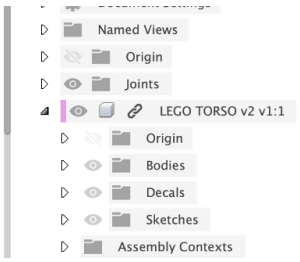
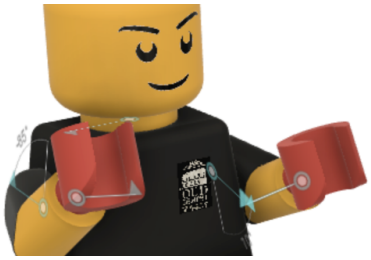




Koen Verbeeck

Wireless Networks: WPAN

Description Assignments

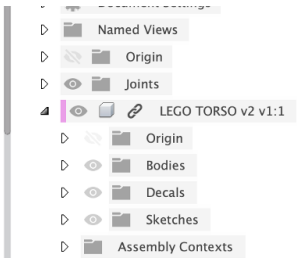
1) Find a minimum of 5 bluetooth devices and note their bluetooth address. Check if some of these devices use the same BT chip. (Tip: your raspberry pi and arduino nano 33 IoT also have BT)
2) Use the Raspberry Pi to transmit and/or read data via a bluetooth connection
    a) You choose how complicated you make the project, be creative!
    b) Document your project.
    c) Tips:
        i. For example, you can control a relay with your cell phone that is bluetooth connected to the raspberry pi
        ii. You can read out a sensor with your arduino which then sends this data to the pi and stores it in a file.
        iii. ...

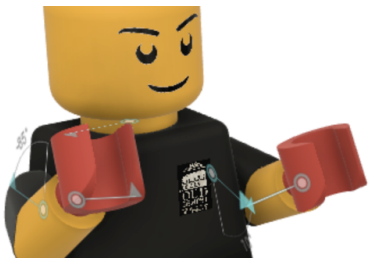
Github Page with Project Details

https://github.com/kverbeeck/BlueDot_Control_RGB_LED

YouTube Video Project Demo

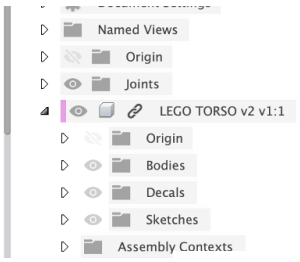
https://www.youtube.com/watch?v=d3yTP8SgePo

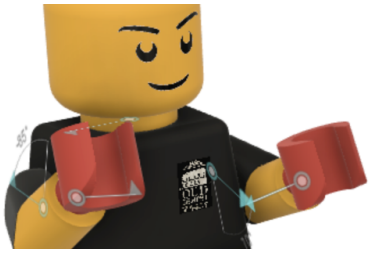




Koen Verbeeck

Table of Contents

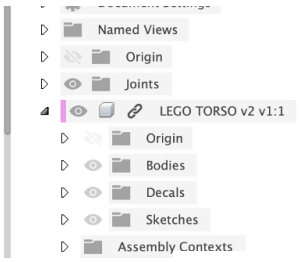


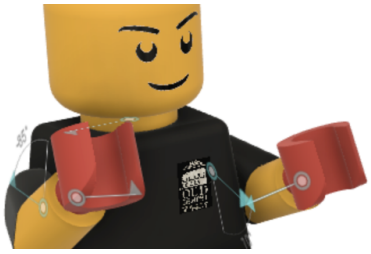


Koen Verbeeck

Assignment 1: Bluetooth Devices

	Vendor	Type	MAC	Chipset Vendor	Chipset
1	Apple	MacBook Pro 2020	3C-22-FB-C0-A1-F9	Broadcom	4364B3 (339S00610 (waarschijnlijk een Apple Wi-Fi/Bluetooth-module)
2	TECKNET	BM30X mouse	DC-2C-26-B3-1E-5B	Broadcom	BCM2042
3	Apple	IPAD 4th Gen	5C-96-9D-BE-32-B2	Broadcom	BCM4334
4	Apple	Iphone 6s	AC-61-EA-F0-B0-A5	Universal Scientific Industrial	339S00043 WiFi/Bluetooth-module
5	Apple	Iphone 8	40-CB-C0-25-4F-72	Apple	USI 170804 339S00397 WiFi/Bluetooth-module
6	Apple	iWatch	E4-E0-A6-76-3C-EC	Apple	W2
7	JBL	Charge 3 Speaker	00-42-79-E3-9A-6F	Qualcomm	CSR8675
8	Moosen	TWS-880Pro Earbuds	00-00-AB-CD-69-4F	Qualcomm	QCC3020
9	Sony	Xperia Tablet Z3	30:75:12:CD:0C:2A	Broadcom	BCM4339
	Samsung	Galaxy S4	4C:A56D:3B:65:6D	Broadcom	BCM4335





Koen Verbeeck

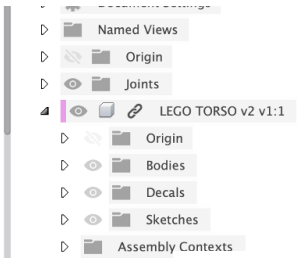
Assignment 2: Raspberry Pi Stable Bluetooth Connection

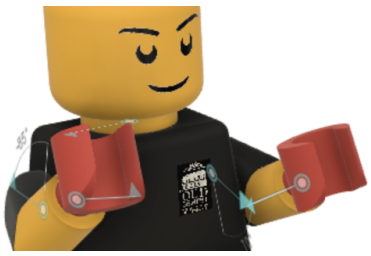
Radio frequency communication (RFCOMM)

- The Bluetooth protocol RFCOMM is a simple set of transport protocols, made on top of the L2CAP protocol, providing emulated RS-232 serial ports (up to sixty simultaneous connections to a Bluetooth device at a time).
- The protocol is based on the ETSI standard TS 07.10.
- RFCOMM is sometimes called *serial port emulation*. The Bluetooth *serial port profile* is based on this protocol.
- RFCOMM provides a simple reliable data stream to the user, similar to TCP. It is used directly by many telephony related profiles as a carrier for AT commands, as well as being a transport layer for OBEX over Bluetooth.
- Many Bluetooth applications use RFCOMM because of its widespread support and publicly available API on most operating systems. Additionally, applications that used a serial port to communicate can be quickly ported to use RFCOMM.
- In the protocol stack, RFCOMM is bound to L2CAP.
- **Programming for Bluetooth in Python** follows the socket programming model and communications between the Bluetooth devices is done through RFCOMM socket.
- RFCOMM is very popular in Bluetooth applications because of its wide support and publically available API.

