Connectivity of Data Foundry

Workshop 1 – IoT dataset

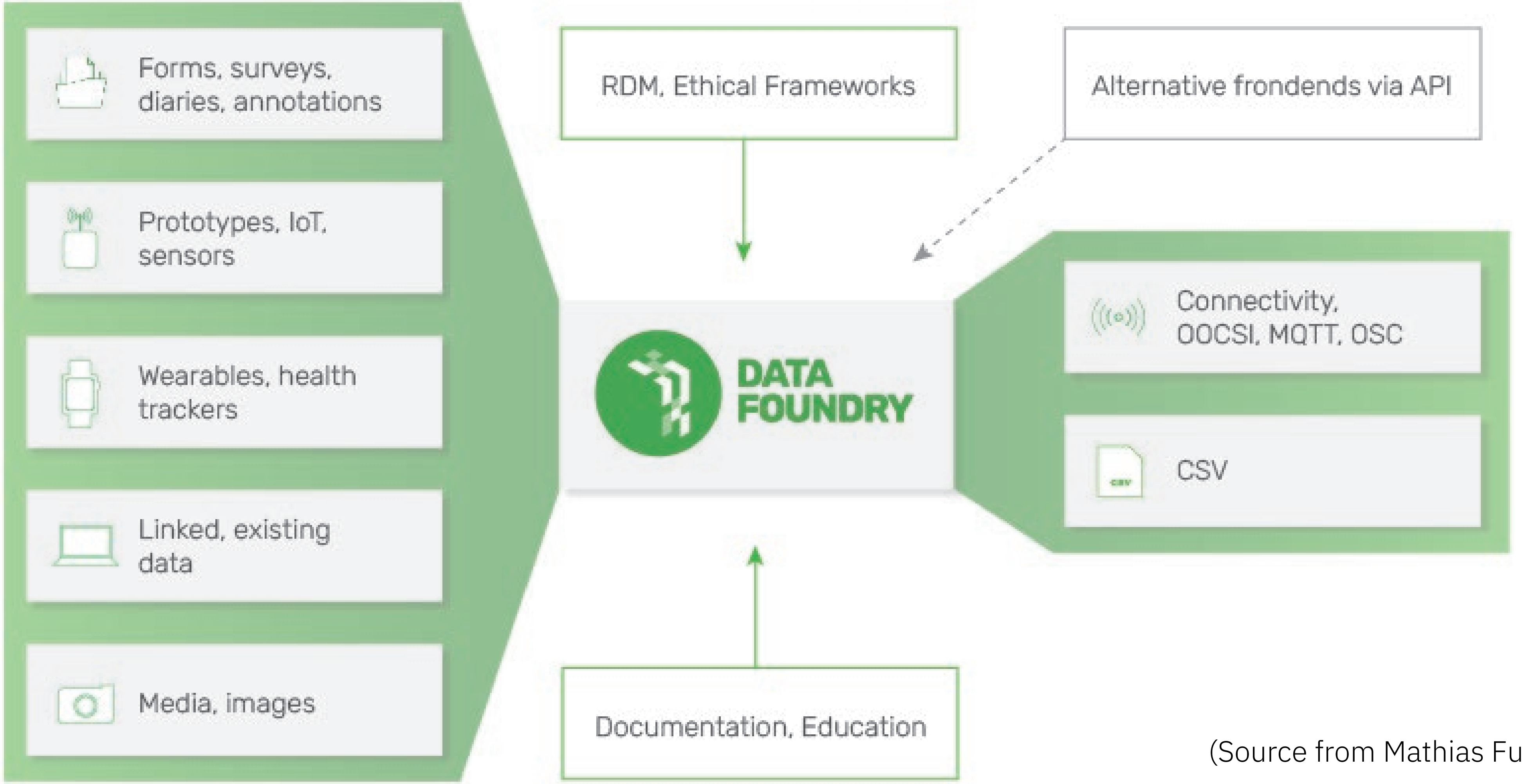
I-Tang (Eden) Chiang

i.chiang@tue.nl

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


Forms, surveys, diaries, annotations
Prototypes, IoT, sensors
Wearables, health trackers
Linked, existing data
Media, images
RDM, Ethical Frameworks
Alternative frondends via API
DATA FOUNDRY
Connectivity, OOCSI, MQTT, OSC
CSV
Documentation, Education

(Source from Mathias Funk)

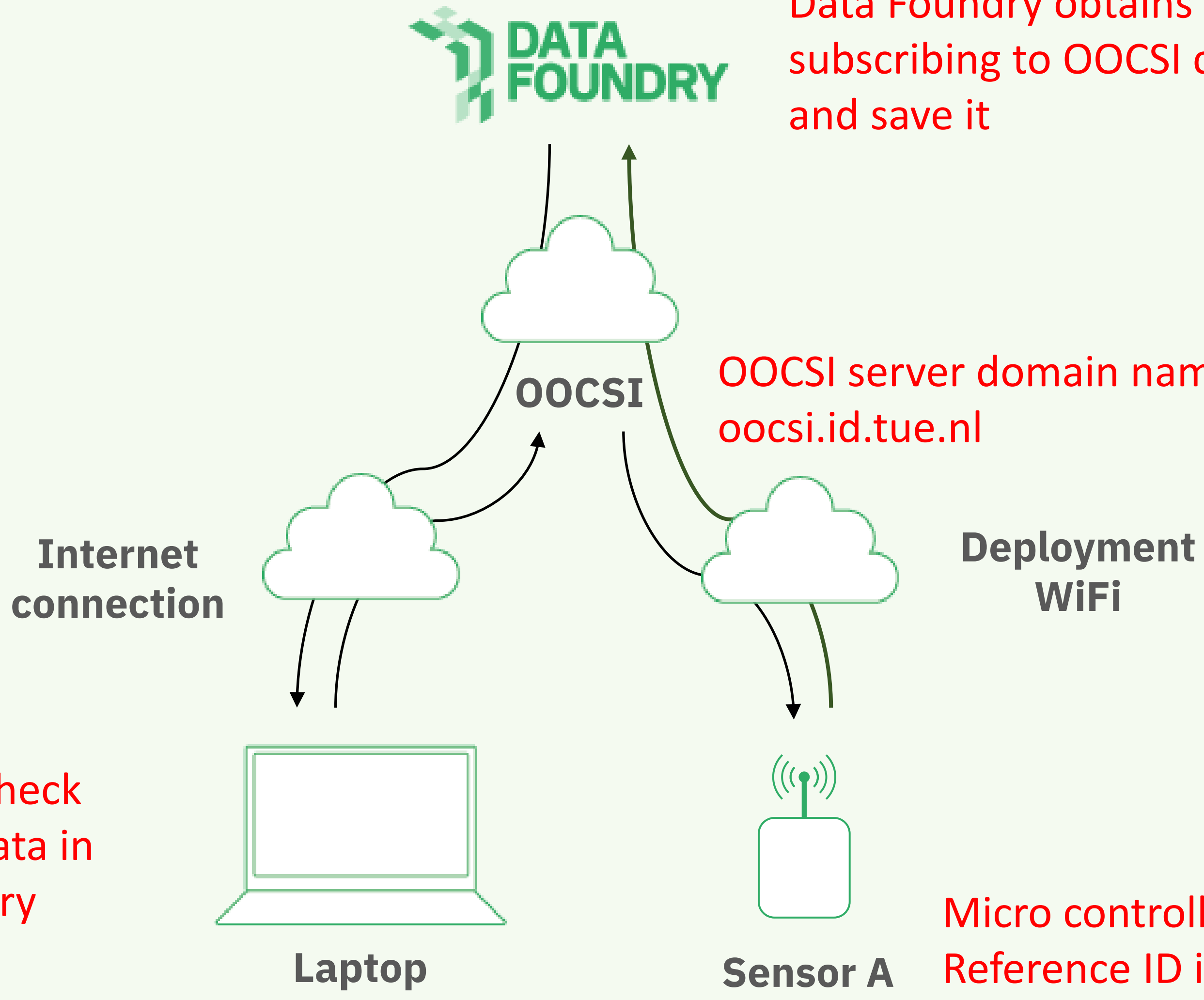
Why using Data Foundry

- 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 OOCSI work together



Data Foundry obtains data by subscribing to OOCSI channel and save it

OOCSI server domain name:
oocsi.id.tue.nl

IoT network in campus:
SSID: iotroam
Password: set in iotroam.org

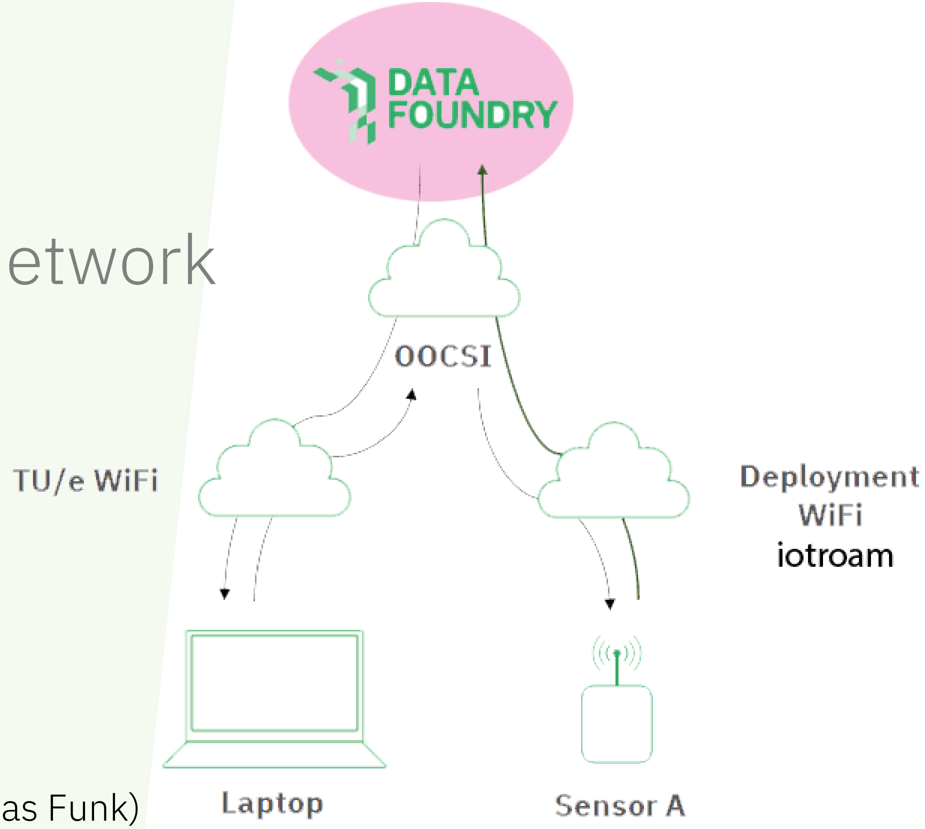
Users can check real-time data in Data Foundry

Micro controller
Reference ID in Data Foundry

(Source from Mathias Funk)

Practice: Data collection by Data Foundry through OOCSI

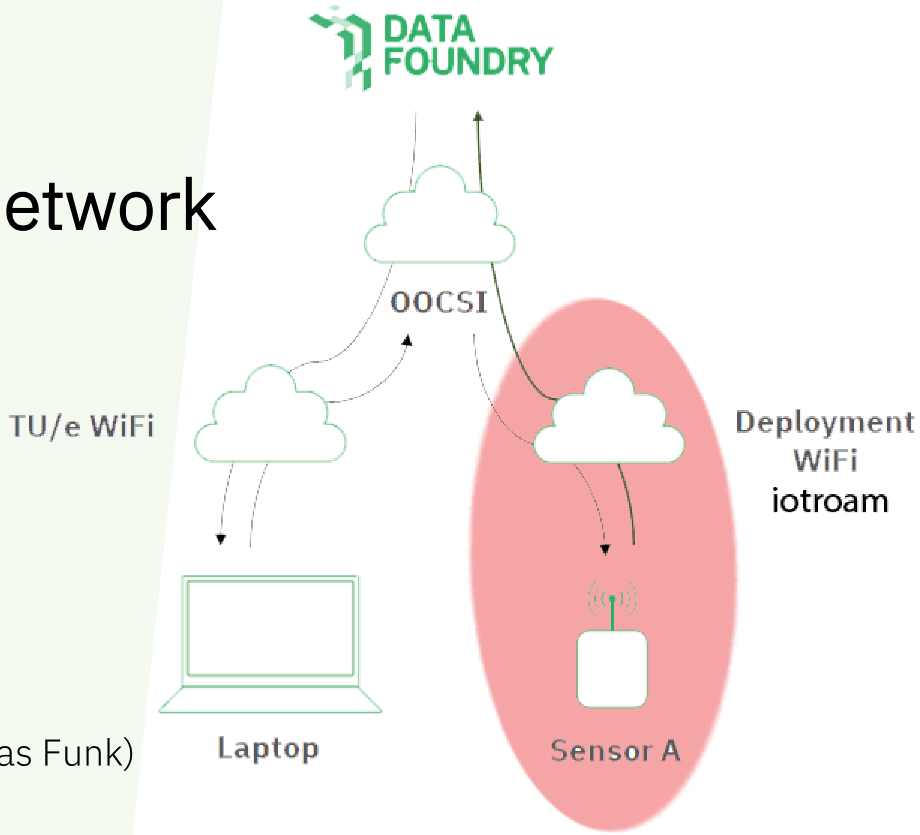
1. Preset of Data Foundry
   1. Create a project → Should be done by Data Foundry automatically
   2. Create an IoT dataset
   3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOCSI
4. Check data in OOCSI UI Client page
5. Save data to own IoT dataset



(Source from Mathias Funk)

