Vol.2016-CDS-16 No.14
2016/6/3

# ECHONET Lite パケット送受信ツール「SSNG for iPhone」の開発

藤田裕之[†1]　杉村博[†1]　村上隆史[†1]　一色正男[†1]

**概要**：iOS デバイス上で動作する ECHONET Lite パケット送受信ツール「SSNG for iPhone」を開発した．ECHONET Lite 仕様書に定義されている機器情報（EOJ, EPC, EDT）の詳細をツール内部に持たせることと iPhone の限られた画面サイズを考慮した UI をデザインすることで，ECHONET Lite パケット作成時のパラメータ設定が飛躍的に容易になった．また受信したデータ解析も容易に行える．このツールは ECHONET Lite エキスパートの作業効率を高めるだけでなく，ECHONET Lite 初心者がプロトコルを理解するためにも活用できる．

**キーワード**：エコーネットライト，ECHONET Lite，HEMS， iPhone，iOS，ツール

# Development of ECHONET Lite Packet Sending and Receiving Tool, SSNG for iPhone

HIROYUKI FUJITA[†1]　HIROSHI SUGIMURA[†1]
TAKASHI MURAKAMI[†1]　MASAO ISSHIKI[†1]

***Abstract***: We have developed a tool "SSNG for iPhone" to send and receive ECHONET Lite packets on an iOS device. Having detail device data in a tool such as EOJ, EPC and EDT defined in the specification of ECHONET Lite and making UI design considering limited screen size of iPhone makes it easy to decide parameters when ECHONET Lite packet is made. Received data analysis can be done easily too. Not only this tool makes ECHONET Lite experts more efficient, but also it helps ECHONET Lite beginners understand how the protocol works.

***Keywords***: ECHONET Lite, HEMS, iPhone, iOS, Tool

## 1. はじめに

Home Energy Management System(HEMS)が普及段階に入り，さまざまな ECHONET Lite 対応機器が販売され一般家庭に設置されるようになった．このような状況で，システム設置作業者や一般ユーザーが手軽に ECHONET Lite 対応機器の動作確認を行えるツールが求められている．

ECHONET Lite 対応機器の動作確認を行うには，任意の ECHONET Lite パケットを送信する機能を有するツールが必要である．このようなツールとしては神奈川工科大学で開発し無償で公開している SSNG[a]や，EneTrace[b]（株式会社シーイーシー），EW-ENET Lite Tester[c]（株式会社日新システムズ），Easy HEMS[d]（株式会社 TSP），えねっとくん[e]（テュフ ラインランド ジャパン株式会社）などの市販の製品がある．これらのツールは製品開発支援を主な目的とし，任意の電文を送信する機能だけでなく，シミュレーター機能やシナリオに基づく自動送信機能や受信した電文の詳細解析機能などの多機能を有する Windows PC 用のアプリケーションソフトウェアである．そのため機器が設置された現場で ECHONET Lite のエキスパートでない者が手軽に動作確認を行うという目的には不適当である．

神奈川工科大学 HEMS 認証支援センターでは ECHONET Lite の普及のためにこれまで幾つかの SDK やツールを開発して公開してきた[1]．今回はこのような背景を元に，現場で容易に動作確認を行えるツールとして iPhone や iPad などの iOS 上で動作する「SSNG for iPhone」を開発し公開した．App Store に「SSNG for iPhone」という名前で登録済み[f]である．

このツールは実用的であるだけでなく ECHONET Lite を学び始めた者がプロトコルの仕組みを理解するためにも大学内で活用されている．

## 2. 解決する課題

エンジニアが ECHONET Lite 機器を新規に設置した場合や設置済み機器の設定変更をした場合に，機器の動作確認を行うために使用するツールを想定する．現在この目的に特化したツールは存在せず，ECHONET Lite 開発支援ツールを利用している．

