



Lecture and tutorial: “IoT modules and its tutorial by using ~~SBC~~ MCU”

Takayuki Fujita
Professor, Deputy Director



AMERI
University of Hyogo
Himeji, JAPAN



Contents

- **Introduce myself**
- **Overview of IoT/MEMS**
- **Instruction for IoT tutorial**

University of Hyogo

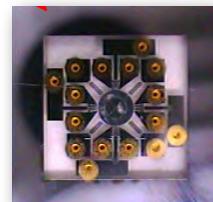
The screenshot shows the University of Hyogo website. At the top, there is a logo and the text "公立大学法人 兵庫県立大学 UNIVERSITY OF HYOGO". Below the logo, there are links for "ホーム" (Home), "English", "アクセスマップ" (Access Map), and "お問い合わせ" (Contact). There are also buttons for "文字サイズ 標準 大" (Text Size Standard Large) and "標準 小" (Standard Small). A navigation bar below has links for "受験生の方へ" (For Admissions), "在学生・教職員の方へ" (For Students and Staff), "卒業生の方へ" (For Graduates), and "社会人へ" (For Professionals). The main content area features a large image of a modern building with the text "Graduate School of Engineering" overlaid. Below the image, there is a banner with the text "工学部 school of engineering" and a link "工学部公式サイトはこちら". The banner also features a photograph of the university's building.



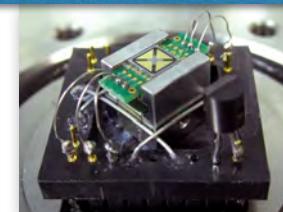
Kobe Beef

My research history

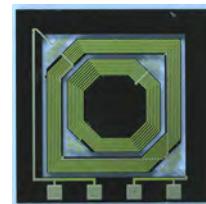
- MEMS physical sensors
 - Gyroscope
 - Accelerometers
- MEMS actuators
 - Mirror device
 - PID control system
- Environmental sensor
 - Multi sensor module for health monitoring



2D gyroscope



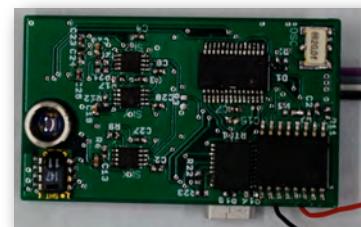
Vibratory beam accelerometer



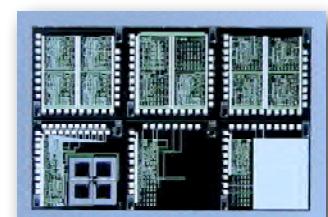
Electromagnetic MEMS mirror



Mirror control system



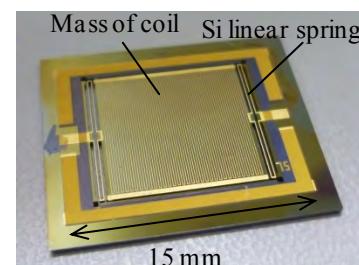
Multi-sensor module



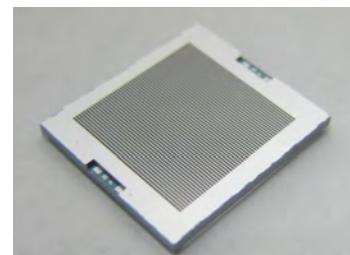
Environmental sensor

Recently, working on

- **Vibration MEMS energy harvester**
- **IoT systems for Medical applications**



Vibration energy harvester



Clean room facilities for MEMS



CleanRoom for MEMS

- Operated & managed by
 - 3 faculties
 - 1 technician
 - ~20 students

- **Full MEMS fabrication process on site**
 - Photomask fabrication, photolithography
 - Deep-RIE, dry etching, wet etching
 - Metallization, Magnetic/PZT film sputtering
 - Electroplating, Polymer-flexible electronics

AMERI, U-Hyogo

**Move to New institute
open Apr. 2022**

**Advanced Medical Engineering
Research Institute,
University of Hyogo**



In the 2nd largest prefectoral hospital in Hyogo



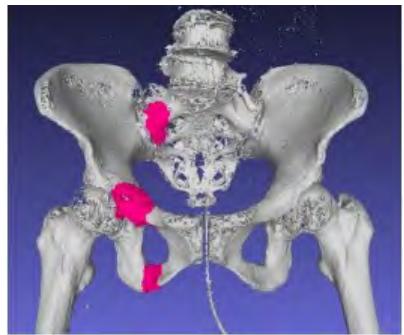
AMERI

Joint research facility

Hospital



Research Topics of AMERI



Healthcare and rehabilitation Eng.

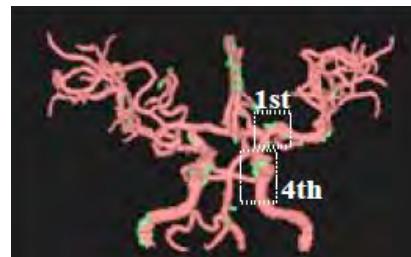
AI-based pelvic
fracture location
determination



Health observation using
adhesive plaster-type
MEMS sensors



Finger joint
estimation robotic
prosthetic hand



AI-based aneurysm
visualization

AI related topic from AMERI will be presented on Monday's Keynote talk

Contents

- Introduce myself
- **Overview of IoT/MEMS**
- Instruction for IoT tutorial

IoT (Internet of Things)



IoT

Internet connection for everything not limited to Human operated device

Human uses the Internet

Smartphone,
Tablet,
PC

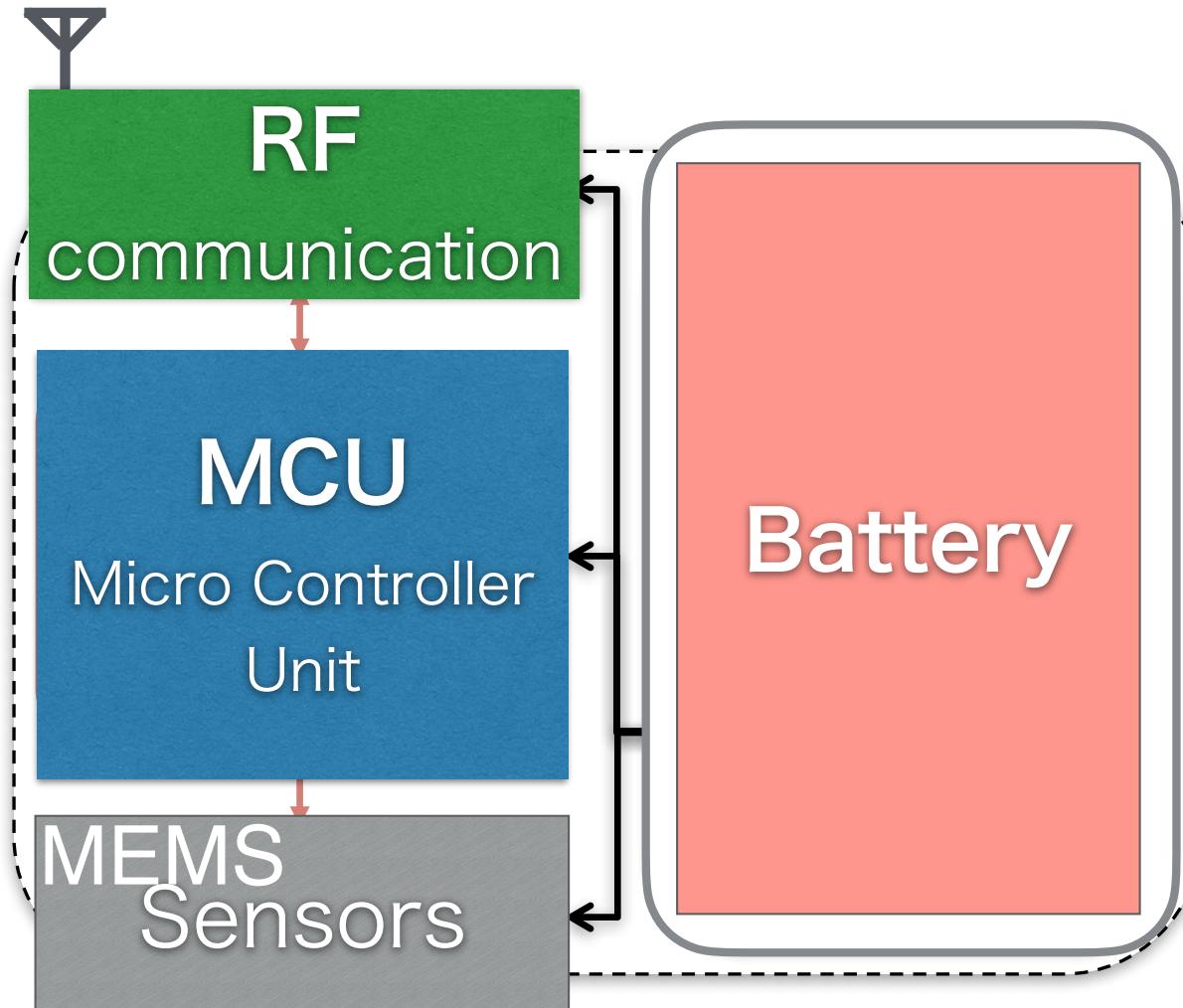


Lots of sensors connect autonomously



IoT node architecture

Typical architecture of IoT node



Architecture of a sensor node

IoT has developed dramatically thanks to **MEMS**

MEMS - Micro Electromechanical Systems

**MEMS = Fusion of ultra small machine
and Integrated Circuitry (IC)**

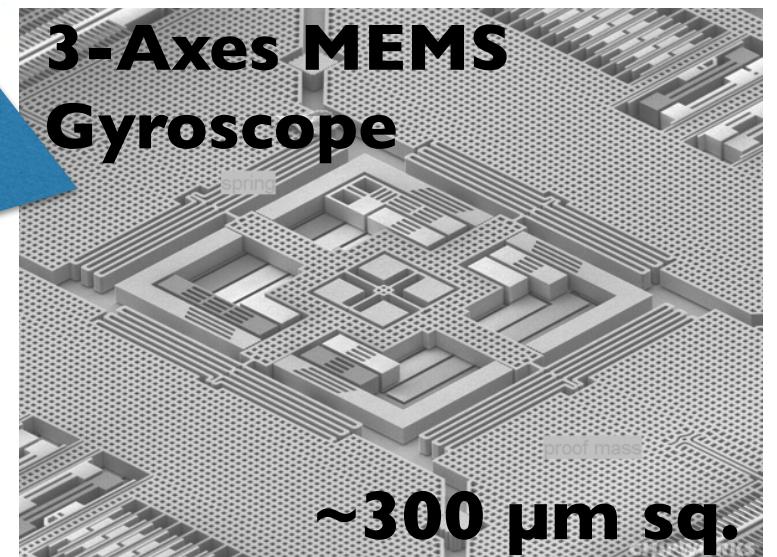
- IMU combo
- Magnetometer
- MEMS microphones
- Pressure sensor
- Humidity + Temperature sensor
- BAW filters and duplexers
- Antenna tuner



**Full of MEMS
sensors in smart
phone**

at least 5 MEMS sensors

Inside?



History of MEMS application



Automotive 35 yrs ago



- Airbag control
- Electronic stability control
- Rollover detection
- Navigation (GPS support)

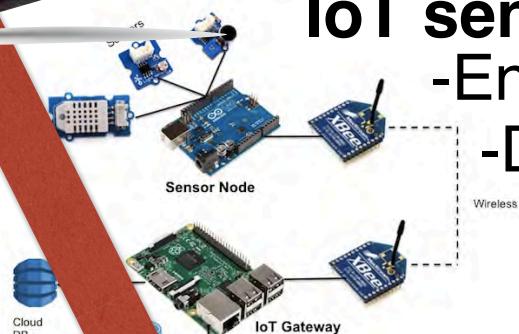


Mobilephone / Entertainment



- Input device
- Navigation

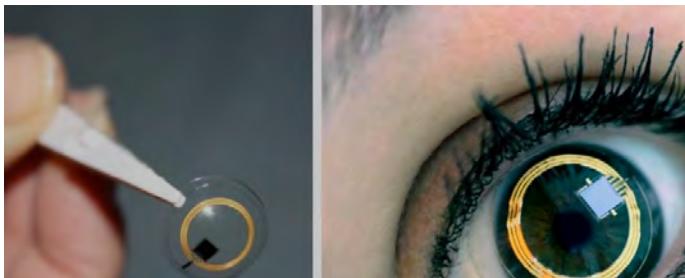
IoT sensor node



- Environmental sensing
- Drones

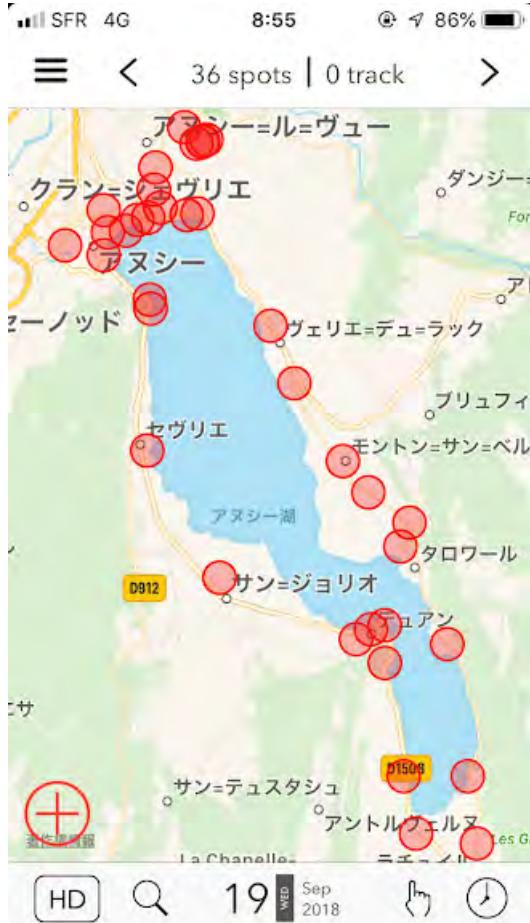
Health

- Medical devices



Performance
Size & Cost
↓ Number ↑

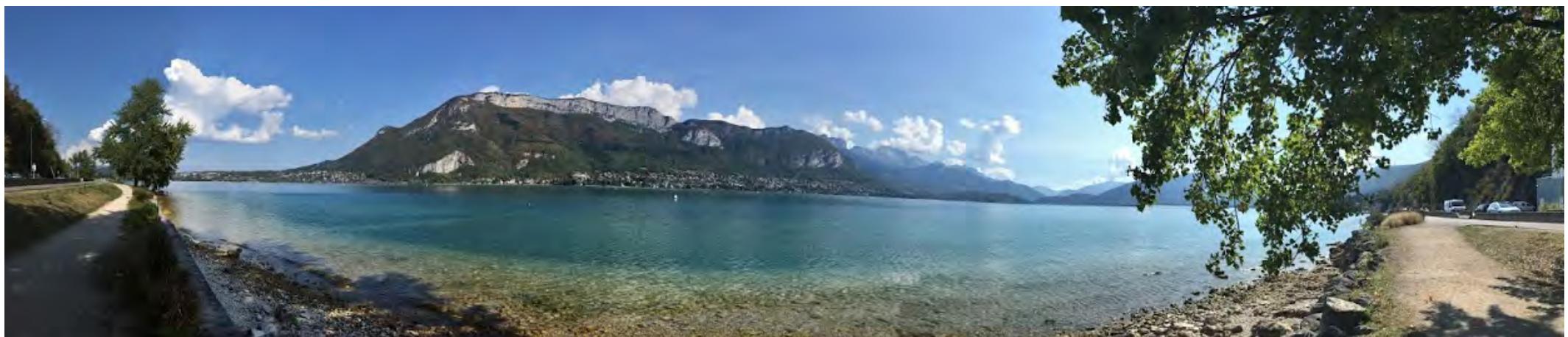
MEMS application > Navigation



Tracking: Cycling around the Lac d'Annecy

Position tracking by combined sensor info.

- GPS 
- Cell
- WiFi
- Geomagnetic
- Gyroscope
 - Angular velocity ∫ Angle
- Accelerometer
 - Acceleration ∫ Velocity ∫ Position



AirTag (not GPS but BLE/UWB)



Thanks to the AirTag(IoT)
The package finally arrived.

15th PM



16th AM



GVA Airport

KIX Osaka Airport

16th PM



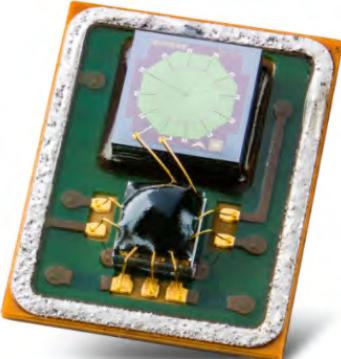
Divonne-les-Bains

16th Evening



Annecy

MEMS device 1: Microphone



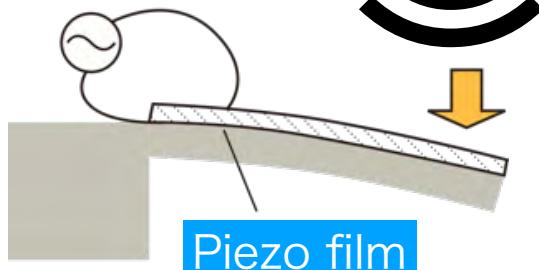
Rice grain size
 4 x 3 mm
© Vesper



Rice grain size

Cantilever with piezoelectric material vibrate by sound

Output voltage



Sound wave

Piezo film

MEMS makes compact, high-performance microphones inexpensive

iPhone 11



© Voista Media

iPhone 11 has 4 microphones
3 4 for main mic.
Others are use for noise cancelation

MEMS device 2: Apple watch etc.

Activity tracking for health care



Various devices of “healthy gadgets”



Similar technology

RF, MCU, battery
&

MEMS sensors
Acc. Gyro,

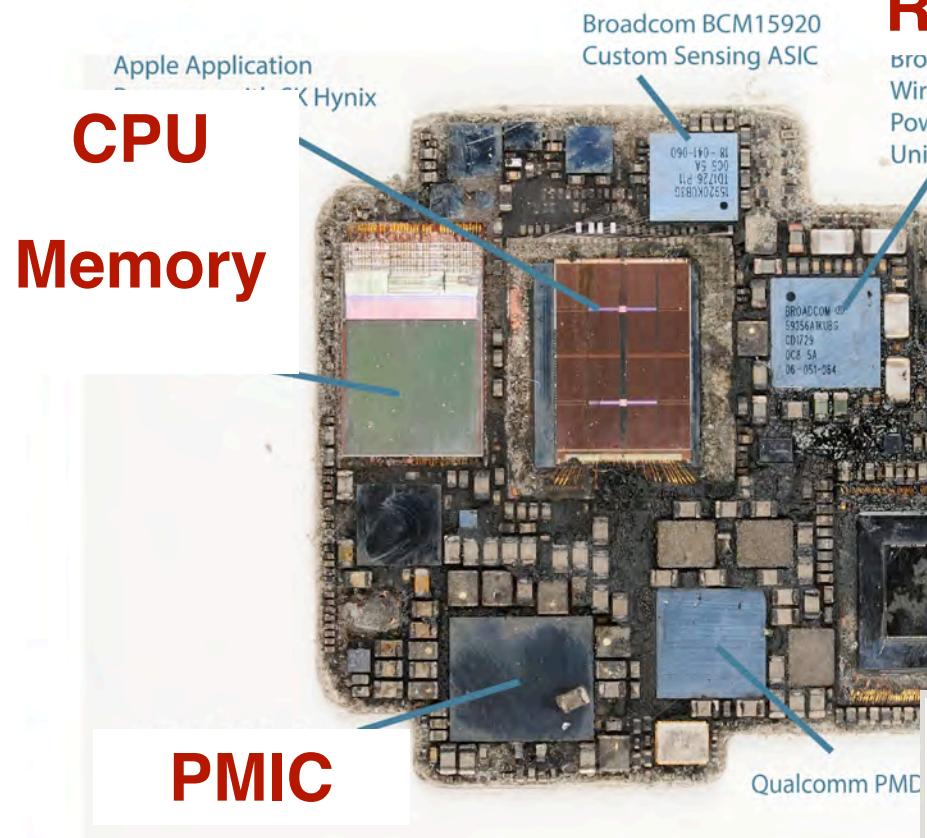
Optical sensor
For SPO₂, heart rate

- Heart rate sensor
- Accelerometer
- Gyroscope
- Bluetooth 4.0 LE



Inside of AppleWatch

(C) TechInsights



Top side

RF/wireless

Broadcom BCM159350
Wireless Charging
Power Management
Unit

Apple 338S00348V
chip (likely)

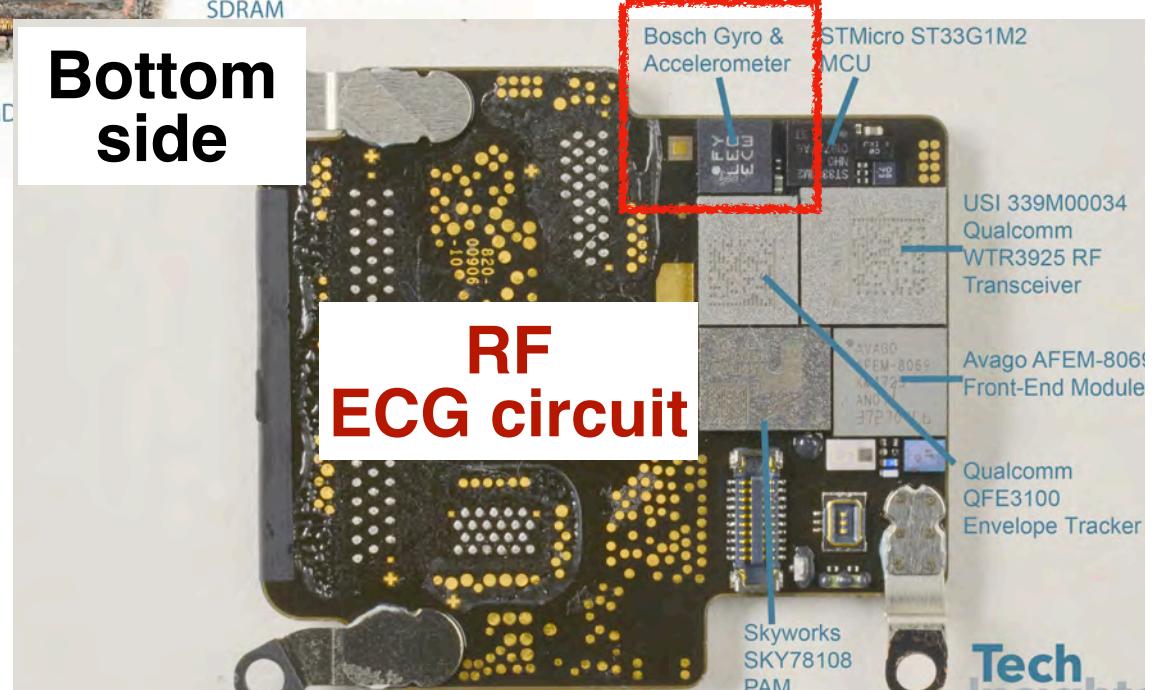
Qualcomm MDM9635M
LTE Modem with
Samsung K4P1G324EH
SDRAM

AppleWatch 3rd gen.



