Bluetooth Examples

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #31

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Introduction:

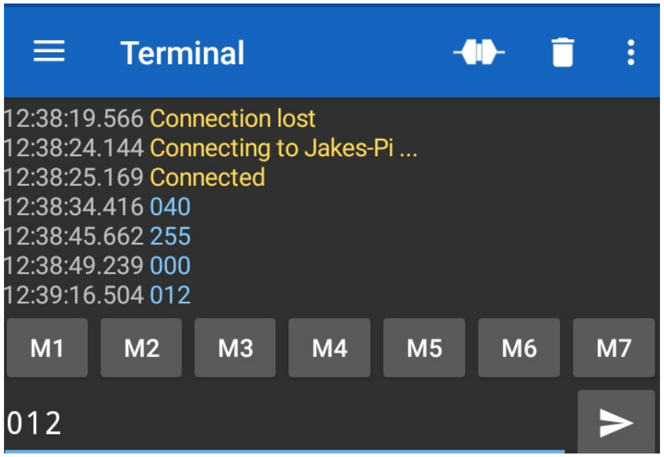
Previous Lecture:

- Send data to your cell phone
- Receive data from your cell phone
- Using bluetooth

In this lecture, we'll build on this to:

- Set the brightness of a NeoPixel string
- Set the color of a NeoPixel string, and
- Control a strobe light

using your cell phone and a bluetooth interface



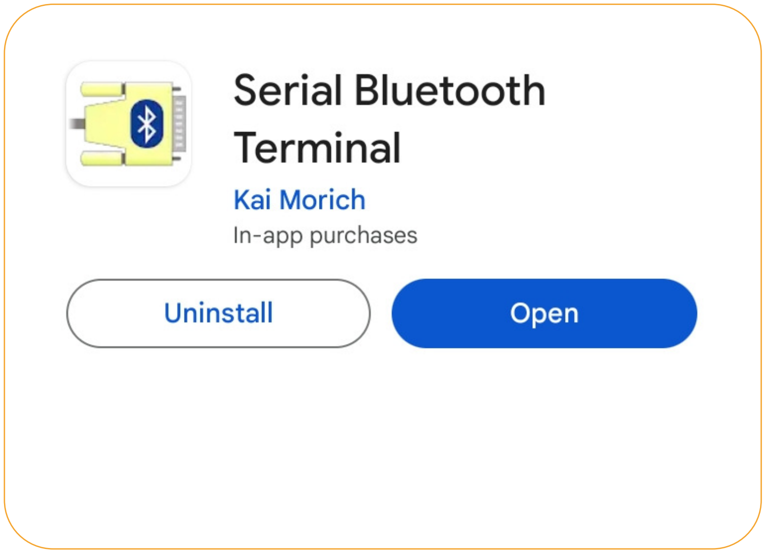
```
12:38:19.566 Connection lost
12:38:24.144 Connecting to Jakes-Pi ...
12:38:25.169 Connected
12:38:34.416 040
12:38:45.662 255
12:38:49.239 000
12:39:16.504 012
```

012

Bluetooth App & Drivers

This lecture assumes

- Serial Bluetooth Terminall app for your cell phone
  - Serial Bluetooth Terminal cell phone app by Kai Morich.
- ble drivers for your Pi-Pico
  - *ble_advertising.py*
  - *ble_simple_peripheral.py*


Serial Bluetooth Terminal

Kai Morich

In-app purchases

Please refer to lecture #30 for details on how to make this bluetooth connection using this app.