従来技術の例として SSNG の操作画面を図 1 に示す．SSNG は Windows PC 上で動作するアプリケーションソフ

†1 神奈川工科大学
Kanagawa Institute of Technology

a) https://smarthouse-center.org/sdk/
b) http://hems.cec-ltd.co.jp/enetrace/
c) http://www.co-nss.co.jp/products/middleware/enetlite-tester.html
d) http://www.tsp-net.co.jp/tsphtml/easyhems.html
e) http://www.xxcal.co.jp/testing/echonet-lite/about.html
f) https://itunes.apple.com/jp/app/ssng-for-iphone/id1076467789?mt=8

トウェアで，ECHONET Lite の全てのパラメータ（EHD[g], TID[h], SEOJ[i], DEOJ[j], ESV[k], OPC[l], EPC[m], PDC[n], EDT[o]）を任意の値に設定して送信可能なツールである．


図 1　SSNG の操作画面

SSNG などの開発支援ツールを動作確認目的として使用する場合には以下の課題が存在する．

- ノートパソコンのサイズと重量は不適当である．立ったまま片手で保持しもう一方の手で操作ができるハンドヘルドデバイスが求められる
- Windows PC のアプリケーションの起動時間は長すぎる．ツールがスリープ状態から使用可能な状態になるまでの時間としては数秒程度が求められる．
- シナリオに基づく自動送信機能や受信した電文の詳細解析機能など動作確認目的としては不要な機能があり，操作を複雑なものにしている．
- IP アドレスやインスタンス番号を選択するには，制御対象の IP アドレスやインスタンス番号があらかじめわかっている必要がある．
- EPC 欄や EDT 欄に入力するデータを決定するためには ECHONET 機器オブジェクト詳細規定[2]を参照する必要がある．この仕様書は約 500 ページの書類であり，この中から必要な情報を見つけ出すのは時間がかかる作業であり，正しい ECHONET Lite パケットを作成するのはエキスパートでなければ難しい．
- EDT 欄にデータを数値入力させることはエラーの原因ともなり不適切である．正しいフォーマットのデータを入力するのは ECHONET Lite のエキスパートでなければ難しい．
- ツールを別のネットワークに接続した場合，新しいネットワークを認識させるためにツールを再起動する必要がある

g) ECHONET Lite Header
h) Transaction ID
i) Source ECHONET Lite Object
j) Destination ECHONET Lite Object
k) ECHONET Lite Service
l) Operation Count
m) ECHONET Lite Property Code
n) Property Data Count
o) ECHONET Lite Data

## 3. 設計方針

２章で記述した課題を検討し，ECHONET Lite 機器の動作確認用ツールの設計方針を以下のように定めた．

### 3.1 iPhone アプリとしてツールを開発する

サイズや重量やアプリケーションの起動時間を考慮し，Windows PC ではなくスマートフォン用のアプリケーションソフトウェアでツールの機能を実現する．開発とデバッグ工数を考慮すると，画面サイズの違いや製品構成のバリエーションが少なくまた最新 OS に対応する端末の割合が高くアプリケーション配布環境が整備されているという理由で iPhone を選択した．

### 3.2 必要最小限の機能を実装する

使用目的が明確なので，機器探索，任意パケット送信，受信データ表示の３機能のみを実装する．シナリオに基づく自動送信機能や受信データのシーケンス解析などの開発支援向け機能は実装しない．送信する ECHONET Lite パケットは正常系のデータを前提とし EHD, TID, PDC はツールが自動で生成することで UI を簡素化する．ECHONET Lite の１パケット内には複数の要求を記述することが可能であり要求の個数を OPC というパラメータで記述するが, 機器の動作確認のためには複数要求を同時に送ることが必須ではないので，送信パケットの OPC は１だけを対応する．SetGet 命令は Set 命令と Get 命令を順次実行することで実現できるので対応しない．

### 3.3 操作マニュアルが不要な UI とする

画面構成は遷移，階層，モーダルを利用せず１画面で実現する．操作はボタンと Picker View のみで実現する．これにより最小限のアクションで必要とされる機能を実行できる．操作マニュアルの代わりにチュートリアルビデオ[p]を用意した．