MEMS
Gyroscope +
Accelerometer

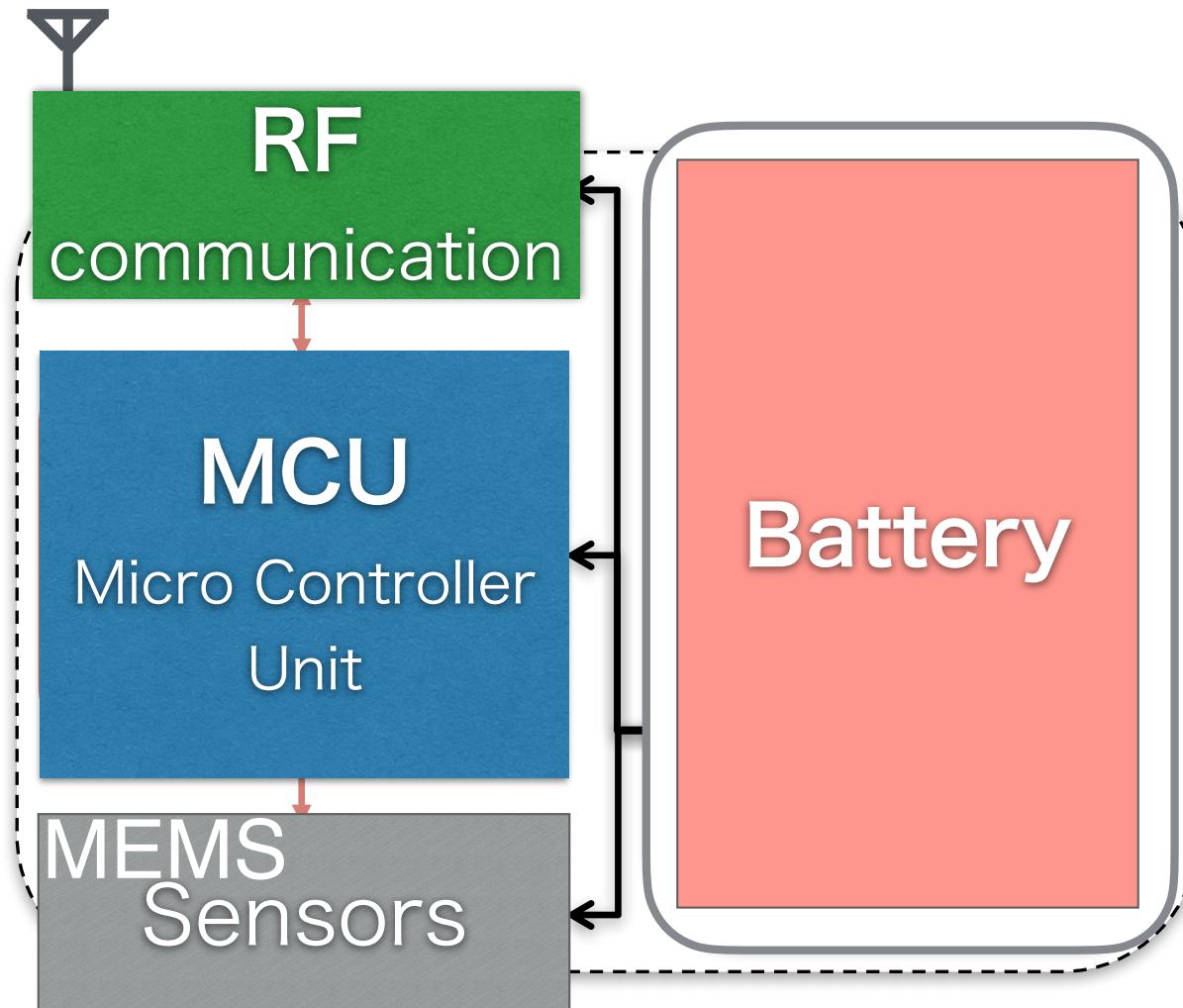
Bottom
side



Tech

IoT node architecture

Typical architecture of IoT node



Architecture of a sensor node

>NEXT

Explains the internal details of IoT applications in the market

IoT on market : IoT Body scale

¥20,000
(USD 150)
worth the price

Withings

Made in France

Smart Body Analyzer

US\$ 149.95

- WiFi connection
- No switch, Easy to use
- Automatically data upload

Pinpoint your weight and body composition

Get high-accuracy weight and body fat measurement as well as your Body Mass Index. Pick the right body type to optimize the fat mass measurement – if you exercise vigorously on a regular basis, switch to "Athlete mode".



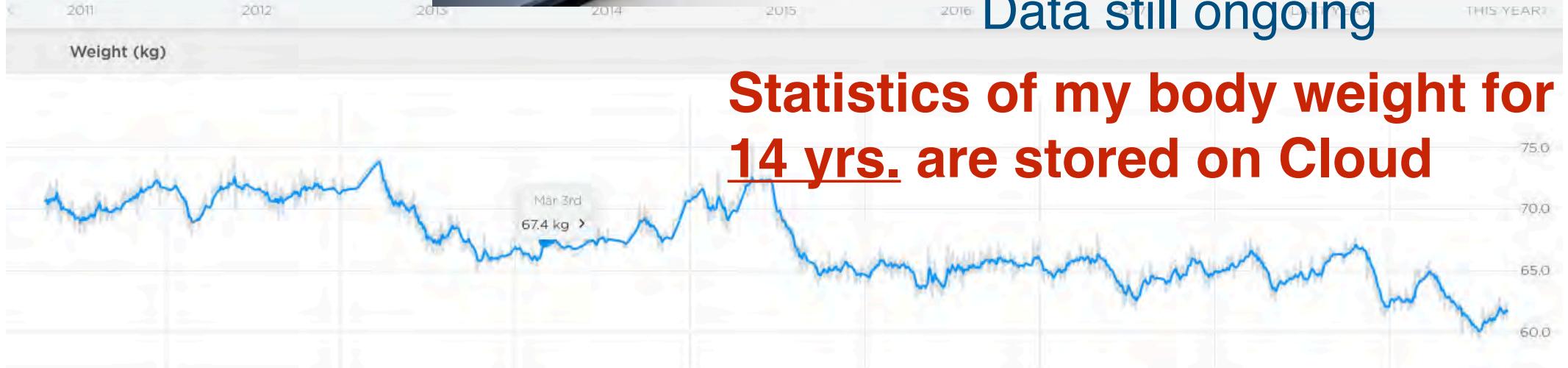
Replaced new one

Data still ongoing

Weight

2011 2012 2013 2014 2015 2016 THIS YEAR

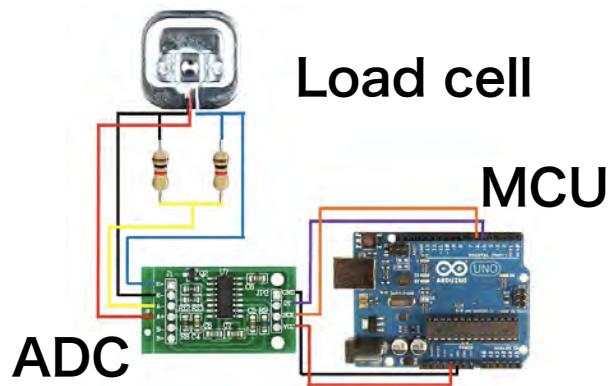
Weight (kg)



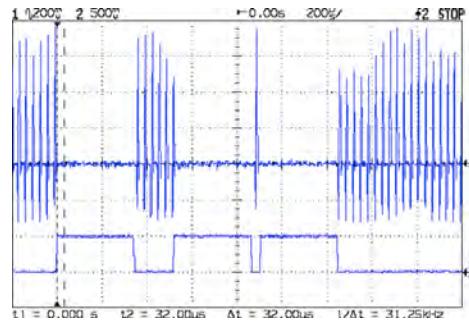
Inside of Withings IoT scale



**Body weight measured
by load-cell (strain sensor)**



Body fat % by impedance measurement



Withings server

wi+things

my account

Don't have an account yet?

mail address

メールアドレス

password

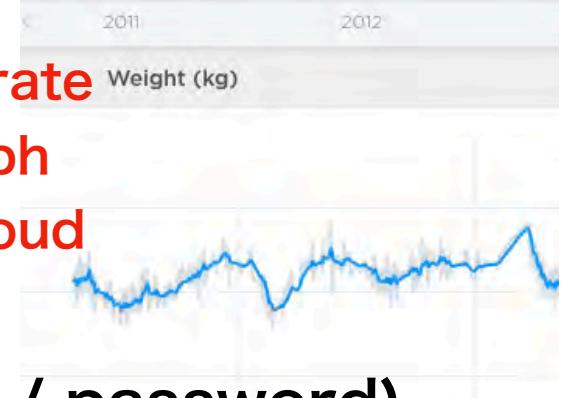
パスワード

I forgot the password?

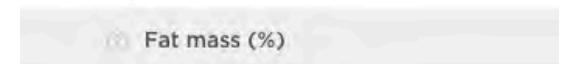


← Weight

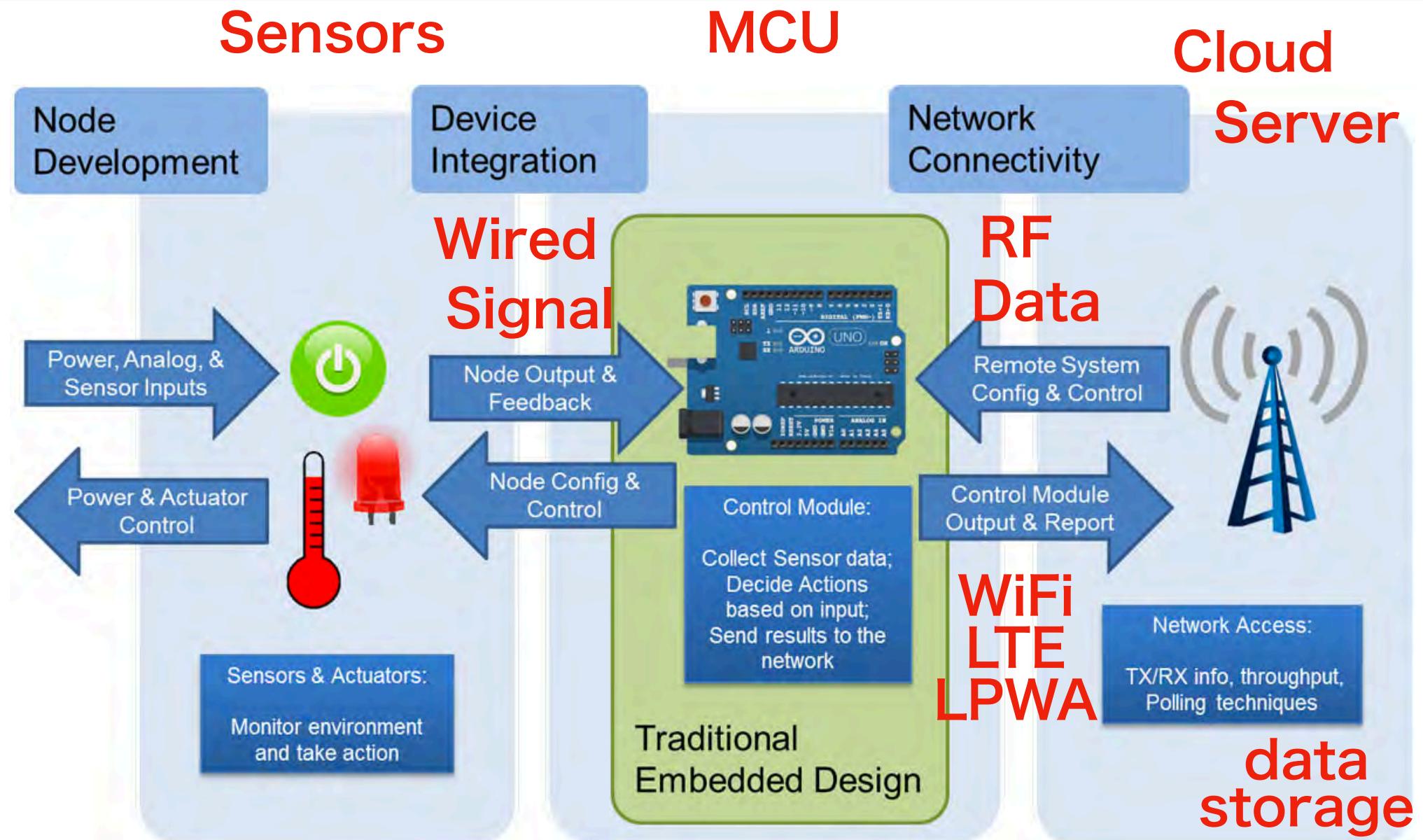
**Generate
graph
on cloud**



**Connect via Wifi (pre-configured SSID / password)
> upload data to Withings cloud server**



Typical configuration IoT



>NEXT

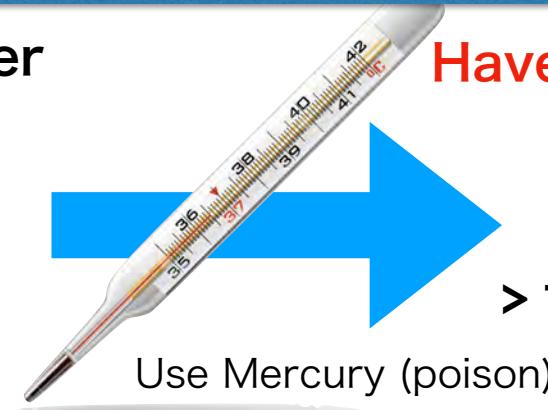
Introduction to the history of measurement,
using temperature as an example

Temperature measurement (primitive method)

ex. Glass tube thermometer

Physical change

> Temperature



Have you ever seen it?

Physical change

> thermal expansion of alcohol



No electricity

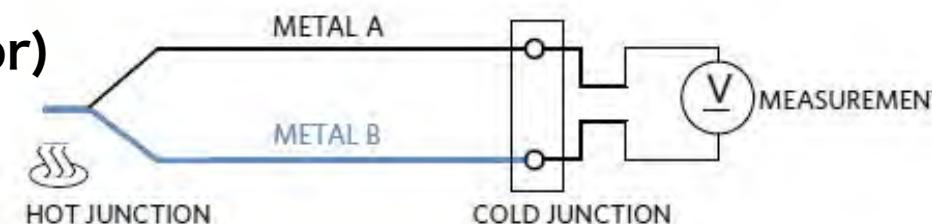
Visual check only

ex. Thermocouple
(temperature sensor)

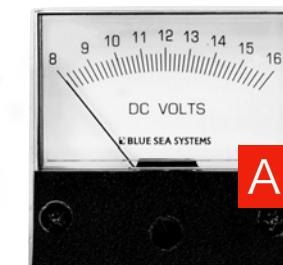


Physical change

> Temperature



Analog meter



Electrical change
> Electromotive force

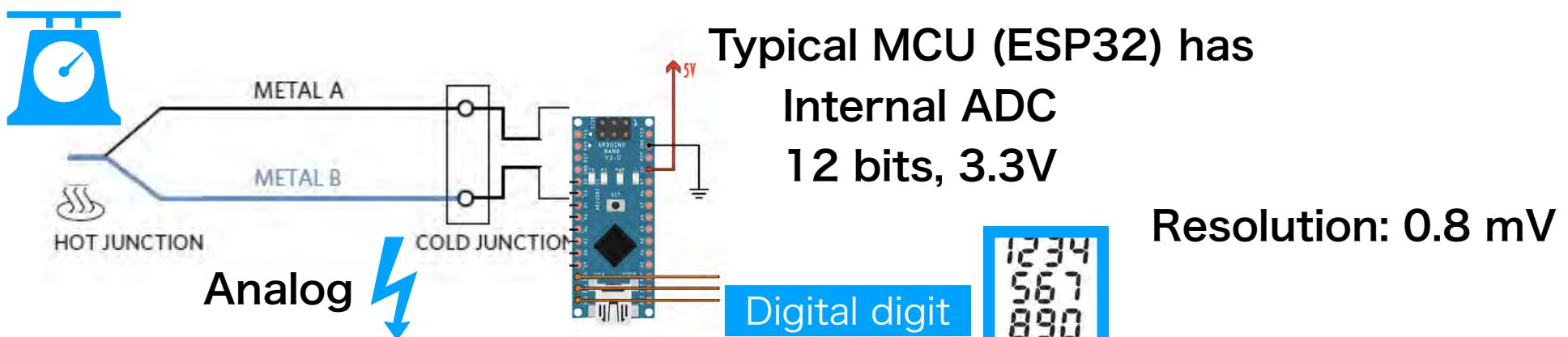
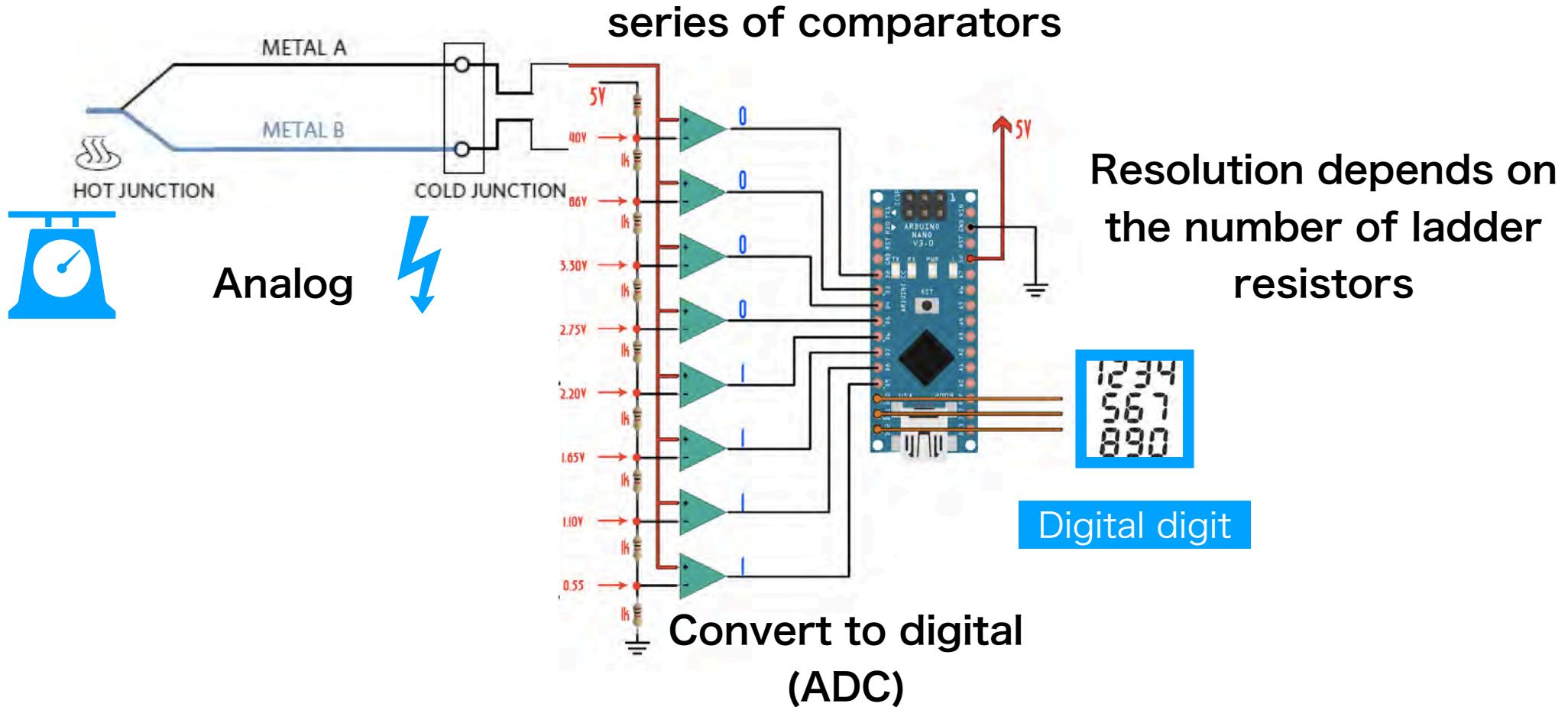


Physical change
> meter movement



Analog values are difficult to connect MCU or SBC

Temperature measurement (w/ ADC)



Recent digital output MEMS sensor



BOSCH

Invented for life

BME280
combo sensor



Use this on tutorial

Fully digital output
with compensation
parameters

Single chip
-Temperature
-Humidity
-Pressure

Main features



Relative Humidity
Measures relative humidity
with a fast response time



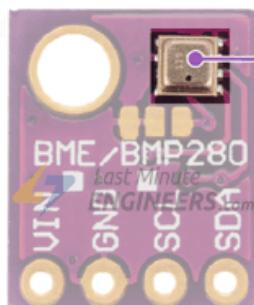
Pressure
Measures barometric
pressure and altitude



Temperature
Measures ambient
temperature



Digital



Sensor



BME280 Chip

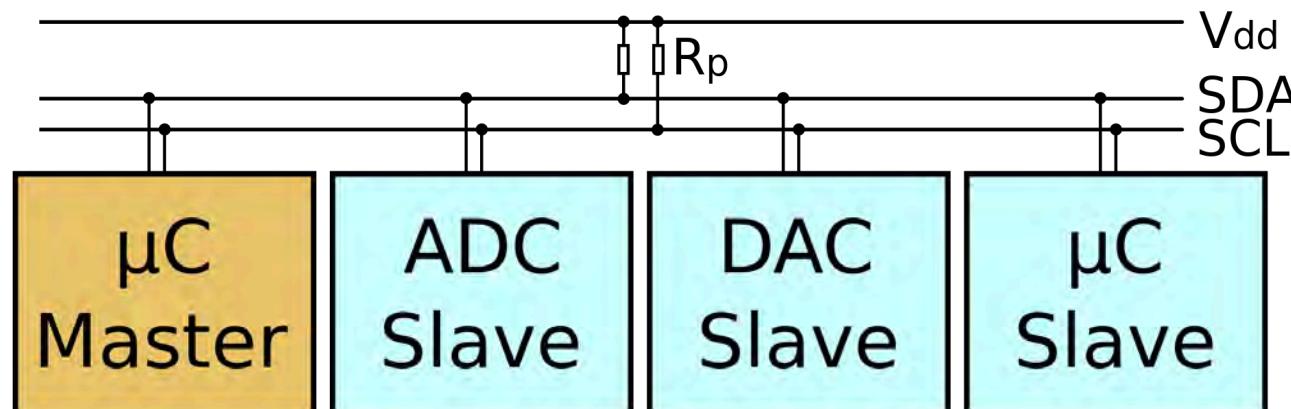
Direct connection
via I₂C/SPI protocol

What is interface
protocol?

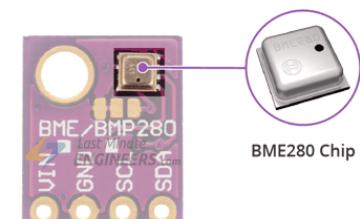
1) Inter-Integrated Circuit (I²C)



I²C (Inter-Integrated Circuit), pronounced *I-squared-C*, is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus. It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication. Alternatively I²C is spelled **I2C** (pronounced I-two-C) or **IIC** (pronounced I-I-C).

