

# MIT AI2 204

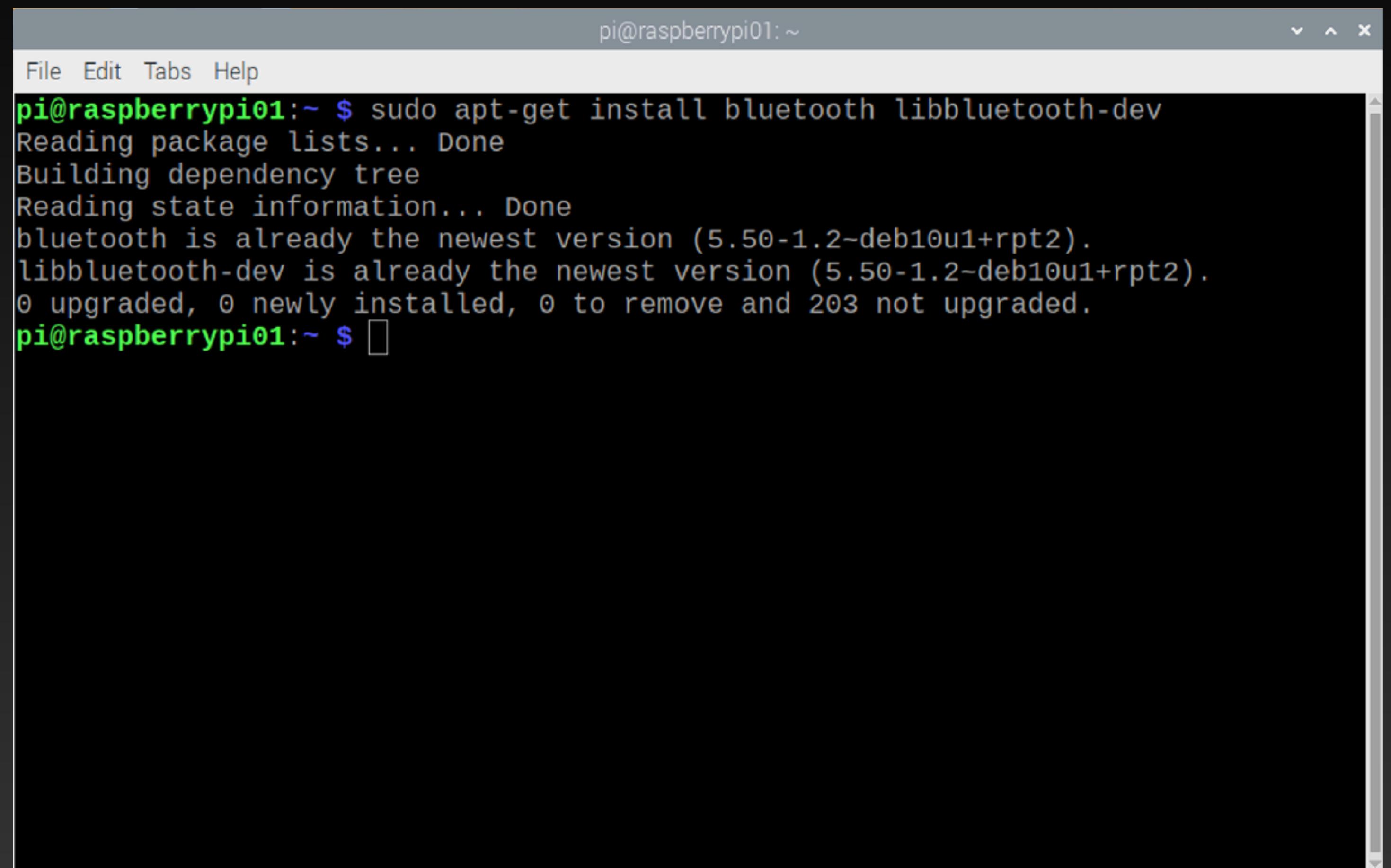
# IoT with MIT App Inventor

Fundamental

Xincheng Tang

# Python Program - Bluetooth setup

```
$ sudo apt-get install  
bluetooth libbluetooth-dev
```

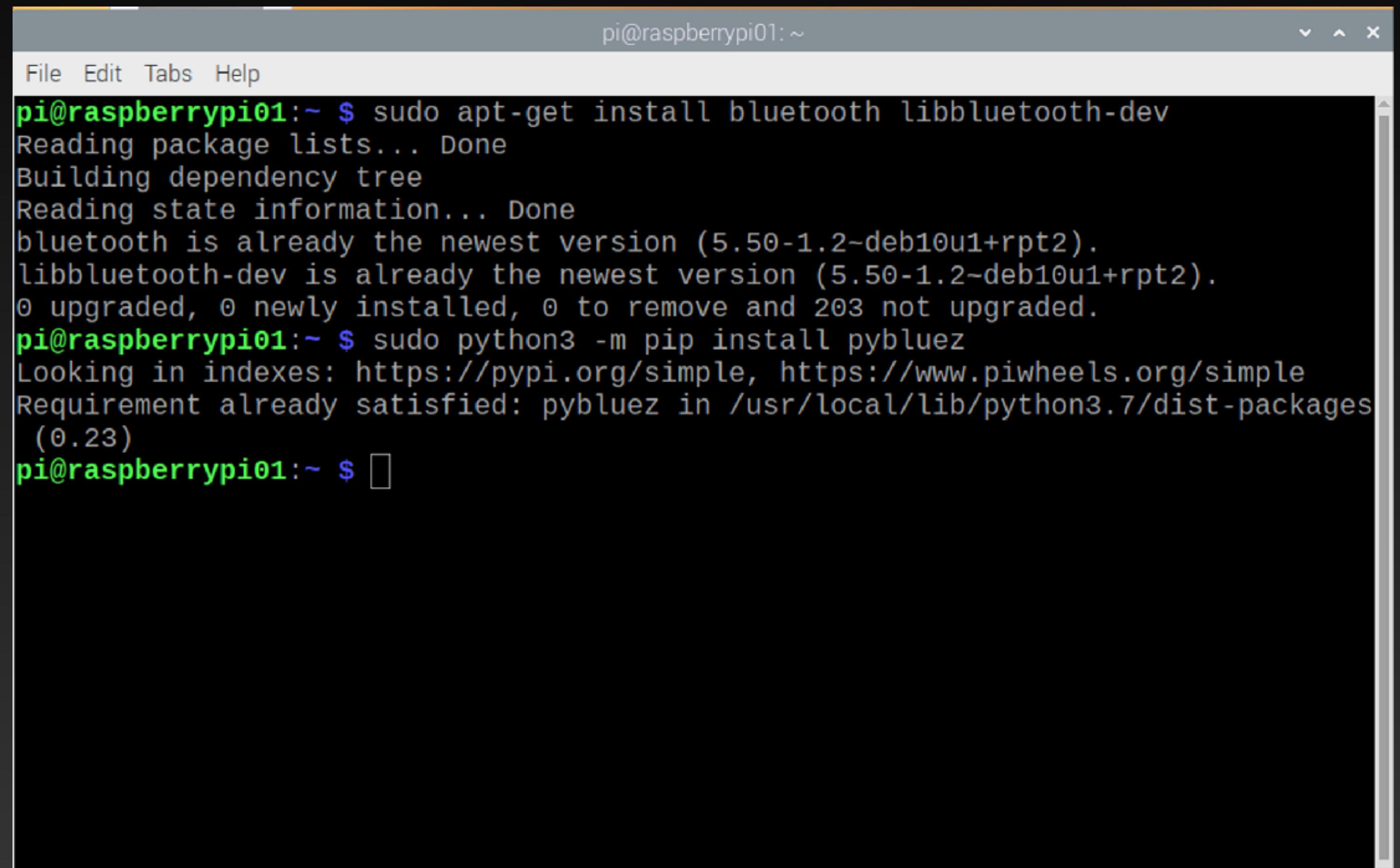


A screenshot of a terminal window titled "pi@raspberrypi01: ~". The window has a standard Linux-style title bar with icons for minimize, maximize, and close. The menu bar includes "File", "Edit", "Tabs", and "Help". The terminal content shows the command \$ sudo apt-get install bluetooth libbluetooth-dev being run. The output indicates that bluetooth and libbluetooth-dev are already at their newest versions (5.50-1.2~deb10u1+rpt2). It also shows 0 upgraded, 0 newly installed, and 203 not upgraded packages.

```
pi@raspberrypi01:~ $ sudo apt-get install bluetooth libbluetooth-dev  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
bluetooth is already the newest version (5.50-1.2~deb10u1+rpt2).  
libbluetooth-dev is already the newest version (5.50-1.2~deb10u1+rpt2).  
0 upgraded, 0 newly installed, 0 to remove and 203 not upgraded.  
pi@raspberrypi01:~ $
```

# Python Program - Bluetooth setup

```
$ sudo python3 -m pip  
install pybluez
```



The screenshot shows a terminal window titled "pi@raspberrypi01:~". The window contains the following command and its output:

```
pi@raspberrypi01:~ $ sudo apt-get install bluetooth libbluetooth-dev  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
bluetooth is already the newest version (5.50-1.2-deb10u1+rpt2).  
libbluetooth-dev is already the newest version (5.50-1.2-deb10u1+rpt2).  
0 upgraded, 0 newly installed, 0 to remove and 203 not upgraded.  
pi@raspberrypi01:~ $ sudo python3 -m pip install pybluez  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Requirement already satisfied: pybluez in /usr/local/lib/python3.7/dist-packages  
(0.23)  
pi@raspberrypi01:~ $
```

# Python Program - Bluetooth setup

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

The screenshot shows a terminal window titled "pi@raspberrypi01: ~". The window title bar also displays "File Edit Tabs Help" and "GNU nano 3.2". The main content of the terminal is the configuration file for the Bluetooth service:

```
[Unit]  
Description=Bluetooth service  
Documentation=man:bluetoothd(8)  
ConditionPathIsDirectory=/sys/class/bluetooth  
  
[Service]  
Type=dbus  
BusName=org.bluez  
ExecStart=/usr/lib/bluetooth/bluetoothd -c  
ExecStartPost=/usr/bin/sdptool add SP  
NotifyAccess=main  
#WatchdogSec=10  
#Restart=on-failure  
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE  
LimitNPROC=1  
ProtectHome=true  
ProtectSystem=full  
  
[Install]
```

At the bottom of the terminal, there is a status bar with the message "[ Read 21 lines ]". Below the status bar, there is a menu bar with the following options: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, ^X Exit, ^R Read File, ^\ Replace, ^U Uncut Text, ^T To Spell, and ^\_ Go To Line.

