Developing Real-Time Face Identification Device Composable with Distributed Applications

Kosuke Hirayama[1](✉), Sachio Saiki[1], and Masahide Nakamura[1,2]

[1] Graduate School of System Informatics, Kobe University, 1-1 Rokkodai, Nada, Kobe, Japan
hirayama@ws.cs.kobe-u.ac.jp, sachio@carp.kobe-u.ac.jp, masa-n@cs.kobe-u.ac.jp

[2] Riken AIP, 1-4-1 Nihon-bashi, Chuo-ku, Tokyo 103-0027, Japan

Abstract. In recent years, many applications which using face identification are developed. However, development load in creation of an application program using the face identification is high because of the tight coupling of each functions and lack of sharability of imaging device. To address this problem, in this research, we propose a new device named "face identification sensor box" which can realize loose coupling of function and high sharability. The proposed device uses cognitive API for face identification and notifies the result to application with Publish/Subscribe messaging model. Using this architecture, face identification results can be shared among with applications and design of application becomes easier.

Keywords: Face identification · Cognitive API · Pub/Sub · IoT · Edge computing

1 Introduction

For the rapid improvement of image processing and AI technology, many applications start to incorporate face identify function, which identifies a person from images obtained from image sensors, by using computer vision. Face identification is a method of finding and extracting a face and calculating positions of face parts, which can be a characteristic part when identifying a person, like eyes or a nose from extracted faces as feature points, and identifying persons by comparing feature points with that of other people registered in advance. In our life, face identification is often used for user authentication, for example, iPhone X, a smartphone released by Apple Inc., has Face ID function which uses face identification to unlock their lock and complete online payment [4]. Furthermore, Universal Studio Japan, a theme park in Japan, have incorporated face identification in the identity confirmation process which is needed when annual pass holders enter the park to improve work efficiency related to identity confirmations and prevent abuses [7]. In addition to user authentication, various usages are emerging. For example, developing a personal talking robot which is capable of comprehending the target user and generate proper subjects for them by providing face identification on the robot [3], and making a system which can prevent dementia patients' going out and wandering without permission [5], and so on.

Face identify function offers novel added values to applications, but on the other hand, when implementing these function to applications, some problems such as high development load due to low reusability of the function, and lack of sharability of resources such as cameras and devices, often occur. To address these problems, in this research, we propose a new device named *Face identification sensor box* which can offer a face identification service which various applications can use. The sensor boxes keep taking photos by a connected camera and trying to detect faces. Then, when a face is detected, the sensor box does face identification and identifies whose face is this. The face identification process is executed by using external cognitive service. After the face identification completed, the sensor boxes notify the result of the identification to subscribed external applications with Pub/Sub messaging model. By using the Face identification sensor box, multiple applications that need face identification in the same space can use the results at their desired timings by using a single sensor box. Also, since we can delegate face identification process to the sensor box, developers can reduce the working load in designing applications, and they may be able to focus on the essence of the services they want to offer. Moreover, we can reduce dependency within the application and expect improvement of scalability. As an example of a set of service which can be realized by using the Face identification sensor box, for instance, in an office, put the device on an entrance, and identify faces when someone pass there, then create attendance management and entering or leaving room record, after that, unlock the computer of the corresponding user's and give information about their schedule and reminder. In this paper, to realize the Face identification sensor box, we design a functional requirement and confirm the effectiveness of the proposed method by a prototype implementation. We have checked that we can offer face identification service to external applications which require these service by using the prototype.

2 Preliminaries

2.1 Pub/Sub Type Messaging

Pub/Sub (Publish/Subscribe) [9] is one of the messaging models used for asynchronous exchanging messages between multiple devices and services. We show a conceptual diagram of Pub/Sub in Fig. 1. In Pub/Sub messaging model, sender clients creating messages and sending them are called *Publisher*, and recipient clients receiving messages are called *Subscriber*. Messages sent by Publishers are stored in the destinations called Topic. On the other hand, Subscribers register (subscribes) to topics in which necessary messages are sent. When a message stored in a topic, this message is sent (delivered) to Subscribers of this topic. By using this messaging model, both Publishers and Subscribers can send and receive messages with no dependency on each other's state. Basically, Publishers

