

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Fundamentos Embebidos

Práctica 3:

Bluetooth Raspberry Pi

Integrantes:

Cárdenas Cárdenas Jorge

Garrido Sánchez Samuel Arturo

Murrieta Villegas Alfonso

Reza Chavarría Sergio Gabriel

Valdespino Mendieta Joaquín

# Práctica 3: Bluetooth Raspberry Pi

## Objetivo:

Comprobar el funcionamiento de los puertos GPIO en la tarjeta Raspberry pi, mediante la utilización del módulo BLUEDOT para PYTHON y su aplicación para Android.

## Introducción:

Una de las características más importantes de los sistemas embebidos es que disponen de una amplia variedad de formas de conexión y esto se debe principalmente a que deben ser versátiles al momento de poder realizar diferentes acciones tanto local como remotamente.

Precisamente una de las principales formas de conexión es el bluetooth que es de manera general una forma de red inalámbrica creada por Bluetooth Special Group a finales de los 90 que se caracteriza principalmente por poder transmitir datos y voz entre diferentes dispositivos a una distancia relativamente cercana pero con gran velocidad.

## Desarrollo y resultados:

### 0. Previo

Como parte anterior a realizar nuestros ejercicios, en primer momento tuvimos que instalar la biblioteca de bluedot para de esa forma poder realizar la conexión con nuestro teléfono Android:



```
pi@murry: ~/Downloads/embedded_systems/practices/practice_3/code
File Edit Tabs Help
pi@murry:~/Downloads/embedded_systems/practices/practice_3/code $ sudo pip3 install bluedot
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting bluedot
  Downloading https://files.pythonhosted.org/packages/f6/4c/dc4d5685cafad4bd2d8ce3bdb251787de5198a81798da0eee76066f9c803/bluedot-2.0.0-py3-none-any.whl
Installing collected packages: bluedot
Successfully installed bluedot-2.0.0
pi@murry:~/Downloads/embedded_systems/practices/practice_3/code $
```

Imagen 1: Instalación de la biblioteca bluedot

Posteriormente, y con base en la documentación de la biblioteca de bluedot, procedimos a conectar el bluetooth de la raspberry con nuestro teléfono.

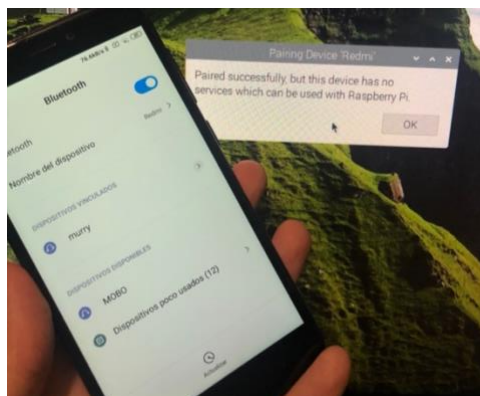


Imagen 2: Conexión exitosa entre nuestra raspberry y teléfono android

## 1. Ejemplos

A continuación, se muestran los resultados obtenidos de cada una de las actividades:

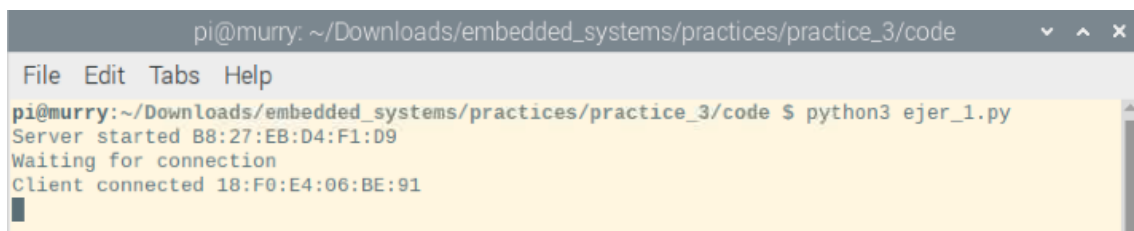
### 1. Ejercicio 1

Para nuestro ejercicio 1 del apartado de ejemplos observamos que es un código simple en donde solamente tenemos dos funciones que imprimen en consola una vez que han sido llamadas, por otro lado, y como el main code del ejercicio instanciamos el objeto BlueDot que es el encargado de la conexión y control mediante el teléfono.

