# サーバー上でのデータベースでセンサーのレジスタを用いた拡張性の高い識別方式

五味 滉人[1,a)]　串田 高幸[1]

**概要**：ソフトウェアはどのセンサーがついているかわからない．なので，センサーの特長量 (レジスタ) を用いてあらかじめ特長量 (レジスタ) とセンサーを紐づけているサーバ上のデータベースの中からセンサーを一意特定する．そのうえで，ユーザがデータシートを読むことやコーディングをしなくてもデータを取れる段階まで自動的にセットアップする．また，サーバを作成してセンサーの特長量 (レジスタ) とセンサー名とライブラリを管理する．Manufacture ID を用いた識別を行い，識別によってライブラリを取得できるようにしたことで，ユーザーが作業しなければならない工程を省くことができた．本研究によって，センサーの識別を行い，自動セットアップをできるようにすることで，IoT の発展に貢献する。

## 1. はじめに

### 背景

Internet of Things(以下 IoT と称する) という用語は，コンピュータ，ラップトップ，スマートフォンの標準的なデバイスを超えて，インターネット接続を拡張する能力を感知または作動させることができるローコンピューティングデバイスのグループとして定義されています．[1] IoT は，人，モノ，環境を統合したもので，スマートホーム，電子商取引，コネクテッド・ヘルスケア，スマートシティの新しいアプリケーションの助けを借りて，私たちの日常生活を一変することができる．IoT には大きな影響力があり，学術界と産業界の両方から大規模な研究開発の関心を集めていて，経済と社会に影響を与えている．また，IoT の普及はコンピューティングのメガトレンドとして進行中であり，2020 年には IoT デバイスの数が 240 億台に達することが示唆されています．また，McKinsey の推定では 2025 年には 250 億から 500 億のデバイスが存在すると予想していて増加傾向にある．[2] [3] なので IoT デバイスの急増により家庭，オフィス，建物，芝生，都市，さらには農業農場にも様々な種類のセンサーが導入されてきている．[4] そのため，デバイスが増加することは，様々なデバイスを相互に接続するデバイスをつなぎ合わせる新たな機会を提供することができる．それらのデバイスを識別する必要性は，あらゆる種類の操作を行う上で重要である．[5] さらに，デバイスに取り付けているセンサーもデバイスと同様に増加した場合，それらのセンサーを識別する必要性は，重要である．

また，IoT には膨大な量のセンシングデバイスが様々な分野やアプリケーションの様々なデータの収集または生成します．これらのデバイスは高速/リアルタイムのデータストリームになる．このようなデータストリームに分析しを適用して，新しい情報を発見し，将来の洞察を予測し，制御の意思決定を行うことは，IoT ビジネスと生活の質を向上させるテクノロジーにとって価値のあるパラダイムにする重要なプロセスになる．[6]

IoT には，ハードウェア，ソフトウェア，データベース，セキュリティの幅広い知識が必要になっていて，IoT Device として ESP-WROOM-32 が実用的なところで使用されている。

センサーの種類に I2C センサーというものがある．I2C は PhilipsSemiconductor によって検出されたシリアルバスプロトコルで低速デバイスをプロセッサに通信するために使用される．これは，データを高速のデバイスが低速のデバイスと通信できるようにするために使用される．I2C の仕様として，I2C はシリアル転送バスであり，8 ビット指向で 2 本の信号線を使用することにより，データ送信が行われる．SCL と SDA の 2 つの信号線があり，SCL はシリアルクロックライン，SDA はシリアルデータラインがある．両方のラインがマスターとスレーブ間データを転送されるために使用される．多くのスレーブデバイスが I2C バスに接続されており，スレーブデバイスは一意の 7 ビットアドレスによって識別が行われる．I2C バスには，アイド

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[a)] C0118116

ル，開始，停止の 3 つの条件があり，その条件は SDA と SCL の 2 本の線を使用して示される．[7]

### 課題

本研究の課題は，1 つ目にデバイスに取り付けているセンサーが増加した場合，それらのセンサーを識別する必要がある点．2 つ目に IoT デバイスにはセンサーが取り付けられているが，どのセンサーがついているかソフトウェア側からではわからない点．3 つ目にセンサーを取り付けてからデータを取得するまでに時間がかかってしまう点の 3 つの課題がある．