### 3.4 IP Address 選択を支援するための機器情報を表示する

IP Address に対応した機器の自ノードインスタンスリスト S（ノードプロファイル:EPC=D6）やメーカーコード（ノードプロファイル:EPC=8A）を自動で取得し，取得したデータをもとに機器オブジェクト名（例：家庭用エアコン，一般照明など）やメーカー名を表示することで，IP アドレス選択が容易になる．

SSNG では制御対象機器のインスタンス番号を指定する必要があるが，本ツールでは取得したインスタンスリスト内のインスタンス番号を利用する．

### 3.5 送信パケットのパラメータは選択肢から選択する

送信パケット作成時のパラメータは値を直接入力するのではなく，選択肢から選択するようにする．そのための UI は Picker View を利用する．選択した値に対応した description を表示する．例えば EOJ で 0130 を選択すると"

p) https://www.youtube.com/watch?v=WjfFF61i5Mw

エアコン"，EPC で 80 を選択すると"動作状態"，EDT で 30 を選択すると"ON"と表示する．これにより「ECHONET 機器オブジェクト詳細規定」を参照しなくても送信用パケットのパラメータを選択することができる．

温度設定値のような連続値に関しては 10 個程度の代表値から選択する．キー入力ではなく選択肢からデータを選択することで操作が容易になるだけでなく，正しいフォーマットで定義された値域内のデータを使用できる．

### 3.6 受信データの簡易解析機能を用意する

SSNG の場合受信データ解析機能がないため，受信したデータを理解するためにはECHONET機器オブジェクト詳細規定を参照する必要がある．本ツールは画面サイズを考慮して以下の解析機能を設ける．

A) 受信したデータは EHD, TID, SEOJ, DEOJ, ESV, OPC, EDT の各要素の間にスペースを入れて表示する(図２参照)．これにより各要素のデータを見つけやすくなる．

B) データ受信時は EDT 値設定用 Picker View を流用し，受信データの EDT 値の意味を検索できるようにする．

C) ノードプロファイルの状変アナウンスプロパティマップ(EPC=0x9D)または Set プロパティマップ(EPC=0x9E)または Get プロパティマップ(EPC=0x9F)のデータを受信した際にデータが記述形式２[3]のビットマップ形式の場合，そのままでは利用できないのでデータをデコードして EPC 番号を表示する．

### 3.7 ノードプロファイルとしての最小限の実装を行う

ツールの実装を容易にするため ECHONET Lite のコントローラオブジェクトとしての実装はしない(Get や Set のコマンドを受信した場合に処理を行わない)．しかし操作命令の送信元のノードオブジェクトから最低限の情報を取得できないと操作を受け付けない機器が存在するので，ツールが以下の EPC に対する GET を受信した場合は適切な値で GET_RES を返答する．またツール起動時にインスタンスリスト通知(EPC=0xD5)をマルチキャストアドレス宛に ESV=INF で送信する．

| |
|---|
| 0x80: 動作状態 |
| 0x8A: メーカーコード |
| 0xD6: 自ノードインスタンスリスト S |

### 3.8 簡易リファレンスブックの機能を持たせる

IP Address の Picker View で 224.0.23.0（マルチキャストアドレス)を選択すると EOJ 欄にツールが実装している全ての機器オブジェクトを表示して選択できるようにする．パラメータ設定機能を利用して EOJ, EPC, EDT の値の意味を検索する機能を実現する．

