

**Course** : Diploma in Electronics & Computer Engineering (EGDF20)

**Module**  : Connected System Design Project (EGE205)

**Laboratory No**. : Lab 6a

**Laboratory Title** : Networking: Setting up a connected system using Multiple BeagleBone Black

Wireless (BBBW) Boards

**Objective** : To use multiple BeagleBone Black Wireless (BBBW) Boards to set up a connected

system.

**Hardware Boards** : BBBW Board x4

**Contents**

1. Setting Up a Connected System using Multiple BeagleBone Black Wireless (BBBW) Boards
   1. Understanding of Connected Systems
   2. Setting up a Python Web Server on a PC
   3. Setting up Multiple Web Clients in Multiple BBBW Boards
   4. Controlling and Monitoring Multiple BBBW Boards via Web Server on a PC

|  |
| --- |
| **Note**:  This lab requires a team to work together to see the ultimate result! Team leader to complete section 1.2 while the rest of the team members to complete section 1.3. For section 1.3, each team member is required to choose only 1 of the BBBW code to be implemented in his / her BBBW board. Upon completion of section 1.2 and 1.3 respectively. The team can proceed to section 1.4 to see the ultimate result of a connected system. Good Luck! |

# **Setting Up a Connected System using Multiple BeagleBone Black Wireless (BBBW) Board**

## Understanding of Connected System

**Connected System** can refer to a collection of interrelated systems consisting of IT devices, sensors, and actuators that can seamlessly interact with each other. By incorporating the functions of sensing, actuation, and control, these systems can **measure**, **describe**, and **analyze** data to **make decisions** and **perform actions** both with and without human intervention.

**How does a typical connected system work?**

A typical connected system ecosystem consists of a central web server and sensor devices that use embedded systems, such as processors, sensors, and communication hardware, to collect, send and act on data they acquire from their environments. They are all inter-connected and usually have a unique ID such as IP address.

Collecting the data is done by transmitting it from the sensor devices to a central web server where data are to be processed and analyzed. Sometimes, these sensor devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices such as, to set them up, give them instructions or access the data.

The connectivity such as Wi-Fi or Bluetooth, networking, and communication protocols such as HTTP, TCP/IP used in a connected system largely depend on the specific applications deployed.

A typical setup of a connected system using BBBW boards is shown in the Figure below.

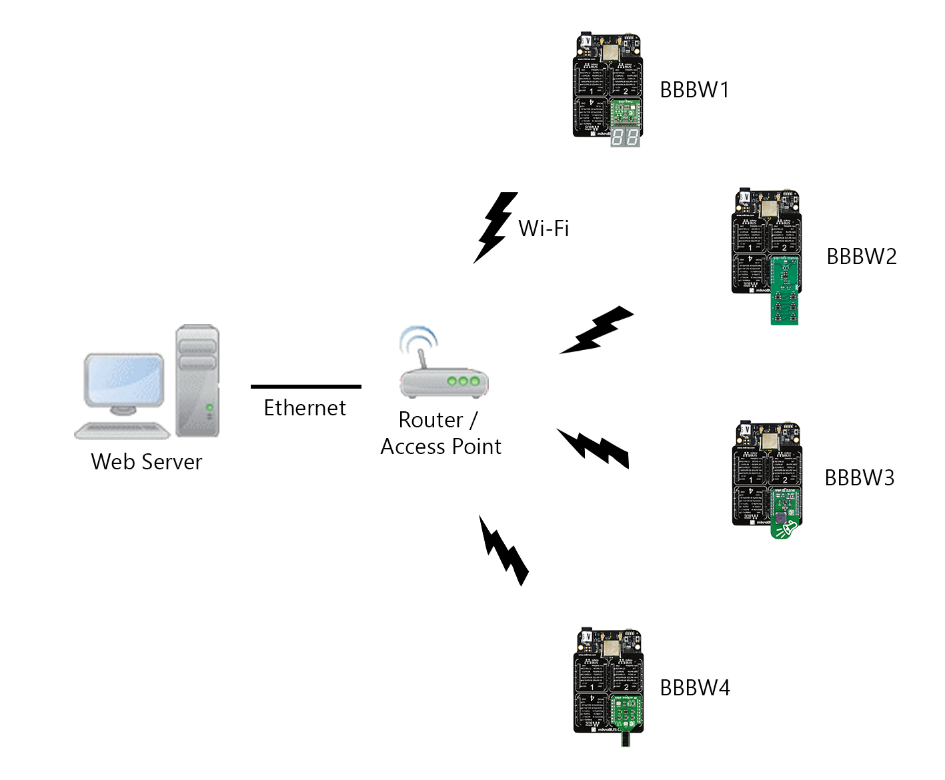


Figure 1.1a: Connected System setup using BBBW Boards

## Setting up a Python Web Server on a PC

**Setting Up the Web Server on the PC**

1. **Create** the following folders according to the folder structure below in the C: drive.

* **MyFirstPythonProject**
  + **MyWebServer**
    - **templates**
    - **static**
      * **css**

1. In the PC, **type** in “**cmd**” in the window search function to search for the Command Prompt softwareand launch it as shown in the Figure below.

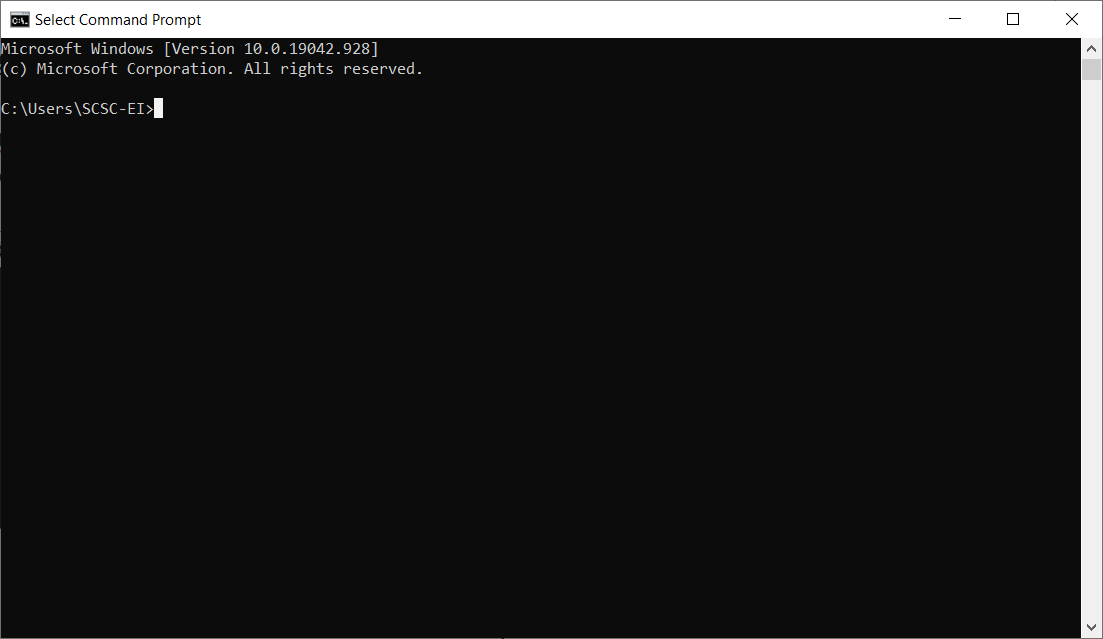


Figure 1.2a: Command Prompt

1. **Skip** step 4 if **flask-socketio** python libraryhas been installed earlier.
2. **Type** in“**pip install flask-socketio**” and **hit** the “Enter” key to download and install the flask-socketio python library as shown in the Figure below.

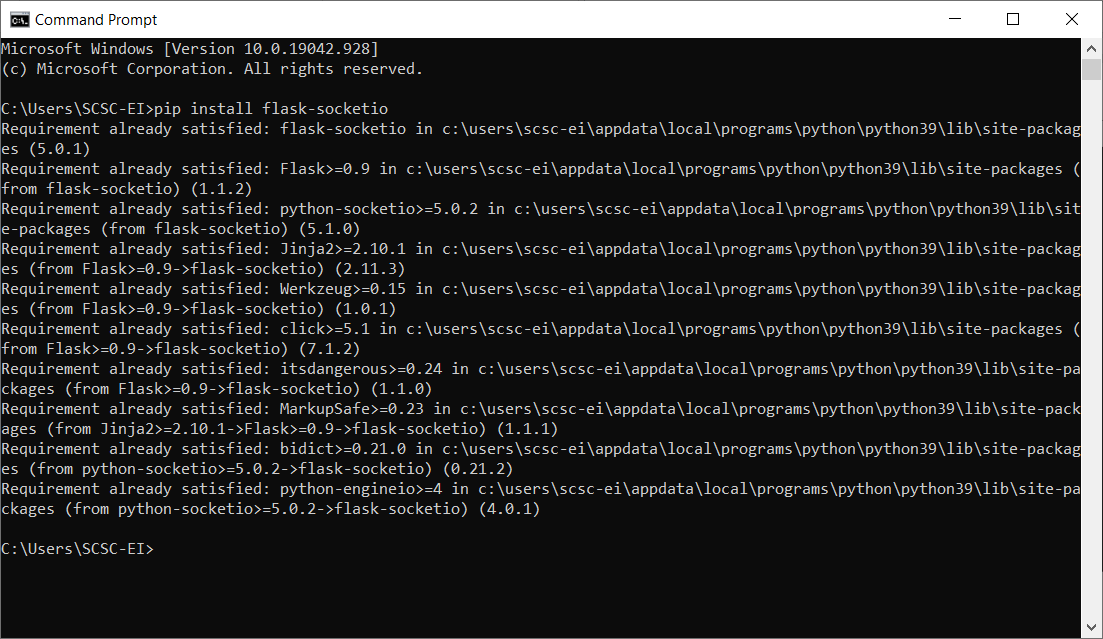


Figure 1.2b: Command Prompt

1. **Type** in“**ipconfig**” and **hit** the “Enter” key to retrieve the IP address of the wireless connection assigned to the PC. In this example, the IP address of 192.168.1.80 is assigned to the PC as shown in the Figure below.

