Bluetooth(Intent intent) when changing. More details on: this link.

- public void Current_State_Bluetooth(Intent intent):

This method changes the textView that gives the Bluetooth state. It is called each time the user turn on or turn off the Bluetooth while the App runs.

- protected void onActivityResult(int requestCode, int resultCode, Intent data):

_This method is called if Bluetooth has turned on. It will search for all the paired devices and add them to the spinner (by adding them first in the ArrayList and them calling the ArrayAdapter as explained before). We also get the UUID (unique Bluetooth identifier) of each paired device and its name to be able to create a common socket with them later. More details on: this link.

- void openBT(UUID uuid, BluetoothDevice mDevice) throws IOException:

This method is used to create the socket between the two devices (phone and HC-05 of the robot) using the UUID. We created the input stream and the output stream of the socket. If the connection is successful, we add a “DISCONNECT” button in the spinner and show a short message (Android Studio Toast) for a few seconds saying “Connected”. If the connection aborted, we show a Toast saying “NOT Connected”.

- void sendData(String m) throws IOException

This method is called in OnJoystickMoved() with one of the 9 positions as parameter. It is important to surround it with a try/catch because Bluetooth communication does not permit us to know if one of the devices closed the socket.

- void beginListenForData()

This method gets the entering Bluetooth data sent from the robot sensors that is then displayed on the TextViews (but display is not treated by this function).

The Input Stream is read until a complete byte is received. The bytes are put in an array while the Input Stream is open. Then the function reads each byte from the first to the last received to find an ASCII letter (a=97, b=98, c=99, t=116, x=120, y=121). The value sent by the sensor follows this letter. We take the next bytes as corresponding value until we get an ASCII space (space=32), and we proceed again.

- public void ByteTreatement(byte[] incomingByte, int iLenght) throws UnsupportedEncodingException

_This method treats the bytes received by the previous method beginListenForData() and update the TextViews with the right data.

a + Value changes distanceFromObstacle,

b + Value changes leftEncoder

c + Value changes rightEncoder

x + Value changes xPosition

y + Value changes yPosition

t + Value changes robotOrientation_

activity_main.xml (layout management)

This file contains all the visual design elements (Spinners, TextViews, Buttons..). It is divided in FrameLayout (link to documentation) and LinearLayout (Link to documentation). They define many parameters of the App view such as : -width/height/weight of elements, backgrounds, orientation of elements (horizontal/vertical).

⚠ It is very important to understand how this file is built to understand how the App manages the different Views.

Miscellaneous

- ic_launcher_background and ic_launcher_foreground are used to define the image of the App on the phone menu.
- string.xml defines the App name.

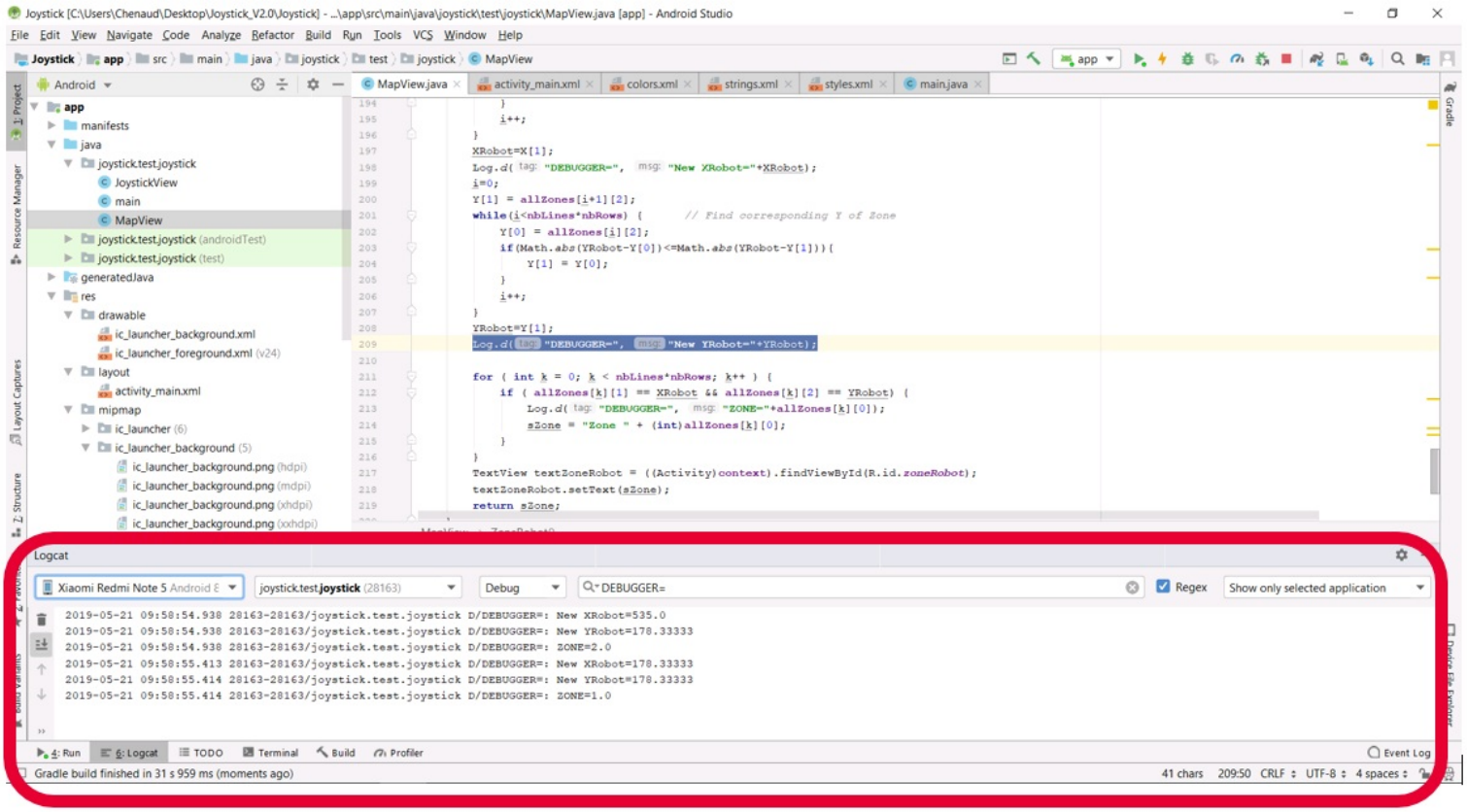
Some help to develop and debug

If you can’t load the App on a phone and meet the common warning : INSTALL_FAILED_INVALID_APK: Split lib_slice_0_apk was defined multiple times Just >Clean Project >Rebuilt Project

https://developer.android.com/ is a platform made for Android developers. You can find everything you need.

https://stackoverflow.com/ for programming problems. Somebody already had the problem you have and someone else told him how to solve it!

For real-time debugging, you will find many time in the code the following line : Log.d("DEBUGGER=", "XXXXXX="+YYYYYYYYY); Then you can read in the debugger (called Logcat, in the bottom-right corner of Android Studio) variables during App use.



Written by Bastien Chenaud - 14/01/2019 V_1.1

Updated on 15/02/2019

Updated on 21/05/2019