## 4. 提案するシステムについて

### 4.1 ユーザーインターフェイスの説明

図２に SSNG for iPhone の画面を示す．この１画面で全ての機能が実装されている．画面は上から操作，設定，表示の３つのブロックで構成されている．

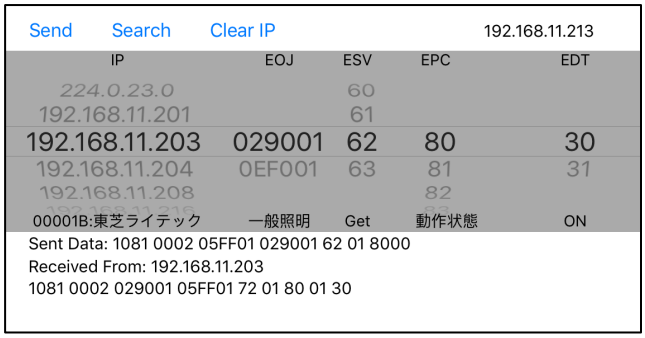


図 2　SSNG for iPhone の画面

#### 4.1.1 操作ブロック

Send, Search, Clear IP のボタンと IP アドレス表示で構成される．

Send ボタン：設定ブロックで選択したパラメータ値で ECHONET Lite パケットを作成し送信する．

Search ボタン：ネットワーク内の ECHONET Lite 機器を探索し，その結果を設定ブロックの IP 欄に表示する．

Clear IP ボタン：設定ブロックの IP 欄を初期化する．

IP アドレス表示部：iPhone の IP アドレスを表示する．WiFi に接続していない場合は「Check Network」というメッセージを表示する．接続する WiFi ネットワークを変更した場合は自動で表示を更新する．

#### 4.1.2 設定ブロック

パケット送信先の IP Address やパケットの各項目（EOJ, ESV, EPC, EDT）のデータを設定するブロックである．データは全て Picker View で選択する．

ツールの起動直後または Clear IP 機能実行直後は IP 欄には 224.0.23.0（マルチキャストアドレス）だけが表示されている．Search 機能を実行すると同一ネットワーク内の ECHONET Lite 機器の IP Address が表示される．

EOJ 欄はノードプロファイルと機器オブジェクトが表示される．同一ノードに異なる機器オブジェクトが存在する場合（例：冷温水熱源機と床暖房）や同一機器オブジェクトで複数のインスタンスが存在する場合にも対応する．IP 欄で 224.0.23.0 を選択すると EOJ 欄にはツールが実装している全ての機器オブジェクトが表示される．

ESV 欄には動作確認に必要なサービス（60:SetI, 61:SetC, 62:Get, 63:Inf_Req）に対応した値が表示される．

EDT 欄は EPC の値に応じて選択肢が表示される．

Picker View で値を選択するとデータに対応する description が各欄の下部に表示される．例えば EOJ 欄の

「029001」に対しては「一般照明」，ESV 欄の「62」に対しては「Get」，EPC 欄の「80」に対しては「動作状態」が表示される．この機能により，ECHONET 機器オブジェクト詳細規定を参照しなくても値を決定することができる．

#### 4.1.3 表示ブロック

送受信データの表示ブロックである．

１ライン目:送信データを表示する．EHD, TID, SEOJ, DEOJ, ESV, OPC, EDT の各要素の間にスペースを挿入することで各要素のデータを見やすくしてある．

２ライン目：受信データの送信元 IP アドレスを表示する

３ライン目:受信データを表示する．EHD, TID, SEOJ, DEOJ, ESV, OPC, EDT の各要素の間にスペースを挿入することで各要素のデータを見やすくしてある．Get_Res だけでなく INF として受信したデータも表示する．

４ライン目：受信データがプロパティマップの記述形式２の場合，データをデコードして EPC のリストを表示する．

### 4.2 代表的な操作の動作説明

#### 4.2.1 機器探索

同一ネットワーク上の ECHONET Lite 機器を探索する方法として ECHONET Lite の規格書[4]では「コントローラからノードへのメッセージ送信による発見」として以下のパケットをマルチキャストアドレス(224.0.23.0)に送信することを推奨している．

[DEOJ=0x0EF001：ノードプロファイルオブジェクト，ESV=0x62：プロパティ値読み出し要求(Get)，EPC=0xD6：自ノードインスタリスト S]

Search ボタンをタップすると上記パケットを送信する．各機器が返信した IP Address と自ノードインスタリスト S 内の機器オブジェクトコードを元に IP 欄と EOJ 欄の値を更新する．

次に探索した各機器からメーカーコードを取得するために以下のパケットをユニキャストで送信する．

[DEOJ=0x0EF001：ノードプロファイルオブジェクト，ESV=0x62：プロパティ値読み出し要求(Get)，EPC=0x8A：メーカーコード]

