

Hasan, Albert, Jamisen, Wenson, Ben, Graham, Vanya

Bluetooth Communication

Critical Functions:

- Connect to Bluetooth between Raspberry Pi and computer
 - Windows done
 - Linux done
 - Mac: ?
- Compatibility with Mac, Windows, and Linux
- Measure RSSI (should be doable via bluetoothctl?)
- Measure transmit power (should be doable via bluetoothctl?)
- Send data over Bluetooth

Stretch Goals:

- Location and description of Bluetooth antenna
- Send a picture over Bluetooth

| | |
|---|----------|
| To manually connect to the Pi: (Windows/Linux) | 1 |
| Getting RSSI Readings: | 2 |
| Connecting RFCOMM: | 3 |

To manually connect to the Pi: (Windows/Linux)

In a Pi terminal:

```
bluetoothctl
```

```
> power on
```

```
> agent on
```

```
> discoverable on
```

```
> pairable on
```

Scan for bluetooth devices via the GUI for your operating system, and connect to the option with your pi's hostname. (usually raspberrypi) The GUI may say "connection failed" because the pi isn't a conventional bluetooth device-- ignore that.

In the bluetoothctl terminal, you should see two messages that look like:

```
[CHG] Device 54:8D:5A:43:9B:B6 Connected: yes  
[CHG] Device 54:8D:5A:43:9B:B6 Connected: no
```

If the bluetooth GUI gets stuck at this step, try removing the Pi from the list of bluetooth devices and re-scanning for it.

Type `connect` followed by the string in the above message-- in my case, the command would be `connect 54:8D:5A:43:9B:B6` but yours will likely be different.

Getting RSSI Readings:

(On the Pi, with bluez installed)

Hcitol scan #returns all bluetooth devices in range

Sudo bltmgmt find #returns info about all the devices in ranges, including RSSI values

```
hci0 dev_found: 72:7C:05:1B:E6:BA type LE Random rssi -86 flags 0x0000
AD flags 0x1a
eir_len 17
  RSSI: -68 dBm (0xbc)
  RSSI: -85 dBm (0xab)
  RSSI: -86 dBm (0xaa)
  RSSI: -86 dBm (0xaa)
  RSSI: -86 dBm (0xaa)
  RSSI: -86 dBm (0xaa)
  RSSI: -87 dBm (0xa9)
  RSSI: -86 dBm (0xaa)
  RSSI: -86 dBm (0xaa)
hci0 dev_found: 7D:BC:3F:D3:B5:EC type LE Random rssi -95 flags 0x0000
AD flags 0x06
eir_len 14
hci0 dev_found: 6C:94:F8:D2:4B:B1 type LE Public rssi -97 flags 0x0000
AD flags 0x1a
eir_len 15
  RSSI: -86 dBm (0xaa)
  RSSI: -86 dBm (0xaa)
  RSSI: -95 dBm (0xa1)
  RSSI: -95 dBm (0xa1)
  RSSI: -95 dBm (0xa1)
  RSSI: -95 dBm (0xa1)
  RSSI: -95 dBm (0xa1)
  RSSI: -95 dBm (0xa1)
  RSSI: -101 dBm (0x9b)
```

If your Pi is already connected via rfcomm, this should also work:

hcitol scan

#REPLACE 12:34:56:78:90 is your device's MAC address

sudo rfcomm connect 0 12:34:56:78:90:00 10 >/dev/null &

hcitol rssi 12:34:56:78:90:00

#or set it to check every 0.5 sec

watch -n 0.5 hcitol rssi 12:34:56:78:90:00

Another option for making the first method easier is to continuously read and isolate the RSSI values:

```
import os
```

```
import re
```

```
os.system("sudo hcitol scan")
```

```
deviceMAC = "9C:B6:D0:0F:10:90"
```

```
#adjust definedrange as needed
for i in range(definedrange):
    #input = os.system("sudo btmgmt find")
    #obj = re.findall(input, "(%s) .*? (rssi) (-\d\d)",
deviceMAC)
    #rssireading = re.findall(obj, "(-\d\d)")
    #print("Reading %s", obj)
```

Connecting RFCOMM:

On Pi:

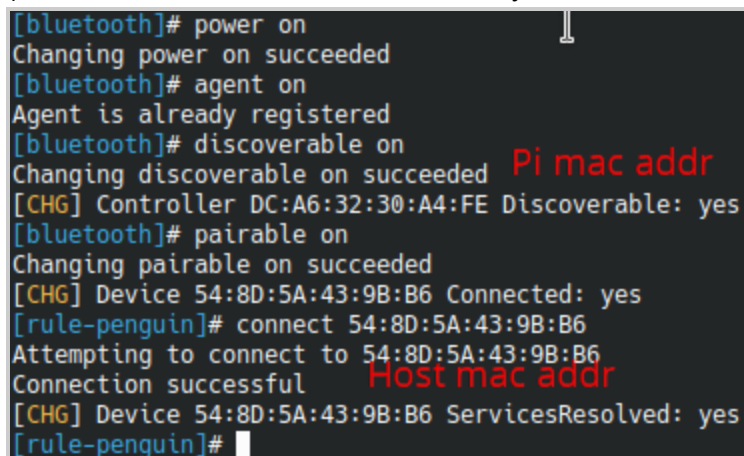
```
sudo rfcomm watch 0 &
```

(where XX...XX is the MAC address of your computer, as found from bluetoothctl)

