

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

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

**Laboratory No**. : Lab 5b

**Laboratory Title** : Networking: Setting up a Web Client using BeagleBone Black Wireless (BBBW) Board

**Objective** : To use a BeagleBone Black Wireless (BBBW) Board to set up a web client.

**Hardware Boards** : BBBW Board x1

Pot Click x1

**Contents**

1. Setting Up a Web Client using BeagleBone Black Wireless (BBBW) Board
   1. Understanding of Web Client and Socket.IO
   2. Setting up a PC Web Server
   3. Setting up the BBBW Board Web Client
   4. Controlling and Monitoring the BBBW Board Web Client via a PC Web Server
   5. Tinkering Time

# **Setting Up a Web Client using BeagleBone Black Wireless (BBBW) Board**

## Understanding of Web Client and Socket.IO Library

**Web client** is a piece of computer hardware or software that accesses a service made available by a server as part of the web client–server model of computer networks. For example, web browsers are web clients that connect to web servers and retrieve web pages for display. Email web clients retrieve email from mail servers. Multiplayer video games or online video games may run as a web client on each computer. The term "web client" may also be applied to computers or devices that run the web client software or users that use the web client software.

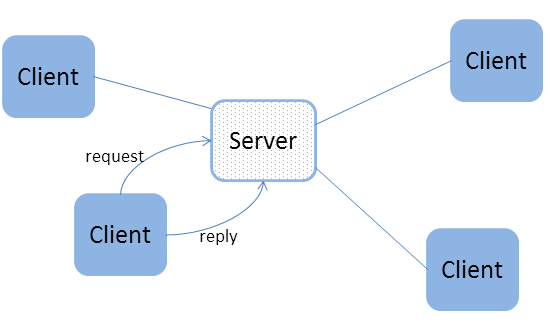


Figure 1.1a: Web client-server Model

**Socket.IO** is a JavaScript library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers. Socket.IO primarily uses the WebSocket protocol with polling as a fallback option while providing the same interface. Although it can be used as simply a wrapper for WebSocket, it provides many more features, including broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O.

**Flask-SocketIO** gives Flask applications access to low latency bi-directional communications between the clients and the server. The client-side application can use any of the SocketIO client libraries in JavaScript, Python, C++, Java and Swift, or any other compatible client to establish a permanent connection to the server.

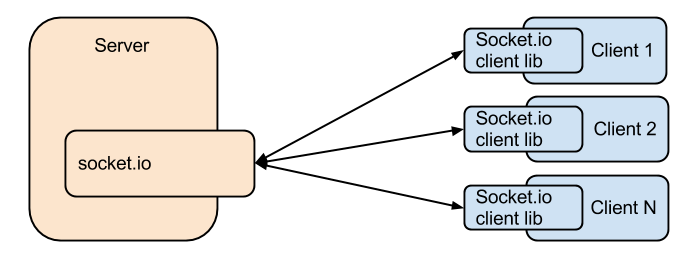


Figure 1.1b: Socket.IO Library

## Setting up a PC Web Server

**Downloading and Installing the Python and Visual Studio Code (VS Code) Software**

1. **Download** the latest python software from [www.python.org](http://www.python.org) as shown in the Figure below.

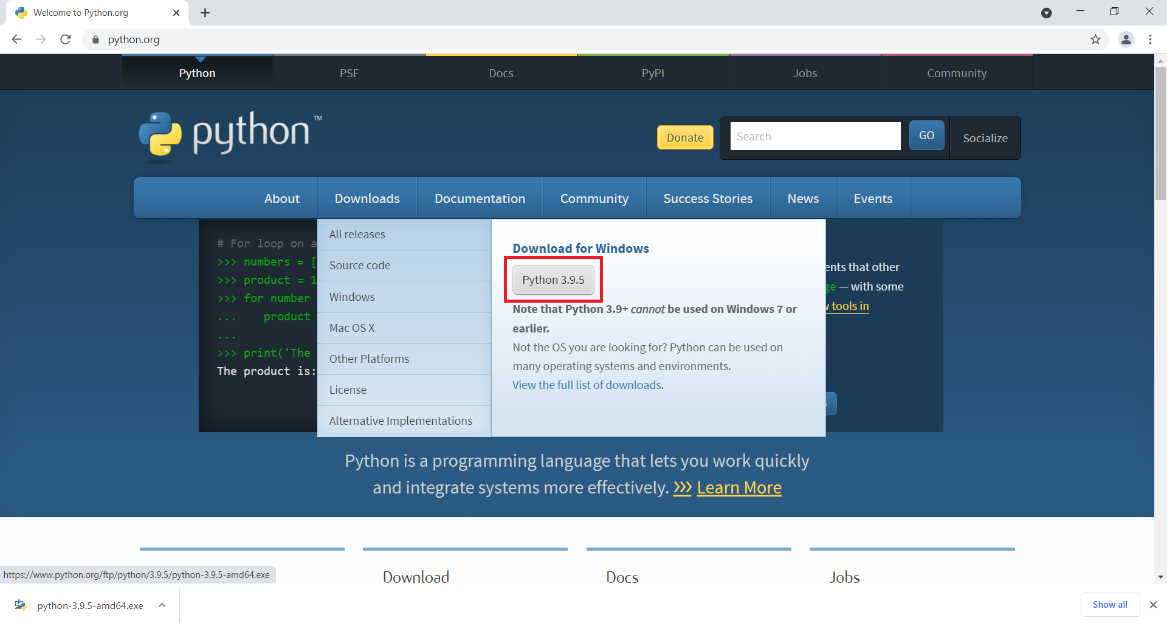


Figure 1.2a: Downloading the Latest Python Software for Windows

1. **Double click** on the installer, **check** on “Add Python 3.9 to Path” and **click** on **“Install Now**” to start the installation process. **Follow** the steps to complete the installation.

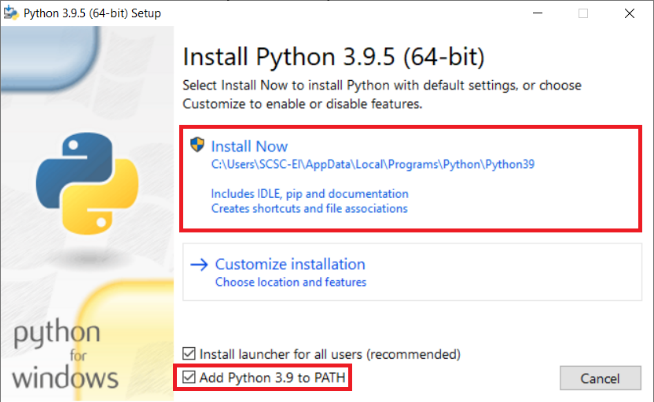


Figure 1.2b: Python Software Installation

1. **Download** the Visual Studio Code (VS Code) software from <https://code.visualstudio.com/> as shown in the Figure below.

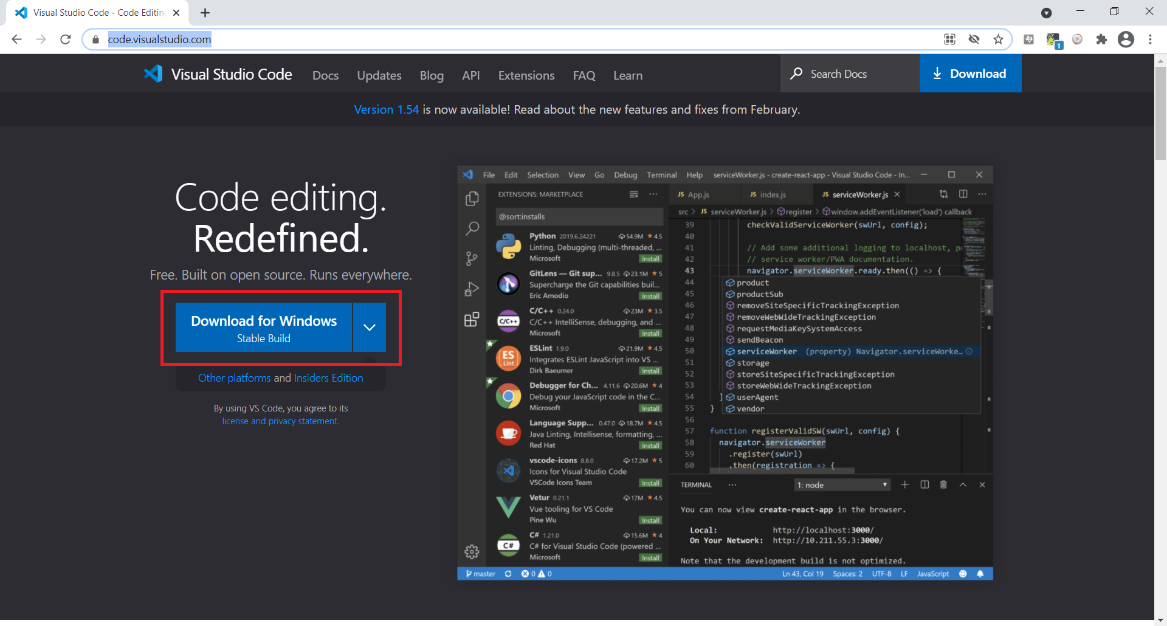


Figure 1.2c: Downloading the Visual Studio Code (VS Code) Software for Windows

1. **Double click** on the installer, **select** the radio button on “I accept the agreeement” and **click** on **“Next**” button as shown in the Figure below.