How to Check the Software and Hardware Version of a Raspberry Pi

```
pi@KoenRPI4B:~ $ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 10 (buster)"
NAME="Raspbian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL=http://www.raspbian.org/RaspbianBugs
```





Koen Verbeeck

We first need to install few Bluetooth related packages

- **BlueZ** is a open source project and official Linux Bluetooth protocol stack. It supports all the core Bluetooth protocols and now become part of official Linux Kernel.
- **Blueman** provides the Desktop interface to manage and control the Bluetooth devices.

```
pi@KoenRPI4B:~ $ sudo apt-get install bluetooth blueman bluez
Reading package lists... Done
Building dependency tree
Reading state information... Done
bluez is already the newest version (5.50-1.2~deb10u1+rpt2).
bluez set to manually installed.
The following additional packages will be installed:
  bluez-obexd gir1.2-appindicator3-0.1 libappindicator3-1 libbluetooth3 libdbusmenu-
glib4 libdbusmenu-gtk3-4 libical3
  notification-daemon python3-gi-cairo
Suggested packages:
  bluez-cups
The following NEW packages will be installed:
  blueman bluetooth bluez-obexd gir1.2-appindicator3-0.1 libappindicator3-1
libbluetooth3 libdbusmenu-glib4 libdbusmenu-gtk3-4
  libical3 notification-daemon python3-gi-cairo
0 upgraded, 11 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,523 kB of archives.
After this operation, 7,722 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

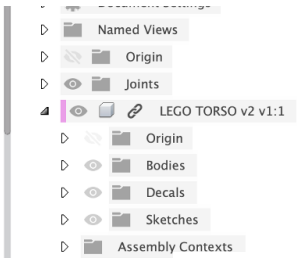
Install the GPIO support libraries for Raspberry Pi

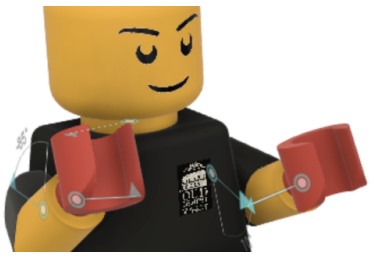
```
pi@KoenRPI4B:~ $ sudo apt-get install python-rpi.gpio
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-rpi.gpio is already the newest version (0.7.0-0.1~bpo10+4).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@KoenRPI4B:~ $
```

Open the bluetoothctl utility by using below command

```
pi@KoenRPI4B:~ $ sudo bluetoothctl
Agent registered
[bluetooth]#
```

Display all the available commands



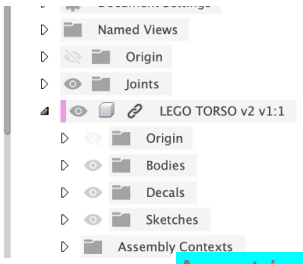


Koen Verbeeck

```
[bluetooth]# help
Menu main:
Available commands:
-------------------
advertise                 Advertise Options Submenu
scan                      Scan Options Submenu
gatt                      Generic Attribute Submenu
list                      List available controllers
show [ctrl]               Controller information
select <ctrl>             Select default controller
devices                   List available devices
paired-devices            List paired devices
system-alias <name>       Set controller alias
reset-alias               Reset controller alias
power <on/off>            Set controller power
pairable <on/off>         Set controller pairable mode
discoverable <on/off>     Set controller discoverable mode
agent <on/off/capability> Enable/disable agent with given capability
default-agent             Set agent as the default one
advertise <on/off/type>   Enable/disable advertising with given type
set-alias <alias>         Set device alias
scan <on/off>             Scan for devices
info [dev]                Device information
pair [dev]                Pair with device
trust [dev]               Trust device
untrust [dev]             Untrust device
block [dev]               Block device
unblock [dev]             Unblock device
remove <dev>              Remove device
connect <dev>             Connect device
disconnect [dev]          Disconnect device
menu <name>               Select submenu
version                   Display version
quit                      Quit program
exit                      Quit program
help                      Display help about this program
export                    Print evironment variables
[bluetooth]#
```

Input following commands to scan for our ANDROID device

