EasyConnect: A Management System for IoT Devices and Its Applications for Interactive Design and Art

Yi-Bing Lin, *Fellow, IEEE*, Yun-Wei Lin, Chang-Yen Chih, Tzu-Yi Li, Chia-Chun Tai, Yung-Ching Wang, Fuchun Joseph Lin, Hsien-Chung Kuo, Chih-Chieh Huang, and Su-Chu Hsu

***Abstract*—Many *Internet of Things* (IoT) technologies have been used in applications for money flow, logistics flow, people flow, interactive art design, and so on. To manage these increasing disparate devices and connectivity options, ETSI has specified end-to-end machine-to-machine (M2M) system architecture for IoT applications. Based on this architecture, we develop an IoT *EasyConnect* system to manage IoT devices. In our approach, an IoT device is characterized by its "features" (e.g., temperature, vibration, and display) that are manipulated by the network applications. If a network application handles the individual device features independently, then we can write a software module for each device feature, and the network application can be simply constructed by including these brick-like device feature modules. Based on the concept of device feature, brick-like software modules can provide simple and efficient mechanism to develop IoT device applications and interactions.**

***Index Terms*—Interactive design, Internet of Things (IoT), machine-to-machine (M2M), wearable device, wireless communications.**

I. Introduction

ACCORDING to a survey of Consumer Electronics Association [1], over 9% of American consumers used wearable devices for fitness and wellness. Vandrico Solutions showed that there are 233 items of wearable devices, and the average price is 373 USD [2]. In the recent years, many *Internet of Things* (IoT) devices (including the wearable devices) have been used in applications for money flow, logistics flow, people flow, interactive art design, and so on. To manage these increasing disparate devices and connectivity options, ETSI has specified end-to-end machine-to-machine (M2M) system architecture for IoT applications [3].

An IoT device can be characterized by its functionalities or "features." For the purpose of description, this paper defines a *feature* as a specific input or output "capability" of the IoT device. For example, a wearable ring with the temperature sensor has the input device feature (IDF) called "temperature" (abbreviated as T-IDF). A pair of wearable glasses with the optical head-mounted display has the output device feature (ODF) called "display" (abbreviated as D-IDF). An IoT device may be connected to the network (i.e., Internet) using wireless communications directly or indirectly through a smart phone. If so, the corresponding software called network application is developed and executed by a server in the network side, which receives or sends the messages from/to the IoT device. When the values of the IDFs are updated, the IoT device will inform the network application to take some actions, and the network application may send the result to the ODF of an IoT device. With this view, the IoT devices interact with each other through their features, and we say that the network application "maps" the IDFs to the ODFs.

Fig. 1(a) illustrates five IoT devices D1, D2, D3, D4, and D5, where the left-hand side of the figure illustrates the IDFs of the devices and the right-hand side of the figure illustrates the ODFs of the devices. The wearable ring D1 has three input device features H-IDF (humidity), T-IDF, and G-IDF (gravity). The smart phone D2 has four input device features T-IDF, G-IDF, M-IDF (microphone), and C-IDF (camera) illustrated in the left-hand side of Fig. 1(a), and three output device features: D-ODF, V-ODF (vibration), and S-ODF (speaker) illustrated in the right-hand side of Fig. 1(a). The wearable glasses D3 have two input device features M-IDF and C-IDF, and one output device feature D-ODF. The bulb D4 has an output device feature L-ODF (luminance). The tail D5 has one output device feature V-ODF (vibration). This special device is the Silver Medal Award artwork "transparent organ" in Salon International Des Invention [4]. The tail of this device wags based on the vibration strength received from its V-ODF. We will elaborate more on this device in Section IV-A.

Lines (1)–(10) in Fig. 1(a) illustrate how these IoT devices interact, where a line connecting an IDF to an ODF represents

Manuscript received January 30, 2015; revised March 19, 2015; accepted March 31, 2015. Date of publication April 15, 2015; date of current version November 18, 2015. This work was supported in part by Academia Sinica AS-102-TP-A06, in part by the ITRI/NCTU JRC Research Project, in part by the ICL/ITRI Project, in part by Microsoft 103Q733, in part by D-link 103C142, in part by MOST 104-2218-E-009-009, and in part by the MoE ATU plan.

Y.-B. Lin is with the Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan, and also with the Institute of Information Science, Academia Sinica, Taipei 11529, Taiwan (e-mail: liny@cs.nctu.edu.tw).

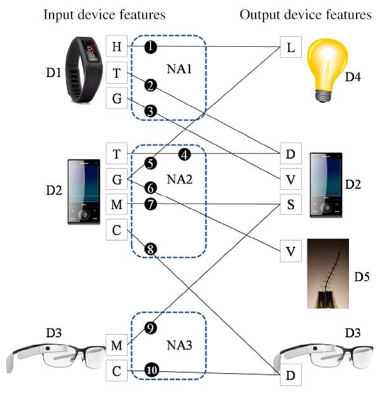
Y.-W. Lin, C.-Y. Chih, T.-Y. Li, C.-C. Tai, Y.-C. Wang, F.-J. Lin, and H.-C. Kuo are with the Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: jyneda@gmail.com; michael66230@gmail.com; alice1992224@gmail.com; darkgerm@gmail.com; yong.cing723@gmail.com; fuchun_lin@cs.nctu.edu.tw; kenny103178@hotmail.com).

C.-C. Huang is with the College of Architecture and Design, Chung Hua University, Hsinchu 30012, Taiwan (e-mail: scottie.c.c.huang@gmail.com).

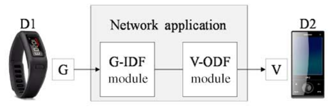
S.-C. Hsu is with the School of Film and New Media, Taipei National University of the Arts, Taipei 11201, Taiwan (e-mail: suchu@techart.tnua.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JIOT.2015.2423286



(a)



(b)

Fig. 1. IoT devices, device features, and the network applications. (a) Connections among D1, D2, D3, D4, and D5. (b) Software modules for G-IDF to V-ODF mapping in NA1.

interactions between the corresponding device features in input and output IoT devices. Such interactions are implemented in network applications. In Fig. 1(a), network application NA1 implements interactions (1)–(3) for D1, D2, and D4, NA2 implements interactions (4)–(8) for D2, D3, D4, and D5, and NA3 implements interactions (9) and (10) for D3 and D2. If a network application handles the individual device features independently, then we can write a software module for each device feature, and the network application can be simply constructed by including these brick-like DF modules. For example, the building blocks for Line (3) in Fig. 1(a) are shown in Fig. 1(b), where the network application NA1 handles G-IDF of D1 through the G-IDF module. This IDF module computes, e.g., the variation of acceleration, and passes the result to the V-ODF module. This ODF module translates the received value to vibration intensity. Then NA1 outputs this vibration intensity to drive the vibration mechanism (V-ODF) of D2.