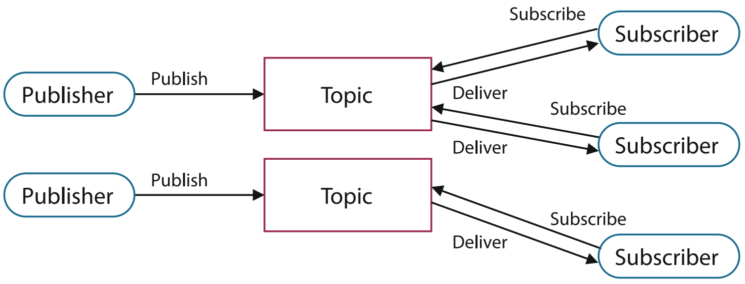


Fig. 1. A conceptual diagram of Pub/Sub

can send messages sequentially regardless of Subscribers' status, and Subscriber can receive only necessary messages. As a result, the performance and scalability of applications can be improved.

2.2 MQTT

MQTT (Message Queue Telemetry Transport) [6] is a protocol to realize asynchronous communication between two endpoints with Pub/Sub messaging model. Since MQTT is an asynchronous communication protocol, MQTT communication can be stable regardless of network reliability, and since MQTT is a lightweight protocol, many IoT (Internet of Things) devices with not high-performance use MQTT.

MQTT defines two entities in the network: Broker and Client. Client corresponds to Publishers and Subscribers in Pub/Sub. Broker works as a Client's mediator and manages the destinations of messages for each of topics. In actual connection, Publishers send messages to Subscribers through Broker. When sending messages, Publishers set a topic of each message. Broker sends the message to Subscribers only which subscribe to the message's topic. In this way, MQTT makes able to Pub/Sub type messaging.

2.3 Cognitive API

With the rapid technological growth of cloud computing, now we can use cloud services to process various tasks which are previously executed on edge sides. Under such a kind of background, various services utilizing artificial intelligence, which becomes explosively popular in recent years, are now getting available as a cloud service. One of the actual cases of cloud service using artificial intelligence is cognitive services. Cognitive services extract useful information from visual/sound/linguistic/knowledge data by analyzing them and enable computers to recognize these types of information. Using cognitive services can realize functions about cognition which are previously difficult for computers. The cognitive service based on images can extract various information from images, and a lot of public clouds provide this kind of services. In general, cloud services are available from outside as APIs. Likewise, users of cognitive services can get various information by sending images to the API. With cognitive services, for example, it is able to analyze human face images and estimate the age, gender, and emotional value of the face. It is also possible to recognize various items such as appliances, furniture, foods, and colors from images. Furthermore, It is able to output tags about the place and weather from the background of an image.

3 Proposed Method

In this chapter, we provide a detailed explanation for the face identification device mentioned in Sect. 1, which various applications can use.

3.1 Problems with Existing Face Identification Application

Face identification function offers novel added values to applications. Currently, many companies such as NEC [8] and Panasonic [2] are offering face identification solutions. However, when implementing these function to applications, developers have to implement services, photo taking and face identification functions for each application. Therefore, each function has tight coupling with applications, and as a result, the reusability of these function may deteriorate. For example, we cannot reuse photo taking and face identification functions although these are a general function. In addition, Occupying one camera by one application makes another problem. In the case where multiple face identification functions required within the same space, although the object person is in the same place, many cameras will be required.

3.2 Face Identification Sensor Box

Based on the problems of existing applications, in this research, we propose Face identification sensor box. The Face identification sensor box is a device to provide some functions to share cameras and face identification services. The sensor box has to be able to provide the same face identification function to multiple applications. In addition, considering when there are multiple sensor boxes, it is required that we can configure the face identification and add or delete users for each device all at once. Then, we make the sensor boxes delegate face identification processes to an outside service, and execute and manage the face identification function in there, to reduce the load of managing sensor boxes. The sensor box also has to be able to deal with many pollings at the same timing, assuming the sensor box is called by multiple applications. And in order to improve the scalability, the sensor box should be able to continue to notify the results of identification. Then, we adopt a connection framework which realizes asynchronous communication.

