STM32 IoT node wifi Lab

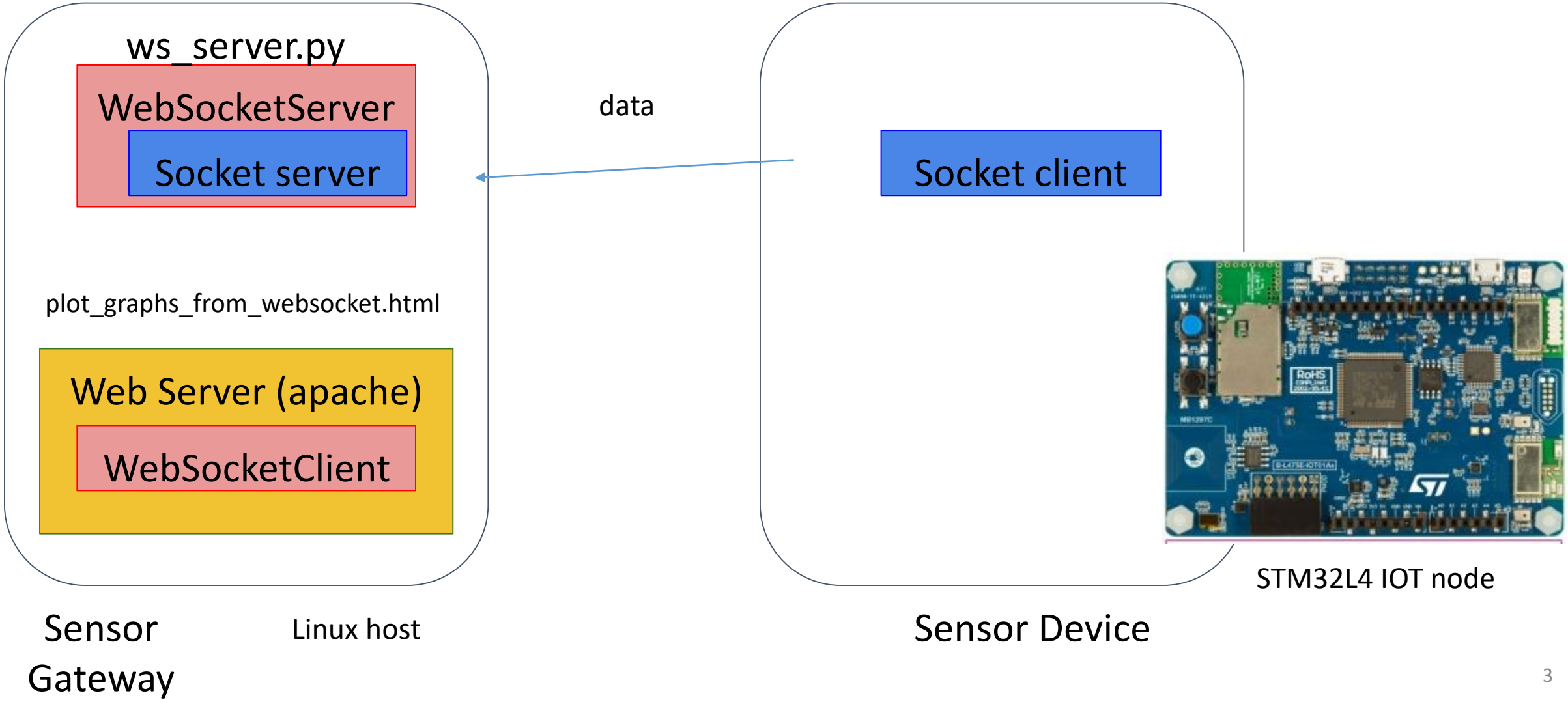
Example: a sensor node with socket client via on-board wifi interface

Experiment overview

- STM32 collects accelerator’s data and sends to a Linux host by wifi
- The Linux host can simply collect the data or
  - The Linux host forward the data to a cloud server – IoT gateway
  - The Linux host analyze and visualize the data
  - The Linux host runs a web server, so other devices(such as smartphone) can access web pages (html) to visualize accelerator’s data on a browser