```
[bluetooth]# power on
Changing power on succeeded
[bluetooth]# agent on
```

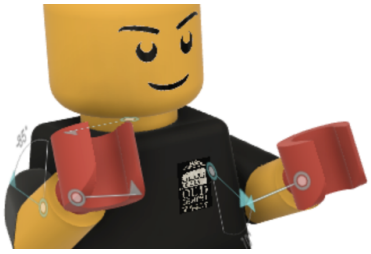


Koen Verbeeck

Agent is already registered
[bluetooth]# discoverable on
Changing discoverable on succeeded
[CHG] Controller DC:A6:32:69:1A:96 Discoverable: yes
[bluetooth]# pairable on
Changing pairable on succeeded
[bluetooth]# scan on
Discovery started
[CHG] Controller DC:A6:32:69:1A:96 Discovering: yes
[NEW] Device 69:01:C5:37:6A:F0 69-01-C5-37-6A-F0
[NEW] Device 6E:E7:5D:B6:F9:2E 6E-E7-5D-B6-F9-2E
[NEW] Device 76:34:79:AA:68:59 76-34-79-AA-68-59
[NEW] Device 30:75:12:CD:0C:2A RSSI: -60
[CHG] Device 30:75:12:CD:0C:2A Xperia Z3 Tablet Compact

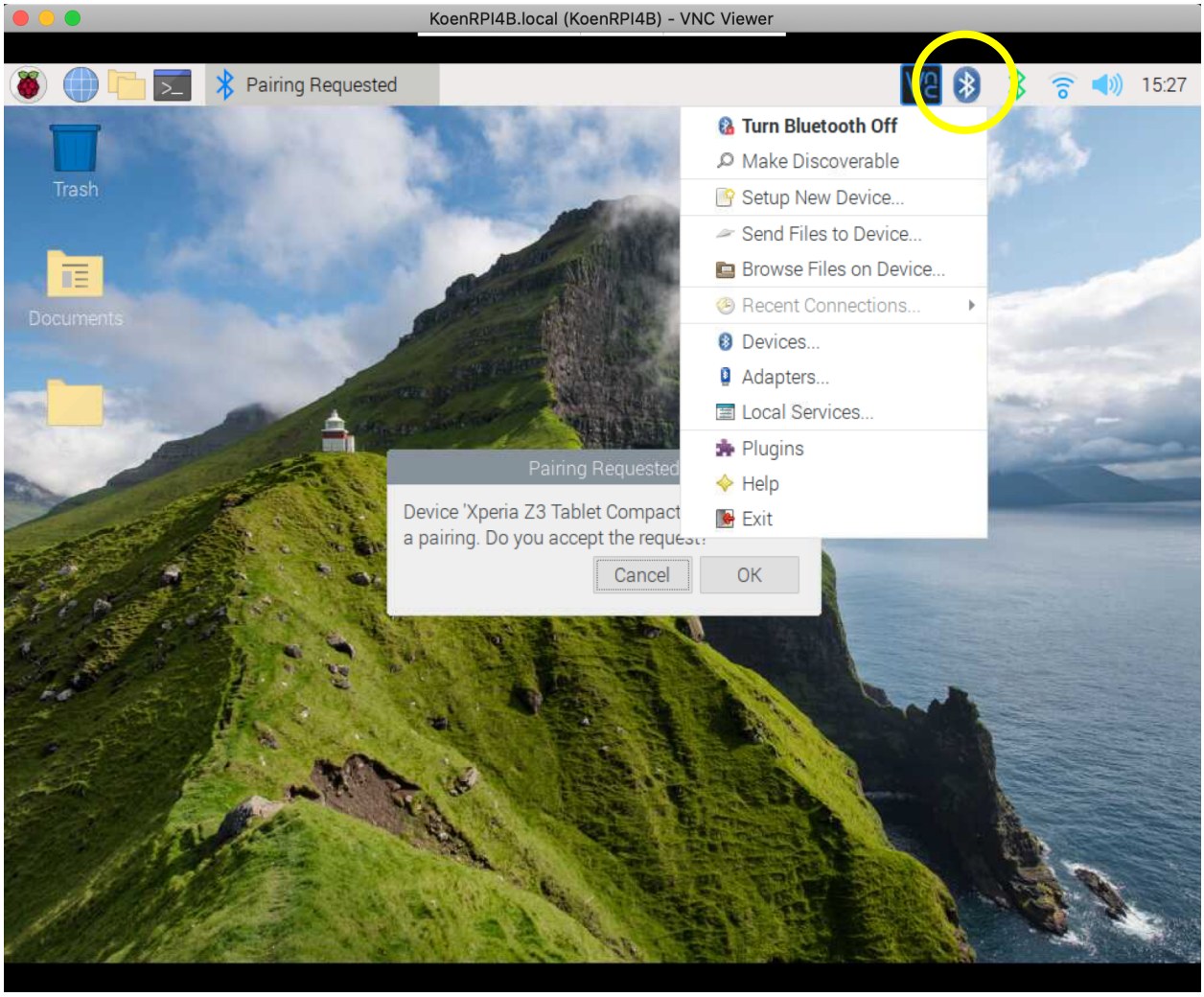
Input following command pair with our ANDROID device
[bluetooth]# pair 30:75:12:CD:0C:2A
Attempting to pair with 30:75:12:CD:0C:2A
[CHG] Device 30:75:12:CD:0C:2A Connected: yes
Request confirmation
[agent] Confirm passkey 859362 (yes/no): yes
[CHG] Device 30:75:12:CD:0C:2A Modalias: usb:v0FCEp01C0d0010
[CHG] Device 30:75:12:CD:0C:2A UUIDs: 00001105-0000-1000-8000-00805f9b34fb
[CHG] Device 30:75:12:CD:0C:2A UUIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device 30:75:12:CD:0C:2A UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device 30:75:12:CD:0C:2A UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Device 30:75:12:CD:0C:2A UUIDs: 00001112-0000-1000-8000-00805f9b34fb
[CHG] Device 30:75:12:CD:0C:2A UUIDs: 00001132-0000-1000-8000-00805f9b34fb