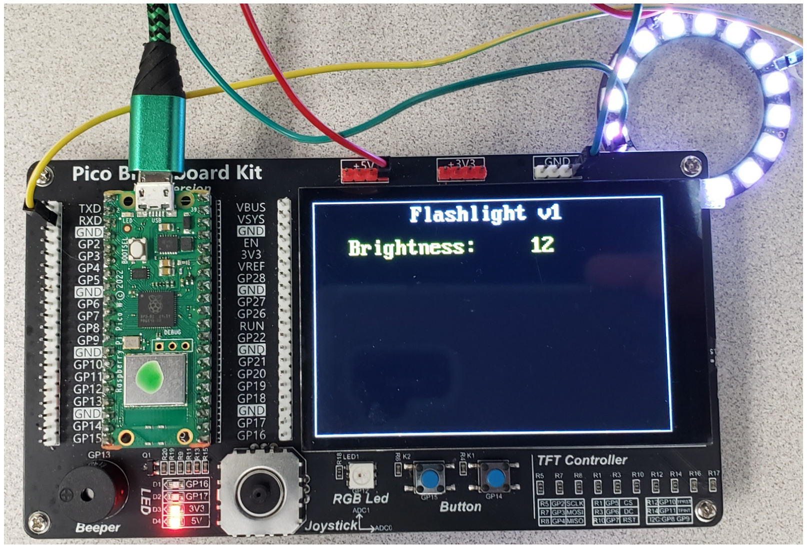
NeoPixel Brightness Control

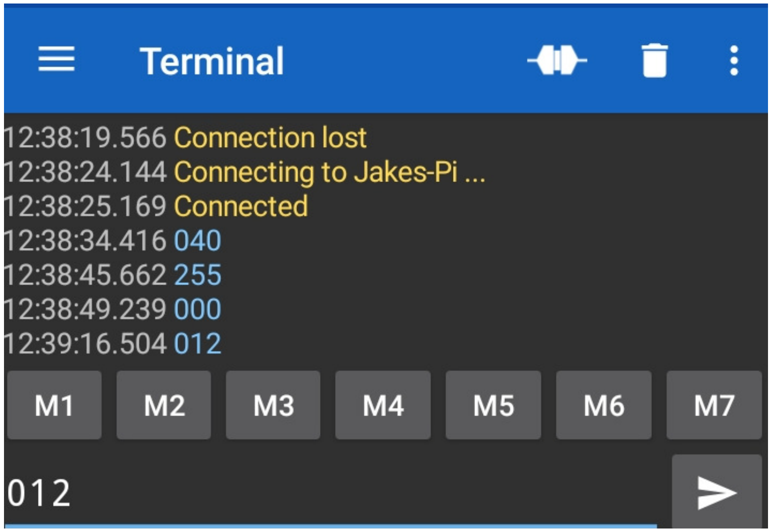
- Flashlight v1.py

Starting out, let's program the Pico to

- Control the brightness of a NeoPixel,
- Using your cell phone
- Through a bluetooth interface.

Assume all LEDs display white light (red / green / blue are the same) with levels from 0 to 255.





```
12:38:19.566 Connection lost
12:38:24.144 Connecting to Jakes-Pi ...
12:38:25.169 Connected
12:38:34.416 040
12:38:45.662 255
12:38:49.239 000
12:39:16.504 012
```

```
012
```

NeoPixel Driver

Step 1: choose a method for driving the NeoPixels.

- Several methods exist
- Let's use the neopixel library that comes with Thonny.
  - Doesn't work with all NeoPixels
  - Does work with the one I'm using

The base code to drive a NeoPixel with 16 elements is as follows:

```
import machine, neopixel

N = 16
p = machine.Pin(1)
np = neopixel.NeoPixel(p, N, bpp=3, timing=1)

np.fill([0,1,2])
np.write()
```

base code for driving a 16-element NeoPixel connected to GP1

Bluetooth Message

- Choose the format for the data that's sent from your cell phone.
  - 000 to 255
- Parse the bluetooth message in on_rx()

For convenience, assume

- Three digits of ASCII text
- Ranging from 000 to 255

on_rx(data):

- Pulls out the first three characters
- Converts to an integer, and

Note:

- Global variables are used to pass data (Level)
- Not the best practice, but works

```
def on_rx(data):
  global Level, flag
  print("Data received: ", data)
  try:
    Level = int(data[0:3])
    Level = min(Level, 255)
    flag
  except:
    print('invalid data entry')
```

