

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

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

**Laboratory No**. : Lab 3b

**Laboratory Title** : HMI: Reading the Analog Inputs using BeagleBone Black Wireless (BBBW) Board

**Objective** : To connect hardware click boards and write python code to read different types

of analog input click boards.

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

: MikroBus Cape x1

: Pot Click x1

: Analog Key Click x1

: BarGraph 2 Click x1

**Contents**

1. Reading the Analog Input using BeagleBone Black Wireless (BBBW) Board
   1. Understanding of Pot Click Hardware Connection
   2. Reading the Analog Data from Pot Click using Python Code
   3. Understanding of Analog Key Click Hardware Connection
   4. Reading the Analog Data from Analog Key Click using Python Code
2. Displaying the Input Data using BeagleBone Black Wireless (BBBW) Board
   1. Developing a Volume Control Indicator using Pot and BarGraph 2 Click
   2. Tinkering Time

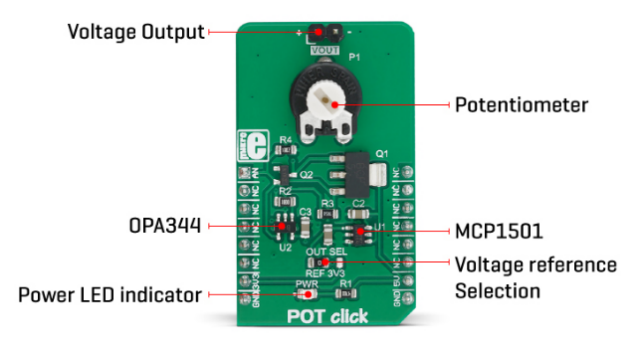
# **Reading the Analog Input using BeagleBone Black Wireless (BBBW) Board**

## Understanding of Pot Click Hardware Connection

**Pot Click** is equipped with an accurate selectable reference voltage output. By employing a high-quality 10mm carbon potentiometer, it can provide very accurate voltage output. It is also equipped with the SMD jumper, which allows the maximum reference voltage to be selected between two typically used values: 2.048V or 3.3V. The output is buffered with a rail-to-rail buffering operational amplifier, provides constant input and output impedance. The output current is limited by the protection circuit at the output to prevent unintentional damage if the output is shorted to GND.

With the ability to adjust the voltage on the AN pin very accurately, it can be used for development of applications that require voltage reference other than what is commonly available as an output from a dedicated IC. This includes HMI applications where the potentiometer can be used to provide fine control the movement of a step motor, or intensity of a LED segment, backlight amount of a TFT screen, and similar applications that require fine control of the voltage reference.

**Pot Click**, and its respective schematic are shown in the Figure below.



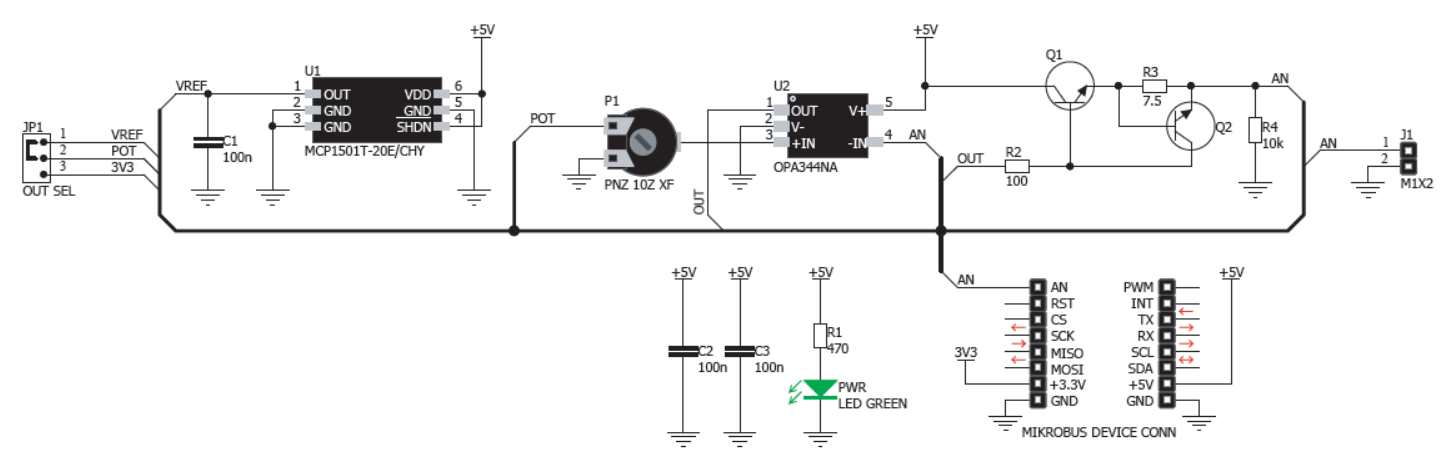


Figure 1.1a: Pot Click and Schematic

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

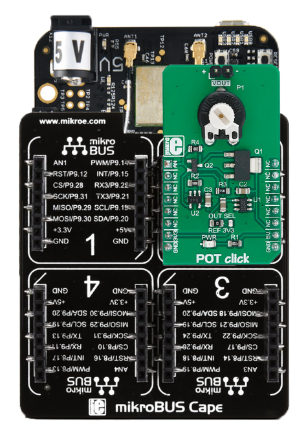


Figure 1.1b: Connecting Pot Click to mikroBUS Cape and BBBW Board

## Reading the Analog Data from Pot 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 “**Pot.py**”.
3. **Double click** on the newly created file “**Pot.py**” and enter the following code into the file under the Editor section.