1 つ目と 2 つ目の課題に対して，既存方式では，ID を付与してその ID を用いて識別を行っている．

3 つ目の課題に対して，既存方式では，回路を組んでからライブラリを作成してデータの取得を行っている．

これらの課題を解決するため，本論文では，サーバー上のデータベースを用いてライブラリを管理して，センサーの Manufacture ID とデータベース内に登録してある Manufacture ID を紐づけてライブラリをダウンロードし，データを取れる段階まで自動的にセットアップするシステムを提案する．

### 各章の概要

第 2 章では本論文の関連研究について説明する．第 3 章では第 1 章で説明した課題を解決をするシステムの提案について説明する．第 4 章では本論文で提案するシステムの実装と実験環境について説明する．第 5 章では本論文で提案するシステムの評価と分析について説明する．第 6 章では本論文で提案したシステムの議論を行う．第 7 章では本論文でのまとめを行う．

## 2. 関連研究

この章では，本研究と関連する既存研究について取り上げ，その違いについて述べる．

Meidan らの論文 [8] では，ネットワークに接続された IoT デバイスを正確に識別するために，ネットワークトラフィックデータに機械学習アルゴリズムを適応している．この研究では，分類器の訓練と評価を行うために，ネットワークトラフィックデータを収集し，ラベル付けを行い，教師付き学習を用いて，分類器の訓練を行っている．しかし，機械学習はコストと時間と初期の精度が低くなってしまうという問題がある．

Aftab らの論文 [5] では，いくつかの IoT プラットフォームで使用されている IoT のための識別スキームを分析し，比較している．この論文では，識別を行う必要性を重要視していて，IoT プラットフォームで，統一された識別方法がないことを問題にしている．

Meidain らの論文 [9] では，ホワイトリストから IoT デバイスの種類を正確に特定することを目的として，ネットワークトラフィックデータから抽出した特長量に対して，教師付き機械学習アルゴリズムである Random Forest を適用している．しかし，機械学習は，コストと時間と初期の精度が低くなってしまうという問題がある．

Eliza らの論文 [10] では，センサーから複数のピクセルに関連する情報を取得し，複数のピクセルのサブセットから各ピクセルに関連する情報の変動を検出し，検出された変動を使用してセンサーの識別子を生成している．しかし，識別子を生成するのに時間がかかってしまうという問題がある。

Tusher Chakraborty らの論文 [4] では，センサーの識別と障害検出の現実的な 2 つの課題を解決するシステムを提案している．これはフォールカーブを用いて課題を解決している．センサーの識別の問題に関してはフォールカーブの 2 つの重要な特性である一意性とメーカー依存を利用することで解決している．しかし，メーカーの依存を利用していてこれではメーカーごとにプログラムをつくなければならないという課題がある．

## 3. 提案

1 章の課題で挙げた 1 つ目と 2 つ目の課題に対して，Sensor 内に設計されている Manufacture ID を取り出し，Manufacture ID を用いて Sensor の識別を行う．3 つ目の課題に対して，Server 上の Database で Library を管理して，Sensor が取り付けられたら Library を使えるように自動的にセットアップして，データ取得を行うことで解決する．

具体的な提案方法として，Sensor 上の Database を用いて Library を管理して，Sensor の Manufacture ID と Database 内に登録してある Manufacture ID を紐づけて Library をダウンロードし，データを取れる段階まで自動的にセットアップするシステムを提案する．全体構成図を図 1 に示す．

提案では，まず Sensor を取り付けてから IoT Device の Storage 内に取り付けた Sensor の Library があるか確認するように設計している．もし，取り付けた Sensor の Library が IoT Device の Storage 内にあった場合はそのまま Sensor Data を取得し，Sensor Data を表示させる．もし，取り付けた Sensor の Library が IoT Device の Storage 内になかった場合は Sensor の Manufacture ID を Sensor から取得して，Manufacture ID を Server に送り，Sensor 内で Manufacture ID と Sensor 名と Library を紐づけて Library を IoT Device の Storage 内に送り，Sensor Data を取得し，Sensor Data を表示させる．

Sensor の識別方法として，Sensor 内に設計されている Manufacture ID を変換して取り出して，Manufacture ID を Server に送り，送られてきた Manufacture ID を Server