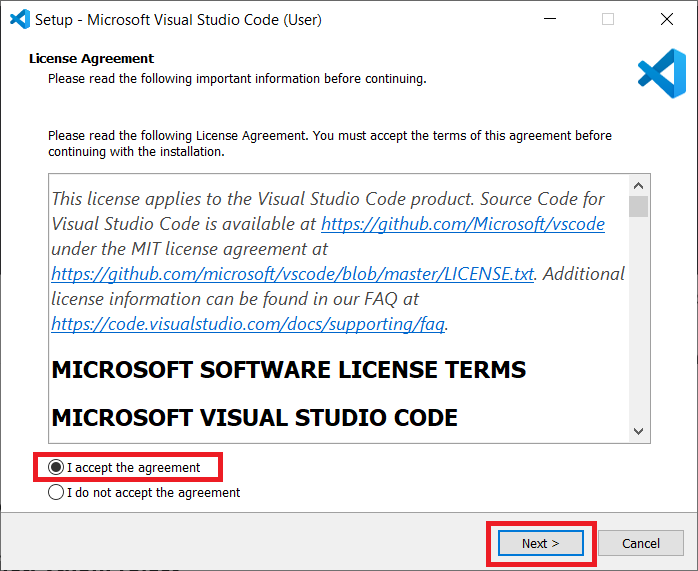


Figure 1.2d: VS Code Software Installation

1. **Click** on **“Next**” button to use the default installation location as shown in the Figure below.

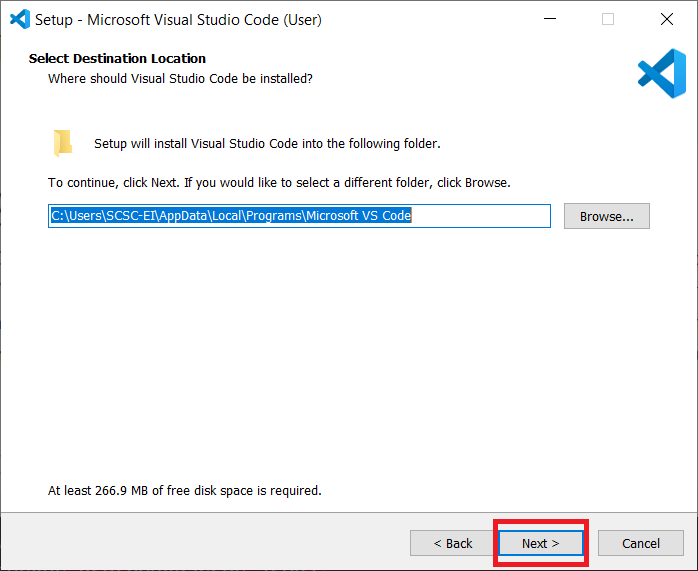


Figure 1.2e: VS Code Software Installation

1. **Click** on **“Next**” button again to use the recommended name for the Start Menu Folder as shown in the Figure below.

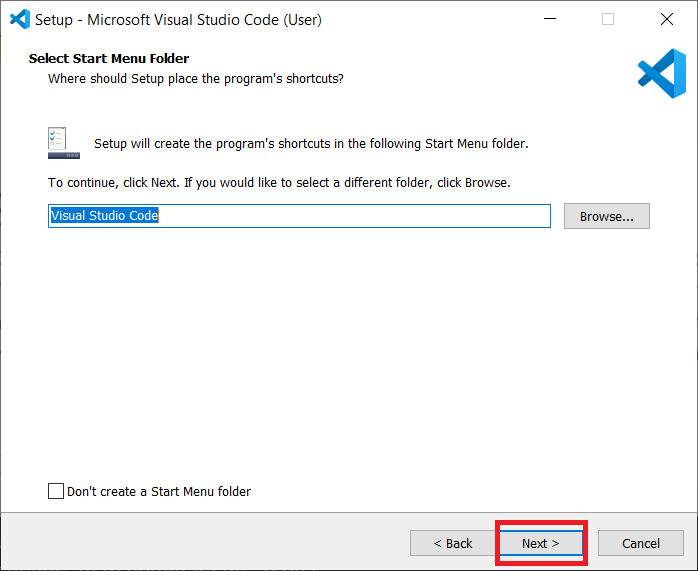


Figure 1.2f: VS Code Software Installation

1. **Check** on “Create a desktop icon” and “Add to PATH (requires shell restart). **Click** on **“Next**” button as shown in the Figure below.

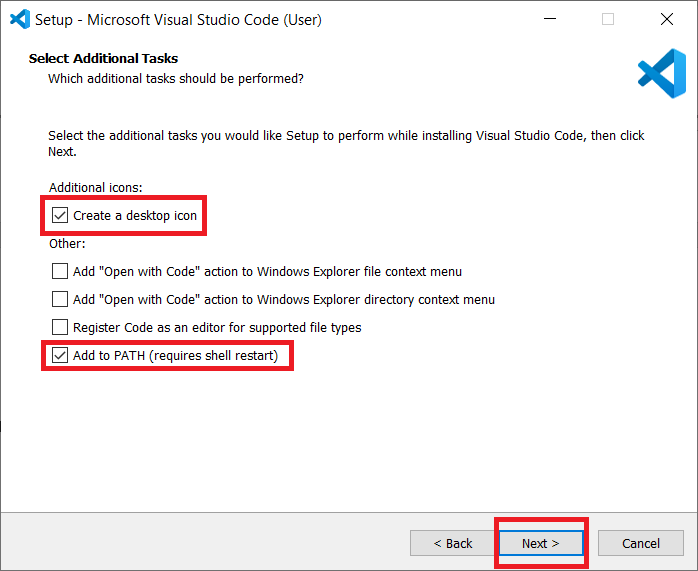


Figure 1.2g: VS Code Software Installation

1. **Click** on **“Install**” button to start the installation as shown in the Figure below.

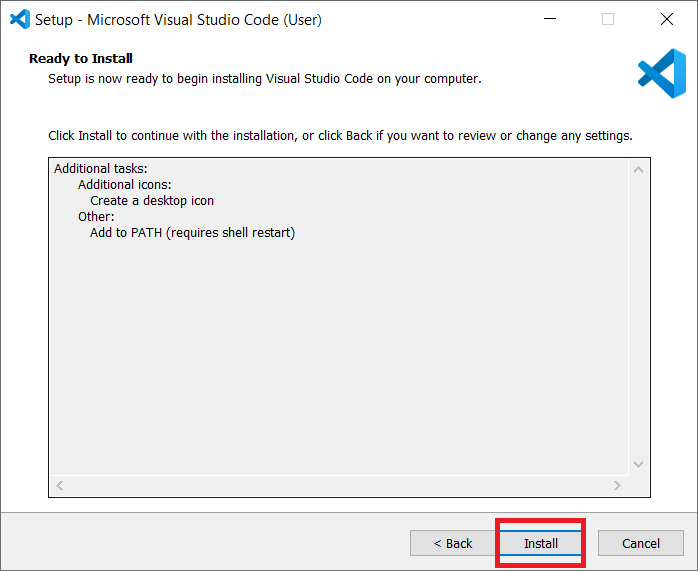


Figure 1.2h: VS Code Software Installation

**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**
      * **images**

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.

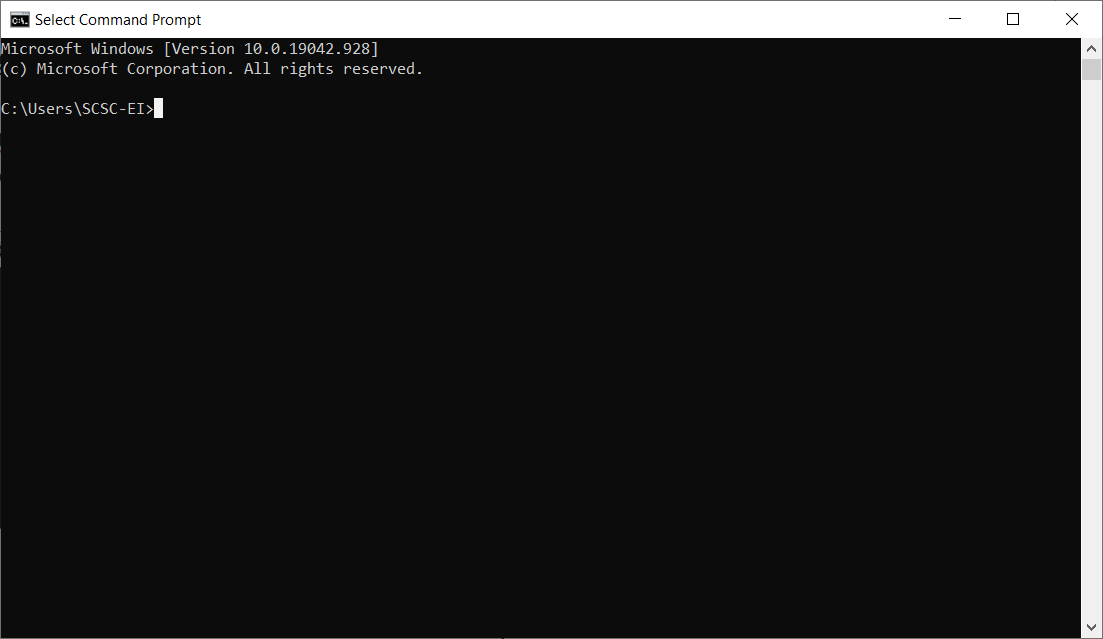


Figure 1.2i: Command Prompt

1. **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.

