

## CG2271: Real-Time Operating Systems

### App-Controlled ESP32 WebServer

We will design a simple ESP32-based WebServer that will control an LED through an Android App.

#### Schematic Design

Component List:

- ESP32 Board
- LED's (any colour) (x1)
- Resistors (180 ohms) (x1)
- Breadboard

#### Step 1: ESP32 Code

Copy and Paste the code from the file ESP32\_WebServer\_ai2.c

Change the values of the **SSID** and **Password** to those of your own hotspot.

Compile and upload the code onto the board.

Open the Serial Monitor and set it to 115200 bps.

Press the "EN" button on the board and you should observe the IP address of the ESP32 being displayed.

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5856
entry 0x400806a8
Connecting to AndroidAP8729
.....
WiFi connected.
IP address:
192.168.43.221
```

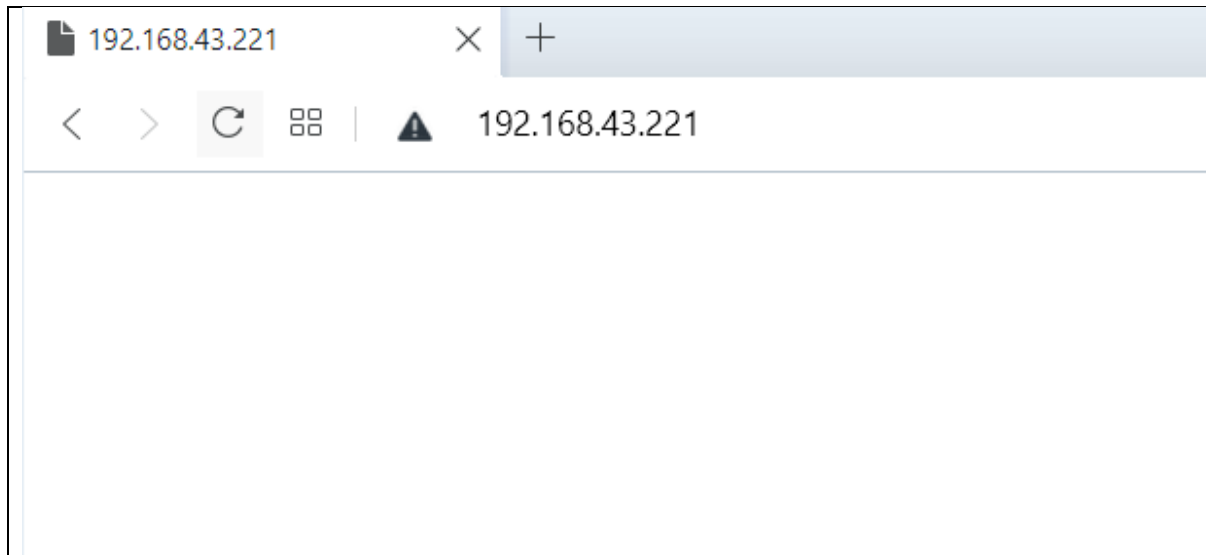
## Step 2: Web Interface Control

We can now test the ESP32 with a Webserver using a standard web browser (Firefox/Opera/etc.) Chrome browsers have been known to have some intermittent issues.

Open up your browser and enter the IP address of your ESP32 in the address bar.

Remember to use your OWN IP for the remaining steps.

For my board it is <http://192.168.43.221/>



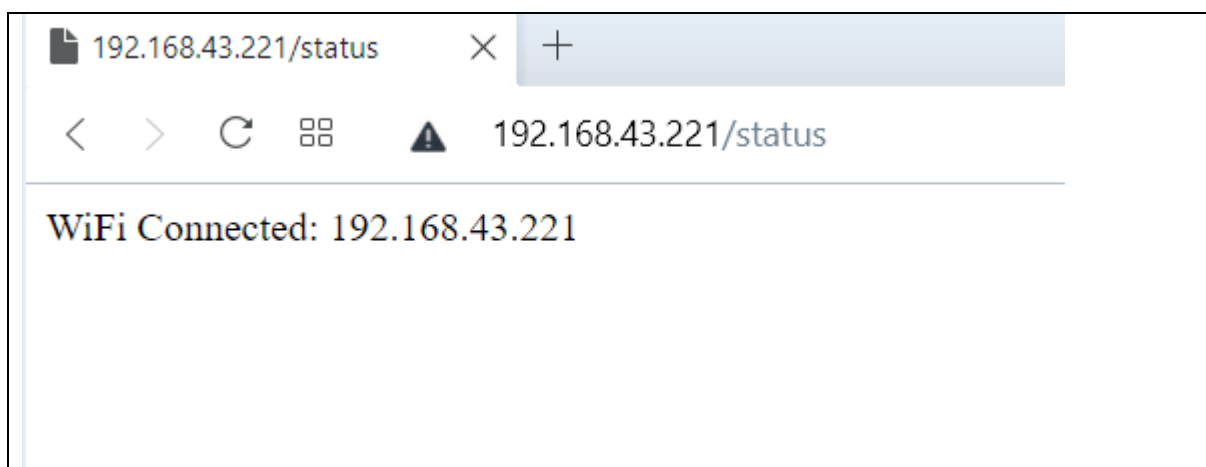
You should expect to see a blank empty webpage.

- Status Check

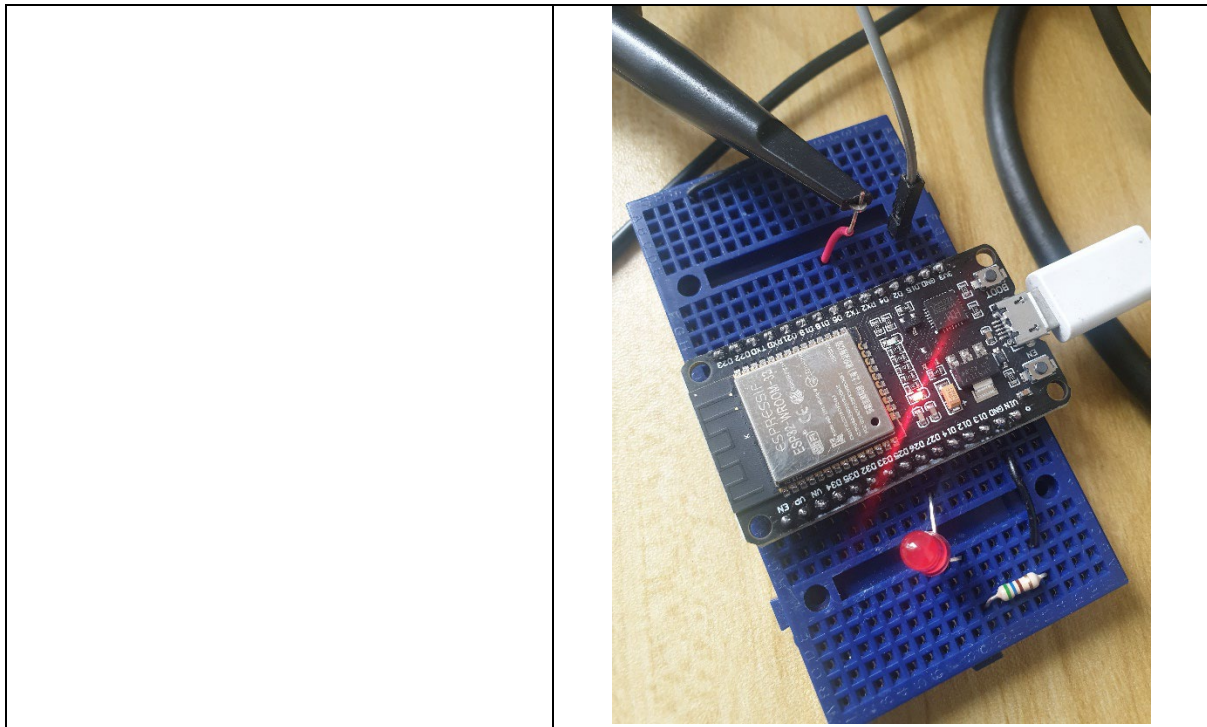
Add “/status” to your Static IP so that it looks like this

<http://192.168.43.221/status>

You should see a response like this.



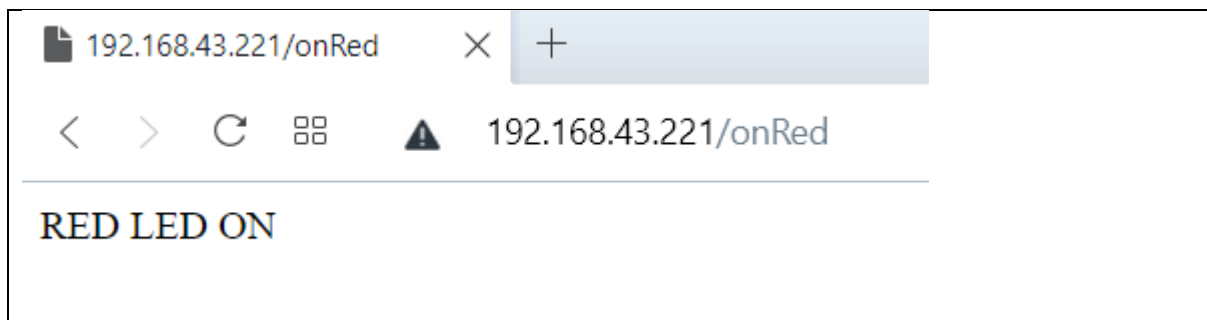
We can now connect up the HW as shown.