If the IDF and the ODF modules are independent of each other, then these software modules can be reused to build the network applications, and effectively speed up the development of the IoT applications. Fig. 1(a) shows that different IoT devices may have similar IDFs/ODFs. For example, D1 and D2 have similar IDFs T-IDF and G-IDF. D2 and D3 have similar IDFs M-IDF and C-IDF, and a similar output device feature D-ODF. D2 and D5 have a similar output device feature V-ODF. Therefore, NAs 1–3 can reuse same software modules to implement the tasks for these similar DFs. In this paper, we show that based on the concept of device feature, brick-like software modules can provide efficient and simple mechanism to develop IoT device applications and communications. Specifically, we develop an IoT *EasyConnect* system to achieve this goal. This paper is organized as follows. Section II describes the EasyConnect architecture. Section III shows how to configure the connections of the IoT devices through EasyConnect.

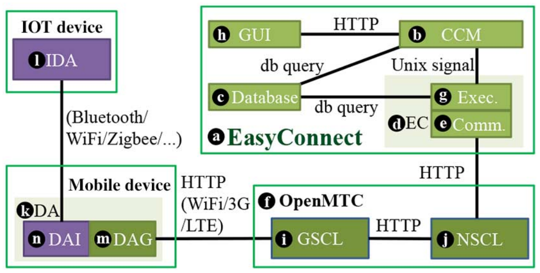


Fig. 2. EasyConnect architecture.

Section IV studies interactive design and art applications that can be easily implemented through EasyConnect. Section V concludes this work with future research directions.

II. EasyConnect

This section describes the IoT device feature management system EasyConnect and its security management. Before we elaborate on the details of EasyConnect, we first overview the previous studies.

A. Previous Work

Most IoT management platforms, such as Philips hue [5] and IoT.est [6], focus on home automations or sensor networks. These platforms have shown excellent performance as they claimed. Philips hue is a personal wireless lighting system which can only be controlled by a smart device (e.g., smartphone). A smartphone is used as the remote controller to manipulate the colors and luminance of the light bulbs. IoT.est is a test-driven IoT service creation environment, which tests the behaviors of IoT devices and services. As an example, consider a motion induction light device (an output IoT device) with a built-in control program. When this program receives a motion event from a motion sensor, it turns ON the light. To test this output IoT device, IoT.est simulates the motion detector by sending a motion event to the control program. Then, it examines the program execution flow, and checks whether the control program will turn ON the light or not. In this way, IoT.est verifies if the control program works correctly. Like Philips hue, IoT.est does not address how to connect multiple input IoT devices to multiple output IoT devices. Also, the functions of these platforms are not as modularized and reuseable as EasyConnect. EasyConnect is particular powerful for interactive design applications. To our knowledge, no general IoT platform has been designed for interactive design applications.

B. IoT Device Feature Management

EasyConnect [Fig. 2(a)] consists of four modules. The *creation, configuration, and management* [CCM; Fig. 2(b)] module systematically categorizes the features of the IoT devices, manages the functions to automatically configure connectivity of IDFs and ODFs, and stores all related information in the *database* module [i.e., a Structured Query Language (SQL) database; see Fig. 2(c)]. The *execution and communication*

[EC; Fig. 2(d)] module consists of two submodules. The *communication* submodule [Fig. 2(e)] interacts with a lower-layer M2M system (such as OpenMTC [7] and OM2M [8]) to retrieve/deliver the IDF/ODF information from/to the IoT devices. In this paper, the IoT devices are physically connected to OpenMTC [Fig. 2(f)], and then transparently communicate with each other as well as the network applications through the EC. The *execution* submodule [Fig. 2(g)] is responsible for execution of network applications for the connected IDFs and ODFs. The GUI [Fig. 2(h)] provides a friendly user interface to quickly establish the connections and meaningful interactions among the IoT devices. Through this GUI, a user instructs the CCM to execute desired tasks through HTTP-based REST APIs to create or set up device features, mapping functions, and connection configurations. Through Unix signals, the CCM instructs the EC to carry out interactions between linked IDFs and ODFs in the preset configuration of IoT devices.

EasyConnect interacts with OpenMTC following ETSI 102 690 case 2 [9]. Both EasyConnect and OpenMTC are installed in a desktop or a notebook. OpenMTC implements the gateway service capability layer [GSCL; Fig. 2(i)] and the network service capability layer [NSCL; Fig. 2(j)] to provide M2M communications. The EC is considered as a network application of OpenMTC. To establish the interaction between EasyConnect and OpenMTC, the EC should register to OpenMTC, and subscribe application resources of the GSCL. The device application [DA; Fig. 2(k)] is installed in a mobile device (e.g., smart phone). The IoT device application [IDA; Fig. 2(l)] connects to OpenMTC indirectly through the DA. In the smart phone, the DA consists of two software components, which are: 1) the device application to the gateway [DAG; Fig. 2(m)] communicates with the GSCL for IDA registration and data exchanges and 2) the device application to IoT device [DAI; Fig. 2(n)] communicates with the IoT device following the message format specified by the IDA (typically a string delivered through Bluetooth). Before an IoT device can be manipulated by EasyConnect, it must register to OpenMTC.

EasyConnect manages IoT devices with scalability and flexibility, where every IoT device is identified by its *device name* and *device model*. For example, the smart phone in Fig. 1(a) has the device name D2, and its device model is HTC one E8. By considering a device model as a set of device features, EasyConnect effectively manages these device features. Since different IoT devices of a device model may involve in a connection configuration, device names are needed to distinguish these same-model devices. The database [Fig. 2(c)] stores IoT device models and the related details in several tables. The *device feature* (DF) table collects possible device features and defines their parameters in the *device feature parameter* (DF-parameter) table. For the IoT devices considered in Fig. 1, the corresponding DF table is illustrated in Table I(a). Every entry in the DF table has two fields, which are: 1) the DF-type field indicates that the device feature is input or output and 2) the #PARM field indicates the number of parameters. Consider the gravity sensor (G-IDF) entry in Table I(a). G-IDF is an IDF that measures three-axis acceleration and produces three acceleration values. Therefore, #PARM=3, and in the DF-parameter table, the G-IDF record lists the three-axis accelerations (x_1, x_2, x_3).

TABLE I
DEVICE FEATURE TABLE

Device feature		
DF-name	**DF-type**	**#PARM**
G (gravity)	Input	3
H (humidity)	Input	1
T (temperature)	Input	1
M (microphone)	Input	1
C (camera)	Input	1
L (luminance)	Output	1
V (vibration)	Output	1
D (display)	Output	1
S (speaker)	Output	1

(a)

Device model	
DM-name	**DM-type**
GARMIN ring	W (wearable)
HTC one E8	P (smartphone)
Google glass	W (wearable)
Bulb Model 1	O (other)
Tail	O (other)

(b)

TABLE II
DEVICE MODEL TO DEVICE FEATURE (DM-DF) TABLE

DM-DF		DM-DF	
DM-name	**DF-name**	**DM-name**	**DF-name**
GARMIN ring	H-IDF	HTC One E8	V-ODF
GARMIN ring	T-IDF	HTC One E8	S-ODF
GARMIN ring	G-IDF	Google glass	M-IDF
HTC one E8	T-IDF	Google glass	C-IDF
HTC one E8	G-IDF	Google glass	D-ODF
HTC one E8	M-IDF	Bulb Model 1	L-ODF
HTC one E8	C-IDF	Tail	V-ODF
HTC one E8	D-ODF		

We note that EasyConnect accumulates possible device features in the DF table and standardizes their parameter formats in the DF-parameter table. In this way, any new IoT device model with device features already existed in the DF table can easily join and share software modules in EasyConnect.