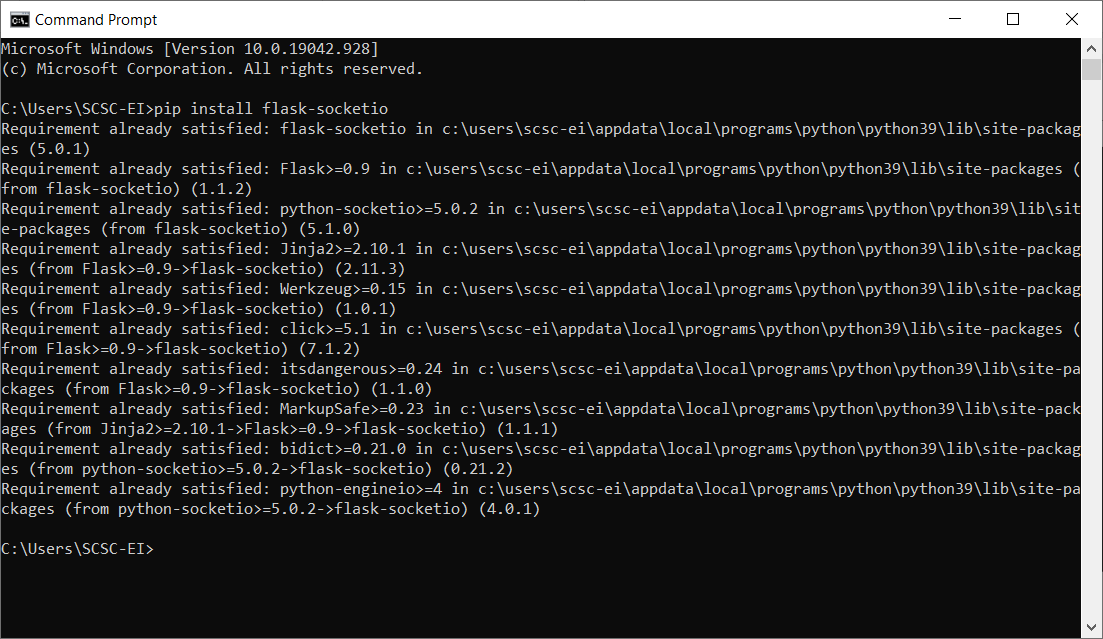


Figure 1.2j: Command Prompt

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.

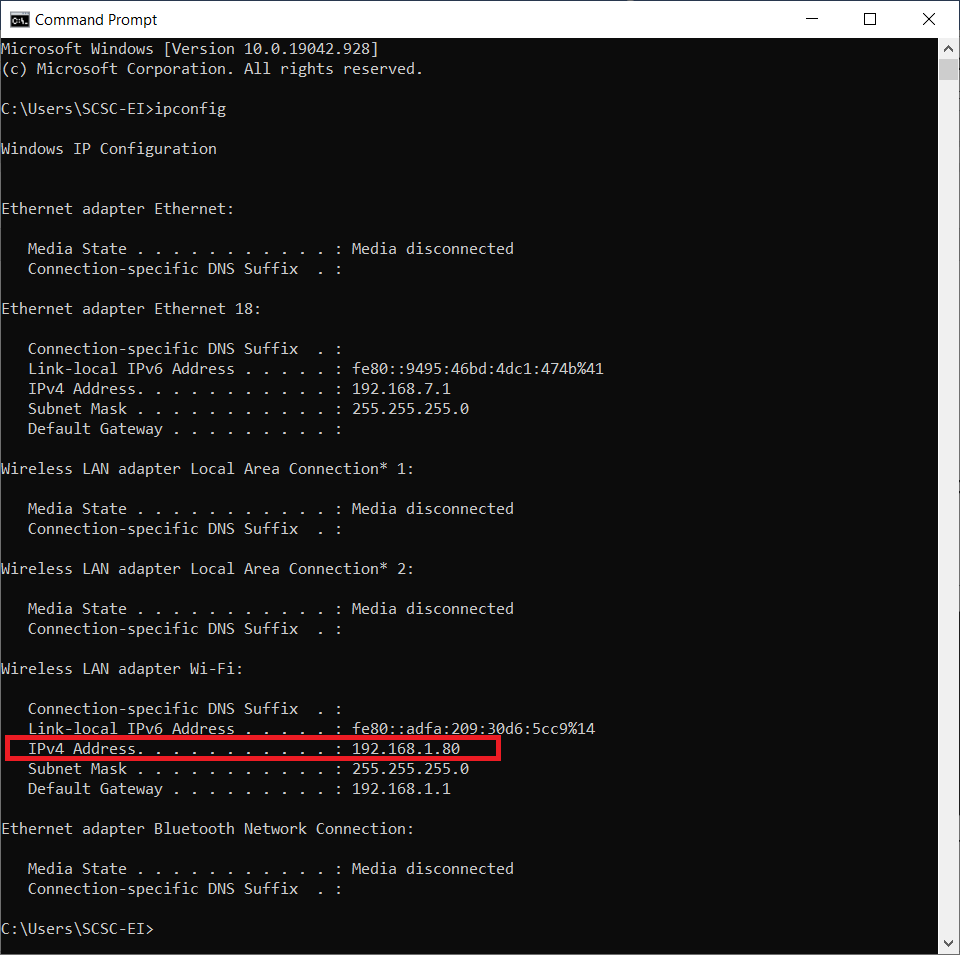


Figure 1.2k: IP Address Assigned to the PC

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.

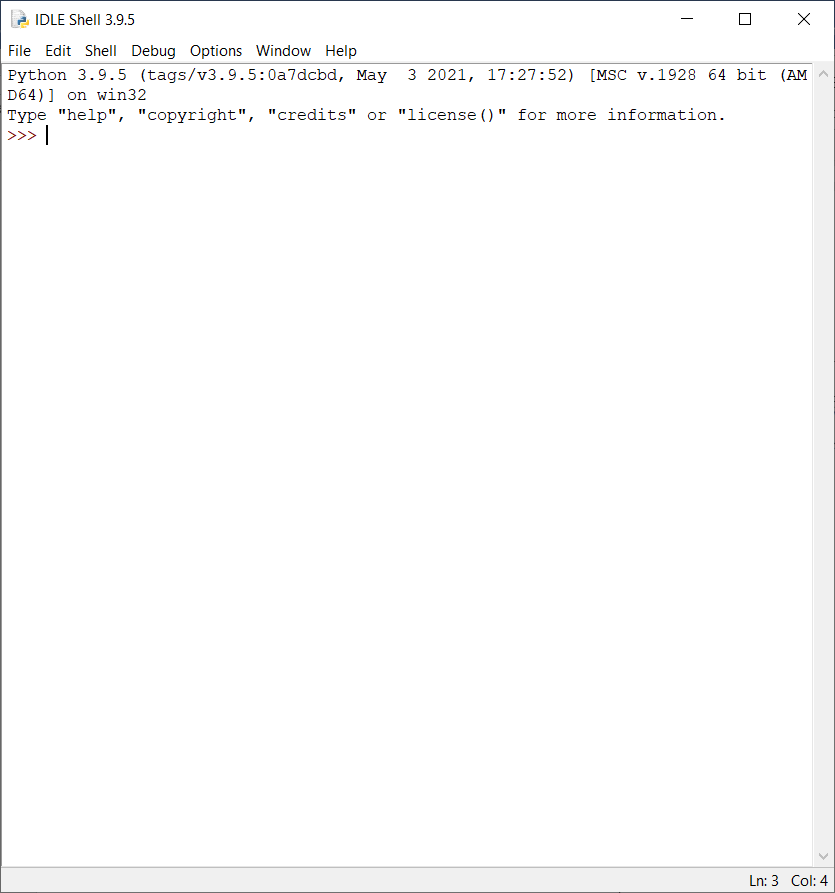


Figure 1.2l: Python IDLE Shell

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.