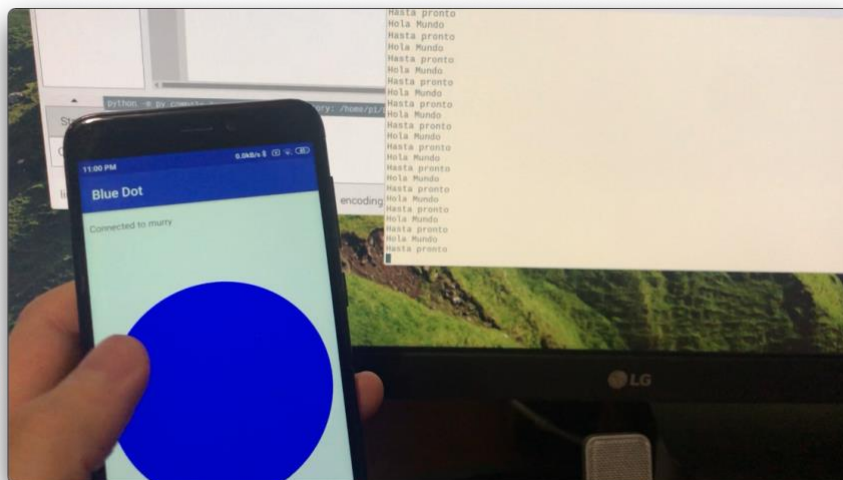
```
1  from blueDot import BlueDot
2  from signal import pause
3  def saludo():
4      print("Hola Mundo")
5
6  def despedida():
7      print("Hasta pronto")
8
9  bd = BlueDot()
10 bd.when_pressed = saludo
11 bd.when_released = despedida
12 pause()
```

*Imagen 3: Código del programa*

A continuación, se muestra el resultado obtenido:



*Imagen 4: Ejecución de nuestro código*



*Imagen 5: Resultado obtenido en la consola y teléfono*

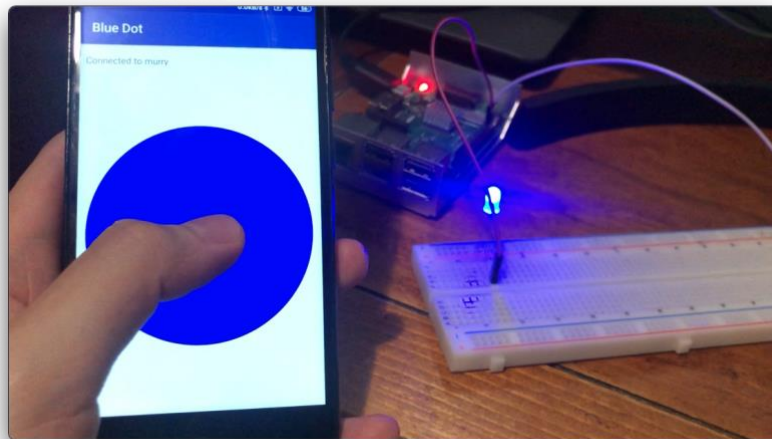
## 2. Ejercicio 2

Para nuestro ejercicio 2 comentamos la primera línea sobre todo porque la biblioteca OS no se empleaba, por otro lado, el presente ejercicio consistió en prender un led considerando el si estaba o no oprimido el pad de blue dot, solo que cabe mencionar que este código realmente no hace una condición directa sino indirecta debido a que se condiciona el estado del led debido a la secuencia de ejecución del código más no porque dependa del estado del pad.

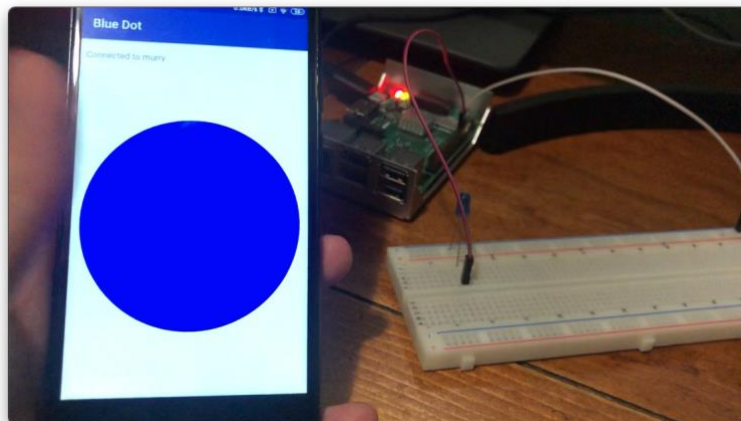
```
1 #import os
2 from blue dot import BlueDot
3 from gpiozero import LED
4
5 bd = BlueDot()
6 led = LED(17)
7 bd.wait_for_press()
8 led.on()
9 bd.wait_for_release()
10 led.off()
```

