

Connectivity of Data Foundry

Workshop – IoT dataset

I-Tang (Eden) Chiang

i.chiang@tue.nl

d.Search Lab @ Atlas 2.125



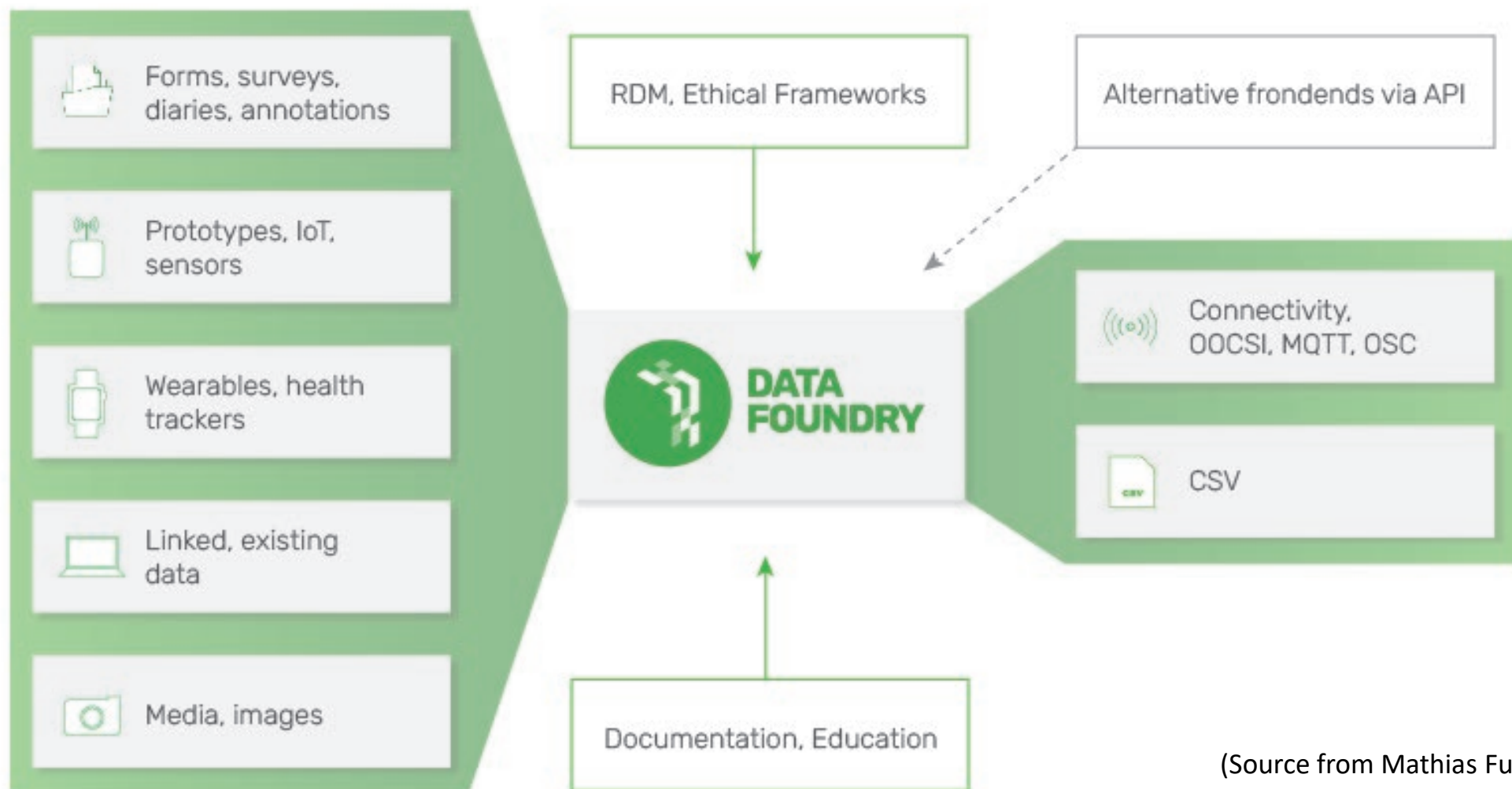


Outline

- About Data Foundry
- Why using Data Foundry
- Practices with IoT dataset and ESPs -
 - Data collection by Data Foundry through OOCSI
 - Data collection by Data Foundry via API (next workshop)



About Data Foundry



(Source from Mathias Funk)



Why using Data Foundry

- Safety
- Simplicity
- Community
- Well organized “Projects” structure
- Offers support for various courses such as:
 - Making Sense of Sensors (Bachelor)
 - Digital Craftsmanship (Bachelor)
 - Data-Enabled Design (Master)
 - Creativity and Aesthetics of data & AI (Master)

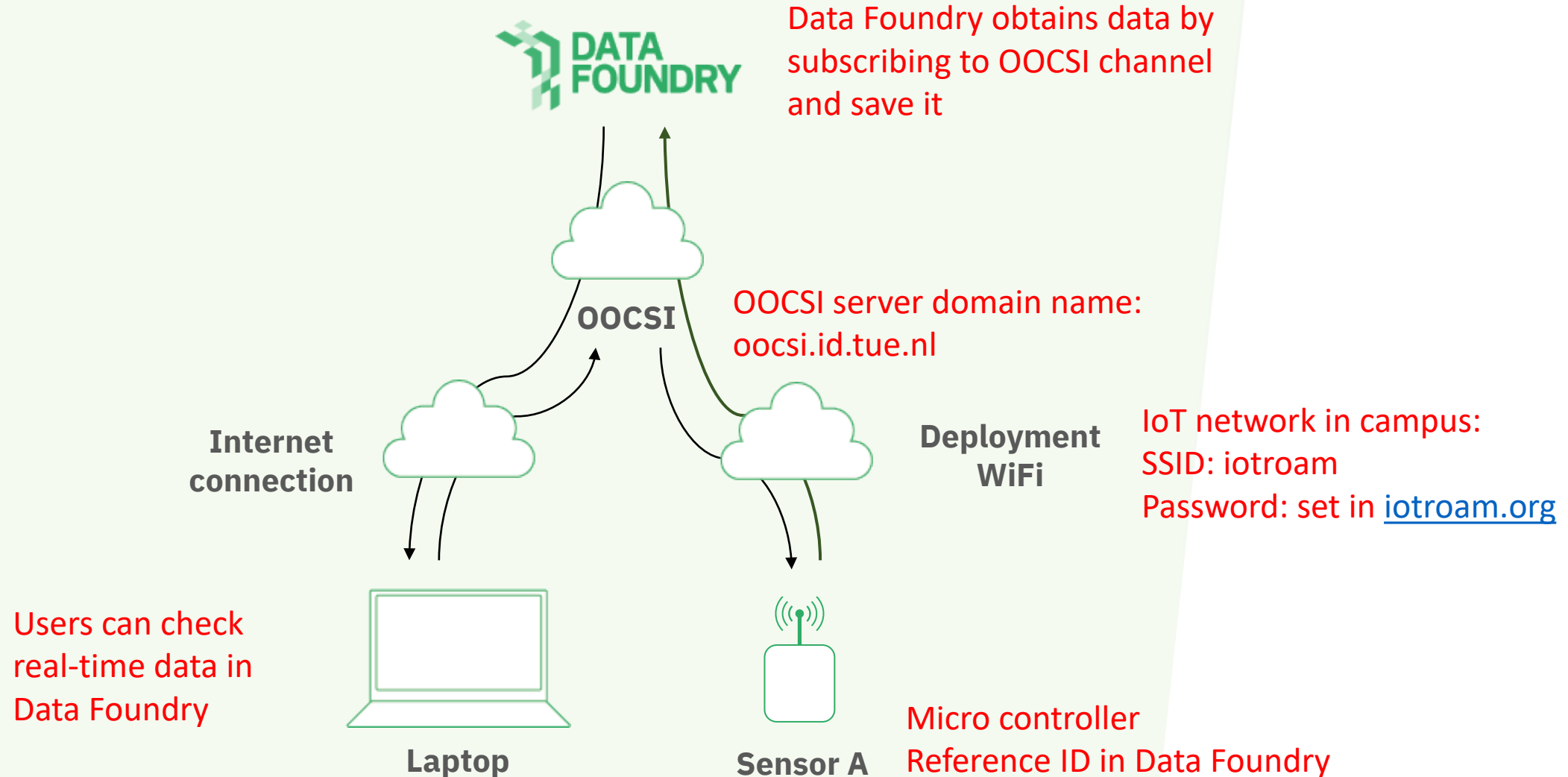


Practice: Data collection by Data Foundry through OOCSI

1. Preset of Data Foundry
 1. Create a project → Should be done by Data Foundry automatically
 2. Create an IoT dataset
 3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOCSI
4. Check data in OOCSI UI Client page
5. Save data to own IoT dataset



How Data Foundry and OOC SI work together



(Source from Mathias Funk)



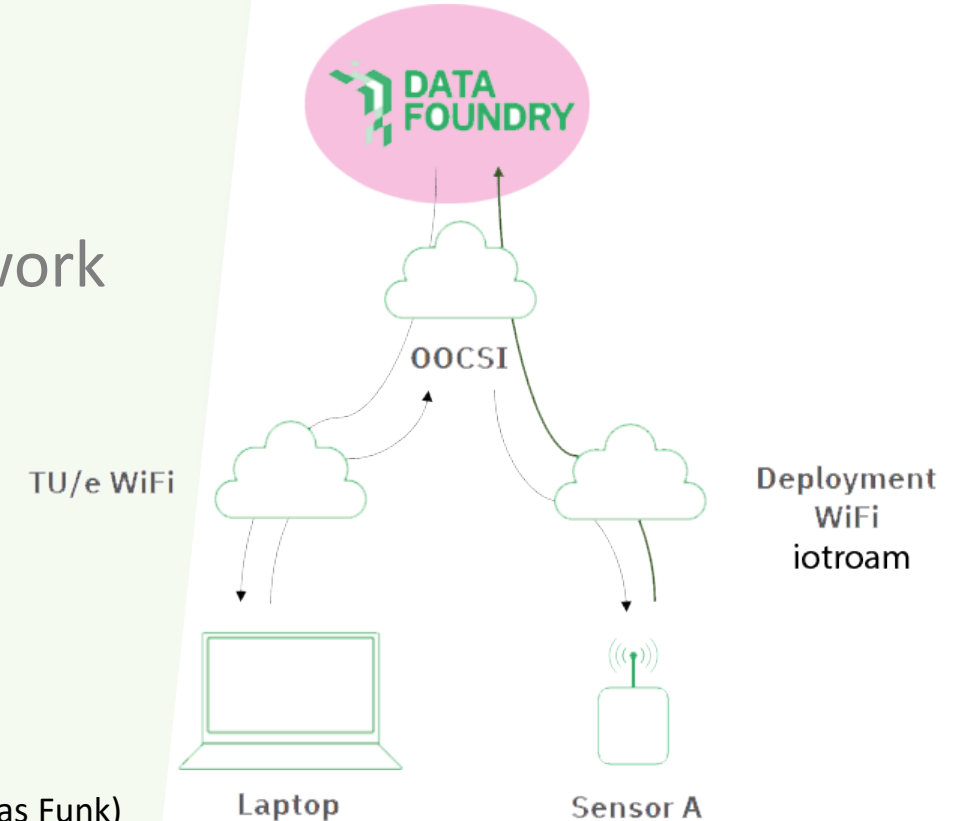
Practice: Data collection by Data Foundry through OOC SI

