```
var Accessory = require('../').Accessory;
var Service = require('../').Service;
var Characteristic = require('../').Characteristic;
var uuid = require('../').uuid;
var mqtt = require('mqtt'); // import mqtt
var options = {
  port: 1883,
  host: 'YourRaspberryPiHostname.local',
  clientId: 'YourAccessoryPi_MQTT_Publisher'
}; // connection options
var client = mqtt.connect(options); // make mqtt client object

var LightController = {
  name: "Your Accessory Name", //name of accessory
  pincode: "031-45-154",
  username: "FA:3C:ED:5A:1A:1A", // MAC like address used by HomeKit
to differentiate accessories.
  manufacturer: "Your Manufacturer", //manufacturer (optional)
  model: "Your Model", //model (optional)
  serialNumber: "Your Serial Number", //serial number (optional)

  power: false, //current power status
  brightness: 100, //current brightness
  hue: 0, //current hue
  saturation: 0, //current saturation

  outputLogs: false, //output logs

  setPower: function(status) { //set power of accessory
    if(this.outputLogs) console.log("Turning the '%s' %s", this.name,
status ? "on" : "off");
    this.power = status;
    if (status) {
      client.publish('Your MQTT Feed', '100'); // on is 100%
    }
    else {
      client.publish('Your MQTT Feed', '0'); // off is 0%
    }
  },

  getPower: function() { //get power of accessory
    if(this.outputLogs) console.log("'%s' is %s.", this.name,
this.power ? "on" : "off");
```

```
    return this.power;
  },

  setBrightness: function(brightness) { //set brightness
    if(this.outputLogs) console.log("Setting '%s' brightness to %s",
this.name, brightness);
    this.brightness = brightness;
    client.publish('Your MQTT Feed', this.brightness.toString()); //
other brightnesses
  },

  getBrightness: function() { //get brightness
    if(this.outputLogs) console.log("'%s' brightness is %s",
this.name, this.brightness);
    return this.brightness;
  },

  setSaturation: function(saturation) { //set brightness
    if(this.outputLogs) console.log("Setting '%s' saturation to %s",
this.name, saturation);
    this.saturation = saturation;
  },

  getSaturation: function() { //get brightness
    if(this.outputLogs) console.log("'%s' saturation is %s",
this.name, this.saturation);
    return this.saturation;
  },

  setHue: function(hue) { //set brightness
    if(this.outputLogs) console.log("Setting '%s' hue to %s",
this.name, hue);
    this.hue = hue;
  },

  getHue: function() { //get hue
    if(this.outputLogs) console.log("'%s' hue is %s", this.name,
this.hue);
    return this.hue;
  },

  identify: function() { //identify the accessory
    if(this.outputLogs) console.log("Identify the '%s'", this.name);
```

```
  }
}

// Generate a consistent UUID for our light Accessory that will
remain the same even when
// restarting our server. We use the `uuid.generate` helper function
to create a deterministic
// UUID based on an arbitrary "namespace" and the word "light".
var lightUUID = uuid.generate('hap-nodejs:accessories:light' +
LightController.name);

// This is the Accessory that we'll return to HAP-NodeJS that
represents our light.
var lightAccessory = exports.accessory = new
Accessory(LightController.name, lightUUID);

// Add properties for publishing (in case we're using Core.js and not
BridgedCore.js)
lightAccessory.username = LightController.username;
lightAccessory.pincode = LightController.pincode;

// set some basic properties (these values are arbitrary and setting
them is optional)
lightAccessory
  .getService(Service.AccessoryInformation)
    .setCharacteristic(Characteristic.Manufacturer,
LightController.manufacturer)
    .setCharacteristic(Characteristic.Model, LightController.model)
    .setCharacteristic(Characteristic.SerialNumber,
LightController.serialNumber);

// listen for the "identify" event for this Accessory
lightAccessory.on('identify', function(paired, callback) {
  LightController.identify();
  callback();
});

// Add the actual Lightbulb Service and listen for change events from
iOS.
// We can see the complete list of Services and Characteristics in
`lib/gen/HomeKitTypes.js`
lightAccessory
```

```
  .addService(Service.Lightbulb, LightController.name) // services
exposed to the user should have "names" like "Light" for this case
  .getCharacteristic(Characteristic.On)
  .on('set', function(value, callback) {
    LightController.setPower(value);

    // Our light is synchronous - this value has been successfully
set
    // Invoke the callback when you finished processing the request
    // If it's going to take more than 1s to finish the request, try
to invoke the callback
    // after getting the request instead of after finishing it. This
avoids blocking other
    // requests from HomeKit.
    callback();
  })
  // We want to intercept requests for our current power state so we
can query the hardware itself instead of
  // allowing HAP-NodeJS to return the cached Characteristic.value.
  .on('get', function(callback) {
    callback(null, LightController.getPower());
  });

// To inform HomeKit about changes occurred outside of HomeKit (like
user physically turn on the light)
// Please use Characteristic.updateValue
//
// lightAccessory
//   .getService(Service.Lightbulb)
//   .getCharacteristic(Characteristic.On)
//   .updateValue(true);

// also add an "optional" Characteristic for Brightness
lightAccessory
  .getService(Service.Lightbulb)
  .addCharacteristic(Characteristic.Brightness)
  .on('set', function(value, callback) {
    LightController.setBrightness(value);
    callback();
  })
  .on('get', function(callback) {
    callback(null, LightController.getBrightness());
  });
```

```
// also add an "optional" Characteristic for Saturation
lightAccessory
  .getService(Service.Lightbulb)
  .addCharacteristic(Characteristic.Saturation)
  .on('set', function(value, callback) {
    LightController.setSaturation(value);
    callback();
  })
  .on('get', function(callback) {
    callback(null, LightController.getSaturation());
  });

// also add an "optional" Characteristic for Hue
lightAccessory
  .getService(Service.Lightbulb)
  .addCharacteristic(Characteristic.Hue)
  .on('set', function(value, callback) {
    LightController.setHue(value);
    callback();
  })
  .on('get', function(callback) {
    callback(null, LightController.getHue());
  });
```