

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

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

**Laboratory No**. : Lab 2a

**Laboratory Title** : Display: Controlling the LED Segment using BeagleBone Black Wireless (BBBW)

Board

**Objective** : To connect hardware click boards, install python library and write python code

to control different types of LED Segment click boards.

**Hardware Boards** : BBBW Board with USB Cable x1

: MikroBus Cape x1

: BarGraph 2 Click x1

: 7Seg Click x1

**Contents**

1. Use of Cape on BeagleBone Black Wireless (BBBW) Board
   1. BBBW Board Cape Expansion Headers and Pinout
   2. Connecting MikroBus Cape to the BBBW Board
2. Use of Python Library on BeagleBone Black Wireless (BBBW) Board
   1. Updating and Upgrading the Software Dependency Repositories in Debian OS
   2. Installing the Blinka Python Library
   3. Using the Blinka Python Library to Blink a LED
3. Controlling the LED Segment using BeagleBone Black Wireless (BBBW) Board
   1. Configuring the (BBBW) Board Boot-up Settings
   2. Understanding of BarGraph 2 Click Hardware Connection
   3. Controlling the BarGraph 2 Click using Python Code
   4. Understanding of 7Seg Click Hardware Connection
   5. Controlling the 7Seg Click using Python Code
   6. Controlling both the BarGraph 2 and 7Seg Clicks using Python Code

# **Use of Cape on BeagleBone Black Wireless (BBBW) Board**

## BBBW Board Cape Expansion Headers and Pinout

BeagleBone Black Wireless (BBBW) board has two expansion headers P8 and P9 with each header provides 46 pins as shown in the Figure below. In total there are 92 pins available for use. Below shows the distribution of pins based on different pin functions:

Power / Ground / Reset (VDD, GND) : 18 pins

General Purpose Input / Output (GPIO) : 19 pins

Analog Input (AIN) : 7 pins

Pulse Width Modulation (PWM) : 5 pins

Serial Peripheral Interface (SPI) : 7 pins

Inter-Integrate Circuit (I2C) : 2 pins

Universal Asynchronous Receiver Transmitter (UART) : 4 pins

Embedded Multi-Media Controller (eMMC) : 10 pins

Liquid-Crystal Display Interface (LCD) : 20 pins

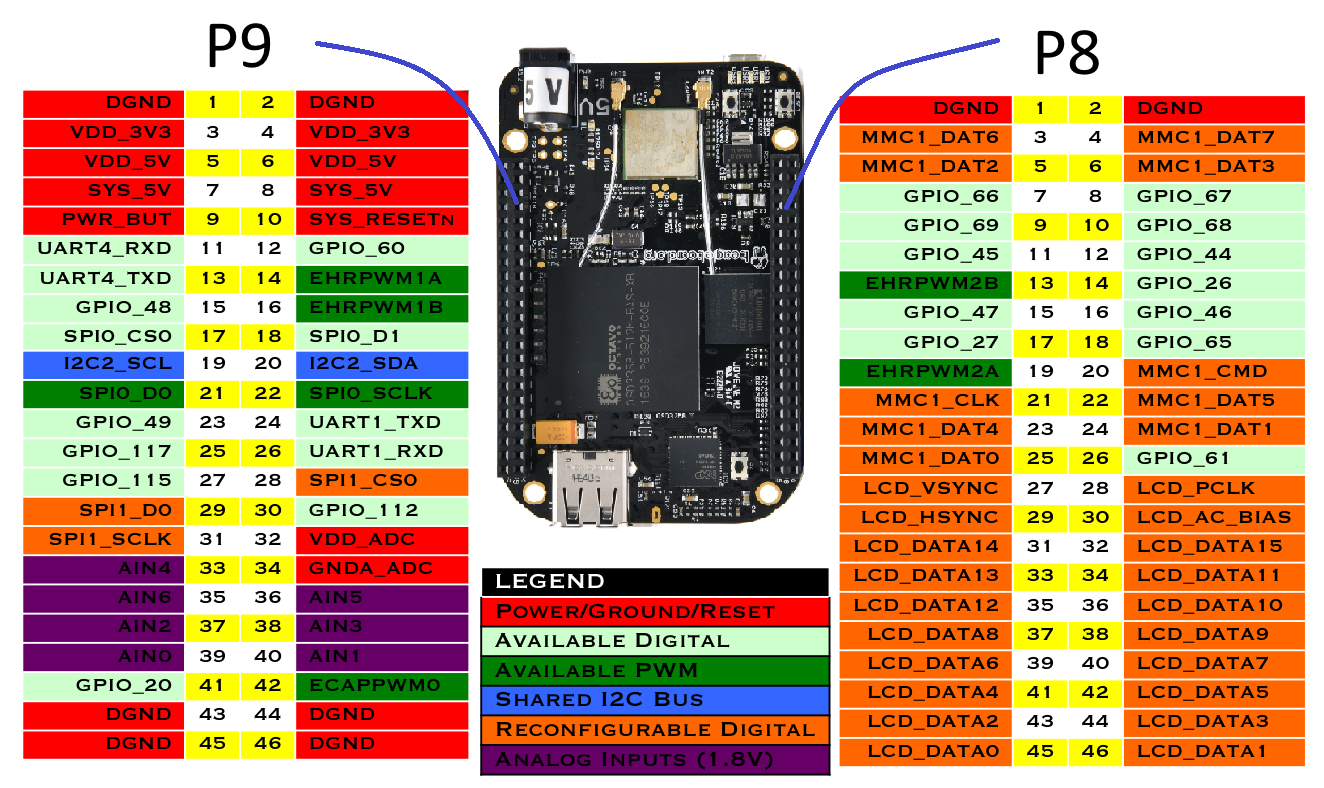


Figure 1.1a: BBBW Board Cape Expansion Headers and Pinout

## Connecting MikroBus Cape to the BBBW Board

The **mikroBUS cape** is an extension for BBBW board and introduces new forms of interaction with the outside world. This simple cape has four mikroBUS sockets that allow more than 130 different types of click boards (aka daughter boards) to be connected to the BBBW board. Additional functionality such as GSM, GPS, ZigBee or thunder detection, proximity and color sensing, hall current detection, speech recognition and many other can be quickly added to the BBBW board. To fully utilize the BBBW board pinout, two additional sets of jumpers allow you to:

1. Switch the AN pin to GPIO on each mikroBUS socket.

*(The cape has been configured to use AN pin as analog pin)*

1. Choose whether to enable UART pins on the first socket or SPI pins on the third socket.

*(The cape has been configured to use it as SPI pins on the third socket)*

The mikroBUS cape and its respective schematic are shown in the Figure below.