1. Preset of Data Foundry

1. Create a project → Should be done by Data Foundry automatically
2. Create an IoT dataset
3. Create a device

2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOC SI
4. Check data in OOC SI UI Client page
5. Save data to own IoT dataset

(Source from Mathias Funk)





Practice: Data collection by Data Foundry through OOCSI

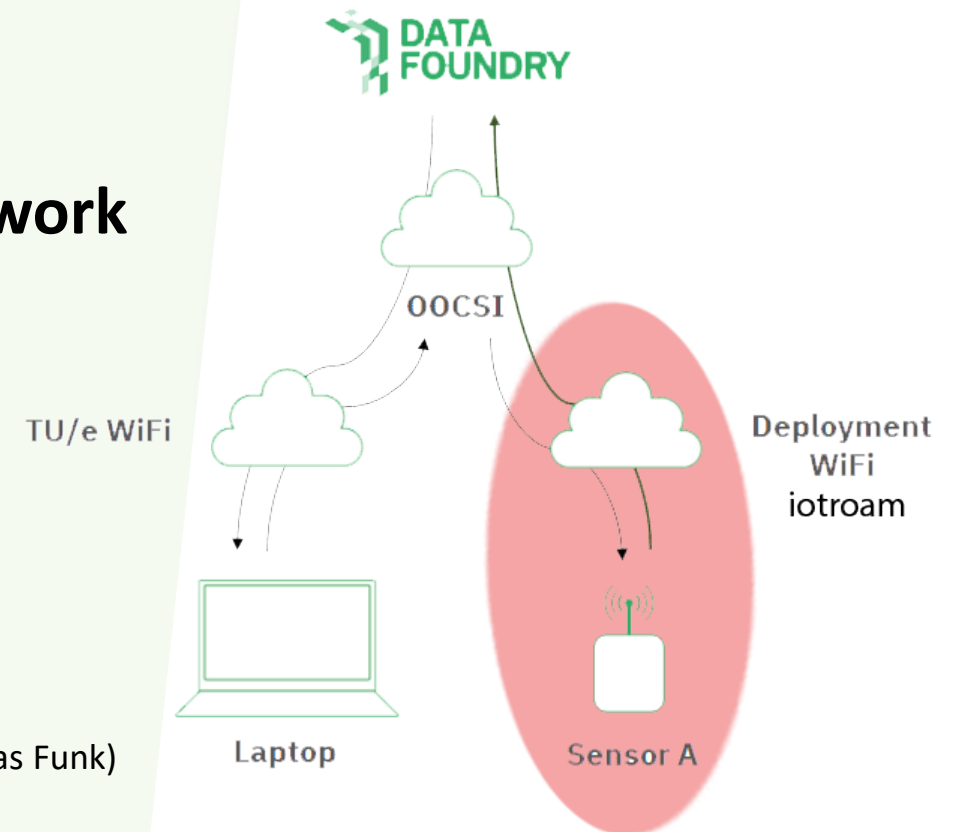
1. Preset of Data Foundry

1. Create a project → Should be done by Data Foundry automatically
2. Create an IoT dataset
3. Create a device

2. **Connect IoT device (ESP32) to iotroam network**

3. Send data through OOCSI
4. Check data in OOCSI UI Client page
5. Save data to own IoT dataset

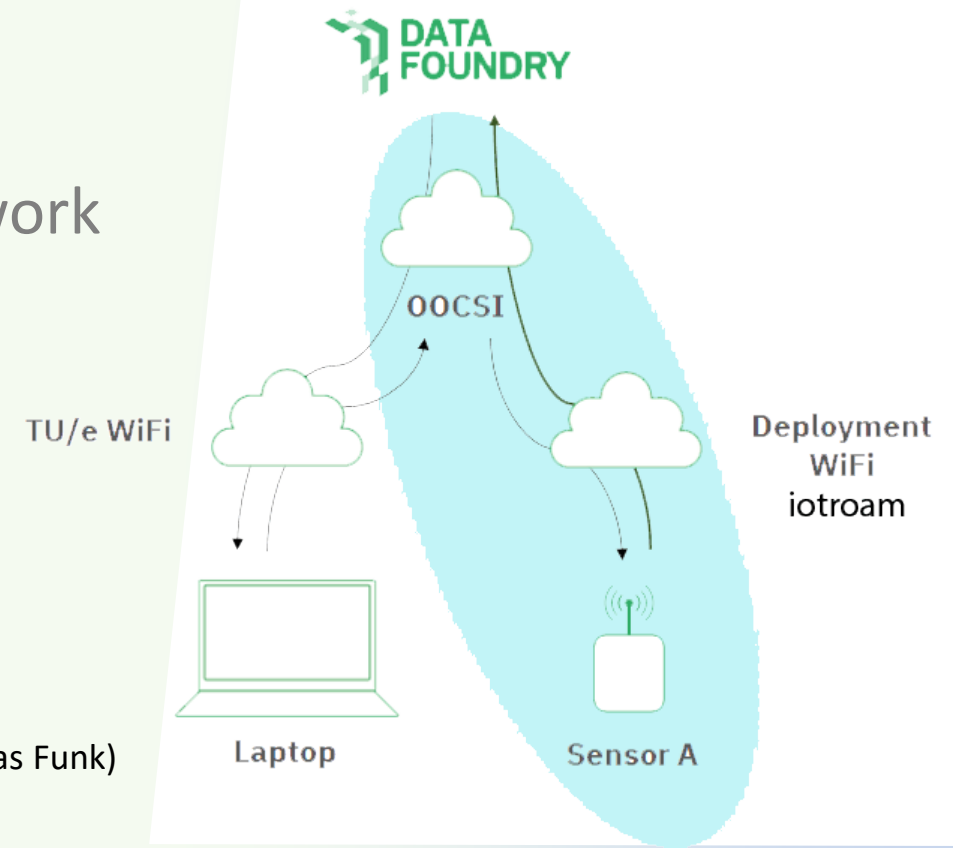
(Source from Mathias Funk)





Practice: Data collection by Data Foundry through OOCSI

1. Preset of Data Foundry
 1. Create a project → Should be done by Data Foundry automatically
 2. Create an IoT dataset
 3. Create a device
2. Connect IoT device (ESP32) to iotroam network
- 3. Send data through OOCSI**
4. Check data in OOCSI UI Client page
5. Save data to own IoT dataset

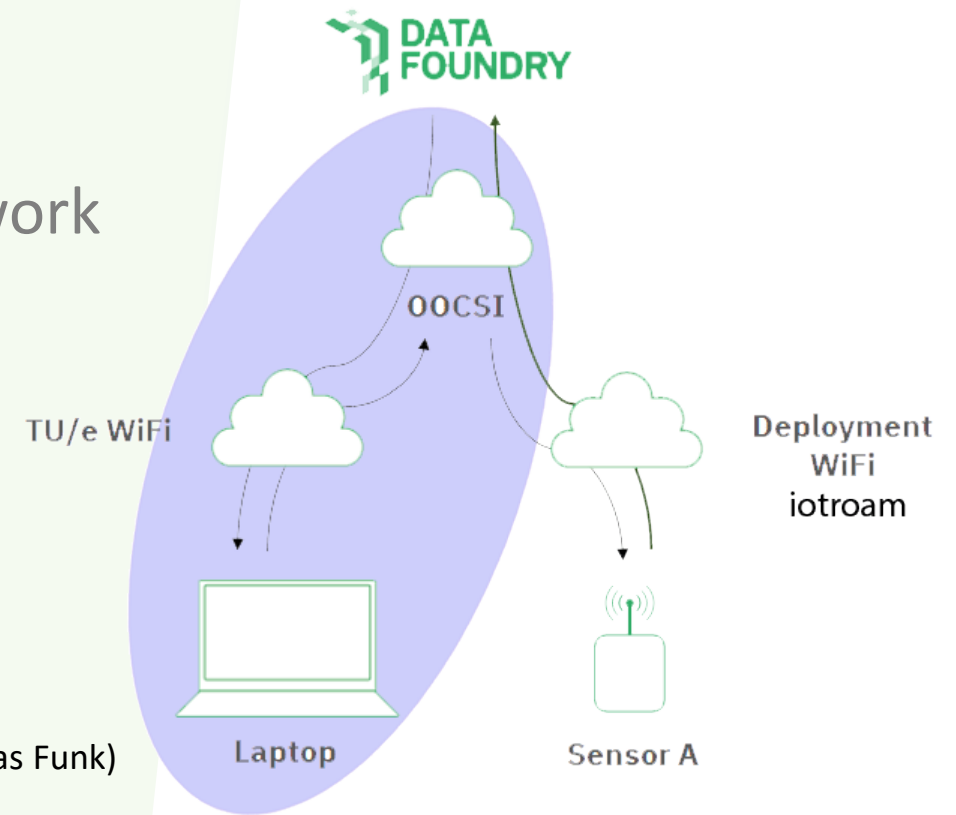


(Source from Mathias Funk)



Practice: Data collection by Data Foundry through OOC SI

1. Preset of Data Foundry
 1. Create a project → Should be done by Data Foundry automatically
 2. Create an IoT dataset
 3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOC SI
- 4. Check data in OOC SI UI Client page**
5. Save data to own IoT dataset

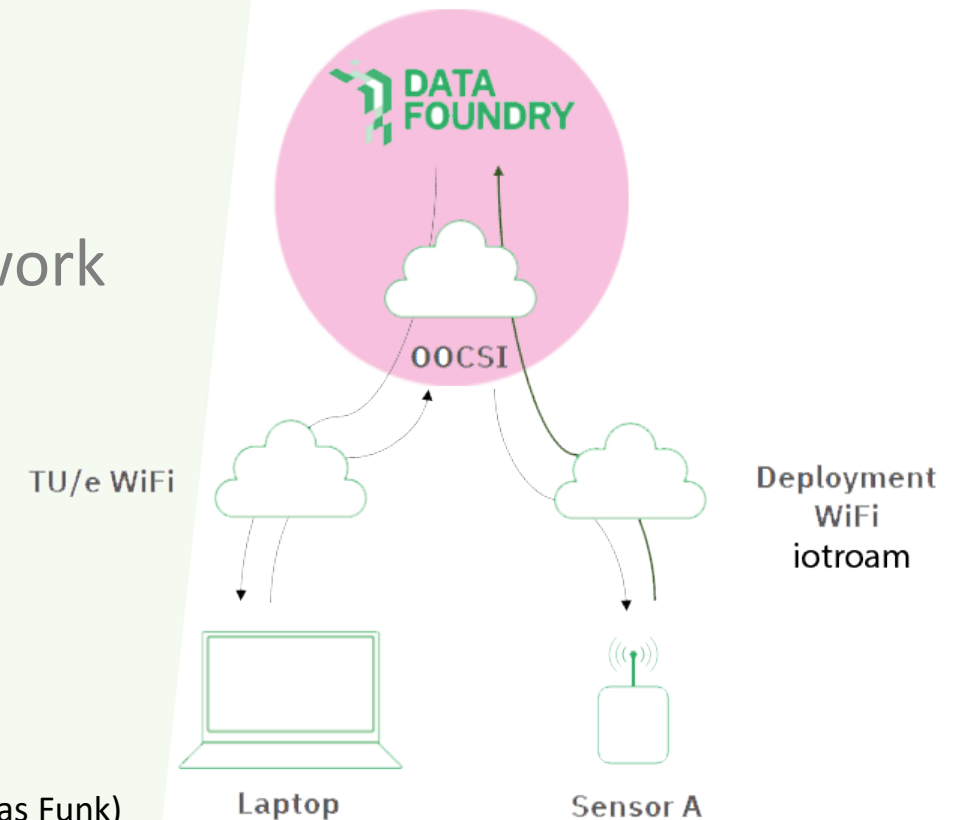


(Source from Mathias Funk)