Practice: Data collection by Data Foundry through OOCSI

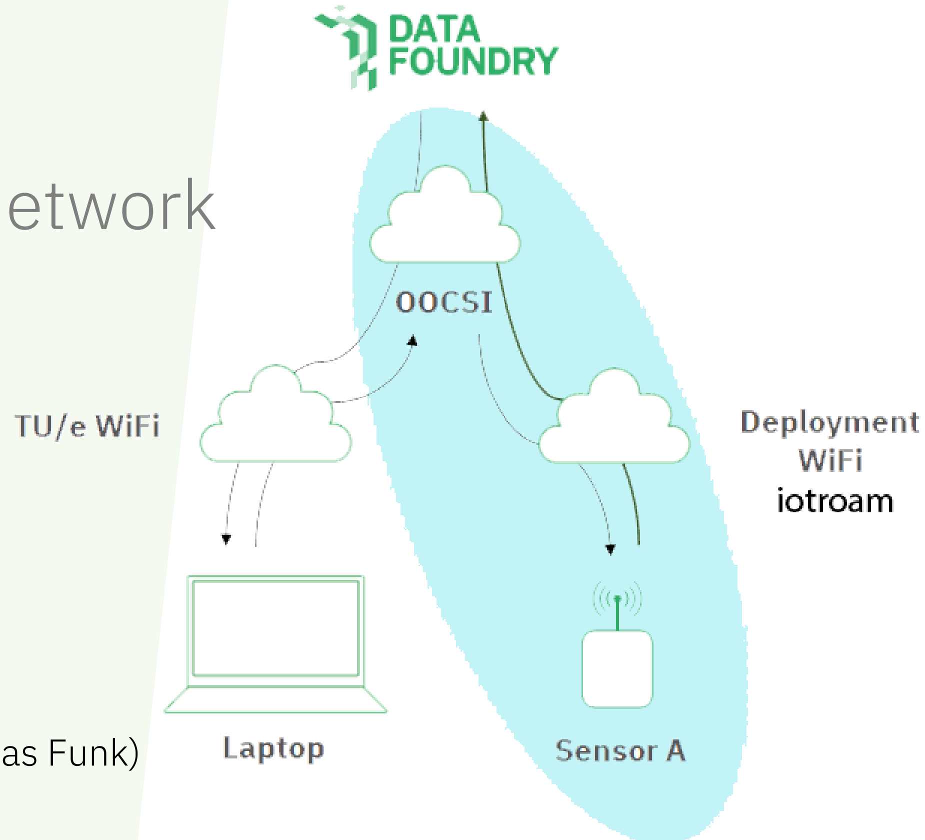
1. Preset of Data Foundry
   1. Create a project → Should be done by Data Foundry automatically
   2. Create an IoT dataset
   3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOCSI
4. Check data in OOCSI UI Client page
5. Save data to own IoT dataset



(Source from Mathias Funk)

Practice: Data collection by Data Foundry through OOCSI

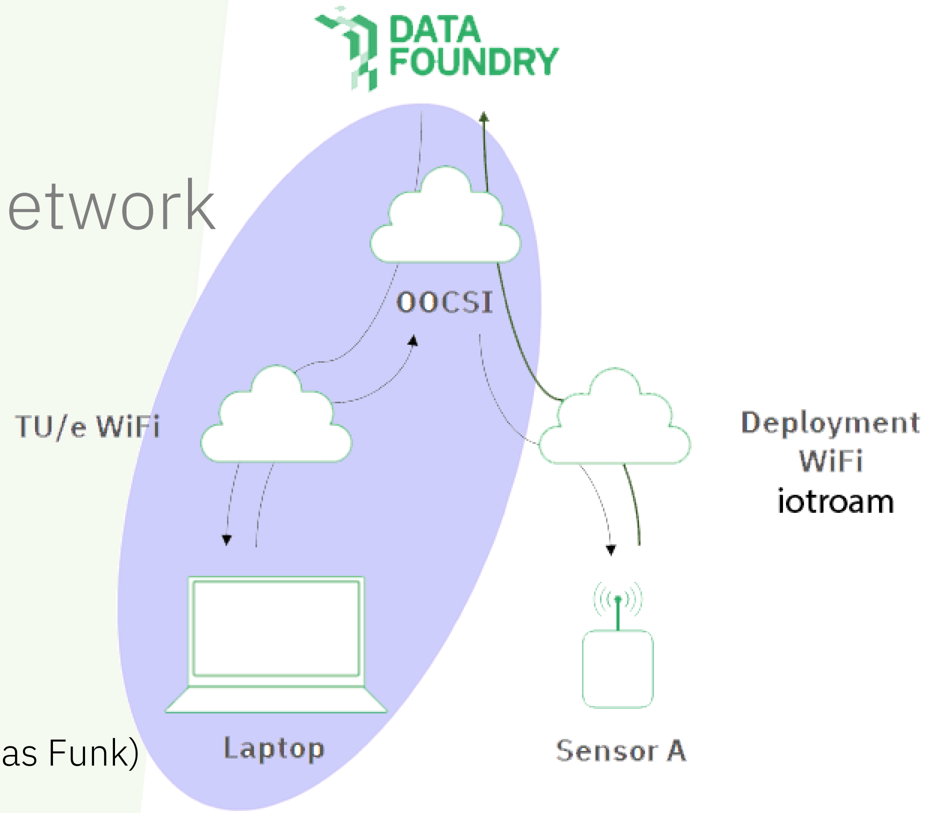
1. Preset of Data Foundry
    1. Create a project → Should be done by Data Foundry automatically
    2. Create an IoT dataset
    3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. **Send data through OOCSI**
4. Check data in OOCSI UI Client page
5. Save data to own IoT dataset



(Source from Mathias Funk)

Practice: Data collection by Data Foundry through OOCSI

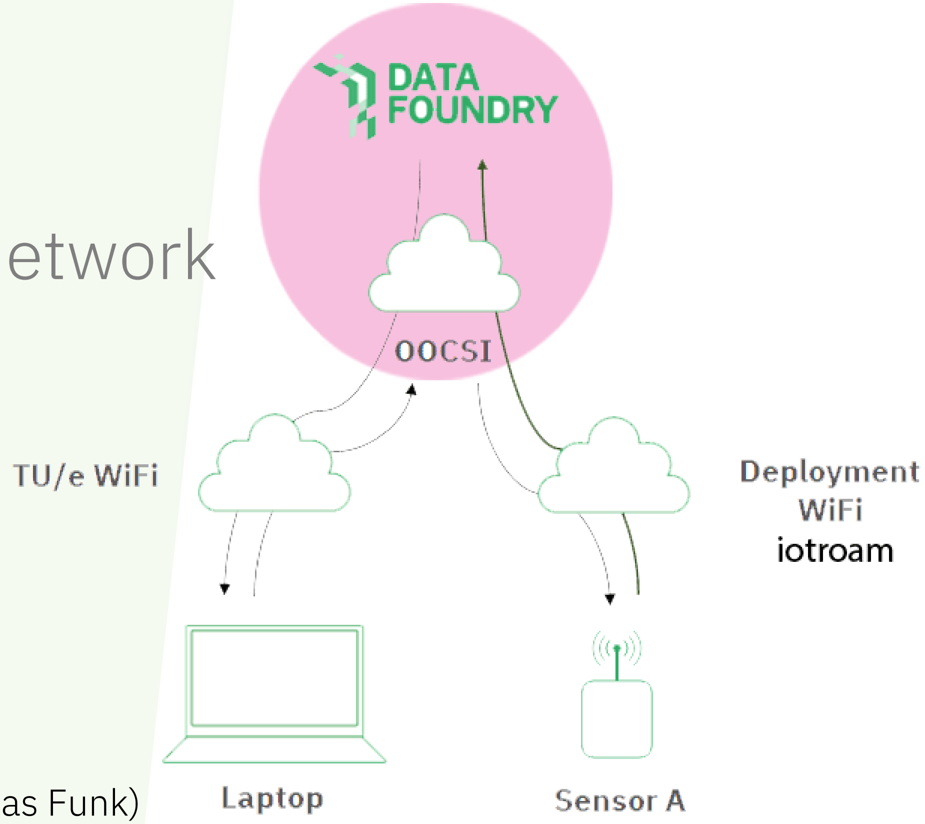
1. Preset of Data Foundry
    1. Create a project → Should be done by Data Foundry automatically
    2. Create an IoT dataset
    3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOCSI
4. **Check data in OOCSI UI Client page**
5. Save data to own IoT dataset



(Source from Mathias Funk)

Practice: Data collection by Data Foundry through OOCSI

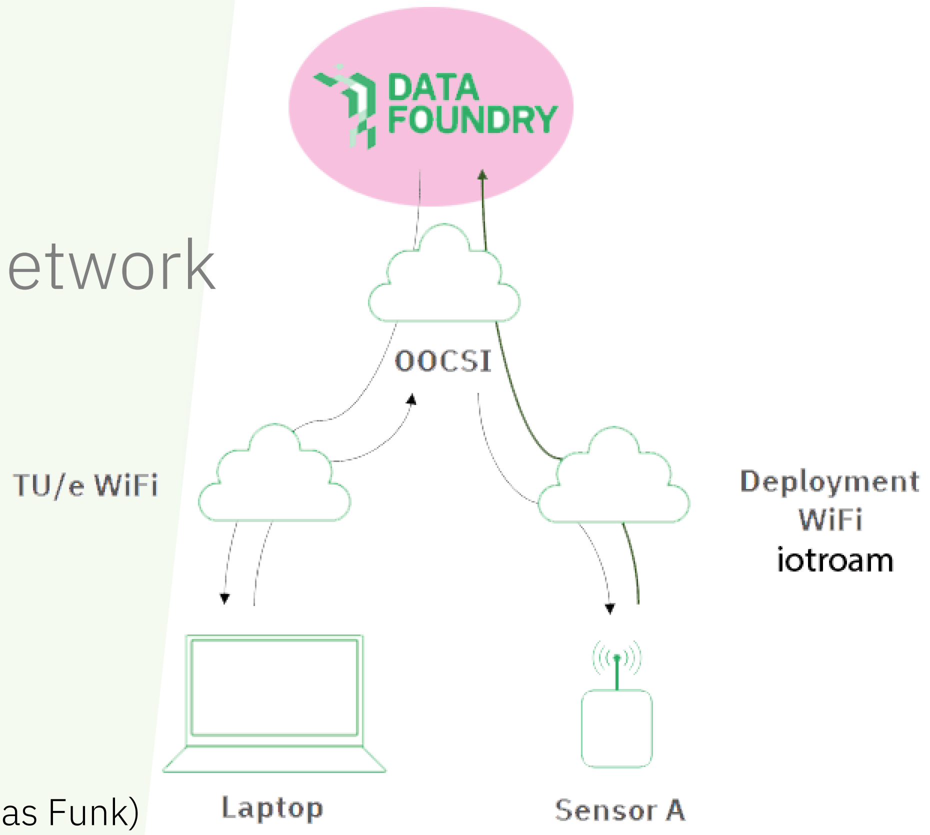
1. Preset of Data Foundry
    1. Create a project → Should be done by Data Foundry automatically
    2. Create an IoT dataset
    3. Create a device
2. Connect IoT device (ESP32) to iotroam network
3. Send data through OOCSI
4. Check data in OOCSI UI Client page
5. Save data to own IoT dataset via OOCSI



(Source from Mathias Funk)

Practice: Data collection by Data Foundry through 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

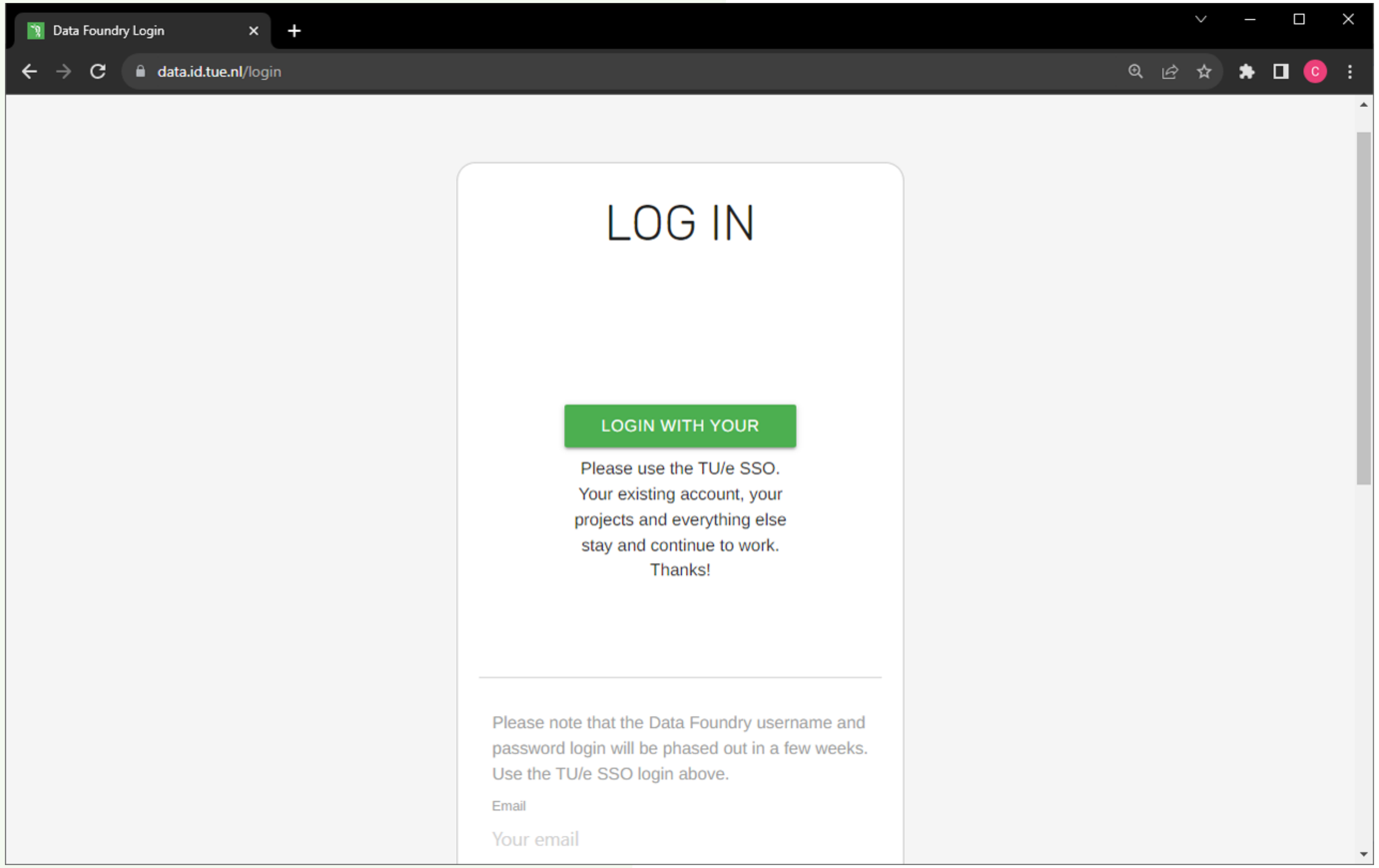


(Source from Mathias Funk)

Step 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!



Please use the TU/e SSO. Your existing account, your projects and everything else stay and continue to work. Thanks!

Please note that the Data Foundry username and password login will be phased out in a few weeks. Use the TU/e SSO login above.