受信したメーカーコードに対応したメーカー名を IP 欄の下部に表示する（図３参照）．

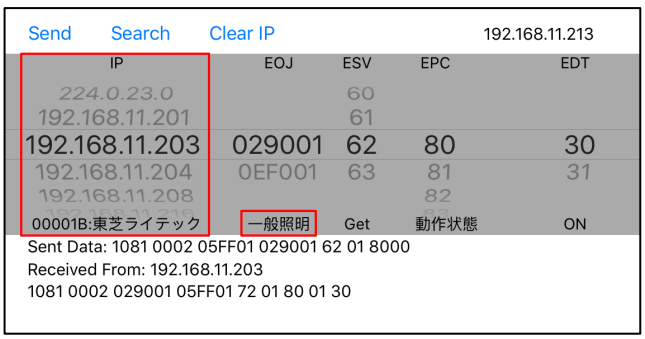


図 3　機器探索画面

#### 4.2.2 ECHONET Lite パケット送信

まず設定ブロックの各 Picker View で値を選択する．

IP 欄で IP Address を選択すると図４のように IP 欄と EOJ 欄の下部にそれぞれメーカー名（例：東芝ライテック）と機器オブジェクト名（例：一般照明）が表示されるので，これを参照して制御対象の IP Address を選択する．

EOJ 欄でノードプロファイルまたは機器オブジェクトを選択する．

ESV 欄でサービスを選択する．

EPC 欄には選択した EOJ と ESV に対応した値が表示されるので EPC 欄下部の description（例：動作状態）を利用して EPC を選択する．

ESV が 60:SetI または 61:SetC の場合は EDT を選択する．

全てのパラメータ設定完了後 Send ボタンをタップするとツールは ECHONET Lite パケットを作成し送信する．

送信データは表示ブロックの１ライン目に表示される．

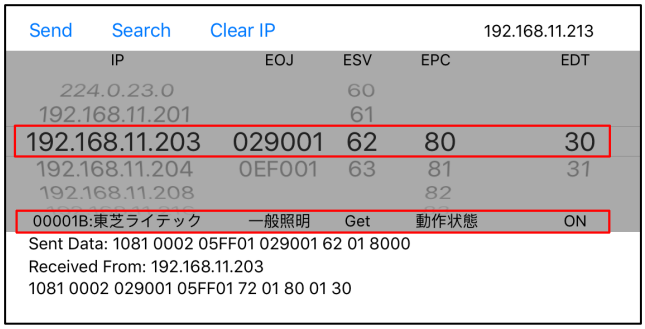


図 4　送信画面

#### 4.2.3 ECHONET Lite パケット受信

ECHONET Lite データを受信すると送信元の IP Address と受信データを表示ブロックに表示する．3.6 節の (A)で説明したように，要素間にスペースを挿入し各要素を見つけやすくしている．