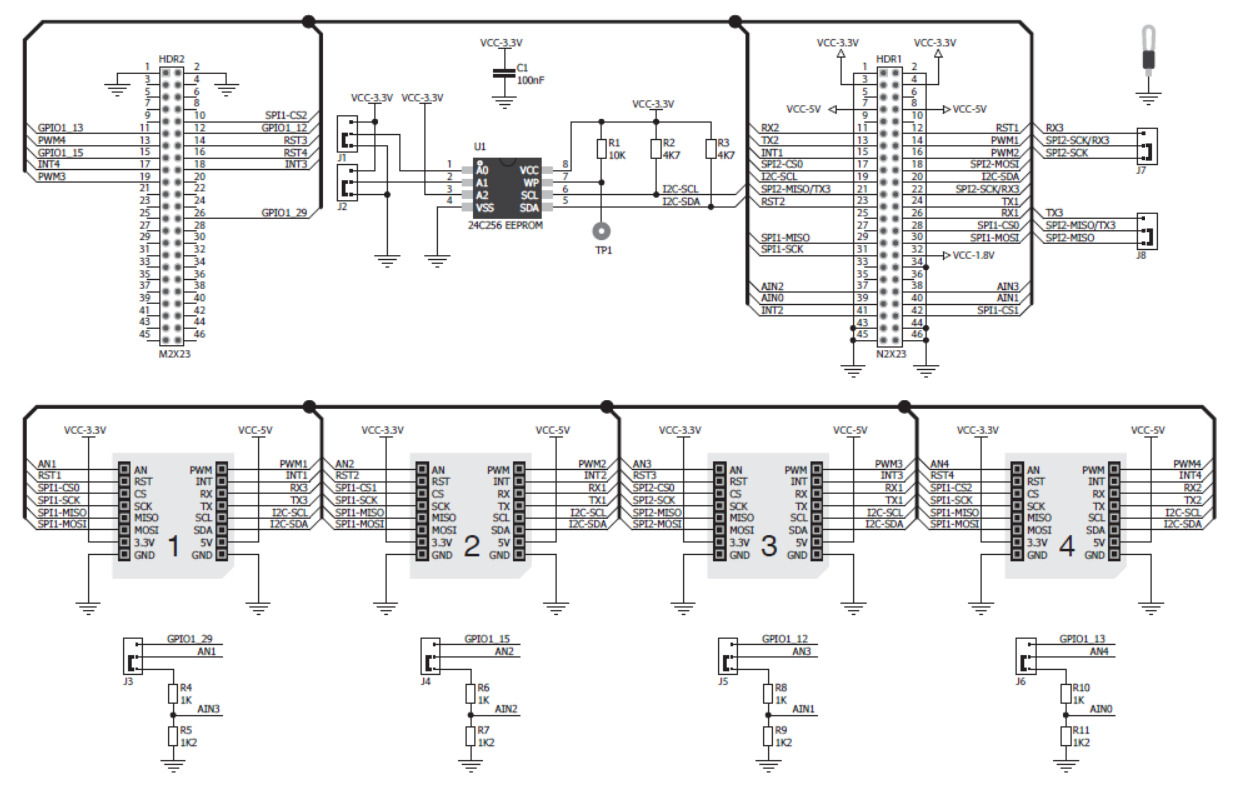


Figure 1.2a: MikroBUS Cape and Schematic

1. **Connect** the mikroCape to the BBBW board’s two expansion headers P8 and P9 with the correct orientation as show in the Figure below.

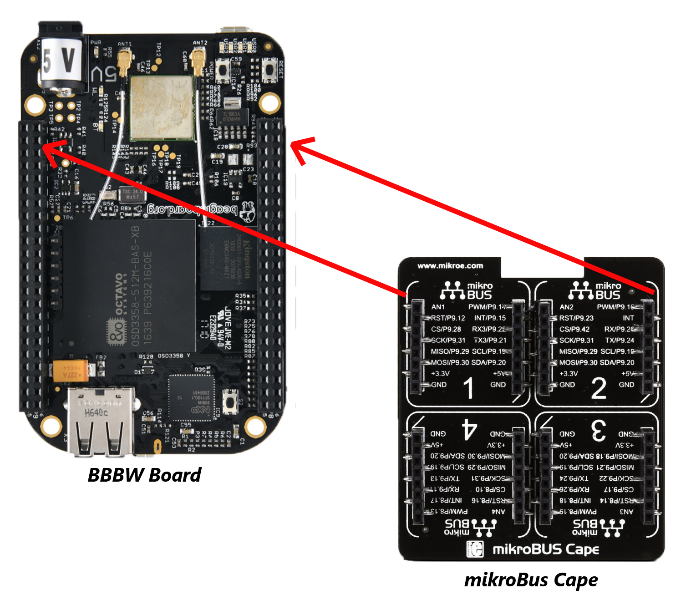


Figure 1.2b: Connecting MikroBUS Cape onto the BBBW Board

# **Use of Python Library on BeagleBone Black Wireless (BBBW) Board**

## Updating and Upgrading the Software Dependency Repositories in Debian OS

**Advanced Package Tool** (APT), is a free-software user interface that works with core libraries to handle the installation and removal of software on Debian, Ubuntu, and related Linux distributions. APT also includes command-line programs for dealing with packages, which use the library. Three such programs are apt, **apt-get** and apt-cache.

1. **Log in** to the BBBW board through SSH using the default username “**debian**” and password “**temppwd**” as shown in the Figure below.

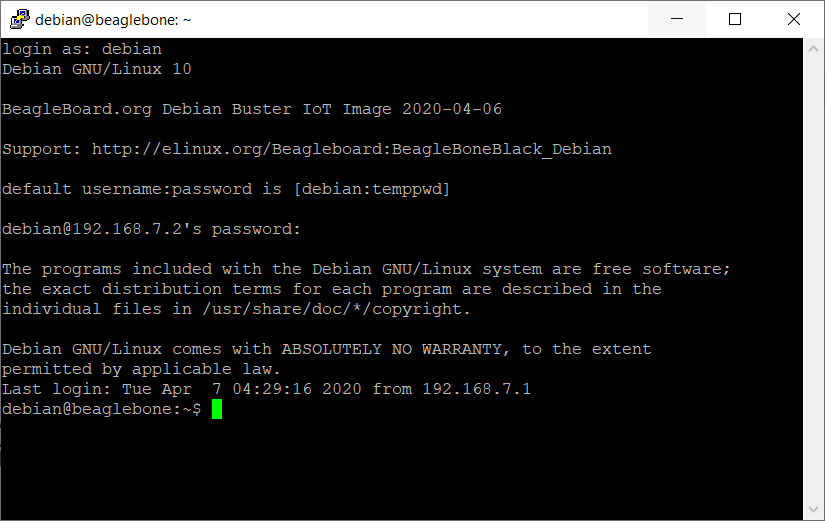


Figure 2.1a: Logging in to BBBW Board

1. **Type** in the command “**iwconfig**” and **hit** the “Enter” key. **Ensure** that the wlan0 is connected to a particular Wi-Fi’s SSID. In this case, the BBBW board has been connected to SSID: “**S435-CSDL\_dev**” as shown in the Figure below.

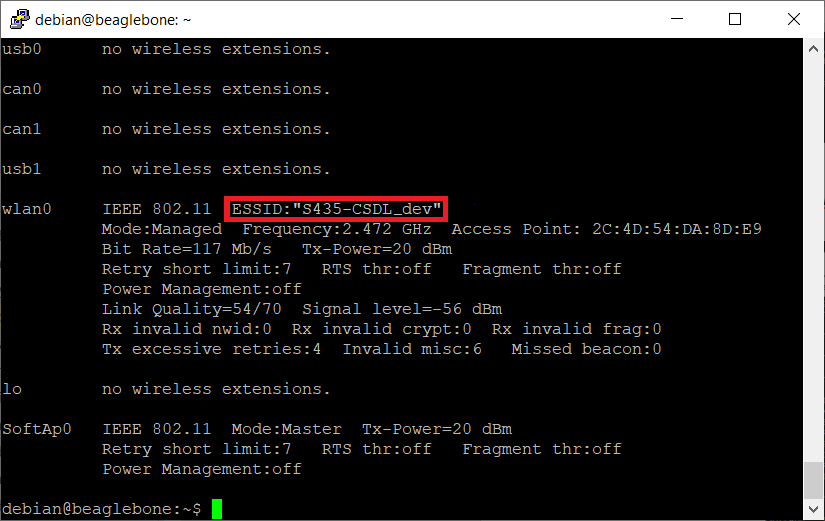


Figure 2.1b: The “iwconfig” Command

1. **Type** in the command “**date**” and **hit** the “Enter” key. This command returns the default UTC date and time configured in the system as shown in the Figure below.



Figure 2.1c: The “date” Command

1. **Type** in the command “**sudo** **date -s “25 Oct 2021 17:10:00”**” (Modify the command to use the current UTC date and time) and **hit** the “Enter” key. This command updates the system date and time according to the current UTC date and time as shown in the Figure below.

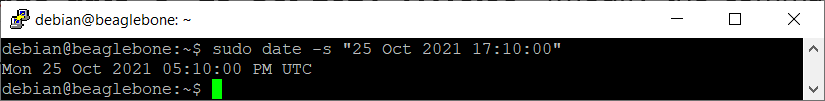


Figure 2.1d: The “date -s” Command

1. **Type** in the command “**sudo apt-get update**” and **hit** the “Enter” key. **Type** in the password “**temppwd**” and **hit** the “Enter” key again. This command synchronizes the Linux package index files from the sources and may take up to 5 minutes to complete the synchronization process as shown in the Figure below.

