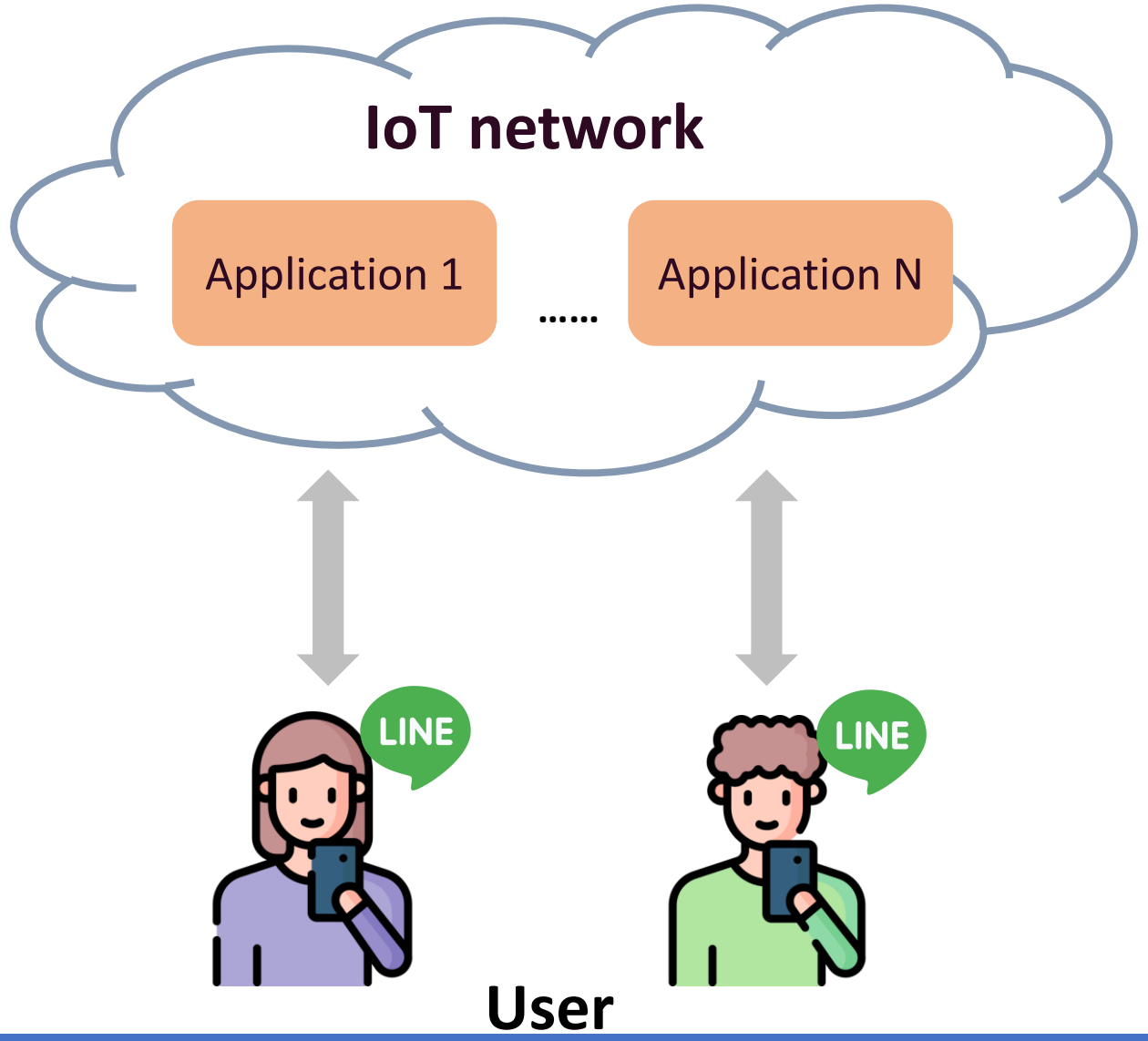
Mini project 4

IoT service platforms and applications

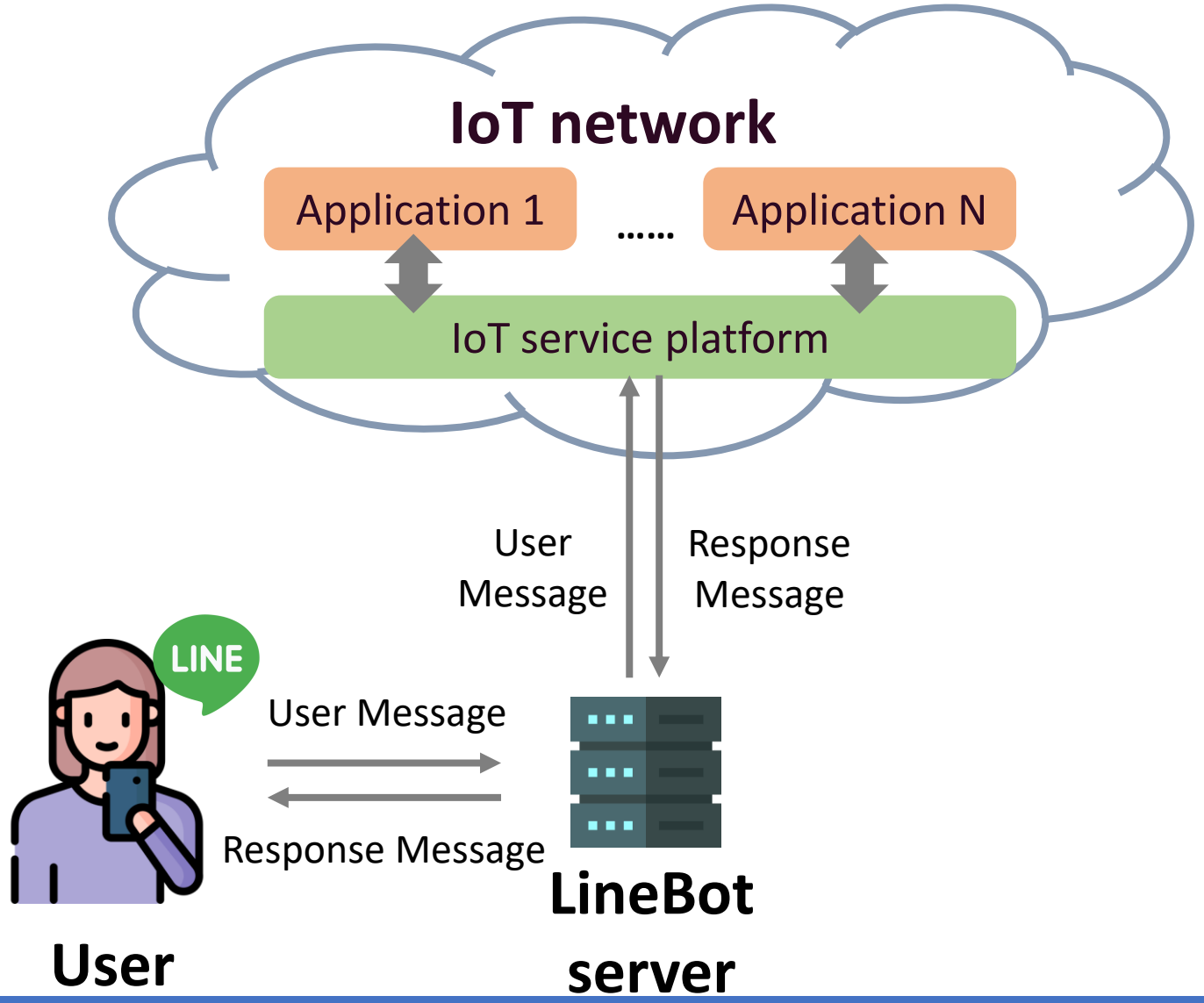
Mini Project 4 “IoT service platforms and applications” - Goal

- Familiar with the operation of IoT service platforms
  - IoTtalk as an example
- Development of an IoT application
  - Focus on “LineBot” related application



Mini Project 4 “IoT service platforms and applications” - Goal

- You will learn:
  - how to deploy application on IoTtalk
  - how to register IoT device to IoTtalk
    - LineBot
  - how to build LineBot
    - deploy LineBot server



Outline

- Overview of IoTtalk
- Introduction to LineBot
- Mini project 4
  - Project description
  - Configuration and application deployment on IoTtalk
  - LineBot tutorial

Overview of IoTtalk

Cellular IoT in 5G

- Cellular IoT connects physical devices to the Internet using essentially the same technology as your smartphone
  - Instead of needing to create a new, private network to operate your IoT devices
- Almost every industry can be transformed with cellular IoT
- The connectivity needs of all industries can be addressed by four multi-purpose IoT segments, which efficiently co-exist in one 5G network. These segments are:
  - Massive IoT
  - Broadband IoT
  - Critical IoT
  - Industrial Automation IoT

Massive IoT

- Massive IoT connectivity targets a large number of low-cost, narrow-bandwidth devices that infrequently send or receive small volumes of data
- These devices can be situated in challenging radio conditions requiring extreme coverage and may rely solely on battery power supply

One 5G network with four multi-purpose IoT connectivity segments			
Massive IoT	Broadband IoT	Critical IoT	Industrial Automation IoT
Low-cost devices Small data volumes Extreme coverage	High data rates Large data volumes Low latency (best effort)	Bounded latencies Ultra-reliable data delivery Ultra-low latency	Ethernet protocol integration Time-Sensitive Networking Clock synchronization as a service
Network slicing, network exposure, network data analytics, device positioning, device battery life			

Broadband IoT

- Broadband IoT connectivity adopts the capabilities of MBB for IoT to provide much higher data rates and lower latencies than Massive IoT, while enabling additional capabilities for IoT, such as:
  - Extended device battery life
  - Extended coverage
  - Enhanced uplink data rates
  - Enhanced device positioning precision
- Broadband IoT is relevant for all industries. Commercial usage today is dominated by personal cars, commercial vehicles, trains, wearables, gadgets, cameras, sensors, actuators and trackers

Critical IoT

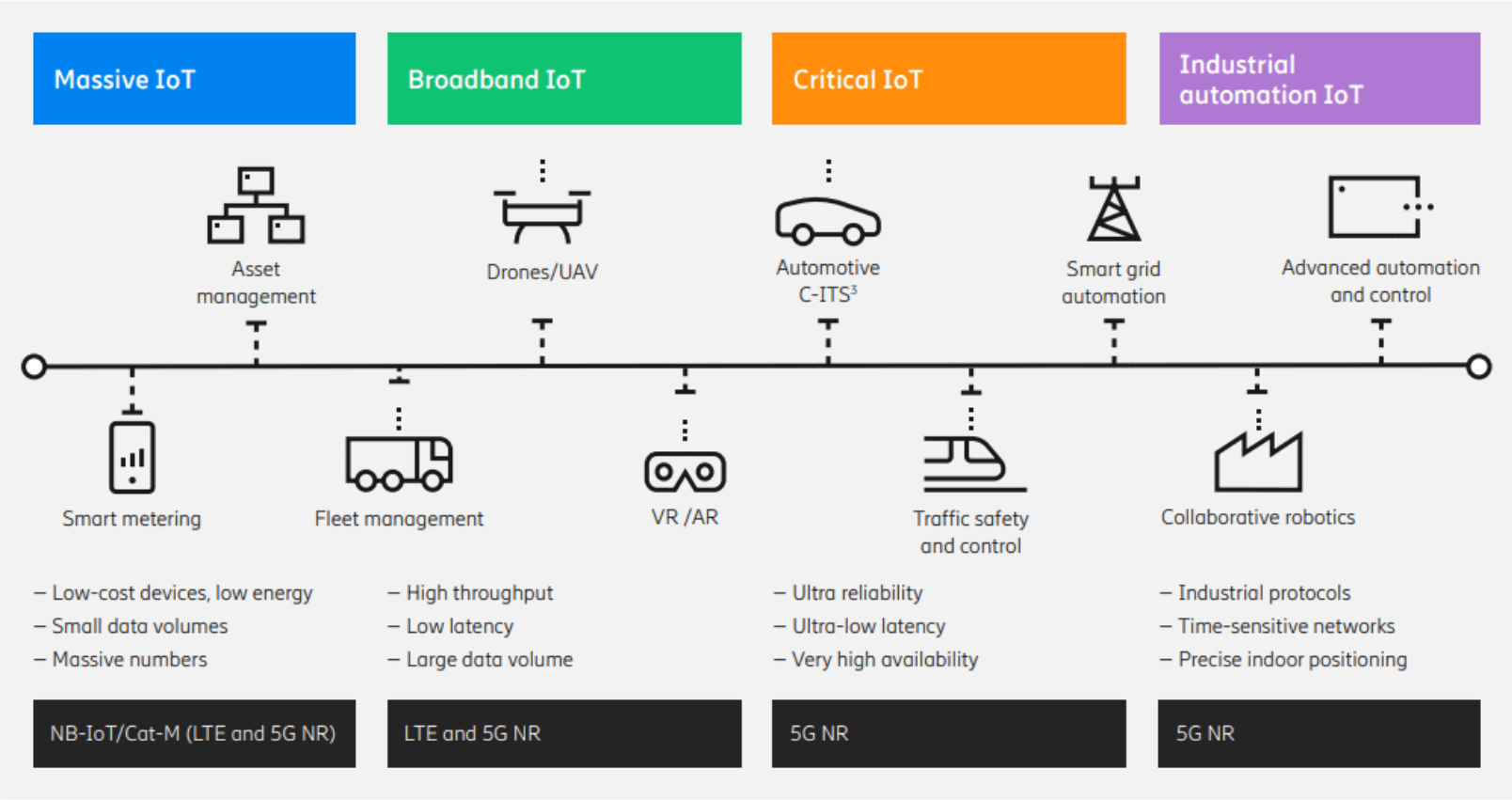
- Critical IoT connectivity is for time-critical communication. It enables data delivery within desired latency bounds.
- In contrast to Broadband IoT, which achieves low latency on best effort basis, Critical IoT can deliver data within specified latency bounds with required guarantee levels, even in heavily loaded networks.
- Typical use cases with demanding combinations of reliability, latency and data rates include:
  - AR/VR, autonomous vehicles, mobile robots, real-time human machine collaboration, …

Industrial Automation IoT

- Industrial Automation IoT aims at enabling seamless integration of cellular connectivity into the wired industrial infrastructure used for real-time advanced automation
- It includes capabilities for integrating 5G systems with real-time Ethernet and Time-Sensitive Networking (TSN) used in industrial automation networks