Email

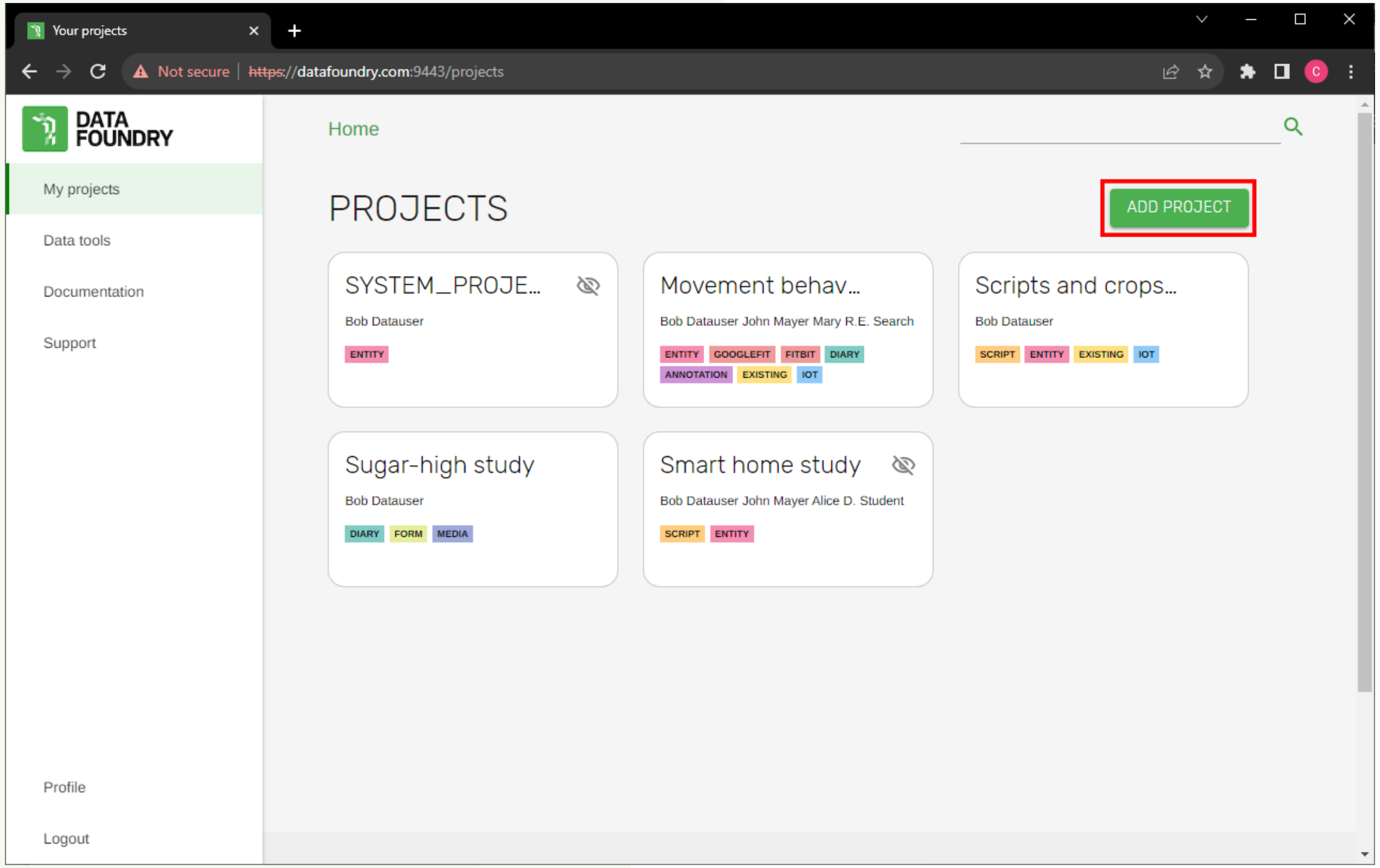
Your email

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

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

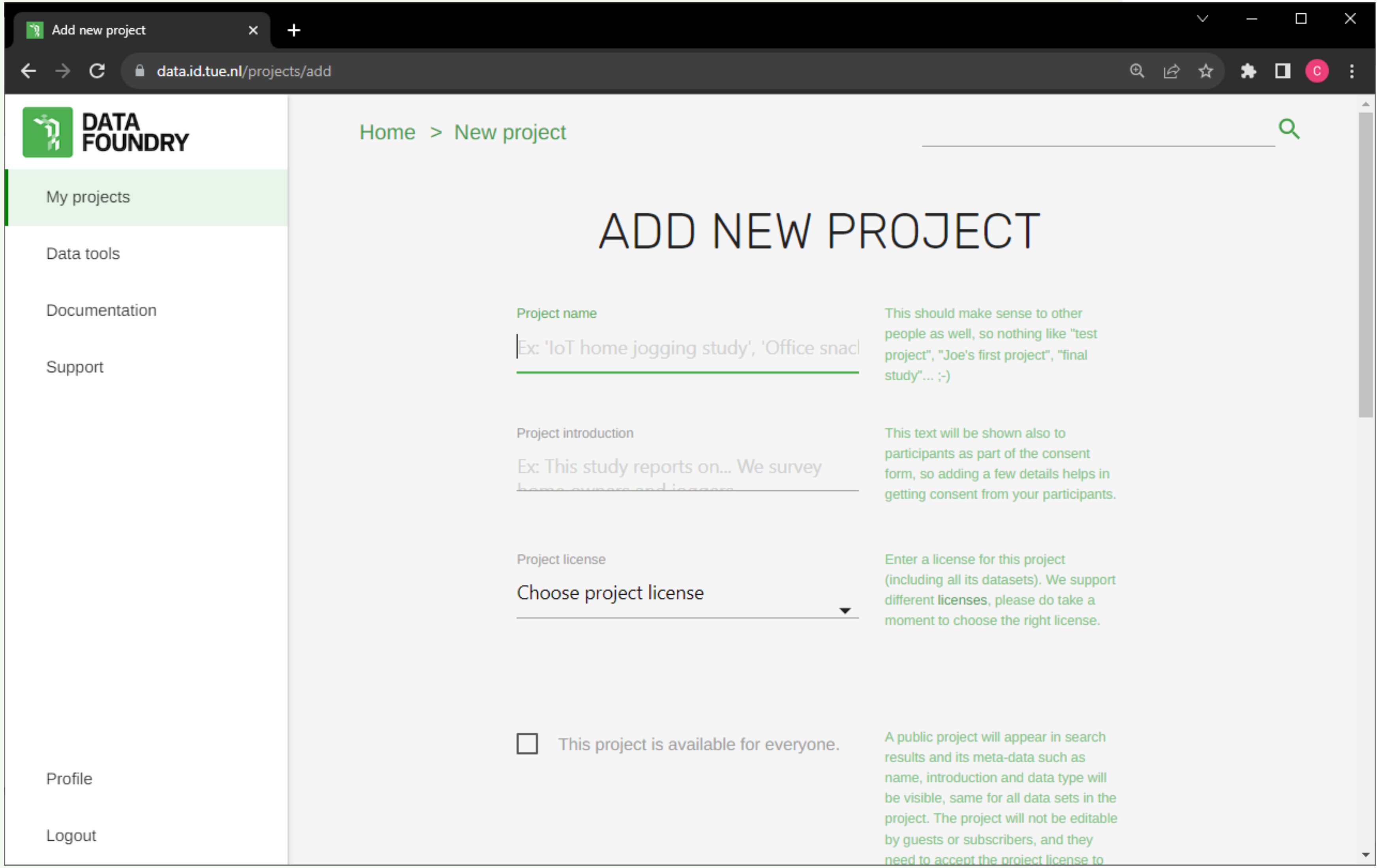
Create a project (should be done automatically after first logging in)

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)



ADD NEW PROJECT

Project name

Ex: 'IoT home jogging study', 'Office snacl

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 joggers

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)

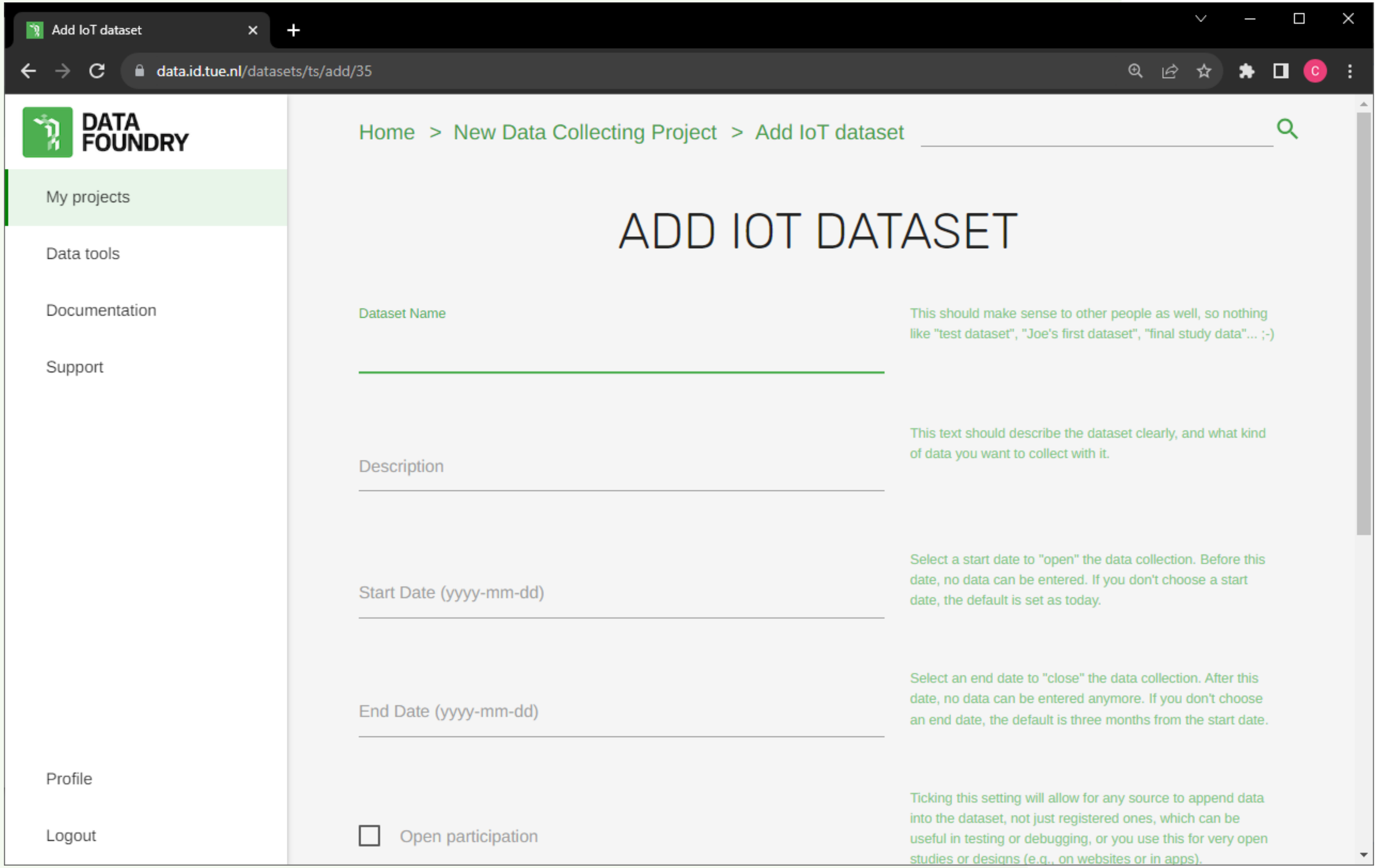
Create an IoT dataset within that project

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)



Home > New Data Collecting Project > Add IoT dataset

ADD IOT DATASET

Dataset Name

This should make sense to other people as well, so nothing like "test dataset", "Joe's first dataset", "final study data"... ;-)

Description

This text should describe the dataset clearly, and what kind of data you want to collect with it.

Start Date (yyyy-mm-dd)

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)

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

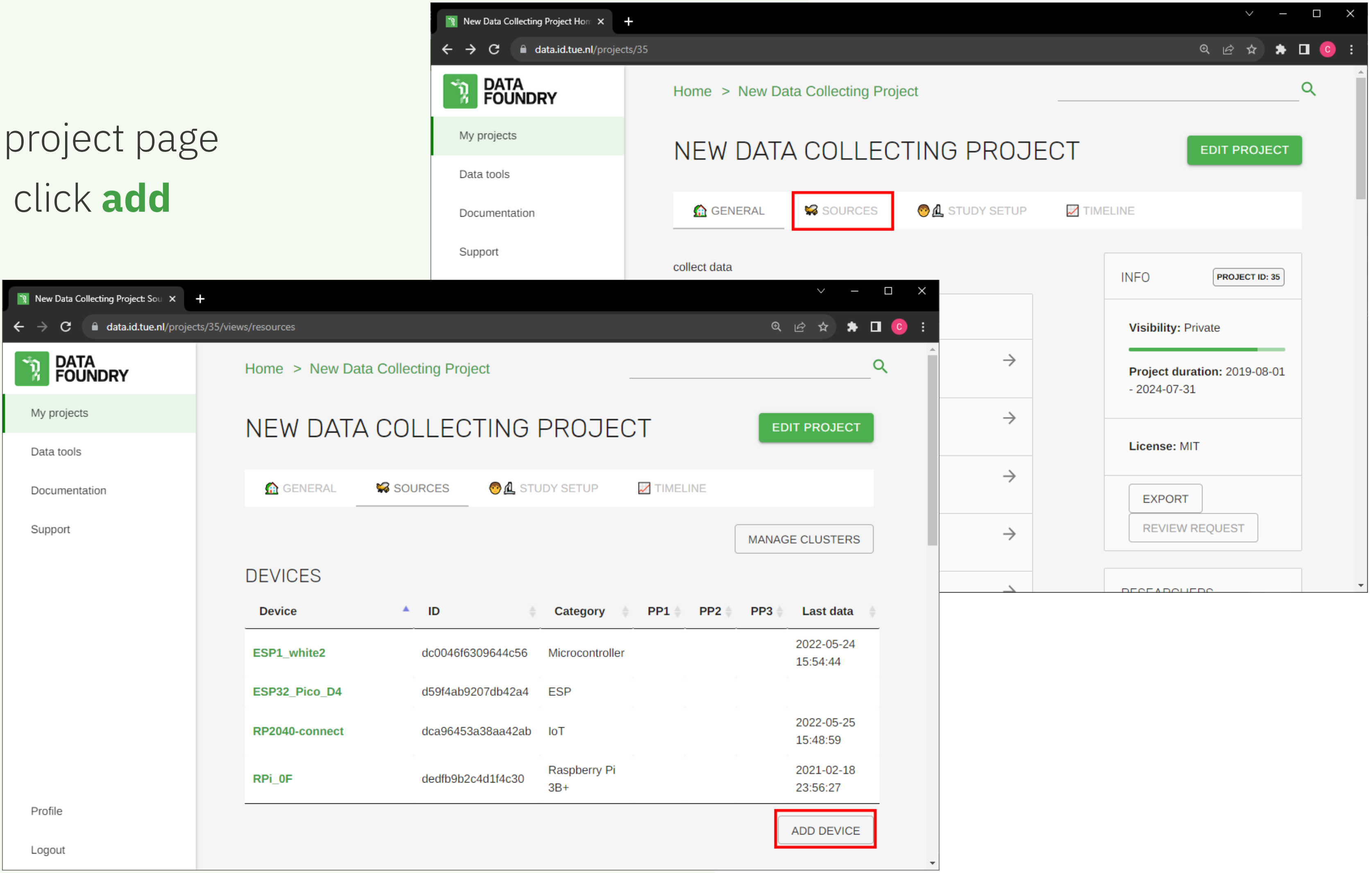
Ticking this setting will allow for any source to append data into the dataset, not just registered ones, which can be useful in testing or debugging, or you use this for very open studies or designs (e.g., on websites or in apps).

