

## ROS <=> Android tutorial

### Introduction

This tutorial is supposed to show you how to connect your Duckiebot to your Smartphone via Bluetooth. Tutorial will focus on connection and robot side programming to make Bluetooth available, not on Android Programming using Java (since this might be too complicated for those who never programmed for Android before). Instead program for Android will be created using MIT App Inventor web application to demonstrate proof of concept. Using Bluetooth on your robot is the core part of this exercise, not programming for your Smartphone.

---

### 1. Setting your Bluetooth device

First, install all required software for Bluetooth to work. All commands are executed on your Duckiebot, so connect to it now.

Install Bluetooth support (skip these two commands if you're using Ubuntu 14.04):

**☒ CSP STUDENTS 2017, SKIP THESE TWO COMMANDS, YOU HAVE UBUNTU 14.04! ☒**

```
duckiebot:~$ sudo apt-get update
duckiebot:~$ sudo apt-get install bluez bluetooth bluez-tools
```

↓ CSP STUDENTS 2017, USE COMMANDS BELOW TO INSTALL BLUEZ ↓

---

**[IMPORTANT]** If you're using Ubuntu 14.04, there's a problem with Bluez being too old. If you're using Ubuntu 14.04 (ALL CSP Duckiebots use 14.04 as for 2017), you need to install newer version of Bluez using these commands below:

Delete old Bluez:

```
duckiebot:~$ sudo apt-get purge bluez
```

Reboot your Duckiebot:

```
duckiebot:~$ sudo reboot now
```

Install new Bluez:

```
duckiebot:~$ sudo add-apt-repository ppa:vidplace7/bluez5
duckiebot:~$ sudo apt-get update
duckiebot:~$ sudo apt-get install bluez bluetooth bluez-tools
```

Plug in your Bluetooth adapter into your Duckiebot. Bring up your Bluetooth adapter:

```
duckiebot:~$ sudo hciconfig hci0 up
```

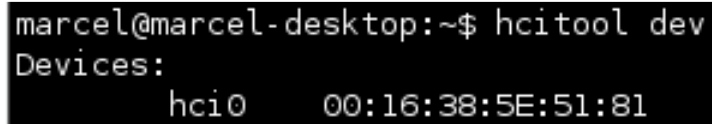
Reboot again, just to be sure. Never hurts to reboot.

```
duckiebot:~$ sudo reboot now
```

-----  
Check if adapter is seen in the system:

```
duckiebot:~$ hcitool dev
```

You should see something like this:

A terminal window with a black background and white text. The prompt is 'marcel@marcel-desktop:~\$'. The command 'hcitool dev' has been entered. The output shows 'Devices:' followed by a table with two columns: 'hci0' and '00:16:38:5E:51:81'.

```
marcel@marcel-desktop:~$ hcitool dev
Devices:
    hci0    00:16:38:5E:51:81
```

**Img 1:** hcitool output.

If you don't see any devices, you probably need to install the driver for your Bluetooth adapter. This might happen if your Bluetooth adapter is rare or too new.

Install python support for the Bluetooth now.

```
duckiebot:~$ sudo apt-get install python-pip python-dev ipython
```

```
duckiebot:~$ sudo apt-get install bluetooth libbluetooth-dev
```

```
duckiebot:~$ sudo pip install pybluez
```

OK, setup is done.

-----

## 2. Pairing Duckiebot with a Smartphone

For communications between two Bluetooth devices to work, they need to be “introduced” first via pairing mechanism.

First, set your Bluetooth name same as your duckiebots’ (you need to distinguish it somehow).

```
duckiebot:~$ sudo hciconfig hci0 name your_duckiebot_name
```

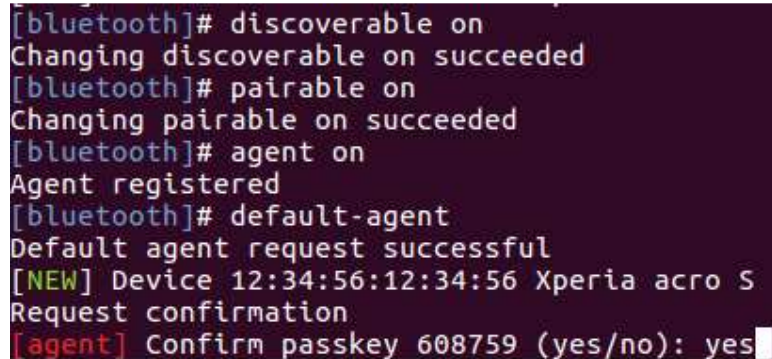
Prepare for pairing. On your duckiebot, launch this command:

```
duckiebot:~$ sudo bluetoothctl
```