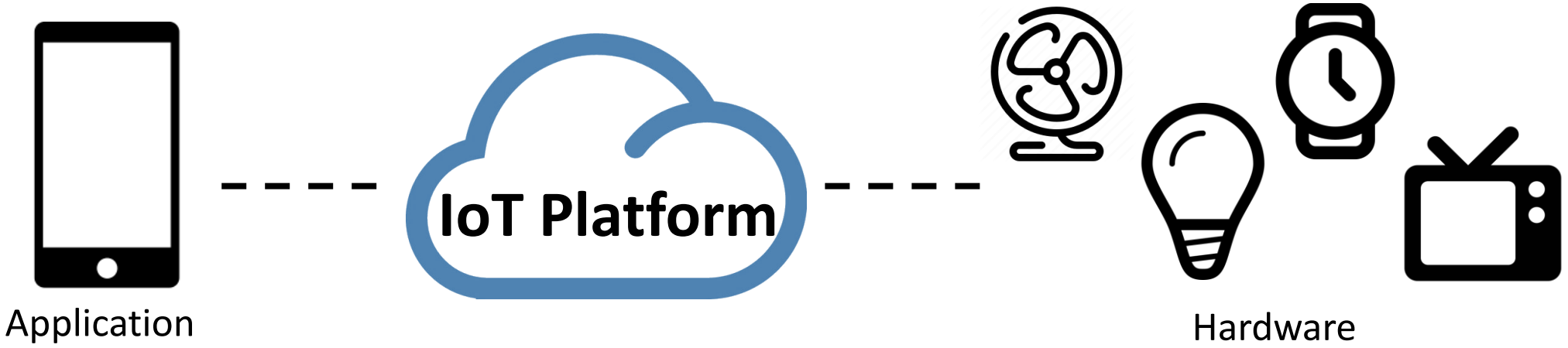
Cellular IoT Use Case

Cellular IoT use case segments



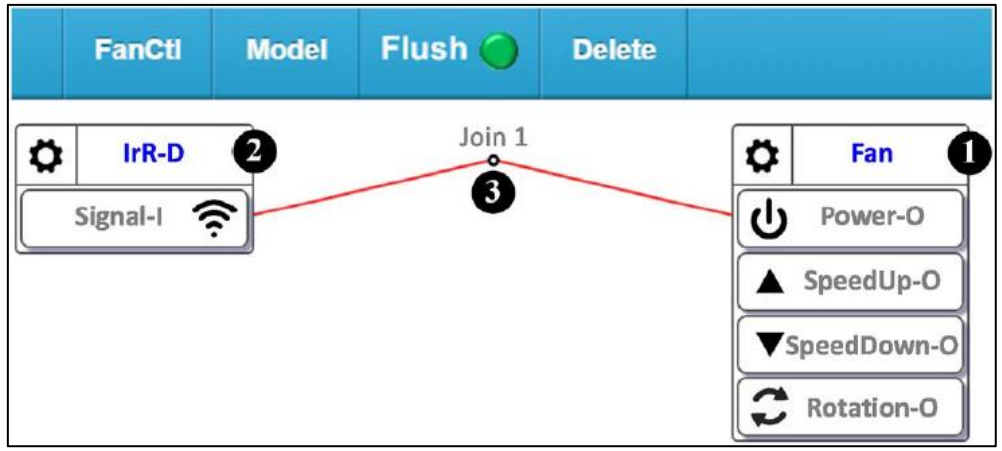
What is an IoT Platform?

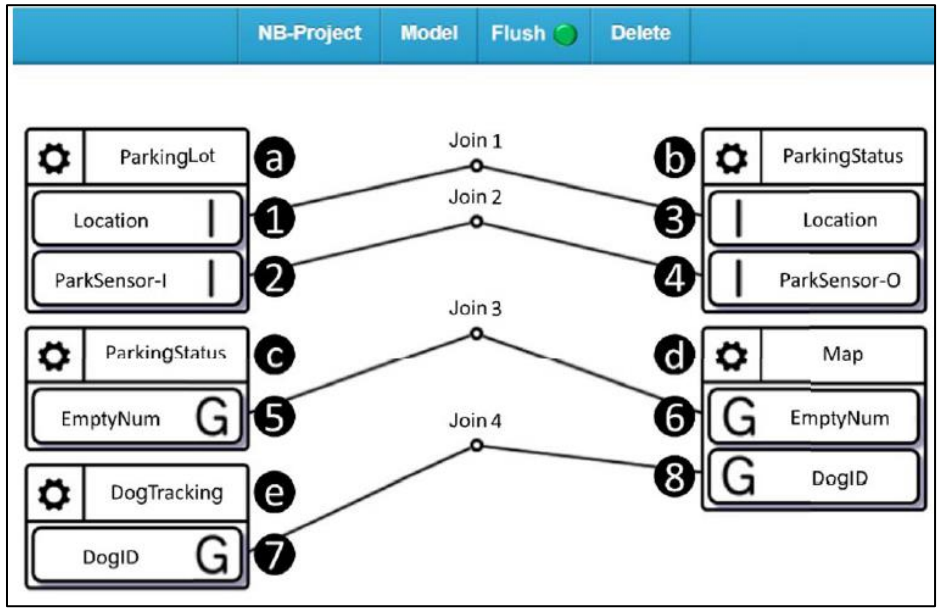
- An IoT platform is a multi-layer technology that enables straightforward provisioning, management, and automation of connected devices within the Internet of Things universe.
- It is commonly referred to as middleware when we talk about how it connects remote devices to user applications and manages all the interactions between the hardware and the application layers.



IoTtalk: an IoT service platform

- IoTtalk is a smart IoT application development platform
- It can connect different IoT application platforms with different specifications and standards
  - Smart home
  - Smart campus
  - Smart farm





Device Feature

- An IoT device can be characterized by its functionalities or “device features”

- For the purpose of description, we define a device feature (DF) as a specific input or output “capability” of the IoT device
  - A wearable ring with the temperature sensor has the input device feature (IDF) called “Temperature”
  - A pair of wearable glasses with the optical head-mounted display has the output device feature (ODF) called “Display”

Network Application

- An IoT device may be connected to the network (i.e., Internet) using wireless communications directly or indirectly through a smartphone

- If so, the corresponding software called network application is developed and executed by a server in the network side
  - Receives or sends the messages from/to the IoT device

Network Application

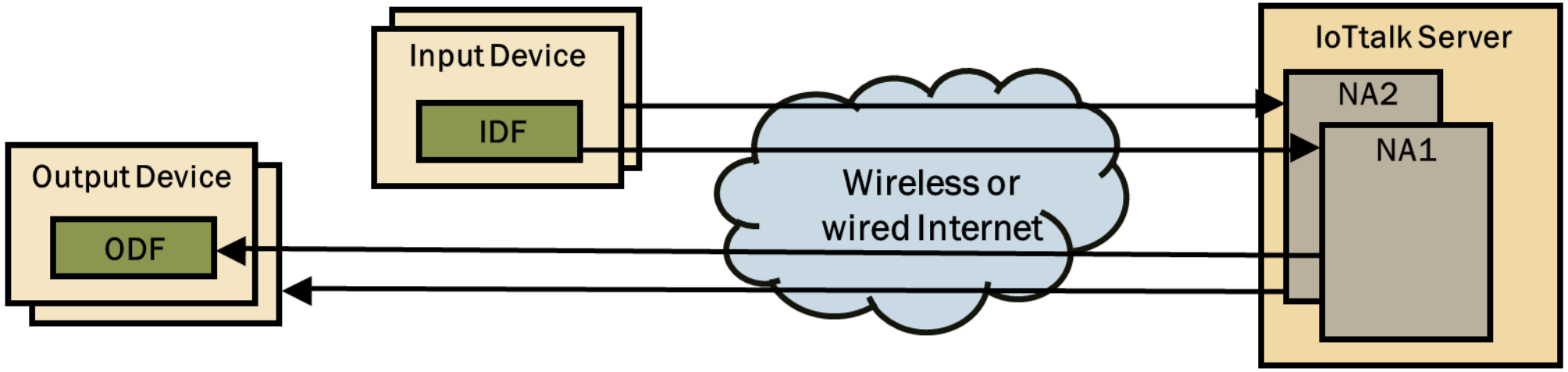
- When the IDFs produce new values

  1) The IoT device inform the network application to take some actions

  2) The network application send the result to the ODF of an IoT device

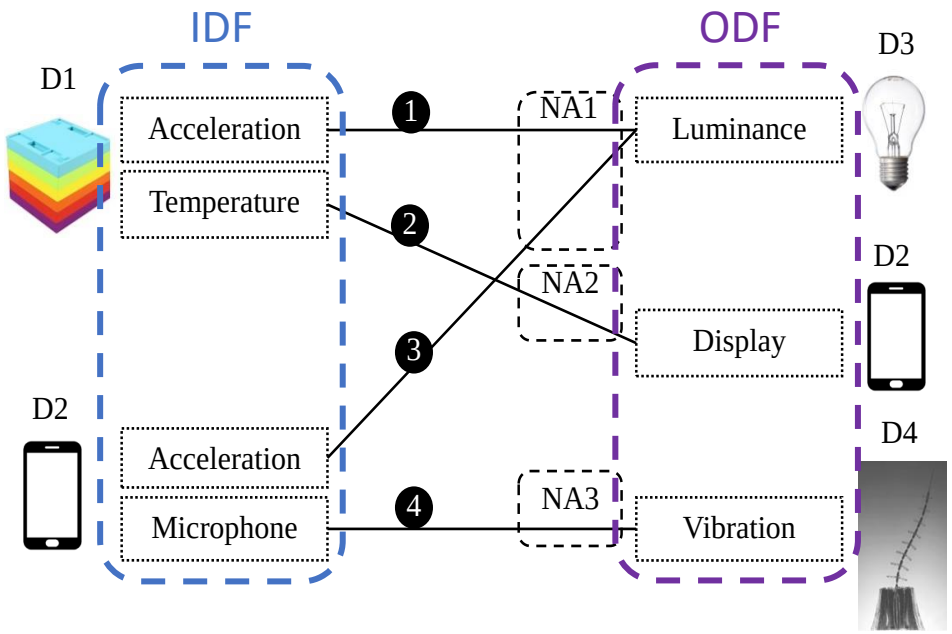
- The IoT devices interact with each other through their features

  - The network application “maps” the IDFs to the ODFs



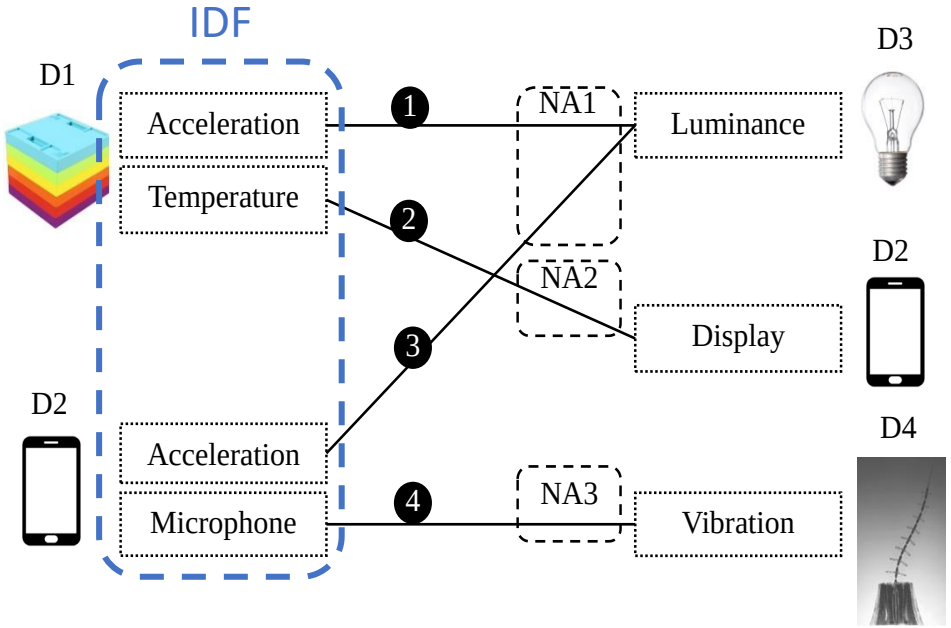
IoTtalk: an IoT service platform

- 4 IoT devices D1, D2, D3 and D4
  - The left-hand side of the figure illustrates the IDFs of the devices
  - The right-hand side of the figure illustrates the ODFs of the devices



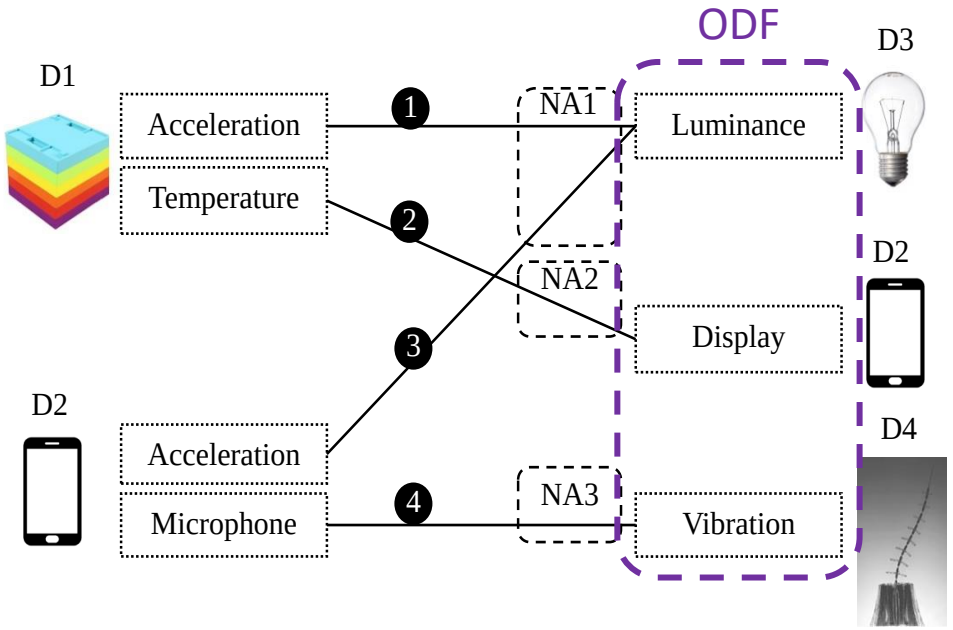
IoTtalk: an IoT service platform

- D1 is a multi-sensor device called MorSensor
  - The sensors on a MorSensor device can be replaced easily by plugging/unplugging the slices of the sensor
  - Two IDFs:
    - Acceleration
    - Temperature
- D2 is a smartphone
  - Two IDFs:
    - Acceleration
    - Microphone