[CHG] Device 30:75:12:CD:0C:2A UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device 30:75:12:CD:0C:2A UUIDs: 00001800-0000-1000-8000-00805f9b34fb
[CHG] Device 30:75:12:CD:0C:2A UUIDs: 00001801-0000-1000-8000-00805f9b34fb
[CHG] Device 30:75:12:CD:0C:2A UUIDs: fa87c0d0-afac-11de-8a39-0800200c9a66
[CHG] Device 30:75:12:CD:0C:2A UUIDs: fa87c0d0-afac-11de-8a39-0800200c9a67
[CHG] Device 30:75:12:CD:0C:2A ServicesResolved: yes
[CHG] Device 30:75:12:CD:0C:2A Paired: yes
Pairing successful

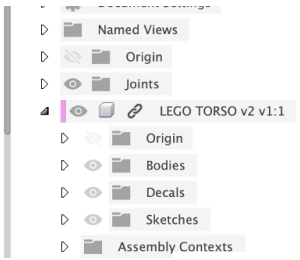
We now have a stable connection to our Android Client, you will need to re-pair your Android client in the Blue Dot app or any Bluetooth Serial Application to the Raspberry Pion reboot of any of the 2 devices. No worries, this will go really fast and the connection will remain stable during your session.

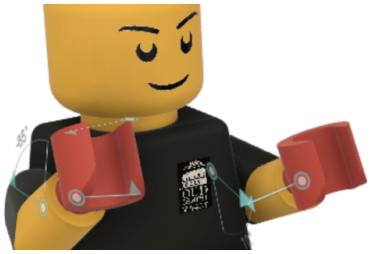


Blueman GUI interface

This is how the Blueman tool looks in the GUI, note the Blue/white Bluetooth Logo. You can also use this to connect to your Android client.







Koen Verbeeck

Assignment 3: Raspberry Pi Blue Dot Project (RGB LED)

The Project

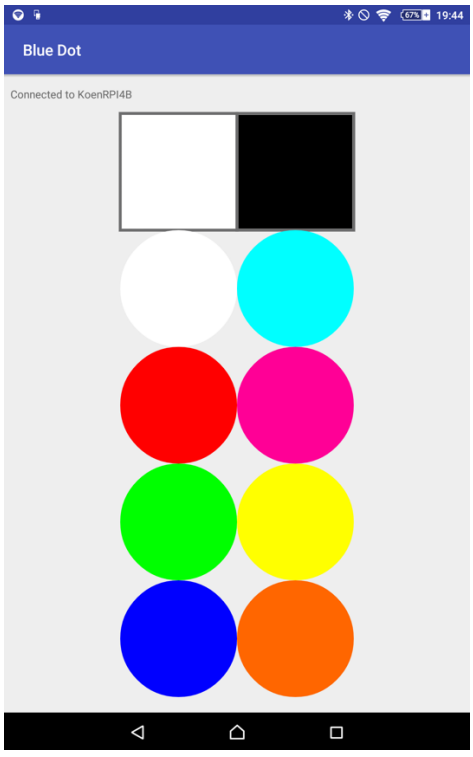
This project will control an RGB LED connected to the GPIO pins of a Raspberry Pi. We will use Blue Dot running on the ANDROID client to control the RGB LED pre-configured colors and their brightness. Below is how it will look on both sides.

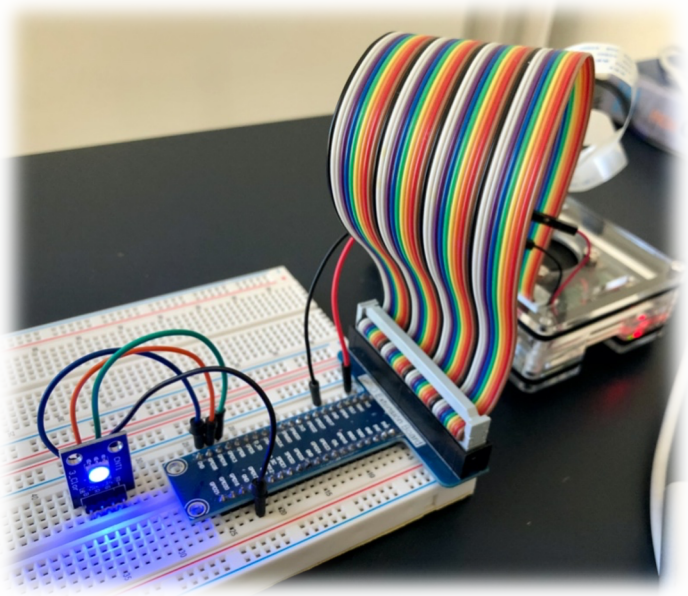
Having already installed **BlueZ** in the previous assignment, we can use this tool to secure a stable connection with our Android device and Blue Dot application.