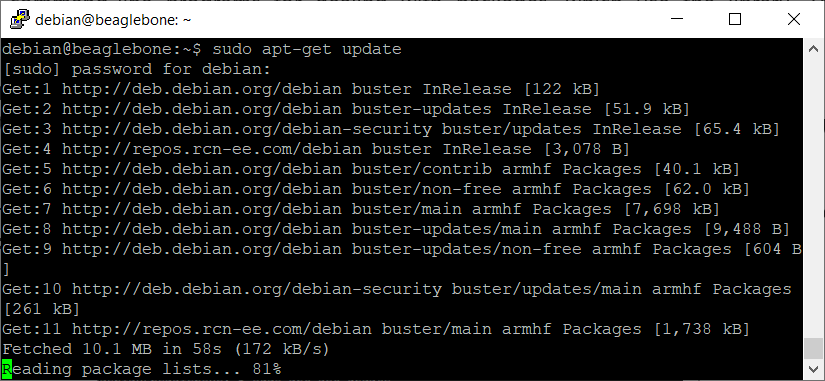


Figure 2.1e: The “apt-get update” Command

1. **Type** in the command “**sudo apt-get update**” and **hit** the “Enter” key again. The completion of the synchronization process is shown in the Figure below.

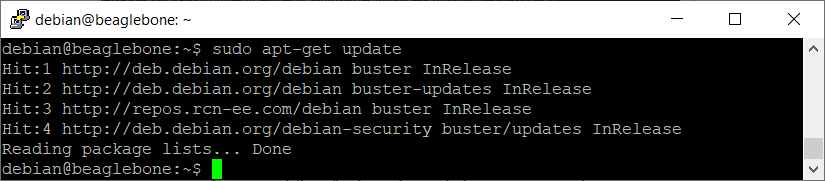


Figure 2.1f: The “apt-get update” Command

1. The BBBW board has just been successfully updated!

|  |
| --- |
| ***[DO NOT DO THE STEPS BELOW]***  ***Additional Information: Upgrade the BBBW board***   1. **Type** in the command “**sudo apt-get upgrade**” and **hit** the “Enter” key. This command installs the latest versions of the packages currently installed on the BBBW board. The installed packages which have new packages available will be retrieved and installed as shown in the Figure below. (Note: An update needs to be performed before the upgrade, so that apt-get knows that new versions of packages are available for installation)     Figure 2.1g: The “apt-get upgrade” Command   1. **Type** “**y**” and **hit** the “Enter” key to continue the upgrading process and it may take up to 15 minutes to complete the upgrading process as shown in the Figure below.     Figure 2.1h: Continue the Upgrading Process   1. **Type** in the command “**sudo apt-get upgrade**” and **hit** the “Enter” key again. **Type** in the password “**temppwd**” and **hit** the “Enter” key again if prompted. The completion of the upgrading process is shown in the Figure below.     Figure 2.1i: The “apt-get upgrade” Command |

## Installing the Blinka Python Library

**Blinka** is a python library that brings CircuitPython APIs to single board computers (SBCs) such as BBBW board. It provides a layer that translates the CircuitPython hardware API to whatever library the Linux board provides as shown in the Figure below.

There are many approaches to install python library. The approach of “**installing from source**” will be used throughout this module. This approach requires a copy of the python library source code to be cloned into the BBBW board before installing it. Hence, a folder named “**PythonLibrary**” will be created in the BBBW board to keep all the python library source code.

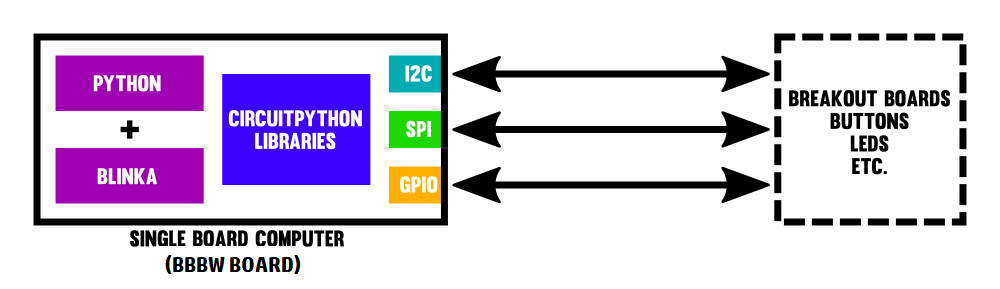


Figure 2.2a: Blinka Python Library

1. **Type** in the command “**pwd**” and **hit** the “Enter” key. It is observed that the current working directory “**/home/debian**” is returned as shown in the Figure below.



Figure 2.2b: Current Working Directory

1. **Type** in the command “**mkdir PythonLibrary**” and **hit** the “Enter” key. This command creates a folder named PythonLibrary inside the current working directory as shown in the Figure below.

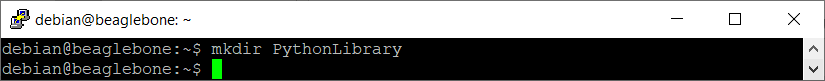


Figure 2.2c: Creating PythonLibrary Folder

1. **Type** in the command “**ls**” and **hit** the “Enter” key. It is observed that the PythonLibrary folder has been created as shown in the Figure below.



Figure 2.2d: PythonLibrary Folder is Created

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

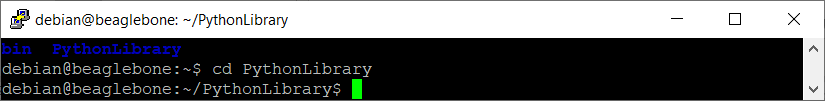


Figure 2.2e: Accessing the PythonLibrary Folder

1. **Type** in the command “**git clone https://github.com/nypege205/Adafruit\_Blinka.git**” and **hit** the “Enter” key. **Type** in the github username “**nypege205**” and password “**ghp\_Dr3jDaeKJ8fgDH06ZrtG1qUKgsmKux3XffG5**” and **hit** the “Enter” key again. This command clones a copy of the Adafruit\_Blinka Python Library repository from github as shown in the Figure below. (Note: Type the git clone command again if the follow error appear -> “*fatal: unable to access 'https://github.com/nypege205/Adafruit\_Blinka.git/': server certificate verification failed. CAfile: none CRLfile: none*”)

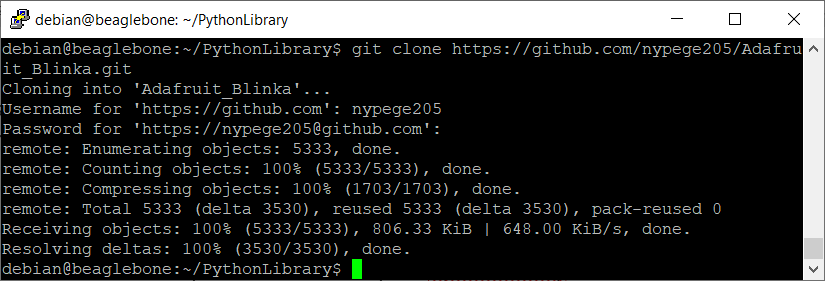


Figure 2.2f: Cloning the Blinka Python Library

1. **Type** in the command “**ls**” and **hit** the “Enter” key. The Adafruit\_Blinka Python Library folder is returned as shown in the Figure below.

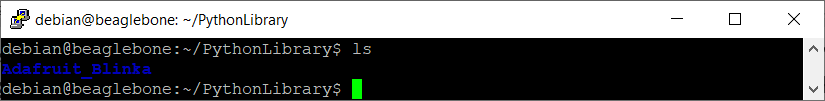


Figure 2.2g: Listing the Adafruit\_Blinka Python Library Folder

1. **Type** in the command “**cd Adafruit\_Blinka**” and **hit** the “Enter” key. This is to access the Adafruit\_Blinka folder as shown in the Figure below.

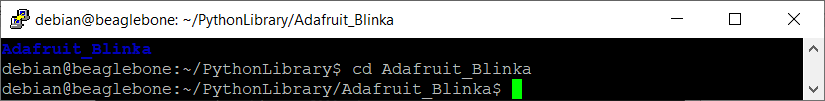


Figure 2.2h: Accessing the Adafruit\_Blinka Folder

1. **Type** in the command “**sudo python3 setup.py install**” and **hit** the “Enter” key. This command installs the Adafruit\_Blinka Python Library. The installation may take up to 5 minutes. The completion of the installation process is shown in the Figure below.

