MapTalk: mosaicking physical objects into the cyber world

Yi-Bing Lin, Min-Zheng Shieh, Yun-Wei Lin & Hsin-Ya Chen

To cite this article: Yi-Bing Lin, Min-Zheng Shieh, Yun-Wei Lin & Hsin-Ya Chen (2018): MapTalk: mosaicking physical objects into the cyber world, Cyber-Physical Systems, DOI: 10.1080/23335777.2018.1510856

To link to this article: https://doi.org/10.1080/23335777.2018.1510856

Published online: 28 Aug 2018.

Submit your article to this journal

View Crossmark data
MapTalk: mosaicking physical objects into the cyber world

Yi-Bing Lin*, Min-Zheng Shieh*, Yun-Wei Lin* and Hsin-Ya Chen*

*Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan; *Information Technology Service Center, National Chiao Tung University, Hsinchu, Taiwan

ABSTRACT

Digital map is considered as a cyber world, which maps visual representations (cyber objects) to the physical objects in the real world and allows the user to interact with these physical objects through their cyber representations. However, it typically requires significant programming effort to create a map application. This paper proposes MapTalk, a web-based visual map platform that allows the user to interact with the physical objects through their cyber representations in a visual map. We show how the administrator can add applications to the map without any programming effort. The novel idea in our approach is to utilise the IoT concept. Specifically, we implement the map as an output IoT device and all physical objects to be mosaicked in the map as input IoT devices. We show how to automatically create the device features of IoT devices when the administrator creates applications in the Map web page. We have deployed over 20 services in MapTalk including smart home, smart farm, tracking (bus, person, dog, etc), video monitoring, travel route planning, sensing of CO2, PM2.5, Internet of Things (IoT), temperature, humidity, and so on.

ABBREVIATION: Internet of Things (IoT)

1. Introduction

Angelina Jolie said: ‘Anytime I feel lost, I pull out a map and stare. I stare until I have reminded myself that life is a giant adventure, so much to do, to see.’ Today, digital map is considered as a cyber world, which maps visual representations (cyber objects) to the physical objects in the real world and allows the user to interact with these physical objects through their cyber representations. Tools such as Google Maps offers the Application Programming Interface (API) to support interaction between the services/applications and the map for visual illustration. Significant amounts of location information are already built in the map tools, which can be conveniently reused by various applications.
In Indonesia, the digital land certificate mapping application was developed [1] to assist the public and land officers to identify the land locations. The coordinates are determined by GPS. As the author reported, the precision of this application is not as good as expected and is in the enhancement process. The work in [2] developed a travel-planning platform based on the Google Maps, which utilises TraNews and data sources to assist planning itineraries to enhance user travel experience. In [3], the authors describe how to integrate the .NET technology with the Google Maps API interface to develop a web portal for driver applications. To add user-friendly interactive applications on websites based on Google Maps, an instant messaging system designed by Flex is described in [4]. In this study, a travel application embedded in a micro-blog system was implemented to display dynamic map content. The aforementioned studies demonstrated the power of map tools and the values of location applications. However, developing new applications in these approaches requires intensive programming efforts. To resolve this issue, we propose MapTalk that allows the user to conveniently develop applications on digital map without programing effort. The novel idea of MapTalk is to utilise the IoT concept. Specifically, we implement the map as an output IoT device and all physical objects to be mosaicked in the map as output IoT devices. We use the term ‘designer’ to describe the person who develops new features of MapTalk and is required the programming skills for Google Maps API, Flask, SQLAlchemy, WebSocket, Python, JavaScript, jQuery, HTML and CSS. An ‘administrator’ is the person who builds the location applications using MapTalk GUI without extra programming effort. This paper is organised as follows. Section 2 describes the cyber representation of a physical object. Section 3 shows how to create an application and its cyber objects in MapTalk. Section 4 elaborates on how to mosaic physical objects into MapTalk. Section 5 describes how the visual map is implemented as an IoT device in MapTalk.

2. Cyber representations of physical objects in MapTalk

MapTalk is a simple version of Integrated Operation Centre (IOC) software that allows efficient and convenient administration of intelligent environments such as smart campus [5]. In National Chiao Tung University (NCTU), the physical objects (school buses, parking lot sensors, washing machines, dryers, curtains, air conditioners, environmental sensors) are managed by the MapTalk IOC (a web page illustrated in Figure 1) that allows the university technical staff to conveniently administrate all smart campus applications under one web-based graphical user interface (GUI).

