事例紹介

「IoTと情報セキュリティの課題」

ソフトウェアテストシンポジウム 2018 新潟

JaSST'18 Niigata

開催 2018年4月20日(金)

今岡工学事務所
今岡 通博

<https://www.facebook.com/imaoka.micihihiro>

imaoca@gmail.com, @imaoca

<http://www.itmedia.co.jp/author/208685/>

<https://fpgastartup.connpass.com/>

<https://blogs.msdn.microsoft.com/devwire/>

今岡 通博

imaoka michihiro

@imaoca

今岡工学事務所 個人事業主代表

組込み系受託開発、FPGA応用製品、書籍等執筆、回路設計、
基板設計、非常勤講師

愛媛県松山市在住

- ・ 元日本DEC(Digital Equipment Corp.)他
- ・ 最近では原稿を書いていることが多い
- ・ セキュリティ・キャンプの講師とか
- ・ ハードウェアスタートアップ支援
- ・ Microsoft MVP(Windows Development)
- ・ FPGAスタートアップ主宰
- ・ Androidの会コミュニティ運営委員
- ・ 愛媛県警察部外通訳員 (英語、西語)

あせ世界 → CPU

あせ (red arrow pointing to 'せ')
あせ (red arrow pointing to 'せ')

- 1. 家
- 2. 藤
- 3. 木
- 4. 鈴
- 5. 布
- 6. 田
- 7. 伊
- 8. 辻

ELVM

↓

—

—

—

Browser window showing the Microsoft MVP profile for michihiro Imaoka (Hiro).

Microsoft MVP
Most Valuable Professional

michihiro Imaoka (Hiro)
Japan
Matsuyama, Ehime

I am individual programmer to a embedded system for FPGA, design schematic and PCB .Writing a book and some article for computer related magazine. See <https://www.facebook.com/imaoka.micihihiro> or twitter: @imaoca

Biography

I live in matsuyama in japan and i was born in 1962. I have a wife and a child.
worked for Digital Equipment Corporation(1985) & S.G, Warburg Sec.(1990).
I have been in Costa Rica for two years as Japan Overseas Cooperation Volunteers(JOCV). as invite professor in Costa Rica Latin university.(1995-1998)
Ehime University Doctor Course (Science & Engineering). (2005-2008)
Also I am trainer for Security Camp for Information Processing Agency(IPA) (2015-)

Recent activities

Date	Activity	Type	Contribution Area
1/15/2018	なぜMicrosoftはFPGAを採用したのか	Article	Windows Hardware Engineering (IoT Mobile and Desktop)

Award Categories
Windows Development

First year awarded:
2014

Number of MVP Awards:
4

Contact
Email: imaoca@gmail.com

<https://mvp.microsoft.com/ja-jp/PublicProfile/5000851>

imaoca@gmail.com

twitter: @imaoca

<https://www.facebook.com/imaoka.micihihiro>

今岡工学事務所船籍のヨット2艇

夢は海上移動オフィス

セッションの内容

- ・バズワードとなっていて久しいIoTですが、日本語では「もののインターネット」と訳されて認知されてはいるものの、その実もっともとらえどころのない技術用語のひとつとなっています。
- ・IoTを支える基盤は組込技術、センシング技術、情報通信技術、クラウド技術の上に成り立っています。これらの複合技術の上に成り立っているテクノロジーゆえすべてを俯瞰的に見通せるスキームが確立されていません。
- ・今回はIoTの本質と、組込開発におけるセキュリティを担保するためのテスト等について事例を交えこれらの課題について議論したいと思います。

IoTの本質

IoTにおけるセキュリティ
を担保するためのテスト

IoTにおけるセキュリティ

- データの改ざん
- なりすまし
- 乗っ取り、踏み台
- マルウェア
- 通信の傍受
- 通信妨害

今回は通信妨害について
考えます

通信妨害に関して
IoTのプロトコルとしてデファクト
になりそうなMQTTというプロトコルを
事例に考えます。

IoTにおけるMQTT のユースケース

インターネットにつながったエンドノードに接続されたLEDをスマートフォンから制御します。

この仕組みを応用すればインターネットにつながった世界中の機器を電氣的に制御可能であることを示唆します。

もっともシンプルなIoT実装モデル。

いくつかの単語

エンドノードにはM5STACKを使用

接続プロトコルはMQTT

スマホではIoT Dashboardを使用

BrokerはMosquitto

MessageはBrokerを介してSubscriberに送られる

Subscriberは受け取りたいMessageのTopicを指定

デモの構成

M5Stackとは

ESP32が入っています。

WiFi とBluetoothの機能

Tensilica の32bit 2コアCPU

メモリー:520KiB SRAM

AD,DA,SPI,I2C,UART,GPIO

カラーLCD内蔵 (320X240)

Arduinoの開発環境を流用

中国から買えば \$ 31だった

Microsoft Azure Certified

詳しくはここを見てください

<http://m5stack.com/>

MQTTとは

Message Queuing Telemetry Transportの略
MQTTはM2MやIoTのための接続プロトコルで
Publisher, Broker, そしてSubscriberで構成
軽量で高速（と言われている）
TCPより上のレイヤー
<http://mqtt.org/>