Sensor gateway and Sensor device

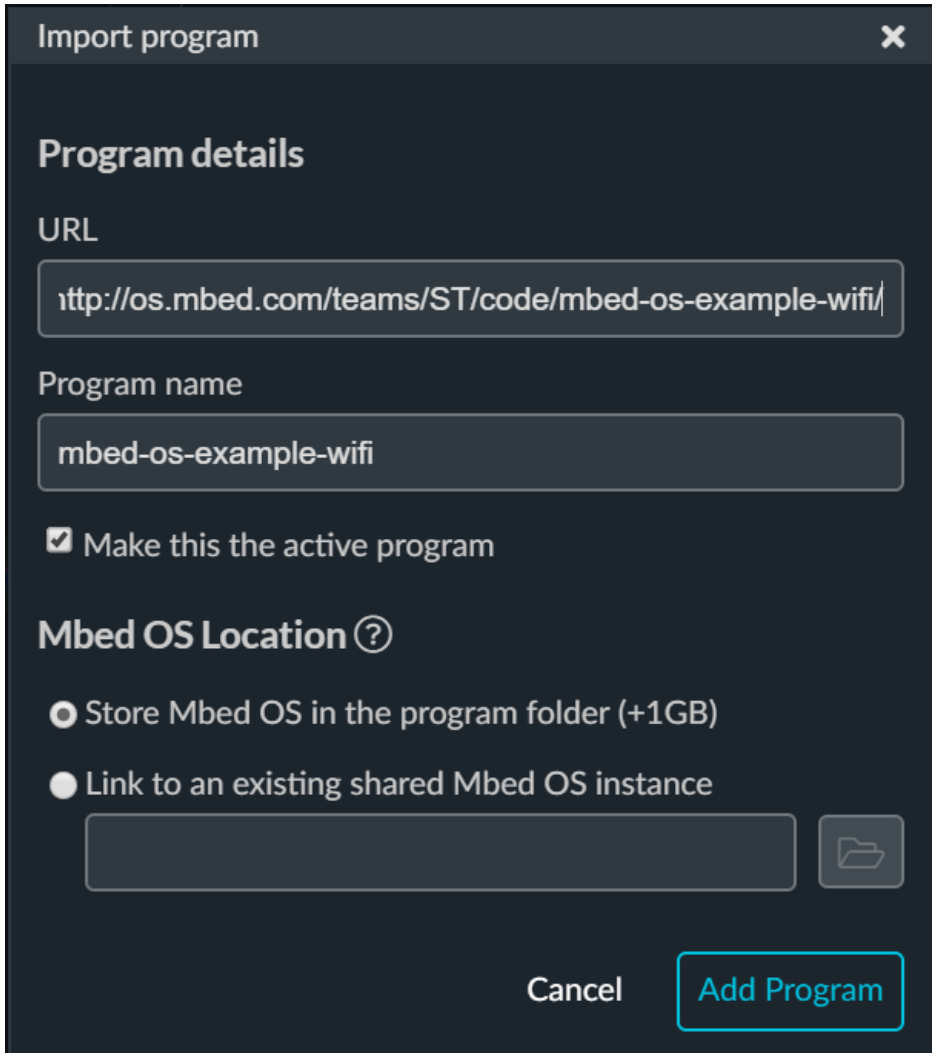
Note the socket server and client



STM32 Mbed Studio Import a Program

- The program brings up the WiFi and the underlying network interface, and uses it to scans available networks, connects to a network, prints interface and connection details and performs simple HTTP operation.
- File → Import Program from URL

  http://os.mbed.com/teams/ST/code/mbed-os-example-wifi/



Mbed OS and components are updatable

Mbed Programs — Mbed Studio

File Edit Selection View Go Terminal Help

Active program

mbed-os-example-wifi-test

Target

DISCO-L475VG-IOT01A (B-L475E-IOT01A)

Build profile

Debug

- mbed_app.json
- README.md
- mbed-os-example-wifi-test
  - mbed-os
  - wifi-ism43362
  - wifi-x-nucleo-idw01m1
  - .hgignore
  - .mbed
  - Jenkinsfile
  - LICENSE
  - main.cpp
  - mbed_app_idw01m1.json
  - mbed_app_idw04a1.json
  - mbed_app_ism43362.json
  - mbed_app.json
  - README.md

Sensors-BSP/... | LF_Clock_config.md | targets.json | main.cpp | README.md .../mbed-os-example-wifi-test

```
# mbed-os-example-wifi #

Wi-Fi example for Mbed OS

## Getting started with the Wi-Fi API ##

This is an example of a Wi-Fi application using the Wi-Fi and network socket APIs that [Mbed

The program brings up the Wi-Fi and the underlying network interface and uses it to scan avai

For more information about Wi-Fi APIs, please visit the [Mbed OS Wi-Fi](https://os.mbed.com/d

### Supported hardware ###

* [NUCLEO-F401RE](https://os.mbed.com/platforms/ST-Nucleo-F401RE/) with [X-NUCLEO-IDW04A1](ht
```