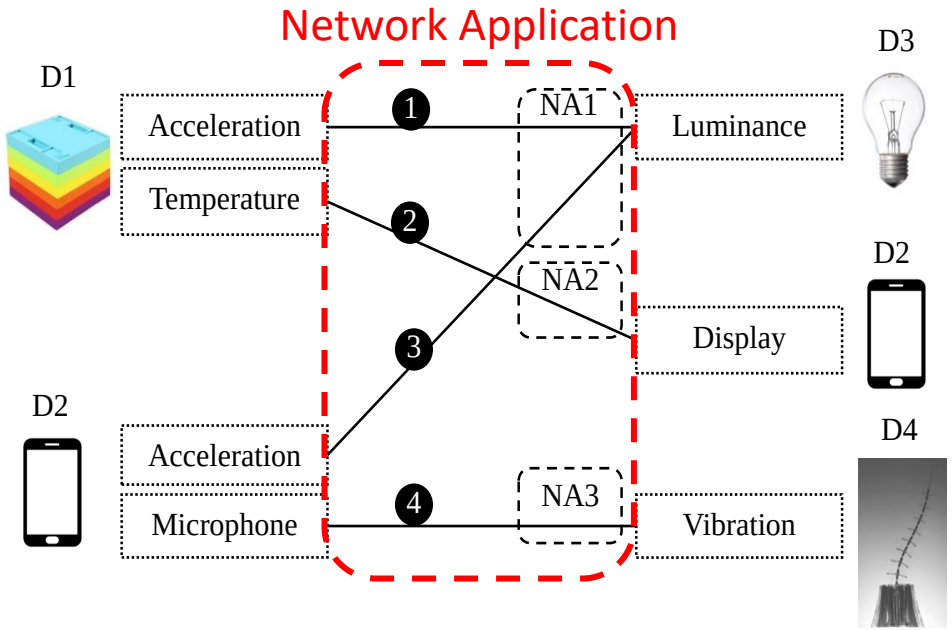
IoTtalk: an IoT service platform

- D2 also has an ODF:
  - Display
- D3 is a bulb
  - One ODF:
    - Luminance
- The Tail D4
  - One ODF:
    - Vibration
  - This special device is the Silver Medal Award artwork “Transparent Organ” in Salon International Des Invention [Huang2014]
  - The tail of this device wags based on the vibration strength received from its Vibration



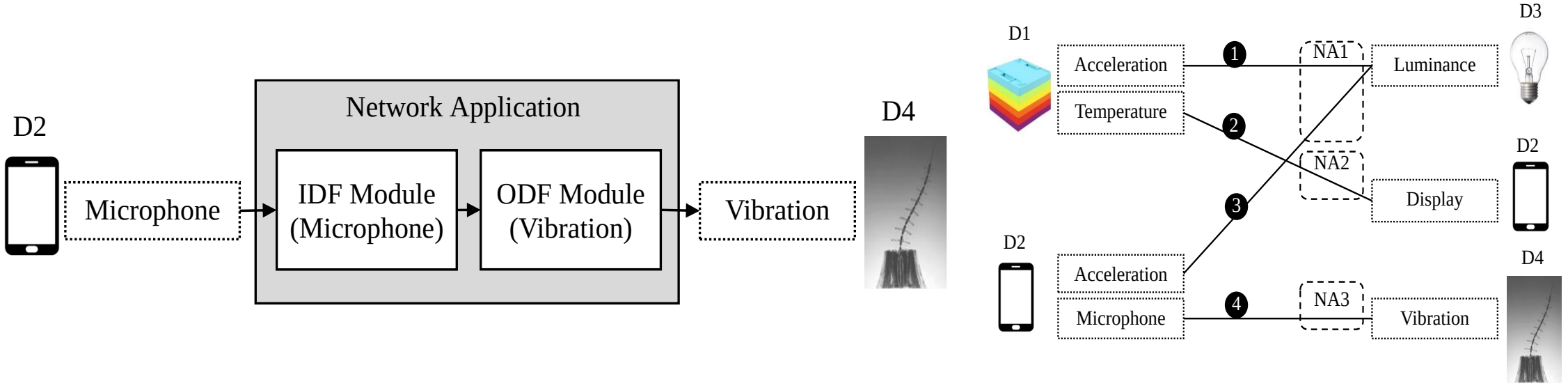
IoTtalk: an IoT service platform

- Lines (1)-(4) in Figure 1 illustrates how these IoT devices interact
  - A line connecting an IDF to an ODF represents interactions between the corresponding device features in input and output IoT devices
  - Such interactions are implemented in Python programs called network applications



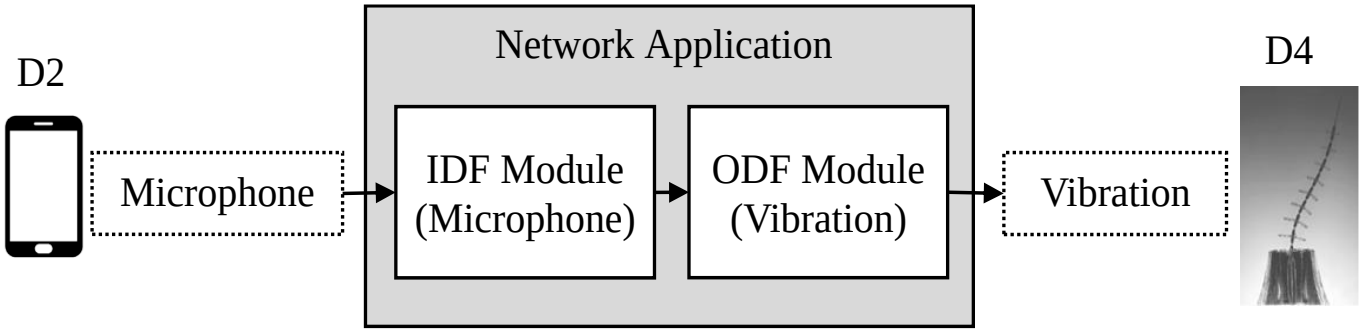
The IoTtalk Platform

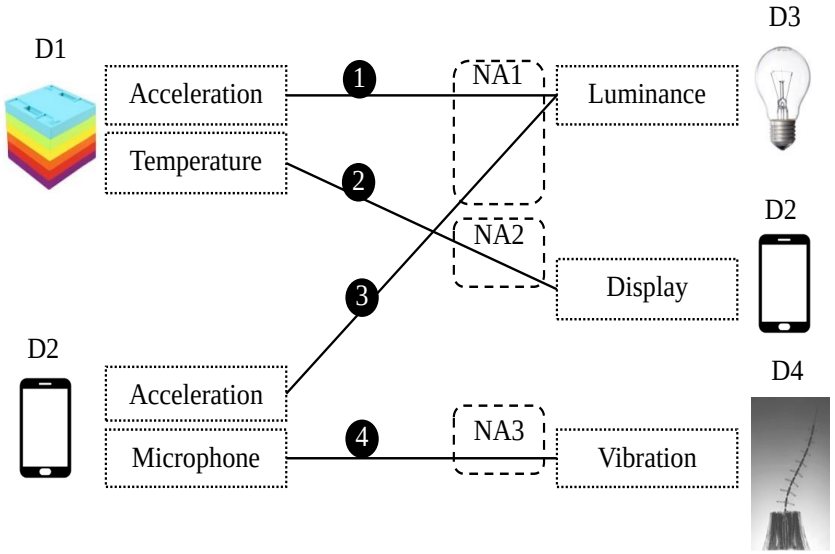
- If a network application handles the individual device features independently, then we can write a software module for each device feature, and the network application can be simply constructed by including these reusable DF modules
- For example, the building blocks for Line (4) in Figure 1 are shown in Figure 2, where the network application NA3 handles Microphone of D2 through the Microphone Module



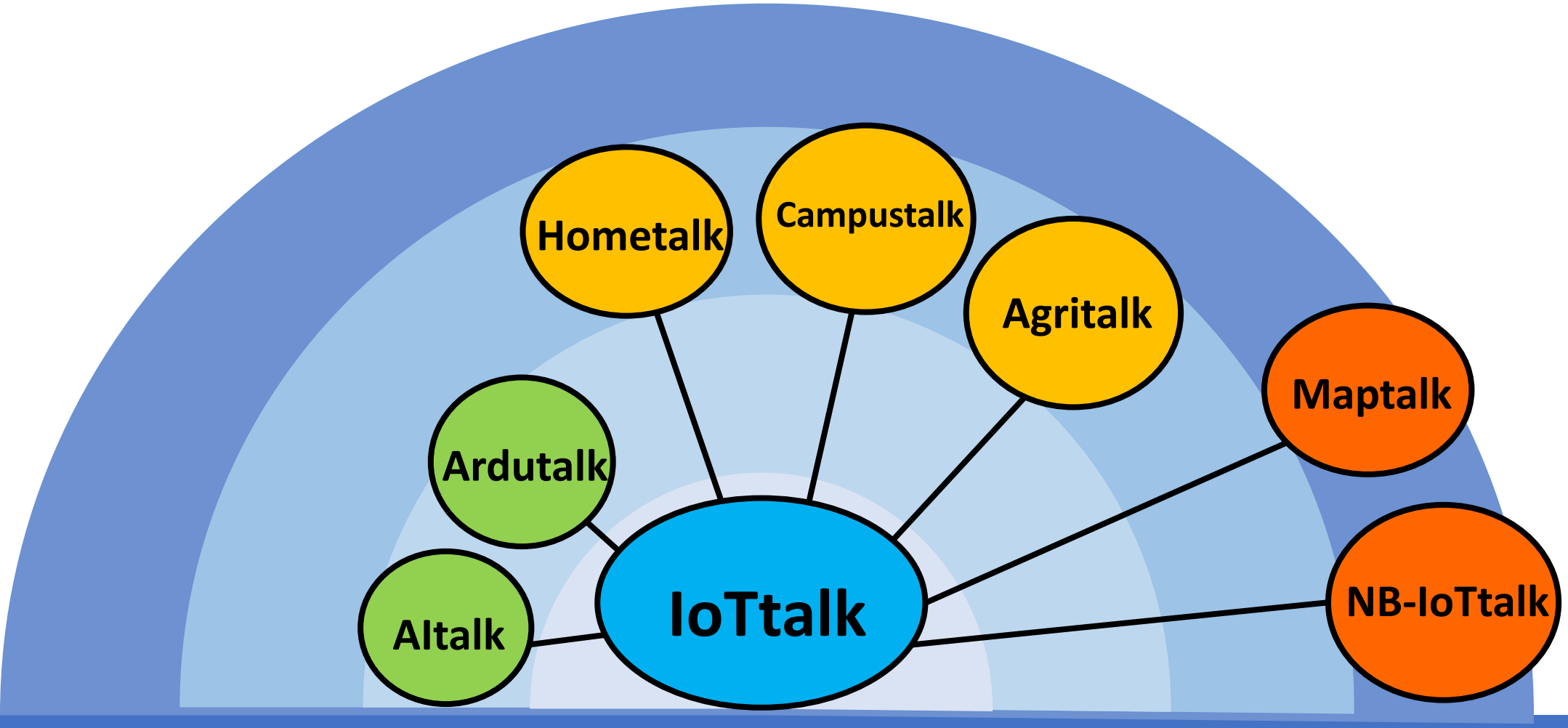
The IoTtalk Platform

- This IDF module computes, e.g., the volume of Microphone, and passes the result to the Vibration Module
- This ODF module translates the received value to the vibration intensity. Then NA3 outputs this intensity to the Vibration ODF to drive the vibration mechanism of D4





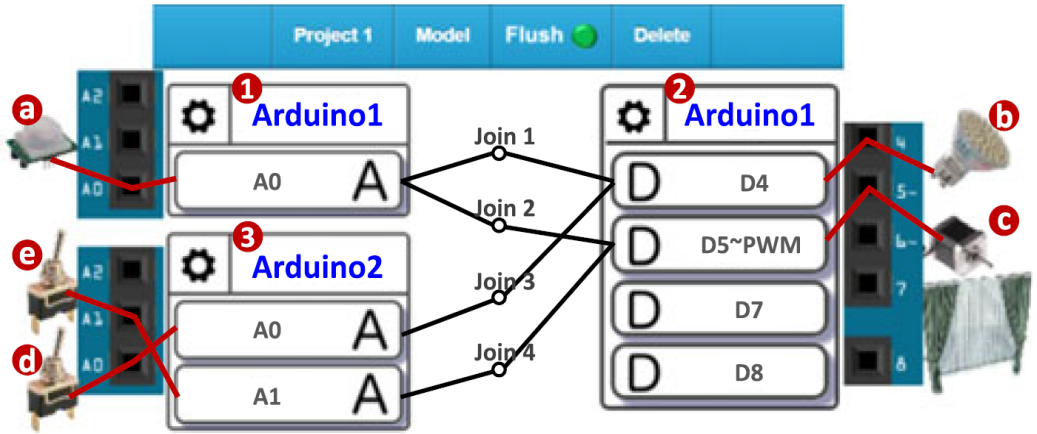
IoTtalk Applications



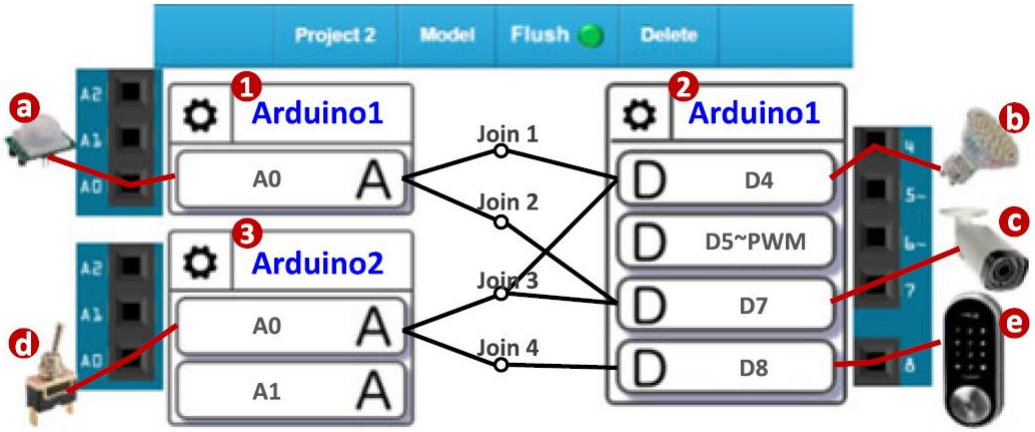
ArduTalk

- ArduTalk allows a user to arbitrarily link and relink sensors to actuators without or with little programming effort, and quickly generate Arduino applications for different purposes.