(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


New Data Collecting Project Hon
data.id.tue.nl/projects/35
DATA FOUNDRY
My projects
Data tools
Documentation
Support
Home > New Data Collecting Project
NEW DATA COLLECTING PROJECT
EDIT PROJECT
GENERAL
SOURCES
STUDY SETUP
TIMELINE
INFO
PROJECT ID: 35
EXPORT
REVIEW REQUEST
New Data Collecting Project: Sou
data.id.tue.nl/projects/35/views/resources
Profile
Logout
MANAGE CLUSTERS
DEVICES
ADD DEVICE

collect data

Visibility: Private

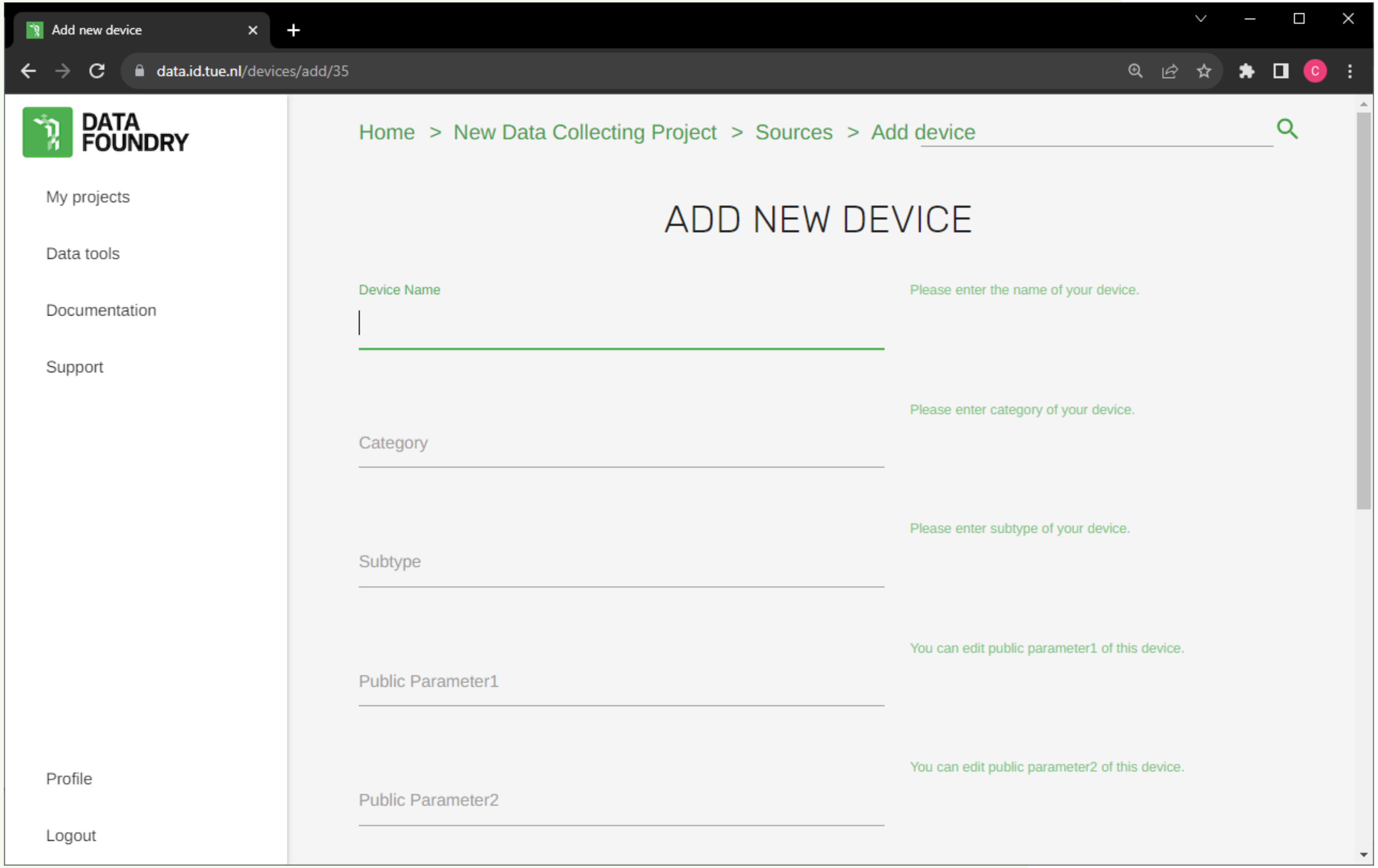
Project duration: 2019-08-01 - 2024-07-31

License: MIT

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 1.3 - CREATING YOUR FIRST DEVICE (2/2)



Add new device
data.id.tue.nl/devices/add/35
DATA FOUNDRY
My projects
Data tools
Documentation
Support
Profile
Logout
Home > New Data Collecting Project > Sources > Add device

ADD NEW DEVICE

Device Name — Please enter the name of your device.

Category — Please enter category of your device.

Subtype — Please enter subtype of your device.

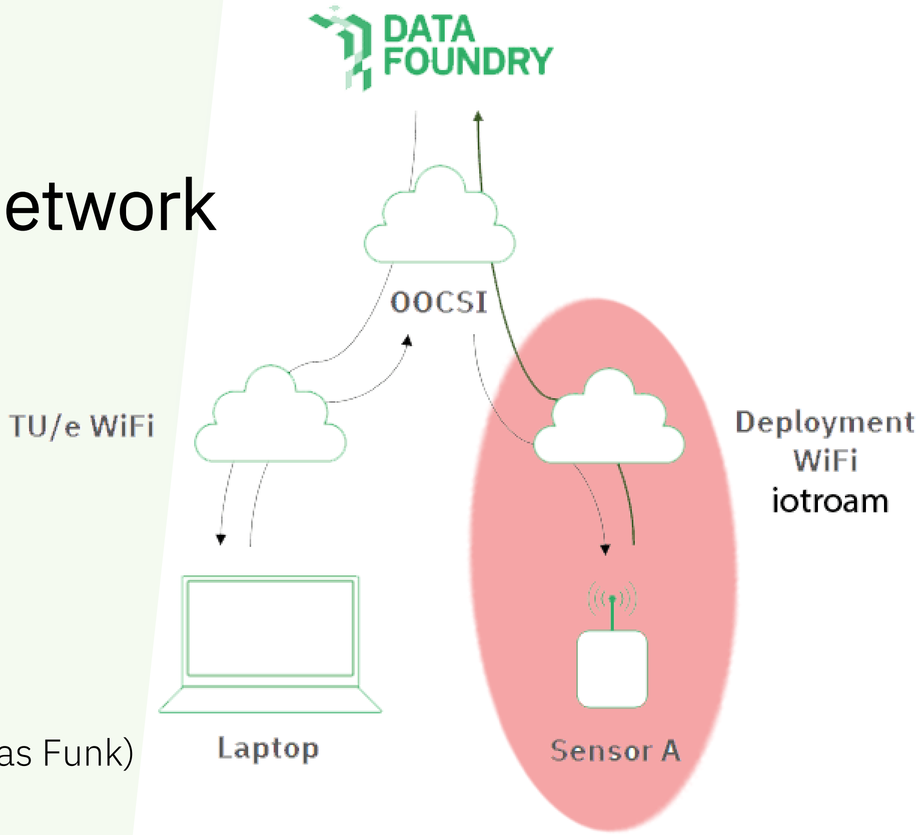
Public Parameter1 — You can edit public parameter1 of this device.

Public Parameter2 — 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)

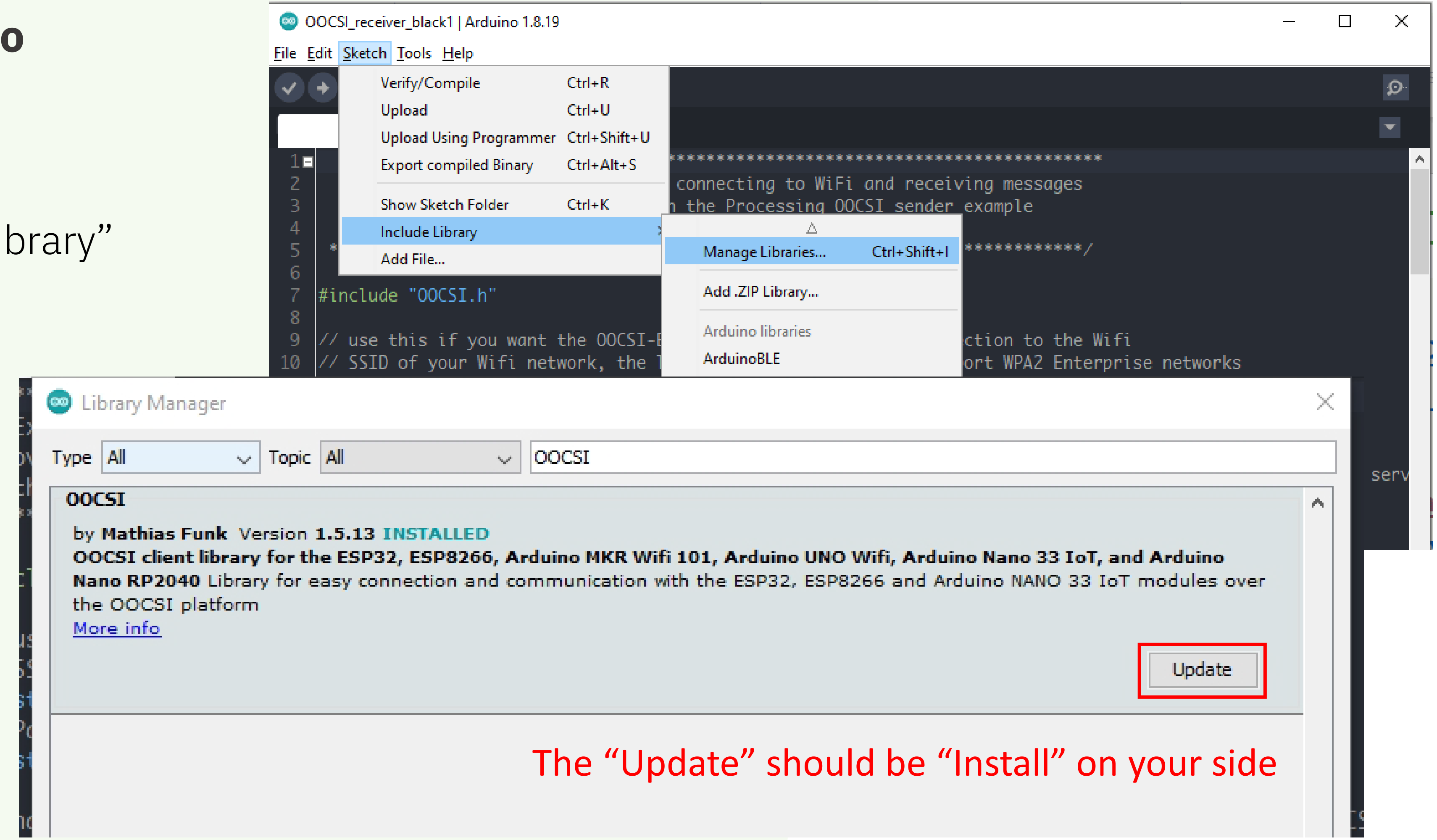
Step 2 – CONNECT IOT DEVICE (ESP32) (1/5)

- 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 OOCSI library to Arduino IDE

1. Open Arduino IDE
2. "Sketch" -> "Include Library" -> "Manage Libraries"
3. Search "OOCSI"
4. Click "Install"
5. More information



OOCSI_receiver_black1 | Arduino 1.8.19
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
ArduinoBLE
Library Manager
Type All
Topic All
OOCSI
Update

```
 ********************************************
 connecting to WiFi and receiving messages
 n the Processing OOCSI sender example

 *                                    *************/

#include "OOCSI.h"

// use this if you want the OOCSI-E                    ction to the Wifi
// SSID of your Wifi network, the                     ort WPA2 Enterprise networks
```

OOCSI

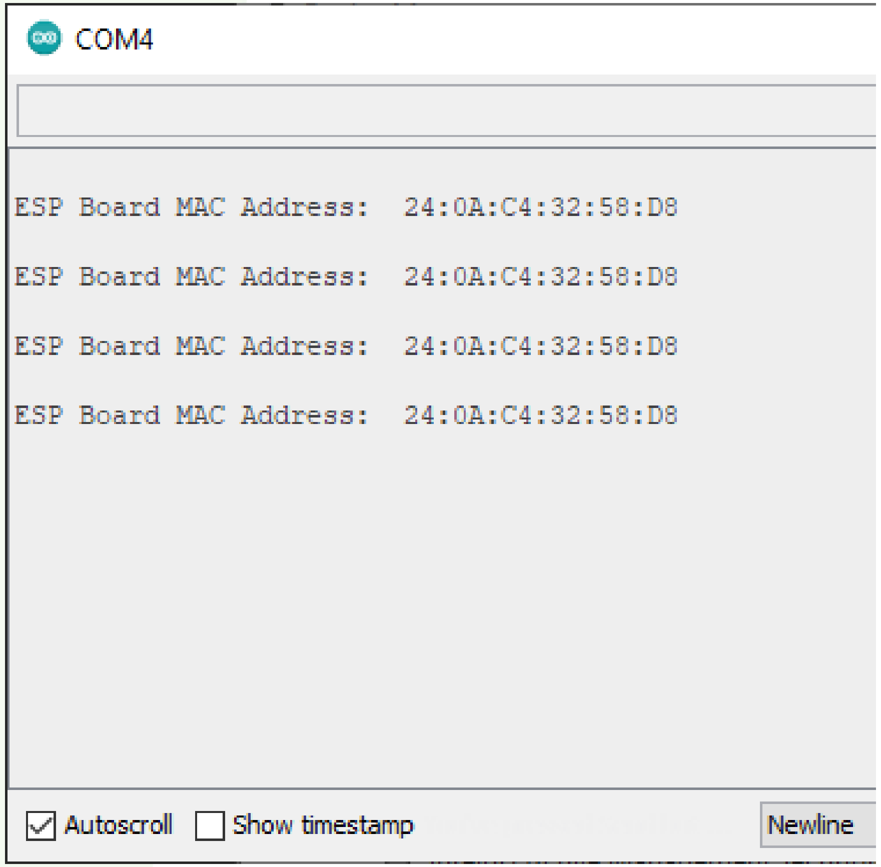
by **Mathias Funk** Version **1.5.13** **INSTALLED**
OOCSI client library for the ESP32, ESP8266, Arduino MKR Wifi 101, Arduino UNO Wifi, Arduino Nano 33 IoT, and Arduino Nano RP2040 Library for easy connection and communication with the ESP32, ESP8266 and Arduino NANO 33 IoT modules over the OOCSI platform
More info