*Imagen 6: Código del programa*

A continuación, se muestra el resultado obtenido:



*Imagen 7: Resultado obtenido tras presionar el pad de nuestro blue dot*



*Imagen 8: Resultado obtenido tras soltar el pad de nuestro blue dot*

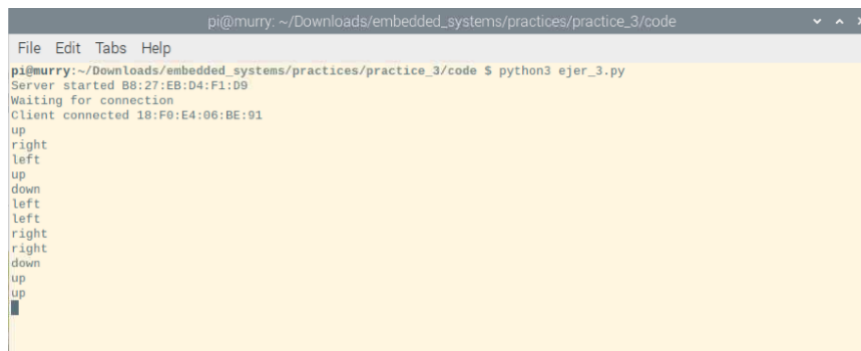
### 3. Ejercicio 3

Para nuestro ejercicio 3 del apartado de ejemplos observamos que es un código simple en donde solamente tenemos una función que al pasarle como parámetro la posición determina la posición en la que se está presionando el pad del bluetooth para así posteriormente imprimir en consola el lugar presionado:

```
1 from bluepy import BlueDot
2 from signal import pause
3
4 def dpad(pos):
5     if pos.top:
6         print("up")
7     elif pos.bottom:
8         print("down")
9     elif pos.left:
10        print("left")
11    elif pos.right:
12        print("right")
13    elif pos.middle:
14        print("fire")
15
16 bd = BlueDot()
17 bd.when_pressed = dpad
18 pause()
```

Imagen 9: Código del programa

A continuación, se muestra el resultado obtenido:



```
pi@murry: ~/Downloads/embedded_systems/practices/practice_3/code
File Edit Tabs Help
pi@murry:~/Downloads/embedded_systems/practices/practice_3/code $ python3 ejer_3.py
Server started B8:27:EB:D4:F1:D9
Waiting for connection
Client connected 18:F0:E4:06:BE:91
up
right
left
up
down
left
left
right
right
down
up
up
```