We show a conceptual diagram of the whole architecture of the sensor box in Fig. 2. “Face identification process” is the component which represents APIs for face identification process with the sensor box, such as Pub/Sub connection

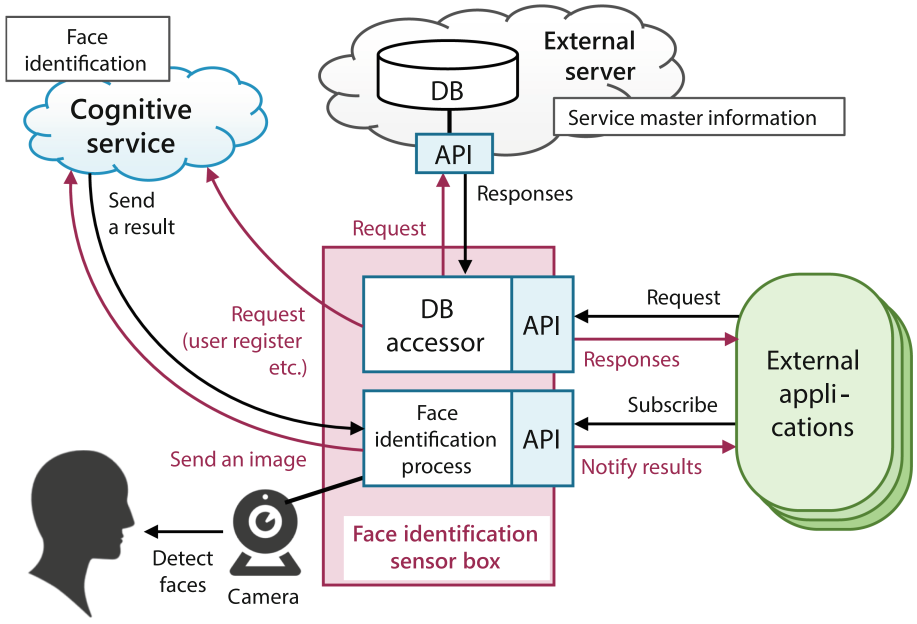


Fig. 2. A conceptual diagram of the whole architecture

(described later) or using a cognitive service. "External server" shown on the sensor box is for building the database which uniquely stores user information and so on separately from the cognitive service. "DB accessor" is used for exchanging data between the database on the external server and external applications.

3.3 Functional Requirement

The face identification with the sensor box has these function:

Service Exposure with Pub/Sub. When applications need face identification function, applications have to register itself to the sensor box. The sensor box stores subscribed applications. After the face identification process, when the sensor box obtains the result, the sensor box notifies the result to all subscribed applications. After that, applications can process their tasks with received result. We adopt Pub/Sub in the connection between the sensor box and applications and make it asynchronous.

In addition to the function to subscribe, we equip the sensor box with a function to unsubscribe for when applications no longer need the results.

Photo Taking and Face Detection at Fixed Intervals. The sensor box keeps getting images from the connected camera. Before the face identification, as a preparation, the sensor box executes face detection locally because it is inefficient to execute face identification for all obtained images. In the face detection process, *whether an image contains a face is only judged.* Thereby, the sensor box can execute face identification only for images containing a face.

Face Identification with Cognitive API. The sensor box executes face identification with face images obtained on the previous section. We use a cognitive API for the face identification to delegate face identification processes to the outside. When the face identification process finished, the sensor box receives the result.

By the way, to execute face identification with cognitive API, we have to register the name and face images of object people and train the machine learning model used for identification beforehand. We can do these operations by accessing endpoints of cognitive API, but, to make applications can do it easily, we have to create wrapper APIs on the sensor box. Also, when we add a new user on cognitive API, some problem may occur on the API side, for example, the API does not accept non-latin characters for user's name, or the API cannot register a detailed profile of users such as a birthday. Therefore, we build a database with a unique data format on an external server. When we add a new user of identification, the sensor box sends only the necessary information to the cognitive API and registers to the API, and, at the same time, sends full information to the external server's database and make it stored also in there. Furthermore, in order to enable applications to use the information stored in the database, we have to prepare ways to refer the database via the sensor box as APIs. To prepare such necessities, we make DB accessor which is a group of APIs to use the database. In addition to user information, we put various setting information such as API key and endpoint to use a cognitive API in the external server and make the external server provide this information as a service master, to enhance the convenience when accessing from each sensor box. All user information is stored in Face API and the external server. As a result, the sensor box itself does not store any user information.