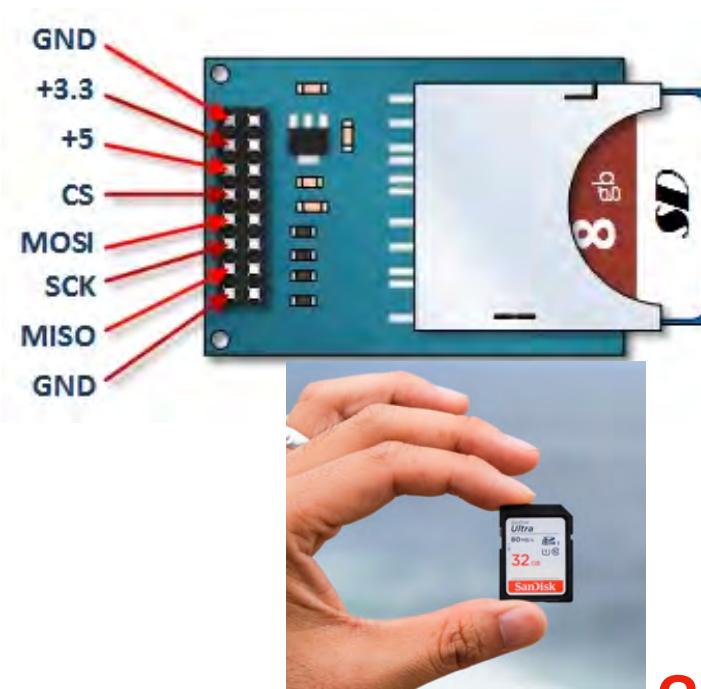


2 digital signal line + 2 power line
Selected by unique-**address**

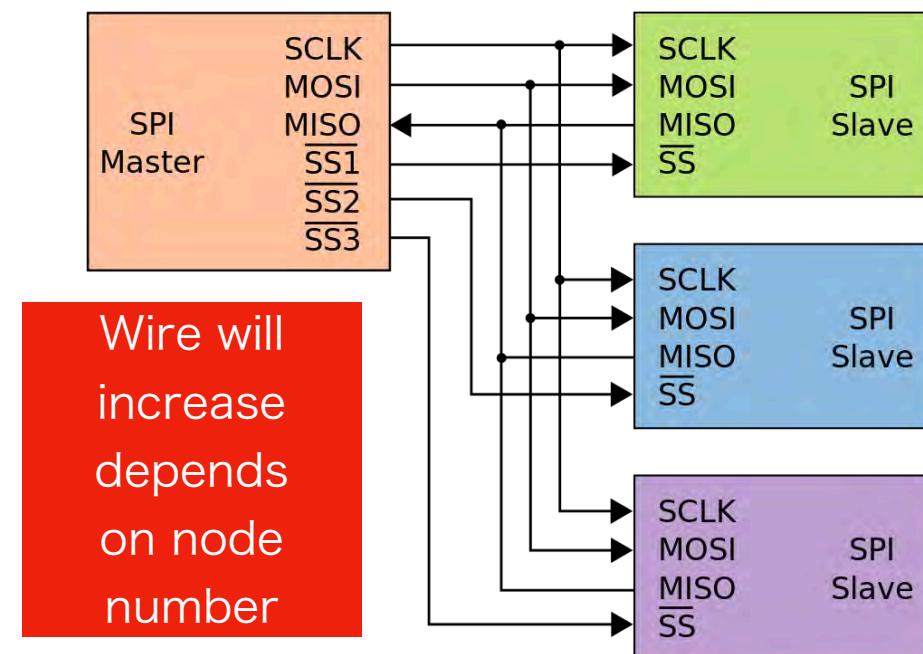


2) Serial Peripheral Interface (SPI)

The **Serial Peripheral Interface (SPI)** is a **synchronous serial communication** interface specification used for short-distance communication, primarily in **embedded systems**. Typical applications include **Secure Digital cards** and **liquid crystal displays**. SPI devices communicate in **full duplex** mode using a **master-slave** architecture with a single master. The master device originates the **frame** for reading and writing. Multiple slave-devices are supported through selection with individual **slave select (SS)**, sometimes called chip select (CS), lines.



High-speed



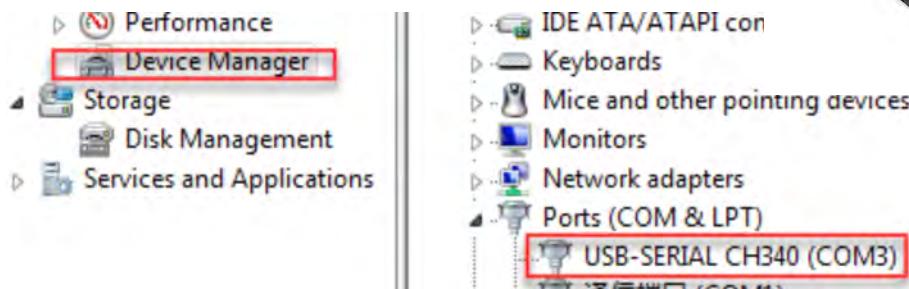
2 digital signal line + 2 power line
+1 CS line for chip select

3) Universal Asynchronous Receiver/Transmitter (UART)

A universal asynchronous receiver-transmitter (UART) is a computer hardware device for **asynchronous serial communication** in which the data format and transmission speeds are configurable. The electric signaling levels are handled by a driver circuit external to the UART. Two common signal levels are **RS-232**, a 12-volt system, and **RS-485**, a 5-volt system.



Old PC has **RS-232C**
Serial port

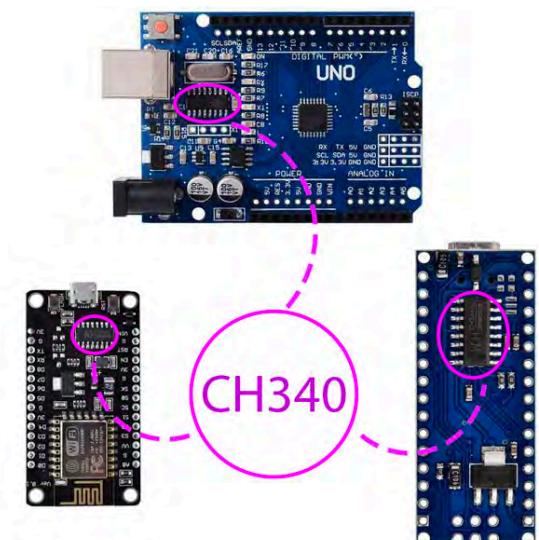


Windows recognize it as **COM**munication port

USB (Universal Serial Bus)
Replaced it

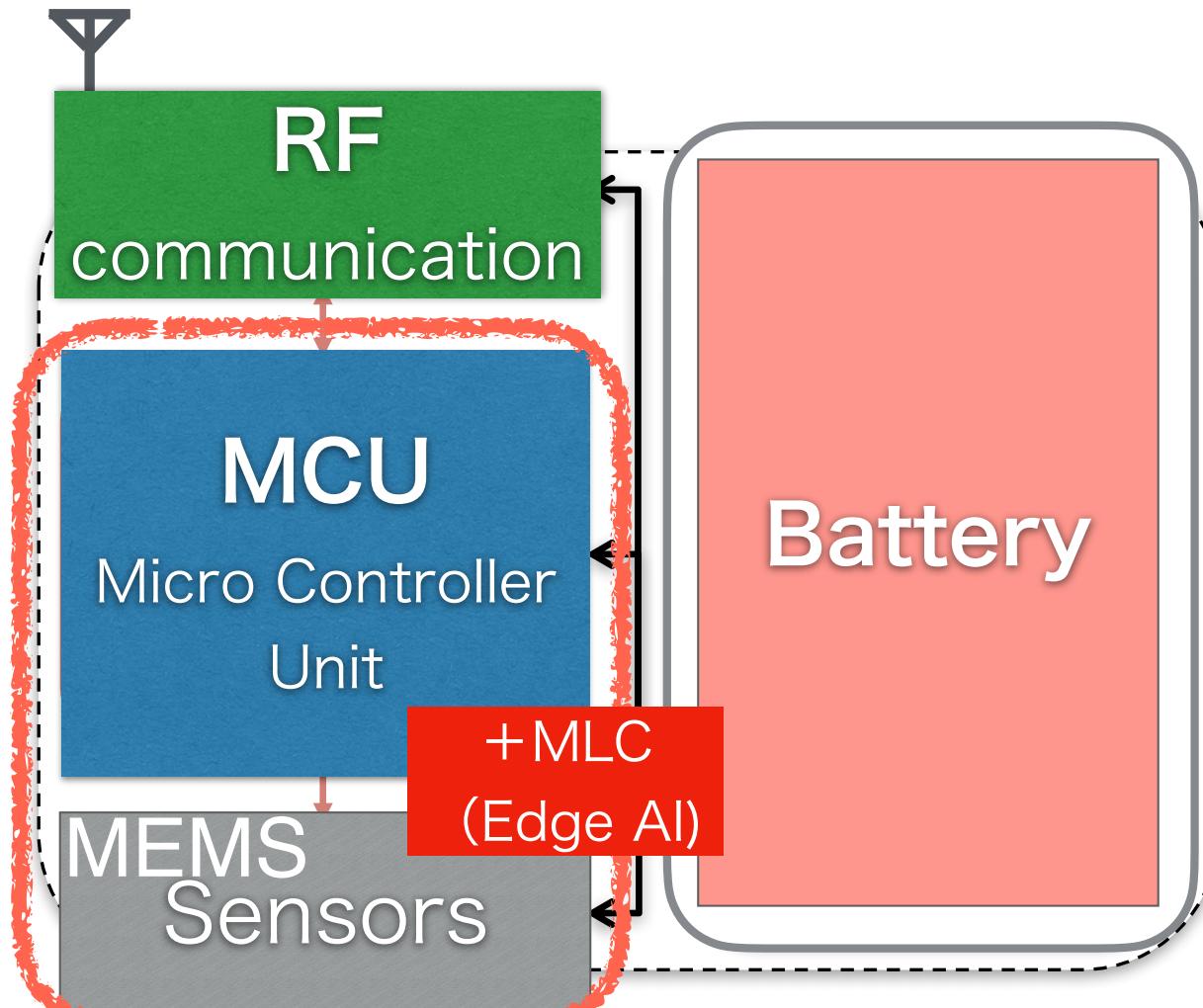


Almost all MCU board
includes USB-UART
converter chip
Ex. CH340



Cutting edge IoT node architecture

Edge AI



Architecture of a sensor node

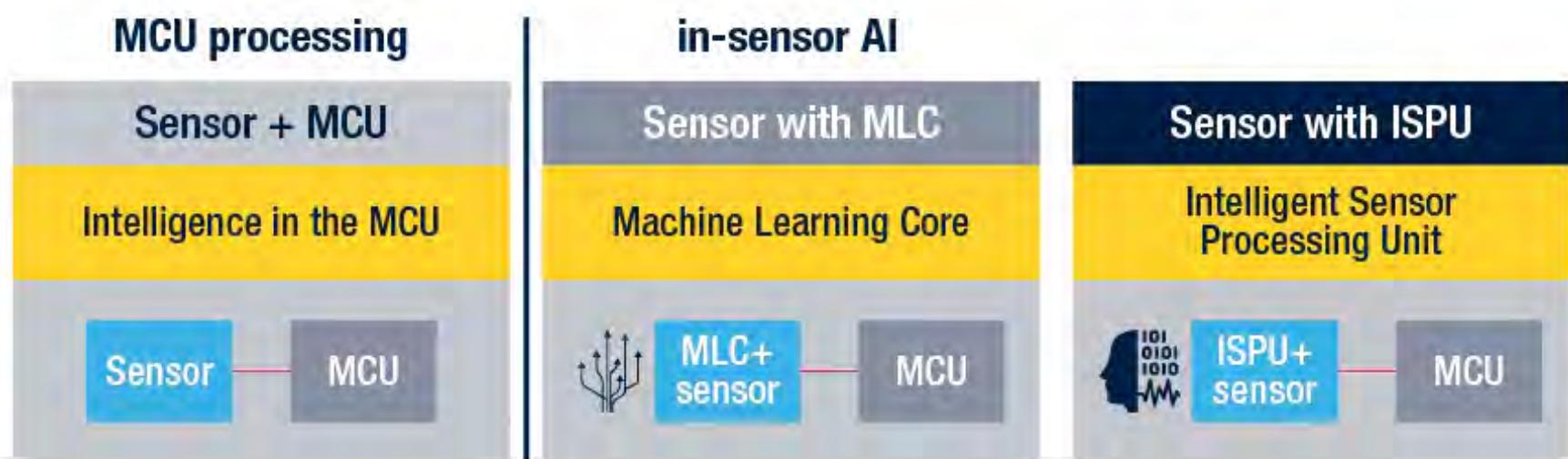
Explains the internal details of IoT applications in the market

Commercial Edge AI from STMicro



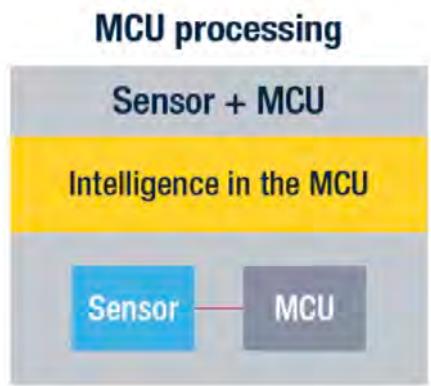
A leading company of MEMS sensor & Edge AI

To ensure developers find the most effective solution in terms of computing capacity and flexibility in programming, ST offers a choice of several technologies for in-sensor processing: sensors with an embedded **machine learning core (MLC)** and sensors with an **intelligent sensor processing unit (ISPU)**.



Edge AI on market

Panasonic e-bike & ST Micro



DETECTING LOW TIRE PRESSURE ON E-BIKES VIA EDGE AI

Problem: Electric-assist bicycles in Japan often suffer punctures caused by low tire pressure.



Implementation:
Edge AI processes data locally and triggers a warning alarm



Approach (no pressure sensor needed):

- Compare rear-wheel motor RPM with front-wheel speed sensor data
- AI detects abnormal differences → indicates low pressure in either front or rear tire



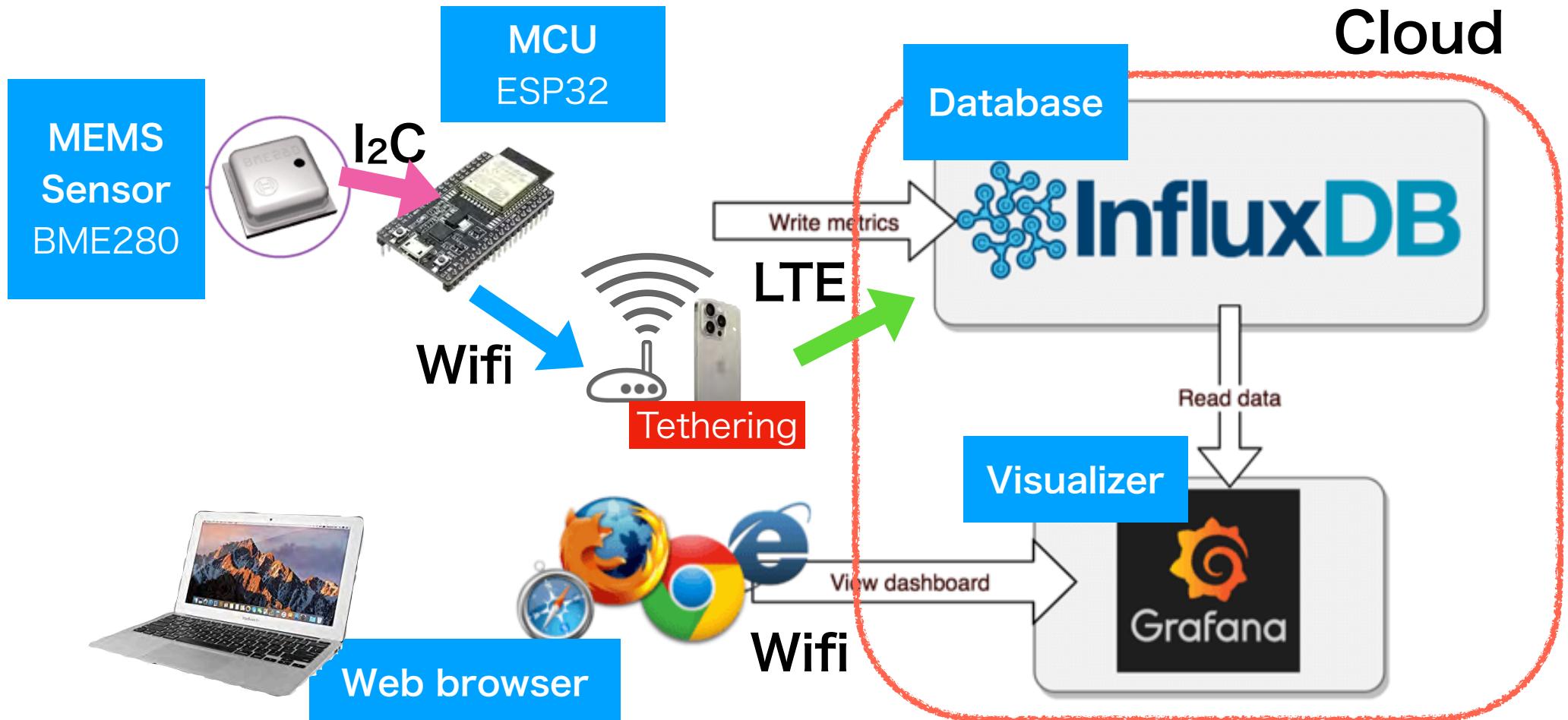
Contents

- **Introduce myself**
- **Overview of IoT/MEMS**
- **Instruction for IoT tutorial**

Goal of tutorial

Sensing data from **MEMS** environmental sensor

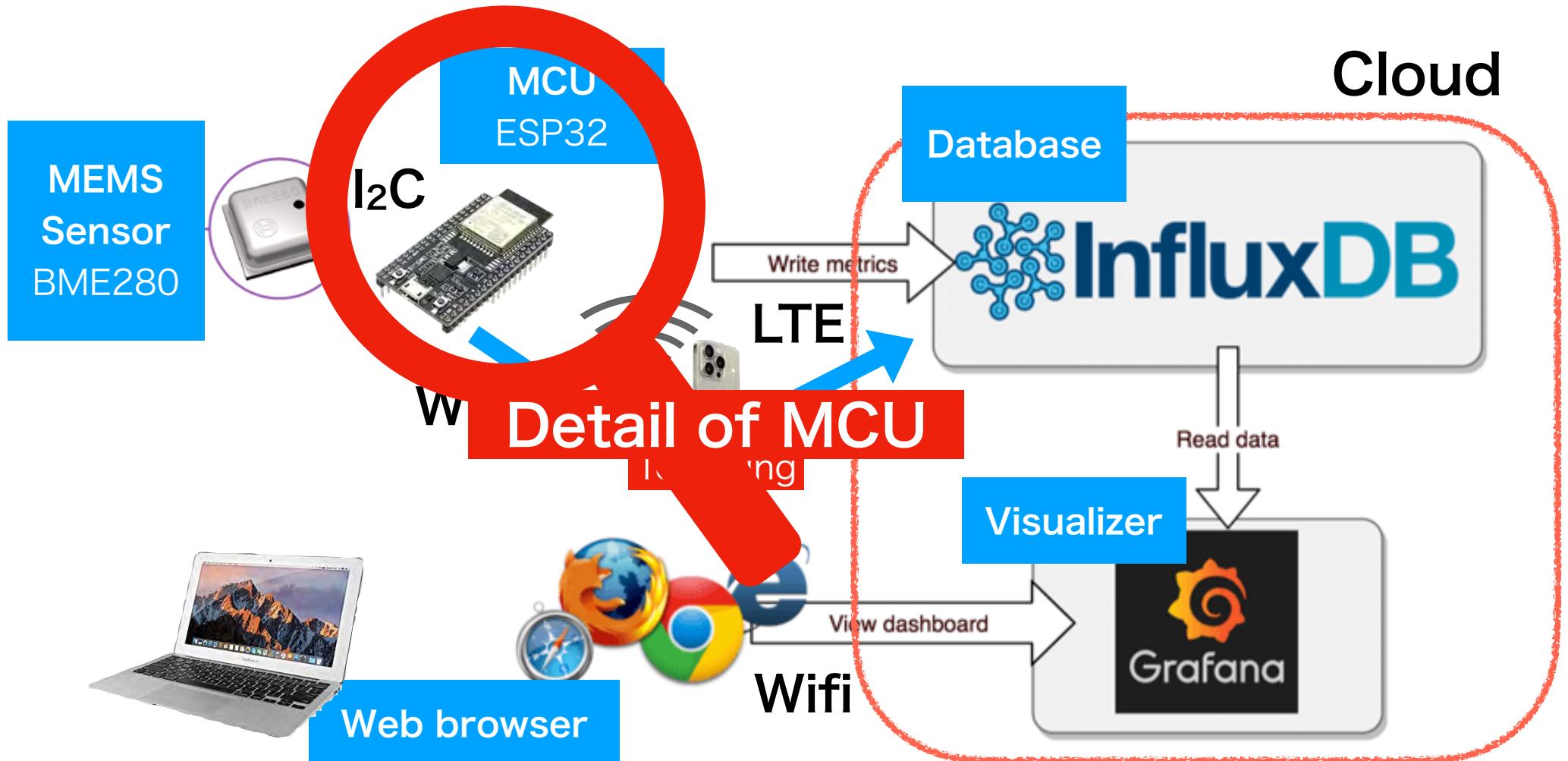
- > gather & processing data on **MCU**
- > transmit to **InfluxDB** database on cloud
- > visualized by **Grafana** > display on web browser



Goal of tutorial

Sensing data from **MEMS** environmental sensor

- > gather & processing data on **MCU**
- > transmit to **InfluxDB** database on cloud
- > visualized by **Grafana** > display on web browser



Wide variety of MCU variations

ARM-based MCU is popular these days

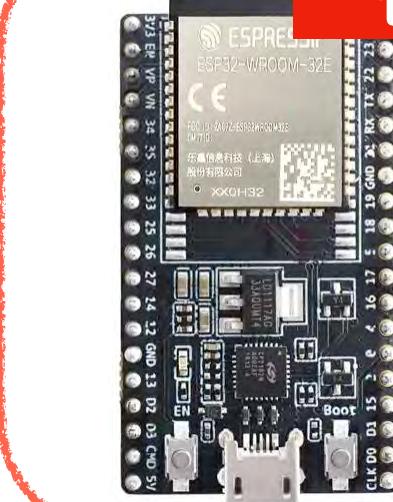
Raspberry Pi Pico



ARM Cortex M0+

ESP32

Use this on
tutorial



Apple silicon also
ARM based core



nRF52840

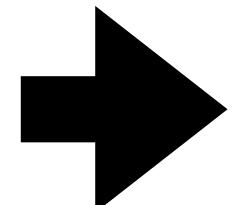


ARM Cortex M4

may be used in Challenge

Each company's original IDE
(Integrated Development Environment)

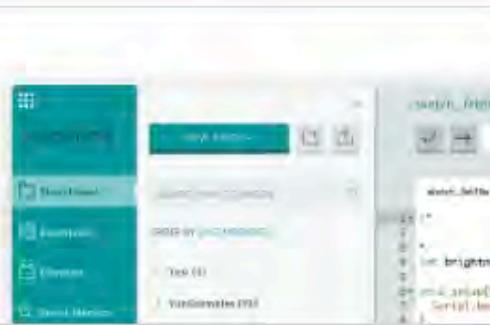
Generic development environment is
useful in terms of asset utilization





Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

[CODE ONLINE](#)[GETTING STARTED](#)

Originally Arduino Development Environment Downloads

But libraries Support Various MCUs



Arduino IDE 2.2.1

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocomplete, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through

Based on C/C++ language

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits

Windows MSI installer

Windows ZIP file

Linux AppImage 64 bits (X86-64)

Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.14. "Mojave" or newer, 64 bits

macOS Apple Silicon, 11. "Big Sur" or newer, 64 bits

[Release Notes](#)

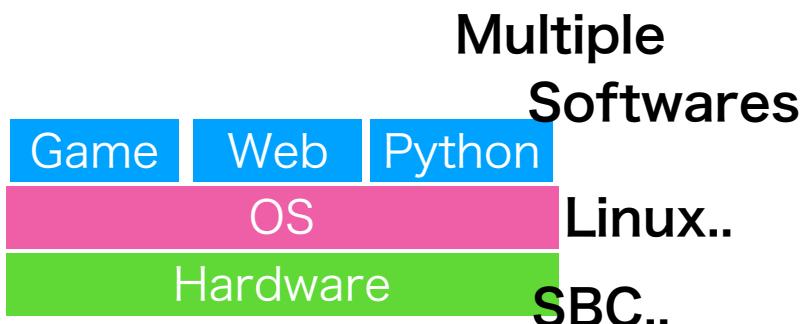
Help

Comparison of SBC and MCU

SBC

Single Board Computer
= Like a small PC

Raspberry Pi5



Very flexible operation
require shutdown procedure

MCU

Micro Controller Unit
= CPU + peripheral I/F

ESP32



No OS
Single software = Firmware
e.g. Temp. Cont.



Very robust operation
just power on/off

This tutorial uses Python on the MCU as an alternative solution

The screenshot shows the MicroPython website. At the top is a dark header with the MicroPython logo (three vertical bars) and the word "MicroPython". To the right are links for "DOWNLOAD", "DOCS", "DISCORD", "DISCUSSIONS", "WIKI", and "STORE". Below the header, the word "MicroPython" is written in large, bold, black letters. To the left of the main content, there's a sidebar with some text and a red box containing the text "Python is an realtime interpreter language". The main content area has a large image of a microcontroller chip resting on a stylized representation of a computer keyboard. Overlaid on the image are three colored boxes: blue (.py script), pink (MicroPython), and green (Hardware). A small image of a microcontroller board is also visible.

