

MIT AI2 204

IoT with MIT App Inventor

Fundamental

X. Tang

IoT with MIT App Inventor

Start 7:05PM

**Please update your name with studentID
In Zoom meeting**

Raspberry Pi Wi-Fi based projects

Project 1 - Getting and displaying the local WiFi parameters

Description: In this project, local Wi-Fi parameters, such as the IP address, MAC address, and signal strength of the Android device are obtained and displayed on its screen.

The Taifun WiFi component extension is used to get the local Wi-Fi parameters.

Getting and Displaying Local Wi-Fi Parameters

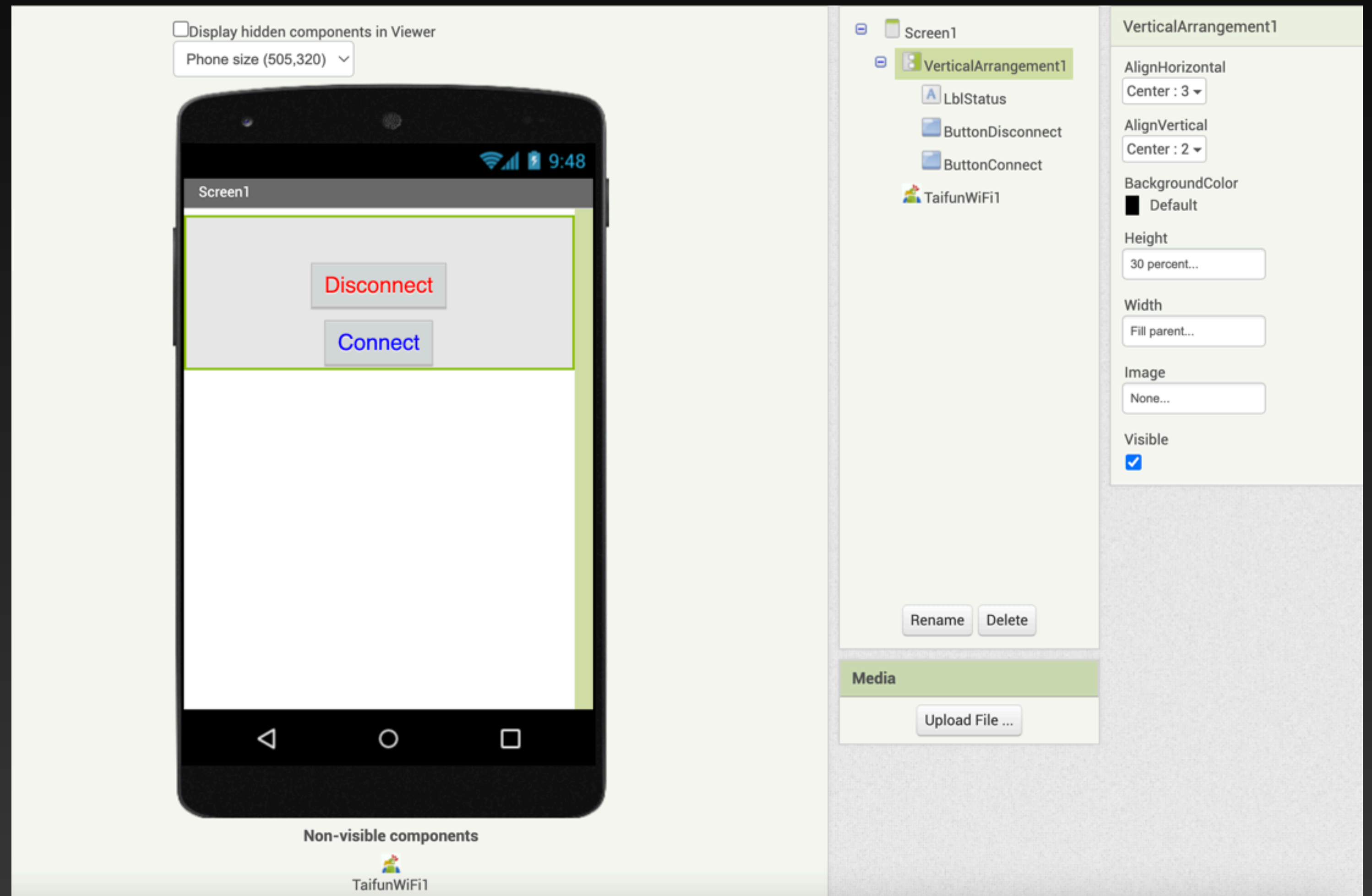
- Create a new project and name it as WIFI_TEST
- Insert a VerticalArrangement with the following configuration:

AlignHorizontal: Center: 3

AlignVertical: Center: 2

Height: 30 percent

Width: Fill parent



Getting and Displaying Local Wi-Fi Parameters

- Insert a Label with the following configuration. This Label will display the local Wi-Fi parameters:

Name: LblStatus

BackColor: None

FontBold: ticked

FontSize: 20

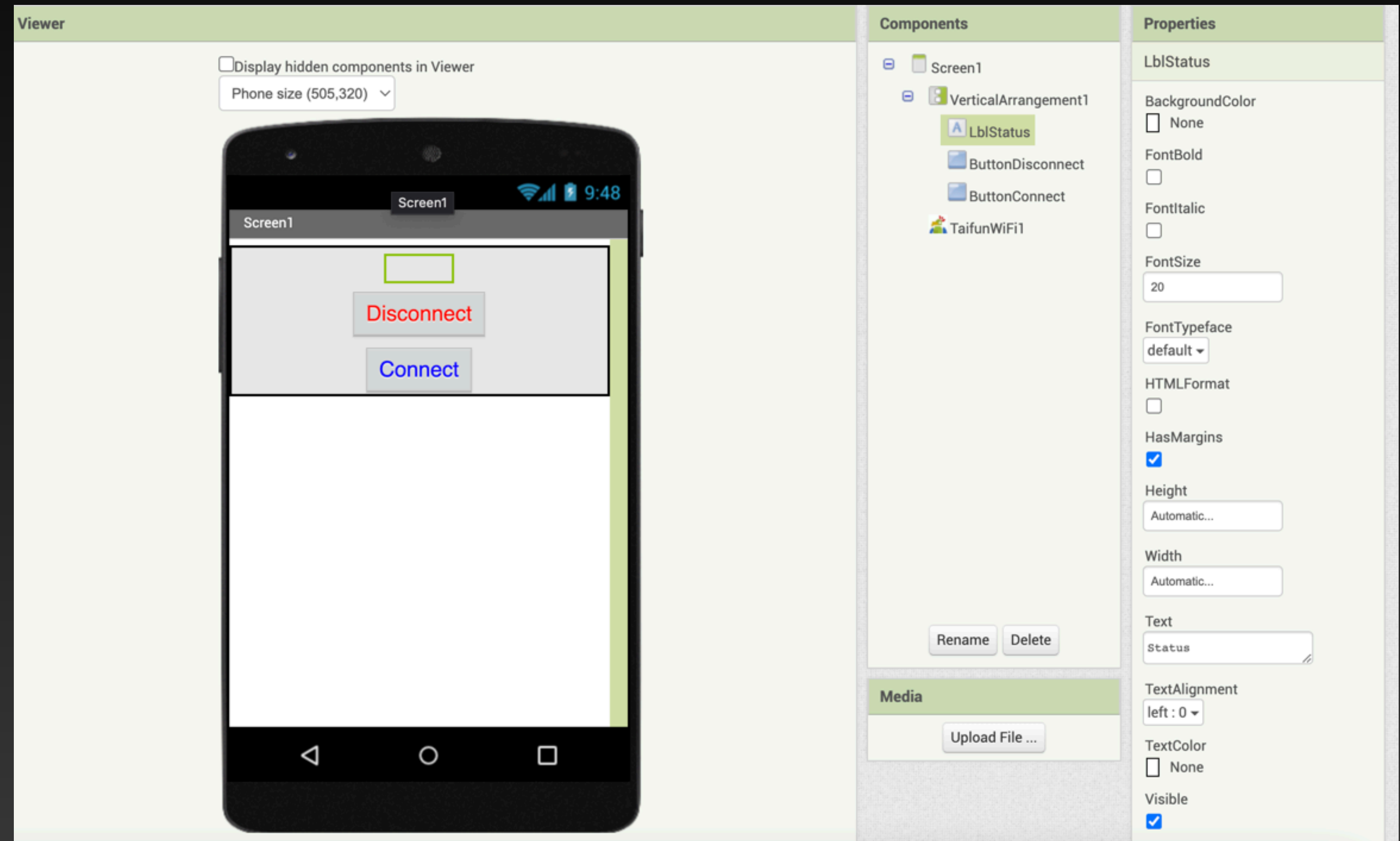
Height: Automatic

Width: Automatic

Text: Status

TextColor: None

Visible: ticked



Getting and Displaying Local Wi-Fi Parameters

- Insert a Button with the following configuration. This button will disconnect from Wi-Fi when clicked:

Name: ButtonDisconnect

BackColor: Default

FontBold: ticked

FontSize: 20

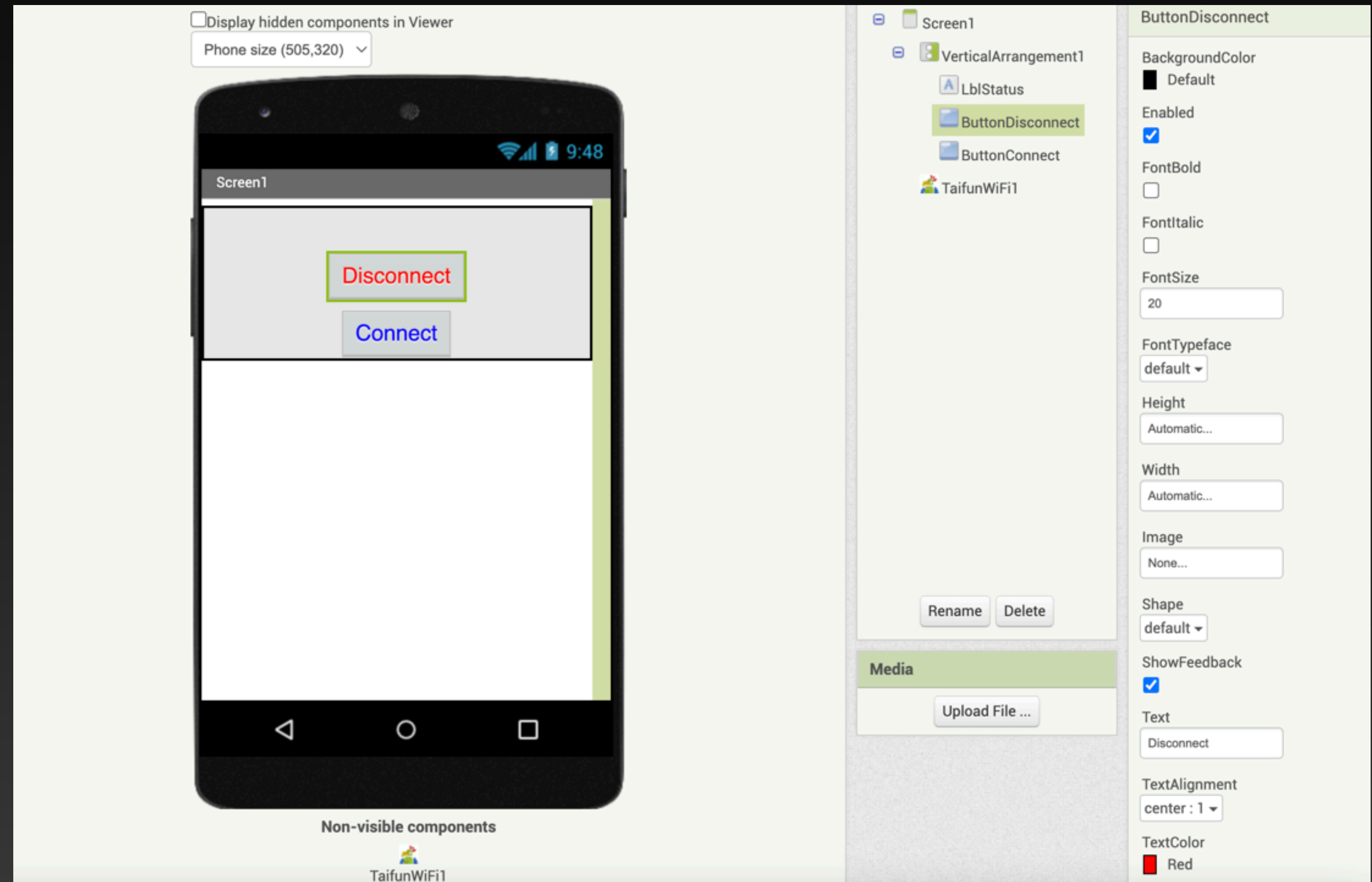
Height: Automatic

Width: Automatic

Text: Disconnect

TextColor: Red

Visible: ticked



Getting and Displaying Local Wi-Fi Parameters

- Insert a Button with the following configuration. This button will connect to Wi-Fi when clicked:

Name: ButtonConnect

BackgroudColor: Default

FontBold: ticked

FontSize: 20

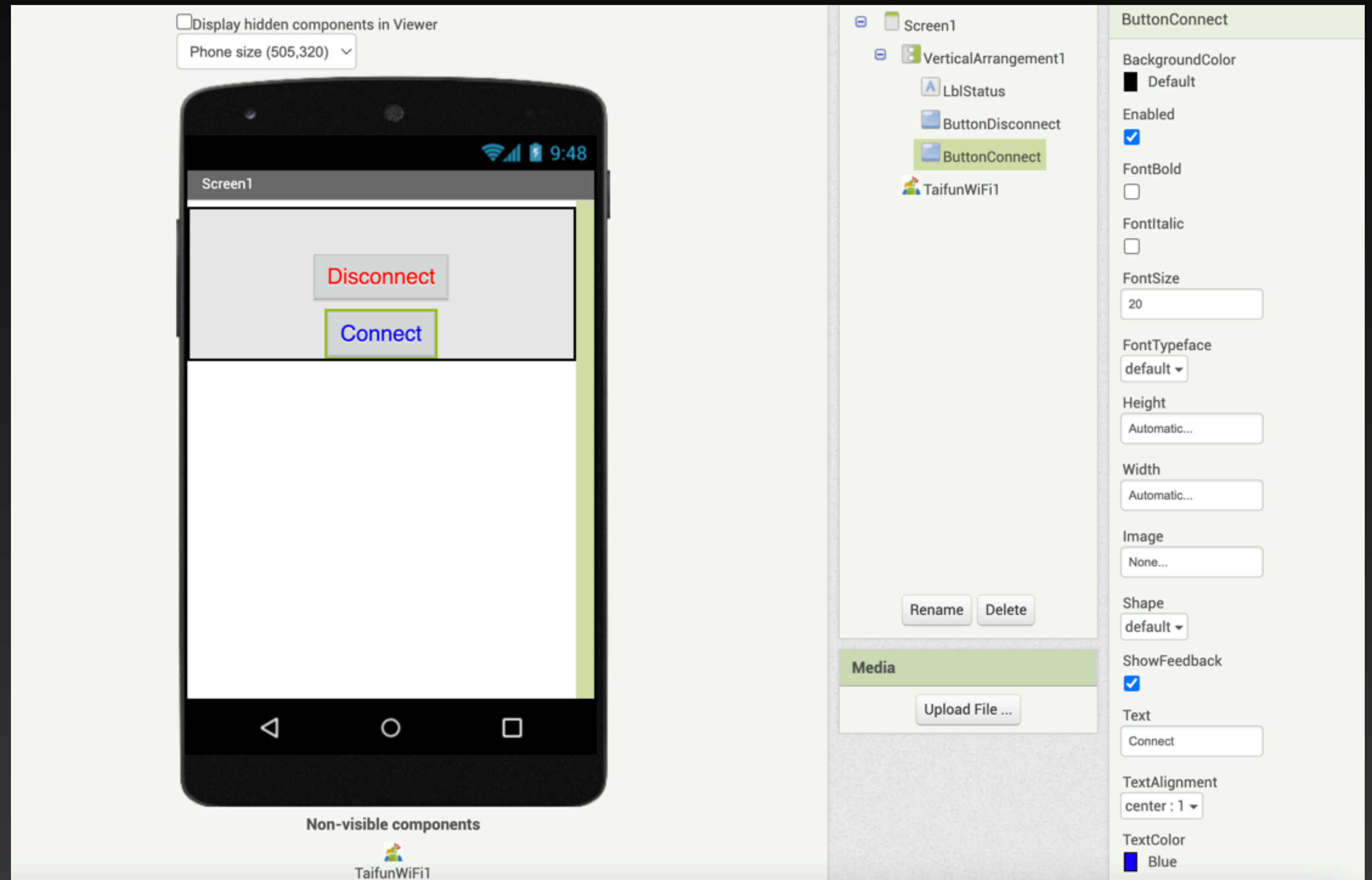
Height: Automatic

Width: Automatic

Text: Connect

TextColor: Blue

Visible: ticked



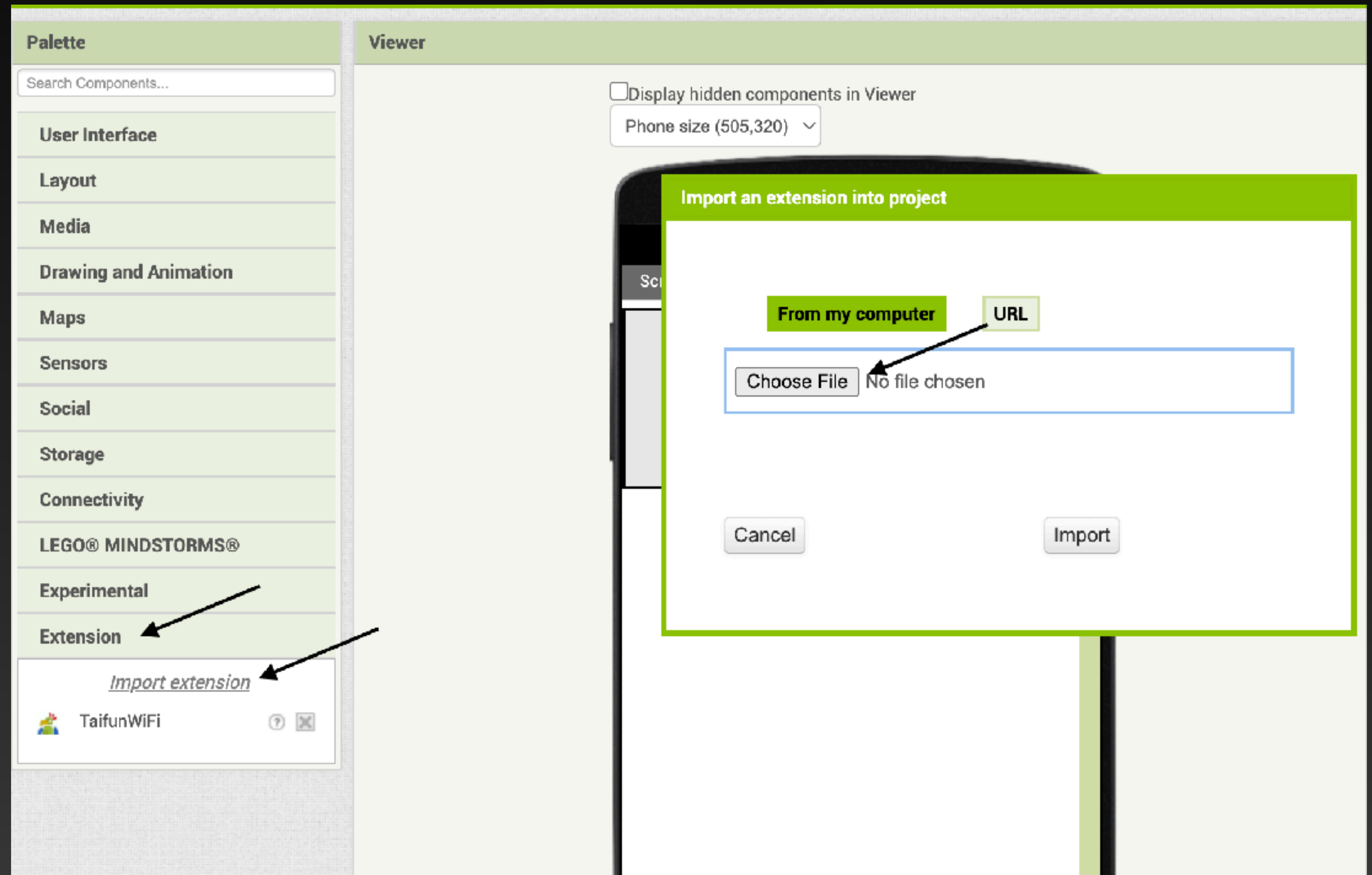
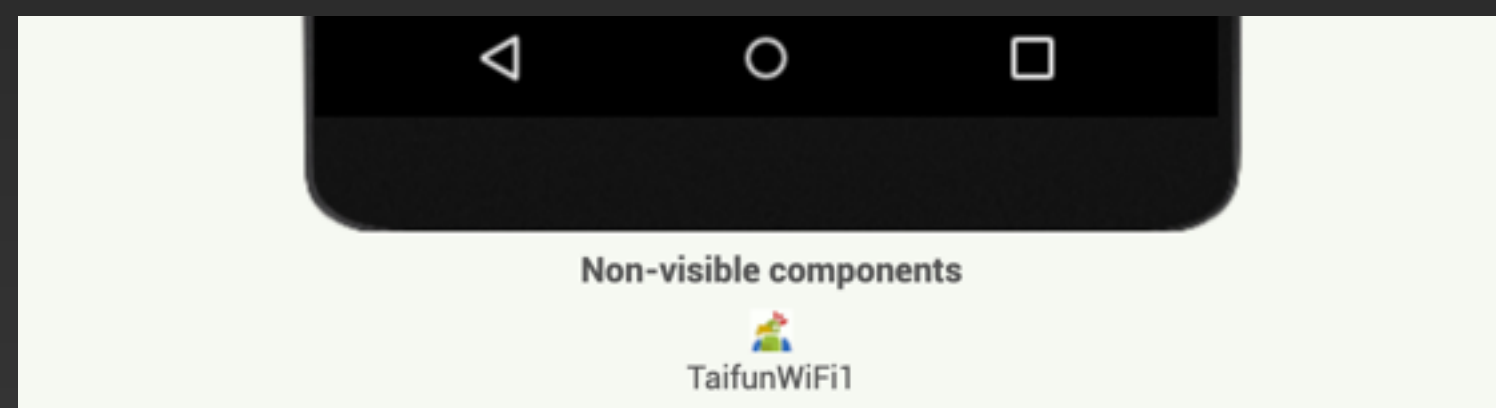
Getting and Displaying Local Wi-Fi Parameters

- Go to the following website:
<https://puravidaapps.com/wifi.php>
- Go to the end of the site and click Download TaifunWifi extension (aix file), download it to a folder

The screenshot shows the Pura Vida Apps website. The header includes navigation links: Pura Vida Apps, Snippets, Tutorials, Extensions, Links, Search, Privacy Policy, and Contact. The main heading is 'App Inventor Extensions'. Below this is a search bar titled 'Search owner of a phone number' powered by BeenVerified. The search bar has fields for 'Enter Area Code', a hyphen, and a phone number, with a 'SEARCH' button. To the left of the search bar is a 'WiFi Manager' extension description. To the right is a 'Download' section with a green background. The green section contains a message about the time spent on the site, a 'Donate' button with a '5 USD' amount, and a Bitcoin donation address: 1Jd8kXLHu2Vkuhi15TWHiQm4uE9AGPYxi8. Below the Bitcoin address is a QR code and the text 'Thank you! Taifun'. At the bottom of the page, there is a list of download links: 'Download TaifunWifi extension (aix file)', 'Download WiFi Test (aix file)', and 'Download Available SSIDs Test (aix file)'. An arrow points to the first link. The footer includes a Creative Commons license notice: 'This work by Pura Vida Apps is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License with attribution (name=Pura Vida Apps and link to the source site) required.'

Getting and Displaying Local Wi-Fi Parameters

- Click Extension tab on the left-hand side of your app inventor project
- Click Import extension and browse to the file you just downloaded