# Python Program - Bluetooth setup

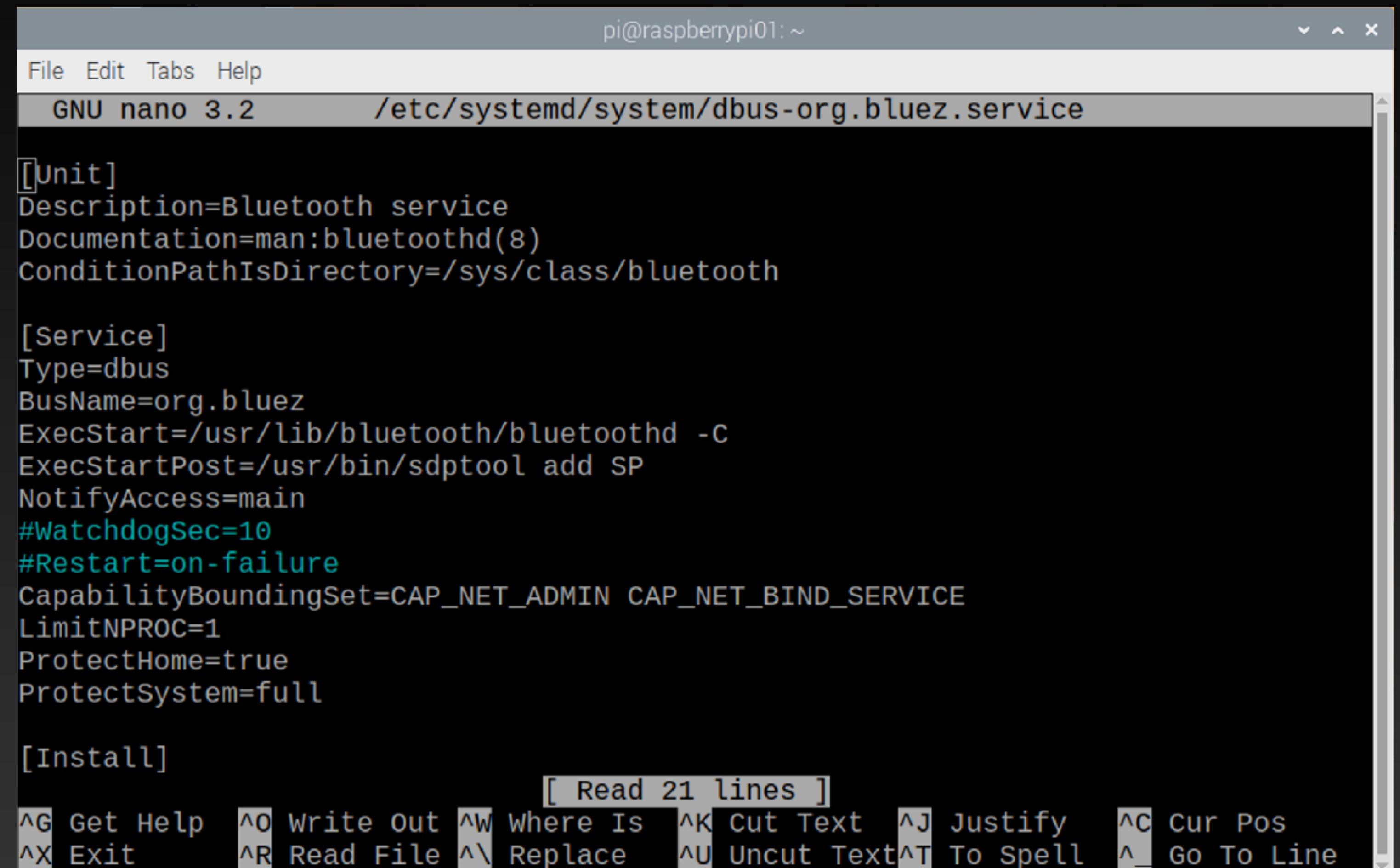
Add -C at the end of the ExecStart = line.

Also add another line after the ExecStart line.

The final two lines should look like below:

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

ExecStartPost=/usr/bin/sdptool add SP

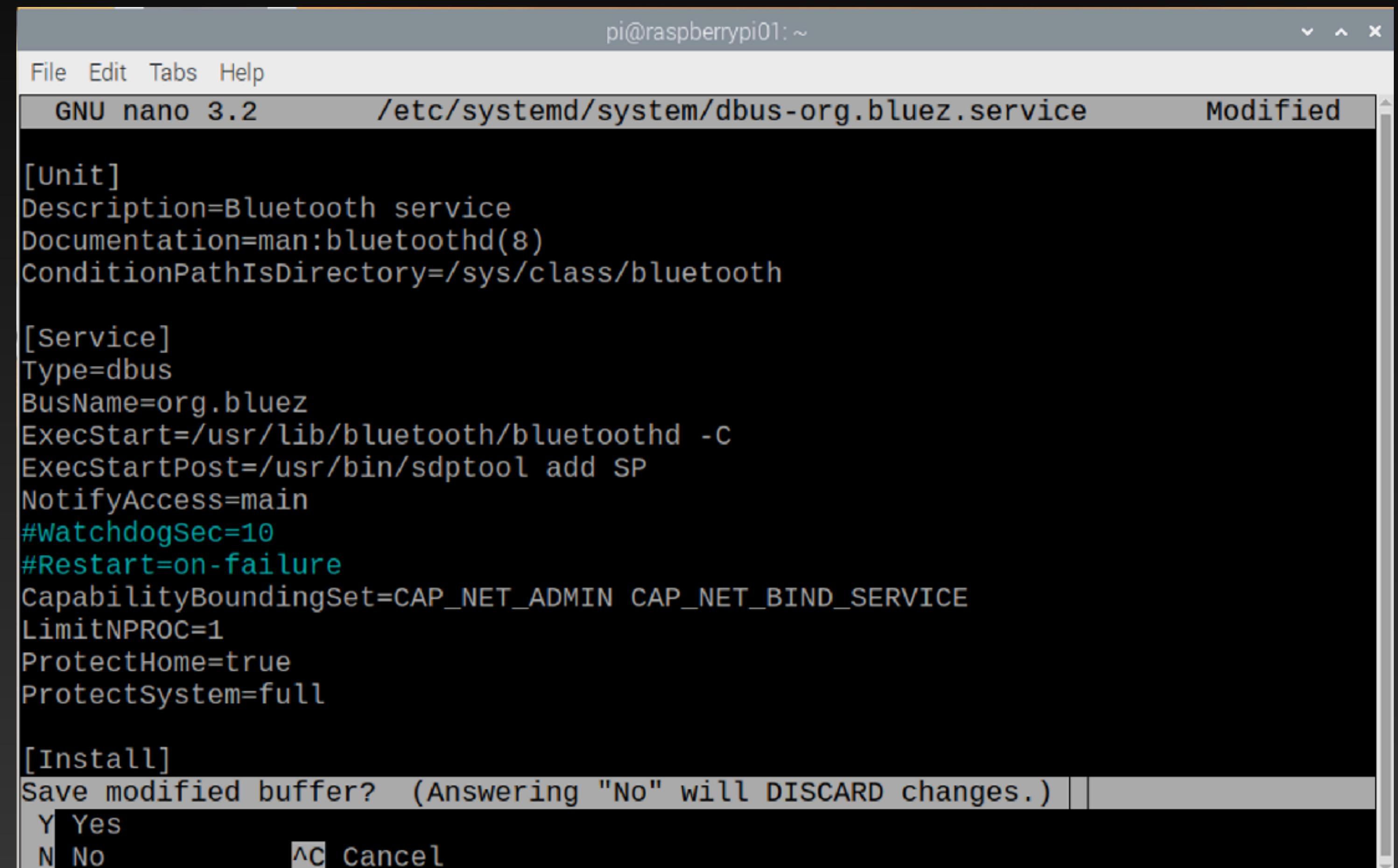


```
pi@raspberrypi01:~  
File Edit Tabs Help  
GNU nano 3.2      /etc/systemd/system/dbus-org.bluez.service  
  
[Unit]  
Description=Bluetooth service  
Documentation=man:bluetoothd(8)  
ConditionPathIsDirectory=/sys/class/bluetooth  
  
[Service]  
Type=dbus  
BusName=org.bluez  
ExecStart=/usr/lib/bluetooth/bluetoothd -c  
ExecStartPost=/usr/bin/sdptool add SP  
NotifyAccess=main  
#WatchdogSec=10  
#Restart=on-failure  
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE  
LimitNPROC=1  
ProtectHome=true  
ProtectSystem=full  
  
[Install]  
[ Read 21 lines ]  
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  
^X Exit     ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

# Python Program - Bluetooth setup

Exit and save the file by entering Ctrl + X, followed by Y.

Press enter after type in Y



```
pi@raspberrypi01: ~
File Edit Tabs Help
GNU nano 3.2      /etc/systemd/system/dbus-org.bluez.service    Modified
[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)
ConditionPathIsDirectory=/sys/class/bluetooth

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/lib/bluetooth/bluetoothd -c
ExecStartPost=/usr/bin/sdptool add SP
NotifyAccess=main
#watchdogSec=10
#Restart=on-failure
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
LimitNPROC=1
ProtectHome=true
ProtectSystem=full

[Install]
Save modified buffer? (Answering "No" will DISCARD changes.)
Y Yes
N No          ^C Cancel
```

# Python Program - Bluetooth setup

Exit and save the file by entering Ctrl + X, followed by Y.

Press enter after type in Y, you will be back to terminal screen

```
pi@raspberrypi01: ~
File Edit Tabs Help
GNU nano 3.2      /etc/systemd/system/dbus-org.bluez.service      Modified
[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)
ConditionPathIsDirectory=/sys/class/bluetooth

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/lib/bluetooth/bluetoothd -c
ExecStartPost=/usr/bin/sdptool add SP
NotifyAccess=main
#WatchdogSec=10
#Restart=on-failure
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
LimitNPROC=1
ProtectHome=true
ProtectSystem=full

