

DOCUMENTATION SDK

SDK Buddy / référence / 2.0

Date	Version	Auteur	Revision	Commentaire
05-11-2021	V2.0.0	Victor Menigault	-	-

[Titre]

[Date de publication]

I. SOMMAIRE

Documentation SDK 1

Sommaire 2

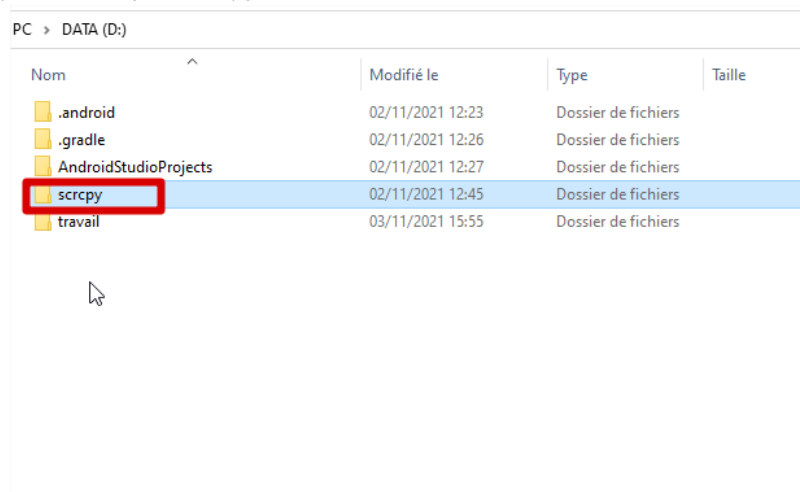
1. Create your Own App
 - a. Connect the robot to your computer
 - b. Settings and creation of your own app
 - c. Configure your programming files
 - d. Building and running your app on Buddy
2. Moving Buddy
 - a. Move straight with a certain speed and distance
3. Titre de 2ème niveau 3
4. Sous-titre de 3ème niveau 3
5. Eléments type liste et tableau 4
 - a. Eléments listes 4
 - b. Liste à puces 4
 - c. Liste numérotée 4
6. 4 - Mise en forme des images 5
 - a. Image pleine largeur 5
 - b. Image latérale 6
 - c. Mosaïque d'images 6
7. 5 - Codes Couleurs 7
 - a. Bleu BUDDY 7
 - b. Gris sombre 7
 - c. Eléments tableau 7
 - d. Eléments cliquable 7

1 – Create your own app

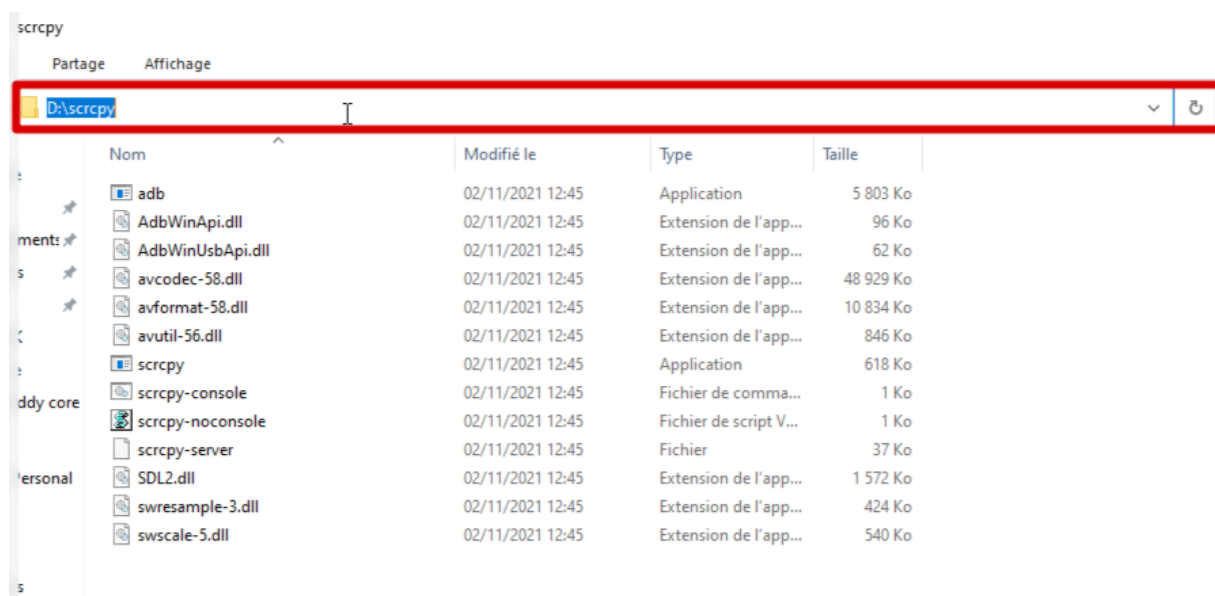
A) Connect the robot to your computer

Step 1 - Download scrcpy and Android Studio

Step 2 - Go to your scrpy file and click on it



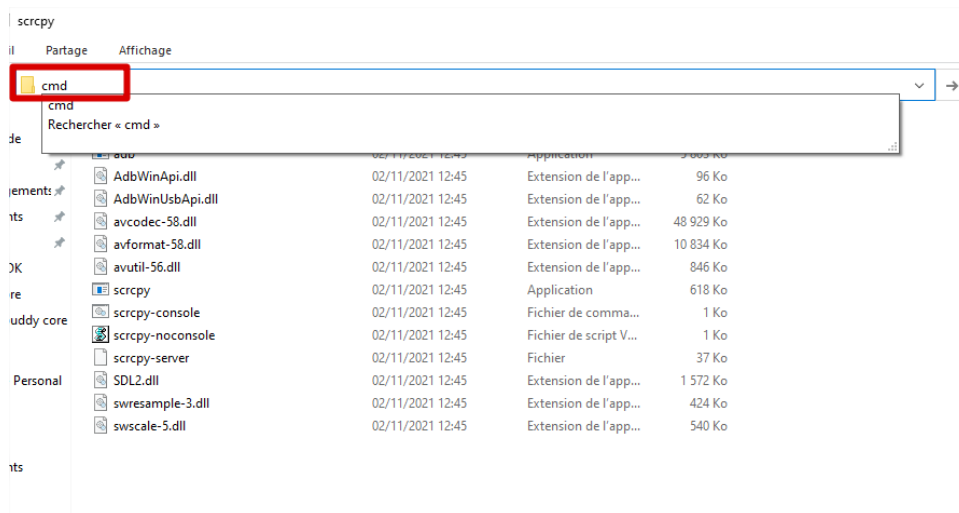
Step 3 - Click on the file bar



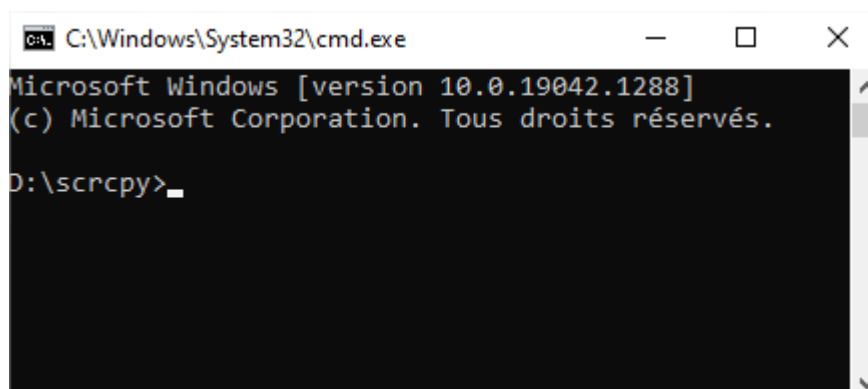
Step 4 - Type cmd and press enter

[Titre]

[Date de publication]

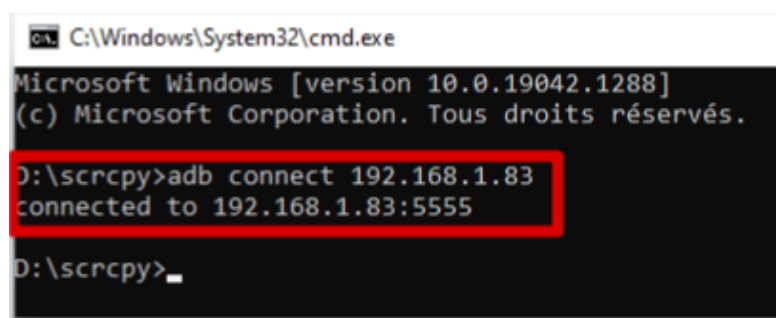


Step 5 - A command window must appear like this :



Step 6 - In This command window write : " adb connect robot's Ip Adress "

And press Enter (The Ip Adress is starting like the following form : 192.168.etc...)