ArduTalk demo room



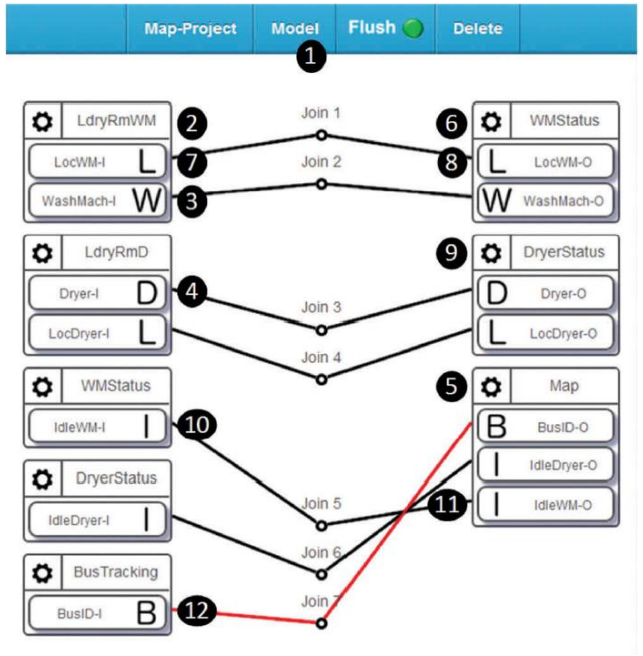
Security light application



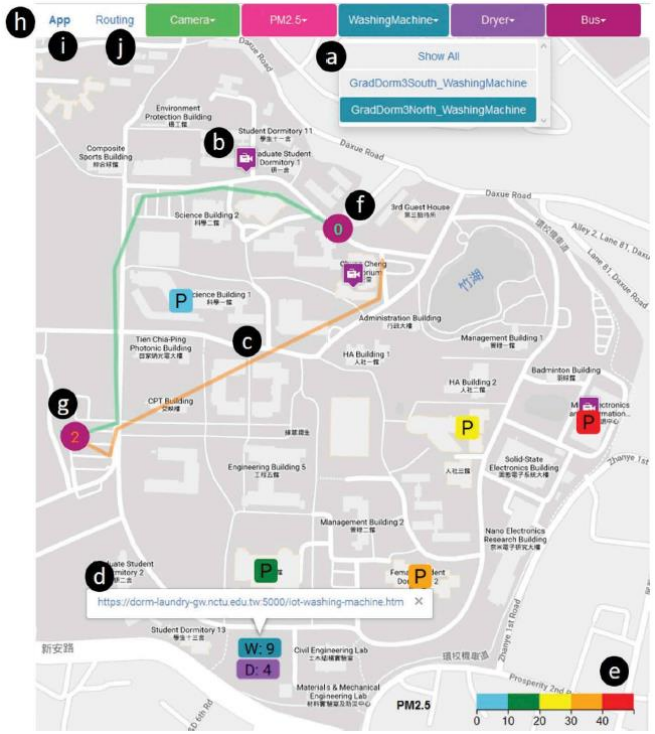
Door opening application

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



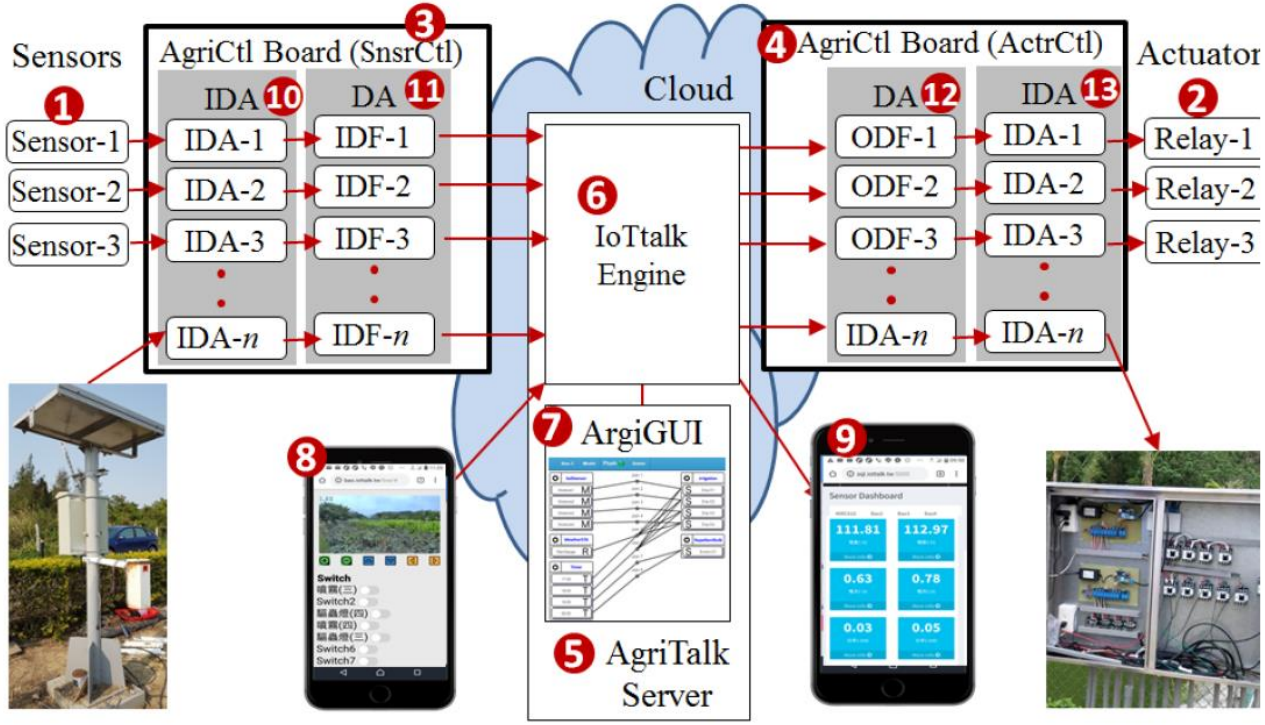
Laundry application



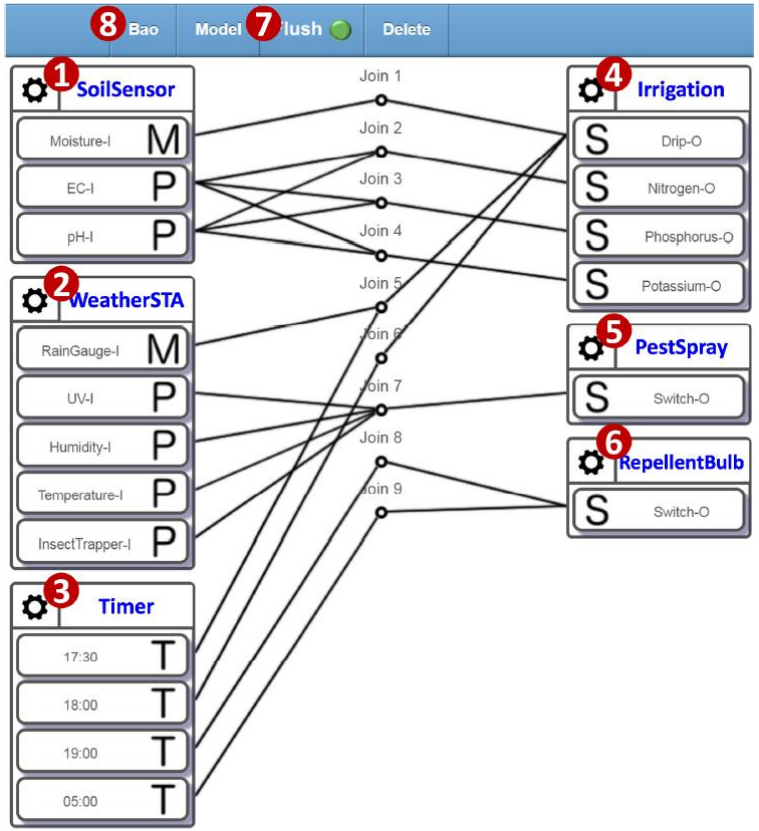
A 2D visual map based on MapTalk

AgriTalk

- AgriTalk, an inexpensive IoT platform for precision farming of soil cultivation



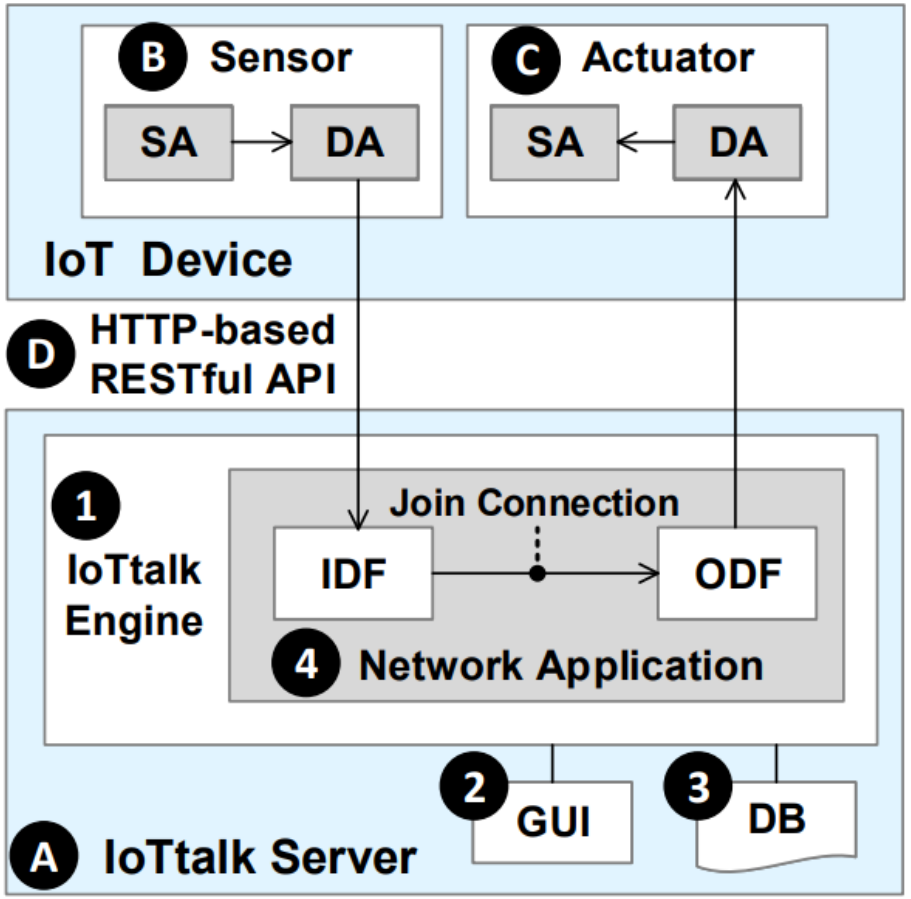
The AgriTalk architecture



Configuring precision farming

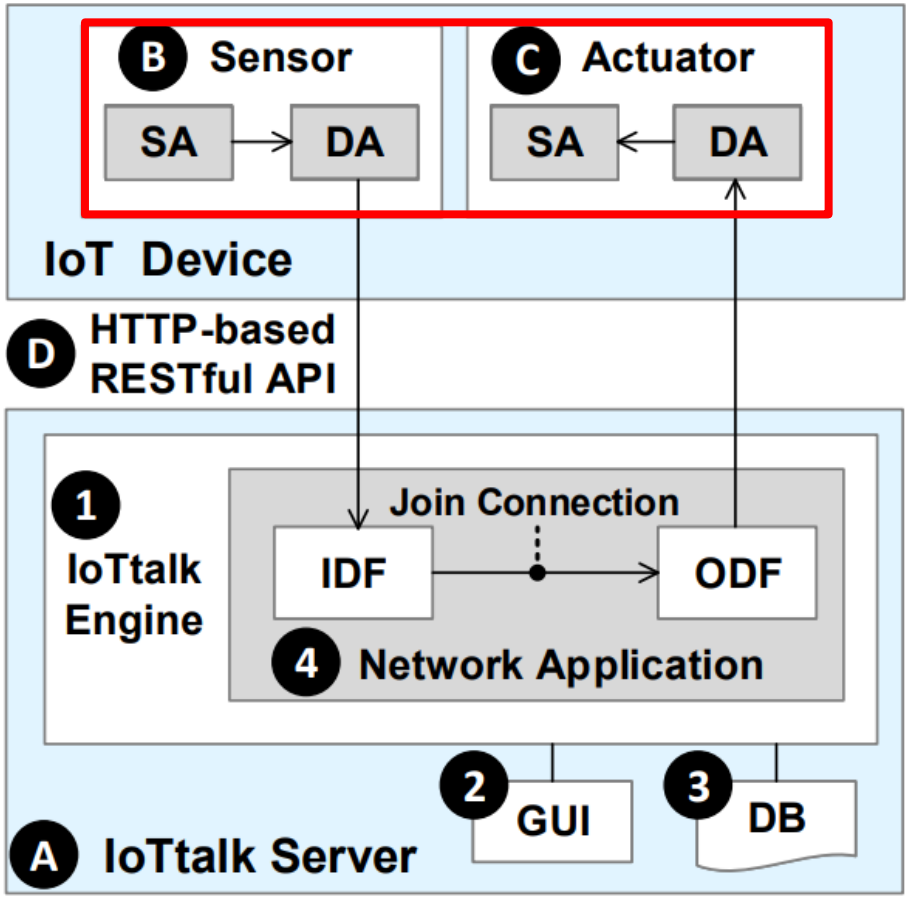
The IoTtalk Architecture

- Network domain (IoTtalk Server)
  - IoTtalk Engine
    - Network Application (NA)
  - GUI
  - Database
- Device domain (IoT Device)
  - Sensor: input IoT device
    - Collect sensing data
  - Actuator: output IoT device
    - Turn actuating results into actions



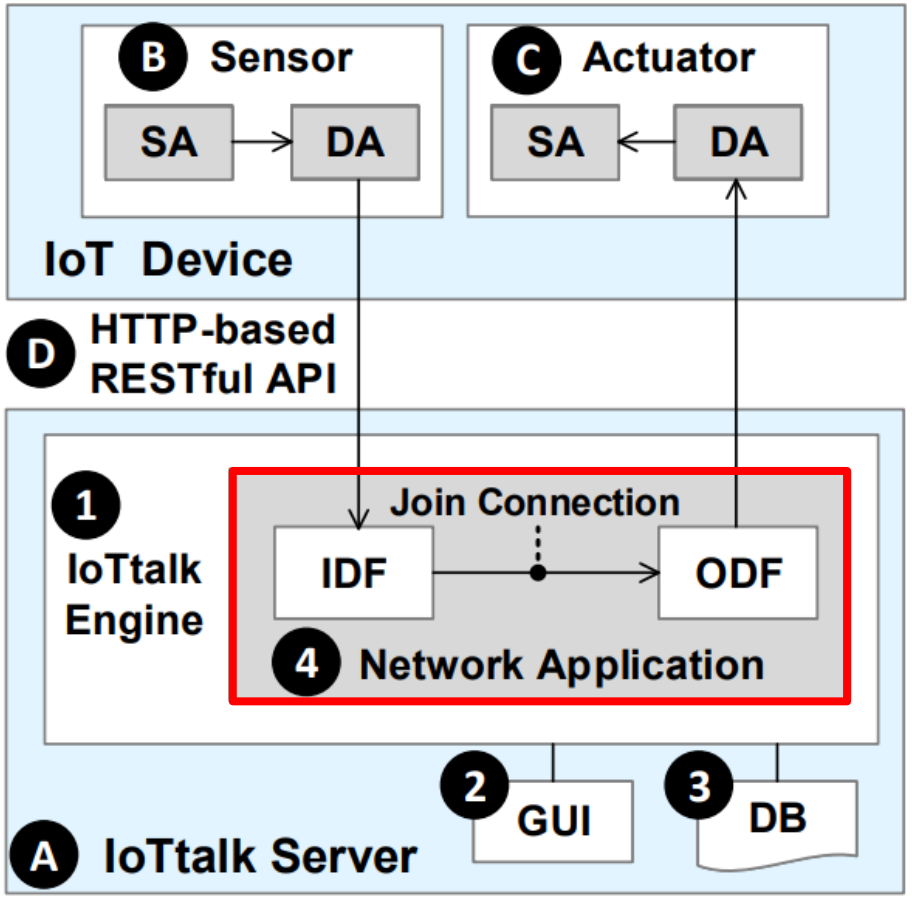
The IoTtalk Architecture

- IoT Device
  - Sensor/Actuator Application (SA)
    - **Sensor**
      - Compute the input sensing data
      - Transmit the data to the IoTtalk engine
    - **Actuator**
      - Receive the actuating result from IoTtalk engine
      - Trigger the corresponding actions
  - Device Application (DA)
    - Handle the HTTP-based RESTful APIs between the SA and the IoTtalk engine