- You should see a new component called TaifunWiFi under Extension. Click and drop it on your Viewer. This is a hidden component and will only be displayed under the phone image as TaifunWiFi1



Getting and Displaying Local Wi-Fi Parameters

- Initialize two variables named ssid and password, Enter YOUR Wi-Fi SSID name and password into these blocks respectively.
- Click ButtonConnect and select when ButtonConnect.Click do. This block will be executed when button Connect is clicked.
- Click on TaifunWiFi1 and select call TaifunWiFi1.ConnectSSID and join blocks ssid and password to this block. This block will establish a connection to the local Wi-Fi router.

The image displays a Scratch script for managing a local Wi-Fi connection using the TaifunWiFi1 library. It includes two main event-driven blocks: one for connecting and one for disconnecting.

Initialization:

- `initialize global ssid to "1234"`
- `initialize global password to "5678"`

When ButtonConnect.Click do:

- `call TaifunWiFi1 .ConnectSSID` (with inputs: `ssid` from global variable, `password` from global variable)
- `set LblStatus . Text to` `join` ("Connected", "Local IP", `call TaifunWiFi1 .LocalIP`, "\n", "MAC Address", `call TaifunWiFi1 .MacAddress`, "\n", "Signal Strength", `call TaifunWiFi1 .SignalStrength`)
- `set LblStatus . TextColor to` blue

When ButtonDisconnect.Click do:

- `call TaifunWiFi1 .Disconnect`
- `set LblStatus . Text to` "Disconnected"
- `set LblStatus . TextColor to` red

Warning Panel:

0 0
Show Warnings

Getting and Displaying Local Wi-Fi Parameters

- Inset Join block and extend it to 8 connectors. Enter blocks as shown to display the IP address, MAC address, and the signal strength.
- Click Button Disconnect disconnects from the Wi-Fi

The image shows a Scratch script for connecting to a Wi-Fi network and displaying its parameters. The script is organized into two main sections: a 'when ButtonConnect.Click' section and a 'when ButtonDisconnect.Click' section.