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);
}
```



COM4

```
ESP Board MAC Address:  24:0A:C4:32:58:D8

ESP Board MAC Address:  24:0A:C4:32:58:D8

ESP Board MAC Address:  24:0A:C4:32:58:D8

ESP Board MAC Address:  24:0A:C4:32:58:D8
```

☑ Autoscroll □ Show timestamp Newline

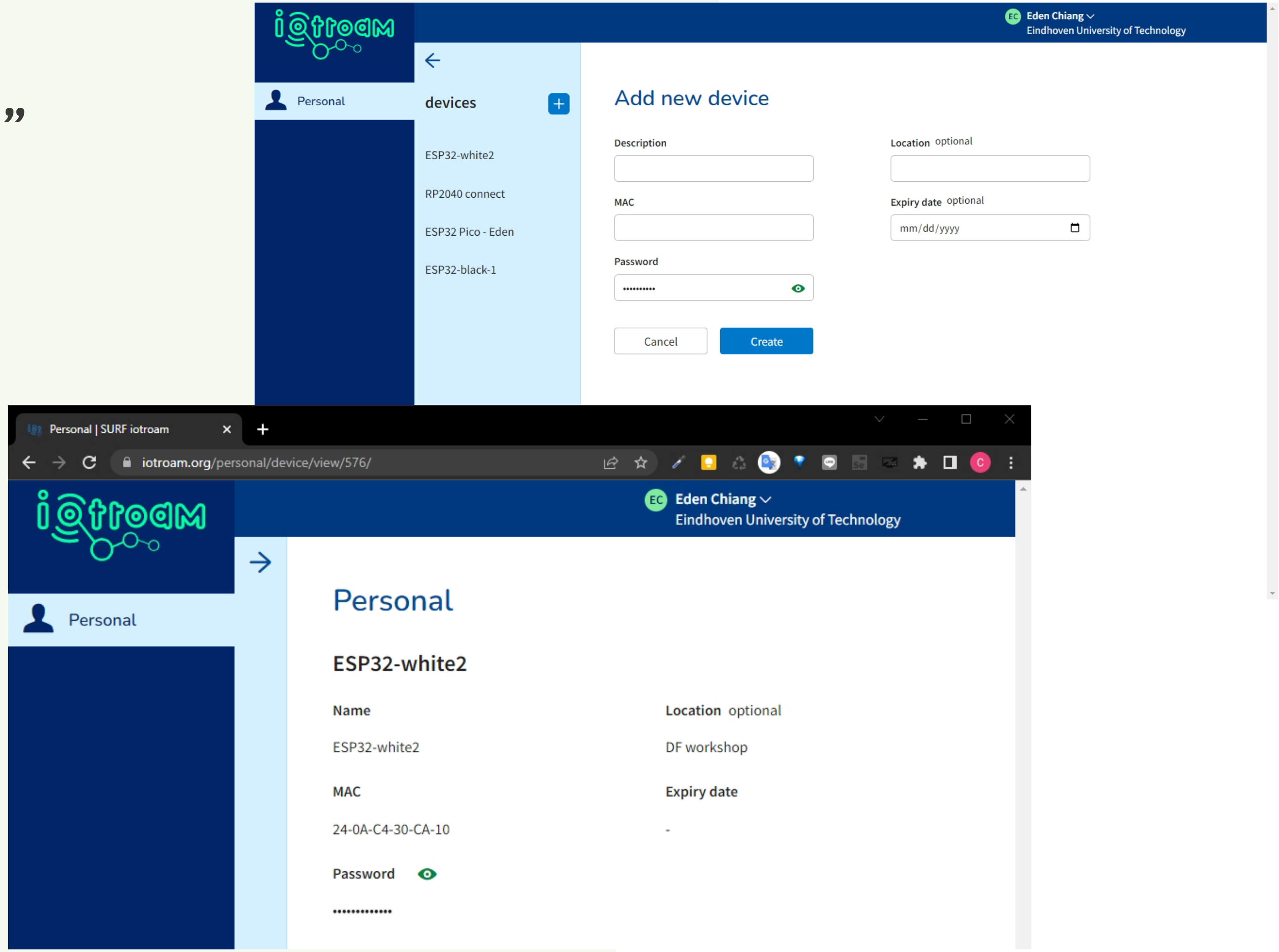
Step 2 – CONNECT IOT DEVICE (ESP32) (4/5)

- 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. Click “Create”
6. Copy the **password**



Add new device

Description

Location optional

MAC

Expiry date optional: mm/dd/yyyy

Password: ••••••••••

Cancel
Create
Personal | SURF iotroam
iotroam.org/personal/device/view/576/
iotroam
Eden Chiang
Eindhoven University of Technology
Personal

Personal

ESP32-white2

Name: ESP32-white2

Location optional: DF workshop

MAC: 24-0A-C4-30-CA-10

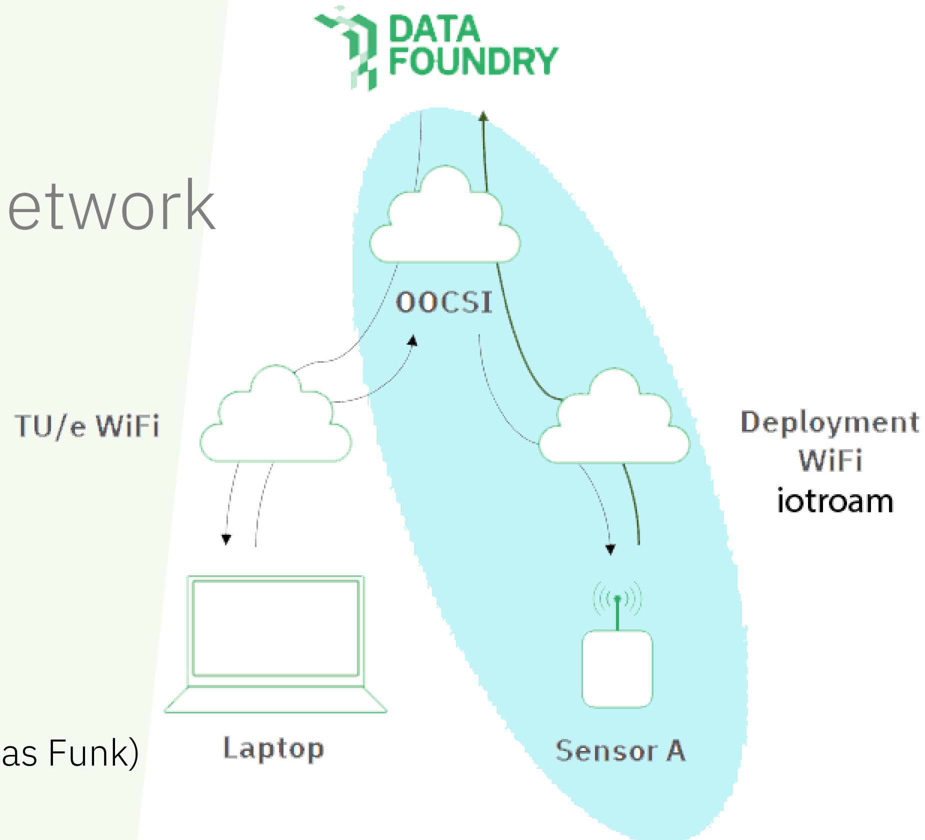
Expiry date: -

Password: ••••••••••••••

(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)

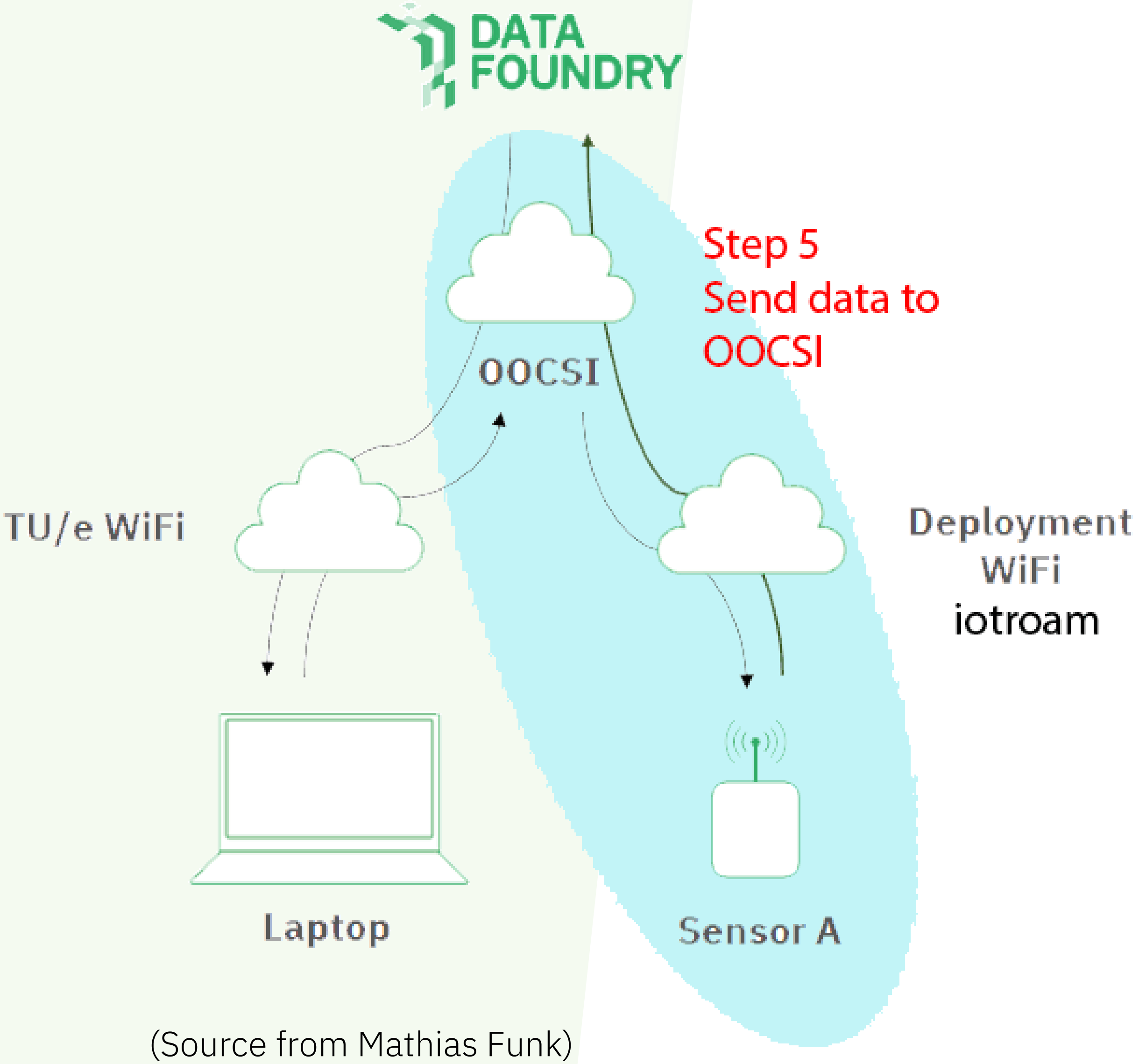
What is OOCSI?

"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



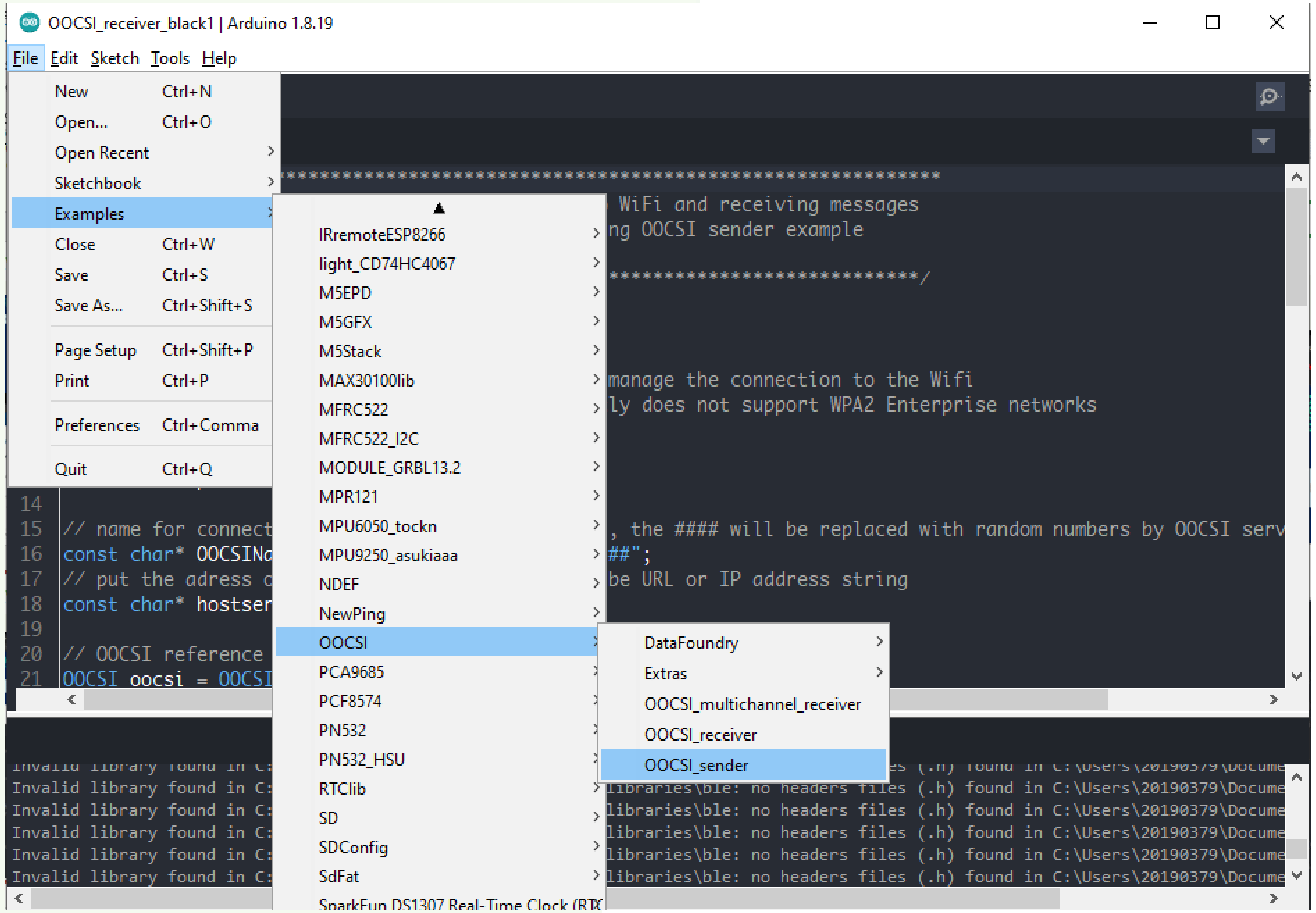
Step 5
Send data to OOCSI

(Source from Mathias Funk)

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

Open example code of sending data to Data Foundry through OOCSI

1. Open Arduino IDE
2. “File” -> “Examples” -> “OOCSI” -> “OOCSI_sender”

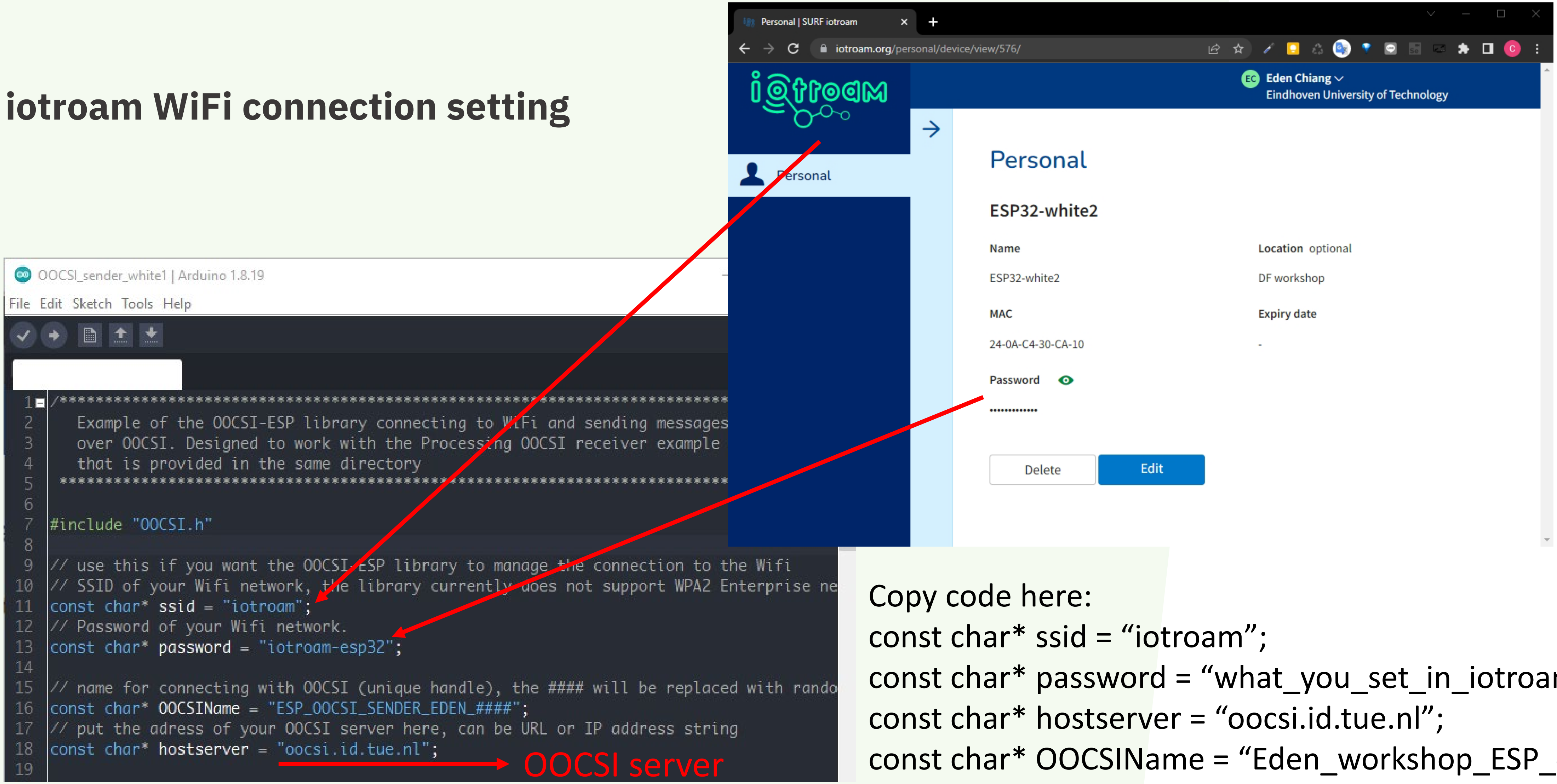