Main Routine

Start with the lights being off

Keep checking the bluetooth serial port

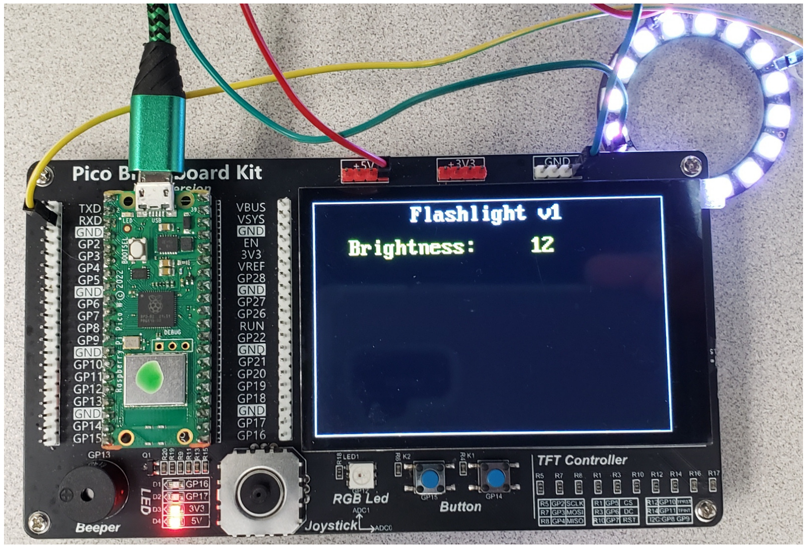
- If a message was received
  - flag = 1
- Set the brightness of the NeoPixel
- Update the LCD display

```
flag = 1
Level = 0
np.fill([Level,Level,Level])
np.write()

while(1):
  if sp.is_connected():
    sp.on_write(on_rx)
  if(flag):
    print('Brightness = ',Level)
    np.fill([Level,Level,Level])
    np.write()
    LCD.Number2(Level, 3, 0, 300, 50, Yellow, Black)
    flag = 0
```

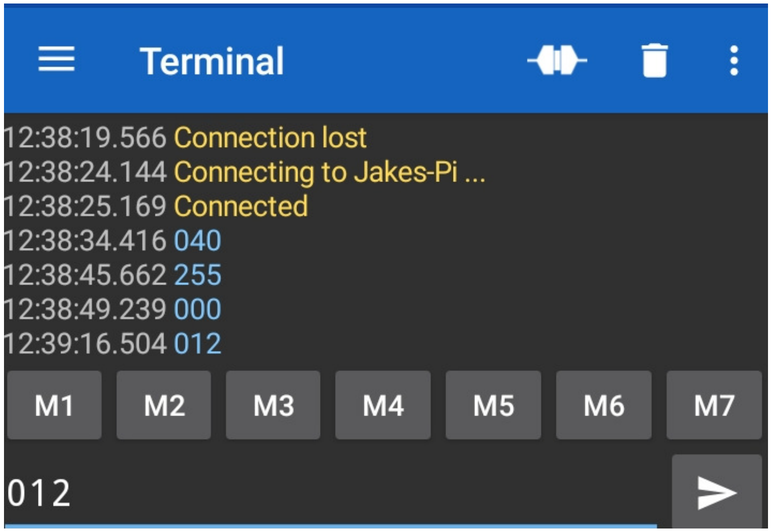
Shell Window

The shell window displays debug information as well. This includes

- The name of the bluetooth connection (64)
- The raw data received, and
- The brightness level pulled from the bluetooth data.



```
Starting advertising
New connection 64
Data received:  b'040\r\n'
Brightness =  40
Data received:  b'255\r\n'
Brightness =  255
Data received:  b'000\r\n'
Brightness =  0
Data received:  b'012\r\n'
Brightness =  12
```



```
12:38:19.566 Connection lost
12:38:24.144 Connecting to Jakes-Pi ...
12:38:25.169 Connected
12:38:34.416 040
12:38:45.662 255
12:38:49.239 000
12:39:16.504 012
```

012

NeoPixel Flashlight (take 2)