The above is the functions necessary for the Face identification sensor box.

4 Implementing the Prototype

4.1 Implementation

Based on the proposed method, we implement a prototype of the Face identification sensor box.

We use the following technologies.

- Development languages: Python 3.5, HTML5, JavaScript, PHP 5.3
- Database: MySQL 5.7

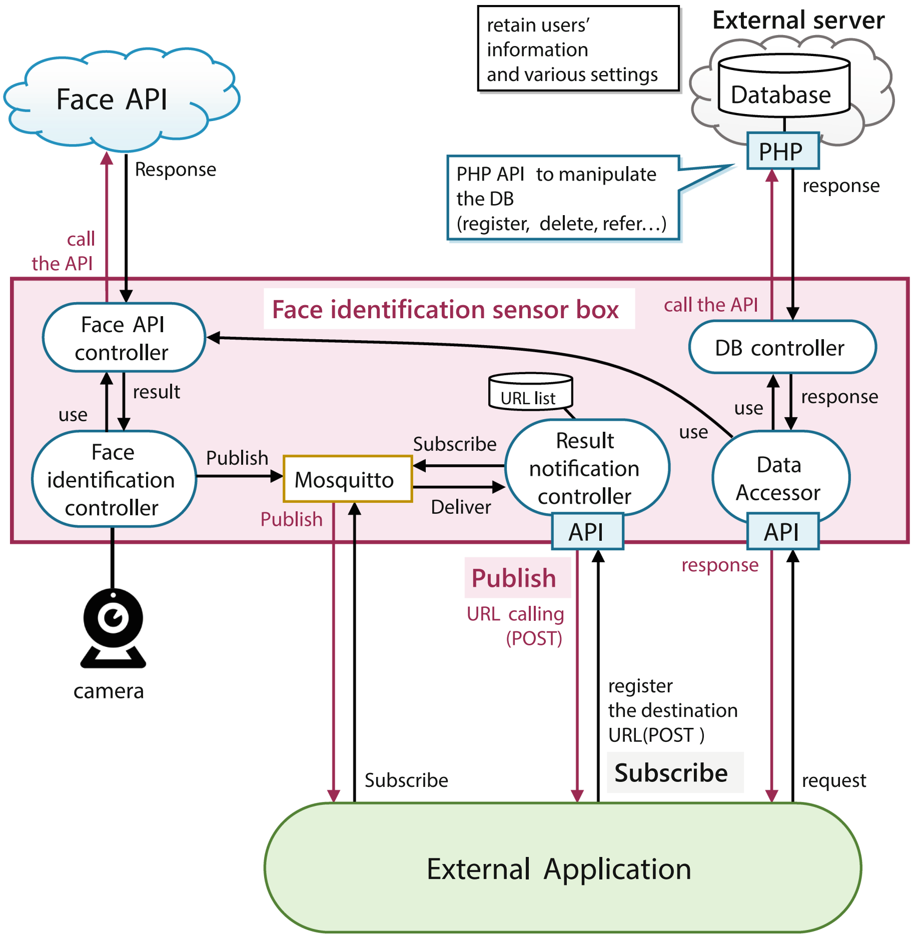


Fig. 3. The overall implementation

– Libraries: Paho MQTT[1], Flask[2], Flask-SocketIO[3], OpenCV[4]
– MQTT broker: Mosquitto[5]
– Cognitive API: Microsoft Face API [1]

Paho MQTT, Mosquitto is an open source implementation of MQTT protocol. Paho MQTT serves as Publisher and Subscriber, and Mosquitto serves as Broker. Flask is one of the web application frameworks. We use Flask to make each

[1] https://www.eclipse.org/paho/.
[2] http://flask.pocoo.org/.
[3] https://flask-socketio.readthedocs.io/en/latest/.
[4] https://opencv.org/.
[5] https://mosquitto.org/.

function WebAPIs. Flask-SocketIO is a library which enables WebSocket connection with Socket.IO on Flask applications. We use Flask-SocketIO to connect with a browser. We use Raspberry Pi 3[6] (OS : Raspbian 9.4), which is a small single board computer, as the prototype edge device.

