**npm install bluetooth-hci-socket**

**npm install bleno**

**Precondition**

**Include files**

var util = require('util');  
var bleno = require('./index');

**Hardware state**

bleno.on('stateChange', function(state) {

| console.log('on -> stateChange: ' + state + ', address = ' + bleno.address); |
| --- |
| **Advertising Start** |

bleno.startAdvertising(***'hello'***, ['fffffffffffffffffffffffffffffff0']);

**Advertising Stop**

bleno.stopAdvertising();

**Events Handled by BLENO**

this.\_bindings.on('**stateChange**', this.onStateChange.bind(this));  
this.\_bindings.on(**'platform'**, this.onPlatform.bind(this));  
this.\_bindings.on(**'addressChange'**, this.onAddressChange.bind(this));  
this.\_bindings.on(**'advertisingStart'**, this.onAdvertisingStart.bind(this));  
this.\_bindings.on(**'advertisingStop'**, this.onAdvertisingStop.bind(this));  
this.\_bindings.on(**'servicesSet'**, this.onServicesSet.bind(this));  
this.\_bindings.on(**'accept'**, this.onAccept.bind(this));  
this.\_bindings.on(**'mtuChange'**, this.onMtuChange.bind(this));  
this.\_bindings.on(**'disconnect'**, this.onDisconnect.bind(this));  
this.\_bindings.on(**'rssiUpdate'**, this.onRssiUpdate.bind(this));

**Defining Services**

bleno.setServices([

new **SampleService**()

]);

**Service List**

function **SampleService**() {

SampleService.super\_.call(this, {

uuid: 'fffffffffffffffffffffffffffffff0',

**//Defining Characteristics**

**characteristics**: [

new **StaticReadOnlyCharacteristic**(),

new **DynamicReadOnlyCharacteristic**(),

new **LongDynamicReadOnlyCharacteristic**(),

new **WriteOnlyCharacteristic**(),

new **NotifyOnlyCharacteristic**(),

new **IndicateOnlyCharacteristic**()

]

});

}

**StaticReadOnlyCharacteristic**

var StaticReadOnlyCharacteristic = function() {

StaticReadOnlyCharacteristic.super\_.call(this, {

**uuid**: **'fffffffffffffffffffffffffffffff1'**,

**properties: ['read'],**

value: new Buffer('value'),

descriptors: [

new BlenoDescriptor({

uuid: '2901',

value: 'user description'

})

]

});

};

util.inherits(StaticReadOnlyCharacteristic, BlenoCharacteristic);

**DynamicReadOnlyCharacteristic**

var DynamicReadOnlyCharacteristic = function() {

DynamicReadOnlyCharacteristic.super\_.call(this, {

**uuid**: **'fffffffffffffffffffffffffffffff2'**,

**properties: ['read']**

});

};

util.inherits(DynamicReadOnlyCharacteristic, BlenoCharacteristic);

DynamicReadOnlyCharacteristic.prototype.**onReadRequest** = function(offset, callback) {

var result = this.RESULT\_SUCCESS;

var data = new Buffer('dynamic value');

if (offset > data.length) {

result = this.RESULT\_INVALID\_OFFSET;

data = null;

} else {

data = data.slice(offset);

}

callback(result, data);

};

**LongDynamicReadOnlyCharacteristic**

var LongDynamicReadOnlyCharacteristic = function() {

LongDynamicReadOnlyCharacteristic.super\_.call(this, {

**uuid**: **'fffffffffffffffffffffffffffffff3'**,

**properties: ['read']**

});

};

util.inherits(LongDynamicReadOnlyCharacteristic, BlenoCharacteristic);

LongDynamicReadOnlyCharacteristic.prototype.**onReadRequest** = function(offset, callback) {

var result = this.RESULT\_SUCCESS;

var data = new Buffer(512);

for (var i = 0; i < data.length; i++) {

data[i] = i % 256;

}

if (offset > data.length) {

result = this.RESULT\_INVALID\_OFFSET;

data = null;

} else {

data = data.slice(offset);

}

callback(result, data);

};

**WriteOnlyCharacteristic**

var WriteOnlyCharacteristic = function() {

WriteOnlyCharacteristic.super\_.call(this, {

**uuid**: **'fffffffffffffffffffffffffffffff4'**,

**properties: ['write', 'writeWithoutResponse']**

});

};

util.inherits(WriteOnlyCharacteristic, BlenoCharacteristic);

WriteOnlyCharacteristic.prototype.**onWriteRequest** = function(data, offset, withoutResponse, callback) {

console.log('WriteOnlyCharacteristic write request: ' + data.toString('hex') + ' ' + offset + ' ' + withoutResponse);

callback(this.RESULT\_SUCCESS);

};

**NotifyOnlyCharacteristic**

var NotifyOnlyCharacteristic = function() {

NotifyOnlyCharacteristic.super\_.call(this, {

**uuid**: **'fffffffffffffffffffffffffffffff5'**,

**properties: ['notify']**

});

};

util.inherits(NotifyOnlyCharacteristic, BlenoCharacteristic);

NotifyOnlyCharacteristic.prototype.**onSubscribe** = function(maxValueSize, updateValueCallback) {

console.log('NotifyOnlyCharacteristic subscribe');

this.counter = 0;

this.changeInterval = setInterval(function() {

var data = new Buffer(4);

data.writeUInt32LE(this.counter, 0);

console.log('NotifyOnlyCharacteristic update value: ' + this.counter);

updateValueCallback(data);

this.counter++;

}.bind(this), 5000);

};

NotifyOnlyCharacteristic.prototype.**onUnsubscribe** = function() {

console.log('NotifyOnlyCharacteristic unsubscribe');

if (this.changeInterval) {

clearInterval(this.changeInterval);

this.changeInterval = null;

}

};

NotifyOnlyCharacteristic.prototype.**onNotify** = function() {

console.log('NotifyOnlyCharacteristic on notify');

};

**IndicateOnlyCharacteristic**

var IndicateOnlyCharacteristic = function() {

IndicateOnlyCharacteristic.super\_.call(this, {

**uuid: 'fffffffffffffffffffffffffffffff6',**

**properties: ['indicate']**

});

};

util.inherits(IndicateOnlyCharacteristic, BlenoCharacteristic);

IndicateOnlyCharacteristic.prototype.**onSubscribe** = function(maxValueSize, updateValueCallback) {

console.log('IndicateOnlyCharacteristic subscribe');

this.counter = 0;

this.changeInterval = setInterval(function() {

var data = new Buffer(4);

data.writeUInt32LE(this.counter, 0);

console.log('IndicateOnlyCharacteristic update value: ' + this.counter);

updateValueCallback(data);

this.counter++;

}.bind(this), 1000);

};

IndicateOnlyCharacteristic.prototype.**onUnsubscribe** = function() {

console.log('IndicateOnlyCharacteristic unsubscribe');

if (this.changeInterval) {

clearInterval(this.changeInterval);

this.changeInterval = null;

}

};

IndicateOnlyCharacteristic.prototype.**onIndicate** = function() {

console.log('IndicateOnlyCharacteristic on indicate');

};