You’ll see the list of devices already paired with your Bluetooth controller.  
Type in these commands (don’t type [bluetooth]#, it will be seen on screen):

```
[bluetooth]# discoverable on  
[bluetooth]# pairable on  
[bluetooth]# agent on  
[bluetooth]# default-agent
```

Say “yes” to all requests which will follow.



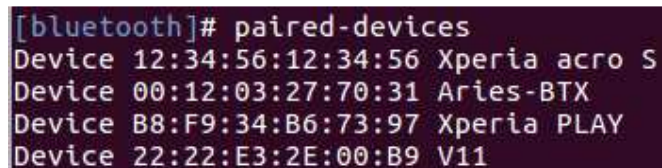
```
[bluetooth]# discoverable on  
Changing discoverable on succeeded  
[bluetooth]# pairable on  
Changing pairable on succeeded  
[bluetooth]# agent on  
Agent registered  
[bluetooth]# default-agent  
Default agent request successful  
[NEW] Device 12:34:56:12:34:56 Xperia acro S  
Request confirmation  
[agent] Confirm passkey 608759 (yes/no): yes
```

**Img 2:** Pairing your smartphone from console.

Now, check if your device is paired. Type

```
[bluetooth]# paired-devices
```

and see if your Smartphone is in the list. It should be there.



```
[bluetooth]# paired-devices  
Device 12:34:56:12:34:56 Xperia acro S  
Device 00:12:03:27:70:31 Aries-BTX  
Device B8:F9:34:B6:73:97 Xperia PLAY  
Device 22:22:E3:2E:00:B9 V11
```

**Img 3:** Check paired devices

All done, you're ready to establish first connection between Robot and Smartphone.

---

### **3. Creating Android application using MIT AppInventor**

Go to <http://ai2.appinventor.mit.edu>

Here you can create a simple application for your android without unspeakable horrors of Android Studio.

For this demo, you may use already provided example.

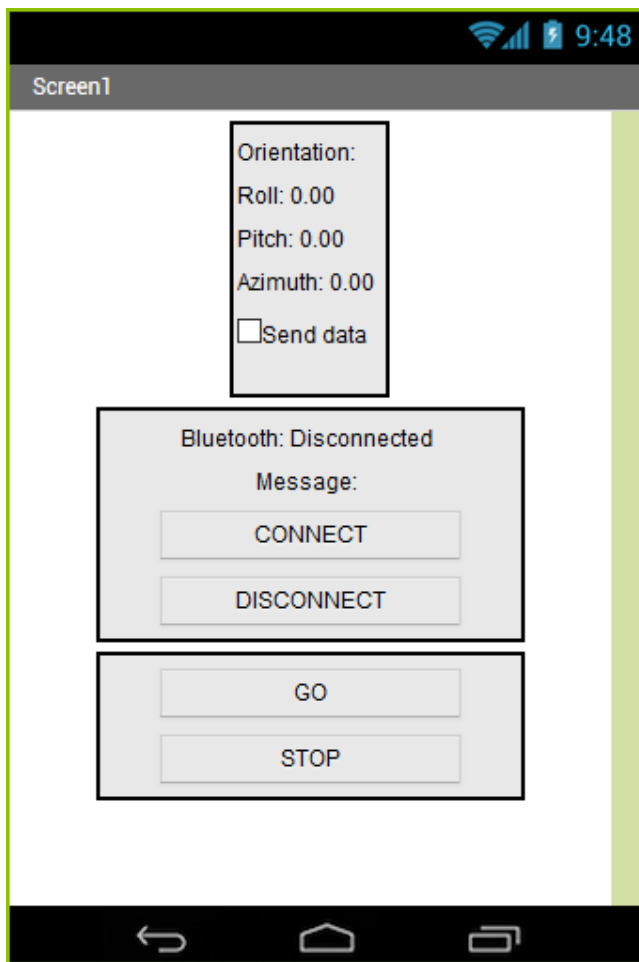
Download the project file from this link:

[https://github.com/YuryXW/csp\\_bluetooth\\_practice/blob/master/BluetoothIMU.aia?raw=true](https://github.com/YuryXW/csp_bluetooth_practice/blob/master/BluetoothIMU.aia?raw=true)

Import this project in AppInventor, then build and download it (use the QR code for this task, it is convenient).