[Install]
File Name to Write: /etc/systemd/system/dbus-org.bluez.service
^G Get Help          M-D DOS Format        M-A Append        M-B Backup File
^C Cancel           M-M Mac Format        M-P Prepend       ^T To Files
```

# Python Program - Bluetooth setup

\$ sudo reboot

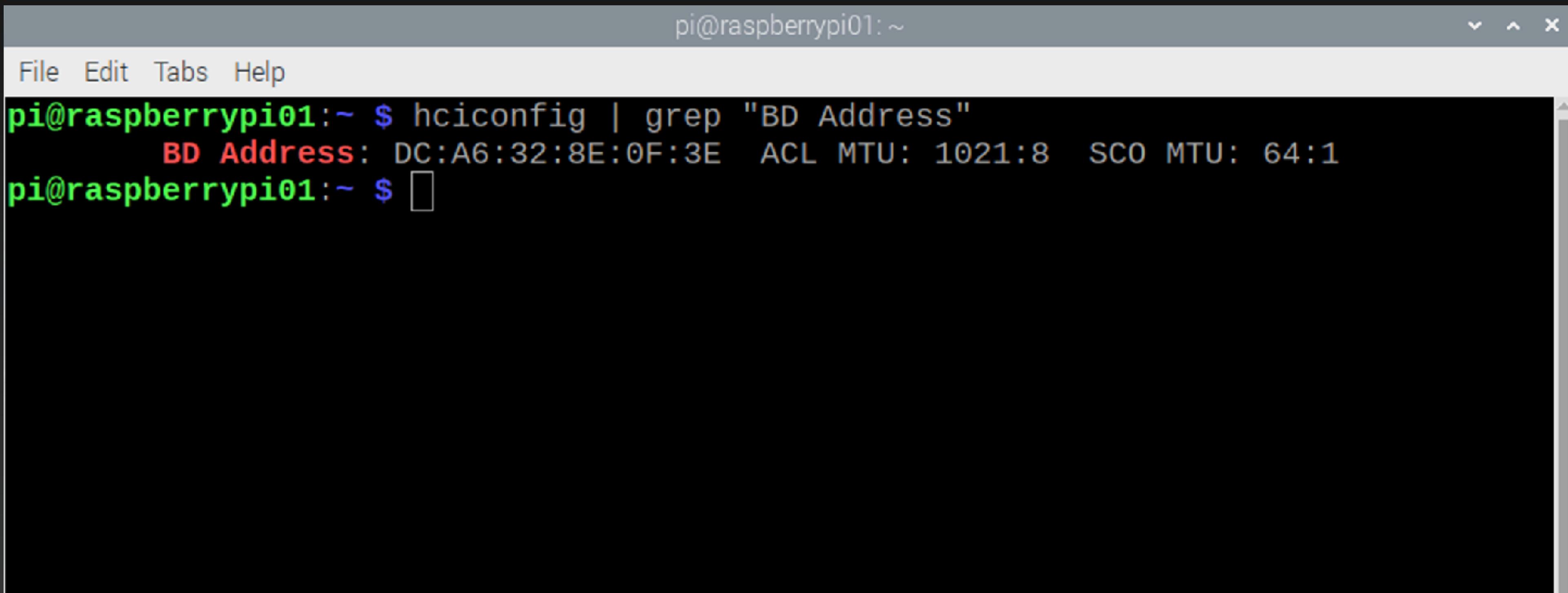
Press enter, your  
Raspberry Pi will  
reboot

```
pi@raspberrypi01:~  
File Edit Tabs Help  
pi@raspberrypi01:~ $ sudo apt-get install bluetooth libbluetooth-dev  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
bluetooth is already the newest version (5.50-1.2-deb10u1+rpt2).  
libbluetooth-dev is already the newest version (5.50-1.2-deb10u1+rpt2).  
0 upgraded, 0 newly installed, 0 to remove and 203 not upgraded.  
pi@raspberrypi01:~ $ sudo python3 -m pip install pybluez  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Requirement already satisfied: pybluez in /usr/local/lib/python3.7/dist-packages  
(0.23)  
pi@raspberrypi01:~ $ sudo nano /etc/systemd/system/dbus-org.bluez.service  
pi@raspberrypi01:~ $ sudo nano /etc/systemd/system/dbus-org.bluez.service  
pi@raspberrypi01:~ $ sudo reboot
```

# Bluetooth Pairing with Android Devices

Open your terminal, type below command:

```
$ hciconfig | grep "BD Address"
```



A screenshot of a terminal window titled "pi@raspberrypi01:~". The window has a standard Linux terminal interface with a menu bar (File, Edit, Tabs, Help) and a scroll bar on the right. The terminal displays the command \$ hciconfig | grep "BD Address" followed by its output. The output shows the BD Address as DC:A6:32:8E:0F:3E and other parameters like ACL MTU and SCO MTU.

```
pi@raspberrypi01:~ $ hciconfig | grep "BD Address"
      BD Address: DC:A6:32:8E:0F:3E  ACL MTU: 1021:8  SCO MTU: 64:1
pi@raspberrypi01:~ $ 
```

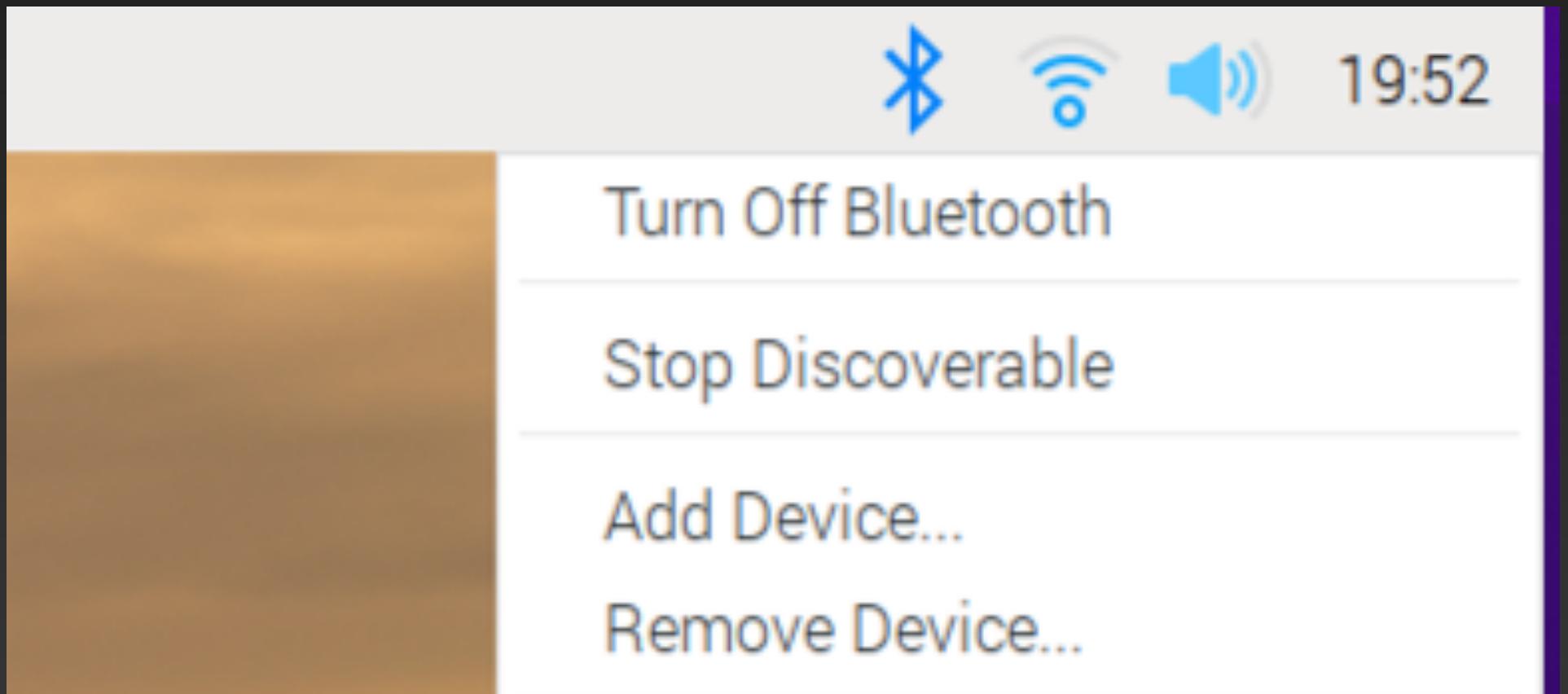
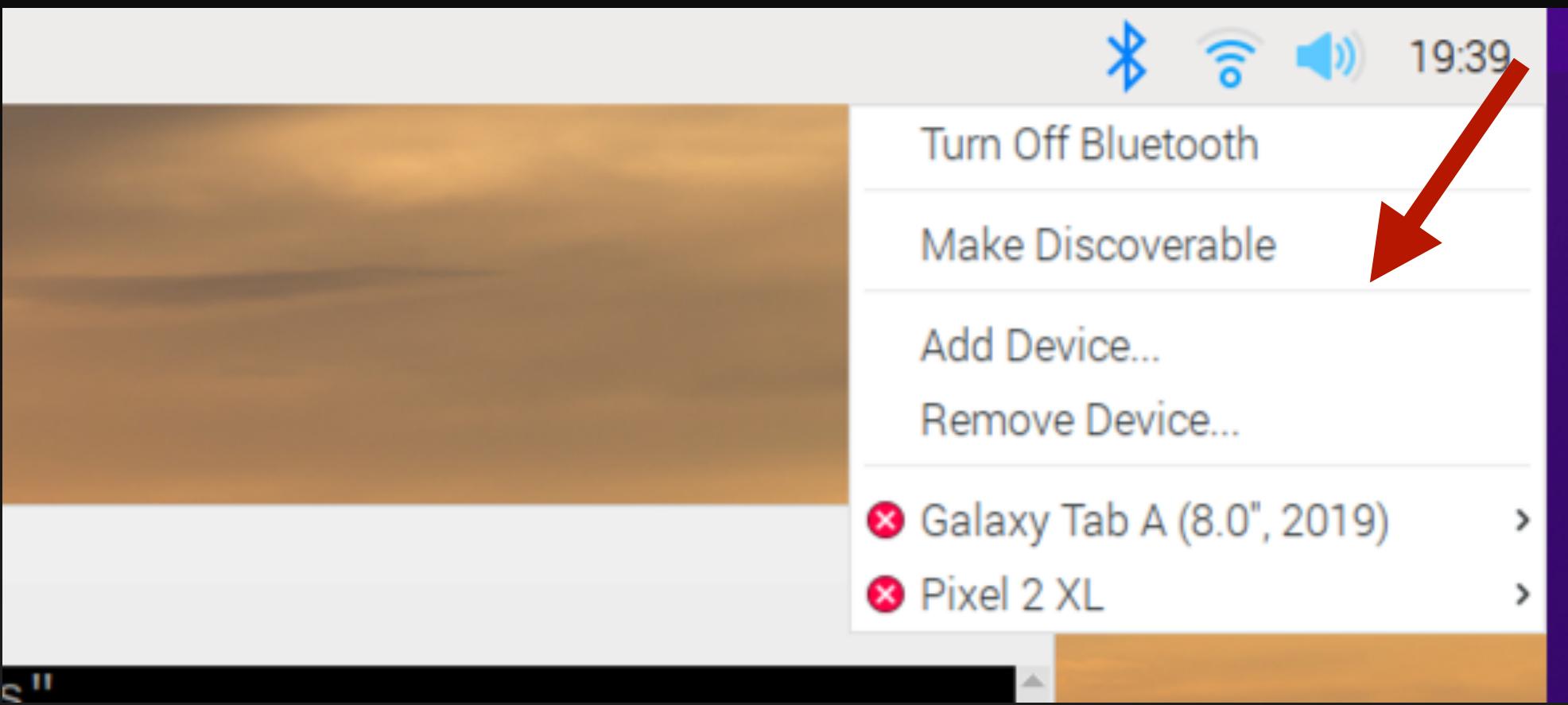
# Bluetooth Pairing with Android Devices

**MAC address:** A media access control address (MAC address) is a unique identifier assigned to a network interface controller (NIC) for use as a network address in communications within a network segment. This use is common in most IEEE 802 networking technologies, including Ethernet, Wi-Fi, and Bluetooth.

# Bluetooth Pairing with Android Devices

Config your raspberry Pi to discoverable.

First make sure the bluetooth is turned on, then click on Make Discoverable, you should see the bluetooth icon is flashing in green and blue color.



# Bluetooth Pairing with Android Devices

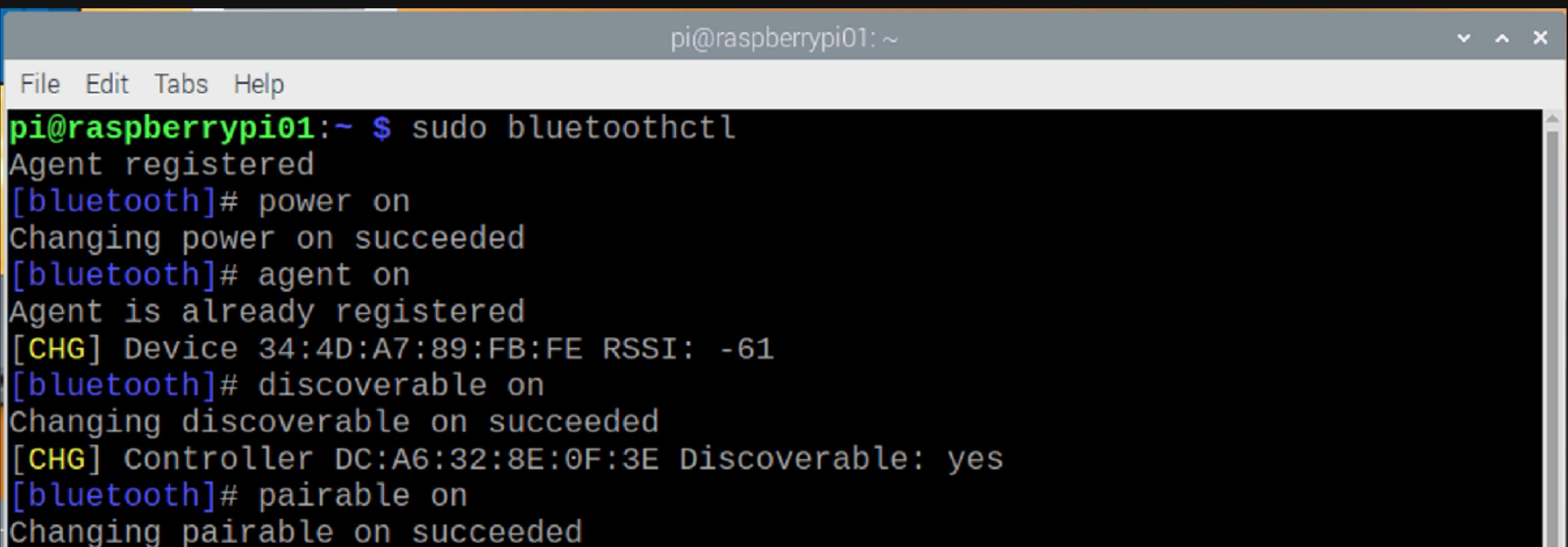
Go to your Android device, find connection under settings.

Pair raspberry pi with your device.

# Bluetooth Pairing with Android Devices

Debug from terminal.

