30. Bluetooth

Introduction:

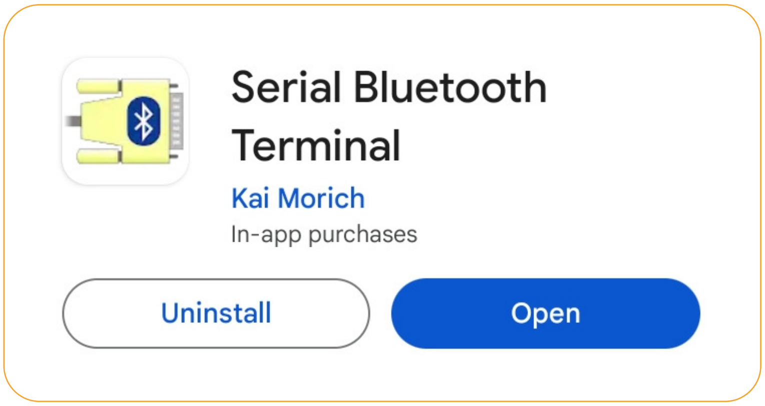
Bluetooth is a way for your Pico board to send and receive data from your cell phone (as well as other devices). With bluetooth, you can

- Send sensor data to you cell phone, such as temperature, pressure, or acceleration readings, or
- Receive data from your cell phone, allowing you to turn on or off lights, set the speed of a motor, and so on.

This lecture presents methods for connecting your Pi-Pico to your cell phone to send and receive data.

Cell Phone App

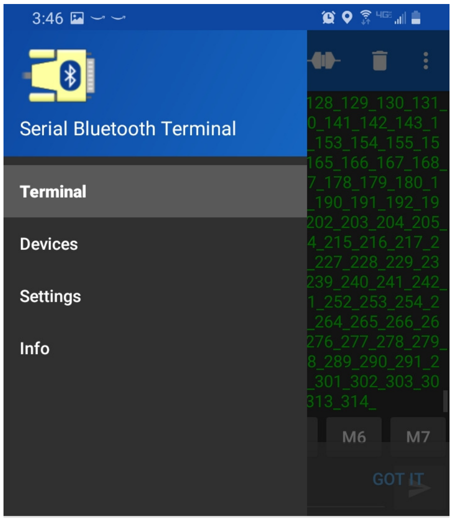
Before sending and receiving data, you need to install a serial bluetooth terminal app on your cell phone. Several exist. The one tested to work with these lecture notes is the Serial Bluetooth Terminal by Kai Morich



Serial Bluetooth Terminal app by Kau Morich

Once you install this app and open it, you will get several options.

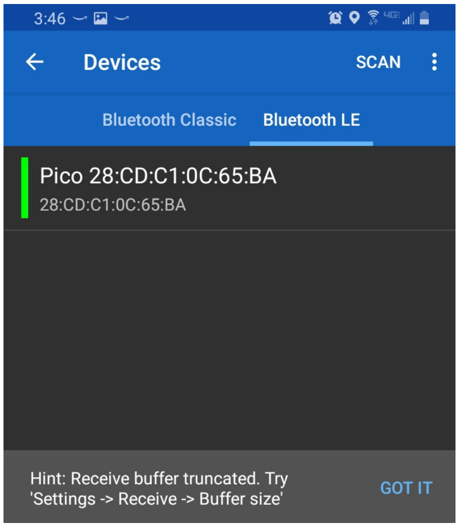
- Terminal allows you to send and receive serial data to your Pico board
- Devices lets you connect (pair) with your Pico board
- Settings allow you to adjust the display
- Info tells you the version you're using and other information



```
128_129_130_131_
0_141_142_143_1
_153_154_155_15
165_166_167_168_
7_178_179_180_1
_190_191_192_19
202_203_204_205_
4_215_216_217_2
_227_228_229_23
239_240_241_242_
1_252_253_254_2
_264_265_266_26
276_277_278_279_
8_289_290_291_2
_301_302_303_30
313_314_
```

Serial Bluetooth Terminal - main menu

Once you run the following programs, you first need to connect to your Pico board. This is done by selecting Devices then selecting your Pico board



Pico 28:CD:C1:0C:65:BA
28:CD:C1:0C:65:BA

Hint: Receive buffer truncated. Try 'Settings -> Receive -> Buffer size'

When you run one of the following programs, a Pico device should show up

Driver Files for your Pico Board

In order to use BlueTooth on your Pico board, two files need to be in the root directory.

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

Open these files and save them to your Pico board using Thonny. Once these files are on your Pico board, you're ready to send and receive serial data.

Bluetooth Transmit

Let's start out with transmitting data. The following program transmits a string to your cell phone.

- Strings are nice since you can see the data on the screen
- Binary data can be seen if you select HEX when receiving data

The following program

- Sets up a bluetooth connection to your cell phone
- Once connected (p.is_connected() == true), a count along with a carriage return and linefeed is sent, once per second.

