ECE 597 Report

Bluetooth Web design

By Liaoyuan Li

Instructor: Dr Jia Wang

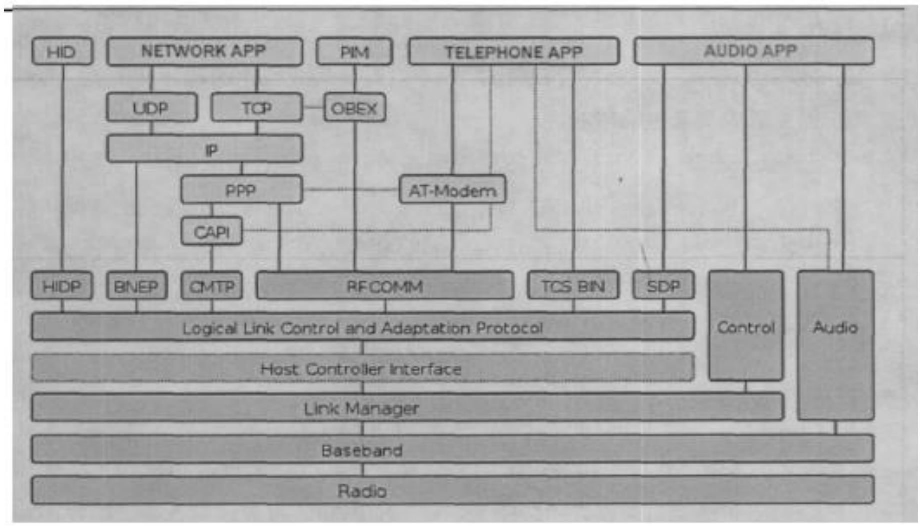
Date: 04/28/2019

Introduction

This report is established to demonstrate how to develop a basic Bluetooth web application by each step and how it works. In this report, the main part of this report is not create an API that enables Bluetooth device to connect to the web through suitable protocol by oneself. On top of that, the basic knowledge of JavaScript and Bluetooth services which are assumed as prerequisites.

Background

Web Bluetooth is a new technology as a feature of low energy and appropriate distance transition, which satisfies with the demands of Internet of Things. Web Bluetooth will take effect on several platforms such as both mobile and desktop, which means lower development costs, more open source control interfaces for various physical products, and more innovation.



The figure above represent all kind of protocol of Bluetooth. The Bluetooth specification adopts a hierarchical structure in its protocol stack to separately filter and transmit data streams, frequency hopping and numbers. According to frame transmission, connection establishment and release, link control, data disassembly and assembly, quality of service, protocol reuse and distribution. Bluetooth 4.0 introduced a new "Low Energy" mode known as "Bluetooth Smart", BLE, or just LE which is limited to about 1Mbps but allows devices to leave their transmitters off most of the time.

Result and Analysis

a) Set a protocol (choose API for Web Services)

Since Chrome browse (the product of Google Company) has released several channel for Communication API whose name is GATT (Generic Attribute Profile). I choose Chrome as develop platform for this web.

b) Add class to web (connection)

Similar to the tutorial, I use below namespace to start connecting a Bluetooth.

```
navigator.bluetooth.requestDevice
```

This is a JavaScript namespace, which avoid some information lose and ensure API to move toward one direction, which means I have to avoid callback and async interactions can be passed around like any other variable.

```
 filters: [{

    services: ['0xffe5'],

  }]

}).then(device => function())
```

```
.then(server => server.

                     getPrimaryService(
```

```
    'heart_rate'))

.then(service => {
      chosenHeartRateService = service;

      return Promise.all([

        service.

              getCharacteristic(

        'body_sensor_location')

          .then(handleBodySensorLocationCharacteristic),

        service.

              getCharacteristic(

        'heart_rate_measurement')

          .then(handleHeartRateMeasurementCharacteristic),

      ]);

    });
```

The Original code is attached in the end of the report.

c) Run

Since I have no android device which the Chrome windows version supports, I didn't get the real-time result from a Bluetooth device's UI. So I use a simulation on my Windows machine instead.

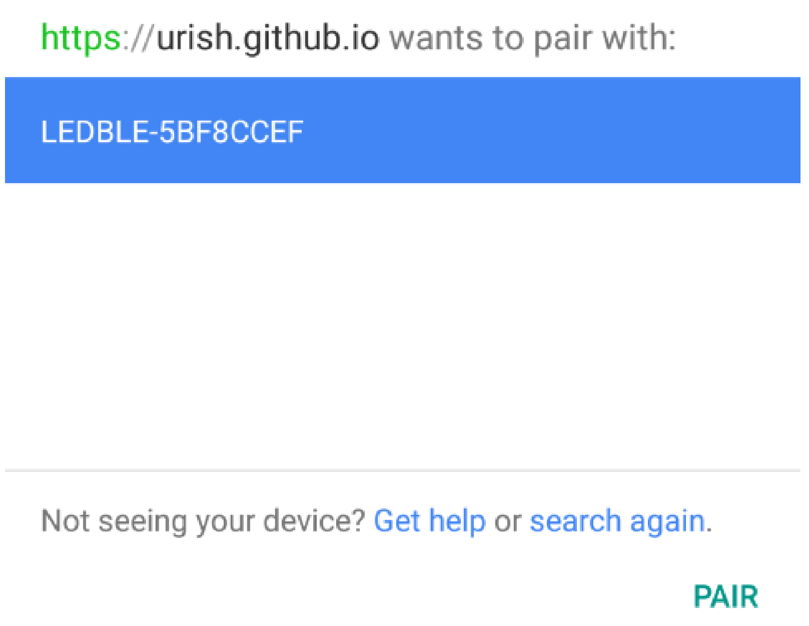
1. Open the Developer Options screen on your Android. See Configure On-Device Developer Options.

