

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

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

**Laboratory No**. : Lab 6b

**Laboratory Title** : A Practical Example of a Connected System – The Connected System Classroom

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

System in a classroom

**Hardware Boards** : BBBW Board x4

Motion Click x1

Pot Click x1

Analog Key Click x1

Reed Click x1

**Contents**

1. Setting Up a Connected System using Multiple BeagleBone Black Wireless (BBBW) Boards
   1. Understanding of Connected Systems Classroom
   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 how a connected system classroom works! 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 working connected system classroom. Good Luck! |

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

## Understanding of Connected System Classroom

**Connected System Classroom** consists of a central PC web server and 4 BBBW boards with each having a click board on it to collect data. They are all inter-connected through Wi-Fi to the access point and have been assigned a unique IP address.

The **first BBBW board** uses the **Motion Click** to detect the present of movement in the classroom. Upon detecting the present of movement, data will be sent to the PC web server dashboard to toggle a motion detection image. The **second BBBW board** uses the **Pot Click** to allow the lighting in the classroom to be adjusted. The brightness adjustment from the Pot Click is reflected real-time on the PC web server dashboard through a yellow filled circle with different transparency value.

The **third BBBW board** uses the **Analog Key Click** to capture the key pressed to unlock the door for accessing to the classroom. For every key pressed, the key number is sent to the PC web server to perform the matching analysis. Once the correct key is matched in this case the key “6”, the door will be unlocked automatically. This action is illustrated by a simple door unlocked animation shown in the PC web server.

The **fourth BBBW board** uses the **Reed Click** to perform door detection. The Reed Click constantly check on the present or absent of the magnet and relay the respective data to the PC web server only after the door is unlocked. When a magnet is brought closer to the Reed Click, it means the door is closed and the respective data is sent to the PC web server dashboard to display an image showing the door is close.

The setup of a connected system classroom using BBBW boards is shown in the Figure below.

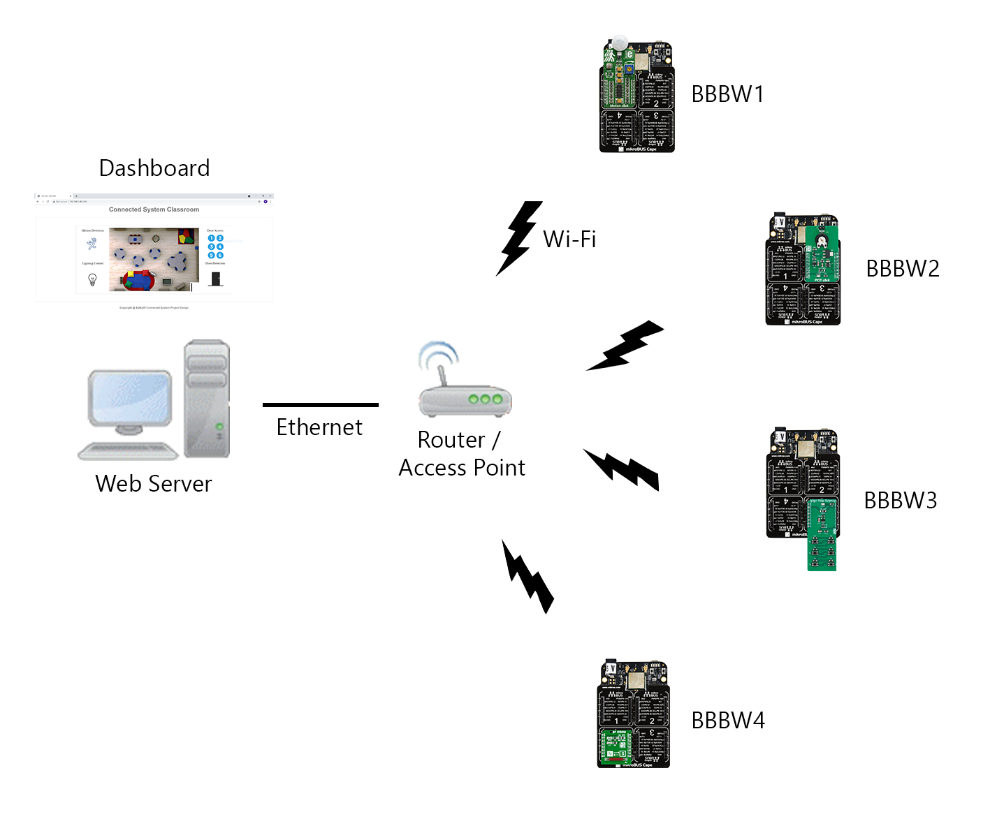


Figure 1.1a: Connected System Classroom setup using BBBW Boards

## 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.

* **ConnectedSystemClassroom**
  + **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.

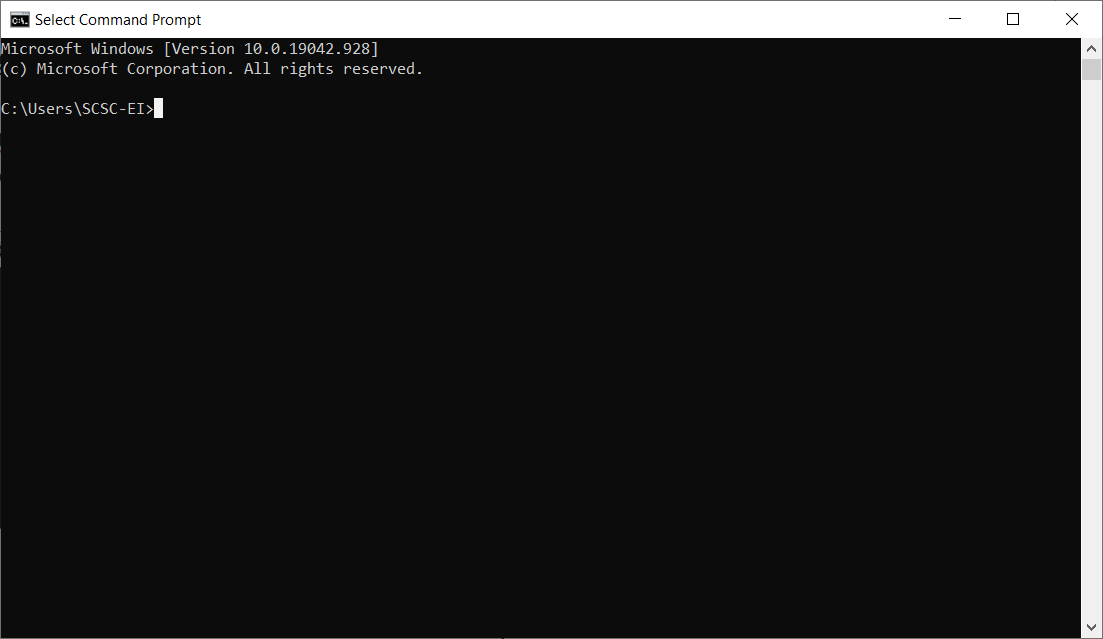


Figure 1.2a: Command Prompt

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.

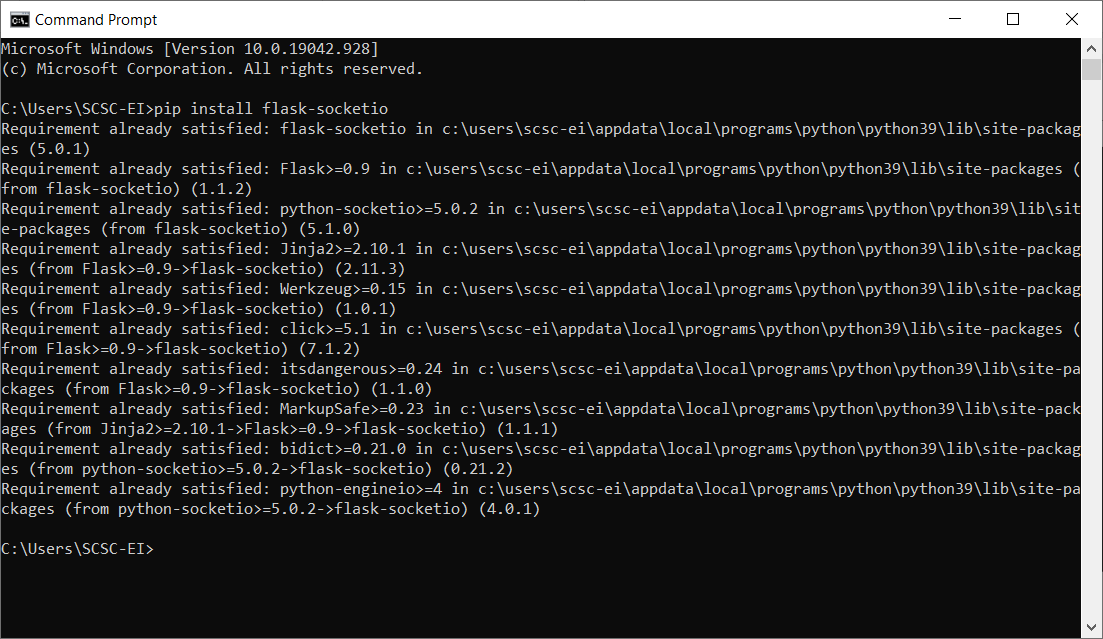


Figure 1.2b: Command Prompt

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.