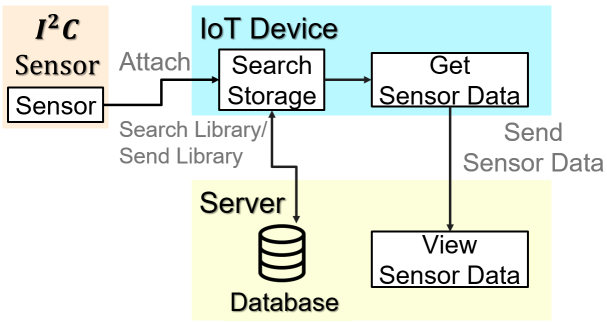


図 1　全体構成図

上の Database に保存されている Manufacture ID と紐づけて識別を行っている．

Sensor を取り付けてからデータを取得する工程まで自動化にして，ユーザがデータシートを読むことやコーディングをする手間を省くことができる設計にしている．

## 4.　実装と実験環境

この章では，提案するソフトウェアの実装と実験環境について述べる．

### 4.1　実装

図 2 にソフトウェア構成図を示す．

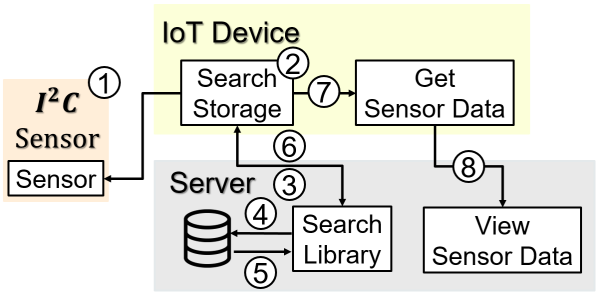


図 2　ソフトウェア構成図

図 2 に示されているソフトウェアの流れの説明は以下に示す。

1. Sensor を取り付ける
2. Sensor から Manufacture ID を取得し、Manufacture ID を用いて IoT Device の Storage 内に Sensor の Library があるかどうかを確認する。
3. IoT Device の Storage 内に Sensor の Library がなかった場合に Manufacture ID を Server に送る。
4. 送られてきた Manufacture ID を用いて、Manufacture ID に対応するセンサーを紐づける
5. 紐づけた情報を用いて Library を探す
6. Server から IoT Device に Library を送る
7. Sensor Data を取得する
8. 取得した Sensor Data を表示する

図 3 にソフトウェアの処理の流れを示す．

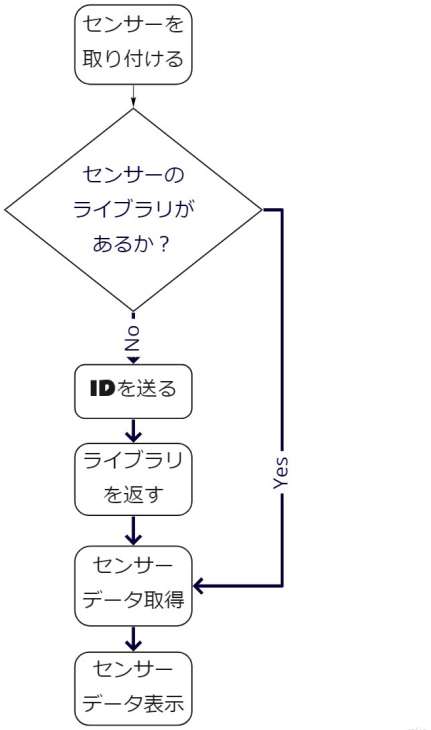


図 3　ソフトウェアの処理の流れ

ソフトウェアの流れとして，まず Sensor を取り付けるところからスタートし，IoT Device 内にその Sensor に対応した Library があるかどうかを確認する．Sensor に対応した Library があった場合，Sensor Data を取得して，Sensor Data を表示させる．Sensor に対応した Library がなかった場合，Server に Manufacture ID を送り，Server 内にある Database を検索して Library を返す．その Library を用いて Sensor Data を取得して，Sensor Data を表示させる．