You can edit this program however you want. All blocks have commentaries to tell you what they do.

---



Direct APK Link.

[https://github.com/YuryXW/csp\\_bluetooth\\_practice/blob/master/BluetoothIMU.apk?raw=true](https://github.com/YuryXW/csp_bluetooth_practice/blob/master/BluetoothIMU.apk?raw=true)

**Img 4:** Smartphone app to be used in this demo.

#### 4. Download the demo package for this tutorial

First, add your user (“ubuntu” by default) to the “dialout” group, so you’ll have an access to virtual serial ports created for Bluetooth.

```
duckiebot:~$ sudo useradd -G dialout ubuntu
```

Next you need to register the Serial Port service. Do this after each reset.

```
duckiebot:~$ sudo sdptool add -channel=22 SP
```

---

**[IMPORTANT]** This command should give you an output with successful result “Serial Port service registered”. If no output was given, everything is broken. Yes, again. Keep calm, deep breath, it’s fixable. To fix this problem do this:

Edit the “/etc/systemd/system/dbus-org.bluez.service” file.

```
duckiebot:~$ sudo nano /etc/systemd/system/dbus-org.bluez.service
```

Change this line:

ExecStart=/usr/lib/bluetooth/bluetoothd

To this:

ExecStart=/usr/lib/bluetooth/bluetoothd --compat

Run this command as well:

```
duckiebot:~$ sudo chmod 777 /var/run/sdp
```

Reboot your Duckiebot:

```
duckiebot:~$ sudo reboot now
```

After reboot, execute this command again:

```
duckiebot:~$ sudo sdptool add -channel=22 SP
```

-----

Change to your working folder, please. Now, git clone the package for this tutorial:

```
duckiebot:~$ cd your_duckietown_working_folder
duckiebot:~$ git clone
https://github.com/YuryXW/csp_bluetooth_practice.git
duckiebot:~$ cd ~/duckietown
duckiebot:~$ source environment.sh
duckiebot:~$ cd catkin_ws
duckiebot:~$ catkin_make
duckiebot:~$ cd ..
duckiebot:~$ source environment.sh
duckiebot:~$ source set_ros_master.sh
duckiebot:~$ source set_vehicle_name.sh your_robot_name
```

Launch the test node:

```
duckiebot:~$ roslaunch csp_bluetooth_demo csp_bluetooth_demo.launch
veh:=your_robot_name
```

Connect to your robot from the cellphone app you've downloaded.

Press "CONNECT" and select your robot's Bluetooth.

You can do two things now. First is to drive around your robot using the Smartphone. First, make sure you can control your duckiebot with a joystick (demo node will publish to the same topic as joystick does, “/your\_robot\_name/joy\_mapper\_node/car\_cmd”). Press “back” button on the joystick to be sure you’ve enabled its state in Duckiebot.

Press “GO” to make robot move forward until you press “STOP”.

Now, check “Send Data” checkbox, this will enable sending the orientation sensor (accelerometer) data to your Duckiebot. Demonstration node you’re using is designed to make your robot steer based on these readings, try to control your Duckiebot this way. Tilt your phone forward and backward to control velocity, and from side to side to control rotation. Tick “Send data” off when you’ve played enough.

Now, test how the phone will react to robot seeing the red line. Move to the red line SLOWLY, and when your Duckiebot will detect it, your phone will vibrate.

---

#### Additional tasks:

For these tasks, you may need to modify the “csp\_bluetooth\_demo\_node.py” file (located in “your\_working\_folder/csp\_bluetooth\_practice/csp\_bluetooth\_demo/src”)

1. Change roll and pitch, so that now Roll controls velocity and Pitch controls omega, so you can hold your phone in landscape mode.

HINT: Check “message\_handler” function.

2. Make your robot stop after it sees the red line and vibrates. There are two ways doing so, you either change your node code, OR you can change the Android code so that your Smartphone will send command for robot to stop after it receives the “VIBRATE” message.