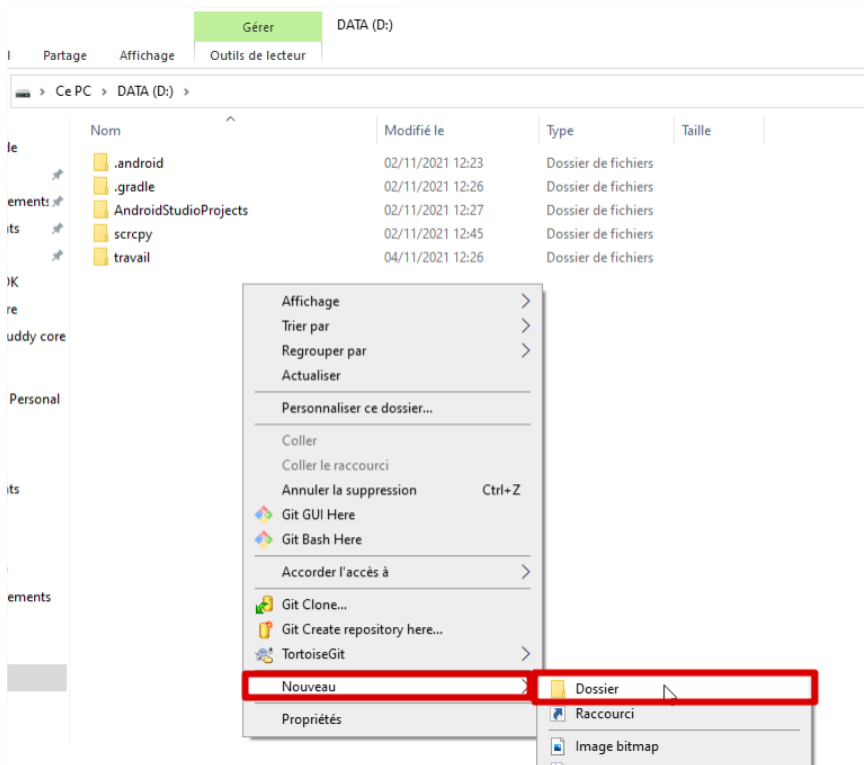
The robot is now connected to the computer

B) Settings and creation of your application's project

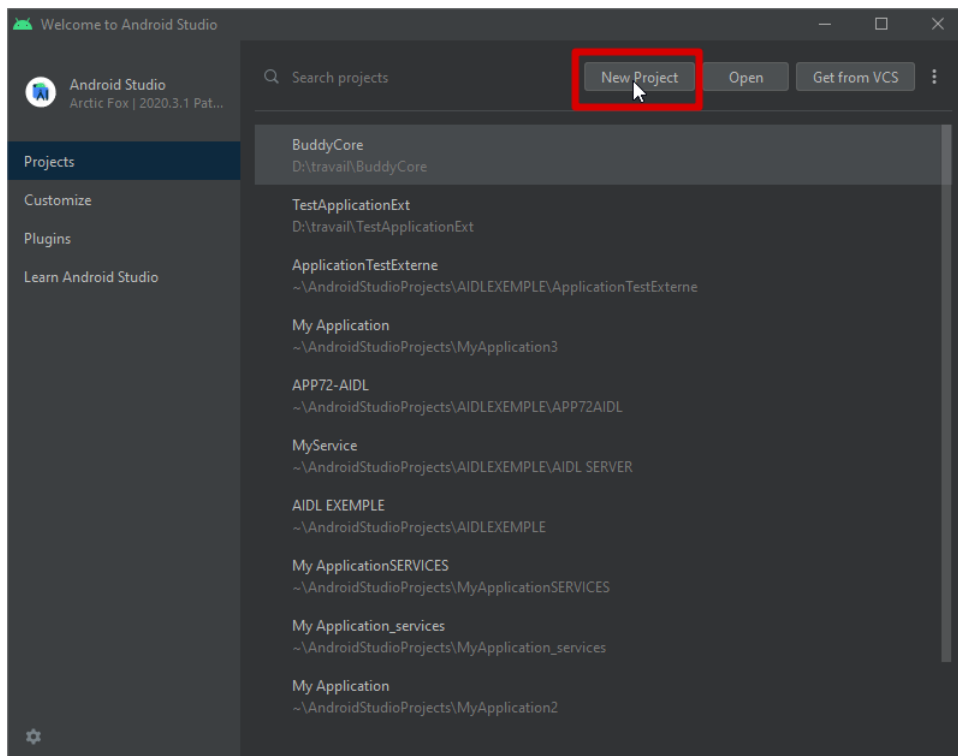
[Titre]

[Date de publication]

Step 1 - Create a new file in which your project will be saved



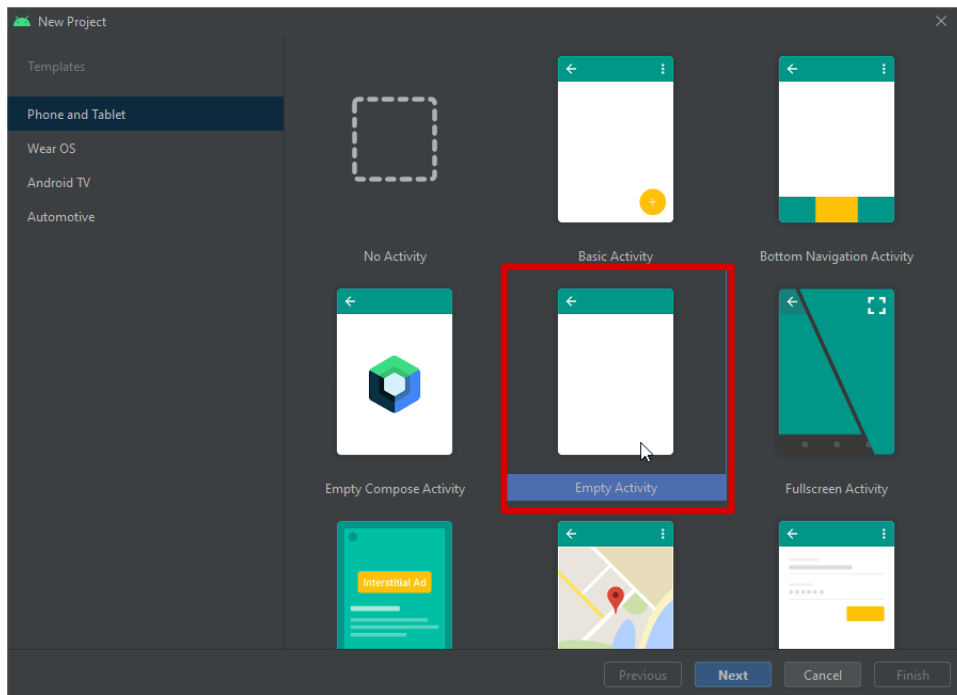
Step 2 – Open Android Studio and select “New project”



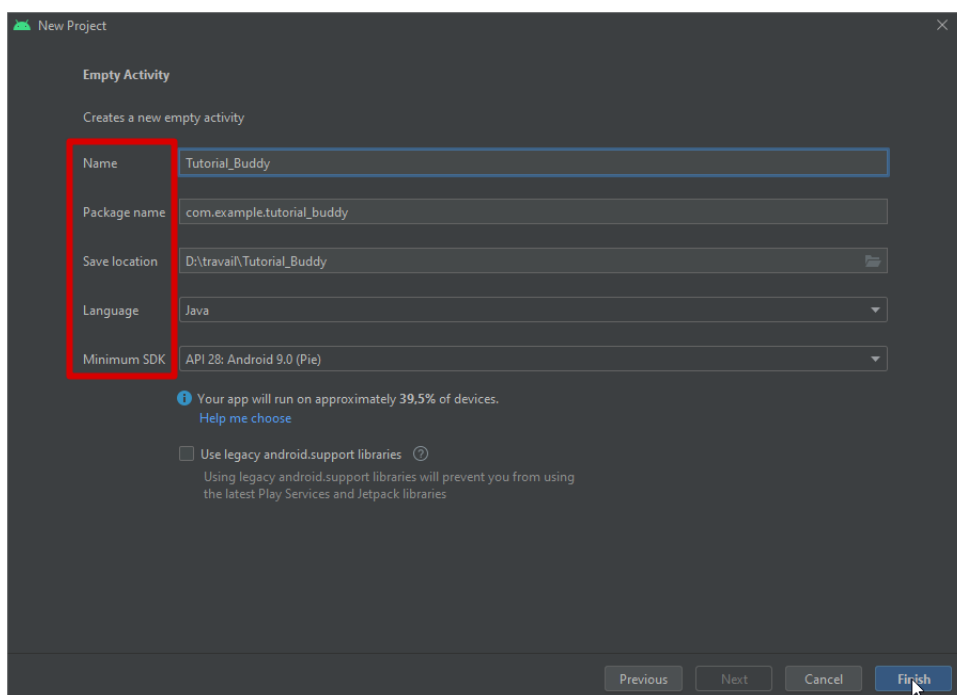
Step 3 – Select the “Empty activity” template and click “Next”

[Titre]

[Date de publication]