Mosquittoとは

MosquittoはMQTTのひとつの実装
もっともMQTT仕様書に忠実な実装で他の実装の
リファレンスとなる

Broker, Client(Publisher, Subscriber)ともオープン
ソース

ポートは通常1883

<https://mosquitto.org/>

Mosquitto

IoT Mqtt Dashboard

Androidで動作するMQTTクライアントアプリ
publisherとsubscriberとして動作可
簡単な操作でボタンなどのGUIを作成可
メッセージが数値の場合グラフ表示が可

M5Stackのプログラミング

```
m5TempPub | Arduino 1.8.5
File Edit Sketch Tools Help
m5TempPub
20
21 WiFiClient espClient;
22 PubSubClient client(espClient);
23 long lastMsg = 0;
24 char msg[50];
25 int value = 0;
26
27 void setup_wifi() {
28
29     delay(10);
30     // We start by connecting to a WiFi network
31     Serial.println();
32     Serial.print("Connecting to ");
33     Serial.println(ssid);
34
35     WiFi.begin(ssid, password);
36
37     while (WiFi.status() != WL_CONNECTED) {
38         delay(500);
39         Serial.print(".");
40     }
41
42     randomSeed(micros());
43
44     Serial.println("");
45     Serial.println("WiFi connected");
46     Serial.println("IP address: ");
47     Serial.println(WiFi.localIP());
48
49     client.setServer(1122);
50     client.setNoDelay(true);
51     client.connect();
52     Serial.println("m5TempPub connected to server");
53 }
54
55 void loop() {
56     if (client.available()) {
57         String data = client.readStringUntil('\n');
58         Serial.print(data);
59         Serial.println();
60     }
61     if (value < 100) {
62         value++;
63     }
64     client.print(value);
65     client.println();
66     delay(1000);
67 }
68
69 http://www.m5stack.com/download/package_m5stack_index.json file signature verification failed.
70 Invalid library found in C:\Users\#michihiro#Documents\#Arduino#hardware\#espressif\#esp32#libraries
71
72 M5Stack: Core-ESP32, QIO, 80MHz, 921600, None on COM7
```

最新版は以下のgithubで公開予定 <https://github.com/imaoca>

スマホからLEDを制御する画面

M5StackのLEDが点灯の様子

メッセージ"0"を受信

メッセージ"1"を受信

通信妨害のテスト

MQTTはIoTのプロトコルに適しており、不安定な回線や通信妨害に強いプロトコルだとされています。特に通信の品質を保証するQoSを何段階かのレベルで設定できるようになっており、通信妨害の耐性およびデータの重要性に応じて選択が可能です。

今回のテストではQoSレベルと通信妨害の耐性について観察します。

QoS(サービス品質)

- QoS 0 メッセージは1回送られる
- QoS 1 メッセージは最低1回以上届けられる
- QoS 2 メッセージ必ず1回届けられる

QoSの測定方法

Publisherから連番のメッセージを送信し

Subscriberがそれを受信し抜けあるいは重複をカウントする。

クライアントとブローカー間の回線に通信妨害発生器を設置する。

通信妨害の階級を段階的に上げていき測定する。

通信妨害の階級はpingの損失レートを基準とする。

QoS実験システム構成

通信妨害発生プログラム(一部抜粋)

```
while(1) {  
  digitalWrite(13, HIGH);  
  delay(500);           // 回線がショート状態の時間(ms)  
  pinMode(13,LOW);  
  delay(500);           //回線が通常状態の時間(ms)  
}
```

回線のショート状態と通常状態を一定間隔で繰り返す。

回線がショート状態

回線が通常状態

Pub/Sub 1 対 1 の場合

障害の周波数を変える（但しデューティ比は50%）

PUB	SUB	損失	重複	ping損失	time
QoS 0	QoS 0	0	0	0	6
QoS 1	QoS 1	0	0	0	8
QoS 2	QoS 2	0	0	0	6
QoS 0	QoS 0	0	0	65% ※1	92
QoS 1	QoS 1	0	0	65% ※1	88
QoS 2	QoS 2	0	0	65% ※1	73
QoS 0	QoS 0	11	0	63% ※2	267
QoS 1	QoS 1	9	0	63% ※2	258
QoS 2	QoS 2	12	0	63% ※2	301
QoS 0	QoS 0	31	0	69% ※3	800
QoS 1	QoS 1	24	0	69% ※3	772
QoS 2	QoS 2	17	0	69% ※3	798

※1 500ms間隔で回線がショート状態となる。 ※2 250ms ※3 100ms

考察

通信妨害の頻度（周波数）を変えても

ping(UDP)にはそれほど影響なし。

MQTTなどTCP系のプロトコルでは、回線が

つながっているトータルの時間は同じで

も(50%は回線がつながっている)障害の

頻度が増すと、エラー率およびスルー

プットに著しい悪影響を及ぼす。

通信妨害の程度が大きいとQoS2の方がエ

ラー率において多少有利。

独自プロトコルとの比較

UDPであるICMPが障害の程度に関わりなく
それなりにパケットが通ることからUDP
ベースのプロトコルを試してみたい。
通常のシーケンスでは回線がショートする
度にARPのやり取りが発生するが、ター
ゲットのMACアドレスは固定とする。

まとめ

ご清聴ありがとうございました。

今岡 通博