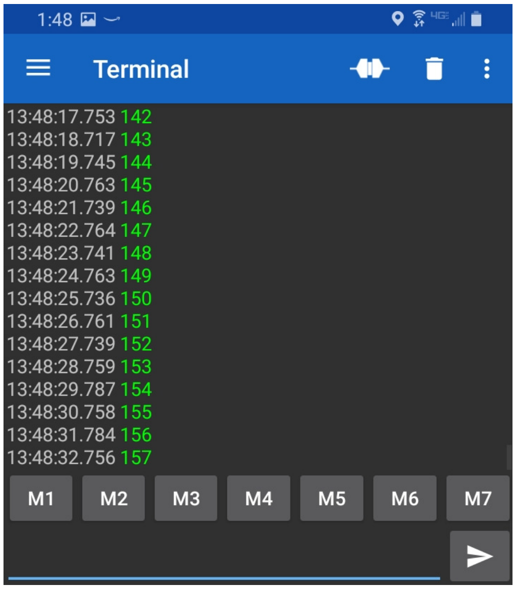
```
import bluetooth
from time import sleep
from ble_advertising import advertising_payload
from ble_simple_peripheral import BLESimplePeripheral

ble = bluetooth.BLE()
p = BLESimplePeripheral(ble)

i = 0
while(1):
    if p.is_connected():
        i += 1
        data = str(i)
        p.send(data + "\r\n")
        print("tx ", data)
    sleep(1)
```

shell

```
tx 1
tx 2
tx 3
```

From the terminal emulator, the result looks like the following with

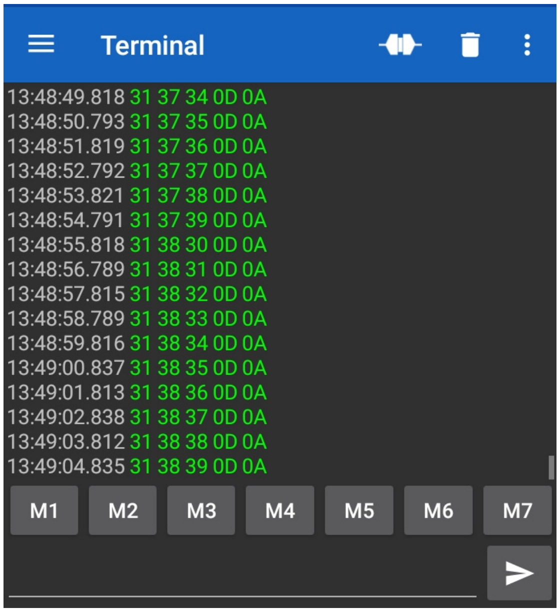
- Settings - Display mode - Text



```
13:48:17.753 142
13:48:18.717 143
13:48:19.745 144
13:48:20.763 145
13:48:21.739 146
13:48:22.764 147
13:48:23.741 148
13:48:24.763 149
13:48:25.736 150
13:48:26.761 151
13:48:27.739 152
13:48:28.759 153
13:48:29.787 154
13:48:30.758 155
13:48:31.784 156
13:48:32.756 157
```

In text mode, the data is displayed as text (green) along with a time-stamp (white)

If you change the display-mode settings to hex, you can see the binary data (useful if the data has non-printable characters, such as a jpg image).



```
13:48:49.818 31 37 34 0D 0A
13:48:50.793 31 37 35 0D 0A
13:48:51.819 31 37 36 0D 0A
13:48:52.792 31 37 37 0D 0A
13:48:53.821 31 37 38 0D 0A
13:48:54.791 31 37 39 0D 0A
13:48:55.818 31 38 30 0D 0A
13:48:56.789 31 38 31 0D 0A
13:48:57.815 31 38 32 0D 0A
13:48:58.789 31 38 33 0D 0A
13:48:59.816 31 38 34 0D 0A
13:49:00.837 31 38 35 0D 0A
13:49:01.813 31 38 36 0D 0A
13:49:02.838 31 38 37 0D 0A
13:49:03.812 31 38 38 0D 0A
13:49:04.835 31 38 39 0D 0A
```

If the display settings are changed to hex, the bytes are displayed message by message

Once you are receiving data, you can change the data you transmit to whatever you like, such as

- Temperature in the room
- Acceleration of the Pi-Pico (has it been moved?)
- Door open or closed (button pressed, not pressed),
- etc.

BlueTooth Receive

In addition to transmitting data to your cell phone, you can receive serial data from your cell phone. This data can be used to control the Pico's operation, such as

- Turn on and off an LED
- Set the color of a NeoPixel
- Turn off an alarm, etc.