図 4 に Server と IoT Device のプログラム役割図を示す．

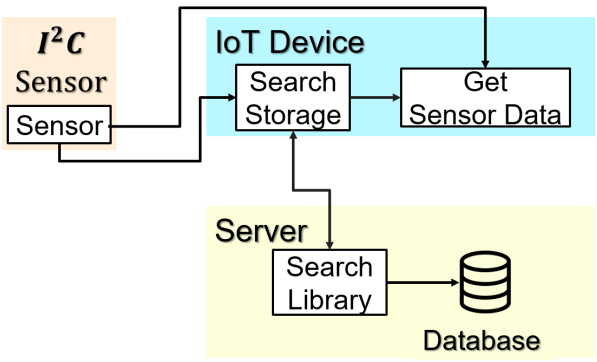


図 4　プログラム役割図

図 4 に示されているシステムそれぞれの説明を以下に示す．

- Search Storage Sensor から Manufacture ID を取得し，IoT Device の Storage 内に Manufacture ID に対応し

た Sensor の Library があるかないかを確認する．あった場合は，そのまま Sensor Data を取得する．なかった場合には，Sensor から取得した Manufacture ID を，urequest を使い POST 形式で Server 側の Library に Manufacture ID を送って Library を受け取る役割である．

- Get Sensor Data
  Server から返ってきた Library を使用し，Sensor Data を取得して表示する役割である．本研究では温度・湿度・気圧データを取得できる BME280 と温度データを取得できる ADT7410 からデータを取得するものとする．
- Search Library
  IoT Device の Search Storage から送られてきた Manufacture ID を用いて，データベース内を検索をして Manufacture ID と紐づけられるライブラリを取得し，それを IoT Device の Manufacture ID Management に返す役割である．
- Database
  本研究で使っている Database の構成を表 1 に示す．この Database は MongoDB で構築している．

表 1　Database 構成表

| センサー名 | ID | ライブラリ |
|---|---|---|
| BME280 | 0x60 | library |
| ADT7410 | 0x19 | library |
| ... | ... | ... |

### 4.2　実験環境

本研究の実験環境を以下に示す。

- IoT Device：ESP-WROOM-32
- Sensor Module：BME280/ADT7410
- Server：Ubuntu 18.04.5 LTS
- Database：MongoDB4.0.21

本研究の IoT Device として ESP-WEOOM-32 を使用した．Sensor Module は温度・湿度・気圧データを取れる BME280 と温度データを取れる ADT7410 を使用した．Server は Linux ディストリビューションの 1 つである Ubuntu に Web サーバソフトウェアの Apache2 を使用しており．その中で MongoDB を使用している．MoongoDB は NoSQL で様々な種類のデータを柔軟に格納でき，データ量や負荷の増加に対応できるため MongoDB を使用して実装を行う．

実験に使用した構成を図 5，図 6，図 7 に示す．実験で Sensor 数を 1 個から 32 個使用する．理由として，ピンの数が 32 個だったため最大数を 32 個とした．実験は Sensor を取り付けてから Senor Data を取得するまでの経過時間を計測，Sensor 内に設計されている Manufacture ID の取得にかかる時間を計測，Database 内にある Library を検索時間を計測する 3 つの実験を行う．実験には実験環境で示したものと同様のものを使用して検証を行う．また，既存方式と提案方式の比較を行う．

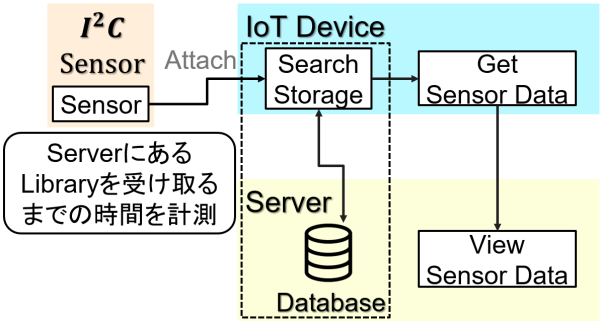


図 5　構成図 (データ取得時間)

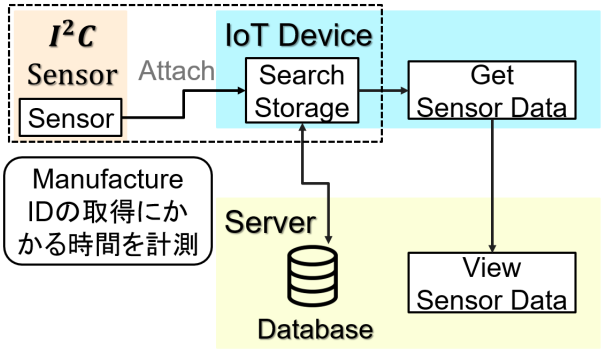


図 6　構成図 (Manufacture ID 取得時間)