MapTalk effectively mosaics physical objects of various IoT applications in a 2D map. Our approach can automatically accommodate IoT applications with few simple steps. The web-based GUI of MapTalk is considered as a cyber world that interacts with the physical objects and visually shows them to the viewer (a user) of these applications. Therefore, MapTalk allows the user to talk and listen to physical objects through their cyber counterparts in the map.
An IoT application is represented as a colored button for a drop-down list (Figure 1(a)). When the button is clicked, the names of all physical objects of that application are shown in the list. The user can select one or all objects. Then the cyber objects (visual representations of the physical objects) are shown in the map according to their physical locations.

Figure 2 illustrates the structure of a cyber object with the following attributes:

**Layout** (Figure 2(a)): A cyber object can be a picture object (e.g. a camera in Figure 1(b)) or a geometric shape object. A geometric shape object can be a rounded rectangle (Figure 3(a)), a circle (Figure 3b)) or a rounded square (Figure 3(c)), which is created with following features.

---

**Figure 1.** A 2D visual map based on MapTalk.
Name (Figure 2(2)): In the space surrounded by the geometric shape, the designer can use a letter to specify the abbreviated object name; for example, ‘W’ for the washing machines in a laundry room (Figure 3(1)) and ‘P’ for PM2.5 (Figure 3(2)). The shape of an object with abbreviated letter is either rounded rectangle (Figure 3(a)) or rounded square (Figure 3(c)). On the other hand, an object without the abbreviated letter is a circle (Figure 3(b)).

Colour (Figure 2(3)): The background colour of a rounded rectangular or a circle object is the same as that of its application drop-down menu button. Different applications are automatically assigned different colours. For example, the colour for washing machines in a laundry room is sky blue (Figure 3(3)), and the colour for a tracked bus is violet red (Figure 3(4)).

Value (Figure 2(4)): A text value (Figure 2(5)) can be shown inside a rounded rectangular or a circle object. For example, the value 9 (Figure 3(5)) represents the number of idle washing machines, and the value 2 (Figure 3 (6)) is the identity of a tracked bus. The colour (Figure 2(7)) can be different for different values.

Figure 2. The structure of cyber objects in MapTalk.

(a) Round rectangle  (b) circle  (c) rounded square

Figure 3. Layouts for geographic shape objects Round rectangle (b) circle (c) rounded square.
(6) fills the inside space of a rounded square object represents the state of the object. The colours are mapped to a range \([\text{Min, Max}]\) of float numbers. For example, in the PM2.5 application, the PM2.5 value range \([1, 10]\) is mapped to colour blue, \([11, 20]\) to green, \([21, 30]\) to yellow, \([31, 40]\) to orange, and any value larger than 41 is mapped to red. The colour spectrum is shown in Figure 1(e).

- **Location (Figure 2 (7))**: A non-tracked object is stationary (Figure 2(8)), and its location is fixed in the map. A tracked object is movable (Figure 2 (9)), and its location is dynamically updated in the map. The tracked objects are used in tracking people, animal, cars, and so on. The trajectory of a tracked object (Figure 2(10)) can be shown with the same colour as the letter in the object (e.g. the trajectory and object letter 2 has the colour orange; see Figure 1(c) and (g)).

**Description (Figure 2 (11))**: When the user clicks the left button of the mouse on a cyber object, a dialog box pops up (Figure 1(d)), which typically contains a text to describe the object or a hyperlink leading to a historical chart (Figure 2 (12)) or action buttons (Figure 2 (13)) allowing the user to interact with the physical objects.

The cyber objects can also be classified into two categories: single-attribute and composite-attribute. A single-attribute object represents a set of homogeneous sensors. A composite-attribute object represents a set consisting of heterogeneous sensors, controllers and/or actuators (to be elaborated in Section 4).

The menu bar (Figure 1(h)) consists of three types of buttons. All applications managed in MapTalk can be found in the App drop-down menu (Figure 1(i)). The routing button (Figure 1(j)) provides travel route planning by allowing the user to specifying the start and destination locations. This application is the same as that offered in Google Maps, and the details are given in Section 5. The menu bar also provides a coloured button for a frequently accessed application (Figure 1(a)).