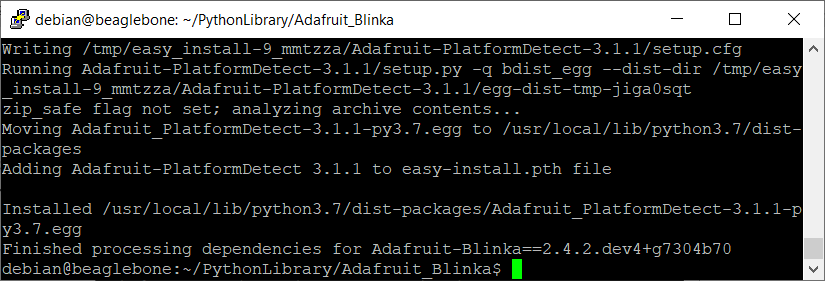


Figure 2.2i: Completion of Adafruit\_Blinka Python Library Installation

1. **Type** in the command “**sudo python3 -m pip install --upgrade --force-reinstall adafruit-blinka Adafruit-PlatformDetect**” and **hit** the “Enter” key.
2. The BBBW board has just been successfully installed with the Adafruit\_Blinka Python Library.

## Using the Blinka Python Library to Blink a LED

1. **Ensure** that the BBBW board is powered up and connected to the computer through a USB cable. **Open** the web browser (preferably Chrome browser) and **type** “**http://192.168.7.2:3000**” in the address bar as shown in the Figure.

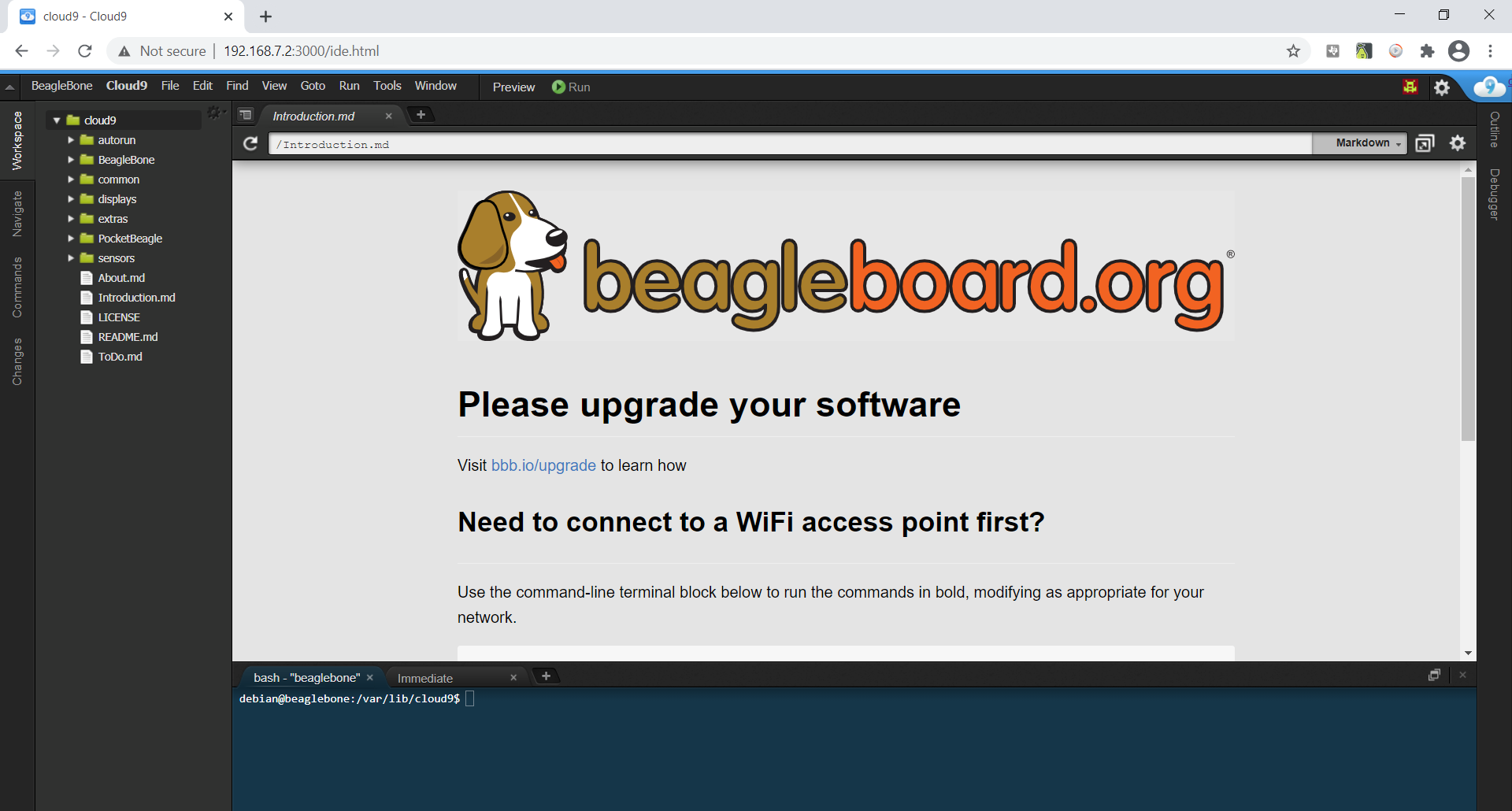


Figure 2.3a: Cloud9 IDE

1. **Right click** on the folder “**MyFirstPythonProject”** and **select** the “**New File**” from the drop-down menu to create a new python file. **Name** the file as “**Blinky.py**” as shown in the Figure below.

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

Figure 2.3b: Creating a New File

1. **Double click** on the newly created file “**Blinky.py**” and **enter** the following code into the file under the Editor section as shown in the Figure below.

|  |
| --- |
| import time  import Adafruit\_BBIO.GPIO as GPIO  GPIO.setup("USR0", GPIO.OUT)  while True:  GPIO.output("USR0", GPIO.HIGH)  time.sleep(0.5)  GPIO.output("USR0", GPIO.LOW)  time.sleep(0.5) |

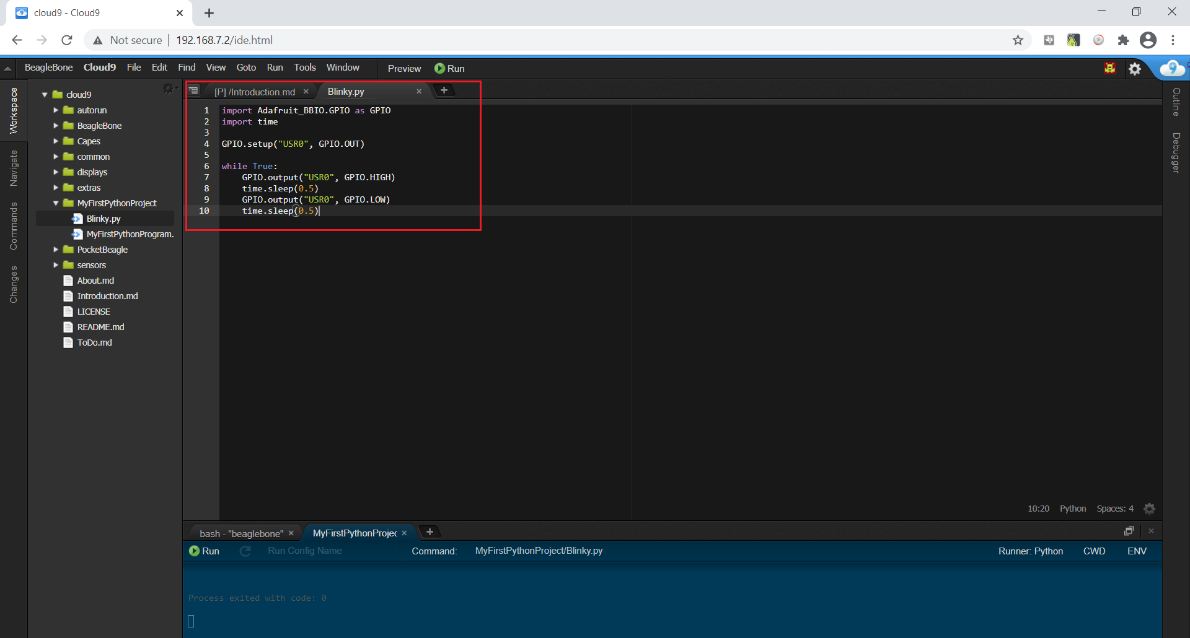


Figure 2.3c: Blinky Python Code

1. **Click** on the “Run” button located beside the Menu Tab to execute the “**Blinky.py**” file. It is observed that the blue LED (USR0) located beside the reset button is blinking at the rate of 0.5s as shown in the Figure below.