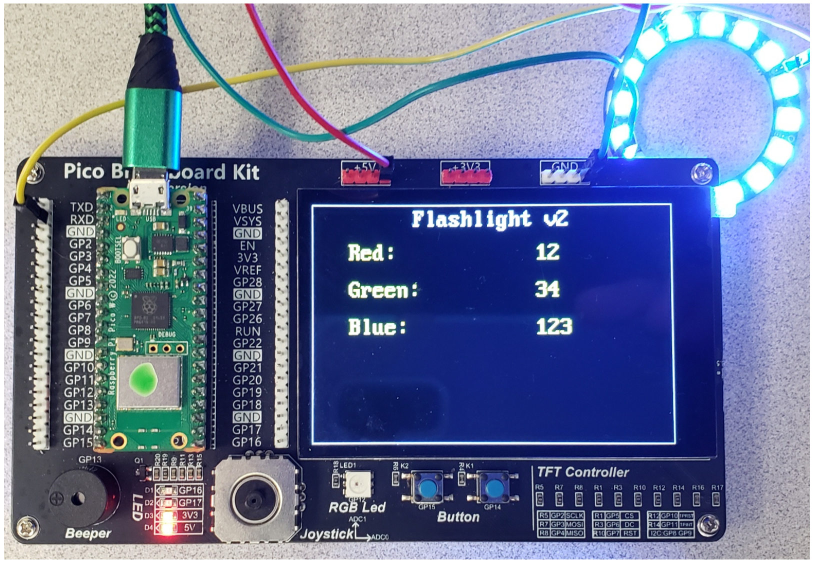
- Flashlight v2.py

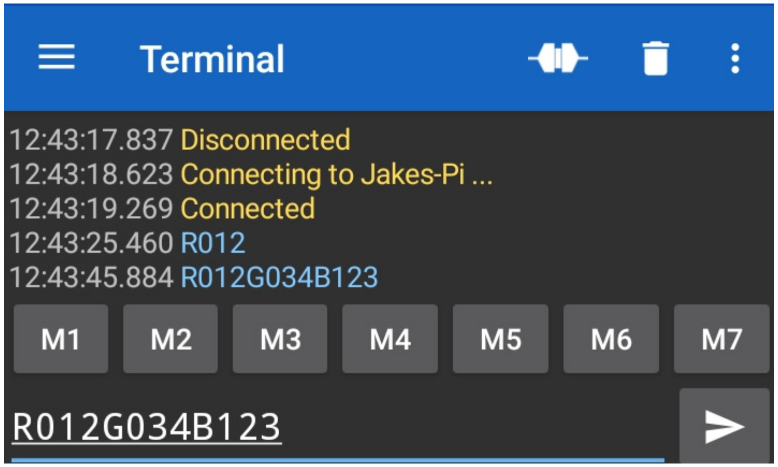
The previous code output white light

- RGB levels all the same

Change the code:

- RGB level for each LED is the same
- The values of RGB are independent
- Allows 16 million colors





```
12:43:17.837 Disconnected
12:43:18.623 Connecting to Jakes-Pi ...
12:43:19.269 Connected
12:43:25.460 R012
12:43:45.884 R012G034B123
```

R012G034B123

Bluetooth Message

Assume the message is of the form

```
RxxxGyyyBzzz
```

The bluetooth rx routine then:

- Pulls out the rgb values
- Assuming a fixed location
- Updates the NeoPixel, and
- Sets a flag

```
def on_rx(data):
  global r, b, b, flag
  print("Data received: ", data)
  try:
    r = int(data[1:4])
    g = int(data[5:8])
    b = int(data[9:12])
    np.fill([r,g,b])
    np.write()
    flag = 1
  except:
    print('format RxxxGxxxBxxx')
```

Note

- All data is returned using globals

Main Routine

Keeps checking the bluetooth serial port

If a message is received (flag is set)

- Update the LCD display

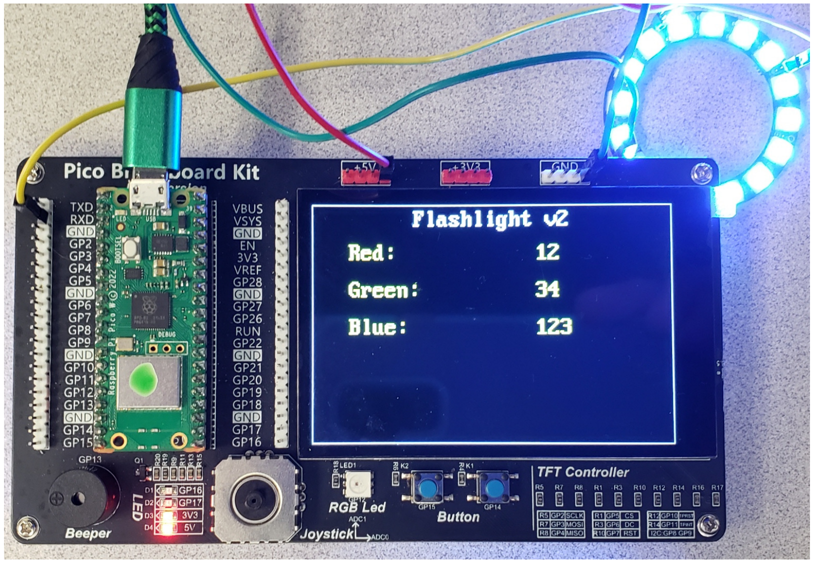
```
flag = 1

while(1):
    if sp.is_connected():
        sp.on_write(on_rx)
    if(flag):
        LCD.Text2(str(r) + '  ', 300, 50, Yellow, Black)
        LCD.Text2(str(g) + '  ', 300, 100, Yellow, Black)
        LCD.Text2(str(b) + '  ', 300, 150, Yellow, Black)
        flag = 0
```