EDT の Picker View を利用して受信したデータの意味を解析することができる．

3.6 節の (C)で説明したように受信したデータがプロパティマップで記述形式２のビットマップ形式の場合，図５に示すように EPC 番号にデコードして表示する．

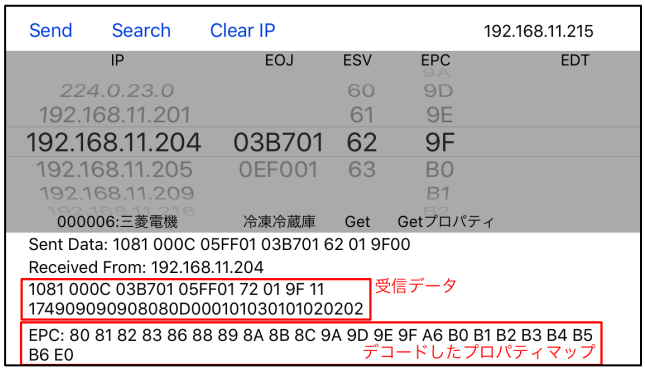


図 5　受信画面

### 4.3 ソフトウェア構成

図６にソフトウェア構成図を示す. GCDAsyncUDPSocket より上のレイヤーのモジュールを今回開発した．開発言語は Swift である．

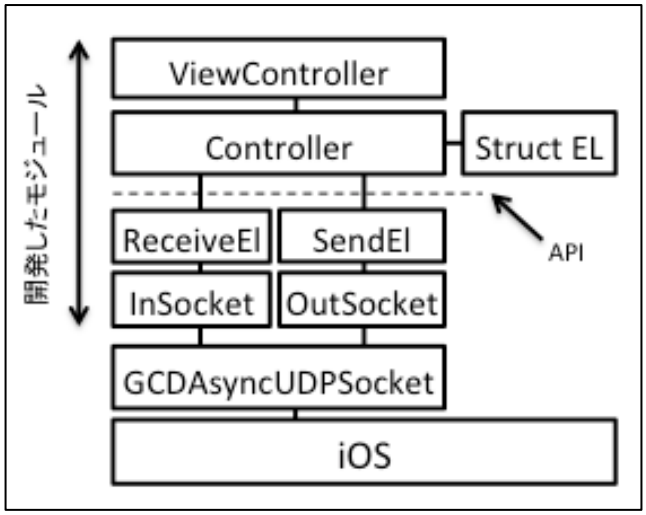


図 6　ソフトウェア構成図

#### 4.3.1 各モジュールの説明

GCDAsyncUDPSocket[q]は Socket を使った UDP の非同期通信 Framework である．Public Domain のライセンスで GitHub から配布されている. Objective-C で記述されているので CocoaPods[r]を利用してインストールした．

InSocket は受信用の Socket を作成しバイナリデータを受信するモジュールである．

OutSocket は送信用の Socket を作成しバイナリデータを送信するモジュールである．

ReceiveEl は InSocket から渡された受信バイナリデータをパースして ECHONET Lite のフレーム（EHD, TID, SEOJ, DEOJ, ESV, OPC, EPC + EDT）に格納する．

SendEL は，ECHONET Lite のフレームに格納されたデータをバイナリデータにパッキングし OutSocket に渡す．

Controller は Model View Controller (MVC) デザインパターンの Model に相当し，送受信のロジックを扱う．ECHONET Lite Packet を受信した場合はデータを解析し，内部状態を更新したり Packet の送信元に返答（Get_Res, INF, SNA）したりする．例えばインスタンスリストやメーカーコードを受信した場合は Controller 内の機器オブジェクトリストを更新する．ViewController から Packet の Send の指示があると UI で設定された値と, increment した TID の値を SendEl に渡して送信の指示を出す．ViewController から Search（機器探索）の指示があると 4.2.1 機器探索で示した ECHONET Lite Packet のデータを設定して SendEl に送信の指示を出す．

StructEL は ECHONET 機器オブジェクト詳細規定に記述されている機器について EPC や EDT などに関連する定数を記述したモジュールである．エアコン（EOJ=0130）の運転モード（EPC=0xB0）を例としてソースコードを以下に示す．順にプロパティ名称，データサイズ，Set 属性，必須項目属性，INF 属性，EDT 値の候補を記述している．

```
struct ElProperty {
    var name: String
    var size: Int
    var set: Bool
    var required: Bool
    var anno: Bool
    var edt: [UInt:String]?
}
static let eoj0130epcB0 = ElProperty(
      name: "運転モード",
      size: 1,
      set: true,
      required: true,
      anno: true,
      edt:[0x41:"自動",0x42:"冷房",0x43:"暖房",0x44:"除湿
",0x45:"送風",0x40:"その他"])
```

ViewController は MVC デザインパターンの Controller に相当し GUI を制御する．

#### 4.3.2 モジュールの再利用と API

ECHONET Lite Packet 送受信機能を再利用することを考慮して ReceiveEl 及び SendEl モジュールと上位のモジュール間のインタフェース(API)を定義した．

```
class ReceiveEl {
    var address = String()
    var tid = [UInt8(), UInt8()]
    var seoj = [UInt8(), UInt8(), UInt8()]
    var deoj = [UInt8(), UInt8(), UInt8()]
    var esv = UInt8()
    var messages = [ElMessage]()
      // 実装部分は省略
}
struct ElMessage {
    var epc = UInt8()
    var edt: [UInt8] = []
}
```