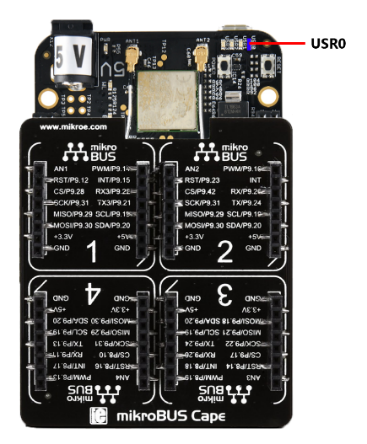


Figure 2.3d: Location of the Blue LED (USR0)

# **Controlling the LED Segment using BeagleBone Black Wireless (BBBW) Board**

## Configuring the (BBBW) Board Boot-up Settings

The BBBW board includes 2 SPIs namely SPI0 and SPI1. Both SPI0 and SPI1 cannot be used at runtime and SPI1 is not available by default as the HDMI interface is utilizing one of the pins. Hence, the following actions need to be done to enable both SPI0 and SPI1 for use.

1. Enable both SPI0 and SPI1 at boot.
2. Disable HDMI interface at boot.
3. **Log in** to the BBBW board through SSH using the default username “**debian**” and password “**temppwd**”.
4. **Type** in the command “**sudo nano /boot/uEnv.txt**“ and **hit** the “Enter” key. **Type** in the password “**temppwd**” and **hit** the “Enter” key again. This command accesses and opens the file **uEnv.txt** found in the /boot folder for editing using “**nano**” which is a simple command-line text editor included in most Linux installations as shown in the Figure below.

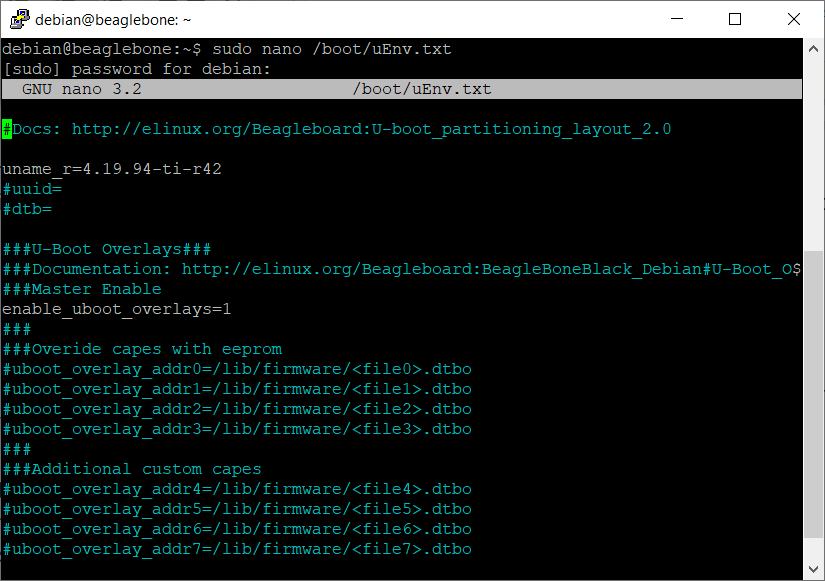


Figure 3.1a: Assessing the Boot File uEnv.txt

1. **Add** the following 2 lines of text into the file uEnv.txt. This is to enable both SPI0 and SPI1 at boot as shown in the Figure below.

|  |
| --- |
| uboot\_overlay\_addr4=/lib/firmware/BB-SPIDEV0-00A0.dtbo  uboot\_overlay\_addr5=/lib/firmware/BB-SPIDEV1-00A0.dtbo |

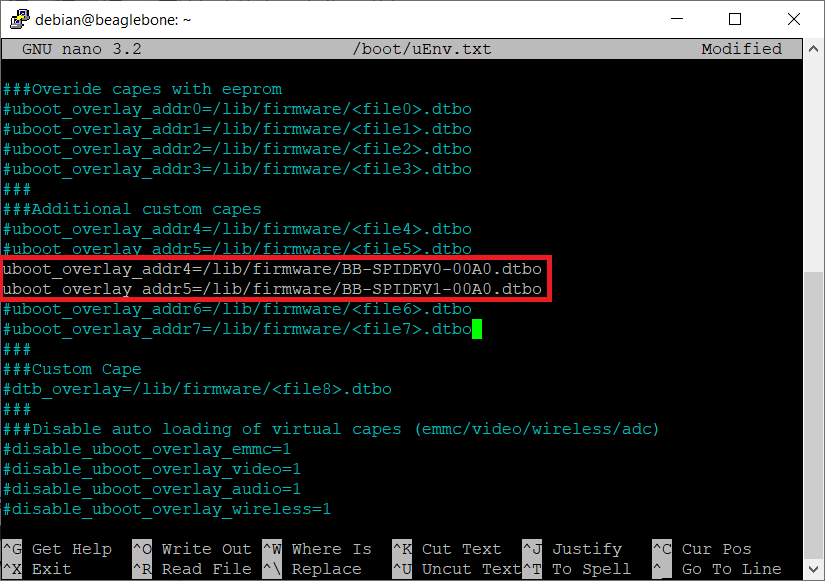


Figure 3.1b: Enabling Both SPI0 and SPI1 at Boot

1. **Remove** the “**#**” for the following 2 lines of text in the file uEnv.txt. This is to disable HDMI interface at boot as shown in the Figure below.