https://bluedot.readthedocs.io/en/latest/gettingstarted.html
https://bluedot.readthedocs.io/en/latest/index.html

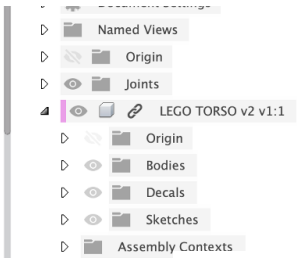
In order to use Blue Dot you will need:

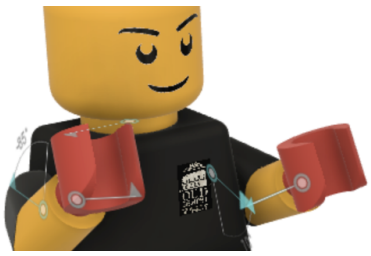
- A Raspberry Pi
  - with built-in Bluetooth (such as the Raspberry Pi 3, 4 or Zero W)
  - or a USB Bluetooth dongle
- An Android device or 2nd Raspberry Pi for the remote
- An Internet connection (for the install)

For this assignment, we will an Android device as the client to control



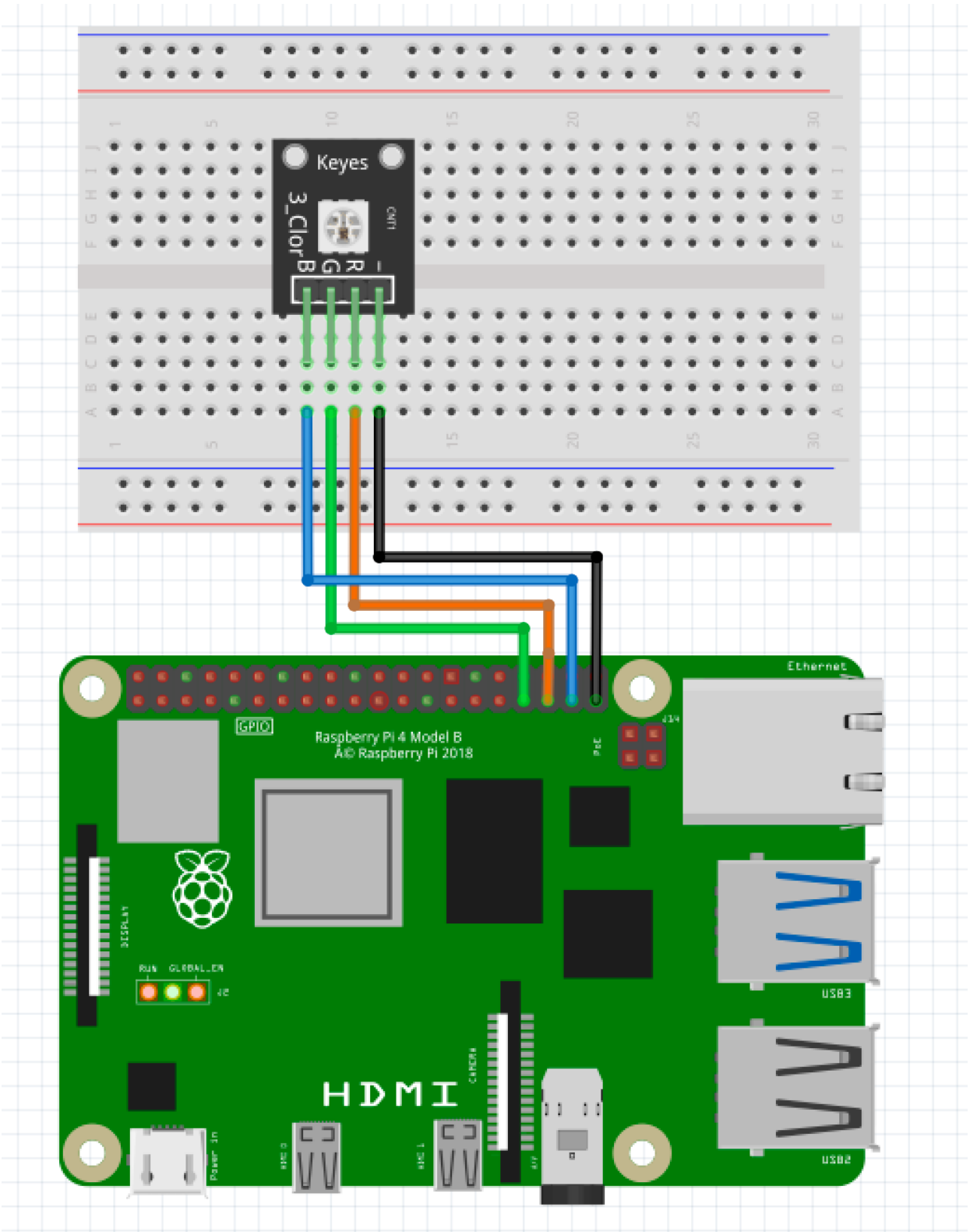


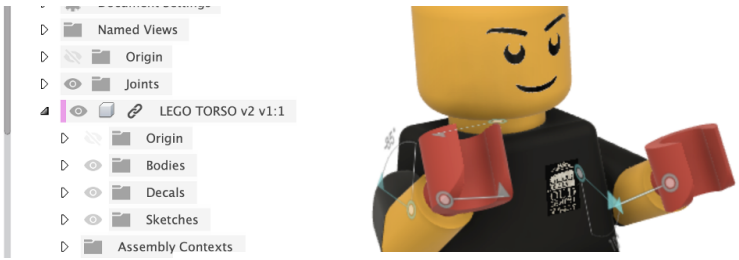




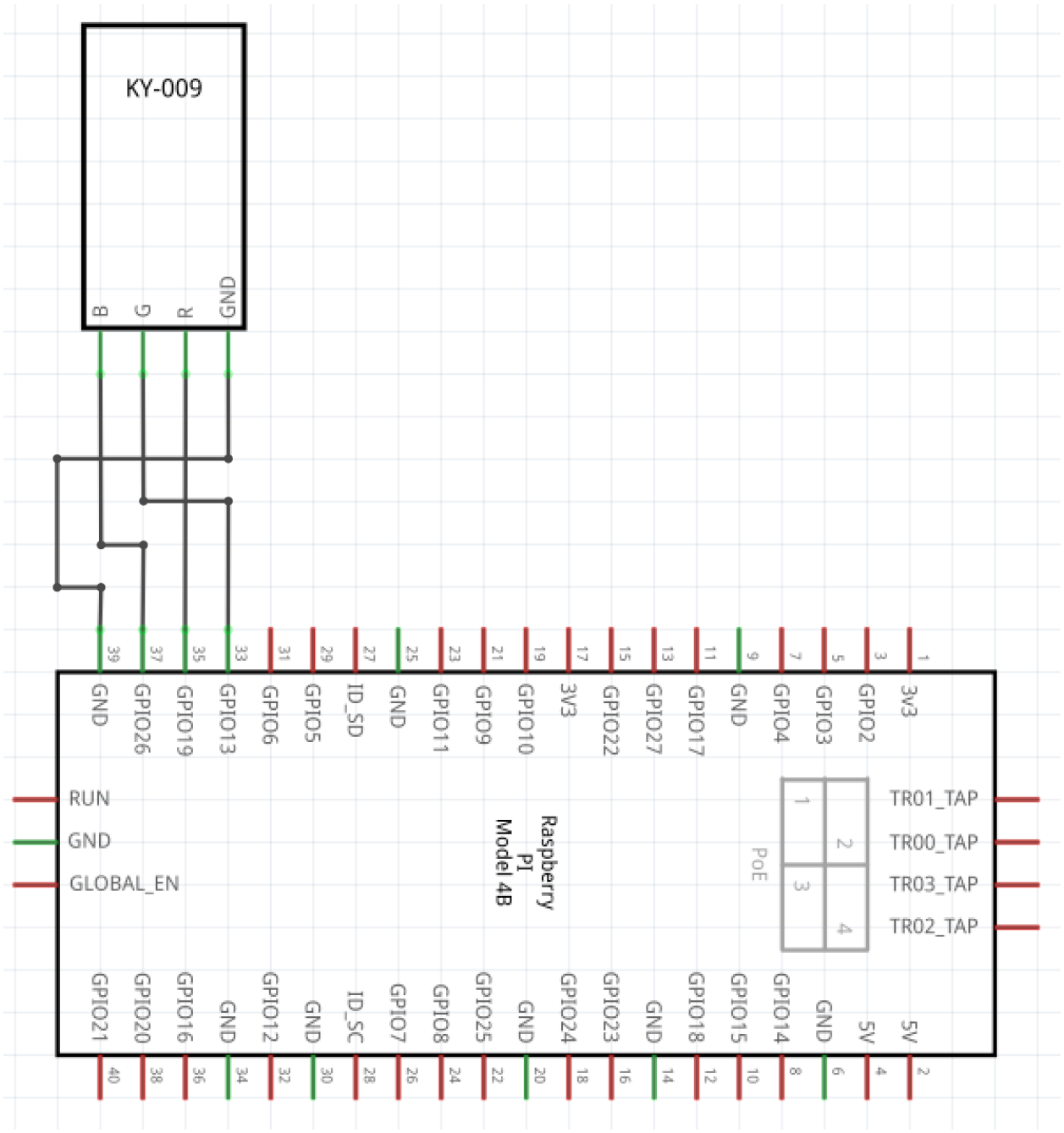
Koen Verbeeck

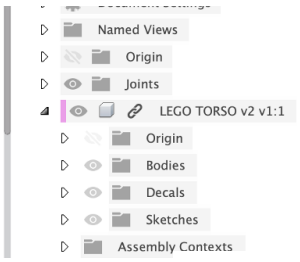
Schematics

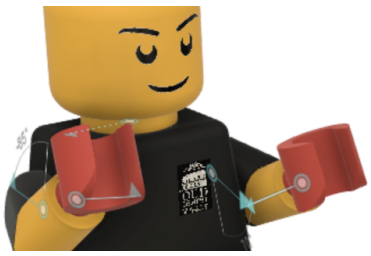




Koen Verbeeck







Koen Verbeeck

Installing Blue Dot on Raspberry Pi

Update the internet sources for your installed packages updates

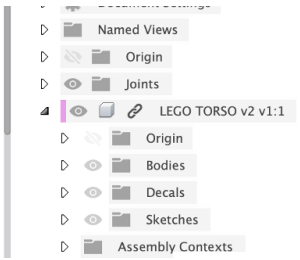
```
pi@KoenRPI4B:~ $ sudo apt-get update
Get:1 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:2 http://archive.raspberrypi.org/debian buster InRelease [32.9 kB]
Get:3 http://raspbian.raspberrypi.org/raspbian buster/main armhf Packages [13.0 MB]
Get:4 http://archive.raspberrypi.org/debian buster/main armhf Packages [372 kB]
Fetched 13.4 MB in 7s (1,960 kB/s)
Reading package lists... Done
pi@KoenRPI4B:~ $
```

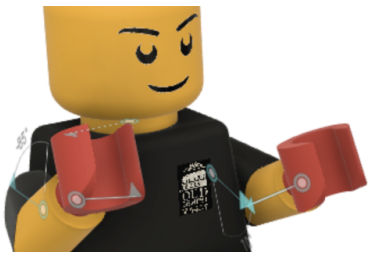
Installs the Bluedot python library