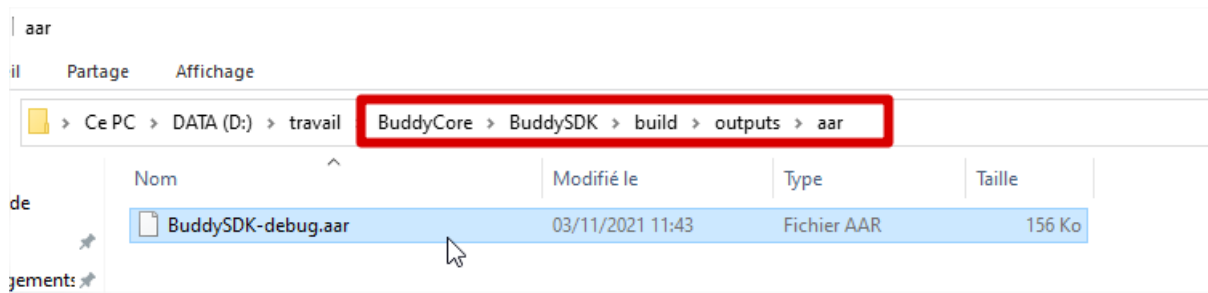
Step 4 – Select your file's preferences



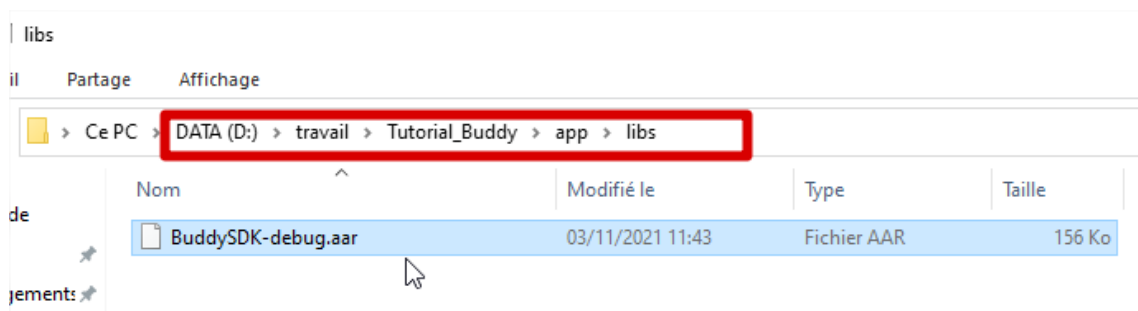
Step 5 – In your file explorer copy the BuddySDK-debug.aar file in the BuddyCore file

[Titre]

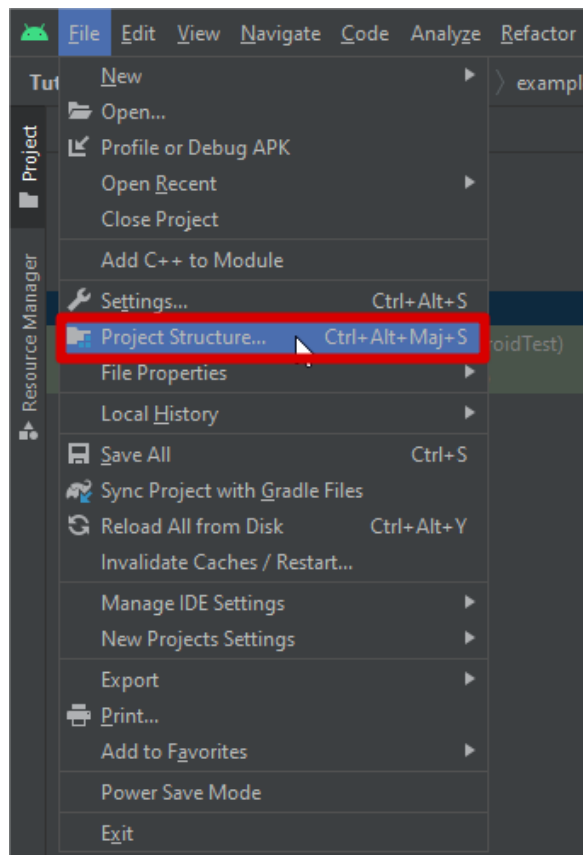
[Date de publication]



Step 6 - Paste this file in the “libs” of your file project you just created



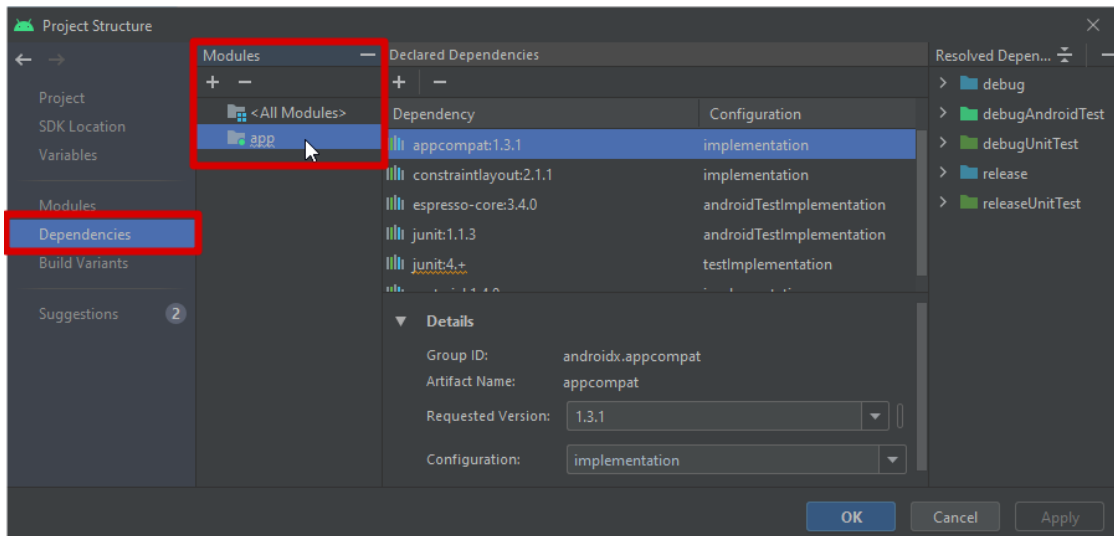
Step 7 – Before anything you have to add the SDK file. For this click on File > Project Structure



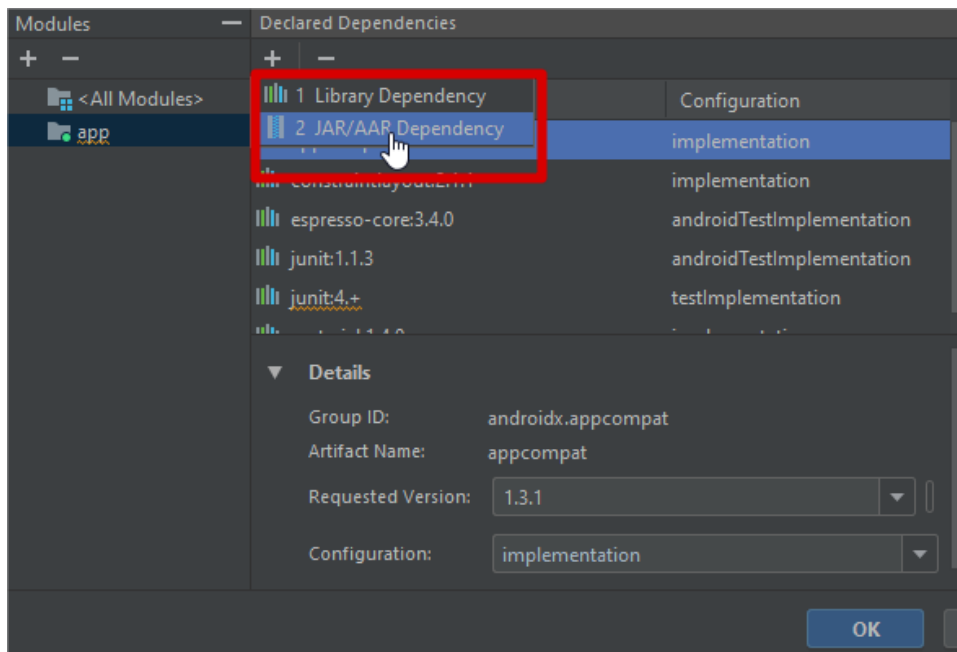
[Titre]

[Date de publication]

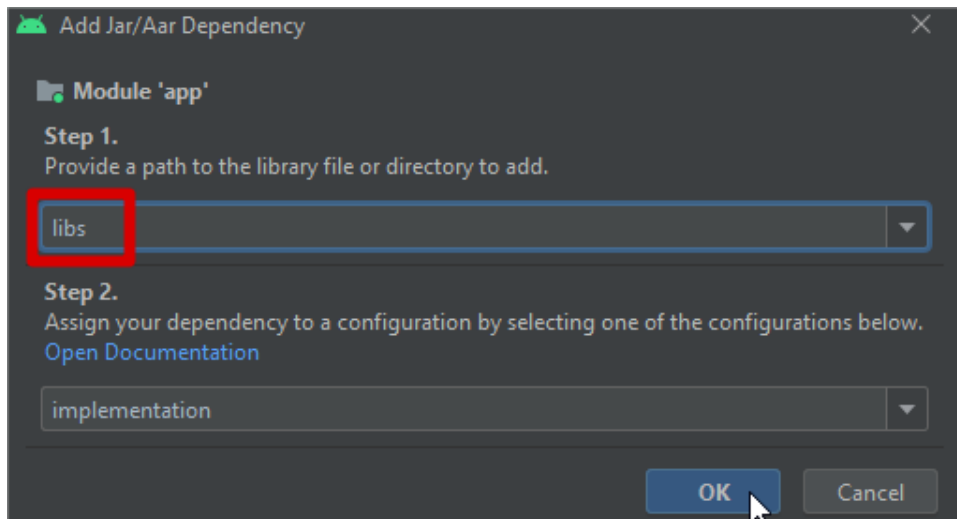
Step 8 – Once you are on the project structure window, click on Dependencies. Then in Modules click on the name of your app, here it's named “app”



