Testing Sensor Unit/ Bluetooth

Requirements

You will need :

- a browser (Firefox or Chromium/Chrome recommended)
- an SSH client – Linux & Mac users will have this built into their system – Windows users should install the excellent puTTY utility – Android users should install the equally impressive JuiceSSH app (available from the Google Play Store as a free version). We assume something similar for Apple tablets but have no access to an Apple tablet to test

Wireless Attachment

Attach your tablet/PC/phone to the Skipper's Mate base unit wirelessly. This will have an SSID of **SkippersMate**, a user name of **skipper** and a password of **mate** (unless you have changed the Hotspot details yourself).

Browse to the Postgresql database

Browse to 192.168.51.254/phppgadmin. This will present you with a user name/password screen – use **pcl** and **pcl** to login.

You should then expand on the pcl database with the + sign, select public / Tables.

Scroll down to the sm_remoteunit table, and select the Browse option. This will show you your active MAC/BD_ADDR for the units – in the **mac** column (inactive ones or ones not requiring a BD_ADDR will be shown with a value of zero). Take a note of the active **mac**s.

Login with SSH client

You will now require your SSH client. Point this to 192.168.51.254 with a user name of **pi** and a password of **raspberry**.

Check for Bluetooth devices

Once presented with the command prompt type in :

```
hcitool scan
```

This can take a few moments to return :

```
Scanning ...
    98:D3:31:30:6C:B3  n/a
```

```
    20:16:12:08:04:47  n/a
    98:D3:37:00:8C:46  HC-06
    20:16:05:23:65:41  HC-05
```

That should give you a list of MAC/BD_ADDR addresses available to your system :

Check for active Bluetooth devices

We are interested in the one which does NOT appear in the database table. Having identified the required address (i.e. missing from table) you will need to check that the Bluetooth device is active.

```
sudo l2ping -c4 <mac address>
```

If we receive 4 results from this *(example) :*

```
pi@skipclient:~ $ sudo l2ping -c4 98:D3:37:00:8C:46
Ping: 98:D3:37:00:8C:46 from 00:15:83:0C:BF:EB (data size 44) ...
4 bytes from 98:D3:37:00:8C:46 id 0 time 14.97ms
4 bytes from 98:D3:37:00:8C:46 id 1 time 40.07ms
4 bytes from 98:D3:37:00:8C:46 id 2 time 17.64ms
4 bytes from 98:D3:37:00:8C:46 id 3 time 22.64ms
4 sent, 4 received, 0% loss
```

then the Bluetooth device is active.

Check Arduino is responding

We now need to test that the Arduino is responding so we need first to attach it to a /dev/rfcomm device :

```
sudo rfcomm bind 99 <mac address>
```

You can now

```
sudo picocom /dev/rfcomm99
```

This should bring up a list of characteristics

```
pi@skipclient:~ $ sudo picocom /dev/rfcomm99
picocom v1.7

port is        : /dev/rfcomm99
flowcontrol    : none
baudrate is    : 9600
parity is      : none
databits are   : 8
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
nolock is      : no
```

```
send_cmd is    : sz -vv
receive_cmd is : rz -vv
imap is        :
omap is        :
emap is        : crcrlf,delbs,

Terminal ready
```

You then hit any alpha character (e.g. a) and you will see either a string of numerics **or** no response whatsoever **or** FATAL: term closed.

No response or FATAL: term closed indicate a problem with the device being tested.

A string of numerics indicates that the device is functioning correctly and may be added to the database.

See Chapter 7 of the Installation Instructions for further steps.