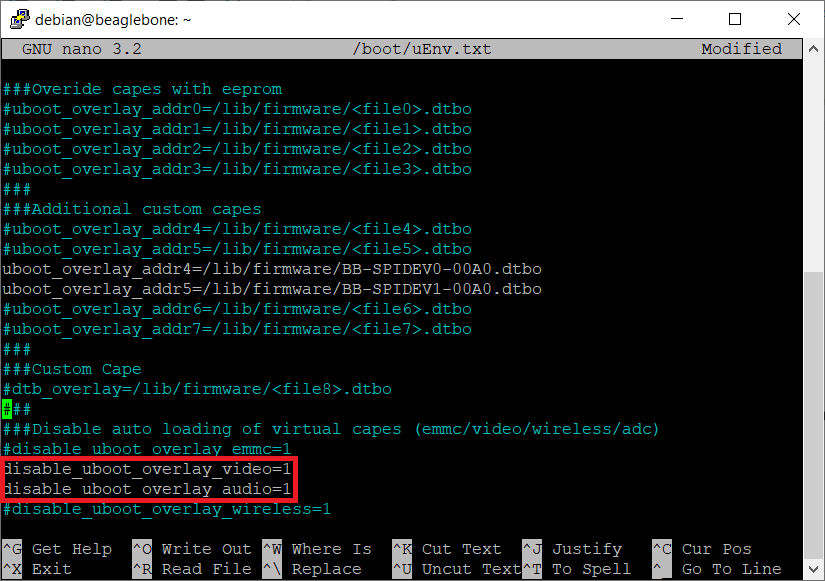


Figure 3.1c: Disabling the HDMI Interface At Boot

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

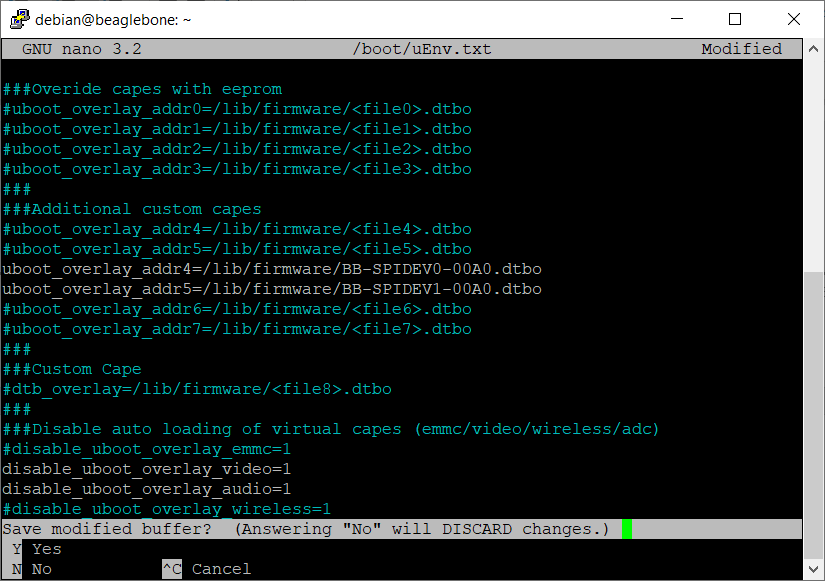


Figure 3.1d: Exiting the nano Text-Editor

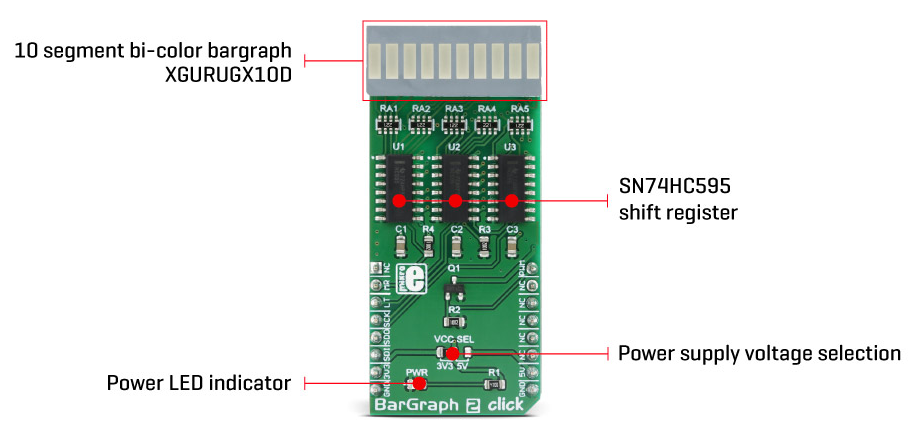
1. Type in the command “**sudo reboot**” and **hit** the “Enter” key. This is to reboot the BBBW board for the new settings to take effect on the next boot-up. **Please note that the SSH connection will be dropped.**

## Understanding of BarGraph 2 Click Hardware Connection

**BarGraph 2 Click** is a 10-segment bar graph display click, which uses a high-quality, multicolor bar graph LED display. The bar graph display is a very popular device for displaying various properties, whether it be an audio level, current/voltage level, position of the encoder, and generally any property that can be displayed in a form of a bar graph. The segments of the on-board bar graph LED display are bright and uniformly colored, providing a nice and clean visual feedback.

Each segment is composed of green and red colored LEDs, making it possible to have various important states marked in different color. It can use green, red, and a combination of these two - resulting in having amber colored segments. For example, a VU-meter can have its highest values shown in red (peaks), middle in yellow and the lowest in green; the position of the encoder can be marked with a single red segment, on a green background, and so on. The bar graph display light intensity can be dimmed, by applying a PWM signal. The Click board offers a lot of possibilities for building a custom bar graph application - VU meters, status indicators, various types of gauges, and similar.

The BarGraph 2 Click, and its respective schematic are shown in the Figure below.



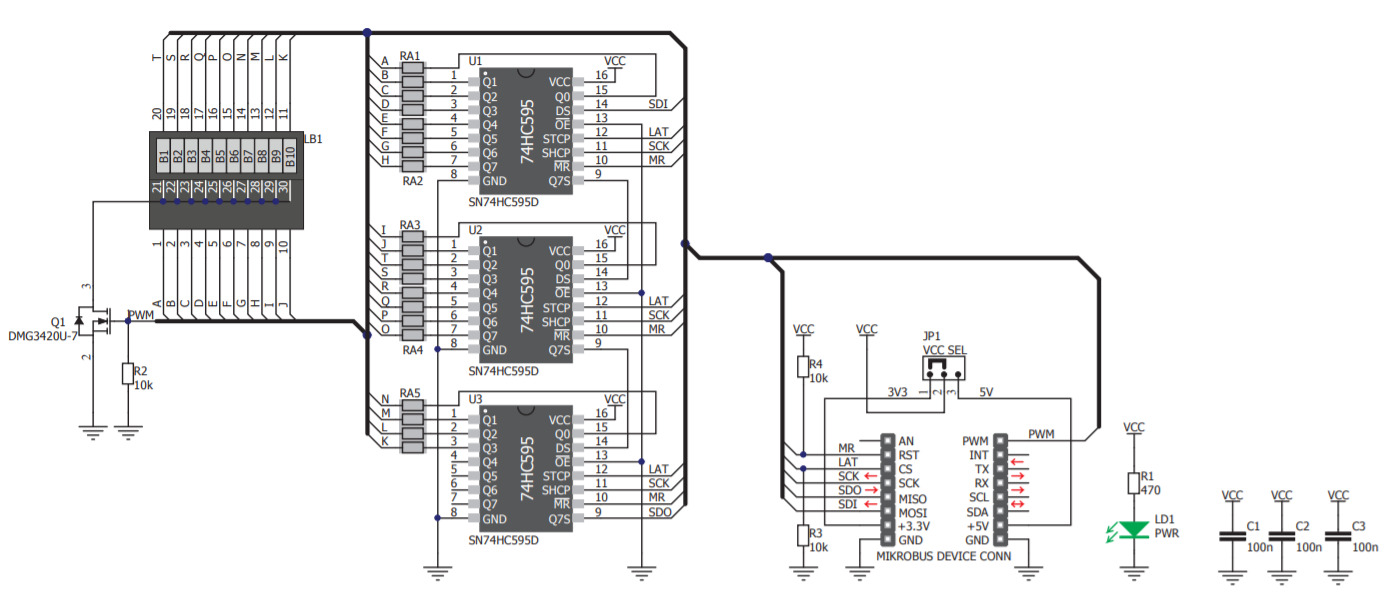


Figure 3.2a: BarGraph 2 Click and Schematic

1. **Connect** the BarGraph 2 Click to the mikroBUS cape and BBBW board as shown in the Figure below.

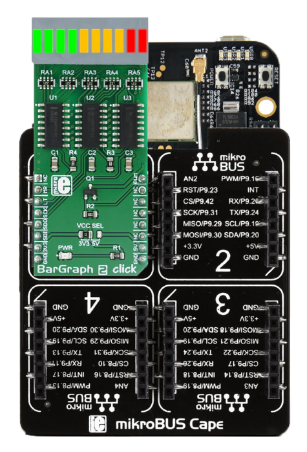


Figure 3.2b: Connecting BarGraph 2 Click to mikroBUS Cape and BBBW Board

## Controlling the BarGraph 2 Click using Python Code

1. **Ensure** that the BBBW board is powered up and connected to the computer through a USB cable. **Open** the web browser (preferably Chrome browser) and **type** “**http://192.168.7.2:3000**” in the address bar as shown in the Figure below.