4.2 Notification Flow of Identification Results

For the prototype, we make the part which notifies results of face identification Pub side, and we make external applications Sub side, then Pub/Sub communication is performed between both side. Inside of the sensor box, the Face identification controller notifies the results to Mosquitto, and Mosquitto assigns them to the specific topic, and the Result notification controller which subscribes this topic receives the results of face identification asynchronously. The Result notification controller works as Publisher to external applications and notifies the identification results to URLs of subscribed external applications. By these process, the Result notification controller works as Broker simulatively.

Actually, the Face identification sensor box also supports Pub/Sub by using MQTT. Mosquitto can be Publisher and send the identification results to external applications. However, we do not assume that MQTT used from applications directly for this time, and we do not use this function.

4.3 Preparation

For the implementation of the prototype, as cognitive API, we use Microsoft Face API (hereinafter called "Face API") which is one of the services of Microsoft Azure. Face API can detect faces, identify faces, analyze emotions, and so on. We can use these functions easily via the cloud. To use face identification function of Face API, firstly we have to make a group (=PersonGroup) and add users (=Person) who are object people of identification. On the registration, an ID (=personId) used by Face API is assigned to each user. By selecting the Person registered, uploading one's face images, training of the learning model of the PersonGroup enables face identification.

We can complete these operations by sending requests including various parameters to the correspondent Face API endpoints. However, to simplify this, we make the Face API controller which is a set of utilities including various operations to Face API.

And we build a database to store users' information and various setting values on an external MySQL server, and we make PHP APIs on the server so that the sensor box can access the database. The database keeps information of users who use face identification in the form [uId, personId, name, reading, birthday, gender, nickname, description]. uId is the unique Id used in the Face identification sensor box, and personId is the Id used in Face API. We make applications which subscribe to the sensor box to use uId mainly. Regarding the

[6] https://www.raspberrypi.org/.

use of the database, to simplify it, we make the DB controller which is a set of utilities to get, write, modify, delete, search data and convert id mutually.

Furthermore, with the Face API controller and the DB controller, we implement the Data accessor as a WebAPI to use these controllers and use functions to register and delete users and so on from external applications. In addition to user registration and deletion, the Data accessor has some functions such as to convert between uId and personId, and to confirm and acquire the existence of user information corresponding to each Id, and applications can get information from the database as needed. The reason why we avoid making each controller API directly is that, when registering new entries, after registering to Face API, we have to store it also in the database, but by gathering these operations together with the controllers, we can register necessary information collectively by POST. As a result, we can simplify the implementation of applications. Also, we can prevent inconsistency between registered information of Face API and of the database.

We show the overall implementation of face identification sensor box described above in Fig. 3.

4.4 Details About the Operation

In this section, we explain the functions of the prototype while associating functional requirements mentioned in Sect. 3. Note that we use URL call for Pub/Sub connection of the prototype.

Service Exposure with Pub/Sub. Applications which need to use the Face identification sensor box have to register their destination URL to the endpoint of the Result notification controller. To complete this operation, applications have to connect to the endpoint such as `(DeviceURL):(PORT):/reg_app_url` with POST request. After the registration, when face identification is executed, the sensor box notifies the result to all registered URLs. Applications can cancel their registration in the same way at any time.

Photo Taking at Fixed Intervals and Face Detection. The sensor boxes keep getting images by a connected camera and detecting a face. For face detection, we use OpenCV which is an image processing library and use Haar Cascade [10] classifier with preliminarily prepared classifier file. When a face is detected in the image, the face identification process starts.