### 3. Creating a location application in MapTalk

The administrator creates and manages the physical objects of a location application through the MapTalk administration web page. This web page is the same as that viewed by the user except that in the App drop-down menu (Figure 1(i)), there is the ‘Add app’ item (see Figure 7(a)). When the administrator clicks on this item, the application setup bar is shown below the menu bar (Figure 4(a)-(l)). This bar consists of eight fields. The Name fields (Figure 4(a)) allow specification of the application name (e.g. WashingMachine). The Location field (Figure 4(b)) implements the location attribute of the cyber object (Figure 2(7)), where either ‘Stationary’ or ‘Movable’ should be selected.

The Layout field (Figure 4(c)) implements the layout attribute (Figure 2(1)), which provides three options. If the Picture item (Figure 4(d)) is selected, the
window pops up a dialog box that allows the administrator to fill the Picture URL (Figure 5) linking to the picture to be shown in the MapTalk GUI. If the ‘No Letter’ item (geometric shape without abbreviated name; see Figure 4(e)) is selected, the geometric shape will be circle, and no abbreviated letter will be shown inside the circle. If a letter field (geometric shape with abbreviated name; see Figure 4(f)) is selected, the geometric shape will be rounded rectangle or rounded square, and the selected letter will be shown inside the shape.

The Value field (Figure 4(g)) implements the value attribute of the cyber object (Figure 2(4)) with two options. If the Text item (Figure 4(h)) is selected, the shape of the cyber object will be rounded rectangle. If the Colour field (Figure 4(i)) is selected, a dialog box (Figure 6) pops up to fill the minimum and the maximum values for the colour range. In this case, the shape of the cyber object is rounded square.

The Quick Access field (Figure 4(j)) allows the administrator to put the created application in the menu bar (e.g. Figure 7(c)) for quick access. After the application is created, the administrator can save the setup (Figure 4(k)) or remove the setup (Figure 4(l)). The created application is always included as an item in the App drop-down menu (Figure 7(b)).

To add a new object in the application (e.g. WashingMachine), the administrator clicks the Add WashingMachine item (Figure 7(d)), and the object setup bar is shown below the menu bar (see Figure 8 (a-c, e, f)). This bar consists of six fields.

The Name field (Figure 8(a)) specifies the object name. The value of the Location field (Figure 8(b)) can be obtained by either manually inputting the latitude and the longitude (Figure 8(c)) or clicking the mouse at the location of the map. After this field is filled, the cyber object is shown in the map (Figure 8(d)).
The Description field (Figure 8(e)) implements the Description attribute of the cyber object (Figure 2(11)), which allows specifying the text description or the hyperlink to a history web site (to be elaborated in the next section). After the cyber object is set up, the administrator clicks the ‘Save’ button (Figure 8(f)) to add this cyber object to the application.

The washing machine example illustrates the creation of a stationary object. For movable objects such as buses, the creation is basically the same except for the following differences. The ‘Movable’ item is selected from the Location field (Figure 4(b)). If the Picture or a Letter field is selected from the Layout field (Figure 4(c)), all buses will have the same appearance in the map. If it is desirable to distinguish the tracked buses, ‘No Letter’ item is selected. In this setup, the geometric shape will be circle. No abbreviated name will be shown inside the circle; instead, the bus
identity is shown. We do not need to create individual bus objects as we did for washing machines in Figure 8. When a new bus joins in this application, its location and identity (an integer) are sent to MapTalk (to be described in the next section) and automatically shown in the map (e.g. Figure 1(f,g)).

4. Mosaicking physical objects into MapTalk

MapTalk is an enhancement of IoTtalk, an IoT application platform for IoT device interaction and management [6–9]. In this platform, an input IoT device sends data to the IoTtalk server for manipulation, and then the results are sent to one or more output IoT devices to drive these devices. MapTalk develops a visual map as an output IoT device called ‘Map’. Any application of the physical objects to be mosaicked in the map is implemented as an input IoT device in MapTalk. For example, a laundry room with 20 washing machines and 10 dryers is implemented as two single-attribute input devices called LdryRmWM and LdryRmD, respectively. The administrator manipulates these devices through the MapTalk GUI. MapTalk interacts with the physical objects of an application based on the concept called ‘device model’. Through the procedure described in [10], two device models for laundry room are built in MapTalk and are included in the device model drop-down list in the menu bar of the MapTalk GUI (Figure 9(1)). When the designer selects ‘Washing Machine’ from the list, the icon for the LdryRmWM device is shown in the left-hand side of the graphical window (Figure 9(2)). The physical objects of the same type are grouped and represented as a single-attribute cyber object in MapTalk through the concept of ‘device feature’ (DF). The DFs of an input device are called input DFs (IDFs). Therefore, the 20 washing machines in the laundry room are represented by the IDF icon WashMach-I inside LdryRmWM (Figure 9(3)). Similarly,
the 10 dryers in the laundry room are represented by the IDF icon Dryer-I inside LdryRmD (Figure 9(4)). We note that all IDF names are appended with ‘-I’ and all ODF names are appended with ‘-O’. To identify the location at the map, both LdryRmWM and LdryRmD also include an IDF called Location-I. This IDF may represent a real location sensor (e.g. the GPS receiver or iBeacon) or a preset latitude and longitude value pair.