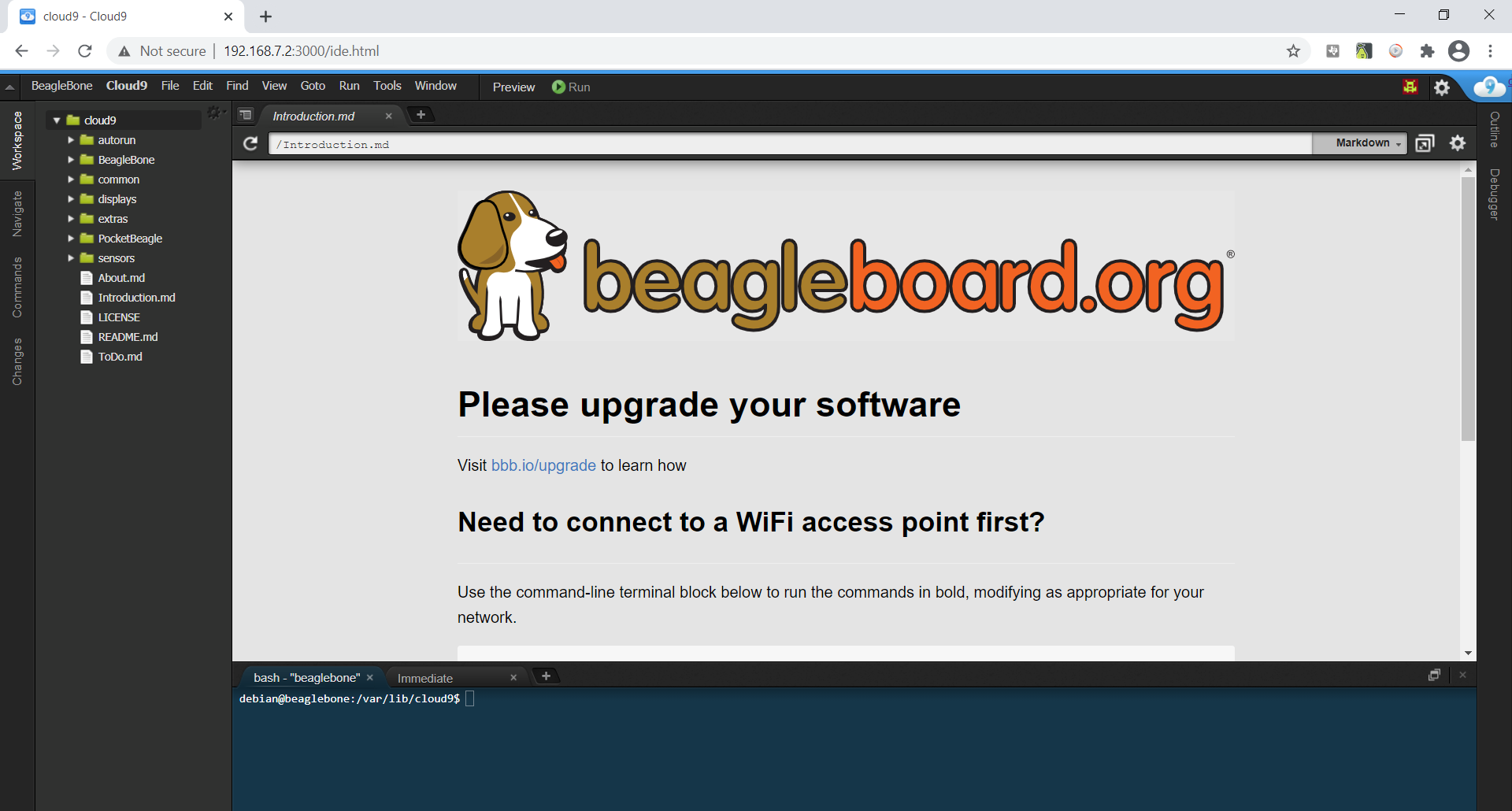


Figure 3.3a: Cloud9 IDE

1. **Right click** on the folder “**MyFirstPythonProject”** and **select** the “**New File**” from the drop-down menu to create a new python file. **Name** the file as “**BarGraph2.py**”.
2. **Double click** on the newly created file “**BarGraph2.py**” and enter the following code into the file under the Editor section.

|  |
| --- |
| import time  from Adafruit\_BBIO.SPI import SPI  import Adafruit\_BBIO.GPIO as GPIO  def BarGraph2Init():  GPIO.setup("P9\_14", GPIO.OUT)  GPIO.setup("P9\_12", GPIO.OUT)  GPIO.output("P9\_14", GPIO.HIGH)  GPIO.output("P9\_12", GPIO.HIGH)  L\_Spi1 = SPI(1,0)  L\_Spi1.mode = 0  return L\_Spi1  def BarGraph2Display(L\_Spi1, L\_NumberOfBar):  if L\_NumberOfBar == 0:  L\_Spi1.writebytes([0x00, 0x00, 0x00])  if L\_NumberOfBar == 1:  L\_Spi1.writebytes([0x00, 0x00, 0x01])  if L\_NumberOfBar == 2:  L\_Spi1.writebytes([0x00, 0x00, 0x03])  if L\_NumberOfBar == 3:  L\_Spi1.writebytes([0x00, 0x00, 0x07])  if L\_NumberOfBar == 4:  L\_Spi1.writebytes([0x00, 0x00, 0x0F])  if L\_NumberOfBar == 5:  L\_Spi1.writebytes([0x00, 0x40, 0x1F])  if L\_NumberOfBar == 6:  L\_Spi1.writebytes([0x00, 0xC0, 0x3F])  if L\_NumberOfBar == 7:  L\_Spi1.writebytes([0x01, 0xC0, 0x7F])  if L\_NumberOfBar == 8:  L\_Spi1.writebytes([0x03, 0xC0, 0xFF])  if L\_NumberOfBar == 9:  L\_Spi1.writebytes([0x07, 0xC0, 0xFF])  if L\_NumberOfBar == 10:  L\_Spi1.writebytes([0x0F, 0xC0, 0xFF])  G\_NumberOfBar = 0  G\_Spi1 = BarGraph2Init()  while True:  BarGraph2Display(G\_Spi1, G\_NumberOfBar)  G\_NumberOfBar += 1  if G\_NumberOfBar == 11:  G\_NumberOfBar = 0    time.sleep(0.3) |

1. The code above uses the functions explained below to make the display on the BarGraph 2 Click possible.

|  |  |
| --- | --- |
| **Function** | **Description** |
| BarGraph2Init() | To initialize the required GPIO and communication channel (SPI) and return the initialized SPI instance to the function call. |
| BarGraph2Display(L\_Spi1, L\_NumberOfBar) | To control the display on the BarGraph 2 Click. The function needs to be supplied with an initialized SPI instance to L\_Spi1 and an integer number ranging from 0-10 to L\_NumberOfBar. |

1. The BarGraph 2 Click consists of 10 yellow and 10 red leds. The yellow and red led can be turned on at the same time to produce an amber color as shown in the Figure below.

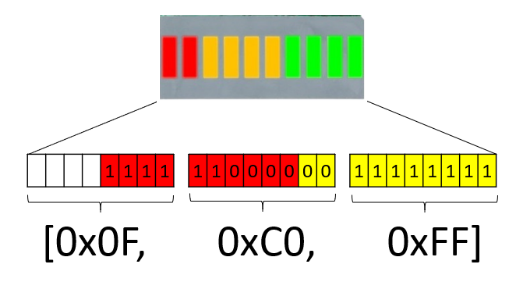


Figure 3.3b: Cloud9 IDE

1. **Click** on the “Run” button located beside the Menu Tab to execute the “**BarGraph2.py**” file. It is observed that the LED bars located on the BarGraph2 Click is turned on progressively from left to right with the color changing from green to amber then red and repeats itself.

## Understanding of 7Seg Click Hardware Connection

**7seg Click** is an accessory board in mikroBUS form factor. It’s a compact and easy solution for adding 2-digit seven-segment display to your device.

7Seg Click and its respective schematic are shown in the Figure below.



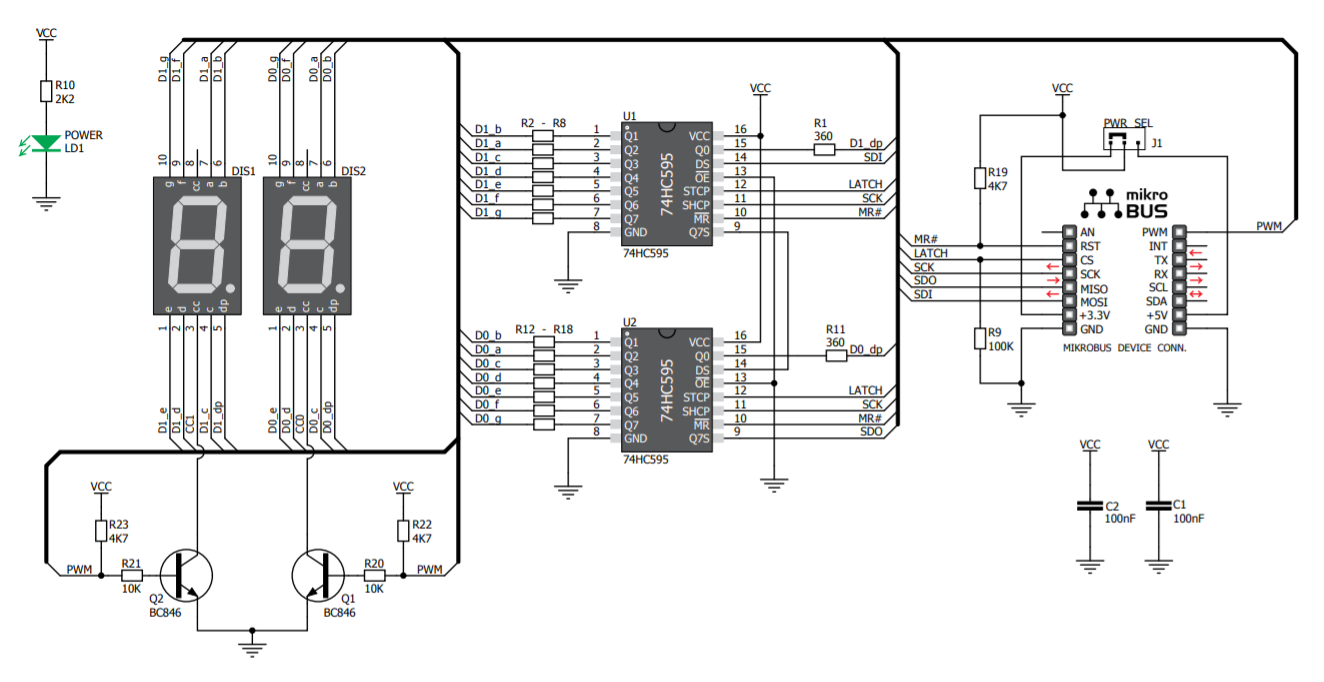


Figure 3.4a: 7Seg Click and Schematic

1. **Connect** the 7Seg Click to the mikroBUS cape and BBBW board as shown in the Figure below.