In the face detection, the sensor box does not execute face identification immediately when a face is detected. Instead, we introduce Face counter to stable face identification. We show the role and behavior of Face counter in Fig. 4. The Face counter increases when a face is detected in the image, and the sensor box considers the face detection completed only after Face counter exceeds a certain threshold. Once the face is detected, the Face counter decreases while the face detection does not succeed, and until the value of Face counter becomes 0, the sensor box does not detect the next face. With the Face counter,

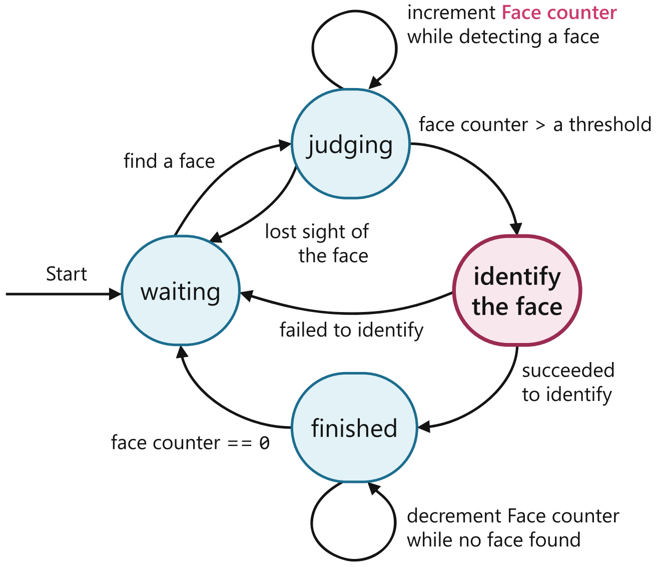
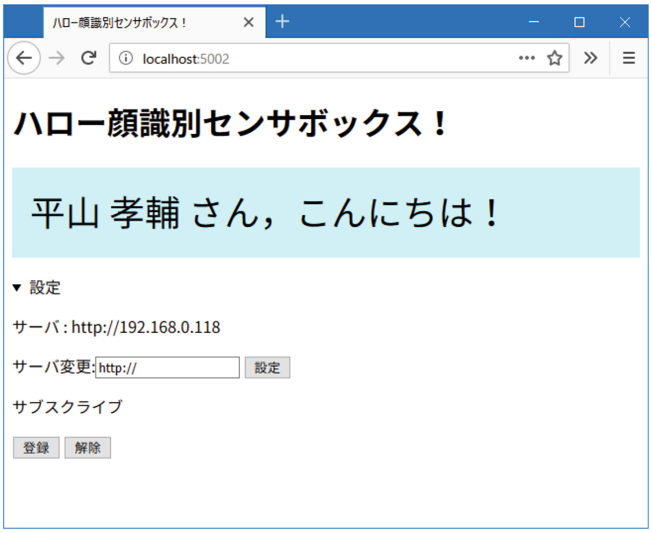


Fig. 4. The role and behavior of Face counter

the sensor box can get face images when users keep still in front of the camera, and the face identification becomes stable. Besides, since Face API (described later) has restrictions on the number and interval of calls, it is impossible to execute massive face identification in a short time, but by using the Face counter, we can reduce the number of face identifications and therefore we can deal with this problem.

Face Identification with Cognitive API. The Face identification controller sends images which are obtained by the face detection to the Face API controller. The Face API controller executes some processes to use Face API, and receive the personId of the person with the highest confidence.

Before identify faces using Face API, we have to register users. We register users through the Data accessor. The Data accessor has an endpoint for user registration in form like `(DeviceURL):(PORT):/register` and by sending necessary information there, we can register new users. The Data accessor also has endpoints to add and delete face images, and these functions are callable by external accesses. The sensor box only notifies uId to applications as a result of the face identification, and does not notify detailed information such as name. Therefore, when applications need detailed information about users, applications have to use the Data accessor to obtain these information using uId as the key. If Face API received a face image of a not registered person and failed to identify, the sensor box notifies "unknown" as a result.


ハロー顔識別センサボックス！

localhost:5002

ハロー顔識別センサボックス！

平山 孝輔 さん，こんにちは！

▼ 設定

サーバ : http://192.168.0.118

サーバ変更: http:// 設定

サブスクライブ

登録 解除

Fig. 5. A screenshot of the sample program

4.5 A Sample Application and Testing

For samples using functions of the prototype of the Face identification sensor box, we made two sample web applications for clients, one is for user registration, and the other is to receive the result of face identification and display the name of the user. These applications access to functions of the sensor box from the outside. We tested whether the sensor box works properly by using these applications.