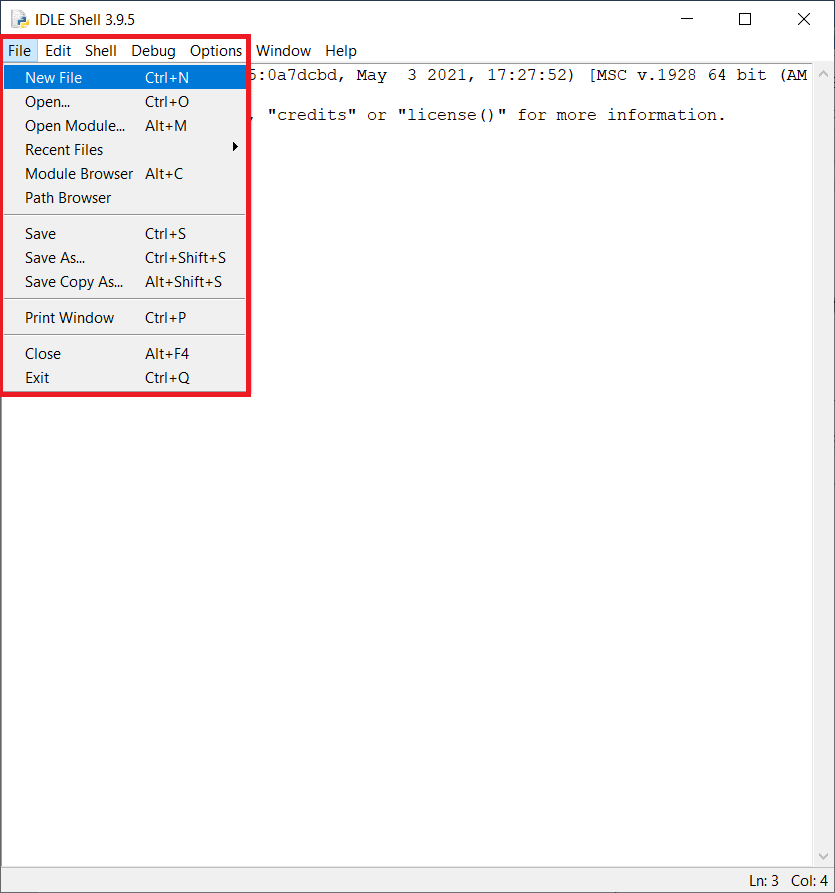


Figure 1.2m: Create new python file

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 BBBWEvent(RxData):  socketio.emit('Web\_BBBWEvent', RxData)  print('Receive Data from BBBW')  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**”.

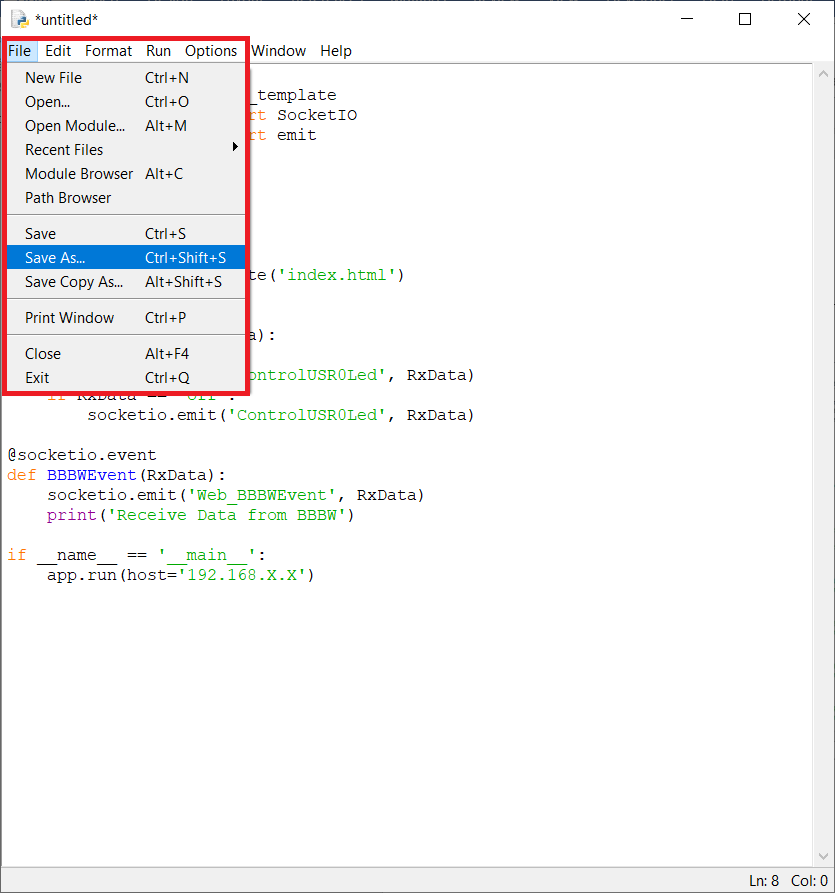


Figure 1.2n: Save the python file

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.

