 This repository Search

Pull requestsIssuesGist

adafruit / Adafruit_Python_BluefruitLE


Watch35Star112Fork41

<> CodeIssues9Pull requests6Projects0WikiPulseGraphs


Python library to simplify access to Bluetooth low energy devices and services on Linux (using bluez) and Mac OSX.

47 commits1 branch0 releases3 contributorsMIT

Branch: masterNew pull requestCreate new fileUpload filesFind fileClone or download

tdicola Add GitHub pull request templateLatest commit 1509151 on 28 May

.github	Add GitHub pull request template	7 months ago
Adafruit_BluefruitLE	Add delay for GLib loop to start on Linux.	a year ago
examples	Better Ctrl-C handling on OSX	a year ago
.gitignore	Major refactor to combine bluez & corebluetooth, part 1 or 2.	2 years ago
LICENSE	Initial commit	2 years ago
README.md	fix Adafruit tutorial URL in the README	a year ago
ez_setup.py	Major refactor to combine bluez & corebluetooth, part 1 or 2.	2 years ago
setup.py	Rename to BluefruitLE library.	a year ago

 README.md

Adafruit Python BluefruitLE

Python library to simplify access to Bluefruit LE (Bluetooth low energy) devices and services on Linux and Mac OSX. Hides all the platform-specific BLE code (using BlueZ on Linux and CoreBluetooth on Mac OSX) and provides a simpler synchronous BLE API which is great for scripts and other automated tasks using BLE. For example you can create a program to read and write data (like sensor readings) with a Bluefruit LE device in UART mode.

See more details from the guide here: <https://learn.adafruit.com/bluefruit-le-python-library/overview>

Note this library provides a Python wrapper for BLE **Central Mode**, meaning it can initiate connections and data exchanges with other BLE Peripherals. It does not allow you to emulate a BLE Peripheral via Python, or provide Python-based access to BLE peripherals connected to your system.

Be aware that this library is early in development! The examples show how to detect and interact with BLE UART devices which are the primary target of the library. More advanced features like getting advertisement data or interacting with other BLE services might not work or be currently supported. In particular BLE support for desktop operating systems is still somewhat buggy and spotty with support. The library has been developed and primarily tested on Python 2.7.x, but if there are issues with Python 3 please raise them so support can be added.

Supported Platforms

The library supports the following platforms:

- Linux, using the latest BlueZ 5.33 release. The library is implemented using BlueZ's experimental DBus bindings for access from languages such as Python.
- Mac OSX, using Apple's CoreBluetooth library. The library depends on PyObjC which Apple includes with their Python version on OSX. Note that if you're using a different Python, like the one provided by Homebrew, be sure to [install PyObjC for that version of Python](#).

What about Windows support?

The library currently does not support Windows since BLE support is limited on that platform. Windows 7 does not support

BLE at all. Windows 8 does not have [BLE device search and pairing APIs](#). Windows 10 appears to have a more capable BLE API but it has not yet been investigated to see if it can be supported by this library yet.

Linux & Raspberry Pi Requirements

On Linux (like with a Raspberry Pi) you'll need to compile and install the latest version of BlueZ, currently version 5.33, to gain access to the Bluetooth LE API it exposes. It's important to use this more recent version of BlueZ as the BLE APIs are still in development and a bit experimental.

Warning: Be careful compiling and installing a later version of BlueZ on top of an existing version if you use a Linux desktop OS like Ubuntu, Debian, etc. You might cause an issue with the new BlueZ version conflicting with the distribution's older BlueZ installation and wreaking havoc with package dependencies. Ubuntu in particular can get into a very bad state since core components depend on BlueZ. Look for a properly packaged BlueZ 5.33 release for your distribution, or install into a virtual machine that can protect your desktop machine from conflicts.

The steps below describe how to install BlueZ 5.33 on a Raspberry Pi running its Raspbian operating system. In a terminal on the Raspberry Pi run:

```
sudo apt-get update
sudo apt-get -y install libusb-dev libdbus-1-dev libglib2.0-dev libudev-dev libical-dev libreadline-dev
wget http://www.kernel.org/pub/linux/bluetooth/bluez-5.33.tar.gz
tar xvfz bluez-5.33.tar.gz
cd bluez-5.33
./configure --disable-systemd
make
sudo make install
sudo cp ./src/bluetoothd /usr/local/bin/
```

Finally you'll need to make sure the bluetoothd daemon runs at boot and is run with the `--experimental` flag to enable all the BLE APIs. To do this edit the `/etc/rc.local` file and add the following line before the `exit 0` at the end:

```
/usr/local/bin/bluetoothd --experimental &
```

Save the changed file and reboot the Pi. Then verify using the command `ps aux | grep bluetoothd` that the bluetoothd daemon is running.

Mac OSX Requirements

On Mac OSX you do not need to install any dependencies to start using the library (the PyObjC library should be installed already by Apple).

Installation

Once the requirements above have been met the library can be installed by running the following command inside its root directory:

```
sudo python setup.py install
```

This will install the library so it can be used by any script on your system.

Alternatively you can run `sudo python setup.py develop` to have the library installed in develop mode where changes to the code (like doing a `git pull`) will immediately take effect without a reinstall.

After the library is installed examine the examples folder to see some examples of usage:

- **list_uarts.py** - This example will print out any BLE UART devices that can be found and is a simple example of searching for devices.
- **uart_service.py** - This example will connect to the first BLE UART device it finds, send the string 'Hello World!' and then wait 60 seconds to receive a reply back. The example uses a simple synchronous BLE UART service implementation to send and receive data with the UART device.
- **device_info.py** - This example will connect to the first BLE UART device it finds and print out details from its device info service. **Note this example only works on Mac OSX!** Unfortunately a bug / design issue in the current BlueZ API

prevents access to the device information service.

- **low_level.py** - This is a lower-level example that interacts with the services and characteristics of a BLE device directly. Just like the `uart_service.py` example this will connect to the first found UART device, send a string, and then print out messages that are received for one minute.

To run an example be sure to run as the root user on Linux using `sudo`, for example to run the `uart_service.py` example:

```
sudo python uart_service.py
```

On Mac OSX the `sudo` prefix to run as root is not necessary.