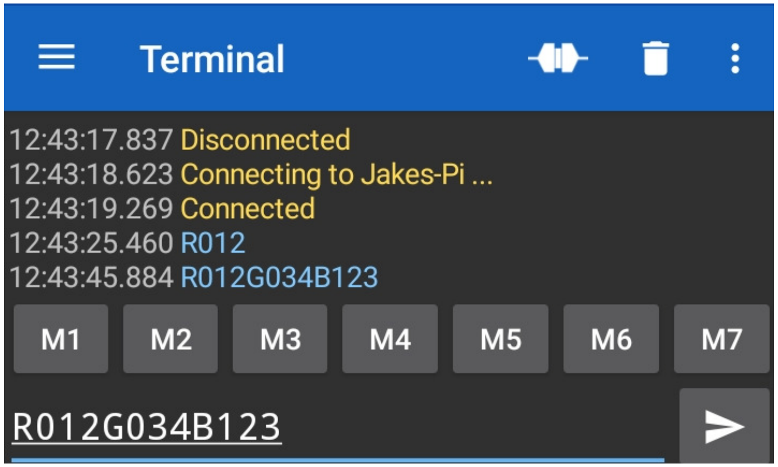
Shell Window

The shell window is just used for debugging. This shows

- The raw message received from your cell phone, and
- The resulting red / green / blue levels which are pulled from this message.

```
-------------------------------
Data:  b'R012G034B123\r\n'
r =  12  g =  34   b =  123
Data:  b'R000G000B000\r\n'
r =  0  g =  0   b =  0
```





```
12:43:17.837 Disconnected
12:43:18.623 Connecting to Jakes-Pi ...
12:43:19.269 Connected
12:43:25.460 R012
12:43:45.884 R012G034B123
```

R012G034B123

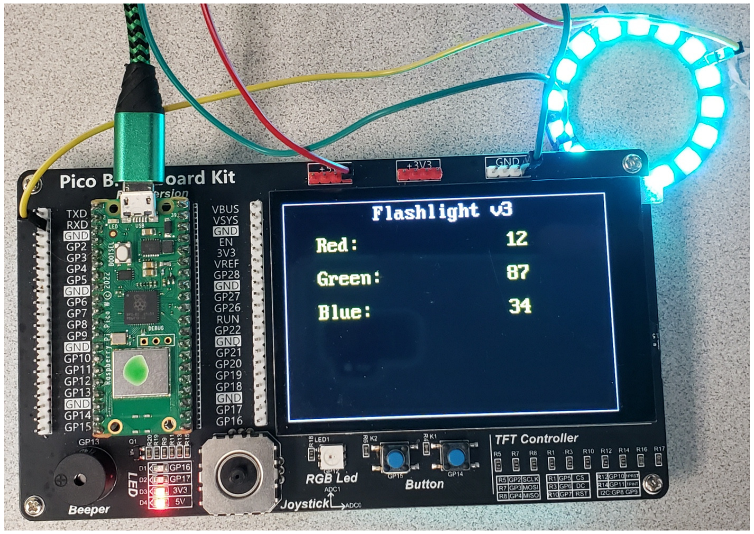
NeoPixel Flashlight (take 3)

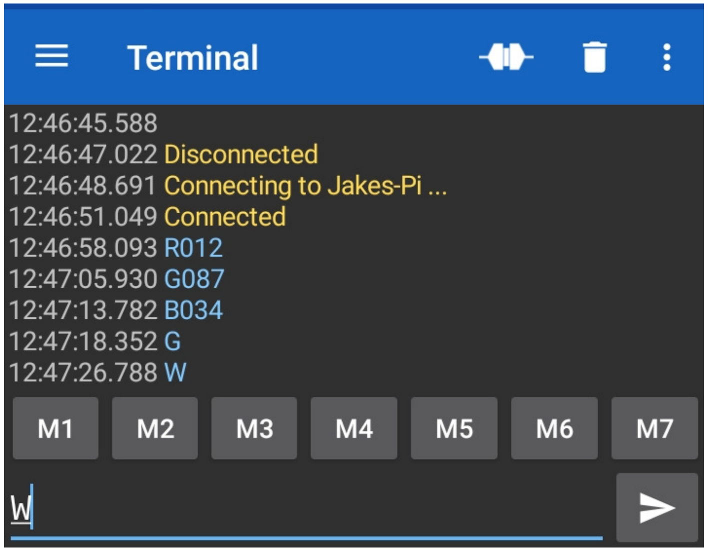
Use instructions to control the NeoPixel