2. Select Enable USB Debugging.

3. On your development machine, open Chrome.

4. Open DevTools.

5. In DevTools, click the Main Menu Main Menu then select More tools > Remote devices.


https://urish.github.io wants to pair with:

LEDBLE-5BF8CCEF

Not seeing your device? Get help or search again.

Android Emulator on windows

In this step, I have to take the security of the web into consideration, that is, since the Bluetooth device has connected to my web. Will the other device which may connect to my web accidently modify mine environment such as write, read or execute? The answer is no. By chrome team: When deciding whether to ship the new API, we should look at several kinds of attackers and defenders:

- *An abusive software developer, trying to do embarrassing or privacy-insensitive things that don't go outside devices' security models.*
- *A malicious software developer, trying to exploit users using nearby Bluetooth devices.*
- *A malicious hardware manufacturer, trying to exploit users or websites who connect to their devices.*
- *A malicious manufacturer/developer, who can push cooperating hardware and software.*
- *Weakly-written device firmware, which doesn't intend to hurt its users, but might be vulnerable to malicious connections.*
- *Weakly-written kernels, which might be vulnerable to either malicious userland software or malicious connections.*

The ultimate decision about whether to ship Web Bluetooth should also take the competitiveness of the web into account, but this article only analyzes the security tradeoffs.

Discussion

So far I have built the web through chrome platform using GATT API and establish a Bluetooth device connection. The security between host and device is good due to the chrome hierarchy setting. Web Bluetooth provides more warning to users than Android or iOS before giving access to the first device. Web Bluetooth also requires the same permission sequence for each additional device, so the malicious developer can't attack devices the user wasn't aware of.

Reference

[BLUETOOTH-ASSIGNED]

Assigned Numbers. Living Standard. URL: https://www.bluetooth.org/en-us/specification/assigned-numbers

[BLUETOOTH-ASSIGNED-CHARACTERISTICS]

Bluetooth GATT Specifications > Characteristics. Living Standard. URL: https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicsHome.aspx

[BLUETOOTH-ASSIGNED-DESCRIPTORS]

Bluetooth GATT Specifications > Descriptors. Living Standard. URL: https://developer.bluetooth.org/gatt/descriptors/Pages/DescriptorsHomePage.aspx

[BLUETOOTH-ASSIGNED-SERVICES]

Bluetooth GATT Specifications > Services. Living Standard. URL: https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx

[BLUETOOTH-SUPPLEMENT6]

Supplement to the Bluetooth Core Specification Version 6. 14 July 2015. URL: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=302735

[BLUETOOTH42]

BLUETOOTH SPECIFICATION Version 4.2. 2 December 2014. URL: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439

[DOM]

Anne van Kesteren. DOM Standard. Living Standard. URL: https://dom.spec.whatwg.org/

[ECMAScript]

ECMAScript Language Specification. URL: https://tc39.github.io/ecma262/

[ENCODING]

Anne van Kesteren. Encoding Standard. Living Standard. URL: https://encoding.spec.whatwg.org/

[HTML]

Anne van Kesteren; et al. HTML Standard. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[PERMISSIONS]

Mounir Lamouri; Marcos Caceres; Jeffrey Yasskin. Permissions. 25 September 2017. WD. URL: https://www.w3.org/TR/permissions/

[PERMISSIONS-REQUEST-1]

Requesting Permissions Spec URL: https://wicg.github.io/permissions-request/

[PROMISES-GUIDE]

Domenic Denicola. Writing Promise-Using Specifications. 16 February 2016. TAG Finding. URL: https://www.w3.org/2001/tag/doc/promises-guide

[RFC2119]

S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

[RFC4122]

P. Leach; M. Mealling; R. Salz. A Universally Unique IDentifier (UUID) URN

Namespace. July 2005. Proposed Standard. URL: https://tools.ietf.org/html/rfc4122

[SECURE-CONTEXTS]

Mike West. Secure Contexts. 15 September 2016. CR. URL: https://www.w3.org/TR/secure-contexts/

[WebIDL]

Boris Zbarsky. Web IDL. 15 December 2016. ED. URL: https://heycam.github.io/webidl/