MicroPython is a lean and efficient implementation of the Python 3 programming language. It includes a full Python interpreter and a rich set of libraries of modules written in C that allow you to control all kinds of hardware. The MicroPython firmware that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects.

MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256k of code space and 16k of RAM.

MicroPython aims to be as compatible with normal Python as possible to allow you to transfer code with ease from the desktop to a microcontroller or embedded system.

Programming language for small resources compatible with Python

Run within just 256k of code space and 16k of RAM.

C++ IDE vs. μ Python for development



C++

On PC



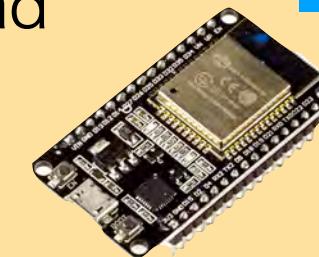
Arduino IDE

Programming & Compiling

Generate binary

Upload

On MCU



Reset, Operation check

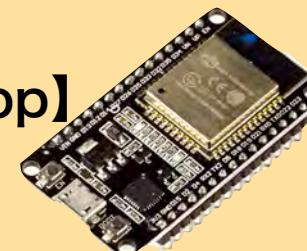
Little complicated

MicroPython



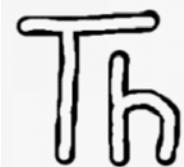
On MCU

Once μ Python farm is uploaded

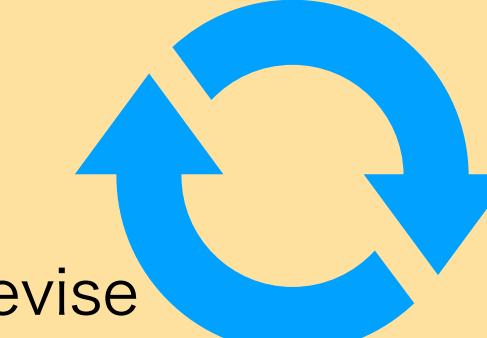


Thonny IDE

On PC



Revise



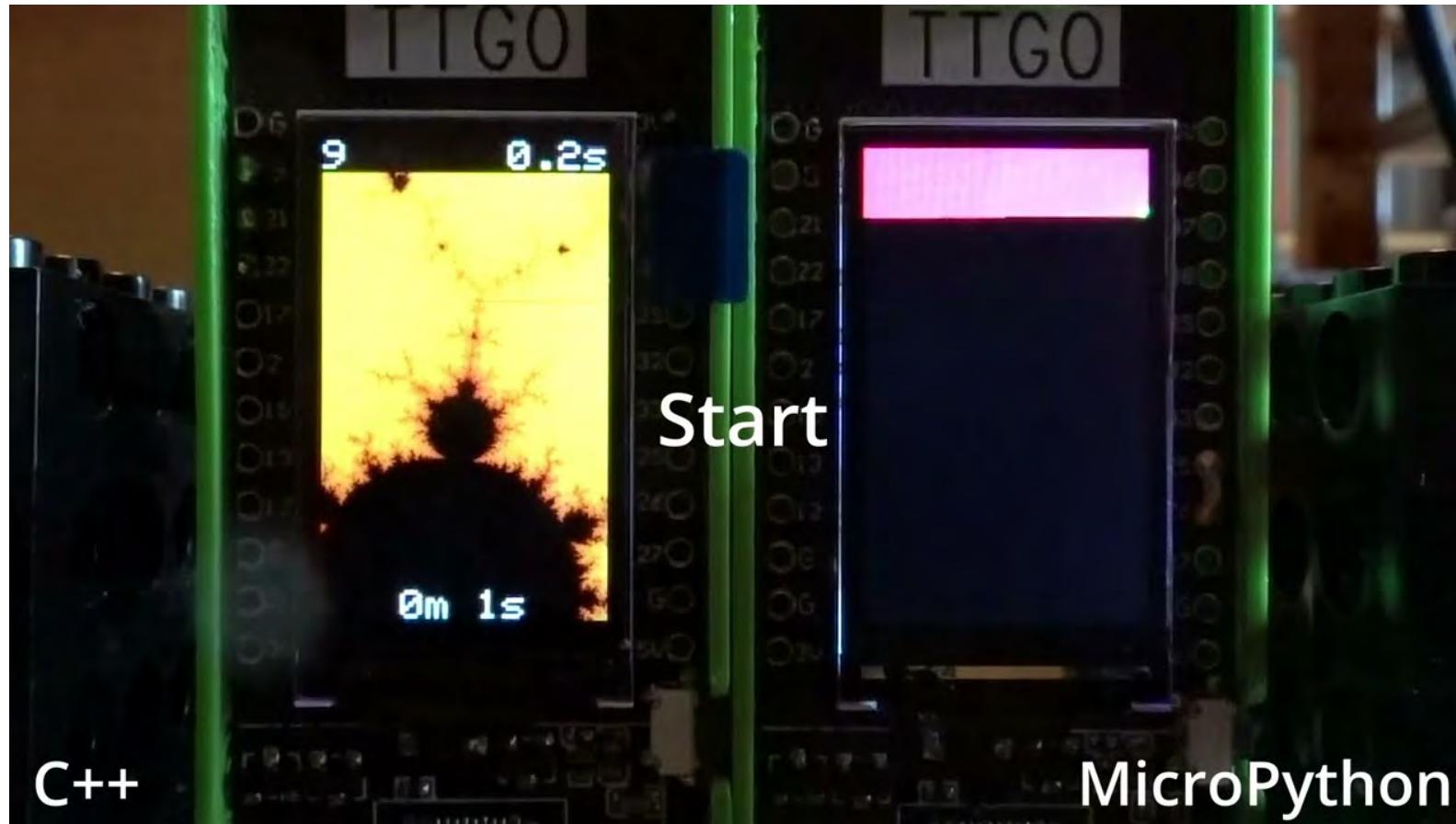
Operation check

Very easy for prototyping

C++ IDE vs. μ Python (performance)



Mandelbrot with ESP32+LED



©YouTube

C++ is 400 times faster than uPython @ ESP32

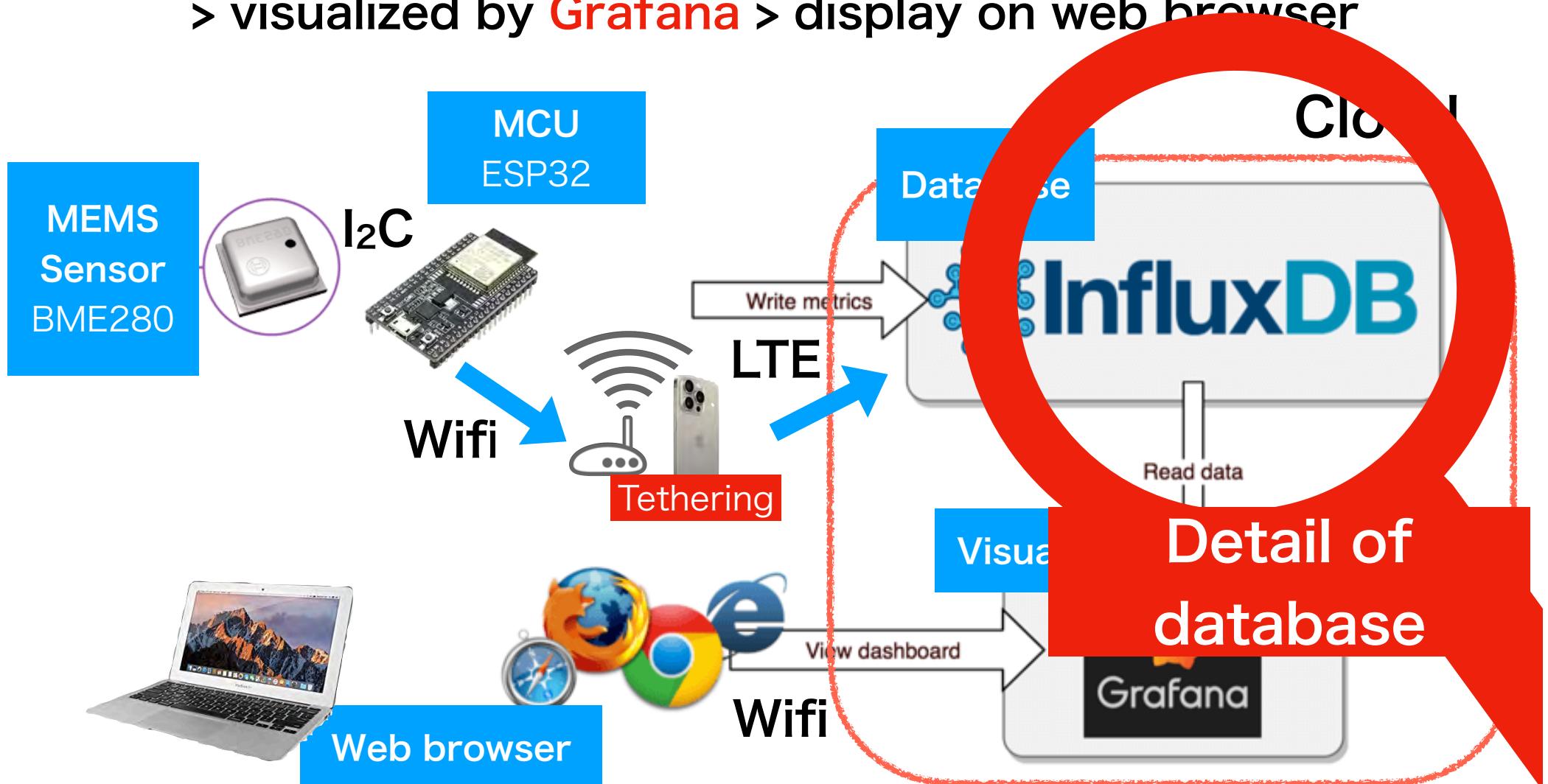
> **μPython is for prototyping**

> C++ is for production

Goal of tutorial

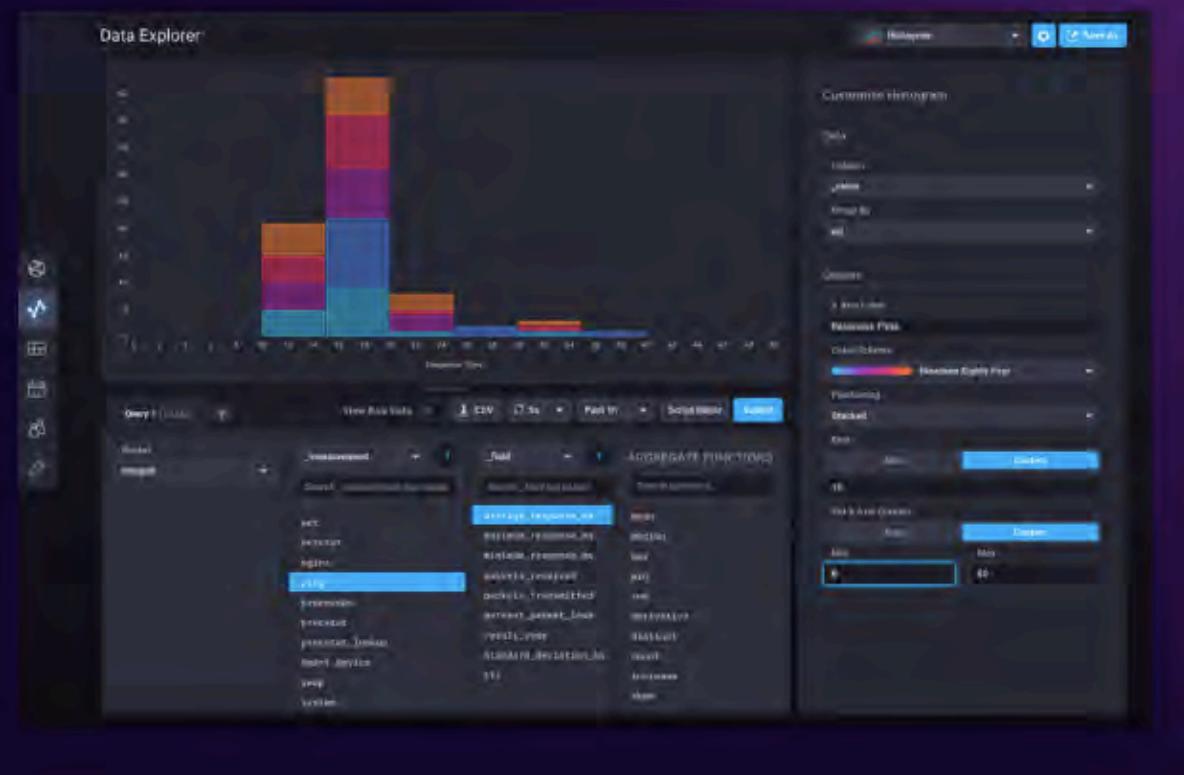
Sensing data from **MEMS** environmental sensor

- > gather & processing data on **MCU**
- > transmit to **InfluxDB** database on cloud
- > visualized by **Grafana** > display on web browser

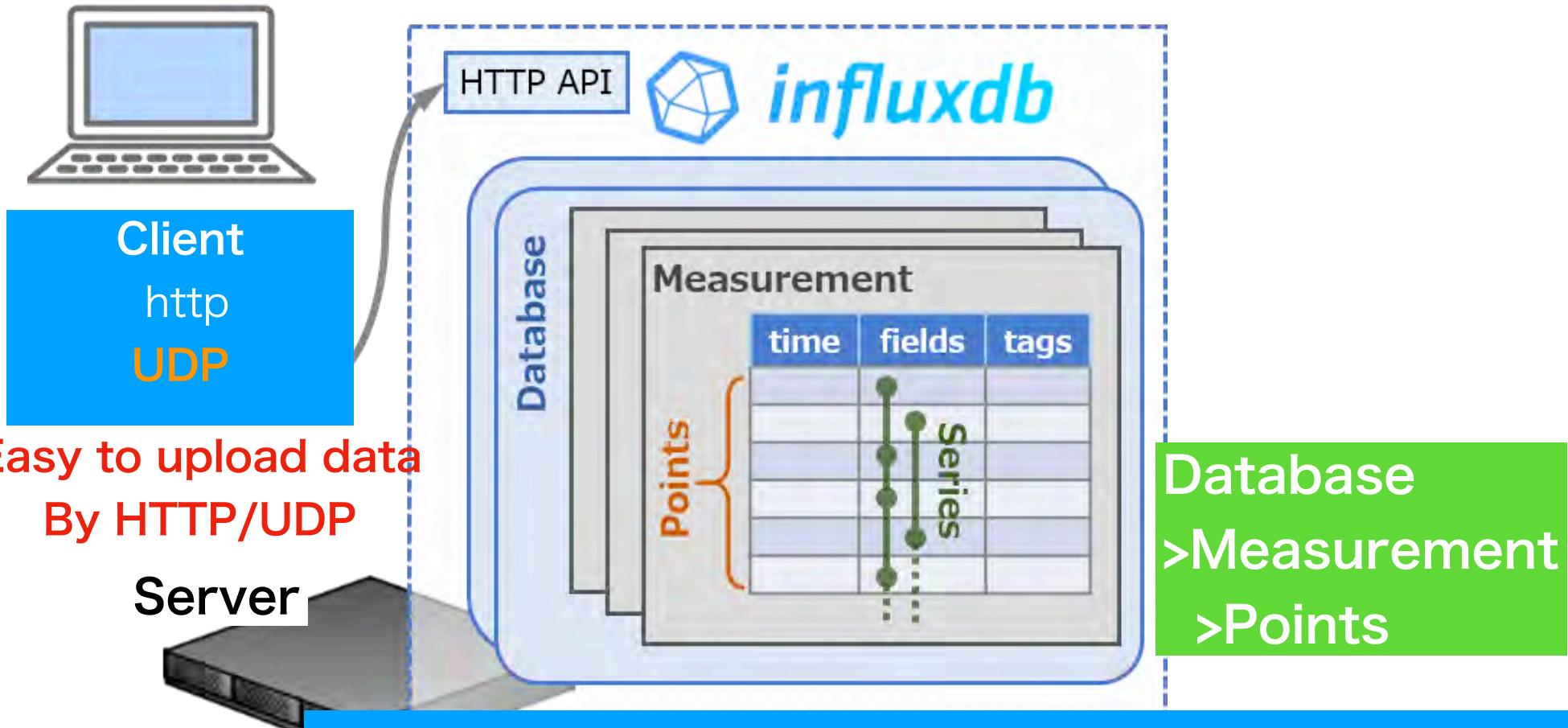


InfluxDB

InfluxDB is a time series database designed to handle high write and query loads.

[Get InfluxDB](#)

InfluxDB is a high-performance **data store** written specifically for **time series data**. It allows for high throughput ingest, compression and real-time querying. InfluxDB is written entirely in Go and compiles into a single binary with no external dependencies. It provides write and query capabilities with a command-line interface, **a built-in HTTP API**.



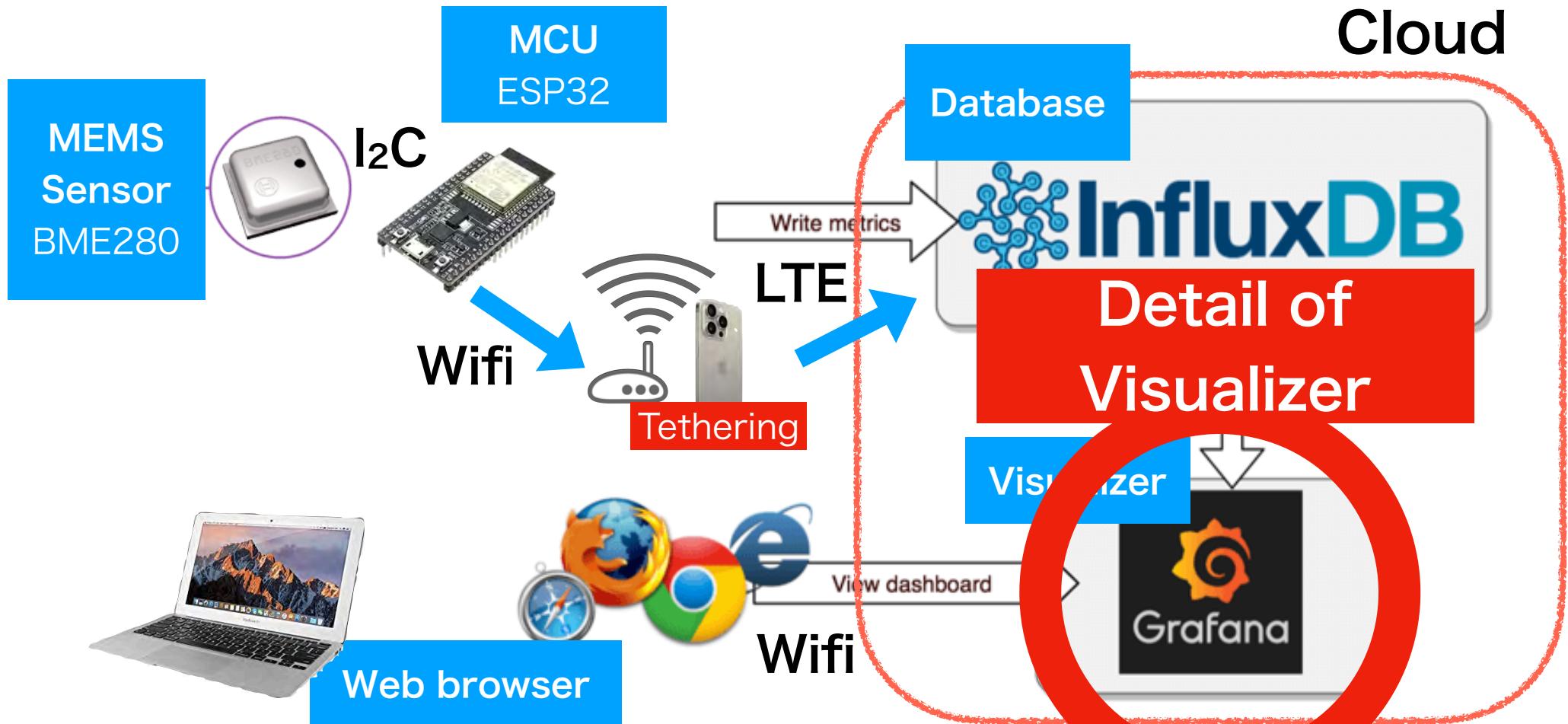
Structure of InfluxDB database

Database	Measurement		Values		
	Timestamp	Location	Temperature	Humidity	Barometer
Point	15977464390000000000	bedroom	23.4	67.8	1023.4
	15977464990000000000	bedroom	23.5	67.9	1023.6
	15977465590000000000	bedroom	23.5	68.1	1023.7

Goal of tutorial

Sensing data from **MEMS** environmental sensor

- > gather & processing data on **MCU**
- > transmit to **InfluxDB** database on cloud
- > visualized by **Grafana** > display on web browser



The leading **open source** software for time series analytics

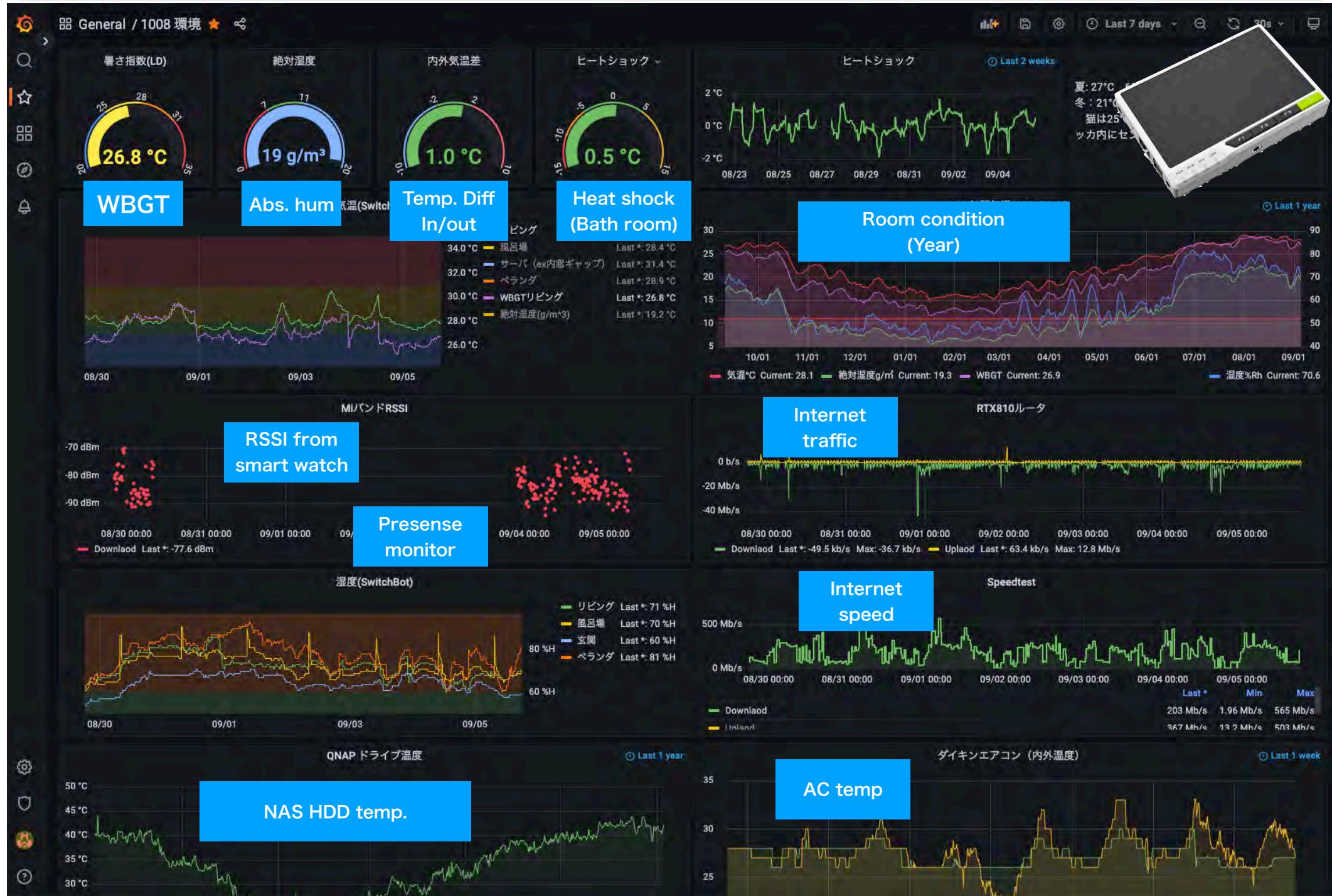


No matter where your data is, or what kind of database it lives in, you can bring it together with Grafana. Beautifully.