- Rxxx   sets the red color
- Gyyy   sets the green color
- Bzzz   sets the blue color
- C   clear the colors
- W   write to the NeoPixel

Similar to the previous code

- Just a different style of controlling the NeoPixel





```
12:46:45.588
12:46:47.022 Disconnected
12:46:48.691 Connecting to Jakes-Pi ...
12:46:51.049 Connected
12:46:58.093 R012
12:47:05.930 G087
12:47:13.782 B034
12:47:18.352 G
12:47:26.788 W
```

on_rx(data)

When a bluetooth message is received

- Deciper the command
- Set the corresponding color
- Update the NeoPixel if a write (W) command was received

Again, data is returned using globals

- sort of like an interrupt routine

```
def on_rx(data):
  global r,g,b,flag
  print("Data received: ", data)
  try:
    cmd = chr(data[0])
    if(cmd == 'R'):
      r = int(data[1:4])
    if(cmd == 'G'):
      g = int(data[1:4])
    if(cmd == 'B'):
      b = int(data[1:4])
      print('b = ', b)
    if(cmd == 'C'):
      r = g = b = 0
    if(cmd == 'W'):
      np.fill([r,g,b])
      np.write()

    except:
        print ('Rxxx/Gxxx/Bxxx/C/W')
```

Main Routine

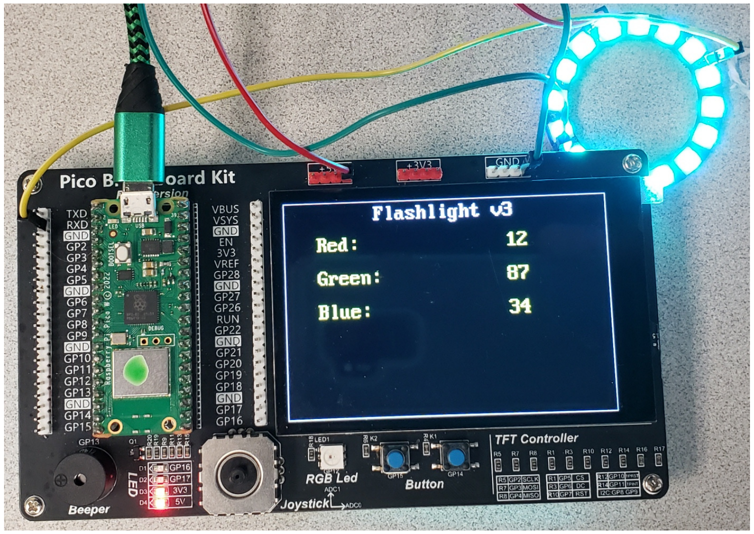
- Passes the text string to the on_rx() routine when received, and
- Displays the brightness on the LCD display

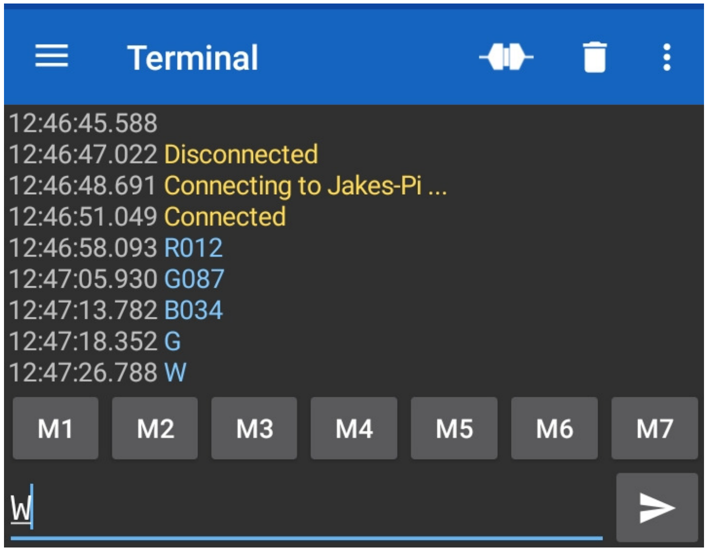
```
flag = 1
r=g=b=0
np.fill([r,g,b])
np.write()

while(1):
    if sp.is_connected():
        sp.on_write(on_rx)
    if(flag):
        LCD.Number2(r, 3, 0, 300, 50, Yellow, Black)
        LCD.Number2(g, 3, 0, 300, 100, Yellow, Black)
        LCD.Number2(b, 3, 0, 300, 150, Yellow, Black)
        flag = 0
```