As an output IoT device, the Map icon is placed in the right-hand side of the graphical window (Figure 9(5)). From the above description, we treat a device model as a collection of DFs. An input (output) device is a set of IDFs (ODFs). The MapTalk server links an IDF to an ODF by two line segments connected to a small ‘Join’ circle. By clicking the circle, the designer can implement the functions to manipulate the data sent from the IDF to the ODF. In Figure 9, WashMach-I of LdryRmWM connects to Map or more appropriately, connects to the WashMach-O ODF of a cyber IoT device called WMStatus (Figure 9(6)). The location of the washing machines is also passed to WMStatus through the LocWM-I (Figure 9(7)) and LocWM-O (Figure 9(8)) connected by Join 1. Similarly, Dryer-I is connected to the IoT device DryerStatus (Figure 9(9)) through Join 3. If an IoT device includes both IDFs and ODFs, then it is represented by an input device and an output device in MapTalk. WMStatus has both input device and output device (Figure 9(6) and (10)). Similarly, DryerStatus is represented by both input and output devices.

Figure 9. Laundry application configured in the MapTalk GUI.
The output device of WMStatus lists the status of each washing machine and the history line chart for that machine in a web page (Figure 10). The input device of WMStatus counts the number of idle washing machines and sends this number out through the IdleWM-I IDF (Figure 9(10)). This IDF is a counter associated with the latitude and longitude pair obtained from LocWM-O. The IdleWM-I IDF is connected to the IdleWM-O ODF of Map (Figure 9(11)) through Join 5, and the map shows the number of idle washing machines in the map (see Figure 1(d); the rounded rectangle indicates nine idle washing machines in this example). The administrator fills the hyperlink for the WMStatus web page in the Description field in the object setup bar in Figure 8(e). Therefore, when the user clicks the hyperlink in Figure 1(d), the web page in Figure 10 pops up.

Figure 9 also includes a BusTracking icon connected to Map through Join 7. The BusID-I IDF (Figure 9(12)) provides the locations of buses been tracked (Figure 1(f, g)). As mentioned before, Figure 1 shows tracking of two buses represented by violet-red circles marked with the spring-green number 0 (the first bus) and the orange number 2 (the second bus) with the tails (colored line segments) indicating the historic traces of the moving buses.

In the Map output device, the ODFs (i.e. BusID-O, IdleWM-O, IdleDryer-O) are automatically created after the administrator has set up the applications (Bus, Washing Machine, and Dryer) in Map web page and clicks the ‘Save’ button (Figure 4(k)).

In Figure 9, the input devices connected directly or indirectly to Map are single-attribute cyber objects. To mosaic a composite-attribute cyber object to Map, there are two alternatives. Typically, a web page is implemented for a mosaic composite-attribute input device. An example is RoomTalk, a smart home application that consists of multiple switches and sensors to control the home appliances. Figure 11 illustrates a part of the web page for RoomTalk. The right-hand side of the figure shows nine buttons for controlling curtains (there are more than 30 buttons in RoomTalk). The web page also mosaics a video streaming platform in the left-hand side of Figure 11. The camera in the video streaming platform shows all actuators (ODFs) controlled by the IDFs of RoomTalk.

The RoomTalk configuration in the IoTtalk GUI is illustrated in Figure 12. In this configuration, no RoomTalk IDFs are connected to Map. The administrator only needs to create the RoomTalk application and the cyber object in the Map web pages (e.g. Figures 4 and 8), and there is no need to make any connection in the MapTalk GUI (e.g. Figure 9).