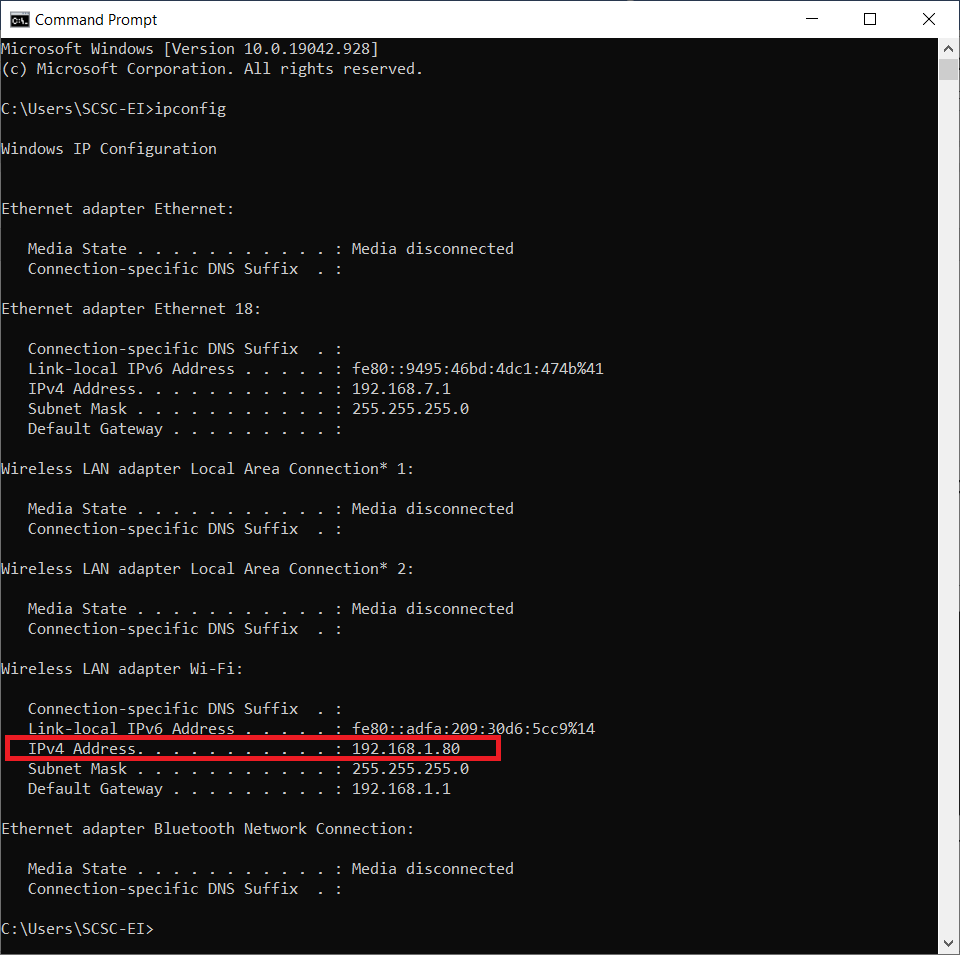


Figure 1.2c: IP Address Assigned to the PC

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.

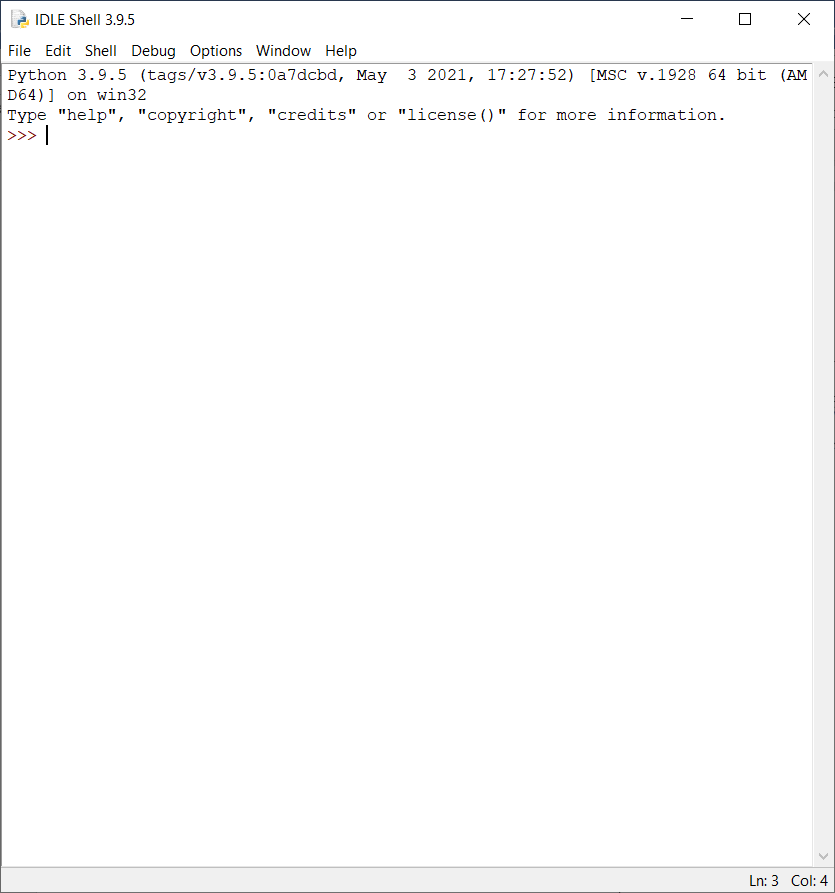


Figure 1.2d: Python IDLE Shell

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.

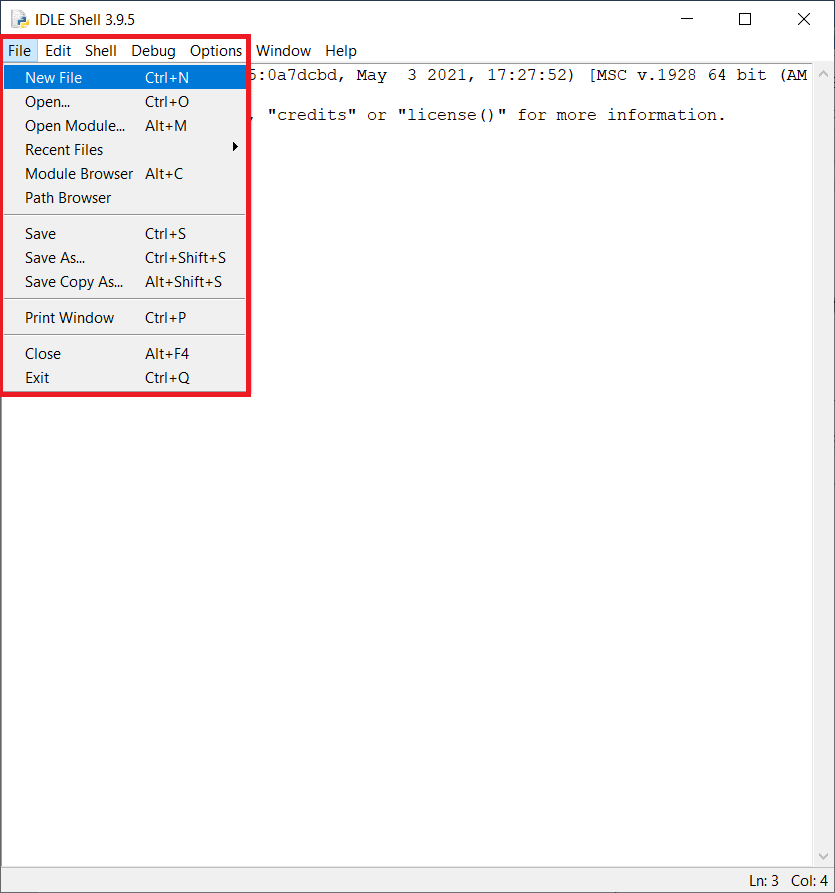


Figure 1.2e: Create new python file

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 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**”.

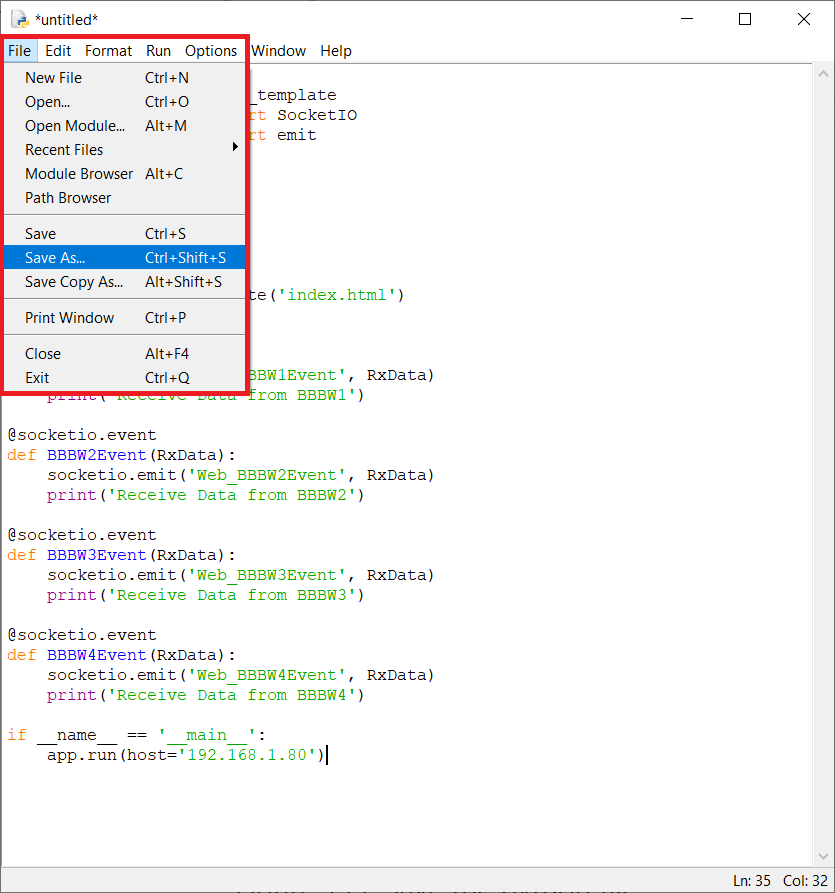


Figure 1.2f: Save the python file

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.