The device models are stored in the *device model* (DM) table. Table I(b) illustrates the DM table for IoT devices D1–D5 in Fig. 1. Every entry in the table has two fields. 1) The DM-name field stores the name of a device model (e.g., Google glass). 2) The DM-type field indicates the type of the device model, which can be "P" (phones or smart phones), "W" (wearable devices), or "O" (other IoT devices).

The device features of a device model are grouped in the *device model to device feature* (DM–DF) table. Table II illustrates the DM–DF table for the IoT devices in Fig. 1. For example, the smart phone HTC one E8 has seven entries in Table II: T-IDF, G-IDF, M-IDF, C-IDF, D-ODF, V-ODF, and S-ODF.

The *device* table records the actual IoT devices which have registered to OpenMTC and can be manipulated in EasyConnect. Table III illustrates the device table for D1–D5 in Fig. 1. An entry in this table has the following fields: the device name, the device model name, the MAC addresses of the IoT device and the connected smart phone, and the presence status. When an IoT device registers to OpenMTC, its presence status is set to "online." If the IoT device is detached from OpenMTC, then its status is "offline."

C. Security Management

Authentication of EasyConnect is considered in two levels. The first level security addresses device authentication between the device and OpenMTC. OpenMTC uses public key infrastructure (PKI) for security management. A certificate

TABLE III
DEVICE TABLE

Device				
D-name	DM-name	IoT MAC-addr	Phone MAC-addr	status
D1	NCTU ring	00:EE:BD:D4:1D:08	50:2E:5C:D4:C7:32	Online
D2	HTC one E8	50:2E:5C:D4:C7:32	50:2E:5C:D4:C7:32	Online
D3	Google glass	01:3C:BC:D2:1D:EE	50:2E:5C:D4:C7:32	Offline
D4	Bulb Model 1	01:EE:BB:A4:CE:08	50:2E:5C:D4:C7:32	Online
D5	Tail	68:EE:66:D4:1B:18	50:2E:5C:D4:C7:32	Offline

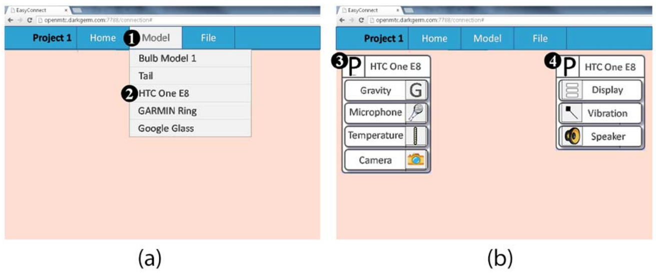


Fig. 3. Project page. (a) The "Model" pull-down menu. (b) Selection of HTC one E8.

authority (CA) is utilized initially to distribute public and private keys, which allows OpenMTC and the device to mutually authenticate each other and agree on a symmetric key. Then, this key is used for secure communication between the device and OpenMTC.

The second level security addresses user authentication when a user logs in to EasyConnect. EasyConnect security is built based on Django [10] which is a high-level Python web framework. Django provides an authentication mechanism to associate an incoming request with the user identification and session identification. It is guaranteed that the EasyConnect manipulations are performed by the authenticated users. Encryption for EasyConnect follows existing protocols for, e.g., 4 G (between the smartphone and OpenMTC) and Bluetooth (between the smartphone and the IoT devices).

For the interactive design applications, EasyConnect is more likely to be operated in a semiopen or a close environment, and therefore, the above security management is appropriate.

III. Configuring the Connections for the IoT Devices

This section shows how to dynamically connect IoT devices D1, D2, and D4 for an interactive design. EasyConnect provides a web-based project page (Fig. 3) for a user to connect the device models. The connection configuration of the device models is then saved as a project (Project 1 in our example). When configuring the connection, actual devices need not exist. When the user executes the project (activates the connection), all IoT devices of the connected device models must register to OpenMTC. In Section III-A, we describe how to select the IoT device models. Then, Section III-B shows how to connect these IoT devices for execution.

A. Device Selection

The project page has a pull-down menu "model" that allows selection of device models. When this menu is pulled down

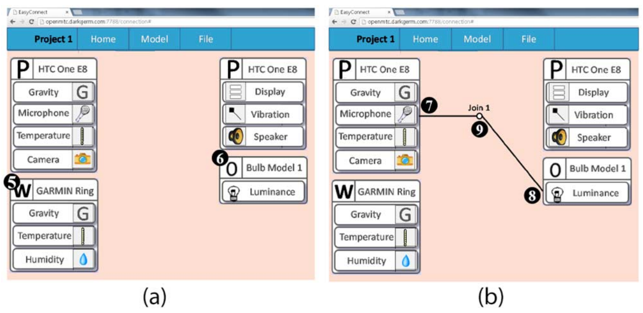


Fig. 4. Project page. (a) To add a ring and a bulb. (b) To link M-IDF of the smart phone to L-ODF of the bulb.

[Fig. 3(1)], the CCM [Fig. 2(b)] sends the SQL command to the database [Fig. 2(c)]:

```
SELECT DM-name FROM "Device Model."
```

The database returns all device model names stored in the DM table, and these device model names are shown in the project page. When the user selects the HTC one E8 item [Fig. 3(2)], the CCM sends the following SQL command to the database:

```
SELECT DM-type FROM "Device Model" WHERE
    DM-name = "HTC one E8."
```

The database obtains the DM-type of HTC one E8 from the DM table and returns "P" (which means phone device) to the CCM. The CCM then asks the database to retrieve all IDFs of HTC one E8 by executing the following SQL command involving the DF, DM–DF, and DM tables:

```
SELECT DF-name FROM "Device Feature"
    LEFT JOIN DM-DF USING(DF-name)
    LEFT JOIN "Device Model" USING(DM-name)
    WHERE DM-name = "HTC one E8" AND DF-type =
    "input."
```

The database returns the input device features G-IDF, M-IDF, T-IDF, and C-IDF to the CCM. Similarly, the CCM queries the database to obtain the ODFs. The CCM then instructs the GUI to plot the HTC one E8 of category "P" with the device features in two icons, which are: 1) the icon in the left-hand side of the project page illustrates the IDFs [Fig. 3(3)] and 2) the icon in the right-hand side illustrates the ODFs [Fig. 3(4)]. The user continues to select GARMIN ring and bulb Model 1 from the "model" menu. The icon of the GARMIN ring of category "W" is shown below the HTC one E8 icon in the left-hand side of the window [Fig. 4(5)]. Since the ring does not have any ODFs, no icon for the ring is shown in the right-hand side of the project window. The icon of the bulb of category "O" is shown below the HTC one E8 icon in the right-hand side of the window [Fig. 4(6)], and no icon is shown in the left-hand side of the project window.