---

## **APPENDIX**

### **A. Setting up RFCOMM channel**

**(this will work with almost any Smartphone including Apple iPhone)**

**(You can skip this part if you want)**

RFCOMM profile is an emulation of serial connection via Bluetooth. It is the most common way to use Bluetooth for communication between two devices. One way is to establish a permanent serial port which will be seen in your system. You'll be able to work with it like with any other real serial ports in the system.

First, add your user ("ubuntu" by default) to the "dialout" group, so you'll have an access to virtual serial ports created for Bluetooth.

```
duckiebot:~$ sudo useradd -G dialout ubuntu
```

Next you need to register the Serial Port service. Do this after each reset.

```
duckiebot:~$ sudo sdptool add -channel=22 SP
```



You can pick up any channel number you like between 1 and 30, for this tutorial 22 is recommended.

---

[**IMPORTANT**] This command should give you an output with successful result. If no output was given, do this:

Edit the “/etc/systemd/system/dbus-org.bluez.service” file.

```
duckiebot:~$ sudo nano /etc/systemd/system/dbus-org.bluez.service
```

Change this line:

```
ExecStart=/usr/lib/bluetooth/bluetoothd
```

To this:

```
ExecStart=/usr/lib/bluetooth/bluetoothd --compat
```

Run this command as well:

```
duckiebot:~$ sudo chmod 777 /var/run/sdp
```

Reboot your Duckiebot:

```
duckiebot:~$ sudo reboot now
```

---

Start Bluetooth server on selected channel.

```
duckiebot:~$ sudo rfcomm listen /dev/rfcomm0 22
```

Now, pick any “Bluetooth Terminal” application you like and try connect to your server form Smartphone. You can download “Bluetooth Terminal” from the Market, there are several apps with this name, pick any, they all do their job pretty well. Don’t stop your server, if you need to enter another Linux command use another terminal here.



<https://play.google.com/store/apps/details?id=ptah.apps.bluetoothterminal&hl=en>

Recommended “Bluetooth Terminal” app for Android.

Q: How to see data you've sent from Smartphone using only Linux commands?

In a new terminal:

```
duckiebot:~$ cat /dev/rfcomm0
```

Now, send anything using Bluetooth Terminal app and see the output in your robot's terminal.

Q: How to send data to smartphone using only Linux commands?

```
duckiebot:~$ echo "Hi there, dude!" > /dev/rfcomm0
```

See the output on your Bluetooth Terminal app.

---

## **B. Using `bluetooth_bridge` to communicate with Bluetooth in ROS.**

**(Skip this part if you're doing in-class demo).**

*This part will tell you how to use "bluetooth\_bridge" created for this demo separately, in case you want to use it in your final project.*

Another option is to use Python directly to communicate with Bluetooth. For this exercise, download and install "bluetooth\_bridge" ROS node. You can use this code as an example of Bluetooth communication in Python. This node will set up Bluetooth server on your robot, wait for incoming connections, and then send messages between ROS and Bluetooth device.

```
duckiebot:~$ cd your_duckietown_working_folder
duckiebot:~$ git clone https://github.com/YuryXW/bluetooth_bridge.git
duckiebot:~$ cd ~/duckietown/catkin_ws
duckiebot:~$ catkin_make
```

This node has two important topics:

"/bluetooth/send", type "std\_msgs.String" - send a "String" message to this topic to send it via bluetooth.

“bluetooth/received”, type “std\_msgs.String” - any message received via Bluetooth will be published through this topic.

[IMPORTANT]: do not forget to set up the Serial Port service on your Duckiebot if you’ve rebooted your robot or did not set it before (sudo rfcomm listen /dev/rfcomm0 22).

Launch [bluetooth\\_bridge](#):

```
duckiebot:~$ roslaunch bluetooth_bridge bluetooth_bridge.launch  
rfcomm_channel:=22
```

Now check if messages are being published in /bluetooth/received topic:

```
duckiebot:~$ rostopic echo /bluetooth/received
```

Send something via Bluetooth Terminal from your smartphone.

Next, send a string to /bluetooth/send topic and see it appear in your Bluetooth Terminal:

```
duckiebot:~$ rostopic pub /bluetooth/send std_msgs/String "Hello" --once
```

This is it, you’ve set up the bridge between ROS and your Bluetooth device (Smartphone, in this case).

---