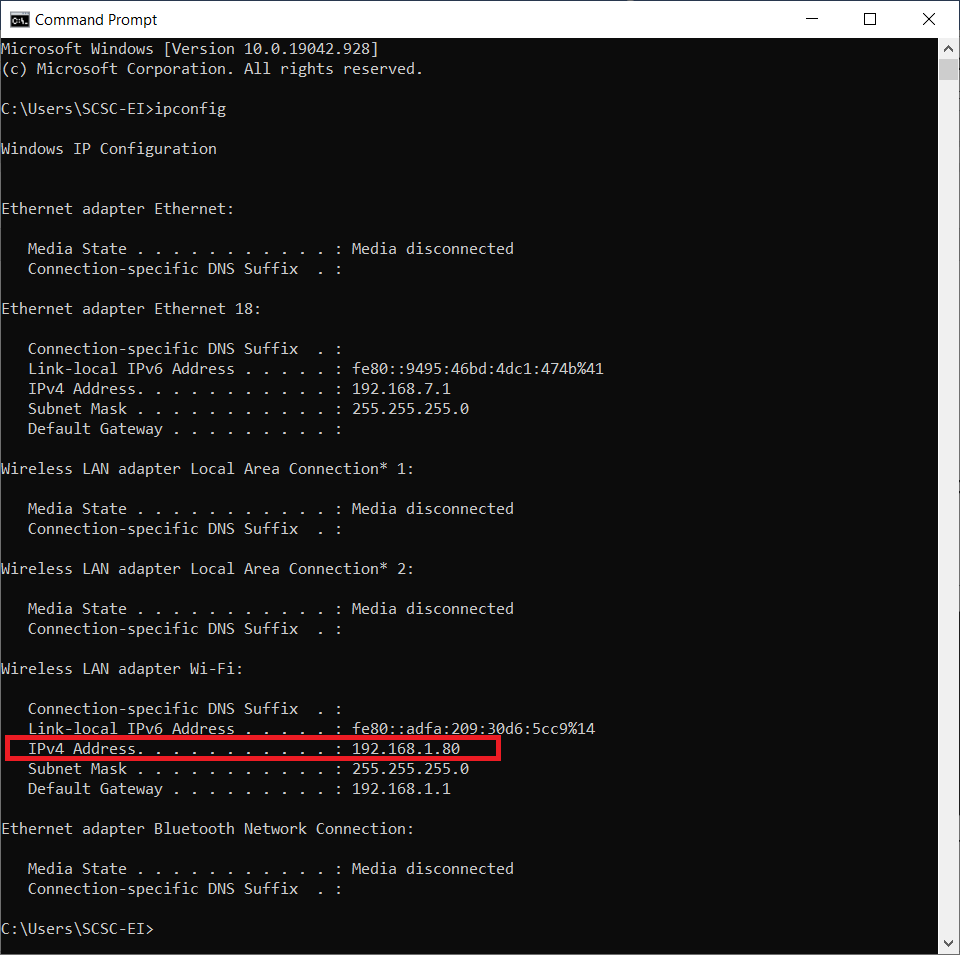


Figure 1.2c: IP Address Assigned to the PC

1. **Type** in “**IDLE**” in the window search function to search for the Python IDLE Shell softwareand launch it as shown in the Figure below.

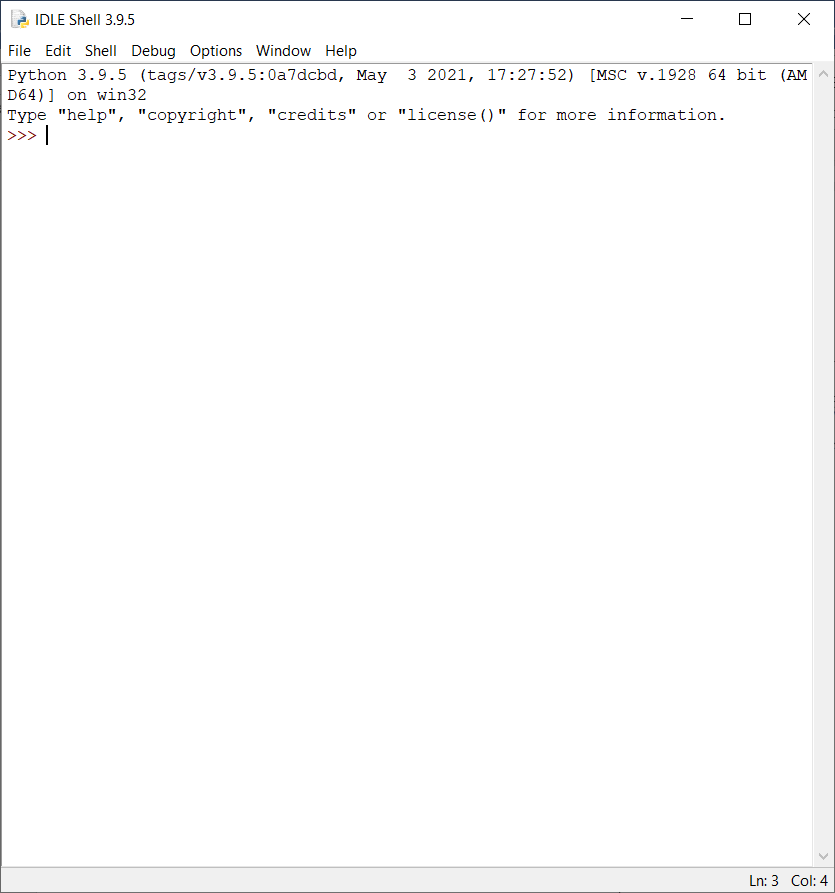


Figure 1.2d: Python IDLE Shell

1. **Click** on the “File” option from the menu toolbar located at the top of the IDLE Shell software and **select** “New File” from the drop-down menu option as shown in the Figure below.

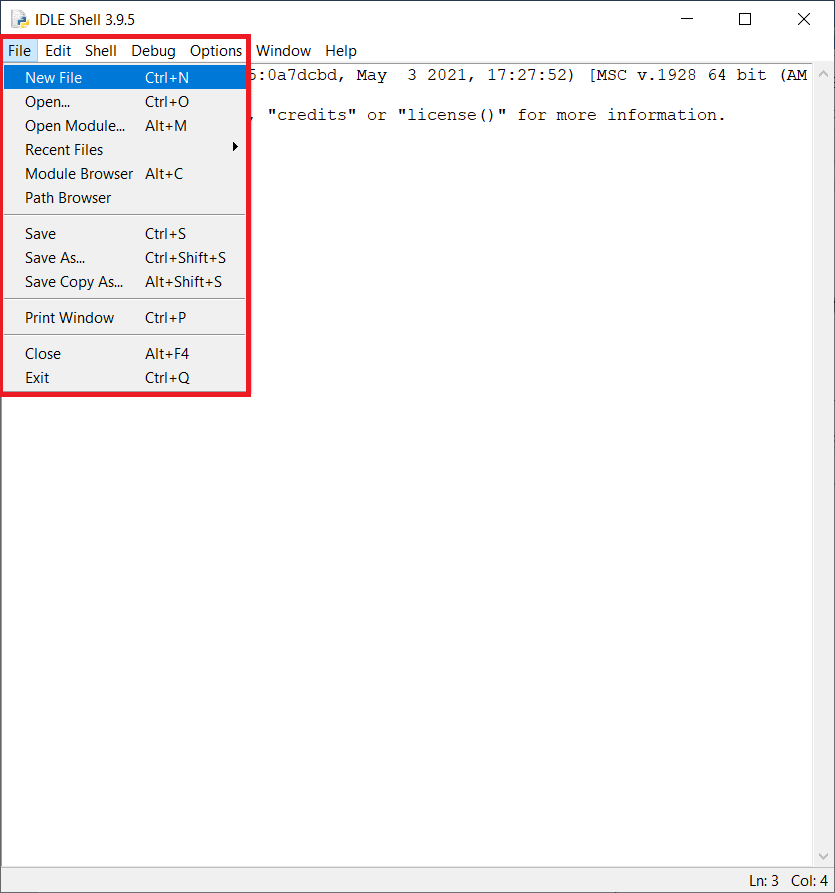


Figure 1.2e: Create new python file

1. **Enter** the following code into the newly created file. **Modify** the IP address of “**192.168.X.X**” according to the IP address you have retrieved from the PC earlier.

|  |
| --- |
| from flask import Flask  from flask import render\_template  from flask\_socketio import SocketIO  from flask\_socketio import emit    app = Flask(\_\_name\_\_)  socketio = SocketIO(app)  @app.route('/')  def index():  return render\_template('index.html')  @socketio.event  def ControlUSR0Led(RxData):  if RxData == 'on':  socketio.emit('ControlUSR0Led', RxData)  if RxData == 'off':  socketio.emit('ControlUSR0Led', RxData)  @socketio.event  def BBBW1Event(RxData):  socketio.emit('Web\_BBBW1Event', RxData)  print('Receive Data from BBBW1')  @socketio.event  def BBBW2Event(RxData):  socketio.emit('Web\_BBBW2Event', RxData)  print('Receive Data from BBBW2')  @socketio.event  def BBBW3Event(RxData):  socketio.emit('Web\_BBBW3Event', RxData)  print('Receive Data from BBBW3')  @socketio.event  def BBBW4Event(RxData):  socketio.emit('Web\_BBBW4Event', RxData)  print('Receive Data from BBBW4')  if \_\_name\_\_ == '\_\_main\_\_':  **app.run(host='192.168.X.X')** |

1. **Click** on the “File” option from the menu toolbar located at the top of the IDLE Shell software and **select** “Save As…” from the drop-down menu option as shown in the Figure below. **Navigate** to the **MyWebServer** folder created earlier and **save** the file using the name of “**WebServer.py**”.