B. Device Feature Connection

The device features are connected in EasyConnect through the mapping functions in IDF and ODF modules. An IDF or an ODF module [Fig. 1(b)] is a Python program that can be quickly created, modified, and executed without compilation. We have implemented several default functions that can be used to develop IDF/ODF modules. Specifically, EasyConnect provides several *device feature functions* (DF functions) for an IDF with DF-parameters $x_1, x_2, \ldots, x_m$.

1) The scalar function returns the scalar value of the DF-parameter vector. That is,

$$\text{Scalar}\,(x_1, x_2, \ldots, x_m) = \sqrt{x_1^2 + x_2^2 + \cdots + x_m^2}.$$

2) The sum function returns the sum of the DF-parameters.
3) The max/min functions return the maximum/minimum values of the DF-parameters.
4) The ith identity function returns the the ith DF-parameter value.
5) EasyConnect also provides functions for every DF-parameter x_i.
6) The normalization function $\text{Norm}(x_i)$ normalizes x_i in the range [0, 1]. Specifically,

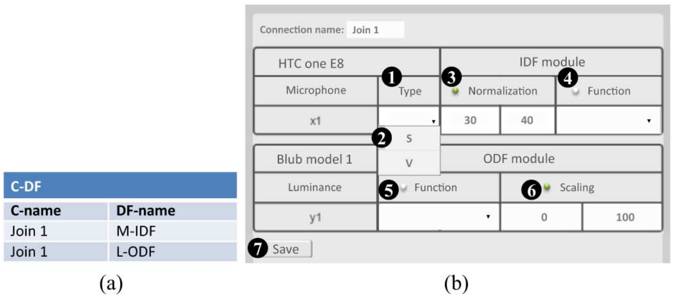
$$\text{Norm}\,(x_i) = \frac{x_i - a}{b - a}$$

where a and b are the minimal and the maximal values of x_i. EasyConnect provides default values for a and b which can be modified by the user. This function is typically used in the IDF module.
7) The scaling function scales (linearly or quadratically) the input to a value in the range [*c, d*]. This function is typically used in the ODF module.
8) A *DF-parameter* function is a k-input function where the inputs are the most recent k samples of a DF-parameter x_i sent from (received by) the IoT device if x_i is an IDF (ODF). A DF-parameter function can be any of the aforementioned DF functions where k DF-parameters are replaced by the most recent k samples of a DF-parameter x_i. A DF-parameter function is "temporal" because it handles the histogram samples of a DF-parameter. On the other hand, a DF function is "spatial" because it handles the values for various DF-parameters of a device feature at the same time.

IDFs from multiple IoT devices may connect to an ODF. We provide the *join* functions that allow joining the IDFs of multiple IoT devices to affect ODFs. These join functions can be the same as DF functions except that the DF-parameter inputs are replaced by IDF inputs from different IoT devices. Consider the project window in Fig. 4(b). By clicking the M-IDF icon [Fig. 4(7)] and the L-ODF icon [Fig. 4(8)], the GUI creates a line [Fig. 4(9)] to link the icons of these two device features. This line is automatically assigned a default name "Join 1." ("Join k" represents the kth connection created in the project; this name can be modified by the user later). The CCM issues the following SQL commands to access the *connection-device feature* (C-DF) table in the database.

```
INSERT INTO "C-DF" ("C-name," "DF-name") VALUES
  ("Join 1," "M-IDF");
INSERT INTO "C-DF" ("C-name," "DF-name") VALUES
  ("Join 1," "L-ODF");
```

In the command, C-name (the connection name) is Join 1, and the DF-names (device feature names) corresponding to this connection are M-IDF and L-ODF, respectively. The database then creates two entries in the C-DF table [see Fig. 5(a)]. This table describes the connection relationship among the device features in the project.



C-DF	
C-name	DF-name
Join 1	M-IDF
Join 1	L-ODF

Connection name: Join 1

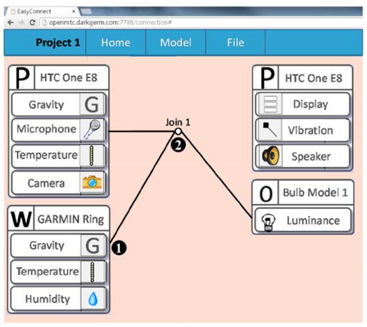
HTC one E8	IDF module			
Microphone	❶ Type	❸ ● Normalization		❹ Function
x1	▾ (❷ S / V)	30	40	▾
Blub model 1	ODF module			
Luminance	❺ Function	❻ ● Scaling		
y1	▾	0	100	

❼ Save

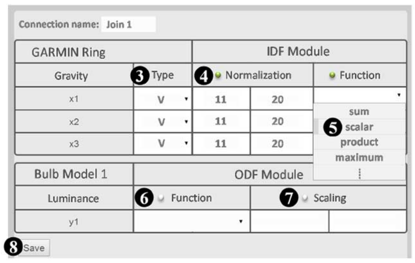
Fig. 5. (a) Connection to device feature (C-DF) table. (b) Workspace for the connection from M-IDF (Microphone) to L-ODF (Luminance).

Then, the database accesses the DF-parameter table to obtain the DF-parameters for both M-IDF and L-ODF and return them to the CCM. The GUI pops up a workspace window to set up the functions in the M-IDF module (HTC one E8) and the L-ODF module [bulb Model 1; see Fig. 5(b)]. In this window, the pull-down menu "type" [Fig. 5(1)] allows the user to specify the type of the IDF-parameter x_1. Let $x_1(j)$ be the jth x_1 sample of M-IDF. When EasyConnect receives the Jth sample $x_1(J)$, "type" of parameter x_1 can be "S" [i.e., the sample value $x_1(J)$] or "V" [i.e., the variation of the two samples $x_1(J) - x_1(J-1)$]. In our example, the user clicks "S" [Fig. 5(2)], and x_1 is the volume received from the microphone. In the IDF module area of the workspace window, two radio buttons are used to enable normalization [Fig. 5(3)] and other functions [Fig. 5(4)]. In this example, the user clicks the button to enable the normalization function where x_1 ranging from 30 to 40 will be normalized in [0,1]. In the ODF module, there are two radio buttons for selecting the functions [Fig. 5(5)] and scaling [Fig. 5(6)], respectively. In our example, the scaling function is triggered where the output range is [0,100]. In this connection configuration, the volume of the microphone is mapped to the luminance of the bulb. This configuration (Project 1) is saved into the database after the user clicks the save button [Fig. 5(7)]. When D2 and D4 in Fig. 1 are attached to EasyConnect, they can interact through the above connection configuration.