Grafana allows you to query, **visualize**, alert on and understand your metrics no matter where they are stored. Create, explore, and share **dashboards** with your team and foster a data driven culture.

Example of Grafana (my home env.) on RPi



Tutorial list

Instruction & sample program

<http://iotwork.org/smb/>

Preparation

0. MicroPython install on ESP32C3 by Thonny IDE

Experiments

1. Blink LED on ESP32
2. Checking the WiFi function of ESP32
3. Wire BME280 to get data
(BME280.py library installation)
4. Upload BME280 data to cloud via UDP from ESP32
~~Revised > Data send by UART to PC > UDP upload~~
5. Visualization on Grafana



(Extra) BLE Beacon RSSI monitoring on Grafana

Today

90' Lecture
10:00 - 11:30

T. Fujita, University of Hyogo, JP
IoT modules & its tutorial by using SBC
BXXX

Lunch break
11:30-13:15

This afternoon

Challenge
13:15 - 14:45
Group : C

90' Tutorial - 13:15 - 14:45
H. Sawada
Image processing for
Mechatronics
CXXX - Group : B

90' Practical work
13:15 - 14:45
T. Fujita
IoT modules tutorial using
SBC
CXXX - Group : A

A

Split to 2 small groups
ex.

Group A1 → A1a, A1b
(mixed universities)

Thursday Sept 18

enge: Start
15 - 11:45
roups : B

45' Keynote
9:00 - 9:45
H. Sawada
AI

Coffe Break

90' Practical work
10:15 - 11:45
T. Fujita
IoT modules tutorial using
SBC
CXXX - Group : C

90' Tutorial - 10:15 - 11:45
H. Sawada
Image processing for
Mechatronics
CXXX - Group : A

C

Friday Sept 19

90' Lecture
8:15 - 9:45
Y. Suzuki, University of Tokyo
Design of broadband vibration energy harvesting device
BXXX

Challenge
10:00 - 11:30
Group : C

Coffe Break

90' Practical work
10:00 - 11:30
T. Fujita
IoT modules tutorial using
SBC
CXXX - Group : B

90' Practica
10:00 - 11:
Y. Suzu
Design of bro
vibration ene
device
CXXX - Gro

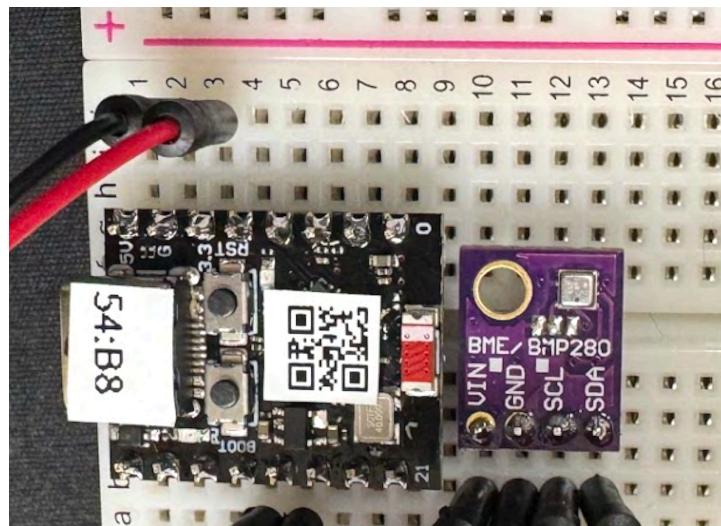
B

Before the tutorial

Split to 2 small teams

ex. Team A1 → A1a A1b
(mixed universities)

GROUP A				TEAM	GROUP B				TEAM	GROUP C				TEAM
PHILIPPE	Nicolas	MAM2	FRA	A1	NONNEROUX FATTAZ	Julien	MAM2	FRA	B1a	THIAW	Sidy	MAM2	FRA	C1a
ENATSU	Koichi	Tokyo	JPN		RAJERISON	Lorry	FISA5	FRA		SALAMEH	Ahmad	MAM2	FRA	
TAKAGI	Yukihiro	Waseda	JPN	A1	AMANO	Sota	Waseda	JPN	B1	OKAWA	Yutaro	Waseda	JPN	C1
CRIADO	Romain	FISA5	FRA		NAGAMINE	Yuma	Waseda	JPN		TSUDA	Kaoru	Waseda	JPN	
GHERBI	Choayb	FISA5	FRA	A1b	UMEKI	Shusuke	Tokyo	JPN	B1b	CARRERA	Alexis	FISA5	FRA	C1b
VALLIER	Lucillien	FISA5	FRA		PESSÉ	Victor	FISA5	FRA		BEAUTIER	Antoine	FISA5	FRA	
REY	Dimitri	FISA5	FRA		BAZIN	Khalil	FISA5	FRA		LO	Baye Djily	FISA5	FRA	
					DOIX	Paul	FISA5	FRA		FRANCHINI	Antoine-Hugo	FISA5	FRA	
DJERABE	Gauthier	MAM2	FRA	A2	PALLIER	Tristan	MAM2	FRA	B2a	LATRECHE	Zakaria	MAM2	LBN	C2a
WANG	Koichi	Tokyo	CHN		SATAKE	Rito	Waseda	JPN		CHIKUSA	Yuto	Waseda	JPN	
TODA	Ittetsu	Waseda	JPN	A2	KAWAHARA	Yuto	Waseda	JPN	B2	YUZAWA	Koki	Waseda	JPN	C2
SHIOKAWA	Yuma	Waseda	JPN		LEMOS	Hervé	FISA5	JPN		SHIMURA	Yuto	Tokyo	JPN	
GUIA	Nicolas	FISA5	FRA	A2b	BEAUFORT	Nicolas	FISA5	FRA	B2b	ROQUES	Alexandre	FISA5	FRA	C2b
ANANI	Nawel	FISA5	FRA		FONSECA	Lucas	FISA5	FRA		GAILLARD	Adrien	FISA5	FRA	
BERNIER	Théo	FISA5	FRA		SOUSA	Khaoula	FISA5	FRA		HEBERT	Tanguy	FISA5	FRA	
MIRET	Thibault	FISA5	FRA		KHATRI	Hamza	MAM2	FRA		SAUNIER	Jérémie	FISA5	FRA	



Each group uses ESP32C3 +
BME280 sensor

+you own PC



End of my talk

See you in Tutorial



Tutorial instruction



Today

90' Lecture

10:00 - 11:30

T. Fujita, University of Hyogo, JP

IoT modules & its tutorial by using SBC

BXXX

Lunch break

11:30-13:15

This afternoon

Challenge
13:15 - 14:45
Group : C

90' Tutorial - 13:15 - 14:45
H. Sawada
Image processing for
Mechatronics
CXXX - Group : B

90' Practical work
13:15 - 14:45
T. Fujita
IoT modules tutorial using
SBC
CXXX - Group : A

A

Split to 2 small teams
ex.

Group A1 → A1a, A1b
(mixed universities)

Thursday Sept 18

45' Keynote
9:00 - 9:45
H. Sawada
AI

Coffe Break

90' Practical work
10:15 - 11:45
T. Fujita
IoT modules tutorial using
SBC
CXXX - Group : C

90' Tutorial - 10:15 - 11:45
H. Sawada
Image processing for
Mechatronics
CXXX - Group : A

enge: Start
15 - 11:45
oups : B

C

Friday Sept 19

90' Lecture
8:15 - 9:45

Y. Suzuki, University of Tokyo
Design of broadband vibration energy harvesting device
BXXX

Coffe Break

Challenge
10:00 - 11:30
Group : C

90' Practical work
10:00 - 11:30
T. Fujita
IoT modules tutorial using
SBC
CXXX - Group : B

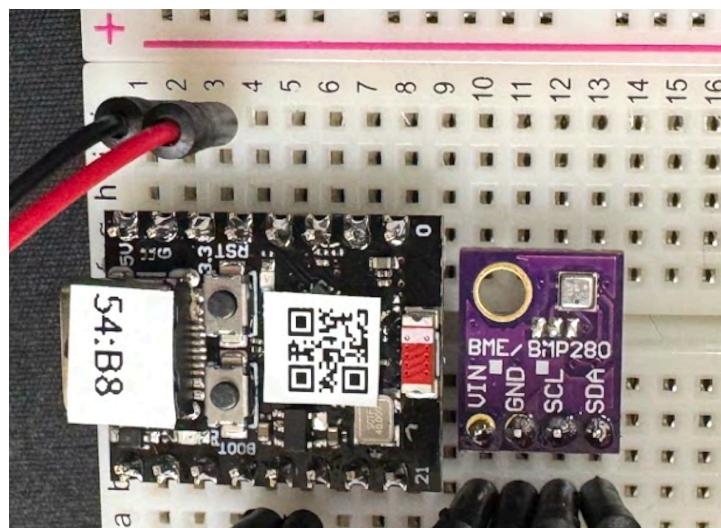
90' Practica
10:00 - 11:
Y. Suzu
Design of bro
vibration ener
device
CXXX - Gro

B

Split to 2 small teams

ex. Team A1 → A1a A1b
(mixed universities)

GROUP A				TEAM	GROUP B				TEAM	GROUP C				TEAM
PHILIPPE	Nicolas	MAM2	FRA	A1	NONNEROUX FATTAZ	Julien	MAM2	FRA	B1	THIAW	Sidy	MAM2	FRA	C1a
ENATSU	Koichi	Tokyo	JPN		RAJERISON	Lorry	FISA5	FRA		SALAMEH	Ahmad	MAM2	FRA	
TAKAGI	Yukihiro	Waseda	JPN	A1	AMANO	Sota	Waseda	JPN	B1	OKAWA	Yutaro	Waseda	JPN	C1
CRIADO	Romain	FISA5	FRA		NAGAMINE	Yuma	Waseda	JPN		TSUDA	Kaoru	Waseda	JPN	
GHERBI	Choayb	FISA5	FRA	A1b	UMEKI	Shusuke	Tokyo	JPN	B1b	CARRERA	Alexis	FISA5	FRA	C1b
VALLIER	Lucillien	FISA5	FRA		PESSÉ	Victor	FISA5	FRA		BEAUTIER	Antoine	FISA5	FRA	
REY	Dimitri	FISA5	FRA		BAZIN	Khalil	FISA5	FRA		LO	Baye Djily	FISA5	FRA	
					DOIX	Paul	FISA5	FRA		FRANCHINI	Antoine-Hugo	FISA5	FRA	
DJERABE	Gauthier	MAM2	FRA	A2	PALLIER	Tristan	MAM2	FRA	B2	LATRECHE	Zakaria	MAM2	LBN	C2a
WANG	Koichi	Tokyo	CHN		SATAKE	Rito	Waseda	JPN		CHIKUSA	Yuto	Waseda	JPN	
TODA	Ittetsu	Waseda	JPN	A2	KAWAHARA	Yuto	Waseda	JPN	B2	YUZAWA	Koki	Waseda	JPN	C2
SHIOKAWA	Yuma	Waseda	JPN		LEMOS	Hervé	FISA5	JPN		SHIMURA	Yuto	Tokyo	JPN	
GUIA	Nicolas	FISA5	FRA	A2b	BEAUFORT	Nicolas	FISA5	FRA	B2b	ROQUES	Alexandre	FISA5	FRA	C2b
ANANI	Nawel	FISA5	FRA		FONSECA	Lucas	FISA5	FRA		GAILLARD	Adrien	FISA5	FRA	
BERNIER	Théo	FISA5	FRA		SOUSA	Khaoula	FISA5	FRA		HEBERT	Tanguy	FISA5	FRA	
MIRET	Thibault	FISA5	FRA		KHATRI	Hamza	MAM2	FRA		SAUNIER	Jérémie	FISA5	FRA	



Each group uses ESP32C3 +
BME280 sensor

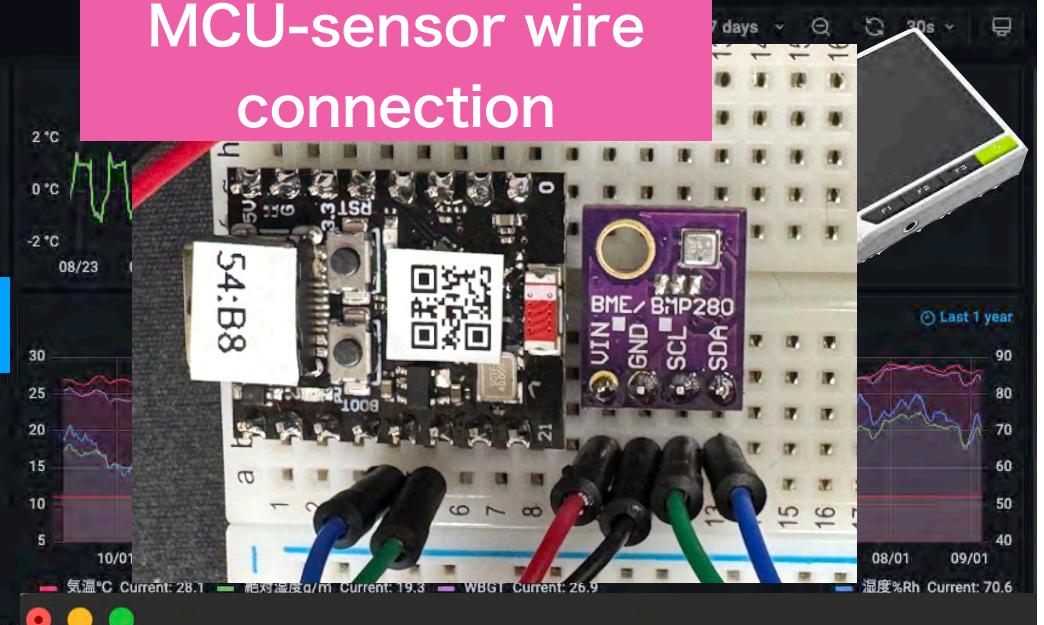
+you own PC



Final goal : Make your own graph



MCU-sensor wire connection



μPython Programming (Copy & paste) on ESP32

```
1 from machine import Pin, SoftI2C, unique_id
2 from time import sleep
3 import BME280
4 import socket
5 import ubinascii
6 import network
7 import time
8
9 # BME280 setup
10 SCL = 22
11 SDA = 21
12 i2c = SoftI2C(scl=Pin(SCL), sda=Pin(SDA))
13 bme = BME280.BME280(i2c)
14
15 # Wi-Fi credentials
16 SSID = "JFWM-WS"
17 PASSWORD = "goodtime"
18
19 print(f"Connecting to Wi-Fi network: {SSID}")
```

Tutorial list

Instruction & sample program

<http://iotwork.org/smb/>

Preparation

0. MicroPython install on ESP32C3 by Thonny IDE

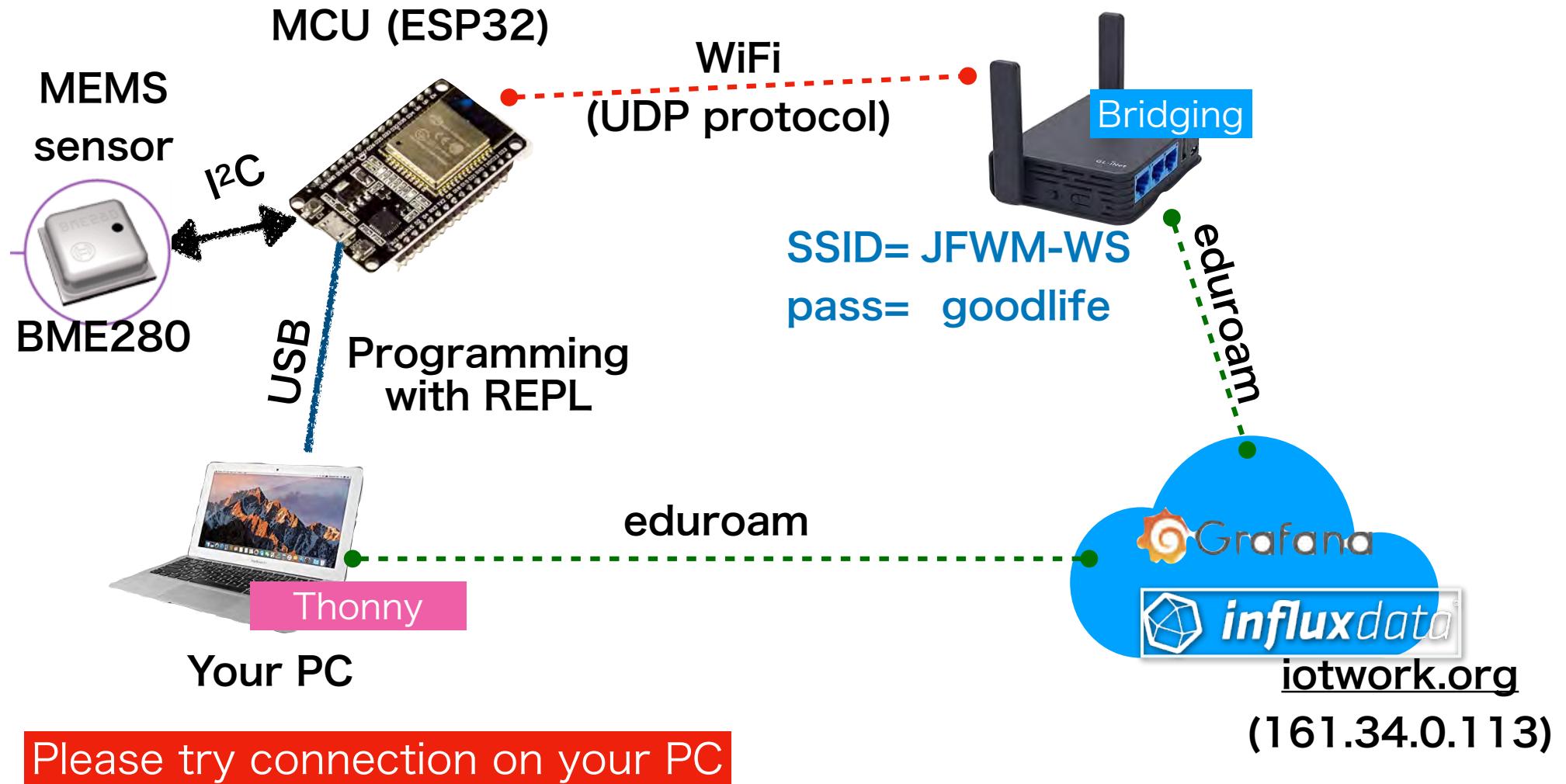
Experiments

1. Blink LED on ESP32
2. Checking the WiFi function of ESP32
3. Wire BME280 to get data
(BME280.py library installation)
4. Upload BME280 data to cloud via UDP from ESP32
~~Revised > Data send by UART to PC > UDP upload~~
5. Visualization on Grafana



(Extra) BLE Beacon RSSI monitoring on Grafana

Overview of connections



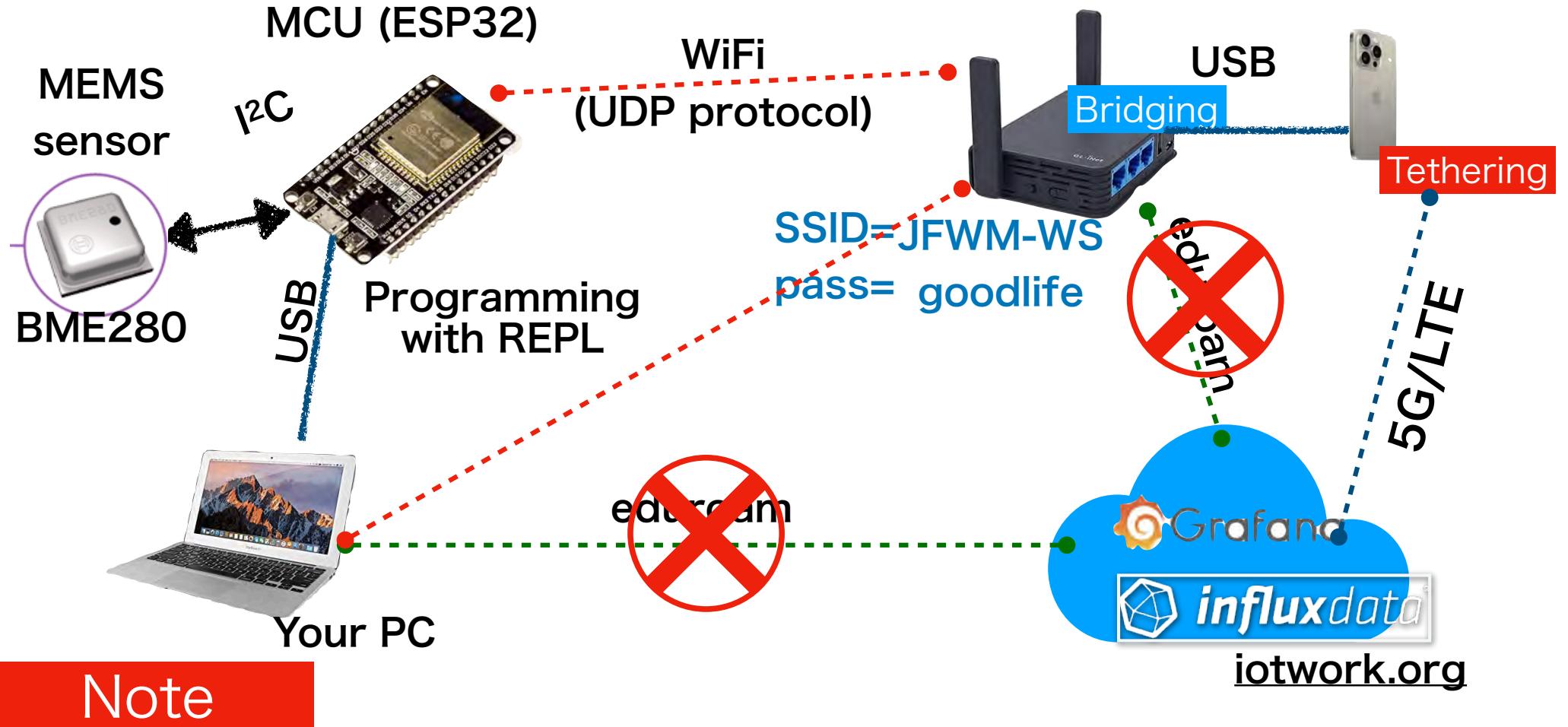
1) Can you to connect <http://iotwork.org/smb/> for instructions ?

Yes

2) Can you to connect <http://iotwork.org:3000> for Grafana ?

Might be No

Overview of connections (revised)



eduroam@USMB's security policy is **restricting** the Grafana communication

Please connect to '**JFWM-WS**' during the tutorial to connect to Grafana.