Problems | C:\Windows\SYSTEM32\cmd.exe | Output | Libraries

mbed-os-example-wifi-test
3 libraries

- mbed-os 5.7.5 — Switch to shared
- wifi-ism43362 v1.0~4
- wifi-x-nucleo-idw01m1 release_v1.0.0-rc3~5

mbed-os-example-wifi-test default* 0 0 — Ln 1, Col 1 LF UTF-8 Spaces: 4 Plain Text

check main.cpp in mbed project

- check `main.cpp`, and also
  - Check the OS version difference, modify codes if needed
  - OS API are evolving, now the recent mbed OS is version is 6.9
- `ISM43362Interface class constructor has been changed,` Change the line (line 24)

```
ISM43362Interface wifi(MBED_CONF_APP_WIFI_SPI_MOSI,
MBED_CONF_APP_WIFI_SPI_MISO, MBED_CONF_APP_WIFI_SPI_SCLK,
MBED_CONF_APP_WIFI_SPI_NSS, MBED_CONF_APP_WIFI_RESET,
MBED_CONF_APP_WIFI_DATAREADY, MBED_CONF_APP_WIFI_WAKEUP, false);
```

→

```
ISM43362Interface wifi(false);
```

socket.connect()api needed to chage

- Change the two lines (line 87~88) in `void http_demo(NetworkInterface *net)`

```
socket.open(net);
response = socket.connect("www.arm.com", 80);
```

→

```
// Show the network address
SocketAddress a;
net->get_ip_address(&a);
printf("IP address: %s\n", a.get_ip_address() ? a.get_ip_address() : "None");
printf("Sending HTTP request to www.arm.com...\n");
// Open a socket on the network interface, and create a TCP connection to
//www.arm.com
socket.open(net);
net->gethostbyname("www.arm.com", &a);
a.set_port(80);
response = socket.connect(a);
```

STM32 wifi setting in the mbed prject

- Open `mbed_app.json`, choose wifi module “WIFI_ISM43362” , and key in AP’s SSID and password

```
int ret = wifi.connect(MBED_CONF_APP_WIFI_SSID, MBED_CONF_APP_WIFI_PASSWORD, NSAPI_SECURITY_WPA_WPA2);
```

- Open `mbed_app.json`, choose wifi module “WIFI_ISM43362” , and key in AP’s SSID and password

```
"config": {
    "wifi-shield": {
        "help": "Options are internal, WIFI_IDW0XX1",
        "value": "WIFI_ISM43362"
    },
    "wifi-ssid": {
        "help": "WiFi SSID",
        "value": "\"[redacted]\""
    },
    "wifi-password": {
        "help": "WiFi Password",
        "value": "\"[redacted]\""
    },
    "wifi-tx": {
        "help": "TX pin for serial connection to external devi
        "value": "D1"
```

Delete some unwanted configurations

Mbed Programs — Mbed Studio

File Edit Selection View Go Terminal Help

Active program

mbed-os-example-wifi

Target

DISCO-L475VG-IOT01A (B-L475E-IOT01A)

Build profile

Debug

- mbed_app_ism43362.json M
- mbed_app.json M
- README.md
- mbed-os-example-wifi-test
  - mbed-os
  - wifi-ism43362
  - wifi-x-nucleo-idw01m1
  - .hgignore
  - .mbed
  - Jenkinsfile
  - LICENSE
  - main.cpp
  - mbed_app_idw04a1.json
  - mbed_app_ism43362.json
  - mbed_app.json
  - README.md

ample-wifi × README.md .../mbed-os-example-wifi-test × main.cpp .../mbed-os-example-wifi-test × mbed_app.json ×

```
18  #include "TCPSocket.h"
19
20  #define WIFI_IDW0XX1    2
21
22  #if (defined(TARGET_DISCO_L475VG_IOT01A) || defined(TARGET_DISCO_F413ZH))
23  #include "ISM43362Interface.h"
24  ISM43362Interface wifi(false);
25
26  #else // External WiFi modules
27
28  #if MBED_CONF_APP_WIFI_SHIELD == WIFI_IDW0XX1
29  #include "SpwfSAInterface.h"
30  SpwfSAInterface wifi(MBED_CONF_APP_WIFI_TX, MBED_CONF_APP_WIFI_RX);
31  #endif // MBED_CONF_APP_WIFI_SHIELD == WIFI_IDW0XX1
32
```

Problems × C:\Windows\SYSTEM32\cmd.exe × Output × Libraries ×

mbed-os-example-wifi
2 libraries

- mbed-os 6.9.0
- wifi-ism43362 master

mbed-os-example-wifi default*+ 0 0 Ln 24, Col 31 LF UTF-8 Spaces: 4 C++

Check the `main()`

```
int main()
{
    int count = 0;
    printf("WiFi example\n\n");
    count = scan_demo(&wifi);
    if (count == 0) {
        printf("No WIFI APNs found - can't continue further.\n");
        return -1;
    }
    printf("\nConnecting to %s...\n", MBED_CONF_APP_WIFI_SSID);
    int ret = wifi.connect(MBED_CONF_APP_WIFI_SSID, MBED_CONF_APP_WIFI_PASSWORD, NSAPI_SECURITY_WPA_WPA2);
    if (ret != 0) {
        printf("\nConnection error\n");
        return -1;
    }
```

```
    printf("Success\n\n");
    printf("MAC: %s\n", wifi.get_mac_address());
    printf("IP: %s\n", wifi.get_ip_address());
    printf("Netmask: %s\n", wifi.get_netmask());
    printf("Gateway: %s\n", wifi.get_gateway());
    printf("RSSI: %d\n\n", wifi.get_rssi());

    http_demo(&wifi);

    wifi.disconnect();

    printf("\nDone\n");
}
```