We may extend Project 1 to join multiple input IoT devices to affect an output IoT device. Consider a game called "voice or vibration" where player 1 makes loud voice to the microphone of a smart phone (D2 in Fig. 1), and player 2 makes big vibration with a wearable ring (D1 in Fig. 1). If the voice dominates, then a lamp (D4 in Fig. 1) becomes brighter. Otherwise (the vibration dominates), the lamp becomes darker. To implement this game in EasyConnect, M-IDF of D2 is used as player 1's input, V-IDF of D1 is used as player 2's input. L-ODF of D4 is the output of this game. "Does voice dominate vibration" is a join function ($z_1 > z_2$?) that compares the normalized M-IDF value (z_1) with the normalized V-IDF value (z_2). If $z_1 > z_2$, then the result $w = 1$. Otherwise ($z_1 < z_2$), the result $w = 0$. In the L-ODF module, the function Increment(w) is selected, which increments the luminance of the bulb if $w = 1$, and

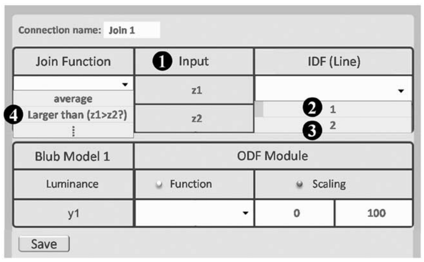


(a)

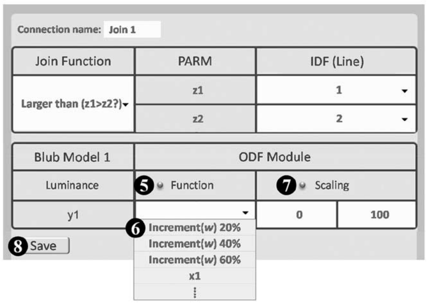


(b)

Fig. 6. Adding a new link. (a) Connecting G-IDF and Join 1. (b) The workspace window for selecting IDF and ODF functions.



(a)



(b)

Fig. 7. Join function workspace. (a) Selecting the join function. (b) Selecting a function in the ODF module.

decrements the luminance if $w = 0$. To implement this game, we extend the "Join 1" link in Fig. 4(b) by adding another line segment as follows: the user clicks the G-IDF icon of GARMIN ring [Fig. 6(1)] and the circle of Join 1 [Fig. 6(2)] to create a new line segment. In response to this linking action, the CCM issues the following SQL command to the database.

INSERT INTO "C-DF" ("C-name," "DF-name") VALUES ("Join 1," "G-IDF").

The database adds a new entry in the C-DF table [this third entry is not shown in Fig. 5(a)]. The GUI then pops up a workspace window to set up the functions for the G-IDF module (GARMIN Ring) and the L-ODF modules [see Fig. 6(b)]. For G-IDF, "V" type is selected for $\boldsymbol{x}_1, \boldsymbol{x}_2,$ and $\boldsymbol{x}_3$ [Fig. 6(3)] to compute the variation of gravity. The DF-parameter ranges of the normalization function [Fig. 6(4)] are set from 11 to 20. Then, the scalar function [Fig. 6(5)] is selected to transfer the vector ($\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$) to its scalar value. Therefore, the G-IDF module produces the normalized variation of gravity. All functions in the ODF module [Fig. 6(6) and (7)] are disabled since the setup of the join function workspace will overwrite previous ODF setups (to be elaborated later). The save button [Fig. 6(8)] is clicked to save this link configuration into the database.

The user clicks the circle of Join 1 [Fig. 6(2)], and a workspace window for selecting the join function is popped up (see Fig. 7) in the project window. The inputs $z_1, z_2, \ldots, z_m$ of a join function are the IDFs linked to the circle through the line segments (and therefore the number of inputs is the number of the line segments). In our example, the first line segment is for M-IDF of D2 and the second line segment is for G-ODF of D1. In the join function workspace window, the "input" [Fig. 7(1)] allows the user to rearrange the order of the parameters. In our example, z_1 is set to Line 1 [M-IDF; Fig.7(2)] and z_2 is set to Line 2 [G-IDF; Fig. 7(3)]. The user clicks the join function to select "larger than ($z_1 > z_2$?)" [Fig. 7(4)]. The function returns 1 if $z_1 > z_2$ is truth and returns 0 if $z_1 > z_2$ is false. The result of the join function w is the input of a function in the L-ODF module [Fig. 7(5)]. The increment (w) 20% function [Fig. 7(6)] is selected, which increases 20% luminance of the bulb if $w = 1$ or decreases 20% if $w = 0$. The result of the increment function is then scaled to a value in [0,100] [Fig. 7(7)]. Finally, the user clicks the save button [Fig. 7(8)] to store the configuration into the database. We note that when any function of the ODF module in the join function workspace is selected, all selected functions of the ODF modules in Figs. 5(b) and 6(b) are disabled.

IV. Examples of IoT Device Connection for Interactive Design and Art

Section III-B describes how connection of device features is achieved by selecting the functions in IDF and ODF modules. This section describes several functions developed in EasyConnect and shows how these functions are used in the IDF and the ODF modules to implement some interesting IoT device applications for interactive design and art. For the artworks described in this section, the original implementations hardwired IDFs with ODFs. In other words, the IDFs and ODFs of these artworks are bound together through nonmodule programs. To develop a new artwork which has the same IDFs or ODFs of an existing artwork, the IDFs and ODFs still need to be rebuilt again since those device features are not reusable. On the other hand, in EasyConnect, a new artwork can be simply constructed by changing the links between IDF and ODF modules to reuse the existing device feature modules. In other words, EasyConnect provides quick linkage and relinkage

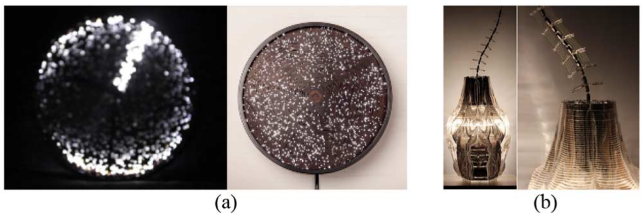


Fig. 8. Artwork examples. (a) Clock inside out (by courtesy of Kuo). (b) Tail of transparent organ.

between IDFs and ODFs. Based on our experiences, development of new artworks can be twice or three times faster (e.g., the development time complexity can be reduced from 4 weeks to 1 week).

A. Direct Connection and Normalization-Scaling

A very simple relationship between an IDF and an ODF is "direct connection" where EasyConnect simply forwards whatever values received from the IDF to the ODF, and none of the functions in the IDF and the ODF modules are triggered. Direct connection is typically used when an input device is linked to a display (D-ODF). In EasyConnect, every IDF is automatically mapped to an appropriate display ODF (for monitoring purpose). If the values of an IDF are texts (e.g., temperature reading), then the IDF can be directly connected to the display of a mobile phone through the short message service (SMS) [11]. In this case, the output IoT device name is a phone number, and EasyConnect sends the short messages targeted at this phone number.

An interesting direct connection example is the artwork "clock inside out," where an optic fiber clock (Fig. 8) is designed to signify the unfolding journey of temporal and spatial transformation in a 2014 exhibition "in my tummy, in my time." The curator of the exhibition stated [12]: when the audience encounters the "clock inside out," they have fallen into a zone different from the external world. Though the hour hand and the minute hand move accordingly with time, the second hand moves based on a person's heartbeat. The question for the person to ponder is: during our presence in the exhibition "in my tummy, in my time," how has our sense of existence been changed? How have our physical sensations and experiences been shaped in this peculiar zone? To implement this artwork in EasyConnect, the optic fiber clock is an output IoT device with the following features: H_S-ODF (second hand), H_M-ODF (minute hand), and H_H-ODF (hour hand). The input devices include a real watch with features W_H-IDF (watch hour) and W_M-IDF (watch minute), and a health ring with the IDF H_B-IDF (heartbeat). EasyConnect simply connects W_H-IDF to H_H-ODF, W_M-IDF to H_M-ODF, and H_B-IDF to H_S-ODF. No functions in the IDF modules (of the watch and the health ring) and the ODF module (of the optic fiber clock) need to be triggered.