I provided eSim for 3 days unlimited in France

The screenshot shows the Klook website interface. At the top, there is a search bar with the placeholder "Search destinations or activities" and a magnifying glass icon. To the right of the search bar are language and currency dropdown menus (set to "JPY"), a "Go to app" link, a "Help" link, a "Recently viewed" section, and a "Cart" button with a notification dot. Below the header is a large promotional image for a travel eSIM. The image features a city skyline with modern skyscrapers and a park in the foreground. Overlaid on the image is a purple and white travel eSIM card graphic. The card has "Travel eSIM" at the top, a small globe icon, and the word "FRANCE" at the bottom. It also includes two promotional icons: "Instant Activation" with a smartphone icon and "Best Market Price" with a price tag icon. In the bottom left corner of the main image, there is a small "klook" watermark. Below the main image, the text "Klook Travel" is visible. Underneath the image, there is a product listing for "5G France eSIM | Orange SFR". The listing includes a rating of "★ 4.8 (29 reviews) · 4K+ booked", a "Save to wishlist" button with a heart icon, and a price of "¥ 1,018" in a large orange button labeled "See selected options".

Klook Travel

5G France eSIM | Orange SFR

★ 4.8 (29 reviews) · 4K+ booked

¥ 1,018

Save to wishlist

See selected options

Let's connect ESP32 to PC



Connect ESP32 board
to Windows PC /Mac
w/μ-USB cable



On MacOS Terminal App. just input

ls /dev/cu.*

```
(base) takayukifujita@MacBook-Air ~ % ls /dev/cu.*  
/dev/cu.BN0085-BT  
/dev/cu.Bluetooth-Incoming-Port  
/dev/cu.usbserial-0001
```



Confirm **COM port number** on
Device Manager

Trouble??

Google

cp210x driver windows

Download the driver
first, I will help you.



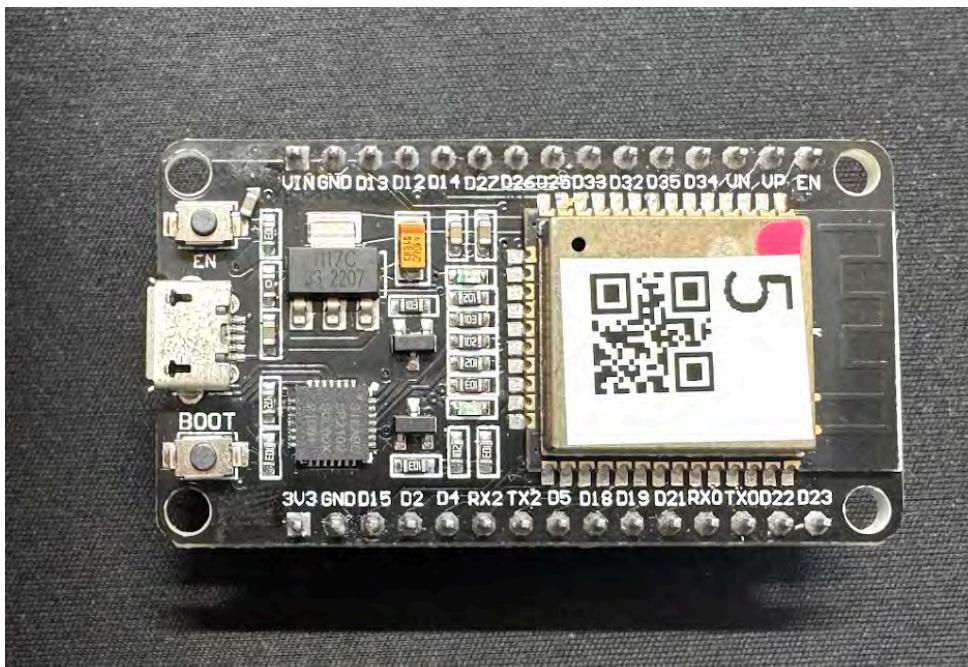
COM3*

column: why start from com3??

System devices
Universal Serial Bus controllers

*depends on PC
environment

ESP32 -> ESP32C3 super mini



C3 super mini in JFWM2025

very cheap and small form factor

1.25 EUROPE/chip

BUT... WiFi connection unstable



AliExpress

140w Pd Module

ESP32-C3 Super Mini WiFi Bluetooth-Compatible Development Board
ESP32 C3 Development Board CORE Board IOT Board for Arduino

5.0 1 Review | 42 sold

Fall Sale - Welcome deal

Starts Sep 15.

€63.58

€12.49 Sale is coming soon

10 pcs, €1.25/pc; Extra 1% off with coins

Color: 10pc Welded

Tutorial list

Instruction & sample program

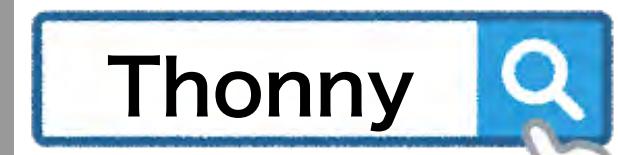
<http://iotwork.org/smb/>

Preparation

0. MicroPython install on ESP32C3 by Thonny IDE

Experiments

1. Blink LED on ESP32
2. Checking the WiFi function of ESP32
3. Wire BME280 to get data
(BME280.py library installation)
4. Upload BME280 data to cloud via UDP from ESP32
~~Revised > Data send by UART to PC > UDP upload~~
5. Visualization on Grafana

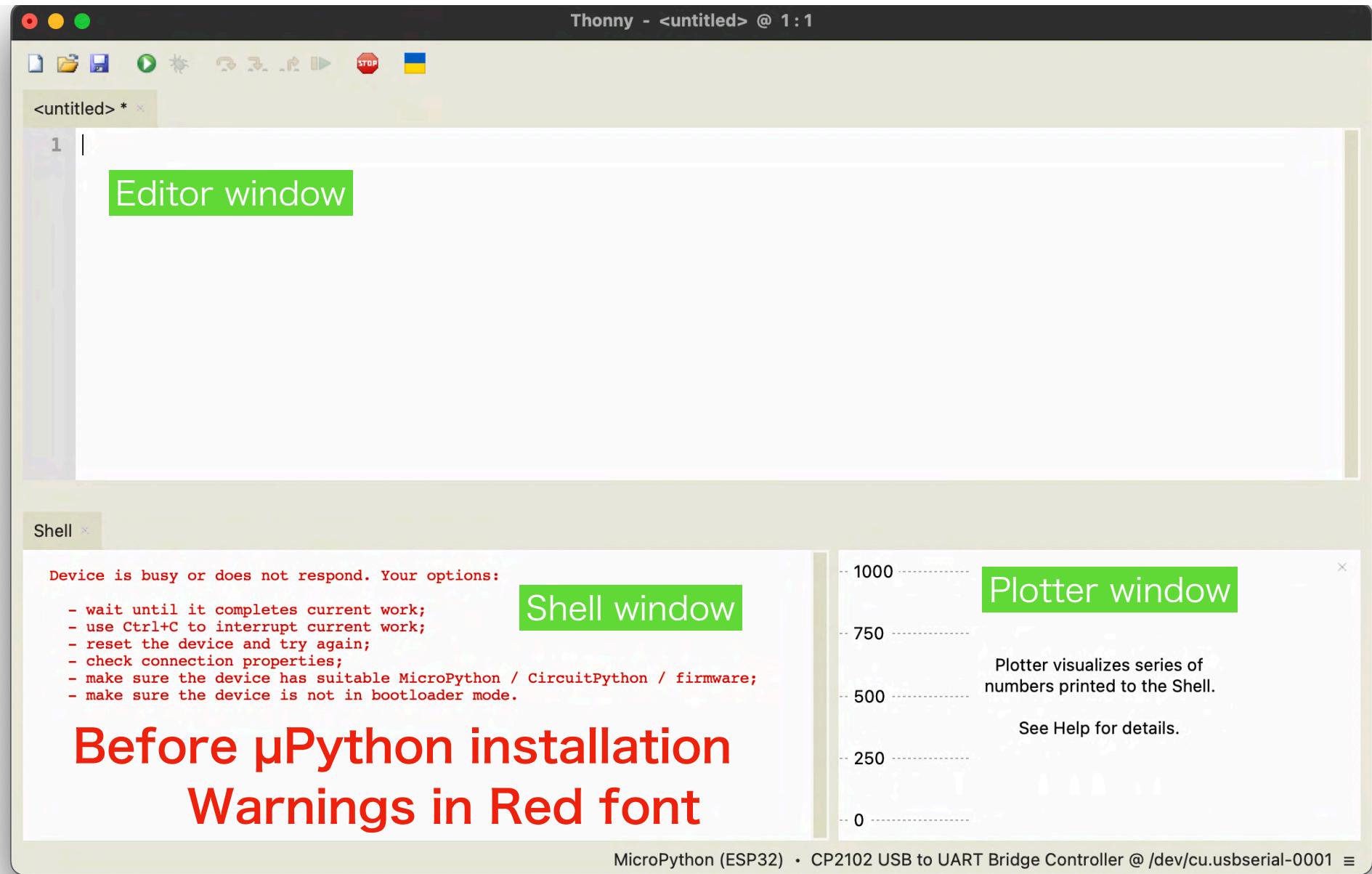


Install and Start

(Extra) BLE Beacon RSSI monitoring on Grafana

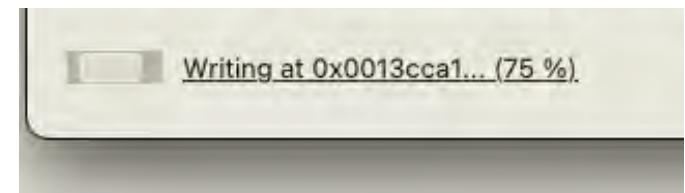
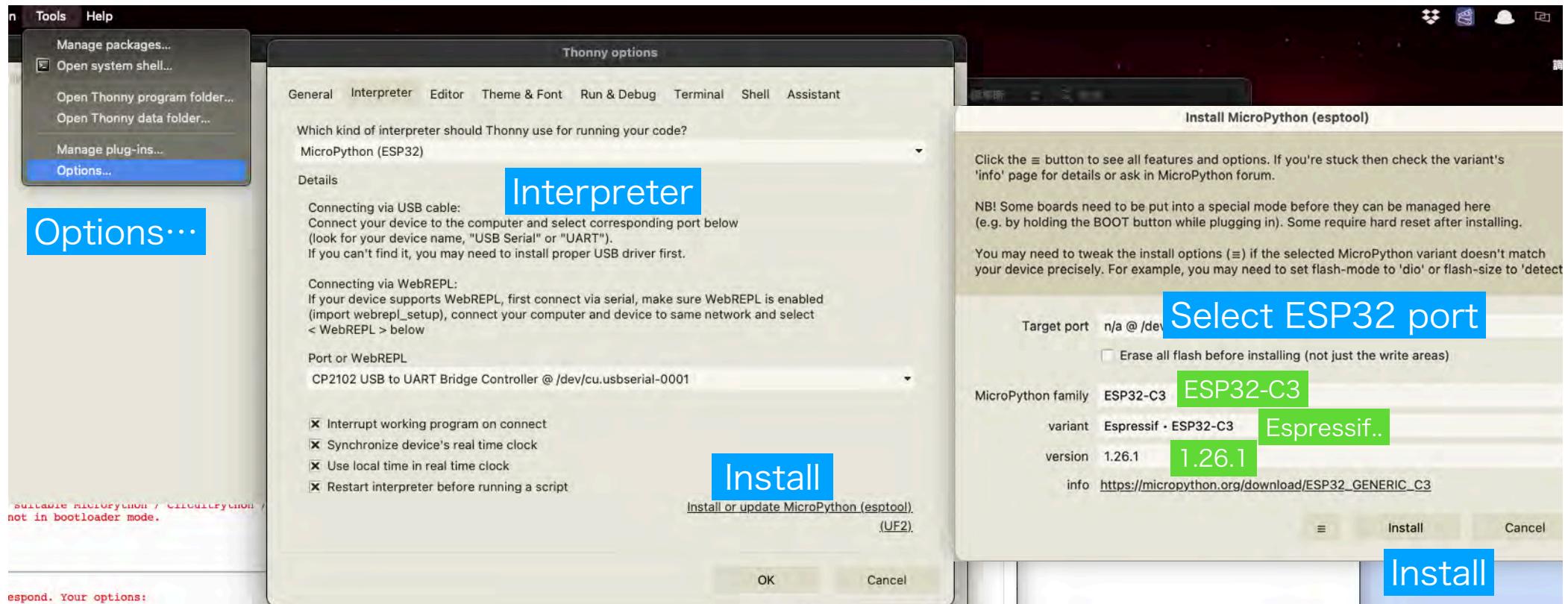
Install μ Python via Thonny app

Sample of Thonny's window



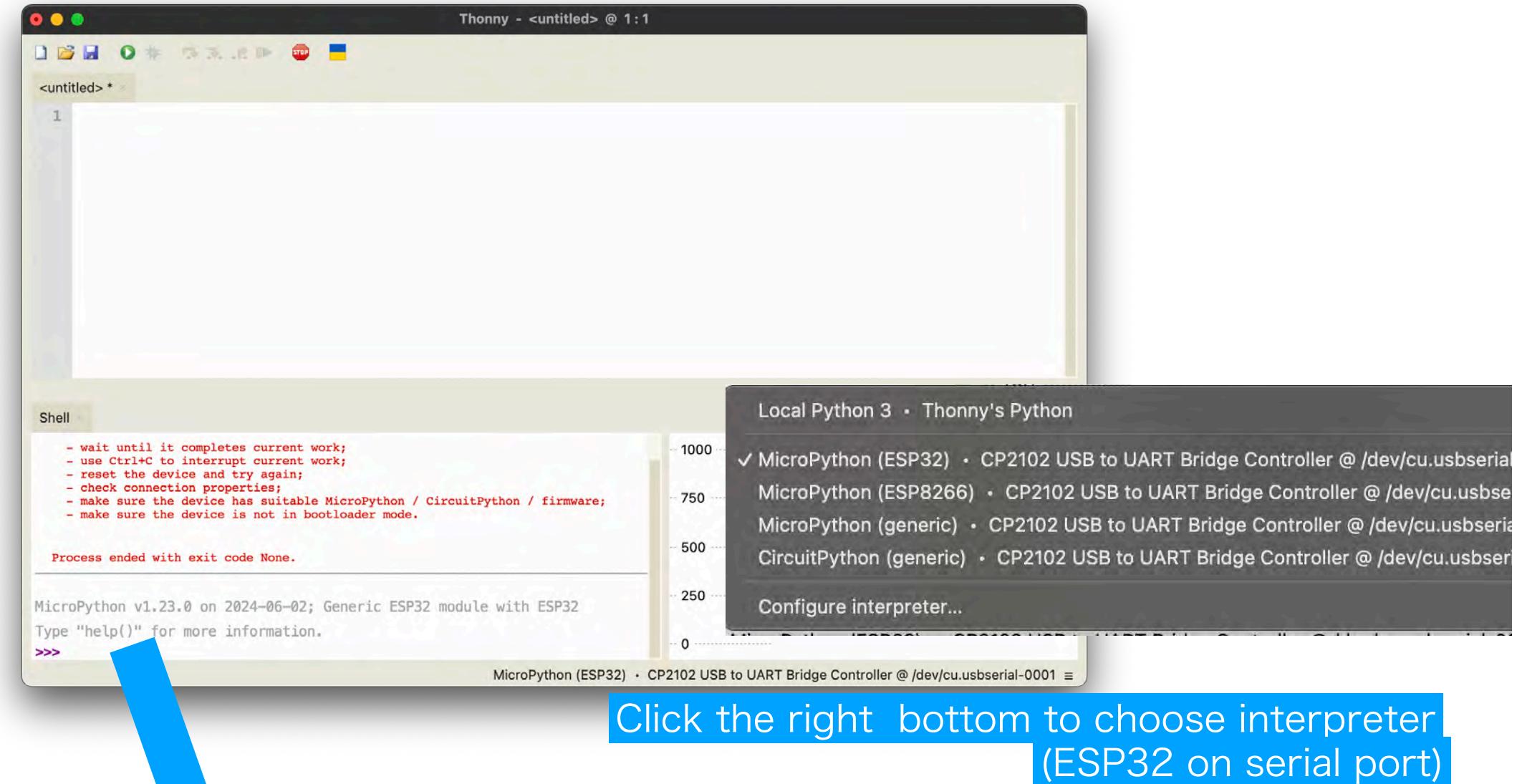
Install μ Python via Thonny app

Download Thonny and install, Run



Wait for writing until “Done”





Now you can use **μPython** on ESP32

MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32
Type "help()" for more information.
=>

Tutorial list

Instruction & sample program

<http://iotwork.org/smb/>

Preparation

0. MicroPython install on ESP32C3 by Thonny IDE

Experiments

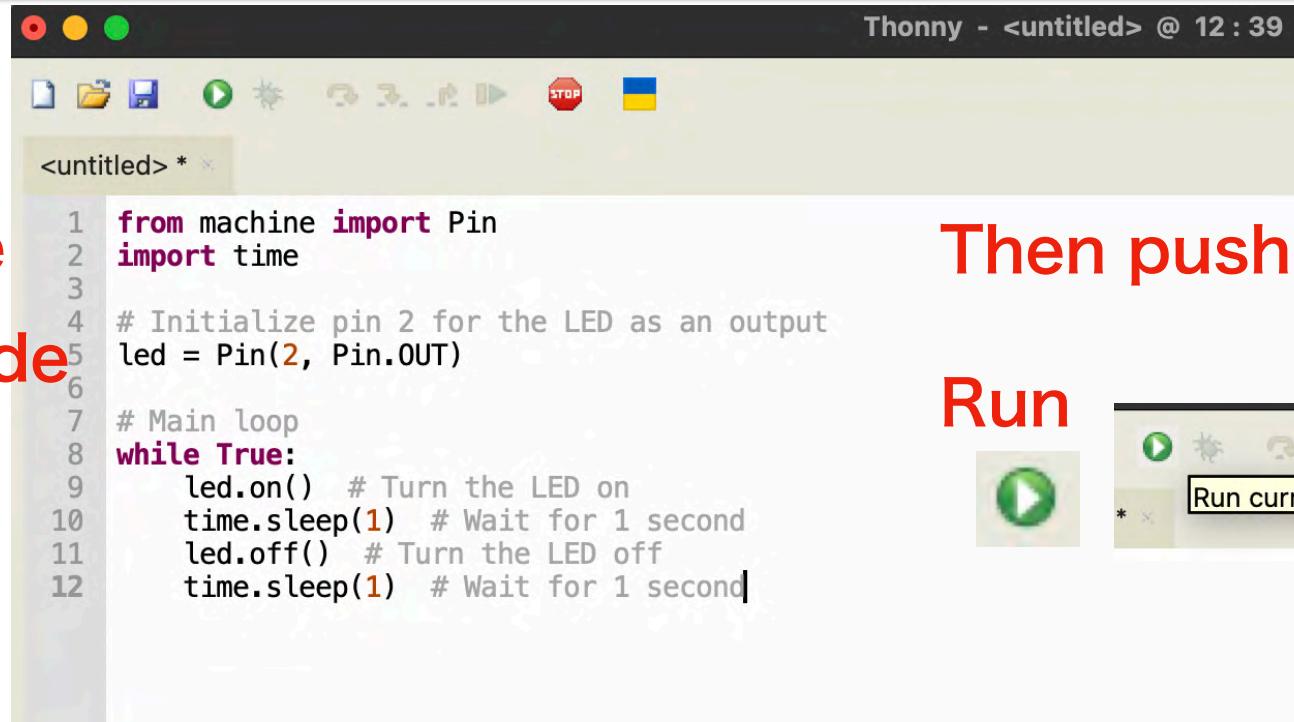
1. Blink LED on ESP32
2. Checking the WiFi function of ESP32
3. Wire BME280 to get data
(BME280.py library installation)
4. Upload BME280 data to cloud via UDP from ESP32
Revised > Data send by UART to PC > UDP upload
5. Visualization on Grafana



(Extra) BLE Beacon RSSI monitoring on Grafana

Ex1) Blink LED

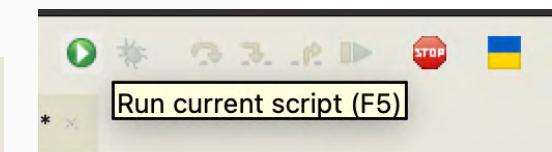
Copy & paste
Python code



The screenshot shows the Thonny IDE interface. The title bar says "Thonny - <untitled> @ 12 : 39". The toolbar includes icons for file operations, run, stop, and a UK flag. The code editor window has a tab labeled "<untitled> *". The code itself is:

```
1 from machine import Pin
2 import time
3
4 # Initialize pin 2 for the LED as an output
5 led = Pin(2, Pin.OUT)
6
7 # Main loop
8 while True:
9     led.on() # Turn the LED on
10    time.sleep(1) # Wait for 1 second
11    led.off() # Turn the LED off
12    time.sleep(1) # Wait for 1 second
```

Then push
Run



Blue LED(pin 2) blinking

Try to change blink intervals

Just change time.sleep and Run



Tutorial list

Instruction & sample program

<http://iotwork.org/smb/>

Preparation

0. MicroPython install on ESP32C3 by Thonny IDE

Experiments

1. Blink LED on ESP32
2. Checking the WiFi function of ESP32
3. Wire BME280 to get data
(BME280.py library installation)
4. Upload BME280 data to cloud via UDP from ESP32
Revised > Data send by UART to PC > UDP upload
5. Visualization on Grafana



(Extra) BLE Beacon RSSI monitoring on Grafana

Ex2) Wi-Fi finder nearby

Copy & paste
Python code

Then push

Run 

```
import network
import time

# Initialize the WiFi interface in station mode
wlan = network.WLAN(network.STA_IF)

# Activate WiFi
wlan.active(True)

print("Nearby WiFi networks:")
for _ in range(5): # Repeat the scan 5 times
    networks = wlan.scan()
    for ssid, bssid, channel, rssi, authmode, hidden in networks:
        print(f"SSID: {ssid.decode()}, RSSI: {rssi} dBm")
    time.sleep(1) # Wait for 1 second
```

Shell

```
SSID: eduroam, RSSI: -72 dBm
SSID: , RSSI: -72 dBm
SSID: eduspot, RSSI: -78 dBm
SSID: eduroam, RSSI: -79 dBm
SSID: , RSSI: -79 dBm
SSID: eduspot, RSSI: -82 dBm
SSID: eduroam, RSSI: -82 dBm
SSID: , RSSI: -82 dBm
SSID: eduroam, RSSI: -84 dBm
SSID: eduspot, RSSI: -86 dBm
SSID: Alice Village by CA, RSSI: -87 dBm
SSID: showroom, RSSI: -89 dBm
SSID: eduspot, RSSI: -89 dBm
SSID: eduroam, RSSI: -89 dBm
SSID: , RSSI: -89 dBm
SSID: eduspot, RSSI: -90 dBm
```

Nearby SSID & RSSI will show



MicroPython (ESP32) • CP2102 USB to UART Bridge Controller @ /dev/cu.usbserial-0001 ≡

Finding SSID “JFWM-WS”