Imagen 10: Ejecución de nuestro código

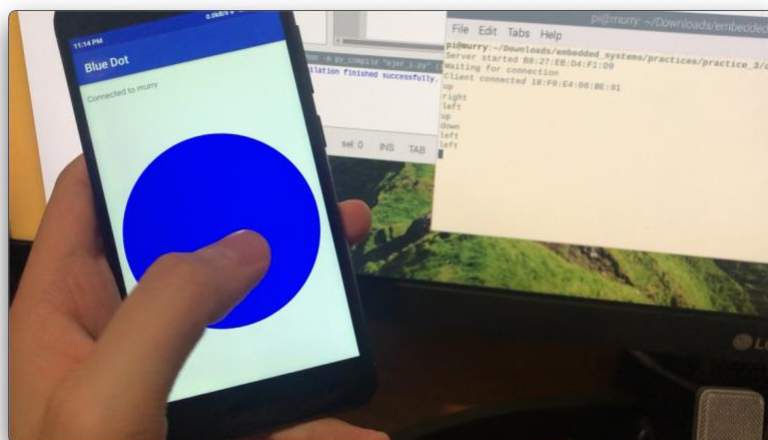


Imagen 11: Resultado obtenido tras oprimir el pad en ciertas parte del mismo

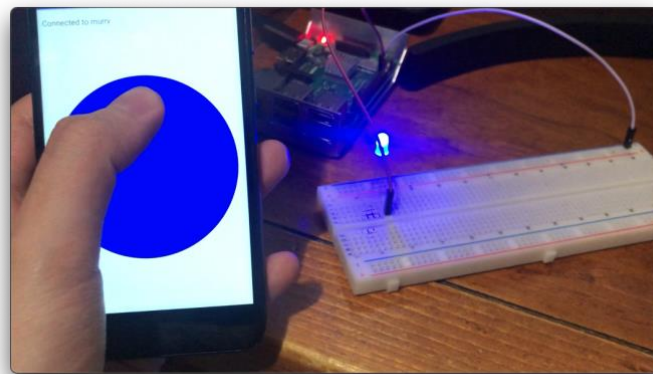
#### 4. Ejercicio 4

Para el ejercicio 4 se empleó nuevamente la posición solo con el objetivo de posteriormente ser usado como variable de intensidad lumínica en el intervalo entre 0 y 1 donde 0 era el valor mínimo obtenido (parte inferior del pad), mientras que el 1 era el valor máximo (Parte superior del pad).

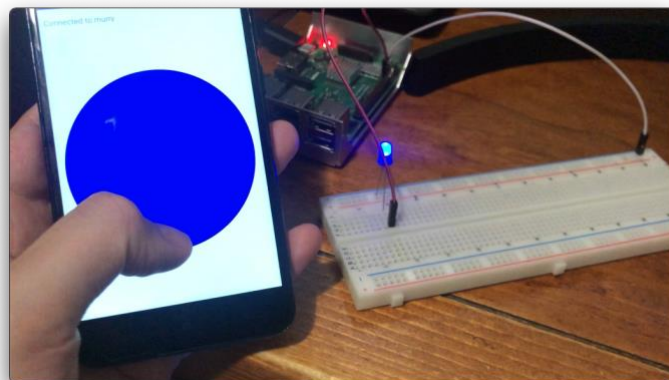
```
1 from bluepy import BlueDot
2 from gpiozero import PWMLED
3 from signal import pause
4 def set_brightness(pos):
5     brightness = (pos.y + 1) / 2
6     led.value = brightness
7     print(brightness)
8
9 led = PWMLED(17)
10 bd = BlueDot()
11 bd.when_moved = set_brightness
12 pause()
```

*Imagen 12: Código del programa*

A continuación, se muestra el resultado obtenido:



*Imagen 13: Resultado obtenido tras presionar en la parte superior el pad, es decir, la máxima intensidad en el led*



*Imagen 14: Resultado obtenido tras presionar en la inferior superior el pad, es decir, la mínima intensidad en el led*



## 2. Ejercicios

### 1. Programa alternativo (Buzzer)

Debido a problemas que tuvo el equipo para emplear un display de 7 segmentos (El display tuvo problemas de fabrica), se desarrolló un programa para emplear el bluedot como forma de prender un buzzer so sumbador electrónico.

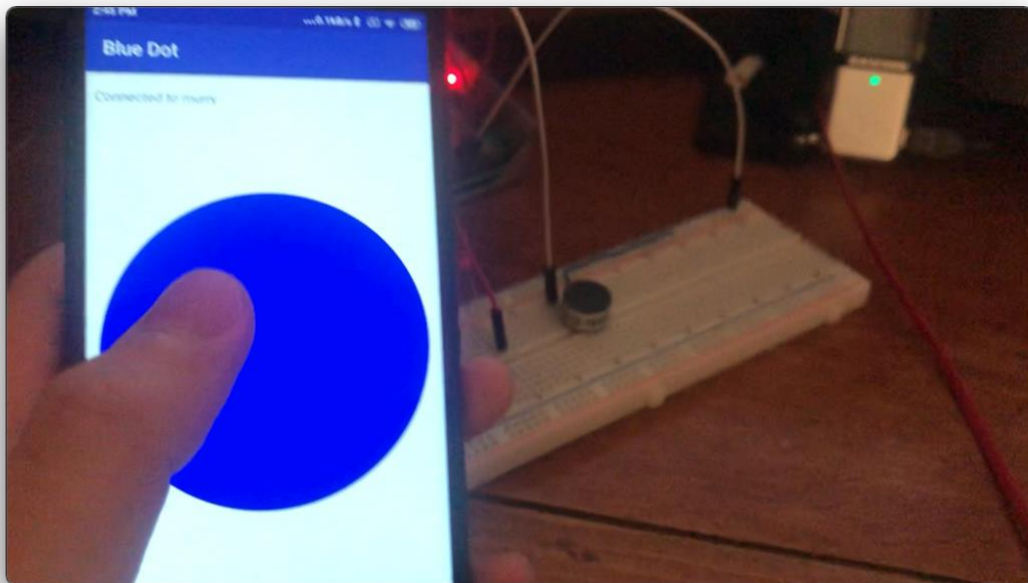
Para el código nos basamos en el ejercicio 2 solamente que en vez de usar un objeto de tipo LED usamos un objeto de tipo Buzzer.