IoTtalk: an IoT service platform

- IoTtalk Engine
  - Device Feature (DF)
    - Classify the IoT devices with their “capabilities”
      - **Input Device Feature (IDF)**
        - Temperature
      - **Output Device Feature (ODF)**
        - Display
  - Network Application (NA)
    - “Map” the IDFs to the ODFs
    - Join connection – service logic of the IoT application



Introduction to LineBot

Introduction to LineBot

- LineBot is a chatbot that can be used on Line
  - Analyze the messages from users
  - Give corresponding response messages based on keywords
- For example: restaurant recommendation LineBot
  - It can recommend restaurants based on user's preference
- LineBot is commonly used by many companies to respond customers' questions
  - Business hours
  - Address

Introduction to LineBot

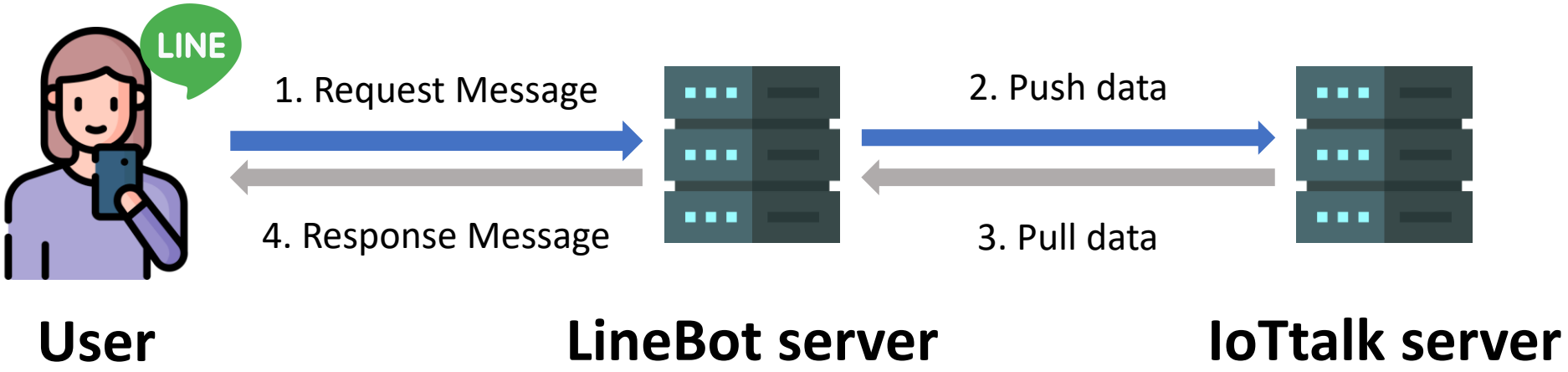
- Developers can design various LineBot according to their own requirements
- Operation procedure of LineBot
  1. User sends the message to Line platform
  2. Line platform transmits the message to the LineBot server
  3. LineBot server processes the message and sends it back to the Line platform
  4. Finally, Line platform response the message to user

Mini project 4

Project description

Introduction

- Design an IoT-based LineBot Application with IoTtalk server
  - Application deployment on IoTtalk
  - Build LineBot
    - LineBot server deployment on your local machine
- You can design desired service based on your scenario



Package

- **linebot_echo.py**
  - LineBot’s sample code for echo response
- **csmapi.py/DAN.py**
  - Some useful APIs about IoTtalk connection
  - Don’t need to modify them
- **app.py**
  - You can follow the comment to modify this file

Spec

- Design the **scenario**
  1. Reply：the status of machines, the information of something, the location of something, etc.
  2. Recommendation： Restaurants, Movies, Attractions, etc.
  3. Other：Game …
- IoTtalk
  - Design your own **Device Model** (at least one)
- LineBot
  - Your LineBot need to interact with IoTtalk(send/receive data to/from IoTtalk)
  - You can modify ‘**app.py**’ to finish mini project 4

Grading Policy - 1

- LineBot (45%)
  - (5%) IoTtalk Devices Registration
  - (5%) Push data to IoTtalk(IDF)
  - (25%) Handle message based on your scenario
    - ➢ Scenario：Building a restaurants recommendation, inputs may be “Chinese”, “Korean”, “American”, etc.
      1. Simple reply (10%)
         - ➢ Reply always be the same restaurant.
      2. Creative reply (25% at most)
         - ➢ Reply will be different restaurants based on certain input.
  - (5%) Pull data from IoTtalk(ODF)
  - (5%) Get result from LineBot App successfully

Grading Policy - 2

- Report (40%)
  - (5%) Scenario Description
  - (10%) How to Design
  - (15%) Screenshots
    - IoTtalk GUI connection, DM/DF creation
    - IDF/ODF Monitor
    - LineBot result
  - (10%) What you learn
- Demo (15%)
  - Introduce your project
  - Demonstration: LineBot and input/output data on IoTtalk
  - Explain your program
  - Reservation form

Submission

- Please upload your mini project 4 to eLearn: **Deadline: 2023-01-17(Tue.) 23:59**
- Your project must be named as follows:
  - Program
    - <Student_ID>_project4.zip
      - csmapi.py
      - DAN.py
      - <Student_ID>_project4.py (This file is modified from '**app.py**')
  - Report
    - The report filename must be "<Student_ID>_project4.pdf"
- Plagiarism Avoidance: Discussion is encouraged. However, plagiarism is not allowed.

  We will use, e.g., "Moss" for similarity comparison and 0 points will be given if plagiarism.

IoTtalk Server

- The IoTtalk project should be named as "**Your_Student_ID**"
- **IoTtalk server**
  - **Student ID: 110061615~110137504**
    - http://140.114.77.73:9999/
  - **Student ID: 111062629~111064558**
    - http://140.114.77.92:9999/
  - **Student ID: 111065503~111164521**
    - http://140.114.77.93:9999/

Note

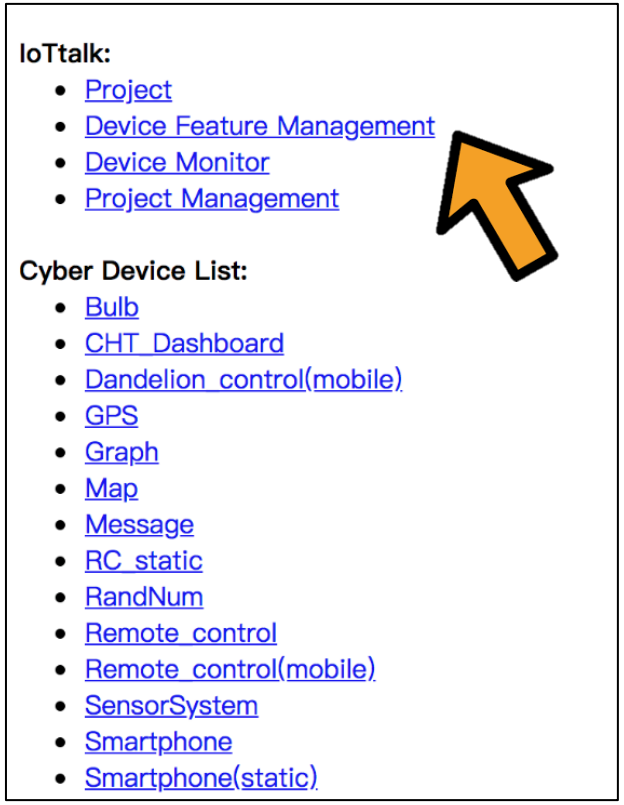
- Please be careful that not to delete the device information of other people
- It is recommended to back up the information you set on IoTtalk
  - E.g. join function setting, device setting…

Mini project 4

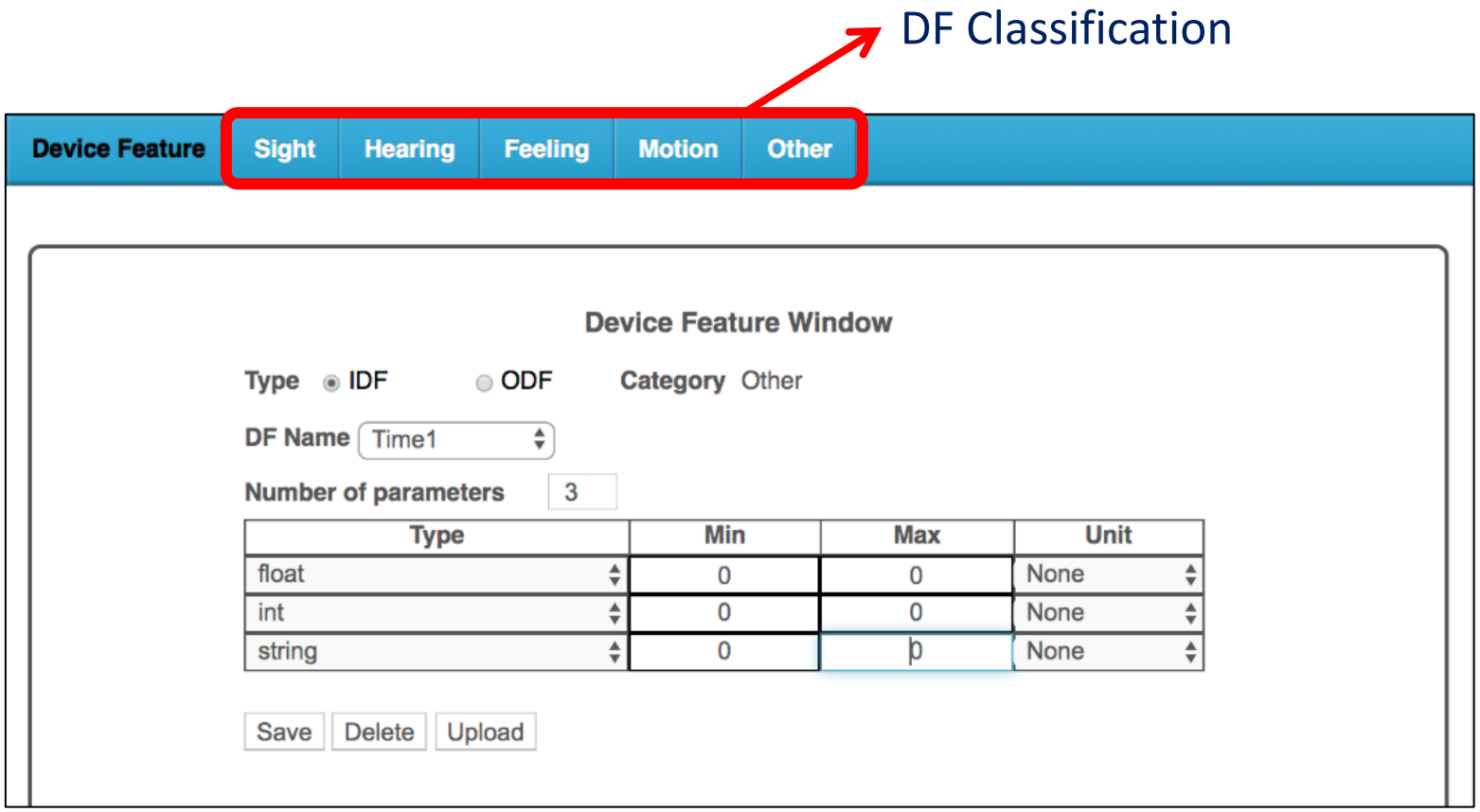
Configuration and application deployment on IoTtalk

Device Feature Management – Device Feature

- Enter '**Device Feature Management**' from IoTtalk Homepage
- We can define the new **DF** in Device Feature Management page



IoTtalk Homepage

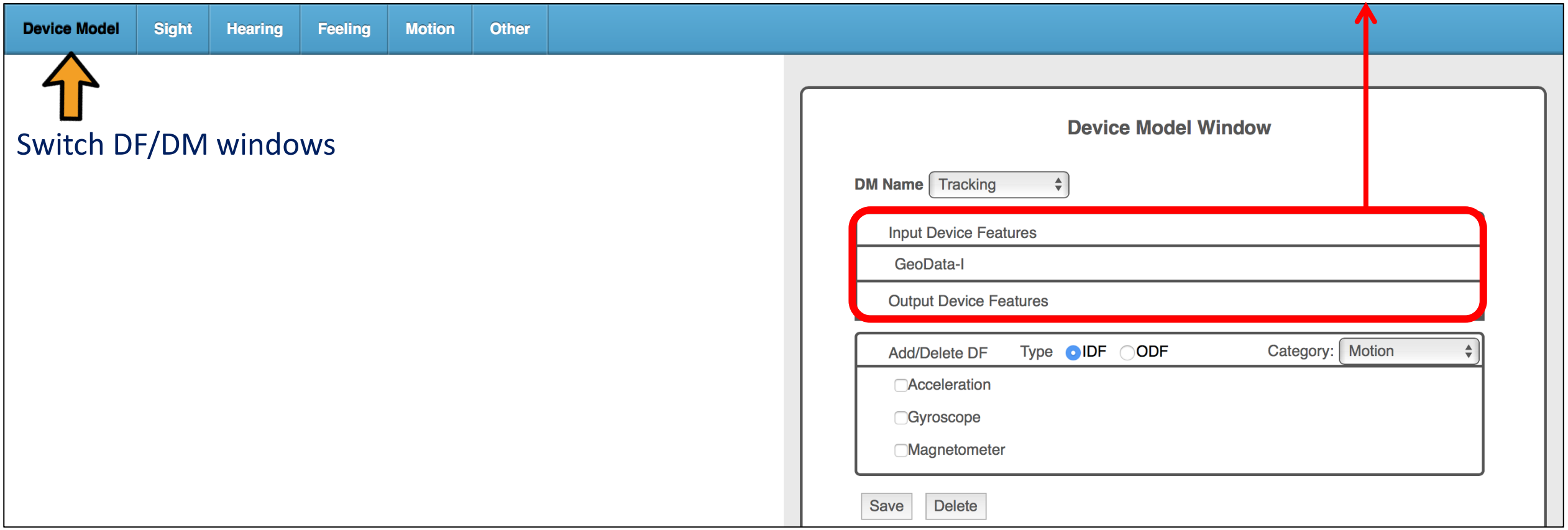


Device Feature Management page - DF

Device Feature Management – Device Model

- We can also define the new **DM** by adding existing DFs in Device Feature Management page
- After saved, the DM can be used in the IoTtalk project