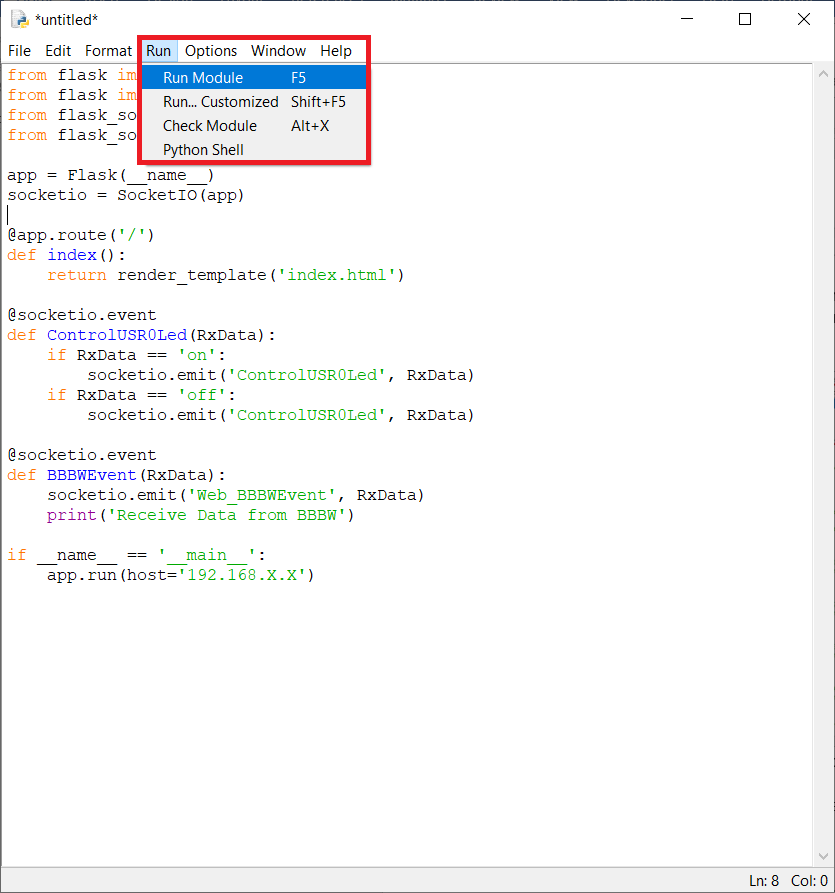


Figure 1.2o: Executing the python file

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

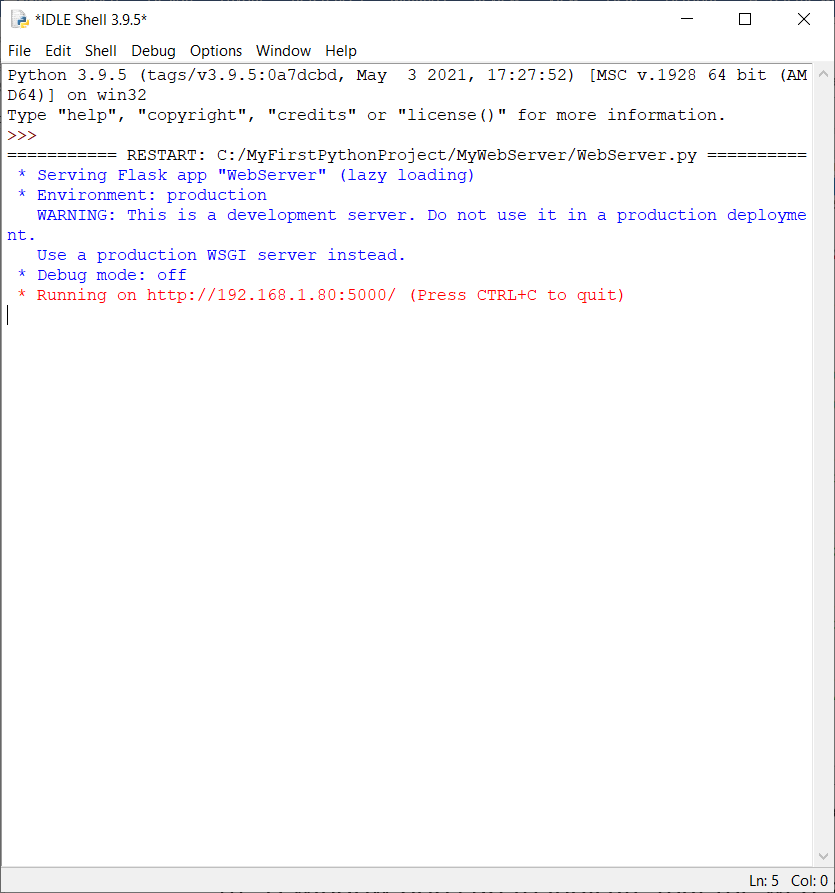


Figure 1.2p: Web Server is Running

**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.

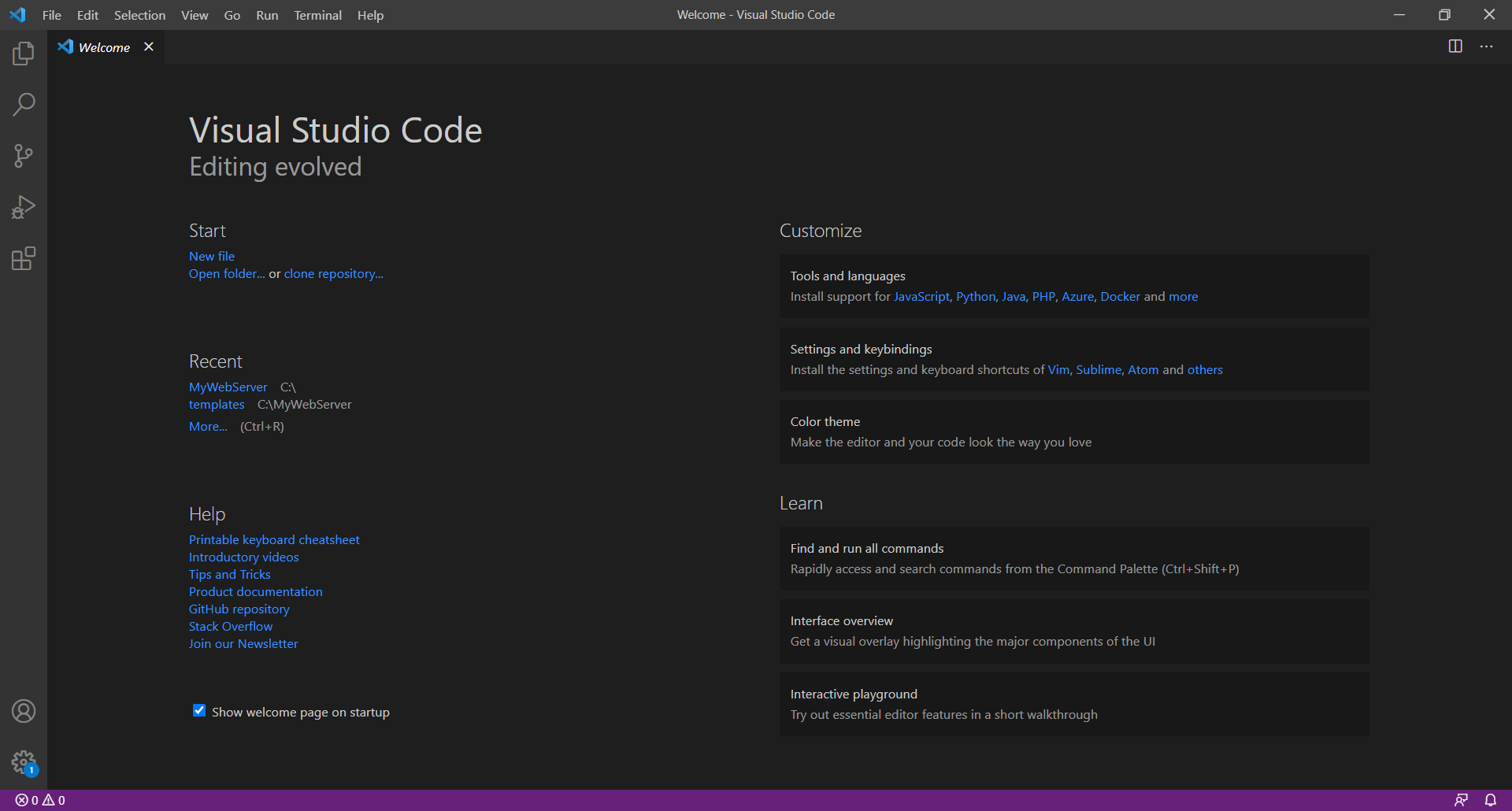


Figure 1.2q: Launching the Visual Studio Code (VS Code) Software

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

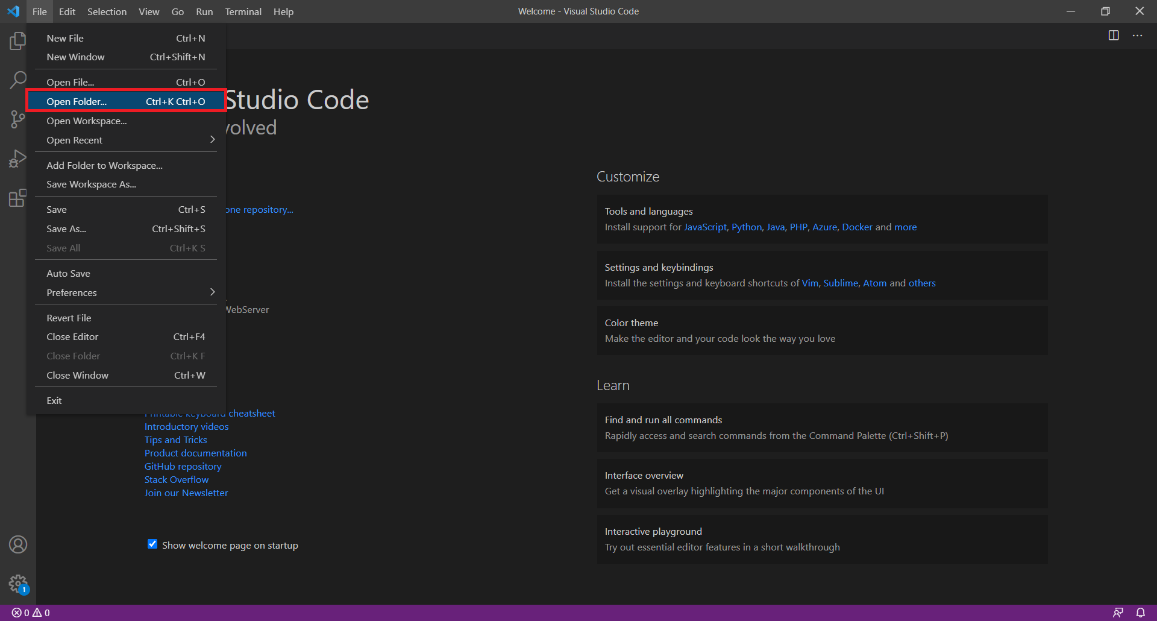


Figure 1.2r: Opening the Project Folder

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

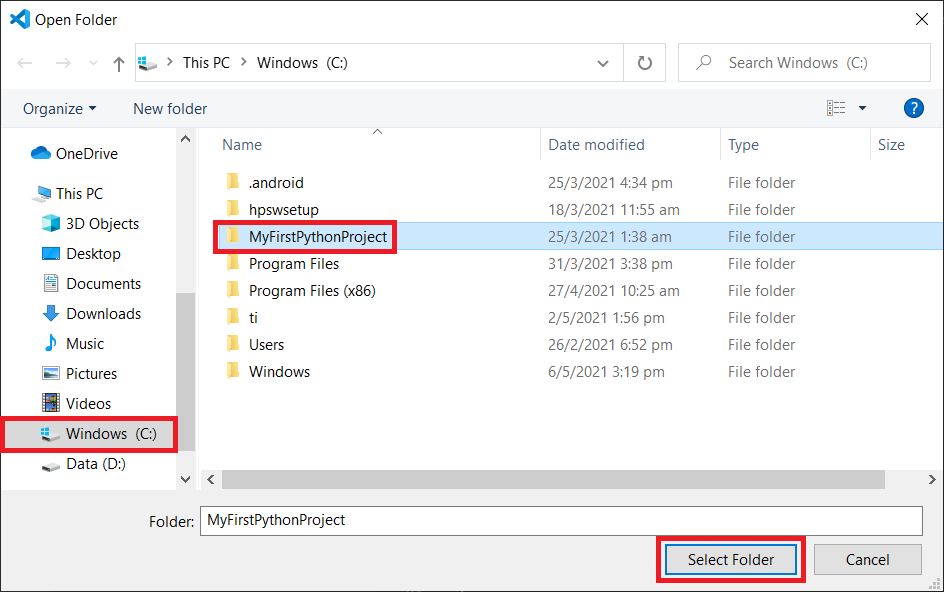


Figure 1.2s: Opening the Project Folder

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.