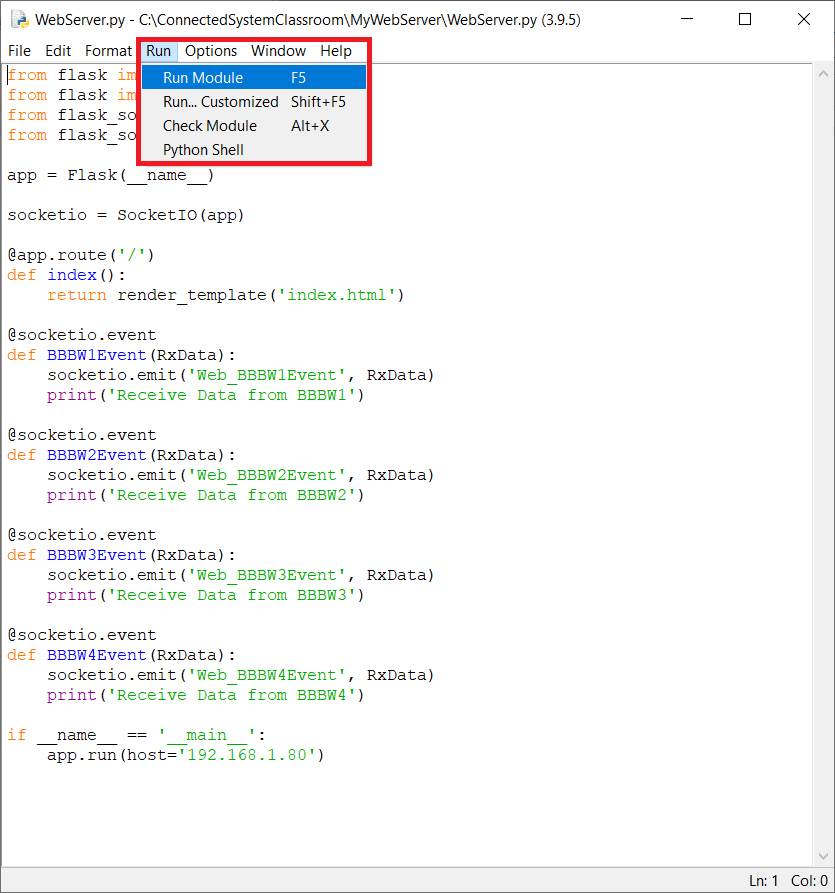


Figure 1.2g: Executing the python file

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

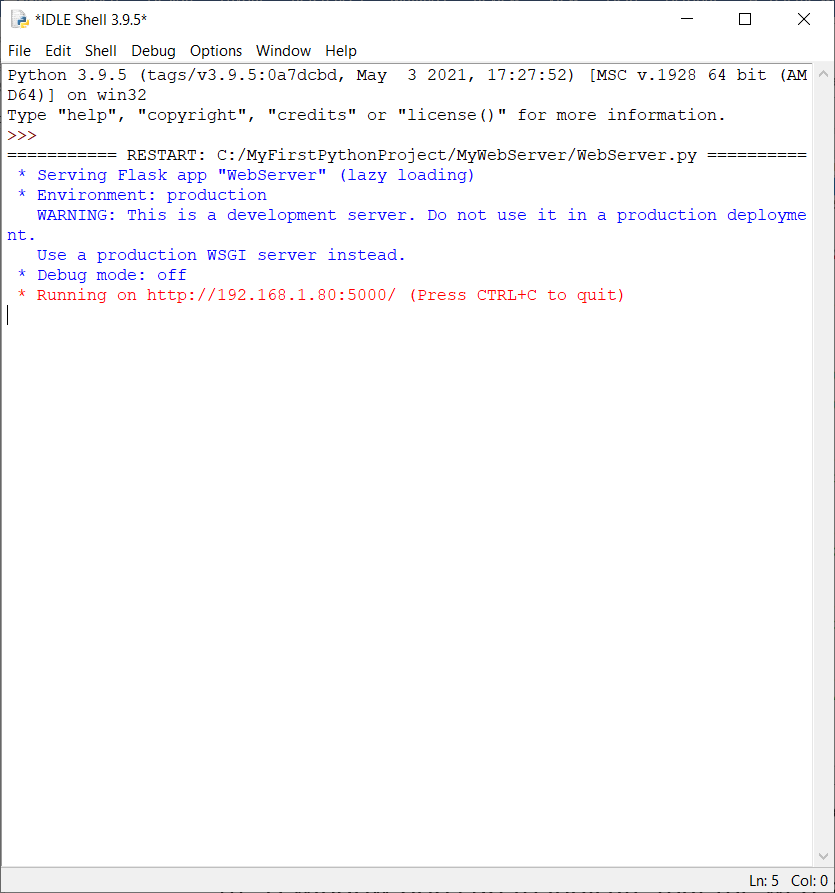


Figure 1.2h: Web Server is Running

**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.

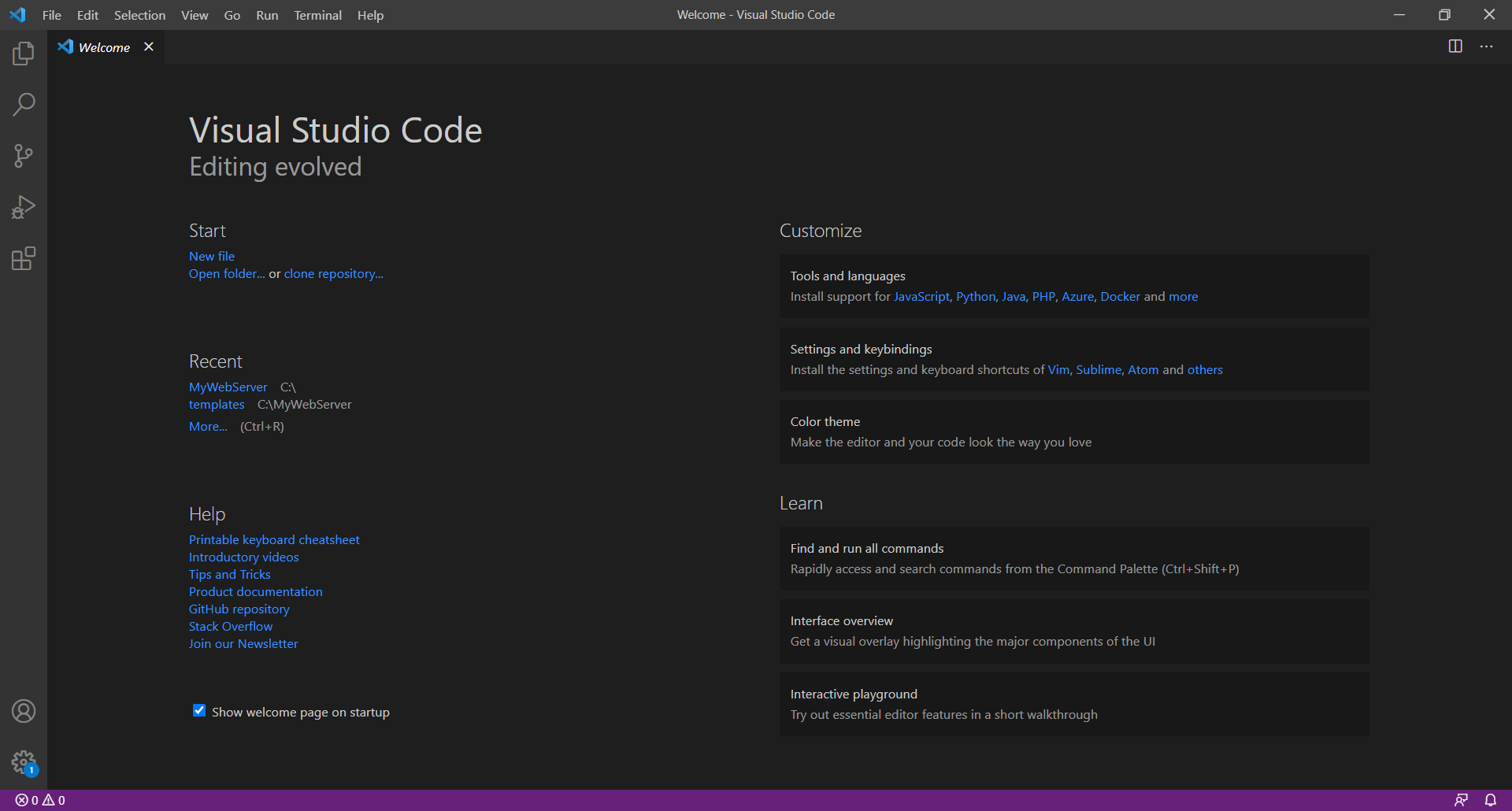


Figure 1.2i: Launching the Visual Studio Code (VS Code) Software

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