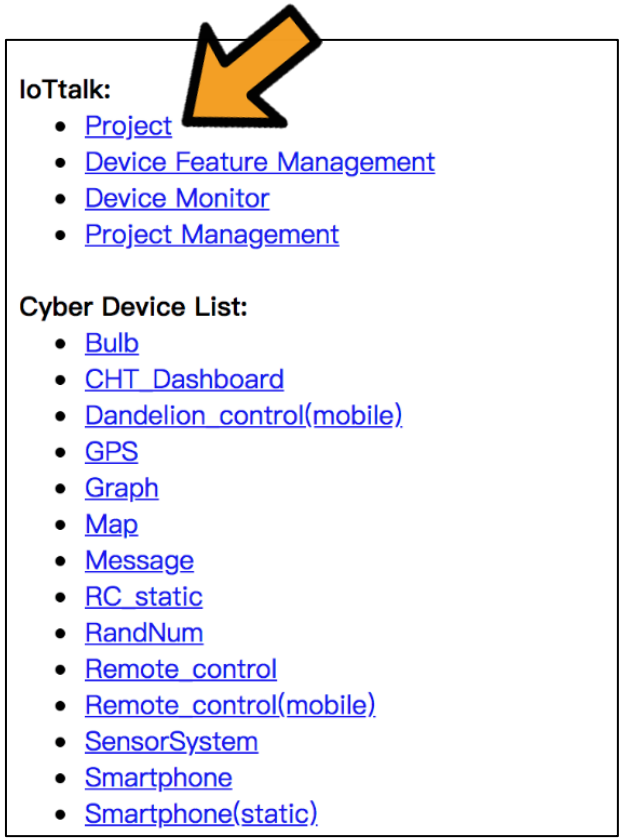
DFs list in DM"Tracking"



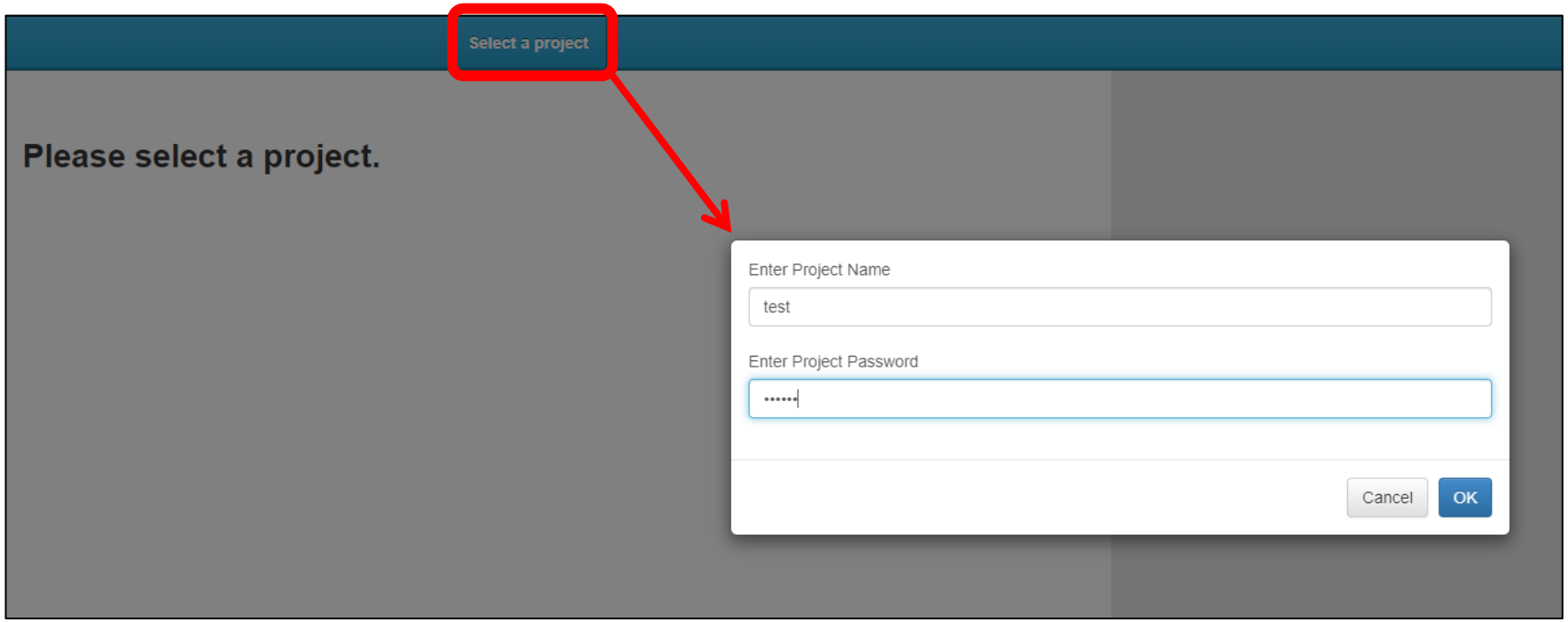
Device Feature Management page - DM

IoTtalk Project Creation

- Enter '**Project**' from IoTtalk Homepage
- Create your own IoTtalk project



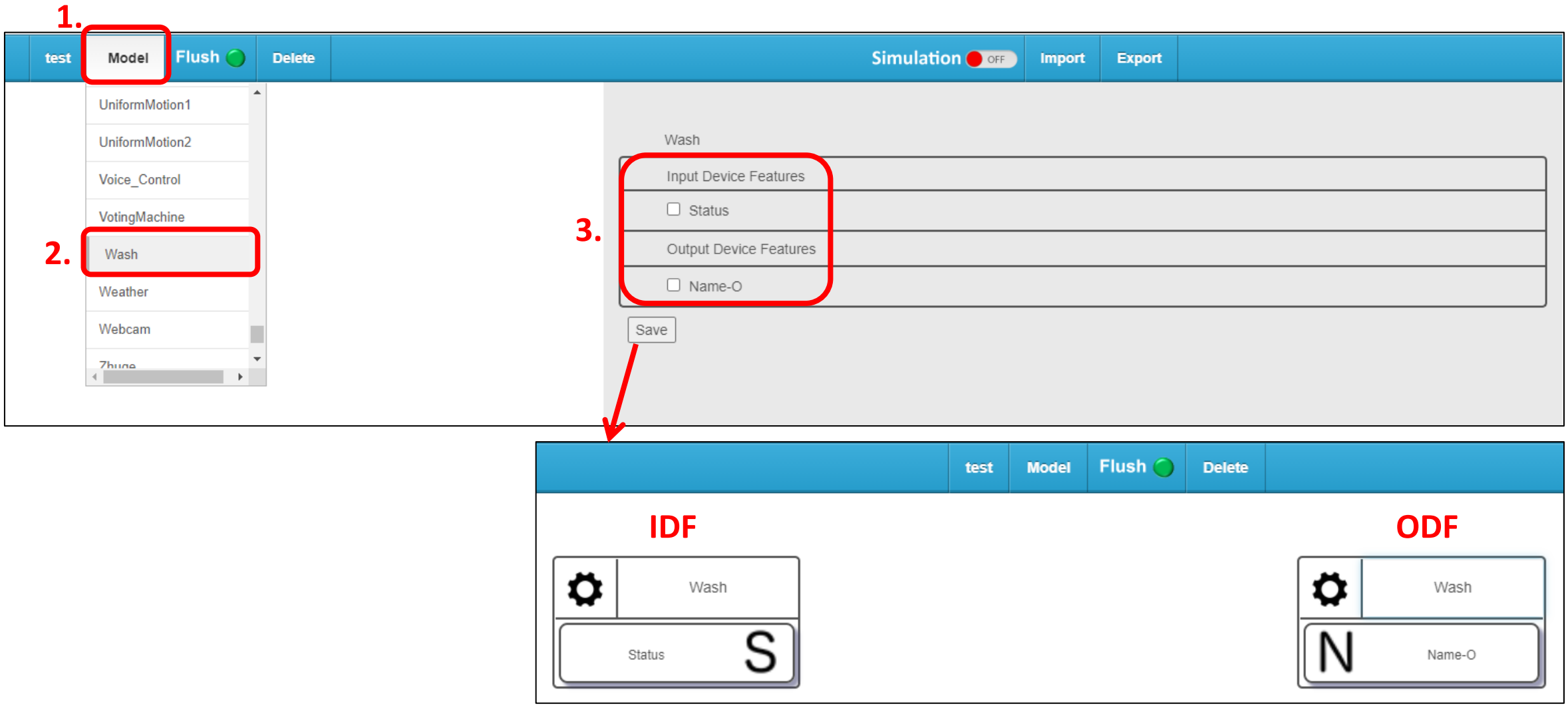
IoTtalk Homepage



Project page

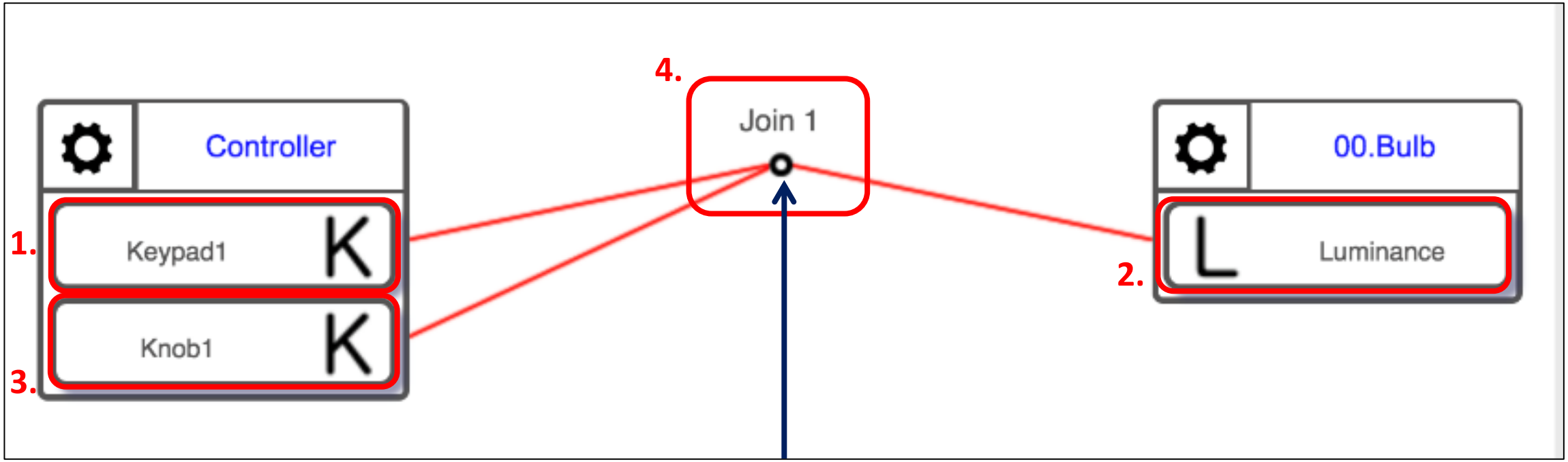
Project Design

- Add those DMs you need in the project



I/O Connection

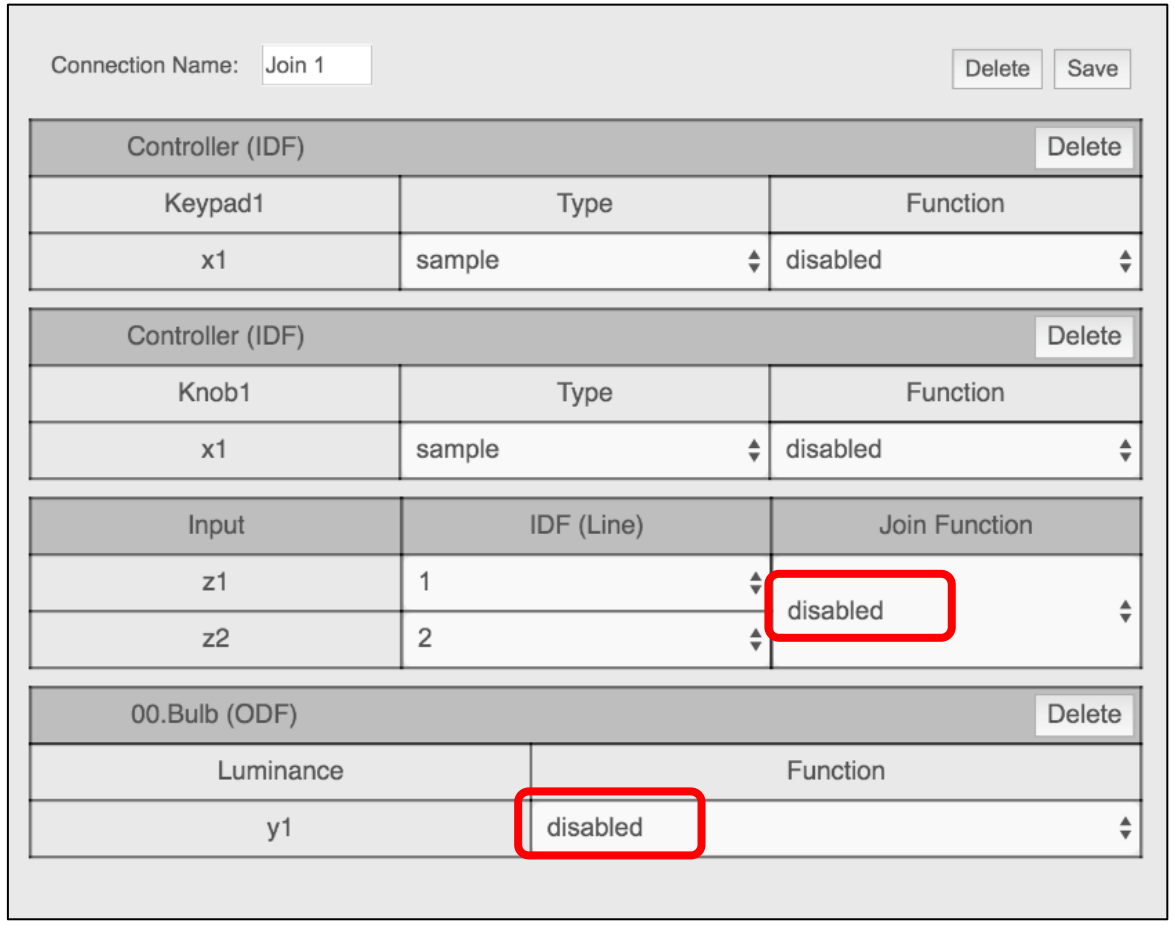
- Add those DMs you need in the project
- Click the DF you want to link, and a red line will appear between the two sides (Join 1)