```
pi@KoenRPI4B:~ $ sudo pip3 install bluedot
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting bluedot
  Downloading https://www.piwheels.org/simple/bluedot/bluedot-2.0.0-py3-none-
any.whl (37 kB)
Installing collected packages: bluedot
Successfully installed bluedot-2.0.0
```

Look for and install Bluedot updatesif available

```
pi@KoenRPI4B:~ $ sudo pip3 install bluedot --upgrade
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: bluedot in /usr/local/lib/python3.7/dist-packages (2.0.0)
pi@KoenRPI4B:~ $
```



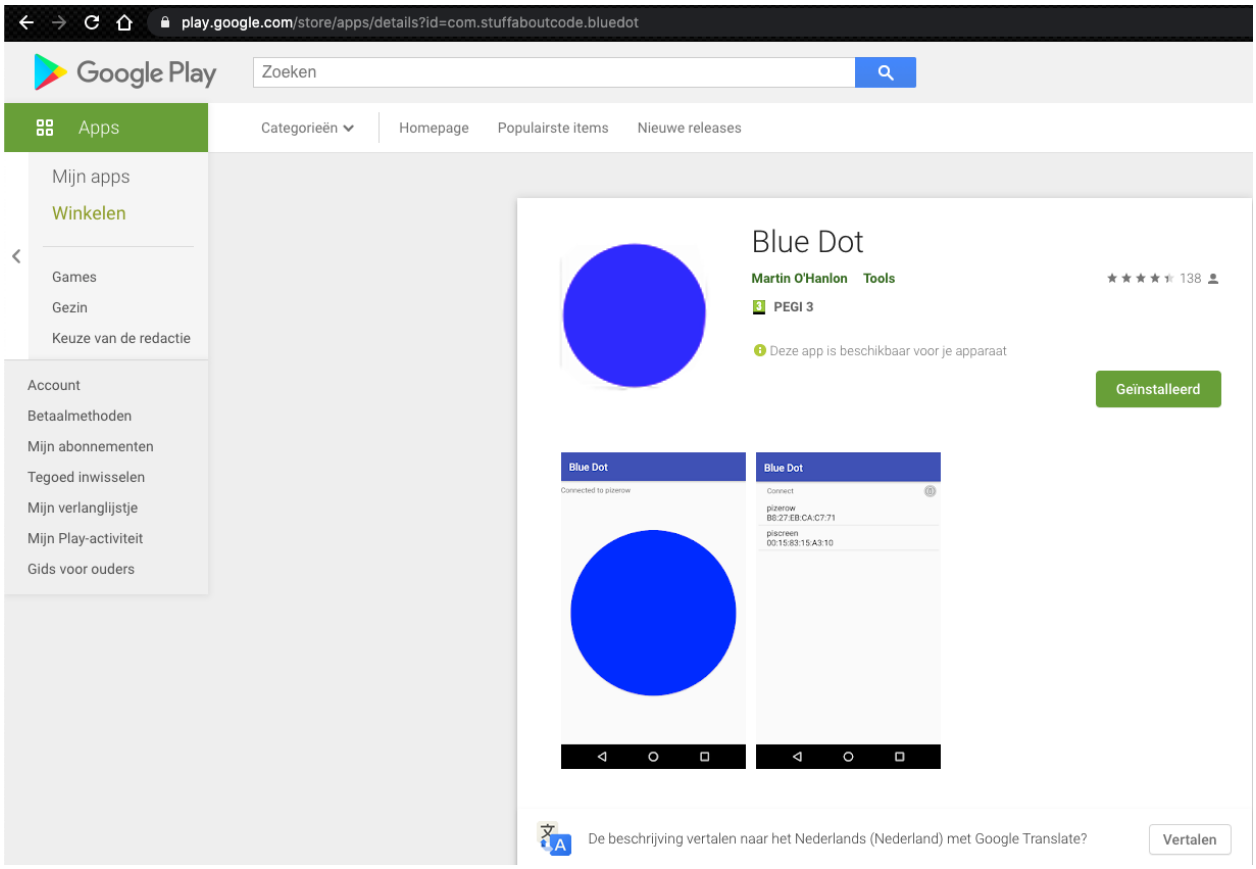


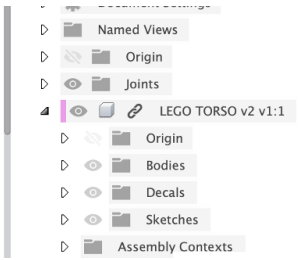
Koen Verbeeck

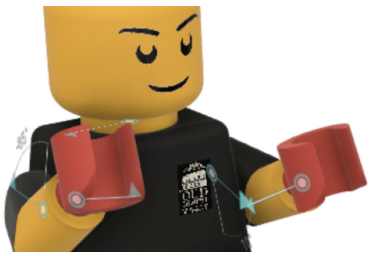
Install Blue Dot on your ANDROID Client

Download the application

The Blue Dot app is available to download from the Google Play store.







Koen Verbeeck

Run the Python code on your Raspberry Pi.

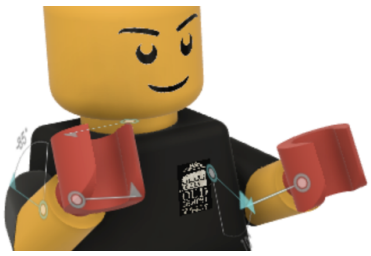
Please find below the code that I wrote for this project. You can find details explaining the code in the comments. Everything you see in the ANDROID app, is coded in Python on the Raspberry Pi.

“