Practice: Data collection by Data Foundry through OOCSI

1. Preset of Data Foundry
 1. Create a project → Should be done by Data Foundry automatically
 2. Create an IoT dataset
 3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOCSI
4. Check data in OOCSI UI Client page
5. **Save data to own IoT dataset via OOCSI**



(Source from Mathias Funk)



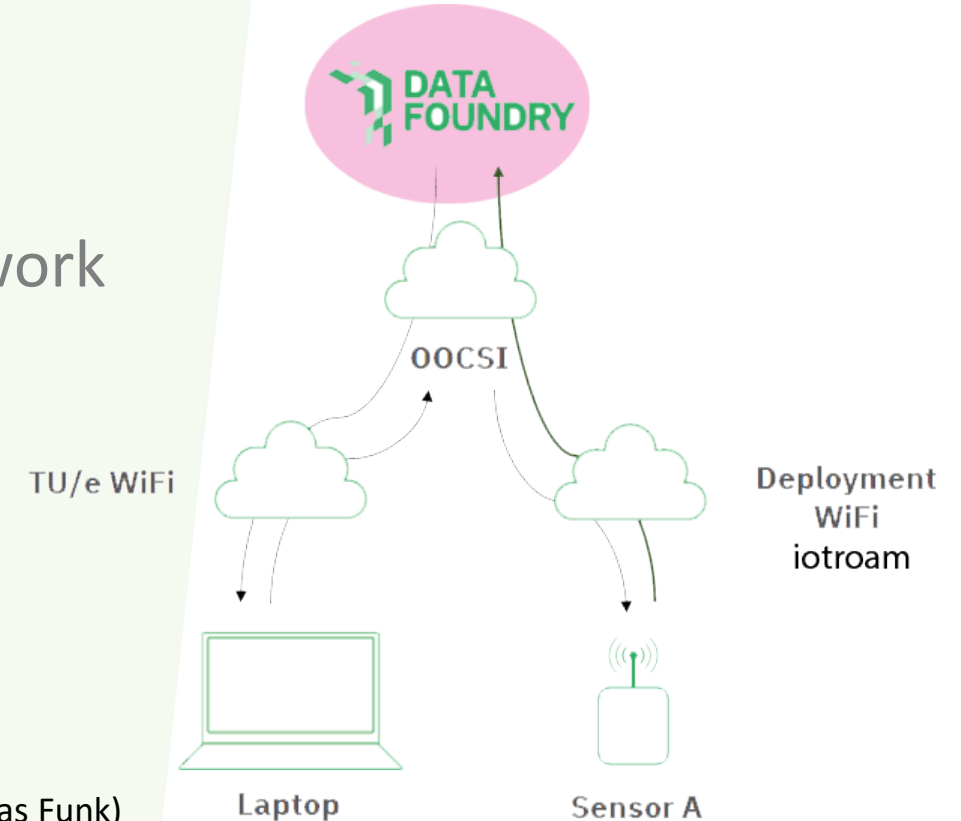
Practice: Data collection by Data Foundry through OOC SI

1. Preset of Data Foundry

1. Create a project → Should be done by Data Foundry automatically
2. Create an IoT dataset
3. Create a device

2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOC SI
4. Check data in OOC SI UI Client page
5. Save data to own IoT dataset

(Source from Mathias Funk)

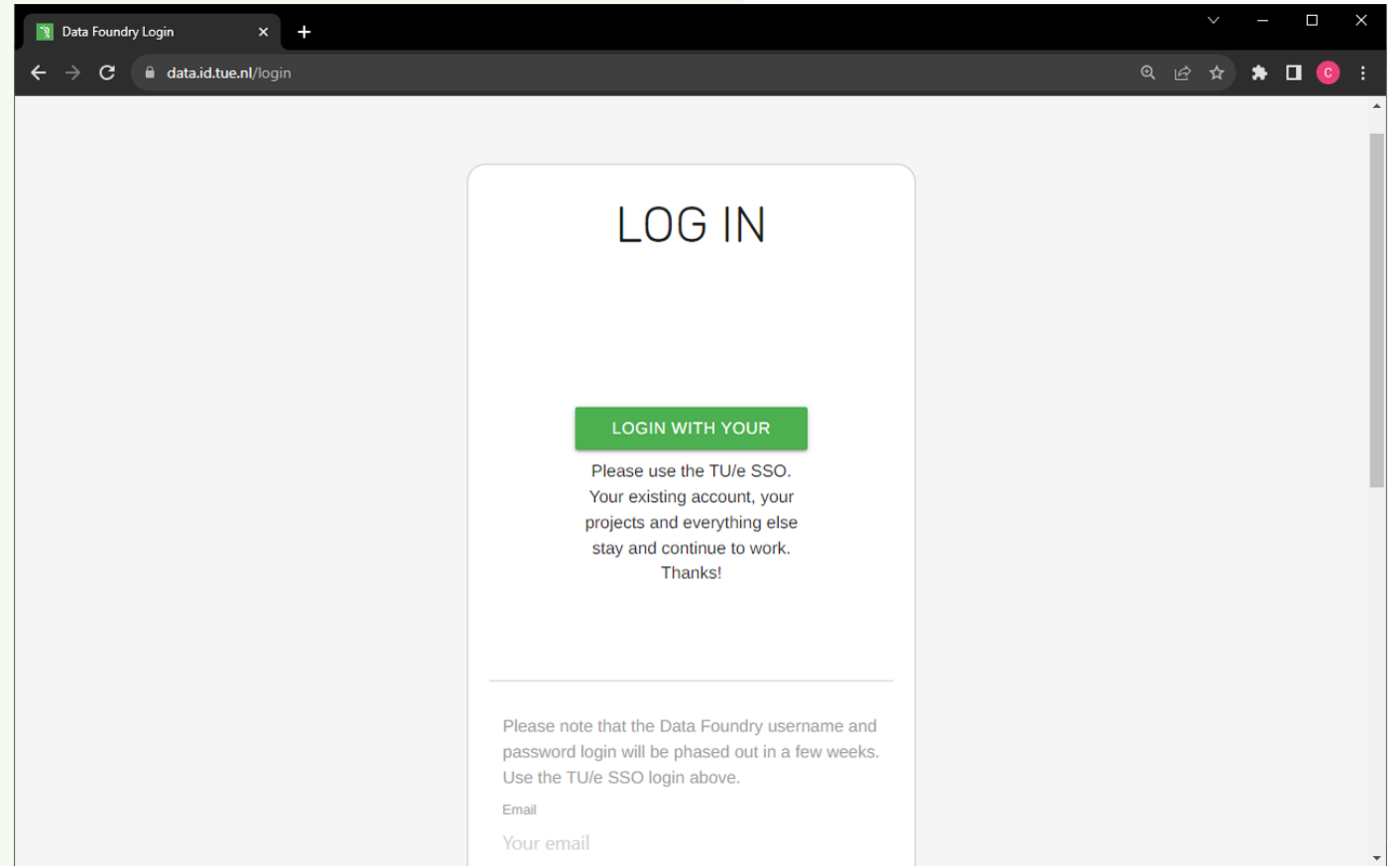




Step 1.1 - CREATING YOUR FIRST PROJECT (1/3)

Log in with TU/e account

1. Browse to <https://data.id.tue.nl>
2. Click on “LOGIN” button at the left side
3. Click on “LOGIN WITH YOUR TU/E ACCOUNT” button
4. Finish the authorization process (to login with TU/e account)
5. You are in!



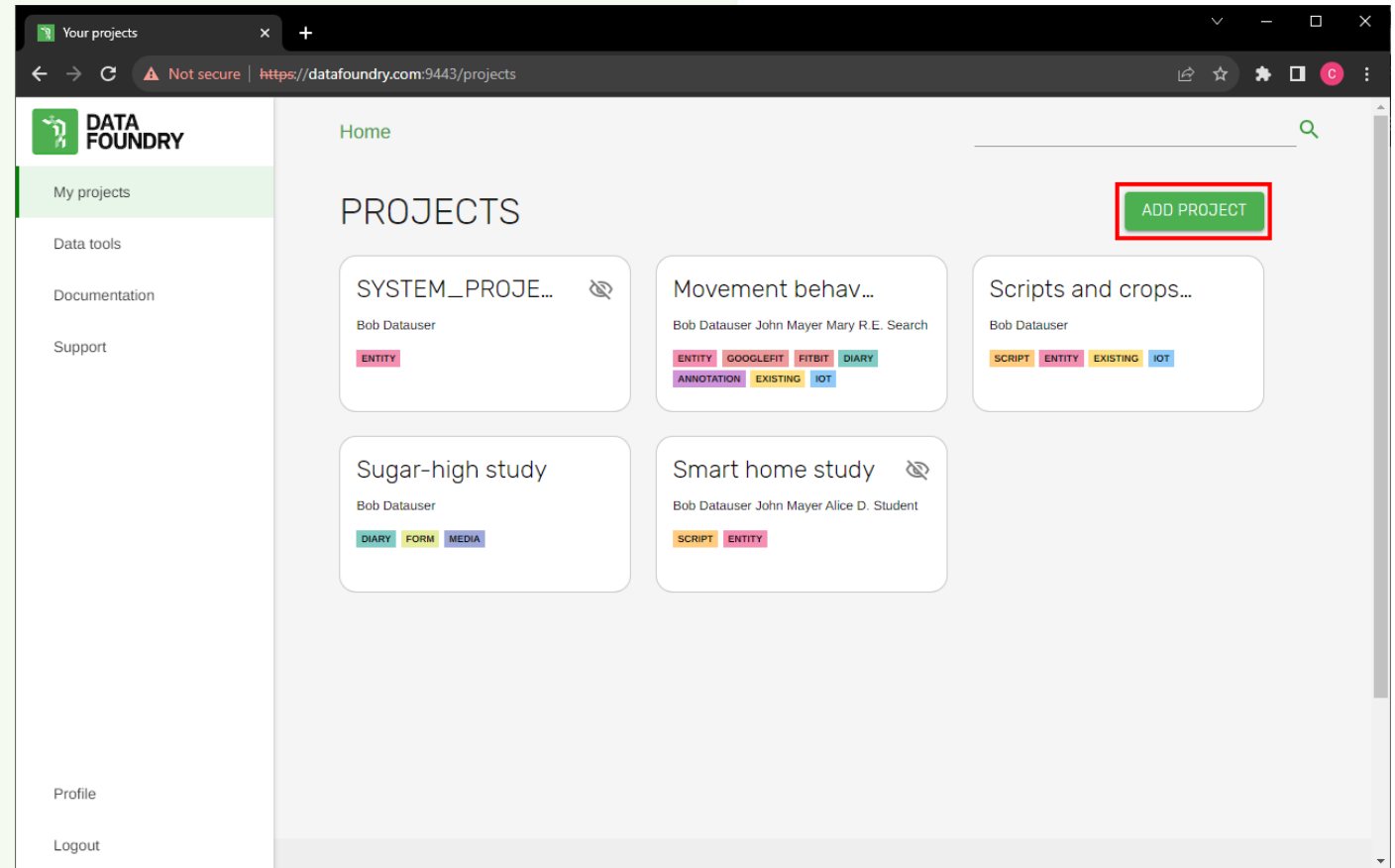
(Source: <https://data.id.tue.nl>)



Step 1.1 - CREATING YOUR FIRST PROJECT (2/3)

Create a project
(should be done automatically after first logging in)

1. Go to the **home** page
2. Click on **add project** on the right.
3. Fill in the **details**
4. Save it



(Source: <https://data.id.tue.nl>)



Step 1.1 - CREATING YOUR FIRST PROJECT (3/3)

Home > New project

ADD NEW PROJECT

Project name
Ex: 'IoT home jogging study', 'Office snack'
This should make sense to other people as well, so nothing like "test project", "Joe's first project", "final study"... :-)

Project introduction
Ex: This study reports on... We survey home owners and tenants
This text will be shown also to participants as part of the consent form, so adding a few details helps in getting consent from your participants.