Left click to show function setting window (next slide)

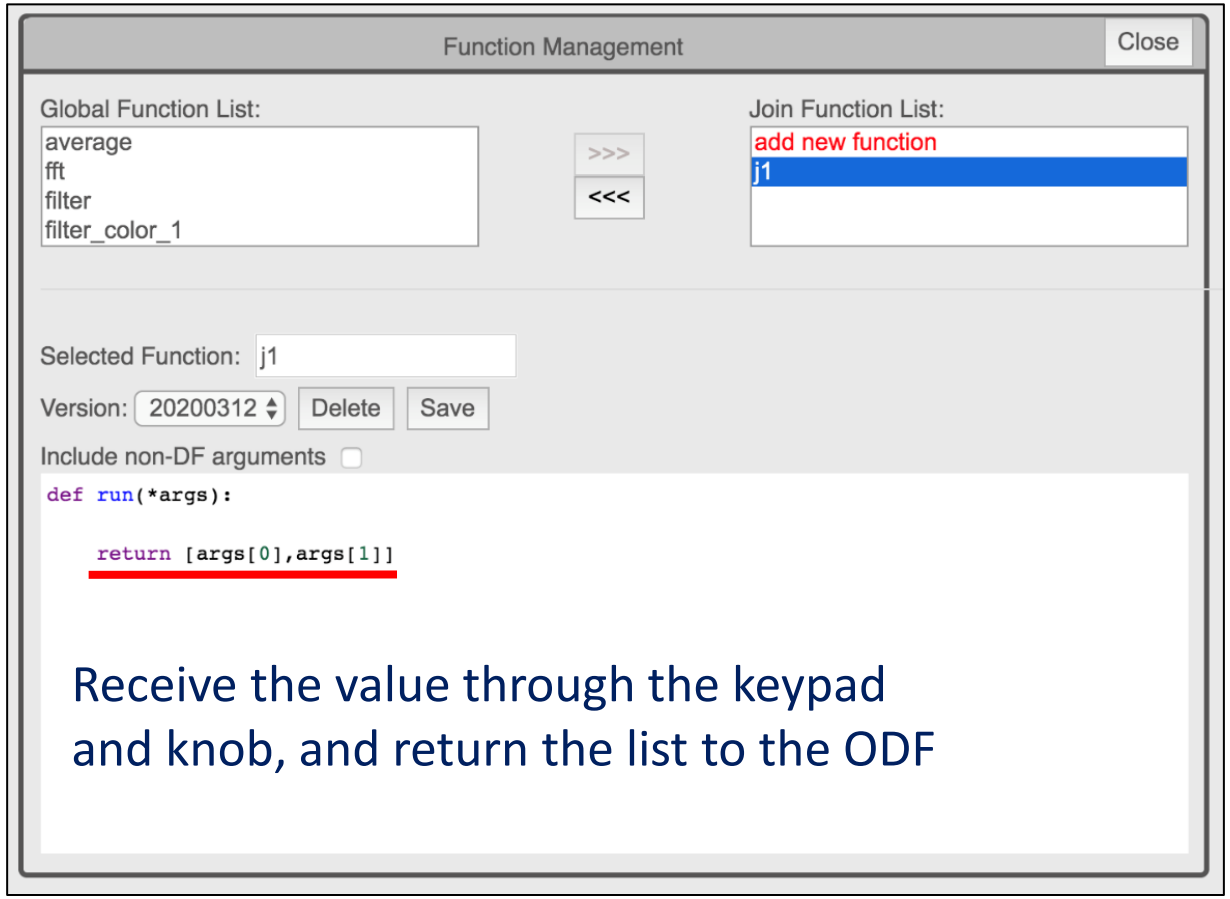
Function Setting(1/3)

- Join Function
  - If there are more than one IDF, this function can set up a connection between IDFs and ODF
- IDFs/ODFs Function
  - Design the required service logic for each IDF/ODF
- Design the functions (red box)
  - Drop down the combobox
  - Click “add new function”

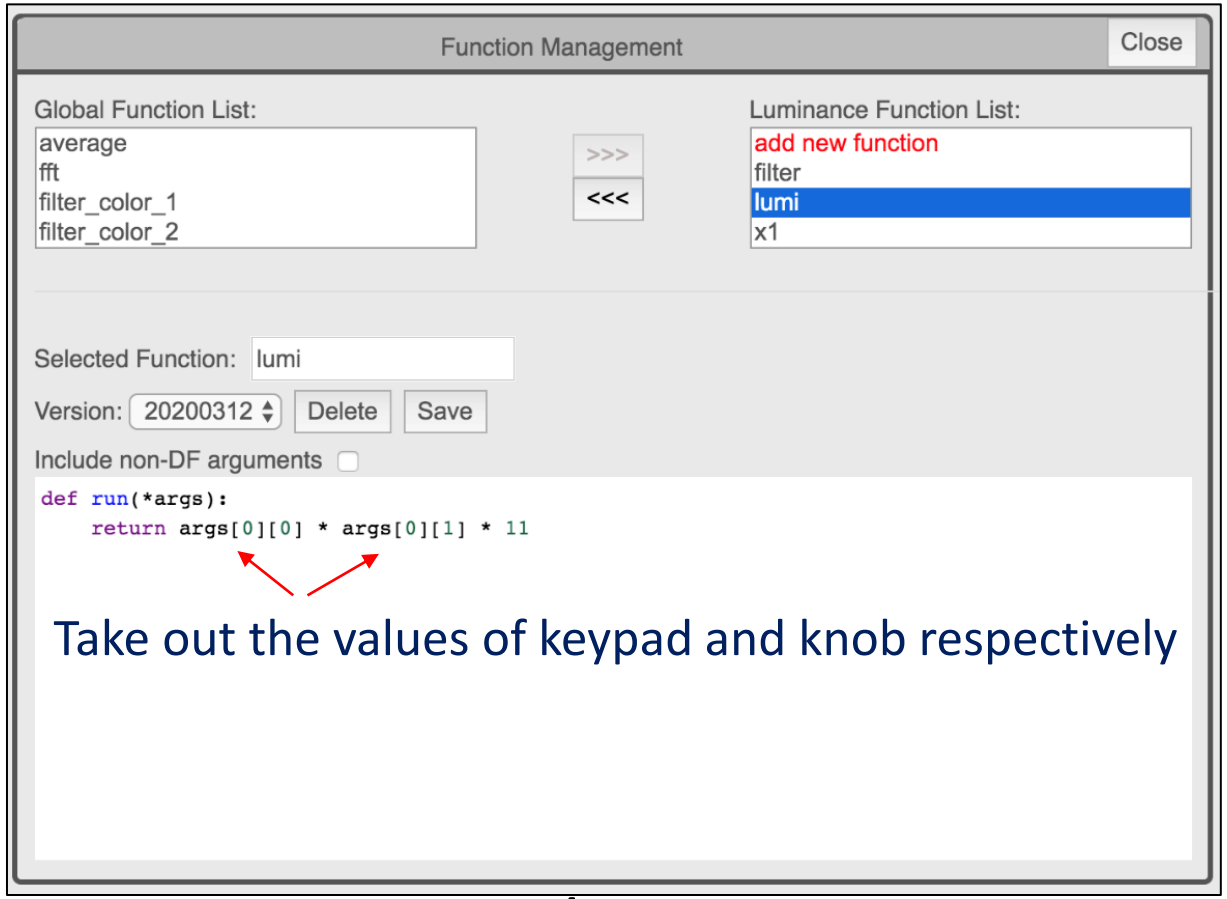


Function Setting(2/3)

- You can use the built-in basic functions directly, or you can define your own
- Take Controller <--> Bulb on page 7 for example



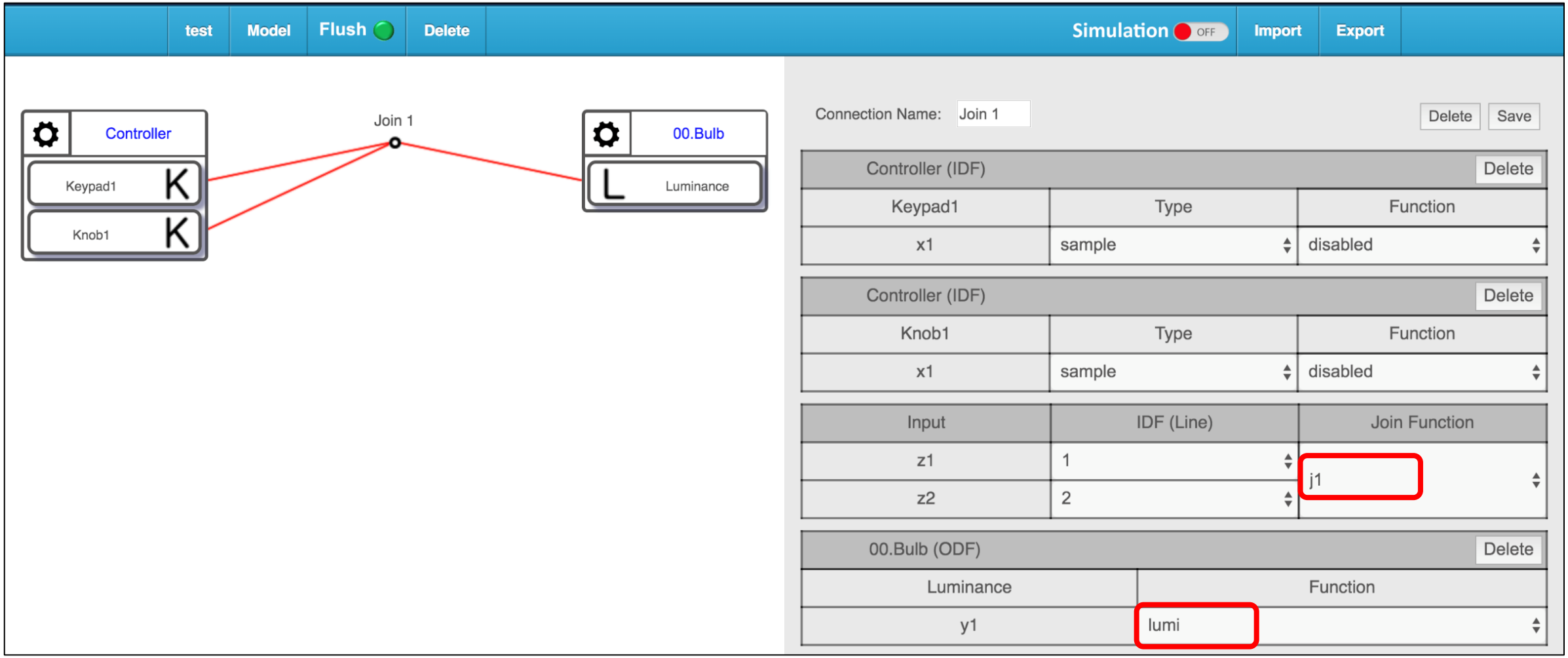
Join Function



IDF/ODF Function

Function Setting(3/3)

- After the new function is saved, it can be selected from the combobox



app.py

- Modify IoTtalk_registration function to complete device registration

Connect to IoTtalk server

Define device profile

Register/Deregister device to IoTtalk

```
def IoTtalk_registration():

        ServerURL = 'http://IoTtalk Server IP:9999' #with non-secure connection


        DAN.profile['dm_name']='Wash’
        DAN.profile['df_list']=['Status','Name-O’]
        #DAN.profile['d_name']= 'Assign a Device Name’

        DAN.device_registration_with_retry(ServerURL, None)
        #DAN.deregister() #if you want to deregister this device, uncomment this
        line #exit() #if you want to deregister this device, uncomment this line
```

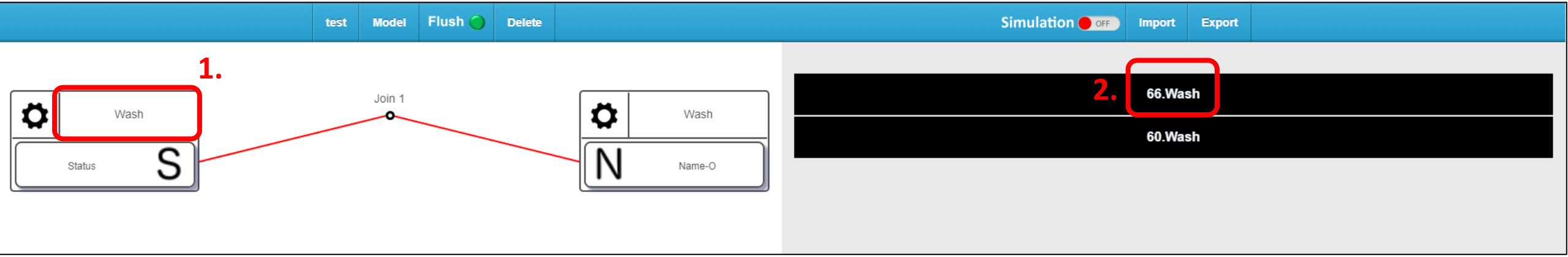
app Execution

- **Execute app.py**
  - ***python app.py***
- Information from terminal
  - **Device name = 66.Wash**
  - The terminal will display the number of registered Device