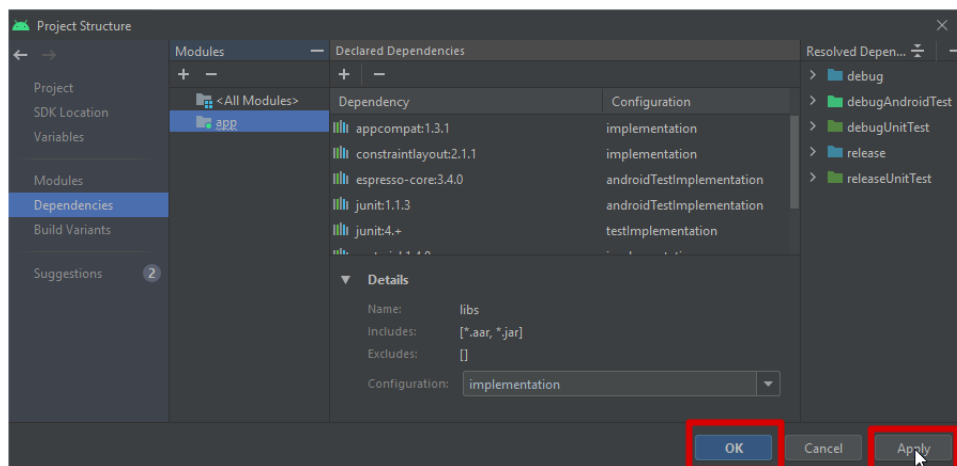
Step 9 – In “Declared Dependencies” click on the “+” symbol and select AAR



Step 10 – Type “libs”, this is the location of the file, and click on “OK”



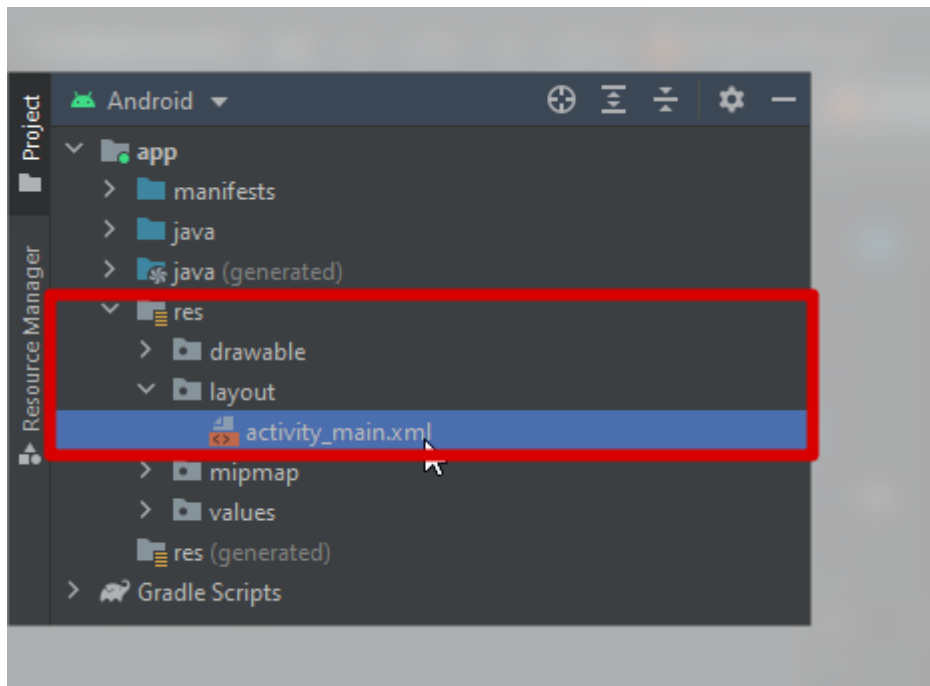
Step 11 – Then click on “Apply” and on “OK”



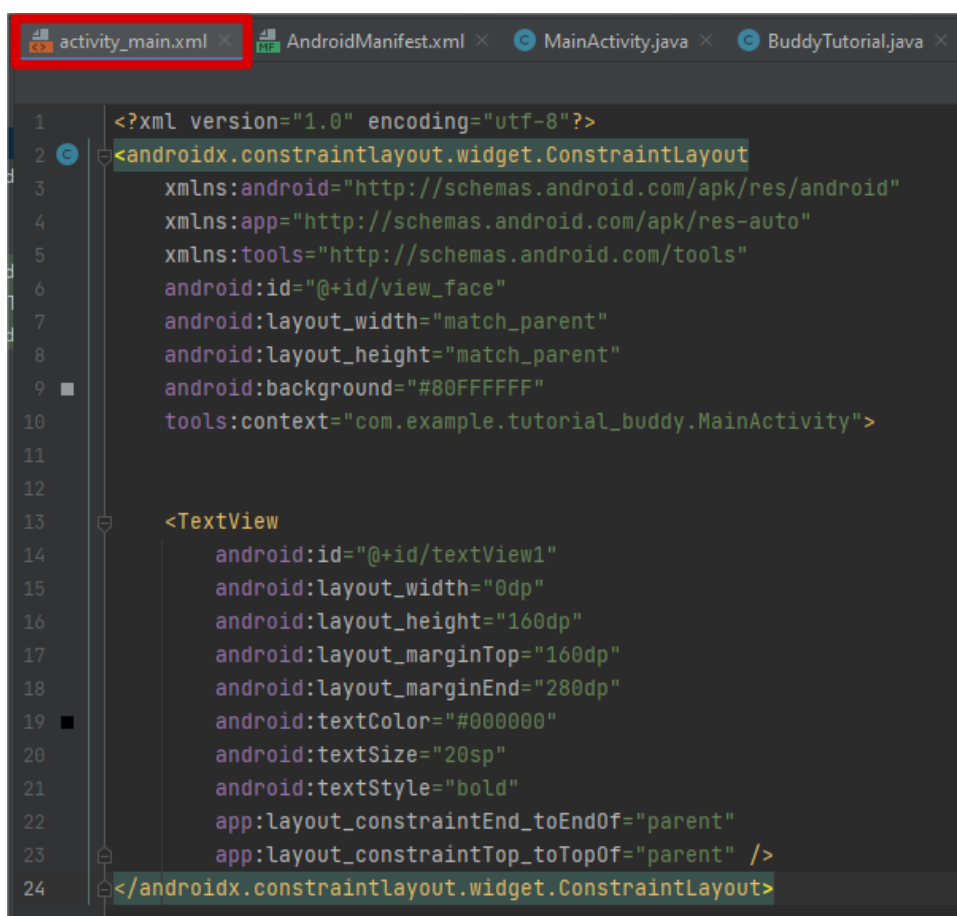
All the technical parameters are now set and your project is created. We can now pass to building the Application

C) Configure your programming files

Step 1 – Now you can start your project. First to define your graphic interface click on Project > res > layout > activity_main.xml



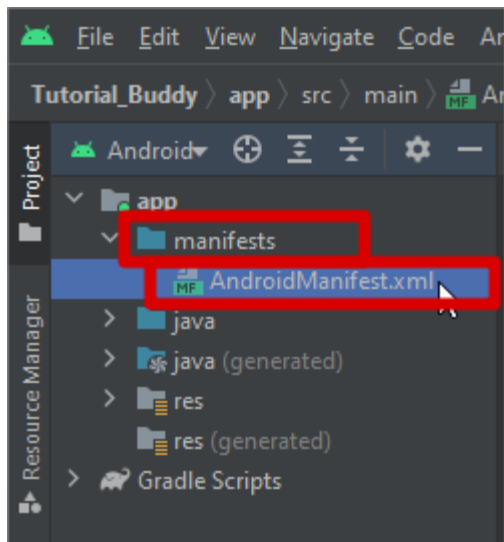
Step 2 – In the activity_main.xml file write this code



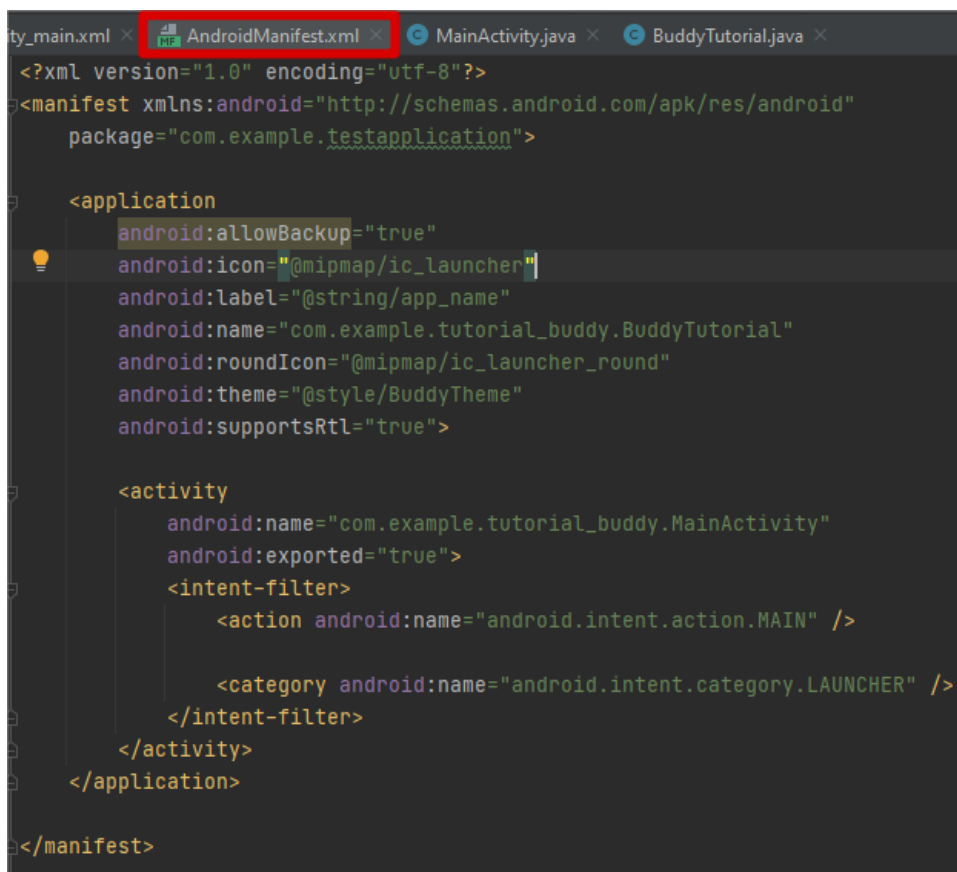
Step 3 – Go to manifests>AndroidManifest.xml