Project license
Choose project license
Enter a license for this project (including all its datasets). We support different licenses, please do take a moment to choose the right license.

☐ This project is available for everyone.
A public project will appear in search results and its meta-data such as name, introduction and data type will be visible, same for all data sets in the project. The project will not be editable by guests or subscribers, and they need to accept the project license to

(Source: <https://data.id.tue.nl>)



Step 1.2 - CREATING YOUR FIRST IOT DATASET (1/2)

Create an IoT dataset within that project

1. Click on **add dataset** in the project page.
2. Select **IoT dataset**
3. Fill in the **details**
4. Save it

Important!

- The title of the dataset should make sense, and the text should describe the dataset clearly.
- The dataset only works in the period it is active.
- The default value for the start date is today and the end date is **3 months** after the start date.



Step 1.2 - CREATING YOUR FIRST IOT DATASET (2/2)

The screenshot shows a web browser window with the URL `data.id.tue.nl/datasets/ts/add/35`. The page title is "ADD IOT DATASET". On the left is a sidebar with the "DATA FOUNDRY" logo and navigation links: "My projects", "Data tools", "Documentation", "Support", "Profile", and "Logout". The main content area contains the following form fields and instructions:

- Dataset Name**: A text input field. Instruction: "This should make sense to other people as well, so nothing like 'test dataset', 'Joe's first dataset', 'final study data'... ;-)"
- Description**: A text input field. Instruction: "This text should describe the dataset clearly, and what kind of data you want to collect with it."
- Start Date (yyyy-mm-dd)**: A date input field. Instruction: "Select a start date to 'open' the data collection. Before this date, no data can be entered. If you don't choose a start date, the default is set as today."
- End Date (yyyy-mm-dd)**: A date input field. Instruction: "Select an end date to 'close' the data collection. After this date, no data can be entered anymore. If you don't choose an end date, the default is three months from the start date."
- Open participation**: A checkbox. Instruction: "Ticking this setting will allow for any source to append data into the dataset, not just registered ones, which can be useful in testing or debugging, or you use this for very open studies or designs (e.g., on websites or in apps)."

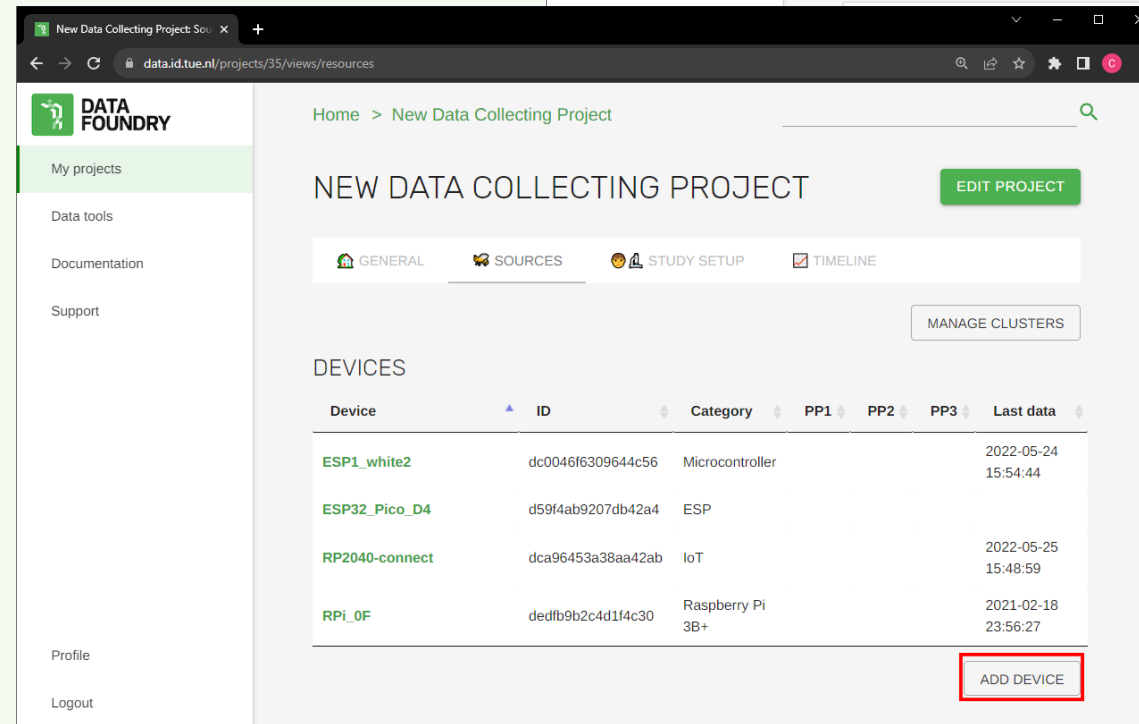
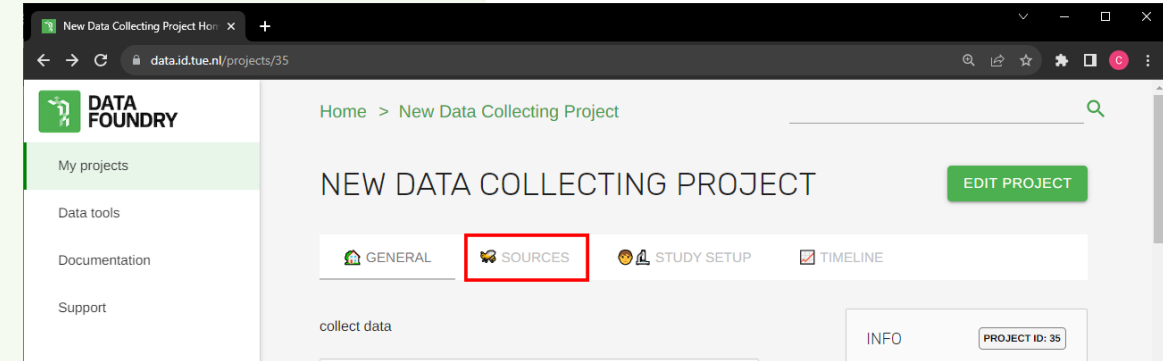
(Source: <https://data.id.tue.nl/>)



Step 1.3 - CREATING YOUR FIRST DEVICE (1/2)

Create a new device

1. Click on **SOURCES** tab on project page
2. Find the devices table and click **add device**
3. Fill in the **details**
4. Save it



(Source: <https://data.id.tue.nl>)



Step 1.3 - CREATING YOUR FIRST DEVICE (2/2)

The screenshot shows a web browser window with the URL `data.id.tue.nl/devices/add/35`. The page is titled "ADD NEW DEVICE" and features a sidebar on the left with the "DATA FOUNDRY" logo and navigation links: "My projects", "Data tools", "Documentation", "Support", "Profile", and "Logout". The main content area contains a breadcrumb trail: "Home > New Data Collecting Project > Sources > Add device". Below the title, there are five input fields, each with a label and a placeholder message:

- Device Name**: Placeholder "Please enter the name of your device."
- Category**: Placeholder "Please enter category of your device."
- Subtype**: Placeholder "Please enter subtype of your device."
- Public Parameter1**: Placeholder "You can edit public parameter1 of this device."
- Public Parameter2**: Placeholder "You can edit public parameter2 of this device."

(Source: <https://data.id.tue.nl>)



Practice: Data collection by Data Foundry through OOCSI

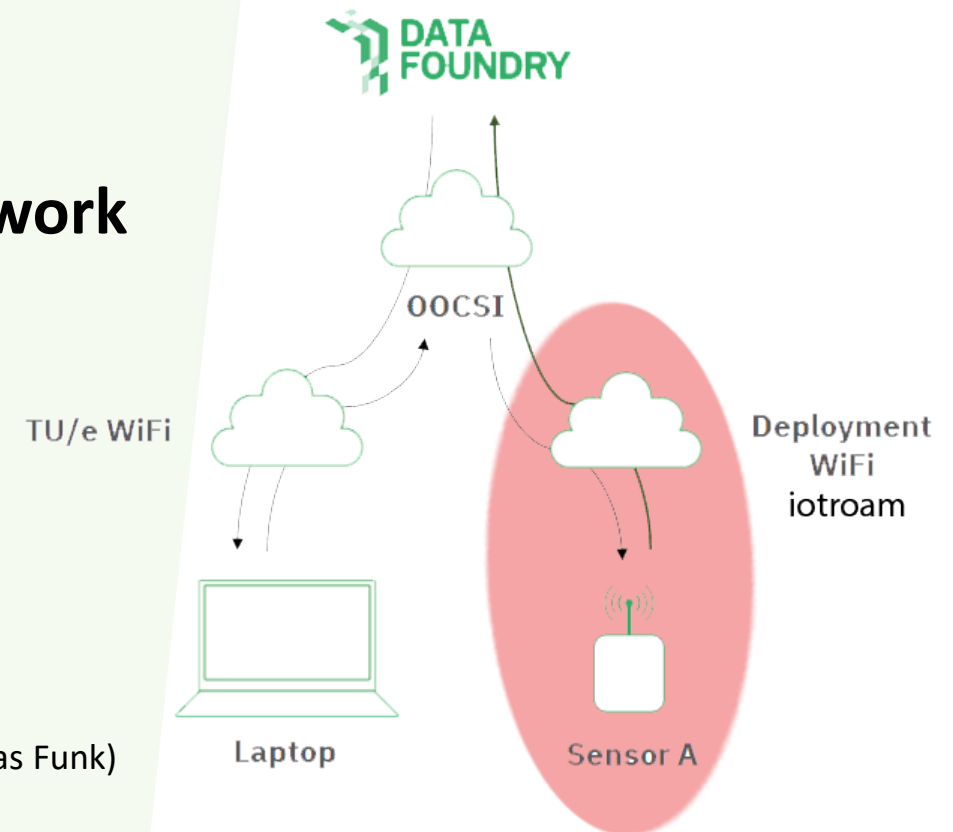
1. Preset of Data Foundry

1. Create a project → Should be done by Data Foundry automatically
2. Create an IoT dataset
3. Create a device

2. **Connect IoT device (ESP32) to iotroam network**

3. Send data through OOCSI
4. Check data in OOCSI UI Client page
5. Save data to own IoT dataset

(Source from Mathias Funk)