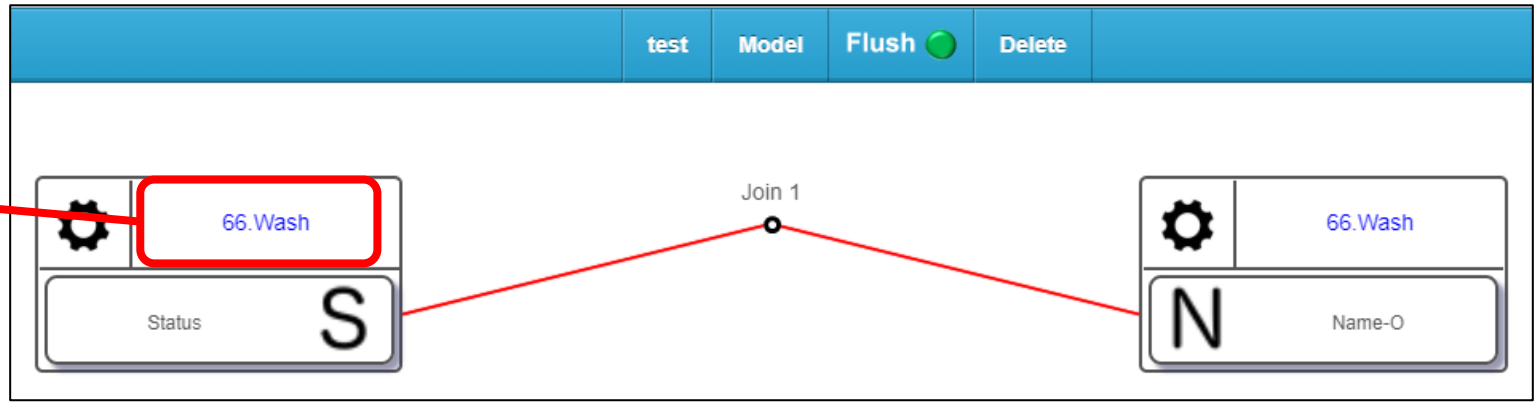
```
Last login: Wed Mar 25 11:13:55 on ttys000
(base) wmnetde-MBP-3:~ jenny$ cd Desktop/
(base) wmnetde-MBP-3:Desktop jenny$ python app.py
IoTtalk Server = http://140.114.77.73:9999
This device has successfully registered.
Device name =  66.Wash
Create control threading
```

Device Binding

- After app.py is executed, the registered device will appear on the right window, and then bind the devices
- **Remember to correspond the device number**



After binding,
the text will be displayed in blue



Other - Monitor

- Click the right button at the Join point
  - can observe the input and output data

IDF Monitor

Sub-stage: Input | Continue | Next | Table | 1 Keypad1

Timestamp	x_1
14:11:46	9.00
14:13:52	6.00
14:14:12	2.00

Multiple Join Monitor

Function | Table

Timestamp	z_F
14:11:46	[9, 0.7751671174611]
14:14:03	[6, 1]
14:14:12	[2, 1]
14:14:15	[2, 0.44487956377575016]
14:14:16	[2, 0.1528807905905573]

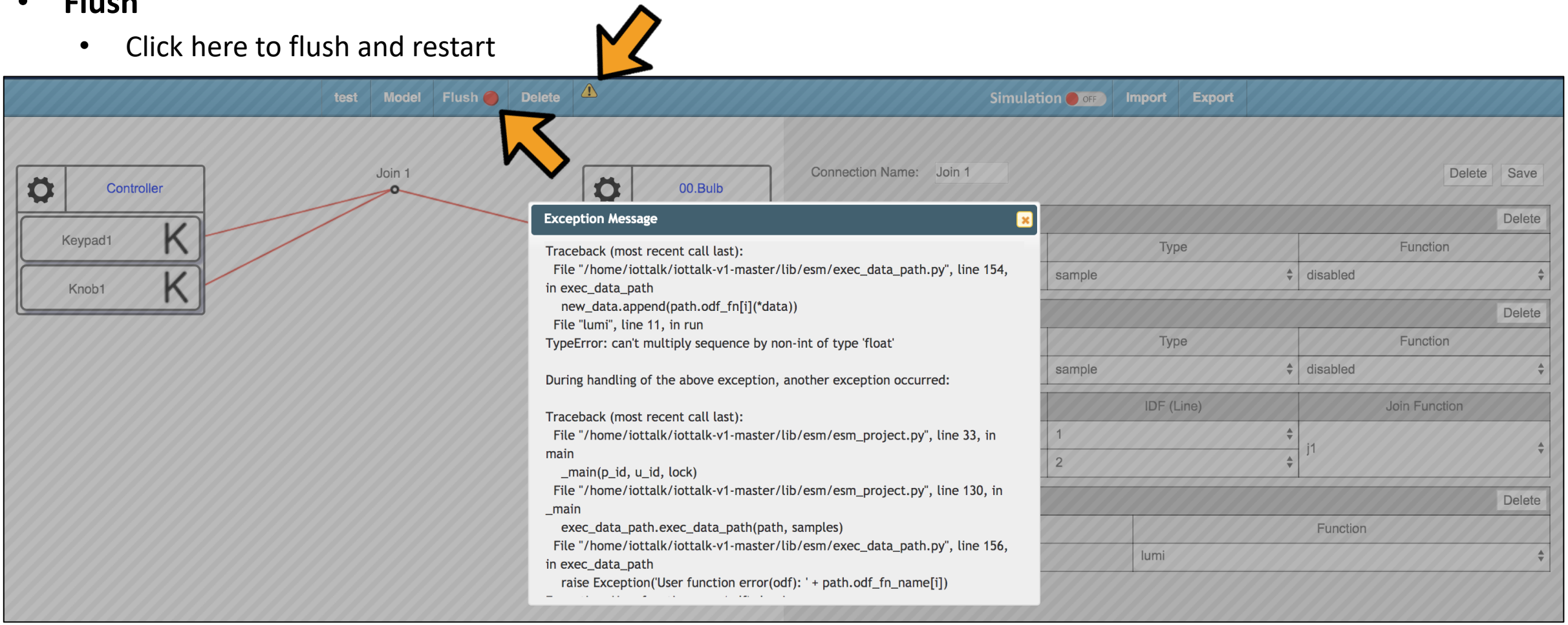
ODF Monitor

Sub-stage: Function | 1 Luminance | Table

Timestamp	$y_{1,F}$
14:11:46	76.74
14:14:03	66.00
14:14:12	22.00
14:14:15	9.79
14:14:16	3.36

Other - Exception message & Flush

- **Exception message**
  - If a warning appears during execution, it should be a mistake in your code.
- **Flush**
  - Click here to flush and restart



Mini project 4

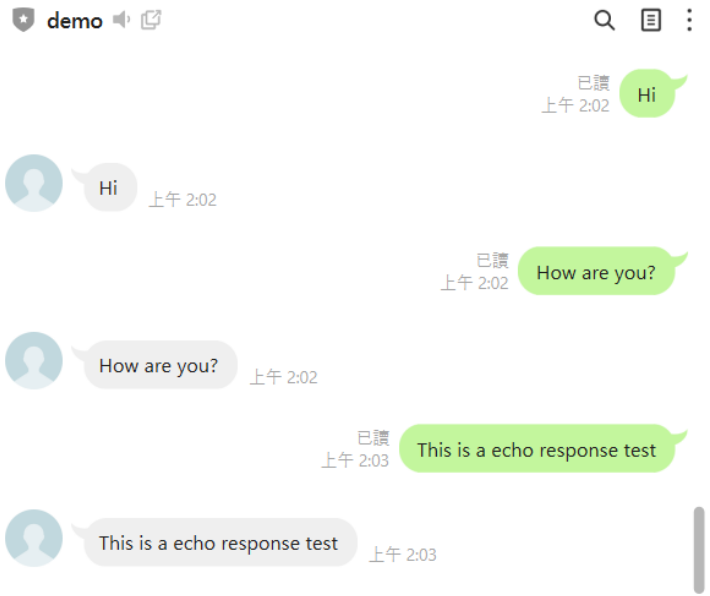
LineBot tutorial

LineBot Application

- **Building a LineBot with Ngrok**
- **Ngrok**
  - Ngrok provides a service that helps developers share sites and apps running on their local machines or servers
- **Preliminary**
  - Line Account (https://developers.line.biz/en/)
  - Ngrok (https://ngrok.com/download)
    - You need to sign up and follow the instruction on the download page to add authtoken

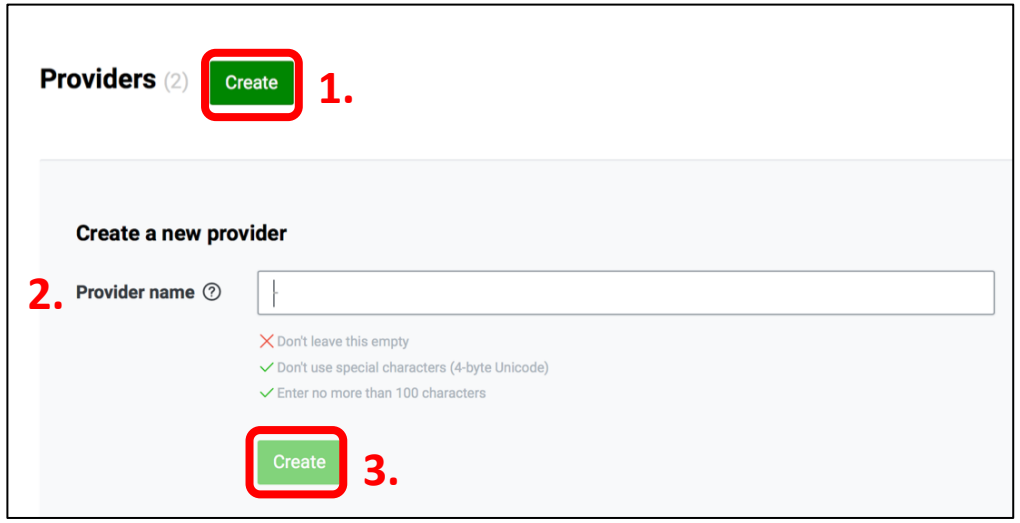
LineBot Sample Package – echo response

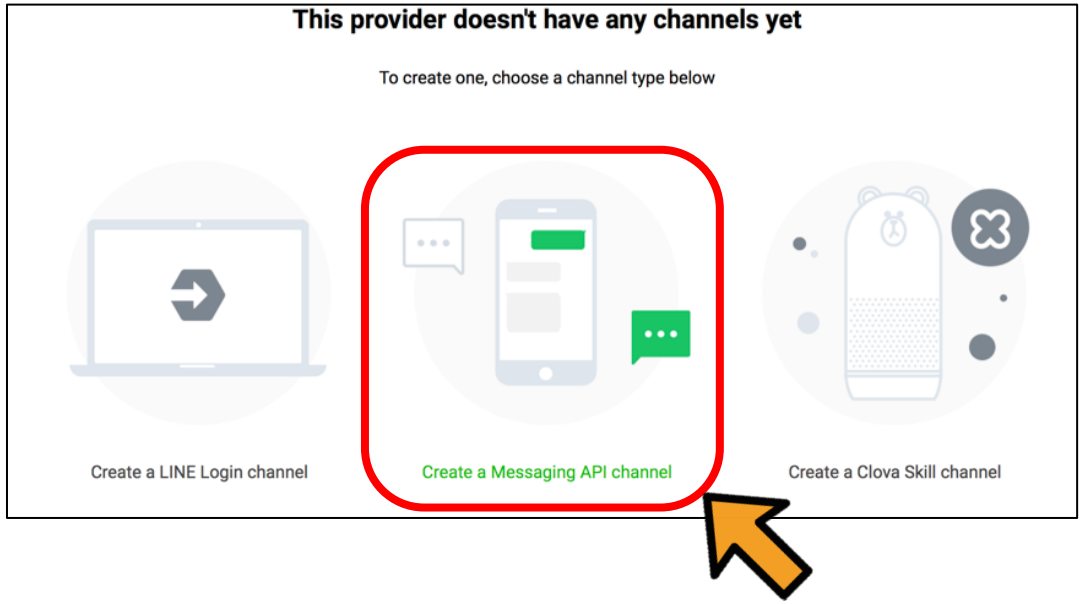
- **Install packages** :
  - flask: pip install flask
  - line-bot-sdk: pip install line-bot-sdk
- **linebot_echo.py**
  - The file is a sample code for echo response
  - The function handle_message() is used to control the message reply



Create a LineBot Channel

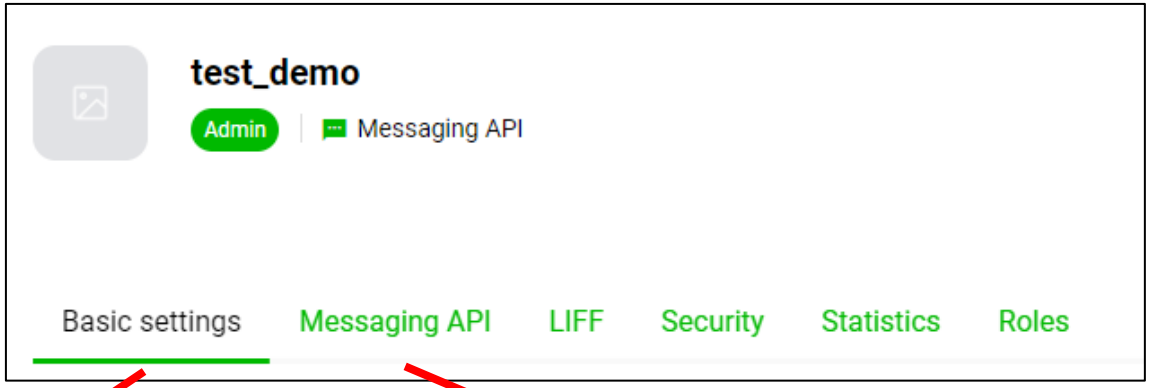
- Enter the Line Control Console (with your Line Account)
- Create a provider
- Choose "Create a Message API channel"
  - Setting some information: Channel type, Provider, Channel name, Channel description, Category, Subcategory, Email address





Get Channel Information

- Record **Channel Access Token** and **Channel Secret**, which will be used in linebot implementation
- Get Channel Secret on **Basic settings** page
- Get Channel Access Token on **Messaging API** page



Execute LineBot program

- **Set Channel Secret and Channel Access Token in linebot_echo.py**

```
# Channel Access Token
line_bot_api = LineBotApi('YOUR CHANNEL ACCESS TOKEN')
# Channel Secret
handler = WebhookHandler('YOUR CHANNEL SECRET')
```

- **Execute linebot_echo.py:** python linebot_echo.py

```
hsuan@hsuan-VirtualBox:~/Mini Project 4/Mini Project 4 Package$ python linebot_echo.py
 * Serving Flask app "linebot_echo" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Port

Start a Ngrok tunnel

- **Start a tunnel :** ngrok http <YOUR PORT>

```
ngrok

Add Single Sign-On to your ngrok dashboard via your Identity Provider: https://ngrok.com/dashSSO

Session Status                online
Account
Version                       3.1.0
Region                        Japan (jp)
Latency                       50ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    [Copy link]                    -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              3       0       0.00    0.00    0.19    0.31

HTTP Requests
-------------

POST /callback                 200 OK
POST /callback                 200 OK
POST /callback                 200 OK
```

NOTE：Link will be updated while restarting this terminal

Message API Settings

- **Webhook Settings**
  - Webhook URL
    - https://{ngrok_link}/callback
    - Click “Update”, do not need to click “Verify”
  - Enable “Use webhook”