```
$ sudo bluetoothctl  
# power on  
# agent on  
# discoverable on  
# pairable on
```



A screenshot of a terminal window titled "pi@raspberrypi01: ~". The window shows the command-line interface for managing Bluetooth connections on a Raspberry Pi. The user has run the following commands:

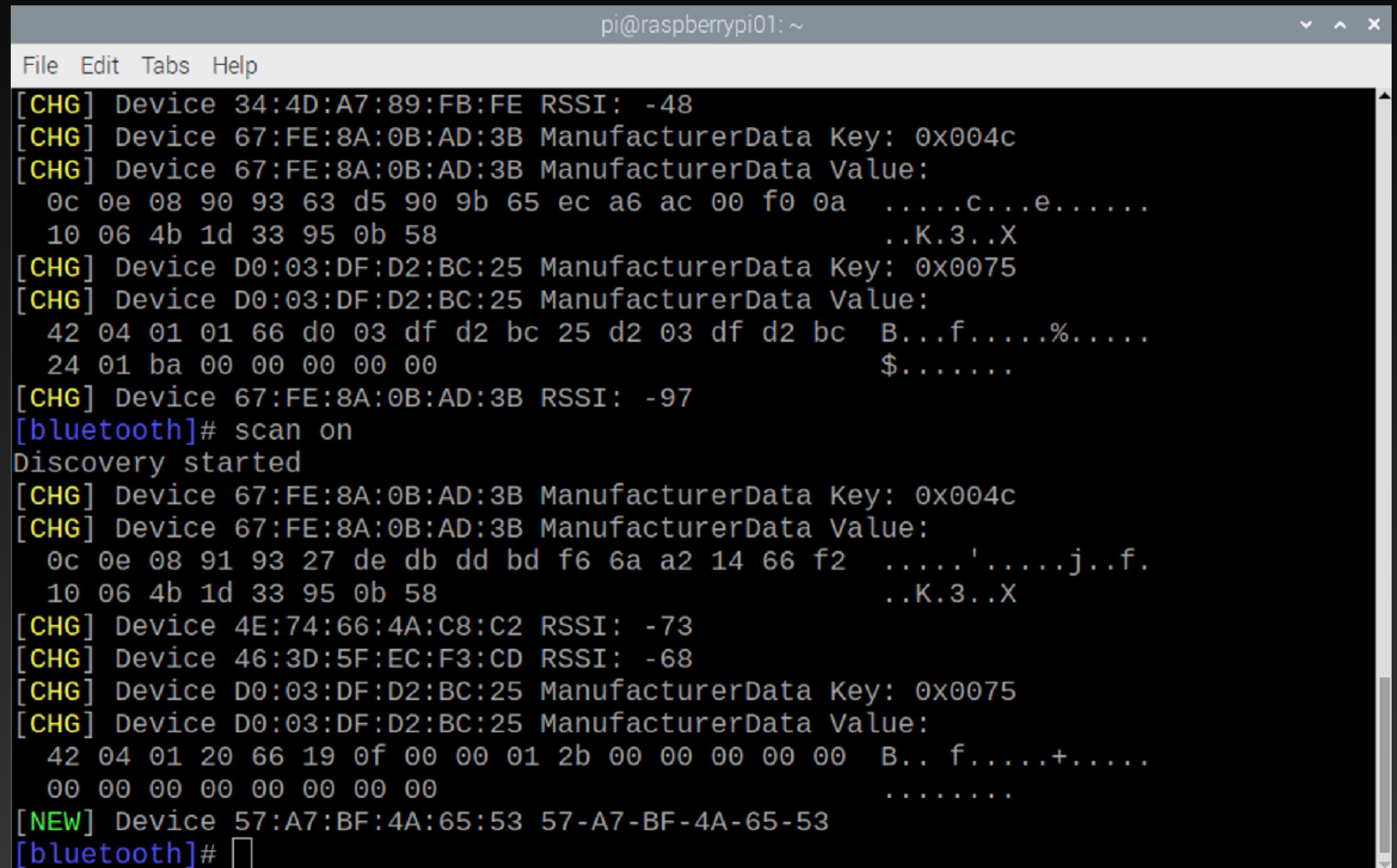
```
pi@raspberrypi01:~ $ sudo bluetoothctl  
Agent registered  
[bluetooth]# power on  
Changing power on succeeded  
[bluetooth]# agent on  
Agent is already registered  
[CHG] Device 34:4D:A7:89:FB:FE RSSI: -61  
[bluetooth]# discoverable on  
Changing discoverable on succeeded  
[CHG] controller DC:A6:32:8E:0F:3E Discoverable: yes  
[bluetooth]# pairable on  
Changing pairable on succeeded
```

# Bluetooth Pairing with Android Devices

```
# scan on
```

After the last command “*scan on*”, you will see your Bluetooth device (Mobile phone) in the list. Make sure that your mobile has Bluetooth turned on and visible by nearby devices. Then copy the MAC address of your device and pair it by using given command:

```
# pair <address of your phone>
```



The screenshot shows a terminal window titled "pi@raspberrypi01:~". The window contains a list of Bluetooth devices discovered by the system. The output is as follows:

```
pi@raspberrypi01:~
```

```
File Edit Tabs Help
```

```
[CHG] Device 34:4D:A7:89:FB:FE RSSI: -48
[CHG] Device 67:FE:8A:0B:AD:3B ManufacturerData Key: 0x004c
[CHG] Device 67:FE:8A:0B:AD:3B ManufacturerData Value:
  0c 0e 08 90 93 63 d5 90 9b 65 ec a6 ac 00 f0 oa  .....c...e.....
  10 06 4b 1d 33 95 0b 58                                ..K.3..X
[CHG] Device D0:03:DF:D2:BC:25 ManufacturerData Key: 0x0075
[CHG] Device D0:03:DF:D2:BC:25 ManufacturerData Value:
  42 04 01 01 66 d0 03 df d2 bc 25 d2 03 df d2 bc  B...f....%.....
  24 01 ba 00 00 00 00 00                               $.....
[CHG] Device 67:FE:8A:0B:AD:3B RSSI: -97
[bluetooth]# scan on
Discovery started
[CHG] Device 67:FE:8A:0B:AD:3B ManufacturerData Key: 0x004c
[CHG] Device 67:FE:8A:0B:AD:3B ManufacturerData Value:
  0c 0e 08 91 93 27 de db dd bd f6 6a a2 14 66 f2  .....'....j..f.
  10 06 4b 1d 33 95 0b 58                                ..K.3..X
[CHG] Device 4E:74:66:4A:C8:C2 RSSI: -73
[CHG] Device 46:3D:5F:EC:F3:CD RSSI: -68
[CHG] Device D0:03:DF:D2:BC:25 ManufacturerData Key: 0x0075
[CHG] Device D0:03:DF:D2:BC:25 ManufacturerData Value:
  42 04 01 20 66 19 0f 00 00 01 2b 00 00 00 00 00  B.. f....+.....
  00 00 00 00 00 00 00 00                               .....
[NEW] Device 57:A7:BF:4A:65:53 57-A7-BF-4A-65-53
[bluetooth]#
```

# Python Program - LED control through bluetooth

Notice that what is returned into variable data is byte objects which is actually arrays of integers.

Here, we are using the built-in decode function to convert them into their equivalent ASCII characters.

In Python3, utf-8 is the default encoding and decoding parameter.

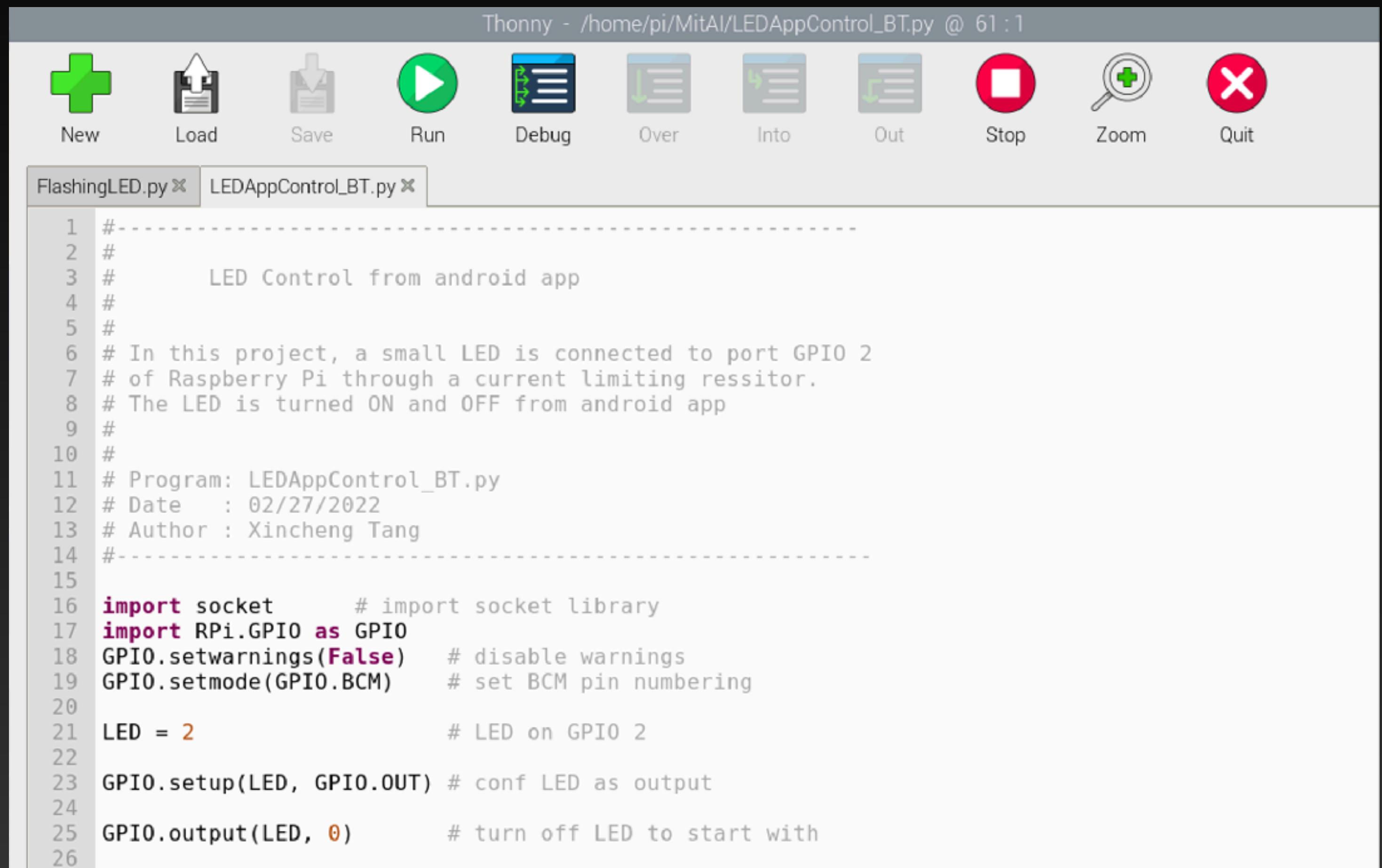
```
Decode MAC to ASCII (utf-8 in python 3)
#-----
#
#      decode Pi signal/data
#
#-----
```

```
try:
    while True:
        data = client.recv(1024)          # Recieve data bytes
        if data.decode('utf-8') == '1':    # 1 received
            GPIO.output(LED, 1)         # LED ON
        elif data.decode('utf-8') == '0':  # 0 received
            GPIO.output(LED, 0)         # LED OFF
    except KeyboardInterrupt:           # Keyboard interrupt
        client.close()
        s.close()
```

# Python Program - LED control through bluetooth

Develop the python program  
to for bluetooth control.

GPIO library setup

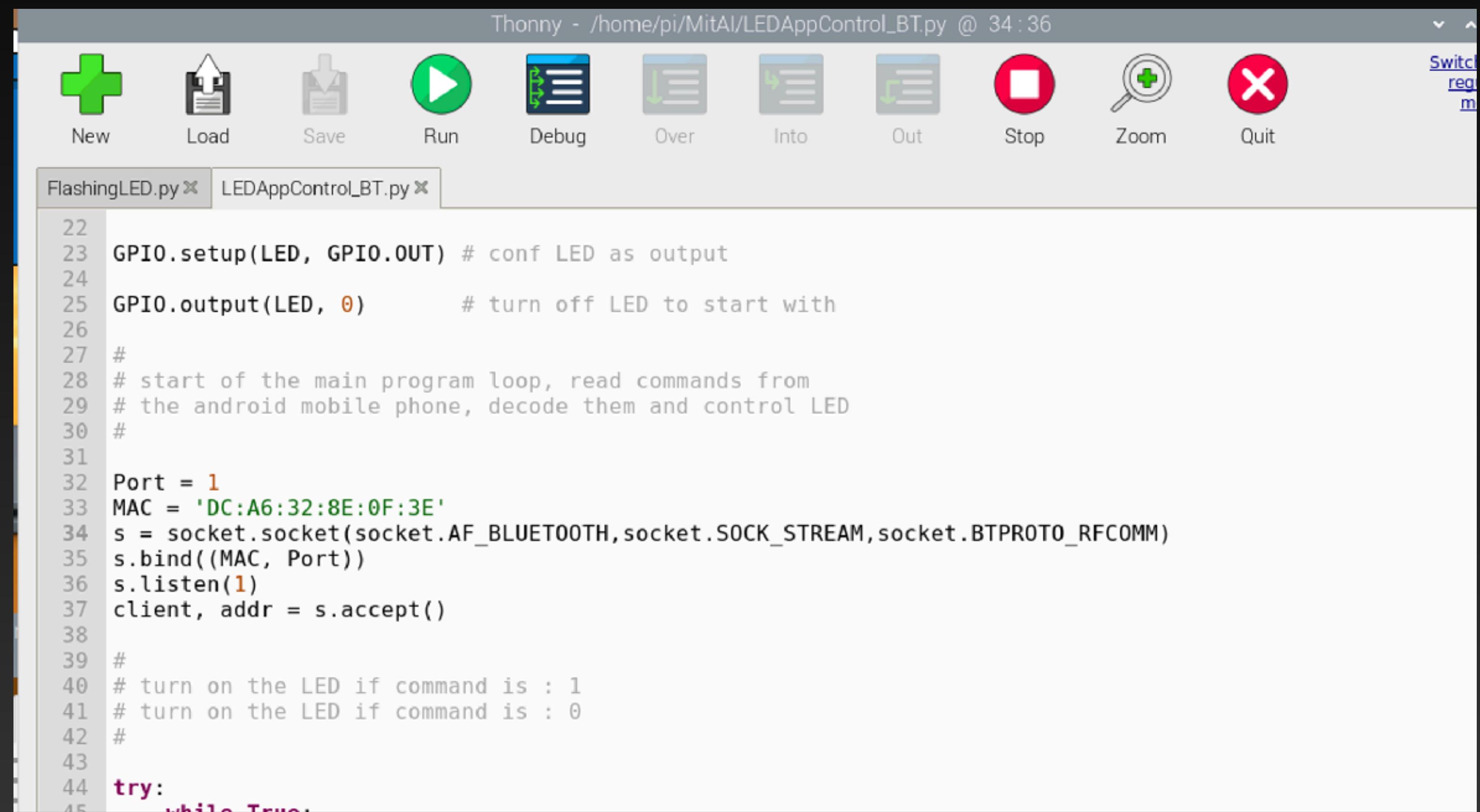


The screenshot shows the Thonny Python IDE interface. The title bar reads "Thonny - /home/pi/MitAI/LEDAppControl\_BT.py @ 61 : 1". The toolbar contains icons for New (green plus), Load, Save, Run (play), Debug, Over, Into, Out, Stop (red square), Zoom (magnifying glass), and Quit (red X). Below the toolbar, two tabs are visible: "FlashingLED.py" and "LEDAppControl\_BT.py", with "LEDAppControl\_BT.py" being the active tab. The code editor displays the following Python script:

```
1  #-----
2  #
3  #      LED Control from android app
4  #
5  #
6  # In this project, a small LED is connected to port GPIO 2
7  # of Raspberry Pi through a current limiting ressistor.
8  # The LED is turned ON and OFF from android app
9  #
10 #
11 # Program: LEDAppControl_BT.py
12 # Date   : 02/27/2022
13 # Author : Xincheng Tang
14 #
15
16 import socket      # import socket library
17 import RPi.GPIO as GPIO
18 GPIO.setwarnings(False)    # disable warnings
19 GPIO.setmode(GPIO.BCM)    # set BCM pin numbering
20
21 LED = 2           # LED on GPIO 2
22
23 GPIO.setup(LED, GPIO.OUT) # conf LED as output
24
25 GPIO.output(LED, 0)     # turn off LED to start with
26
```

# Python Program - LED control through bluetooth

Setup socket to receive signal from physical device.

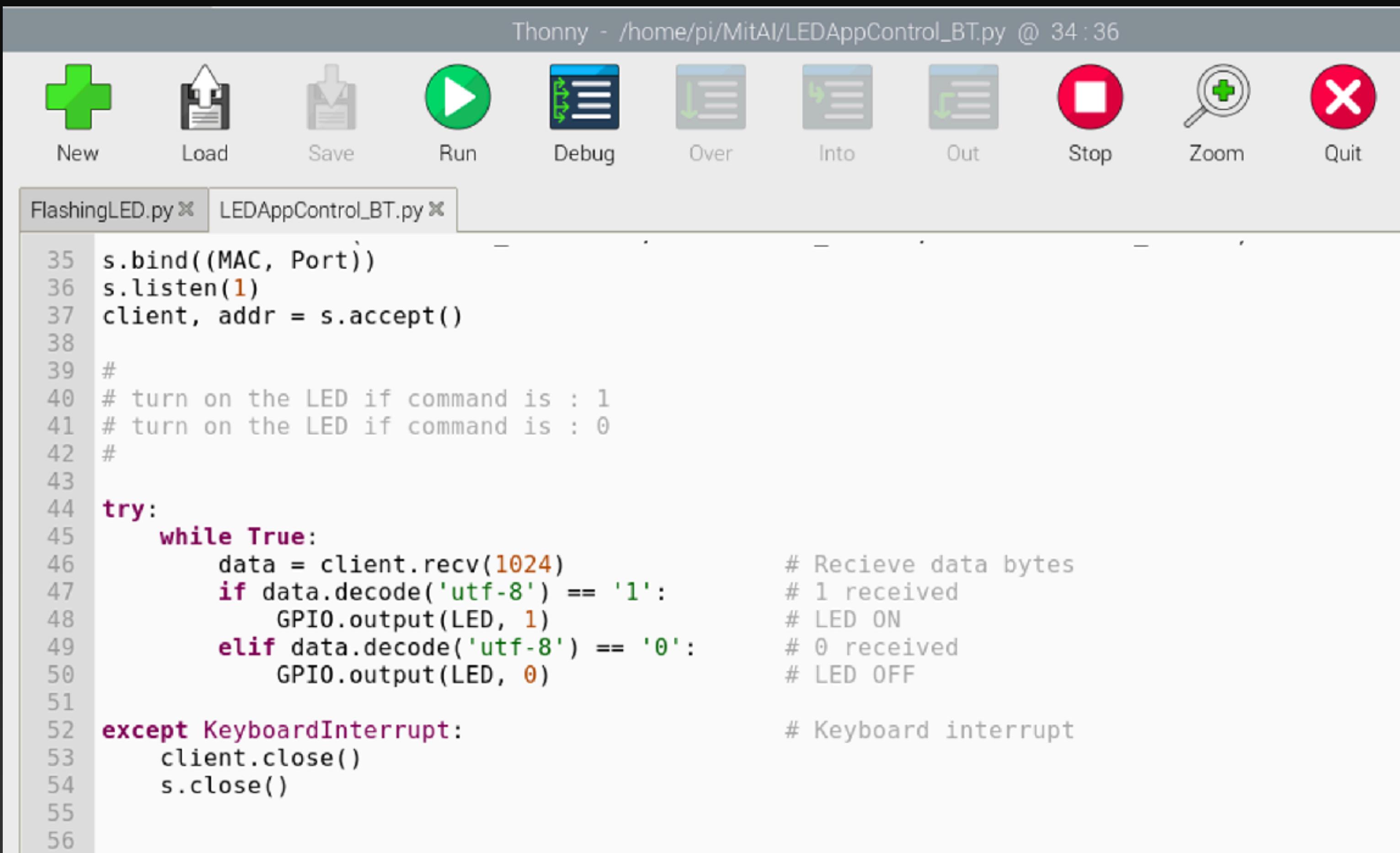


The screenshot shows the Thonny Python IDE interface. The title bar reads "Thonny - /home/pi/MitAI/LEDAppControl\_BT.py @ 34:36". The toolbar contains icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. Below the toolbar, two tabs are visible: "FlashingLED.py x" and "LEDAppControl\_BT.py x", with "LEDAppControl\_BT.py x" being the active tab. The code editor displays the following Python script:

```
22 GPIO.setup(LED, GPIO.OUT) # conf LED as output
23 GPIO.output(LED, 0)      # turn off LED to start with
24 #
25 # start of the main program loop, read commands from
26 # the android mobile phone, decode them and control LED
27 #
28
29
30
31
32 Port = 1
33 MAC = 'DC:A6:32:8E:0F:3E'
34 s = socket.socket(socket.AF_BLUETOOTH,socket.SOCK_STREAM,socket.BTPROTO_RFCOMM)
35 s.bind((MAC, Port))
36 s.listen(1)
37 client, addr = s.accept()
38 #
39 #
40 # turn on the LED if command is : 1
41 # turn on the LED if command is : 0
42 #
43
44 try:
45     while True:
```

# Python Program - LED control through bluetooth

Decode the signal.

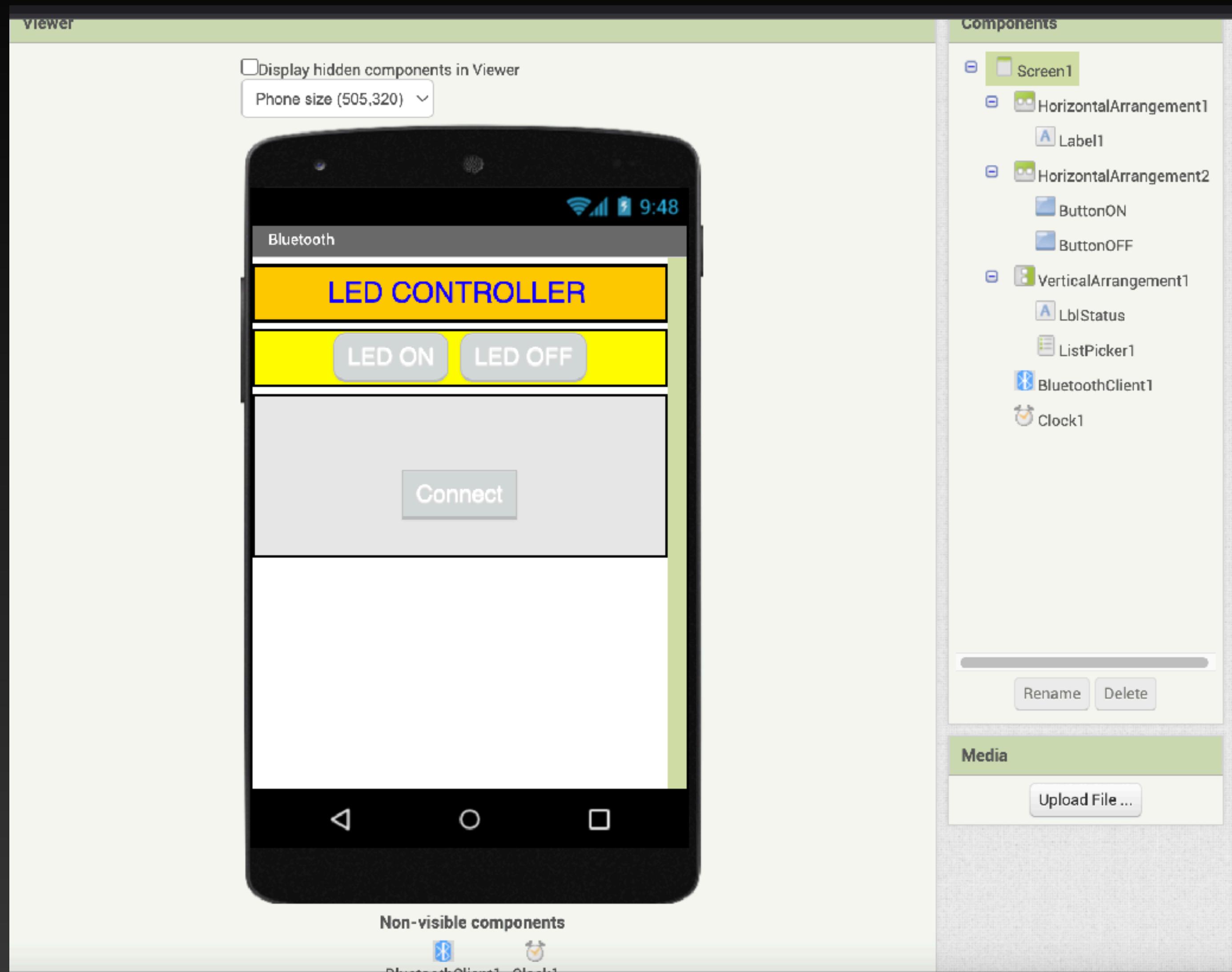


The screenshot shows the Thonny Python IDE interface. The title bar reads "Thonny - /home/pi/MitAI/LEDAppControl\_BT.py @ 34:36". The toolbar contains icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. Below the toolbar, two tabs are visible: "FlashingLED.py" (selected) and "LEDAppControl\_BT.py". The code in "LEDAppControl\_BT.py" is as follows:

```
35 s.bind((MAC, Port))
36 s.listen(1)
37 client, addr = s.accept()
38 #
39 # turn on the LED if command is : 1
40 # turn on the LED if command is : 0
41 #
42 #
43
44 try:
45     while True:
46         data = client.recv(1024)                      # Recieve data bytes
47         if data.decode('utf-8') == '1':                 # 1 received
48             GPIO.output(LED, 1)                        # LED ON
49         elif data.decode('utf-8') == '0':               # 0 received
50             GPIO.output(LED, 0)                        # LED OFF
51
52 except KeyboardInterrupt:                         # Keyboard interrupt
53     client.close()
54     s.close()
```

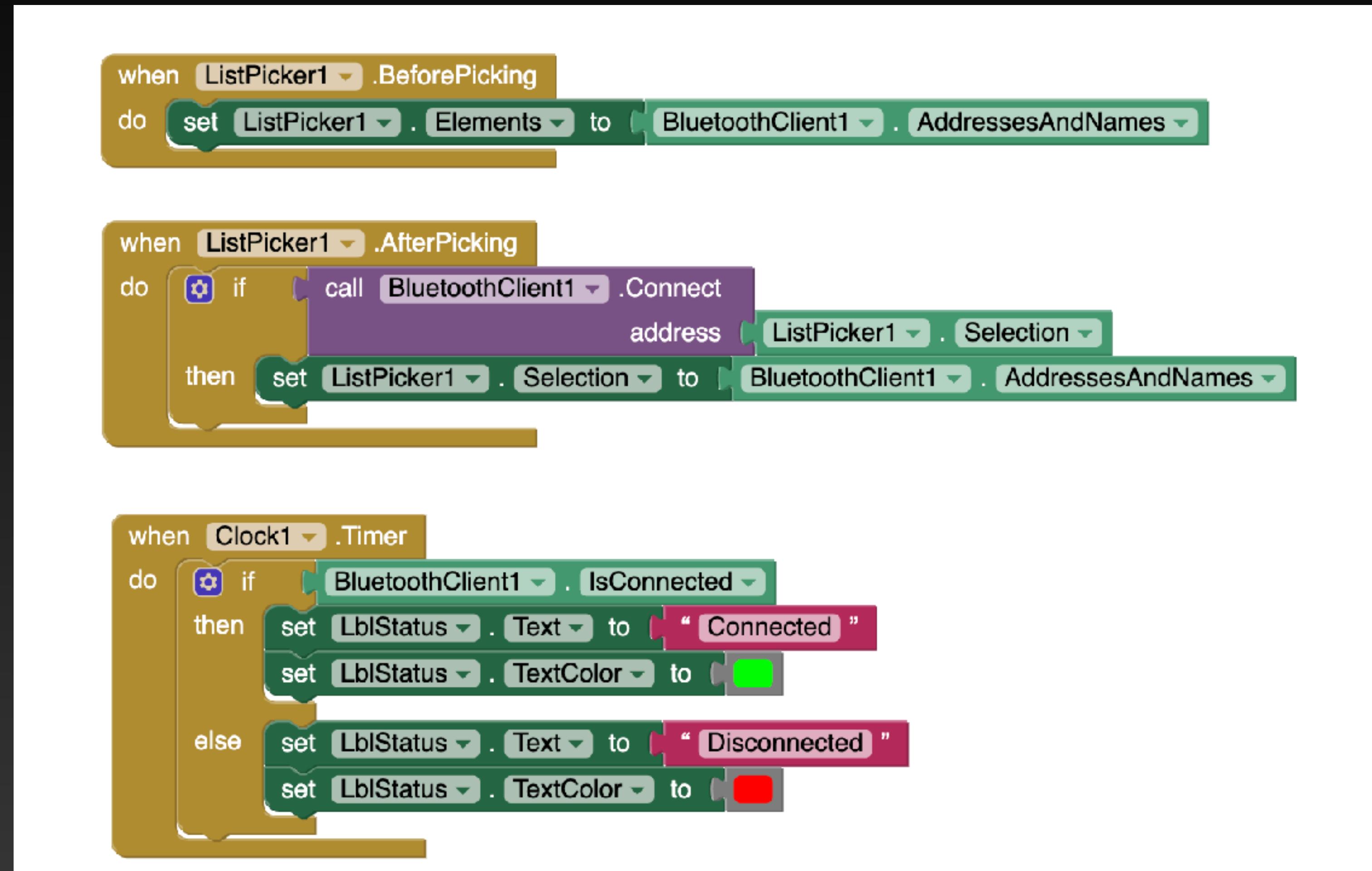
# App Inventor - LED control through bluetooth

Application designer  
build.



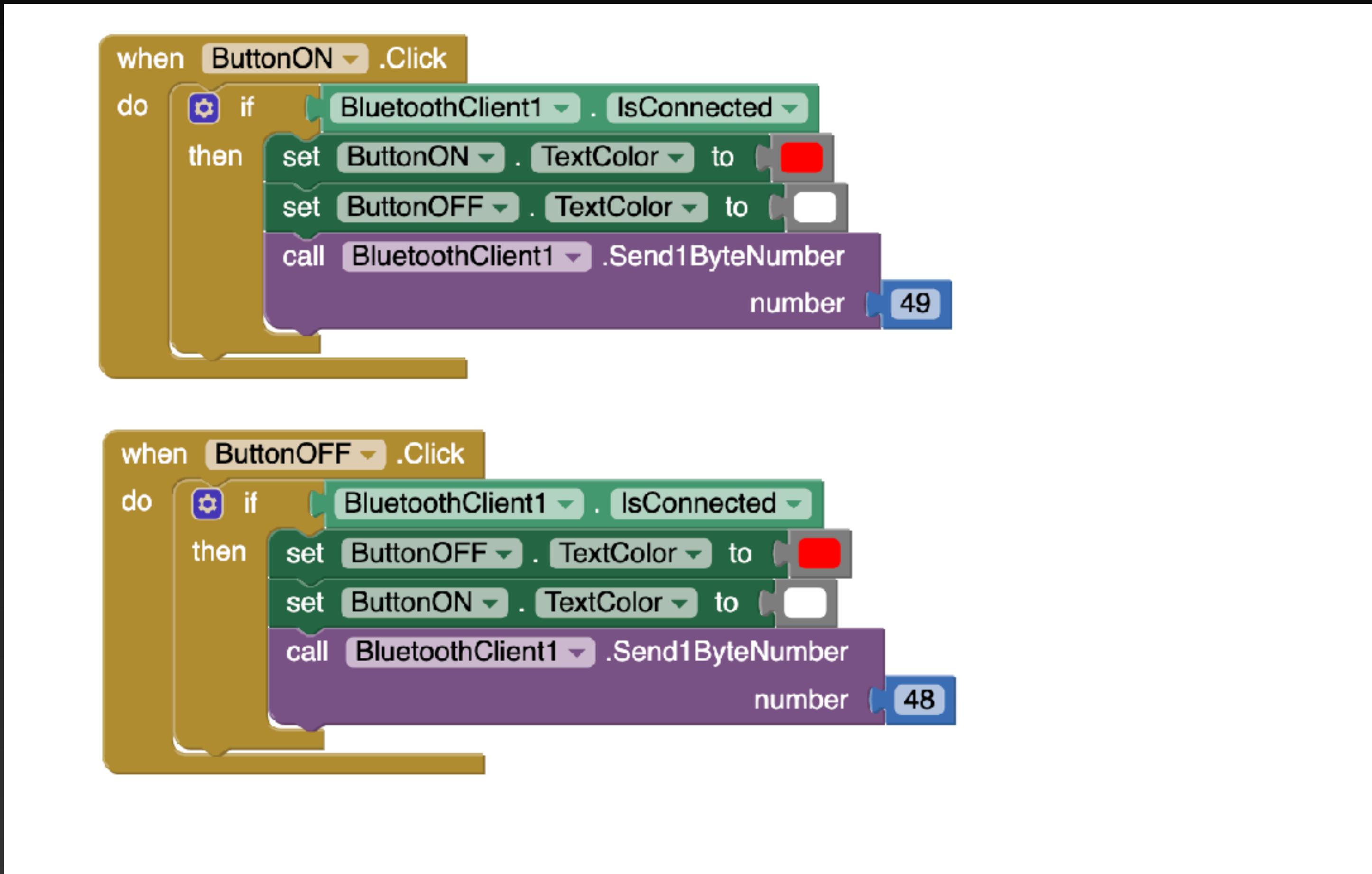
# App Inventor - LED control through bluetooth

Application blocks  
build.



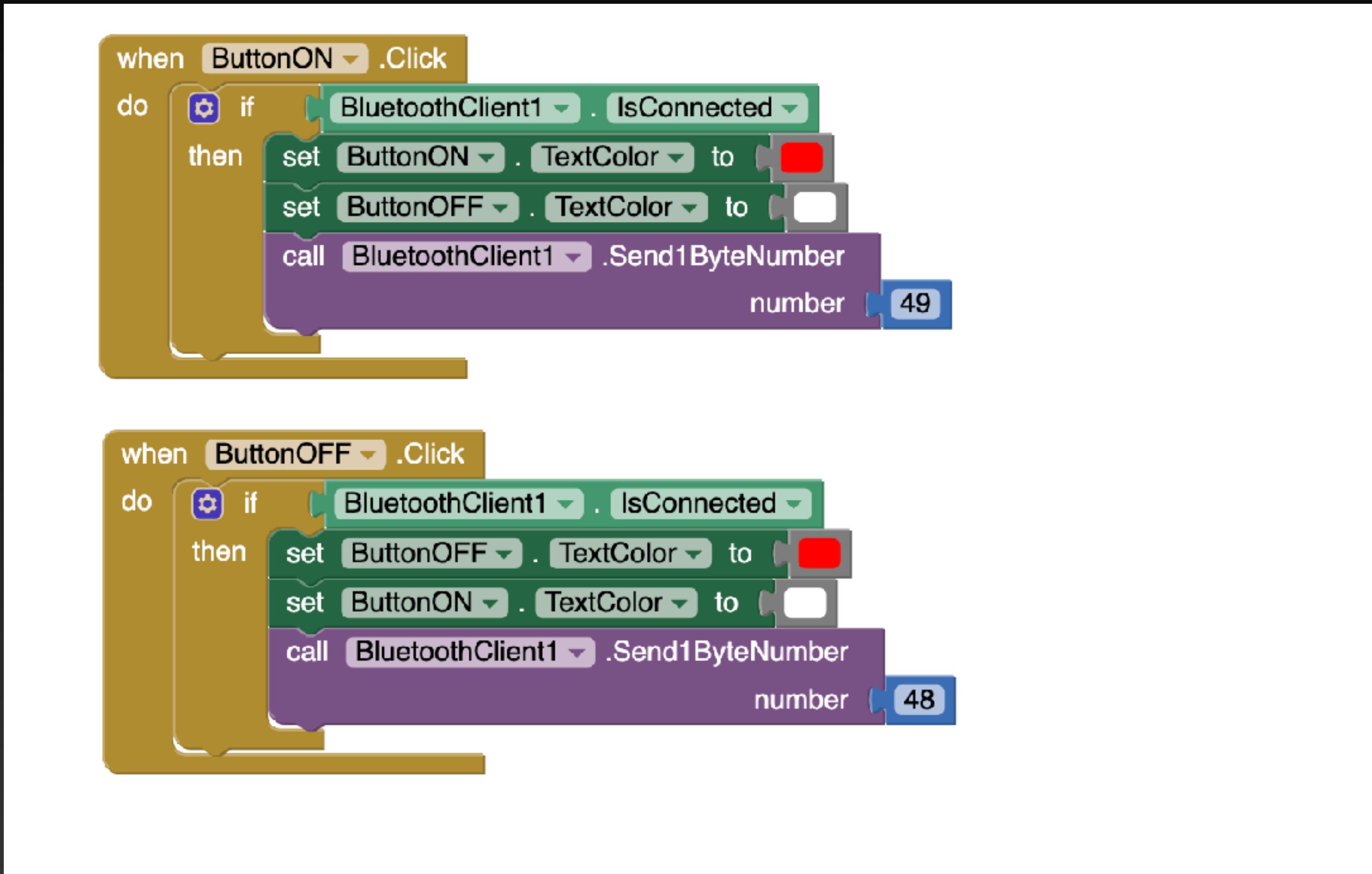
# App Inventor - LED control through bluetooth

Application blocks  
build.



# App Inventor - LED control through bluetooth

Application blocks  
build.