Step 2 – CONNECT IOT DEVICE (ESP32) (1/5)

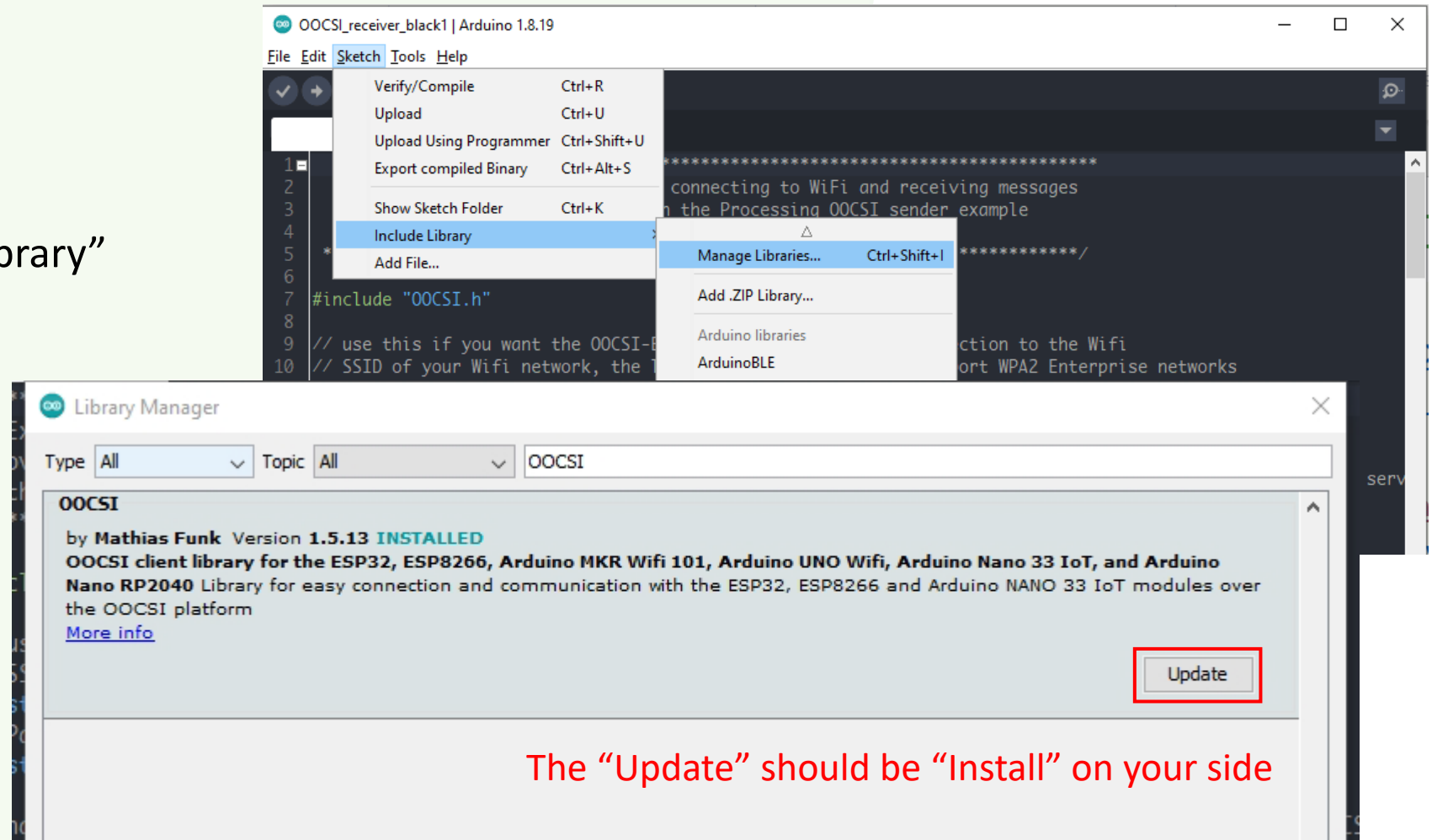
- Arduino IDE
 - [Install Arduino IDE](#)
 - [Add ESP32 board package to Arduino IDE](#)
- Install ESP32 USB driver
 - Connect ESP32 board and check first, try the drivers if it's not working
 - Windows machine
 - [Download driver](#)
 - [How to install](#)
 - Mac machine
 - Try to connect directly and check



Step 2 – CONNECT IOT DEVICE (ESP32) (2/5)

Install OOC SI library to Arduino IDE

1. Open Arduino IDE
2. “Sketch” -> “Include Library”
-> “Manage Libraries”
3. Search “OOCSI”
4. Click “Install”
5. [More information](#)



The “Update” should be “Install” on your side

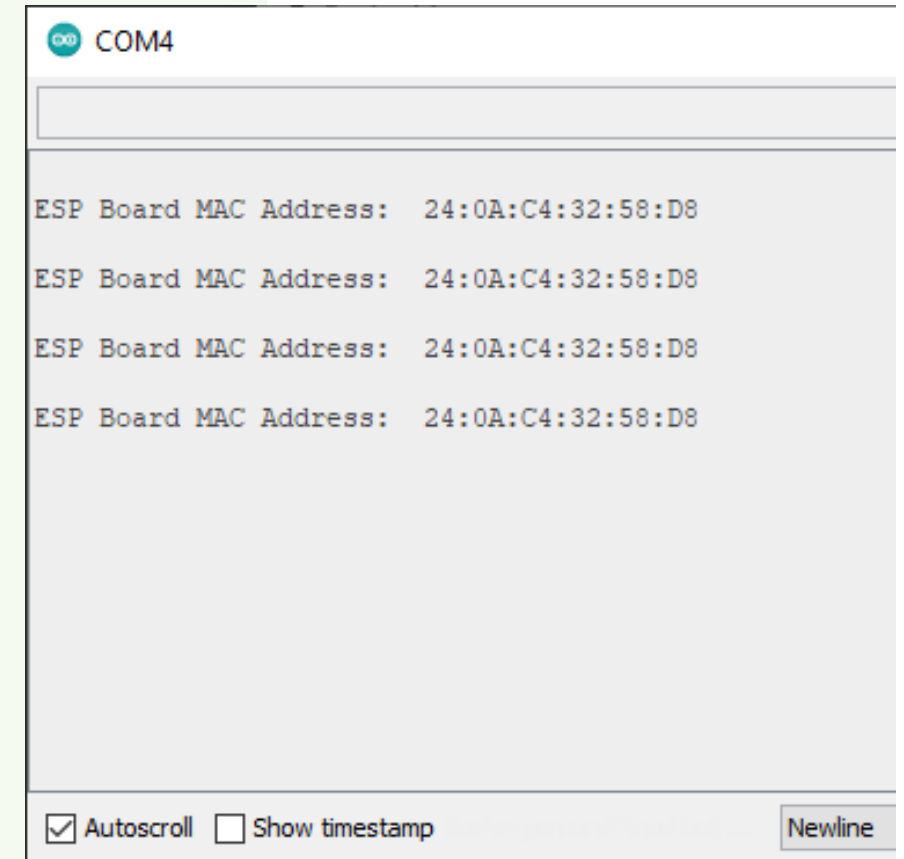


Step 2 – CONNECT IOT DEVICE (ESP32) (3/5)

- Paste the following code to a new sketch of Arduino IDE and save:

```
#include <WiFi.h>
void setup(){
  Serial.begin(115200);
}

void loop(){
  Serial.println();
  Serial.print("ESP Board MAC Address: ");
  Serial.println(WiFi.macAddress());
  delay(3000);
}
```





Step 2 – CONNECT IOT DEVICE (ESP32) (4/5)

- Check the MAC address of the IoT device
 - [Upload a sketch to IoT device with Arduino IDE](#)
 - Choose the **board** we're using:
"Tools" -> "Board:...something_here..." -> "ESP32 Arduino" -> "DOIT ESP32 DEVKIT V1"
 - Select the **COM port** you checked [on the Device Manager](#):
"Tools" -> "Port" -> "COM #", # is a number with 1 or 2 digits
 - Serial Monitor: "Tools" -> "Serial Monitor"
 - The MAC address would be a string of 6 x 2 characters combined with ":", which looks like -- ab:83:fd:83:e4:89
 - Copy the MAC address you got



Step 2 – CONNECT IOT DEVICE (ESP32) to iotroam network (5/5)

Register IoT device to “iotroam” network

1. Register page: <https://iotroam.org>
2. Click on “add device” or “+”
3. Fill in the MAC address and other details
4. Set the “Expiry date” as tomorrow
5. Set and copy the password
6. Click **Create**

The screenshot shows the 'Add new device' form in the iotroam personal dashboard. The form includes fields for Description, MAC, Password, Location (optional), and Expiry date (optional). The 'Create' button is highlighted in blue.

Field	Value
Description	
MAC	
Password	*****
Location (optional)	
Expiry date (optional)	mm/dd/yyyy

The screenshot shows the 'Personal' device view in the iotroam personal dashboard. The device is named 'ESP32-white2' and has a MAC address of '24-0A-C4-30-CA-10'. The 'Create' button is highlighted in blue.

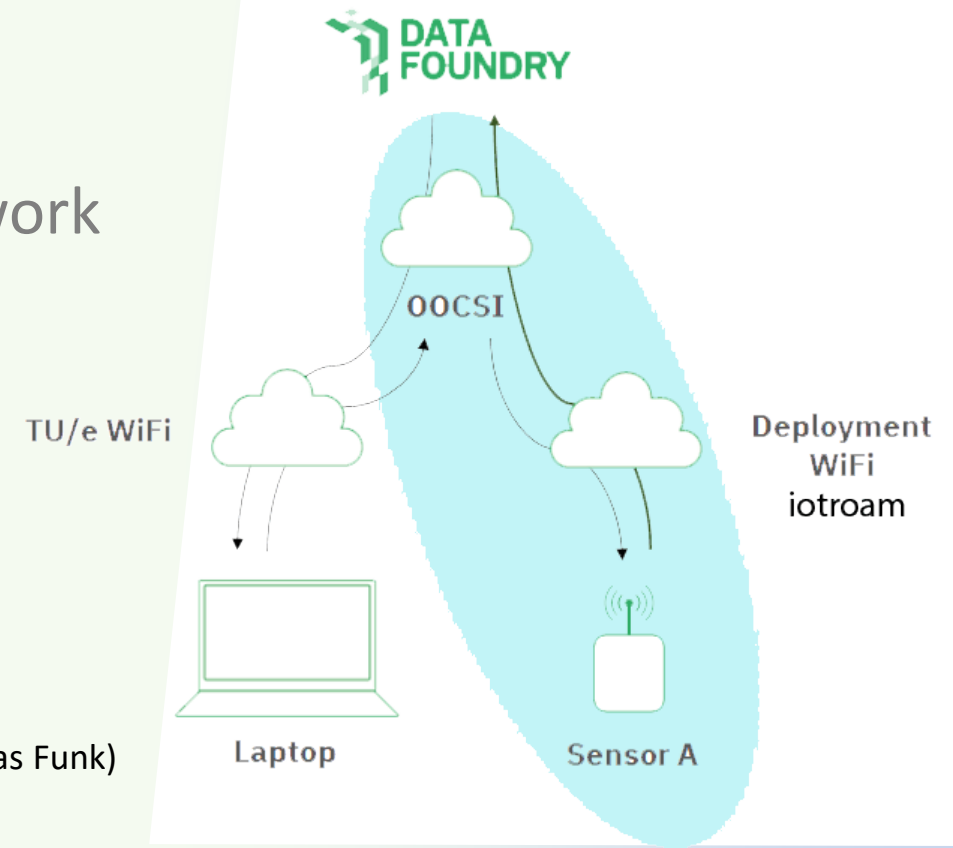
Field	Value
Name	ESP32-white2
MAC	24-0A-C4-30-CA-10
Location (optional)	DF workshop
Expiry date	-