Initialization:

- `initialize global ssid to "1234"`
- `initialize global password to "5678"`

Connect Section (when ButtonConnect.Click):

- `do` block containing:
 - `call TaifunWiFi1 .ConnectSSID` block with inputs `ssid` (connected to `get global ssid`) and `password` (connected to `get global password`).
 - `set LblStatus . Text to` block connected to a `join` block.
 - `set LblStatus . TextColor to` block connected to a blue color swatch.

Join Block:

- The `join` block has 8 connectors. The first connector is connected to the `Text` input of the `set LblStatus . Text to` block.
- The subsequent connectors are connected to:
 - `"Connected"` (text)
 - `"Local IP"` (text)
 - `call TaifunWiFi1 .LocalIP` (block)
 - `"\n"` (text)
 - `"MAC Address"` (text)
 - `call TaifunWiFi1 .MacAddress` (block)
 - `"\n"` (text)
 - `"Signal Strength"` (text)
 - `call TaifunWiFi1 .SignalStrength` (block)

Disconnect Section (when ButtonDisconnect.Click):

- `do` block containing:
 - `call TaifunWiFi1 .Disconnect` block.
 - `set LblStatus . Text to` block connected to `"Disconnected"` (text).
 - `set LblStatus . TextColor to` block connected to a red color swatch.

Warning Panel:

At the bottom left, there is a 'Show Warnings' panel with two warning icons (a yellow triangle and a red X) and a count of 0 for each.

Getting and Displaying Local Wi-Fi Parameters

- Connect your program through AI companion, and click on connect in your app.
- What did you see on your mobile screen?

Raspberry Pi Wi-Fi based projects

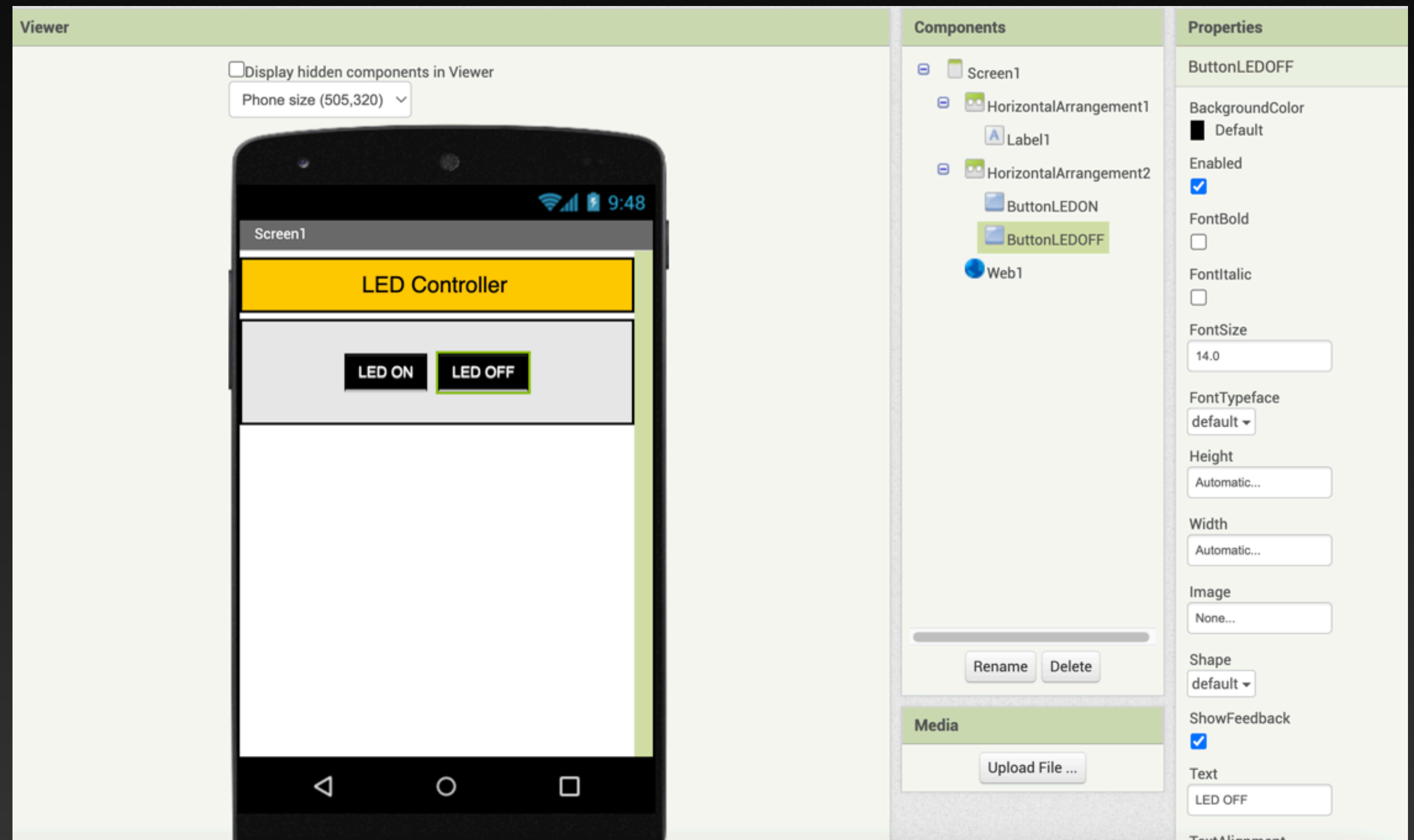
Project 2 - Web server to control LED

Description: In this project, an LED is connected to the Raspberry Pi and is controlled from an Android mobile phone using a web server application.

Use the same circuit program from previous project to setup LED on raspberry Pi.