A continuación, se muestro código en Python:

```
1 from gpiozero import Buzzer
2 from bluedot import BlueDot
3
4 bd = BlueDot()
5 bz = Buzzer(26)
6 bd.wait_for_press()
7 bz.on()
8 bd.wait_for_release()
9 bz.off()
```

*Imagen 15: Código del programa*

A continuación, se muestran los resultados en físico o hardware:



*Imagen 16 : Se observa el uso del buzzer a través del bluedot*

## 2. Ejercicio 2: Semáforo que cambia vía bluetooth

Para el ejercicio 2 nos basamos en el ejercicio la práctica 2 del semáforo solo con la diferencia de que la condición o valor que iba a cambiar los estados de nuestro semáforo en esta ocasión sería el valor de la posición dado por nuestro pad.

Por último, el rango del pad para poder determinar qué leds iban o no estar prendidos fue el de mayor a 0.6 verde, entre 0.4 y 0.6 amarillo y menor a 0.4 rojo.

A continuación, se muestra el código del programa:

```
1 #Construya un semáforo peatonal utilizando sus leds y
2 #protoboard, dicho semáforo debe cambiar cuando se presione el BlueDot.
3 from bluepy import BlueDot
4 from signal import pause
5 from gpiozero import LEDBoard
6 leds = LEDBoard(17,27,22)
7
8 def state(pos):
9
10     dimmValue = (pos.y + 1) / 2
11     if dimmValue > 0.6:
12         print("Verde")
13         leds.value = (1,0,0)
14     elif dimmValue >= 0.4 and dimmValue <= 0.6:
15         print("Amarillo")
16         leds.value = (0,1,0)
17     elif dimmValue < 0.4:
18         print("Rojo")
19         leds.value = (0,0,1)
20     print(dimmValue)
21
22
23 leds.value = (1,0,0)
24 bd = BlueDot()
25 bd.when_pressed = state
26 pause()
```

Imagen 17: Código del programa

A continuación, se muestran los resultados en físico o hardware:

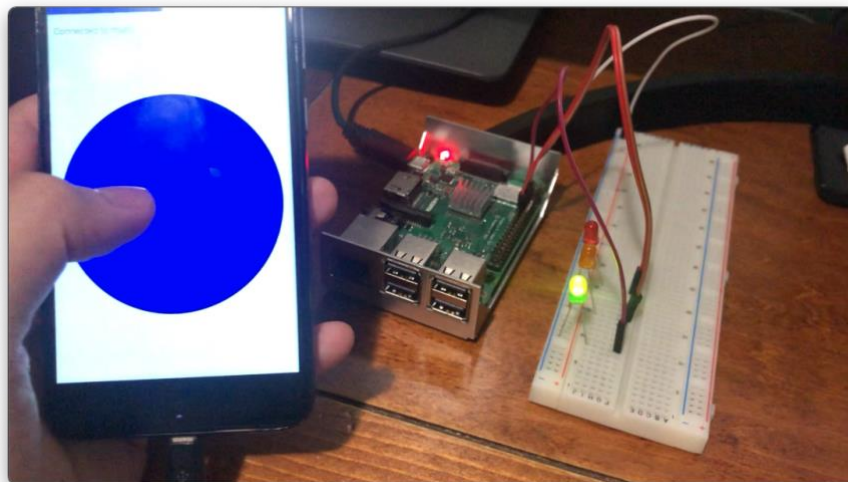
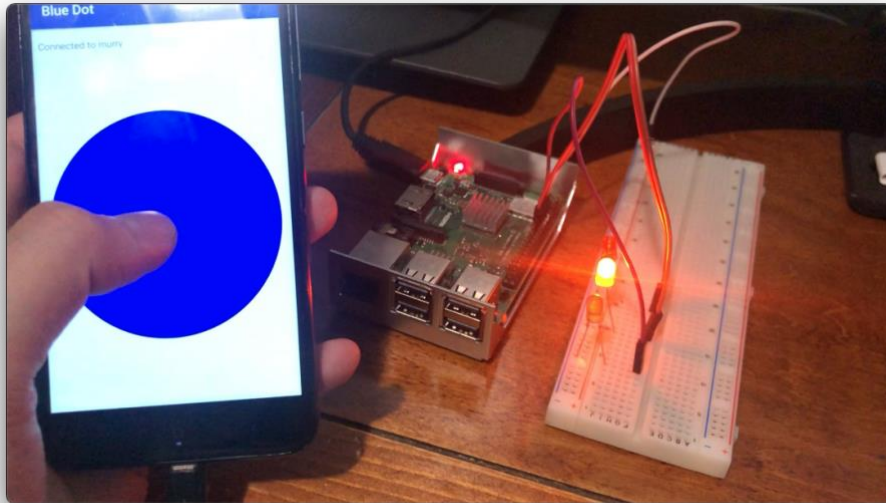
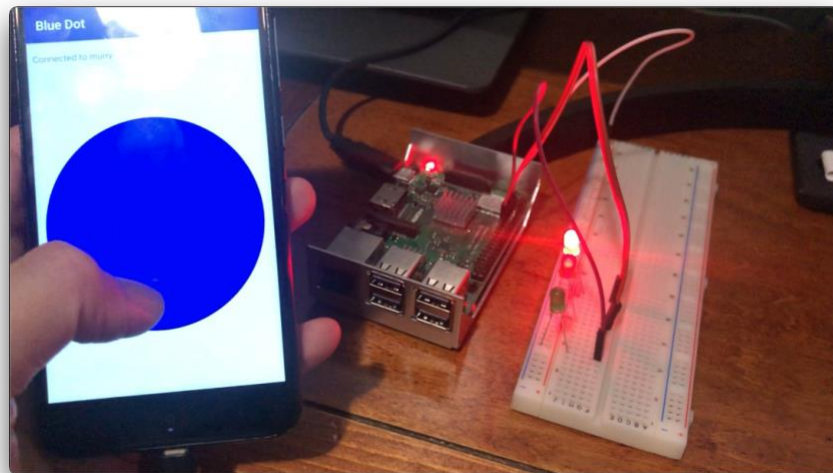


Imagen 18 : Se observa el uso del pad para establecer el color verde como encendido





*Imagen 19 : Se observa el uso del pad para establecer el color amarillo como encendido*



*Imagen 20 : Se observa el uso del pad para establecer el color rojo como encendido*

### 3. Ejercicio 3: Atenuación y cambio de colores vía bluetooth

Para el ejercicio 3 empleamos directamente el código anterior con el objetivo de poder usar la variable posición como parte de la iteración de los colores, la diferencia, es que además de cambiar el color agregamos algunos cambios para que la posición también estableciera el brillo de cada uno de los leds

A continuación, se muestra el código del programa:

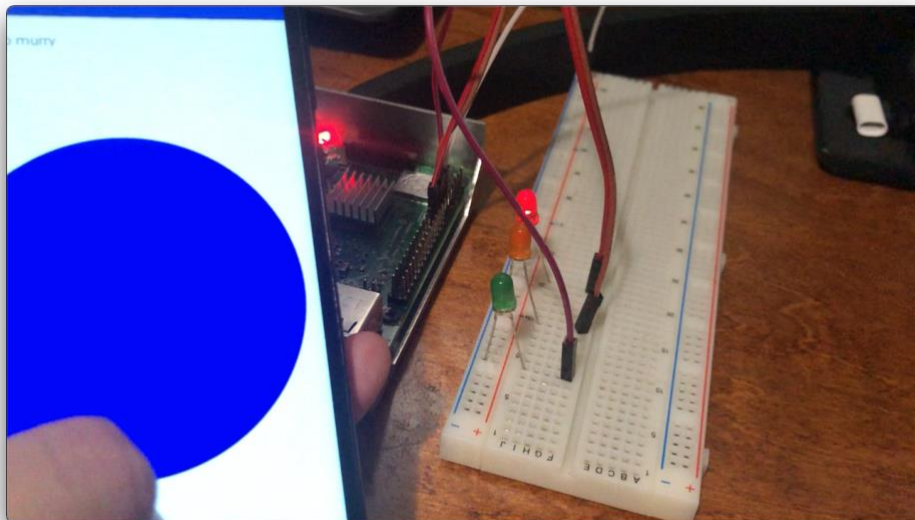
```

1  from bluedot import BlueDot
2  from gpiozero import PWMLED
3  from signal import pause
4
5  def set_brightness(pos):
6      brightness = (pos.y + 1) / 2
7
8      if brightness > 0.6:
9          print("Verde")
10         led1.value = brightness
11         led2.value = 0
12         led3.value = 0
13
14         elif brightness >= 0.3 and brightness <= 0.6:
15             print("Amarillo")
16             led1.value = 0
17             led2.value = brightness
18             led3.value = 0
19
20         elif brightness < 0.3:
21             print("Rojo")
22             led1.value = 0
23             led2.value = 0
24             led3.value = brightness
25         print(brightness)
26
27     led1 = PWMLED(17)
28     led2 = PWMLED(27)
29     led3 = PWMLED(22)
30
31     led1.value = 1
32     bd = BlueDot()
33     bd.when_moved = set_brightness
34     pause()

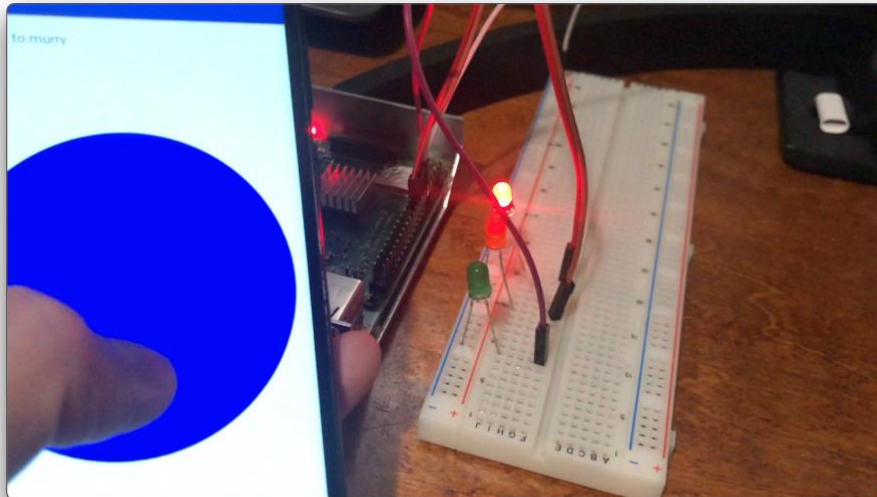
```

*Imagen 21: Código del programa*

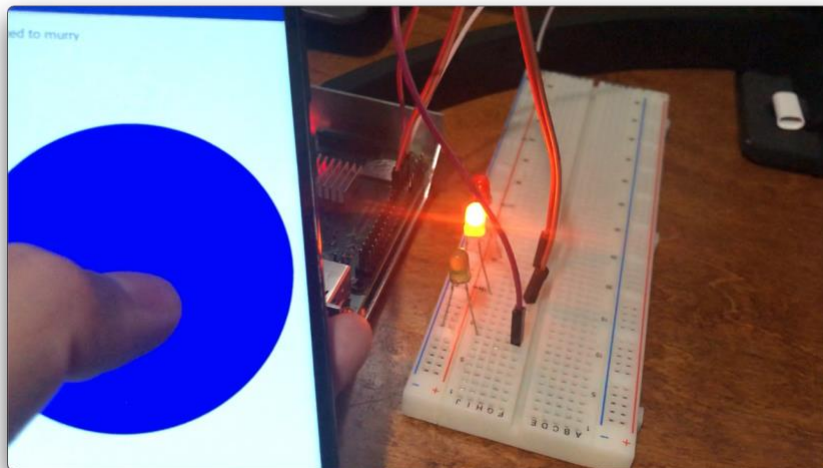
A continuación, se muestran los resultados en físico o hardware, para poder observar a detalle el efecto de la atenuación de cada led se recomienda ver el vídeo de la práctica:



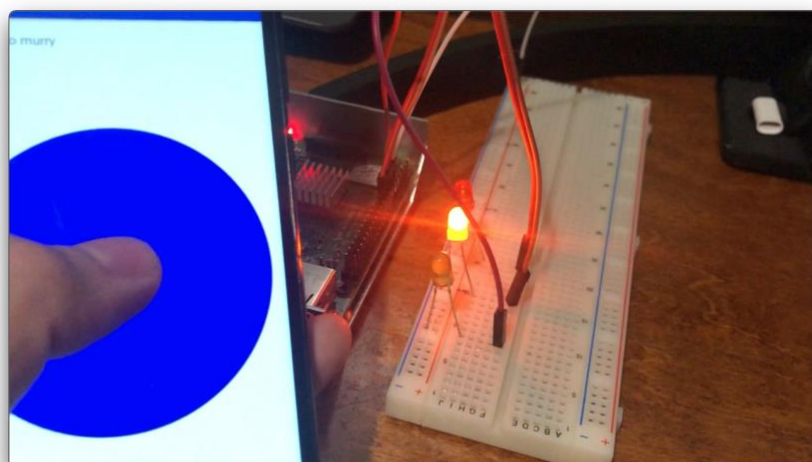
*Imagen 22 : Se observa el uso del pad para establecer el color rojo a baja intensidad como encendido*



*Imagen 23 : Se observa el uso del pad para establecer el color rojo a alta intensidad como encendido*



*Imagen 24 : Se observa el uso del pad para establecer el color naranja a baja intensidad como encendido*



*Imagen 235: Se observa el uso del pad para establecer el color naranja a alta intensidad como encendido*

## **Conclusiones:**

### **Cárdenas Cárdenas Jorge**

Esta práctica nos permitió implementar de manera muy sencilla el control por bluetooth de nuestra Raspberry Pi, esto mediante el uso de la biblioteca Bluedot que provee una API de comunicación entre un dispositivo móvil con bluetooth y la tarjeta mediante funciones en Python. Aunque se tratan de ejercicios muy sencillos como prender leds, son un primer acercamiento bastante interesante al manejo y control de los sistemas embebidos de forma remota, a la par de que permiten reafirmar los conocimientos adquiridos las prácticas anteriores, así como ampliar el conocimiento y manejo de las bibliotecas en Python que nos permitirán desarrollar el proyecto final.

### **Garrido Sánchez Samuel Arturo**

El control de interfaces de hardware siempre son el principal objetivo de los sistemas embebidos, ya sea para desplegar o recabar información. La mayoría de las veces puede realizarse a través de una conexión alámbrica, pero se puede hacer uso de los elementos inalámbricos como bluetooth, que gracias a que la Raspberry cuenta con dicha característica nos fue posible utilizar la aplicación para Android Bluedot que envía la información para la Raspberry y así poder desplegar los leds de la manera en que se lo indiquemos.

### **Murrieta Villegas Alfonso**

En la presente práctica aprendimos a emplear uno de los métodos o formas de conexión inalámbrica más comunes de los sistemas embebidos, el bluetooth, es sin duda un recurso más que valioso principalmente por su facilidad de uso además por la cantidad de dispositivos que hoy en día emplean este método de conexión para transmitir datos.

Por último, empleando la biblioteca de Bluedot es como aprendimos a emplear esta conexión para poder interactuar desde nuestro teléfono con diverso hardware que teníamos conectado a nuestra Raspberry Pi .

### **Reza Chavarria Sergio Gabriel**

Mediante el uso de la comunicación Bluetooth nos permitió el control y manejo de los diferentes programas de manera remota. Esto permitirá expandir la comunicación remota para los futuros proyecto mediante la biblioteca de BlueDot y así hacer que nuestro sistema a desarrollar como proyecto final tenga posibilidad de la interacción con los elementos conectados y los elementos de comunicación a distancia.

### **Valdespino Mendieta Joaquín**

En la presente practica pudimos tener control sobre los puertos, configurándolos y dándoles una funcionalidad, tal que pudiesen realizar diferentes tareas de forma remota con la conexión Bluetooth, cosa que puede ampliar el campo de aplicaciones que pueden realizar con la raspberry, aunque es fundamental hoy en día, todo ello gracias al desarrollo de bibliotecas como BlueDot para su implementación, pudiendo comprender y realizar dicha conexión, ademas de los ejercicios correspondientes de manera satisfactoria.

## **Referencias:**

1. GPIO ZERO Docs. *gpiozero*. Recuperado el 21 de Septiembre de 2021, de <https://gpiozero.readthedocs.io/en/stable/>
2. Raspberry Pi. *Raspberry Pi OS*. Recuperado el 11 de Septiembre de 2021, de <https://www.raspberrypi.org/software/>