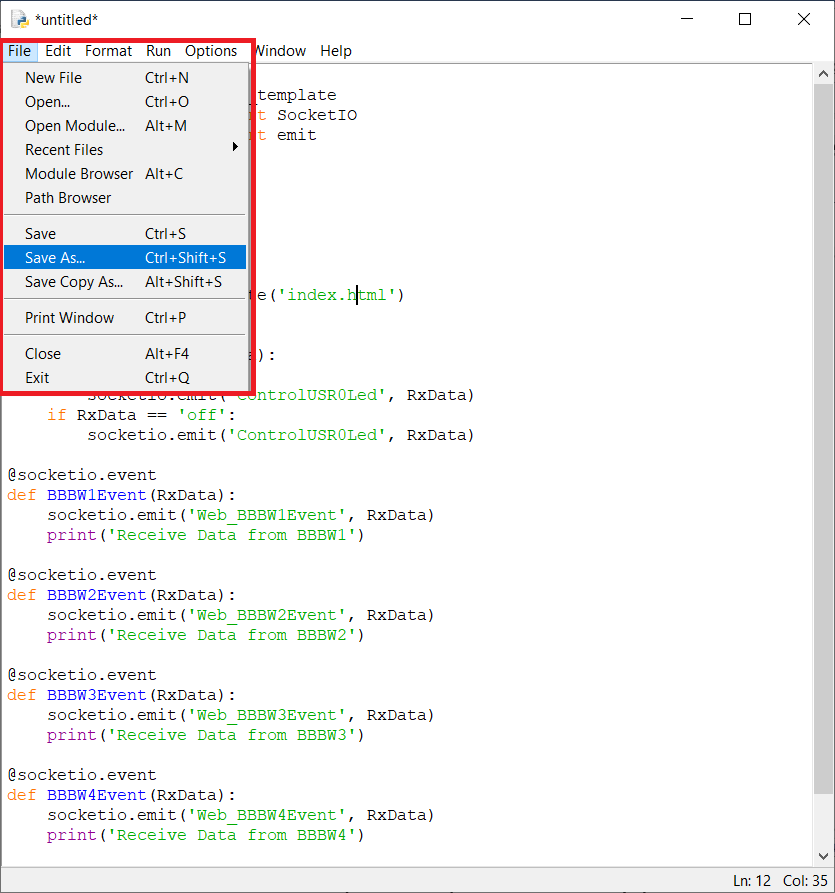


Figure 1.2f: Save the python file

1. **Click** on the “Run” option from the menu toolbar located at the top of the IDLE Shell software and **select** “Run Module” from the drop-down menu option to execute the “**WebServer.py**” file as shown in the Figure below.

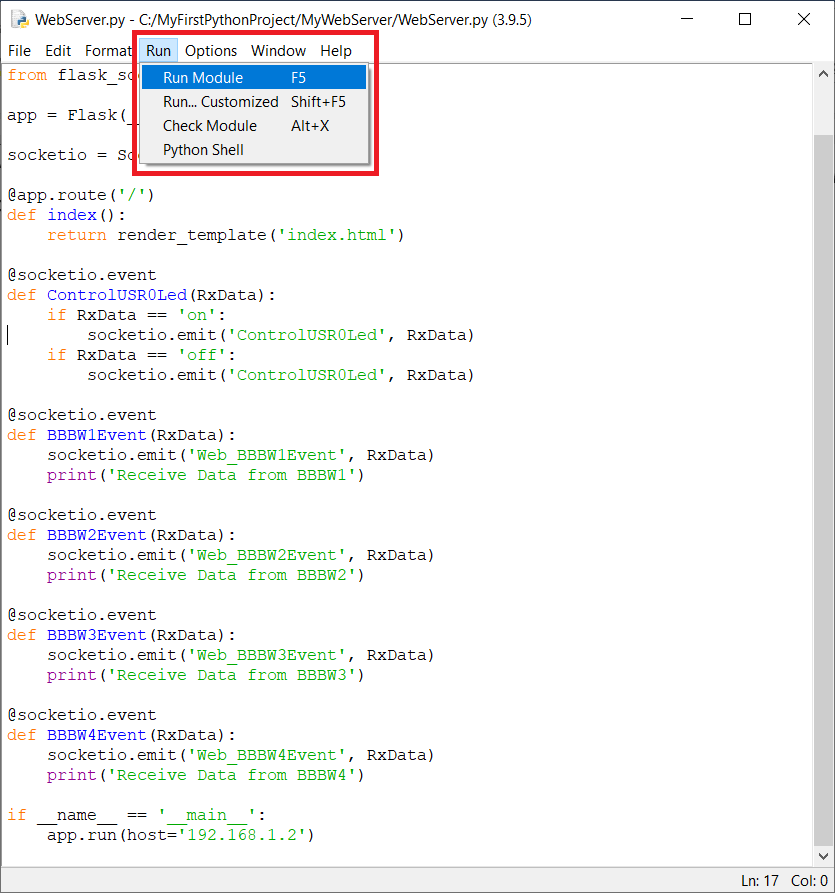


Figure 1.2g: Executing the python file

1. A window pops up to indicate that the web server is up and running as shown in the Figure below.

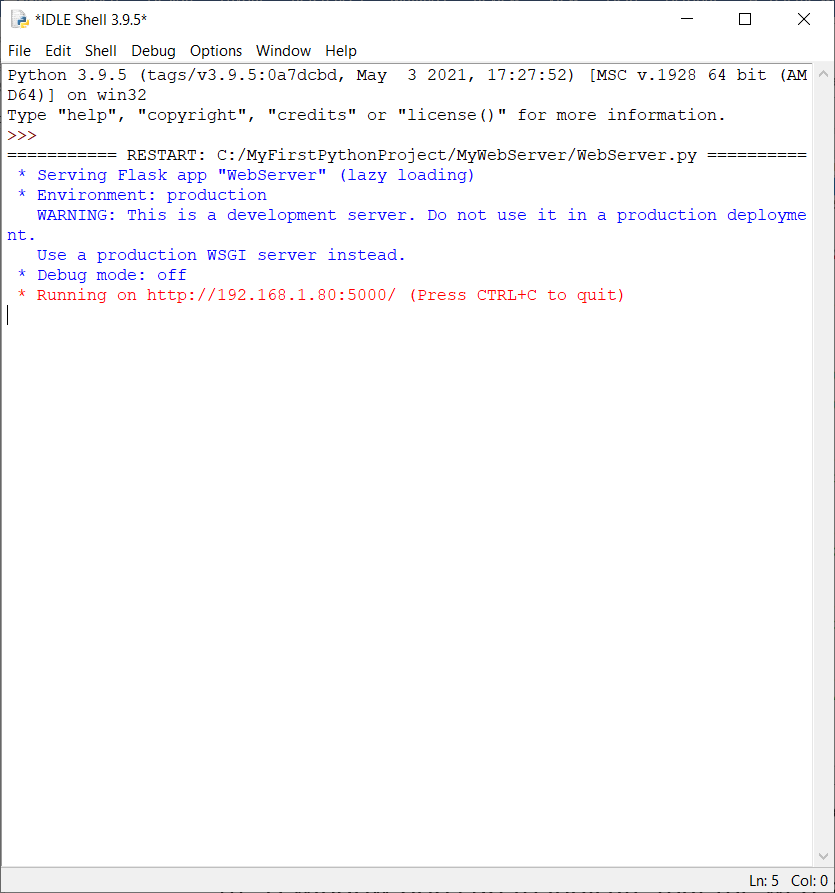


Figure 1.2h: Web Server is Running

**Setting Up the Web Pages**

1. **Look** for Visual Studio Code(VS Code)softwareby using the window search function and launch it as shown in the Figure below.

