PyBluez

Release master

Sep 16, 2019

Contents

Python extension module allowing access to system Bluetooth resources.

CHAPTER 1

Table of contents

1.1 Installing PyBluez

PyBluez can be installed on GNU/Linux, Windows and macOS systems and is compatible with Python 2.7 and 3.

Note: Before you install **PyBluez** please install the dependencies required for your system as described in the sections below.

Installing PyBluez using pip

Open a terminal (command prompt on Windows) and enter

```
pip install pybluez
```

(there are also binaries for Windows platform on PyPI or here - Unofficial Windows Binaries for Python Extension Packages)

For experimental Bluetooth Low Energy support (only for Linux platform - for additional dependencies please take look at: ble-dependencies)

```
pip install pybluez\[ble\]
```

Installing PyBluez from source

Download a stable release from https://github.com/pybluez/pybluez/releases

or download the latest version using the links below.

master.zip	master.tar.gz

Extract the zip or tar and cd to the extracted file directory, then:

```
python setup.py install
```

for Bluetooth Low Energy support (GNU/Linux only):

```
pip install -e .\[ble\]
```

1.1.1 GNU/Linux Dependencies

- Python 2.7 or more recent version
- Python distutils (standard in most Python distros, separate package python-dev in Debian)
- BlueZ libraries and header files

1.1.2 Windows Dependencies

- Windows 7/8/8.1/10
- Python 3.5 or more recent version

PyBluez requires a C++ compiler installed on your system to build CPython modules.

For Python 3.5 or higher

- Microsoft Visual C++ 14.0 standalone: Build Tools for Visual Studio 2017 (x86, x64, ARM, ARM64)
- Microsoft Visual C++ 14.0 with Visual Studio 2017 (x86, x64, ARM, ARM64)
- Microsoft Visual C++ 14.0 standalone: Visual C++ Build Tools 2015 (x86, x64, ARM)
- Microsoft Visual C++ 14.0 with Visual Studio 2015 (x86, x64, ARM)

Note: Windows 10 users need to download and install the Windows 10 SDK

More details here

- Widcomm BTW development kit 5.0 or later (Optional)

1.1.3 macOS Dependencies

- Xcode
- PyObjc 3.1b or later (https://pythonhosted.org/pyobjc/install.html#manual-installation)

1.2 Contributors

Original Author - Albert Huang.

Maintainers:

- Ryan Govostes
- Piotr Karulis

Other contributors:

- Travis Peters
- Michal Maruska
- Young-Bo
- Thomas Leveil
- Yurii Shevchuk
- Hugo Sales
- kedos
- Chad Spensky
- clach04
- Boris Belousov
- Ratmir Karabut
- zeripath
- Brennan Ashton
- Haim Daniel
- demanbart
- Clemens Wolff
- Arusekk
- Colin Atkinson
- Vitali Lovich
- bbregeault
- Eduardo Marossi
- Bill Crawford
- Christian Clauss

1.3 Contributing to PyBluez

This project is not currently under active development.

Contributions are strongly desired to resolve compatibility problems on newer systems, address bugs, and improve platform support for various features. Here are some guidelines to follow.

1.3.1 Compatibility Issues

Please submit compatiblity issues by opening an issue explaining the problem clearly and providing information necessary to reproduce the issue, including sample code and logs.

If you have long logs, please post them on https://gist.github.com and link to the Gist in your issue.

1.3.2 Bugs

Please submit bug reports by opening an issue explaining the problem clearly using code examples.

1.3.3 Coding

Todo: Develop PyBluez coding standards and style guides.

1.3.4 Documentation

The documentation source lives in the docs folder. Contributions to the documentation are welcome but should be easy to read and understand. All source documents are in restructured text format.

Todo: Develop PyBluez documentation standards and style guides.

1.3.5 Commit messages and pull requests

Commit messages should be concise but descriptive, and in the form of an instructional patch description (e.g., "Add macOS support" not "Added macOS support").

Commits which close, or intend to close, an issue should include the phrase "fix #234" or "close #234" where `#234` is the issue number, as well a short description, for example: "Add logic to support Win10, fix #234". Pull requests should aim to match or closely match the corresponding issue title.

1.3.6 Copyrights

By submitting to this project you agree to your code or documentation being released under the projects *license*. Copyrights on submissions are owned by their authors. Feel free to append your name to the list of contributors in `contributors.rst` found in the projects docs folder as part of your pull request!

1.4 PyBluez API

The Pybluez API provides a suite of classes and functions.

1.4.1 Classes

BluetoothSocket

Todo: Add documentation for the BluetoothSocket methods.

class `bluetooth.`**`BluetoothSocket`**(*proto=<sphinx.ext.autodoc.importer._MockObject object>*, *_sock=None*)

Bases: `object`

A Bluetooth Socket representing one endpoint of a Bluetooth connection.

Parameters **`proto`** (*`int`*) – The protocol the socket will use. The options are HCI, L2CAP, RFCOMM, or SCO. The default is RFCOMM.

Note: RFCOMM is the only protocol available for Windows and macOS systems.