Shell Window

The shell window displays debug information, such as the raw data received

```
-------------------------

New connection 64
Data received:  b'\r\n'
Data received:  b'R012\r\n'
Data received:  b'G087\r\n'
Data received:  b'B034\r\n'
Data received:  b'W\r\n'
Data received:  b'C\r\n'
```

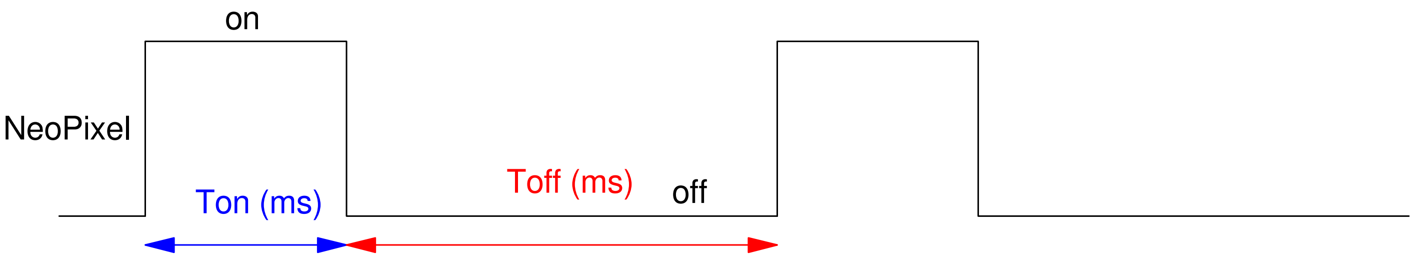




```
12:46:45.588
12:46:47.022 Disconnected
12:46:48.691 Connecting to Jakes-Pi ...
12:46:51.049 Connected
12:46:58.093 R012
12:47:05.930 G087
12:47:13.782 B034
12:47:18.352 G
12:47:26.788 W
```

NeoPixel Strobe Light

Finally, let's build a strobe light. For this function, set the on-time and off-time of the NeoPixels



Assume for this problem that

- All 16 NeoPixels are either off (00/00/00) or on (255/255/255)
- The on time can be adjusted from 1ms to 100ms
- The off time can be adjusted from 1ms to 100ms

Timer Interrupts

Use timer interrupts since timing is important

- Ton, Toff determine the on and off times
- Each interrupt sets up the next interrupt using a one-short
- The strobe light is turned off by skipping one of the one-shot initializations.
  - deinit() would also work

```
tim = Timer()

def L_On(timer):
  global Ton, Status
  tim.init(period = Ton, mode=Timer.ONE_SHOT, callback=L_Off)
  np.fill([255,255,255])
  np.write()

def L_Off(timer):
  global Toff, Status
  if(Status):
    tim.init(period = Toff, mode=Timer.ONE_SHOT, callback=LOn)
  np.fill([0,0,0])
  np.write()
```

on_rx() routine

- Nxxx sets the on time is milliseconds (000 to 999)
- Fxxx sets the off time in milliseconds (000 to 999)
- G turns on the strobe light (go), and
- S stops the strobe light (stop)

```
def on_rx(data):
  global Ton, Toff, Status, flag
  print("Data received: ", data)
  try:
    cmd = chr(data[0])
    if(cmd == 'N'):
      Ton = int(data[1:4])
    if(cmd == 'F'):
      Toff = int(data[1:4])
    if(cmd == 'G'):
      Status = 1
      tim.init(period = Ton, mode=Timer.ONE_SHOT, callback=L_On)
    if(cmd == 'S'):
      Status = 0

  except:
    print('invalid data entry')
```

Main Routine

- Passes data to on_rx() when a bluetooth message is received, and
- Displays the status of the strobe light.