The input devices for single-attribute cyber objects in Figure 9 can also be implemented by the input device for a composite-attribute cyber object. Suppose that the washing machines and the dryers mentioned in Figure 9 are located in the same laundry room. Then they can be implemented as a composite-attribute cyber object by merging both LdryRmWM and LdryRmD together with a temperature sensor into an input device called LdryRm
The location of LdryRm is specified at LocLR-I, and LdryRmWM and LdryRmD are represented by the WashMach-I and the Dryer-I IDFs, respectively. All IDFs are connected to an output device called LdryRmStatus (Figure 13(2)) that is similar to WMStatus and DryerStatus. A web page is automatically created for LdryRmStatus (similar to the one illustrated in Figure 10)

In the first alternative to mosaic a composite-attribute input device into Map, no IDFs of the input device is connected to the ODFs of Map. Instead, the administrator simply inputs the hyperlinks of the RoomTalk and the LdryRmStatus web pages in the Description field (Figure 8(e)). In this alternative, when the user clicks the RoomTalk (LdryRm) cyber object in the map, the RoomTalk (LdryRmStatus) web page hyperlink pops up. If the user clicks the hyperlink, the web page will be shown on the screen for manipulation.

Figure. 14 illustrates the relationship between LdryRm, RoomTalk and Map. In the first alternative, although all of these three IoT devices connected to MapTalk server (Figure. 14 (1), (2), and (3)), their IDFs are not connected to the Map ODFs. Instead, the hyperlinks of LdryRm and RoomTalk are mosaicked in the map (Figure. 14 (4) and (5)).
In the second alternative, some IDFs of a composite-attribute input device are connected to the ODFs of Map (see Join 5 in Figure 15). In this example, the administrator has created a temperature application, which results in the ODF Temperature-O of Map in the MapTalk GUI (Figure 15(1)). When the administrator connects Temperature-I of LdryRm (Figure 15(2)) and RoomTalk (Figure 15(3)) to Temperature-O of Map, the temperatures of the laundry room and the smart home demo room will be directly shown in the visual map through the Temperature application of Map. In this alternative, the physical...
and cyber temperature objects also interact through the MapTalk server (i.e. through (1), (2) and (6) in Figure 14).

5. The map implementation as an IoT device

This section describes how Map is implemented as an IoT device. Depending on the context, we will use the term ‘user’ (Figure 16(8)) to represent ‘administrator’ and ‘user’ interchangeably. Figure 16 illustrates the functional blocks of the Map device. The MsgHandler (Figure 16(1)) is the core engine of the Map device, which is responsible for interaction with the MapTalk server (Figure 16(2)) through three tasks.

- The Register task (Figure 16(3)) connects the Map device to the MapTalk server. The Map device interacts with the MapTalk through MQTT or HTTP. After registration, the user/administrator can manipulate the map as described in Section 3.
- The Update task (Figure 16(4)) informs the MapTalk server of the change for Map. When the administrator adds a new application (e.g. Washing Machine) to Map, the MapTalk server reflects such changes of Map icon in Figure 9 by automatically showing the IdleWM-O ODF.
- In the Pull task (Figure 16(5)) the MsgHandler receives the data of the cyber objects (IDFs) from the MapTalk server through the Map’s ODFs. The MsgHandler then asks the DB Control (Figure 16(6)) to store the ODF data (with the format [lat, lng, name, value, timestamp]) in the database Map DB (Figure 16(7)). The protocol exercised between the MsgHandler and the DB Control is HTTP.
- After an input device (e.g. LdryRmWM) has been connected to Map described in Section 4, the IDFs of the input device send the data to
Map’s ODFs through the Join connections. The MsgHandler retrieves the data through the Pull task described above. Also, the MsgHandler instructs the Map Control (Figure 16(10)) to update the values of the corresponding cyber objects (Figure 2(4)) and then instructs the Graphic
Render (Figure 16(11)) to show the results to the user following the message path (2) → (5)→(1)→(1) → (10) → (11)→(8) in Figure 16.

- The DB Control is implemented by Flask, which interacts with the Map DB through ORM API. Besides storing the ODF data, the Map DB also stores the app buttons (see Figure 1(a)) with the format [name, location, layout, value, quick access], the setups for every stationary cyber object (see Figure 3(a,c)) with the format [name, location, description], and the setups for every movable cyber object (see Figure 3(b)) with the format [name, description].

- When the user manipulates the application through the Map web page, the User Event Handler (Figure 16(9)) catches the events and dispatches these events to other components of the Map device.

- When the administrator adds a new application (e.g. washing machines) by selecting the ‘Add app’ item (Figure 7(a)), the User Event Handler instructs the Graphic Render to generate the application setup bar (Figure 4(a)-(l)) through the message path (8)(9) → (11) → (8) in Figure 16.