受信データをアクセスするには property として定義された各変数の値を読み出す．EPC, PDC, EDT で構成される要求が複数の場合にも対応するために ElMessage という Struct を定義し，それを Array に格納した messages という property を定義した．

q) https://github.com/jbenet/ios-ntp/blob/master/network-udp/GCDAsyncUdpSocket.h
r) https://cocoapods.org

```
class SendEl {
    var seoj = [UInt8(), UInt8(), UInt8()]
    var deoj = [UInt8(), UInt8(), UInt8()]
    var esv  = UInt8()
    var messages = [ElMessage]()
    func send(address: String) -> Bool
      // 実装部分は省略
}
```

データを送信するには各 property に値を書き込み，送信先 IP address を引数として send という method を実行する．

この API を利用することで，ReceiveEl, SendEl, InSocket, OutSocket, GCDAsyncUDPSocket のモジュールを再利用して効率的に ECHONET Lite 関連のソフトウェアが開発できる．これまでに iPhone 用アプリ「温度センサーEL」[s]や，研究[5]に記載されている省エネ空調制御システム「ヤドリギ」で利用されている．

### 4.4 アプリケーションライフサイクル

iOS はハードウェアリソースの有効活用のためにアプリケーションが Inactive になった後しばらくすると UDP 通信の Socket を close してしまう．ただし Inactive 状態が数秒以内で再び Active に戻る場合は Socket が close されないこともある．そこで Socket の状態を監視するのではなく Inactive になった時点でアプリケーションが Socket を close し Active になった時点でアプリケーションが Socket を open する実装とした．これによりシンプルな実装で安定して Socket 通信が行える．また Socket を open するタイミングで iPhone 自身の IP Address を読み出し UI の IP Address 表示を更新する．これにより WiFi ネットワークの接続先を変更した場合にも自動で対応できる．

### 4.5 対応機器実装レベル

本ツールが対応している機器の種類や機能のレベルに関して説明する．

ECHONET 機器オブジェクト詳細規定に記載されている機器情報は膨大な量である．Appendix G を例にすると総ページ数は 493，機器の種類は 178，EPC 数は機器オブジェクトに依存するが，例えばエアコンの場合 64 である．各 EPC に対して設定値に対応する EDT が定義されているが，例えばエアコンの運転モード（EPC=0xB0）に対応する EDT 数は自動, 冷房, 暖房, 除湿, 送風, その他の 6 である．

全ての機器オブジェクトの EPC, EDT を実装するのは大変なので今回はセンターに設置されている 20 種類の機器に関して実装した．EPC に関しては必須機能とよく使われる機能を実装した．EDT に関しては，エアコンの運転モードのような離散値に関しては全ての設定値を，温度設定値のような連続値に関しては 10 個程度の代表値を実装した．以下に実装した機器の EOJ と機器種別名を記載する．

0011:温度センサー, 0012:湿度センサー, 002D:気圧センサー, 0130:エアコン, 0133:換気扇, 0135:空気清浄機, 0260:電動ブラインド, 026B:温水器, 026F:電気錠, 0272:給湯器, 0273:浴室暖房乾燥, 0279:太陽光発電, 027A:冷温水熱源機, 027B:床暖房, 027D:蓄電池, 0287:分電盤, 0288:低圧スマート電力量メータ, 0290:一般照明, 03B7:冷凍冷蔵庫, 05FF:コントローラ

## 5. 評価

神奈川工科大学ホームエレクトロニクス開発学科の学生５人に以下の内容で評価をしてもらった．

1. チュートリアルビデオを視聴する
2. ツールを使って以下の操作を行う
2.1 エアコンの電源を ON，動作モードを冷房，設定温度を 25 度にする
2.2 一般照明の電源を ON，動作モードを RGB モード，色をマゼンタにする

評価項目は以下の通り

- チュートリアルビデオを視聴して使用方法が理解できたか
- SSNG と比較して操作性はどうか

評価結果

- ５人ともチュートリアルビデオの視聴だけで使用方法を理解し，必要な操作を容易に実現できた
- ５人とも SSNG for iPhone が SSNG より使い易いと評価した

