Archived:Find bluetooth devices silently

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Introduction

Here is a package containing a source code and a sis file of an implementation of silent bluetooth devices search. It is a simple module which provides a single method, called discover.

Overview

This method can be called asynchronously (by passing a callable object as parameter) or synchronously (without parameter). It returns a list containing all found bluetooth devices MAC addresses.

Code Snippets

Here is an example of how to use the bluetooth search method synchronously.

```
import silent_bt

print silent_bt.discover()
```

Here is an example of how to use the bluetooth search method asynchronously.

```
def cb_func(devices):
    print devices

silent_bt.discover(cb_func)
```

With this method we can choose a device to be searched like in the example below:

```
import silent_bt

def watch_device(device):
    found_devices = []
    if type(device) == str:
        if device in silent_bt.discover():
            found_devices.append(device)
            return found_devices
    elif type(device) == list:
        for dev in device:
            if dev in silent_bt.discover():
                found_devices.append(dev)
                return found_devices
    return None

def find(device):
    device_found = watch_device(device)
    if device_found is not None:
        print "device found: %s"% device_found[0]
    else:
```

```
        find(device)

find("00:1f:5b:df:21:6c")
```

Output

```
device found: 00:1f:5b:df:21:6c
```

or verify if a device has entered or has exited the range area of the searching device:

```
import silent_bt

old_list = []
new_list = []
def observer():
    global old_list
    global new_list
    new_list = silent_bt.discover()
    for dev in old_list:
        if dev not in new_list:
            print "%s has exited" % dev
    for dev in new_list:
        if dev not in old_list:
            print "%s has entered" % dev
    old_list = new_list
    observer()

observer()
```

Output

```
00:1f:5b:df:21:6c has entered
2c:1d:aa:cb:22:2c has entered
00:1f:5b:df:21:6c has exited
```