- When the administrator presses the ‘Save’ button (Figure 4(k)), the User Event Handler sends the app setup information to the MsgHandler. The MsgHandler invokes the Update task to the MapTalk server to create related ODF (i.e. IdleWM-O) for the Washing Machine (message path (1)→(4) → →(2)). The MsgHandler also issues an HTTP request that instructs the DB Control to save the setup information to the Map DB (message path (1)→(6)→(7)).

- After creation of an application, the administrator can add a cyber object to the application by pressing, for example, the ‘Add WashingMachine’ item (Figure 7(d)). The User Event Handler requests the Graphic Render to show the object setup bar (Figure 8(a-f)). When the administrator clicks a location in the map, the User Event Handler instructs the Map Control to use the Google Maps JavaScript API of the Google Maps Platform (Figure 16(14)) to obtain the latitude and the longitude of the hit position (message path
When the administrator presses the ‘Save’ button in Figure 8(f), the User Event Handler dispatches the event to the MsgHandler. The MsgHandler sends an HTTP request to the DB Control to save the new object in the Map DB and retrieves the information of the application and the cyber objects (message path (1) → (6) → (7) → (6) → (1)). The MsgHandler then sends them to the Graphic Render to generate the application quick access drop-down list button in the menu bar and all cyber object items of the drop-down list in the Map web page (message path (1) → (11) → (8)). In parallel, the User Event Handler instructs the Map Control to send an HTTP request to the Google Maps Platform to obtain the JavaScript API for future usage (message path (9) → (10) → (14) → (10)).

When the user selects the ‘show all’ item or a cyber object item in an application drop-down menu button (Figure 1(a)), the User Event Handler dispatches the event to the Map Control. The Map Control uses the Google Maps JavaScript API to layout all cyber objects (or a specific object) and shows them in the map (message path (8) → (9) → (10) → (11) → (8)).

If the user clicks the Routing button (Figure 1(j)), the event is dispatched to the Map Control. The Map Control instructs the Graphic Render to show a pop up window (Figure 17(a)) that allows the user to specify the starting point and the destination (message path (8) → (9) → (10) → (11) → (8)). After the user has set up the starting point/destination and
clicks the ‘Start Routing’ button (Figure 17(b)), the User Event Handler dispatches the event to the Map Control. The Map Control invokes the directions service of the Google Maps Platform to obtain the suggested travel path, and draw it through the JavaScript API. The result is sent to the Graphic Render to show in the map (Figure 17(c)). The message path is (8)→(9)→(10)→(14)→(10)→(11)→(8).

- When the user clicks on a camera cyber object (Figure 1(b)), the User Event Handler dispatches the event to the Map Control. The Map Control orients the center of the map above the camera cyber object and sends the streaming request to the Video Control (Figure 16(12)). The message path is (8)→(9)→(10)→(12). The Video Control then sends the corresponding RTSP request to the Streaming Server (Figure 16(13)), and the video is viewed by the user through the message path ((13)→(12)→(11)→(8)). The user watches the video in the map as illustrated in Figure 18.

6. Conclusions

This paper proposed MapTalk, a web-based visual map platform that enables the user to interact with the physical objects through their cyber representations in a visual map. To allow the administrator to add applications to the map without any programming effort, the implementation of MapTalk is not trivial. The novel idea in our approach is to utilise the IoT concept. Specifically, based on IoTtalk, we implement the map as an output IoT device and all physical objects to be mosaicked in the map as input IoT devices. We showed how to automatically create the device features of an IoT device when the

\[\text{Figure 18. Video streaming in MapTalk.}\]
administrator creates the corresponding application in the Map web page. The advantages of MapTalk include:

- Cyber physical interaction of visual map is achieved through a well-established IoT platform.
- Any physical system (such as school buses, laundry rooms, etc.) is automatically represented as a cyber IoT device and then easily accommodated in the map for illustration.
- No programming effort is required to add a new application in the map.

In NCTU MapTalk IOC, we have included over 20 services including smart home, smart farm, tracking (bus, person, dog, etc), video monitoring, travel route planning, sensing of CO2, PM2.5, temperature, humidity, and so on.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Ministry of Education [the SPROUT Project-Center for Open Intelligent Connectivity of NCTU]; Ministry of Science and Technology of the Republic of China [Grant 106-2221-E-009 –006 and Grant 106-2221- E-0]; Academia Sinica [AS-105-TP-A07].

ORCID

Yi-Bing Lin  http://orcid.org/0000-0001-6841-4718

References