Tutorial list

Instruction & sample program

<http://161.34.0.113/>



Preparation

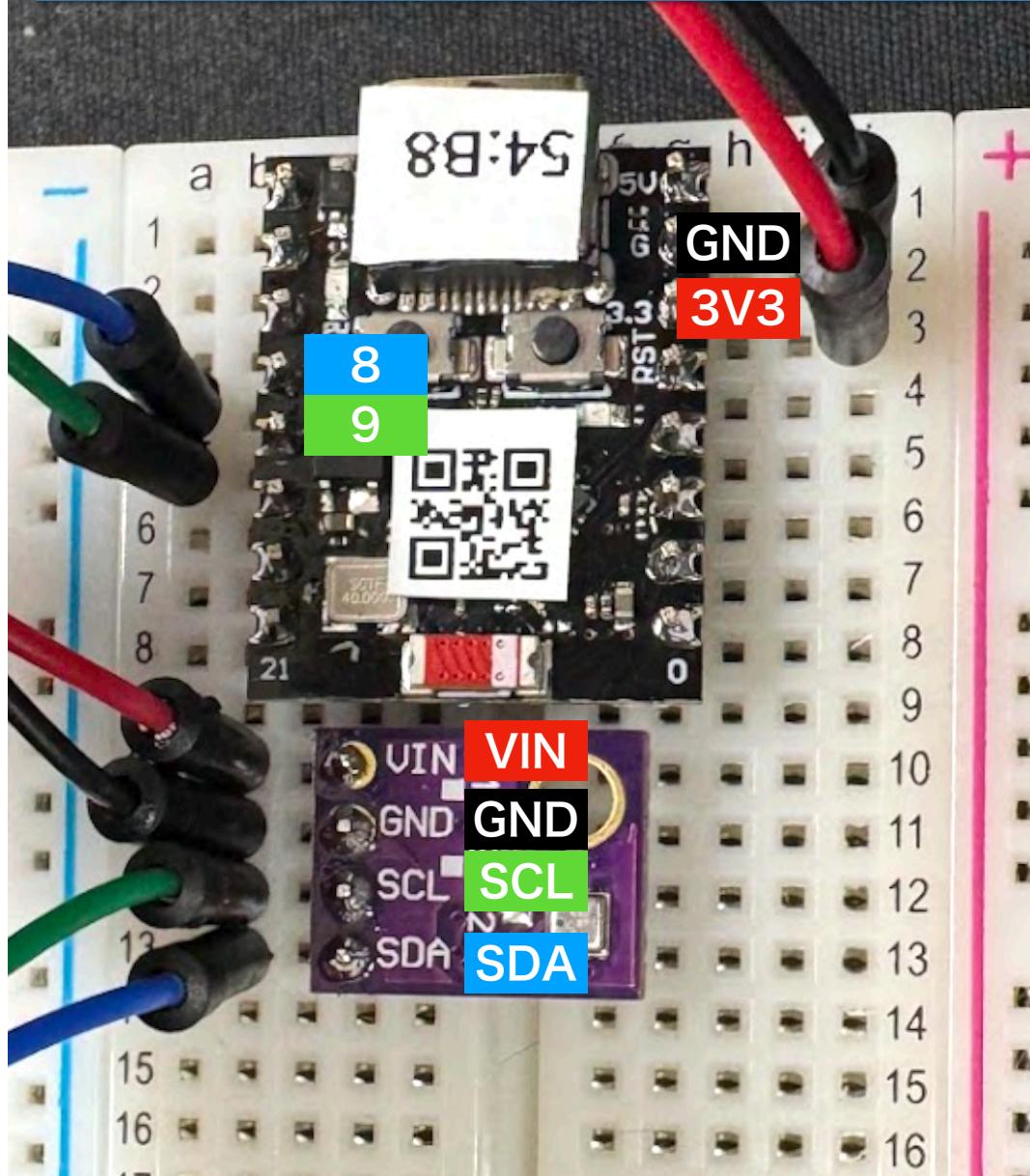
MicroPython install on ESP32 by Thonny IDE

Experiments

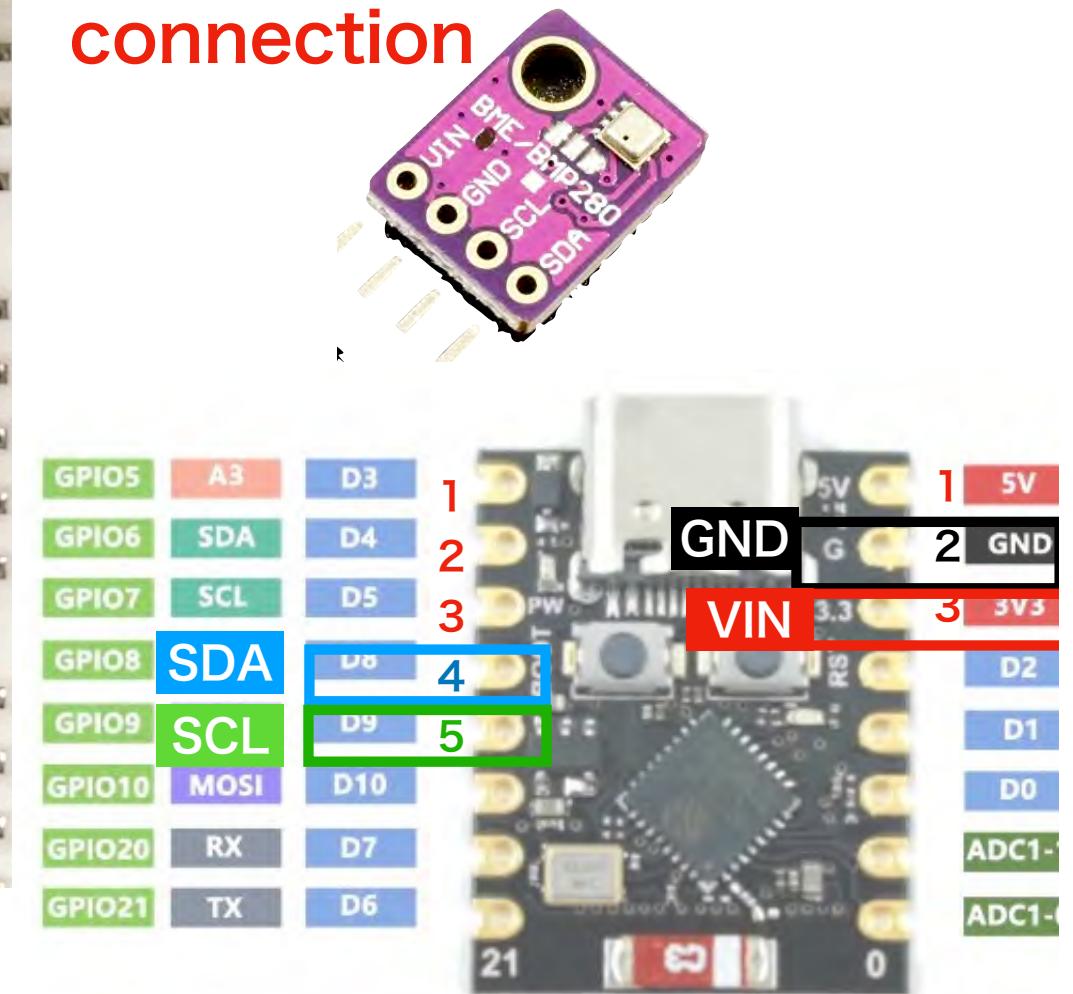
1. Blink LED on ESP32
2. Checking the WiFi function of ESP32
3. Wire BME280 to get data
(BME280.py library installation)
4. Upload BME280 data to cloud via UDP
5. Visualization on Grafana

(Extra) BLE Beacon RSSI monitoring on Grafana

Wire connection on breadboard



4 wires for I2C connection



Ex3) BME280 data get

Copy & paste
Python code

```
Thonny - <untitled> @ 17 : 13

<untitled> *

2 from time import sleep
3 import BME280
4
5 # BME280 setup
6 SCL = 22
7 SDA = 21
8 i2c = SoftI2C(scl=Pin(SCL), sda=Pin(SDA))
9 bme = BME280.BME280(i2c=i2c)
10
11 while True:
12     temp = bme.temperature
13     hum = bme.humidity
14     press = bme.pressure
15     print(f"Temperature: {temp}°C, Humidity: {hum}%, Pressure: {press}hPa")
16
17     sleep(1)

Shell >
>>> %Run -c $EDITOR_CONTENT
MPY: soft reboot
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
ImportError: no module named 'BME280'

>>>
```

You will get error message
no module named 'BME280'

Need to install 'BME280.py' library

MicroPython (ESP32) • CP2102 USB to UART Bridge Controller @ /dev/cu.usb0

Then push

Run



```
iotwork.org/smb/bme280

from machine import I2C
import time

# BME280 default address.
BME280_I2CADDR = 0x76

# Operating Modes
BME280_OSAMPLE_1 = 1
BME280_OSAMPLE_2 = 2
BME280_OSAMPLE_4 = 3
BME280_OSAMPLE_8 = 4
BME280_OSAMPLE_16 = 5

# BME280 Registers
BME280_REGISTER_DIG_T1 = 0x88 # Trimming
BME280_REGISTER_DIG_T2 = 0x8A
BME280_REGISTER_DIG_T3 = 0x8C
BME280_REGISTER_DIG_P1 = 0x8E
BME280_REGISTER_DIG_P2 = 0x90
BME280_REGISTER_DIG_P3 = 0x92
BME280_REGISTER_DIG_P4 = 0x94
BME280_REGISTER_DIG_P5 = 0x96
BME280_REGISTER_DIG_P6 = 0x98
BME280_REGISTER_DIG_P7 = 0x9A
BME280_REGISTER_DIG_P8 = 0x9C
```

<http://iotwork.org/smb/bme280.txt>

Copy on clip-board

Ex3) BME280 data get

1) Create new editor window

2) Copy & paste 'BME280.py'

```
from machine import I2C
import time

# BME280 default address.
BME280_I2CADDR = 0x76

# Operating Modes
BME280_OSAMPLE_1 = 1
BME280_OSAMPLE_2 = 2
BME280_OSAMPLE_4 = 3
BME280_OSAMPLE_8 = 4
BME280_OSAMPLE_16 = 5

# BME280 Registers

BME280_REGISTER_DIG_T1 = 0x88 # Trimming parameter registers
BME280_REGISTER_DIG_T2 = 0x8A
BME280_REGISTER_DIG_T3 = 0x8C

BME280_REGISTER_DIG_P1 = 0x8E
BME280_REGISTER_DIG_P2 = 0x90
BME280_REGISTER_DIG_P3 = 0x92
BME280_REGISTER_DIG_P4 = 0x94
BME280_REGISTER_DIG_P5 = 0x96
BME280_REGISTER_DIG_P6 = 0x98
BME280_REGISTER_DIG_P7 = 0x9A
BME280_REGISTER_DIG_P8 = 0x9C
BME280_REGISTER_DIG_P9 = 0x9E
```

it to ESP32

4) Set file name 'BME280.py'

Save > [OK]

Ex3) Get BME280 data

Press initial tab (Untitled..)

Then push

Run 

```
from time import sleep
import BME280

# BME280 setup
SCL = 22
SDA = 21
i2c = SoftI2C(scl=Pin(SCL), sda=Pin(SDA))
bme = BME280.BME280(i2c=i2c)

while True:
    temp = bme.temperature
    hum = bme.humidity
    press = bme.pressure
    print(f"Temperature: {temp}°C, Humidity: {hum}%, Pressure: {press}hPa")
    sleep(1)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
```

MPY: soft reboot

Temperature: 22.13°C, Humidity: 83.26%, Pressure: 691.48hPa
Temperature: 24.99°C, Humidity: 32.04%, Pressure: 953.16hPa
Temperature: 24.98°C, Humidity: 32.03%, Pressure: 953.11hPa
Temperature: 24.98°C, Humidity: 32.03%, Pressure: 953.19hPa
Temperature: 24.98°C, Humidity: 32.02%, Pressure: 953.19hPa
Temperature: 24.98°C, Humidity: 32.01%, Pressure: 953.14hPa
Temperature: 24.99°C, Humidity: 32.02%, Pressure: 953.14hPa

You will get data from BME280 sensor

-500 Temperature: ● °C, Humidity: ● %, Pressure: ● .14hPa

MicroPython (ESP32) · CP2102 USB to UART Bridge Controller @ /dev/cu.usbserial-0001 ≡

Ex4) BME280 data upload to InfluxDB

Copy & paste

Then push

The screenshot shows the Thonny IDE interface. The title bar says "Thonny - <untitled> @ 42 : 19". The toolbar includes icons for file operations, a terminal, and a "Run" button. The code editor window has tabs for "<untitled> *" and "[BME280.py] *". The code itself is as follows:

```
38 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
39
40 def send_to_influxdb(temp, hum, press):
41     measurement = f"ble_rssi_{mac_address}"
42     data = f"{measurement} temp={temp},hum={hum},press={press}"
43     sock.sendto(data.encode(), (INFLUXDB_IP, INFLUXDB_PORT))
44
45 while True:
46     temp = bme.temperature
47     hum = bme.humidity
48     press = bme.pressure
49     print(f"Temperature: {temp}°C Humidity: {hum}% Pressure: {press} hPa")
50
51     send_to_influxdb(temp, hum, press)
52
53     sleep(1)
```

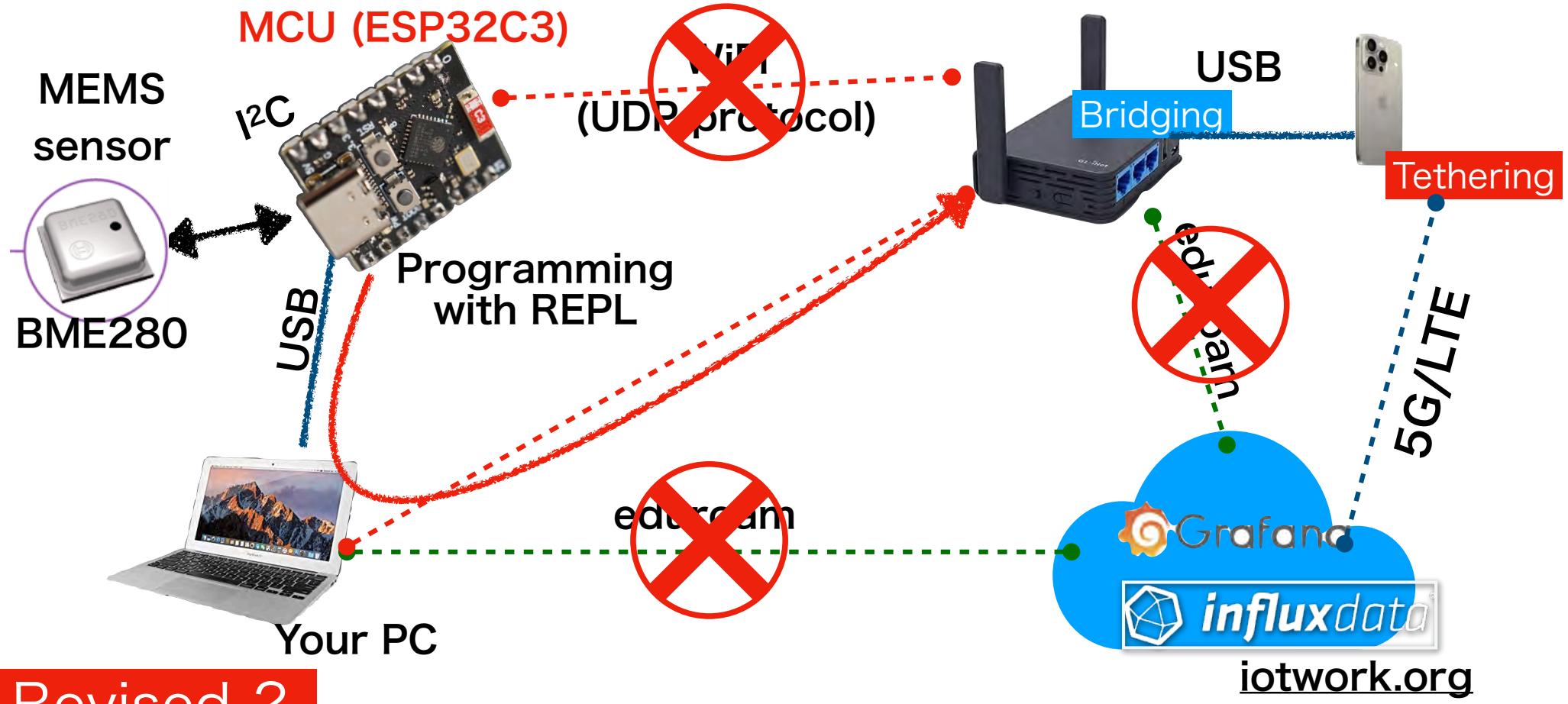
A large red watermark text "Temperature, Humidity, and Pressure" is overlaid on the code editor. Below the code editor, a red box contains the text "Data will be uploaded to InfluxDB on Cloud (iotwork.org)".

The shell window at the bottom left shows repeated text: "Waiting for Wi-Fi connection...".

The main area of the window has a red watermark text "Via Wifi-LTE(Tethering) connection" and "SSID: JFWM-WS, Pass: goodlife".

A red box at the bottom right contains the text "Cannot work because of unstable WiFi".

Overview of connections (revised)



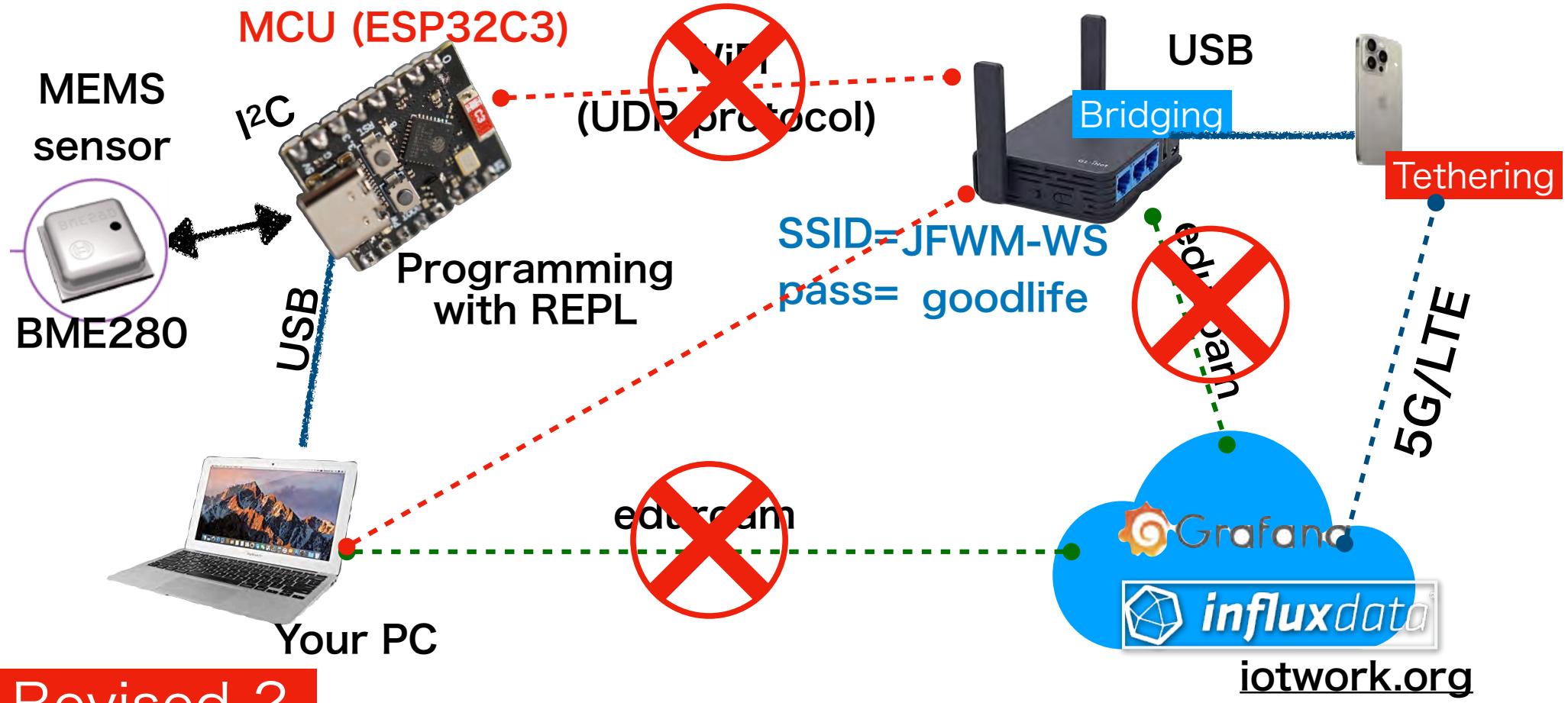
Revised-2

Because of the low quality WiFi of ESP32C3 super mini,
connection unstable

ESP32C3 get data from BME280 > send to PC via USB

ESP32C3 operate with standalone program [main.py]

Overview of connections (revised)



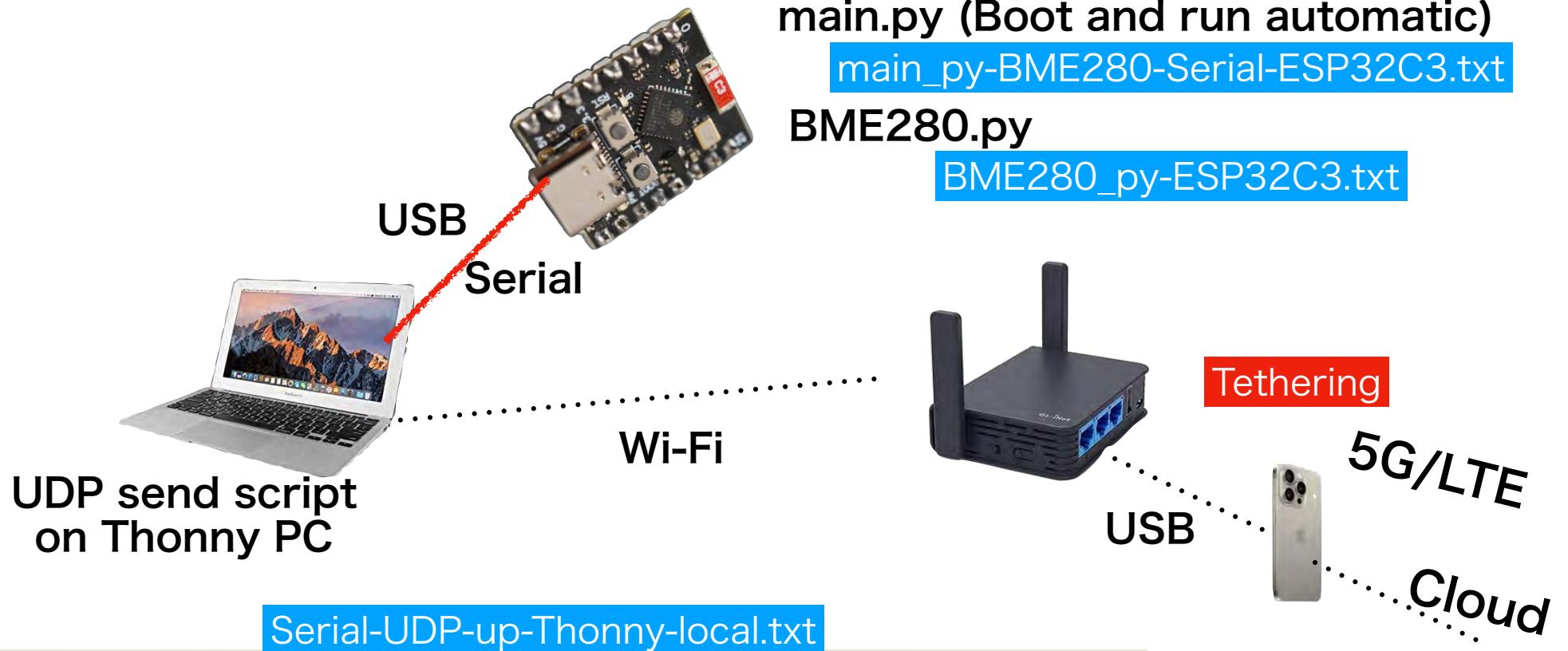
Revised-2

Because of the low quality WiFi of ESP32C3 super mini,
connection unstable

ESP32C3 get data from BME280 > send to PC via USB

ESP32C3 operate with standalone program [main.py]