In the following program, the Pico is looking for the message "toggle" followed by a carriage return and linefeed. Once detected, the LED on the Pico board is toggled.

```
# Import necessary modules
from machine import Pin
import bluetooth
from ble_simple_peripheral import BLESimplePeripheral

ble = bluetooth.BLE()
sp = BLESimplePeripheral(ble)

led = Pin(16, Pin.OUT)
led.value(0)

# Define a callback function to handle received data
def on_rx(data):
    print("Data received: ", data)
    if data == b'toggle\r\n':
        led.toggle()

while True:
    if sp.is_connected():
        sp.on_write(on_rx)
```

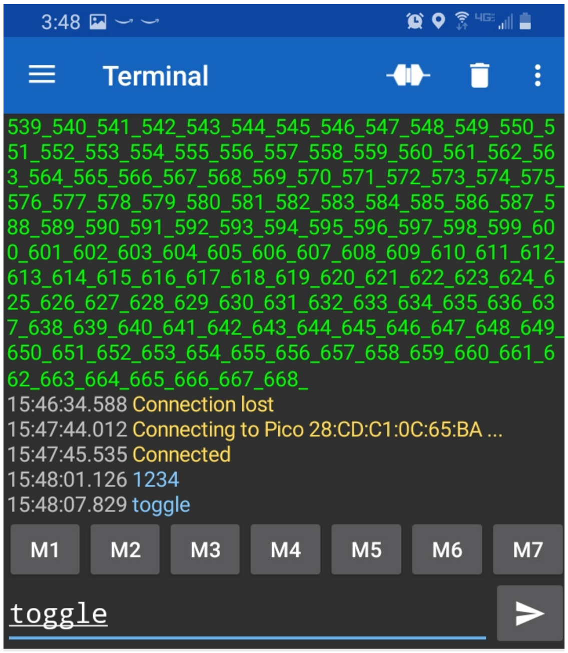
shell

```
Data received:  1234\r\n
Data received:  toggle\r\n
```

Once you run this program and pair with your cell phone, you can send messages.

- On the bottom of the screen, type in a message (such as *1234* or *toggle*) then hit *send* (the arrow to the right)
- On the terminal connected to your Pi-Pico you should see the message *Data received:* along with the message

If the correct sequence is received (*toggle* followed by a carriage return and line feed), the LED connected to GP16 is toggled.



```
539_540_541_542_543_544_545_546_547_548_549_550_5
51_552_553_554_555_556_557_558_559_560_561_562_56
3_564_565_566_567_568_569_570_571_572_573_574_575_
576_577_578_579_580_581_582_583_584_585_586_587_5
88_589_590_591_592_593_594_595_596_597_598_599_60
0_601_602_603_604_605_606_607_608_609_610_611_612_
613_614_615_616_617_618_619_620_621_622_623_624_6
25_626_627_628_629_630_631_632_633_634_635_636_63
7_638_639_640_641_642_643_644_645_646_647_648_649_
650_651_652_653_654_655_656_657_658_659_660_661_6
62_663_664_665_666_667_668_
15:46:34.588 Connection lost
15:47:44.012 Connecting to Pico 28:CD:C1:0C:65:BA ...
15:47:45.535 Connected
15:48:01.126 1234
15:48:07.829 toggle
```

```
toggle
```

You can add short-cuts.

- Press and hold M1. This allows you to insert a message for M1. Click on the check mark when done.
- Press and hold M2 to add a message for M2

Each time you press

- M1 message is sent out along with a carriage return and line feed.
- M2 message is sent out along with a carriage return and line feed

This allows you to output seven messages (toggle seven devices, turn some on, turn some off, etc) with the touch of a button.

For example, the following program allows you to toggle the LEDs and beeper:

- Pico-LED: message = b'LED0\r\n'
- GP16 LED: message = b'LED1\r\n'
- GP17 LED: message = b'LED2\r\n'
- Beeper: message = b'Beep\r\n'

```
# Import necessary modules
from machine import Pin
import bluetooth
from ble_simple_peripheral import BLESimplePeripheral

ble = bluetooth.BLE()
sp = BLESimplePeripheral(ble)

led0 = Pin("LED", Pin.OUT)
led1 = Pin(16, Pin.OUT)
led2 = Pin(17, Pin.OUT)
beeper = Pin(13, Pin.OUT)

# Define a callback function to handle received data
def on_rx(data):
    print("Data received: ", data)
    if data == b'LED0\r\n':
        led0.toggle()
    if data == b'LED1\r\n':
        led1.toggle()
    if data == b'LED2\r\n':
        led2.toggle()
    if data == b'Beep\r\n':
        beeper.toggle()

while True:
    if sp.is_connected():
        sp.on_write(on_rx)
```

shell

```
Data received:  LED0\r\n
Data received:  LED0\r\n
Data received:  LED1\r\n
Data received:  LED1\r\n
Data received:  LED2\r\n
Data received:  LED2\r\n
Data received:  Beep\r\n
Data received:  Beep\r\n
```

Summary

In order to connect your Pico board to your cell phone using a bluetooth connection, two files need to be added to the Pico board:

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

Once added, you can send and receive serial data to your cell phone. This allows you to monitor and control a device through your cell phone.

References

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