これにより当初の目的である「設置作業者や一般のユーザーが手軽に ECHONET Lite の動作確認ができるツールの開発」が達成できたと考えられる．

## 6. 結論

### 6.1 実現方法

携帯性に優れタップやスワイプなどのシンプルな動作で操作できるというスマートフォンの特徴を活かしたツールを実現することができた．

### 6.2 UI デザイン

画面を操作・設定・表示の３ブロックに分け，データ選択を全て Picker View で行い, 操作ボタンは大きく文字サイズにメリハリをつけることで，限られた画面サイズにもかかわらず使い易い UI デザインを実現できた．

### 6.3 実装方法

Swift 言語を利用したオブジェクト指向のモジュール設計や MVC デザインパターンを利用することにより ECHONET Lite パケット送受信機能を再利用しやすく実装することができた．

s) https://itunes.apple.com/jp/app/wen-dusensael/id1060923743?mt=8

## 7. 今後の課題

### 7.1 対応する EOJ, EPC, EDT の充実

現在の実装では ECHONET Lite で定義されている機器オブジェクト全体の約 1/9 にしか対応していない．対応している機器オブジェクトの全ての EPC や EDT に対応しているわけではない．また ECHONET 機器オブジェクト詳細規定は１年に２度改定されている．

対応する EOJ, EPC, EDT を充実しタイムリーにアップデートしていくためには現在の実装のように機器毎の情報をプログラムにハードコーディングするのではなく，まず ECHONET 機器オブジェクト詳細規定の機器情報全体をデーターベース化し，そこから JSON や XML ファイルに変換した上でプログラムに読み込む仕組みを検討したい．

### 7.2 EDT 値設定に対する柔軟性

現在の実装では連続値として定義されている EDT 値の設定方法は Piker View 上の候補値からの選択であり，候補値以外の値は設定できない．スライダーなどの操作性を損なわない方法で任意の EDT 値を入力できる方法も検討したい．

また低圧スマート電力量メータの積算電力量計測値履歴１(EPC=E2)のように複数のデータ（積算履歴収集日と積算電力量計測値）が１つの EDT で表現されているものもある．このような値への対応も検討課題である．

### 7.3 複数 OPC への対応

AIF 認証[t]の試験項目には複数 OPC のパケット送信がある．製品の高度な動作確認を考慮すると複数 OPC のパケットへの対応も検討課題である．

## 8. おわりに

SSNG for iPhone は ECHONET Lite 規格の普及促進の一助として開発した．すでに Apple 社の App Store に「SSNG for iPhone」という名前で登録済みで，誰でも利用できる状態にある．多くの人に使っていただき少しでも使いやすくするための改良を加えていくつもりである．

**謝辞** SSNG for iPhone の評価にご協力頂いた神奈川工科大学ホームエレクトロニクス開発学科の学生に，謹んで感謝の意を表する．

## 参考文献

[1] 藤田裕之. HEMS サービス開発の要となる ECHONET Lite の実行環境や国際競争力を考慮した SDK の開発. 神奈川工科大学研究報告 2016.3 ISSN 2188-2878
[2] “ECHONET 機器オブジェクト詳細規定 Release G Revised”. http://echonet.jp/spec_object_rg_revised, 2015-06-20
[3] “ECHONET 機器オブジェクト詳細規定 Release G Revised”. http://echonet.jp/spec_object_rg_revised, 2015-06-20, 付録 1
[4] “ECHONET Lite SPECIFICATION Version 1.12 第５部 ECHONET Lite システム設計指針”, Sep. 30, 2015, p4-1
[5] 小田原健雄, 岡本健司, 藤田裕之, 三栖貴行, 一色正男. 暖寒色を利用した省エネ空調制御システムの構築. 情報処理学会第７８回全国大会

t) http://echonet.jp/wp/wp-content/uploads/pdf/Notification/jp/General/20151225/3_AIF 認証制度の概要.pdf