This artwork can also be easily extended with the normalization-scaling mapping. In such a connection, the value of an IDF is first normalized to a value in the range [0, 1] in the IDF module, and then scaled to another value in the range [*c, d*] in the ODF module. The optic fiber clock has optic-fiber-related output features such as F_L-ODF (brightness of the fiber; equivalent to L-ODF) and F_C-ODF (color of the fiber). Many IDFs defined in EasyConnect can interestingly and entertainingly control the optic-fiber ODFs through normalization-scaling. For example, T-IDF of D1 in Fig. 1(a) can be connected to F_C-ODF or F_L-ODF of the optic fiber clock, which shows how temperature affects the color and brightness of the clock.

The sound-to-light application described in Section III-B is implemented with normalization-scaling mapping. Another good example is "transparent organ" [4]. This artwork attempted to bring new characteristics in creation of objects through biologically inspired computing and bionic design. According to the basic rules of symmetry and recursion, transparent organ presents the most natural growth "forms" by self-reorganization and synergy. In this way, fabrication of artificial nature is demonstrated by simulating growth forms and behavior with adaptation. Based on three-dimensional (3-D) rotation and reflection symmetry, digital fabrication techniques were used to generate the organic forms through the movement of the whisker-like sculpture; i.e., the "tail" [D5 in Fig. 1; see also Fig. 8(b)]. The input IoT devices of EasyConnect can provide rich and colorful interactions with the tail of transparent organ. In the design of transparent organ, the tail is an output IoT device that swings in three directions, and its ODF is vibration (V-ODF) with three parameters $(y_1,\ y_2, y_3)$ representing movement in three dimensions.

Many IDFs in EasyConnect can interact with the tail. To use, e.g., the gravity sensor of wearable ring D1 in Fig. 1 to drive the tail, EasyConnect connects G-IDF (with parameters $x_1,\ x_2,$ and x_3) to V-ODF. In D1's IDF module, we trigger the Norm function to normalize $x_i (i = 1,\ 2,\ 3)$. In the tail's ODF module, for every parameter y_i, the identity function is selected, where the normalized x_i value is mapped to y_i. Then, the scaling function is triggered to scale the y_i value to drive the vibration of the tail in the ith dimension. With simple normalization scaling mapping in EasyConnect, the tail easily mimics the movement of a person wearing the D1 ring.

B. Multistage Switches

Multistage switches are popular functions used to map IDFs to ODFs. An n-stage switch function receives an IDF value x ranging from *a* to *b*, and returns the value $y = [n \times \mathrm{Norm}(x)]$. Clearly, this equation is a special case of normalization-scaling where $\mathrm{Norm}(x)$ is executed in the IDF module and *y* is an output of the scaling function with the integer range [0, $n - 1$] in the ODF module. For $n = 2$, the scaling function is an ON–OFF (two-stage) switch often used in home automation applications. Consider a scenario that uses a smart phone to control the switches of a lamp and a fan. In our solution, a remote controller (e.g., an app in a smart phone) registers to EasyConnect with two buttons (B-IDFs), one for the lamp and another for the fan. The lamp and the fan are plugged in wireless switches that turn ON and OFF of the power. Both wireless switches are considered as output IoT devices connecting to OpenMTC through, e.g., Bluetooth, and register to EasyConnect with the switches (S_W-ODFs). The remote controller's B-IDFs are then connected to these S_W-ODFs by the two-stage switch function with values 0 (OFF) and 1 (ON). This application triggers the Norm function in the B-IDF module of the remote controller. In each of the S_W-ODF modules for the bulb and the fan, the scaling function with the integer range [0, 1] is triggered.

Consider another application that controls the fan speed by the temperature. We first register, e.g., D1 in Fig. 1(a) to EasyConnect with T-IDF and a fan with S_{pd}-ODF (speed). Suppose that the fan is controlled by the S_{pd}-ODF with four speeds: OFF, low, medium, and high. When T-IDF of D1 is linked to S_{pd}-ODF of the fan, a four-stage switch function is selected, which returns value 0 (OFF) if the temperature is below 25 °C, value 1 (low) if the temperature is below 27 °C and above 25 °C, value 2 (medium) if the temperature is below 29 °C and above 27 °C, value 3 (high) if the temperature is above 29 °C. In the T-IDF module of D1, the normalization function is triggered, where the temperature ranges from 23 °C to 31 °C. (if the actual temperature is below 23 °C, then the T-IDF value is 23 °C. Similarly, if the temperature is above 31 °C, the T-IDF value is 31 °C.) In the S_{pd}-ODF module of the fan, the scaling function is triggered with the integer range [0, 3].

C. Minimum and Maximum

The min/max functions return the minimum/maximum values of the same-type IDFs from different IoT devices. There are several variations of this function; e.g., the function may return the identity for the IoT device (e.g., the device name) with the minimum/maximum IDF value. An example using the min function is given below. Following the concept of Buddhist mediation in the Dharma Drum Buddhist College, Huang *et al.* attempted to create a physical and psychological empties representing zero, to generate energy by concentrating attention, and to reach the essence of Zen by integrating nature and the environment [13]. Specifically, they built an interactive meditation garden including Zen farm and Zen fountain systems. Through the concept of ambient intelligence [14], the heartbeat detection technology and water media are used to assist people to exercise "meditating like water and watering deep." In the Zen fountain application, people hold "calm stones" that are actually heartbeat sensors [Fig. 9(a)]. The application assumes that the smaller the heart rate variability (HRV), the more the person enters the Zen mode (more peaceful and calm). Note that the HRV can be easily obtained by selecting the "V" type in the H-IDF module. The participants stand around a fountain that is spraying water [Fig. 9(b)], and the participants' HRVs are automatically measured through the calm stones [Fig. 9(c)] when they exercise Zen. The fountain water will spray toward the position of the person with the minimum HRV value. This is the so-called "meditating on water". The min function (that returns calm stone identity) is used to implement the Zen fountain. Suppose that n persons participate in the Zen fountain exercise. Then, n calm stones register to EasyConnect with the input device feature H-IDF (heartbeat). On the other hand, the fountain is an IoT device with an ODF called position (P-ODF) that receives the position value (from 1 to n) and sprays the water to that position. People are required to stand at the positions from 1 to n around the fountain, which correspond to the calm stone

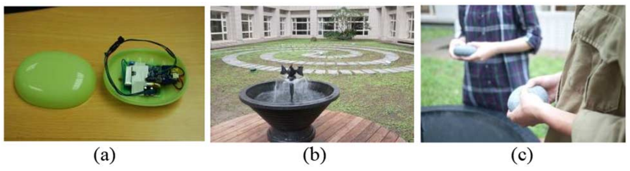


Fig. 9. Zen Fountain. (a) Calm stone. (b) Fountain for meditation on water. (c) Participants' heart rate variations are automatically measured through the calm stones.