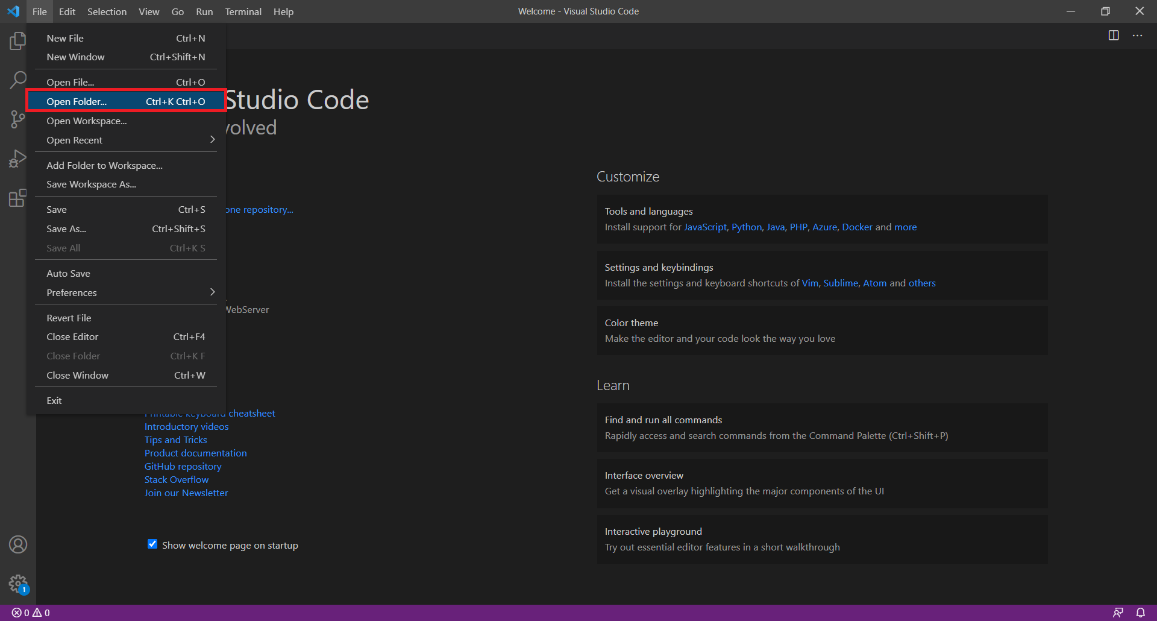


Figure 1.2j: Opening the Project Folder

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

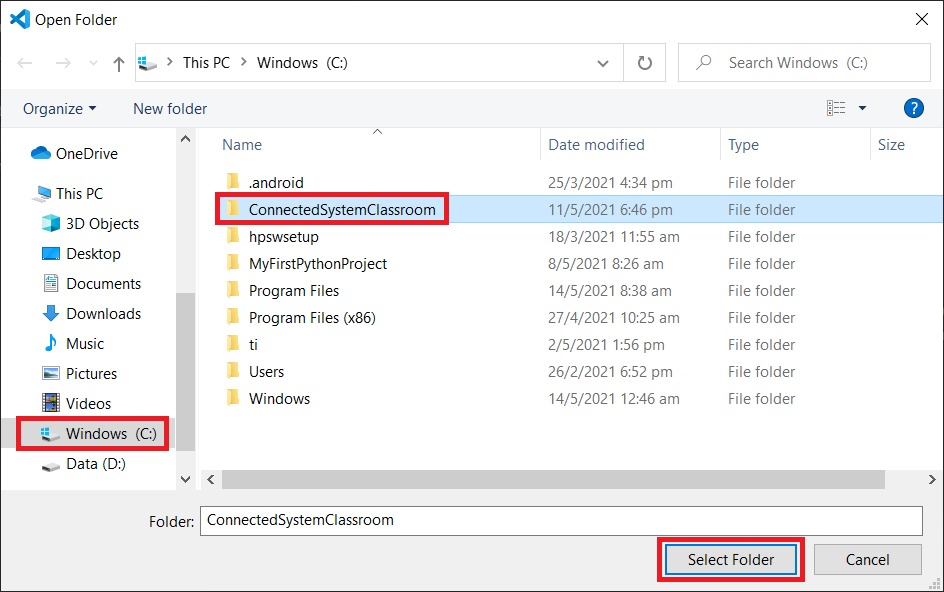


Figure 1.2k: Opening the Project Folder

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

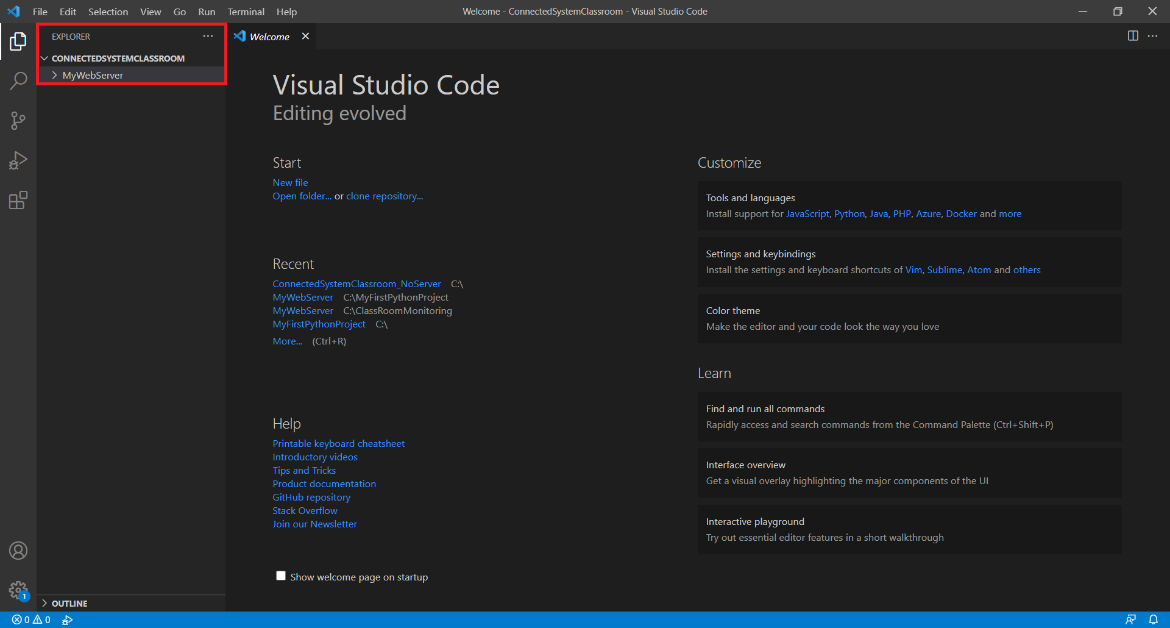


Figure 1.2l: Opening the Project Folder

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.