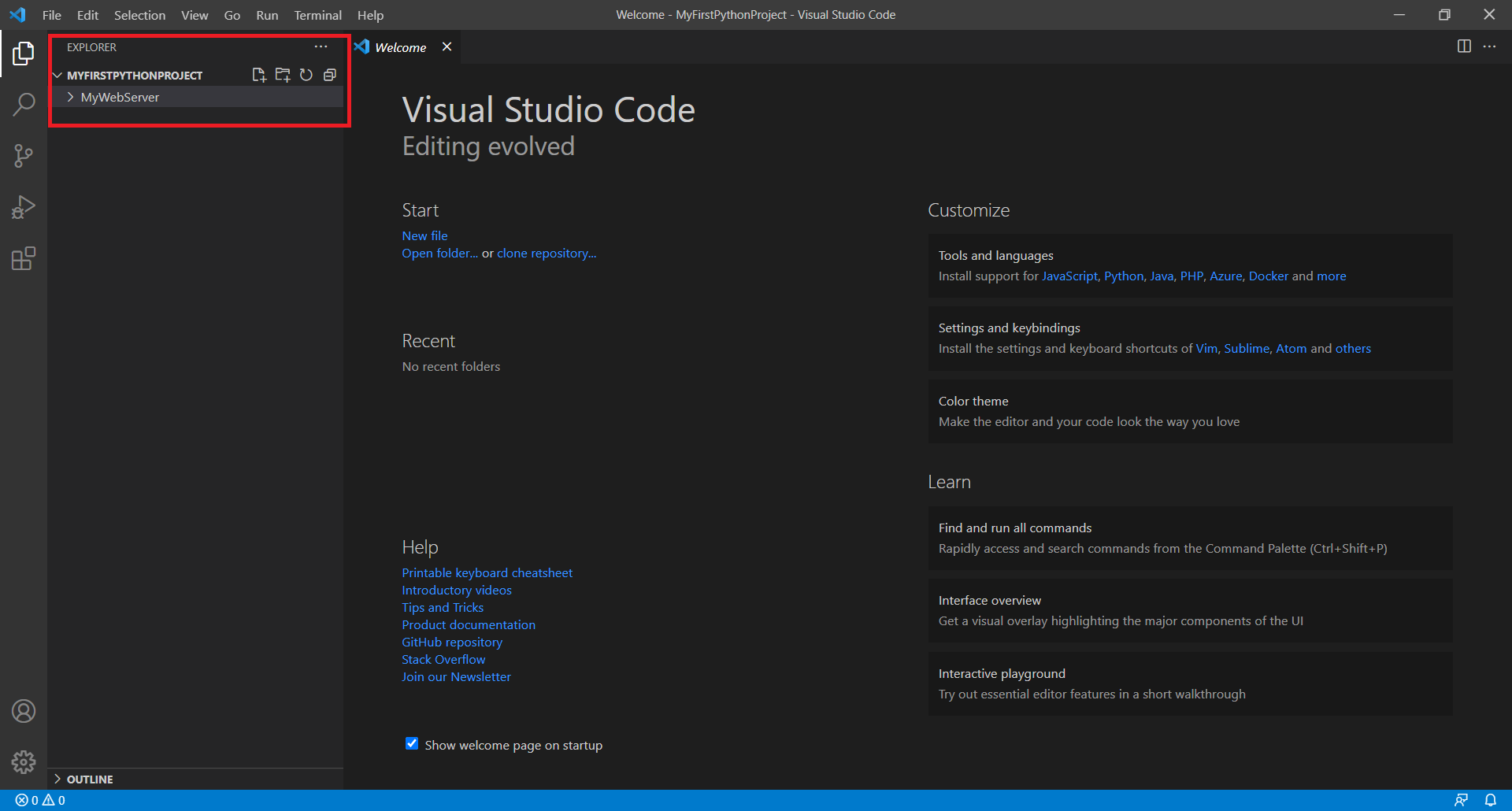


Figure 1.2t: Opening the Project Folder

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.

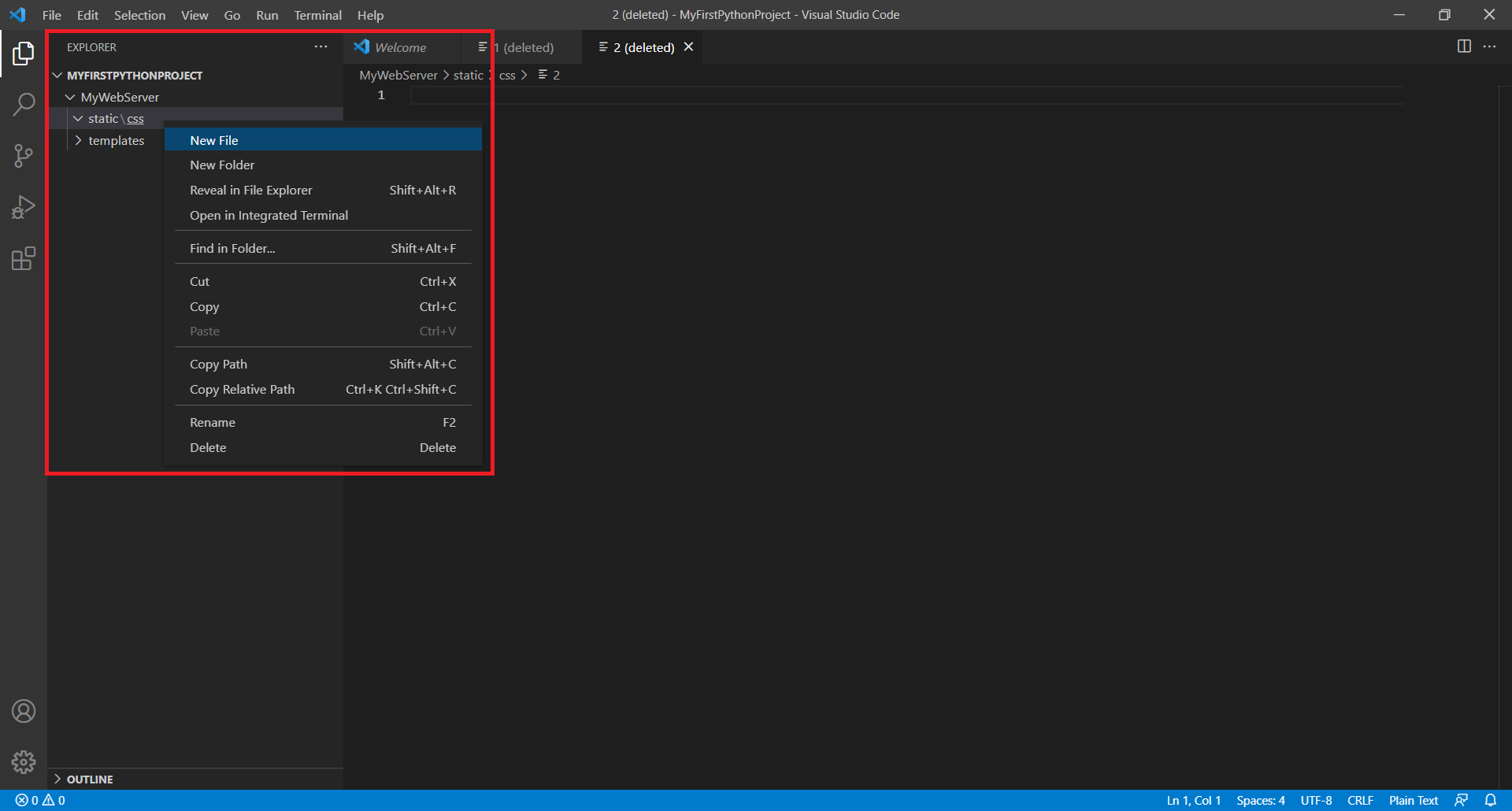


Figure 1.2u: Create a css file

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. **Download** the LightBulb.png file from Blackboard. **Copy** and **paste** the image to the images folder inside static folder created earlier.
2. **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>  <!Provide path to link to styling file, image files and 3rd party javascript library––>  <link rel='stylesheet' href="{{ url\_for('static', filename='css/style.css') }}">  <img id="lightbulb" src="{{url\_for('static', filename='images/LightBulb.png')}}" style="display: none;">  <!Use of Javascript library socket.io.js to connect to the python web server––>  <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'**);  var lightCanvas = document.getElementById("centerCanvas");  var lightContext = lightCanvas.getContext("2d");  var lightBulbImage = document.getElementById("lightbulb");    $('OnButton').click(function(){  socket.emit('ControlUSR0Led','on');  });  $('OffButton').click(function(){  socket.emit('ControlUSR0Led','off');  });    //Image Printing  lightContext.drawImage(lightBulbImage, 175, 110);  //Event triggered when python web server received data from BBBW1  socket.on('Web\_BBBWEvent', function(RxData) {  LightingControl\_AdjustBrightness(RxData.data);  console.log(RxData.data);  });    //LightingControl\_AdjustBrightness Function  //Adjust the transparency level of the yellow circle based on the value received from the Pot Click  function LightingControl\_AdjustBrightness(BrightnessLevel)  {  lightContext.globalAlpha = 1.0;  lightContext.clearRect(0, 0, 400, 300);  lightContext.drawImage(lightBulbImage, 175, 110);  lightContext.globalAlpha = BrightnessLevel;  lightContext.arc(200, 150, 80, 0, 2 \* Math.PI);  lightContext.fillStyle = 'yellow';  lightContext.fill();  lightContext.globalAlpha = 1.0;  lightContext.font = "15px Arial";  lightContext.fillStyle = 'Black';  lightContext.fillText("Estimated Lux Value:", 100, 260);  lightContext.fillText(Math.floor(Math.abs(BrightnessLevel\*400)), 260, 260);  }  });  </script>  </head>  <body>  <center>  <h1>Light Brightness Control System</h1>  <table>  <tr>  <td>  <h5>Controlling the BBBW Board</h5>  <OnButton><button>ON</button></OnButton>  <OffButton><button>OFF</button></OffButton>  </td>  </tr>    <tr>  <td>  <canvas id="centerCanvas" width="400" height="300" style="border:1px solid #d3d3d3;" >  Your browser does not support the HTML5 canvas tag.  </canvas>  </td>  </tr>  </table>  <h5>Copyright @ EGE205 Connected System Project Design</h5>  <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, **“192.168.86.64:5000”** is used it is observed that the following user interface is printed on the browser as shown in Figure below.