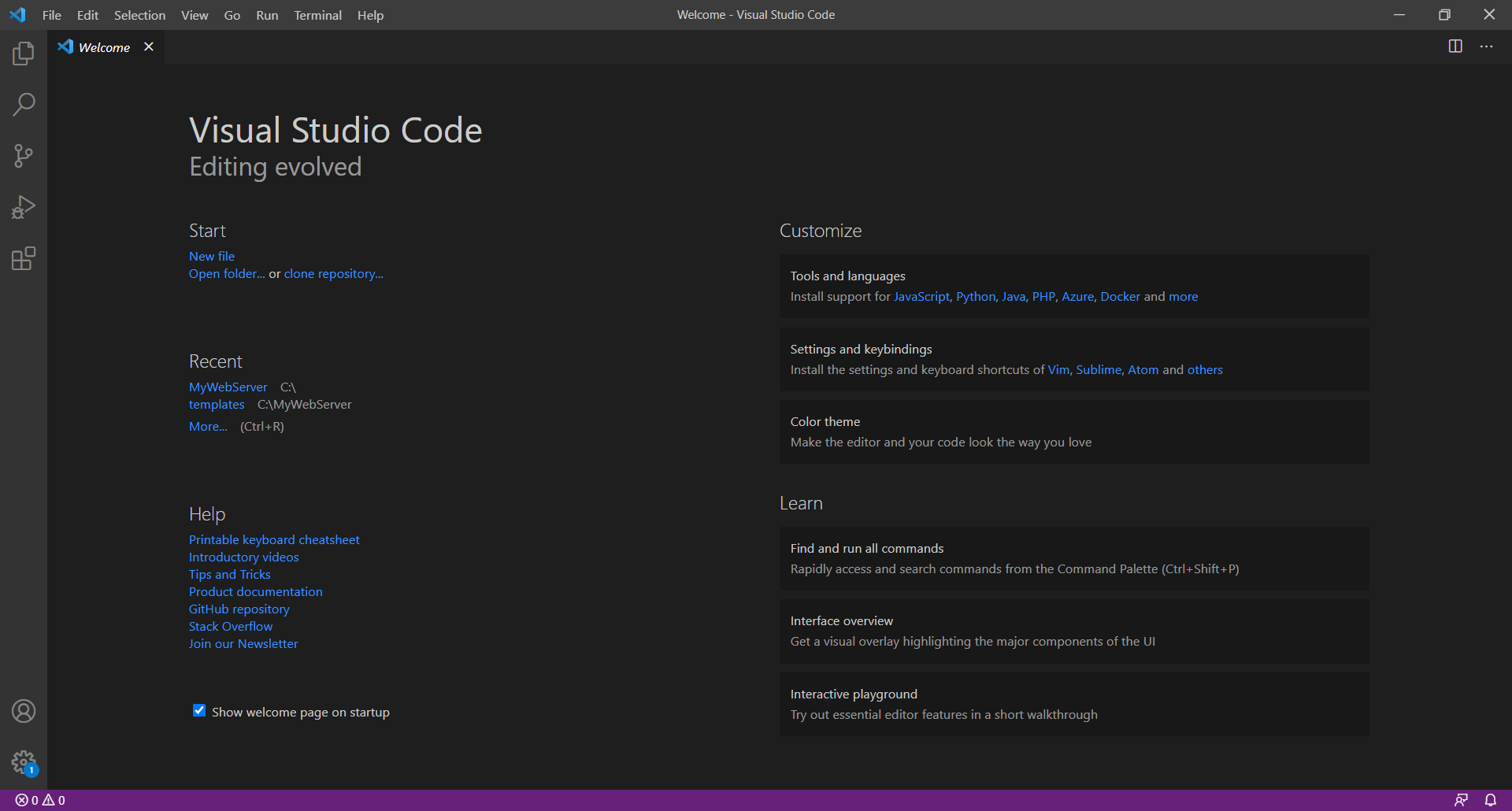


Figure 1.2i: Launching the Visual Studio Code (VS Code) Software

1. **Select** File > Open Folder… from the top menu in the Visual Studio Code(VS Code)softwareas shown in the Figure below.

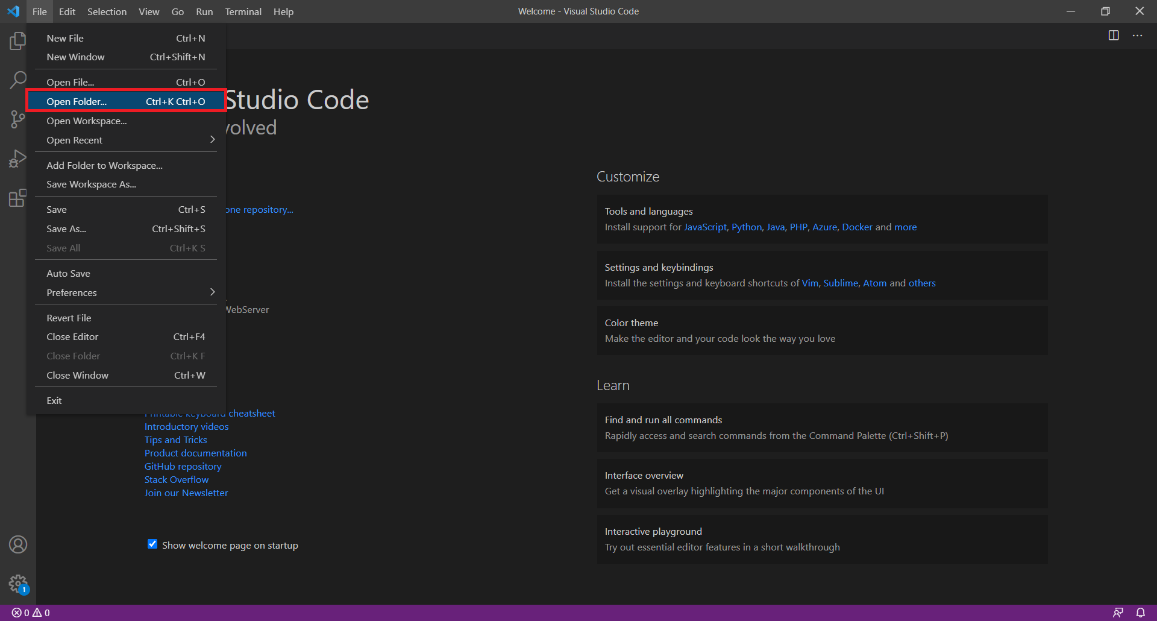


Figure 1.2j: Opening the Project Folder

1. **Navigate** to the MyFirstPythonProject folder created earlier in the C: drive and **click** on the “Select Folder” button as shown in Figure below.

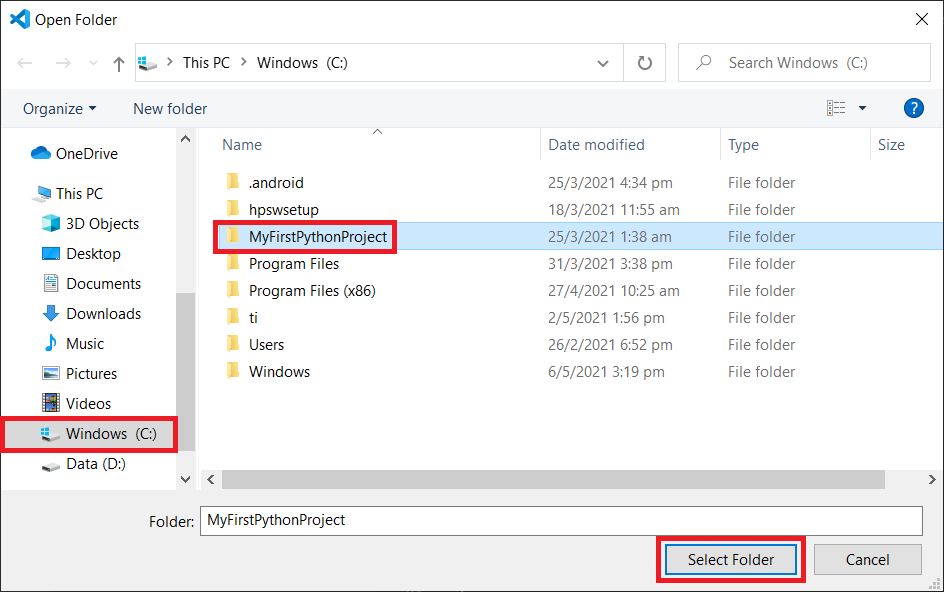


Figure 1.2k: Opening the Project Folder

1. **Ensure** that the MyFirstPythonProject folder is opened and shown on the left panel of the Visual Studio Code (VS Code) software as shown in Figure below.

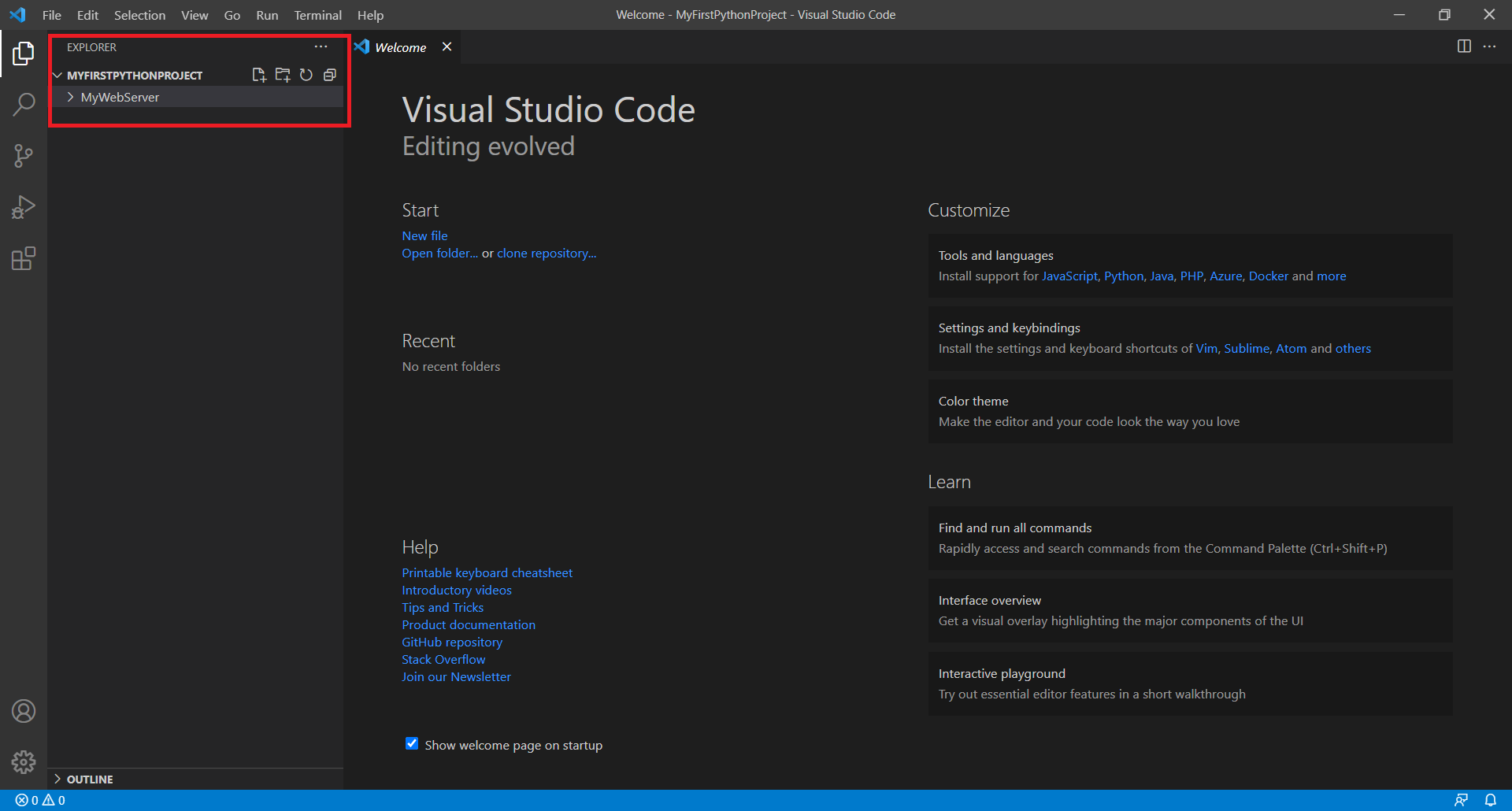


Figure 1.2l: Opening the Project Folder

1. **Expand** the MyWebServerfolder and **right-click** on the “**css**’’. **Select** “New File” from the drop-down menu to **create** a css file in the **css** folder and **name** it as “**style.css**” as shown in the Figure below.