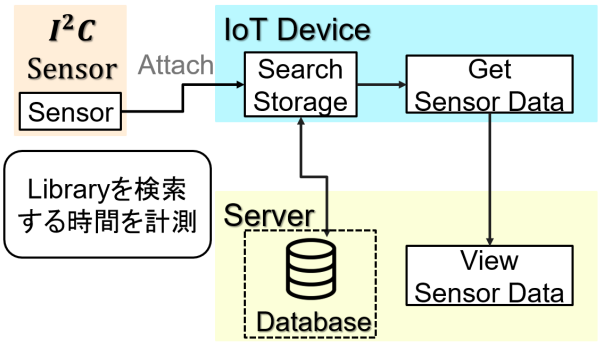


図 7　構成図 (検索時間)

## 5.　評価と分析

この章では本実験で得られた結果の評価および分析について述べる．

既存方式では回路を組んでからプログラムを作ってデータ取得という流れで手間がかかっていたが，提案方式では回路を組んでからデータ取得という流れでプログラムを作るという工程がなくなり手間を減らすことができる．既存

方式よりも提案方式の方がよいと考える．

次に図 7 に Sensor1 個を取り付けてからデータ取得にかかる時間のグラフを示す．図 8 に Sensor2 個を取り付けてからデータ取得にかかる時間のグラフを示す．グラフは X 軸が回数，Y 軸が時間 (msec) を示している．時間計測を 50 回ずつ繰り返し，Sensor1 個の時の平均時間は 427msec となった．また Sensor2 個の時の平均時間は 659msec となった．

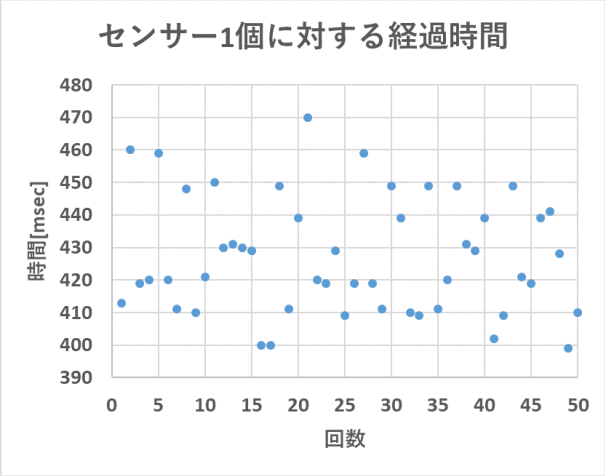


図 8　センサー 1 個にかかる検索時間

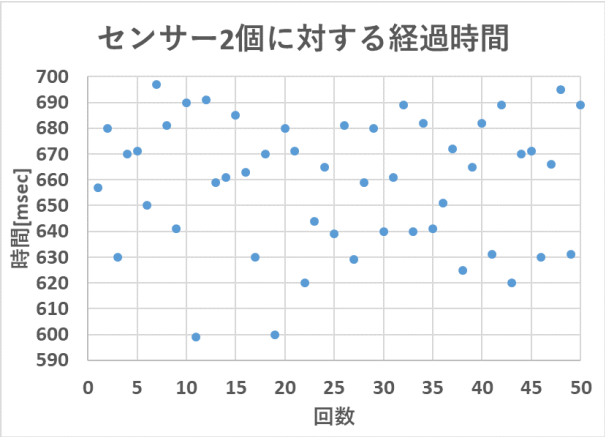


図 9　センサー 2 個にかかる検索時間

次に図 9 に Sensor 内に設計されている Manufacture ID の取得にかかる時間のグラフを示す．グラフは X 軸が Sensor 数，Y 軸が Sensor 内に設計されている Manufacture ID の取得にかかる経過時間を示している．Sensor1 個にかかる時間を 32 回計測した．1 個の場合の時間は 9msec かかって，32 個の場合の時間は 288msec になると予想していたが，実際は 240 秒となった．そのため予想よりも 16 %の時間を短縮することができた．