|  |
| --- |
| import time  import Adafruit\_BBIO.ADC as ADC    ADC.setup()  while True:  DigitalValue = ADC.read("P9\_37")  AnalogVoltage = DigitalValue \* 1.8  print("Digital Value: %f, Analog Voltage: %f" % (DigitalValue, AnalogVoltage))  time.sleep(0.3) |

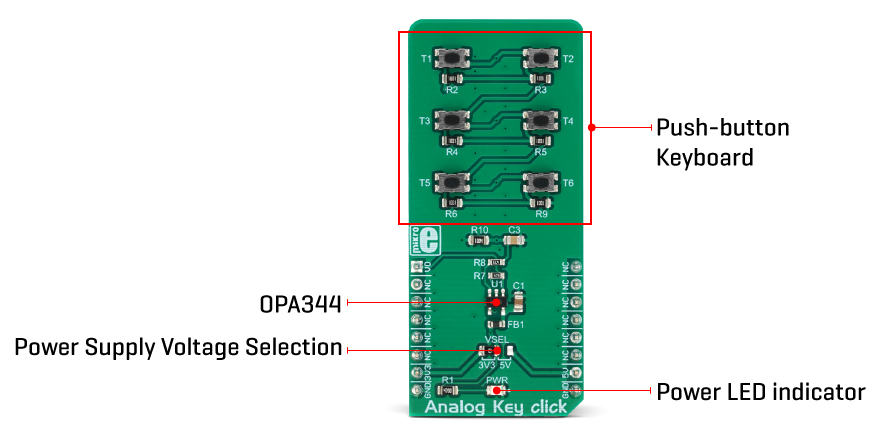
1. **Click** on the “Run” button located beside the Menu Tab to execute the “**Pot.py**” file. It is observed that the digital value and analog voltage are printed at the output console window.
2. **Turn** the knob located on the **Pot Click** clockwise and anti-clockwise to see the changes in the digital value and analog voltage printed at the output console window. Do note that the range of the digital value is 0-1 while the range of the analog voltage is 0-1.8 v.

## Understanding of Analog Key Click Hardware Connection

**Analog Key Click** is an analog keyboard which contains six tactile pushbuttons that is used to select one of six different voltage levels. The idea behind this click is very simple: six resistors form a voltage divider. The resistors are connected in series between the VCC and the GND. Each button selects one of the six middle taps, allowing six different voltage levels to be selected. The voltage is available at the AN pin of the mikroBUS, which is additionally protected by an operational amplifier, configured as a buffer. This allows both protection and a proper impedance at the analog input pin of the microcontroller.

This type of keyboard can be used as password terminals for small alarm systems, for selecting an option in various embedded applications, and for all kinds of small DIY projects where low pin count is a big concern.

Analog Key Click, and its respective schematic are shown in the Figure below.