(Source: <https://iotroam.org>)



Practice: Data collection by Data Foundry through OOCSI

1. Preset of Data Foundry
 1. Create a project → Should be done by Data Foundry automatically
 2. Create an IoT dataset
 3. Create a device
2. Connect IoT device (ESP32) to iotroam network
- 3. Send data through OOCSI**
4. Check data in OOCSI UI Client page
5. Save data to own IoT dataset



(Source from Mathias Funk)



Step 3 – SEND DATA THROUGH OOCSI (1/5)

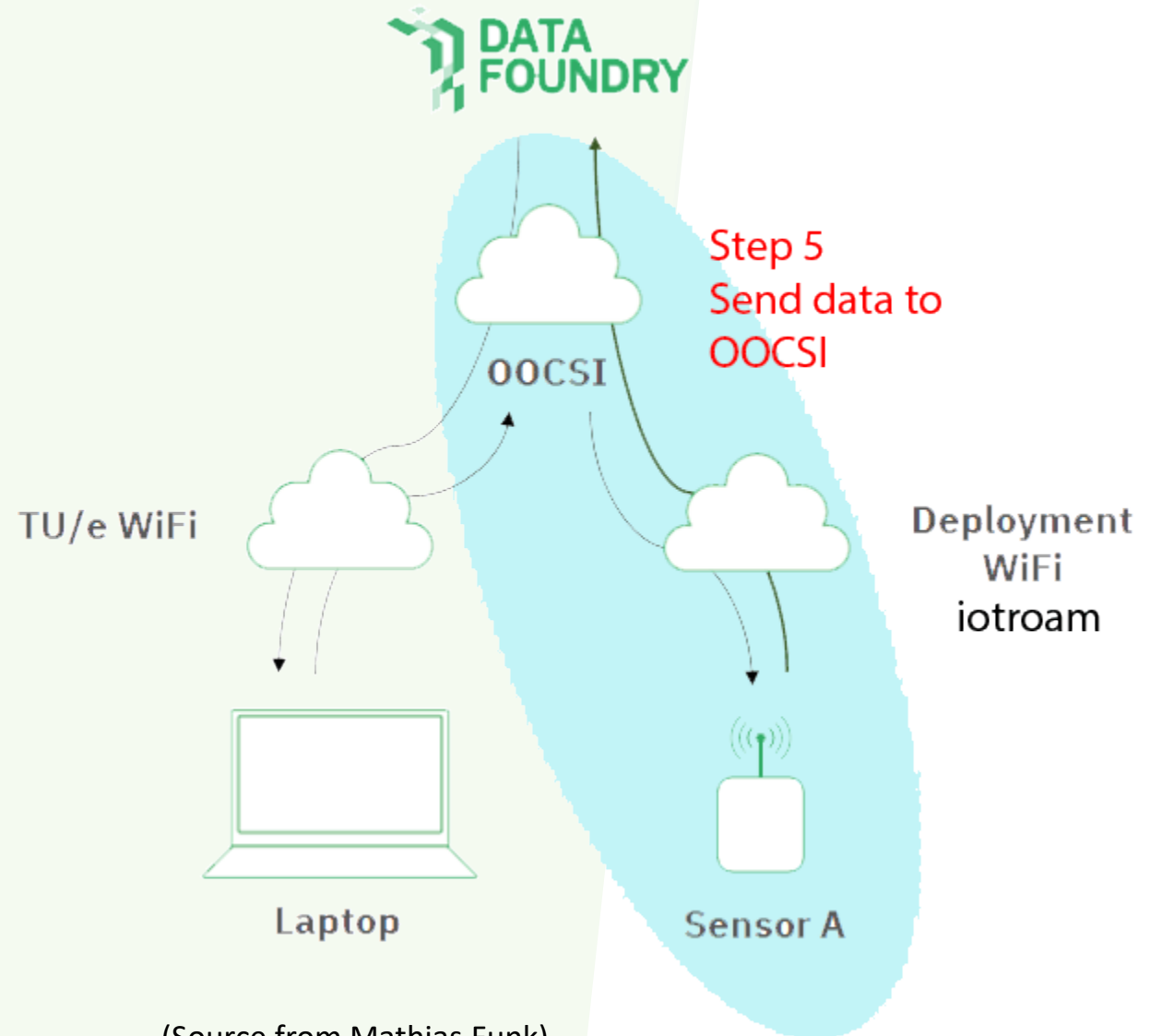
What is OOCSI?

“ OOCSI is a design prototyping *middleware* that allows ‘clients’ across platforms and programming languages to communicate via a ‘server’ (or ‘broker’ if you prefer middleware terminology). ”

-- Mathias Funk, OOCSI creator

Website: <https://oocsi.net/>

Github: <https://github.com/iddi/oocsi>



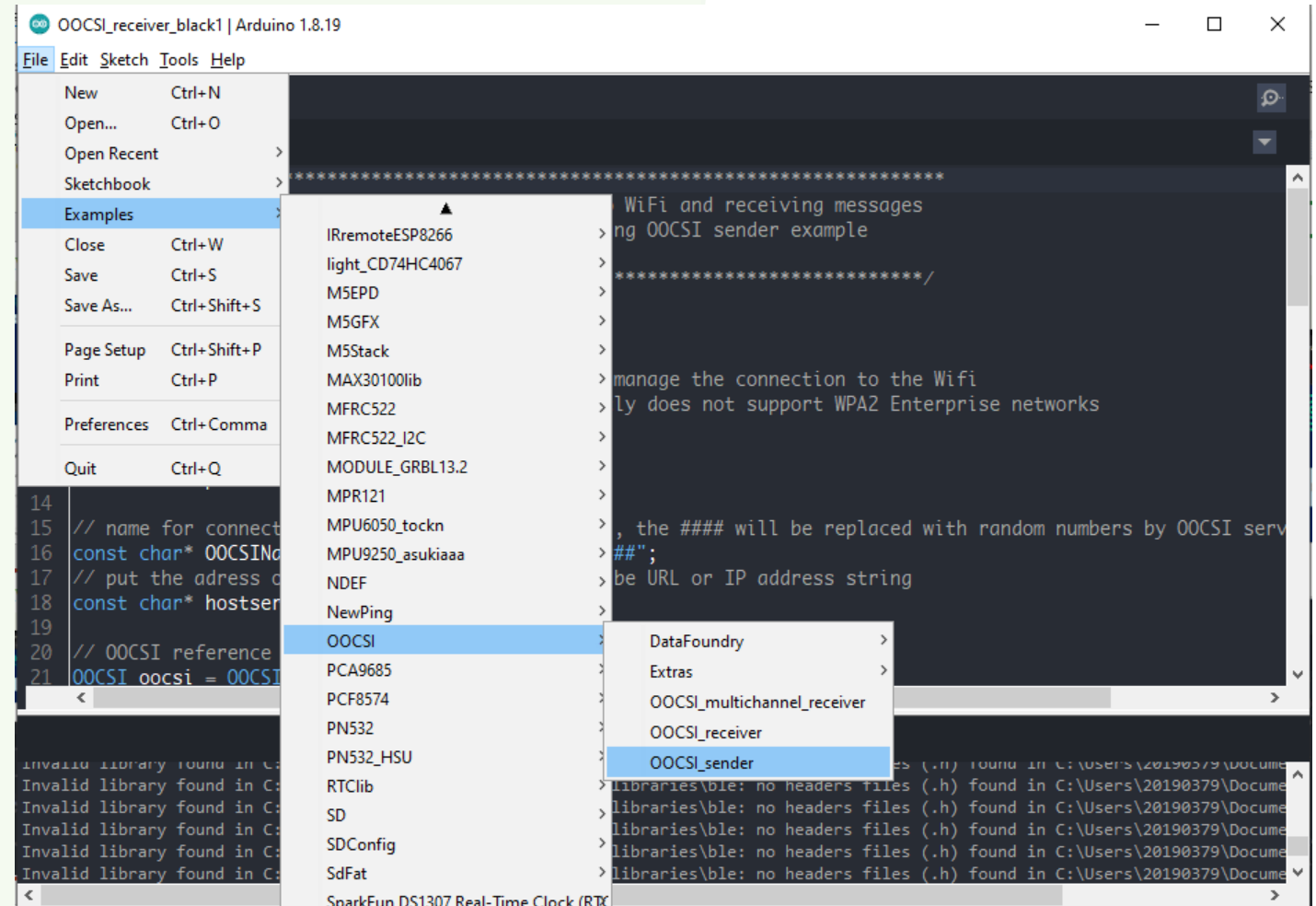
(Source from Mathias Funk)



Step 3 – SEND DATA THROUGH OOCSSI (2/5)

Open example code of sending data to Data Foundry through OOCSSI

1. Open Arduino IDE
2. “File”
 - > “Examples”
 - > “OOCSSI”
 - > “OOCSSI_sender”





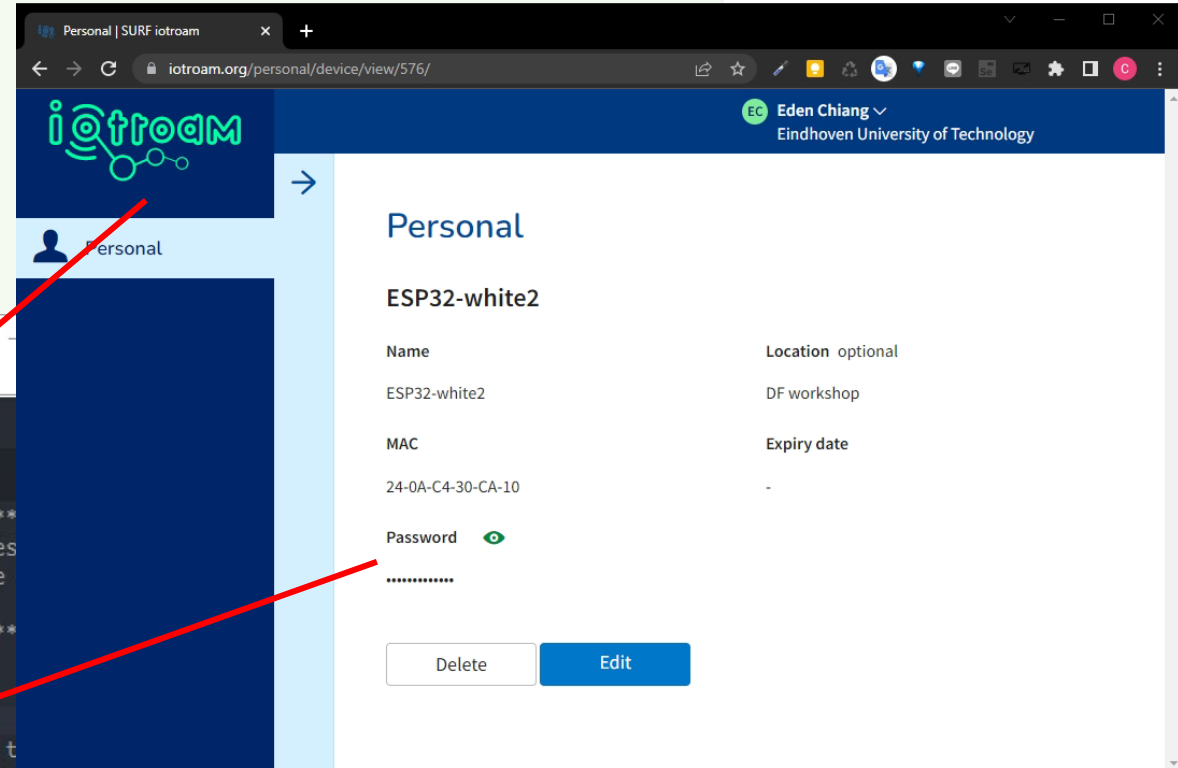
Step 3 – SEND DATA THROUGH OOCSI (3/5)