[Titre]

[Date de publication]



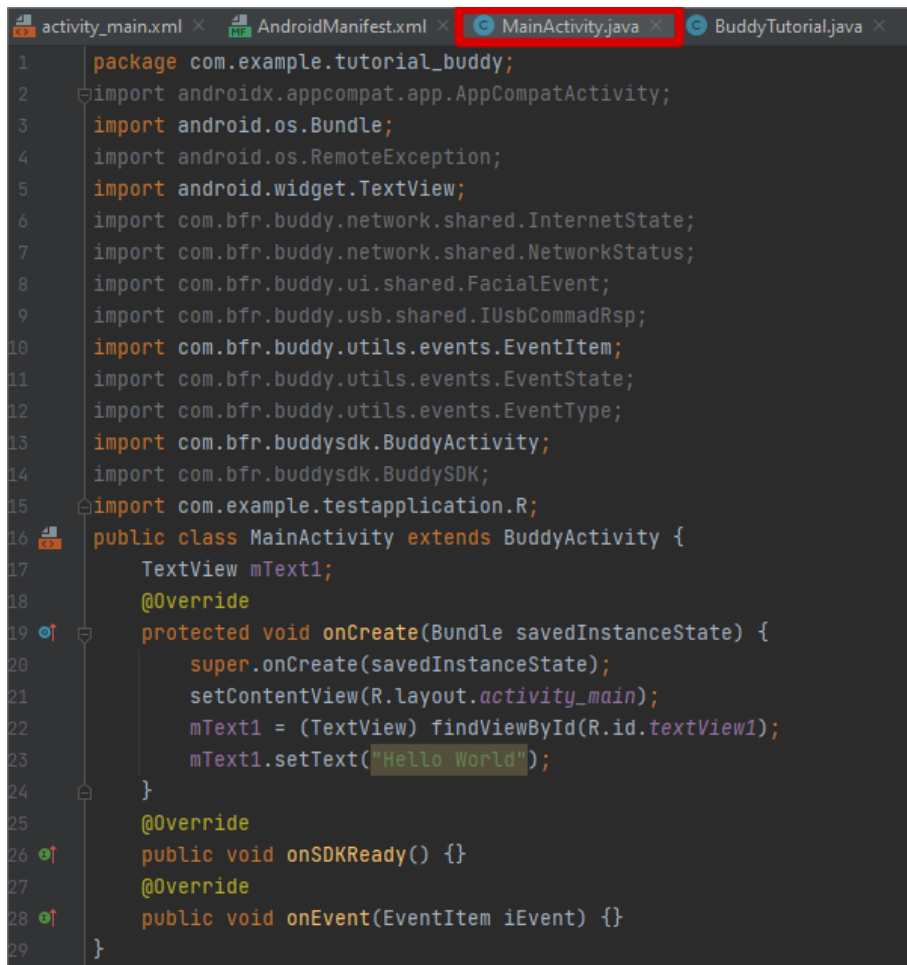
Step 4 - In the AndroidManifest.xml modify your code so it looks like that:



Step 5 - In the MainActivity.java write this basic code

[Titre]

[Date de publication]

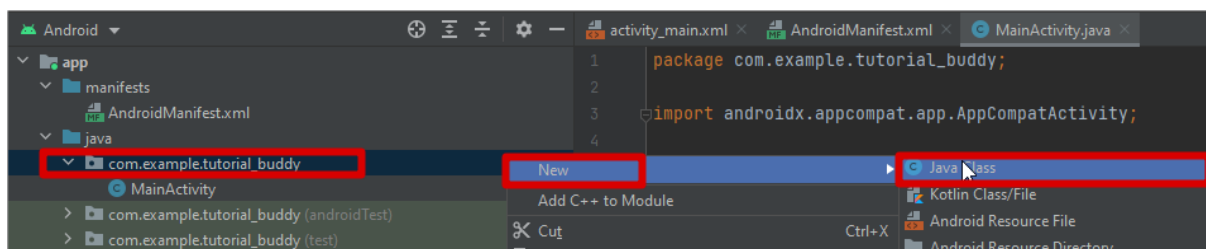


```

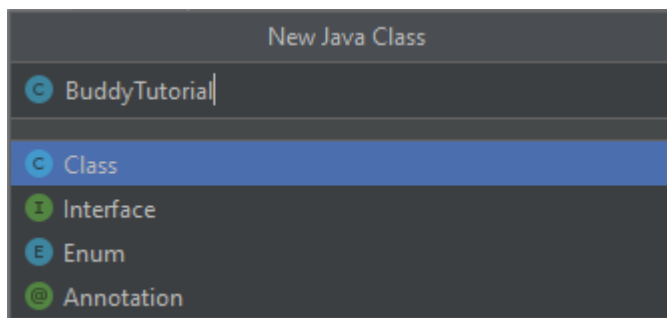
1 package com.example.tutorial_buddy;
2 import androidx.appcompat.app.AppCompatActivity;
3 import android.os.Bundle;
4 import android.os.RemoteException;
5 import android.widget.TextView;
6 import com.bfr.buddy.network.shared.InternetState;
7 import com.bfr.buddy.network.shared.NetworkStatus;
8 import com.bfr.buddy.ui.shared.FacialEvent;
9 import com.bfr.buddy.usb.shared.IUsbCommadRsp;
10 import com.bfr.buddy.utils.events.EventItem;
11 import com.bfr.buddy.utils.events.EventState;
12 import com.bfr.buddy.utils.events.EventType;
13 import com.bfr.buddysdk.BuddyActivity;
14 import com.bfr.buddysdk.BuddySDK;
15 import com.example.testapplication.R;
16 public class MainActivity extends BuddyActivity {
17     TextView mText1;
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22         mText1 = (TextView) findViewById(R.id.textview1);
23         mText1.setText("Hello World");
24     }
25     @Override
26     public void onSDKReady() {}
27     @Override
28     public void onEvent(EventItem iEvent) {}
29 }

```

Step 6 – In the project three, in Java, click right on the “MainActivity” file and select “New” > “Java Class”



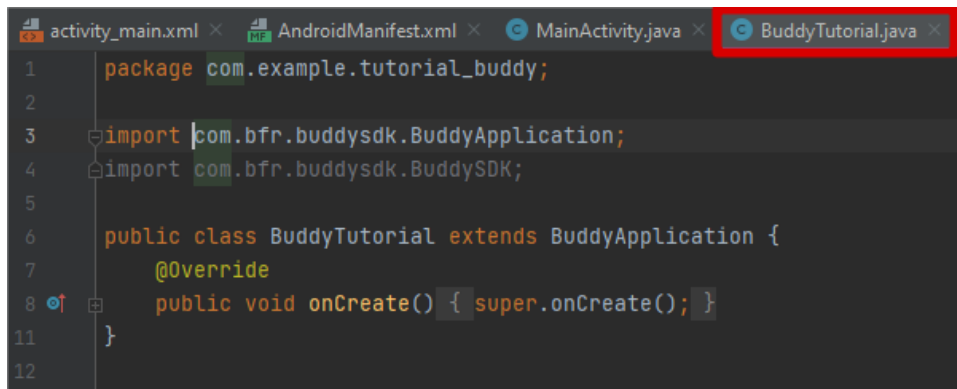
Step 7 – Give it a name and press “Enter”



Step 8 - Once you created a new class write this code in it :

[Titre]

[Date de publication]

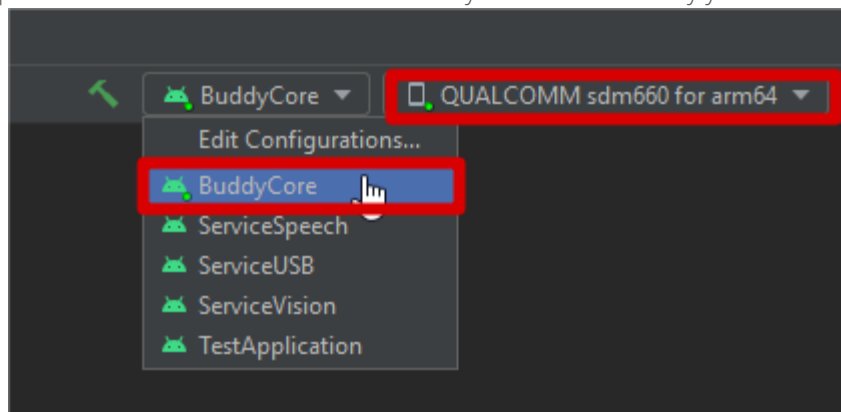


```
1 package com.example.tutorial_buddy;
2
3 import com.bfr.buddysdk.BuddyApplication;
4 import com.bfr.buddysdk.BuddySDK;
5
6 public class BuddyTutorial extends BuddyApplication {
7     @Override
8     public void onCreate() { super.onCreate(); }
9
10 }
11
12
```