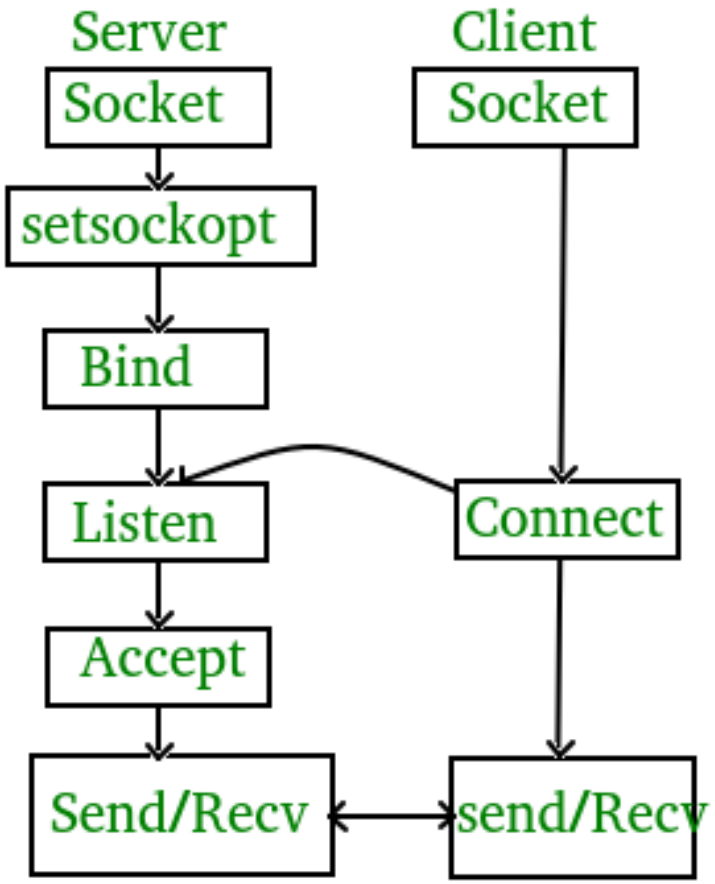
Compile the project and run

- The sensor node shows a successful connecting to a base station, showing its IP address, etc. and sending a HTTP request and get response.

```
Connecting to AspireRad10...
Success

MAC: C4:7F:51:94:4A:4B
IP: 192.168.43.7
Netmask: 255.255.255.0
Gateway: 192.168.43.1
RSSI: -49

Sending HTTP request to www.arm.com...
IP address: 192.168.43.7
Sending HTTP request to www.arm.com...
sent 37 [GET / HTTP/1.1]
recv 64 [HTTP/1.1 301 Moved Permanently]

Done
```

Homework

- Write some codes to read the sensor value, such as 3D Accelerator and 3D gyro, and send to a Linux/Windows hosts and Visualize with some kind of GUI tools (such as using Python, https://mode.com/blog/python-data-visualization-libraries/)



Reference: Add the sensor reading loop in `main()`

```
while (1){
    ++sample_num;
    BSP_ACCELERO_AccGetXYZ(pDataXYZ);
    float x = pDataXYZ[0]*SCALE_MULTIPLIER, y = pDataXYZ[1]*SCALE_MULTIPLIER,
          z = pDataXYZ[2]*SCALE_MULTIPLIER;
    int len = sprintf(acc_json,"{\"x\":%f,\"y\":%f,\"z\":%f,\"s\":%d}",(float)((int)(x*10000))/10000,
                      (float)((int)(y*10000))/10000, (float)((int)(z*10000))/10000, sample_num);

    response = socket.send(acc_json,len);
    if (0 >= response){
      printf("Error seding: %d\n", response);
    }
    wait(0.1);
}
```

Reference: Socket server in Python

```
#!/usr/bin/env python3

import socket

import numpy as np
import json
import time
import random

HOST = 'X.X.X.X'        # Standard loopback interface address
PORT = 65431            # Port to listen on (use ports > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            data = conn.recv(1024).decode('utf-8')
            print('Received from socket server : ', data)
```