`accept`()
Accept a connection.

Returns A tuple containing a `BluetoothSocket` and a Bluetooth address.

Return type tuple

Raises `BluetoothError` – When an attempt to accept a connection fails.

`bind`(*addrport*)
Bind the socket to a local address and port.

Parameters **`addrport`** (`tuple`) – A tuple of the form (address str, port int)

Raises `BluetoothError` – When an attempt to bind the socket fails.

`close`(**args*, ***kwargs*)

`connect`(**args*, ***kwargs*)

`connect_ex`(**args*, ***kwargs*)

`dup`()
Duplicate the socket

Returns A new `BluetoothSocket` connected to the same system resource.

Return type *BluetoothSocket*

`family`

`fileno`(**args*, ***kwargs*)

`get_l2cap_options`(*sock*, *mtu*)
Gets L2CAP options for the specified L2CAP socket. Options are: omtu, imtu, flush_to, mode, fcs, max_tx, txwin_size.

`getpeername`(**args*, ***kwargs*)

`getsockname`(**args*, ***kwargs*)

`getsockopt`(**args*, ***kwargs*)

`gettimeout`(**args*, ***kwargs*)

`listen`(**args*, ***kwargs*)

`makefile`(**args*, ***kwargs*)

`proto`

`recv`(**args*, ***kwargs*)

`recvfrom`(**args*, ***kwargs*)

`send`(**args*, ***kwargs*)

`sendall`(**args*, ***kwargs*)

`sendto`(**args*, ***kwargs*)

`set_l2cap_mtu`(*sock*, *mtu*)
Adjusts the MTU for the specified L2CAP socket. This method needs to be invoked on both sides of the connection for it to work! The default mtu that all L2CAP connections start with is 672 bytes.

mtu must be between 48 and 65535, inclusive.

set_l2cap_options(*sock*, *options*)
Sets L2CAP options for the specified L2CAP socket. The option list must be in the same format supplied by get_l2cap_options().

setblocking(**args*, ***kwargs*)

setl2capsecurity(**args*, ***kwargs*)

setsockopt(**args*, ***kwargs*)

settimeout(**args*, ***kwargs*)

shutdown(**args*, ***kwargs*)

timeout

type

DeviceDiscoverer

Todo: Add documentation for the DeviceDiscover class and its methods.

class bluetooth.**DeviceDiscoverer**(*device_id=-1*)
Bases: object

Skeleton class for finer control of the device discovery process.

To implement asynchronous device discovery (e.g. if you want to do something *as soon as* a device is discovered), subclass DeviceDiscoverer and override device_discovered () and inquiry_complete ()

cancel_inquiry()
Call this method to cancel an inquiry in process. inquiry_complete will still be called.

device_discovered(*address*, *device_class*, *rssi*, *name*)
Called when a bluetooth device is discovered.

address is the bluetooth address of the device

device_class is the Class of Device, as specified in [1] passed in as a 3-byte string

name is the user-friendly name of the device if lookup_names was set when the inquiry was started. otherwise None

This method exists to be overriden.

[1] https://www.bluetooth.org/foundry/assignnumb/document/baseband

fileno()

find_devices(*lookup_names=True*, *duration=8*, *flush_cache=True*)

find_devices (lookup_names=True, service_name=None, duration=8, flush_cache=True)

Call this method to initiate the device discovery process

lookup_names - set to True if you want to lookup the user-friendly names for each device found.

service_name - set to the name of a service you're looking for. only devices with a service of this name will be returned in device_discovered () NOT YET IMPLEMENTED

ADVANCED PARAMETERS: (don't change these unless you know what you're doing)

duration - the number of 1.2 second units to spend searching for bluetooth devices. If lookup_names is True, then the inquiry process can take a lot longer.

flush_cache - return devices discovered in previous inquiries

`inquiry_complete``()`
Called when an inquiry started by find_devices has completed.

`pre_inquiry``()`
Called just after find_devices is invoked, but just before the inquiry is started.

This method exists to be overriden

`process_event``()`
Waits for one event to happen, and proceses it. The event will be either a device discovery, or an inquiry completion.

`process_inquiry``()`
Repeatedly calls process_event () until the device inquiry has completed.

1.4.2 Functions

advertise_service

`bluetooth.`**`advertise_service`**`(`*sock*, *name*, *service_id=”*, *service_classes=[]*, *profiles=[]*, *provider=”*, *description=”*, *protocols=[]*`)`
Advertise a service with the local SDP server.

Parameters

- **`sock`** (`BluetoothSocket`) – The *`BluetoothSocket`* to use for advertising a service. The socket must be a bound, listening socket.
- **`name`** (*`str`*) – The name of the service and service_id (if specified). This should be a string of the form “XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX”, where each ‘X’ is a hexadecimal digit.
- **`service_classes`** (*`list`*) – a list of service classes belonging to the advertised service.

  Each service class is represented by a 16-bit or 128-bit UUID.

UUID Type	Format
Short 16-bit	XXXX
Full 128-bit	XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

  where each ‘X’ is a hexadecimal digit.