Fig. 10. One million heartbeats. (a) Heartbeat collection mechanism. (b) Twin fetuses struggling in the world of the mother's womb.

identities. After the connection configuration of the IoT devices (calm stones and the fountain) are activated in EasyConnect, the min function is selected to receive the HRV values from these n H-IDFs and produce the position (the identity) of the calm stone with the minimum HRV value. The result is sent to P-ODF of the fountain. In this application, the minimum HRV value is calculated for every 15 s.

D. Other Functions

The sum function accumulates the same-type IDF values from several IoT devices at the same time (a DF function) or the values of a single IDF over a period of time (a DF-parameter function). An application for this function is "one million heartbeats" [15]. This application shows twin fetuses struggling in the "world" of the mother's womb (Fig. 10). One million heartbeats were collected from the participants, and the behaviors of participants will determine the characteristics of the babies and the experience of the mother. This artwork explores collective social behavior. In the original design, the heartbeats were collected in public [Fig. 10(a)]. With EasyConnect, the heartbeats can be more effectively and privately collected through personal health rings anytime and anywhere, and the sum function is used to produce the result to drive the display in Fig. 10(b).

The sum function has many variations. For example, the counter function is a special case of the sum function where the value of an IDF is either 1 (to count up), 0 (no action) or 1 (to count down). As another variantion, the average function divides the result of the sum function by the number of accumulated IDF samples. This function is used in the public interactive art "Zen Farm" described in Section IV-C. In this application, n participants wear heartbeat rings to measure their HRV values. The output device is a water flow controller (F-ODF) connected to a plant irrigation system. Through the average function, EasyConnect maps the average HRV value to F-ODF to drive the water flow of the plant irrigation system (Fig. 11). If the participants enter more into the Zen mode, more water will be sent to irrigate the plants.

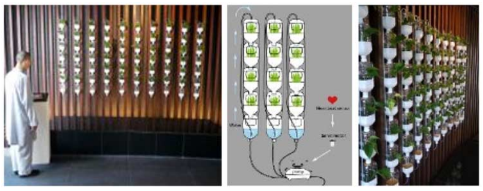


Fig. 11. Zen Farm: plant irrigation.

Fig. 12. Music-based applications. (a) Kinetic rain moving sculpture artwork in the Singapore Changi Airport. (b) Man–machine music interaction.

Another category of interesting EasyConnect applications is for music. We are considering pianos or other music instruments as device features, and trying to link, e.g., piano melody to motion artworks. In this way, EasyConnect can more efficiently and flexibly create interactive art designs such as kinetic rain moving sculpture artwork in the Singapore Changi Airport [Fig. 12(a)]. On the other hand, music expressing various emotions may define fear, anger, sadness, joy, disgust, trust, anticipation, and surprise. These emotions can be the values for M-IDF [music or microphone in Fig. 1(a)]. Then, M-IDF can connect to the color (F_C-ODF) and the brightness (F_L-ODF) and affect the light presentation of, e.g., the optic fiber clock in Section IV-A. Also, music performance of a musician can interact with a music playing machine through M-IDF and V-ODF [see Fig. 12(b)] [16].

V. Conclusion

This paper described the design and implementation of EasyConnect, a device feature management system that effectively and flexibly links the IoT devices. In our approach, an IoT device is characterized by its "features" (e.g., temperature, vibration, and display) that are manipulated by the network applications. If a network application handles individual device features independently, then we can write a software module for each device feature, and the network application can be conveniently constructed by including these brick-like device feature modules. By considering a device model as a set of device features, EasyConnect effectively classifies these device features, and manipulates them through the corresponding functions. In EasyConnect, the functions are used to develop device feature modules, which are easily created through Python script language, and can be executed immediately without compilation.

We described how to attach IoT devices to EasyConnect, and how these attached devices can be easily linked to perform interesting interactions through a friendly GUI executed in a smart phone, a tablet, a notebook, or a desktop. EasyConnect is especially powerful in creating interactive artworks and designs. We showed how home automation applications and artworks can be implemented through EasyConnect. It is particular interesting to see how multiple input IoT devices can join to affect the behaviors of output IoT devices. In the future, we will create more interesting functions for the device feature module software. By provisioning connection flexibility, EasyConnect does not slow down the interaction between the IoT devices. Since the art examples described in Section IV involve less than 100 devices with no or little background traffic in the communication paths of these devices, the EasyConnect approach has the same time-complexity performance as the existing "hardwire" designs (i.e., the messages are delivered in real time).

For a small-scale application, OpenMTC is utilized well as the platform but is not used to exhibit its networking strength. The OpenMTC's role will be essential when EasyConnect is expanded to accommodate generic M2M devices of any kinds and any scales [17], and to apply big data analytics to extract useful information.

As compared with the previously proposed approaches, the advantages of our approach are: 1) the IoT functions are more easily modularized and reuseable and 2) multiple input features can be flexibly and conveniently combined to affect (i.e., control) the ODFs.

This paper provided appropriate IoT security managements for interactive design. In the future, we will consider security issues for other IoT applications demanding a higher level of security. For example, according to the ETSI M2M specifications, keys can be derived hierarchically from the root key to a connection key then to an application key to ensure a maximum level of security.

References

[1] K. Tillmann, *Understanding the Connected Health and Wellness Market*. Las Vegas, NV, USA: Consumer Electron. Assoc., 2014.

[2] Vandrico Solutions, *Wearables Market Insights: Q2.14*. North Vancouver, BC, Canada: Vandrico Inc., 2014.

[3] F. J. Lin, Y. Ren, and E. Cerritos, "A feasibility study on developing IoT/M2M applications over ETSI M2M architecture," in *Proc. Int. Workshop Internet Things Technol.*, Seoul, Korea, Dec. 2013, pp. 558–563.

[4] S. Huang, "Transparent organ," in *Salon International Des Invention*, Geneva, Switzerland, 2014.

[5] Philips hue. (2014). "Meet hue," [Online]. Available: http://www.developers.meethue.com/

[6] S. De, F. Carrez, E. Reetz, R. Tonjes, and W. Wang, "Test-enabled architecture for IoT service creation and provisioning," *Future Internet Lect. Notes Comput. Sci.*, vol. 7858, pp. 233–245, 2013.

[7] S. Wahle, T. Magedanz, and F. Schulze, "The OpenMTC framework—M2M solutions for smart cities and the Internet of Things," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw.*, Jun. 2012, pp. 1–3.

[8] M. B. Alaya, Y. Banouar, T. Monteil, C. Chassot, and K. Drira, "OM2M: Extensible ETSI-compliant M2M service platform with self-configuration capability," *Procedia Comput. Sci.*, vol. 32, pp. 1–1186, 2014.

[9] *Machine-to-Machine Communications (M2M): Functional Architecture V2.1.1*, ETSI TS 102 690, Oct. 2013.