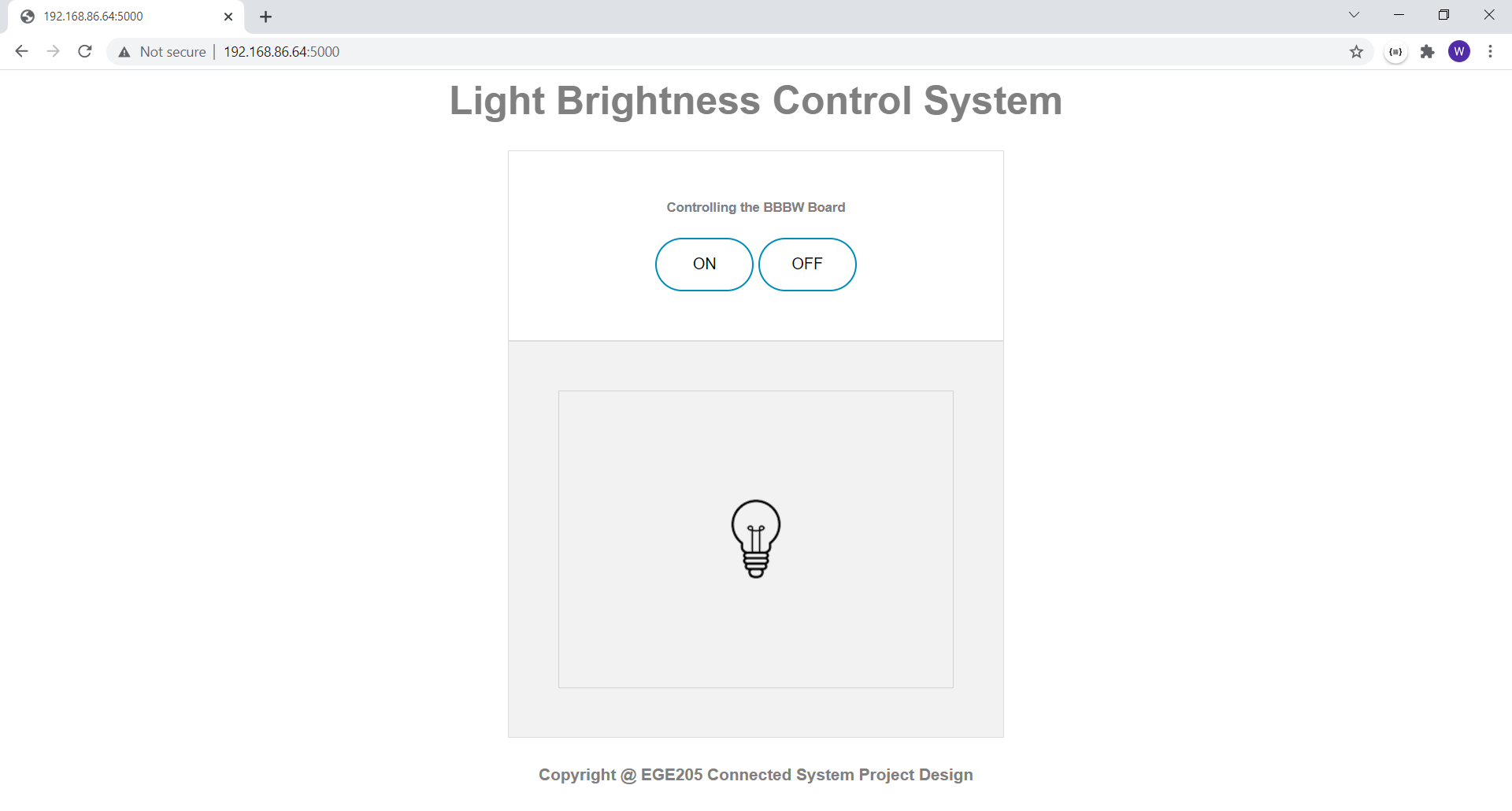


Figure 1.2v: Accessing the Web Server on PC

## Setting up the BBBW Board Web Client

**Setting up the BBBW Web Client**

1. **Connect** the Pot Click to the mikroBUS cape and BBBW board as shown in the Figure below.

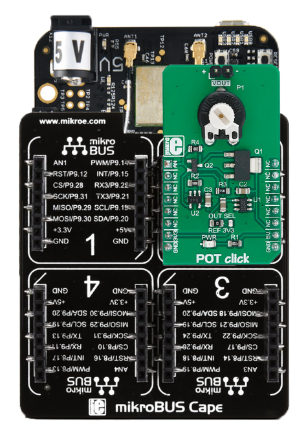


Figure 1.3a: Connecting Pot Click to mikroBUS Cape and BBBW Board

**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. **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 1.3b: 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 1.3c: Ensuring PythonLibrary Folder Is Created

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

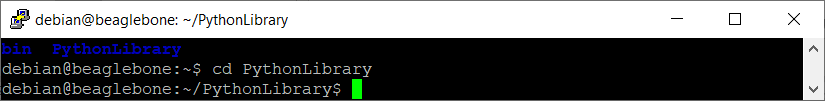


Figure 1.3d: Accessing the PythonLibrary Folder

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.

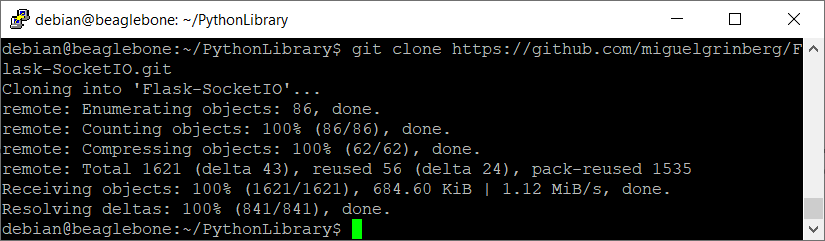


Figure 1.3e: Cloning the Flask-SocketIO Python Library

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

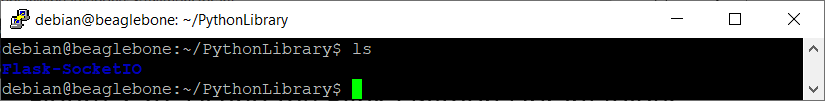


Figure 1.3f: Listing the Flask-SocketIO Python Library Folder

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

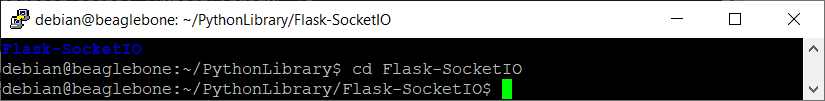


Figure 1.3g: Accessing the Flask-SocketIO Folder

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.

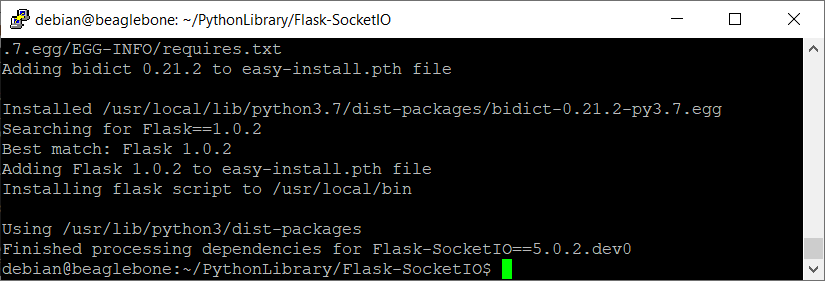


Figure 1.3h: Completion of Flask-SocketIO Python Library Installation

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

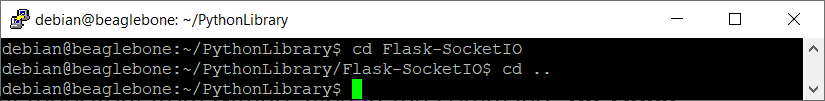


Figure 1.3i: Exit to Flask-SocketIO Folder

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.