```
from bluedot import BlueDot
# The signal.signal() function allows defining custom handlers to be executed when a signal is received.
from signal import pause
# import PWMLED module from gpiozero library to brightness/color control RGB LED
from gpiozero import PWMLED

# RGB LED
greenLed = PWMLED(13)  # initialize GPIO13 as a LED pin
redLed = PWMLED(19)  # initialize GPIO19 as a LED pin
blueLed = PWMLED(26)  # initialize GPIO26 as a LED pin

# Create 10 virtual bluedot buttons side by side, by setting the number of cols to 2 and rows to 5
bd = BlueDot(cols=2, rows=5)
bd[0, 0].color = (255, 255, 255)  # On (R, G, B) >> White
bd[0, 0].square = True  # On (square button)
bd[0, 0].border = True  # On (border around button)
bd[1, 0].color = (0, 0, 0)  # Off (R, G, B) >> Black
bd[1, 0].square = True  # Off (square button)
bd[1, 0].border = True  # Off (border around button)
bd[0, 1].color = (255, 255, 255)  # White (R, G, B) >> White
bd[0, 2].color = (255, 0, 0)  # Red (R, G, B) >> Red
bd[0, 3].color = (0, 255, 0)  # Green (R, G, B) >> Green
bd[0, 4].color = (0, 0, 255)  # Blue (R, G, B) >> Blue
bd[1, 1].color = (0, 255, 255)  # Cyan (R, G, B) >> Cyan
bd[1, 2].color = (255, 0, 150)  # Magenta (R, G, B) >> Magenta
bd[1, 3].color = (255, 255, 0)  # Yellow (R, G, B) >> Yellow
bd[1, 4].color = (255, 102, 0)  # Yellow (R, G, B) >> Orange
#bd[1, 4].visible = False  # Will hide the last button, only 9 are needed.


# Callback functions if a button is pressed
def on_button(pos):
    print("On button is pressed")
    redLed.value = 1  # full brightness
    greenLed.value = 1  # full brightness
```

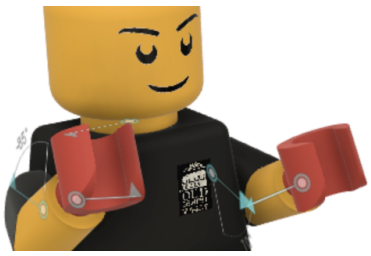
Koen Verbeeck

```
    blueLed.value = 1  # full brightness

def off_button(pos):
    print("Off button is pressed")
    redLed.value = 0  # off
    greenLed.value = 0  # off
    blueLed.value = 0  # off