Two python script saved in ESP32



Serial-UDP-up-Thonny-local.txt

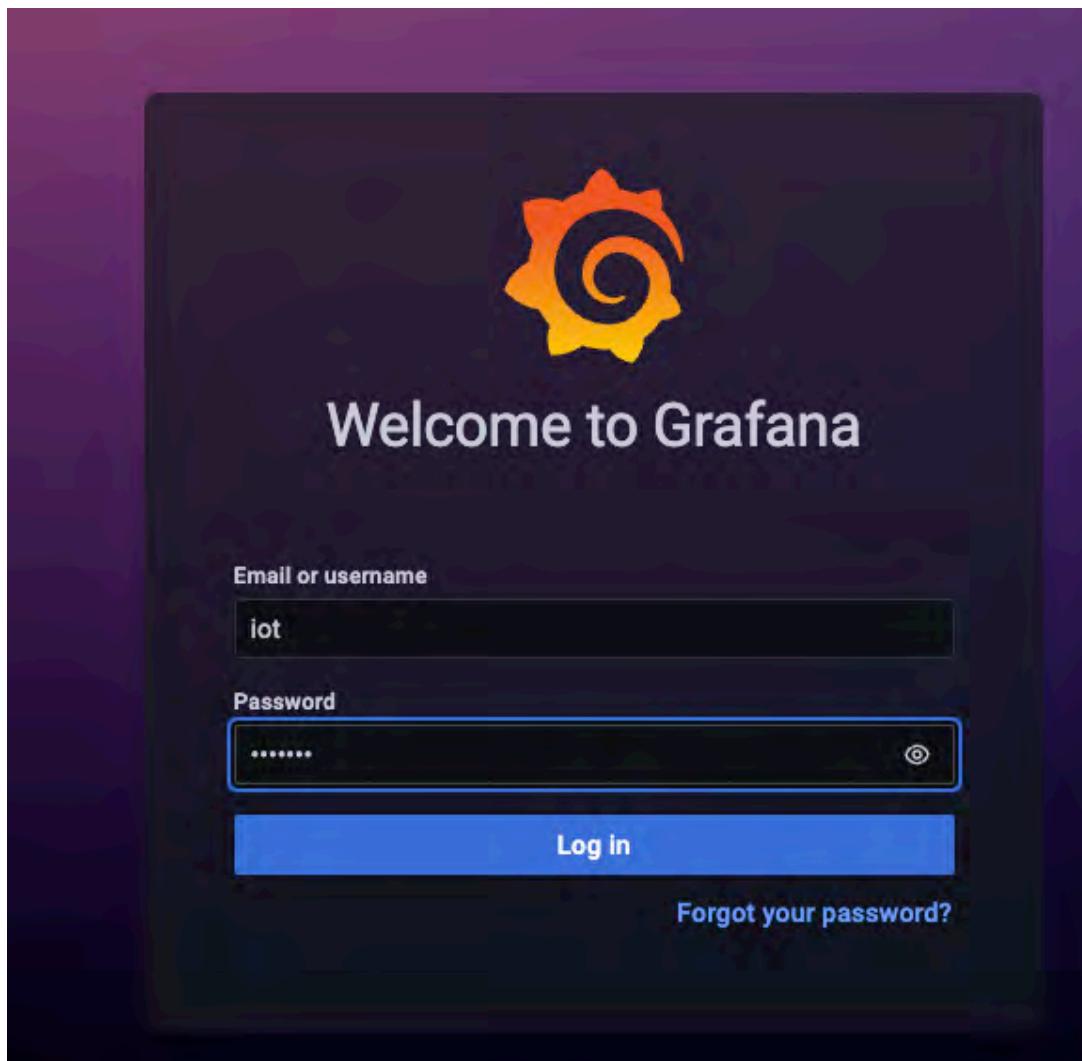
```
[ main.py ] [ BME280.py ] <untitled> * <untitled> *
1 # forward_serial_to_udp_simple.py
2 import socket, sys, time
3 import serial
4
5 # ===== Adjust these settings to your environment =====
6 SERIAL_PORT = "/dev/tty.usbmodem1101"      # Windows → "COM7"
7 #SERIAL_PORT = "/dev/ttyACM0"      # Linux
8 #SERIAL_PORT = "/dev/tty.usbmodem11101" # macOS
9
10 BAUD = 115200
11 UDP_HOST = "iotwork.org"    # InfluxDB server host
```



Data visualization on Grafana

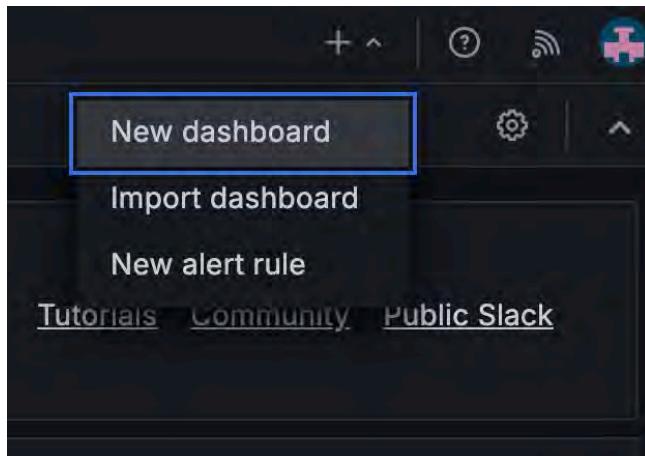
Access grafana server on cloud via AP
(SSID: JFWM-WS, pass: goodlife)

<http://iotwork.org:3000/>

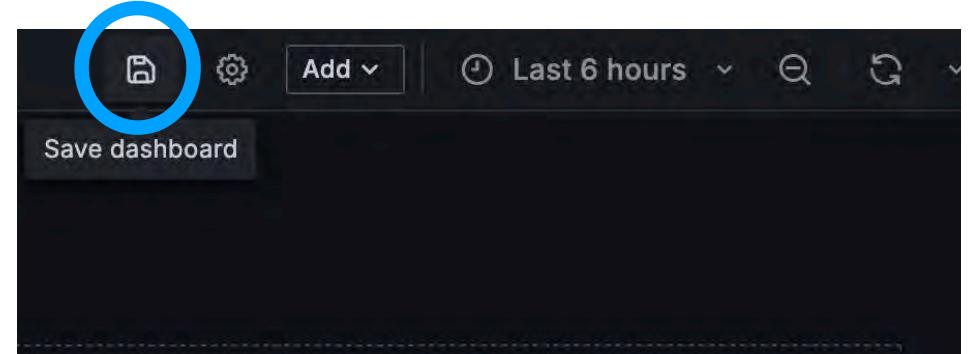


user: **iomt**
pass: **iomt2023**

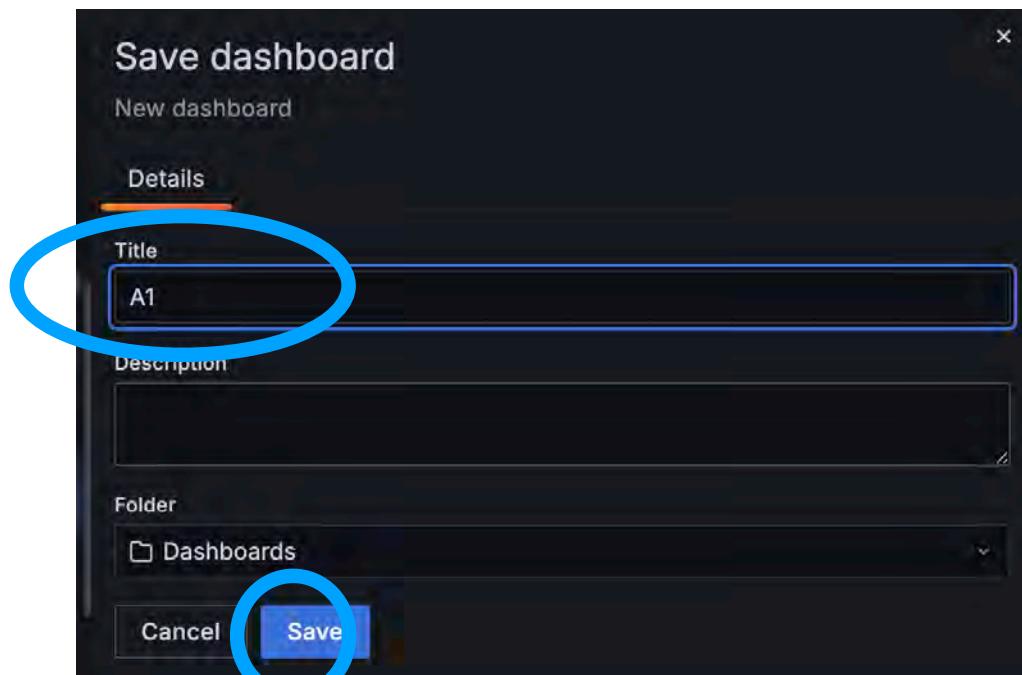
(1) create “Dashboard” from +



(2) Save...

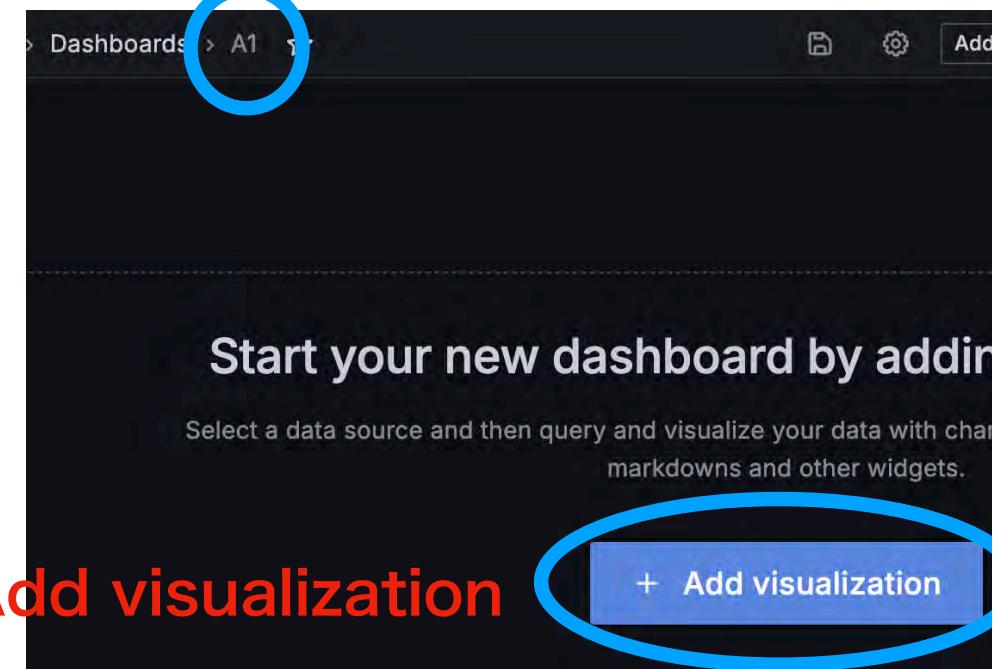


(3) Input title as your group A1a, A1b…C3



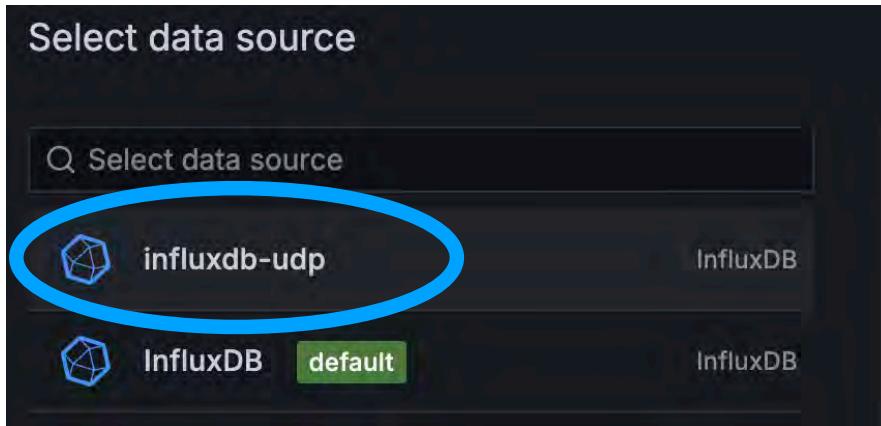
(4) Save

Confirm your group name



(5) Add visualization

(6) Select data source



Data from ~~ESP32~~ Thonny@PC is
stored in influxdb-udp

(7) Press 'influxdb-udp'
> Edit panel appears

The screenshot shows the Grafana dashboard editor. At the top, there's a navigation bar with 'Home', 'Dashboards', 'A1', and 'Edit panel'. Below that are buttons for 'Table view', 'Fill', and 'Actual'. A red warning icon with 'Panel Title' is present. The main area says 'No data'. At the bottom, there's a toolbar with 'Query 1', 'Transform data 0', 'Alert 0', and a 'Data source' dropdown set to 'influxdb-udp'. The 'Query' tab is selected. The query editor shows a query starting with 'FROM default select measurement WHERE' and 'SELECT field(value) mean()'. There are also '+' and '-' buttons for modifying the query.

Query 1

Transform data 0

Alert 0

Data source influxdb-udp

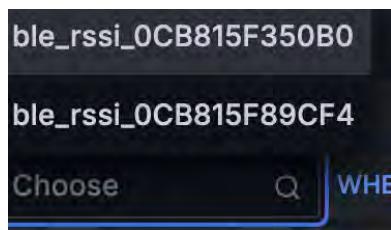
Query ... MD = auto = 814 Interval = 30s Query inspector

A (influxdb-udp)

FROM default select measurement WHERE +

SELECT field(value) × mean() × +

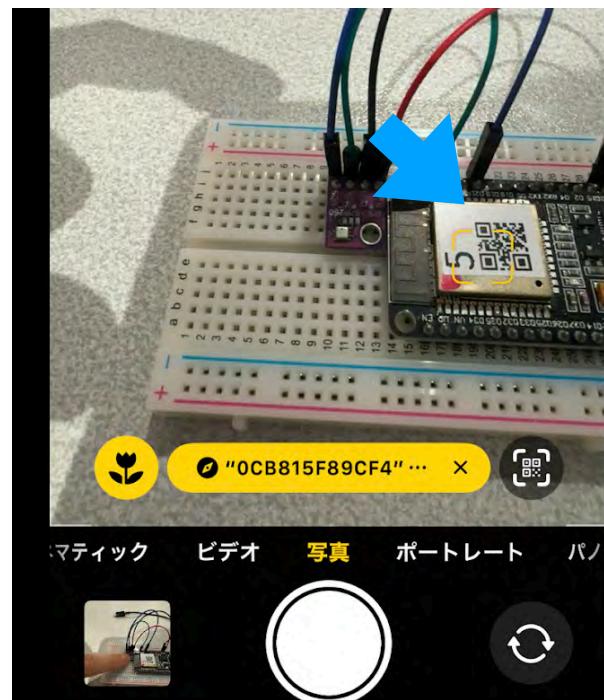
(8) Select measurement



Finding ble_rssi_?????????

????????? is MAC address
of your ESP32

You can get address by
QR code reader



Or labelled number on
ESP32 and address

Select your group's address one

Query 1 Transform data 0 Alert 0

Data source influxdb-udp ? Query ... MD = auto = 814 Interval = 30s Query inspector

A (influxdb-udp)

FROM default select measurement WHERE +

SELECT field(value) × mean() × +

(9) Select field (value)

Transform data 0 Alert 0

default ble_rssi_0CB815F350B0 × W

field Choose X me

time(hum) X +

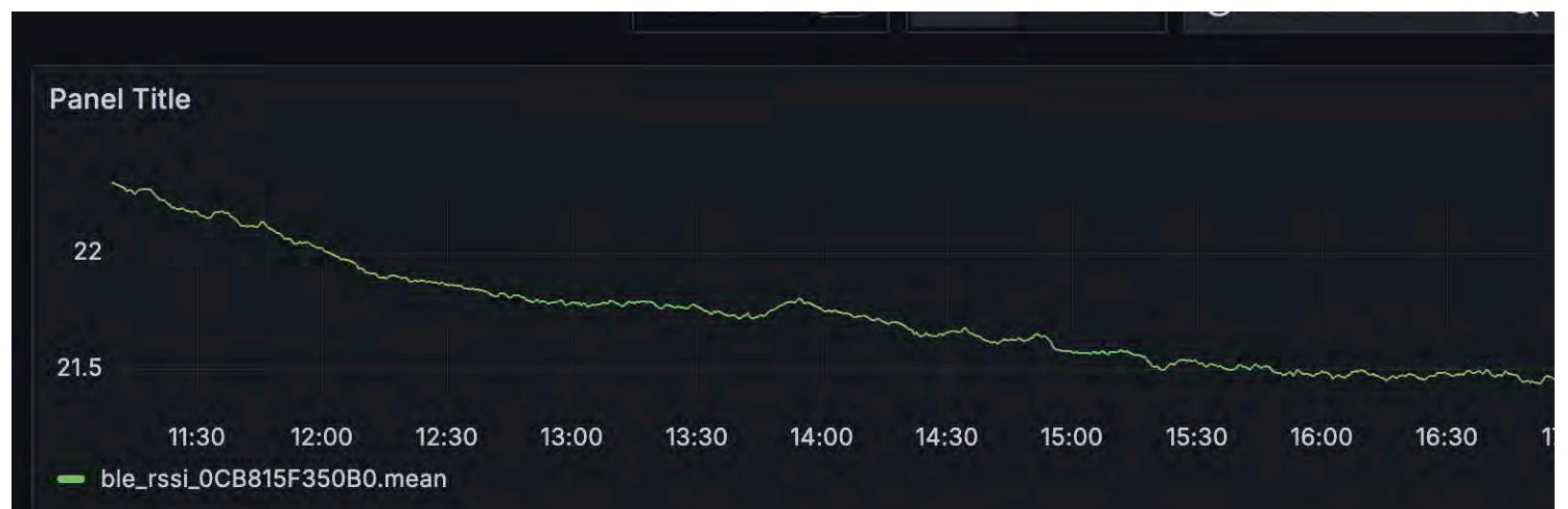
(opti press TIME asc)

(opti temp (optional))

You can choose
hum: Humidity
press: Pressure
temp: Temperature

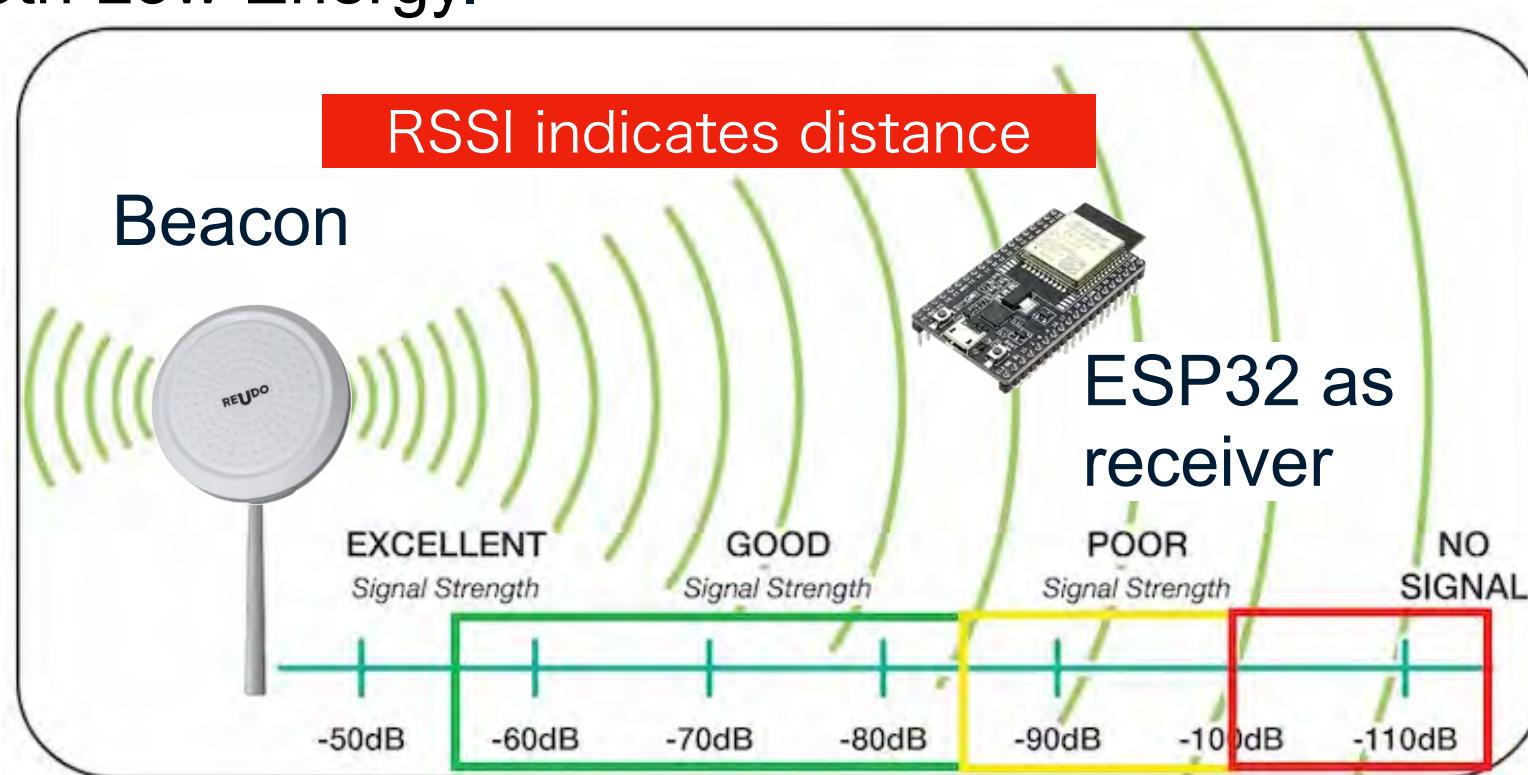
Details will
be explained
on site

(10)
Finally, get a graph
for selection



Opt) Beacon RSSI upload InfluxDB

Bluetooth Low Energy (BLE) beacons are small, battery-powered devices that transmit data using Bluetooth Low Energy.



Before programing
Power on Beacon



Press and hold the button



LED flashes
and SW ON

Opt) Beacon RSSI upload InfluxDB

Copy & paste

Then push

Thonny - <untitled> @ 17:6

<untitled> * [BME280.py] *

```
8 SSID = "Duplex_loft"
9 PASSWORD = "DUPLEX_LOFT"
10
11 # InfluxDB server details
12 INFLUXDB_IP = "161.34.0.113"
13 INFLUXDB_PORT = 8089
14
15 # Target MAC addresses (add or remove as needed)
16 TARGET_MACS = [
17     "ce3f413259a9", # Fujita's Mi-band
18 ]
19 print("Processing target MAC addresses...")
20 target_macs = [mac.replace(':', '').lower() for mac in TARGET_MACS]
21 print(f"Loaded {len(target_macs)} target MAC addresses")
22
23 print("Initializing BLE...")
24 ble = BLE()
25 ble.active(True)
26 print("BLE initialized")
27
28 print(f"Connecting to Wi-Fi network: {SSID}")
29
```

Shell

```
Target device found. MAC: ce3f413259a9, RSSI: -70
Sent to InfluxDB: ble_rssi_0CB815F89CF4 Uplink_RSSI=-51,ce3f413259a9=-70
Target device found. MAC: ce3f413259a9, RSSI: -76
Sent to InfluxDB: ble_rssi_0CB815F89CF4 Uplink_RSSI=-49,ce3f413259a9=-76
Target device found. MAC: ce3f413259a9, RSSI: -71
Sent to InfluxDB: ble_rssi_0CB815F89CF4 Uplink_RSSI=-52,ce3f413259a9=-71
Target device found. MAC: ce3f413259a9, RSSI: -70
Sent to InfluxDB: ble_rssi_0CB815F89CF4 Uplink_RSSI=-51,ce3f413259a9=-70
Target device found. MAC: ce3f413259a9, RSSI: -69
Sent to InfluxDB: ble_rssi_0CB815F89CF4 Uplink_RSSI=-51,ce3f413259a9=-69
Target device found. MAC: ce3f413259a9, RSSI: -71
Sent to InfluxDB: ble_rssi_0CB815F89CF4 Uplink_RSSI=-50,ce3f413259a9=-71
Target device found. MAC: ce3f413259a9, RSSI: -70
Sent to InfluxDB: ble_rssi_0CB815F89CF4 Uplink_RSSI=-52,ce3f413259a9=-70
Target device found. MAC: ce3f413259a9, RSSI: -71
Sent to InfluxDB: ble_rssi_0CB815F89CF4 Uplink_RSSI=-52,ce3f413259a9=-71
```

Run 

Modify the TARGET address to your Beacon address



Read QR code



Visualize on Grafana

MicroPython (ESP32) • CP2102 USB to UART Bridge Controller @ /dev/cu.usbserial-0001 =

RSSI values for specific Beacon address and uplink to AP will be uploaded