1/3/2016

Asynchronous Bluetooth Server Tutorial

Pranshu Bansal, Shashank Jere, Azim Muqtadir, Likun Wang
EE202C – DR WILLIAM KAISER

1 CONTENTS

2 PREAMBLE

The development was all performed in python to allow for ease and quick pace of development. The Bluetooth API is similar in C and therefore transitioning it should not be too difficult.

2.1 INTRODUCTION

In this tutorial, we will explore Bluetooth and how to do message passing between a central Bluetooth enabled server and multiple clients. To do this we must examine:

1. Threaded asynchronous server-client architecture
2. Bluetooth protocols and API's

2.2 TERMINOLOGY

Anchor node/Bluetooth Server – any Intel Edison running the server_bluetooth code. I recommend using the large Edison boards for this as they will be fixed in position due to size and power availability

Client Node/Wearable Device/Bluetooth Client/Roaming Agent – any Intel Edison running the client_bluetooth code. I recommend using the small spark red breakout boards as they have a portable battery source and can be moved around a room.

Bdaddr – Bluetooth **D**evice **Addr**ess

2.3 THINGS NEEDED

- At least one client node (moveable battery powered Intel Edison)
- At least one anchor node (fixed point Intel Edison)
- Micro USB-cables
- PC or Mac

2.4 KNOWN ISSUES

If the file "Bluetooth_client.py" says something like

- Error code 13: permission denied
- Error Code 111: connection refused
- Error code 110 (timed out) is ok: just wait for it to try again

Don't worry, you didn't do anything wrong. This is an ongoing issue that has yet to be debugged. The fixes you can try include:

- Attempting to pair the devices manually using bluetoothctl tool
- Install bluetooth service (https://software.intel.com/en-us/blogs/2015/05/19/communicate-to-arduino-code-with-your-android-phone-by-bluetooth-serial-port)
- Reflashing the board

2.5 FILES REQUIRED:

- bluetooth_globs.py – contains common variables for all files
- Bluetooth_rssi.py – has API to get RSSI information for proximity detection purposes

- Client_bluetooth.py – file to run on wearable roaming Edison devices
- Client_webserver.py – has API to push messages to a SQL server
- Get_bdadder.py – has function to return calling devices Bluetooth address
- Server_bluetooth.py – code to run on fixed place server Edison

2.6 DEPENDANCIES

Library	Usage
import bluetooth	bluetooth library
import bluetooth._bluetooth as bluez	low level API for bluetooth
import datetime	timekeeping purposes
import json	parsing for SQL database
import os	forking etc.
import socket	establishing socket connections
import struct	for RSSI detection
import sys	exit, other system calls
import time	sleep/delay purposes
import urllib	opening websites
from bluetooth_globs import	file containing globals for each other file
from bluetooth_rssi import get_closest_device	returns bluetooth address of closest device based on highest RSSI
from bluetooth_rssi import get_devices	returns list of all nearby bluetooth devices that are recognized by our system
from bluetooth_rssi import get_my_bdaddr	returns the bluetooth address of "myself"

3 ASYNCHRONOUS SERVER INTRODUCTION

In a nutshell, these servers are a central point that allow for multiple clients to connect to them and simultaneously service their requests. If you know how they work feel free to skip the explanation and just dive straight into the source code provided.

If you are unfamiliar with asynchronous servers read the following links:

- http://ruslanspivak.com/lsbaws-part1/
- http://ruslanspivak.com/lsbaws-part2/
- http://ruslanspivak.com/lsbaws-part3/

4 COMMUNICATION OVER BLUETOOTH

Python has a very nice API in the PyBluez library (which can be installed via: pip install pybluez). This allows for Bluetooth communication over sockets. Since the asynchronous webserver seen above uses sockets, all we have to do is change the protocol from a web socket to a Bluetooth socket and it should function in a very similar manner.

The Bluetooth library is explained very nicely in the following link:

- https://people.csail.mit.edu/albert/bluez-intro/index.html

5 RUNNING THE CODE

1. Push all files included with the tutorial onto all of your Edison boards
    a. I recommend using MobaXterm and pushing via SSH/SFTP
    b. Simply drag dropping the files over USB to the Edison will not let the files be viewable within the Edison
2. Install libraries as required
    a. E.g. pip install pybluez
3. Find the Bluetooth address of each anchor node by running "python get_bdaddr.py"
4. Append the Bluetooth address of each anchor node to the list "known_devices" in "bluetooth_globs.py" – NB: this is very laborious, but the code will not work without a list of recognized devices
5. For the anchor nodes: run command: "python server_bluetooth.py"
6. For the client nodes: run command: "python client_bluetooth.py"

6 WHAT YOU SHOULD SEE

The functionality of the code on the server/client will be similar to the video posted

https://www.youtube.com/watch?v=fVwv8R93XY8&feature=youtu.be&list=PLHdckC77jfRvqTpLHdA7C9zo-rngn5cch

Full development and presentation demonstration videos can be seen at:

https://www.youtube.com/playlist?list=PLHdckC77jfRvqTpLHdA7C9zo-rngn5cch