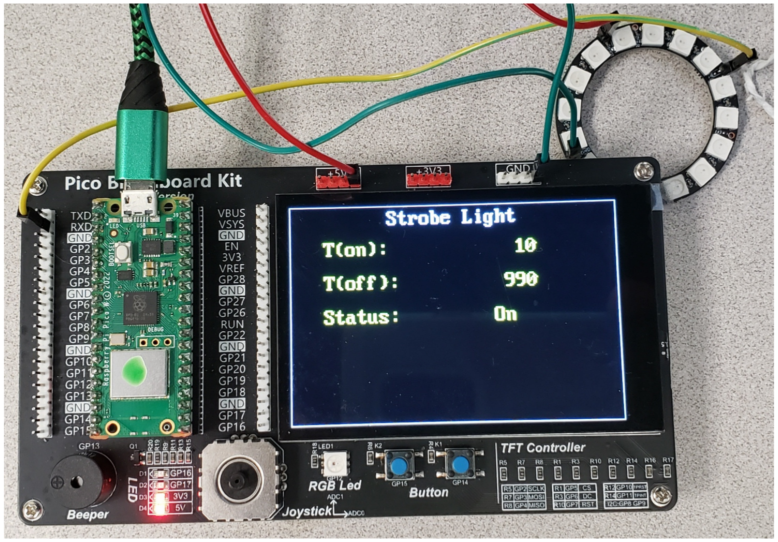
```
np.fill([0,0,0])
np.write()
flag = 1

while(1):
    if sp.is_connected():
        sp.on_write(on_rx)
    if(flag):
        LCD.Number2(Ton, 3, 0, 300, 50, Yellow, Black)
        LCD.Number2(Toff, 3, 0, 300, 100, Yellow, Black)
        if(Status == 1):
            LCD.Text2('On ', 300, 150, Yellow, Black)
        else:
            LCD.Text2('Off', 300, 150, Yellow, Black)
        flag = 0
```

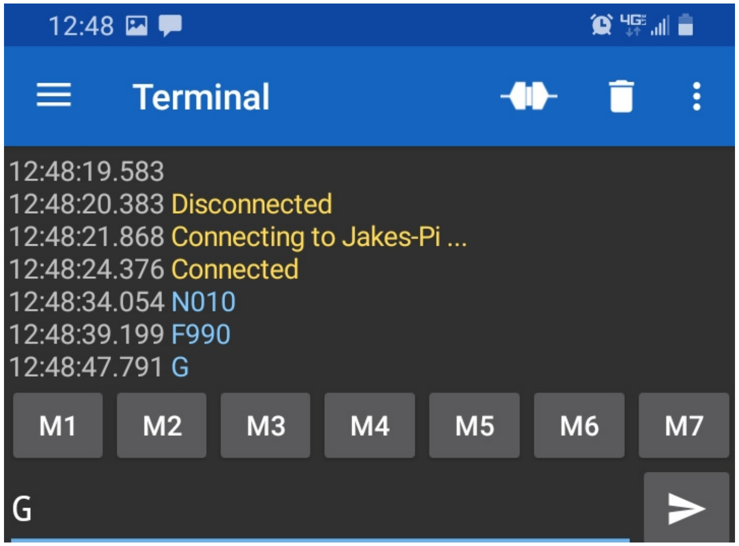
Shell Window

Finally, debug information is displayed in the shell window. This shows the raw messages received from your cell phone.

```
-------------------

Data received:  b'N010\r\n'
Data received:  b'F490\r\n'
Data received:  b'G\r\n'
Data received:  b'S\r\n'
```

Shell window shows the raw messages received on the bluetooth link





```
12:48:19.583
12:48:20.383 Disconnected
12:48:21.868 Connecting to Jakes-Pi ...
12:48:24.376 Connected
12:48:34.054 N010
12:48:39.199 F990
12:48:47.791 G
```

G

Summary

Using the libraries

- *ble_advertising.py*
- *ble_simple_peripheral.py*

you are able to send data from your cell phone to your Pi-Pico. With a little coding, different commands can be sent to the Pico, controlling its operation, such as the brightness, color, or flashing rate of a NeoPixel. Other functions and commands are possible and only limited by the imagination of the programmer.

References

https://electrocredible.com/raspberry-pi-pico-w-bluetooth-ble-micropython