[10] Django. (2014). "The web framework for perfectionists with deadlines," [Online]. Available: https://www.djangoproject.com/

[11] Y. B. Lin and A. C. Pang, *Wireless and Mobile All-IP Networks*. Hoboken, NJ, USA: Wiley, 2005.

[12] W. S. Lai and C. H. Kuo, private communication, 2014.

[13] Y. H. Huang, S. C. Hsu, S. C. Lin, and C. H. Tsai, "ZEN–Construction of interactive meditation garden for ease and mindfulness," in *Proc. Int. Conf. Innovation Design*, 2012, pp. 1–6.

[14] P. Mace, “Other perspective on ambient intelligence,” *Password Mag.*, vol. 23, no. 5, p. 13, 2005.
[15] S. C. Hsu, J. Y. Lin, Y. C. Chen, J. S. Lin, and K. H. Chang, “One million heartbeats,” in *Proc. ACM Int. Conf. Multimedia*, 2007, pp. 365–366.
[16] H. H. Huang, W. H. Li, Y. J. Chen, and C. C. Wen, “Automatic violin player,” in *Proc. World Congr. Intell. Control Autom.*, 2012, pp. 3892–3897.
[17] L. A. Grieco, M. Ben Alaya, T. Monteil, and K. Drira, “Architecting information centric ETSI-M2M systems,” in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PERCOM Workshops)*, Mar. 24–28, 2014, pp. 211–214.

Yi-Bing Lin (M’96–SM’96–F’03) received the Bachelor’s degree from National Cheng Kung University, Tainan City, Taiwan, in 1983, and the Ph.D. degree from the University of Washington, Washington, DC, USA, in 1990.

From 1990 to 1995, he was a Research Scientist with Bellcore (Telcordia), Piscataway, NJ, USA. He then joined the National Chiao Tung University (NCTU), Hsinchu City, Taiwan, where he remains. In 2010, he became a Lifetime Chair Professor of NCTU, and in 2011, the Vice President of NCTU. In 2014, he became a Deputy Minister with the Ministry of Science and Technology, Taipei, Taiwan. He is also an Adjunct Research Fellow with the Institute of Information Science, Taipei, Taiwan, and also with Research Center for Information Technology Innovation, and a member of Board of Directors with the Chunghwa Telecom, Taipei City, Taiwan. He authored *Wireless and Mobile Network Architecture* (Wiley, 2001), *Wireless and Mobile All-IP Networks* (Wiley, 2005), and *Charging for Mobile All-IP Telecommunications* (Wiley, 2008).

Dr. Lin serves on the Editorial Board of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He has been a General or Program Chair for prestigious conferences including ACM MobiCom 2002. He has been a Guest Editor for several journals including the IEEE TRANSACTIONS ON COMPUTERS. He is on the Advisory Board or the Review Board of various government organizations including the Ministry of Economic Affairs, Ministry of Education, Ministry of Transportation and Communications, and National Science Council. He is a Fellow of the AAAS, ACM, and IET. He was the recipient of numerous research awards including the 2005 NSC Distinguished Researcher, the 2006 Academic Award of Ministry of Education, the 2008 Award for Outstanding Contributions in Science and Technology, Executive Yuen, the 2011 National Chair Award, and the TWAS Prize in Engineering Sciences, 2011 (The Academy of Sciences for the Developing World).

Yun-Wei Lin received the B.S. degree in computer and information science from Aletheia University, Taipei, Taiwan, in 2003, and the M.S. and Ph.D. degrees in computer science and information engineering from National Chung Cheng University, Chiayi, Taiwan, in 2005 and 2011, respectively.

His research interests include mobile *ad hoc* networks, wireless sensor networks, vehicular *ad hoc* networks, and M2M communications.

Chang-Yen Chih was born in Hualien, Taiwan, in 1992. He received the B.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2014, and is currently working toward the Master’s degree at the Institute of Computer Science and Engineering, National Chiao Tung University.

His research interests include software-defined networks, FreeBSD, Vim, and SCP foundation.

Tzu-Yi Li received the B.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2014, and is currently working toward the Master’s degree at the Institute of Network Engineering, National Chiao Tung University.

Her research interests include IoT platforms and applications.

Chia-Chun Tai received the B.S. degree in computer science from the National Chiao Tung University, Hsinchu, Taiwan, in 2014, and is currently working toward the Master’s degree at the Institute of Network Engineering, National Chiao Tung University.

His research interests include IoT platforms and applications.

Yung-Ching Wang was born in Penghu, Taiwan, in 1992. He received the B.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2014, and is currently working toward the Master’s degree at the Institute of Computer Science and Engineering from National Chiao Tung University.

His research interests include IoT platforms and applications.

Fuchun Joseph Lin received the B.S. and M.S. degrees in computer science from National Chiao Tung University, Hsinchu, Taiwan, and the Ph.D. degree in computer and information science from The Ohio State University, Columbus, OH, USA.

He is a Professor with the Department of Computer Science and Associate Chief Director with Microelectronics and Information Systems Research Center, National Chiao Tung University (NCTU). Before joining NCTU in August 2012, he was a Chief Scientist with Applied Research of Telcordia Technologies, Piscataway, NJ, USA, where he focused on M2M communications and next-generation mobile networks. He was with Telcordia Technologies, Piscataway, NJ, USA, for 20 years and AT&T Bell Laboratories, Murray Hill, NJ, USA, for 4 years. He has authored more than 50 journals and conference papers. He has filed more than ten patents.

Dr. Lin was also active in SDOs such as IEEE, 3GPP, and ATIS and contributed to the formation of NGSON, ISB, and NGN architecture and specifications.

Hsien-Chung Kuo received the B.S. degree in computer science from the National Chiao Tung University, Hsinchu, Taiwan, in 2013, and is currently working toward the Master’s degree at the Institute of Computer Science and Engineering, National Chiao Tung University.

His research interests include IoT/M2M platforms and applications.

(Scottie) Chih-Chieh Huang received the Ph.D. degree in architecture from the National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan, in 2011.

He is an Assistant Professor and a Taiwanese Media Artist with Chung Hua University, Hsinchu, Taiwan. He is the Director of the Biologically Inspired Objects (BIO) Laboratory, Hsinchu, Taiwan, and the Innovation and Creativity Center (ICC), Chung Hua University, Hsinchu, Taiwan. His work has been exhibited nationally and internationally including Salone Satellite, ISEA Exhibition, iF Design Exhibition, Red Dot Design Exhibition, Avignon Off Festival, Holland Animation Film Festival, ACM SIGGRAPH Art Gallery, as well as the Prague Quadrennial. His research and works have been featured in many international academic publications, such as ACM, IEEE, Springer, MIT Press, IGI Global, as well as Sandu Cultural Media. His research interests include the use of interactive media for developing digital art, kinetic sculptures, and futuristic products.

Prof. Huang currently serves on the Digital Art Committee of ACM SIGGRAPH.

Su-Chu Hsu is the Director of the Center for Art and Technology, Taipei National University of the Arts (TNUA), Taipei, Taiwan, where she is a member of the faculty. She created and built the Graduate School of Art and Technology and was the first Chair. She is known for her work in net art, digital art, digital creative life and learning. She is a leading proponent for the integration of art and technology in Taiwan.