iotroam WiFi connection setting

```
OOCSI_sender_white1 | Arduino 1.8.19
File Edit Sketch Tools Help

1 //*****
2 // Example of the OOCSI-ESP library connecting to WiFi and sending messages
3 // over OOCSI. Designed to work with the Processing OOCSI receiver example
4 // that is provided in the same directory
5 //*****
6
7 #include "OOCSI.h"
8
9 // use this if you want the OOCSI-ESP library to manage the connection to t
10 // SSID of your Wifi network, the library currently does not support WPA2 Enterp
11 const char* ssid = "iotroam";
12 // Password of your Wifi network.
13 const char* password = "iotroam-esp32";
14
15 // name for connecting with OOCSI (unique handle), the #### will be replaced wit
16 const char* OOCSIName = "ESP_OOCSI_SENDER_EDEN_####";
17 // put the adress of your OOCSI server here, can be URL or IP address string
18 const char* hostserver = "oocsi.id.tue.nl";
19
```

OOCSI server



(Source: <https://iotroam.org>)

Copy code here:

```
const char* ssid = "iotroam";
const char* password = "what_you_set_in_iotroam";
const char* hostserver = "oocsi.id.tue.nl";
const char* OOCSIName = "Eden_workshop_ESP_####";
```



Step 3 – SEND DATA THROUGH OOCSSI (4/5)

Copy the ID of the IoT device

1. Click on **SOURCES** tab on project page
2. Find the newly created IoT device from the devices table
3. Copy the ID

The screenshot shows the 'NEW DATA COLLECTING PROJECT' page in the Data Foundry interface. The 'SOURCES' tab is selected and highlighted with a red box. Below the tabs, there is a table titled 'DEVICES'. The 'ID' column of this table is highlighted with a red box. The table contains the following data:

Device	ID	Category	PP1	PP2	PP3	Last data
ESP1_white2	dc0046f6309644c56	Microcontroller				2022-05-24 15:54:44
ESP32_Pico_D4	d59f4ab9207db42a4	ESP				
RP2040-connect	dca96453a38aa42ab	IoT				2022-05-25 15:48:59
RPi_0F	dedfb9b2c4d1f4c30	Raspberry Pi 3B+				2021-02-18 23:56:27

(Source: <https://data.id.tue.nl>)



Step 3 – SEND DATA THROUGH OOCSSI (5/5)

Configuration for OOCSSI in code

1. Back to OOCSSI_sender in Arduino IDE
2. Replace the channel name as “eden_esp32_test” in the line starts with “oocsi.newMessage”
3. To set device ID: Add one line code between the “oocsi.newMessage(...);” and “oocsi.sendMessage();”:
oocsi.addString(
 “device_id”,
 “PASTE_DEVICE_ID_HERE”);
4. Upload the code to IoT device

Copy code here:

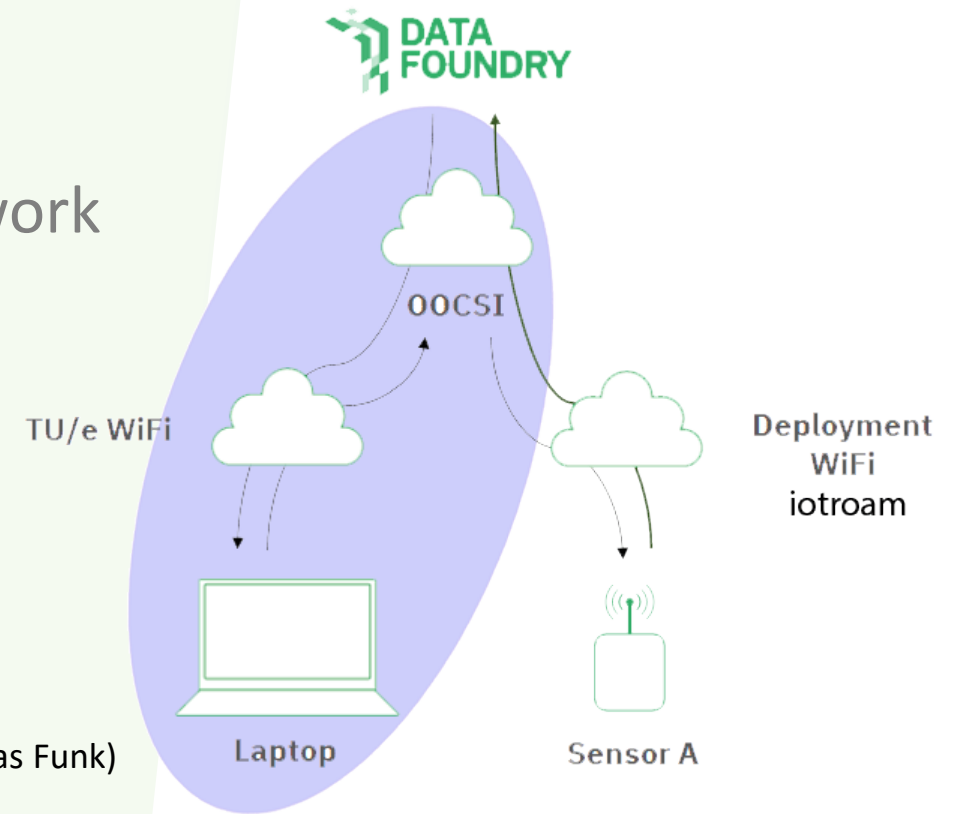
```
oocsi.newMessage(“eden_esp32_test”);  
oocsi.addString(“device_id”, “your_device_reference_ID”);
```

```
OOCSi_sender_white1 | Arduino 1.8.19  
File Edit Sketch Tools Help  
[Upload Button] [Verify Button] [Serial Monitor Button] [Serial Plotter Button] [USB Button] [Power Button]  
  
76 if (isButtonPressed) {  
77     if (not isSending) {  
78         isButtonPressed = false;  
79         isSending = true;  
80         msgTime = currentTime;  
81         // create a new message  
82         oocsi.newMessage("eden_esp32_test"); 2  
83  
84         // add data (primitive data types int, float, long, string)  
85         // the labels such as "count" or "timestamp" are completely free to c  
86         oocsi.addFloat("float_point", sin(millis()));  
87         oocsi.addLong("time", (long) millis());  
88         oocsi.addString("from", "ESP1");  
89         oocsi.addString("device_id", "dc0046f6309644c56"); // esp1-white2  
90     3 oocsi.addString("from", "ESP3");  
91     // oocsi.addString("device_id", "d13c15833107f4ab7"); // esp3-black  
92  
93  
94     // this command will send the message; don't forget to call this afte
```



Practice: Data collection by Data Foundry through OOC SI

1. Preset of Data Foundry
 1. Create a project → Should be done by Data Foundry automatically
 2. Create an IoT dataset
 3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOC SI
- 4. Check data in OOC SI UI Client page**
5. Save data to own IoT dataset



(Source from Mathias Funk)



Step 4 – CHECK DATA ON OOC SI UI CLIENT PAGE

Check data on UI Client page

1. Open OOC SI UI Client page:
<https://oocsi.id.tue.nl/test/visual>
SUBSCRIBE to channel:
“eden_esp32_test”
2. Check whether the data is coming into the channel
3. Unsubscribe the channel after checking

The screenshot shows the OOC SI UI Client page in a web browser. The page has a header with the OOC SI logo and a navigation bar. The main content area is divided into two panels. The left panel, titled 'Subscribe and receive data', shows a 'Channel name' field with the value 'eden_esp32_test'. Below this are three buttons: 'SUBSCRIBE' (blue), 'UNSUBSCRIBE' (red), and 'CLEAR' (grey). A scrollable list of JSON data is displayed below the buttons. The right panel, titled 'Send data', shows a 'Channel name' field and a text area containing the JSON object: `{"number": 123, "text": "content", "boolean": true}`. Below the text area are 'SEND' (blue) and 'CLEAR RESPONSES' (grey) buttons. At the bottom of the right panel is a checkbox labeled 'Send as a call'.

Test OOC SI for Web code on this server directly with a client, connected as webclient_1696856793391. Open a [new client window](#) in another tab.

Subscribe and receive data
Channel name
eden_esp32_test

SUBSCRIBE UNSUBSCRIBE CLEAR

```
{ "floatArray":  
  [55,65,75], "device_id": "d0364b2a9bdde4850", "c  
world!", "count": 40, "float_point": -0.756664574, "t  
[45,55,60]}  
  
{ "floatArray":  
  [55,65,75], "device_id": "d0364b2a9bdde4850", "c  
world!", "count": 40, "float_point": -0.459662557, "t  
[45,55,60]}  
  
{ "floatArray":  
  [55,65,75], "device_id": "d0364b2a9bdde4850", "c  
world!", "count": 40, "float_point": 0.972700059, "ti  
[45,55,60]}  
  
{ "floatArray":  
  [55,65,75], "device_id": "d0364b2a9bdde4850", "c  
world!", "count": 40, "float_point": -0.750748336, "t
```

Send data
Channel name

```
{ "number": 123, "text": "content",  
  "boolean": true}
```