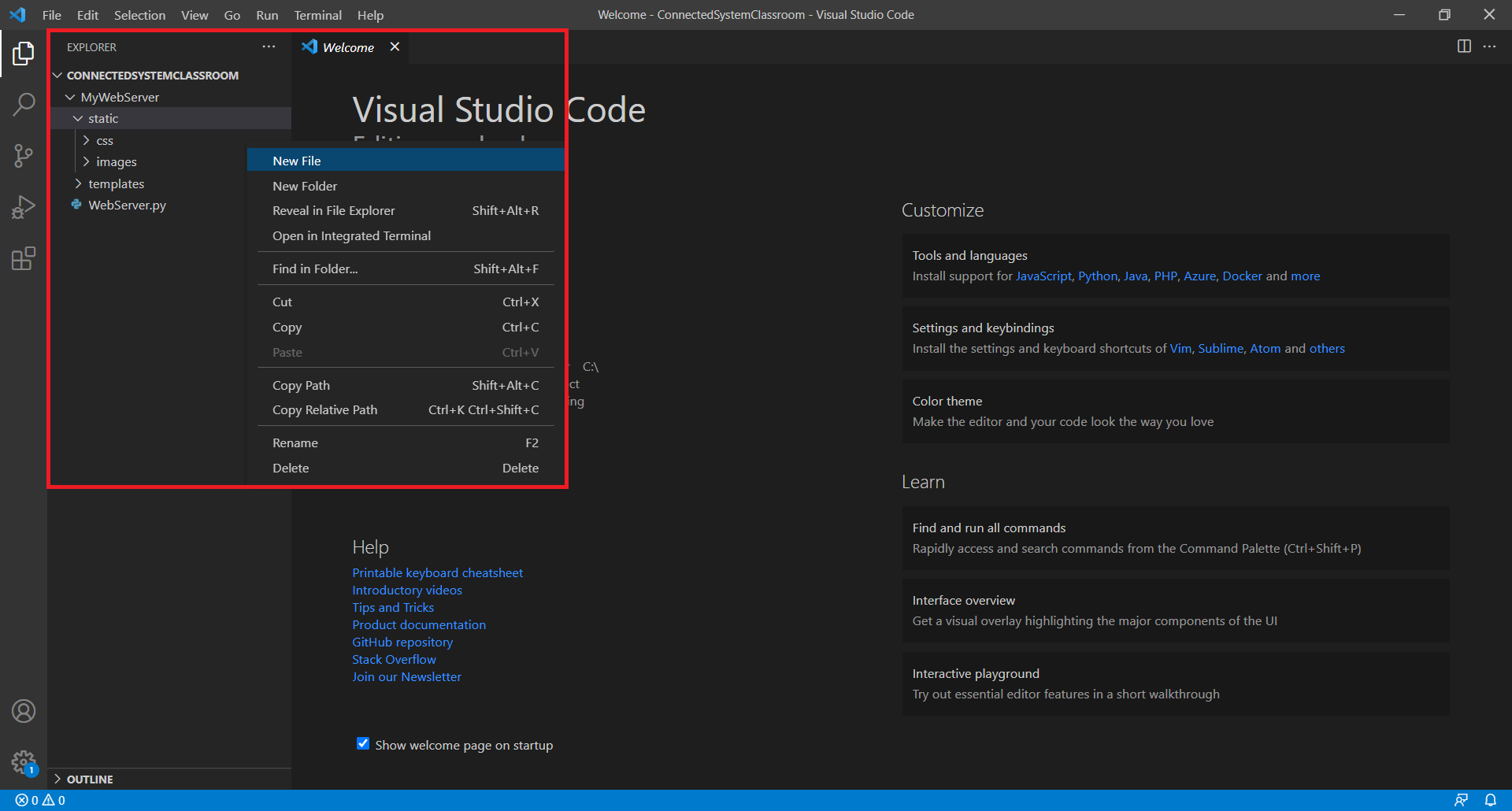


Figure 1.2m: Create a css file

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;  } |

1. **Download** the lab6b\_images.zip file from Blackboard. **Unzip** the file and **copy** and **paste** all the 7 images 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="floorplan" src="{{url\_for('static', filename='images/Floorplan.png')}}" style="display: none;">  <img id="motionsensoriconblue" src="{{url\_for('static', filename='images/MotionSensorIconBlue.png')}}" style="display: none;">  <img id="motionsensoriconred" src="{{url\_for('static', filename='images/MotionSensorIconRed.png')}}" style="display: none;">  <img id="lightbulb" src="{{url\_for('static', filename='images/LightBulb.png')}}" style="display: none;">  <img id="keypad" src="{{url\_for('static', filename='images/Keypad.png')}}" style="display: none;">  <img id="dooropen" src="{{url\_for('static', filename='images/DoorOpen.png')}}" style="display: none;">  <img id="doorclose" src="{{url\_for('static', filename='images/DoorClose.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()  {  //Initialization and declaration of global variable  var socket = io.connect(**'http://192.168.X.X:5000'**);  var floorPlanCanvas = document.getElementById("centerCanvas");  var floorPlanContext = floorPlanCanvas.getContext("2d");  var floorPlanImage = document.getElementById("floorplan");  var motionCanvas = document.getElementById("topLeftCanvas");  var motionContext = motionCanvas.getContext("2d");  var motionSensorIconBlueImage = document.getElementById("motionsensoriconblue");  var motionSensorIconRedImage = document.getElementById("motionsensoriconred");    var lightCanvas = document.getElementById("bottomLeftCanvas");  var lightContext = lightCanvas.getContext("2d");  var lightBulbImage = document.getElementById("lightbulb");    var doorAccessCanvas = document.getElementById("topRightCanvas");  var doorAccessContext = doorAccessCanvas.getContext("2d");  var keyPadImage = document.getElementById("keypad");  var doorDetectCanvas = document.getElementById("bottomRightCanvas");  var doorDetectContext = doorDetectCanvas.getContext("2d");  var doorOpenImage = document.getElementById("dooropen");  var doorCloseImage = document.getElementById("doorclose");    var BlinkingFlagStatus = 0;  var DoorOpenTurning;  var DoorRotateCount = 0;  var DoorOpenCloseStatus = 0;    //Image Printing  floorPlanContext.drawImage(floorPlanImage, 0, 0);  motionContext.drawImage(motionSensorIconBlueImage, 70, 37);  lightContext.drawImage(lightBulbImage, 75, 50);  doorAccessContext.drawImage(keyPadImage, 50, 0);  doorDetectContext.drawImage(doorCloseImage, 50, 33);    //Door Panel Printing  floorPlanContext.beginPath();  floorPlanContext.lineWidth = 5;  floorPlanContext.lineCap = "round";  floorPlanContext.moveTo(550,420);  floorPlanContext.lineTo(550,375);  floorPlanContext.stroke();    //Event triggered when python web server received data from BBBW1  socket.on('Web\_BBBW1Event', function(RxData) {  $('bbbw1').text(RxData.data);  MotionDetection\_BlinkImage();  });    //Event triggered when python web server received data from BBBW2  socket.on('Web\_BBBW2Event', function(RxData) {  $('bbbw2').text(RxData.data);  LightingControl\_AdjustBrightness(RxData.data);  });  //Event triggered when python web server received data from BBBW3  socket.on('Web\_BBBW3Event', function(RxData) {  $('bbbw3').text(RxData.data);  DoorAccess\_KeyCheck(RxData.data);  });    //Event triggered when python web server received data from BBBW4  socket.on('Web\_BBBW4Event', function(RxData) {  $('bbbw4').text(RxData.data);  if(DoorOpenCloseStatus == 1)  {  DoorDetection\_OpenCloseCheck(RxData.data);  }  });    //MotionDetection\_BlinkImage Function  //Toggle between motionSensorIconBlueImage & motionSensorIconRedImage when motion is detected  function MotionDetection\_BlinkImage()  {  //Clear the existing image  motionContext.globalAlpha = 1.0;  motionContext.clearRect(0, 0, 200, 155);    if (BlinkingFlagStatus == 0)  {  BlinkingFlagStatus = 1;  motionContext.drawImage(motionSensorIconRedImage, 70, 37);  }  else  {  BlinkingFlagStatus = 0;  motionContext.drawImage(motionSensorIconBlueImage, 70, 37);  }  }    //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, 200, 155);  lightContext.drawImage(lightBulbImage, 75, 50);  lightContext.globalAlpha = BrightnessLevel;  lightContext.arc(100, 70, 40, 0, 2 \* Math.PI);  lightContext.fillStyle = 'yellow';  lightContext.fill();  }    //DoorAccess\_KeyCheck Function  //Highlight the individual number key based on the value received from the Analog Key Click  function DoorAccess\_KeyCheck(KeyPressed)  {  doorAccessContext.globalAlpha = 1.0;  doorAccessContext.clearRect(0, 0, 200, 155);  doorAccessContext.drawImage(keyPadImage, 50, 0);  if(KeyPressed == 1)  {  doorAccessContext.beginPath();  doorAccessContext.globalAlpha = 0.5;  doorAccessContext.arc(72, 20, 24, 0, 2 \* Math.PI);  doorAccessContext.fillStyle = 'white';  doorAccessContext.fill();  }  if(KeyPressed == 2)  {  doorAccessContext.beginPath();  doorAccessContext.globalAlpha = 0.5;  doorAccessContext.arc(127, 20, 24, 0, 2 \* Math.PI);  doorAccessContext.fillStyle = 'white';  doorAccessContext.fill();  }  if(KeyPressed == 3)  {  doorAccessContext.beginPath();  doorAccessContext.globalAlpha = 0.5;  doorAccessContext.arc(72, 75, 24, 0, 2 \* Math.PI);  doorAccessContext.fillStyle = 'white';  doorAccessContext.fill();  }  if(KeyPressed == 4)  {  doorAccessContext.beginPath();  doorAccessContext.globalAlpha = 0.5;  doorAccessContext.arc(127, 75, 24, 0, 2 \* Math.PI);  doorAccessContext.fillStyle = 'white';  doorAccessContext.fill();  }  if(KeyPressed == 5)  {  doorAccessContext.beginPath();  doorAccessContext.globalAlpha = 0.5;  doorAccessContext.arc(72, 130, 24, 0, 2 \* Math.PI);  doorAccessContext.fillStyle = 'white';  doorAccessContext.fill();  }  if(KeyPressed == 6)  {  doorAccessContext.beginPath();  doorAccessContext.globalAlpha = 0.5;  doorAccessContext.arc(127, 130, 24, 0, 2 \* Math.PI);  doorAccessContext.fillStyle = 'white';  doorAccessContext.fill();    floorPlanContext.translate(550, 420);  DoorOpenTurning = setInterval(DoorOpenTurningAnimation, 50);  }  }    //DoorOpenTurningAnimation Function  function DoorOpenTurningAnimation()  {  floorPlanContext.globalAlpha = 1.0;  floorPlanContext.clearRect(-10, 0, 60, -53);    floorPlanContext.beginPath();  floorPlanContext.lineWidth = 5;  floorPlanContext.lineCap = "round";  floorPlanContext.moveTo(0,0);  floorPlanContext.lineTo(0,-45);  floorPlanContext.stroke();  floorPlanContext.rotate(0.1);  DoorRotateCount++;  if (DoorRotateCount == 17)  {  DoorRotateCount = 0;  floorPlanContext.rotate(-1.7);  floorPlanContext.translate(-550, -420);  floorPlanContext.globalAlpha = 1.0;  doorDetectContext.clearRect(0, 0, 200, 155);  doorDetectContext.drawImage(doorOpenImage, 50, 28);  DoorOpenCloseStatus = 1;  clearInterval(DoorOpenTurning);  }  }    //DoorDetection\_OpenCloseCheck Function  function DoorDetection\_OpenCloseCheck(DoorDetectionStatus)  {  if (DoorDetectionStatus == 1)  {  doorDetectContext.globalAlpha = 1.0;  doorDetectContext.clearRect(0, 0, 200, 155);  doorDetectContext.drawImage(doorCloseImage, 50, 33);  }  else  {  doorDetectContext.globalAlpha = 1.0;  doorDetectContext.clearRect(0, 0, 200, 155);  doorDetectContext.drawImage(doorOpenImage, 50, 28);  }  }  });  </script>  </head>  <body>  <center>  <h1>Connected System Classroom</h1>  <table>  <tr>  <td>  <h3>Motion Detection</h3>  <canvas id="topLeftCanvas" width="200" height="155">  Your browser does not support the HTML5 canvas tag.  </canvas>  </td>  <td rowspan = "2">  <canvas id="centerCanvas" width="620" height="440" style="border:1px solid #d3d3d3;" >  Your browser does not support the HTML5 canvas tag.  </canvas>  </td>  <td>  <h3>Door Access</h3>  <canvas id="topRightCanvas" width="200" height="155">  Your browser does not support the HTML5 canvas tag.  </canvas>  </td>  </tr>    <tr>  <td>  <h3>Lighting Control</h3>  <canvas id="bottomLeftCanvas" width="200" height="155">  Your browser does not support the HTML5 canvas tag.  </canvas>  </td>  <td>  <h3>Door Detection</h3>  <canvas id="bottomRightCanvas" width="200" height="155" >  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, **“192.168.1.80:5000”** is used it is observed that the following user interface is printed on the browser as shown in Figure below.