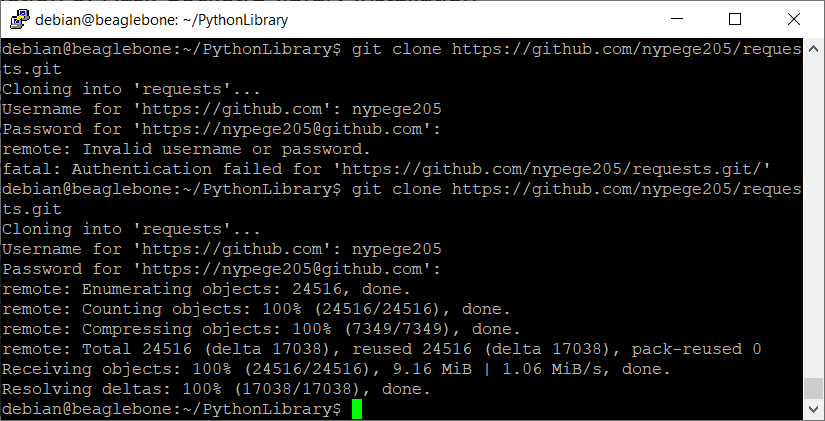


Figure 1.3j: Cloning the requests Python Library

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

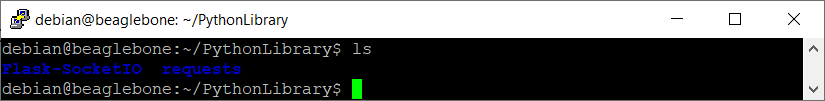


Figure 1.3k: Listing the requests Python Library Folder

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

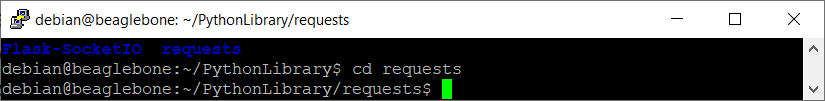


Figure 1.3l: Accessing the Flask-SocketIO folder

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.

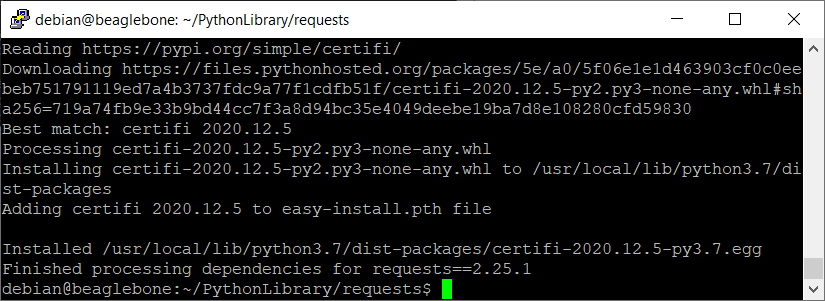


Figure 1.3m: Completion of requests Python Library Installation

**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 1.3o: 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.

|  |
| --- |
| import socketio  import time  import Adafruit\_BBIO.GPIO as GPIO  import Adafruit\_BBIO.ADC as ADC  GPIO.setup('USR0', GPIO.OUT)  ADC.setup()  sio = socketio.Client()  @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  OldDigitalValue = 0  while True:  try:  NewDigitalValue = ADC.read("P9\_37")  print("Digital Value: %f" % (NewDigitalValue))  if(abs(NewDigitalValue - OldDigitalValue) > 0.1):  sio.emit('BBBWEvent', {'data': NewDigitalValue})  print('Data sent!')  OldDigitalValue = NewDigitalValue  except:  print('Unable to transmit data.')  pass  time.sleep(0.5) |

1. **Click** on the “Run” button located beside the Menu Tab to execute the “**WebClient.py**” file.

## Controlling and Monitoring BBBW Board Web Client via a PC Web Server

1. **Ensure** that the web server on PC is running.
2. **Ensure** that the BBBW board is running.
3. **Turn** the knob on the Pot Click and **observe** that the data (brightness color and lux value) from the BBBW board is streamed into the web server on PC as shown in the Figure below.

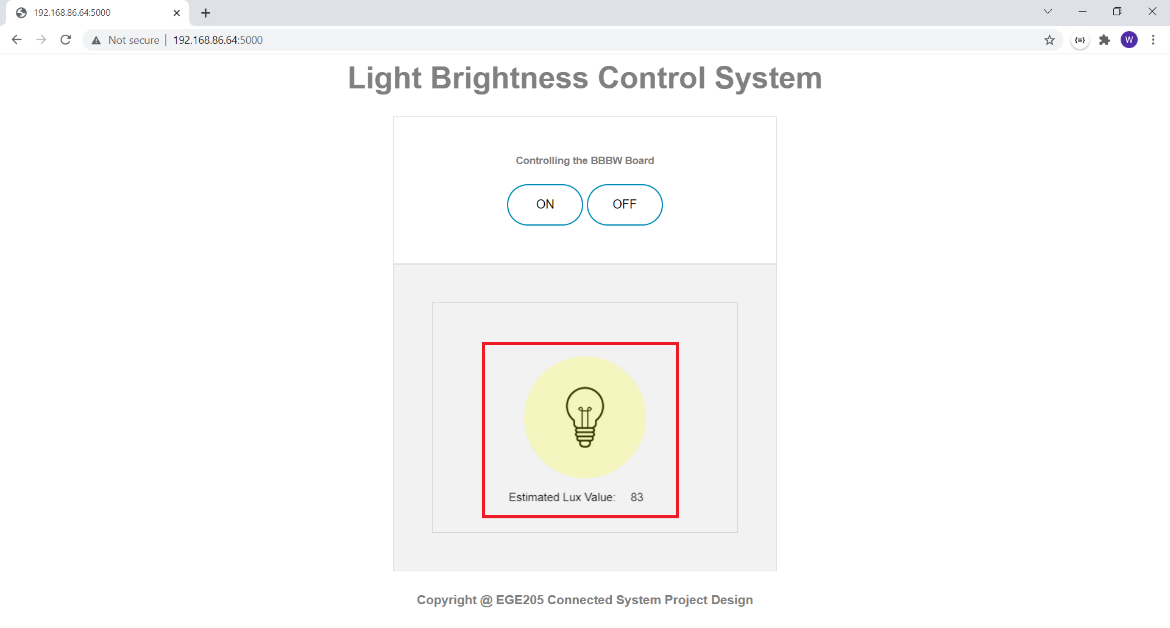


Figure 1.4a: Data Streamed into the Web Server on PC

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 the BBBW board is turned on and off.

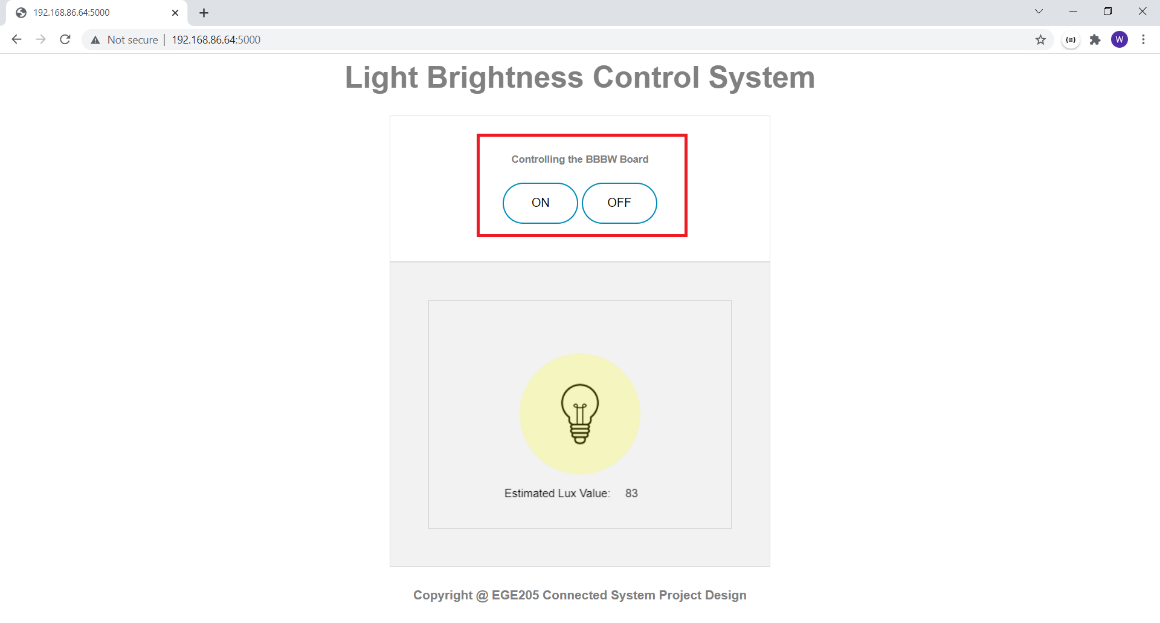


Figure 1.4b: Controlling the BBBW Board from the Web Server on PC

## Tinkering Time

1. Together with a classmate or two, **think** of a simple application that can stream in data from the click board you have learnt and display it through the web server webpage.
2. **Connect** the selected clicks to the mikroBUS cape and BBBW board.
3. **Create** a python file in Cloud9 IDE and start writing your code.
4. **Present** your complete work to your lecturer for advice.
5. **Share** your work with your other classmates and teach them how you do it if they are interested.

*Congratulations! You have successfully completed the Lab5b. Good job! You are now ready to develop a bigger scale connected system!*