Firstly, we use the user registration application and register some users on a browser. This application sends information which filled in a certain form to the Data accessor. The Data accessor stores the information to the database, and registers it to Face API. Then, Face API issues personId. personId is represented in hexadecimal form like

```
c0a094ab-e8b0-4368-...
```

After that, a user takes some face photos on the browser, then, this application sends the photos to Face API through the Data accessor.

Next, we start the results receiver application and send its URL to the Result notification controller to subscribe the sensor box. If the subscribe process succeeds, the sensor box returns 200 HTTP status code.

After the above preparations are completed, the user turns one's face to the camera connected to the sensor box, then the sensor box executes face identification. When the face identification completed, Face API returns personId as result. The sensor box converts the personId to uId and sends it to subscribed URLs. The result receiver application receives the result and sends it to the Data accessor to get various information from uId. The Data accessor has an

API which returns various information of a user registered in JSON format by using uId as a key. By sending a uId to URL `/get_data_by_uid` by POST, the Data accessor returns the JSON including information of the user registered in advance. The JSON is like

```
{‘description‘:None, ‘personId‘:‘c0a094ab-e8b0-...‘,
‘gender‘:‘m‘, ‘uId‘:‘hirako‘, ‘birthday‘:‘1996-01-24‘,
‘nickname‘:None, ‘name‘:‘平山 孝輔‘,
‘reading‘:‘ヒラヤマ コウスケ‘}
```

Finally, this application extracts the ‘`name`’ attribute from the JSON and displays (It means “Hello, Kosuke Hirayama !” in Japanese.)

```
平山 孝輔 さん，こんにちは！
```

We show the screenshot of this application in Fig. 5.

5 Conclusion

In this paper, we proposed the Face identification sensor box which can realize development efficiency, effective use of resources and so on in developing applications which has face identification function. In the proposed method, by using the cognitive API of cloud service for face identification, we can easily realize high-quality face identification function and manage it. Also, by incorporating Pub/Sub for communication with applications, the sensor box can asynchronously send the identification results and secure scalability capable of bearing uses from many applications. The future tasks are not only to incorporate this system we implemented into a large scale external application and evaluate its performance, but also to expand the sensor box to cognitive sensor box which can realize various cognitive service other than face identification. In addition, considering the security risk, we will have to put an authentication process before accessing functions of the sensor box from outside, and, to prevent abuses in face identification, we will also have to make the method by which the sensor box can combine authentication methods other than face identification, such as password, with user authentication process.

Acknowledgments. This research was partially supported by the Japan Ministry of Education, Science, Sports, and Culture [Grant-in-Aid for Scientific Research (B) (16H02908, 18H03242, 18H03342), Grant-in-Aid for Scientific Research (A) (17H00731)], and Tateishi Science and Technology Foundation (C) (No.2177004).

References

1. Face API - Facial recognition software – Microsoft Azure. https://azure.microsoft.com/en-us/services/cognitive-services/face/
2. Faceproface recognitionpanasonic security system. https://security.panasonic.com/Face_Recognition/
3. Facial recognition: coming to a gadget near you. https://techxplore.com/news/2019-01-facial-recognition-gadget.html
4. iphone xr - face id - apple. https://www.apple.com/iphone-xr/face-id/
5. A measure to use face identification system to prevent going out from medical/care facilities without permission – face identification based wandering prevention system for dementia patients [lykaon]. https://www.facial-lykaon.com/topics/wandering_prevention.php (in Japanese)
6. MQTT. http://mqtt.org/
7. New biometric identification tools used in theme parks — NEC. https://www.nec.com/en/global/about/mitatv/03/2.html
8. Public safety: Products & solutions — NEC. https://www.nec.com/en/global/solutions/safety/Technology/FaceRecognition/index.html
9. Birman, K., Joseph, T.: Exploiting virtual synchrony in distributed systems. SIGOPS Oper. Syst. Rev. **21**(5), 123–138 (1987)
10. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, vol. 1, p. I, December 2001