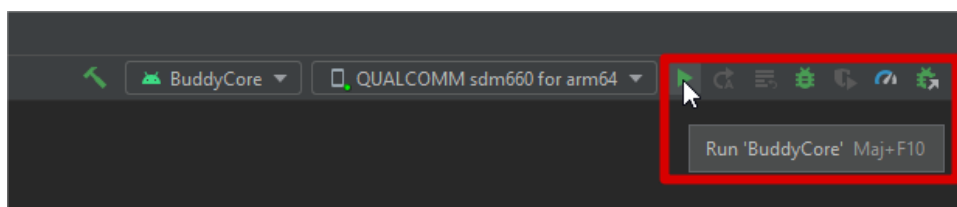
The programming base is now set, we can pass to building the app

D) Building and running your app on Buddy

Step 1 – In Android Studio select the BuddyCore file and verify you’re aiming at the robot



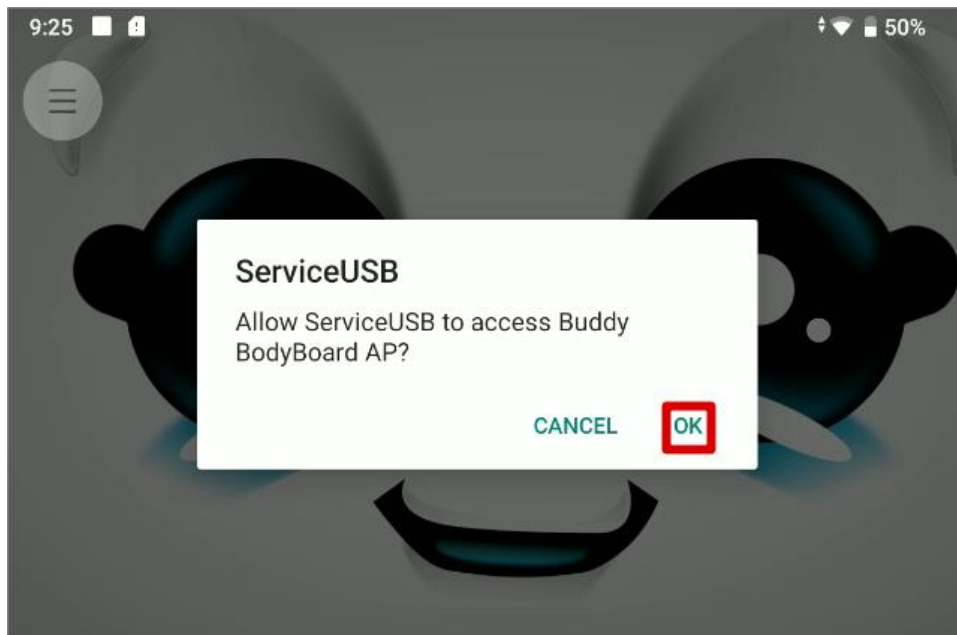
Step 2 – Then click on “run”



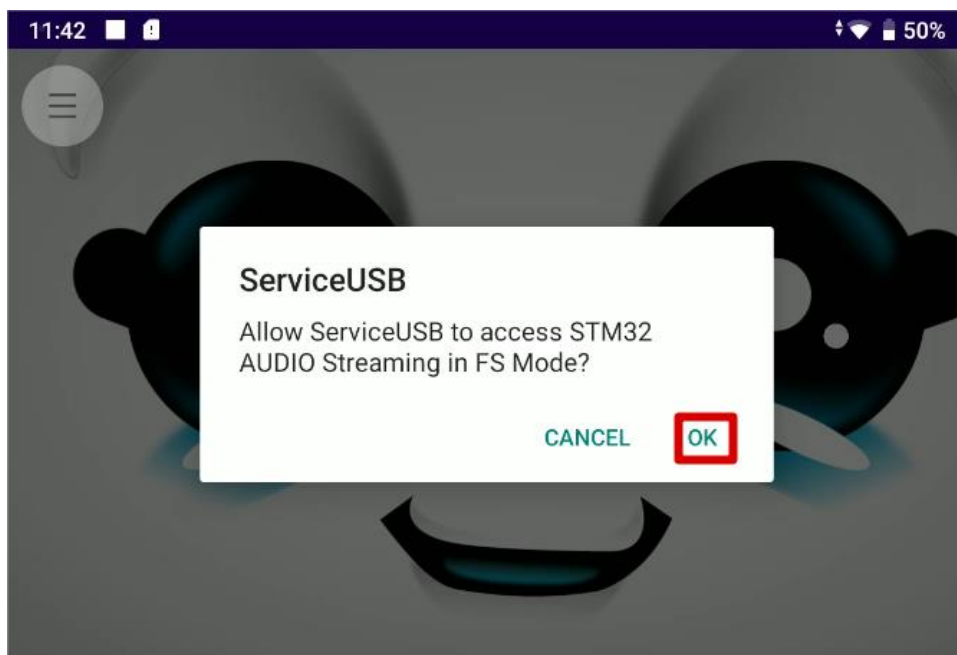
Step 3 – On your robot, click on “OK” to Allow ServiceUSB to access Buddy BodyBoard AP

[Titre]

[Date de publication]



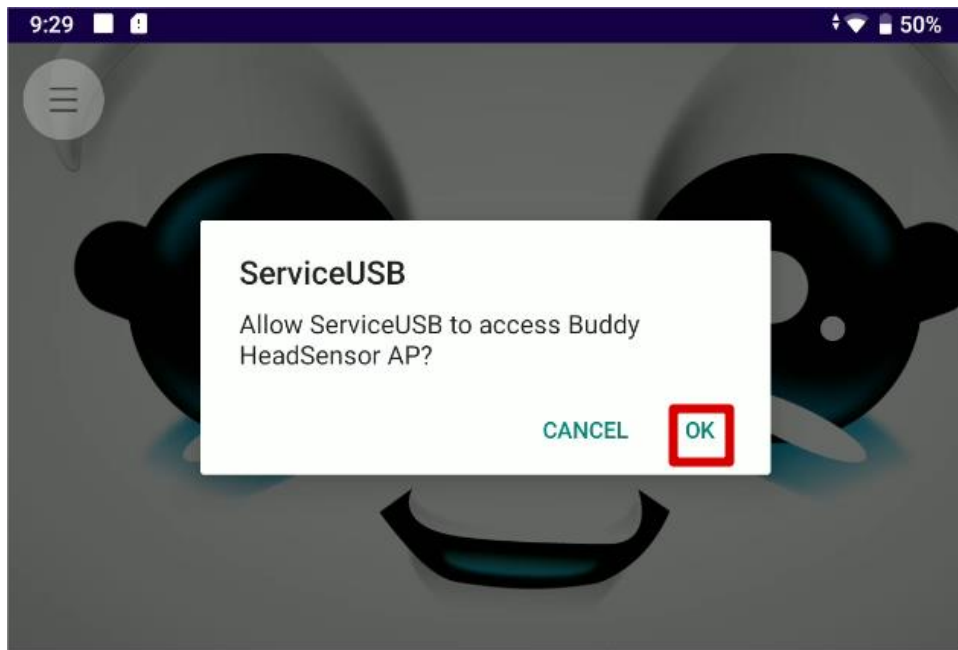
Step 4 – Then click on “OK” to allow AUDIO Streaming



Step 5 – Finally click on “OK” to “Allow ServiceUSB to access Buddy HeadSensor AP”

[Titre]

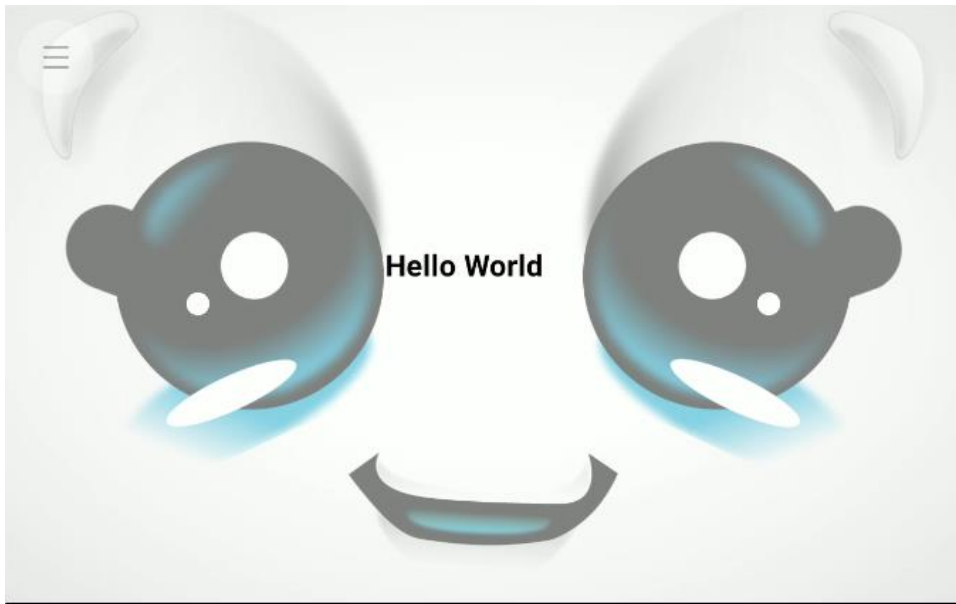
[Date de publication]