```
*****************************************************************
 WiFi and receiving messages
ng OOCSI sender example
*****************************/

manage the connection to the Wifi
ly does not support WPA2 Enterprise networks

// name for connect                  , the #### will be replaced with random numbers by OOCSI serv
const char* OOCSINa                  ##";
// put the adress o                  be URL or IP address string
const char* hostser

// OOCSI reference
OOCSI oocsi = OOCSI
```

```
Invalid library found in C:          libraries\ble: no headers files (.h) found in C:\Users\20190379\Docume
Invalid library found in C:          libraries\ble: no headers files (.h) found in C:\Users\20190379\Docume
Invalid library found in C:          libraries\ble: no headers files (.h) found in C:\Users\20190379\Docume
Invalid library found in C:          libraries\ble: no headers files (.h) found in C:\Users\20190379\Docume
Invalid library found in C:          libraries\ble: no headers files (.h) found in C:\Users\20190379\Docume
```

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

iotroam WiFi connection setting



OOCSI_sender_white1 | Arduino 1.8.19

```
/*****************************************************************************
   Example of the OOCSI-ESP library connecting to WiFi and sending messages
   over OOCSI. Designed to work with the Processing OOCSI receiver example
   that is provided in the same directory
 *****************************************************************************

#include "OOCSI.h"

// use this if you want the OOCSI-ESP library to manage the connection to the Wifi
// SSID of your Wifi network, the library currently does not support WPA2 Enterprise ne
const char* ssid = "iotroam";
// Password of your Wifi network.
const char* password = "iotroam-esp32";

// name for connecting with OOCSI (unique handle), the #### will be replaced with rando
const char* OOCSIName = "ESP_OOCSI_SENDER_EDEN_####";
// put the adress of your OOCSI server here, can be URL or IP address string
const char* hostserver = "oocsi.id.tue.nl";
```

OOCSI server

Personal | SURF iotroam

iotroam.org/personal/device/view/576/

Eden Chiang
Eindhoven University of Technology

Personal

ESP32-white2

Name: ESP32-white2

Location optional: DF workshop

MAC: 24-0A-C4-30-CA-10

Expiry date: -

Password: ••••••••••••

Copy code here:

const char* ssid = “iotroam”;

const char* password = “what_you_set_in_iotroam”;

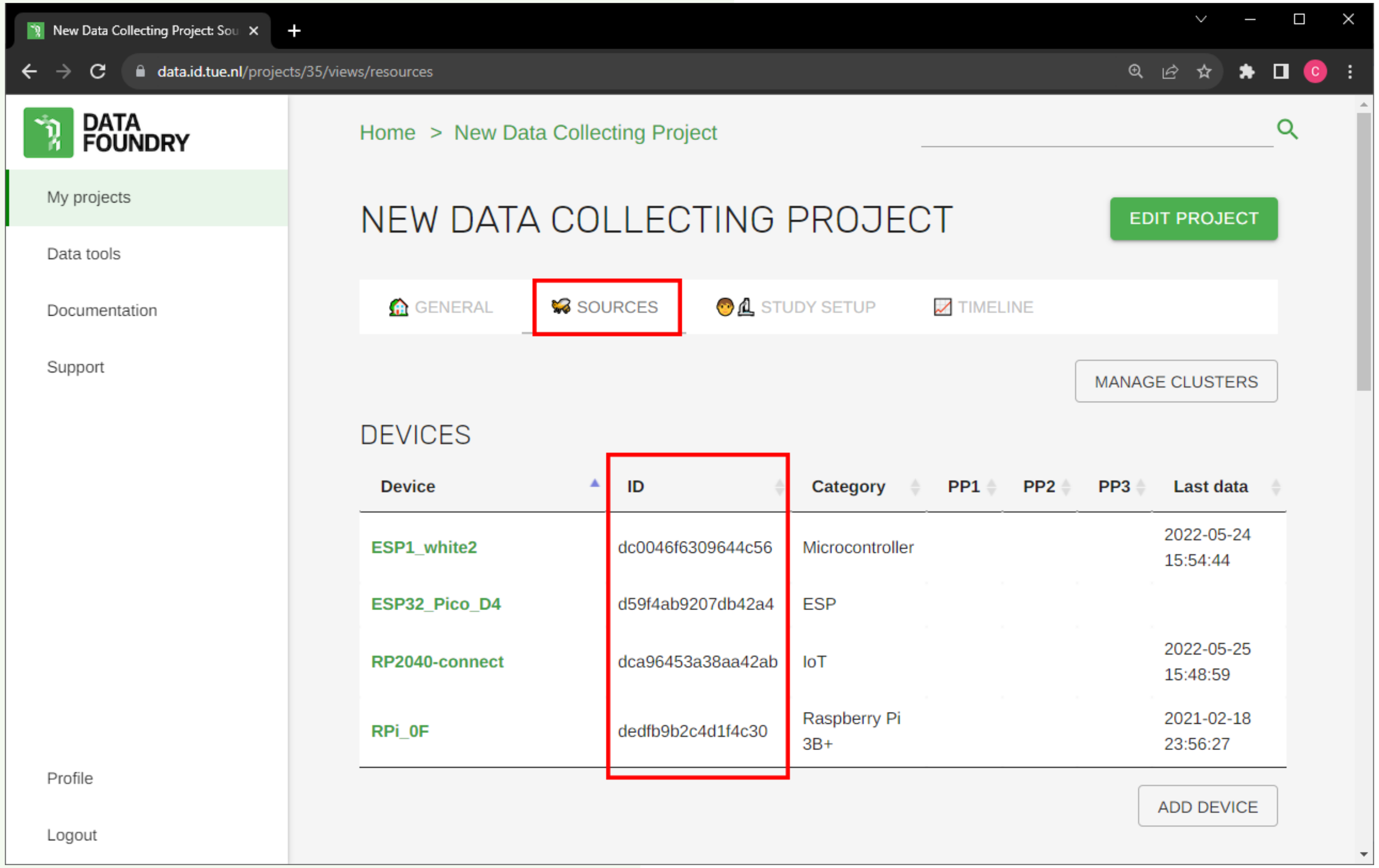
const char* hostserver = “oocsi.id.tue.nl”;

const char* OOCSIName = “Eden_workshop_ESP_####”;

Step 3 – SEND DATA THROUGH OOCSI (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



NEW DATA COLLECTING PROJECT

DEVICES

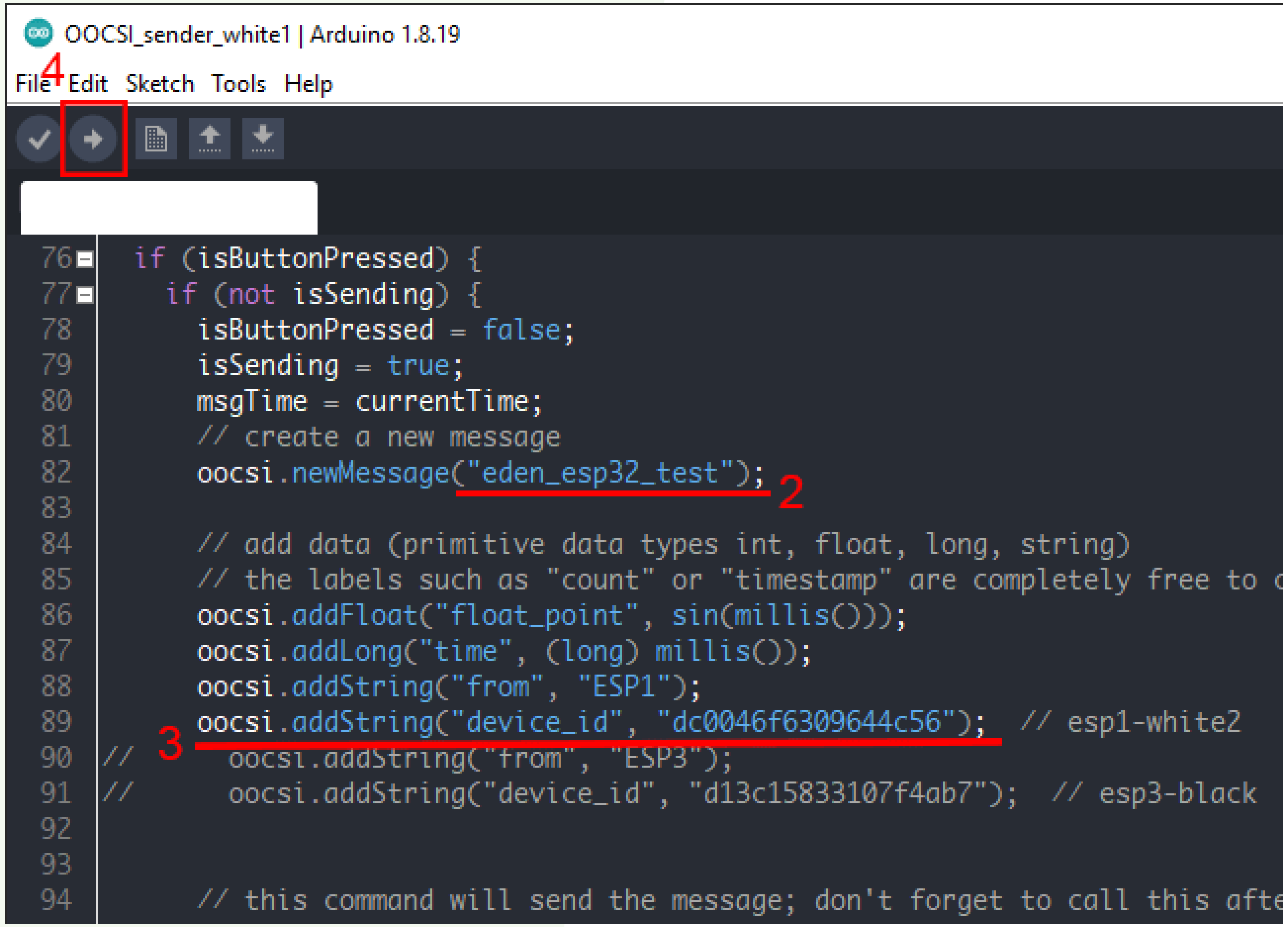
Device	ID	Category	PP1	PP2	PP3	Last data
ESP1_white2	dc0046f6309644c56	Microcontroller				2022-05-24 15:54:44
ESP32_Pico_D4	d59f4ab9207db42a4	ESP				
RP2040-connect	dca96453a38aa42ab	IoT				2022-05-25 15:48:59
RPi_0F	dedfb9b2c4d1f4c30	Raspberry Pi 3B+				2021-02-18 23:56:27

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

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

Configuration for OOCSI in code

1. Back to OOCSI_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

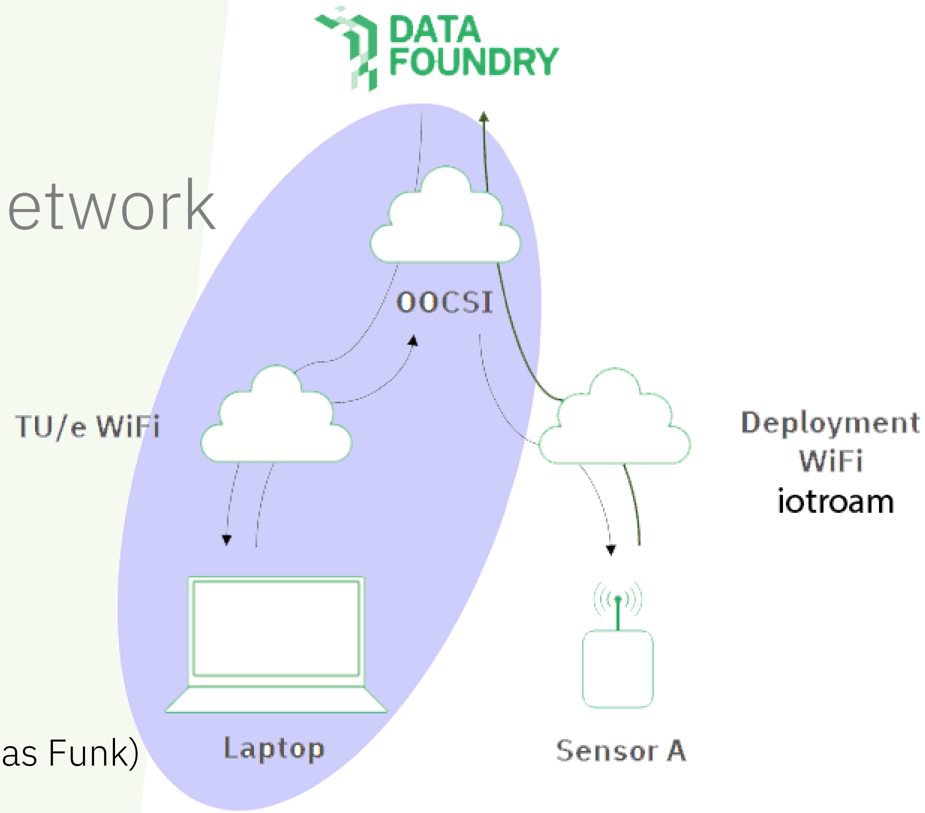

OOCSI_sender_white1 | Arduino 1.8.19
File Edit Sketch Tools Help
4
2
3