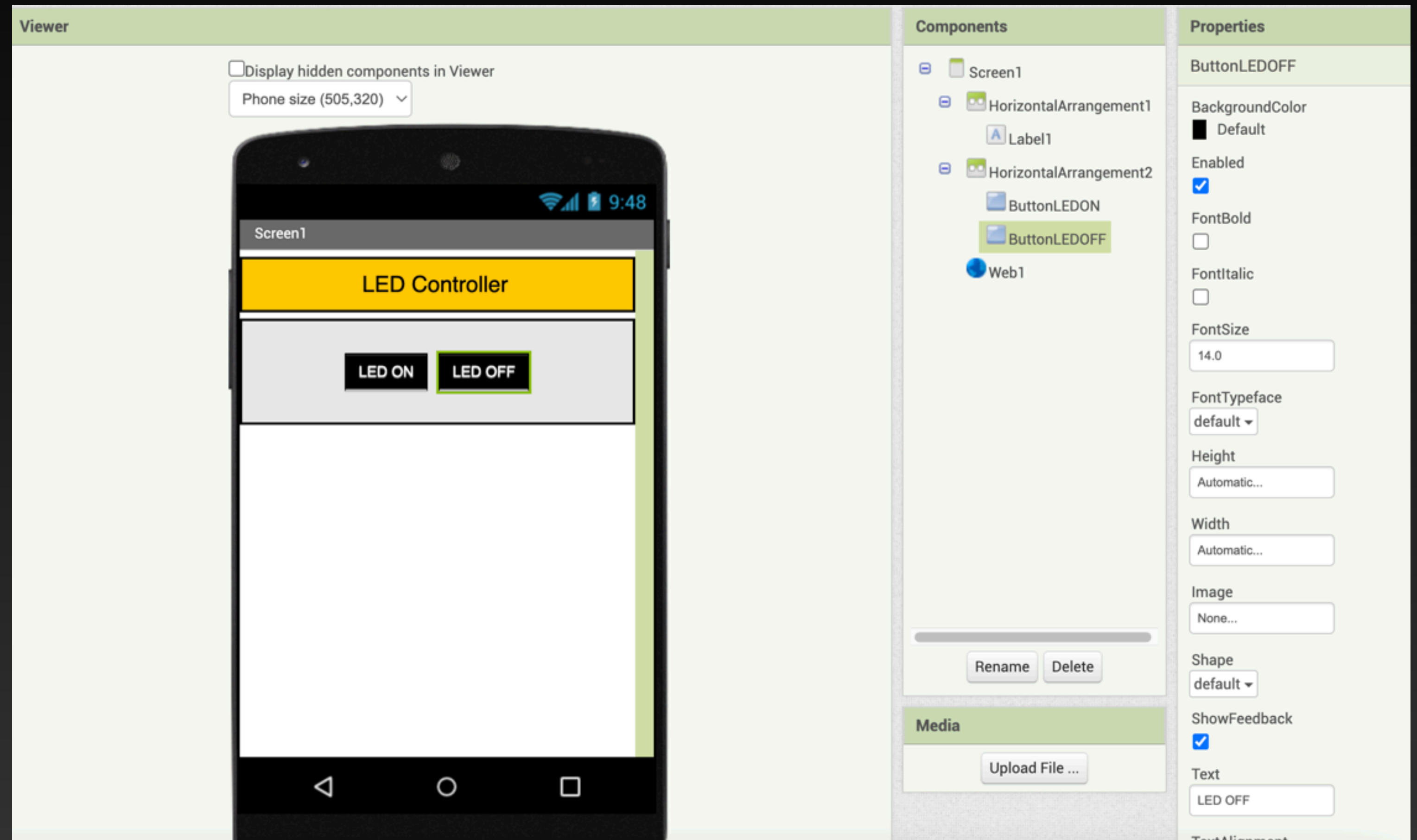
Web Server to Control LED

- Create a new project and name it as WEB_LED
- Insert a HorizontalArrangement and insert a Label on it with its Text set to LED CONTROLLER
- Insert another HorizontalArrangement



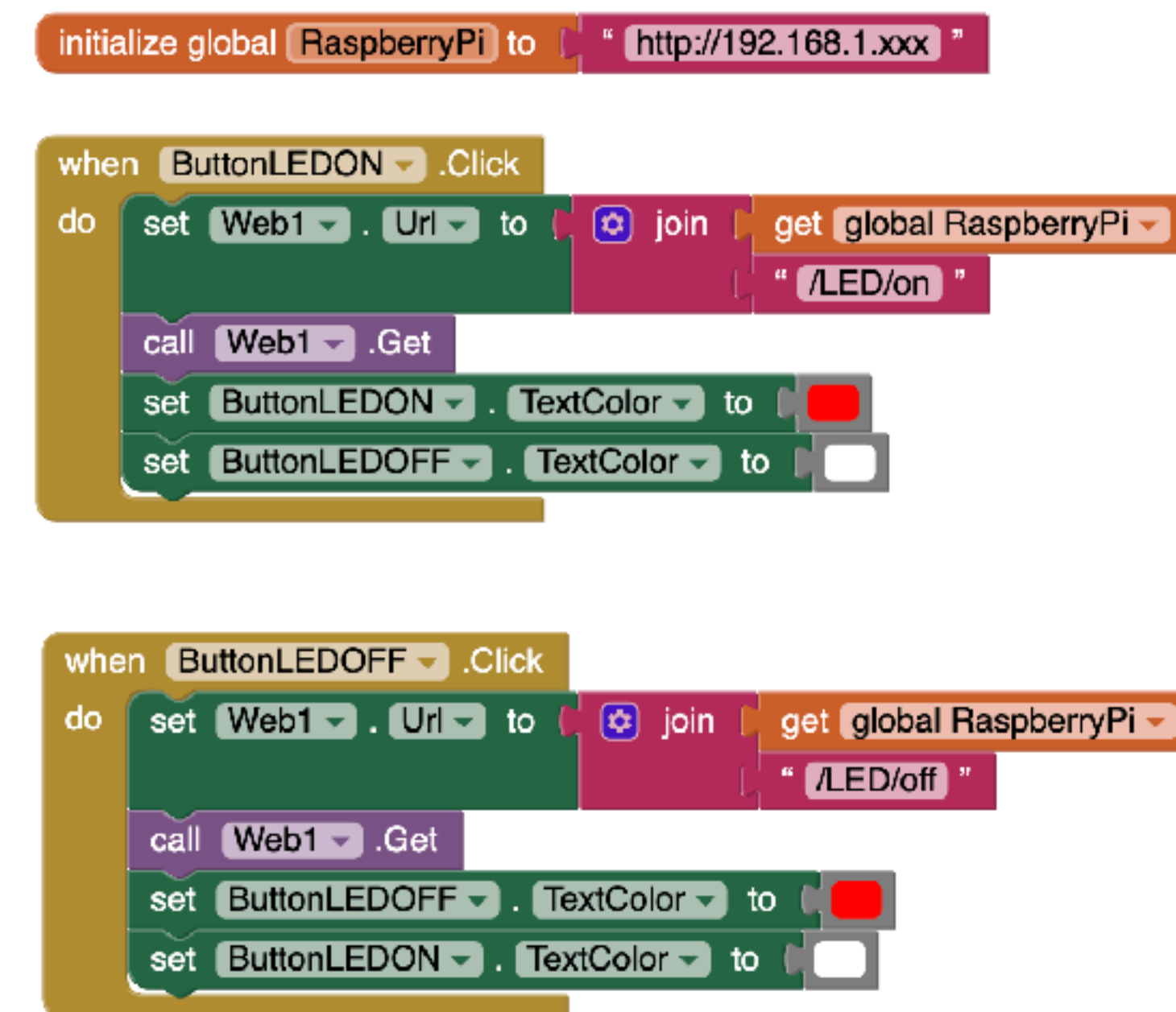
Web Server to Control LED

- Insert two buttons on the HorizontalArrangement with the names ButtonON and ButtonOFF, with their texts set to LED ON and LED OFF respectively
- Click the Connectivity tab and insert a Web component on the Viewer. This is a hidden component.
- Initialize a variable called RaspberryPi and set it to the IP address of your RaspberryPi