# LED Control App - Bluetooth

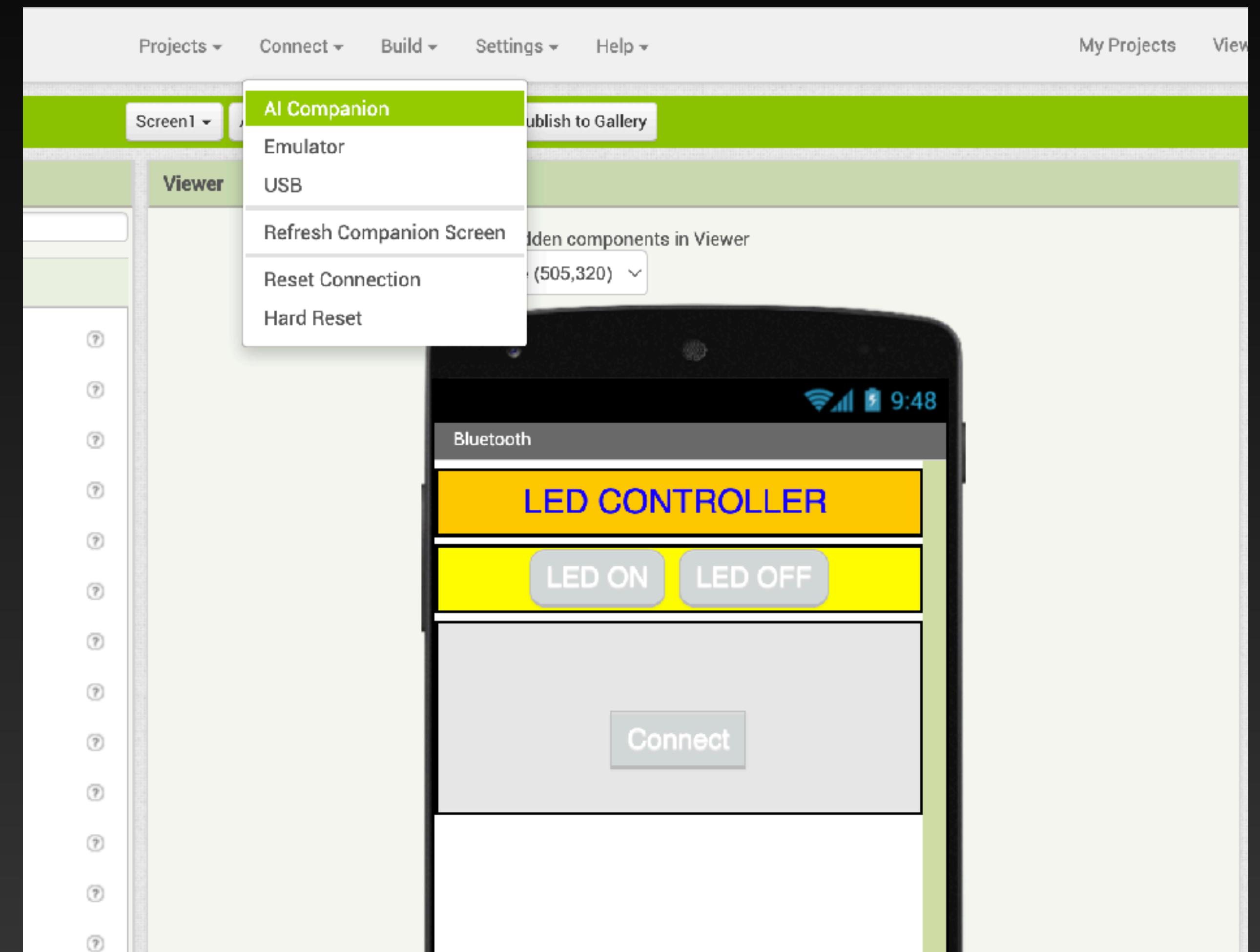
## Test steps:

- Make sure that Bluetooth is enabled on your Android device.
- Make Bluetooth discoverable by clicking the Bluetooth icon on the top right-hand side of your raspberry Pi.
- You may have to accept to pair the two Bluetooth devices. If it asks for a password, enter 1234.
- Run your python program from GUI.

# LED Control App - Bluetooth

Test steps - continued :

Generate QR code through AI companion, and scan from your android device.

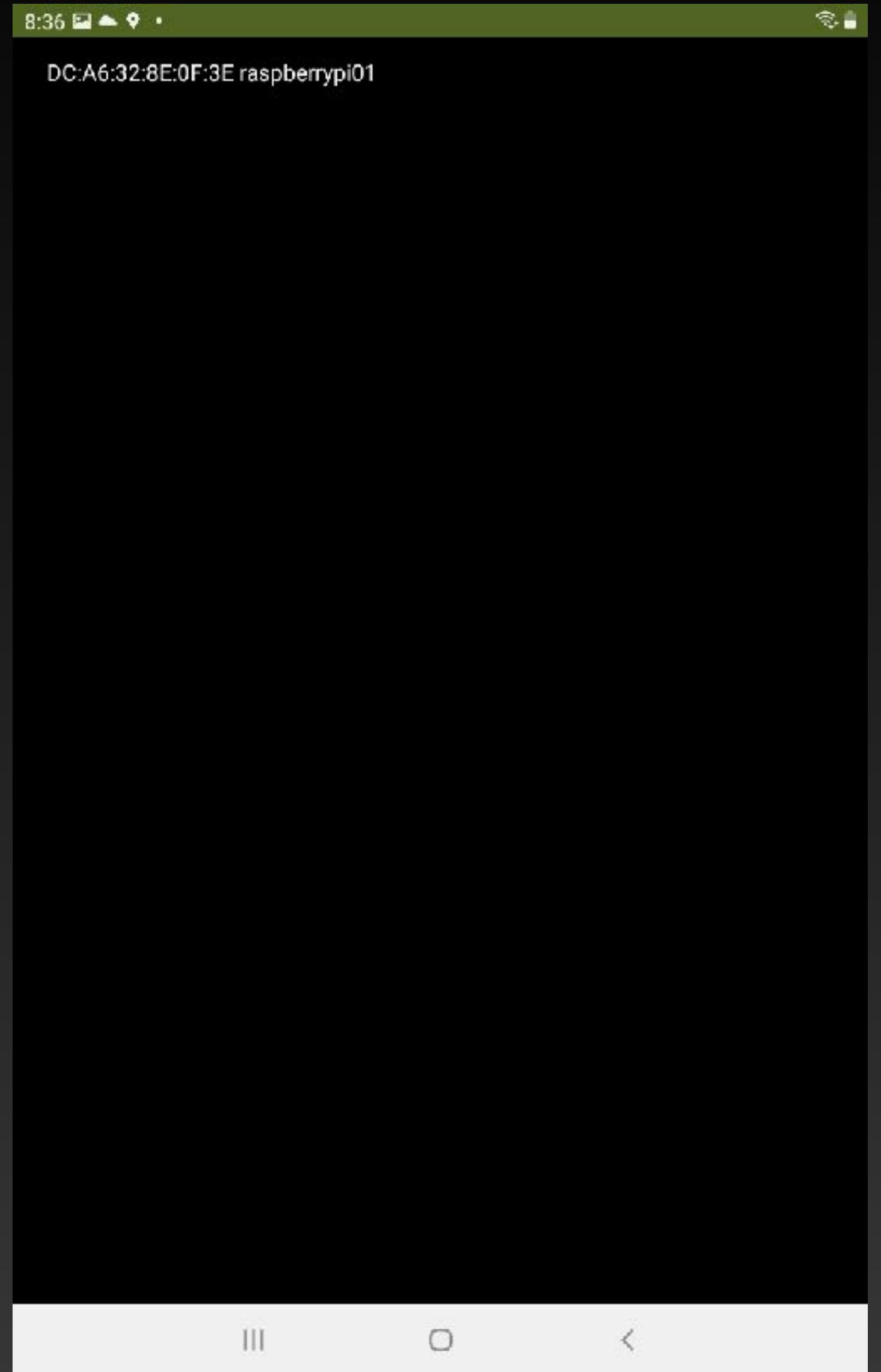


# LED Control App - Bluetooth

Test steps - continued :

Start the mobile application on your Android device and click button Connect.

You should see the reachable bluetooth devices, click on raspberry Pi.

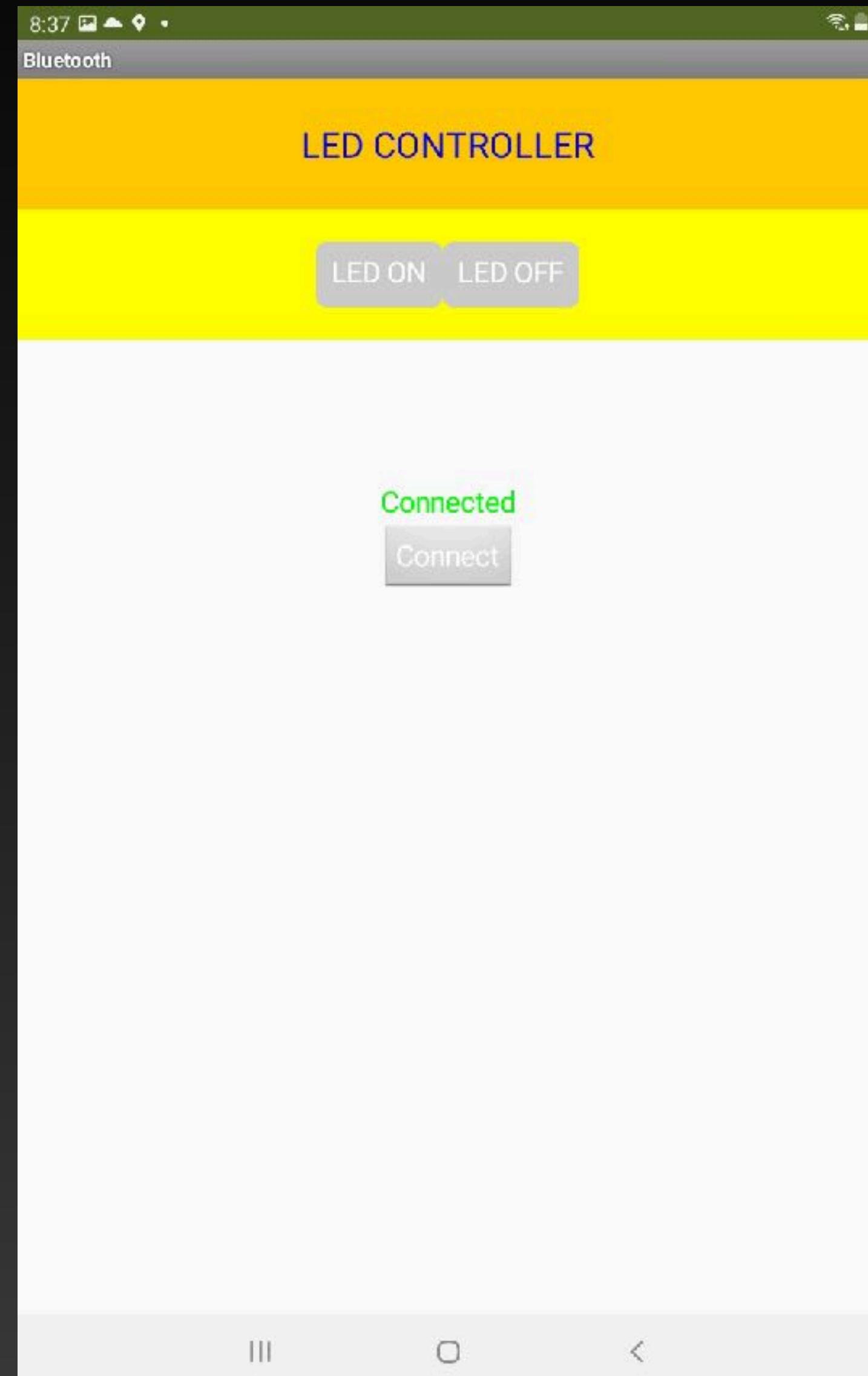


# LED Control App - Bluetooth

Test steps - continued :

After click on raspberryPi,  
your app will show  
Connected!

Now you can test your  
application.



# Python Program - LED control through bluetooth

Homework - Read through class materials to pair your android device with Raspberry Pi, finished code in class MIT app and Python control code.