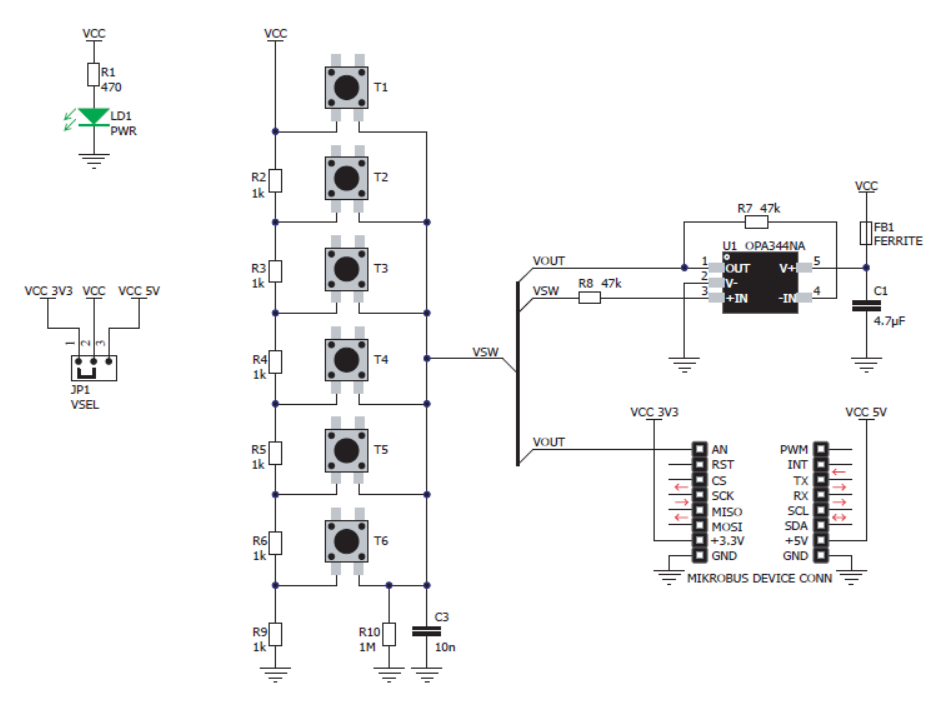


Figure 1.3a: Analog Key Click and Schematic

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

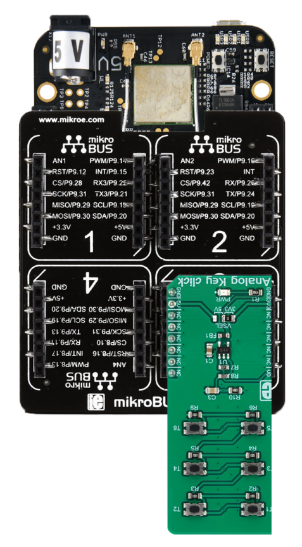


Figure 1.3b: Connecting Analog Key Click to mikroBUS Cape and BBBW Board

## Reading the Analog Data from Analog Key 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 “**AnalogKey.py**”.
3. **Double click** on the newly created file “**AnalogKey.py**” and enter the following code into the file under the Editor section.

|  |
| --- |
| import time  import Adafruit\_BBIO.ADC as ADC  ADC.setup()  while True:  DigitalValue = ADC.read("P9\_40")  if DigitalValue >= 0.00 and DigitalValue < 0.10:  print("No Key is Pressed")  elif DigitalValue > 0.16 and DigitalValue < 0.18:  print("T6 Key is Pressed")  elif DigitalValue > 0.33 and DigitalValue < 0.35:  print("T5 Key is Pressed")  elif DigitalValue > 0.50 and DigitalValue < 0.52:  print("T4 Key is Pressed")  elif DigitalValue > 0.67 and DigitalValue < 0.69:  print("T3 Key is Pressed")  elif DigitalValue > 0.84 and DigitalValue < 0.86:  print("T2 Key is Pressed")  elif DigitalValue > 0.90 and DigitalValue < 1.10:  print("T1 Key is Pressed")  time.sleep(0.3) |

1. **Click** on the “Run” button located beside the Menu Tab to execute the “**AnalogKey.py**” file. It is observed that the text “**No Key is Pressed**” is printed at the output console window.
2. **Press** on each of the Analog Key located on the **Analog Key Click** and **observe** the text printed at the output console window.

# **Displaying the Input Data using BeagleBone Black Wireless (BBBW) Board**

## Developing a Volume Control Indicator using Pot and BarGraph 2 Click

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

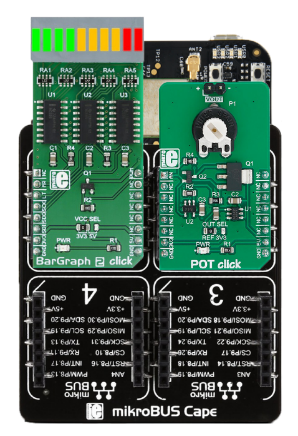


Figure 3.2a: Connecting both the Pot and BarGraph 2 Click to mikroBUS Cape and BBBW Board

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 “**VolumeControlIndicator.py**”.
3. **Double click** on the newly created file “**VolumeControlIndicator.py**” and start entering the code below 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.

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

1. **Enter** the code of the 2 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 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]) |

1. **Enter** the main code that reads the user input from Pot Click and display the corresponding number of led segment bars on the BarGraph 2 Click as shown below.

|  |
| --- |
| ADC.setup()  G\_NumberOfBar = 0  G\_Spi1 = BarGraph2Init()  BarGraph2Display(G\_Spi1, G\_NumberOfBar)  while True:  DigitalValue = ADC.read("P9\_37")  G\_NumberOfBar = int(DigitalValue \* 17)  BarGraph2Display(G\_Spi1, G\_NumberOfBar) |

1. **Click** on the “Run” button located beside the Menu Tab to execute the “**VolumeControlIndicator.py**” file.
2. **Turn** the knob located on the Pot Click clockwise and anti-clockwise.
3. **Observe and compare** the output on theBarGraph 2 Click with your teammates and **consult** your lecturer for advice if it is not the same.
4. **Write** the observation in the white box below for future reference if needed.

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

## Tinkering Time

1. Together with a classmate or two, **think** of a simple application that can use either the Pot Click or Analog Key Click to capture user’s input and feedback to user through BarGraph 2 Click, 7Seg Clicks, 8x8 R Click or OLED B Click.
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 Lab3b. Good job! Take a good break and stay tune for next lab. More exciting lab exercises coming to you!*