SEND CLEAR RESPONSES

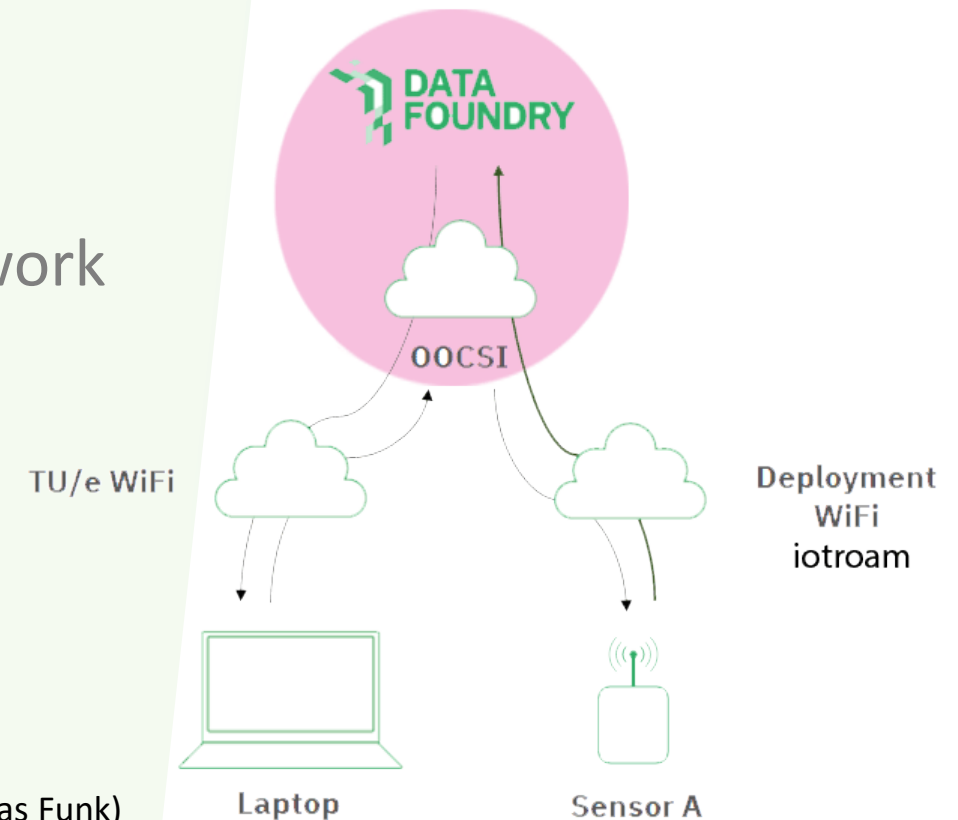
☐ Send as a call

(Source: <https://oocsi.id.tue.nl>)



Practice: Data collection by Data Foundry through OOCSI

1. Preset of Data Foundry
 1. Create a project → Should be done by Data Foundry automatically
 2. Create an IoT dataset
 3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOCSI
4. Check data in OOCSI UI Client page
5. **Save data to own IoT dataset via OOCSI**



(Source from Mathias Funk)



Step 5 – SAVE DATA INTO OWN IOT DATASET VIA OOCSI (1/4)

Configuration of OOCSI channel

1. On Data Foundry, open the IoT dataset page created at the beginning
2. Define the OOCSI channel name under the **OOCSI STREAM** tab (second one by the left side) in Configuration block
3. Click “SAVE”

View dataset x Add device | SURF iotroam x +

data.id.tue.nl/datasets/929

DATA FOUNDRY

My projects

Data tools

Documentation

Support

CONFIGURATION

HTTP-POST OOCSI STREAM CSV/JSON TOKE...

Subscribe to OOCSI channel as the data source for this dataset

You can use the OOCSI network to receive and store data directly into the dataset. This is option of platforms and technologies.

To subscribe to a channel for incoming data, you need to provide the channel name below.

Channel name

eden_esp32_test

SAVE

Delete t
unsubsc

OOCSI Diagnostics

Last successful event:

(Source: <https://data.id.tue.nl>)



Step 5 – SEND DATA TO OWN IOT DATASET VIA OOCISI (2/4)

Configuration with OOCISI channel in OOCISI_sender code

1. Back to OOCISI_sender code
2. Update the OOCISI channel name for sending message as the same one you defined in your IoT dataset
3. Upload code to IoT device

```
OOCSI_sender_white1 | Arduino 1.8.19
File Edit Sketch Tools Help

76 if (isButtonPressed) {
77   if (not isSending) {
78     isButtonPressed = false;
79     isSending = true;
80     msgTime = currentTime;
81     // create a new message
82     oocsi.newMessage("eden_esp32_test");
83
84     // add data (primitive data types int, float, long, string)
85     // the labels such as "count" or "timestamp" are completely free to choose
86     oocsi.addFloat("float_point", sin(millis()));
87     oocsi.addLong("time", (long) millis());
88     oocsi.addString("from", "ESP1");
89     oocsi.addString("device_id", "dc0046f6309644c56"); // esp1-white2
90     // oocsi.addString("from", "ESP3");
91     // oocsi.addString("device_id", "d13c15833107f4ab7"); // esp3-black
92
93
94     // this command will send the message; don't forget to call this after creating the message
95     oocsi.sendMessage();
96
```



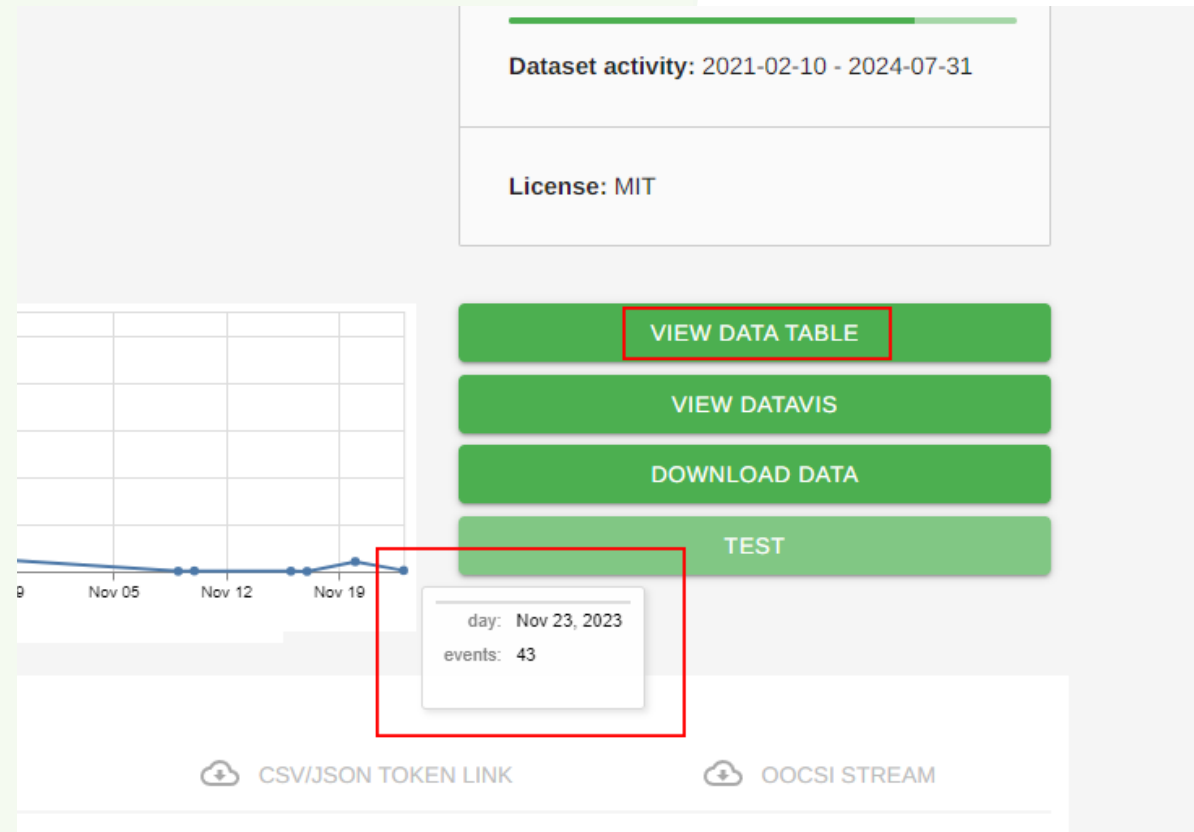
Step 5 – SEND DATA TO OWN IOT DATASET VIA OOCISI (3/4)

Store upcoming data in Data Foundry

Open and refresh the IoT dataset page and check **the blue dots on the chart** of the page or “**VIEW DATA TABLE**” and check the latest log.

The **blue dots** represent the events captured by Data Foundry. You can check the detail by hovering the dots with mouse.

As the number of events is increasing by every refreshing the page, which means the IoT dataset is set properly.



(Source: <https://data.id.tue.nl>)



Step 5 – SEND DATA TO OWN IOT DATASET VIA OOCISI (4/4)

Download your data as a CSV file

Click on the download button on the right to download the data directly as a CSV file.

	A	B	C	D	E
1	id,device_id,ts,activity,pp1,pp2,pp3,color,position				
2	1,240,2020-09-29T10:42:48.985,,,,,0,0				
3	2,240,2020-09-29T10:42:49.566,,,,,0,0				
4	3,240,2020-09-29T10:42:50.163,,,,,0,0				
5	4,240,2020-09-29T10:42:50.763,,,,,0,0				
6	5,240,2020-09-29T10:42:51.266,,,,,88,49				
7	6,240,2020-09-29T10:42:51.866,,,,,55,78				
8	7,240,2020-09-29T10:42:52.462,,,,,113,74				
9	8,240,2020-09-29T10:42:52.965,,,,,113,74				
10	9,240,2020-09-29T10:42:53.467,,,,,130,99				
11	10,240,2020-09-29T10:42:54.065,,,,,89,112				
12	11,240,2020-09-29T10:42:54.664,,,,,158,198				
13	12,240,2020-09-29T10:42:55.164,,,,,194,142				

Dataset activity: 2021-02-10 -
2024-07-31

License: MIT

VIEW DATA TABLE

VIEW DATAVIS

DOWNLOAD DATA

TEST

JSON TOKEN...



OOCISI STREAM

(Source: <https://data.id.tue.nl>)



FAQ

Q. ESP32 is not uploaded automatically after clicking the upload button?

A. You have to press the **BOOT** button on the ESP32 board when the “**Connecting...**” message is showing up in the console window of Arduino IDE as uploading the sketch to ESP32.

Q. Why is some sensors (pins) will not work as the Wi-Fi module is working?

A. This is because the design of this model, when the Wi-Fi module is working, only **ADC2** pins are available, which means **ADC1** will be occupied by Wi-Fi module; otherwise, **ADC1** pins will work. Check the pinout [here](#).

Q. Does DOIT ESP32 DevKit V1 (the one we're using in the workshop) have **5V** output pin?

A. Yes, but 5V output is **only available** by the **Vin** pin as the ESP32 is powered by **USB cable**.

Q. What is the case of the data can be saved into IoT dataset without **device_id**?

A. This happened as the target dataset is set as “**open participation**” (which can be found on the edit page of the dataset), which means data from any resources are available for the dataset. Otherwise, the **device_id** is required for identifying the sources in the project.



Reference links (1/2)

- Data Foundry: <https://data.id.tue.nl/>
 - [FAQ of Data Foundry](#)
- OOCISI: <https://oocsi.id.tue.nl/>
- Practical exercises with Data Foundry (Github): [PlayWithDataFoundry](#)
- [How to get MAC address and register to iotroam?](#)
- IoT network in TU/e: [lotroam](#)



Reference links (2/2)

- Arduino IDE
 - [Install Arduino IDE](#)
 - [Add ESP32 board package to Arduino IDE](#)
- Install ESP32 USB driver
 - Connect ESP32 board and check first, try the drivers if it's not working
 - Windows machine
 - [Download driver](#)
 - [How to install](#)
 - Mac machine
 - Try to connect directly and check
- [Check port of Windows and Mac machines](#)



Real things