図 7 に関しては今後の行う予定とする．

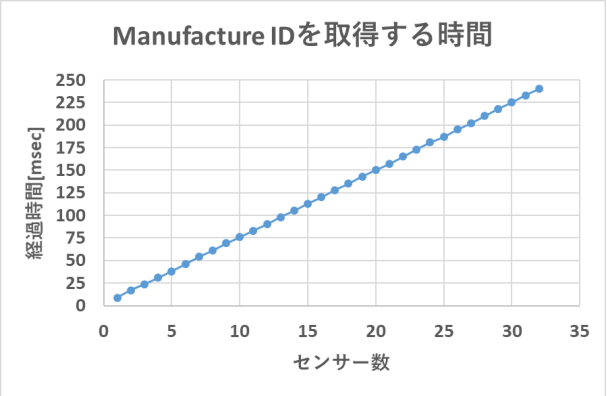


図 10　Manufacture ID の取得にかかる時間

## 6. 議論

この章では，提案・実験・実験環境、評価に関する議論について述べる．

本研究では，Server 上の Database を用いて Library を管理して，Sensor の Manufacture ID と Database 内に登録してある Manufacture ID を紐づけて Library をダウンロードし，データを取れる段階まで自動的にセットアップするシステムを提案した．しかし，課題も残っている．

まず，送られてくる Library が一行で返ってくることについてである．これは Database に保存するときに文字数の問題で 1 行でまとめることしかできなかったという点である。これは一行でも実行できるようにして改善できるようにする。

次に，回路を組んでからすぐに Manufacture ID を取得して，Manufacture ID を送り，Library が返ってくるという工程が手動でやらなければならないという点である．これは ESP32 が起動したときに実行されるプログラムに書き込むことで改善できるようにする．

今後はこの 2 点を改善できるように検討を行っていく．

## 7. おわりに

本研究では 3 つの課題を解決するために，Server 上の Database を用いて Library を管理して，Sensor の Manufacture ID と Database 内に登録してある Manufacture ID を紐づけて Library をダウンロードし，データを取れる段階まで自動的にセットアップするシステムを提案した．今後の課題として，本研究には Server にある Database にある Library の数が少ないので、今後増やしていきたい．また，Manufacture ID や Library 取得を自動化していきたい．本研究によって，Sensor の識別を行い，自動セットアップによってデータ取得までの手間を減少させ，IoT の発展に貢献できるのである．

## 参考文献

[1] Kotak, J. and Elovici, Y.: IoT Device Identification Using Deep Learning, *arXiv preprint arXiv:2002.11686* (2020).

[2] Zhang, J., Li, G., Marshall, A., Hu, A. and Hanzo, L.: A new frontier for IoT security emerging from three decades of key generation relying on wireless channels, *IEEE Access*, Vol. 8, pp. 138406–138446 (2020).

[3] Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.-R. and Tarkoma, S.: Iot sentinel: Automated device-type identification for security enforcement in iot, *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, pp. 2177–2184 (2017).

[4] Chakraborty, T., Nambi, A. U., Chandra, R., Sharma, R., Swaminathan, M. and Kapetanovic, Z.: Sensor Identification and Fault Detection in IoT Systems, *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, New York, NY, USA, Association for Computing Machinery, p. 375–376 (online), DOI: 10.1145/3274783.3275190 (2018).

[5] Aftab, H., Gilani, K., Lee, J., Nkenyereye, L., Jeong, S. and Song, J.: Analysis of identifiers in IoT platforms, *Digital Communications and Networks*, Vol. 6, No. 3, pp. 333–340 (2020).

[6] Mohammadi, M., Al-Fuqaha, A., Sorour, S. and Guizani, M.: Deep learning for IoT big data and streaming analytics: A survey, *IEEE Communications Surveys & Tutorials*, Vol. 20, No. 4, pp. 2923–2960 (2018).

[7] Kumari, R. S. S. and Gayathri, C.: Interfacing of MEMS motion sensor with FPGA using I2C protocol, *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pp. 1–5 (online), DOI: 10.1109/ICIIECS.2017.8275932 (2017).

[8] Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J. D., Ochoa, M., Tippenhauer, N. O. and Elovici, Y.: ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis, *Proceedings of the symposium on applied computing*, pp. 506–509 (2017).

[9] Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N. O., Guarnizo, J. D. and Elovici, Y.: Detection of Unauthorized IoT Devices Using Machine Learning Techniques (2017).

[10] Du, E. Y., Schneider, J. K. and Ganti, S.: Sensor identification (2015).

 