```
  if (isButtonPressed) {
    if (not isSending) {
      isButtonPressed = false;
      isSending = true;
      msgTime = currentTime;
      // create a new message
      oocsi.newMessage("eden_esp32_test");

      // add data (primitive data types int, float, long, string)
      // the labels such as "count" or "timestamp" are completely free to c
      oocsi.addFloat("float_point", sin(millis()));
      oocsi.addLong("time", (long) millis());
      oocsi.addString("from", "ESP1");
      oocsi.addString("device_id", "dc0046f6309644c56");  // esp1-white2
//      oocsi.addString("from", "ESP3");
//      oocsi.addString("device_id", "d13c15833107f4ab7");  // esp3-black


      // this command will send the message; don't forget to call this afte
```

# Copy code here:
oocsi.newMessage("eden_esp32_test");
oocsi.addString("device_id", "your_device_reference_ID");

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

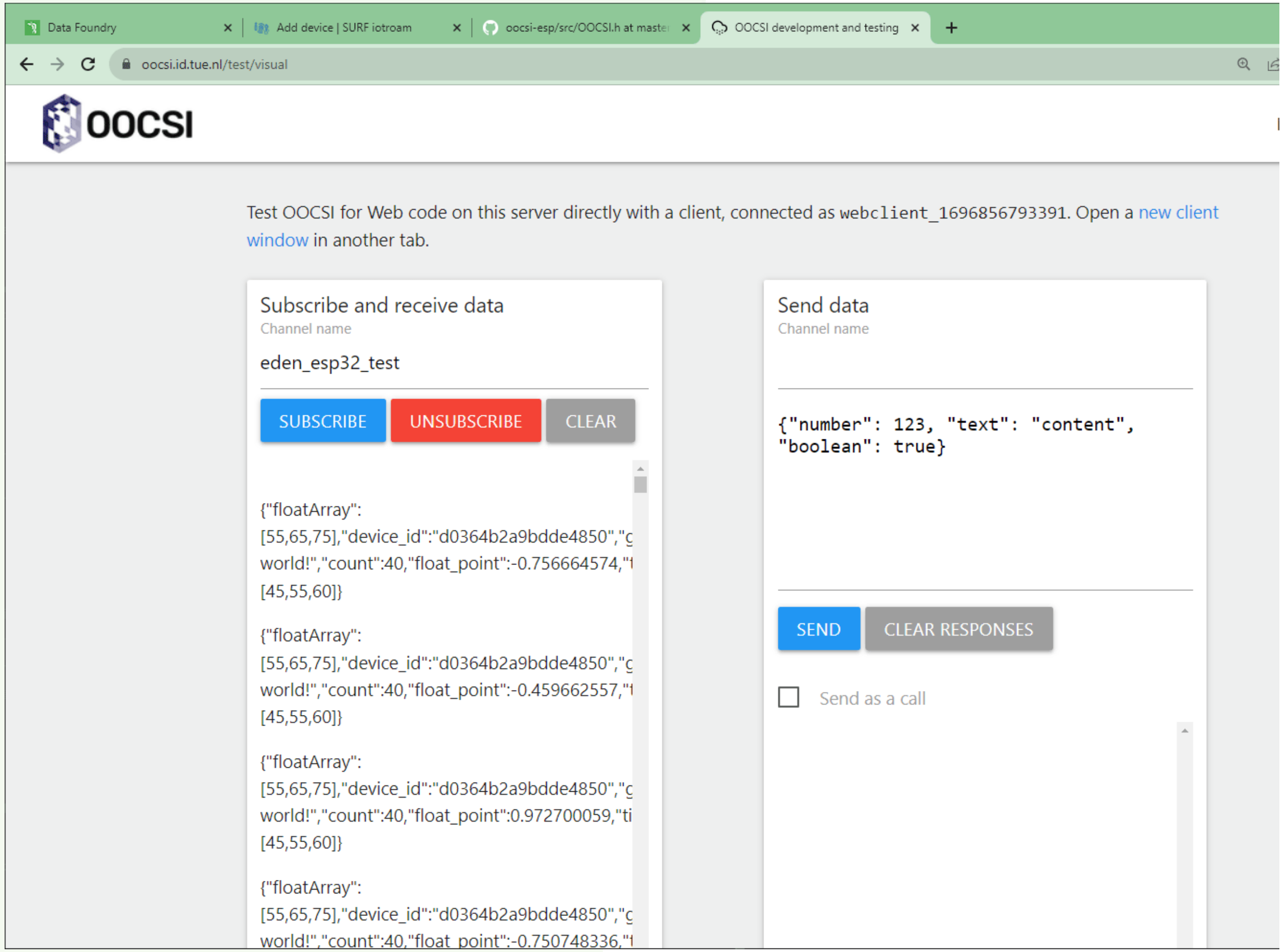


(Source from Mathias Funk)

Step 4 – CHECK DATA ON OOCSI UI CLIENT PAGE

Check data on UI Client page

1. Open OOCSI 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


Data Foundry
Add device | SURF iotroam
oocsi-esp/src/OOCSI.h at master
OOCSI development and testing
oocsi.id.tue.nl/test/visual
OOCSI
Subscribe and receive data
Channel name
SUBSCRIBE
UNSUBSCRIBE
CLEAR
Send data
Channel name
SEND
CLEAR RESPONSES
Send as a call

Test OOCSI for Web code on this server directly with a client, connected as `webclient_1696856793391`. Open a new client window in another tab.

Channel name: eden_esp32_test

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

{"floatArray":
[55,65,75],"device_id":"d0364b2a9bdde4850","g
world!","count":40,"float_point":-0.459662557,"t
[45,55,60]}

{"floatArray":
[55,65,75],"device_id":"d0364b2a9bdde4850","g
world!","count":40,"float_point":0.972700059,"ti
[45,55,60]}

{"floatArray":
[55,65,75],"device_id":"d0364b2a9bdde4850","g
world!","count":40,"float_point":-0.750748336,"t
```

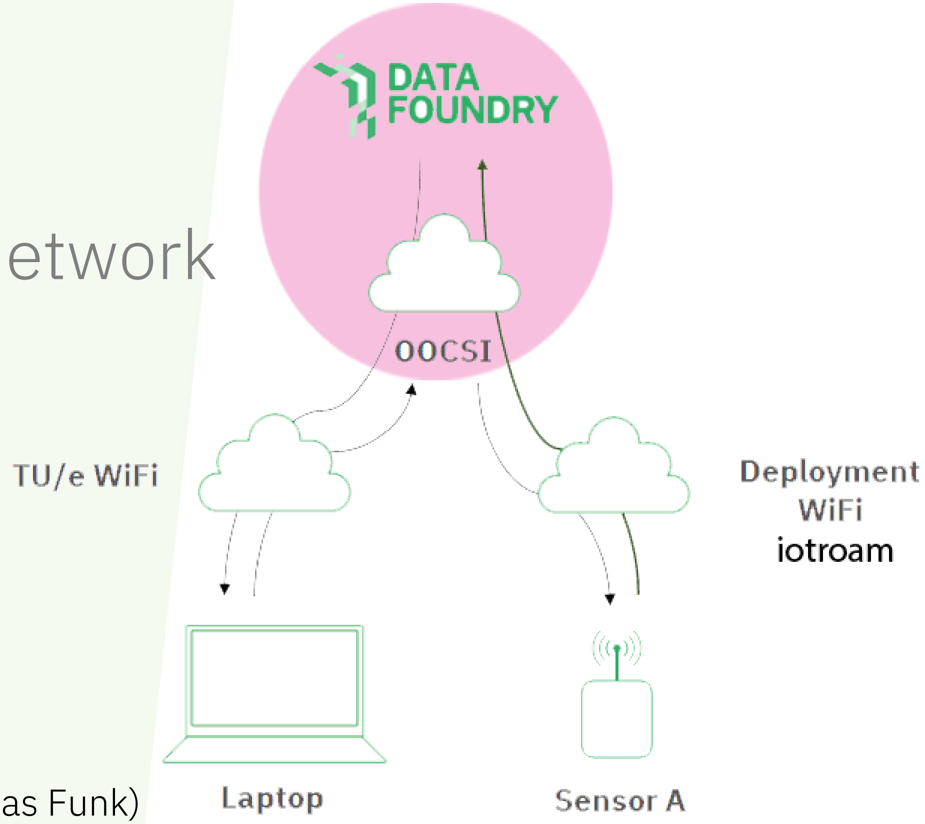
```
{"number": 123, "text": "content",
"boolean": true}
```

□ 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

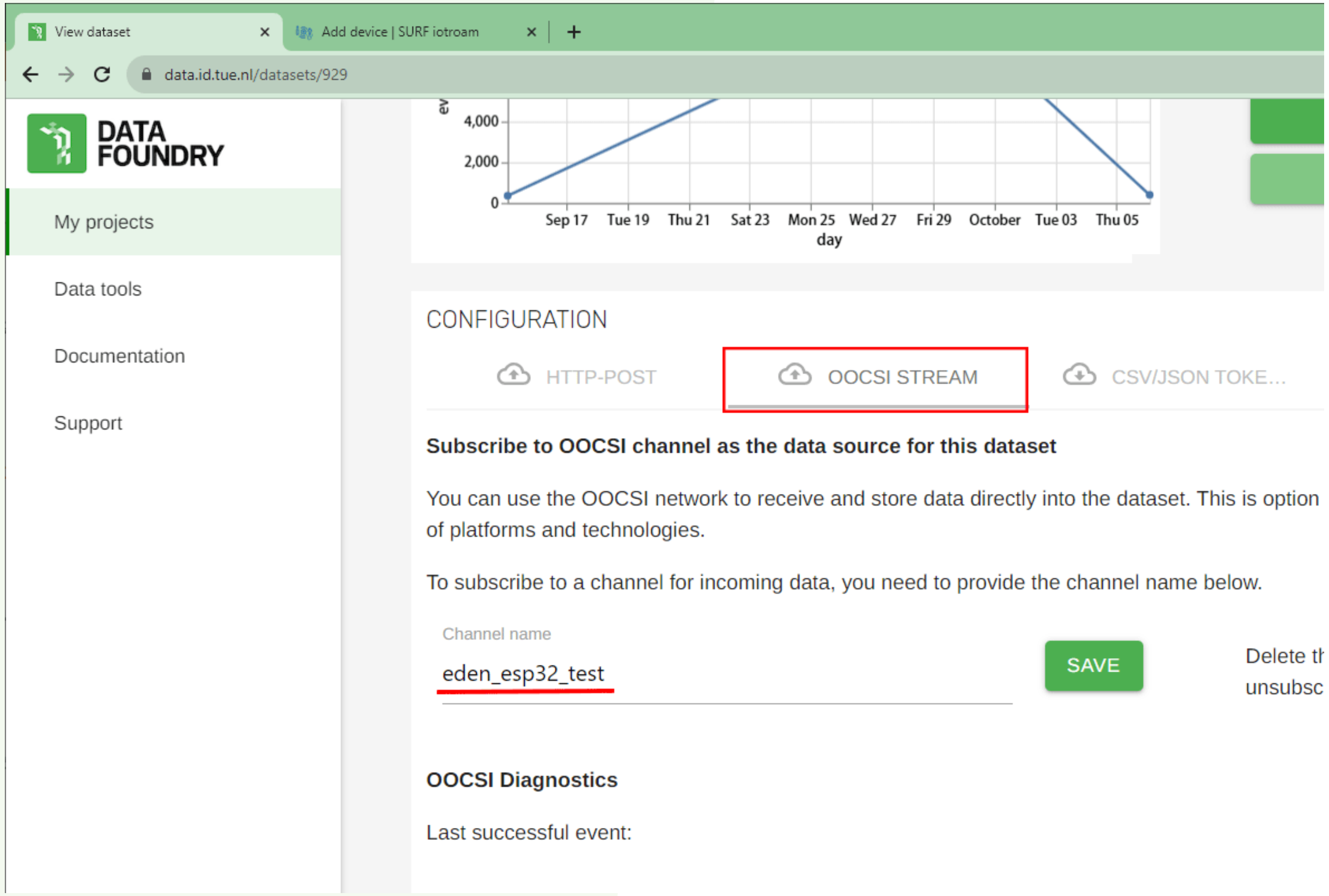


(Source from Mathias Funk)