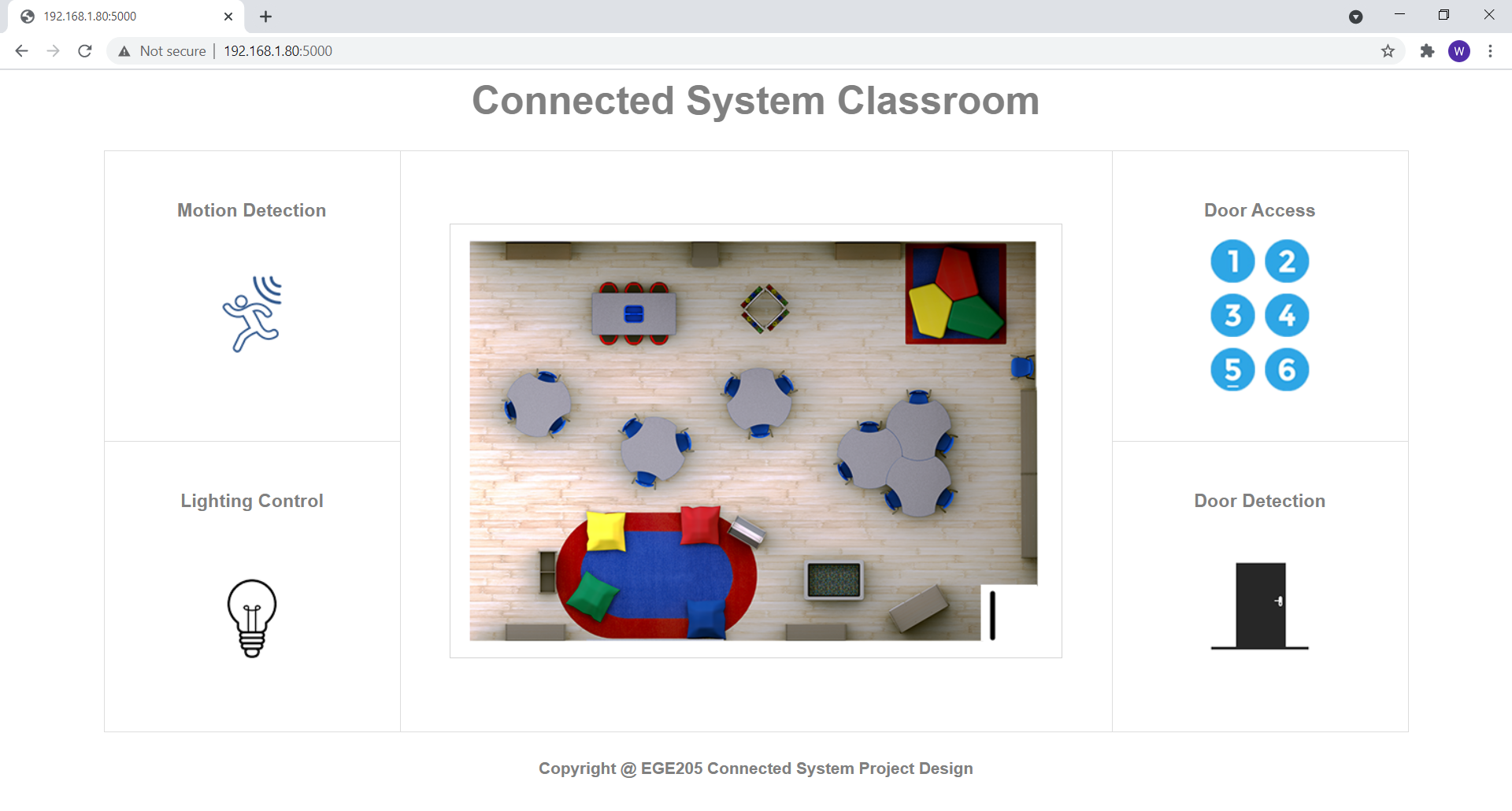


Figure 1.2n: Accessing the Web Server on PC

## Setting up Multiple Web Clients in Multiple BBBW Boards

**Setting up the Individual BBBW Boards**

1. **Choose** **ONLY** 1 of the BBBW boards setup as shown below to be implemented in your BBBW board. **Discuss** among the team members to avoid using the same setup below.

**For BBBW1 Board**

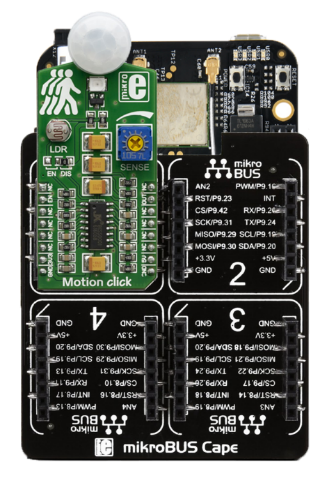


Figure 1.3a: Connecting Motion Click to mikroBUS Cape and BBBW Board

**For BBBW2 Board**

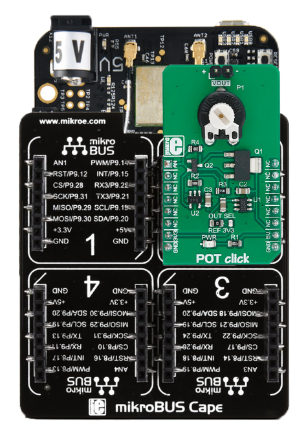


Figure 1.3b: Connecting Pot Click to mikroBUS Cape and BBBW Board

**For BBBW3 Board**

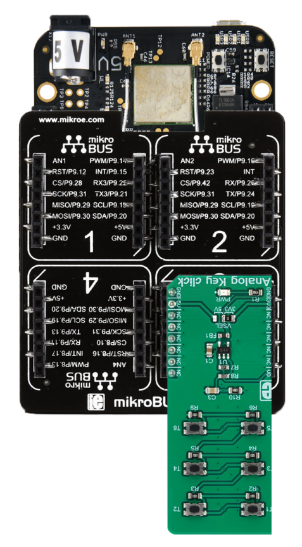


Figure 1.3c: Connecting Analog Key Click to mikroBUS Cape and BBBW Board

**For BBBW4 Board**

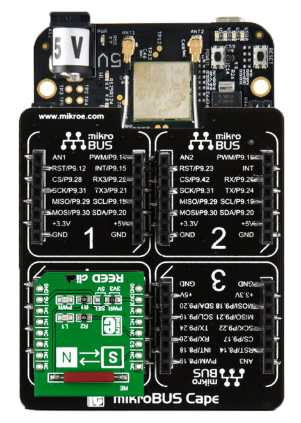


Figure 1.3d: Connecting Reed Click to mikroBUS Cape and BBBW Board

**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. **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 1.3e: 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 1.3f: Ensuring PythonLibrary Folder Is Created

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

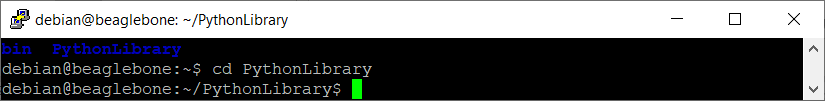


Figure 1.3g: Accessing the PythonLibrary Folder

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.

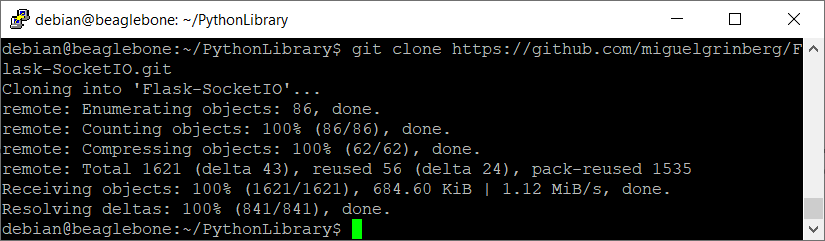


Figure 1.3h: Cloning the Flask-SocketIO Python Library

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

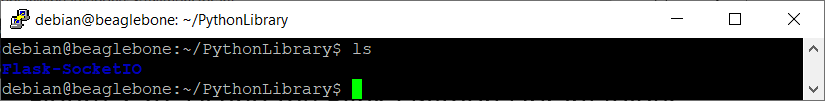


Figure 1.3i: Listing the Flask-SocketIO Python Library Folder

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

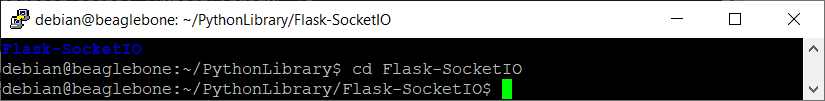


Figure 1.3j: Accessing the Flask-SocketIO Folder

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.

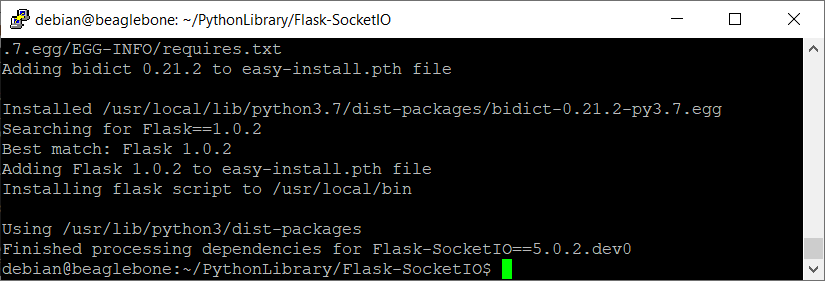


Figure 1.3k: Completion of Flask-SocketIO Python Library Installation

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

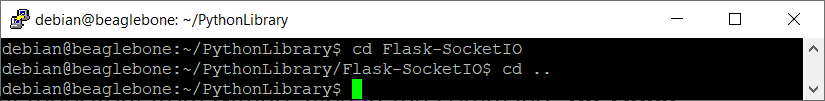


Figure 1.3l: Exit to Flask-SocketIO Folder

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.

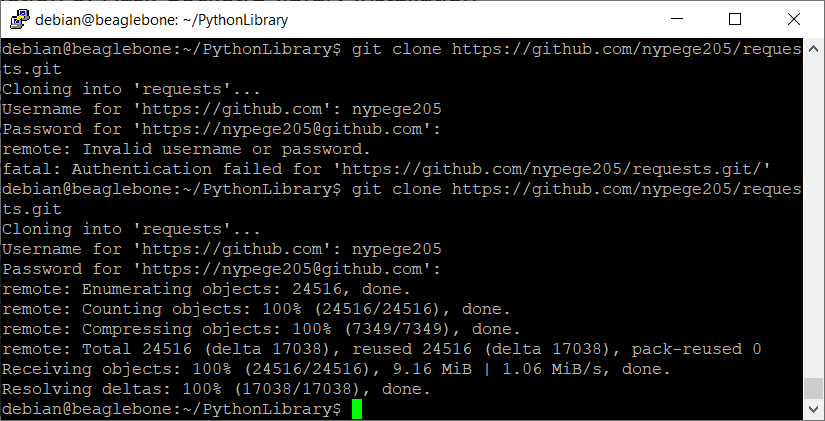


Figure 1.3m: Cloning the requests Python Library

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

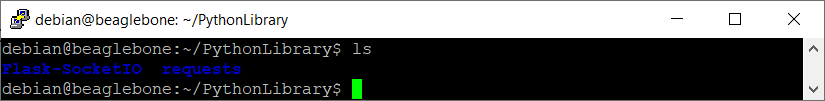


Figure 1.3n: Listing the requests Python Library Folder

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

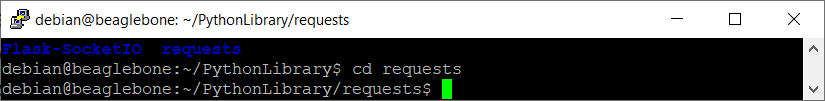


Figure 1.3o: Accessing the Flask-SocketIO folder

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.