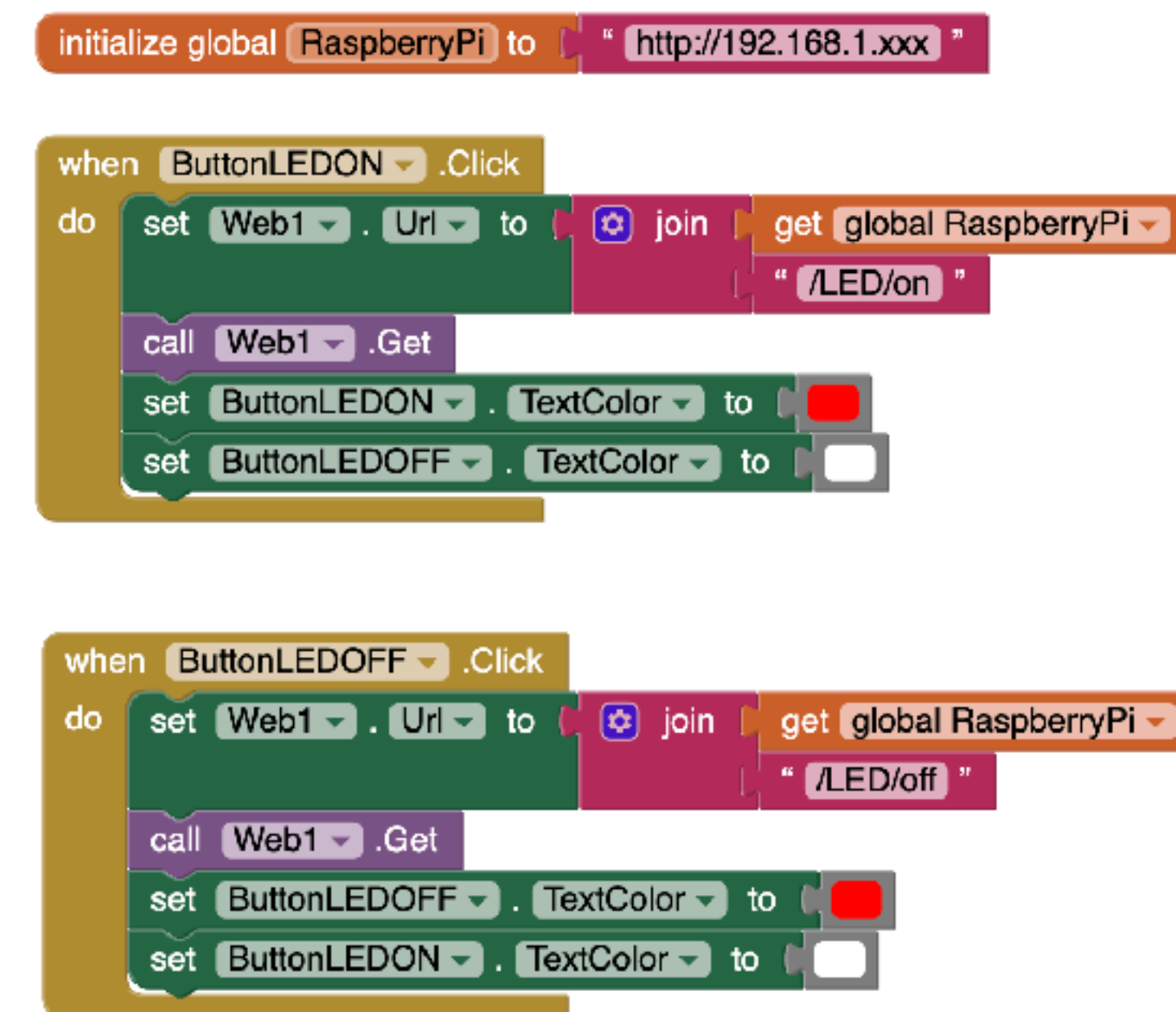
Web Server to Control LED

- Initialize a variable called RaspberryPi and set it to the IP address of your RaspberryPi
- Click ButtonON and select when ButtonON.Click do. This block will be executed when button LED ON is clicked.
- Click on Web1 and select set Web1.Url to.
- Insert a Join block and set the URL to the IP address of your URL and add string /LED/on to this block.



Web Server to Control LED

- Click on Web1 and select call Web1.Get. In this project, the URL is set to <http://192.168.1.xxx/LED/on>
- Set the color of ButtonON to red , and the color of ButtonOFF to white
- Repeat for the ButtonOFF as shown in the second group of blocks. But this time set the URL to <http://192.168.1.xxx/LED/off>