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”



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

Delete th unsubsc

OOCSI Diagnostics

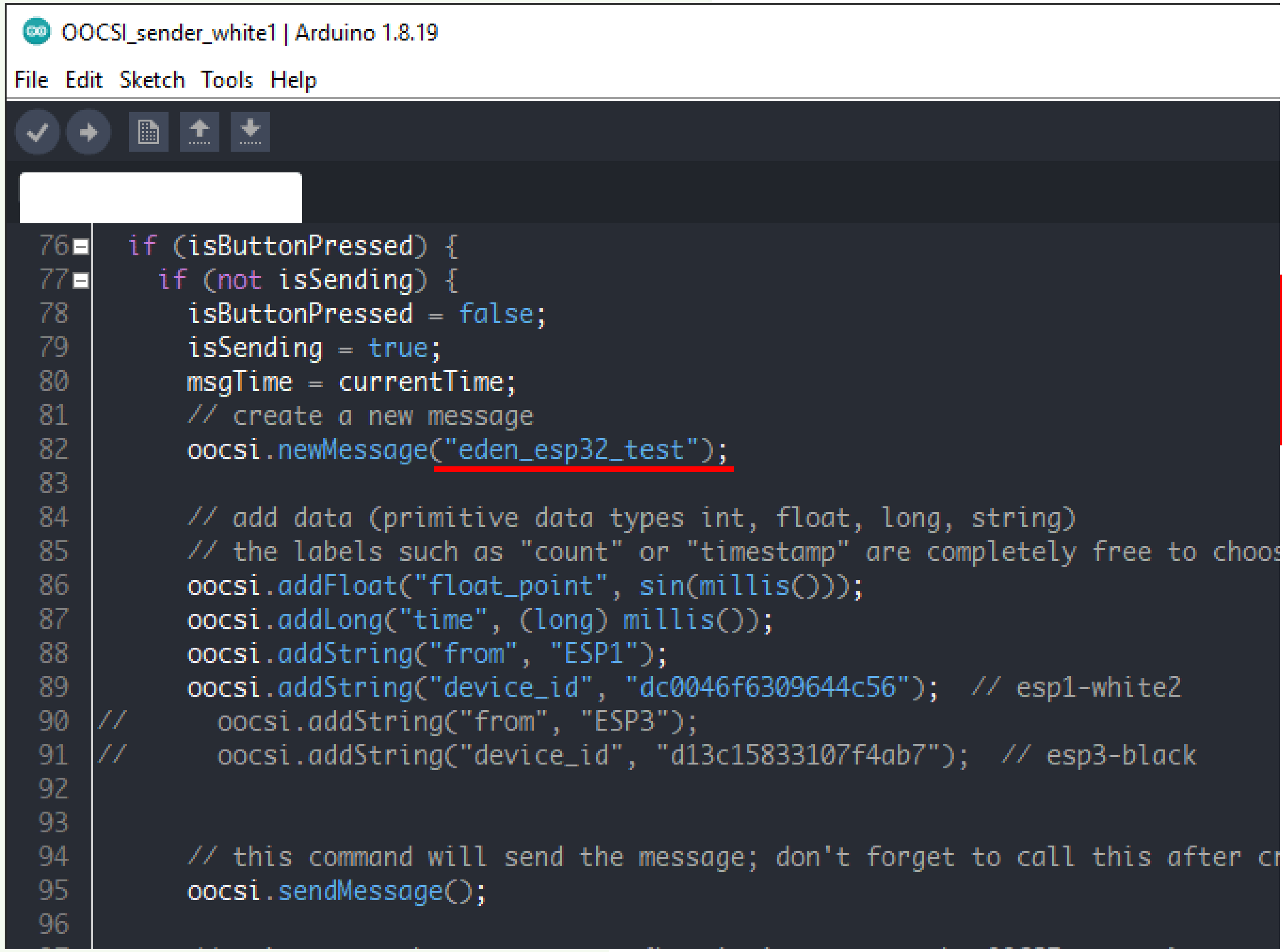
Last successful event:

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

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

Configuration with OOCSI channel in OOCSI_sender code

1. Back to OOCSI_sender code
2. Update the OOCSI 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

```
 if (isButtonPressed) {
  if (not isSending) {
   isButtonPressed = false;
   isSending = true;
   msgTime = currentTime;
   // create a new message
   oocsi.newMessage("eden_esp32_test");

   // add data (primitive data types int, float, long, string)
   // the labels such as "count" or "timestamp" are completely free to choos
   oocsi.addFloat("float_point", sin(millis()));
   oocsi.addLong("time", (long) millis());
   oocsi.addString("from", "ESP1");
   oocsi.addString("device_id", "dc0046f6309644c56");  // esp1-white2
//    oocsi.addString("from", "ESP3");
//    oocsi.addString("device_id", "d13c15833107f4ab7");  // esp3-black


   // this command will send the message; don't forget to call this after cr
   oocsi.sendMessage();

```

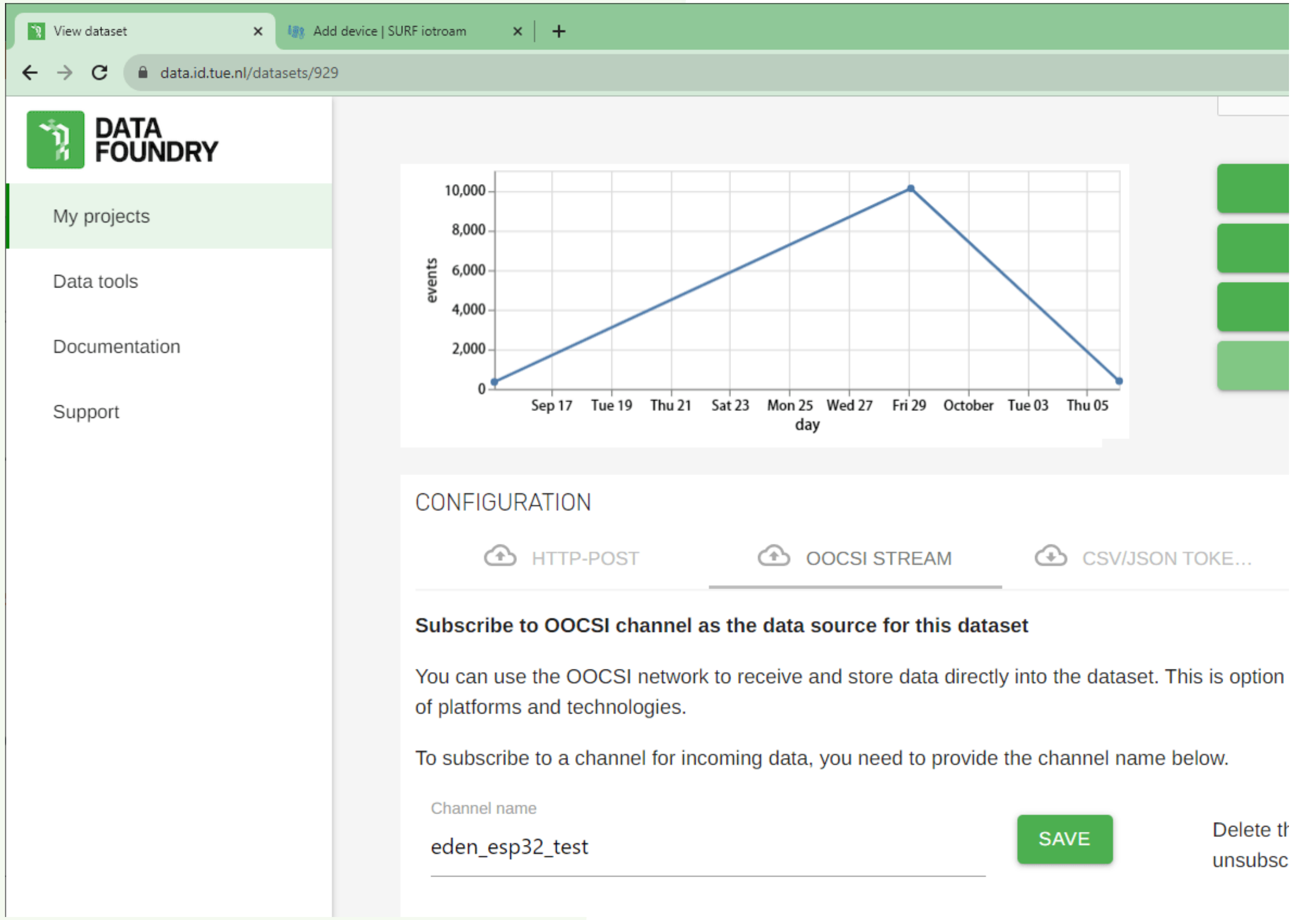
Step 5 – SEND DATA TO OWN IOT DATASET VIA OOCSI (3/4)

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.



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

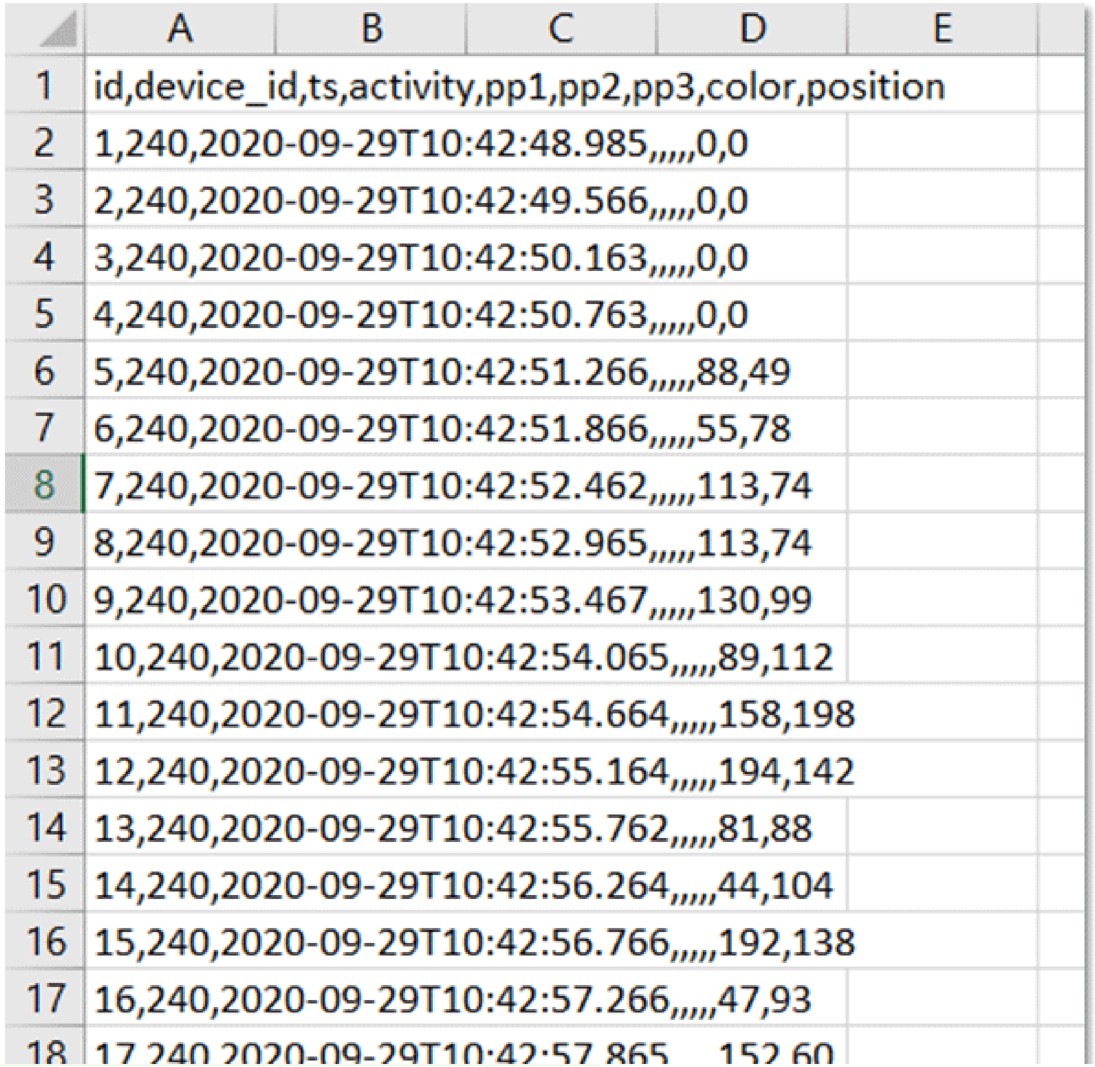
Delete tl unsubsc

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

Step 5 – SEND DATA TO OWN IOT DATASET VIA OOCSI (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				
14	13,240,2020-09-29T10:42:55.762,,,,,81,88				
15	14,240,2020-09-29T10:42:56.264,,,,,44,104				
16	15,240,2020-09-29T10:42:56.766,,,,,192,138				
17	16,240,2020-09-29T10:42:57.266,,,,,47,93				
18	17,240,2020-09-29T10:42:57.865,,,,,152,60				

Reference links (1/2)

- Data Foundry: https://data.id.tue.nl/
- OOCSI (host server): https://oocsi.id.tue.nl/
- PlayWithDataFoundry, practical exercises with Data Foundry: https://github.com/edenchiang/PlayWithDataFoundry
  or on Data Foundry:
  https://data.id.tue.nl/documentation/practice
- IDWiki (how to get MAC address and register to iotroam): https://idwiki.tue.nl/wiki/Connectivity
- Iotroam (available IoT network in TU/e): https://iotroam.org/

Reference links (2/2)

- Arduino IDE 1.8: https://www.arduino.cc/en/software#legacy-ide-18x
- ESPs installation:
  - To Arduino IDE: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/
  - USB Drivers:
    - CP210x chip: (we use this in the workshop!) https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads
    - CH340 chip: https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers
- Check port of Windows and Mac machines: reference here

Real things