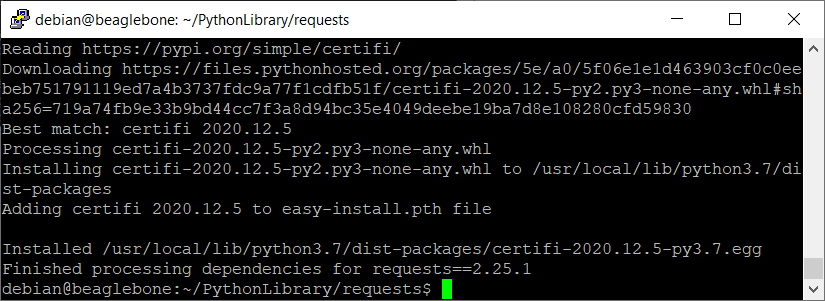


Figure 1.3p: Completion of requests Python Library Installation

**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 1.3q: 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 according to the board setup you have chosen earlier to be implemented in your BBBW board.

**For BBBW1 Board**

|  |
| --- |
| import socketio  import time  import Adafruit\_BBIO.GPIO as GPIO  sio = socketio.Client()  GPIO.setup("P9\_15", GPIO.IN)  @sio.event  def connect():  print('Connection established.')  @sio.event  def disconnect():  print('Disconnected from server.')  while True:  try:  sio.connect(**'http://192.168.X.X:5000'**)  break  except:  print("Try to connect to the server.")  pass    MotionDetectionStatus = 0  while True:  try:  MotionDetectionStatus = GPIO.input("P9\_15")  if MotionDetectionStatus:  sio.emit('BBBW1Event', {'data': MotionDetectionStatus})  print('Data sent!')  print("Motion is Detected")  else:  print("No Motion is Detected")  except:  print('Unable to transmit data.')  pass  time.sleep(0.5) |

**For BBBW2 Board**

|  |
| --- |
| import socketio  import time  import Adafruit\_BBIO.ADC as ADC  sio = socketio.Client()  ADC.setup()  @sio.event  def connect():  print('Connection established.')  @sio.event  def disconnect():  print('Disconnected from server.')  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('BBBW2Event', {'data': NewDigitalValue})  print('Data sent!')  OldDigitalValue = NewDigitalValue  except:  print('Unable to transmit data.')  pass  time.sleep(0.5) |

**For BBBW3 Board**

|  |
| --- |
| import socketio  import time  import Adafruit\_BBIO.ADC as ADC  sio = socketio.Client()  ADC.setup()  @sio.event  def connect():  print('Connection established.')  @sio.event  def disconnect():  print('Disconnected from server.')  while True:  try:  sio.connect(**'http://192.168.X.X:5000'**)  break  except:  print("Try to connect to the server.")  pass  KeyPressed = 0  while True:  try:  DigitalValue = ADC.read("P9\_40")  if DigitalValue < 0.100:  print("No Key is Pressed")  KeyPressed = 0  elif DigitalValue < 0.175:  print("T6 Key is Pressed")  KeyPressed = 6  elif DigitalValue < 0.350:  print("T5 Key is Pressed")  KeyPressed = 5  elif DigitalValue < 0.525:  print("T4 Key is Pressed")  KeyPressed = 4  elif DigitalValue < 0.700:  print("T3 Key is Pressed")  KeyPressed = 3  elif DigitalValue < 0.875:  print("T2 Key is Pressed")  KeyPressed = 2  elif DigitalValue == 1.00:  print("T1 Key is Pressed")  KeyPressed = 1  if(KeyPressed != 0):  sio.emit('BBBW3Event', {'data': KeyPressed})  print('Data sent!')  except:  print('Unable to transmit data.')  pass  time.sleep(1) |

**For BBBW4 Board**

|  |
| --- |
| import socketio  import time  import Adafruit\_BBIO.GPIO as GPIO  sio = socketio.Client()  GPIO.setup("P8\_10", GPIO.IN)  @sio.event  def connect():  print('Connection established.')  @sio.event  def disconnect():  print('Disconnected from server.')  while True:  try:  sio.connect(**'http://192.168.X.X:5000'**)  break  except:  print("Try to connect to the server.")  pass  PreviousDoorDetectionStatus = 0  while True:  try:  CurrentDoorDetectionStatus = GPIO.input("P8\_10")  if CurrentDoorDetectionStatus:  print("Magnet is Detected (Door Closed)")  else:  print("No Magnet is Detected (Door Opened)")  if (abs(CurrentDoorDetectionStatus - PreviousDoorDetectionStatus) > 0):  sio.emit('BBBW4Event', {'data': CurrentDoorDetectionStatus})  print('Data sent!')  PreviousDoorDetectionStatus = CurrentDoorDetectionStatus  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.
2. **Ensure** the web server on the PC is up and running as shown in the Figure below.

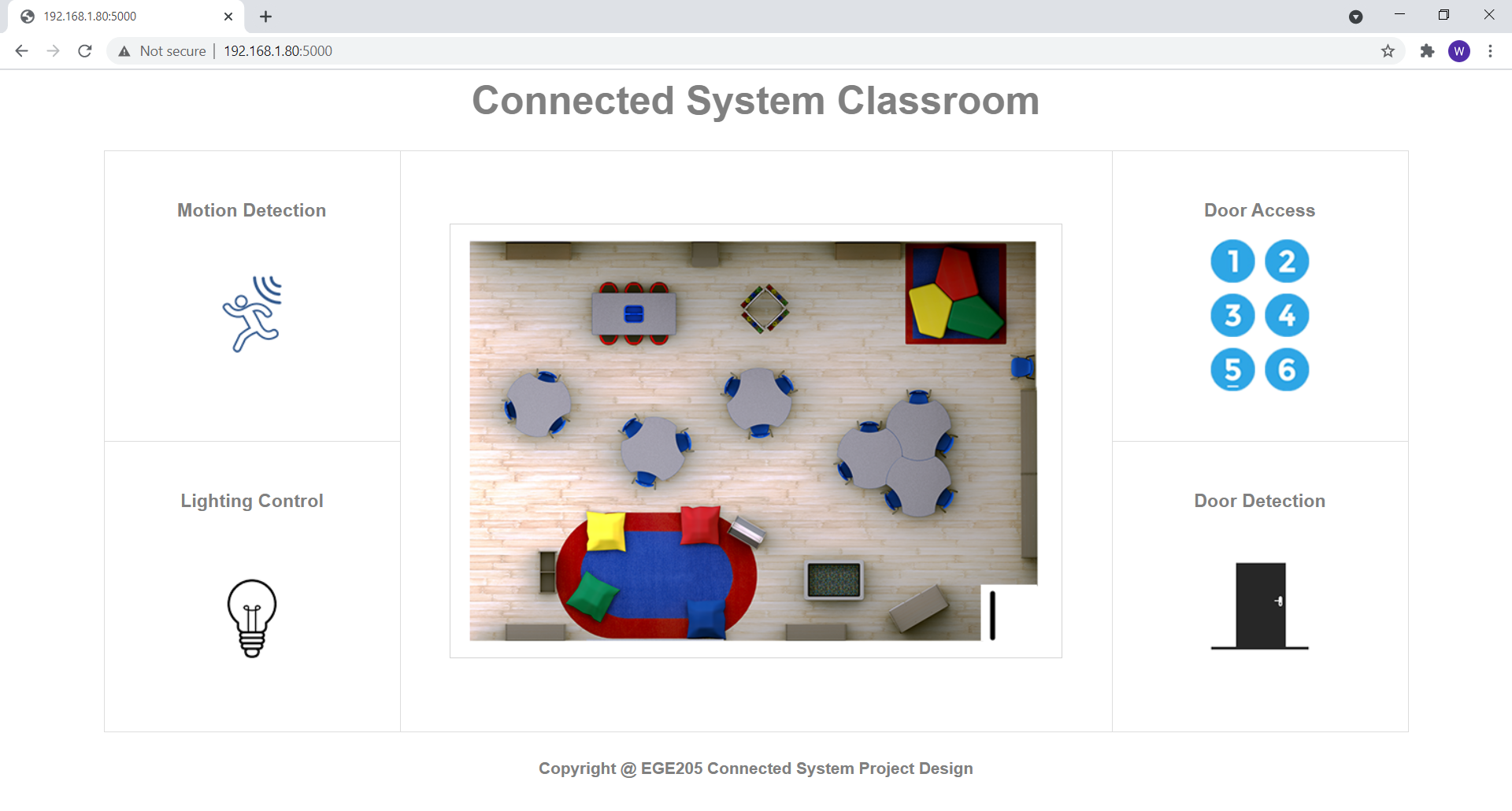


Figure 1.3r: PC Web Server

1. **Wave** your hand in front of the Motion Click or **tune** the potentiometer on the Pot Click or **press** the key “6” on the Analog Key Click then **bring** a magnet closer to the Reed Click. It is observed that the respective BBBW boards are sending data to the web server dashboard.