  There are some constants for standard services, e.g. SERIAL_PORT_CLASS that equals to “1101”. Some class constants provided by PyBluez are:

SERIAL_PORT_CLASS	LAN_ACCESS_CLASS
DIALUP_NET_CLASS	HEADSET_CLASS
CORDLESS_TELEPHONY_CLASS	AUDIO_SOURCE_CLASS
AUDIO_SINK_CLASS	PANU_CLASS
NAP_CLASS	GN_CLASS

- **`profiles`** (*`list`*) – A list of service profiles that thie service fulfills. Each profile is a tuple with (uuid, version). Most standard profiles use standard classes as UUIDs.

PyBluez offers a list of standard profiles, for example SERIAL_PORT_PROFILE. All standard profiles have the same name as the classes, except that _CLASS suffix is replaced by _PROFILE.

- **`provider`** (*`str`*) – A text string specifying the provider of the service
- **`description`** (*`str`*) – A text string describing the service
- **`protocols`** (*`list`*) – A list of protocols

Note: A note on working with Symbian smartphones: bt_discover in Python for Series 60 will only detect service records with service class SERIAL_PORT_CLASS and profile SERIAL_PORT_PROFILE

discover_devices

`bluetooth.`**`discover_devices`**(*duration=8*, *flush_cache=True*, *lookup_names=False*, *lookup_class=False*, *device_id=-1*, *iac=10390323*)

Perform a Bluetooth device discovery.

This function uses the first available Bluetooth resource to discover Bluetooth devices.

Parameters

- **`lookup_names`** (*`bool`*) – When set to True *`discover_devices()`* also attempts to look up the display name of each detected device. (the default is False).
- **`lookup_class`** (*`bool`*) – When set to True *`discover_devices()`* attempts to look up the class of each detected device. (the default is False).

Returns

Returns a list of device addresses as strings or a list of tuples. The content of the tuples depends on the values of lookup_names and lookup_class as detailed below.

lookup_class	lookup_names	Return
False	False	list of device addresses
False	True	list of (address, name) tuples
True	False	list of (address, class) tuples
True	True	list of (address, name, class) tuples

Return type list

find_service

`bluetooth.`**`find_service`**(*name=None*, *uuid=None*, *address=None*)

Use to find available Bluetooth services.

This function uses the service discovery protocol (SDP) to search for Bluetooth services matching the specified criteria and returns the search results.

The search criteria are defined by passing one or more parameters to the function.

If no criteria are specified then a list of all nearby services detected is returned. If more than one criteria is specified, then the search results will match all the criteria specified.

Parameters

- **`name`** (*`str or None`*) – The friendly name of a Bluetooth device.

- **uuid** (*str or None*) – A valid 16-bit or 128-bit UUID.

UUID Type	Format
Short 16-bit	XXXX
Full 128-bit	XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

where each 'X' is a hexadecimal digit.

- **address** (*str or None*) – The Bluetooth address of a device or "localhost". If "localhost" is provided the function will search for Bluetooth services on the local machine.

Returns

The search results will be a list of dictionaries. Each dictionary represents a search match having the following key/value pairs.

Key	Value
host	the bluetooth address of the device advertising the service.
name	the name of the service being advertised.
description	a description of the service being advertised.
provider	the name of the person/organization providing the service.
protocol	either 'RFCOMM', 'L2CAP', None if the protocol was not specified, or 'UNKNOWN' if the protocol was specified but unrecognized.
port	the L2CAP PSM number if the protocol is 'L2CAP', the RFCOMM channel number if the protocol is 'RFCOMM', or None if the protocol wasn't specified.
service-classes	a list of service class IDs (UUID strings). Possibly empty
profiles	a list of profiles the service claims to support. a profile takes the form of (UUID, version) pairs. Possibly empty.
service-id	the Service ID of the service. None if it wasn't set See the Bluetooth spec for the difference between Service ID and Service Class ID List

Return type list

lookup_name

`bluetooth.lookup_name(address, timeout=10)`

Look up the friendly name of a Bluetooth device.

This function tries to determine the friendly name (human readable) of the device with the specified Bluetooth address.

Parameters **address** (*str*) – The Bluetooth address of the device.

Returns The friendly name of the device on success, and None on failure.

Return type str or None

Raises *BluetoothError* – When the provided address is not a valid Bluetooth address.

stop_advertising

`bluetooth.`**`stop_advertising`**`(`*sock*`)`

Try to stop advertising a bluetooth service.

This function instructs the local SDP server to stop advertising the service associated with socket. You should typically call this right before you close socket.

Parameters **`sock`** (`BluetoothSocket`) – The *`BluetoothSocket`* to stop advertising the service on.

Raises *`BluetoothError`* – When SDP fails to stop advertising for some reason.

1.4.3 Exceptions

BluetoothError

`exception` `bluetooth.`**`BluetoothError`**

Bases: `OSError`

Raised when a bluetooth function or method fails for a Bluetooth I/O related reason.

1.5 License

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

```
  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.


  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.
```

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

Indices and tables:

- genindex
- modindex
- search

Index