- Switch ON Red LED

Add “/onRed” to your Static IP so that it looks like this

<http://192.168.43.221/onRed>

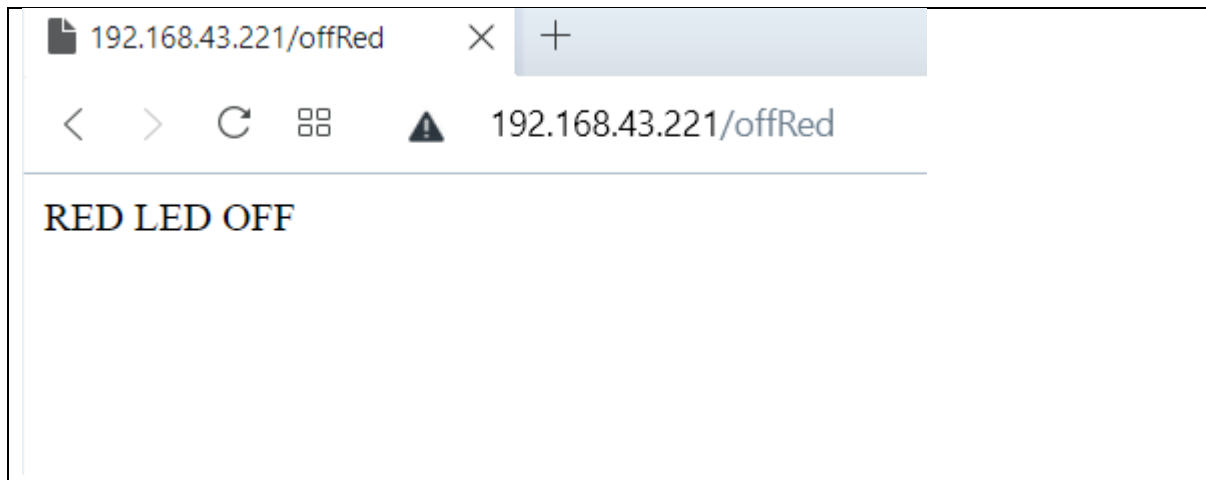


You should see a similar text on your webpage while observing that the Red LED connected to GPIO26 is ON. If you do not see the LED switched ON, check your HW connection.

- Switch Off Red LED

Add “/offRed” to your Static IP so that it looks like this

<http://192.168.43.221/offRed>



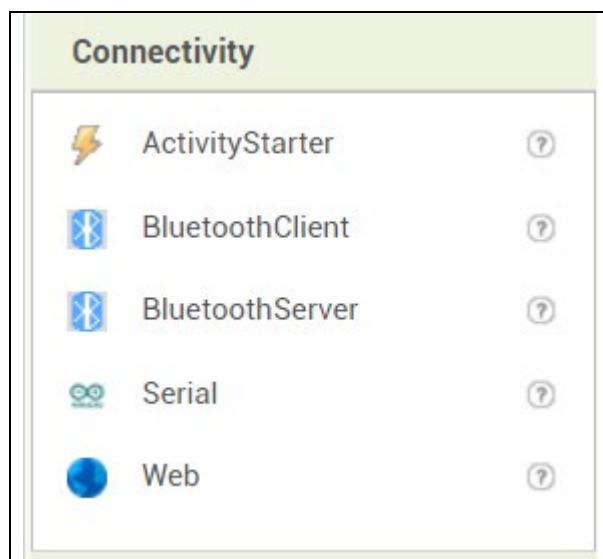
You should see a similar text on your webpage while observing that the Red LED connected to GPIO26 is OFF.

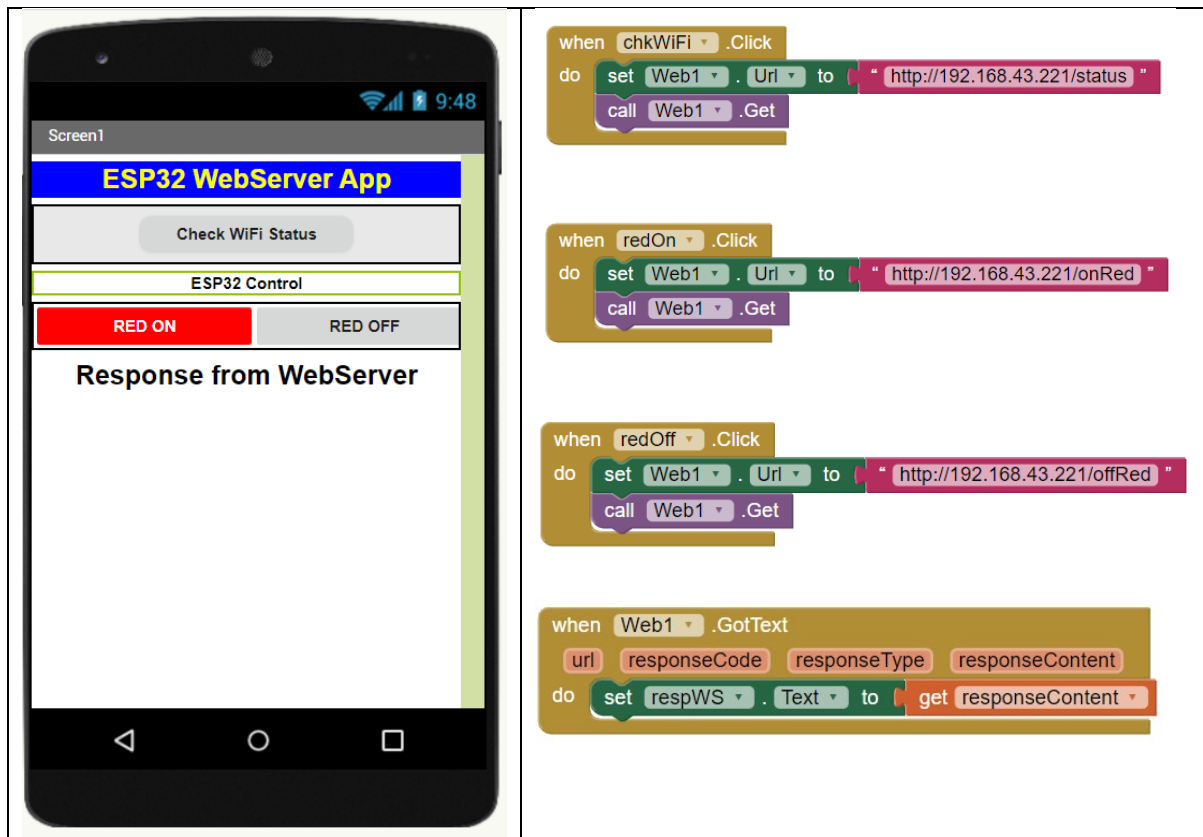
- **Designing the User-Interface**

From the earlier example, you can see that we can send commands to the ESP32 through a Web Interface designed using HTML code. That is an option you can explore on your own. In this section, you are going to learn how to send commands through an App designed using the MIT App Inventor. It is an intuitive block-based App development platform. If you are new to it, I would strongly recommend that you view the videos from <http://www.docr.sg/apps.html> to get a good grasp of the basic concepts before proceeding.

The user-interface shown has some buttons to send similar web URL's to the ESP32 webserver. This allows the ESP32 to perform the necessary action and send us the appropriate response.

- You must add "Web" from the Connectivity Tab to your App. It will appear as a non-visible component in your design.





With the above code, you should be able to control the LED connected to Pin26 of the ESP32.