**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.

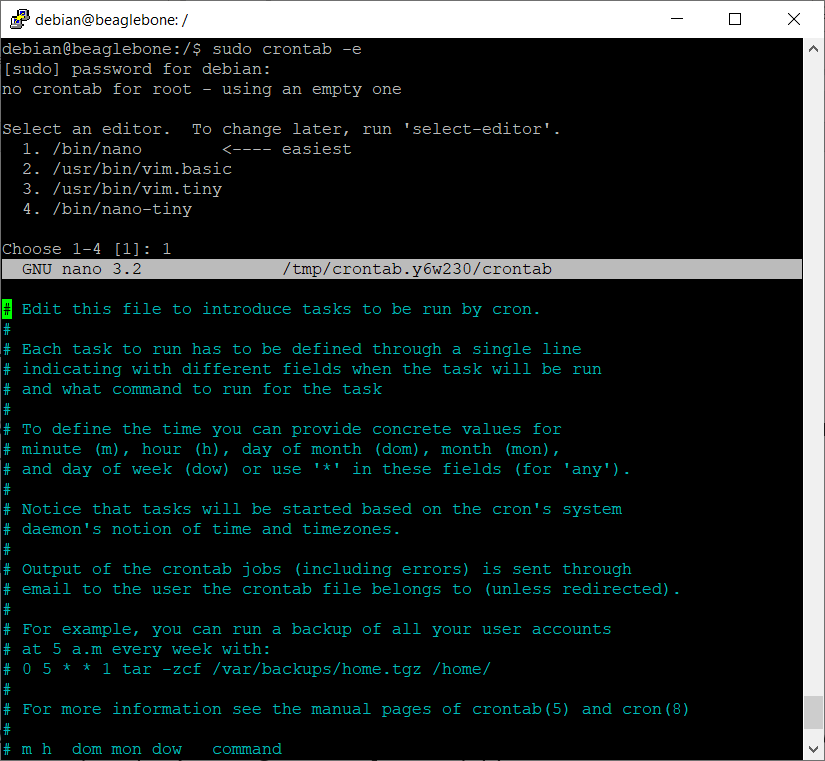


Figure 1.3s: Accessing Crontab

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.

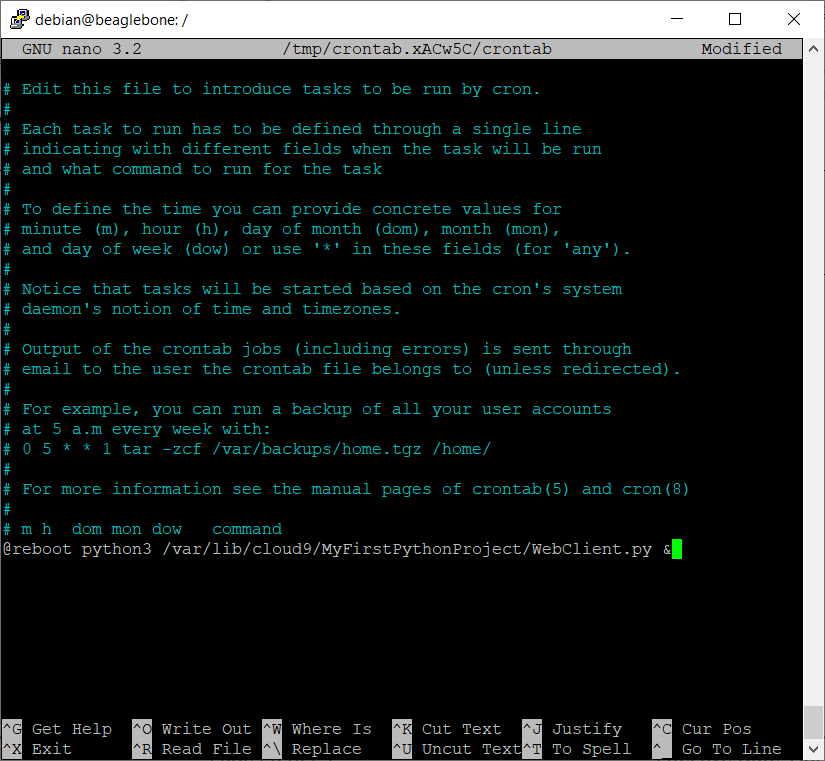


Figure 1.3t: Adding WebClient.py To Be Ran At Boot

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.

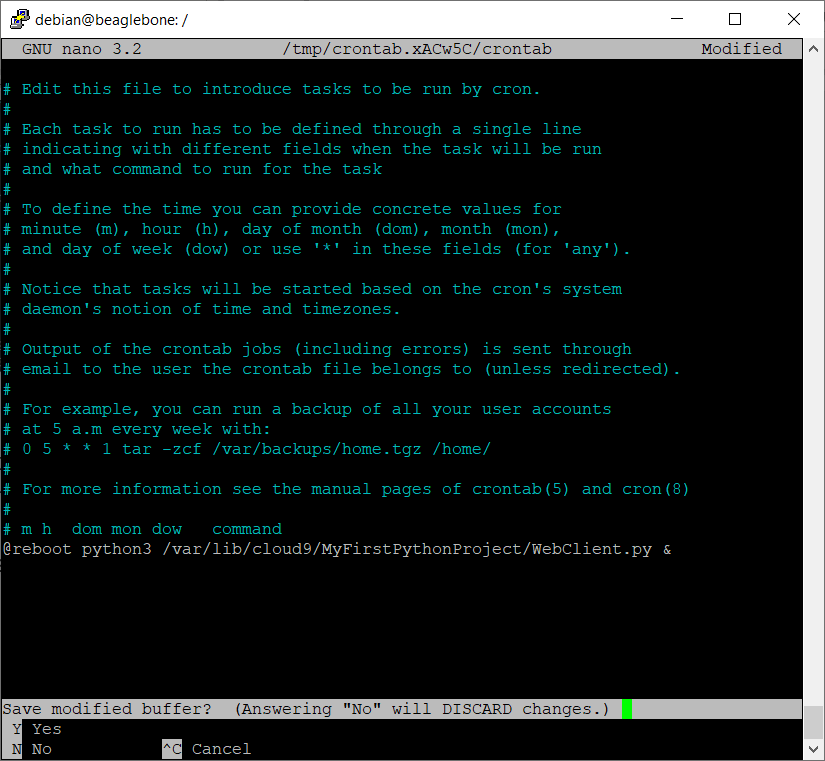


Figure 1.3u: Exiting the nano Text-editor

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. **Wave** your hand in front of the Motion Click, **tune** the potentiometer on the Pot Click, **press** the key “6” on the Analog Key Click and **bring** a magnet closer to the Reed Click. It is observed that the respective BBBW boards are sending data to the web server dashboard as shown in the Figure below.

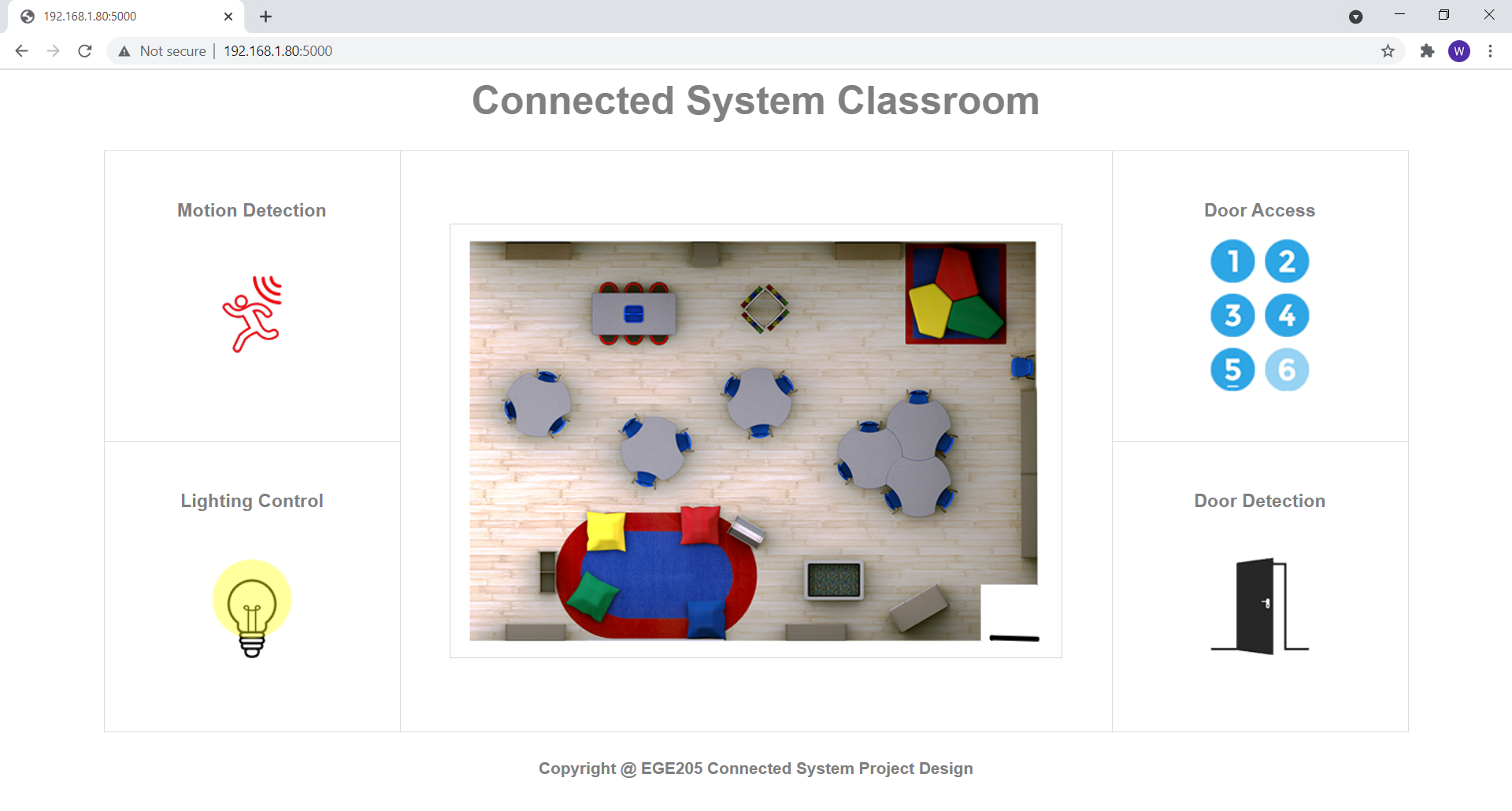


Figure 1.4a: Data Streamed into the Web Server on PC

*Congratulations! You have successfully completed the Lab6b. Good job! You should be now having higher confidence to develop your own connected system!*