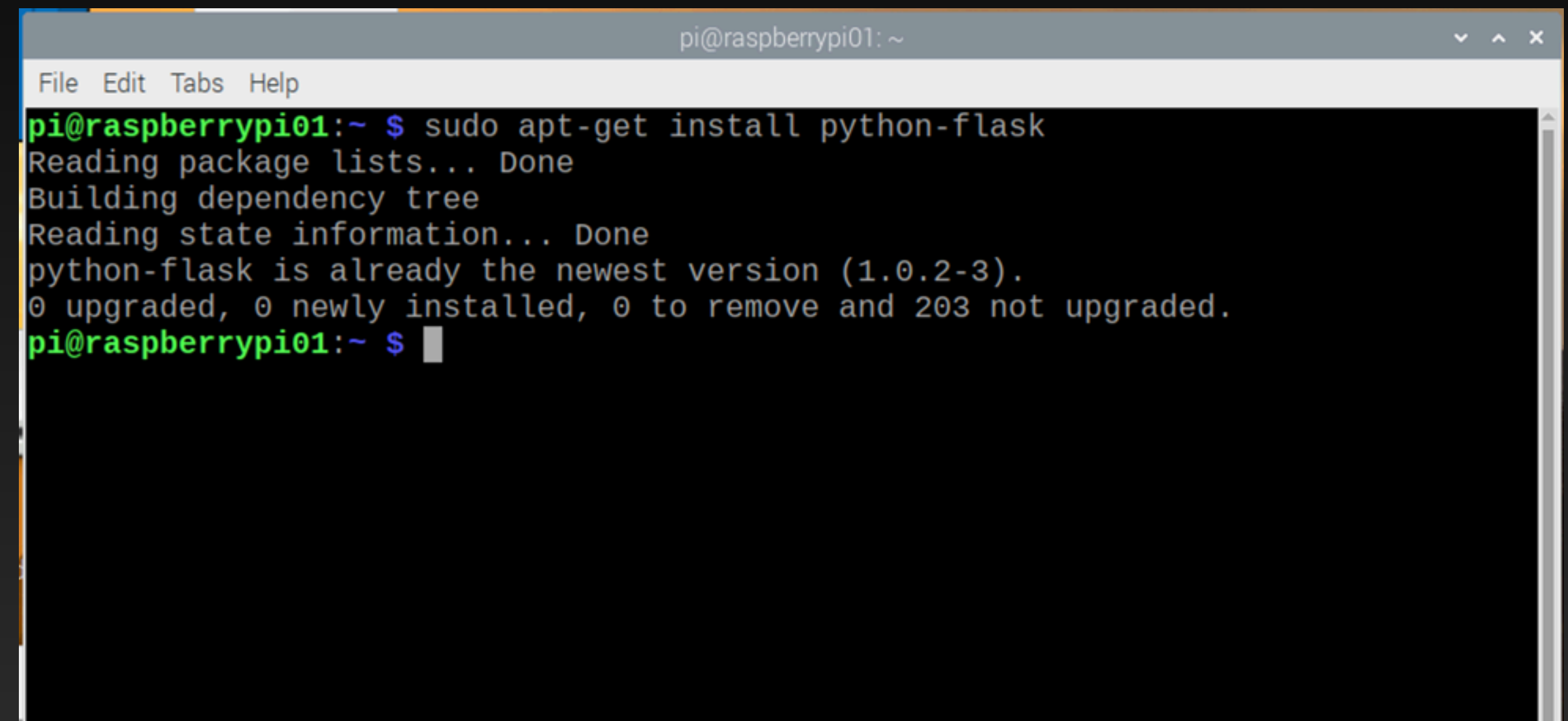


Web Server to Control LED

Python program: Python program is based on Flask which is a simple micro-framework written in Python. It is free of charge and can be used to create a web server on Raspberry Pi to control GPIO ports over the Internet.

Flask should already be available in Python on your Raspberry Pi, but if it isn't, it can be installed using command below:

```
sudo apt-get install python3-flask
```

A terminal window titled 'pi@raspberrypi01: ~' with a menu bar containing 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows the command 'sudo apt-get install python-flask' being executed. The output indicates that the package is already the newest version (1.0.2-3) and that no packages need to be upgraded, installed, or removed. The prompt returns to 'pi@raspberrypi01:~ \$' with a cursor.

```
pi@raspberrypi01:~ $ sudo apt-get install python-flask
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-flask is already the newest version (1.0.2-3).
0 upgraded, 0 newly installed, 0 to remove and 203 not upgraded.
pi@raspberrypi01:~ $
```

Web Server to Control LED

Python program:

Import flask and gpiozero library

```
from flask import Flask,render_template
```

```
from gpiozero import LED
```

```
app=Flask(__name__)
```

```
13 #-----
14
15 from flask import Flask, render_template # import flask web server library
16 from gpiozero import LED
17
18 app = Flask(__name__)
19
20
21 led = LED(2)
22
23 #
24 #start the main program loop, read command from android app
25 #through local web server, decode the msg and control LED
26 #
27
```

Web Server to Control LED

Python program:

Set LED to 2 which is the GPIO 2 port it is connected to. This port is configured as an output and the LED is turned OFF at the beginning of the program.

Led = LED(2)

```
13 #-----
14
15 from flask import Flask, render_template # import flask web server library
16 from gpiozero import LED
17
18 app = Flask(__name__)
19
20
21 led = LED(2)
22
23 #
24 #start the main program loop, read command from android app
25 #through local web server, decode the msg and control LED
26 #
27
```


Web Server to Control LED

Python program:

An `app.route` is created with parameters `device` and `action`. For every actuator, we must have an action. If the action is on, the LED is turned ON, otherwise, if the action is off, the LED is turned OFF.

```
@app.route("/<device>/<action>")
```

```
def action(device, action):
```

```
    actuator = LED
```

```
    if action == "on":
```

```
        led.on()
```

```
    if action == "off":
```

```
        led.off()
```

```
    return ""
```

```
22
23 #
24 #start the main program loop, read command from android app
25 #through local web server, decode the msg and control LED
26 #
27
28 @app.route("/<device>/<action>")
29
30 def action(device, action):
31     actuator = LED
32     if action == "on":
33         led.on() # turn on LED
34     if action == "off":
35         led.off() # turn off LED
36     return ""
37
```

Web Server to Control LED

Python program:

Notice that the LED can be turned ON by entering the URL: <http://192.168.1.xxx/LED/on> to our web browser. Similarly, the LED can be turned OFF by the URL: <http://192.168.1.xxx/LED/off>

```
if __name__ == '__main__':  
    app.run(debug=True, port=80,  
            host='0.0.0.0', use_reloader=False)
```

```
22  
23 #  
24 #start the main program loop, read command from android app  
25 #through local web server, decode the msg and control LED  
26 #  
27  
28 @app.route("/<device>/<action>")  
29  
30 def action(device, action):  
31     actuator = LED  
32     if action == "on":  
33         led.on() # turn on LED  
34     if action == "off":  
35         led.off() # turn off LED  
36     return ""  
37  
38 if __name__ == '__main__':  
39     app.run(debug=True, port=80, host='0.0.0.0', use_reloader=False)  
40 |
```

Web Server to Control LED

Python program:

To test your program, go to the folder where your project file is stored, start python program on your Raspberry Pi from command line as:

```
sudo python3  
204_LEDWebServergpiozero.py
```

```
pi@pi5:~ $ cd PythonCode/  
pi@pi5:~/PythonCode $ sudo python3 204_LEDWebServergpiozero.py  
* Serving Flask app '204_LEDWebServergpiozero'  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on all addresses (0.0.0.0)  
* Running on http://127.0.0.1:80  
* Running on http://192.168.1.219:80  
Press CTRL+C to quit  
192.168.1.219 - - [26/Apr/2025 18:40:15] "GET /LED/on HTTP/1.1" 200 -  
192.168.1.219 - - [26/Apr/2025 18:40:15] "GET /favicon.ico HTTP/1.1" 404 -  
192.168.1.219 - - [26/Apr/2025 18:40:28] "GET /LED/off HTTP/1.1" 200 -  
192.168.1.225 - - [26/Apr/2025 18:41:12] "GET /LED/on HTTP/1.1" 200 -  
192.168.1.225 - - [26/Apr/2025 18:41:12] "GET /favicon.ico HTTP/1.1" 404 -  
192.168.1.225 - - [26/Apr/2025 18:41:18] "GET /LED/on HTTP/1.1" 200 -  
192.168.1.225 - - [26/Apr/2025 18:41:20] "GET /LED/off HTTP/1.1" 200 -
```

Python Program - LED control through bluetooth

Home project1 - Develop app to control LED through speech commands

Home project2 - Develop app to control 3 LEDs through WIFI