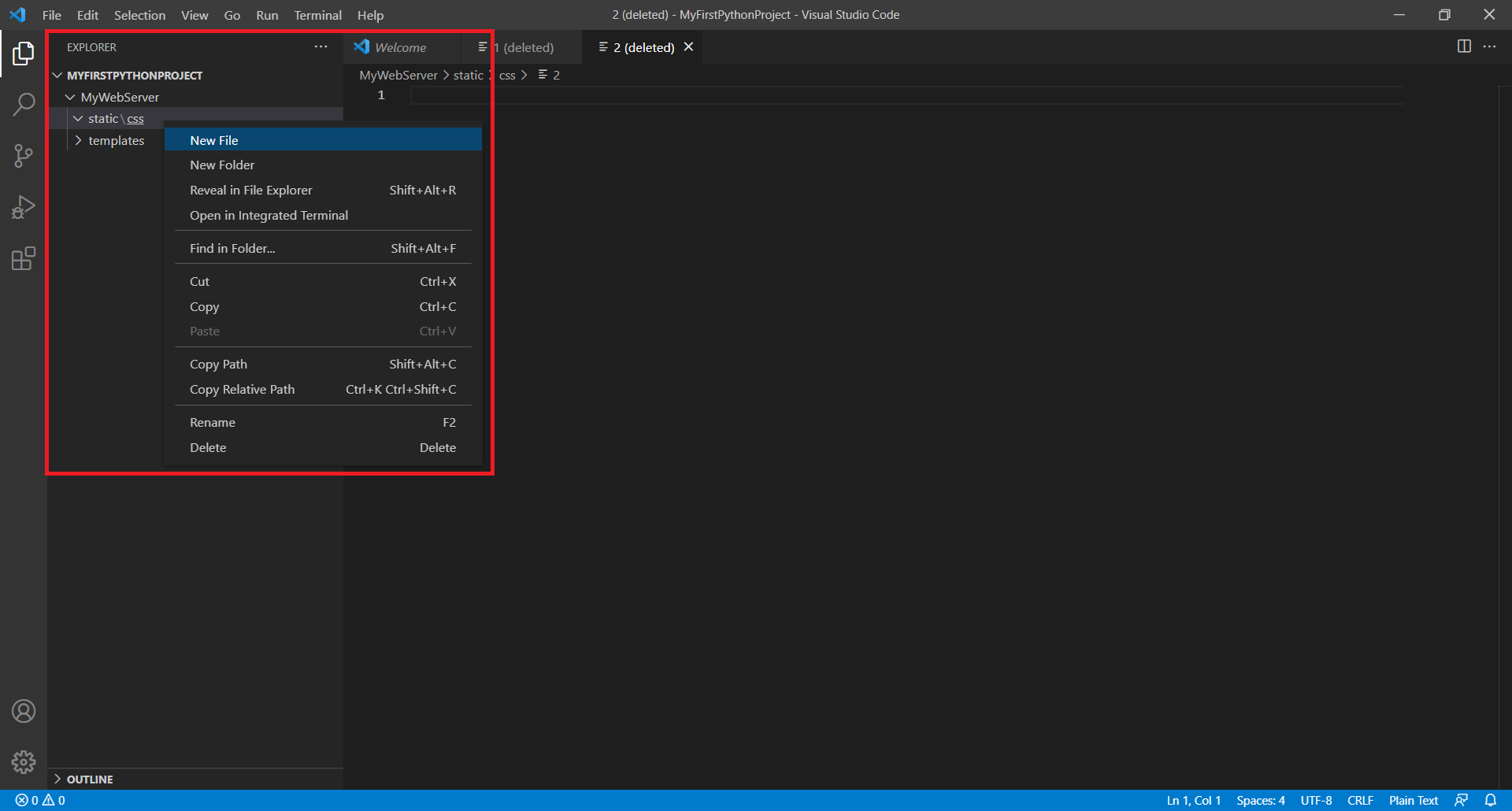


Figure 1.2m: Create a css file

1. **Enter** the following code into the “**style.css**” file and **save** the file.

|  |
| --- |
| body {  font-family: Arial, Helvetica, sans-serif;  font-size: 20px;  color: gray;  }  button {  border-radius: 30px;  padding: 16px 32px;  font-size: 16px;  transition-duration: 0.4s;  cursor: pointer;  width: 100px;  background-color: white;  border: 2px solid #008CBA;  }  button:hover {  background-color: #008CBA;  color: white;  }  table, td {  padding: 50px;  text-align: center;  border-collapse: collapse;  border: 1px solid #ddd;  }  tr:nth-child(even) {  background-color: #f2f2f2;  }  td:hover {  background-color:#ddd;  } |

1. **Create** a html file in the **templates** folder and name it as “**index.html**”. Enter the following code into the file. **Modify** the IP address of “**192.168.X.X**” according to the IP address you have retrieved from the PC earlier.

|  |
| --- |
| <html>  <head>  <link rel='stylesheet' href="{{ url\_for('static', filename='css/style.css') }}">  <script src='https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js'></script>  <script src='https://cdnjs.cloudflare.com/ajax/libs/socket.io/3.1.2/socket.io.js'></script>  <script type='text/javascript'>  $(document).ready(function(){  var socket = io.connect(**'http://192.168.X.X:5000'**);    $('OnButton').click(function(){  socket.emit('ControlUSR0Led','on');  });  $('OffButton').click(function(){  socket.emit('ControlUSR0Led','off');  });    socket.on('Web\_BBBW1Event', function(RxData) {  $('bbbw1').text(RxData.data);  });    socket.on('Web\_BBBW2Event', function(RxData) {  $('bbbw2').text(RxData.data);  });    socket.on('Web\_BBBW3Event', function(RxData) {  $('bbbw3').text(RxData.data);  });    socket.on('Web\_BBBW4Event', function(RxData) {  $('bbbw4').text(RxData.data);  });  });  </script>  </head>  <body>  <center>  <h1>My First Connected System!</h1>  <table style='width:100%'>  <tr>  <td colspan="2">  <h3>Controlling all the BBBW Boards</h3>  <OnButton><button>ON</button></OnButton>  <OffButton><button>OFF</button></OffButton>  </td>  </tr>  <tr>  <td style='width:50%'>  <h3>BBBW 1</h3>  <h4>I am a Random Number Generator!</h4>  <p>Your Lucky Number is <bbbw1>???</bbbw1></p>  </td>  <td style='width:50%'>  <h3>BBBW 2</h3>  <h4>I am a Clock!</h4>  <p>The Local Time Now is <bbbw2>???</bbbw2></p>  </td>  </tr>  <tr>  <td style='width:50%'>  <h3>BBBW 3</h3>  <h4>I am a Mathematician!</h4>  <p><bbbw3>1 + 1 = 2</bbbw3></p>  </td>  <td style='width:50%'>  <h3>BBBW 4</h3>  <h4>I am a Weather Station!</h4>  <p>The Current Temperature in Singapore is <bbbw4>???</bbbw4> C</p>  </td>  </tr>  </table>  <center>  </body>  </html> |

**Accessing the Web Server**

1. **Launch** the internet browser of a PC (chrome) or mobile phone (safari) that are connected to the same Wi-Fi network in which the BBBW board is connected to.
2. **Type** the following IP address with port number “**192.168.X.X:5000**” on the URL address bar of the browser and **hit** the Enter key. **Replace** the **X** in the IP address according to the IP address you have retrieved from the PC earlier. In this example, **“192.168.1.2:5000”** is used it is observed that the following user interface is printed on the browser as shown in Figure below.