Step 6 – Now that you launched BuddyCore you must see the Buddy's face appear on your screen



Step 7 – Then launch the application project you crated. If you complited each tasks presented in this tutorial, you might see an "Hello world" appear on the screen



Now everything is set so you can start programming your own application

2 – Moving Buddy

l) Moving straight with a certain speed and distance

1) Usefull functions

1) Start the wheels

Before doing any wheels movements you have to start the wheels motor. Here is the function we will use for it :

```
int iLeft, int iRight, IUsbCommadRsp iCallback  
BuddySDK.USB.enableWheels();
```

This function has got 3 parameters :

Int iLeft => the left wheel (iLeft = 0 => off ; iLeft = 1 => on)

Int iRight - the right wheel (iRight = 0: off; iRight = 1 => on)

IUsbCommadRSP iCallback - returns "something you set" onSuccess, "something you set" onFailed

[Titre]

[Date de publication]

Do not forget to add in the "import" section :

```
import android.os.RemoteException;
```

2) Start Buddy moves :

Important ! This function won't be working if you don't start the wheels motor with the function presented in 1) Start the wheels

To make buddy move, write this function:

```
float iSpeed, float iDistance, IUsbCommadRsp iRspCallback  
BuddySDK.USB.moveBuddy(iSpeed, iDistance, iRspCallback);
```

This function has got 3 parameters :

Float iSpeed = the robot speed (m/s), (+): forward, (-): Backward, max: 0.7m/s

Float iDistance = distance to reach (Meter)

IUsbCommadRSP iCallback - returns "something you set" in the case of onSuccess, "something you set" in the case of onFailed

Do not forget to add in the "import" section :

```
import android.os.RemoteException;
```

In the following part you will be able to see a tutorial about how to make those functions work and how to make Buddy move

2) Concrete example

1) Manifest file Setting

Go the the AndroidManifest.xml file and change the android:theme by :

android:theme="@style/BuddyTheme"

```
ty_main.xml x AndroidManifest.xml x MainActivity.java x BuddyTutorialTest.java x
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.testapplication">

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Tutorial_Buddy_test"
    android:name="com.example.tutorial_buddy_test.BuddyTutorialTest"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:theme="@style/BuddyTheme"
    android:supportsRtl="true">

    <activity
      android:name="com.example.tutorial_buddy_test.MainActivity"
      android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

If you do not know where to find the manifest, then find how on the tutorial of 1- Create your own app
> C) Configure your programming files

2) Layout

I am bout to create a kind of app o decide when buddy has to move.

First I define the layout.

For this I start bu adding 2 buttons :

The first will be to start the wheels, he will command the first function described in 1) Start the wheels

The second button will command the start of the deplacment of our robot described in 2) Start Buddy moves

```

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/view_face"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#80FFFFFF"
    tools:context="com.example.avance.MainActivity">

```

At the start of your code it's important to add the red squared lines for the interface to be correct.

Then add 2 buttons we talked about :

```

<Button
    android:id="@+id/button_advance"
    android:layout_width="200dp"
    android:layout_height="50dp"
    android:text="Advance"
    android:textSize="20sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.209"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.385" />

<Button
    android:id="@+id/button_enable_wheels"
    android:layout_width="200dp"
    android:layout_height="50dp"
    android:text="Enable wheels"
    android:textSize="20sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.209"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.152" />

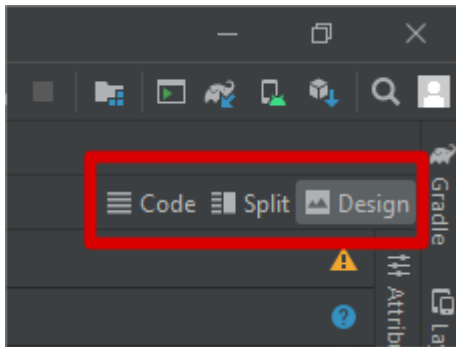
```

The "advance" button is used to, once it's clicked on, start the motor.

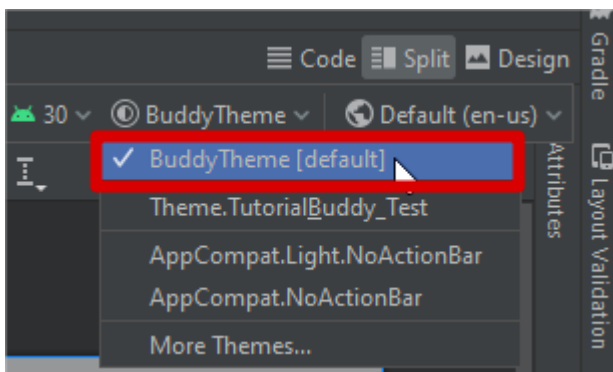
The "Enable wheels" button is used to, once it's clicked on, start the wheels motor.

Important ! Give each of the buttons an Id, it will be important in the programming of the java file.

Finally, on the top right of your screen, go in the Design part



choose "BuddyTheme[default]"



3) MainActivity java file

Now we are in the most important part of your project : the programming part.

Step 1 – Go in the MainActivity Java file and start by importing the needed functions. Here are the ones I needed for making the robbot advance:

```
import static android.service.controls.ControlsProviderService.TAG;
import android.os.Bundle;
import android.os.RemoteException;
import android.util.Log;
import android.widget.Button;
import android.widget.TextView;
import com.bfr.buddy.utils.events.EventItem;
import com.bfr.buddy.usb.shared.IUsbCommadRsp;
import com.bfr.buddysdk.BuddyActivity;
import com.bfr.buddysdk.BuddySDK;
import com.example.testapplication.R;
```

Step 2 – In the public class MainActivity create 2 buttons you'll link to the two you created in the layout

```
public class MainActivity extends BuddyActivity {
    TextView mText1;//defining a text parameter so we show the text we want
    Button mButtonEnable ;//definning buttons Enable ( will be used to enable wheels motors )
    Button mButtonAdvance;//definning buttons Advance ( will be used to make buddy advance )
```

[Titre]

[Date de publication]

In the onCreate location, we can see the 2 ID's of the buttons and of the textView we crated in the layout file in part 2) Layout

```
//link with user interface
mText1 = findViewById(R.id.textView1);
//linking the id of the buttons of Layout to buttons in the code
mButtonEnable=findViewById(R.id.button_enable_wheels);
mButtonAdvance = findViewById(R.id.button_advance);
mText1.setText(" "); //setting no text
```

Step 3 – In onCreate, set the button enable_wheels OnClickListener, this will execute your command if the button get clicked

```
// Listener for button to enable the wheels
mButtonEnable.setOnClickListener(view -> {
```

We will use the enableWheels function for this part

```
int iLeft, int iRight, IUsbCommadRsp iCallback
BuddySDK.USB.enableWheels();
```

Step 4 – As we want the motor wheels to start we decide to set the parameters of the enableWheels function commanded by the button set in Step 3 in the onCreate location

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // Log.i(TAG, "wheels create");
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //link with user interface
    mText1 = findViewById(R.id.textView1);
    //linking the id of the buttons of Layout to buttons in the code
    mButtonEnable=findViewById(R.id.button_enable_wheels);
    mButtonAdvance = findViewById(R.id.button_advance);
    mText1.setText(" "); //setting no text

    // Listener for button to enable the wheels
    mButtonEnable.setOnClickListener(view -> {
        int turnOnRightWeel = 1;//setting the right weel motor on On of "int" type (On=1) (Off=0)
        int turnOnLeftWeel = 1;//setting the Left weel motor on On of "int" type (On=1) (Off=0)

        BuddySDK.USB.enableWheels(turnOnLeftWeel, turnOnRightWeel, new IUsbCommadRsp.Stub() {
```

As in this example we want the robot to go straight we activate both motors of each wheel

Step 5 – Now we can see the third parameter of the enableWheels function, the IUsbCommadRsp.Stub().