On Linux host:

```
sudo rfcomm connect /dev/rfcomm0 XX:XX:XX:XX:XX:XX &
```

(where XX...XX is the MAC address of your Pi, as found from bluetoothctl)



```
[bluetooth]# power on
Changing power on succeeded
[bluetooth]# agent on
Agent is already registered
[bluetooth]# discoverable on
Changing discoverable on succeeded
[CHG] Controller DC:A6:32:30:A4:FE Discoverable: yes
[bluetooth]# pairable on
Changing pairable on succeeded
[CHG] Device 54:8D:5A:43:9B:B6 Connected: yes
[rule-penguin]# connect 54:8D:5A:43:9B:B6
Attempting to connect to 54:8D:5A:43:9B:B6
Connection successful
[CHG] Device 54:8D:5A:43:9B:B6 ServicesResolved: yes
[rule-penguin]#
```

If you are getting repeated connection refused errors, try running the commands without the ampersand to run them in the foreground-- it seems that background scripts have issues asking for sudo privileges on some versions of Linux. Running it once will allow you to authorize it manually.

To disconnect RFComm link: (Pi or Linux/Mac host)

```
sudo rfcomm release XX:XX:XX:XX:XX:XX
```

Alternatively:

```
sudo rfcomm release all
```

On Windows host:

Go to Control Panel > Devices and Printers > Unspecified > BLE Device xxxxx

Go to Services tab, note the serial port listed under SPP

Replace the serial port with that port in code, and run as is

Python scripts can be run from both ends to transfer data using the PySerial library. Additional Python helper functions are provided via the example code on GitHub:

```
getNext(s)
```

Reads the next available byte from the RfComm stream. Unlike serial.read, it will wait for data to become available -- this attribute is common to all reading functions detailed in this documentation.

Arguments:

[serial] s - The PySerial port to be read.

Returns:

[byte] The next byte of data available.

```
getInt(s, precision=4)
```

Reads the next several bytes of data as an unsigned integer value.

Arguments:

[serial] s - The PySerial port to be read.

[int] precision - The number of bytes to be read. Default 4 is equivalent to most languages' "int", while 8 is equivalent to most languages' "long".

Returns:

[int] The value read.

```
getFile(s, path, buf=2048)
```

Reads the incoming data as a file, with the first four bytes denoting its size. Individual files of greater than 4GB are not supported.

Arguments:

[serial] s - The PySerial port to be read.

[str] path - The location to save the file to. Any file at this path will be overwritten.

[int] buf - The size to allow to be written at a time- larger buffers give a very minor performance improvement. Values of 4096 or greater do not work reliably.

Returns: None.

```
sendInt(s, val, precision=4)
```

Sends the specified value to the serial port.

Arguments:

[serial] s - The PySerial port to be used.

[int] val - The value to be sent.

[int] precision - The number of bytes to be read. Default 4 is equivalent to most languages' "int", while 8 is equivalent to most languages' "long".

Returns: None.

```
sendFile(s, path, buf=2048)
```

Sends the contents of a file to the serial port, with the first four bytes denoting its size. Individual files of greater than 4GB are not supported.

Arguments:

[serial] s - The PySerial port to be read.

[str] path - The location to read the file from.

[int] buf - The size to allow to be read at a time- larger buffers give a very minor performance improvement. Values of 4096 or greater do not work reliably.

Returns: None.

Bluetooth specifications for raspberry pi 4

(<https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/>):

- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE

Location of bluetooth antennas on the raspberry pi

Location of bluetooth antenna on PC

Most computers come with built in wifi cards (top & bottom right), and these allow for wifi connection, but sometimes they also have bluetooth capabilities as well.

It is also possible to get a usb adapter (bottom left) that has a bluetooth antenna if your computer does not have it built in.

File Transfer: Obexpushd

Another method we were able to successfully implement on Windows and Mac for filetransfer via bluetooth involved obexpushd without using serial ports.

A general summary of its functionality:

-install obexpushd

-modify a bluez file to include -C (a compatibility flag) at the end of a particular line

-restart

-pair the pi with your PC and set it to listen

-Either use the command we listed to send a file from the Pi or use your computer's built in functionality to send a file to the Pi

Here's the python code we were running to implement it:

```
import os
```

```
os.system("sudo apt-get install obexpushd")
```

```

fin = open("/etc/systemd/system/dbus-org.bluez.service", "rt")
data = fin.read()
data = data.replace('ExecStart=/usr/lib/bluetooth/bluetoothd',
'ExecStart=/usr/lib/bluetooth/bluetoothd -C')
fin.close()
fin = open("/etc/systemd/system/dbus-org.bluez.service", "wt")
fin.write(data)
fin.close()

f = open("bluetoothshare", "x")
f.write("@reboot sudo mkdir /bluetooth \n @reboot sudo obexpushd -B
-o /bluetooth -n \n @reboot sudo sdptool browse local")

f.close()

os.system("/home/pi/Documents/bluetoothshare.txt")

print("rebooting")
os.system("sudo reboot now")

import os

os.system("bluetoothctl")
Type Discoverable on
Now use your computer's bluetooth MAC address to pair it with the pi
Once connected, you should see a notification in your terminal with a MAC address in
the format 00:00:00:00:00:00
devicemacaddress = raw_input("Right click to copy it and then paste
it here: ")
os.system("exit")

filename = raw_input("Name of file to be sent: ")

os.system("bluetooth-sendto --device=%s %s", (devicemacaddress,
filename))

```