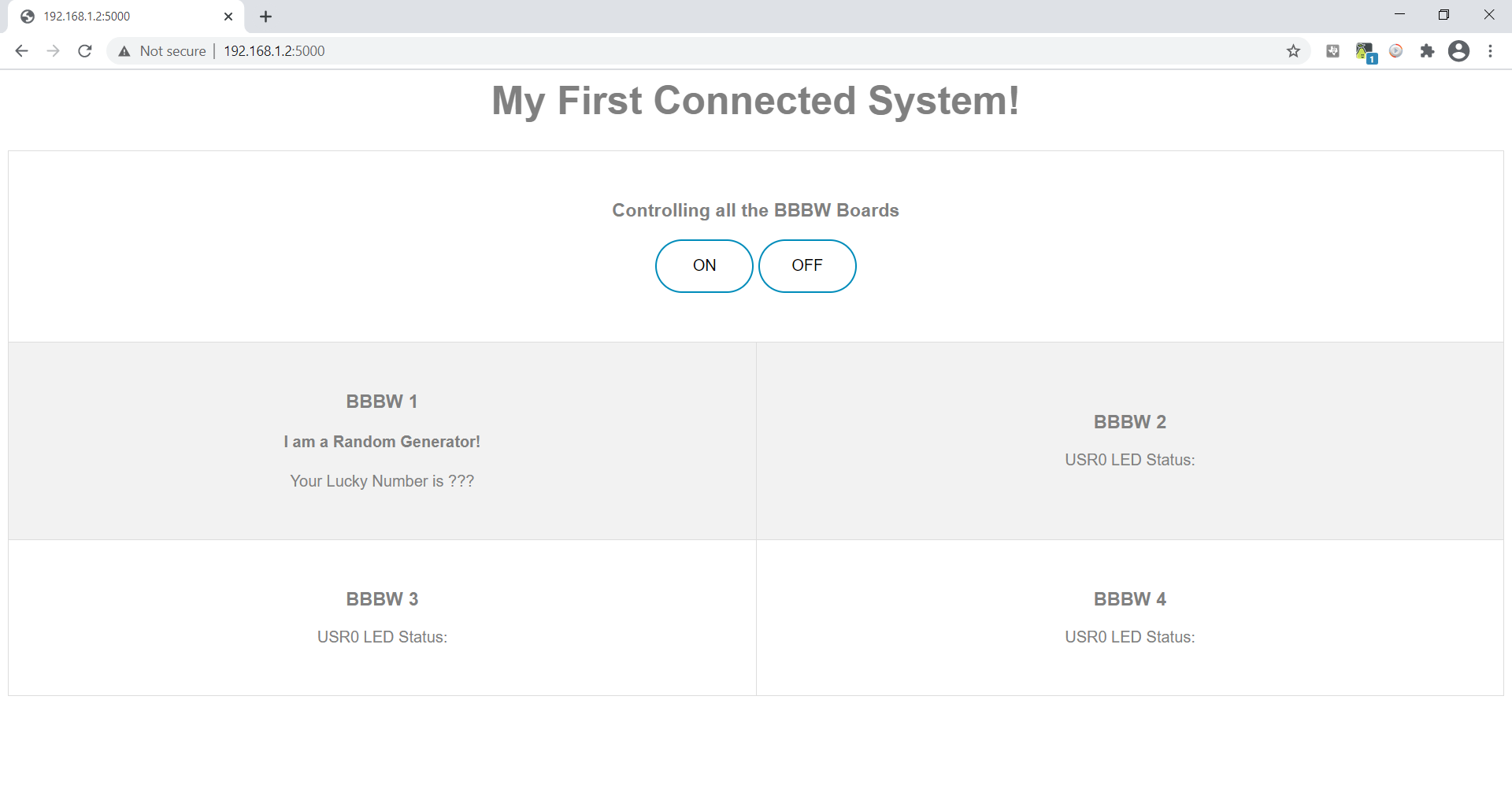


Figure 1.2n: Accessing the Web Server on PC

## Setting up Multiple Web Clients in Multiple BBBW Boards

**Installation of Python Library**

1. **Log in** to the BBBW Board through SSH using the default username “**debian**” and password “**temppwd**”.
2. **Use** the Connmanctl CLI command to connect the BBBW Board to the same Wi-Fi that the PC Web Server is connected to.
3. **Skip** step 4 to 15 if **Flask-SocketIO** and **requests** python libraryhas been downloaded and installed earlier.
4. **Type** in the command “**pwd**” and **hit** the “Enter” key. Ensure that the current working directory is “**/home/debian**” as shown in the Figure below.



Figure 1.3a: Current Working Directory

1. **Type** in the command “**ls**” and **hit** the “Enter” key. **Confirm** that the folder named PythonLibrary has been created as shown in the Figure below. If it is not created, **type** in the command “**mkdir PythonLibrary**” to create the folder.



Figure 1.3b: Ensuring PythonLibrary Folder Is Created

1. **Type** in the command “**cd PythonLibrary**” and **hit** the “Enter” key as shown in the Figure below.

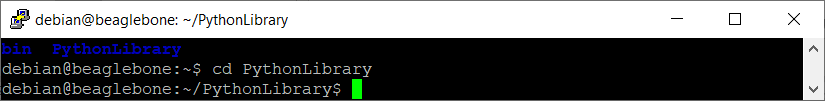


Figure 1.3c: Accessing the PythonLibrary Folder

1. **Type** in the command “**git clone https://github.com/nypege205/Flask-SocketIO.git**” and **hit** the “Enter” key. **Type** in the github username “**nypege205**” and password “**ghp\_Dr3jDaeKJ8fgDH06ZrtG1qUKgsmKux3XffG5**” and **hit** the “Enter” key again. A copy of the Flask-SocketIO Python Library is now cloned into the PythonLibrary folder as shown in the Figure below.

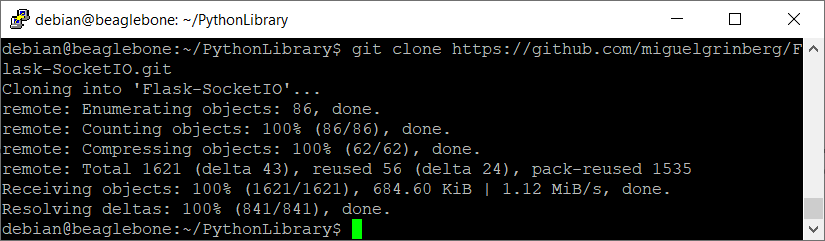


Figure 1.3d: Cloning the Flask-SocketIO Python Library

1. **Type** in the command “**ls**” and **hit** the “Enter” key. A folder named Flask-SocketIO is listed as shown in the Figure below.

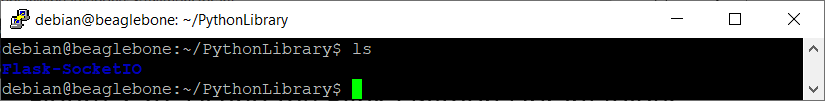


Figure 1.3e: Listing the Flask-SocketIO Python Library Folder

1. **Type** in the command “**cd Flask-SocketIO**” and **hit** the “Enter” key as shown in the Figure below.

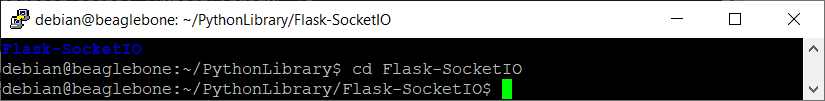


Figure 1.3f: Accessing the Flask-SocketIO Folder

1. **Type** in the command “**sudo python3 setup.py install**” and **hit** the “Enter” key. **Type** in the password “**temppwd**” and **hit** the “Enter” key again. The installation may take up to 5 minutes. The completion of the installation process is shown in the Figure below.

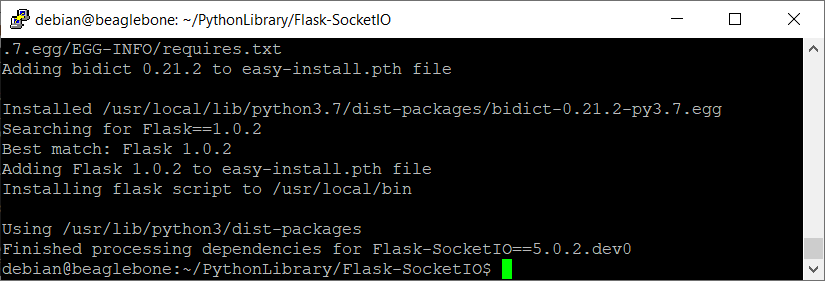


Figure 1.3g: Completion of Flask-SocketIO Python Library Installation

1. **Type** in the command “**cd ..**” and **hit** the “Enter” key to exit the Flask-SocketIO directory/folder as shown in the Figure below.

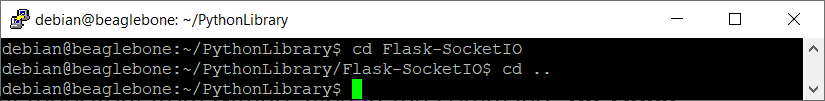


Figure 1.3h: Exit to Flask-SocketIO Folder

1. **Type** in the command “**git clone https://github.com/nypege205/requests.git**” and **hit** the “Enter” key. **Type** in the github username “**nypege205**” and password “**ghp\_Dr3jDaeKJ8fgDH06ZrtG1qUKgsmKux3XffG5**” and **hit** the “Enter” key again. A copy of the Flask-SocketIO Python Library is now cloned into the PythonLibrary folder as shown in the Figure below.

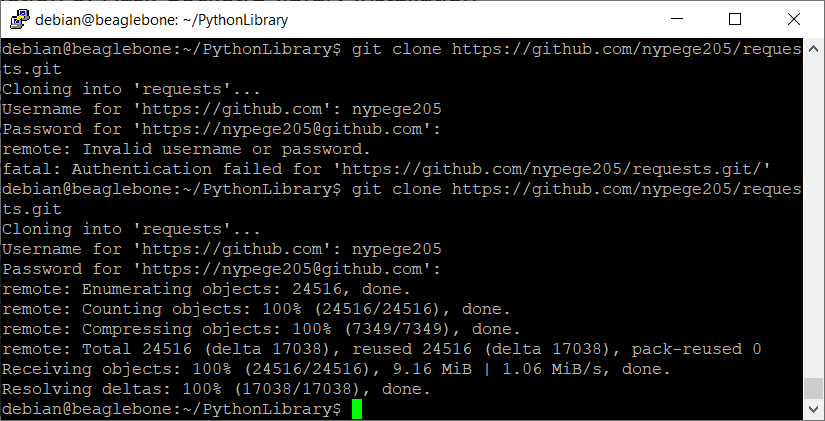


Figure 1.3i: Cloning the requests Python Library

1. **Type** in the command “**ls**” and **hit** the “Enter” key. A folder named “**requests**” is listed as shown in the Figure below.

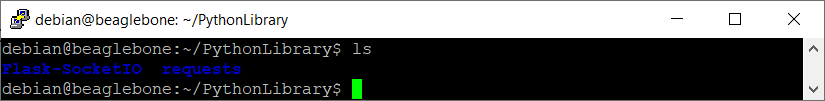


Figure 1.3j: Listing the requests Python Library Folder

1. **Type** in the command “**cd requests**” and **hit** the “Enter” key as shown in the Figure below.

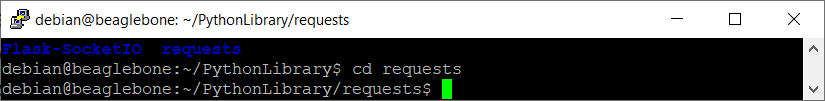


Figure 1.3k: Accessing the Flask-SocketIO folder

1. **Type** in the command “**sudo python3 setup.py install**” and **hit** the “Enter” key. **Type** in the password “**temppwd**” and **hit** the “Enter” key again. The installation may take up to 5 minutes. The completion of the installation process is shown in the Figure below.