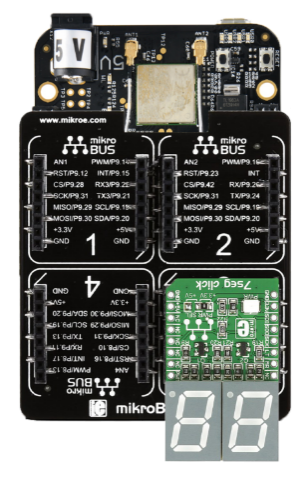


Figure 3.4b: Connecting 7Seg Click to mikroBUS Cape and BBBW Board

## Controlling the 7Seg Click using Python Code

1. **Ensure** that the BBBW board is powered up and connected to the computer through a USB cable. **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 python file. Name the file as “**7Seg.py**”.
3. **Double click** on the newly created file “**7Seg.py**” and enter the following code into the file under the Editor section.

|  |
| --- |
| import time  from Adafruit\_BBIO.SPI import SPI  import Adafruit\_BBIO.GPIO as GPIO  def SevenSegInit():  GPIO.setup("P8\_19", GPIO.OUT)  GPIO.setup("P8\_14", GPIO.OUT)  GPIO.output("P8\_19", GPIO.HIGH)  GPIO.output("P8\_14", GPIO.HIGH)  L\_Spi0 = SPI(0,0)  L\_Spi0.mode = 0  return L\_Spi0  def SevenSegDisplay(L\_Spi0, L\_Number):  DigitList = [0x7E, 0x0A, 0xB6, 0x9E, 0xCA, 0xDC, 0xFC, 0x0E, 0xFE, 0xDE]  OnesDigit = L\_Number % 10  TensDigit = L\_Number / 10  L\_Spi0.writebytes([DigitList[int(OnesDigit)], DigitList[int(TensDigit)]])  G\_Number = 0  G\_Spi0 = SevenSegInit()  while True:  SevenSegDisplay(G\_Spi0, G\_Number)  G\_Number += 1  if G\_Number == 100:  G\_Number = 0    time.sleep(0.3) |

1. The code above uses the functions explained below to make the display on the 7Seg Click possible.

|  |  |
| --- | --- |
| **Function** | **Description** |
| SevenSegInit() | To initialize the required GPIO and communication channel (SPI) and return the initialized SPI instance to the function call. |
| SevenSegDisplay(L\_Spi0, L\_Number) | To control the display on the 7Seg Click. The function needs to be supplied with an initialized SPI instance to G\_Spi0 and an integer number ranging from 0-99 to L\_Number. |

1. **Click** on the “Run” button located beside the Menu Tab to execute the “**7Seg.py**” file. It is observed that the 7 segments digits located on the 7Seg Click are counting from 00-99 and repeats itself.

## Controlling both the BarGraph 2 and 7Seg Clicks using Python Code

1. **Connect** both the BarGraph 2 and 7Seg Clicks to the mikroBUS cape and BBBW board as shown in the Figure below.

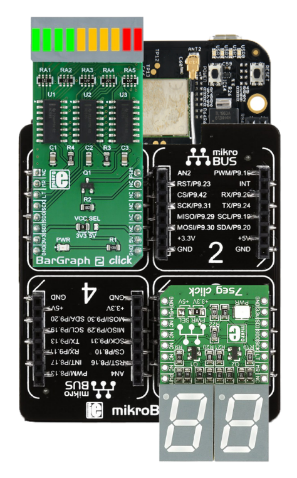


Figure 3.6a: Connecting both The BarGraph 2 and 7Seg Click to mikroBUS Cape and BBBW Board

1. **Ensure** that the BBBW board is powered up and connected to the computer through a USB cable. **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 python file. **Name** the file as “**BarGraph2\_7Seg.py**”.
3. **Double click** on the newly created file “**BarGraph2\_7Seg.py**” and start entering code into the file under the Editor section.
4. **Enter** the code that imports all the necessary library to be used in the program as shown below.

|  |
| --- |
| import time  from Adafruit\_BBIO.SPI import SPI  import Adafruit\_BBIO.GPIO as GPIO |

1. **Enter** the code of the 4 functions to be called in the program as shown below.

|  |
| --- |
| def BarGraph2Init():  GPIO.setup("P9\_14", GPIO.OUT)  GPIO.setup("P9\_12", GPIO.OUT)  GPIO.output("P9\_14", GPIO.HIGH)  GPIO.output("P9\_12", GPIO.HIGH)  L\_Spi1 = SPI(1,0)  L\_Spi1.mode = 0  return L\_Spi1  def SevenSegInit():  GPIO.setup("P8\_19", GPIO.OUT)  GPIO.setup("P8\_14", GPIO.OUT)  GPIO.output("P8\_19", GPIO.HIGH)  GPIO.output("P8\_14", GPIO.HIGH)  L\_Spi0 = SPI(0,0)  L\_Spi0.mode = 0  return L\_Spi0  def BarGraph2Display(L\_Spi1, L\_NumberOfBar):  if L\_NumberOfBar == 0:  L\_Spi1.writebytes([0x00, 0x00, 0x00])  if L\_NumberOfBar == 1:  L\_Spi1.writebytes([0x00, 0x00, 0x01])  if L\_NumberOfBar == 2:  L\_Spi1.writebytes([0x00, 0x00, 0x03])  if L\_NumberOfBar == 3:  L\_Spi1.writebytes([0x00, 0x00, 0x07])  if L\_NumberOfBar == 4:  L\_Spi1.writebytes([0x00, 0x00, 0x0F])  if L\_NumberOfBar == 5:  L\_Spi1.writebytes([0x00, 0x40, 0x1F])  if L\_NumberOfBar == 6:  L\_Spi1.writebytes([0x00, 0xC0, 0x3F])  if L\_NumberOfBar == 7:  L\_Spi1.writebytes([0x01, 0xC0, 0x7F])  if L\_NumberOfBar == 8:  L\_Spi1.writebytes([0x03, 0xC0, 0xFF])  if L\_NumberOfBar == 9:  L\_Spi1.writebytes([0x07, 0xC0, 0xFF])  if L\_NumberOfBar == 10:  L\_Spi1.writebytes([0x0F, 0xC0, 0xFF])  def SevenSegDisplay(L\_Spi0, L\_Number):  OnesDigit = L\_Number % 10  TensDigit = L\_Number / 10  L\_Spi0.writebytes([DigitList[int(OnesDigit)], DigitList[int(TensDigit)]]) |

1. **Enter** the main code that call the 4 functions to control both the BarGraph 2 and 7Seg Click as shown below.

|  |
| --- |
| G\_NumberOfBar = 1  G\_Number = 0  NumCounter = 0  DigitList = [0x7E, 0x0A, 0xB6, 0x9E, 0xCA, 0xDC, 0xFC, 0x0E, 0xFE, 0xDE]  G\_Spi0 = SevenSegInit()  G\_Spi1 = BarGraph2Init()  while True:  SevenSegDisplay(G\_Spi0, G\_Number)  BarGraph2Display(G\_Spi1, G\_NumberOfBar)    G\_Number += 1  if G\_Number == 100:  G\_Number = 0    NumCounter += 1  if NumCounter == 10:  NumCounter = 0  G\_NumberOfBar += 1  if G\_NumberOfBar == 11:  G\_NumberOfBar = 1    time.sleep(0.3) |

1. **Click** on the “Run” button located beside the Menu Tab to execute the “**BarGraph2\_7Seg.py**” file.
2. **Observe** and **compare** the output on both the BarGraph 2 and 7Seg Clicks with your teammates and **consult** your lecturer for advice if it is not the same.
3. **Write** the program output in the white box below for future reference if needed.

|  |
| --- |
| *Right click and select “New comment” to insert your program as a comment.* |

*Congratulations! You have successfully completed the Lab2a. Good job! Take a good break and stay tune for next lab. More exciting lab exercises coming to you!*