# Callback functions if a slider is used
# The BlueDotPosition.x property returns a value from -1 (far down) to 1 (far up).
# Using this value you can create a slider (Dimmer) which goes vertically through the
middle:
def set_bright_white(pos):
    print("White brightness slider is being sweeped")
    brightness = (pos.y + 1) / 2
    redLed.value = brightness
    greenLed.value = brightness
    blueLed.value = brightness

def set_bright_red(pos):
    print("Red brightness slider is being sweeped")
    brightness = (pos.y + 1) / 2
    redLed.value = brightness
    greenLed.value = 0
    blueLed.value = 0

def set_bright_green(pos):
    print("Green brightness slider is being sweeped")
    brightness = (pos.y + 1) / 2
    redLed.value = 0
    greenLed.value = brightness
    blueLed.value = 0

def set_bright_blue(pos):
    print("Blue brightness slider is being sweeped")
    brightness = (pos.y + 1) / 2
    redLed.value = 0
    greenLed.value = 0
    blueLed.value = brightness

def set_bright_cyan(pos):
    print("Cyan brightness slider is being sweeped")
    brightness = (pos.y + 1) / 2
    redLed.value = 0
```



Koen Verbeeck

```
    greenLed.value = brightness
    blueLed.value = brightness

def set_bright_magenta(pos):
    print("Magenta brightness slider is being sweeped")
    brightness = (pos.y + 1) / 2
    redLed.value = brightness
    greenLed.value = 0
    blueLed.value = brightness * (1 / 255) * 150

def set_bright_yellow(pos):
    print("Yellow brightness slider is being sweeped")
    brightness = (pos.y + 1) / 2
    redLed.value = brightness
    greenLed.value = brightness
    blueLed.value = 0

def set_bright_orange(pos):
    print("Yellow brightness slider is being sweeped")
    brightness = (pos.y + 1) / 2
    redLed.value = brightness
    greenLed.value = brightness * (1 / 255) * 102
    blueLed.value = 0

while True:
    # The Infinite while loop will monitor for Blue Dot button presses and button sweeps
    # and will run the needed callback function.
    # bd.when_pressed = button_pressed
    bd[0, 0].when_pressed = on_button
    bd[1, 0].when_pressed = off_button
    bd[0, 1].when_moved = set_bright_white
    bd[0, 2].when_moved = set_bright_red
    bd[0, 3].when_moved = set_bright_green
    bd[0, 4].when_moved = set_bright_blue
    bd[1, 1].when_moved = set_bright_cyan
    bd[1, 2].when_moved = set_bright_magenta
    bd[1, 3].when_moved = set_bright_yellow
    bd[1, 4].when_moved = set_bright_orange

    # Cause the process to sleep until a signal is received;
    # the appropriate handler will then be called. Returns nothing.
    pause()
"
```



Ssh to you raspberry Pi using a terminal window. Copy the code to your Pi and the run it, below is how it should look while its waiting for the ANDROID client to pair. MAC address of your Raspberry Pi is shown here.

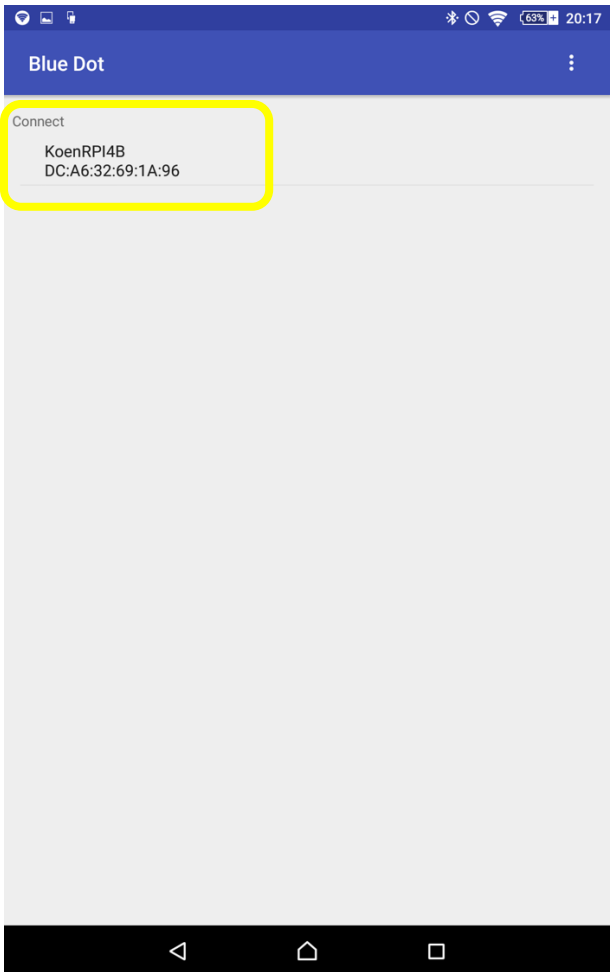
```
pi@KoenRPI4B:~/Desktop/exercise $ sudo python3 RGB_Control_BlueDot_Dimmer.py
Server started DC:A6:32:69:1A:96
Waiting for connection
```

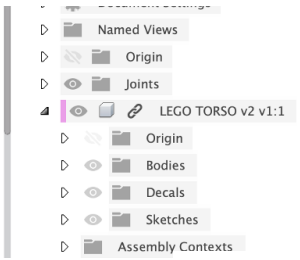
Once Connected. MAC address of your ANDROID device is shown here.

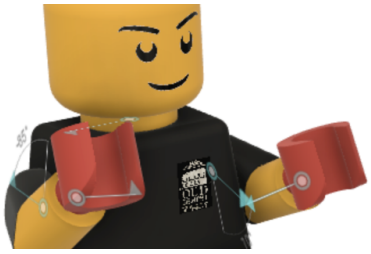
```
Client connected 30:75:12:CD:0C:2A
```

Connect your ANDROID device

You should now see the MAC address of your Raspberry, tap on it to connect.







Koen Verbeeck

Playing with the App

The 2 square buttons are on/off switches, the dots are sliders with pre-programmed colors. Change the brightness by slide vertically over the dots. These colors can be changed altering the code and an infinite number of colors (dots) can be added. See the YouTube Video on the first page of this document for a demo.