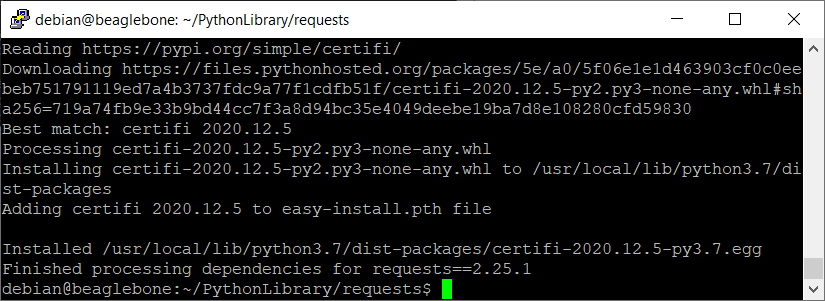


Figure 1.3l: Completion of requests Python Library Installation

**Creating and Executing Python File in Cloud9 IDE**

1. **Open** the web browser (preferably Chrome browser) and **type** “**http://192.168.7.2:3000**” in the address bar.
2. **Right click** on the folder “**MyFirstPythonProject”** and **select** the “**New File**” from the drop-down menu to create a new html file. Name the file as “**WebClient.py**” as shown in the Figure below.

|  |  |
| --- | --- |
|  |  |

Figure 1.3t: Creating a New File

1. **Double click** on the newly created python file “**WebClient.py**” and **enter** the following code into the file under the Editor section. **Modify** the IP address of “**192.168.X.X**” according to the IP address you have retrieved from the PC earlier.
2. **Choose** **ONLY** 1 of the following code to be implemented in the BBBW board. **Discuss** among the team members to avoid using the same code below.

**For BBBW1 Board**

|  |
| --- |
| import socketio  import random  import time  import Adafruit\_BBIO.GPIO as GPIO  sio = socketio.Client()  GPIO.setup('USR0', GPIO.OUT)  @sio.event  def connect():  print('Connection established.')  @sio.event  def disconnect():  print('Disconnected from server.')    @sio.event  def ControlUSR0Led(RxData):  if RxData == 'on':  GPIO.output('USR0', GPIO.HIGH)  print('USD0 LED is ON!')  if RxData == 'off':  GPIO.output('USR0', GPIO.LOW)  print('USD0 LED is OFF!')  while True:  try:  sio.connect(**'http://192.168.X.X:5000**')  break  except:  print("Try to connect to the server.")  pass  while True:  try:  RandomNum = random.randint(0,999)  sio.emit('BBBW1Event', {'data': RandomNum})  print('Data sent!')  except:  pass  print('Unable to transmit data.')  time.sleep(2) |

**For BBBW2 Board**

|  |
| --- |
| import socketio  import random  import time  import Adafruit\_BBIO.GPIO as GPIO  sio = socketio.Client()  GPIO.setup('USR0', GPIO.OUT)  @sio.event  def connect():  print('Connection established.')  @sio.event  def disconnect():  print('Disconnected from server.')    @sio.event  def ControlUSR0Led(RxData):  if RxData == 'on':  GPIO.output('USR0', GPIO.HIGH)  print('USD0 LED is ON!')  if RxData == 'off':  GPIO.output('USR0', GPIO.LOW)  print('USD0 LED is OFF!')  while True:  try:  sio.connect(**'http://192.168.X.X:5000**')  break  except:  print("Try to connect to the server.")  pass  while True:  try:  LocalTime = time.asctime(time.localtime(time.time()))  sio.emit('BBBW2Event', {'data': LocalTime})  print('Data sent!')  except:  pass  print('Unable to transmit data.')  time.sleep(2) |

**For BBBW3 Board**

|  |
| --- |
| import socketio  import random  import time  import Adafruit\_BBIO.GPIO as GPIO  sio = socketio.Client()  GPIO.setup('USR0', GPIO.OUT)  @sio.event  def connect():  print('Connection established.')  @sio.event  def disconnect():  print('Disconnected from server.')    @sio.event  def ControlUSR0Led(RxData):  if RxData == 'on':  GPIO.output('USR0', GPIO.HIGH)  print('USD0 LED is ON!')  if RxData == 'off':  GPIO.output('USR0', GPIO.LOW)  print('USD0 LED is OFF!')  while True:  try:  sio.connect(**'http://192.168.X.X:5000**')  break  except:  print("Try to connect to the server.")  pass  while True:  try:  num1 = random.randint(0,999)  num2 = random.randint(0,999)  result = num1 + num2  TxData = '{:3d} + {:3d} = {:4d}' .format(num1, num2, result)  sio.emit('BBBW3Event', {'data': TxData})  print('Data sent!')  except:  pass  print('Unable to transmit data.')  time.sleep(2) |

**For BBBW4 Board**

|  |
| --- |
| import socketio  import random  import time  import requests  import json  import Adafruit\_BBIO.GPIO as GPIO  sio = socketio.Client()  GPIO.setup('USR0', GPIO.OUT)  @sio.event  def connect():  print('Connection established.')  @sio.event  def disconnect():  print('Disconnected from server.')    @sio.event  def ControlUSR0Led(RxData):  if RxData == 'on':  GPIO.output('USR0', GPIO.HIGH)  print('USD0 LED is ON!')  if RxData == 'off':  GPIO.output('USR0', GPIO.LOW)  print('USD0 LED is OFF!')  while True:  try:  sio.connect(**'http://192.168.X.X:5000**')  break  except:  print("Try to connect to the server.")  pass  while True:  try:  TxData = requests.get("https://api.data.gov.sg/v1/environment/air-temperature")  TxDataObject = json.loads(TxData.text)  TemperatureReading = TxDataObject['items']  for TemperatureReadings in TemperatureReading:  TemperatureValue = TemperatureReadings['readings']  sio.emit('BBBW4Event', {'data': TemperatureValue[0]["value"]})  print('Data sent!')  except:  pass  print('Unable to transmit data.')  time.sleep(2) |

1. **Click** on the “Run” button located beside the Menu Tab to execute the “**WebClient.py**” file.
2. **Ensure** the web server on the PC is up and running. It is observed that the respective BBBW boards are sending data to the web server as shown in the Figure below.

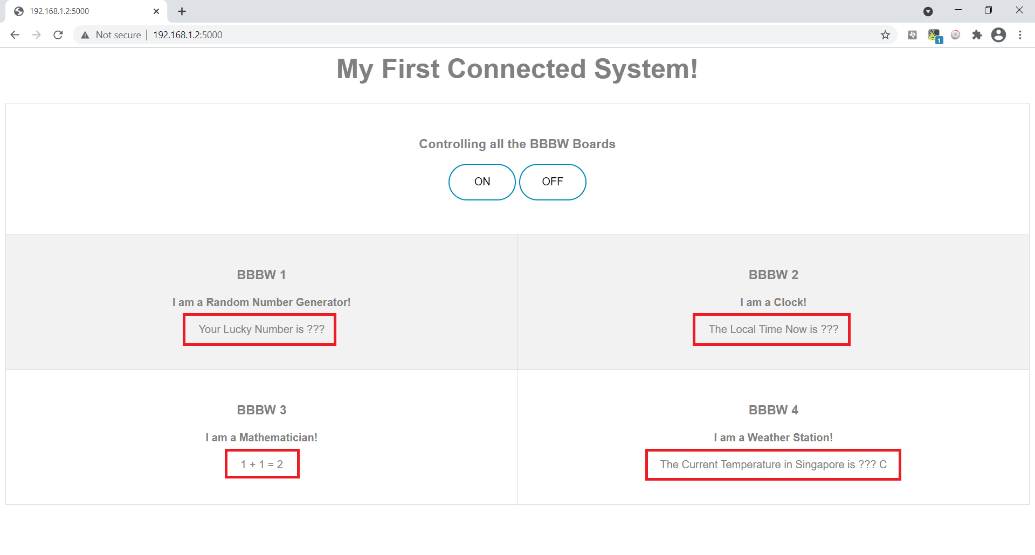


Figure 1.3u: BBBW Sending Data to Web Server

**Setting Python File to Run at Boot**

1. **Type** in the command “**sudo crontab -e”** and **hit** the enter key. Type in the password “**temppwd**” and **hit** the “Enter” key again. **Cron** is a task management tool built-in into Debian. **Type** **“1”** to choose the nano editor as shown in the Figure below.

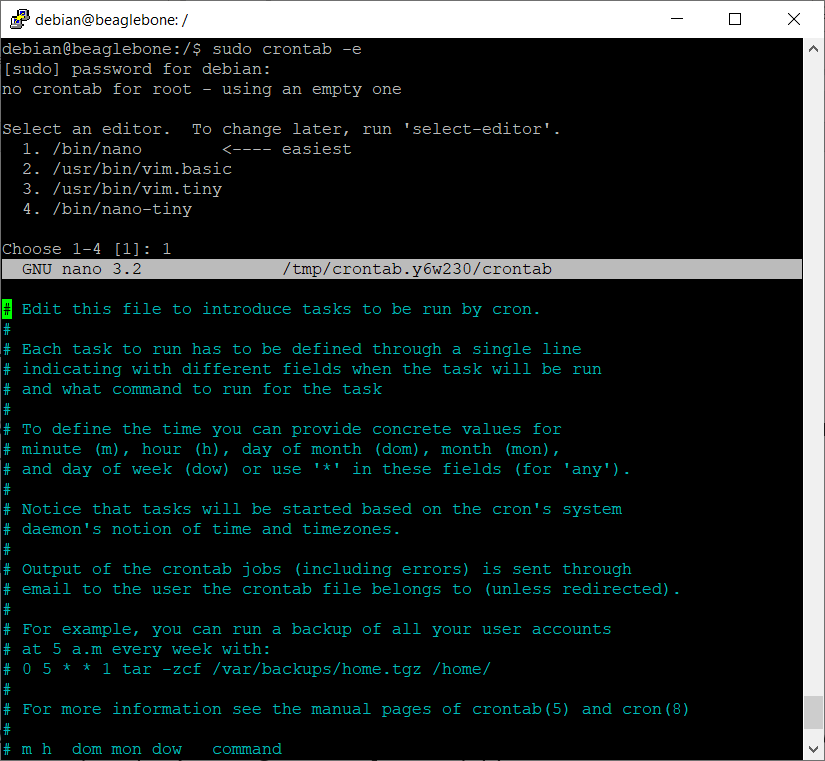


Figure 1.3v: Accessing Crontab

1. **Add** the text **“@reboot python3 /var/lib/cloud9/MyFirstPythonProject/WebClient.py &**” into the crontab using nano. This is to add the python file **WebClient.py** to be ran at boot as shown in the Figure below. **Nano** is a simple command-line text editor included in most Linux installations.