We have 2 callbacks function and you can those function to do some task if you want to be informed about the `enableWheels` function success or failure. Here we decide to show a message describing the state of success or failure

```
BuddySDK.USB.enableWheels(turnOnLeftWheel, turnOnRightWheel, new IUsbCommadRsp.Stub() {

    @Override
    public void onSuccess(String s) throws RemoteException {
        //in Case of success of enabeling the wheels we decide to show some text at screen
        mText1.setText("wheels are on ");
        Log.i(TAG, "wheels are on");
    }

    @Override
    public void onFailed(String s) throws RemoteException {
        //In case of failure we want to be inform of the reason of the failure
        Log.i(TAG, "Wheels enable failed because :" + s);
    }
});
```

Now the motor of each wheel is set on, we can pass to the part of advancing the robot

Step 6 – In onCreate, set the button `mButtonAdvance` `OnClickListener`, this will execute your `AdvanceFunction` if the button get clicked

```
// Listener for button to make the robot go forward or backward
mButtonAdvance.setOnClickListener(view -> AdvanceFunc());
//in case of click on the button, call of the function AdvanceFunc()
```

We will use this following function to move buddy

```
float iSpeed, float iDistance, IUsbCommadRsp iRspCallback
BuddySDK.USB.moveBuddy();
```

Step 7 – Define the speed and distance settings for the `moveBuddy()` function in the `AdvanceFunc()`

```
//function called for clicking on the Advance button
private void AdvanceFunc() {
    //Here we decide to go forward of 0.5meters with a 0.5m/s speed
    float speed = 0.5F; //definition of the speed (>0 to go forward , <0 to go backward)
    float distance = 0.5F; //definition of the distance to praccour(ALWAYS>0)
```

Step 8 – Write the `moveBuddy` function with each of its defined parameter

```
private void AdvanceFunc() {
    //Here we decide to go forward of 0.5meters with a 0.5m/s speed
    float speed = 0.5F; //definition of the speed (>0 to go forward , <0 to go backward)
    float distance = 0.5F; //definition of the distance to praccour(ALWAYS>0)

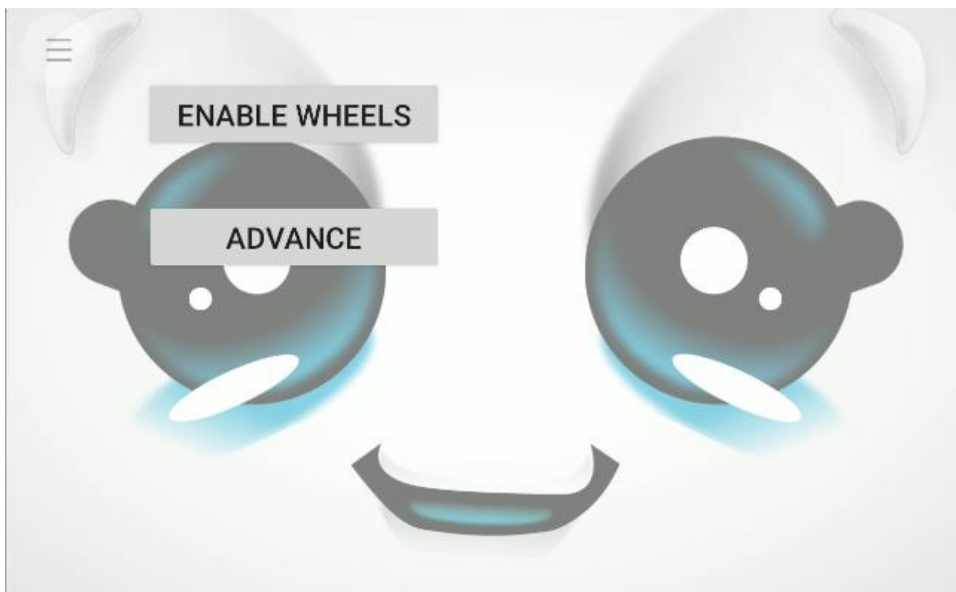
    //call of the function to make buddy go forward or backwars
    BuddySDK.USB.moveBuddy(speed, distance, new IUsbCommadRsp.Stub() {
```

Step 9 – Now let's see the callbacks function set by `IUsbCommadRsp.Stub()` of this `moveBuddy` function. Once again you can set what you want in the yellow underline zone to ensure of sucess or failure of each part.

```
//call of the function to make buddy go forward or backwars
BuddySDK.USB.moveBuddy(speed, distance, new IUsbCommadRsp.Stub() {
    @Override
    public void onSuccess(String s) throws RemoteException {
        Log.i(TAG, msg: "AdvanceFunct: sucess");//in case of sucess show in the logcat window 'sucess'
    }

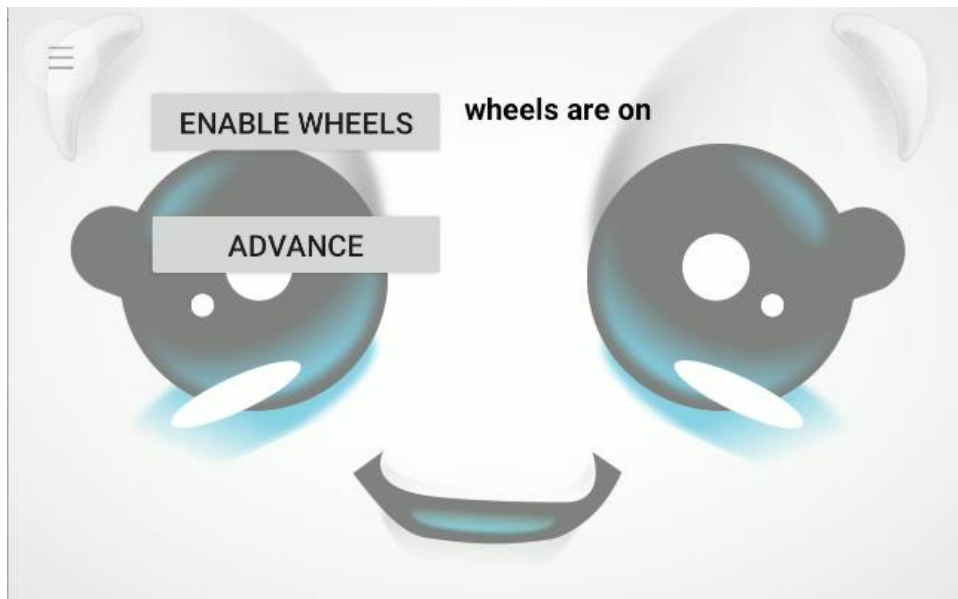
    @Override
    public void onFailed(String s) throws RemoteException { //in case of failure to achieve this function
        Log.i(TAG, msg: "AdvanceFunct: fail");//show in the logcat window a message
        mText1.setText("Fail to advance");//show on the screen of the robot 'Fail to advance'
    }
});
```

Step 10 – Final task :launch BuddyCore and then launch the app we just created and you should see this appeared on the screen of your robot :



Now you just have to click on Enbale Weels and Then on Advance and your app is now working.

This message happens when we click on enable Wheels, it shows wheels are on an the robot can start moving



We showed You how to use functions in android studio, the idea is the same for every other moving robot's functions

II) Other functions to move the robot

As we saw in the previous part how to use the functions we will now focus only on the function, its parameters, its use.

1 WheelRotate

[WheelRotate](#)(float speed, com.bfr.usbservice.IUsbCommadRsp RspCallback)

Start the robot to turn on itself at a given speed.

```
float iSpeed, IUsbCommadRsp iRspCallback
BuddySDK.USB.WheelRotate();
```

Parameters :

Float iSpeed = give the speed of the rotation of the wheel.

The speed is between -0.7 m/s to 0.7m/s

The callback return "Success" in case of success and "fail" in case of failure

(Works but automaticly goes to onFail case)

[Titre]

[Date de publication]

2 Rotate Buddy (rotation speed + defined angle)

[rotateBuddy](#)(float speed, float angle, com.bfr.usbservice.IUsbCommadRsp RspCallback)

To rotate the robot at a given angle and speed call this function

```
//function to rotate Buddy with a certain angle
BuddySDK.USB.rotateBuddy(iSpeed, iDegree, new IUsbCommadRsp.Stub() {
```

Parameters :

Float iSpeed : define the speed in degree per seconds

Float iAngle : define the angle in degree you want the robbot to rotate

The callback return "Success" in case of success and "fail" in case of failure

3 Stop Motors

[emergencyStopMotors](#)(com.bfr.usbservice.IUsbCommadRsp RspCallback)

To Stop the motors immediately with highest deceleration as possible call this function :

```
//function to stop motors
BuddySDK.USB.emergencyStopMotors(new IUsbCommadRsp.Stub() {
```

Parameters :

The callback return "Success" in case of success and "fail" in case of failure

4 Move Buddy

[moveBuddy](#)(float speed, float distance, com.bfr.usbservice.IUsbCommadRsp RspCallback)

Move the robot straight at a defined speed and distance.

Fully described in the part 2 – Moving Buddy

5 Enable the wheels of the robot

[enableWheels](#)(int left, int right, com.bfr.usbservice.IUsbCommadRsp RspCallback)

Enable the Buddy's motors

Fully described in the part 2 – Moving Buddy

6 Move straight undefinitly

[moveWheelStraight](#)(int speed, com.bfr.usbservice.IUsbCommadRsp RspCallback)

Start the robot to go straight at a given speed call this function :

[Titre]

[Date de publication]

```
int iSpeed, IUsbCommadRsp iRspCallback  
BuddySDK.USB.moveWheelStraight();
```

Parameters :

Float iSpeed= define the speed in m/second

The callback return "Success" in case of success and "fail" in case of failure

(Not working – advance but stop after 1 min)

We showed You how to use moving robot's functions. Now we will see the sensors

II) Functions using sensors

1

com.bfr.usbservice

Class BdbusServer.UsbImpl

java.lang.Object

android.os.Binder

com.bfr.usbservice.IUsbAidlInterface.Stub

com.bfr.usbservice.BdbusServer.UsbImpl

All Implemented Interfaces:

android.os.IBinder, android.os.IInterface, com.bfr.usbservice.IUsbAidlInterface

Enclosing class:

BdbusServer