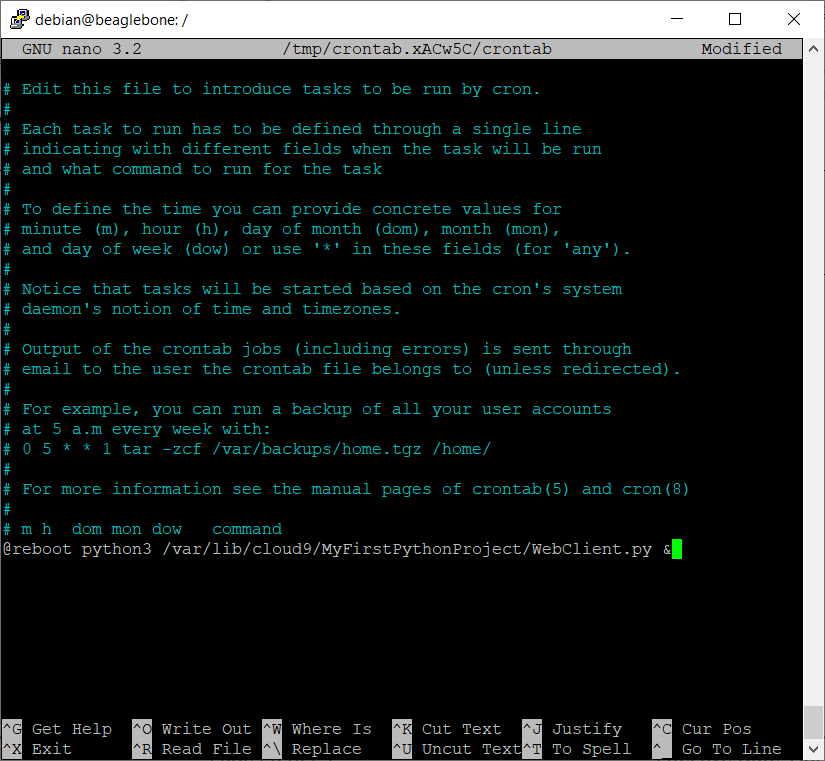


Figure 1.3w: Adding WebClient.py To Be Ran At Boot

1. Press the “**Ctrl + X**” key followed by “**y**” and “**Enter**” key to save the modified file and exit the nano text-editor as shown in the Figure below.

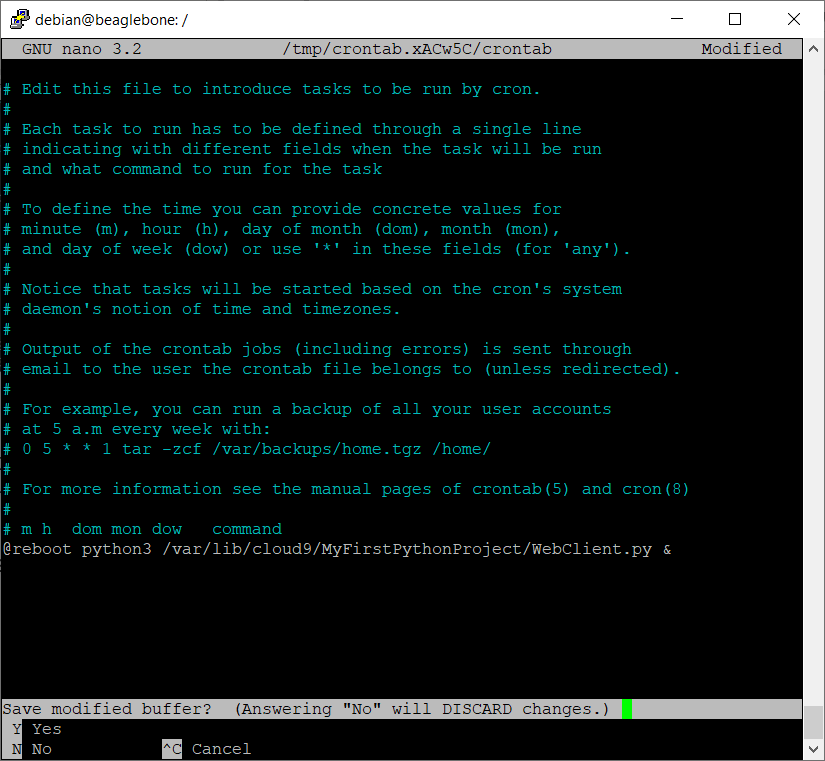


Figure 1.3x: Exiting the nano Text-editor

1. Type the command “**sudo reboot**” and **hit** the “Enter” key to reboot the BBBW board. The new settings will take effect on the next boot-up.

## Controlling and Monitoring Multiple BBBW Boards via Web Server on a PC

1. **Ensure** that the web server on PC is running.
2. **Ensure** that all the 4 BBBW boards are rebooted and is running.
3. **Observe** that the data from the 4 BBBW boards are streamed into the web server on PC as shown in the Figure below.

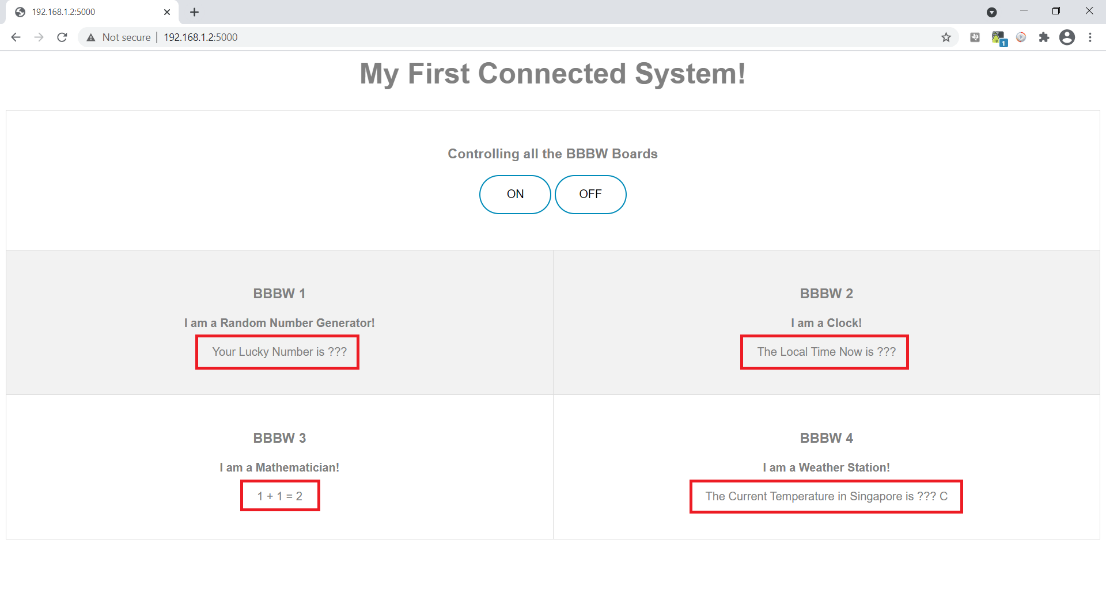


Figure 1.4a: Data Streamed into the Web Server on PC

1. **Click** on the “**ON**” or “**OFF**” button on the webpage as shown in the Figure below. It is observed that that the USR0 led located at all the 4 BBBW boards are turned on and off synchronously.

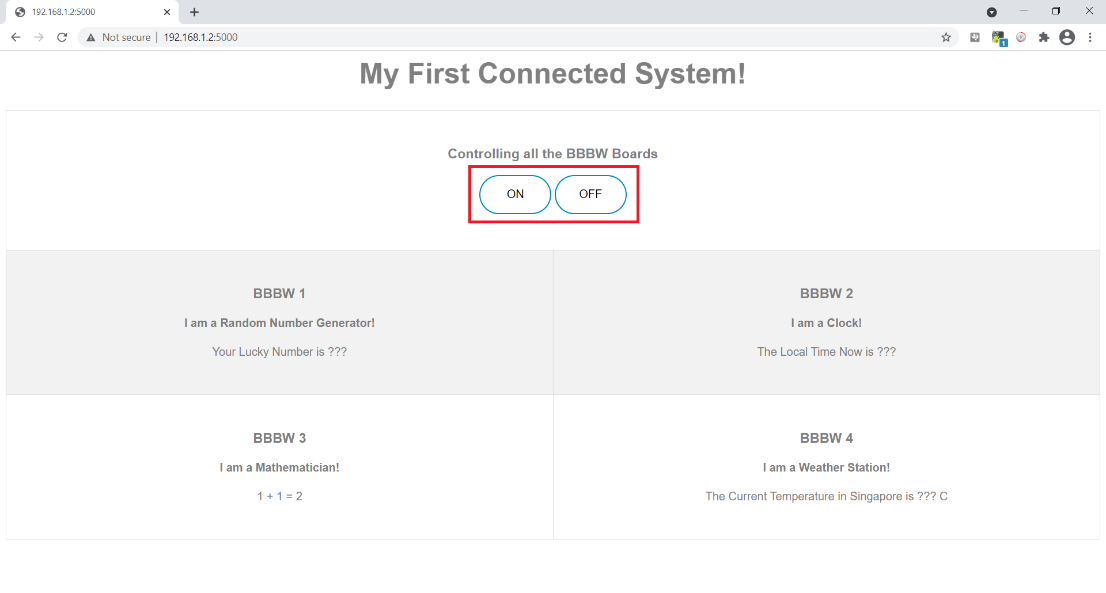


Figure 1.4b: Controlling the BBBW Boards from the Web Server on PC

*Congratulations! You have successfully completed the Lab6a. Good job! You are now ready to develop your own connected system!*